Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки. 2015. Т. 19, № 4. С. 785–798
ISSN: 2310-7081 (online), 1991-8615 (print) doi: http://dx.doi.org/10.14498/vsgtu1437

Математическое моделирование, численные методы и комплексы программ

УДК 004.434; 004.453

ИНСТРУМЕНТАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РАЗРАБОТКИ И ПОДДЕРЖКИ ИСПОЛНЕНИЯ ПРИЛОЖЕНИЙ НАУЧНЫХ ВЫЧИСЛЕНИЙ В КЛАСТЕРНЫХ СИСТЕМАХ

Ю. С. Артамонов, С. В. Востокин

Самарский государственный аэрокосмический университет
имени академика С. П. Королёва (национальный исследовательский университет),
Россия, 443086, Самара, Московское ш., 34.

Аннотация

Мотивация: Для разработки приложений научных вычислений существует множество различных инструментов. Большинство из них ориентированы на сам процесс написания программ, но часто требуются приложения для организации процесса вычислений и поддержки командной разработки. Описана специфика разработки приложений научной направленности, сделан акцент на характерных проблемах разработки такого ПО. **Классификация систем управления вычислительными задачами:** Приводится классификация систем по способу организации вычислений и уровню абстракции вычислений от физического оборудования. **Инструменты разработки Templet:** Рассматривается инструментарий для разработки приложений, включающий в себя библиотеки параллельных вычислений, сервис запуска и отслеживания задач, подсистему мониторинга состояния кластера. Тесное взаимодействие инструментов позволяет эффективно организовать работу команды над приложением научной направленности. **Решение прикладных задач при помощи инструментов Templet:** Инструментарий применяется для решения практических задач в области моделирования поведения многомерных динамических систем. Показан подход, позволяющий разделить работу над приложением на системный и прикладной уровни. **Заключение:** Сделан вывод о возможностях применения техник проектирования и преимуществах, которые даёт использование инструментария.

Образец для цитирования

Артамонов Ю. С., Востокин С. В. Инструментальное программное обеспечение для разработки и поддержки исполнения приложений научных вычислений в кластерных системах // *Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки*, 2015. Т. 19, № 4. С. 785–798. doi: 10.14498/vsgtu1437.

Сведения об авторах

Юрий Сергеевич Артамонов (artamonov@about.me), аспирант, каф. информационных систем и технологий.

Сергей Владимирович Востокин (д.т.н., проф.; easts@mail.ru; автор, ведущий переписку), профессор, каф. информационных систем и технологий.

Ключевые слова: инструмент, сервис, библиотека, параллельное программирование, окружение, запуск, мониторинг, вычисления, кластер.

doi: http://dx.doi.org/10.14498/vsgtu1437

Мотивация. Для разработки программ существует множество инструментов — среды разработки, компиляторы и трансляторы, библиотеки, парсеры, отладчики и многое другое. Практически все они ориентированы на создание конечного продукта и лишь немногие — на итеративную разработку и поддержку. Отличительными особенностями приложений для научных вычислений являются, с одной стороны, короткий жизненный цикл приложения, при котором часто не требуется отдельная документация, тестирование в различных ОС, оформление графического интерфейса и другие финальные этапы разработки программного обеспечения (ПО), с другой — большое количество вносимых изменений, требующее запуска приложения и получения результатов [1, p. 50–54]. Программы научных вычислений очень специализированы и довольно сильно зависят от целевого окружения (кластера, сети, систем хранения данных), что делает разработку таких приложений непохожей на разработку системного и прикладного ПО.

Помимо специфики самих приложений стоит отметить ряд характерных проблем, возникающих в процессе работы над проектом:

1) растущая сложность алгоритмов и архитектуры приложений;
2) окружения, в которых производится запуск, зачастую имеют нестандартный набор ПО и периферийных устройств;
3) сложный и длительный процесс развёртывания ПО в целевом окружении;
4) для тестирования может потребоваться производительное окружение;
5) оборудование и инфраструктура могут быть уникальны и их нельзя воссоздать для тестирования;
6) необходимо соблюдать регламент работы с целевым окружением, определённое время работы на кластере и лимит нагрузки;
7) доступ к окружению может быть ограничен локальной сетью организации.

Под окружением здесь понимается некоторый программно-аппаратный комплекс: совокупность физических или виртуальных ЭВМ, каналов связи, периферийных устройств и ПО, необходимого для работы системы. Например, это может быть кластер серверов, единичный сервер или же Desktop Grid-система.

Процесс организации работы над приложением также может быть сложным, поскольку одно приложение может разрабатываться целым коллективом. Причём командная разработка всё чаще требуется при работе над приложениями научных вычислений. Во многом это обусловлено тем, что команда может разрабатывать многие модули и подсистемы параллельно, к тому же в команде часто требуются как специалисты по предметной области, так и системные программисты, специалисты по параллельным вычислениям.

