# НОВЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ВЫЧИСЛИТЕЛЬНОЙ ХИМИИ В РАСПРЕДЕЛЕННЫХ СРЕДАХ

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

Решение многих задач вычислительной химии (моделирование молекулярных структур и их динамики, поведение и энергетика сложных химических реакций, наномоделирование) невозможно без грид вычислений. Например, предполагаемое время расчета нанотрубок, легированных металлами и состоящих из $n*10^3$ атомов – 3-4 года для однопроцессорной системы. Особенности проведения квантово-химических расчетов в различных грид средах (gLite, Unicore, Globus) описаны авторами ранее [1], а также сформулированы в статье в данном сборнике трудов [Волохов и др.]. Здесь же описаны оригинальные вычислительные методики, разработанные авторами для решения специфических проблем запуска ПО вычислительной химии в распределенных средах. Сразу заметим, что данные методики могут быть применены для решения широкого круга и других задач, не связанных с химией.

Работа с «пучками» формально независимых заданий

Метод разработан для решения задач, распадающихся на громадную совокупность независимых заданий, число которых зависит от количества параметров задачи или от «сетки» разбиения искомой области данных. Таковыми являются многопараметрические задачи вычислительной химии и химической физики, требующие последовательного перебора большого количества входных параметров. При этом полная задача разбивается на огромное количество независимых подзадач (каждая определяется группой значений совокупности параметров). Как вариант - решение задач на больших областях исходных данных, когда результат в каждой решаемой точке не зависит от «соседей». Задача автоматизации процесса разбиения полной задачи на фрагменты («нарезка») определяет полноту получаемого решения и удобство пользования системой. Типичный пример - туннельные реакции под воздействием электромагнитного излучения,где параметрами являются разные характеристики излучения. Задача имеет высокую вычислительную сложность, однако вычисления в каждой точке сетки в ней происходят независимо друг от друга, поэтому возможно разбить область вычислений на множество непересекающихся подобластей и каждую из них рассчитывать на разных узлах. Схожим примером могут служить траекторные расчеты химических реакций, например, реакция $H_2 + O_2$. Расчет ее представляет собой компьютерное моделирование с использованием классических траекторий элементарного акта столкновения. Для расчета полного сечения реакции для набора необходимой статистики следует расчитать до десятков миллионов траекторий с учетом углов взаимной ориентации молекул, начальных колебательных и вращательных квантовых чисел, параметров столкновения и т.п. Как правило, расчет независимого задания в одном из узлов расчетной многомерной «сетки» занимает от нескольких минут до первых часов, однако, общее время расчета становится нереальным для одиночной вычислительной системы. Поэтому был создан метод запуска “пучков” независимых заданий для использования всех доступных ресурсов распределенной среды с использованием грид технологий.

Для разных распределенных сред была разработана методика запуска подмножества независимых заданий (составляющих в совокупности вычислительную задачу) и получения результатов с удаленных ресурсных узлов. Первоначально метод был реализован в локальной гетерогенной вычислительной среде с использованием middleware Condor, что показало высокую эффективность метода. Затем метод был испытан на ряде задач в средах Nimrod и X-Com на географически распределенных вычислительных ресурсах. Далее, на языке Perl был написан комплекс скриптов для формирования “пучков” заданий, их запуска и получения результатов счета с использованием пользовательских интерфейсов (UI) сред gLite и Unicore. Для решения многопараметрических задач квантовой химии были разработаны методы формирования «пучков» независимых заданий с варьирующими параметрами – до $10^4$, в перспективе до $10^7$ «атомарных» заданий на задачу. Для выбранных областей данных авторскими скриптами производится «нарезка» областей данных или входных параметров, формирование пулов независимых заданий, создание очередей запуска и отправки заданий на брокер ресурсов. После запуска периодически запускаемые средствами ОС (например, по *cron*) скрипты UI ведут мониторинг выполнения заданий (опрашивая брокер ресурсов или непосредственно удаленные ресурсные узлы), контроль тайм-аутов, перезапуск неудачных заданий и сбор результатов (с использованием базы данных и таблиц в ней, контролирующих состояние заданий – «ожидание», «запуск», «выполнение» и т.д.). По окончании расчетов проводится сборка «атомарных» результатов в единый выходной файл. Для части задач (требующих значительного числа параллельных независимых расчетов) в настоящее время создаются механизмы по разбиению областей данных (или расчетов) на большие независимые «подсетки» или независимые задания, передачи всех их интерфейсам распределенных сред с последующим запуском на параллельных узлах и «сборки» финальных результатов из множества полученных независимых. Это позволяет достигнуть разумного баланса между собственно временем счета заданий и накладными расходами по передаче

