Руководство пользователя

Оглавление

Политика безопасности ресурсов ЦОД

Использование вычислительных ресурсов ЦОД, автоматически означает Ваше согласие с нашей политикой безопасности. Внимательно прочитайте этот документ и следуйте его предписаниям.

Обратная связь

В случае возникновения любых проблем при работе на кластерах ЦОД, пожалуйста, напишите письмо по адресу «help.computing@nrcki.ru», указав в нём:

- ваше имя,
- подробное описание проблемы,
- название машины, которую вы использовали для входа на кластер (например, ui2.computing.kiae.ru),

Доступ к ресурсам

В настоящее время зарегистрированные пользователи работают на кластере HPC2, доступ с машины ui2.computing.kiae.ru.

Вход на головные машины кластеров осуществляется с использованием протокола SSH. Для соединения можно использовать любой SSH-клиент, поддерживающий протокол SSH версии 2. Например,

- Putty (Windows),
- SSH Secure Shell Client (Windows, также есть версия для UN*X, но OpenSSH лучше),
- OpenSSH (UN*X, входит в комплект практически всех современных UN*X-подобных операционных систем).

Для копирования файлов можно использовать следующие утилиты:

- scp (UN*X, входит в состав OpenSSH);
- WinSCP (Windows, графический интерфейс, две панели: локальная и удаленная машины),
- pscp.exe/psftp.exe (Windows, текстовой интерфейс),
- SFTP-клиент из пакета SSH Secure Shell Client.

Создание SSH-ключей

Для аутентификации на кластерах используются SSH-ключи. Это пара файлов, один из которых содержит то, что называется закрытым ключом, а второй содержит открытый ключ. Закрытый ключ обычно защищен паролем, его не показывают никому во избежание разных неприятностей. Открытый ключ может быть показан кому угодно; более того, его нужно поместить на сервер, чтобы получить к нему доступ. Зная открытый ключ, в-принципе, можно восстановить закрытый, но из-за сложности разложения чисел на простые множители на современных машинах это займет достаточно большое количество времени.

Имея только открытый ключ, можно проверить, есть ли у вас соответствующий ему закрытый ключ. Поскольку предполагается, что

закрытый ключ есть только у вас (именно поэтому его так важно никому не показывать), то при успешном окончании проверки можно утверждать, что это именно вы пытаетесь получить доступ к серверу.

Процесс создания SSH-ключей несколько отличается для разных операционных систем.

Создание SSH-ключей на Unix-системах

Предполагается, что в системе установлен пакет OpenSSH, содержащий утилиту ssh-keygen. Запускаем ее:

```
$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vasya/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your public key has been saved in /home/vasya/.ssh/id_rsa.pub.
The key fingerprint is:
22:f7:7b:96:c5:4b:b2:0d:2d:9f:5f:67:57:48:46:37 vasya@some.host.in.the.net
```

Если вы хотите сохранить ключи не в /home/vasya/.ssh/id_rsa, а в другом месте, необходимо ввести полный путь в ответ на приглашение «Enter file in which to save the key». Начинающему пользователю, не планирующему использовать различные SSH-ключи для доступа к различным ресурсам, лучше оставить значения по умолчанию.

После окончания работы ssh-keygen закрытый ключ будет находиться в файле /home/vasya/.ssh/id_rsa, а открытый – в файле /home/vasya/.ssh/id_rsa.pub. Последний файл необходимо переслать в службу регистрации пользователей ЦОД. Пожалуйста, не перепутайте.

Чтобы использовать нестандартное имя файла закрытого ключа, нужно, либо указывать его расположение каждый раз с помощью ключа «-i», либо добавить в файл ~/.ssh/config следующие строчки:

```
Host ui2.computing.kiae.ru
    IdentityFile ~/.ssh/id_rsa-hpc2
```

При использовании одного и того же открытого ключа для доступа к нескольким кластерам нужно в настройках указать имя узла, содержащее

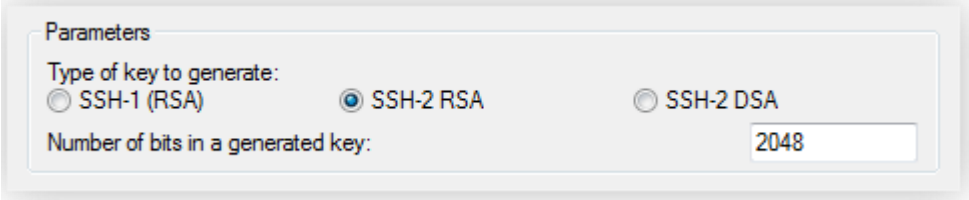
символ «*» - шаблон, который соответствует любому количеству любых символов:

```
Host ui*.computing.kiae.ru
    IdentityFile ~/.ssh/id_rsa-kiae
```

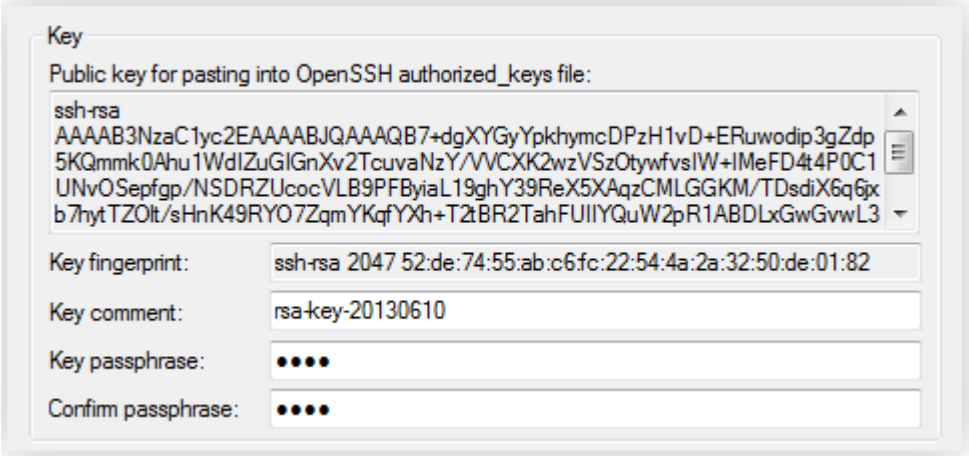
Создание SSH-ключей на Windows-системах

Putty

Для создания SSH-ключа необходимо скачать программу PuTTYgen с сайта Putty, запустить ее и выбрать в качестве типа ключа «SSH-2 RSA», а его размер - равным 2048 бит:

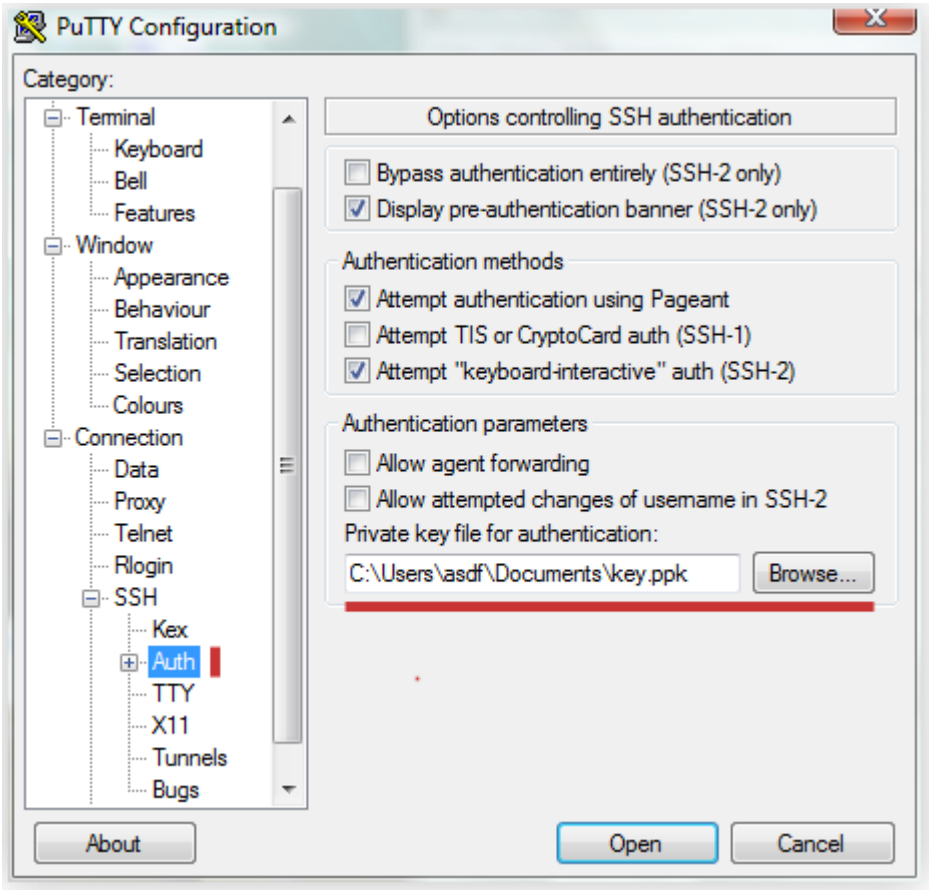


После нажатия кнопки «Generate» PuTTYgen создает ключ и предлагает ввести для него пароль:



Пароль нужно выбирать такой, чтобы враг его не раскрыл, а владелец - не забыл. Также можно ввести комментарий (любой).

