Advanced Computer Architecture Prof. M. Ferretti, 2018-2019

Parallel programming project

The final exam of the course "Advanced Computer Architecture" consists of **three compulsory parts**: a **written test**, a **parallel programming project** (detailed in this document), and the **discussion of the project**. An oral exam is always possible, but not mandatory.

A **group of one or two students** should carry out the project; larger groups will not be approved. The project **must comply with the specifications listed in this document** and it must be described in a **technical report** to be defended in an oral presentation.

Cloud implementation. In addition to a PC implementation, the project must also be deployed on a Virtual Machine (VM), properly set up on the Google Platform in IaaS mode; the VM must be used within the assigned budget to carry out a scaling analysis on a moderate number of cores (up to 32).

The project report must be delivered **within the day of one of the scheduled dates for the written test** ("appelli"), as published by the Faculty web site. The discussion will be held about one week later (the exact schedule will be published on the course web site).

The technical project report must be sent as pdf or doc document to Prof. Ferretti (marco.ferretti@unipv.it) and to Dr. Musci (mirto.musci@unipv.it). No printed version is accepted. A late delivery prevents access to the final exam.

The discussion of the project by the group can last a maximum of **15 minutes**. Exceeding this time constraint will negatively affect the final grade. Students should present their work using a laptop of their own; it is a good idea to bring a **VGA adapter** to the discussion. Students in the same group must defend their project together, must jointly present their work but be prepared to answer individual question, and must clearly identify in the presentation their **personal contribution** to the project.

Aim of the project

The focus of the project is to apply basic parallel programming skills to devise a working solution to a given problem. No advanced programming skills are needed; on the other hand, critical thinking is expected.

The **OpenMP** model is the framework of choice for this project. Candidates can freely adopt any other language (e.g. PThread, CUDA, etc.), provided they communicate such a choice in advance. For a CUDA implementation on the Google Cloud Platform, a specific request must be made in advance to the professor.

Structure of the report

Students **must choose their project among those listed in the following section**. In alternative, students may propose a project of their own, provided the choice is approved by either the professor or the assistant.

The report must be structured as follows:

1. **Analysis of the serial algorithm**. The student should properly introduce and describe the serial algorithm of choice. Please do not copy and paste Wikipedia pages, we will notice. If needed, the student can produce examples, graphs, figures, measures, application instances, and so on.
2. **A-priori study of available parallelism**. The student should review the serial code and outline the functionalities, code blocks and/or data structures that can be considered for parallelization. The student should discuss multiple strategies for parallelization, explaining why some part can be parallelized and some cannot, and the expected overall results. In particular, all shared data structures should be outlined clearly in the text and, if needed, an appropriate synchronization strategy must be discussed.
3. **OpenMP parallel implementation**. The student must implement the strategy discussed in the previous point using C or C++. Python and Java can be used for supplementary code or alternate versions. The code must be properly commented. Multiple implementations for different parallelization strategies are welcome, and pros/cons of either should be highlighted. Source code from external sources **can be used**, but it must be clearly indicated in the report what it is and what it is not an original contribution. The report should not contain the entire source code, but only the most critical and interesting parts. The complete source code should be attached to the report in a separate file.
4. **Testing and debugging**. The student should carefully debug their code and design appropriate test cases, feeding large amounts of diverse data to the code. A static code analysis is advised, with emphasis on the access to shared resources and memory.
5. **Performance analysis**. The student should analyze the performance of his implementation in term of execution time, memory occupancy (if needed) and *speedup*. A **theoretical assessm**ent (through Amdahl's law) is required, as well as an analysis of the results, on multiple architectures if possible. A scalability analysis (i.e. performance varying the number of threads) is mandatory: the **VM on Google Cloud Platform** must be used to extend this analysis to large number of cores. **Problem size analysis** (i.e. performance varying input size) is mandatory. Remember that the advantages of parallel implementations can only be appreciated by feeding large amount of data as input. In order to do so, you must generally use dynamic memory allocation (e.g. malloc) to store your data.

Quality over quantity:

Using large number of pages, figures, graphs, code snippet or examples is often unnecessary. Synthesis, focus and critical thinking will be greatly appreciated.

Evaluation

The project will be evaluated in a letters scale from A+ (excellent) to F (fail). The grade will be finalized only after the project discussion. The **evaluation** will take into account:

- Clarity and effectiveness of presentation;
- Depth, correctness and originality of the theoretical analysis;
- Technical skills and documentation of the implementation;
- Number and quality of multiple parallel solutions, if present;
- Critical thinking in the performance evaluation and analysis.
- **Significance of the results**: as parallel programming is mostly concerned about performance, a good project **must** also provide significant speedup.

Note: problem complexity will be taken into consideration for the evaluation; students dealing with easier problems should expect less leniency in the evaluation than students dealing with hard, long and/or complex problems.

Please keep in mind that we take plagiarism very seriously. Use of external sources is not prohibited, but encouraged; however, every occurrence must be referenced.
A violation of this implicit code of trust could result in the dismissal from the final exam.

Proposed projects

In the following, some well-known problems, topics and/or algorithms are listed. Some are from mostly theoretical areas, such as graph analysis or mathematics, and some from applied ones, such as computer vision, artificial intelligence and physics.

A **large degree of freedom** is awarded to each group in the definition of the scope and the extent of their projects. In case of doubt, please contact the professors.

Any of the algorithms listed below is automatically approved for the project. Further proposals can be put forward, and will be evaluated by the professor and the assistant.

For most project, a reference serial implementation is easily found by searching the web or by referring to any book on Algorithms and Data Structures (such as to the "bible" on informatics, **The Art of Computer Programming** by Donald Knuth), Machine Learning, Computer Vision and the like.

The **openCV** library is advised for all ancillary operations on images (such as I/O and memory management).

Computer Vision:

1. Mathematical morphology (with applications);
2. Optical flow
3. (Generalized) Hough Transform
4. Haar/Wavelet Transform
5. (other Computer Vision problems, to be discussed with the professors)

Physics & Mathematics

1. Heat propagation (3D only)
2. Wave equation (2D only)
3. Fast Fourier Transform (2D)
4. (any other finite element problems, to be discussed with the professors)
5. Optimized matrix multiplication and inversion
6. (other linear algebra optimization problems to be discussed with the professors)

Artificial Intelligence and Machine Learning

1. Forward and back propagation on artificial neural networks
2. SVM, PCA, singular value decomposition (any of these)
3. K-means clustering
4. (other supervised/unsupervised learning problems to be discussed with the professors)

Other:

1. Pathfinding with applications to robotics or videogames
2. Substring matching with application to genomics/proteomics