Программно-аппаратный вычислительный комплекс «СКИФ-Аврора ЮУрГУ»

Суперкомпьютер «Торнадо ЮУрГУ»

Руководство пользователя

88746109.007.ЮУрГУ.И3.2

МОСКВА
2012

АННОТАЦИЯ

Данный документ является частью технической, эксплуатационной и сервисной документации для Программно-аппаратный вычислительный комплекс «СКИФ-Аврора ЮУрГУ» высокопроизводительной вычислительной системы, разрабатываемой на основании договора № 0369100017612000103_45278, заключенным между Федеральным государственным бюджетным образовательным учреждением высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет) и ЗАО «РСК Технологии».

В документе приведено общее описание высокопроизводительного вычислительного комплекса «СКИФ-Аврора», созданного в Федеральном государственном бюджетном образовательном учреждении высшего профессионального образования «Южно-Уральский государственный университет» (ЮУрГУ) в г. Челябинск.

СОДЕРЖАНИЕ

1. Введение

Данный документ является частью технической, эксплуатационной и сервис-ной документации для Программно-аппаратный вычислительный комплекс «СКИФ-Аврора ЮУрГУ» высокопроизводительной вычислительной системы, разрабатываемой на основании договора № 0369100017612000103_45278, заключенным между Федеральным государственным бюджетным образовательным учреждением высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет) и ЗАО «РСК Технологии».

Высокопроизводительная вычислительная система для решения задач высокопроизводительных вычислений (далее – вычислительная система) – вычислительная система, реализованная на базе инновационной системы с жидкостным охлаждением РСК «Торнадо». Основной функцией данной вычислительной системы является решение задач с высокой вычислительной нагрузкой.

Кроме этого, вычислительная система обладает такими свойствами, как:

- высокой энергоэффективностью решения за счет отвода тепла с помощью жидкостной системы охлаждения;
- высокой вычислительной плотностью, необходимой для реализации сверхбольших суперкомпьютеров с высокоскоростными вычислительными сетями;
- высокой отказоустойчивостью за счет отсутствия движущихся частей, таких как вентиляторы системы охлаждения и жёсткие диски.

Настоящий документ является достаточным для ознакомления конечным пользователем вычислительной системы для решения задач с высокой вычислительной нагрузкой.

2. Описание системы

Вычислительный кластер представляет собой 480 вычислительных узлов и два сервера (управления и доступа пользователей), объединенных между собой с помощью транспортной сети Infiniband и двух Ethernet сетей (мониторинга и управления заданиями).

Вычислительные узлы имеют сквозную нумерацию вида «nodeXXX» (где X, число от 001 до 480). Каждый вычислительный узел имеет в своем составе:

- Сопроцессор Intel Xeon Phi, имеющий имя вида «nodeXXX-mic0». Внутри вычислительного узла возможна адресация по «mic0».
- Сетевой интерфейс Infiniband (имя Ib0 в пределах хоста или nodeXXX-ib0 в пределах кластера)
- Сетевой интерфейс Ethernet (eth0)

Таблица 1 – Описание системы

Планировщик ресурсов SLURM	
Очередь пользовательских заданий	Имя – **work** (просмотр состояния «*sinfo –p work»)*
Диапазон адресов узлов	node001-node480
Установленное по умолчанию время выполнения задачи (пользователь может установить самостоятельно)	3 дня
Максимально доступное время выполнения задачи	14 дней
Network	
Единая точка доступа пользователей - сервер «login»	Адрес - **login.tornado.hpc.susu.ac.ru**
Сетевые интерфейсы	
Интерфейс сети управления заданиями - eth0	10.4.4.0/22
Интерфейс транспортной сети - Ib0	10.4.8.0/22
Intel Xeon Phi	
Адресация	nodeXXX-mic0 внутри кластера, mic0 адресация внутри ВУ
Доступные модули пакета Environmental Modules	
parallel/mpi.intel/4.1.0.024	Intel MPI
compilers/composer_xe/2013_sp1	Компиляторы Intel Composer XE
compilers/cplusplus/gnu/4.4.6	Компилятор GNU C
compilers/cplusplus/intel/13.0.1	Компилятор Intel C++
compilers/fortran/gnu/4.4.6	Компилятор GNU Fortran
launcher/mic	Набор скриптов для запуска MPI-задач с использованием сопроцессора Intel Xeon Phi
dot	Добавляет текущую директорию к переменной окружения PATH
launcher/slurm(default)	Включает поддержку запуска MPI-программ для утилиты “srun”
launcher/intel	Включает SLURM-механизм взаимодействия между вычислительными узлами для утилиты “mpiexec.hydra”

3. Назначение и условия применения

Основные технические характеристики вычислительной системы представлены в таблице 3.1.

Табл. 3.1. Технические характеристики вычислительной системы

Параметр	Значение для одного узла	Значение для Вычислителя в целом
Теоретическая пиковая производительность	1,447 Тфлопс	694,56 Тфлопс
Теоретическая пиковая производительность процессоров Intel Xeon X5680	0,371 Тфлопс	178,08 Тфлопс
Теоретическая пиковая производительность сопроцессоров Intel Xeon Phi	1,076 Тфлопс	206,592 Тфлопс
Количество сопроцессоров Intel Xeon Phi 7110X	1 (node[001-192])	192 (node[001-192])
Объем памяти сопроцессора Intel Xeon Phi	8 Гб	1 536 Гб
Тип используемых процессорных чипов	Intel Xeon X5680 6 ядер, тактовая частота ядра 3,33 ГГц, QPI 8,0 ГТ/с, размер кэша 12 МБ	
Характеристики коммуникационной и транспортной сети Вычислителя	Infiniband QDR 40 Гбит/с с полной бисекционной пропускной способностью.	
Сеть Gigabit Ethernet управления заданиями и мониторинга	Обеспечение доступа ко всем вычислительным узлам и управляющему серверу со скоростью 1 Гбит/с	
Количество процессорных чипов	2	960
Количество процессорных ядер	12	5760
Объем памяти	24 Гб (4 ГБ на ядро) на node[192-480], 48 Гб (8 ГБ на ядро) на node[001-192]	16128 ГБ
Количество узлов	-	480
Тип дисков	80 ГБ SATA SSD	
Количество дисков	1	480

Функциональная схема высокопроизводительной вычислительной системы представлена на рисунке 3–1.

