COMP5811M
Parallel & Concurrent Programming 2019

Coursework 1

20 marks (20% of module assessment)

Issue date: 21 October 2019, 11:00 pm
Due date: 15 November 2019, 11:00 pm

This coursework involves programming with threads, protecting access to the shared data, and using synchronising operations to communicate between asynchronous threads.

Marks will be awarded for (a) correctness, (b) effective use of C++ language features and library support, and (c) professionalism -- a well-structured program with comments.

Problem Description.

Background. The problem concerns a parallel implementation of the N-queen problem on shared-memory architectures using C++ standard thread library. The N-queen problem asks how N queens can be placed on an NxN chessboard so that no two of them can attack each other. An N-queen problem can have multiple solutions e.g. 8-queen problem has 92 distinct solutions. You are provided with a sequential code which, for a given *N*, counts the number of solutions to the N-queen problem.

Assignment Project Exam Help

We know that a valid solution must have one queen per row. We start with placing the first queen on the first row; as there is no other queen on the board yet, the first queen can be placed in any of the N columns on the first row. At this stage, we have built N partially valid board states with one queen on the first row. Then, for each valid board state, we consider all valid positions for the second queen on the second row; by the end of this stage, we have built all partially valid board states with two queens on the first and the second row. The procedure continues with placing the third queen on the third row and so on. Every time a complete solution is found the counter is incremented. You can imagine that this algorithm traverses a complete tree of (partially) valid board states and increments the counter whenever a complete solution is found. However, there is no need to build this tree explicitly. To efficiently implement this algorithm, we can use recursion. The given code uses a recursive function and bitwise operation to keep track of valid positions on the board. You can find more details about this implementation here (http://www.cl.cam.ac.uk/~mr10/backtrk.pdf, page 2-3).

https://powcoder.com

Add WeChat powcoder

Task. The sequential code finds the valid positions for one queen and then recursively solves the problem for (N-1) queens. The search space and so the computation time for the N-queen problem grow exponentially. You can time the given sequential code for various numbers of queens to see the increasing runtime. Your task for this coursework focuses on exploiting parallelism to solve the N-queen problem.

Given the structure of the sequential algorithm, a straightforward solution is to launch one (asynchronous) thread for each valid position of the *i*-th queen on the *i*-th row, where each thread counts the number of solutions for the corresponding board states. However, this easy solution would launch too many threads, with many of them just burning CPU cycles and not doing much useful computation, consequently your parallel code would end up being much slower than the original sequential code.

However, dividing the main task in a recursive fashion is a good idea which suits the N-queen problem, you just need to assign these recursively unfolding sub-problems to a limited number of threads. Instead of starting a new thread for every recursive call, your program has to:

I. Launch a limited number of threads,
II. Store the unfolding sub-problems in a suitable data structure, and
III. let a free thread pick up a sub-problem to solve. A free thread has nothing else to do, either because it has finished processing all its sub-problems or because it is waiting for other threads.

Even though the number of threads in your parallel program is now bounded to a given maximum value, due to overhead of launching threads and communication between them your parallel program will still be slower

than the sequential one; to see the performance benefit of using parallelism, your parallel recursive code should switch to the sequential mode for smaller values of N, e.g. when N < 13.

Your tasks are the following:

1. Implement a parallel N-queen solver as described above. Your program must accept three parameters: N (number of queens), the maximum number of threads, and the point where your program switches to sequential mode.

2. Provide a short report (Maximum length: half page of a A4 size paper) on the performance of your parallel code. You need to report your platform specifications and your compiler settings as well as the observed speedup for a few cases (3 cases is enough) where you start to see the benefit of parallelism. You need to include the sequential and parallel run-times for each case as well as the parameter settings under which you got the results.

Advice

You have been given a sequential code that for a given N counts the number of solutions to the N-queen problem. You can use this code in your solution. Your parallel code should use a similar recursive structure, where concurrent threads unfold sub-problems and then they solve these sub-problems. To keep track of unfolding sub-problems you can use a suitable container form C++ standard library such as std::stack and std::queue.

To report the performance of you code, for this coursework, you can use Linux *time* command. I suggest you first time the sequential code for various N (e.g. 1 < N < 18). Then, try to time your parallel code using various input parameters; you may need to play with the input parameters in order to see speed up.

Assignment Project Exam Help

If your program fails because it reaches the maximum stack size on your platform, you need to adjust the switch-to-sequential-mode parameter.

https://powcoder.com

Marking Scheme

Add WeChat powcoder

Marks will be allocated as follows:

- Thread Management 4 marks
- Managing sub-problems 6 marks
- Assigning sub-problems to threads 4 marks
- Professionalism (structure, commenting) 3 marks
- Performance measurement 3 marks

Submission

Your work should be submitted through the Minerva. An entry will be created for Coursework 1 submission. You should submit a single C++ program source file and a short performance report in pdf format. To submit your files, you can ZIP them or use a unix TAR file. Other formats e.g. RAR or BZIP are NOT acceptable and WILL NOT BE MARKED, resulting a grade of 0 for this coursework. To get the full marks, your report should include platform specification, compiler setting, and a table of wall-clock run-times and achieved speed up for 3 cases where you benefit from parallelism.

Plagiarism

Your attention is drawn to the University's rules regarding plagiarism. You are not allowed to assist other students in implementing a solution to the coursework, or to seek assistance from elsewhere, e.g. programming sites such as "stackoverflow". We monitor such sites, and have colleagues who can and will alert us to attempts to solicit help. We reserve the right to interview some or all students to confirm that work submitted is indeed their own work.

Questions

Any questions on the coursework should be directed to Mai Elshehaly, email M.H.Elshehaly@leeds.ad.uk