DOI: 10.15514/ISPRAS-2022-34(1)-9

Алгоритмы планирования вычислений с учетом избыточности и неопределенности

[1] *А.Г. Феоктистов, ORCID: 0000-0002-9127-6162 <agf65@icc.ru>*
[1] *Р.О. Костромин, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>*
[1] *С.А. Горский, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>*
[1] *И.В. Бычков, ORCID: 0000-0002-1765-0769 <bychkov@icc.ru>*
[2,3,4] *А.Н. Черных, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[1,5] *О.Ю. Башарина, ORCID: 0000-0002-7151-782X <basharinaolga@mail.ru>*

[1] *Институт динамики систем и теории управления им. В.М. Матросова СО РАН, 664033, Россия, Иркутск, ул. Лермонтова, 134, а/я 29*
[2] *Центр научных исследований и высшего образования Мексика, 22860, Нижняя Калифорния, Энсенада, ш. Тихуана-Энсенада, 3918*
[3] *Южно-Уральский государственный университет, 454080, Россия, Челябинск, проспект Ленина, 76*
[4] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, Москва, ул. А. Солженицына, 25*
[5] *Иркутский государственный университет, 664003, Иркутск, ул. Карла Маркса, 1, Россия*

Аннотация. В статье предложены новая модель и алгоритмы планирования вычислений в пакетах программ. Их отличительной особенностью является использование времени выполнения модулей пакетов на ресурсах среды в процессе их непрерывной интеграции, доставки и развертывания. Предложенные модель и алгоритмы позволяют строить избыточные схемы решения задачи. На практике избыточность схемы позволяет адаптировать ее к изменяющимся характеристикам ресурсов среды и повысить надежность вычислений. Построение схем решения задач показано на модельных примерах.

Ключевые слова: распределенные вычисления; научные приложения; схема решения задачи; планирование вычислений; избыточность; неопределенность

Для цитирования: Феоктистов А.Г., Костромин Р.О., Горский С.А., Бычков И.В., Черных А.Н., Башарина О.Ю. Алгоритмы планирования вычислений с учетом избыточности и неопределенности. Труды ИСП РАН, том 34, вып. 1, 2022 г., стр. 123-140. DOI: 10.15514/ISPRAS–2022–34(1)–9

Благодарности: Исследование выполнено при поддержке РФФИ, проект № 19-07-00097-а. Имитационная модель среды разработана при частичной поддержке Министерства науки и высшего образования Российской Федерации, проект «Технологии разработки и анализа предметно-ориентированных интеллектуальных систем группового управления в недетерминированных распределенных средах».

Redundancy and Uncertainty-Based Algorithms for Computation Planning

[1] *A.G. Feoktistov, ORCID: 0000-0002-9127-6162 <agf@icc.ru>*
[1] *R.O. Kostromin, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>*
[1] *S.A. Gorsky, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>*
[1] *I.V. Bychkov, ORCID: 0000-0002-1765-0769 <bychkov@icc.ru>*
[2,3,4] *A.N. Tchernykh, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[1,5] *O.Yu. Basharina, ORCID: 0000-0002-7151-782X <basharinaolga@mail.ru>*

[1] *Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the RAS, 134, Lermontov st., Irkutsk, postbox 292, 664033, Russia*
[2] *Centro de Investigación Científica y de Educación Superior, 3918, Ensenada-Tijuana Highway, Ensenada, 22860, Mexico*
[3] *South Ural State University, Chelyabinsk, 76, Lenin prospekt, Chelyabinsk, 454080, Russia*
[4] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[5] *Irkutsk State University, 1, KarlMarks St., Irkutsk, 664003, Russia*

Abstract. Nowadays, the development and use of workflow-based applications (distributed applied software packages) are some of the key challenges in terms of preparing and carrying out large-scale scientific experiments in distributed environments with heterogeneous computing resources. The environment resources can be represented by clusters of personal computers, supercomputers, and private or public cloud platforms and differ in their computational characteristics. Moreover, the composition and characteristics of resources change in dynamics. Therefore, computations planning and resource allocation in the considered environments are important problems. In this regard, we propose new algorithms for computation planning taking into account redundancy and uncertainty in such distributed applied software packages. Compared to other algorithms of a similar purpose, the proposed algorithms use evaluations of workflow execution makespan obtained in the process of continuous integration, delivery, and deployment of applied software. The proposed algorithms provide the construction of redundant problem-solving schemes that allow us to adapt them to the dynamic characteristics of computational resources and improve distributed computing reliability. The algorithms are based on a theory of conceptual modeling computational processes. We demonstrate the process of constructing problem-solving schemes on model examples. In addition, we show the utility in using redundancy for increasing the distributed computing reliability In comparison with some traditional meta-schedulers.

Keywords: distributed computing; scientific applications; workflow; computation planning; redundancy; uncertainty

For citation: Feoktistov A.G., Kostromin R.O., Gorsky S.A., Bychkov I.V., Tchernykh A.N., Basharina O.Yu. Redundancy and Uncertainty-Based Algorithms for Computation Planning. Trudy ISP RAN/Proc. ISP RAS, vol. 34, issue 1, 2022, pp. 123-140 (in Russian). DOI: 10.15514/ISPRAS–2022–34(1)–9

Acknowledgements. The study is supported by the Russian Foundation of Basic Research, project no. 19-07-00097. Developing the simulation model of the computing environment was supported in part by the Ministry of Science and Higher Education of the Russian Federation, project «Technologies for the development and analysis of subject-oriented intelligent group control systems in non-deterministic distributed environments».

1. Введение

Распределенные вычисления представляют собой эффективный подход к решению ресурсоемких вычислительных задач посредством использования множества вычислительных устройств, которые в общем случае являются разнородными по своим вычислительным и программно-аппаратным характеристикам. При этом распределенные вычисления можно считать высокопроизводительными, если они существенно ускоряют процесс решения задачи. В данном случае, термин *распределенные вычисления* определяет

весь спектр вопросов, связанных с планированием и распределением ресурсов вычислительной среды, а также управлением ими с целью достижения максимальной эффективности решения научных и прикладных задач.

Для таких сред мы разрабатываем масштабируемые научные и прикладные приложения – распределенные пакеты прикладных программ (РППП), модули программного обеспечения (ПО) которых могут выполняться на ресурсах среды с различными вычислительными характеристиками. Как правило, такие пакеты создаются на основе модульного подхода. Модули представляют собой прикладные программы или сервисы, снабженные спецификациями, которые отражают информацию о назначении, способах применения, форматах входных и выходных параметров, а также другие необходимые для вычислительного процесса сведения. Процесс решения задачи в РППП основывается на проведении многовариантных расчетов [1].

В рамках проблемы управления распределенными вычислениями выделяются две взаимосвязанные задачи [2]:

- планирование вычислений (построение схемы решения задачи, определяющей порядок выполнения модулей пакета);
- управление ими на ресурсах вычислительной среды, назначенных для выполнения прикладного программного обеспечения (модулей).

Статья посвящена вопросам, относящимся к решению первой задачи. Разработка и применение подхода к решению второй задачи детально рассмотрены в [3]. Средства для решения обеих задач реализуются в рамках инструментального комплекса Orlando Tools (OT) [4].

