Supercomputing for Climate Workflows

Enric Millán Iglesias

Final Report

REVISION HISTORY AND APPROVAL RECORD

Revision	Date	Purpose
0	21/06/2024	Document creation
1	12/08/2024	Document revision
2	22/08/2024	Document revision

DOCUMENT DISTRIBUTION LIST

Name	E-mail
ENRIC MILLÁN IGLESIAS	enric.millan.iglesias@estudiantat.upc.edu
MANUEL GIMÉNEZ DE CASTRO MARCIANI	manuel.gimenez@bsc.es
ALBERTO ABELLO GAMAZO	alberto.abello@upc.edu

WRITTEN BY:		REVIEWED AND APPROVED BY:	
Date	22/08/2024	Date	dd/mm/yyyy
Name	Enric Millán Iglesias	Name	Zzzzzzz Wwwwwww
Position	Project Author	Position	Project Supervisor

0. CONTENTS

1. TIME PLAN UPDATED

A posteriori real Time Plan

Work packages remain the same as the ones in the Critical Report, there only was a change in scheduling due to some issues explained in the section *2.5- Unplanned Issues*. The change consists of extending the second work package one more week, and as a consequence, delaying one week the third work package. Below is a summarized (with the dates updated, the content did not change) version of the work packages and the updated Gantt chart:

WP1: Context for the project and thesis definition - From 17/06/24 to 12/07/24 (4 weeks)

Get to know the people from the department (from the Computational Earth Sciences group), the basics of working at BSC, and get context for the project. Install software dependencies and define thesis.

WP2: Experiments Execution - From 15/07/24 to 02/08/24 (3 weeks)

Learn how to automate and parallelize experiment executions and start performing them.

WP3: Results Analysis - From 05/08/24 to 16/08/24 (2 weeks)

Develop a script to perform data analysis, explore data fittings, and test the hypotheses.

WP4: Write Critical Review and Final Report - From 15/07/24 to 23/08/24 (6 weeks)

Write both reports, review all the work done and set everything up for submission.

2. PROJECT DESCRIPTION

Contents:

2.1- Context

This project builds from Manuel Giménez de Castro Marciani's thesis (https://upcommons.upc.edu/handle/2117/404041). In his thesis, Manuel studied the relative impact of task aggregation, or wrapping, which is a technique meant for computational workflows that bundles jobs into a single submission to be sent to remote schedulers. Manuel is a PhD student at the BSC, working in the Models and Workflows team of the Computational Earth Sciences group.

2.2- Motivation

Experiments inside the Earth Science community, which include all kinds of climate models and simulations, can be lengthy and comprise several steps with many dependencies. The community has traditionally focused on increasing the performance of the models, but the overall execution of the workflow, including the queue time, has received little interest. Aiming to reduce the time spent in queue, the developers of Autosubmit, a workflow manager developed by BSC for climate simulations, weather forecast simulations, and air quality simulations, came up with task aggregating, or wrapping. It is believed that this technique reduces queue time, and it has so far been utilized indiscriminately.

2.3- Objective

The objective of this project is to analyze the impact of wrappers on the queue time by performing a statistical analysis and finding the most significant variables or factors from the workflow geometry and the scheduling algorithm of supercomputers.

More specifically, the objective is to test the theses originally proposed in the Critical Review document:

1.- The impact of the wrapper should be less significant the smaller the resources required (allocated CPUs and runtime) by the workflow (set of jobs) is.

2.- The bigger the fair share value of the user sending the jobs (fair share is an important factor to determine priority), the less impact wrappers should have.

For this task, we successfully completed basic guides on how to use different tools, from version control systems such as Git (using GitLab), to running containerized applications with Docker (a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers) and, finally, automatizing and parallelizing the executions of the experiments using Nextflow, a workflow management tool.