Отметим наиболее важные аспекты командной работы над приложениями:

1. Ни один крупный проект сейчас не может обойтись без использования систем контроля версий (VCS), ведь в процессе разработки код

проекта претерпевает значительные изменения, их все требуется отслеживать. Обязательно должна быть возможность вернуться к более ранней версии [2].

2. В проекте должен быть простой и эффективный механизм сборки приложения без использования графических утилит, поддерживающий управление зависимостями и различные целевые архитектуры [3]. Следует отметить, что для вычислений всё чаще используются языки, которым не требуется компиляция в машинно-зависимый код: Java, Python, R, Lisp, Scheme. Использование этих языков существенно упрощает работу над проектом, но не стоит забывать о накладных расходах при исполнении в виртуальной машине (на работу интерпретатора).
3. Поскольку тестовый запуск может требовать значительных ресурсов, большое значение приобретают модульные и интеграционные тесты [4]. Причём такие тесты должны запускаться довольно часто и, по возможности, автоматически при поступлении новых изменений от участников проекта.
4. Крайне важно разделить тестовые запуски и запуски вычислений для получения конечных результатов. Следует организовать тестовое окружение, если это возможно, и тестовые запуски проводить в нём. Запуск приложения на суперкомпьютере для целей тестирования и отладки требует части ценных разделяемых ресурсов, что нерационально, кроме того, результаты тестирования могут быть получены с существенной задержкой в случае загруженности основных ресурсов.

Для решения этих проблем разработано множество инструментов, позволяющих организовать работу над проектом, упростить запуск приложений и обработку результатов. В этой работе речь пойдёт о системах управления вычислительными задачами как основном инструменте ресурсоёмких научных вычислений.

1. Классификация систем управления вычислительными задачами. Системы управления вычислительными задачами можно разделить на несколько специализированных классов:

1) низкоуровневые системы управления распределёнными ресурсами (пакетные системы);
2) Desktop Grid-системы;
3) системы виртуализации вычислительных ресурсов;
4) высокоуровневые платформы распределённых вычислений.

Низкоуровневые системы управления распределёнными ресурсами призваны утилизировать доступные вычислительные ресурсы и максимизировать пропускную способность, управляя процессом назначения физических ресурсов пользовательским задачам. Пакетные системы управляют такими физическими ресурсами, как процессорное время, оперативная память, дисковое пространство и пропускная способность сети. К таким системам относятся [5]: Platform LSF, Portable Batch System (PBS), Sun Grid Engine (SGE), IBM Load Leveler и Condor. Пакетные системы обычно имеют следующие подсистемы:

1) подсистема управления задачами;
2) подсистема управления физическими ресурсами;
3) подсистема планирования и управления очередями.

Подсистема управления задачами — интерфейс для пользователей, которые взаимодействуют с системой, запрашивая ресурсы для задач и контролируя статус их исполнения. Подсистема управления физическими ресурсами решает две основных задачи: контроль использования ресурсов и сбор статистики их использования. В пакетных системах назначение ресурсов для задач зависит от политики планирования и использования ресурсов. Эта политика определяется администратором системы и является конфигурацией для подсистемы планирования и управления очередями. Основные достоинства пакетных систем:

1) пригодны для организации вычислений на различном оборудовании и операционных системах;
2) хорошо подходят для большого числа перезапусков задач;
3) позволяют организовать работу множества пользователей;
4) документированы, широко известны и повсеместно применяются.

Недостатки:

1) требуется обучение пользователей, поскольку им необходимо знать особенности физического оборудования кластера для запуска задач, такие как число ядер процессоров, оперативная память, скорость чтения и записи данных, топология сети;
2) пакетные задания трудно отлаживать;
3) зависшие задачи могут держать занятыми ресурсы всё отведённое им время.

Desktop Grid-системы: как и пакетные системы приложения desktop-grid, нацелены на утилизацию вычислительных ресурсов, однако они существенно отличаются от них. Пакетные системы задействуют оборудование организаций: суперкомпьютеры, кластеры и другое дорогостоящее оборудование. В отличие от пакетных систем, Desktop Grid задействуют обычные компьютеры [6], а связь между ними организуется через сеть Интернет, кроме того, компьютеры могут быть выключены или отключены от сети. Участники проектов Desktop Grid не являются экспертами и разработчиками ПО, они участвуют в вычислениях, поскольку им это интересно. Проекты не контролируют своих участников и не могут предотвратить злонамеренное поведение, например, предоставление ложных результатов.

Системы Desktop Grid могут иметь клиентское приложение, реализованное при помощи веб-апплета или отдельного настольного приложения. Если клиентское приложение реализовано в виде веб-апплета, то участникам проекта достаточно открыть веб-страницу приложения и запустить исполнение заданий. Апплет реализован на языке Java и может использовать ресурсы настольного компьютера с согласия пользователя. К таким приложениям относятся: Charlotte, Javelin и Bayanihan. В случае использования отдельного настольного приложения пользователю потребуется установить и запустить клиентское приложение, такой клиент требуется, например, для BOINC, XTremweb и Entropia [7].

