Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук
Центр коллективного пользования «Дальневосточный вычислительный ресурс»

Руководство пользователя кластера IRUS17

(версия 1.1 от 03.02.2018 г.)

Владивосток
2018 г.

Содержание

СПИСОК СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

СХД - Система Хранения Данных

MPI - Message Passing Interface, Интерфейс Передачи Сообщений

ОС - Операционная Система

NFS - Network File System, Сетевая Файловая Система

ПО - Программное Обеспечение

1. Краткое техническое описание кластера

Многопроцессорная вычислительная система (кластер) IRUS17 имеет традиционную кластерную архитектуру. Структурная схема IRUS17 показана на рис. 1. Система состоит из 4 основных компонентов (выделено серым): узел доступа, узел управления, расчетное поле и система хранения данных.

Узел доступа (или **пользовательский узел**) - это узел, на который разрешается удаленный доступ пользователей. Данный узел отвечает за авторизацию пользователя, предоставление ему доступа к домашнему каталогу для размещения собственных данных и программ, предоставление средств настройки среды работы для компиляции и запуска программ, и предоставление средств для запроса ресурсов и запуска задач.

Узел управления - это узел, на котором работают основные системные службы, обеспечивающие работу всей системы в целом.

Расчетное поле (или **вычислительное поле**) - это группа узлов, на которых происходит выполнение задач пользователей. Узлы из этой группы выделяются пользователям по запросу. За составление расписания и выделение узлов из этой группы отвечает специальное программное обеспечение, расположенное на управляющем узле.

Система хранения данных - это выделенный узел, к которому подключены массивы жестких дисков. Узел СХД отвечает за хранение пользовательской информации и обеспечения к ней доступа со всех остальных узлов системы.

Компоненты системы соединены между собой сетями различного вида, в зависимости от решаемых с их помощью задач. Всего в составе системы работает три сети: управляющая сеть, сеть хранения данных и коммуникационная сеть.

Управляющая сеть используется для передачи команд управления между узлами системы. Посредством этой сети передается служебный трафик: настройки, выделение адресов, авторизация пользователей и других служб и сервисов системы. Именно через управляющую сеть происходит передача команд на запуск заданий пользователей.

Сеть хранения данных используется для передачи данных при обращении к общесистемным сетевым директориям. Таких директорий всего три:

1. системная - для хранения системных данных и настроек; пользователь не имеет к ней доступа;
2. домашние директории пользователей - для хранения данных пользователей;
3. каталог предустановленного программного обеспечения - для установки и использования общедоступного программного обеспечения.

Коммуникационная сеть используется исключительно для связи узлов между собой и предназначена только для передачи данных параллельных программ с помощью MPI. Другие данные по этой сети не передаются. Все реализации MPI, предустановленные в системе, настроены на использование этой сети при компиляции и запуске параллельных программ.

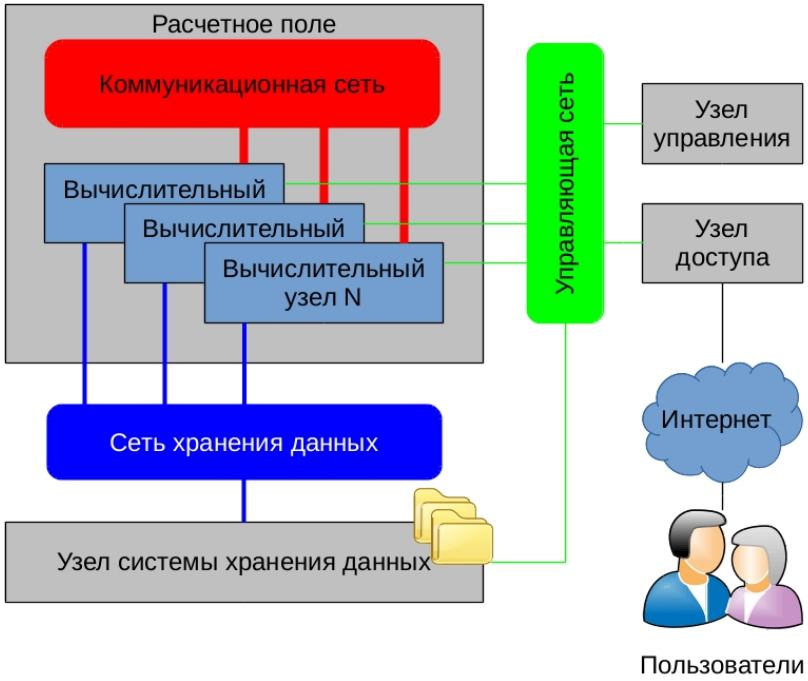


Рис. 1. Структурная схема кластера IRUS17.

В таблице 1 кратко приведены основные технические характеристики системы.

Таблица 1. Краткие технические характеристики кластера IRUS17

Характеристика	**Значение**
Официальное название системы	IRUS17
Количество вычислительных узлов, шт	40
Количество процессоров на узле, шт	2
Количество ядер у каждого процессора, шт	20
Технология Hyper-threading (HT)	присутствует(1)
Всего ядер на узле, шт	40
Всего ядер в расчетном поле, шт	1600
Модель процессора (базовая частота, ГГц)	Intel Xeon E5-2698 v4 (2,2)
Пиковая производительность системы, TFLOPS	56,32
Пиковая производительность системы с учетом понижения частоты AVX, TFLOPS	46,08

Управляющая сеть	Gigabit Ethernet
Сеть хранения данных	Infiniband FDR (56 Гбит/с)
Коммуникационная сеть	Intel Omni-Path (100 Гбит/с)
Объем системы хранения, Тбайт	216(2)
Операционная система	CentOS Linux 7.2

(1) Технология HT представляет каждое ядро в ОС в виде двух логических процессоров. Фактически на каждом узле доступно 80 ядер, Однако, технология HT не добавляет вычислительной мощности ядру. Пользователям необходимо убедиться, что их программное обеспечение действительно адаптировано для работы с использованием технологии HT. В противном случае, рекомендуется ориентироваться на 40 ядер на каждом узле при заказе ресурсов.

