Р11-2008-68

А. Г. Долбилов[1], Ю. П. Иванов[1,2]

ЭЛЕМЕНТ GRID-СИСТЕМЫ LCG-2 В ЛЯП

[1]Объединенный институт ядерных исследований, Дубна
[2]Технический университет им. Федерико Санта-Мария, Вальпараисо, Чили

Долбилов А. Г., Иванов Ю. П. P11-2008-68
Элемент Grid-системы LCG-2 в ЛЯП

Дано описание компьютерного кластера Лаборатории ядерных проблем ОИЯИ, на котором реализован второй в ОИЯИ вычислительный элемент Grid-системы LCG-2. Рассмотрена конфигурация системы, позволяющая эффективное совместное использование кластера как локальными пользователями ЛЯП, так и другими в рамках Grid-вычислений коллаборации ATLAS. На примерах показаны основные этапы подготовки и счета как обычных задач на самом кластере, так и с использованием Grid, начиная с получения сертификатов, запуска задач и заканчивая получением результатов. Рассматриваются перспективы развития кластера.

Работа выполнена в Лаборатории ядерных проблем им. В. П. Джелепова ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2008

Dolbilov A. G., Ivanov Yu. P. P11-2008-68
Grid-System Element of LCG-2 in DLNP

We present here the description of the computer cluster at the Dzhelepov Laboratory of Nuclear Problems of JINR, where the second Grid node at JINR was realized. The configuration of the system, which allows effective joint usage of cluster resources both for local users and for others within the framework of the ATLAS collaboration is examined. Examples are given for basic stages of preparing and running ordinary cluster jobs and with the Grid usages, starting from obtaining CA certificates, submitting jobs and retrieving the results. Perspectives of the cluster upgrade are discussed.

The investigation has been performed at the Dzhelepov Laboratory of Nuclear Problems, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2008

ВВЕДЕНИЕ

Идея совместного использования вычислительных ресурсов компьютеров, объединенных в сеть, появилась, пожалуй, одновременно с самим понятием «компьютерная сеть». Сначала были развиты методы и способы для информационного обмена: от простейшего обмена файлами (FTP) до сетевых файловых систем (NFS, AFS), электронная почта и, наконец, Web-сервисы, которые для многих вообще являются синонимом понятия «сеть».

Распределение *вычислений* (т. е. исполняемого кода, а не только данных) появилось позднее, поскольку поддержка такого рода взаимодействия компьютеров требует довольно больших усилий даже для компьютерных кластеров с одинаковой процессорной архитектурой, не говоря уже о гетерогенных системах.

Простейшим вариантом распределенных вычислений является, например, обычная пакетная система (типа PBS или Condor), в рамках которой пользователь, «поставив» свою задачу в «очередь», не должен сам беспокоиться о том, как, когда и на какой конкретно машине будет выполнено его задание. Как правило, такие системы предполагают не только одинаковость аппаратной и программной среды, но и одинаковость средств авторизации пользователя, например, через синхронизацию пользовательских данных на всех машинах кластера. Помимо данных авторизации необходима и «синхронизация» файлов пользователя («домашняя» директория и пр.). Все это может быть достигнуто достаточно простыми средствами на кластерах масштаба «предприятия», т. е. при наличии централизованного администрирования. Однако более крупные масштабы, например, объединение счетных ресурсов нескольких организаций, требуют существенного расширения механизмов авторизации пользователя в рамках такого «гиперкластера».

В середине 1990-х гг. начались разработки методов распределенных вычислений, получившие впоследствии название «Grid». В работах [1] и [2], считающихся классическим введением в Grid, приведены основные понятия и принципы, которые легли в основу этого подхода. Для идентификации пользователя в Grid-среде было предложено использовать CA-сертификаты (Certificate Authority), признаваемые всеми «участниками» объединения (т. е. на всех кластерах, входящих в Grid). Однако задавать права каждого пользователя на каждом кластере напрямую на основе CA-сертификатов было

бы слишком громоздкой задачей. Было введено понятие «виртуальной организации» (VO — Virtual Organization) — объединения пользователей (возможно, из географически различных мест) в единую группу. При этом ресурсы (время счета, память, диск и прочее), выделяемые пользователю на каждом конкретном кластере, определяются VO, в которую входит пользователь (CA-сертификат необходим при проверке принадлежности к той или иной VO). На каждом кластере могут быть свои списки «поддерживаемых» VO и выделяемых им ресурсов. Помимо авторизации были сформулированы и принципы предоставления задаче пользователя как вычислительных (времени, размера выделяемой памяти и прочего), так и файловых ресурсов (механизмов загрузки данных задачи с удаленной машины на машину, где собственно и будет проводиться ее счет, а также передачи результатов счета в обратном направлении). Для реализации всех этих принципов был также создан специальный программный комплекс Globus Toolkit [3]. Набор программ и утилит, реализующий такую структуру распределенных вычислений и ставший дополнительным («промежуточным») уровнем между базисной операционной системой и программой пользователя, получил название «middleware».

Именно этот подход был принят в ЦЕРН для хранения и обработки данных экспериментов ALICE, ATLAS, CMS и LHCb на создаваемом коллайдере LHC. Комплекс программ, обеспечивающих работу Grid middleware на десятках тысяч машин в сотнях организациях, входящих в эти коллаборации, получил название LCG (LHC Computing Grid). Ресурсы LCG Grid доступны также и целому ряду других Grid-комплексов, среди которых как международные (EGEE — Enabling Grids for E-SciencE, OSG — Open Science Grid и пр.), так и национальные и региональные, например, INFN Grid в Италии [5] и RDIG (Russian Data Intensive Grid) в России [6]. Возможность счета конкретной задачи на любой из машин, входящей в различные Grid-структуры, каждая из которых имеет свои особенности, приводит к перегруженности программного комплекса: необходимо, например, учитывать особенности конкретной пакетной системы (PBS, Condor и пр.), установленной на конкретном счетном элементе Grid и мн. др. В настоящий момент в качестве основной версии LCG middleware принята реализация gLite [7].

