DOI: 10.15514/ISPRAS-2019-31(2)-7

Непрерывная интеграция функционального наполнения распределенных пакетов прикладных программ в Orlando Tools

А.Г. Феоктистов, ORCID: 0000-0002-9127-6162 <agf65@icc.ru>
С.А. Горский, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>
И.А. Сидоров, ORCID: 0000-0002-2398-5426 <ivan.sidorov@icc.ru>
Р.О. Костромин, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>
Е.С. Фереферов, ORCID: 0000-0002-7316-444X <fereferov@icc.ru>
И.В. Бычков, Scopus ID: 56308077400 <bychkov@icc.ru>

Институт динамики систем и теории управления им. В.М. Матросова СО РАН, 663033, Россия, г. Иркутск, ул. Лермонтова, д. 134

Аннотация. В статье представлен новый подход к решению важных практических проблем комплексной отладки, совместного тестирования и анализа времени выполнения версий программных модулей в распределенной вычислительной среде. Эти проблемы возникают в процессе поддержки непрерывной интеграции функционального наполнения (прикладного программного обеспечения) распределенных пакетов прикладных программ (научных приложений). Исследование ориентировано на пакеты, которые используются для проведения крупномасштабных экспериментов, осуществляемых в рамках междисциплинарных исследований, в гетерогенных распределенных вычислительных средах, интегрирующих Grid и облачные вычисления. Научная новизна предложенного подхода заключается в объединении методологии создания распределенных пакетов прикладных программ с современной практикой разработки программного обеспечения на основе его непрерывной интеграции с использованием знаний о специфике решаемых задач. Средства непрерывной интеграции, разрабатываемые в рамках предложенного подхода, существенно расширяют спектр ее возможностей применительно к процессам создания и использования таких пакетов в сравнении с известными инструментами. Фундаментальной основой их функционирования является концептуальная модель, в рамках которой осуществляется спецификация, планирование и выполнение процессов непрерывной интеграции прикладного программного обеспечения с привязкой к конкретным предметным данным и решаемым задачам. Использование разрабатываемых средств на практике ведет к снижению числа ошибок и сбоев прикладного программного обеспечения при разработке и применении пакетов, что, в свою очередь, существенно сокращает время проведения крупномасштабных вычислительных экспериментов и повышает эффективность использования ресурсов гетерогенной распределенной вычислительной среды. Результаты практических экспериментов по применению прототипа системы непрерывной интеграции прикладного программного обеспечения показывают его высокую эффективность.

Ключевые слова: распределенная вычислительная среда; пакеты прикладных программ; программное обеспечение; непрерывная интеграция

Для цитирования: Феоктистов А.Г., Горский С.А., Сидоров И.А., Костромин Р.О., Фереферов Е.С., Бычков И.В. Непрерывная интеграция функционального наполнения распределенных пакетов прикладных программ. Труды ИСП РАН, том 31, вып. 2, 2019 г., стр. 83-96. DOI: 10.15514/ISPRAS-2019-31(2)-7

Благодарности. Исследования выполнены при поддержке РФФИ, проект № 19-07-00097-а.

Continuous integrating modules of distributed applied software packages in Orlando Tools

A.G. Feoktistov, ORCID: 0000-0002-9127-6162 <agf@icc.ru>
S.A. Gorsky, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>
I.A. Sidorov, ORCID: 0000-0002-2398-5426 <ivan.sidorov@icc.ru>
R.O. Kostromin, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>
E.S. Fereferov, ORCID: 0000-0002-7316-444X <fereferov@icc.ru>
I.V. Bychkov, Scopus ID: 56308077400 <bychkov@icc.ru>

Matrosov Institute for System Dynamics and Control Theory of SB RAS, 134, Lermontov st., Irkutsk, 664033, Russia.

Abstract. We propose a new approach to solving important practical problems of complex debugging, joint testing, and analysis of the execution time of software module versions in a heterogeneous distributed computing environment that integrating Grid and cloud computing. These problems arise in the process of supporting the continuous integration of modules of distributed applied software packages. The study focuses on the packages that are used to conduct large-scale computational experiments. The scientific novelty of the proposed approach is to combine the methodology for creating the packages with modern software development practices based on its continuous integration using knowledge about the specifics of the problems being solved. Our contribution is multifold. We expanded the capabilities of continuous integration tools by developing new additional tools for the markup and transformation of data from poorly structured sources and predicting modules execution time. In addition, we developed a technological scheme of the joint applying our developed tools and external systems for continuous integration. Therefore, we provide a more large range of capabilities of continuous integration in relation to the processes of creating and using the packages in comparison with the well-known tools. The fundamental basis of their functioning is a new conceptual model of the packages. This model supports the specification, planning, and execution of software continuous integration processes taking into account the specific subject data and problems being solved. Applying the developed tools in practice leads to a decrease in the number of errors and failures of applied software in the development and use of the packages. In turn, such decrease significantly reduces the time for large-scale computational experiments and increases the efficiency of using resources of the environment. The results of practical experiments on the use of system prototype for continuous integration of applied software show their high efficiency.

Keywords: distributed computing environment; applied software packages; software; continuous integration

For citation: Feoktistov A.G., Gorsky S.A., Sidorov I.A., Kostromin R.O., Fereferov E.S., Bychkov I.V. Continuous integrating modules of distributed applied software packages in Orlando Tools. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 2, 2019. pp. 83-96 (in Russian). DOI: 10.15514/ISPRAS-2019-31(2)-7

Acknowlwdgement. The studies were supported by the Russian Foundation for Basic Research, project No. 19-07-00097-a.

