Высокоуровневые средства разработки Grid-приложений для инфраструктуры EGEE*

О.В. Сухорослов

В статье рассматривается набор программных средств, облегчающих разработку Grid-приложений в рамках инфраструктуры EGEE. Данные средства ориентированы на решение типовых задач, возникающих при создании подобных приложений. За счет скрытия от программиста деталей взаимодействия с промежуточным ПО и предоставления готовых каркасов для организации распределенных вычислений, рассматриваемые средства позволяют уменьшить время и затраты на разработку Grid-приложений.

1. Введение

EGEE (Enabling Grids for E-sciencE) [1] является крупнейшей в мире Grid-инфраструктурой, ориентированной на поддержку исследований в различных областях науки. В настоящее время в EGEE входит свыше 140 организаций и около 300 ресурсных центров из 50 стран. Российские организации-участники EGEE поддерживают национальный сегмент данной инфраструктуры в рамках консорциума RDIG (Russian Data Intensive Grid).

Несмотря на внушительный объем вычислительных ресурсов (около 80 тысяч процессорных ядер), круг сегодняшних пользователей и приложений EGEE относительно узок. Обусловлено это главным образом сложностью освоения и использования промежуточного ПО Grid. Как следствие, наблюдается недостаток Grid-приложений, позволяющих пользователю сформулировать интересующую его задачу через привычный проблемно-ориентированный интерфейс и преобразующих данную задачу в вычислительные задания, запускаемые в Grid (Рис. 1).

Реализация подобных приложений затруднена отсутствием высокоуровневых средств разработки, скрывающих от программиста низкоуровневые детали взаимодействия с Grid и позволяющих, тем самым, уменьшить время и затраты на разработку приложения. Кроме того, отсутствуют готовые реализации в Grid типовых схем распределенных вычислений, таких как “управляющий-рабочие”, позволяющие использовать их как каркас для создания новых приложений. Таким образом, плохо развит изображенный на Рис. 1 уровень прикладного инструментария.

В статье рассматривается ряд программных средств, нацеленных на преодоление описанного барьера между разработчиками приложений и промежуточным ПО Grid. Библиотека jLite предоставляет высокоуровневый программный интерфейс для разработки платформонезависимых Grid-приложений на языке Java. Интерфейс командной строки jLite CLI позволяет осуществлять запуск Grid-заданий на любой платформе с поддержкой Java. Инструментарий MaWo предоставляет универсальный каркас для реализации вычислений по схеме “управляющий-рабочие” в Grid.

2. Интерфейс прикладного программирования jLite

Базовым программным обеспечением Grid-инфраструктуры EGEE является промежуточное ПО gLite [2]. Несмотря на то, что в состав gLite входят интерфейсы прикладного программирования (API), пользоваться данными интерфейсами сложно по ряду причин. Так, существующие API для языка Java распределены по множеству библиотек со сложными внешними зависимостями. По большей части данные API предоставляют доступ к низкоуровневым операциям базовых сервисов Grid. Наконец, отсутствует подробная документация по использованию этих библиотек. Доступные примеры использования часто подразумевают наличие стандартного пользовательского окружения gLite (т.н. User Interface) и операционной системы Scientific

* Работа выполнена при поддержке фонда РФФИ (грант № 08-07-00430-а) и Президиума РАН (программа фундаментальных исследований 15П).

Linux. Это затрудняет разработку платформонезависимых Grid-приложений, которые могут быть размещены на произвольном сервере или рабочей станции пользователя при наличии среды выполнения Java.

Рис. 1. Уровни программного обеспечения Grid.

