Task aggregation as a strategy to optimize Earth System Model workflows in HPC: assessing real scenarios with EC-Earth

Pablo Goitia[1,2], Manuel G. Marciani[1], Miguel Castrillo[1], and Mario C. Acosta[1]

[1]Barcelona Supercomputing Center (BSC), Barcelona, Spain
[2]University of Cantabria (UC), Santander, Spain

Correspondence: Pablo Goitia (pablo.goitia@bsc.es)

Abstract.

Earth System Models (ESMs) are commonly executed as complex workflows consisting of numerous interdependent tasks that comprise steps such as model and data deployment, simulation, data transfer, and post-processing. Workflows facilitate the execution of long high-resolution configurations by splitting the runtime of the simulation into different tasks, in order to ensure frequent checkpointing and to comply with the restrictions of the large High-Performance Computing (HPC) machines where they are executed. These machines are frequently congested due to their high demand and, therefore, implement scheduling policies to share the resources among the users. This complicates the execution of long ensemble simulation workflows, where each successive simulation task can only be submitted once the preceding one finishes, causing queue time to accumulate, which is the duration for which the jobs wait for the HPC platform to allocate the required resources for their execution.

To alleviate this issue, we propose achieving shorter times-to-response, which are the intervals from the first submission to the completion of the final task, by applying a solution to reduce subsequent requests for resources and, consequently, reducing queue times. This solution, known as task aggregation, consists of grouping multiple tasks and submitting them as a single job, respecting their dependencies, and without altering their underlying logic. This technique separates the workflow definition from how individual tasks are submitted and executed, allowing the workflow manager to make decisions during runtime.

In this paper, we performed the first controlled assessment on the effects of task aggregation by conducting concurrent pairs of climate simulations in production machines, with the sole difference that one uses aggregation. These simulations were executed on three European supercomputers: MeluXina, MareNostrum 4, and MareNostrum 5. We measured the evolution of the fair share, a scheduling factor that normally plays a major role in the priority of the jobs. Besides absolute gains, we compute the impact of aggregation by comparing consolidated performance metrics for climate models within the community.

We prove the benefits of the task aggregation using EC-Earth3, a widely used European community climate model that shares main features with many other ESMs and has a representative workload. The experimental findings of our research indicate that across the three evaluated supercomputing platforms, *applying task aggregation decreases total queue times by 11.17 to 12.33 times compared to a workflow that does not*, representing an improvement of up to 23% more years simulated in the same time span in the case of the platform with the highest congestion.

Therefore, results have shown that task aggregation proves to be beneficial for long climate simulations. Moreover, we have credible reasons to believe that any vertical (also called chained) workflow should benefit from using it. We explain that this

reduction in the time-to-solution comes from the decrease in the number of submitted jobs and the utilization of the machine. By aggregating tasks, we had many times less jobs queued and, albeit their longer length, we observed that they are in queue less time than if they had been submitted individually. Our results also show that the user's jobs are being held in queue in spite of their utilization due to the fair share being influenced by the other members of the HPC account, which is a direct consequence of the fair share policy of the machine.

1 Introduction

Over the past decades, the High-Performance Computing (HPC) domain has undergone remarkable growth, currently featuring sophisticated heterogeneous systems that have allowed the expansion of applications in disciplines such as Earth sciences, where one of the most ambitious current projects is the development of digital twins of the Earth, within the European Commission's Destination Earth project (Hoffmann et al., 2023).

In the weather and climate modeling field, the improvement of, for example, climate change studies relies on the development and continuous enhancement of 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. They are usually executed on prominent supercomputing platforms, structured as extensive workflows with hundreds of thousands of interdependent tasks—the indivisible piece of computation defined by the workflow—which usually require considerable parallel resource allocation and a significant amount of time to finish. In the case of the Coupled Model Intercomparison Project Phase 6 (CMIP6) exercise (Eyring et al., 2016), some participating simulations took 83 days to complete, and a very high-resolution (5 km horizontal grid) 30 year global simulation of Destination Earth required three months to run. That time, which spans between the submission of the first job—the computation that is executed in remote—and the completion of the very last one, will hereafter be referred to as time-to-solution. Besides execution time, this metric takes into account failures and the time spent waiting in the queue of the HPC's scheduler until the necessary resources to run are available.

Traditionally, developers often work to enhance both the computational and energy efficiency of the Earth System Model simulations (Irrmann et al., 2022), with the aim of making better use of resources, which, in turn, leads to the capability to perform more accurate simulations. This is a critical task for the Earth science community because of the cost of the simulations, on the order of millions of core hours spent, and significant due to the impact of its results on the climate change assessment. These enhancements directly impact metrics such as the Simulated Years Per Day (SYPD), which is the division of the total time in years simulated by the ESM by its runtime in days, and the Actual Simulated Years Per Day (ASYPD), which considers the queue time and failed executions alongside the runtime. Both performance metrics were proposed by Balaji et al. (2017) in order to benchmark and compare ESM simulations, and have now become standard.

However, there exists another less studied concern within the community, namely the long queue times at highly congested HPC platforms that each job of the simulation of these models has to overcome. The only estimate for climate simulations of the impact of queue time in performance comes from Acosta et al. (2024), where the authors reported queue time overheads that represent 10% to 20% of the total runtime for simulations of the CMIP6 exercise. Therefore, we see that this time can have a severe impact on the time-to-solution.

Other remarks on job queue times come from works that characterize the utilization of HPC platforms (Jones et al., 2017; Schlagkamp et al., 2016; Rodrigo et al., 2018), but out of all of them, we highlight the article by Patel et al. (2020). The authors noted that “the queue wait times for resources are increasing rapidly and are often more than 7x the run time, especially for large jobs” and also that “HPC resource management strategies may have ‘unintentional‘ unfairness side–effects". This findings support our motivation that queue times are sometimes unreasonable in certain scenarios, like the one aforementioned about Earth sciences applications.

