gLite Введение в GRID-технологии на основе Scientific Linux.

«Грид»: что,

ЧАСТЬ 1 Слово «грид» нынче в моде, но лишь немногие в курсе, что оно означает. И уж совсем немногие знают, что оно имеет отношение к Linux. **Андрей Е. Шевель** поможет вам попасть в число этих немногих.

Наш эксперт

Андрей Е. Шевель
Зав. отделом вычислительных систем в Отделении физики высоких энергий Петербургского института ядерной физики Российской академии наук. Основной научный интерес – распределенные вычислительные системы.

На данном уроке мы сделаем краткий обзор простейших возможностей, особенностей и терминологии распределенных вычислительных систем с архитектурой «грид» (grid) на основе комплекта пакетов промежуточного ПО *gLite*. Он был сформирован для решения задач распределенной обработки данных в экспериментах на ускорителе элементарных частиц Large Hadron Collider (LHC) – Большой Адронный Коллайдер (БАК) в Женеве (Швейцария). Несмотря на то, что составители комплекта имели в виду в основном научную обработку данных, *gLite* представляет собой весьма общий набор инструментов для организации распределенной обработки в любой отрасли.

Немного теории

В начале 90-х несколько ученых, работавших в области вычислительной техники (среди них было несколько лидеров современного консорциума Globus), пришли к мысли, что для проведения больших вычислений в различных областях знаний (моделирование мировой экономики, климата планеты или отдельного региона, рождения звезд, ядерных взрывов) недостаточно традиционного на тот момент подхода: проведения расчетов в одном/двух местах с использованием либо больших монолитных вычислителей, либо кластеров, нередко – специально настроенных под конкретный класс задач. Так, кластер, настроенный для моделирования океана, лишь с трудом мог бы быть использован, скажем, в экономике.

Имелся целый ряд причин, которые препятствовали использованию одного и того же кластера в различных отраслях и даже разными организациями: например, применялись ПО и оборудование, хорошо подходившее для одних задач и не слишком – для других. Имели место и административные, и финансовые барьеры: различные кластеры приобретаются и поддерживаются из разных фондов. При использовании удаленных кластеров должны были решаться проблемы сетевой безопасности, причем таким образом, чтобы пользователю не приходилось постоянно вводить пароль (иначе затрудняется выполнение автоматизированных скриптов)... Перечисление можно продолжить.

С другой стороны, для больших задач, как правило, недостаточно одного кластера. Иными словами, возникает необходимость интегрировать ресурсы многих кластеров, которые, вообще говоря, распределены по странам и континентам, организациям и компаниям, зданиям и офисам. Более того, кластеры имеют различное «подчинение», различные источники финансирования, различный стиль работы администраторов. Как объединить столь непохожие стихии?

Во-первых, понадобилось разработать программное обеспечение, которое дало технические предпосылки для объединения разнородных вычислительных установок для нескольких классов задач:

- Больших вычислительных проблем, где требуется главным образом процессорное время.
- Больших задач, где требуется использовать огромный объем данных, много петабайтов.
- Больших задач, в решение которых вовлечено значительное число специалистов, проживающих в разных городах, странах и континентах.

Естественно, что имеются проблемы, которые совмещают все перечисленные требования.

Процесс осознания полезности интегрирования распределенных компьютерных ресурсов начался в явном виде с выхода сборника статей нескольких десятков авторов, объединенных в книгу «The Grid: blueprint for a new computing infrastructure» [Сеть: генеральный план новой компьютерной инфраструктуры], которая была опубликована под редакцией Яна Фостера [Ian Foster] и Карла Кесельмана [Carl Kesselman] в 1999 году (см. www.amazon.com/Grid-Blueprint-New-Computing-Infrastructure/dp/1558604758). В книге были последовательно описаны методы и средства вычислений, используемые в различных отраслях научной и инженерной деятельности: в физике, химии, медицине, геологии, биологии, космических исследованиях и т.д. Даже беглый взгляд обнаруживал, что в методах самых разных областей знания имеется масса общего. Были выделены основные классы вычислений.