2.4- Methodology

To evaluate the impact of wrappers, we performed several experiments. These experiments consist of simulations of workloads of a real supercomputer. The BSC version of Slurm Simulator was used to simulate the logs of real machines provided by Dr. Dror G. Feitelson in the Parallel Workloads Archive (https://www.cs.huji.ac.il/labs/parallel/workload/), specifically the CEA Curie machine logs (https://www.cs.huji.ac.il/labs/parallel/workload/l_cea_curie/index.html).

This workload was chosen because it is one of the largest publicly available logs, with more than 20 months worth of data, from a scientific general-purpose platform. But we selected only a week of it because the system was not normally utilized to the fullest as opposed to modern systems such as MareNostrum 4 and LUMI.

In this section we explain how we set up the experiments, how simulations are executed, how we automate and parallelize these executions and how we perform an analysis of the resutls.

2.4.1- Workload modification

To execute the simulations, the original workload of the selected week is altered by adding a synthetic workflow (group of jobs). This workflow is made of identical jobs which request 96 cores and have a runtime of 30 minutes..

We added a total of 48 different workflows for this study (each one individually). These workflows were submitted in 6 different times of the original workload, with the criteria of deploying the same workflow in different moments to avoid being misled by abnormal utilization.

The 48 workflows were designed so that they would vary on different features, such as size (number of jobs of the workflow), type (vertical, jobs with dependencies, or horizontal, without dependencies), fair share value of the user launching the job (fair share is an important factor for the scheduler when computing priority) and aggregation condition: wrapped (aggregated as a single job) or not wrapped. This made up for a total of 288 experiments performed during this project.

Specifically, we chose the 48 experiments' configurations to make all possible combinations of the previously mentioned variables for the following values:

- Size: 2, 8, 14, 20.
- Type: vertical or horizontal.
- Fair share: 0.1, 0.5, 1.0.
- Aggregation condition: wrapped or not wrapped.

We selected these values of size and fair share to have a diverse set of experiments while keeping it possible to run in the 10 week period available for the project (the number of possible experiments to perform was determined considering time limitations, as explained later).

To add these workflow configurations to the original portion of the workload, we developed a Python script during this work that generated a total of 288 trace, or workload, files ready to be used as input by the SLURM simulator.

These trace files contain information about all the jobs submitted during the selected week plus the added workflow jobs of each experiment, and are codified in Standard Workload Format (SWF). SWF was chosen in order to ease the use of workload logs and models. With it, programs that analyze workloads or simulate system scheduling need only be able to parse a single format, and can be applied to multiple workloads.

This format comprises a lot of parameters for each job, from identifiers such as *Job Number*, *User ID* or *Group ID* to detailed information about the execution of each job with variables such as *Submit Time*, *Wait Time*, *Run Time, Requested Time*, *Number of Allocated Processors*, *Requested Number of Processors*, *Used Memory* or *Requested Memory* among many others. More information on the Standard Workflow Format can be found in Dror G. Feitelson's repository (https://www.cs.huji.ac.il/labs/parallel/workload/swf.html).

To be able to identify and control the fair share, the added workflow in the output files, we set the *User ID* and *Group ID* to 723. We checked that the selected week of the whole workload did not have any users (or groups) sending jobs with this ID.

For the experiments where the fair share is not 1, the usage of user 723 (the one sending the added workflows) had to be altered to meet the selected fair share value (0.1 or 0.5). To fulfill this requirement, our script adds an initial job, before the workload starts, sent by user 723. The resources requested by this job are meant to control the fair share value of the user.

2.4.2- Execution of simulations

Once the workload files for all experiments are ready, that is with its workflow included, it is time to run the simulations with the aforementioned SLURM simulator. To use the simulator with the generated trace files, we used the available Python script called *launcher.py*. This script takes as input the path of the input trace file, the path of the output file where the simulation output should be printed on, and the total simulated time in seconds for the simulation to run.

By default, the time used was 10,000,000 (1e7) seconds, which is approximately 3 months of simulation time. This value was deemed satisfactory because it allowed the included workflow to run in its entirety. It was used by Manuel and tested by us.

