(HPSC **5576** ELIZABETH JESSUP)

HIGH PERFORMANCE SCIENTIFIC COMPUTING

:: Homework / 13

:: Student / Florian Rappl

1 problem / **10** points

Selected System: **Frost**

Task is *Running HPL on Frost or Janus*

The penultimate step of building or installing any supercomputer is answering the question: "Does my system make the Top500?" (The ultimate step is, of course, running your real programs!)

This lab exercise gives you the opportunity to configure and run the HPCC HPL benchmark on a small collection of nodes in a larger supercomputer. Because HPL measures one type of numerical application performance, it works best on systems with large quantities of RAM, powerful processors, and high-throughput networks.

You may work with a single lab partner for this exercise. Please take notes as you complete each activity, including capturing important command lines, relevant output, and answers to the questions. Submit a copy to the Moodle. Only one submission per group is required, so please be sure to put both names on top of the page.

References:

- http://icl.cs.utk.edu/hpcc/
- http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html#_What_is_HPL?
- http://netlib.org/utk/people/JackDongarra/PAPERS/hplpaper.pdf

Procedure:

We'll be benchmarking HPL on your choice of Frost or Janus because it's relatively easy to do -- all of the prerequisites are installed, and we have quick access to the queues. We'll be doing limited sizes, such as a few nodes in a Frost partition or a few nodes in Janus, to keep the turnaround time short.

1. Download HPL

Download and expand HPL.

```
wget http://netlib.org/benchmark/hpl/hpl-2.0.tar.gz
tar xzvf hpl-2.0.tar.gz
```

2. Compile HPL with its default options

Instructions for compiling are located in hpl-2.0/INSTALL. The following are guidelines and tips for compiling.

It's possible to start with a sample makefile for a typical Linux system. To ensure that we get everything right, though, we'll start with a generic blank Makefile:

```
cd hpl-2.0/setup
bash ./make_generic
cp Make.UNKNOWN ../Make.hpsc
cd ..
```

Edit the configuration file:

1. In *Make.hpsc*, set the ARCH to hpsc, being careful not to add any spaces after 'hpsc'.
2. Set TOPdir to the location where you untarred the tarball, (ex. *$(HOME)/hpl-2.0*).
3. Comment out the MPdir, MPinc, and MPlib variables. You don't need them if you use the mpi compiler wrapper scripts.
4. Set the LAlib variable stanzas for the appropriate BLAS library as follows:
5. On Frost, it's */contrib/bgl/lib/libgoto.a* -- just put that in LAlib.
6. Be very careful not to add any spaces at the end of lines!
7. On Frost only, change the Fortran/C linking compatibility option in F2CDEFS from -DAdd_, which adds an extra underscore to subroutine calls, to -DNoChange, which does not. This is required to properly link with BLAS.
8. Finally, update the script to use the systems MPI-enabled compilers.

On Frost we have

```
CC = mpxlC
LINKER = mpxlC
```

Finally, compile HPL inside the hpl-2.0 directory:

```
make arch=hpsc
```

Let one of us know once you have successfully compiled HPL on your cluster.

Note: If you run into an error compiling, and need to recompile, you should clean out the old build completely before starting again:

```
make arch=hpsc clean_arch_all
make arch=hpsc
```

3. Benchmark with some of the default options

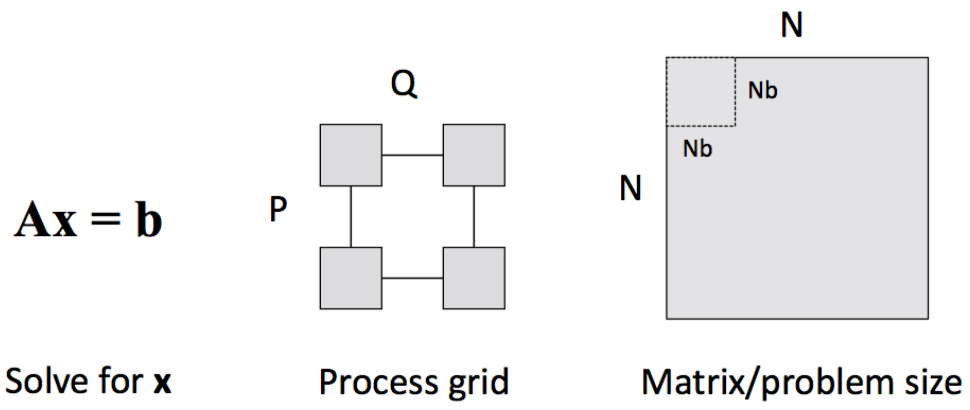
The HPL benchmark is controlled with the configuration file HPL.dat. The default HPL.dat is located in the hpl-2.0/bin/hpsc/ directory, and without reconfiguration, will run many tests on relatively small problem sizes.

Note that the default HPL file is configured to run on four nodes as defined by the process grid:

```
2 1 4        Ps
2 4 1        Qs
```

So, run HPL from the bin/hpsc directory on 4 nodes.