Одной из систем Desktop Grid является BOINC (Berkley Open Infrastructure for Network Computing). Основные цели BOINC [8]:

1) снизить порог вхождения в область публичных вычислений;
2) разделить ресурсы между проектами;
3) поддержать различные применения Desktop Grid-вычислений;

4) мотивировать участников.

Плюсы использования BOINC:

1) возможность задействовать значительные вычислительные ресурсы длительное время;
2) бесплатный доступ к вычислениям;
3) хорошо подходит для постоянных исследований и переборных задач;
4) прост для участников.

Недостатки:

1) нестабильная вычислительная мощность сети компьютеров;
2) различные по оборудованию компьютеры;
3) ограниченный размер одного вычислительного задания;
4) необходимость запуска дубликатов заданий, поскольку они исполняются в недоверенном окружении;
5) труднопрогнозируемый срок исполнения заданий.

Системы виртуализации вычислительных ресурсов нацелены на использование виртуальных машин и контейнеров изоляции ресурсов в качестве единицы исполнения. Виртуализация — механизм абстракции физического оборудования и системных ресурсов операционной системы. В настоящее время технологии виртуализации тесно связаны с концепцией облачных вычислений. Облачные вычисления по своей природе требуют задействовать технологии, позволяющие отделить приложения от операционной системы и реального оборудования. Существуют два метода программной виртуализации:

1) динамическая трансляция;
2) паравиртуализация.

При динамической трансляции инструкции гостевой ОС транслируются гипервизором в безопасные инструкции, после чего управление возвращается гостевой ОС. В методе паравиртуализации требуется модифицировать исходный код ядра гостевой ОС, гипервизор предоставляет гостевой ОС специальный API для работы с системными ресурсами, такими как страницы памяти.

Помимо программной виртуализации многие гипервизоры поддерживают технологии аппаратной виртуализации, например, Intel VT-x и AMD Pacific. Аппаратная поддержка требует реализации дополнительного набора инструкций, упрощающих выполнение на аппаратном уровне операций для предоставления прямого доступа к ресурсам процессора из гостевых систем. К системам виртуализации относятся гипервизоры: Xen, VMware и KVM.

Приложения высокопроизводительных вычислений могут получить выигрыш от виртуализации несколькими путями [9]:

1) гипервизор может гарантировать фиксированное выделение ресурсов для гостевых ОС, таких как процессорное время и оперативная память;
2) узлы виртуального кластера могут быть запущены на различных ядрах одного или нескольких реальных узлов, причём конфигурацию узлов и их физическое расположение можно менять во время работы;
3) изоляция пользовательских процессов делает системы на базе виртуализации более устойчивыми к сбою пользовательского ПО;
4) гипервизор берёт на себя обеспечение безопасности процессов и данных.

Недостатки:

1) сложность ПО виртуализации;
2) виртуализация требует дополнительных ресурсов процессора и памяти на размещение гостевой ОС;
3) долгий (по сравнению с запуском приложения) холодный старт новых виртуальных машин.

Стоит отдельно выделить технику контейнеризации приложений как способ изоляции приложений и частичной виртуализации ресурсов. В отличие от гипервизоров контейнерная виртуализация не требует виртуализации гостевой ОС. Она задействует механизмы ядра операционной системы, которые позволяют изолировать процессы и выделить для них физические ресурсы системы [10]. Такой подход реализован в системах OpenVZ, LXC и Docker.

И хотя контейнерная виртуализация по сравнению с виртуальными машинами требует меньших ресурсов, она всё же имеет недостатки:

1) меньший уровень изоляции пользовательского ПО на уровне разделяемых ресурсов;
2) меньший уровень безопасности приложений и данных.

Платформы распределённых вычислений представлены как отдельный класс систем, поскольку задействуют другие подходы к организации распределённых вычислений. Например, используют другие системы для реального исполнения задач и, самое главное, предоставляют высокий уровень абстракции от физических ресурсов. Этому классу систем принадлежат следующие:

1) Globus Toolkit — позволяет запускать задачи с использованием планировщиков PBS, Condor, Platform LSF [11];
2) Hadoop — набор библиотек и служебных программ для организации вычислений в парадигме MapReduce [12];
3) CLAVIRE — многопрофильная инструментально-технологическая среда, позволяющая строить распределённые приложения из готовых вычислительных узлов, определяя потоки данных [13].

CLAVIRE реализует концепцию композитных приложений, в которой наибольшее значение имеют потоки данных. Для моделирования таких потоков используется графическая нотация процессов Workflow, позволяющая определить потоки данных между частями приложения и собрать из них приложение. Моделирование потоков данных предполагает, что все необходимые вычислительные алгоритмы уже разработаны, а в облаке есть готовые (запущенные или стартующие по требованию) узлы. Поэтому такой процесс работы над приложениями непригоден для регулярного использования и исполнения массовых расчётов. Композитные приложения лучше подходят для постоянно исполняемых вычислений, например, мониторинга состояния транспортных систем или прогнозирования наводнений [14].

