УДК 004.272.2+004.75+544.18

ДИНАМИЧЕСКИ ФОРМИРУЕМЫЕ ПАРАЛЛЕЛЬНЫЕ СРЕДЫ В УСЛОВИЯХ ГРИД-ПОЛИГОНОВ: ПРОБЛЕМЫ И РЕШЕНИЯ

В. М. Волохов[1], Д. А. Варламов[1,2], А. В. Пивушков[1], Н. Ф. Сурков[1], А. В. Волохов[1]

Статья посвящена анализу проблем выполнения сложно сконфигурированных параллельных прикладных пакетов в распределенных средах и описанию некоторых способов разрешения подобных проблем. Описаны методы создания и запуска в различных грид-средах (gLite, Unicore, Globus GT4) динамически формируемых ("виртуальных") параллельных сред исполнения для обеспечения запуска параллельных прикладных задач. Разработанные методы позволяют запускать сложные, требующие предустановки и настройки кластера, параллельные приложения на не сконфигурированных заранее произвольных ресурсах грид-полигонов. Приведены примеры реализации данного метода для квантово-химического пакета GAMESS-US на различных грид-полигонах в условиях всех трех вышеупомянутых сред. Работа выполнена при поддержке РФФИ, грант № 11–07–00686-а. Статья рекомендована к публикации Программным комитетом Международной научной конференции "Параллельные вычислительные технологии" (ПАВТ–2011; http://agora.guru.ru/pavt2011).

Ключевые слова: распределенные вычисления, грид, виртуализация, MPI, вычислительная и квантовая химия, GAMESS-US.

1. Введение. На основе долговременного опыта работы в различных распределенных средах (gLite, Unicore, Globus GT4 [1]) в условиях основных российских грид-полигонов (EGEE(EGI)-RDIG, Национальная нанотехнологическая сеть, СКИФ-Полигон) авторами ранее был сделан вывод [2–4], что значительными препятствиями (среди многих прочих) на пути применения грид-технологий в вычислительной химии являются следующие проблемы:

— грид-полигоны могут содержать в своем составе разнородные ресурсные узлы (обусловливается разнообразием операционных систем, применением различных типов сетевых сервисов, способами реализации параллельных сред, системами управления задачами, политикой безопасности и др.);

— для многих ресурсоемких параллельных приложений необходимо создавать вычислительную среду, состоящую из "приватных" конфигурационных настроек, дополнительных служб, специфичных параллельных сред, специализированных мест хранения данных и прочих компонентов, что плохо коррелирует с особенностями настройки кластеров в целом;

— невозможно (или избыточно трудоемко) перенастраивать работающие вычислительные ресурсы (особенно суперкомпьютеры или кластеры класса "production farms") для целей распределенных вычислений под нужды конкретных прикладных пакетов.

Разумеется, эти проблемы распространяются на запуски в грид-средах любых сложно сконфигурированных прикладных пакетов. Одной из стратегий решения этих проблем (или хотя бы части из них) может стать применение различных технологий виртуализации вычислительных ресурсов и приложений.

В 2009–2010 гг. в рамках Программы фундаментальных исследований Президиума РАН № 13 на 2009–2011 гг. "Проблемы создания национальной научной распределенной информационно-вычислительной среды на основе развития грид-технологий и современных телекоммуникационных сетей" авторами были продолжены исследования по применимости различных методов виртуализации в условиях распределенных сред. Среди прочих задач проекта была поставлена цель: изучить и применить на реальных расчетах в области химии два класса технологий виртуализации расчетов:

— адаптация ряда квантово-химических пакетов к работе в условиях разных распределенных сред

[1] Институт проблем химической физики РАН, пр. акад. Семенова, д. 1, 142432, г. Черноголовка; В. М. Волохов, зав. отд., e-mail: vvm@pro.icp.ac.ru; Д. А. Варламов, ст. науч. сотр., e-mail: dima@icp.ac.ru; А. В. Пивушков, ст. науч. сотр., e-mail: pav@icp.ac.ru; Н. Ф. Сурков, ст. науч. сотр., e-mail: surkov@icp.ac.ru; А. В. Волохов, ст. инж., e-mail: vav@.icp.ac.ru

[2] Институт экспериментальной минералогии РАН, ул. Институтская, д. 4, 142432, г. Черноголовка; Д. А. Варламов, ст. науч. сотр., e-mail: dima@iem.ac.ru

(gLite, Unicore, Globus Toolkit) с использованием динамически формируемых (виртуальных) переносимых параллельных сред;

— создание и применение виртуальных машин на существующих ресурсных узлах различных распределенных сред с целью расширения их функциональности (решение различных прикладных задач на базе разных программных архитектур, разделение ресурсов, повышение безопасности, вычислительные эксперименты) и отработки устойчивости узлов.

