System and Device Programming Projects

Device Programming Part (Prof. Quer)

A.Y. 2022-2023

Projects will undergo the following rules and restrictions:

- Projects are optional, and the exam is passed when the written test (both parts) is passed.
- Each student can take a project only once in his curriculum path. The project must be selected before taking any written examination tests. Each project must be chosen by groups of 2 or (at most) 3 students.
- Projects will be assigned to groups of students on a first-come-first-served basis.
- All projects taken in 2021-2022 must be completed and delivered before or, at the latest, during the examination session of February 2023. The beginning of a course in the A.Y. 2022-2023 will automatically erase all pending projects.
- The evaluation of each project will follow a short presentation given by the group of candidates during one examination session. To enroll for the presentation, each group must upload the project on the portal web page before the written test of that examination section. Students must register for the exam in that session even if they only discuss the project and do not take the written test.
- Marks may differ for different students within the same group, depending on the effort they put into the project and the final oral presentation. Projects may amend the final mark, adding a value ranging from -2.0 to +6.0 marks. Projects assigned by the instructors and **not completed** by the group will be evaluated (for all students within the group) with a final value of **-2.0**. Once the final mark for a project has been assigned, there is no time limit to its validity.

Projects will be assigned with the following protocol:

- Phase 1 (selection phase)
    - By **Friday the 12th of May**, each group of students willing to take a project must indicate two group works (first and second choice)
        - Before that deadline, there will be the opportunity to discuss, with the instructors, characteristics and details of all projects either in real-time or off-line

- The choices must be stated by uploading on the portal, at the page "Elaborati", a **single** file with **name**, **content**, and **description** equal to the following **unique** string:

  rn1_rn2_rn3_p1_p2

  Where:

  - **rn1**, **rn2**, and **rn3** are the register numbers of the two or (at most) three students forming the group. Register numbers must be reported in increasing order.
  - **p1** and **p2** are the works selected by the group (namely, first choice **p1**, and second choice **p2**), i.e., q1, q2, etc., for Quer's projects, and c1, c2, etc., for Cabodi's projects.

  Again, please notice that only one student for each group must upload this information.
- Two correct examples of this file name, content, and comment to insert on the "Elaborati" web page are the following:
  - 134567_146789_189345_q1_c2, the students of register numbers 134567, 146789, and 189345 selected Quer's project number 1 as a first choice and Cabodi's project number 2 as their second choice.
  - 176789_191352_c3_q4, the students of register numbers 176789 and 191352 selected Cabodi's project number 3 as a first choice and Quer's project number 4 as the second choice.
- Notice that only **one student for each group** must upload this file.

- Phase 2 (tentative assignment phase)
  - By **Friday, the 19th of May,** the instructors will publish a list of assignments on the portal web page, i.e., a list in which each group will be assigned a single work following a first-come-first-served algorithm and trying to maximize the student's preferences.
- Phase 3 (final assignment phase)
  - By **Friday, the 26th of May**, each group will have the possibility to accept or reject the final assignment and eventually, **but only in particular cases**, to select a different one:
    - Accepting the project will be the default; no action will be necessary.
    - Rejecting (or changing) the project will be possible during a further discussion (either on or off-line) with the instructors.

Project Q1: An IOT Application on Capping Devices

Project's summary

AROL S.p.A. manufactures complex types of machinery that fill and cap bottles and jars and produces capsules. 95% of components and applications are designed in-house. World leader in its sector, it has offices in Turin and other Italian cities and the United States, Mexico, Brazil, China, and India.
Similarly, to what happened in the "automotive" world a few years ago, AROL's activities require growing IT skills both for centralized data management (involving cloud and edge computing, databases, etc.), as for the development of efficient procedures for the control and verification of the behavior of machinery (involving issues relating to operating systems and system programming, web programming, machine learning, etc.).

AROL's main goal is to equipe packaging machines with hardware devices ("edge" devices) and software to transfer machinery data to company servers ("cloud" devices). AROL already has an internal Proof-of-Concept (POC) web application, developed during a previous thesis work, that enables the unified management of a vast "fleet" of machines working in the packaging sector, carrying out performance level monitoring using a customizable and dynamic dashboard.
The current application uses **React.js** for the front end, **Express.js** for the back end, and relies on a PostgreSQL RDBMS for application data storage and a **MongoDB NoSQL DB** for application sensor data storage. All the additional modules/functionalities proposed below will be developed inside this framework. Thus, the project not only strengthens programming abilities but also forces the candidates to work in an existing well-formed, and already-running application.

Problem Definition

AROL wants to expand this POC application to improve the existing functionalities by integrating additional modules. To this respect, at least the **four following projects** are feasible:

1. A user and a machinery management module. The primary purpose of these modules is to allow AROL employees to create and modify machinery configurations and allow AROL employees to manage the application user specifications. The module must enable users to assign types of machinery to a client/customer company and implement a permission

system. The security feature must allow individually assign permissions to each user and enforce an access restriction mechanism, ensuring that only users with the proper permissions can access protected system resources.

