ПОДХОДЫ К РАЗРАБОТКЕ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ДЛЯ ПРОВЕДЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ ФИЗИЧЕСКИХ ЭКСПЕРИМЕНТОВ

В.В. Березовский, А.И. Попов

Вычислительные науки занимают важное место в теоретической физике, проводя мост к экспериментальной и, в некоторых случаях, подменяя ее в решении сложных задач. Они влекут использование компьютерных расчетов и симуляций для решения физических задач, таких, где аналитическое решение затруднено или невозможно сегодня. Сегодня известно большое количество подходов и методик решения одних и тех же задач и, как следствие, существует множество программных средств, как для численного решения решения какой-либо частной задачи, так и несколько более универсальных программных пакетов решения задач в некоторой области, реализующих те или иные методики. Ряд задач требуют организации взаимодействия этих программных средств, как друг с другом, так и с базами данных и другими хранилищами информации. Это задачи многомасштабного моделирования процессов и структур, постановка численных экспериментов, визуализации результатов расчетов и т.д. Например, рассматривая задачу селективности цеолитных мембран, требуется одновременно работать на трех масштабных уровнях. Это континуальный масштаб, где, используя перколяционные модели, описывается протекание газовых потоков сквозь кристаллическую структуру мембраны. Параметры модели берутся на другом масштабном уровне из классических расчетов молекулярной динамики и моделирования адсорбции с помощью метода Монте-Карло. Потенциалы поля сил взаимодействия атомов и молекул, применяемые в этих микро-масштабных моделях, подбираются и верифицируются с использованием квантово-химических расчетов из первых принципов(ab-initio) на третьем масштабном уровне, например, пользуясь теорией функционала плотности. Каждый из выше приведенных методов реализован в одном, или чаще всего, нескольких программных пакетах, имеющих близкую сферу применения, но часто различные алгоритмы, подходы к решению задач и реализации. При проведении комплексного исследования, такого как в выше приведенном примере, необходимо организовать последовательную работу различных программных пакетов, так, чтобы результат работы одного, подавался в качестве входных данных второго. Следует заметить отсутствие стандартов на форматы входных и выходных файлов, и, как следствие, необходимость постоянной ручной работы для организации поставки данных от вывода одного пакета на ввод другого. В отдельных случаях это решается созданием дополнительного программного обеспечения (ПО) для конвертации форматов, или простого преобразования одних структур данных в другие, но, тем не менее, универсальных средств, применимых к произвольной задаче, на сегодняшний день не существует.

Ввиду ограничений объема статьи, авторы опускают обзор существующих и разрабатываемых сегодня систем построения и поддержки вычислительных потоков и остановятся на разрабатываемой в Поморском государственном университете среде для поддержки вычислительных экспериментов. В работе выделяются основные требования к среде и рассматриваются некоторые аспекты реализации основанной на грид технологии среды для постановки многошаговых вычислительных экспериментов.

Постановка задачи

Предлагаемая в данной работе система должна удовлетворять следующим важным принципам:

Интероперабельность вычислительного ПО. Главной целью разрабатываемой вычислительной системы является предоставление возможности использования существующего вычислительного ПО, как широко используемого, такого как GULP, DL_POLY, GROMACS, SIESTA, NWChem, и т.д., так и любые другие созданные для решения специфических задач утилиты и программы. Это поднимает вопросы по организации поставки необходимых данных с вывода одного вычислительного пакета на ввод другого, хранения, анализа и визуализации промежуточных и финальных результатов работы вычислительного ПО. Эти вопросы требуют разработки общего представления множества существенных параметров описывающих произвольные модели исследуемых физических систем. В данном случае «существенные» означают такое множество, которое содержит полную информацию о системе, необходимую для запуска любого вычислительного приложения. Требуется разработать формат универсального представления мгновенного(в отдельный момент времени) и локального(в отдельной точке пространства, возможно фазового) набора наблюдаемых физических величин, и связывание их в траектории. Решение этой задачи позволит реализовать интероперабельность существующего вычислительного ПО, тем самым подготовит их возможность интеграции в грид.