2. Обзор связанных работ

Автоматическое планирование взаимодействия компонентов научного рабочего процесса является сложной задачей, поскольку трудно определить их функциональность и семантику используемых ими данных [5]. В частности, необходимы разработка и применение специальных средств описания и использования алгоритмических и схемных знаний о предметной области научного рабочего процесса. Например, в качестве фундаментальной основы представления таких знаний может быть задействовано концептуальное моделирование [6].

Поэтому в системах поддержки научных рабочих процессов (workflow) наиболее распространены подходы, основанные на процедурной форме построения таких процессов [7]. Среди них Askalon [8], Condor DAGMan [9], GridAnt [10], GridWay [11], Kepler [12], Taverna [13], Triana [14] и UNICORE [15].

Наиболее популярными из них являются следующие подходы:

- применение универсальных языков разметки, таких, как XML [16], и также их расширений (например, XML Process Definition Language [17] для описания рабочих процессов, относящихся к любым сферам деятельности, и Business Process Modeling Language [18] для описания рабочих процессов различного типа);
- использование специальных форматов описания заданий по выполнению научных рабочих процессов (например, форматы заданий в Condor DAGMan и GridWay);
- графическое представление научных рабочих процессов в виде ориентированных циклических или ациклических графов (например, сети Петри [19] и UML диаграммы [20]).

Однако при увеличении числа компонентов научных рабочих процессов и их параметров восприятие таких рабочих процессов с помощью вышеупомянутых подходов существенно затрудняется. Это обусловлено ограниченностью области экрана компьютера, на котором отражается описание научного рабочего процесса. Во многих случаях разработчики научных

рабочих процессов и их конечные пользователи видят только отдельные фрагменты описаний и зачастую не могут проследить связи с другими фрагментами, находящимися за пределами экрана. Отчасти это нивелируется использованием иерархических форм представления научных рабочих процессов.

В этой связи наиболее продвинутые системы поддержки научных рабочих процессов (например, OT и Pegasus [21]) включают средства автоматического планирования взаимодействия компонентов приложений. Такие системы более предпочтительны для рабочих процессов, которые характеризуются большим числом своих компонентов и взаимосвязей между ними [22].

Системы GridAnt и UNICORE поддерживают конкретные научные рабочие процессы. Для таких научных рабочих процессов используемое прикладное ПО их компонентов и источники данных полностью или частично предопределены. Это зачастую обусловлено необходимостью использования конкретных сенсоров, управляющего оборудования, непереносимого ПО и др.

В сравнении с ними системы Askalon, Condor DAGMan, GridWay, Kepler, Pegasus и Triana ориентированы на абстрактные научные рабочие процессы. Компоненты таких научных рабочих процессов не связаны с конкретными источниками данных, прикладным ПО и ресурсами. Они могут запускаться на ресурсах, назначаемых для их выполнения статически или динамически локальными менеджерами ресурсов или метапланировщиками среды.

Дополнительным преимуществом систем OT и Taverna является то, что они поддерживают оба типа научных рабочих процессов. Это обеспечивает гибкость в выборе средств построения научных рабочих процессов.

Система поддержки научного рабочего процесса является частным случаем пакета прикладных программ. В отличие от научного рабочего процесса пакет реализует решение целого класса задач в конкретной предметной области. Решение задач осуществляется на модели предметной области (вычислительной модели). Такая модель формализует описание исследуемой системы, ее объектов, их взаимосвязей, а также происходящих в ней событий и процессов. В рамках направления исследований, связанного с разработкой и применением пакетов, разработан ряд известных подходов к автоматическому синтезу абстрактных программ на вычислительных моделях.

Работа [23] посвящена дедуктивному методу синтеза программ. Этот метод используется на статическом этапе построения программы для композиционного программирования в целом. Вопросы, связанные с избыточностью и неопределенностью, а также назначением ресурсов, требуют дополнительного развития.

Новый подход к построению параллельных асинхронных абстрактных программ требуемой длины представлен в [24]. Система булевых ограничений используется в качестве постановки задачи. В том числе учитываются ограничения на число доступных процессоров однородных узлов вычислительной среды и модельные временные задержки при выполнении модулей абстрактной программы.

В [25] рассмотрена проблема синтеза параллельных рекурсивных программ по непроцедурной постановке задачи. Предложено специальное исчисление для ее решения.

Система фрагментированного программирования LuNA, предназначенная для разработки библиотек параллельных численных программ, представлена в [26]. Она базируется на использовании вычислительной модели и ориентируется на высокопроизводительный однородный кластер [27].

В отличие от рассмотренных выше работ, использующих вычислительные модели, мы предлагаем модель и алгоритмы планирования вычислений в разнородной распределенной среде с учетом реального времени выполнения модулей, полученного во время их тестирования на ресурсах среды [28]. Представлены специализированные структуры данных,

необходимые для реализации данных моделей и алгоритмов. Приведены модельные примеры, иллюстрирующие применение предложенных моделей и алгоритмов.

3. Модель

Структура модели. Представим вычислительную модель РППП следующей структурой:

$$M_{DASP} = \langle PAR, O, M, N, R^{in}, R^{out}, \tilde{R}, \tilde{\tilde{R}}, PR, NPPF, NPPF', S\rangle, \quad (1)$$

где PAR – множество параметров пакета (обрабатываемых данных), O – множество вычислительных операций над полем параметров, M – множество модулей, являющихся конкретными программными реализациями абстрактных операций, N – множество разнородных ресурсов распределенной вычислительной среды, на которых размещены модули. Каждая операция $o_i \in O$ интерпретируется как абстрактный процесс вычисления множества ее выходных параметров $PAR_i^{out} \subset PAR$ по множеству ее входных параметров $PAR_i^{in} \subset PAR$, $PAR_i^{in} \cap PAR_i^{out} = \emptyset$.

Связь между параметрами и операциями задается соответственно булевыми матрицами R^{in} и R^{out} размерности $n_o \times n_p$. Элемент матрицы $r_{ij}^{in} = 1$ ($r_{ij}^{in} = 0$) указывает, что j-й параметр является (не является) входным для i-й операции, $i = \overline{1, n_o}$, $j = \overline{1, n_p}$. Аналогично, элемент матрицы $r_{ij}^{out} = 1$ ($r_{ij}^{out} = 0$) указывает, что j-й параметр является (не является) выходным для i-й операции.

Связь между модулями и операциями представлена булевой матрицей $\tilde{R}$ размерности $n_m \times n_o$. Элемент матрицы $\tilde{r}_{ij} = 1$ ($\tilde{r}_{ij} = 0$) определяет, что i-й модуль реализует (не реализует) j-ю операцию, $i = \overline{1, n_m}$, $j = \overline{1, n_o}$. Эта связь имеет тип «один-ко-многим». Поэтому на нее накладывается следующее ограничение целостности:

$$\bigvee_{j=1}^{n_o}\left(f_j \vee g_j\right) = 0, \quad f_j = \bigvee_{i=1}^{n_m - 1}\bigvee_{k=i+1}^{n_m}\left(\tilde{r}_{ij} \wedge \tilde{r}_{kj}\right), \quad g_j = \bigwedge_{i=1}^{n_m}\bar{\tilde{r}}_{ij}. \quad (2)$$

Ограничение (2) определяет, что одна операция может быть реализована только одним модулем. В тоже самое время, один модуль может реализовывать несколько операций.

