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

Thesis Management

Author
Guillem Ramírez Miranda

Director
Marta Garcia Gasulla

Co-director
David Vicente Dorca

Tutor
Julian David Morillo Pozo

GEP Tutor
Joan Sardà Ferrer

15th December 2020

Contents

List of Figures

List of Tables

Chapter 1

Context and scope of the project.

1.1 Context

1.1.1 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.

1.1.2 Terms and concepts

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

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.

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.

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.

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).

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 1.1 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 1.1: MareNostrum 4 machine. By Gemmaribasmaspoch, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=61308843

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.

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.

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 1.2 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 1.3 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 1.4 shows the imitation of Figure 1.3 with tracedraw.

Figure 1.2: Sample Paraver Histogram. Own compilation.

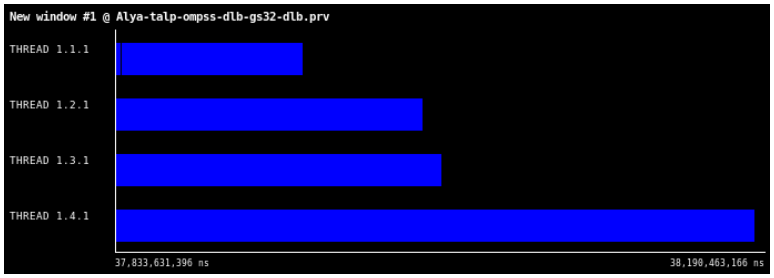

Figure 1.3: Sample Paraver trace. Own compilation

1.1.3 Problem to be resolved

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

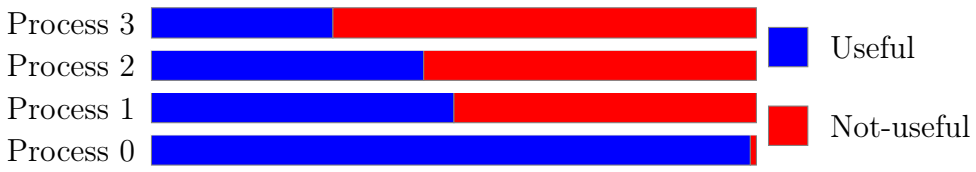

Figure 1.4: Sample trace representation with tracedraw. Own compilation

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 1.3.1 we will discuss the solutions to the problem by defining our objectives.

1.1.4 Stakeholders

The following groups will benefit from the work.

Propulsion Technologies BSC group

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

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].

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

1.2 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.

1.3 Scope

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

1.3.1 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.

1.3.2 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.

1.4 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[3] 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[4] 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 and we can keep track of it.

[3]Google office suite: https://www.google.com/intl/es/drive/
[4]Gitlab Community Edition: https://about.gitlab.com/install/ce-or-ee/

Chapter 2

Time planning

2.1 Description of tasks

2.1.1 Project management

Project management is a compulsory task for any project.

Meetings

Regular meetings are contemplated to keep track of the progress of tasks and to make decisions. It is planned to do approximately 2.5 hour of meetings per week. Time estimated: 45h

Context and scope of the project

Definition of context and scope of the project. Time estimated: 30h

Time Planning

Definition of the tasks and the time planning of the project. Time estimated 10h

Budget and sustainability

Compute the monetary costs of the project. Definition of the sustainability report. Time estimated 10h

Integration and final document

Review, fix and integrate the last 3 tasks into a final document that groups all the information related to the project management. Time estimated 15h

2.1.2 Hands-on

These tasks are intended to get introduced to the application analysed and ensure we are analysing the correct thing.

Contact with application developers

We will meet with the application developers where they will introduce us to the application module. The following points will be worked:

- Basic explanation of the science between the module and the input case.
- Localization of the code and the input case.
- Explanation of the compilation process of the code.
- Explanation of how to run the code.
- Download the code and the input case
- Compilation of the code
- First runs of the code.

Time estimated: 10h

Study the compilation process

It is very important to understand the compilation process and the different options that the build chain allows. This is a simple task but helps a lot in minimizing the possible errors we can encounter adding or modifying the code in the future.

Time estimated: 15h

Run and test the application

Every user in a HPC machine has its own environment. As the code is in development process and I may need to modify it an make my own installations thus some things from the application may differ. It is a critical task to perform some runs and verify with the application developers that we are not having errors, we are running the correct thing and the intended thing to analyse.

Time estimated: 15h

2.1.3 Performance analysis

Trace extraction

Once everything it is sanity-checked, the environment is ready and we are familiarized with the application the next step is to extract traces of the executions. As we will use the POP method we will need traces from executions from 1 process to 48 process and from 1 node to 16 nodes. This process is long and needs to operate with data.

Time estimated: 30h

Modelfactors

Once we have the traces we have to use basic analysis tool in order the extract the POP metrics from the executions. This process require a few hours to process the traces but nothing compared to extracting them.