- Распределенные вычисления.
- Высокопроизводительные вычисления.
- Вычисления по требованию, т.е. такие, которые не требуют постоянного доступа к вычислительным ресурсам, а лишь по мере необходимости.
- Обработка больших объемов данных.
- Коллаборационные вычисления (где участвует целая армия специалистов, рассеянных по всему свету).

Естественно, сразу возникли мысли о том, как организовать компьютерную инфраструктуру таким образом, чтобы можно было интегрировать необходимые ресурсы почти автоматически. Иными словами, использовать их так, как мы сейчас используем электрическую сеть, т.е. создать некоторый вычислительный grid (грид). Здесь уместно заметить, что этот термин есть литературная метафора, а не строго определенное понятие.

Для реализации такого амбициозного подхода понадобилось решение нескольких задач:

откуда и зачем?

» Безопасность, которая включает аутентификацию пользователя (имя пользователя, его идентификатор, полномочия, принадлежность к конкретной группе пользователей – виртуальной организации).

» Универсальное программное окружение, которое позволяло бы определить свойства кластера и его узлов, не выполняя вход в систему – иными словами, специальный сервер, который бы автоматически собирал с кластеров информацию об их состоянии (доступен, недоступен) и архитектурных особенностях (число доступных узлов, состав узлов, прочее). Клиент на вашем компьютере может связаться со специальным сервером, «ресурсным брокером», который подберет кластер, наиболее подходящий для вашего задания. На первый взгляд, задача может показаться тривиальной, но если число кластеров составляет многие десятки, сотни или тысячи, то ответ уже не столь очевиден.

» Мониторинг ресурсов различных кластеров, линий связи в Интернете с тем, чтобы иметь представление о загруженности тех или иных ресурсов.

» Передача данных от кластера к кластеру. Средства обеспечения надежной передачи больших объемов информации за минимальное время и в каком-то смысле оптимальным образом. Создание копий (реплик) данных на нескольких кластерах.

» Наконец, администрирование больших групп пользователей – виртуальных организаций.

Инициатива стартовала примерно в 2000 году. Почти в одно время начались работы над несколькими проектами в разных организациях: исследовательских центрах, университетах. Среди первых, естественно, были те группы и организации, которые уже столкнулись с задачами, упомянутыми выше:

» интернациональные группы ученых (коллаборации) в области исследования космического пространства и исследования земли с космических аппаратов;

» колаборации исследователей в области биологии и медицины;

» коллаборации исследователей по изучению свойств материи на ускорителях заряженных частиц (экспериментальная ядерная физика и физика высоких энергий).

Все разработки такого типа (грид) выполнялись совместно, большим числом специалистов из самых разных отраслей знаний. Было создано большое число сайтов, посвященных грид-разработкам. Среди основных можно отметить www.globus.org – исторически первый сайт разработчиков грид-продуктов; www.ggf.org – свободное международное объединение разработчиков для обсуждения и продвижения перспективных средств распределенных вычислений. Среди русскоязычных ресурсов сайтов упомянем www.egee-rdig.ru и www.gridclub.ru. Читатель запросто сможет найти сотни других привлекательных сайтов на данную тему.

В течение нескольких лет в рамках ряда проектов были созданы десятки решений, которые стали называть middleware (промежуточное ПО, т.е. такие программы/библиотеки, которые служат интерфейсами между различными программными слоями). Среди крупнейших можно упомянуть проекты: Open Science Grid (OSG) – opensciencegrid.org, Enabling Grids for E-sciencE (EGEE) – public.eu-egee.org/intro, NorduGrid – www.nordugrid.org.

Пакеты-middleware

Почти все разработки в области грид-архитектур являются открытыми, а созданное ПО – свободно распространяемым, поэтому Linux, очевидно, является для них естественной платформой. Для этих целей мировое научное сообщество организовало специальный дистрибутив – Scientific Linux (www.scientificlinux.org). Основой для него на сегодняшний день служат исходные тексты Red Hat Enterprise Linux. В настоящее время Scientific Linux используется на многих тысячах компьютеров и сотнях кластеров в различных уголках планеты. Таким образом, надежность этой системы наравне с приемлемой функциональностью не вызывает сомнений. Российское зеркало сайта Scientific Linux вместе с переводом описаний доступно по адресу www.scientificlinux.ru.