(2) 216 Тбайт - это «сырой» объем, не учитывающий требований надежности и отказоустойчивости хранения данных. Фактически для пользовательских данных выделено порядка 150 Тбайт свободного пространства.

2. Доступ пользователей

Доступ пользователей осуществляется удаленно только по протоколу SSH (версия 2) с использованием методов авторизации по паролю или пары открытого/закрытого ключа.

Доменное имя узла доступа в сети Интернет: irus17.cc.dvo.ru

Все пользователи должны работать через этот узел.

Обращаем внимание, что доступ к узлу контролируется системой предупреждения вторжений и защиты от атак! По умолчанию система ЗАПРЕЩАЕТ доступ к узлу с неизвестного IP-адреса. Для постоянных пользователей Администраторы активируют разрешающие правила (обычно с достаточно широким диапазоном адресов). В настоящее время, система пропускает соединения пользователей от практически всех провайдеров Приморского края, включая мобильных операторов, а также ряда других территориально распределенных пользователей. Если вы не можете получить доступ, необходимо обратиться к Администраторам системы для активации правил по вашему IP-адресу. Заявки, как правило, выполняются в течение суток. В случае, если Вы хотите получать доступ к системе находясь в поездках, вы также можете в любое время обратиться к Администраторам. Все контакты Администраторов указаны в конце данного руководства.

В процессе работы пользователям доступно два основных способа взаимодействий с системой: командный (терминал) и файловый интерфейсы.

Для работы пользователей удаленно со своих рабочих мест рекомендуется использовать следующее программное обеспечение.

- Для операционной системы Windows:
  - Putty - для командного взаимодействия (http://www.putty.org/)
  - WinSCP - для файлового взаимодействия (https://winscp.net/eng/download.php)
- Для операционной системы Linux:
  - openssh - для командного взаимодействия
  - sshfs - для файлового взаимодействия

Для остальных операционных систем пользователям рекомендуется обратиться к Администраторам. В приложении 1 показан пример настройки Putty для IRUS17.

При работе через терминал пользователю доступна оболочка командной строки BASH (Bourne-again Shell). В случае необходимости установки и поддержка других оболочек, в том числе необходимых для работы прикладных расчетных пакетов, пользователю необходимо обратиться к Администраторам.

```
[user@irus17 FAST@~]$ _
```

3. Описание структуры файловой системы кластера

Файловая система IRUS17 построена по традиционному для семейства Unix-подобных ОС иерархическому принципу. Каждый узел кластера имеет собственный образ файловой системы, в котором размещаются файлы операционной системы, которая управляет данным узлом. Размер файловой системы составляет 64 Гбайта и предназначен только для хранения системных файлов, необходимых для работы узла. Все пользовательские данные располагаются на сетевом диске, доступ к которому осуществляется через NFS.

На кластере выделено 4 раздела, которые доступны через NFS. Описание разделов представлено в таблице 2

Таблица 2. Сетевые разделы

Точка монтирования	**Размер**	**Назначение**
/home/Fast	110 Тбайт	Для размещения временных данных пользователей в процессе расчета. Физически построено для более быстрой записи данных. Не может использоваться для долговременного хранения данных. ***Резервного копирования нет. В случае сбоя системы файлы из этого раздела стираются. При проведении профилактических работ файлы могут быть удалены.***
/home/Safe	50 Тбайт	Для размещения долговременных данных пользователей. **Резервное копирование — 1 раз в день.**
/common	5,5 Тбайт	Для размещения каталога предустановленного программного обеспечения общего пользования.
/system	5,5 Тбайт	Для размещения системной информации. Используется только администраторами, пользователям недоступно. Совместно использует дисковое пространство с /common.

В разделах Fast и Safe для каждого пользователя выделена домашняя директория, в которой он может располагать свои файлы. Структура домашней директории не регламентируется. Пользователь имеет право размещать файлы и директории в любом удобном порядке. Домашняя директория пользователя доступна на чтение и запись только этому пользователю. Работа с файлами и директориями через командный интерфейс осуществляется с использованием основных команд Linux: ls, cd, mkdir, rm, cp, mv и так далее. Параметры для этих команд представлены в справочных руководствах, установленных на кластере (команда man, см. Приложение 2). Для удобства использования на кластере предустановлен Midnight Commander (команда mc), который имеет интерфейс аналогичный (Far Manager, Total Commander и другим подобным менеджерам файлов). Аналогично можно использовать WinSCP, чтобы удаленно управлять своими файлами. Также допускается монтирование через ssh (sshfs) для удобства удаленной работы пользователя.

Домашним каталогом по умолчанию при входе пользователя является каталог, расположенный в разделе /home/Safe !!

В командной строке имеется вспомогательное поле, которое обозначает, в каком разделе сетевой файловой системы располагается текущая директория. Приглашение командной строки IRUS17 выглядит следующим образом

```
[user@irus17 SAFE@~]$ pwd
/home/Safe/user
```

В квадратных скобках слева направо указывается: имя пользователя и имя узла через разделитель '@', после пробела название раздела и текущая директория также через разделитель '@'. Название раздела может содержать следующие значения:

- 'SAFE', тогда текущая директория принадлежит точке монтирования /home/Safe
- 'FAST', тогда текущая директория принадлежит точке монтирования /home/Fast
- '' (пустая строка), тогда текущая директория находится за пределами указанных точек монтирования

На вычислительных узлах (узлах расчетного поля) разделы Fast и Safe имеют одно важное отличие. **На запись доступен только раздел Fast, раздел Safe имеет режим доступа только для чтения. Таким образом, в процессе счета пользователь может использовать только директорию /home/Fast/$USER для записи как промежуточных расчетных данных, так и окончательных результатов.** По окончанию счета пользователю рекомендуется все важные данные переносить из раздела Fast в раздел Safe для долговременного и безопасного хранения. Для удобства работы с двумя разделами, составляющими домашнюю директорию пользователя, предусмотрено несколько вспомогательных команд, расширяющих стандартный набор команд Linux, приведенных выше.

mkwd (make working directory) - воссоздает структуру директорий верхнего уровня в разделах Fast и Safe по заданной текущей директории.

