ТЕХНОЛОГИИ ВИРТУАЛИЗАЦИИ РЕСУРСОВ И ПРИЛОЖЕНИЙ ВЫЧИСЛИТЕЛЬНОЙ ХИМИИ ДЛЯ ИСПОЛЬЗОВАНИЯ НА GRID ПОЛИГОНАХ

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

Наиболее существенными препятствиями для применения GRID технологий в вычислительной химии стали следующие проблемы: (а) гетерогенность ресурсов; (б) необходимость создания для многих ресурсоемких приложений системы из конфигурационных настроек, хранилищ данных, сервисов и прочих компонентов вычислительной инфраструктуры; (в) невозможность (избыточная трудоемкость) перенастройки вычислительных ресурсов (особенно класса “production farms”) для целей распределенных вычислений (из-за особенностей ОС и прикладного ПО, архитектуры, требований безопасности).

Данные проблемы во многом могут быть решены применением технологий виртуализации вычислений на уровне: (а) виртуальных распределенных ресурсов, (б) виртуализованных «контейнеров-приложений» и (в) полнофункциональных виртуальных машин (ВМ) в роли исходящих распределенных заданий. Появление новых типов и архитектур процессоров (виртуализация на уровне ядра, интегрированные гипервизоры на уровне материнских плат), новых типов ПО уровня ОС и middleware со встроенными средствами виртуализации, технологии быстрой передачи данных, удешевление Интернет-трафика привели к «взрывному» росту исследований и рынка ПО виртуализации в последние годы.

Термин «виртуализация» далее используется в двух смыслах: виртуализация вычислительного GRID ресурса и виртуализация вычислительного объекта, перемещаемого в GRID среде. Потребность в виртуализации для распределенных сред продиктована необходимостью создания и поддержки стандартных механизмов взаимодействия между пользователями и вычислительными ресурсами (сервисами), одинаковых со стороны ресурса (поставщика сервисов) и со стороны пользователя (вернее, используемого им интерфейса).

Спектр использования средств виртуализации и виртуальных машин широк: объединение различных вычислительных и управляющих сервисов на единых физических ресурсах, возможность разработки и тестирования прикладного ПО в различных средах, миграция приложений и служб. Возможность снизить затраты на приобретении дополнительных физических серверов, электропитание и охлаждение для них, возможность виртуальной изоляции чрезмерно ресурсоемких или потенциально опасных приложений, мобильность и управляемость виртуальных ресурсов и приложений создают ощутимые преимущества для вычислительных центров и обеспечивают создание виртуализованной вычислительной инфраструктуры.

Создание виртуальных вычислительных ресурсов в виде ВМ (виртуальных машин) на базе существующих физических узлов

Сегодня для комфортной работы многих прикладных пакетов требуется создание системы из приложений, служб, хранилищ данных и других компонентов вычислительной инфраструктуры, которые часто плохо совместимы с режимами работы ресурсного узла в целом. Для ряда пакетов ПО и сервисов нужно создавать комплексные среды с необходимым набором приложений и политиками безопасности. Ключевыми требованиями являются скорость и простота предоставления таких сред, тщательная изоляция друг от друга, квотирование вычислительных ресурсов для каждой среды, независимость от базовых настроек узла. Часто 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. Данная задача может быть решена путем установки и настройки ВМ на существующих ресурсных узлах. Они будут обслуживать сервисы и конкретные входящие приложения с наибольшей эффективностью и практически полной безопасностью для базовой инфраструктуры ресурсного узла (включая ОС, прикладное ПО, квотирование ресурсов). Данная идея не является сугубо новой (проекты «Дубна-Грид», «Скиф-Home»), однако только развитие производительности узлов, архитектуры процессоров и соответствующего ПО позволяет на новом уровне подойти к решению данной проблемы.

При создании виртуальных машин в качестве ресурса пользователю распределенной среды может быть предоставлена изолированная виртуальная вычислительная среда, по своим свойствам не уступающая физическому серверу, в которой может быть запущен любой вычислительный сервис. Приложения, реализованные в ВМ, в этом случае не зависят от «хозяйской» операционной системы и ее окружения. Еще одним преимуществом такой виртуализации является полное отделение конкретной службы или программы от внешней среды (как и внешней среды от нее). Это возможно благодаря использованию дополнительного слоя программного обеспечения – виртуализованного оборудования, позволяющего выполнять обычное приложение, как если бы оно было запущено на отдельном компьютере. Как может быть использована данная технология в условиях GRID среды? Имеется возможность создать образ виртуальной машины (машины с предустановленной ОС и полностью сконфигурированными приложениями, нацеленной на решение конкретных задач), который не требует перенастройки физического узла под конкретные задачи.