Библиотека jLite [3-4] решает описанные проблемы путем реализации высокоуровневого интерфейса прикладного программирования, содержащего операции аналогичные командам пользовательского окружения gLite. Данный интерфейс скрывает от программиста сложность низлежащего промежуточного ПО и его конфигурации. Библиотека реализована на базе существующих API gLite и внешних компонентов, таких как Apache Axis, Java CoG Kit, Condor ClassAd и т.д. Дистрибутив jLite включает все внешние зависимости, что упрощает его установку. Кроме того, для использования jLite не требуется инсталляция пользовательского окружения gLite. Это позволяет использовать библиотеку на любой платформе с поддержкой Java, включая Windows, что открывает новые возможности для разработчиков Grid-приложений. Библиотека jLite доступна для загрузки и использования на условиях лицензии Apache License 2.0.

Текущая реализация jLite API (версия 0.1, февраль 2008 г.) включает все базовые операции, связанные с запуском заданий в Grid-инфраструктуре EGEE:

- Создание временного прокси-сертификата пользователя с помощью сервиса виртуальных организаций VOMS;
- Делегация прокси-сертификата сервису управления заданиями WMS;
- Поиск вычислительных ресурсов, удовлетворяющих требованиям Grid-задания;
- Собственно запуск задания, включающий загрузку входных данных на сервис WMS (поддерживаются следующие типы заданий: normal, collection, parametric);
- Мониторинг статуса выполнения Grid-задания;
- Загрузка результатов выполнения задания на локальную машину.

Предоставляя простой в использовании и переносимый программный интерфейс, jLite тем самым облегчает создание разнообразных Grid-приложений на базе инфраструктуры EGEE, таких как проблемно-ориентированные оболочки, порталы или Grid-сервисы. Среди похожих проектов стоит отметить инициативу Simple API for Grid Applications (SAGA) [5], нацеленную на создание стандартного API для различных платформ Grid и языков программирования по образу стандарта MPI. В настоящее время реализация данного API для платформы gLite еще не

завершена. В отличие от SAGA, библиотека jLite ориентирована на одну платформу Grid и один язык программирования.

В данный момент jLite проходит апробацию в нескольких приложениях, два из которых описаны ниже. Дальнейшее развитие библиотеки планируется проводить по двум направлениям. Во-первых, это расширение имеющейся функциональности, например – запуск workflow-заданий (тип DAG). Во-вторых, это добавление новой функциональности, такой как доступ к информационным сервисам и сервисам хранения данных.

3. Интерфейс командной строки jLite CLI

jLite CLI представляет собой платформонезависимый интерфейс командной строки, реализованный на основе описанной выше библиотеки jLite и входящий в состав ее дистрибутива [4]. Поскольку интерфейс jLite API близок по функциональности к командам пользовательского окружения gLite, то реализация соответствующего интерфейса командной строки представляется естественной. Полезность же данного приложения вытекает из недостатков существующего интерфейса командной строки gLite.

Для запуска заданий в инфраструктуре EGEE пользователю необходимо иметь удаленный доступ к специальной машине с установленным пользовательским окружением gLite, откуда собственно и выполняются различные команды по запуску и управлению Grid-заданиями. Установка пользовательского окружения на машину пользователя на практике связана с большим количеством технических трудностей, поскольку требует наличия операционной системы Scientific Linux и значительных усилий по установке, настройке и сопровождению.

Предпринималось несколько попыток перенести пользовательское окружение gLite на другие операционные системы, среди которых наиболее востребованной является Windows. Один из подходов заключается в создании виртуальной машины с пользовательским окружением [6]. Второй подход состоит в прямой компиляции пользовательского окружения под ОС Windows [7]. Оба подхода имеют свои недостатки: в первом случае – дополнительная нагрузка на систему, связанная с виртуализацией, во втором – многочисленные проблемы с компиляцией кода, написанного под ОС Linux.

В отличие от стандартного пользовательского окружения, jLite CLI может быть установлен на любой распространенной операционной системе, при наличии среды выполнения Java. Кроме того, данный интерфейс может быть легко перенесен с компьютера на компьютер вместе со всеми настройками. Это открывает новые возможности как для пользователей Grid, так и разработчиков Grid-приложений. Справедливости ради, отметим, что jLite CLI реализует только базовую часть функциональности стандартного пользовательского окружения. Тем не менее, этой функциональности достаточно для многих задач.

