Вычислительный кластер КарНЦ РАН

Краткое руководство пользователя

Оглавление

0. О чем этот текст

Данное руководство предназначено для пользователей вычислительного кластера КарНЦ РАН. Цель руководства — дать начальные сведения о кластере КарНЦ РАН, его структуре, приемах и методах работы.

Это руководство не содержит информации о параллельном программировании. Этот текст не является руководством по операционной системе Linux. Большинство команд, приведенных в данном руководстве, имеют большое количество параметров и нюансов, и подробное их рассмотрение проводится в специальной литературе.

1. Что такое кластер

В этом разделе описывается кластер КарНЦ РАН.

Вычислительный кластер — это группа вычислительных узлов, объединенных высокоскоростными каналами связи, представляющая с точки зрения пользователя единую вычислительную систему. Основное предназначение вычислительного кластера — выполнение большого количества расчетов. Основная характеристика вычислительного кластера — производительность вычислений, которая измеряется числом арифметических операций в секунду. В отличие от персональных компьютеров кластер способен выполнять параллельные вычисления.

1.1. Аппаратная часть

Кластер КарНЦ РАН состоит из следующих основных компонентов:

10 вычислительных узлов;
1 управляющий узел;
система хранения данных, подключенная к управляющему узлу;
системная сеть, объединяющая узлы.

Управляющий узел — это компьютер, который позволяет получить доступ к ресурсам кластера. Также управляющий узел организует работу кластера и управляет его ресурсами.

Управляющий узел кластера КарНЦ РАН имеет следующие характеристики:

2 CPU Quad-Core Intel Xeon 5430 2.66 GHz, cache 12 Mb, 1333MHz;
RAM: 4Gb DDR2-667;
6 HDD: 146Gb SAS.

Вычислительный узел — это многопроцессорный компьютер, на котором выполняются задачи пользователя. Задача пользователя может занимать несколько вычислительных узлов, даже кластер целиком. Одновременное выполнение нескольких задач на одном процессоре не допускается.

Характеристики вычислительного узла:

2 CPU Quad-Core Intel Xeon 5430 2.66 GHz, cache 12 Mb, 1333MHz;
RAM: 4Gb DDR2-667;
HDD: 250Gb SATA.

Управляющий и вычислительные узлы объединены в локальную сеть на основе технологии Infiniband, которая обеспечивает пропускную способность 20 Gbit/s.

Теоретическая вычислительная мощность кластера составляет 851 GFlops, а лучший результат при использовании тестов Linpack составил 634 GFlops.

Также в состав кластера входит система хранения данных, полезный дисковый объем которой составляет 1 Тб.

1.2. Программное обеспечение

В качестве операционной системы на всех узлах установлена SuSE Linux Enterprise Server 10.

В состав системного программного обеспечения входит пакет средств для параллельного программирования Intel Cluster Toolkit Compiler Edition:

компилятор Intel® C++ и Fortran;
библиотека Intel® MPI Library;
инструменты Intel® Trace Analyzer и Intel® Trace Collector;
библиотека Intel® Math Kernel;
эталонные тесты Intel® MPI.

Также установлен параллельный отладчик Total View Debugger.

В качестве технологий параллельного программирования доступны технологии OpenMP и MPI (mvapich-1.1.0, mvapich2-1.2p1, openmpi-1.3). Из прочих средств разработки установлены GNU C++, GNU Fortran, Perl, Python.

На кластере используется система управления заданиями Slurm.

2. Функциональность кластера

В данном разделе перечисляются основные действия, которые может выполнять пользователь кластера.

Цикл работы пользователя с кластером состоит из следующих этапов:

1. Удаленный вход на кластер.
2. Копирование данных между кластером и компьютером пользователя.
3. Редактирование исходных текстов программ.
4. Компиляция программ.
5. Управление очередью выполнения задач.
6. Завершение сеанса.

Обязательными пунктами являются первый и последний — пользователь должен получить доступ к кластеру, и рано или поздно завершить сеанс работы с кластером. Все остальные действия могут варьироваться: например, редактировать исходные программы пользователь может и на своем компьютере, пользователь может зайти на кластер, чтобы проверить результаты выполнения задач, и т.д. Некоторые действия могут выполняться параллельно: необязательно дожидаться окончания расчетов одной задачи, чтобы поставить в очередь новое задание. Более того, несколько пользователей могут одновременно и независимо работать с кластером, не мешая друг другу.

Кластер КарНЦ РАН — многопользовательская система, его ресурсы (процессорное время и дисковое пространство) ограничены, поэтому для пользователей кластера существуют некоторые пределы на использование его ресурсов: так, время вычислений одного задания не превышает по умолчанию 3 суток; максимальный доступный объем дискового пространства, доступного обычному пользователю составляет 3 Гб; пользователь не может просматривать содержимое домашних каталогов других пользователей и некоторых системных каталогов.

3. Основы работы с кластером

В данном разделе подробно описываются основные действия пользователя кластера, указанные в разделе 2.