Термин "виртуализация" был использован авторами в двух основных смыслах: виртуализация вычислительных грид-ресурсов и сервисов с использованием виртуальных машин (ВМ) и виртуализация вычислительного объекта (приложения), перемещаемого в грид-среде. Потребность в виртуализации приложений для распределенных вычислительных сред продиктована необходимостью создания и поддержки стандартных механизмов взаимодействия между приложениями и вычислительными ресурсами (сервисами), одинаковых со стороны ресурса (поставщика сервисов) и со стороны приложения (вернее, созданного для него интерфейса) вне зависимости от настроек конкретного вычислительного ресурса.

Основные результаты в области виртуализации ресурсов были описаны авторами в других публикациях [4–6], в данной же статье более детально рассмотрен один из разработанных авторами методов виртуализации вычислительного объекта (параллельного приложения), перемещаемого и выполняемого в грид-среде. Метод основан на реализации виртуальной, динамически формируемой параллельной MPI-среды, передаваемой на ресурсные узлы грид-полигонов в форме "виртуального контейнера". Метод включает в себя предварительную адаптацию (и иногда рекомпиляцию) программы (прикладного пакета) для работы в роли приложения в составе "контейнера". Далее следует создание прототипа виртуальной параллельной среды исполнения, формирование собственно "контейнера" и процесс выполнения его как исходящего грид-задания на неподготовленном удаленном ресурсном грид-узле (т.е. без предустановки и настройки собственно прикладного пакета). Для чего это необходимо? Конечный пользователь грид-среды перестает быть зависимым от наличия на ресурсных узлах предустановленных прикладных пакетов (что во многом зависит от администраторов кластеров), используемых параллельных сред, особенностей настроек кластеров. В идеале возможен запуск большинства сложно сконфигурированных прикладных пакетов без привязки к конкретным ресурсным узлам, что всегда являлось ахиллесовой пятой грид-сред.

2. Анализ механизма выполнения стандартных грид-заданий в различных распределенных средах (для параллельных приложений). Сегодня для эффективной и комфортной работы прикладных пакетов даже в условиях локального кластера требуется создание разветвленной информационно-вычислительной инфраструктуры, включающей в себя вспомогательные приложения, службы, сетевые сервисы, хранилища данных и прочие компоненты, которые зачастую плохо совместимы с режимами работы ресурсного узла в целом. Для ряда пакетов прикладного ПО и сервисов следует создавать комплексные среды с необходимым набором приложений и элементов политики безопасности. Ключевыми требованиями являются скорость и простота предоставления таких сред, их тщательная изоляция друг от друга, квотирование вычислительных ресурсов для каждой среды, независимость от базовых настроек узла. Зачастую все это необходимо обеспечить без прерывания работы узлов и остановки вычислительной среды, особенно в рамках суперкомпьютеров и кластеров класса "production farms", т.е. ресурсных узлов, не допускающих остановок и переконфигурирования системы.

Другой фундаментальной проблемой решения задач (особенно параллельных) в условиях распределенных вычислений является необходимость виртуализации программных сред для исходящих задач. Например, для проведения параллельных вычислений требуется наличие установленной на ресурсных узлах какой-либо системы параллельного программирования (MPI, OpenMP и др.) или предустановленных специфичных математических библиотек. Широко используемые в настоящее время пакеты прикладных программ (ППП) вычислительной химии (GAMESS, Gaussian, NAMD и др.), как, впрочем, и большинство инженерных пакетов (ANSYS, Abacus, FlowVision и т.п.), отличаются сложностью конфигураций и повышенными требованиями к среде выполнения, особенно для проведения параллельных расчетов. Они требуют обязательной настройки большого количества переменных окружения операционной системы до запуска параллельного приложения на каждом из использующихся процессоров. Такая настройка, как правило, осуществляется в два этапа:

— при ручной (или полуавтоматической) установке ПО системным администратором на каждом узле ресурсного сайта на уровне операционной системы (например, при формировании сайтов виртуальной организации, требующей единых настроек ПО);

— при настройке соответствующих скриптов запуска задания для каждого пользователя, согласно требованиям как приложения, так и системы параллельного программирования;

— при установке пакетов из центрального репозитория в рамках виртуальной организации.