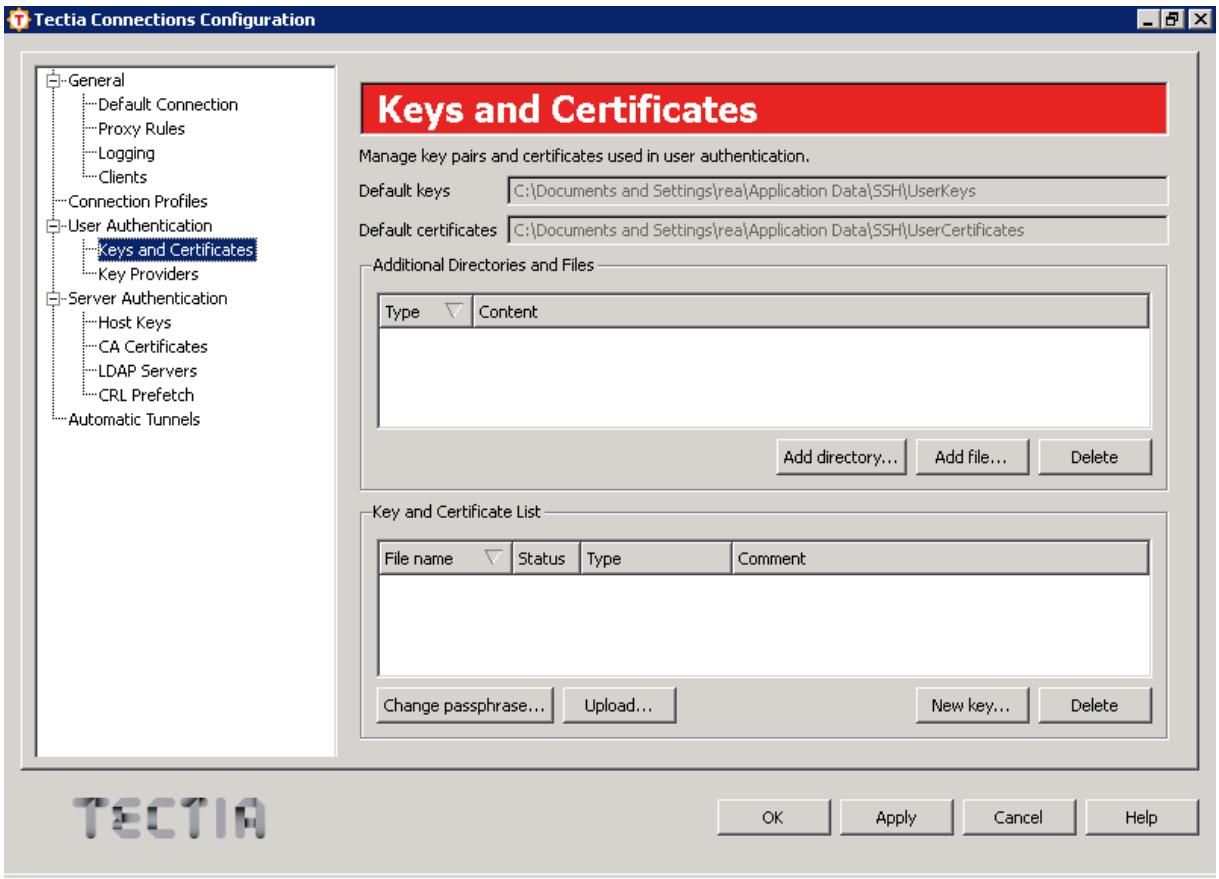
Сохраняем private key (закрытый ключ) в файл mykey.ppk, а public key (открытый ключ) - в файл id_rsa.pub. Файл id_rsa.pub необходимо переслать в службу регистрации пользователей ЦОД, а местоположение закрытого ключа ввести в соответствующее поле программы Putty:



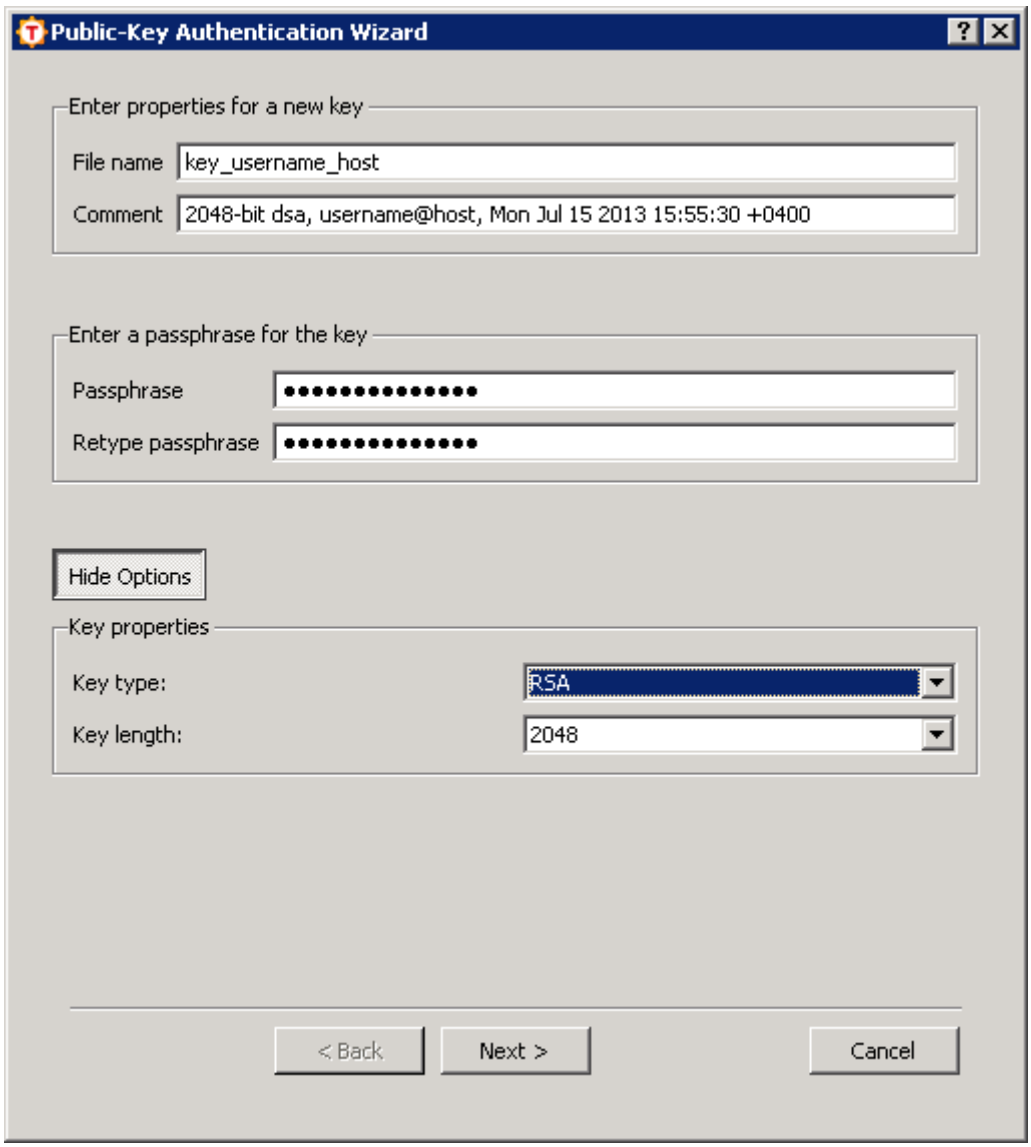
Tectia SSH Client (ssh.com)

Tectia SSH Client - это преемник Windows SSH Client от ssh.com. Некоторые пользователи любят работать с этим клиентом, поэтому он включен в данное руководство.

