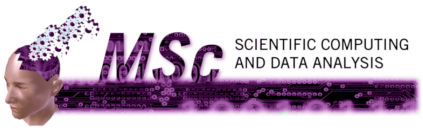

Coursework: High-Performance Computing

Module: Introduction to HPC (PHYS 52015)
Term: Michaelmas Term, 2024
Lecturer: Dr. Christopher Marcotte

Submission Please submit a **zip archive** containing at least two PDF files (`part1.pdf` & `part2.pdf`) and two code files (`part1.c` & `part2.c`).

Deadlines Consult the MISCADA learning and teaching handbook for submission deadlines.

Plagiarism and collusion Students suspected of plagiarism, either of published work or work from unpublished sources, including the work of other students, or of collusion will be dealt with according to Computer Science and University guidelines.

Coursework Description

Throughout the course we have considered simple programming problems largely distinct from the typical day-to-day practice of scientific computing. In this assignment you will experience a sliver of that practice by inheriting a codebase it is your responsibility to parallelize while maintaining correctness.

Consider the following partial differential equation (PDE),

$$\begin{aligned} \dot{u} &= \nabla^2 u + f(u,v), \quad \hat{n}\cdot\nabla u = 0, \\ \dot{v} &= d\nabla^2 v + g(u,v), \quad \hat{n}\cdot\nabla v = 0, \end{aligned}$$

which is a variant of the FitzHugh-Nagumo model, with

$$\begin{aligned} f(u,v) &= u(1-u)(u-\beta) - v + RI_c, \\ g(u,v) &= \epsilon(\alpha u - v). \end{aligned}$$

This is an example of a reaction-diffusion system — the reaction is the non-differential term on the right-hand-side, and the diffusion is the first term on the right-hand-side. Both fields are subject to 'no-flux' boundary conditions, where their normal derivative at the boundary is identically zero. These models produce a wide array of patterns, from cardiac arrhythmias to spots and stripes like those seen on animal coats. See here for an interactive example of this particular model[1].

You will be supplied with a two-dimensional reaction-diffusion code `serial.c` which reads parameters from a header file `params.h`, simulates the above equations, and tracks the solution norms,

$$w_u(t) = (\delta x\cdot\delta y)\sum_{(i,j)} u(t,x_i,y_j)^2, \quad w_v(t) = (\delta x\cdot\delta y)\sum_{(i,j)} v(t,x_i,y_j)^2, \tag{1}$$

over time, where $(i,j)\in[0,N)^2$ range over the indices of the array.

[1] Note the simulation may not work under some combinations of GPU, operating system, and browser. Windows & Chromium, macOS & Firefox, or Linux & Chromium are known to work.

Your assignment will be to parallelize the code using OpenMP and MPI, and to explain your decisions with theoretically sound arguments and measurements of performance.

Implementation Notes

- You should preserve the model parameter values; the simplest way is to not modify `params.h` and only modify the provided `serial.c` in the production of your `part1.c` and `part2.c`.
- The boundary conditions are folded into the evaluation of the diffusion term — when `i+/-1` or `j+/-1` exceeds the range of `u` or `v`, then the code just mirrors these 'ghost points' across the boundary back into the domain, e.g., `u[-1] = u[0]`.[2] Your implementation should retain this behavior for the physical boundaries.
- For scaling results you should measure the executable time, rather than the time for any subsection of the program outside of `main()`, e.g. using the unix command `time` or appropriate timing constructs covered in the course.
- The supplied run scripts are for *verification* – ensuring your code runs correctly and can be marked. You are allowed and encouraged to develop your own for running your coursework submission.

Report Notes

- The aim of your report is to demonstrate understanding of the concepts from the module, beyond the code you have submitted. This includes analysis of the performance, and clear understanding of the parallelism in-play.
- Your report should be clearly written with appropriate headings, and mathematics used correctly toward explaining the results.
- Example reports and marks are available on the module page, under `Course Materials/HPC: Week 0/High performance computing/`. Please consider the example reports when writing your own.