Пример. Пусть текущая директория пользователя будет

```
[user@irus17 SAFE@somedir]$ pwd
/home/Safe/user/subdir1/subdir2/somedir
```

Предполагается, что данная директория уже создана пользователем в разделе Safe каким-либо образом. Тогда, в результате выполнения команды mkwd будет создана директория somedir в разделе Fast с воссозданием всех поддиректорий верхнего уровня.

```
/home/Fast/user/subdir1/subdir2/somedir
```

Аналогична обратная ситуация. Если текущая директория пользователя располагается в разделе Fast, то выполнение mkwd приведет к созданию аналогичной директории в разделе Safe. Если данная директория уже существует, то действие команды будет проигнорировано. Обратите внимание, что внутреннее содержимое самой директории не синхронизируется и не воссоздается. Для этого имеется другая расширенная команда (см. далее).

cdfast (change directory on Fast partition) - изменить текущую директорию на аналогичную директорию, но в разделе Fast. Если данная директория отсутствует, то будет выдана соответствующая ошибка. Если текущей директорией уже является директория в разделе Fast, то действие команды будет проигнорировано.

```
[user@irus17 SAFE@somedir]$ pwd
/home/Safe/user/subdir1/subdir2/somedir
[user@irus17 SAFE@somedir]$ cdfast
[user@irus17 FAST@somedir]$ pwd
/home/Fast/user/subdir1/subdir2/somedir
```

или

```
[user@irus17 SAFE@somedir]$ pwd
/home/Safe/user/tempdir/somedir
[user@irus17 SAFE@somedir]$ cdfast
/home/Fast/user/tempdir/somedir: каталог отсутствует
[user@irus17 SAFE@somedir]$ pwd
/home/Safe/user/tempdir/somedir
```

cdsafe (change directory on Safe partition) - изменить текущую директорию на аналогичную директорию, но в разделе Safe. Работа данной команды аналогична команде **cdfast**.

syncwd (synchronize working directory) - синхронизирует содержимое текущей директории в разделах Safe и Fast. **Обратите внимание на следующие особенности работы данной команды**:

а) перед выполнением синхронизации текущий каталог запуска данной команды будет проверен и при необходимости воссоздан в разделах Fast или Safe (делается автоматически, не требует от пользователя дополнительных действий);

б) когда данная команда выполняется в текущей директории, расположенной в разделе Safe, то выполняется синхронизация только в направлении Safe → Fast; новые и измененные файлы текущей директории (с вложенными поддиректориями) будут скопированы из директории раздела Safe в директорию раздела Fast; **если в директории Fast были другие файлы, которых по каким-то причинам не было в Safe, то эти файлы будут стерты из раздела Fast**;

в) когда данная команда выполняется в текущей директории, расположенной в разделе Fast, то выполняется синхронизация только в направлении Fast → Safe; **новые и измененные файлы текущей директории (с вложенными поддиректориями) будут перенесены (с последующим удалением) из директории раздела Fast в директорию раздела Safe**; если в директории Safe были другие файлы, которых по каким-то причинам не было в Fast, то эти файлы останутся нетронутыми;

Таким образом, общая политика работы команды syncwd состоит в том, чтобы директории в разделе Fast считать временными и удалять из них файлы при каждой синхронизации, и, наоборот, директории в разделе Safe считать постоянными и сохранять в них все файлы, которые пользователь специально не удалял.

Пример. Пусть пользователь собирается провести вычислительный эксперимент и подготовил необходимые входные данные для его выполнения. Предполагается, что для проведения вычислительного эксперимента пользователь создал отдельную директорию, где будет размещать все необходимые данные эксперимента.

```
[user@irus17 SAFE@exp1]$ pwd
/home/Safe/user/experiments/exp1
[user@irus17 SAFE@exp1]$ ls
exp1.inp input1.dat input2.dat
```

Пусть для запуска и проведения расчетов необходимо наличие трех файлов exp1.inp, input1.dat и input2.dat. Обращаем внимание, что в данном примере все действия пользователь проводит на узле доступа в разделе Safe, который доступен на запись. Тогда запуск команды syncwd приведет к выполнению следующих действий. Сначала будет запущена команда mkwd, которая воссоздаст структуру директорий в разделе Fast, т. е. появится директория/home/Fast/user/experiments/exp1. После этого будет проведена синхронизация содержимого и три файла будут скопированы из указанной директории в разделе Fast в директорию в разделе Safe.

```
[user@irus17 SAFE@exp1]$ syncwd
```

```
[user@irus17 SAFE@exp1]$ cdfast
[user@irus17 FAST@exp1]$ pwd
/home/Fast/user/experiments/exp1
[user@irus17 FAST@exp1]$ ls
exp1.inp input1.dat input2.dat
```

Предположим, что после выполнения счета в рабочей директории эксперимента появились файлы-результаты вычислений

```
[user@irus17 FAST@exp1]$ ls
exp1.inp input1.dat input2.dat output1.dat output2.dat
```

Тогда повторный запуск **syncwd** в директории эксперимента в разделе Fast приведет к переносу всех 5 файлов в соответствующую директорию в разделе Safe. Директория эксперимента в Fast будет пустой, директория эксперимента в разделе Safe будет содержать все необходимые файлы.

```
[user@irus17 FAST@exp1]$ ls
[user@irus17 FAST@exp1]$ cdsafe
[user@irus17 SAFE@exp1]$ ls
exp1.inp input1.dat input2.dat output1.dat output2.dat
```

4. Предустановленное программное обеспечение

Предустановленное программное обеспечение располагается в разделе /common. Для удобства использования данного программного обеспечениея на кластере развернута система модулей (команда module), которая позволяет настроить сеанс работы пользователя на некоторую группу необходимого ПО.