В ОИЯИ работы по созданию LCG-сегмента начались в 2002 г. На базе серверов ЛИТ была создана Grid-структура сегмента JINR-LCG2: «вычислительный элемент» (CE — Computing Element), «файловый элемент» (SE — Storage Element), «брокер ресурсов» (RB — Resource Broker) и прочие Grid-«сервисы», необходимые для полноценного функционирования сегмента.

В начале 2004 г. в ЛЯП по инициативе дирекции была создана неформальная группа, получившая впоследствии название LGD (LNP Grid Development), в которую вместе с авторами статьи вошли К. И. Грицай и В. Г. Ольшевский. В результате работы этой группы в ЛЯП был создан второй в ОИЯИ «вычислительный элемент» сегмента JINR-LCG2.

В первом разделе рассмотрены основные особенности конфигурации созданного кластера. Далее на простых примерах показаны возможности его использования для счета локальных задач (как обычных, так и многопроцессорных MPI-заданий) в пакетном режиме. После чего рассказано о работе с Grid: рассмотрен весь путь от получения CA-сертификата пользователя до подготовки, запуска и получения результатов Grid-задачи. В заключении подведены итоги и рассмотрены пути дальнейшего развития кластера.

1. СТРУКТУРА КЛАСТЕРА

Изначально перед группой LGD задача об организации распределенных вычислений ставилась в несколько более общей формулировке: имея в ЛЯП как минимум несколько сотен компьютеров в достаточно неплохой конфигурации, большинство из которых в каждый конкретный момент просто находятся в состоянии ожидания ввода команд, рассмотреть возможные способы использования «простаивающих» счетных мощностей. Поиски решения этой задачи естественным образом привели к рассмотрению Grid-технологий, которые на тот момент были для авторов совершенно новым направлением исследований.

Вначале были сделаны (надо признать, достаточно робкие) попытки использовать для организации распределенной работы собственно Globus Toolkit. Однако вскоре стало ясно, что для достижения хоть какого-то практического решения стоявшей проблемы необходимы достаточно большие дополнительные усилия по созданию реально работающего Grid-комплекса. Вместо этого было решено использовать разработки LCG Grid, тем более что как ЛЯП, так и ОИЯИ в целом участвуют в экспериментах ЦЕРН, в частности, в коллаборации ATLAS.

Как все задачи экспериментов на LHC, так и сама реализация LCG Grid предполагают использование ОС Linux. Однако даже привлечение готовых пакетов программ в данном случае тоже не является тривиальной задачей. Дело в том, что надо было создать «стандартную» конфигурацию Linux машины, которая имеет все необходимые для профессиональной работы программные пакеты (компиляторы, библиотеки программ, текстровые процессоры, мультимедийные средства и т. д.). И в то же время это «рабочее место» должно быть также и «рабочим узлом» (WN — Worker Node) и/или Grid-«интерфейсом пользователя» (UI — User Interface) в терминологии LCG. Первое (т. е. пользовательские программы) требует установки и постоянного обновления достаточно «свежих» версий как самой операционной системы, так и всех приложений. При этом LCG middleware во многом консервативно. Достаточно сказать, что базовой ОС для всех типов узлов LCG Grid до сих пор является Scientific Linux 3 (SL3) или аналогичные ему дистрибутивы на

основе Red Hat Enterprise Linux 3 (RHEL3). Соответствующие ему наборы компиляторов, стандартных библиотек и приложений за прошедшее с момента появления RHEL3 время успели заметно обновиться в новых версиях: в марте 2007 г. появилась версия RHEL5, а в мае 2007 г. — аналогичная ей SL5. Желание пользователей работать с новыми версиями программ вполне естественно, в том числе и потому, что в новых версиях появилась стабильная поддержка современных 64-битовых процессоров, в то время как в SL3 она еще слаба, да и сам дистрибутив LCG middlware базируется на 32-битовой архитектуре i386. Лишь в начале 2008 г. для некоторых типов узлов Grid появились версии LCG middleware (gLite 3.1) на базе SL4 и с поддержкой 64-битовой архитектуры.

Чтобы максимально удовлетворить потребности локальных пользователей, было решено сделать в качестве базисной ОС для WN и UI машин версию Scientific Linux 4 (SL4) (на тот момент это была самая «свежая» ОС). Отметим, что на кластере ЛЯП работа с SL4 ведется с 2005 г., тогда как дистрибутивы LCG middleware для WN и UI былы созданы лишь весной 2007 г. Центральный кластер ЛИТ также перешел на SL4 лишь в том же году.

Одновременно была усилена «защита» системы. Для авторизации пользователей была выбрана система Kerberos 5, хотя многие приложения поддерживают лишь более «слабую» версию — Kerberos 4. Такое повышение уровня «безопасности» потребовало, в частности, и дополнительных усилий по интеграции с файловой системой AFS, которая используется для хранения директорий пользователей. Общий объем дискового пространства на созданном в рамках кластера AFS-сервере около 1,2 Тбайт, что пока вполне достаточно как для файлов пользователей, так и для программ. Заметим в связи с этим, что на сегодняшний день в ОИЯИ есть несколько AFS-серверов, и есть желание объединить их в единую структуру. Для AFS стандартным является Kerberos 4. И если «рецепты» по переводу AFS на Kerberos 5 известны, то для интеграции с пакетной системой запуска задач PBS (реально установлена ее реализация Torque) потребовалась модернизация системы PSR (Password Storage and Retreive), используемой PBS для получения токена AFS на время выполнения программы пользователя для доступа к его файлам. Пришлось добавить в стандартный код PSR возможность аутентификации с использованием Kerberos 5. Заметим, что именно этот вариант PSR и был взят за основу при обновлении центрального кластера ЛИТ весной 2007 г. Из прочих модификаций стоит также упомянуть и установку современной версии Secure Shell, в которой в отличие от ранних версий поддержка Kerberos 5 реализована в полном объеме, что позволило осуществить «прозрачный» файловый обмен между машинами кластера (это нужно, в частности, для работы системы PBS) с использованием в качестве средства авторизации host «ключей» Kerberos 5.

