Parallel and Distributed Systems: paradigms and models

Final Project: Academic year 2022—2023

The Project

Aim

The final project of the Parallel and distributed systems: paradigms and model course for the A.Y. 22—23 consists in providing a parallel implementation of one of the following subjects. There are two subjects (plus a free choice one) in the "Applications" section and two (plus a free choice one) in the "Run time supports" section. The former subjects consist in the implementation of small applications or kernels in parallel, and require two implementations, one developed using native C++ threads and one developed using FastFlow. The latter subjects consist in the implementation of a small library implementing/supporting the execution in parallel of a particular parallel computation pattern. In this case, it is required to provide again two implementations, one developed using native C++ threads and the other one developed using a limited subset of FastFlow components.

Project assignment

Each student must pick up **one** (and only one) of the subjects and send an email to the professor (Subject "SPM23: project choice") with the indication of the choice. After email confirmation the student may start working on the subject agreed.

Project development

The project must be considered first from a theoretical viewpoint. Students must consider the problem at hand, figure out viable solutions, implement a sequential version providing useful insights on the weight and kind of different computations involved and then design a proper solution among those investigated and start the actual project coding. The wrong way to work consists in coding different solutions directly and then filtering out the "good" one (please consider what stated in this respect during lessons and project discussion).

Project submission

After implementing the project, students must deliver the project code and a PDF report by email, by the term stated on esami.unipi.it, Subject "SPM23 project submission". The date on esami.unipi.it is the deadline for the submission of the projects, EOB. We will take some time to review the submitted code and report and then an agenda will be published with a suitable number of slots to reserve the oral exam date.

Exam

The oral exam will consist of a discussion of the project, and of some questions related to the more theoretical subjects covered during the lessons and will take place in the days following submission term, right after all the submitted project for the term have been analyzed and evaluated.

Project subjects

Section A: Application projects

A.1 Huffman coding

Huffman code is an optimal prefix code that is commonly used for lossless data compression. We require to implement a parallel application reading an ASCII file, computing the Huffman code and using the code to write a “compressed” version of the original file. Measures relative to parallelization should be taken both including and excluding read (write) times of the input (output) file in (from) memory.

(see https://en.wikipedia.org/wiki/Huffman_coding)

A.2 Genetic TSP

Given a list of cities and the distances between each pair of cities, the TSP algorithm aims at finding the shortest possible route that visits each city exactly once and returns to the origin city. The genetic algorithm to be implemented assumes a path is represented in a chromosome as a vector of integers P. P[k] = h, means the path visits city h as k-th city. Assuming a random initial population of chromosomes representing paths, crossover, mutation and other genetic algorithm techniques can be applied to make the population evolve to represent better and better solutions. The population evolution has to be implemented in parallel. Genetic TSP implementation should take as a parameter the number of iterations/generations to be computed before returning the result.

(see https://en.wikipedia.org/wiki/Genetic_algorithm)

A.3 Free choice

In case you have some application or kernel, for instance from your domain of interest, and you are interested in working on a structured parallel implementation, you may pick up this “free choice” subject. The subject must be agreed with the professor. This means you have to write a few lines (half a page max) describing the kind of application/kernel you want to parallelize and send a message to professor (Subject “SPM22 project choice”). After approval, you can start working on the subject. The free choice application/kernel should be implemented in two versions, C++ native threads and FastFlow, as for the other application subjects.

Section B: Runtime support projects

B.1 Autonomic Farm

We require to provide the implementation of a Farm pattern suitable to adapt the number of workers among a min and a max value to provide the best service time and efficiency possibly according to the actual input pressure. Adaptation should be implemented in such a way that it is fully managed in an autonomic manner. To check the implementation features, synthetic benchmarks can be used. The implementation of the autonomic farm should be implemented in such a way declaration is separated from the computation of the pattern.

B.2 “Stencil” parallel pattern

We require to provide a new parallel pattern implementing parallel computation of a number of iterations of a stencil operator over bidimensional matrixes. The stencil pattern should take as parameters a function **function<T(vector<T>)> f** computing the stencil on a single item (plus neighborhood), a **vector<pair<int,int>> neighborhood** representing the index offsets of the items

belonging (along with the item) to the neighborhood (as an example, a cross stencil will be defined by the vector of pairs <<-1,0>, <1,0>, <0,-1>, <0,1>>), and an **int iter** number of iterations to be computed. In addition, the pattern should provide the **operator()** that applied to a **vector<vector<T>>** data set computes iter iterations of a stencil pattern and returns a pointer to the computed matrix. All stencils always include the current item. Borders should be excluded from iteration computation, that is new item values have to be computed only for those positions with all the neighborhood accessible.

B.3 Free choice

In case you have some parallel pattern, for instance from your domain of interest, and you are interested in working on a structured parallel implementation, you may pick up this "free choice" subject. The subject must be agreed with the professor. This means you must write a few lines (half a page max) describing the kind of pattern you want to implement along with an idea of some different cases where the pattern may be effectively exploited, if available and send a message to professor (Subject "SPM22 project choice"). After approval, you can start working on the subject. As for the other pattern subjects, the implementation must be provided either using native C++ threads or FastFlow, limited to either the base patterns ff_pipe and ff_farm (with all their variants), or to the FF Buidling block components.

Report

The report submitted along with the project code:

- Should be at most 10 pages
- Should not include known material, such as project problem description, known performance formulas, and so on
- Should detail the process leading to the final implementation (different alternatives, considered, evaluations used to pick up the final alternative, problems faced in the implementation and particular solutions taken, etc.)
- Should include some discussion relative to the motivation of the differences between achieved performance figures and the ideal/predicted ones, in particular pointing out qualitatively and quantitively, the major sources of overhead.
- Should include experiments performed on the remote virtual machine, with proper plots (B&W only, please do not use colors)
- Should include the instructions needed to compile and run the code

Methodology

In all cases, either in case of applications or in case of run time supports, it is required that a) first the problem is analyzed, and different possible alternative implementations are defined, b) then the alternatives are evaluated and c) finally the candidate solution is implemented. At the end, measured results must be compared with the predicted/expected figures and differences (if any) should be properly analyzed and motivated.