Распределенная информационная система. Необходимо обеспечить координированное использование вычислительных ресурсов, баз данных и хранилищ при отсутствии централизированного управления этими ресурсами. Следует учесть возможность свободного включения новых ресурсов и соответственно масштабируемость системы, а также, использование уже существующих ресурсов (например, кристаллографических баз данных). Решение этой задачи позволит создать распределенную информационно-вычислительную систему.

Локальная автономия. Отдельный ресурс — компонент системы, являясь ее фрагментом, является

также полноценным локальным ресурсом, возможно со своей отдельной политикой администрирования.

Независимость узлов. Узлы системы равноправны в получении услуг системы и независимы друг от друга. В любой отдельный момент времени, узел системы может выйти из ее состава — динамичность среды. С учетом этого, требуется обеспечить выполнение поставленных задач, и возможность восстановления после ошибок связанных с выходом узла из системы. Учет этих вопросов позволит включать в систему ресурсы, уже входящих в состав действующих организаций, и, возможность формирования виртуальных организаций, создаваемых для решения выделенных задач.

Непрерывность операций. Требуется обеспечить возможность непрерывного доступа к ресурсам независимо от их расположения и вне зависимости от операций, выполняемых на локальных узлах, что позволит не отрывать их от их оригинальной деятельности.

Качество обслуживания(QoS). Необходимо предоставить определенный уровень качества обслуживания, используя ресурсы системы таким образом, чтобы была возможность выдерживать требования к качеству обслуживания по различным параметрам.

Коллективный разум. Необходимо предоставить возможность разделения получаемых пользователями системы результатов, алгоритмов и методик вычислительных экспериментов и обработки. Эта возможность должна стимулировать накопление, совершенствование и публикацию знаний. Следует также организовать возможность совместной работы многих участников в проведении вычислительных экспериментов и групповой разработки программного обеспечения и рабочих потоков(workflow).

Безопасность. Кроме обычных вопросов информационной безопасности, таких как, предоставление единой точки аутентификации и авторизации, цифровой подписи и шифрования данных, здесь возникают еще несколько. Вычислительное ПО обычно не свободное. Даже если оно не коммерческое, чаще всего оно распространяется под академической лицензией. Любая лицензия налагает какие-либо ограничения на использование идущего с ней ПО. Таким образом, при формировании рабочих потоков, необходимо учитывать, как то, какими лицензионными соглашениями связан пользователь, так и то, как лицензия установленного вычислительного пакета позволяет использовать результаты полученные с его помощью. Второй вопрос заключается в формировании списка цитирования. При проведении вычислительного эксперимента параметры эксперимента, его методика и последовательность действий, в основном, берутся из чьей-либо предыдущей работы, с, возможно, небольшими изменениями. Таким образом, необходимо вести учет, какая работа и где, была использована при проведении эксперимента. Также необходимо, разработать механизм, препятствующий возможности отказа от факта использования чужого интеллектуального труда в своей работе.

Повсеместные вычисления. Следует разработать интерфейс к системе поддерживающий принципы повсеместных вычислений. Устройство, с помощью которого пользователь работает с системой (персональный компьютер, мобильное устройство, браузер) является кэшем системы, отображающим ведущуюся именно в данный момент работу. Пользователь может произвольно менять свою точку входа, т.е. географическое положение, компьютер и приложение посредством которого он подключается к системе. Необходимо разработать интерфейс позволяющий реализовать аналог полнофункционального рабочего стола, который доступен с любого узла подключенного к интернет.

Интеграция вычислительного ПО

Многочисленные существующие на данный момент программные средства в вычислительных науках плохо скоординированы, и не являются приспособленными для работе в грид среде (grid-enabled). На сегодняшний день, это лишь разрозненные попытки вынести тот, или иной программный пакет в грид среду, зачастую, или закрытые, в рамках коммерческих систем, или, в результате частичного или полного изменения программного кода. Другой возможный подход - это инкапсуляция существующего ПО в грид сервис. Это даст возможность организовывать рабочие потоки из множества интероперабельных грид сервисов.

