Инструментальный комплекс поддержки высокопроизводительных вычислений в предметно-ориентированных гетерогенных средах

С.А. Горский[1], Р.О. Костромин[1], А.Г. Феоктистов[1], И.В. Бычков[1]

[1]Институт динамики систем и теории управления им. В.М. Матросова СО РАН, Лермонтова 134, Иркутск, Россия, 664033

Аннотация. В настоящее время современные средства разработки распределенных пакетов прикладных программ не обеспечивают поддержку непрерывной интеграции прикладного программного обеспечения в полной мере. Это негативным образом отражается на качестве программного обеспечения и увеличивает как сроки его разработки, так и время проведения экспериментов. В статье, предложен новый подход, обеспечивающий внедрение непрерывной интеграции как прикладного, так и системного программного обеспечения в процесс разработки пакетов. Практическая значимость результатов исследования обусловлена уменьшением числа ошибок и сбоев при разработке и применении пакетов. В свою очередь, это значительно сокращает время, необходимое для проведения масштабных экспериментов, и повышает эффективность использования ресурсов среды.

1. Введение

Научное приложение (пакет прикладных программ) представляет собой комплекс программ (модулей), предназначенных для решения определенного класса задач в конкретной предметной области. В таком приложении вычислительный процесс описывается схемой решения задачи, определяющей информационно-логические связи между модулями в процессе вычислений. В качестве одной из разновидностей схемы решения задачи можно рассматривать описание рабочего процесса (workflow [1]). Быстрое развитие технологий распределенных вычислений привело к значительным изменениям в архитектуре пакетов прикладных программ [2]. Она сохранила модульную структуру, но стала распределенной и ориентированной на гетерогенные распределенные вычислительные среды (ГРВС).

Использование гибридных моделей облачных и грид-вычислений в ГРВС [3] повышает эффективность и гибкость использования ресурсов, однако в то же время порождает новые проблемы для систем управления вычислениями распределенных пакетов прикладных программ (РППП) в процессе решения крупномасштабных задач. Эти проблемы обусловлены как существующими различиями в моделях облачных и грид-вычислений [4-6], так и постоянным изменением программно-аппаратных и информационных ресурсов среды, влекущим за собой необходимость решения задач реконфигурации вычислительных сред пакетов, модификации их функционального наполнения и/или разработки нового программного обеспечения, 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. Вышеперечисленные задачи лежат в той или иной мере в области непрерывной интеграции программного обеспечения [7].

Проведенный анализ известных универсальных инструментальных средств, поддерживающих функции непрерывной интеграции программных проектов (например, CircleCI [8], Jenkins [9], TeamCity [10], Travis [11], GitLab [12] и др. [13, 14]), показал, что они не обеспечивают нужный уровень поддержки непрерывной интеграции необходимый для средств создания научных приложений [15]. В частности, такие средства зачастую не готовы в полной мере поддерживать сложный процесс непрерывной интеграции в сочетании с концептуальным моделированием, традиционно применяемым в такого рода приложениях, и комплексным использованием разнородных и сложно структурированных знаний (предметно-ориентированных, а также специализированных относительно свойств программно-аппаратной инфраструктуры среды и административных политик в ее узлах). В статье предложен новый подход к обеспечению непрерывной интеграции функционального наполнения РППП, базирующийся на слиянии методологии построения таких пакетов с современной практикой разработки программного обеспечения на основе его непрерывной интеграции с использованием предметно-ориентированных знаний.

2. Разработка распределенных пакетов прикладных программ

Авторами разработан инструментальный комплекс Orlando Tools [16] для поддержки проведения крупномасштабных научных и прикладных исследований. Данный комплекс позволяет разрабатывать РППП и обеспечивает построение предметно-ориентированных вычислительных сред для этих пакетов. В рамках таких сред могут интегрироваться различные вычислительные инфраструктуры, поддерживающие как облачные, так и грид-вычисления. Средства управления вычислениями в Orlando Tools интегрированы с мультиагентной системой [17] и системой метамониторинга ГРВС [17]. Архитектура Orlando Tools, представленная в статье, расширена новой подсистемой непрерывной интеграции функционального и системного наполнения РППП. Она включает следующие основные компоненты: интерфейс пользователя, конструктор модели, базу знаний, исполнительную подсистему, средства непрерывной интеграции и базу расчетных данных. Формирование вычислительных инфраструктур пакетов осуществляется с помощью специального сервера Orlando Tools. Пользовательский интерфейс обеспечивает доступ пользователя к компонентам Orlando Tools. Он реализован в виде веб-приложения с использованием языков JavaScript и PHP.

