Parallel Design Patterns

Assessed Coursework, 2025
PART TWO

The deadline for PART TWO of the assessed coursework is Friday 28th of March at 4pm. You will submit this via the PDP learn pages.

About the coursework

Part two of the coursework follows on from your initial work on part one. However, these two pieces are marked independently and your grade for part two will not depend on the answers given in part one.

In this part of the coursework you will:

- Write parallel code which parallelises the medical research company's model using the **event based coordination pattern [75%]**
- Write a report that explains your implementation and performance **[25%]**

Your code should be written in C or C++ and parallelised with MPI. You must base your parallel implementation upon the serial code which has been provided, with the results remaining unchanged.

The serial code implements a version of the brain simulation model.

- Writing parallel code that follows the event-based coordination pattern correctly, but with a single event handler per UE for the existing serial code is sufficient to receive a mark of up to around 60%
- To achieve a mark in the 60% range you should implement a distributed event-based coordination version that supports multiple event handlers per MPI process.
    - You will receive a higher mark in this range if you are able to demonstrate that your code can perform well for larger brain configurations.
- An excellent solution (for distinction level marks) should implement the two previous bullet points and split the code into two parts:
    1. **A framework** implementing the event-based coordination pattern for a generic simulation but contains no problem specific code for this specific brain simulation model.
    2. **Problem specific code** which uses the framework to solve the problem for the medical research company as detailed in the details of the model.

Your code should compile and run on either Cirrus and/or ARCHER2, and at-least across multiple nodes (distributed memory parallelism). You can target ARCHER2 and/or Cirrus and use any of the compilers available on those machines. Ensure you state which compiler you are using and target machine(s) in a README file submitted with your source code.

Code [75%]

Your code should:

- Compile and run on either Cirrus and/or ARCHER2.
- Parallelise the serial medical research company's model using the event based coordination pattern, based on the provided serial code.
- Be clear and adopt a clean design.

- Be packaged neatly with a **README** file describing how the code should be built, run and which configurations are supported. Should also include a **makefile** for building the code and **submission script** to run your executable on either Cirrus and/or ARCHER2 compute node(s). ***You should make it clear which machine your code will run on if it does not run on both.***
- Be adequately commented to a level that would allow others to work on your parallel code in the future.

Performance and the ability to handle larger problem sizes are considerations in this assessment. Whilst the focus here is on the parallelisation, there are some aspects of the existing serial code which are less than optimal, and credit will be provided if you address these as part of your solution too. ***You are free to change any part of the serial code that you wish as long as the result is correct.***

Output

As per the serial code, a summary of the status of the simulation should be written out to file upon model termination. In the serial code there is also a report on performance (the number of updates per simulated nanosecond), this need not be present in the parallel version.

Provided Configurations

There are four configurations provided, ranging from small to largest.

You will see that there are also some configuration values hard coded as preprocessor pragmas or parameters in the serial code. You are free to change any of these if you wish.

Report [25%]

Your report should mainly focus on the design of your implementation and the resulting performance you have obtained. You should explain how you have applied the event-based coordination pattern to the problem, and if your code supports multiple event handlers per MPI process then you should describe how this has been achieved. Any optimisations made to the original code should also be mentioned.

If you have developed a framework as part of an excellent solution, then you should document how it is designed to be used by the user and where the split between mechanism and policy lies.

Credit will be given in the report for exploring the performance and scaling properties of the parallelised code, for instance via weak or strong scaling experiments on either Cirrus or ARCHER2. A discussion about aspects of the code's design that help or hinder performance will also be rewarded, along with highlighting any fundamental limitations. ***You should also explicitly mention which configuration(s) are supported by your code.***

A discussion around how you guarantee correctness of your model, will gain you some extra credit, but lengthy descriptions or explanations about the output are not required to gain a very good mark.

Serial code provided to you

https://git.ecdf.ed.ac.uk/nbrown23/brain_pdp is the GitLab repository where the medical research company's serial code is located. Note that you should keep an eye on this as I will commit bug fixes here and will only email out about updates if these are very significant. You

can use the history functionality of GitLab to see what has changed from one commit to another and merge this in.