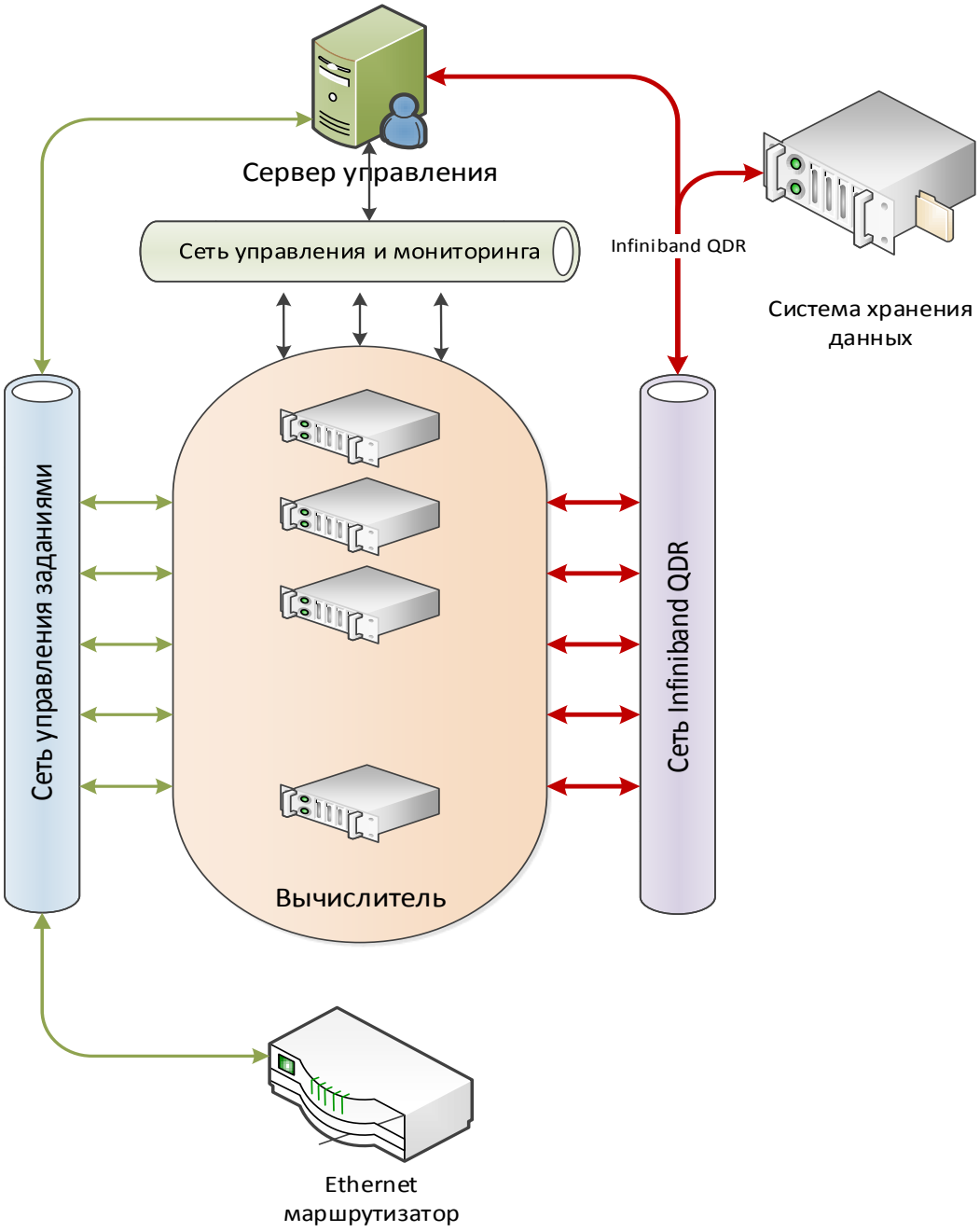

Рис. 3–1. Функциональная схема вычислительной системы

В состав вычислительной системы входят Вычислитель, сервер управления Вычислителем, сервер доступа пользователей и система хранения данных. Для связи компонентов в единую систему и с сетевой инфраструктурой ЮУрГУ используется несколько сетей, каждая из которых служит для выполнения определённых функций.

Вычислитель представляет собой кластерную систему из множества узлов, объединённых между собой высокопроизводительной коммуникационной сетью, реализованной по технологии Infiniband QDR. Коммуникационная сеть поддерживает реализацию основных примитивов библиотеки MPI и построена по топологии полной бисекционной пропускной способности.

Все узлы Вычислителя, сервер управления и сервер доступа пользователей, подключены к двум сетям Ethernet, одна из которых используется для управления и мониторинга вычислительных узлов, а вторая для управления заданиями.

192-а узла Вычислителя оснащены сопроцессором Intel Xeon Phi 7110X. Объем и тип оперативной памяти каждого сопроцессора 8 ГБ GDDR5. Данные сопроцессоры предназначены для ускорения выполнения команд x86 с векторными расширениями.

4. Подготовка к работе

4.1 Требования к квалификации пользователя

Квалификация пользователя, допускаемого к эксплуатации вычислительной системы, должна обеспечивать эффективное функционирование системы во всех заданных режимах.