Time estimated: 10h

Focus of analysis

In this stage of the project we will study the results of the modelfactors. From the results we will obtain insights into what is reducing the performance of the application. With this, we will use the other BSC tools to find the bottlenecks. Once the bottlenecks are found we will need to quantize the real impact on the performance to know which bottleneck is better to optimize. This bottleneck will be called the "Focus of analysis".

Knowing the time estimated is tricky as it depends on the easy is to identify the bottlenecks and the analysis in general, but, based in previous expiriences we estimate a total of 35h

Feedback to application developers

At this point of the project, a presentation and a conclusions of the analysis will be prepared and presented to the application developers.

Estimated time: 15h

2.1.4 Optimization

Design implementation

Based on the results of the previous section, we will decide what we will do to attack the bottleneck. The decision and the design it is an unknown for now. It is expected to be a total of 30h long.

Implementation

This phase is the great unknown of the project since we do not know what we are going to optimize or what we are going to do to optimize it. It is also the main task of the project and the one that will surely take the most hours. However, we cannot know how long it will take, since for now what we are going to do is unknown. Based on previous experiences and the time limit of the project we expect to invest 100h in this phase.

2.1.5 Testing implementation

It is mandatory to test and ensure that the modifications the code still lead to a correct program. A proper testing suite in collaboration with the application developers will be done.

Time estimated: 30h

Evaluation of the improvements

Once the job is done it is very important to evaluate the performance of the application with the improvements. This task consists of gathering traces and timing data from both versions, the one which is improved and the original one and elaborating a final conclusions of how is performing the optimization.

Time estimated: 40h

2.1.6 Final milestone

It is necessary to write the documentation before ending the project. Two sub-tasks are expected:

- Memory redaction. Time estimated: 70h
- Presentation preparation. Preparation of the final presentation of the work, this includes the material and the training for the presentation. Time estimated : 15h

2.1.7 Task dependencies and summary of tasks

Table 2.1 shows a summary of tasks and it's dependencies. Almost each tasks depends on its predecessor making this project really sequential.

Id	Name	Time (h)	Dependencies
T1	Project managment	110	
T1.1	Meetings	45	
T1.2	Context and scope of the project	30	T1.1
T1.3	Time planning	10	T1.2
T1.4	Budget and sustainability	10	T1.3
T1.5	Integration and final document	15	T1.4
T2	Hands-on	40	
T2.1	Contact with application developers	10	
T2.2	Study the compilation process	15	T2.1
T2.3	Run and test the application	15	T2.2
T3	Performance analysis	90	
T3.4	Trace extraction	30	T2.3
T3.5	Modelfactors	10	T3.4
T3.6	Focus of analysis	35	T3.5
T3.7	Feedback to application developers	15	T3.6
T4	Optimization	200	
T4.1	Design implementation	30	T3.6
T4.2	Implementation	100	T4.1
T4.3	Testing implementation	30	T4.2
T4.4	Evaluation of the improvements	40	T4.3
T5	Final milestone	85	
T5.1	Memory redaction	70	T4.3
T5.2	Presentation prepration	15	T5.1
Total			525

Table 2.1: Summary of tasks. Own compilation.

2.1.8 Task resources

Human resources

The BSC researcher will be in charge of the project and will be the main responsible of all tasks.

Other human resources are:

- The project director that is responsible of tracking the status of the project and giving feedback and suggestions to the researcher.
- The application developers are also human resources needed as they are in charge of introducing the researcher to the program (T1) and giving support if any problem with the application is encountered.

Material resources

- Dell Latitude 7490. Laptop that the project author will use for all the tasks.
- MareNostrum 4 supercomputer used for tasks T2, T3 and T4.
- Control versioning server used for keeping track of the changes to the code and the documentation. Tasks T1, T3.7, T4 and T6
- A Mailserver used for communication with the project director and the application developers.
- LaTeXused for writing documentation. Tasks T1, T3.7, T4 and T6.
- Ganttproject for the Gantt diagram tracking. Task T1.3.
- A text editor for writing the code, the documentation and in general manipulating files. All the tasks involved.
- Office supplies.

2.2 Gantt

The start of the project is expected by the 13/9/2020 which is the approximate start date of the thesis management course. The project is intended to end before the 29 of January that is the date of defence.

Figure 2.1 shows the gantt diagram of the project.