Связь между ресурсами и модулями представлена булевой матрицей $\tilde{\tilde{R}}$ размерности $n_n \times n_m$. Элемент матрицы $\tilde{\tilde{r}}_{ij} = 1$ ($\tilde{\tilde{r}}_{ij} = 0$) определяет, что j-й модуль может быть выполнен на i-м ресурсе, $i = \overline{1, n_n}$, $j = \overline{1, n_m}$. Эта связь имеет тип «многие-ко-многим». Она означает, что один модуль может выполняться на разных ресурсах. В тоже самое время, несколько разных модулей могут быть размещены на одном и том же ресурсе.

Схема решения задачи. Для представления схемы решения задачи (ярусно-параллельная форма алгоритма решения задачи) будем использовать булеву матрицу S размерности $n_t \times n_o$, где n_t – это число ярусов схемы. Элемент матрицы $s_{ij} = 1$ ($s_{ij} = 0$) означает, что j-я операция размещена (не размещена) на i-м ярусе схемы, $i = \overline{1, n_t}$, $j = \overline{1, n_o}$. Матрица U размерности $n_o \times n_o$ представляет сведения о предшествовании операций при выполнении схемы S. Элемент $u_{ij} = 1$ ($u_{ij} = 0$) означает, что завершение операции o_j предшествует (не предшествует) выполнению операции o_i и $PAR_i^{in} \cap PAR_j^{out} \neq \emptyset$.

Непроцедурная постановка задачи. Схема S строится по непроцедурной постановке задачи $NPPF$. В рамках такой постановки задачи задаются множество $S^{in} \subset PAR$ исходных параметров, значения которых известны, и множество $S^{out} \subset PAR$ целевых параметров, значения которых должны быть вычислены. На основе связи между параметрами и операциями требуется определить множество $S^o \subset O$ операций, которые нужно выполнить, чтобы вычислить значения целевых параметров. Таким образом, непроцедурную постановку задачи $NPPF$ можно определить следующим образом:

$$NPPF: S^{in}, S^{out} \rightarrow S^o?, \quad (3)$$

где символ ‘?’ означает, что соответствующее множество не определено.

Назначение ресурсов. Как было отмечено выше, в данном контексте понятие схемы решения задачи соответствует абстрактному научному рабочему процессу. Назначение ресурсов для выполнения модулей, реализующих операции схемы S, осуществляется в процессе распределения вычислительных заданий с помощью мультиагентной системы (МАС) в рамках тендера вычислительных работ, детально рассмотренного в [29-31].

Таким образом, в результате проведения вышеупомянутого тендера мы получаем номер ресурса j для каждой i-й операции схемы S. На основе данной привязки определяются оценки времени (E_s^{asynch}), стоимости (E_s^{cost}) и надежности (E_s^r) выполнения операций схемы решения задачи.

Оценка времени выполнения схемы. Для оценки времени выполнения заданий мы используем модели, рассмотренные в [32]. Здесь мы приводим формулу вычисления такой оценки, обобщенную и уточненную для асинхронного режима выполнения схемы S с учетом возможных запусков (удалений) виртуальных машин и возникновений отказов узлов, системного и прикладного ПО.

Пусть матрица D размерности $n_s \times n_s$ отражает оценки размера данных, передаваемых между операциями, n_s – число операций схемы. Элемент матрицы $d_{ij} \geq 0$ показывает размер данных, передаваемых операцией o_i операции o_j. Матрица W размерности $n_s \times n_s$ представляет пропускную способность интерконекта между ресурсами, на которых будут запущены модули, реализующие операции. Элемент матрицы $w_{ij} \geq 0$ показывает пропускную способность интерконекта между ресурсами с модулями, реализующими операции o_i и o_j.

Оценка E_s^{asynch} времени выполнения схемы в асинхронном режиме вычисляется по следующей формуле:

$$E_s^{\text{asynch}} = \max_{i=1,n_s} e_i^\tau,$$

$$e_i^\tau = e_i^q + e_i^{vml} + e_i^{run} + e_i^{res} + e_i^{vmr} + \max_{\forall j \in \overline{1,n_s}: u_{ij}=1, i \neq j}\left(e_j^\tau + e^{dtr}\right),$$

$$e^{dtr} = \frac{d_{ji}}{c_{ji}(t) w_{ji}},$$

где переменные интерпретируются следующим образом:

- e_i^τ (e_j^τ) – оценка периода времени τ с начала выполнения схемы и до завершения операции o_i (o_j);
- $e_i^q \geq 0$ – оценка времени ожидания модуля, реализующего операцию o_i в очереди локального менеджера ресурсов;
- $e_i^{run} > 0$ – оценка времени выполнения модуля до его успешного завершения или отказа;
- $e_i^{res} \geq 0$ – оценка времени на рестарт модуля в случае возникновения отказа какого-либо типа;
- $e_i^{vml} \geq 0$ и $e_i^{vmr} \geq 0$ – экспериментально определяемые оценки времени, необходимого на запуск и завершение виртуальных машин;
- $0 < c_{ji}(t) \leq 1$ – коэффициент уменьшения пропускной способности интерконнекта в момент времени t между ресурсами, связанными с выполнением операций o_i и o_j.

Оценка стоимости выполнения задания. Данная оценка определяется по следующей формуле:

$$E_s^{cost} = \sum_{i=1}^{n_s} \pi_i e_i^{run} + \pi_i^*,$$

где $\pi_i \geq 0$ – себестоимость единицы процессорного времени ресурса, $\pi_i^* \geq 0$ – торговая наценка, закладываемая провайдером данного ресурса. В частности, для пользователей

суперкомпьютерных центров коллективного пользования или приватных облачных платформ как π_i, так и π_i^* могут быть равны 0.

Оценка надежности. Получение оценки надежности выполнения схемы решения задачи базируется на использовании методов логико-вероятностного анализа, предложенных И.А. Рябининым применительно к исследованию сложных технических систем [33]. Введем следующие обозначения:

- x – это набор булевых переменных (параметров) x_{ij_i}, отражающих события завершения ($x_{ij_i} = 1$) или незавершения ($x_{ij_i} = 0$) операции o_i на j_i-м ресурсе, $i \in \overline{1, n_o}$, $j_i \in \overline{1, n_r}$;
- y_i – булева функция, определяющая надежность выполнения i-й операции на j_i-м ресурсе;
- $p_{ij_i}(t)$ – вероятность выполнения операции o_i на j_i-м ресурсе в момент времени t (стационарный коэффициент готовности, характеризующий j_i-й ресурс в произвольный момент времени t по статистическим результатам его работы).

Функция y_i определяется следующей формулой:

$$y_i = \begin{cases} x_{ij_i}, & \text{если } \exists k: u_{ik} = 1,\ i \neq k, k \in \overline{1, n_o}, \\ x_{ij_i} \wedge z_i, & \text{в противном случае,} \end{cases} \qquad z_i = \bigwedge_{\forall k: w_{ik}=1, i \neq k} y_k. \tag{4}$$

В результате выполнения всех подстановок в соответствии с (3) и учетом информации о предшествовании операций, представленной матрицей U, мы получим булеву функцию

$$q_s = \bigwedge_{i: s_{li}=1} x_{ij_i},\ \ l \in \overline{1, n_l}, \tag{5}$$

определяющую надежность выполнения схемы S. Для получения вероятностного показателя надежности схемы S производится переход от функции (5) к вероятностной функции

$$P(t) = \prod_{i: s_{li}=1} p_{ij_i}(t) \tag{6}$$

с помощью соответствующих правил преобразования [34]. Функция (6) вычисляет вероятность (оценку надежности E_s^r) выполнения схемы S без резервирования ресурсов в момент времени t в процессе планирования. Определение вероятности выполнения схемы S с резервированием ресурсов рассмотрено в [34]. Однако резервирование зачастую приводит к снижению эффективности использования ресурсов. Поэтому в данной работе предложен подход к определению вероятности выполнения схемы S без резервирования дополнительных ресурсов.