Для создания SSH-ключей в меню «Edit» необходимо выбрать пункт «Tectia Connections…», а в узле дерева слева «User Authentication» - «Keys and Certificates»:

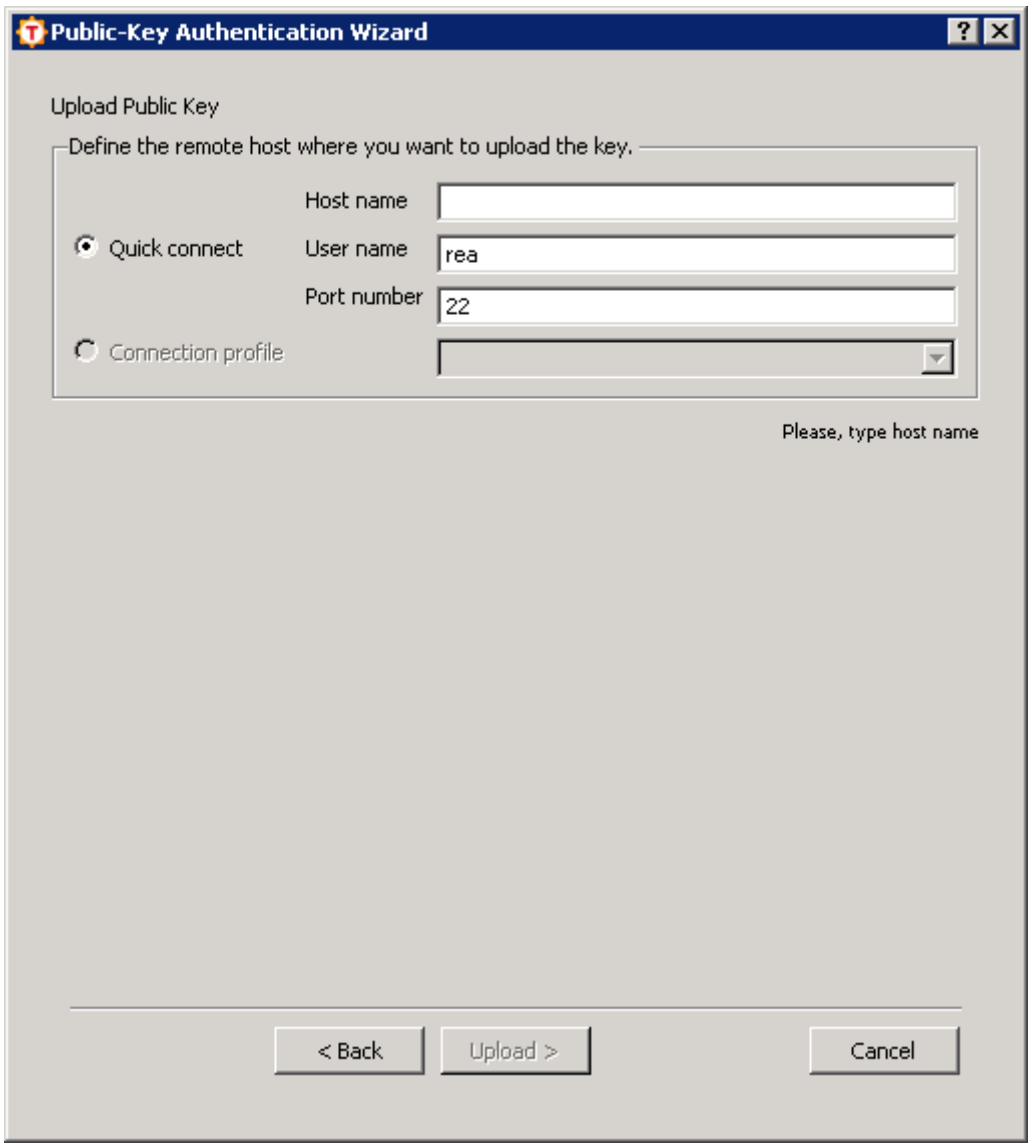


После нажатия кнопки «New key…» появляется следующее окно, в котором необходимо ввести параметры создаваемого ключа:

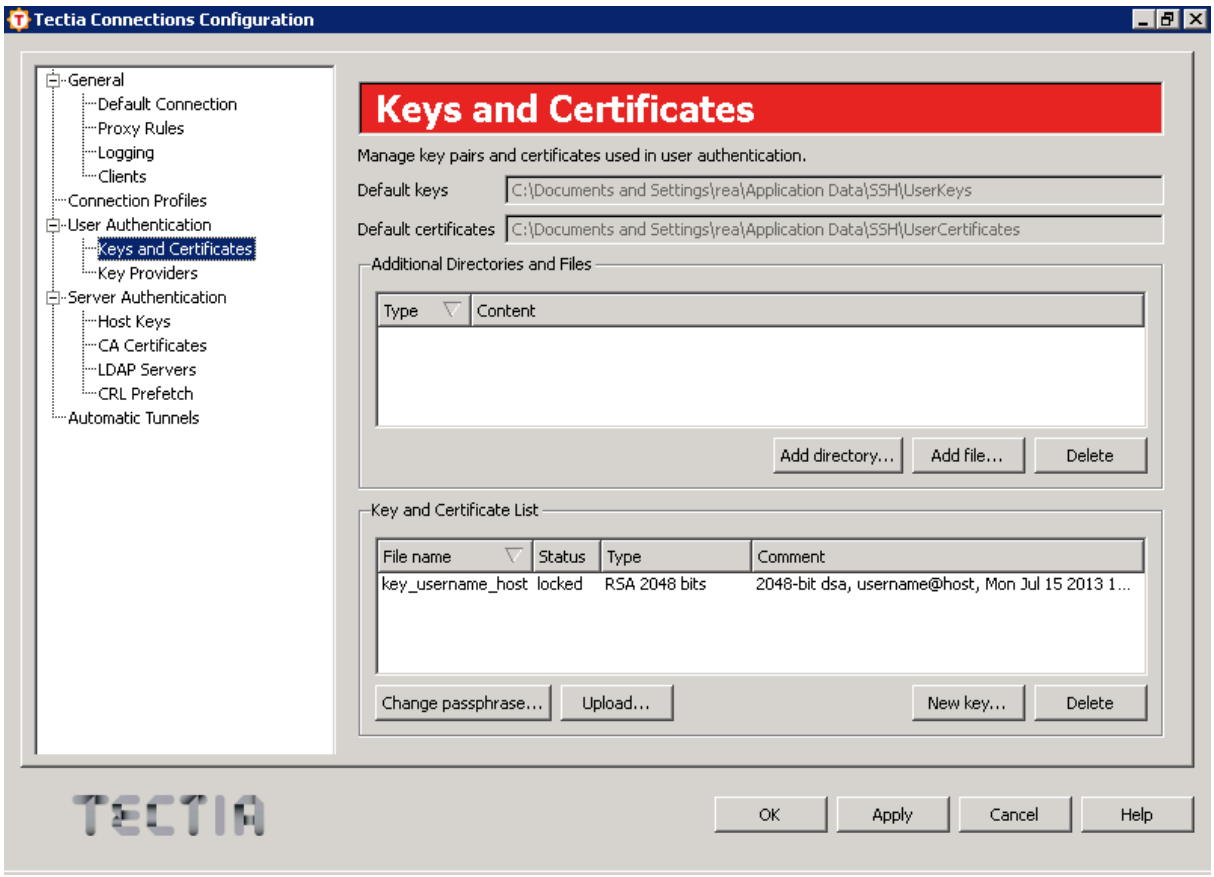


- «File name» - естественно, имя файла открытого и закрытого ключей.
- В поле «Comment» можно ввести комментарий, например, для чего предназначен создаваемый ключ.
- Поля «Passphrase» и «Retype passphrase» предназначены для двукратного ввода пароля закрытого ключа.
- После нажатия кнопки «Advanced Options», нужно выбрать тип ключа «RSA» и количество бит - не менее 2048.

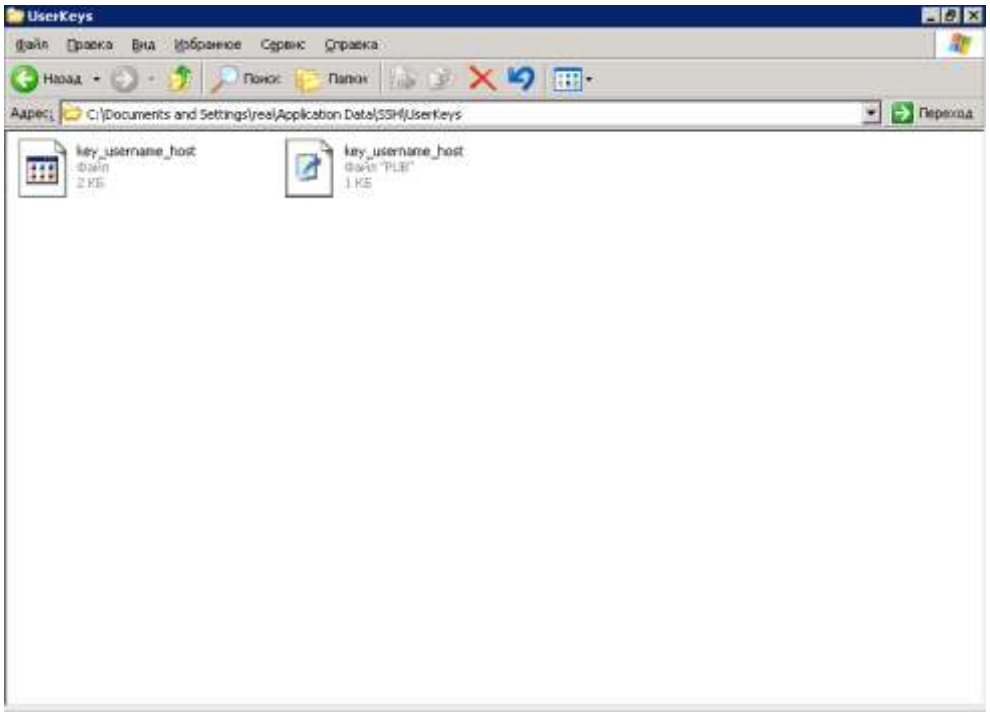
Нажатие кнопки «Next» инициирует создание ключевой пары, что может занять некоторое время. После чего появится окно, предлагающее загрузить ключ на сервер:



Ничего никуда загружать не нужно, нажимаем «Cancel» и снова приходим к первоначальному окну, в котором будут отражены все созданные публичные ключи.