Десятки различных разработок в области грид-систем вылились в естественный параллелизм в каких-то аспектах. Поэтому, когда конечный пользователь пытается применять относительно разрозненные программные пакеты, могут возникать нестыковки, затрудняющие использование модели «грид» в конкретном окружении.

В связи с этими обстоятельствами, крупные проекты составили свои комплекты пакетов промежуточного ПО с таким расчетом, чтобы конечный потребитель смог использовать модель вычислений «грид» не только в многолюдных коллаборациях, но и в относительно небольших коллективах. К наиболее известным комплектам промежуточного ПО относятся:

» *Advanced Resource Connector* (www.nordugrid.org/middleware) – продвинутый исследовательский коннектор из проекта NorduGrid;

» *Virtual Data Toolkit* (vdt.cs.wisc.edu//index.html) – виртуальный инструментарий для управления данными из проекта **OSG**;

» *gLite* (www.glite.org) – часть проекта **EGEE**.

Последний из них мы сейчас и рассмотрим более подробно.

WLCG

За данной аббревиатурой скрывается словосочетание Worldwide LHC Computing Grid – Всемирный вычислительный грид для Большого Адронного Коллайдера. Все пользователи WLCG/EGEE объединены в ряд групп – виртуальных организаций (virtual organization, VO). Каждая VO (и, следовательно, каждый член виртуальной организации) имеет доступ к некоторому набору компьютерных ресурсов, распределенных по планете. Любой пользователь, который планирует работать в рамках конкретной виртуальной организации, должен зарегистрироваться, т.е. внести свои персональные данные. В рамках регистрационной процедуры пользователь получает специальный электронный личный сертификат, который выдается центром сертификации – Certification Authority (CA). Сертификат представляет собой два небольших файла с закодированной информацией о пользователе, организации, выдавшей сертификат, сроке его действия и другими служебными данными (LXF93). Один из файлов содержит открытый ключ пользователя, второй – секрет-

ный (закрытый) ключ. Последний защищен паролем, который знает лишь его владелец.

После успешной регистрации пользователь может получить доступ к ресурсам WLCG/EGEE с использованием выданного ему ранее личного сертификата. Доступ к ресурсам VO осуществляется посредством инфраструктуры грид-безопасности – Grid Security Infrastructure (GSI). GSI позволяет опознать пользователя по его сертификату, обеспечив защищенный обмен данными через открытую компьютерную сеть (локальную или Интернет). Технически, GSI базируется на использовании Public Key Infrastructure (PKI) – инфраструктуры шифрования с открытым ключом с использованием стандарта X509.

Секретный ключ пользователя применяется для генерации другого, временного (действующего обычно несколько часов или десятков часов) сертификата, так называемого прокси (proxy certificate). Этот прокси используется в гриде для опознания пользователя на различных сервисах, а также для делегирования полномочий. Таким образом, единожды введя пароль для секретного ключа при создании прокси, пользователь может не делать этого в дальнейшем, поскольку инфраструктура GSI будет применять его прокси каждый раз, когда потребуется аутентификация на каком-то из сервисов грида. Этот процесс может происходить двумя путями:

» Специальное имя пользователя из прокси сравнивается со списком имен на хосте в файле *grid-mapfile*. Если специальное имя пользователя обнаруживается, то там же будет указано, какую учетную запись следует использовать на данном хосте.

» Второй метод основывается на использовании факта принадлежности к конкретной VO, механизме Virtual Organization Membership Service (VOMS) и сопутствующих служб.

Создание временного прокси выполняется командой

```
woms-proxy-init
```

Эта утилита попросит ввести пароль (вернее, парольную фразу), который был использован при получении сертификата. После инициализации прокси можно пробовать запускать задания.

Запуск заданий

Задание(я) формируется пользователем с помощью специального языка описания заданий – Job Description Language (JDL). Например

```
Executable = "/bin/date";
StdOutput  = "std.out";
StdError   = "std.err";
```

Естественно, настоящие задания выглядят сложнее. Запуск выполняется командой обращения к Workload Management System (WMS) – специальной системе управления загрузкой заданий. Более сложные задания могут содержать ссылки на грид-файлы, которые рассматриваются ниже, и рабочие области данных (группа файлов или директория), которые в большинстве грид-систем называются «песочницей» (sandbox). Имеются вводные «песочницы», которые нужны для запуска задания, и выводные «песочницы», которые сформированы выполненным заданием.

