Инструкция пользователя SMH11

1. Адреса

Управляющей машиной в кластере SMH11 является первый вычислительный узел, который имеет два адреса

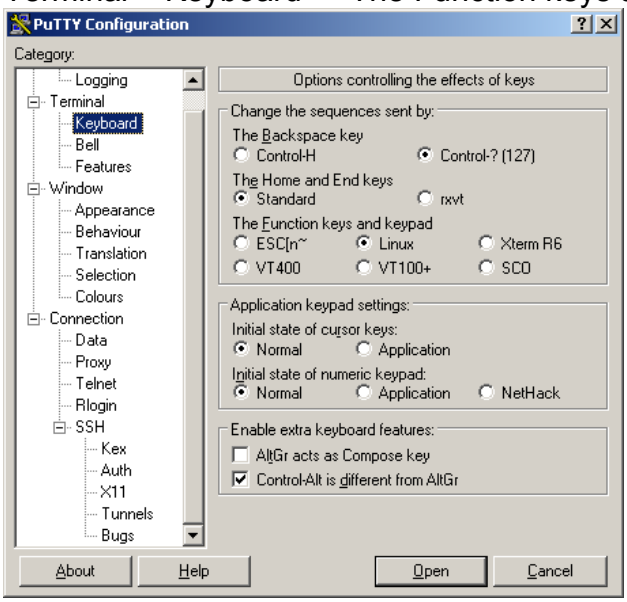
smh11.cc.dvo.ru
smh11-s1.cc.dvo.ru

Вычислительные узлы кластера имеют адреса:
1. узлы с 4мя процессорами AMD Opteron(tm) Processor 6164 HE и 64Gb памяти
smh11-s1.cc.dvo.ru – smh11-s17.cc.dvo.ru
2. узлы с 4мя процессорами AMD Opteron(tm) Processor 6174 и 128Gb памяти
smh11-x1.cc.dvo.ru – smh11-x10.cc.dvo.ru
1. узлы с 2мя процессорами Intel Xeon и 32Gb памяти и 2мя GPU Tesla 2050
smh11-t1.cc.dvo.ru – smh11-t8.cc.dvo.ru

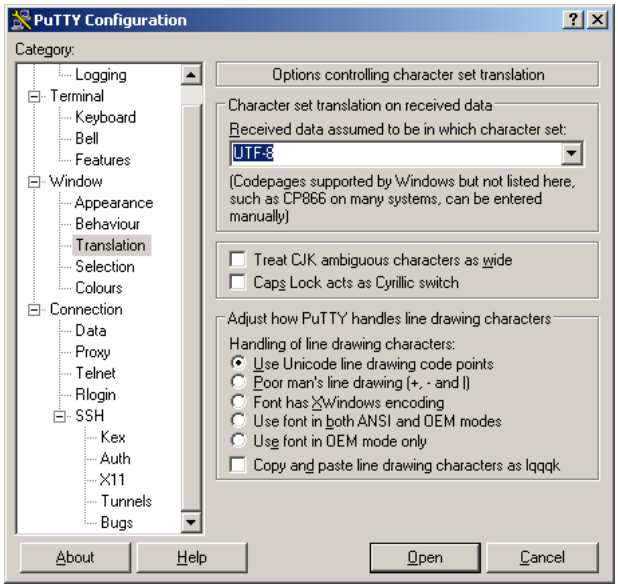
2. Доступ пользователя, копирование данных и программ

Доступ на узлы, отличные от управляющего предоставляется при наличии задач запущенных на этих узлах. Если задач не запущено, то пользователям теоретически нет необходимости в доступе на вычислительные узлы.
Все узлы используют пользовательские директории, физически расположенные на NFS сервере. На узлах эти директории имеют путь вида /home/user, где user – это логин пользователя.
Доступ пользователя к кластеру производится по протоколу SSH. Пользователи Windows могут использовать программу терминал-клиент putty.