Поле «Default keys» показывает, где сохраняются открытый и закрытый ключ. Имя файла закрытого ключа не имеет расширения, а имя файла открытого ключа имеет расширение «.pub»:

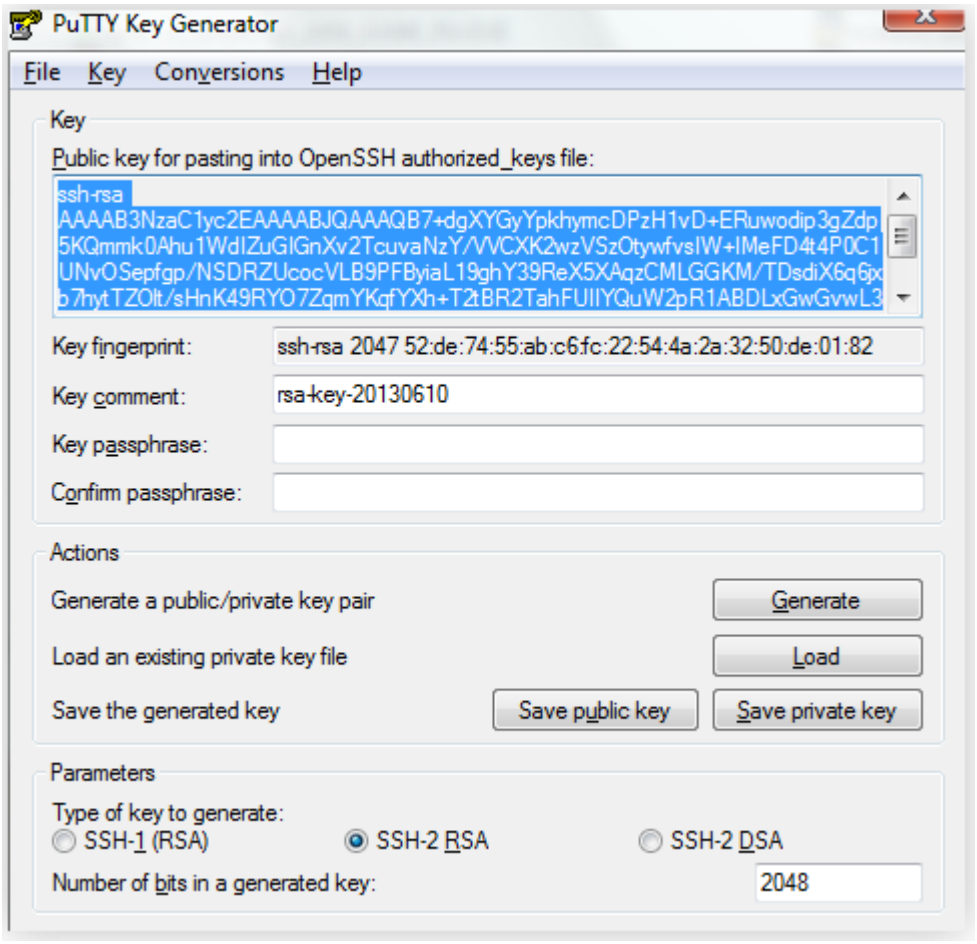


SSH-ключи, сгенерированные в разных операционных системах, имеют различные форматы. Поэтому использование ключей, созданных в одной системе, для доступа к ресурсам того же сервера из другой операционной системы требует преобразования форматов. Иначе система не будет распознавать закрытый ключ.

Экспорт ключей из Putty/Tectia в формат OpenSSH

Экспорт закрытого ключа, полученного при помощи программ Putty, или Tectia SSH Client, из Windows-системы в Unix осуществляется утилитой Putty key generator.

Шаг номер 1: загрузка существующего закрытого ключа в Putty key generator (меню «File», пункт «Load private key»):



Шаг номер 2: в меню «Conversions» выбрать пункт «Export OpenSSH key», ввести имя файла и выбрать каталог для хранения закрытого ключа в

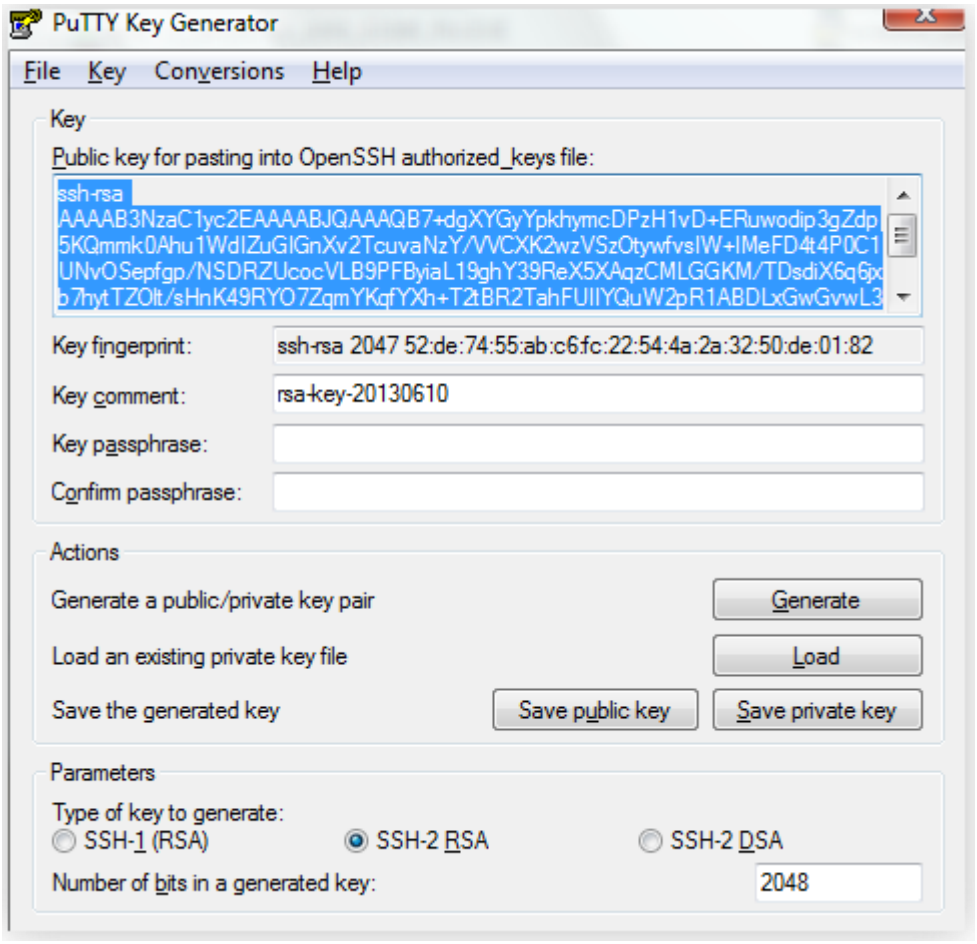
новом формате. Пароль закрытого ключа не изменяется после преобразования в другой формат.

Шаг номер 3: перенос закрытого ключа на Unix-машину. С точки зрения безопасности лучше использовать какой-нибудь offline-носитель, а не пересылать ключ через сеть или электронную почту.

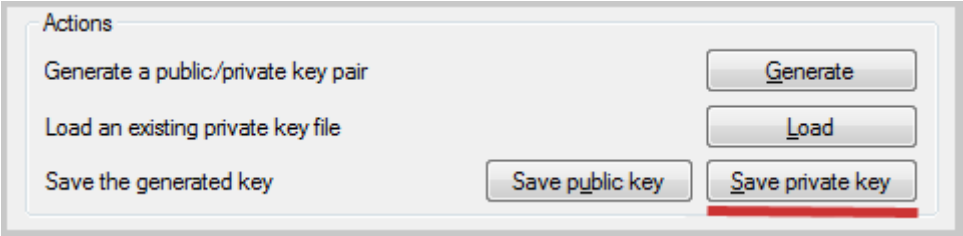
Экспорт ключей из OpenSSH в формат Putty/Tectia

Экспорт закрытого ключа из Unix-системы, использующей OpenSSH, в Windows-систему, где предполагается работать с помощью программ Putty или Tectia SSH Client, также осуществляется утилитой Putty key generator.