1. Введение

В настоящее время применение высокопроизводительных вычислений стало неотъемлемой составляющей процесса поддержки проведения крупномасштабных экспериментов по решению больших научных и прикладных задач в различных сферах человеческой деятельности. В зависимости от масштабности решаемых задач в вычислительную инфраструктуру могут быть включены персональные компьютеры (ПК), серверы, кластеры, ресурсы центров коллективного пользования, Grid-системы и облачные платформы. В общем случае, организуется гетерогенная распределенная вычислительная среда (ГРВС), в которой выполняются приложения, характеризующиеся разной степенью масштабируемости вычислений, чувствительности к неоднородности ресурсов, потребности в виртуализации ресурсов среды, а также необходимости интеграции модели своей предметной области с информацией о программно-аппаратной инфраструктуре среды и административных политиках, определенных для ее ресурсов. Приложения, чувствительные к неоднородности

ресурсов, выполняются, как правило, в однородных узлах кластера или в виртуальной среде. Потребность в ней возникает также и у приложений, использующих программное обеспечение (ПО), отличное от установленного в узлах среды. Применение высокопроизводительных вычислений связано с отображением алгоритмов решения задач на архитектуру вычислительной среды [1, 2].

Можно выделить отдельный класс приложений – распределенные пакеты прикладных программ (РППП), которые характеризуются применением модульного подхода, высокой степенью масштабируемости и возможностью их выполнения на разнородных ресурсах среды. Пользователи РППП заинтересованы в максимальном использовании вычислительных мощностей ГРВС. В модели предметной области РППП вычислительный процесс представляется в виде схемы решения задачи, которая тесно коррелирует с понятием рабочего процесса (workflow). Системы разработки и применения workflow можно рассматривать как частный случай РППП.

Как правило, ГРВС характеризуется постоянным изменением своих программно-аппаратных и информационных ресурсов. Это влечет за собой необходимость решения проблем реконфигурации вычислительных сред РППП, модификации их библиотек программных модулей и/или разработки нового ПО, поддержки корректности взаимодействия различных версий программных модулей в рамках единой схемы решения задачи, учета условий применения этих версий, комплексирования изменяющихся источников предметной информации со структурами данных РППП, прогнозирования времени выполнения модулей разных версий с целью оптимизации показателей функционирования выделяемых им ресурсов и эффективности решения задач. Для решения вышеперечисленных проблем могут быть использованы в той или иной мере средства непрерывной интеграции ПО.

Однако поддержка такой интеграции по-прежнему является нетривиальной проблемой для инструментальных средств организации РППП, включая системы создания и применения workflow [3]. Данные средства зачастую не готовы в полной мере поддерживать сложный процесс непрерывной интеграции в сочетании с концептуальным моделированием, традиционно применяемым в таких пакетах, а также использованием предметно-ориентированных знаний в сочетании со специализированными знаниями о программно-аппаратной инфраструктуре среды и административных политиках, установленных в ее узлах.

В этой связи в статье предложен новый подход к обеспечению непрерывной интеграции функционального наполнения РППП, базирующийся на слиянии методологии построения таких пакетов с современной практикой разработки ПО на основе его непрерывной интеграции с использованием предметно-ориентированных знаний. В рамках этого подхода расширены возможности средств непрерывной интеграции за счет разработки новых дополнительных инструментов для разметки и преобразования данных из слабоструктурированных источников, а также прогнозирования времени выполнения модулей РППП. Разработана технологическая схема совместного использования разработанных инструментов и внешних систем для непрерывной интеграции.

Оставшаяся часть статьи структурирована следующим образом: во втором разделе статьи приведен краткий обзор средств и обсужден ряд важных проблем непрерывной интеграции ПО. В третьем разделе рассмотрены вопросы, связанные с разработкой РППП в Orlando Tools. В четвертом разделе предложена технологическая схема непрерывной интеграции функционального наполнения пакетов. В следующих двух разделах особое внимание уделено используемому подходу к разметке и трансформации данных из слабоструктурированных источников, а также разработанной модели оценки времени выполнения модулей. Результаты практического применения разработанного прототипа подсистемы непрерывной интеграции в Orlando Tools приведены в седьмом разделе. Заключительный раздел обобщает результаты исследования.

2. Обзор средств непрерывной интеграции программного обеспечения

В процессе разработки сложных программных комплексов перед их разработчиками возникает необходимость организации взаимодействия между функциональными подсистемами таких комплексов. Подсистемы могут создаваться различными разработчиками с использованием широкого спектра языков программирования и в ориентации на функционирование под управлением разнообразных программно-аппаратных платформ. Основным назначением непрерывной интеграции является выявление и устранение проблем взаимодействия отдельных подсистем программного комплекса между собой путем автоматизации их сборки, отладки и совместного тестирования [4].

На сегодняшний день разработан широкий набор средств, обеспечивающих автоматизацию процессов непрерывной интеграции в ходе разработки сложных программных комплексов. В их числе системы CircleCI [5], Jenkins [6], TeamCity [7], Travis [8], GitLab [9] и многие другие средства [10, 11]. Каждая система имеет свои специфические особенности с точки зрения обеспечения функциональных возможностей, последовательности выполнения действий пользователями данной системы и ее администрирования. Все они обладают определенными преимуществами и недостатками.

Некоторые их них (например, CruiseControl.NET [12] или Apache Gump [13]) жестко привязаны к языку программирования, на котором осуществляется разработка ПО, с целью максимального использования возможностей данного языка в связке со специализированными средствами управления библиотеками программ. В качестве примера такого средства можно привести систему Conan [14] для языка C++. Другие средства предоставляют доступ только в режиме работы облачного сервиса (например, CircleCI или TeamCity) и не позволяют разместить весь необходимый набор средств непрерывной интеграции на своих ресурсах. Возникают определенные сложности с интеграцией таких средств, как BuildMaster [15] и Travis [16], со средами разработки из-за использования разных форматов представления данных и рабочих процессов.