The input files are those previously generated trace files in the Standard Workflow Format, a total of 288.

The output files consist of a text file recording all executed jobs in order, where each job is depicted as a line with several identifiers and variables, many which are present in the SWF. As mentioned in the previous subsection, for experiments where the user has a fair share different than 1, this output file will have an extra job of the user executing the workflow (besides the workflow of

the experiment) being the first of the whole simulation to control the fair share of user 723 for the given experiment. This job has to be disregarded in the analysis.

With respect to the simulation itself, the SLURM simulator is executed in a Docker container, which is launched by the *launcher.py* script. Docker allows to run containerized applications in an easy way. Additionally, we used Docker Desktop to ease the monitoring of experiments, since the *launcher.py* runs one container for each experiment that at the same time is running the SLURM simulator with the trace file inputted. This made Docker Desktop ideal to supervise these container runs when parallelizing different executions. Moreover, this application's guides and tutorials were used during this project to understand what a container is and how Docker works.

2.4.3- Automation and parallelization of simulations using Nextflow

When executing a simulation individually, the total amount of time required often was between 1 and 1.5 hours of wall time. This meant that executing the 288 simulations would take around 400 hours. This posed a problem regarding the viability of performing the experiments, given that the work plan designated a total of 2 weeks (10 days * 8 hours/day = 80 hours), so not only automation but also parallelization became a need.

For this task, we considered several options. When automatizing executions, Manuel used Cylc, a workflow manager (https://cylc.github.io/). However, the experiments were not executed in parallel. Moreover, Cylc is specially designed for cyclic workflows, and the workflow for this project consists of 288 independent executions of a script with varying parameters. Therefore, we developed a novel workflow utilizing Nextflow, a scientific workflow system which automatically parallelized the processes defined.

This decision was heavily influenced by the addition of a new member to the Models and Workflows Team by the time this project was being developed, who is a former developer of this tool and helped during the design of the Nextflow code used in this project.

The code itself is a script called “automate_experiments.nf” (.nf is the extension of Nextflow codes) of less than 100 lines. The code is open-source and available in the GitLab repository of the project, and it follows the next structure:

1.- Definition of global variables such as the output path where results of each experiment are saved into, or the path where the launcher python script is among others.

2.- Definition of the first process, called *update_status*, which receives the path of an trace file (.swf) and executes a python script called *check_completed.py* that checks if the experiment of the corresponding file is completed or not based on how many times the userId 723 appears in the output text file of the corresponding experiment. The script then updates a csv file containing all experiments names and their status (completed or not) called *experiments_status.csv*.

3.- Definition of the second process, called *execute_simulation*, which again receives the path of a trace file and a “True” boolean value from the previous process (this is to ensure that no instance of the second process starts before all instances of the first process are done). The process first checks the previously updated csv with the status of every experiment. If the trace file received corresponds to a completed experiment, the process ends. If not, the process sets up a Docker container and runs the BSC Slurm Simulator using the launcher.py script.

4.- Definition of the workflow that Nextflow will execute. First, we define a channel, that is, a list of inputs, containing all .swf files. Then, Nextflow checks this channel and executes the process *update_status* and then the *execute_simulation*, once its dependency is met.

Additionally, the first process is limited to run one of each instance at a time (no parallelization) to avoid race conditions. On the other hand, the second process is limited to 6 parallel executions at a time. We chose this number of parallel executions, after some testing, because it was the highest number of parallel executions we could make with the available hardware.

Besides the simulator’s maximum simulated time, we set up another for Nextflow of 1.25h (1:15). This was due to two reasons: firstly, since we only needed the simulation to run until the last job of the workflow to study in each experiment, 1.25 was enough time for all experiments; secondly, as explained in the next section, some simulations got stuck and this also allowed us to stop them.