К достоинствам платформ распределённых вычислений стоит отнести развитые средства разработки приложений, многообразие способов запуска и мониторинг исполнения приложений. Недостатки:

1) сложность отладки;
2) специализированный процесс разработки приложений, требующий задействования средств платформы (Hadoop, CLAVIRE);
3) сокрытие деталей взаимодействия приложений с оборудованием.

2. Инструменты разработки Templet. Веб-сервис автоматизации параллельных вычислений на суперкомпьютере «Сергей Королёв» Самарского го-

сударственного аэрокосмического университета Templet состоит из 3 компонентов:

1) библиотеки параллельного программирования Templet SDK;
2) веб-сервис для запуска и мониторинга задач на суперкомпьютере Templet Web;
3) подсистема мониторинга состояния кластера.

Templet SDK — комплект инструментов для разработки параллельных приложений. Включает в себя препроцессор исходных текстов и библиотеку времени исполнения. Препроцессор позволяет разработать и отладить последовательную программу на локальной машине, а затем запустить код на исполнение на суперкомпьютере в параллельном режиме (код будет получен автоматически из последовательной версии, для параллельной работы будет использоваться API POSIX Threads). Применение инструментов Templet SDK не требует от пользователя знаний методов параллельного программирования в UNIX или Windows и позволяет сосредоточиться на решении прикладной задачи.

Веб-приложение для управления проектами и задачами Templet Web, развёрнутое по адресу http://templet.ssau.ru/templet, позволяет произвести следующие действия:

- начать разработку проекта параллельного приложения при помощи шаблонов вычислительных методов пакета Templet SDK: портфель задач, конвейер и другие;
- организовать частное облако окружений для запуска проектов;
- организовать работу команды над проектом с применением VCS;
- получать доступ к общим окружениям и запущенным задачам;
- разворачивать приложение в тестовом и целевом окружениях;
- отслеживать работу приложения во время продолжительных вычислений;
- получать уведомления о статусе задач в целевых окружениях;
- управлять бинарными зависимостями шаблонов и проектов для различных архитектур и платформ.

Шаблоны проектов позволяют предоставлять готовые структуры проектов и стандартизировать пакеты поставки приложений в целевые окружения. Проекты строятся на основе шаблонов и могут хранить код в системах контроля версий (VCS), в проект подключаются общие окружения для запуска задач. Задачи и окружения проекта доступны всей команде проекта, что позволяет эффективно организовать коллективную работу. Результаты вычислений сохраняются в архиве, к ним можно получать доступ не только по завершении задач, но и позднее. Сервис ведёт учёт использования ресурсов суперкомпьютера и предоставляет пользователям статистику по запущенным задачам и загрузке ресурсов. Это помогает оценить максимальный объём ресурсов, который сможет использовать приложение, с учётом допустимого времени ожидания вычислительной задачи в очереди пакетной системы.

Templet Web относится к классу высокоуровневых систем и задействует в качестве исполняющего механизма планировщик Torque или отдельно запускаемый процесс. В системе поддерживаются окружения под управлением ОС Linux (в том числе виртуальные) и суперкомпьютер «Сергей Королёв». Система имеет расширяемую архитектуру, что позволяет добавлять поддер-

живаемые окружения по мере её развития. В Templet Web основной акцент сделан на простом процессе разработки приложений и интеграции средств разработки и платформы для исполнения. Сервис работает на двух языках — русском и английском, он доступен для использования не только студентам и преподавателям СГАУ, но и любому стороннему пользователю.

В основе механизма взаимодействия системы и целевых окружений лежит прямое подключение к окружению по безопасному протоколу SSH (Secure Shell). Большим преимуществом такого подхода является возможность использования неподготовленных окружений на базе ОС Linux, поскольку для работы веб-сервиса в этом случае не требуется установки дополнительного ПО и произведения настройки окружения для взаимодействия с системой. Недостатком такого решения является необходимость иметь прямое подключение от веб-сервиса к серверу окружения (или к узлу управления в случае суперкомпьютера). Зачастую высокопроизводительные кластеры недоступны для использования за пределами сети организации, выходом в таком случае может стать организация защищённой виртуальной сети (VPN). Стоит отметить, что установка дополнительного ПО (например, агента на стороне кластера) сопряжена с большими сложностями, ведь политика безопасности окружения может ограничивать возможность запуска сторонних приложений в режиме демона.

Подсистема мониторинга суперкомпьютера «Сергей Королёв» собирает детальную информацию по использованию суперкомпьютера:

– состояние очереди пакетной системы;
– запрашиваемые задачами ресурсы;
– состояние узлов суперкомпьютера.

Подсистема работает независимо от веб-сервиса, что позволяет производить обслуживание и обновление веб-сервиса Templet Web без остановки сбора информации о функционировании кластера. Информация, собираемая подсистемой мониторинга, используется для решения задач прогнозирования доступных вычислительных ресурсов кластера [15]:

– прогноз количества доступных вычислительных ресурсов;
– прогноз момента запуска вычислительной задачи в очереди пакетной системы.