Избыточность. В модели (1) может наблюдаться избыточность относительно вычисления одних и тех же параметров разными операциями, т.е. могут существовать такие i и j, что $PAR_i^{out} \cap PAR_j^{out} \neq \emptyset,\ i, j \in \overline{1, n_l}$. Поэтому схема S в общем случае может содержать несколько альтернативных сценариев решения задачи.

Модель (1) не имеет избыточности относительно вычисления одних и тех же параметров разными операциями, если выполняется следующее условие:

$$\bigvee_{j=1}^{n_p} (f_j \vee g_j) = 0,\ \ f_j = \bigvee_{i=1}^{n_o - 1} \bigvee_{k=i+1}^{n_o} (r_{ij}^{out} \wedge r_{kj}^{out}),\ \ g_j = \bigwedge_{i=1}^{n_0} r_{ij}^{out}. \tag{7}$$

С целью обеспечения возможности устранения избыточности в процессе построения схемы решения задачи операции $o_1, o_2, \dots, o_{n_o} \in O$ снабжаются приоритетами $pr_{1j}, pr_{2j}, \dots, pr_{n_o j}$ относительно данного j-го параметра. Приоритеты операций автоматически определяются следующим образом:

$$pr_{ij} = \begin{cases} 0, & i \in \overline{1, n_o},\quad j \in \overline{1, n_p}: r_{ij}^{out} = 0, \\ \dfrac{j}{i}, & \forall i \in \overline{1, n_o},\ \ j \in \overline{1, n_p}: r_{ij}^{out} = 1. \end{cases} \tag{8}$$

Очевидно, что чем меньше номер операции i, тем больше ее приоритет относительно j-го параметра в (8). Формируемая таким образом вещественная матрица приоритетов PR может быть изменена в дальнейшем разработчиком или конечным пользователем пакета вручную путем изменения мест параметров и операций в соответствующих списках параметров и операций. Такие изменения осуществляются с учетом экспертного опыта разработчиков и конечных пользователей пакета, обладающих алгоритмическими и схемными знаниями относительно операций пакета.

Новая непроцедурная постановка задачи. Для устранения избыточности мы используем следующую новую постановку задачи:

$$NPPF': S^{in}, S^{out}, PR_{j_1}, PR_{j_2}, \dots, PR_{j_l} \rightarrow S^o?, \tag{9}$$

где $j_1, j_2, \dots, j_l \in \overline{1, n_p}$.

При планировании на основе такой постановки задачи в модели (1) остаются операции, для которых выполняются условия $r_{i_k j_k}^{out} = 1$, $r_{ij_k}^{out} = 0$, где $i_k = \underset{\mu=1,|PR_k|}{\operatorname{argmax}} pr_\mu$, $i_k \in \overline{1, n_o}$, $j_k \in \overline{1, n_p}$, $i = \overline{1, n_o},\ i \neq i_k,\ k = \overline{1, l},\ l$ – число параметров, значения которых вычисляются разными операциями.

Эти операции имеют наивысшие приоритеты относительно параметров $par_{j_1}, par_{j_2}, \dots, par_{j_l} \in PAR$. При этом остальные операции с меньшими приоритетами становятся недопустимыми и временно скрываются в модели (1) в процессе планирования. Тем самым матрицы S^{in} и S^{out} специализируются в соответствии с постановкой задачи. Таким образом, обеспечивается выполнение условия (7) и построение схемы с единственным сценарием решения задачи по ее постановке (9).

В случае, когда требуется построение схемы, включающей несколько сценариев решения задачи, применяется постановка задачи (3).

В целом модель (1) является развитием вычислительной модели, представленной в [35].

4. Алгоритмы планирования вычислений

Рассмотрим алгоритмы планирования схем с одним или несколькими сценариями решения задачи в РППП на рассмотренной выше модели (1).

Псевдокод алгоритма $A_1(O, S^{in}, S^{out}, n_o \rightarrow S, n_l)$ построения схемы с несколькими альтернативными сценариями решения задачи по ее постановке (3) представлен на рис. 1. Предполагается, что для каждой j-й операции из O известны PAR_j^{in} и PAR_j^{out}. Булевы переменные C_f и A_f используются в качестве флагов. Флаг $C_f = 1$ ($C_f = 0$) означает, что схема решения задачи построена (не построена). Флаг $A_f = 1$ ($A_f = 0$) означает, что на очередной итерации планирования, хотя бы одна операция была включена (ни одна операция не была включена) в схему.

После инициализации вспомогательных переменных в цикле выполняется проверка на возможность включения каждой j-й операции в схему S. После завершения цикла оцениваются текущие результаты планирования. Проверяем, разрешима проблема или неразрешима. В первом случае, мы готовим данные для следующей итерации планирования и приступаем к ней. Во втором случае, алгоритм завершает свою работу.

Алгоритм A_1 имеет вычислительную сложность $O(n_o)$.

В общем случае в результате выполнения алгоритма A_1 на модели (1), для которой условие (7) не выполняется, мы получим схему с несколькими сценариями решения задачи, которая может также включать ненужные вычисления. С целью обеспечения возможности редукции ненужных вычислений разработан алгоритм $A_2(O, S, S^{out}, n_o, n_l \rightarrow S)$, псевдокод которого показан на рис. 2. Предполагается, что для каждой j-й операции из O известны PAR_j^{in} и PAR_j^{out}.

В процессе работы алгоритма A_2 на каждом уровне схемы для каждой j-й операции происходит проверка вычисления ею значений параметров из целевого множества S^{out*}. Если проверка выполняется, то вспомогательное множество S^{in*} объединяется со множеством P_j^{in} входных параметров операции. В противном случае, операция удаляется из схемы решения задачи. После проверки всех операций i-го уровня схемы S^{out*} объединяется с S^{in*} $\forall i > 1$.

```
1  // Инициализация вспомогательных переменных
2  O* = O; S^in* = S^in; S^out* = ∅;
3  i = 1;
4  for j = 1..C step 1 do
5    s_ij = 0;
6  end do
7  C_f = 0; A_f = 0;
8  // Проверка возможности включения операций в схему
9  for j = 1..n_o step 1 do
10   if (o_j^* = 1) ∧ (PAR_j^in = ∅ ∨ PAR_j^in ⊆ S^in*) then
11     s_ij = 1; A_f = 1; S^out* = S^out* ∪ PAR_j^out; o_j^* = 0;
12   end if
13 end do
14 // Проверка достижения целевого множества параметров
15 S^in* = S^in* ∪ S^out*;
16 if S^out ⊆ S^in* then
17 // Задача разрешима
18   C_f = 1;
19 end if
20 // Проверка разрешимости задачи для очередной итерации планирования
21 if A_f ∨ C_f = 0 then
22 // Задача неразрешима
23   return NULL;
24 else
25   if A_f ∨ C̄_f = 0 then
26 // Схема построена
27     n_l = i;
28     return < S, n_l >;
29   else
30 // Подготовка данных для очередной итерации планирования
31     S^out* = ∅;
32     i = i + 1;
33     for j = 1..n_o step 1 do
34       s_ij = 0;
35     end do
36     A_f = 0;
37 // Переход на очередную итерацию планирования
38     goto 9;
39   end if
40 end if
```

Рис. 1. Псевдокод алгоритма A_1
Fig. 1. Pseudocode of the algorithm A_1

На заключительной стадии осуществляется удаление всех i-х строк матрицы S, для которых выполняется условие $\bigvee_{j=1}^{n_o} s_{ij} = 0, n_l = n_l - n_0$, где n_0 – это число удаленных строк. Удаление строк выполняется с помощью вспомогательной подпрограммы $Delete\_rows()$.