Как показывает сравнительный анализ средств непрерывной интеграции, GitLab является одной из наиболее перспективных систем подобного назначения. Она обеспечивает тесную интеграцию процессов автоматического тестирования ПО и хранения его исходного кода с помощью репозитория Git [17], а также запуск тестов на серверах сборки программного обеспечения с применением таких средств, как сетевой протокол безопасного доступа SSH [18], скрипты на языке программирования Shell [19], программный комплекс VirtualBox [20] для виртуализации различных операционных систем (Microsoft Windows, Linux, FreeBSD, macOS, Solaris/OpenSolaris, ReactOS, DOS и других систем), программные продукты виртуализации компании Parallels [21], программные комплексы для автоматизации развертывания и управления приложениями в среде виртуализации Docker [22] и Kubernetes [23]. Возможность установки GitLab разработчиком ПО на собственных вычислительных ресурсах позволяет обеспечить необходимый уровень безопасности и гибкости всей системы непрерывной интеграции в целом. В табл. 1 приведен сравнительный анализ обеспечения важных функциональных возможностей наиболее популярными средствами непрерывной интеграции.

Табл.1. Результаты сравнительного анализа
Table 1. Results of the comparative analysis

Функциональная возможность	Travis CI	TeamCity	Jenkins	CircleCI	GitLab
Установка системы на ресурсах разработчика	–	+	+	–	+
Поддержка тестирования ПО для ОС Linux	+	+	+	+	+

Поддержка тестирования ПО для ОС Windows	–	+	+	–	+
Мониторинг ПО	–	–	–	–	+
Наличие репозитория образов контейнеров	–	–	–	–	+
Проверка качества программного кода	–	–	+	–	+

Информация, извлекаемая из источников предметных данных (например, веб-страниц [24]), и результаты решения задач, получаемые в результате выполнения РППП, зачастую являются сложно структурированными, неоднородными и подверженными частым изменениям. В этой связи в процессе непрерывной интеграции требуется применение гибкой, основанной на знаниях модели, позволяющей определять взаимосвязи между первичной информацией и структурами данных, используемыми такими пакетами. Известные средства непрерывной интеграции не поддерживают такой возможности.

Отладка и тестирование ПО на разных программно-аппаратных платформах потенциально позволяют в то же время выполнять и оценку времени выполнения этого ПО с учетом различий характеристик используемых ресурсов. Для решения этой задачи (нетривиальной для существующих средств непрерывной интеграции) необходима разработка новых методов и средств прогнозирования времени выполнения программ.

3. Orlando Tools: разработка и применение распределенных пакетов прикладных программ

Разработка приложений для проведения научных и прикладных исследований осуществляется с помощью инструментального комплекса Orlando Tools [24]. Этот комплекс обеспечивает построение предметно-ориентированных вычислительных сред, в которых интегрируются различные вычислительные инфраструктуры, поддерживающие, как Grid, так и облачные вычисления.

В описании модели предметной области РППП выделяются три концептуально обособленных слоя знаний – вычислительный, схемный и продукционный, над которыми формируются постановки задач и строятся схемы их решения. Вычислительный слой знаний реализуется программными модулями пакета, которые представляют его функциональное наполнение. Параметры и операции пакета отражают схемные знания. Параметры отражают значимые характеристики и свойства предметной области. Операции выступают в качестве отношений вычислимости между двумя подмножествами параметров предметной области. Такое отношение обуславливает возможность вычисления искомых значений параметров первого подмножества, когда известны значения параметров второго подмножества. Модули пакета представляют собой программную реализацию операций. Спецификация каждого модуля включает информацию об исполняемой прикладной программе (наименовании, версии, входных и выходных параметрах, процессах сборки и компиляции, инструкциях по запуску, допустимых классах ресурсов для выполнения). Возможность выполнения операций в процессе решения задачи в зависимости от текущего хода вычислений и состояния ресурсов ГРВС определяется продукциями, которые формируют продукционный слой знаний.

На вычислительной модели пакета формулируются постановки задач (формализованные описания условий задач в терминах параметров и операций). В общем случае постановка задачи определяет:

- входные параметры (данные, необходимые для решения задачи);
- выходные параметры (результаты решения задачи);
- операции, которые могут или должны быть выполнены в процессе решения задачи над полем параметров;

- ограничения, определяющие возможность выполнения операций.

По сформулированной постановке задачи строится ее схема решения, отражающая информационно-логические связи между операциями пакета.

4. Общая схема непрерывной интеграции

Архитектура Orlando Tools [25] расширена функционирующим прототипом системы непрерывной интеграции. Общая схема взаимодействия Orlando Tools с подсистемами непрерывной интеграции приведена на рис. 1. Запуск процессов непрерывной интеграции осуществляется на следующих четырех этапах, связанных с разработкой РППП:

- внесение изменений в исходный код модулей;
- добавление новой версии модуля;
- создание новых образов виртуальных машин для выполнения модулей;
- разработка новых спецификаций баз данных.

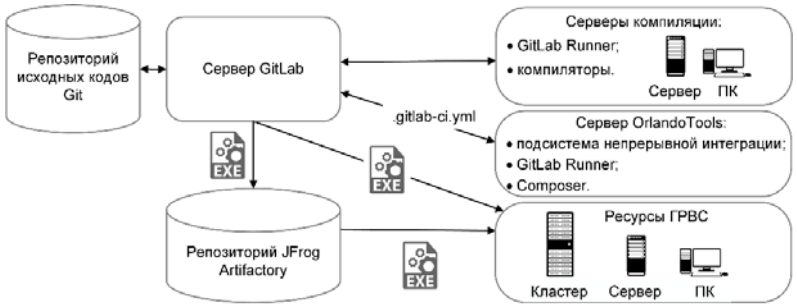

Рис. 1. Схема поддержки непрерывной интеграции
Fig. 1. Scheme of supporting the continuous integration

В Orlando Tools в качестве системы управления версиями исходного кода модулей используется репозиторий Git, доступ к которому обеспечивается средой GitLab. При внесении в эту систему изменений, связанных с разработкой нового или модификацией уже существующего ПО, автоматически осуществляется его компиляция на специализированных серверах или узлах ГРВС с помощью предустановленного агента Gitlab Runner.