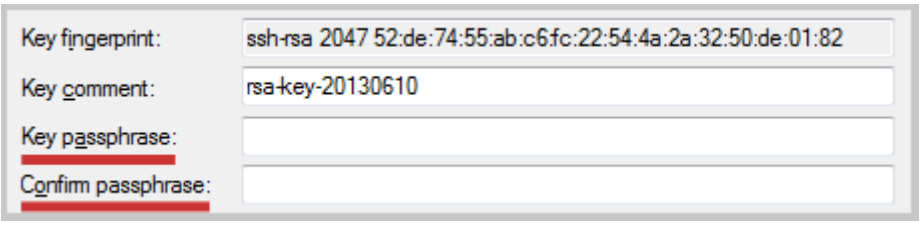
Шаг номер 1: в меню «Conversions» выбрать пункт «Import key», указать файл с ключом в формате OpenSSH, в открывшемся окне ввести пароль данного закрытого ключа. В результате получаем:



Шаг номер 2: нажать окне кнопку «Save private key»,



выбрать имя файла и каталог, в котором будет храниться файл закрытого ключа в новом формате. Пароль ключа после в новом формате не изменяется, но существует возможность его изменить в процессе преобразования. Для этого перед сохранением ключа в новом формате необходимо ввести новый пароль в поля «Key passphrase» и «Confirm passphrase»:



Типичные проблемы, возникающие при работе с SSH-ключами

При переносе SSH-ключей между различными операционными системами иногда возникает необходимость определить, в каком формате записаны ключи. Остановимся на различиях в форматах ключей, используемых в разных операционных системах.

Так выглядит закрытый ключ для Unix-систем:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,F45627B370E24DB1

XD4u/JN7snja3Wl1Dj7eoIGKiAx8IOALvLbmUTik4liCMxk3OPW1eOTQxAV7CJom
PMfH5kOMAber45D4yiQzGYGc/mxStfNjtm8V7j6N4acb6E4BKRUGmv7ZkLrw9csr
PpZzSQ5wTYiXRzJE55r5Q+CWd2V5qVJypAW2Q8pUMGF6A+DSxno1LSc8TwQ43o8G
FTGw8bCEgnfhXmGGLm6lirbXOQIJRCqbszBlPGZ4uNNaLAkbs7O85dqjKFsTMXLv
XCloDn+8wFr8eg7mZAkAjs0jz9l0PBDAYxivzrw97GRmemHynqsDSaE4F6pxaed9
lhxLwtK0VZH1ofEZUdT64lHJUX4oow9t2r7ajrItlnGBo7redY4yTaLKArxSOdVG
AKOO0PyAv4J2Mm0F8bNwYecUskpu+efr4n2343jJKmXONc+KxfZjdDBFsVFkmpv3
p4AyryKisiT/BbgAdLYXomVKeMSBQe6eDUSQtIabirOBsk63aiJAy5j5yzqzudJ8
IzmCuNB7p4SkmqiPoqW7vPR5G2PlK4G790vszcYr221w5QE3keVB/iRszyJrQHl8
k9LzX3oejumGknLubihJqelWBDP54tGlL88YB9uPQ4jSUFOv8+M68ebBONVh1eB9
uUe0C3FwxUHm53i7v8DnDFki8onnk6mUxa8pgIJJBUMdDnie0CGh1QAT6NiLE2tT
```

```
mnBIG7pHxU3HbnUdA+yDbvvfvkoP6Lc5XWCQsn8SNjDv2gPkNrYRbxaRjgOykTiQ
ZHEEU+JYwzyOtw6tbuOOUfcH4BqKhVc0YwZrDDKHOXAo0PgGcuxoE8EAKZptV3lz
XyFZjtfw5KvulRmwjuMydu0Alg5qo8cpyCSFCyxRQlWjyiPwOI3w7ixLpZFpQGMQ
bmVrgLKB/XdgPnmXj7K/6KD2YU2FxZjCFdbGdDUY6E/cBAHD/7sHjGV6CXJ72ZkO
oGWhLgkRk/Dy9doysm6DwCiLS7K/cddUkZcFKvxzBdmOaTt+jlB2tXKDvRAJIwrm
GiEx2LlKjbbgoTrV+rjuFFgVhsHualxP52NsvujQZVpeFtomZ/amk3ceOMTFTkab
QJcb/zOWjG+PrtiQ6BR/Te0kl44S2L3AR5AOCVD13k1nEOZ1yHyCtti04xM7JavP
Jy+RSUmIt7hSD9A0e4nHHXEhPZnGgG8ekVrR6FEQ+0FbvYLpv05Ir+igQSMftZwA
YdBjA16KeJL4jKAOWzVe5tdA0BQcJvPjzPK97N6HkCrbcmSy7mQAsXCZ7BcInwwa
UD8zOeH8Ii8atzy+YM+bQRoRfQLkzpJ3pVe68ZwqKFHl+YQNlh0sCJ3slihLKKVR
mUXutpa2c385Yb9djLifKaSdPYG3rutmdx7HY1JzYvvkIau+ixiO1H6dETI8tLZC
4l1FTSisGt4LZyH3WgPpQzhiWs0KX9yIQ/0lqRhgEL8zqqDm3jo/jIQdFMVfHBlI
YSKjoySUN3GXk8MfBsxxgbJRNwfcdiuB5qcsAxYkNVJgczHScEoM5NoatlHVlbyV
Ymu8j7BqNZ5a1W/YYadEV2prdQeAUOTX8yGVq14MU/5X6uTZyCj9fQ==
-----END RSA PRIVATE KEY-----
```

Детали могут отличаться, но первая и последняя строки почти всегда имеют вид

```
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----
```

либо

```
-----BEGIN DSA PRIVATE KEY-----
-----END DSA PRIVATE KEY-----
```

Так выглядит закрытый ключ для Putty:

```
PuTTY-User-Key-File-2: ssh-rsa
Encryption: aes256-cbc
Comment: rsa-key-20130610
Public-Lines: 6
AAAAB3NzaC1yc2EAAAABJQAAAQB7+dgXYGyYpkhymcDPzH1vD+ERuwodip3gZdp5
KQmmk0Ahu1WdIZuGIGnXv2TcuvaNzY/VVCXK2wzVSzOtywfvsIW+IMeFD4t4P0C1
UNvOSepfgp/NSDRZUcocVLB9PFByiaL19ghY39ReX5XAqzCMLGGKM/TDsdiX6q6j
xb7hytTZOlt/sHnK49RYO7ZqmYKqfYXh+T2tBR2TahFUIIYQuW2pR1ABDLxGwGvw
L3T2laIT5Lt3//6ioCT0jlIkBuVii1hlm2quiyqogDs1U1ihOGL0wKNvhpds4Hfd
wyRCjU1WJkZGGrLkdtvlg4AQhM7J8FVrYhTNZe90GJZXjnJP
Private-Lines: 14
Xp4pdSAaE8Ftj9Va68OgwyTtO1V75m7cLmSHJSxzVaQCpw8zzn2gPeFnl/7fuusN
Q1cZ8uDdpzI5E2+iEtM5hSn4SoOfH5eyhSa+FYObxj2IN+ylwS4vIbPmj7pY4Mom
BGvxPdgZzKSITblxiE6YSFCNq4CmTIjtg/OXU5/slw7oifdEPcvPDudaGtztmv4a
zSMPO9tvgrk1C5hzY5FfDoKpZHK94mgyjypHt7NEAbEls7GQCn7qTsng2Au2xeu/
l5ZDmb52zu/iLYb2JALnz+h5qyE2XFHsVjwyY+rzxstEWgkgCZdCrbOexHrgIlyc
sKr7H3c3pJ7ZsVsXFvoWLVRcFhq/8Sa7rTr7ONkaUBPUiF00fdp2UiC3bI5U9FlQ
MRhXKEfogEdJDZdWvGibHyX2on8PkJzDZCuQaG/K6pm92VHhv8AYMs4ADfrUVg6k
Y7A5q+sDtXM1QAiL/Qct1Hr8uAp8sLfKYsnBSc1teTHdIjugEjfOn8zMR96vgFEi
bXWcA8o4WROMJ42f9Dy3hTsg7kf4ZcGmY8ua3deYABu3KuAttUNTshAF5qfPGjEk
gR53PoPgLzA1IeYIgX9w9INHQgfolsWguuTb5i1UdXKnk7SwXxmyl1o9F3DSRGu9
pl7bKLQGjrDcaLyxavUyZoHb1zwiDoGBJ4pefC1e1LKJ6OUiWXCWcUsTIeuj9q+L
eEr4RWr59A/11jGvbrTi2Qjxp71aqHmQjsiXonT/bl6Xydb3zS8e4AmJJifxu179
```

```
EgfXBPePYarjX6PPivt6uURnRDyTVU6jXJl4ddIudtpUpY14M4BekA4MZ9m2PMsx
S5syWlWFvfPIoHMLpdDWV/D2kimvamcEbbCxK0uI9VlVHNmTotI49OYt0XaHrZGS
Private-MAC: 464d517a6c7b61184a4f7f1ec2ad4f33c1b15294
```

Характерными особенностями данного формата является первая строка вида

