CSE 6220 Introduction to High Performance Computing
Spring 2014
Programming Assignment 1
Due Tuesday February 4

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-slave paradigm, where processor with rank 0 will act as the master processor and the rest of the processors act as slaves. 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 slave processors. The slave processor will explore all the remaining solutions to 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 slave processor will get a task with first k positions filled and will perform a search for the remaining $n - k$ positions. When a slave is done with its task, it will report the outcome and request for

more work. The master processor will continually dispatch work until all the tasks are over. At this stage, it will send a message to all slaves to terminate and terminates itself.

Develop two versions of such a parallel program − one that stops as soon as one solution is found, and the other that stops after the entire search space is explored and all solutions are found.

Develop the following:

1. Your well-commented, easy to read, program.
2. A listing of all possible solutions for the 8-queens problem.
3. A graph plotting the run-time of the program vs. the number of processors for
   - the case where you terminate after finding one solution.
   - the case where you terminate after finding all solutions.

   Play with the value of k and report the graphs and corresponding value of k that give the highest performance. What observations can you make?

Submission guidelines

The assignment is due on **Feburary 4, 11:55 PM EST** on T-Square. You may wish to work in teams of two. One member from each team should submit a zip/tar file containing the following

1. A text file containing the names of all team members along with their contribution in terms of percentage of work done.
2. All source files.
3. A README containing instructions for compiling the program. You may include any additional information you think might be helpful.
4. A report in pdf format. Include the graphs generated and any observations that you can make about the performance. Report any ideas or optimizations of your own that you might have applied and how that might have affected the performance.

Output format

You may choose to produce separate executables for the two versions. Execution of your program should produce a text file containing m lines, m being the number of solutions. For version 1, $m = 1$.

Each solution line should contain n numbers, i^{th} number indicating the position of queen in column i. The numbers should be separated by single space character. Row and column numbers start from 0.