Большим преимуществом является мобильность виртуальных машин (переносимых при необходимости с одного сервера на другой) и их толерантность к аппаратным сбоям (в случае поломки физического узла копия виртуальной машины запускается на другом). Заметим, что коммерческое ПО виртуализации в настоящее время позволяет производить перенос виртуальных машин между физическими узлами без их останова (!).

Нами для создания рабочих виртуальных машин был выбран гипервизор Xen (свободно распространяемый вариант), входящий в состав ОС Scientific Linux 5 (SL), на которой реализован ресурсный GRID узел ИПХФ РАН.

ВМ SL, сконфигурированная как 2-х процессорный рабочий узел с поддержкой PBS Torque и MPI для ресурсного узла gLite, была запущена на физической машине и введена в список рабочих узлов на Computing Elements ресурсного узла gLite, после чего с ее использованием был просчитан ряд задач. С точки зрения внешнего пользователя, никаких отличий от физического узла в использовании виртуальной машины не наблюдалось, т.е. в составе ресурсного узла произошло добавление расчетного узла с примерно теми же вычислительными параметрами, что и на физических машинах.

Для ВМ на основе Ubuntu 8.10 была произведена конфигурация в роли расчетных узлов созданного в ИПХФ ресурсного сайта СКИФ-Полигона на базе middleware Unicore. Созданные виртуальные машины были перенесены на расчетные ноды ресурсного GRID узла ИПХФ (ОС Scientific Linux 4.5, gLite) и запущены под управлением гипервизора Xen. Тестирование показало их полную работоспособность в данной обстановке.

Проведен эксперимент по запуску ВМ Ubuntu с выполнением базового сервиса ресурсного узла Unicore – TSI (Target system interface). В настоящее время в составе ресурсного сайта ИПХФ на базе Unicore («СКИФ-Полигон») работают 2 виртуальные машины в качестве расчетных узлов на базе физических машин, входящих в состав ресурсного узла gLite. А на базе физической машины, поддерживающей базовые компоненты ресурсного узла Unicore (Gateway; UAS, Unicore atomic services; XUUDB, Unicore user database) протестирована работа изолированной ВМ с поддержкой сервиса TSI (Target system interface).

Показана возможность установки, запуска и работы виртуальных машин с поддержкой различных сервисов на существующих физических узлах без значительного вмешательства в рабочее пространство (workspace) распределенных и/или параллельных сред ресурсных узлов. Это дает возможность разворачивать распределенные вычислительные полигоны на уже существующих вычислительных кластерах без кардинальной перенастройки их аппаратно-программной конфигурации, что особенно важно для постоянно работающих центров класса «production farm».

Среди обнаруженных проблем виртуализации отметим учет ресурсов физического узла и мониторинг выполняемых ресурсным узлом задач, т.е. учет ресурсов, востребованных ВМ, пока проводится на уровне гипервизора и не вполне корректно оценивается внешним мониторингом распределенной среды со стороны CE (Computing Element) или MON (Monitor Box) среды gLite, что может вести к недоучету ресурсов и завышенными ожиданиям со стороны входящих задач. Однако, в планах развития большинства гипервизоров показана возможность решения этих проблем.

Создание «виртуального контейнера» с образом среды исполнения параллельного приложения

Одним из вариантов виртуализации прикладного программного обеспечения для работы в распределенных средах становится виртуализованное приложение, которое в виде «виртуального контейнера» доставляется на ресурсный узел вместе со всеми конфигурационными настройками, относящимися к операционной системе и приложению, и не требует процедуры предварительной установки. 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 ними в качестве GRID задачи с последующим запуском в изолированном «контейнере». Так могут быть решены проблемы установки, настройки, несовместимости с операционной системой и другими программами, разрешаются конфликты одинаковых приложений.

Нами был разработана технология создания виртуальных «контейнеров», перемещаемых стандартными средствами распределенного middleware и запускаемых на удаленном ресурсном узле распределенной среды.