2.4.4- Analysis of results

For the task of analyzing the results, we developed an IPython Notebook. The IPython Notebook, also known as the Jupyter Notebook, is an interactive computational environment which combines code execution, rich text, mathematics, plots and rich media.

The complete notebook can be found in this GitLab repository. The notebook follows the next structure:

1.- Data Loading

2.- Data Cleaning and Pre-processing

3.- Plot Based Analysis

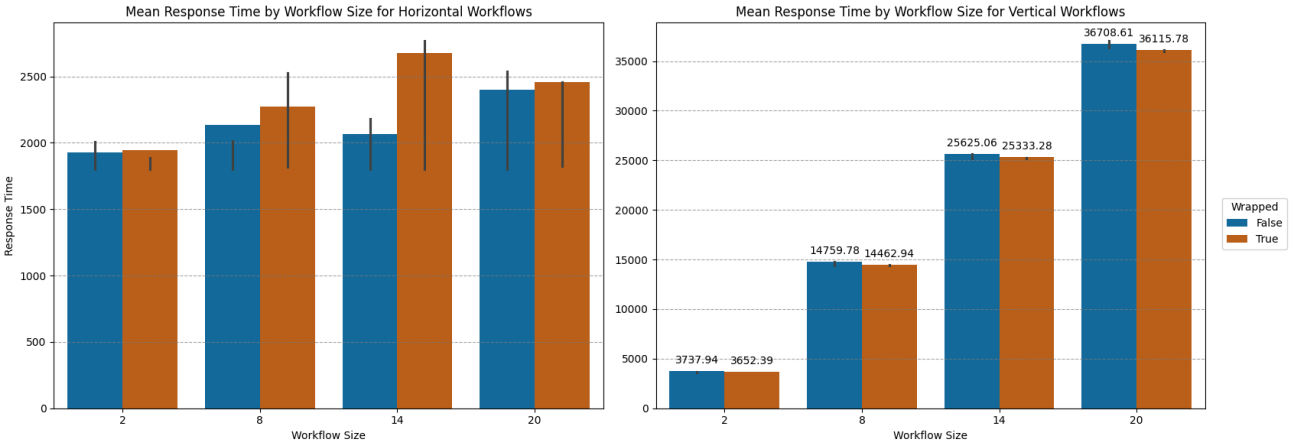
- 3.1- Response Time Analysis
- 3.2- Normalized Response Time Analysis
- 3.3- Speedup Analysis

4.- Feature Importance Analysis

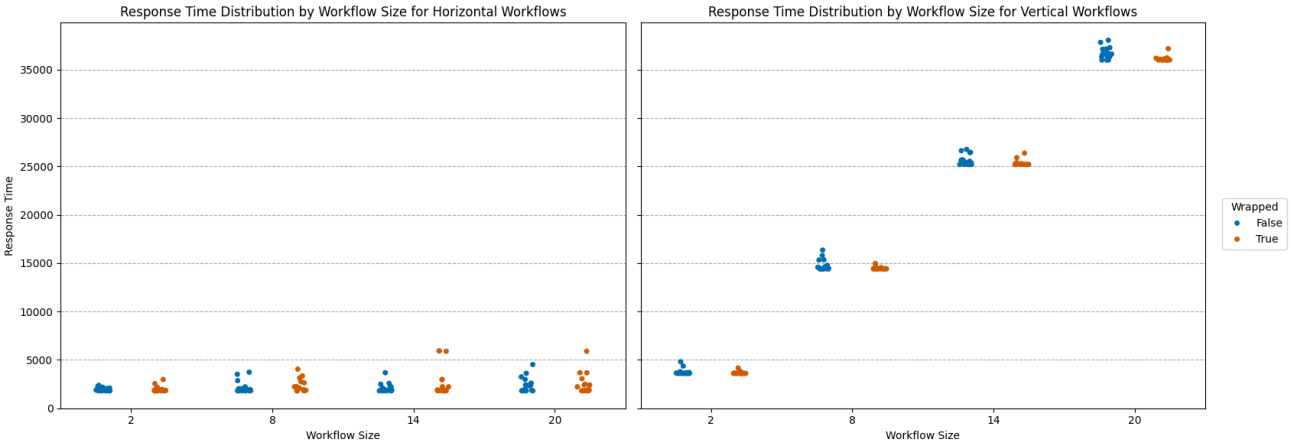
- 4.1- Random Forest Regression
- 4.2- Linear Regression
- 4.3- Correlation Based Importance