```
PuTTY-User-Key-File-2: ssh-rsa
```

вслед за которой довольно часто следуют строки вида

```
Encryption: aes256-cbc
Comment: rsa-key-20130610
```

Пароли и SSH-ключи

Пароли и закрытые части SSH-ключей нельзя показывать и передавать кому бы то ни было, даже службе поддержки, или администратору, с которым вы переписываетесь, или разговариваете по телефону. Любой, кто спрашивает о вашем пароле - это человек, который пытается методами социальной инженерии узнать секретные сведения и получить доступ на кластер ЦОД. Исключений не бывает, пожалуйста, запомните это.

Нельзя передавать пароль и/или SSH-ключ коллегам и близким. Если кто-то хочет получить доступ к кластеру ЦОД, он должен заполнить заявку, и она обязательно будет рассмотрена.

При обнаружении факта передачи пароля или SSH-ключа другому лицу доступ виновному в этом на ресурсы ЦОД будет закрыт до момента выяснения всех обстоятельств, а, возможно, и навсегда.

Установленное ПО

Компиляторы

- Intel C/C++ Compiler 14.0.2: компилятор запускается командой `icc`. Для компиляции MPI-приложений нужно использовать команду `mpicc`.

- Intel Fortran Compiler 14.0.2: компилятор запускается командой `ifort`. Для компиляции MPI-приложений нужно использовать команды `mpif77` и `mpif90`.

Библиотеки MPI

- Platform MPI 9: может быть использована как с программами, скомпилированными с помощью Intel C/C++ и Intel Fortran, так и с программами, скомпилированными с помощью GNU C/C++.

Доступные файловые системы

На суперкомпьютерных ресурсах используется два различных типа файловых систем, NFS и Lustre.

Домашняя файловая система

Для поддержки домашних каталогов пользователей (тех каталогов, в которые пользователь попадает при входе на login-узлы) используется файловая система NFS. Эти каталоги доступны только с login-узлов, серверы вычислительного поля их не видят. Все login-узлы всех кластеров работают с одним общим файловым пространством на NFS. Домашние каталоги на NFS не подвергаются периодической принудительной очистке, но для сохранности данных пользователям рекомендуется периодически копировать на свои рабочие машины и хранилища данные, исходные коды и сценарии запуска, поскольку в настоящее время ЦОД не обладает механизмом создания резервных копий.

Существуют пользовательские каталоги, доступные пользователю по имени /home/users/$USERNAME, и групповые каталоги, доступные как /home/groups/gABCD для чтения и записи всем членам указанной группы. В файловой системе NFS установлены групповые квоты для полного пространства, которое занимают как индивидуальные пользовательские каталоги, так и групповой каталог. Квоты назначаются в соответствии с

данными, указанными в заявках на регистрацию группы, они могут быть расширены (разумным образом), если это необходимо. По вопросам увеличения квот обращайтесь в службу поддержки help.computing@nrcki.ru.

Технически NFS располагается на нескольких независимых серверах, при этом домашний каталог каждого из пользователей находится на определённом сервере. Это означает, что при проблемах с одним NFS-сервером пострадают все пользователи с домашними каталогами, располагающимися на нём, но другие пользователи будут иметь возможность продолжать работу.

Параллельная файловая система

У каждого из суперкомпьютеров есть локальная параллельная файловая система, основанная на Lustre. Она доступна как с login-узла этого суперкомпьютера, так и с узлов его вычислительного поля и должна использоваться для хранения временных данных расчётов и программного обеспечения. Каждый кластер имеет доступ только к своей параллельной файловой системе.

Организация каталогов на Lustre сходна с домашней файловой системой:

- есть каталоги для каждого пользователя, они доступны ему по имени `/s/ls2/users/$USERNAME`,
- существуют групповые каталоги, доступные как `/s/ls2/groups/gABCD`

Эти каталоги предназначены для хранения временных данных в процессе вычислений и переноса входных/выходных данных с/на домашнюю файловую систему. (В не столь отдалённом будущем) они будут периодически очищаться от старых файлов.

Для более длительного хранения объектов (например, программного обеспечения, вспомогательных данных, которые часто используются для многих вычислительных задач, и т.п.) предусмотрены каталоги с другими именами:

- каталоги для каждого пользователя называются `/s/ls2/u-sw/$USERNAME`,
- каталоги для группы называются `/s/ls2/g-sw/gABCD`.

Квоты на дисковое пространство на файловой системе Lustre не установлены. Поэтому задача организации разумного совместного использования доступного объема хранилища целиком ложится на пользователей. Неиспользуемые данные должны стираться, или переноситься на NFS.

Копирование данных между файловыми системами

Для копирования данных между файловыми системама NFS и Lustre вручную рекомендуется использовать командамиы cp, rsync и др.

Файловая система Lustre является параллельной файловой системой и предназначена для хранения данных, к которым обращается значительная часть узлов вычислительного поля. В результате при ее использовании не по назначению пользователь будет испытывать сложности при работе с файлами. Будут долго открываться и сохраняться файлы в редакторе, долго будет работать команда ls и т.п.

Гораздо удобнее использовать Lustre и домашнюю файловую систему по их назначению.

Настройка программного окружения

Каждый пользователь может настроить программное окружение по своему желанию в соответствии с используемым программным обеспечением, используя несложную систему под названием modules. Например, чтобы воспользоваться компиляторами от Intel, нужно выполнить команду:

```
module load intel-compilers
```

а так настраивается компиляция параллельной задачи с использованием библиотеки MPI:

```
module load mpi
```

Всё это можно сделать одной командой:

```
module load intel-compilers mpi
```

Следующая команда позволяет узнать, какие пакеты программ имеются в системе

```
module avail
```

а команда

```
module list
```

выдает список уже используемых в данном сеансе модулей.

Более подробную информацию о системе «modules» можно получить, запустив команду man module.

Для настройки пользовательского окружения на головных машинах достаточно поместить в файл ~/.bash_profile следующие строки:

```
module load mpi intel-compilers
```

Эта строка говорит о том, что пользователь будет использовать библиотеки MPI (mpi) и компиляторы Intel C/C++ и Intel Fortran (intel-compilers).

В сценариях запуска задач нужно указывать те же строки, что и в ~/.bash_profile. В соответствующем разделе будут приведены подробные примеры.

Запуск задач

Запуск задач осуществляется командой sbatch. Подробную документацию по этой команде можно получить, выполнив команду man sbatch, ниже будут рассмотрены только базовые функции.

Запуск простого сценария

Важно: batch-система умеет запускать на выполнение только сценарии. Нельзя с её помощью запустить исполняемый файл.

В качестве примера рассмотрим запуск на выполнение сценария test.sh, находящегося в текущей директории и содержащего следующие строчки:

```
#!/bin/sh
#SBATCH -D /s/ls2/users/eygene
#SBATCH -o %j.out
#SBATCH -e %j.err
#SBATCH -t 01:00:00
#SBATCH -p hpc2-16g-3d
hostname
df
date
sleep 10
date
```

После создания сценарий нужно сделать исполняемым при помощи команды:

```
chmod +x test.sh
```

Для запуска сценария на выполнение используем команду sbatch test.sh;

```
$ sbatch test.sh
sbatch: Submitted batch job 19
```

Поставив задачу в очередь, команда сообщила её идентификатор. В данном случае - 19. Пользуясь этим идентификатором, можно, например, узнать текущий статус задачи:

```
$ squeue -j 19
  JOBID PARTITION    NAME     USER  ST       TIME  NODES NODELIST(REASON)
     19 hpc2-16g-3d  test.sh   eygene  PD       0:00      2 (JobHeld)
```

В данном случае задача ещё не была запущена. Состояние запущенной задачи выглядит так:

```
  JOBID PARTITION        NAME     USER  ST       TIME  NODES NODELIST(REASON)
     19 hpc2-16g-3d   test.sh   eygene   R       0:01      2 n[29-30]
```

Значение различных полей таблицы интуитивно понятны.

Командой scancel можно удалить задачу из очереди: scancel 19.

После успешного завершения задачи в каталоге /s/ls2/users/eygene, из которого задача запускалась, образуются файлы 19.out и 19.err. В них записано содержимое потоков стандартного вывода и стандартной ошибки

выполненной программы. Файлы сохраняются в каталоге /s/ls2/users/eygene, поскольку в сценарии содержится строка:

```
#SBATCH -D /s/ls2/users/eygene
```

которая заставляет планировщик при запуске задания переходить в указанный каталог. Этот каталог должен существовать на параллельной файловой системе, поскольку только её видят рабочие узлы, на которых выполняется задание (см. выше).

Название выходных файлов тоже определено в сценарии соответствующими строками:

```
#SBATCH -o %j.out
#SBATCH -e %j.err
```

В этих директивах «%j» заменяется идентификатором задачи.

Директива

```
#SBATCH -t 01:00:00
```

определяет максимальное время выполнения задачи (в данном случае - один час). Если возможно приблизительно оценить верхнюю границу времени выполнения задачи, ее необходимо указать, поскольку это позволит планировщику действовать более разумно. При этом необходимо понимать, что ваша задача будет «убита», если время ее выполнения превысит указанное. Поэтому лучше ставить время с небольшим запасом.

Директива