Был проведен анализ процедуры исполнения типичного параллельного задания на ресурсном узле GRID, позволивший определить требования к создаваемому виртуальному образу среды исполнения, а также принципиальную возможность динамической организации среды исполнения для тестируемых типов ресурсов. Проведен анализ разных систем параллельного программирования для выбора оптимального виртуального образа среды исполнения параллельного приложения. В качестве базового пакета для разработки виртуального образа среды исполнения параллельного приложения был выбран пакет Mpich-2.

В качестве распределенных ресурсов GRID используются, как правило, узлы в виде кластеров рабочих станций с операционной системой и некоторым набором приложений, настроенных для работы в среде GRID и специфичных для каждого из этих распределенных ресурсов. Для проведения параллельных вычислений требуется наличие установленной на ресурсных узлах системы параллельного программирования (MPI, OpenMP), причем стандарты на установку данного ПО отсутствуют. Как правило, на ресурсных узлах GRID

среды можно ожидать наличия только библиотек стандарта MPI-1. Использование параллельных приложений в настоящее время в среде GRID также крайне ограничено как возможностями брокера ресурсов, который не распознает тип параллельного задания, так и отсутствием предустановленных на кластерах необходимых Runtime библиотек, а также отсутствием стандартов на размещение таких библиотек. Поэтому запуск параллельного сложно сконфигурированного ресурсоемкого задания на произвольном ресурсе среды GRID часто неэффективен, либо неудачен.

Был проведен анализ предоставляемых псевдопользователям распределенных сред (так называемым «mapped users») прав доступа к ОС расчетного узла, которые определяют возможности работы внешнего задания с локальной файловой системой, другими приложениями, исполняемыми модулями и утилитами. Такой анализ позволил определить требования к создаваемому виртуальному образу среды исполнения со стороны ОС ресурсного узла, а также принципиальную возможность динамической организации среды исполнения для тестируемых ресурсов.

На основе версии Mpich-2 в вычислительном центре ИПХФ РАН для кластера, работающего под Scientific Linux, из исходных текстов была скомпилирована (после соответствующей доработки авторами) модифицированная библиотека MPI-2. Основная причина использования менее распространенной версии стандарта MPI-2 заключается в том, что, как показали проведенные тесты, стандарт 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 всегда рассматривается как базовая.

При создании виртуального образа среды исполнения параллельного приложения было принято решение ввести ряд ограничений/допущений для ресурсных узлов: (а) использована архитектура x86, наиболее частая на ресурсных узлах; (б) на расчетных узлах GRID ресурса используется операционная система на базе клонов RedHat; (в) для коммуникаций между узлами используется интерфейс TCP/IP и беспарольный доступ по ssh; (г) не использованы поставляемые производителями пакетов сетевые драйверы. Однако, принципиальных ограничений на работу в более «расширенных» условиях, включая переход на 64-битную среду, нет.

Были выбраны два статических места инсталляция библиотек – по месту загрузки исполняемого приложения и общедоступная на большинстве расчетных узлов директория /tmp. Разработка системы динамического компилирования и инсталляции библиотек на данном этапе не рассматривалась.

В качестве первичного тестового приложения был использована программа вычисления числа pi из пакета Mpich-2. Исходная тестовая программа была доработана с учетом особенностей запуска прикладных приложений, был получен ее исполняемый модуль и скрипты запуска с использованием библиотек Mpich-2. Тестовый модуль и перемещаемый пакет Mpich-2 были собраны и упакованы в единый пакет, для запуска которого в среде GRID (gLite) была разработана серия низкоуровневых скриптов. Отметим, что нет принципиальных препятствий для формирования такого пакета для работы в условиях других распределенных сред (типа Unicore).

Принята следующая схема запуска: на удаленный ресурсный узел сети GRID через брокер ресурсов передается главный скрипт и упакованный модуль, содержащий исполняемые файлы приложения, необходимые MPI-2 библиотеки и скрипты запуска и выполнения приложения, т.е. динамически сформированный «виртуальный контейнер». После доставки «контейнера» на ресурсный узел проводится ряд шагов по месту исполнения, позволяющие произвести запуск «кольца» серверов mpd на узле GRID с последующим запуском параллельного приложения как обычного распределенного задания с финальной передачей результатов на брокер ресурсов и затем пользователю.

