Виртуализация вычислительной среды в ГРИД

Д.А.Варламов, Н.Ф.Сурков, В.М.Волохов, А.В.Пивушков

Данная статья посвящена описанию метода динамического формирования виртуальной среды выполнения для запуска сложно сконфигурированных прикладных пакетов вычислительной химии (на примере пакета GAMESS-US) в условиях параллельных сред на произвольных ГРИД ресурсах различных полигонов. Метод включает создание «виртуального контейнера» с образом среды исполнения, запуск его как исходящего ГРИД задания, развертывание на удаленном узле, настройку среды, исполнение задания, сбор результатов и «очистку» узла. Данный метод позволяет расширить возможности работы со сложными пакетами ПО в ГРИД средах.

Введение

Решение многих задач вычислительной химии, таких как моделирование молекулярных структур и их динамики, поведение и энергетика сложных химических реакций, наномоделирование невозможно без применения ГРИД вычислений. Например, время расчета поведения нанотрубок, легированных металлами, состоящих из $n*10^3$ атомов может достигать 3-4 лет для однопроцессорной системы. Детальнее особенности проведения квантово-химических расчетов в различных ГРИД средах описаны авторами ранее [1-4], а также в статье в данном сборнике трудов (Волохов и др., «Технологии ГРИД в вычислительной химии»)

На основе опыта работы в различных распределенных вычислительных средах авторами был сделан вывод, что наиболее существенными препятствиями на пути применения ГРИД технологий в вычислительной химии (а в общем случае – для любых сложно сконфигурированных прикладных пакетов ПО) стали следующие проблемы:

- разнородность доступных распределенных вычислительных ресурсов (на уровне архитектур процессоров, различных операционных систем, сетевых настроек, используемых параллельных сред и т.п.);
- необходимость создания для многих ресурсоемких параллельных приложений целой системы из конфигурационных настроек, дополнительных служб, специфичных параллельных сред, хранилищ данных и прочих компонентов информационно-вычислительной инфраструктуры;
- невозможность (или избыточная трудоемкость) перенастройки существующих вычислительных ресурсов (особенно класса “production farms”) для целей распределенных вычислений (что связано с особенностями операционных систем и используемого прикладного ПО, архитектуры, требований безопасности и т.д.) или под нужды конкретных прикладных пакетов.

Одним из способов решения данных проблем может стать применение интенсивно развиваемых в последнее время технологий виртуализации, включающих: (а) создание распределенных ресурсов и сервисов на базе виртуальных машин, (б) формирование виртуализованных «контейнеров-приложений» как единых распределенных задач; (в) создание полнофункциональных виртуальных машин, выступающих в роли исходящих/входящих распределенных заданий. Появление новых типов и архитектур процессоров (с поддержкой виртуализации на уровне ядра, интегрированных гипервизоров – на уровне материнских плат и т.п.), новых типов ПО уровня операционных систем и middleware со встроенными средствами виртуализации, развитие технологий быстрой передачи данных, удешевление Интернет-трафика дают основание предполагать, что технологии виртуализации займут ведущих мест в работе ГРИД ресурсов и инфраструктур.

Термин «виртуализация» используется авторами в двух основных смыслах: виртуализация вычислительных ГРИД ресурсов и сервисов и виртуализация вычислительного объекта (ОС, приложение и т.д.), перемещаемого в ГРИД среде. Потребность в виртуализации для распределенных вычислительных сред продиктована необходимостью создания и поддержки стандартных механизмов взаимодействия между пользователями и вычислительными ресурсами (серви-

сами), одинаковых со стороны ресурса (поставщика сервисов) и со стороны пользователя (вернее, используемого им интерфейса).

В 2009 году в рамках Программы фундаментальных исследований Президиума РАН № 1 на 2009-2011 годы «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 для ГРИД сред. Среди прочих задач проекта была поставлена цель изучить и применить (на реальных расчетах в области химии) ряд технологий виртуализации, включая:

- 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;
- адаптация прикладного программного обеспечения для работы в роли приложений в составе виртуальных динамически формируемых параллельных сред;
- В перспективе – создание образов виртуальных машин (с встроенными прикладными пакетами сложных конфигураций) для запуска их как заданий на ГРИД ресурсах.

