Universidad de Cantabria

Facultad de Ciencias

Autosubmit Wrappers to Speed Up Scientific Production

Envío automático de Wrappers para acelerar la producción científica

Trabajo de Fin de Grado para acceder al

Grado en Ingeniería Informática

Autor: Pablo Goitia González

Director: José Luis Bosque Orero

Co-Director: Manuel Giménez de Castro

Julio - 2024

Agradecimientos

Llegar hasta aquí no ha sido tarea fácil y, por esta razón, no quería desaprovechar la oportunidad de agradecer y reconocer a quienes que me han acompañado a lo largo del camino.

En primer lugar, a mi familia, por su cariño y apoyo incondicional. Soportar las frustraciones de un universitario debe de ser duro, pero ellos han cumplido con creces.

A mis amigos, a los de siempre y a los que han ido llegando para quedarse, por estar en las buenas y en las malas, por creer siempre en mí.

A los profesores del Grado, porque todos, de alguna manera, formáis parte de lo que soy ahora.

A José Luis Bosque, por su gran labor como director en este proyecto, y a Manuel Giménez, por todas las cosas que he aprendido bajo su supervisión y por su empeño en mantener mi moral siempre alta. También, cómo no, a Miguel Castrillo y Francesca Macchia, por abrirme, una vez más, las puertas de su equipo para poder desarrollar mi Trabajo de Fin de Grado. *Molte grazie.*

A todos mis compañeros y amigos del *Barcelona Supercomputing Center*, muy especialmente al que ha sido mi mentor durante buena parte de mi paso por el BSC. *Muito obrigado*, Bruno.

Y a todos vosotros, deciros que esto no es más que el principio de todo, y que espero con ilusión lo que sea que esté por venir.

Resumen

Los Modelos del Sistema Terrestre (ESMs, por sus siglas en inglés) son modelos numéricos complejos que simulan tanto los distintos componentes del sistema terrestre como las interacciones entre ellos. Los ESMs constituyen la base de la producción científica en el ámbito de las Ciencias de la Tierra, por lo que trabajar en su optimización computacional constituye una tarea crítica.

Estos modelos son sistemas complejos y su puesta en marcha normalmente conlleva la ejecución de múltiples pasos asociados a los varios componentes que los integran, el tratamiento de los datos, y la larga extensión temporal de las simulaciones climáticas. A lo largo de este estudio, evaluaremos el impacto de la agregación de tareas en simulaciones reales de estos modelos ejecutadas en varios supercomputadores europeos.

En este contexto, la agregación de tareas – o wrapping – es un mecanismo que nos permite enviar varias tareas agrupadas en una sola a las plataformas remotas. Gracias a estudios anteriores, sabemos que agrupar tareas en *wrappers* verticales – es decir, en los que cada tarea depende de la anterior –, implica la reducción del tiempo que pasan las distintas tareas en la cola del planificador, es decir, el tiempo que esperan a que el sistema les conceda los recursos necesarios para ser ejecutadas. Al reducir los tiempos de cola, lo que estaríamos consiguiendo en última instancia es reducir el tiempo total de ejecución del flujo de tareas, o *workflow*.

Como base de nuestra investigación utilizaremos el modelo comunitario europeo EC-Earth3, uno de los ESMs más reconocidos en Europa por sus contribuciones al proyecto CMIP6. El *workflow* que lo implementa ha sido portado a múltiples supercomputadores europeos como MeluXina de LuxProvide, HPC2020 de ECMWF o MareNostrum 4 y MareNostrum 5 del Barcelona Supercomputing Center.

Los resultados experimentales de nuestro estudio muestran que para las tres plataformas de supercomputación testadas *los tiempos de cola agregados para un experimento de simulación que utiliza wrappers verticales son de 11 a 12 veces más cortos en comparación con los de un experimento que no los utiliza.*

* * *

Palabras clave

Computación de Alto Rendimiento, Supercomputación, Agregación de Tareas, Slurm, Workflow, Modelos del Sistema Terrestre, Simulación del Clima.

Abstract

Earth System Models (ESMs) are complex numerical models that simulate both the different components of the Earth and the interactions between them. The ESMs are the basis of scientific production in the field of Earth Sciences, so working on their computational optimization is an essential endeavor.

These models are complex systems and their implementation usually involves the execution of multiple steps associated with the various components that integrate them, data processing, and the long time extension of the climate simulations. Throughout this research, we will assess the effects of task aggregation on real-life simulations of these models conducted on various European supercomputers.

In this context, task aggregation, also known as wrapping, is a strategy that enables us to bundle multiple tasks and submit them as a single job *wrapper* to remote platforms. Previous research states that organizing tasks into vertical wrappers – where each task is dependent on the preceding one – results in shorter waiting times in the scheduler queue, which is the duration for which they wait for the system to allocate the required resources for execution. Consequently, reducing queue times leads to a decrease in the overall execution time of the workflow.

We will utilize the European community model EC-Earth3 as the basis of our research, one of the best known ESMs in Europe. The workflow that executes it has been ported for use on several European supercomputers, including MeluXina from LuxProvide and HPC2020 from ECMWF together with MareNostrum 4 and MareNostrum 5 at the Barcelona Supercomputing Center.

The experimental findings of our research indicate that across the three evaluated supercomputing platforms *the total queue times for an experiment utilizing vertical wrappers are 11 to 12 times shorter compared to an experiment without them.*

* * *

Keywords

High Performance Computing, Supercomputing, Task Aggregation, Slurm, Workflow, Earth System Models, Climate Simulation.

Contents

List of Figures

List of Tables

List of Codes

Acronyms

API Application Programming Interface.

BSC Barcelona Supercomputing Center.

CES Computational Earth Sciences.

CESGA Centro de Supercomputación de Galicia.

CHSY Core Hours per Simulated Year.

CLI Command Line Interface.

CMIP Coupled Model Intercomparison Project.

CMOR Climate Model Output Rewriter.

CSC Finish IT Center for Science.

ECMWF European Centre for Medium-Range Weather Forecasts.

ESM Earth System Model.

FIFO First In, First Out.

FLOPS Floating Point Operations per Second.

GUI Graphical User Interface.

HBM High Bandwidth Memory.

HPC High Performance Computing.

HTC High Throughput Computing.

IFS Integrated Forecasting System.

KIT Karlsruhe Institute of Technology.

MPI Message Passing Interface.

MPMD Multiple Program Multiple Data.

NEMO Nucleus for European Modelling of the Ocean.

NIWA National Institute of Water and Atmospheric Research.

OLCF Oak Ridge Leadership Computing Facility.

SBU System Billing Units.

Chapter 1

Introduction

In this chapter, we will go through the inspiration behind this research, the objectives we aim to achieve, and the contributions this project will make to the Earth sciences field.

1.1 Motivation

Over the past few decades, the High Performance Computing (HPC) domain has experienced unprecedented expansion [1], with the current landscape featuring heterogeneous systems that merge traditional highly parallel CPU-based architectures with increasingly prominent GPU-accelerated partitions, alongside other emerging technologies. Advancements in the processing capacity of these chips have made it possible to create the first exascale supercomputer, named Frontier, at the Oak Ridge Leadership Computing Facility in the United States. It is capable of performing over a quintillion floating point operations per second, or 10^{18} FLOPS, and continues to hold the number one position on the TOP500 list [2].

An increasing number of businesses and institutions are working together and investing in the creation of machines that drive the progress of science, society, and industry. Historically, Europe has been home to supercomputers that ranked highly in the TOP500 list, such as BSC's MareNostrum. In 2018, in order to maintain Europe's leadership in supercomputing, the European Commission established the EuroHPC Joint Undertaking alliance to coordinate the efforts of the member countries, with the aim of digitally transform the economy, enhance industries, and reinforce digital sovereignty across the continent. As a result of this collective initiative, several supercomputers have been deployed throughout Europe, emphasizing LUMI in Finland, Leonardo in Italy, and MareNostrum 5 in Spain, which are the three most powerful machines in Europe. [3]

The capabilities of these new heterogeneous supercomputers have enabled individuals to broaden their applications to include large scale simulations, deep learning, artificial intelligence, and even the development of digital twins [4] of the human body or our planet [5], with the aim of enhancing our comprehension of our environment and nature.

This expansion drives researchers around the world to seek methods for maximizing the capabilities of these machines as they incorporate emerging technologies, which have a growing number of users demanding their resources. As a result, there is an aspiration to find a balance between system performance and energy consumption [6]. This research can be performed at different levels, such as by enhancing existing workload managers [7, 8] or refining current workflow managers to optimize the use of available resources during task submission [9].

In the field of Earth sciences, the day-to-day work involves the development and con-

tinuous enhancing of different Earth System Models (ESMs). These models are numerical representations of the different components of the Earth system, such as the atmosphere, the ocean, the land, or the carbon cycle, and are executed on supercomputing platforms organized in large workflows consisting of up to hundreds of thousands of interdependent jobs. Optimizing the simulations of these Earth System Models is a critical task for the Earth science community, as a significant portion of its scientific production relies on them. These optimization tasks can be approached on several fronts. One of them is trying to minimize the time that the jobs inside the workflow spend in the workload manager's queue, thus reducing the overall execution time of the entire model. Giménez de Castro et al. [10] explored the task aggregation solution by conducting multiple experiments with a Slurm Workload Manager simulator. They stated, among other findings, that applying task aggregation to predominantly vertical workflows, i.e., where each job is dependent on the previous one, which are also prevalent in most ESMs, improves overall queue durations.

One of these ESMs, which also be the objective of our study, is EC-Earth3 [11], a highly modular model that makes it possible to simulate different configurations of climate components. Developed by the European research consortium EC-Earth, it is renowned for its significant contributions to the CMIP phase 6 project [12], with the subsequent impact on the creation of the Sixth Assessment Report (AR6) [13], and consequently influencing the design and implementation of climate policies.

1.2 Objective

Our research aims to evaluate the impact of task aggregation – or *wrapping* – in order to reduce total queueing times in real climate simulations – thus, the overall execution time –, using a workflow used in production to run one of the widely known Earth System Models in Europe, EC-Earth3. Wrapping consists in grouping multiple tasks of these workflows into a single larger job, which can then be dispatched to a remote scheduler like Slurm at once. Slurm is a well-known workload manager utilized on the supercomputing platforms in which we will execute our experiments throughout this project. The model workflow will be executed on top-tier supercomputers across Europe using one of the best known workflow managers in this field, Autosubmit [9].

This study seeks to determine whether task aggregation reduces queueing times in the remote scheduler, thus reducing the overall execution time of the workflows.

1.3 Contribution

This study will contribute to the Earth sciences field as the first controlled assessment of the impact on wrapper utilization within real-life workflows, requiring more than half a million CPU hours on some state-of-the-art European supercomputers, including MareNostrum 4, MareNostrum 5, or MeluXina.

This project has been developed in collaboration with the Computational Earth Sciences group of the Department of Earth Sciences of the *Barcelona Supercomputing Center-Centro Nacional de Supercomputación*, to which the author and co-director of this paper belong.

1.4 Working plan

The initial step in the working plan for this project involves assessing existing research by other authors regarding supercomputer workload and task aggregation.

After understanding the given context, we can proceed to set up our experiments. This process involves selecting a workflow from the various ESMs available. The chosen workflow will eventually be the EC-Earth3 workflow.

Each workflow operates on specific computing platforms, so we will have to decide on which ones will be used for our experiments. Choosing multiple platforms is critical because, first, it provides us with a broader range of cases for our subsequent analysis, and second, there is a chance that experimentation on any single platform may encounter issues and fail. From each platform, we will extract data on both queue times and the various parameters that the scheduler considers when assigning task priorities. With these data, we can determine whether, in the various scenarios analyzed, task aggregation in vertical workflows contributes to improve queue times and the overall execution time of the workflow.

1.5 Document organization

This document is structured into five chapters, with the present one being Chapter 1, which introduces the project outline. The subsequent content is arranged as follows:

- *Chapter 2* makes an overview of the previous research on the findings of other authors who previously studied both workloads in HPC platforms and task aggregation as a solution to reduce the job queuing times. Moreover, it provides an thorough explanation on all the concepts that will be treated throughout this research, as the aforementioned Earth System Models, Autosubmit, Slurm and wrappers.
- The *Chapter 3* details all the experimentation process, from the workflow selection, platform setup, and, finally, their execution.
- On *Chapter 4* we present and discuss the execution results.
- And, lastly, on *Chapter 5* we present all the conclusions we have extracted from what we observed after finishing the experimentation.

Chapter 2

Background

Throughout this chapter, we will walk through the findings of various authors who have previously studied task dispatching to HPC platforms, focusing on how job size impacts queuing times and the advantages of using task aggregation techniques. Subsequently, we will introduce all the tools and concepts that will be systematically used in the different stages of this research, which are crucial to our understanding.

2.1 State of the art

The increasing demand for computing resources in HPC centers, enterprise computing, or cloud providers has driven increased investment in the research of new techniques to achieve an optimal use of HPC resources. This motivates the paper by Patel et al. [1], which focuses on the characterization of submitted jobs. The authors state that there is a lack of characterization studies that leads researchers to base their developments on old characteristics, and this results in system administrators being reluctant to apply it to new machines. The main purpose of their research is to help other researchers find the latest trends and aspects they may need to develop reliable strategies for resource management.

To do so, they examined workload on two leadership-class supercomputers hosted in the Aragonne National Laboratory, Intrepid and Mira, analyzing "trends and characteristics for over three billion compute hours, 750 thousand jobs, and spanning a decade". Along the way, they discovered some interesting insights, highlighting that jobs are becoming larger and longer, and that medium-sized jobs consume most of the resources. They also refer to the trend of utilization of resources, indicating that users keep overestimating the run times of their jobs, requesting wider wallclock. Another conclusion on this last topic is that user jobs tend to be similar, using similar resources, but users with similar resource consumption may have different queue times for their jobs.

Exploring further methods to reduce the overall execution time of the jobs, Giménez de Castro et al. [10] evaluated the impact of combining tasks into a single – that is, *task aggregation* – to reduce the total execution time of the jobs by shortening the queueing time. The objective of their study is to define a theoretical base that could lead to the reduction of the total time of the execution of workflows applied to the Earth Sciences use cases, and understanding how supercomputers are utilized fundamentally drive if this technique is useful or not.

They performed an experiment using "dynamic workloads – where job arrival time plays a role – with a workflow composed of multiple jobs and a static workload – where all jobs in the workload are submitted at the same time – varying job and user factors that play a role into the scheduling" utilizing a Slurm Workload Manager simulator.

The conclusions they reached serve as the theoretical foundation of our project: task aggregation is generally beneficial in most scenarios with vertically structured workflows, whereas its effectiveness in horizontal workflows depends on the user's previous usage and the current state of the machine.

2.2 Earth System Models

Gettelman and Rood [14] explain in their book that "a model, in essence, is a representation of a system". Models can be physical or abstract, and abstract models often are mathematical. Earth System Models (ESMs) [11], in particular, are complex numerical models that integrate traditional elements of climate models, including atmospheric and oceanic physical models, with additional components for simulating sea ice, land, and optional elements such as biophysical and biogeochemical processes or the most advanced treatment of aerosols.

The significance of ESMs stems from our need to understand the complex interaction of the components in a changing climate environment [13]. According to Döscher et al. [11], they are "the primary source of information for understanding the Earth's climate feedbacks, for attributing changes to specific drivers, for future climate projections and predictions, and for the development of mitigation policies". The extensive scale and intricate nature of these models require collaboration between various research institutes and universities for their development and maintenance. An example of this is EC-Earth3, developed by the European research consortium *EC-Earth*. It is a comprehensive Earth System Model that allows the integration of various climate models to create different configurations.

The Earth Sciences Department at the BSC is heavily involved in the development of EC-Earth3, but also makes use of other models such as MONARCH [15], HERMES [16], and CALIOPE [17].

2.2.1 EC-Earth3

This study will focus on analyzing the impact of task aggregation within the EC-Earth3 model, in particular. This choice is primarily motivated by its significant contributions to the CMIP6 project and the subsequent influence it has on the scientific community and policy-making. In addition, its workflow is ideally suited for our research. In the next chapter, we will provide a detailed discussion of its features.

EC-Earth3 [11] is a modular ESM that includes model components for the atmosphere, ocean, sea ice, land surface, dynamic vegetation, atmospheric composition, ocean biogeochemistry and the Greenland ice sheet. Table 2.1 outlines the individual components and the domains they cover, according to Döscher et al. [11].

Component	Domain
ECMWF's IFS [18]	It covers atomosphere and land, and includes a coupling interface to facilitate boundary data exchange with other components.
NEMO [19]	It models the ocean component. It integrates LIM [20] and PISCES [21] to encompass both ice and biogeochemical processes in the ocean, respectively.

Continued on next page

Table 2.1: Overview of the EC-Earth3 model components, according to Döscher et al. [11].

LPJ-GUESS [22, 23]	Dynamic vegetation, land use, and terrestrial biogeochemical processes.
TM5 [24]	Atmospheric chemical processes and aerosols.
PISM [25]	Greenland ice sheet.

Table 2.1: Overview of the EC-Earth3 model components, according to Döscher et al. [11]. (Continued)

EC-Earth3 provides various parameters for different model configurations and different HPC platforms during both build and run times. Additionally, it centralizes all the initial and forcing data files for the entire EC-Earth community.

Executing any configuration based on those components in a concurrent way requires message-passing with common data structures and a coupling interface. EC-Earth3 follows a Multiple Program Multiple Data (MPMD) approach, so it needs Message Passing Interface (MPI) to synchronize the execution of its components. To facilitate coupling, the *OASIS3-MCT* library [26] was created to enable the exchange of multidimensional coupling fields among various models on their respective grids. The *grid* refers to how the model segments the Earth into smaller parts for simulation. A finer grid, which means having a higher resolution, generally implies a better accuracy [27], but demands more computational resources.

Furthermore, it is necessary to standardize the results. To do so, EC-Earth3 employs Climate Model Output Rewriter (CMOR) to convert the output of the models into a "CF-compliant" (Climate and Forecast) netCDF format, ensuring that they meet the requirements of the CMIP projects. The process of standardizing the output is referred to *cmorization*.

2.3 Workflow Managers

HPC applications can be made up of a few to thousands of jobs with interdependencies that must be dispatched in a specific order. Manually submitting every single task from a workflow of such size to the HPC platforms, taking into account their dependencies, and solving possible errors during execution is not viable without an automated processing tool, referred to as a *workflow manager*.

The main purpose of a workflow manager is to coordinate the execution of a *workflow*, minimizing human intervention as much as possible. Although there exist many workflow engines, only a few of them are specially designed to meet the needs of climate research, and some of the well-known in this field are Autosubmit [9], by BSC (Spain); Cylc [28], by NIWA (New Zealand); and ecFlow [29], by ECMWF (international). Each of them has its strengths and weaknesses and is implemented with particular use cases in mind.

2.3.1 Autosubmit Workflow Manager

The experiments in this project will be executed with Autosubmit 3, developed by the Computational Earth Sciences (CES) group at BSC. Initially launched in 2011, it is tailored for earth sciences applications, but serves as a versatile general-purpose workflow manager.

Autosubmit consists in a Command Line Interface (CLI) application built in Python that integrates the capabilities of an experiment manager, workflow orchestrator, and

monitor in a self-contained application. The experiment manager allows the definition and configuration of experiments, supported by a hierarchical database that ensures reproducibility and traceability. The orchestrator is designed to run and monitor complex workflows in research and operational mode by managing their dependencies and interfacing with local and remote hosts. These workflows can involve many tasks that have to be executed on one to multiple platforms. It is a robust software that can handle a variety of setbacks, being able to recover automatically from, for example, punctual network or I/O errors.

It is currently used in the Earth Sciences Department of BSC to run the majority of models, such as EC-Earth3, MONARCH, and CALIOPE, and also operational toolchains or data download workflows. Over the years, it has been used on external HPCs hosted at centers including CESGA (Spain), CSC (Finland), LuxProvide (Luxembourg), ECMWF (UK), OLCF (US), and KIT (Germany).

It has contributed to various European research projects and operates in different operational systems. During the following years, its newer version (Autosubmit 4) will support some of the Earth's Digital Twins as the Digital Twin of the Ocean or the Climate Adaptation Digital Twin in the Destination Earth [4] initiative from the European Commission.

