

大工数学科学学院高性能服务器
快速入门手册

Lenovo 1800 Quick Start

Contributers: JJCAO

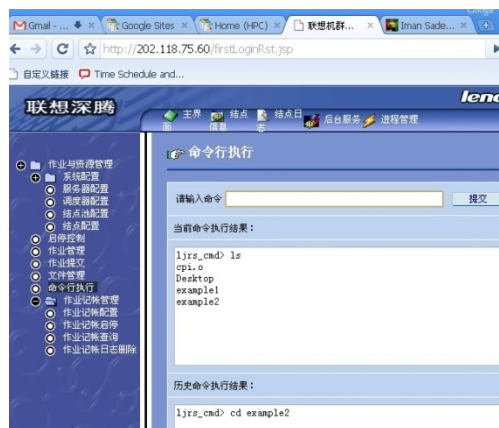
公司网址: <http://sites.google.com/site/hpcdut/>

文档版本: 1.0

1. Preparation

For conducting c++ parallel computation on the cluster, we have to master two kinds of login modes: web and shell.

“命令行执行” is a very bad tool. Forget it please. Only very few commands can be run in it successfully. Hence, for mastering the cluster, we have to incur to a shell tool, such as *SSH Secure Shell*.

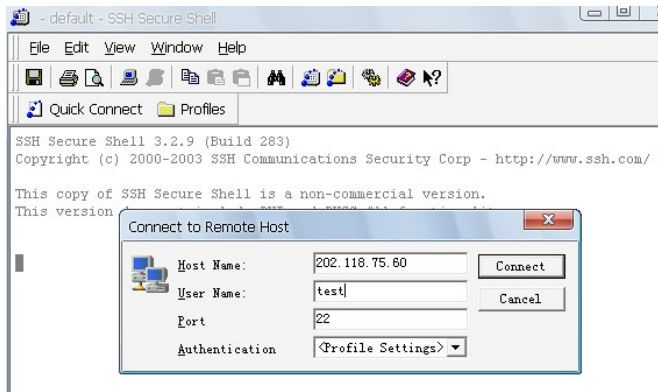


1.1. Login from the web

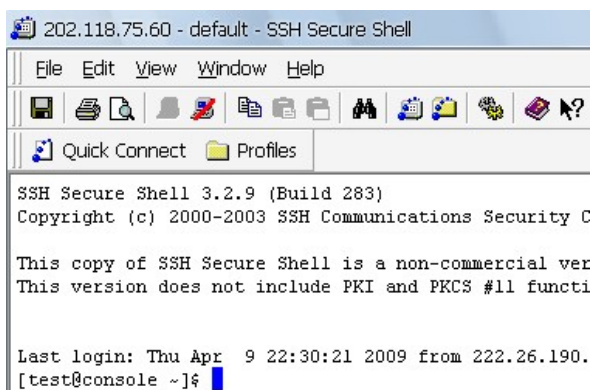


1.2. Login from the shell

We recommend SSH Secure Shell for login. The host name is the web address of the cluster.



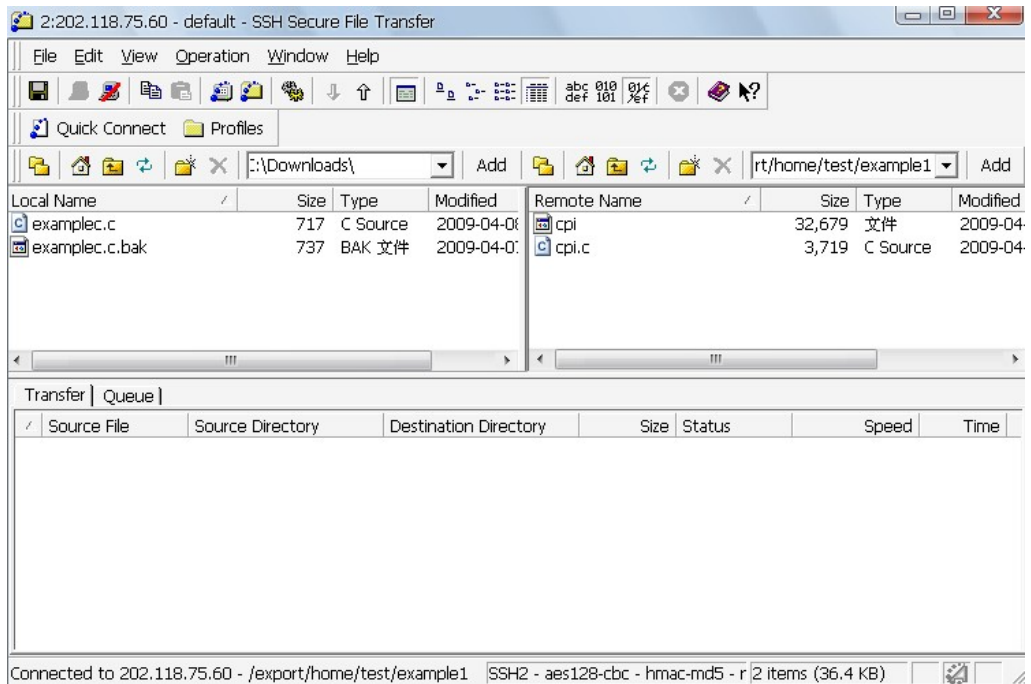
After login, we are on the *console* node actually. It is evidently if we login using a shell tool. For example in [test@console], *test* is user name, and *console* is the node name.



We can not compile any code on the console node, and it has to be done on a computation node, such as c0101, c0102, ... c0116. We can use the command *rsh* to switch from one node to another node, referring to step 4 of section 2. While *rsh* always failed to run on the web, we have to recur to the shell. Furthermore, shell offers more functions and agility.

