COIS 4350H Assignment 3

This assignment is worth 10% of your final grade.

# Theory Questions:

100-200 words each. 1 mark each.

1. SIMD vs MIMD: Summarise the key differences between SIMD and MIMD based on <https://www.geeksforgeeks.org/computer-architecture-flynns-taxonomy/> . Come up with a (single) simple example of SIMD vs MIMD. Note that most literature on SIMD vs MIMD essentially presents them as working at different levels – SIMD being things like instructions that work on vectors of data, whereas MIMD is more distributed vs shared memory, but really SIMD and MIMD operate at the same level of abstraction.
2. OpenMP is a shared memory model (threads) whereas MPI is a distributed memory model (processes). If you wanted to have global variables in MPI how you create them? (Hint: this is explicitly not what MPI was designed to do, the point of this question is to make you think about why that doesn’t work well).
3. Discuss strategies for detecting deadlock, particularly in MPI (imagine this as something for a debugging tool, rather than a solution). Yes, there are research papers on this. Don’t copy them.
4. Compare the advantages and disadvantages of a different sparse matrix formats: Dictionary of Keys, List of lists, compressed sparse rows/columns. Give an example. (The Wikipedia page on Sparse Matrices has most of the info you need here, but I want you to illustrate the difference on the same sample matrix)

# Programming + Theory (6)

1. This is the same problem as in A2 – but now you’re doing in MPI. With a very tiny theory addition at the end.

Build a large 2D matrix of floats(or doubles), cells are defined in the next paragraph in C++ (large enough that you need to create it on the heap using the keyword “new”, I’d say probably a 100x100 grid is good enough, but for testing start with something like 5x5). Your simulation is just a big loop that changes state each iteration through it.

Your simulation, basic overview: At the start of your simulation every cell starts with a value of zero. At the first frame in the middle of the leftmost edge (position 0, 49 if 0,0 is bottom left) spikes to a value of 100 000.

Each frame, every cell looks at itself and all neighbouring cells, and calculates 5% of the difference between that point and the average of its neighbours (so a cell with a value of 1000, and 8 neighbours each valued 100 does 0.05\*(1000-8\*100/8) =45, and then sends that to each neighbour. After the first iteration the 1000 becomes 1000 – (8\*45) =640, and each neighbour is 145. At an edge remember that there are fewer neighbours and corners even less.   
  
The way to do this so it’s consistent is to have basically two copies of your system state. Each iteration looks only at the previous state to determine what it will be next state (track the changes to each cell, apply that to the previous state).

At a wall, essentially nothing happens (ultimately it reflects back, but you don’t really need to do much to make that happen).   
  
Run the simulation until no cell has a value larger than 12. Count the iterations. Optimize using **MPI (feel free to use OpenMP and MPI). You need at least 4 MPI processes.** You should try this as a halo exchange problem (imagine if the global system couldn’t fit in memory in each process, even though 100x100 easily can) to try how it works and see the problems.

A picture containing shoji, building

Description automatically generated  
  
I want you to discuss your strategies for achieving this parallelisation, and how much benefit you get as you write your algorithms. With OpenMP there isn’t that much to talk about in A3 compared to what you did in A2.

It should be reasonably obvious this is basic approximation of the behaviour of a wave, without the accurate maths. If you want to do a proper Navier stokes model you can, but the point here is really the parallelization, not the physics.

Discuss how you designed/parallelised the system, dealt with concurrency issues (~200 -300 words).

I don’t expect what you do to be perfect, but take a stab at doing trying to solve it.

**Also discuss but don’t implement anything related to** how you would change this around to minimise latency at the nanosecond or picosecond level – (think about the message format for MPI, and if you could order the bytes in a way that minimises latency in processing). This is a real problem in high frequency trading.