В основе подсистемы мониторинга используются конвейеры из подпрограмм-агентов [16]. Каждый агент реализует небольшую хорошо инкапсулированную функциональность:

– получение данных о состоянии кластера;
– разбор данных и извлечение ключевых показателей;
– сканирование всех собранных данных;
– создание новой записи о состоянии в базе данных (БД);
– обновление записи о состоянии.

Такой набор микрофункциональных компонентов позволяет строить N конвейеров под несколько классов задач и исполнять масштабные операции в виде N параллельных конвейеров. Причём каждый агент исполняется конкурентно (может исполняться параллельно). Гибкая архитектура конвейеров на базе агентов используется при масштабных обновлениях БД мониторинга, например, при расчёте нового показателя для всех исторических данных. В штатном режиме работы мониторинга используется один конвейер команд:

1) получить слепок состояния кластера;
2) выделить показатели;
3) создать новую запись в БД.

Агенты реализованы на языке Scala [17], для организации взаимодействия агентов используется библиотека Akka Actors, обеспечивающая слой абстракции и позволяющая исполнять агенты конкурентно [18].

Данные подсистемы мониторинга доступны в виде графиков загрузки кластера в веб-сервисе Templet Web. Открытые данные мониторинга доступны на сайте проекта (`http://templet.ssau.ru/wiki/открытые_данные`).

3. Решение прикладных задач при помощи инструментов Templet. Сервис Templet [19] используется в СГАУ при работе над приложениями научных вычислений. Одно из приложений, в разработке которых использовался инструментарий Templet, — приложение расчёта параметров многомерных динамических систем. В самом приложении задействован Templet SDK [20], запуски выполнялись через сервис Templet Web.

Задача анализа многомерных динамических систем с помощью сечений Пуанкаре содержит независимые по данным этапы вычисления координат пересечения траекторий с плоскостью сечения [21]. Распараллеливание вычислений одной траектории является сложной задачей, поскольку инструкции расчёта одной траектории сильно связаны по данным. Для решения этой задачи использовался шаблон Templet SDK "Портфель задач". Распараллеливание приложения для этого шаблона состоит в реализации функции извлечения задачи из портфеля, её обработки и помещения результатов работы в портфель. Последовательность выполнения при параллельном исполнении сохраняется только для функции извлечения задач. Эксклюзивным доступом к данным портфеля обладают функции извлечения задачи и помещения результатов в портфель.

Исходная задача была декомпозирована и представлена в виде системного (Templet SDK и шаблон "Портфель задач") и прикладного уровней. Шаблон вычислительного метода реализуется системными программистами и может быть повторно использован во многих проектах. Прикладные программисты или специалисты в предметной области разрабатывают приложение для вычислений в терминах и идиомах шаблона вычислительного метода, что избавляет их от необходимости ручного распараллеливания кода.

Templet SDK, включающий набор шаблонов вычислительных методов, упростил разработку параллельного приложения, которое затем запускалось при помощи веб-сервиса Templet Web. Инструментарий позволяет запускать приложение в нескольких режимах: отладка, эмуляция, приложение Win32, POSIX-приложение, распределённое MPI-приложение. Режим эмуляции позволяет оценить максимальное ускорение (оптимистичную оценку) для большого числа процессоров без реального запуска на суперкомпьютере.

Заключение. Инструментарий имеет большое значение при разработке приложений научной направленности, поскольку крайне важно грамотно выбрать платформу для организации разработки, средства распараллеливания и организации высокопроизводительных вычислений. Набор инструментов Templet предоставляет не только сервис для запуска приложений, но и библиотеки шаблонов параллельных приложений, средства для разработки и отладки прикладных решений. Он позволяет прикладным программистам

сосредоточить усилия на работе вычислительных алгоритмов, снизить трудоёмкость разработки и сэкономить ресурсы на стадии тестирования и отладки.

Благодарности. Авторы выражают благодарность Самарскому государственному аэрокосмическому университету за поддержку этого исследования. Работа выполнена при частичной поддержке Российского фонда фундаментальных исследований (проект № 15–08–05934-а) и при государственной поддержке Министерства образования и науки РФ в рамках реализации мероприятий Программы повышения конкурентоспособности СГАУ среди ведущих мировых научно-образовательных центров на 2013–2020 годы.

ORCIDs
Юрий Сергеевич Артамонов: http://orcid.org/0000-0002-9301-7041
Сергей Владимирович Востокин: http://orcid.org/0000-0001-8106-6893

