Advanced Computer Architecture
Prof. M. Ferretti
2024-2025

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. The report must specify the role of each student in project development.

Cloud implementation. The project must be deployed on clusters of Virtual Machines (VM), properly set up on the Google Platform in IaaS mode; the VM clusters must be used within the assigned budget to carry out a scaling analysis taking into account the limitations of the Google Education Programme (Nov 2024: currently there is an overall limit to the numer of vcores available, which is 24 within a given region, and 32 overall; student wishing to obtain more resources/quotas can apply directly with Google here) and with a suitable strategy along the following directions:

A) Fat cluster: a cluster designed with few, "powerfull" VM, hosting a large number of vcores, with proper memory

B) Light cluster: a cluster designed with many, light-weighted vcores (tipically 2 vcores)

C) Intra Regional / Infra Regional cluster: the VMs are located within a single region vs more regions

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). However, delivery of projects after August 31 is prohibited – **all project must be submitted within August 31, 2025**, so that project discussion is carried out within September 2025.

The technical project report must be sent as pdf or doc document to Prof. Ferretti (marco.ferretti@unipv.it) and to Dr. Santangelo (luigi.santangelo@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 will present online their work in a Zoom session that will be scheduled after delivery of the project. 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 **MPI** model is the framework of choice for this project. Candidates can adopt other programming frameworks (e.g. PThread, CUDA, etc.) only for additional, non-mandatory extra implementatios. For a CUDA implementation on the Google Cloud Platform, there may be limitations on the VM that can be instantiated.

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 theassistant.

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. Use of profiling tools is strongly suggested. 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 communication&synchronization strategy must be discussed. The **a-priori theoretical assessment** (through Amdahl's law) of the speed-up is **mandatory.**
3. **MPI parallel implementation**. The student must implement the strategy discussed in the previous point using C or C++. Python 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. 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.
5. **Performance and scalibility analysis**. The student should analyze the performance of his implementation in term of execution time, memory occupancy (if needed) and *speedup*. Besides the a-priori theoretical assessment (through Amdahl's law), an analysis of actual results is mandatory. A scalability analysis is mandatory: **strong** and **weak** scalability must be assessed on **Google Cloud Platform** must be used. Strong scalability implies carrying out a given fixed-size problem on increasing resources (clusters of increasing dimension), weak scalability implies "fixed-size problem per computing node" that is a problem that grows in size as more computing resource are added.
6. **The report must describe the individual contribution to the project of each student in the group**.

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 allowed only for marginal contributions in the project, and 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).

Data set used in the project must have ***realistic cardinalities***: this is true for strong scalability analysis and is of course intrinsical to weak scalabilty analysis. Project based on "toy" cases will be rejected. As an instance, images smaller than 512x512 pixels are not acceptable; for Information Retrieval subject, it is suggested to search public available corpora (in excess of 100 Wikepedia pages as a rough estimate).

Computer Vision:

1. Mathematical morphology (only within a complete application);
2. Optical flow
3. (Generalized) Hough Transform
4. 2D 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 only)
4. (any other finite element problems, to be discussed with the professors)
5. Optimized matrix multiplication and inversion (rectangular matrix mandatory for multiplication)
6. Sparse matrix multiplication and inversion
7. (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)

Information Retrieval

1. Parallel TF-IDF Calculation: Compute Term Frequency-Inverse Document Frequency (TF-IDF) values for a document corpus.
2. Parallel k-Means for Document Clustering: Cluster documents based on their feature vectors using k-means clustering.
3. Parallel Cosine Similarity for Document Matching: Compute pairwise cosine similarity between document vectors in a large corpus.

Other:

1. N-body problem
2. Substring matching with application to genomics/proteomics