Вычислительный кластер «Академик В.М. Матросов»

Руководство пользователя

Версия 2.2

Авторы: Новопашин А.П., Сидоров И.А., Скоров В.В., Костромин Р.О

Оглавление

Предисловие

Данное руководство[1] предназначено для пользователей вычислительного кластера (ВК) «Академик В.М. Матросов». Цель руководства – предоставить базовые сведения о кластере, методах и средствах работы с ним.

[1] Данное руководство не является учебным пособием по параллельному программированию. Описание большинства команд, приведенных в руководстве, представлено в сжатом виде.

1. О кластере

Вычислительный кластер – это совокупность вычислительных узлов, объединенных высокоскоростными каналами связи, представляющая с точки зрения пользователя единый вычислительный ресурс. Основное предназначение вычислительного кластера – выполнение большого объема расчетов, с которым не справляются современные персональные компьютеры. По типу архитектуры кластер относится к системам с распределенной памятью («память распределена по узлам»), при этом каждый узел кластера в отдельности представляет собой систему с общей (разделяемой) памятью. Основная характеристика вычислительного кластера – производительность вычислений, которая измеряется числом операций в секунду.

1.1. Аппаратное обеспечение

В состав ВК «Академик В.М. Матросов» входят:

- 120 вычислительных узлов с процессорами x86_64:
  - 60 узлов T-Blade V205S (тип «A»), образующих **сегмент I**
  - 60 узлов в составе 30 лезвий Supermicro SBI-7228R (тип «B»), образующих **сегмент II**
- вычислительный сервер с графическими процессорами (GPU) NVidia Tesla
- вычислительный сервер с сопроцессорами Intel Xeon Phi
- 2 узла доступа (основной и резервный)
- 2 управляющих узла (основной и резервный)
- система хранения данных Panasas ActiveStor 40 ТБ
- коммуникационная сеть – QDR Infiniband
- транспортная и сервисная сети – Gigabit Ethernet
- вспомогательная (инженерная) инфраструктура: система аппаратных шкафов, система энергоснабжения, система бесперебойного электропитания, климатическая система, система автоматического газового пожаротушения и др.

Вычислительный узел – многопроцессорный, многоядерный компьютер, на котором выполняются задания (считаются задачи) пользователей. Задача пользователя может занимать один вычислительный узел, несколько вычислительных узлов или все вычислительные узлы одного сегмента.

Вычислительный узел типа «A» содержит:

- два 16-ядерных процессора AMD Opteron 6276 «Interlagos»: 2.3 ГГц, 16 МБ L3 кэш, 4 FLOP/cycle
- 64 ГБ оперативной памяти DDR3-1600

Вычислительный узел типа «B» содержит:

- два 18-ядерных процессора Intel Xeon E5-2695 v4 «Broadwell»: 2.1 ГГц, 45 МБ L3 кэш, 16 FLOP/cycle
- 128 ГБ оперативной памяти DDR4-2400

Суммарная пиковая (теоретическая) производительность вычислительных узлов сегмента I и сегмента II – **90,24 TFlops**[2]. **Максимальная достигнутая производительность на тесте HPL** (High Performance Linpack) с использованием 2160 ядер вычислительных узлов типа «B» (сегмент II) – **60,13 TFlops** (26 -е место в 26-ой редакции списка Top-50[3], 04.04.2017 г.).

Узлы компиляции. В каждом сегменте выделено по одному узлу для компиляции и тестирования программ перед запуском: узел node001[4] – в сегменте I; узел sm101[5] – в сегменте II.

[2] TFlops – терафлопс – величина, используемая для измерения производительности компьютеров, показывающая, сколько операций с плавающей точкой в секунду выполняет данная вычислительная система (1 терафлопс = 1 триллион операций с плавающей точкой в секунду).

[3] http://top50.supercomputers.ru/?page=archive&rating=26

[4] Вычислительные узлы сегмента I именуются в виде nodexxx, где xxx – порядковый номер узла от 001 до 060.

[5] Вычислительные узлы сегмента II именуются в виде smyyy, где yyy – порядковый номер узла от 101 до 160.

Вычислительный сервер с графическими процессорами (GPU) NVIDIA Tesla содержит:

- четыре 8–ядерных процессора AMD Opteron 6220 3 ГГц
- 256 ГБ оперативной памяти DDR3-1600
- 2 накопителя на жестких дисках объемом 600 ГБ каждый
- 2 модуля ускорения вычислений NVIDIA Tesla C2070 («Fermi»), каждый из которых имеет:
    - 448 ядер с тактовой частотой 1,15 ГГц суммарной пиковой производительностью (на операциях с двойной точностью) 515 GFlops
    - объем встроенной памяти GDDR5 6 ГБ

Вычислительный сервер с сопроцессорами Intel Xeon Phi содержит:

- два 4-ядерных процессора Intel Xeon E5-2609 2.4 ГГц
- 32 ГБ оперативной памяти DDR3-1600
- 2 накопителя на жестких дисках объемом 1 ТБ каждый
- 2 сопроцессора Intel Xeon Phi 5110P «Knights Corner» с архитектурой MIC[6], каждый из которых имеет:
    - 60 ядер с тактовой частотой 1,053 ГГц суммарной пиковой производительностью (на операциях с двойной точностью) 1,01 TFlops
    - объем встроенной памяти GDDR5 8 ГБ

Узел доступа – многопроцессорный компьютер, который позволяет пользователю получить доступ к ресурсам кластера. На данном узле осуществляется подготовка заданий для кластера, работа с входными данными и результатами вычислений.

ВК «Академик В.М. Матросов» имеет в своем составе два узла доступа со следующими характеристиками:

- два 6-ядерных процессора Intel Xeon X5670 2.93 ГГц
- 48 ГБ оперативной памяти DDR3-1333
- 4 накопителя на жестких дисках объемом 300 ГБ каждый

Управляющий узел – многопроцессорный компьютер, который управляет ресурсами вычислительного кластера, организует работу очередей заданий, выполняет мониторинг компонентов вычислительного кластера и ряд других вспомогательных функций. Доступ пользователей к управляющему узлу закрыт.

ВК «Академик В.М. Матросов» имеет в своем составе два управляющих узла со следующими характеристиками:

- два 16-ядерных процессора AMD Opteron 6276 2.3 ГГц
- 64 ГБ оперативной памяти DDR3-1600
- 8 накопителей на жестких дисках объемом 450 ГБ каждый

Система хранения данных (СХД) предназначена для организованного хранения данных и обеспечения высокоскоростного параллельного доступа к ним.