Therefore, if we manage to minimize the time that the workflow jobs spend in the workload manager’s queue, the overall execution time of the entire model would also be reduced, and consequently the ASYPD would increase. With this in mind, the developers of Autosubmit (Manubens-Gil et al., 2016), a workflow manager specifically tailored for Earth sciences use cases, implemented a technique called *wrapping*, which consists of aggregating multiple tasks of these workflows into a single larger task—known as a *wrapper*—to be then dispatched to a remote scheduler like Slurm at once. The tasks are not altered, they are just joined and their dependencies and the internal logic are still respected. Task aggregation only specifies how the jobs should be submitted to the scheduler. Model features such as restart files remain untouched and can be leveraged alongside task retries to recover simulations in the event of failures during the runtime of the aggregated tasks.

However, to the best of our knowledge, task aggregation has been used without validating its effectiveness for queue reduction. In a recent paper, Marciani et al. (2025) explored the task aggregation solution by conducting multiple experiments in a simulated environment, analyzing the relationship between the job’s request, the queue time, and the fair share factor, which is the Slurm scheduling factor that balances the utilization of computational resources between users by prioritizing less served users over high-usage ones. By default, Slurm computes this factor considering the usage of the user and also its account—a group of users that normally shares a project. Therefore, depending on the configuration, even if a user has not been using the machine, its jobs will be penalized because of the usage done by its peers. Among other findings, they state that applying it to vertical workflows, i.e., where each task depends on the previous one, which is also prevalent in most ESMs, reduces overall queue durations.

This solution was also identified in other fields, such as life sciences, with the “grouping” of the popular Snakemake workflow manager (Mölder et al., 2021) and the HyperQueue plugin (Beránek et al., 2024) of the material sciences workflow manager, Aiida (Huber et al., 2020). Moreover, authors such as Mickelson et al. (2020) have proposed using one of Cylc’s (Oliver et al., 2019) features to submit tasks altogether for those facing long queue times. Furthermore, systems such as Radical-PILOT (Merzky et al., 2022) or Parsl (Babuji et al., 2019) are more sophisticated solutions aimed at High Throughput Computing (HTC) or massively parallel workflows.

Building on these findings and addressing the lack of practical validation, our objective is to measure the impact of task aggregation in real scenarios of climate simulations to determine whether queue times and, therefore, the overall workflow duration is reduced while relating it to the fair share factor.

To do so, we compared the performance between wrapped and unwrapped simulation workflows on three top-tier supercomputers with Slurm across Europe: MareNostrum 4 and the new MareNostrum 5 (Banchelli et al., 2025), hosted at the Barcelona Supercomputing Center (BSC) in Spain; and MeluXina, hosted at LuxProvide in Luxembourg, using the EC-Earth3 model (Döscher et al., 2022) running with the Autosubmit workflow manager.

This work contributes to the Earth sciences field as the first controlled assessment of the impact of task aggregation on the queue times in production climate simulation workflows. Moreover, this solution is transversal to any application executed in congested HPC platforms which have vertical (or chained) job dependencies.

This document is structured into five sections, the present one being Section 1. Section 2 proceeds with the Background, presenting all the concepts needed to understand as they will be used systematically throughout this research, such as task aggregation, Slurm, the EC-Earth3's workflow, Autosubmit, and the performance metrics. Section 3 details the methodology, explaining the experimental procedures and the particular configurations of the simulation experiments for each platform. In Section 4 we compile, analyze, and discuss the results of the previous experiments, and we explain why the fair share is relevant for queue time and how it impacts the performance of task aggregations. Then, in Section 5, we present the conclusions we have drawn from the observed results.

2 Background

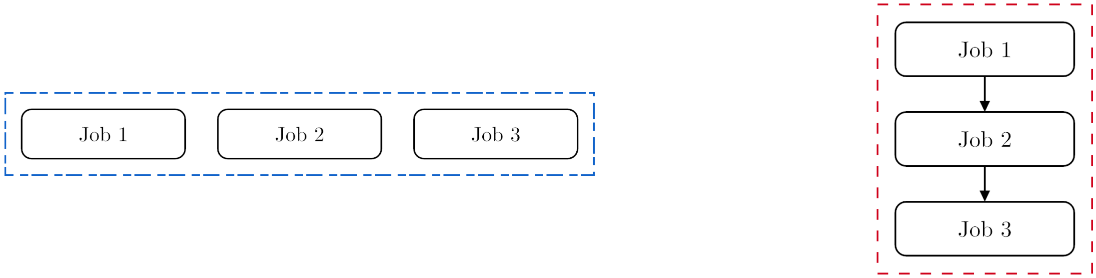
Throughout this section, we present the tools and concepts that are used in this research. This includes a definition for task aggregation, its forms, multiple purposes and particular implementations. Also, we provide a review of Slurm—the workload manager present in all the machines used in this work—and its default scheduling policy. Moreover, we introduce the Autosubmit workflow manager and the EC-Earth3's workflow, alongside the metrics we used to evaluate its performance.

2.1 Task aggregation

In the HPC context, task aggregation consists of combining workflow tasks into a single job. These aggregated tasks are then sent to remote platforms together and are seen as a larger job by the workload manager (i.e. Slurm, PBS). The dependencies among the underlying tasks are respected and neither of them is altered, thus allowing the workflow manager to make decisions on how the tasks should be submitted to be executed by the remote platform. There are different forms of aggregation that can be configured according to the workflow requirements.

The fundamental forms of aggregation are *horizontal* and *vertical*. A horizontal aggregation (Fig. 1a) consists of a group of independent tasks that can be executed concurrently, and a vertical aggregation (Fig. 1b) consists of a series of tasks that depend on each other sequentially.