На все эти, а также многие другие неупомянутые модификации потребовалось время. Да еще и не зависящие от авторов факторы внесли свою лепту.

Но как бы то ни было, в начале лета 2006 г. «вычислительный элемент» ЛЯП начал свою жизнь. В первую очередь это относится к возможности работы пользователей в обычном локальном (т. е. без использования Grid) режиме.

На первом этапе LGD-кластер состоял из следующих машин (все компьютеры однопроцессорные Pentium 4, 3,0 ГГц CPU, 1 Гбайт RAM):

Машина	Система	Функции
lgdce01	SL3	CE
lgd2	SL4	WN+UI
lgd3	SL4	WN+UI
lgd4	SL4	WN+UI
lgdafs	SL4	Kerberos и AFS-сервер

Отметим, что лишь на одной машине lgdce01 была установлена ОС SL3, что, впрочем, в данном случае некритично, поскольку реально пользователь работает на других машинах: как «интерфейс пользователя» (UI) так и «рабочий узел» (WN) работают под SL4, т. е. все задачи пользователя выполняются в среде SL4. В конце 2006 г. к этому списку WN+UI добавились принадлежащие различным группам ЛЯП еще три машины. Они имели более современную конфигурацию на базе двухпроцессорных Intel Xeon: proton (3,0 ГГц, 2 Гбайт RAM) и nemo (2,4 ГГц CPU, 1 Гбайт RAM) из НЭОФПЭ и bes3 (2,4 ГГц CPU, 512 Мбайт RAM) из НЭОВП.

Компьютер lgdce01 работает как «вычислительный элемент» CE, обеспечивая, в частности, работу пакетной системы (сервер Torque) и планировщика задач (Maui) для локальных пользователей. Остальные машины (lgd2, lgd3...) — это рабочие станции пользователя. На них установлены компиляторы (C, C++, FORTRAN), библиотеки программ CERNLIB, текстровые процессоры (TEX/LATEX, Open Office), web-браузеры (Mozilla Seamonkey, FireFox), почта (pine, Mozilla Thunderbird), мультимедийные средства (звук и видео CD/DVD, Skype) и т. д.

Помимо обычной работы (как на обычном desktop-компьютере) пользователь может посылать свои задачи и на длительный счет, при этом реальное их выполнение будет проходить на свободной машине кластера. Рассмотрим теперь на простых примерах подготовку и запуск локальных (не Grid) заданий на кластере.

2. ЗАПУСК ЗАДАЧ В ПАКЕТНОМ РЕЖИМЕ

Перед использованием пакетной системы пользователю необходимо сохранить свой пароль в системе PSR. Для этого достаточно набрать команду

pbspwstore и ввести свой пароль (здесь и далее в примерах консольные сообщения не всегда будут приводиться полностью):

```
~> pbspwstore
...
Enter your AFS password:    ← введите пароль
...
```

Эту команду надо вызвать лишь один раз перед началом работы с пакетной системой, а также при каждом последующем изменении пользователем своего пароля. Без этого задача пользователя, запущенная пакетной системой PBS, не сможет получить доступ к его файлам, расположенным на AFS-сервере.

Поставить свою задачу в очередь заданий пользователь может командой qsub. Аргументом этой команды служит имя командного файла (скрипта), который содержит всю последовательность команд для выполнения пакетной системой; т. е. это обычный скрипт (например, командных оболочек sh, bash, tcsh или даже, к примеру, perl или python), в котором допускается указание дополнительных параметров для пакетной системы в виде строк, начинающихся с #PBS (для большинства оболочек это, естественно, просто комментарий), а также использование переменных системной среды (environment variables) вида PBS_O_*name*. Собственно счет задачи и заключается в запуске пакетной системой этого скрипта в «домашней директории» пользователя. Как правило, *name* соответствует значению системной переменной среды на момент запуска команды qsub. Например, значению переменной PBS_O_HOME при выполнении задания пакетной системой присвоено значение переменной HOME, т. е. «домашней директории» пользователя, PBS_O_HOST — имя машины, с которой запущено задание, а PBS_O_WORKDIR — это имя текущей директории, из которой вызвана команда qsub, что удобно использовать при запуске задач из поддиректорий.

2.1. Простая задача. Рассмотрим простой пример: в директории пользователя work/gamma есть программа gamma.f, которая использует Γ-функцию GAMMF из библиотеки CERNLIB и выдает результат как на консоль, так и в файл gamma.out:

```
PROGRAM  GAMMA
REAL     GAMMF,X,Y
EXTERNAL GAMMF
OPEN  (1,FILE='gamma.out')
X = 1.5
Y = GAMMF(X)
PRINT    *, X,Y
WRITE (1,*) X,Y
END
```

Компилируем программу с указанием подключения библиотеки CERNLIB

```
~/work/gamma> g77 -o gamma gamma.f $(cernlib genlib)
```

и создаем простейший скрипт `gamma.sh` для ее запуска пакетной системой:

```
cd $PBS_O_WORKDIR
./gamma
```

Скрипт выполняет переход в директорию на момент вызова `qsub` (мы будем далее пускать `qsub` из директории, где находится и сама программа) и запускает программу `gamma` из этой директории. Вместо указания рабочей директории через переменную `$PBS_O_WORKDIR` можно при желании и явно указать `work/gamma`, но вариант с `$PBS_O_WORKDIR` представляется более универсальным: например, при перемещении всех файлов в другую директорию не потребуется менять содержимое скрипта `gamma.sh`.