Алгоритм A_2 имеет вычислительную сложность $O(n_l \times n_o)$.


```
1  // Инициализация переменных
2  S^in* = ∅; S^out* = S^out;
3  // Редукция ненужных вычислений
4  for i = n_l..1 step −1 do
5    for j = 1..n_o do
6      if s_ij = 1 step 1 then
7        if PAR_j^out ∩ S^out* ≠ ∅ then
8          S^in* = S^in* ∪ PAR_j^in;
9        else
10         s_ij = 0;
11       end if
12     end if
13   end do
14   if i > 1 then
15     S^out* = S^out* ∪ S^in*; S^in* = ∅;
16   end if
17 end do
18 // Удаление i-й строки матрицы S: ⋁_{j=1}^{n_o} s_ij = 0
19 Delete_rows(S, n_l → S, n_l);
20 return S;
```

Рис. 2. Псевдокод алгоритма A_2
Fig. 2. Pseudocode of the algorithm A_2

```
1  // Инициализация вспомогательных переменных
2  O* = O;
3  // Специализация множества O*
4  for k = 1..l step 1 do
5    i_k = argmax_{μ=1,|PR_k|} pr_μ;
6    for i = 1..n_o step 1 do
7      if (r_{ij_k}^out = 1) ∧ (i ≠ i_k) then
8        o_i^* = 0;
9      end if
10   end do
11 end do
12 // Специализация множеств R^in и R^out
13 Specialize(O*, R^in, R^out → R^in, R^out);
14 // Применение алгоритма A_1
15 A_1(O*, S^in, S^out, n_o → S, n_l);
16 if S ≠ NULL then
17 // Схема построена. Применение алгоритма A_2
18   A_2(O*, S, S^out, n_o, n_l → S);
19   return S;
20 end if
21 // Задача неразрешима
22 return NULL;
```

Рис. 3. Псевдокод алгоритма A_3
Fig. 3. Pseudocode of the algorithm A_3

Для построения схемы с единственным сценарием решения задачи разработан алгоритм $A_3\left(O, R^{in}, R^{out}, S^{in}, S^{out}, n_o, PR_{j_1}, PR_{j_2}, \dots, PR_{j_l}, l \to S, n_l\right)$. Его псевдокод приведен на рис. 3.

После инициализации вспомогательного множества O^* осуществляется его специализация в соответствии с заданными приоритетами операций. Затем специализируются множества R^{in} и R^{out} входных и выходных параметров операций с помощью вспомогательной подпрограммы $Specialize()$. Применяется алгоритм A_1. Если схема S построена, то после применения алгоритма A_2 алгоритм A_3 завершает свою работу. В противном случае задача неразрешима. Разработчик или конечный пользователь РППП могут изменить условия постановки задачи ($S^{in}, S^{out}, PR_{j_1}, PR_{j_2}, \dots, PR_{j_l}$) и вновь запустить процесс планирования.

Алгоритм A_3 имеет вычислительную сложность $O(n_0 \times \max\{l, n_l\})$.

5. Пример

В данном разделе мы демонстрируем аспекты планирования вычисления на простом модельном примере. Пусть множества PR, O и M модели (1) заданы следующим образом: $PR = \{pr_1, pr_2, pr_3, pr_4, pr_5, pr_6\}$, $O = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$, $M = \{m_1, m_2, m_3, m_4, m_5, m_6\}$.

Мы специфицируем операцию следующим образом:

$$operation\_name(a\ set\ of\ inputs \to a\ set\ of\ ouputs\ ).$$

Таким образом, операции имеют следующие спецификации: $o_1(pr_1 \to pr_2)$, $o_2(pr_2 \to pr_3)$, $o_3(pr_2 \to pr_3)$, $o_4(pr_2 \to pr_3)$, $o_5(pr_3 \to pr_4)$, $o_6(pr_4 \to pr_5)$, $o_7(pr_2 \to pr_6)$. Операции o_2, o_3 и o_4 вычисляют значение одного и того же параметра pr_3. Для этих операций автоматически формируется множество $PR_3 = \{1.50, 1.00, 0.75\}$ их приоритетов относительно параметра pr_3. Очевидно, что в модели имеется вычислительная избыточность. Модуль m_2 реализует две операции o_2 и o_7. На практике, это распространенная ситуация, когда в зависимости от качества или структуры значения параметра над ним выполняются концептуально разные операции, реализуемые одной и той же программой. В этом случае, с концептуальной точки зрения, значения выходных параметров могут иметь разный смысл.

Связи между объектами вышеупомянутых множеств представлены матрицами R^{in}, R^{out}, U и $\tilde{R}$:

$$R^{in} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&1&0&0&0&0 \end{bmatrix},\ R^{out} = \begin{bmatrix} 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{bmatrix},\ U = \begin{bmatrix} 0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&0 \end{bmatrix},\ \tilde{R} = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1 \\ 0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \end{bmatrix}.$$

Рассмотрим случай постановки задачи

$$NPPF: S^{in}, S^{out} \to S^o?$$

где $S^{in} = \{pr_1\}$ и $S^{out} = \{pr_4\}$.

Стадии планирования вычислений и построения схемы S представлены в табл. 1, где n – номер стадии (нулевая стадия – это подготовка исходных структур данных), x – случайное событие применения операции при выполнении схемы, $\hat{p}_1(x)$–$\hat{p}_7(x)$ – округленные вероятностные оценки включения операций o_1–o_7 в схему S, $\hat{E}$ – показатель неопределенности процесса планирования, представленный информационной энтропией:

$$\hat{E} = \sum_{i=1}^{m_j} \hat{p}_i(x) E_i(x),$$

$$E_i(x) = \begin{cases} 0, & \text{if } \hat{p}_i(\mathrm{x}) = 0, \\ -log_2 \hat{p}_i(x) & \text{otherwise,} \end{cases}$$

m_j – число операций – кандидатов на применение при выполнении схемы на j-й стадии планирования. Стадии 1-4 осуществляются в рамках метода прямой волны, реализуемого

алгоритмом A_1. Стадии 5-8 выполняются в рамках метода обратной волны, реализуемого алгоритмом A_2. На заключительной стадии работы алгоритма A_2 удаляются все строки матрицы S, все элементы которых равны 0.