В состав текущей реализации jLite CLI (версия 0.1, февраль 2008 г.) входят следующие команды, оформленные в виде bat- и shell-скриптов:

- **proxy-init** – создает прокси-сертификат с атрибутами VOMS (аналог команды voms-proxy-init пользовательского окружения gLite);
- **proxy-info** – отображает информацию о существующем прокси-сертификате (аналог команды voms-proxy-info);
- **proxy-delegate** – делегирует прокси-сертификат сервису управления заданиями (аналог команды glite-wms-job-delegate-proxy)
- **proxy-destroy** – уничтожает прокси-сертификат (аналог команды voms-proxy-destroy);
- **job-match** – отображает список вычислительных ресурсов, удовлетворяющих требованиям задания (аналог команды glite-wms-job-list-match);
- **job-submit** – осуществляет запуск задания (аналог команды glite-wms-job-submit);
- **job-status** – отображает информацию о статусе задания (аналог команды glite-wms-job-status);
- **job-output** – загружает выходные файлы задания (аналог команды glite-wms-job-output);
- **job-cancel** – производит отмену задания (аналог команды glite-wms-job-cancel).

Перспективной альтернативой интерфейсу командной строки Grid является среда gEclipse [8] с развитым графическим интерфейсом на основе платформы Eclipse. Данная среда также реализована на языке Java и поддерживает работу на различных операционных системах.

4. Инструментарий для реализации вычислений по схеме “управляющий-рабочие”

Значительная часть решаемых в Grid задач допускает декомпозицию на множество независимых подзадач. Это позволяет реализовать параллельное решение задачи путем распределения подзадач между вычислительными узлами Grid. При этом эффективность и время решения задачи в Grid во многом определяется выбранной стратегией распределения подзадач.

Простейшая стратегия, заключающаяся в статическом назначении подзадач отдельным Grid-заданиям, часто приводит к неравномерной загрузке узлов и плохо контролируемому общему времени вычислений. Связано это может быть как с неравномерностью размера самих подзадач и невозможностью априори оценить этот размер, так и с неизбежной гетерогенностью и ненадежностью узлов Grid.

Более эффективной на практике является стратегия динамического распределения подзадач в соответствии с известной схемой "управляющий-рабочие". В рамках Grid данная схема может быть реализована следующим образом (Рис. 2). Запускается некоторое число Grid-заданий, которые представляют собой рабочие процессы. Данные процессы связываются по сети с управляющим процессом, который выполняется на выделенной машине вне Grid. Управляющий процесс хранит список задач (подзадач исходной задачи) и распределяет их между рабочими процессами.

Для балансировки нагрузки между рабочими процессами может использоваться простой и в то же время эффективный механизм распределения задач, заключающийся в активном "вытаскивании" задач рабочими процессами. В этом случае управляющий процесс играет пассивную роль, обрабатывая приходящие запросы рабочих процессов. При регистрации рабочий процесс запрашивает у управляющего процесса задачу, выполняет ее, передает полученный результат управляющему процессу, запрашивает новую задачу и так далее, до тех пор, пока список невыполненных задач не будет исчерпан.

Поскольку узлы Grid являются ненадежными, а Grid-задания имеют ограниченное время выполнения, то рабочие процессы могут непредсказуемым образом выходить из процесса вычислений. Таким образом, также необходим механизм обнаружения и восстановления после отказов рабочих процессов.

Очевидно, что механизмы запуска рабочих процессов в Grid, их взаимодействия с управляющим процессом, распределения задач и обработки отказов в рамках описанной схемы являются в значительной мере общими и слабо зависят от решаемой задачи. Это позволяет создать универсальный каркас (framework) для разработки подобных Grid-приложений, содержащий готовые реализации данных механизмов и допускающий их настройку и расширение.

