ИСП
Институт Системного Программирования
им. В.П. Иванникова
Российской Академии наук

ISSN 2079-8156 (Print)
ISSN 2220-6426 (Online)

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том 30, выпуск 1

Volume 30, issue 1

Москва 2018

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН

Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings/

Содержание

Table of Contents

The effect of partiality and adaptivity on the complexity of FSM state identification problems

[1]*H.Yenigun <yenigun@sabanciuniv.edu>*
[2,3,4]*N. Yevtushenko <nyevtush@gmail.com>*
[5]*N. Kushik <natalia.kushik@telecom-sudparis.eu >*
[5]*J. López <jorge.lopez@telecom-sudparis.eu >*
[1] *Sabanci University, Istanbul, Turkey*
Orta Mahallesi, Sabancı Ünv. No:27, 34956 Tuzla/İstanbul
[2] *Tomsk State University, 634050, Tomsk, 36 Lenin str.*
[3] *Ivannikov Institute for System Programming, RAS,*
109004, Moscow, 25 Solzhenitsyn str.
[4]*National Research University Higher School of Economics (HSE)*
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia
[5]*SAMOVAR, CNRS, Télécom SudParis/Université Paris-Saclay, Evry, France*
9 Rue Charles Fourier, 91000 Évry

Abstract. State identification is a long standing problem in the area of Finite State Machine (FSM) based modeling and testing of discrete event systems. For the identification of the current state of the system, so-called homing and synchronizing experiments with FSMs are used whereas for the initial state identification one can perform a distinguishing experiment. The homing, synchronizing, and distinguishing experiments are known as "gedanken" experiments, and the sequences for these experiments can be derived for deterministic and nondeterministic, partial and complete specification FSMs that are used to formally represent the required behavior of systems under investigation. The problems of checking the existence and derivation of homing, synchronizing, and distinguishing sequences are known to become harder as a specification FSM turns to be nondeterministic and partial. It is also known that in some cases the complexity can be reduced through a 'switch' from preset to adaptive experiment derivation. In this paper, we study how the partiality and adaptivity affect the complexity of checking the existence of homing/synchronizing/distinguishing sequences for deterministic and nondeterministic FSMs and visualize the complexity issues via appropriate figures. We also mention that the existing solutions to state identification problems are widely used for verification and testing of finite state transition systems.

Key words: Finite State Machines (FSMs), state identification problems, complexity.

DOI: 10.15514/ISPRAS-2018-30(1)-1

Для цитирования: Yenigun H., Yevtushenko N., Kushik N., López J. The effect of partiality and adaptivity on the complexity of FSM state identification problems. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 7-24. DOI: 10.15514/ISPRAS-2018-30(1)-1

1. Introduction

The state identification problem using gedanken experiments with Finite State Machines (FSMs) is a long standing problem. The first results were obtained by Moore [1] and have been then improved by many researchers. For the identification of the current state of the system, so-called homing and synchronizing experiments are used whereas for the initial state identification one can perform a distinguishing experiment. The first results on the state identification problem were obtained for complete deterministic FSMs [1-4] while nowadays the homing, synchronizing, and distinguishing sequences are derived for deterministic and nondeterministic, observable and non-observable, partial and complete specification FSMs that are used to formally model the required behavior of systems under investigation. References [1-16] present only a short list of existing papers on this topic.

An *FSM* is a 4-tuple with finite non-empty sets of states, inputs and outputs; it moves to the next state producing an output when an input is applied. An FSM is complete and deterministic if at each state for each input, there is exactly one transition. FSM state identification experiments include homing/synchronizing/distinguishing experiments (and corresponding input sequences), which are known to be either preset or adaptive. A sequence is *adaptive* if the next input to be applied to an IUT is chosen based on the previously observed outputs; otherwise, the sequence is *preset*. *Homing* and *synchronizing* sequences are used for identifying the current state of the machine under experiment while *distinguishing* sequences identify its initial state. The methods for deriving homing/synchronizing/distinguishing sequences are well elaborated for complete and deterministic FSMs. In this case, the length of most such sequences is polynomial with respect to the number of FSM states but it is nearly to impossible to derive a complete specification for modern interactive digital systems due to their complexity. Moreover, current specifications often include various options for output responses under the same input. That is the reason why nowadays nondeterministic and partial FSM models attract a lot of attention [17, 18].

The problems of checking the existence and derivation of homing, synchronizing, and distinguishing sequences are known to become harder as the specification FSM turns to be nondeterministic and partial. It is also known that in some cases the complexity can be reduced through a ‘switch’ from preset to adaptive experiment derivation [6]. Partiality and adaptivity can be considered like two forces working in the opposite directions. Partiality tends to make the problems more complex, and adaptivity tends to make the problem solutions simpler. Correspondingly, it is interesting to study the dynamics here. Why in some cases partiality beats adaptivity, and why in some cases adaptivity beats partiality? Thus, one of the contributions of the paper is to pose this question as a new problem in this area.

In this paper, we also collect the results of how the partiality and adaptivity affect the complexity of checking the existence of homing/synchronizing/distinguishing sequences for deterministic and nondeterministic FSMs. Given a complete deterministic strongly connected reduced FSM, the problem of checking the existence of preset homing and synchronizing sequences is in P [5] while for distinguishing sequences it is PSPACE-complete [6]; the latter means that there exists a complete deterministic FSM such that the length of a shortest distinguishing sequence is exponential with respect to the FSM size. The polynomial complexity is preserved for adaptive homing/synchronizing sequences and the problem of checking the existence of an adaptive distinguishing sequence also 'falls into' P, i.e., in the latter case, the adaptivity reduces the problem complexity. For partial deterministic FSMs, the complexity of checking the existence of an adaptive distinguishing sequence is also in P, i.e., for distinguishing sequences the partiality does not destroy the polynomial complexity. That is not the case for homing and synchronizing sequences, since given a partial deterministic reduced strongly connected FSM, the problem of checking the existence of an adaptive homing or synchronizing sequence is PSPACE-complete [14].

For nondeterministic complete observable FSMs, checking the existence of a preset homing/synchronizing/distinguishing sequence is PSPACE-complete [5, 6, 15] and in this paper, we show that it is the same for partial machines. For nondeterministic complete FSMs the adaptivity reduces the complexity of the problem of checking the existence of a homing/synchronizing sequence as the problem 'falls into' P [8, 13]. For distinguishing sequences, it is proven that there exists a class of FSMs where the length of a shortest adaptive distinguishing sequence is exponential with respect to the number of FSM states [16]. Moreover, in this paper, we strengthen this result by proving the same result for 2-input FSMs. For partial nondeterministic observable FSMs, the problem of checking the existence of an adaptive homing/synchronizing sequence is shown to be PSPACE-hard and in this paper, we show that it is PSPACE-complete. We also show that the problem of checking the existence of an adaptive distinguishing sequence for complete nondeterministic FSMs is out of P. Finally, all the results on the complexity of the existence check of homing/synchronizing/distinguishing sequences for deterministic and nondeterministic, complete and partial FSMs are collected together and the complexity issues are visualized via appropriate figures.

Therefore, the main contributions of the paper are as follows. First, we identify the phenomenon of the dependency between partiality and adaptivity, and their influence on the complexity of "gedanken" experiments for FSMs. Second, we collect and visualize the known results in the area. Third, we close some gaps in the area, in particular, we show that differently from deterministic machines the adaptivity does not help to reduce the complexity of adaptive distinguishing experiments for nondeterministic 2-input FSMs.

The structure of the paper is as follows. Section 2 contains the preliminaries. Section 3 is devoted to exhibit how partiality and adaptivity affect the FSM state

identification problems for deterministic FSMs while nondeterministic FSMs are considered in Section 4. Section 5 concludes the paper.

2. Preliminaries

A *Finite State Machine* (FSM) S is a 4-tuple (S, I, O, h), where S is a finite set of states; I and O are finite non-empty disjoint sets of inputs and outputs; $h \subseteq S \times I \times O \times S$ is a *transition relation*, where a 4-tuple $(s, i, o, s') \in h$ is a *transition*. We consider that the machine S is non-initialized, i.e., it can start working at any state of the set S, unless the opposite is stated explicitly. An FSM S = (S, I, O, h) is *complete* if for each pair $(s, i) \in S \times I$ there exists a pair $(o, s') \in O \times S$ such that $(s, i, o, s') \in h$; otherwise, the machine is *partial*. Given a partial FSM S, an input i is a *defined* input at state s if there exists a pair $(o, s') \in O \times S$ such that $(s, i, o, s') \in h$. In this case, we say that input i can take the machine from state s to state s' and the set of all states where input i can take the machine from state s is the i-successor of state s. An FSM S is *nondeterministic* if for some pair $(s, i) \in S \times I$, there exist at least two transitions $(s, i, o_1, s_1), (s, i, o_2, s_2) \in h$, such that $o_1 \neq o_2$ or $s_1 \neq s_2$. An FSM S is *single-input* if at each state there is at most one defined input, i.e., for each two transitions $(s, i_1, o_1, s_1), (s, i_2, o_2, s_2) \in h$ at state s it holds that $i_1 = i_2$, and S is *output-complete* if for each pair $(s, i) \in S \times I$ such that the input i is defined at state s, there exists a transition from s with i for every output in O. An FSM is *observable* if for each state s and input i it holds that if (s, i, o, s_1'), $(s, i, o, s_2') \in h$ then $s_1' = s_2'$; otherwise, the machine is *non-observable*. In this paper, we consider only observable FSMs.

In usual way, the FSM behavior is extended to sequences of inputs and outputs, i.e., input/output sequences α/β, $\alpha \in I^*$, $\beta \in O^*$. Given a state s and an input sequence $\alpha.i$, the input sequence $\alpha.i$ is a *defined input sequence* at state s if α is a defined input sequence at state s and i is a defined input at each state of the α-successor of s. The set $out(s, \alpha)$ includes all possible output responses for the defined sequence α at state s. A *trace* of S at state s is a sequence of input/output pairs of sequential transitions starting from state s. As usual, for state s and a sequence $\gamma \in (IO)^*$ of input-output pairs, the γ-*successor* of state s is the set of all states that are reached from s by trace γ. If γ is not a trace at state s then the γ-successor of state s is empty or we simply say that the γ-successor of state s does not exist. As usual, the input (output) sequence of γ is the *input* (*output*) *projection* of γ. For an observable FSM S, for any sequence $\gamma \in (IO)^*$, the cardinality of the γ-*successor* of state s is at most one. In this paper, an FSM under experiment is considered to be *strongly connected*, i.e., we assume that for every two states s_1 and s_2 there is a trace that can take the machine from state s_1 to state s_2, i.e., state s_2 is *reachable* from state s_1 via some trace.

If an FSM has the assigned initial state s_0 then it is an *initialized* FSM (S, s_0, I, O, h). An initialized FSM S is *acyclic* if the set of traces at the initial state is finite, i.e., the

FSM transition diagram has no cycles. An initialized, single-input, output-complete, and observable FSM P such that each state is reachable from the initial state, over input alphabet *I* and output alphabet *O* with an acyclic transition graph is a *test case* (over alphabets *I* and *O*). A test case P over alphabets *I* and *O* is a *test case for FSM* S that can be partial and nondeterministic if for each trace $\gamma(io)$ of FSM P at the initial state, it holds that if γ is a trace at a state *s* of S then *i* is a defined input at all states in the γ-successor of *s*. A state *p* of P without any transitions is a *deadlock* state. A trace from the initial state to a deadlock state is a *complete trace* of P. Note that when S is complete, any test case P over input alphabet *I* and output alphabet *O* is a test case for S and if $|I| > 1$ then a test case P is a partial FSM. According to [8], a test case P for S specifies an adaptive experiment with S. The *length* or the *height* of a test case is the length of a longest trace from the initial state to a deadlock state.

A test case P for an FSM S is a *distinguishing test case* (*DTC*) for S if for each complete trace γ of P, the trace γ is a trace of at most one state of the FSM S. A test case P for an FSM S is a *homing test case* (*HTC*) for S if for each complete trace γ of P, the γ-successor of the set *S* has at most one state. An FSM S is *adaptively distinguishing (adaptively homing)* [8] if there exists a DTC (HTC) for S.

Given a possibly nondeterministic observable partial FSM S, an input sequence α is a *distinguishing sequence* if α is a defined input sequence at each state and for every two different states s_1 and s_2, $out(s_1, \alpha)$ and $out(s_2, \alpha)$ do not intersect. An input sequence α is a *homing sequence* if it is a defined input sequence at each state and for each input/output sequence α/β, the non-empty α/β-successors of two initial states coincide. The sequence α is a *synchronizing sequence* if it is a defined input sequence at each state and for every two states s_1 and s_2, the α-successors of s_1 and s_2 are singletons and coincide.

As synchronizing sequences are usually constructed for finite automata, researchers also use the notion of a finite automaton (FA, without empty messages, initial and final states) [19]. The sequence α is a *synchronizing sequence* for an FA [20] if α is a defined sequence of actions at each state of the automaton and for every two states s_1 and s_2, the α-successors of s_1 and s_2 are singletons and coincide. If such a sequence exists, the automaton is called *synchronizing*.

3. How adaptivity and partiality affect the complexity of distinguishing/homing/synchronizing experiments for deterministic FSMs

In this section, we consider possibly partial deterministic FSMs, which are reduced and strongly connected. Given a possibly partial deterministic FSM S, S is *reduced* if for each two different states s_1 and s_2 there exists an input sequence α that is a defined input sequence at both states such that output responses to α at states s_1 and s_2 are different. For other kinds of deterministic FSMs more results on homing/synchronizing sequences can be found in [5, 14].

3.1 Distinguishing experiments

It is known [6] that the problem of checking the existence of a distinguishing sequence for deterministic complete FSMs is PSPACE-complete and the length of such a sequence can be exponential with respect to the number of FSM states. A distinguishing sequence for a possibly partial deterministic FSM can be derived by the following procedure.

Algorithm 1 Deriving a distinguishing sequence for a deterministic possibly partial FSM

Input: Deterministic possibly partial FSM $\mathtt{S} = (S, I, O, h)$

Output: A distinguishing sequence for FSM $\mathtt{S}$ or the reply «There is no distinguishing sequence for the FSM $\mathtt{S}$ »

Step 1. Derive a truncated successor tree for the FSM $\mathtt{S}$. Each node in the tree is labeled by a set of subsets of S of cardinality 1 or 2, i.e., the label of a node is a subset of the set $S^2 = \{ S' \mid S' \subseteq S \text{ and } 1 \leq |S'| \leq 2 \}$. The root of the tree is labeled by the set of all state pairs, i.e., by the set of all pairs $\overline{s_p, s_q}$, $s_p, s_q \in S$, $p < q$. Given a non-leaf node of the tree that is labeled with a set $P \subseteq S^2$, there exists an edge labeled by an input i from this node if and only if i is a defined input at every state of each pair in $\cup P$, where $\cup P$ is the union of the elements of P, and states of any pair of P do not have the same io-successor (for any output $o \in O$). If this is the case, then the edge labeled with i leads to the node labeled with the set $Q = \{ q \mid q$ is the non-empty io-successor of p, $p \in P$, $o \in O \}$. A node at the k^{th} level, $k \geq 0$, labeled with a set $P \subseteq S^2$ is a *leaf* if one of the following conditions holds.

Rule 1: The set P has only singletons.

Rule 2: There exists a node at the j^{th} level, j, $j < k$, labeled with a set R such that P contains each pair of R that is not a singleton.

Rule 3: There does not exist an input defined at every state of each pair in $\cup P$.

Rule 4: For each input i, states of some pair of P have the same io-successor.

Step 2. If all the tree paths are terminated using Rules 2, 3 and 4 then return reply «There is no distinguishing sequence for the FSM $\mathtt{S}$». If there exists a path terminated using Rule 1, then the sequence α labeling this path is a distinguishing sequence for the FSM $\mathtt{S}$.

Due to the above procedure, for each input the set of pairs of states of the given FSM is considered. On the other hand, the problem is known to be PSPACE-complete for deterministic complete FSMs and thus the following statement holds.

Proposition 1. The problem of checking the existence of a distinguishing sequence for a possibly partial deterministic FSM is PSPACE-complete.

In [6], the authors show that the existence check and the derivation of a DTC[1] for a complete deterministic FSM is in P. In [10], Hierons and Türker presented a method

[1] In [10], a DTC is called as an *Adaptive Distinguishing Sequence* (ADS)

to check the existence of a DTC for a partial deterministic FSM by augmenting the partial FSM up to a complete FSM and show that the upper bound on the height of the DTC is polynomial with respect to the number of FSM states. The upper bound on the length of a shortest adaptive distinguishing sequence was improved in [11].

Correspondingly, the conclusion can be drawn that the problem of checking the existence of a DTC for a possibly partial deterministic FSM is in P, i.e., the FSM partiality does not destroy the polynomial complexity for DTCs. To illustrate the effects of adaptivity and partiality on checking the existence of a distinguishing sequence for a deterministic FSM we present these results in Fig. 1. We hereafter notice that when the upper bound of the shortest distinguishing/homing/synchronizing sequence is known to be exponential but the tight upper bound is unknown, we denote this fact by putting $O(2^n)$ for the sequence length.

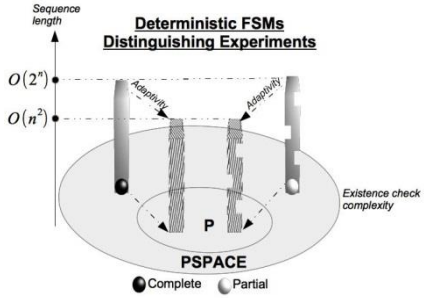


Fig. 1. How partiality and adaptivity affect the complexity of distinguishing experiments for deterministic FSMs

3.2 Homing experiments

It is known [4, 5] that a homing sequence always exists for deterministic complete strongly connected reduced FSMs, and the length of a shortest homing sequence is polynomial, $O(n^2)$, with respect to the number n of FSM states. The upper bound of the length of the homing experiment remains the same for the adaptive case [21]. An algorithm that is very similar to Algorithm 1 can be used when deriving a preset homing sequence for a possibly partial deterministic FSM. The condition "states of any pair of P do not have the same *io*-successor (for any output $o \in O$)" at Step 1 and Rule 4 should be deleted.

Proposition 2. The problem of checking the existence of a homing sequence for possibly partial deterministic FSM is in PSPACE.

In [14], the authors also show the hardness of this problem. In fact, they prove that the problem of checking the non-emptiness of the language of the product of k finite automata can be reduced to the problem of checking the existence of a homing

sequence for partial reduced strongly connected deterministic FSM and thus, the following statement holds.

Proposition 3 [14]. 1) The problem of checking the existence of a homing sequence for deterministic reduced strongly connected partial FSMs is PSPACE-complete. 2) The problem of checking the existence of an adaptive homing sequence for (unreduced and reduced) deterministic strongly connected partial FSMs is PSPACE-complete.

Correspondingly, the conclusion can be drawn that the problem of checking the existence of a homing sequence for possibly partial deterministic FSM is PSPACE-complete even for the class of reduced strongly connected FSMs, i.e., the FSM partiality destroys the polynomial complexity for preset and adaptive homing sequences. We present the impact of adaptivity and partiality on checking the existence of a homing sequence in Fig. 2.

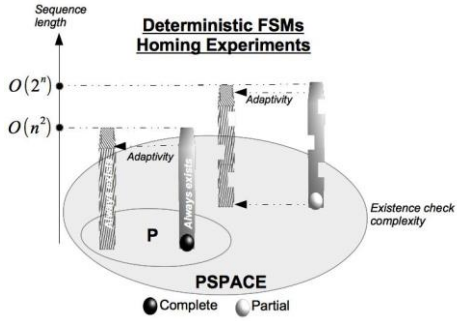


Fig. 2. How partiality and adaptivity affect the complexity of homing experiments for deterministic FSMs

3.3 Synchronizing experiments

Given an FSM, the problem of deriving a synchronizing sequence can be reduced to deriving such a sequence for an automaton that is obtained by erasing transition outputs. Correspondingly, the problem of checking the existence of a synchronizing sequence for complete deterministic FSMs is known to have the polynomial complexity [5]. However, for the automaton that is obtained from a partial FSM by erasing the output action at each transition, the problem is PSPACE-complete [22]. Using a bit modified termination rules in Algorithm 1 for getting a synchronizing sequence one can conclude the problem of checking the existence of a synchronizing sequence for deterministic partial FSMs is PSPACE-complete.

Once a homing sequence is constructed for a deterministic strongly connected possibly partial FSM, an adaptive synchronizing sequence can be constructed similar to that for complete machines [5], i.e., by prolonging a homing sequence

with a corresponding transfer sequence that takes a machine under investigation to a given state *s*. Therefore, the following statement can be established.

Proposition 4. The problem of checking the existence of a synchronizing test case for deterministic strongly connected partial FSMs is PSPACE-complete.

Correspondingly, the conclusion can be drawn that the problem of checking the existence of a synchronizing sequence for possibly partial deterministic FSM is PSPACE-complete even for the class of reduced strongly connected FSMs, i.e., the FSM partiality destroys the polynomial complexity for adaptive synchronizing sequences. The impact of adaptivity and partiality on checking the existence of a synchronizing sequence for a deterministic FSM is presented in Fig. 3.

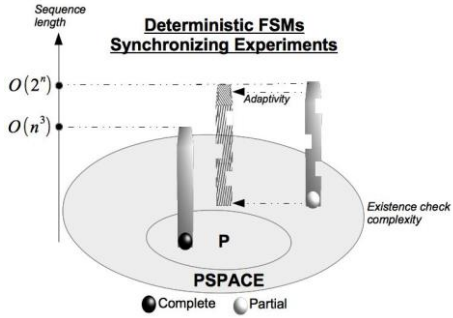


Figure 3: How partiality and adaptivity affect the complexity of synchronizing experiments for deterministic FSMs

4. How adaptivity and partiality affect the complexity of distinguishing/homing/synchronizing experiments for nondeterministic FSMs

In this section, we study how adaptivity and partiality affect the complexity of distinguishing/homing/synchronizing experiments for nondeterministic FSMs.

4.1 Distinguishing experiments

In order to describe the set of all distinguishing sequences for a nondeterministic possibly partial FSM we use the procedure for deriving an appropriate automaton proposed in [23] with slight modifications. For a nondeterministic FSM $\mathsf{S} = (S, I, O, h)$, $S = \{s_1, s_2, \ldots, s_n\}$, we derive an automaton S^2_{dist} such that the set of (all) synchronizing sequences of this automaton coincides with the set of (all) distinguishing sequences of FSM S, i.e., $L_{dist}(\mathsf{S}) = L_{synch}(\mathsf{S}^2_{dist})$.

Algorithm 2 for deriving the automaton S^2_{dist}

Input: Possibly partial observable FSM $\mathsf{S} = (S, I, O, h)$

Output: The automaton S^2_{dist}

States of S^2_{dist} are pairs (s_j, s_k), $j < k$, and the designated state *sink* while actions are inputs of the FSM S;

For each input $i \in I$
  For each state (s_j, s_k) of the automaton S^2_{dist}
    Add to the automaton S^2_{dist} the transition $((s_j, s_k), i, sink)$ if states s_j and s_k are separated by i, i.e., $out(s_j, i) \cap out(s_k, i) = \varnothing$;
    Add to the automaton S^2_{dist} the transition $((s_j, s_k), i, (s_p, s_t))$, $p < t$ and $j < k$, if for each $o \in O$, the io-successors of states s_j and s_k do not coincide and $\{s_p, s_t\}$ is the io-successor of the set $\{s_j, s_k\}$ for some $o' \in O$;
  EndFor
  Add to the automaton S^2_{dist} the transition $(sink, i, sink)$ for each input $i \in I$;
EndFor

Based on the construction of S^2_{dist}, similar to [23], the following result can be established.

Proposition 5. An input sequence α is a distinguishing sequence for the FSM S if and only if α is a synchronizing sequence for S^2_{dist}.

This result means that the set of all distinguishing sequences of the possibly partial and nondeterministic FSM S coincides with the set of all synchronizing sequences of the automaton S^2_{dist}, i.e., $L_{dist}(S) = L_{synch}(S^2_{dist})$.

Given a partial observable nondeterministic FSM S, the automaton S^2_{dist} derived by Algorithm 2 can be nondeterministic and partial, since for some pair (s_j, s_k), $j < k$, of states of FSM S, there can be no transition to different pairs under an input i if states s_j and s_k have the same non-empty io-successor for some output o or the input i is an undefined input for some state of the pair. Therefore, taking into account the complexity of the existence check for the synchronizing experiments and the fact that the problem of deriving a distinguishing sequence for complete deterministic FSMs is PSPACE-complete, we conclude the following.

Proposition 6. The problem of checking the existence of a distinguishing sequence is PSPACE-complete for observable complete and partial nondeterministic FSMs.

The length of a shortest distinguishing sequence for complete observable machines with n states is known to be $O(2^{n^2})$ and cannot be more for partial observable nondeterministic FSMs according to Algorithm 1 that can be applied for nondeterministic observable FSMs.

We now show that the problem of checking the existence of an adaptive distinguishing test case for a complete observable FSM is PSPACE-hard. In order to prove this, we will show that for any integer n, one can construct a 2-input FSM S such that the size of FSM S is polynomial in n, but the minimal length of an adaptive test case for a subset of n states of S is exponential in n. In fact, this strengthens the previous result [16] where the exponential height of a DTC was proven for an FSM with the exponential number of inputs.

Let $n \geq 2$ be an integer and $p_1, p_2, \ldots, p_n$ be the first n different primes considered in increasing order. Furthermore, let $\Sigma_n = p_1 + p_2 + \ldots + p_n$ be the sum of the first n primes, and let $\Pi_n = p_1 \times p_2 \times \ldots \times p_n$ be the product of the first n primes. For $n \geq 2$,

we show that there exists an FSM $\mathbf{S}_n$ with Σ_n states such that the minimal length of an adaptive test case for a subset with n states equals Π_n. Note that Σ_n is polynomial in n and Π_n is exponential in n. The state set of the FSM is $S = \{1, 2, \ldots, \Sigma_n\}$. We consider the set of states partitioned into n subsets $S_1, S_2, \ldots, S_n$, where $S_j = \{ \Sigma_j - p_j + 1, \Sigma_j - p_j + 2, \ldots, \Sigma_j\}$, for $1 \le j \le n$. An FSM has two inputs i_1 and i_2 and the set of outputs is $\{0, \Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$. The transitions under i_1 constitute a cycle of length p_j for the states in S_j, for $1 \le j \le n$, with the same output 0. Formally, for a state $k \in S_j$, for $1 \le j \le n$, we have the transition $(k, i_1, 0, K)$ where $K = k + 1$ if $k < \Sigma_j$ and $K = \Sigma_j - p_j + 1$ if $k = \Sigma_j$. For a state Σ_j, for $1 \le j \le n$, we have the transition $(\Sigma_j, i_2, \Sigma_j, \Sigma_j)$. Finally, for a state $k \in S_j \backslash \{\Sigma_j\}$, for $1 \le j \le n$, we have the transitions $(k, i_2, \Sigma_1, \Sigma_1)$, $(k, i_2, \Sigma_2, \Sigma_2)$, …, $(k, i_2, \Sigma_n, \Sigma_n)$. Transitions under i_2 distinguish only states of the subset $b = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$. An example of such an FSM for $n = 3$ is shown below inTab. 1. The number of states is $\Sigma_3 = 2 + 3 + 5 = 10$.

Table 1. FSM $\mathbf{S}_n$ for n = 3

	S_1		S_2			S_3				
	1	2	3	4	5	6	7	8	9	10
i_1	2/0	1/0	4/0	5/0	3/0	7/0	8/0	9/0	10/0	6/0
i_2	2/2 5/5 10/10	2/2	2/2 5/5 10/10	2/2 5/5 10/10	5/5	2/2 5/5 10/10	2/2 5/5 10/10	2/2 5/5 10/10	2/2 5/5 10/10	10/10

By definition, only states $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ can be distinguished by i_2; therefore, until this subset is reached states of any other subset of cardinality more than two cannot be distinguished. Moreover, input i_2 cannot be applied when analyzing such a subset due to merging reasons. Given the initial subset $c = \{1, \Sigma_1+1, \Sigma_2+1, \ldots, \Sigma_{n-1}+1\}$ of n states, the subset b can be reached from c only when input i_1 is applied Π_n times, that is known to be exponential in n.

Proposition 7. The length of a shortest DTC for $\mathbf{S}_n$ is at least Π_n.

The height of a shortest adaptive distinguishing test case for a complete observable FSM with n states is known to reach $2^n - n - 1$ [8, 16]. However, the machines of

the proposed class have exponential number of inputs with respect to the number of states. On the other hand, given an arbitrary observable FSM, we still have no procedure for deriving an adaptive distinguishing experiment by using a memory of polynomial size. For this reason, Fig. 4 has only the upper bound on the length of an adaptive distinguishing experiment.

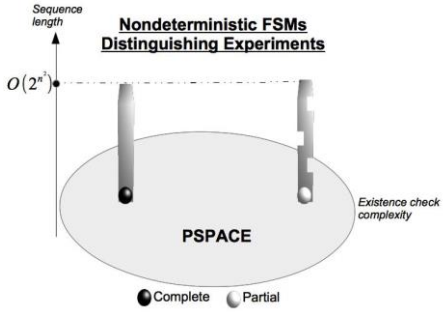


Fig. 4. How partiality and adaptivity affect the complexity of distinguishing experiments for nondeterministic FSMs

4.2 Homing experiments

For complete reduced deterministic FSMs a homing sequence always exists. For complete nondeterministic but observable FSMs the problem becomes PSPACE-complete. In order to describe the set of all homing sequences for a nondeterministic possibly partial FSM we again use the procedure (with slight modifications) for deriving an appropriate automaton in [23].

For a nondeterministic FSM $\mathsf{S} = (S, I, O, h)$, $S = \{s_1, s_2, \ldots, s_n\}$, we derive an automaton S^2_{home} such that the set of (all) synchronizing sequences of this automaton coincides with the set of (all) homing sequences of FSM S, i.e., $L_{home}(\mathsf{S}) = L_{synch}(\mathsf{S}^2_{home})$. The derivation of this automaton is very close to S^2_{dist}: we do not care if for some $o \in O$, the *io*-successors of states s_j and s_k coincide.

Differently from complete nondeterministic FSMs, the automaton S^2_{home} can be partial and nondeterministic. Similar to [32], it can be shown that the set of synchronizing sequences of the automaton S^2_{home} coincides with the set of all homing sequences of the FSM S.

Proposition 8. An input sequence α is a homing sequence for the FSM S if and only if α is a synchronizing sequence for S^2_{home}.

The length of a shortest homing sequence for a complete observable FSM with n states is known to reach $2^{n-1} - 1$ [8]. This means that the complexity of checking the existence of a homing sequence for nondeterministic observable FSMs cannot be in NP. Therefore, taking into account the complexity of the existence check for the synchronizing sequences, we conclude the following.

Proposition 9. The problem of checking the existence of a homing sequence is PSPACE-complete for observable complete and partial nondeterministic FSMs.
For complete FSMs the complexity of checking the existence of a homing test case is in P [8]. For partial deterministic FSMs it was shown that the problem is PSPACE-complete [14], i.e., the partiality destroys the polynomial complexity for adaptive homing sequences. The impact of adaptivity and partiality on homing experiments for nondeterministic FSMs is shown in Fig. 5.

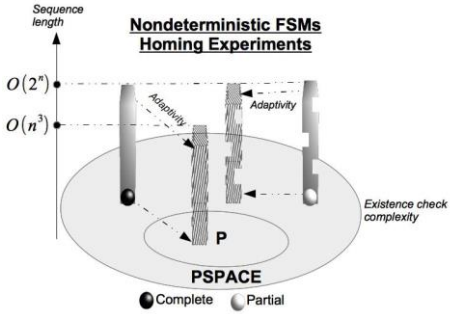


Fig. 5. How partiality and adaptivity affect the complexity of homing experiments for nondeterministic FSMs

4.3 Synchronizing experiments

As the problem of checking the existence of a homing sequence for complete and partial nondeterministic FSMs is PSPACE-complete, the problem of checking the existence of a synchronizing sequence for complete and partial nondeterministic FSMs is not easier, but it is known to have the same complexity.
For adaptive experiments, it was shown that for complete nondeterministic FSMs the existence check of an adaptive synchronizing sequence/test case is in P [12].

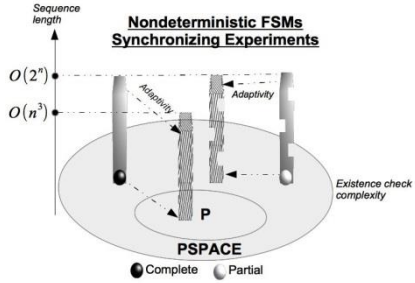


Fig. 6. How partiality and adaptivity affect the complexity of synchronizing experiments for nondeterministic FSMs

The problem of checking the existence of an adaptive homing sequence for deterministic partial FSMs is PSPACE-complete, and thus, the existence check of an adaptive synchronizing sequence for nondeterministic observable partial FSMs is PSPACE-hard. The impact of adaptivity and partiality on synchronizing experiments for nondeterministic FSMs is shown in Fig. 6.

5. Conclusions

In this paper, we have considered the problems of checking the existence of homing, synchronizing, and distinguishing experiments for various FSM types, namely, for complete and partial, deterministic and nondeterministic FSMs. We studied how the adaptivity and partiality influence the complexity of the existence check for such experiments as well as the length of the corresponding sequences. As a conclusion, we can say that in general, for distinguishing experiments the partiality does not increase the complexity but it is not the case for homing/synchronizing sequences. For the sake of simplicity, we visualized the obtained complexity results via appropriate figures. Thus, there are the following contributions. A new problem has been introduced of studying the dependencies how partiality and adaptivity influence the complexity issues of "gedanken" experiments for FSMs. We also close some open issues in the area; in particular, we show that differently from deterministic machines the adaptivity does not help to reduce the complexity of adaptive distinguishing experiments for nondeterministic FSMs (even for 2-input FSMs). All the known results have been collected together and the complexity issues have been visualized via appropriate figures. A simple, yet important conclusion from this study is that the partiality and adaptivity work in opposite directions on the complexity of the state identification problems. When we consider partiality and adaptivity together, in some cases partiality is more dominant and it makes the problem more complex, and in some cases, adaptivity is more dominant and it makes the problem easier. It is interesting to study what are the characteristics of the problem in order to be able to decide which force, adaptivity or partiality, wins, and why.

An interesting question for future research covers the complexity of deriving such experiments for various FSM types and the tight upper bounds on the length/height of a shortest distinguishing/homing/synchronizing experiment (if it exists). Moreover, the complexity issues are very interesting for non-observable FSMs and in fact, there are not many papers on this topic. We also mention that appropriate FSM classes can be considered where the complexity goes down compared to a general case. The issues listed above form the challenges for the nearest future work.

Acknowledgements

This work was supported by the Russian Science Foundation (RSF), project #16-49-03012, and by the Scientific and Technological Research Council of Turkey (TUBİTAK), project #114E921.

References

[1]. Moore E.F. Gedanken-experiments on sequential machines. Automata Studies (Annals of Mathematical Studies), 1956, 1, pp. 129-153.
[2]. Gill A. State-identification experiments in finite automata. Information and Control, 1961, pp. 132-154.
[3]. Gill A. Introduction to the Theory of Finite-State Machines. 1962, 207 p.
[4]. Kohavi Z. Switching and Finite Automata Theory. 1979, 658 p.
[5]. Sandberg S. Homing and Synchronization Sequences. Lecture Notes in Computer Science, 2005, 3472, pp. 5–33.
[6]. Lee D., Yannakakis M. Testing Finite-State Machines: State Identification and Verification. IEEE Transactions on Computers, 1994, 43 (3), pp. 306-320.
[7]. Alur R., Courcoubetis C., Yannakakis, M. Distinguishing tests for nondeterministic and probabilistic machines. Proc. of the 27th ACM Symposium on Theory of Computing, 1995, pp. 363-372.
[8]. Kushik N., El-Fakih K., Yevtushenko, N. Adaptive homing and distinguishing experiments for nondeterministic finite state machines. Lecture Notes in Computer Science, 2013, 8254, pp. 33-48.
[9]. Kushik N., Yevtushenko N. On the Length of Homing Sequences for Nondeterministic Finite State Machines. Lecture Notes in Computer Science, 2013, 7982, pp. 220-231.
[10]. Hierons R.M., Türker U.C. Distinguishing Sequences for Partially Specified FSMs. Lecture Notes in Computer Science, 2014, 8430, pp. 62-76.
[11]. Yenigün H., Yevtushenko N., Kushik N. Some classes of finite state machines with polynomial length of distinguishing test cases. Proc, of International Symposium on Applied Computing (SAC'2016), 2016, pp. 1680-1685.
[12]. Kushik N., Yevtushenko N., Yenigun H. Reducing the complexity of checking the existence and derivation of adaptive synchronizing experiments for nondeterministic FSMs. Proc. of Intern. Workshop on Domain Specific Model-based Approaches to Verification and Validation (AMARETTO'2016), 2016, pp. 83-90.
[13]. Kushik N., El-Fakih K., Yevtushenko N., Cavalli A.R. On adaptive experiments for nondeterministic finite state machines. Software Tools for Technology Transfer, Springer, 2016, 18 (3), pp. 251-264.
[14]. Yenigun H., Yevtushenko N., Kushik N. The complexity of checking the existence and derivation of adaptive synchronizing experiments for deterministic FSMs. Information Processing Letters, 2017, 127, pp. 49-53.
[15]. Kushik N. G., Kulyamin V. V., Evtushenko N. V. On the complexity of existence of homing sequences for nondeterministic finite state machines. Programming and Computer Software, 2014, 40(6), pp. 333-336.
[16]. El-Fakih K., Yevtushenko N., Kushik N. Adaptive distinguishing test cases of nondeterministic finite state machines: test case derivation and length estimation. Formal Asp. Comput., 2018, 30(2), pp. 319-332.

[17]. N. Kushik, J. López, A. Cavalli, N. Yevtushenko. Improving Protocol Passive Testing through "Gedanken" Experiments with Finite State Machines. In Proceedings of Intern. Conf. on Software Quality, Reliability & Security (QRS'2016), 2016, pp. 315-322.
[18]. Yenigün H., Kushik N., López J., Yevtushenko N., Cavalli A.R. Decreasing the complexity of deriving tests against nondeterministic finite state machines. Proc. of East-West Design & Test Symposium (EWDTS), 2017, IEEE Xplore, IEEE, pp. 1-4.
[19]. Hopcroft, J.E., Ullman, J.D. Introduction to Automata Theory, Languages, and Computation (1st ed.), 1979, 521 p.
[20]. Volkov, M.V. Synchronizing automata preserving a chain of partial orders. Lecture Notes in Computer Science, Vol. 4783, pp. 27–37 (2007)
[21]. Hibbard T.N. Least Upper Bounds on Minimal Terminal State Experiments for Two Classes of Sequential Machines. J. ACM, 1961, 8(4), pp. 601-612.
[22]. Pavel V. Martyugin. Careful Synchronization of Partial Automata with Restricted Alphabets. Proceedings of the Intern. Conf. Computer Science in Russia (CSR'2013), 2013, pp. 76-87.
[23]. Kushik N., Yevtushenko N. Describing Homing and Distinguishing Sequences for Nondeterministic Finite State Machines via Synchronizing Automata. Lecture Notes in Computer Science, 2015, 9223, pp. 188-198.

Влияние частичности и адаптивности на сложность задачи идентификации состояний автомата

[1]Х. Йенигун <yenigun@sabanciuniv.edu>
[2,3,4]Н. Евтушенко <nyevtush@gmail.com>
[5]Н. Кушик <natalia.kushik@telecom-sudparis.eu >
[5]Х. Лопез <jorge.lopez@telecom-sudparis.eu >
[1] Университет Сабанчи, Стамбул, Турция
[2] Национальный исследовательский Томский государственный университет, 634050, Томск, пр. Ленина 36
[3] Институт системного программирования им. В.Р. Иванникова РАН, 109004, Москва, ул. Солженицына 25
[4]Национальный исследовательский университет Высшая школа экономики, 101000, Москва, ул. Мясницкая, д. 20
[5]SAMOVAR, CNRS, Телеком Южный Париж/Университет Париж Сакле, Еври, Франция

Аннотация. Задача идентификации состояний в конечном автомате была и остается актуальной, поскольку используется во многих приложениях, в частности, при моделировании и тестировании дискретных управляющих систем. Для идентификации текущего состояния системы строятся так называемые установочные и синхронизирующие эксперименты с конечными автоматами, в то время как для идентификации начального состояния системы используются диагностические или различающие эксперименты. Установочные, синхронизирующие, диагностические или различающие эксперименты известны как «умозрительные» эксперименты с конечными автоматами, и методы синтеза входных последовательностей для таких

экспериментов (если существуют) в настоящее время определены для недетерминированных и детерминированных, полностью и частично определенных автоматов, описывающих эталонное поведение системы. Известно, что проблема проверки существования и построения установочных, синхронизирующих, диагностических или различающих экспериментов существенно усложняется, если эталонное поведение системы описывается недетерминированным и частичным автоматом, что достаточно часто случается при описании сложных современных систем. Известно также, что в некоторых случаях сложность можно понизить, переключившись на адаптивный (условный) эксперимент с системой. В настоящей работе мы исследуем влияние частичности автомата и адаптивности эксперимента на сложность проверки существования и построения установочных, синхронизирующих, диагностических и различающих экспериментов для детерминированных и недетерминированных автоматов и иллюстрируем соответствующую сложность с использованием подходящих рисунков.

Ключевые слова: конечные автоматы, задача идентификации состояний, сложность.

DOI: 10.15514/ISPRAS-2018-30(1)-1

Для цитирования: Йенигун Х., Евтушенко Н., Кушик Н., Лопез Х. Влияние частичности и адаптивности на сложность задачи идентификации состояний автомата. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(1)-1

Список литературы

[1]. Moore E.F. Gedanken-experiments on sequential machines. Automata Studies (Annals of Mathematical Studies), 1956, 1, pp. 129-153.
[2]. Gill A. State-identification experiments in finite automata. Information and Control, 1961, pp. 132-154.
[3]. Gill A. Introduction to the Theory of Finite-State Machines. 1962, 207 p.
[4]. Kohavi Z. Switching and Finite Automata Theory. 1979, 658 p.
[5]. Sandberg S. Homing and Synchronization Sequences. Lecture Notes in Computer Science, 2005, 3472, pp. 5–33.
[6]. Lee D., Yannakakis M. Testing Finite-State Machines: State Identification and Verification. IEEE Transactions on Computers, 1994, 43 (3), pp. 306-320.
[7]. Alur R., Courcoubetis C., Yannakakis, M. Distinguishing tests for nondeterministic and probabilistic machines. Proc. of the 27th ACM Symposium on Theory of Computing, 1995, pp. 363-372.
[8]. Kushik N., El-Fakih K., Yevtushenko, N. Adaptive homing and distinguishing experiments for nondeterministic finite state machines. Lecture Notes in Computer Science, 2013, 8254, pp. 33-48.
[9]. Kushik N., Yevtushenko N. On the Length of Homing Sequences for Nondeterministic Finite State Machines. Lecture Notes in Computer Science, 2013, 7982, pp. 220-231.
[10]. Hierons R.M., Türker U.C. Distinguishing Sequences for Partially Specified FSMs. Lecture Notes in Computer Science, 2014, 8430, pp. 62-76.
[11]. Yenigün H., Yevtushenko N., Kushik N. Some classes of finite state machines with polynomial length of distinguishing test cases. Proc, of International Symposium on Applied Computing (SAC'2016), 2016, pp. 1680-1685.

[12]. Kushik N., Yevtushenko N., Yenigun H. Reducing the complexity of checking the existence and derivation of adaptive synchronizing experiments for nondeterministic FSMs. Proc. of Intern. Workshop on Domain Specific Model-based Approaches to Verification and Validation (AMARETTO'2016), 2016, pp. 83-90.

[13]. Kushik N., El-Fakih K., Yevtushenko N., Cavalli A.R. On adaptive experiments for nondeterministic finite state machines. Software Tools for Technology Transfer, Springer, 2016, 18 (3), pp. 251-264.

[14]. Yenigun H., Yevtushenko N., Kushik N. The complexity of checking the existence and derivation of adaptive synchronizing experiments for deterministic FSMs. Information Processing Letters, 2017, 127, pp. 49-53.

[15]. Kushik N. G., Kulyamin V. V., Evtushenko N. V. On the complexity of existence of homing sequences for nondeterministic finite state machines. Programming and Computer Software, 2014, 40(6), pp. 333-336.

[16]. El-Fakih K., Yevtushenko N., Kushik N. Adaptive distinguishing test cases of nondeterministic finite state machines: test case derivation and length estimation. Formal Asp. Comput., 2018, 30(2), pp. 319-332.

[17]. N. Kushik, J. López, A. Cavalli, N. Yevtushenko. Improving Protocol Passive Testing through "Gedanken" Experiments with Finite State Machines. In Proceedings of Intern. Conf. on Software Quality, Reliability & Security (QRS'2016), 2016, pp. 315-322.

[18]. Yenigün H., Kushik N., López J., Yevtushenko N., Cavalli A.R. Decreasing the complexity of deriving tests against nondeterministic finite state machines. Proc. of East-West Design & Test Symposium (EWDTS), 2017, IEEE Xplore, IEEE, pp. 1-4.

[19]. Hopcroft, J.E., Ullman, J.D. Introduction to Automata Theory, Languages, and Computation (1st ed.), 1979, 521 p.

[20]. Volkov, M.V. Synchronizing automata preserving a chain of partial orders. Lecture Notes in Computer Science, Vol. 4783, pp. 27–37 (2007)

[21]. Hibbard T.N. Least Upper Bounds on Minimal Terminal State Experiments for Two Classes of Sequential Machines. J. ACM, 1961, 8(4), pp. 601-612.

[22]. Pavel V. Martyugin. Careful Synchronization of Partial Automata with Restricted Alphabets. Proceedings of the Intern. Conf. Computer Science in Russia (CSR'2013), 2013, pp. 76-87.

[23]. Kushik N., Yevtushenko N. Describing Homing and Distinguishing Sequences for Nondeterministic Finite State Machines via Synchronizing Automata. Lecture Notes in Computer Science, 2015, 9223, pp. 188-198.

О возможностях автоматного описания параллельной композиции временных автоматов

А.С. Твардовский <tvardal@mail.ru>
А.В. Лапутенко <laputenko.av@gmail.com>
Национальный исследовательский Томский государственный университет, 634050, Россия, Томск, пр. Ленина 36

Аннотация. Конечные автоматы широко используются для анализа и синтеза дискретных систем. При описании систем, поведение которых зависит от времени, конечный автомат расширяется введением временных аспектов и вводится понятие временного автомата. В настоящей статье рассматривается проблема построения параллельной композиции для двух моделей временных автоматов, а именно, для автоматов с таймаутами и автоматов с временными ограничениями. Две эти формы временных автоматов не являются взаимозаменяемыми и являются более частными случаями общей модели временного автомата, содержащего как таймауты, так и временные ограничения. Мы также считаем, что все выше упомянутые модели временных автоматов имеют целочисленные выходные задержки (выходные таймауты). Автоматы-компоненты работают в режиме диалога, по завершении которого композиция выдаёт внешний выходной символ. При решении задач анализа для системы взаимодействующих конечных автоматов с использованием классических методов такая композиция обычно описывается единственным автоматом. В работе показывается, что в общем случае, в отличие от случая классических конечных автоматов, наличия «медленной внешней среды» и отсутствия осцилляций недостаточно для описания поведения композиции детерминированным автоматом с одной временной переменной, если входные символы могут поступать не только в целочисленные, но и рациональные моменты времени. Тем не менее, определяется класс систем, в которых каждое внешнее входное воздействие инициирует диалог между компонентами, что позволяет описать поведение такой композиции детерминированным автоматом с одной временной переменной. В частности, рассматривается последовательная композиция временных автоматов, которая удовлетворяет такому ограничению. Другое ограничение продиктовано наличием таймаутов, значение которого в каждом из состояний должно превышать величину выходной задержки при обработке любого перехода в этом состоянии.

Ключевые слова: конечный автомат; входные и выходные таймауты; временные ограничения; композиция.

DOI: 10.15514/ISPRAS-2018-30(1)-2

Для цитирования: Твардовский А.С., Лапутенко А.В.. О возможностях автоматного описания параллельной композиции временных автоматов. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 25-40. DOI: 10.15514/ISPRAS-2018-30(1)-2

1. Введение

Формальные модели широко используется для анализа и синтеза программного и аппаратного обеспечения, и одной из таких моделей является классический конечный автомат [1]. Конечный автомат описывает поведение системы с конечным числом состояний, которая может переходить из состояния в состояние при наличии входного воздействия, выдавая при этом выходную реакцию. Модель конечного автомата обладает естественной реактивностью и позволяет строить для управляющих систем проверяющие тесты с гарантированной полнотой обнаружения ошибок (неисправностей) [2].

При описании поведения современных управляющих систем необходимо учитывать временные аспекты в их поведении. Для этого модель конечного автомата расширяется введением временной переменной, и вводится понятие временного автомата [3-6]. Временная переменная постоянно увеличивает своё значение и «сбрасывается» в 0 при поступлении входного символа или выдаче выходного символа. Для описания временных аспектов в автоматной модели используются временные ограничения [4], (входные) таймауты [5] и выходные задержки, иногда называемые выходными таймаутами. Входной таймаут определяет максимальное время ожидания входного воздействия для каждого состояния автомата. Если входной символ не был подан до истечения таймаута, то автомат может спонтанно перейти в другое состояние (без поступления входного и выдачи выходного символа). Временные ограничения представляют собой интервалы на переходах, ограничивающие время, в течение которого переход может быть выполнен. Выходные задержки (выходные таймауты) отражают время, затрачиваемое автоматом на выполнение перехода. В общем случае, временной автомат содержит таймауты, временные ограничения и выходные задержки. Однако в настоящей работе мы рассматриваем автоматы только с таймаутами и автоматы только с временными ограничениями, полагая, что выходные задержки существуют в обоих случаях. Отметим, что два рассматриваемых класса временных автоматов не являются взаимозаменяемыми, т.е. существует автомат с таймаутами, который не может быть описан автоматом с временными ограничениями, и наоборот [7].

Для описания поведения сложных систем обычно используется система взаимодействующих автоматов, т.е. композиция более простых в некотором смысле автоматов-компонент. В случае операции параллельной композиции, достаточно часто используемой при описании поведения взаимодействующих программных систем, автоматы-компоненты работают в режиме диалога, выдавая реакцию на внешний входной символ после окончания обмена сообщениями. Известно [8], что поведение композиции классических

детерминированных автоматов может быть описано конечным автоматом, если внешняя среда является «медленной», т.е. следующий входной символ на систему подается только после выдачи выходной реакции на предыдущий входной символ. После перехода к такому представлению для анализа полученного автомата могут быть использованы алгоритмы классической теории автоматов [1]. В настоящей работе показывается, что в отличие от случая классических автоматов, наличия «медленной внешней среды» может оказаться недостаточно для описания детерминированным автоматом с одной временной переменной поведения композиции временных автоматов в рациональные моменты времени. Соответственно, мы определяем классы композиций автоматов с временными ограничениями и автоматов с таймаутами, поведение которых в рациональные моменты времени может быть описано детерминированным временным автоматом.

2. Основные определения

В данной работе мы рассматриваем два типа временных автоматов, а именно автоматы с таймаутами и автоматы с временными ограничениями. Такие временные автоматы являются достаточно простыми расширениями классического конечного автомата, и мы ниже приводим их формальные определения [7].

Под автоматом с таймаутами понимается пятёрка $\mathsf{S} = (S, I, O, h_\mathsf{S}, \Delta_\mathsf{S})$, где I – *входной* алфавит, O – *выходной* алфавит, S – конечное непустое множество *состояний*, $h_\mathsf{S} \subseteq (S \times I \times O \times S \times Z)$ – *отношение переходов*, Z – множество целых неотрицательных чисел, определяющих число тактов времени между поступлением входного символа и выдачей выходного, $\Delta_\mathsf{S}\colon S \to S \times (\mathrm{N} \cup \{\infty\})$ – функция (входного) *таймаута*, определяющая для каждого состояния максимальное время ожидания входного символа, N – множество натуральных чисел. Иными словами, если в некотором состоянии входной сигнал не поступает в течение определенного времени (таймаута), то автомат может спонтанно изменить своё состояние. Если $\Delta_\mathsf{S}(s) = (s', T)$ и в состоянии s в течение T тактов времени на автомат не было подано ни одного входного символа, то автомат переходит в состояние s'. После перехода в состояние s' отсчет времени начинается с 0. В случае, когда $\Delta_\mathsf{S}(s) = (s, \infty)$, автомат может ожидать входной символ в состоянии s бесконечно долго. Если в автомате есть переход (s, i, o_1, s_1, d) и входной символ будет подан в состоянии s менее чем через T тактов времени после перехода автомата в текущее состояние, то автомат перейдет в состояние s_1 и через d тактов времени выдаст выходной символ. Число d тактов времени между подачей входного символа и выдачей выходного символа называется *выходной задержкой* (*выходным таймаутом*). Для того чтобы избежать противоречий при интерпретации входных и выходных таймаутов, мы полагаем, что в каждом состоянии входной таймаут больше задержки, необходимой для обработки любого входного символа. Автомат с

таймаутами называется *полностью определённым*, если для любой пары $(s, i) \in S \times I$, то есть для любого входного символа i, поступающего на вход автомата в состоянии s, существует переход $(s, i, o, s', d) \in h_S$. Если для любой пары $(s, i) \in S \times I$, существует единственный переход $(s, i, o, s', d) \in h_S$, то автомат с таймаутами называется *детерминированным*, иначе – *недетерминированным*.

Под *автоматом с временными ограничениями* понимается четвёрка $S = (I, S, O, h_S)$, где I – входной алфавит, O – выходной алфавит, S – конечное непустое множество состояний, $h_S \subseteq (S \times I \times O \times S \times \Pi \times Z)$ – отношение переходов, Π – множество интервалов из промежутка $[0; \infty)$, и Z – множество целых неотрицательных чисел. Соответственно, *кортеж* (s, i, o, s', g, d) описывает переходы из состояния s в состояние s' под действием входного символа i, поступившего в момент времени t, $t \in g$, после перехода автомата в текущее состояние с выдачей выходного символа o через d тактов времени после поступления входного символа. Если для любых двух кортежей $(s, i, o_1, s_1, g_1, d_1)$, $(s, i, o_2, s_2, g_2, d_2) \in h_S$ справедливо соотношение $g_1 \cap g_2 = \varnothing$, то автомат с временными ограничениями называется *детерминированным*, иначе – *недетерминированным*. Автомат с временными ограничениями называется *полностью определённым по входным символам*, если для любой пары $(s, i) \in S \times I$, то есть для любого входного символа i, поступающего на вход автомата в состоянии s, существует кортеж $(s, i, o, s', g, d) \in h_S$. Автомат с временными ограничениями называется *полностью определённым*, если автомат полностью определён по входным символам, и для каждой пары $(s, i) \in S \times I$ объединение всех временных интервалов в состоянии s по входному символу i равно $[0; \infty)$. В качестве примера технической системы, поведение которой описывается временным автоматом, можно привести систему сигнализации, рассмотренную в [9].

Временным входным символом называется пара (i, t), где i – символ входного алфавита, t – (вещественное) время поступления входного символа после перехода автомата в текущее состояние. *Временным выходным символом* называется пара (o, d), где o – символ выходного алфавита, d – выходная задержка. Последовательность временных входных символов (i_1, t_1), (i_2, t_2) …, (i_n, t_n) называется *временной входной последовательностью*; последовательность временных выходных символов (o_1, d_1), (o_2, d_2) …, (o_n, d_n) называется *временной выходной последовательностью*. Аналогично конечным автоматам, временная выходная последовательность, соответствующая временной входной последовательности α, поступившей на автомат в состоянии s, называется (*выходной*) *реакцией* автомата в состоянии s на последовательность α.

В настоящей работе мы рассматриваем детерминированные полностью определённые временные автоматы, т.е. автоматы, в которых каждому состоянию, входному символу и моменту времени соответствует единственный переход.

3. Композиция временных автоматов

Сложные системы обычно являются композициями более простых в некотором смысле систем, и в настоящей работе мы рассматриваем параллельную композицию [10-12] двух детерминированных полностью определенных временных автоматов (рис. 1).

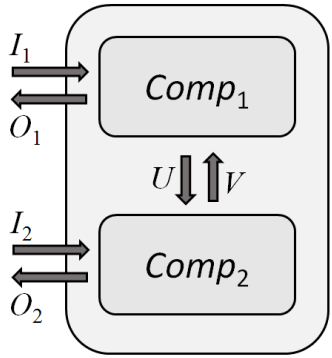


Рис. 1. Параллельная композиция двух временных автоматов
Fig. 1. Parallel composition of two TFSMs

Работа параллельной композиции может быть описана следующим образом. Пусть временной входной символ подаётся на компоненту $Comp_1$ ($Comp_2$), которая может отреагировать внутренним или внешним выходным символом. Если был произведён внешний выходной символ, то композиция готова принять следующее внешнее входное воздействие; однако, в этом случае, к значению временной переменной компоненты $Comp_2$ ($Comp_1$) прибавляется время, в течение которого $Comp_1$ ($Comp_2$) взаимодействовала с внешней средой. Если компонента $Comp_1$ ($Comp_2$) выдает внутреннюю выходную реакцию, то компонента $Comp_2$ ($Comp_1$) обрабатывает это воздействие в соответствии с предписанным ей поведением и производит внутреннее или внешнее выходное воздействие. Предполагается, что следующий входной символ подаётся на композицию только после получения реакции на предыдущий входной символ, т.е. внешняя среда является «медленной».

С целью построения проверяющих тестов для композиции классическими методами (см. например, [13]), поведение такой композиции достаточно часто описывается одним автоматом, и для классических конечных автоматов методы построения композиции достаточно хорошо изучены. Известно [8], что, если компоненты композиции являются детерминированными полностью определенными автоматами, то поведение параллельной композиции описывается полностью определенным детерминированным автоматом при наличии «медленной внешней среды» и отсутствии осцилляций, т.е. бесконечного диалога между компонентами. В этом случае состояния автомата-композиции обычно представляют собой пару состояний компонент.

Задача построения композиций временных автоматов рассматривалась в ряде работ [10-12], в которых состояния композиции сохраняют не только состояния компонент, но также и текущее значение временной переменной одной или обеих компонент. В [10-11] было предложено синтезировать композицию путём построения дерева преемников, в корне которого находятся начальные состояния компонент. В [12] предлагается перейти от временных автоматов к соответствующим полуавтоматам и построить композицию путём их пересечения. Однако далее мы показываем, что оба подхода применимы лишь для ограниченного класса систем и в общем случае поведение композиции детерминированных временных автоматов не может быть описано детерминированным временным автоматом.

4. Описание композиции автоматов с таймаутами детерминированным автоматом

В настоящем разделе мы рассматриваем проблемы, возникающие при описании одним автоматом композиции временных автоматов, и в частности, показываем, в каком случае поведение параллельной композиции временных автоматов может быть описано детерминированным временным автоматом.

4.1 Проблема описания композиции

Как было отмечено выше, для вычисления реакции композиции необходимо знать значение временной переменной каждой из компонент. Однако, если одна из компонент взаимодействует с внешней средой без внутреннего диалога с другой компонентой, то нарушается «синхронизация» между часами компонент и может оказаться, что однозначно вычислить значение временной переменной в состоянии автомата-композиции становится невозможным. Причиной является тот факт, что компонента, не взаимодействующая с внешней средой «накапливает» вещественное значение временной переменной и в зависимости от момента поступления внешнего входного воздействия может «перейти» или «не перейти» целочисленный порог, изменяющий поведение компоненты. Вообще говоря, последнее согласуется с результатами в области временных полуавтоматов [14], где показано, что не для всякого временного полуавтомата с несколькими временными переменными существует эквивалентный полуавтомат с одной временной переменной.

Рассмотрим пример параллельной композиции двух автоматов с таймаутами, представленный на рис. 2. В начальном состоянии временные переменные обеих компонент равны нулю. Если подать на композицию временную входную последовательность (i_{11}, 1.8), (i_{11}, 1.8), то компонента S выдаст выходную временную последовательность (o_{11}, 3), (o_{11}, 3), «сбрасывая» значение своей временной переменной в ноль после выдачи каждого выходного символа. При этом, автомат P перейдет в состояние p_2 по таймауту

8 и значение временной переменной автомата P будет равно 1.6, что вместе со значением таймаута, по которому осуществился переход в состояние p_2, равно сумме времени подачи каждого входного временного символа и времени задержки при выдаче каждого выходного временного символа автоматом S. Если затем подать входной временной символ (i_{11}, 5.3), то автомат S, перейдя из состояния s_1 в состояние s_2 по таймауту 4, передаст внутренний выходной временной символ (u, 1) на автомат P.

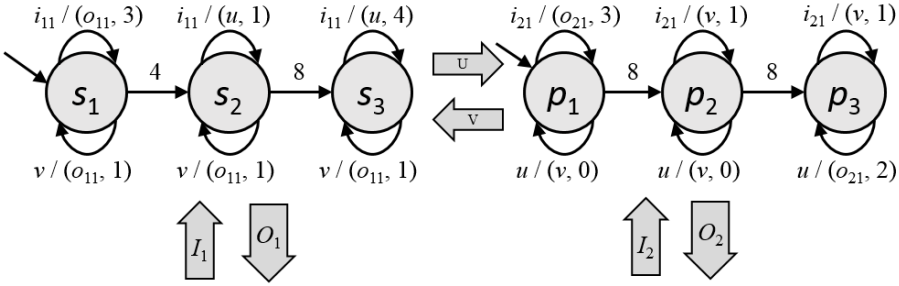


Рис. 2. Композиция временных автоматов S и P
Fig. 2. Parallel composition of TFSMs S and P

В этот момент времени автомат P будет находиться в состоянии p_2 со значением временной переменной, равным 7.9, и передаст внутренний выходной символ (v, 0) на первую компоненту, которая, находясь в состоянии s_2 выдаст временной выходной символ (o_{11}, 2). Если же в качестве третьего временного символа подать не (i, 5.3), а (i, 5.4), то компонента S по-прежнему перейдет по таймауту 4 в состояние s_2 и выдаст компоненте P внутренний выходной символ (u, 1). Но за этот временной промежуток временная переменная автомата P примет значение 8 и автомат перейдет в состояние p_3 и выдаст внешний выходной символ (o_{21}, 3). Таким образом, реакции композиции не совпадают на временные входные последовательности (i_{11}, 1.8), (i_{11}, 1.8), (i_{11}, 5.3) и (i_{11}, 1.8), (i_{11}, 1.8), (i_{11}, 5.4), в то время как третий входной символ подается в промежутке (5, 6), который не может быть уменьшен при описании временного автомата, поскольку значения таймаутов являются целыми неотрицательными числами.

Таким образом, если входные символы могут подаваться на компоненты в вещественные моменты времени, то описание такой композиции детерминированным автоматом с конечным числом состояний в общем случае невозможно, и таким образом, в отличие от классических автоматов, для описания детерминированным полностью определенным автоматом композиции временных автоматов недостаточно наличия «медленной» внешней среды и отсутствия осцилляций.

Как показывает приведенный ниже пример, сказанное выше справедливо и для композиции автоматов с временными ограничениями. Рассмотрим пример композиции автоматов S и P на рис. 3. Непосредственной проверкой можно

убедиться, что реакции композиции не совпадают на временные входные последовательности (i_{11}, 0.2), (i_{11}, 1.1) и (i_{11}, 0.2), (i_{11}, 1.9), в то время как второй входной символ подается в промежутке (1, 2), который не может быть уменьшен при описании временного автомата, поскольку границы интервалов являются целыми неотрицательными числами.

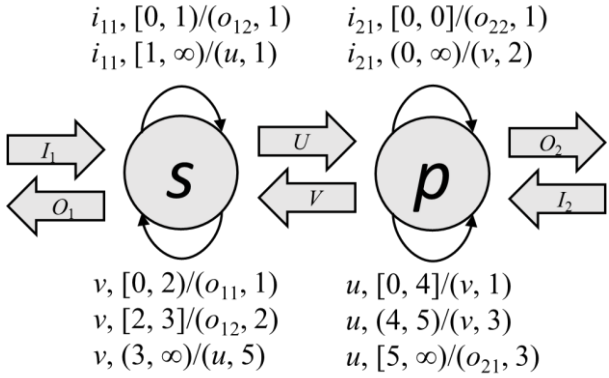


Рис. 3. Композиция временных автоматов S *и* P
Fig. 3. Parallel composition of TFSMs S *and* P

Следует отметить, что выше описанная ситуация не имеет места, если входные воздействия могут подаваться на компоненты только в целочисленные моменты времени, и в этом случае композиция детерминированных полностью определенных автоматов является детерминированным полностью определенным автоматом при условиях «медленной» внешней среды и отсутствия осцилляций.