In addition to the typical features of a workflow manager, Autosubmit integrates a variety of tools aimed at developing and enhancing efficiency. We are especially interested in its **wrapping capabilities** and the possibility of using **extended headers and footers** for our research. The first one, to submit multiple jobs at once to the remote supercomputing platforms, and the latest, to stick a custom script to the simulation jobs in order to collect status data from the scheduler.

2.3.2 Aggregation levels for climate research

As Autosubmit is specifically tailored for climate research, tasks within a workflow adhere to a hierarchical structure that serves as an abstraction layer for this particular application. Therefore, Autosubmit is capable of running climate simulations that include various *startdates* and *members*. The process of executing a member involves running multiple *chunks*, which can be subdivided into *splits*.

- *Startdates* represent the initial aggregation level within the Autosubmit workflow hierarchy. They serve to group different instances of the simulation cycle differentiated by the initial conditions. Each *startdate* encompasses as many elements as *startdates* specified in the configuration.
- *Members* represent the second level of aggregation. They are used to group different instances of the simulation cycle, which differ by minor changes in the initial conditions, known as perturbations.
- The *chunks* represent individual sequential iterations within the Autosubmit simulation cycle. Dividing the simulation into smaller sequential pieces aims to fit the resource allocation limits provided by the underlying computer platform and to create implicit checkpoints for restarting the simulation in case of execution errors.
- *Splits* represent the most granular level in the hierarchy, and are employed when tasks at the *chunk* level require additional breakdown.

2.3.3 Architecture of Autosubmit

Autosubmit is composed of three primary components: the Autosubmit CLI core, its Application Programming Interface (API), and its Graphical User Interface (GUI) [30]. Figure 2.1 illustrates the particular distributed deployment at the CES group at BSC, and how Autosubmit operates as an interface between its users and the other components of the system.

The first components to be emphasized in that schema are the remote schedulers, where Autosubmit dispatches the jobs in the user's workflow. This workflow manager supports submission to multiple platforms, from a single job to multiple at once. As mentioned above, with its wrapping functionality, it also allows us to submit sets of jobs that will be treated as a single unit by the scheduler in an automated way.

Autosubmit also requires a technical infrastructure to hold all the experiments. For that, it has a SQLite [31] experiment database, where each experiment is uniquely identified by its *expid*, or experiment id. All data from the experiment will be stored on the file system and all the resources with which the user normally interacts will be located in a dedicated directory for each experiment. That directory will host the experiment configuration files, the project itself – that is, the model files – along with automatically generated logs and reports. All of this information is used to generate useful analytics for both users and developers.

Users can check the progress and status of their experiments through the Autosubmit GUI, which communicates with the Autosubmit API to handle all the requests.

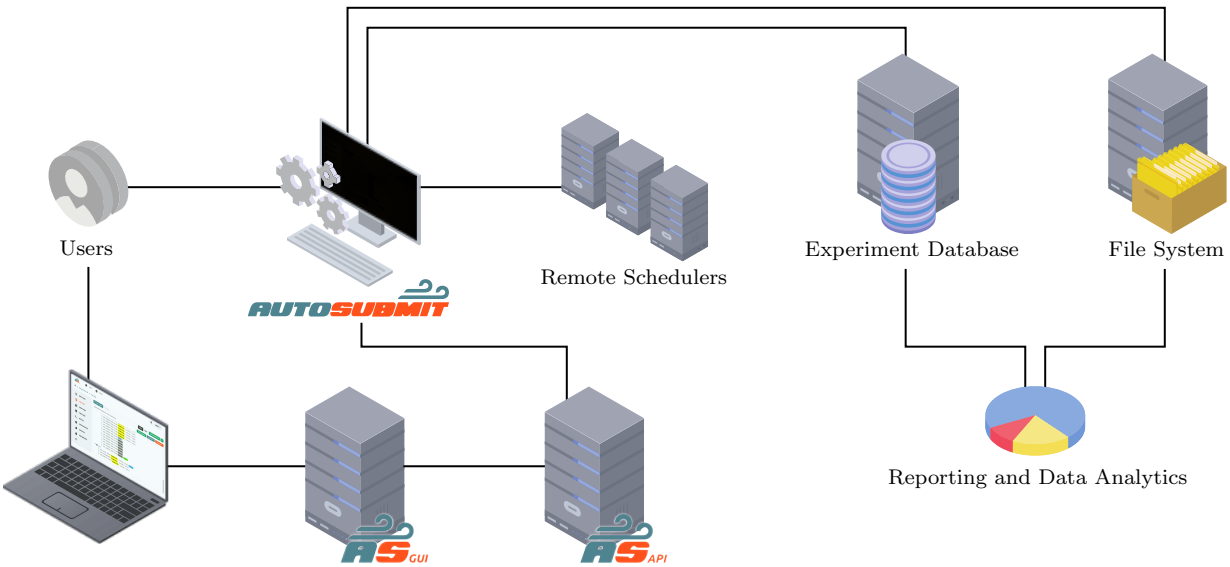

Figure 2.1: The distributed architecture of the Autosubmit Workflow Manager at the CES group at BSC.

It is important to note that Autosubmit can be deployed locally in a non-distributed way, thus the diagram above serves merely as reference.

2.3.4 Experiment logical organization

Autosubmit creates the directory structure shown in Figure 2.2 throughout the life cycle of the experiment.

The first folder to highlight is *conf*, which contains all the *INI files* [32], a kind of configuration file that stores key-value pairs for attributes arranged in sections. The next most important folder is *proj*. It is dynamically created while creating the experiment, a process that involves cloning the project source, which contains user scripts and project code. In our case, it will include the executables and templates of an ESM, along with a custom script to monitor the scheduler's status during workflow execution on the remote platform.

Autosubmit uses the *tmp* directory to save logs and store the scripts that will be submitted to the HPC machines, as well as other miscellaneous files. Additionally, the *status* and *pkl* directories are used to store the status of the workflow jobs and information about the experiment workflow. Lastly, *plot* contains all visualization output files, that is, the workflow diagrams.

```
/autosubmit
└ /expid
    ├ /conf
    ├ /pkl
    ├ /plot
    ├ /proj
    ├ /status
    └ /tmp
```

Figure 2.2: The directory tree of an Autosubmit experiment.

2.4 Auto-models

Auto-model is the term used in the Department of Earth Sciences at BSC to refer to an Earth System Models that has been adapted for execution with the Autosubmit workflow manager. For instance, the auto-model corresponding to EC-Earth3 is called *Auto-EC-Earth3*. Each of these climate models has a dedicated team of *auto-modelers* responsible for maintaining the workflow, porting it to different platforms, and analyzing and enhancing its performance.

2.4.1 Auto-model configuration

While the most intricate aspect of the auto-model is located in the experiment's *proj* directory, regardless of the auto-model we select, our study will not require an in-depth examination of it. However, it will be essential to thoroughly understand the specific configuration files for the experiment found in the *conf* directory.

The configuration files for an experiment are *autosubmit*, *expdef*, *jobs*, *platforms*, and *proj*. The details of their content are provided in Table 2.2.

INI file name	Content
autosubmit_[expid].conf	Includes parameters to control the workflow behavior or enable additional features, such as wrappers or email alerts.

Continued on next page

Table 2.2: Description of Autosubmit configuration files.

expdef_[expid].conf	Defines *startdates*, *members*, *chunks* (indicating number and size), or even *splits*. It also contains the source of the experiment (such as a Git repository or a local folder) and the path of the project configuration file.
jobs_[expid].conf	Defines the workflow to be executed: the scripts to run, job dependencies, computational resources required for the job on the HPC (or local) platform, and the platform to which each task will be submitted. Additionally, it allows to override parameters previously defined in the platforms configuration file, and supports additional functionalities such as the extended header and footers we will use later on.
platforms_[expid].conf	It hosts the configurations for each remote HPC platforms the workflow will use, such as the *hostname*, *username*, *scheduler*, *account*, *QoS* and other technical details.
proj_[expid].conf	Holds project-specific variables that Autosubmit will replace in the job scripts that are submitted.

Table 2.2: Description of Autosubmit configuration files. (Continued)

2.5 Slurm Workload Manager

Slurm [33] is a widely recognized workload manager, notable for its open-source nature, fault tolerance, high scalability and modularity. It is currently utilized by prominent TOP500 [2] supercomputers such as Frontier, LUMI, Leonardo, and MareNostrum 5.

Among its features, Jette and Wickberg [33] emphasize its open-source nature and user-friendliness for system administrators, its portability due to being written in C, and its flexibility through the integration of numerous plugins thanks to its high modularity. It is highly scalable, as it consists of highly concurrent daemons designed to support large supercomputers with thousands of scheduled jobs. It includes some fault tolerance, allowing jobs to continue even if some nodes fail. Slurm also incorporates security features to authenticate each communication, ensuring that users or processes cannot impersonate each other.

2.5.1 Scheduling

Depending on the plugin selected in the *slurm.conf* file [34], Slurm offers *basic* or *multifactor* scheduling behavior. This affects how Slurm assigns priorities to jobs, which are computed as integer values.

Basic scheduling

Slurm employs a First In, First Out (FIFO) scheduling strategy to serve the highest-priority jobs within a queue sorted in reverse order based on their arrival time.

This algorithm can be particularly advantageous for High Throughput Computing (HTC), where the objective is to complete the maximum number of tasks in the least amount of time. Slurm, as noted by Jette and Wickberg [33], can get a throughput rate of up to 100 jobs per second with an optimal setup and hardware. In contrast, if any job is found unable to start, those with lower priority on the same partition will remain pending. The FIFO scheduling approach may also introduce several issues, as highlighted

by Almaaitah et al. [35] in their paper: one of these is the *convoy effect*, where smaller tasks experience significant delays because they must wait for a larger task to release resources. Another issue is fragmentation, which occurs when the availability of free resources, such as processors or memory, is less than required by a job, causing the job to remain in queue until sufficient resources are freed by other jobs or to reach a *starvation* [36], a situation of indefinite blocking of a task. Moreover, they mention that if shorter tasks are prioritized over longer ones, it would lead to inefficient resource utilization.

Multifactor scheduling

In HPC systems, it is typical to see the *multifactor* scheduling plugin activated. This plugin is an implementation of an algorithm that considers various parameters when prioritizing tasks or organizing the queue. Among the parameters considered when calculating priority are *age*, *fairshare*, *job size*, *QoS*, *partition*, *niceness*, *association*, and a site-managed value. Each of these factors can be configured so that some have greater influence than others in establishing priority. [33, 37]

The *age* factor determines the duration for which a job has been in the queue while being qualified for schedule, and the size of the job is quantified as the amount of resources required by the task.

The *queue* component is linked with each node partition, much like *QoS* is related to the defined level of service quality, involving some parameters such as the maximum number of jobs per user. Conversely, the user may wish to assign a task priority, known as *nice value*, similar to the niceness in process scheduling within an operating system [36]. The determination of the site factor is assigned to the priority algorithm determined by the system administrator.

Fairshare represents the discrepancy between the allocated – or promised – resources and the consumed ones, ranging from 0 to 1. It seeks to measure if an association – the tuple *user, account, partition, cluster in Slurm* – is receiving more or fewer services compared to the resources allocated within a specified time frame. The objectives are to ensure fairness and to maximize machine utilization, while avoiding rigid boundaries. The standard algorithm used for *fairshare* calculation is the *Fair Tree*.

The main principle around which *Fair Tree* is to allow sharing responsiveness among users of the same *account*, which are groups of users, normally on a per project basis. The *Fair Tree* algorithm computes the *Level Fair Share*, which pertains to resource distribution across the hierarchical tree levels, and considers factors as the *Raw Usage*, which concerns the user's historical machine usage, and *Raw Share*, which is related to the potential usage by the user. The thesis by Giménez de Castro et al. [10] delves deeper into the behavior of fair share algorithms in Slurm.

Backfill scheduling

Slurm integrates a pre-enabled backfill plugin. The purpose of this backfill mechanism is to minimize fragmentation, allowing the initiation of lower-priority jobs as long as they do not alter the anticipated start time of higher-priority tasks, allowing jobs to run out of strict priority order when enough resources are available. [38]

2.6 Wrappers

In the task aggregation domain, a wrapper is a larger job composed of smaller ones that can be submitted to a workload manager, such as Slurm.

The most elemental types of wrapper are *vertical* and *horizontal*. Figures 2.3 and 2.4 represent how sets of black-box jobs are vertically and horizontally wrapped, respectively. Jobs inside a vertical wrapper follow a hierarchy in which each one depends on its previous, while in a horizontal wrapper, jobs have no dependencies between themselves. In the first case, jobs must be executed sequentially to meet the dependencies, but in the latest, jobs run concurrently.

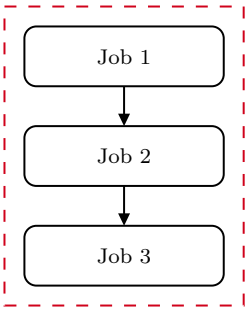

Figure 2.3: Vertical wrapper example. Each job within the wrapper depends on the one before it.

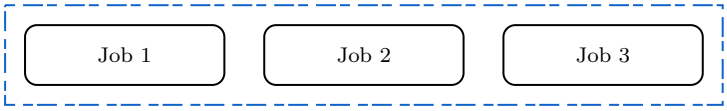

Figure 2.4: Horizontal wrapper example. Tasks inside the wrapper execute independently and concurrently.

It is possible to merge these elemental types of wrapper to create more flexible wrapping techniques: *horizontal-vertical* and *vertical-horizontal*.

The *horizontal-vertical* wrapper, shown in Figure 2.5, enables the concurrent submission of sets of tasks that must synchronize before advancing to the subsequent horizontal stage. In other words, it allows for the dispatch of horizontal wrappers organized within a vertical wrapper.

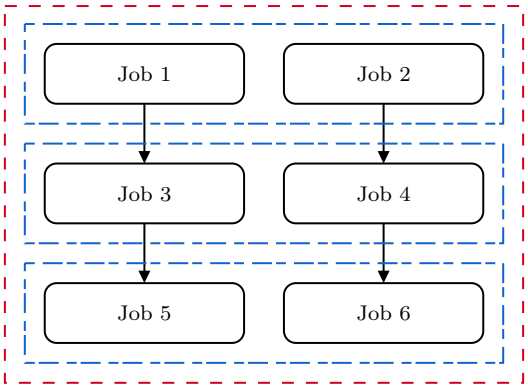

Figure 2.5: *Horizontal-vertical* wrapper, combining horizontal wrappers into a single vertical wrapper.

The *vertical-horizontal* wrapper, depicted in Figure 2.6, enables the packaging of a vertical sequence of tasks that are independent of the horizontal ones. Consequently, it is not necessary for all horizontal tasks to be completed before moving on to the next horizontal layer.

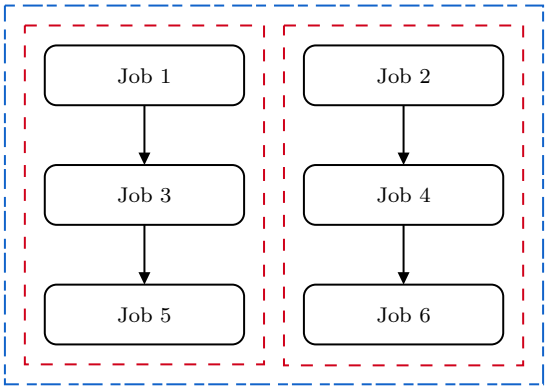

Figure 2.6: *Vertical-horizontal* wrapper example, combining vertical wrappers into a single horizontal wrapper.

The Autosubmit workflow manager supports all these types of wrapper, which must be defined in the *autosubmit_[expid].conf* INI file with the *wrapper* attribute [39], indicating the type of wrapper, its jobs, and the minimum and maximum size. It also implements different wrapping policies. We must select one between *flexible*, *mixed* and *strict* depending on our needs. The *flexible* policy is the most lax because if there is not a minimum of jobs to be grouped, they will be sent to the platform individually anyway, while the *strict* policy always waits until there are enough tasks to create a wrapper. The *mixed* one will wait for a minimum number of jobs to create the wrapper, except in the cases where a task has failed.

In the following snippet, an example set-up where vertical wrappers are applied to a collection of jobs called "SIM", which stands for *simulation*, with a wrapper size between 5 and 10 jobs, following a flexible policy (default).