СХД ВК «Академик В.М. Матросов» имеет следующие технические характеристики:

- емкость – 40 ТБ
- совокупная производительность – 150 ГБ/с
- максимальная пропускная способность – 1600/1500 МБ/с (запись/чтение)
- параллельная файловая система – PanFS, обеспечивающая прямой доступ вычислительных модулей к данным хранилища без использования дополнительных серверов

Коммуникационная сеть обеспечивает коммуникации между процессами параллельных приложений на вычислительных узлах с использованием протокола MPI (Message Passing Interface), а также доступ узлов к СХД.

[6] http://software.intel.com/ru-ru/xeon-phi/mic

Коммуникационная сеть ВК «Академик В.М. Матросов» построена на основе технологии QDR[7] (Quad Data Rate) InfiniBand и имеет топологию Fat Tree. Каждый вычислительный узел подключен к коммуникационной сети с помощью коммутатора Mellanox IS5200, обладающего следующими техническими характеристиками:

- агрегированная производительность – 17,28 Тб/с
- задержка маршрутизации от порта к порту – от 100 до 300 нс
- эффективная пропускная способность 4х порта – 32 Гб/с

Транспортная и сервисная сети предназначены для обеспечения служебных коммуникаций между всеми узлами вычислительного кластера и построены на основе технологии Gigabit Ethernet.

1.2. Программное обеспечение

В качестве операционной системы на всех узлах ВК «Академик В.М. Матросов» используется CentOS Linux.

В комплект системного программного обеспечения кластера входит набор инструментов для параллельного программирования Intel Cluster Studio XE 2013[8], включающий следующие компоненты:

- Intel C++ Compiler XE и Fortran Compiler XE – компиляторы с поддержкой OpenMP;
- Intel Debugger – отладчик;
- Intel Math Kernel Library – библиотека производительных математических функций;
- Intel Integrated Performance Primitives – библиотека высокооптимизированных подпрограмм на языке C++;
- Intel Threading Building Blocks – библиотека шаблонов на языке C++ для эффективной реализации высокоуровневого параллелизма;
- Intel MPI Library – реализация протокола MPI (Message Passing Interface);
- Intel MPI Benchmarks – средства оценки производительности основных функций передачи сообщений;
- Intel Trace Analyzer and Collector – средства анализа производительности MPI-приложений;
- Intel Inspector XE – средства выявления ошибок памяти и многопоточности;
- Intel VTune Amplifier XE – профилировщик одно- и многопоточных программ.

Для программирования графических ускорителей, установленных на выделенном узле ВК «Академик В.М. Матросов», пользователям предоставляется инструментарий NVIDIA CUDA Toolkit[9]. Инструментарий обеспечивает комплексную среду разработки программ на языках C и C++, включает в себя специализированный компилятор, математические библиотеки и инструменты для отладки и оптимизации производительности приложений.

В числе прочих средств разработки, установленных на ВК «Академик В.М. Матросов»: GNU C++, GNU Fortran, GNU GDB, Perl, Python.

ВК «Академик В.М. Матросов» является ресурсом коллективного пользования, работающим в режиме пакетной обработки заданий. В качестве системы управления заданиями (менеджера ресурсов) используется система Torque[10].

[7] Вычислительные узлы типа «B», входящие в состав одного вычислительного модуля (в одном модуле – 20 узлов), объединены между собой с помощью встроенного в модуль InfiniBand-коммутатора стандарта FDR (Fourteen Data Rate), обеспечивающего эффективную пропускную способность 4х-порта до 56 Гб/с.
[8] https://software.intel.com/en-us/articles/intel-cluster-studio-xe-documentation
[9] https://developer.nvidia.com/cuda-toolkit
[10] http://www.adaptivecomputing.com/products/open-source/torque/

2. Работа пользователя с кластером

Цикл работы пользователя с кластером состоит из следующих этапов:

- Удаленный вход на кластер.
- Копирование данных между кластером и компьютером пользователя.
- Редактирование исходных текстов программ.
- Компиляция программ.
- Запуск задач и работа с очередью.
- Завершение сеанса.

Первый и последний этапы являются обязательными: пользователь должен получить доступ к кластеру и рано или поздно завершить сеанс работы с ним. Все остальные действия могут варьироваться: например, пользователь может редактировать файлы с исходными данными и текстами программ на своем персональном компьютере или заходить на кластер только для того, чтобы оценить результаты счета задачи и т.п. Некоторые действия могут выполняться параллельно: необязательно дожидаться завершения счета одной задачи, чтобы поставить в очередь на выполнение следующую.

Рассмотрим подробно основные этапы работы пользователя с ВК «Академик В.М. Матросов».

2.1. Удаленный вход на кластер

Физически ВК «Академик В.М. Матросов» располагается в специально выделенном помещении, непосредственный доступ в которое ограничен. Пользователи получают доступ к ресурсам кластера посредством удаленного входа[11] на узел доступа[12]. Предварительно каждый пользователь должен пройти процедуру регистрации[13] (получить логин и пароль для входа).

В состав ВК включены два узла доступа, работающие по схеме «основной - резервный»:
Внешний IP-адрес основного узла доступа (access1): **84.237.30.108**. Символьное имя основного узла доступа: **matrosov.icc.ru**.
Внешний IP-адрес резервного узла доступа (access2): **84.237.30.109**. Символьное имя резервного узла доступа: **matrosov2.icc.ru**.

Процедура входа на узел доступа зависит от операционной системы компьютера пользователя. Ниже описаны два основных варианта.

Для пользователей Linux

Для доступа к кластеру наберите в командной строке:

```
ssh username@hostname
```

username – ваш логин, **hostname** – IP-адрес (или символьное имя) узла доступа.

На соответствующий запрос системы введите пароль, полученный при регистрации (при первом входе на кластер рекомендуется сменить пароль командой **passwd**, и в дальнейшем менять его не реже одного раза в месяц).

Пример:
Для входа на ВК пользователь **tester** должен набрать:

```
ssh tester@84.237.30.108
```

[11] Удаленный вход осуществляется с использованием протокола SSH-2 (Secure Shell. Version 2.x).
[12] Для удаленного доступа к вычислительному серверу с GPU NVidia Tesla воспользуйтесь инструкцией http://hpc.icc.ru/documentation/GPU_ug.rtf. Для удаленного доступа к вычислительному серверу с сопроцессорами Intel Xeon Phi воспользуйтесь инструкцией http://hpc.icc.ru/documentation/PHI_ug.rtf .
[13] http://hpc.icc.ru/foruser/registration.php

либо

```
ssh tester@matrosov.icc.ru
```

Для пользователей Windows

Для удаленного подключения из Windows необходимо использовать специальную программу – SSH-клиент, например, PuTTy[14].

