МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(национальный исследовательский университет)»

РАБОТА НА ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ BladeSystem HP c3000 В ГРИД-СРЕДЕ СГАУ И В ГРИД-СРЕДЕ «УНИВЕРСИТЕТСКИЙ КЛАСТЕР»

Методическое пособие

САМАРА
2010

УДК 535.42
ББК 22.343

Составители:

ПОПОВ Сергей Борисович, КАЗАНСКИЙ Николай Львович,

СЕРАФИМОВИЧ Павел Григорьевич

С момента своего первого появления в 1998 году в списке 500 самых быстрых компьютеров в мире Linux-кластеры прошли путь от неясного научного эксперимента до доминирующей силы в технологии суперкомпьютерных вычислений. При этом количество Linux-кластеров в списках Top 500 выросло от одной системы в 1998 году (1 кластер, 1 система под Linux) до четырех пятых списка в 2008 г. (400 кластеров, 458 систем под Linux).

Управление кластерами Linux требует набора навыков, которые обычно не встречаются среди ИТ-администраторов локальных систем или небольших сетей — оно требует всестороннего знания работы с сетями, операционных систем и практически всех подсистем архитектуры.

В данном руководстве даны базовые навыки работы на вычислительном кластере. Обсуждаются вопросы пакетного управления заданиями пользователей.

Пособие предназначено для студентов специальностей и направлений "Прикладная математика и информатика", "Прикладные математика и физика".

УДК 535.42
ББК 22.343

ОГЛАВЛЕНИЕ

1 Быстрый старт

1.1 Способы доступа пользователя на кластер

1.1.1 Удаленный доступ на кластер для компиляции и запуска расчетных программ пользователя

Пользователи операционной систем Linux могут воспользоваться стандартным ssh-клиентом. Пользователям Microsoft Windows рекомендуется использовать программу PuTTy (http://www.chiark.greenend.org.uk/~sgtatham/putty/ download.html)

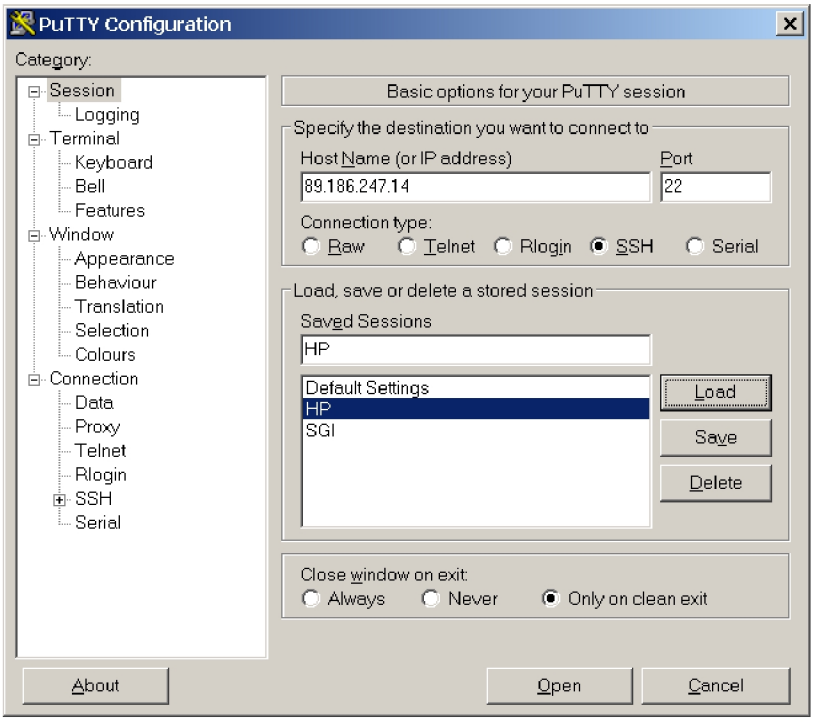

Рис. 1. Конфигурация программы PuTTy

При первом запуске программы в окне HostName набрать IP адрес кластера – 89.186.247.14. В поле Saved Sessions ввести любое удобное название (на рис.1 выбрано название «HP»).

В поле Category — Window — Translation — Character set translation on received data выбрать UTF-8 (см. рис. 2). Вернуться в окно Category — Session. Нажать кнопку «Save». Под именем HP будут сохранены настройки.

При последующих запусках PuTTy в окне конфигурации выбрать «HP» и щелкнуть кнопку “Load”. В окне Host Name появится IP адрес кластера – 89.186.247.14.

Нажать кнопку “Open”. Произойдет соединение с кластером, откроется окно терминала кластера. В этом окне сначала ввести имя пользователя, затем пароль.

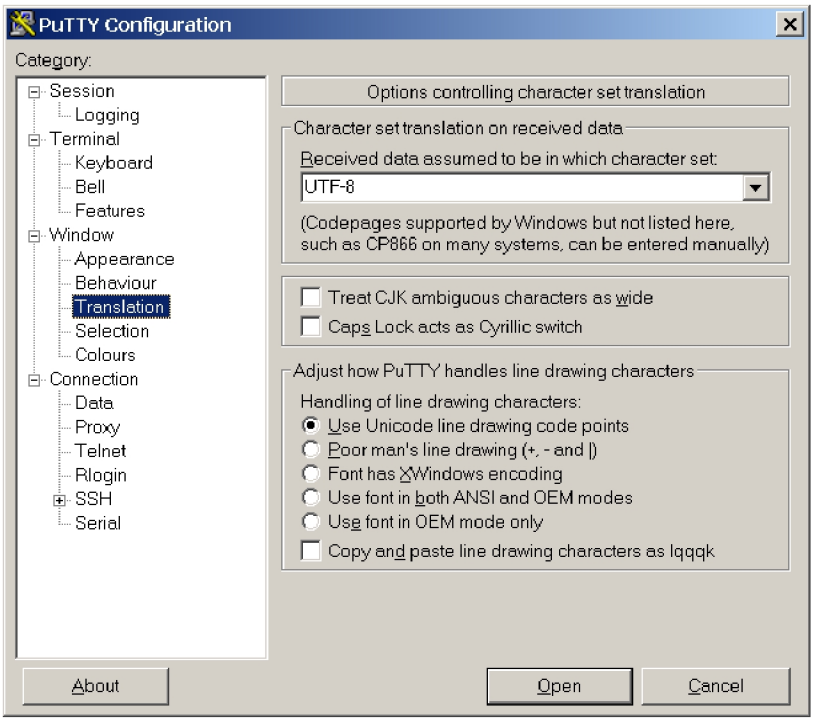

Рис. 2. Выбор таблицы кодировки символов

1.1.2 Удаленный доступ на кластер для копирования файлов между персональным компьютером пользователя и кластером

Сетевой файловый менеджер WinSCP может быть использован для файловых операций с удаленными и локальными файлами. Скачать программу WinSCP можно по адресу http://winscp.net.

При первом запуске программы WinSCP необходимо ввести в поле Host name IP адрес кластера – 89.186.247.14, а в поле User name имя пользователя (см. рис. 3). Поле Password можно не заполнять, его вы будете вводить при каждом следующем соединении. Введенные данные сохраняются кнопкой “Save”.

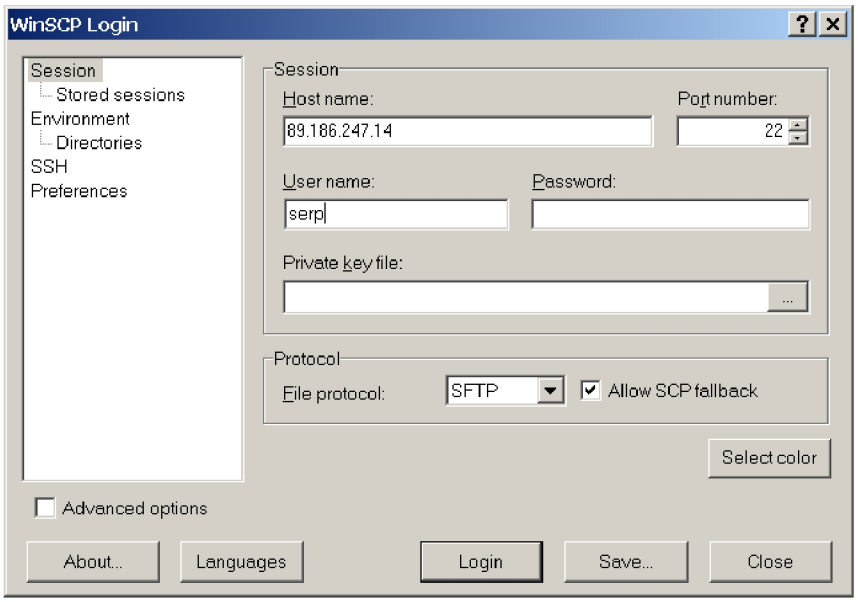

Рис. 3. Стартовое окно программы WinSCP при первом запуске

При последующих запусках программы окно WinSCP Login будет с заполненными личными данными (рис. 4).

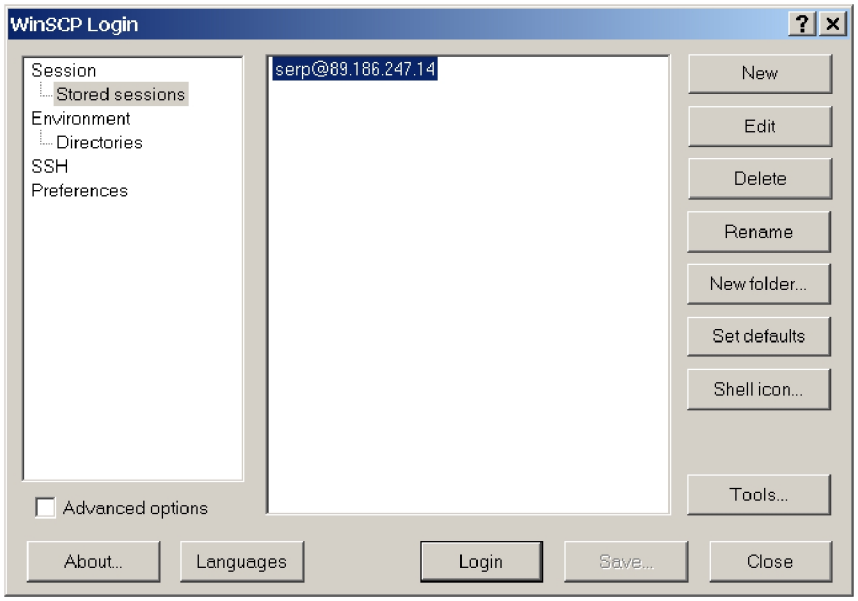

Рис. 4. Окно соединения

Соединение начинается выбором пункта “Login”. В окне Server prompt надо ввести свой пароль (рис. 5). Для соединения с удаленной ЭВМ необходимо время. Поэтому окно программы WinSCP появится с некоторой задержкой. В зависимости от варианта, выбранного при установке программы откроется окно, по внешнему виду сходное с Windows Explorer либо двухпанельное окно в стиле Total Commander. В этом окне будет отображена

файловая система кластера и вы можете копировать файлы на кластер и обратно также, как это делается в Windows (рис. 6). Поддерживаются операции Drag and Drop.