Запуск пакета GAMESS-US в составе «виртуального контейнера»

Используемые в вычислительной химии прикладные пакеты (GAMESS, Gaussian, CPMD, NAMD) требуют обязательной настройки большого количества переменных окружения операционной системы до запуска параллельного приложения на ресурсном узле, а также значительной настройки ОС не только вычислительного элемента, но и всех подчиненных ему рабочих узлов. Такая настройка обычно осуществляется в два этапа: (а) при установке прикладного ПО системным администратором на каждом расчетном узле ресурсного центра на уровне операционной системы; (б) при настройке соответствующих скриптов запуска задания для каждого пользователя как для приложения, так и для системы параллельного программирования.

Применение «виртуальных контейнеров» позволяет передавать заранее настроенную среду как единое задание, не требующее дополнительного конфигурирования и сложной процедуры установки и настройки.

Выбранный подход по созданию виртуального образа среды исполнения на основе перемещаемого пакета MPI-2 был применен для компиляции модифицированного пакета исходных кодов GAMESS и показал свою эффективность.

GAMESS-US (http://www.msg.ameslab.gov/GAMESS) – одна из наиболее популярных программ для теоретического исследования свойств химических систем, позволяет рассчитывать энергию, структуры молекул, частоты их колебаний, а также разнообразные свойства молекул в газовой фазе и в растворе, в основном и в возбужденных состояниях. Основное направление – развитие методов расчета сверхбольших молекулярных систем;

Работы по распараллеливанию GAMESS начались в 1991 году. Однако, использование методов передачи сообщений MPI получило применение только с 1999 г., когда в пакете GAMESS была реализована модель интерфейса с распределенным размещение данных (DDI – Data Distributed Interface). Последняя версия интерфейса DDI, оптимизированная для многопроцессорных SMP-архитектур общего вида, была выпущена в мае 2004 г. Сейчас 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 GRID 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.

Конфигурации GAMESS с использованием MPI библиотек разработаны только для ряда мейнфреймов (Cray, IBM, SGI), а в общем случае конфигурации с MPI не рекомендуются.

Была поставлена задача разработки оригинальной конфигурации и сборки исполняемого файла GAMESS с использованием библиотеки MPI-2 из исходных текстов пакета GAMESS, учитывая то, что установка библиотек MPI на многопроцессорных системах и кластерах под управлением Linux в настоящее время является общепринятой.

Получить работоспособную конфигурацию пакета GAMESS для MPI-1 не удалось по ряду принципиальных причин, поэтому была использована новая версия пакета Mpich, соответствующая стандарт MPI-2, в котором коренным образом изменена система запуска параллельных заданий и устранены недостатки по передаче переменных окружения и поддержке системы запуска. Перед запуском задания на нодах узла 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, на узлах GRID пользователю root запрещен любой удаленный доступ между узлами, поэтому каждый псевдопользователь GRID должен запускать собственное локальное кольцо mpd серверов, основанное на использовании непривилегированных портов.

После рекомпиляции GAMESS был сформирован «виртуальный контейнер» из модифицированного GAMESS, бинарных библиотек и исполняемых файлов Mpich-2, скриптов развертывания и запуска на создаваемой параллельной среде. Серия запусков с использованием тестовых примеров пакета GAMESS была проведена на ресурсном узле ИПХФ (ru-Chernogolovka-IPCP-LCG2) через брокер ресурсов RDIG. Далее тестирование было проведено на «чистом» от установки GAMESS ресурсном узле RDIG (BO RGSTEST – НИИЯФ МГУ, lcg38.sinp.msu.ru). Были проведены успешные запуски пакета GAMESS (рассчитаны тестовые примеры различного уровня сложности из дистрибутива GAMESS), подтвердившие полную работоспособность разработанной технологии.

В перспективе более общим вариантом данной технологии является использование виртуальных машин как исходящих распределенных заданий, что позволит гарантировать GRID пользователю необходимое качество обслуживания, не затрагивающее при этом работу основных служб ресурсных узлов. В чем смысл данной технологии в условиях GRID среды? Пользователь получает возможность создать образ виртуальной машины (машины с полностью сконфигурированными приложениями, нацеленной на решение конкретной задачи), который затем передается на распределенный ресурс и действует там как GRID-приложение, не требуя настройки данного узла под конкретные задачи. Это существенно облегчит 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). Каковы 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%).

Таким образом, показана широкая степень применимости различных методов виртуализации для работы с приложениями вычислительной химии в условиях распределенных и параллельных сред.