Основные результаты в области виртуализации ГРИД ресурсов и приложений были описаны авторами ранее [5,6], в данной же статье детально описан метод динамического формирования виртуальной среды исполнения на *не подготовленном* удаленном ГРИД ресурсе с использованием технологии «виртуального контейнера».

Анализ среды выполнения ГРИД заданий в разных средах

Сегодня для комфортной работы многих приложений даже на уровне локального кластера требуется создание целой системы из приложений, служб, сетей, хранилищ данных и прочих компонентов современной информационно-вычислительной инфраструктуры, которые зачастую плохо совместимы с режимами работы ресурсного узла в целом. Для ряда пакетов прикладного ПО и сервисов нужно создавать комплексные среды с необходимым набором приложений и политиками 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 делать без прерывания работы узлов и остановки вычислительной среды, особенно в рамках «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 (например, 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, 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:

(1) 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);

(2) при настройке соответствующих скриптов запуска задания для каждого пользователя, согласно требованиям как приложения, так и системы параллельного программирования.

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 среды ГРИД,

поскольку на подобном ресурсе могут отсутствовать необходимые системные файлы.

Для решения данной проблемы авторами был проведен анализ процедуры исполнения типичного параллельного задания на ресурсном узле ГРИД (для сред gLite и Unicore), позволивший определить требования к создаваемому виртуальному образу среды исполнения, а также принципиальную возможность формирования динамической среды исполнения для тестируемых ресурсов. Также был сделан анализ систем параллельного программирования для выбора оптимального виртуального образа среды исполнения параллельного приложения на ГРИД ресурсах. В результате в качестве базового пакета для разработки виртуального образа среды исполнения параллельного приложения был выбрана среда Mpich-2.

В настоящее время в качестве распределенных ресурсов ГРИД используются, как правило, узлы в виде Linux-кластеров рабочих станций с операционной системой и некоторым набором приложений, настроенных для работы в среде ГРИД и специфичных для каждого из этих распределенных ресурсов. Единые стандарты на установку определенных типов программного обеспечения отсутствуют (если нет таковых в рамках какой-либо ВО). Поэтому на ресурсных узлах ГРИД среды, как правило, можно ожидать наличия только библиотек стандарта MPI-1. Добавим, что использование параллельных приложений в настоящее время в среде ГРИД ограничено как возможностями брокера ресурсов, который часто не распознает тип параллельного задания, так и отсутствием предустановленных на кластерах необходимых Run-time библиотек, а также отсутствием стандартов на размещение таких библиотек. Поэтому запуск параллельного сложно сконфигурированного задания на не указанном явно ресурсном узле распределенной среды обычно неэффективен, либо неудачен.

Проведенная в 2008-2009 годах работа была направлена на преодоление этих недостатков и преодоления специфичных особенностей среды ГРИД для запуска параллельных приложений.

Был проведен анализ предоставляемых псевдопользователям распределенных сред (так называемым «mapped users») прав доступа к ОС расчетного узла, которые определяют в свою очередь возможности работы внешнего задания с локальной файловой системой (обычно NFS), другими приложениями, исполняемыми модулями и утилитами. Для проведения такого анализа был самостоятельно разработан ряд оригинальных тестовых примеров, испытания которых позволили определить порядок запуска поступающих от брокера ресурсов заданий, в том числе:

- присвоенные заданиям имена пользователей и их права;
- создаваемые временные директории и требуемые файловые иерархии, в том числе на NFS ресурсах;
- доступность различных системных средств исполнения заданий (параллельные среды, доступ к очередям PBS и т.п.)

Это позволило определить требования к создаваемому виртуальному образу среды исполнения со стороны ОС ресурсного узла, а также принципиальную возможность динамической организации среды исполнения для тестируемых ресурсов.

Был сделан анализ имеющихся систем параллельного программирования для выбора оптимального виртуального образа среды исполнения параллельного приложения, в результате чего был выбран стандарт MPI (Message-Passing Interface, т.е. интерфейс для передачи сообщений), который предоставляет стандартные спецификации для библиотек передачи сообщений. В ИПХФ РАН длительное время используется реализация Mpich (свободно распространяемый Open Source Project, http://www.mcs.anl.gov/research/projects/mpich2), разработанная в Argonne National Laboratory.

