HPC Project1: High Performance Linpack

April 2024

1 Project Assistant Meetings

The class project assistant is Ryan Scherbarth. All teams must meet with Ryan twice. Here is his booking link. Once in the first week of the project and once in the second week. These meetings are to check your progress and to help you get unstuck if you run into trouble.

You should also consult with Maisy and I in our office hours.

2 Introduction

In this project you will run the High Performance Linpack benchmark on your team's cluster, as well as the Hopper and Wheeler Clusters. This project will teach you that how you divide a computation across CPUs and compute nodes is critical to the performance of HPC applications.
You have two goals for each cluster:

1. Find a combination of parameters that maximised the FLOPS reported by HPL on your machine.

2. Describe in your report how your choice of parameters resulted in your benchmark score, and relate that to the hardware you are using.

Make sure you start early. You cannot predict how busy Wheeler and Hopper will be so don't get stuck waiting for hours for your jobs to even start close to the deadline.

I have compiled HPL code for you to use. Refer to the Slurm scripts in the github repository to see the compiler used and the Lmod module names.

2.1 High Level Overview of HPL

The following is mostly taken from Jack Dongarra's HPL Lab page: HPL.

HPL measures the floating point execution rate for solving a system of linear equations. It relies on an MPI library and a linear algebra library for communication and solving the linear equations. The output file gives you the number of floating point operations executed per second. This is the value we want to maximise. Gflop/s is a rate of execution - billion (10^9) of floating point operations per second. Whenever this term is used it refers to 64-bit floating point operations and the operations are either addition or multiplication (a "fused" multiply/add is counted as two floating point operations).

The theoretical peak is based not on an actual performance from a benchmark run, but on a paper computation to determine the theoretical peak rate of execution of floating point operations for the machine. This is the number manufacturers often cite; it represents an upper bound on performance. That is, the manufacturer guarantees that programs will not exceed this rate - sort of a "speed of light" for a given computer. The theoretical peak performance is determined by counting the number of floating-point additions and multiplications (in full precision) that can be completed during a period of time, usually the cycle time of the machine. For example, an Intel Itanium 2 at 1.5 GHz can complete 4 floating point operations per cycle or a theoretical peak performance of 6 GFlop/s.

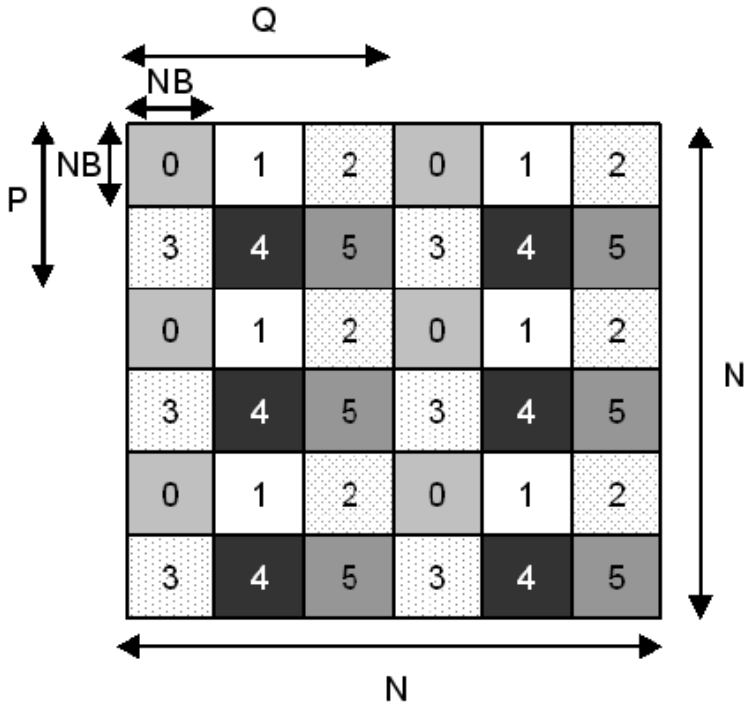
Intel no longer publishes FLOPS per cycle for their processors. Instead they have a table with total GFlops. One of the cluster CPUs is listed there:

Xeon Processor GFlop table.

