2SC7695 – Low cost optimization of acoustic wave propagation code performance

Instructors: Stephane Vialle
Department: DOMINANTE - MATHÉMATIQUES, DATA SCIENCES, DOMINANTE - INFORMATIQUE ET NUMÉRIQUE
Language of instruction: ANGLAIS
Campus: CAMPUS DE PARIS - SACLAY
Workload (HEE): 80
On-site hours (HPE): 48,00

Description

Project topic in partnership with INTEL.

Regardless of the type of application running on parallel machines in an HPC (high performance computing) environment, and regardless of their level of efficiency (in terms of performance or energy footprint), we can easily see that the impact of input parameters is generally not negligible. In particular, we can act on :

- the parameters of the parallel algorithm used, such as the size and shape of the domains performing a sub-problem partitioning,
- HPC implementation parameters controlling for example *cache blocking* to reduce data access times,
- parallel application deployment parameters, controlling the process-MPI vs. threads-OpenMP distribution and the placement of these computational tasks to efficiently occupy CPU cores,
- parameters controlling the thread scheduling policy (usually controlled through environment variables).

These different input parameters allow to improve the execution of HPC code on a computing platform. But it remains extremely difficult to understand exactly how the application behaves in the processor architecture and even if we could make the application more efficient by modifying the source code, the dependency on test cases and the execution environment would still be preponderant. In this context where the parameter space can be of important dimension, the use of optimization algorithms appears fundamental to converge towards a global (or at least local) minimum of a cost function expressing the execution time and the energy footprint. We will therefore use optimization methods and algorithms to optimize the operation of an HPC calculation code.

However, each execution of a test case of an HPC application can be long, even on a parallel machine. The parameter space is large and an optimization algorithm can require tens of thousands of executions of the application (or even more). What is achievable on the scale of a computing kernel remains unbearable on the scale of a complete application.

It is therefore necessary to choose optimization methods that do not consume too many experiments, and then to optimize their use to reduce the footprint of the targeted HPC code, without this pre-study consuming too many computing resources! This amounts to **"looking for the least expensive optimization method"** to find an optimum between convergence speed (of the optimization algorithm) and the quality of the minimization of execution time and energy footprint (of the targeted HPC application).

Technical details of the system :

We will start this project from :

- a high performance computing code (HPC code) that simulates the 3D acoustic wave equation in a homogeneous isotropic finite-difference medium, and runs on multi-core PC clusters (in MPI + OpenMP),
- a genetic algorithm capable of iterating on many parameters of the HPC code such as: size and shape of domains and *cache blocking*, compilation flags, environment variables, number and placement of threads..., but which remains sequential and limited to a single machine.

The genetic algorithm thus calls successively the acoustic wave simulation code in different configurations, and can only call it in a multithreaded version on one machine (not in a distributed version on a cluster). This allows nevertheless to search for the optimal configuration on each PC, and then to apply it on each PC when running on a cluster of PCs.

The *finite differences* being quite explicit in terms of the number of floating operations, we can easily count the number of points processed or the number of floating operations performed during an execution, and deduce a processing speed in Giga points/s or GFlops/s. We can then try to minimize the simulation time (or maximize the processing speed). We can also dedicate and collect some hardware counters to obtain the exact power consumption of the processor and memory, to study in detail the energy impact of our optimizations.

But a genetic algorithm is a population-based optimization method that is computationally resource-intensive, and the implementation provided uses only one PC. The search for an optimal configuration of the code can therefore be very long and prohibitive. To overcome this limitation we will adopt two approaches:

- Firstly, we will identify optimization methods that are likely to converge in relatively few tests towards an efficient configuration of the HPC code (i.e. leading to fast and low power consumption executions). We will be able to study optimization methods with trajectories (Hill Climbing, Simulated Annealing, Tabu search..., with or without gluttonous behaviour), or other methods with a population less greedy than genetic algorithms (ant colonies...). A very good solution will thus be sought in a rather long time, which would then allow to execute many simulations of very optimized acoustic wave propagations.

- One will then try to experiment an optimization method, or a variant of the one previously experimented, which would converge "very quickly" towards a solution of only "good enough" quality. Such (very fast) optimization could then be integrated into the application in the form of dynamic pre-processing, at the beginning or during execution on parallel machines, and constitute a self-optimization mechanism for the application.