During this subsection we will review the most interesting insights obtained from the analysis with respect to our original theses. Extra information can be found in the notebook, which is fully commented. Note that when talking about response time, we are referring to the time the workflow takes to end from submission (queue time plus execution time), and when talking about normalized response time we are referring to the response time divided by the execution time.

First of all, with regard to the first thesis of this project, which states that the less resources (CPUs and runtime) needed by the workflow the less impact wrappers will have, we have the following plot:

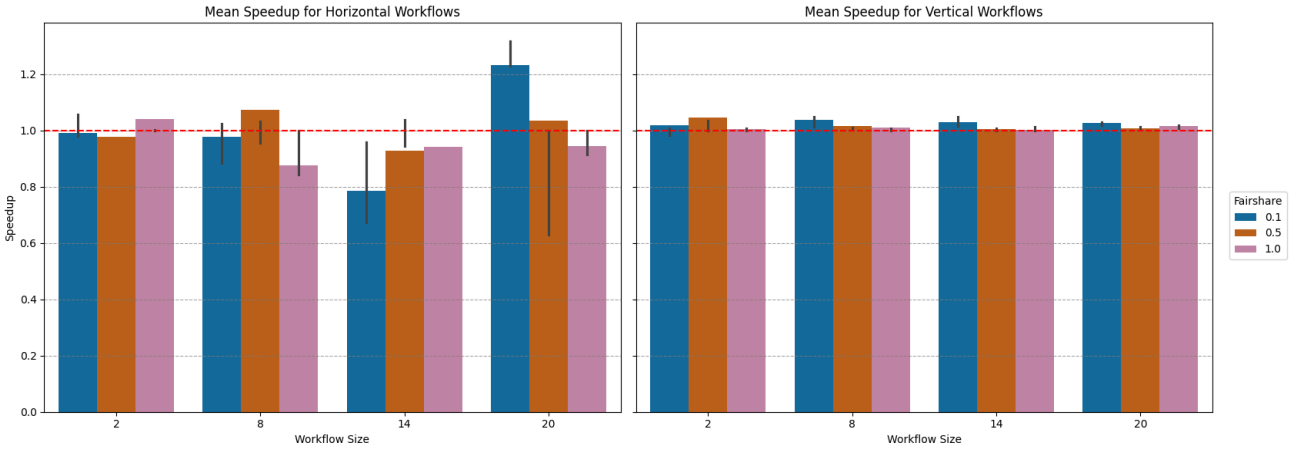


As we can see, this is supported by the horizontal workflows, where we see that the smaller size (2) is the one with the least difference (on average) between wrapped and unwrapped response time. However, we can't claim this yet for vertical workflows, where wrappers seem to have a pretty similar impact for all sizes. The next plots gives us a more detailed view of the same information:



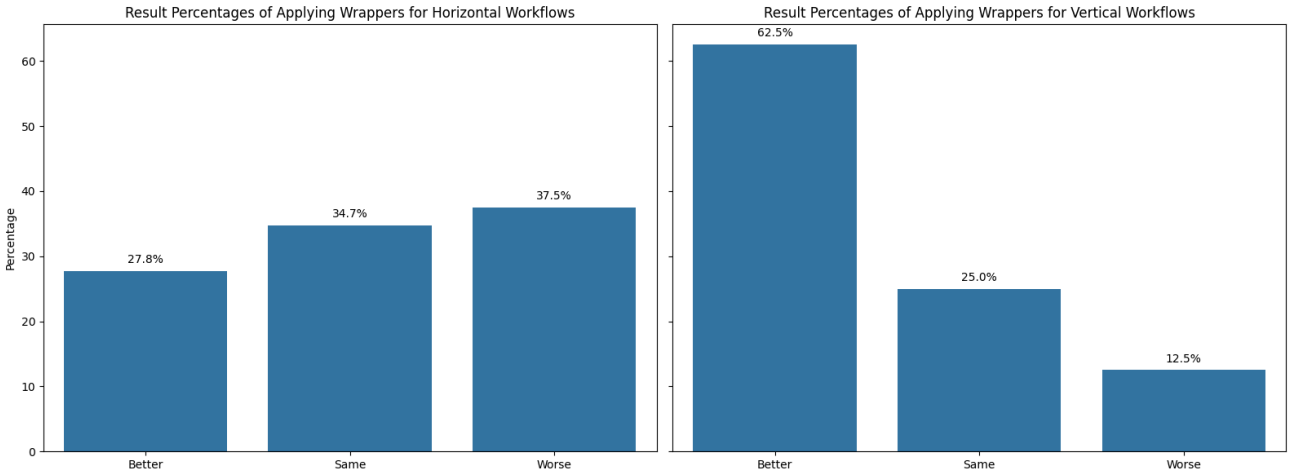
Now, we can see that also for vertical workflows our thesis seems to be consistent. We can see that variability, and thus impact of wrappers, apparently increases with size. It is important to note that in this project, size equates with resources because the workflow is made of identical jobs, which is not normally true in real cases.

Our second thesis states that the larger the fair share (of the user sending the workflows), the less impact wrappers will have. We will analyze the following plot, which shows the mean speedup (unwrapped response time divided by wrapped response time for the same experiment) obtained by wrapping the workflows with different sizes and fair share values:

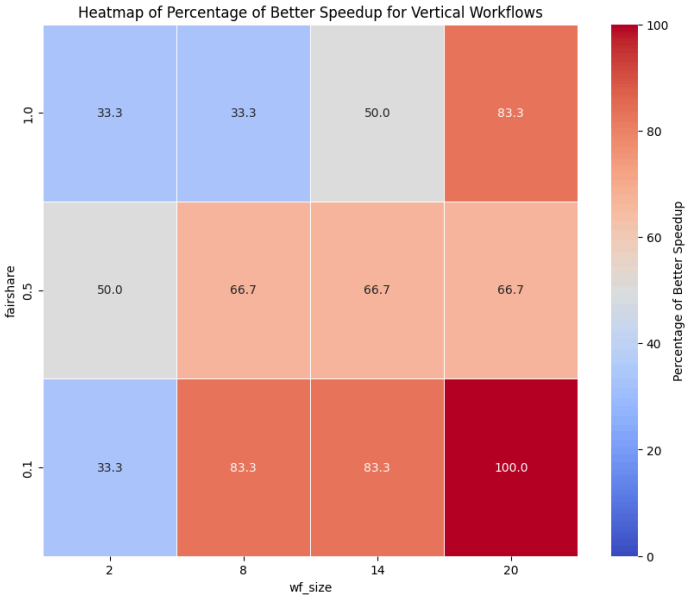


A speedup value of 1 (dashed red line) means that the response time is the same for both the wrapped and unwrapped version of an experiment. As we can see, on average, all vertical workflows have a speedup above 1. For vertical workflows too, except for the ones of size 2. We can see that the lowest fair share value does indeed get the best speedup, which seems to be consistent with both our theses. In the case of the workflow of size 2 we have that the smaller the workflow, the more it benefits from backfill (backfill scheduling allows other jobs to use the reserved job slots, as long as these jobs do not delay the start of another job) and therefore less from wrapping. With respect to the horizontal workflows, we have inconclusive results. As seen in the previous plots, and explained in the notebook, horizontal workflows have much more variability and wrappers tend to worsen or have no impact for this type of workflow.