В случае успешной компиляции, осуществляется тестирование модуля. Тестовые данные должны быть подобраны разработчиком пакета таким образом, чтобы время тестирования не превышало предельных значений (по умолчанию до 1 минуты), установленных администратором ГРВС. Тестирование модуля также осуществляется с помощью GitLab Runner.

Второй этап связан с добавлением новой версии модуля в РППП. В случае ее успешного тестирования на предыдущем этапе статус модуля помечается как «стабильный» и его бинарная версия добавляется в репозиторий JFrog Artifactory [26]. Данный репозиторий служит в качестве хранилища бинарных версий модулей и предоставляет интерфейсы для взаимодействия с различными пакетными менеджерами (например, Advanced Packaging Tool [27] или Yum Package Manager [28]). При включении новой версии модуля в репозиторий осуществляется автоматический запуск тестирования схем решения задач, операции которых реализуются данным модулем.

Создание новых образов виртуальных машин, с помощью которых в дальнейшем будут выполняться модули в узлах ГРВС, осуществляется в рамках третьего этапа. В случае добавления нового образа виртуальной машины с характеристиками, совпадающими с характеристиками требований к среде выполнения в спецификациях стабильных модулей,

производится запуск таких модулей для тестирования их корректного выполнения в новом образе виртуальной машины.

Последний этап связан с подготовкой спецификаций предметных баз данных для модулей, которые характеризуются обработкой больших объемов данных (BigData), которые не могут быть переданы в виде файлов для входных и выходных параметров этих модулей в узел ГРВС. Организация передачи и обработки таких объемов данных требует использования дополнительных предметных баз данных и веб-сервисов, предоставляющих возможности выборки данных по определенным критериям. В Orlando Tools такие базы данных специфицируются в формате JSON. В случае разработки новой спецификации предметной базы данных осуществляется запуск модулей, использующих эти базы, с целью тестирования процессов обмена данными.

5. Инструментальное средство разметки и трансформации данных из слабоструктурированных источников

Необходимость получения значений предметных данных, содержащихся в слабоструктурированных источниках (базах данных или отдельных файлах разных форматов, созданных ранее субъектами различных видов деятельности в рамках предметной области и содержащих результаты экспериментов или статистические показатели) часто возникает в процессе создания и применения РППП. Такие данные требуются для задания исходных параметров задач пакета. В связи с этим, его разработчики вынуждены осуществлять нетривиальное преобразование извлекаемых предметных данных к определенной форме их представления в пакете.

Рис. 2. Схема работы средства разметки и трансформации данных из слабоструктурированных источников

Fig. 2. Tool operation scheme of the markup and transformation of data from semi-structured sources

В системе непрерывной интеграции разработана схема разметки и трансформации данных из слабоструктурированных источников в виде файлов электронных таблиц (CSV, MS Excel) к структурированной форме представления значений параметров РППП в виде баз данных или XML-, JSON- файлов. В рамках данной схемы разработчику пакета предоставляется возможность визуальной настройки и спецификации процесса трансформации данных, необходимых для решения задач конкретного класса. На этапе разметки разработчик формирует дерево сущностей, указывая в каких диапазонах слабоструктурированных документов эти сущности находятся. Отдельным атрибутам могут быть заданы правила трансформации, например, разделение на несколько атрибутов или удаление частей значений. Дерево сущностей может быть дополнено классами для обеспечения формирования сложных целевых структур. Знания разработчика о разметке и правилах трансформации сохраняются в виде структурной спецификации, которые могут быть

применены многократно при решении типовых задач извлечения и трансформации данных (например, по использованию статистической информации за разные периоды). Кроме того, сформированные структурные спецификации могут быть использованы для автоматизации создания прикладных программных систем для работы с данными целевых структур. Общая схема разметки и трансформации представлена на рис. 2.

6. Оценка времени выполнения модулей

Разработана специальная модель оценки времени выполнения модулей схемы решения задачи. Модуль в процессе его тестирования запускается на эталонном узле в программной среде для профилирования программ. В процессе выполнения модуля определяются его временные затраты на работу с различными компонентами узла с учетом его вычислительных характеристик. Путем сравнения характеристик эталонного и целевого узлов прогнозируется время выполнения модуля на целевом узле с некоторой погрешностью. В рамках разработанной модели формируются наборы $CR = \{cr_1, cr_2, \dots, cr_m\}$ и $CT = \{ct_1, ct_2, \dots, ct_m\}$ характеристик эталонного и целевого узлов и определяются их значения. Между cr_i и ct_i устанавливается взаимно однозначное соответствие, $i = \overline{1,m}$. Затем формируется множество $P = \{p_1(d), p_2(d), \dots, p_n(d)\}$. Элемент $p_j(d)$, $j = \overline{1,n}$ отражает вычислительную нагрузку узла при выполнении модуля (число выполненных целочисленных операций, число операций с плавающей точкой, число обращений к оперативной памяти и кэш-памяти разных уровней, число промахов таких обращений, число сессий чтения с диска и записи на диск и другие показатели) в зависимости от объема d обрабатываемых данных этим модулем. Далее определяются функции $f_l(CR, P)$, вычисляющие время работы l-го компонента узла при обработке вычислительной нагрузки, $l = \overline{1,k}$. Время выполнения модуля в эталонном узле оценивается выражением

$$\hat{T}_r(d) = \sum_{l=1}^{k} f_l\big(CR, P, g(d)\big),$$

где $g(d)$ – это интерполяционная функция, определяющая зависимость от объема данных d экспериментальным путем. Погрешность ε данной оценки определяется из разности реального времени $T_r(d)$ выполнения модуля и $\hat{T}_r(d)$:

$$\varepsilon = T_r(d) - \hat{T}_r(d).$$

Время $\hat{T}'_t(x)$ выполнения модуля в целевом узле оценивается выражением

$$\hat{T}'_t(x) = \hat{T}_t(d) + \varepsilon \frac{\hat{T}_r(d)}{\hat{T}_t(d)}, \hat{T}_t(d) = \sum_{l=1}^{k} f_l\big(CT, P, g(d)\big).$$

