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 computation time of the code developed in phase 1, exploring shared memory parallelism with OpenMP directives. In this assignment students should follow a **methodology to develop parallel programs**, with 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 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 number of atoms should be set to 5000**;
- 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 2-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 2 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, 27-Nov-23.

The defence of this assignment will be performed during the oral presentation of the WA-Phase 3 (in Jan'24).

Evaluation

The evaluation of this work will consider:

(i) the selected **approach to explore parallelisms**, its **implementation** with OpenMP and **code legibility (60%);**

(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 **(25%).**

Annex - A simple `Makefile`

The job submission must include a `Makefile` that generates two executables, `MDseq.exe` and `MDpar.exe` in the base directory. All source files should be placed in a subdirectory (e.g. `src`).
In the example, the program can be run with `make runseq` for the sequential execution, and `make runpar` for the parallel execution; `make runpar` should run the program with the number of threads that maximises its performance (i.e. that minimises its execution time).

```
CC       = gcc
SRC      = src/
CFLAGS   = # select optimization flags (e.g., O2 or O3)

.DEFAULT_GOAL = all

all: MDseq.exe MDpar.exe

MDseq.exe: $(SRC)/MDseq.cpp
	module load gcc/11.2.0;
	$(CC) $(CFLAGS) $(SRC)MDseq.cpp -lm -o MDseq.exe

MDpar.exe: $(SRC)/MDpar.cpp
	module load gcc/11.2.0;
	$(CC) $(CFLAGS) $(SRC)MDpar.cpp -lm -fopenmp -o MDpar.exe

clean:
	rm ./MD*.exe

runseq:
	./MDseq.exe < inputdata.txt

runpar:
	./MDpar.exe < inputdata.txt
```