Поскольку с точки зрения пользователя кластер представляет из себя единую машину под управлением Linux, для управления кластером необходимо знать основные принципы работы в Linux-системах. Здесь мы приведем команды, необходимые для выполнения основных действий, указанных в предыдущем разделе. Некоторые основные команды Linux приведены в разделе 4.

3.1. Удаленный вход на кластер

Физически кластер КарНЦ РАН располагается в специально предназначенном отдельном помещении, и непосредственный доступ пользователей к нему ограничен. Кластер подключен к локальной сети КарНЦ РАН, и поэтому для получения доступа к его ресурсам необходимо выполнить так называемый удаленный[1] вход на кластер. Прежде всего, вы должны пройти процедуру регистрации[2], и получить логин и пароль для доступа к его ресурсам.

Для входа на кластер необходимо знать его IP-адрес. Внутри сети КарНЦ РАН адрес кластера 192.168.1.230, а внешний IP-адрес **82.196.66.9**. Это означает, что если вы пытаетесь подключиться к кластеру, находясь при этом в локальной сети КарНЦ РАН, вы должны указывать внутрисетевой IP-адрес кластера, а внешний IP-адрес используется при подключении из сети Интернет.

Удаленный вход на кластер выполняется с использованием протокола SSH-2. Процедура входа зависит от операционной системы компьютера, с которого выполняется вход, поэтому мы опишем 2 основных варианта:

Для пользователей

Для доступа к кластеру наберите в командной строке (не забудьте нажать Enter в конце строки):

```
ssh username@hostname
```

«username» в данном случае — это ваш логин, а «**hostname**» — это IP-адрес кластера. На запрос системы вы должны ввести пароль, полученный при регистрации. Рекомендуем сменить пароль при первом входе на кластер, и менять его не реже одного раза в месяц.

Например, если ваш логин — «**tester**», и вы находитесь в сети КарНЦ РАН, вы должны ввести:

```
ssh tester@192.168.1.230
```

Для пользователей Windows

Для подключения из Windows необходимо использовать специальную программу — SSH-клиент. Мы рекомендуем программу PuTTY[3].

Для установки программы PuTTY скопируйте на свой компьютер файл putty.exe и запустите

1 Удаленный в смысле «находящийся на расстоянии».
2 http://cluster.krc.karelia.ru
3 http://ru.wikipedia.org/wiki/PuTTY

его. В появившемся окне в строке «Host Name (or IP address)» укажите IP-адрес кластера (например, **192.168.1.230**), выберите протокол SSH, и начните сессию. Откроется новое окно с приглашением «login as:». Вам необходимо ввести имя пользователя, а дальше система запросит пароль.

Если все прошло успешно, появится приглашение командной строки:

```
tester@krc-master:~>
```

Это означает, что вы успешно подключились к кластеру и находитесь в своем домашнем каталоге. Отметим сразу же, что путь к домашнему каталогу /home/<username>

Примечание: Если вы получаете доступ к кластеру в первый раз, операционная система выдаст предупреждение, что ей неизвестен узел с адресом 192.168.1.230 (либо 82.196.66.9, если вы находитесь вне сети КарНЦ РАН), и спросит, согласны ли вы продолжить подключение:

```
The authenticity of host '192.168.1.230 (192.168.1.230)' can't be
established.
RSA key fingerprint is f2:36:67:91:d2:46:e8:7e:3b:00:86:5b:db:9b:15:14.
Are you sure you want to continue connecting (yes/no)?
```

Здесь вам необходимо согласиться, набрав слово **yes**, и нажав Enter. Смысл сообщения системы следующий: ваш компьютер пока не уверен, что хост с адресом 192.168.1.230 тот, за кого он себя выдает[4]. После вашего подтверждения хост 192.168.1.230 будет добавлен к списку известных хостов,

```
Warning: Permanently added '192.168.1.230' (RSA) to the list of known
hosts.
```

и система продолжит работу.

3.2. Копирование данных между кластером и компьютером пользователя

Еще одно отличие работы в режиме удаленного доступа от работы с персональным компьютером состоит в том, что вы не имеете физического доступа к кластеру, то есть вы не сможете вставить DVD-диск, либо flash-накопитель в кластер. Поэтому для того, чтобы скопировать необходимые файлы на кластер, либо забрать информацию с него, нужно использовать команду scp либо sftp, если вы работаете в Unix-системе, а пользователи Windows могут воспользоваться одной из многочисленных программ, предназначенных для удаленного копирования файлов (например, WinSCP[5]).

Опишем действия, необходимые для копирования файлов между двумя компьютерами с использованием протокола SCP (secure copy — защищенное копирование).

4 Для этого нужно убедиться, что «слепок ключа» (key fingerprint) совпадает с вышеприведенным.

5 http://winscp.net/eng/docs/lang:ru

Для пользователей Unix и Linux

Упрощенный формат команды scp следующий[6]:

scp [[user@]host1:]file1 ... [[user@]host2:]file2

Команда scp копирует файлы между двумя компьютерами в сети. Первый параметр указывает на копируемый файл на одном из компьютеров, второй параметр указывает путь его размещения на другом компьютере. Как видно, копирование файла осуществляется между двумя компьютерами host1 и host2, причем команда scp должна выполняться на одном из этих хостов. При этом с компьютера host1 файл копируется на host2, и, если необходимо, для подключения к удаленному хосту нужно будет указать логин и пароль пользователя, имеющего доступ к этому хосту.