Для установки PuTTY скачайте на свой компьютер файл **putty.exe** и запустите его. В появившемся окне «Session» в строке «Host Name (or IP address)» укажите символьное имя (или IP-адрес) узла доступа, выберите протокол SSH и начните сессию. В открывшемся окне с приглашением «login as:» наберите имя пользователя, нажмите **Enter** и далее в ответ на запрос системы введите пароль, полученный при регистрации.

Если аутентификация прошла успешно, на экране появится приглашение командной строки, например:

```
tester@access1 ~ $
```

Это означает, что вы подключились к кластеру и находитесь в своем домашнем каталоге **/home/<username>** на узле доступа access1.

Примечание: При первой попытке входа на кластер операционная система вашего компьютера выдаст предупреждение о том, что ей неизвестен хост с адресом **84.237.30.108**, и спросит, согласны ли вы продолжить подключение:

```
The authenticity of host '84.237.30.108 (84.237.30.108)' can't be
established.
RSA key fingerprint is
c9:82:1e:0f:95:07:70:ec:14:97:d7:34:31:50:63:76.
Are you sure you want to continue connecting (yes/no)?
```

Для подключения наберите слово **Yes** и нажмите **Enter** (при использовании PuTTy – нажмите кнопку **Да**), хост **84.237.30.108** будет добавлен к списку известных хостов:

```
Warning: Permanently added '84.237.30.108' (RSA) to the list of
known hosts.
```

2.2. Структура файловой системы кластера

Для работы на ВК «Академик В.М. Матросов» необходимо иметь представление о логической структуре файловой системы, при которой часть данных располагается на локальных жестких дисках узлов, часть данных – в СХД Panasas.

Локальные системные каталоги – системные каталоги операционной системы Linux, присутствующие на жестком диске каждого узла кластера:

Корневой каталог **/** является основой файловой системы. Все остальные каталоги и файлы располагаются в рамках структуры (дерева), порожденной корневым каталогом, независимо от их физического местонахождения.

Каталоги **/etc, /boot** содержат конфигурационные файлы, данные для загрузки операционной системы.

Каталоги **/sbin**, **/bin**, **/lib**, **/usr**, **/var** содержат перечень системных и прикладных утилит операционной системы Linux, библиотеки, а также данные для системных процессов.

Каталог **/tmp** используется для хранения временных данных системных процессов.

[14] https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Сетевые каталоги – каталоги **/share** и **/home**, физически расположенные в СХД Panasas и доступные с любого узла кластера по сетевому интерфейсу QDR Infiniband.

Каталог **/share** содержит системные программные средства, используемые для компиляции пользовательских программ и организации их параллельного выполнения на узлах кластера.

Домашний каталог пользователя **/home/<username>** – каталог, в котором пользователь **username** размещает программы и данные, выполняет компиляцию программ, осуществляет подготовку задач к запуску, хранит результаты расчетов и т.д.

Локальные пользовательские каталоги /store и /store2 – физически размещены на локальных дисках вычислительных узлов кластера.

Примечание: Каталог **/store2** доступен только на определенных узлах сегмента II (подробности по использованию **/store2** см. в разделе 2.6.2).

Размер дискового пространства в каталоге **/store**:
- для узлов сегмента I – не менее 400 ГБ (тип дисков – HDD);
- для узлов сегмента II – не менее 200 ГБ (тип дисков – SSD).

Размер дискового пространства в каталоге **/store2**:
- для узлов сегмента II (sm101-120) – не менее 400 ГБ (тип дисков – HDD).

При интенсивной работе задачи с данными, находящимися в домашнем каталоге пользователя **/home/<username>** (физически размещен в СХД), пользователь может наблюдать замедление счета, связанное с задержками доступа к данным по сети. Для решения этой проблемы рекомендуется заранее произвести перемещение (копирование) файлов с исходными данными из сетевого каталога **/home/<username>** в локальный каталог **/store/<username>** или **/store2/<username>** на выделенном для решения задачи вычислительном узле. При запуске задачи потребуется указать абсолютный путь к этим данным.

Примечание: Если каталог **/store/<username>** отсутствует, необходимо создать его командой

```
$ mkdir /store/<username>
```

Аналогично – для **/store2/<username>** .

Запись результатов счета задачи (если их объем превышает 100 МБ) также рекомендуется осуществлять в локальный каталог **/store/<username>** или **/store2/<username>**. По завершении счета задачи вы можете скопировать полученные результаты в свой домашний каталог.

Примечание: С узлов доступа (access1, access2) работа с данными, расположенными в локальных каталогах **/store** вычислительных узлов, осуществляется по абсолютному пути **/store/nodes/<nodename>/<username>**, где **nodename** – символьное имя узла.

Для **/store2** – путь **/store2/nodes/<nodename>/<username>** .

Пример:

Копирование файла из домашней директории в локальный каталог **/store** для узла node002:

```
tester@access1 ~ $ cd /store/nodes/node002
tester@access1 /store/nodes/node002 $ mkdir tester
tester@access1 /store/nodes/node002 $ ls -l
drwxr-xr-x 2 tester users 4096 Oct 3 11:10 tester
tester@access1 /store/nodes/node002 $ cd tester/
tester@access1 /store/nodes/node002/tester $ cp ~/data/bigfile.in ./
```

Вход[15] на node002 и проверка содержимого каталога:

```
tester@access1 /store/nodes/node002/tester $ ssh node002
[tester@node002 tester]$ cd /store/tester/
[tester@node002 tester]$ ls -lh
-rw-r--r-- 1 tester users 2Gb Oct 3 11:14 bigfile.in
```

2.3. Копирование файлов между кластером и компьютером пользователя

Работая в режиме удаленного доступа, пользователь не имеете возможности выполнять прямое копирование файлов на кластер с внешних носителей и в обратном направлении. Для передачи файлов необходимо использовать команды **scp** или **sftp** при подключении с Unix-системы, либо специальные утилиты, предназначенные для удаленного копирования файлов (например, WinSCP[16]), при подключении с компьютера на базе Windows.

Рассмотрим действия, необходимые для копирования файлов между двумя компьютерами с использованием протокола SCP[17].