Пользователь должен пройти общую и специальную подготовку по работе со средствами вычислительной системы и средствами вычислительной техники.

Общая подготовка должна включать в себя получение навыков работы с программным обеспечением в объеме навыков пользователей вычислительной системы.

Специальная подготовка должна включать в себя получение навыков работы с системным и прикладным обеспечением вычислительной системы в объеме навыков ее использования.

4.2 Вход в систему и аутентификация

4.2.1 Общие принципы

Политика работы вычислительной системы подразумевает интерактивный вход пользователя на консоли вычислительных узлов, который возможен только из планировщика задач, основной интерфейс к которому представлен на сервере доступа пользователей. В каждый конкретный момент времени планировщик выделяет вычислительный узел в единоличное использование пользователю, запросившему такой доступ.

4.2.2 Получение реквизитов для удалённого доступа

Необходимые параметры и настройки для обеспечения доступа предоставляет Администратор вычислительной системы.

Для доступа к системе необходима учётная запись и пароль.

4.2.3 Удалённый терминальный доступ

Удалённый терминальный доступ пользователя обеспечивается средствами SSH.

SSH (англ. Secure SHell — «безопасная оболочка») — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

В среде Linux используется приложение ssh, для операционных систем семейства Microsoft – PuTTY. Приложение доступно для свободного скачивания по адресу http://www.chiark.greenend.org.uk/~sgtatham/putty/.

Получение удаленного терминального доступа к системе описано в 5.1.1

Для удалённой работы с файлами используется подсистема SSH — SFTP, описанная в 5.2

5. Описание операций

5.1 Терминальный доступ

Каталог пользователя доступен на каждом узле вычислительной системы в директории вида /home/user, где user — имя пользователя.

В примере представлен вывод клиента SSH для командной строки.

5.1.1 Получение удалённого терминального доступа к системе по паролю. Пользователь с помощью утилиты «ssh» получает доступ к системе и на запрос пароля вводит свой пароль к серверу доступа:

```
% ssh user@login.tornado.hpc.susu.ac.ru
Password:
Last login: Sat Jan 19 10:46:12 2013 from 178.209.98.254
[user@login ~]#
```

5.1.2 Настройка ssh-ключей

При первом входе пользователя на кластер автоматически генерируется ssh-ключ, предназначенный для доступа на вычислительные узлы кластера.

В случае, если публичный ключ пользователя ранее существовал, он не будет автоматически добавлен в файл ~/.ssh/authorized_keys, и пользователю необходимо проверить его наличие. Для этого нужно убедиться, что в файле authorized_keys присутствует запись для данного пользователя. Пример записи:

```
user@login:~$ cat ~/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDMuB9A5ZkHQlxdU6U0IN0ChAR0On/UxuaVclW4tLjHT+wY
WkCh6yCGtVT6z4xbJBA79r6tR1RBsSxB7IvgE+BuFkbuEhD8nU2e13JnJF7A0w6DFY+VPt6fhxe3
+lZchfj7OFZglKQrDIzZYgDRUV+K0g+/lT1br2Os8XyRZYPDxYOmrERfbRB+593cHygBqsrN4VJ+
E254LPFh5ZiDIqQ9j2Xlurjewjgljqop XTG0MwN2Qd9M/XUjC11lTRoOBgqv2JifDGgKCUJdNUnu
WVt5HOl4UAg9BZDS0V/wangwBY/OslnF2eYO/4U5qGfcu1mpmGcMOGT0AWHWM2SovoNx      us-
er@home
```

5.1.3 Получение имени текущей рабочей директории

```
user@login:~$ pwd
/home/user
```

5.1.4 Просмотр содержимого директории

```
user@login:~$ ls
schrodinger  test2
```

5.1.5 Просмотр атрибутов избранного файла в директории

```
user@login:~$ ls -lad act_test1.tgz
-rw-r--r-- 1 user users 3305845 Sep 22  2011 act_test1.tgz
Примеры переходов по дереву директорий:
user@login:~$ cd opt/
user@login:~/opt$ pwd
/home/user/opt
```

```
user@login:~/opt$ ls -la
total 32
drwxr-xr-x  8 user users 4096 Oct 17  2011 .
drwxr-xr-x 32 user users 4096 Aug  3 14:50 ..
drwxr-xr-x  2 user users 4096 Oct 19  2011 bin
drwxr-xr-x  6 user users 4096 Oct 17  2011 fftw
drwxr-xr-x  2 user users 4096 Oct 19  2011 include
drwxr-xr-x  3 user users 4096 Oct 19  2011 lib
drwxr-xr-x  4 user users 4096 Oct 17  2011 share
user@login:~/opt$ cd ..
user@login:~$ cd /
user@login:/$ pwd
/
user@login:/$ cd
user@login:~$ pwd
/home/user
user@login:~$
```

5.2 Использование SFTP

SFTP (англ. SSH File Transfer Protocol) — протокол прикладного уровня, предназначенный для копирования и выполнения других операций с файлами поверх надёжного и безопасного соединения. Протокол разработан группой IETF как расширение к SSH-2, однако SFTP допускает реализацию и с использованием иных протоколов сеансового уровня.

5.2.1 Получение удалённого доступа к системе

В примере представлен вывод клиента SFTP для командной строки:

```
% sftp login.tornado.hpc.susu.ac.ru
Connected to login.tornado.hpc.susu.ac.ru.
```

5.2.2 Получение имени текущей рабочей директории

```
sftp> pwd
Remote working directory: /home/user
```

5.2.3 Получение списка файлов в текущей директории

```
sftp> ls
schrodinger test2
sftp>
```

5.2.4 Загрузка файла

```
sftp> put test
Uploading test to /home/user/test
test
100%    0     0.0KB/s   00:00
```

5.2.5 Получение файла

```
sftp> get test
Fetching /home/user/test to test
```

5.2.6 Выход из клиента

```
sftp> bye
```

5.3 Конфигурация среды пользователя

5.3.1 Использование Environmental Modules

Для управления множественными версиями различных прикладных программных пакетов и библиотек на вычислительной системе установлен пакет Environmental Modules.