Оценка времени выполнения модуля, получаемая с помощью предложенной модели, является достаточно грубой. Кроме того, автоматический выбор интерполяционной функции $g(d)$ для некоторых классов модулей является трудно реализуемым. Тем не менее, как показано в следующем разделе, такая оценка позволяет в ряде случаев существенно улучшить пользовательские оценки, а также оценки, определяемые на основе вычислительной истории.

7. Применение непрерывной интеграции в Orlando Tools

В качестве примера применения непрерывной интеграции рассматривается разработка модулей РППП, предназначенного для тестирования модификаций алгоритма мультистарта на задачах поиска глобального минимума многоэкстремальных функций. Модель схемы решения подобных задач приведена на рис. 3, где $z_1 - z_{13}$ и $o_1 - o_4$ – это соответственно параметры и операции вычислительной модели. Назначение параметров и операций, а также особенности процесса решения задач детально рассмотрены в [29]. В данной модели схемы z_9 является составным параметром, а z_3 и z_{12} представляют параллельные списки данных, элементы которых обрабатываются q экземплярами операции o_3. В пакете разработан набор модулей, реализующих операции $o_1 - o_4$ и представляющих этапы тестируемых

модификаций алгоритма применительно к различным многоэкстремальным функциям. Процесс решения таких задач требует многократного обновления модулей.

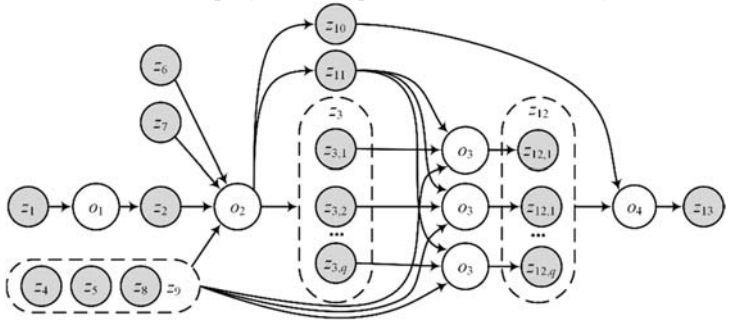

Рис. 3. Модель схемы решения задачи
Fig. 3. Problem-solving scheme model

Спецификация объектов вычислительной модели. На рис. 4 приведены спецификации основных объектов вычислительной модели в Orlando Tools, используемые в процессе тестирования модулей.

```
<module>
  <name>Имя_модуля</name>
  <parameters>Входные_параметры>
              Выходные_параметры
  </parameters>
  <signature>Команда_запуска
  </signature>
  <cores>Число_ядер</cores>
  <walltime>Время_останова</walltime>
  <run_mode>Режим_запуска</run_mode>
  <repository>описание_модуля
  </repository>
</module>
```

a)

```
<operation>
  <name>Имя_операции</name>
  <parameters>Входные_параметры>
              Выходные_параметры
  </parameters>
  <run_condition>Условие_запуска
  </run_condition>
  <while_flag>Признак_повторения
  </while_flag>
  <module_name>Имя_модуля</module_name>
  <split_condition>Условие_расщепления
  </split_condition>
  <task_name>Имя_задачи</task_name>
</operation>
```

b)

```
<parameter>
  <name>Имя_параметра</name>
  <extention>Расширение</extention>
  <list>Число_элементов</list>
</parameter>
```

c)

```
<task>
  <name>Имя задачи</name>
  <parametrs>Входные_параметры>
             Выходные_параметры</parametrs>
  <operations>Список_операций</operations>
  <modules>Список_версий_модулей</modules>
  <test>описание_данных</test>
</task>
```

d)

Рис. 4. Спецификации объектов вычислительной модели: модуль (a), операция (b), параметр (c) и схема решения задачи (d)
Fig. 4. Object specifications in the computational model: module (a), operation (b), parameter (c), and problem-solving scheme (d)

Новые элементы спецификации, необходимые для поддержки процесса непрерывной интеграции, выделены полужирным шрифтом. В спецификации модуля элемент <repository> содержит информацию о размещении этого модуля в репозитории и его зависимости от других модулей. Элемент <test> спецификации задачи включает информацию о тестовых данных. Информация в обоих элементах представлена в формате JSON, требуемом для их дальнейшей обработки с помощью пакетного менеджера Composer. Обе спецификации расширены дополнительными элементами необходимыми для работы Orlando Tools, помещенными в элемент "orlando". Они игнорируются пакетным менеджером Composer и позволяют связать модули пакета программ с их реализациями в репозиториях, а также задать наборы входных и выходных данных для тестирования схемы решения задачи. Элемент

<modules> является необязательным. Он указывает версии модулей для данной схемы решения задачи. Примеры описаний в формате JSON приведены на рис. 5.

```
{"orlando": {
  "Имя_модуля": {
        "dir": "директория_модуля",
    "exe": "исполняемый_файл"}},
  "repositories": [Репозитории],
  "require": {Список модулей с
             указанием версий}}
```

a)

```
{"orlando": {
  "fileio": "Входные_файлы>
             выходные_файлы"},
  "repositories": [Репозитории],
  "require": {Пакет_с_тестовыми
             _данными}}
```

b)