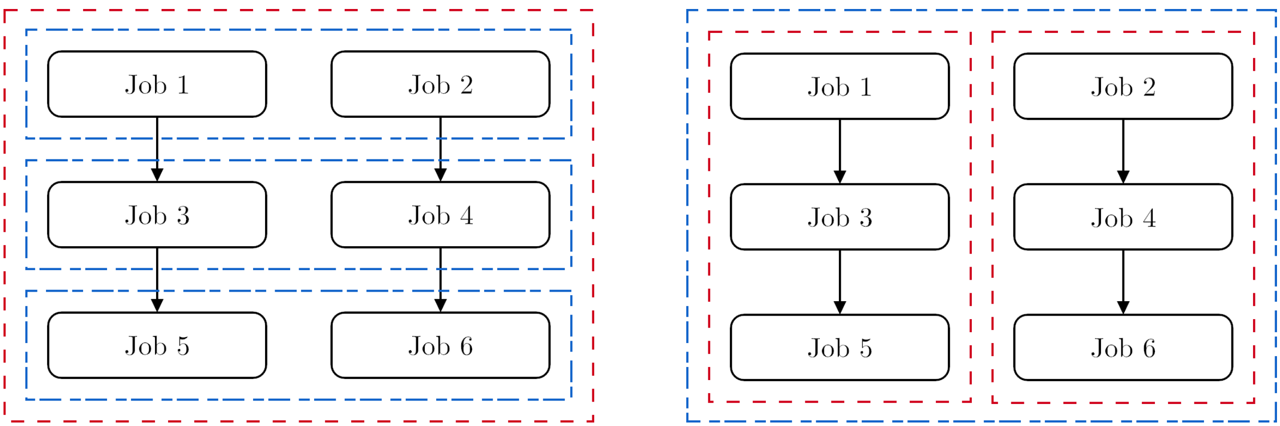
The amount of tasks that can be wrapped is often constrained by queuing policies. For horizontal aggregations, tasks can be added as long as parallel resources are available. In vertical aggregations, however, the limitation arises when the available wallclock is reached. Naturally, task aggregation is a viable strategy only when the allocation size permits bundling multiple tasks.



(a) Horizontal aggregation example. The tasks within the job are independent and can be executed concurrently.

(b) Vertical aggregation example. Every task inside the job relies on the preceding one, except for the initial task.

Figure 1. Fundamental types of aggregation: horizontal (a) and vertical (b).



(a) Horizontal–vertical aggregation, vertically aggregating horizontally aggregated tasks.

(b) Vertical–horizontal aggregation, horizontally aggregating vertically aggregated tasks.

Figure 2. Types of hybrid aggregation arrangements: horizontal–vertical (a) and vertical–horizontal (b).

In addition, two hybrid variations combine the prior possibilities. The first is a *horizontal–vertical* arrangement, characterized by a vertical aggregation that contains several horizontally aggregated tasks (Fig. 2a). Conversely, in the *vertical–horizontal* form, a horizontal aggregation contains several vertically aggregated tasks (Fig. 2b).

Beyond being beneficial for reducing queue times in vertical workflows, as Marciani et al. (2025) explains, aggregation is mainly utilized for two reasons: vertical aggregation prevents repeatedly queuing tasks to a congested machine, thereby making the most out of the responsiveness, while horizontal aggregation also combines tasks into a larger job in order to make efficient use of the parallel resources.

Furthermore, aggregation was also identified elsewhere than in Earth sciences. In particular, Snakemake (Mölder et al., 2021), a well-known workflow manager among life sciences applications, implements the same feature, but is called *grouping*. Moreover, AiiDa (Huber et al., 2020), popular in the field of materials sciences, implements aggregation through the Hyper-

Queue (Beránek et al., 2024) plugin, and the Pegasus workflow manager (Deelman et al., 2015) also implements aggregation with *job clustering*.

In addition, there is a long history of Pilot-Job systems (Turilli et al., 2018) that provide a more sophisticated version of aggregation. These systems also allocate large resources on the platform to be utilized later by multiple workflow tasks. Some authors call this *binding* (Merzky et al., 2019). Moreover, most systems provide enhancements to the scheduling of the tasks executed in the allocation.

Although aggregation share the multi-tenancy of tasks in an allocation aspect of the Pilot-Job systems, they are a simpler implementation: the parallel request is added to get the amount of resources to be requested (e.g. CPUs), and the wallclock of each task in the critical path—the longest path between the beginning of the workflow and any node—is added. Therefore, the underlying logic is simple: parallel launches for the tasks that run in parallel and a loop that keeps track of the tasks with subsequent dependencies.

Since the vertical aggregations were motivated to reduce queue time, we will only explore this form in this paper.

2.2 Slurm Workload Manager

All the HPC platforms used in this study use Slurm (Jette and Wickberg, 2023) as their job scheduler. Slurm is a widely recognized workload manager, known for its open source development, fault tolerance, and extensibility. Almost all leading TOP500 European supercomputers, such as LUMI, Leonardo, or MareNostrum 5, rely on it (Suarez et al., 2025).

2.2.1 Scheduling on Slurm

In order for someone to use a Slurm system, its user has to be assigned to an account by the system administrators. This is done through an *association*. An account is a set of users or accounts, normally created on a per-project basis. This structure creates a tree, with the root account at the top-most level and users as the leaves.

In a lack of resources environment, Slurm queues jobs ordered by their *priority* integer. By default, the priority is computed with the *multifactor* algorithm (SchedMD, 2025b). Slurm schedules jobs in strict priority order, with the exception of the *backfill* algorithm, which is used to increase the throughput of the machine by allowing small jobs to cut the queue if no job with higher priority is delayed. Thus, the bigger the job in terms of wallclock or parallel resources, the less likely it is to cut the queue. Therefore, the backfill algorithm is more unlikely to find a spot for the aggregated tasks, thus increasing the queue time because the aggregation could not take advantage of it.

2.2.2 Multifactor scheduling

The multifactor algorithm computes the priority of the job from multiple properties of the submission, such as the time spent in the queue, called *age*, the *fair share*, the *size*, or the *quality of service (QoS)*, among other specifications for the submission of jobs. Each of the properties is quantified by a floating-point ranging from zero to one called “factor”, and each factor is then associated to a weight. Then, the priority of the job is computed as the sum of these factors multiplied by their corresponding weights. System administrators control how much each property influences the final priority by altering the weights associated with the factors. In our experience, the QoS factor is commonly given the highest weight, followed by the age and the fair share, and, finally, the size.

By default, Slurm computes the size factor as the ratio of CPUs requested to the total available in the system. This means that larger submissions are prioritized by the scheduler. In order to compensate this behavior, system administrators usually utilize the QoS factor to benefit smaller jobs.

System administrators configure multiple QoS, which apply restrictions on the size of the jobs in terms of computational resources requested and maximum runtime, and associate a *QoS-priority* value. The factor is then computed as the ratio between each of the QoS-priorities and the maximum QoS-priority.

