Running HPL

1. Before doing this, you **MUST** have already installed HPL.
2. Go into your `NCSIPARI2011_exercises` directory:

   ```
   %  cd  ~/NCSIPARI2011_exercises
   ```

3. Check to make sure that you're in your `NCSIPARI2011_exercises` directory:

   ```
   %  pwd
   ```

4. 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**.

5. 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
   ```

6. Change directory into your `HPL_exercise` directory, like this:

   ```
   %  cd  HPL_exercise
   ```

7. Make sure that you're in your `HPL_exercise` directory, like this:

   ```
   %  pwd
   ```

8. See what's in this directory, like this:

   ```
   %  ls
   ```

   You should see some subdirectories, such as `HPL_0001p`.

9. Go into the first such subdirectory:

   ```
   %  cd  HPL_0001p1t
   ```

10. Using your preferred Unix text editor (for example, `nano`, `pico`, `vi`, `emacs`), edit the batch script file `hpl_0001p1t.bsub`.

    In particular:

    (a) Change `yourusername` to your user name.

    (b) Change `youremailaddress@yourinstitution.edu` to your e-mail address.

11. While you're editing the batch script file, carefully read its contents.
12. Also examine the file named `HPL_0001p1t.dat`, which contains the input parameters for this run.

13. 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.

14. 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
    ```

15. 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
    ```

16. 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.

17. 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
    ```

18. 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).

19. Go into your `HPL_0001p2t` directory:

    ```
    % cd ../HPL_0001p2t
    ```

20. 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.

21. 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?

22. Submit this batch job using the `bsub` command.

23. Monitor its progress using the `bjobs` command.

24. 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**?

25. 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**?

26. 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:

a. 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.)

b. Find the nearest multiple of 256 * 3 * 5 that is just below the value calculated.

c. Then divide by 2, 4 and 8 to get the 4 `N` values needed.

Explain why this would be a good idea.