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Keyes D. E., et al. Multiphysics simulations: Challenges and opportunities // *International Journal of High Performance Computing Applications*, 2013. vol. 27, no. 1. pp. 4–83. doi: 10.1177/1094342012468181; Technical Report no. ANL/MCS-TM-321 Rev. 1.1, 2012; doi: 10.2172/1034263.
2. Sullivan B. Making Sense of Revision-Control Systems // *Commun. ACM*, 2009. vol. 52, no. 9. pp. 56–62. doi: 10.1145/1562164.1562183.
3. Neitsch A., Wong K., Godfrey M. W. Build System Issues in Multilanguage Software / *28th IEEE International Conference on Software Maintenance (ICSM)*. Riva del Garda, Trento, Italy, 2012. pp. 140–149. doi: 10.1109/ICSM.2012.6405265.
4. Ettl M., Neidhardt A., Brisken W., Dassing R. Continuous Software Integration and Quality Control during Software Development / *Seventh General Meeting (GM2012) of the international VLBI Service for Geodesy and Astrometry (IVS)* (Madrid, Spain, March 4–9, 2012); eds. D. Behrend, K. D. Baver. National Aeronautics and Space Administration, 2012. pp. 227–230.
5. Yan Y., Chapman B. *Comparative Study of Distributed Resource Management Systems – SGE, LSF, PBS Pro, and LoadLeveler*, 2008. 19 pp., http://www.dcc.fc.up.pt/~ines/aulas/1213/CG/papers/RMSComparison.pdf
6. Anderson D. P. Boinc: A system for public-resource computing and storage / *Fifth IEEE/ACM International Workshop on Grid Computing*, 2004. pp. 4–10. doi: 10.1109/grid.2004.14.
7. SungJin Choi, et al. Characterizing and Classifying Desktop Grid / *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*. Rio de Janeiro, Brazil, 2007. pp. 743–748. doi: 10.1109/CCGRID.2007.31.
8. Korpela E. J. SETI@home, BOINC, and Volunteer Distributed Computing // *Annual Review of Earth and Planetary Sciences*, 2012. vol. 40, no. 1. pp. 69–87. doi: 10.1146/annurev-earth-040809-152348.
9. Mergen M. F., Uhlig V., Krieger O., Xenidis J. Virtualization for high-performance computing // *SIGOPS Oper. Syst. Rev.*, 2006. vol. 40, no. 2. pp. 8–11. doi: 10.1145/1131322.1131328.
10. Xavier M. G., et al. Performance evaluation of container-based virtualization for high performance computing environments / *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2013. pp. 233–240. doi: 10.1109/pdp.2013.41.
11. Foster I. Globus toolkit version 4: Software for service-oriented systems // *J. Comput. Sci. Technol.*, 2006. vol. 21, no. 4. pp. 513–520. doi: 10.1007/s11390-006-0513-y ; *Network and Parallel Computing* / Lecture Notes in Computer Science, 3779. Berlin, Heidelberg: Springer, 2005. pp. 2–13. doi: 10.1007/11577188_2.
12. Taylor R. C. An overview of the Hadoop, MapReduce, HBase framework and its current applications in bioinformatics // *BMC Bioinformatics*, 2010. vol. 11 (Suppl 12). pp. S1. doi: 10.1186/1471-2105-11-s12-s1.

13. Knyazkov K. V., Kovalchuk S. V., Tchurov T. N., Maryin S. V., Boukhanovsky A. V. CLAVIRE: e-Science infrastructure for data-driven computing // *Journal of Computational Science*, 2012. vol. 3, no. 6. pp. 504–510. doi: 10.1016/j.jocs.2012.08.006.
14. Ivanov S. V., Kosukhin S. S., Kaluzhnaya A. V., Boukhanovsky A. V. Simulation-based collaborative decision support for surge floods prevention in St. Petersburg // *Journal of Computational Science*, 2012. vol. 3, no. 6. pp. 450–455. doi: 10.1016/j.jocs.2012.08.005.
15. Артамонов Ю. С. Основные подходы прогнозирования доступных вычислительных ресурсов в кластерных системах / *Перспективные информационные технологии (ПИТ 2014)*: Труды Международной научно-технической конференции; ред. С. А. Прохоров. Самара: Изд-во Самарского научного центра РАН, 2014. С. 305–310, http://templet.ssau.ru/wiki/lib/exe/fetch.php?media=pit2014:prediction.pdf.
16. Agha G. A., Kim W. Actors: A unifying model for parallel and distributed computing // *Journal of Systems Architecture*, 1999. vol. 45, no. 15. pp. 1263–1277. doi: 10.1016/S1383-7621(98)00067-8.
17. Haller P., Odersky M. Scala Actors: Unifying thread-based and event-based programming // *Theoretical Computer Science*, 2009. vol. 410, no. 2–3. pp. 202–220. doi: 10.1016/j.tcs.2008.09.019.
18. Shams M. I., Vivek S. Integrating task parallelism with actors / *Proceedings of the ACM international conference on Object oriented programming systems languages and applications (OOPSLA '12)*. New York, NY, USA, 2012. pp. 753–772. doi: 10.1145/2384616.2384671.
19. Артамонов Ю. С., Востокин С. В., Назаров Ю. П. Templet – Сервис непрерывной интеграции для разработки высокопроизводительных приложений / *Высокопроизводительные параллельные вычисления на кластерных системах*: Материалы XII Всероссийской конференции. Нижний Новгород: Изд-во НГУ, 2012. С. 82, http://www.hpcc.unn.ru/file.php?id=713.
20. Востокин С. В. Препроцессор языка Templet: инструмент программирования в терминах модели «процесс-сообщение» // *Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки*, 2014. №3(36). С. 169–182. doi: 10.14498/vsgtu1334.
21. Востокин С. В., Дорошин А. В., Артамонов Ю. С. *Программный комплекс Templet. Организация прикладных вычислений на базе суперкомпьютера «Сергей Королёв»*, 2014. 5 с., http://templet.ssau.ru/wiki/_media/pit2014/templetweb.pdf.