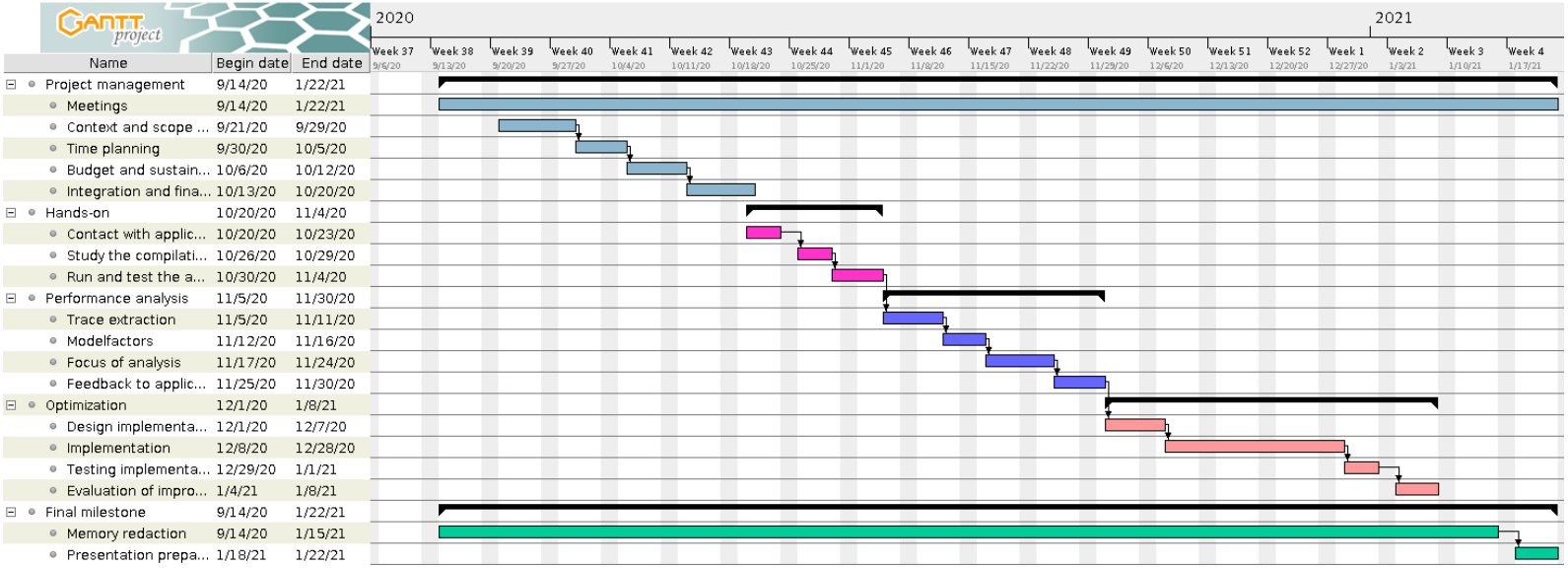

Figure 2.1: Gantt diagram. Own compilation.

2.3 Risk management: alternative plans and obstacles

- **MareNostrum4 not available**: The unavailability of the machine can lead to delays in almost all tasks. In general when the machine becomes unavailable in 1 day the issue is solved. We can expect a delay of maximum a 5% of the expected time. If a catastrophic event happens to the machine and it becomes unavailable during a notable period of time it is contemplated to move the study to a similar machine that BSC disposes of. This implies an increment of 20h to move the project and adapt to the new environment. Tasks affected: T.2, T.3 and T.4.

- **Misinterpretation of a metric during the analysis**: This can lead to small delays of maximum 5h. From previous experiences we can expect to not affect the project timing and this risk will be lesser as the researcher is more experienced. Tasks affected: T.3

- **Incorrect runs**: This risk affects the project for maximum 2h as it is fast to detect and fast to fix. Tasks affected: T.3 and T.4

- **Machine noise**: As this risk is contemplated and it is solved by gathering the proper number of samples it is already contemplated on the duration of the tasks and won't affect the timing of the project. Tasks affected: T.3 and T.4.

Chapter 3

Budget and sustainability

3.1 Budget

3.1.1 Staff

The following roles are considered for developing the tasks described in the Gantt diagram.

- Junior researcher.**Cost per hour:** 10 €/ h
- Project manager. **Cost per hour:** 20.24 €/ h

Table 3.1 shows given the tasks and the time to complete defined in the Gantt the hours each role focus on and the cost.

Task	Junior Researcher	Project manager
Project managment	20 h	90 h
Hands-on	30 h	10 h
Performance analysis	80 h	10 h
Optimization	150 h	50 h
Final Milestone	65 h	20 h
Total (€)	3450	3643
Total aggregated (€)	7093	

Table 3.1: Cost estimate for each role in the project. Own compilation

3.1.2 Material resources

The estimate cost of the material resources are detailed in table 3.2. Mind that:

- Cost and amortization of MareNostrum4 machine is not included as some of the data needed to calculate that must remain confidential.

- Some of the software used in the machine it is not free, neither as in freedom and as in beer. This cost is not included as it is also confidential.
- The office costs are illustrative as the project is developed in the K2M building in Campus Nord.
- Lifetimes are calculated by considering a 8 hours per day use during 5 years for electronic devices and 10 years for the furniture. Usage time may differ from real usage due to COVID-19 global pandemic.

