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.

并行设计模式

2025 年课程评估
第二部分

评估课程作业第二部分的截止日期为 3 月 28 日星期五下午 4 点。您将通过 PDP 学习页面提交此作业。

关于课程

课程作业的第二部分紧接着第一部分的作业。但是，这两部分是独立评分的，第二部分的成绩不取决于第一部分的答案。

在本部分课程中，您将：

- 编写并行代码，使用以下代码并行化医学研究公司的模型
  基于事件的协调模式 [75%]
- 撰写一份报告，解释你的实施和表现[25%]

您的代码应使用 C 或 C++ 编写，并使用 MPI 进行并行化。您必须以提供的串行代码为基础进行并行实现，且结果保持不变。

串行代码实现了大脑模拟模型的一个版本。

- 正确编写遵循基于事件的协调模式的并行代码，但对于现有的串行代码，每个 UE 只有一个事件处理程序，足以获得高达 60% 左右的分数

- 为了达到 60% 范围内的分数，您应该实现一个基于分布式事件的协调版本，该版本支持每个 MPI 进程的多个事件处理程序。
  - 如果您能够证明您的代码可以在更大的大脑配置中表现良好，您将在这个范围内获得更高的分数。
- 一个优秀的解决方案（针对优异水平标记）应该实现前面的两个要点，并将代码分成两部分：

  1. 为通用模拟实现基于事件的协调模式的框架，但不包含针对此特定大脑模拟模型的问题特定代码。

  2. 使用框架为医学研究公司解决问题的特定问题代码，如模型细节中所述。

您的代码应在 Cirrus 和/或 ARCHER2 上编译并运行，并且至少跨多个节点（分布式内存并行）。您可以以 ARCHER2 和/或 Cirrus 为目标，并使用这些机器上可用的任何编译器。请确保在随源代码一起提交的 README 文件中说明您正在使用的编译器和目标机器。

代碼 [75%]

您的代码应该：

- 在 Cirrus 和/或 ARCHER2 上编译并运行。
- 根据提供的串行代码，使用基于事件的协调模式并行化串行医学研究公司的模型。

- 清晰并采用简洁的设计。

- 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

- 包装整齐，带有 README 文件，描述应如何构建、运行代码以及支持哪些配置。还应包含用于构建代码的 makefile 和提交脚本，以便在 Cirrus 和/或 ARCHER2 上运行可执行文件

  计算节点。如果您的代码不能在两台机器上运行，您应该明确说明它将在哪台机器上运行。

- 进行充分的注释，以便其他人将来能够处理您的并行代码。

性能和处理更大规模问题的能力是本次评估的考虑因素。虽然这里的重点是并行化，但现有串行代码的某些方面并不理想，如果您在解决方案中也解决这些问题，我们将给予积分。只要结果正确，您可以随意更改串行代码的任何部分。

输出

根据串行代码，在模型终止时，模拟状态摘要应写入文件。串行代码中还有一份性能报告（每纳秒模拟的更新次数），这在并行版本中不需要存在。

提供的配置

提供四种配置，从小到大。

您将看到，串行代码中还有一些配置值被硬编码为预处理器指令或参数。您可以根据需要随意更改其中任何一项。

举报 [25%]

您的报告应主要关注您的实现设计以及您获得的性能。您应该解释如何将基于事件的协调模式应用于问题，如果您的代码支持每个 MPI 进程的多个事件处理程序，那么您应该描述这是如何实现的。还应提及对原始代码所做的任何优化。

如果您已经开发了一个框架作为优秀解决方案的一部分，那么您应该记录它是如何设计以供用户使用的，以及机制和策略之间的分歧在哪里。

报告中将对探索并行化代码的性能和扩展属性给予奖励，例如通过 Cirrus 或 ARCHER2 上的弱或强扩展实验。讨论代码设计中有助于或阻碍性能的方面以及强调任何基本限制也将获得奖励。您还应明确说明您的代码支持哪些配置。

围绕如何保证模型正确性的讨论将为您赢得一些额外的学分，但不需要对输出进行冗长的描述或解释即可获得很好的分数。

向您提供序列号

https://git.ecdf.ed.ac.uk/nbrown23/brain_pdp 是医学研究公司序列号所在的 GitLab 存储库。请注意，您应该密切关注这一点，因为我将在此处提交错误修复，并且只有在这些更新非常重要时才会通过电子邮件发送更新。您