Для пользователей Linux

Упрощенный формат[18] команды **scp**:

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

Команда **scp** выполняет копирование файлов между двумя компьютерами сети. Первый параметр – копируемый файл на компьютере host1, второй параметр – путь его размещения на компьютере host2. Если для копирования требуется подключение к удаленному компьютеру, необходимо указать логин и пароль пользователя, имеющего к нему доступ.

Примечание: При подключении к удаленному хосту команда **scp** использует протокол SSH, что требует от пользователя указания пароля для удаленного доступа.

Примеры:
Для копирования файла **myprog.cpp** из домашнего каталога **/home/tester/** на кластере в каталог **/home/user/test** рабочего компьютера пользователю **tester** потребуется набрать команду:

```
scp tester@84.237.30.108:/home/tester/myprog.cpp /home/user/test/
```

В данном примере не используются необязательные части второго параметра, так как предполагается, что пользователь **tester** выполняет копирование, находясь за своим рабочим компьютером (нет необходимости указывать его сетевой адрес и логин для подключения).

Для копирования файла **myprog.cpp** в обратном направлении (с рабочего компьютера в домашний каталог на кластере) пользователь **tester** должен набрать команду:

```
scp /home/user/test/myprog.cpp tester@84.237.30.108:/home/tester/
```

Для копирования директории вместе со всем ее содержимым необходимо использовать параметр **–r**:

```
scp -r /home/user/test tester@84.237.30.108:/home/tester/
```

[15] Вход пользователя на вычислительный узел возможен только после резервирования данного узла системой Torque (см. раздел 2.6).
[16] https://winscp.net
[17] Secure Copy Protocol – протокол защищенного копирования.
[18] Здесь и далее выражение в квадратных скобках в описании команды является необязательным.

Для пользователей Windows

Рассмотрим процедуру копирования файлов на примере программы WinSCP. После установки и запуска программа WinSCP запросит у вас (по аналогии с PuTTY) данные для подключения: символьное имя (IP-адрес) узла доступа, логин и пароль. После успешной аутентификации откроется окно, подобное Windows Explorer, либо двухпанельное окно в стиле Norton Commander (в зависимости от варианта, выбранного при установке программы). После этого вы получите возможность копировать файлы между рабочим компьютером и кластером привычным для вас способом.

Кроме копирования программа WinSCP позволяет выполнять другие операции с файлами: редактирование, переименование, удаление, изменение прав доступа и прочие.

Монтирование удаленных[19] каталогов

Для того чтобы получить доступ к содержимому удаленного каталога, можно воспользоваться утилитой SSHFS, позволяющей монтировать удаленные каталоги в указанную вами директорию по защищенному протоколу SSH.

Пример:
Для монтирования домашнего каталога, расположенного на вычислительном кластере Blackford[20], вам необходимо выполнить следующие действия:

1) Создать директорию, в которую будет осуществляться монтирование удаленного каталога:

```
mkdir ~/blackford_dir
```

2) Смонтировать удаленного каталог, указав абсолютный путь каталога на целевой машине и непосредственно каталог монтирования:

```
sshfs tester@blackford.icc.ru:/home/tester ~/blackford_dir
```

После выполнения указанных действий вы сможете работать с файлами в директории **~/blackford_dir** как с локальными файлами вашей домашней директории.

Примечание: Монтирование удаленных каталогов выполняется только на узлах доступа и предназначено для упрощения доступа к файлам, расположенным на удаленных ресурсах. На вычислительных узлах смонтированные директории будут отсутствовать. Таким образом, задачи пользователя не должны взаимодействовать со смонтированными директориями – необходимо выполнить предварительное копирование файлов с удаленных ресурсов в локальный каталог.

2.4. Редактирование файлов на кластере

Процесс разработки программного обеспечения содержит этап редактирования файлов с исходными данными и текстами программ. Можно вносить необходимые изменения в файлах на своем рабочем компьютере, а затем копировать их на кластер, однако в ряде случаев удобнее выполнять редактирование прямо на кластере. Для этого можно воспользоваться любым текстовым редактором. Одним из наиболее простых в освоении является редактор **mcedit**, встроенный в оболочку Midnight Commander.

Midnight Commander – файловый менеджер с текстовым интерфейсом, функционирующий в соответствии с теми же принципами, что и Norton Commander, Far Manager, Total Commander.

[19] Под словом «удаленных» подразумевается «расположенных удаленно».
[20] http://hpc.icc.ru/hardware/blackford.php

Midnight Commander поддерживает все основные операции с файлами и каталогами: создание, удаление, копирование, перемещение, изменение прав доступа и другие. Для запуска Midnight Commander используется команда **mc**.

Экран Midnight Commander состоит из двух панелей, в которых отображаются списки каталогов и файлов. Панель, в которой находится курсор, является активной. Выбор активной панели осуществляется с помощью клавиши **Tab**.

В нижней части экрана расположена командная строка и панель горячих клавиш **F1-F10:**

- **F1** – помощь;
- **F2** –меню пользователя;
- **F3** – просмотр файла (каталога), на который указывает курсор;
- **F4** – редактирование выбранного файла с помощью встроенного редактора **mcedit**;
- **F5** – копирование выбранного файла (каталога);
- **F6** – переименование/перемещение выбранного файла (каталога);
- **F7** – создание каталога;
- **F8** – удаление выбранного файла (каталога);
- **F9** – вызов верхнего меню;
- **F10 –** выход из Midnight Commander.

Для выполнения групповых операций с несколькими файлами и каталогами активной панели предварительно отметьте их с помощью клавиши **Ins**. Для создания файла используйте сочетание клавиш **Shift+F4** (вызов редактора **mcedit**). Для сохранения изменений нажмите клавишу **F2**. Для выхода из редактора используйте клавишу **F10** или **Esc**.

2.5. Компиляция программ

Для компиляции MPI-программ рекомендуется использовать команды, подключающие автоматически заголовочные файлы и библиотеки MPI:

`mpicc`	– для программ на C
`mpicxx`	– для программ на C++
`mpif77 / mpif90`	– для программ на Fortran 77/90

Перечисленные команды позволяют осуществлять сборку программ, используя как компиляторы Intel, так и свободно распространяемые компиляторы семейства GNU Compiler Collection (кроме **mpif90**).

Для явного указания компиляторов Intel используйте следующие ключи:

`mpicc -cc=icc`	– для программ на C
`mpicxx -cxx=icpc`	– для программ на C++
`mpif77 -fc=ifort`	– для программ на Fortran 77/90

Для явного указания компиляторов GCC используйте дополнительные ключи:

`mpicc -cc=gcc`	– для программ на C
`mpicxx -cxx=g++`	– для программ на C++
`mpif77 -fc=g77`	– для программ на Fortran 77

Опция **–o name** позволяет задать имя получаемого исполняемого файла (по умолчанию **a.out**).

