ECE574: Cluster Computing – Homework 7
More MPI

Due: Thursday 28 March 2019, 11:00am

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 http://web.eece.maine.edu/~vweaver/classes/ece574/ece574_hw7_code.tar.gz`
     to avoid the hassle of copying it back and forth.
   - Decompress the code
     `tar -xzvf ece574_hw7_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#6 was not working properly.
   - Take your code from HW#6 and fix it so it works.
   - When I graded HW#6 I sent a semi-corrected version of your code if it had issues. Note that I did not necessarily fix all issues.
   - 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).

     After each gather you might want to do something like the following (this is just example code, you can do it any way that works).

```
struct image_t tail;
int tailstart,tailsize;

/* Only run this in rank0 */
    tail.x=image.x;
    tail.y=image.y;
    tail.depth=image.depth;
    tail.pixels=calloc(image.x*image.y*image.depth,sizeof(char));

generic_convolve(&image,&tail,sobel_x_filter,
    (image.y/numtasks)*numtasks,image.y);

tailstart=(image.y/numtasks)*numtasks;
tailsize=image.y-(image.y/numtasks)*numtasks;
tailsize*=image.x*image.depth;

memcpy(&sobel_x.pixels[tailstart*image.x*image.depth],
    &tail.pixels[0],tailsize);
```

- 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, but use slurm (as I've fixed the slurm problem).

   - Copy your `sobel_coarse.c` code from HW#6 over top of the one in HW#7
   - To run things using slurm do the following:
     `sbatch -n 4 ./time_coarse.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 (just as a sanity check to be sure slurm/mpi is working).

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#6, then you can re-use that code, just be sure to report what you did and the timing results.

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

   Various things that might improve performance:

   - Reading the image from all nodes rather than just rank 0.
   - Scattering the image info (only sending the part needed rather than sending it all). This is tricky as you will need to send 2 extra rows, it's not an even split.
   - Parallelize the combine code.
   - Using some of the more advanced MPI calls. We didn't cover all of them in class.

   Report before and after times for 1, 4, 8, and 16 cores as well as speedup and parallel efficiency.

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

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

   If you want some background on how the Pi Cluster is built, you can see the paper here: **A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement** by Cloutier, Paradis and Weaver. `https://www.mdpi.com/2079-9292/5/4/61/htm`

   - To log in, first log into the haswell-ep machine as per usual.
   - Next you will need to ssh one more time,
     `ssh pi-cluster`
     Your password should be the same as it was on the Haswell machine.
   - Copy your code to the pi cluster. First "make clean" then cd down a directory ("cd ..") and do
     `scp -r ece574_hw07_code pi-cluster:`

- Run `make clean` and `make` to recompile for the ARM architecture.
- This is a real cluster, so the head-node does not do calculation. You can test your code on the head node to be sure it works and gives the expected results, i.e.,
  `mpiexec -n 4 ./sobel_coarse butterfinger.jpg`
- Now it is time to run in parallel on the cluster.
- Run your code for 1, 2, 4, 8, and 16 ranks (replace X below with number of ranks).
  `sbatch -n X time_coarse.sh`
  Note by default slurm will place the ranks to be on the same node as 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. Your results will appear in your NFS-mounted home directory.
- Remember you can use commands like `sinfo` and `squeue` to see what's going on with the cluster.
- 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 `hw07_submit.tar.gz`. E-mail this file to me.
- e-mail the file to me by the homework deadline.