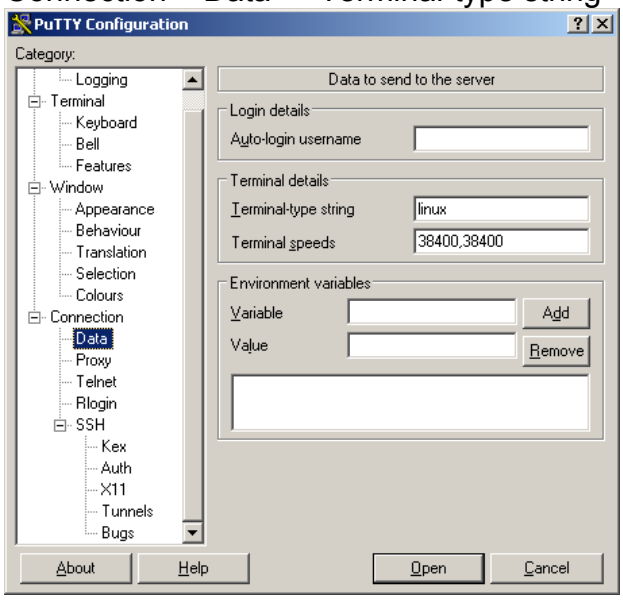
Для навигации по файловой системе можно пользоваться программой mc (midnight commander). Если вместо рамок в mc отрисовываются различные символы, то измените настройки putty:
Terminal > Keyboard > "The Function keys and keypad" = linux



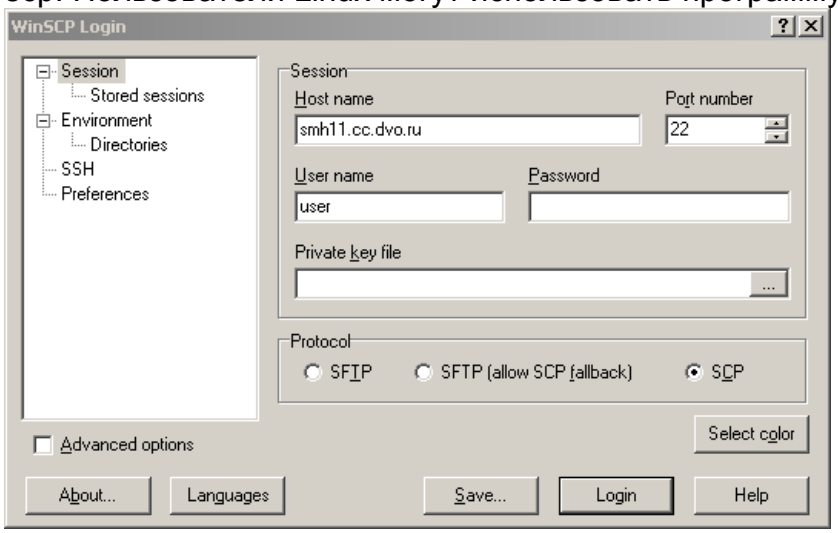
Window > Translation > Character set - выставляем правильную кодировку



Connection > Data > "Terminal-type string" пишем linux



Копирование данных и исходных текстов программ осуществляется по протоколу scp. Пользователи Linux могут использовать программу WinSCP:



3. Компиляция MPI программ

Компиляция программ на языке “Си” осуществляется при помощи команды mpicc.

Компиляция программ на языке “C++” осуществляется при помощи команды mpicxx.

Вариант №1. Компиляция из командной строки

Выполнение следующей команды:

mpicxx -o ExecutableFile *.cpp

откомпилирует все файлы в текушей директории с расширением “cpp” и сделает исполняемый файл с названием ExecutableFile.

Аналогично для программы на языке “Си”

mpicc -o ExecutableFile *.c

Вариант №2. Компиляция с использованием Makefile

Необходимо создать в директории с исходными файлами программы на языке “C++” файл под названием Makefile 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 $(ATDIR)/include
INCLUDE_DIR=-I ../

CFLAGS =  -O2 $(INCLUDE_DIR)
CPPFLAGS =
LNKFLAGS= -lstdc++

all: $(OUTFILE)

rebuild : clean ALL

CC=mpicxx
LNK=$(CC)
PREP=%%s ^\


.cpp.o :
	$(CC) $(CFLAGS) $(CPPFLAGS) -c $< -o $@

%.o: %.cpp	$(CC) $(CFLAGS) $(CPPFLAGS) -c $< -o $@

$(OUTFILE): $(OBJECTS)
	$(LNK)  -o $(OUTFILE) $(LNKFLAGS) $(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.
Второй вариант является более гибким, так как позволяет настраивать опции компилятора в переменных CFLAGS, CPPFLAGS, LNKFLAGS, INCLUDE_DIR.