Конструктор модели предназначен для декларативной спецификации алгоритмических знаний, а также знаний о программно-аппаратных и административных характеристиках узлов ГРВС. Алгоритмические знания включают вычислительные знания о программных модулях для решения задач в предметных областях РППП, схемные знания о модульной структуре модели и алгоритмов, продукционные знания для поддержки принятия решений по выбору оптимальных алгоритмов решения задачи в зависимости от состояния ресурсов среды. Алгоритмические знания создаются разработчиками РППП, представляются ими в виде вычислительных моделей в текстовом или графическом виде и хранятся в базе знаний Orlando Tools. Знания о программно-аппаратных и административных характеристиках узлов ГРВС извлекаются администраторами среды и хранятся в базе знаний системы метамониторинга. Описание рассмотренных категорий знаний в Orlando Tools детально рассмотрено в [17, 18].

Исполнительная подсистема включает интерпретатор схем решения задач, планировщик вычислений и генератор заданий. Интерпретатор осуществляет выполнение схем решения задач, включая обработку управляющих конструкций ветвления и циклов. Планировщик производит декомпозицию схем решения задач на подсхемы с целью оптимизации распределения вычислительной и коммуникационной нагрузки на ресурсы ГРВС. Декомпозиция может быть выполнена как в статическом режиме до начала вычислений, так и

динамически в процессе вычислений. После декомпозиции схемы решения задачи специальный генератор формирует задания для выполнения ее подзадач на выбранных ресурсах.

База расчетных данных предназначена для хранения постановок задач, схем их решения, исходных данных и результатов вычислений, а также поддержки проведения тестирования новых или модифицированных модулей РППП как в отдельности, так и в составе схем.

Сервер Orlando Tools поддерживает пользовательский веб-интерфейс и реализует функции исполнительной подсистемы. Он позволяет включить в вычислительную инфраструктуру пакета следующие компоненты: кластеры с локальными системами управления заданиями (SLURM, HTCondor или PBS Torque); отдельные узлы с операционной системой Linux с непереносимым программным обеспечением; кластеры выделенных (облачных) или невыделенных ресурсов, создаваемые с использованием специальных виртуальных машин Orlando Tools, в которых размещаются модули пакетов и системное программное обеспечение для управления процессами их выполнения; удаленные ресурсы, подключаемые через API-интерфейс грид или веб-сервисов.

3. Непрерывная интеграция в Orlando Tools

Процесс разработки функционального наполнения РППП, как правило, включает следующие основные этапы: разработка и модификация программного кода модулей, их компиляция, отладка и тестирование в разнородной среде, размещение скомпилированных версий модулей в репозитории или установка модулей на ресурсы, модификация вычислительной модели пакета, отладка и тестирование схем решения задач с использованием новых или модифицированных версий модулей. Многократное повторение этих этапов работ разработчиком РППП актуализует создание инструментальных средств для автоматизации процесса непрерывной интеграции функционального наполнения таких пакетов.

На рисунке 1 представлены компоненты Orlando Tools, связанные со схемой организации непрерывной интеграции при разработке РППП. В Orlando Tools используется два типа репозиториев: репозитории модулей и репозитории РППП.

Модули в Orlando Tools являются в основном исполняемыми программами. Для организации непрерывной интеграции на уровне отдельных модулей существуют хорошо развитые инструменты такие как GitLab. Большинство этих средств использует протокол Git [19], что позволяет унифицировать работу с ними до определенного уровня. Однако более глубокая интеграция вынуждает использовать специализированные API данных систем. По этой причине после анализа доступных на текущий момент средств непрерывной интеграции был выбран наиболее развитый инструмент GitLab для интеграции с Orlando Tools.