Табл. 1. Планирование на основе $NPPF: S^{in}, S^{out} \to S^o$?
Table 1. Planning on the base of $NPPF: S^{in}, S^{out} \to S^o$?

n	S	S^o	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$	$\hat{p}_4$	$\hat{p}_5$	$\hat{p}_6$	$\hat{p}_7$	$\hat{E}$
0	[0 0 0 0 0 0 0]	$\emptyset$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.81
1	[1 0 0 0 0 0 0]	$\{o_1\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
2	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_7\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
3	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_7\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
4	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	2.78
5	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_7\}$	0.16	0.16	0.16	0.16	0.16	–	0.16	2.54
6	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5, o_7\}$	0.20	0.20	0.20	0.20	+	–	0.20	2.32
7	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5\}$	0.25	0.25	0.25	0.25	+	–	–	2.00
8	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&1&1&0&0&0\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_3, o_4, o_5\}$	+	0.33	0.33	0.33	+	–	–	1.58

В табл. 1 символы ‘+’ и ‘–‘ означают соответственно включение операции в схему и ее исключение из кандидатов на включение. В результате построена схема с тремя сценариями решения задачи, определяемыми применением альтернативных операций o_2, o_3 или o_4. Поэтому в схеме остается неопределенность, которая устраняется в дальнейшем в рамках тендера вычислительных работ.

Теперь рассмотрим случай постановки задачи

$$NPPF: S^{in}, S^{out}, PR_3 \to S^o?$$

Стадии планирования вычислений и построения схемы S представлены в табл. 2. Схема S строится с помощью алгоритма A_3. В рамках нулевой стадии происходит специализация множества O. Затем из алгоритма A_3 вызываются алгоритмы A_1 и A_2, которые соответственно реализуют методы прямой и обратной волн (этапы 1-4 и 5-8). В результате построена схема с единственным сценарием решения задачи. В этом случае неопределенность полностью устранена.

Табл. 2. Планирование на основе $NPPF: S^{in}, S^{out}, PR_3 \to S^o$?
Table 2. Planning on the base of $NPPF: S^{in}, S^{out}, PR_3 \to S^o$?

n	S	S^o	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$	$\hat{p}_4$	$\hat{p}_5$	$\hat{p}_6$	$\hat{p}_7$	$\hat{E}$
0	[0 0 0 0 0 0 0]	$\emptyset$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
1	[1 0 0 0 0 0 0]	$\{o_1\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32

2	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\end{bmatrix}$	$\{o_1, o_2, o_7\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
3	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_7\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
4	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_6, o_7\}$	0.20	0.20	–	–	0.20	0.20	0.20	2.32
5	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_7\}$	0.25	0.25	–	–	0.25	–	0.25	2.00
6	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&1\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_5, o_7\}$	0.33	0.33	–	–	+	–	0.33	1.58
7	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\end{bmatrix}$	$\{o_1, o_2, o_5\}$	1.00	+	–	–	+	–	–	0.00
8	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&1&0&0\end{bmatrix}$	$\{o_1, o_2, o_5\}$	+	+	–	–	+	–	–	–

6. Повышение надежности вычислений путем применения схемы с несколькими сценариями решения задач

Избыточные вычисления могут успешно использоваться в том случае, когда происходят отказы определенных типов или заранее назначенные ресурсы недоступны. Например, это возможно при отказе или перегрузке необходимого ресурса.

Пусть схема решения задачи содержит альтернативный сценарий выполнения фрагмента операций, модули которого могут выполняться на другом ресурсе, возможно имеющем другие вычислительные характеристики. Например, он работает под управлением другой операционной системы. В этом случае мы можем выполнять остальные операции схемы, используя этот ресурс.

Для определения фрагмента схемы решения задачи, операции которого выполняются или ожидают своего запуска, введем следующие обозначения:

- $O_s \subseteq O$ – множество операций схемы S;
- $O_e \subset O_s$ – множество операций, которые выполнены, $n_e = |O_e|$ – число таких операций;
- $O_u = O_s \backslash O_e$ – множество операций, выполняющихся или ожидающих своего запуска, $n_u = |O_e|$ – число таких операций.

Схема S_r, включающая операции из O_u, является остаточной схемой решения задачи. $S_r^{in} = S^{in} \cup \left(\cup_{k=1}^{n_e} P_{i_k}^{out}\right)$ и $S_r^{out} = S^{out} \cap \overline{\left(S^{in} \cup \left(\cup_{k=1}^{n_e} P_{i_k}^{out}\right)\right)}$ – это множества ее входных и выходных параметров, где $i_k \in \overline{1, n_o}: o_{i_k} \in O_e$. Множество S_r^{in} включает промежуточные результаты вычислений.

Для определения остаточной схемы S_r мы используем понятие, введенное А.П. Ершовым [36]. Как правило, остаточная схема может включать несколько сценариев решения проблемы. Мы можем редуцировать схему, удалив избыточные операции относительно операции с выбранным приоритетом. Схемы, полученные таким образом для разных приоритетов операций, являются эквивалентными.

Будем считать, что две схемы S_r^1 и S_r^2 являются эквивалентными, если они удовлетворяют следующему условию:

$$S_r^{out} \subseteq \left(S_r^{in1} \cup \left(\cup_{k=1}^{n_1} P_{i_k}^{out}\right)\right) \cap \left(S_r^{in2} \cup \left(\cup_{l=1}^{n_2} P_{l_k}^{out}\right)\right),$$

где $O_s^1 \neq O_s^2$, $O_s^1, O_s^2 \subseteq O_s$, $o_{i_k} \in O_s^1$, $o_{i_l} \in O_s^2$, $S_r^{in1}, S_r^{in} \subseteq S^{in}$, n_1 (n_2) – это число операций в множестве O_s^1 (O_s^2), $0 < k, l \leq n_u$.

Таким образом, в отличие от [37], в рамках управления распределенными вычислениями мы переходим от конкретных модулей к абстрактным операциям, используя приоритеты. Кроме того, мы уточняем логику принятия решения при выборе сценария дальнейшего решения задачи (рис. 4).

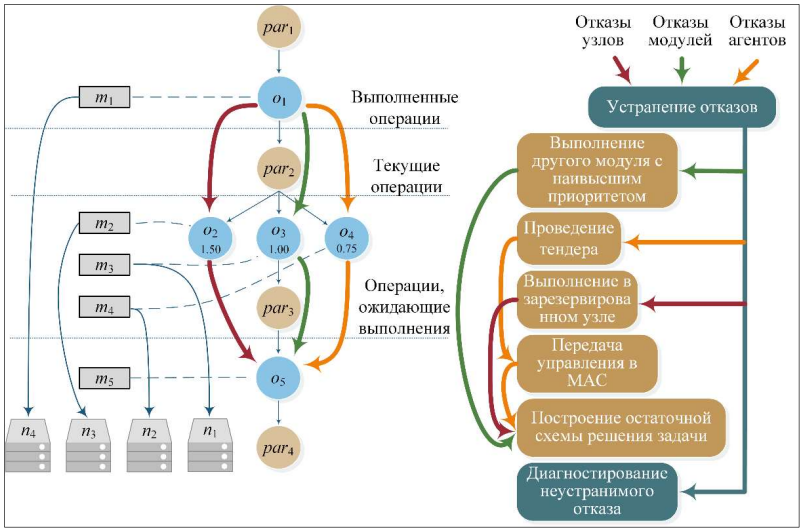

Рис. 4. Блок-схема выбора сценария
Fig. 4. Block diagram for selecting a scenario

На этой блок-схеме идентифицируется отказ. Если прерванная операция предыдущего сценария решения задачи не может быть выполнена, то выбирается сценарий с альтернативной операцией с наивысшим приоритетом.