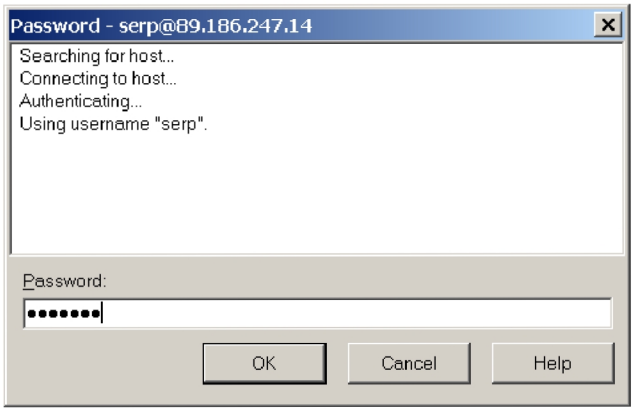

Рис. 5. Окно ввода пароля

Кроме того, программа WinSCP позволяет проводить другие операции с файлами:редактирование, переименование, удаление, изменение прав доступа и прочее.

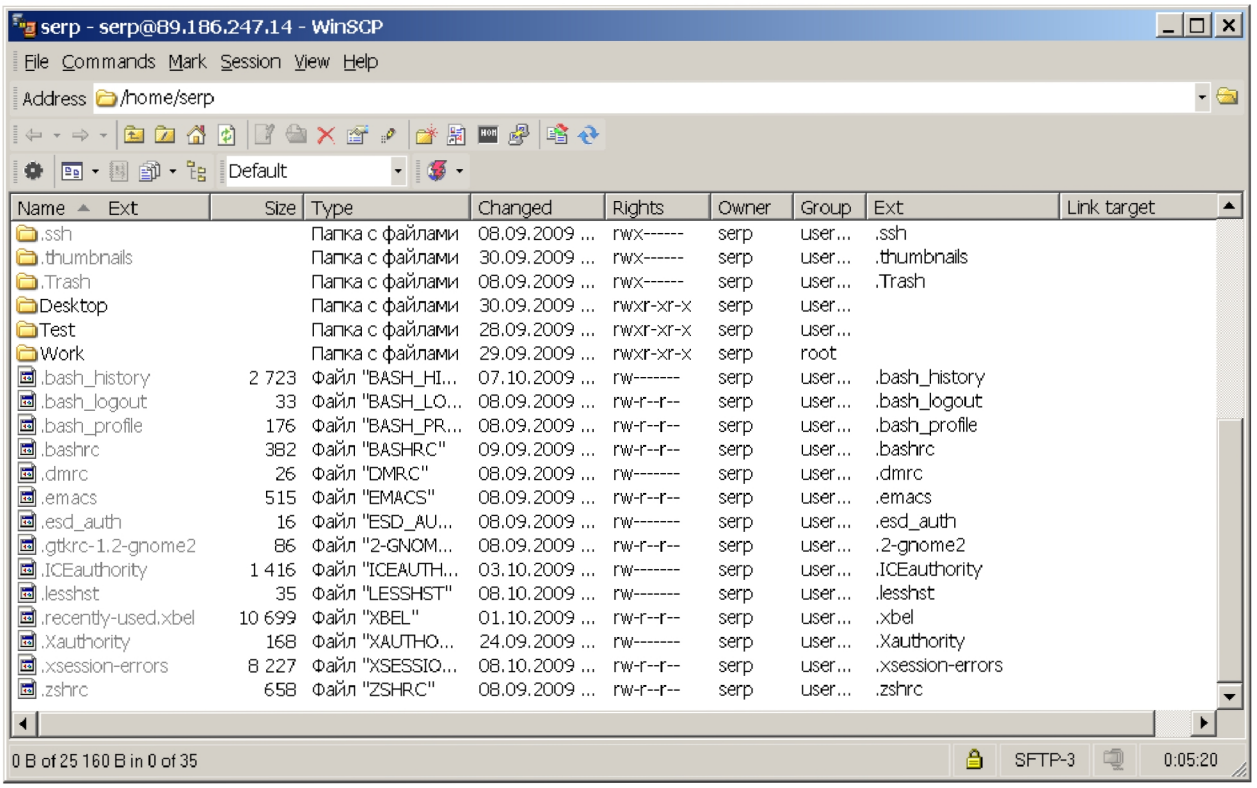

Рис. 6. Рабочее окно программы WinSCP

Завершение работы с программой и прекращение связи проводится клавишей F10 или выбором в меню пункта Commands/Quit. Требуется подтвердить окончание работы в окне завершения.

1.2 Настроить окружение выполнения MPI программы

Настроить окружение выполнения MPI программы можно с использованием команд утилиты Switcher.

1.2.1 Посмотреть текущее состояние

[tester@master ~]$ **switcher mpi --show**

user:default=lam-7.1.4 user:exists=1

1.2.2 Посмотреть список возможных настроек

[tester@master ~]$ **switcher mpi --list**

openmpi-1.2.4

mpich-ch_p4-gcc-1.2.7

lam-7.1.4

1.2.3 Установить окружение выполнения MPI программы

В примере будет установлено окружение mpich-ch_p4-gcc-1.2.7

[tester@master ~]$ **switcher mpi --add-attr default mpich-ch_p4-gcc-1.2.7**

Warning: mpi:default already has a value:

lam-7.1.4

Replace old attribute value (y/N)? y

Attribute successfully set; new attribute setting will be effective for

future shells

1.2.4 Перезагрузить программную оболочку

[tester@master ~]$ **bash**

1.2.5 Проверить установленные настройки

[tester@master ~]$ **switcher mpi --show**

user:default=mpich-ch_p4-gcc-1.2.7

user:exists=1

и/или

[tester@master ~]$ **which mpirun**

/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun

и/или

[tester@master ~]$ **cexec which mpirun**

********** *oscar_cluster* **********

--------- n1---------

/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun

--------- n2---------

/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun

--------- n3---------

/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun

--------- n4---------

/opt/mpich/1.2.7/ch_p4/gcc/bin/mpirun

1.3 Подготовка исполняемого файла MPI приложения

1.3.1 Редактирование текста программ пользователя

Процесс разработки программного обеспечения, независимо от размеров и предназначения программы содержит этап редактирования исходных текстов программы. Конечно, вы можете изменять свои программы на рабочем компьютере, а потом копировать их на кластер. Также вы можете редактировать файлы прямо на кластере. Для этого на кластере имеется несколько текстовых редакторов разной степени удобства: vim, ed, emacs,

joe. Наиболее простым для освоения, на наш взгляд, является редактор, встроенный в оболочку MidnightCommander.

Для запуска этой программы используется команда mc.

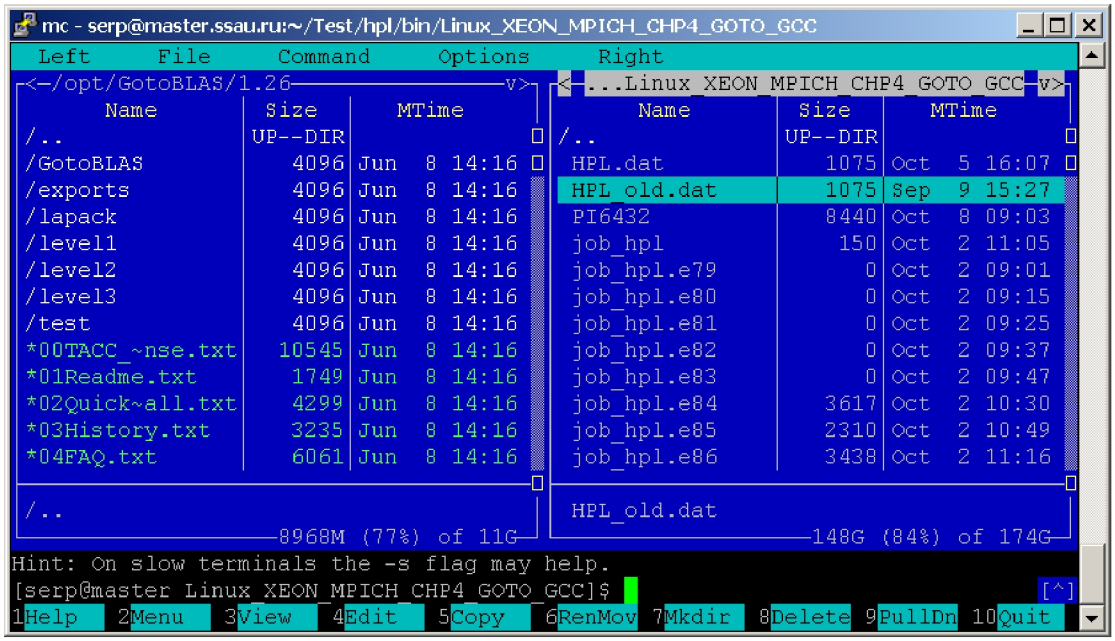

Рис. 7 Окно оболочки MidnightCommander

Midnight Commander — это файловый менеджер с текстовым интерфейсом. Его предназначение — упростить основные действия пользователя, связанные с управлением файлами. Принцип работы Midnight Commander такой же, как и у Far Manager, TotalCommander, или Norton Commander. Экран состоит из двух панелей, в которых отображается список файлов и каталогов в выбранных каталогах, и пользователь может выполнять некоторый набор действий над этими файлами. В нижней части экрана расположена командная строка и панель горячих клавиш F1-F10. Можно вызвать верхнее меню, нажав клавишу F9. Одна из панелей всегда является активной, а в ней курсор установлен на активный файл. Пользователь может выполнять действия либо с активным файлом или каталогом, либо групповые операции со всеми объектами активной панели. Также доступны некоторые общие операции: поиск файлов, помощь по работе с MC, выполнение команд операционной системы и т.д.