Запущенное задание в *gLite* может находиться в нескольких состояниях (см. рис. 1):

» **SUBMITTED** – задание запущено пользователем, но еще не обработано сетевым или прокси-сервером планирования запуска (WM).

» **WAITING** – задание обработано сервером прокси, но еще не обслужено сервером загрузки.

» **READY** – заданию уже назначен сайт, где оно будет выполняться, но оно пока не передано на этот сайт.

» **SCHEDULED** – задание ожидает в очереди на исполнение на сайте, где оно должно быть выполнено.

» **RUNNING** – задание выполняется.

» **DONE** – задание завершено.

» **ABORTED** – задание завершено аварийно системой планирования загрузки.

» **CANCELLED** – задание завершено досрочно по команде пользователя.

» **CLEARED** – рабочие области задания («песочница») переданы на компьютер пользователя.

Перед запуском полезно выяснить, какие, собственно, ресурсы будут доступны для выполнения такого простого задания. Это можно сделать командой

```
glite-wms-job-list-match -a JobShownAbove.jdl
```

Здесь может быть показан список сайтов (кластеров), где имеет смысл выполнять данное задание. Запустить же его можно командой

```
glite-wms-job-submit -a JobShownAbove.jdl
```

Если все пройдет нормально, вы получите в ответ строку, которая представляет собой идентификатор задания, вроде

```
https://lcg16.sinp.msu.ru:9000/Ug5Vkpwd14zONrfndJ_Iyg
```

Заметим, что это не обычный URL, а просто уникальный идентификатор задания. Используя его, можно посмотреть текущее состояние задания командой

```
glite-wms-job-status jobId
```

Аналогично, чтобы прекратить выполнение задания, применяется команда

```
glite-wms-job-cancel jobId
```

Можно запускать как неинтерактивные (пакетные), так и интерактивные задания, включающие взаимодействие с пользователем, задания с использованием MPI, организацию больших групп заданий и т.д. Кроме этого, существует возможность описания взаимозависимости заданий. Например, задание A1 должно выполняться раньше заданий A2 и A3, причем задание A3 должно запускаться на исполнение только после успешного выполнения заданий A1 и A2. Наконец, возможно использовать так называемые параметрические задания, когда необходимо запустить серию однотипных вычислений, отличающихся одним параметром.

Возникает естественный вопрос: а где же выполняются эти задания? Где хранятся данные? В терминологии проекта *gLite* для этого имеется специально организованные сайты и серверы:

» Сайт *«вычислительный элемент»* (Computational Element – CE). Обеспечивает выполнение заданий; иными словами, за термином CE скрывается вычислительный кластер. Как правило (но не обязательно), недалеко от CE имеется один или несколько так называемых *«элементов памяти»*.

» Сайт *«элемент памяти»* (Storage Element – SE). Это сайт, который обеспечивает хранение и передачу данных в больших объемах. Элемент SE может быть использован как ближайшим CE, так и многими другими, в том числе удаленными.

» Сервер *«рабочий узел»* (Worker Node – WN). Термином WN обозначается один из узлов компьютерного кластера, т.е. один из нескольких (многих) компонентов CE.

» Сервер *«пользовательский интерфейс»* (User Interface – UI). Термин UI означает любой компьютер (сервер, настольный ПК или ноутбук), где установлено клиентское программное обеспечение системы *gLite*.

Сайты (CE и SE) могут физически находиться как в одном помещении (например, в машинном зале), так и в разных городах, странах, на разных континентах или планетах.

Таким образом, задание, запущенное в приведенном выше примере, выполняется на одном из доступных CE.