Интеграция с GitLab также позволяет объединить все средства разработки РППП в одном месте для случаев, когда разработчику требуется внести незначительные правки в исходный код модулей, и ему не требуются дополнительные возможности специализированных средств разработки. Непрерывная интеграция в Orlando Tools подразумевает, что каждый модуль подверженный частым изменениям должен иметь хотя бы один GitLab репозиторий. Дополнительно главный репозиторий модуля хранит информацию об интерфейсе модуля и инструкции по его установке на ресурсы вычислительной среды. Можно указать дополнительные репозитории, с которыми связан главный репозиторий модуля, выстраивая дерево их зависимостей.

Бесплатная версия GitLab не позволяет проводить мониторинг за репозиториями нижних уровней для запуска процесса непрерывной интеграции. В нее включена только возможность запуска непрерывной интеграции в репозиториях верхних уровней посылкой специальных сигналов, что не всегда удобно, так как в общем случае от одного репозитория нижнего уровня может зависеть несколько репозиториев верхнего уровня, или в листьях дерева репозиториев могут находится репозитории сторонних библиотек. В дальнейшем планируется организовать мониторинг изменений, происходящих в репозиториях модуля, и позволить запускать цепочки непрерывной интеграции, проходящие от листьев к корню дерева зависимостей репозиториев.

Второй тип репозиториев хранит информацию о самом РППП и напрямую не связан с организацией непрерывной интеграции. Он предназначен для хранения описания РППП по

версиям. Дополнительно репозиторий РППП хранит тестовые данные для схем пакета так как они могут существенно различаться от версии к версии. Хранение РППП по версиям позволяет вернутся к старым версиям пакета при необходимости проведения повторных расчетов.

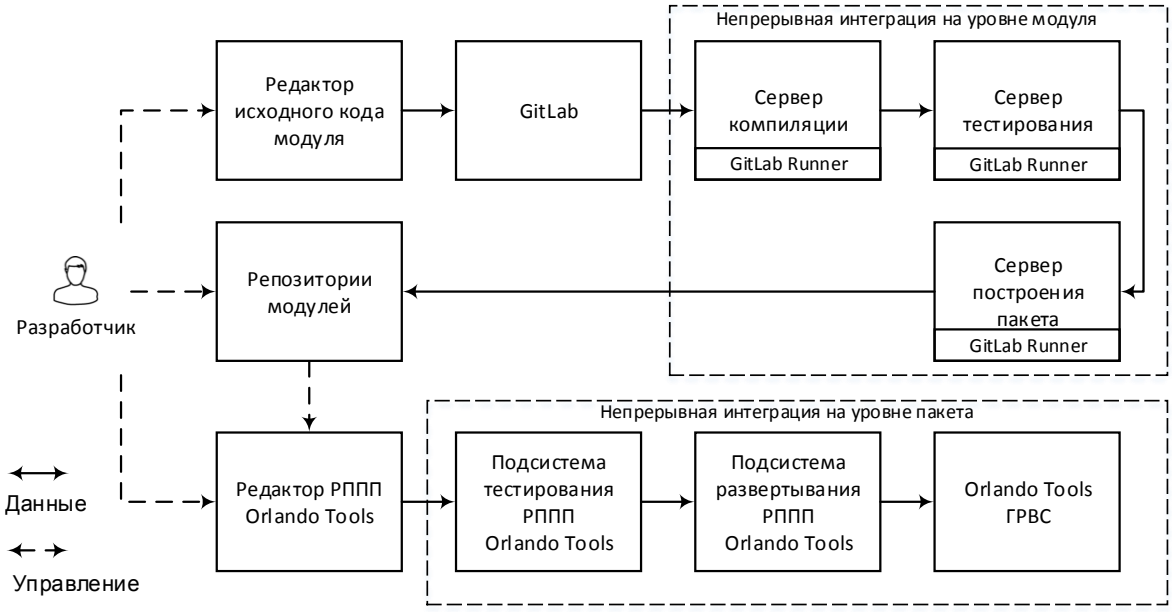

Рисунок 1. Схема непрерывной интеграции.

Непрерывная интеграция на уровне РППП реализуется в самом Orlando Tools и включает автоматизацию и проверку корректности прохождения следующих этапов: отслеживание изменений в репозиториях модулей пакета с возможностью настройки цепочек зависимостей; установку модулей на тестовые ресурсы; запуск тестов модулей после установки; запуск тестов РППП и последующую установку модулей на ресурсы ГРВС; запуск тестов модулей после установки. Схема прохождения этих этапов представлена на рисунке 1.