При этом традиционный подход со статическим линкованием необходимых библиотек (не говоря уже о динамическом варианте) к исполняемому модулю (пакету) часто не способен создать *полностью работоспособное* параллельное задание на произвольном ресурсе грид-среды, поскольку на подобном ресурсе попросту могут отсутствовать необходимые системные файлы или не поддерживаться необходимая приложению параллельная среда.

Для решения данной проблемы авторами был проведен анализ процедуры исполнения типичного параллельного задания на ресурсном грид-узле (для сред gLite, Unicore, Globus GT4). Он позволил определить требования к создаваемому виртуальному образу среды исполнения, а также принципиальную возможность формирования динамической среды исполнения для тестируемых ресурсов. Кроме того, был сделан анализ систем параллельного программирования для выбора оптимального виртуального образа среды исполнения параллельного приложения на грид-ресурсах. В качестве базового пакета для разработки виртуального образа среды исполнения параллельного приложения был выбрана среда Mpich-2. Детальнее этот анализ и последующая работа с библиотеками MPI-2 описаны в [2, 4]. Отметим, что на данном этапе работ были введены некоторые ограничения для используемых программных сред:

— использованы только аппаратные архитектуры x86 и em64t;

— на расчетных узлах грид-ресурсов предполагается использование операционной системы Linux (клоны на базе RedHat — собственно RedHat, ScientificLinux, Fedora и т.п.), что связано с особенностями размещения системного программного обеспечения, хотя принципиальных ограничений на использование других ветвей Linux-дистрибутивов нет;

— по стандарту настройки ресурсных грид-узлов для коммуникации между расчетными узлами используется интерфейс TCP/IP и беспарольный доступ по ssh (включая копирование файлов), возможна поддержка NFS-ресурсов;

— некоторые версии прикладных пакетов с целью повышения производительности вычислений имеют привязку к сетевым продуктам конкретных производителей и используют поставляемые этими производителями низкоуровневые драйверы; нами пока такие версии, несмотря на их высокую эффективность, использоваться не будут.

После анализа процедуры выполнения грид-задания в разных распределенных средах и выбора среды параллельных вычислений была разработана технология создания динамически формируемых образов исполняемых сред, или виртуальных “контейнеров”. Был сформирован перемещаемый программный пакет MPI-2. Полученный пакет далее использовался в качестве базового прототипа для разработки виртуального образа среды исполнения конкретных параллельных приложений.

В качестве тестового приложения была использована программа вычисления числа π (‘cpi.c’) из пакета Mpich-2, правильность работы которой легко проверяется в параллельной среде с различным количеством узлов. Исходная тестовая программа была доработана с учетом особенностей запуска прикладных приложений на грид-узлах, был получен ее исполняемый модуль и скрипты запуска с использованием библиотек MPI-2. Тестовый модуль и перемещаемый пакет MPI-2 были собраны и упакованы в единый “контейнер”, для запуска которого в тестируемых грид-средах была разработана серия низкоуровневых скриптов пользовательского интерфейса.

Была принята следующая схема запуска: на удаленный ресурсный грид-узел сети через брокер ресурсов (или непосредственно — это возможно в Globus) передается главный скрипт и упакованный “контейнер”, содержащий исполняемые файлы, необходимые системные библиотеки, файлы конфигурации и данных. Далее главный скрипт выполняет (*упрощенно*) следующую последовательность шагов: сбор информации о текущем грид-узле, распаковка “контейнера” в директории псевдопользователя грид-среды и перемещение файлов в общедоступную область, настройка среды, запуск сервера mpd (с правами mapped-user) на стартовом узле и проведение его тестирования, распределение необходимых файлов по списку свободных узлов, запуск “кольца” серверов mpd, запуск параллельного приложения и его работа как обычного распределенного задания с последующей передачей результатов на брокер ресурсов и пользовательский интерфейс, удаление всех библиотек и временных файлов со всех узлов. Более детально последовательность работы “контейнера” описана в [2–4].