Теперь все готово для отправления этой задачи на счет. Ставим ее в очередь заданий командой `qsub`, которая при успешном завершении своей работы сообщает номер, по которому можно следить за состоянием задачи:

```
~/work/gamma > qsub gamma.sh
46666.lgdce01.jinr.ru
~/work/gamma >
```

Посмотреть статус выполняемых заданий можно командой `qstat`:

```
~/work/gamma > qstat
Job id           Name       User       Time Use S Queue
---------------- ---------- ---------- -------- - -----
46571.lgdce01    STDIN      atlas001   00:00:07 R atlas
46585.lgdce01    STDIN      patl002    08:45:15 R atlas
46666.lgdce01    gamma.sh   yupi       00:00:00 C serial
```

Как видим, в нашем случае задача уже завершена (имеет статус «`C`», т. е. `Complete`). В результате в директории появились новые файлы с именами скрипта и добавочным расширением `.o`<*номер*> (в нем то, что печаталось командой `PRINT` на `stdout` — «стандартное устройство вывода») и `.e`<*номер*> (`stderr` — сообщения об ошибках, которых у нас нет и поэтому этот файл пуст), а также файл `gamma.out`, который был создан программой:

```
~/work/gamma > ls -l
total 21
-rwxr-xr-x 1 yupi lnup 16418 Oct 26 21:42 gamma
-rw-r--r-- 1 yupi lnup   187 Oct 26 21:41 gamma.f
-rw-r--r-- 1 yupi lnup    19 Oct 26 21:43 gamma.out
-rw-r--r-- 1 yupi lnup    26 Oct 26 21:40 gamma.sh
-rw------- 1 yupi lnup     0 Oct 26 21:43 gamma.sh.e46666
-rw------- 1 yupi lnup    19 Oct 26 21:43 gamma.sh.o46666
```

Эта задача завершилась быстро, но при большой загрузке кластера может пройти заметное время до момента собственно счета. Получить более детальную информацию о статусе конкретной задачи можно командой `qstat` с параметром «`-f`» и указанием номера задания (ниже мы приводим лишь несколько начальных строк статуса):

```
~/work/gamma > qstat -f 46666
Job Id: 46666.lgdce01.jinr.ru
    Job_Name = gamma.sh
    Job_Owner = yupi@lgd2.jinr.ru
    resources_used.cput = 00:00:00
    resources_used.mem = 1408kb
    resources_used.vmem = 8352kb
    resources_used.walltime = 00:00:01
    job_state = C
    queue = serial
    ...
```

При необходимости можно удалить задачу из очереди командой `qdel` с указанием номера задания. Подробнее о работе всех этих команд `qsub`, `qstat`, `qdel` можно посмотреть, например, с помощью команды `man`.

2.2. Дополнительные параметры счета. Дополнительные параметры для пакетной системы могут быть указаны как параметры команды `qsub`. Например, указать, что задаче нужен 1 ч счетного времени (и это будет учтено планировщиком пакетной системы при оценке приоритета задачи) можно командой

```
~/work/gamma > qsub -N Gamma -l cput=1:00:00 gamma.sh
```

Те же самые указания можно ввести и в скрипте задачи. Например, в начало скрипта `gamma.sh` можно добавить следующие строчки, которые присвоят задаче имя `Gamma`, установят имена файлов `stdin` и `stdout`, укажут требуемые ресурсы (время и память), а также сообщат пакетной системе, куда и при каких событиях отправить письмо (здесь указаниями для пакетной системы являются лишь строки, начинающиеся с `#PBS`, строки с `###` — комментарии как для пакетной системы, так и для интерпретатора команд):

```
### Set job name
#PBS -N Gamma

### Set names for 'stdout' and 'stderr' files
#PBS -o gamma-run.out
#PBS -e gamma-run.err

### Job CPU time (in seconds or [[HH:]MM:]SS)
#PBS -l cput=60

### Job memory (in b, kb, mb, gb or tb)
#PBS -l mem=400mb
```

```
### Mail address
#PBS -M boris@jinr.ru

### Mail at job start (b) and end (e)
#PBS -m be
```

Теперь при запуске задачи на счет и при его окончании будут отправлены письма по указанному адресу (можно перечислить и несколько адресов, записав их через запятую), а результаты записаны в указанные файлы:

```
~/work/gamma > qsub gamma.sh
46696.lgdce01.jinr.ru
~/work/gamma > qstat
Job id           Name    User       Time Use S Queue
---------------- ------- ---------- -------- - -----
46571.lgdce01    STDIN   atlas001   00:00:07 R atlas
46585.lgdce01    STDIN   patl002    14:08:33 R atlas
46696.lgdce01    Gamma   yupi       00:00:00 C serial
~/work/gamma > ls -l
total 21
-rwxr-xr-x  1 yupi lnup 16418 Oct 26 21:42 gamma
-rw-r--r--  1 yupi lnup   187 Oct 26 21:41 gamma.f
-rw-r--r--  1 yupi lnup    19 Oct 27 03:12 gamma.out
-rw-------  1 yupi lnup     0 Oct 27 03:12 gamma-run.err
-rw-------  1 yupi lnup    19 Oct 27 03:12 gamma-run.out
-rw-r--r--  1 yupi lnup   442 Oct 27 03:06 gamma.sh
```

2.3. Запуск MPI-задачи. Рассмотрим теперь пример запуска программы, использующей MPI (Message Passing Interface), которая требует для своей работы одновременно несколько процессоров. В директории `work/mpi` есть программа `multi.c`, которая должна быть загружена на несколько машин. Программа очень проста: каждый загруженный процесс лишь печатает свой индекс в комплексе процессов и имя машины, на которой он загружен (информацию об использовании MPI при работе с C, C++ и FORTRAN можно найти на сайте [8], здесь же лишь отметим, что на кластере установлена MPI-реализация MPICH2):

```
#include "mpi.h"
#include <stdio.h>

int main(int argc, char** argv) {
  FILE *F;
  char  buff[200];
  char  host[100];
  int   rank,size;
  F = popen("hostname","r");
  fscanf(F,"%s",host);
  pclose(F);
  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  MPI_Comm_size(MPI_COMM_WORLD,&size);
```

```
  printf("I'm %d of %d at host '%s'\n",rank,size,host);
  MPI_Finalize();
  return 0;
}
```