В следующем разделе мы вводим дополнительные ограничения для композиции временных автоматов и рассматриваем класс систем, для которых поведение композиции временных автоматов можно описать детерминированным автоматом с одной временной переменной.

4.2 Достаточное условие для описания композиции детерминированным автоматом

Из описанного выше примера видно, что описание поведения параллельной композиции детерминированным автоматом может оказаться невозможным, когда входные воздействия могут поступать на композицию в вещественные моменты времени. Далее мы вводим класс систем, для которых такое описание возможно.

Теорема 1. Параллельная композиция автоматов с таймаутами S и P может быть описана детерминированным временным автоматом, если в компоненте S не существует перехода (s, i, s', o) $\in h_S$ (в компоненте P не существует перехода (p, i, p', o) $\in h_P$), где i и o – внешние входной и выходной символы, и

в построенной параллельной композиции в каждом состоянии входной таймаут больше, чем выходные задержки для обработки каждого входного символа.

Доказательство. Пусть в параллельной композиции автоматов с таймаутами S и P, где I_s и O_s (I_p и O_p) – внешние входной и выходной алфавиты компоненты S (P), V и U – входной и выходной (выходной и входной) алфавиты компоненты S (P) и любой переход автоматов S и P по внешнему входному символу приводит к диалогу между компонентами. В этом случае, каждая из компонент обязательно выполнит хотя бы один переход между поступлением внешнего входного символа и выдачей внешнего выходного символа и сбросит время в 0. Соответственно, в момент выдачи внешнего выходного символа, временная переменная соответствующей компоненты сбрасывается в 0, в то время как время второй компоненты становится равным задержке последнего перехода компоненты, выдавшей внешний выходной символ, т.е. имеет целочисленное значение. Таким образом, в момент выдачи внешнего выходного символа временная переменная каждой из компонент целочисленная, и поведение параллельной композиции может быть описано автоматом с таймаутами с конечным числом состояний.

Иными словами, если в рассматриваемой композиции любое внешнее воздействие приводит к диалогу между компонентами, то после выдачи композицией внешнего выходного символа значение временной переменной каждой из компонент целочисленное и может быть сохранено в состоянии детерминированного временного автомата, описывающего работу композиции. Согласно определению, для того чтобы построенная параллельная композиция описывалась автоматом с таймаутами, достаточно, чтобы в каждом состоянии входной таймаут был больше, чем выходные задержки для обработки каждого входного символа.

Например, если в композиции автоматов S и P на рис. 2 заменить переходы по внешним выходным символам таким образом, чтобы любое внешнее входное воздействие приводило к диалогу между компонентами (рис. 4), то композиция такой системы может быть построена с использованием метода из [12]. В этом случае, по каждому автомату с таймаутами строится соответствующий классический полуавтомат, и композиция формируется путем пересечения полученных полуавтоматов.

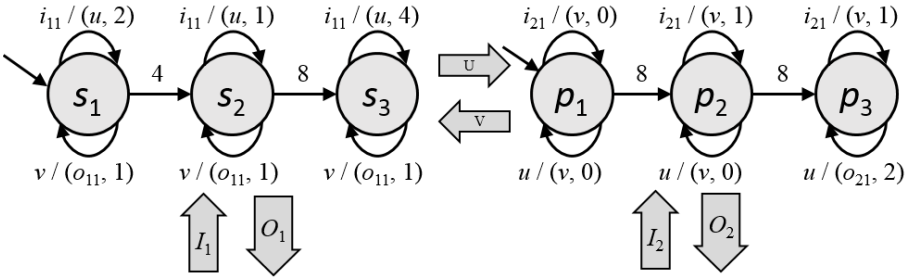


Рис. 4. Композиция временных автоматов S и P после изменения перехода
Fig. 4. Composition of TFSMs S and P after changing the transition

На рис. 5 приведен автомат с таймаутами, описывающий композицию двух автоматов с таймаутами, рассмотренных выше.

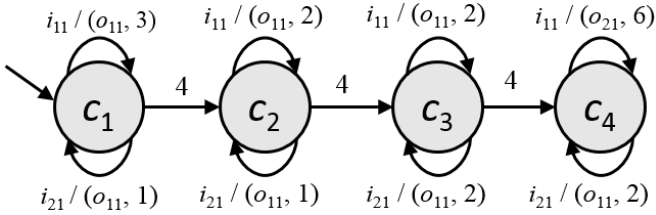


Рис. 5. Временной автомат C, описывающий работу композиции автоматов S и P, представленных на рис. 4
Fig. 5. The composed TFSM C of composition TFSMs S and P from fig. 4

Таким образом, мы можем сформулировать достаточное условие для описания композиции временных автоматов детерминированным временным автоматом, которое заключается в наличии «медленной внешней среды», отсутствии осцилляций и наличии диалога между компонентами в ответ на любое внешнее воздействие.

Аналогично можно сформулировать достаточное условие для описания параллельной композиции автоматов с временными ограничениями детерминированным конечным автоматом. В этом случае, композиция может быть построена с использованием метода из [10]. Состояния композиции представляют собой четвёрки (s, t_s, p, t_p), где s и p – состояния первой и второй компоненты соответственно, а t_s и t_p – значения их временных переменных. Композиция формируется на основе построения дерева преемников, в корне которого находятся начальные состояния компонент с нулевыми значениями переменных.

Теорема 2. Параллельная композиция автоматов с временными ограничениями S и P может быть описана детерминированным временным автоматом, если не существует перехода (s, i, o, s', g_s, d_s) $\in h_S$ ((p, i, o, p', g_p, d_p) $\in h_P$), где i и o – внешние входной и выходной символы.

Действительно, подобно композиции автоматов с таймаутами, если любое внешнее воздействие приводит к диалогу между компонентами, то после выдачи композицией внешнего выходного символа значение временной переменной каждой из компонент целочисленное и может быть сохранено в состоянии детерминированного временного автомата, описывающего работу композиции. Например, если в композиции автоматов S и P на рис. 3 заменить переходы по внешним выходным символам таким образом, чтобы любое внешнее входное воздействие приводило к диалогу между компонентами (рис. 6), то соответствующий автомат, описывающий композиции может быть построен методом из [10] и представлен на рис. 7.

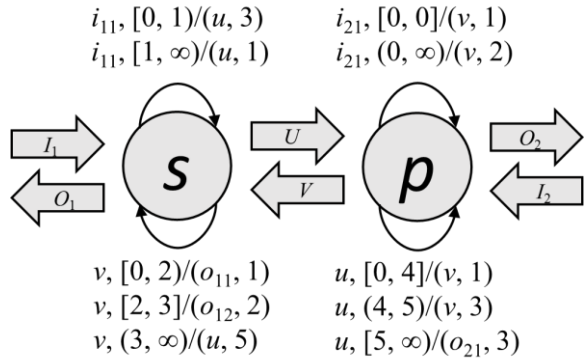


Рис. 6. Композиция временных автоматов S и P после изменения перехода
Fig. 6. Composition of TFSMs S and P after changing the transition

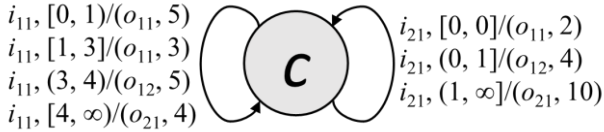


Рис. 7. Временной автомат C, описывающий работу композиции автоматов S и P, представленных на рис. 6
Fig. 7. The composed TFSM C of composition TFSMs S and P from fig. 6

4.3 Частные случаи композиции временных автоматов

Следует отметить, что на практике часто рассматриваются более простые частные случаи композиции. Так, например, в [10] рассматривается композиция встроенной компоненты с так называемым с контекстом, где лишь контекст взаимодействует с внешней средой. В этом случае можно доказать теоремы, аналогичные теоремам 1 и 2, т.е. для описания такой композиции детерминированным автоматом контекст обязательно должен инициировать диалог со встроенной компонентой при подаче каждого внешнего входного символа. Другим удобным для практики случаем является последовательная композиция, в которой лишь одна из компонент принимает внешние входные символы, в то время как другая компонента принимает входные воздействия от другой компоненты и выдает внешние выходные символы (рис. 8).

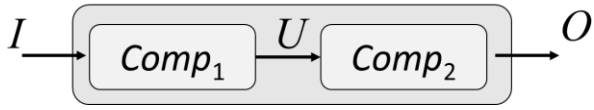


Рис. 8. Последовательная композиция двух временных автоматов
Fig. 8. Sequential composition of two TFSMs

Работа последовательной композиции выглядит следующим образом. Внешний входной символ поступает на первую компоненту, которая выполняет соответствующий переход и выдаёт внутренний выходной символ. Вторая компонента принимает внутренний входной символ и производит внешний выходной символ, тем самым завершая обработку внешнего воздействия. Пример такой композиции представлен на рис. 9. В этом случае выполнены достаточные условия теорем 1 и 2, т.е. поведение последовательностей композиции временных автоматов можно описать детерминированным автоматом с одной временной переменной.

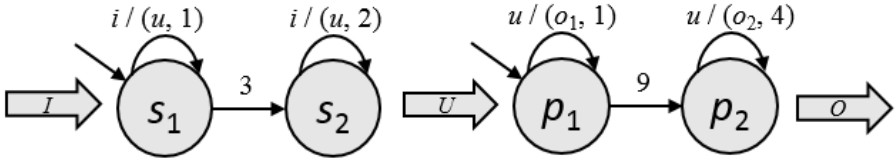


Рис. 9. Последовательная композиция двух временных автоматов S и P
Fig. 9. Sequential composition of TFSMs S и P

Временной автомат, описывающий композицию автоматов на рис. 9, представлен ниже (рис. 10).

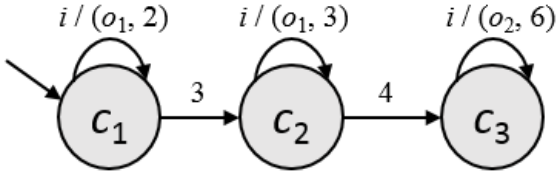


Рис. 10. Временной автомат C, описывающий работу композиции автоматов S и P, представленных на рис. 9
Fig. 10. The composed TFSM C of composition TFSMs S and P from fig. 9

5. Заключение

В настоящей статье рассмотрена задача автоматного описания композиции автоматов с таймаутами и автоматов с временными ограничениями. Показано, что в случае, когда временные переменные могут принимать вещественные значения, поведение параллельной композиции детерминированных временных автоматов не может быть описано детерминированным временным автоматом даже при наличии «медленной» внешней среды. Устанавливается достаточное условие для возможности такого описания, которое заключается в наличии «медленной» внешней среды, отсутствии осцилляций и возникновении диалога между компонентами в ответ на любое внешнее

воздействие. Этому условию удовлетворяет, в частности, последовательная композиция временных автоматов.

Отметим , что открытым остаётся вопрос о построении композиции временных автоматов общего вида [7], содержащих как таймауты, так и временные ограничения. Из приведённых в работе результатов следует, что композиция автоматов с таймаутами и временными ограничениями также не всегда может быть описана детерминированным временным автоматом. Также остаётся открытым вопрос о возможности использования недетерминированных автоматов для описания рассмотренных композиций.

Благодарности

Работа выполнена при поддержке гранта РНФ № 16-49-03012.

Список литературы

[1]. Гилл А. Введение в теорию конечных автоматов. М.: Издательство Наука, 1966, 272 с.

[2]. Rita Dorofeeva, Khaled El-Fakih, Stephane Maag, Ana R. Cavalli, Nina Yevtushenko FSM-based conformance testing methods: A survey annotated with experimental evaluation. Information and Software Technology, 2010, 52, pp. 1286-1297.

[3]. M. Merayo, M. Nunez, I. Rodriguez. Formal testing from timed finite state machines, Comput. Netw. 52 (2) (2008) 432–460.

[4]. El-Fakih K., Yevtushenko N., Simão A. A practical approach for testing timed deterministic finite state machines with single clock. Science of Computer Programming, Elsevier, 2014, Vol. 80, pp. 343–355.

[5]. Maxim Zhigulin, Nina Yevtushenko, Stéphane Maag, Ana R. Cavalli. FSM-Based Test Derivation Strategies for Systems with Time-Outs. Proc. of the 11th International Conference on Quality Software, QSIC2011, IEEE, 2011. pp. 141-149.

[6]. Larsen, K.G. Testing real-time embedded software using UPPAAL-TRON: an industrial case study. Proceedings of the 5th ACM international conference on Embedded software, 2005, pp. 299–306.

[7]. Bresolin D. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. Intern Conf. GANDALF, 2014, pp. 203–216.

[8]. Villa, T., Yevtushenko, N., Brayton, R.K., Mishchenko, A., Petrenko, A., Sangiovanni-Vincentelli, A. The Unknown Component Problem. Theory and Applications. Springer, 2012, 312 p.

[9]. Лапутенко А. В., Громов М. Л., Торгаев С. Н.. Реализация и тестирование системы сигнализации на баре микроконтроллера STM32F407VG. Изв. вузов. Физика, 2016, Т. 59, № 12, стр. 174-176.

[10]. Gromov M., Tvardovskii A., Yevtushenko N. Testing Systems of interacting Timed Finite State Machines with the Guaranteed Fault Coverage. International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices, 2016, pp. 96–99.

[11]. Gromov M., Tvardovskii A., Yevtushenko N. Testing Components of Interacting Timed Finite State Machines. Proceedings of IEEE East-West Design & Test Symposium (EWDTS), 2016, pp. 193–196.

[12]. O. Kondratyeva, M. Gromov. To the Parallel Composition of Timed Finite State Machines. Proceedings of the 5th Spring/Summer Young Researches' Colloquium on Software Engineering, 2011, pp. 94-99.
[13]. Alexandre Petrenko, Nina Yevtushenko, Gregor von Bochmann: Fault Models for Testing in Context. FORTE 1996, pp. 163-178
[14]. Tripakis S. Folk Theorems on the Determinization and Minimization of Timed Automata. In: Larsen K.G., Niebert P. (eds) Formal Modeling and Analysis of Timed Systems. FORMATS 2003. Lecture Notes in Computer Science, vol. 2791, pp. 222-226.

On the possibilities of FSM description of Parallel composition of Timed Finite State Machines

A.S. Tvardovskii <tvardal@mail.ru>
A.V. Laputenko <laputenko.av@gmail.com>
National Research Tomsk State University,
Tomsk, av. Lenina, 36, 634050, Russia

Abstract. Finite State Machines (FSMs) are widely used for analysis and synthesis of digital components of control systems. In order to take into account time aspects, timed FSMs are considered. In this paper, we address the problem of deriving a parallel composition of two types of Timed Finite State Machines (TFSM), namely, FSMs with timeouts and FSMs with timed guards. These two TFSM types are not interchangeable and are particular cases of a more general TFSM model that has timeouts and timed guards. We also assume that all of considered TFSMs have output delays (output timeouts). When considering the parallel composition, component FSMs work in the dialog and the composition produces an external output when interaction between components is finished. In this work, it is shown that in the general case, unlike classical FSMs, a "slow environment" and the absence of livelocks are not enough for describing the behavior of a composition by a complete deterministic FSM with a single clock. The latter occurs when inputs can be applied to TFSMs not only at integer time instances but also at rational. A class of systems for which the behavior can be described by a complete deterministic TFSM is specified. Those are systems where both component TFSMs are participating in the dialog when an external input is applied; a sequential composition of TFSMs is a particular case of such composition.

Keywords: Timed Finite State Machines; input and output timeouts, timed guards, composition.

DOI: 10.15514/ISPRAS-2018-30(1)-2

For citation: Tvardovskii A.S., Laputenko A.V. On the possibilities of FSM description of parallel composition of timed Finite State Machines. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 25-40 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-2

References

[1]. Gill A. Introduction to the Theory of Finite-State Machines. 1962, 207 p.

[2]. Rita Dorofeeva, Khaled El-Fakih, Stephane Maag, Ana R. Cavalli, Nina Yevtushenko FSM-based conformance testing methods: A survey annotated with experimental evaluation. Information and Software Technology, 2010, 52, pp. 1286-1297.
[3]. M. Merayo, M. Nunez, I. Rodriguez. Formal testing from timed finite state machines, Comput. Netw. 52 (2) (2008) 432–460.
[4]. El-Fakih K., Yevtushenko N., Simão A. A practical approach for testing timed deterministic finite state machines with single clock. Science of Computer Programming, Elsevier, 2014, Vol. 80, pp. 343–355.
[5]. Maxim Zhigulin, Nina Yevtushenko, Stéphane Maag, Ana R. Cavalli. FSM-Based Test Derivation Strategies for Systems with Time-Outs. Proc. of the 11th International Conference on Quality Software, QSIC2011, IEEE, 2011. pp. 141-149.
[6]. Larsen, K.G. Testing real-time embedded software using UPPAAL-TRON: an industrial case study. Proceedings of the 5th ACM international conference on Embedded software, 2005, pp. 299–306.
[7]. Bresolin D. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. Intern Conf. GANDALF, 2014, pp. 203–216.
[8]. Villa, T., Yevtushenko, N., Brayton, R.K., Mishchenko, A., Petrenko, A., Sangiovanni-Vincentelli, A. The Unknown Component Problem. Theory and Applications. Springer, 2012, 312 p.
[9]. Laputenko, A., Gromov, M., Torgaev S., Testing alarm system implemented on STM32F407VG. Izvestija vysshih uchebnyh zavedenij. Fizika. [Russian Physics Journal]. vol. 59, issue 12, pp.174-176, 2016 (in Russian).
[10]. Gromov M., Tvardovskii A., Yevtushenko N. Testing Systems of interacting Timed Finite State Machines with the Guaranteed Fault Coverage. International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices, 2016, pp. 96–99.
[11]. Gromov M., Tvardovskii A., Yevtushenko N. Testing Components of Interacting Timed Finite State Machines. Proceedings of IEEE East-West Design & Test Symposium (EWDTS), 2016, pp. 193–196.
[12]. O. Kondratyeva, M. Gromov. To the Parallel Composition of Timed Finite State Machines. Proceedings of the 5th Spring/Summer Young Researches' Colloquium on Software Engineering, 2011, pp. 94-99.
[13]. Alexandre Petrenko, Nina Yevtushenko, Gregor von Bochmann: Fault Models for Testing in Context. FORTE 1996, pp. 163-178
[14]. Tripakis S. Folk Theorems on the Determinization and Minimization of Timed Automata. In: Larsen K.G., Niebert P. (eds) Formal Modeling and Analysis of Timed Systems. FORMATS 2003. Lecture Notes in Computer Science, vol. 2791, pp. 222-226.

Stuck-At-Faults Tester as a Web-Service

N.A. Shalyapina <nat.shalyapina@gmail.com>
A.A. Zaytsev <z.sania@mail.ru>
S.V. Batratskiy <pride080993@gmail.com>
M.L. Gromov <maxim.leo.gromov@gmail.com>
Tomsk State University,
634050, Russia, Tomsk, Lenin av., 36

Abstract. In this paper, we tell about a web-service we would like to develop. There are two goals we aim at, when developing this service. The first one is to give researchers a platform, where they could conduct preliminary experiments with different methods of test generation for digital circuits, in order to check different ideas. The second one is an opportunity to share implementations of new developed methods "on-the-fly". The web-service development procedure was splitted into three stages: the architecture design, a light version implementation and the actual implementation. This paper tells about first two stages. There are two types of web-service architectures – with monolithic kernel and with microkernel – and our architecture has the properties of both types. The intention was to have monolithic kernel, since the desired functionality is not that hard to implement. However, the property of being extensible by implementations of new methods implies that part of the functions (namely the methods implementations) should be designed as separate sub-services. The light version implementation was done for the only method: method of fault domain enumeration for the stuck-at-faults fault model. It proved that the designed architecture is viable. However, some issues with the architecture were discovered. A mechanism of on-the-fly deployment of a new method is unclear, since it is not obvious, how to satisfy possible dependences of the implementation. Also, the architecture does not follow the classical web-service design: the service has states, that should not be, if a service is intended to be the classical one. The resolution of these issues is left for the future.

Keywords. Stuck-At-Faults, Combinational Logical Circuit, Web Service, Test.

DOI: 10.15514/ISPRAS-2018-30(1)-3

For citation: Shalyapina N.A., Zaytsev A.A., Batratskiy S.V. Gromov M.L. Stuck-At-Faults Tester as a Web-Service. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 41-54. DOI: 10.15514/ISPRAS-2018-30(1)-3

1. Introduction

The modern world is hard to imagine without a variety of digital circuits. Even if a device does an analogue job, say, play music, most probably somewhere inside this device there is a digital circuit. Like any other utilitarian objects, each digital circuit is expected to perform particular work. Therefore, a problem, if this work is *done correctly*, is an actual problem. This problem appears in almost all stages of a circuit's life cycle: development, production and use.

There are many aspects of *doing a work right* by a circuit: correct functions are implemented, the circuit works quickly enough to be used, the circuit consumes limited amount of energy, etc. For example, when it comes to the correct implementation of desired functions, they usually talk about *functional testing* of a circuit [1, 2].

There are many different methods for test generation [1], and every day new ones appear (see, for example, [2]). In addition, new fault models for circuits are being developed, that can describe faults, which occur in real life, in some sense better. In such a situation, at least two questions arise. Is a new method better or worse than already existing methods? Can existing methods detect faults, introduced by a new fault model? Experiments can answer these questions. To conduct at least the preliminary experiments, a researcher should have a "rapid" access to already existing methods of test generation for digital circuits, as well as the opportunity to share with other researchers his or her new method.

Apparently, a web-service can provide such an access. However, for the best of our knowledge, a web-service that provides resources to construct tests for digital circuits using different methods does not exist. Therefore, in our research group under the leadership of Nina Yevtushenko, within the RSF project № 16-49-03012, it was decided to implement such a service.

We splitted the development of the service into several stages. The first stage is to design architecture of the service. The second one is to implement a light version of the service, to see on practice, what hardships we shall meet during the main implementation, what decisions we shall need to do and what white spots there are in the architecture. And the final stage is the actual implementation of the service.

This paper describes the first two stages of the development: architecture design and a light version implementation of the service. For the light version of the service, we have chosen a classical fault model – stuck-at-faults [1] – and a classical method of test derivation – fault domain enumeration [1]. The stuck-at-faults fault model supposes that the only faults that can occur in the circuits are shorts to the ground or to the power wire. The fault domain enumeration is the method when every possible implementation of the circuit (defined by the fault model) is enumerated and input stimuli are searched, which can show the difference between the right circuit (the specification) and the enumerated one.

The rest of the paper is organized as follows. In Section 2, the description of the designed architecture of the web-service is given. Section 3 is devoted to the light

version of the service implementation; it includes the short description of the enumeration of the fault domain method for the single stuck-at-faults fault domain. Section 4 concludes the paper. In Section 5, we provide acknowledgements for the paper support.

2. The Architecture of the Web-Service for Tests Derivation

When designing a web-service one should make two main choices: *monolithic kernel* or *microkernel* and *SOAP* [4] or *REST* [5].

2.1. Monolithic Kernel vs Microkernel

A monolithic kernel supposes, that all logics of a service are implemented as a single application, which has outside only data sources (databases). This approach promises fast and easy deployment procedure together with better performance.

On the other hand, microkernel has always been considered as a more flexible solution: each function of the big service is implemented as a stand-alone micro-service. This approach provides natural mechanisms of functionality extensions and updates (including security updates).

For our case, we have chosen somewhat hybrid approach (Fig. 1). We cannot use a monolithic kernel, since we would like to implement a possibility of on-the-fly functionality extension (adding new test generation methods on-the-fly). However, the microkernel approach is not an option either. First, rapid analysis showed, that apart of test generation methods implementations, the rest of the functionality is not that large, to be splitted into different micro-services. Second, we would not like to provide a separate database or an HTTP-server for each of micro-services.

2.2. SOAP or REST

SOAP (*Simple Object Access Protocol*) [4] and REST (*Representational State Transfer*) [5] are two possible ways, how to organize interaction between a server and a client of a web-service.

SOAP, being a protocol, works on top of HTTP and specifies how calls of remote functions should be done and how the result should be returned. It is based on XML and is able to process very complicated messages (for example, when a function returns an object, containing lists of other objects etc.). But of course, for such a flexibility there is a price: increased traffic and a need of an XML parser at both ends of the communication channel.

In the same time REST is not a protocol, it is basically a prescription, how to perform HTTP requests and how these requests should be understood at the server side. It does not require such amount of traffic or special parser as SOAP does, and it is much easier to implement. However, it is usually considered as less safe and it definitely is not that flexible as SOAP is (example with an object containing lists of objects is very tricky for REST).

At present, we have chosen REST as a way of server-client communication, but when the service grows, we suppose to introduce SOAP.

2.3. The Architecture

When all the choices are done, we are ready to present the architecture of our web-service (Fig. 1). The service consists of the following parts. *The engine*, which implements the main functionality of the service. *HTTP server*, which provides HTTP transport and the rest of HTTP functionality. A PHP program that provides *Web API* of the service. A small *Simple PHP client*, available via an Internet browser. *The database*, which keeps information about users (username, password), running jobs (process ID, start date-time, stop date-time) and existing implementations of test generation methods (ID, path-to-run). And of course a bunch of test generation methods implementations, available as *Tool scripts*.

The functionality of each element is clear from its description above, but the service engine. The engine can be implemented either as a stand-alone program or as a PHP program, combining Web API. The functions, which we consider the basic ones for our service, are as follows.

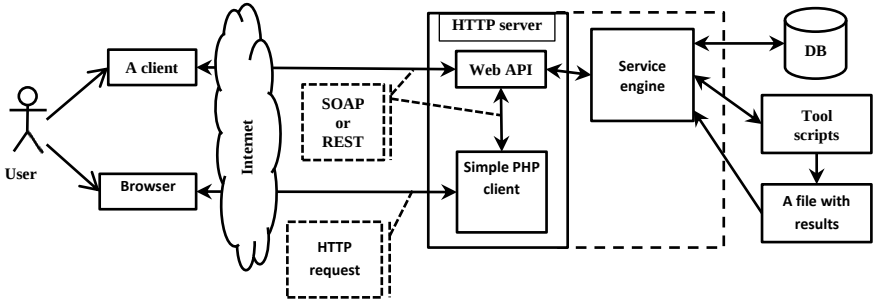


Fig. 1. The web-service architecture

1. *The authentication*. A user is asked to provide a username and a password. This function is always available; the rest functions are available only for successfully authenticated users.
2. *A job request*. A user send a request to run specific tool to build a test.
3. *A request for a list of running jobs*. A test generation is a long-term job, it takes a lot of time. For that reason, a user should have a mechanism to see, which of his/her job requests are finished, and which are not.
4. *A request for a list of available methods*. Since it is supposed, that the service will be expanded on-the-fly, the number of method may change and a user should be able to see the change.
5. *A request for a generated test*. When a job is done and the test is ready, a user should be able to get the result.

6. *A request for a new method deployment*. A user provides a program, which implements a method for a test generation he or she would like to share and the service deploys it at its side.
7. *Jobs sanitization*. A test generation is a long-term job; it can take days or even more. We suppose, that the service is not the production one rather than for a research. We limited the time of the job to complete to 24 hours. Also, a host, which runs the server, may lose the power supply. In this case, all running jobs are lost, but no records are done about this fact in the database, and the lost jobs will be marked as "in progress" forever. The jobs sanitization mechanism resolves both issues. It searches for the jobs, which are running for too long, and stops them, and looks for the lost jobs and makes certain record about this fact to the database.

Among all the functions, the deployment function seems to be the trickiest function of the service engine. The problem is that a researcher's implementation of a new method may have some dependences which may not be satisfied on the host machine, and for now we do not see, how to overcome this issue.

2.4. The Web-Service Which is not A Web-Server

One can notice that the service we describe here is not a classical web-service. It happens because a test generation procedure is a long-term process. When a user requests to build a test, the server-side may not be blocked by the request, it should stay responsive for this particular user and other users. Therefore, the server-side changes its state from "ready for requests" to "ready for requests and running one job in the background". Then "… two jobs in the background", then "… two jobs in the background, but another ones, and those two are ready". And so on. The classical approach supposes that the server-side is stateless [6].

However, apart from the described issue, the rest of our design follows the classical web-service design and therefore we tend to call it a web-service. In the future, we would like to rethink on this issue.

3. Light Version of the Web-Service

In this Section, we describe an implementation of a light version of the web-service. First, we describe the simplifications made. Then we describe the stuck-at-fault fault model and the fault domain enumeration method, which we have chosen for the implementation. And at last we give some implementation details.

3.1. Simplifications

We decided to implement only a part of the service engine functions, namely: "The authentication", "A job request", "A request for a list of running jobs", "A request for a generated test", and partly "Jobs sanitization" (only the time to complete restriction).

The web API is abandon. The engine and the simple PHP client should be implemented as one PHP program. However, the "Jobs sanitization" and "A job request" are shared between the engine and the tool script. And the tool script is a shell-script which implements the fault domain enumeration method. In the light version it is the only method implemented, that is why instead of several tool scripts we have only one.

According to the selected engine functions, the database needs only two tables (entities): the user information table and the job information table.

3.2. The Fault Model and the Method

3.2.1. Logical Circuits

There are two main classes of digital circuits: combinational circuits and sequential circuits. The first ones are usually referred as memoryless circuit without feedbacks and the second ones are the rest circuits [1, 7]. A digital circuit can be seen from different perspectives (behavioural, structural and physical) and at different level of abstraction (system, register, logical, schematic) [1, 7]. From the point of view of the service that we design, it does not matter, which kind of a digital circuit or from what perspective and at what level is given, everything depends on a test generation method implementation (a tool script) being called, whether it can or not to handle the call. For the light version of the service, we have chosen *combinational circuits*, considered from the *behavioural* and *structural* perspectives at the *logical* level, tested against the *stuck-at-faults* fault model using the *fault domain enumeration method*.

3.2.2. Behaviour and Structure of a Digital Circuit

Behavioural model at the logical level of a combinational digital circuit with n input pins and m output pins is defined as an *ordered system of m Boolean functions*, mapping Boolean vectors of the length n into the Boolean set [1, 2, 7]: $f_i(x_1,.., x_n) \in \{\mathbb{B}^n \to \mathbb{B}\}$, where $\mathbb{B} = \{0, 1\}$ is the Boolean set, $i \in \{1, \ldots, m\}$. Later on in this paper, we shall omit the word "ordered", but will always mean that, if the opposite is not specially denoted.

Two ordered system of Boolean functions $f_i(x_1,.., x_n)$ and $f'_i(x_1,.., x_n)$, $i \in \{1, \ldots, m\}$, are equivalent, if for each $i \in \{1, \ldots, m\}$ it holds, that $f_i = f'_i$ (corresponding Boolean functions are equivalent).

A structural model at the logical level of a combinational digital circuit is called a *logical combinational circuit*. *A logical combinational circuit (or just a circuit)* for a combinational digital circuit with n input pins and m output pins is a directed acyclic graph [2] $G = \langle V, E \rangle$, where V – is non-empty, finite set of *nodes*, $E \subseteq V \times V$ – is a set of ordered pairs $\langle v_1, v_2 \rangle$ called *edges*, and every node $v \in V$ is either *an input pole* or *a gate* or *an output pole*. Poles correspond to the pins of the digital combinational circuit. There should be exactly n input poles and m output poles. An

input pole has no ingoing edges, and at least one outgoing edge. An output pole has no outgoing edges and exactly one ingoing edge. A gate has at least one ingoing and at least one outgoing edge and is associated with some Boolean function g: $\mathbb{B}^k \to \mathbb{B}$, where arity k equals to the number of ingoing edges for this gate.

A logical circuit with n input poles and m output poles maps Boolean vectors of length n into m Boolean values. This mapping is computed as follows. Let there is a vector $\langle \alpha_1 \ldots \alpha_n \rangle \in \mathbb{B}^n$ (*an input vector*). Before start, if there are any mappings of the circuit's edges and vertices, drop these mappings. When a node is mapped to some Boolean value, its outgoing edges are also mapped to this very value. For each $j = 1,\ldots,n$ the input pole j is mapped to the value α_j. When every ingoing edge of a gate is mapped to some value γ_l, the node becomes mapped to the value $g(\gamma_1, \ldots, \gamma_k)$, where g is the function associated with the gate. Due to our definition of the logical circuit, each output pole i eventually is mapped to some Boolean value β_i. So, at some point we are able to collect *an output vector*

$$\langle \beta_1 \ldots \beta_m \rangle = \langle f_1(\alpha_1, \ldots, \alpha_n), \ldots, f_m(\alpha_1, \ldots, \alpha_n) \rangle \in \mathbb{B}^m.$$

The fact, that the circuit G maps vector $\langle \alpha_1 \ldots \alpha_n \rangle$ to the vector $\langle \beta_1 \ldots \beta_m \rangle$ we shall denote as $G(\langle \alpha_1 \ldots \alpha_n \rangle) = \langle \beta_1 \ldots \beta_m \rangle$.

Therefore, every logical circuit is associated with a system of Boolean functions $f_i(x_1,.., x_n)$, $i \in \{1, \ldots, m\}$. This system is a behavioural model of the underlying digital circuit. Apparently, that every logical circuit has the only corresponding system of Boolean functions, and we say that the logical circuit *implements* the system of the Boolean functions. However, every system of Boolean functions may have several corresponding logical circuits (structural implementations).

3.2.3. Tests and Fault Models

Two logical circuits G and F are *equivalent* if they implement the same system of Boolean functions. This fact we shall note as $G \cong F$. The non-equivalence of two circuits G and F we shall denote as $G \ncong F$.

Let a logical circuit be given. They say, that there is *stuck-at-zero (stuck-at-one) fault* [1] at the edge $e = \langle v_1, v_2 \rangle$ if this edge is constantly mapped to *zero* (*one*), regardless from the any other mappings. Stuck-at-zero (stuck-at-one) fault means, that node v_2 is virtually associated with Boolean function $g(x_1, \ldots, x_{l-1}, 0, x_{l+1},\ldots, x_n)$ (with function $g(x_1, \ldots, x_{l-1}, 1, x_{l+1},\ldots, x_n)$), where l is the index of the edge e and $g(x_1, \ldots, x_n)$ is the Boolean function actually associated with the node v_2. One can consider the case, when several edges are faulty in the circuit, but for the purpose of this paper, we consider only single stuck-at-faults.

A fault model is a triple $\langle S, \cong, \Omega \rangle$, where S – is a logical circuit (usually called *a specification*), describing a correct behaviour of a system; Ω – is a set (usually called *a fault domain*), containing logical circuits, which are considered as possible (correct and incorrect) implementations of the system of Boolean functions defined by S; and $\cong$ – is the equivalence relation. Every circuit from Ω has the same numbers of input and output poles as S does.

A test case t for the fault model $\langle S, \cong, \Omega\rangle$ is a Boolean vector $\langle \alpha_1 \ldots \alpha_n\rangle \in \mathbb{B}^n$, where n – is the number of the input poles of the circuit S. *A test T* for the fault model $\langle S, \cong, \Omega\rangle$ is a finite set of test cases for the fault model $\langle S, \cong, \Omega\rangle$. A test T for the fault model $\langle S, \cong, \Omega\rangle$ is called *sound* if for each circuit $G \in \Omega$ such that $G \cong S$ and for each $t \in T$ it holds, that $G(t) = S(t)$. A test T for the fault model $\langle S, \cong, \Omega\rangle$ is called *complete* if for each circuit $G \in \Omega$ such that $G \not\cong S$ there exists $t \in T$ such that $G(t) \neq S(t)$. A test T for the fault model $\langle S, \cong, \Omega\rangle$ is called *exhaustive* if it is sound and complete.

In this work, we consider a fault domain, which contains S and every possible circuit got from S by introduction every possible single stuck-at-zero or stuck-at-one fault. The fault model with this fault domain is called *the stuck-at-faults fault model*. Since S is finite graph with finite number of edges, the fault domain in this case is finite. Namely, the number of circuits in fault domain equals to $2\cdot|E|$, where E is the set of edges.

3.2.4. Miter

Given two (ordered) systems of Boolean functions $f_i(x_1,.., x_n)$ and $f'_i(x_1,.., x_n)$, $i \in \{1, \ldots, m\}$. A *miter* for them is the following function

$$f_M = (f_1(\boldsymbol{x}) \oplus f'_1(\boldsymbol{x})) \vee \ldots \vee (f_m(\boldsymbol{x}) \oplus f'_m(\boldsymbol{x})),$$

where $\boldsymbol{x} = \langle x_1 \ldots, x_n\rangle$ is the vector of arguments. Note, that the order of the functions in the systems is important.

Proposition. *Two systems of Boolean functions* $f_i(x_1,.., x_n)$ and $f'_i(x_1,.., x_n)$, $i \in \{1, \ldots, m\}$ *are not equivalent if and only if there exists such a vector* $\boldsymbol{\alpha} \in \mathbb{B}^n$ *that* $f_M(\boldsymbol{\alpha}) = 1$, *where* f_M *– is the miter for the give systems.*

If there exists such a vector $\boldsymbol{\alpha} \in \mathbb{B}^n$ that $f_M(\boldsymbol{\alpha}) = 1$, then the miter is called *satisfiable* and $\boldsymbol{\alpha}$ is called *satisfying vector*. In contrary, if for any $\boldsymbol{\alpha} \in \mathbb{B}^n$ $f_M(\boldsymbol{\alpha}) = 0$, then the miter is called *unsatisfiable*.

Of course, a miter can be built for logical combinational circuits as well. A miter for logical circuits G and G' – is a circuit M, which we get by matching input poles of G to corresponding input poles of G' (v_1 to v'_1, v_2 to v'_2 etc.), so we have n input poles. Then we combine every output pole u_i of G with corresponding output pole u'_i of G' by XOR gate, and then combine all outgoing edges of those XOR gates by disjunction gate. And at last outgoing edge of the disjunction gate comes to only output pole u of M. Now, if we compute mapping, done by the structural miter we shall get the same function f_M.

3.2.5. The Method of Fault Domain Enumeration

The idea of the method is quite simple. Let the stuck-at-faults fault model $\langle S, \cong, \Omega\rangle$ be given. Since for the case of stuck-at-faults fault model the set Ω is finite, and its cardinality is polynomial on the edges number of S, we can enumerate it. At the beginning, a test under construction T is empty. Get the next circuit G from Ω.

Check, whether T contains such a vector $\boldsymbol{t}$ that $G(\boldsymbol{t}) \neq S(\boldsymbol{t})$. If it does, take the next circuit from Ω and repeat. If it does not, build a miter for G and S and check, whether it is unsatisfiable. If it is not (i.e., satisfiable), then take a counter example for unsatisfiability as a test case and put it in T.

Proposition. *The above procedure provides an exhaustive test.*

There are some notes to the procedure above. First, it does not guarantee that the resulting test is minimal on number of test cases. But for the goals of this work we do not aim at getting optimal tests. Second, the problem of unsatisfiability check is not an easy task by itself [8]. It is known to be NP-complete and may bring a lot of headache in general case. Fortunately, this problem is quite popular among researches and many powerful tools were developed to solve it, which perform well in most real life cases. For example, see works on MiniSAT [9].

3.3. Implementation Details

3.3.1. Operating System, HTTP Server, DBMS, and the Engine

The available to us host machine, where we can deploy the service, runs under FreeBSD operating system. It has MySQL database management system and Apache HTTP server with the support of PHP installed. Therefore, we had no need of choosing the right environment for our web-service. The triple Apache+MySQL+PHP proved to be the reliable basis for any project of any level of complexity.

The engine of the light version of the web-service was implemented in PHP. As it was noted earlier, we abandon the web API, and the small PHP client was implemented together with the engine as one program.

3.3.2. The Method Implementation

To implement the method of fault domain enumeration for the stuck-at-faults fault model as a tool, we use the tool called ABC [10]. This tool has built-in instruments for a miter construction and checking the miter for being unsatisfiable. If the miter is not unsatisfiable, ABC provides a counter example.

The only thing we implemented in addition to ABC is an instrument for faulty circuit's enumeration. This instrument is implemented as a stand-alone program, which takes a logical circuit (specification) and a folder to store the result as an input and saves files with faulty logical circuits in the designated folder.

Since the chosen fault model describes faults in the structural terms (in terms of logical circuits), the method implementation requires digital circuits under test be given as logical circuits. ABC understands logical circuits in ISCAS'89 format (also known as `.bench` format). This format simply describes directed graph. For example, a code like this

```
INPUT(x1)
```

```
INPUT(x2)
OUTPUT(y)
y = AND(x1, x2)
```

states, that there are two input poles, named `x1` and `x2`, one output pole, named `y`, and one gate associated with the conjunction (AND), but without a name. Let call this gate *g*. Then the edges are ⟨`x1`, *g*⟩, ⟨`x2`, *g*⟩ and ⟨*g*, `y`⟩.

To finish the method implementation, the faults enumerator and ABC should be called in the right order with the right parameters. This is done by a shell-script, which in terms of the architecture of the web-service we call *the tool script*.

3.3.3. Jobs sanitization

The most of the functions implemented in the engine of the light version of the web-service are quite straightforward and we shall not describe them. However, the implemented part of the "Jobs sanitization" (namely, the time to complete restriction) and "A job request" functions are worthy of mention and description.

When the engine gets the request from a user to start a new job of test derivation, it runs the script and delegates all interactions with the database to the script. The script makes all necessary records in the database: the date-time of the script start, the ID of the user, who requested this job, process ID of the job. When the script finishes the task, it again makes necessary records about that: the task was normally finished, when the task was finished.

The implementation of the time to complete restriction is done with the use of `timeout` utility built into FreeBSD [11]. This utility starts the specified program, waits for specified amount of time and if the program is still running, it sends to the program the termination signal. The script catches this signal and before termination, it makes a record into the database, that certain task was artificially stopped. That is the engine does not run the script directly, it runs the `timeout` utility, specifying that the script should be run and should finish within 24 hours. The rest is done by the `timeout` utility.

This approach to implementation of the "A job request" and "Jobs sanitization" functions helped to guarantee the responsiveness of the web-service whilst the requested jobs are run in the background and successfully sanitized.

4. Conclusion

In this paper, we presented architecture of the web-service we would like to develop. The main goal of the web-service is to give for researchers a platform, where they can do preliminary rapid experiments with different test generation methods for digital circuits. And, as the second feature of the web-service, researchers can share an implementation of a new method they developed.

The architecture has properties of both a microkernel service and a monolithic kernel services. Also, the analysis of the architecture shows, that in its current state

it is not the classical architecture of a web-service, because web-services are supposed to be stateless, and our service by design has states. This issue needs some further research.

Another issue that needs to be resolved in the future is the deployment of new methods implementations on the service. For now, it is not clear, how to guarantee possible dependences.

As the second stage of the development of the web-service, we implemented the light version of the service. We took only minimal necessary functionality, and the only test generation method – the fault domain enumeration method for the stuck-at-faults fault model for combinational logical circuits. This implementation proved that the designed architecture is viable. In addition, the implementation showed, that some functions of the service engine (like the "Jobs sanitization"), probably, are better be moved away from the engine.

Acknowledgements

This work is partially supported by the grant of the Russian Science Foundation № 16-49-03012. The authors would like to thank Andrey Laputenko for presenting the result of the paper at the ESRCT 2017 workshop. In addition, the authors would like to thank the reviewer of the paper, whose constructive comments helped to make this paper better.

References

[1]. Skobcov Yu.A., Skobcov V.Yu. Logical modeling and testing of digital devices. Doneck: IAMM NAS of Ukraine, DonNTU, 2005. 436 p. (in Russian)

[2]. Zakrevskij A.D., Pottosin Yu. V., Cheremisinova L.D. Fundamentals of logic design. Minsk: UIIP NAN of Belorus, 2006. 254 p. (in Russian)

[3]. Chernov A.V., Sergeeva E.A. Autocorrelational testing of digital combinational circuits. Sovremennye problemy nauki i obrazovaniya [Modern problems of science and education], 2013, № 6 (in Russian)

[4]. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). https://www.w3.org/TR/soap12/, 05.02.2018.

[5]. Wilde E., Pautasso C. REST: From Research to Practice. Springer Science & Business Media, 2011. 528 p.

[6]. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures. Chapter 5. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 05.02.2018.

[7]. Harris D., Harris S. Digital Design and Computer Architecture. Morgan Kaufmann, 2012, 712 p.

[8]. Silva L.G. e, Silveira L.M. and Marques-Silva J.P. Algorithms for Solving Boolean Satisfiability in Combinational Circuits. In Proceedings of DATE'99, 1999, pp. 526-530.

[9]. Niklas Eén, Niklas Sörensson. The MiniSAT. http://minisat.se/Main.html, 16.10.2017.

[10]. ABC: A System for Sequential Synthesis and Verification. https://people.eecs.berkeley.edu/~alanmi/abc/, 16.10.2017.

[11]. FreeBSD Manual Pages. timeout. https://www.freebsd.org/cgi/man.cgi?query=timeout&sektion=1, 05.02.2018.

Тесты на константные неисправности как веб-сервис

Н.А. Шаляпина <nat.shalyapina@gmail.com>
А.А. Зайцев <z.sania@mail.ru>
С.В. Батрацкий <pride080993@gmail.com>
М.Л. Громов <maxim.leo.gromov@gmail.com>
Томский государственный университет,
634050, Россия, Томск, пр. Ленина, 36.

Аннотация. В этой статье рассказывается о разрабатываемом нами веб-сервисе. Разрабатывая этот сервис, мы преследуем две цели. Первая – предложить исследователям платформу, где они могли бы проводить предварительные эксперименты с различными методами генерации тестов для цифровых схем, для проверки различных идей. Вторая – возможность «на лету» поделиться реализациями новых методов. Процедура разработки веб-сервиса была разделена на три этапа: дизайн архитектуры, реализация облегчённой версии и фактическая реализация. В этой статье рассказывается о первых двух этапах. Есть два типа архитектур веб-сервисов – с монолитным ядром и микроядром – и наша архитектура обладает свойствами обоих типов. Мы стремились к тому, чтобы получить монолитное ядро, поскольку желаемая функциональность не так уж трудно реализовать. Однако расширяемость реализациями новых методов подразумевает, что часть функций (а именно, реализации методов) должны быть разработаны как отдельные под-сервисы. Реализация легкой версии была выполнена для единственного метода: метода перебора области неисправности для модели константных неисправностей. Она показал, что разработанная архитектура жизнеспособна. Однако были обнаружены некоторые проблемы с ней. Механизм развертывания добавляемого «на лету» метода неясен, так как неясно, как удовлетворить возможные зависимости реализации. Кроме того, архитектура не соответствует классическому дизайну веб-сервиса: у сервиса есть состояния, которых не должно быть, если сервис классифицирован как классический. Решение этих вопросов остается на будущее.

Ключевые слова. Константные неисправности, комбинационные схемы, Web сервис, проверяющие тесты.

DOI: 10.15514/ISPRAS-2018-30(1)-3

Для цитирования: Шаляпина Н.А., Зайцев А.А., Батрацкий С.В., Громов М.Л. Тесты на константные неисправности как веб-сервис. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 41-54. DOI: 10.15514/ISPRAS-2018-30(1)-3

Список литературы

[12]. Скобцов Ю.А., Скобцов В.Ю. Логическое моделирование и тестирование цифровых устройств. Донецк, ИПММ НАН Украины, ДонНТУ, 2005,436 с.

[13]. Закревский А.Д., Поттосин Ю.В., Черемисинова Л.Д. Fundamentals of logic design. Минск, Национальная академия наук Беларуси, 2006, 254 с.

[14]. Чернов А.В., Сергеева Е.А. Автокорреляционное тестирование цифровых комбинационных схем. Современные проблемы науки и образования, 2013, № 6
[15]. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). https://www.w3.org/TR/soap12/, 05.02.2018.
[16]. Wilde E., Pautasso C. REST: From Research to Practice. Springer Science & Business Media, 2011. 528 p.
[17]. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures. Chapter 5. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 05.02.2018.
[18]. Harris D., Harris S. Digital Design and Computer Architecture. Morgan Kaufmann, 2012, 712 p.
[19]. Silva L.G. e, Silveira L.M. and Marques-Silva J.P. Algorithms for Solving Boolean Satisfiability in Combinational Circuits. In Proceedings of DATE'99, 1999, pp. 526-530.
[20]. Niklas Eén, Niklas Sörensson. The MiniSAT. http://minisat.se/Main.html, 16.10.2017.
[21]. ABC: A System for Sequential Synthesis and Verification. https://people.eecs.berkeley.edu/~alanmi/abc/, 16.10.2017.
[22]. FreeBSD Manual Pages. timeout. https://www.freebsd.org/cgi/man.cgi?query=timeout&sektion=1, 05.02.2018.

Towards the methods of analysis malicious applications for Android operating system

Sergey Staroletov <serg_soft@mail.ru>
Polzunov Altai State Technical University,
Lenin avenue 46, Barnaul, 656038, Russia

Abstract. It is considered to the problem of analysis of Android applications to study a malicious behaviour. The methods of analysis are presented, the general method, which combines different analysis techniques (static, dynamic, decompilation, debugging, logging) is proposed, and information of our software based on it is given.

Keywords: analysis of program behavior; malicious software; data leaks; Java; software and technical expertise

DOI: 10.15514/ISPRAS-2018-30(1)-4

For citation: Staroletov S.M. Towards the methods of analysis malicious applications for Android operating system. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 55-68. DOI: 10.15514/ISPRAS-2018-30(1)-4

1. Necessity for the work

In the recent years, mobile phones with a mobile operating system are widely being used, and now we observe the explosive growth of mobile devices in the world.

According to forecasts [1], in 2018 the number of smartphones in the world will be over 50% of the total number of phones (see fig. 1).

More and more people do their daily tasks by using software for phones, including making the financial transactions.

However, the computer literacy for today's smartphone users is not keeping pace with the progress in the field of mobile devices. It can mean that soon we are going to have about 2.5 billion potential victims of intruders; they would use mobile phones as an instrument to steal private data and money.

It is known that in the past year hackers in Russia were stolen 349 million rubles [2] from the owners of phones under Android OS (about 2 rubles in mean from each Russian citizen), which is five times more than a year ago.

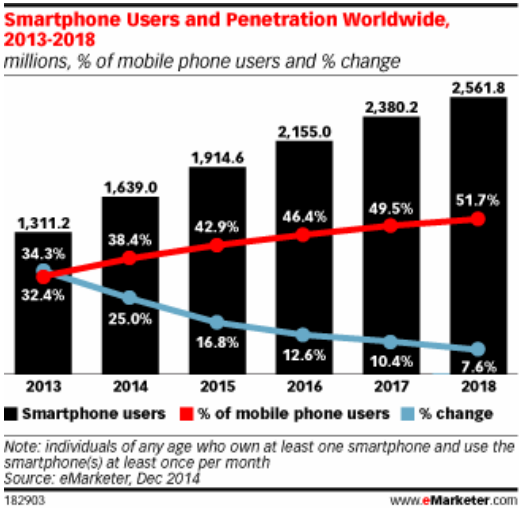


Рис. 1. Количество смартфонов в мире удвоится за последние 5 лет [1]
Fig. 1. Number of smartphones in the world will be doubled for the last 5 years [1]

Choosing the Android OS by intruders is primarily caused by its prevalence, availability of cheap Chinese phones under it.

Here we have a problem: typical users (especially from the countryside) are not even aware that they work with a minicomputer which can hook a trojan program and such program could get access to user data, receive commands to work following the hacker's request.

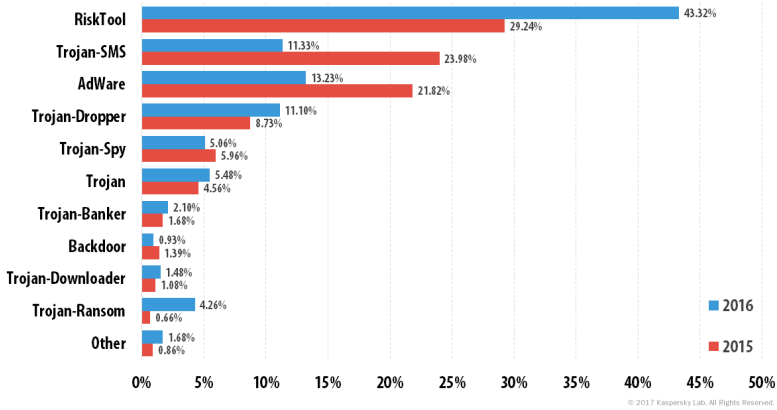


Рис. 2. Распределение нового вредоносного программного обеспечения по типу в 2015 и 2016 годах (Касперский)
Fig. 2. Distribution of new mobile malware by the type in 2015 and 2016 (Kaspersky)

According to a Kaspersky's report (see fig. 2), the most frequent malware types now are “RiskTools” (special software or software with vulnerabilities used by intruders to enter and control the system), “Trojan-SMS” (trojans for sending SMS to short numbers) and some other trojan types including “Trojan-Banker” (trojans for stealing money from a bank account).

In 2015, local representatives of the Police addressed to the Department of Applied Mathematics at Altai State Technical University in order to develop methods, algorithms, and software for the analysis of the smartphones injured by the trojans (like types as “SMS” and “Bankers”), to do digital forensic science to analyze the malicious applications. We have a situation now: there are no generally accepted methods to do it.

2. The goals of the analysis

Android OS itself is a special Linux Kernel, which contains Dalvik virtual machine for execution of Java applications using their own API.

The application for Android is an apk (Android Package) file, which is a zip archive and it contains some compiled Java-classes of the application in the form of .dex file (Dalvik EXecutable), resources and application's descriptor AndroidManifest.xml within.

A trojan here – is a dual-purpose Android application. It looks like an ordinary application but it was created to steal data or money from victims.

A victim – is a people who handed his smartphone with some trojans to the Police for the analysis since he had suffered by unknown intruder's actions to withdraw his money via the phone.

According to the problem described in the previous section, the goals of the analysis are:

- identify malicious applications among given applications from given phones;
- make some proof of harmfulness of given application;
- study algorithms of the work of applications without having sources of it;
- find the remote hosts which application communicates to, discover sending content, format and protocols);
- discover user's private data leaks;
- compare various malicious applications and group them;
- develop and test a general method to make suchlike analysis.

So, we need dynamic and static methods for analysis the behavior of Android applications. About 50 real recent malicious applications from the real victims (who had lost the money because of intruder's actions) were analyzed and identified, and methods for it were developed.

3. Related work

There is a high number of research papers devoted to malicious applications for Android but most often to identify whether or not the given application is a trojan. For example, in the work [3] authors explain how to detect a malicious program for Android with using machine learning and static analysis methods. But it is only a part of current research, the main our goal is to make the analysis of the behaviour and prove the harmfulness by the application's actions. The fact of malware can be detected by using VirusTotal tool [5], which checks the signature of given application by a lot of modern anti-viruses.

The Pennsylvania State University, Duke University and Intel lab have developed the series of patches [4] for Android Kernel and libraries to track the data passing through the functions calls in running Android applications to prevent privacy leaks (taints). This approach can be applied to our research with some changes to do dynamic analysis of application's behavior.

4. The analysis

4.1 Malicious applications identifying

The detection of malicious applications in a phone is not so difficult. As a rule, such dual-purpose applications have a small size, a name and an icon disguised as the popular applications (WhatsApp, Skype, Flash Player, Kaspersky, Sberbank Online, etc.).

Since the majority of identified by this research viruses carry out sending messages in a background to short phone numbers or performing USSD-requests, it is possible to identify them by looking at the permissions of applications.

If the Android app uses some API, it must request the appropriate permissions from the system. As a rule, such a request is made during the installation of the application; however, users just ignore the warnings, even if they say that the application will send SMS that can cost money or it is going to control the phone as an administrator.

Permissions are set in AndroidManifest.xml and can be seen in the information about an application. An example of malicious application's permissions is given on table 1.

Табл. 1. Разрешения приложения Android
Table 1. Android application's permissions

android.permission.INTERNET
android.permission.WRITE_SMS
android.permission.READ_SMS
android.permission.SEND_SMS
android.permission.RECEIVE_SMS

The application can be sent for an analysis to one of the anti-viruses (for example, DrWeb online), as well as viewed in the Virus Encyclopedia for its possible actions. Now, we are using the online tool VirusTotal [5], which accumulates information about the signatures of malware files from various antivirus software. However, applications are modified, and some recent viruses could not be recognized, so this analysis is made for educational purposes only.

When the malicious application is detected, a further analysis on the phone would be inappropriate. As a rule, the application as an apk file is retrieved, and the browser's logs and download history in the phone are analyzed to search from where such application could be obtained.

Next, the application is analyzing on a desktop computer running Android OS emulator (included in the Android Studio application development tools). Application analysis is complicated because we deal with a compiled application without the presence of the source code.

4.2 Low-level debugging the Android applications

Without the presence of the source code, there is a way to understand the logic of the application – to analyze it with using a debugger. Starting with version 6.6, an interactive debugger IDA Pro [6] supports debugging applications for Dalvik virtual machine.

It is possible to trace the logic of the application, explorer function calls from a Java library, set breakpoints on them and analyze the data transferred as parameters, etc.

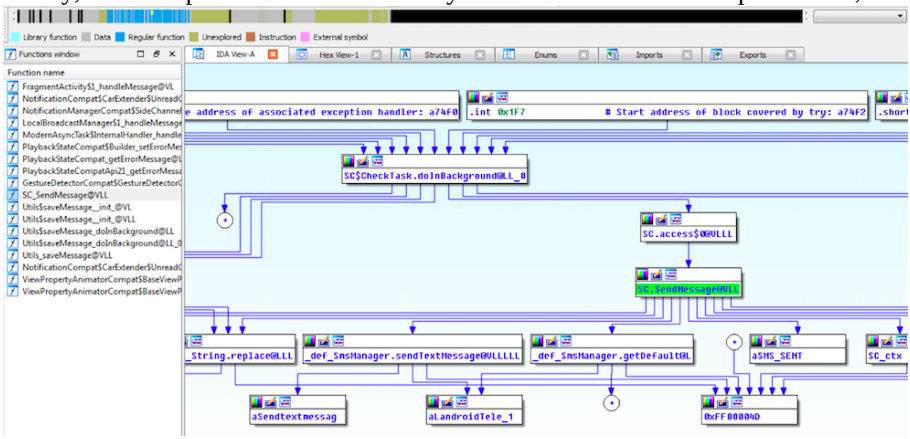


Рис. 3. Исследование графа поведения в IDA
Fig 3. Discovering a behaviour graph with IDA

With using IDA we can use a feature "proximity browser" for some kind of static analysis (see fig. 3). Once .dex file is loaded, the app creates a graph containing

functions, variables, other objects and the relationships between them. Without actual running the debugging it is possible to find a suspicious function call (like sending the SMS) and view a path for its execution.

With using the debugger we can identify, for example, process of interaction with an intruder's server (in this case, addresses and sending data were not explicitly programmed in the code but collected in the values of local variables), and this approach was used to prove the harmfulness work with SMS messaging with a remote banking in one trojan application.

However, using this method of analysis requires knowledge of the virtual machine assembler and approaches of low-level debugging, the process of analysis is long and complicated. In addition, IDA Pro is a paid product.

The future research work may be applied here to create own solution to detect calls from application's disassembled code started from some usable functions from Android API liked by the intruders (send and receive SMS, networking, etc) and programmable create a graph to do future analysis.

4.3 Analysis of network interactions in the Android applications

To determine what data is passed over the network to/from the analysing application, we can propose a very simple way – to install a proxy server on a host that is running the phone emulator, and make the settings in the Android emulator phone instance to specify the address and port of the proxy server.

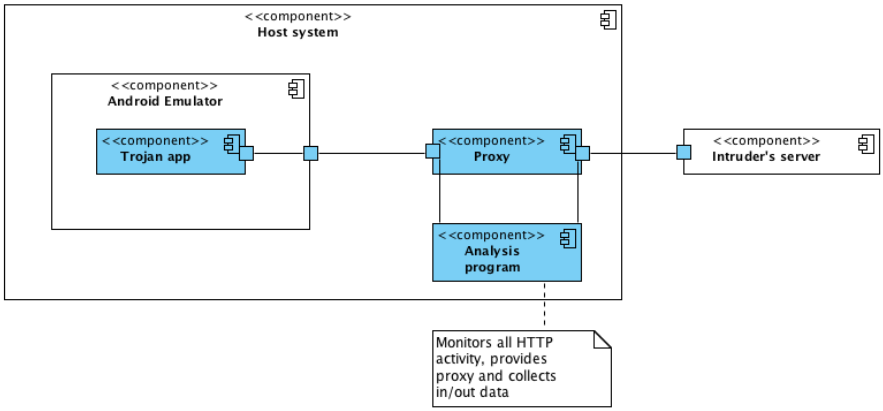


Рис. 4. Прокси-сервер позволяет выполнять мониторинг трафика
Fig. 4. A proxy server offers to monitor the traffic

After that, all the HTTP connections will go through a proxy, which may log us information is sent and received by the applications on the phone (the process is shown on fig. 4). As a result, we can identify intruder's domains, through which the trojan applications receive data and transmit the control commands.

Having the recorded network data, we also can match various versions of the malware, identifying the groups of trojans by comparing the remote hosts and interactions data.

4.4 Tracing the application's logic with a patched OS kernel

To understand the logic behavior of Android applications, it is decided to use the method of dynamic analysis with logging tools built into Android.

However, the standard "adb logcat" command creates a very sparse dump of the application's operations during the run and not intended for the analysis.

The TaintDroid project [4] was created to tell the user about private data leaks during the work of Android OS applications in the form of push notifications on the top of a phone screen. It patches Android kernel and library code and inserts the own code to reveal and process taint data.

They had to do a lot of work to pass some additional arguments between every library function call in order to collect the data (a cut from the patch for the file vm/InlineNative.cpp is shown on table 2).

Табл. 2. Фрагмент патча TaintDroid [4]
Table 2. A fragment of the TaintDroid [4] patch

```
@@ -291,8 +310,13 @@ bool javaLangString_compareTo(u4 arg0, u4 arg1
 /*
  * public boolean equals(Object anObject)
  */
+#ifdef WITH_TAINT_TRACKING
+bool javaLangString_equals(u4 arg0, u4 arg1, u4 arg2, u4 arg3,
+    u4 arg0_taint, u4 arg1_taint, struct Taint* rtaint,
JValue* pResult)
+#else
 bool javaLangString_equals(u4 arg0, u4 arg1, u4 arg2, u4 arg3,
     JValue* pResult)
+#endif /*WITH_TAINT_TRACKING*/
```

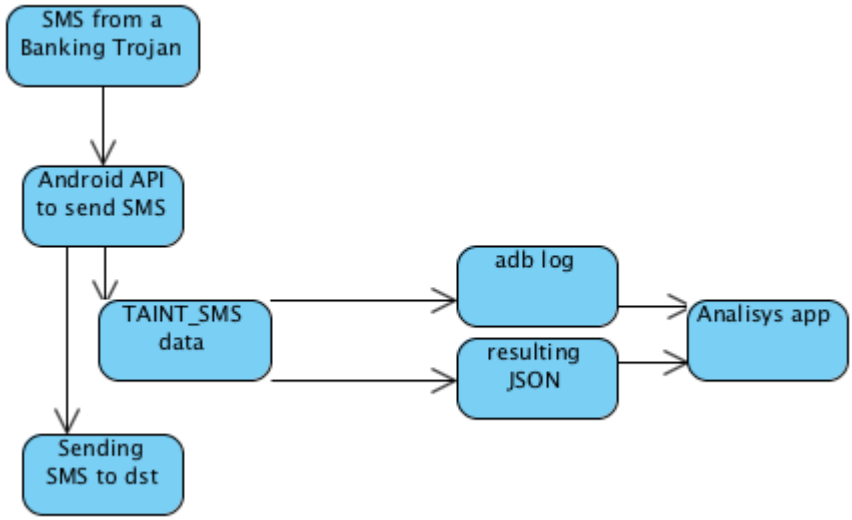


Рис. 5. Пример пути СМС с использованием DroidBox
Fig. 5. An example of an SMS flow with using DroidBox

After doing it they have inserted their code to check the fact of opened files, network access, used encryption algorithms and data pass to encrypt/decrypt functions, sending and receiving SMS (a number and a text of the message), phone calls from applications, class loading into memory – just what is necessary for the analysis of malicious code.