The serial code is provided to you, and you might wish to optimise at the code level around *how* this is implemented or called from other parts of the code. Fundamentally, whilst I will give credit for optimising existing serial code that has not been written optimally, I do not expect you to undertake algorithmic or behavioural changes.

If you find bugs (rather than poor practice) in the serial code, then let us know and we will fix.

Details of the brain simulation model

The model that the medical research company have described exhibits the following behaviour (this is the same as was provided for part one of the coursework):

- The brain is represented as a graph
    - Neurons are nodes in the graph, and connections are edges between these nodes
    - A small number of additional nodes represent nerves. These serve as end-points where signals are either inputs to (i.e. an external stimulus) or outputs from (i.e. the brain's response) the model
    - Individual connections (graph edges) can either be unidirectional (one way), or bidirectional (both ways)
    - Whilst there will be connections linking a specific neuron to many others in the brain, there is no guarantee that there is a connection between every pair of neurons
    - The graph is provided to the model via an input file, which is read on initialisation and remains fixed throughout code execution.
- Nerves generate random signals at random intervals
    - This signal gets sent to any connected neurons
    - All signals have a type associated with them
- Neurons receive signals from connected nodes (neurons and nerves), manipulate these signals and transmit this to connected nodes (neurons/nerves)
    - The neurons are of different types which is driven by their location in the brain
        - This type determines how frequently signals are generated by the neuron and the neuron modifies signals that are received
    - The neurons also have xyz-coordinates representing their geographic locations in the brain
    - Neurons (the number of them, and their type) are fixed and do not change throughout the simulation
    - Neurons maintain an internal state, which changes depending on the number of signals they have received in a specific timeframe
        - This state impacts how the neuron will modify signals that it receives before sending them on
    - The value of a signal determines how many nodes (neurons or nerves) the signal is sent to. After a signal has been manipulated by the neuron then the higher this value the more nodes it will be sent to.
- Edges connect two neurons or a neuron and a nerve
    - The connections have several weightings, one for each possible message type
        - This weighting will modify the value of the signal that is transmitted just before it is sent
    - These connections have a maximum value of a signal that can be transmitted through them
        - The value of a signal therefore determines how many nodes (neurons or nerves) that it will be sent to. Anything larger than the capacity of an

edge will need to be transmitted in a round robin fashion to multiple neurons/nerves
        - For signals with large values, these will wrap around nodes if the number of nodes connected to the neuron is exhausted
- The input configuration file also defines nerve inputs (firing of signals at specific nerves) at specific points in time during the simulation. This simulates some sort of input to the brain
- All outputs from nerves are logged as these represent outputs from the brain, as well as the number of neuron firings that occur in each part of the brain

Having Difficulty?

If you are struggling to get all the aspects running in parallel, then you may wish to limit your parallelisation to a simpler subset of the functionality. If this is the case, then you should state clearly in your report which simplifications you have made. A working code for a simplified model could gain as good a mark or better than a broken code attempting the full model.

A code that does not quite work might be good enough to pass as long as the ideas are correct, and the code is accompanied by a good quality report. If a non-working code is submitted, the report should explain the parts that *do* work, should describe the symptoms of why the program is not working and the steps taken to try and fix the problems. If all else fails, use the report to describe how you would have parallelised the code given more time. Describe the code that you *have* submitted.

Details of the problem size

The medical research company have input configurations on several different scales. Currently they run their serial simulation code with the following parameters, and this is classed as a small configuration:

- 124 neurons and 3086 connections

Their medium configuration size is as follows, currently the serial simulation is much slower when simulating this:

- 170 neurons and 5335 connections

Ideally, the medical research company would like to run with the following parameters, but these are currently not possible due to the excessive runtime required:

- A large problem with 272 neurons and 10249 connections
- An even larger problem with 1058 neurons and 46958 connections
- They are also keen on simulating some of the smaller problem sizes (e.g. small and medium) for more simulated minutes too