Процесс непрерывной интеграции в РППП может быть инициирован модификацией репозитория одного из модулей или описания РППП. Результатом работы данной схемы будет или сообщение разработчику об ошибках, возникших на том или ином этапе разработки, или готовая к использованию версия РППП установленная на вычислительные ресурсы ГРВС. Перед установкой новых версий модулей на вычислительные ресурсы проводится проверка не задействованы ли эти ресурсы в вычислительном процессе. Предложенная схема позволяет снизить трудозатраты на организацию непрерывной интеграции в РППП, тем самым повысив эффективность разработки РППП в Orlando Tools.

Orlando Tools ориентирован на предметных специалистов, что накладывает дополнительные требования к интерфейсам разработчика и пользователя. В связи с этим разработан метод модификации языка описания модели предметной области, позволяющий адаптировать синтаксис языка к требованиям предметных специалистов. Ниже приведен пример такой адаптации. Синтаксис спецификаций основных объектов модели, комплексирующей знания для описания процессов непрерывной интеграции функционального наполнения РППП со знаниями, представленными вычислительными моделями пакетов, приведен на рисунке 2.

```
Проект Имя_РППП
Параметр Имя_параметра Расширение;
Операция Имя_операции модуль | задача Имя_модуля | Имя_задачи (Входные_параметры>
Выходные_параметры);
Модуль Имя_модуля (Входные_параметры > Выходные_параметры)
Имя_репозитория Имя_проекта;
Задача Имя_задачи (Входные_параметры > Выходные_параметры);
```

Рисунок 2. Синтаксис спецификаций основных объектов концептуальной модели.

```
Проект grad
Параметр Function txt; Параметр Params txt; Параметр StartPoints txt
[ArrayCount_0_s]; Параметр EndPoints txt [ArrayCount_0_s];
Параметр PointCount txt; Параметр Result txt; Параметр Dimensions txt;
Параметр Limits txt; Параметр ArrayCount txt;
Модуль generate (Dimensions,Limits,PointCount,Function,Params>
ArrayCount,StartPoints[]) gitorlando generate.master;
Модуль gradient (Dimensions,Function,StartPoints,Params>EPoints) gitorlando
gradient.master;
Модуль complete (Dimension,EndPoint[],ArrayCount>Result) gitorlando
complete.master;
Операция Generate модуль generate
(Dimensions,Limits,PointCount,Function,Params>ArrayCount,StartPoints);
Операция Gradient модуль gradient
(Dimensions,Function,StartPoints,Params>EndPoints) [ArrayCount_0_s];
Операция Complete модуль complete (Dimensions,EndPoints,ArrayCount>Result);
Задача grad (Dimensions,Limits,PointCount,Function,Params>Result);
```