Он позволяет гибко настраивать переменные окружения и пакетных задач для использования тех или иных версий ПО и отслеживания их зависимостей. Также использование Environmental Modules позволяет гибко управлять разными версиями приложения.

Пакет состоит из модулей, описанных в modulefiles, которые доступны по адресу: /opt/basis/modules/Modules.

Так же пользователь может создавать свой набор пользовательских файлов в домашней директории.

Каждый модуль содержит информацию, необходимую для настройки окружения под конкретное приложение (путем настройки таких переменных, как PATH, MANPATH, INCLUDE, LD_LIBRARY_PATH и т.д.).

Модули могут быть динамически (или автоматически) подружены и выгружены, в свободном режиме. Поддерживаются все популярные shell, включая bash, ksh, zsh, sh, csh, tcsh, в том числе такие интерпретаторы, как perl.

По умолчанию при входе в систему для пользователя автоматически загружаются следующие модули:

- Модуль Intel Composer XE 2013 (compilers/composer_xe/2013_sp1)
- Модуль Intel MPI (parallel/mpi.intel/4.1.0.024)

Для просмотра подгруженных модулей выполните следующую команду:

```
$ module list
Currently Loaded Modulefiles:
  1) compilers/composer_xe/2013_sp1    2) parallel/mpi.intel/4.1.0.024
```

5.3.2 Список доступных модулей

Для просмотра списка доступных модулей выполните команду:

```
bash-4.1$ module avail
```

5.3.3 Выбор компилятора

Для того чтобы подгрузить компилятор C++ (по умолчанию - Intel) выполните команду:

```
$ module load compilers/cplusplus
```

```
$ module list
Currently Loaded Modulefiles:
 1)  modules                                                    2)  compil-
ers/cplusplus/intel/13.0.1
$ $CC --version
icc (ICC) 13.0.1 20121010
Copyright (C) 1985-2012 Intel Corporation.  All rights reserved.
```

5.3.4 Замена компилятора

Для того чтобы заменить Intel C/C++ на GNU выполните команду:

```
$     module     switch     compilers/cplusplus/intel/13.0.1     compil-
ers/cplusplus/gnu/4.4.6
```

5.3.5 Использование библиотек

Для выбора библиотеки Intel MPI выполните команду:

```
$ module load parallel/mpi.intel
$ module list
Currently Loaded Modulefiles:
1) compilers/cplusplus/intel/13.0.1   2) parallel/mpi.intel/4.1.0.024
$ mpiexec --version
Intel(R) MPI Library for Linux, 64-bit applications, Version 4.1  Build
20120831
Copyright (C) 2003-2012 Intel Corporation.  All rights reserved.
```

5.4 Компиляция приложений

5.4.1 Компиляция MPI-приложений с использованием Intel Xeon Phi

MPI-приложения использующие вычислительные мощности сопроцессора Intel Xeon Phi могут быть скомпилированы для запуска в 3-х различных режимах. Режимы запуска MPI-задач на Intel Xeon Phi описаны в 6.2.1

5.4.1.1 Режим разгрузки (Offload Mode)

Компиляция MPI-программ, оптимизированных для использования Intel Xeon Phi в offload-mode, ничем не отличается от компиляции обычных MPI-программ. В общем случае команда имеет вид

```
mpicc <input_file> -o <output_file>
```

Пример компиляции MPI-программы сортировки описанной в 6.2.3

```
mpicc ./tbo_sort.c -o ./tbo_sort.x
```

5.4.1.2 «Родной» режим сопроцессора (Native Mode)

Для компиляции MPI-программы запускаемой исключительно на сопроцессоре утилите “*mpicc”* необходимо указать опцию *–mmic*. Пример компиляции:

```
mpicc -mmic ./tbo_sort.c -o ./tbo_sort.mic
```

5.4.1.3 Симметричный режим (Symmetrical Mode)

Для запуска MPI-программы с использованием CPU и сопроцессора в симметричном режиме необходимо скомпилировать два бинарных файла, один как обычную MPI-программу, а второй как MPI-программу запускаемую в «native mode». Пример будет выглядеть так:

```
mpicc ./tbo_sort.c -o ./tbo_sort
mpicc -mmic ./tbo_sort.c -o ./tbo_sort.mic
```

Запуск MPI-задач на Intel Xeon Phi описан в п. 6.2

5.4.2 Компиляция с использования Intel MKL (Math Kernel Library)

Для использования Intel MKL достаточно передать компилятору ключ –mkl. Например

```
icc –mkl <input_file> -o <output_file>
mpicc –mkl <input_file> -o <output_file>
```

5.5 Планировщик задач (SLURM)

5.5.1 Состояние планировщика

Для получения информации о состоянии и планировщика выполните команду “sinfo”:

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
work*        up 5-00:00:00     20  drain node[006,031-042,044-050]
work*         up 5-00:00:00      92   alloc node[001-005,007-030,043,051-
112]
debug        up 10-00:00:0     20  drain node[006,031-042,044-050]
debug         up 10-00:00:0      92   alloc node[001-005,007-030,043,051-
112]
```

5.5.2 Запуск MPI-задач

MPI-задачи на кластере могут быть запущены в пакетном или интерактивном режимах.

В пакетном режиме пользователь создает скрипт для запуска задачи и использует утилиту “sbatch”. Задача отправляется в очередь и будет выполнена как только будут доступны запрошенные ресурсы. Вывод результата выполнения задачи будет записан в файл slurm-<номер задачи в очереди>.out в директории, откуда осуществлялся запуск задачи.

В интерактивном режиме пользователь либо использует для запуска задачи утилиту “srun”, либо самостоятельно выделяет необходимое количество ВУ и запускает за-

дачу с помощью утилиты mpiexec.hydra или “srun”. Вывод результата выполнения задачи будет произведен в консоль.

5.5.2.1 Пакетный запуск MPI-задачи (sbatch)

SLURM поддерживается Intel® MPI Library начиная с версии 4.0 Update 3 напрямую через Hydra PM.

Для запуска пакетной задачи используется утилита *sbatch*.

Основные ключи утилиты *sbatch*:

```
-N, --nodes          указывает количество необходимых узлов
-n, --ntasks         общее количество запущенных процессов
--ntasks-per-node    задает количество процессов запускаемых на каждом вычис-
                     лительном узле
-t, --time           время доступности выделенных ВУ  (в минутах)
```

В качестве параметра ей передается исполняемый скрипт, который будет запущен на первом из выделенных вычислительных узлов. Должен быть загружен модуль «launcher/slurm». Внутри скрипта выполняется команда *mpiexec.hydra,* которой в качестве параметров передается кол-во выделенных узлов и исполняемая программа.

Данный пример запускает на выполнение MPI-программу `hello_mpi.x` на 4-х вычислительных узлах с использованием «*mpiexec.hydra»*:

```
bash-4.1$ cat hello_hydra.sh
#!/bin/sh
mpiexec.hydra -n $SLURM_NNODES ./hello_mpi.x
bash-4.1$ module load launcher/slurm
bash-4.1$ sbatch -N 4 ./hello_hydra.sh
Submitted batch job 133
```

Скрипт для пакетного запуска 4-х экземпляров MPI-программы `hello_mpi.x` на каждом из 3-х вычислительных узлах с использованием «*mpiexec.hydra»* выглядит так:

```
bash-4.1$ cat hello_hydra.sh
#!/bin/sh
mpiexec.hydra -perhost 4 -n 12 ./hello_mpi.x
bash-4.1$ module load launcher/slurm
bash-4.1$ sbatch -N 3 ./hello_hydra.sh
Submitted batch job 135
```

Скрипт для пакетного запуска 2-х экземпляров MPI-программы `hello_mpi.x` на каждом из 4-х вычислительных узлах с использованием «*srun»* выглядит так:

```
bash-4.1$ cat hello_hydra.sh
#!/bin/sh
srun -N 4 --ntasks-per-node 2 -n 8 ./hello_mpi.x
bash-4.1$ module load launcher/slurm
bash-4.1$ sbatch -N 3 ./hello_hydra.sh
Submitted batch job 135
```

Выводы всех вышеописанных скриптов будет записан в файл slurm-<номер задачи в очереди>.out в директории, откуда осуществлялся запуск задачи.

5.5.2.2 Интерактивный запуск MPI-задачи (srun)

Основные ключи утилиты srun:

```
-N, --nodes          указывает количество необходимых узлов
--ntasks-per-node    задает количество процессов запускаемых на каждом вычис-
                     лительном узле
-n, --ntasks         общее количество запущенных процессов
-t, --time           время доступности выделенных ВУ  (в минутах)
```

Допустим, нам необходимо запустить по X экземпляров MPI-задачи на Y хостах. Для этого выполните следующие действия:

- Загрузите модуль launcher/slurm (если не загружен) командой:

```
module load launcher/slurm
```

- Используйте команду srun для запуска MPI-задачи:

```
srun -N Y --ntasks-per-node=X -n X*Y ./hello_mpi.x
```

Далее задача будет запущена, а результат выполнения будет выведен на экран:

```
Hello, world, I am 6 of 6
Hello, world, I am 5 of 6
Hello, world, I am 4 of 6
Hello, world, I am 3 of 6
Hello, world, I am 2 of 6
Hello, world, I am 1 of 6
```

5.5.2.3 Интерактивный запуск MPI-задачи (mpiexec.hydra)

Основные ключи утилиты “mpiexec.hydra”:

```
-perhost <n>         Запустить n процессов на каждом узле
-n                   Общее количество процессов
```

Для получения информации о всех доступных опциях утилиты “mpiexec.hydra” используйте команду:

```
mpiexec.hydra --help
```

Основные ключи утилиты “salloc”:

```
-N, --nodes          указывает количество необходимых узлов
-t, --time           время доступности выделенных ВУ (в минутах)
```

Для получения информации о всех доступных опциях утилиты “salloc” используйте команду:

```
man salloc
```

Для запуска MPI-приложения *hello_mpi.x* на X вычислительных узлах в интерактивном режиме выполните следующие действия:

- Выделите X вычислительных узлов из кластера:

```
salloc -N X
```

При наличии ресурсов будет выведено сообщение об успешном выделении вычислительных узлов:

```
salloc: Pending job allocation 301
salloc: job 301 queued and waiting for resources
salloc: job 301 has been allocated resources
```

```
salloc: Granted job allocation 301
```

- Загрузите модуль launcher/slurm (если не загружен) командой:

```
module load launcher/slurm
```

- Используйте команду “*mpiexec.hydra*” для запуска MPI-задачи:

```
mpiexec.hydra –perhost 1 ./hello_mpi.x
```

Далее задача будет запущена, а результат выполнения будет выведен на экран.