Приведем пример. Будем считать, что вы находитесь на своем рабочем компьютере, подключенном к локальной сети КарНЦ РАН. Предположим, ваш логин на кластере «**tester**» и вы хотите скопировать файл myprog.cpp, находящийся на кластере в каталоге /home/tester/ на свой рабочий компьютер в каталог /home/user/test. Для этого вам нужно ввести следующую команду:

scp tester@192.168.1.230:/home/tester/myprog.cpp /home/user/test/[7]

В этом примере мы не используем необязательные части второго параметра, потому что мы уже находимся на своем компьютере и нам не нужно указывать его адрес в сети и логин для подключения.

Обратно, если вам необходимо скопировать файл myprog.cpp с рабочего компьютера обратно на кластер, вы должны использовать команду

scp /home/user/test/myprog.cpp tester@192.168.1.230:/home/tester/

Здесь тоже все понятно — копируемый файл указан в первом параметре, а второй параметр указывает, что файл должен быть скопирован на кластер в домашний каталог пользователя tester.

Примечание: Обратите внимание, что команда scp использует протокол ssh, и поэтому от вас потребуется вводить пароль для удаленного доступа.

Для пользователей Windows

Рассмотрим процедуру копирования файлов на примере программы WinSCP. После установки WinSCP запустите программу. При запуске программа запросит у вас информацию, необходимую для подключения: нужно указать в соответствующих полях IP-адрес кластера, логин и протокол для соединения, затем нажать кнопку Login. Если введенная информация корректна, откроется окно, по внешнему виду сходное с Windows Explorer либо двухпанельное окно в стиле Total Commander (в зависимости от варианта, выбранного при установке программы). В этом окне будет отображена файловая система

6 Здесь и далее при описании команд выражения в квадратных скобках считаются необязательными. Сами скобки вводить не нужно.

7 К сожалению, эта и следующая команда не поместились на одной строке. Тем не менее, команды необходимо вводить в одну строку.

кластера и вы можете копировать файлы на кластер и обратно так же, как это делается в Windows.

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

3.3. Редактирование исходных текстов программ

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: ed, emacs, vim, joe. Наиболее простым для освоения, на наш взгляд, является редактор, встроенный в оболочку Midnight Commander.

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, Total Commander, 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 запуска этой программы используется команда mc.

Для редактирования файла необходимо клавишами управления курсором выбрать нужный файл, и нажать клавишу F4. Запустится редактор текстовых файлов, выйти из которого можно, 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.

3.4. Компиляция программ

Как было сказано ранее, на кластере установлены компиляторы Intel® C++ и Fortran и свободно распространяемый компилятор gcc. Кроме того, имеется ряд специальных компиляторов для отдельных версий MPI, а именно, MVAPICH, MVAPICH2, MPICH, OpenMPI.

Выбор типа MPI

Прежде, чем перейти к этапу компиляции, следует определиться с используемой версией MPI. Это касается и запуска уже скомпилированных программ, таких как Firefly, PC GAMESS (обычно такие программы сопровождаются информацией о необходимой версии MPI).

По умолчанию все пользователи кластера используют версию **MVAPICH**. Убедиться в этом можно, набрав в консоли команду

```
mpi-selector --query
```

Для изменения версии MPI можно воспользоваться утилитой **mpi-selector-menu**.

```
mpi-selector-menu
Current system default: mvapich-intel
Current user default:   <none>

   "u" and "s" modifiers can be added to numeric and "U"
   commands to specify "user" or "system-wide".

1. IntelMPI
2. mpich2
3. mvapich-gnu-debug
4. mvapich-intel
5. mvapich-intel-debug
6. mvapich2-intel
7. mvapich2-intel-debug
8. openmpi-intel
U. Unset default
Q. Quit

Selection (1-8[us], U[us], Q):
```

В ответ следует указать номер, соответствующий нужной версии MPI, и подтвердить выбор, нажав **u**. После этого для сохранения изменений следует **завершить сеанс** работы с кластером и **вновь начать сеанс** (т. е. выйти с кластера командой exit и заново войти по ssh).

Редактирование исходных кодов программы

При написании программ следует учитывать требуемый тип MPI. Для этого при подключении заголовочного файла необходимо явно указать

Для компиляции программ, написанных с использованием технологий параллельного программирования, используются специальные компиляторы:

mpicc для программ на языке C/C++:
mpif77 и mpif90 для программ на языке Fortran.

Достаточно подробную справку по использованию компиляторов вы можете получить, запустив компилятор с параметром --help (например, набрав в командной строке mpif77 --help). Здесь мы приведем основную форму запуска компилятора:

```
mpicc -o test test.cpp
```

Ключ -o указывает, что мы хотим скомпилировать файл test.cpp и результат компилирования должен быть помешен в файл test. Если мы просто выполним команду

```
mpicc test.cpp
```

то в текущем каталоге появится исполняемый файл a.out.