Рисунок 3. Спецификация на языке CDML.

```
<pack> <project> <name>grad</name>
<parameter> <name>Function</name> <extention>txt</extention> </parameter>
<parameter> <name>Params</name> <extention>txt</extention> </parameter>
<parameter> <name>StartPoints</name> <extention>txt</extention>
<list>ArrayCount_0_s</list> </parameter>
<parameter> <name>EndPoints</name> <extention>txt</extention>
<list>ArrayCount_0_s</list> </parameter>
<parameter> <name>PointCount</name> <extention>txt</extention> </parameter>
<parameter> <name>Result</name> <extention>txt</extention> </parameter>
<parameter> <name>Dimensions</name> <extention>txt</extention> </parameter>
<parameter> <name>Limits</name> <extention>txt</extention> </parameter>
<parameter> <name>ArrayCount</name> <extention>txt</extention> </parameter>
<operation> <name>Generate</name>
<input>Dimensions,Limits,PointCount,Function,Params>ArrayCount,StartPoints</input>
<condition>1</condition> <while>0</while> <type>module</type>
<module>generate</module> </operation>
<operation> <name>Gradient</name>
<input>Dimensions,Function,StartPoints,Params>EndPoints</input>
<condition>1</condition> <while>0</while> <type>module</type>
<module>gradient</module> <list>ArrayCount_0_s</list> </operation>
<operation> <name>Complete</name>
<input>Dimensions,EndPoints,ArrayCount>Result</input> <condition>1</condition>
<while>0</while> <type>module</type>
<module>complete</module> </operation>
<module> <name>generate</name> <parameter>Dimensions,Limits,PointCount,Function,
Params>ArrayCount,StartPoints[]</parameter> <signatura>gen %%3 %%1 %7 %2 %%6 %4
%5</signatura> <cores>1</cores> <walltime>10</walltime> <type>node</type>
<repo>gitorlando generate.master</repo> <server> <servername>matrosov</servername>
<folder>/home/sgorsky/grad2016/</folder> </server> </module>
<module> <name>gradient</name> <parameter>Dimensions,Function,StartPoints,Params>
EPoints</parameter> <signatura>grad %%1 %3 %5 %4 %2</signatura> <cores>32</cores>
<walltime>10</walltime> <type>node</type> <repo>gitorlando gradient.master</repo>
<server> <servername>matrosov</servername>
<folder>/home/sgorsky/grad2016/</folder> </server> </module>
<module> <name>complete</name> <parameter>Dimension,EndPoint[],ArrayCount>
Result</parameter> <signatura>res %%1 %2 %%3 %4</signatura> <cores>1</cores>
<walltime>10</walltime> <type>node</type> <repo>gitorlando complete.master</repo>
<server> <servername>matrosov</servername>
<folder>/home/sgorsky/grad2016/</folder> </server> </module>
<task> <name>grad</name>
<parameter>Dimensions,Limits,PointCount,Function,Params>Result</parameter>
<plan_type>1</plan_type> <plan>Generate,Gradient,Complete</plan> </task>
</project></pack>
```

Рисунок 4. Спецификация на языке XML.

Ключевые слова **Параметр**, **Операция**, **Модуль** и **Задача** соответствуют следующим объектам вычислительной модели: параметр, операция, модуль и схема решения задач. Ключевое слово **Проект** отмечает начало нового проекта в описании РППП. Элементы спецификации **Имя_репозитория** и **Имя_проекта** поддерживают процесс непрерывной интеграции. Они указывают имя хранилища и имя проекта модуля в хранилище соответственно. В спецификации модуля эти элементы содержат информацию о местонахождении этого модуля в хранилище и его зависимости от других модулей. Дополнительная информация о модуле, необходимая для его запуска и тестирования, хранится в репозитории модуля в специальных файлах. На рисунках 3 и 4 приведены примеры спецификаций и результат их конвертирования в модель пакета для решения задач поиска глобального минимума многоэкстремальных функций [20].

В данной спецификации на языке CDML отсутствует информация о модуле (строка запуска, тип модуля и прочая информация). Это сокращает описание РППП и делает его понятней для разработчика. Вся дополнительная информация хранится в репозитории модуля. Она считывается из репозитория при разборе спецификации и помещается в базу знаний Orlando Tools. Модификация описания модуля возможна либо напрямую в репозитории или через редактор РППП. Построение схем решения задач производится автоматически по непроцедурной постановке задачи с учетом информационно-логических связей между модулями по их входным и выходным параметрам.

4. Экспериментальный анализ

Использование предложенной схемы непрерывной интеграции позволяет автоматизировать процессы развертывания РППП на новых ресурсах ГРВС.

Эксперименты по использованию непрерывной интеграции в Orlando Tools проведены применительно к РППП, предназначенному для решения задачи поиска глобального минимума многоэкстремальной функции. Развертывание пакета и решение задачи было проведено с использованием непрерывной интеграции и без нее.

Экспериментальные результаты показали существенное ускорение вычислений и сокращение временных затрат конечных пользователей РППП на подготовку расчетов. Использование непрерывной интеграции позволило сократить время подготовки эксперимента и решения задачи соответственно на 79% и 14%. Все расчеты выполнены в Центре коллективного пользования (ЦКП) «Иркутский суперкомпьютерный центр СО РАН» (ИСКЦ) [21].