Here is an HPL results sheet that you should find useful for determining the theoretical max for one of the clusters: HPC Challenge Benchmark Record.

3 Tuning HPL

You need to modify the input data file HPL.dat. The code provided in the github repository below will allow you to automate this process to some extent by defining a template file and scripts to run jobs with different HPL parameters. This file contains information about the problem sizes, machine configuration, and algorithm features to be used by the executable.



What problem size (matrix dimension N) should I use? In order to find out the best performance of your system, the largest problem size fitting in memory is what you should aim for. The amount of memory used by HPL is essentially the size of the coefficient matrix. So for example, if you have 4 nodes with 256 Mb of memory on each, this corresponds to 1 Gb total, i.e., 125 M double precision (8 bytes) elements. The square root of that number is 11585. One definitely needs to leave some memory for the OS as well as for other things, so a problem size of 10000 is likely to fit. As a rule of thumb, 80% of the total amount of memory is a good guess. If the problem size you pick is too large, swapping will occur, and the performance will drop. If multiple processes are spawned on each node (say you have 2 processors per node), what counts is the available amount of memory to each process.

HPL uses the block size NB for the data distribution as well as for the computational granularity. From a data distribution point of view, the smallest NB, the better the load balance. You definitely want to stay away from very large values of NB. From a computation point of view, a too small value of NB may limit the computational performance by a large factor because almost no data reuse will occur in the highest level of the memory hierarchy. The number of messages will also increase. Efficient matrix-multiply routines are often internally blocked. Small multiples of this blocking factor are likely to be good block sizes for HPL. The bottom line is that "good" block sizes are almost always in the [32 .. 256] interval. The best values depend on the computation

/ communication performance ratio of your system. To a much less extent, the problem size matters as well. Say for example, you empirically found that 44 was a good block size with respect to performance. 88 or 132 are likely to give slightly better results for large problem sizes because of a slightly higher flop rate.

The process grid dimensions (PxQ) depends on the physical interconnection network you have. Assuming a Infiniband mesh or a switch, HPL "likes" a 1:k ratio with k in {1..3}. In other words, P and Q should be approximately equal, with Q slightly larger than P. Examples: 2 x 2, 2 x 4, 2 x 5, 3 x 4, 4 x 4, 4 x 6, 5 x 6, 4 x 8 ... If you are running on a simple Ethernet network, there is only one wire through which all the messages are exchanged. On such a network, the performance and scalability of HPL is strongly limited and very flat process grids are likely to be the best choices: 1 x 4, 1 x 8, 2 x 4.

4 Reading

These readings go into more depth than you need for this project. Focus on exploring N, NB, and PxQ. If you want to vary other parameters you a free to do so, but make sure you have results for the simple things first.

This paper discusses LINPACK and HPL. Use it to get an overview of what LINPACK and HPL are for. The LINPACK Benchmark: Past, Present, and Future

This page has an overview of the algorithm and how the work is distributed across nodes: HPL Algorithm.

This paper has many tables showing the HPL performance of various clusters and an relatively concise description of HPL in the introduction. Performance of Various Computers Using Standard Linear Equations Software

5 Code

Pull this git repo: https://github.com/gmfricke/HPL. The repository contains an HPL template and Slurm scripts for Wheeler and Hopper.

6 Report Format

Your report will be no more than 5 pages. You should illustrate your observations with plots like those from the homeworks. In particular, you should attempt to illustrate how performance depends on the parameters you vary, with GFLops on the y-axis (independent variable) and the parameter you are interested in on the x-axis (dependent variable).

We will share a report template with each team.

7 Extra Credit

The performance of each team on the HPL benchmark will be ranked. Members of the team with the highest verified mean performance across the two clusters will receive 3 percentage points of extra credit on their *final grade for the course*. The second ranked team will receive 2 points, and the 3rd ranked team will receive 1 point.