Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Bachelor Degree in Informatics Engineering
Computer Engineering Specialization
Degree Final Project
Thesis management course

Performance analysis and optimization of a combustion simulation

First assignment: Context and scope of the project

Author
Guillem Ramírez Miranda

Director
Marta Garcia Gasulla

Co-director
David Vicente Dorca

Tutor
Julian David Morillo Pozo

GEP Tutor
Joan Sardà Ferrer

September 29, 2020

Contents

List of Figures

1 Context

1.1 Introduction

Computational science is a growing field that allows researchers to predict the behaviour of systems. For example, a physicist can write a program to predict the trajectory of a sphere instead of calculating it by hand.

What if we bring this further? We can try to optimize the aero-dynamism of a plane to save fuel, the behaviour of the human body cells to a recently designed drug or even given a genome figure out the risk of cancer.

Computing these are at a high cost. It is not conceivable to solve the problems on our laptops as the number of calculations and data required to process is overwhelming. To solve these tasks, we need supercomputers.

Supercomputers are machines with a huge compute power oriented to scientific and technique jobs. These machines are many small machines interconnected. Developers must write their code in a manner that the program is capable of run in parallel.

Writing efficient parallel programs is a difficult task and even more so if the developer's specialization is not computer science. In this work, we will study and improve Alya [1], a real use case application. In particular, we will focus on the module in charge of computing combustion processes.

This work is under an Educational Cooperation Agreement between the Facultat d'Informàtica de Barcelona and the Barcelona Supercomputing Center.

The study has been developed within a Curricular internship in the Barcelona Supercomputing Center precisely in the High-Level Support Team (HLST) inside the Operations department and developed under a collaboration with the Best Practices for Performance and Programmability group. This work is impulsed by the Performance Optimization and Productivity [2] (POP) Center of Excellence under a Performance Analysis request.

1.2 Terms and concepts

In the following sections you will find basic concepts to better understand the project.

1.2.1 High-performance computing

High-performance computing (HPC) refers to the act of grouping compute power to do massive and complex computations and data processing. Nowadays HPC is associated to supercomputers.

1.2.2 Supercomputer

A supercomputer is a machine designed for HPC. The basic structure of a supercomputer consists of multiple computing nodes interconnected. Usually, a node is a shared memory system which can run programs by itself and may have add-ons like accelerators.

1.2.3 Parallel Programming model

Parallel programming models [3] add an abstraction to parallel computer architectures. It defines constructs that allow to operate with the parallel machine performing different actions. For example, sending and receiving messages between processes, reading and writing to the shared memory or spawning tasks in form of threads, all depending on the kind of programming model.

The two main types of parallel programming models used are:

- **Shared memory:** Parallel processes share a global memory address space that they can use for sharing data, synchronization. This type of model is optimal for Multi-Processor systems and cannot be used when connecting multiple machines (i.e., among multiple nodes) as they do not share memory.
- **Message passing:** Parallel processes are independent and they use messages to synchronize and share data. This type of model is ideal to exploit the interconnection network between nodes although it can be used within a single node.

1.2.4 Message Passing Interface (MPI)

The message passing interface [4] is a parallel programming model that defines a standard interface for communications between processes. This allows applications to run in numerous nodes. The processes use the standard interface to send messages to each other using the underlying interconnection network of the machine. This interface allows, among other things, process synchronization, data-sharing via asynchronous and synchronous messages, gathering and scattering data, parallel input-output (I/0).

1.2.5 MareNostrum 4

MareNostrum 4 [5] is a supercomputer managed by the Barcelona Supercomputing Center and managed by the Operations department. It is divided in two blocks, the general purpose block which represents the majority of the computing power and the emergent technologies block that aims to test new technologies. Figure 1 shows a picture of the supercomputer located in the "Torre Girona" chapel.

In this work we will be running the application on the general purpose block. The general purpose block consists of 3456 nodes each node containing 2 Intel Xeon 8160 with 24 cores each running at 2.1 GHz. The interconnection network consists of an Intel Omni-Path full-fat tree at 100 Gbps.

Figure 1: MareNostrum 4 machine. By Gemmaribasmaspoch - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=61308843

1.2.6 Alya

Alya [1] is the application we will be studying. It is code oriented to high-performance computational mechanics that aims to solve engineering coupled problems. In other words, Alya consists of multiple modules that can be joined to solve a specific problem. For this work, we will focalise the analysis into a new Alya module devoted to simulate combustion processes. We will work with a real industry use case in the combustion industry.

Alya supports a wide variety of programming models but the module we are given to analyse only supports MPI. It is contemplated to add support for other programming models if we find out that doing so improves the performance.

1.2.7 Performance Analysis

Running real use case scientific applications in supercomputer environments is crucial to identify flaws that deplete its performance when running on several machines.

Performance analysis refers to the process of gathering data from application executions, study the data and finally identify bottlenecks of the program.

In the state-of-the-art performance analysis, there is a wide variety of methodologies. However, this work is under the POP Center of Excellence resultantly we chose its methodology [6]. The application of the methodology is tricky, [7] shows the application of the methodology.

