CSE 6220-A Introduction to High Performance Computing
Spring 2016
Programming Assignment 1
Due February 11 2016

The n-queens problem is to position n queens on an $n \times n$ chessboard such that no two queens threaten each other. The problem was first published in the German chess magazine Schack in 1948. Let (i, j) and (k, l) denote the respective positions of two queens. The queens are said to threaten each other if

- $i = k$, or
- $j = l$, or
- $|i - k| = |j - l|$

Consider the standard 8×8 version of the problem. A brute force search algorithm will try 64 positions for the first queen, 63 for the next etc., to result in 1.78×10^{14} possibilities, too enormous to enumerate computationally. The search space can be significantly reduced by a few simple heuristics. First, realize that any valid solution must have exactly one queen in each row and one in each column. Thus, a possible solution to the problem can be represented by an array of n numbers. The first number denotes the row position of the queen in the first column etc. The different possibilities can be systematically explored in the following way:

Suppose that the first i entries of the array are already filled and there are no conflicts so far. Then, try each of the n possibilities for the next position. For each possibility, first check if the configuration so far is still valid. If not, reject and move on to the next possibility. If valid, then recursively try all possible ways of filling the remaining columns first, before advancing to the next possibility on the same column. When the array is full and the solution is valid, print the solution. One can come up with more sophisticated strategies to further reduce the search space and you are welcome (but not required) to do so.

The objective of the programming assignment is to write a parallel program to solve the n-queens problem. Your solution should follow the following outline: We use the master-worker paradigm, where processor with rank 0 will act as the master processor and the rest of the processors act as workers. Processor 0 will initiate the search by starting to fill an array of positions as described before. However, whenever a specific depth k is reached, a copy of the array will be dispatched to one of the worker processors. The worker processor will explore all the remaining possible combinations and report any solution(s) found to the master processor. As soon as the task is dispatched, the master processor will continue by exploring the next position on the same column etc. to generate another task and dispatch it.

One can view the master processor as performing a search limited to the first k positions. Each worker processor will get a task with first k positions filled and will perform a search for the remaining $n - k$ positions. When a worker is done with its task, it will report the outcome and

request more work. The master processor will continually dispatch work until all the tasks have been send out. At this stage, it will send a message to all workers to terminate. Then the master process will wait for remaining solutions, print out all found solutions and finally terminate itself.

Code Framework

We provide a code framework to help you get started. We require you to implement your solution within the provided framework. The framework is hosted on Georgia Tech's Enterprise GitHub page at: github.gatech.edu/pflick3/sp16-cse6220-prog1. The README file contained in the repository provides a description of the organization of the framework. In the event that we have to update the framework, we will publish the updates in this repository and notify the class via T-Square.

Submission Policies

You are expected to work in teams of two. One member from each team should submit a zip file (file name : assign_1.zip) containing the following on T-Square

1. A text file containing the names of all team members along with their contribution to the assignment (file name : names.txt)
2. All source files. Your program should be well commented and easy to read
3. A PDF report containing the following (file name : report.pdf)
    - Short design description of your algorithm(s)
    - Brief explanation of your communication protocols/mechanisms
    - Runtime plots by varying the problem size n, the parameter k, and the number of processors. Give observations on how your algorithm behaves under these variations
    - Play with the value of k and report the graphs and the value of k that yields the highest performance. What observations can you make?
    - Explain any ideas or optimizations of your own that you might have applied and how they might have affected the performance

The program will be graded on correctness and use of the appropriate parallel programming features.

Output format

Execution of your program (implemented within the provided framework) will produce a text file containing m lines, m being the number of solutions. Each solution line contains n numbers, i^{th} number indicating the position of queen in column i. The numbers are separated by a single space character. Row and column numbers start from 0.

Notes

We will enforce the specified polices strictly. You will risk losing points if you do not adhere to them. We will check all submissions for plagiarism. Any cases of plagiarism will be taken seriously, and dealt in accordance with the GT academic honor code (honor.gatech.edu).