Для редактирования файла необходимо клавишами управления курсором выбрать нужный файл, и нажать клавишу F4. Запустится редактор текстовых файлов, выйти из которогоможно, нажав клавишу F10 либо Esc. Чтобы сохранить изменения, необходимо нажать клавишу F2, либо выбрать нужный вариант при выходе из редактора.

Для создания нового файла нужно нажать Shift+F4 (при нажатой клавише Shift нажмите клавишу F4). Откроется окно редактора, и при сохранении изменений программа предложит ввести имя сохраняемого файла.

1.3.2 Компиляция программ пользователя

Для компиляции и сборки параллельных программ используются следующие утилиты:

- *mpicc* — для программ написанных на языке программирования C;
- *mpixx (mpiCC)* — для программ написанных на языке программирования C++;
- *mpif77 и mpif90* — для программ написанных на языке программирования Fortran.

Синтаксис данных утилит во многом похож на синтаксис компилятора ***gcc***, более полная информация о синтаксисе доступна по команде **`имя_утилиты --help`** (напр., **`mpif77 –help`**).

Например, для обработки программы myprog.c можно выполнить команду

```
mpicc -o myprog -lm myprog.c
```

Ключ `-o` указывает, что результат компилирования должен быть помещен в файл `myprog`. Если мы просто выполним команду

```
mpicc  -lm myprog.c
```

то в текущем каталоге появится исполняемый файл `a.out`.

Если программа представлена не одним файлом, а несколькими исходными и заголовочными файлами, то для сборки можно использовать Makefile или shell-скрипт.

1.4 Запустить MPI приложение на кластере

Выделение ресурсов и запуск приложений на кластере обеспечивает система пакетной обработки заданий Torque и менеджер ресурсов MAUI. Поэтому для того чтобы запустить MPI приложение на кластере необходимо выполнить следующее:

- подготовить PBS-задание

- поставить PBS-задание в очередь на выполнение.

1.4.1 Подготовка PBS-задания

PBS–задание это некоторый скрип написанный на языке командного интерпретатора, которой содержит как директивы для самой системы пакетной обработки на выделение ресурсов, так и директивы для запуска задачи пользователя.

Пример. Для запуска MPI программы PBS- задание может быть оформлено следующим образом:

```
#PBS -l walltime=00:30:00,nodes=4:ppn=8
#PBS -q workq@master
#PBS -N job_name
#PBS -o /home/tester/out
#PBS -e /home/tester/err
#!/bin/sh
cd  /home/tester/hpl/bin/Linux_XEON_MPICH_CHP4_GOTO_GCC/
mpirun -np 32 -machinefile $PBS_NODEFILE ./xhpl
```

Значение параметров PBS задания смотрите в соответвующем разделе данного пособия.

1.4.2 Постановка PBS-задания в очередь на выполнение

После того как PBS задание готово, его необходимо поставить в очередь на выполнение командой

qsub [имя PBS-задания]

[tester@master ~]$ **qsub job_hpl**

1.4.3 Мониторинг запущенного задания

Мониторинг очереди заданий может быть выполнен с использованием терминальных команд системы пакетной обработки Torque или менеджера ресурсов MAUI.

1.4.4 Состояние очереди заданий

[tester@master ~]$ **qstat**

1.4.5 Полная инфомация по заданию

qstat -f [Job ID]

1.4.6 Информация о состоянии очереди заданий от менеджера ресурсов

[tester@master ~]$ **/opt/maui/bin/showq**

1.4.7 Полная информация по узлам кластера

[tester@master ~]$ **pbsnodes**

2. Система пакетной обработки заданий torque

2.1 Обзор torque

В этой главе описываются общие принципы, которые реализует система пакетной обработки заданий torque. Рассматриваются ее компоненты, представление физических вычислительных узлов в системе, понятие задания и ресурса.

2.1.1 Общая характеристика

Torque - одна из версий системы PBS (Portable Batch System - система пакетной обработки заданий). Torque управляет загрузкой вычислительных комплексов, состоящих из определенного количества вычислительных узлов, работающих под управлением операционной системы семейства Unix.Система пакетной обработки заданий (далее - СПО) необходима при одновременном выполнении заданий (jobs) несколькими пользователями на одном вычислительном комплексе.

В результате применения СПО вычислительные ресурсы используются оптимально: сводится к минимуму как перегрузка какого-либо одного узла (об узлах см. далее по тексту), так и его простой. Torque обычно применяется в областях, где высока интенсивность использования вычислительных мощностей.

Таким образом, torque обеспечивает контроль над вычислительными ресурсами, что, в конечном итоге, снижает зависимость от системных администраторов и операторов, освобождая их для решения других задач. Также torque дает возможность контролировать выполнение заданий, используя очереди и планировщик заданий.

Принцип работы torque заключается в следующем. Задания создаются и управляются сервером заданий. Клиенты СПО взаимодействуют с сервером заданий, который предоставляет соответствующие сервисы. Пользователь взаимодействует с СПО посредством утилит командной среды. Сервер заданий является демоном (daemon), который осуществляет

постановку заданий в очередь, управление очередями и выполнение задания от имени клиента СПО. Сервисы, предоставляемые сервером заданий, доступны посредством утилит командной строки (batch utilities), которые запускают пользователи. В следующем разделе описаны компоненты torque.

Утилиты torque можно запускать как из командной строки операционной системы, так и посредством графического интерфейса. Набор, синтаксис и семантика (т.е. выполняемые операции) пакетных утилит соответствуют стандарту POSIX 1003.2d. Графический интерфейс в настоящем руководстве рассматриваться не будет.

2.1.2 Структура torque

В этом разделе рассматриваются основные компоненты torque и их взаимосвязь. Схема компонентов представлена на рис. 8.

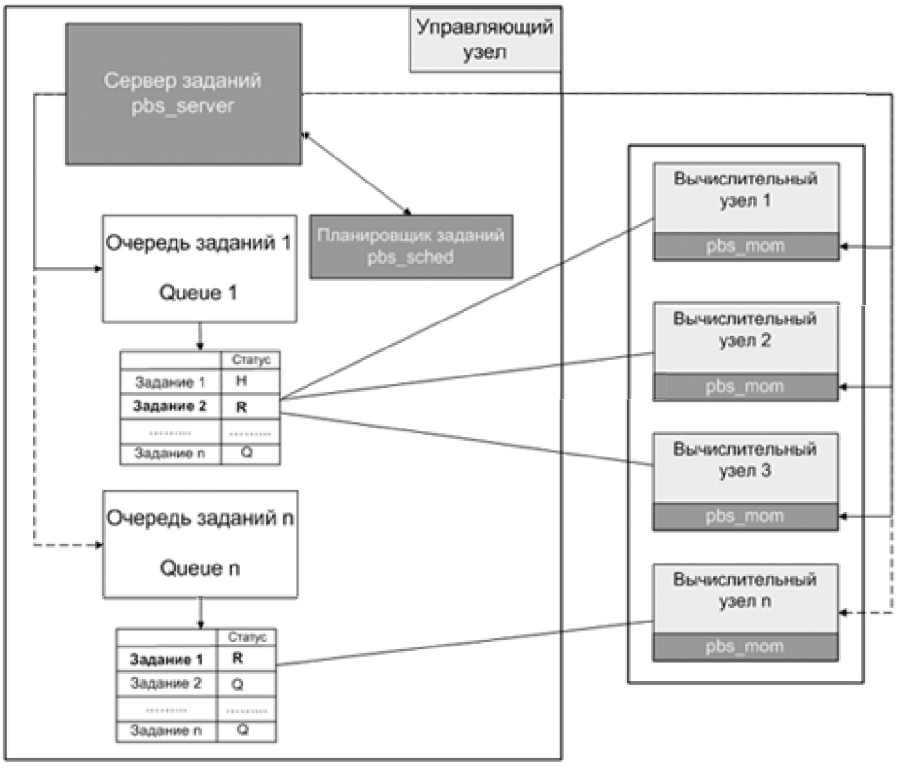

Рис. 8 Взаимодействие компонентов torque

3.1.2.1 Сервер заданий

Демон сервера заданий (Job Server daemon) – центральная точка torque. В настоящем Руководстве будет применяться термин scepвepї или имя процесса pbs_server. Все команды и другие процессы-демоны взаимодействуют с сервером посредством сети по протоколу IP. Главная задача сервера - обеспечить базовые сервисы для исполнения пакетных заданий, такие как получение/создание задания (batch job), изменение задания, обеспечение надежности функционирования системы заданий путем защиты от неполадок в системе, и исполнение задания.

Обычно имеется единственный сервер, управляющий конкретным набором ресурсов. Однако в общем случае серверов может быть несколько.

2.1.2.2 Сервис вычислительного узла: процесс pbs_mom

Клиент, запускающий задания (Job Executor; или просто – клиент заданий) – служба (service) операционной системы, физически осуществляющая запуск задания. Эта служба, называемая pbs_mom, неформально обозначается MOM, поскольку является snpедкомї (mother) всех исполняемых заданий. MOM запускает задание, когда получает его копию с сервера заданий. MOM создает новый сеанс от имени одного из пользователей (которым не является root), зарегистрированных в системе. Запуск задания производится в сеансе оболочки данного пользователя. Например, если пользователь работает в оболочке csh, то MOM создает сеанс, где запускается как файл .login, так и файл .cshrc. MOM также ответственен за предоставление пользователю результата работы задания, по умолчанию выводимого на консоль сервера. Этот результат может быть сохранен в нужном месте. Сервис MOM запускается на вычислительном узле (или узлах), который будет выполнять задание.