Компилируем программу

```
~/work/mpi > mpicc -o multi multi.c
```

и создаем скрипт `multi.sh`, в котором указываем, что задача требует три машины и на каждой по два процесса (т. е. всего комплекс из шести процессов). Собственно загрузка программы на все требуемые компьютеры будет сделана командой `mpiexec`:

```
### Job nodes to run
#PBS -l nodes=3:ppn=2
cd $PBS_O_WORKDIR
mpiexec multi
```

Запускаем на счет и смотрим статус выполнения:

```
~/work/mpi > qsub multi.sh
46703.lgdce01.jinr.ru
~/work/mpi > qstat
Job id          Name        User       Time Use S Queue
--------------- ----------- ---------- -------- - -----
46571.lgdce01   STDIN       atlas001   00:00:07 R atlas
46585.lgdce01   STDIN       patl002    14:50:42 R atlas
46703.lgdce01   multi.sh    yupi       00:00:00 C parallel
```

По завершении работы (статус «C» — `Complete`) в директории появилась пара файлов (`stderr` — пустой файл сообщения об ошибках и stdout — результат работы программы):

```
~/work/mpi > ls -l
total 9
-rwxr-xr-x  1 yupi lnup 5926 Oct 27 03:52 multi
-rw-r--r--  1 yupi lnup  397 Oct 27 02:35 multi.c
-rw-r--r--  1 yupi lnup   79 Oct 27 03:52 multi.sh
-rw-------  1 yupi lnup    0 Oct 27 03:54 multi.sh.e46703
-rw-------  1 yupi lnup  204 Oct 27 03:55 multi.sh.o46703
~/work/mpi > cat multi.sh.o46703
I'm 0 of 6 at host 'lgd3.jinr.ru'
I'm 1 of 6 at host 'lgd3.jinr.ru'
I'm 5 of 6 at host 'bes3.jinr.ru'
I'm 4 of 6 at host 'bes3.jinr.ru'
I'm 2 of 6 at host 'lgd4.jinr.ru'
I'm 3 of 6 at host 'lgd4.jinr.ru'
```

Отметим, что во всех рассмотренных выше случаях мы нигде не указывали, в какую именно очередь нужно поставить задание. В настоящее время

для локальных пользователей существуют две счетные очереди: `serial` (обычные задания) и `parallel` (MPI-задачи). Команда `qsub` направляет задание сначала в очередь `batch` и далее пакетная система сама в зависимости от требований задачи (т. е. памяти, времени, количества процессоров и т. д.) определяет нужную очередь. Помимо очередей для локальных задач на кластере созданы также и очереди для счета задач системы Grid, к рассмотрению работы в которой мы сейчас и переходим.

3. ПОДГОТОВКА И ЗАПУСК ЗАДАЧ GRID

Прежде чем мы перейдем к практической работе с Grid, полезно ознакомиться с тем, что же такое LCG. В приложении дан краткий обзор полезных ссылок как непосредственно по LGD, так и по Grid-вычислениям вообще. Здесь же мы ограничимся рассмотрением процесса счета Grid-задачи в самых общих чертах.

На сегодняшний день LCG является, по сути, несколько эклектичной сборкой различных программных пакетов на основе, как уже отмечалось выше, Globus Toolkit, который используется для идентификации пользователей и их заданий, доступа к файлам, а также для сбора информации о загрузке компьютеров и распределения заданий по машинам разных организаций. В конце концов задание пользователя попадает «в руки» той или иной пакетной системы, установленной на кластере машин. Globus Toolkit дает и другие возможности распределенного счета, однако на данный момент в основном счет в LCG проходит примерно таким образом.

1. Пользователь на машине с установленным пакетом программ LCG UI (User Interface), идентифицирует себя в LCG. При этом устанавливается также его принадлежность к «виртуальной организации» (VO), к примеру, ATLAS.

2. Подготовленное задание (обычная или MPI-программа) отправляется в Grid-систему, где после соответствующего анализа (исходя из требований задания и реальной загрузки различных кластеров) принимается решение, на какой именно «счетный элемент» (CE) его отправить.

3. Поступив на выбранный CE, задание попадает уже в локальную очередь данного CE, где и выполняется с правами виртуального пользователя из соответствующей VO. Например, для члена VO ATLAS это может быть `atlas007`. Такая «виртуализация» пользователя необходима, поскольку реальное имя пользователя известно только системе, с которой было отправлено Grid-задание, а на всех остальных CE его, естественно, нет.

4. Пользователь может посмотреть статус отправленной задачи и по завершении работы задания загрузить с удаленной машины полученные файлы с результатами счета.

Более детально о структуре LCG можно прочесть в «LCG-2 User Guide», гл. 3. Именно на это «руководство пользователя» (см. приложение) мы и далее будем ссылаться при рассмотрении конкретных примеров.

Первый этап практической работы в Grid LCG — это получение индивидуального пользовательского CA-сертификата. Практически же это означает получение пары файлов из «центра сертификации» (используется система с двумя «ключами» — privat и public). Получив эти «ключи», необходимо зарегистрироваться в своей VO, например, ATLAS. (Пока что в LCG пользователь может быть членом лишь одной VO.) Только после этого возможен запуск заданий Grid. Рассмотрим все эти этапы по порядку.