Одной из первых реализаций схемы “управляющий-рабочие” для Grid явился пакет Condor MW [9-10], предоставляющий каркас для разработки приложений на языке C++ в рамках системы Condor. Существует несколько подобных разработок, поддерживающих запуск рабочих процессов в инфраструктуре EGEE. Пакет DIANE (DIstributed ANalysis Environment) [11] реализует схему “управляющий-рабочие” на языке Python. Для запуска рабочих процессов используется пакет Ganga, поддерживающий запуск заданий в EGEE при наличии пользовательского окружения gLite. Инструментарий ProActive [12] содержит обширный набор средств реализации параллельных и распределенных вычислений на базе платформы Java, в том числе каркас для реализации схемы “управляющий-рабочие”. ProActive поддерживает запуск рабочих процессов в EGEE через машину с пользовательским окружением gLite.

Разрабатываемый автором инструментарий MaWo реализует каркас для организации вычислений по схеме “управляющий-рабочие” в инфраструктуре EGEE. Инструментарий реализован на языке Java. В отличие от DIANE и ProActive, MaWo не требует доступа к пользовательскому окружению gLite. Для запуска рабочих процессов в Grid используется описанная ранее библиотека jLite (Рис. 2).

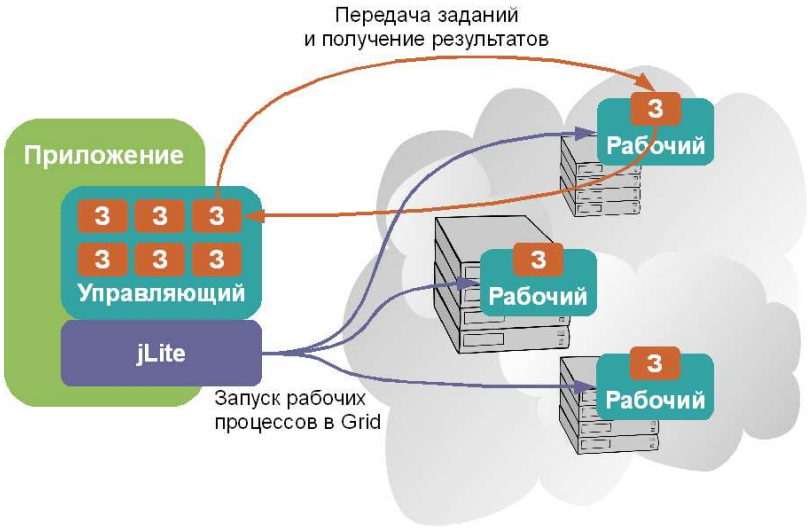

Рис. 2. Схема работы MaWo.

В рамках текущей реализации MaWo программист описывает отдельную задачу (подзадачу исходной задачи) в виде Java-класса, хранящего начальные данные задачи и реализующего алгоритм ее решения. В качестве вычислительных модулей решения задач могут также использоваться произвольные исполняемые файлы и программы на других языках.

Кроме того, программисту требуется подготовить управляющий компонент, реализующий логику разбиения исходной задачи на подзадачи и управляющий процессом вычислений. Данный компонент, встраиваемый в управляющий процесс, вызывается в начале вычислений для генерации начального списка задач. Кроме того, управляющий компонент вызывается во время получения результата каждой задачи, тем самым, позволяя сохранить полученный результат, а также сгенерировать при необходимости новые задачи и поместить их в очередь управляющего процесса. Также управляющий компонент используется для определения момента остановки вычислений.

При запуске вычислений разработчик передает MaWo управляющий компонент и требуемое число рабочих процессов. После этого на локальной машине запускается сервер с управляющим процессом и происходит запуск в Grid соответствующего числа заданий, содержащих код рабочего процесса и вычислительного модуля задачи. Пользователь может просматривать информацию о ходе вычислений через Web-интерфейс, встроенный в сервер с управляющим процессом.