In order to prioritize those jobs that have been waiting for longer, Slurm quantifies the time spent in queue by the job through the age factor. This factor grows linearly from zero and reaches one when the job spends a time in the queue specified by the system administrator.

Finally, the fair share factor is the quantification of the user's right to the machine depending on its utilization. Its main purpose is to balance the responsiveness of the machine among users, preventing high-consuming users from starving the underserved ones. Slurm computes fair share with the *Fair Tree* algorithm by default.

2.2.3 The Fair Tree algorithm

We explain an equivalent approach of the Fair Tree algorithm (SchedMD, 2025a) to help illustrate how the utilization of the account impacts individual users.

The Fair Tree algorithm can be thought of as ordering the users according to their utilization of the computational resources while also accounting their account's. The algorithm starts with an empty array, which will store the sorted users, and traverses depth-first the user and account tree starting from the root account. At each step, it sorts the children (either account or user) by the *Level Fair Share*, which is the ratio of the *account-priority* assigned during the account creation with respect to their *utilization*—namely raw usage. If the child is a user, it is added to the sorted array. If it is an account, the algorithm recurses. Once all the users in the tree have been evaluated, they are assigned a fair share, which is the division of their position in the sorted array with respect to the total number of users in the machine.

In Figure 3, we depict an execution of the Fair Tree algorithm. The algorithm starts at the `root` account and computes the *Level Fair Share* (LF) of each of the children, which are `acc 1`, `acc 2` and `user 2`. Because the lowest LF belongs to an account, the algorithm recurses. The `acc 2` has two users, `user 3` and `user 1`, with LF 1 and 7, respectively. Therefore, `user 3` is the first user to be added to the sorted array of users, and then `user 1`. Since there are no other children, the algorithm moves to `acc 1`. This account has only one user that is added to the sorted array. Finally, the algorithm returns to `user 2`, which is the last user to be added to the array. The fair share of `user 3` will be 0.25 because it is at the first position of the array and there are four users. The same logic applies to the remaining users.

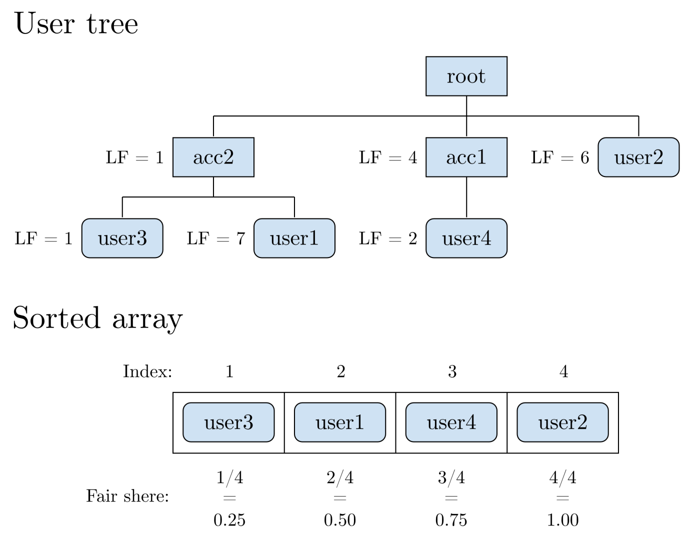


Figure 3. Tree structure of users in Slurm with their corresponding *Level Fair Share* and the associated sorted array of users.

Due to the depth-first traversal, the Fair Tree algorithm makes all the users under an account responsible for their fair share. For example, if a single user uses heavily the machine, it will negatively impact the account level fair share, and therefore the fair share of all of its peers.

This shared nature of the fair share factor inspires task aggregation. The idea is to make the most out of the resources granted because every job, regardless of its request in computing resources, will have its priority impacted by the usage of the group.

2.3 Autosubmit Workflow Manager

HPC applications may require the orchestration of several tasks with complex interdependencies that must be respected when dispatching them to the remote platforms. Manually submitting every single task of a workflow, taking into account their dependencies and dealing with possible errors during execution, is usually not viable without an automated processing tool, known as a *workflow manager*.

Autosubmit (Manubens-Gil et al., 2016) is a workflow and experiment manager specifically tailored for climate and air quality HPC applications. It is responsible for recording the experiments, their configuration, the execution of the workflow, handling failures, and tracking provenance. As climate and air-quality experiments have hundreds of configurations and produce large amounts of data, the role of the experiment manager is to organize the execution of the simulations by storing configurations and logs in a FAIR way (Puiggros et al., 2025). Additionally, this ensures reproducibility, since rerunning a simulation is a matter of copying an experiment.

Autosubmit is the only workflow manager in Earth sciences that implements task aggregation. It is done through the "wrappers", which are sent to the remote platforms following the policy indicated by the user. The wrapping policies in Autosubmit are *flexible*, *strict*, and *mixed*. The flexible policy is a best-effort policy. This means that if aggregating the required minimum of tasks is not possible, they are submitted to the platform individually. The strict policy will only send the minimum number of wrapped tasks to the remote platform. The mixed one will wait for a minimum number of tasks to create the wrapper, except in the cases where a job has failed.

2.4 Auto-EC-Earth3

The workflow implementation of EC-Earth3 in Autosubmit, also called Auto-EC-Earth3 (Ferrer et al., 2025), is a complete end-to-end workflow which includes the compilation of the various binaries, synchronization of the initial conditions, simulation, post-processing of results, diagnostics, and fault tolerance. Auto-EC-Earth3 leverages each of the model's restart files, which are generated on a per-chunk basis by all the coupled components to recover the simulation in case of failures.

2.5 Performance statistics of climate models

The Earth sciences community developed a set of metrics aiming to compare the performance of their models. These were introduced in the work of Balaji et al. (2017) and quickly became the norm.

Out of all the metrics that the authors introduce, we highlight the SYPD and the ASYPD. The first represents the total simulated time of the climate model, in years, divided by the runtime of the ESM in days. The latter is similar, but added the time spent in queue and because of failures.

3 Methods

In this section, we outline the steps to set up the workflow on each supercomputing platform, the wrapping configurations, the simulation configuration, and the process to fetch the meaningful Slurm factors and timestamps of execution.