Next, we found Droidbox [7] – a framework for the dynamic analysis of applications for Android, which is built on the top of TaintDroid's patches and consists of two parts:

- a Python script to start and produce results;
- a modified OS kernel, containing TaintDroid's patches and additional patches for improving the logging tools.

These patches work as follows: because of some added additional lines to the OS code to preserve TaintDroid data to the log (see fig. 5), information of applications' behavior now can be accessible with 'adb logcat' command and can be processed with some external software.

It should be stressed that some viruses perform checks the OS version of Android, the phone name, phone number, the IMEI, in order to determine the work in the emulator with standard predefined values and possible to stop working after such detection. So, we have to use one's own binary patches to the system image with the replacement of the required parameters with some new values without having to recompile the entire kernel.

4.5 Decompilation and source code analysis

If the log during the time of the analysis of the application shows us no activity (for example, the server that receives a command is not working now), we can try to decompile the application code to observe the logic.

As every Java application could be decompiled to a source code (especially if the application developer did not use any obfuscation algorithms after building the code), we can try to decompile the application and get some semblance of its source code. For such decompilation firstly we apply dex2jar program [8] (to convert from a.dex archive inside of an .apk file to a set of .class files containing compiled Java code), and Java decompiler program (a tool for process .class and obtain .java files from them).

Because of a difficult Java structure, currently there is not exist a good decompiler for all cases, so we propose to use some decompilers simultaneously (FernFlower [9], CPR [10], Jadx [11], Procyon [12]), and to select the best result after decompiling.

The results of decompiling to a source code can be used against intruders to prove the harmfulness of the application.

However, if the application file structure is just restored (though not their contents), then it can also be used for comparing the trojans from the various crime groups.

The process of restoring normal application code and understanding what it does – is a handwork, but not a complicated enough. The issue of combating obfuscation requires special studies. For example, this is a fragment of decompiled code that possible had been obfuscated before distributed, it is unreadable and currently the method of source code observing is not used for such files:

Табл. 3. Фрагмент декомпилированного обфусцированного кода
Table 3. A fragment of decompiled obfuscated code

```
var1_5 = var9_1[0]
var9_1 = Class.forName(Application.onCreate("\ub559\u4623
\u069b\u920c\u5140\u162b\u13b7\u927b\uff0c\ubf32\u1c91
\u4ae2"))
var12_9 = var9_1.getConstructor(new Class[]{var9_1, String.class})
var10_11 = Class.forName(Application.onCreate("\ub552
\u462c\u0689\u921f\u5101\u162b\u13bc\u927b\uff29\ubf34
\u1c93\u4af3\u4a06\u3b4a\u4b63\u895b\u8aa6\u19b8\u62da
\uecc1\u2174\u912e\uc452"))
var14_12 = var10_11.getMethod(Application.onCreate
("\ub554\u4627\u0699\u9229\u5107\u1630"), new
Class[]{String.class, Integer.TYPE})
```

4.6 The general method

The proposed general method is given on fig. 6. Suppose we have a phone with some possible trojans. We look at the applications, get them from the phone and select them to make the analysis by some indirect indications (like name, icon, size, permissions). And possible we try to check some applications in the antivirus databases.

If we found a suspicious application, we are going to make a comprehensive analysis. The app in binary classes form could be checked by a static analyzer for possible security vulnerabilities (currently the static analyzer has not been selected yet). After, the app goes to the decompilation, and if the result of it is good the code behavior can be observed from the sources.

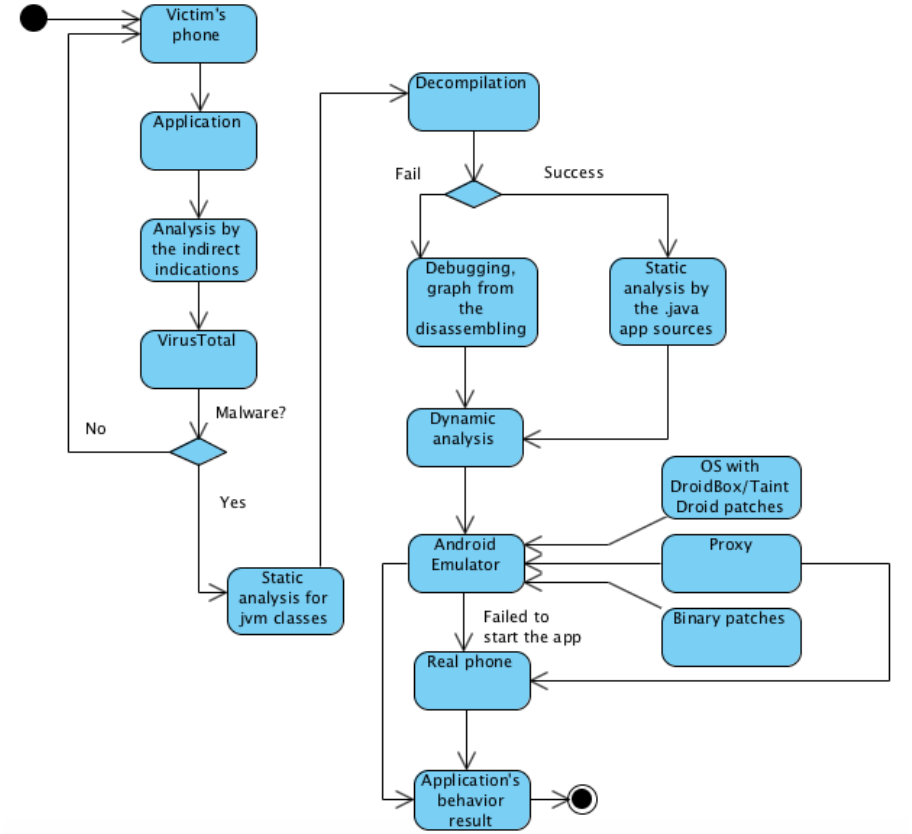


Рис. 6. Предлагаемый процесс анализа
Fig 6. Proposed process of analysis.

In addition, sources could be statically analyzed by the static analyzer, which works with .java sources. If the decompilation has failed, the app goes to the debugger and the behavior graph, which is received after disassembling is observing.

For analysis the dynamic behavior we use the emulator with a modified OS kernel for logging, it is patched to change the IMEI, phone and so on, and proxy to detect a network activity. Some applications cannot be run from the emulator for various reasons, they could be run in the real phone with using the proxy monitoring.

After finishing the process, we will have some data about the application's behavior. The completeness of the resulting data depends on the application but in the almost all cases of analysis that we did, the information about the application is sufficed to threat is it a malware or not.

4.7 Developing an application based on proposed methods of the analysis

Our students have developed [13] a small aggregated single-window application that performs semi-automatically the functions described here, without having to manually run all the utilities, copying files, and so on. It can be used by a specialist of digital forensic science. The application communicates with the Android Platform Tools for managing connected phone, the emulator and examines the logs.

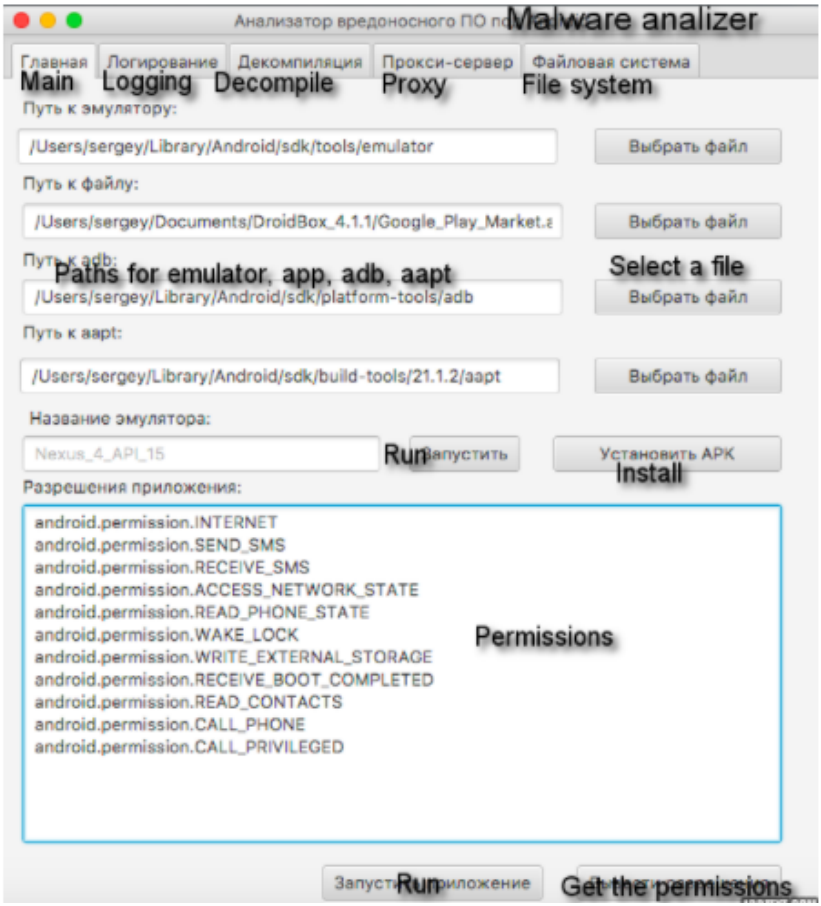
The app allows (see fig. 7):

- getting the permissions and copy the application from an Android phone. y malicious applications among given applications from given phones;
- make requests to the VirusTotal to check the application in the antivirus databases;
- start the Android emulator with the patched kernel and system image, make the dynamic analysis of the application, display the collected logs;
- decompile the application by the described decompilers and navigate through the decompiled files;
- run the integrated proxy server and monitor application's networking activity.
- Later in this application, we plan to generate reports in the standard form for digital forensic science.

5. Results and conclusions

- The article described the basic methods of analysis the behavior of malicious applications for Android. The general analysis process is proposed. Currently, the results are actually being used to make proofs of committing computer crimes (it is a research work by the contract).
- An extension of this work will be a higher automation of the analysis as well as analysis of the applications that have sophisticated protection, some

strong research work for call graph building from the disassembled and obfuscated code, providing patches for modern Android kernels.



-

Рис. 7. Приложение-помощник было разработано для упрощения анализа
Fig. 7. The helping application has been developed for simplify the analysis

References

[1]. Emarketer.com: 2 Billion Consumers Worldwide to Get Smart(phones) by 2016. Available under the link: https://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694. 03.03.2018.

[2]. Vedomosti.ru. Hackers have stolen from Android owners 349 million rubles for four quarters. Available under the link: https://www.vedomosti.ru/technology/articles/2016/10/13/660728-hakeri-ukrali-android. 03.03.2018 (in Russian)

[3]. Arp D. et al. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. NDSS, 2014, vol. 14, pp. 23-26

[4]. Enck W. et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS), 2014, vol. 32, №. 2, pp. 5.
[5]. VirusTotal – Free Online Virus, Malware and URL Scanner. Available under the link: https://www.virustotal.com. 03.03.2018
[6]. Debugging Dalvik programs with IDA. Hex-Rays. Available under the link: https://www.hex-rays.com/products/ida/support/tutorials/debugging_dalvik.pdf. 03.03.2018
[7]. pjlantz/droidbox: Dynamic analysis of Android apps. Available under the link: https://github.com/pjlantz/droidbox. 03.03.2018
[8]. dex2jar. Tools to work with android .dex and java .class files. Available under the link: https://sourceforge.net/projects/dex2jar. 03.03.2018
[9]. Fernflower is the first actually working analytical decompiler for Java. Available under the link: https://github.com/JetBrains/intellij-community/tree/master/plugins/java-decompiler/engine. 03.03.2018
[10]. CFR – another java decompiler. Available under the link: http://www.benf.org/other/cfr/. 03.03.2018
[11]. jadx – Dex to Java decompiler. Command line and GUI tools for produce Java source code from Android Dex and Apk files. Available under the link: https://github.com/skylot/jadx. 03.03.2018
[12]. Procyon/Java Decompiler. Available under the link: https://bitbucket.org/mstrobel/procyon/wiki/Java%20Decompiler. 03.03.2018
[13]. Abalmasov A.V., Staroletov S.M. Development of a malware analysis system for the Android platform. Bachelor's work. AltaiSTU, 2016. Available under the link: http://new.elib.altstu.ru/diploma/download_vkr/id/70003. 03.03.2018 (in Russian).

Методы анализа вредоносного программного обеспечения под ОС Android

С.М. Старолетов <serg_soft@mail.ru>
Алтайский государственный технический университет им. И.И. Ползунова, 656038 Барнаул, проспект Ленина, 46

Аннотация. В статье рассматривается проблема анализа приложений под ОС Андроид с целью выявления вредоносного поведения. Представлены методы анализа такого рода приложений, которые применялись при проведении реальных программно- и научно-технических экспертиз (статические, динамические проверки, отладка, декомпиляция, логирование). Описаны процесс анализа, существующие приложения, а также собственное программное обеспечение.

Ключевые слова: анализ поведения программ; вредоносное программное обеспечение; утечки данных; Java; программно-техническая экспертиза

DOI: 10.15514/ISPRAS-2018-30(1)-4

Для цитирования: Старолетов С.М. Методы анализа вредоносного программного обеспечения под ОС Android. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 55-68. DOI: 10.15514/ISPRAS-2018-30(1)-4

Список литературы

[1]. Emarketer.com: 2 Billion Consumers Worldwide to Get Smart(phones) by 2016. Available under the link: https://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694. 03.03.2018.

[2]. Vedomosti.ru. Хакеры украли у владельцев Android 349 млн рублей за четыре квартала. Доступно по ссылке: https://www.vedomosti.ru/technology/articles/2016/10/13/660728-hakeri-ukrali-android. 03.03.2018

[3]. Arp D. et al. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. NDSS, 2014, vol. 14, pp. 23-26

[4]. Enck W. et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS), 2014, vol. 32, №. 2, pp. 5.

[5]. VirusTotal - Free Online Virus, Malware and URL Scanner. Available under the link: https://www.virustotal.com. 03.03.2018

[6]. Debugging Dalvik programs with IDA. Hex-Rays. Available under the link: https://www.hex-rays.com/products/ida/support/tutorials/debugging_dalvik.pdf. 03.03.2018

[7]. pjlantz/droidbox: Dynamic analysis of Android apps. Available under the link: https://github.com/pjlantz/droidbox. 03.03.2018

[8]. dex2jar. Tools to work with android .dex and java .class files. Available under the link: https://sourceforge.net/projects/dex2jar. 03.03.2018

[9]. Fernflower is the first actually working analytical decompiler for Java. Available under the link: https://github.com/JetBrains/intellij-community/tree/master/plugins/java-decompiler/engine. 03.03.2018

[10]. CFR – another java decompiler. Available under the link: http://www.benf.org/other/cfr/. 03.03.2018

[11]. jadx – Dex to Java decompiler. Command line and GUI tools for produce Java source code from Android Dex and Apk files. Available under the link: https://github.com/skylot/jadx. 03.03.2018

[12]. Procyon/Java Decompiler. Available under the link: https://bitbucket.org/mstrobel/procyon/wiki/Java%20Decompiler. 03.03.2018

[13]. Абалмасов А.В., Старолетов С.М. Разработка системы анализа вредоносного ПО на платформе Android. Бакалаврская работа. АлтГТУ, 2016. Доступно по ссылке: http://new.elib.altstu.ru/diploma/download_vkr/id/70003. 03.03.2018

Asynchronous Distributed Algorithms for Static and Dynamic Directed Rooted Graphs

[1]*I.B. Burdonov* < *igor@ispras.ru* >
[1]*A.S. Kossatchev* <*kos@ispras.ru*>
[1,2,4]*V.V. Kuliamin* < *kuliamin@ispras.ru*>
[1,2]*A.N. Tomilin* <*tom@ispras.ru*>
[1,3]*V.Z. Shnitman* <*vzs@ispras.ru*>
[1]*Ivannikov Institute for System Programming of the RAS,*
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia
[2]*Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[3]*Moscow Institute of Physics and Technology (State University),*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
[4]*National Research University Higher School of Economics (HSE)*
11 Myasnitskaya Ulitsa, Moscow, 101000, Russia

Abstract. The paper provides a review of distributed graph algorithms research conducted by authors. We consider an asynchronous distributed system model represented by a strongly connected directed rooted graph with bounded edge capacity (in a sense that only a bounded number of messages can be sent through an edge in a given time interval). A graph can be static or dynamic, i.e. changing. For a static graph we propose a spanning (in- and out-) tree construction algorithm of time complexity $\boldsymbol{O}(n / k + d)$, requiring $\boldsymbol{O}(n\, d\, log\, \Delta^+)$ message size and the same size of memory of each computing agent located in graph vertex, where n is the number of vertices of the graph, k is the capacity of an edge, d is the maximum length of simple path in the graph, Δ^+ is the maximum outdegree of the vertices. The spanning trees constructed can be used in distributed computation of a function of the multiset of values assigned to graph vertices in a time not greater than $3d$. In a dynamic graph we suppose that $k = 1$ and an edge can appear, disappear, or change its end. We propose a dynamic graph monitoring algorithm than delivers information on any change to the root of the graph in $\boldsymbol{O}(n)$ or $\boldsymbol{O}(d)$ after the changes are stopped. We also propose graph exploration and marking algorithm with time complexity $\boldsymbol{O}(n)$. The marking provided by it is used in distributed computation of a function of the multiset of values assigned to dynamic graph vertices, which can be performed in time $\boldsymbol{O}(n^2)$ with messages of size $\boldsymbol{O}(log\, n)$ or in time $\boldsymbol{O}(n)$ with messages of size $\boldsymbol{O}(n\, log\, n)$.

Keywords: distributed algorithms; asynchronous systems; directed graph; rooted graph; dynamic graph; parallel computations

DOI: 10.15514/ISPRAS-2018-30(1)-5

For citation: Burdonov I.B., Kossatchev A.S., Kuliamin V.V., Tomilin A.N., Shnitman V.Z Asynchronous Distributed Algorithms for Static and Dynamic Directed Rooted Graphs. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 69-88. DOI: 10.15514/ISPRAS-2018-30(1)-5

1. Introduction

This paper considers distributed graph algorithms. To treat them formally, we must use a formal computation model, and the choice of such a model critically influences the complexity of possible algorithms. In this work we use the following model: a set of computing agents located in the vertices of the graph under consideration (an algorithm is intended to compute some information about this graph) and interacting by passing messages through graph arcs [1–3].

Such computing agents are called in various ways in the literature: tasks [1], processors [2], processes [3], or state machines [4] (if they are actually state machines, maybe not finite ones). We suppose that the agents can interact only in a peer-to-peer way. A round of agent's operation can include message reception, changing the agent's state, and sending another message. In [1] such an agent is called reactive entity or message-driven entity. To be able to send a message through some arc, an agent should refer to this arc. To make this possible, all arcs incident to one vertex are supposed to be enumerated, so an agent just use the arc's number. In case of directed graph, this enumeration includes only arcs outgoing from the vertex. We also suppose that am agent has no information on other vertices – it knows only the number of outgoing arcs in its vertex, and the vertex identifier if the graph is enumerated.

Here and in [1] computations on directed graph are considered, although computations on undirected graph are more usual, see [2, 3]. Note, that algorithms solving the same tasks on directed graphs are considerably more complex. An example is provided by the task of graph traversal by a single message, with agents in the vertices being finite state machines. For undirected graphs it is solved by well-known Tarry algorithm [2, 22] with time complexity O(m), where m is the number of graph arcs. This complexity is minimal, because it coincides with lower time bound for any algorithm for this task. For directed graphs lower time bound for the same task is O(nm), where n is the number of graph vertices, and the best known algorithm has time complexity O(nm + n2 log log n) [16]. It is still unknown whether the gap between lower bound estimate and actual algorithm complexity can be reduced for directed graphs.

Synchrony or asynchrony of computation agents is also an important aspect of the model. In synchronous models the agents work in lockstep mode, i.e., all the agents make one computation step simultaneously, so the main complexity is related with number of communication acts [18, 19]. In asynchronous model, the time of message transport can differ for different arcs and different messages. Usually the time of message transport through one arc is assumed to be bounded by some

constant, which can be taken as 1 tic. The algorithm time complexity in asynchronous model is the worst case time of its operation. Also in asynchronous model, an arc becomes a buffer of messages sent by one arc end and not yet accepted by the other, usually organized as a queue. This queue can be unbounded or can has some maximum number of messages that can be stored, called an arc capacity. It is so an important characteristic of a model [1]. In this work, we consider a model with bounded arc capacity.

We also assume that graphs can have multiple arcs leading from one vertex to another one, and can have loops leading from a vertex to itself (some authors called such graphs pseudographs or multigraphs). In opposite to our approach, most works on distributed computations consider graphs that cannot have multiple arcs or loops.

For distributed asynchronous algorithms, the memory used is an important characteristic. It is true both for the inner memory of a computation agent (which can be regarded as a logarithm of the number of its states) and for the size of messages used in communication. Therefore, further we estimate the inner memory of agents and the size of messages used in an algorithm as functions of the graph complexity.

In synchronous models all the agents start their operation simultaneously and the end of operation is often not specified explicitly. In asynchronous models the algorithm operation is initiated by an external Start message, which is accepted by an agent in some graph vertex, which than starts sending messages to other vertices, agents in which start their operation after accepting them [3]. We assume that a starting vertex is fixed, it is called the root of a graph.

The other problem with distributed computation is termination detection [2] – how to notice the end of algorithm operation. We assume that the algorithm is finished when the root agent sends to the environment special Finish message, possibly containing some result data. A procedure helping the root agent to learn that the algorithm work is finished is nontrivial in asynchronous models.

One of important tasks of distributed computations is computation of a global function on a graph [3]. In this work, we consider a task of computing a function of a multiset of values assigned to graph vertices. An agent located in a vertex knows the value assigned to it. To compute the global function value we use the notion of aggregate function, which global value can be computed from values for nonintersecting subsets of vertices, and minimal aggregate extension, which exists for any function [11, 12, 14, 21].

Such computation tasks are usually solved with the help of broadcast and convergecast operations. The broadcast problem is a problem of communication organization from one vertex to all other vertices. The convergecast problem is a problem of communication organization from all vertices to some single vertex. To solve broadcast and convergecast problems a spanning tree of the graph is used. In a directed graph broadcast is performed as query message passing along forward or spanning out-tree, which has the root coinciding with the root of the graph and arcs

leading from an ancestor vertex to its descendants. Convergecast is performed as answer message passing along backward or spanning in-tree, with arcs leading from descendant vertices to their ancestor, and finally to the root. To use spanning in-tree for correct convergecast an agent in a vertex should know the number of incoming backward arcs, because it can send its answer only after accepting all the answers of agents located in the descendants.

Algorithm of spanning tree construction for undirected graph uses information propagation with feedback [3]. It is performed by broadcast with the help of flooding (sending messages along all arcs that are not used before), and then convergecast is performed to mark out the spanning tree arcs and assign the number of descendants to each vertex. This algorithm uses the possibility to send a message along an arc, which is used to transfer a previously accepted message. Directed graph has no such a possibility. Instead, one can use for the same task a path from the arc's end to the root and a path from the root to the arc's start. To construct these paths, broadcast messages should accumulate the description of the path passed as a sequence of arc numbers. In addition, convergecast also uses flooding, but takes into account that several floodings initiated by different vertices are working in parallel on the graph.

In many applications of distributed computations on graphs, including communication networks, VLSI design, assembly planning, the base graph can be slightly changed from time to time, that is, its vertices or arcs can be inserted or deleted. In the last decade, there is a growing interest to algorithms solving certain tasks on such dynamic graphs. If one allows only arc insertion or deletion, keeping the set of vertices stable, such a graph is described by dynamic connectivity structure, a data structure that dynamically maintains information about the connected vertices. Some works consider a case when the set of vertices can be also changed [20].

The following problems are often considered on dynamic graphs: connectivity determination, computation of the shortest path between the given vertices, spanning tree construction, etc. The usual goal of dynamic graph algorithms is to update efficiently the solution of a problem after changes, rather than having to re-compute it from scratch each time. Such algorithms provide incremental updates for the previous solution. Our approach is different; we consider possibility to solve some task without any previously gathered data on a graph, which is still changing. We treat directed dynamic graphs with stable vertices set and changing arcs, an arc can disappear, appear, or change its end. We also consider a problem of graph monitoring: how to gather actual information on graph structure in its root agent. Of course, this information gathering have sense if changes are not too quick. We give an estimate of time needed for graph structure information update in the root agent, so if no changes occur on this interval, one can trust the data gathered.

Another problem on dynamic graphs considered here is computation of a function on a multiset of values assigned to graph vertices. In this case, we cannot use

spanning trees because of possible changes in graph structure. In such a setting one can send queries to all vertices from the root and then gather the answers, but answers from different vertices should be distinguished. There are different ways to distinguish them. One technique uses static enumeration of vertices and their identifiers, in this case computation complexity is O(n) and message size is O(n log n). Another technique uses linear order on vertices and sending answers according to this order, in this situation message size can be O(log n), but total computation time becomes O(n2). These methods can be generalized, namely, one can construct virtual spanning tree of the graph with the help of additional virtual arcs and use it to gather answers from vertices. If such a spanning tree has w leaf vertices, then messages used can be of size O(w log n) and total computation time is O(n2 / w).

Below in Section 2 we consider static graphs, their marking out algorithm, and parallel computation on static graph. Section 3 presents the dynamic graphs, monitoring algorithm, and parallel computation on dynamic graph. The conclusion summarizes the paper.

2. Static Graphs

In this section we suppose that distributed computation is performed by agents located in vertices of a directed graph and interacting by passing messages through graph arcs, along their direction. The graph is static, i.e., stays unchanged during the computation. An agent has no information on graph structure, it knows only the number of arcs outgoing from its vertex. Outgoing arcs are numbered, so an agent can send a message along an arc using its number.

Agents operation is asynchronous. On single step of its operation, an agent can accept all messages sent to it through incoming arcs, perform some internal computations, and send several messages through some outgoing arcs. An arc is a buffer with capacity k, i.e., in one time not more than k messages can be sent along this arc and not yet accepted. If an arc already has k unaccepted messages, an agent located in its starting vertex cannot send new messages along it. We suppose that message size is bounded, so arc capacity put a bound on the size of stored data. This can be implemented in two ways: one can has k messages of some constant size on an arc or one message of a size that is k time bigger. The difference is that in the first case it is possible to send more messages, if the number of unaccepted messages on an arc is less than k. We consider here algorithms using the first option, they can send several messages of bounded size along one arc, but not greater than k in a time.

A message can be sent along only one arc, along all outgoing arcs, or along some subset of outgoing arcs. Nevertheless, we assume that a message waits until all outgoing arcs are capable to transport it. This assumption can increase the total computation time, but make unnecessary separate signals about ability of arcs to

carry a message, a single signal concerning all arcs (that a message can be sent along any of them) is sufficient.

To estimate time complexity of computation we count time of internal operation of an agent negligible. We also assume that a message is transported through an arc in one tic, i.e., not later than one tic after sending it is accepted on the end of the arc.

We also use the following notation: n is the number of graph vertices, m is the number of graph arcs, d is the maximum length of a simple path (passing each vertex not more than once), $\Delta+$ is the maximum number of arcs outgoing from one vertex, $\Delta-$ is the maximum number of arcs incoming to one vertex.

2.1 Marking out a graph

We consider in this subsection a problem of graph exploration. The solution of this problem should be an algorithm that starts by accepting an external Start message in the root agent, performs marking out of the graph by computing and storing in agents' internal memory some data that provides a local information on graph structure, and finishes by sending Finish message from the root agent as an answer to the Start one.

The graph marking out includes the following.

Spanning out-tree, directed from the root. We call its arcs forward arcs. An agent located in each vertex should store all the outgoing forward arcs. Other outgoing arcs are called chords.

Spanning in-tree, directed to the root. Its arcs are called backward arcs. A vertex can have not greater than one outgoing backward arc (the root has no one). An agent located in the vertex (except for the root) should store the number (id) of the backward arc.

An agent located in the vertex should store the number of backward arcs incoming to it. This number is necessary to gather correctly information along the in-tree, the messages accepted along backward arcs are counted until the counter reaches this number that means that all the data from the preceding part of in-tree are already collected.

This marking out can be considered as a result of graph exploration, sufficient to perform further exploration of the graph or distributed computations on it in an efficient way.

The algorithm of marking out is described in details and with proofs in [10, 11]. Here we provide only the general description of it and give complexity estimates without proofs. The algorithm has three phases: construction of spanning out-and in-trees, construction termination detection, and setting counters of incoming backward arcs.

2.1.1 Construction of spanning out-and in-trees

This step uses 4 message types: Start is used to initiate the operation in the root and to set vertex identifies, Search is used to find paths from any vertex to the root, Forward marks out the spanning out-tree, Backward marks out the spanning in-tree.

The root agent accepts Start message from the environment and this initiates the graph marking out. Start message is sent further with flooding pattern: an agent accepts the first Start message, sends it along all outgoing arcs, and ignores all other incoming Start messages (accepts them and does nothing in addition). A message of this kind accumulates a vector – the sequence of arc numbers used along its way (each agent before sending the message further appends the sequence in it with the number of the arc it uses to send it). The vector of the first Start message accepted in the vertex becomes the identifier of this vertex. The root identifier is an empty vector.

A Search message looks for a path from a vertex to the root. An agent that accepts the first Start message creates and sends Search messages along all outgoing arcs. Such a message contains an initiator vector (the identifier of the vertex where it is created) and accumulates backward vector – the sequence of arc numbers it passes. An agent stores internally the set of identifiers of initiators of Search messages it already processed. After accepting Search message, a non-root agent looks for its initiator identifier in this set. If the initiator is already processed, the message is ignored. Else, the initiator identifier is stored and the agent sends Search messages along all outgoing arcs with their backward vectors appended by numbers of corresponding arcs.

The root agent accepting Search message creates Forward message putting the initiator vector and backward vector in it. Forward message is moved along the path described by the initiator vector. To make this possible, each agent before sending the message along the first arc of the vector marks this arc as a forward one and removes its number from the vector in the message. If an agent accepts the Forward message with empty initiator vector, then it is the initiator. It creates Backward message, put backward vector from the Search message in it, and sends it along backward path.

Backward message moves along backward path in the same way – each agent accepting it before sending it along the first arc of the backward vector, marks this arc as a backward one (stores its number in a special memory slot), and removes its number from the backward vector in the message. Backward arc number can be already stored in a vertex agent; in this case, the agent rewrites it. This guarantees that no cycles along backward arcs appears in the end.

2.1.2 Construction termination detection

Construction termination detection is based on counting of arc starts and ends, when this numbers become equal all the arcs are explored and so the construction step is complete. To implement this idea the root agent stores the value of known arc starts

minus known arc ends. After the operation start, the counter equals to the number of outgoing arcs from the root.

To count arc starts every Search message has an additional field storing the number of arcs outgoing from the message initiator. After accepting such a message, the root agent adds this field value to the counter.

To count arc ends we add two special message types, Finish and Minus. All agents have the counter of incoming arcs. When Forward message is accepted by its goal agent (the initiator), it sends Finish messages along all outgoing arcs. The root agent does this just after sending Start messages. When Finish message is accepted, the agent increments the counter of incoming arcs. Minus message is created and sent to the root agent along the backward arc just after the backward arcs is set in the vertex for the first time. The first Minus message contains the value of the counter of incoming arcs. If a Finish message is accepted after sending the first Minus message, additional Minus message with value 1 is created and sent to the root agent. When the root agent accepts Minus message, it decreases its counter of arc starts on the value of the message field. When this counter becomes 0, the construction step is finished.

2.1.3 Setting counters of incoming backward arcs

After the construction of in-tree is finished, it becomes possible to set the counters of incoming backward arcs. For this purpose two additional message kinds, Begin and End are used. Begin messages are created by the root agent and broadcasted along the out-tree – each agent after accepting such a message creates its copies and send them along all outgoing forward arcs. It also creates and sends along the backward arc End message with initial flag set to true.

After accepting End message with raised initial flag, an agent increments its incoming backward arcs counter, sets the flag to false, and sends the message along backward arc (if it isn't the root agent). End messages with dropped initial flag are just sent along backward arcs. The root agent counts accepted End messages. When their number reaches the number of registered initiators, the counter setting and so the whole algorithm operation are finished.

2.1.4 Algorithm complexity

The time complexity of the algorithm is $O(n / k + d)$. The size of internal agent memory used is $O(nd \log \Delta+)$. The maximum size of message used is $O(d \log \Delta+)$. Note, that here d means the maximum length of non-intersecting path, not the graph diameter. This is a consequence of asynchrony; the forward path from the root to some vertex may appear to be the maximal non-intersecting path, not a path of minimum possible length between these two vertices.

2.2 Parallel computations and aggregate functions

In this subsection we consider the problem of parallel computation of a value of a function on graph. We suppose that each graph vertex has some value assigned to it (vertex agent has special operation providing this value) and we need to compute the value of some function on this multiset of values.

Let X denote a set, from which values assigned to graph vertices are taken. X# is a set of all finite multisets of elements from X. Multisets are considered, because different graph vertices may have equal values assigned.

A function $g : X\# \to Y$ is called aggregate when $\exists e : Y \to Y \ \forall a, b \in X\#$ $g(a \cup b) = e(g(a), g(b))$. That is, g value on a multiset can be computed by parts. In this situation e is called an aggregator of g.

Examples of aggregate functions are the following: sum $\Sigma : \{a1,...,an\} \to a1 +...+ an$ has $e : (a, b) \to a + b$, mimimum $\min : \{a1,...,an\} \to \min(a1,...,an)$ has $e : (a, b) \to \min(a, b)$, sum of squares $Q : \{a1,...,an\} \to a12 +...+an2$ has $e : (a, b) \to a + b$.

Not all the functions are aggregate, for example, arithmetical mean is not an aggregate function. But one can expand function $f : X\# \to Y$ as a composition $f = hg$, where $g : X\# \to Z$ is aggregate and $h : Z \to Y$ is just some function. In this case g is called an aggregate extension of f. An aggregate extension can help to compute f by parts, by computing g by parts and then computing h once in the end. Some extensions aren't helpful actually, e.g., one can take an identity function on X# as g and f itself as h. To prevent this situation, we use minimal aggregate extension. Intuitively, minimal aggregate extension keeps minimum information to make possible further computation of f. Formally, $g : X\# \to B$ is a minimal aggregate extension of $f : X\# \to Y$, if g is its aggregate extension and for each $g' : X\# \to B'$, which is an aggregate extension of f there is $j : B' \to B$, such that $g = jg'$. Minimal aggregate extension exists for every function on multisets and is unique up to isomorphism.

Examples of minimal aggregate extension are the following: for arithmetical mean function $f : \{a1,...,an\} \to (a1 + ... + an) / n$ the minimal aggregate extension is provided by function $g : \{a1,...,an\} \to ((a1 + ... + an), n)$ and corresponding $h : (a, n) \to a / n$, for root mean square $f : \{a1,...,an\} \to ((a12 +...+an2) / n)1/2$ the minimal aggregate extension is provided by function $h : (a, n) \to (a / n)1/2$.

Aggregate functions theory is a modification of theory of inductive functions on finite sequences, see [21]. Detailed proofs can be found in [11].

2.3 Parallel computations on static graph

In this subsection we describe an algorithm for computation of a function of values assigned to graph vertices (details can be found in'[11, 12]). This algorithm uses the marking out provided by the algorithm described above: spanning in-and out-trees

and counters of incoming backward arcs. Note, that this marking out can be constructed once and further used for many computations.

Computation is started when the root agent accepts an external **Start** message with specification of three functions *h*, *g*, *e*. One need to compute the value of $f = hg$ on the multiset of values from X assigned to graph vertices. g is a minimal aggregate extension of f, and e is an aggregator function of g. Specification of a function can be actually a program to compute it.

The computation is based on broadcast along spanning out-tree and convergecast along spanning in-tree of the graph. **Request** messages containing specifications of g and e are transferred along forward arcs. **Response** messages are transferred along backward arcs and contain value of g on a subset of values assigned to vertices of a subtree of in-tree.

When an agent of a leaf vertex of in-tree (it has incoming backward arcs counter equal to zero) accepts **Request**, it computes the function g of the value assigned to its vertex, and sends the computed value in the field of **Response** message along the backward arc. Agent of a non-leaf vertex also computes the value of g, but doesn't send it until getting **Response** messages from all incoming backward arcs, using the counter to detect this situation. When it gets all the responses, it can compute the complete value of g for the subtree having the corresponding vertex as its root, and then sends the result with **Response** message along its backward arc. When the root agent get all the responses, it can compute the value of f by applying h to the value of g, and sends the final value to the environment.

The total computation time of this algorithm is $\boldsymbol{O}(3d)$, ignoring the computation complexity of g, h, e, the memory size of an agent is bounded by $\boldsymbol{O}(\Delta^+ + log\ \Delta^- + x + y)$, message size is $\boldsymbol{O}(x + y)$, where x is the size of specification of g and e, and y is the size of g values.

3. Dynamic graphs

In this section we consider computations on dynamic graphs, i.e., graphs, which arcs can change while the set of vertices doesn't change. There are three possible changes of arcs.

- An arc can appear. This event produces a special signal to arc start agent; the appearance signal contains the number of new arc.
- An arc can disappear. We assume that if there is a message being transported along the arc, the arc start agent gets a signal with the number of disappeared arc, and all the messages sent along it and not yet accepted also disappear. If there are no such messages, then no signal is generated. However, in case the agent tries to send a message along the disappeared arc, it also gets a disappearance signal.
- An arc can change its end vertex. No signal is generated on this event.

This model requires minimal number of additional signals to provide agents with data on arc changes. We assume that a special release signal is sent to an agent in the arcs starting vertex when a message of the arc is accepted and new message can be sent along it. We also suppose that arc capacity is 1 – only single message can be transported at a time.

If an arc changes are too frequent, two unprocessed signals can emerge, and in this case we assume that one of them is lost. The second signal can be only an appearance signal, so the rules of signal collapse are as follows.

- From two appearance signals only the last one is retained.
- From disappearance and appearance signals only the appearance one is retained.
- From release and appearance signals any one can be retained.

We ignore the time of agent's internal operation and consider the time of one message transport through an arc as bounded by one tic. Too frequent changes can make impossible message transport along an arc; to prevent this we suppose that some arcs are *long-living*. A long-living arc always transfers at least one message while it is in a stable state. So, long-living arcs are changed not more frequent than once in a tic. We also suppose that changes preserve strong connectedness of the graph. More precisely, the subgraph made of long-living arcs is always strongly connected and contains all the vertices of the graph.

The algorithms presented below are operating when graph is changing, so they use the following heuristics. All the data an agent need to send another agent should be put in a single message, because only a single message is guaranteed to be transported along an arc. Also, a message should be sent each time it becomes possible, that is, the necessary arc appears or is released. Otherwise, the message data may not reach other agents because the arc may change its end several times without any signals to the sender.

3.1 Dynamic graph monitoring

We assume that the graph is enumerated, its vertices has numbers from 1 to n, and each agent knows the corresponding vertex number.

We consider the problem of graph monitoring: to collect complete information on graph structure in the root agent (this information is also gathered in all other vertex agents). Due to permanent changes, one cannot guarantee the correctness of this information, but we can require that the information on arc change becomes known to all agents not later than in some finite time T_0. So, if in time T_0 an arc doesn't change, the agents has correct information about it.

The algorithm is described in details and with proofs in [13], here we provide only the general description and complexity estimates without proofs.

3.1.1 Arc description and messages

The algorithm works as follows. Each agent has an internal storage of arc descriptions. An arc description is a triple of the number of start vertex, the arc number in its start, and the number of the end vertex (if it is known, else it is replaced with 0). Each message also contains a set of arc descriptions. Each time an agent accepts a message, it compares his internal arc descriptions with the ones in the message, copies to its internal memory the descriptions of arcs it doesn't know, and replaces the descriptions, which are more recent in the message. After that, it sends its internal set of descriptions in messages along all outgoing arcs.

In case an arc appears, the start vertex agent creates the new arc description with zero end number. The start vertex agent is also the first one who learns about arc disappearance. In case of arc end change the end agent can discover this, if it accepts a message with another end number (or zero) in arc description. To make this noticeable, the message contains the identifier of the arc used to transfer it.

3.1.2 Arc rank

Since an arc can change several times, an agent should determine whether it has more recent arc description than the one in a message or not. For this purpose, we use a number field in arc description called *rank* and showing "a version number" of the description. When the rank is greater, the corresponding description is more recent and should replace an older one. When an agent learns about a new arc or arc change and creates or modifies an arc description, it should increase the rank in it.

To be sure that in every other description the rank is smaller, it isn't always enough to increase it in a new description by 1. Several arc end changes can lead to the situation when different end agents assign to a new description rank value greater by 1 than the old one. To resolve this, the arc start agent accepting a message with newer arc description always adds additional 1 to updated arc description in its storage. Accordingly, accepting an appearance or disappearance signal the arc start agent increases the arc description rank by 2.

3.1.3 Estimates

The estimate of time needed to an agent to learn about arc change is $T_0 = \boldsymbol{O}(n)$. The estimate of time needed to transport information on all changes after they stop is $T_1 = \boldsymbol{O}(d)$. The size of agent's internal memory and the message size are both $\boldsymbol{O}(m \log \Delta^+ nv)$, where v is the maximum number of changes of one arc. This is caused by the fact the each agent actually stores the full description of graph structure and the same full description is contained in a message. The multiplier v is caused by the arc rank field in messages.

In case when all arc are long-living, we can eliminate v multiplier. In such a situation, the number of arc changes is not greater than total time of operation, which is $\boldsymbol{O}(n)$. So, one can count ranks modulo some number of the value $\boldsymbol{O}(n)$, and the estimate of message size becomes $\boldsymbol{O}(m \log \Delta^+ n)$.

3.2 Parallel computations on a dynamic graph

For parallel computation on dynamic graph, we need to mark it out, which is not performed by monitoring algorithm. Moreover, one cannot use the same method with broadcast of **Request** and convergecast of **Response** messages, because changes in arcs of in-or out-trees may require modifications in spanning trees. Also, successful message transport along arcs is not guaranteed.

However, as one can see on monitoring algorithm example, assumption on arcs long-liveness can guarantee that information from any vertex can be delivered to any other vertex. To do this, vertex agents should send all gathered information along any arc any time the arc is able to transfer it.

This approach is sufficient to deliver **Request** messages to all vertices. One can gather responses in the same way in the root and perform all the computations by the root agent, but this requires very large messages that should contain all the data from all known vertices. Use of spanning in-tree and backward arc counters helps to minimize the size of **Response** messages; along backward arcs one can send only the summary information on the subtree, having the start vertex as its root.

In case of dynamic graph the idea is to use a virtual spanning in-tree, constructed using additional virtual arcs connecting arbitrary vertices. In addition, we can choose the form of the virtual in-tree, since its height h (the maximum length of a path from its leaf to the root) determines the computation time (the computations can be performed in parallel on different paths only, on one path to the root they are performed sequentially) and its width w (the number of leafs) determines the maximum message size.

For the given n and w one can construct the tree of minimum height with n vertices and w leafs. It is a fan-like tree homeomorphic to star graph with w rays (branches), each ray contains h or $h - 1$ vertices, where $h = \lceil (n - 1) / w \rceil$. The vertices on such a tree (besides the root) can be enumerated by two-component index, the ray number from 1 to w and the number of the vertex on the ray from 1 to h, smaller numbers are closer to the root. The backward arc counter in such a tree is necessary only in the root (and it has value w). Other agents should know only the status of its vertex whether it is a leaf (counter value is 0) or not (counter value is 1).

Below we present a general description of the algorithm and its complexity estimates. Details and proofs can be found in [14].

3.2.1 Marking out a dynamic graph

Marking is performed in two phases; at the first phase the root agent gathers data on all vertices using **Forward** messages, at the second one the root constructs the virtual spanning in-tree and sends it to all the vertices using **Backward** messages.

The root agent after accepting the **Start** message from the environment creates **Forward** message, which further circulates on the graph accumulating the arc descriptions (with the same fields as in monitoring algorithm). Any agent accepting this message updates its internal storage with its data, updates the message with the

descriptions, which are more recent in its storage, and sends the updated message along all outgoing arcs.

We need additional assumptions to guarantee that the root agent learns about all the vertices. First, we assume that in the clash of release and appearance signals the release signal is retained. This helps to guarantee the successful transport of a message, since release signal cannot be lost now. Second, we need more stable *initial arcs*. Initial arc is stable from the start of algorithm work until the first message is transported along it. We also assume that all the vertices are reachable from the root along initial arcs.

Using this assumptions an agent can register only arcs appearing until it accepts the first message (and ignore arc appearing signals after this). This helps to register all the initial arcs, and hence all the vertices becomes known to the root agent after some time. The root agent can check the condition that it knows ends of all known arcs, and due to assumptions, this condition is sufficient to conclude that it knows all vertices.

When the root agent learns about all vertices, the second phase starts. The root agent creates the description of virtual fan-like tree, which is sent in **Backward** message. The tree description consists of descriptions of vertices. Each vertex description includes the vertex number, its index in the tree, and the flag stating whether it is a leaf vertex or not. Each agent accepting **Backward** message first time, stores the description of its vertex, removes it from the tree description, stores the modified description, and sends it with **Backward** message along all outgoing arcs. It also stops resending of **Forward** messages. Accepting **Backward** message second time or further, it constructs and stores the intersection of sets of vertex descriptions stored internally and in the message, and stores and sends this intersection. After some time the description of virtual tree in **Backward** messages becomes empty, and then this description in the root agent becomes empty. This is the end of the marking of the graph.

3.2.2 Function computation

To compute a function of a multiset of values assigned to graph vertices we use request-response scheme, as for static graph. But in case of dynamic graph, all the information should be sent in one message along all the arcs that can transport it (that is, each time just after appearance or release signals). Therefore, each request message should also contain a partial response. One response value for each tree branch can be contained in a message, and it is accompanied by the index of the agent, which computed this response. The response value is at first set by the agents located in the leafs of virtual trees, then it is modified by the agents of the next virtual tree vertices in the direction to the root, until the root agent gathers responses from all the root neighbors in the virtual tree. Accepting **Request** message, an agent also stops to send **Forward** and **Backward** messages.

After the end of computation, **Request** messages can still move along the arcs. Their movement can be stopped by next computation, to do this the root agent can enumerate computation tasks requested by the environment and put the number of task in **Request** message. So, **Request** message contains the number of task, the specification of *g* and *e* functions, the set of pairs (the value of *g* for *i*-th branch, the index of response creator).

3.2.3 Estimates

The time of graph marking out is $\boldsymbol{O}(n)$, the time of function computation is $\boldsymbol{O}(n^2 / w)$. The size of agent internal memory and the size of a message is $\boldsymbol{O}(m \log n\Delta^+)$ on the marking out phase and $\boldsymbol{O}(x + \log N + wy + w \log n)$, where N is the maximum number of tasks, x is the size of specification of g and e, and y is the size of g values. For $w = 1$ one have total computation time $\boldsymbol{O}(n^2)$ and message size $\boldsymbol{O}(x + \log N + y + \log n)$, for $w = n$ one have minimal total computation time $\boldsymbol{O}(n)$ and maximal message size $\boldsymbol{O}(x + \log N + ny + n \log n)$. To remove dependence on N, one can use the same modification as in monitoring algorithm if all arcs are long-living, task number can be countered modulo some number of value $\boldsymbol{O}(n^2 / w)$.

4. Conclusion

The paper presents algorithms for distributed computation of functions on directed graphs, which is performed by agents assigned to graph vertices, communicating by message passing along directed arcs, and knowing only the number of arcs outgoing from the corresponding vertex.

At first we consider static graph, and describe an algorithm that performs the marking out of the graph preparing it for further computations. The marking includes marks of forward arcs, making up a spanning out-tree, along which all the vertices can be reached from the root, marks of backward arcs, making up a spanning in-tree, along which responses from all the vertices can be gathered in the root, and incoming backward arcs counters, which are used to collect responses correctly. The time complexity of the marking algorithm is $\boldsymbol{O}(n / k + d)$. The size of internal agent memory and message size used are $\boldsymbol{O}(nd \log \Delta^+)$. The computation algorithm for a function of multiset of values assigned to graph vertices uses the marking constructed to complete computations in $\boldsymbol{O}(d)$. Here n is the number of vertices, d is the length of maximal non-self-intersecting path, Δ^+ is maximum outdegree, k is the arc capacity, the maximum number of messages that can be sent along an arc and not yet accepted.

Next we concern parallel computations on a dynamic graph that can change its structure during computation. We present a graph monitoring algorithm providing information about most recent changes to all the vertex agents in $\boldsymbol{O}(n)$. We also describe a parameterized algorithm for parallel computation of a function on a dynamic graph. The algorithm parameter w can be chosen from the interval from 1 to n and helps to balance computation time and the size of messages used. The time

of computation is $\boldsymbol{O}(n^2 / w)$ and the message size and agent memory size used is $\boldsymbol{O}(w \log n)$ (ignoring memory for function specification and result storage).

References

[1]. Barbosa V.C. An introduction to distributed algorithms. MIT Press, Cambridge, MA, USA, 1996

[2]. Kshemkalyani A.D., Singhal M. Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press, March 2011

[3]. Raynal M. Distributed Algorithms for Message-Passing Systems. Springer Publishing Company, Incorporated, 2013

[4]. Schneider F.B. The State Machine Approach. A Tutorial. Fault-Tolerant Distributed Computing. LNCS 448, 1990, pp. 18-41

[5]. Burdonov I.B., Kossatchev A.S. Testing of automata system. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 1, 2016, pp. 103-130. DOI: 10.15514/ISPRAS2016-28(1)-7 (in Russian).

[6]. Burdonov I.B., Kossatchev A.S. Automata system: composition according to graph of links. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 1, 2016, pp. 131-150. DOI: 10.15514/ISPRAS-2016-28(1)-8 (in Russian).

[7]. Burdonov I.B., Kossatchev A.S. Automata system: determinism conditions and testing. Trudy ISP RAN/Proc. ISP RAS, vol. 28, issue 1, 2016, pp. 151-184. DOI: 10.15514/ISPRAS-2016-28(1)-9 (in Russian).

[8]. Burdonov I.B., Kossatchev A.S. Testing of automata system. Vestnik Tomskogo gosudarstvennogo universiteta. [Bulletin of Tomsk State University. Management, Computer Science and Informatics], № 1, 2017, pp. 67-75 (in Russian).

[9]. Burdonov I.B., Kossatchev A.S. Generalized model of automata system. Vestnik Tomskogo gosudarstvennogo universiteta. [Bulletin of Tomsk State University. Management, Computer Science and Informatics], № 4(37), 2016, pp. 89-97 (in Russian).

[10]. Burdonov I.B., Kossatchev A.S. Buidling direct and back spanning trees by automata on a graph. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 6, 2014, pp. 57-62. DOI: 10.15514/ISPRAS-2014-26(6)-4

[11]. Burdonov I.B., Kossatchev A.S., Kuliamin V.V. Parallel computations on a graph. Programming and Computer Software, vol. 41, № 1, 2015, pp. 1-13. DOI: 10.1134/S0361768815010028.

[12]. Burdonov I.B., Kossatchev A.S., Kuliamin V.V. Parallel calculations by automata on direct and back spanning trees of a graph. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 6, 2014, pp 63-66. DOI: 10.15514/ISPRAS-2014-26(6)-5

[13]. Burdonov I.B., Kossatchev A.S. Monitoring of dynamically changed graph. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 1, 2015, pp. 69-96. DOI: 10.15514/ISPRAS2015-27(1)-5 (in Russian).

[14]. Burdonov I.B., Kossatchev A.S. Parallel Calculations on Dynamic Graph. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 2, 2015, pp. 189-220. DOI: 10.15514/ISPRAS-2015-27(2)-12 (in Russian).

[15]. Burdonov I.B., Kossatchev A.S. Analysis of directed graph by a set of unmoving automata. Programmnaya inzheneriya [Software Engineering], vol. 8, № 1, pp. 16-25 (in Russian)

[16]. Bourdonov I.B. Traversal of an Unknown Directed Graph by a Finite Robot. Programming and Computer Software, vol. 30, № 4, 2004, pp. 188-203. DOI: 10.1023/B:PACS.0000036417.58183.64
[17]. Bourdonov I.B. Backtracking Problem in the Traversal of an Unknown Directed Graph by a Finite Robot. Programming and Computer Software, vol. 30, № 6, 2004, pp. 305-322. DOI: 10.1023/B:PACS.0000049509.66710.3a
[18]. Lynch, N.A. Distributed algorithms. The Morgan Kaufmann Series in Data Management Systems. Kaufmann, San Francisco, Calif., 1996
[19]. Peleg D. Distributed computing – A Locality-sensitive approach. SIAM Monographs on Discrete Mathematics and Applications. 2000
[20]. Demetrescu C., Finocchi I., and Italiano G.F. Dynamic Graphs. In Handbook of Data Structures and Applications, sec. 36, 2004
[21]. Kushnirenko A.G., Lebedev G.V. Programming for mathematicians. Nauka, Glavnaya redaktsiya fiziko-matematicheskoi literatury [Science, the main edition of physical and mathematical literature], Moscow, 1988 (in Russian)
[22]. Tary G. Le probl`eme des labyrinthes. Nouv Ann Math 14, 1895.

Асинхронные распределенные алгоритмы на статических и динамических ориентированных корневых графах

[1]И.Б. Бурдонов <igor@ispras.ru>
[1]А.С. Косачев <kos@ispras.ru>
[1,2,4]В.В. Кулямин <kuliamin@ispras.ru>
[1,2]А.Н. Томилин <tom@ispras.ru>
[1,3]В.З. Шнитман <vzs@ispras.ru>
[1]Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25
[2]Московский государственный университет имени М.В. Ломоносова, 119991 ГСП-1 Москва, Ленинские горы, МГУ имени М.В. Ломоносова, 2-й учебный корпус, факультет ВМК
[3]Московский физико-технический институт, 141700, Московская область, г. Долгопрудный, Институтский пер., 9
[4]Национальный исследовательский университет «Высшая школа экономики» 101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Эта статья представляет собой обзор серии работ авторов, посвящённых исследованию распределенных систем. Рассматривается асинхронная модель распределённой системы, представленную сильно связанным ориентированным корневым графом, с ограниченной емкостью дуги (в том смысле, что только ограниченное количество сообщений может быть отправлено по дуге за определенный интервал времени). Граф может быть статическим или динамическим, т.е. меняющимся во времени. Для статического графа предлагается алгоритм построения

прямого и обратного остовных деревьев с оценкой времени $\boldsymbol{O}(n / k + d)$, размером памяти в вершине и сообщения $\boldsymbol{O}(n\, d\, log\, \Delta^+)$, где n – число вершин графа, k – емкость дуги, d – длина максимального пути, Δ^+ – максимальная полустепень исхода вершин. Построенные кушниренко используются в распределенном алгоритме вычисления функции от мультимножества значений, приписанных вершинам графа, за время не более $3d$. В динамическом графе предполагается, что $k = 1$, дуга может появляться, исчезать или менять свой конец. Предлагается алгоритм мониторинга динамического графа, который доставляет в корень информацию о каждом изменении в графе за время $\boldsymbol{O}(n)$ или $\boldsymbol{O}(d)$ после прекращения изменений. Также предлагается алгоритм сбора информации о вершинах графа и разметки графа за время $\boldsymbol{O}(n)$. Эта разметка используется в алгоритме вычисления функции от мультимножества на динамическом графе за время $\boldsymbol{O}(n^2)$ с размером сообщения $\boldsymbol{O}(log\, n)$ или за время $\boldsymbol{O}(n)$ с размером сообщения $\boldsymbol{O}(n\, log\, n)$.

Ключевые слова: распределенные алгоритмы; асинхронные системы; ориентированный граф; корневой граф; динамический граф; параллельные вычисления

DOI: 10.15514/ISPRAS-2018-30(1)-5

Для цитирования: Бурдонов И.Б., Косачев А.С., Кулямин В.В., Томилин А.Н., Шнитман В.З. Асинхронные распределенные алгоритмы на статических и динамических ориентированных корневых графах. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 69-88. DOI: 10.15514/ISPRAS-2018-30(1)-5

Список литературы

[1]. Valmir C. Barbosa. An introduction to distributed algorithms. // MIT Press, Cambridge, MA, USA, 1996

[2]. A.D. Kshemkalyani, M. Singhal. Distributed Computing: Principles, Algorithms, and Systems. Cambridge University Press, March 2011. 756 pages

[3]. Michel Raynal. Distributed Algorithms for Message-Passing Systems. Springer Publishing Company, Incorporated, 2013. 500 pages

[4]. Fred B. Schneider. The State Machine Approach. A Tutorial. Fault-Tolerant Distributed Computing. LNCS 448, 1990, pp. 18-41

[5]. Бурдонов И.Б., Косачев А.С. Тестирование системы автоматов. Труды ИСП РАН, том 28, вып. 1, 2016 г., стр. 103-130. DOI: 10.15514/ISPRAS2016-28(1)-7

[6]. Бурдонов И.Б., Косачев А.С. Система автоматов: композиция по графу связей. Труды ИСП РАН, том 28, вып. 1, 2016 г., стр. 131-150. DOI: 10.15514/ISPRAS-2016-28(1)-8

[7]. Бурдонов И.Б., Косачев А.С. Система автоматов: условия детерминизма и тестирование. Труды ИСП РАН, том 28, вып. 1, 2016 г., стр. 151-184. DOI: 10.15514/ISPRAS-2016-28(1)-9

[8]. Бурдонов И.Б., Косачев А.С. Тестирование системы автоматов. Вестник Томского государственного университета. Управление, вычислительная техника и информатика, №1, 2017 г., стр. 67-75.

[9]. Бурдонов И.Б., Косачев А.С. Обобщенная модель системы автоматов. Вестник Томского государственного университета. Управление, вычислительная техника и информатика, №4(37), 2016 г., стр. 89-97.

[10]. Burdonov I.B., Kossatchev A.S. Buidling direct and back spanning trees by automata on a graph. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 6, 2014 г., pp. 57-62. DOI: 10.15514/ISPRAS-2014-26(6)-4

[11]. Бурдонов И.Б., Косачев А.С., В.В. Кулямин. Параллельные вычисления на графе. Программирование, том 41, № 1, 2015 г., стр. 3-20.

[12]. Burdonov I.B., Kossatchev A.S., Kuliamin V.V. Parallel calculations by automata on direct and back spanning trees of a graph. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 6, 2014 г., pp 63-66. DOI: 10.15514/ISPRAS-2014-26(6)-5

[13]. Бурдонов И.Б., Косачев А.С. Мониторинг динамически меняющегося графа. Труды ИСП РАН, том 27, вып. 1, 2015 г., стр. 69-96. DOI: 10.15514/ISPRAS2015-27(1)-5

[14]. Бурдонов И.Б., Косачев А.С. Параллельные вычисления на динамически меняющемся графе. Труды ИСП РАН, том 27, вып. 2, 2015 г., стр. 189-220. DOI: 10.15514/ISPRAS-2015-27(2)-12

[15]. Бурдонов И.Б., Косачев А.С. Исследование ориентированного графа коллективом неподвижных автоматов. Программная инженерия, том 8, № 1, 2017 г., стр. 16-25

[16]. Бурдонов И.Б. Обход неизвестного ориентированного графа конечным роботом. Программирование, том 30, №4, 2004 г., стр.11-34.

[17]. Бурдонов И.Б. Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом. Программирование, том 30, №6, 2004, стр.6-29.

[18]. Lynch, Nancy A.: Distributed algorithms. The Morgan Kaufmann Series in Data Management Systems. Kaufmann, San Francisco, Calif., 1996, pp. 904. ISBN 1-55860-348-4.

[19]. David Peleg. Distributed computing — A Locality-sensitive approach. SIAM Monographs on Discrete Mathematics and Applications. 2000, 359 pp.

[20]. Camil Demetrescu, Irene Finocchi, and Giuseppe F. Italiano. Dynamic Graphs. In Handbook of Data Structures and Applications. Oct 2004, 36-1 -36-20. ISBN: 978-1-58488-435-4.

[21]. Кушниренко А.Г., Лебедев Г.В. Программирование для математиков, Наука, Главная редакция физико-математической литературы, Москва, 1988.

[22]. Tary G. Le probl`eme des labyrinthes. Nouv Ann Math 14, 1895.

Bitcoin Users Deanonimization Methods

S.M. Avdoshin <savdoshin@hse.ru>
A.V. Lazarenko <avlazarenko @edu.hse.ru>
School of Software Engineering,
National Research University Higher School of Economics,
20, Myasnitskaya st., Moscow, 101000 Russia

Abstract. Bitcoin is the most popular cryptocurrency on the planet. It relies on strong cryptography and peer-to-peer network. Bitcoin is gaining more and more popularity in criminal society. That is why Bitcoin is often used as money laundering tool or payment method for illegal products and services. In this paper we explore various methods for Bitcoin users deanonimization, which is an important task in anti-money laundering process and cybercrime investigation.

Keywords: bitcoin; cryptocurrency; deanonymization

DOI: 10.15514/ISPRAS-2017-30(1)-6

For citation: Avdoshin S.M., Lazarenko A.V. Bitcoon Users Deanonimization Methods. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 89-102. DOI: 10.15514/ISPRAS-2018-30(1)-6

1. Introduction

Cryptocurrencies and blockchain technology are gaining more and more popularity nowadays. Besides hype around technology, there are a lot of useful and practical properties around namely, they are

- Decentralization – peer-to-peer networks without central authority are used for maintaining blockchain systems
- Transparency – every transaction and public address are accessible for everyone in the network.
- Irreversibility – any transaction is irreversible so it is very hard to rewrite the history of transactions.

Cryptocurrencies and blockchain technology are a perfect tool for mitigating the need of a middleman for transaction processing. Blockchain systems are leveraging the necessity of central authority. Transactions transparency helps to prevent theft and fraud, thus, it becomes crucially important for government and voting systems.

Bitcoin currently is the most widespread cryptocurrency on the planet. Bitcoin is a decentralized electronic payment system, which was introduced by a man or a group of people using an alias of Satoshi Nakamoto [1]. Bitcoin is based on peer-to-peer network and probabilistic distributed consensus protocol. Electronic coin is defined as a chain of digital signatures. If Alice wants to send bitcoins to Bob, then she should digitally sign a hash of the previous transaction and the public key of the next owner and add this information to the end of the coin.

The source and destination addresses in bitcoin are defined as hashes of public keys. Hashes are providing users with a certain degree of anonymity. All the transactions in bitcoin are public, so the bitcoin is pseudo-anonymous system.

Transactions in Bitcoin are processed to verify their integrity, authenticity and correctness by a group of so-called Miners (bitcoin nodes used for transaction verification and creation of new blocks). Transactions are grouped into the block and then processed by the Miners. Miner advertises a block to the rest of the network by appending it to the end of the blockchain. If the block is successfully verified by other Miners, the block is added to the end of the blockchain, and miner that proposed the block is earning a reward (fixed price for new block mining and corresponding transaction fees).

There are five [2] major components in bitcoin system:

- Users that create new wallets, transfer payments and save bitcoins on exchanges; typically they use one of the publicly available bitcoin clients;
- Miners that mine bitcoin blocks and process transactions; miners invest money into hardware that mine bitcoins and install specific software for that purpose;
- Testers, developers and entrepreneurs are improving bitcoin system and proposing new features; they are forming a specific technical community around bitcoin ecosystem;
- Bitcoin exchanges are the places where fiat money can be exchanged for bitcoins; typical examples are Poloniex [3], Bitstamp [4], Localbitcoins [5];
- Wallets store users' coins and provide features for making payments via internet connection.

Any user can create a fresh new Bitcoin wallet with a single click of the mouse without revealing any personal information like email, phone number or name. That is why Bitcoin is a perfect payment method for illegal activities. While being pseudo anonymous, Bitcoin helps to preserve the anonymity of the criminal in an easy way.

There were a lot of cases, where Bitcoin was used as a payment method, money laundering tool or a specific target for hackers:

- Silk Road was the most widespread darkmarket collapsed by the FBI; the Silk Road showed the society that Bitcoin could be used for criminal transactions as well as legitimate transactions [6];

- WannaCry is a ransomware cryptoworm, which targeted computers running Microsoft operating system demanding ransom payments in Bitcoins [7];
- BTC-e cryptocurrency exchange managed by Russian citizens; the exchange was shut down by USA for money laundering [8];
- Numerous ICO hacking cases; for example, hackers stole 7 million dollars from CoinDash ICO [9].