The following plots show this and help reinforce our two theses, although only for vertical workflows:



It is clear that vertical workflows are the ones that benefit from wrappers. Now, we check these percentages with more detail and see if they are consistent with our two theses with the next plot, where we have in the x-axis the workflow size and in the y-axis the fair share. Within each of the blocks we have the percentage of experiments that the speedup was greater than one.



We can see that vertical workflows are the type of workflow which is mostly consistent with both our theses. The highest percentages of improvement can be found in the column of largest size, fitting with the first thesis, and in the row of lowest fair share, fitting with the second thesis.

Additional plots and analyses regarding the content previously mentioned can be found in the python notebook of the project.

2.5- Unplanned issues

During the project, we faced several unplanned issues, especially related with the execution of the simulations.

A bug was introduced which made the Slurm simulator not take into account the fair share. Around 30 simulations were executed with a configuration that did not take into account the fair share value, so those were not useful for this project, meaning that we had to change the configuration and rerun these simulations.

Also, with respect to the automation of simulations, some minor issues arised. As explained in a previous section, the Nextflow script was set to execute 6 simultaneous executions, and it did without posing any problems related with the resources of the laptop. However, on average, on about half of the executions the simulator did not finish.

To try to solve this problem, the time limit for the Nextflow process was gradually increased up to 2 hours, without getting any better results. Seeing this, the time limit was set again to 1.25 hours, since this time allowed to run all executions on schedule even if half of them got stuck every time.

These two issues combined caused the second work package (execution of experiments) to need an extra week and consequently the third work package (analysis of results) to be delayed one week. This did not imply any major problems since there were two weeks destined to writing the report, a task that was done simultaneously with other tasks.

Finally, after running all the simulations, and during the analysis with the IPython Notebook. We realized that the priority values for jobs with fair share value 0.1 were around 40,000 (Data Cleaning and Pre-processing section). This priority value could only be obtained with a fair share value around 0.4, implying that the methodology to control the fair share using an initial job is wrong. Solving this issue would be a priority in future related work, however, the general insights obtained in the analysis are still maintained, since we mostly refer to smaller or bigger fair share values and not the exact values (0.4 is still lower than 0.5 and 1.0).

3. CONCLUSIONS

The main conclusions of this project can be summarized in the following points:

- Vertical workflows seem to benefit in most cases from the use of wrappers, and in most of the cases where they can be perjudicial, the negative impact is really small.
- Horizontal workflows have a much higher variability and it is unclear under which circumstances they could benefit from wrappers. This indicates the necessity of more experimentation.
- The positive impact of wrappers for vertical workflows is most noticeable for the larger workflows and for the users with lowest fair share values, fitting with our two initial theses.

4. REFLECTION DOCUMENT

- Things that could have been done better by the company staff or supervisor

- Things that could have been done better by Telecom-BCN staff or supervisor

- Things that could have been done better by the author

- Learning outcomes

I learned what wrappers are and the potential they have to save queue time (and therefore resources) from workflows being submitted to supercomputers. I learned the basics about the SLURM scheduler and the SLURM simulator. I also learned the basics about workflow managers, specifically learning how to use Nextflow to automate and parallelize the simulations. In order to run Manuel's scripts and the simulator I learned what containers are and how to use Docker. To do version control for all the code developed, I improved my knowledge of git and the GitLab interface. I learned the groups and teams organization used at BSC, as well as team collaboration methodologies by attending several meetings. Finally, I also attended some optional "Writing Parties", where coworkers wrote essays, papers, reports, etc, and shared tips and advice on writing.

- Self assessment

I stuck to the work plan and completed the tasks on schedule. Even though some issues arose during the internship, Manuel and I quickly addressed them and we were able to continue progressing on the project without any major delays.