Quarter number
ST7

Prerequisites (in terms of CS courses)
First year courses:

- SG1 common course "Systèmes d'Information et Programmation" (1CC1000)
- ST2 common course "Algorithmique et complexité" (1CC2000)

Courses of the ST:

- ST7 common course "Optimisation" (2CC3000)
- ST7 specific course "Méthodes et algorithmes parallèles pour l’optimisation" (2SC7610)

Others prerequisites:

- Parts of common course "CIP - Convergence, Intégration et Probabilités" (1SL1000)
- Parts of common course "EDP - Equations aux dérivées partielles" (1SL1500)
- Knowledge of linear algebra will also be needed

Syllabus

Main steps of the study :

- Presentation of the provided computing kernel, complementary courses on the methodologies of characterization and acceleration of HPC code (*hardware counters, roofline modeling, NUMA placement, vectorization...*), handling of the remote computing resources of the CentraleSupélec Teaching Data Center with experimentation of the codes provided by INTEL.
- Identification of promising optimization methods for the problem, and not launching too many HPC simulations of acoustic waves. The development of hybrid methods could be considered.
- Development of a 1st solution in sequential Python with call of the simulation C code provided for the study.
- Development of a first optimization-simulation campaign on multi-core machines, with management of a weekly quota of calculation hours. Identification of a solution that best reduces the footprint of the HPC acoustic wave simulation code.
- Development and experimentation of a 2nd solution, allowing to search "very quickly" for a "good enough" quality configuration, in order to integrate this search into the simulation application at the pre-processing stage.
- Development of a second optimization-simulation campaign on multi-core machines, with management of a weekly quota of calculation hours.
- The study will end with a report and an oral presentation to evaluate:
    - the investigative approach adopted,
    - the quality of the solution found: in terms of the speed of convergence of each optimization algorithm tested, and the computation time and energy consumption of the optimized parallel simulation,
    - the management of the quota of calculation resources that will have taken place during the project.

Rmk: The different groups of students will implement different optimization methods.

Class components (lecture, labs, etc.)

Part 1 (40HEE) :

- Steps 1 and 2: Complementary lectures on HPC code optimizations and on the configuration parameters of the simulation code provided, handling of the remote computing resources of the

Teaching Data Center with experimentation of the initial solution, and identification of two promising optimization methods.

- Step 3: sequential implementation in Python of a first optimization method calling parallel simulation code and HPC, first optimization-simulation campaign on a multi-core computing server.
- Step 4: execution of an optimization-simulation campaign on parallel machines, with the management of a quota of hours. Analysis of the obtained performances.

- Intermediate report, and presentation of the progress and the work planned in the 2nd part.

Part 2 - *final sprint* (40HEE):

- Step 5: identification of a method to search "very quickly" for a "fairly good" configuration. New implementation in Python.
- Step 6: new optimization-simulation campaign on parallel machines, with the management of a quota of hours. Analysis of the obtained performances.

- Final report and full oral presentation.

Grading

This project will be evaluated by a midterm talk at the end of part 1 (40HEE), and by a final talk at the end of part 2 (*final sprint* 40HEE). Talks will be done by the entire team, but will lead to individual marks in case of strongly heterogeneous teams.Each talk evaluation will consider the overall quality of the talk, of the slides and of the progress summary. Each talk mark will be 50% of the total mark.

Resources

Teaching Staff:

- **H. Talbot** (CentraleSupélec & CVN) et **S. Vialle** (CentraleSupélec & LISN)
- **Ph. Thierry** (INTEL)

Workplace and computing resources:

- Students will work at CentraleSupelec, in a classroom with electrical outlets and reliable wifi Internet access.
- Students will use their laptops to connect to remote PC clusters at Data Center for Education of CentraleSupelec

- Final oral exam will take place at CentraleSupelec the last afternoon of the project.

Learning outcomes covered on the course

At the end of this project, students will be able to:

- **Learning Outcome 0 (AA0):** to identify the parameters impacting the execution of a parallel code, and to configure its execution,
- **Learning Outcome 1 (AA1):** to choose and configure optimization methods converging with a limited number of experiments,
- **Learning Outcome 2 (AA2):** to develop a sequential Python code, calling parallel codes on parallel architectures,
- **Learning Outcome 3 (AA3)**: to deploy intensive simulations on remote computing resources,
- **Learning Outcome 4 (AA4)**: to identify the limits of the study according to the available computational resources
- **Learning Outcome 5 (AA5):** to manage a quota of calculation resources during an intensive calculation campaign.

Description of the skills acquired at the end of the course

- C4: Have a sense of value creation for his company and his customers
- C7: Know how to convince
- C8: Lead a project, a team