Тестовый вариант “контейнера” (на примере нескольких простых параллельных задач) был отлажен на ресурсном грид-центре ИПХФ (его сайты использовались в качестве удаленного ресурса) для сред gLite, Unicore, Globus. Дальнейшее тестирование было проведено на ресурсных узлах RDIG в рамках BO RGSTEST (узлы НИИЯФ МГУ), а также на узлах СКИФ-Полигона и ГридННС, что показало применимость данного метода для большинства ресурсных сайтов данных сетей. В 2010 г. эта библиотека была адаптирована для работы с 64-битными вычислительными архитектурами, а также 64-битными версиями прикладных пакетов (для части которых разработчиками была существенно изменена схема параллели-

зации расчетов).

Разработка системы динамического компилирования приложения на ресурсном узле и инсталляции дополнительных библиотек на данном этапе работ не рассматривалась.

3. Реализация динамически формируемой параллельной среды исполнения для запуска квантово-химического пакета GAMESS-US. Для проведения технологических испытаний разработанного метода на примере прикладного пакета был выбран квантово-химический пакет GAMESS-US.

GAMESS-US (http://www.msg.ameslab.gov/GAMESS) — одна из наиболее популярных программ для теоретического исследования свойств химических систем, уступает по известности лишь программному комплексу Gaussian, позволяет рассчитывать энергию, структуры молекул, частоты их колебаний, а также разнообразные свойства молекул в газовой фазе и в растворе как в основном, так и в возбужденных состояниях. Основное направление — развитие методов расчета сверхбольших молекулярных систем. Основные программные модули GAMESS-US поддерживают параллельный режим вычислений как на многопроцессорных компьютерах, так и на кластерах. Пакет отличается сложностью установки и конфигурации, а также требует нестандартных настроек параллельной среды вычислений.

Работы по распараллеливанию GAMESS-US начались еще в 1991 г. Однако использование методов передачи сообщений MPI получило применение только с 1999 г., когда в пакете GAMESS-US была реализована модель интерфейса с распределенным размещением данных (DDI — Data Distributed Interface). Последняя версия интерфейса DDI, которая была оптимизирована для многопроцессорных SMP-архитектур общего вида, особенно работающих с памятью в стиле System V, была выпущена только в мае 2004 г. В настоящее время практически все *ab initio*[3] методы, включенные в пакет GAMESS, могут использовать параллельные вычисления.

Интерфейс DDI использует в качестве базовой “сокетную” TCP/IP модель межпроцессорных коммуникаций. Использование такого метода распараллеливания для работы на локальном кластере достаточно эффективно и довольно просто в конфигурации, но при работе в грид-средах возникает ряд принципиальных проблем:

а) необходимо заранее явно указывать используемые расчетные узлы (что обычно нереально);

б) неправильно оценивается загруженность расчетных узлов (учитывается только первый расчетный узел);

в) отсутствует возможность контроля выполнения удаленной задачи средствами middleware распределенной среды;

г) на ряде современных кластеров (например, “Чебышёв” в НИВЦ МГУ) сокетная модель неработоспособна из-за политики безопасности кластера.

Конфигурации же GAMESS-US с использованием библиотеки MPI авторами пакета разработаны только для ряда мейнфреймов известных производителей (Cray, IBM, SGI). До последнего времени конфигурации с MPI не рекомендовались, и желающим предлагалось экспериментировать с такими конфигурациями самостоятельно.

Для работы с пакетом GAMESS-US в грид-среде на ресурсных узлах ИПХФ РАН первоначально была установлена последняя наиболее широко распространенная версия Mpich-1.2.7 (см. выше), которая является реализацией стандарта MPI-1. Достоинством данной версии является то, что она явно включает в себя интерфейс Globus-2, основанный на Globus Runtime System, что было бы эффективно для запуска Globus-заданий. Однако получить работоспособную конфигурацию пакета GAMESS-US для MPI-1 не удалось по двум основным причинам.