Ранее на его основе в Суперкомпьютерном центре ИПХФ РАН для кластера на базе операционной системы ScientificLinux из исходных текстов была скомпилирована (после модификации участниками проекта) библиотека MPI. Основная причина использования менее распространенной версии стандарта MPI-2 (а не MPI-1) заключается в том, что использование стандарта MPI-1 не позволяет создать полностью однородную среду исполнения задания на узлах кластера, несмотря на ряд дополнительно разработанных сторонних программных пакетов. Стандартом MPI-2 однородная среда исполнения задания на всех процессорах рассматривается как базовая. Более того, с 2002 года стандарт MPI-1 не поддерживается Argonne National Laboratory, и была выпущена новая версия пакета Mpich, соответствующая стандарту MPI-2, в котором коренным образом изменена система запуска параллельных заданий и устранены указанные недостатки. Перед запуском задания осуществляется запуск кольца серверов mpd («mpd

ring»), одна из задач которых состоит в выравнивании среды окружения на главном и подчиненных узлах. В более общем случае, кольцо серверов может запускаться пользователем root, а остальные пользователи могут использовать это глобальное кольцо, однако, обычно на узлах ГРИД ресурса пользователю root запрещен любой удаленный доступ между узлами, и, следовательно, каждый псевдопользователь ГРИД должен запускать собственное локальное кольцо серверов (основанное на использовании непривилегированных портов).

Поэтому в качестве базового пакета для разработки виртуального образа среды исполнения параллельного приложения на распределенных ресурсах был выбран Mpich-2. Данный пакет представляет собой переносимую реализацию полных спецификаций MPI для широкого класса параллельных вычислительных средств, включающих кластеры рабочих станций и блоки массивно-параллельных процессоров (MPPs). Mpich-2 содержит, наряду с самими библиотеками MPI, программные средства для работы с программами MPI. Программные средства включают в себя переносимый стартовый механизм, несколько профилирующих библиотек для изучения производительности программ MPI.

При создании виртуального образа среды исполнения параллельного приложения на первых стадиях были введены ряд ограничений для ресурсных узлов:

- использована аппаратная архитектура x86, в настоящее время проводятся также работы с 64-битной версией;
- на расчетных узлах ГРИД ресурса используется операционная система Linux (клоны на базе RedHat – собственно RedHat, ScientificLinux, Fedora и т.п.), что связано с особенностями размещения системного ПО, хотя принципиальных ограничений на использование других ветвей Linux дистрибутивов нет;
- по стандарту настройки ресурсных узлов ГРИД для коммуникации между расчетными узлами используется интерфейс TCP/IP и беспарольный доступ по ssh (включая копирование файлов), возможна поддержка NFS ресурсов;
- Некоторые версии 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.

В процессе тестирования было выбрано 2 потенциальных статических места инсталляции библиотек – по месту загрузки исполняемого приложения (т.е. в home директории mapped-user) и использование общедоступной на большинстве расчетных узлов директории /tmp. Также рассмотрена (для кластеров с бесдисковыми рабочими узлами) возможность использования общих NFS ресурсов, доступных (на запись) ГРИД пользователю.

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

Создание прототипа виртуального «контейнера» с образом среды исполнения параллельного приложения

После анализа процедуры выполнения ГРИД задания (в разных средах) и выбора среды параллельных вычислений была разработана технология создания динамически формируемых образов исполняемых сред, или виртуальных «контейнеров»

В соответствии с определенными выше требованиями ресурсных узлов был сформирован перемещаемый программный пакет MPI-2, тестирование которого на ГРИД кластере ИПХФ РАН с использованием тестовых примеров показало его полную работоспособность. Полученный пакет в дальнейшем использовался в качестве *базового* прототипа для разработки виртуального образа среды исполнения конкретных параллельных приложений, в том числе сложных прикладных пакетов.

В качестве первичного тестового приложения был 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 средах ГРИД (для сред gLite и Unicore) была разработана серия низкоуровневых скриптов пользовательского интерфейса.