Для оптимизации используйте дополнительные ключи компиляторов, например:

```
mpif77 –fast –o program program.f
```

Для получения только объектного модуля используйте опцию **–c**:

```
mpicc –c program2.c –o program.o
```

Для сборки многомодульных приложений пользуйтесь утилитой **make**[21] и соответствующими

[21] http://www.gnu.org/software/make/manual/

файлами спецификаций **Makefile**.

> **Примечание:** Ввиду существенных различий характеристик вычислительных узлов и узлов доступа компиляцию и тестирование программ необходимо выполнять на специально выделенных для этих целей узлах компиляции: node001 или sm101.

Пример:

```
tester@access1 ~ $ ssh sm101
[tester@sm101 ~]$ cd examples/src
[tester@sm101 src]$ make
Вывод сообщений сборки…
[tester@sm101 src]$ ./cpi
Process 0 on sm101
pi is approximately 3.1416009869231254, Error is 0.0000083333333323
wall clock time = 0.000072
```

2.6. Запуск задач и работа с очередью

Для запуска задач (прикладных программ) на счет и последующего контроля за их выполнением в ВК «Академик В.М. Матросов» используется система управления заданиями Torque[22] – свободно распространяемая версия системы пакетной обработки Portable Batch System (PBS). Эта система автоматически распределяет задания пользователей по свободным вычислительным ресурсам кластера, выполняет планирование очередей заданий, осуществляет контроль за исполнением заданий.

2.6.1. Просмотр очереди заданий

Следить за состоянием задания в очереди позволяет команда **tasks**. Запуск этой команды без параметров выдает на экран сводную информацию о состоянии вычислительных ресурсов в сегментах I и II и списки заданий в соответствующих очередях в табличной форме.
Столбы таблиц со списками заданий имеют следующие значения:

- **Job ID** – уникальный идентификатор задания;
- **Username** – имя владельца задания;
- **Jobname** – имя задачи;
- **S** – статус задания (**Q** – находится в очереди, **R** – выполняется, **C** – завершилось (информация о выполненных заданиях хранится 5 мин), **E** – ошибка при выполнении задания);
- **Nodes** – количество узлов, запрошенное для выполнения задания;
- **Cores** – количество ядер, запрошенное для выполнения задания;
- **Queue time** – время добавления задания в очередь;
- **Start time** – время старта задачи на вычислительных узлах;
- **Run time** – время выполнения задания на вычислительных узлах;
- **Time left** – время, оставшееся до завершения задания.
- **Resources** – номера узлов, используемых для выполнения задания.

> **Примечание:** Утилита **task** не входит в стандартный пакет Torque и является средством, разработанным сотрудниками ИДСТУ СО РАН. Формат вызова и функциональные возможности данной утилиты могут отличаться от приведенных выше.

Для просмотра всех задач в очереди (в том числе других пользователей) необходимо запустить команду **tasks** с ключом **-a**. Для просмотра состояния ресурсов и заданий в очереди отдельного сегмента используйте команду **tasks.amd** (для сегмента I) и команду **tasks.intel** (для сегмента II).

> **Примечание:** Если при выполнении задания произошла ошибка (статус задания **E**), и вы не знаете ее причину, обратитесь за разъяснением в службу технической поддержки с подробным описанием своих действий.

[22] http://www.adaptivecomputing.com/products/open-source/torque/

Ниже показан пример[23] выполнения команды **tasks**.

```
================================== Segment I ==================================
60 nodes with 2x16 AMD Opteron 6276 Interlagos 2.3 GHz; 64 GB RAM per node
Nodes   [Total: 59     Free: 48     Busy: 1      Down: 10   ]
Cores   [Total: 1888   Free: 1536   Busy: 32     Down: 320  ]
--------------------------------------------------------------------------------
Job ID  Username     Jobname          S  Nodes Cores Run time     Resources
------- ------------ ---------------- -- ----- ----- ------------ --------------
454490  tester       example1         C  10    320   00:00:02     [002-011]
454491  tester       example2         R  1     32                 [012]
454492  tester       example3         Q  1     32
--------------------------------------------------------------------------------

================================== Segment II ==================================
60 nodes with 2x18 Intel Xeon E5-2695 v4 Broadwell 2.1 GHz; 128 GB RAM per node
Nodes   [Total: 59     Free: 54     Busy: 0      Down: 5    ]
Cores   [Total: 2124   Free: 1944   Busy: 0      Down: 180  ]
--------------------------------------------------------------------------------
Job ID  Username     Jobname          S  Nodes Cores Run time     Resources
------- ------------ ---------------- -- ----- ----- ------------ --------------
454489  tester       example          R  1     36    02:13:11     [131]
--------------------------------------------------------------------------------
```

2.6.2. Постановка заданий в очередь

Для того чтобы запустить задачу на счет потребуется подготовить специальный файл с описанием задания для Torque в виде командного скрипта и поставить его в очередь командой **qsub**. В этом файле пользователь должен указать необходимые для решения задачи ресурсы: число узлов, число процессорных ядер в каждом из них, объем оперативной и дисковой памяти, время выполнения задания.

Если количество и тип запрашиваемых ресурсов не противоречит конфигурации имеющихся ресурсов (т.е. вы не просите ресурсов больше, чем доступно в системе) и действующей политике выделения ресурсов[24], задание будет поставлено в очередь. После того как освободится требуемое количество ресурсов, Torque запустит задачу на счет.

В качестве примера ниже приведен скрипт запуска программы **sleep**, которая во время счета занимает не более 5 минут одно процессорное ядро.

Скрипт: sleep.pbs

```
#!/bin/bash
#PBS -N sleep
#PBS -l nodes=1:ppn=32
#PBS -l walltime=05:00

cd $PBS_O_WORKDIR

echo "Start date: `date`"
sleep 60
echo " End date: `date`"
```

Постановка задания в очередь осуществляется с помощью команды **qsub.amd** (для сегмента I) и **qsub.intel** (для сегмента II). Команда **qsub** добавляет задание в очередь сегмента I.

Пример:

```
qsub.intel sleep.pbs
qsub.amd sleep.pbs
qsub sleep.pbs
```

[23] Столбцы таблиц здесь и далее показаны выборочно.