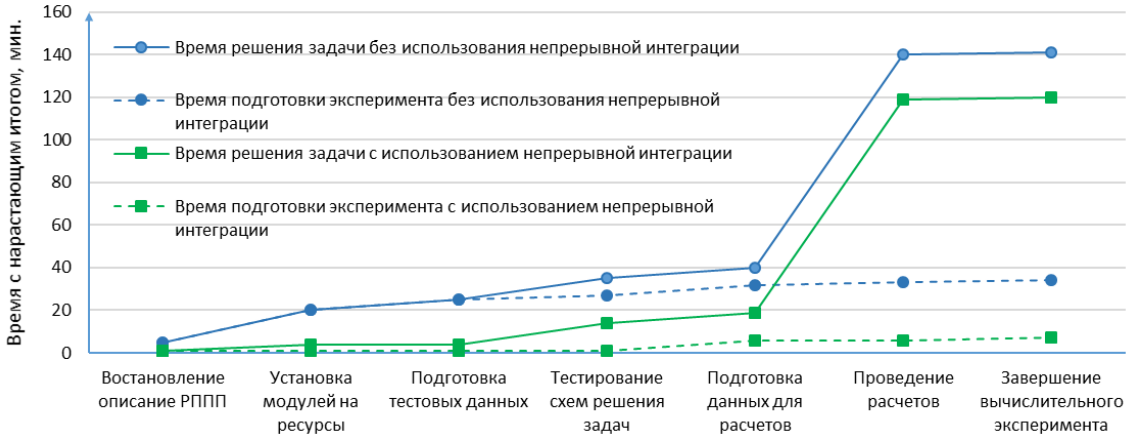

Рисунок 5. Затраты времени на развертывание пакета, подготовку и проведение расчетов.

5. Заключение

Предложена новая схема поддержки непрерывной интеграции функционального наполнения РППП с целью выявления его потенциальных проблем, возникающих в ГРВС. В рамках данной схемы пакеты создаются с помощью инструментального комплекса Orlando Tools, применяемого в ЦКП ИСКЦ для организации параллельных и распределенных вычислений.

Процесс разработки РППП характеризуется частыми изменениями программных модулей пакетов. Применение предложенной схемы обеспечивает существенное снижение трудозатрат на интеграцию прикладного программного обеспечения разрабатываемых пакетов, повышение его надежности и сокращение времени проведения крупномасштабных экспериментов в целом.

6. Благодарности

Исследование выполнено в рамках проекта IV.38.1.1 программы фундаментальных исследований СО РАН, а также поддержано РФФИ, проект № 19-07-00097-а.

7. Литература

[1] Barker, A. Scientific workflow: a survey and research directions / A. Barker, J. Hemert // Lecture Notes in Computer Science. – 2007. – Vol. 4967. – P. 746-753. DOI: 10.1007/978-3-540-68111-3_78.
[2] Ильин, В.П. Технологии вычислительного программирования / В.П. Ильин, И.Н. Скопин // Программирование. – 2011. – Т. 37, № 4. – С. 53-72.
[3] Feoktistov, A. Orlando Tools for Scientific Applications Development and Use: Convergence of Grid and Cloud Computing / A. Feoktistov, S. Gorsky, I. Sidorov, R. Kostromin, A. Edelev, L. Massel // Суперкомпьютерные дни в России: Труды международной конференции. – М.: Изд-во МГУ, 2018. – С. 378-389.
[4] Rings, T. Grid and cloud computing: opportunities for integration with the next generation network / T. Rings, G. Caryer, J. Gallop, J. Grabowski, T. Kovacikova, S. Schulz, I. Stokes-Rees // Journal of Grid Computing. – 2009. – Vol. 7(3). – P. 1-19. DOI: 10.1007/s10723-009-9132-5.
[5] Roman, R. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges / R. Roman, J. Lopez, M. Mambo // Future Generation Computer Systems. – 2018. – Vol. 78. – P. 680-698. DOI: 10.1016/j.future.2016.11.009.
[6] Varghese, B. Next generation cloud computing: New trends and research directions / B. Varghese, R. Buyya // Future Generation Computer Systems. – 2018. – Vol. 79. – P. 849-861. DOI: 10.1016/j.future.2017.09.020.
[7] Duvall, P.M. Continuous Integration: Improving Software Quality and Reducing Risk / P.M. Duvall, S. Matyas, A. Glover // Addison Wesley Professional, 2007. – 336 p.
[8] Sochat, V. Containershare: Open Source Registry to build, test, deploy with CircleCI / V. Sochat // The Journal of Open Source Software. – 2018. – Vol. 3(28). – P. 1-3. DOI: 10.21105/joss.00878.
[9] Soni, M. Jenkins 2.x Continuous Integration Cookbook / M. Soni, A.M. Berg // Packt Publishing, 2017. – 438 p.
[10] Machiraju, S. Deployment via TeamCity and Octopus Deploy / S. Machiraju, S. Gaurav // DevOps for Azure Applications, 2018. – P. 11-38. DOI: 10.1007/978-1-4842-3643-7_2.
[11] Beller, M. Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub / M. Beller, G. Gousios, A. Zaidman // Proceedings of the 14th International Conference on Mining Software Repositories, 2017. – P. 356-367. DOI: 10.1109/MSR.2017.62.
[12] Gruver, G. Start and Scaling Devops in the Enterprise / G. Gruver – BookBaby, 2016. – 100 p.
[13] Shahin, M. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices / M. Shahin, M.A. Babar, L. Zhu // IEEE Access. – 2017. – Vol. 5. – P. 3909-3943. DOI: 10.1109/ACCESS.2017.2685629.
[14] Wolff, E.A. Practical Guide to Continuous Delivery // E.A Wolff – Addison-Wesley Professional, 2017. – 265 p.
[15] Феоктистов, А.Г. Непрерывная интеграция функционального наполнения распределенных пакетов прикладных программ / А.Г. Феоктистов, С.А. Горский, И.А. Сидоров, Р.О. Костромин, Е.С. Фереферов, И.В. Бычков // Труды ИСП РАН. – 2019. – Том 31, № 2. – С. 83-96.

