CM30225 Parallel Computing
Assessed Coursework Assignment 1

Set:	Mon 22 Oct 2018
Due:	5pm., Monday 19 Nov 2018, via Moodle
Marks:	15% of course total
Set by:	Russell Bradford
Environment:	Balena, pthreads and C

The objective of this assignment is to give you experience of using low-level primitive parallelism constructs on a shared memory architecture; plus a feel for how parallel problems scale on such architectures.

The background is the solution of differential equations using a method called the *relaxation technique*. This is done by having an array of values and repeatedly replacing a value with the average of its four neighbours; excepting boundary values, which remain at fixed values. This is repeated until all values settle down to within a given precision.

This assignment is to implement relaxation, using C and pthreads on Balena.

For this assignment you will be simply repeatedly averaging sets of four numbers.

Your solver will expect an initial square array of `double` values, an integer dimension (the dimension of the array), the number of threads to use, and a precision to work to.

The edges of the array form the boundary, values there are fixed; iterate the averaging until the new values of all elements differ from the previous ones by less than the precision.

Run your code on varying sizes of arrays and varying numbers of cores in a shared memory configuration.

A complete solution consists of

- properly commented source code, together with any extra information you feel will help, such as describing your approach to the parallelisation of your solution, how you avoided race conditions, and so on
- (excerpts from) the results of your **correctness testing** of your code
- comments, graphs or any other relevant details of your **scalability investigation**, with particular reference to speedup and efficiency

Hand in your writeup as plain text or a PDF, *not* as DOC, DOCX or DOC variant (or MS spreadsheets)

Hand in the code in a text (C) file that can be compiled, separate from your writeup, *not* as a listing in the PDF (check the upload to Moodle did not convert your program file to HTML)

All answers must be placed on Moodle by 5 p.m. on Monday 19 Nov 2018. CHECK YOUR HANDIN. Moodle has been known to lose coursework.

Feedback

Feedback on this assignment will be provided via Moodle, normally within three semester weeks. There will be general feedback that applies to many people and some individual feedback. Please read your individual feedback in the context of the general feedback.

Anonymous Marking

This unit will be marked anonymously, so you may wish not to put your name on your submission.

Notes

- This assignment is testing you on the use of low-level primitives in C, so any use of higher level constructs (OpenMP, monitors, etc.) will be marked down
- It is your choice on what low-level primitives to use: if you are unsure what is classed as low-level, ask. "Low-level" certainly includes mutexes, barriers, semaphores, condition variables and `pthread` create and join. For the purposes of this assignment, atomics are not primitives
- It is your choice on how to parallelise the solution, e.g., how to partition the workload
- Balena can provide up to 16 cores in a shared memory configuration
- See the Unit web page and BUCS' web pages for how to use Balena
- If you have problems with Balena, email `hpc-support@bath.ac.uk` and give as much detail as you can about the problem. Note this is purely for problems with the machine: any other problems are your own!
- Note that the queue will get long as the hand-in date approaches. This is not an excuse for a late hand-in
- This coursework is not a complex software engineering exercise: keep it clean and simple (a single `.c`/`.h` plus a single PDF is enough)
- This coursework is not intended to assess your ability to write C, but excessively poor code will hamper your ability to get high marks (always compile with maximum warnings set, e.g., `-Wall -Wextra -Wconversion` for gcc)
- Keep the code listing to a maximum of 80 columns width: super-wide lines are harder to read and tries to put too much into a single line
- Don't use screenshots, they are always difficult to read. If you need to include relevant data, cut and paste it into your document
- Don't include hundreds of data files and log files: use your judgement as to what is the right level of information
- Don't leave this assignment to the last minute: parallel programming is more subtle than you think
- The mark that appears on Moodle is provisional, and subject to revision. The SAMIS mark is definitive

Assessment Criteria

- Normal standards for coding and commenting apply
- High marks will be gained by versions that are properly tested and have a good scalability investigation including thoughtful commentary on topics including speedup, efficiency, scalability and other relevant information
- Medium marks will be gained by versions that work but have a limited amount of testing, or a limited scalability investigation, or the comments on scalability are weak

- Low to failing marks will be gained by versions that don't work or have poor testing, or have poor or no scalability investigation, or poor or no comments on scalability

This is individual coursework. Plagiarism is not acceptable: all work must be your own.

If you have a good reason for an extension to the deadline for this coursework, please apply to the the Director of Studies, John Power.