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 5
MAX_WRAPPED = 10
POLICY = flexible
```

Code 2.1: Example of wrapper configuration in Autosubmit 3.15.

Chapter 3

Methodology

3.1 Overview

The main objective of this work is to analyze the impact of task aggregation in real-life simulations on different HPC platforms. To do so, in this chapter we will go through a series of steps that involve, first of all, conveniently choosing the auto-model we want to execute. Subsequently, we will request access to the supercomputing platforms that will run our simulations.

3.2 The auto-model

3.2.1 Selecting the auto-model

We are seeking an auto-model able to run for an extended period, enabling us to encounter various usage patterns and time frames, including day and night cycles, weekends, and periods of both low and high platform demand. Furthermore, the auto-model must be executable on a *broad range of supercomputing platforms* so that we can determine which are the most appropriate for our research. Additionally, the model-platform pair should be *commonly utilized by Earth scientists* at the BSC, as this would enable us to clone existing – and tested – experiments as a basis for ours by adapting them to meet certain needs, setting up the usage of wrappers, changing the number of chunks to simulate and its size, and removing the overhead produced, for example, by the file transfer between the remote platform and the host. Otherwise, we would have to build our experiments from scratch, and that implies creating a new Autosubmit experiment that uses a pre-compiled version of the auto-model's base adapted to the platform we want to use, and it is undoubtedly better for us to take advantage of the work that has already been done than delving that far into the auto-model's configurations with the complications that it entails. Moreover, having *high-quality documentation* is crucial as it allows us to resolve issues independently without needing support from the auto-modelers.

Different auto-models have been evaluated throughout the project, including EC-Earth3's and some of its models separately. In the end, we opted for *Auto-EC-Earth3*, which perfectly fulfills our requirements. Besides the EC-Earth3 model, we also explored other workflows, such as the NEMO *standalone*, which also met the criteria, ported to LUMI, the flagship supercomputer at the CSC, in Finland. Although we gained access to the platform, we finally discarded this particular possibility, finding it more practical to use the same model across all platforms, thereby reducing the time required for learning and research before conducting any experiments.

We will now outline the rationale behind selecting *Auto-EC-Earth3* for our research:

- It is currently running on a wide variety of powerful supercomputers across Europe, such as MareNostrum 4, hosted at BSC in Spain; MeluXina, hosted at LuxProvide in Luxembourg; and HPC2020, hosted at ECMWF in Italy. In addition, since the internal pre-release of MareNostrum 5, prior to the power-off of MareNostrum 4, auto-modelers have been working on porting the auto-model to BSC's newest platform.

- Its developers execute a testing suite periodically on the remote platforms to ensure that all the updates they perform over the auto-model are correctly working on real life scenarios. That testing suite is composed by special Autosubmit experiments called "test cases", which have a given number of chunks, start dates, members and HPC platform, and are differentiated from others because they have a reserved first letter "t" at the expid [40]. Each testing case aims to simulate a particular model configuration on an specific HPC. Auto-EC-Earth3 developers generate a new tag in the GitLab repository nearly every week and validate its operation by running this testing suite.

  In other words, we have a set of reliable experiments for different supercomputing platforms that can be used as the basis for ours. In addition, having experiments that someone has already executed as a reference will allow us to know the approximate consumption in *CPU hours* for each task in the workflow. This will allow us to adjust the size of our workflows to adapt them to the resources available on each platform. *CPU hours* serve as a metric to measure the resource consumption of our tasks.

- It has high-quality and extensive documentation that provides detailed insights into the auto-model's internal operation, step-by-step guidance for running an Auto-EC-Earth3 experiment, platform-specific usage instructions, including a tutorial for porting the auto-model to a new HPC, a section on Frequently Asked Questions addressing common issues, among other resources for its developers.

  Unfortunately, this documentation [41] is hosted in the GitLab repository of the project, which is private because EC-Earth3 is partially based on OpenIFS, a component that requires a license from ECMWF [42].

3.2.2 Auto-EC-Earth3

Auto-EC-Earth3 is the software package that contains all the files required to run the EC-Earth3 model with Autosubmit.

Includes essential files such as templates, configuration files, supporting software, and even the EC-Earth3 code itself. Thanks to Autosubmit capabilities, Auto-EC-Earth3 offers an automated solution to create, manage, and monitor EC-Earth3 experiments on various HPC platforms, handling everything from model compilation to post-processing of output files [41].

The workflow

Auto-EC-Earth3 is composed of a set of tasks with different purposes, including setup, synchronization of the file system, simulation, postprocessing of results, or diagnostics. Presented in Figure 3.1 is a streamlined version of the workflow capable of simulating up to S *startdates*, containing M *members* with N *chunks*.

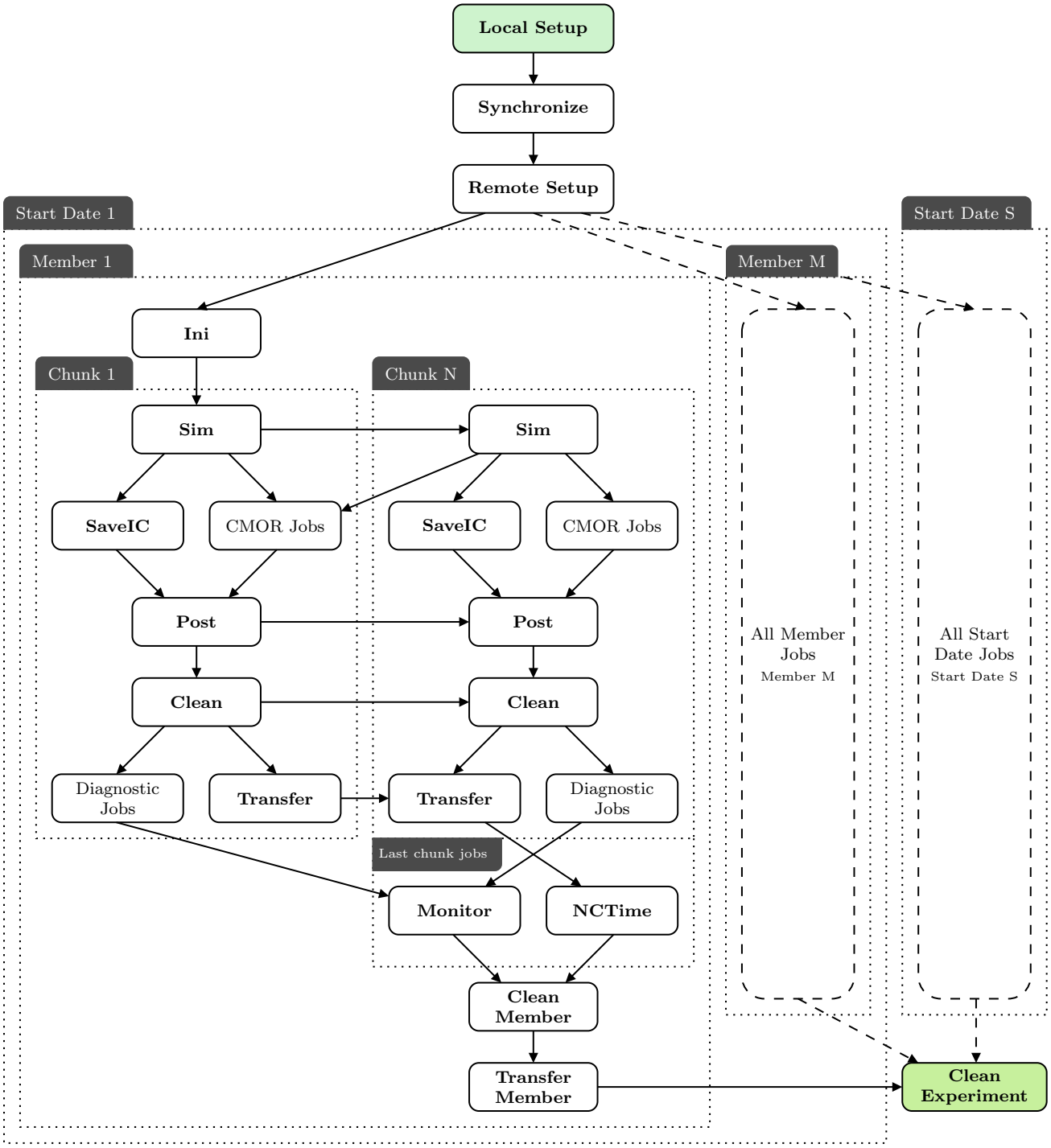

Figure 3.1: A simplified view of an Auto-EC-Earth3 workflow with its common tasks.

Tasks are organized to ensure that the workflow can be easily scaled, taking into account its numerous interdependencies.

There are three common tasks independently of the number of *startdates*, *members*, and *chunks* that are responsible for initializing the execution of the workflow locally and remotely. In real experiments, they will be named *LOCAL_SETUP*, *SYNCHRONIZE*, and *REMOTE_SETUP*. There is another exclusive initialization task per member (*INI*). Within the chunk, there are several tasks focused, initially, on simulating (*SIM*) and, subsequently, on post-processing the output of the simulation. These tasks are interdependent, sometimes even between other chunks in the same member, and there are also other chunk-related tasks which depend exclusively on the last chunk.

Figure 3.1 visually illustrates the complexity of why a climate system model needs to be executed using a workflow orchestrator such as Autosubmit, as seen in Chapter 2.3.

The following table provides a brief overview of the roles of the workflow tasks seen above [43]:

Set of tasks	**Description**
LOCAL_SETUP	Checks configuration files and creates the template files for the *SIM* jobs.
SYNCHRONIZE	Sets up the scratch experiment folder at the HPC platform.
REMOTE_SETUP	Compiles the model components.
INI	Sets up inidata folders for each member.
SIM	Runs the simulation.
SAVEIC	Creates the requested initial conditions.
CMOR jobs	These are several jobs that can be executed separately for the *cmorization* of the outputs.
POST	Post-processes the output of the simulation.
Clean jobs	These are the jobs responsible for deleting the chunk files (*CLEAN*) or the member directories (*CLEAN_MEMBER*).
Diagnostics jobs	These are specialized jobs to compute diagnostics. There exists both *EARTHDIAGS* or *ESMVALTOOL* jobs, each associated with a different diagnostics tool.
Transfer jobs	*TRANSFER* and *TRANSFER_MEMBER* jobs move chunk- or member-related files to local or intermediate storage.
MONITOR	Generates plots from diagnostics for online monitoring of the simulations.
NCTIME	Checks the time consistency and completeness of the CMOR outputs.

Table 3.1: A brief explanation of the Auto-EC-Earth3 jobs.

3.2.3 An approach to what our experimentation requires

As discussed in Section 3.2.1, our experiments will be based on test cases from the Auto-EC-Earth3 testing suite. These test cases differ in the number of *startdates*, *members*, *chunks*, the HPC, and even the climate models they couple. For simplicity, we will define the same workflow for all platforms and adjust its configurations or try different coupled model combinations depending on the available resources on each one. For that reason, it is essential to note that this analysis aims to evaluate task aggregation for each HPC platform individually, without performing cross-platform performance comparisons. Given that the experiments on each platform will not be identical, such a comparison would be unfair.

Figure 3.1 presented a complete Auto-EC-Earth3 workflow, with all its tasks and their interdependencies. Experiments in real life typically follow that schema, although certain tasks may be omitted. However, our study only needs to focus on large computationally intensive jobs, like the *SIM* tasks, while other tasks, involving transfer, cleaning, monitoring or post-processing, can be regarded as *overhead*, as we do not need to handle the climate model outputs, and discarding them will allow us to save valuable CPU hours,

memory, storage, and network bandwidth. Then, we can proceed to prune all the unnecessary jobs in the execution tree, leaving only those ones that are required to initialize the experiment locally and remotely, and the simulation tasks themselves. It would not be necessary for our research to have multiple *startdates* or *members*.

Putting these requirements together results in the simplified workflow represented in Figure 3.2, consisting of *LOCAL_SETUP*, *SYNCHRONIZE*, and *REMOTE_SETUP* for workflow initialization, *INI* to initialize the member, and as many chunks composed only by its *SIM* job as we can simulate on each platform. The quantity, denoted as N in the figure, will depend on the number of CPU hours granted by the project managers and the CPU hours consumption of each *SIM* task.

Furthermore, trimming the generic tree makes it more apparent that simulation tasks are vertically organized and can be easily combined into wrappers.

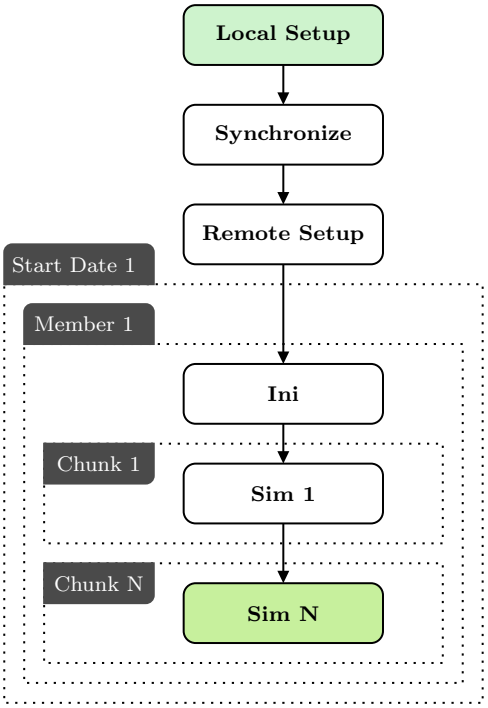

Figure 3.2: Our custom Auto-EC-Earth3 workflow with a single startdate and member and N chunks. Chunks have been simplified to be reduced to its minimum expression: the simulation job.

3.2.4 Wrapping Auto-EC-Earth3

Our investigation requires applying task aggregation to the large computationally intensive jobs we had previously mentioned: the *SIM* jobs. The wrapper size and policy would vary on each experiment depending on the number of simulation jobs we are able to execute in.

If we take the workflow in Figure 3.2 and group the *SIM* jobs into wrappers of, for example, 4 jobs, also following the *strict* Autosubmit policy, the resulting workflow would be as represented in Figure 3.3. The Autosubmit configuration required to set up this specific wrapper is also shown in Code 3.1.

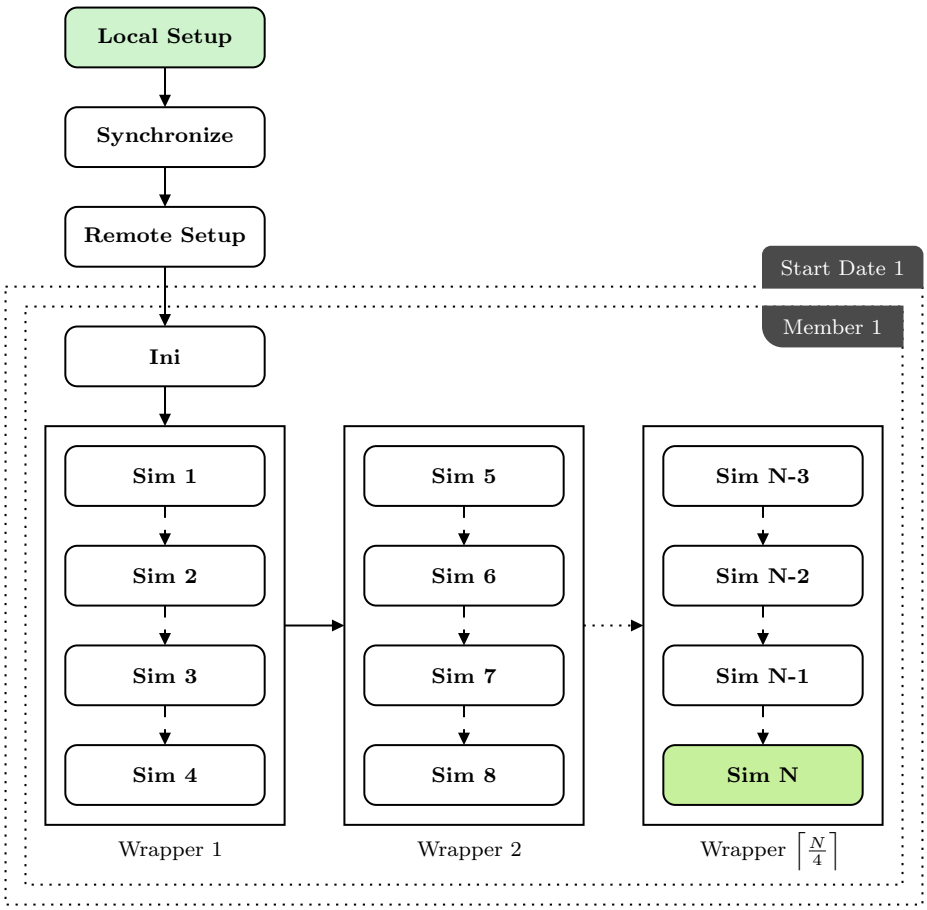

Figure 3.3: The custom workflow in Figure 3.2 with N chunks consisting solely by *SIM* jobs split into 4 job wrappers.

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 4
MAX_WRAPPED = 4
POLICY = strict
```

Code 3.1: Autosubmit wrapper configuration for the workflow in Figure 3.3.

3.3 Collecting metrics

Autosubmit records comprehensive details related to the workflow execution, including the dispatch dates of tasks, how long they were in the queue, and the execution times, in SQLite databases. The capability of Autosubmit to collect these data alleviates the burden of collecting metrics manually.

For our research, we require not just the different times mentioned above, but also keeping track of the status of the platform while our workflow is running. We will collect three *share* parameters from the platform to represent the status of the platform. Although Autosubmit does not offer a direct method to do it, it integrates an *extended header* functionality that we can utilize to append a custom code snippet to the header of each designated job.

Then, to obtain these *share* parameters from the platform, we will write a script that reads and appends them to a CSV file. Initially, the script will call *sshare* [44] to fetch the three required metrics. This call to *sshare* will differ based on the platform and will require specifying the user and the *account* for each.

The Code 3.2 contains a *sshare* command which filters by the three metrics susceptible to be analyzed, without considering either the user or the account; thus, the amount of information shown may vary by platform.

```
sshare --format=RawUsage,FairShare,LevelFS,Partition
```

Code 3.2: The *sshare* command to retreive all the usage, *fairshare*, and *level fairshare* metrics from Slurm. The system administrator can restrict the information it returns.

It is important to note that the command output must be properly formatted according to the HPC platform in order to correctly dump the parameters in the CSV file, considering that the default output may vary depending on the Slurm version.

The remainder of the script should identify the currently running job's name and link it with a UTC timestamp. The UTC time simplifies the output understanding when running in supercomputers across different time zones. Additionally, it will determine the path to the CSV file to append the new entry. This can be achieved by utilizing the environment variables that Autosubmit generates for its scripts; hence a deep examination of Autosubmit's source code is necessary. Finally, the data of each entry is appended to the CSV file following the format "*Job Name*,*Start Time*,*Raw Usage*,*Fair Share*,*Level FS*".

The complete bash script can be found in Code A.1 of Appendix A.

3.4 HPC Platforms

Auto-EC-Earth3 is a well-known workflow regularly used in production experiments and, for this purpose, it is executed on a wide variety of HPC platforms. To increase the reliability of our conclusions we will execute our simulations on various platforms where the auto-model has been ported. Therefore, we will run simulations in MeluXina, HPC2020, and both MareNostrum 4 and MareNostrum 5, although the latter will not be accessible until shortly before the project's conclusion.

Now that we know which platforms we will use, it is time to carefully prepare the execution of the simulation platform. Each platform has its particularities, software stacks, features, architecture, security restrictions, submission mechanisms, and even a long list of known common errors for given applications, so we must adjust our experimentation to each one. Moreover, we must keep in mind that resources are neither unlimited nor free, so the scale of our experiments will have to be meticulously adjusted to the portion of the account, or CPU hours, that the project managers allow us to use. Our simulations will allocate a considerable amount of resources and will be remarkably extensive to accurately evaluate the effects of task aggregation; therefore, we need to be very careful.

As an additional step, it is very important to take into account that Autosubmit requires to be able to establish password-less connections to all the utilized platforms. This can be done by creating a key pair and adding a new entry to our user's *.ssh/config* file, specifying the hostname, the port, the user, and the path to the key for authentication, among other optional parameters following the example above:

```
Host mn_login1
    HostName glogin1.bsc.es
    User [user]
    IdentityFile ~/.ssh/id_rsa_mn
```

Code 3.3: Example of host entry inside the user's *.ssh/config* file. It enables access to the host as the specified user using its *rsa* key.

3.5 Experimentation

The methodology involves running one workflow with wrappers and another without, which serves as a reference. To ensure that both the wrapped and reference experiments begin under the same conditions – or Slurm share parameters, namely, *fairshare* 2.5.1 –, they must be conducted simultaneously. Therefore, we will coordinate two user accounts per platform, with each account running a single experiment.

In the subsequent sections, we will describe these supercomputing platforms, how we managed to access them, the experiments to be run, and the different faced problems.

3.5.1 MareNostrum 4

Platform overview

MareNostrum 4 is a Spanish supercomputer that was operational from June 2017 to April 2024. It belongs to a lineage of supercomputers that traditionally have been the most powerful in Spain and have achieved notable positions in the TOP500 list, reaching 13th place within that list after its launch[1]. The MareNostrum 4 supercomputer serves as an excellent resource for conducting research and generating knowledge. It is utilized across various fields, including physics, biomedicine, earth sciences, engineering, and industry.

The system includes a general-purpose partition containing 48 racks and 3,456 nodes, each node being equipped with two Intel Xeon Platinum processors, having 24 cores each. Together, the general purpose block sums up to 165,888 cores, capable of delivering up to 11.15 PetaFLOPS, or 11.15×10^{15} Floating Point Operations per Second. Furthermore, there is a dedicated block for emerging technologies, which includes an MN4 CTE-Power cluster equipped with IBM POWER9 processors and Nvidia Volta GPUs, an MN4 CTE-AMD cluster with AMD Rome processors and AMD Radeon Instinct MI50 GPUs, and a last MN4 CTE-ARM cluster featuring ARMv8 processors. [45]

MareNostrum 4 is the first platform we will use and there are many reasons that motivate this decision. Firstly, we have previous experience [46] in running various workflows on it through Autosubmit, so we already have knowledge about its operation. This experience will be particularly beneficial for troubleshooting issues that could arise during executions. In addition, it is the most utilized among the scientists in the Department as it is hosted by the BSC itself, and we can use the experience of our colleagues. The last, but not least, is the fact that this project started in late February, when MareNostrum 5 was almost ready for production, and MareNostrum 4 was reaching the end of its life cycle, so it was a priority to get results as soon as possible, and it would be possible to try different wrapping configurations if there is enough time. Finally, on April 26th MareNostrum 4 was powered off.

[1] https://www.top500.org/lists/top500/2017/06/. Last accessed: July 3, 2024

This platform uses Slurm as its batch scheduler. MareNostrum 4 users report that in certain periods, it can take a significant amount of time for tasks to be processed in the MareNostrum 4 queues, sometimes more than a day. Figure 3.4 illustrates the reason by representing the overall status of the jobs in the scheduler from 3rd to 24th August 2023. Since queues are quite full, then, task aggregation should be notoriously beneficial on this platform. We will check whether this hypothesis holds true, once both the queuing times and the user's share values are obtained.