1.2.8 BSC tools

As said, for a performance analysis we need data about the execution of programs and tools to visualize the data to get an insight of the bottlenecks.

The main BSC tools we are going to use are:

- **Extrae** [8]: This is the central tool of all the collection of tools. It allows the user to hook up Extrae to the application and gather data during the execution. We can trace events such as the MPI calls, PAPI [9] counters, other programming models events and even manually instrument the code using Extrae API. All the data extracted is dumped into a file we call trace.
- **Basic Analysis** is a set of scripts that using existing BSC tools automatically computes the POP methodology metrics.
- **Paraver** [10] [11] is a powerful desktop application that allows to load up traces and study them. The two main ways to study a trace are using a histogram or a timeline. Histograms allow to study the tendency of some certain counter in the execution. Figure 2 shows an example histogram where the *y-axis* shows MPI-processes and the *x-axis* shows different frequency values. The color represents how frequent is that frequency value during the execution, green is not frequent and blue is frequent. Time-lines allow to analyse the behaviour of the code in certain regions. For example, the pattern of MPI calls, the memory access patterns, the executions of tasks on a shared-memory programming model. Figure 3 shows a time-line where *y-axis* show MPI-processes and *x-axis* show time. The color of a process at a certain time show what the process is doing. Blue means the process is computing something useful while black means the process is idle, communicating or in a synchronization. Looking at traces in the Paraver UI it is not the best visual appealing experience. For the sake of comprehensiveness we will use tracedraw [12] package for drawing more visual appealing traces. Figure 4 shows the imitation of Figure 3 with tracedraw.

1.3 Problem to be resolved

This project is motivated by the Propulsion Technologies BSC group applying to a POP Performance Analysis assessment.

Figure 2: Sample Paraver Histogram

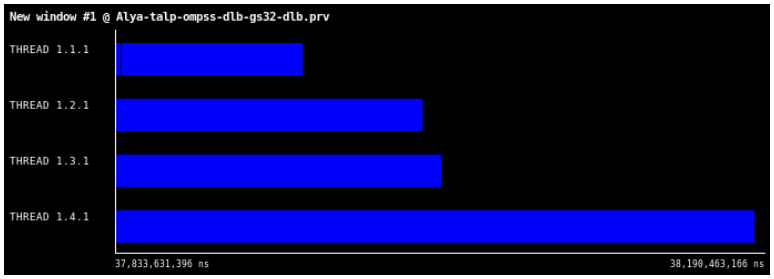

Figure 3: Sample Paraver trace

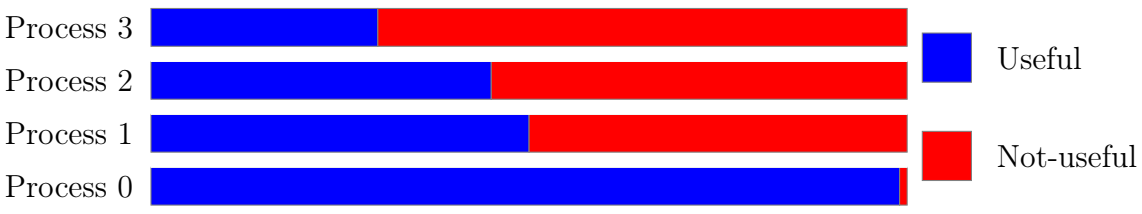

Figure 4: Sample trace representation with tracedraw

The problem to solve is that the combustion module they programmed is slow. We are asked to find out why and if possible, apply optimizations and modifications to the code to improve its performance and efficiency. In Section 3.1 we will discuss the solutions to the problem by defining our objectives.

1.4 Stakeholders

The following groups will benefit from the work.

1.4.1 Propulsion Technologies BSC group

This is the principal stakeholder as it is the developer of the module and the applicant for the analysis.

1.4.2 Scientific community

This beneficiary is also benefited from the work since the results and conclusions of the study are public and open to the scientific community. More in detail, the combustion community will be benefited as the results of the study and the possible optimizations will be shared and used in the newly approved Center of Excellence in Combustion (CoEC)[1].

1.4.3 Society

POP is an European Center of Excellence, this means that the European Union states that the work done under the Center of Excellence (POP) will have a positive impact to the European Society.

In the case of optimizing the combustion module it will impact because combustion researchers will be able to simulate faster which can lead to new discoveries on more energy efficient and pollute less combustions.

Optimizing a code also implies that the application users will use less time the machine which means that they will use less energy for their research.

[1] https://www.bsc.es/news/bsc-news/the-ec-approves-two-new-centres-excellence-high-performance-computing-applications-led-bsc

2 Justification

For studying Alya as said we will use the POP methodology because:

- The study is a POP service.
- It is a methodology promoted by an European Center of Excellence, meaning it is in the state-of-the-art of performance analysis methods and it is widely use in the academia.
- It is a well-known method and has been in part developed inside BSC.
- It is the methodology I have learnt during my formation period in BSC.

We will use the BSC toolset for extracting data to apply the POP methodology because:

- They are well-integrated with the POP methodology.
- The tools are well-known and developed in BSC. In the situation that we find a bug or we have a problem using the tools, we can directly contact the application developers as they are co-workers.
- It is the toolset I have learnt during my formation period in BSC and during the computer engineering degree at PAR[2] subject.