Для вывода доступного через систему модулей ПО используется команда module available или сокращенно:

```
[user@irus17 SAFE@~]$ module avail
---------------------------------- /common/modulefiles/mpi ----------------------------------
impi-2017.2.174       mvapich2_intel_hfi-2.1  openmpi_intel_hfi-1.10.4
mvapich2_gcc_hfi-2.1  openmpi_gcc_hfi-1.10.4
-------------------------------- /common/modulefiles/devel ----------------------------------
gcc-5.4.0  intel-2017.2.174
-------------------------------- /common/modulefiles/apps -----------------------------------
boltztrap-1.2.5_intel        lammps-2017.03.31-serial  ShengBTE-v1.1.1-8a63749_intel
ffmpeg                       mplayer-1.3.0             tasker/2.2_intel
```

```
gnuplot-5.1                  nasm-2.14rc0              tasker/2.2_mvapich2_gcc
gromacs-5.0.7_intel          octave-4.2.1_intel        tasker/2.2_mvapich2_intel
lammps-2017.03.31-mpi_intel  phonopy-1.11.10.66
------------------------------- /common/modulefiles/libs --------------------------------
fftw-3.3.6-pl2_intel  hdf5-1.8.18_intel  netcdf-4.4.1.1_intel        proj4-4.4
gdal-2.2.0            mkl-20170002       netcdf-fortran-4.4.4_intel  x264
```

Все ПО разбито на 4 группы.

- mpi - это установленные и поддерживаемые на кластере реализации MPI;
- devel - средства разработки (компиляторы, отладчики, профилировщики и т. п.);
- libs - вспомогательные библиотеки (математические, обработки данных, преобразования форматов и т. п.);
- apps - прикладные расчетные пакеты (также пре- и постпроцессинг).

В системе модулей отражается только то ПО, которое либо является свободным, либо Администраторы имеют лицензию на его предоставление. Размещение ПО под частной лицензией пользователя не допускается. Такое проприетарное ПО пользователь должен размещать в своей домашней директории.

Для настройки сеанса пользователя на конкретный пакет осуществляется командой module load

```
[user@irus17 SAFE@~]$ module load gromacs-5.0.7_intel
```

Данная команда встраивает в сеанс пользователя необходимые переменные среды (дополняет значения этих переменных), которые помогают системе находить исполняемые и другие вспомогательные файлы, относящиеся к данному пакету. Чаще всего настраиваются такие переменные, как

- PATH - пути поиск исполняемых файлов;
- LD_LIBRARY_PATH - пути поиска подгружаемых библиотек;
- PKG_CONFIG_DATA - пути поиска конфигурационных файлов пакетов для pkgconfig.

Детали настроек каждого пакета можно посмотреть в /common/modulefiles. Если Вы заметили ошибку или есть предложения по дополнению работы модуля, просьба сообщить об этом Администраторам.

Список текущих загруженных модулей пользователя можно посмотреть с помощью команды module list

```
[user@irus17 SAFE@~]$ module list
gromacs-5.0.7_intel
```

При каждом новом сеансе работы состояние загрузки модулей сбрасывается. Если Вы постоянно работаете с одним и тем же набором модулей, их можно поставить в автозагрузку. Для этого достаточно в файл .bash_profile (с точкой!!)

5. Компиляция программного обеспечения

На кластере IRUS17 доступны следующие компиляторы языков C/C++/Fortran:

- GCC4.8 - установлен вместе с операционной системой и не требует загрузки специальных модулей
- GCC5.4 - доступен через модуль gcc-5.4.0
- Intel - набор компиляторов от Intel в составе пакет Intel Parallel Studio Cluster Edition, доступен через модуль intel-2017.2.174.

Все версии компиляторов поддерживают стандарт OpenMP (разные его версии для разных компиляторов, уточняйте детали в руководствах по компиляторам).

Компиляция производится по всем правилами, принятым в Linux. Доступны для использования системы сборки make (через Makefile) и cmake (CmakeFile). Также возможно использование GNU Build System (или Autotools), включающую autoconf/automake.

Для компиляции MPI-приложений на кластере поддерживается 5 реализаций MPI. Все они доступны через соответствующие модули:

- impi-2017.2.174 - Intel MPI (базовый компилятор Intel C/C++/Fortran)
- mvapich2_gcc_hfi-2.1 MVAPICH2 (базовый компилятор от GCC)
- mvapich2_intel_hfi-2.1 MVAPICH2 (базовый компилятор от Intel)
- openmpi_gcc_hfi-1.10.4 OpenMPI (базовый компилятор от GCC)
- openmpi_intel_hfi-1.10.4 OpenMPI (базовый компилятор от Intel)

Все компиляторы уже настроены на использование коммуникационной сети кластера для передачи данных через MPI. По умолчанию поддерживается следующая настройка внутренних подсистем MPI для передачи сообщений: если MPI-процессы располагаются на одном узле, использовать общую память для передачи сообщений, в противном случае, использовать интерфейс Intel Omni-Path (сокращение HFI - Host Fabric Interface). В случае тонкой оптимизации производительности пользователь может самостоятельно задавать доступные внутренние настройки и режимы работы MPI для его конкретных задач.

Компиляция MPI-программ осуществляется с помощью соответствующих компиляторов-оберток над базовым компилятором (mpicc/mpicxx/mpiifort и т.д.). Отметим, что в случае использования базового компилятора от Intel, пользователю необходимо дополнительно загрузить модуль intel-2017.2.174 перед загрузкой модуля MPI.

Примеры компиляции

Пример 1. Компиляция из командной строки

Выполнение следующей команды:

```
[user@irus17 FAST@~]$ mpicxx –o ExecutableFile *.cpp
```

откомпилирует все файлы в текушей директории с расширением «cpp» и сделает исполняемый файл с именем ExecutableFile.