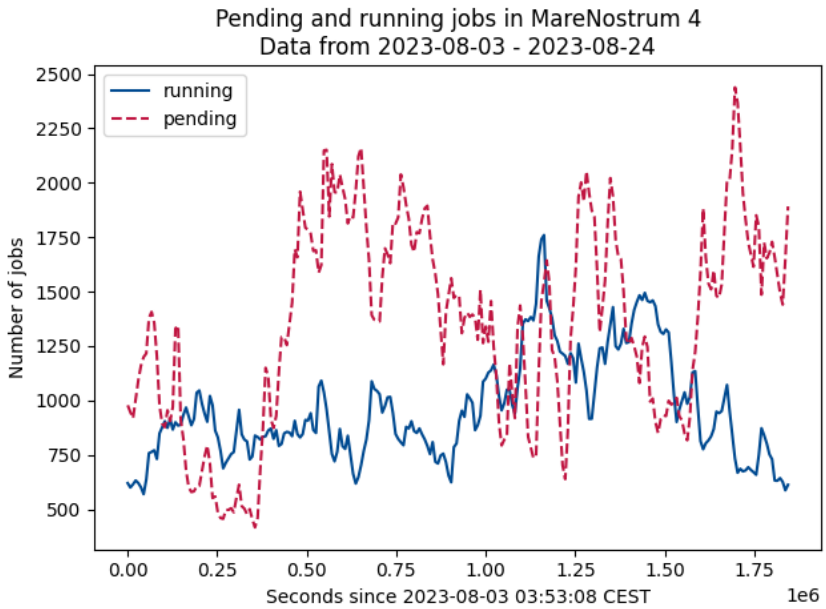

Figure 3.4: Running and pending jobs at MareNostrum 4, from 3rd to 24th August 2023. Graph by Giménez de Castro et al. [10] Data extracted from the BSC operation's HPC portal.

Available resources for our project

For the MareNostrum 4 supercomputer, which is managed by the BSC, we utilize the computational budget of the Earth Sciences department to execute the simulations.

Experiment overview

As previously stated, for a fair comparison, we will have to run one workflow concurrently with wrappers and one without them, which implies creating and configuring two Autosubmit experiments that will be launched from two different user accounts. The objective is for both experiments to begin with an analogous *fairshare* value to monitor their progression as the tasks are launched, hence we will delay running any work on the platform until we are certain that the experiments are entirely functional. One scheduling advantage MareNostrum 4 offers for our research compared to other platforms is that the *fairshare* value is reset at the beginning of each month. This reset ensures that our initial tasks can be placed in the queue as fairly as possible, maintaining parity with other users in the group. Consequently, if at any time our experiments encounter errors and the *fairshare* value becomes skewed between the two accounts, we can stop and restart the experiments the following month to obtain more accurate results.

At the time of calculating the maximum experiment length we can afford for the available computing hours budget, on average, each simulation job *SIM* within this workflow

will consume about 1,663.36 CHSY, or Core Hours per Simulated Year. Thus, for instance, a 50-year simulation would require a total of 83,168 CPU hours. We believe that a 50-year simulation should be adequate for share and performance metrics to demonstrate the effects of task aggregation, whether its impact over queuing times is positive or negative. These 50 years could be conveniently spread over 50 chunks of 12 months, so we could say that each chunk consumes the cited CHSY, and we would comfortably remain under the budget, leaving sufficient allowance to rerun single jobs or even fully rerun the simulations in case of error.

The 50 chunks to be simulated can be divided into 5 wrappers of 10 jobs each. This wrapper size should be large enough to observe significant improvements if our hypothesis regarding vertical wrappers holds true. The wrapping policy will be *flexible* in order to let Autosubmit decide when to wrap or what to do in case a job inside a wrapper fails.

Regarding errors, it is worth noting that, in parallel executions, it is quite common for a node to occasionally fail during runtime, causing the task to fail as well. It is important to consider this aspect when configuring the experiments, as a failure in one of the 50 simulation tasks could lead to the downfall of the entire workflow. Fortunately, Autosubmit provides a mechanism for retrials, enabling the restoration of the workflow execution from the most recent successful task if an error occurs. Given the flexibility we were provided, we will set the maximum number of retrials to 6 for our experiments on MareNostrum 4, a deliberately overestimated value. For other platforms, the configured number of retrials must be considered when determining the resources available for our simulation. This is because the worst-case execution time, assuming 50 chunks of 12 months each, would be $50 \times CHSY \times retrials + 1$.

In summary, we will perform one experiment that incorporates wrappers and another without them, following the schema in Figure 3.2. The experiments will follow an Auto-EC-Earth3 workflow with one of the most complex – and complete – configurations, the T255L91-ORCA1L75-LIM3-PISCES-LPJG-TM5, divided into 50 chunks of 12 months. The simulation jobs in the wrapped experiment will be organized into 5 wrappers with 10 jobs each, and the count of retrials will be set to 6. Each experiment is estimated to require 83,168 CPU hours, totaling 166,336 CPU hours for both, excluding any retrial of tasks that may fail incidentally.

Configuring and launching experiments

Every single operation on Autosubmit CLI will be performed from the Department's Autosubmit virtual machine, which contains environment modules for each Autosubmit version. This is also a reminder that we need to determine which Autosubmit version ran the test case we are copying from to ensure full compability. The test case we selected for our experiments on MareNostrum 4 uses version 3.15.0b, so our experiments will also be executed with that version. The command required to load that specific environment is *module load autosubmit/3.15.0b-foss-2015a-Python-2.7.9*.

From now on, we will focus on creating and properly configuring the wrapped and unwrapped experiments. In order to create the experiments using as base the selected variation of the test case, we will use the *autosubmit expid* command as follows:

```
autosubmit expid -y a6bi -H marenostrum4 -d "My Auto-EC-Earth3 running
↪  with wrappers on MareNostrum 4."
autosubmit expid -y a6bi -H marenostrum4 -d "My Auto-EC-Earth3 running
↪  without wrappers on MareNostrum 4."
```

Code 3.4: The *autosubmit expid* commands to create two new experiments based on *a6bi* to be executed on MareNostrum 4.

The first command retrieved a new *expid* for which will be the wrapped experiment, the *a6zi*. The process is the same for the unwrapped experiment, which has been named *a6zs*.

Now, it is time to configure the experiments. The procedure will first be to configure the *a6zi* wrapped experiment. As we introduced in Chapter 2.4.1, the experiments are configured through the INI configuration files, and Table 3.2 details the modifications applied to each to perform this task.

Configuration file	**Modifications**
autosubmit_a6zi.conf	Defined the wrapper parameters over the *SIM* jobs, with a size between 5 and 10 jobs, following a *flexible* policy. Code 3.5 details the configuration. Email notifications were also activated in order to get notified when a job execution fails. The number of retrials have been adjusted to 6.
expdef_a6zi.conf	Modified to define one unique startdate, *18500101*, and a member, *fc0*. Configured 50 chunks of 12 month each. Code 3.6 lists the menctioned experiment-related configurations. The remaining configurations were left as they was, such as the source Git tag or the specific commit.
jobs_a6zi.conf	It hosted the complete job list of the workflow seen in Table 3.1, but it had to be pruned in order to have only the jobs shown in Figure 3.2: *LOCAL_SETUP*, *SYNCHRONIZE*, *REMOTE_SETUP*, *INI* and the *SIM* jobs. This task is not as simple as it seems, because the interdependency definitions had been had to be removed too, making necessary to change some values in the *proj_a6zi.conf* file. Adding the path to the extended header was also necessary, which will be placed in the project directory, inside the *header* folder, after the sources have been downloaded.
platforms_a6zi.conf	Added the necessary HPC platforms to launch the experiment, which are the MareNostrum 4 login node, which serves as dispatching node, and the data transfer node. Each individual platform has some associated parameters such as the scheduler, the project, the user, or even the queue, which have been left as it was. Another parameter is the maximum wall clock time (*MAX_WALLCLOCK*), which defines how long the resources will be allocated for the job. This duration is set to 48 hours, and if a job runs beyond this limit, Slurm will terminate it.
proj_a6zi.conf	Disabled several flags associated with the jobs that were deleted recently.

Table 3.2: Changes applied to the configuration files of the *a6zi* (wrapped) experiment in MareNostrum 4.

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 5
MAX_WRAPPED = 10
POLICY = flexible
```

Code 3.5: The wrapping configuration for the *a6zi* experiment in MareNostrum 4.

```
[experiment]
DATELIST = 18500101
MEMBERS = fc0
CHUNKSIZEUNIT = month
CHUNKSIZE = 12
NUMCHUNKS = 50
CALENDAR = standard
```

Code 3.6: The experiment configuration for *a6zi* in MareNostrum 4.

Once the configuration for *a6zi* is finished, it is time to apply the same modifications to the *a6zs* experiment, except for the wrapper configuration shown in Code 3.5. Then, we can run the *autosubmit create* command to prepare our workflows for execution. Autosubmit will fetch the project sources and generate the job scripts that will be submitted to the remote platform, including the wrapper scripts for *a6bi*. These commands are listed as follows:

```
autosubmit create -cw a6zi
autosubmit create a6zi
```

Code 3.7: The *autosubmit create* commands to create the *a6bi* and *a6bs* experiments. The "-cw" flag is used to indicate Autosubmit that our experiment will use wrappers.

Since these workflow uses githooks, we had to load the updated Git module. Moreover, due to the use of submodules in EC-Earth to point to the different model and workflow components, we also had to configure the Git credential cache to prevent repeated authentication requests.

To monitor user share metrics in the Slurm scheduler using the extended header, we need to adjust the script found in Appendix A to run the *sshare* command as shown in Code 3.8. This command retrieves the necessary metrics for the current user without needing to specify an account, since our MareNostrum 4 users are associated with only one account. The custom implementation of the script for MareNostrum 4 will be placed within the *header* folder inside the *proj* directory.

```
sshare --format=RawUsage,FairShare,LevelFS -h -P -U
```

Code 3.8: The *sshare* command adjusted to MareNostrum 4.

After confirming that the Autosubmit virtual machine can establish password-less connections to the platforms specified in *platforms.conf*, the experiments can be executed. This involves using the *nohup* and *autosubmit run* commands together. The *nohup* command allows the process to run in the background, remaining active even if the console session ends, and logs the output to a file. The *autosubmit run* command will start the workflow execution.

```
nohup autosubmit run a6zi &
nohup autosubmit run a6zs &
```

Code 3.9: The *autosubmit run* commands to start the workflow execution, for both *a6bi* and *a6bs* experiments.

Experiments finalization

After several 168,170.45 CPU hours of simulation, both experiments finished successfully. Since clean-related jobs were removed from the workflow, we need to delete all experiment-related files and directories on the remote platform, as its file system is intended only for temporary data storage. Prior to this, we must gather all the logs and CSV files generated by our header script containing the shared metrics. Both experiments occupied more than 8 TB:

```
[user1]@login0:-> du -hs /gpfs/scratch/[proj]/[user1]/a6zi/*
4,0T        /gpfs/scratch/[proj]/[user]/a6zi/18500101
...
[user2]@login0:-> du -hs /gpfs/scratch/[proj]/[user2]/a6zs/*
4,0T        /gpfs/scratch/[proj]/[user]/a6zs/18500101
...
```

Code 3.10: Size of the experiment output for both *a6bi* and *a6bs* in MareNostrum 4.

Exploring different wrapping configurations

Although the results obtained from the previous simulations are sufficient to draw our conclusions, given that we have enough margin to conduct additional experiments, it would be beneficial to conduct a new simulation with a different wrapping configuration, for example, by varying the size from 10 to 5 jobs, a reduction that should allow us to appreciate upward variations in queuing times because the jobs sent are smaller. This involves replicating some of the steps previously performed, just altering the reference experiments to our own and modifying the wrapping configuration in the *autosubmit.conf* file of the experiment to be wrapped as follows:

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 2
MAX_WRAPPED = 5
POLICY = flexible
```

Code 3.11: Another possible wrapping configuration for the wrapped experiment in MareNostrum 4.

With the updated configuration, the experiment without wrappers will be a duplicate of *a6zs*, while the wrapped one will be identical to *a6zi* except for the modification we made to the parameter *MAX_WRAPPED*, which represents the maximum size of the wrapper. However, after running the experiments, a general disk usage issue emerged over the weekend of April 21. The *hard* quota limit for our department's partition was surpassed, causing numerous experiments to fail, including ours. The disk usage is illustrated in the graph below, which shows the historical occupancy data, clearly highlighting the source of our errors.

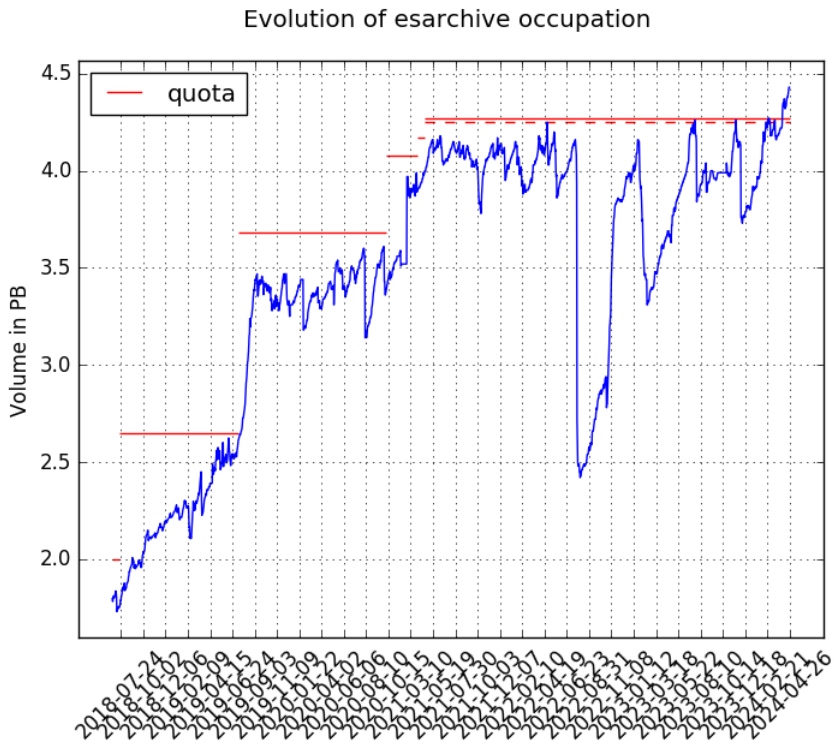

Figure 3.5: *esarchive* partition occupation history, until the MareNostrum 4 power-off. Graph from [47].

After this, and considering the imminent shutdown date of MareNostrum 4 on April 26th, we opted to discontinue these additional experiments since there wouldn't be sufficient time to conduct them.

3.5.2 MeluXina

Platform overview

MeluXina is Luxembourg's LuxProvide supercomputing project and a part of the EuroHPC JU supercomputers. It is built upon the EVIDEN BullSequana XH2000 architecture. This heterogeneous system is notable for its accelerated module's peak performance of 10.52 PetaFLOPS, featuring 200 nodes each equipped with 2 AMD Rome CPUs with 32 cores and 4 NVIDIA A100-40 GPUs. The cluster module includes 573 nodes with 2 AMD Rome CPUs, each with 64 cores. Additional modules comprise a secondary accelerator module with 20 FPGA nodes, a cloud module with 20 cloud VM host nodes, a storage module, and a Large Memory module designed for memory-intensive workloads, similar to typical CPU nodes. [48]

Upon its release in June 2021, its performance secured the accelerated module the 36th position on the TOP500 list[2].

Through a brief examination of previous experiments executed on this platform by other users, we noted that tasks dispatched to the scheduler are executed shortly after being queued. Feedback from colleagues also indicates that queues in MeluXina are generally not very full. Therefore, although we anticipate a decrease in queue times after applying our task aggregation method, we do not expect significant changes.

[2] https://top500.org/lists/top500/2021/06/. Last accessed: July 3, 2024

Available resources for our project

In the case of MeluXina, our two experiments will have to fit within a budget of a total of 200,000 CPU hours. This is a strict limit, so we must plan as precisely as possible how we want to use the machine.

Experiment overview

In the same way as in MareNostrum 4, our experimentation will be based on creating two Auto-EC-Earth3 experiments that will be launched to the remote platform at the same time. In the case of MeluXina, the *fairshare* value is never reset, so we must be careful not to launch any task prior to the execution of our workflows so that they start in as similar conditions as possible.

This time, the experiments will be based on a simplified MeluXina-specific test case called T255L91-ORCA1L75-LIM3. Again, T255L91 and ORCA1L75 refer to the grid resolution of the IFS and NEMO models, and LIM3 will run as part of the ocean model. The *expid* of this reference experiment is *t0d8*.

Considering that we have a budget of 100,000 CPU hours per experiment, we must calculate how many chunks we can run on this platform. This test case consists of a single startdate, a single member, and two chunks simulating 1 month. Each chunk takes an average of 7 minutes and 1 second to execute, i.e. 421 seconds. We want to simulate 12-month chunks, so that the size of the jobs submitted to the platform is large enough to appreciate the benefits of task aggregation on queue times.

Firstly, it is necessary to calculate the CPU hours required for each simulated chunk. The formula below will be used for this purpose. The number of processors, which is set to 1280, is derived from the test case parameters. Then,

$$Core\ Hours\ Per\ Chunk = Processors \cdot \overline{T_{execution}} \cdot \frac{ChunkSize_{New}}{ChunkSize_{Old}}$$

$$Core\ Hours\ Per\ Chunk = 1280 \cdot 421 \cdot \frac{1\ hour}{3600\ seconds} \cdot \frac{12}{1}$$

$$Core\ Hours\ Per\ Chunk = 1,796.27$$

This means that in the best-case scenario, without counting possible failures in the execution of the jobs, we could execute up to 55 chunks within our budget:

$$Max\ Runnable\ Chunks = \left\lfloor \frac{100,000}{1,796.27} \right\rfloor = 55$$

Ideally, we aim to divide the 50 chunks into groups of 10 jobs each, maintaining consistency between different platform experiments. Opting for this setup results in a minimal margin for job retrials. Considering the success rate in MareNostrum 4, this configuration is viable, but extreme caution and regular monitoring are required in order to halt both processes if an error occurs and resume their execution when the problem is solved.

In summary, we will execute two Auto-EC-Earth3 workflows utilizing a T255L91-ORCA1L75-LIM3 setup. Each workflow will have one startdate and one member simulating 50 chunks of 12 months each. The simulation tasks will be organized into wrappers of

10 tasks, with a minimum of 5, following the *flexible* Autosubmit policy. It is anticipated that the combined cost of the two experiments will remain near the budget of our project.

Configuring and launching experiments

We will once again employ the Autosubmit virtual machine to create and execute our experiments. However, this time we need to load the module for Autosubmit version 3.15.14 along with the appropriate Git version.

After preparing the environment, we can proceed to create the experiments. For simplicity, we will first set up the experiment using wrappers, and once configured, we will create the experiment without wrappers to avoid having to apply the configuration twice. The command for creating the experiment with wrappers is provided below:

```
autosubmit expid -y t0d8 -H meluxina -d "My Auto-EC-Earth3 running with
↪  wrappers on MeluXina."
```

Code 3.12: The *autosubmit expid* command to create the wrapped experiment based on *t0d8* to be executed on MeluXina.

Autosubmit has returned the *expid a72x* for our experiment. We can now start modifying the configuration files:

Configuration file	Modifications
autosubmit_a72x.conf	Defined the wrapper policy indicated in Code 3.13 for the *SIM* jobs. For this experiment, the policy has been established as *flexible*, with a wrapper size ranging from 5 to 10 jobs. The number of retrials has been reduced to 2, and email notifications have also been activated.
expdef_a72x.conf	As listed on Code 3.14, a unique startdate, *19900201*, a member, *fc00*, and 50 chunks of 12 months each have been defined.
jobs_a72x.conf	All the overhead tasks had been removed, and its dependencies undone as we did for MareNostrum 4, leaving only the jobs in Figure 3.2, i.e., the initialization and the *SIM* tasks. The location of the extended header has also been indicated and will be placed in the same directory that was specified for MareNostrum 4.
platforms_a72x.conf	The platforms were already set up correctly, so we only needed to update the usernames and remove the platforms we will not use.
proj_a72x.conf	Disabled several flags associated with the jobs previously deleted.

Table 3.3: Changes applied to the configuration files of the *a72x* (wrapped) experiment in MeluXina.

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 5
MAX_WRAPPED = 10
POLICY = flexible
```

Code 3.13: The wrapping configuration for the *a72x* experiment in MeluXina.

```
[experiment]
DATELIST = 19900201
MEMBERS = fc00
CHUNKSIZEUNIT = month
CHUNKSIZE = 12
NUMCHUNKS = 50
CALENDAR = standard
```

Code 3.14: The experiment configuration for *a72x* in MeluXina.

With the experiment with wrappers properly configured, we can execute the following Autosubmit command to generate the experiment without wrappers.