3.1 Experimental overview

The main objective of this paper is to quantify the impact of task aggregation in production simulations on different HPC platforms. We did this by executing pairs of equal workflows at the same time using two different users under the same account in the CPU partitions of MareNostrum 4, MareNostrum 5, and MeluXina, running with Autosubmit, with the aim of having the same usage on both users so that they had the same initial fair share, and to ensure both executions find the same workload conditions. These systems have 3,456, 6,192, and 573 nodes, respectively. One of the workflows used wrappers and the other did not, serving as a reference.

The workflows we ran were based on testing configurations of Auto-EC-Earth3 (Garcia Lopez et al., 2025), which are configurable and scalable enough to allow for long simulations, enabling us to cover the daily cycle of utilization of the HPC platforms, encountering multiple workload conditions and possible congestion scenarios. We varied the coupled components

Table 1. Specific simulation parameters for each supercomputing platform.

	Chunks				Wrappers		Other information
Platform	Quantity	Size (months)	Nodes	Wallclock (hours)	Quantity	Size (chunks)	Additional coupled models
MareNostrum 4	50	12	11	3:30	5*	10*	PISCES-LPJG-TM5
MareNostrum 5	40	12	7	2:30	4	10	–
MeluXina	50	12	10	1:30	5	10	–

[*] Although the experiment was configured to create wrappers of 10 tasks, the flexible wrapping policy of the workflow manager decided, at runtime, to distribute them among four wrappers of ten tasks, one of eight, and two loose tasks.

according to their availability on the respective platforms, as well as the computational resources available for each. Since we were not interested in the output of the models, but rather in analyzing the behavior of the scheduler facing the large computationally intensive tasks of the workflow, we removed all the post-processing. Therefore, the workflow was pruned to leave only the tasks required to setup and run the simulation.

3.2 Specific simulation configurations on each platform

Table 1 summarizes the particular configurations we chose for each platform. We varied the number of years simulated, as well as the configuration of coupled models that will be simulated, depending on the available budget for our study on the different HPC platforms. All simulations ran IFS (Buizza et al., 2018) with grid *T255L91* and NEMO (Madec and the NEMO System Team, 2022) with grid *ORCA1L75*. The LIM3 model (Rousset et al., 2015) was enabled on all supercomputers. In MareNostrum 4 we executed the fully-coupled mode, having the PISCES (Aumont et al., 2015), LPJ-GUESS (Smith et al., 2014), and TM5 (van Noije et al., 2014) models enabled.

Taking into account the substantial differences between workflows, we stress that the analysis in this work aims to evaluate task aggregation for each HPC platform individually, without assessing cross-platform performance comparisons.

The enabled models as well as the number of chunks, which impact the length of the simulation in terms of simulated time frames under the same initial conditions, depended on the availability of resources on each platform. We simulated 40 years in MareNostrum 5, and 50 years in MareNostrum 4 and MeluXina. These values proved to be adequate because they spanned multiple days to finish the simulation and, therefore, capturing the daily cycle of usage.

Regarding the wrapping strategy, we decided to create wrappers of 10 tasks (i.e. 10 chunks) following the flexible policy of Autosubmit in all the simulations. As for the size of the wrappers, we considered 10 tasks each, a typical configuration that still fits within the restrictions of the platforms.

The experiment configuration presented in Table 1 was set up in accordance with the instructions in the Autosubmit documentation (Autosubmit Team, 2023). Appendix A contains the detailed steps for the proper configuration of the wrapped workflow with wrappers for the MareNostrum 5 platform as an example.

Table 2. Queue time statistics and performance metrics (SYPD, ASYPD) for wrapped and unwrapped simulations by platform.

	MareNostrum 4		MareNostrum 5		MeluXina	
Statistic	Wrapped	Unwrapped	Wrapped	Unwrapped	Wrapped	Unwrapped
Queue time statistics						
Number of jobs	7	50	4	40	5	50
Mean (s)	2,224.86	3,479.46	35.25	39.95	16.20	19.42
Std (s)	3,178.46	9,474.73	22.74	22.39	10.96	10.98
Min (s)	0.00	0.00	18.00	3.00	4.00	0.00
Median (s)	38.00	123.00	27.5	39.00	14.00	19.5
Max (s)	8,344.00	49,595	68.00	85.00	34.00	58.00
Total (s)	15,574.00	173,973	141.00	1,598.00	81.00	999.00
Performance metrics						
SYPD	7.6	7.8	14.3	14.3	19.9	21.1
ASYPD	7.4	5.7	14.3	14.2	19.9	20.8

3.3 Collecting metrics

To measure the variability in queue times, we collected from Autosubmit the time spent in queue and the execution time for each job. Conversely, to monitor the status of the platforms during the execution of the workflow and understand the causes of such variability, we gathered some Slurm parameters, including *fair share* and *raw usage*, by appending a custom script to the beginning of each task (Goitia and Giménez de Castro, 2025).

4 Results and discussion

In this section, we present and discuss the results obtained from our experiments with the Auto-EC-Earth3 workflow, analyzing the total queue times, times-to-solution, SYPD and ASYPD, and the impact of the user's usage on the fair share of the simulations executed with and without wrappers on MareNostrum 4, MareNostrum 5, and MeluXina. We also examine common trends across all platforms and the limitations identified in the experiment.

4.1 Statistics, performance metrics, and machine utilization of the simulations

Table 2 presents the summary statistics for the queue times in seconds and the performance metrics of the jobs executed on each platform, divided into those that were wrapped and those that did not.

In the case of **MareNostrum 4**, we see that both simulations, the reference and the wrapped one, had about the same SYPD metric, with a small difference that we attribute to system fluctuations. This is expected because they are running the same workload under the same machine conditions.

However, the sum of the queue time for all the simulation jobs of the experiment using wrappers was 15,574 seconds, or 4.33 hours, while the reference experiment, which did not use wrappers, had a total queue time of 173,973 seconds, or 2.01 days. This means that the queue time in the experiment without wrappers is 11.17 times greater than in the experiment in which we used them. This also translates to an ASYPD of 7.4 for the wrapped experiment and 5.7 for the reference.