Рис. 5. Пример описания информации о модуле (a) и тестовых данных (b)
Fig. 5. An example of the description of information about the module (a) and test data (b)

Разработка модулей пакета. На рис. 6 показаны результаты оценки среднего времени работы разработчика пакета, затрачиваемого им на добавление или модификацию одного модуля с использованием автоматизации непрерывной интеграции в Orlando Tools (a) и путем неавтоматизированного применения сторонних систем поддержки такого процесса в системах управления workflow, таких как Condor DAGMan [30] (н/а). Данные результаты показывают существенное сокращение времени (с/в), затрачиваемого разработчиком в первом случае, когда этапы компиляции исполняемого кода модуля, тестирования его скомпилированной версии, ее размещение в репозитории и тестирование модуля в составе схемы решения задачи выполняются в Orlando Tools автоматически, без его прямого участия. Сокращение времени во многом обусловлено исключением накладных расходов, связанных с запуском и завершением работы сторонних систем непрерывной интеграции, преобразованием и передачей данных между ними.

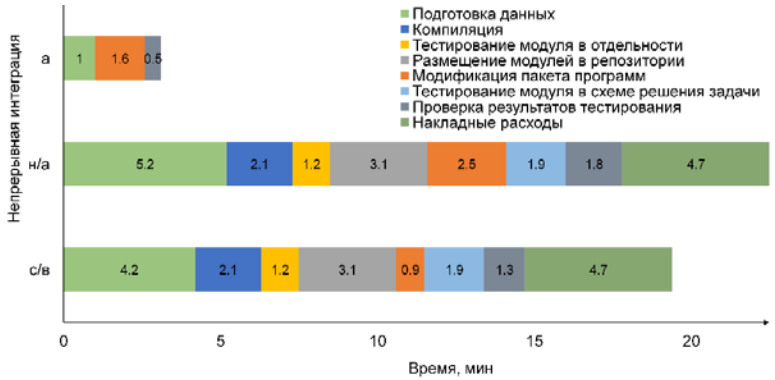

Рис. 6. Время выполнения непрерывной интеграции
Fig. 6. Continuous integration runtime

Прогнозирование времени выполнения схемы решения задачи. Схема решения задачи (рис. 3) выполнялась на десяти узлах двух вычислительных кластеров по двум сценариям: с оценкой времени выполнения ее модулей в узлах с помощью рассмотренной выше модели (о) и без такой оценки (б/о). Оценка времени выполнения модулей применялась для пропорционального распределения вычислительной нагрузки между узлами с учетом отличий их характеристик (узел кластера 1: 2 процессора AMD Opteron 6276, 16 ядер, 2.3 GHz, 64 GB оперативной памяти; узел кластера 2: 2 процессора Intel Xeon CPU X5670, 18 ядер, 2.1 GHz, 128 GB оперативной памяти). Числа узлов кластера 1 (кластера 2) в процессе эксперимента изменялось от 0/10 (10/0) до 10/0 (0/10). На рис. 7 представлены время решения задачи, средняя загрузка процессоров, ускорение и эффективность вычислений при

соотношении числа m узлов кластера 1 к числу n узлов кластера 2. Ускорение и эффективность вычислены относительно времени решения задач на 1 узле кластера 2.

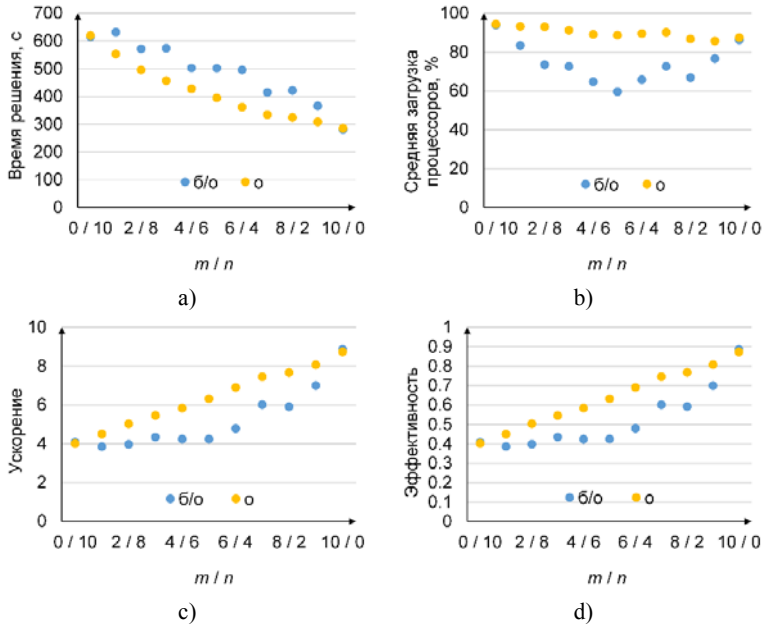

Рис. 7. Экспериментальные данные: время решения задачи (a), средняя загрузка процессоров (b), ускорение (c) и эффективность (d)

Fig. 7. Experimental data: problem-solving time (a), average processor load (b), speedup (c), efficiency (d)

Прогнозные оценки времени решения задачи, использованные в процессе распределения вычислительной нагрузки при разном соотношении узлов кластеров 1 и 2, показаны на рис. 8 в сравнении оценками пользователя, оценками, полученными на основе вычислительной истории и реальным временем решения задачи. Оценка пользователя, как правило, существенно завышена и основана на его практическом опыте решения задачи в однородной среде. Усредненная оценка, полученная на основе вычислительной истории, так же дает значительную погрешность, так как на основе такой истории сложно учесть различия в характеристиках узлов. Таким образом, наименьшую погрешность в большинстве случаев обеспечивает прогнозная оценка.

