ECE574: Cluster Computing – Fall 2026

Homework 8: More MPI

Due: Friday 6 November 2026, 5:00pm

1. **Background**

   - In this homework we will continue developing the Sobel code for MPI, and run it on an actual cluster.

2. **Setup**

   - For this assignment, first log into the same Haswell-EP machine we used in previous homeworks. As a reminder, use the username handed out in class and ssh in like this
     `ssh -p 2131 username@weaver-lab.eece.maine.edu`
   - Download the code template from the webpage. You can do this directly via
     `wget https://web.eece.maine.edu/~vweaver/classes/ece574/ece574_hw8_code.tar.gz`
     to avoid the hassle of copying it back and forth.
   - Decompress the code
     `tar -xzvf ece574_hw8_code.tar.gz`
   - Run `make` to compile the code.

3. **Coarse Grained Code (1 point)**

   - Note: you only need to do this step if your code from HW#7 was not working properly.
   - Take your code from HW#7 and be sure if works. If it doesn't, let me know, and I'll help you get it working for this assignment.
   - Common Remaining issue #1: loop chunk size did not match gather chunk size. Be sure your gather grabs the same amount of data that you actually calculated on each node.
   - Common Remaining issue #2: tail-end (when the image size is not a multiple of number of ranks).
   - As a reminder, if you run `md5sum out.jpg` on your results, you should get
     - butterfinger.jpg : `c32a974e2716590755fd739634d48ee3`
     - space_station_hires.jpg : `7a17b02fe7e4e676b575f6f66ba4fa01`

4. **Coarse Grained Code / Slurm / Haswell-EP (1 point)**

   Run your coarse-grained code on the Haswell-EP machine again.

   - Copy your `sobel_complete.c` code from HW#7 over top of the one in HW#8
   - To run things using slurm do the following:
     `sbatch -n 4 ./time_complete.sh`
     Where `-n 4` says to run it on 4 cores.
   - Run on 1, 2, and 4 cores and report the results in the README

5. **Fine Grained Code (4 points)**

   Use some method to improve the parallelism and see if it improves the runtime. If you already did this for HW#7, then you can re-use that code, just be sure to report what you did and the timing results.

   Note: do *not* use OpenMP for this part of the assignment.

   Copy your working `sobel_complete.c` file over `sobel_fine.c` and edit it.

   Various things that might improve performance:

   - Parallelize the combine code.
   - Using some of the other more advanced MPI calls. We didn't cover all of them in class.
   - Scattering the image pixels at the start instead of using broadcast (only sending the part needed rather than sending it all). This is tricky as for Sobel will need to send 2 extra rows (the one before and after your data, because the algorithm uses the -1 and +1 rows).

   Briefly mention what you did, as well as report times for 1, 4, 8, and 16 cores on haswell-ep. Also report speedup and parallel efficiency.

6. **Pi Cluster (4 points)**

   Run your code on the Raspberry Pi 4 Cluster. Either the fine or coarse is fine, but specify which one you used in the README.

   - To log in, first log into the haswell-ep machine as per usual.
   - Copy your code to the pi cluster. First "make clean" then cd down a directory ("cd ..") and do
     `scp -O -r ece574_hw8_code pi-cluster2:`

     Your password should be the same as on haswell-ep, I copied over the password hashes.
   - Next you will need to ssh one more time,
     `ssh pi-cluster2`
   - Change into the proper directory `cd ece574_hw8_code`
   - Run `make clean` and `make` to recompile for the ARM64 architecture.
   - You can test on the head node. Please do not use ssh to log into the sub-nodes. Unlike some clusters, this cluster is set up as 5 nodes where the head node (node00) is also set up to take part in parallel jobs. This does mean this could affect performance measurements if other people are running on the head note.
   - You can test your code on the head node to be sure it works and gives the expected results, i.e.,
     `mpiexec -np 4 ./sobel_fine butterfinger.jpg`
   - Now it is time to run in parallel on the cluster. NOTE: since this a cluster of relatively low-end Raspberry Pis things might run a bit slower than you are used to.
   - Run your code for 1, 2, 4, 8, 16 and 20 ranks (replace X below with number of ranks).
     `sbatch -n X time_fine.sh`
     Note by default slurm will place the ranks to be on the same node if possible, so likely in the 4 rank case they will all run on one Pi (which has 4 cores).

- Note that while you can queue up multiple runs at once (the whole point of a batch system) if your runs all output to the same `out.jpg` then you will only have the image results from the final run.
- **Report your timing in the README and comment on the scaling behavior on the Pi cluster. It might not scale well, think about why.**
- Remember you can use commands like `sinfo` and `squeue` to see what's going on with the cluster.
- If for some reason a job gets stuck and you want to kill it, you can use `scancel` to do that.
- If you have any issues with the cluster, please let me know.

7. **Submitting your work.**

   - Be sure to edit the README to include your name, as well as the timing results, and any notes you want to add about your something cool.
   - Run `make submit` and it should create a file called `hw08_submit.tar.gz`. E-mail this file to me.
   - e-mail the file to me by the homework deadline.