3 Scope

In this section you will find a definition of the objectives of the projects and the identification of possible risks.

3.1 Objectives

The project is divided into two main objectives.

- Perform a rigorous Performance Analysis following the POP methodology and identify bottlenecks provided by a real and industry use case in combustion state-of-the-art.
- Knowing the bottlenecks, propose, implement and test optimizations to the code that improve the performance and efficiency of the code..

Other sub-objectives are:

- Learn and gain experience on the performance analysis and optimization field.

[2]Parallelism: https://www.fib.upc.edu/en/studies/bachelors-degrees/bachelor-degree-informatics-engineering/curriculum/syllabus/PAR

- Document and explain Performance Analysis finding to developers and scientists responsible of the code that are not necessary experts in the performance analysis field.

3.2 Potential obstacles and risks

The following potential obstacles and risks have been identified:

- **MareNostrum4 not available** A risk we must take into account is that MarenNostrum4 gets stopped, by a maintenance, overheating issues or other errors. If this happens the possibility of moving the study to another similar machine is contemplated.
- **Misinterpretation of a metric during the analysis:** This risk can lead the study to incorrect conclusions. However, by daily feedback with the director of the work and bi-monthly group meetings, we minimize this risk.
- **Incorrect runs:** This is a common risk that occurs when performing different tests of an application forgetting to change a parameter and leading to an incorrect run. These are usually easy to detect as numbers obtained does not match the expected. Still, if the risk is not detected the feedback received during the development should avoid this risk.
- **Machine noise:** Large supercomputer machines use complex environments that can lead to system noise. System noise refers to abnormal events that interrupt the application making counters and timings to have erratic values. Take an appropriate number of samples and doing a correct statistical analysis will avoid this risk.

4 Methodology and rigour

The methodology relays in constant feedback between Marta Garcia (the project director) and me. As tasks are sequential, once a task is done the results are shared with the director and conclusions and definition of the next task is discussed. The feedback and the discussion are done via email, personal meetings or group meetings.

This work has been developed during the global pandemic of COVID-19 so telematic meetings are contemplated.

Another important aspect to cover is "Version Control" of the project. A git repository has been created for keeping track of the POP deliverables. Another git repository is created for tracking the changes to the code.

For storing the data we will use a "google spreadsheet" data sheet with the corresponding information about the runs and the tests performed. However, trace files and run logs are kept inside the MareNostrum 4 shared storage.

An issue page in the internal gitlab of the Best Practices for Performance and Programmability group has been created. In this page I will be posting updates about the work so the members of the team can be updated about the state of the project.

References

[1] M. Vazquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Aris, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, and J. M. Cela, "Alya: Towards exascale for engineering simulation codes," 2014.

[2] "Performance optimisation and productivity," POP-coe, accesed on 26/09/2020. [Online]. Available: https://pop-coe.eu/

[3] C. Kessler and J. Keller, "Models for parallel computing: Review and perspectives," *Mitteilungen - Gesellschaft für Informatik e.V., Parallel-Algorithmen und Rechnerstrukturen*, vol. 24, p. 13–29, 2007. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-40734

[4] D. W. Walker and J. J. Dongarra, "Mpi: a standard message passing interface," *Supercomputer*, vol. 12, pp. 56–68, 1996.

[5] "Marenostrum 4 techincal information," Barcelona Supercomputing Center, accessed on 26/09/2020. [Online]. Available: https://www.bsc.es/marenostrum/marenostrum/technical-information

[6] M. Wagner, S. Mohr, J. Giménez, and J. Labarta, "A structured approach to performance analysis," in *International Workshop on Parallel Tools for High Performance Computing*. Springer, 2017, pp. 1–15.

[7] F. Banchelli, K. Peiro, A. Querol, G. Ramirez-Gargallo, G. Ramirez-Miranda, J. Vinyals, P. Vizcaino, M. Garcia-Gasulla, and F. Mantovani, "Performance study of hpc applications on an arm-based cluster using a generic efficiency model," in *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE, 2020, pp. 167–174.

[8] "Bsc tools extrae website," Barcelona Supercomputing Center, accessed on 26/09/2020. [Online]. Available: https://tools.bsc.es/extrae

[9] D. Terpstra, H. Jagode, H. You, and J. Dongarra, "Collecting performance data with papi-c," in *Tools for High Performance Computing 2009*. Springer, 2010, pp. 157–173.

[10] V. Pillet, J. Labarta, T. Cortes, and S. Girona, "Paraver: A tool to visualize and analyze parallel code," in *Proceedings of WoTUG-18: transputer and occam developments*, vol. 44, no. 1. Citeseer, 1995, pp. 17–31.

[11] "Bsc tools paraver website," Barcelona Supercomputing Center, accessed on 26/09/2020. [Online]. Available: https://tools.bsc.es/paraver

[12] G. Ramirez-Miranda, "Tracedraw repository," accessed on 26/09/2020. [Online]. Available: https://github.com/teleportex/tracedraw.sty