[24] Актуальная информация о действующих квотах на выделение ресурсов размещена на сайте ЦКП ИСКЦ по адресу https://hpc.icc.ru/foruser/queue_quotas.php

Строки файла задания, начинающиеся с **#PBS**, содержат параметры для команды **qsub**. Эти параметры можно также указывать явно при вызове **qsub**. Например:

```
qsub -N sleep -l nodes=1:ppn=32 -l pvmem=100mb -l walltime=5:00 sleep.pbs
```

Параметры, отвечающие за выделение ресурсов, можно указывать через запятую:

```
qsub -l nodes=1:ppn=32,pvmem=100mb,walltime=5:00 sleep.pbs
```

Ниже приведены основные параметры команды **qsub**.

- **-d path**
  Рабочая директория задачи. Если директория явно не задана, рабочей считается домашняя директория пользователя.
- **-e path**
  -o path
  Имена файлов ошибок (**stderr**) и стандартного вывода (**stdout**). По умолчанию: **<имя_задачи>.e<job_id>** и **<имя_задачи>.o<job_id>** в текущей директории.
- **-j oe**
  -j eo
  Объединение файлов вывода/ошибок: **oe** – файлы объединяются в стандартный файл вывода, **eo** – в файл ошибок.
- **-m aben**
  События, при наступлении которых происходит отправка уведомлений на e-mail: **a** – в случае аварийного завершения задачи, **b** – в момент запуска задачи, **e** – в момент завершения задачи, **n** – не отправлять уведомления. Можно указать несколько букв из **a**, **b**, **e** или одну букву **n**. По умолчанию используется **a**.
- **-M e-mail**
  Адрес получателя или список адресов получателей (через запятую), которым будут отправлены уведомления.
- **-N name**
  Имя задачи.
- **-l resource_list**
  Перечень запрашиваемых ресурсов:
  - **nodes=M:ppn=P**
    Запрашивается M узлов[25], на каждом из которых будет использовано P процессорных ядер (P должно находиться в диапазоне от 1 до 32 – для сегмента I; от 1 до 36 – для сегмента II). Например, **nodes=3:ppn=32** – запрос трех узлов с 32 процессорными ядрами на каждом узле, **nodes=6:ppn=16** – запрос шести узлов по 16 процессорных ядер на каждом узле (в этом случае оставшиеся 16 процессорных ядер на узле могут быть использованы для решения других задач). Если атрибут **ppn** не указан, то по умолчанию выделяется одно процессорное ядро.
  - **pmem=size**
    pvmem=size
    Размер физической и виртуальной памяти соответственно в расчете на один процесс. Размер указывается с помощью целого числа и суффикса: **b**, **kb**, **mb**, **gb**.
  - **walltime=time**
    Максимальное время счета задачи (максимальное время выполнения задания)[26], по истечении которого выполнение программы будет завершено. По умолчанию – 24 часа. Пример: **walltime=1:45:00** указывает на завершение выполнения задания через 1 час 45 минут.

[25] Максимальное количество узлов, доступных для одной задачи, определяется действующей политикой выделения ресурсов: https://hpc.icc.ru/foruser/queue_quotas.php

[26] Максимальное время счета задачи определяется действующей политикой выделения ресурсов: https://hpc.icc.ru/foruser/queue_quotas.php

Примечание: При формировании файлов заданий и команд для запуска заданий следует учитывать различия узлов сегмента I и сегмента II по количеству процессорных ядер. При использовании всех ядер узла сегмента I необходимо указать ppn=32, при использовании всех ядер узла сегмента II – ppn=36.

Примечание: Для выделения задаче вычислительных узлов, на которых доступен дополнительный пользовательский каталог **/store2** емкостью не менее 400 ГБ (см. раздел 2.2), необходимо указать атрибут **hdd** (nodes=M:hdd:ppn=P).

В следующем примере показан скрипт запуска задачи, использующей коммуникационную библиотеку MPI. Скрипт запрашивает 2 вычислительных узла, на каждом из которых будет задействовано 32 процессорных ядра. Общее число требуемых процессорных ядер для задачи – 64. Максимальное время счета – 10 минут.

Скрипт: cpi.pbs

```
#!/bin/bash

#PBS -N ExampleCPI
#PBS -l nodes=2:ppn=32,walltime=00:10:00

cd $PBS_O_WORKDIR

/share/apps/bin/mpiexec ~/examples/src/cpi
```

Для запуска задачи, использующей одновременно MPI и OpenMP, в скрипте запуска необходимо явно указать, какое количество MPI-процессов должно быть запущено на каждом узле, и какое количество OpenMP-потоков должно быть доступно каждому процессу. В приведенном ниже примере показаны параметры запуска на каждом из 3-х узлов по 18 MPI-процессов с последующим распараллеливанием каждого процесса на 2 потока. При этом для задачи будут выделены узлы, на которых доступен дополнительный локальный пользовательский каталог /store2 (указан атрибут hdd).

Скрипт: cpi.pbs

```
#!/bin/bash

#PBS -N MpiOpenMP
#PBS -l nodes=3:hdd:ppn=36,walltime=00:10:00

cd $PBS_O_WORKDIR

/share/apps/bin/mpiexec -perhost 18 -genv OMP_NUM_THREADS 2 ~/mpi_openmp_app
```

2.6.3. Получение информации о занятости ресурсов кластера

Просмотреть информацию о занятости вычислительных ресурсов кластера можно с помощью команды **qfree**. Пример вывода команды **qfree** содержит два раздела: данные о загрузке вычислительных узлов и данные о загрузке процессорных ядер на вычислительных узлах:

```
============================== Segment I ======================================
60 nodes with 2x16 AMD Opteron 6276 Interlagos 2.3 GHz; 64 GB RAM per node
Nodes   [Total: 59     Free: 47     Busy: 2      Down: 10   ]
Cores   [Total: 1888   Free: 1504   Busy: 64     Down: 320  ]
============================== Segment II =====================================
60 nodes with 2x18 Intel Xeon E5-2695 v4 Broadwell 2.1 GHz; 128 GB RAM per node
Nodes   [Total: 59     Free: 54     Busy: 0      Down: 5    ]
Cores   [Total: 2124   Free: 1944   Busy: 0      Down: 180  ]
```

Примечание: Утилита **qfree** не входит в стандартный пакет Torque и является средством, разработанным сотрудниками ИДСТУ СО РАН. Формат вызова и функциональные возможности данной утилиты могут отличаться от приведенных выше.