3.1. Получение сертификата и регистрация в VO. Последовательность действий для получения сертификата подробно описана на странице CA-сайта RDIG:

http://ca.grid.kiae.ru/RDIG/certificates/obtain.html

Внизу этой странички (раздел «Я хочу получить») надо выбрать пункт «пользовательский сертификат» и заполнить появившуюся форму. В пункте «Организация» надо выбрать «JINR, jinr.ru». После заполнения формы нажать кнопку «Далее» и скачать скрипт `user_cert-request.sh` и PDF-файл `user_cert_form.pdf` с частично заполненным бланком запроса сертификата. Полученный скрипт надо запустить, как сказано на сайте RDIG, «на машине с UNIX-совместимой рабочей средой и установленным пакетом OpenSSL». Это можно сделать на любой из пользовательских машин кластера с установленным LCG «интерфейсом пользователя» (User Interface), например, lgd2.jinr.ru. Доступ извне на все эти машины только по `ssh`. С них же в дальнейшем и будет запуск заданий для счета в LCG.

Запустите полученный скрипт, например, так:

```
~> sh user_cert-request.sh
```

В результате в «домашней директории» пользователя будут созданы поддиректория `.globus` и в ней три файла: `usercert.pem`, `userkey.pem` и `userreq.mail`. При этом также будет выдан на экран некий длинный ключ, последние 10 цифр которого надо запомнить — они понадобятся для заполнения бланка запроса. Запомнить надо будет и введенный пароль: если забудете, то придется получать сертификат заново!

Файл `userkey.pem` — это уже окончательный privat key. Файл `usercert.pem` на этом этапе всего лишь напоминание, что на его месте должен впоследствии быть другой, полученный из центра сертификации. Для

этого отправьте файл `userreq.mail` по электронной почте в центр сертификации, например, так:

```
~> mail rdig-cagrid.kiae.ru < userreq.mail
```

или любой другой почтовой программой. Главное, чтобы этот файл был в «теле» сообщения, т. е. отправлен как обычный текст, а не прикреплен к письму (attached). Через какое-то время вы получите ответ, что ваш запрос принят. Будет также прислан его серийный номер (serial number) — он тоже нужен для заполнения бланка запроса сертификата. С заполненным бланком запроса надо обратиться к авторизованному представителю RDIG CA. В ОИЯИ это Д. А. Олейник (ЛИТ, к. 406Б, тел. 6-23-02). И лишь только после этого останется дождаться, когда центр сертификации свяжется с ответственным в ОИЯИ и пришлет очередное письмо, которое на этот раз уже будет содержать окончательный сертификат. Можно сохранить весь текст этого письма или же фрагмент, начинающийся с

```
-----BEGIN CERTIFICATE-----
```

и оканчивающийся

```
-----END CERTIFICATE-----
```

в файле `usercert.pem`, который надо скопировать в директорию `.globus`.

Получив файлы сертификата CA можно переходить к этапу регистрации в VO. Вся процедура регистрации описана на страничке ЦЕРН

http://lcg.web.cern.ch/lcg/Users/registration/registration.html

Поскольку п. 1 и 2, указанные на этой страничке, относящиеся к получению сертификата, уже пройдены, то можно сразу перейти к п. 3, т. е. к добавлению сертификата в ваш браузер. Эта процедура описана на страничке

http://lcg.web.cern.ch/lcg/Users/registration/registration.html

В п. 4 предлагается определиться с виртуальной организацией. Надо выбрать ту, задачи которой вы будете считать. После выбора соответствующей VO для перехода на страничку регистрации надо будет ввести пароль вашего сертификата и далее заполнить очередную форму. После чего останется дождаться письма, которое уведомит о включении вас в выбранную вами VO. На этот этап тоже может уйти несколько дней.

Как видим, получение сертификатов и регистрация в VO, что необходимо для работы в LCG, — довольно длительный процесс. Более того, срок их действия ограничен (1 год). И через год тем, кто решит продолжить вычисления в LCG, придется снова получать CA-сертификаты.

Пройдя этап регистрации в Grid, можно начинать работу с LCG. На примере простой программы, которая просто печатает текст «`Hello, Grid!`», пройдем шаг за шагом весь путь от ее подготовки и запуска на счет до по-

лучения файлов результата. Но сначала надо научиться работать с сертификатами. Без этого вы «чужой» для LCG — ни одна программа Grid не будет работать с вами!

3.2. Работа с сертификатами пользователя. Итак, на машине с установленным UI, например, `lgd2.jinr.ru` в поддиректории `.globus` должны быть файлы сертификата с указанными правами доступа (здесь файл `usercert.p12` — это файл сертификата для web-браузера и непосредственно для запуска задачи в LCG не нужен):

```
~ > ls -al .globus
total 14
drwxr-xr-x   2 yupi lnup 2048 Dec 26  2006 .
drwxr-xr-x  29 yupi root 6144 Oct 28 18:43 ..
-r--r--r--   1 yupi lnup 2478 Dec 26  2006 usercert.p12
-r--r--r--   1 yupi lnup 1899 Dec 26  2006 usercert.pem
-r--------   1 yupi lnup  963 Dec 21  2006 userkey.pem
```

Самый первый этап работы — генерация прокси-сертификата, который необходим для работы всех остальных команд. Запускаем команду и вводим свой пароль:

```
~ > grid-proxy-init
Your identity: /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov
Enter Grid pass phrase for this identity:
Creating proxy ........................................... Done
Your proxy is valid until: Mon Oct 29 06:46:36 2007
```

Посмотреть статус этого сертификата можно командой

```
~ > grid-proxy-info
subject  : /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov/
                                              CN=proxy
issuer   : /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov
identity : /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov
type     : full legacy globus proxy
strength : 512 bits
path     : /tmp/x509up_u6003
timeleft : 11:59:06
```

Прокси-сертификат действителен ограниченное время (по умолчанию 12 ч). Досрочно прервать его действие можно командой

```
~ > grid-proxy-destroy
```