Рассматривая поставленные задачи с точки зрения открытой архитектуры грид сервисов (OGSA)[5], когда система строится на основе взаимодействия набора независимых или слабо связанных сервисов, вычислительное ПО, в услугах которого мы нуждаемся для проведения каждой ступени численного эксперимента, следует рассматривать как сервис. Этот сервис будет являться обобщением(виртуализацией) такого ресурса как отдельное приложение. Для того, чтобы отойти от вопросов конкретной инсталляции ПО на вычислительном ресурсе (персональный компьютер, кластер, грид инфраструктура), необходимо включить в этот сервис также вычислитель. Таким образом, отдельный сервис это представление в обобщенной форме такого ресурса как отдельный вычислитель+отдельная программа. Такое представление позволит нам абстрагироваться от таких непринципиальных различий ресурсов как способ запуска программы(постановка задачи в очередь), размещение входных и выходных файлов, и т.д. Пользуясь пяти-уровневым представлением протоколов грид [6], такой ресурс находится на базовом уровне (уровне фабрикатов) и, для возможности предоставления своих услуг другим, должен иметь свое представление на уровне ресурсов.

Таким образом, следует разработать промежуточное ПО уровня фабрикатов, являющееся объектом, который инкапсулирует в себя сложный ресурс вида вычислитель+программа. Ввиду того, что функциональность этого промежуточного ПО заключается в получении задания, запуск программы, ожидания завершения, передаче результатов и т.д., не принципиальна к различиям ресурсов (различие программ и

различие вычислителей), следует абстрагироваться от них. Необходимо отделить эту ключевую функциональность от непосредственных действий по запуску отдельной программы на отдельном вычислителе. Следует формализовать эти непосредственные действия, с помощью использования шаблонов, и языка преобразования, например, используя XML для шаблона и XSLT для его преобразования на конкретной платформе.

Также следует разработать промежуточное ПО для представления объекта уровня фабрикатов на уровне ресурсов. Оно должно позволить виртуализировать ресурс, предоставить способы управления им и его обнаружение. Здесь происходит инициация ресурса, его мониторинг и учет его использования, согласование политик безопасности использования ресурса. Сервис строящийся на основе этого промежуточного ПО уровня ресурсов должен иметь жесткую семантику и строго специфицированный интерфейс. Также. как и для фабриката, здесь можно отделить основную функциональность от специфичной для отдельного ресурса функциональности, однообразно для всех ресурсов конкретизируя взаимодействие с отдельным фабрикатом. Взаимодействие промежуточного ПО уровня фабрикатов и уровня ресурсов осуществляется посредством стандартных и открытых интернет протоколов