заданий по распределенной среде. При тестировании метода были сформированы и направлены на распределенные ресурсы ВО RGSTEST и СКИФ-полигона "пучки" заданий, осуществлен мониторинг их выполнения, "сборка" результатов с различных ресурсов. Для ВО RGSTEST было задействовано до 400 процессоров на различных ресурсных узлах (Москва, Протвино, Харьков, Черноголовка), для СКИФ-Полигона – доступные на момент счета CPU узлов (ИПХФ в роли удаленного ресурса, узлы ИПС РАН, СКИФ-Cyberia (Томский ГУ), СевКазГУ, ННГУ) — всего до 120. В настоящее время метод «пучков» адаптируется к работе на полигоне ГридННС в среде Globus.

В целом метод более оптимален для работы через брокер ресурсов, однако, позволяет работать и в режиме point-to-point, т.е. при непосредственном обращении UI к ресурсным узлам. Функциональность метода работы с «пучками» заданий интегрирована в Грид-портал ИПХФ РАН (http://grid.icp.ac.ru) для решения многопараметрических задач .

Метод создания «виртуальных контейнеров» для параллельных приложений

На основе опыта работы авторами был сделан вывод, что значительными препятствиями на пути применения грид технологий в вычислительной химии (а в целом – любых сложно сконфигурированных прикладных пакетов) являются следующие проблемы:

- гетерогенность распределенных вычислительных узлов (на уровне архитектур процессоров, операционных систем, сетевых настроек, параллельных сред и т.п.);
- необходимость создания вычислительной среды для многих ресурсоемких параллельных приложений, состоящей из конфигурационных настроек, дополнительных служб, специфичных параллельных сред, хранилищ данных и прочих компонентов;
- невозможность (или избыточная трудоемкость) перенастройки работающих вычислительных ресурсов (особенно класса “production farms”) для целей распределенных вычислений или под нужды конкретных прикладных пакетов.

Одним из способов решения большинства этих проблем может стать применение технологий виртуализации, включающих: (а) создание распределенных ресурсов и сервисов на базе виртуальных машин, (б) формирование виртуальных «контейнеров-приложений» как единых распределенных задач; (в) создание полнофункциональных виртуальных машин, выступающих в роли исходящих/входящих распределенных заданий.

Рассмотрим один из методов виртуализации вычислительного объекта (параллельного приложения), перемещаемого и выполняемого в грид среде. Метод основан на реализации виртуальной динамически формируемой параллельной MPI среды в форме «виртуального контейнера». Он включает адаптацию программы (прикладного пакета) для работы в роли приложения в составе «контейнера», создание образа виртуальной среды исполнения, формирование собственно «контейнера» и процесс выполнения его как исходящего грид задания на **неподготовленном** удаленном грид узле (т.е без его предустановки). Применение метода показано на примере пакета GAMESS-US, одного из наиболее востребованных пользователями квантово-химических пакетов.

Для работы многих приложений на уровне локального кластера требуется создание целой системы из приложений, служб, сетей, хранилищ данных и прочих компонентов вычислительной инфраструктуры, которые часто плохо совместимы с режимами работы ресурсного узла в целом. Для пакетов прикладного ПО и сервисов нужны 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. Часто это необходимо делать без прерывания работы узлов и остановки вычислительной среды, особенно для «production farms», не допускающих долгих остановок и реконфигурирования системы.

Другой проблемой решения задач (особенно параллельных) в условиях распределенных вычислений является необходимость виртуализации программных. Например, для проведения параллельных вычислений требуется наличие установленной на ресурсных узлах системы параллельного программирования (MPI, OpenMP и др.) или предустановленных специфичных математических библиотек. Используемые пакеты прикладных программ вычислительной химии (GAMESS, Gaussian, NAMD и др.), как и большинство инженерных пакетов, отличаются сложностью конфигураций и повышенными требованиями к среде выполнения, особенно для параллельных расчетов. Они требуют обязательной настройки большого количества переменных окружения операционной системы до запуска параллельного приложения на каждом из расчетных узлов. Такая настройка обычно осуществляется в два этапа:

1. при ручной/полуавтоматической установке ПО системным администратором на каждом узле ресурсного сайта на уровне операционной системы (например, при формировании сайтов виртуальной организации);
2. при настройке соответствующих скриптов запуска задания для каждого пользователя, согласно требованиям приложения и системы параллельного программирования.