Сразу после запуска на узле Grid рабочий процесс связывается по сети с управляющим процессом, передавая ему информацию о платформе и характеристиках узла Grid. Данная информация может быть в последствии использована для реализации более сложных стратегий распределения задач. После регистрации рабочий процесс начинает запрашивать у управляющего процесса задачи и выполнять их в соответствии с описанной ранее простой стратегией.

Для контроля процесса вычислений и обнаружения отказов рабочие процессы периодически отправляют управляющему процессу служебные сообщения с информацией о своем состоянии. Если в течение определенного промежутка времени от рабочего процесса не поступало служебных сообщений, то данный процесс помечается как отказавший и выполняемые им задачи помещаются обратно в очередь невыполненных задач. Также при отказе рабочего процесса автоматически происходит запуск нового Grid-задания с тем, чтобы поддерживать заданное в начале вычислений число рабочих процессов. Исключением является ситуация, когда вычисления близки к завершению, и есть свободные рабочие процессы, - в этом случае новое задание не запускается.

Отметим, что в рассмотренной реализации все сетевые вызовы направлены от рабочих процессов к управляющему процессу. Выбор подобного подхода обусловлен тем, что вычислительные узлы Grid, как правило, находятся за межсетевым экраном, запрещающим входящие

соединения. Передача команд от управляющего к рабочим процессам осуществляется внутри ответов на приходящие служебные сообщения. Для взаимодействия между процессами используется технология промежуточного ПО Ice [13-14].

В рамках тестирования инструментария MaWo было проведено несколько вычислительных экспериментов в Grid, состоящих в выполнении большого количества синтетических вычислительных задач с изменяемыми параметрами. Предварительные результаты экспериментов подтверждают применимость описанного подхода и реализованного инструментария для организации подобного рода вычислений. В дальнейшем планируется провести более детальный анализ результатов экспериментов с вычислением таких показателей, как ускорение и эффективность. Помимо этого планируется реализовать решение нескольких содержательных задач, а также провести детальное сравнение функциональности MaWo и аналогичных разработок.

Литература

1. EGEE: [http://www.eu-egee.org/], 11.02.2009.
2. gLite: [http://glite.web.cern.ch/glite/], 11.02.2009.
3. O.V. Sukhoroslov. jLite: A Lightweight Java API for gLite // Proceedings of the 3rd International Conference "Distributed Computing and Grid-technologies in Science and Education", GRID'2008, Dubna, Russia, 30 June - 4 July, 2008.
4. jLite: [http://jlite.googlecode.com/], 11.02.2009.
5. Simple API for Grid Applications: [http://saga.cct.lsu.edu/], 11.02.2009.
6. GILDA VM UI: [https://gilda.ct.infn.it/VirtualServices.html], 11.02.2009.
7. Russo, D. and Scibilia, F. 2007. Grid2Win: Porting gLite to Windows Based Platforms // Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure For Collaborative Enterprises (June 18 - 20, 2007), pp. 300-301.
8. gEclipse: [http://www.geclipse.eu/], 11.02.2009.
9. Condor MW: [http://www.cs.wisc.edu/condor/mw/], 11.02.2009.
10. Goux, J.-P., Kulkarni, S., Linderoth, J., and Yoder, M. An enabling framework for master-worker applications on the computational grid // Proceedings of the Ninth International Symposium on High Performance Distributed Computing (Pittsburgh, Pennsylvania, Aug. 2000), pp. 43-50.
11. DIANE: [http://cern.ch/diane/], 11.02.2009.
12. ProActive: [http://proactive.inria.fr/], 11.02.2009.
13. ZeroC Ice: [http://www.zeroc.com/], 11.02.2009.
14. Сухорослов О.В. Промежуточное программное обеспечение Ice // Проблемы вычислений в распределенной среде / Под ред. А.П. Афанасьева. Труды ИСА РАН. Т. 32. - М.: Издательство ЛКИ, 2008. – 288 с. (с. 33-67)