1) Из-за особенностей запуска исполняемого задания GAMESS-US, который осуществляется скриптом, активизирующим более 150 переменных окружения. На главном узле среда создается правильно, однако механизм передачи переменных окружения на подчиненные узлы в библиотеке Mpich-1 стандартно отсутствует. В пакет Mpich был включен безопасный сервер (“secure server”), одной из задач которого являлась ликвидация этого недостатка, но из-за неполной совместимости с операционными системами ряда RedHat (типа ScientificLinux) эту функцию безопасного сервера использовать не удалось. При запуске задания на локальном узле пакет Mpich-1 использует такие команды оболочки, как “.” , “eval”, “exec”, которые не наследуют среду окружения запускающего процесса, что ведет к краху дочерних процессов;

2) Особенности реализации команды запуска параллельных заданий mpirun пакета Mpich-1. Запуски заданий на главном и подчиненных узлах существенно различаются — строки команды удаленного запуска (rsh или ssh) на подчиненных узлах дополняются служебными переменными. В результате стандартное расположение передаваемых в командной строке аргументов задания GAMESS-US нарушается и

[3]Ab initio (лат. от начала): в физике — решение задачи из первых основополагающих принципов без привлечения дополнительных эмпирических предположений. Обычно подразумевается прямое решение уравнений квантовой механики.

не распознается, что ведет к краху запуска.

В ИПХФ РАН с целью расширения функциональности применения пакета GAMESS-US в грид-сетях была поставлена задача разработки оригинальной конфигурации и сборки из исходных текстов исполняемого файла пакета GAMESS-US с использованием библиотеки MPI-2.

После установки библиотек MPI стандарта 2.0 (версия 1.0.3 пакета Mpich-2) была проведена соответствующая модификация конфигурационных скриптов пакета GAMESS-US (compddi, comp, compall, lked), а также программных модулей ddi_init.c и ddi_base.h. Был полностью переписан соответствующий раздел в запускающем скрипте rungms, который сначала запускает кольцо серверов mpd, а затем уже и само задание. После сборки исполняемого файла было проведено его тестирование на включенных в пакет GAMESS-US примерах файлов данных и получено совпадение результатов. Запуск параллельных заданий осуществляется командой mpiexec, которая не имеет указанных выше недостатков команды mpirun (библиотеки MPI-1).

Использование модифицированной авторами библиотеки MPI позволило впервые из отредактированных исходных текстов свободно распространяемого квантово-химического пакета GAMESS-US получить исполняемое задание для работы в параллельной среде под управлением MPI. Выбранный авторами подход по созданию виртуального образа среды исполнения на основе перемещаемого пакета MPI-2 показал свою продуктивность и в этом случае. Была проведена компиляция модифицированных исходных кодов GAMESS-US с использованием библиотек MPI-2 и получен бинарный пакет. Затем была создана система компоновки необходимых системных файлов (библиотеки, исполняемые системные файлы), собственно модифицированного GAMESS-US, сопутствующих конфигурационных файлов и настроечных скриптов, файлов данных в единый "контейнер", выступающий в роли исходящего задания распределенной среды. Запуск подобного контейнера аналогичен описанному выше для прототипа.

Окончательный размер "виртуального контейнера" для GAMESS-US (64-битной MPI-версии) составляет около 10 мегабайт, что весьма удовлетворительно для запуска пакета в распределенных средах, даже для относительно низкоскоростных сетей.

Серия первичных запусков (с использованием внутренних тестовых примеров собственно пакета GAMESS-US) вплоть до получения положительного результата тестов (равнозначность поведения сокетных и MPI-вариантов) была проведена на ресурсном грид-сайте ИПХФ РАН (grid-ce.icp.ac.ru) для сред gLite, Unicore, Globus (узлы использовались как удаленные ресурсы полигонов EGI-RDIG, Скиф-Полигона, ГридННС), т.е. запуск задач шел через грид-инфраструктуру). Дальнейшее успешное тестирование было проведено на удаленных ресурсных узлах RDIG в рамках BO RGSTEST (узлы НИИЯФ МГУ, lcg38.sinp.msu.ru, среда gLite). Были проведены успешные запуски пакета GAMESS-US с применением данной технологии (рассчитаны тестовые примеры молекулярных структур из дистрибутива GAMESS, например серия *ab initio* расчетов по оптимизации геометрии в 15-атомной системе ($P_3O_9H_3$) на уровне HF/6-31G*), подтвердившие полную работоспособность разработанной технологии.