2.6.4. Получение выделенного узла

В некоторых случаях пользователю может потребоваться выделенный узел для запуска задач в, так называемом, «ручном режиме» с целью отладки программ, запуска многопоточных приложений, использующих общую память узла и т.п. Для получения выделенного узла пользователю доступна утилита **qgetnode**.
Пример использования утилиты:

```
tester@access1 ~ $ qgetnode
qsub: waiting for job 622.master to start
qsub: job 622.master ready
[tester@node027 ~]$
```

В данном примере после вызова команды **qgetnode** происходит формирование и добавление в очередь нового задания, поиск и резервирование свободного узла, автоматическое перенаправление пользователя на выделенный узел. Время использования выделенного узла по умолчанию составляет 1 час и может быть изменено параметром **-walltime**. Максимальное значение данного параметра – 24 часа.

Пример:

```
tester@access1 ~ $ qgetnode -waltime "02:00:00"
```

Получение выделенного узла с помощью команды **qgetnode** подразумевает резервирование всех ресурсов вычислительного узла (32 процессорных ядра, 64 ГБ оперативной памяти), после чего ответственность за эффективное использование полученных ресурсов (запуск 32 экземпляров однопоточных программ, запуск программ с 32 потоками и т.п.) ложится на пользователя. При регулярном несоблюдении данного условия доступ пользователя к команде **qgetnode** может быть приостановлен.

Примечание: Утилита **qgetnode** не входит в стандартный пакет Torque и является средством, разработанным сотрудниками ИДСТУ СО РАН. Формат вызова и функциональные возможности данной утилиты в дальнейшем могут отличаться от приведенных выше.

2.6.5. Удаление заданий из очереди

Для удаления задания из очереди используется команда **qdel** с перечислением идентификаторов заданий, которые необходимо удалить. Например:

```
qdel 599
```

удалит из очереди задание с id=599;

```
qdel all
```

удалит из очереди все задания текущего пользователя, выполнившего эту команду.

2.7. Завершение сеанса

Для завершения сеанса удаленного доступа используется команда **exit**. Пользователь может завершить свою сессию путем нажатия в командной строке сочетания клавиш **Ctrl+D**.

Примечание: Завершение сеанса работы с кластером не влияет на исполнение заданий, поставленных в очередь.

3. Основы работы в Linux

3.1. Операции с файлами и каталогами

Файловая система – это средство организации, хранения и именования данных в виде файлов. Файловая система представляет собой древовидную структуру, начинающуюся от корневого каталога. В каждом каталоге могут находиться другие каталоги и файлы. Местоположение файла задается в виде последовательности вложенных каталогов (пути) к файлу. Путь может быть относительным и абсолютным. Абсолютный путь к файлу начинается от корневого каталога, обозначаемого символом **«/»**, относительный путь – от текущего каталога. Например, абсолютный путь к файлу **test.cpp**, находящемуся в каталоге **/home/tester/bin**, будет выглядеть следующим образом:

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

Относительный путь из каталога **/home/tester** будет таким:

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

Примечание: Символом **«..»** (две точки) обозначается предыдущий по отношению к текущему каталог. Если требуется явно указать системе, что она должна использовать файл из текущего каталога, то можно воспользоваться символом **«.»** (одна точка). Например, запись **./test.cpp** эквивалентна указанию абсолютного пути к файлу **test.cpp**, находящемуся в текущем каталоге.

Определение текущего каталога

Для определения текущего каталога командной строки используется команда **pwd**.

Смена каталога

Для перехода в произвольный каталог используется команда **cd**. Например, команда

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

сделает каталог **/home/tester** текущим каталогом командной строки. Команда **cd /** сменит каталог на корневой.

Каждому пользователю соответствует специальный каталог, называемый домашним каталогом. Этот каталог расположен в директории **/home**, и его название совпадает с именем пользователя. Пользователь располагает полными правами доступа к домашнему каталогу и его подкаталогам: может создавать файлы и каталоги, редактировать и удалять их, переименовывать, перемещать, а также запускать программы на выполнение.

Примечание: Для обозначения домашнего каталога используется символ **«~»** («тильда»). Т.е. команду **cd /home/tester/bin** пользователь **tester** может заменить командой

```
cd ~/bin
```

Список объектов в каталоге

Для просмотра списка объектов в каталоге используется команда **ls**. Команда, вызванная без параметров, выведет список файлов и каталогов в текущем каталоге. Команда

```
ls /home/tester
```

выдаст на экран содержимое домашнего каталога пользователя **tester**. Команда **ls** имеет множество параметров. Например, вызов команды с ключом **-a** выведет все объекты каталога, включая

системные, ключ **-l** позволит получить полную информацию об этих объектах.

Создание и удаление каталога

Для создания каталога используется команда

```
mkdir <name>
```

Если указано только имя каталога, он будет создан в текущем каталоге. Можно указать как абсолютный, так и относительный путь к создаваемому каталогу. 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>
```

Эта команда удаляет только пустые каталоги, поэтому перед удалением любого каталога необходимо очистить его содержимое. Для рекурсивного удаления каталога и всего содержимого используется команда:

```
rm –rf <name>
```

Создание файлов

Текстовый файл можно создать в любом текстовом редакторе, указав имя создаваемого файла при запуске редактора. Для создания файла в Midnight Commander нажмите **Shift+F4**. Другой способ создания файла – скопировать уже существующий файл.

Для создания файла нулевой длины используется команда **touch**. Например,

```
touch ~/test
```

создаст пустой файл **test** в домашнем каталоге.

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 в параметре **<source>**. Исходные объекты не изменяются.

Команда **mv** перемещает либо переименовывает файл или каталог. 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 буфера для удаленных файлов (аналог «Корзины» в Windows), поэтому удаленный файл восстановить практически невозможно.

Вывод текстового файла на экран

Команда **less** выводит содержимое текстового файла на экран. Эта команда удобна тем, что позволяет просматривать содержимое файла постранично, 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 поиск назад;
q – выход.

3.2. Перенаправление вывода

Если результат работы программы или команды не помещается полностью на экране, существует несколько способов, позволяющих прочитать вывод полностью. Первый способ – нажать клавишу **Scroll Lock** и перемещать содержимое экрана с помощью клавиш управления курсором, нажать клавишу **Scroll Lock** повторно для выхода из режима просмотра.

Второй способ – перенаправление результатов работы программы (команды) в текстовый файл. Для этого используется специальный символ перенаправления вывода **«>»** («больше»). Так, после выполнения команды