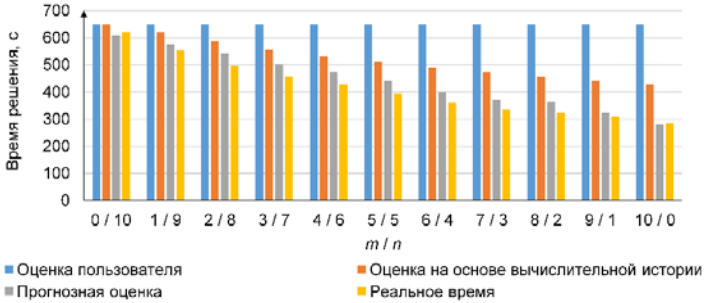

Рис. 8. Время решения задачи

Fig. 8. Problem-solving time

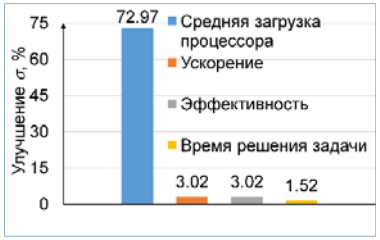

Рис. 9. Улучшение среднеквадратического отклонения критериев

Fig. 9. Improving the standard deviation of criteria

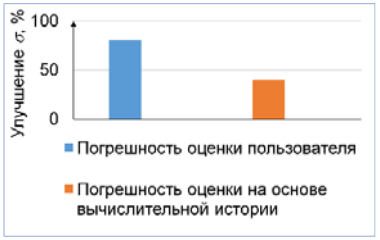

Рис. 10. Улучшение среднеквадратического отклонения погрешностей оценок

Fig. 10. Improving the standard deviation of evaluation errors

На рис. 9 приведено улучшение среднеквадратического отклонения σ критериев решения задачи при распределении нагрузки в ГРВС с учетом прогнозных оценок. Улучшение касается как эффективности использования ресурсов, так и процесса решения задачи. Рис. 10 показывает улучшение среднеквадратического отклонения σ погрешностей оценок времени решения задачи, полученных от пользователя и на основе вычислительной истории, в процентах. Расчеты выполнены в ИСКЦ СО РАН [31].

8. Заключение

В рамках предложенной системы непрерывной интеграции модулей РППП, в отличие от известных средств подобного назначения, реализуются как традиционные функции управления версиями программ, автоматизации их сборки и тестирования, так и новые функции извлечения и структуризации предметных знаний, унификации процессов сборки модулей как на выделенных серверах, так и на машинах разработчиков РППП путем использования специализированных виртуальных машин, синтеза тестовых схем решения задач на концептуальной модели и исследования свойств выполнения модулей относительно характеристик ГРВС. Результаты практического использования прототипа системы непрерывной интеграции показывают существенное улучшение различных показателей процесса выполнения вычислений в ГРВС.

Список литературы/References

[1] Il'in V.P., Skopin I.N. About performance and intellectuality of supercomputer modeling. Programming and Computer Software, vol. 42, no. 1, 2016, pp. 5-16.

[2] Massobrio R., Nesmachnow S., Tchernykh A., Avetisyan A., Radchenko G. Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities. Programming and Computer Software, vol. 44, no. 3, 2018, pp. 181-189.

[3] Deelman E., Peterka T., Altintas I., Carothers C.D., van Dam K.K., Moreland K., Parashar M., Ramakrishnan L., Taufer M., Vetter J. The future of scientific workflows. The International Journal of High Performance Computing Applications, vol. 32, no. 1, 2018, pp. 159-175.

[4] Krol M., Rene S., Ascigil O., Psaras I. ChainSoft: Collaborative Software Development using Smart Contracts. In Proc. of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, 2018, pp. 1-6.

[5] Sochat V. Containershare: Open Source Registry to build, test, deploy with CircleCI. The Journal of Open Source Software, vol. 3, no. 28, 2018, pp. 1-3.

[6] Soni M., Berg A.M. Jenkins 2.x Continuous Integration Cookbook. Packt Publishing, 2017, 438 p.

[7] Machiraju S., Gaurav S. Deployment via TeamCity and Octopus Deploy. In S. Machiraju, S. Gaurav. DevOps for Azure Applications. Apress, 2018, pp. 11-38.

[8] Beller M., Gousios G., Zaidman A. Oops, My Tests Broke the Build: An Explorative Analysis of Travis