4. Запуск MPI программ

Для синхронного запуска MPI программ необходимо использовать команду, являющуюся частью пакета SLURM управляющего выделением ресурсов кластера:

srun –n X ExecutableFile [optional params]

где **X** обозначает требуемое число процессов, а **ExecutableFile** означает путь к исполняемому файлу.
Пример:

```
~/projects/Test $ srun -n 2 ./Test
srun: Job is in held state, pending scheduler release
srun: job 300 queued and waiting for resources
srun: job 300 has been allocated resources
…
```

При синхронном запуске задачи выполнение команды srun закончится только тогда, когда завершится выполнение MPI программы, при этом вывод MPI программы будет отображён в терминале. Для запуска MPI программ, требующих значительного времени исполнения лучше использовать команду, также являющуюся частью пакета SLURM управляющего выделением ресурсов кластера:

sbatch –n X script [optional params]

При этом **script** – это специально составленный для выполнения MPI задачи скрипт. Он может содержать в себе множество шагов необходимых для получения конечного результата, в том числе сложное управление выполнением составления. Инструкции по составлению скрипта можно прочесть:

man sbatch

Крайне упрощённый вариант использования команды **sbatch** предлагает команда-обёртка submit, синтаксис которой повторяет команду srun:

submit –n X ExecutableFile [optional params]

Пример:

```
~/projects/Test $ submit -n 8 ./Test
Submitted batch job 308
```

Однако выполнение команды submit заканчивается, в момент постановки команды в очередь. Так как MPI задача будет выполняться не в интерактивном режиме её вывод будет перенаправлен в файл. В приведённом примере задача Test поставлена в очередь под номером 308. Вывод исполнявшейся задачи был перенаправлен в файл slurm-308.out

5. Проверка очереди

Для проверки очереди задач необходимо использовать команду:

squeue

Эта команда является частью пакета SLURM управляющего выделением ресурсов кластера. Примерный вывод команды следующий:

~/projects/Test $ squeue

```
  JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
    314  standard    task1  user_01  PD       0:00      1 smh11-s2
    316  standard    task2  user_02  PD       0:00      1 smh11-s3
    318  standard    task3  user_01  PD       0:00      1 smh11-s4
    320  standard    task4  user_02  PD       0:00      1 smh11-s5
    322  standard    task5  user_03  PD       0:00      1 smh11-s6
    327  standard    task6  user_01  PD       0:00      1 smh11-s7
    329  standard    task7  user_02  PD       0:00      4 smh11-s[6-7,9-10]
    331  standard    task8  user_03  PD       0:00      1 smh11-s[9-10]
```

Альтернативный вид отображения данных об очереди задач даёт команда

showq

Эта команда является частью пакета MAUI ответственного за составление расписания выполнения задач на кластере.

Примерный вывод команды следующий:

~/projects/Test $ showq

```
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME
341                 user_01    Running     5    INFINITY  Thu Nov  3 16:05:38

     1 Active Job        5 of  480 Processors Active (1.04%)
                         5 of   10 Nodes Active      (50.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME


0 Idle Jobs

BLOCKED JOBS----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

324                 user_03   Deferred     1    INFINITY  Thu Nov  3 15:23:17
326                 user_03   Deferred     1    INFINITY  Thu Nov  3 15:23:17
328                 user_03   Deferred     1    INFINITY  Thu Nov  3 15:23:17
320                 user_03   Deferred     1    INFINITY  Thu Nov  3 15:23:18

Total Jobs: 5   Active Jobs: 1   Idle Jobs: 0   Blocked Jobs: 4
```

В случае, если задача поставлена в очередь с ошибочными параметрами её можно отменить командой scancel.