На **рис. 1** представлена блок-схема выполнения задания в гриде. В колонке справа приведена последовательность состояний задания в процессе выполнения (они отмечены буквами от «b» до «j»). Слева имеется блок-схема различных элементов грид-архитектуры, которые обрабатывают задание. Слева вверху мы видим, как пользователь запускает задание с компьютера UI («пользовательский интерфейс»). Действию пользователя по запуску задания (команда glite-wms-job-submit) приписана литера a, которая не показана на рисунке. Далее UI связывается с сервером Resource Broker (RB) – брокером ресурсов грида. Чтобы найти подходящий вычислитель-

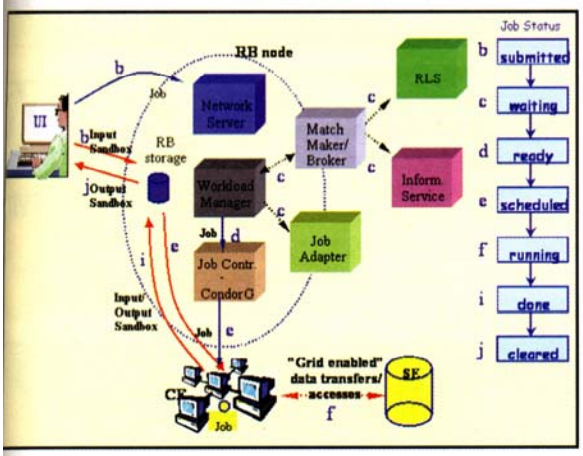

Рис. 1. Блок-схема прохождения заданий.

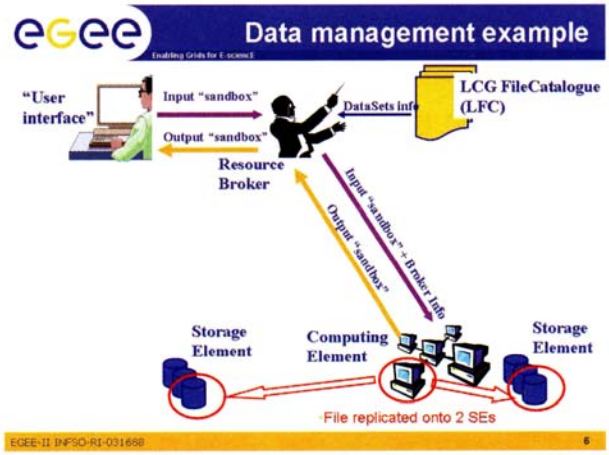

Иллюстрация взята из презентации EGEE Tutorial, 23rd APAN Meeting, Manila, Jan 22, 2007.

Рис. 2. Роль брокера ресурсов в *gLite*.

ый элемент CE и, если необходимо, элемент SE, сервер RB использует другие информационные системы грида:

- **Replica Location Service** (RLS) – сервис нахождения реплик файлов (с использованием LFC);
- **Information Service** (IS) – информационный сервис, который предоставляет сведения о ресурсах (CE, SE), доступных в настоящее время (мы займемся ими в следующий раз).
- **Match-Maker-Broker** – сервис, который производит подбор подходящих элементов CE и SE для выполнения задания.

В конечном счете брокер ресурсов играет роль дирижера при запуске задания (см. **рис. 2**).

Как мы видим, UI передает на RB само задание и вводную «песочницу» – это стадия **b** (SUBMITTED) на **рис. 2**. Далее производится выбор подходящих элементов для исполнения задания – стадия **c** (WAITING). Затем выполняется передача задания на соответствующий элемент CE и другие служебные операции для постановки задания в очередь на конкретном CE – **d** (READY). После установки в очередь на конкретном кластере (элементе CE) задание находится в стадии **e** (SCHEDULED). Здесь оно обслуживается системой пакетной обработки заданий (одной из следующих: Condor, LSF, PBS (ProlOpenlTorque), SGE). Во время выполнения задание может использовать данные из элемента SE или генерировать новые файлы и записывать их в элемент SE с использованием грид-инструментария – стадия **f** (RUNNING). Наконец, задание завершается – стадия **i** (ENDED). Финальная часть для грид – стадия **j** (CLEARED), когда выводная «песочница» пересылается в финальную точку.

Итак, мы рассмотрели запуск и жизненный цикл простых заданий. Ответ на второй вопрос – откуда эти задания берут данные – и некоторые другие будет дан в следующий раз. LXF

Через месяц Доступ к данным, передача файлов и информационный сервис *gLite*.