In order to fight with Bitcoin related crimes, different governments are enhancing law enforcement and make special training on cryptocurrency.

The following paper is an overview of existing methods. We propose novel classification for deanonimization methods.

2. Threat Model

The main goal of the attacker is to tie real names or IP addresses to transactions and bitcoin addresses. Instead of real name an attacker could tie email, phone number, username or any other digital identifier to the transaction.

An attacker is able to access all public information on the Internet: private and public forums, websites and social networks. Thus, an attacker could reveal real name of the particular person by parsing all the available information. Another approach is to "overhear" imprecise transaction information from users [10]. For example, an attacker may overhear "Alice, its Bob. I will send you 45$ bitcoins tomorrow morning".

Besides public information, an attacker can inject malicious bitcoin nodes into the network in order to eavesdrop IP addresses and try to link certain transactions to the client IP addresses. If the attacker will use both attack vectors together, he will be able to increase the deanonimization accuracy. Almost every deanonimization method consists of two phases: data collection phase and data analysis phase. Data collection could be online: using malicious bitcoin nodes to eavesdrop on traffic and address propagation mechanism. Data collection could be offline: for analyzing historical data parsed from blockchain an attacker doesn't have the need to use any additional components besides bitcoin client.

3. Deanomization

The deanonimization process is the process of linking public bitcoin address to the digital identifier of the user or his IP address. The process itself is divided onto two layers: P2P network layer and transactions layer.

It is very important to define the owner of the bitcoin public address.

We will define the owner of the bitcoin address as holder of the corresponding private key. So, if the exchange or online website is used for transferring bitcoins, and user doesn't have access to the private key, the exchange or website is the owner of the bitcoin address.

The ownership of bitcoin wallet is a complicated task. For example, if the exchange is holding the private key of a particular person then she is not able to control her bitcoins directly. Instead, the user is using external service which is managing the corresponding private key.

Deanonimization methods (see Fig. 1) could be divided into two categories they are passive and active ones.

Passive methods don't interact directly with bitcoin peer-to-peer network. Passive methods only use data which is parsed from the blockchain or any other public information source. Passive methods often rely on comprehensive graph analysis techniques and various heuristics related to bitcoin protocol.

Active methods are using malicious bitcoin nodes and social engineering techniques. Malicious nodes are the nodes with modified software under control of the attacker. Such nodes are used for traffic interception or direct communication with other peers in the network. Social engineering attacks are suitable for deanonimization of partially unknown users in the transaction chain.

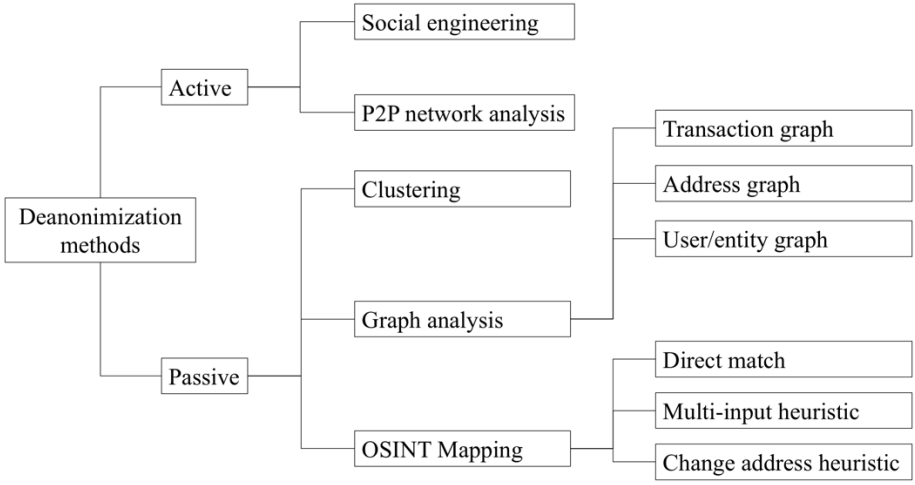


Fig. 1. Deanonimization methods classification

3.1 Open source intelligence on Bitcoin wallets

Without external information collected from various sources there is no way of finding the owner of the bitcoin wallet.

Information gathering stage can be classified into two different categories, namely, they are passive gathering and active gathering.

With active gathering an attacker is trying to find public address of the target by direct communication with it. Direct communication is an attempt to establish contact with target and find out the address during conversation or a request for payment [11]. This method is the most reliable, because a seller will not lie about his public address after the deal.

Another active approach uses malicious bitcoin nodes for traffic eavesdropping. This scenario will help to an attacker to collect IP addresses.

In passive gathering an attacker is trying to collect data from various data sources being in public access. There are various categories of data sources where an attacker can find digital names of bitcoin public address users: websites, forums, social networks, mining pools, wallets, bank exchanges, non-bank exchanges, vendors, gambling, laundry services. There are various information aggregators related to bitcoin wallets accessible online [12, 13].

The main goal of the information gathering stage is to collect as much tags for bitcoin wallets as possible because almost every deanonimization technique will be much more efficient in the wild with solid background on wallets participated in transactions.

3.2 Passive methods

3.2.1 Direct match

This is the easiest deanonimization method. An attacker is trying to find the owner of the bitcoin address by searching it in public sources. In case of success an attacker will find the corresponding digital identity.

3.2.2 Multi-input heuristic

Authors of the paper [14] proposed multi-input transaction heuristics. Multi-input transaction occurs when user wishes to perform a payment, and the payment amount exceeds the value of each of the available BTCs in user's wallet. Existing Bitcoin clients choose a set of BTCs from user's wallet and perform the payment through multi-input transactions. The straightforward conclusion is that if these BTCs are owned by different addresses, then the input addresses belong to the same user.

3.2.3 Change address heuristic

Change ("Shadow") addresses: bitcoin network generates a new address, so-called "shadow" address [1], at which each sender can collect back the "change". Using this heuristic, we can easily find the initial users wallet. Change addresses is the mechanism used to give money back to the input user in a transaction as bitcoins can be divided only by being spent.

All heuristic based methods heavily rely on direct match technique. Without properly collected data all the heuristics are useless for searching the real name of the person

3.2.4 Clustering

Authors of paper [11] proposed clustering techniques based on two previous heuristics. Using the first heuristic researchers were able to partition the network into 5,579,176 clusters of users (they started with 12 056 684 public keys). Authors used transaction graph and address graphs.

Authors of paper [11] enhanced second heuristics proposed by [14]. If an attacker can identify change addresses, she can therefore potentially cluster not only the input addresses for a transaction (according to Heuristic 1), but also the change address and the user himself. In addition, in custom usage of the Bitcoin protocol it is possible to specify the change address for a given transaction. Thus far, one common usage of this setting that authors of [11] have observed has been to provide a change address that in fact is the same as the input address.
Overall the authors proposed a new clustering heuristic based on change address, allowing us to cluster addresses belonging to the same user. Using proposed technique researchers were able to identify major institutions (like exchanges and gambling websites) and interaction between them using only a small number of identified transactions.

3.2.5 Fingerprinting

In work [15] authors show that third-party web tracker can deanonimize users of cryptocurrencies. For example, when someone is paying on the shopping website, there is enough information to deanonimize a person in future. Since the online tracking is a very comprehensive and efficient tool in modern internet, the leakage of bitcoin payment data is a serious threat for today's users.
There are two options in fingerprinting process:

- Single transaction linkage. The purpose of the attack is to link a web user to a transaction on the cryptocurrency blockchain. If the tracker has access to the receiving address, it trivially enables linkage. Another case is when tracker knows approximate price and time of transaction. Attacker just search the logs of transactions.
- Cluster intersection. Complementary attack where the adversary aims to identify the cluster of addresses in the victims' Bitcoin wallets. The aim of the attack is to link two purchases of the same users to the blockchain. The further processing just uses known graph intersection attack methods.

3.2.6 Deanonimization with graph analysis

Bitcoin wallet owner's privacy is a very fragile thing. Once broken it is very hard to get it back. Public address is anonymous only when nobody knows the owner of the address. That is why it is highly recommended to use new bitcoin address for every new payment.
In combination with described passive methods graph analysis can help an attacker to reveal the real identity of the bitcoin wallet. For example, if we know intermediaries in the chain, we can use that information to manually find the real name using social networks or social engineering techniques.
Another example of graph analysis is community detection algorithms and centrality metrics. We can detect the community of friends or neighbors, find people in the middle of the chain who are implicated in illegal activity.

Authors of paper [16] used Page Rank on Directed Address graph. The main purpose of it was to determine the most interesting nodes. The technique is able to determine large Bitcoin gambling websites and marketplaces.
We are assured that sophisticated deanonimization techniques designed for social networks will also work on bitcoin transactions graph. That could increase the percentage of deanonimized users significantly.
Graphs that are described later are the main tool for passive bitcoin addresses deanonimization process.

3.2.7 Transaction graph

The whole blockchain can be viewed as acyclic transaction graph (see fig. 2) G = {T, E}, where T is a set of transactions stored in the blockchain, E – set of unidirectional edges between these transactions. G represents the flow of coins between transactions in the blockchain over time.
The set of input and output coins in a transaction can be viewed as weights on the edges of G. In particular, each incoming edge in a transaction carries a timestamp and the number of coins that forms an input for these transactions.
Transaction graph is the main graph in deanonimization attacks. Address graph and user/entity graph are constructed using transaction graph.

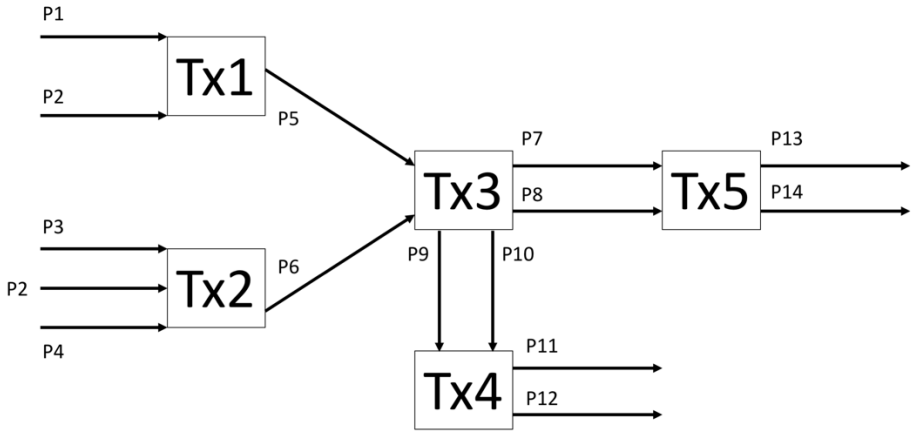


Fig. 2. Bitcoin transaction graph

3.2.8 Address graph

While traversing the transaction graph we can easily infer the relationship between various input and output addresses (public keys and these relations can be used for generation of address graph (see fig. 3), G = {P, E}, where P is a set of Bitcoin addresses and E are the edges connecting these addresses.

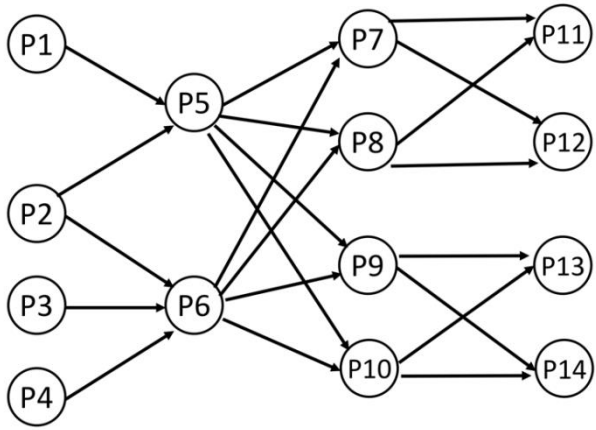


Fig. 3. Bitcoin addresses graph

3.2.9 User/entity graph

By using the address graph along with a number of heuristics, which are derived from Bitcoin protocol, the next step is to create an entity graph (see fig. 4) by grouping addresses that seem to belong to the same user.

3.3 Active methods

3.3.1 Social engineering

This method is quite exotic in case of bitcoin users deanonimization. However, it works perfectly in case of investigations. For example, if you only know the one-time pseudonym and associated bitcoin address, you can perform social engineering attacks.

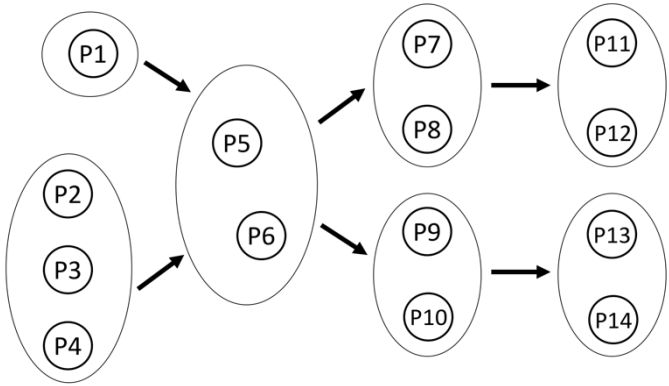


Fig. 4. Bitcoin user/entity graph

3.3.2 P2P network analysis

The Bitcoin P2P network contains two classes of nodes: servers and clients. Clients are nodes that do not accept in- coming TCP connections (e.g., nodes behind NAT), whereas servers do accept incoming connections. Clients and servers have different networking protocols and anonymity concerns. For instance, clients do not relay transactions. The focus of the deanonimization techniques is on servers.

All the attacks on P2P network are based on transaction broadcasting mechanism. An attacker should capture the IP address that initiated the transaction broadcast. If an attacker has resources, which are similar to the ISP, the IP-based deanonimization vector could be a powerful tool though.

Various researchers used gossip-based flooding protocols [17] to show that it is possible to deanonimize users using the linkage of users IP address with his pseudonym in the Bitcoin network.

In 2015 Bitcoin community responded to proposed attack by changing the network's flooding mechanism to a different protocol known as diffusion.

The attacks used "supernode" that is connected to active Bitcoin nodes and listens to the transaction traffic relayed by hones nodes. Using this technique, the linkage accuracy was up to 30% [17].

New version of protocol uses independent exponential delays. However, researchers in [17, 18] argue that it is unclear if such change actually protects against proposed attacks [17]. In diffusion spreading, each source or relay node transmits the message to each of its uninfected neighbors with an independent, exponential delay of rate λ.

The attacks in [17] use a supernode that is connected to most of the servers in the Bitcoin network. The supernode can make multiple connections to each honest server, with each connection coming from a different (IP address, port). Hence, the honest server does not realize that the supernode's connections are all from the same entity. The supernode can compromise arbitrarily many of a server's unused connections, up to the hard limit of 125 total connections.

The supernode in [17] also observes the timestamps at which messages are relayed from each honest server. Since the adversary maintains multiple active connections to each server, it receives the message multiple times from each server.

Supernodes are used for transaction and IP address matching via guessing the correct entry node set of a particular user. The supernode is trying to intercept clients IP propagation and correlate it with announced transaction. This attack comes for IP address spreading mechanism in Bitcoin. Such an attack achieves 86% IP matching probability on testnet (34% in average on the main net in 2013).

The paper [18] shows that new protocol is not effective against peer-to-peer traffic monitoring attacks.

4. Commetrcial Implementation

There exist various commercial and scientific tools closely related to this topic.

- BitIodine [19] – an open and modular blockchain analysis framework that allows to perform complex queries on transaction, group addresses together by controlling entity, and build clusters on top of blockchain data.
- BitConeView [20] is a graphical tool for the analysis of flows in the blockchain.
- Startups: Chainalysis, Blockchain Intelligence Group, Elliptic, Blockseer.

5. Future work

In future, we are going to try state-of-the-art social network deanonimization techniques on bitcoin transactions graph and adopt existing algorithms to fight with mixers. We will use graph based deanonimization techniques on other popular cryptocurrencies as well in order to prove that this technique is feasible with any cryptocurrency that is not especially designed for anonymous payment purposes.

Another open question is the possibility of achieving good accuracy on main net using P2P network level deanonimization techniques with significant amount of resources.

6. Conclusion

We have presented Bitcoin deanonimization techniques and provided classification of them.

Bitcoin privacy is an emergent field of research, which is gaining more and more attention from researchers all over the world. Being a popular payment tool along criminals, the society needs tools and suitable laws to fight with illegal usage of bitcoin.

References

[1]. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System [Electronic resource]. Bitcoin [Official website]. URL: https://bitcoin.org/bitcoin.pdf (accessed: 22.10.2017).

[2]. M. Conti, E.S. Kumar, C. Lal, S. Ruj. A Survey on Security and Privacy Issues of Bitcoin [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/abs/1706.00916 (accessed: 22.10.2017)

[3]. Poloniex – Bitcoin/Digital Asset Exchange [Electronic resource]. Poloniex [Official website]. URL: https://poloniex.com (accessed: 22.10.2017)

[4]. Bitstamp – buy and sell bitcoin [Electronic resource]. Bitstamp [Official website]. URL: https://www.bitstamp.net (accessed: 22.10.2017)

[5]. Localbitcoins [Electronic resource]. Localbitcoins [Official website]. URL: https://localbitcoins.com/ru/ (accessed: 22.10.2017)

[6]. M.Santori. Silk Road Goes Dark: Bitcoin Survives Its Biggest Market's Demise [Electronic resource]. Coindesk [Official website]. URL: https://www.coindesk.com/bitcoin-milestones-silk-road-goes-dark-bitcoin-survives-its-biggest-markets-demise/ (accessed: 22.10.2017)

[7]. WannaCry ransomware attack [Electronic resource]. Wikipedia [Official website]. URL: https://en.wikipedia.org/wiki/WannaCry_ransomware_attack (accessed: 22.10.2017)

[8]. J.J.Roberts. Bitcoin Site Fined $110 Million for Money Laundering, Owner Arrested for Hacking [Electronic resource]. Fortune [Official website]. URL: http://fortune.com/2017/07/27/btc-e-digital-currency/ (accessed: 22.10.2017)
[9]. W.Zhao. $7 Million Lost in CoinDash ICO Hack [Electronic resource]. Coindesk [Official website]. URL: https://www.coindesk.com/7-million-ico-hack-results-coindash-refund-offer/ (accessed: 22.10.2017)
[10]. M.Fleder, M.S.Kester, S.Pillai. Bitcoin Transaction Graph Analysis [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/pdf/1502.01657.pdf (accessed: 22.10.2017)
[11]. S.Meiklejohn, M.Pomarole, G.Jordan. A Fistful of Bitcoins: Characterizing Payments Among Men with no Names [Electronic resource]. UC San Diego [Official website]. URL: https://cseweb.ucsd.edu/~smeiklejohn/files/imc13.pdf (accessed: 22.10.2017).
[12]. Bitcoin Address Tags [Electronic resource]. Blockchaininfo [Official website]. URL: https://blockchain.info/ru/tags (accessed: 22.10.2017)
[13]. Bitcoin Address Checker [Electronic resource]. BitcoinWhosWho [Official website]. URL: http://bitcoinwhoswho.com (accessed: 22.10.2017)
[14]. E. Androukli, G.O. Karame Evaluating User Privacy in Bitcoin [Electronic resource]. Cryptology ePrint Archive [Official website]. URL: https://eprint.iacr.org/2012/596.pdf (accessed: 22.10.2017)
[15]. S.Goldfeder. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/pdf/1708.04748.pdf (accessed: 22.10.2017)
[16]. M.Fleder, M.S. Kester, S. Pillai. Bitcoin Transaction Graph Analysis [Electronic reosurce]. Cornell University Library [Official website]. URL: https://arxiv.org/abs/1502.01657 (accessed: 22.10.2017)
[17]. A. Biryukov, D. Khovratovich. Deanonymisation of clients in Bitcoin P2P network [Electronic resource]. ACM DL [Official website]. URL: https://dl.acm.org/citation.cfm?id=2660379 (accessed: 22.10.2017)
[18]. G. Fanti, P.Viswanath. Anonymity Properties of the Bitcoin P2P Network [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/abs/1703.08761 (accessed: 22.10.2017)
[19]. M. Spagnuolo, F. Maggi. BitIodine: Extracting Intelligence from the Bitcoin Network [Electronic resource]. FC & DS 2014 [Official website]. URL: http://fc14.ifca.ai/papers/fc14_submission_11.pdf (accessed: 22. 10. 2017)
[20]. BitConeView – first graphical tool for the analysis of flows in blockchain [Electronic resource]. BitConeView [Official website]. URL: http://www.bitconeview.info (accessed: 22.10.2017)

Методы деанонимизации пользователей биткоин

С.М. Авдошин <savdoshin@hse.ru>
А.В. Лазаренко <avlazarenko @edu.hse.ru>
Департамент программной инженерии,
Национальный исследовательский университет "Высшая школа экономики",
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Bitcoin является самой популярной криптовалютой на планете. В основе Bitcoin лежат криптография и пиринговая сеть. Будучи псевдоанонимной криптовалютой, Bitcoin очень часто используется преступным сообществом для отмывания денег или оплаты нелегальных товаров и услуг. В данной работе мы представляем ращличные методы для деанонимизации пользователей Bitcoin, что является чрезвычайно важной задачей при расследовании киберпреступлений и противодействию отмыванию денег.

Ключевые слова: биткоин; криптовалюты; деанонимизация.

DOI: 10.15514/ISPRAS-2018-30(1)-6

Для цитирования: Авдошин С.М., Лазаренко А.В. Методы деанонимизации пользоватетелей биткоин. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 89-102. DOI: 10.15514/ISPRAS-2018-30(1)-6

Список литературы

[1]. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System [Electronic resource]. Bitcoin [Official website]. URL: https://bitcoin.org/bitcoin.pdf (дата обращения: 22.10.2017).
[2]. M. Conti, E.S. Kumar, C. Lal, S. Ruj. A Survey on Security and Privacy Issues of Bitcoin [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/abs/1706.00916 (дата обращения: 22.10.2017)
[3]. Poloniex – Bitcoin/Digital Asset Exchange [Electronic resource]. Poloniex [Official website]. URL: https://poloniex.com (дата обращения: 22.10.2017)
[4]. Bitstamp – buy and sell bitcoin [Electronic resource]. Bitstamp [Official website]. URL: https://www.bitstamp.net (дата обращения: 22.10.2017)
[5]. Localbitcoins [Electronic resource]. Localbitcoins [Official website]. URL: https://localbitcoins.com/ru/ (дата обращения: 22.10.2017)
[6]. M.Santori. Silk Road Goes Dark: Bitcoin Survives Its Biggest Market's Demise [Electronic resource]. Coindesk [Official website]. URL: https://www.coindesk.com/bitcoin-milestones-silk-road-goes-dark-bitcoin-survives-its-biggest-markets-demise/ (дата обращения: 22.10.2017)
[7]. WannaCry ransomware attack [Electronic resource]. Wikipedia [Official website]. URL: https://en.wikipedia.org/wiki/WannaCry_ransomware_attack (дата обращения: 22.10.2017)
[8]. J.J.Roberts. Bitcoin Site Fined $110 Million for Money Laundering, Owner Arrested for Hacking [Electronic resource]. Fortune [Official website]. URL: http://fortune.com/2017/07/27/btc-e-digital-currency/ (дата обращения: 22.10.2017)

[9]. W.Zhao. $7 Million Lost in CoinDash ICO Hack [Electronic resource]. Coindesk [Official website]. URL: https://www.coindesk.com/7-million-ico-hack-results-coindash-refund-offer/ (дата обращения: 22.10.2017)
[10]. M.Fleder, M.S.Kester, S.Pillai. Bitcoin Transaction Graph Analysis [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/pdf/1502.01657.pdf (дата обращения: 22.10.2017)
[11]. S.Meiklejohn, M.Pomarole, G.Jordan. A Fistful of Bitcoins: Characterizing Payments Among Men with no Names [Electronic resource]. UC San Diego [Official website]. URL: https://cseweb.ucsd.edu/~smeiklejohn/files/imc13.pdf (дата обращения: 22.10.2017).
[12]. Bitcoin Address Tags [Electronic resource]. Blockchaininfo [Official website]. URL: https://blockchain.info/ru/tags (дата обращения: 22.10.2017)
[13]. Bitcoin Address Checker [Electronic resource]. BitcoinWhosWho [Official website]. URL: http://bitcoinwhoswho.com (дата обращения: 22.10.2017)
[14]. E. Androukli, G.O. Karame Evaluating User Privacy in Bitcoin [Electronic resource]. Cryptology ePrint Archive [Official website]. URL: https://eprint.iacr.org/2012/596.pdf (дата обращения: 22.10.2017)
[15]. S.Goldfeder. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/pdf/1708.04748.pdf (дата обращения: 22.10.2017)
[16]. M.Fleder, M.S. Kester, S. Pillai. Bitcoin Transaction Graph Analysis [Electronic reosurce]. Cornell University Library [Official website]. URL: https://arxiv.org/abs/1502.01657 (дата обращения: 22.10.2017)
[17]. A. Biryukov, D. Khovratovich. Deanonymisation of clients in Bitcoin P2P network [Electronic resource]. ACM DL [Official website]. URL: https://dl.acm.org/citation.cfm?id=2660379 (дата обращения: 22.10.2017)
[18]. G. Fanti, P.Viswanath. Anonymity Properties of the Bitcoin P2P Network [Electronic resource]. Cornell University Library [Official website]. URL: https://arxiv.org/abs/1703.08761 (дата обращения: 22.10.2017)
[19]. M. Spagnuolo, F. Maggi. BitIodine: Extracting Intelligence from the Bitcoin Network [Electronic resource]. FC & DS 2014 [Official website]. URL: http://fc14.ifca.ai/papers/fc14_submission_11.pdf (дата обращения: 22. 10. 2017)
[20]. BitConeView – first graphical tool for the analysis of flows in blockchain [Electronic resource]. BitConeView [Official website]. URL: http://www.bitconeview.info (дата обращения: 22.10.2017)

The Principles of Life Cycle Supporting System for Mission-Critical Systems

B.A. Pozin <bpozin@ec-leasing.ru>
National Research University Higher School of Economics,
20 Myasnitskaya Ulitsa, Moscow, 101000, Russia
EC-leasing Company,
125 Varshavskoye shosse, Moscow, 117405, Russia

Abstract. The set of outline and guidelines as well as tools for automation to ensure continuity of business-continuity in the lifecycle of mission critical systems are considered. This complex is called the Life Cycle Supporting System (LCSS). The aim of the system is to reduce the risk level of the realization of critical errors in the system and application software throughout the life cycle of mission critical system, reducing operational risks and total cost of ownership of mission critical system. LCSS in the ISO / IEC / IEEE 15288 terms is enabling system. LCSS is created for life cycle of mission critical system support in the organization-owner.

Keywords: enabling system; Information System Life Cycle; Information System Release; Life Cycle Supporting System (LCSS); LCSS infrastructure; Mission-Critical System; Release Management

DOI: 10.15514/ISPRAS-2018-30(1)-7

For citation: Pozin B.A. The Principles of Life Cycle Supporting System for Mission – Critical Systems. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 103-114. DOI: 10.15514/ISPRAS-2018-30(1)-7

1. Problems and risks in the life cycle of mission – critical systems

Mission-critical information systems (MCS) quite common in the financial field, public administration, retail and wholesale trade, transport and other industries where information systems are deeply embedded in the business processes.

As a rule, they perform the functions of “back office”, i.e. accounting for financial and material flows, the analytical systems of the state or corporate level, or other functionality that is crucial to the business. Such systems are increasingly being

built as centralized, have the contour of information security, as processed and accumulated in the system data have a state or commercial secret.

Upon completion of the development of MCS is the transfer of ownership of the MCS owner, the system is its asset. In this regard, the remainder of the life cycle of the system responsible for the quality of the operation and maintenance of the system at the right level of performance lies with the owner of the asset. The aim of organization-owner is to ensure business-continuity in the system lifecycle. For this type of activity in the organization-owner MCS must be put automated processes to ensure reduction of planned and unplanned downtimes, operational risks and total cost of ownership of MCS.

The need for systematic efforts to maintain compliance with the functional and performance characteristics of the destination MCS in the state responsible usability and thereby preserve the asset accounts for the fact that most of its value in the life cycle of the system belongs to does not development. Costs for maintenance and development in the life cycle (15-20 years and more) MCS application software constitute 70-80%, and the actual development of the order of 20%. At lower timing tracking (about 10 years) the share of maintenance costs remains significant (60% vs. 40% for the development). However, the major potential losses that need to reduce or minimize, the risks are being implemented during the phases of system operation and maintenance.

These primarily include the following risks:

- loss of efficiency MCS
    - due to lack of working out of their architecture, systems engineering platform, software and information systems;
    - due to incorrect support of their systems engineering platform;
- violation of the rules of functioning of the system of organization-owner (by activity) due to:
    - inadequate performance or availability of MCS (taking into account the characteristics of the existing and future load flows);
    - shortcomings allocation of responsibilities and duties of staff;
    - lack of trained staff;
- loss of integrity of MCS during their maintenance and development.

Experience shows that the main causes of the onset of risks in MCSs are the human factor and a poor understanding by managers of the complexity and labor-intensive process of formation and use of services, implemented MCS, conditions of their effective application. This leads to a substantial increase in the maintenance costs of their MCS and software that implements the basic functionality of the services included in the MCS. Neglecting the issues of ensuring the service MCS and application software (AS) during their operation, maintenance and development leads to unreasonably high additional costs and consequently to an increase in the value of the MCS lifecycle.

It is essential that after entering the MCS service or new services sales price risks of default functions required by the system or part of it (sub-system, service, etc.) increases significantly, since MCS is directly involved in the implementation of business functions. Failure or improper execution of a business function because of defects in the MCS leading to direct losses of business, which are much more significant - sometimes several orders of magnitude, than the cost of MCS-holder organization for the development of MCS and its AS. That's why the main risks for businesses using IP implemented in stages of operation and maintenance systems, this is a potentially major loss, which should be minimized.

The most important **goal** in the life cycle of MCS is **the design and implementation of systematic action to prevent or reduce the impact of such risks, primarily related to the presence of defects, with the untimely or inadequate quality of their elimination, with changes to the system and application software, with the modernization of equipment or machines of MCS totally.** Eliminating the effects of display defects requires an analysis of the problems encountered in hardware, software, personnel actions. To address the problems involved quite a large number of specialists, and problems of the resolution should be carried out as soon as possible. There is a real need to specifically deal with the methods of software life cycle of information systems, particularly in the area of MCSs serving medium and large business.

2. MCS Life Cycle Supporting System

Experience shows that the primary method of ensuring the specified MCS performance and reduce these costs is to create a life-cycle supporting system (LCSS), which establishes and regulates the life cycle processes and automating these processes, the so-called enabling system (supporting system). According to [1] LCSS as supporting the system is in addition to consideration of MCS throughout the stages of its life cycle, but not necessarily directly contribute to its functioning.

LCSS is the same system as the main MCS in the sense that it is being developed under a separate Requirements Specification and documented. Its purpose is to maintain the main MCS in a serviceable condition for any changes made to the MCS, due to both the development of its functionality, and its scaling or upgrade (change of operating system versions, hardware, architecture). LCSS operation makes it possible, for example, to maintain system-wide MCS characteristics while significantly increasing load compared to envisaged in the original MCS Technical Specifications, in which the main MCS was created. The objects provide LCSS are the main components of the MCS, their relationship, the types of the changes, as well as a team, carrying out support and MCS development.

Ensuring operation, maintenance and development of MCS in its life cycle is implemented team of specialists with different skills, so it is important to structure, organize and regulate the activities of this group. One of the most important LCSS functions is delineation of areas of responsibility of staff, the establishment of rules and regulation of the work of the team, expected results of certain operations in the

process of purposeful activity for the operation, maintenance and development of MCS at all stages of its life cycle [2,5].

MCSs automate the organization's activities, so their use is subject to internal rules and regulations, especially in the field of information security (IS). For this reason, the operation of such systems involved employees of the organization-owner or its operator, authorized by a certain set of documents and the types of activities that are essential in terms of the MCS lifecycle. Public organizations usually have a number of rules that restrict the use of material resources for authorized activities. In addition, various experts should be involved in the life cycle of MCS, including representatives of the various activities that shape the rules of work, automation needs to MCS requirements. By virtue of this LCSS construction must be based on certain principles described below:

- focuses on reducing total cost and complexity of operation, maintenance and development of MCS as a whole, not just on development speed of its constituent parts or systems;
- covers all the roles of personnel involved in the life cycle of MCS, and regulates the processes of the life cycle of MCS, both within the organization-owner and with cooperating organizations - contractors;
- supports processes and provide a systematic quality control, including for suppliers of hardware, software and services
- has as the focus interests of MCS organization-owner, it has to be created on the side of the owner and operated by and / or MCS operator;
- has to be deployed predominantly in the contour IS of MCS owner and provides secure interfaces with contractors;
- automates the processes of life cycle the most responsible that provides quality and labor-intensive processes which request attracting a large number of staff with relatively low qualification.

3. Processes of MCS Life Cycle Supporting System

When creating MCS Life Cycle Supporting System (LCSS) must take into account the impact that the requirements to the MCS functioning and development company and business experience, exploit MCS as well as suppliers of products and services involved in the creation of intellectual property and participating in its life cycle (Fig. 1).

In the operation of MCSs is necessary not only to respond quickly to emerging incidents, but also to carry out the current work to ensure the operation of backing up and restoring data, control loading system components to prevent abnormal situations, resource allocation settings to improve the performance provided by the IT - based services accumulating experience exploit MCS and user requests, etc. Requirements for MCS staff to ensure the operation reflected in the regulations and the operating instructions.

During the operation form as techniques and regulations accompanying the components of the system and describes how to interact with the operation of the service staff, develop new possibilities of MCS, as well as with the process of tracking the MCS AS for the transmission of the last information on the MCS and AS manifested defects. In addition, the accumulated experience of the implementation of automated business processes, there are proposals for the functional development of MCS, which are transmitted to the functional development process.

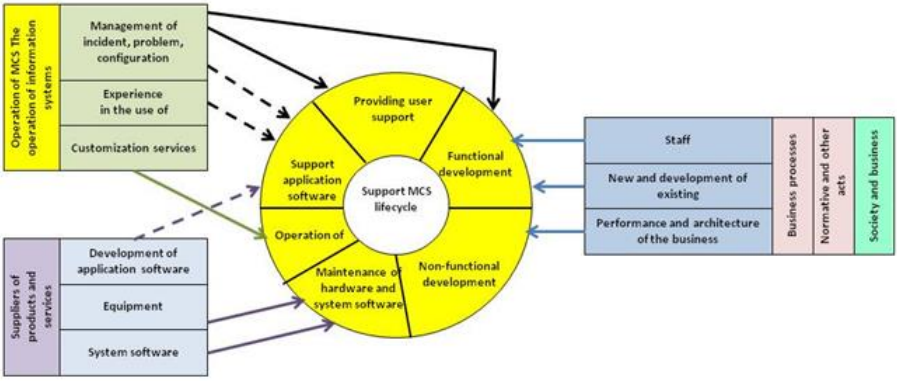


Fig.1. External factors and processes influenced to MCS LCSS

During the operation of the MCS vendors of included in the composition of the products and IT services are developing these products and services. Their plans for the release of new versions of the products do not depend on MCS development plans. Because of this, there is a need to replace the already operating as part of the MCS system software versions with new ones, which are announced as the best suppliers. The need for high availability MCS requires verification of compatibility of new versions of the system software with the other elements of the system, included in the composition of the technical means, as well as compatibility checks AS and new versions of system software. As for the tools used in the development of AS, the circuit works the same: you want to check compatibility of new versions of tools and architecture AS, non-infringement of AS functionality after the transition to new versions of the tools, such as software compatibility code generated by the new version of the compiler, while reassembly of MCS AS. This work is usually carried out on the stands for AS maintenance.

Society and business development leads to the fact that over time there needs to change MCS architecture due to changes in business architecture organization, due to the need of MCS productivity growth to hold its modernization, to automate new business processes or sub-processes, to implement requirements related to personnel development organization. These requirements are implemented functional development process (inclusion of new functions in the AS) and a non-functional

development (MCS modernization). Development needs arise in business (or the public administration bodies), especially for those businesses that are not quite satisfied with the existing level of automation of their operations. They arise in the event of new legislative acts and other documents that require changes in the way the data or generating analytical information (e.g., reporting) for business management or management of social processes.

LCSS regulates the composition, order of implementation and automation of the life cycle of MCS processes (see. Fig. 2). These processes for the projects, attracted by the staff, nature and results of operations are divided into two groups: processes of operation and processes for development.

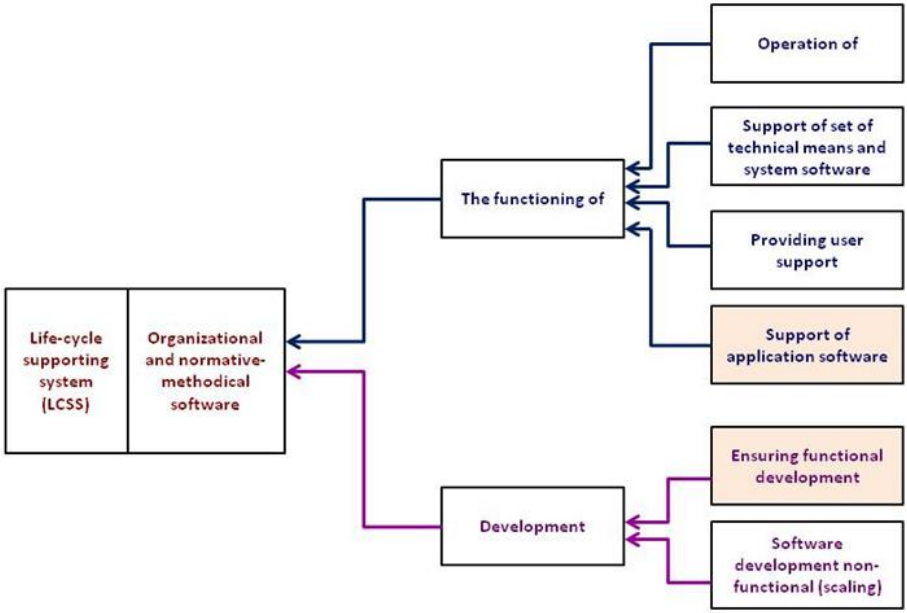


Fig. 2. Life Cycle Supporting System Processes

The first group consists of the following processes:

- ensuring the operation;
- provision of support of the technical means (Complex of Technical Means - CTM) and system software (SS);
- providing support to users;
- providing maintenance to AS [4,5]
- the composition of the second group consists of the following processes:
- providing MCS functional development;
- ensuring MCS non-functional development (scaling) [3].

The regulation process is carried out with the help of institutional and normative and methodological support - set of relevant documents. The composition and content of these processes, as well as the maintenance of the complex documents discussed below.

4. Content of MCS Life Cycle Supporting System

MCS Life Cycle Supporting System (Fig. 3) represents a set of:

- complex normative-methodical and organizational - administrative documents,
- personnel who carries out maintenance and development of MCS, its system and application software on all over the life cycle of MCS from idea creation to remove the system from the service,
- tools, databases and repositories to automate the activities of personnel involved in the provision of MCS lifecycle,
- infrastructure i.e. hardware and software tools used to automate activities of LCSS staff.

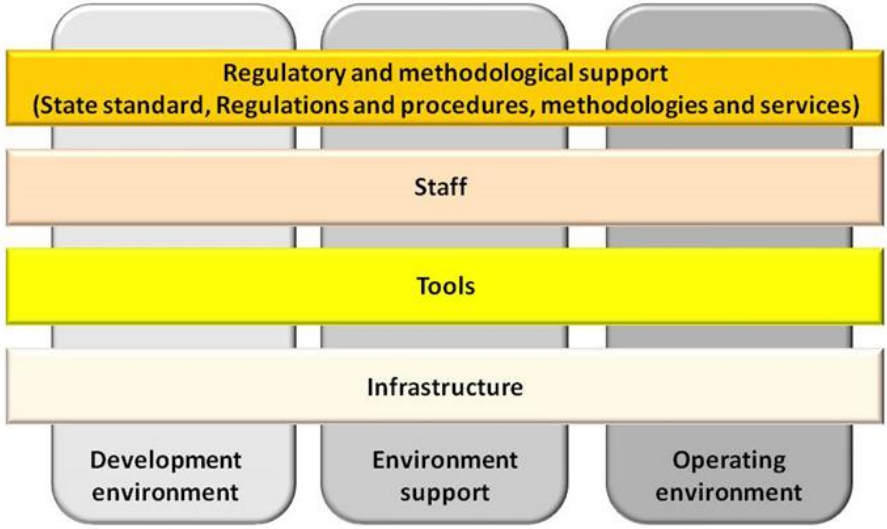


Fig.3. Content of MCS Life Cycle Supporting System

All components of this set of tools determines the actions and / or placed in one of three areas: the development area belonging to contractor (s), maintenance area belonging to the organization, the owner of the MCS or perhaps rented them fully or partially and operation area usually belonging to the organization, the owner of the MCS and a characteristic highest information security requirements. Maintenance

area in any case should be in the MCS Information Security contour. There must be defined security policy, data types and exchange procedures for transferring data from one area to another within the MCS lifecycle.

5. The complex normative-methodical and organizational – administrative documents

Enter the new MCS in the action leads to changes in the organization of MCS-owners. In most organizations, these changes of personnel roles and interaction of units are regulated in terms of the use of MCS in the changed business processes. However, the need to maintain the relevance of MCS for its use also requires changes in the organization:

- determining the order of formation of change requests MCS functionality,
- connection of new facilities to its use,
- organization of MCS exploitation and work with the new system by the users,
- retraining of end users to use the new functionality of MCS features (new services), etc.

These changes are introduced in the organizations in the form of normative - methodical guidance documents (for example, standards organizations) and organizational and administrative documents (orders, regulations, instructions, etc.).

The main purpose of the formation of the complex of these documents is to create policies and procedures that describe the delineation of areas of responsibility of the owner of MCS-organization of staff while ensuring its operation and development. It is expected that as a result of the changes recorded in this complex document, will reduce the total cost of labor input on the functioning of the organization and development of MCS. The documents should be regulated life cycle processes (see. Item 3) that run the personnel organization of the owner of the system in parallel with the development process, operation and maintenance of MCS. The documents must include a description of the role models of personnel in the performance of the MCS organization owner lifecycle processes, including life cycle stages, tasks each role, the results of each role in the phases of the stages, the procedure and the form of the transfer of results and accountability for results between the roles in the implementation processes.

Documents must also regulate:

- procedure of formation and fixing the functional requirements for the production of MCS release, the general rules of its planning, initiation, implementation, in cooperation with contractors, monitoring the progress of work and results, the procedure for release documenting and completing;

- kinds of products, of which the MCS is going, the procedure for their acceptance of contractors and input control as an integral part of MCS criteria;
- the procedure of implementation of MCS products, scope of work on the target platform, and with the use of information security;
- the procedure to implement the integration of products, control of the functional integrity of purpose and performance, documentation, the quality assurance of integrated systems on impersonal data and its tests are impersonal and real data;
- procedure to enter the tested system in operation.

The necessity and importance of creating such documents due to the fact that these activities are determined, ultimately, the ability to reduce total costs. In working out their estimated ability to perform work on their own organization or with the assistance of an authorized contractor (s) with the unconditional implementation of the IS requirements. It is important to recall that the vast majority of these activities is carried out within the IS contour of the organization.

The basis for the formation of normative and methodological support LCSS are basic and subsequent standards, the composition of which is shown in Fig. 4. These standards define the properties of processes and quality requirements as a life cycle processes and system organization and the quality of the owner of the MCS quality.

6. Conclusion

In the life cycle of MCS LCSS responsible role is extremely high, both from the standpoint of enabling system of conservation investments in MCSs, increasing its lifetime and reduce its total cost of ownership. Understanding this leads to the need to lay the necessity of creating and commissioning LCSS simultaneously with the development of a MCS. It is necessary to carry out the necessary investments in support of LCSS in the feasibility study on the establishment of a MCSs. What is important is the experience of about 10 years of practical development and use LCSS. He showed that the increase in the number of changes in the MCS around 40% did not lead to the urgent need to increase the staff responsible for the maintenance and functional development of middleware systems and responsible uses automated technology to carry out their functions. Of course, the complexity does not limit possibilities are endless. However, the very fact quite accurately describes the subject matter and result.

The key issue is the implementation of a road map for the establishment and commissioning of the MCS LCSS, namely the following key steps in the organization of IP-owners:

- There must be put in place regulations governing the interaction of functional units (functional customer) organization and IT – Service;
- There should be developed and put into action common to organize documents: Terms and regulations of interaction of staff in dealing with the

tasks of ensuring the functioning of the MCS, its maintenance and development, to ensure the distribution of responsibilities between people with all levels of management: strategic, tactical and operational and regulate this distribution establishing criteria for the evaluation of human activity;

- In addition to the infrastructure MCS operation must be regulated by a separate document and the infrastructure to ensure the functioning of the MCS, its maintenance and development, provides a process for the implementation of the modified AS of the MCS on the target platform, as well as releases of acceptance from the developer - the infrastructure without diverting operation resources;

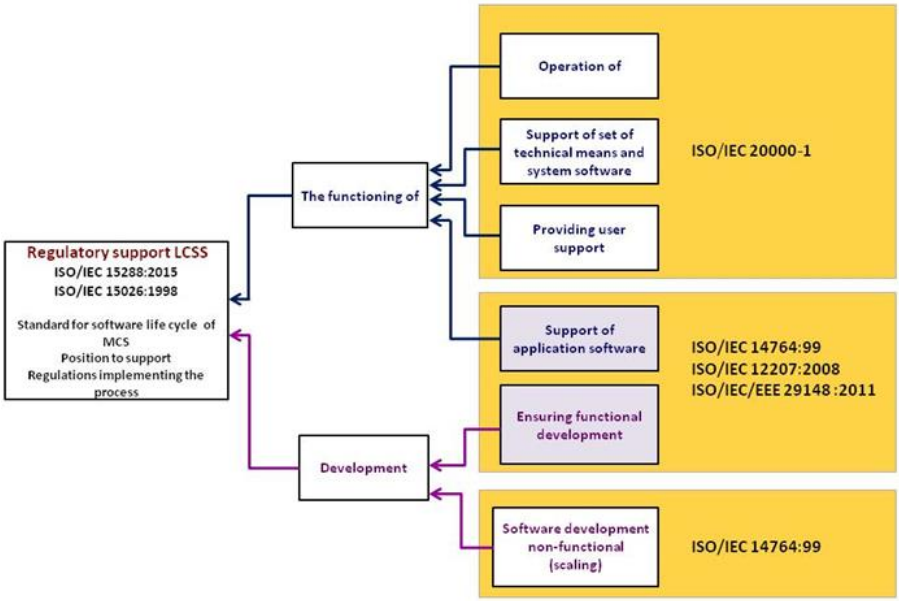


Fig. 4. Regulatory support MCS lifecycle.

- The most systemically – significant, time-consuming and repetitive processes should be automated, especially the requirements management processes, delivery of tasks and monitoring their implementation, planning and configuration management versions and releases of MCS AS, automated functional and load testing.

References

[1] ISO/IEC/IEEE 15288: 2015 Systems and software engineering. Systems Life Cycle Processes.

[2] ISO/IEC 14764:2006 Software Engineering – Software Life Cycle Processes – Maintenance
[3] Forsberg K., Mooz H., The Relationship of System Engineering to the Project Cycle. – Center for Systems Management, 1995, 12pp.
[4] Pozin B., Galakhov I. Models in Performance Testing. Programming and Computer Software, Vol. 37, No. 1, 2011, pp.15-25. DOI: 10.1134/S036176881101004X
[5] ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes

Принципы построения системы обеспечения жизненного цикла ответственных систем

Позин Б.А. <bpozin@ec-leasing.ru>
Национальный исследовательский университет Высшая школа экономики, 101000, г. Москва, ул. Мясницкая, д. 20
ЗАО «ЕС-лизинг», Варшавское шоссе 125, 117405, Москва, Россия

Аннотация. Рассмотрен комплекс мер и средств автоматизации для обеспечения непрерывности бизнеса в жизненном цикле ответственных систем. Этот комплекс получил название система обеспечения жизненного цикла (СОЖЦ). Целью системы является снижение уровня рисков от проявления критических ошибок в системном и прикладном программном обеспечении на всем жизненном цикле ответственной системы, снижение эксплуатационных рисков и совокупной стоимости владения ответственными системами. СОЖЦ в терминах ISO/IEC/IEEE 15288 представляет собой обеспечивающую систему (enabling system). СОЖЦ создается для обеспечения деятельности организации-собственника ответственной системы.

Ключевые слова: обеспечивающая система; система обеспечения жизненного цикла (СОЖЦ); ответственная система; процессы жизненного цикла информационной системы; выпуск ИС; управление выпусками; инфраструктура СОЖЦ.

DOI: 10.15514/ISPRAS-2018-30(1)-7

Для цитирования: Позин Б.А. Принципы построения системы обеспечения жизненного цикла ответственных систем. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 103-114. DOI: 10.15514/ISPRAS-2018-30(1)-7

Список литературы

[1] ISO/IEC/IEEE 15288: 2015 Systems and software engineering. Systems Life Cycle Processes
[2] ISO/IEC 14764:2006 Software Engineering – Software Life Cycle Processes – Maintenance
[3] Forsberg K., Mooz H., The Relationship of System Engineering to the Project Cycle. – Center for Systems Management, 1995, 12 p.

[4] Позин Б.А., Галахов И.В. Модели в нагрузочном тестировании. Программирование, том 37, №. 1, 2011, стр. 20-35

[5] ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes.

Применение AVX512-векторизации для увеличения производительности генератора псевдослучайных чисел

[1,2] *М.С. Гуськова <maria.guskova@rambler.ru>*
[1,2,3] *Л.Ю. Бараш <barash@itp.ac.ru>*
[1,2,3,4] *Л.Н. Щур <shchur@chg.ru>*
[1] *Научный центр РАН в Черноголовке*
142432 Черноголовка, Россия
[2] *Национальный исследовательский университет «Высшая школа экономики»,*
101000 Москва, Россия
[3] *Институт теоретической физики им. Л.Д. Ландау,*
142432 Черноголовка, Россия
[4] *Вычислительный центр им. А.А. Дородницына ФИЦ ИУ РАН, 119333*
Москва, Россия

Аннотация. В работе описывается применение наборов SIMD инструкций (Single Instruction Multiple Data) для параллелизации алгоритма генерации псевдослучайных чисел. Дан обзор расширений MMX, SSE, AVX2, AVX512, реализующих принцип SIMD. Приведен пример реализации генератора LFSR113 с использованием расширения AVX512. Приведен сравнительный анализ скоростей работы различных реализаций алгоритма.

Ключевые слова: псевдослучайные числа; SIMD инструкции; технология AVX-512.

DOI: 10.15514/ISPRAS-2018-30(1)-8

Для цитирования: Гуськова М.С., Бараш Л.Ю., Щур Л.Н. Применение AVX512-векторизации для увеличения производительности генератора псевдослучайных чисел. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 115-126. 10.15514/ISPRAS-2018-30(1)-8

1. Введение

Генерация равномерно распределенных случайных чисел необходима для компьютерного моделирования методами Монте-Карло и молекулярной динамики [1]. Для генерации случайных чисел используются генераторы псевдослучайных чисел (ГПСЧ). ГПСЧ использует детерминированные алгоритмы для вычисления чисел, но полученная таким способом последовательность обладает свойствами случайной последовательности. При этом к ГПСЧ предъявляются следующие требования [2].

- Последовательность псевдослучайных чисел должна быть равномерно распределенной, а подпоследовательности одинаковой длины должны быть равновероятными. С практической точки зрения, последовательность псевдослучайных чисел должна пройти набор статистических тестов на равномерное распределение и отсутствие корреляций, кроме того, желательно наличие теоретического обоснования хороших статистических свойств генератора.
- Период генератора должен быть достаточно большим.
- Последовательность должна быть воспроизводимой, чтобы можно было повторить эксперимент.
- Должен быть алгоритм пропуска кусков. Параллельные потоки могут использовать различные куски одной и той же последовательности.
- Должна существовать эффективная реализация генератора с точки зрения скорости вычислений и использования оперативной памяти.

Для ряда задач, использующих методы Монте-Карло, генерация случайных чисел занимает значительную часть вычислительного времени, и увеличение производительности генерации является важной задачей.

В настоящей работе для повышения производительности генератора случайных чисел будут использоваться операции SIMD (Single Instruction Multiple Data). Такие операции позволяют применять одну инструкцию одновременно к нескольким элементам данных. Мы детально рассмотрим применение нового расширения AVX512, которое поддерживается процессорами Intel, начиная с 2017 года.

2. Технология SIMD

С 1997 года по сегодняшний день, компания Intel выпустила 9 расширений набора инструкций архитектур Intel 64 and IA-32 для поддержки векторизации. Это MMX технология, расширения SSE, SSE2, SSE3, SSSE3 и SSE4, AVX, AVX2, AVX512 [3].

Каждое расширение содержит набор SIMD-инструкций для работы с упакованными целыми числами и с упакованными числами с плавающей точкой. Каждому расширению также соответствует свой набор регистров (см. рис.1).

Технология Intel MMX(Multimedia Extensions) впервые была реализована в процессорах семейства Intel Pentium MMX в 1997 году и предназначалась для эффективной обработки аудио- и видеопотоков. MMX инструкции позволяют работать с байтами, словами или двойными словами, расположенными в MMX регистрах. Всего таких регистров 8 (MMX0-MMX7), они 64-битные и соответственно могут содержать одновременно 8 байт или 4 слова, или 2 двойных слова, или 1 учетверенное слово. Это расширение может использоваться для обработки целочисленных массивов с помощью SIMD инструкций.

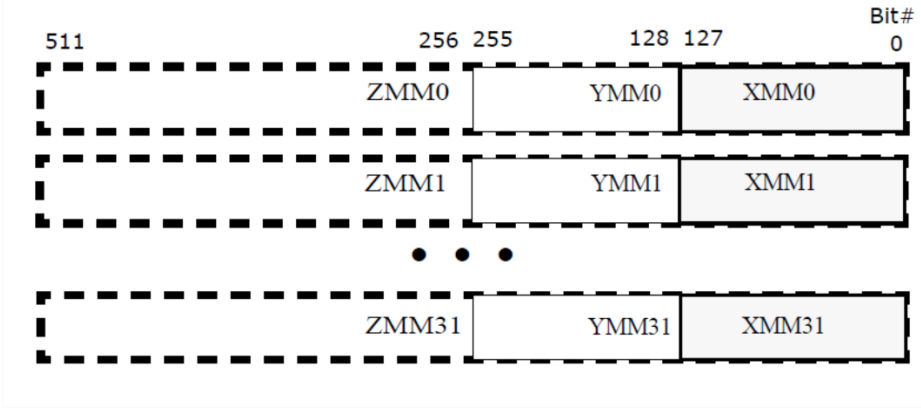


Рис. 1. Регистры XMM, YMM, ZMM
Fig. 1. Registers XMM, YMM, ZMM

Расширение SSE (Streaming SIMD Extensions) появилось в семействе процессоров Intel Pentium III и AMD Athlon XP в 1999 году. SSE является расширением MMX технологии. Расширение включает в себя 8 128-битных XMM регистров (XMM0-XMM7), набор инструкций для работы с числами с плавающей точкой одинарной точности, расположенных в XMM регистрах и набор инструкций для работы с целыми числами в MMX регистрах.

Расширение SSE2 было представлено в процессорах Intel Pentium 4 и Intel Xeon в 2001 году, а также в AMD Opteron и Athlon64 в 2003 году. Инструкции расширений позволяют обрабатывать числа с плавающей точкой двойной точности в XMM регистрах и целые числа, расположенные в регистрах MMX и XMM регистрах. Набор целочисленных инструкций в SSE2 содержит новые IA-32 SIMD операции для работы с 128-битными регистрами и расширяет существующие 64-битные инструкции до работы со 128 битами.

Расширение SSE3 впервые появилось в процессорах Intel Pentium 4 Prescott в 2004 году, а также в AMD Athlon64 в 2005 году. В SSE3 были добавлены 13 новых инструкций, которые улучшают производительность SSE, SSE2 и операций с плавающей точкой.

Расширение SSSE3 (Supplemental SSE) было представлено в процессорах Intel Xeon 5100 серий и семействе процессоров Intel Core 2 в 2006 году. Были добавлены 32 SIMD-инструкции для работы с целочисленными данными.

Расширение SSE4 содержит 54 новые инструкции: 47 из них содержится в SSE4.1 и 7 - в SSE4.2. Оно было впервые реализовано в процессорах семейства Intel Penryn в 2008 году и AMD Bulldozer в 2011 году.

Расширение AVX (Advanced Vector Extensions) является значительным улучшением расширений SSE. Оно поддерживается процессорами Intel и AMD, начиная с 2011 года. Новые YMM регистры - 256-битные. Кроме SIMD

инструкций над MMX регистрами, почти все 128-битные SIMD функции имеют AVX-эквивалент, который имеет трех-операндный синтаксис. Также добавлены совершенно новые инструкции, которых не было в предыдущих расширениях. Расширение использует новый префиксный код VEX, позволяющий работать с новыми YMM регистрами. Большинство инструкций не требует выравнивания по 16- или 32-битным границам. 256-битные AVX-инструкции используют трех-операндный синтаксис, а некоторые – четырех-операндный.

В расширение AVX2 добавлены целочисленные инструкции, ранее доступные только для 128-битных регистров. Здесь же появились новые функциональные операции распространения (broadcast) и перестановок (permute), сдвиговые операции, которые сдвигают элементы вектора на разное количество битов, для извлечения несмежных элементов из памяти. Расширение AVX2 поддерживается процессорами Intel и AMD, начиная с семейств Intel Haswell (2013) и AMD Excavator (2015).

Новый набор инструкций AVX512 для процессоров архитектуры x86-64 компания Intel представила в июле 2013 г., однако первое семейство поддерживающих эту технологию процессоров Intel Xeon Phi x200 (кодовое имя Knights Landing) было выпущено в июне 2016 г.

AVX512 подразделяется на несколько отдельных наборов, для каждого из которых существует свой идентификационный бит. Основной набор AVX512 Foundation включен во все реализации AVX512.

Список отдельных наборов инструкций AVX512:

- F (Foundation) - основной набор инструкций, расширяющий большинство существующих AVX инструкций для работы с 512-битными регистрами общего назначения;
- CDI (Conflict Detection Instructions) обеспечивает лучшую векторизацию циклов;
- ERI (Exponential and Reciprocal Instructions) - инструкции для работы с трансцендентными функциями (экспонента, логарифм, тригонометрические функции);
- PFI (Prefetch Instructions) - инструкции для префетчинга - предварительной загрузки данных и инструкций, что позволяет ускорять дальнейшее выполнение программы;
- BW (Byte and Word Instructions) - инструкции для работы с байтами и словами;
- DQ (Doubleword and Quadword Instructions) - инструкции для работы с двойными и четверными словами (32- и 64-битными целыми числами и числами с плавающей точкой);
- VL (Vector Length Extensions) обеспечивает возможность работы со 128- и 256-битными регистрами (младшими частями 512-битных регистров).

Наборы F, CDI, ERI, PFI впервые реализованы в процессорах Intel Xeon Phi x200 (Knights Landing) в 2016 году, а BW, DQ, VL – в процессорах семейства

Intel Xeon Scalable, выпущенных в 2017 году. Инструкции AVX512 предназначены для работы с 32 новыми 512-битными регистрами общего назначения (ZMM), расширяющими существующие 128- (XMM) и 256-битные (YMM) регистры. Регистры XMM0-XMM15 доступны для инструкций SSE, AVX-128 (AVX, AVX2); регистры YMM0-YMM15 - для AVX256 (AVX, AVX2); регистры ZMM0-ZMM31 - для AVX512F. Для доступа к регистрам XMM16-XMM31 и YMM16-YMM31 необходим набор инструкций AVX512VL.

Микроархитектура процессоров, реализующих AVX512, также подразумевает наличие 8 регистров-масок (k0-k7), позволяющих гибко работать с частями регистров общего назначения. Разрядность этих регистров равна 16 (64 в случае AVX512BW) и позволяет маскировать 16 элементов 512-разрядных регистров при работе с 32-разрядными числами с плавающей точкой или двойными словами. В случае работы с числами с плавающей точкой двойной точности или четверными словами используется только 8 бит масочных регистров.

Мы рассмотрим применение AVX512-векторизации на примере генератора псевдослучайных чисел LFSR113.

3. Генератор псевдослучайных чисел LFSR113

LFSR113 (Linear Feedback Shift Register) является комбинированным генератором из четырех сдвиговых регистров [4]. Метод генерации, основанный на сдвиговых регистрах, использует свойства линейных операций по модулю 2 над битами x_n. Рассмотрим последовательность нулей и единиц:

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \; mod \; 2 \qquad (1)$$

Это линейно-рекуррентное соотношение в поле Z_2 , состоящем из двух элементов, нуля и единицы. Оно называется сдвиговым регистром и имеет период длины $p = 2^k - 1$ тогда и только тогда, когда полином $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$, называемый характеристическим полиномом рекуррентной последовательности, является примитивным полиномом. Для одного сдвигового регистра выходная последовательность определяется, как $u_n = \sum_{k=1}^{L} x_{ns+k-1} 2^{-k}$, где размер шага s и длина слова $L -$ целые положительные числа.

Генераторы, основанные на сдвиговом регистре, быстрые. Они обладают большой длиной периода при условии правильного выбора примитивных полиномов. Однако в генераторах этого класса были обнаружены корреляции, которые могут привести к систематическим ошибкам в расчетах Монте-Карло [5,6,7]. Для улучшения статистических свойств были разработаны модификации ГСР. LFSR113 является таким комбинированным генератором.

Пусть имеется J рекуррентных последовательностей (1) и $j-$ая последовательность имеет характеристический полином $P_j(z)$ степени k_j. Пусть он является примитивным, тогда период последовательности равен $p_j =$

$2^{k_j} - 1$. Предположим, что $P_j(z)$ не имеют общих делителей, и p_j являются взаимно простыми. Для каждой последовательности нулей и единиц определим выходную последовательность при помощи формулы $u_{j,n} = \sum_{k=1}^{L} x_{ns_j+k-1} 2^{-k}$. Тогда для комбинированного генератора выходная последовательность определяется, как $u_n = u_{1,n} \oplus \cdots \oplus u_{J,n}$ [4]. Период генератора равен $p = (2^{k_1} - 1) \times \cdots \times (2^{k_J} - 1)$.

Для LFSR113 длина слова $L = 32$, количество сдвиговых регистров $J = 4$,

$$x_{1,n} = x_{1,n-31} \oplus x_{1,n-25}, s_1 = 18,$$
$$x_{2,n} = x_{2,n-29} \oplus x_{2,n-27}, s_2 = 2,$$
$$x_{3,n} = x_{3,n-28} \oplus x_{3,n-15}, s_3 = 7,$$
$$x_{4,n} = x_{4,n-25} \oplus x_{4,n-22}, s_4 = 13.$$

Таким образом, период генератора LFSR113 равен $p = (2^{31} - 1)(2^{29} - 1)(2^{28} - 1)(2^{25} - 1) \approx 10^{34}$. Для генератора доказано точное свойство равнораспределения вероятностей в размерности порядка 30 [4]. Статистические тесты выявляют дополнительные корреляции в выходных последовательностях LFSR113, но найденные корреляции связаны исключительно с тем, что выходные биты данных генераторов имеют линейную структуру по построению, что не является серьезным недостатком данных генераторов [8]. Для генератора LFSR113 был разработан эффективный метод пропуска кусков последовательностей и инициализации параллельных потоков случайных чисел [9,10]. Для генератора ранее были разработаны эффективные реализации с использованием SSE- и AVX2-векторизаций [11,13,14,15], а также реализации для графических процессоров [9,16]. Генератор включен в библиотеки генерации случайных чисел GNU Scientific Library [12], RNGAVXLIB [13,14,15], PRAND [9], clRNG [16], и др.

Реализация алгоритма на языке Си представлена в табл. 1. Перед первым вызовом генератора необходимо задать (u_1, u_2, u_3, u_4) – начальное состояние, состоящее из первых четырех выходных значений. u_1, u_2, u_3, u_4 могут принимать любые значения, большие, чем $1, 7, 15, 127$ соответственно.

4. AVX512-реализация параллельной генерации четырех выходных последовательностей псевдослучайных чисел для алгоритма LFSR113

Для реализации алгоритмов использовался набор инструкций AVX512F. Описание использованных команд можно найти в [3].