Аналогично для программы на языке C:

```
[user@irus17 FAST@~]$ mpicc –o ExecutableFile *.c
```

Пример 2. Компиляция с использованием Makefile

Необходимо создать в директории с исходными файлами программы на языке “C++” файл под названием Makefile примерно следующего содержания:

```
# Makefile – Universal makefile for compiling all cpp files
# into executable with name of current directory
SOURCES=$(wildcard *.cpp)
OBJECTS=$(SOURCES:%.cpp=%.o)
CURDIR=$(shell basename `pwd`)
```

```
OUTFILE=$(shell basename `pwd`)
INCLUDE_DIR=-I ../
CFLAGS =  -O2 $(INCLUDE_DIR)
CXXFLAGS = $(CFLAGS)
LDFLAGS= -lstdc++

all: $(OUTFILE)

rebuild : clean ALL

CC=mpicxx
LNK=$(CC)

.cpp.o :
  $(CC) $(CXXFLAGS) -c $< -o $@
%.o: %.cpp
  $(CC) $(CXXFLAGS) -c $< -o $@
$(OUTFILE): $(OBJECTS)
  $(LNK) -o $(OUTFILE) $(LDLAGS) $(OBJECTS)
clean:
  $(RM) $(OBJECTS) $(OUTFILE)
```

При наличии такого файла команда make, выполненная в этом каталоге, скомпилирует все файлы с расширением «cpp» и сделает исполняемый файл с названием, совпадающим с названием текущей директории. Вариант использования make является более гибким, так как позволяет настраивать опции компилятора в различных переменных CFLAGS, CXXFLAGS, LDFLAGS и другие, а также использовать гибкую систему правил сборки для пакетов с объемным исходным кодом.

6. Управление заданиями пользователя

Для управления ресурсами кластера, а именно ресурсами узлов расчетного поля, используется система SLURM Workload Manager (в прошлом просто SLURM от Simple Linux Utility for Resource Management). Диспетчер данной системы следит за тем, кто, в каком объеме и на какое время занимает ресурсы узлов кластера. В настоящее время единицей выделения ресурсов является одно физическое ядро процессора (включая работы в режиме гиперпоточности). SLURM имеет четыре основные компоненты:

диспетчер системы (или slurmctld) - работает на узле управления и отвечает за всю работу системы

диспетчер узла (или slurmd) - работает на каждом узле расчетного поля (отдельный экземпляр) и отвечает за прохождение расчетной задачи через данный узел. А именно,

обеспечение запуска MPI приложения, контроль использования количества ядер, контроль времени исполнения, принудительное завершение приложения и другие функции.

диспетчер базы данных (или slurmdbd) - работает на узле управления и отвечает за сбор всей статической информации о работе всех расчетных задач всех пользователей.

пользовательские утилиты (или sbatch/salloc/srun/scancel/sacct/sstat и ряд других) - предназначены для взаимодействия пользователя с диспетчерами SLURM. Взаимодействие пользователя со SLURM чаще всего сопровождается указанием уникального номера задания, которое автоматически назначается пользователю после успешного выделения ресурсов. По данном номеру задания пользователь может получить всю информацию о его работе.

6.1 Основные команды пользователя SLURM для управления заданиями

sinfo - состояние узлов кластера
squeue - состояние очереди заданий кластера
srun - непосредственный запуск заданий
salloc - запрос ресурсов и их использование в интерактивном режиме
sbatch - фоновый запуск заданий (через очередь)
scancel - отмена задания (прерывание исполнения, если в данный момент считается)
sacct - статистика обработанных заданий
sstat - статистика исполняемых заданий
scontrol - управление диспетчерами SLURM
sreport - отчетная информация

Опишем основные команды, которые чаще всего, необходимы пользователю для запуска заданий

6.2 Запрос ресурсов кластера (sbatch/salloc/srun)

Данная группа команд используется для указания системе запустить некоторым образом программу пользователя на заданном количестве ресурсов на заданное время. Чаще всего требуются следующие 3 параметра.

-n <число> : кол-во запрашиваемых ядер, соответствующих количеству параллельных процессов задания пользователя; в соответствии с этим числом из набора доступных

выделяется набор узлов кластера, которые далее будут использоваться для запуска задания пользователя;

-N <число> : минимальное кол-во узлов кластера, необходимое для запуска задания

-t DD-HH:MM:SS : время в формате ДНЕЙ-ЧАСОВ:МИНУТ:СЕКУНД, на которое требуется выделить узлы/ядра кластера;

Примеры

```
salloc -N 2 -n 80 -t 5-00:00:00 myjob.sh
```

Запросить минимум два полностью свободных от других задач узла кластера, на которых необходимо запустить задание myjob.sh, которому требуется 80 ядер

```
salloc -n 80 -t 5-00:00:00 myjob.sh
```

Запросить некоторое количество узлов, на которых набирается 80 свободных ядер (не факт, что их будет 2 в связи с тем, что могут исполняться и другие задания), на которых необходимо запустить задание myjob.sh, которому требуется 80 ядер.

```
salloc -n 16 -t 5-00:00:00 myjob.sh
```

Аналогично верхнему примеру. Здесь тоже не гарантируется, что ядра могут быть выделена на одном узле. Хотя по умолчанию система будет в первую очередь выделять те ядра, которые с точки зрения физической топологии располагаются ближе друг к другу (ядра одного процессора).

```
salloc -N 1 -n 64 -t 12:00:00 myjob.sh
```

В данном случае задание будет заблокировано, так как пользователь запросил несовместимое количество ресурсов. **На каждом узле кластера доступно не более 40 ядер.**

```
salloc -N 1 -n 40 -t 9-12:00:00 myjob.sh
```

В данном случае задание тоже будет заблокировано, так как пользователь запросил время счета, превышающее текущий установленный предел. **В настоящее время максимально время счета одной задачи не может быть выше 7 дней.**

```
salloc -N 10 -n 400 -t 12:00:00 myjob.sh
```

Это тоже пример задания, которое будет по умолчанию заблокировано. Пользователь запросил количество ядер и узлов, превышающее текущий установленный предел. В настоящее время пользователю доступно для счета всех его задач не больше 6 узлов (240 ядер) единовременно.