Была принята следующая схема запуска: на удаленный ресурсный узел сети ГРИД через брокер ресурсов (или непосредственно – как, например, в Globus) передается главный скрипт и упакованный «контейнер», содержащий исполняемые файлы, необходимые системные библиотеки, файлы конфигурации и данных. Далее главный скрипт реализует следующую последовательность шагов по месту исполнения:

1. Сбор начальной информации о текущем ресурсном узле ГРИД
2. Распаковка «контейнера» в рабочей директории псевдопользователя ГРИД и перемещение библиотек в локальную директорию /tmp на текущем локальном узле (ТЛУ) или в доступную NFS область.
3. Создание файла mpd.conf (на базе шаблона) на ТЛУ для запуска MPI-2 сервера mpd.
4. Переопределение на ТЛУ текущего значения ряда переменных среды окружения для ГРИД пользователя.
5. Запуск сервера mpd (с правами mapped-user) на стартовом узле и проведение его run-time тестирования.
6. Сбор информации о доступных узлах и их текущем состоянии (обычно опрос PBS сервера). Список свободных узлов собирается в файл mpd.hosts, необходимый для запуска «кольца» серверов mpd.
7. Распределение необходимых библиотек по списку свободных узлов по протоколу ssh (или в определяемую NFS директорию)
8. Запуск «кольца» серверов mpd на ресурсном узле ГРИД и его тестирование.
9. Запуск параллельного приложения и его работа как обычного распределенного задания с последующей передачей результатов на брокер ресурсов и затем – на пользовательский интерфейс
10. Удаление всех библиотек и созданных временных файлов со всех узлов или из NFS папки

Отметим, что возможно распространение необходимых файлов на доступные расчетные узлы как через ssh, так и путем организации единой папки на общем NFS ресурсе.

Работа тестового варианта пакета была отлажена на ресурсном узле ГРИД ИПХФ РАН (использовался как удаленный ресурс) в условиях сред gLite и Unicore. Дальнейшее тестирование было проведено на ресурсных узлах RDIG в рамках ВО RGSTEST (узлы НИИЯФ МГУ).

Создание виртуального «контейнера» на примере параллельного квантово-химического приложения GAMESS-US

Для тестирования разработанного метода на примере конкретного прикладного пакета был выбран квантово-химический пакет GAMESS-US.

GAMESS-US (http://www.msg.ameslab.gov/GAMESS) – одна из популярных программ для теоретического исследования свойств химических систем, уступает по известности лишь 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 рабочих станций UNIX. Пакет отличается сложностью установки и конфигурации, а также требует нестандартных настроек параллельной среды вычислений.

Работы по распараллеливанию GAMESS-US начались еще в 1991 году. Однако использование методов передачи сообщений MPI получило применение только с 1999 г., когда в пакете GAMESS-US была реализована модель интерфейса с распределенным размещение 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 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 распределенного middleware; (г) на ряде современных кластеров (например «Чебышёв» в НИВЦ МГУ) сокетная модель неработоспособна из-за политик безопасности кластера.

Конфигурации же GAMESS-US с использованием библиотеки MPI авторами пакета разработаны только для ряда мейнфреймов известных производителей (Cray, IBM, SGI). В общем случае конфигурации с MPI не рекомендуются, и желающим предлагается экспериментировать с такими конфигурациями самостоятельно.

Для работы с пакетом GAMESS-US в среде ГРИД 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 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:

- во-первых, из-за особенностей запуска исполняемого задания GAMESS-US, которая осуществляется скриптом, активизирующем более 150 переменных окружения. На главном узле среда создается правильно, но 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 операционной системой ScientificLinux эту функцию безопасного сервера использовать не удалось. При запуске задания на локальном узле пакет Mpich-1 использует команды оболочки такие, как ‘.’ , `eval`, `exec`, которые не наследуют среду окружения запускающего процесса, что ведет к краху дочерних процессов;
- во-вторых – особенности реализации команды запуска параллельных заданий `mpirun` пакета Mpich-1. Запуск 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 строчных аргументов задания GAMESS-US нарушается и не распознается, что ведет к краху запуска.

В ИПХФ РАН с целью расширения функциональности применения пакета GAMESS-US в сети ГРИД 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 Mpich2) 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.

Серия первичных запусков (с использованием внутренних тестовых примеров собственно пакета GAMESS-US) вплоть до получения положительного результата тестов (равно-значность поведения сокетных и MPI вариантов) была проведена на ресурсном сайте ГРИД ИПХФ РАН (grid-ce.icp.ac.ru) для сред gLite и Unicore (узлы использовались как удаленные ресурсы, т.е. запуск задач шел через ГРИД инфраструктуру). Дальнейшее успешное тестирование было проведено на удаленных ресурсных узлах RDIG в рамках ВО 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.