Технология запуска пакета GAMESS-US с использованием динамически формируемых параллельных сред исполнения интегрирована в высокоуровневые web-интерфейсы работы с пакетом GAMESS-US для трех вышеописанных грид-полигонов, входящих в состав грид-портала ИПХФ РАН (http://grid.icp.ac.ru), что позволяет использовать данную технологию даже неискушенному пользователю. Механизм работы "контейнера" идентичен таковому для низкоуровневых клиентских интерфейсов. Были проведены тестовые запуски GAMESS через web-интерфейс на ресурсные сайты ИПХФ с решением задач из набора приложений к пакету.

4. Основные проблемы применения метода "виртуального контейнера" в распределенных средах. Было обнаружено, что на ряде кластеров (например, в Курчатовском РНЦ), предоставленных в качестве грид-ресурсов, запрещено или сильно ограничено использование скриптовых языков, что, естественно, противодействует запуску пришедшего на ресурс "виртуального контейнера". Для разрешения данной проблемы нами были проведены работы по переводу всех действий по развертыванию и настройке "контейнеров" в полностью бинарные исполняемые программы, которые действуют аналогично, но при этом не требуют доступа к shell языкам. Таким образом, впервые разработана технология запуска GAMESS-US в грид-средах в виде единого откомпилированного бинарного файла. При этом входящая задача порождает единственный процесс, который распаковывает библиотеки и бинарные системные файлы, собственно прикладной пакет, файлы данных, настраивает среду исполнения, в том числе параллельную среду Mpich-2, запускает параллельные процессы GAMESS-US, собирает полученные результаты, удаляет "мусор" и отправляет выходные данные на пользовательский интерфейс грид-среды.

На части доступных нам кластеров выявлено также, что в качестве политики безопасности запрещена передача файлов между расчетными узлами по беспарольному ssh, что в какой-то мере может быть решено

использованием общедоступных NFS-ресурсов, но в таком случае “контейнер” должен конфигурироваться для каждого приложения по-своему.

Не совсем корректно проводится мониторинг выполнения грид-задания на удаленном узле, что осложняет контроль пользователя.

5. Выводы и перспективы. В результате применения описанной технологии пользователь получает единое виртуальное приложение, которое в виде “виртуального контейнера” доставляется на ресурсный узел вместе со всеми конфигурационными настройками, относящимися к операционной системе, и поддержкой необходимых параллельных протоколов, не требуя процедуры предварительной установки и настройки. Далее “виртуальный контейнер” самостоятельно разворачивается (с использованием серии скриптов или в форме единичного бинарного процесса) на всех доступных пользователю полигона узлах грид-ресурса, подготавливая среду для исполнения параллельного приложения с последующим его запуском. При этом отсутствуют конфликты приложения с другими, уже установленными на узле программами и даже с другими экземплярами этого же приложения. Суть виртуализации приложения заключается в создании персональной копии необходимой части системных файлов и настроек операционной системы и доставке приложения совместно с этой информацией с последующим запуском в изолированном “контейнере”. Проведенные авторами эксперименты в этой области показали, что так могут быть решены проблемы установки, настройки, несовместимости с операционной системой и другими программами, разрешаются конфликты одинаковых приложений. Заметим, что данная технология применима для запуска подобных приложений и в условиях локальных кластеров без необходимости настройки расчетных узлов.

В более далекой перспективе одним из вариантов данной технологии предполагается использование (по аналогии с описанным “контейнером”) виртуальных машин (ВМ) как исходящих распределенных заданий, что позволит гарантировать пользователю необходимое качество обслуживания, не затрагивающее при этом работу основных служб ресурсных узлов. Таким образом, пользователю распределенной среды может быть предоставлена полностью изолированная виртуальная вычислительная среда, по своим свойствам не уступающая физическому серверу, в которой может быть предоставлен любой его собственный вычислительный сервис.

Приложения, реализованные в ВМ, в этом случае абсолютно не зависят от операционной системы и окружения, в котором ВМ выполняется. Пользователь получает возможность создать образ виртуальной машины с предустановленной операционной системой и полностью сконфигурированными приложениями, нацеленной на решение конкретной задачи. Этот образ затем передается на распределенный ресурс и исполняется там как грид-приложение, не требуя настройки данного узла под конкретные задачи. Это существенно облегчает адаптацию прикладного ПО для работы в распределенных средах. Дополнительным плюсом служит то, что эти технологии в принципе позволяют запускать образы виртуальных машин с операционными системами, отличными от установленных на ресурсах (например, Windows ВМ на Linux-кластере). Следует отметить потенциальные недостатки данного метода: это — размеры передаваемых заданий (могут достигать первых гигабайтов), “накладные” расходы на виртуализацию (до 15–20% от мощности ресурса, при оптимальной конфигурации они могут быть снижены до уровня 5–7%), необходимость установки и настройки гипервизоров ВМ на расчетных узлах.