Отдельно отметим, что все описанные пределы активны по умолчанию. В некоторых случаях Администраторы для отдельных заданий могут отменить действие пределов и, таким образом, разрешить выполнение заданий на большом количестве ядер на большое время. Такие запуски требуют предварительного согласования с Администраторами. Пользователю необходимо направить запрос по электронной почте с кратким указанием причины такой необходимости. Все запросы рассматриваются в индивидуальном порядке и зависят от текущей общей загрузки кластера.

Все три команды sbatch/alloc/srun принимают указанные выше параметры. Все три команды отправляют запрос на ресурсы, и, если таковые ресурсы имеются в наличии, то переходят к запуска задания. Отличие состоит в том, что делает команда дальше с параметром, указывающим на исполняемое задание, а также в том, что произойдет, если свободных ресурсов в данный момент не окажется.

Интерактивный запуск

salloc [запрос ресурсов] [<команда> [<параметры команды> …]]

После выделения ресурсов salloc запустит (на узле доступа, с которого работает пользователь!!!) заданную команду. По окончанию исполнения команды произойдет автоматическое освобождение ресурсов. Данную команду удобно использовать вместе с интерпретатором командной строки

```
[user@irus17 FAST@~]$ salloc [запрос ресурсов] /bin/bash -l
salloc: Granted job allocation NNNN
[user@irus17 FAST@~]$ exit
salloc: Relinquishing job allocation NNNN
[user@irus17 FAST@~]$
```

В этом случае пользователю запускается новый экземпляр оболочки, который имеет привязку к выделенным ресурсам, и пользователю сообщается НОМЕР ЕГО ЗАДАНИЯ. Пользователь может самостоятельно перейти на выделенный узел командой ssh irus17-ABC (где ABC - номер узла) и осуществить необходимый запуск заданий. Поиск выделенного узла можно осуществить командой squeue (см. далее). По выходу из данного экземпляра оболочки произойдет автоматическое освобождение ресурсов.

Фоновый запуск

sbatch [запрос ресурсов] [<bash-скрипт> [<параметры скрипта> ...]]

```
[user@irus17 FAST@~]$ sbatch -n 20 ./testjob.sh
Submitted batch job NNNN
[user@irus17 ~]$
```

sbatch сразу запросит выделение указанных ресурсов, выведет уникальный номер задания пользователю в соответствии с запросом и на этом завершит свое выполнение. Запуск скрипта уже будет контролироваться диспетчерами SLURM самостоятельно. Пользователю лишь необходимо дождаться окончания выполнения задания.

Параллельный запуск

srun [запрос ресурсов] [<MPI-команда> [<параметры команды> ...]]

После выделения ресурсов srun запустит (на выделенном узле(-ах) заданную MPI-программу). По окончанию исполнения произойдет автоматическое освобождение ресурсов. Данную команду удобно использовать для очень коротких запусков при наличии свободных ресурсов. Если в качестве команды srun будет предоставлено не MPI-программа, то запущенно такое количество экземпляров, которое соответствует параметру -n.

6.3 Особенности запуска параллельных заданий (MPI)

Запуск параллельных длительных вычислительных заданий рекомендуется производить следующим образом. Пользователь готовит скрипт запуска (bash-скрипт), который будет запускаться через sbatch. Содержимое скрипта может быть достаточно произвольным до собственно момента запуска MPI-программы. Пример файла

```
#!/bin/bash
user-cmd1
user-cmd2
#... здесь может идти любые допустимые в bash команды

# ПАРАЛЛЕЛЬНЫЙ ЗАПУСК MPI
srun mympiprog1 param11 param12

# ПАРАЛЛЕЛЬНЫЙ ЗАПУСК MPI
srun mympiprog2 param21 param22

user-cmd3
user-cmd4
#... здесь может идти любые допустимые в bash команды

#... ПАРАЛЛЕЛЬНЫЙ ЗАПУСК MPI

#... здесь может идти любые допустимые в bash команды
```

Синтаксически данный файл должен представлять из себя обычный скрипт, написанный на языке командной оболочки. Допускается использовать все, что поддерживается самим интерпретатором командой строки, который указывается в первой строчке скрипта. Как указывалось выше, поддержка других интерпретаторов допускается. Пользователь должен проверить, что данный файл является исполняемым

```
[user@irus17 FAST@~]$ chmod +x myjob.sh
```

Необходимо обратить внимание на параллельный запуск. Он должен выполняться с помощью команды SLURM srun (по аналогии с mpirun, предлагаемой всеми реализациями MPI). В скрипт у srun можно переопределять параметры запуска. Например, можно изменить значение параметра -n - кол-во параллельных процессов, которые будут созданы при запуске параллельнйо программы. Если srun ничего не указывается, то по умолчанию переносятся все параметры из sbatch и salloc.

В одном скрипте допускается поочередный запуск множества MPI-программ. Каждый srun в порядке следования в скрипте будет выполняться по завершению предыдущего.

Непосредственный запуск скрипта осуществляется следующим образом. Когда стало известно какие узлы выделены под задачу пользователя, но одном из этих узлов (главный узел) диспетчером узла запускается данный скрипт. Скрипт выполняется интерпретатором

последовательно в соответствии с содержимым скрипта от имени пользователя, запустившего задание. Когда встречается srun, он автоматически начинает взаимодействие с SLURM, и производит запуск MPI-программы. Осуществляет разбиение и порождение параллельных процессов, обеспечивает настройку сетевой среды. По завершению параллельного запуска, все процессы уничтожаются, и продолжается последовательное выполнение скрипта.

