ECE 574 – Cluster Computing
Lecture 13

Vince Weaver

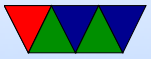
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

5 March 2019

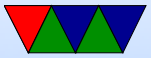
Announcements

- HW#5, still grading
- HW#6 – due date extended until Tuesday the 12th



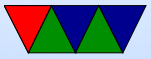
Midterm on 7 March 2019

- Closed book, closed notes.
- Performance
  - Speedup, Parallel efficiency
  - Strong and Weak scaling
- Definition of Distributed vs Shared Memory
- Know why changing order of loops can make things faster
- Pthread Programming
  - Know about race condition, deadlock



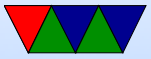
- Know roughly the layout of a pthreads program. (define pthread_t thread structures, pthread_create, pthread_join)
- Know why you'd use a mutex.

- OpenMP Programming
  - parallel directive
  - scope
  - section
  - for directive
- Know about MPI

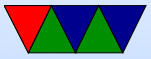


HW#5 Review

- Have to put "parallel" either in separate directive, or in sections.
- Also time measurement outside parallel area (time in each section is the same with or without threads, the difference is they can happen simultaneously). i.e. be sure to measure wall clock, not user, time
- Don't nest parallel! remove sections stuff for fine.
- Also, does it makes sense to parallelize the most inner loop of 3?

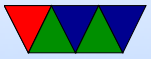


- Also what if you mark variables private that shouldn't be? scope!
- Also if have sum marked private in inner loop, need to make sure it somehow gets added on the outer (reduction).
- Be careful with bracket placement. Don't need one for a for, for example.
- Also, remember as soon as you do parallel everything in the brackets runs on X threads. So if you parallel, have loops, then a for... those outer loops are each running X times so you're calculating everything X times over.



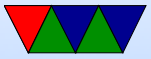
This isn't a race condition because we don't modify the inputs so it doesn't matter how many times we calc each output.

- Does dynamic vs static vs chunksize affect our code? 9 muls and adds should take consistent size. When might it not? Cache!



HW#6

- Having trouble getting slurm working with MPI :(
- Suggested coarse implementation
  - Get rank and size
  - Load the jpeg. Only in Rank0. Could you load it in all? Why or why not?
  - Need to tell other processes the size of our images. image.x, image.y, image.depth. Why? So can allocate proper sized structures on each.
  - How can do this? Just send 3 integers. Could set up



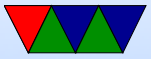
custom struct but not worth it. How send this array of 3 vars? Set up array. Bcast it? Send/receive to each, one at a time? Which is most efficient?

- Allocate space for the output images

```
new_image.pixels=malloc(image.x*image.y*image.depth*sizeof(char));
sobel_x.pixels
sobel_y.pixels
```

- Use MPI_Bcast to broadcast image data from rank0 to other ranks. Note that Bcast acts as a send from the root source (usually root 0) but as a receive on all other ranks (there's no need to separately have the other ranks receive)

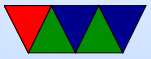
```
result = MPI_Bcast(image.pixels,                    /* buffer */
```



```
                image.x*image.y*image.depth,            /* count */
                MPI_CHAR,                               /* type */
                0,                                  /* root source */
                MPI_COMM_WORLD);
```

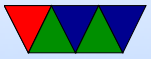
- Split up the work, you know your rank and total, so if 4 and you are #2, then you should calculate for X/4, so 0..(X/4-1), (x/4)..(x/4*2-1), etc. How to handle non-even multiple? Last rank should calc extra
- Once it is done, send back. How? MPI_Gather();

```
MPI_Gather(new_image.pixels,              /* source buffer */
           sobel_x.depth*sobel_x.x*(sobel_x.y/numtasks),   /* count */
           MPI_CHAR,                        /* type */
           sobel_x.pixels,                  /* receive buffer */
           sobel_x.depth*sobel_x.x*(sobel_x.y/numtasks),   /* count */
           MPI_CHAR,                        /* type */
           0,                               /* root source */
           MPI_COMM_WORLD);
```



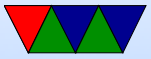
Note, it gathers from the beginning of the buffer, but put it in the right place on the root. Also, how to handle the leftover bit?

- Suggest you just do combine in rank#0, will in next HW do more fine grained
- Write out result. Remember to only write out on rank#0 (what happens if do so on all?)



Additional notes on MPI

- Hard to think about. Running on different machine, so setting variables *does not* get set on all, like it does with OpenMP or pthreads
- Tricky: before you can send to rest, they have to know how big of an area to allocate to store it in. How will they know this?
- MPI does not give good error messages. OpenMPI worse than MPICH. Will often get segfault, hang forever, or



weird stuff where it runs 4 single-threaded copies of program rather than one 4-threaded

- Many of the commands are a bit non-intuitive