```
autosubmit expid -y a72x -H meluxina -d "My Auto-EC-Earth3 running without
↪  wrappers on MeluXina."
```

Code 3.15: The *autosubmit expid* command to create the unwrapped experiment based on *a72x* to be executed on MeluXina.

Autosubmit returned the *expid a74v* for this experiment. The only change needed in its configuration files is to remove the Code segment 3.13 from *autosubmit_a74v.conf* to disable the wrappers.

Once the modification is done, we can create the experiment with *autosubmit create*, as indicated in Code 3.16.

```
autosubmit create -cw a72x
autosubmit create a74v
```

Code 3.16: The *autosubmit create* commands to create the *a72x* and *a74v* experiments.

Similarly to the MareNostrum 4 experiments, it is necessary to copy the extended header script to the directory specified in the configuration. The script is the same as the one shown in Appendix A, considering that an account may be linked to multiple accounts. The *sshare* command construction will require indicating the account ID.

After making sure that the virtual machine can access MeluXina via SSH without a password, we can launch the experiments:

```
nohup autosubmit run a72x &
nohup autosubmit run a74v &
```

Code 3.17: The *autosubmit run* commands to start the workflow execution, for both *a72x* and *a74v* experiments.

Experiments finalization

Experiments on MeluXina required less execution time, and when both experiments were completed, it was necessary to clean up the experiments-related files on the remote file system, which occupied more than 6 TB in total.

```
[[user1]@login03 -]$ du -hs /project/scratch/[proj]/[user1]/a72x/*
3.1T        /project/scratch/[proj]/[user1]/a72x/19900201
...
[[user2]@login03 -]$ du -hs /project/scratch/[proj]/[user2]/a74v/*
3.1T        /project/scratch/[proj]/[user2]/a74v/19900201
...
```

Code 3.18: Size of the experiment output for both *a72x* and *a74v* in MeluXina.

In total, the combined CPU hours used for both experiments amounted to 150,221.02.

3.5.3 ECMWF HPC2020

Platform overview

HPC2020 is a cluster of the European Centre for Medium-Range Weather Forecasts located in a data center in Bologna, Italy. It is a general purpose cluster with a compute partition with 7,488 nodes with AMD Epyc Rome processors, totaling 128 cores per node, organized in Bull Sequana XH2000 high-density racks, and another "General Purpose and Interactive Login" partition with 208 nodes running at slightly higher frequency and with more memory per node than normal compute nodes. [49]

Setting up this platform involved configuring TOTP (Time-based One-Time Password) for our ECMWF user as a two-factor authentication mechanism. The platform's login nodes can be accessed through the Teleport software, which provides an SSH Jump Host and single sign-on, although it requires re-authentication every 12 hours. Teleport will enable us to access HPC2020 through an ECMWF gateway in Bologna, *jump.ecmwf.int*.

Furthermore, even though this platform also employs Slurm as its workload manager, tasks need to be submitted through a proprietary service called *ECaccess*, which relies on a certificate that is valid for one week and can be generated by the user with TOTP.

Just as we did in the case of MeluXina, we reviewed old experiments that other users ran on HPC2020, and the queue times, again, are relatively low, so it is expected that the queues are not very full and that the use of wrappers will not have much impact on queue times.

Available resources for our project

Due to the high demand for this platform, we had to delay the experiments several times. Eventually, we received a total budget of 1 million SBUs, or System Billing Unitss. One CPU hour equals 17.06 SBU.

Then, we can use 500,000 SBUs for each experiment, that is, 29,308.32 CPU hours.

Experiment overview

Again, the goal is to run two experiments with and without wrappers synchronously. As with MeluXina, since the *fairshare* is not reset periodically, we will have to be careful not to send tasks to the machine prior to execution.

Taking into account that our budget for this platform is much lower than for the others, we will take as a reference a simplified test case like the MeluXina one, but specific for this

platform. It will be the same Auto-EC-Earth3 configuration, the T255L91-ORCA1L75-LIM3, with *expid t0d9*.

To calculate the number of 12-month chunks we can run, we just need to know the average chunk execution time, 1980 seconds, the chunk size, which would be 1 month, and the number of processors we will allocate for our *SIM* jobs. In this case, 768 processors. Substituting these values in the formula defined in Section (3.5.2), we have:

$$Core\ Hours\ Per\ Chunk = 768 \cdot 1980 \cdot \frac{1\ hour}{3600\ seconds} \cdot \frac{12}{1}$$

$$Core\ Hours\ Per\ Chunk = 1,267.20$$

Then, in the best case, if we do not have any failures, we will be able to run up to 23 chunks.

$$Max\ Runnable\ Chunks = \left\lfloor \frac{29,308.32}{1,267.20} \right\rfloor = 23$$

The final decision was to run 25 12-month chunks for each experiment, which would allow us to have 5 wrappers of 5 *SIM* jobs without exceeding the budget too much. Of course, it is imperative to properly monitor the experiment to stop it in case of failure, correct it, and resume execution.

So, in summary, we will run two experiments based on a simplified Auto-EC-Earth3 configuration, the T255L91-ORCA1L75-LIM3. They will have a single startdate, member and 25 chunks of 12 months of simulation. Simulation tasks will be grouped in 5 wrappers of 5 jobs each. The number of retrials will be kept at 2, as in MeluXina.

Configuring and launching experiments

In this case, we will not launch the experiments from the Autosubmit virtual machine, as it does not have the software required to submit jobs to HPC2020. Instead, we will use a *hub* machine of the department. The reference test case operates on Autosubmit version 3.15.14, which requires loading the corresponding module, along with the appropriate Git version and the *ECaccess* module. This process includes authenticating through the module to generate a connection certificate, valid for one week. Now, we can proceed to create the experiments using the same approach as in MeluXina: we will first generate the experiment with wrappers and then clone it. The command to create the experiment with wrappers is as follows:

```
autosubmit expid -y t0d9 -H ecmwf-hpc2020 -d "My Auto-EC-Earth3 running
↪  with wrappers on HPC2020."
```

Code 3.19: The *autosubmit expid* command to create the wrapped experiment based on *t0d9* to be executed on ECMWF HPC2020.

This command returned a new *expid* for our experiment, *a76w*, and created its directory structure. With this, we can move on to the configuration:

Configuration file	Modifications
autosubmit_a76w.conf	The wrapping policy is the one illustrated in Code 3.20. It is a *strict* policy, with wrappers on the *SIM* jobs with a size of 5 jobs. A *strict* policy has been established because, considering the scale of the experiment, we need to ensure that Autosubmit does not dispatch wrappers smaller than the desired size so that we can appreciate the improvements. Retrials have been limited to 2 attempts, and email alerts have also been enabled.
expdef_a76w.conf	A startdate, *18500101*, a member, *fc00*, and 25 chunks of 12 months have been defined. The specific configuration is represented in the Code 3.21.
jobs_a76w.conf	All unnecessary tasks and their dependencies have been eliminated, leaving only the workflow initialization and simulation tasks. Additionally, the extended header has been set up for the *SIM* jobs.
platforms_a76w.conf	Since the platforms were already configured, it was only necessary to change the user names. It should be noted that in the case of the *ecmwf-hpc2020* platform, it was necessary to explicitly indicate that jobs must be submitted with the *ECaccess* software. We can see a reduced version of this configuration in Code 3.22.
proj_a76w.conf	Disabled several flags associated with the jobs previously deleted.

Table 3.4: Changes applied to the configuration files of the *a76w* (wrapped) experiment in ECMWF HPC2020.

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 5
MAX_WRAPPED = 5
POLICY = strict
```

Code 3.20: The wrapping configuration for the *a76w* experiment in ECMWF HPC2020.

```
[experiment]
DATELIST = 18500101
MEMBERS = fc00
CHUNKSIZEUNIT = month
CHUNKSIZE = 12
NUMCHUNKS = 25
CALENDAR = standard
```

Code 3.21: The experiment configuration for *a76w* in ECMWF HPC2020.

```
[ecmwf-hpc2020]
TYPE = ecaccess
VERSION = slurm
...
```

Code 3.22: Reduced *ecmwf-hpc2020* platform configuration for the *a76w* experiment in ECMWF HPC2020.

After setting up the experiment with wrappers, we can create the reference experiment using the Autosubmit command below:

```
autosubmit expid -y a76w -H ecmwf-hpc2020 -d "My Auto-EC-Earth3 running
↪  without wrappers on HPC2020."
```

Code 3.23: The *autosubmit expid* command to create the wrapped experiment based on *a76w* to be executed on ECMWF HPC2020.

The *expid* for this new experiment is *a773*. Before running it, we need to delete the Code 3.20 section from the file *autosubmit_a773.conf*. After that, both experiments can be launched.

```
autosubmit create -cw a76w
autosubmit create a773
```

Code 3.24: The *autosubmit create* commands to create the *a72x* and *a74v* experiments.

To track the platform's status during execution, the header script will be copied to the directory specified in the configuration. This script is a variation of the one in A, which runs the same Slurm command, but handles the output differently, because this platform uses a different version of Slurm.

After ensuring that the ECaccess certificate is valid, we can proceed to run the experiments.

```
nohup autosubmit run a76w &
nohup autosubmit run a773 &
```

Code 3.25: The *autosubmit run* commands to start the workflow execution, for both *a72x* and *a74v* experiments.

Experiments finalization

Even though the experiments were properly configured, an unidentified error in Autosubmit prevented the creation of wrappers, resulting in the tasks being submitted individually to the platform. This bug cannot be resolved on time, thus we have opted to suspend these experiments and defer their execution for the future.

3.5.4 MareNostrum 5

Platform overview

MareNostrum 5 is currently one of the 3 pre-exascale supercomputers of the EuroHPC JU [3], and the latest version of a family of state-of-the-art supercomputers that began in 2004. In April 2024, it officially replaced MareNostrum 4, which is also the subject of study in this investigation (see 3.5.1). It has reaffirmed its status as the leading supercomputer in Spain and has even been ranked among the 3 most powerful supercomputers in Europe. Worldwide, it entered the TOP500 for the first time in the list of November 2023[3], and its accelerated partition (ACC) secured the 8th position and continues to hold it, while its general-purpose partition (GPP) achieved the 19th position. With its heterogeneous architecture, the applications inherited from its predecessor can be expanded to include fields such as medicine, the creation of digital twins of the Earth and the human body, artificial intelligence, or deep learning.

[3] https://top500.org/lists/top500/list/2023/11/. Last accessed: July 2, 2024

MareNostrum 5 consists of multiple partitions, with the main one being the general-purpose partition (MareNostrum 5 GPP), featuring 6408 standard nodes each equipped with two Intel Sapphire Rapids processors with 56 cores each, which translates into 112 cores per node. Furthermore, there are 72 High Bandwidth Memory (HBM) nodes, each also with 112 cores per node. This partition achieves a peak performance of 45.9 PetaFLOPS. The accelerated partition (MareNostrum 5 ACC) comprises 1120 nodes, each containing 2 Intel Sapphire Rapids processors and 4 Nvidia Hopper GPUs, achieving a peak performance of 260 PetaFLOPS. There are other smaller compute partitions destinated to pre- or post-process, or to Next Generation Technologies: the NGT ACC and the NGT GPP partitions, this last one made up of 408 Nvidia Grace ARM processors.

Available resources for our project

As in the MareNostrum 4 supercomputer, which was also managed by the BSC, we will utilize the computational budget of the Earth Sciences department to execute the simulations on MareNostrum 5.

Experiment overview

Experimentation on this platform will also consist of running two experiments simultaneously. One of the administrative distinctions with respect to MareNostrum 4 is that the *fairshare* value is never reset. Consequently, similar to previous platforms, we need to be especially cautious to avoid running any job in either of the two accounts that will run the experiments.

In this case, instead of using the regular test cases, the experiments will utilize a specific test case for MareNostrum 5, which is still being set up by one of the Auto-EC-Earth3 auto-modelers. This is because the porting of the auto-model to this platform is still ongoing. The experiment is the *t0n4* and operates with a T255L91-ORCA1L75-LIM3 configuration.

Using this test case as a base experiment, we will conduct simulations consisting of 40 chunks, each simulating 12 months, organized into 4 wrappers of 10 jobs. The retrial count can once again be adjusted to 6.

Configuring and launching experiments

Following the methodology of the previous platforms, we will load the corresponding Autosubmit version in the virtual machine, in this case, version 3.15.18, alongside the Git module with the correct version to prevent problems during download, like those experienced with MareNostrum 4. We will first create the experiment with wrappers and then replicate it to use it as a reference experiment.

```
autosubmit expid -y t0n4 -H marenostrum5 -d "My Auto-EC-Earth3 running
↪  with wrappers on MareNostrum 5."
```

Code 3.26: The *autosubmit expid* command to create the wrapped experiment based on *t0n4* to be executed on MareNostrum 5.

autosubmit expid returned the*expid a7d4* for our experiment. Once created the experiment, we can proceed to the configuration:

Configuration file	Modifications
autosubmit_a7d4.conf	Added the wrapping policy in Code 3.27. It is *flexible*, with wrappers of 5 to 10 *SIM* jobs. The count of retrials have been adjusted to 6, and email notifications have been also set up.
expdef_a7d4.conf	Defined a startdate, *19900201*, a member, *fc00*, and 40 chunks of 12 months have been defined, as the Code 3.28 indicates.
jobs_a7d4.conf	Deleted both extra jobs and their dependencies. The extended header has also been configured for the *SIM* jobs.
platforms_a7d4.conf	The platforms were already configured, only the usernames needed to be modified.
proj_a7d4.conf	Disabled several flags associated with the jobs previously deleted.

Table 3.5: Changes applied to the configuration files of the *a7d4* (wrapped) experiment in MareNostrum 5.

```
[wrapper]
TYPE = vertical
JOBS_IN_WRAPPER = SIM
MIN_WRAPPED = 5
MAX_WRAPPED = 10
POLICY = flexible
```

Code 3.27: The wrapping configuration for the *a7d4* experiment in MareNostrum 5.

```
[experiment]
DATELIST = 19900201
MEMBERS = fc00
CHUNKSIZEUNIT = month
CHUNKSIZE = 12
NUMCHUNKS = 40
CALENDAR = standard
```

Code 3.28: The experiment configuration for *a7d4* in MareNostrum 5.

Once the experiment with wrappers has been set up, we can create the unwrapped experiment with the following Autosubmit command:

```
autosubmit expid -y a7d4 -H ecmwf-marenostrum5 -d "My Auto-EC-Earth3
↪  running without wrappers on MareNostrum 5."
```

Code 3.29: The *autosubmit expid* command to create the wrapped experiment based on *a7d4* to be executed on MareNostrum 5.

The new *expid* is *a7d5*. It is necessary to delete the wrapper-specific configuration from its *autosubmit_a7d5.conf* file. Once that is done, both experiments can be initiated.

```
autosubmit create -cw a7d4
autosubmit create a7d5
```

Code 3.30: The *autosubmit create* commands to create the *a7d4* and *a7d5* experiments.

To collect Slurm *share* metrics, the header script will be placed in the directory defined in the configuration. This script will be the same as the one used for MareNostrum 4, in which specifying the account was not necessary. At this point, we are ready to launch the experiments.

```
nohup autosubmit run a7d4 &
nohup autosubmit run a7d5 &
```

Code 3.31: The *autosubmit run* commands to start the workflow execution, for both *a7d4* and *a7d5* experiments.

Experiments finalization

After the successful completion of both simulations, it was necessary to remove the files related to our experiments from *scratch*, which needed nearly 5 TB of space.

```
[[user1]@glogin1 -]$ du -hs /gpfs/scratch/[proj]/[user1]/a7d4/*
2.6T        /gpfs/scratch/[proj]/[user1]/a7d4/19900201
...
[[user2]@glogin1 -]$ du -hs /gpfs/scratch/[proj]/[user2]/a7d5/*
2.6T        /gpfs/scratch/[proj]/[user2]/a7d5/19900201
...
```

Code 3.32: Size of the experiment output for both *a7d4* and *a7d5* in MareNostrum 5.

In total, both experiments required a total of 105,181.20 CPU hours to execute.

Chapter 4

Experimentation results

In this chapter, we will present and discuss the results obtained from our experiments with the Auto-EC-Earth3 workflow, comparing scenarios with and without wrappers across MareNostrum 4, MeluXina, and MareNostrum 5. No results are available for HPC2020 as we were unable to properly run the wrapped experiment on that platform.

4.1 MareNostrum 4

Platform status

The header script executed before each simulation task (A.1) has given us insightful information about the scheduling factors of the user. The figure 4.1, derived from the data in Tables B.1 and B.2 found in Appendix B, represents the progression of the *fairshare* value throughout the execution of the workflow on one side, and on the other side, the Raw Usage, or the utilization in *core seconds*, of our tasks for both users.

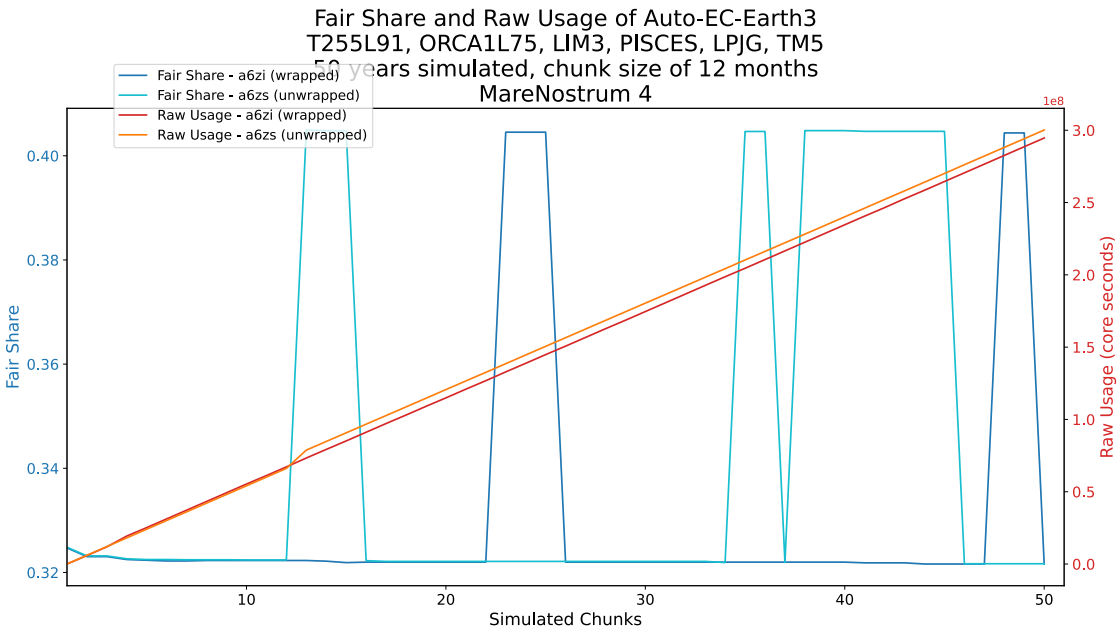

Figure 4.1: Fair Share and Raw Usage of Auto-EC-Earth3 T255L91-ORCA1L75-LIM3-PISCES-LPJG-TM5. Simulating 50 chunks of 12 months each. Results for MareNostrum 4.

The graph shows nearly identical *fairshare* and *raw usage* values for the two accounts

involved in the experiments, starting at an elevated value. Although there is a usage of approximately 3×10^8 *core seconds*, this utilization does not significantly impact the *fairshare* value, which does not decrease as much as expected. This indicates that our usage is considerably lower relative to the other members of our group and the rest of the machine.

Several peaks in the evolution of *fairshare* have been identified. Each peak, considering the significant increase in the value, indicates that some other group used more resources than they should with respect to ours.

Queuing times

Data from the Autosubmit database reveal that the total queuing time for the experiment using wrappers is 15577 seconds (4.33 hours), while the reference experiment, which does not use wrappers, has a queuing time of 173973 seconds (2.01 days). To evaluate the actual improvement gained in the queueing times of the experiment with wrappers compared to the reference experiment, represented in Figure 4.2, we can determine the SpeedUp:

$$SpeedUp = \frac{T_{unwrapped}}{T_{wrapped}} = \frac{173973}{15577} = 11.17$$

The time required to queue all tasks in the experiment without wrappers is 1116,86% greater than the total queueing time for the experiment where we used the task aggregation technique. Figure 4.2 clearly represents this improvement.