При выборе **IntelMPI** в качестве используемой версии MPI рекоммендуется использовать компиляторы **icc** и **ifort**.

При использовании в качестве MPI версии **mvapich2-1.8**, необходимо при компиляции программ в обязательном порядке указывать опции -L/usr/local/lib -lpmi для подключения библиотеки **libpmi** системы управления очередью Slurm. Убедиться в том, что линковка осуществлена правильно, можно с помощью утилиты ldd в формате ldd <имя_исполняемого_файла>. Например, компиляция тестового файла может быть выполнена с помощью команды

```
mpicc test.cpp -L/usr/local/lib -lpmi -o testmpi
```

Для проверки правильности линковки в этом случае можно воспользоваться командой

```
ldd testmpi
```

3.5. Управление очередью задач

Для того, чтобы выполнить программу на кластере, необходимо поставить ее в очередь вычислений. В качестве системы управления заданиями на кластере КарНЦ РАН используется система Slurm.

Постановка задачи в очередь

Чтобы поставить задачу в очередь, используется команда run:

```
srun [options] prog [prog_args]
```

prog и prog_args – исполняемый файл задачи и ее аргументы соответственно.

Приведем значения основных параметров:

-n	количество процессов (в случае MPI задачи);
-N	количество узлов (полезно для OpenMP Задач);
-t	лимит времени счета в минутах;
-A	профиль, с которого будет запущена задача;
--exclusive	получить узел в эксклюзивное пользование задачей
--mail-type	события, по которым отправить уведомление письмом
--mail-user	почтовый адрес, на который отправить уведомление

Обязательным является указание параметра **-n** или **-N**.

Желательно указывать также параметр **-t**, т. к. алгоритм размещения задач в очереди может отдавать приоритет более коротким задачам, не нарушая общей последовательности выполнения (используется алгоритм Backfill). По умолчанию максимальное время вычисления задачи на кластере равно 72 часам.

Дополнительно параметр **-A** позволяет указать один из доступных пользователю профилей. Разные профили обладают разными ограничениями. В частности, профиль **singlecore** позволяет запускать одновременно до 8 задач в однопроцессорном режиме, в то время как под другим профилем пользователь может ставить в очередь не более 3 задач. Параметр **--exclusive** позволяет получить узел в монопольное пользование и не делить его ресурсы с другими задачами. Это важно прежде всего для задач OpenMP, а также для задач, использующих большой объем оперативной памяти.

Пример:

```
srun -n 16 -t 10 my_test 50
```

Данная команда поставит задачу 'my_test' с параметром '50' в очередь с лимитом работы в 10 минут и запросом на 16 процессоров.

```
srun -n 1 -A singlecore my_test3
```

Данная команда поставит задачу my_test3 в очередь для выполнения в однопроцессорном режиме.

После запуска задачи ее стандартный вывод и стандартный поток ошибок **будет перенаправлен в консоль**. Команда run будет выполняться в **интерактивном режиме**, позволяя таким образом передавать программы команды или вводить входные данные, которые будут также считываться с консоли. Кроме того, система управления заданиями позволяет отслеживать состояние задачи непосредственно нажатием Ctrl+C в процессе работы задачи.

Для того, чтобы запустить задачу в фоновом режиме и освободить консоль, рекомендуется воспользоваться услугами **виртуальной консоли**. Для этого необходимо выполнить команду **screen**

```
screen

Screen version 4.00.02 (FAU) 5-Dec-03

Copyright (c) 1993-2002 Juergen Weigert, Michael Schroeder

Copyright (c) 1987 Oliver Laumann

This program is free software; you can redistribute it and/or modify it under the terms of

...

                    [Press Space or Return to end.]
```

В ответ на приглашение ознакомиться с лицензией и продолжить работу, следует нажать клавишу ввода или пробел. Пользователь получает в распоряжение виртуальный терминал. Внутри виртуального терминала используются абсолютно те же команды, что и в обычном терминале, просматривается содержимое тех же файлов итд. Польза виртуального терминала в том, что его можно «приостановить», затем осуществить выход из сеанса работы на кластере, а в дальнейшем «возобновить» работу. При этом после завершения сеанса на кластере виртуальный терминал не будет завершен системой и продолжит функционирование до следующего возобновления его работы пользователем. Пользователь может запускать одновременно неограниченное число виртуальных терминалов.

Находясь внутри виртуального терминала (после ввода команды **screen**), пользователь может осуществлять как редактирование исходных текстов, компиляцию, так и запуск задач при помощи команды run. При этом если пользователь желает оставить задачу выполняться **в интерактивном режиме**, он может сразу осуществить операцию «отключения» от виртуального терминала. Также существует возможность запуска нескольких заданий «**в фоновом режиме**», для чего после команды run ставится символ &:

```
srun -n 1 -A singlecore my_test3 &
```

Запустив нужные задачи на выполнение, пользователь может выполнить «отключение» от терминала при помощи нажатия последовательности клавиш **Ctrl+A**, и сразу же **Ctrl+D**. В ответ пользователь получит сообщение об «отключении» от виртуального терминала:

```
[detached]
```

Затем пользователь может выполнить выход из сеанса на кластере при помощи команды logout, exit либо нажатием Ctrl+D.