6.4 Запрос состояния выполнения

Для того чтобы узнать состояние узлов кластера используется команды sinfo.

```
[user@irus17 FAST@~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
standard*    up 7-00:00:00     16  down* irus17-[25-40]
standard*    up 7-00:00:00      7  alloc irus17-[4-10]
standard*    up 7-00:00:00     17   idle irus17-[1-3,11-24]
```

Основной вывод состоит из шести столбцов

- PARTITION - раздел кластера. На IRUS17 всего один раздел standard, который активен по умолчанию и всегда используется для запуска заданий. Фактически раздел можно представлять как очередь, в которую задания попадают по заявкам от пользователей.
- AVAIL - состояние раздела
- TIMELIMIT - ограничение на время счета в очереди
- NODES - количество узлов в очереди с заданным состоянием и именами
- STATE - состояние узлов
  - down - выключены в административном порядке
  - alloc - используются для счета
  - mix - используются для счета (обычно обозначает частичное использование ресурсов)
  - maint - узлы на техническом обслуживание, временно выведены из счета
  - idle - свободны
- NODELIST - сетевые имена узлов. Квадратные скобки обозначают диапазоны узлов. Например, irus17-[1-3,11-24] обозначает узлы с сетевыми именами irus17-1, irus17-2, irus17-3, irus17-11, irus17-12 и т. д. irus17-24. По этим именам пользователь может с помощью SSH переходить на узлы и отслеживать непосредственную работу его расчетов. Выход с узла осуществляется стандартной командой exit.

```
[user@irus17 FAST@~]$ ssh irus17-1
[user@irus17-1 FAST@~]$ exit
[user@irus17 FAST@~]$
```

Для того чтобы узнать состояние очередей на кластере используется команды squeue

```
[user@irus17 FAST@~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            105328  standard  nl_.bat    user1 PD       0:00      4 (QOSMaxNodePerUserLimit)
            107333  standard amp_400.   user2  R       1:02      1 irus17-8
            107334  standard amp_400.   user2  R       1:02      1 irus17-9
            105301  standard kap2.bat    user1  R    8:56:00      4 irus17-[4-7]
            105321  standard nl2_.bat    user1  R 2-10:52:46      1 irus17-10
```

Основной вывод состоит из семи столбцов

- JOBID - уникальный номер задания
- PARTITION - очередь на кластере (всегда standard)
- NAME - название задания (можно задавать параметром --name через sbatch/salloc/srun; по умолчанию имя скрипта или команды, указанной при запуске задания)
- USER - имя пользователя, инициатора данного задания
- ST - состояние задания
  - PD - запуска отложен, ожидает ресурсов (PENDING)
  - R - исполняется (RUNNING)
  - другие состояния скорее носят вспомогательный характер, их можно посмотреть в справочном руководстве (man squeue)
- NODES - количество используемых узлов
- NODELIST(REASON) - либо имена используемых узлов, либо вероятная причина того, почему данное задание ожидает в очереди
  - Resources - задание ожидает, потому что нет свободных ресурсов
  - Priority - задание ожидает, потому что впереди в очереди есть более приоритетные задания
  - другие причины чаще всего означают виды ограничений, по которым данное задание сейчас не может быть запланировано на запуск

По команде squeue --start можно посмотреть планируемое время запуска задания, если таковое уже было определено.

По команду sprio можно посмотреть приоритет задания, которое находится в очереди и запланировано. Указываются балы по четырем критериям, из которых складывается приоритет:

- время нахождения в очереди (AGE): чем дольше ждет, тем выше приоритет;
- коэффициент справедливости (FAIRSHARE): чем больше использовал кластер за период планирования (неделя), тем приоритет ниже, чтобы быть справедливым по отношению к другим ожидающим ресурсы пользователям;
- размер задания (JOBSIZE): чем больше задание, тем ниже приоритет, чтобы давать возможность небольшим заданиям проходить при наличии свободных ресурсов.
- вид очереди (PARTITION): всегда постоянный с весовым коэффициентом 1000.

6.5 Отмена задания

Отмена задания осуществляется командой scancel и указанием номер отменяемого задания

```
[user@irus17 FAST@~]$ scancel 12345
```

6.6 Статистика обработанных заданий

Для получения статистики выполненных заданий можно использовать команды sacct. Основные параметры ее использования

- -S <дата-время> : выводит информацию о заданиях с этой даты
- -E <дата-время> : выводит информацию о заданиях по эту дату
- -o <format> : список столбцов вывода информации. Параметр --helpformat выводит весь список поддерживаемых имен столбцов. Столбы указывается через запятую без пробелов. После имени столбца может идти знак процента и число (например, «%20»), что означает использовать 20 символов для вывода информации в этом столбце. Процент используется для случаев, когда информации больше, чем предусмотрено стандартным размером столбца
- -u : имя пользователя, для которого выводит статистику