You may want to reduce the number of test cases run by setting the lines "# of panel fact" and "# of recursive panel fact" to '1', since they will probably not make much difference for a test on this number of nodes.



If you are on Frost, run in coprocessor mode on 4 nodes:

```
cd hpl-2.0/bin/hpsc
cqsub -n 4 -t 00:30:00 ./xhpl
```

4. Benchmark with an intelligently selected HPL problem size

Using Jack Dongarra's "rule of thumb" for sizing HPL to available memory as displayed[1]. Then, check the HPL Calculator → http://hpl-calculator.sourceforge.net/ .

Note: You can change any of the options in HPL.dat that you want, but for this exercise it's recommended to limit your changes to the first twelve lines of HPL.dat (those that control the parameters N, NB, P, and Q.)

```
N: order of coefficient matrix (problem size)
NB: blocking factor
P: columns of processors
Q: rows of processors
```

Make sure that P and Q test the proper number of processors. Each is described in more detail in the FAQ and tuning guides.

Use Dongarra's rule of thumb to select an appropriate N for 4 Frost nodes (4 tasks). Note that running HPL with a large problem size can take a very long period of time, so make sure to reduce the other variables to run fewer test cases (e.g., processor grids and blocking factors).

Solutions

Question (1):

What is the expected peak performance of a 4-node Frost partition in coprocessor mode (e.g., using 1 core/node)?

Each node is a PowerPC-440 CPU with two floating-point units core. All 8192 processors are thus possible of sustaining 22.936 trillion floating-point operations per second (TFLOPs): (4096 nodes) * (2 cores/node) * (2 FPUs/core) * (2 ops/FPU-cycle) * (700*10^6 cycles/sec)

[1] http://www.netlib.org/benchmark/hpl/faqs.html | http://www.netlib.org/benchmark/hpl/tuning.html | http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html

Answer:

By just dividing 22.936 trillion floating-point operations per second through 2 * 1024, because we use just half of the processors (coprocessor mode) and 1/1024 of the nodes we obtain

$$11.1992188 \cdot 10^9 \frac{\text{flops}}{\text{s}}.$$

Question (2):

What is the actual best performance achieved by the default HPL.dat configuration?

Answer:

The actual best performance by the default configuration was about

$$2.168 \cdot 10^7 \frac{\text{flops}}{\text{s}}.$$

Question (3):

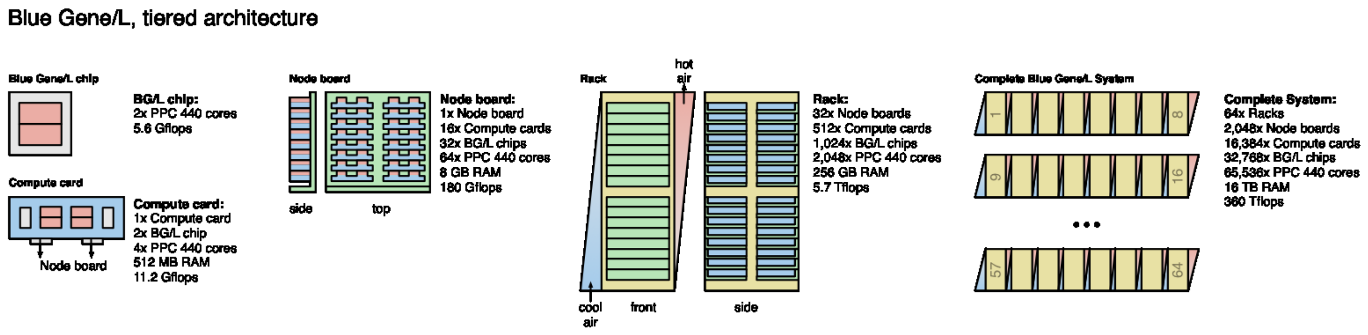
How did you calculate N?

Answer:

Following the advice of Jack I came up with (coincidence that Frost has the same spec?!)

$$\sim \frac{4}{5}\sqrt{\frac{1}{8} n \cdot M} \approx 14 \cdot 10^3,$$

when I use $n = 4$ nodes and $M = 512$ MB per compute node as specified here[2], given this picture:



Question (4):

How do your results for the larger HPL problem compare to the default run?

Answer:

It is a lot better than the default one and did perform approximately a 400 times better.

[2] http://en.wikipedia.org/wiki/File:BlueGeneL_schema.png / when we see that 2 processors (2nodes) have 512 MB we assume 256 / node.

Question (5):

How do your results compare to the calculated theoretical peak FLOP rate for the processors you are running on?

Answer:

I obtained $8.2 \cdot 10^9$ flops/s which is already close to the theoretical region.

Question (6):

What percentage of peak performance did you obtain?

Answer:

I obtained **73%** of the theoretical peak. This is quite remarkable as it's said that you can never achieve more than 80% of the machines theoretical peak.

Conclusion:

I found out that $NB = 128$ and $P \times Q = 1 \times 4$ works best for the BlueGene/L with 4 nodes. I picked the rough number of 14000 for the problem size since the equation told me so and lower ones like 10000 have not been on that level of performance as 14000 was in the end.