We observe a large difference between the maximum and minimum of the time spent in the queue. Both wrapped and unwrapped simulations had jobs that started immediately and others that had to wait for hours. In the unwrapped case, we see one job that was 49,595 seconds, or 13.78 hours, in queue. The larger standard deviation of the unwrapped jobs also indicates that the distribution of the queue time of the unwrapped jobs is more right heavy tailed than the wrapped one.

Figure 4 illustrates the evolution of the fair share and the usage of both our users, the one running the unwrapped simulation and the other running the wrapped one. We see that both fair shares remained constant, except for two sudden increases, even though our users were increasingly billed with more and more usage. This means that our simulation's resources were not a match to our peers', so the Fair Tree algorithm was not penalizing us, as our users' priority—given by the fair share—is dominated by the usage of the other users of the account.

We attribute the skewness of the queue time distribution in both cases to the heavy usage of the platform. But it is particularly acute for the unwrapped workflow, given that there are more jobs being queued under the same congested scenario and fair share.

As a consequence of the reduction in the queue time and long queue times relative to the simulation runtime, we saw a significant positive impact on the total execution time of our experiments. We observe in MareNostrum 4 that the difference between the SYPD and ASYPD of the wrapped experiment is of 0.2 whereas in the unwrapped case it is 2.1. This equates to a relative difference of 2.6% and 26.9%, respectively. The latter value is in line with the overhead observed by Acosta et al. (2024).

MareNostrum 5 experiments had SYPD values that were about the same as in the case of MareNostrum 4.

As for the total queue times for the wrapped and unwrapped experiments, we see that they were 141 and 1,598 seconds, respectively. This means that the reduction in the time spent in the queue of the experiment with wrappers compared to the reference experiment is a factor of 11.33.

The amplitude of the observed queue time values is much smaller compared to MareNostrum 4, with maximums of tens of seconds. This was expected because of the novelty of the machine at the time the experiments were performed and therefore there was not much congestion.

In Figure 5, we can see how the utilization of our users impacts the fair share. Compared to MareNostrum 4, in MareNostrum 5 the factor generally decreases, except for one instance when another account temporarily utilized more of the machine than we did.

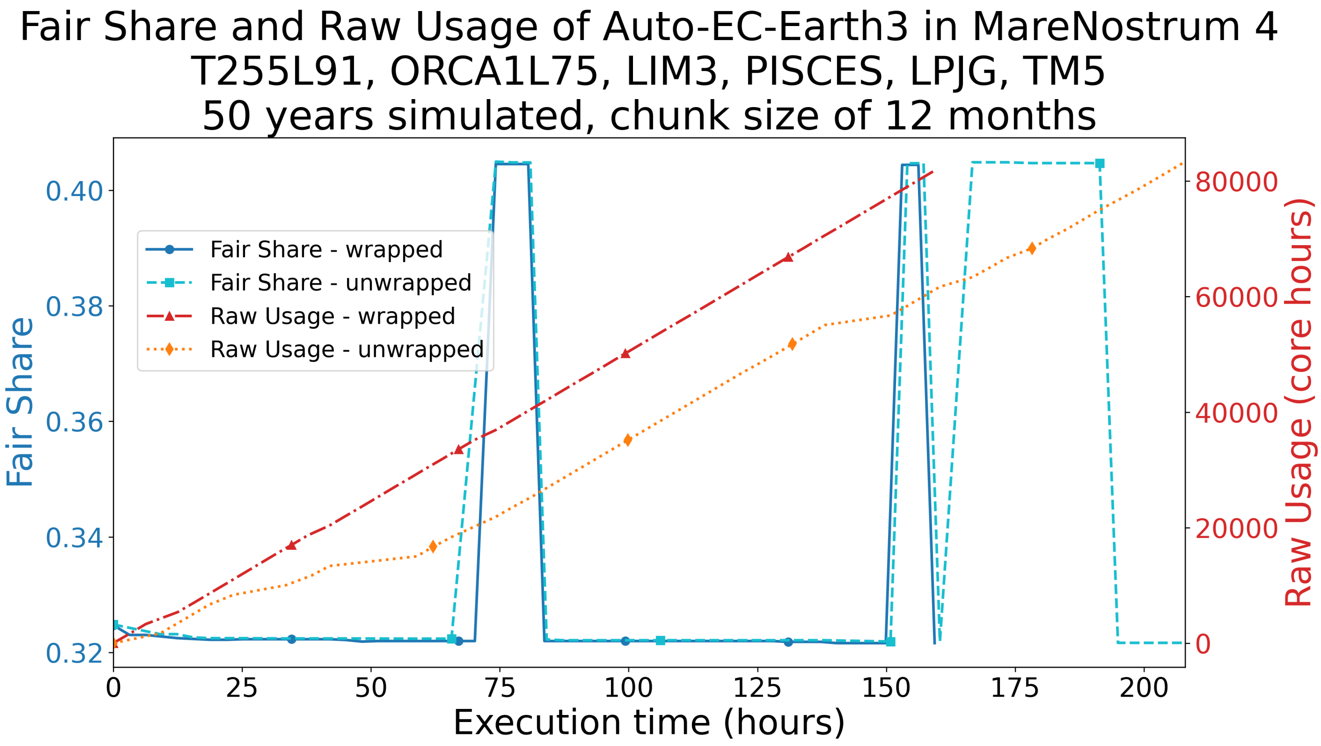


Figure 4. Fair Share and Raw Usage of Auto-EC-Earth3 with the T255L91-ORCA1L75-LIM3-PISCES-LPJG-TM5 configuration. 50 years simulated with a chunk size of 12 months. Results for MareNostrum 4.

As for the time-to-solution, we have that the total time was of 2.80 days for the wrapped workflow and 2.81 days for the reference one. This means that the time in the queue was small relative to the runtime of the simulation. This is also reflected in the low spread of 0.1 between SYPD and ASYPD in the reference simulation.

We find **MeluXina**'s queue times similar to MareNostrum 5. The aggregated queue times for the experiments with and without wrappers executed in MeluXina are 81 and 999 seconds, respectively. This corresponds to a reduction of 12.33 times in the total queue time.

In Table 2, we find that every statistic, with the exception of the minimum and standard deviation, is smaller in the wrapped case, although in both cases it does not exceed a hundred seconds.