2.1.2.3 Представление вычислительных узлов

Вычислительные узлы кластера представляются в torque определенным образом. Прежде чем обсуждать работу с вычислительными узлами (compute nodes) кластера, необходимо ввести некоторые определения.

Вычислительный узел (compute node) – это отдельная компьютерная система (или просто компьютер) с одним образом (image) операционной системы, унифицированным виртуальным адресным пространством, одним или более процессором и одним или более IP-адресом. Часто термин исполняющий хост (execution host) также используется для обозначения узла. Компьютер, содержащий несколько процессоров и работающий под управлением одной операционной системой, является одним узлом. Узлы делятся на два типа: узлы общего типа (cluster node) и разделяемые по времени узлы (timeshared).

Узел общего типа (cluster node) – узел, назначением которого является параллельное выполнение заданий. Если такой узел имеет более одного виртуального процессора, они могут быть назначены разным заданиям (англ. jobshared – распределены между заданиями) или использованы для выполнения единственного задания (англ. exclusive – эксклюзивный доступ).

Такая возможность непрерывно распределять ресурсы каждого узла важна для некоторых приложений, работающих одновременно на нескольких узлах (multi-node applications). Обратите внимание, что torque обязывает придерживаться схемы sодин-к-одномуї при выделении виртуальных процессоров (см. далее) заданию. Таким образом, один ВП работает только с одним заданием.

Разделяемый во времени узел (timeshared node). В противоположность узлам общего типа, такие узлы всегда могут обслуживать несколько заданий одновременно. Часто термин sхостї (host) используется вместо термина sузелї (node) совместно с термином “timeshared”. Разделяемый во времени узел никогда не будет эксклюзивно выделен для выполнения единственного задания.

Виртуальный процессор (VP – virtual processor; далее ВП). Для узла может декларироваться наличие одного или нескольких виртуальных процессоров. Слово sвиртуальныйї используется, поскольку обозначенное число виртуальных процессоров может не соответствовать числу реальных процессоров в узле. Число ВП в узле по умолчанию есть число реально функционирующих ядер физических процессоров.

Атрибуты узла

Вычислительные узлы кластера настраиваются в torque установкой атрибутов. Атрибуты также используются в файле конфигурации узлов (см. раздел 2.1.2.3). Список основных атрибутов приведен в этом разделе. Установка атрибутов командой qmgr описывается в разделе 2.2.2.1.

comment

Комментарий для узла.

max_running

Максимальное количество заданий, которое может быть одновременно запущено на узле.

max_user_run

Максимальное число заданий, принадлежащих одному пользователю, которое допускается одновременно выполнять на узле.

no_multinode_jobs

Если этот атрибут имеет значение true, то задания, которые запрашивают для своего запуска несколько узлов, не будут выполняться на данном узле.

np

Количество виртуальных процессоров.

ntype

Задает тип узла. Типы узлов описаны выше, в предисловии к данному разделу. Значения могут быть следующими: *time-shared, cluster*.

properties

Свойства узла, определяемые пользователем. Значением может быть любая строка, начинающаяся с буквенного символа или такие строки, разделенные запятой.

resources_available

Ресурсы, доступные на узле. Конкретный ресурс задается после символа точки ".":

resources_available.ncpus. Соответственно, все ресурсы, доступные torque, можно указывать в качестве значения этого атрибута. Понятие ресурсов описывается в разделе 2.1.4.

state

При помощи этого атрибута можно задать или просмотреть статус (state) узла. Возможные значения для состояния: *free, offline, down, job_busy, job_exclusive, busy, state_unknown*. Первые два состояния может установить пользователь, остальные – только системные процессы.

Файл конфигурации узлов

Вычислительные узлы, где запускаются задания, определяются при взаимодействии сервера и других компонентов torque (в частности, планировщика заданий, см. раздел 2.1.2.4). Взаимодействие возможно благодаря файлу конфигурации nodes. Файл располагается по следующему пути:

PBS_HOME/server_priv/nodes

В файле содержится список узлов и их атрибуты. Без списка узлов сервер не сможет создать взаимодействовать с MOM посредством специального потока (communication stream). У MOM также не будет возможности отчитываться о запущенных заданиях и уведомлять сервер о завершении задания. Здесь *PBS_HOME* - переменная окружения, содержащая путь к рабочей директории torque. Простой файл конфигурации узлов создается в процессе установки torque. Этот файл содержит только название хоста, с которого была запущена инсталляция. Этот узел будет считаться разделяемым по времени (timeshared). Файл конфигурации узлов можно изменять двумя путями. Если сервер не запущен, это можно делать напрямую в текстовом редакторе. Если же сервер работает, следует использовать команду qmgr для изменения списка узлов.

Файл конфигурации представляет из себя обычный текстовый файл, каждая строка которого записана в форме:

node_name[:ts] [attributes]

Здесь *node_name* - это сетевое имя узла. Опциональный параметр "*:ts*" добавляется к имени, указывая таким образом, что узел является разделяемым по времени (timeshared). Также узлы могут иметь ассоциированные с ними атрибуты. Атрибуты перечисляются в виде:

attribute_name=value

Например, выражение np=<число> может быть использовано для определения числа виртуальных процессоров на узле. Если это выражение не указано для узла общего типа (cluster node), то число виртуальных процессоров будет равно 1. Пример файла содержит Листинг 5.

Листинг 5

```
master:- #cat /var/torque/server_priv/nodes
node-1 np=4
```

node-2 np=4

node-3 np=4

node-4 np=4

node-5 np=4

node-6 np=4

node-7 np=4

node-8 np=4

node-9 np=4

node-10 np=4

master:- #

Для вывода сведений об узлах используется команда pbs_nodes.

2.1.2.4 Планировщик заданий

Демон планировщика заданий (Job scheduler daemon), процесс которого называется *pbs_sched*, занимается распределением ресурсов между заданиями. Он определяет, когда данное задание будет запущено и какие ресурсы ему будут выделены. Планировщик взаимодействует с MOM на узлах, запрашивая у них состояние системных ресурсов; а также с сервером заданий для получения списка заданий, доступных для выполнения. Планировщик использует файл конфигурации узлов для определения узла или узлов, где будет запущено задание.

2.1.3 Понятие задания

Задание torque представляет собой абстрактную сущность, состоящую из набора команд и параметров. Задание представляется пользователю в виде скрипта для оболочки (shell),

содержащего требования к ресурсам, атрибуты задания и набор команд, которые необходимо выполнить.

Единожды создав скрипт задания, им можно пользоваться столько раз, сколько необходимо, также возможна его модификация. Задание сначала необходимо поставить в очередь torque (submit), затем из этой очереди оно будет передано на один узлов для выполнения. Очередей заданий (batch queue) может быть несколько. После установки torque очередей заданий не существует. Необходимо сначала создать очередь заданий, а затем уже ставить их в эту очередь.

Настроенный вариант системы включает одну очередь заданий.

Задание может быть обычным (regular) и интерактивным (interactive). Обычное задание ставится в очередь и затем ожидает своего выполнения, результат будет записан в указанное пользователем место. Интерактивное задание отличается тем, что потоки ввода и вывода перенаправляются соответственно на экран и клавиатуру, соответственно, команды задания вводятся с клавиатуры непосредственно. Об интерактивных заданиях более подробно будет рассказано далее.

2.1.3.1 Пример задания

Вот пример простого скрипта задания:

```
1 #!/bin/sh
2 #PBS -l walltime=1:00:00
3 #PBS -l mem=400mb
4 #PBS -l ncpus=4
5 #PBS -j oe
6
7 ./subrun
```

Первая строка является стандартной для любого скрипта с описанием задания, она определяет, какая оболочка используется для исполнения сценария. Оболочка sh используется поумолчанию для запуска сценария, но можно использовать и другую. Строки со 2-й по 5-ю являются директивами torque. Система будет читать скрипт до тех пор, пока не найдет первую строку, которая не является валидной директивой torque, и останавливается. Это означает, что оставшаяся часть сценария содержит список команд или задач, которые пользователь желает запустить. При выполнении данного примера, torque обнаружит такие команды в строках 6 и 7.

Далее будет приведено описание команды qsub, выступающей в роли командного интерпретатора. Она используется в том числе для постановки задания в очередь torque. Любая опция, которую определяется в команде qsub, может также выступать в роли директивы внутри скрипта torque.

Строки 2–4 определяют опцию ресурса “–l”, далее следует запрос определенного ресурса. Конкретно, строки 2–4 сообщают, что запрашиваются ресурсы, объем которых предполагает: не более 1 часа на выполнение, а также 400 Мб памяти и 4 процессоров.

Строка 5 не является директивой запроса ресурса. Опция “–j oe” требует, чтобы torque объединила (join) потоки вывода stdout и stderr в единый поток stdout.

И наконец, строка 7 является командой для выполнения, которую пользователь хочет запустить. Данный пример запускает программу *subrun*. Хотя в примере имеется только одна команда, можно добавить необходимое количество программ, заданий и шагов.

2.1.3.2 Понятие атрибутов задания

Задание обладает определенным набором атрибутов, значения которых задаются изначально при создании задания и могут быть изменены в процессе его выполнения. Примеры атрибутов задания - название задания, время выполнения, путь к выходному файлу и др.

Атрибуты задания задаются при постановке задания в очередь. Также их можно изменить уже после этого по различным причинам (например, была сделана ошибка при определении ресурсов или истекло время выполнения задания и его необходимо продлить). Независимо от причины изменения атрибутов, для этого имеется команда *qalter*.

Большинство атрибутов может изменить владелец задания, которым может быть пользователь, поставивший задание в очередь командой qsub, либо произвольная учетная запись, указанная при постановке в очередь. Тем не менее, если задание уже выполняется, то лимиты ресурсов не могут быть изменены. Такими ными ресурсами являются: процессорное время, обычное время, число задействованных процессоров, объем памяти.

Примером атрибутов задания может служить название задания, его идентификатора, желаемое время начала выполнения задания.

2.1.3.3 Очереди заданий

Очередь torque - это сущность, содержащая задания. Torque поддерживает два типа очередей:

1. Исполняемая очередь (execution queue), содержащая задания, готовые для выполнения. Задания могут запускаться только из очередей этого типа.
2. Очередь перемещения (routing queue), в которой находятся задания, предназначенные для перестановки в другие очереди, в том числе те, которые находятся на других серверах заданий.