Позднее можно вернуться к «отключенному» терминалу, осуществив вход на кластер и набрав команду

```
screen -r
```

В случае если пользователь оставил несколько виртуальных серверов в «отключенном» режиме, система предупредит его об этом и попросит выбрать, указав «номер» оставленного сеанса.

Для завершения виртуального сеанса также можно нажать комбинацию **Ctrl+D**.

Для того, чтобы перенаправить вывод программы в файл, или подать на ввод программе файл, необходимо использовать параметры **-i, -o, -e.**

Например, команда

```
srun -n 16 -t 10 -o my_out.txt my_test 50 &
```

осуществит запуск в фоновом режиме и перенаправит стандартный вывод программы в файл my_out.txt. Имя выходного файла можно сгенерировать автоматически на основании таких данных, как идентификатор задачи, имя узла или номер процесса. Для этого в указании имени файла можно использовать специальные конструкции вида **%j** — для подстановки номера задачи, **%t** — для подстановки номера процесса, **%N** — для имени узла, **%n** — для номера узла. При этом можно использовать ограничение на количество символов в подстановке, например **%4j** подставит 4 цифры номера задачи (с ведущими нулями, если необходимо).

Постановка скрипта очередь

Отметим, что хотя и допустимо использовать сочетание команд run и screen для запуска программы на вычисление в фоновом режиме, желательно воспользоваться возможностями команды sbatch. Эта команда предназначена для постановки задачи в так называемом пакетном режиме. Тем самым возможно использовать выделенные ресурсы для вычисления нескольких задач. Также запуск в пакетном режиме позволяет проводить предварительные и заключительные действия по запуску задачи (например, инициализацию переменных окружения, создание файлов, очистку временных файлов и т. п.).

Для запуска необходимо создать скрипт, содержащий, в простейшем случае, команду запуска задачи:

```
cat myscript

#!/bin/bash

srun my_test 50
```

Постановка задачи в пакетном режиме выполняется с помощью следующей команды:

```
sbatch -n 16 -t 10 myscript
```

Далее система автоматически запланирует соответствующие ресурсы, создаст файл вида slurm-<номер задачи>.out и перенаправит в него вывод консоли. Затем, при доступности ресурсов, будут выполнены последовательно команды из скрипта, результат выполнения команд будет записан в указанный файл.

Пакетный режим позволяет завершать сессию доступа к кластеру без создания виртуалных консолей, задачи, поставленные в пакетном режиме, не теряются после завершения сессии.

Параметры команды sbatch аналогичны параметрам команды run. Отметим три возможных способа передачи параметров:
1) с помощью параметров командной строки (наиболее приоритетный)
2) с помощью переменных окружения
3) с помощью параметров в скрипте.

Для передачи параметров с помощью скрипта необходимо перечислить их во второй строке скрипта в следующем виде:

```
cat myscript

#!/bin/bash

#SBATCH -n 16 -t 10

srun my_test 50
```

Просмотр состояния очереди

В любой момент Вы можете посмотреть состояние очереди командой squeue:

```
squeue [options]
```

Приведем значения **необязательных** параметров:

-A	профиль, для которого отображается информация;
-i N	итеративный режим (обновляет информацию каждые N сек.);
-a	показывать задачи всех пользователей;

Удаление задачи из очереди

Для удаления задачи из очереди используется команда scancel с перечислением задач, которые необходимо удалить. Так, команда

```
scancel 485
```

удалит из очереди задание с id=485. Команда

```
scancel -u user
```

удалит из очереди все задачи пользователя user, выполнившего эту команду, причем задачи других пользователей удалены не будут. Команда

```
scancel -p main
```

удалит из очереди main все задачи пользователя, запустившего ее.

3.6. Завершение сеанса

Для завершения сеанса удаленного доступа используется команда exit. После выполнения этой команды вы можете снова получить доступ к кластеру, осуществив удаленный вход.

Также завершить сессию можно, нажав Ctrl+D в командной строке.

> **Примечание:** Завершение сеанса прекращает сеанс работы с кластером, но не останавливает программы, выполняющиеся на кластере в фоновом режиме, и не влияет на других пользователей.

4. Основные принципы работы в Linux

В данном разделе описываются основные принципы работы в Linux и приводятся основные команды.

4.1. Операции с файлами и каталогами

Файловая система — это способ организации файлов, то есть можно сказать, что любой файл или каталог, хранящийся на кластере, находится в определенном месте файловой системы. Каждая программа работает в определенном каталоге, который называют текущим каталогом программы. Так как пользователь ведет диалог с операционной системой в специальной программе — командной оболочке, можно сказать, что текущий каталог командной оболочки является текущим каталогом пользователя. Чтобы определить текущий каталог, используется команда pwd. Если вы используете Midnight Commander, текущий каталог отображается в активной панели программы.