2. A fully configurable and dynamic synthetic data generation tool that can generate artificial machinery sensor data. The tool's operational parameters must be utterly configurable from an interactive user interface. Sequential and parallel solutions are possible depending on the efficiency of the simulation and its accuracy (e.g., how many devices are simulated at the same time). The tool must also be able to dynamically generate sensor data from a configurable sensor set and allow users to create and modify machinery configurations that can be used to configure the tool quickly. Synthetic data generation can be realized algorithmically or using AI/Machine Learning techniques. Different programming languages and frameworks, such as React for the front end (configuration panel) and Python for the back end (synthetic data generation), can be used.
3. A real-time notification module: during their operation, types of machinery generate a multitude of data and data codes. Some of them may violate security thresholds that have been preliminarily defined. The primary purpose of this module is to enable real-time notifications of any anomaly during operation. Sequential and parallel solutions are possible depending on the efficiency of the simulation and its accuracy (e.g., how many devices are surveilled simultaneously). Special care must be taken to provide an efficient solution limiting unnecessary and repetitive access to the database. The module must also store a history of the alarms triggered for each device. Future developments may include real-time support for a "chat." In this case, the main goal is to implement a real-time chat solution that allows effective and quick communication between AROL employees and AROL clients/customer companies.
4. For mobile fans, AROL forecast the use of a cross-platform **mobile** version of the current POC application. The main goal here is to translate the front end of the existing application from a desktop web application into a mobile application. This mobile version will also rely on the currently available backend infrastructure, and it will be implemented using a cross-platform framework of choice, such as React Native.

Each of the above projects (all named Q1, inside Q1 the selection if free) can be developed by one or more groups of two or three candidates (better three). They cover several system and device programming topics, involving knowledge and skills ranging from web design (front-end and back-end), manipulating potentially significant and distributed databases, to the mobile application and infrastructure security. For this reason, each project is well suited to be easily extended to an **MS thesis** work for interested developers.

The above projects may also include a visit to Canelli, the leading production site of AROL in Piemonte, to visit the production plan, discuss the project with the personnel, make the experience, possibly discuss the evolution of the work for an MS thesis, and for future temporary or stable job positions.

Required Background

General programming and parallel-programming skills. A knowledge of the mentioned applications **React.js, Express.js**, **PostgreSQL RDBMS**, **MongoDB NoSQL DB**.

Working Environment

The project will be followed by the academic coordinator and the student that developed the initial application). The AROL internal staff will be available for further specifications and advice.
The thesis can be carried out at home, at the Department of Control and Computer Science premises, or (partially) at the Turin office of AROL.

Constraints

The original application will be available on the AROL GitHub site.
Each group will deliver its results using the same approach and including:

- The source files.
- A README markdown including a short "user manual", i.e., a document describing how to compile and run the program, under which system, which API, etc.
- A DOCUMENTATION text file (written in Word, Latex, Mark-down, etc.) including a short "designer manual", i.e., a document including:
    - The technological stack
    - A system design diagram
    - All main design choices (what is the business logic, how the data structure has been organized, etc.)
- An OVERHEAD set (organized in PowerPoint or similar) to be used during the project discussion in the project evaluation phase.

Contact

Prof. Stefano Quer (stefano.quer@polito.it) and Ing. Mario Deda (mario.deda@studenti.polito.it).

References

All the material will be described and given directly to the group involved. Regular meetings are advised to check the project's status and solve possible problems.

Project Q2: Graph Partitioning

Project's summary

Many applications of computer science involve processing large graphs, and often these graphs need to be partitioned into pieces that do not overlap. Dividing a graph into p partitions is called p-way partitioning. This project requires implementing and comparing a sequential and a parallel version of a graph partitioning algorithm.

Problem Definition

We consider graphs $G = (V, E)$ with two weight functions W_1: $V \rightarrow R$ mapping vertices to real-valued weights and W_2: $E \rightarrow R$ mapping edges to real-valued weights.
A p-way partitioning of G is a division of G into p sub-graphs in which the vertices in each subset do not overlap and satisfy specific properties:

- The sum of the weights of the nodes in each subgraph is balanced.
- The sum of the weights of the edges crossing between subsets is minimized.

Partitioning a graph has many applications in WEB search (Page Rank), placement and routing of VLSI circuits, social networks, etc.
Following the indicated references, this project implies.

- Reading large benchmark graphs from long-term memory.
- Implementing at least one sequential version of a partitioning algorithm.
- Implementing at least one parallel version of a partitioning algorithm.
- Comparing the sequential and the parallel version of the algorithm.

The comparison must consider the computation time and the memory usage of the main phases (at least, graph loading and path computation) and compare the sequential version with the parallel one with an increasing number of threads (i.e., 1, 2, 3, 4, etc.) on graphs of increasing size and complexity.
Optional: Compare the result of the written tool with publicly available applications such as hMetis, or the one available in Python in scikit-learn.

Required Background

Background aspects are all covered within the following courses:

- Advanced problem-solving and programming classes (e.g., "Algorithm and Programming")
- "System and Device Programming" course, i.e., concurrent programming.

Working Environment

Students can work on their laptop or desktop. Each group can decide whether to develop the application under a Unix-like or a Windows system. The implementation can be done in C, C++, or C++ using the boost library (for GPU optimizations). Parallel algorithms are described on the papers reported in the reference section. These works are available in the reference directory.

Constraints

All projects must be delivered including the following material:

- The source C/C++ files.
- A README text file (written in plain ASCII) including a short "user manual", i.e., a document describing how to compile and run the program, under which system, which API, etc.
- A DOCUMENTATION text file (written in Word, Latex, Mark-down, etc.) including a short "designer manual", i.e., a document including:
    - All main design choices (how the reading part has been performed, how the data structure has been organized, how the parallelism has been designed, etc.)
    - The experimental evaluation of the tools, i.e., tables or graphics reporting a reasonable set of experimental results. The experimental evidence should include a comparison (in terms of memory and of elapsed time) between the original sequential version of the tool and the 2-3 parallel versions with different parallelization levels (i.e., with 1, 2, 4, 8, etc., threads) and different size of the input graph (up to millions of nodes).
- An OVERHEAD set (organized in PowerPoint or similar) to be used during the project discussion in the project evaluation phase.

Contact

Prof. Stefano Quer (stefano.quer@polito.it).

References

- See https://en.wikipedia.org/wiki/Graph_partition for a theoretical introduction on the topic.

- See references reported in the directory dedicated to this project for technical details.