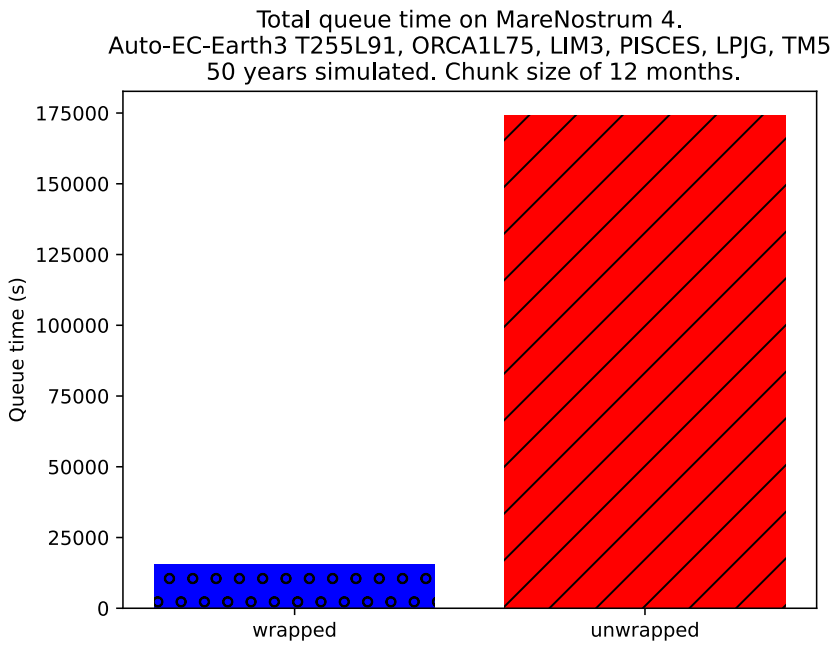

Figure 4.2: Total queue times for the *a6zi* (wrapped) and *a6zs* (unwrapped) experiments in MareNostrum 4. The Auto-EC-Earth3 configuration was T255L91-ORCA1L75-LIM3-PISCES-LPJG-TM5, simulating 50 chunks of 12 months each.

4.2 MeluXina

In the case of MeluXina, in Figure 4.3, we can observe how the values of *fairshare* and *raw usage* follow roughly the same pattern for both the user that runs with the wrappers and the one that runs the reference experiment.

Both experiments begin with nearly the same share values for each user. The initial value of *fairshare* is quite high, which means that our group in the machine does not use it as much as in MareNostrum 4.

In this case, machine usage, which accumulates over 5×10^8 *core seconds* per experiment at the end, influences the value that the *fair share* acquires during the runtime. The decrease in the *fair share* is an indicator that we are making an intensive use of the machine compared to the rest of the users of our group, leading Slurm to penalize us to allow them to execute their jobs.

Platform status

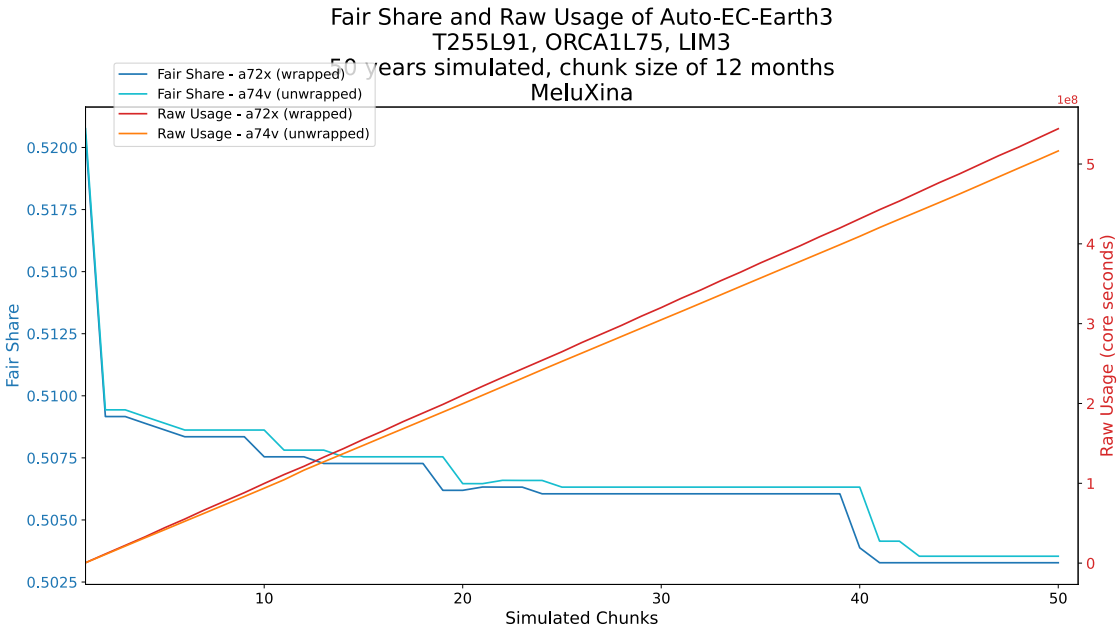

Figure 4.3: Fair Share and Raw Usage of Auto-EC-Earth3 T255L91-ORCA1L75-LIM3. Simulating 50 chunks of 12 months each. Results for MeluXina.

Queuing times

The aggregated queue times for the experiments with and without wrappers executed in MeluXina are 81 and 971 seconds, respectively. Calculating the SpeedUp will allow us to quantify the benefit of implementing task aggregation on this workflow:

$$SpeedUp = \frac{T_{unwrapped}}{T_{wrapped}} = \frac{971}{81} = 11.99$$

We can see this gain in Figure 4.4.

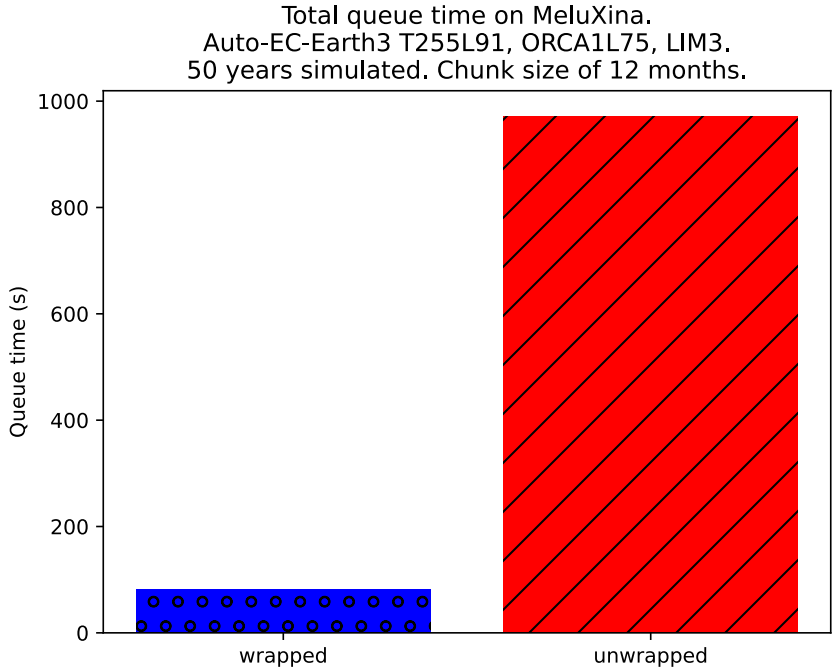

Figure 4.4: Total queue times for the *a72x* (wrapped) and *a74v* (unwrapped) experiments in MeluXina. The Auto-EC-Earth3 configuration was T255L91-ORCA1L75-LIM3, simulating 50 chunks of 12 months each.

4.3 MareNostrum 5

Platform status

Figure 4.5 represents the share values before starting the simulation jobs in the Auto-EC-Earth3 workflows executed on MareNostrum 5.

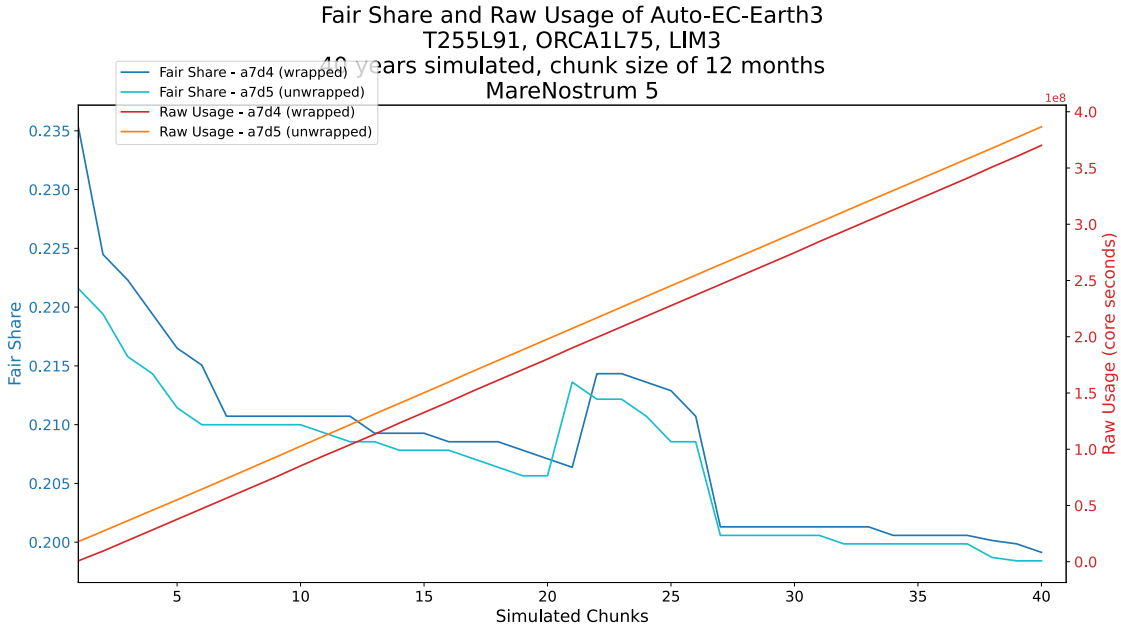

Figure 4.5: Fair Share and Raw Usage of Auto-EC-Earth3 T255L91-ORCA1L75-LIM3. Simulating 40 chunks of 12 months each. Results for MareNostrum 5.

The first thing to note about these results is that neither the *fairshare* nor the *raw usage* start with similar values. The user who ran the workflow without wrappers exhibits greater *raw usage* and a reduced *fairshare*, indicating prior utilization of the machine. Despite this, we can see that both users' utilization tend to follow the same pattern.

The initial *fairshare* value is very low compared to what we have observed on the other platforms, indicating that our group is using less of the machine than some other group. Moreover, the decreasing trend in *fairshare* as we utilize the machine means that our workload is higher than that of other users.

For a period, Slurm boosted our *fairshare*, likely due to the starting of resource-intensive jobs in other groups. However, after several hours, the machine load increased beyond the level it was at before the boost and *fairshare* starts to decline following the original trajectory.

Queuing times

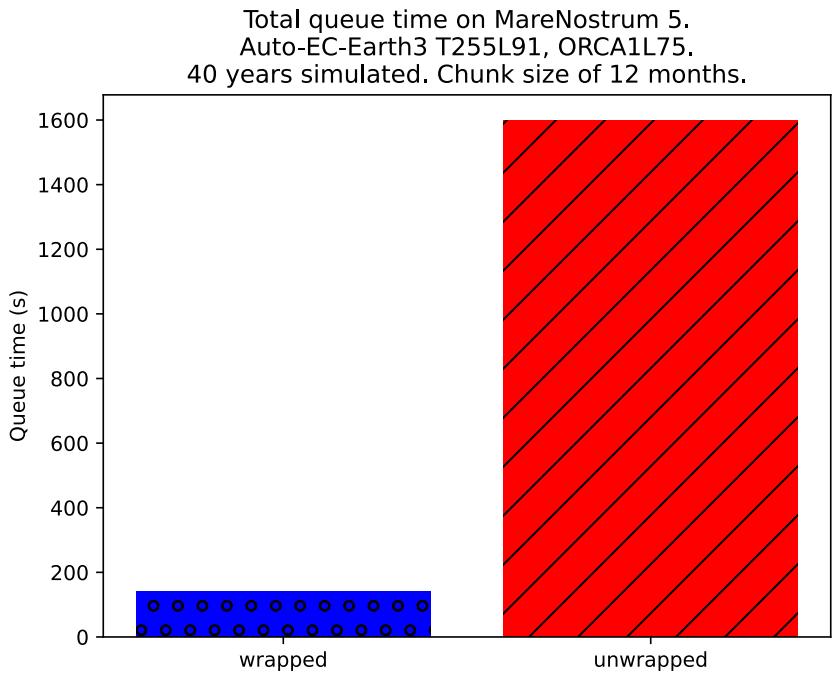

Figure 4.6: Total queue times for the *a7d4* (wrapped) and *a7d5* (unwrapped) experiments in MareNostrum 5. The Auto-EC-Earth3 configuration was T255L91-ORCA1L75, simulating 40 chunks of 12 months each.

Figure 4.6 illustrates a notable reduction in queue times for the wrapped experiment compared to the reference. The aggregated queue times are 141 and 1598 seconds for each respective experiment. The SpeedUp, which is 11.33, further demonstrates the substantial enhancement in queuing times achieved through wrapping.

$$SpeedUp = \frac{T_{unwrapped}}{T_{wrapped}} = \frac{1598}{141} = 11.33$$

Chapter 5

Conclusions and future work

Conclusions

In this study, we analyzed how task aggregation affects queueing times using workflows based on Auto-EC-Earth3. The range of platforms and model configurations, particularly of EC-Earth3, permitted the submission of jobs of different sizes to platforms with very different levels of utilization, allowing us to evaluate different scenarios separately.

Our experimental results show that *the total queue times for an experiment utilizing vertical wrappers are 11 to 12 times shorter compared to an experiment without them.* This finding is consistent across all three platforms that we tested, which had very different resource availability, as evidenced by the graphs in Chapter 4 derived from the *share* metrics in Appendix B.

It has also been observed that machine occupancy affects the overall queue time proportionally for both wrapped and unwrapped workflows. Therefore, users might not notice significant improvements, as seen with MeluXina and MareNostrum 5, where even a substantial relative gain amounts to less than half an hour for workflows exceeding 2 days in duration. Conversely, in the case of MareNostrum 4, the same gain results in nearly 2 days saved, which users will appreciate.

Therefore, using vertical wrappers will always result in a major improvement that can lead to a varying degree of reduction in queue time, and consequently, the overall workflow execution time, based on the machine's resource availability and workload.

Future work

After assessing the impact of task aggregation techniques on a range of top-tier supercomputers across Europe, we have sufficient information to start the development of an automated decision-making policy within the Autosubmit workflow manager.

This policy should evaluate, prior to generating the jobs for submission to the remote Slurm scheduler, whether the kind of wrapper selected and the state of the user scheduling factors for a specific user and queue allow accelerating the workflow execution, as using vertical wrappers reduces queue times and, consequently, the overall execution time.

Once implemented in production, it is anticipated to significantly boost the production efficiency for the Department or any research group opting to use the new functionality. It should be noted that Autosubmit is employed beyond just the Earth Sciences Department at the BSC, meaning that any enhancements at its scheduling level will benefit the experiments conducted by all its users, even from other institutions.

Bibliography

[1] Tirthak Patel et al. "Job Characteristics on Large-Scale Systems: Long-Term Analysis, Quantification, and Implications". In: *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. 2020, pp. 1–17. DOI: 10.1109/SC41405.2020.00088.

[2] "TOP500". URL: https://top500.org/. *Last accessed: June 30, 2024.*

[3] "Our supercomputers - European Union". URL: https://eurohpc-ju.europa.eu/supercomputers/our-supercomputers. *Last accessed: July 2, 2024.*

[4] Jörn Hoffmann et al. "Destination Earth – A digital twin in support of climate services". In: *Climate Services* 30 (2023), p. 100394. ISSN: 2405-8807. DOI: 10.1016/j.cliser.2023.100394.

[5] Chenyu Tang et al. "A roadmap for the development of human body digital twins". In: *Nature Reviews Electrical Engineering* 1.3 (2024), pp. 199–207. ISSN: 2948-1201. DOI: 10.1038/s44287-024-00025-w. URL: https://doi.org/10.1038/s44287-024-00025-w.

[6] Sergi Palomas Martinez. "Automatic load-balance method for coupled Earth System Models". Master thesis. UPC, Facultat d'Informàtica de Barcelona, Departament d'Arquitectura de Computadors, June 2022. URL: http://hdl.handle.net/2117/376508.

[7] Adrián Herrera et al. "A Simulator for Intelligent Workload Managers in Heterogeneous Clusters". In: *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 2021, pp. 196–205. DOI: 10.1109/CCGrid51090.2021.00029.

[8] Esteban Stafford and José Luis Bosque. "Performance and energy task migration model for heterogeneous clusters". In: *The Journal of Supercomputing* 77.9 (2021), pp. 10053–10064. ISSN: 1573-0484. DOI: 10.1007/s11227-021-03663-1. URL: https://doi.org/10.1007/s11227-021-03663-1.

[9] Domingo Manubens-Gil et al. "Seamless management of ensemble climate prediction experiments on HPC platforms". In: *2016 International Conference on High Performance Computing & Simulation (HPCS)*. 2016, pp. 895–900. DOI: 10.1109/HPCSim.2016.7568429.

[10] Manuel Giménez de Castro Marciani. "Evaluating the impact of task aggregation in workflows with shared resources environments". Master thesis. UPC, Facultat d'Informàtica de Barcelona, Departament d'Arquitectura de Computadors, Oct. 2023. URL: http://hdl.handle.net/2117/404041.

[11] Ralf Döscher et al. "The EC-Earth3 Earth system model for the Coupled Model Intercomparison Project 6". In: *Geoscientific Model Development* 15.7 (Apr. 2022), pp. 2973–3020. ISSN: 19919603. DOI: 10.5194/gmd-15-2973-2022.

[12] V Eyring et al. "Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization". In: *Geoscientific Model Development* 9.5 (2016), pp. 1937–1958. DOI: 10.5194/gmd-9-1937-2016.

[13] *Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, H. Lee and J. Romero (eds.)]* Tech. rep. Geneva, Switzerland: IPCC, 2023. DOI: 10.59327/IPCC/AR6-9789291691647.

[14] Andrew Gettelman and Richard Rood. *Demystifying Climate Models*. Vol. 2. June 2016. ISBN: 978-3-662-48959-8. DOI: 10.1007/978-3-662-48959-8.

[15] M Klose et al. "Mineral dust cycle in the Multiscale Online Nonhydrostatic AtmospheRe CHemistry model (MONARCH) Version 2.0". In: *Geoscientific Model Development* 14.10 (2021), pp. 6403–6444. DOI: 10.5194/gmd-14-6403-2021.

[16] M Guevara et al. "HERMESv3, a stand-alone multi-scale atmospheric emission modelling framework – Part 2: The bottom-up module". In: *Geoscientific Model Development* 13.3 (2020), pp. 873–903. DOI: 10.5194/gmd-13-873-2020.

[17] J Benavides et al. "CALIOPE-Urban v1.0: coupling R-LINE with a mesoscale air quality modelling system for urban air quality forecasts over Barcelona city (Spain)". In: *Geoscientific Model Development* 12.7 (2019), pp. 2811–2835. DOI: 10.5194/gmd-12-2811-2019.

[18] Roberto Buizza et al. *The development and evaluation process followed at ECMWF to upgrade the Integrated Forecasting System (IFS)*. European Centre for Medium Range Weather Forecasts, 2018. DOI: 10.21957/xzopnhty9.

[19] Gurvan Madec and the NEMO System Team. "NEMO ocean engine". In: *Scientific Notes of IPSL Climate Modelling Center, 27. Zenodo* (2022). ISSN: 1288-1619. DOI: 10.5281/zenodo.6334656.