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

请注意，您应该密切关注这一点，因为我将在这里提交错误修复，并且只有当这些更新非常重要时才会通过电子邮件发送更新，可以使用 GitLab 的历史记录功能查看从一次提交到另一次提交的变化并将其合并。
串行代码已提供给您，您可能希望在代码级别优化如何实现或从代码的其他部分调用它。从根本上讲，虽然我会对优化尚未最佳编写的现有串行代码表示赞赏，但我并不期望您进行算法或行为更改。

如果您在串行代码中发现错误（而非不良做法），请告知我们，我们会进行修复。

大脑模拟模型的细节

医学研究公司描述的模型表现出以下行为（这与课程第一部分提供的行为相同）：

- 大脑以图形表示 o 神经元是图形中的节点，连接是这些节点之间的边 o 少量其他节点代表神经。这些节点作为端点，信号要么输入（即外部刺激），要么输出（即

  大脑的反应）模型
  o 单个连接（图边）可以是单向的（一种方式），也可以是双向的（两种方式） o 虽然会有连接将特定的神经元与大脑中的许多其他神经元连接起来，但不能保证每对神经元之间都有连接 o 图通过输入文件提供给模型，该文件在初始化时读取并在整个代码执行过程中保持不变。

- 神经以随机间隔产生随机信号 o 该信号被发送到任何连接
  的神经元 o 所有信号都有与之相关的类型

- 神经元从连接的节点（神经元和神经）接收信号，操纵这些信号并将其传输到连接的节点（神经元/神经）o 神经元有不同的类型，这取决于它们在大脑中的位置

  这种类型决定了神经元产生信号的频率，并且神经元会修改接收到的信号 o 神经元还具有 xyz 坐标，代表它们在大脑中的地理位置 o 神经元（它们的数量和类型）是固定的，在整个模拟过程中不会改变 o 神经元保持内部状态，该状态会根据它们在特定时间范围内接收到的信号数量而变化

  此状态影响神经元在发送之前如何修改所接收的信号 o 信号的值决定了信号发送到多少个节点（神经元或神经）。信号被神经元处理后，该值越高，发送到的节点就越多。

- 边连接两个神经元或一个神经元和一个神经元 o 连接有多个权重，每个权重对应一种可能
  的消息类型
  这种加权将在信号发送之前修改信号的值 o 这些连接具有可以通过它们传输的信号的最大值

  - 因此，信号的值决定了它将被发送到多少个节点（神经元或神经）。任何大于

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

边缘需要以循环方式传输到多个神经元/神经

- 对于值较大的信号，如果连接到神经元的节点数量耗尽，这些信号将环绕节点

- 输入配置文件还定义了模拟过程中特定时间点的神经输入（向特定神经发射信号）。这模拟了对大脑的某种输入

- 所有神经输出都被记录下来，因为它们代表大脑的输出，以及大脑每个部分发生的神经元放电次数

有困难吗？

如果您难以让所有方面并行运行，那么您可能希望将并行化限制为更简单的功能子集。如果是这种情况，那么您应该在报告中明确说明您进行了哪些简化。简化模型的工作代码可以获得与尝试完整模型的损坏代码一样好的分数，甚至更好。

只要思路正确，并且代码附有高质量的报告，即使代码不能完全工作，也足够通过。如果提交了不能工作的代码，报告应该解释能工作的部分，描述程序不能工作的症状以及尝试解决问题的步骤。如果其他方法都失败了，请使用报告描述如果有更多时间，您将如何并行化代码。描述您提交的代码。

问题规模的详细信息

该医学研究公司有几种不同规模的输入配置。目前，他们使用以下参数运行串行模拟代码，这被归类为小配置：

- 124 个神经元和 3086 个连接

它们的中等配置大小如下，目前串行模拟在模拟这个时要慢得多：

- 170 个神经元和 5335 个连接

理想情况下，医学研究公司希望使用以下参数运行，但由于所需运行时间过长，目前无法实现：

- 具有 272 个神经元和 10249 个连接的大问题
- 一个更大的问题，有 1058 个神经元和 46958 个连接
- 他们也热衷于模拟一些较小的问题规模（例如小型和中型），以获得更多模拟时间