Очередей обоих типов может быть несколько. Также очереди имеют свои атрибуты. Более подробно об атрибутах см. в Руководстве администратора torque.

2.1.3.4 Возможные состояния задания

Задания в очередях могут находиться в различных состояниях. Возможные состояния перечислены далее.

Сокращение Описание	
C	complete; Задание успешно завершило свою работу
E	exit; Прерывание работы задания
H	hold; Задание заблокировано
Q	queued; Задание поставлено в очередь и готово для выполнения
R	running; Задание выполняется
T	transiting; Задание перемещается в другое место (очередь)
W	waiting; Задание ожидает, пока подойдет очередь для его выполнения, например, задание может ожидать определенного времени для своего выполнения или завершения выполнения другого задания, от котого зависит. Задание в этом состоянии не может быть выполнено
S	suspended; Пауза в работе задания

2.1.4 Понятие ресурса. Типы ресурсов, управляемых torque

Задание может запросить для запуска множество разных ресурсов, таких как процессоры, память, время (обычное и процессорное). Также может понадобиться дисковое пространство. Список ресурсов определяется с использованием опции –l список_ресурсов команды qsub или в скрипте задания. Таким образом выделяются ресурсы, необходимые для выполнения

задания или определяется их лимит, который может быть выделен. Если лимит не устанавливается для какого-либо ресурса, то он считается равным бесконечности.

Аргумент список_ресурсов записывается в виде:

resource_name[=[value]][,resource_name[=[value]],...]

Здесь resource_name – название ресурса, value – значение. Значения могут представляться в нескольких единицах измерения, зависящих от природы самого ресурса (например, время записывается в соответствующем формате [[часы:минуты]:секунды[.миллисекунды]; размер памяти указывается в байтах - b, килобайтах - kb, мегабайтах - mb).

Ресурсы могут быть следующих видов: количество процессоров, объем памяти, требуемое ПО, объем виртуальной памяти, количество времени и др.

2.2 Настройка torque

СПО предоставляет вычислительную среду, абстрагирующую пользователя от физической (аппаратной) реализации вычислительного кластера. Пользователь определяет задания, которые необходимо выполнить. Система в нужный момент принимает решение о запуске и возвращает результаты операции. Если все доступные узлы заняты, то СПО ожидает, пока ресурсы будут доступны. С точки зрения torque, задание сначала создается, а затем ставится в очередь.

Еще один пример задания содержит Листинг 6. Обратите внимание, что опции всех команд чувствительны к регистру.

Листинг 6

```
test@master:-> cat ./test.script
#!/usr/bin/csh
```

```
#PBS -d /home/test
#PBS -l ncpus=4
#PBS -N Hostname
/bin/hostname
test@master:->
```

2.2.1 Взаимодействие torque с пользовательской средой

Чтобы системная среда надлежащим образом взаимодействовала с torque, необходимо проверить несколько моментов. В большинстве случаев среда настраивается системным администратором.

Чтобы torque работала правильно, необходимо выполнение следующих условий:

- необходимо, что все скрипты запуска оболочки были корректными;
- пользователь должен иметь учетную запись, отличную от root всех на вычислительных узлах (подробности см. в следующем разделе).

2.2.1.1 Настройка пользовательской среды на примере оболочки csh

В выполнении пользовательского задания могут возникнуть сложности, если скрипты запуска пользовательской оболочки (например, для оболочки csh - это файлы .cshrc, .login или .profile; для оболочки bash - .bashrc) содержат команды, которые пытаются использовать стандартные потоки. Подобная последовательность команд в таких файлах должна быть пропущена путем проверки переменной окружения PBS_ENVIRONMENT. Вот пример использования подобной методики в файле .login:

```
...\\
setenv MANPATH /usr/man: /usr/local/man :$MANPATH
if ( ! $?PBS\_ENVIRONMENT ) then
    использование стандартных потоков ( например , вывод на консоль )
endif
```

Нужно внимательно относиться к тем командам в сеансовых файлах пользователя, которые выводят на консоль какой-либо текст при работе в torque. Как и в предыдущем примере, команды, которые выводят текст в поток stdout, не должны быть выполнены при запуске через torque. Это достигается так же, как и в приведенном примере с файлом .login, а именно:

```
...\\
setenv MANPATH /usr/man: /usr/local/man :$MANPATH
if ( ! $?PBS\_ENVIRONMENT ) then
    команды , выполняющие стандартный вывод
endif
```

При запуске задания torque, «выходное состояние» (“exit status”) последней команды, выполненной в задании, являющееся отчетом для оболочки выполнения, будет таковым и для torque. Это важно для зависимых заданий и для построения цепочек заданий. Однако последняя исполненная команда может не быть последней командой в задании. Такое может иметь место, если задание выполняется в оболочке csh на хосте и там имеется файл .logout. В данной ситуации последняя команда, выполненная из файла .logout, не является командой задания.

Чтобы предотвратить это, необходимо сохранить выходное состояние в файле .logout путем запоминания его в начале файла, а затем выполнить выход с этим статусом в конце, как показано ниже:

```
set EXITVAL = $status
    содержимое файла .logout
exit $EXITVAL
```

2.2.1.2 Переменные окружения

Для задания в системе torque существует некоторое количество переменных окружения. Одни переменные берутся из пользовательской среды и передаются заданию, другие создаются самой torque, третьи могут явно создаваться пользователем для эксклюзивного использования заданием torque.

Примечание Переменные окружения torque существуют в сеансе, создаваемом командой qsub. В обычной оболочке, из которой происходит запуск qsub, эти переменные не видны.

Все переменные, существующие в задании, имеют имена, начинающиеся с “PBS_”. Некоторые из них также предваряются заглавной O: “PBS_O_”, что говорит о происхождении переменной из среды выполнения задания (например, пользовательской).

Далее приведен короткий пример, демонстрирующий использование наиболее полезных переменных и их типичных значений:

```
PBS_O_HOME=/home/test
PBS_O_LOGNAME=test
PBS_O_PATH=/usr/new/bin:/usr/local/bin:/bin
PBS_O_WORKDIR=/share/hpl/bin/
```

Полный список переменных окружения torque приведен в приложении А.

2.2.2 Команды настройки torque

В этой главе описываются команды настройки torque, такие как qmgr, pbsnodes.

3.2.2.1 Настройка узлов с помощью команды qmgr

Команда qmgr предоставляет пользователю интерфейс взаимодействия с сервером заданий torque. Эта команда позволяет настраивать узлы и их атрибуты. Qmgr можно также как интерактивный интерфейс к менеджеру torque.

Описание команды qmgr

Команда считывает директивы из стандартного потока ввода, синтаксис директив проверяется и соответствующий запрос отсылается к одному или нескольким серверам заданий. По умолчанию команду может выполнять только пользователь root.

Синтаксис команды qmgr таков:

qmgr [–a] [–c command] [–e] [–n] [–z] [server...]

Опции команды описываются далее.

Опции команды qmgr

–a	Прервать работу qmgr в случае любых синтаксических ошибок или запросов, отклоненных сервером.
–c < “команда” >	Выполнение единственной команды и

	завершение работы qmgr.
–e	Перенаправить эхо-вывода в стандартный поток.
–n	Только проверка синтаксиса, без выполнения команд.
–z	Не выводить сообщения об ошибках в стандартный поток ошибок.

Если команда qmgr запускается без опции –c и стандартный поток вывода ассоциирован с терминалом, qmgr выведет приглашение и директивы будут считываться с клавиатуры (стандартного потока ввода).

Создание и удаление узлов

Указав опцию –c при выполнении qmgr, можно задать команду создания нового или удаления существующего узла. Указанные операции необходимо при помощи qmgr всегда, если процесс pbs_server запущен.

Для добавления нового узла используйте подкоманду screateї команды qmgr:

qmgr "create node node_name [<атрибут>=<значение>]"

Например:

qmgr -c "create node node003"

Для изменения параметров узла после создания узла используйте подкоманду set:

set node node_name [attribute[+|-]=value]

Символы “+” и “-” следует использовать, если атрибут допускает несколько значений.

Для удаления узлов используется подкоманда delete:

Qmgr -c "delete node mars"

Qmgr -c "delete node Pluto"

Примеры работы с командой qmgr

Далее приводятся примеры работы с qmgr. Листинг 7 содержит пример вывода информации об объектах типа sserverї. В этом примере qmgr используется с опцией –c.

Листинг 7

```
master:- # qmgr -c "list server"
Server master.localdomain
server_state = Active
scheduling = True
total_jobs = 0
state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
default_queue = batch
log_events = 511
mail_from = adm
scheduler_iteration = 600
node_check_rate = 150
tcp_timeout = 6
pbs_version = 2.1.8
master:- #
```

В листинге 8 приводится пример изменения типа узла в интерактивном режиме команды qmgr:

Листинг 8

```
master:- #qmgr
Max open servers: 4
Qmgr: set node node-32 ntype=time-shared
```

Qmgr:

Листинг 9 приводит пример выполнения команды “list queue” для вывода информации об объекте типа “queue” (очередь), который называется “batch”:

Листинг 9

```
test@master:-> qmgr -c "list queue batch"
Queue batch
queue_type = Execution
total_jobs = 0
state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
mtime = Tue Sep 4 15:36:09 2007
enabled = True
started = True
```

2.2.2.2 Команда pbsnodes

Команда pbsnodes может быть использована для получения сведений об узлах и изменения их состояния.

Синтаксис команды pbsnodes следующий:

pbsnodes [-a|-l|-s][-c узлы][-d узлы][-o узлы][-r узлы][узел1 узел2 . . .]

Пример запуска команды без опций содержит Листинг 10 :

Листинг 10

```
master:- #pbsnodes
node-1
state = down
np = 4
ntype = cluster
```

node-2

state = down

np=4

ntype = cluster

Далее приводится описание опций команды.