В таблицах 1 и 2 представлены реализации алгоритма для генератора LFSR113. В таблице 1 приведен алгоритм на языке ANSI C, в таблице 2 – алгоритм, использующий AVX512-векторизацию. Первая часть AVX512-алгоритма, приведенного в таблице 2, выполняет те же самые действия, что и алгоритм на ANSI C, приведенный в таблице 1, но каждая операция применяется сразу к вектору из шестнадцати чисел. Состояние генератора

задается четырьмя числами, а в регистры ZMM поместятся одновременно 4 таких состояний. Таким образом, AVX512-векторизация позволяет параллельно вычислять четыре выходные последовательности. Вторая выходная последовательность инициализируется при помощи пропуска 2^{108} чисел в первой выходной последовательности, третий поток – 2^{109}, четвертый поток – 2^{110} .

Чтобы вычислить новое состояние генератора, старое записывается в регистр *ZMM1* при помощи команды **vmovaps**. Эта команда работает с данными, выровненными по 64-битным границам, и выполняется гораздо быстрее, чем аналогичная команда **vmovups,** для работы которой не требуется выравнивание. В регистр *ZMM5* помещаются первые 16 чисел массива *lfsr113_const512*. Трех-операндный синтаксис позволяет записывать результат в регистр, отличный от входных операндов. Поэтому следующая команда **vpandd** вычисляет

$z1 \,\&\, 4294967294U, z2 \,\&\, 4294967288U, z3 \,\&\, 4294967280U, z4 \,\&\, 4294967168U$

одновременно для всех четырех потоков и помещает результат в регистр *ZMM2*. Далее команд **vpsllvd** производит сдвиг влево каждого слова регистра *ZMM2* на заданное количество бит. Теперь регистр *ZMM2* содержит

$(z1 \,\&\, 4294967294U) \ll 18,\ (z2 \,\&\, 4294967288U) \ll 2,$

$(z3 \,\&\, 4294967280U) \ll 7,\ (z4 \,\&\, 4294967168U) \ll 13$ для всех четырех потоков. Следующие команды завершают вычисление новых состояний.

Инструкция **vmovaps** записывает их в структуру *state*. Далее необходимо вычислить $z_1 \oplus z_2 \oplus z_3 \oplus z_4,\ z_5 \oplus z_6 \oplus z_7 \oplus z_8,\ z_9 \oplus z_{10} \oplus z_{11} \oplus z_{12},$ $z_{13} \oplus z_{14} \oplus z_{15} \oplus z_{16}$. В регистре *ZMM1* содержатся только что вычисленные состояния $z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}, z_{11}, z_{12}, z_{13}, z_{14}, z_{15}, z_{16}$. Далее команда **vpshufd $78** перемешивает числа в *ZMM1*, и результат сохраняет в *ZMM2:* $z_3, z_4, z_1, z_2, z_7, z_8, z_5, z_6, z_{11}, z_{12}, z_9, z_{10}, z_{15}, z_{16}, z_{13}, z_{14}$. После применения XOR к *ZMM1* и *ZMM2*, регистр *ZMM3* содержит $z_1 \oplus z_3, z_2 \oplus z_4, z_3 \oplus z_1, z_4 \oplus z_2, z_5 \oplus z_7, z_6 \oplus z_8, z_7 \oplus z_5, z_8 \oplus z_6, z_9 \oplus z_{11}, z_{10} \oplus z_{12}, z_{11} \oplus z_9, z_{12} \oplus z_{10}, z_{13} \oplus z_{15}, z_{14} \oplus z_{16}, z_{15} \oplus z_{13}, z_{16} \oplus z_{14}$. После выполнения команды **vpshufd $255** регистр *ZMM2* содержит $z_4 \oplus z_2, z_4 \oplus z_2, z_4 \oplus z_2, z_4 \oplus z_2, z_8 \oplus z_6, z_8 \oplus z_6, z_8 \oplus z_6, z_8 \oplus z_6, z_{12} \oplus z_{10}, z_{12} \oplus z_{10}, z_{12} \oplus z_{10}, z_{12} \oplus z_{10}, z_{16} \oplus z_{14}, z_{16} \oplus z_{14}, z_{16} \oplus z_{14}, z_{16} \oplus z_{14}$. После очередного применения команды XOR, регистр *ZMM3* содержит $z_1 \oplus z_3 \oplus z_4 \oplus z_2, 0, z_3 \oplus z_1 \oplus z_4 \oplus z_2, 0, z_5 \oplus z_7 \oplus z_8 \oplus z_6, 0, z_7 \oplus z_5 \oplus z_8 \oplus z_6, 0, z_9 \oplus z_{11} \oplus z_{12} \oplus z_{10}, 0, z_{11} \oplus z_9 \oplus z_{12} \oplus z_{10}, 0, z_{13} \oplus z_{15} \oplus z_{16} \oplus z_{14}, 0, z_{15} \oplus z_{13} \oplus z_{16} \oplus z_{14}, 0$. Далее при помощи команды **vpermd** выходные числа каждой последовательность помещаются в младшие 128 бит регистра *ZMM3*: $z_1 \oplus z_3 \oplus z_4 \oplus z_2, z_5 \oplus z_7 \oplus z_8 \oplus z_6, z_9 \oplus z_{11} \oplus z_{12} \oplus z_{10}, z_{15} \oplus z_{13} \oplus z_{16} \oplus z_{14}$. Последняя инструкция **vmovaps** записывает их в массив *ans*.

Табл. 1. Реализации алгоритма LFSR113 генерации псевдослучайных чисел на ANSI C
Table. 1. Implementation of algorithm LFSR113 for generation of pseudo-random numbers with ANSI C

```
unsigned u1, u2, u3, u4;
unsigned int lfsr113_ansi_generate_(){
   unsigned b;
   b = (( u1 <<  6) ^ u1) >> 13;
   u1 = (( u1 & 4294967294U) << 18) ^ b;
   b = (( u2 <<  2) ^  u2) >> 27;
   u2 = (( u2 & 4294967288U) <<  2) ^ b;
   b = (( u3 << 13) ^  u3) >> 21;
   u3 = (( u3 & 4294967280U) <<  7) ^ b;
   b = (( u4 <<  3) ^  u4) >> 12;
   u4 = (( u4 & 4294967168U) << 13) ^ b;
   return ( u1 ^  u2 ^  u3 ^  u4);
}
```

Табл. 2. AVX512-реализация параллельной генерации четырех последовательностей псевдослучайных чисел для алгоритма LFSR113
Table 2. AVX512-реализация параллельной генерации четырех последовательностей псевдослучайных чисел для алгоритма LFSR113

```
typedef struct{
  unsigned u[16] __attribute__ ((aligned(64)));
} lfsr113_state;
unsigned lfsr113_consts512[64] __attribute__ ((aligned(64)))
={4294967294U,4294967288U,4294967280U,4294967168U,
4294967294U,4294967288U,4294967280U,4294967168U,
4294967294U,4294967288U,4294967280U,4294967168U,
4294967294U,4294967288U,4294967280U,4294967168U,     6,2,13,3,
6,2,13,3,   6,2,13,3,   6,2,13,3,   13,27,21,12,   13,27,21,12,
13,27,21,12,  13,27,21,12,  18,2,7,13,  18,2,7,13,  18,2,7,13,
18,2,7,13};
unsigned    ind[16]    __attribute__    ((aligned(64)))    =
{0,4,8,12,0,4,8,12,0,4,8,12,0,4,8,12};
void    lfsr113_avx512_generate_four_(lfsr113_state*    state,
unsigned int * ans){
    asm volatile(
    "vmovaps (%0),%%zmm1\n"\
    "vmovaps (%1),%%zmm5\n"\
     "vpandd %%zmm1, %%zmm5, %%zmm2\n"\
     "vpsllvd 192(%1),%%zmm2,%%zmm2\n"\
    "vpsllvd 64(%1),%%zmm1,%%zmm4\n"\
```

```
      "vpxord %%zmm4,%%zmm1,%%zmm4\n"\
      "vpsrlvd 128(%1),%%zmm4,%%zmm4\n"\
      "vpxord %%zmm4,%%zmm2,%%zmm1\n"\
      "vmovaps %%zmm1,(%0)\n"\
      "vpshufd $78,%%zmm1,%%zmm2\n"\
      "vpxord %%zmm2,%%zmm1,%%zmm3\n"\
      "vpshufd $255,%%zmm3,%%zmm2\n"\
      "vpxord %%zmm2,%%zmm3,%%zmm3\n"\
      "vmovaps (%3),%%zmm4\n"\
      "vpermd %%zmm3, %%zmm4,  %%zmm3\n"\
      "vmovaps %%xmm3, (%2)\n"\
      "": :
"r"(state->u),"r"(lfsr113_consts512),"r"(ans), "r"(ind));
   }
```

5. Скорость генерации

Были измерены скорости генерации псевдослучайных чисел для различных реализаций генератора LFSR113. В таблице 3 представлены скорости генерации для ANSI C версии, и для версий, которые используют наборы команд SSE4.1, AVX2 и AVX512F. Измерения были проведены на Intel Xeon Platinum 8162 (2 GHz). Был использован компилятор GCC версии 4.8.5.

Для данного генератора исходная ANSI C версия весьма эффективна за счет использования только быстрых однотактовых логических и сдвиговых операций. Более того, при компиляции ANSI C версии с ключом оптимизации –O3, компилятор автоматически оптимизирует алгоритм, используя при этом AVX инструкции, поэтому скорость генерации для такой версии превосходит скорость SSE-реализации.

Табл. 3. Производительность реализаций генератора LFSR113
Table. 3. Performance of implementations for LFSR113 generator

	ANSI C Gbit/s	SSE4.1 Gbit/s	AVX2 Gbit/s	AVX512 Gbit/s	AVX512/ ANSI C	AVX512 / SSE	AVX512/ AVX2
-O0	4.17	4.64	15.72	31.92	7.65	6.88	2.03
-O3	14.24	4.67	20.11	39.37	2.76	8.43	1.96

Статья подготовлена в рамках государственного задания ФАНО России № 0033-2014-0010.

Список литературы

[1]. Landau D.P., Binder K., A Guide to Monte Carlo Simulations in Statistical Physics, 4th edition, Cambridge University Press, Cambridge, 2015.

[2]. Бараш Л. Ю., Щур Л.Н. Генерация случайных чисел и параллельных потоков случайных чисел для расчетов Монте-Карло, Моделирование и анализ информационных систем, 19(2), 2012, стр. 145–162.

[3]. Intel® 64 and IA-32 architectures software developer's manual combined volumes 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4, http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

[4]. L'Ecuyer P., Tables of maximally equidistributed combined LFSR generators, Mathematics of Computation 68(225), 1999, pp. 261–269.

[5]. A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, Monte Carlo simulations: Hidden errors from ''good'' random number generators, Phys. Rev. Lett. 69, 1992, pp. 3382–3384.

[6]. P. Grassberger, On correlations in "good" random number generators, Phys. Lett. A 181, 1993, pp. 43–46.

[7]. F. Schmid and N. B. Wilding, Errors in Monte Carlo simulations using shift register random number generators, Int. J. Mod. Phys. C 6, 1995, pp. 781–787.

[8]. L'Ecuyer P., Simard R. TestU01: A C library for empirical testing of random number generators, ACM Transactions on Mathematical Software 33(4), 2007, article 22.

[9]. L.Yu.Barash, L.N.Shchur, PRAND: GPU accelerated parallel random number generation library: Using most reliable algorithms and applying parallelism of modern GPUs and CPUs, Computer Physics Communications, 185(4), 2014, pp. 1343-1353.

[10]. Л.Ю. Бараш, Л.Н. Щур, О генерации параллельных потоков псевдослучайных чисел, Программная инженерия, №1, 2013, стр. 24-32.

[11]. Бараш Л. Ю., Гуськова М. С., Щур Л. Н. Использование AVX-векторизации для увеличения производительности генерации случайных чисел, Программирование, 43(3), 2017, стр. 22–40.

[12]. M. Galassi et al, GNU Scientific Library Reference Manual - Third Edition, Network Theory Ltd., 2009, ISBN: 0954612078.

[13]. M.S. Guskova, L.Yu. Barash, L.N. Shchur, RNGAVXLIB: Program library for random number generation, AVX realization, Computer Physics Communications, 200, 2016, pp. 402–405.

[14]. Barash L. Y., Shchur L. N. RNGSSELIB: Program library for random number generation. More generators, parallel streams of random numbers and Fortran compatibility, Computer Physics Communications 184(10), 2013, pp. 2367–2369.

[15]. L.Yu. Barash, L.N. Shchur, RNGSSELIB: Program library for random number generation, SSE2 realization, Comput. Phys. Commun., 182 (7), 2011, pp. 1518-1527.

[16]. P. L'Ecuyer, D. Munger, N. Kemerchou, clRNG: A library for uniform random number generation in OpenCL, 2015, http://www-labs.iro.umontreal.ca/~simul/clrng/

Applying AVX512 vectorization to improve the performance of a random number generator

[1,2] *M.S. Guskova <maria.guskova@rambler.ru>*
[1,2,3] *L.Yu. Barash <barash@itp.ac.ru>*
[1,2,3,4] *L.N. Shchur <shchur@chg.ru>*
[1] *Science Center in Chernogolovka, 142432 Chernogolovka, Russia*
[2] *National Research University Higher School of Economics, 101000 Moscow, Russia*
[3] *Landau Institute for Theoretical Physics, 142432 Chernogolovka, Russia*
[4] *Dorodnicyn Computing Centre, FRC CSC RAS, 119333 Moscow, Russia*

Abstract. The generation of uniformly distributed random numbers is necessary for computer simulation by Monte Carlo methods and molecular dynamics [1]. Generators of pseudo-random numbers (GPRS) are used to generate random numbers. GPRS uses deterministic algorithms to calculate numbers, but the sequence obtained in this way has the properties of a random sequence. For a number of problems using Monte Carlo methods, random number generation takes up a significant amount of computational time, and increasing the generation capacity is an important task. This paper describes applying SIMD instructions (Single Instruction Multiple Data) to parallelize generation of pseudorandom numbers. We review SIMD instruction set extensions such as MMX, SSE, AVX2, AVX512. The example of AVX512 implementation is given for the LFSR113 pseudorandom number generator. Performance is compared for different algorithm implementations.

Keywords: Pseudo random numbers; SIMD; AVX512 technology

DOI: 10.15514/ISPRAS-2018-30(1)-8

For citation: Guskova M.S., Barash L.Yu., Shchur L.N. Applying AVX512 vectorization to improve the performance of a random number generator. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 115-126. DOI: 10.15514/ISPRAS-2018-30(1)-8

References

[1]. Landau D.P., Binder K., A Guide to Monte Carlo Simulations in Statistical Physics, 4th edition, Cambridge University Press, Cambridge, 2015.
[2]. Barash L.Y., Shchur L.N., Generation of Random Numbers and Parallel Random Number Streams for Monte Carlo Simulations. Modeling and Analysis of Information Systems, 19(2), 2012, pp. 145-162 (in Russian).
[3]. Intel® 64 and IA-32 architectures software developer's manual combined volumes 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4, http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html
[4]. L'Ecuyer P., Tables of maximally equidistributed combined LFSR generators, Mathematics of Computation 68(225), 1999, pp. 261–269.

[5]. A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, Monte Carlo simulations: Hidden errors from ‘‘good’’ random number generators, Phys. Rev. Lett. 69, 1992, pp. 3382–3384.

[6]. P. Grassberger, On correlations in “good” random number generators, Phys. Lett. A 181, 1993, pp. 43–46.

[7]. F. Schmid and N. B. Wilding, Errors in Monte Carlo simulations using shift register random number generators, Int. J. Mod. Phys. C 6, 1995, pp. 781–787.

[8]. L'Ecuyer P., Simard R. TestU01: A C library for empirical testing of random number generators, ACM Transactions on Mathematical Software 33(4), 2007, article 22.

[9]. L.Yu.Barash, L.N.Shchur, PRAND: GPU accelerated parallel random number generation library: Using most reliable algorithms and applying parallelism of modern GPUs and CPUs, Computer Physics Communications, 185(4), 2014, pp. 1343-1353.

[10]. L.Yu. Barash, L.N. Shchur, On the generation of parallel streams of pseudorandom numbers, Software Engineering Vol.1, 2013, pp. 24–32 (in Russian).

[11]. L.Yu. Barash, M.S. Guskova, L.N. Shchur, Employing AVX vectorization to improve the performance of random number generators, Programming and Computer Software, 43(3), 2017, pp. 145-160.

[12]. M. Galassi et al, GNU Scientific Library Reference Manual - Third Edition, Network Theory Ltd., 2009, ISBN: 0954612078.

[13]. M.S. Guskova, L.Yu. Barash, L.N. Shchur, RNGAVXLIB: Program library for random number generation, AVX realization, Computer Physics Communications, 200, 2016, pp. 402–405.

[14]. Barash L. Y., Shchur L. N. RNGSSELIB: Program library for random number generation. More generators, parallel streams of random numbers and Fortran compatibility, Computer Physics Communications 184(10), 2013, pp. 2367–2369.

[15]. L.Yu. Barash, L.N. Shchur, RNGSSELIB: Program library for random number generation, SSE2 realization, Comput. Phys. Commun., 182 (7), 2011, pp. 1518-1527.

[16]. P. L’Ecuyer, D. Munger, N. Kemerchou, clRNG: A library for uniform random number generation in OpenCL, 2015, http://www-labs.iro.umontreal.ca/~simul/clrng/

Dealing with not Fully Described Objects in Decision Support Systems: Alternative Approaches

[1] *Valery N. Yudin <yudin@ispras.ru>*
1,2 Leonid E. Karpov <mak@ispras.ru>
1 Ivannikov Institute for System Programming of the Russian Academy of Sciences
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
2 Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia

Abstract. Not fully described objects may be seen in many different areas and applications – from medicine and up to space apparatus control. Starting to de-sign and develop a decision support system, which should work with not fully described objects, we may choose between alternatives. One of two approaches compared in this article is based on logical deduction according to a priory specified rules. Here the "IF-THEN" productions are intensively used. Another approach, which is often called case-based, assumes the presence of case base filled by real and/or artificial (model) cases. This second approach does not insist on rules and object models, but it is much closer to the mental model used when humans are thinking.

Keywords: decision support system; rule-based approach; case-based reasoning; case base; classes of equivalence; objects; features; measure of closeness.

DOI: 10.15514/ISPRAS-2018-30(1)-9

For citation: Yudin V.N., Karpov L.E. Dealing with not Fully Described Objects in Decision Support Systems: Alternative Approaches. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 127-136. DOI: 10.15514/ISPRAS-2018-30(1)-9

1. Introduction

There are many practically used decision support programming systems (DSS) nowadays (see examples in [1-3]). In these systems the most difficult for analysis and decision making are the situations when the interacted objects have informal characteristics, that means that it is difficult to discover main factors influencing the objects and relations actual for those objects. Establishing the exact object behavior model is often not possible because of lack of knowledge about the objects themselves and about the environment in which they are functioning and interacting. Nevertheless, operations with such objects are often even more

important than operations with well-formalized objects. Some methods for operating with not fully-described objects are not enough productive.

2. Rules and cases

There exists a very important example of such an object – it is a human organ-ism. The application area for which human organism is of principle interest is medicine in general and medicinal diagnostics in particular. Programming DSS used for physician's decision support for diagnostics and cure choosing are often based on mathematical methods. It is easy to see two alternatives in dealing with the objects mentioned above: rule-based and case-based deduction.

In production rule approach, knowledge is represented with the set of several rules formulated in the "IF ... – THEN ..." form. These rules are used to de-duct the conclusion from input data. Such a method of reasoning is an example of a direct logical deduction.

Decision trees are often used in medicine. They represent a particular case of decision trees when the conclusion starts with checking some sign assigned to the tree root, and continues by moving along the tree to its leaves where the different decisions are located. Decision trees with single entry named tree root are often treated as a particular case of deduction rules.

Rule-based deduction makes it possible to incorporate knowledge into the system with the help of descriptive logic. Rules are following each other in a definite order, which is helpful for understanding them, but in fact does not create any order relation.

Rule-based models are much less structured and reflect the order in general action flow to smaller extent than many other approaches. They are really helpful in situations with relatively small set of limits applied to active actions, and where, consequently, a small amount of rules may define a very comprehensive scheme of interactions of component parts of integrated system.

Conditions in rule notation are in fact rule premises. They consist of one or several pairs "attribute – value" with logical "AND", "OR", "NOT" connectives. The conclusion denotes some fact or some instruction for definite action, which should be fulfilled according to the rule. Logical deduction mechanism is looking for those rules that include the facts entered, and then actualize the appropriate rules. The rule becomes active if the fact entered corresponds with the rule condition. In such a case, the active rule conclusion becomes a fact too. When all the activated rules respond, the final conclusion may be proved or rejected.

In practice, one can see a reverse logical deduction where the reasoning starts from the assumption made for possible final conclusion and moves towards the facts that may confirm the hypothesis.

Case-based reasoning (see [4-5]) is focused on knowledge about previous situations, or cases, stored in "case base". Decision made in conditions that are treated as similar to some situation met earlier, after being adapted is applied to the current

case. This method in conjunction with differential set (a set of potentially suitable decisions) is optimal for not-fully described case if there is a lack of time and resources. The case itself, if it was announced as similar, is the basis of the decision.

Decision making modeling human reasoning is practically used in many different areas of human activity. There exist a broad spectrum of possible applications, and control of poorly formalized objects is among them. We may treat the human organism as an object under control. It cannot be described with relatively simple mathematical model; thus, case-based reasoning methodology for physician's DSS may be considered as having very good prospects. Case-based reasoning is a method of looking for similar problem situations, which takes into account previous experience of problem solving. Instead of searching the solution from the very beginning, an attempt is done to use the solution found earlier in similar situation, and then adapting it to the changing situation of the current case. After this current case is processed, it (and its solution) is added to the case base, from where it can be used later.

Each case may consist of the situation description with the problem to be solved included, and the list of actions that were used to solve the appropriate problem. The solution may have the form of the previous case or of suggested typical example of problem solving method.

Accumulated collection of cases, which can be replenished with modeled or met in practice cases, forms the so called "case base". System built on this principles is in fact self-learning: the more cases are stored in the case base, the wider are the limits of their possible values, the higher is the probability to find "the most suitable" case, and therefore, the higher is the quality of the final solution.

3. Metrics and measures

Most part of existing approaches to building case-based reasoning DSS is focusing on only one aspect: choosing the most suitable cases. The basis of all the approaches to case selection is the method of estimation of similarity of previous cases and the current case. In these systems the metrics is defined in feature space. The point corresponding to the current case is then defined in this space, and, according to the chosen metrics, the closest case point for the current case is defined. Depending on case feature types, different metrics may be chosen: Chebyshev, Euclidean, Hamming, Mahalanobis, Manhattan, Akritean, Minkowski, Zhuravlyov (see [6-8]) distances and many others. Still there are situations when no metrics can be introduced. In these cases, measure of closeness is used instead of metrics. In its turn, measure of closeness may be defined by different ways, for example, in the form of case selection rule.

Structuring of a case set is also very useful. Different methods, in particular Data Mining approach, make it possible to clarify hidden knowledge about the application area. Classes of case equivalence may be established basing on various techniques: with the help of expert knowledge, or using the learning samples, or by clustering the case base. Breaking the case base into equivalence classes is the way

of speeding-up case searching process: cases belonging to some class are announced similar by the definition. Unfortunately, this measure of closeness is not absolutely adequate for estimating interrelations between current case and previous cases, especially when this current case falls into equivalence classes' intersection.

Image recognition problems often assume that object descriptions are founded on the set of features and signs, and this set is common for objects of all the classes. In other words, classes of equivalence and investigated objects may be located in unified feature space and have the dimensions. Industrial applications often break this condition. Not only real objects, but also classes' descriptions may have their own unique feature space. As an example, in medicine each disease (every disease may be treated as a separate class of equivalence) may be characterized with its own set of significant features. In addition, a case under investigation may have feature set that is absolutely different with those feature sets that were initially entered into the system. Relations between current case and classes of equivalence may be retrieved with the help of classes' projections on object feature set. Not fully described case may fall into the projection, to which it does not belong only because of lack of the feature that may prevent it from ambiguity.

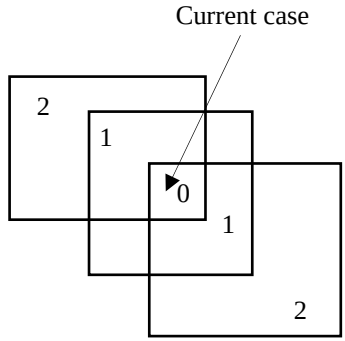


Fig. 1. Estimating the measure of closeness between the current case and previous cases

Estimating of the current case assumes the comparison of the corresponding point of feature space with spatial location of classes' projections ([9]). Analogues are the cases that belong to the class, to which current case belongs, these cases are considered as the most closest to that current case. If current case falls into classes' intersection, then the analogues belonging to the same intersection will be treated as close ones. Depending on the complexity of intersection, we may divide all analogues into separate groups (fig. 1). Analogues from the inner part of intersection are naturally considered as more close to the current case than analogues that are belonging only to one class. And, of course, analogues of the highest rank are located in the intersection of all the classes from the differential set.

Now the definition of offered measure of closeness can be finally formulated. *It is the distance between the current case and previous case evaluated as the difference of the number of classes that include the current case, and the number of classes that include the previous case.*

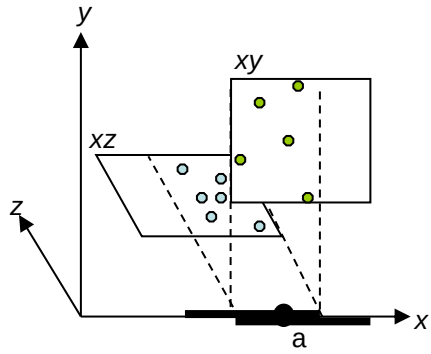


Fig 2. Using analogues for defining missing features of the current case

Initial case selection may not bring any tangible result. For example, the presence of a single "high temperature" feature will give a plenty of analogues. The physician should either agree that this poor set of features will not help to solve the problem or should expand feature set.

Analogues themselves contain the information about the features that should be retrieved. On fig. 2 one-dimensional current case $x=a$ falls into projection of classes *xy* and *xz*. To be compared with analogues from *xy* class, it lacks the *y* feature, and for *xz* class – it needs *z* feature to be added.

Thus, case selection cycle may be split into stages:

1. searching analogues for current case;
2. estimating the validity of the selected set (done "manually"). If "yes", the selection succeeds. If "no", execute the next stage;
3. building a ranked list of additional features in order to differentiate classes (these features may be found in previous cases);
4. making an attempt to retrieve additional features (done "manually"). Some features can never be exposed. If this is the case, the cycle stops with negative result.

4. Classes and features

The method described here was implemented in the programming DSS designed and developed in the Institute for System Programming of Russian academy of

sciences (ISP RAS) in cooperation with specialists from Moscow Regional Research and Clinical Institute ("MONIKI") (see [9-13]).

Physician is working in terms of "*disease*", "*treatment strategy*", "*patient*", "*sign*". The system provides access to the case base and automatically converts these terms into corresponding system notions "*class*", "*object*", "*feature*".

In some countries, there exists hands-on experience to carry out minimal but absolutely needed set of research actions before the diagnosis is stated. In our country physicians often set the diagnosis in much more comprehensive conditions, when not all the signs are known. The research actions are usually directed from very simple to more complex and expensive, starting from complaints and external examination and moving towards laboratory and instrumental research. From one point of view, if the diagnosis is rather clear from very first stages, there is no need to fulfill expensive analysis. From an-other side, the final solution may be delayed after making additional investigations, or wait till new symptoms appear. If feature set allows choice among several diseases, all of them are included into differential set.

Before starting the estimation process, physician needs not only to estimate the sufficiency of feature set, but also select those features that are linked with the differential set. In order not to overload the estimation process by extra features, the system automatically selects the needed signs, and if the set appeared is not full, will show still absent features. The final solution either to estimate partially filled set, or to continue with additional research, should be made by physician.

When there is a lack of time or resources, it may be impossible to expose all the absent features. Some new notions should be introduced, for instance, "*persistent feature combination*" and "*feature rank within its class according to the degree of its informativeness*". To define the priority of feature retrieving, additional selection criteria of new features searching should be defined according to frequency of feature used in the application, to object category, to feature availability.

Obviously, not all the features that are mentioned in the class description have the same informativeness. For example, in medicine there exist (pathognomonic) symptoms, that have absolute diagnostic value (markers of cancer, infarction, different types of hepatitis) and make it possible to determine the particular disease. Not only in medicine, but in much more general case, ideal features have the highest level of informativeness. They identify their classes unambiguously and can never be met in other classes. However, even in medicine ideal features sometimes can't be discovered while diagnosing the appropriate disease or its particular stage or form, or while a particular. If no ideal feature is found, some other features that are typical for the hypothesized disease should be investigated. There are features that appear in some classes with a much greater probability than that of their occurrence in other classes. These features are called controlling or causal features (bilirubin, hepatic enzymes in hepatitis). To get the final solution, we must not be focusing on one such feature, it should be considered in a combination with some other features. There also exist the third type of features – attendant features. They do not

characterize classes (in medicine, for example, some symptoms may accompany the disease: high temperature or erythrocyte sedimentation rate, and so on). The presence of these features may be treated as a necessary but not sufficient condition of belonging to the class. Their role in differentiating class is negligible. In summary, it may be stated that the features of the classes in the case base are ranked from the ideal to causal and then to attendant. Only ideal features may reliably identify the state of the object. If there is an ideal feature for some class, even if there are a number of attendant features, the fact of belonging of an object to a certain class can be stated only with some probability.

5. Advantages and drawbacks

In order to exactly define the current case class, it's always possible to operate with one single sign. Possible relations with other features should be taken into account. For example, a consistently observed combination of symptoms, defined in medicine as a syndrome, has a special diagnostic value. It is very close to Data Mining method known as association analysis. This method is very useful and often successful in processing of classes' descriptions in the case base. It helps to retrieve consistent feature combinations and ranks. Despite the fact that medicine is a precedent science, a lot of case examples in literature are built in terms of decision trees. Still, the lack of controlling (determining) feature may block the entry point into the production rule-based mechanism (for example, into the decision tree or into one of its nodes).

The DSS for transplantology designed and developed in ISP RAS and MONIKI is the concretization of general conceptual case-based complex objects control system. Usually, information support for physicians in similar DSS is limited to the "waiting list" card catalogues. Such kind of DSS are intended to provide initial selection of "donor-recipient" pair. All the problems of pair survival are beyond the competence of many practically used DSS. This point significantly differs them from the designed system. In fact, the main problem is in choosing of right tactics of patient support that should be followed after surgery transplantation operation. Many input parameters and contradictory factors should be taken into account. The role of DSS is dramatically increasing. Only automated systems may help the physician to solve conflicting problems. Human organism as an object under control can't be described with relatively simple mathematical model, that's why case-based reasoning methods for DSS may be considered as very promising.

6. Conclusion

Production rule-based model provides ease of perception and modification, simple mechanism of deduction, but has a number of disadvantages: dissimilarity with human mental structures of knowledge representation, uncertainty in rule interrelations and others.

The main advantage of case-based reasoning is simplicity and ease of implementation, while the drawback is in its inability to create models and rules that can generalize previous experience. One of the main problems of this method is the difficulties with correct selection of appropriate cases, which rests on the assessment of the similarity of the precedent and the current case. However, in certain circumstances, particularly when there is the need to work with not fully described objects, case-based reasoning method has notable advantage over other approaches.

Acknowledgements

Authors are really thankful to Russian Foundation for Basic Research that is supporting their work in 18-07-01211_a project.

References

[1]. Hillary Don. Decision making in critical care. University of California School of Medicine, San Francisco, California, B.C. Decker Inc., The C.V. Mosby company, 1995.
[2]. Chike. F. Oduoza. Decision support system based on effective knowledge management framework to process customer order enquiry. Decision Support Systems, Editor Chiang S. Jao, pp. 406, January 2010, INTECH, Croatia.
[3]. Prabhu Murugesan, Senthil Prabhu Natarajan, Lakshmi Karthigeyan. Clinical decision support systems. Computer Sciences Corporation, 2014.
[4]. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications, 7(1):39-59, 1994.
[5]. Klaus-Dieter Althof, Eric Auriol, Ralph Barlette, and Michel Manago. A Review of Indus-trial Case-Based Reasoning Tools. AI Intelligence, 1995.
[6]. Greg Grudic and Jane Mulligan. Outdoor Path Labeling Using Polynomial Mahalanobis Distance, Robotics: Science and Systems II, 2006.
[7]. Elena Deza and Michel Marie Deza. Encyclopedia of Distances. Springer. 2009.
[8]. Zhuravlyov Yu. I. Selected scientific works, Magistr [Master], Moscow, 1998 (in Russian).
[9]. Yudin V.N. Measure of closeness in Case-based Reasoning System. Dokladyi 12-y Vserossiyskoy konferentsii Matematicheskie Metodyi Raspoznavaniya Obrazov (MMRO-12) [Proc. 12-th All-Russia Conf. on Mathematical Methods in Pattern Recognition], MAKS Press, Moscow, 2005, pp. 241-244 (in Russian).
[10]. Karpov L.E., Yudin V.N.. Integration of data Mining and Case-Based Reasoning methods in medical diagnostics and treatment choosing, Sbornik dokladov 13-j Vserossijskoj konferentsii Matematicheskie metody raspoznavaniya obrazov [Proc. of 13-th All-Russian conference Math. methods of pattern recognition], October 2007, MAKS Press, 2007, pp. 589-591, ISBN 978-5-317-02060-6, http://www.mmro.ru/files/mmro13.pdf (in Russian).
[11]. Yudin V.N., Karpov L.E., Vatazin A. V. Data mining and case-based reasoning methods in physician's decision making support software system, Almanah klinicheskoy meditsinyi [Almanac of Clinical Medicine], Moscow, 2008, v. 17, part 1, pp. 266-269, ISSN 2072-0505 (in Russian).

[12]. Yudin V.N., Karpov L.E., Vatazin A.V. Cure process as an adaptive control of human organism in software system “Physician’s Partner”. Almanah klinicheskoy meditsinyi [Almanac of Clinical Medicine], Moscow, 2008, v. 17, part 1, pp. 262-265, ISSN 2072-0505 (in Russian).

[13]. Valery Yudin, Leonid Karpov. The Case-Based Software System for Physician’s Decision Support. Lecture Notes in Computer Science Sublibrary: SL 3, Springer Verlag, Berlin, Heidelberg, 2010, pp. 78-85

Работа с неполностью описанными объектами в системах поддержки принятия решений: альтернативные подходы

[1] *В.Н. Юдин <yudin@ispras.ru>*

[1,2] *Л.Е. Карпов <mak@ispras.ru>*

[1] *Институт системного программирования им. В. П. Иванникова РАН*
Москва, 109004, ул. Солженицына, д. 25

[2] *Московский Государственный университет имени М. В. Ломоносова*
Москва, 119991, ГСП-1, Ленинские горы, д. 1

Аннотация. Неполностью описанные объекты могут встретиться в самых разных предметных областях и приложениях - от медицины до управления космическими кораблями. Начиная работу по проектированию и разработке системы поддержки принятия решений, которая должна работать с неполностью описанными объектами, необходимо предварительно выбрать один из альтернативных подходов. В основе одного из двух подходов, рассматриваемых в этой статье, находится логический вывод по заранее зафиксированным правилам. В таком подходе интенсивно используются продукции вида "ЕСЛИ-ТО". Другой подход, который часто называется прецедентным, предполагает наличие базы прецедентов, наполненной реальными и/или искусственными (модельными) случаями. Для этого второго подхода правила и модели объектов не являются необходимыми, но он значительно ближе к модели принятия решений, которая используется человеком в процессе мышления.

Ключевые слова: неполностью описанные объекты; поддержка принятия решений; логический вывод на основе порождающих правил; деревья решений; вывод на основе прецедентов; база прецедентов; метрика случаев; мера близости случаев; дифференциальный ряд случая; информативность признаков случая.

DOI: 10.15514/ISPRAS-2018-30(1)-9

Для цитирования: Юдин В.Н., Карпов Л.Е. Работа с неполностью описанными объектами в системах поддержки принятия решений: альтернативные подходы. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 127-136. 10.15514/ISPRAS-2018-30(1)-9

Список литературы

[1] Hillary Don. Decision making in critical care. University of California School of Medicine San Francisco, California, B. C. Decker Inc., The C. V. Mosby company, 1995.

[2] Chike. F. Oduoza. Decision support system based on effective knowledge management framework to process customer order enquiry. Decision Support Systems, Editor Chiang S. Jao, ISBN 978-953-7619-64-0, pp. 406, January 2010, INTECH, Croatia.

[3] Prabhu Murugesan, Senthil Prabhu Natarajan, Lakshmi Karthigeyan. Clinical decision support systems. Computer Sciences Corporation, 2014.

[4] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodologi-cal variations, and system approaches. AI Communications, 7(1):39-59, 1994.

[5] Klaus-Dieter Althof, Eric Auriol, Ralph Barlette, and Michel Manago. A Review of Indus-trial Case-Based Reasoning Tools. AI Intelligence, 1995.

[6] Greg Grudic and Jane Mulligan. Outdoor Path Labeling Using Polynomial Mahalanobis Distance, Robotics: Science and Systems II, 2006.

[7] Elena Deza and Michel Marie Deza. Encyclopedia of Distances. Springer. 2009.

[8] Журавлев Ю. И. Избранные труды. М., Магистр, 1998.

[9] Юдин В. Н., Мера близости в системе вывода на основе прецедентов. Математические методы распознавания образов. Сборник докладов 12-й Всероссийской конференции, стр. 241-244, М., МАКС ПРЕСС, 2005

[10] Карпов Л. Е., Юдин В. Н., Интеграция методов добычи данных и вывода по прецедентам в медицинской диагностике и выборе лечения. Математические методы распознавания образов. Сборник докладов 13-й Всероссийской конференции, стр. 589-591

[11] Юдин В. Н., Карпов Л. Е., Ватазин А. В. Методы интеллектуального анализа данных и вывода по прецедентам в программной системе поддержки врачебных решений. Альманах клинической медицины, МОНИКИ, Москва, 2008 г., стр. 266-269

[12] Юдин В. Н., Карпов Л. Е., Ватазин А. В. Процесс лечения как адаптивное управление человеческим организмом в программной системе "Спутник врача". Альманах клинической медицины, т. XVII в двух частях, ч. 1. МОНИКИ, Москва, 2008 г., стр. 262-265

[13] Valery Yudin, Leonid Karpov. "The Case-Based Software System for Physician's Decision Support". Lecture Notes in Computer Science Sublibrary: SL 3, Springer Verlag, Berlin, Heidelberg, 2010, pp. 78-85

Array Database Internals

V.A. Pavlov <vlad.pavlov24@gmail.com>
B.A. Novikov < b.novikov@spbu.ru >
Saint Petersburg State University,
13B Universitetskaya Emb., St Petersburg, Russia, 199034

Abstract**.** After huge amount of big scientific data, which needed to be stored and processed, has emerged, the problem of large multidimensional arrays support gained close attention in the database world. Devising special database engines with support of array data model became an issue. Development of a well-organized database management system which stands on completely uncommon data model required performing the following tasks: formally defining a data model, building a formal algebra operating on objects from the data model, devising optimization rules on logical level and then on the physical one. Those tasks has already been completed by creators of different array databases. In this paper array formalization, core algebra and optimization techniques are revised using examples of AML, RasDaMan, SciDB – developed array database management systems with different algebras and optimization approaches.

Ключевые слова: array databases; overview; formal array algebra; array query processing; array query optimization; AML; RasDaMan; SciDB

DOI: 10.15514/ISPRAS-2018-30(1)-10

For citation: Pavlov V.A., Novikov B.A. Array Database Internals. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 137-160. DOI: 10.15514/ISPRAS-2018-30(1)-10

1. Introduction

Recently in many scientific fields database users need to support and process new non-traditional data structures. Among such uncommon structures are different hierarchical structures, graphs, as well as *arrays*. It's worth noting, that such a need is not explained by the subjective preferences of database users, it is fully justified by the real state of things for users and their requirements for processing the data under study. In this paper we will partially consider what is offered to users who need to store and process array data and how storage and processing are made efficient. But first, we let us understand more precisely, with what kind of data such users have to deal with.

The data referred to is also called *multidimensional discrete data* (MDD) or *raster data* [1]. Such data is homogeneous, each element has some index (represented by a vector in a d-dimensional Euclidean space) and, hence, has some adjacent

elements. This data is typically huge. On a more intuitive level, MDD data can be imagined as huge multidimensional cubes. For such a cube, each cell has a discrete multidimensional index and contains a value of a fixed type. An example of a 3D cube may be the following: a series of images[2] obtained from two Huble telescopes cameras for some a period of time, say. a year. As it has been said, in real life those cubes are usually of tera- or even petabyte scale: for instance, Large Hadron Collider (LHC), producing raster cubes during its work, after a day of functioning and filtering produced data generates multidimensional data sizing over 5 terabytes [3].

Granted, such cubes are not just stored as they often demand some kind of analysis. Usually the analysis to be done is not trivial due to the fact that the need for intelligent raster data processing arises in such fields as: natural sciences, medicine, census, multimedia and OLAP. Demand for efficient storage and processing of huge raster data cubes states a problem of devising special tools and algorithms. The specificity of the data in use is another factor increasing the need in a specialized storage systems. Raster data has several peculiarities induced by its properties mentioned above. These peculiarities include: large size of a single raster data value (a single cube may occupy several disk pages instead of a part of a disk page in case of conventional data types, e.g. numeric values); lack of index support due to absence of natural ordering of cubes, etc. Those peculiarities make efficient processing of raster data different from processing conventional data types in terms of storage and optimization techniques.

Fortunately, the problem of optimized storage and efficient processing of raster data has already been faced by authors of special extensions for existing relational/object-relational databases (such as Terralib [4], PostGIS [5], SpatialLite [6], Oracle GeoRaster [7]) and creators of *array databases* standing on specially devised array data models [8]. Today there are several array database management systems (*ADBMS*) such as RasDaMan [9], SciDB [10], which are still maintained and intensively developed with the aim to continuously improve and to conform to rapidly increasing scientific demands. In each ADBMS much attention is paid to optimization as optimizing queries is crucial when processing queries operating with petabyte sized data cubes.

There are two ways of optimization: logical and physical ones. Logical optimization is usually based on formal algebra standing behind the array model. Physical optimization is typically achieved by devising special storage scheme and/or data retrieving order[11], [12]. In the current work we will briefly review theoretical basement and optimization techniques considering three distinct developed ADBMSs : RasDaMan, AML and SciDB.

The reader should be aware of the fact that the aim of this paper is to present and analyze different data models, algebras and optimization techniques used in some array databases and certainly **NOT** to compare those databases in order to determine advantages of one over another. The paper does **NOT** intend to characterize the databases anyhow so that the given characteristics are based on subjective opinions.

2. Diving In

To study the the theoretical basement along with optimization techniques in ADBMSs it is important to understand a generic algorithm following which a new ADBMS can be built.

First, an array term should be formally defined as it is a central object of interest in such systems. Obviously, in array DBMS algebras, the main object of all operations is an *array*. For best of our knowledge, all the existing algebras define an array mostly similarly. Formally, an array is a function defined on index domain D to some value set V. Value sets differ in different systems. Index domain D is represented as Cartesian product of finite amount of ordered sets $I_1, I_2, ..., I_k$. The value k is called an dimensionality ($valence$) of the array. Applying A function to a vector $(i_1, i_2, ..., i_k)$ is associated with getting the array's cell value.

Second, a formal algebra is introduced. Algebras are mathematical structures where several operations with some core objects are defined. Operations with those objects return an object from the same algebra. One of the most important features of algebras is an ability to construct expressions in them by combining application of algebra's operations. As algebras operations are closed, result of an expression evaluation is again an object from the algebra. In simpler words, a formal algebra enables to construct complex expressions value of which do not leave the algebra. In reality in different systems underlying algebras start to differ. Several existing algebras in existing array dbms are described further.

Third, logical optimization rules are introduced. Complex expressions can be overburdened with unnecessary operations and elimination them simplifies the expression benefiting in less execution complexity.

Fourth, physical optimization rules are derived. Logical optimization of an expression is not sufficient for actually executing the query in the most efficient way. In most cases a single query can be executed differently accounting "physical" information which tells how the queried data is actually stored.

Fifth, the query language is introduced to give a user of the system of the system a convenient high-level language taking the user away from lower-level algebra language.

In the current paper we will look at how the first four steps were followed for each of the highlighted array databases, ignoring high level query languages as they do not contribute much to understanding the theoretical essentials of ADBMS.

2.1 Baumann's array algebra

We will now optimization process is organized in RasDaMan ADBMS, explaining its core model and formal algebra - Baumann's array algebra [13]. The overview of optimization techniques is mostly based on PhD thesis [14] of Ronald Ritsch. To present the entire data model the following terms should be explained:

- Multidimensional intervals and spatial domains
- MDD types, values and elementary operations on them
- Derived operations on MDD data
- Extended relational model with MDD support

2.1.1 Multidimensional Intervals and Spatial Domains

An m-interval X is defined as follows: let $D = Z^n$, then $X = [l_1 : h_1,...,l_n : h_n]$ is $\{(x_1, x_2,..., x_n) \in D \mid (l_i \le x_i \le h_i \forall i \in 1..n\}$. Multiple probe functions are defined on the multidimensional intervals:

- Domain function : $dom(X) = D$
- Dimension function : $dim(X) = n$
- Lower bound function : $lo(X) = (l_1,...,l_n)$, where $lo_i(X)$ denotes l_i
- Upper bound function : $hi(X) = (h_1,...,h_n)$, where $hi_i(X)$ denotes h_i
- Extent function : $card(X) = \Pi_{i=1}^{n}(hi_i(X) - lo_i(X) + 1)$

Usually a multidimensional interval represents an index set of a multidimensional array, therefore it is commonly called a *spatial domain*. It is convenient to restrict the possible value type of an interval by introducing a *spatial domain type*. More precisely, a spatial domain type σ^d is a set of admissible d-dimensional domains $\delta^d \subseteq 2^{Z^d}$. Union of all δ^d over all non-negative integers d will be denoted as δ. Then, it is possible to define multiple operations from δ to δ:

- Trimming along dimension i with one dimensional m-interval B
  $trim(X, i, B) = \{(x_1, x_2,..., x_n) \in X \mid lo(B) \le x_i \le hi(B)$
- Slicing along dimension i at point b
  $slice(X, i, b) = \{(x_1,..., x_{i-1}, x_{i+1},...x_n) \mid (x_1,..., x_{i-1}, b, x_{i+1},...x_n) \in X\}$

2.1.2 MDD values and types

An MDD value a over base type T and spatial domain D with $T \in \tau$ and $D \in \sigma$ is a set of (coordinate, value)-tuples defined by a mapping function a: $D \to T$. Then an MDD value a would be $\{(x, a(x)) \mid a(x) \in T, x \in D\}$. Defining an MDD type is accompanied by definition of several probe functions:

- $base(a) = T$

- $sdom(a) = D$

An MDD type M with base type T and spatial domain D with $T \in \tau$ and $D \in \sigma$ is defined as $\{a \mid a$ is MDD value over base type T and spatial domain D, $T \in \tau, D \in \sigma\}$. An MDD type with base type T and spatial domain D is denoted by $[[T, D]]$.

Further, such operation as 'cell access' will be used very frequently on MDD arrays, therefore it should be defined now.

Let $a = \{(x, a(x)) \mid a(x) \in T, x \in D^d\} \in [[T, D^d]]$ be an MDD value and $x \in D^d$ be a d-dimensional point, then the access operator $[]$: $[[T, D^d]] \times (Z)^d \to T$ is defined as: $a[x] = a(x)$.

2.1.3 Core operations

As reported in [14] Baumann's array algebra stands on two basic operators. They are listed below.

- Array constructor $MARRAY(X, x, e_x)$

Let D be an spatial domain, x be a point variable, e_x be an expression with result type T which contain free occurrences of identifier x. Then $MARRAY(X, x, e_x)$ returns an array $a \in [[T, D]]$ for which $A[x] = e_x$.

- Condense operator $COND(op, X, x, e_{x,A})$

Let X be an array, op be a commutative and associative operation on $T \times T \to T$, x be a free identifier, e_x be an expression with result type T, which contains free occurrences of identifier x and MDD array A. Then $COND(op, X, x, e_{x,A})$ returns a scalar value s being equal to $OP_{x \in X} e_{x,A}$

In some works [13], [8], [15] Baumann's array algebra is defined with additional sort operator $SORT(X, i, e)$, which can be defined as follows: let X be an array, i be a dimension number, r be an expression of some type E on which a total ordering is defined. Let S be a one-dimensional array representing a permutation of elements of a set $J = \{lo(X_i), ..., hi(X_i)\}$ sorted by $r(slice(X, i, j)$ where $j \in J$. In less formal words, $SORT$ operator allows to sort hyperplanes of a hypercube along given dimension by value of r expression evaluated on each hyperplane. In [15] it is used for modeling of such a geo-raster operations as $top-k$ and $median$ operations.

It should be mentioned that from optimization point, it is very important to understand what expression e or operation op is provided for an operator to make best possible optimizations. As reported in [14], all operations for $MARRAY$'s expression e are classified into seven disjoint classes:

- CE_1 : Constant expression
- CE_2 : Cell access with probing point x
- CE_3 : Simple expression on cell at probing point x
- CE_4 : Cell access with cluster preserving index expression
- CE_5 : Access to small neighbourhood of probing point x
- CE_6 : Simple expression on two cells at probing point x
- CE_7 : General expression

Those types of operations help to formally classify all probable expressions provided for cell expression and to exploit that knowledge for finer optimization.

Similarly, all the operations for $COND$'s operation op are considered to be from one of the following class:

- CA_1 : Cell access with probing point x
- CA_2 : Simple expression on cell with probing point x
- CA_3 : General expression

Generally speaking, classes CE_1 and CA_7 incorporate expressions to which an optimizer cannot apply any modifications to optimize the $MARRAY$ performance.

2.1.4 Derived operations

When core operation are defined, multiple additional ones are built on top the low-level core operations to have a convenient notation for frequently used typical operations. Such operations are called *derived* operations and there are several types of them: geometric, induced, binary induced, aggregate induced.

Geometric operations

These operations are some special cases of application of $MARRAY$ constructor. Let A be an MDD of type $[[T, D]]$, K be a result of application of $trim$ with some parameters to multidimensional interval D, i be a dimension number, v some valid point along i-th dimension of domain D. Then the following operations can be defined:

- $trimming_K(A) = MARRAY(K, x, A[x])$
- $slice_{(i,v)}(A) = MARRAY(slice(sdom(A), i, v), x, A[x])$

All $MARRAY$ cell expressions in geometric derived operations belong to class CE_2.

Induced operations

These operations are again some special cases of application of $MARRAY$ constructor. Let A be an MDD of type $[[T, D]]$, B be an MDD of type $[[T', D]]$, op be a function $T \to D$, $binOp$ be a function $T \times T' \to T''$. Then the following operations are defined:

- Unary induced operation $\overline{op}(A) : [[T, D]] \to [[T', D]]$ which is the replacement to $MARRAY(D, x, op(A[x]))$
- Binary induced operation: $\overline{binOp}(A, B) : [[T, D]] \times [[T', D]] \to [[T'', D]]$ which is equivalent to $MARRAY(D, x, binOp(A[x], B[x]))$
- Left induced operation: $\overline{op}_{left}(A, const) : [[T, D]] \times T' \to [[T'', D]]$ which is a shorter notation for $MARRAY(D, x, binOp(A[x], const))$
- Right induced operation: $\overline{op}_{right}(const, A) : T' \times [[T, D]] \to [[T'', D]]$ which is equivalent notation for $MARRAY(D, x, binOp(const, A[x]))$
- Note that in case of unary induced operations $MARRAY$ cell expression belongs to CE_3 class and in case of binary induced operations to class CE_6.

Aggregate induced operations

The main convenient operation is $reduce$ operation on which other derived aggregates are based. Let A be an MDD of type $[[T, D]]$, op be an associative and commutative binary operation defined and closed on T. Then $reduce$ operation can be defined as follows: $reduce(op, D, A)$ = $COND(op, D, x, A[x])$. Then several convenient operations can be defined as follows:

- $sum\_cells(A) = reduce(+, sdom(A), x, A[x])$
- $mult\_cells(A) = reduce(*, sdom(A), x, A[x])$

- $avg\_cells(A) = \dfrac{sum\_cells(A)}{card(sdom(A))}$
- $min\_cells(a) = reduce(\min, sdom(A), x, A[x])$
- $max\_cells(a) = reduce(\max, sdom(A), x, A[x])$

2.1.5 Extended relational model

In order to examine MDD specific optimization techniques in combination with set based query processing, the MDD Model is integrated into an adapted Relational Model. The attribute domain of multi-dimensional values can be specified differently:

- Base type is fixed. Spatial is domain unknown
- Base type is fixed. Spatial domain's dimensionality is fixed
- Base type is fixed. Spatial domain is fixed
- Base type is fixed. Spatial domain is fixed with its physical representation (violation of 'hidden physical representation' principle)

The type of attribute domain sets amount of restrictions on MDD values that can be effectively used by an optimizer.

Formally speaking, let $D_i \in \delta$ be spatial domains, $T_i \in \tau$ be types and d_i be positive integers. Let A_i be attributes with $dom(A_i) \in \{[[T_i]], [[T_i, d_i]], [[T_i, D_i]], T_i\}$. Then the mutidimensional relation R is defined as $R \subseteq dom(A_1) \times dom(A_2) \times ... \times dom(A_n)$. When attribute domains are formalized in the model, three relational operations are defined. Let R and S be relations, $cond$ be a predicate on R, op_j be a function defined on R and returning either scalar or multidimensional values for each j. Then relational operations are defied as follows:

- Selection Application σ

  $\sigma_{cond}(R) = \{t \in R \mid cond(t)\}$
- Cross product $\times$

  $\times(R, S) = \{t \mid t = (r_1, r_2, ..., r_n, s_1, s_2, ..., s_n), (r_1, ..., r_n) \in R, (s_1, ..., s_n) \in S\}$
- Application α

  $\alpha_{op_1,...,op_s}(R) = \{t \mid t = (op_1(y), ..., op_s(y)), y \in R\}$

The first two operations are very similar to those in canonical relational model, however, the third operator differs significantly. What it does is application of the provided operators to all of the tuples Relational projection can be expressed through projection operation with special functions returning an element of a tuple at the specific position.

2.1.6 Formalizing Array Query Processing

An array query can be represented as a special graph. Each node of the graph is represented by an operator which comprises the query. This graph is a tree and consists of set trees and element trees. Set trees include relational operations as inner nodes and MDD relations(see definition above) as leaves, whereas element trees' inner nodes are MDD/logical operations. Leaves are MDD constants/iterators. There are also some specific kinds of trees for naming convenience: *condition trees* and *operation trees*. The former are element trees representing boolean multidimensional expressions attached to a select node. The latter are element trees representing some multidimensional expression attached to an application node.

The mere query graph represents data flow between operator nodes. A single edge transfers only particular type of data, thus all edges can be classified into distinct categories depending on type of data flowing through it. The whole edge set is divided into non-intersection sets of three types: relational, dimensional, scalar types. Data edges carrying relations comprises relation sets, edges carrying raster data - dimensional ones, and edges carrying non-dimensional or scalar values are those forming scalar sets. According to this classification, the graph is partitioned in subgraphs of maximal size, each of which contains only one sort of edges. Such subgraphs are called *areas* and are in particular called *relational data areas (RDAs)*, *dimensional data areas (DDAs)* and *scalar data areas (SDAs)*, depending on the type of edges they contain. Optimizing data flow in DDAs is of primary importance in extended relational model for array query processing.

In general, when a query optimizer receives a query it processes it in three stages:

- Rewriting
- Transformation
- Execution

During *rewriting* a query is represented in a *normal query form* which is based on logical optimization rules and the following key principles:

- Eager constant subexpressions evaluation.

This saves computational resources as those expressions might be needed to be computed for each cell of MDD objects

- Boolean expressions normalization aimed at application of optimization rules.

All boolean expressions are transformed to CNF or DNF depending on the predicates in order to let the optimizer detect patterns for application of logical optimization rules

- Prepare induction expressions for the application of optimization rules.

Such a modification leverages associativity and distributivity of induced operations, which can replace MDD operations by scalar ones, dramatically reducing computation cost of the expression

In [14] more than one hundred logical optimization rules are presented. Those rules are mainly based on *load optimization* for geometric operations; exploitation of operations' beneficial properties (such as associativity, distributivity) for induced, binary induced and aggregate operations; movement of individual α, σ subexpressions through cross product operation for set trees.

In the first case, geometric operations are pushed down to multidimensional nodes serving as data sources for upper nodes. This potentially reduces amount of I/O needed to process a single MDD value by an upper node. An example of such a rule is $trimming_D(unOpInduced(A)) \rightarrow unOpInduced(trimming_D(A))$

where $unOpInduced$ is an unary induced operation, A is an MDD value.

In the second case, rewriting expressions leveraging beneficial properties of an operation on which some induced, binary induced or aggregation operation is based may reduce the amount of multidimensional operations in an expression. A remarkable example of such a rule is

$$binOp(binOpInduced(A,b),c) \rightarrow binOpInduced(A,binOp(b,c))$$

In the third case, computation effort is potentially diminished by reducing the amount of multidimensional operations performed. For example, the amount of multidimensional predicate evaluation may be diminished as in case of application of the rule

$$\sigma_{condR and\ condS}(R \times S) \rightarrow \sigma_{condR}(R) \times \sigma_{condS}(S)$$

where R, S are relations, $condR, condS$ - predicates defined on R and S respectively.

In general, query rewriting is reported to be notably profitable if following heuristics are taken into account:

- Perform geometric operations eagerly (load optimization rewriting)
- Reduce number and overall cardinality of Dimensional Data Areas as much as possible

Abiding by this rule lets to diminish the number of edges in DDAs by applying rules that eliminate MDD expressions or transform them into scalar ones.

- Perform applications eagerly

Similar to pushing down projections while using greedy algorithms in relational query processing [16], which is not always the best choice [17]. However, lack of join conditions lets sieving down application into cross product to reduce amount of

tuples for which given functions are applied. Application is an expensive operation as MDD values typically have plenty of cells to operate on.

- Perform selections eagerly

This heuristics also resembles the common heuristics in relational query processing as noted in [18]. Selections are pushed down to diminish operation sets as early as possible. Scalar predicates are given priority over MDD ones.

- Search for common subexpressions

Common subexpressions are stored as intermediate results. Doing one unit of work several times is useless and even costly when operating on large MDD values.

During *transformation* logical plan operations are mapped to physical plan's operations. Such a mapping is not the distinctive feature of RasDaMan, for example, AML (described in Sec. 2.2) also maps logical operations to physical ones. Typically multiple physical plans are valid and semantically equivalent. Those might be analytically compared via usage of special array cost models which were devised for corresponding algebras in [19], [14]. However, being able to just compare physical plans using cost functions is not always sufficient, it is crucial that some physical plan refinement techniques are exploited whose aim is to try to reduce the cost of a physical plan by accounting physical layout of the processed MDD values and to adjust the iteration order correspondingly. In RasDaMan system each type operation is considered separately.

Transformation of induced and aggregate operations is pretty straightforward. Main idea of transformation in such cases is to provide parallel tile processing. However, transformation of binary induced operations is a bit more tricky and we will focus on them in more detail further.

Binary induced operations are optimized by trying to find the optimal tile traversing order over tiles comprising binary induced operands. Finding optimal tile traversing order minimizes disk reads as the main bottleneck during MDD values processing, leveraging efficient exploitation of main memory. The problem can be formalized using graph terminology. Let $G = (V, E)$ be a graph, where $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \varnothing$, so that V_1, V_2 represent tile sets forming the first and the second operands of the binary induced operation respectively. The edge $(v_1, v_2) \in E$ if and only if tiles corresponding to v_1, v_2 need to be processed simultaneously during binary induced operation performance. The result graph is a bipartite graph, as any edge from E has the one end in V_1 and another in V_2. An edge in the graph may intuitively be perceived as an indicator of represents a need for holding two tiles from different sets in main memory for faster processing. However, to process the tiles in main memory those tiles should be, obviously, loaded into the main memory first, unless they are already in place. Loading of a tile is an expensive operation as it is directly related to expansive disk I/O, hence the amount loads should be diminished as much as possible. If there is no cache and only two tiles can be held in main memory simultaneously at a time then the

problem of minimization of disk access can be formulated as follows: find a vertex traversing order $k_1, k_2, \mathrm{K}\ k_n$, minimizing amount of disk access.

The tile traverse algorithm for binary induced operations has been pondered in [14]. Disk access minimization problem is reduced to the problem of minimization of a special *cost* function that is defined as $cost(k_1, k_2, ..., k_i) = cost(k_1, k_2, ..k_{i-1}) + (isAdj(k_i, k_{i-1})\ and\ k_i > 1)\ ?\ 1 : 2$.

The minimum is obtained when the sequence $k_1, k_2, \mathrm{K}\ , k_n$ forms a Hamilton cycle. In generic case determination of whether such a sequence of vertices exist is known to be an NP-complete problem [20].

However, the restriction to visit each vertex only once can be relaxed. Such an approach has been used in [21] for facing the problem of finding the optimal tile traversing order for array join - a special case of binary induces operation. In this paper the requirement of exclusive visit of each vertex is replaced with requirement of visiting each edge only once with intention to minimize multiple reads of a tile. Authors exploit Hierhozer and Weiner necessary and sufficient condition to find an Euler circuit, i.e. a circuit that visits each edge in the graph only once. As the Hierholzer and Wiener criteria claims [22] the presence of Euler circuit in a graph is equivalent to its connectivity and all vertices having even degrees. Authors split the graph G into connected components, augmenting each with extra auxiliary edges so that each vertex has an odd degree and Euler circuit exists. For each component an Euler circuit is computed and tile traverse path is determined. The result traverse path is built from a random permutation of components' paths. The authors report that it is still an open question how this approach may be adapted to either a distributed environment where each tile load may have its own cost and or to a presence of an arbitrary sized cache.

2.2 AML algebra

2.2.1 Array abstraction

In *Array Manipulation Language (AML)* [23] an array A is set by its shape S, domain D (which is conceptually similar to Array's algebra value set), and the mapping M that establishes the link between the array's shape and domain. $S[i]$ represents the extent of A along i-th dimension, as all cells of an AML array has lower bound equal to zero. A vector x is said to be in array's shape S iff $\forall i 0 \le x[i] < S[i]$. Said that, we can define the mapping M more formally as a function that returns some value from D for every $x \in S$ and a special $null \notin D$ element otherwise.

2.2.2 Algebra operations

AML makes use of *bit patterns* and several probe functions defined on those patterns (*index* and *count*). The $index(P,k)$ function returns the index of the $k-1$-th 1 in the bit pattern P if it does not equal to (0), otherwise the function returns 0. The $count(P,k)$ function will return number of 1 up to k-th position in the bit pattern B.

Let A,B be the multidimensional arrays with different shapes S_1, S_2 and common domain D with d dimensions, P, P_i be some bit patterns, D_f, R_f be some shapes called *domain* and *range* boxes respectively; then the core operations on arrays in AML algebra are:

- Subsection. $C = SUB_i(A,P)$

The SUB_I operator iterates over hyperplanes along i-th dimension and, if allowed by the corresponding bit in P, concatenates the hyperplane to the result array.

- Merging $C = MERGE_i(A,B,P,\delta)$

The $MERGE$ operator cyclically glues together hyperplanes cut along i-th dimension according to pattern P. Hyperplane are taken from A for set bit in the pattern and from B otherwise. If source array has no values hyperplane is filled with δ value.

- Function application. $APPLY(A, f, D_f, R_f, P_0, \mathrm{K}, P_{d-1})$

Iterates over slabs of shape D_f from A, checks whether that slab is "allowed" by all of the P_i patterns (by looking through corresponding bits over all patterns and rejecting the slab if some of the bits is not set). If the slab is "allowed" function f is applied to the slab and a slab of shape R_f is obtained. The result slab is "glued" to the result array from the side indicated by iteration position (more formally to ($count(P_0,i_1),\mathrm{K},count(P_{d-1},i_{d-1})$ where vector (i_1,K,i_n) represents the iteration vector)).

The $APPLY$ operator might seem to resemble the $MARRAY$ operator defined in Baumann's array algebra. Indeed, $MARRAY$ operator can be called as a more restrictive version of $MARRAY$ operator with range boxes equalling a single array cell. In AML function f is a black box just as an expression of class CE_7 for $MARRAY$ operator.