Поступила в редакцию 18/V/2015;
в окончательном варианте — 27/VI/2015;
принята в печать — 08/VIII/2015.

Vestn. Samar. Gos. Techn. Un-ta. Ser. Fiz.-mat. nauki
[J. Samara State Tech. Univ., Ser. Phys. & Math. Sci.], 2015, vol. 19, no. 4, pp. 785–798

ISSN: 2310-7081 (online), 1991-8615 (print) **doi:** http://dx.doi.org/10.14498/vsgtu1437

MSC: 68M14, 68M20

TOOLING SOFTWARE FOR DEVELOPMENT AND EXECUTION SUPPORT OF SCIENTIFIC COMPUTING APPLICATIONS IN CLUSTER SYSTEMS

Yu. S. Artamonov, S. V. Vostokin

Samara State Aerospace University,
34, Moskovskoye sh., Samara, 443086, Russian Federation.

Abstract

Rationale: Many different tools exist for development of scientific computing applications. Most of them are focused on the process of writing software code, but often there is a need for applications that organize the computation process and support team development. The article describes application development specifics in the field of science-oriented computing and highlights individual issues in the development of such software. **Classification of task management systems:** The systems are classified by means of computing process organization and the layer of hardware abstraction. **Templet development tools:** The tools for application development considered in the article include parallel programming libraries, a task running and monitoring service and the monitoring subsystem for SSAU cluster. Close interaction between these tools enables effective teamwork for scientific application development. **Applied problems solved by Templet tools:** Tooling is used to solve practical issues in the field of modeling multi-dimensional dynamic systems behavior. The article demonstrates an approach that splits application development into system-level and applied development layers. **Conclusion:** The article concludes about the use of design techniques and the benefits provided by software development tools.

Keywords: tooling, service, library, parallel programming, environment, monitoring, computing, cluster.

doi: http://dx.doi.org/10.14498/vsgtu1437

Acknowledgments. Our thanks go to Samara State Aerospace University (SSAU) for its continuous support to this research. This work was partially supported by Russian Foundation for Basic Research (project no. 15–08–05934-a) and by the Ministry of Education and Science of the Russian Federation in the framework of the implementation of the Program of increasing

Please cite this article in press as:
Artamonov Yu. S., Vostokin S. V. Tooling software for development and execution support of scientific computing applications in cluster systems, *Vestn. Samar. Gos. Tekhn. Univ., Ser. Fiz.-Mat. Nauki* [J. Samara State Tech. Univ., Ser. Phys. & Math. Sci.], 2015, vol. 19, no. 4, pp. 785–798. doi: 10.14498/vsgtu1437. (In Russian)

Authors Details:
Yuriy S. Artamonov (artamonov@about.me), Postgraduate Student, Dept. of Information Systems & Technology.
Sergey V. Vostokin (Dr. Techn. Sci.; easts@mail.ru; Corresponding Author), Professor, Dept. of Information Systems & Technology.

the competitiveness of SSAU among the world's leading scientific and educational centers over the period from 2013 till 2020.

ORCIDs
Yuriy S. Artamonov: http://orcid.org/0000-0002-9301-7041
Sergey V. Vostokin: http://orcid.org/0000-0001-8106-6893