При этом даже традиционный подход со статическим линкованием библиотек (не говоря уже о динамическом) к исполняемому модулю часто не способен создать полностью работоспособное параллельное задание на произвольном ресурсе среды грид из-за отсутствия на ресурсе необходимых системных файлов.

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 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 описан здесь [2].

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 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 (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 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 узле грид, распаковка «контейнера» в директории псевдопользователя грид 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. Детально последовательность работы «контейнера» также описана здесь [2]

Тестовый вариант «контейнера» был отлажен на ресурсном узле грид ИПХФ (в качестве удаленного ресурса) для сред gLite и Unicore. Дальнейшее тестирование было проведено на ресурсных узлах RDIG в рамках BO RGSTEST (узлы НИИЯФ МГУ).

GAMESS-US (http://www.msg.ameslab.gov/GAMESS) – одна из популярных программ для теоретического исследования свойств химических систем, 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.

В пакете GAMESS-US ранее была реализована модель интерфейса с распределенным размещение данных (DDI – Data Distributed Interface), которая была оптимизирована для многопроцессорных SMP-архитектур общего вида, особенно работающих с памятью в стиле System V. 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 (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 кластеров сокетная модель неработоспособна из-за политик безопасности кластера.

Конфигурации же GAMESS-US с использованием библиотеки MPI авторами пакета разработаны только для ряда мейнфреймов (Cray, IBM, SGI). В общем случае конфигурации с MPI не рекомендуются.

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 (реализация 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 GAMESS-US для MPI-1 не удалось по двум причинам:

- запуск исполняемого задания GAMESS-US осуществляется скриптом, активизирующим 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 “secure server”, одной из задач которого являлась ликвидация этого недостатка. Но из-за неполной совместимости с клонами RedHat эту функцию безопасного сервера использовать не удается. 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;
- 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. Стандартное расположение строчных аргументов задания GAMESS-US нарушается, что ведет к краху запуска.

После установки библиотек MPI стандарта 2.0 (версия 1.0.3 пакета Mpich2) была проведена 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. Запуск параллельных заданий осуществляется командой mpiexec, которая не имеет указанных выше недостатков команды mpirun (библиотеки MPI-1).

Использование модифицированной авторами библиотеки MPI позволило **впервые** получить исполняемое задание для работы GAMESS-US в параллельной среде под управлением MPI. 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, модифицированного GAMESS-US, конфигурационных файлов и скриптов настройки, файлов данных в единый «контейнер», выступающий в роли исходящего задания распределенной среды. Запуск контейнера аналогичен описанному выше для прототипа.

Серия первичных запусков была проведена на ресурсном сайте грид ИПХФ РАН для сред gLite и Unicore (запуск задач через грид инфраструктуру). Дальнейшее тестирование было проведено на ресурсных узлах RDIG в рамках BO RGSTEST (узлы НИИЯФ МГУ, 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).

Ввиду того, что на ряде кластеров запрещено или ограничено использование скриптовых языков, были 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 бинарного файла. При этом входящая задача порождает единичный процесс, который распаковывает библиотеки и системные файлы, прикладной пакет, файлы данных, настраивает среду исполнения, запускает параллельные процессы GAMESS-US, собирает полученные результаты, удаляет «мусор» и отправляет выходные данные на пользовательский интерфейс грид среды.

Таким образом, разработан метод создания виртуальных перемещаемых «контейнеров», которые содержат: «персональные» копии системных файлов и библиотек, скрипты по настройке операционной системы, необходимые файловые «деревья», собственно приложение, файлы данных и т.п. Использование таких «контейнеров» позволяет проводить запуски сложных прикладных пакетов **без их предустановки** на узлы грид сетей.

В качестве примера использован классический квантово-химический пакет GAMESS-US, для которого создан работоспособный «виртуальный контейнер» для сред gLite и Unicore.

ЛИТЕРАТУРА:

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", М.: МГУ, 2010, т.11, № 1, с.42-49
2. В.М. Волохов, Д.А. Варламов, Н.Ф. Сурков, А.В. Пивушков Виртуальные вычислительные среды: использование на GRID полигонах // Вестник ЮУрГУ, серия «Математическое моделирование и программирование», 2009, № 17 (150), вып. 3, с.24-35.