[20] C. Rousset et al. "The Louvain-La-Neuve sea ice model LIM3.6: global and regional capabilities". In: *Geoscientific Model Development* 8.10 (2015), pp. 2991–3005. DOI: 10.5194/gmd-8-2991-2015.

[21] O. Aumont et al. "PISCES-v2: an ocean biogeochemical model for carbon and ecosystem studies". In: *Geoscientific Model Development* 8.8 (2015), pp. 2465–2513. DOI: 10.5194/gmd-8-2465-2015.

[22] B. Smith et al. "Implications of incorporating N cycling and N limitations on primary production in an individual-based dynamic vegetation model". In: *Biogeosciences* 11.7 (2014), pp. 2027–2054. DOI: 10.5194/bg-11-2027-2014.

[23] M. Lindeskog et al. "Implications of accounting for land use in simulations of ecosystem carbon cycling in Africa". In: *Earth System Dynamics* 4.2 (2013), pp. 385–407. DOI: 10.5194/esd-4-385-2013.

[24] T. P. C. van Noije et al. "Simulation of tropospheric chemistry and aerosols with the climate model EC-Earth". In: *Geoscientific Model Development* 7.5 (2014), pp. 2435–2475. DOI: 10.5194/gmd-7-2435-2014.

[25] R. Winkelmann et al. "The Potsdam Parallel Ice Sheet Model (PISM-PIK) – Part 1: Model description". In: *The Cryosphere* 5.3 (2011), pp. 715–726. DOI: 10.5194/tc-5-715-2011.

[26] A. Craig, S. Valcke, and L. Coquart. "Development and performance of a new version of the OASIS coupler, OASIS3-MCT_3.0". In: *Geoscientific Model Development* 10.9 (2017), pp. 3297–3308. DOI: 10.5194/gmd-10-3297-2017.

[27] R. Haarsma et al. "HighResMIP versions of EC-Earth: EC-Earth3P and EC-Earth3P-HR – description, model computational performance and basic validation". In: *Geoscientific Model Development* 13.8 (2020), pp. 3507–3527. DOI: 10.5194/gmd-13-3507-2020.

[28] Hilary Oliver et al. "Workflow Automation for Cycling Systems". In: *Computing in Science & Engineering* 21.4 (2019), pp. 7–21. DOI: 10.1109/MCSE.2019.2906593.

[29] Avi Bahra. "Managing work flows with ecFlow". *ECMWF Newsletter*. DOI: 10.21957/nr843dob.

[30] Wilmer Uruchi, Miguel Castrillo, and Daniel Beltrán. "Autosubmit GUI: A Javascript-based Graphical User Interface to Monitor Experiments Workflow Execution". In: *Journal of Open Source Software* 6.59 (2021), p. 3049. DOI: 10.21105/joss.03049.

[31] Kevin P Gaffney et al. "SQLite: past, present, and future". In: *Proceedings of the VLDB Endowment* 15.12 (). DOI: 10.14778/3554821.3554842.

[32] Eric S Raymond. *The Art of UNIX Programming*. Pearson Education, 2003. ISBN: 978-0-13-142901-7.

[33] Morris A. Jette and Tim Wickberg. "Architecture of the Slurm Workload Manager". In: *Job Scheduling Strategies for Parallel Processing. JSSPP 2023. Lecture Notes in Computer Science, vol 14283*. Ed. by Klusáček Dalibor, Julita Corbalán, and Rodrigo Gonzalo P. Cham: Springer Nature Switzerland, 2023, pp. 3–23. DOI: 10.1007/978-3-031-43943-8.

[34] "Slurm Workload Manager - slurm.conf". URL: https://slurm.schedmd.com/slurm.conf.html. *Last accessed: June 30, 2024.*

[35] Njoud O Almaaitah et al. "Performance-driven scheduling for malleable workloads". In: *The Journal of Supercomputing* 80.8 (2024), pp. 11556–11584. ISSN: 1573-0484. DOI: 10.1007/s11227-023-05882-0.

[36] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating System Concepts*. 10th ed. Wiley, 2018. ISBN: 978-1-119-43925-7.

[37] Shawn Hoopes. "Priority and Fair Trees". In: *Priority and Fair Trees*. Sept. 2019. URL: https://slurm.schedmd.com/SLUG19/Priority_and_Fair_Trees.pdf.

[38] A W Mu'alem and D G Feitelson. "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling". In: *IEEE Transactions on Parallel and Distributed Systems* 12.6 (2001), pp. 529–543. DOI: 10.1109/71.932708.

[39] "Autosubmit Wrappers. Documentation". URL: https://autosubmit.readthedocs.io/en/latest/userguide/wrappers/. *Last accessed: June 19, 2024.*

[40] "Create an Experiment. Autosubmit Documentation". URL: https://autosubmit.readthedocs.io/en/latest/userguide/create/. *Last accessed: June 04, 2024.*

[41] "Auto-ECEarth3 Documentation. GitLab". URL: https://earth.bsc.es/gitlab/es/auto-ecearth3/-/wikis/home. *Private repository. Last accessed: June 03, 2024.*

[42] "OpenIFS ECMWF Project". URL: https://www.ecmwf.int/en/research/projects/openifs. *Last accessed: June 04, 2024.*

[43] "Auto-ECEarth3 Jobs. GitLab". URL: https://earth.bsc.es/gitlab/es/auto-ecearth3/-/wikis/auto-ecearth_jobs. *Private repository. Last accessed: June 11, 2024.*

[44] "Slurm Workload Manager - sshare". URL: https://slurm.schedmd.com/sshare.html. *Last accessed: July 2, 2024.*

[45] "MareNostrum 5 — BSC-CNS". URL: https://www.bsc.es/es/marenostrum/marenostrum-5. *Last accessed: July 2, 2024.*

[46] Pablo Goitia et al. "Profiler integration in a python-based workflow manager". URL: https://earth.bsc.es/wiki/lib/exe/fetch.php?media=degree_students:bsc_report.pdf. *Last accessed: July 2, 2024.*

[47] "Occupation growth of "esarchive" partition". URL: https://www.bsc.es/projects/earthscience/esarchive/plot.html. *Last accessed: June 03, 2024.*

[48] "System overview - MeluXina User Documentation". URL: https://docs.lxp.lu/system/overview/. *Last accessed: July 3, 2024.*

[49] "HPC2020 User Guide". URL: https://confluence.ecmwf.int/display/UDOC/HPC2020+User+Guide. *Last accessed: July 3, 2024.*

Appendices

Appendix A

Scripts

Gathering metrics on the share status of the platforms.

```bash
#!/bin/bash
#
# This script fetches the current usage, fairshare, and level fairshare.
# Recorded data is stored in a CSV file upon every submitted job.
#
# BSC-CNS - Earth Sciences, 2024

# Pick rawusage, which is in core seconds, the current fairshare, and
# the fairshare among users in our group.
# The -h flag is used to avoid having headers, -P to separate metrics
# with "|", and -U to pick the metrics associated with the current user.
# The sed command is used to replace '|' for ','.
SLURM_OUT=$(sshare --format=RawUsage,FairShare,LevelFS -hPU -A ******* |
↪  sed -r -e "s/[ \|\ ]+/,/g")

# Get job name
JOB_NAME=$(basename $job_name_ptrn)

# Record current time in UTC following ISO8601 format to prevent
# complications when used on global platforms.
DATE=$(date --utc -Iseconds)

# Compute the CSV path
EXPID=$(echo $JOB_NAME | cut -d'_' -f1)
OUTPUT_FILE="$(dirname $job_name_ptrn)/${EXPID}_SLURM_DATA.csv"

# If the output file does not exists, create it
if [ ! -f $OUTPUT_FILE ]; then
    echo "Job Name,Start Time,Raw Usage,Fair Share,Level FS" >
    ↪  $OUTPUT_FILE
fi

# Append metrics to the output file
echo "$JOB_NAME,$DATE,$SLURM_OUT" >> $OUTPUT_FILE
```

Code A.1: The bash script used to retreive Slurm share metrics of usage, fairshare, and level fairshare, for a specified user and account. Tested on systems running Slurm 23.02.

Appendix B

Auto-ECEarth3 complete list of share metrics

MareNostrum 4

Wrapped

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a6zi_18500101_fc0_1_SIM	2024-03-19_13:52:45	10983	0.324708	2074.227310
a6zi_18500101_fc0_2_SIM	2024-03-19_17:02:54	5883927	0.323038	3.827168
a6zi_18500101_fc0_3_SIM	2024-03-19_20:11:27	11903127	0.323038	1.897283
a6zi_18500101_fc0_3_SIM	2024-03-19_23:13:28	13351959	0.322760	1.694321
a6zi_18500101_fc0_4_SIM	2024-03-20_02:23:20	19360599	0.322482	1.185026
a6zi_18500101_fc0_5_SIM	2024-03-20_05:32:25	25221399	0.322343	0.913999
a6zi_18500101_fc0_6_SIM	2024-03-20_08:42:09	31240599	0.322204	0.741621
a6zi_18500101_fc0_7_SIM	2024-03-20_11:52:29	37259799	0.322204	0.626971
a6zi_18500101_fc0_8_SIM	2024-03-20_15:01:33	43278999	0.322298	0.544842
a6zi_18500101_fc0_9_SIM	2024-03-20_18:09:58	49298199	0.322298	0.480050
a6zi_18500101_fc0_10_SIM	2024-03-20_21:18:23	55317399	0.322298	0.428526
a6zi_18500101_fc0_11_SIM	2024-03-21_00:27:15	61178199	0.322298	0.390675
a6zi_18500101_fc0_12_SIM	2024-03-21_03:36:46	67197399	0.322298	0.358560
a6zi_18500101_fc0_13_SIM	2024-03-21_07:50:00	73310055	0.322298	0.334449
a6zi_18500101_fc0_14_SIM	2024-03-21_10:58:10	79269591	0.322168	0.310632
a6zi_18500101_fc0_15_SIM	2024-03-21_14:06:16	85130391	0.321890	0.289568
a6zi_18500101_fc0_16_SIM	2024-03-21_17:15:16	91149591	0.321984	0.270769
a6zi_18500101_fc0_17_SIM	2024-03-21_20:23:12	97168791	0.321984	0.254766
a6zi_18500101_fc0_18_SIM	2024-03-21_23:30:47	103029591	0.321984	0.241028
a6zi_18500101_fc0_19_SIM	2024-03-22_02:38:50	109048791	0.321984	0.230403
a6zi_18500101_fc0_20_SIM	2024-03-22_05:47:14	114909591	0.321984	0.220325
a6zi_18500101_fc0_21_SIM	2024-03-22_08:54:31	120928791	0.321984	0.211063
a6zi_18500101_fc0_22_SIM	2024-03-22_12:02:05	126789591	0.321984	0.202514
a6zi_18500101_fc0_23_SIM	2024-03-22_16:06:04	132881655	0.404530	0.194801
a6zi_18500101_fc0_24_SIM	2024-03-22_19:15:18	138807927	0.404530	0.189200
a6zi_18500101_fc0_25_SIM	2024-03-22_22:23:52	144827127	0.404530	0.182911
a6zi_18500101_fc0_26_SIM	2024-03-23_01:31:55	150687927	0.321984	0.176761
a6zi_18500101_fc0_27_SIM	2024-03-23_04:40:33	156707127	0.321984	0.170758
a6zi_18500101_fc0_28_SIM	2024-03-23_07:49:57	162726327	0.321984	0.165248
a6zi_18500101_fc0_29_SIM	2024-03-23_10:58:04	168587127	0.321984	0.160250
a6zi_18500101_fc0_30_SIM	2024-03-23_14:06:39	174606327	0.321984	0.155279
a6zi_18500101_fc0_31_SIM	2024-03-23_17:14:12	180625527	0.321984	0.150626

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a6zi_18500101_fc0_32_SIM	2024-03-23_20:24:10	186644727	0.321984	0.146332
a6zi_18500101_fc0_33_SIM	2024-03-23_23:34:04	192676599	0.321984	0.142174
a6zi_18500101_fc0_34_SIM	2024-03-24_02:43:48	198665703	0.321984	0.138238
a6zi_18500101_fc0_35_SIM	2024-03-24_05:52:51	204526503	0.321984	0.134630
a6zi_18500101_fc0_36_SIM	2024-03-24_09:02:22	210545703	0.321984	0.131126
a6zi_18500101_fc0_37_SIM	2024-03-24_12:11:25	216564903	0.321984	0.127816
a6zi_18500101_fc0_38_SIM	2024-03-24_15:20:11	222584103	0.321984	0.124645
a6zi_18500101_fc0_39_SIM	2024-03-24_18:29:04	228603303	0.321984	0.121671
a6zi_18500101_fc0_40_SIM	2024-03-24_21:39:30	234622503	0.321984	0.118819
a6zi_18500101_fc0_41_SIM	2024-03-25_00:49:08	240641703	0.321845	0.116169
a6zi_18500101_fc0_42_SIM	2024-03-25_03:58:02	246502503	0.321845	0.113716
a6zi_18500101_fc0_43_SIM	2024-03-25_07:07:25	252634167	0.321845	0.111302
a6zi_18500101_fc0_44_SIM	2024-03-25_10:17:08	258517671	0.321622	0.107738
a6zi_18500101_fc0_45_SIM	2024-03-25_13:27:00	264536871	0.321622	0.105494
a6zi_18500101_fc0_46_SIM	2024-03-25_16:36:16	270556071	0.321622	0.103310
a6zi_18500101_fc0_47_SIM	2024-03-25_19:46:20	276575271	0.321622	0.101286
a6zi_18500101_fc0_48_SIM	2024-03-25_22:56:51	282594471	0.404388	0.099367
a6zi_18500101_fc0_49_SIM	2024-03-26_02:05:45	288613671	0.404388	0.097564
a6zi_18500101_fc0_50_SIM	2024-03-26_05:15:12	294632871	0.321622	0.095822

Table B.1: *a6zi* (wrapped) experiment execution parameters on MareNostrum 4.

Unwrapped

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a6zs_18500101_fc0_1_SIM	2024-03-19_13:52:45	10896	0.324847	2079.237521
a6zs_18500101_fc0_2_SIM	2024-03-19_22:57:25	6070224	0.323178	3.723313
a6zs_18500101_fc0_3_SIM	2024-03-20_02:11:16	12074112	0.323178	1.898252
a6zs_18500101_fc0_4_SIM	2024-03-20_05:21:46	18086448	0.322621	1.280967
a6zs_18500101_fc0_5_SIM	2024-03-20_08:33:01	24084528	0.322482	0.965336
a6zs_18500101_fc0_6_SIM	2024-03-20_13:10:41	30106896	0.322482	0.780776
a6zs_18500101_fc0_7_SIM	2024-03-20_23:21:32	36096000	0.322437	0.658668
a6zs_18500101_fc0_8_SIM	2024-03-21_04:19:53	42121008	0.322437	0.572539
a6zs_18500101_fc0_9_SIM	2024-03-21_07:50:00	48114864	0.322437	0.509582
a6zs_18500101_fc0_10_SIM	2024-03-22_00:45:35	54101328	0.322401	0.460324
a6zs_18500101_fc0_11_SIM	2024-03-22_03:57:45	60105744	0.322401	0.420420
a6zs_18500101_fc0_12_SIM	2024-03-22_07:29:47	66122304	0.322401	0.384382
a6zs_18500101_fc0_12_SIM	2024-03-22_12:18:54	72783024	0.322401	0.352916
a6zs_18500101_fc0_13_SIM	2024-03-22_16:07:24	78783744	0.404947	0.328564
a6zs_18500101_fc0_14_SIM	2024-03-22_19:21:48	84768624	0.404808	0.309983
a6zs_18500101_fc0_15_SIM	2024-03-22_22:48:18	90738720	0.404808	0.292149
a6zs_18500101_fc0_16_SIM	2024-03-23_01:58:23	96733104	0.322262	0.275548
a6zs_18500101_fc0_17_SIM	2024-03-23_05:07:12	102690000	0.322123	0.261065
a6zs_18500101_fc0_18_SIM	2024-03-23_08:16:48	108659568	0.322123	0.247792
a6zs_18500101_fc0_19_SIM	2024-03-23_11:26:22	114651840	0.322123	0.235969
a6zs_18500101_fc0_20_SIM	2024-03-23_14:37:07	120665760	0.322123	0.224997
a6zs_18500101_fc0_21_SIM	2024-03-23_17:45:15	126617904	0.322123	0.215103
a6zs_18500101_fc0_22_SIM	2024-03-23_20:53:49	132566352	0.322123	0.206147

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a6zs_18500101_fc0_23_SIM	2024-03-24_00:03:33	138552288	0.322123	0.197787
a6zs_18500101_fc0_24_SIM	2024-03-24_03:14:20	144563040	0.322123	0.190061
a6zs_18500101_fc0_25_SIM	2024-03-24_06:23:47	150538944	0.322123	0.182997
a6zs_18500101_fc0_26_SIM	2024-03-24_09:32:40	156501648	0.322123	0.176629
a6zs_18500101_fc0_27_SIM	2024-03-24_12:42:20	162495504	0.322123	0.170533
a6zs_18500101_fc0_28_SIM	2024-03-24_15:51:51	168488832	0.322123	0.164828
a6zs_18500101_fc0_29_SIM	2024-03-24_19:14:00	174448368	0.322123	0.159525
a6zs_18500101_fc0_30_SIM	2024-03-24_22:29:44	180418992	0.322123	0.154603
a6zs_18500101_fc0_31_SIM	2024-03-25_01:38:11	186367440	0.322123	0.150109
a6zs_18500101_fc0_32_SIM	2024-03-25_04:47:53	192348624	0.322123	0.145927
a6zs_18500101_fc0_33_SIM	2024-03-25_07:56:24	198306576	0.322123	0.141903
a6zs_18500101_fc0_34_SIM	2024-03-25_20:41:14	204252384	0.321900	0.137223
a6zs_18500101_fc0_35_SIM	2024-03-25_23:57:19	210255216	0.404666	0.133681
a6zs_18500101_fc0_36_SIM	2024-03-26_03:05:40	216212640	0.404666	0.130384
a6zs_18500101_fc0_37_SIM	2024-03-26_06:16:13	222164784	0.321900	0.127185
a6zs_18500101_fc0_38_SIM	2024-03-26_12:35:13	228142800	0.404831	0.124140
a6zs_18500101_fc0_39_SIM	2024-03-26_15:57:44	234125568	0.404831	0.121189
a6zs_18500101_fc0_40_SIM	2024-03-26_19:13:45	240137904	0.404831	0.118391
a6zs_18500101_fc0_41_SIM	2024-03-27_00:09:38	246135984	0.404692	0.115775
a6zs_18500101_fc0_42_SIM	2024-03-27_03:20:39	252152016	0.404692	0.113201
a6zs_18500101_fc0_43_SIM	2024-03-27_06:55:23	258096768	0.404692	0.110751
a6zs_18500101_fc0_44_SIM	2024-03-27_10:05:40	264102768	0.404692	0.108607
a6zs_18500101_fc0_45_SIM	2024-03-27_13:15:38	270115632	0.404692	0.107148
a6zs_18500101_fc0_46_SIM	2024-03-27_16:49:17	276171792	0.321672	0.105870
a6zs_18500101_fc0_47_SIM	2024-03-27_20:16:29	282152448	0.321672	0.103947
a6zs_18500101_fc0_48_SIM	2024-03-27_23:31:04	288145776	0.321672	0.101950
a6zs_18500101_fc0_49_SIM	2024-03-28_02:41:29	294141744	0.321672	0.100042
a6zs_18500101_fc0_50_SIM	2024-03-28_05:52:36	300165168	0.321672	0.098211

Table B.2: *a6zs* (unwrapped) experiment execution parameters on MareNostrum 4.

