departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Concurrency and Parallelism 2019-20

Project — Parallel Patterns with C and OpenMP (v1.0)

João Lourenço

March 29, 2020

Abstract

This document describes the project assignment for the course of Concurrency and Parallelism 2019-20. In a nutshell, you are asked to join form a group with another colleague and implement a set of parallel programming patterns using OpenMP.

1 Introduction

In the last weeks, we have been learning and discussing about Patterns for Parallel Programming and loop parallelization. In this project you are asked to collaborate with another colleague and implement a set of Patterns for Parallel Programming using OpenMP. You are supposed to implement **at least** the following patterns: `Map`, `Reduce`, `Scan`, `Pack`, `Gather`, `Scatter`, `Pipeline`, `Farm`, and to optimize your code to be as fast and as scalable as possible.

1.1 Working rules

The following working rules apply. As this list is not exhaustive, in case of doubt do not hesitate in asking me.

- The project work will be done in grops of 2 students (*unless explicitly authorized otherwise*).
- Each group shall clone the given main repository, create a new empty project in one the publicly available Git hosting sites (e.g., github, gitlab, bitbucket) and define this new project as the master remote. Remember that your new repository must be **private** and shared only among the two group members (and me, look for me by my email "`joao.lourenco@fct.unl.pt`" or by the ID "`joaomlourenco`"… it is probably me).
- The group must respect and use the "*Git Workflow*". Each member will have a local copy of the repository in his/her computer where individual work shall take place.
- The group members shall make extensive use of branching (and merging). One branch for each work-task.
- Each student will commit his/her own work into the group shared repository.

- Commit messages must describe clearly what was changed/added in that commit (and why). **Commit messages are very important. Take you time to write a good commit message!**
- Remember to push your commits to your group repository.
- Each project must have a `README.md` file with instructions how to compile and run the test programs.
- The project report must have the look and feel of a research paper, using the IEEE template for Computer Society Journals (`https://goo.gl/Xtjdh4`), with a maximum of 4 pages (optional work, acknowledgments and references may be on the 5th page).

1.2 Project grading aspects

The Project's grade [$\mathcal{P}$] will take into consideration the following aspects. The order is not relevant and the list is not exhaustive. In case of doubt, do not hesitate in asking.

- The project will be graded according to the following criteria:
  - Projects that fail to compile and execute with the following procedure will receive the final grade of $\mathcal{P} = 0$ points, i.e., the following (or something very similar) must work to run all the tests

    ```
    clone <your_repository>
    cd  <your_repository>
    make
    ./main [OPTIONS] NUMBER
    ```
  - The quality of the work as perceived from project report;
  - The quality of the text in the project report (well organized, complete, the text/message is clear, no misspellings, etc)
  - The project's code look and feel;
  - A perfect project that implements all (and only) the above listed patterns will have a top grande of 17 (over 20) points.
- Some points are reserved for complementary work/achievements, such as:
  - Some points will be given to the implementation of one or more patterns (unlisted above). Please discuss with me the *signature/interface* of the patterns you want to implement.
  - Some points will be given to the implementation of more unit/integration functions. Extra points if they are shared publicly with the colleagues and used and acknowledged by them in their reports.

1.3 Student grading aspects

The Individual Students's grade [$\mathcal{S}$] will take into consideration the following aspects. The order is not relevant and the list is not exhaustive. In case of doubt, do not hesitate in asking.

- Relevance of the student's commit messages.
- Relevance of the code committed by the student.
- Students that do not exhibit evidences of working on the project (e.g., with no commits, or with nearly empty/non-meaningful commits, or with insufficient/unclear commit messages) will receive a grade $\mathcal{S} = 0$ fail the course.

1.4 Student/Project Final grading rule

- Formula: $\mathcal{G} = 0.7 * \mathcal{P} + 0.3 * \mathcal{S}$

2 Project Description

In this lab work you are given a working sequential version of the parallel patterns using the C programming language. You are asked to study the given code and create an optimized (parallel) version of the code using OpenMP.

2.1 Given Version

You may replicate a working sequential version of the project with

```
gitclonehttps://bitbucket.org/joaomlourenco/parallel_patterns_seq.git
```

This repository contains two directories/folders: `code` and `doc`. The former contains the base source code, and the latter will contain your Project Report as a PDF file. Please name your report as `report_AAAAA_BBBBB.pdf`, where `AAAAA` and `BBBBB` are the numbers of the group members sorted in increasing order (lowest to highest).

After cloning the repository, create a new empty repository in a hosting service of your choice and name it

```
cp2019-20_project_AAAAA_BBBBB.pdf
```

where `AAAAA` and `BBBBB` are the numbers of the group members sorted in increasing order (lowest to highest). Define your new repository as the new remote master and push the code into it.

2.2 Code Structure

debug.c debug.h main.c patterns.c patterns.h unit.c unit.h

The project include the following source files:

Files	**Description**
`debug.c debug.h`	Functions for printing the contents of the array(s), useful for debugging (activated with the option "-d").
`patterns.c patterns.h`	The patterns to be implemented. The ".c" file contains a sequential implementation of the patterns.
`unit.c unit.h`	Functions for unit testing of each pattern.
`main.c`	The main program.

2.3 Reporting

The project report must have the look and feel of a research paper, using the IEEE template for Computer Society Journals (`https://goo.gl/Xtjdh4`), with a maximum of 4 pages (optional work description and evaluation, contributions, acknowledgments and references may be on the 5th page).

In the extra page please describe:

- Your additional work (extra patterns or other relevant work).
- How did the two group members split the work? Please give an estimative of the workload for each students, e.g., Workload distribution: X 75% and Y 25% (I expect a workload distribution as even as possible). If you cannot get an agreement on the workload distribution, present the perspective from both of you.
- Acknowledge colleagues who contributed to your project and explain their contribution. This may include direct support on coding/debugging/testing/... or indirect support by way of Piazza or Slack or some other channel.
- Listing the sources of information (References) is a must for the credibility of your own work.

2.4 Work plan

Your job is to make an optimized parallel version (using OpenMP) of all the patterns listed in the files `patterns.c/patterns.h`!. Optionally, may also implement one or more extra patterns. If you do, please discuss with me the interface/signature for these patterns.

You may follow these steps:

1. Setup you project environment as described above.
2. Compile the given version. Study the source code and understand how it works.
3. Discuss with your colleagues how to split the work.
4. Implement a parallel version of each pattern, each one in its own branch (remember the *git workflow*).
5. Compile and run the tests and confirm the results.
6. For each pattern, measure its perfocrmance/scalability/scaled scalability. You may experiment with different numbers of processors (by setting the environment variable `OMP_NUM_THREADS`).
7. Optimize the given code.
8. Go back to item 5. until satisfied.
9. Write the report. Revise the report. Please put your report in the `doc` directory and name it as `report_AAAAA_BBBBB.pdf`, where `AAAAA` and `BBBBB` are the numbers of the group members sorted in increasing order (lowest to highest).
10. *Optional:* implement and optimize some more parallel patterns.

11. *Optional:* implement and share some more tests (unit or integration tests).
12. *Optional:* complete the report and revise again.

Please remember to use the *git workflow* and commit regularly your changes, and please always write meaningful commit messages.

Please remember to reserve LOTS OF TIME to testing and evaluating your project and writing your report. Your grade will depend mostly on this!!

3 Questions/Discussion

Please ask your questions using the Piazza or Slack system. Either public (if possible) or private (if really necessary).