In the evolution of fair share and utilization of our users in Fig. 6, we observe how our utilization is inversely related to the fair share factor.

The time-to-solution was 2.51 days for the wrapped experiment, compared to 2.38 days for the reference one. As in MareNostrum 5, the relative time in queue with respect to the runtime of the simulation was small, resulting in only minor improvements in the time-to-solution. This is also observed in the narrow difference between SYPD and ASYPD of the reference simulation.

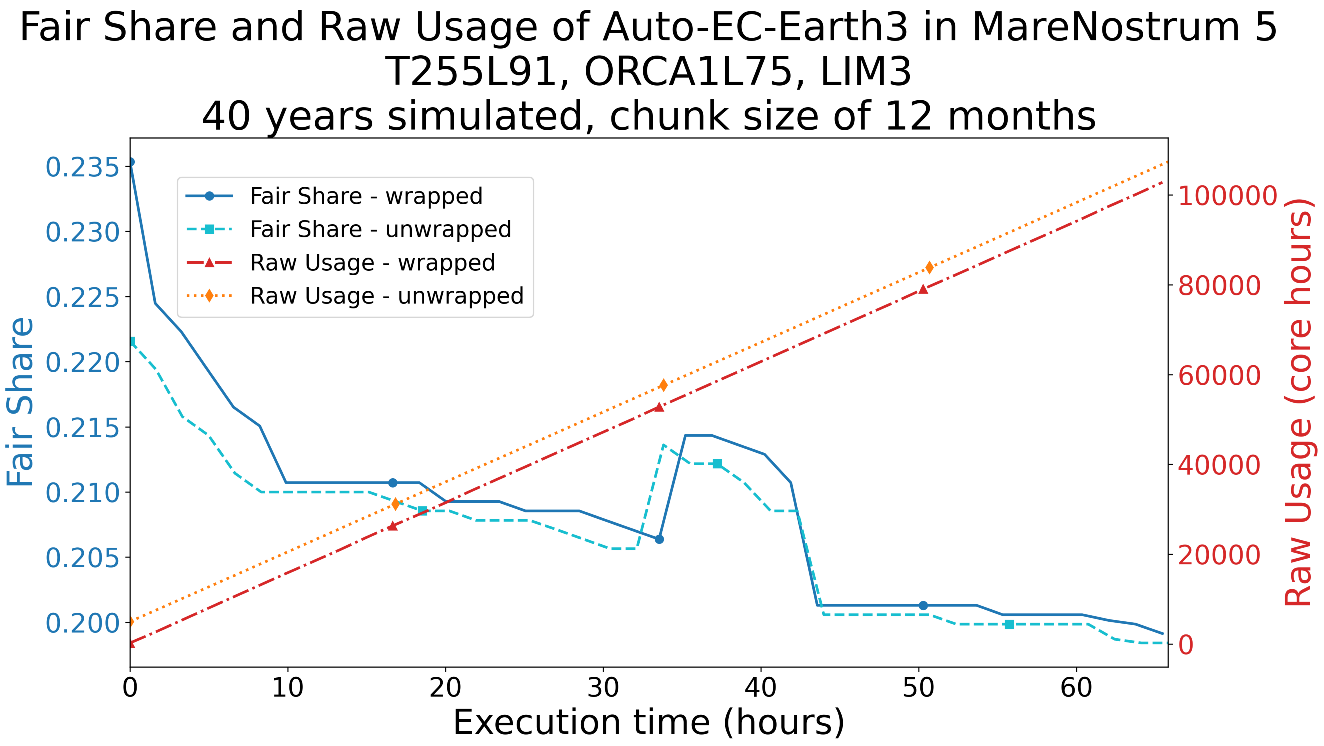


Figure 5. Fair Share and Raw Usage of Auto-EC-Earth3 with T255L91-ORCA1L75-LIM3. Simulating 40 years with a chunk size of 12 months. Results for MareNostrum 5.

4.2 Common trends to all platforms

Across all platforms, we see that despite the larger submissions, we did not see any significant impact of the backfill algorithm. Wrappers consistently reduce the total time spent in the queue.

This is attributed to two factors: first, there are fewer jobs in the wrapped scenario, and second, wrapped jobs typically spend less time in the queue, both on average and median, compared to unwrapped ones. This is surprising, and we attribute it to an opportune status of the machine for these few tasks of the wrapped case—five or seven—in contrast to those 50 of the unwrapped case.

Furthermore, it was observed that the impact of the use of wrappers on the time-to-solution varied depending on our group's utilization of the machine. When comparing MareNostrum 4 and MareNostrum 5, which had the same scheduling configurations, we see that task aggregation had a greater impact in the first. From the group utilization we can grossly estimate the machine utilization by observing how our usage impacts in the fair share.

This site variability was also observed by Acosta et al. (2024), where the authors found that some platforms had negligible queue times, while others reached 20% of the total simulation runtime.