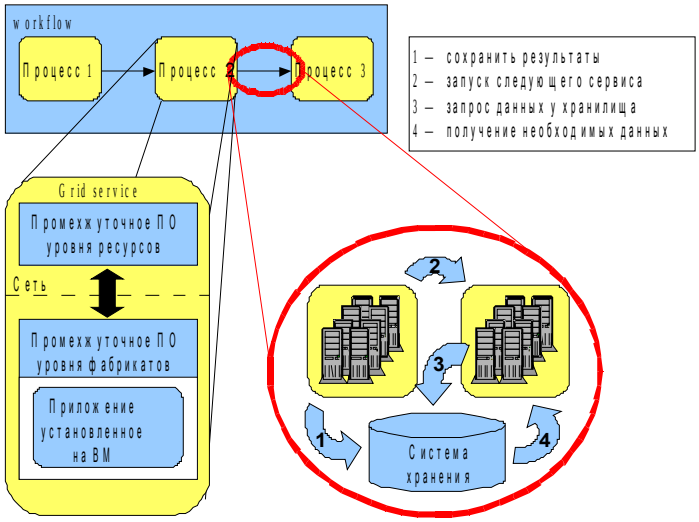

Рис. 1

Таким образом, каждая отдельная программа, установленная на некоторой вычислительной системе (компьютер, высокопроизводительная система) – отдельный ресурс (вычислитель+программа) который будет представлен парой взаимодействующих через сеть процессов уровней фабрикатов и ресурсов.(рис.1). Оба процесса реализованы двумя общими для всех ресурсов промежуточными приложениями. И представление этой программы на уровне ресурсов, является грид-сервисом предоставляющим стандартизированные услуги. Эта концепция хорошо укладывается в рамки спецификации WSRF когда полученный, описанным выше образом, грид-сервис будет представлен в виде WS-Resource с поддержкой состояний и своим жизненным циклом. Для обеспечения интероперабельности вычислительного ПО, необходимо организовать взаимодействие этих сервисов. Как отмечалось раннее, для передачи результатов работы одной программы на вход другой, необходимо производить преобразования форматов данных и файлов. Организация преобразования каждый-каждый трудоемка и имеет плохую масштабируемость, поэтому, предлагается разработать промежуточный формат удовлетворяющий форматам всех включенных в систему программ, это позволит, реализовав преобразование данных из формата данных программы в промежуточный и обратно, организовывать многоступенчатое выполнение задач. Преобразование в и из промежуточного формата можно также попытаться формализировать, тем самым увеличив универсальность системы. Обеспечение взаимодействия сервисов следует организовать тем же образом — через посредника. В данном случае посредником будет являться система хранения. Следует разработать промежуточное ПО являющееся интерфейсом к хранилищу, где будут храниться результаты работы программ в промежуточном формате. Этот сервис должен иметь единый с вычислительными сервисами интерфейс. Таким образом, интероперабельность вычислительных сервисов будет обеспечена тем, что один вычислительный сервис сохранит результат своей работы, в общем, для взаимодействующих сервисов, хранилище, а другой затем возьмет из него необходимые ему данные. Такой подход позволит решить не только задачу интероперабельности вычислительного ПО, но и позволит, благодаря уже существующим наработкам в грид технологии, решить и задачи создания распределенной системы, локальной автономии, независимости узлов , непрерывности операций и прозрачности расположения.

Графическое представление рабочих потоков.

Желательно, чтобы система управления рабочими потоками в среде грид (Grid Workflow Management

System, GWMS) предоставляла своим пользователям инструмент для наглядного описания последовательности работ с ветвлениями, итерациями, параллелизмом, потоками данных, передаваемых от одного действия к другому, предопределенными действиями и указанием исполнителей действий. Чтобы позволить пользователям сосредоточиться на логике процесса и отвлечься от технических деталей выполнения работ, в подобных инструментах применяются различные графические нотации. В качестве таких абстрактных графических языков (Abstract Grid Workflow Language, A-GWL) в разных проектах применяются сети Петри, Business Process Modeling Notation (BPMN), диаграммы активностей (Activity Diagram, AD) из языка UML (UML AD), а также разрабатываемые вновь нотации [3, 4, 5].

В данной работе за основу графического языка описания рабочих потоков предлагается принять UML AD [6]. Прежде всего, язык UML соответствует предъявленным выше требованиям, широко изучается, применяется и поддерживается, а UML AD очень похожи на другие AD (например, привычные схемы алгоритмов и программ), благодаря чему для большинства пользователей освоение соответствующего инструмента займет минимальное время. Кроме того, к процессу, описанному при помощи AD, применим разработанный за последние десятилетия обширный аппарат методов анализа параллельных граф-схем алгоритмов. В частности, возникают дополнительные возможности для проведения верификации рабочего потока на логическом уровне [7]. Важным является и наличие некоторых наработок в области использования UML AD для моделирования и спецификации грид-приложений, например [8]. Однако основным аргументом в пользу UML AD является то, что их применение соответствует объектно-ориентированному подходу к проектированию, и при построении диаграммы акцент делается на определении исполнителей работ. В объектно-ориентированном проектировании исполнителями являются экземпляры классов (объекты), а в случае грид-системы речь может идти о сервисах. UML AD можно связать с диаграммой классов, а диаграмму классов – с множеством доступных сервисов и онтологией предметной области. Это обеспечит перспективу для дальнейших исследований в направлении автоматизации поиска наиболее подходящих сервисов для выполнения работ и, соответственно, нисходящего проектирования рабочего потока.