Resource	Cost (€)	Lifetime (h)	Usage (h)	Amortization (€)
Dell latitude 7490	2500	9600	525	136.72
Generic display	300	9600	525	16.4
Generic keyboard	20	9600	525	1.09
Generic mouse	10	9600	525	0.54
Office furniture	1000	19200	525	27.34
Total	3830	-	-	182.03

Table 3.2: Amortization estimate for material resources. Own compilation

Table 3.3 shows the fixed monthly expenses. A final calculation on the part of the costs that are related to the project is added in a column. For example, if is a cost related to all the office users is divided by the number of the workers in the office.

Resource	Monthly cost (€)	Total cost (€)	Project cost (€)
Office rent	5000	25000	250
Office supplies	100	500	5
Watter and electricity	1000	5000	50
Total	6100	30500	305

Table 3.3: Fixed costs and total for project. Own compilation

3.1.3 Contingencies

It is estimated a 15% of the total of the amortizations and the staff costs for any contingency we could encounter during the project development.

$$Cont = (Amort + Staff) * 0.15 = (7093 + 182.03) * 0.15 = 1091.2545\ e$$

Incident	Extra staff time (h)	Material costs (€)	Total (€)
MareNostrum 4 unavailable	16.5	0	165
Machine change	20	0	200
Incorrect runs	2	0	20
Total	28.5	0	385

Table 3.4: Cost prediction of incidentals during the project. Own compilation.

3.1.4 Incidentals

In the previous chapter we commented the possible incidentals. Table 3.4 shows how the incidentals affect the budget.

We don't contemplate possible material costs associated to the MareNostrum 4 unavailable as it is the BSC the responsible of the machine maintenance and we are just users.

3.1.5 Total

Table 3.5 shows the grouped predicted budget of the whole project. Notice that from this cost we have omitted software licenses and compute hours costs which are confidential.

Concept	Cost (€
Staff	7093
Amortizations	182.03
Fixed costs	305
Contingencies	1091.25
Incidentals	385
Total	9056.28

Table 3.5: Total costs of the project. Own compilation

3.2 Sustainability report

3.2.1 Self-assessment of the current domain of sustainability competition

The sustainability is a field where I was not conscious to think about but indeed I did.

I'm always thinking on the social and economical aspects of my projects as I'm interested on this fields. A common question I use to ask myself is how

this project will affect the society? This is obviously thinking about the social and the economical aspects. Non-ethical technological products are becoming a tendency, changing people vote intention, governments tracking its citizens are examples of this and examples of developers that not thought about the social and economical aspects of their projects.

I have been ignoring or keeping apart thinking on the environmental aspects of my projects. This is usually because my projects don't use to impact on this aspect but now that I think that is the most important topic to think about it. At least on High-performance computing field that aims to solve or to find faster the solution to the problem.

I have also the concern to use and promote the free (as in freedom) software and hardware to avoid companies to be able to modify lifetime of products and therefore, generate more e-waste.

3.2.2 Economic Dimension

Reflection on the cost estimated

The cost estimated it is not precise as some parts are omitted for confidentiality. The estimations I made try to be realistic with the current price standard in Barcelona, Spain but I'm almost sure I miss predicted some estimations as I'm not experienced on this also not in charge of making the expenses.

Useful Life

My project aims to improve the computation time of a simulation. Hopefully if we improve the simulation the computation times will be lesser, therefore, scientists will have to invert less money on computational resources or will reach faster to conclusion which leads to scientific progress.

3.2.3 Environmental Dimension

Environmental impact of the project development

The environmental impact estimation has not been done. It is a very complex calculation as it involves the MareNostrum4 fabrication and electrical consumption and it is used by a lot of users. Scoping this in the project means making a lot of estimations which is not possible.

Minimize environmental impact

This is not possible in the project. The machine is already built.

The only way we can act is in being conscious the electricity that consumes the machine and the laptop and try to use efficient the resources. This implies performing the appropriate number of runs.

Environmental impact of the project

The project will have a positive environmental impact. This research will be useful in order to make programs more efficient and reduce the running time of the applications which implies less power consumption.

3.2.4 Social Dimension

Personal growth

This project will make me grow on my young research career and will make me improve me technical and social skills. Contacting and explaining the project to the application developers, analysing, thinking and developing an optimal solution, discussing with my project manager all the aspects of the project, writing technical reports are things that will make me grow as a professional and as person.

Social improvement

This project aims to accelerate the combustion research as its objective is to make the scientific simulation to go faster. This can impact the society in many ways, from making less pollute motors to making faster planes. We cannot predict what research we will accelerate.

Is there a real need for the project?

There is a need to improve this simulations because they are really slow. A solution is needed and we can provide a step towards this.

Bibliography

[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