```
#SBATCH -p hpc2-16g-3d
```

Указывает имя очереди («hpc2-16g-3d»), в которую будет помещена задача. Параметры очередей подробно описаны в разделе «Список очередей и политика их использования»

Вышеприведенные строчки эквивалентны передаче утилите sbatch параметров командной строки «-o . -e .». Это справедливо для любых параметров команды sbatch.

Запуск одиночной задачи

Кроме запуска сценариев пользователям часто приходится запускать уже откомпилированные программы, подавать им на вход файлы с данными для расчёта и сохранять результаты в других файлах.

Как правило, аргументы передают программе из сценария, а вопрос копирования файлов на машину (с машины), где задача будет выполняться, легко решается, поскольку все узлы кластера видят единое файловое пространство разделяемой файловой системы. Задачу можно запускать из любого каталога на головном узле, а приведенный ниже сценарий позаботится о том, чтобы задача «увидела» все файлы в этом каталоге:

```
#!/bin/sh
#SBATCH -D /s/ls2/users/eygene/my-precious-task
#SBATCH -o %j.out
#SBATCH -e %j.err
#SBATCH -t 01:00:00
#SBATCH -p hpc2-16g-3d

module load intel-compilers

`pwd`/program mytask.in | tee mytask.log."$SLURM_JOBID"
```

Запускаемая программа называется “program”, она принимает на вход один аргумент, имя входного файла. В нашем случае - это “mytask.in”.

Конструкция “| tee mytask.log."$SLURM_JOBID"” применяется для перенаправления вывода программы в файл с именем “mytask.log.идентификатор_задачи”. Для программ, выводящих результаты своей работы на экран, это удобно, поскольку можно контролировать работу программы просматривая содержимое указанного файла. Если программа ничего не выводит на экран, эту конструкцию можно опустить, оставив только “`pwd`/program mytask.in”.

Запуск параллельной задачи

Материал этого раздела служит дополнением к предыдущему.

Как было указано выше, в сценарии запуска задачи окружение должно настраиваться точно так же, как и как в файле ~/.bash_profile. Пусть используются следующие директивы команды module load: mpi и intel-

compilers. Соответственно, первым отличием сценария запуска MPI-задачи от сценария для одиночной задачи является наличие строк

```
module load mpi intel-compilers
```

Строго говоря, отличие состоит только в загрузке модуля «mpi».

Следующим отличием является добавление одной строки вида «#SBATCH …», говорящей о том, что задаче нужно для работы 100 ядер:

```
#SBATCH -n 100
```

Последней модификацией в сценарии является добавление слова “mpisub” в начало команды запуска задачи.

В результате сценарий запуска параллельной задачи имеет вид:

```
#!/bin/sh
#SBATCH -D /s/ls2/users/eygene/my-precious-task
#SBATCH -n 100
#SBATCH -o %j.out
#SBATCH -e %j.err
#SBATCH -t 01:00:00
#SBATCH -p hpc2-16g-3d

module load mpi intel-compilers

$MPIRUN `pwd`/program mytask.in | tee mytask.log."$SLURM_JOBID"
```

Список очередей и политика их использования

Очереди поделены на классы. Очереди одного класса различаются по максимальному времени выполнения задачи и общему приоритету задач, но используют одни и те же физические ресурсы. Чем больше максимальное время выполнения задачи в очереди, тем меньше приоритет задач в этой очереди, при прочих равных условиях. Такой поход заставляет пользователей помещать короткие задачи в подходящие очереди, это позволяет планировщику лучше планировать ресурсы.

В настоящее время существует два класса очередей для локальных пользователей:

- «debug»:

    - класс представлен одной отладочной очередью для параллельных и последовательных задач, «hpc2-debug-10m»;
    - очередь состоит из одного узла с двумя четырехядерными процессорами Intel Xeon E5450 с тактовой частотой 3.00 ГГц и 16 Гбайт оперативной памяти;
    - все пользователи могут запускать задачи в эту очередь;
    - пользователь может поставить в эту очередь не более одной задачи;
- «hpc2-16g»:
    - класс очередей для параллельных и однопроцессорных задач;
    - очереди состоят из одинаковых счётных узлов с двумя четырехядерными процессорами Intel Xeon E5450 с тактовой частотой 3.00 ГГц и 16 Гбайт оперативной памяти;
    - задачи разных пользователей получают различный приоритет, который влияет на время ожидания задач в очереди;

В классе debug только одна очередь, debug-10m, с пределом времени счета в 10 минут;

В классе hpc2-16g есть следующие очереди:

- hpc2-16g-3d: очередь для задач, длящихся не более трех дней;
- hpc2-16g-1w: очередь для задач, длящихся не более недели;
- hpc2-16g-1m: очередь для задач, длящихся не более месяца (31 день); доступ в эту очередь открывается только по договоренности.

При необходимости использования очередей с ограниченным доступом, обращайтесь в службу поддержки пользователей help.computing@nrcki.ru.

Общие ограничения ресурсов в очередях

Во всех очередях действуют ограничения на размер виртуальной памяти для одного процесса. При попытке распределения памяти в сумме

более установленного предела, системные функции распределения памяти возвращают нулевой указатель. Дальнейших ход событий зависит от задачи, которую принудительно не останавливают. Возможны следующие варианты:

- задача завершается сама, если не может пережить отсутствия нужного количества памяти
- задача продолжает считать, если она без этой памяти сможет обойтись
- либо выпадает в core, если в программе не предусмотрена проверка успешности выделения памяти.

Текущие ограничения на виртуальную память: 1,945,600 Кбайт виртуальной памяти на процесс.

Образцы цитирования ЦОД для публикаций

При публикации результатов, полученных с использованием ресурсов ЦОД, пользователи должны в своих работах размещать ссылки на Центр обработки данных в соответствии с приведенными ниже примерами:

```
Результаты работы были получены с использованием вычислительных ресурсов
ЦОД НИЦ «Курчатовский институт» (http://computing.nrcki.ru/).
The results of the work were obtained using computational resources of
MCC NRC “Kurchatov Institute” (http://computing.nrcki.ru/).
```

Иногда задаваемые вопросы

Как узнать, на каких узлах выполнялась программа

Список узлов содержится в переменной окружения SLURM_JOB_NODELIST. Значением этой переменной может быть, например, «n[14,26-29]». Это означает, что задача выполнялась на узлах n14, n26, n27, n28 и n29.

Простейшим способом сохранить эти данные в выводе задачи будет добавление следующей строчки в сценарий запуска задачи:

```
echo "CPU list: $SLURM_JOB_NODELIST"
```

Строчку можно добавить, например, после директив «module load».

Каким образом распределяются физические машины для каждой задачи

Чтобы различные пользовательские задачи не конкурировали из-за оперативной памяти, задачам выделяются разные физические машины. Например, запросив 20 ядер, задача получит 3 физических узла, на которых ей будет выделено 8+8+4 ядер (если каждый узел оснащен восемью процессорными ядрами), но считать на этих узлах до своего окончания будет только эта задача. Из этого следует:

1. Количество процессов лучше выбрать кратным количеству ядер на узлах.
2. Возможна ситуация, при которой количество занятых всеми задачами ядер меньше полного количества ядер в кластере, но свободных физических машин нет, и вновь запускаемые задачи встают в очередь.

Как получить образ памяти (core dump) для программ, скомпилированных с помощью Intel Fortran.

Образ памяти иногда необходим для анализа ошибок в программе. Для всех ошибок «severe» в Intel Fortran образ памяти по умолчанию на диск не сбрасывается. Для получения образа необходимо установить значение переменной окружения decfort_dump_flag в «y»:

```
export decfort_dump_flag=y
```

При необходимости получения дополнительной информации об именах функций, номерах строк в исходных файлах и тому подобном нужно скомпилировать отладочный образ программы. Для этого необходимо указать при компиляции и компоновке флаг '-g', оповещающий инструментарий о том, что в исполняемый файл требуется включить отладочную информацию.

Отладочный образ для Intel Fortran позволяет при возникновении системных ошибок получить на экране информацию о последовательности вызова процедур, в которой будут присутствовать имена переменных, процедур и исходных файлов.

У меня не работает самая простая MPI-программа на Fortran. Спасите!

Есть очень простая программа, которая выглядит так:

```
      program mpi_probe
       include 'mpif.h'
       integer :: irank,isize,ierr
       call MPI_INIT(ierr)
       call MPI_COMM_RANK(MPI_COMM_WORLD,irank,ierr)
       call MPI_COMM_SIZE(MPI_COMM_WORLD,isize,ierr)
       print *,'irank isize',irank,isize
       call MPI_FINALIZE(ierr)
       end
```

Программа запускается, но сразу валится в segmentation fault со следующей диагностикой:

```
forrtl: severe (174): SIGSEGV, segmentation fault occurred
Image              PC                Routine            Line        Source

libc.so.6          000000358F72E2B0  Unknown               Unknown  Unknown
libmpi.so.1        0000002A95755D7B  Unknown               Unknown  Unknown
libmpi.so.1        0000002A9575EE08  Unknown               Unknown  Unknown
a.out              0000000000402CFB  Unknown               Unknown  Unknown
a.out              0000000000402CA2  Unknown               Unknown  Unknown
libc.so.6          000000358F71C3FB  Unknown               Unknown  Unknown
a.out              0000000000402BEA  Unknown               Unknown  Unknown
```

Причин этому, конечно, может быть несколько. Одна из самых часто встречающихся ошибок состоит в том, что в той же директории, из которой компилируется программа, существует файл «mpif.h». Причем файл этот, скорее всего, не от HP-MPI или от HP-MPI, но другой версии. В результате не совпадают прототипы функций, типы переменных и т.д.

В системе уже есть mpif.h из комплекта с библиотеки HP-MPI, который автоматически загружается из правильного места, если компилятор видит инструкцию «include 'mpif.h'». Система должна найти этот заголовочный файл сама, поскольку он является системным, принадлежит конкретной версии конкретной библиотеки MPI и ни в коем случае не является частью пользовательской программы.

Статические массивы размером более 2GB в Intel Fortran.

Если при компоновке программы возникают сообщения следующего вида

```
/tmp/ifort0ivMR0.o(.text+0xda): In function `output_t_':
: relocation truncated to fit: R_X86_64_PC32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x18a): In function `output_t_':
: relocation truncated to fit: R_X86_64_PC32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x18f): In function `output_t_':
: relocation truncated to fit: R_X86_64_32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x386): In function `output_t_':
: relocation truncated to fit: R_X86_64_PC32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x46d): In function `output_t_':
: relocation truncated to fit: R_X86_64_PC32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x472): In function `output_t_':
: relocation truncated to fit: R_X86_64_32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x59a): In function `output_t_':
: relocation truncated to fit: R_X86_64_32 eos_par_mp_dens_
/tmp/ifort0ivMR0.o(.text+0x5cd): In function `output_t_':
: relocation truncated to fit: R_X86_64_PC32 eos_par_mp_dens_
/tmp/ifortwpWEfe.o(.text+0xc): In function `pm_':
: relocation truncated to fit: R_X86_64_PC32 tabhad_
/tmp/ifortwpWEfe.o(.text+0x5e): In function `pm_':
: relocation truncated to fit: R_X86_64_32S tabhad_
/tmp/ifortwpWEfe.o(.text+0x6a): In function `pm_':
: additional relocation overflows omitted from the output
```

то, скорее всего, в программе используются статические массивы размером более 2GB. Чтобы такие программы правильно собирались, к ключам компилятора на стадии сборки самой программы нужно добавить параметры «-mcmodel medium -shared-intel».

Еще одним вариантом является использование ключа «-mcmodel large», но этот ключ нужно использовать с осторожностью, поскольку большинству программ не нужен и приводит только к увеличению их размера. Подробности можно узнать на техническом форуме Intel.

Как заставить работать FTP в Midnight Commander.

По умолчанию в Midnight Commander протокол FTP работает в активном режиме. Настройки кластера не позволяют использовать данный режим. Поэтому необходимо перенастроить Midnight Commander на использование пассивного режима. Для этого в файле настроек «.mc/ini», в раздел «[Midnight-Commander]» нужно добавить следующую строку

```
ftpfs_use_passive_connections=1
```

Если переменная «ftpfs_use_passive_connections» уже была определена, то нужно установить её значение равным единице, как показано выше.

В результате, файл «.mc/ini» будет выглядеть так (многоточиями отмечены пропущенные строчки, не имеющие отношения к проблеме):

```
…
[Midnight-Commander]
…
ftpfs_use_passive_connections=1
…
```

Sbatch выдает сообщение «No partition specified or system default partition»

Такое сообщение возникает, если не указана очередь, в которую запускается задача. Очередь можно указать при помощи ключа командной строки «-p имя_очереди» или директивы SBATCH в файле сценария запуска:

```
#SBATCH -p имя_очереди
```

При компиляции программ на языке Fortran компилятором Intel выдается предупреждение о функции feupdateenv

При компиляции программ на Fortran компилятор фирмы Intel может выдать следующее предупреждение:

```
/opt/intel/ifc/9/lib/libimf.so: warning: warning: feupdateenv is not
implemented and will always fail
```

Это всего лишь предупреждение, и оно не влияет на работу программ. Ниже приведено объяснение компании Intel, находящееся в заметках к компилятору Intel C/C++ 9.x:

```
In some earlier versions of Intel C++ Compiler, applications built
for Intel EM64T linked by default to the dynamic (shared object)
version of libimf, even though other libraries were linked statically.
In the current version, libimf is linked statically unless -i-dynamic
is used. This matches the behavior on IA-32 systems. You should use
-i-dynamic to specify the dynamic Intel libraries if you are linking
against shared objects built by Intel compilers.

A side effect of this change is that users may see the following
message from the linker:
```

```
warning: feupdateenv is not implemented and will always fail

This warning is due to a mismatch of library types and can be
ignored. The warning will not appear if -i-dynamic is used.
```

Я поместил в .bash_profile «module load …», но при заходе на машину выдается сообщение об ошибке

Файл ~/.bash_profile в редакторе vi выглядит следующим образом:

```
module load openmpi intel-compilers
~
~
~
".bash_profile" [dos] 2L, 54C
```

При заходе на кластер выдается сообщение об ошибках:

```
: No such file or directory
-bash: module: command not found
```

Причина ошибки в том, что файл .bash_profile создан в системе DOS/Windows, а затем перенесен в Unix. Это видно по наличию символов «[dos]» в строке статуса редактора vi. В этом случае содержимое файла ~/.bash_profile значения не имеет. К ошибке приводит то, что файл был создан в DOS/Windows.

Решение проблемы состоит в перекодировке файла из DOS/Windows в Unix-кодировку. Для этого можно использовать команду «fromdos имя_файла», которая перекодирует файл из кодировки DOS (CP866) в кодировку Unix (KOI8-R). В нашем случае нужно дать команду «fromdos ~/.bash_profile».

Программа на языке Fortran завершается, говоря «MPI_ERR_TYPE: invalid datatype»

Скорее всего, в программе не объявлены параметры MPI_REAL, MPI_INTEGER и другие, отвечающие за тип передаваемых данных при приеме/пересылке. Для решения проблемы нужно включить инструкцию "include 'mpif.h'" в программу. Причина может быть и более простой: указан неправильный (или незарегистрированный) тип данных MPI.

В приведённой ниже программе

```
program main
include 'mpif.h'

print *, "m = ", mpi_real
call test
end

subroutine test
print *, "m-test = ", mpi_real
end
```

значение переменной «mpi_real» внутри подпрограммы «test» не совпадает с (правильным) значением «mpi_real» внутри «main», поскольку область видимости «правильной» mpi_real ограничена программой «main». Для использования «mpi_real» внутри подпрограммы «test» нужно написать так:

```
program main
include 'mpif.h'

print *, "m = ", mpi_real
call test
end

subroutine test
include 'mpif.h'
print *, "m-test = ", mpi_real
end
```

Работа с кириллическим текстом

На головных узлах кластера по умолчанию настроена кодировка KOI8-R. Поэтому для правильной работы клиентской программы с кириллической кодировкой достаточно указать ей данную кодировку.

Причиной проблем могут стать файлы в кириллических кодировках, отличных от KOI8-R, копируемые на головные узлы кластера. На головных узлах установлены программы «fromwin», «towin», «fromdos» и «todos». Как видно из их имен, эти программы предназначены для преобразования файлов из/в кодировки Windows (CP1251) и DOS (CP866). Аргументамм этих программ служат имена файлов, которые необходимо преобразовать в соответствующую кодировку.

Странные падения программ на Fortran

Иногда программы, полностью или частично написанные на Fortran, валятся в segmentation fault без видимой причины. При этом отладчик указывает точное место возникновения ошибки - машинная инструкция «call».

Очень часто причиной является переполнение стека программы. Некоторые авторы программ на Fortran создают большие локальные массивы в подпрограммах без использования динамической памяти. При компиляции таких программ необходимо использовать ключ компилятора «-heap-arrays». В этом случае все локальные массивы перемещаются в динамическую память, значительно облегчая нагрузку на стек.

Хочется использовать больше памяти, чем есть на ядро

Если параллельной задаче требуется для каждого вычислительного процесса больше памяти, чем приходится на одно ядро на рабочем узле, и при этом количество процессов равно количеству ядер на узле, то задача не пойдёт из-за нехватки памяти. Чтобы этого избежать, необходимо следующей директивой выделить на каждый процесс не одно ядро, а несколько

```
#SBATCH --cpus-per-task N
```

где N — это число ядер, выделяемых для каждого процесса.

Эффективно это приведет к тому, что каждому из процессов будет доступно не менее чем N*k GB оперативной памяти на машинах с k GB/ядро.