Пример

```
[user@irus17 FAST@~]$ sacct -u user-S 2017-07-01T00:00:00 -E 2017-07-05T23:59:59 -o JobId,JobName%10,State%20,Start%24,Elapsed
```

```
       JobID    JobName                State                    Start    Elapsed
------------ ---------- -------------------- ----------------------- ----------
58043              mdrun    CANCELLED by 5003      2017-07-04T12:11:07   00:00:05
58043.0            mdrun    CANCELLED by 5003      2017-07-04T12:11:07   00:00:05
58044              mdrun    CANCELLED by 5003      2017-07-04T12:12:01   00:00:02
58044.0            mdrun    CANCELLED by 5003      2017-07-04T12:12:01   00:00:02
58045          mdrun_mpi    CANCELLED by 5003      2017-07-04T12:12:58   00:00:05
58045.0        mdrun_mpi    CANCELLED by 5003      2017-07-04T12:12:58   00:00:05
58046          mdrun_mpi    CANCELLED by 5003      2017-07-04T12:13:19   00:00:03
58046.0        mdrun_mpi    CANCELLED by 5003      2017-07-04T12:13:19   00:00:03
58047          mdrun_mpi             COMPLETED      2017-07-04T12:13:50   00:00:09
58047.0        mdrun_mpi             COMPLETED      2017-07-04T12:13:50   00:00:09
58052          mdrun_mpi             COMPLETED      2017-07-04T12:17:21   00:01:19
58052.0        mdrun_mpi             COMPLETED      2017-07-04T12:17:21   00:01:19
58057          mdrun_mpi    CANCELLED by 5003      2017-07-04T12:20:28   00:00:15
58057.0        mdrun_mpi    CANCELLED by 5003      2017-07-04T12:20:28   00:00:15
58058          mdrun_mpi             COMPLETED      2017-07-04T12:20:46   00:01:28
58058.0        mdrun_mpi             COMPLETED      2017-07-04T12:20:46   00:01:28
58063          mdrun_mpi             COMPLETED      2017-07-04T12:24:11   00:13:51
58063.0        mdrun_mpi             COMPLETED      2017-07-04T12:24:11   00:13:51
```

7. Контактные данные администраторов

Адрес группы поддержки вычислительных ресурсов ЦКП:

690041, г. Владивосток, ул. Радио, д. 5, каб. 404

Институт автоматики и процессов управления ДВО РАН

ЦКП «Дальневосточный вычислительный ресурс»

Телефон: *+7(423)232-07-02*

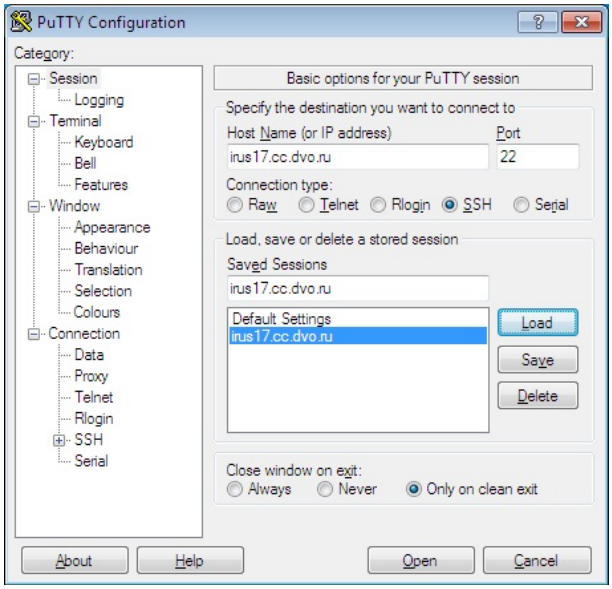
E-mail: cc@dvo.ru

Официальный сайт: https://cc.dvo.ru

Сайт базовой организации: http://www.iacp.dvo.ru/

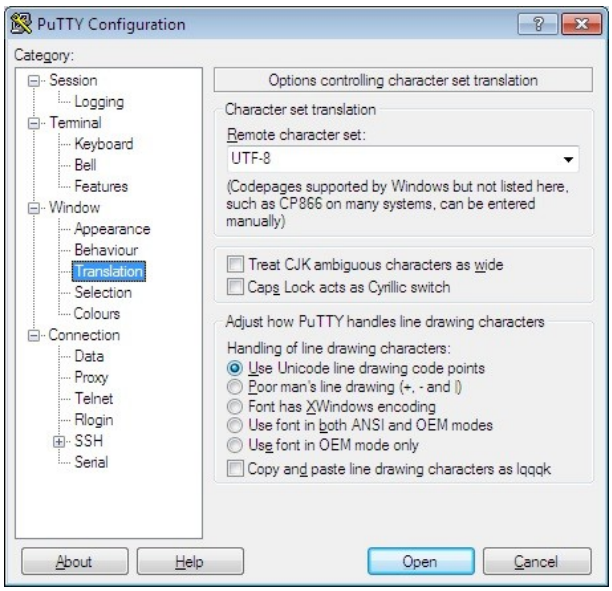
Приложение 1. Настройка PuTTY

1. Открыть Putty, в полях Host Name и Saved Session задать irus17.cc.dvo.ru



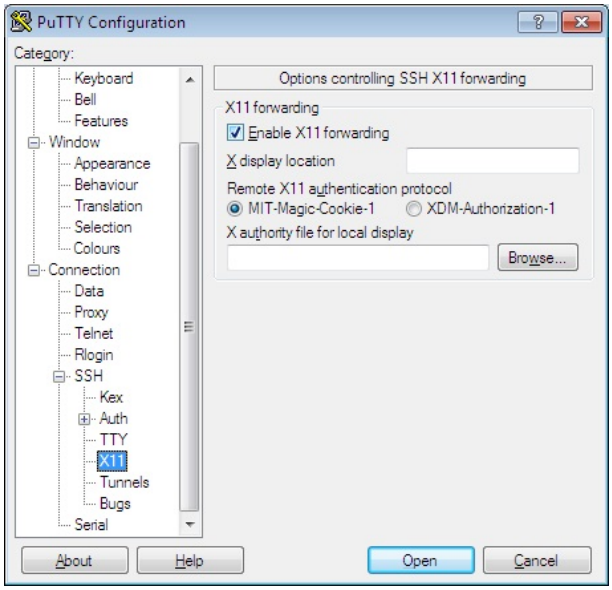
2. Нажать Save

3. Перейти в раздел настроек Translation и поставить кодировка UTF-8



4. Перейти в раздел настроек Connection → SSH → X11

5. Если необходима работа с X11-приложениями, то поставить галочку Enable X11 forwarding.,



6. Вернуться в основной раздел Session и еще раз нажать Save

7. Нажать Open для открытия сессии пользователя.

Приложение 2. Краткий справочник команд man

h - вызвать помощь по самому man

клавиши курсора - перемещение по тексту

/шаблон - поиск текста по шаблону

q - выход из man