Файловая система представляет собой древовидную структуру, начинающуюся от корневого каталога. В каждом каталоге могут находиться другие каталоги и файлы. Поскольку каждый файл находится в определенном месте файловой системы, существует путь к файлу. Путь может быть относительным и абсолютным. Абсолютный путь — это путь к файлу, который начинается от корневого каталога (обозначаемого символом «/»), а относительный путь начинается от текущего каталога. Например, если имеется файл test.cpp, находящийся в каталоге /home/tester/bin, то абсолютный путь к этому файлу будет выглядеть, как

```
/home/tester/bin/test.cpp
```

а относительный путь из каталога /home/tester/work будет таким:

```
../bin/test.cpp
```

Примечание: Обратите внимание на символ «..» (две точки). Таким символом обозначается предыдущий по отношению к текущему каталог. Другими словами, относительный путь выглядит так: в предыдущем каталоге по отношению к текущему (а для каталога work это будет каталог tester), находится каталог bin, в котором есть файл test.cpp. Используется также символ «.» (одна точка). Этот символ в пути обозначает текущий каталог. Его использование обусловлено тем, что иногда системе необходимо явно указать, что она должна использовать файл из текущего каталога, то есть необходимо явно указать путь к файлу, который может быть достаточно длинным. В этом случае можно использовать относительный путь, к примеру ./test.cpp для файла test.cpp из текущего каталога.

Смена каталога

Для того, чтобы перейти в произвольный каталог, используется команда cd. Например,

```
cd /home/tester/
```

сделает каталог /home/tester текущим каталогом командной строки. Команда cd / переходит к корневому каталогу.

Также у каждого пользователя существует специальный каталог, называемый домашним каталогом. Этот каталог находится в каталоге /home и его название совпадает с именем пользователя. В этом каталоге и подкаталогах пользователь имеет все права доступа: он может создавать файлы и каталоги, удалять их, переименовывать и перемещать, а также запускать программы на

выполнение.

> **Примечание:** Для домашнего каталога используется обозначение «~» (тильда). То есть, команду cd /home/tester/bin пользователь tester может заменить командой
>
> ```
> cd ~/bin
> ```

Список объектов в каталоге

Для того, чтобы посмотреть список файлов в каталоге, используется команда ls. Команда, вызванная без параметров, выводит список файлов текущего каталога. А команда

```
ls /home/tester
```

выведет на экран содержимое домашнего каталога пользователя tester. Команда ls имеет множество параметров, например -a выводит все объекты каталога, включая системные, а ключ -l позволяет получить полную информацию об этих объектах.

Создание и удаление каталога

Команда

```
mkdir <name>
```

создает каталог с указанным именем. Если указано только имя каталога, он будет создан в текущем каталоге, но можно указать путь к создаваемому каталогу, как абсолютный, так и относительный. Например, находясь в корневом каталоге, пользователь tester может создать каталог data в своем домашнем каталоге, выполнив команду

```
mkdir /home/tester/data
```

Для удаления каталога используется команда

```
rmdir <name>
```

Нужно учесть, что эта команда удаляет только пустые каталоги, поэтому перед удалением любого каталога необходимо удалить из него все файлы и каталоги.

Создание файлов

Текстовый файл можно создать в любом текстовом редакторе, указав имя создаваемого файла при запуске редактора. Если вы пользуетесь Midnight Commander, для создания нового текстового файла нажмите Shift+F4. Третий способ создания файла — скопировать уже существующий файл.

Если вы хотите создать новый файл нулевой длины, можно воспользоваться командой touch. Например,

```
touch ~/test
```

создает пустой файл test в домашнем каталоге. Строго говоря, команда touch предназначена для изменения времени доступа к файлу, но, если указанного файла не существует, то она создает его.

Копирование, перемещение и удаление файлов и каталогов

Команда cp позволяет скопировать файл или каталог. Формат команды следующий:

```
cp <source> <destination>
```

В каталоге <destination> создается копия файла, каталога, либо списка файлов или каталогов, указанных параметром <source>. Исходные объекты не изменяются.

Команда mv перемещает либо переименовывает файл или каталог. Отличие от копирования заключается в том, что исходные объекты либо получают новое имя, либо перемещаются в новое место.

```
mv <source> <destination>
```

Для удаления файлов используется команда rm.

```
rm <name>
```

Примечание: Удаление файлов и каталогов — необратимая операция! На кластере нет промежуточного места для удаленных файлов (в Windows это место называется «Корзина»), поэтому удаленный файл восстановить практически невозможно.

Вывод текстового файла на экран

Команда less выводит содержимое текстового файла на экран. Эта программа удобна тем, что позволяет просматривать содержимое файла постранично и умеет искать данные в файле.

```
less test.cpp
```

Используйте для перемещения по тексту следующие клавиши:

f либо пробел — следующая страница;
b либо Esc-v — предыдущая страница;
/шаблон — поиск шаблона вперед по тексту;
n — повторить поиск вперед;
?шаблон — поиск шаблона назад по тексту;
N — повторить погиск назад;
q — выход.