Информацию о дополнительных возможностях работы с прокси-сертификатами, в частности, о получении сертификата на более длительное время, можно найти в гл. 4 (разд. 4.4) в «LCG-2 User Guide». Это необходимо для запуска длительных заданий. При этом потребуется имя прокси-сервера (для ОИЯИ это lcgpx01.jinr.ru):

```
~ > myproxy-init -s lcgpx01.jinr.ru -d -n
Your identity: /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov
Enter Grid pass phrase for this identity:
Creating proxy ..................................... Done
Proxy Verify OK
Your proxy is valid until: Sun Nov  4 18:55:20 2007
A proxy valid for 168 hours (7.0 days) for user /C=RU/O=RDIG/
  OU=users/ OU=jinr.ru/CN=Yuri P. Ivanov now exists on
  lcgpx01.jinr.ru.
```

Для просмотра статуса и досрочной отмены действия этого сертификата есть команды:

```
~ > myproxy-info -s lcgpx01.jinr.ru -d
username: /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov
   owner: /C=RU/O=RDIG/OU=users/OU=jinr.ru/CN=Yuri P. Ivanov
  timeleft: 167:53:40  (7.0 days)

~ > myproxy-destroy -s lcgpx01.jinr.ru -d
Default MyProxy credential for user /C=RU/O=RDIG/OU=users/
  OU=jinr.ru/CN=Yuri P. Ivanov was successfully removed.
```

3.3. Запуск задания в LCG и получение результатов. Сначала создадим программу печати сообщения «Hello, Grid!». Для нашей цели достаточно сделать, например, просто скрипт `Hello.sh`:

```
#!/bin/bash
echo Hello, Grid!
```

При желании можно, конечно же, воспользоваться и любым другим языком программирования, главное, чтобы при ее загрузке она выдавала на консоль этот текст. После чего надо создать соответствующее этой программе описание (Job Description) для LCG — файл `Hello.jdl`, в котором перечислены требования задачи. Помимо указаний на сопутствующие файлы (input, output, error)), там могут быть, например, указания на специализированные программы и библиотеки, необходимые для ее выполнения. Подробнее смотри Гл. 6. в «LCG-2 User Guide». В нашем случае этот файл прост:

```
Executable    =  "Hello.sh";
StdOutput     =  "Hello.out";
StdError      =  "Hello.err";
InputSandbox  = {"Hello.sh"};
OutputSandbox = {"Hello.out","Hello.err"};
```

Итак, у нас есть как сама программа (файл `Hello.sh`), так и ее описание (файл `Hello.jdl`). Отправляем ее на счет командой (замените лишь аргумент `-vo dteam` на соответствующий вашей VO, например, `-vo atlas`):

```
~ > edg-job-submit -vo dteam Hello.jdl

Selected Virtual Organisation name (from --vo option): dteam
Connecting to host lcgrb01.jinr.ru, port 7772
Logging to host lcgrb01.jinr.ru, port 9002

**********************************************************
                         JOB SUBMIT OUTCOME
  The job has been successfully submitted to the Network Server.
  Use edg-job-status command to check job current status.
  Your job identifier (edg_jobId) is:

    - https://lcgrb01.jinr.ru:9000/WpF1fM-SIbg8J5u3SDnT7g

**********************************************************
```

Задание было успешно отправлено на счет в LCG, где ему был присвоен идентификатор

https://lcgrb01.jinr.ru:9000/WpF1fM-SIbg8J5u3SDnT7g ,

по которому можно отслеживать процесс прохождения задания «по инстанциям» счета. Например, получить текущий статус задания можно командой

```
~ > edg-job-status https://lcgrb01.jinr.ru:9000/
                          WpF1fM-SIbg8J5u3SDnT7g

*********************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://lcgrb01.jinr.ru:9000/
                                WpF1fM-SIbg8J5u3SDnT7g
Current Status:    Ready
Status Reason:     unavailable
Destination:       ce01.novsu.ac.ru:2119/jobmanager-lcgpbs-dteam
reached on:        Tue Oct 24 20:59:12 2007
*********************************************************
```

Задача проходит через ряд состояний (в примере выше Ready) и по завершении достигает полной готовности (статус Done). Только тогда можно получить и результат (т. е. загрузить файлы, перечисленные в параметре OutputSandbox, к себе на машину:

```
~ > edg-job-get-output https://lcgrb01.jinr.ru:9000/
                             WpF1fM-SIbg8J5u3SDnT7g

Retrieving files from host: lcgrb01.jinr.ru
  (for https://lcgrb01.jinr.ru:9000/WpF1fM-SIbg8J5u3SDnT7g)

**********************************************************
                      JOB GET OUTPUT OUTCOME
```

```
 Output sandbox files for the job:
 - https://lcgrb01.jinr.ru:9000/WpF1fM-SIbg8J5u3SDnT7g
 have been successfully retrieved and stored in the directory:
 /data/jobOutput/yupi_RVDXfI4A7Bx3jCzJlj3TUw

************************************************************
```

В результате работы последней команды полученные при счете файлы были загружены на машину пользователя (т. е. в нашем случае все описанные в `Hello.jdl` файлы были помещены в директорию `/data/jobOutput/...`):

```
~ >  ls -l /data/jobOutput/yupi_RVDXfI4A7Bx3jCzJlj3TUw/
total 4
-rw-r--r--   1 yupi lnup    0 Oct 25 00:48 Hello.err
-rw-r--r--   1 yupi lnup   13 Oct 25 00:48 Hello.out
```

при этом статус задания стал «Cleared». И вот, наконец, наш результат:

```
~ > cat /data/jobOutput/yupi_RVDXfI4A7Bx3jCzJlj3TUw/Hello.out
Hello, Grid!
```

ЗАКЛЮЧЕНИЕ И ПЕРСПЕКТИВЫ

«Вычислительный элемент» ЛЯП сегмента JINR-LCG2 работает уже почти два года. Первоначально на нем поддерживался счет задач всех основных экспериментов LHC (ALICE, ATLAS, CMS и LHCb). Однако в конце 2006 г. было принято решение о специализированном его использовании для так называемых «коротких задач» (не более 3 ч счетного времени) коллаборации ATLAS. Такое решение было обосновано тем, что на тот момент в кластер входило всего четыре однопроцессорных WN-машины, что позволяло одновременный счет до восьми заданий, поскольку на поддерживающем multithreading процессоре Pentium 4 возможен эффективный счет двух задач одновременно. Членам коллаборации ATLAS, в частности, работающим в ЛЯП (А. С. Жемчугов и др.), было нужно пусть и небольшое число, но достаточно доступных процессоров, не перегруженных другими Grid-заданиями.