Если имеются такая операция и доступный ресурс для ее выполнения, то осуществляется переключение на соответствующий сценарий и формируется остаточная схема решения задачи. Затем проводится тендер вычислительных работ с использованием рассмотренных выше оценок времени, стоимости и надежности схемы. После назначения ресурсов для выполнения всех операций схемы, начинается ее выполнение.

Мы сравниваем несколько ключевых характеристик MAC в отношении отказоустойчивости при выполнении схемы решения задачи с двумя традиционными метапланировщиками Condor DAGMan и GridWay. Среди них – поддержка отказоустойчивости в случае отказа ресурсов, системы и прикладного программного обеспечения, а также средняя продолжительность перезапуска. Мы моделируем выполнение потоков заданий, сгенерированных с использованием истории вычислений реальных потоков заданий для широкого спектра практических задач [38]. Мы используем имитационную модель гетерогенной распределенной вычислительной среды [39]. Она была адаптирована к модели (1).

Результаты сравнения представлены в табл. 3. Эксперименты показали, что все сравниваемые системы успешно устраняют проблемы в случае отказов ресурсов или системы. Процент не устраненных отказов этих типов очень низок.

Табл. 3. Результаты сравнения
Table 3. Comparison results

Метапланировщик / Meta-scheduler	**Поддержка отказоустойчивости при отказе ресурса / Fault-tolerance support in case of resource failure**	**Поддержка отказоустойчивости при отказе системы / Fault-tolerance support in case of system failure**	**Поддержка отказоустойчивости при отказе прикладного ПО / Fault-tolerance support in case of applied software failure**	**Средняя продолжительность перезапуска, сек. / Average restart makespan, s**
Condor DAGMAn	+/–	+/–	–	8.05
GridWay	+/–	+/–	–	8.11
MAC	+/–	+/–	+/–	6.23

В то же время Condor DAGMan и GridWay не имеют инструментов для обработки отказов прикладного ПО. Кроме того, такие метапланировщики не поддерживают избыточные обработку заданий с вычислительной избыточностью. Поэтому решение задачи под их контролем может быть прекращено в результате отказа ресурса.

В отличие от таких метапланировщиков, MAC может в некоторых случаях перезапускать вычисления за счет использования вычислительной избыточности. Наличие приоритетов операций значительно ускоряет принятие решений и перезапуск вычислительного процесса.

7. Заключение

Статья посвящена проблемам планирования вычисления в РППП. Исследование базируется на концептуальном моделировании вычислительных процессов.

В данной работе модифицирована и дополнена новыми понятиями предложенная ранее концептуальная модель РППП. В частности, введены приоритеты выполнения операций, которые автоматически формируются и пересчитываются при изменении спецификации модели. Предложена новая постановка задачи, в рамках которой используются такие приоритеты. При избыточности модели приоритеты обеспечивают возможность установить порядок выполнения операций, тем самым смягчая неопределенность при попытке построить схему с единственным сценарием решения задачи. Разработаны специализированные алгоритмы построения схем решения задач на модели с избыточностью и неопределенностью. Показана полезность избыточности модели для повышения надежности вычислений по сравнению с традиционными метапланировщиками.

В рамках будущих исследований предполагается реализовать автоматическое определение приоритетов операций на основе дополнительной информации о модулях, реализующих операции (оценках времени выполнения и размера передаваемых данных, статистики по отказам и др.). В настоящее время такая информацию уже собирается в процессе непрерывной интеграции программного обеспечения распределенных пакетов прикладных программ.

Список литературы / References

[1] Casanova H., Legrand A. et al. Heuristics for scheduling parameter sweep applications in grid environments. In Proc. of the 9th Heterogeneous Computing Workshop, 2000, pp. 349-363.

[2] Casavant T.L., Kuhl J.G. A Taxonomy of Scheduling in General–Purpose Distributed Computing Systems. IEEE Transactions on Software Engineering, vol. 14, issue 2, 1988, pp. 141-154.

[3] Черных А.Н., Бычков И.В. и др. Смягчение неопределенности при разработке научных приложений в интегрированной среде. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 151-171 / Tchernykh A., Bychkov I.V. et al. Mitigating Uncertainty in Developing Scientific Applications in Integrated Environment. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 151-172 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–11.

[4] Feoktistov A., Gorsky S. et al. Collaborative Development and Use of Scientific Applications in Orlando Tools: Integration, Delivery, and Deployment. Communications in Computer and Information Science, vol. 1087, 2020, pp. 18-32.

[5] Cardoso J, Sheth A. Semantic E-Workflow Composition. Journal of Intelligent Information Systems, vol. 21, issue 3, 2003, pp.191-225.

[6] Mineau G.W., Missaoui R., Godinx R. Conceptual modeling for data and knowledge management. Data & Knowledge Engineering, vol. 33, issue 2, 2000, pp. 137-168.

[7] Yu J., Buyya R. A taxonomy of workflow management systems for grid computing. Journal of Grid computing, vol. 3, issue 3-4, 2005, pp. 171-200.

[8] Fahringer T., Prodan R. et al. ASKALON: a Grid application development and computing environment. In Proc. of The 6th IEEE/ACM International Workshop on Grid Computing, 2005, pp. 1-10.

[9] Tschager T., Schmidt H.A. Condor, DAGwoman: enabling DAGMan-like workflows on non-Condor platforms. In Proc. of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, 2012, article no. 3, pp. 1-6.

[10] Amin K., Laszewski G. et al. GridAnt: a client-controllable grid workflow system. In. Proc. of the 37th Annual Hawaii International Conference on System Science, 2004, pp. 1-10.

[11] Carrion I.M., Huedo E., Llorente I.M. Interoperating Grid infrastructures with the GridWay metascheduler. Concurrency Computation, vol. 27, issue. 9, 2015, pp. 2278-2290.

[12] Barseghian D., Altintas I. et al. Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis. Ecological Informatics, vol. 5, issue 1, 2010, pp. 42-50.

[13] Missier P., Soiland-Reyes S. et al. Taverna, Reloaded. Lecture Notes in Computer Science, vol. 6187, 2010, pp. 471-481.

[14] Vahi K., Harvey I. et al. A General Approach to Real-Time Workflow Monitoring. In Proc. of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, 2012, pp. 108-118.

[15] Benedyczak K., Bala P. et al. Key aspects of the UNICORE 6 security model. Future Generation Computer Systems, vol. 27, issue 2, 2011, pp. 195-201.

[16] Extensible Markup Language (XML). Available at: https://www.w3.org/XML, accessed 13.07.2021.

[17] XML Process Definition Language. Available at: https://www.w3.org/TR/xmlschema-0, accessed 13.07.2021.

[18] Guizania K., Ghannouchia S.A. An approach for selecting a business process modeling language that best meets the requirements of a modeler. Procedia Computer Science, vol. 181, 2021, pp. 843-851.

[19] Mo'Minov B.B., Eshankulov, K. Modelling Asynchronous Parallel Process with Petri Net. International Journal of Engineering Advanced Technology, vol. 8, issue 5S3, 2019, pp. 400-405.

[20] Unified Modeling Language (UML) Diagrams. Available at: https://www.uml.org, accessed 13.07.2021.

[21] Deelman E., Vahi K. et al. Pegasus, a workflow management system for science automation. Future Generation Computer Systems, vol. 46, 2015, pp. 17-35.

[22] Blythe J., Deelman E. et al. The Role of Planning in Grid Computing. In Proc. Of the Thirteenth International Conference on Automated Planning and Scheduling, 2003, pp. 153–163.