Part 1: OpenMP

In this assessment, you will compile and run a serial two-dimensional reaction-diffusion code, and compare its performance against a parallelized version that you will write. The serial code is made of five functions, `init`, `dxdt`, `step`, `norm`, and `main`. The expectations for your parallel implementation are to use OpenMP `#pragma`s to:

- Parallelise the function `init`.
- Parallelise the function `dxdt`.
- Parallelise the function `step`.
- Parallelise the function `norm`.

Your code should be in a single *C* file named `part1.c`. Your code ***must*** compile and run with the provided submission script, and produce the same outputs as the serial code in a file named `part1.dat`. That does not mean you must develop or test your code with these constraints, only that the submitted code must follow them.

[2] See the exercise on the heat equation for a reference.

Report

Explain and justify your parallelization strategy, using arguments based in theory covered in the course and your scaling results. Investigate the **strong** scaling of your implementation. Report scaling results using *transferable* metrics in your report. Additional questions you may wish to consider in your report are listed below. Your report should be no more than one (1) page (plus images), in a file named `part1.pdf`.

Questions to consider: What options for parallelisation are available? Why are some more suitable than others? What difficulties arise in the parallelisation? Where are the necessary synchronisation points? The solution norm requires the generation of a single output number from an `N`-by-`N` array; what patterns are available for this function? How did you avoid data races in your solution? Is your parallelisation approach the best option? What alternative approaches could be used?

Part 2: MPI

In this part of the assessment, return to the serial implementation of the two-dimensional reaction diffusion system, and parallelize the code using MPI calls, breaking down the original problem domain into distinct regions on each process. Your implementation should:

- Reproduce the initialization of `u` and `v` across processes to match the serial code.
- Correctly exchange *necessary* information of `u` and `v` across processes.
- Correctly calculate the norms of `u` and `v` across all ranks.
- Correctly evaluate the PDE on all ranks, with consideration for physical and non-physical boundaries.

Your code should be a single *C* file called `part2.c`. Your code should compile and run with the provided submission script (using 4 MPI processes), and produce the same outputs as the serial code in a file named `part2.dat`. That does not mean you must develop or test your code with these constraints, only that the submitted code must follow them.

Report

Explain and justify your parallelization strategy, using arguments based in theory covered in the course and your scaling results. Investigate the **weak** scaling of your implementation. Report scaling results using *transferable* metrics in your report. Additional questions you may wish to consider in your report are listed below. Your report should be no more than one (1) page (plus images), in a file named `part2.pdf`.

Questions to consider: What topologies for distribution are available with 4 MPI processes? Why might some be preferred over others? What difficulties arise in the parallelisation? The solution norm requires the generation of a single output number from an large distributed array — what patterns are available for this problem? What if we assume that `u` and/or `v` change slowly compared to the time-step — do any further optimizations for data exchanges become available? What are some constraints on the possible domain sizes and number of MPI processes for your solution?

Marking

Each part of your submission will be considered holistically, e.g. your code and report for Part 1 will be considered in tandem so that discrepancies between them (e.g., describing a

parallel strategy in your report that is substantially different from the one you employed in your code) will affect your marks. Your code will be run for correctness on Hamilton. If you develop your programs on your own machine, then **you should test that it works on Hamilton with the provided submission scripts**.

Submission	Points	Description
`part1.c`	25	Correct and efficient parallelization of the serial code using OpenMP, producing correct outputs. Good code practices demonstrated.
`part1.pdf`	25	Description and justification of parallelisation scheme, presentation of transferable strong scaling results, and sufficient analysis of the scaling results using concepts and theory from the course.
`part2.c`	25	Correct and efficient parallelization of the serial code using MPI, producing correct outputs. Good code practices demonstrated.
`part2.pdf`	25	Description and justification of parallelisation scheme, presentation of transferable weak scaling results, and analysis of the scaling results using concepts and theory from the course.