2.2.3 Optimization

AML optimization techniques, similarly to those in Baumann's algebra, can be classified into logical and physical ones. In AML those techiques are applied in three phases, so called *rewriting phase* and *plan generation phase*, *plan refinement phase*.

During *rewriting phase* an AML optimizer receives an AML query, builds query expression tree, and transforms it to semantically equivalent one using algebra's transformation rules. The transformed expression is guaranteed to be evaluated to the same result as the intial one, however the transformed query is hoped to be executed faster for some reason (e.g. due to operating on smaller amount of cells as in case of *load optimization* in Baumann's array algebra, see Sec. 2.1.6). In [19], [23] several algebraic rules are presented. The set of logical optimization rules is not so diverse compared to those devised in [14]. The approaches used by AML optimizer is similar to those exploited by Baumann's Algebra optimzer for *load optimization*. Examples of AML optimizer's query transformations are: merging several subsample operators into one, pushing subsample through merge in some cases, pushing subsamples through apply in some cases, etc. As AML's algebra operators by nature are superior to Baumann's Algebra ones in terms of flexibility (e.g. bit pattern support in all elementary operators, differently sized range boxes for function application operator), logical optimization rules become more complex, accompanied with extra initial conditions and overburdened by auxiliary bit pattern calculations. For example, some generic optimizations exploited by a Baumann's algebra optimizer, like reducing the cardinality of processed cells set before function application, cannot be simply borrowed by AML optimizer for use. This occurs due to the fact that additional logic is introduced by bit patterns and possibly different sizes of range and domain boxes. However, overall relative complexity of AML's logical transformation rules does not diminish the AML optimization potential. At *plan generation phase* rewritten expression tree is mapped to a physical plan just as during *Transformation* phase of Baumann's optimizer. The physical plan is represented by a directed graph where a vertex represents a logical operator, while an edge depicts a data flow. Every operator expects a stream of non-overlapping chunks (*tiles* in RasDaMan terminology) of some particular shape and in some particular order as an input. Operators produce non-overlapping array chunks of particular shape and in some particular order as an output. An operator may have some parameters which specifies its behavior. The summary on the physical operators is provided in table 1.

Table 1. AML physical operations

Name	Input Stream	Parameters	Description
APPLY_P	1	function	Applies a function to an array
LEAF_P	0	array	Reads array from disk
COMBINE_P	> 0	combination	Maps input slabs (in each

		map	dimension) to output slabs. Used for *MERGE* and *SUB*
REGROUP_P	1	-	changes input chunks' shapes
REORDER_P	1	-	permutes input chunks

The building of the physical operator tree is based on recursive top-down traversing of the expression tree. The algorithm step can be determined depending on the currently processed node of the expression tree. Algorithm of building the physical plan tree is summarized in tab. 2.

Table 2. AML physical plan tree building algorithm

Current tree root node is ...	Action
Nonleaf *APPLY* with domain box D_f and range box R_f	Add **APPLY_P** and **REGROUP_P** operators where * **REGROUP_P** precedes **APPLY_P** * **APPLY_P** input chunk shape matches D_f ; output – R_f * Application mask are taken from *APPLY* patterns
SUB or *MERGE* with n leaves	Find max tree of *SUB* and *MERGE* rooted at current node. Translate it to n-ary **COMBINE_P** and n **REGROUP_P** operators
Leaf *APPLY*	Translate it to **LEAF_P**

The main aims of the optimizer on *plan refinement phase* are to remove no-op operators and specify chunk ordering of each operator. Chunk iteration order directly affects the amount of data buffed by an operator, therefore the optimizer tries to minimize the memory requirement so that try to execute entirely in memory not to spent effort on materializing intermediate results of evaluation. For d-dimensional array consisting of q chunks there are $q!$ iteration orders. If a plan consists of multiple operators consuming several arrays then considering all iteration orders becomes exponentially expensive. AML authors decided that the optimizer should consider only $\overline{d}$ iteration orders for each operator, where $\overline{d}$ is the maximum dimensionality of an array consumed in the plan. Each of those $\overline{d}$ iteration order differs in the primary dimension by which all the input chunks are ordered. Other sort dimensions are taken in dimension number increasing order. Authors claim that *Hilbert* curve [24] or Z-order [25] could be considered by an optimizer as those orders might be related to secondary storage scheme, but those types of orders are neglected for the sake of simplicity.

For physical plan consisting of w operators there will be $\overline{d}^{w}$ iteration orders. There is another problem that even if an optimal iteration order is found for some operator it does not mean that optimizer is done with this operator. There might be another dependent operator expecting output chunk stream of the just considered operator in a completely different order. However, instead of examination of another possible chunk ordering it might be more beneficial to insert a reorder operator between the producer and the consumer operators. Having addressed the aforementioned problems, AML authors devised a cost-based algorithm reducing the complexity of assigning chunk iteration orders to operators down to $O(w\overline{d}^{2})$. The algorithm is briefly discussed below.

Let $C_i(x)$ be cost of choosing output to be returned in i-th order for operator x. Let $Y(x)$ be set of x neighbors. If the x is **LEAF_P** then cost function is equal to operator memory cost (defined for each operator separately), otherwise: $C_i(x) = c_i(x) + \sum_{y \in Y(x)} \left(\min\left(C_i(y), \min_{i \neq j} \left(C_j(y) + reorderCost(j,i,x,y)\right)\right)\right)$

Here $reorderCost(j,i,x,y)$ shows the additional cost augmented after inserting a **REORDER_P** operator between x and y operator reordering j-th ordered chunk stream into i-th order one.

2.3 SciDB

To the best of our knowledge the formal SciDB algebra description with possible derived optimization techniques has not been yet published. This may be explained by the fact that, formally speaking, there is no formal algebra in SciDB, as it will be seen below, and all operators are initially considered to be user defined functions (*UDF*s). There are some built-in operators, but they do not form any algebra, and are still considered UDFs, as SciDB pays much attention to extensibility. Despite this fact, here we try to formalize our knowledge about SciDB and structure it using the plan used for AML and Baumann's array algebra. First we define an array, then list built-in elementary operations of SciDB and finally try to explore how SciDB query optimizer works.

2.3.1 Array abstraction

SciDB operates on collection of n-dimensional arrays each of which again may be represented as a mapping

$$A: I_1 \times I_2 \times \mathrm{K} \times I_n \rightarrow (V_1 \times V_2 \times \mathrm{K} \times V_k) \cup \{NULL\}$$

Sets I_1, I_2, , I_n are closed subsets of Z^n just as in case of Baumann's array algebra. Value associated with each index vector (i_1, K, i_n) represents an array's *cell*. What needs more attention here is the nature of a SciDB array's cell. As it can

be seen from formal SciDB array definition the cell value is actually a heterogeneous *tuple* or a special *NULL* value. The presence of *NULL* value gives lets SciDB to store *sparse* arrays just out of the box. Cells which are mapped to *NULL* are called *empty*. Each tuple element can be addressed by so called *attribute* name. It should be mentioned, there are some constraints on value sets V_j, which restrict a tuple value to be of one of some predefined types: fixed length string, number, etc. However, users of SciDB are given an opportunity to compose custom types (user defined types – *UDT*s). One of the features of the SciDB array data model is that it is *nested*, allowing an array cell contain another array.

2.3.2 Algebra operators

One of the most distinguishable feature of SciDB is that as reported in [26] it has no built-in operators, forming some rigor algebraic system. Authors claim that all operators are in fact UDFs and SciDB has some embedded UDFs, which to some extent may be perceived as elementary operators forming SciDB base algebra. However, this might seem as contradiction to the SciDB ideology. In [27] the term 'algebra' is used, but formalism is avoided and just built-in operator usage examples are provided. Below we describe SciDB built-in operators mentioned in [27] and specified in online SciDB documentation [28].

Let

- A be an n-dimensional SciDB array, B be m-dimensional array
- V be a *array index values* for A,
  where *array index values* can be defined as a set of pairs $\{(i_1, v_1), \mathrm{K}, (i_p, v_p)\}$ where for every $0 \le j < n$: $0 \le i_j < n$, i_j values are distinct and v_j is an admissible index value for array's dimension j.
- Q be a predicate containing free occurrence of A cell's dimensional index
- E be a predicate containing free occurrences of A cell's attributes
- F be a function mapping A cell attributes values to some admissible cell type

Then the set of the following operators may be described:

- $Slice(A,V)$

Gets out the subarray of dimensionality $dim(A) - |V|$ from array A, taking A as buffer array and sequentially cutting out hyperplanes from buffer array based on elements in V.

- $Subsample(A,Q)$

Gets out the subarray from array A building it from those cells whose index over dimension j satisfies the predicate Q for every admissible j.

- $SJoin(A,B)$

Combines the cells' values of two input arrays if cells have the identical dimensional indices.

- $Filter(A,E)$

Gets an array of the same dimensionality as A where each cell is taken from A *iff* E evaluates to true on it, or considered *empty* otherwise.

- $Apply(A,F)$

Gets a new array applying F to a cell (substituting cell's corresponding attribute values to F) and storing the result of the calculation in the result array's cell with the same dimension index.

It should be mentioned that SciDB orients on high extensibility, which explains the shift of SciDB towards UDFs (and UDTs). ScidDB provides a special facility that enables to extend the aforementioned set of operators with custom ones, written in C++. Custom operators are required to take array input(s) and return array output(s). Moreover, SciDB supports defining own aggregates (user defined aggregates – *UDA*s), increasing the degree of extensibility even more.

2.3.3 Optimization

The shift to the paradigm 'everything is defined by a user' makes query planning a much harder task. The SciDB optimizer operates on 'blackbox' operators which may theoretically be optimized, but in general case their nature is too generic for the optimizer to determine optimizations it might perform. However, compared to AML and RasDaMan systems, SciDB tries to overcome this difficulty with optimizations basing more on physical data storage and parallelism. Those optimizations are discussed below.

When a query is got an optimizer builds a logical plan ding all the required semantic checks. As it is stated in [27] the optimizer will produce a complete physical plan corresponding to the built logical plan, where possible. Otherwise it will split the query plan into subplans consisting of pipelinable operators and execute them (in parallel, taking into account the physical structure, discussed further). One logical optimization of SciDB query optimizer mentioned in [29] is detecting commuting operations and pushing them down in the query tree. However, due to generic nature UDFs finding such operators is typically a luck.

When physical information comes in use, SciDB optimizer starts to 'breathe easier'. A SciDB instance is supposed to run on multiple nodes, adhering to *shared nothing*

design. A central *system catalog* exists, storing meta information about user-defined extentions, data distribution, etc. By such SciDB enables to provide a high level of parallelizm and related performance improvements accounting the fact that array data manipulations are known to be CPU bound([14], [29]). SciDB optimizer makes use of distributed architecture and performs several related optimizations discussed in [29]. For example, the optimizer examines the logical tree for blocking operations, i.e. those which require a temporary array to be constructed (e.g. operations demanding redistribution of data in order to execute). In [29] built-in SciDB optimizer is reported to be an incremental and cost-based one. It means that the optimizer picks the best choice for the first subtree to execute making use of a cost model for plan evaluation. The same paper states the SciDB optimizer can be called a 'simple optimizer' which tries to minimize amount of date movement and increase the level of parallelism.

However, different optimization techniques has been recently proposed, which might be used in SciDB environment. Those include devising *iterative array processing* model for a parallel array engine proposed in [30], optimization of SciDB's *Filter* operator proposed in [31], shuffle join optimization framework for the SciDB array data model presented in [32], etc.

3. Possible directions of investigation

Based on the overview of the optimization process provided above we summarize some directions that are available for further investigation. Under no circumstances should this list be perceived as complete one. The list below is just a set of noticed future work directions which is actually much larger.

- **Baumann's Array Algebra. Array Query Processing. Commutativity of slice and trimming is not accounted during optimization**

How can this property be exploited for load optimization?

- **Baumann's Array Algebra. Array Cost Model. Approximating selectivity of predicates containing MDD expressions using common techniques for AQP approach**.

In relational query processing there are three well-known techniques: Sampling, Parametric, Histogram-based Techniques. RasDaMan creators opted for Histogram-based approach, saying that parametric techniques have a problem that real distributions (especially those of operations results) are seldom accurately approximated by mathematical distributions. The problem is very serious in case of raster image data. Sampling techniques are reports to be very flexible and tolerant to updates. The main disadvantage of such an approach is that sampling has have considerable I/O and CPU overhead and lacks computation result reusability. How serious is that overhead and how disadvantages overweigh advantages?

- **Baumann's Array Algebra. Cell expressions analyzing**

It is possible to analyze cell expressions for the elementary operations (*MARRAY* , *COND*) in order to optimize disk access, e.g. detect tiles which should be cached iteration over MDD value.

- **Baumann's Array Algebra. Optimization of binary induced operations. Array join problem. Optimization for relaxed restrictions**

When the tile graph is built as in [21] the cost of fetching a tile from disk is considered the same for all tiles. However, this might be not the case for a distributed environment or in presence of cache with size allowing to hold more than 1 tile of each operand. The approach presented in [33] might be considered.

- **AML. Plan refinement. Iteration order**

Authors claim that *Hilbert* curve or Z -order could be accounted by an optimizer during plan refinement as those orders might be related to storage scheme, but those types of orders are dismissed from consideration for the sake of simplicity. Can such a simplification be revisited?

4. Conclusion

In the current paper we have investigated the theoretical background of array databases exploring three different mature array database management systems: RasDaMan, AML, SciDB. We have looked at those database from a fixed perspective: firstly, we explore the data model the system uses to simply define an *array*; secondly, we examine what formal algebra the system constructs above arrays; thirdly, we take a closer look on algebraic optimizations (logical optimizations) and those applied when information about physical storage and retrieval of data is taken into account (physical level optimizations). We collect some possible directions of further investigation for considered array databases. The list of directions mainly contain ones outlined by the authors of the array databases themselves.

References

[1]. Peter Baumann. Raster Data Management and Multi-Dimensional Arrays, pages 2332-2339. Springer US, Boston, MA, 2009.
[2]. Hubble telescope images. https://www.spacetelescope.org/images/.
[3]. Large hadron collider storage. http://lhcb-public.web.cern.ch/lhcb-public/en/Data
[4]. Gilberto Câmara, Lúbia Vinhas, Karine Reis Ferreira, Gilberto Ribeiro De Queiroz, Ricardo Cartaxo Modesto De Souza, Antônio Miguel Vieira Monteiro, Marcelo Tílio De Carvalho, Marco Antonio Casanova, and Ubirajara Moura De Freitas. TerraLib: An Open Source GIS Library for Large-Scale Environmental and Socio-Economic Applications, pages 247-270. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
[5]. PostGIS official web page. https://postgis.net/.
[6]. SpatialLite web page. https://www.gaia-gis.it/fossil/libspatialite/home.
[7]. Oracle GeoRaster documentation. https://docs.oracle.com/cd/B19306_01/appdev.102/b14254/geor_intro.htm.

[8]. Baumann P. and Holsten S. A comparative analysis of array models for databases. In Database Theory and Application, Bio-Science and Bio-Technology. Communications in Computer and Information Science, volume 258, 2011.
[9]. Rasdaman home page. http://www.rasdaman.org/.
[10]. Paul G. Brown. Overview of scidb: large scale array storage, processing and analysis. In SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 963-968, 2010.
[11]. Paulo Jorge Pimenta Marques. Arbitrary tiling of multidimensional discrete data cubes in the rasdaman system. 1998.
[12]. L.T. Chen, R. Drach, M. Keating, S. Louis, D. Rotem, and A. Shoshani. Efficient organization and access of multi-dimensional datasets on tertiary storage systems. Information Systems, 20(2):155-183, 1995. Scientific Databases.
[13]. Peter Baumann. A database array algebra for spatio-temporal data and beyond. 06 1999.
[14]. Roland Ritsch. Optimization and evaluation of array queries in database management systems. 12 1999.
[15]. A. G. Gutierrez and P. Baumann. Modeling fundamental geo-raster operations with array algebra. In Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), pages 607-612, Oct 2007.
[16]. Frank P. Palermo. A data base search problem. 01 1974.
[17]. Joseph M. Hellerstein. Optimization techniques for queries with expensive methods. ACM Trans. Database Syst., 23(2):113-157, June 1998.
[18]. A. Swami. Optimization of large join queries: Combining heuristics and combinatorial techniques. SIGMOD Rec., 18(2):367-376, June 1989.
[19]. Arunprasad P. Marathe and Kenneth Salem. Query processing techniques for arrays. SIGMOD Rec., 28(2):323-334, June 1999.
[20]. Hamiltonian cycle. http://mathworld.wolfram.com/HamiltonianCycle.html.
[21]. P. Baumann and V. Merticariu. On the efficient evaluation of array joins. In 2015 IEEE International Conference on Big Data (Big Data), pages 2046-2055, Oct 2015.
[22]. Ethan Kim. Comp 251: Data structures and algorithms. https://ethkim.github.io/TA/251/eulerian.pdf.
[23]. Arunprasad P. Marathe and Kenneth Salem. A language for manipulating arrays. In Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB '97, pages 46-55, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
[24]. Hilbert curve. http://www4.ncsu.edu/ njrose/pdfFiles/HilbertCurve.pdf.
[25]. Z-curve general information. http://wiki.gis.com/wiki/index.php/Z-order_(curve).
[26]. P. Cudre-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik. A demonstration of scidb: A science-oriented dbms. Proc. VLDB Endow., 2(2):1534-1537, August 2009.
[27]. Paul G. Brown. Overview of scidb: Large scale array storage, processing and analysis. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pages 963-968, New York, NY, USA, 2010. ACM.
[28]. Scidb doucumentation. https://paradigm4.atlassian.net/wiki/spaces/ESD/overview.
[29]. Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman. The architecture of scidb. In Proceedings of the 23rd International Conference on Scientific and Statistical Database Management, SSDBM'11, pages 1-16, Berlin, Heidelberg, 2011. Springer-Verlag.

[30]. Emad Soroush, Magdalena Balazinska, Simon Krughoff, and Andrew Connolly. Efficient iterative processing in the scidb parallel array engine. In Proceedings of the 27th International Conference on Scientific and Statistical Database Management, SSDBM '15, pages 39:1-39:6, New York, NY, USA, 2015. ACM.

[31]. Sangchul Kim, Seoung Gook Sohn, Taehoon Kim, Jinseon Yu, Bogyeong Kim, and Bongki Moon. Selective scan for filter operator of scidb. In Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM '16, pages 28:1-28:4, New York, NY, USA, 2016. ACM.

[32]. Jennie Duggan, Olga Papaemmanouil, Leilani Battle, and Michael Stonebraker. Skew-aware join optimization for array databases. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, pages 123-135, New York, NY, USA, 2015. ACM.

[33]. Weijie Zhao, Florin Rusu, Bin Dong, and Kesheng Wu. Similarity join over array data. In Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16, pages 2007-2022, New York, NY, USA, 2016. ACM.

Базы данных для обработки массивов: взгляд изнутри

В.А. Павлов <vlad.pavlov24@gmail.com>
Б.А. Новиков < b.novikov@spbu.ru >
Санкт-Петербургский государственный университет,
199034, Россия, Санкт-Петербург, Университетская набережная, д. 13б

Аннотация. После появления огромного количества научных данных, которые необходимо было хранить и обрабатывать, в мире баз данных возникла задача поддержки больших многомерных массивов. Стала необходимой разработка специальных баз данных, которые основывались бы на модели данных, "сердцем" которой было понятие массива (array). Разработка хорошо организованной системы управления базой данных, базирующейся на нетрадиционной модели данных, требовала решения следующих задач: формальное определение модели данных, основывающейся на понятии массива; построение формальной алгебры, работающей с объектами модели; разработка правил оптимизации запросов на логическом уровне, а затем и на физическом. Эти задачи уже решались создателями специальных баз данных, настроенных на обработку и хранение массивов (array databases). В данной работе рассматриваются понятия массива, формальные алгебры и методы оптимизации запросов в таких развитых базах данных, как RasDaMan, AML, SciDB – базах данных, ориентированных на хранение и обработку крупных многомерных массивов.

Ключевые слова: базы данных для обработки массивов; обзор; формальная алгебра баз данных для обработки массивов; обработка запросов к базам данных для обработки массивов; оптимизация запросов к базам данных для обработки массивов; AML; RasDaMan; SciDB

DOI: 10.15514/ISPRAS-2018-30(1)-10

Для цитирования: Павлов В.А., Новиков Б.А. Базы данных для обработки массивов: взгляд изнутри. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 137-160. DOI: 10.15514/ISPRAS-2018-30(1)-10

Список литературы

[1]. Peter Baumann. Raster Data Management and Multi-Dimensional Arrays, pages 2332-2339. Springer US, Boston, MA, 2009.

[2]. Hubble telescope images. https://www.spacetelescope.org/images/.

[3]. Large hadron collider storage. http://lhcb-public.web.cern.ch/lhcb-public/en/Data

[4]. Gilberto Câmara, Lúbia Vinhas, Karine Reis Ferreira, Gilberto Ribeiro De Queiroz, Ricardo Cartaxo Modesto De Souza, Antônio Miguel Vieira Monteiro, Marcelo Tílio De Carvalho, Marco Antonio Casanova, and Ubirajara Moura De Freitas. TerraLib: An Open Source GIS Library for Large-Scale Environmental and Socio-Economic Applications, pages 247-270. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[5]. PostGIS official web page. https://postgis.net/.

[6]. SpatialLite web page. https://www.gaia-gis.it/fossil/libspatialite/home.

[7]. Oracle GeoRaster documentation. https://docs.oracle.com/cd/B19306_01/appdev.102/b14254/geor_intro.htm.

[8]. Baumann P. and Holsten S. A comparative analysis of array models for databases. In Database Theory and Application, Bio-Science and Bio-Technology. Communications in Computer and Information Science, volume 258, 2011.

[9]. Rasdaman home page. http://www.rasdaman.org/.

[10]. Paul G. Brown. Overview of scidb: large scale array storage, processing and analysis. In SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 963-968, 2010.

[11]. Paulo Jorge Pimenta Marques. Arbitrary tiling of multidimensional discrete data cubes in the rasdaman system. 1998.

[12]. L.T. Chen, R. Drach, M. Keating, S. Louis, D. Rotem, and A. Shoshani. Efficient organization and access of multi-dimensional datasets on tertiary storage systems. Information Systems, 20(2):155 - 183, 1995. Scientific Databases.

[13]. Peter Baumann. A database array algebra for spatio-temporal data and beyond. 06 1999.

[14]. Roland Ritsch. Optimization and evaluation of array queries in database management systems. 12 1999.

[15]. A. G. Gutierrez and P. Baumann. Modeling fundamental geo-raster operations with array algebra. In Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), pages 607-612, Oct 2007.

[16]. Frank P. Palermo. A data base search problem. 01 1974.

[17]. Joseph M. Hellerstein. Optimization techniques for queries with expensive methods. ACM Trans. Database Syst., 23(2):113-157, June 1998.

[18]. A. Swami. Optimization of large join queries: Combining heuristics and combinatorial techniques. SIGMOD Rec., 18(2):367-376, June 1989.

[19]. Arunprasad P. Marathe and Kenneth Salem. Query processing techniques for arrays. SIGMOD Rec., 28(2):323-334, June 1999.

[20]. Hamiltonian cycle. http://mathworld.wolfram.com/HamiltonianCycle.html.

[21]. P. Baumann and V. Merticariu. On the efficient evaluation of array joins. In 2015 IEEE International Conference on Big Data (Big Data), pages 2046-2055, Oct 2015.

[22]. Ethan Kim. Comp 251: Data structures and algorithms. https://ethkim.github.io/TA/251/eulerian.pdf.

[23]. Arunprasad P. Marathe and Kenneth Salem. A language for manipulating arrays. In Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB '97, pages 46-55, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[24]. Hilbert curve. http://www4.ncsu.edu/ njrose/pdfFiles/HilbertCurve.pdf.
[25]. Z-curve general information. http://wiki.gis.com/wiki/index.php/Z-order_(curve).
[26]. P. Cudre-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik. A demonstration of scidb: A science-oriented dbms. Proc. VLDB Endow., 2(2):1534-1537, August 2009.
[27]. Paul G. Brown. Overview of scidb: Large scale array storage, processing and analysis. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pages 963-968, New York, NY, USA, 2010. ACM.
[28]. Scidb doucumentation. https://paradigm4.atlassian.net/wiki/spaces/ESD/overview.
[29]. Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman. The architecture of scidb. In Proceedings of the 23rd International Conference on Scientific and Statistical Database Management, SSDBM'11, pages 1-16, Berlin, Heidelberg, 2011. Springer-Verlag.
[30]. Emad Soroush, Magdalena Balazinska, Simon Krughoff, and Andrew Connolly. Efficient iterative processing in the scidb parallel array engine. In Proceedings of the 27th International Conference on Scientific and Statistical Database Management, SSDBM '15, pages 39:1-39:6, New York, NY, USA, 2015. ACM.
[31]. Sangchul Kim, Seoung Gook Sohn, Taehoon Kim, Jinseon Yu, Bogyeong Kim, and Bongki Moon. Selective scan for filter operator of scidb. In Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM '16, pages 28:1-28:4, New York, NY, USA, 2016. ACM.
[32]. Jennie Duggan, Olga Papaemmanouil, Leilani Battle, and Michael Stonebraker. Skew-aware join optimization for array databases. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, pages 123-135, New York, NY, USA, 2015. ACM.
[33]. Weijie Zhao, Florin Rusu, Bin Dong, and Kesheng Wu. Similarity join over array data. In Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16, pages 2007-2022, New York, NY, USA, 2016. ACM.

Оптимизация доступа к страницам памяти в системах, использующих программную реализацию глобального страничного кеша

Е.И. Гусев <eugene@levelextra.ru>
Национальный технический университет Украины
«Киевский политехнический институт имени Игоря Сикорского»,
Украина, 03056, г. Киев - 56, проспект Победы, 37

Аннотация. В статье рассматривается способ обработки распределённых страниц в Oracle Real Application Clusters (Oracle RAC) и проводится его сравнение с другими известными способами в контексте сравнения архитектур доступа к страницам. В результате выявления недостатков традиционного способа, применяемого в Oracle RAC, предлагается новый способ доступа, в основе которого лежит введение еще одного состояния страницы – состояния «разгрузки», повышающее эффективность обработки распределённых страниц за счёт снижения количества пересылок между узлами при обработке горячих страниц.

Ключевые слова: Oracle RAC; распределённая страница; shared nothing; shared everything; Global Cache Fusion; СУБД

DOI: 10.15514/ISPRAS-2018-30(1)-11

Для цитирования: Гусев Е.И. Оптимизация доступа к страницам памяти в системах, использующих программную реализацию глобального страничного кеша. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 161-182. DOI: 10.15514/ISPRAS-2018-30(1)-11

1. Введение

Данная работа посвящена изложению одной из основных идей диссертации [1], защищённой в июне 2017 года. Распространение технологий облачных вычислений вывело проблему общего ресурса в контексте масштабирования систем управления базами данных (СУБД) на новый уровень. Облачные технологии для большинства вычислительных задач (включая рендеринг, трансформацию и т. д.) привнесли и снижение стоимости ресурсов и упрощение администрирования за счёт консолидации. Однако необходимо отметить, что для СУБД ими решается только одна задача – повышение эффективности управления, в ущерб стоимости ресурсов, так как требования к

мощности при консолидации возрастают, а возможности использовать для консолидации дешёвые облачные технологии отсутствует. Главная причина здесь – неэффективные технологии обработки общего ресурса, не учитывающие особенности облачных систем и, как следствие, не масштабирующиеся.

Под масштабированием СУБД мы в данном случае будем понимать трансформацию одноузловой системы в многоузловую. Невозможность пропорционального увеличения мощности базы данных (БД) добавлением в облако серверов решается, как правило, консолидацией всех общих ресурсов в единую центральную СУБД и наращиванием её аппаратной мощности. Результатом сложившейся практики является ситуация, когда БД в системах облачных вычислений становится общим ресурсом, используемым всеми вычислительными мощностями.

Другой, довольно часто применяемый приём, – существенное изменение архитектуры приложений для снижения интенсивности обращений к БД часто за счёт снижения контроля за целостностью данных. Яркая иллюстрация такого случая – NoSQL DB [2-7].

Применение же классических способов блокирования при масштабировании облачных (впрочем, и других распределённых) СУБД приводит к значительному увеличению времени обработки и блокирования по сравнению с одноузловыми системами.

Всё это обуславливает кризис в базах данных, выражающийся как в появлении новых архитектур, тесно связанных с приложениями (как, например, NoSQL DB [2-7]) для разгрузки БД от запросов либо облегчения самого запроса, так и новом витке развития дорогостоящих программно-аппаратных комплексов (Oracle Exadata [8], IBM DB2 pureScale [9]). Собственно, невысокие возможности облачных систем консолидировать системы с общим ресурсом и обусловили кризис в развитии СУБД в контексте развития систем облачных вычислений.

Рассматривая модели обслуживания облачных систем, следует отметить, что предоставление СУБД как сервиса клиенту подходит под классификацию «платформа как услуга», или PaaS (platform as a service) [10-12]. Хотя некоторые исследователи относят облачные СУБД к SaaS (software as a service) [13], это, учитывая определение, данное в [11], неверно, поскольку в определении чётко говорится о «несложных покупке и сопровождении программного обеспечения и инфраструктуры, лежащей в основе сервиса».

Среди компаний, поставляющих услуги облачных СУБД, можно выделить такие, как Oracle – на основе собственной СУБД [14], Enterprise DB – на основе PostgreSQL [15], Caspio.com – на основе MSSQL [16], ClearDB [17], SkySQL [18] и другие – на основе mySQL. Достаточно подробная классификация сегодняшних облачных систем приведена в работе [19].

Однако все примеры работающих облачных сервисов ориентированы на невысокие требования обслуживания, а главным плюсом этих сервисов

является снижение расходов за счёт отсутствия затрат на владение сервером. При увеличении требований к обслуживанию компании (те же Oracle, Microsoft, Enterprise DB) предлагают владеть собственными серверами.

Таким образом, все перечисленные решения являются коммерческой нишей, а не возможностью увеличивать собственную производительность (в том числе в реальном времени) за счёт использования облака. Несомненно, о чем свидетельствует успешность этих проектов, такая ниша востребована, однако задачу увеличения производительности за счёт облака, в том числе за счёт частного облака (private cloud [11]) или корпоративного (community cloud [11]), они не решают.

2. Сравнение архитектур СУБД в контексте организации облака

Ключевым фактором при построении облачных систем является выбор архитектуры. На сегодняшний день можно выделить следующие основные используемые архитектуры: shared nothing [20], shared disk [21], архитектуру shared everything, в которой используется программно-реализованный глобальный кеш (Global Cache Fusion) [22] (далее в статье – shared everything), а также решения на основе асинхронной или синхронной репликации, примеры которых будут рассмотрены ниже. Кроме того, несмотря на явную неэффективность с точки зрения производительности, в некоторых случаях используются системы, использующие распределённые транзакции [23], которые в сущности являются частным случаем синхронной репликации. Приведём плюсы и минусы каждой их этих архитектур на примерах существующих реализаций.

Shared Disk. Эта архитектура довольно редко используется в СУБД. Среди примеров можно привести Oracle Parallel Server [24], систему, которая уже много лет не поддерживается. Причина довольна проста – для синхронизации данные надо записать на общий диск и прочитать оттуда. Поскольку скорости обращения к дисковым устройствам (время отклика) ниже, чем скорость обращения к памяти, в последующих версиях корпорация Oracle перешла на архитектуру shared everything. Архитектура shared disk согласно [24] также присуща и облачным решениям на основе IBM DB2, хотя использование технологии pureScale [9] переводит эти системы в категорию shared everything.

Стоит отметить, что для систем с небольшим количеством общих ресурсов архитектура shared disk из-за своей простоты может быть полезна. Ещё один недостаток этой архитектуры – низкая производительность файловых систем, корректно работающих с разделяемым несколькими узлами диском, из-за необходимости обеспечения когерентности кешируемых данных.

Резюмируем: shared disk демонстрирует невысокую пригодность к обработке OLTP-трафика.

Решения на основе репликации. Решения на основе репликации, как синхронной, так и асинхронной, очень широко распространены. Можно упомянуть standby сервера, на которые переносится нагрузка read-only (например, связанная с отчётностью) [25] или так называемая нагрузка read-mostly, содержащая преимущественно чтение, но, кроме того, очень редко выполняющая запись [25]. Кроме того, стоит упомянуть системы выборочной репликации (например, Oracle Golden Gate [26], Quest SharePlex [27]), а также системы, основанные на применении (часто фильтрованных) бинарных журналов mySQL. Одной из наиболее удачных систем на основе применения бинарных журналов является Percona XtraDB Cluster от компании Percona [28].

Однако, несмотря на широкое распространение, все перечисленные системы страдают одним недостатком: на каждый узел необходимо реплицировать изменение со всех остальных узлов. Это накладывает жёсткое ограничение: **суммарный объём изменений всех систем не должен превышать возможности самого медленного узла на применение (накат) изменений**. На практике такие системы редко содержат число узлов, большее 3, именно из-за этого ограничения.

Что касается систем, использующих распределённые транзакции (например, [29]), то они являются частным случаем синхронной репликации. Можно было бы много говорить о неэффективности наиболее распространённых алгоритмов 2PL [30] и 2PC [31], но не они являются слабым местом систем, использующих эти алгоритмы (в контексте функционирования PaaS [11]), а сама архитектура на основе репликации. Возможно, именно поэтому модернизированные алгоритмы блокирования и фиксации транзакции [32-34] не находят широкого применения в контексте роста популярности облачных систем.

Shared nothing. Примером успешных реализаций этих подходов является mysql NDB cluster [35], ElasTras [36], Teradata [37], xkoto Gridscale [38]. Стоит отметить, что на этой архитектуре построено наибольшее количество облачных систем. Однако, как будет показано ниже (разд. 3), shared nothing выдвигает требования к OLTP-трафику и только для определённых классов трафика архитектура эффективна (в контексте миграции в облако).

Оптимизация размещения данных по узлам в зависимости от трафика представляет собой серьёзную задачу, решение которой позволит системам, базирующихся на shared nothing соответствовать требованиям PaaS [11]. Всё это требует более подробного анализа проблем масштабирования shared nothing систем в зависимости от трафика.

Оценивая архитектуру **shared everything**, реализованную в системе Oracle Real Application Clusters (Oracle RAC) [39], необходимо констатировать возможность масштабирования системы без явных ограничений на соотношение чтения и записи по сравнению с архитектурой на основе взаимной репликации, несомненное преимущество в быстродействии по

сравнению с архитектурой shared disk и отсутствие зависимости масштабируемости от трафика, присущее системам shared nothing.

Тем не менее, необходимо указать и на ограничения архитектуры shared everything. В первую очередь – это высокие требования ко времени отклика между узлами. Следствием этого является тот факт, что глобальная сеть в качестве межузлового транспорта может использоваться только при условии существенных ограничений на трафик, которые, как будет показано в этой работе, в конечном счёте накладывают ограничения на интенсивность изменения общих ресурсов. Причём интенсивность изменений ограничена не соотношением изменений в общем трафике при заданном количестве узлов как в архитектуре на основе взаимной репликации, а абсолютная величина интенсивности ограничена непосредственно временем отклика сети. Так же, как и для *shared nothing,* необходим детальный анализ недостатков архитектуры, особенно в сравнении *shared everything* и *shared nothing*.

Отметим также, что система Oracle RAC (взятая в качестве прототипа программно реализованной архитектуры shared everything) имеет широкий спектр внедрений, обеспечивающий распространённость системы и влияющий на качество её сопровождения, что упрощает исследование, и повышает ценность полученных результатов.

3. Основные недостатки архитектуры shared nothing

Основная проблема в реляционной модели для *shared nothing* – это отсутствие всех необходимых данных на узле, производящем обработку. При этом, как правило, система способна распознавать запросы, для которых все необходимые данные расположены на одном узле и в идеале, если весь трафик будет состоять из таких запросов (назовём их «локализируемые», хотя также может использоваться термин одноузловые [40,41]), то можно говорить об успешном масштабировании. Обработка остальных запросов (назовём их «нелокализируемые»), как правило, реализуется всеми узлами параллельно: т. е. идентичный запрос обрабатывается на каждом узле системы с данными этого узла.

Нелокализируемыми часто становятся запросы, обращающиеся к глобальным индексам, запросы с соединением, а также все запросы, не содержащие ключ разбиения на узлы. Более подробно такие случаи рассмотрены в [1]. Стоит отметить и другой вариант обработки «нелокализованного запроса» в случае общего пространства адресации страниц (далее упоминаемый как «альтернативный»), когда обработкой запроса занимается один узел, запрашивающий недостающие страницы с других узлов.

Сразу же стоит указать, что для хранилищ данных (data warehouse) первая стратегия является доминирующей – при обработке запросом больших объёмов данных, целесообразность вовлечения всех узлов для доступа ко всем данным целесообразна, а для OLTP – нет. Например, компания Teradata чётко позиционирует свою распределённую систему как хранилище данных [37].

Логично предположить, что вторая «альтернативная» непараллельная модель обработки будет более пригодной для OLTP-систем shared nothing, однако практика показывает, что в реализации OLTP-систем shared nothing [41-46] в основном рассматриваются трафики, состоящие только из «локализируемых» запросов, а проработка варианта обработки «нелокализируемых» запросов в них отсутствует.

И действительно бывают трафики, для которых равномерность распределения [45-46] или выбор объектов для шардинга (для минимизации соединений таблиц [41]) определяют эффективность масштабирования. Но тогда нужно точно определить, для каких трафиков это возможно. В случае облачной СУБД, это означает необходимость изучения инженерами облачного провайдера взаимосвязей ресурсов в транзакциях перед тем как получить возможность подтверждения возможности размещения СУБД в облаке, что весьма странно для PaaS[11].

Когда это действительно возможно? Например, в ситуации обновления информации об абоненте или формирования выписки по абоненту, если каждому узлу назначается свой список абонентов, непересекающийся со списками абонентов других узлов, как предлагается в работе [44].

Назовём «полностью локализируемым» трафик, обладающий такими свойствами: все запросы включают в себя идентификатор сущности и являются локализируемыми, критерием локализации (назначения узла обработчика) выступает идентификатор сущности, указываемый в запросе. Для примера [44] уже поисковый запрос с фильтром по дате оплаты, трансформируется в n запросов ко всем узлам, где n – количество узлов. С обработкой трафика, состоящего только из «локализируемых» запросов, связана и популярность NoSQL в контексте масштабирования в облаке. Так же, как и случай, рассмотренный в [44], по большей части хорошо масштабируются в облачных системах и трафики социальных сетей. Заметим, что «полностью локализируемый» трафик всегда масштабируем в облаке.

И здесь необходимо обратить внимание на очень интересный факт, меняющий в контексте изучения критериев трансформации СУБД в облачную местами причину и следствие: NoSQL-СУБД могут эффективно обслуживать только «полностью локализируемый» трафик, а значит, базы данных таких систем легко автоматически разбивать на партиции по первичному ключу. Т.е. задача помещения таких задач в облако, удовлетворяющее условиям PaaS, – тривиальна. Это мы и наблюдаем на практике.

С инженеров облачного провайдера автоматически снимается задача анализа пригодности трафика для облака, организованного на архитектуре shared nothing. Критерием пригодности размещения в облаке становится использование NoSQL-СУБД. Выбор разработчиком NoSQL при проектировании приложения автоматически выполняет работу за инженера облачного провайдера.

Нужно отметить и тот факт, что непосредственно на выборку и модификацию данных при обработке OLTP запросов тратиться намного меньше времени, чем на сумму обслуживания структур управления общим ресурсом, сетевого взаимодействия, целостности, безопасности, разбора (parsing) и других задержек запроса. Во многом такое распределение задержек стало причиной популяризации NoSQL-СУБД, хотя наиболее значащим фактором является эффективность масштабирования и финансовая доступность. Во всяком случае, впечатляющих результатов сравнения одноузловых систем MongoDB (или другой NoSQL-СУБД) и Oracle в пользу первой системы автору этой статьи наблюдать не доводилось.

Возвращаясь к «альтернативному» подходу при обработке нелокализируемых запросов трафика, отметим ключевое значение индексов для OLTP-систем, что приводит к малому количеству страниц, используемых каждым запросом, что указывает на перспективность такого подхода. Однако отсутствие реализаций этого подхода и выбор традиционного подхода (плодящего «нелокализируемые» запросы на каждом узле) заставляет серьёзно задуматься о причинах такой ситуации.

Очевидно, что «альтернативный» вариант обработки нелокализируемых данных требует закладывания возможности глобальной адресации страниц внутри облачной системы shared nothing на уровне самой СУБД, а современные реализации архитектуры shared nothing часто проектируются в расчете на одноузловое использование без поддержки такой возможности. Однако, что будет показано ниже, альтернативный вариант может уступать в эффективности и архитектуре shared everything.

Резюмируя недостатки shared nothing, необходимо констатировать неэффективность обработки нелокализируемых запросов в OLTP-системах, что связано с отсутствием проработки «альтернативного» варианта и архитектурной неэффективностью классического подхода. Примеры успешных реализаций кластеров shared nothing при обработке OLTP-трафика **всегда содержат ключ разбиения на партиции в условиях поиска**, что далеко не всегда присуще OLTP-трафику. Можно говорить о невысокой эффективности использования подобных систем в качестве обработчиков OLTP-трафика, если опираться на существующие реализации этой архитектуры.

4. Проблема нарастающей очереди как ключевой недостаток архитектуры shared everything

Для архитектуры *shared everything* [22] ограничением масштабируемости может стать проблема нарастающей очереди [47]. Суть проблемы состоит в том, что при превышении некоторого уровня интенсивности запросов система теряет работоспособность. И эта интенсивность зависит от среднего времени блокирования ресурсов, входящих в транзакции трафика, и взаимосвязей транзакций и ресурсов [1].

Следствием является то, что наращивание количества узлов не позволяет увеличить интенсивность трафика при сохранении его характера. Т.е. масштабируемость ограничена некоторым уровнем интенсивности, который обратно пропорционалена среднему времени обработки «критически горячего» класса транзакций. «Критически горячим» в данном случае называется класс транзакций, для которого при повышении интенсивности трафика величина, обратная показателю интенсивности выполнения транзакций этого класса, превышает среднее время выполнения транзакций этого класса. Рассмотрим этот эффект подробнее.

Для начала определимся с терминологией. Транзакции могут быть разных классов. Будем считать транзакции принадлежащими к одному классу, если они конкурируют за один и тот же набор классов ресурсов, соблюдают один и тот же порядок обработки классов ресурсов, а также поддерживают один и тот же режим выполнения (чтение или изменение). В свою очередь, классом ресурсов будем называть ресурсы, относящиеся к одинаковым логическим сущностям.

Поясним эти громоздкие определений на примере общеизвестного трафика TPC-C. Здесь классы транзакций – это New-Order, Payment, Order-Status, Delivery, Stock-Level, а классы ресурсов – это строки соответствующих таблиц customer, warehouse, district, orders, order_line и т.д. Кроме того, классами ресурсов являются входы индексов на соответствующие таблицы.

Предположим, что мы имеем систему, на которую поступают заявки с интенсивностью v. Пусть в каждой заявке содержится одна транзакция какого-либо класса. У системы имеется множество общих ресурсов. Обработка заявки занимает время t. Если заявка, которая поступила в систему, обратившись к общему ресурсу, обнаруживает, что этот ресурс заблокирован, она ожидает его освобождения. Если более одной заявки ожидает общего ресурса, доступ к ресурсу осуществляется в порядке очереди FIFO.

Пусть время обработки транзакции некоего класса равно t. Для 1-й заявки время выполнения равно t. Если одновременно в обработке две заявки, то время обработки второй заявки – это t плюс время ожидания окончания обработки текущей заявки t_x. Заметим, что t_x меньше t, т.к. вторая заявка поступила после первой. Тогда, если в обработке три заявки (очередь из двух заявок), то время обработки 3-й заявки – t_x+2t. Таким образом, при очереди из n заявок,время обработки $(n+1)$-ой заявки составит t_x+nt.

Несложно показать, что при превышении показателя интенсивности $1/t$ очереди в системе будут нарастать, и система уйдёт в так называемую перегрузку, т. е. перестанет обрабатывать заявки. Более интересным свойством будет и то, что при образовавшейся во время локального пика очереди интенсивность перехода в перегрузку будет значительно ниже $1/t$.

Важно отметить, что ожидание заблокированных ресурсов увеличивают время обработки заявки, а увеличение времени обработки заявки, в свою очередь, увеличивает вероятность блокирования ресурса. Кроме того, для классов

транзакций, которые пересекаются по ресурсам, увеличение времени обработки одного из классов ресурсов может привести к перегрузке. Всё это, несмотря на простоту рассмотренных ситуаций и очевидность выводов, создаёт серьёзные проблемы для программной реализации глобального кеша в реализации архитектуры shared everything в Oracle RAC [22].

Выход из этого положения один – сокращать время обработки заявки. Однако стратегий реализации при страничной организации глобального кеша несколько –: сокращение времени доступа к странице, сокращение времени блокирования ресурсов транзакцией (в том числе за счёт распараллеливания выборки страниц необходимых для транзакции [1]) и сокращение длительности обработки ресурсов транзакцией.

Заметим, что компания Oracle решает проблему первым способом, уменьшая время отклика межузлового канала на основе технологии InfiniBand [48]. Кроме того, с учетом существующих мощностей процессоров и относительной быстроты процессорной обработки OLTP-транзакций по сравнению с сетевыми задержками при этой обработке нецелесообразным является сокращение длительности обработки ресурсов транзакцией. В данной статье мы ограничимся первой стратегией, однако, в отличии от подхода Oracle, сосредоточимся на сокращении среднего времени доступа к странице путем изменения способов доступа.

5. *Основное преимущество архитектуры shared everything*

Продолжим сравнение архитектур shared everything и shared nothing. Поскольку для OLTP-систем «альтернативный» подход выигрышнее, сравнение будем проводить между программной реализацией архитектуры shared everything, применяемой в Oracle RAC, и «альтернативным» вариантом shared nothing. Если рассматривать в качестве критерия эффективности предельную интенсивность, при которой сохраняется способность обрабатывать заявки определённого трафика, то преимущество shared everything можно достичь за счёт сокращения среднего времени доступа к странице. Это обеспечивается кешированием страниц, но требует более сложного механизма обеспечения когерентности. Более подробно алгоритм обработки страниц описан в [1]; ниже приводится его краткое описание.

Ключевой идеей является назначение каждой странице узла мастера, управляющего состояниями копий страницы в локальных кешах каждого узла. Выделяются такие состояния (называемые также блокировками): XCUR (xc, exclusive current) – монопольное текущее, SCUR (sc, shared current) – совместное текущее, CR (consistent read) – целостное чтение. Первое состояние необходимо для внесения изменений, второе используется для чтения последней версии страницы, а третье также используется при чтении и обеспечивает то, что версия страницы соответствует некоторому моменту в прошлом.

Стоит отметить, что реализация в Oracle мультиверсионности за счёт версионности страниц является плюсом при организации многоузловой системы, поскольку CR-копии страниц не требуют синхронизации между узлами. Как следствие мастер-узел отслеживает преимущественно первые два состояния (xc и sc). XCUR-блокировкой в каждый момент времени может владеть только один узел. Блокировкой же SCUR может владеть сколько угодно узлов. Снятие со страницы XCUR-блокировки (переход в SCUR или XCUR на другом узле) приводит к переводу состояния страницы на отдающем узле в CR (через промежуточное состояние PI), а снятие SCUR – к переводу состояния всех страниц, где был SCUR, также в CR. Отметим существование промежуточного состояние PI, необходимого для гарантирования целостности, но к рассматриваемым вопросам сравнения эффективности архитектур не относящееся.

Сама последовательность пересылок, реализованная в Oracle для обеспечения когерентности, достаточно проста: при доступе к странице сначала идёт обращение к мастер-узлу с запросом необходимой блокировки (на страницу XCUR или SCUR), затем мастер шлёт узлу, владеющему в данный момент блокировкой, инструкцию переслать текущую версию страницы запросившему узлу, после чего происходит пересылка самой страницы. После этого узел может начать обработку страницы. Параллельно с этим идёт оповещение мастер-узла о получении страницы (и соответственно блокировки xc или sc на страницу). Таким образом, для доступа к странице мы вынуждены выполнить три пересылки.

Здесь важно отметить два фактора, влияющих на количество пересылок: наличие в кеше узла, выполняющего обработку нужной версии страницы с требуемой блокировкой, и то, что узел, обрабатывающий страницу, оказывается мастером для неё. В обоих случаях уменьшается количество пересылок, и, как следствие, сокращается среднее время доступа к странице.

Необходимо обратить внимание на замечательный факт: для статических данных время доступа к странице сокращается существенно, поскольку через некоторое время все активно используемые страницы на каждом узле будут закешированы (получат блокировку SCUR) всеми узлами. Это является главным плюсом архитектуры, реализованной в Oracle RAC, по сравнению с другими архитектурами.

Но стоит сказать и о главном минусе по сравнению с «альтернативным» подходом архитектуры shared nothing (в первую очередь): три пересылки до начала обработки страницы против двух (без учёта ожидания очереди к странице) в случае незакешированных данных. (В работе [1] показана второстепенность фактора мастер-узла по сравнению с кешированием, поэтому сосредоточимся на эффекте кеширования.) При сравнении, если полагать отсутствие очередей, получается интересная ситуация: если при обращении к горячим страницам одного класса преобладает чтение, то лучшее

среднее время обращения показывает стратегия RAC, а при преобладании операций записи «альтернативный подход».

Но нужно отметить и то, что для именно для «горячих» страниц с увеличением интенсивности будут образовываться очереди. Как будет показано ниже, именно при их образовании создаётся описанный выше эффект нарастающей очереди и видна перспективность подхода, применяемого Oracle. Кроме того, если преобладают операции записи в горячую страницу (что легко отследить на при формировании очереди запросов к мастер-узлу), выполняется простые, но эффективные действия: узлу, запросившему блокировку SCUR, узел, владеющий блокировкой XCUR, не удовлетворяя SCUR, отдаёт CR-версию на самый последний (current) момент времени. Таким образом, интенсивность заявок, приводящая к созданию очереди, сокращается до интенсивности записи в горячую страницу.

Для демонстрации эффективности shared everything при возникновении очередей рассмотрим возможную реализацию когерентности при «альтернативном» подходе. Напомним, что после изменения страницы её нужно вернуть узлу-владельцу. И возникает дилемма: или при одновременном запросе страницы несколькими узлами только один из них сможет вносить изменения, или узел владелец распараллелит изменения в страницы, а при получении их будет выступать арбитром?

Второй подход требует глубокой проработки и рамках этой статьи рассматриваться не будет, а при применении первого подхода мы наблюдаем явное преимущество подхода Oracle, поскольку шаг очереди увеличивается в два раза: сначала пересылка владельцу страницы изменённой копии, а потом её же пересылка следующему в очереди узлу. Обратим внимание, что если узлам дать возможность пересылать страницу друг другу напрямую, то мы получим как раз подход, применяемый Oracle, плюс третью пересылку (от обрабатывающего узла мастеру на фоне обработки страницы) для контроля того, где находится текущая версия страницы.

Следствием увеличения шага очереди в два раза является уменьшение уровня интенсивности, приводящего к переходу в перегрузку, также примерно в два раза. Таким образом, получаем весьма интересные выводы: плюсом архитектуры shared everything, реализованной в RAC, является лучшая устойчивость к перегрузкам при доступе к горячим страницам, а минусом – большее (до полутора раз) время доступа при обращении к таким «холодным» страницам, для которых запись преобладает над чтением (холодным настолько, что к ним практически не образуется очередь).

Обратим внимание и на тот факт, что при увеличении количества узлов системы при том же соотношении числа операций чтения и записи эффективность применения кеша снижается, что также говорит о том, что при доступе к «холодным страницам» альтернативный подход shared nothing лучше. Поэтому актуальна задача разработки такого способа доступа к

страницам, который будет обладать полезными свойствами как первого, так и второго подходов и будет эффективно решать проблему нарастающей очереди.

6. Метод назначения обработчика ресурса

Для повышения эффективности требуется решить задачу распределения между входящими в систему узлами всех имеющихся во всех очередях транзакций и последовательности обработки страниц при условии, что целевой функцией качества размещения является снижение времени пересылок между узлами при сохранении требований ACID. Поскольку эту задачу приходится решать в реальном времени, необходимо использовать простые и эффективные алгоритмы планирования.

Очевидно, что наилучший вариант получится, если все транзакции, связанные в очередь, выполнять на одном узле. Но тогда мы упрёмся в ограничения масштабируемости. Попробуем для горячих страниц локализовать обработку на одном узле. Примером реализации локализации, является метод разгрузки очереди, описанный в [49], с изменениями, описываемыми ниже в данной статье. Результатом применения метода является «разгрузка», устранение очереди к распределённой странице памяти, очереди, возникшей в результате того, что уровень интенсивность к «горячей» странице кратковременно или долговременно превысила граничный уровень интенсивности «перегрузки».

Идея метода достаточно проста: если нарастает очередь, то необходимо снизить количество пересылок между узлами. Для этого выбирается узел, который будет монопольно вносить изменения в «горячую» страницу (в дальнейшем – хостер). При этом на чтение страница (в состоянии CR) отдаётся каждому запросившему её узлу.

Возникает вопрос, как именно хостеру будут передаваться изменения, которые необходимо внести в страницу. Первый вариант простой: узел, не являющийся хостером страницы (в дальнейшем инициатор), сначала нужную ему страницу читает (запрашивает у хостера в состоянии CR на самый последний момент времени), затем формирует запись об изменениях, которые нужно внести в страницу (аналогично тому, как это делается для журнала, обеспечивающего долговечность (durability) ACID-транзакций), и эту запись пересылает хостеру вместе со старыми данными для внесения в страницу. Хостер в случае соответствия старых и новых данных (на уровне строки) вносит изменения и возвращает получившуюся страницу инициатору.

Недостаток такого подхода в том, что в случае «горячей» строки может возникать большое количество исключений, на обработку каждого из которых потребуется две дополнительные пересылки; более того, эти пересылки будут увеличивать ожидание в очереди.

Второй вариант более интересен, но и более сложен: узлу хостеру отправляется кусок кода SQL-запроса выполняющего все действия над этой страницей (возможно, и читающий другие страницы; назовём его SQL-bit).

Первой сложностью является то, что операции изменения транзакции выполняются в разных узлах. Эта сложность обходится, если изменения рассматриваемой транзакции, выполняемые хостером, ещё раз журнализовать в узле-инициаторе. Для обеспечения большей надёжности это может быть и преимуществом.

Вторая сложность – это прерывание выполнения SQL-запроса на инициаторе вызовом удалённой процедуры обработки горячей страницы на хостере. Сложность здесь чисто техническая: требуется интеграция интерпретатора SQL с механизмом распределённой адресации (и кеширования) страниц.

Изложим возможную структуру обмена сообщениями для реализации второго варианта упомянутого метода.

1. При возникновении длинной очереди мастер-узел должен перевести страницу в режим разгрузки, т.е. в некоторый момент времени (когда образовалась очередь) узлы, ожидающие очереди, должны быть уведомлены о переводе страницы в режим разгрузки, и о том, какой узел назначен хостером. Это выдвигает требования к мастер-узлу отслеживать состояние очередей к страницам и выбирать момент для включения режима разгрузки. Кроме того, мастер должен уметь отключать состояние разгрузки для страницы и оповещать соответствующие узлы о таком изменении.
2. На запросы страниц другими инициаторами мастер-узел отвечает, что страницы находятся в состоянии разгрузки с указанием узла-хостера, которому надо пересылать SQL-bit. Инициаторы запоминают, какие страницы находятся в состоянии разгрузки, чтобы в дальнейшем обращаться к ним, минуя мастера.
3. Инициатор формирует и пересылает хостеру свой SQL-bit.
4. Хостер вносит изменения в страницу и в журнал упреждающей записи, а затем пересылает инициатору текущую версию страницы в состоянии CR и свою журнальную запись об изменениях.
5. Инициатор получает страницу в состоянии «как будто он сам внёс в неё изменения и сразу же отправил дальше по запросам других узлов», и заносит журнальную запись хостера в собственный журнал упреждающей записи.

Принципиально заложить в алгоритм возможность мигрировать с узла, вносящего изменения, на другой узел для обеспечения динамической балансировки нагрузки в случае наличия нескольких горячих ресурсов транзакции. т.е. рассылку мастером сообщений о переназначении хостера.

Условимся говорить, что страница находится в состоянии «разгрузки», если к ней начал применяться описанного алгоритм. Выходом из «разгрузки» условимся называть процесс, по результатам которого прекращается применение описанного алгоритма и снова начинает применяться классический способ доступа к странице, применяемый в Oracle RAC.

Важным нюансом в этом алгоритме является минимизация ресурсов на мониторинг возникновения очереди. Кроме того, выход из состояния «разгрузки», оценка целесообразности выхода и приоритизация назначения узла, выполняющего «разгрузку», т.е. всё то, что относится к переходным процессам, связанным с «разгрузкой», выходят за рамки данного исследования.

7. Сокращение количества пересылок при доступе к странице для страниц SCUR mostly

Замечательным свойством мастер-узла при использовании способа доступа, применяемого в Oracle RAC, является возможность отслеживания статистики нахождения в состояниях SCUR, XCUR и «разгрузки». Для страниц SCUR mostly, т.е. таких, которые в основном читаются, но иногда меняются, имеет смысл при удовлетворении блокировки SCUR после XCUR пересылать еще одну копию SCUR-страницы мастер-узлу.

Тогда при последующих SCUR-запросах (т.е. запросах на чтение страницы) вместо трёх мы будем выполнять две пересылки, как и при «альтернативном» подходе shared nothing. Это простое решение позволит сократить среднее время доступа к странице, однако важно отметить, что при преобладании записи над чтением дополнительная пересылка мастер-узлу становится нецелесообразной.

8. Заключение

В заключение статьи обратим внимание на то, что очереди за страницы не являются единственным источником конфликтов в OLTP-системах. Во многих трафиках основной причиной перегрузки является конфликт за ресурсы одного класса, располагаемые на страницах. Основное отличие состоит в том, что блокировка ресурса (на примере Oracle RAC) обеспечивается до завершения транзакции, а блокировка страницы (и, как следствие, отправка её следующему в очереди узлу) прекращается после внесения изменений.

Несмотря на взаимосвязанность задержек, порождаемых этими конфликтами, для каждого из них имеется своя стратегию разрешения, и данная статья вообще не касается способов организации блокировкия ресурсов и разрешения конфликтов транзакций. Это безусловно является актуальной темой для исследований, и идеи работы [1] в этом направлении необходимо развивать.

Также стоит отметить, что рассматриваемый в статье «альтернативный подход», можно классифицировать не как разновидность *shared nothing*, а как реализацию *shared everything*, отличную от применяемой в Oracle RAC реализации общей памяти на основе Global Cache Fusion (программно реализованного глобального кеша).

Подчеркнем важную роль третьего состояния страницы (разгрузка) для обеспечения страничной организации в любых распределённых системах, а не

только в СУБД. Наконец, еще раз заметим, что эффект от применения разгрузки зависит от трафика, и наибольший эффект происходит в случае коротких транзакций трафика.

Список литературы

[1]. Гусев Е.И. Способы организации совместного доступа к распределённым страницам памяти в системах облачных вычислений. Диссертация на соискание учёной степени кандидата технических наук. НТУ "КПИ" им. Сикорского, Киев, 2017. 156 страниц.

[2]. Кузнецов С.Д., Посконин А.В. Системы управления данными категории NoSQL. Программирование, том 40, № 6, 2014, стр. 34-47

[3]. Berndt D.J., Lasa R., McCart J. SiteWit Corporation: SQL or NoSQL? That is the Question!. Journal of Information Technology Education: Discussion Cases, Volume 6, 2017, pp. 04 University of South Florida, 2012, p. 14-15. DOI:10.28945/3920.

[4]. Бурмистров А.В., Белов Ю.С. Недостатки реляционных баз данных. Электронный журнал: наука, техника и образование, 2015, № 3. стр. 25-34.

[5]. Селезнев К. От SQL к NoSQL и обратно. Открытые системы. СУБД, 2012. No 2. Доступно по ссылке: https://www.osp.ru/os/2012/02/13014127/, 10.01.2018.

[6]. Padhy R.P., Patra R.M., Satapathy S.C. RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's. International Journal of Advanced Engineering Sciences and Technologies, 2011, vol. 11 (1), pp. 15-30.

[7]. Мухина Ю. Р. Обзор NoSQL решений управления данными. Управление в современных системах. 2013. No 1. стр.68-73.

[8]. Weiss R. Technical Overview of the Oracle Exadata Database Machine and Exadata Storage Server. ORACLE white paper, June 2012. Доступно по ссылке: http://www.oracle.com/technetwork/database/exadata/exadata-technical-whitepaper-134575.pdf, 10.01.2018.

[9]. IBM Documentation: DB2 pureScale Feature road map (online). Доступно по ссылке: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.licensing.doc/doc/c0056030.html, 10.01.2018.

[10]. Rackspace Support "Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS," Rackspace, October 22, 2013. Доступно по ссылке: https://support.rackspace.com/how-to/understanding-the-cloud-computing-stack-saas-paas-iaas/ , 10.01.2018.

[11]. Mell P., Grance T. The NIST Definition of Cloud Computing. National Institute of Science and Technology. Special Publication 800-145,October 25, 2011. Доступно по ссылке https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf, 10.01.2018.

[12]. Butler B. PaaS Primer: What is platform as a service and why does it matter?. Network World, February 11, 2013. Доступно по ссылке: https://www.networkworld.com/article/2163430/cloud-computing/paas-primer-what-is-platform-as-a-service-and-why-does-it-matter-.html, 10.01.2018.

[13]. Kadam M, Jidge P., Tambe S., Tayade E. Cloud Service Based On Database Management System , Int. Journal of Engineering Research and Applications. ISSN:2248-9622, Vol. 4, Issue 1(Version 3), January 2014, pp. 303-306.

[14]. Oracle Infrastructure and Platform Cloud Services Security. Oracle white paper, November 2016. Доступно по ссылке:

https://cloud.oracle.com/opc/iaas/whitepapers/Oracle_Cloud_Security_Whitepaper.pdf, 10.01.2018.

[15]. Achieving HIPAA compliance with Postgers Plus Cloud Database. EnterpriseDB white paper. EnterpriseDB Corporation, 2015. https://www.enterprisedb.com/hipaa-compliance-postgres-plus-cloud-database 10.01.2018.

[16]. Online Database Software. Custom Database Applications. Caspio. Доступно по ссылке: https://www.caspio.com/, 10.01.2018.

[17]. ClearDB - The Ultra Reliable, Geo Distributed Data Services Platform. Доступно по ссылке: http://w2.cleardb.net/ , 10.01.2018.

[18]. SkySQL Makes Highly Available Databases Easy, with MariaDB Enterprise | MariaDB Доступно по ссылке: https://mariadb.com/about-us/newsroom/press-releases/skysql-makes-highly-available-databases-easy-mariadb-enterprise, 10.01.2018.

[19]. Николаенко А. Год облачных СУБД. Открытые системы.СУБД, 2013, No 9. Доступно по ссылке: https://www.osp.ru/os/2013/09/13038286/ , 10.01.2018.

[20]. Shared disk architecture – Wikipedia. Доступно по ссылке: https://en.wikipedia.org/wiki/Shared_disk_architecture, 10.01.2018.

[21]. Shared-nothing architecture – Wikipedia. Доступно по ссылке: https://en.wikipedia.org/wiki/Shared_nothing_architecture, 10.01.2018.

[22]. Parallel Execution with Oracle Database 12c Fundamentals. Oracle White Paper, December 2014. Доступно по ссылке http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-parallel-execution-fundamentals-133639.pdf, 10.10.2018.

[23]. Taniar D., Leung C. H. C., Rahayu W., Goel S. High Performance Parallel Database Processing and Grid Databases. Ch. 10. Wiley Publishing, 2008, isbn: 9780470107621, pp. 289-320.

[24]. Bauer M. Oracle8i Parallel Server Concepts, Release 2 (8.1.6), Part No. A76968-01, December 1999. Доступно по ссылке: https://docs.oracle.com/cd/A87860_01/doc/server.817/a76965.pdf, 10.01.2018.

[25]. Oracle Active Data Guard, Real-Time Data Protection and availability. Oracle White Paper, October 2015. Доступно по ссылке: http://www.oracle.com/technetwork/database/availability/active-data-guard-wp-12c-1896127.pdf, 10.01.2018.

