Work Assignment
Phase 2

Learning outcomes

This assignment phase aims to explore shared memory parallelism (OpenMP-based) to improve the overall execution time.

Introduction

Students are requested to improve the execution time by exploring shared-memory parallelism with OpenMP directives. The starting point for this second phase is the improved version of phase 1 with two changes:

1. Data size should increase from **SIZE=42 to SIZE=84** (8 fold-increase in data size).
2. A new solver, more efficient and suitable for parallel execution, must be used. This solver is a red-back implementation with convergence check (see attached file for the new solver code).

In this assignment, students should follow a structured methodology to develop parallel programs, covering the following steps:

(i) **identify** the application hot-spots (code blocks with high computation time);

(ii) **analyse** and present the alternatives to explore parallelism within the hot-spots identified in **(i)**;

(iii) **select** an approach to explore parallelism, justified by a scalability analysis;

(iv) **implement** and **optimise** the approach on the SeARCH cluster (compute node(s) on `cpar` queue);

(v) **measure** and **discuss** the performance of the proposed solution.

Groups, submission format and dates

The work assignment should be performed by the same student's groups from previous phase.

Submission rules are the same with minor changes (**in bold**) in order to allow performance evaluation:

- **the size of the problem in this phase duplicates (SIZE=84)**;
- the work must be submitted through the e-learning platform, compressed into a zip file that, when unzipped, should generate a base directory whose name is the groups elements, e.g., `a43000_pg54000`. It should include:
  - a **1-page PDF report** with **all** relevant information using the same IEEE template (in https://www.ieee.org/conferences/publishing/templates.html); **longer reports are penalized**; annexes can be added beyond these 1 pages , but these might be read **or not** by the evaluator;
  - a subdirectory with all source code (please, do not submit executables, **or other files**);
  - **a new `Makefile` is requested in the base directory, that generates and runs the executable** (see example in annex).

Submission deadline: 23:59, 19-Nov-24.

The defence of this assignment will be performed during the oral presentation of the WA-Phase 3 (in Jan'25).

Evaluation

The evaluation of this work will consider:

(i) the selected **approach to explore parallelism**, its **implementation** with OpenMP and **code legibility (65%);**

(ii) the **execution time** of the parallel implementation; the number of PUs is specified in the `Makefile` **(15%);**

(iii) the **report quality,** including **strong scalability analysis**, profiling and other models and metrics that explain the results **(20%).**

Annex - A simple ***`Makefile`***

The job submission must include a `Makefile` that generates one executable `fluid_sim` in the base directory. All source files should be placed in a subdirectory (e.g. `src`).

The program will be run for testing using `make runseq` for the sequential version and `make runpar` for the parallel version. Any necessary parameters, environment variables, etc., should be specified to minimize the program's execution time.

Suggestion: To simplify the process, you can use the same or different executables. Ensure that the sequential version runs on a single thread and that the parallel version uses the number of threads that minimizes execution time.

```
CPP = g++ -Wall -fopenmp
SRCS1 = main.cpp fluid_solver.cpp EventManager.cpp

all:  phase2

phase2:
       $(CPP) $(SRCS) -o fluid_sim

clean:
       @echo Cleaning up...
       @rm fluid
       @echo Done.


runseq:
       ./fluid_sim_seq

runpar:
       ./fluid_sim
```