Table 1: Marking rubric for the summative coursework. Please see the report marking criteria in the Appendix.

Submission format

Your submission should be a single zip file, uploaded to gradescope, containing at least `part1.c`, `part2.c`, `part1.pdf`, and `part2.pdf` to be considered for full credit.

I will not accept emailed coursework submissions.

Appendix

Generic coursework remarks

Stick exactly to the submission format as specified. If you alter the format (submit an archive instead of plain files, use Word documents rather than PDFs, ...), the marker may refuse to mark the whole submission. Markers will not ask for missing files. If you have to submit code, ensure that this code does compile and, unless specified otherwise, does not require any manual interaction. Notably, markers will not debug your code, change parameters, or assess lines that are commented out.

All of MISCADA's deadlines are hard deadlines: In accordance with University procedures, submissions that are up to 5 working days late will be subject to a cap of the module pass mark. Later submissions will receive a mark of zero. If you require an extension, please submit an official extension request including medical evidence and/or acknowledgement by college. Do not contact the lecturers directly, as lecturers are not entitled to grant extensions. Details on extensions and valid reasons to grant extended deadlines can be found in the Learning and Teaching Handbook.

It is the responsibility of the student to ensure that there are sufficient backups of their work and that coursework is submitted with sufficient slack. Submit your coursework ahead of time. If in doubt, submit early versions. Technical difficulties (slow internet connection around submission deadline, lost computer hardware, accidentially deleted files, ...) will not be mitigated. Please see https://www.dur.ac.uk/learningandteaching.handbook/6/2/6/ for further information regarding illness and adverse circumstances affecting your academic performance.

If collusion or plagiarism are detected, both students who copy and students who help to copy can be penalised. Do not share any coursework with other students, do not assist

other students, cite all used sources incl. figures, code snippets, equations, ... Please see https://www.dur.ac.uk/learningandteaching.handbook/6/2/4 and https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/1 for further information.

Coursework is to be treated as private and confidential. Do not publish the whole or parts of the coursework publicly. This includes both solutions and the plain coursework as handed out.

Generic report quality criteria

Where summative coursework is assessed through written work in the form of a report, the report will be assessed against some generic criteria.

The relevant grade bands (as percentages) are

0–49 Fail

50–59 Pass

60–69 Merit

70–79 Distinction

80–100 Outstanding

A fail-level report displays an unsatisfactory knowledge and understanding of the topic. The setup and evaluation of any experimental studies is incomplete. It contains many omissions or factual inaccuracies. Limited in scope and shows little or no evidence of critical thinking and application of the course material to the problem. No recognition of limitations of the approach or evaluation. Experimental data are generally presented incorrectly, or without clarity.

A pass-level report displays some knowledge and understanding of the topic. The setup and evaluation of any experimental studies is competent. May contain some omissions or factual inaccuracies. Evidence of critical thinking and application of the course material to the problem occurs in some places. Has some recognition of limitations of the approach or evaluation. Most experimental data are presented correctly and clearly.

A merit-level report displays good knowledge and understanding of the topic as presented in the course material. The setup and evaluation of any experimental studies is careful and detailed. Broadly complete in scope, with few or no errors. Evidence of critical thinking and application of the course material to the problem is mostly clear throughout. Recognises limitations of the approach or evaluation, and has some discussion on how to overcome them. Experimental data are presented correctly and clearly.

A distinction-level report displays effectively complete knowledge and understanding of the topic. The setup and evaluation of any experimental studies is well-motivated and near-flawless. Effectively no errors. Evidence of critical thinking and application of the course material to the problem is clear throughout, and some of the discussion goes beyond the taught material. Recognises limitations of the approach or evaluation, and presents plausible approaches to overcome them. Experimental data are presented carefully and with attention to detail throughout.

An outstanding-level report is similar to a distinction-level report but is effectively flawless throughout, and shows a significant independent intellectual contribution going beyond the taught material.