**Running HPL**

1. Before doing this, you **MUST** have already installed HPL.
2. Go into your NCSIPARI2011\_exercises directory:

% **cd ~/NCSIPARI2011\_exercises**

1. Check to make sure that you’re in your NCSIPARI2011\_exercises directory:

% **pwd**

1. Copy the directory named HPL\_exercise from Henry’s NCSIPARI2011\_exercises directory:

% **cp -r ~hneeman/NCSIPARI2011\_exercises/HPL\_exercise .**

**NOTE**: The period (“dot”) at the end of the cp command means “to the current working directory” and is **VERY IMPORTANT**.

1. You should now have your own copy of the HPL\_exercise directory, as a subdirectory of your NCSIPARI2011\_exercises directory. Check to make sure that you do:

% **ls**

Note that this command is lower case L followed by lower case S (that is, “ell ess” which is short for “list”), **NOT** “one ess.”

You should see a list of files and subdirectories, one of which should be:

HPL\_exercise

1. Change directory into your HPL\_exercise directory, like this:

% **cd HPL\_exercise**

1. Make sure that you’re in your HPL\_exercise directory, like this:

% **pwd**

1. See what’s in this directory, like this:

% **ls**

You should see some subdirectories, such as HPL\_0001p.

1. Go into the first such subdirectory:

% **cd HPL\_0001p1t**

1. Using your preferred Unix text editor (for example, nano, pico, vi, emacs), edit the batch script file hpl\_0001p1t.bsub.

In particular:

* 1. Change yourusername to your user name.
  2. Change youremailaddress@yourinstitution.edu to your e-mail address.

1. While you’re editing the batch script file, carefully read its contents.
2. Also examine the file named HPL\_0001p1t.dat, which contains the input parameters for this run.
3. Submit the batch script file hpl\_0001p1t.bsub to the batch scheduler:

% **bsub < hpl\_0001p1t.bsub**

**NOTICE** the less than symbol < which is **EXTREMELY IMPORTANT**.

You should get back output something like this:

Job <######> is submitted to queue <pari\_q>.

where ###### is replaced by the batch job ID for the batch job that you’ve just submitted.

1. Check the status of your batch job:

% **bjobs**

You’ll get one of the following outputs, either:

No unfinished job found

(if you get this right after the bjobs command, try it several more times, because sometimes there’s a pause just before the batch job starts showing up, as below),

OR:

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

4081250 yourusername PEND pari\_q sooner1 hpl\_0001p1t Oct 17 9:58

where ###### is replaced by a batch job ID number, and yourusername is replaced by your user name, and where PEND is short for “pending,” meaning that your job is waiting to start,

OR:

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

4081250 yourusername RUN pari\_q sooner1 c127 hpl\_0001p1t Oct 17 9:58

1. You may need to check the status of your batch job repeatedly, using the bjobs command, until it runs to completion. **This may take several minutes (occasionally much longer).**

You’ll know that the batch job is done when it no longer appears in your list of batch jobs:

No unfinished job found

1. Once your job has finished running, ﬁnd the *standard output* and *standard error* files from your job:

% **ls -ltr**

Using this command, you should see files named

hpl\_0001p1t\_######\_stdout.txt

and

hpl\_0001p1t\_######\_stderr.txt

(where ###### is replaced by the batch job ID).

These files should contain the output of hpl\_0001p1t. Ideally, the stderr file should have length zero.

1. Look at the contents of the standard output file:

% **cat hpl\_0001p1t\_######\_stdout.txt**

(where ###### is replaced by the batch job ID).

You may want to look at the stderr file as well:

% **cat hpl\_0001p1t\_######\_stdout.txt**

1. What percentage of the theoretical peak of the hardware you’re running on did you achieve?

Hint: These chips are 2.0 GHz (2 billion clock cycles per second), and can perform up to 4 Floating point OPerations per clock cycle per core. Each CPU chip has 4 cores, and each compute node has 2 CPU chips. (For this first run, you’re using only a single core.)

HPL reports speeds in GFLOPs (“gigaflops,” meaning billions of FLoating point OPerations per Second).

1. Go into your HPL\_0001p2t directory:

% **cd ../HPL\_0001p2t**

1. Edit your hpl\_0001p2t.bsub batch script file.

In this file, notice that we’ve changed the number of threads to 2 but kept the number of MPI processes at 1.

1. Also examine the file named HPL\_0001p2t.dat, which contains the input parameters for this run.

How does this run’s parameters differ from the previous? Why?

1. Submit this batch job using the bsub command.
2. Monitor its progress using the bjobs command.
3. When it completes, find its stdout and stderr files, and examine them.

**How** does this run differ from the previous run?

Is the output what you expected? **Why or why not**?

1. Do the same sequence of steps (#16 - #20) with hpl\_0002p1t.bsub.

In this file, notice that we’ve changed the number of MPI processes to 2 but kept the number of threads per process at 1. So this is the opposite approach from the previous.

**How** does this run’s parameters differ from the previous? **Why**?

1. Now, create new directories with names like

HPL\_0001p4t

hpl\_0004p4t

and figure out how to change them to do the appropriate run.

Specifically, in the HPL\_whatever.dat file, you’ll want to change the following values: P, Q, Ns.

For the various values of N, we recommend doing the following:

1. Calculate:

sqrt(0.75 \* nodes \* 16 \* 1024 \* 1024 \* 1024 / 8)

(Explanation: Each compute node on Sooner has 16 GB of RAM; each double precision value is 8 bytes; you want to use only part of the RAM, because the operating system needs part of it, and you really really don’t want to use swap disk.)

1. Find the nearest multiple of 256 \* 3 \* 5 that is just below the value calculated.
2. Then divide by 2, 4 and 8 to get the 4 N values needed.

**Explain** why this would be a good idea.