ECE 574 – Cluster Computing
Lecture 13

Vince Weaver

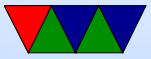
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

28 February 2023

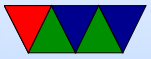
Announcements

- HW#4, HW#5: still grading
- HW#6 posted, will be due 10th

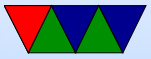


HW#4 Review

- Difficult part was picking bounds
- Other difficult part was avoiding race condition when passing parameters to the threads
  - Usual way is to just allocate a patameter struct for each thread
  - Can do complex locking, though that's not optimal
  - Please don't try to use usleep() to try to avoid the race condition by playing with the timings. This is extremely fragile and not a good idea.

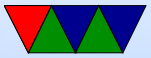


- The example code problem is an example where deadlock can occur

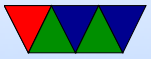


Midterm on 2 March 2023

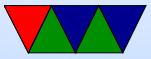
- Can bring one page (8.5" by 11" one sided) of notes. Otherwise closed notes, computers, cell-phones, Beowulf cluster, etc.
- Performance
  - Speedup, Parallel efficiency
  - Strong and Weak scaling
- Definition of Distributed vs Shared Memory
- Know why changing order of loops can make things faster



- 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

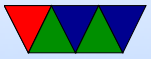


- Know about MPI



HW#6 Preview

- 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



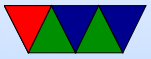
3 vars? Set up array. Bcast it? Send/receive to each, one at a time? Which is most efficient?

- 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 */
```

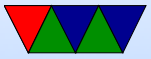


```
MPI_COMM_WORLD);
```

- 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();

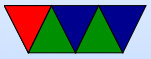
```
MPI_Gather(new_image.pixels,              /* source buffer */
           sobel_x.depth*sobel_x.x*(sobel_x.y/numtasks),   /* count */
           MPI_CHAR,                      /* type */
           sobel_x.pixels,                /* receive buffer */
           sobel_x.depth*sobel_x.x*(sobel_x.y/numtasks),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

Note, it gathers from the beginning of the buffer, but



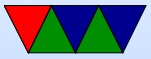
put it in the right place on the root. Also, how to handle the leftover bit?

- 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?)



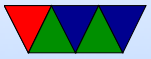
Additional notes on MPI

- 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



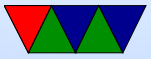
weird stuff where it runs 4 single-threaded copies of program rather than one 4-threaded

- Many of the commands are a bit non-intuitive

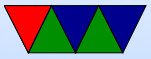


MPI Debugging (HW#6) notes

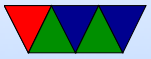
- MPI is *not* shared memory
- Picture having 4 nodes, each running a copy of your program *without* MPI.
  Also picture the various MPI routines as a network socket (or web browser query).
  Things initialized the same in all will have same values, no need to initialize.
  Things initialized in only one node will need to be somehow broadcast for the values to be the same in all.



- Problems debugging memory issues.
  Valgrind should work, but Debian compiles MPI with checkpoint support which breaks Valgrind :(
  Mpirun supposed to have -gdb option, doesn't seem to work.
- What does work is `mpiexec -n num xterm -e gdb ./your_app` but this depends on you running X11 plus logging into Haswell-EP with X forwarding (-Y) enabled
- The bug most people hit is improper bounds, leading to segfault. You can debug that with printfs of your bounds
- MPI does give useful error messages sometimes

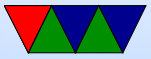


- Some of the problem is malloc/calloc

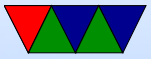


Other MPI Notes

- `MPI_Gather( sendarray, 100, MPI_INT, rbuf, 100, MPI_INT, root, comm);`
  rbuf ignored on all but root
- All collective ops are blocking by default, so you don't need an implicit barrier
- MPI_Gather(), same as if each process did an MPI_Send() and the root note did in a loop MPI_Receive() incrementing the offset.



- `MPI_Gather()` aliasing
  cannot gather into same pointer, will get an aliasing error
  Can use `MPI_IN_PLACE` instead of the send buffer on rank0.
  Why is this an error? Partly because you cannot alias in Fortran. Just avoids potential memory copying errors.
  What happens if your gathers overlap?
- Can you handle non-even buffer sizes with MPI_Gather?
  No. Two options.
  - One, just handle in one of other threads (either master



or send/receive from other)

- Two, use `MPI_Gatherv()` where you specify the displacement and sizes of what you want to gather