ECE574: Cluster Computing – Homework 2

Due: Thursday 1 February 2024, 11:00am

1. **Background**
   - We will be using the HPL Linpack benchmark for this test.
   - Create a document that contains the data described in the Analysis sections below. A .pdf or .txt file is preferred but I can accept MS Office format if necessary.

2. **Connecting to the Haswell-EP Server**
   - For this assignment, log into the Haswell-EP server as described on the account slip that I handed out in class.
   - To connect, use ssh
     - On Linux or OSX you will do the following (replace username with the one on the slip):
       `ssh -p 2131 username@weaver-lab.eece.maine.edu`
     - On a Windows machine you'll want to get a program such as `putty`
     - Some possibly helpful directions can be found here, be sure you connect to port 2131:
       `https://web.eece.maine.edu/~vweaver/classes/ece571_2013s/using_ssh.html`
   - Be sure to change your password using the `passwd` command
   - You can change some of your login info with `chfn` if you'd like

3. **Aggregate measurements**
   - `time` tool
     - Prepare a working directory. On the server you can create a directory named `ece574_hw1` with a command like:
       `mkdir ece574_hw1`
       You can change into it with
       `cd ece574_hw1`
     - Copy `xhpl` and `HPL.dat` to this local directory:
       `cp /opt/ece574/hpl/* .`
       `xhpl` is a precompiled HPL-2.3 Linpack/OpenBLAS executable, and `HPL.dat` is a configuration file set up to run Linpack with N=20000
     - The copy command should also have copied a file called `time_hpl.sh`. We will be using the "slurm" job submission tool in this class. The Haswell-EP machine is a shared machine, and the job submission tool makes sure that there are never more jobs running than cores are available.
     - To submit a job, you create a shell script (such as the provided `time_hpl.sh`) and submit it with a command such as:
       `sbatch ./time_hpl.sh`
       The job will be queued up (you can check the queue with the `squeue` command.)
       Once it runs, the output will be created in the current directory with a filename such as `slurm.haswell-ep.X.out` which contains the output, where X is the job number.

- The time tool actually prints its output to stderr, which can be found in `slurm.haswell-ep.X.err`
  NOTE: there also might be an error about libibverbs in there, you can ignore that. I tried getting infiniband support going at one point and the MPI library is complaining about that.
- Try submitting the job and looking at the output just to make sure it works.
- You can change the number of threads used by this version of HPL by setting the `OMP_NUM_THREADS` environment variable. You can do this by modifying the `export OMP_NUM_THREADS=1` line in the shell script.
  You can modify the script for each run, or you can create multiple copies of the script.
- Analysis:
  (a) Run xhpl as described above with 1, 2, 4, 8, 16, and 32 threads.
  Record the time reported by HPL (in the `.out` file) as well as the GFLOPs values in your writeup. Note that this might take a few minutes to complete running. All runs are using the same problem size, N=20000.
  (b) Put the following in your writeup:
  i. List the speedup of 2, 4, 8, 16, and 32 threads (versus 1 thread).
  ii. List the parallel efficiency of 2, 4, 8, 16 and 32 threads.
  iii. Does this benchmark show strong scaling? Why or why not?
  iv. Do the results gathered contain enough info to say if the benchmark exhibits weak scaling?
  v. Look at the results of the `time` command in the `.err` file. The real value is probably higher than the value reported by HPL. Why is that?
  vi. The user time might be much larger than the real time (especially for runs like the 8-thread case). Explain what that result means.

4. **Profiling with perf**

- Create a slurm submission script that runs
  `perf record`
  on HPL while measuring 4 threads. NOTE! Be sure you run `perf record` on the `xhpl` executable, not on time or sbatch.
- The perf record run should finish and put its results in a `perf.dat` file. Use `perf report` and `perf annotate` to analyze this file. Answer the following questions.
- Questions:
  (a) What function accounts for most of the run time? (you can tell this by the highest percentage in `perf report`)
  (b) What assembly language instruction is reported as taking the most time? (you can tell this by the highest percentage in `perf annotate`)
  (c) In class we discussed "skid", an effect where modern CPUs have trouble stopping precisely when they get an interrupt and thus might attribute execution time to the wrong instruction. Can you tell from your `perf annotate` results whether this is happening or not? If you can't tell, instead state why this kind of analysis can be difficult.

5. Submitting your work.

    - Create the document containing the data as well as answers to the questions asked.
    - Please make sure your name appears in the document.
      Also include your account username (i.e. ece574-0)
    - e-mail the file to me (`vincent.weaver@maine.edu`) by the homework deadline.