Регулярно проводятся обновления программ как для всех компонент операционной системы и приложений, так и LCG-middleware. Часть обновлений идет в «автоматическом» режиме, но для ряда программ (в том числе самой пакетной системы счета задач и особенно LCG-системы, установка и настройка которой, увы, все еще далеки от совершенства) требуется кропотливая «ручная работа».

Созданная конфигурация машин кластера, в частности WN и UI, представляется вполне сбалансированным вариантом Linux-машины, удовлетворяющим достаточно широким требованиям пользователя как по разработке

программ и для работы с desktop-приложениями, так и по счету задач в пакетном режиме, либо с использованием возможностей Grid.

Общее количество «рабочих элементов» (WN) в кластере (даже с учетом, как уже было отмечено выше, трех двухпроцессорных машин в рабочих группах) все еще невелико, что позволяет говорить о нем, скорее, как о прототипе нового счетного кластера ЛЯП, который должен прийти на смену существующему старому кластеру лаборатории, состоящему из 10 двухпроцессорных машин на базе уже устаревших Pentium III 500 МГц). Весной 2007 г. в ЛИТ была расширена и радикально обновлена центральная часть сегмента JINR-LCG2. Сейчас в него входят 60 WN на базе двухпроцессорных 64-битовых Intel Xeon. Сегмент ЛЯП пока все еще ожидает таких перемен. Как вариант возможного развития кластера представляется разумным установка дополнительных новых WN в рабочих группах лаборатории по примеру уже упомянутых трех компьютеров. Планируется обновление и расширение кластера ЛЯП с одновременным переходом на современную 64-битовую архитектуру. Такой переход, однако, тоже не будет простым. Он создаст и новые проблемы, в частности, для работы LCG middleware: ведь до сих пор полноценной сертифицированной 64-битовой версии для SL4 нет, хотя именно этот вариант аппаратной конфигурации системы постепенно становится доминирующим. Одним из авторов (Ю. П. И) в настоящее время проводится изучение и решение возникающих при таком переходе проблем на базе кластера университета Вальпараисо (Чили), входящего в Grid-структуру EELA [9].

Благодарности. Авторы признательны своим коллегам по группе LGD и сотрудникам ЛИТ, создавшим сегмент LGD-2 в ОИЯИ, за помощь и поддержку, в особенности В. В. Мицыну, без советов и разъяснений которого разработка и поддержка Grid-кластера ЛЯП были бы куда более трудоемким процессом. Авторы благодарны В. А. Беднякову за постоянную моральную поддержку и за мотивацию к написанию самой этой работы, а также И. Н. Чурину за помощь в подготовке текста статьи.

ПРИЛОЖЕНИЕ. ГДЕ ИСКАТЬ ИНФОРМАЦИЮ ПО GRID

Приведем список основных ссылок как собственно по LCG, так и по Grid-вычислениям вообще, что тоже может оказаться полезным для начинающих работать с Grid. В качестве «предварительного» чтения можно ознакомиться с материалами на сайте

http://gridcafe.web.cern.ch ,

где основные идеи Grid-вычислений изложены на популярном уровне (обзор «с птичьего полета»), хотя с практической точки зрения они мало что дают. Основной сайт LCG

http://lcg.web.cern.ch/lcg

содержит исчерпывающую информацию по установке, администрированию и использованию LCG. Пользователям LCG можно начать со странички

http://lcg.web.cern.ch/lcg/users/support.html

В частности, здесь расположен и «LCG-2 User Guide». Несколько устаревшим, но все же еще полезным является «LCG-2 User Scenario» — краткое практическое руководство, в котором на простых примерах показано, как послать задание в LCG, проверить его состояние и получить файлы результата. Его можно найти, например, на страничке

https://edms.cern.ch/document/498081

В принципе, можно начать свою работу с LCG, ознакомившись лишь с этим текстом. Однако для более глубокого понимания принципов и возможностей LCG (особенно при необходимости обращения к функциям Grid из самих программ) его недостаточно. Многочисленную дополнительную документацию можно найти, например, на сайтах

http://lcg.web.cern.ch/lcg/documents.html
https://gus.fzk.de/pages/docu.php

Последняя ссылка требует наличия в браузере Grid-сертификата пользователя. Для «продвинутых» пользователей можно порекомендовать документцию

http://www.globus.org/alliance/publications/papers.php

где, в частности, можно найти работы [1] и [2], считающиеся «классическим» введением в Grid. Переводы этих текстов на русский язык, а также много другой полезной информации как на русском, так и на английском языках можно найти на сайтах

http://grid.jinr.ru
http://www.gridclub.ru

ЛИТЕРАТУРА

1. *Foster I., Kesselman C., Tuecke S.* // Intern. J. Supercomp. Appl. 2001. V. 15(3). P. 200–222.
2. *Foster I. et al.* Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002; http://www.globus.org/alliance/publications/papers/ogsa.pdf
3. *Foster I., Kesselman C.* // Intern. J. Supercomp. Appl. 1997. V. 11(2). P. 115–128; http://www.globus.org.
4. http://lcg.web.cern.ch.
5. http://grid.infn.it.
6. http://rus.egee-rdig.ru.
7. http://glite.web.cern.ch.
8. http://www-unix.mcs.anl.gov/mpi
9. http://www.eu-eela.org

Получено 7 мая 2008 г.

Редактор *Е. В. Сабаева*

Подписано в печать 27.06.2008.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 1,19. Уч.-изд. л. 1,73. Тираж 310 экз. Заказ № 56217.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@jinr.ru
www.jinr.ru/publish/