узел1 узел2 ...	Если указаны только наименования узлов без дополнительных опций, то выводится состояние этих узлов.
–a	Выводит список всех узлов и значения каждого атрибута узла.
–c узлы	Изменяет состояние down или offline на free, т.е. узел становится доступным для выполнения заданий.
–d узлы	Устанавливает состояние down для указанных узлов. Эти узлы будут в дальнейшем не доступны для выполнения заданий. Если команда приводится без списка узлов, то все узлы переводятся в состояние down.
–l	Выводит список всех узлов.
–o узлы	Переводит указанные узлы в состояние offline, даже если они на данный момент используются. Это состояние не может быть изменено никакими автоматическими, не зависимыми от пользователя, средствами. Листинг 7 содержит результат выполнения команды с этой опцией.
-r узлы	Отменяет перевод в состояние offline для указанных узлов.
-s	Определяет сервер заданий, на который

	будет послан запрос.

Листинг 11

```
master:- #pbsnodes -o node-1
master:- #pbsnodes
node-1
state = down, offline
np = 4
ntype = cluster
node-2
state = down
np=4
ntype = cluster
```

2.3 Использование torque

В этой главе описывается команды, предназначенные для взаимодействия с пользователем, запускающим задания и контролирующим их выполнение. Рассматривается команда постановки задания в очередь qsub, а также команды изменения состояния задания qhold и qrls. Даются сведения о команде получения информации о заданиях qstat.

2.3.1 Команда qsub

В этом разделе рассмотрена команда qsub, предназначенная для постановки задания с различными параметрами в одну из существующих очередей.

Допустим, что ранее скрипт с описанием задания находится в файле под названием test.script.

Поставим в очередь соответствующее задание, используя команду qsub. Листинг 12 содержит пример создания задания.

Листинг 12

```
master:/home/test #su test
test@master:~> qsub ./test.script
57.master.localdomain
test@master:->
```

После успешной постановки в очередь задания, torque возвращает идентификатор задания (job identifier), каковым в данном примере является “57.master.localdomain”. Формат идентификатора таков:

число.название–сервера.домен

Идентификатор необходим для любого действия, затрагивающего задание, такого как проверка статуса задания, модификации задания, отслеживания или удаления задания.

В приведенном примере задание ставилось в очередь torque; предварительно читались директивы ресурсов, содержащихся в скрипте с заданием. Но существует способ перекрыть (override) атрибуты ресурсов, содержащиеся в скрипте, путем определения их в командной строке. Фактически любая операция постановки задания в очередь или директива, которая определяется в скрипте задания, может быть перекрыта в командной строке через qsub. Это особенно полезно, если нужно просто поставить в очередь новый экземпляр задания без редактирования скрипта.

Пример содержит Листинг 13.

Листинг 13

```
test@master:-> cat ./test.script
#!/usr/bin/csh
#PBS -d /home/test
#PBS -l ncpus=4
#PBS -N Hostname
/bin/hostname
test@master:-> qsub ./test.script
58.master.localdomain
```

В этом примере значения, равные 16 процессорам и 4 часам времени, перекроют значения, определенные в скрипте задания. Нужно также учитывать, что не требуется использовать ключ –l для каждого ресурса. Можно комбинировать запросы нескольких ресурсов, разделив их запятой. Пример такого сценария содержит Листинг 14.

Листинг 14

```
#!/usr/bin/csh
#PBS -l walltime=1:00:00, mem@0mb
#PBS -l ncpus=4
#PBS -oe
./subrun
```

Команда qsub предлагает различные опции для постановки задания в очередь, которые описаны ниже. Еще раз стоит напомнить, что опции чувствительны к регистру.

Определение учетной записи для задания

Опция –A

Задает строку, описывающую локальную учетную запись, ассоциированную с заданием. Строка, заданная в качестве аргумента, не интерпретируется сервером заданий.

Дата и время выполнения задания

Опция –a дата_время

Опция задает дату и время, когда задание станет доступным для выполнения. Аргумент задается в формате: *[[[[ВВ]ГГ]ММ]ДД]ччмм[.сс]*. Здесь:

- дата: ВВ – первые две цифры года (век); ГГ – последние две цифры года; ММ – две цифры месяца; ДД – две цифры дня месяца;
- время: чч – часы; мм – минуты; сс – секунды.

Квадратные скобки означают, что наличие части аргумента не обязательно. Если не указан месяц, то по умолчанию будет выбран текущий, если задан будущий день. Иначе будет выбран следующий месяц. Если не указан день месяца, то будет выбрана сегодняшняя

дата, если указано будущее время. Иначе будет выбран следующий день. Например: если указано время "1110"(11:10), а на данный момент уже 11:15, то задание будет доступно для запуска на следующий день в 11:10.

Листинг 15 содержит пример применения этой опции.

Листинг 15

```
test@master:-> date
Втр Сен 4 16:16:58 MSD 2007
test@master:-> qsub -a 1700 ./test.script
59.master.localdomain
```

Интерактивные задания

Опция –I

Опция позволяет сделать задание интерактивным. Задание ставится в очередь обычным образом, но при его исполнении стандартные потоки ввода, вывода и ошибок подключаются к терминалу, на котором запущена qsub. Если опция –I указана в командной строке или в директиве внутри сценария, задание становится интерактивным. Если же сценарий набран с клавиатуры, будут обработаны директивы, но исполняемые команды не будут включены в задание.

Когда задание начнет свою работу, все данные, поступающие из потока ввода, будут переданы в терминальную сессию, где работает qsub.

После постановки интерактивного задания в очередь, работа qsub не будет прервана, она будет продолжена до завершения выполнения задания (job terminate), его отмены (job aborted), или принудительного выхода из qsub (Ctrl+C). Если работа qsub будет завершена до запуска задания, появится запрос о выходе из среды. При получении подтверждения, задание не будет выполнено.

При выполнении задания, прерывания от клавиатуры передаются в qsub. Строки, начинающиеся со знака “тильда” () и содержащие специальные последовательности, интерпретируются qsub. Распознаваемые специальные последовательности включают в себя:

- ~. - прерывание выполнения qsub. Работа задания также будет прервана.

- ~susp - приостанавливает выполнение qsub, если она запущена в оболочке C shell. “susp” - специальный символ, обычно - Ctrl+Z.
- ~asusp - приостанавливает часть, отвечающую за ввод в qsub, но вывод разрешен и сообщения продолжают отображаться. Работает также в оболочке C shell. “asusp” – символ дополнительной приостановки (auxiliary suspend), обычно Ctrl+Y.

Перенаправление потоков

Опция –e

Опция –o

Опции перенаправляют вывод, позволяя задать имена файлов, в которые будет перенаправлен стандартный вывод (поток stdout) и помещаться ошибки (поток stderr). Опция “–o” задается для потока stdout, опция “–e” – для потока stderr.

Аргумент sпутьї задается в виде:

[hostname:]path_name

Здесь hostname - имя хоста, path_name – путь на заданном хосте. Допустимы абсолютные и относительные пути.

Пример сценария, выводящего на экран название хоста, поток вывода которого перенаправлен в файл mylog, содержит Листинг 16.

Листинг 16

```
test@master:-> qsub -o ./mylog ./test.script
73.master.localdomain
test@master:-> cat ./test.script
#!/usr/bin/csh
#PBS -d /home/test
#PBS -l ncpus=4
#PBS -N Hostname
/bin/hostname
test@master:-> cat ./mylog
node-32
```

test@master:->

Пауза в работе задания

Опция –h

Пауза в работе задания. Опция переводит задание в состояние пользовательской блокировки в момент постановки в очередь. Опция работает аналогично команде qhold. До тех пор, пока блокировка не будет снята, задание не доступно для выполнения.

Объединение потоков

Опция –j

Опция позволяет объединить стандартный поток вывода задания и его поток ошибок. Аргумент “join” может принимать значения: “oe” – в этом случае поток ошибок stderr будет перенаправлен в поток вывода stdout; “eo” – поток вывода stdout будет перенаправлен в поток ошибок stderr. Если в качестве аргумента указана буква “n” или аргумент опущен, перенаправления не происходит и результат работы двух потоков будет находиться в двух отдельных файлах.

Пример:

% qsub –j oe mysubrun

Перенаправление потоков на исполняющий узел

Опция –k keep

Опция позволяет перенаправить потоки вывода и ошибок на исполняющий узел. Аргумент может содержать буквы “e” и “o” в любой комбинации, а также букву “n”. Значение “e” размещает поток ошибок на исполняющем хосте, в домашней папке пользователя, чье задание исполняется. Название файла потока - название_задания.последовательность. Здесь sпоследовательностьї – первая часть идентификатора задания, содержащая числовую последовательность. Пример:

% qsub –k oe mysubrun

Определение ресурсов для задания

Опция –l “выражение”

Аргумент “выражение” опции –l интерпретируется одним из трех способов: либо он обозначает список ресурсов, запрашиваемых для выполнения задания; либо определяет список узлов; либо использует логические выражения для определения ресурсов.

Отсылка по электронной почте

Опции –m опции_отправки, –M список_респондентов Опции настраивают параметры уведомления по электронной почте. Опция s–mï определяет условия, при которых сервер посылает уведомление о выполнении задания. Аргумент «опции_отправки» является строкой, состоящей:

1. либо только из символа “n”;
2. либо из одного или более символов: “a”, “b”, “e”.

В первом случае уведомления не отсылаются. Во втором случае буквы определяют условия отсылки: a - прерывание задания (abort); b - начало выполнения задания (begin); e - завершение выполнения задания (end).

Пример:

% qsub –m ae mysubrun

Опция “-M“ декларирует список пользователей, кому будут отосланы уведомления. Аргумент для этой опции записывается в виде:

пользователь[@хост][, пользователь[@хост],...]

Если аргумент пустой и задана опция “-m”, то уведомления будут отсылаться пользователю- владельцу задания, от чьего имени запущена qsub.

Пример:

% qsub –M james@pbspro.com mysubrun

Изменение названия задания

Опция –N название

Опция определяет название задания. Название должно состоять из печатаемых символов, с первым буквенным символом, пробелы не допускаются. Длина имени не может превышать 15 символов. Если название не указано, то заданию присваивается имя файла сценария, заданное в командной строке. В случае ввода задания с клавиатуры, в консольном режиме, заданию присваивается имя stdin.