[16] Феоктистов, А.Г. Язык спецификации вычислительных моделей в масштабируемых пакетах прикладных программ / А.Г. Феоктистов, С.А. Горский // Современные наукоемкие технологии. – 2016. – № 7. – С. 84-88.
[17] Бычков, И.В. Мультиагентное управление вычислительной системой на основе метамониторинга и имитационного моделирования / И.В. Бычков, Г.А. Опарин, А.Г. Феоктистов, И.А. Сидоров, В.Г. Богданова, С.А. Горский // Автометрия. – 2016. – Т. 52, № 2. – С. 3-9. DOI: 10.15372/AUT20160201.
[18] Бычков, И.В. Мультиагентный алгоритм распределения вычислительных ресурсов на основе экономического механизма регулирования их спроса и предложения / И.В. Бычков, Г.А. Опарин, А.Г. Феоктистов, А.Н. Кантер // Вестник компьютерных и информационных технологий. – 2014. – № 1. – С. 39-45. DOI: 10.14489/vkit.2014.01.pp. 039-045.
[19] Git [Электронный ресурс]. – Режим доступа: https://git-scm.com/ (30.11.2019).
[20] Бычков, И.В. Масштабируемое приложение для поиска глобальных минимумов многоэкстремальных функций / И.В. Бычков, Г.А. Опарин, А.Н. Черных, А.Г. Феоктистов, С.А. Горский, Р. Ривера-Родригес // Автометрия. – 2018. – Т. 54, № 1. – С. 98-105. DOI: 10.15372/AUT20180113.
[21] Иркутский суперкомпьютерный центр СО РАН [Электронный ресурс]. – Режим доступа: http://hpc.icc.ru (30.11.2019).

Toolkit for supporting high-performance computing in subject-oriented heterogeneous environments

S.A. Gorsky[1], R.O. Kostromin[1], A.G. Feoktistov[1], I.V. Bychkov[1]

[1]Matrosov Institute for System Dynamics and Control Theory SB RAS, Lermontov street 134, Irkutsk, Russia, 664033

Abstract. Nowadays, modern tools for developing scientific applications for distributed computing systems (distributed applied software packages) do not fully support continuous integration of applied software. This negatively affects the quality of the software and increases both the software development time and the experiment carrying out time. In the paper, we propose a new approach that ensures the continuous integration of both applied and system software into the package development process. The practical significance of the study results is due to a decrease in the number of errors and failures in the development and application of packages. In turn, this significantly reduces the time needed for carrying out large-scale experiments and increases the efficiency of the environment resource use.