MeluXina

Wrapped

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a72x_19900201_fc00_1_SIM	2024-04-23_12:28:30	492800	0.520485	33.621002
a72x_19900201_fc00_2_SIM	2024-04-23_13:39:57	11441920	0.509164	1.409773
a72x_19900201_fc00_3_SIM	2024-04-23_14:51:49	22193920	0.509164	0.762471
a72x_19900201_fc00_4_SIM	2024-04-23_16:04:17	32945920	0.508890	0.535293
a72x_19900201_fc00_5_SIM	2024-04-23_17:15:54	44465920	0.508621	0.413838
a72x_19900201_fc00_6_SIM	2024-04-23_18:29:13	55217920	0.508351	0.346197
a72x_19900201_fc00_7_SIM	2024-04-23_19:40:26	66737920	0.508351	0.297906
a72x_19900201_fc00_8_SIM	2024-04-23_20:53:00	77489920	0.508351	0.265794
a72x_19900201_fc00_9_SIM	2024-04-23_22:04:09	88241920	0.508351	0.241509
a72x_19900201_fc00_10_SIM	2024-04-23_23:15:37	99761920	0.507543	0.221291
a72x_19900201_fc00_11_SIM	2024-04-24_00:27:30	110913280	0.507543	0.206183

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a72x_19900201_fc00_12_SIM	2024-04-24_01:39:17	121248000	0.507543	0.193820
a72x_19900201_fc00_13_SIM	2024-04-24_02:51:38	132768000	0.507274	0.182770
a72x_19900201_fc00_14_SIM	2024-04-24_04:03:19	143520000	0.507274	0.174062
a72x_19900201_fc00_15_SIM	2024-04-24_05:15:42	155040000	0.507274	0.166067
a72x_19900201_fc00_16_SIM	2024-04-24_06:28:29	165792000	0.507274	0.159601
a72x_19900201_fc00_17_SIM	2024-04-24_07:40:33	177312000	0.507274	0.153543
a72x_19900201_fc00_18_SIM	2024-04-24_08:52:19	188064000	0.507274	0.148563
a72x_19900201_fc00_19_SIM	2024-04-24_10:04:03	198816000	0.506193	0.144129
a72x_19900201_fc00_20_SIM	2024-04-24_11:15:56	210336000	0.506193	0.139871
a72x_19900201_fc00_21_SIM	2024-04-24_12:28:10	221571840	0.506326	0.136367
a72x_19900201_fc00_22_SIM	2024-04-24_13:39:52	232569600	0.506322	0.133413
a72x_19900201_fc00_23_SIM	2024-04-24_14:51:42	243321600	0.506322	0.130777
a72x_19900201_fc00_24_SIM	2024-04-24_16:03:07	254073600	0.506053	0.128016
a72x_19900201_fc00_25_SIM	2024-04-24_17:14:21	264825600	0.506053	0.125502
a72x_19900201_fc00_26_SIM	2024-04-24_18:26:01	276345600	0.506053	0.123030
a72x_19900201_fc00_27_SIM	2024-04-24_19:37:32	287097600	0.506053	0.120906
a72x_19900201_fc00_28_SIM	2024-04-24_20:49:38	297849600	0.506053	0.118946
a72x_19900201_fc00_29_SIM	2024-04-24_22:01:11	309369600	0.506053	0.116992
a72x_19900201_fc00_30_SIM	2024-04-24_23:13:00	320121600	0.506053	0.115297
a72x_19900201_fc00_31_SIM	2024-04-25_00:25:35	331669760	0.506053	0.113604
a72x_19900201_fc00_32_SIM	2024-04-25_01:38:29	342301440	0.506053	0.112149
a72x_19900201_fc00_33_SIM	2024-04-25_02:49:55	353821440	0.506053	0.110662
a72x_19900201_fc00_34_SIM	2024-04-25_04:02:11	364573440	0.506053	0.109360
a72x_19900201_fc00_35_SIM	2024-04-25_05:15:51	376093440	0.506053	0.108047
a72x_19900201_fc00_36_SIM	2024-04-25_06:27:49	386845440	0.506053	0.106893
a72x_19900201_fc00_37_SIM	2024-04-25_07:39:36	397597440	0.506053	0.105804
a72x_19900201_fc00_38_SIM	2024-04-25_08:50:58	409117440	0.506053	0.104695
a72x_19900201_fc00_39_SIM	2024-04-25_10:03:45	419869440	0.506053	0.103717
a72x_19900201_fc00_40_SIM	2024-04-25_11:17:03	431389440	0.503878	0.102723
a72x_19900201_fc00_41_SIM	2024-04-25_12:29:48	442835200	0.503277	0.101776
a72x_19900201_fc00_42_SIM	2024-04-25_13:43:03	453584640	0.503277	0.100942
a72x_19900201_fc00_43_SIM	2024-04-25_14:55:55	465104640	0.503277	0.100186
a72x_19900201_fc00_44_SIM	2024-04-25_16:10:06	476624640	0.503277	0.099608
a72x_19900201_fc00_45_SIM	2024-04-25_17:23:21	487376640	0.503277	0.098979
a72x_19900201_fc00_46_SIM	2024-04-25_18:36:44	498896640	0.503277	0.098221
a72x_19900201_fc00_47_SIM	2024-04-25_19:49:56	510416640	0.503277	0.097506
a72x_19900201_fc00_48_SIM	2024-04-25_21:03:38	521168640	0.503277	0.096867
a72x_19900201_fc00_49_SIM	2024-04-25_22:17:40	532688640	0.503277	0.096115
a72x_19900201_fc00_50_SIM	2024-04-25_23:31:19	544208640	0.503277	0.094786

Table B.3: *a72x* (wrapped) experiment execution parameters on MeluXina.

Unwrapped

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a74v_19900201_fc00_1_SIM	2024-04-23_12:28:30	486912	0.520755	34.042702
a74v_19900201_fc00_2_SIM	2024-04-23_13:36:34	10844672	0.509434	1.506818
a74v_19900201_fc00_3_SIM	2024-04-23_14:45:08	21309952	0.509434	0.791781

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a74v_19900201_fc00_4_SIM	2024-04-23_15:53:43	31747072	0.509159	0.561349
a74v_19900201_fc00_5_SIM	2024-04-23_17:00:51	41992192	0.508890	0.435643
a74v_19900201_fc00_6_SIM	2024-04-23_18:08:55	52385792	0.508621	0.365024
a74v_19900201_fc00_7_SIM	2024-04-23_19:16:32	62689792	0.508621	0.314050
a74v_19900201_fc00_8_SIM	2024-04-23_20:24:37	73088512	0.508621	0.280246
a74v_19900201_fc00_9_SIM	2024-04-23_21:32:14	83412992	0.508621	0.252793
a74v_19900201_fc00_10_SIM	2024-04-23_22:40:49	93862912	0.508621	0.231593
a74v_19900201_fc00_11_SIM	2024-04-23_23:49:57	104438272	0.507812	0.214808
a74v_19900201_fc00_11_SIM	2024-04-24_00:01:28	106038272	0.507812	0.212709
a74v_19900201_fc00_12_SIM	2024-04-24_01:09:33	116436992	0.507812	0.200366
a74v_19900201_fc00_13_SIM	2024-04-24_02:18:09	126902272	0.507812	0.189079
a74v_19900201_fc00_14_SIM	2024-04-24_03:25:45	137244672	0.507543	0.179560
a74v_19900201_fc00_15_SIM	2024-04-24_04:33:51	147633152	0.507543	0.171937
a74v_19900201_fc00_16_SIM	2024-04-24_05:42:28	158075392	0.507543	0.164826
a74v_19900201_fc00_17_SIM	2024-04-24_06:50:35	168433152	0.507543	0.158635
a74v_19900201_fc00_18_SIM	2024-04-24_07:58:41	178816512	0.507543	0.153559
a74v_19900201_fc00_19_SIM	2024-04-24_09:06:47	189251072	0.507543	0.148664
a74v_19900201_fc00_20_SIM	2024-04-24_10:15:27	199700992	0.506462	0.144264
a74v_19900201_fc00_21_SIM	2024-04-24_11:25:04	210330112	0.506462	0.140359
a74v_19900201_fc00_22_SIM	2024-04-24_12:34:41	220931072	0.506595	0.137134
a74v_19900201_fc00_23_SIM	2024-04-24_13:44:18	231537152	0.506591	0.134360
a74v_19900201_fc00_24_SIM	2024-04-24_14:56:29	242258432	0.506591	0.131548
a74v_19900201_fc00_25_SIM	2024-04-24_16:05:57	252741632	0.506322	0.128892
a74v_19900201_fc00_26_SIM	2024-04-24_17:13:57	263127552	0.506322	0.126591
a74v_19900201_fc00_27_SIM	2024-04-24_18:22:24	273544192	0.506322	0.124254
a74v_19900201_fc00_28_SIM	2024-04-24_19:30:56	283958272	0.506322	0.122103
a74v_19900201_fc00_29_SIM	2024-04-24_20:38:56	294382592	0.506322	0.120253
a74v_19900201_fc00_30_SIM	2024-04-24_21:47:26	304837632	0.506322	0.118367
a74v_19900201_fc00_31_SIM	2024-04-24_22:54:57	315164672	0.506322	0.116626
a74v_19900201_fc00_32_SIM	2024-04-25_00:03:24	325635072	0.506322	0.115098
a74v_19900201_fc00_33_SIM	2024-04-25_01:11:55	336141312	0.506322	0.113549
a74v_19900201_fc00_34_SIM	2024-04-25_02:20:56	346678272	0.506322	0.112099
a74v_19900201_fc00_35_SIM	2024-04-25_03:29:25	357140992	0.506322	0.110841
a74v_19900201_fc00_36_SIM	2024-04-25_04:37:55	367608832	0.506322	0.109559
a74v_19900201_fc00_37_SIM	2024-04-25_05:45:52	378004992	0.506322	0.108349
a74v_19900201_fc00_38_SIM	2024-04-25_06:53:55	388431872	0.506322	0.107284
a74v_19900201_fc00_39_SIM	2024-04-25_08:02:22	398866432	0.506322	0.106192
a74v_19900201_fc00_40_SIM	2024-04-25_09:11:26	409421312	0.506322	0.105169
a74v_19900201_fc00_41_SIM	2024-04-25_10:23:24	420416512	0.504145	0.104191
a74v_19900201_fc00_42_SIM	2024-04-25_11:33:23	430889472	0.504145	0.103264
a74v_19900201_fc00_43_SIM	2024-04-25_12:40:53	441236992	0.503539	0.102411
a74v_19900201_fc00_44_SIM	2024-04-25_13:49:22	451679232	0.503539	0.101623
a74v_19900201_fc00_45_SIM	2024-04-25_14:58:19	462211072	0.503539	0.100919
a74v_19900201_fc00_46_SIM	2024-04-25_16:09:19	473034752	0.503539	0.100376
a74v_19900201_fc00_47_SIM	2024-04-25_17:20:50	483935232	0.503539	0.099699
a74v_19900201_fc00_48_SIM	2024-04-25_18:31:54	494756352	0.503539	0.098992
a74v_19900201_fc00_49_SIM	2024-04-25_19:41:52	505457152	0.503539	0.098316
a74v_19900201_fc00_50_SIM	2024-04-25_20:53:23	516388352	0.503539	0.097662

Table B.4: *a74v* (unwrapped) experiment execution parameters on MeluXina.

MareNostrum 5

Wrapped

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a7d4_19900201_fc00_1_SIM	2024-06-14_18:14:07	797174	0.235337	298.045981
a7d4_19900201_fc00_2_SIM	2024-06-14_19:50:07	9463510	0.224475	25.186246
a7d4_19900201_fc00_3_SIM	2024-06-14_21:28:58	18871510	0.222303	12.679496
a7d4_19900201_fc00_4_SIM	2024-06-14_23:08:12	28279510	0.219406	8.497225
a7d4_19900201_fc00_5_SIM	2024-06-15_00:47:42	37687510	0.216510	6.404242
a7d4_19900201_fc00_6_SIM	2024-06-15_02:27:40	47095510	0.215062	5.146422
a7d4_19900201_fc00_7_SIM	2024-06-15_04:07:35	56503510	0.210717	4.307392
a7d4_19900201_fc00_8_SIM	2024-06-15_05:49:24	65911510	0.210717	3.707136
a7d4_19900201_fc00_9_SIM	2024-06-15_07:30:34	75319510	0.210717	3.256294
a7d4_19900201_fc00_10_SIM	2024-06-15_09:11:38	85197910	0.210717	2.890378
a7d4_19900201_fc00_11_SIM	2024-06-15_10:52:52	94718806	0.210717	2.613687
a7d4_19900201_fc00_12_SIM	2024-06-15_12:34:34	103973142	0.210717	2.387928
a7d4_19900201_fc00_13_SIM	2024-06-15_14:15:58	113381142	0.209269	2.198755
a7d4_19900201_fc00_14_SIM	2024-06-15_15:56:44	123259542	0.209269	2.030956
a7d4_19900201_fc00_15_SIM	2024-06-15_17:37:50	132667542	0.209269	1.894345
a7d4_19900201_fc00_16_SIM	2024-06-15_19:18:40	142075542	0.208545	1.775864
a7d4_19900201_fc00_17_SIM	2024-06-15_21:02:57	151953942	0.208545	1.666934
a7d4_19900201_fc00_18_SIM	2024-06-15_22:43:22	161361942	0.208545	1.575046
a7d4_19900201_fc00_19_SIM	2024-06-16_00:23:55	170769942	0.207820	1.493009
a7d4_19900201_fc00_20_SIM	2024-06-16_02:05:15	180177942	0.207096	1.419403
a7d4_19900201_fc00_21_SIM	2024-06-16_03:46:42	190064182	0.206372	1.349886
a7d4_19900201_fc00_22_SIM	2024-06-16_05:26:50	199429846	0.214337	1.290164
a7d4_19900201_fc00_23_SIM	2024-06-16_07:07:14	208837846	0.214337	1.235381
a7d4_19900201_fc00_24_SIM	2024-06-16_08:47:34	218245846	0.213613	1.184879
a7d4_19900201_fc00_25_SIM	2024-06-16_10:28:02	227653846	0.212889	1.138838
a7d4_19900201_fc00_26_SIM	2024-06-16_12:07:44	237061846	0.210717	1.096626
a7d4_19900201_fc00_27_SIM	2024-06-16_13:48:29	246469846	0.201303	1.057838
a7d4_19900201_fc00_28_SIM	2024-06-16_15:28:39	255877846	0.201303	1.021765
a7d4_19900201_fc00_29_SIM	2024-06-16_17:09:10	265285846	0.201303	0.988349
a7d4_19900201_fc00_30_SIM	2024-06-16_18:50:12	274693846	0.201303	0.957374
a7d4_19900201_fc00_31_SIM	2024-06-16_20:31:38	284550294	0.201303	0.926962
a7d4_19900201_fc00_32_SIM	2024-06-16_22:13:42	293922230	0.201303	0.900055
a7d4_19900201_fc00_33_SIM	2024-06-16_23:54:15	303330230	0.201303	0.874704
a7d4_19900201_fc00_34_SIM	2024-06-17_01:33:57	312738230	0.200579	0.850811
a7d4_19900201_fc00_35_SIM	2024-06-17_03:14:18	322146230	0.200579	0.828293
a7d4_19900201_fc00_36_SIM	2024-06-17_04:55:37	331554230	0.200579	0.807039
a7d4_19900201_fc00_37_SIM	2024-06-17_06:36:03	340962230	0.200579	0.787480
a7d4_19900201_fc00_38_SIM	2024-06-17_08:17:07	350840630	0.200145	0.762800
a7d4_19900201_fc00_39_SIM	2024-06-17_09:59:07	360248630	0.199856	0.745102
a7d4_19900201_fc00_40_SIM	2024-06-17_11:41:28	370127030	0.199134	0.730330

Table B.5: *a7d4* (wrapped) experiment execution parameters on MareNostrum 5.

Unwrapped

Job Name	Start Time (+00:00)	Raw Usage	Fair Share	Level FS
a7d5_19900201_fc00_1_SIM	2024-06-14_18:04:25	17916324	0.221579	13.243435
a7d5_19900201_fc00_2_SIM	2024-06-14_19:43:06	27103236	0.219406	8.817363
a7d5_19900201_fc00_3_SIM	2024-06-14_21:23:26	36423428	0.215786	6.583782
a7d5_19900201_fc00_4_SIM	2024-06-14_23:03:32	45743620	0.214337	5.265715
a7d5_19900201_fc00_5_SIM	2024-06-15_00:43:20	55046564	0.211441	4.392472
a7d5_19900201_fc00_6_SIM	2024-06-15_02:23:27	64415364	0.209993	3.771170
a7d5_19900201_fc00_7_SIM	2024-06-15_04:05:12	73895492	0.209993	3.305496
a7d5_19900201_fc00_8_SIM	2024-06-15_05:47:45	83490084	0.209993	2.929691
a7d5_19900201_fc00_9_SIM	2024-06-15_07:29:09	93006276	0.209993	2.644185
a7d5_19900201_fc00_10_SIM	2024-06-15_09:12:33	102666724	0.209993	2.399861
a7d5_19900201_fc00_11_SIM	2024-06-15_10:54:59	112212708	0.209269	2.208708
a7d5_19900201_fc00_12_SIM	2024-06-15_12:36:55	121746148	0.208545	2.039880
a7d5_19900201_fc00_13_SIM	2024-06-15_14:19:48	131314084	0.208545	1.902112
a7d5_19900201_fc00_14_SIM	2024-06-15_16:00:43	140761284	0.207820	1.783182
a7d5_19900201_fc00_15_SIM	2024-06-15_17:42:07	150200644	0.207820	1.673554
a7d5_19900201_fc00_16_SIM	2024-06-15_19:25:08	159768580	0.207820	1.581912
a7d5_19900201_fc00_17_SIM	2024-06-15_21:09:25	169529380	0.207096	1.496592
a7d5_19900201_fc00_18_SIM	2024-06-15_22:50:16	178984420	0.206372	1.422943
a7d5_19900201_fc00_19_SIM	2024-06-16_00:30:43	188420644	0.205648	1.356294
a7d5_19900201_fc00_20_SIM	2024-06-16_02:12:16	197907044	0.205648	1.292832
a7d5_19900201_fc00_21_SIM	2024-06-16_03:53:54	207395012	0.213613	1.238260
a7d5_19900201_fc00_22_SIM	2024-06-16_05:35:46	216829668	0.212165	1.188304
a7d5_19900201_fc00_23_SIM	2024-06-16_07:17:48	226344292	0.212165	1.140461
a7d5_19900201_fc00_24_SIM	2024-06-16_09:00:17	235847940	0.210717	1.097820
a7d5_19900201_fc00_25_SIM	2024-06-16_10:41:34	245284164	0.208545	1.057413
a7d5_19900201_fc00_26_SIM	2024-06-16_12:22:42	254704708	0.208545	1.021276
a7d5_19900201_fc00_27_SIM	2024-06-16_14:03:43	264178564	0.200579	0.988107
a7d5_19900201_fc00_28_SIM	2024-06-16_15:43:10	273489348	0.200579	0.956863
a7d5_19900201_fc00_29_SIM	2024-06-16_17:24:26	282908324	0.200579	0.927803
a7d5_19900201_fc00_30_SIM	2024-06-16_19:05:09	292316324	0.200579	0.900974
a7d5_19900201_fc00_31_SIM	2024-06-16_20:45:39	301699236	0.200579	0.875509
a7d5_19900201_fc00_32_SIM	2024-06-16_22:27:29	311154276	0.199855	0.850590
a7d5_19900201_fc00_33_SIM	2024-06-17_00:09:13	320621860	0.199855	0.828345
a7d5_19900201_fc00_34_SIM	2024-06-17_01:49:04	329956164	0.199855	0.807258
a7d5_19900201_fc00_35_SIM	2024-06-17_03:29:19	339348484	0.199855	0.787212
a7d5_19900201_fc00_36_SIM	2024-06-17_05:09:15	348723556	0.199855	0.768192
a7d5_19900201_fc00_37_SIM	2024-06-17_06:49:34	358140964	0.199855	0.750639
a7d5_19900201_fc00_38_SIM	2024-06-17_08:30:07	367541124	0.198699	0.728952
a7d5_19900201_fc00_39_SIM	2024-06-17_10:11:48	377063588	0.198413	0.712371
a7d5_19900201_fc00_40_SIM	2024-06-17_11:53:06	386587620	0.198413	0.699941

Table B.6: *a7d5* (unwrapped) experiment execution parameters on MareNostrum 5.