```
Hello, world, I am 4 of 4
Hello, world, I am 3 of 4
Hello, world, I am 2 of 4
Hello, world, I am 1 of 4
```

Возьмем другой пример. Допустим, нам необходимо запустить по X экземпляров MPI-задачи на Y хостах . Для этого выполните следующие действия

- Выделите Y вычислительных узлов из кластера:

```
salloc –N Y
```

При наличии ресурсов будет выведено сообщение об успешном выделении вычислительных узлов:

```
salloc: Pending job allocation 302
salloc: job 302 queued and waiting for resources
salloc: job 302 has been allocated resources
salloc: Granted job allocation 302
```

- Загрузите модуль launcher/slurm (если не загружен) командой:

```
module load launcher/slurm
```

- Используйте команду “*mpiexec.hydra*” для запуска MPI-задачи:

```
mpiexec.hydra -perhost X -n X*Y ./hello_mpi_new.x
```

где ключ `-perhost` задает количество процессов запускаемых на каждом вычислительном узле, а ключ `-n` общее количество запущенных процессов.

Далее задача будет запущена, а результат выполнения будет выведен на экран.

```
Hello, world, I am 6 of 6
Hello, world, I am 5 of 6
Hello, world, I am 4 of 6
Hello, world, I am 3 of 6
Hello, world, I am 2 of 6
Hello, world, I am 1 of 6
```

5.5.3 Освобождение выделенных узлов

Для освобождения узлов, выделенных командой “*salloc”,* пользователь может воспользоваться комбинацией клавиш “Ctrl+D” или командой *exit*:

5.5.4 Просмотр состояния задач

Для получения списка активных задач используйте команду *squeue*:

```
user@login:~$ squeue
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
   3799      work   neg.sh  user2  PD       0:00     32 (Resources)
   3726      work   nhhid0   user1   R 3-10:01:30      4 node[028-030,070]
   3727      work   nhhid1   user1   R 3-10:01:30      4 node[062-064,074]
   3731           work      102nh0      user1      R  3-00:19:48           4
node[021,043,065,094]
   3732      work   102nh1   user1   R 2-22:29:27      4 node[066-069]
   3733      work   102nh2   user1   R 2-21:04:34      4 node[022-025]
   3734      work   102nh3   user1   R 2-20:16:37      4 node[001-004]
   3735      work   102nh4   user1   R 2-20:16:37      4 node[095-098]
   3780       work 4lip2_ob    user1    R 2-00:08:42      42 node[007-020,026-
027,051-061,079-093]
   3783        work   chx_325    user1    R 1-23:15:49       18 node[005,071-
073,099-112]
   3807      work     bash   user   R       1:23      2 node[075-076]
```

5.5.5 Удаление задачи

Для удаления запущенной задачи необходимо знать её идентификатор. Для удаления задачи с известным ID выполните команду:

```
user@login:~$ scancel 3807
salloc: Job allocation 3807 has been revoked.
```

5.5.6 Переменные окружения, используемые планировщиком SLURM

При выделении ресурсов или запуске задач планировщик автоматически прописывает в переменные окружения актуальную служебную информацию. Ниже приведен список этих переменных с описанием:

- **$SLURM_JOB_CPUS_PER_NODE** – количество процессорных ядер, которое может быть использовано задачей на каждом выделенном вычислительном узле (по умолчанию, 32)
- **$SLURM_JOBID** – идентификатор текущей аллокации ресурсов
- **$SLURM_JOB_ID** – аналогично $SLURM_JOBID
- **$SLURM_JOB_NODELIST** – список выделенных вычислительных узлов
- **$SLURM_JOB_NUM_NODES** – количество выделенных вычислительных узлов
- **$SLURM_NNODES** – аналогично $SLURM_JOB_NUM_NODES
- **$SLURM_NODE_ALIASES** – псевдонимы выделенных вычислительных узлов (не используется в данной инсталляции)
- **$SLURM_NODELIST** – аналогично $SLURM_JOB_NODELIST
- **$SLURM_SUBMIT_DIR** – путь до директории, в которой находился текущий пользователь в момент выделения ресурсов
- **$SLURM_TASKS_PER_NODE** – количество процессов, которые могут быть одновременно запущены на одном вычислительном узле (по умолчанию равно количеству ядер, в данной конфигурации - 24)

6. Использование Intel Xeon Phi

6.1 Выделение узла с Intel Xeon Phi

6.1.1 Выделение сопроцессоров Intel Xeon Phi (salloc)

Для выделения 6-ти сопроцессоров Intel Xeon Phi выполните в консоли следующую команду:

```
[user@login ~]$ salloc –N 3 –-gres=mic:1
salloc: Granted job allocation 385
```

После выполнения команды имена выделенных узлов присваиваются переменной окружения SLURM_NODELIST.

Например:

```
[user@login ~]$ echo $SLURM_NODELIST
node[056-057]
```

В нашем случае, это вычислительные узлы с 55-го по 57-ой включительно.

6.1.2 Интерактивный доступ к Intel Xeon Phi

Для доступа к сопроцессорам сначала выделите необходимое количество узлов как описано в п. 6.1.1.

Интерактивный доступ к сопроцессорам может быть получен либо непосредственно с выделенных вычислительных узлов, либо с сервера доступа пользователей. Сопроцессоры именуются добавлением к имени узла, содержащего данный сопроцессор, слова “mic0”. Например, доступ к сопроцессору “mic0” с выделенного вычислительного узла “node056”:

```
ssh mic0
```

Доступ к сопроцессору “mic0”, расположенному на выделенном вычислительном узле “node059” с сервера доступа пользователей:

```
ssh node059-mic0
```

6.1.2.1 Интерактивный доступ с сервера доступа пользователей:

Выполните следующие команды:

```
[user@login ~]$ salloc -N 1 --gres=mic:1
salloc: Granted job allocation 387
[user@login ~]$ echo $SLURM_NODELIST
Node055
```

Для доступа к сопроцессору на вычислительном узле “node055” выполните:

```
[user@login ~]$ ssh node055-mic0
Warning: Permanently added 'node055-mic0' (RSA) to the list of known hosts.
~ $
```

6.1.2.2 Интерактивный доступ с выделенного узла:

Выполните следующие команды:

```
[user@login ~]$ salloc -N 1 --gres=mic:1
salloc: Granted job allocation 389
[user@login ~]$ echo $SLURM_NODELIST
Node056
[user@login ~]$ ssh node056
Warning: Permanently added 'node056' (RSA) to the list of known hosts.
Last login: Mon Jan 28 17:35:41 2013 from 10.65.0.1
[user@node056 ~]$
```

При доступе с выделенного вычислительного узла обращаться к сопроцессору можно непосредственно по его имени “mic0”. К примеру, для получения доступа к сопроцессору введите:

```
[user@node056 ~]$ ssh mic0
Warning: Permanently added 'mic0' (RSA) to the list of known hosts.
~ $
```

6.2 Запуск задач на Intel Xeon Phi

6.2.1 Режимы запуска задач на Intel Xeon Phi

Задачи на Intel Xeon Phi могут быть запущены в следующих режимах:

- Симметричный режим (symmetrical mode). Программа компилируется в два бинарных файла для запуска на CPU и сопроцессоре ВУ одновременно. Для коммуникации между экземплярами программы используется Intel MPI.
- Режим разгрузки (offload mode). Части программного кода используют оптимизацию под Intel Xeon Phi. Программа запускается на вычислительном узле и задействует мощности сопроцессора для ускорения выполнения оптимизированных участков кода.
- “Родной” режим сопроцессора (native mode). Программа пишется и компилируется для запуска непосредственно на сопроцессоре.

6.2.2 Запуск симметричной MPI-задачи

Для данной ситуации понадобятся два отдельных бинарных файла одной программы собранные для запуска на CPU и сопроцессоре ВУ. Процесс компиляции и сборки программ для запуска в данном режиме описан в п.5.4.1.3.

Схема именования бинарных файлов:

1. Имя файла для запуска на CPU – “<имя>” (например, “compute_phis”)
2. Имя файла для запуска на сопроцессоре – “<имя>.mic” (в нашем случае, “compute_phis.mic”)

Должен быть подгружен модуль “launcher/mic”.

Для запуска задачи мы используем утилиту “sbatch” и скрипт “symmetric_run.sh”. Вывод результата выполнения задачи будет записан в файл " slurm-<номер задачи в очереди>.out ” в директории откуда был произведен запуск скрипта.

Пример вывода скрипта при запуске его на одном вычислительном узле:

```
[user@login]$ sbatch -N 1 symmetric_run.sh ./hello_sym
Submitted batch job 3159
[user@login]$ cat slurm-3159.out
Master rank       0 ( 12 threads) of       3 with PID 119270 is running on
node051
 Slave rank       1 ( 12 threads) of       3 with PID 119271 is running on
node051
 Slave rank       2 (240 threads) of       3 with PID 106601 is running on
node051-mic0
```

6.2.3 Запуск MPI-задачи в режиме разгрузки

Процесс компиляции и сборки программ для запуска в данном режиме описан в п.5.4.1.1.

Рассмотрим MPI-программу осуществляющую простую сортировку чет/нечет ряда чисел от 1 до 20 с помощью сопроцессора Intel Xeon Phi.

Выделяем два вычислительных узла:

```
[user@login]$ salloc -N 2 --gres=mic:1
salloc: Granted job allocation 617
```

Запускаем по одному процессу на двух вычислительных узлах:

```
[user@login]$ mpiexec.hydra -perhost 1 ./tbo_sort.x
  Checking for Intel(R) MIC Architecture (Target CPU) devices on host
node057...
    Number of Target devices installed: 2
  Checking for Intel(R) MIC Architecture (Target CPU) devices on host
node056...
    Number of Target devices installed: 2
  Unsorted original values...first twenty (20) values:
    Evens and Odds:
               1    2    3    4    5    6    7    8    9   10
              11   12   13   14   15   16   17   18   19   20
  Sorted results...first ten (10) values each:
    Evens:     2    4    6    8   10   12   14   16   18   20
    Odds :     1    3    5    7    9   11   13   15   17   19
   Primes:     2    3    5    7   11   13   17   19   23
Hello from Intel Xeon Phi, I am 2 of 2
  Unsorted original values...first twenty (20) values:
    Evens and Odds:
               1    2    3    4    5    6    7    8    9   10
              11   12   13   14   15   16   17   18   19   20
  Sorted results...first ten (10) values each:
    Evens:     2    4    6    8   10   12   14   16   18   20
    Odds :     1    3    5    7    9   11   13   15   17   19
   Primes:     2    3    5    7   11   13   17   19   23
Hello from Intel Xeon Phi, I am 1 of 2
```

6.2.4 Запуск MPI-задачи в «родном» режиме сопроцессора

Вам понадобятся бинарный файла собранный для запуска на сопроцессоре Intel Xeon Phi. Процесс компиляции и сборки программ для запуска в данном режиме описан в п. 5.4.1.2.

Имя бинарного файла должно заканчиваться на “.mic”. Например, “compute_power.mic”

Должен быть подгружен модуль “launcher/mic”.

Для запуска задачи мы используем утилиту “sbatch” и скрипт “native_run.sh”. Вывод результата выполнения задачи будет записан в файл " slurm-<номер задачи в очереди>.out ” в директории откуда был произведен запуск скрипта.

Пример вывода скрипта при запуске его на одном вычислительном узле:

```
[user@login]$ sbatch -N 1 native_run.sh ./hello_sym.mic
Submitted batch job 3169
[user@login]$ cat slurm-3169.out
Master rank       0 (240 threads) of       1 with PID 108421 is running on
node051-mic0
```

6.3 Дополнительные сведения

Для успешного использования сопроцессора рекомендуется ознакомится со следующими разделами на сайте (http://software.intel.com/mic-developer) производителя устройства и прилагаемыми документами:

- Архитектура устройства:

  (http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner)

- Обзор программирования для процессоров Intel Xeon и Intel Xeon Phi:

  (http://software.intel.com/sites/default/files/article/358336/an-overview-of-programming-for-intel-xeon-processors-and-intel-xeon-phi-coprocessors.pdf)

- Краткое введение для разработчика Intel Xeon Phi:

  (http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide)

- Руководство разработчика системного ПО Intel Xeon Phi: (https://secure-software.intel.com/sites/default/files/article/334766/intel-xeon-phi-systemsoftwaredevelopersguide.pdf)
- Справочник по архитектуре и набору команд Intel Xeon Phi: (http://software.intel.com/sites/default/files/forum/278102/327364001en.pdf)

Также Intel Xeon Phi Developer Zone (http://software.intel.com/mic-developer) содержит большое количество справочного и учебного материала по настройке, использованию и программированию Intel Xeon Phi.

7. Ссылочная документация

Для получения более подробной информации по нижеперечисленным продуктам обратитесь к соответствующим ссылкам:

7.1 Пакет Environmental Modules:

- Основная документация (http://modules.sourceforge.net/man/module.html)

7.2 Планировщик задач SLURM.

- Основная документация (https://computing.llnl.gov/linux/slurm/documentation.html)
- Ответы на основные вопросы (https://computing.llnl.gov/linux/slurm/faq.html)

7.3 Сопроцессор Intel Xeon Phi

- Страница продукта (http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html)
- Страница разработчика (http://software.intel.com/mic-developer)
- Building a Native Application for Intel® Xeon Phi™ Coprocessors (http://software.intel.com/en-us/articles/building-a-native-application-for-intel-xeon-phi-coprocessors)
- Programming and Compiling for Intel® Many Integrated Core Architecture (http://software.intel.com/en-us/articles/programming-and-compiling-for-intel-many-integrated-core-architecture)
- Math Kernel Library Automatic Offload for Intel® Xeon Phi™ Coprocessor (http://software.intel.com/en-us/articles/math-kernel-library-automatic-offload-for-intel-xeon-phi-coprocessor)
- Using the Intel® MPI Library on Intel® Xeon Phi™ Coprocessor Systems (http://software.intel.com/en-us/articles/using-the-intel-mpi-library-on-intel-xeon-phi-coprocessor-systems)
- Differences in Floating-point Arithmetic (http://software.intel.com/sites/default/files/article/326703/floating-point-differences-sept11.pdf)
- Intel® Math Kernel Library Link Line Advisor полезный инструмент для определения как компилировать и линковать с MKL для разных сценариев. (http://software.intel.com/sites/products/mkl/)

7.4 Intel Parallel Studio XE 2013

- Страница продукта (http://software.intel.com/ru-ru/intel-parallel-studio-xe)
- Evaluation Guide Portal (http://software.intel.com/en-us/evaluation-guides/)

8. Возможные проблемы и способы их устранения

- Проблема: при попытке пользователя получить доступ на выделенный вычислительный узел или сопроцессор Intel Xeon Phi система запрашивает пароль.
  Возможная причина: Нет записи публичного ключа в файле *~/.ssh/authorized_keys*
  Решение: проверьте наличие записи публичного ключа в файле *~/.ssh/authorized_keys*

9. Список сокращений

Термин	Описание
BMC	Baseboard Management Controller – встроенный контроллер управления
BIOS	Basic Input/Output System – базовая система ввода-вывода
CPU	Central Processing Unit. Центральный процессор.
DHCP	Dynamic Host Configuration Protocol – протокол динамического конфигурирования узла, автоматическое получение сетевых настроек
DDR	Double data rate – двукратная скорость
DIMM	Dual In-line Memory Module – модуль памяти с двухрядным расположением микросхем
Infiniband	Высокоскоростная коммутируемая последовательная шина
IPMI	Intelligent Platform Management Interface – интеллектуальный интерфейс управления платформой
KVM	Keyboard, Video and Mouse – удалённое управление посредством перенаправления видео, клавиатуры и мыши
LAN	Local Area Network – локальная компьютерная сеть
LDAP	Упрощённый протокол доступа к каталогам, протокол LDAP
MPI	Message Passing Interface - программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.
PCI-Express, PCI-E	Высокоскоростная шина последовательного ввода-вывода
NIC	Network Interface Controller – сетевой контроллер
NMI	Non-Maskable Interrupt – немаскируемое прерывание
QDR	Quad Data Rate – четырехкратная скорость
QPI	Quick Path Interconnect – тип системной шины
RAID	Redundant array of independent disks – избыточный массив независимых жёстких дисков
SATA	Serial Advanced Technology Attachment — последовательный интерфейс обмена данными, используемый для подключения накопителей
SEL	System Event Log – аппаратный журнал системы
SFTP	SSH File Transfer Protocol — протокол прикладного уровня, предназначенный для копирования и выполнения других операций с файлами поверх надёжного и безопасного соединения.
SLURM	Менеджер ресурсов с открытым кодом для вычислительных систем под управлением Linux.
SSH	Secure Shell — «безопасная оболочка» — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование соединений (например, для передачи файлов).
TCP/IP	Протокол управления передачей / межсетевой протокол

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм	*Номера листов (страниц)*				*Всего листов (страниц) в док.*	*№ разреш. документа*	*Подпись*	*Дата*	*Примечание*
	Измененных	*Замененных*	*Новых*	*Аннулированных*					