4.2. Перенаправление вывода

Возможно, вы уже заметили, что многие программы и команды выводят результат работы на экран, и полностью вывод на одном экране не помещается. Есть несколько способов прочитать такой текст. Первый способ — нажать клавишу «Scroll Lock» и можно будет перемещать содержимое экрана с помощью клавиш управления курсором. Для выхода из этого режима нажмите клавишу «Scroll Lock» еще раз.

Второй способ заключается в том, что вывод на экран результатов работы любой программы перенаправляется в текстовый файл. Для этого используется специальный символ перенаправления вывода «>» (символ «больше»). Так, после выполнения команды

```
cmd1>file
```

в текущем каталоге появится файл с именем file, в котором будет содержаться весь текстовый вывод программы cmd1. Если такой файл существовал, он будет перезаписан заново. Если использовать 2 символа перенаправления вывода подряд, весь вывод будет дописан к концу файла, то есть команда

```
cmd1>>file
```

не будет перезаписывать файл file, а дополнит его новыми данными.

4.3. Работа с системой документации

Основной способ получения информации в Linux — чтение так называемых **manpages** (manual pages — руководства). Большинство объектов Linux, начиная от основных понятий, заканчивая командами операционной системы описаны в таких руководствах. Для того, чтобы прочитать руководство по какой-нибудь команде, наберите в командной строке

```
man <object>
```

Например, команда

```
man man
```

выведет на экран руководство по использованию команды man.

Если страница руководства существует, она будет выведена на экран. Для перелистывания страниц используйте клавиши управления курсором, PgUp, PgDown, Home, End, а для завершения просмотра нажмите клавишу q.

Также многие программы могут запускаться в режиме выдачи помощи. Для этого их надо запускать с ключом --help (два знака «минус»help). В этом случае на экран будет выведена краткая справка по использованию программы. Вот, к примеру, частичный результат выполнения команды man --help:

```
usage: man [-c|-f|-k|-w|-tZT device] [-adlhu7V] [-Mpath] [-Ppager] [-Slist]
[-msystem] [-pstring] [-Llocale] [-eextension] [section] page ...
-a, --all find all matching manual pages.
-d, --debug emit debugging messages.
...
-h, --help show this usage message.
```

Здесь видно, что команда man имеет множество параметров, большинство из которых необязательны, кроме параметра page.

> **Примечание:** К сожалению, практически вся помощь дается на английском языке, и указанный способ получения справки может оказаться бесполезным. В этом случае рекомендуем воспользоваться интернетом — существует множество достаточно полных руководств по работе в Linux.

4.4. Права доступа

Каждый пользователь Linux входит в одну основную и, возможно, некоторые дополнительные группы. Узнать подробную информацию о пользователе и его членстве в группах можно с помощью команды id.

Любой файл или каталог в Linux имеет пользователя-владельца и группу-владельца. Права доступа на файл или каталог можно определить для трех категорий: пользователь-владелец, группа-владелец, прочие. Права могут быть следующие: чтение (r), запись (w), выполнение (x), и отсутствие соответствующих прав (-). Таким образом, права на конкретный файл обычно выглядят так:

```
rwxr-xr-x
```

Первые 3 символа — права владельца, затем идут права группы, а последние 3 символа описывают права остальных пользователей. В данном примере видно, что владелец может изменять, запускать на выполнение и читать файл, а все прочие не могут изменять файл, но могут читать и выполнять его. Если рассматривать наличие некоторого права как 1, а его отсутствие как 0, то данная строка примет следующий вид

```
111101101
```

и, представив это число в восьмеричном виде (заменив каждую тройку цифр соответствующим числом от 0 до 7), получим более краткую запись прав доступа

```
755
```

Для изменения прав доступа служит команда chmod. В качестве параметра она может принимать права доступа в краткой записи и файл, к которому применяются эти права. Например, команда

```
chmod 700 test
```

запретит любой доступ к файлу test всем, кроме владельца, а команда

```
chmod 777 test
```

разрешит всем полный доступ.

4.5. Дисковые квоты

На кластере КарНЦ РАН установлены ограничения на использование дискового пространства. Команда quota позволяет узнать эти ограничения и объем, занимаемый файлами пользователя в текущий момент. Например, если пользователь tester выполнит эту команду, он увидит следующую информацию:

```
quota
Disk quotas for user tester (uid nnnn):
    Filesystem   blocks  quota      limit      grace  files  quota  limit  grace
     /dev/md0    1024    52428800   57761680   29     0      0
```

Здесь указано, что на устройстве /dev/md0 он хранит 29 файлов и каталогов, которые суммарно занимают 1024 блока (размер блока — 1 Кб), квота составляет 52428800 блоков (или 50 Гб[8]), а

8 50 Гб = 50*1024 Мб = 51200*1024 Кб = 52428800 Кб

жесткий лимит — 57761680 блоков (или 55 Гб).