2. C++ Parallel Computation

1. Setting environment (**This step is needed only after the user is just created or some new software is installed**)
 - a) [test@console]\$ cp /export/lenovo-hpc/bashrc.example .bashrc
2. Making a new dir for the computation, first.
 - a) [test@console ~]\$ mkdir example1
3. Uploading necessary files into the dir *example1*
 - a) Use SSH Secure File Transfer



b) or On the web



4. Compiling (**it must be done on a computational node**, such as c0101, ... c0116)

a) [test@console ~]\$ rsh c0102

b) [test@c0102 ~]\$ mpicc -o example1/cpi example1/cpi.c

If you meet any error, there will be some output on the screen.

5. Setting a job script (Our cluster uses HCA card of Qlogic, so the mpi used is [QLogic MPI](#).)

a) [test@console ~]\$ cp /export/lenovo-hpc/ljrstest example1/ljrstest

b) The content of the script is:

```
echo this script pid is $$
echo Jobing directory is $LJRS_O_JOBDIR
echo $LJRS_O_JOBDIR = $LJRS_O_JOBDIR
cd $LJRS_O_JOBDIR
echo Runing on host `hostname`
echo Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
echo \ $LJRS_NODEFILE=$LJRS_NODEFILE
echo `cat $LJRS_NODEFILE`
```

```

NPROCS=`wc -l < $LJRS_NODEFILE`
echo This job has allocated $NPROCS nodes
mpirun -m $LJRS_NODEFILE -np $NPROCS /export/home/YOU PROGRAM
echo `date`

```

- c) Edit the script. Replace “YOU PROGRAM” with absolute path of your executable file. (In this example, replace YOU PROGRAM with /export/home/test/example1/cpi)
- d) After editing, the new script is (Keeping the file in a Linux format by editing it on Linux or using software such as EditPlus on Windows):

```

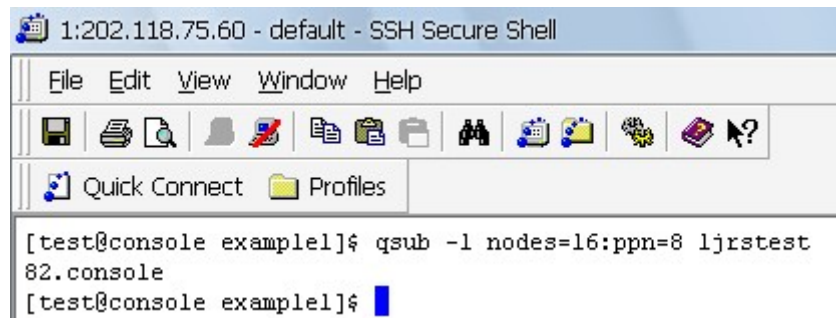
echo this script pid is $$
echo Jobing directory is $LJRS_O_JOBDIR
echo $LJRS_O_JOBDIR = $LJRS_O_JOBDIR
cd $LJRS_O_JOBDIR
echo Runing on host `hostname`
echo Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
echo \ $LJRS_NODEFILE=$LJRS_NODEFILE
echo `cat $LJRS_NODEFILE`
NPROCS=`wc -l < $LJRS_NODEFILE`
echo This job has allocated $NPROCS nodes
mpirun -m $LJRS_NODEFILE -np $NPROCS /export/home/test/example1/cpi
echo `date`

```

6. Committing the job on the web or on the console

a) On the console

- i. [test@console example1]\$ qsub -l nodes=16:ppn=8 ljrstest



```

1:202.118.75.60 - default - SSH Secure Shell
File Edit View Window Help
[test@console example1]$ qsub -l nodes=16:ppn=8 ljrstest
82.console
[test@console example1]$

```

- nodes: number of nodes
- ppn: number of cores of CPUs (The CPU of our cluster is quad-core CPU)
- There will be two standard output files: ljrstest.o82 (normal output) and ljrstest.e82 (error info)

b) On the web

- i. 作业与资源管理/作业提交

7. Check the result
In this example, result is in ljrstest.o82.

3. Job management

More info is in 联想深腾 1800 作业调度与资源管理系统用户手册.doc.

3.1. Examine standard outputs

If the job submitted was done, you can find two outputs files in the dictionary where the job script is, namely *.e??, and *.o??. 'e' stands for error info, 'o' normal output info, and '??' job id. An example for normal output is:

```
[nistest@console ~]$ cat test.o1
this script pid is 24271
Jobing directory is /export/home/nistest
= /export/home/nistest
Runing on host c0101
Time is Tue Jun 17 14:01:39 CST 2008
Directory is /export/home/nistest
This jobs runs on the following processors:
$LJRS_NODEFILE=/usr/local/spool/ljrs/aux//11.console
c0101 c0101 c0101 c0101 c0102 c0102 c0102 c0102
This job has allocated 8 nodes
g0101 g0101 g0101 g0101 g0102 g0102 g0102 g0102
```

```
running /export/home/nistest/cpi on 8 LINUX ch_p4 processors
Created /export/home/nistest/PI24280
pi is approximately 3.1416009869231249, Error is 0.0000083333333318
wall clock time = 0.003906
Tue Jun 17 14:01:39 CST 2008
```

We can compare this output with the c++ source file generated it, /export/lenovo-hpc/cpi.c.

3.2. If your job is always queued

4. FAQ

1. Why I got a compilation error (catastrophic error: could not open source file "iostream.h") when including iostream.h?
Use <iostream> instead.
2. Why I got a compilation error (catastrophic error: could not open source file "iostream") when including iostream?
Change the postfix from c to cpp.