Для указания распределения ролей область, занимаемая UML AD, разбивается на дорожки (swimlines). На дорожке отображаются только те работы, за которые отвечает конкретный объект. Как уже сказано, в качестве объектов следует рассматривать сервисы. Но задачи, для которых предназначена разрабатываемая система, таковы, что количество исполнителей зачастую совпадает с количеством работ, т.е. при стандартном в UML использовании дорожек на каждой из них будет размещаться только одна активность. Большое количество дорожек на диаграмме ухудшит ее наглядность (это важно при проектировании, во время презентации результатов научному сообществу и т.д.). Поэтому дорожки могут использоваться некоторым иным образом, чем в UML. Например, на первую дорожку можно помещать только те работы, для которых проектировщик определил и программу, и вычислителя; на вторую — работы, для которых проектировщик указал только программу, а выбор вычислителя предоставил системе; на третью — работы, поиск сервиса для которых полностью возложен на систему. Поэтому инструмент визуального проектирования должен предоставлять возможность выбора нотации AD: с отображением дорожек, как в UML; с более компактным отображением сервисов (например, непосредственно на активностях); без указания сервисов и др.

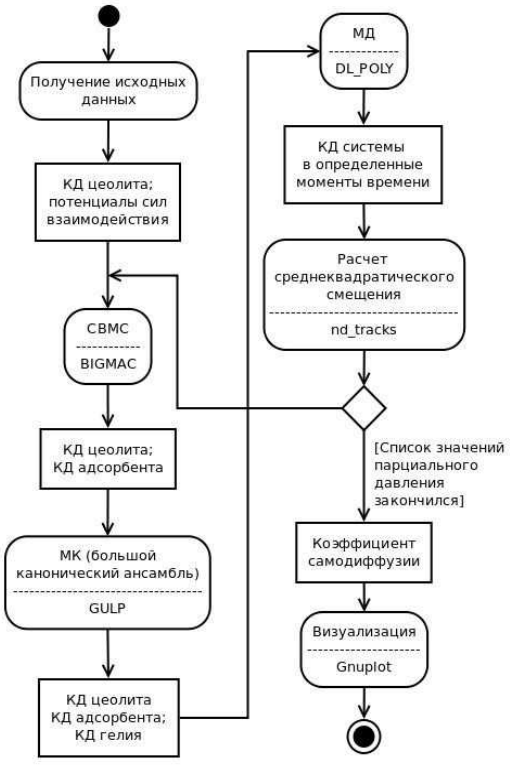

Рис. 2

В качестве примера на рис.2 приведена AD для процесса моделирования диффузии гелия в составе смеси с сильно-адсорбирующимся компонентом сквозь поры цеолитной мембраны.(Сокращения: КД – кристаллографические данные, CBMC – configurational-bias Monte Carlo, МК – метод Монте-Карло, МД – молекулярная динамика).

Трансляция AD в текстовое описание последовательности вызовов работ (Concrete Grid Workflow Language, C-GWL) является обязательным функциональным требованием к инструменту визуального проектирования рабочих потоков. В настоящее время разработаны языки как для описания требований к отдельным работам: Grid Job Definition Language (GJDL) [5], Job Submission Description Language (JSDL) [9] и др., – так и для представления последовательности работ: Web Services Flow Language (WSFL), Business Process Execution Language for Web Services (BPEL4WS) [5] и др.

Работа выполнена при поддержке РФФИ, проект 11-07-

06804-моб_г

ЛИТЕРАТУРА:

1. I. Foster, et al, The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
2. I. Foster, et al, The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. of High Performance Computing Applications 15(3), 200.(2001)
3. HOME – A-Ware. URL: http://www.a-ware-project.eu/aware.html
4. J. Yu, et al, A taxonomy of workflow management systems for grid computing. Journal of Grid Computing, 3(3),. 171.(2005)
5. The Grid Workflow Forum. URL: http://gridworkflow.org/snips/gridworkflow/space/start
6. UML 2.3. URL: http://www.omg.org/spec/UML/2.3
7. Закревский А.Д. Параллельные алгоритмы логического управления. Минск.: Ин-т техн. кибернетики АН БССР, 1999
8. T. Fahringer, et al, Teuta. Tool Support for Performance Modeling of Distributed and Parallel Applications. International Conference on Computational Science, Tools for Program Development and Analysis in Computational Science, Krakow, Poland, June 2004.
9. A. Anjomshoaa, et al, Job Submission Description Language (JSDL) Specification, Version 1.0. Global Grid Forum.(2005)