[23] Matskin M., Tyugu E. Strategies of structural synthesis of programs and its extensions. Computing and Informatics, vol. 20, issue 1, 2001, pp. 1-26.

[24] Опарин Г.А., Новопашин А.П. Булевы модели и методы планирования параллельных абстрактных программ. Автоматика и телемеханика, вып. 8, 2008 г., стр. 166-175 / Oparin G.A., Novopashin A.P. Boolean models and planning methods for parallel abstract programs. Automation and Remote Control, vol. 69, no. 8, 2008, pp. 1423-1432.

[25] Новосельцев В.Б. Синтез параллельных рекурсивных программ в структурных функциональных моделях. Программирование, том 33, вып. 5, 2007 г., стр. 75-81 / Novoseltsev V.B. Synthesis of parallel recursive programs in structural functional models. Programming and Computer Software, vol. 33, no. 5, pp. 293-298.

[26] Malyshkin V.E., Perepelkin V.A. LuNA fragmented programming system, main functions and peculiarities of run-time subsystem. Lecture Notes in Computer Science, vol. 6873, 2011, pp. 53-61.

[27] Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. Наука. Сибирское отделение, 1988 г., 128 стр. / Valkovsky V.A., Malyshkin V.E. Synthesis of parallel programs and systems on computational models. Nauka. Siberian branch, 1988, 128 p. (in Russian).

[28] Gorsky S., Kostromin R. et al. Orlando Tools: Supporting High-performance Computing in Distributed Environments. In Proc. of the 6th International Conference on Information Technology and Nanotechnology, 2020, pp. 1-6.

[29] Feoktistov A., Tchernych A. et al. Knowledge Elicitation in Multi-Agent System for Distributed Computing Management. In Proc. of the 40th International Convention on information and communication technology, electronics and microelectronics, 2017, pp. 1350-1355.

[30] Bychkov I., Feoktistov A. et al. Machine Learning in a Multi-Agent System for Distributed Computing Management. In Proc. of the International Conference Information Technology and Nanotechnology. Session Data Science, CEUR-WS Proceedings, vol. 2212, 2018, pp. 89-97.

[31] Feoktistov A., Kostromin R., Tchernykh A. Agent Behavior Model for Distributed Computing Management in the Environment with Virtualized Resources. In Proc. of the 41st International Convention on information and communication technology, electronics and microelectronics, 2018, pp. 1153-1158.

[32] Feoktistov A.G., Basharina O.Yu. Predicting runtime of computational jobs in distributed computing environment. In Proc. of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments, CEUR-WS Proceedings, 2020, vol. 2638, pp. 109-117.

[33] Ryabinin I.A. Logical probabilistic analysis and its history. International Journal of Risk Assessment and Management, vol. 18, issue 3-4, 2015, pp. 256-265.

[34] Feoktistov A.G., Sidorov I.A. Logical-Probabilistic Analysis of Distributed Computing Reliability. In Proc. of the 39th International Convention on information and communication technology, electronics and microelectronics, 2016, pp. 247-252.

[35] Bychkov I., Oparin G. et al. Conceptual Model of Problem-Oriented Heterogeneous Distributed Computing Environment with Multi-Agent Management. Procedia Computer Science, vol. 103, 2017, pp. 162-167.

[36] Ershov A.P. On Mixed Computation: Informal Account of the Strict and Polyvariant Computation Schemes. In Control Flow and Data Flow: Concepts of Distributed Programming. Springer Study Edition, vol. 14, 1985, pp. 107-120.

[37] Feoktistov A., Kostromin R. et al. Multi-Agent Algorithm for Re-Allocating Grid-Resources and Improving Fault-Tolerance of Problem-Solving Processes. Procedia Computer Science, 2019, vol. 150, pp. 171-178.

[38] Tchernykh A., Feoktistov A. et al. Orlando Tools: Development, Training, and Use of Scalable Applications in Heterogeneous Distributed Computing Environments. Communications in Computer and Information Science, vol. 979, 2019, pp. 265-279.

[39] Bychkov I.V., Oparin G.A. et al. Multiagent control of computational systems on the basis of meta-monitoring and imitational simulation. Optoelectronics, Instrumentation and Data Processing, vol. 52, issue 2, 2016, pp. 107-112.

Информация об авторах / Information about authors

Александр Геннадьевич ФЕОКТИСТОВ – кандидат технических наук, доцент, заведующий лабораторией параллельных и распределенных вычислительных систем. Области исследования: распределенные пакеты программ, мультиагентные технологии, имитационное моделирование, складская логистика.

Alexander Gennadevich FEOKTISTOV – Ph.D., Associate Professor, Head of the Laboratory of Parallel and Distributed Computing Systems. Fields of research: distributed software packages, multi-agent technologies, simulation modeling and warehouse logistics

Роман Олегович КОСТРОМИН – кандидат технических наук, младший научный сотрудник. Область исследования: мультиагентные средства управления распределенными вычислениями.

Roman Olegovich KOSTROMIN – Ph.D., Junior Researcher. Field of research: multi-agent tools for distributed computing management.

Сергей Алексеевич ГОРСКИЙ – кандидат технических наук, старший научный сотрудник. Сфера научных интересов: параллельные и распределенные вычисления, математическое моделирование и сервис-ориентированное программирование.

Sergei Alexeevich GORSKY – Ph.D., Senior Researcher. Research interests: parallel and distributed computing, mathematical modeling, and service-oriented programming.

Игорь Вячеславович БЫЧКОВ – академик РАН, доктор технических наук, профессор, директор Института динамики систем и теории управления им. В.М. Матросова СО РАН. Области исследования: искусственный интеллект, геоинформационные системы, веб-технологии, математическое моделирование и облачные вычисления.

Igor Vyacheslavovich BYCHKOV – Academician of RAS, Ph.D., Professor, Director of the Matrosov Institute for System Dynamics and Control Theory of SB RAS. Fields of research: artificial intelligence, geoinformation systems, web-technologies, mathematical modeling, and cloud computing.

Андрей Николаевич ЧЕРНЫХ получил степень кандидата наук в Институте точной механики и вычислительной техники РАН. Он является профессором Центра научных исследований и высшего образования в Энсенаде, Нижняя Калифорния, Мексика. В научном плане его интересуют многоцелевая оптимизация распределения ресурсов в облачной среде, проблемы безопасности, планирования, эвристики и метаэвристики, интернет вещей и т.д.

Andrei Nikolaevitch TCHERNYKH received his PhD degree at the Institute of Precision Mechanics and Computer Engineering of the Russian Academy of Sciences. He is holding a full professor position in computer science at CICESE Research Center, Ensenada, Baja California, Mexico. He is interesting in grid and cloud research addressing multiobjective resource optimization, both, theoretical and experimental, security, uncertainty, scheduling, heuristics and meta-heuristics, adaptive resource allocation, and Internet of Things.

Ольга Юрьевна БАШАРИНА – кандидат технических наук, доцент, научный сотрудник ИДСТУ СО РАН, доцент Иркутского государственного университета, Сфера научных интересов: исследование операций, имитационное моделирование и распределенные вычисления.

Olga Yurevna BASHARINA – Ph.D., Research Officer of ISDCT SB RAS, Associate Professor of the Irkutsk State University. Research interests: operations research, simulation modeling and distributed computing.