Пример:

% qsub –N myName mysubrun

Приоритет задания

Опция –p приоритет

Опция устанавливает приоритет задания. Аргумент является числом от -1024 до 1023 (вклю чительно). По умолчанию задание не имеет приоритета, что эквивалентно установке нулевого значения аргумента. Опция распределяет приоритеты для заданий, которыми владеет текущий пользователь. Следует обратить внимание, что устанавливаемый приоритет служит толь ко ориентиром для планировщика заданий. Планировщик может выбрать свой собственный приоритет.

Пример:

% qsub –p 120 mysubrun

Определение очереди или сервера

Опция –q назначение

Опция определяет очередь или сервер. Эта опция определяет параметры постановки задания в очередь, задавая название очереди, сервера или очереди на сервере. Команда будет передана тому серверу, который указан в аргументе. Если аргумент представляет собой название очереди, задание на сервере перемещается в заданную очередь. Если опция –q не задана, задание ставится в очередь, определенную по умолчанию. Сервер в этом случае

также выбирается заданным по умолчанию. Формат аргумента опции таков: [очередь[@хост]].

Листинг 17 содержит пример сценариев, ставящих задание в очередь batch2. Перед этим выводятся параметры очередей batch и batch2.

Листинг 17

```
test@master:-> qmgr -c "list queue batch"
Queue batch
queue_type = Execution
total_jobs = 0
state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
mtime = Tue Sep 4 15:36:09 2007
enabled = True
started = True
test@master:-> qmgr -c "list queue batch2"
Queue batch2
queue_type = Execution
total_jobs = 0
state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0
mtime = Tue Sep 4 16:44:09 2007
enabled = True
started = True
test@master:-> qsub -q batch2 ./test.script
66.master.localdomain
test@master:->
```

Оболочка интерпретации сценария

Опция –S список_путей

Опция задает используемую оболочку (shell), которая используется для интерпретации сценария. Аргумент scписок_путейї задается в следующем формате: путь[@host][, путь[@host],. . .].

Для одного хоста можно указать только один путь и только один путь можно указать без соответствующего имени хоста. Если опция –S не определена, предполагается, что аргументом является пустая строка, поэтому используется текущая оболочка для пользователя на исполняемом хосте.

Примеры:

% qsub –S /bin/tcsh mysubrun

% qsub –S /bin/tcsh@mars,/usr/bin/tcsh@jupiter mysubrun

Переменные задания

Опция –v список_переменных

Опция задает переменные, которые будут доступны заданию в процессе его выполнению. Имена переменных должны быть разделены запятыми. Переменные и их значения будут переданы заданию после определения их в списке.

Пример:

qsub –v DISPLAY,myvariable2 mysubrun

Зависимые задания

Опция –W список_зависимостей

Опция определяет зависимости между заданиями, или, другими словами, очередность запуска заданий. Листинг 18 содержит пример создания зависимых заданий. Сначала создаются задания с номерами 75 и 76. Затем задание 77 становится зависимым от заданий с номерами 75 и 76 с помощью рассматриваемой опции. После этого используется команда qstart для запуска заданий в очереди batch, а затем запускается задание 75.

Листинг 18

master:/home/test # qstop batch

master:/home/test #qsub ./test.script

75.master.localdomain

test@master:-> qsub ./test.script

76.master.localdomain

test@mater:-> qsub -W dependÏterok:75:76 ./test.script

test@mater:-> exit

exit

master:/home/test # qstart batch

master:/home/test # qrun 75

master:/home/test #

2.3.1.1 Скрипты, запускаемые перед и после выполнения задания

torque позволяет выполнять скрипты перед запуском задания и после завершения его выполнения - пролог- и эпилог-скрипты. Эти скрипты можно использовать, например, для:

- выполнения инициализации;
- освобождения занятых ресурсов, например, удаления временных директорий, после выполнения;
- записи какой-либо информации в выходной файл задания.

Для запуска пролог- и эпилог-скриптов должны выполняться следующие условия:

1. текст самого сценария должен находиться в директории (pbs_home)/mom_priv, имя файла - prologue для пролог-скрипта и epilogue - для эпилог-скрипта;
2. владельцем должен быть пользователь root;
3. у пользователя root должны быть права на чтение и запуск;
4. прав записи в скрипт не должно ни у кого кроме root.

В качестве скрипта может выступать как сценарий оболочки (shell script), так и исполняемый файл.

Аргументы, передаваемые в пролог- и эпилог-скрипты

Внутри скрипта будут доступны следующие аргументы:

Для пролог- и эпилог-скриптов

argv[1] Идентификатор задания argv[2] Имя пользователя, который запускает задание argv[3] Название группы, из-под которого запускается задание

Только для эпилог-скриптов

argv[4] Название задания

argv[5] Идентификатор сессии (session id)

argv[6] Список требуемых ресурсов

argv[7] Список использованных ресурсов

argv[8] Название очереди

argv[9] Строка учетной записи (account string), если существует

Для скриптов определены также следующие параметры: – рабочей директорией является домашняя папка пользователя; – поток ввода – при запуске потоком ввода является стандартный файл системы, ассоциированный с потоком; – поток вывода – потоки вывода и ошибок скриптов ассоциированы с файлами вывода и ошибок задания. Но если задание является интерактивным, потоки вывода и ошибок указывают на файл /dev/null.

2.3.2 Выполнение программ MPI

Для запуска mpi-приложения в torque используется команда mpirun-ipath-ssh, исполняемой внутри сценария. Ниже приведен простейший пример запуск теста High Performance Linpack через систему пакетной обработки заданий на всех процессорах вычислительного кластера.

Пример:

Тест High Performance Linpack установлен в каталоге /share/hpl, исполняемый файл под данную архитектуру находится в /home/test/hpl. Скрипт run.test находится в том же каталоге и выглядит следующим образом:

```
#bin/bash
```

NP=‘cat ${PBS_NODEFILE} | wc -l‘ mpirun-ipath-ssh -np ${NP} -m ${PBS_NODEFILE} ./xhpl

Запуск скрипта осуществляется по команде

“qsub -l nodes#:ppn=2 -d /home/test/hpl /home/test/hpl/run.test”

В результате работы команды в очередь на выполнение будет поставлена задача со следующими параметрами:

- Количество требуемых ресурсов: 16 узлов по 2 процессора на каждом;
- Рабочая директория задачи: /home/test/hpl
- Исполняемый файл: /home/test/hpl/run.test
-

2.3.3 Удаление заданий. Команда qdel

В системе torque имеется команда qdel для удаления заданий. Эта команда удаляет задания в том порядке, в котором указаны их идентификаторы в списке параметров. Удаленное задание более не будет управляться torque. Задание может быть удалено владельцем, оператором или администратором torque. Команда qdel является одной из причин, по которой задание может быть удалено. Другими причинами являются:

- превышение допустимого предела используемых ресурсов;
- завершение работы сервера заданий.

При этом задания удаляются автоматически, без вмешательства пользователя.

Пример командного удаления содержит Листинг 19. В этом примере впервые применена команда qstat, которая выводит информацию заданиях.

Листинг 19

```
test@master:-> qstat
Job id          Name             User            Time Use S Queue
------------------ ---------------- ---------- -------- - -----
78.master       SuperEngine      test                  0 Q batch
test@master:-> qdel 78
test@master:-> qstat
test@master:->
```

2.3.4 Изменение атрибутов задания. Команда qalter

Команда qalter применяется для модификации атрибутов задания.

Синтаксис команды qalter таков:

qalter атрибуты список_заданий

Здесь “атрибуты” эквивалентны опциям команды qsub, они модифицируются соответствующими значениями. Если в списке присутствует один или несколько атрибутов, которые не могут быть изменены либо динамически, либо по другой причине, то не изменятся и все остальные атрибуты.

Листинг 20 содержит пример использования команды qalter. Происходит модификация наименования задания.

Листинг 20

master:/home/test # qstop batch

master:/home/test # su test

test@master:-> qsub ./test.script

78.master.localdomain

test@master:-> qalter -N SuperEngine 78

2.3.5 Изменение состояния заданий. Команды qhold и qrls

Система torque поддерживает две команды, работающих в паре, позволяющих “блокировать” (hold) и “восстановить” (release) задание. Блокировка задания, или установка его состояния в “hold” означает, что задание не может быть выполнено, пока оно не будет восстановлено (release), т.е. метка “hold” не будет снята соответствующей командой.

Команда qhold выдает серверу запрос на установку одной или нескольких меток “hold” на одно или несколько заданий. Блокированное задание не может быть выполнено. Имеется три типа блокировки: пользовательская (user), операторская (operator) и системная (system).

Пользователь может установить пользовательскую блокировку на любое задание, которым он владеет. Оператор, который является пользователем с особыми sоператорскимиї привилегиями, может устанавливать пользовательскую или операторскую блокировку. Пользователь с правами менеджера может устанавливать любой тип блокировки.

2.3.5.1 Команда qhold

Синтаксис команды qhold таков:

qhold [-h hold_list] job_identifier ...

Параметр hold_list определяет тип блокировок, устанавливаемых для задания. Этот аргумент является строкой, состоящей из кратких обозначений одного или нескольких типов

блокировок в любой комбинации: n (none) - нет блокировки; u (user) - пользовательская; o (operator) - операторская; s (system) - системная.

Если опция -h не указана, устанавливается пользовательская блокировка ко всем заданиям с указанными в списке job_identifier идентификаторами.

Если задание, на который указывает один из идентификаторов в списке job_identifier, уже поставлено в очередь, блокировано или находится в одном из состояний ожидания (waiting states), то все, что происходит - это добавление метки “hold” к заданию. Затем задание устанавливается в заблокированное состояние, если оно находится в очереди на исполнение.

Если же задание уже выполняется, то дополнительно происходит прерывание задания. Если операционной системой поддерживаются контрольные точки (checkpoint) или перезапуск (restart), запрос на блокировку задания вызывает следующее:

1. задание устанавливается в состояние контрольной точки (checkpointed);
2. ресурсы, ассоциированные с заданием очищаются;
3. задание устанавливается в блокированное состояние в очереди на выполнение.

В случае, если операционная система не поддерживает контрольные точки или перезапуск, команда qhold только запрашивает состояние блокировки. Таким образом, никакого эффекта не будет до тех пор, пока задание не будет перезапущено (командой qrerun).

