Parallel and distributed systems: paradigms and models

Final project—Academic year 2019-2020—Version 0.1

Project

The project consists in picking up one of the subjects listed below and designing a parallel application solving the proposed problem. Actually, we would like to have two applications, one using only the C++ threads (C++ STD and Pthreads only) and one using FastFlow. Not necessarily the two applications would completely share the same design. The applications should be eventually used to measure performance achieved on the machines whose access has been provided to the students. Project is an individual assignment. We will not accept "group" projects. This project assignment is valid for all the exam terms of the Academic year 2019—2020. Students must agree with the professor the choice of the subject. He/she should write a message to the professor (Subject: SPM 2020 project choice) with the project chosen. Before starting to work to the project he/she must have an explicit approval. Professor may ask the student to change his/her choice in case the choice is overbooked. A list of the project subjects will be maintained on classroom.

Project subjects

1. Brute force Sudoko solver.

Sudoko is the well-know game where on a 9x9 board you should place numbers from 1 to 9 such that each of the 9 3x3 sub boards contains all numbers from1 to 9 and no column and row contains twice the same number.

The implementation required is a brute force search implementation. After reading the initial board, you should assign to all the empty cells a list with the numbers that possibly may fill the cell. Then, you may pick up one or more of the possible assignments, assume it as "done" and go on looking for the other cell values. Parallelism can be exploited in the exploration of the subtrees generated when filling up one empty cell will all the possible numbers (one subtree per number).

2. Odd-even sort

Odd even sort works exchanging adjacent, out-of-order items in the vector to be sorted. Sorting proceeds in iterations. At each iteration, first each even position item is compared with the following (odd) one, and, in case they are out of order, they are swapped; then the odd position items are compared with the following (even position) ones and swapped if they happen to be out of order. Iterations go on up to the point no more swaps took place.

3. Travelling salesman problem with genetic algorithms

TSP is the well-known problem requiring finding the shortest path visiting just once all the nodes (cities) in a graph. Graph arcs represents connections among cities (roads) and are marked with the distance in between the source and destination city.

Genetic algorithms mimic genetic evolution of species. A set of chromosomes are defined and used to simulate a given number of "generations". Each generation is made by chromosomes deriving from parent

ones through *cross-over* or *mutation*. Cross-over takes two parent chromosomes and generates a chromosome made by segments from either from parent A or from parent B corresponding positions. Mutations randomly changes (or exchanges) positions in a single chromosome. Cross-over and mutations occur with specific probabilities. A fitness function may be used to select the "best" chromosomes to participate in the chromosome reproduction. A genetic algorithm works as follows:

```
read initial population P
for(all generations) {
        select random parents, enforcing fitness.
        apply cross-over and mutation according to probabilities.
        add new generated individual to new population.
}
```

To solve the TSP, chromosomes should be lists of cities numbers representing possible paths. Cross-over may be implemented picking up two random indexes in the list i and j with i<j and substituting the segment from I to j in the first chromosome with the order list of cities not appearing in the resto of the first chromosome taken from the second one in the order they are stored there. Mutation can be implemented exchanging two random positions i and j in the chromosome. Fitness is the length of the path represented in the chromosome.

4. Student choice

Students may autonomously propose an application to parallelize as subject for the project. In this case, the student must present to the professor a description of the problem chosen (single page. Possible with some reference). The description should include a) the problem and b) what the student assumes to be the possibilities for parallelism exploitation. Once approved, the student may work on the project as if it was one of those proposed in this document.

Project delivery

Once completed, the project must be delivered to the professor by email, by one of the exam terms. The email must be made such that the subject is "SPM Project submission" and must have, as attachments, the code developed (all the files needed to recompile and rerun the project) and a report (PDF) of max 10 pages describing the main design choices, the expected performances, the actual implementation and results achieved and a comparison among the design/development and performances of the C++ and the FF solutions.