6. Заключение. Разработан метод создания динамически формируемых параллельных вычислительных сред в виде перемещаемых “виртуальных контейнеров”, которые содержат: “персональные” копии необходимых системных файлов и библиотек, скрипты по настройке операционной системы, необходимые файловые “деревья”, собственно приложение, файлы данных и т.п. После динамического создания “контейнера” на пользовательском интерфейсе он средствами распределенной среды через брокер ресурсов доставляется на удаленный ресурсный узел как обычное грид-задание, “разворачивается” там, настраивает среду узла “под себя” и запускается как обычное параллельное приложение. По окончании работы приложения происходит “очистка” среды выполнения, возврат результатов на пользовательский интерфейс и приведение расчетных узлов в исходное состояние. Созданы два варианта “контейнеров”: с использованием скриптовых языков и как единого бинарного задания. В настоящее время этот метод применим для исполнения на ресурсных узлах, поддерживающих ОС системы Linux, т.е. типичных кластерах, интегрированных в грид-среды.

В качестве примера использован классический квантово-химический пакет GAMESS-US, для которого создан работоспособный “виртуальный контейнер” для сред gLite, Unicore, Globus GT4. Метод создания “контейнеров” интегрирован в высокоуровневые web-интерфейсы пакета GAMESS-US в составе грид-портала ИПХФ и реализован для трех российских грид-полигонов. Нет принципиальных ограничений для создания подобных “контейнеров” для других прикладных пакетов, требующих специфических параметров окружения и нестандартных настроек параллельных сред выполнения. Большинство про-

блем применения подобных “контейнеров” связано с политикой безопасности на ресурсных узлах. Эти проблемы могут быть решены в рамках виртуальных организаций (ВО) различных грид-полигонов.

Применение данного метода виртуализации приложений позволяет существенно расширить круг доступных грид-ресурсов для выполнения на них сложно сконфигурированных прикладных пакетов.

СПИСОК ЛИТЕРАТУРЫ

1. *Волохов В.М., Варламов Д.А., Пивушков А.В., Покатович Г.А., Сурков Н.Ф.* Технологии ГРИД в вычислительной химии // Вычислительные методы и программирование. 2010. **11**, № 1. 175–182.
2. *Волохов В.М., Варламов Д.А., Сурков Н.Ф., Пивушков А.В.* Виртуальные вычислительные среды: использование на GRID полигонах // Вестник ЮУрГУ. Сер. Математическое моделирование и программирование. 2009. № 17 (150). Вып. 3. 24–35.
3. *Варламов Д.А., Волохов В.М., Пивушков А.В., Сурков Н.Ф.* Виртуализация параллельных приложений квантовой химии для запуска на ресурсных узлах распределенных сред // 3-я Межд. науч. конф. “Суперкомпьютерные системы и их применение”. SSA’2010. Минск, 2010. **2**. 12–16.
4. *Варламов Д.А., Волохов В.М., Пивушков А.В., Сурков Н.Ф.* Виртуализация вычислительной среды в ГРИД // “Параллельные вычислительные технологии 2010” (ПаВТ-2010), Уфа, март 2010. Челябинск: Изд-во ЮУрГУ, 2010. 63–70.
5. *Волохов В.М., Пивушков А.В., Сурков Н.Ф., Варламов Д.А., Волохов А.В.* Новые методы решения задач вычислительной химии в распределенных средах // “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи.” 20–25 сентября 2010, Новороссийск. М.: Изд-во Моск. гос. ун-та, 2010. 181–184.
6. *Волохов В.М., Пивушков А.В., Волохов А.В., Варламов Д.А.* Реализация нескольких независимых ресурсных грид-сайтов на едином физическом пространстве кластера // X Межд. конф. “Высокопроизводительные параллельные вычисления на кластерных системах”. HPC-2010. Пермь, 2010. **1**. 119–124.

Поступила в редакцию
10.03.2011