2.3.5.2 Команда qrls

Команда qrls снимает блокировку с задания. Тем не менее, пользователь, который выполняет эту команду, должен обладать необходимыми привилегиями, чтобы снять данную блокировку.

Правила, действующие для установки блокировок, аналогично действуют и для их снятия.

Синтаксис команды:

qrls [-h hold_list] job_identifier …

Листинг 21 содержит пример, который демонстрирует, как используется команды qhold и qrls.

Листинг 21

```
test@master:-> qsub ./test.script
79.master.localdomain
test@master:-> qstat
Job id        Name             User            Time   Use  S  Queue
------------------- ---------------- ------------------ -------- - -----
79.master     HostName         test                  0  Q  batch
test@master:-> qhold -h u 79
test@master:-> qstat
Job id        Name             User            Time   Use  S  Queue
------------------- ---------------- ------------------ -------- - -----
79.master     HostName         test                  0  H  batch
test@master:-> qrls 79
test@master:-> qstat
Job id        Name             User            Time   Use  S  Queue
------------------- ---------------- ------------------ -------- - -----
79.master     HostName         test                  0  Q  batch
test@master:->
```

2.3.6 Информация о заданиях. Команда qstat

Для получения информации о заданиях и сервере заданий имеется команда qstat. Запрашиваемая информация выводится в стандартный поток вывода. При запросе состояния задания, на которое пользователь не имеет прав (привилегий), это состояние отображено не будет.

2.3.6.1 Стандартная информация о заданиях

Выполнение qstat без каких-либо опций отображает информацию о заданиях в формате по умолчанию. Отображается следующая информация:

- идентификатор задания, присвоенный системой;
- название задания, присвоенное инициатором задания;
- владелец задания;
- используемое процессорное время;
- состояние (статус) задания;
- очередь, в которой находится задания.

Пример вывода стандартной информации о задании содержит, например, Листинг 17.

2.3.6.2 Расширенная информация о заданиях

Если задать опцию “-a” для команды qstat, то будет для задания будет отображена следующая информация (в дополнение к основной):

- идентификатор сессии (Session ID);
- требуемое количество узлов;

- число параллельных задач (tasks);
- требуемый объем памяти;
- требуемое количество времени;
- количество времени, которое задание находится в текущем состоянии.

Пример вывода расширенной информации о задании приводит Листинг 22.

Листинг 22

```
test@master:-> qsub ./test.script
86.master.localdomain
test@master:-> qsub ./test.script
87.master.localdomain
test@master:-> qsub ./test.script
88.master.localdomain
test@master:-> qstat -a
                                                                  Req'd  Req'd
    Elap
Job ID                  Username    Queue  Jobname       SessID        NDS   TSK
    Memory       Time   S       Time
--------------------    --------  --------  ----------  ------  -----  ---  ------  -----  -  -----
86.master.localdomai    test        batch  Hostname --   --       4      --     –      Q
87.master.localdomai    test        batch  Hostname --   --       4      --     --     Q
88.master.localdomai    test        batch  Hostname --   --       4      --     --     Q
```

Приложение 1. Обзор необходимых команд Linux.

Ниже приводятся некоторые наиболее употребляемые команды Linux. Большинство этих команд можно выполнить на управляющем узле с помощью MidnightCommander. Однако на вычислительных узлах MidnightCommander, как правило, отсутствует.

Чтобы получить более полную информацию по любой отдельной команде command, нужно ввести

```
man command
```

Выход из описания команды производится при нажатии клавиши «q».

Работа с каталогами

`pwd` – показывает название текущей директории;

`cd dir` – устанавливает текущим каталогом каталог с именем dir, вызов команды cd без параметров возвращает в домашний каталог /home/username ($HOME);

`mkdir subdir` – создает новый подкаталог с именем subdir;

`rmdir subdir` – удаляет пустой подкаталог с именем subdir;

`ls` – показывает список файлов и подкаталогов текущей директории,

`ls dir` – показывает список файлов и подкаталогов каталога dir;

`ls -A` - показывает все файлы, в том числе и скрытые;

`ls -l` - показывает атрибуты (владелец, разрешение на доступ, размер файла и время последней модификации);

`mv oldname newname` - изменяет имя подкаталога или перемещает его;

`cp -R dirname destination` - копирует подкаталог dirname в другое место destination.

Работа с файлами

`file filename(s)` - определяет тип файла (например, ASCII , JPEG image data и др.);

`cat filename(s)` - показывает содержание файлов (используется только для текстовых файлов!);

`more filename(s)` - действует так же, как и cat, но позволяет листать страницы;

`less filename(s)` – улучшенный вариант команды more;

`head filename` - показывает первые десять строк файла filename;

`tail filename` - показывает последние десять строк файла filename;

`wc filename(s)` - показывает число строк, слов и байт для указанного файла;

`rm filename(s)` - уничтожает файлы или директории, для рекурсивного удаления следует использовать rm с ключом -rf.

`cp filename newname` - создает копии файлов с новыми именами;

`cp filename(s) dir`- копирует один или более файлов в другой каталог;

`mv oldname newname` - изменяет имя файла или каталога;

`mv filename(s) dir` - перемещает один или более файлов в другой каталог;

`find dir -name filename` - пытается локализовать файл (подкаталог) filename рекурсивно в подкаталоге dir.

Другие полезные команды

`passwd` - изменяет пароль пользователя системы Linux; требует подтверждения старого;

`who` – показывает, кто в настоящее время работает в сети;

`finger` – дает более подробную информацию о пользователях сети;

`write` – позволяет послать сообщение пользователю, работающему в сети в данное время;

`top` - отображает информацию о процессах, использующих процессоры узла;

`ps –U user_name` - показывает номера процессов(pid), инициированных пользователем user_name;

`kill xxxxx` – досрочно завершает работы процесса с номером xxxxx;

`killall proc_name` - досрочно завершает работу процесса proc_name;

`date` - отображает дату и время;

`cal` – показывает календарь.

`exit` – выйти из терминала

`clear` – очистить окно терминала

`du dir` – показывает занятое место в директории dir

Приложение 2. Примеры PBS скриптов

Подробно о возможностях PBS Torque можно узнать из руководства пользователя.

Ниже приведены несколько примеров использования Torque.

```
#PBS -o $DIR/stdout.log
```

Определяет имя файла, в который будет перенаправлен стандартный поток stdout

```
#PBS -e $DIR/stderr.log
```

Определяет имя файла, в который будет перенаправлен стандартный поток stderr

```
#PBS -l nodes=8:ppn=2:cpp=1
```

Определяет какое количество узлов и процессоров на них необходимо задействовать.

nodes - количество узлов

ppn - число процессоров на узле

cpp - число процессов на процессоре

```
#PBS -l walltime=20:00:00
```

Определяет максимальное время счета задания

```
#PBS -l mem=1000mb
```

Определяет количество необходимой оперативной памяти

```
cat $PBS_NODEFILE | grep -v master | sort | uniq -c | awk '{printf
"%s:%s\n", $2, $1}' >
$PBS_O_WORKDIR/temp.tmp
```

Составляет список узлов в необходимом формате, на которых будет запущена задача и записывает их в файл temp.tmp

```
cd $PBS_O_WORKDIR
/usr/bin/mpirun -m temp.tmp -np 100 ./a.out
```

Запускает на узлах указанных в файле temp.tmp задачу 100 раз.

Пример скрипта:

```
#PBS -o $DIR/stdout.log
#PBS -e $DIR/stderr.log
#PBS -l nodes=50:ppn=2
#PBS -l walltime=20:00:00
```

```
#PBS -l mem=1000mb
cat $PBS_NODEFILE | grep -v master | sort | uniq -c | awk '{printf
"%s:%s\n", $2, $1}' >
$PBS_O_WORKDIR/script1.temp.sh.mf
cd $PBS_O_WORKDIR
/usr/bin/mpirun -m script1.temp.sh.mf -np 100 ./a.out
```

Здесь будет запущена параллельная программа a.out на 50 узлах, с каждого узла будет использоваться 2 процессора. Файл вывода стандартного потока stdout — stdout.log, стандартного потока stderr — stderr.log.

$DIR содержит путь к файлам stdout.log и stderr.log, например может принимать значение /home/user_name. Под задачу отведено 20 часов. Необходимое количество памяти 1000 мегабайт.

Для запуска последовательной программы first можно использовать следующий скрипт:

```
#PBS -o $DIR/stdout.log
#PBS -e $DIR/stderr.log
#PBS -l walltime=10:00
#PBS -l mem=100mb
./first
```

При запуске программы через команду qsub заданию присваивается уникальный целочисленный идентификатор.

`qdel` – утилита для удаления задачи.

В случае, если задача уже запущена, процесс ее работы будет прерван. Синтаксис данной утилиты следующий:

```
qdel [-W время задержки]идентификатор задачи
```

Выполнение такой команды удалит задачи с заданными идентификаторами через указанное время. Если часть вычислительных узлов, на которых выполнялась задача, недоступны, то принудительно удалить ее с сервера можно путем добавления ключа -p.

Приложение 3. Переменные окружения планировщика Torque

Далее приводится список переменных окружения Torque и примеры их значений.

PBS_JOBNAME=env

PBS_ENVIRONMENT=PBS_BATCH

PBS_O_WORKDIR=/home/test

PBS_TASKNUM=1

PBS_O_HOME=/home/test

PBS_MOMPORT=15003

PBS_O_QUEUE=batch

PBS_O_LOGNAME=test

PBS_O_LANG=en_US.UTF-8

PBS_JOBCOOKIE=3088939E7FAA7F4414578D7A806955

PBS_NODENUM=0

PBS_O_SHELL=/bin/bash

PBS_JOBID=93.master.localdomain

PBS_O_HOST=master.localdomain

PBS_VNODENUM=0

PBS_QUEUE=batch

PBS_O_MAIL=/var/spool/mail/test

PBS_O_PATH=/home/test/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:

/usr/games:/opt/gnome/bin:/opt/kde3/bin:

/usr/lib/mit/bin:/usr/lib/mit/sbin

Список литературы

[1] http://www.clusterresources.com/torquedocs/index.shtml

[2] http://www.clusterresources.com/torquedocs21/usersmanual.shtml

[3] http://supercomputer.susu.ru/users/instructions/torque_manual.pdf