CS F422 Parallel Computing

Project – Exercises

Exercise II

Consider the algorithm design technique "Backtracking":

- *When it is difficult to design an algorithm that constructs the correct solution to an input instance of a decision problem, one can fall back on "searching" for solutions; searching is done by trying out permutations / combinations of components of a solution until one of the permutations / combinations turns out to be the correct solution*.
  - For instance, if the problem is to solve a system of equations, one can "choose" values for each variables and verify whether they will form solutions.
- *To ensure that the search process is systematic, one starts out with an empty solution and construct a tree where branches (or edges) correspond to decisions and states (or vertices) correspond to (partial) solutions.*
  - For instance, in the equation solving problem, one can work by assigning values to one variable at a time: each value assigned to a variable results in a different branch and each state (i.e. a partial solution) is a subset of variables for which values have been assigned.
- *The tree is traversed - as it is constructed - in a depth-first-order i.e. one path at a time;*
  - For instance, in the equation solving problem, a path would assign one value to each variable
- *If a path leads to a dead-end i.e. a state that is (internally) inconsistent, then we backtrack i.e. go back to a "previous" decision and try an alternative path:*
  - For instance, in the equation solving problem: a dead-end would mean that the values assigned to a subset of variables would not permit any value to be assigned to an unassigned variable; in which case we "undo" an assignment to one of the variables and try assigning another value to that same variable.
- *Chronological backtracking i.e. backtracking to the "immediately previous" decision can be implemented by using a stack (i.e. the same stack used in tree traversal).*

[*read* **Goodrich and Tamassia, Algorithm Design** *for more details if you are not familiar with the technique*.]

A parallel algorithm for backtracking would traverse (i.e. construct) multiple paths in parallel. As soon as one path leads to a correct solution exploration of all paths can be stopped.

Design and implement a template parallel algorithm that schedules path computations given p nodes, with c cores per node. Note that scheduling has to be dynamic because paths may evolve at uneven rates.

Demonstrate your template with full solution for at least two example problems and at least one of which exhibits uneven rates of growth. Design your own interface for assigning computation and data to your tasks. Your interface has to be simpler and at a higher level than that of MPI.

Your implementation must use MPI for message passing and either PThreads or OpenMP for shared memory computations.

Example problems:

- (Re)Solving a query given a Prolog program
- Generating ***sudoku*** puzzles (a more complex version of this problem is to generate crossword puzzles where clues are internal i.e. some letters of the words are filled in)
- Applying firewall rules on incoming packets in a network (To obtain faster response time per packet, multiple rules are to be applied in parallel; To obtain faster throughput multiple packets are processed in parallel; a correct solution would do both)