```
cmd1 > file
```

в текущем каталоге появится файл с именем **file**, который будет содержать весь текстовый вывод программы **cmd1**. Если такой файл существовал, он будет перезаписан заново. Если указать два символа перенаправления вывода подряд:

```
cmd1 >> file
```

то весь вывод будет дописан к концу файла.

3.3. Работа с системой документации

Основной способ получения информации в Linux – чтение, так называемых, **man pages** (manual pages – страницы руководства). Большинство объектов Linux, начиная с основных понятий, заканчивая командами операционной системы, описаны в таком руководстве. Для того чтобы посмотреть справку по той или иной команде, наберите в командной строке

```
man <object>
```

Команда

```
man man
```

выведет на экран руководство по использованию команды **man**.

Для перелистывания страниц используются клавиши управления курсором **PgUp**, **PgDn**, **Home**, **End**, для завершения просмотра – **q**.

Кроме того, многие программы могут запускаться в режиме выдачи помощи. Для этого надо запустить их с ключом **--help** (два знака «минус» help) . В результате на экран будет выдана краткая справка по использованию программы. Например, 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:** В большинстве случаев помощь выводится на английском языке. Для получения русскоязычной справки воспользуйтесь интернетом.

3.4. Права доступа к файлам и каталогам

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 к файлам и каталогам определяются для трех категорий: пользователя-владельца, группы-владельца и прочих. 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, следующие 3 символа – права группы, последние 3 символа – права остальных пользователей. В данном примере видно, что владелец может читать, изменять и запускать файл на выполнение, 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
```

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 параметров она может принимать права доступа в краткой записи и имя файла, к которому применяются эти права. Например, команда

```
chmod 700 test
```

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

```
chmod 777 test
```

откроет всем полный доступ к файлу.

3.5. Основные команды Linux

Еще раз перечислим команды, речь о которых шла выше.

mc	запуск файлового менеджера Midnight Commander
man <command>	руководство по использованию **<command>**
pwd	получение имени текущего каталога
cd <dir>	переход в каталог **dir**
ls <dir>	получение списка 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 (каталога)
mv <source> <destination>	перемещение либо переименование файла (каталога)
passwd	смена пароля
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

Дополнительную информацию по этим и другим командам можно получить в справочнике:
http://hpc.icc.ru/documentation/cmnds.pdf

Приложение. Пример сеанса работы на кластере

Приведем пример сеанса работы с кластером для пользователя **tester**. В левом столбце перечисляются команды пользователя (выделены жирным шрифтом) и ответ системы, справа – комментарии. Любая команда записывается в одну строку, даже если на странице она разнесена на две или более строки.

1. Вход на кластер

```
ssh tester@84.237.30.108
Password:

tester@access1 ~ $
```

Начинаем сеанс удаленного доступа. Система запрашивает пароль. Вводим пароль.
Авторизация прошла успешно (пользователь **tester** находится в своем домашнем каталоге на кластере).

2. Создание каталога

```
tester@access1 ~ $ mkdir src
```

Создаем каталог **src**.

3. Просмотр списка файлов

```
tester@access1 ~ $ ls s*

src
```

Проверяем список файлов, начинающихся с символа **s**.
Каталог **src** существует.

4. Переход в каталог

```
tester@access1 ~ $ cd ./src
tester@access1 ~/src $
```

Переходим в каталог **src**.
Изменилось приглашение командной строки.

5. Удаленное копирование файла

```
tester@access1 ~/src $ scp
myname@myhost:/home/myname/progs/test.cpp
/home/tester/src
Password:

100% 2834 0.9KB/s 00:00 скопировано
```

Копируем файл **test.cpp** с удаленного компьютера **myhost**.

Система запрашивает пароль пользователя **myname** для входа в компьютер **myhost**.

6. Компиляция программы

```
tester@access1 ~/src $ ssh sm101

tester@sm101 ~/src $ mpicxx -o test
./test.cpp

tester@sm101 ~/src $ exit
```

Переход на узел компиляции

Компилируем файл **test.cpp** в исполняемый файл **test**.

Выход из узла компиляции

7. Просмотр списка файлов

```
tester@access1 ~/src $ ls test*

test.cpp test
```

Проверяем, какие файлы, начинающиеся с подстроки **test**, имеются в текущем каталоге.
Найдены файлы **test** и **test.cpp**.

8. Просмотр состояния очереди

```
tester@access1 ~/src $ tasks
```

Оцениваем состояние ресурсов и очереди заданий утилитой **tasks**. См. пример в разделе 2.6.1.

9. Получение информации о занятости ресурсов

```
tester@access1 ~/src $ qfree
```

Проверяем наличие свободных ресурсов.

```
================== Segment I ====================
60 nodes with 2x16 AMD Opteron 6276 Interlagos 2.3
GHz; 64 GB RAM per node
Nodes [Total: 59   Free: 48   Busy: 1   Down: 10 ]
Cores [Total: 1888 Free: 1536 Busy: 32  Down: 320]
```

В сегменте I: узлов всего – 60, из них 48 – свободно, 1 – занят, 10 – недоступно; процессорных ядер всего – 1888, из них свободно – 1536, занято – 32.

10. Создание файла задания для Torque

```
tester@access1 ~/src $ mcedit task.pbs
```

Открываем файл с именем **task.pbs** для редактирования утилитой **mcedit**.
Редактируем атрибуты задания.
Закрываем редактор клавишей **F10.**

(см. раздел 2.6)
F10

11. Постановка задания в очередь

```
tester@access1 ~/src $ qsub.amd task.pbs
797.master
```

Ставим задание в очередь сегмента I.
Номер задания в очереди.

12. Ожидание завершения задания

```
tester@access1 ~/src $ tasks
```

Для оценки текущего состояния задания в очереди выполняем команду **tasks** .

13. Просмотр списка файлов

```
tester@access1 ~/src $ ls -l
```

Выводим список файлов текущей директории **src**.
В текущей директории появились файлы **TestPBS.o797** и **TestPBS.e797**, содержащие данные потоков **stdout** и **stderr** запущенного задания.

```
total 128
-rw-r--r-- 1 tester users 210  Oct 7 17:10 task.pbs
-rw-r--r-- 1 tester users 8432 Oct 7 17:10 test
-rw-r--r-- 1 tester users 323  Oct 7 17:10 test.cpp
-rw-r--r-- 1 tester users 50   Oct 7 17:11 TestPBS.e797
-rw-r--r-- 1 tester users 733  Oct 7 17:11 TestPBS.o797
```

14. Завершение работы с кластером

```
tester@access1 ~/src $ exit
```

Завершаем сеанс работы.