REFERENCES

1. Keyes D. E., et al. Multiphysics simulations: Challenges and opportunities, *International Journal of High Performance Computing Applications*, 2013, vol. 27, no. 1, pp. 4–83. doi: 10.1177/1094342012468181; Technical Report no. ANL/MCS-TM-321 Rev. 1.1, 2012; doi: 10.2172/1034263.
2. Sullivan B. Making Sense of Revision-Control Systems, *Commun. ACM*, 2009, vol. 52, no. 9, pp. 56–62. doi: 10.1145/1562164.1562183.
3. Neitsch A., Wong K., Godfrey M. W. Build System Issues in Multilanguage Software, *28th IEEE International Conference on Software Maintenance (ICSM)*. Riva del Garda, Trento, Italy, 2012, pp. 140–149. doi: 10.1109/ICSM.2012.6405265.
4. Ettl M., Neidhardt A., Brisken W., Dassing R. Continuous Software Integration and Quality Control during Software Development, *Seventh General Meeting (GM2012) of the international VLBI Service for Geodesy and Astrometry (IVS)* (Madrid, Spain, March 4–9, 2012); eds. D. Behrend, K. D. Baver. National Aeronautics and Space Administration, 2012, pp. 227–230.
5. Yan Y., Chapman B. *Comparative Study of Distributed Resource Management Systems – SGE, LSF, PBS Pro, and LoadLeveler*, 2008, 19 pp., http://www.dcc.fc.up.pt/~ines/aulas/1213/CG/papers/RMSComparison.pdf
6. Anderson D. P. Boinc: A system for public-resource computing and storage, *Fifth IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 4–10. doi: 10.1109/grid.2004.14.
7. SungJin Choi, et al. Characterizing and Classifying Desktop Grid, *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*. Rio de Janeiro, Brazil, 2007, pp. 743–748. doi: 10.1109/CCGRID.2007.31.
8. Korpela E. J. SETI@home, BOINC, and Volunteer Distributed Computing, *Annual Review of Earth and Planetary Sciences*, 2012, vol. 40, no. 1, pp. 69–87. doi: 10.1146/annurev-earth-040809-152348.
9. Mergen M. F., Uhlig V., Krieger O., Xenidis J. Virtualization for high-performance computing, *SIGOPS Oper. Syst. Rev.*, 2006, vol. 40, no. 2, pp. 8–11. doi: 10.1145/1131322.1131328.
10. Xavier M. G., et al. Performance evaluation of container-based virtualization for high performance computing environments, *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2013, pp. 233–240. doi: 10.1109/pdp.2013.41.
11. Foster I. Globus toolkit version 4: Software for service-oriented systems, *J. Comput. Sci. Technol.*, 2006, vol. 21, no. 4, pp. 513–520. doi: 10.1007/s11390-006-0513-y ; *Network and Parallel Computing*, Lecture Notes in Computer Science, 3779. Berlin, Heidelberg, Springer, 2005, pp. 2–13. doi: 10.1007/11577188_2.
12. Taylor R. C. An overview of the Hadoop, MapReduce, HBase framework and its current applications in bioinformatics, *BMC Bioinformatics*, 2010, vol. 11 (Suppl 12), pp. S1. doi: 10.1186/1471-2105-11-s12-s1.
13. Knyazkov K. V., Kovalchuk S. V., Tchurov T. N., Maryin S. V., Boukhanovsky A. V. CLAVIRE: e-Science infrastructure for data-driven computing, *Journal of Computational Science*, 2012, vol. 3, no. 6, pp. 504–510. doi: 10.1016/j.jocs.2012.08.006.
14. Ivanov S. V., Kosukhin S. S., Kaluzhnaya A. V., Boukhanovsky A. V. Simulation-based collaborative decision support for surge floods prevention in St. Petersburg, *Journal of Computational Science*, 2012, vol. 3, no. 6, pp. 450–455. doi: 10.1016/j.jocs.2012.08.005.

15. Vostokin S. V. The basic syntax of TEMPLET markup language for representing process-per-message model, *Perspektivnye informatsionnye tekhnologii (PIT 2014)* [Advanced information technologies (PIT 2014)], Proceedings of the Technical Conference; ed. S. A. Prokhorov. Samara, Samara Research Center, 2014, pp. 305–310 (In Russian), http://templet.ssau.ru/wiki/lib/exe/fetch.php?media=pit2014:prediction.pdf.
16. Agha G. A., Kim W. Actors: A unifying model for parallel and distributed computing, *Journal of Systems Architecture*, 1999, vol. 45, no. 15, pp. 1263–1277. doi: 10.1016/S1383-7621(98)00067-8.
17. Haller P., Odersky M. Scala Actors: Unifying thread-based and event-based programming, *Theoretical Computer Science*, 2009, vol. 410, no. 2–3, pp. 202–220. doi: 10.1016/j.tcs.2008.09.019.
18. Shams M. I., Vivek S. Integrating task parallelism with actors, *Proceedings of the ACM international conference on Object oriented programming systems languages and applications (OOPSLA '12)*. New York, NY, USA, 2012, pp. 753–772. doi: 10.1145/2384616.2384671.
19. Artamonov Yu. S., Vostokin S. V., Nazarov Yu. P. Templet – Continuous integration service for the development of high performance applications, *Vysokoproizvoditel'nye parallel'nye vychisleniia na klasternykh sistemakh* [High-performance parallel computing on cluster systems], Proceedings of XII All-Russian Conference. Nizhny Novgorod, NSU Publ., 2012, pp. 82 (In Russian), http://www.hpcc.unn.ru/file.php?id=713.
20. Vostokin S. V. The Templet Language Preprocessor: A Programming Tool for Process-per-Message Modeling, *Vestn. Samar. Gos. Tekhn. Univ. Ser. Fiz.-Mat. Nauki*, 2014, no. 3(36), pp. 169–182 (In Russian). doi: 10.14498/vsgtu1334.
21. Vostokin S. V., Doroshin A. V., Artamonov Yu. S. *The software package Templet. Organizing applications into the supercomputer "Sergei Korolev"*, 2014, 5 pp. (In Russian), http://templet.ssau.ru/wiki/_media/pit2014/templetweb.pdf.

Received 18/V/2015;
received in revised form 27/VI/2015;
accepted 08/VIII/2015.