[26]. Oracle GoldenGate 12c: Real-Time Access to Real-Time Information. Oracle White Paper, March 2015. Доступно по ссылке: http://www.oracle.com/us/products/middleware/data-integration/oracle-goldengate-realtime-access-2031152.pdf, 10.01.2018.

[27]. Chu T. Top Five Reasons to Choose SharePlex® Over Oracle GoldenGate. Quest Software, November, 2011. Доступно по ссылке: http://www.dlt.com/sites/default/files/Quest-Shareplex-Whitepaper.pdf, 10.01.2018.

[28]. Percona XtraDB Cluster Release5.7.17-29.20 Operations Manual Доступно по ссылке: https://learn.percona.com/download-percona-xtradb-cluster-5-7-manual, 10.01.2018.

[29]. Xiai Yan, Jinmin Yang, Qiang Fan. An Improved Two-phase Commit Protocol Adapted to the Distributed Real-time Transactions. Przegląd Elektrotechniczny (Electrical Review), ISSN 0033-2097, R. 88 NR 5b/2012, pp. 27-30.

[30]. Bernstein P. A., Hadzilacos Goodman N.: Concurrency Control and Recovery in Database Systems, Addison Wesley Publishing Company, ISBN 0-201-10715-5. 1987 pp. 49-53.

[31]. Open Group Standard DRDA, Version 5, Volume 3: Distributed Data Management (DDM) Architecture // ISBN: 1-931624-93-3 Document Number: C114. pp. 831-832.

[32]. Gray J., Lamport L. Consensus on Transaction Commit. Microsoft Research. 1 January 2004 revised 19 April 2004, 8 September 2005. Доступно по ссылке: https://www.microsoft.com/en-us/research/publication/consensus-on-transaction-commit/, 10.01.2018.

[33]. Mahmoud H. A., Arora V., Nawab F., Agrawal D., El Abbadi A. Maat: Effective and scalable coordination of distributed transactions in the cloud. Proceedings of the VLDB Endowment, Volume 7, No 5. January 2014, pp. 329–340.

[34]. Keidar I. Dolev D. Increasing the Resilience of Distributed and Replicated Database Systems.Journal of Computer and System Sciences (JCSS). December 1998, Issue 57 (3), pp. 309–324. DOI:10.1006/jcss.1998.1566

[35]. MySQL :: MySQL 5.7 Reference Manual :: 21 MySQL NDB Cluster 7.5 and NDB Cluster 7.6. Доступно по ссылке: https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html, 10.01.2018.

[36]. Das S., Agarwal S., Agrawal D., El Abbadi A. ElasTraS: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud. UCSB Computer Science Technical Report 2010-04, pp. 1-14.

[37]. The Teradata Scalability Story, A Teradata White Paper, EB-3031 0701, 2001, NCR Corporation. Available at: http://www3.cs.stonybrook.edu/~sas/courses/cse532/fall01/teradata.pdf, 10.01.2018.

[38]. Gridscale® Database Virtualization Software. Technical whitepaper, xkoto, Inc. Item: GS-WP-EN-20080930. 2008. Доступно по ссылке: http://www.tech-21.com.hk/download/Gridscale_Technical_White_Paper.pdf , 10.01.2018.

[39]. Michalewicz M., Clouse B., McHugh J. Oracle Real Application Clusters (RAC). Oracle White Paper. June 2013. Доступно по ссылке: http://www.oracle.com/technetwork/database/options/clustering/rac-wp-12c-1896129.pdf, 10.01.2018.

[40]. Кузнецов С.Д. Транзакционные параллельные СУБД: новая волна. Труды Института системного программирования, т. 20, М., ИСП РАН, 2011, стр. 189-251

[41]. Stonebraker M., Madden S., Abadi D.J., Harizopoulos S., Hachem N., Helland P. The End of an Architectural Era (It's Time for a Complete Rewrite). Proceedings of VLDB, 2007, Vienna, Austria, pp. 1150-1160.

[42]. Бойченко А.В., Рогожин Д. К., Корнеев Д. Г. Алгоритм динамического масштабирования реляционных баз данных в облачных средах. Экономика, Статистика и Информатика No 6 (2), 2014. стр. 461-465.

[43]. Чистов В.А., Лукьянченко А.В. Автоматизация масштабирования высоконагруженных баз данных MySQL. Современные наукоёмкие технологии, 2016, 6-2, стр. 315-319.

[44]. Горобец В.В. Математические модели и алгоритмы оптимизации размещенения данных транзакционных систем. Диссертация на соискание ученой степени кандидата технических наук. Новочеркасск, Южно-Российский государственный политехнический университет (НПИ) имени М. И. Платова, 2015. 210 страниц.

[45]. Зернов А.С., Ожиганов А.А. Горизонтальное масштабирование базы данных с использованием консистентного хеширования. Известия высших учебных заведений. Приборостроение. 2017. Т. 60. № 3. стр. 234-238.

[46]. Caio H. Costa, João Vianney, Paulo Maia, Francisco Carlos M. B. Oliveira. Sharding by Hash Partitioning - A Database Scalability Pattern to Achieve Evenly Sharded Database Clusters. 17th International Conference on Enterprise Information Systems (ICEIS 2015), At Barcelona, Spain. DOI: 10.5220/0005376203130320.

[47]. Гусев Е.И. Исследование области применения неблокирующего алгоритма фиксации распределённых транзакций. Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. 2012. Випуск 57. стр. 76-80.
[48]. InfiniBand – Wikipedia. Доступно по ссылке: https://en.wikipedia.org/wiki/InfiniBand, 10.01.2018
[49]. Гусев Е.И. Оптимизация доступа к распределённым страницам памяти в cloud computing системах основанных на shared everything архитектуре используя метод разгрузки очередей. Проблеми інформатизації та управління. 2015. – Том 4, № 52. стр. 17-21.

Optimizing access to memory pages in software-implemented global page cache systems.

E.I. Gusev <eugene@levelextra.ru>
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute",
37, Prosp. Peremohy, Kyiv, Ukraine, 03056.

Abstract. This article is based on a thesis “Techniques of organizations shared access to distributed memory pages in cloud computing systems”, defensed in Igor Sikorsky Kyiv Polytechnic Institute in 2017. The paper describes distributed pages processing in Oracle Real Application Clusters (Oracle RAC) and compares it with other known processing methods. The comparison comprises analysis of different architectures (including shared nothing, shared disk and “based on a replication” architecture) in the context of SQL query processing and asserts reasonableness of distributed pages approach (also known as Global Cache Fusion) choice for cloud DBMS. As a result researching the Global Cache Fusion approach there was revealed main drawback of Oracle RAC systems – “increasing queue problem”: impossibility process requests after intensity of requests exceeds threshold intensity, which is inversely proportional packet sent time between nodes. To eliminate “increasing queue problem” during distributed page access the new access method is proposed, which is based on the initiation of one more page state - the "unloading" state, which increases the efficiency of processing distributed pages by reducing the number of transfers between nodes during hot page processing. The considered method can be used not only in cloud DBMS but also in other cloud systems in a case if they use page-organized distributed memory architecture.

Keywords: Oracle RAC; distributed page; shared nothing; shared everything; Global Cache Fusion; RDBMS.

DOI: 10.15514/ISPRAS-2018-30(1)-11

For citation: Gusev E.I. Optimizing access to memory pages in software-implemented global page cache systems. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 161-182 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-11

References

[1]. Gusev E.I. Techniques of organizations shared access to distributed memory pages in cloud computing systems. Thesis for a Candidate of Technical Sciences degree, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kiev, 2017. 156 pages.

[2]. Kuznetsov S. D., Poskonin A. V. NoSQL data management systems. Programming and Computer Software. November 2014, Volume 40, Issue 6, pp. 323-332. DOI: 10.1134/S0361768814060152.

[3]. Berndt D.J., Lasa R., McCart J. SiteWit Corporation: SQL or NoSQL? That is the Question!. Journal of Information Technology Education: Discussion Cases, Volume 6, 2017, pp. 04 University of South Florida, 2012, p. 14-15. DOI:10.28945/3920.

[4]. Burmistrov A.V. Belov Y.S. Disadvantages of relational databases. [Online journal: Science, technology and education]. Elektronny zhurnal: Nauka, tekhnika i obrazovanie]. ISSN 2312-8267(Print), ISSN 2413-5801(Online), 2015 No. 3. pp. 25-34 (in Russian).

[5]. Seleznev K. From SQL to NoSQL and back again. Open Systems. DBMS, ISSN 1028-7493, 2012. No 2 (in Russian). Available at: https://www.osp.ru/os/2012/02/13014127/, 10.01.2018.

[6]. Padhy R.P., Patra R.M., Satapathy S.C. RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's. International Journal of Advanced Engineering Sciences and Technologies, 2011, vol. 11 (1), pp. 15-30.

[7]. Mukhina Y.R. NoSQL solutions of data management rewiew. Upravlenie v sovremennyh sistemah, ISSN 2311-1313, 2013. No 1. pp. 68-73 (in Russian).

[8]. Weiss R. Technical Overview of the Oracle Exadata Database Machine and Exadata Storage Server. ORACLE white paper, June 2012. Available at: http://www.oracle.com/technetwork/database/exadata/exadata-technical-whitepaper-134575.pdf, 10.01.2018.

[9]. IBM Documentation: DB2 pureScale Feature road map (online). Available at: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.db2.luw.licensing.doc/doc/c0056030.html, 10.01.2018.

[10]. Rackspace Support "Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS," Rackspace, October 22, 2013. Available at: https://support.rackspace.com/how-to/understanding-the-cloud-computing-stack-saas-paas-iaas/ , 10.01.2018.

[11]. Mell P., Grance T. The NIST Definition of Cloud Computing. National Institute of Science and Technology. Special Publication 800-145,October 25, 2011. Доступо по ссылке https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf, 10.01.2018.

[12]. Butler B. PaaS Primer: What is platform as a service and why does it matter?. Network World, February 11, 2013. Available at: https://www.networkworld.com/article/2163430/cloud-computing/paas-primer-what-is-platform-as-a-service-and-why-does-it-matter-.html, 10.01.2018.

[13]. Kadam M, Jidge P., Tambe S., Tayade E. Cloud Service Based On Database Management System , Int. Journal of Engineering Research and Applications. ISSN:2248-9622, Vol. 4, Issue 1(Version 3), January 2014, pp. 303-306

[14]. Oracle Infrastructure and Platform Cloud Services Security. Oracle white paper, November 2016. Available at: https://cloud.oracle.com/opc/iaas/whitepapers/Oracle_Cloud_Security_Whitepaper.pdf, 10.01.2018.

[15]. Achieving HIPAA compliance with Postgers Plus Cloud Database. EnterpriseDB white paper. EnterpriseDB Corporation, 2015. https://www.enterprisedb.com/hipaa-compliance-postgres-plus-cloud-database 10.01.2018.
[16]. Online Database Software. Custom Database Applications. Caspio. Available at: https://www.caspio.com/, 10.01.2018.
[17]. ClearDB - The Ultra Reliable, Geo Distributed Data Services Platform. Available at: http://w2.cleardb.net/ , 10.01.2018.
[18]. SkySQL Makes Highly Available Databases Easy, with MariaDB Enterprise | MariaDB Available at: https://mariadb.com/about-us/newsroom/press-releases/skysql-makes-highly-available-databases-easy-mariadb-enterprise, 10.01.2018.
[19]. Nikolayenko A. Year of cloud DBMS. Open Systems. DBMS, ISSN 1028-7493, 2013, No 9 (in Russian), Available at: https://www.osp.ru/os/2013/09/13038286/ , 10.01.2018.
[20]. Shared disk architecture – Wikipedia. Available at: https://en.wikipedia.org/wiki/Shared_disk_architecture, 10.01.2018.
[21]. Shared-nothing architecture – Wikipedia. Available at: https://en.wikipedia.org/wiki/Shared_nothing_architecture, 10.01.2018.
[22]. Parallel Execution with Oracle Database 12c Fundamentals. Oracle White Paper, December 2014. Available at http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-parallel-execution-fundamentals-133639.pdf, 10.10.2018.
[23]. Taniar D., Leung C. H. C., Rahayu W., Goel S. High Performance Parallel Database Processing and Grid Databases. Ch. 10. Wiley Publishing, 2008, isbn: 9780470107621, pp. 289-320
[24]. Bauer M. Oracle8i Parallel Server Concepts, Release 2 (8.1.6), part No. A76968-01, December 1999. Available at: https://docs.oracle.com/cd/A87860_01/doc/server.817/a76965.pdf, 10.01.2018.
[25]. Oracle Active Data Guard, Real-Time Data Protection and availability. Oracle White Paper, October 2015. Available at: http://www.oracle.com/technetwork/database/availability/active-data-guard-wp-12c-1896127.pdf, 10.01.2018.
[26]. Oracle GoldenGate 12c: Real-Time Access to Real-Time Information. Oracle White Paper, March 2015. Available at: http://www.oracle.com/us/products/middleware/data-integration/oracle-goldengate-realtime-access-2031152.pdf, 10.01.2018.
[27]. Chu T. Top Five Reasons to Choose SharePlex® Over Oracle GoldenGate. Quest Software, November, 2011. Available at: http://www.dlt.com/sites/default/files/Quest-Shareplex-Whitepaper.pdf, 10.01.2018.
[28]. Percona XtraDB Cluster Release5.7.17-29.20 Operations Manual Available at: https://learn.percona.com/download-percona-xtradb-cluster-5-7-manual, 10.01.2018.
[29]. Xiai Yan, Jinmin Yang, Qiang Fan. An Improved Two-phase Commit Protocol Adapted to the Distributed Real-time Transactions. Przegląd Elektrotechniczny (Electrical Review), ISSN 0033-2097, R. 88 NR 5b/2012, pp. 27-30
[30]. Bernstein P. A., Hadzilacos Goodman N.: Concurrency Control and Recovery in Database Systems, Addison Wesley Publishing Company, ISBN 0-201-10715-5. 1987 pp. 49-53.
[31]. Open Group Standard DRDA, Version 5, Volume 3: Distributed Data Management (DDM) Architecture // ISBN: 1-931624-93-3 Document Number: C114. pp. 831-832
[32]. Gray J., Lamport L. Consensus on Transaction Commit. Microsoft Research. 1 January 2004 revised 19 April 2004, 8 September 2005. Available at:

https://www.microsoft.com/en-us/research/publication/consensus-on-transaction-commit/, 10.01.2018.

[33]. Mahmoud H. A., Arora V., Nawab F., Agrawal D., El Abbadi A. Maat: Effective and scalable coordination of distributed transactions in the cloud. Proceedings of the VLDB Endowment, Volume 7, No 5. January 2014, pp. 329–340

[34]. Keidar I. Dolev D. Increasing the Resilience of Distributed and Replicated Database Systems.Journal of Computer and System Sciences (JCSS). December 1998, Issue 57 (3), pp. 309–324. DOI:10.1006/jcss.1998.1566

[35]. MySQL :: MySQL 5.7 Reference Manual :: 21 MySQL NDB Cluster 7.5 and NDB Cluster 7.6. Available at: https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html, 10.01.2018.

[36]. Das S., Agarwal S., Agrawal D., El Abbadi A. ElasTraS: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud. UCSB Computer Science Technical Report 2010-04, pp. 1-14.

[37]. The Teradata Scalability Story, A Teradata White Paper, EB-3031 0701, 2001, NCR Corporation. Available at: http://www3.cs.stonybrook.edu/~sas/courses/cse532/fall01/teradata.pdf, 10.01.2018.

[38]. Gridscale® Database Virtualization Software. Technical whitepaper, xkoto, Inc. Item: GS-WP-EN-20080930. 2008. Available at: http://www.tech-21.com.hk/download/Gridscale_Technical_White_Paper.pdf , 10.01.2018.

[39]. Michalewicz M., Clouse B., McHugh J. Oracle Real Application Clusters (RAC). Oracle White Paper. June 2013. Available at: http://www.oracle.com/technetwork/database/options/clustering/rac-wp-12c-1896129.pdf, 10.01.2018.

[40]. Kuznetsov S. D. Transactional Massive-Parallel DBMSs: A New Wave. Trudy ISP RAN/Proc. ISP RAS, 2011, vol. 20, pp. 189-251.

[41]. Stonebraker M., Madden S., Abadi D.J., Harizopoulos S., Hachem N., Helland P. The End of an Architectural Era (It's Time for a Complete Rewrite). Proceedings of VLDB, 2007, Vienna, Austria, pp. 1150-1160.

[42]. Boichenko A.V., Rogojin D.K., Korneev D.G. Algorithm for dynamic scaling relational database in clouds. Ekonomika, statistika I informatika. [Economy, statistics and informatics], ISSN 2500-3925 (Print), 2014. No 6 (2), pp. 461-465 (in Russian).

[43]. Chistov V.A., Lukyanchenko A.V. Automation of scaling high loaded MySQL databases. Sovremennye naukoyomkie tekhnologii [Modern high-tech], ISSN 1812-7320, 2016, 6-2, pp. 315-319 (in Russian).

[44]. Gorobets V.V. Mathematical models and algorithms for optimizing the distribution of transaction system data. Thesis for a Candidate of Technical Sciences degree, Platov South-Russian State Polytechnic University (NPI), Novocherkassk, 2015. 210 pages (in Russian).

[45]. Zernov A.S., Ozhiganov A.A. Horizontal scaling of database using consuistent hashing. Izvestiya vysshih uchebnyh zavedeniy. Priborostroenie. [higher educational institutions news. Instrumentation], ISSN 2500-0381, 2017. vol. 60. № 3. pp. 234-238 (in Russian).

[46]. Caio H. Costa, João Vianney, Paulo Maia, Francisco Carlos M. B. Oliveira. Sharding by Hash Partitioning - A Database Scalability Pattern to Achieve Evenly Sharded Database Clusters. 17th International Conference on Enterprise Information Systems (ICEIS 2015), At Barcelona, Spain. DOI: 10.5220/0005376203130320

[47]. Gusev E.I. Implementation sphere researching of distributed transaction nonblocking commit algorithm. Visnyk NTUU "KPI" Informatics, operation and computer science, ISSN 0135-1729, 2012, Issue 57, pp. 76-80 (in Russian).

[48]. InfiniBand – Wikipedia. Available at: https://en.wikipedia.org/wiki/InfiniBand, 10.01.2018

[49]. Gusev E.I. Optimization of access to distributed pages in a cloud computing systems based on shared everything architecture using unload queue method. Problemy informatyzatsii ta upravlinnia. [Problems of informatization and management], ISSN 2073-4751, 2015. vol. 4, No 52. pp.17-21 (in Russian).

Численное исследование влияния формы торцов колеблющихся пластин на гидродинамическое сопротивление в диапазоне больших амплитуд колебания

А.Н. Нуриев <nuriev_an@mail.ru>
А.М. Камалутдинов <islamui@hotmail.com>
О.Н. Зайцева <olga_fdpi@mail.ru>
Казанский (Приволжский) федеральный университет,
420008, Россия, Казань, Кремлёвская, д. 18

Аннотация. В работе проводится численное моделирование обтекания гармонически осциллирующих тонких пластин с разной формой торцов в диапазоне чисел Рейнольдса 10<Re<600. Для описания движения жидкости решается полная нестационарная система уравнений Навье-Стокса. Задача рассматривается в плоской постановке. Численная модель реализуется на базе открытой платформы OpenFOAM. Рассматривается вопрос о влиянии формы торцов на гидродинамическое сопротивление в режимах с интенсивным вихреобразованием. Проводится анализ структуры течения, распределения давления по поверхности пластин, выполняется расчет коэффициентов сопротивления для разных амплитуд колебания. Результаты исследования показывают, что изменение формы торцов приводит к смещению точек отрыва вихрей с пластины. Это сказывается на распределении давления по поверхности пластины. Так у усеченных пластин разница между измеренным давлением на правой и левой сторонах пластины в окрестности торцов оказывается меньше, чем у прямоугольных. Это, в конечном счете, приводит к снижению результирующего аэродинамического сопротивления усеченных пластин. В рассматриваемом диапазоне параметров значения коэффициента сопротивления для прямоугольной пластины лежат в среднем на 14% выше. Полученные результаты хорошо объясняют большой разброс данных между проведенными ранее экспериментальными и численными исследованиями, так как практически во всех численных исследованиях сечение пластины принимают прямоугольным. В тоже время в экспериментах обычно используются образцы с усеченными торцами. Соответствующие данные для каждого из этих типов пластин хорошо согласуются с полученными в рамках данного исследования результатами.

Ключевые слова: вязкая жидкость; тонкие пластины; гармонические осцилляции; коэффициент гидродинамического сопротивления; форма торцов, численное моделирование

DOI: 10.15514/ISPRAS-2018-30(1)-12

Для цитирования: Нуриев А.Н., Камалутдинов А.М., Зайцева О.Н. Численное исследование влияния формы торцов колеблющихся пластин на гидродинамическое сопротивление в диапазоне больших амплитуд колебания. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 183-194. DOI: 10.15514/ISPRAS-2018-30(1)-12

1. Введение

Первые исследования взаимодействия осциллирующих пластин с неподвижной вязкой жидкостью проводились еще в 60 - 70-х года прошлого века в связи с изучением воздействия волн на элементы конструкций при морском строительстве. Реализованные в этот период гидродинамические эксперименты [1], [2] и созданные на базе них модели вихревого взаимодействия [3], позволили определить основные параметры задачи, порядок гидродинамических сил и структуры течений, создаваемых в жидкости осциллирующими пластинами, для области больших амплитуд колебания. Для случая малых амплитуд колебания в это же время была сформулирована [4] обобщенная на случай тонких пластин линеаризованная теория Стокса. Долгое время эти результаты составляли основную базу по оценке воздействия среды на осциллирующие пластины. Однако, развитие новых прикладных областей, таких как атомная микроскопия и робототехника, разработка наноэлектрических устройств и пьезоэлектрических микро-вентиляторов, создание новых методов оценки демпфирующих свойств материалов [5]-[7], дало толчок к дальнейшему развитию и расширению исследования данной проблематики.

В современный период исследования задача взаимодействия осциллирующих пластин с неподвижной вязкой жидкостью часто рассматривается в контексте проблемы определения аэродинамического демпфирования свободных или вынужденных колебаний удлиненных консольно-закрепленных балок прямоугольного поперечного сечения. В предположении, что длина балок существенно превышает их ширину и толщину, взаимодействие балок с воздухом рассматривается в рамках квазидвумерной гипотезы, согласно которой аэродинамические силы определяются в каждом сечении балки путем изучения плоского движения газа, вызванного гармоническими осцилляциями тонкой жесткой пластины. Такой подход лег в основу многочисленных экспериментальных и численных исследований [5]-[11], проводимых в последнем десятилетии.

Экспериментальные данные (см., например, [1], [6]) в целом указывают на широкие границы применимости двумерных (или квазидвумерных в случае балки) моделей обтекания пластин ($\mathrm{Re} < 1000$). В то же время результаты численного моделирования при относительно высоких числах Рейнольдса ($\mathrm{Re} : 1000$) [5], [10], дают завышенные оценки (на 20-30%) аэродинамического сопротивления по сравнению с данными экспериментов.

Результаты настоящего исследования показывают, что в корне этих проблем лежат упрощенные предположения (используемые в численных моделях) о геометрии сечения пластины.

Практически во всех численных исследованиях сечение принимают бесконечно тонким (см. например [10]) или прямоугольным (см. например [5], [8]). В тоже время в экспериментах обычно используются образцы (конечных размеров) с усеченными торцами [1], [6]. Как показывают настоящие расчеты эта, казалась бы несущественная, деталь имеет большое значение для режимов с интенсивным вихреобразованием, где форма торцов определят точку отрыва вихрей.

2. Постановка задачи.

Пластина шириной b и толщиной h совершает колебания в горизонтальной плоскости в вязкой несжимаемой жидкости по гармоническому закону

$$s = A\sin\omega t,$$

где s – горизонтальные перемещения, A, ω амплитуда и частота колебаний соответственно.

При нормировке пространственных координат, времени и скорости на $b/2$, $b/2/U_0$, $-U_0$ соответственно (где U_0 – амплитуда скорости колебаний), система уравнений движения жидкости записывается в декартовой системе координат как

$$\begin{cases} \dfrac{\partial U}{\partial t} + U\cdot\nabla U = -\nabla p + \dfrac{2}{\mathrm{Re}}\Delta U \\ \nabla\cdot U = 0 \end{cases}, \qquad (1)$$

где $U = U(u,v)$ – безразмерная скорость, p – безразмерное давление, Re – число Рейнольдса, ν – кинематическая вязкость жидкости. Уравнение движения цилиндра в нормированных переменных записываются в виде

$$s = \frac{KC}{\pi}\sin(\pi t / KC). \qquad (2)$$

Здесь $KC = 2U_0\pi / b\omega = 2\pi A / b$ – второй безразмерный управляющий параметр задачи – число Кейлигана-Карпентера или безразмерная амплитуда колебаний.

Безразмерная толщина пластины *h/b* является третьим безразмерным параметром задачи, в настоящем исследовании считается постоянной и равной 1/10 (как и в экспериментальной работе [6]).

Таким образом, комплекс из двух параметров (Re, KC) полностью определяет течение жидкости около осциллирующей пластины. Иногда удобно использовать также их отношение

$$\beta = \frac{\mathrm{Re}}{KC} = \frac{b^2\omega}{2\pi\nu},$$

играющее роль колебательного числа Рейнольдса.

Для численного решения задачи (1), (2) осуществляется переход в подвижную систему координат связанную с пластиной. В этом случае для сохранения системы движения в форме (1) в новой неинерциальной системе координат определяется новое давление:

$$p = p_0 + x\sin(\pi t / KC)\pi / KC.$$

Здесь первое слагаемое p_0 – давление в неподвижной системе координат, а второе вклад от инерциальных составляющих.

На границе пластины в новой системе координат задаются условия прилипания:

$$u = v = 0.$$

На бесконечности изменение скорости определяется по следующему гармоническому закону:

$$u = \cos(\pi t / KC), \quad v = 0.$$

Вычисление гидродинамических сил, действующих на пластину со стороны жидкости, в представленной безразмерной постановке проводится по формуле:

$$F = \int_S p\mathrm{n}ds - \int_S \bar{\sigma}\cdot \mathrm{n}ds,$$

где $\bar{\sigma}$ – тензор вязких напряжений, S – поверхность цилиндра, n – внутренняя нормаль к поверхности пластины.

Полученный таким образом вектор силы F можно разложить на вертикальную составляющую F_y – подъемную силу, и горизонтальную F_x, стоящую и из сил сопротивления и инерциальных сил. Инерциальные составляющие возникают вследствие ускорения жидкости и состоят из двух частей: силы инерции присоединенных масс, возникающей из-за локального ускорения вблизи пластины и силы Фруда-Крылова, которая связанна с градиентом давления, созданным в жидкости для моделирования осциллирующего потока.

Для аппроксимации влияния инерциальных и вязких составляющих горизонтальной силы используется формула Морисона [12]:

$$F_x = \pi C_m \frac{du_\infty}{dt} + C_d \left|u_\infty\right| u_\infty,$$

где u_∞ – скорость жидкости на бесконечности, C_m – коэффициент инерциальных сил, C_d – коэффициент сопротивления.

3. Численное решение.

Численное решение задачи проводится в пакете OpenFOAM [13] на основе расчетной схемы описанной в [14]. Течение моделируется в прямоугольной области размерами 60×40, в центре которой располагается исследуемая пластина. В используемой декартовой системе координат (Ox, Oy), стороны расчетной области параллельны основным осям, колебания происходят вдоль оси Ox.

Для дискретизации расчетной области используются блочные сетки двух типов. Первый тип – структурированные сетки. Повышение разрешающей способности вблизи пластины на этих сетках достигается за счет линейного сгущения узлов в направление нормалей к сторонам пластины. Это обеспечивает плавное постепенное изменение размера ячеек в области. На сетках второго типа помимо сгущения используется дробление ячеек в окрестности границ, что приводит к нарушению условия регулярности и достаточно резкому изменению размеров ячеек в зоне стыковки измельченной и основной сетки. Однако это позволяет существенно повысить разрешающую способность, не увеличивая многократно общее количество ячеек расчетной сетки. Максимальное количество ячеек используемых расчетных сеток достигает $3 \cdot 10^5$.

Дискретизация системы уравнений движения жидкости проводится по методу конечных объемов (FVM) в декартовой системе координат. Дискретные значения составляющих скорости и дискретные давления локализуются в центрах ячеек расчетных сеток. Для вычисления объемных интегралов по контрольному объему используется общая процедура Гаусса. Для аппроксимации градиента давления в расчетах применяется линейная интерполяция. В диффузионных слагаемых при дискретизации оператора Лапласа нормальные градиенты скорости на поверхности ячейки аппроксимируются с помощью симметричной схемы второго порядка с поправкой на неортогональность (в случае пластины с острыми торцами) [13], [14].

Для интерполяции переменных в конвективных слагаемых используется гибридная схема Спалдинга предложенная в работах [15] (аналог широко применяемой в конечно-элементной дискретизации схемы «Streamline upwind»). Она представляет собой комбинацию линейной и противопоточной интерполяций. Линейная интерполяция применяется в области, где сеточное число Рейнольдса (или число Пекле) $\mathrm{Re}_h < 2$. Применение схемы, однако, требует особой аккуратности, первый порядок точности противопоточной интерполяции может привести к существенному влиянию на решение численной диффузии. Как показывают результаты работ [16]-[18], для рассматриваемого класса задач, гибридная схема обеспечивает хорошее согласование численных результатов с экспериментальными данными в

широком диапазоне чисел Рейнольдса. Негативное влияние численной диффузии при этом можно минимизировать повышением разрешающей способности сетки вблизи обтекаемого тела и контролировать посредством изучения сеточной сходимости.

Для дискретизации системы уравнений по времени используется неявная схема Эйлера. Шаг по времени во всех расчетах выбирается из условия – максимальное число Куранта не превышает значения 0.1.

Решение дискредитированной задачи проводится с помощью метода PISO [19] (в реализации, изложенной в [14]). Решение системы уравнений для давления выполняется на основе метода сопряженных градиентов (PCG) с геометро-алгебраическим многосеточным предобуславливателем (GAMG). Системы уравнений для компонент скорости решается методом бисопряженных градиентов (PBiCG) с предиктором на основе неполной LU факторизации. Расчеты выполняются распределенным образом по технологии MPI с применением метода декомпозиции области решения.

4. Результаты

Для получения представления о влиянии формы торцов пластин на сопротивление в режимах с интенсивным вихреобразованием рассмотрим диапазон больших амплитуд колебания $4 \le KC \le 10$.

Течение вокруг пластины в этом диапазоне имеет периодический отрывной характер (см. рис. 1). На каждом полупериоде с пластины срывается пара вихрей. Срывы поочередно происходят противоположных углов пластины, за это режим часто называют диагональным [1], [3].

Как видно по рис. 1, структура течений вокруг усеченной и прямоугольной пластин в целом остается очень похожей, в частности идентичными остаются структура, размеры и положение, формируемых около пластин вихревых пар. Однако в окрестности торцов пластин все же наблюдаются видимые различия течения, связанные с изменением точек отрыва потока: для усеченных пластин – это вершина острого угла, у пластин с прямоугольным поперечным сечением отрыв происходит в вершине прямого угла со стороны набегающего потока.

Этот фактор сказывается на распределении давления по поверхности пластины (рис. 2). У усеченных пластин разница между давлением на правой и левой сторонах пластины в окрестности торцов меньше, чем у прямоугольной. Это, в конечном счете, приводит к снижению результирующего аэродинамического сопротивления пластины.

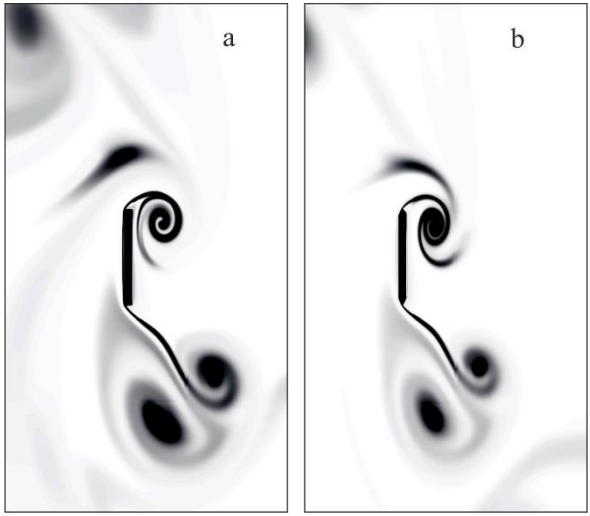


Рис. 1. Структура течения вблизи пластин при $\beta = 55$, $KC = 7$. a - прямоугольная пластина, b – усечённая пластина. Визуализация с помощью краски
The flow structure near the plates at $\beta = 55$, $KC = 7$. a is a rectangular plate, b is a truncated plate. Visualization using paint

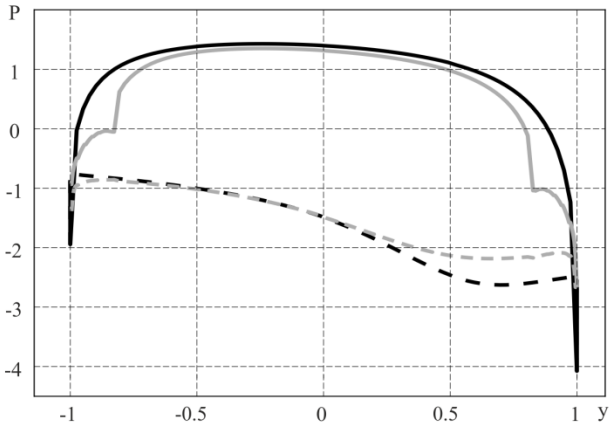


Рис. 2. Распределение давления на пластине при $\beta = 55$, $KC = 7$. Чёрная линия - прямоугольная пластина, серая линия – усечённая пластина. Пунктирная линия – левая сторона пластины, сплошная линия – правая сторона
Fig. 2. The pressure distribution on the plate at $\beta = 55$, $KC = 7$. The black line is a rectangular plate, the gray line is a truncated plate. The dashed line is the left side of the plate, the solid line is the right side

Графики изменения коэффициента сопротивления C_d в зависимости от KC для пластин с разной формой торцов представлены на рис. 3.

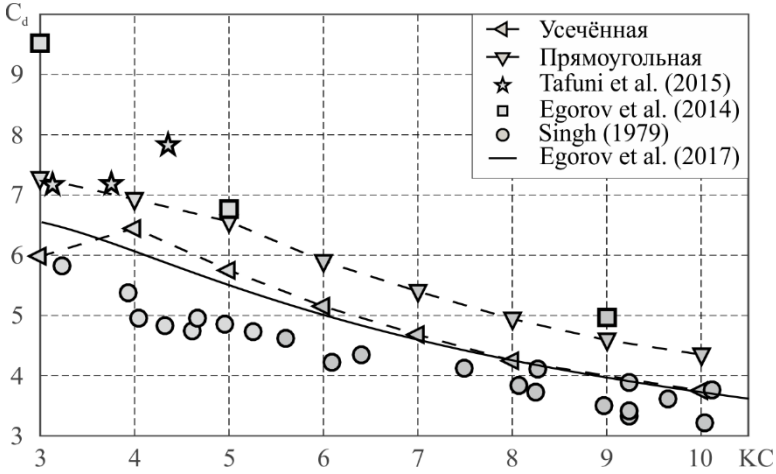


Рис. 3. Зависимость C_d от KC

Fig. 3. Dependence C_d on KC

Как видно, рассчитанные значения коэффициентов сопротивления для усеченной пластины хорошо согласуются с экспериментальными данными [1],[6], полученными для образцов с аналогичной геометрией. В тоже время как значения коэффициента сопротивления для прямоугольной пластины лежат в среднем на 14% выше в области большинства полученных ранее результатов численного моделирования [5], [10].

Список литературы

[1]. Singh S. Forces on Bodies in Oscillatory Flow. Phd thesis, University of London, 1979.

[2]. Keulegan G.H., Carpenter L.H. Forces on cylinders and plate in an oscillating fluid. Journal of Research of National Bureau of Standards, vol. 60, no. 5, pp. 423 – 440, 1958. DOI: 10.6028/jres.060.043

[3]. Graham J.M.R. The forces on sharp-edged cylinders in oscillatory flow at low Keulegan Carpenter numbers, Journal of Fluid Mechanics, vol.97, no. 02, pp. 331 – 346, 1980. DOI: 10.1017/s0022112080002595

[4]. Tuck E.O. Calculation of unsteady flows due to small motions of cylinders in a viscous fluid. Journal of Engineering Mathematics, vol.3, no. 1, pp. 29 – 44, 1969. DOI: 10.1007/BF01540828

[5]. Егоров А.Г., Камалутдинов А.М., Нуриев А.Н., Паймушин В.Н. Теоретико-экспериментальный метод определения параметров демпфирования на основе исследования затухающих изгибных колебаний тест-образцов: 2. Аэродинамическая составляющая демпфирования. Механика композитных материалов, 2014, том 50, вып. 3, стр. 379-396. DOI: 10.1007/s11029-014-9413-3

[6]. Егоров А.Г., Камалутдинов А.М., Нуриев А.Н., Паймушин В.Н. Экспериментальное определение демпфирования колебаний пластины вязкой жидкостью. Доклады Академии наук, 2017, том 474, вып. 2, стр. 172-176 DOI: 10.7868/S0869565217140079

[7]. Егоров А.Г., Камалутдинов А.М., Паймушин В.Н., Фирсов В.А. Теоретико-экспериментальный метод определения коэффициента аэродинамического сопротивления гармонически колеблющейся тонкой пластины. Прикладная механика и техническая физика, 2016, том 57, вып. 2 (336), стр. 96-104. DOI: 10.15372/PMTF20160210

[8]. Aureli M., Porfiri M. Low frequency and large amplitude oscillations of cantilevers in viscous fluids Applied Physics Letters, vol. 96, no. 16, p. 164102, 2010. DOI: 10.1063/1.3405720

[9]. Aureli M., Porfiri M., Basaran M.E. Nonlinear finite amplitude vibrations of sharp-edged beams in viscous fluids. Journal of Sound and Vibration, vol. 331, no. 7, pp. 1624 – 1654, 2012. DOI: 10.1016/j.jsv.2011.12.007

[10]. Tafuni A., Sahin. I. Non-linear hydrodynamics of thin laminae undergoing large harmonic oscillations in a viscous fluid. Journal of Fluids and Structures, vol. 52, pp. 101 – 117, 2015. DOI: 10.1016/j.jfluidstructs.2014.10.004

[11]. Jalalisendi M., Panciroli R., Cha Y., Porfiri M. A particle image velocimetry study of the flow physics generated by a thin lamina oscillating in a viscous fluid. J. Appl. Phys., vol. 115, no. 5, p. 054901, 2014. DOI: 10.1063/1.4863721

[12]. Morison J. R., Johnson J. W., Schaaf S. A. The Force Exerted by Surface Waves on Piles. Journal of Petroleum Technology, vol. 2, no. 05, pp. 149–154, 1950. DOI: 10.2118/950149-g

[13]. Greenshields C. OpenFOAM User Guide. CFD Direct, Available: https://cfd.direct/openfoam/user-guide/, accessed 12.12.2017

[14]. Нуриев А.Н., Зайцева О.Н. Решение задачи об осциллирующем движении цилиндра в вязкой жидкости в пакете OpenFOAM. Вестник Казанского технологического университета, 2013, том 16, No 8, стр. 116-123.

[15]. Spalding D.B. A novel finite difference formulation for differential expressions involving both first and second derivatives. International Journal for Numerical Methods in Engineering, vol. 4, no. 4, pp. 551 –559, 1972. DOI: 10.1002/nme.1620040409

[16]. Нуриев А.Н. Численное моделирование течений, возникающих около гармонически осциллирующего цилиндра, в диапазоне умеренных значений колебательного числа Рейнольдса. Материалы XI Международной конференции по неравновесным процессам в соплах и струях. (NPNJ'2016) 25-31 мая 2016 г. Алушта, М: Изд-во МАИ, 2016, С. 260-261

[17]. Justesen P. A numerical study of oscillating flow around a circular cylinder. J. Fluid Mech, vol. 222., pp. 157–196, 1991. DOI: 10.1017/S0022112091001040

[18]. An H., Cheng L., Zhao M. Steady streaming around a circular cylinder in an oscillatory flow. Ocean Engineering, vol. 36, no. 14., pp. 1089–1097, 2009. DOI: 10.1016/j.oceaneng.2009.06.010

[19]. Issa R.I. Solution of the implicitly discretised fluid flow equations by operator-splitting. J. Comput. Phys, 1986, vol. 62, no. 1, P. 40-65. DOI:10.1016/0021-9991(86)90099-9

Dependence of hydrodynamic forces acting on oscillating thin plates on the shape of edges in the range of large oscillation amplitudes.

A.N. Nuriev <nuriev_an@mail.ru>
A.M. Kamalutdinov <islamui@hotmail.com>
O.N. Zaitseva <olga_fdpi@mail.ru>
Kazan Federal University
18 Kremlyovskaya street, Kazan, Russia, 420008

Abstract. In this work, numerical simulation of the viscous flow past harmonically oscillating thin plates with different shapes of edges is carried out in the Reynolds number range 10 <Re <600. To describe the motion of a fluid, a complete nonstationary system of Navier-Stokes equations is solved. The problem is considered in a plane formulation. The numerical model is implemented on the basis of the open-source OpenFOAM package. The effect of the shape of edges on the hydrodynamic drag in regimes with intense vortex formation is considered. The structure of the flow and the pressure distribution over the plate surface are analyzed, the drag coefficient for different oscillation amplitudes is calculated. The results of the study show that the change of the shape of edges leads to the shift the flow separation points. This has noticeable effect on the pressure distribution on the plate surface. For truncated plates, the difference between the pressure distribution on the right and left sides of the plate in the vicinity of the edges is less than for the rectangular plate. This leads to a decrease the aerodynamic drag of the truncated plate. In the considered range of parameters the values of the drag coefficient for a rectangular plate lie (on the average) 14% higher. The obtained results well explain the large spread of data between the earlier experimental and numerical studies, since in almost all numerical studies the cross section of the plate is assumed rectangular. .At the same time, samples, which are usually used in experiments, have truncated (triangular) edges. The corresponding data for each of these types of plates are in good agreement with the results obtained in this study.

Keywords: viscous fluid; thin plates; oscillatory motion; drag coefficient; edge shape; numerical simulation

DOI: 10.15514/ISPRAS-2018-30(1)-12

For citation: Nuriev A.N., Kamalutdinov A.M., Zaitseva O.N. Dependence of hydrodynamic force. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 183-194 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-12

References

[1]. Singh S. Forces on Bodies in Oscillatory Flow. Phd thesis, University of London, 1979.
[2]. Keulegan G.H., Carpenter L.H. Forces on cylinders and plate in an oscillating fluid. Journal of Research of National Bureau of Standards, vol. 60, no. 5, pp. 423 – 440, 1958. DOI: 10.6028/jres.060.043

[3]. Graham J.M.R. The forces on sharp-edged cylinders in oscillatory flow at low Keulegan Carpenter numbers, Journal of Fluid Mechanics, vol.97, no. 02, pp. 331 – 346, 1980. DOI: 10.1017/s0022112080002595

[4]. Tuck E.O. Calculation of unsteady flows due to small motions of cylinders in a viscous fluid. Journal of Engineering Mathematics, vol.3, no. 1, pp. 29 – 44, 1969. DOI: 10.1007/BF01540828

[5]. Egorov A.G., Kamalutdinov A.M., Nuriev A.N., Paimushin V.N. Theoretical-experimental method for determining the parameters of damping based on the study of damped flexural vibrations of test specimens 2. Aerodynamic component of damping. Mechanics of Composite Materials, vol. 50, no. 3, pp. 267–278, 2014. DOI: 10.1007/s11029-014-9413-3.

[6]. Egorov A.G., Kamalutdinov A.M., Nuriev A.N., Paimushin V.N. Experimental determination of damping of plate vibrations in a viscous fluid. Doklady Physics, vol. 62, no. 5, pp. 257 – 261, 2017. DOI: 10.1134/S1028335817050068

[7]. Egorov A.G., Kamalutdinov A.M., Paimushin V.N., Firsov V.A. Theoretical-experimental method of determining the drag coefficient of a harmonically oscillating thin plate. Journal of Applied Mechanics and Technical Physics, vol. 57, no. 2, pp. 275 – 282, 2016. DOI: 10.1134/S0021894416020103

[8]. Aureli M., Porfiri M. Low frequency and large amplitude oscillations of cantilevers in viscous fluids Applied Physics Letters, vol. 96, no. 16, p. 164102, 2010. DOI: 10.1063/1.3405720

[9]. Aureli M., Porfiri M., Basaran M.E. Nonlinear finite amplitude vibrations of sharp-edged beams in viscous fluids. Journal of Sound and Vibration, vol. 331, no. 7, pp. 1624 – 1654, 2012. DOI: 10.1016/j.jsv.2011.12.007

[10]. Tafuni A., Sahin. I. Non-linear hydrodynamics of thin laminae undergoing large harmonic oscillations in a viscous fluid. Journal of Fluids and Structures, vol. 52, pp. 101 – 117, 2015. DOI: 10.1016/j.jfluidstructs.2014.10.004

[11]. Jalalisendi M., Panciroli R., Cha Y., Porfiri M. A particle image velocimetry study of the flow physics generated by a thin lamina oscillating in a viscous fluid. J. Appl. Phys., vol. 115, no. 5, p. 054901, 2014. DOI: 10.1063/1.4863721

[12]. Morison J. R., Johnson J. W., Schaaf S. A. The Force Exerted by Surface Waves on Piles. Journal of Petroleum Technology, vol. 2, no. 05, pp. 149–154, 1950. DOI: 10.2118/950149-g

[13]. Greenshields C. OpenFOAM User Guide. CFD Direct, Available: https://cfd.direct/openfoam/user-guide/, accessed 12.12.2017

[14]. Nuriev A.N., Zaytseva O.N. Solution to the problem of oscillatory motion of a cylinder in a viscous fluid in the OpenFOAM package. Heald of Kazan Technological University vol. 16, no. 8, pp. 116-123, 2013.

[15]. Spalding D.B. A novel finite difference formulation for differential expressions involving both first and second derivatives. International Journal for Numerical Methods in Engineering, vol. 4, no. 4, pp. 551 –559, 1972. DOI: 10.1002/nme.1620040409

[16]. Nuriev A.N. Numerical simulation of the flow around the harmonically oscillating cylinder in the range of moderate vibrational Reynolds numbers. Proceedings of the XI international conference on nonequilibrium processes in nozzles an jets. Izdatelstvo MAI, Publishing House, pp. 260-261, 2016.

[17]. Justesen P. A numerical study of oscillating flow around a circular cylinder. J. Fluid Mech, vol. 222., pp. 157–196, 1991. DOI: 10.1017/S0022112091001040

[18]. An H., Cheng L., Zhao M. Steady streaming around a circular cylinder in an oscillatory flow. Ocean Engineering, vol. 36, no. 14., pp. 1089–1097, 2009. DOI: 10.1016/j.oceaneng.2009.06.010

[19]. Issa R.I. Solution of the implicitly discretised fluid flow equations by operator-splitting. J. Comput. Phys, 1986, vol. 62, no. 1, P. 40-65. DOI:10.1016/0021-9991(86)90099-9

Сравнение эффективности решателей разреженных систем линейных алгебраических уравнений на основе методов BiCGStab и FGMRES

И.К. Марчевский <iliamarchevsky@mail.ru>
В.В. Пузикова <vvp@dms-at.ru>
МГТУ им. Н.Э. Баумана,
105005, Россия, г. Москва, ул. 2-я Бауманская, дом 5

Аннотация. В данной работе проведено сравнение эффективности решателей разреженных систем линейных алгебраических уравнений, построенных на основе одних из наиболее быстрых итерационных методов, метода BiCGStab (метода бисопряженных градиентов со стабилизацией) и метода FGMRES (гибкого метода обобщенных минимальных невязок). В работе представлены оценки трудоемкости выполнения одной итерации рассматриваемых методов. Получено условие, которому должна удовлетворять размерность подпространства Крылова в методе FGMRES для того, чтобы трудоемкость одной итерации данного метода была меньше трудоемкости одной итерации метода BiCGStab. Кроме того представлена модификация метода FGMRES, позволяющая останавливать алгоритм до очередного рестарта в случае достижения заданной точности. На языке C++ на основе представленных алгоритмов методов BiCGStab и FGMRES (в том числе с ILU- и многосеточным предобуславливанием) разработаны решатели разреженных систем линейных алгебраических уравнений. Сравнение эффективности разработанных решателей проводилось на разностных аналогах уравнений Гельмгольца и Пуассона. Системы были взяты из тестовой задачи о моделировании обтекания кругового профиля, совершающего вынужденные поперечные колебания. Разностная схема для решения задачи строится на прямоугольной структурированной сетке интегро-интерполяционным методом LS-STAG – методом погруженных границ с функциями уровня. Вычислительные эксперименты показали, что метод FGMRES на задачах такого класса демонстрирует более высокую скорость сходимости по сравнению с методом BiCGStab. Время проведения расчета при использовании метода FGMRES сократилось более чем в 6.5 раз без предобуславливания и примерно в 3 раза с предобуславлванием. Реализация модифицированного алгоритма метода FGMRES также сравнивалась с аналогичным решателем из библиотеки Intel® Math Kernel Library. Вычислительные эксперименты показали, что разработанная реализация метода FGMRES позволила получить ускорение по сравнению с использованием Intel® MKL в 3.4 раза без предобуслвливания и в 1.4 раза при использовании ILU-предобуславливания.

Ключевые слова: разреженные системы линейных алгебраических уравнений; методы крыловского типа; метод BiCGStab; метод FGMRES; предобуславливание; многосеточный метод; уравнение Гельмгольца; уравнение Пуассона; метод погруженных границ LS-STAG; библиотека Intel Math Kernel Library

DOI: 10.15514/ISPRAS-2018-30(1)-13

Для цитирования: Марчевский И.К., Пузикова В.В. Сравнение эффективности решателей разреженных систем линейных алгебраических уравнений на основе методов BiCGStab и FGMRES. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 195-214. DOI: 10.15514/ISPRAS-2018-30(1)-13

1. Введение

Численное моделирование используется для решения множества научных и промышленных задач. При этом значительная часть вычислительных ресурсов тратится на решение разреженных систем линейных алгебраических уравнений (СЛАУ) большого размера, возникающих при дискретизации соответствующих дифференциальных или интегро-дифференциальных уравнений. Для решения таких систем, как правило, используются различные итерационные методы [1].

Отметим, что существуют также прямые методы решения разреженных систем, в частности решатель PARDISO [2]. Автор этого метода совместно с коллективом разработчиков реализовал его в программе "PARDISO 5.0.0 Solver Project", которая согласно информации на сайте данного проекта [3] представляет собой потокобезопасное, высокопроизводительное, надежное, эффективное для памяти и простое в использовании программное обеспечение для решения больших разреженных симметричных и несимметричных линейных систем уравнений в многопроцессорных системах с общей памятью и распределенной памятью. Кроме того, данный решатель реализован в библиотеке высокооптимизированных математических алгоритмов Intel® Math Kernel Library (MKL), в документации [4] которой он позиционируется как высокопроизводительный параллельный прямой решатель для систем с разреженными матрицами, оказывающийся эффективнее итерационных решателей при решении систем с числом уравнений менее ста тысяч. Тем не менее, даже при соблюдении указанного ограничения на размер СЛАУ возникают случаи, когда использование прямого решателя PARDISO приводит к увеличению времени счета в 9-10 раз по сравнению с итерационными методами [5]. По этой причине в данной работе далее будут рассматриваться исключительно итерационные методы решения СЛАУ.

Затраты вычислительных ресурсов при проведении моделирования можно существенно сократить за счет выбора для решения СЛАУ итерационных методов, обладающих высокой скоростью сходимости и низкой трудоемкостью на проведение одной итерации. К сожалению, несмотря на то, что возможность выбора итерационного метода для решения СЛАУ имеется

во многих коммерческих программных пакетах, применяемых при решении широкого класса задач механики сплошной среды (ANSYS [6], NASTRAN [7], FLUENT [8], STAR-CD [9] и др.), пользователи крайне редко пользуются ей, предпочитая оставлять те методы решения линейных систем, которые в программном пакете выбраны «по умолчанию». Однако при работе с открытыми свободно распространяемыми пакетами, такими как OpenFOAM [10], пользователям приходится явно указывать методы решения СЛАУ, поскольку установки «по умолчанию» отсутствуют. Пакет Kratos [11] также позволяет выбирать решатели из поддерживаемых внешних библиотек, таких как Intel® MKL [4] и AMGCL [12]. У пакета OpenFOAM существует расширенная версия [13], в которой пользователю доступно больше вариантов решателей СЛАУ. Помимо выбора решателя из числа предлагаемых и настроек его параметров в указанных пакетах существует возможность добавления своего решателя. Решатель при этом может быть как реализован "с нуля", так и базироваться на наработках из таких открытых библиотек, как AMGCL [12], PETSc [14] и др. При этом необходимо не только выбрать эффективный итерационный метод, но и построить на основе его эффективный вычислительный алгоритм.

Целью данной работы является сравнение эффективности решателей разреженных систем линейных алгебраических уравнений, построенных на основе одних из наиболее быстрых итерационных методов, метода BiCGStab (метода бисопряженных градиентов со стабилизацией [15]) и метода FGMRES (гибкого метода обобщенных минимальных невязок [16]), относящихся к проекционным методам крыловского типа [1].

2. Постановка задачи

Рассмотрим систему линейных алгебраических уравнений

$$Ax = b, \tag{1}$$

возникающую при дискретизации уравнения $Lu = f$. Здесь $A \in M(R)_{N_d \times N_d}$; $x, b \in R^{N_d}$; L – дифференциальный либо интегро-дифференциальный оператор; f – известная функция; u – искомая функция. Будем считать, что матрица A является разреженной (имеет M_d ненулевых элементов, причем M_d много меньше N_d^2), невырожденной ($\det A \neq 0$) и не обладает специальными свойствами (симметрией, положительной определенностью). Для решения систем вида (1) с точностью ε далее будем использовать методы BiCGStab и FGMRES, в том числе в сочетании с ILU-предобуславливателем [17] и многосеточным [18].

В качестве тестовой задачи рассмотрим задачу о моделировании обтекания кругового профиля, совершающего в невозмущенном потоке вынужденные

поперечные колебания по заданному закону [19]. Разностная схема для решения задачи строится на прямоугольной структурированной сетке интегро-интерполяционным методом LS-STAG [20] – методом погруженных границ [21] с функциями уровня [22]. На каждом шаге расчета по времени решение задачи сводится к решению уравнения Гельмгольца

$$(\Delta + k^2)u_1 = f_1 \tag{2}$$

для прогноза скорости u_1 и уравнения Пуассона

$$\Delta u_2 = f_2 \tag{3}$$

для поправки давления u_2 (здесь Δ – оператор Лапласа). Разностные аналоги уравнений (2) и (3) представляют собой системы линейных алгебраических уравнений вида (1), для которых $N_d = 71040$, $M_d = 354128$. Разностный аналог уравнения Пуассона (3) в отличие от разностного аналога уравнения Гельмгольца (2) является плохо обусловленной системой [23], поэтому требует особой тщательности при выборе решателя.

3. Метод BiCGStab

Алгоритм метода BiCGStab с предобуславливанием [1] показан ниже (4).

$$\begin{aligned}
&r^0 = b - Ax^0,\ p^0 = r^0,\ \forall r_*^0 : (r_*^0, r^0) \neq 0 \\
&for \quad n = 0,1,\text{K}, \quad while\,(\| r^{n+1} \|_2 \geq \varepsilon),\ do \\
&\quad y^n = M^{-1} p^n \\
&\quad \alpha_n = \frac{(r_*^0, r^n)}{(Ay^n, r_*^0)} \\
&\quad s^n = r^n - \alpha_n A y^n \\
&\quad z^n = M^{-1} s^n \\
&\quad \omega_n = \frac{(Az^n, s^n)}{(Az^n, Az^n)} \\
&\quad x^{n+1} = x^n + \alpha_n y^n + \omega_n z^n \\
&\quad r^{n+1} = s^n - \omega_n A z^n \\
&\quad \beta_n = \frac{\alpha_n (r^{n+1}, r_*^0)}{\omega_n (r^n, r_*^0)} \\
&\quad p^{n+1} = r^{n+1} + \beta_n (p^n - \omega_n A y^n).
\end{aligned} \tag{4}$$

Здесь x^n – n-е итерационное приближение к искомому решению x; $r^n = b - Ax^n$ – вектор невязки на n-й итерации; $p^n = x^{n+1} - x^n$ – вектор коррекции на n-й итерации; M – матрица предобуславливателя [23]. Если матрицу M положить равной единичной матрице соответствующей размерности, приведенный выше алгоритм становится алгоритмом метода BiCGStab без предобуславливания. Явного обращения матрицы M при проведении вычислений не производится: вместо этого решаются системы $My^n = p^n$ и $Mz^n = s^n$. При использовании ILU-предобуславливания для решения этих систем используется неполная LU-факторизация (ILU-факторизация) матрицы A, а при использовании многосеточного предобуславливателя – многосеточный метод. В рамках данной статьи на алгоритмах построения предобуславливателей подробно останавливаться не будем.

Трудоемкость выполнения одной итерации алгоритма будем определять по числу умножений. При этом затраты на выполнение шагов, связанных с предобуславливанием, учитывать не будем. Таким образом, трудоемкость одной итерации алгоритма (4) без предобуславливания составляет

$$\lambda_{BiCGStab} = 2M_d + 11N_d. \quad (5)$$

4. Метод FGMRES

Метод FGMRES, как и метод BiCGStab, относится к методам крыловского типа [1]. Основное различие между этими методами заключается в способе построения базиса в подпространстве Крылова: в методе BiCGStab для этого используется биортогонализация Ланцоша, а в FGMRES – ортогонализация Арнольди [1]. Алгоритм метода FGMRES с рестартами через каждые m итераций и гибким предобуславливанием [1] показан ниже (6).

Здесь $\overline{H}_m \in M(R)_{(m+1)\times m}$ – матрица коэффициентов ортогонализации $H_m \in M(R)_{m\times m}$ (верхняя матрица Хессенберга), дополненная строкой $(0 \quad \text{K} \quad 0 \quad h_{m+1,m})$; $e^1 = (1 \quad 0 \quad \text{K} \quad 0)^T \in R^{m+1}$; M_j – матрица предобуславливателя на j-й итерации. Если матрицу M_j на каждой итерации положить равной единичной матрице соответствующей размерности, приведенный выше алгоритм становится алгоритмом метода FGMRES без предобуславливания (GMRES). Таким образом, отличие алгоритмов с предобуславливанием и без состоит только в одном шаге ($z^j = M_j^{-1} v^j$); при этом явного обращения матрицы M_j при проведении вычислений не производится: вместо этого решается система $M_j z^j = v^j$.

Отметим, что векторы v^1,K,v^m образуют ортогональный базис в подпространстве Крылова размерности m, порожденным вектором v^1 и матрицей A, $K_m = K_m(A,v^1) = span\{v^1, Av^1, A^2v^1,\mathrm{K}, A^{m-1}v^1\}$ [1].

$$
\begin{aligned}
&Start:\ r^0 = b - Ax^0,\ \beta = \| r^0 \|_2,\ v^1 = r^0/\beta\\
&for\quad j = 1,\mathrm{K},m\\
&\qquad z^j = M_j^{-1}v^j\\
&\qquad w^j = Az^j\\
&\qquad for\quad i = 1,\mathrm{K},j\\
&\qquad\qquad h_{i,j} = (w^j, v^i)\\
&\qquad\qquad w^j = w^j - h_{i,j}v^i\\
&\qquad h_{j+1,j} = \| w^j \|_2\\
&\qquad v^{j+1} = w^j / h_{j+1,j}\\
&Z_m = \left[z^1 \quad \mathrm{K} \quad z^m\right],\ \overline{H}_m = \left\{h_{i,j}\right\}\\
&y^m = \arg\min_y \| \beta e^1 - \overline{H}_m y \|_2\\
&x^m = x^0 + Z_m y^m\\
&if\ (\| b - Ax^m \|_2 \geq \varepsilon)\\
&\qquad x^0 = x^m\\
&\qquad goto\ Start.
\end{aligned}
\qquad (6)
$$

Из-за того, что в алгоритме (6) критерий останова проверяется через каждые m итераций, количество итераций при использовании данного алгоритма всегда будет кратно m. Средняя трудоемкость одной итерации алгоритма (6) без предобуславливания составляет

$$
\lambda_{FGMRES} = \frac{\sum_{j=1}^{m}\left(M_d + 2N_d + \sum_{i=1}^{j} 2N_d\right)}{m} = \frac{(M_d + 2N_d)m + 2N_d\sum_{j=1}^{m} j}{m} = \qquad (7)
$$

$$
= \frac{(M_d + 2N_d)m + N_d m(m+1)}{m} = M_d + (3+m)N_d.
$$

Из формул (5) и (7) следует, что трудоемкость одной итерации метода FGMRES оказывается меньше трудоемкости одной итерации метода BiCGStab при выполнении следующего условия:

$$m < \frac{M_d}{N_d} + 8 . \qquad (8)$$

Поскольку при построении разностного аналога в методе LS-STAG используется пятиточечный шаблон дискретизации, можно считать, что $M_d \approx 5N_d$. Тогда из условия (8) получается, что число m в алгоритме (6) должно быть меньше 13.

5. Модификация алгоритма метода FGMRES

Недостаток алгоритма (6) заключается в том, что критерий останова проверяется не на каждой итерации алгоритма. Из-за этого получается, что могут выполняться "лишние" итерации. Построим модификацию алгоритма (6), позволяющую устранить данный недостаток.

Алгоритм (6) требует решения линейной задачи наименьших квадратов $y^m = \arg\min_y \| \beta e^1 - \overline{H}_m y \|_2$, т. е. решения системы

$$\overline{H}_m y = \beta e^1 . \qquad (9)$$

Для этого используем QR-разложение, построенное при помощи метода вращений [1]: умножим (9) слева на матрицу $Q_m = \Omega_m \cdot \mathrm{K} \cdot \Omega_1$, $Q_m, \Omega_i \in M(R)_{(m+1)\times(m+1)}$, $i = \overline{1, m}$, где Ω_i – матрица вращений Гивенса: i-й и $(i+1)$-й диагональные элементы этой матрицы равны $c_i = h_{i+1,i}/\Delta_i$, $\Delta_i = \sqrt{h_{i,i}^2 + h_{i+1,i}^2}$, остальные диагональные элементы – единицы; $(i+1)$-й элемент i-й строки равен $s_i = h_{i,i}/\Delta_i$, i-й элемент $(i+1)$-й строки равен $(-s_i)$, остальные элементы – нули. В итоге получаем систему

$$\overline{R}_m y = \bar{g}^m . \qquad (10)$$

Здесь $\overline{R}_m = Q_m \overline{H}_m \in M(R)_{(m+1)\times m}$, $g^m = Q_m \left(\beta e^1\right) = (\gamma_1 \quad \mathrm{K} \quad \gamma_{m+1})^T \in R^{m+1}$. После удаления из (10) последнего уравнения получаем

$$R_m y_m = g_m , \qquad (11)$$

где $R_m \in M(R)_{m\times m}$ – верхнетреугольная матрица, что позволяет решить эту систему при помощи обратного хода метода Гаусса. Полученное решение y^m системы (11) также является решением задачи (9) и позволяет вычислить

итерационное приближение x^m из (6). При этом для нормы вектора невязки из (6) справедливо равенство $\| b - Ax^m \|_2 = |\gamma_{m+1}|$ [1]. Благодаря этому свойству критерий останова можно проверять на каждой итерации без решения системы (11), и, если он выполняется, вычислять решение, не дожидаясь m-й итерации. При этом удобно хранить матрицы Z_m и H_m по столбцам (packed storage). Для того, чтобы пересчитывать на j-й итерации элементы j-го столбца матрицы H_m и элементы правой части системы (11) g_j и $g_{j+1} = \gamma$, также необходимо хранить векторы косинусов и синусов $c = (c_1 \quad \text{K} \quad c_m)^T, s = (s_1 \quad \text{K} \quad s_m)^T \in R^m$.

Тогда алгоритм (6) можно переписать следующим образом так, как показано ниже (13).