CI with GitHub. In Proc, of the 14th International Conference on Mining Software Repositories, 2017, pp. 356-367.
[9] Gruver G. Start and Scaling Devops in the Enterprise. BookBaby, 2016, 100 p.
[10] Shahin M., Babar M.A., Zhu L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. IEEE Access, 2017, pp. 3909-3943.
[11] Wolff E. A Practical Guide to Continuous Delivery. Addison-Wesley, 2017, 265 p.
[12] CruiseControl.NET (online). Available at: http://sourceforge.net/projects/ccnet, accessed 10.12.2018.
[13] Apache Gump (online). Available at: https://gump.apache.org, accessed 10.12.2018.
[14] Conan C/C++ package manager (online). Available at: https://www.conan.io, accessed 10.12.2018.
[15] BuildMaster (online). Available at: https://inedo.com/buildmaster, accessed 10.12.2018.
[16] Heckel T. (2015) Meet Travis CI: Open Source Continuous Integration, InfoQ (online). Available at: https://www.infoq.com/news/2013/02/travis-ci, accessed 10.12.2018.
[17] Chacon S., Straub B. Pro git. Apress, 2014, 419 p.
[18] Barrett D., Silverman R., Byrnes R. SSH: The Secure Shell. O'Reilly, 2005, 672 p.
[19] Blum R. Linux Command Line and Shell Scripting Bible, Wiley, 2017, 816 p.
[20] Colvin H. VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox. CreateSpace Independent Publishing Platform, 2015, 70 p.
[21] Крупин А. Обзор пакета Parallels Remote Application Server для виртуализации рабочих мест: все включено. 3DNews – Daily Digital Digest (online). Доступно по ссылке: https://3dnews.ru/931400, дата обращения 10.12.2018 / Krupin A. Overview of the Package Parallels Remote Application Server for Workplace Virtualization: All-inclusive, 3DNews – Daily Digital Digest, 2016. Available at: https://3dnews.ru/931400, accessed 10.12.2018 (In Russian).
[22] Smith R. Docker Orchestration. Packt Publishing – ebooks Account, 2017, 284 p.
[23] Luksa M. Kubernetes in Action. Manning Publications, 2018, 624 p.
[24] Varlamov M.I., Turdakov D.Y. A survey of methods for the extraction of information from Web resources. Programming and Computer Software, vol. 42, no. 5, 2016, pp. 279-291.
[25] Feoktistov A., Kostromin R., Sidorov I.A., Gorsky S.A. Development of Distributed Subject-Oriented Applications for Cloud Computing through the Integration of Conceptual and Modular Programming. In Proc. of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, 2018, pp. 256-261.
[26] JFrog Artifactory (online). Available at: https://jfrog.com/artifactory, accessed 10.12.2018.
[27] Debian Package Tracking System – Advanced Packaging Tool (online). Available at: https://packages.qa.debian.org/a/apt.html, accessed 10.12.2018.
[28] Yum Package Manager (online). Available at: http://yum.baseurl.org, accessed 10.12.2018.
[29] Bychkov I.V., Oparin G.A., Tchernykh A.N., Feoktistov A.G., Gorsky S.A., Rivera-Rodriguez R. Scalable Application for Searching Global Minimum of Multiextremal Functions. Optoelectronics, Instrumentation and Data Processing, vol. 54, no. 1, 2018, pp. 83-89.
[30] Tannenbaum T., Wright D., Miller K., Livny M. Condor – A Distributed Job Scheduler. In Beowulf Cluster Computing with Linux, The MIT Press, 2002, pp. 307-350.
[31] Иркутский суперкомпьютерный центр СО РАН (online). Доступно по ссылке: http://hpc.icc.ru, дата обращения 10.12.2018 / Irkutsk Supercomputer center of SB RAS. Available at: http://hpc.icc.ru, accessed 10.12.2018.

Информация об авторах / Information about authors

Александр Геннадьевич ФЕОКТИСТОВ – кандидат технических наук, доцент, ведущий научный сотрудник лаборатории параллельных и распределенных вычислительных систем Института динамики систем и теории управления им. В.М. Матросова СО РАН. Основное направление исследований – теория и практика разработки методов и инструментальных программных средств организации распределенных вычислений для решения фундаментальных и прикладных ресурсоемких задач.

Alexander Gennadievitch FEOKTISTOV – Candidate of Technical Sciences, Associate Professor, Leading Researcher of the Laboratory of Parallel and Distributed Computing Systems of the Matrosov Institute of System Dynamics and Control Theory of SB RAS. The main line of his research is the theory and practice of developing methods and tool software for organizing distributed computing for solving fundamental and applied resource-intensive tasks.

Сергей Алексеевич ГОРСКИЙ – кандидат технических наук, научный сотрудник Института динамики систем и теории управления им. В.М. Матросова СО РАН. В число его научных интересов входят многоагентные системы, облачные вычисления, виртуализация, параллельная обработка, теория графов, булева алгебра, булевы функции.

Sergey Alekseevich GORSKY - Candidate of Technical Sciences, Researcher at the Matrosov Institute of System Dynamics and Control Theory of SB RAS. His research interests include multi-agent systems, cloud computing, virtualization, parallel processing, graph theory, Boolean algebra, Boolean functions.

Иван Александрович СИДОРОВ – кандидат технических наук, научный сотрудник Института динамики систем и теории управления им. В.М. Матросова СО РАН. Его научные интересы включают параллельное и распределенное программирование, облачные вычисления, виртуализацию.

Ivan Sergeevitch SIDOROV – Candidate of Technical Sciences, Researcher at the Matrosov Institute of System Dynamics and Control Theory of SB RAS. His research interests include parallel and distributed processing, cloud computing, virtualization.

Роман Олегович КОСТРОМИН – аспирант Института динамики систем и теории управления им. В.М. Матросова СО РАН. Его научные интересы включают распределенные прикладные программные пакеты, схемы решения проблем, многоагентное управление, отказоустойчивость.

Roman Olegovich KOSTROMIN – PhD student of the the Matrosov Institute of System Dynamics and Control Theory of SB RAS. His research interests include distributed application software packages, problem solving schemes, multi-agent management, and fault tolerance.

Евгений Сергеевич ФЕРЕФЕРОВ – кандидат технических наук с 2014 года, учёный секретарь Института динамики систем и теории управления им. В.М. Матросова СО РАН. В число его научных интересов входят распределенные вычислительные среды, имитационное моделирование, автоматизация разработки, информационные системы, приложения баз данных.

Evgeny Sergeevich FEREFEROV – Candidate of Technical Sciences since 2014, academic secretary of the the Matrosov Institute of System Dynamics and Control Theory of SB RAS. His research interests include distributed computing environments, simulation, development automation, information systems, database applications.

Игорь Вячеславович БЫЧКОВ – доктор наук, профессор, академик РАН, директор Института динамики систем и теории управления им. В.М. Матросова СО РАН. Основные научные интересы: интеллектуальная технология обработки пространственно-распределенных данных, технология автоматизации создания программных систем на основе метаописаний.

Igor Vyacheslavovich BYCHKOV – Doctor of Science, Professor, Academician of the Russian Academy of Sciences, Director of the Matrosov Institute of System Dynamics and Control Theory of SB RAS. His research interests include intellectual technology for processing spatially distributed data, technology for automating the creation of software systems based on meta descriptions.