Ввиду того, что на ряде кластеров (например, в Курчатовском РНЦ) запрещено или затруднено использование скриптовых языков, нами проведены работы по переводу всех действий по развертыванию и настройке подобных «контейнеров» в полностью бинарные исполняемые программы, которые действуют схожим образом, но не требуют доступа к shell языкам. Таким образом, впервые разработана технология запуска GAMESS-US в ГРИД среде 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 mpich2, запускает параллельные процессы GAMESS-US, собирает полученные результаты, удаляет «мусор» и отправляет выходные данные на пользовательский интерфейс ГРИД среды.

В результате 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 на всех выделенных узлах ресурса ГРИД, 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 перспективе более общим вариантом данных технологий является использование (по аналогии с описанным «контейнером») виртуальных машин 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 данные 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. Прежде всего – это размер передаваемых

заданий (может достигать первых гигабайтов) и «накладные» расходы на виртуализацию (до 15-20% от мощности ресурса, при оптимальном конфигурации они могут быть снижены до уровня 5-7%).

Заключение

Разработан метод создания виртуальных перемещаемых «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» он средствами распределенной среды как обычное ГРИД задание доставляется на удаленный ресурсный узел, «разворачивается», настраивает среду узла «под себя» и запускается как обычное параллельное приложение. По окончании работы приложения происходит «очистка» среды выполнения и возврат результатов на пользовательский интерфейс. Созданы два варианта «контейнеров»: с использованием скриптовых языков и как единого бинарного задания. В настоящее время этот метод применим для исполнения на 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 и Unicore. Нет принципиальных ограничений для создания подобных «контейнеров» для других прикладных пакетов, требующих специфических параметров окружения и нестандартных настроек параллельных сред выполнения.

Применение данного метода виртуализации позволит существенно расширить круг доступных ГРИД ресурсов для выполнения на них сложно сконфигурированных прикладных пакетов.

Литература

1. В.М.Волохов, Д.А.Варламов, А.В.Пивушков, Н.Ф.Сурков, Г.А.Покатович ГРИД и вычислительная химия // "Вычислительные методы и программирование", М.: МГУ, 2009, т.10, № 2,с.78-88
2. С.М. Алдошин, В.М. Волохов, Д.А. Варламов, А.В. Пивушков Вычислительная химия в среде ГРИД: параллельные и распределенные вычисления // Вторая международная конференция «Суперкомпьютерные системы и их применение» SSA’2008, Минск, октябрь 2008; Минск, ОИПИ НАН Беларуси, с.114-118
3. Варламов Д.А., Волохов В.М., Пивушков А.В., Сурков Н.Ф., Покатович Г.А. Распределенные и параллельные вычисления в области химии на ресурсном узле ГРИД ИПХФ РАН // Сб. науч.тр. “Distributed Computing and Grid-Technologies in Science and Education: Extended Proceedings of the 3rd Intern.Conf.”, Дубна, изд-во ОИЯИ, 2008, с.127-130
4. В.М. Волохов, Д.А. Варламов, А.В. Пивушков Крупномасштабные задачи химии на параллельных и распределенных вычислительных полигонах: современное состояние и перспективы // “Научный сервис в сети Интернет: решение больших задач», Всероссийская научная конференция, (г. Новороссийск, 22-27 сентября 2008) – М.; Изд-во МГУ, 2008, с.210-212
5. В.М. Волохов, Д.А. Варламов, Н.Ф. Сурков, А.В. Пивушков Виртуальные вычислительные среды: использование на ГРИД полигонах // Вестн. ЮУрГУ, серия «Математическое моделирование и программирование», 2009, № 17 (150), вып. 3, с.24-35.
6. Д.А.Варламов, В.М.Волохов, А.В.Пивушков, Н.Ф.Сурков Технологии виртуализации ресурсов и приложений вычислительной химии для использования на ГРИД полигонах // «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность», Труды Всероссийской суперкомпьютерной конференции (21-26 сентября 2009 г., г. Новороссийск). – М.: Изд-во МГУ, 2009. – 524 с., с.378-381