Удобнее использовать эту команду с параметром -s, который выводит информацию в более удобном виде, например:

```
quota -s
Disk quotas for user tester (uid nnnn):
    Filesystem   blocks   quota     limit     grace  files  quota  limit  grace
     /dev/md0    1M       51200M    56408M    29     0      0
```

Если пользователь tester превысит квоту в 50 Гб, то он получит на свою электронную почту сообщение примерно такого содержания:

```
Здравствуйте!

Вы превысили квоту на использование дискового пространства на кластере КарНЦ РАН.
Размер квоты Вы можете посмотреть командой quota -s.

Пожалуйста, удалите ненужные файлы на следующих файловых системах:

Домашний каталог /home/имя_пользователя (/dev/md0)

                                Block limits                    File limits
Filesystem      used            soft          hard        grace  used soft  hard  grace
/dev/md0  +-    53428800        52428800      57761680    6days  124  0     0

Спасибо!
                Служба поддержки кластера КарНЦ РАН
```

Здесь написано, что в домашнем каталоге пользователя файлы занимают 53428800 Кб, что превышает квоту в 50 Гб. Однако, в течении льготного периода, указанного в поле **grace** (в данном случае это 6 дней) пользователь может создавать новые файлы, если общий объем его файлов не превышает жесткого лимита в 55 Гб, при этом каждый день он будет получать новое письмо о превышении квот. После окончания льготного периода пользователь не сможет создавать новые файлы до тех пор, пока общий объем его файлов не станет меньше 50 Гб.

4.6. Основные команды

Еще раз перечислим команды, речь о которых шла выше.

mc	запуск файловой оболочки Midnight Commander;
quota -s	получить информацию о дисковых квотах;
man <object>	руководство по использованию object;
pwd	выдает имя текущего каталога;
cd <dir>	переход в каталог dir;
ls <dir>	список объектов в каталоге dir;

mkdir <dir> создание каталога dir;

rmdir <dir> удаление пустого каталога dir;

rm <file> удаление файла file;

rm -d <dir> удаление пустого каталога dir;

rm -r <dir> удаление каталога dir с подкаталогами;

cp <source> <destination>
копирование файла или каталога;

mv <source> <destination>
перемещение либо переименование файла или каталога;

passwd смена пароля пользователя;

id получение информации о пользователе;

ssh <user>@<host>
удаленный доступ пользователя user на машину с адресом host;

exit завершение сеанса;

scp <user1>@<host1>:<file1> <user2>@<host2>:<file2>
удаленное копирование файлов.

Приложение

Пример сеанса работы на кластере

Предположим, на кластере зарегистрирован пользователь с логином tester. Приведем примерный вариант сеанса работы с кластером. В левом столбце приведены команды пользователя и ответ кластера, справа — наши комментарии. Текст, набранный жирным шрифтом обозначает команды пользователя. Каждая команда при вводе занимает одну строку, даже если на странице она была разнесена на 2 строки, как первая из команд scp.

1. вход на кластер

ssh tester@192.168.1.230 — начинаем сеанс удаленного доступа

Password: — система запрашивает пароль

tester@krc-master:~> — авторизация прошла успешно, пользователь tester находится на кластере в своем домашнем каталоге

2. создание каталога

tester@krc-master:~>**mkdir scr** — создаем каталог src

3. просмотр списка файлов

tester@krc-master:~>**ls s*** — проверим список файлов, начинающихся с символа s

src — да, каталог src есть

4. переход в каталог

tester@krc-master:~>**cd ./src** — перейдем в каталог src

tester@krc-master:~/src> — обратите внимание, как изменилось приглашение командной строки

5. удаленное копирование файла

tester@krc-master:~/src>**scp myname@myhost:/home/myname/progs/test.cpp /home/tester/src** — скопируем файл test.cpp с компьютера myhost

Password: — запрос пароля на компьютер myhost для пользователя myname

test.cpp 100% 2834 0.9KB/s 00:00 скопировано

6. компиляция программы

tester@krc-master:~/src>**mpicxx -o test ./test.cpp** — скомпилируем файл test.cpp в исполняемый файл test

7. просмотр списка файлов

tester@krc-master:~/src>**ls test*** — проверим, какие файлы, начинающиеся с подстроки test, есть в текущем каталоге

test.cpp test — да, появился файл test

8. постановка задачи в очередь

tester@krc-master:~/src>**screen**	запуск виртуального терминала
tester@krc-master:~/src>**run -n 10 ./test**	поставим задачу test в очередь main
tester@krc-master:~/src>**squeue**	проверим состояние очереди
tester@krc-master:~/src>Ctrl+A, Ctrl+D	“отключение” сеанса

Спустя какое-то время…

9. задача выполнена

tester@krc-master:~/src> **screen -r**	возвращение к сеансу
SLURM Job_id=341 Name=hostname Ended	расчет окончен, можно смотреть результаты
tester@krc-master:~/src>Ctrl+D	“завершение” вирт. терминала

10. просмотр списка файлов

tester@krc-master:~/src>**ls**	посмотрим, какие файлы появились

11. удаленное копирование файлов

tester@krc-master:~/src>**scp 1.txt myname@myhost:/home/myname/prog**

скопируем файлы результатов обратно на рабочий компьютер

Password: снова запрос пароля

12. завершение сеанса работы с кластером

tester@krc-master:~/src>**exit** закончим сеанс работы