Средняя трудоемкость одной итерации алгоритма (13) без предобуславливания составляет совпадает с оцениваемой по формуле (7) для итераций, предшествующих рестарту, и равна

$$\lambda_{FGMRES-mod} = M_d + (3+k)N_d \tag{12}$$

для итераций, после которых не было рестарта (т.е. сработал критерий останова). Поскольку $k \le m$, получаем, что $\lambda_{FGMRES-mod} \le \lambda_{FGMRES}$. Соответственно, если размерность m подпространства Крылова удовлетворяет условию (8), то трудоемкость одной итерации модифицированного алгоритма метода FGMRES (13) оказывается меньше трудоемкости одной итерации метода BiCGStab, также как и трудоемкость одной итерации основного алгоритма метода FGMRES (6). При этом благодаря проверке критерия останова на каждой итерации алгоритма (13) вычислительные затраты на решение СЛАУ уменьшаются по сравнению с использованием исходного алгоритма (6).

6. Вычислительные эксперименты

Для описанной выше тестовой задачи о моделировании обтекания кругового профиля, совершающего в невозмущенном потоке вынужденные поперечные колебания по заданному закону [19], были проведены расчеты, состоящие из 301 шага по времени. Таким образом, в рамках каждого расчета решалась серия СЛАУ, состоящая из 602 разностных аналогов уравнений Гельмгольца (2) и 301 разностного аналога уравнений Пуассона (3). Системы решались с точностью $\varepsilon = 10^{-6}$.

$$
\begin{aligned}
&Start:\ r^0 = b - Ax^0,\ \beta = \| r^0 \|_2,\ w^0 = r^0,\ \gamma = \beta,\ k = m \\
&for \quad j = 1, \mathrm{K}\ , m \\
&\qquad v^j = w^{j-1} / \beta \\
&\qquad z^j = M_j^{-1} v^j \\
&\qquad w^j = A z^j \\
&\qquad for \quad i = 1, \mathrm{K}\ , j \\
&\qquad\qquad h_{i,j} = (w^j, v^i) \\
&\qquad\qquad w^j = w^j - h_{i,j} v^i \\
&\qquad for \quad i = 1, \mathrm{K}\ , j-1 \\
&\qquad\qquad \tilde{h} = h_{i,j},\ h = h_{i+1,j} \\
&\qquad\qquad h_{i,j} = c_i \tilde{h} + s_i h \\
&\qquad\qquad h_{i+1,j} = -s_i \tilde{h} + c_i h \\
&\qquad \tilde{h} = h_{j,j},\ \beta = \| w^j \|_2 \\
&\qquad h_{j,j} = \sqrt{\tilde{h}^2 + \beta^2} \\
&\qquad c_j = \beta / h_{j,j} \\
&\qquad s_j = \tilde{h} / h_{j,j} \\
&\qquad g_j = \gamma c_j \\
&\qquad \gamma = -\gamma s_j \\
&\qquad if\ (|\gamma| < \varepsilon) \\
&\qquad\qquad k = j \\
&\qquad\qquad j = m + 1 \\
&Z_k = \left[z^1 \quad \mathrm{K} \quad z^k \right],\ H_k = \left\{ h_{i,j} \right\},\ g^k = \left(g_1 \quad \mathrm{K} \quad g_k \right)^T \\
&x^m = x^0 + Z_k H_k^{-1} g^k \\
&if\ (|\gamma| \geq \varepsilon) \\
&\qquad x^0 = x^m \\
&\qquad goto\ Start.
\end{aligned}
\tag{13}
$$

Табл. 1. Минимальное, среднее и максимальное число итераций, а также среднее квадратичное отклонение (СКО) числа итераций при решении серии разностных аналогов уравнений Гельмгольца разработанными решателями
Table 1. The minimum, average and maximum iterations number, as well as the standard deviation (SD) of iterations number when solving a series of the Helmholtz equations difference analogues by developed solvers

Метод	Минимальное	Среднее	Максимальное	СКО
BiCGStab	1	24	58	5
FGMRES	0	33	52	5
BiCGStab+ILU	1	2	2	0
FGMRES +ILU	0	2	2	0
BiCGStab+MG	1	2	4	0
FGMRES +MG	0	2	4	0

Табл. 2. Минимальное, среднее и максимальное число итераций, а также среднее квадратичное отклонение (СКО) числа итераций при решении серии разностных аналогов уравнений Пуассона разработанными решателями
Table 2. The minimum, average and maximum iterations number, as well as the standard deviation (SD) of iterations number when solving a series of the Poisson equations difference analogues by developed solvers

Метод	Минимальное	Среднее	Максимальное	СКО
BiCGStab	88	588	1671	300
FGMRES	98	105	107	1
BiCGStab+ILU	16	105	254	49
FGMRES +ILU	89	92	93	0
BiCGStab+MG	6	24	177	13
FGMRES +MG	5	7	13	1

Напомним, что размерность систем равнялась $N_d = 71040$, а матрицы систем содержали $M_d = 354128$ ненулевых элементов. В решателях реализованы алгоритмы (6) и (13). Размерность подпространства Крылова в решателях, основанных на методе FGMRES, была выбрана с учетом условия (8) и равнялась $m = 12$. Методы использовались как без предобуславливания, так и с ILU- и многосеточным предобуславливанием.

В табл. 1 и 2 собраны статистические сведения о сходимости разработанных решателей при решении серий разностных аналогов уравнений Гельмгольца и Пуассона соответственно: указаны минимальное, среднее и максимальное число итераций, а также среднее квадратичное отклонение (СКО) числа

итераций. Гистограммы распределения числа итераций для нескольких наиболее интересных случаев (решение серий разностных аналогов уравнений Гельмгольца и Пуассона рассматриваемыми методами без предобуславливания, а также уравнения Пуассона решателями с ILU-предобуславливанием) представлены на рис. 1–3.

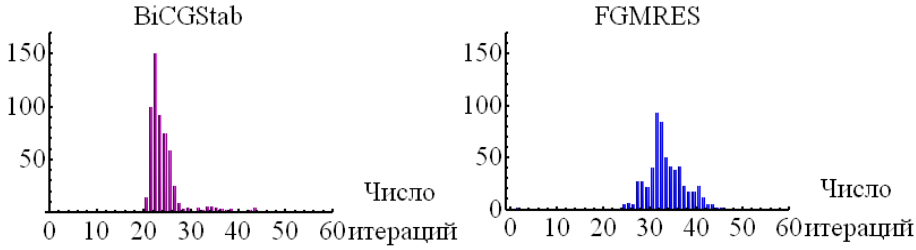


Рис. 1. Гистограмма распределения числа итераций при решении серии разностных аналогов уравнений Гельмгольца методами BiCGStab (слева) и FGMRES (справа) без предобуславливания

Fig. 1. A histogram of the iterations number distribution when solving a series of the Helmholtz equations difference analogues by the methods BiCGStab (left) and FGMRES (right) without preconditioning

Из табл. 1 и рис. 1 видно, что распределение числа итераций при решении серии разностных аналогов уравнений Гельмгольца (1) имеет примерно одинаковые статистическиее характеристики как в случае использования решателей, основанных на методе BiCGStab, так и в случае использования решателей, основанных на методе FGMRES. При этом необходимо отметить, что итерации модифицированного алгоритма метода FGMRES (12) менее дорогостоящие с вычислительной точки зрения чем итерации метод BiCGStab, поскольку размерность подпространства Крылова m в используемой реализации алгоритма метода FGMRES удовлетворяет полученному выше условию (8).

Иная картина наблюдается при решении серии плохо обусловленных систем, являющихся разностными аналогами уравнений Пуассона (3): видно, что при решении систем решателями, основанными на методе BiCGStab, число итераций изменяется в очень широком диапазоне и имеет значительное среднее квадратичное отклонение (табл. 2). При решении этих же систем решателями, основанными на методе FGMRES, напротив, число итераций мало зависит от шага по времени и имеет СКО, близкое к нулю (рис. 2 и 3). При этом среднее число итераций при использовании метода FGMRES оказывается в 5.60 раз меньше при сравнении методов без предобуслвливания, в 1.14 раз меньше при сравнении методов с использованием ILU-предобуславливания и в 3.43 раза – с использованием многосеточного предобуславливателя. Таким образом, за наименьшее число итераций

рассматриваемые разностные аналоги уравнения Пуассона позволяет решать метод FGMRES с многосеточным предобуславливанием.

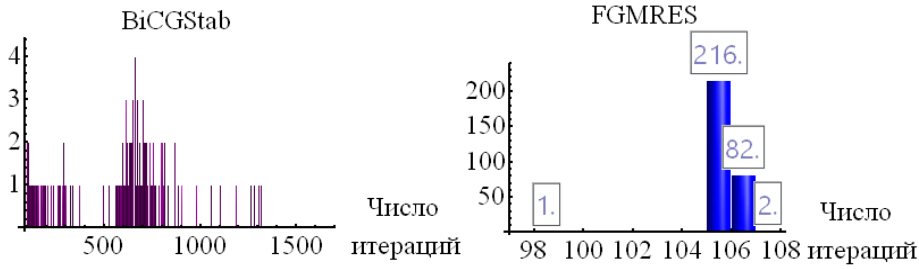


Рис. 2. Гистограмма распределения числа итераций при решении серии разностных аналогов уравнений Пуассона методами BiCGStab (слева) и FGMRES (справа) без предобуславливания

Fig. 2. A histogram of the iterations number distribution when solving a series of the Poisson equations difference analogues by the methods BiCGStab (left) and FGMRES (right) without preconditioning

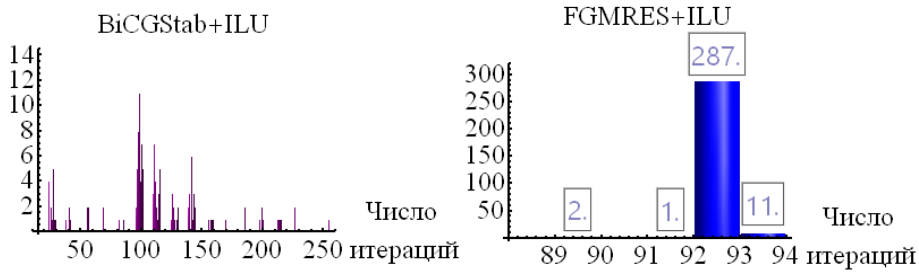


Рис. 3. Гистограмма распределения числа итераций при решении серии разностных аналогов уравнений Пуассона методами BiCGStab (слева) и FGMRES (справа) с ILU-предобуславливанием

Fig. 3. A histogram of the iterations number distribution when solving a series of the Poisson equations difference analogues by the methods BiCGStab (left) and FGMRES (right) with ILU preconditioning

Для разностного аналога уравнения Пуассона, который решается в рассматриваемой тестовой задаче на первом шаге по времени на рис. 4 в логарифмическом масштабе показано убывание нормы невязки при использовании разработанных решателей. Видно, что при использовании решателей, основанных на методе FGMRES, норма невязки убывает строго монотонно, причем после очередного рестарта скорость убывания резко увеличивается. При использовании решателей, основанных на методе BiCGStab, напротив, норма невязки убывает не монотонно: на итерациях с большими номерами (для рассматриваемого примера – больше 50)

наблюдаются резкие увеличения нормы невязки, характер расположения точек на графике в некоторые моменты расчета становится хаотичным. Эти эффекты наблюдаются для методов FGMRES и BiCGStab как при решении системы без предобуславливания, так и при решении рассматриваемой СЛАУ с реализованными предобуславливателями. Из всего этого можно сделать вывод, что сам метод FGMRES действует не только на высокочастотные компоненты ошибки, убираемые на первых итерациях, но и на низкочастотные, в то время как метод BiCGStab обладает преимущественно сглаживающими свойствами [23].

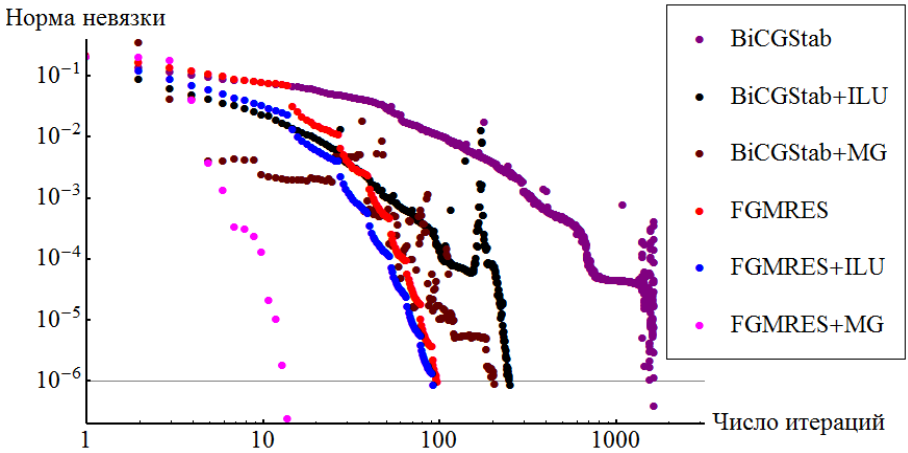


Рис. 4. Убывание нормы невязки при решении разностного аналога уравнения Пуассона разработанными решателями

Fig. 4. Decrease in the residual norm when solving the Poisson equation difference analogue by the developed solvers

Таким образом, метод FGMRES на задачах рассматриваемого класса демонстрирует более высокую скорость сходимости по сравнению с методом BiCGStab, причем особенно сильно это наблюдается при решении плохо обусловленных систем линейных алгебраических уравнений.

Определим, какое влияние оказывает использование того или иного решателя на общую продолжительность решения рассматриваемой тестовой задачи. На диаграмме на рис. 5 указано время проведения всех расчетов в мс. Вычислительные эксперименты проводились на рабочей станции, построенной на платформе Intel H81 с использованием двухядерного процессора Intel Core i3-4350T (Haswell) с поддержкой HyperThreading (4 логических ядра), работающем на частоте 3100 МГц. Станция оснащена 8 ГБайт оперативной памяти DDR3-1333, SSD-накопителем Crucial объемом 128 ГБайт и жестким диском Seagate объемом 1 ТБайт.

Из рис. 5 видно, что время проведения расчета при использовании метода FGMRES сократилось более чем в 6.5 раз без предобуславливания и примерно в 3 раза с предобуславлванием по сравнению с использованием аналогичных решателей, основанных на методе BiCGStab. При этом несмотря на то, что использование многосеточного предобуславливания позволяет решать рассматриваемые разностные аналоги уравнений Пуассона за наименьшее число итераций, использование данного предобуславливателя и для разностных аналогов уравнений Гельмгольца приводит к небольшому увеличению времени счета по сравнению с использованием ILU-предобуславливания, поскольку сокращения числа итераций при этом не происходит, а вычислительные затраты на предобуславливание увеличиваются.

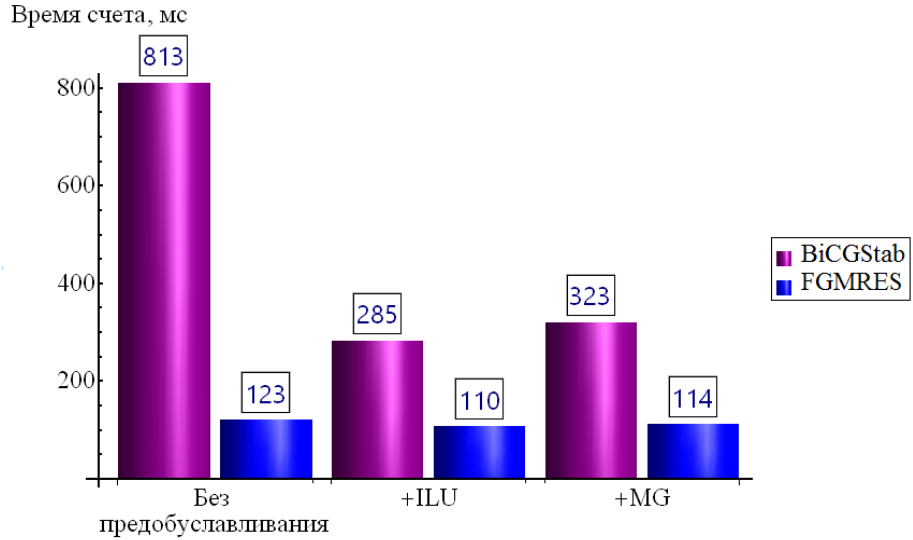


Рис. 5. Время счета при использовании различных решателей

Fig. 5. Computational time at different solvers usage

Проверим также, насколько правильным был выбор размерности подпространства Крылова m в используемой реализации алгоритма метода FGMRES исходя из полученного выше условия (8). Для этого сравним время счета при использовании решателей на основе метода FGMRES при $m=12$ и $m=20$. Из рис. 6 видно, что увеличение m действительно приводит к росту вычислительных затрат на проведение одной итерации и, соответственно, увеличению времени счета. Наиболее сильно это наблюдается при использовании методов без предобуславливания, поскольку при этом для

решения СЛАУ выполняется больше итераций. В этом случае время счета различается почти в 2 раза.

Также сравним разработанную реализацию модифицированного алгоритма метода FGMRES (12) с аналогом из библиотеки высокооптимизированных математических алгоритмов Intel® Math Kernel Library (MKL). В данной библиотеке не реализован многосеточный предобуславливатель, поэтому будем сравнивать только работу решателей без предобуславливания и с ILU-предобуслваливанием. Из рис. 6. видно, что разработанная реализация модифицированного алгоритма метода FGMRES при $m=12$ позволила получить ускорение по сравнению с использованием Intel® MKL в 3.4 раза без предобуслвливания и в 1.4 раза при использовании ILU-предобуславливания.

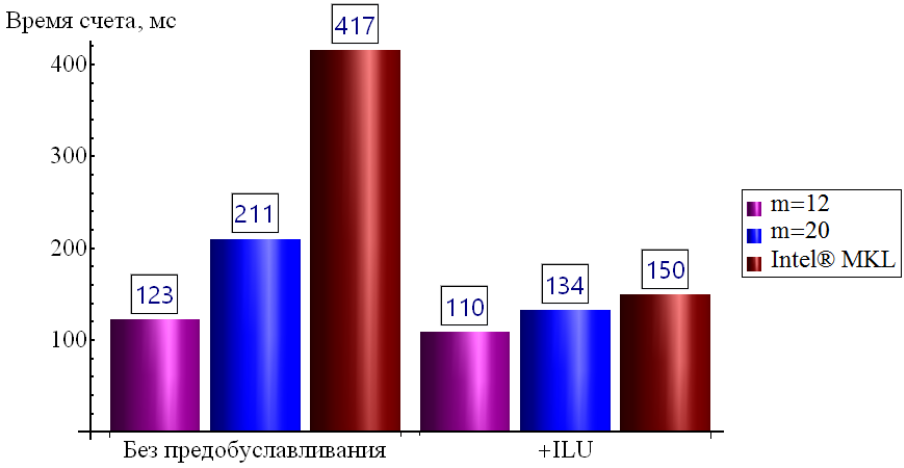


Рис. 6. Время счета при использовании различных реализаций метода FGMRES
Fig. 6. Computational time at different FGMRES method implementations usage

Выше было продемонстрировано, что использование многосеточного предобуславливания позволяет значительно сократить вычислительные затраты при решении разностных аналогов уравнений Пуассона, но при этом менее дорогостоящее с вычислительной точки зрения ILU-предобулавливание позволяет решать разностные аналоги уравнений Гельмгольца за то же число итераций, что и многосеточное предобуславливание (табл. 1). Поэтому логично при решении рассматриваемой тестовой задачи для решения разностных аналогов уравнений Гельмгольца использовать решатели с ILU-предобулавливанием, а для решения разностных аналогов уравнений Пуассона – решатели с многосеточным предобуславливанием. Результаты таких вычислительных экспериментов представлены на рис. 7.

Из рис. 7 видно, что применение различных методов предобуславливания позволило сократить время счета при использовании метода FGMRES

примерно в 2 раза по сравнению с применением одного и того же предобуславливателя для решения всех СЛАУ. При этом выбор значения размерности подпространства Крылова m, удовлетворяющего условию (8) позволяет получить ускорение примерно в 1.4 раза по сравнению с использованием бо́льших значений m. При этом ускорение по сравнению с использованием аналогичных решаелей, основанных на методе BiCGStab составляе примерно 4.2 раза.

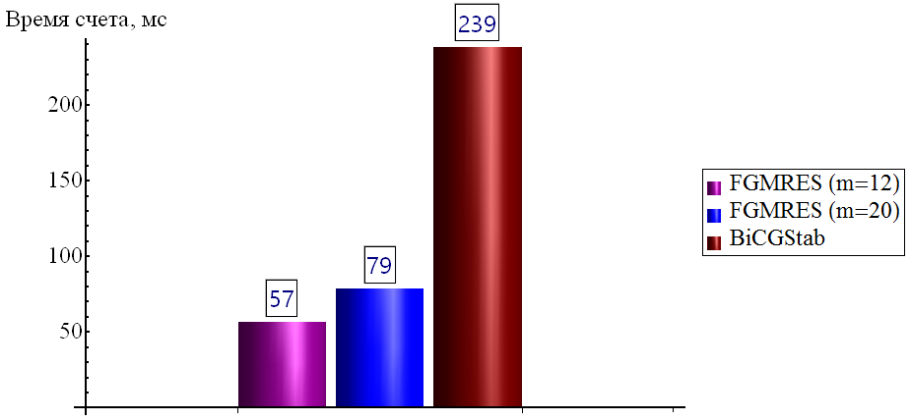


Рис. 7. Время счета при использовании решателей с ILU-предобуславливанием для разностных аналогов уравнений Гельмгольца и решателей с многосеточным предобуславливанием для разностных аналогов уравнений Пуассона

Fig. 7. Computational time when solving the Helmholtz equations difference analogues by the methods with ILU preconditioning and the Poisson equations difference analogues by the methods with multigrid preconditioning

7. Заключение

Исследованы решатели разреженных систем линейных алгебраических уравнений, построенные на основе методов BiCGStab и FGMRES. Получены оценки трудоемкости выполнения одной итерации этих методов. Кроме того, получено условие, которому должна удовлетворять размерность подпространства Крылова в методе FGMRES для того, чтобы трудоемкость одной итерации данного метода была меньше трудоемкости одной итерации метода BiCGStab. Представлена модификация метода FGMRES, позволяющая останавливать алгоритм до очередного рестарта в случае достижения заданной точности. На основе представленных алгоритмов разработаны решатели разреженных систем линейных алгебраических уравнений. разработаны решатели разреженных систем линейных алгебраических уравнений. Разработанные решатели позволяют решать системы как без

предобуславливания, так и использовать ILU- и многосеточное предобуславливание.

Сравнение эффективности разработанных решателей проводилось на разностных аналогах уравнений Гельмгольца и Пуассона из тестовой задачи о моделировании обтекания кругового профиля, совершающего вынужденные поперечные колебания. Вычислительные эксперименты показали, что метод FGMRES на задачах такого класса демонстрирует более высокую скорость сходимости по сравнению с методом BiCGStab. Время проведения расчета при использовании метода FGMRES сократилось более чем в 6.5 раз без предобуславливания и примерно в 3 раза с предобуславлванием. Для сравнениия эффективности также использовались аналогичные алгоритмы, реализованные в библиотеке высокооптимизированных математических алгоритмов Intel® Math Kernel Library. Расчеты показали, что разработанная реализация модифицированного алгоритма метода FGMRES позволила получить ускорение по сравнению с использованием Intel® MKL в 3.4 раза без предобуслвливания и в 1.4 раза при использовании ILU-предобуславливания.

Работа выполнена за счет гранта Российского научного фонда (проект №17-79-20445).

Список литературы

[1]. Saad Y. Iterative Methods for Sparse Linear Systems. N.Y.: PWS Publ., 1996. 547 p.
[2]. Schenk O., Gartner K. Solving unsymmetric sparse systems of linear equations with PARDISO // J. of Future Generation Computer Systems. 2004. № 20. P. 475–487.
[3]. PARDISO. URL: http://www.pardiso-project.org/ (accessed: 15.10.2017).
[4]. Intel® Math Kernel Library – Documentation. URL: https://software.intel.com/en-us/articles/intel-math-kernel-library-documentation (accessed: 15.10.2017).
[5]. Марчевский И.К., Пузикова В.В. Исследование эффективности распараллеливания вычислений при моделировании течений вязкой несжимаемой среды методом LS-STAG на системах с общей памятью. Вычислительные методы и программирование. 2015. Т. 16. С. 595–606.
[6]. ANSYS All Products. URL: http://www.ansys.com/products/all-products (accessed: 15.10.2017).
[7]. MSC Nastran. URL: http://www.mscsoftware.ru/products/msc-nastran (accessed: 15.10.2017).
[8]. ANSYS Fluent. URL: http://www.ansys.com/Products/Fluids/ANSYS-Fluent (accessed: 15.10.2017).
[9]. STAR-CD. URL: https://mdx.plm.automation.siemens.com/star-cd (accessed: 15.10.2017).
[10]. OpenFOAM® Documentation. URL: https://www.openfoam.com/documentation/ (accessed: 15.10.2017).
[11]. What is KRATOS. URL: http://www.cimne.com/kratos/about_whatiskratos.asp (accessed: 15.10.2017).
[12]. AMGCL documentation. URL: http://amgcl.readthedocs.io/en/latest/ (accessed: 15.10.2017).

[13]. Foam-extend. URL: https://sourceforge.net/projects/foam-extend/ (accessed: 15.10.2017).
[14]. Portable, Extensible Toolkit for Scientific Computation. URL: https://www.mcs.anl.gov/petsc/ (accessed: 15.10.2017).
[15]. Van der Vorst H.A. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems. SIAM J. Sci. Stat. Comp. 1992. № 2. P. 631–644.
[16]. Saad Y. A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Comput. 1993. № 14. P. 461–469.
[17]. Баландин М.Ю., Шурина Э.П. Методы решения СЛАУ большой размерности. Новосибирск: Изд-во НГТУ, 2000. 70 с.
[18]. Wesseling P. An introduction to multigrid methods. Chichester: John Willey & Sons Ltd., 1991. 284 p.
[19]. Guilmineau E., Queutey P. A numerical simulation of vortex shedding from an oscillating circular. J. Fluid Struct. 2002. № 16. P. 773–794.
[20]. Cheny Y., Botella O. The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. J.Comp. Phys.2010. №229. P.1043-1076.
[21]. Mittal R., Iaccarino G. Immersed boundary methods. Annu. Rev. Fluid Mech. 2005. № 37. P. 239–261.
[22]. Osher S., Fedkiw R.P. Level set methods and dynamic implicit surfaces. N. Y.: Springer, 2003. 273 p.
[23]. Марчевский И.К., Пузикова В.В. Анализ эффективности итерационных методов решения систем линейных алгебраических уравнений. Математическое моделирование и численные методы. 2014. № 4. С. 37–52.

The efficiency comparison of solvers for sparse linear algebraic equations systems based on the BiCGStab and FGMRES methods

I.K. Marchevsky <iliamarchevsky@mail.ru>
V.V. Puzikova <vvp@dms-at.ru>
BMSTU, 5 2nd Baumanskaya st., Moscow, Russian Federation, 105005

Abstract. The efficiency comparison of solvers for sparse linear algebraic equations systems based on one of the fastest iterative methods, the BiCGStab method (bi-conjugate gradient method with stabilization), and the FGMRES method (flexible method of generalized minimal residuals) is presented in this study. Estimates of computational cost per one iteration are presented for the considered methods. The condition imposed on the Krylov subspace dimensionality in the FGMRES is obtained. When this condition is fulfilled, the computational cost per one iteration of the FGMRES method is less than the computational cost per one iteration of the BiCGStab. In addition, the FGMRES modification, which allows to stop the algorithm before the next restart in case of achieving the specified accuracy, is presented. Solvers on the basis of presented the BiCGStab and FGMRES methods algorithms including ILU and multigrid preconditioning are developed on the C++ language for sparse linear algebraic equations systems. The efficiency comparison of developed solvers was carried out on the difference analogs of the Helmholtz and Poisson equations. The systems

were taken from the test problem about simulation of the flow around a circular profile, which makes forced transverse oscillations. The difference scheme for the problem solution is constructed by the LS-STAG method (immersed boundaries method with level-set functions). Computational experiments showed that the FGMRES demonstrates a higher convergence rate on problems of this class in comparison with the BiCGStab. The FGMRES usage allowed to reduce the computation time by more than 6.5 times without preconditioning and more than 3 times with preconditioning. The implementation of the modified FGMRES algorithm was also compared with a similar solver from the Intel® Math Kernel Library. Computational experiments showed that the developed FGMRES implementation allowed to obtain acceleration in comparison with Intel® MKL by 3.4 times without preconditioning and by 1.4 times with ILU-preconditioning.

Keywords: sparse linear algebraic equations systems; Krylov-space methods; the BiCGStab method; the FGMRES method; preconditioner; multigrid method; Helmholtz equation; Poisson equation; the LS-STAG immersed boundary method; Intel® Math Kernel Library

DOI: 10.15514/ISPRAS-2018-30(1)-13

For citation: Marchevsky I.K., Puzikova V.V. The efficiency comparison of solvers for sparse linear algebraic equations systems based on the BiCGStab and FGMRES methods. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 195-214 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-13

References

[1]. Saad Y. Iterative Methods for Sparse Linear Systems. N.Y.: PWS Publ., 1996. 547 p.
[2]. Schenk O., Gartner K. Solving unsymmetric sparse systems of linear equations with PARDISO // J. of Future Generation Computer Systems. 2004. № 20. P. 475–487.
[3]. PARDISO. URL: http://www.pardiso-project.org/ (accessed: 15.10.2017).
[4]. Intel® Math Kernel Library – Documentation. URL: https://software.intel.com/en-us/articles/intel-math-kernel-library-documentation (accessed: 15.10.2017).
[5]. Marchevsky I.K., Puzikova V.V. Efficiency investigation of computation parallelization for viscous incompressible flow simulation on systems with shared memory. *Vychislitel'nye metody i programmirovanie* [*Computational methods and programming*], 2015. Vol. 16. P. 595–606 (in Russian)
[6]. ANSYS All Products. URL: http://www.ansys.com/products/all-products (accessed: 15.10.2017).
[7]. MSC Nastran. URL: http://www.mscsoftware.ru/products/msc-nastran (accessed: 15.10.2017).
[8]. ANSYS Fluent. URL: http://www.ansys.com/Products/Fluids/ANSYS-Fluent (accessed: 15.10.2017).
[9]. STAR-CD. URL: https://mdx.plm.automation.siemens.com/star-cd (accessed: 15.10.2017).
[10]. OpenFOAM® Documentation. URL: https://www.openfoam.com/documentation/ (accessed: 15.10.2017).
[11]. What is KRATOS. URL: http://www.cimne.com/kratos/about_whatiskratos.asp (accessed: 15.10.2017).

[12]. AMGCL documentation. URL: http://amgcl.readthedocs.io/en/latest/ (accessed: 15.10.2017).
[13]. foam-extend. URL: https://sourceforge.net/projects/foam-extend/ (accessed: 15.10.2017).
[14]. Portable, Extensible Toolkit for Scientific Computation. URL: https://www.mcs.anl.gov/petsc/ (accessed: 15.10.2017).
[15]. Van der Vorst H.A. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems. SIAM J. Sci. Stat. Comp. 1992. № 2. P. 631–644.
[16]. Saad Y. A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Comput. 1993. № 14. P. 461–469.
[17]. Balandin M.Yu., Shurina E.P. Metody reshenija SLAU bol'shoj razmernosti [Methods for large SLAE solving]*Novosibirsk*: NGTU publ., 2000. 70 p. (in Russian)
[18]. Wesseling P. An introduction to multigrid methods. Chichester: John Willey & Sons Ltd., 1991. 284 p.
[19]. Guilmineau E., Queutey P. A numerical simulation of vortex shedding from an oscillating circular. J. Fluid Struct. 2002. № 16. P. 773–794.
[20]. Cheny Y., Botella O. The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. J.Comp. Phys.2010. №229. P.1043-1076.
[21]. Mittal R., Iaccarino G. Immersed boundary methods. Annu. Rev. Fluid Mech. 2005. № 37. P. 239–261.
[22]. Osher S., Fedkiw R.P. Level set methods and dynamic implicit surfaces. N. Y.: Springer, 2003. 273 p.
[23]. Marchevsky I.K., Puzikova V.V. The efficiency analysis of iterative methods for solving linear algebraic equations systems. *Matematicheskoe modelirovanie i chislennye metody* [*Mathematical modeling and numerical methods*], 2014. № 4. P. 37-52 (in Russian)

Математическое моделирование эволюции завихренности при пространственном обтекании тел методом вихревых петель

С.А. Дергачев <sadergachev@mail.ru>
МГТУ им. Н.Э. Баумана,
105005, г. Москва, ул. 2-я Бауманская, д.5. стр.1

Аннотация. Моделирование гидродинамических явлений, связанных с обтеканием подвижных деформируемых тел является актуальной инженерной задачей для различных технических приложений. В данной статье описан метод вихревых петель, позволяющий моделировать обтекание тел без необходимости предварительного задания линий схода завихренности. Приведено верифицирование методики на экспериментальных данных обтекания крыла конечного размаха и сферы. Показана точность, достаточная для применения программы и методики в инженерных приложениях. Сделаны выводы о возможности перехода к подвижным и деформируемым телам. Программа реализована на языке C++ с использованием библиотеки MPI для организации пересылки данных при параллельных вычислениях.

Ключевые слова: гидродинамика; нестационарные нагрузки; вихревые петли; вихри; метод вихревых петель

DOI: 10.15514/ISPRAS-2018-30(1)-14

Для цитирования: Дергачев С.А. Математическое моделирование эволюции завихренности при пространственном обтекании тел методом вихревых петель. Труды ИСП РАН, том 30, вып. 1, 2018 г., стр. 215-226. DOI: 10.15514/ISPRAS-2018-30(1)-14

1. Введение

Вычисление нестационарных гидродинамических нагрузок на подвижные деформируемые трехмерные объекты в безграничном объеме несжимаемой среды является актуальной задачей в различных инженерных приложениях.

Существуют сеточные и бессеточные подходы к вычислению этого вида нагрузок. При условии значительных перемещений и деформаций тела нестационарные нагрузки с точки зрения быстродействия расчетов удобнее вычислять бессеточными методами вычислительной гидродинамики. В случае отрывного обтекания тела несжимаемой средой нестационарные нагрузки во многом определяются процессами вихреобразования. В вихревых методах

вычислительной гидродинамики непосредственно моделируется возникновение и эволюция завихренности, а поле скоростей и давление вычисляются по известному полю завихренности. Вихревые бессеточные методы различаются по способам моделирования завихренности.

Наиболее известен метод дискретных вихрей, получивший развитие в научной школе С.М. Белоцерковского [1]. При моделировании пространственных течений завихренность заключена в замкнутых вихревых рамках, состоящих из наборов отрезков. Рамки в следе соединены в тонкие вихревые пелены. Узлы вихревых рамок перемещаются по полю скорости. Поверхность тела также моделируется замкнутыми вихревыми рамками. Рамки, моделирующие поверхность тела перемещаются в соответствии с перемещениями или деформациями тела. Рамки формируются при сходе завихренности с заданных линий на поверхности тела. Линии схода назначаются заранее перед проведением расчета, что затрудняет моделирование гладких деформируемых тел. Кроме того, как показывает практика расчетов, невозможность разрыва и перезамыкания вихревых пелен может служить причиной вычислительной неустойчивости в случае взаимодействия нескольких пелен или пелены с обтекаемой поверхностью.

В методах вихревых элементов завихренность моделируется отдельными, не связанными между собой, частицами - вортонами, компактными носителями завихренности. Основная часть завихренности в них сконцентрирована в ограниченной области: в точке, в эллипсоиде или на отрезке. При расчете обтекания тел обычно используется гипотеза Лайтхилла-Чорина, согласно которой вортоны генерируются по всей поверхности обтекаемого тела. Такой подход позволяет эффективно моделировать отрывы потока с гладких деформируемых тел. Однако, в силу теорем Гельмгольца во всем пространстве вокруг вортона распределена еще некоторая часть завихренности, называемая дополнительной. Дополнительная завихренность обеспечивает соленоидальность поля вихревого элемента. Ее интенсивность стремится к нулю на бесконечном удалении от компактной области. Однако результаты расчетов показывают, что дополнительная завихренность может приводить к неправильному вычислению нагрузок на цилиндрические тела [2].

Известно, что в природе завихренность имеет свойство образовывать замкнутые вихревые структуры: нити, кольца, петли. Данное свойство формализовано в теоремах Гельмгольца. Моделированию эволюции вихревых нитей, колец и петель посвящено значительное количество исследований [3,4]. Замкнутая непересекающаяся ломаная линия – вихревая петля, составленная из вортонов-отрезков, может быть использована в качестве вихревого «суперэлемента». Если вортоны, моделирующие такие элементы, имеют одинаковую интенсивность и связанны между собой дополнительная завихренность отсутствует, поскольку вся завихренность заключена в вихревых петлях.

Применение вихревых петель для расчета нестационарных гидродинамических нагрузок в инженерных задачах потребовало разработки

эмпирических моделей их эволюции, генерации на поверхности обтекаемого тела и вычисления на основе петель поля давления и нагрузок на конструкцию [5]. Целью данной работы является верификация разработанных моделей, алгоритмов и программы по известным экспериментальным данным, а также определение допустимых значений параметров расчета.

2. Постановка задачи

Моделируется течение несжимаемой среды, для которой эффекты вязкости учитываются только как причина рождения завихренности на поверхности тела и как причина перезамыкания вихревых петель.

Используются два уравнения гидродинамики: уравнение неразрывности и уравнение сохранения импульса (уравнение Гельмгольца):

$$div\,\vec{V} = 0;\ \frac{d\vec{\Omega}}{dt} = rot(\vec{V} \times \vec{\Omega});$$

где $\vec{V}(\vec{r},t)$ – нестационарное трехмерное поле скоростей; $\vec{r}$- радиус-вектор точки в неподвижной системе координат; $\vec{\Omega} = rot\vec{V}$- завихренность. Заданы граничные условия отсутствия возмущений на бесконечности и условие прилипания на обтекаемой поверхности. Начальные условия соответствуют режиму бесциркуляционного обтекания тела. Требуется в течение заданного промежутка времени определить распределение давления по поверхности тела и вычислить действующие на него нагрузки.

В рамках вихревого метода уравнение неразрывности удовлетворяется тождественно, поскольку поле скорости восстанавливается по полю завихренности в соответствии с законом Био-Савара. Уравнение сохранения импульса заменяется системой обыкновенных дифференциальных уравнений, описывающей в лагранжевой постановке эволюцию завихренности, которая моделируется системой из Np вихревых петель, каждая из которых состоит из Nv вортонов-отрезков. Все вихревые петли в расчёте имеют одинаковую интенсивность Г. Данная величина задается в составе исходных данных. Вклад в поле скоростей от одного вортона-отрезка единичной интенсивности определяется формулой:

$$\vec{V}_{ki}(\vec{P}) = \frac{1}{4\pi}\frac{\vec{a}}{|\vec{a}|}\left(\frac{\vec{S}_1 \cdot \Delta\vec{P}_{ki}}{|\vec{S}_1|} - \frac{\vec{S}_2 \cdot \Delta\vec{P}_{ki}}{|\vec{S}_2|}\right),$$

$$\vec{a} = \vec{S}_1 \times \Delta\vec{P}_{ki},\ \vec{S}_1 = \vec{P} - \vec{P}_{ki},\ \vec{S}_2 = \vec{P} - \vec{P}_{ki+1},\ i = 1,\ldots, Np;\ k = 1,\ldots Nv$$

Для устранения сингулярности в расчетах вводится радиус линейного сглаживания скорости - радиус вихревого элемента ε. Величины Np и Nv изменяются в процессе расчета из-за рождения новых петель на обтекаемом теле, а также из-за изменения длины и перезамыкания петель.

Блок-схема алгоритма решения задачи приведена на рис. 1.

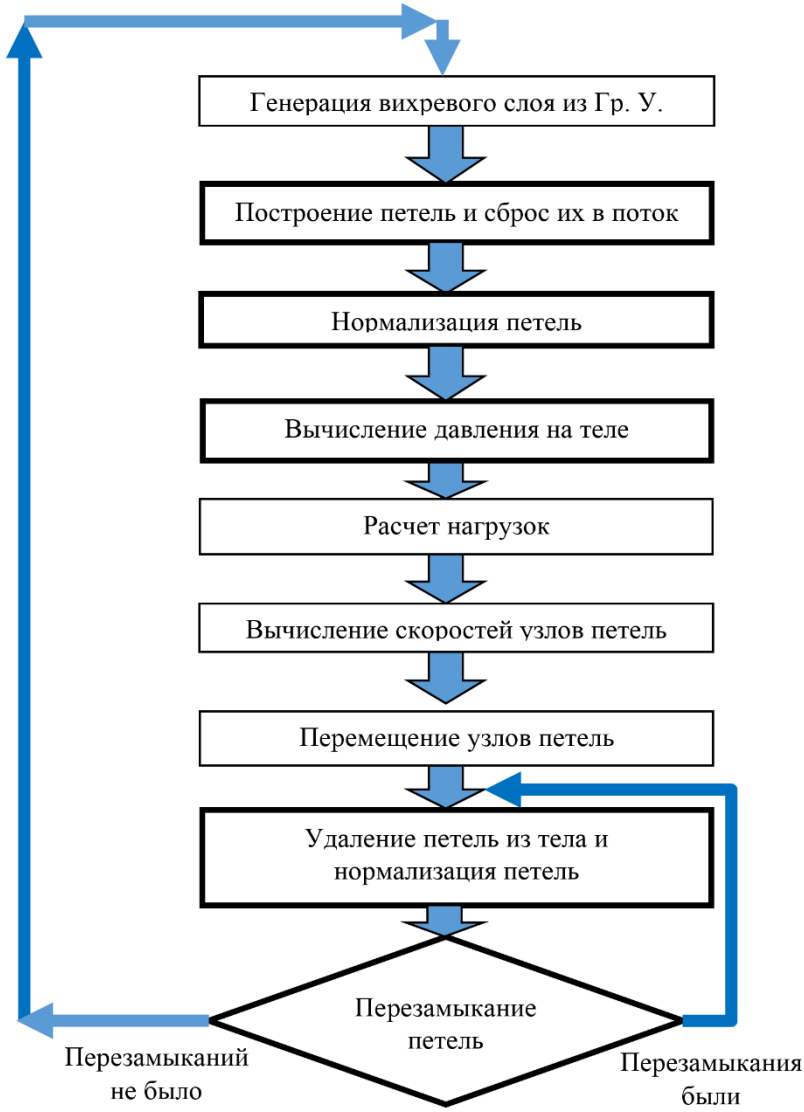


Рис. 1. Схема организации работы программы. Жирным выделены распараллеленные процедуры

Fig. 1. Scheme of the organization of the program. Parallel procedures are marked in bold.

Поверхность обтекаемого тела моделируется набором из N плоских панелей. Каждая панель представляет собой рамку из вортонов-отрезков, а в центре панели расположена контрольная точка. На каждом расчетном шаге из

условия обеспечения непротекания в контрольных точках (равенства нулю нормальной составляющей скорости) из решения СЛАУ находятся интенсивности рамок.

Далее распределение интенсивности панелей на поверхности тела сглаживается. По полученному скалярному полю, начиная от полюса с максимальной на данном шаге интенсивностью, осуществляется построение линий уровня с шагом Г. Вдоль этих линий уровня формируются новые петли. После создания петли сбрасываются в поток - отодвигаются от тела на некоторую величину Δ.

Сформированные таким образом петли подвергаются нормализации: элементы петель попавшие внутрь тела из-за погрешностей интегрирования заменяются на элементы петель проложенные по поверхности на удалении Δ, для обеспечения устойчивости расчета осуществляется устранение сильных изломов (т.н. шпилек) и проводится перераспределение узлов вдоль петель для того чтобы обеспечить длину отрезков равную заданному параметру a. Алгоритмы нормализации петель сформированы по эмпирическим законам, путём верификации расчетов с экспериментальными данными [5]. Величина φ определяет минимальный угол между отрезками. Если угол между отрезками меньше, то данное место считается шпилькой и узел переносится ближе для уменьшения угла до допустимой величины. В расчетах получено оптимальное значение 160°.

Давление в контрольных точках определяются аналогом интеграла Коши-Лагранжа [6], адаптированным под метод вихревых петель:

$$p(\vec{r},t) = p_\infty + \rho_\infty \left(\frac{|\vec{V}_\infty|^2}{2} - \frac{|\vec{V}(\vec{r},t)|^2}{2} - \frac{1}{4\pi\,\Delta t} \cdot \sum_{i=1}^{N_p} A_i(\gamma_i - \gamma_i^{*-1}) + \sum_{k=1}^{K}\sum_{i=1}^{N_k} \Gamma \vec{V}_{k\,i}(\vec{r}) \cdot \vec{V}(\vec{r}_{k\,i},t) \right),$$

где A_i – телесный угол под которым видна панель из точки вычисления давления.

На следующем этапе расчёта определяются скорости в узлах вихревых петель. Численное интегрирование движения узлов по траекториям жидких частиц проводится методом Эйлера первого порядка в заданным шагом по времени dt. Для компенсации погрешностей интегрирования после перемещения узлов снова проводится коррекция фрагментов петель, пересекающих тело и нормализация петель.

В конце шага определяются близко расположенные участки петель. При близком расположении (ближе, чем величина μ) противоположно направленных участков петель осуществляется их перезамыкание. В результате петли могут объединяться или разделяться на части. Процедура перезамыкания сопровождается нормализацией петель и производится в цикле до тех пор, пока не будут обработаны все петли.

Описанный алгоритм реализован в виде программы MVortexLoops на языке C++ с использованием библиотеки MPI для распараллеливания вычислений. На разработанную программу получено свидетельство о государственной регистрации №2017616752. Возможность параллельных вычислений была реализована для наиболее трудоёмких процедур: определения скоростей; нормализации петель; коррекции элементов петель, пересекающих тело; перезамыкания петель.

Решение СЛАУ не распараллеливалось, поскольку перемещение и деформация тел на данном этапе не моделировалось и обращения матрицы системы проводилось лишь на первом шаге расчёта обтекания тел.

3.Результаты моделирования

На первом этапе верификации производилось моделирование обтекания крыла конечного размаха (длина крыла 0,5, хорды =0,1), имеющего профиль NACA-2217 и удлинение 5,0 потоком с единичной скоростью $\vec{V}(\vec{r},t)=\{1{,}0{,}0\}$ при различных углах атаки.

Поверхность модели разбита на четырехугольные и треугольные панели. Общее число панелей поверхности тела равно N = 5212. Разбиение торца профиля на панели приведено на рис. 2. Видно, что задняя кромка затуплена, для уменьшения взаимного влияния панелей на верхней и нижней поверхностях профиля, а дискретизация профиля в передней части является достаточно грубой, что связано с необходимостью соблюдать одинаковый размер ячеек для всей модели. Верхняя и нижняя поверхности разбиты на прямоугольные рамки. При этом ширина рамок соответствует стороне граничных треугольников на торцах крыла.

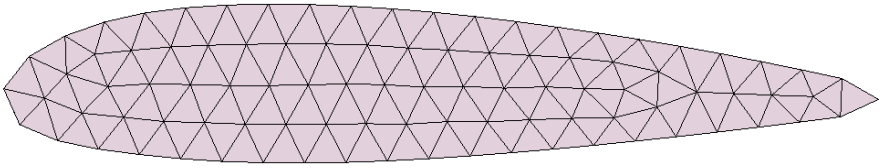


Рис. 2. Разбиение боковой поверхности профиля на панели
Fig. 2. Wing lateral surface partition

В расчетах радиус вихревого элемента взят равным ε=0,002, шаг по времени Δt=0,00219, длина отрезка a=0,001, интенсивность петель Γ=0.00526, параметры алгоритма расчета эволюции петель $\Delta=0.0001$, $\varphi=160°$, $\mu=0.04$.

На рис. 3 приведены фазы самоорганизации за крылом вихревых жгутов из вихревых петель (красные линии): рождение и унос разгонного вихря и формирование за крылом П-образной системы Прандтля. Также на рисунке видна присоединенная к крылу завихренность в виде системы вихревых петель, а также отдельные вихревые петли, формирующие тонкую вихревую пелену за острой кромкой. Таким образом в расчетах с использованием

вихревых петель получена качественно верная картина формирования вихревого следа.

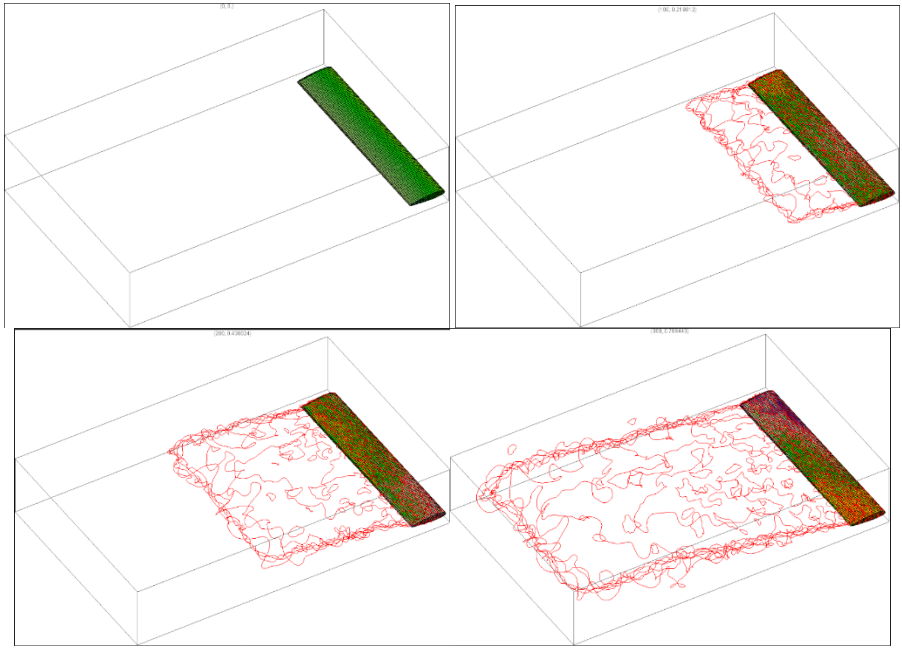


Рис. 3. Развитие вихревого следа за крылом
Fig. 3. Vortex wake evolution behind the wing

На рис. 4 приведено сравнение рассчитанных в скоростной системе координат аэродинамических коэффициентов с экспериментальными данными [7]. Коэффициенты сил и момента вычисляются качественно верно, ошибка в определении подъемной силы и момента не превышают 8%, что допустимо для проведения инженерных расчётов. Относительная ошибка лобового сопротивления велика в силу малости величины лобового сопротивления профиля, однако абсолютное значение погрешности также допустимо для инженерных расчетов. Причина завышенного значения лобового сопротивления и заниженного значения подъемной силы, возможно, заключена в грубости модели профиля, примененного для расчетов. Здесь требуются дополнительные методические исследования. На втором этапе верификации было проведено моделирование обтекания сферы единичного диаметра. Поверхность сферы разбита на N=1917 четырехугольных панели. Вектор скорости набегающего потока был единичным $\vec{V}(\vec{r},t) = \{1,0,0\}$. Всего было выполнено около 520 расчетов. При этом варьировались параметры расчетной схемы: шаг по времени dt, радиус сглаживания скорости вблизи

петли ε, дистанция перезамыкания петель μ, длина отрезка a, дистанция от поверхности тела до узлов петель при создании Δ, интенсивность петель Γ.

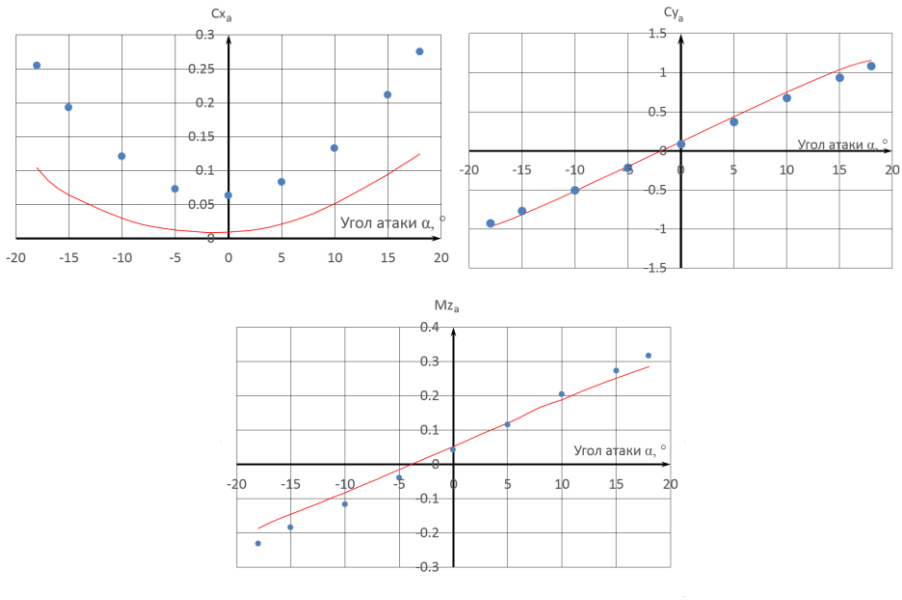


Рис. 4. Аэродинамические коэффициенты. Вычисленные значения – точки, данные экспериментов - линии [7]
Fig. 4. Aerodynamics coefficient. Calculated value – points, experimental data – lines [7]

Результаты расчетов сравнивались с результатами распределения коэффициента давления, экспериментально полученными для 4 значений числа Рейнольдса вблизи кризиса сопротивления сферы, как показано на рис.5.

При расчетах обтекания сферы использовался кластер 10Р МСЦ РАН. Каждый запуск программы проводился на 16 ядрах, а длительность счета соответствовала 960 минутам. Периоды вычисленного переходного режима (количество сделанных шагов) во всех расчетах различны и зависят от конкретных расчетных параметров.

Критерием качества итеративного подбора параметров расчетной схемы являлось среднеквадратичное отклонение точек №1,2,3,4 (см. Рис.5) в плоскости OXY значений коэффициента давления, полученных при расчете, от экспериментально полученных значений при соответствующих числах Рейнольдса. В результате для каждого режима обтекания получены наилучшие сочетания параметров, представленные в табл. 1. В таблице первый столбец соответствует номеру расчета, далее шесть параметров расчета, T – длительность переходного режима, y1…y4 – среднеквадратичные

отклонения результатов расчета от данных эксперимента. Распределения коэффициента давления для найденных сочетаний параметров показаны на рис. 6.

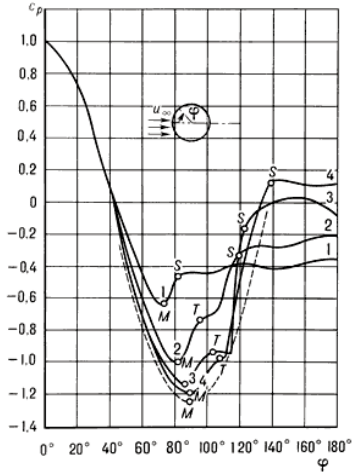


Рис. 5. Распределение коэффициента давления вдоль образующей шара 1- Re=157200, c_x=0,471; 2- Re=251300, c_x=0,313; 3- Re=298500, c_x=0,151; 4- Re=424500, c_x=0,143 [8]
Fig. 5. The distribution of the pressure coefficient along the generatrix of the sphere 1- Re=157200, c_x=0,471; 2- Re=251300, c_x=0,313; 3- Re=298500, c_x=0,151; 4- Re=424500, c_x=0,143 [8].

Результаты (см табл. 1) показывают, что разработанная методика позволяет моделировать гидродинамические нагрузки при больших числах Рейнольдса, когда влияние сил вязкости мало по сравнению с влиянием сил инерции в жидкости. При этом шаг по времени примерно соответствует отношению среднего линейного размера панели к скорости невозмущенного набегающего потока.

Данные рекомендации по выбору расчетных параметров соответствуют рекомендациям при расчёте методом вихревых элементов. При меньших значениях числа Рейнольдса количественно схожие результаты получаются при увеличении шага по времени, увеличении радиуса сглаживания и увеличении дистанции отодвигания петель от поверхности тела при рождении.

Табл. 1 Параметры и результаты моделирования обтекания сферы
Table 1 Parameters and simulation results sphere in the flow

NN	ε	dt	a	μ	Δ	Γ	T	y1	y2	y3	y4
482	0.0087	0.23	0.102	0.06	0.042	0.01	395	0.049	0.148	0.293	0.343

521	0.0005	0.10	0.052	0.12	0.016	0.01	195	0.1539	0.0631	0.168	0.214
550	0.0035	0.027	0.02	0.08	0.00125	0.02	21.6	0.287	0.150	0.050	0.076
309	0.0035	0.025	0.02	0.08	0.00125	0.02	28.2	0.3477	0.2109	0.090	0.046

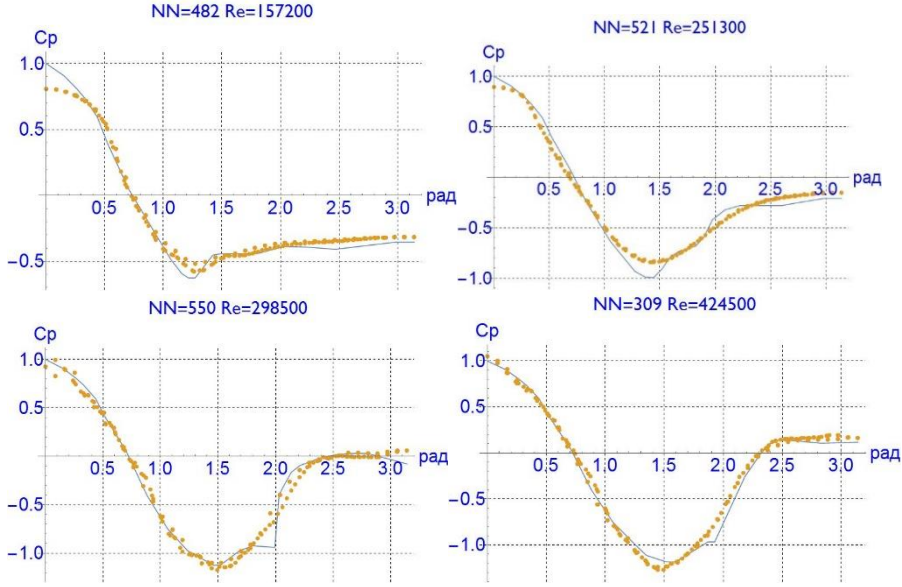


Рис. 6. Сравнение распределения коэффициента давления, полученного в расчете(точки) и экспериментально (линии)
Fig. 6. Comparison of the pressure coefficient distribution obtained in the calculation (points) and experimentally (lines)

Следует учесть, что в разработанной методики влияние вязкости на взаимодействие между петлями не учитывается. Для корректного моделирования влияния вязкости на процессы рождения и взаимодействия вихревых петель методика должна быть доработана, например, с использованием модели обмена интенсивностями между вортонами. Однако, поскольку непосредственной целью разработанной методики является моделирование гидродинамических нагрузок при больших числах Рейнольдса (около $7 \cdot 10^6$), методика может быть использована без доработок.

4. Заключение

Проведенная верификация разработанных моделей, алгоритмов и программы MVortexLoops по известным экспериментальным данным показала, что использование замкнутых вихревых петель в качестве вихревого «суперэлемента» позволяет качественно верно моделировать нестационарное

отрывное обтекание крыльев и гладких тел. При этом количественные значения аэродинамических коэффициентов и распределения давления могут быть найдены с приемлемой для инженерных приложений погрешностью при соответствующем выборе параметров эмпирических моделей.

Применение разработанной модификации метода вихревых элементов позволяет, с одной стороны, не задавать до начала расчета линии отрыва вихревых пелен, и, с другой стороны, позволяет исключить вредное влияние дополнительной завихренности вортонов-отрезков. Дальнейшее развитие будет направлено на моделирование гидродинамических нагрузок на подвижные деформируемые тела.

Работа выполнена с использованием средств Межведомственного суперкомпьютерного центра Российской академии наук.

Список литературы

[1]. Трехмерное отрывное обтекание тел произвольной формы. Под ред. С.М. Белоцерковского. М.: ЦАГИ, 2000, 265 с.

[2]. Щеглов Г.А. Модификация метода вихревых элементов для расчета гидродинамических характеристик гладких тел. Вестник МГТУ им Н.Э. Баумана. Сер. Машиностроение, 2009, №2, с. 26-35

[3]. Chorin, A.J. Hairpin removal in vortex interactions II, Jour. of Comput. Phys., 1993. №107, . p. 1-9,

[4]. WeibmannS., Pinkall U., Filament-based smoke with vortex shedding and variational reconnection. ACM Transactions on Graphics, 2010, Vol. 29, No. 4, Article 115. DOI: 10.1145/1778765.1778852

[5]. Дергачев С.А., Щеглов Г.А.. Моделирование обтекания тел методом вихревых элементов с использованием замкнутых вихревых петель. Научный вестник МГТУ ГА, 2016, № 223(1), с.19-25 DOI: 10.26467/2079-0619-2016--223-19-27

[6]. Андронов П.Р., Гувернюк С.В., Дынникова Г.Я. Вихревые методы расчета нестационарных гидродинамических нагрузок. М.: Изд-во Моск. ун-та, 2006, 184 с.

[7]. Б.А. Ушаков. Атлас аэродинамических профилей крыльев. Издание БНТ НКАП при ЦАГИ, 1940, 84 с.

[8]. Авиация: Энциклопедия. Гл. ред. Г.П. Свищёв М.: Большая Российская энциклопедия, 1994. 736 с.

Mathematical simulation of vorticity evolution in the case of spatial flow around bodies by the method of vortex loops

S.A. Dergachev <sadergachev@mail.ru>
Bauman Moscow State Technical University,
5/1, 2-nd Baumanskaya st., Moscow. Russia, 105005

Abstract. Simulation of hydrodynamic phenomena associated with flow around movable deformable bodies is an actual engineering task of various technical problems. This article describes the method of vortex loops, which allows to simulate the flow around bodies

without the need to preset the lines of vorticity retreat. Verification of the technique on the experimental data on the flow past the wing and the sphere is given. The accuracy is shown sufficient for application of the program and methodology in engineering applications. Conclusions are made about the possibility of transition to mobile and deformable bodies. The program is implemented in C ++ with the use of the MPI library to organize the transfer of data in parallel calculations.

Keywords: hydrodynamics; non-stationary loads; vortex loops; vortices; method of vortex loops.

DOI: 10.15514/ISPRAS-2018-30(1)-14

For citation: Dergachev S.A. Mathematical simulation of vorticity evolution in the case of spatial flow around bodies by the method of vortex loops. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 1, 2018, pp. 215-226 (in Russian). DOI: 10.15514/ISPRAS-2018-30(1)-14

References

[1]. Three-dimensional detached flow around bodies of arbitrary shape. S.M. Belotserkovsky Ed.. Moscow: TsAGI, 2000. - 265 p. (in Russian)

[2]. Shcheglov G.A. Modification of the vortex element method for calculating the hydrodynamic characteristics of smooth bodies. Vestnik MGTU im N.E. Baumana. Ser. Mashinostroenie [Herald of the Bauman Moscow State Technical University. Ser. Mechanical engineering]. 2009. № 2. p. 26-35 (in Russian)

[3]. Chorin, A.J. Hairpin removal in vortex interactions II. Jour. of Comput. Phys., 1993, №107, P. 1-9.

[4]. WeibmannS., Pinkall U., Filament-based smoke with vortex shedding and variational reconnection. ACM Transactions on Graphics, 2010. Vol. 29, No. 4, Article 115

[5]. Dergachev S.A., Scheglov G.A. Simulation of flow around the bodies by means of the method of vortex elements with use of closed vortex loops. Nauchny vestnik MGTU GA [Scientific Herald of the MGTU GA], 2016. No. 223(1), p.19-25. (in Russian). DOI: 10.26467/2079-0619-2016--223-19-27

[6]. Andronov P.R., Gouvernyuk S.V., Dynnikova G.Ya. Vortex methods for calculating unsteady hydrodynamic loads. Moscow: Izd-vo Mosk. University, 2006, 184 p. (in Russian)

[7]. Ushakov B.A. Atlas of airfoils. Edition of the BNT NCAP at TsAGI, 1940, 84 p. (in Russian)

[8]. Aviation: Encyclopedia. G.P. Svishchev Ed. Moscow: The Great Russian Encyclopedia, 1994, 736 p. (in Russian)