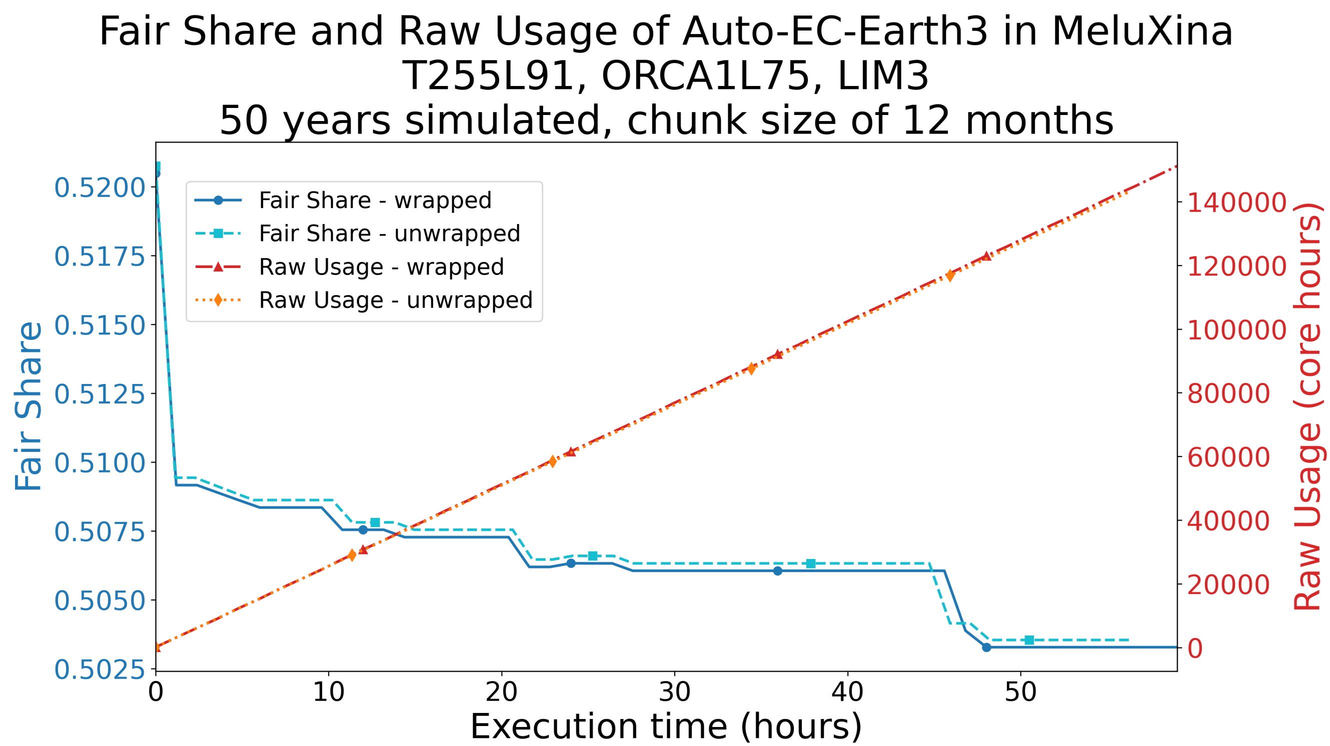


Figure 6. Fair Share and Raw Usage of Auto-EC-Earth3 T255L91-ORCA1L75-LIM3. Simulating 50 years with a chunk size of 12 months. Results for MeluXina.

4.3 Limitations

Even though this work is the first to test the task aggregation technique by utilizing over half a million core hours on three major flagship machines, we have seen some limitations that are important to point out.

We aimed to have the same usage recorded on both users running the simulations, one with wrappers and the other without, since it fundamentally impacts the fair share. Due to the scheduling policy of MareNostrum 5 at the time, where the machine did not periodically clear the usage of the users, this was impossible to do. However, after analyzing the results, we found that the impact of this difference was not significant in the queue times because the machine was not congested.

Moreover, in this work we carried out twin simulations, to great expenditure, but a more systematic study on larger simulations, such as those in CMIP6, would be beneficial. This would require adopting a different methodology from the one used in this paper, given the high computational cost of such simulations.

Our tests were carried out only on machines with the Fair Tree algorithm, not covering any other platform that does not use it. Whether it would be a different Slurm configuration or a totally different scheduler. For the former, the Fair Tree is the default algorithm, and we are not aware of any Slurm-based HPC platform that does not use it. For the latter, we are not aware of any European platform that does not use Slurm. Outside Europe, some Japanese HPC systems utilize Fujitsu's Technical Computing Suite (TCS) (Fujitsu, 2021), notably the Fugaku flagship system, and some American systems, such as El Capitan at Lawrence Livermore National Laboratory, use the Flux scheduler (Ahn et al., 2020).

Because we have focused on the most compute intensive part, we stripped the original Auto-EC-Earth3 workflow of other tasks than the simulation that were not relevant for its execution. Therefore, we removed the tasks for postprocessing the results, saving the initial conditions, diagnostics, data transferring, and cleanup. All of these tasks are necessary parts for the execution of the model, and are subject to the queue as much as the ESM itself. Therefore, we should analyze how much these smaller tasks interfere with the ASYPD of the simulation and if aggregating them improves the time-to-solution.

5 Conclusions

In this study, we bring to the forefront the discussion of the long queue times of climate simulation workflows executed on congested HPC platforms. This is a transversal issue with a notable impact on the productivity of the Earth sciences community that the few authors studying the performance of these simulations have estimated between 10% and 20% of the runtime of the most important simulations around the world for climate change studies. Moreover, this issue has also been identified in other fields where shared HPC platforms are used.

In order to mitigate the time that lengthy simulations spend in queue, we study a technique to reduce the total number of submissions of the workflow tasks to be executed in the HPC platforms, grouping them into a single one while respecting their dependencies and the logic of the underlying software. This technique allows to separate the workflow description from the execution, giving freedom to the workflow manager and the user to decide how and when to group tasks. We refer to this approach as *task aggregation*.

Studies have been made on the effectiveness of this technique in simulated environments, and this work provides an answer to the natural question of whether such positive results would also be transferred to real environments, performing the first assessment on the impact of task aggregation on the queue times of real large-scale climate simulation workflows, running on renowned supercomputing platforms across Europe: MareNostrum 4, MareNostrum 5, and MeluXina.

Our experimental results show that *applying task aggregation decreases total queue times by 11.17 to 12.33 times compared to a workflow that does not*. This finding is positive across all three platforms that we tested, and the reduction is strongly related to the size of the aggregation.

We observed that single jobs stood about the same time, or less, in queue as their longer grouped counterparts despite the backfill algorithm. This means that the reduction comes mainly from having fewer jobs in queue. These findings were observed in a variety of model configurations and platforms with very different levels of utilization.

Machine occupancy has been found to influence the total time workflows spend in queues, resulting in a variable degree of reduction in both queue time and overall workflow execution time. We estimate it from the fair share factor and from the absolute time in queue. We observed in the highly congested MareNostrum 4 that the factor's dynamic was unrelated to our utilization, although we used over 80,000 core hours—or 1,500 node hours. On the other hand, we saw in MeluXina how our utilization was inversely correlated with the fair share.

In light of the results, we encourage Earth sciences and other communities to apply task aggregation from the workflow side in situations of congestion, as we expect it to be beneficial in long workflow executions with a vertical pattern that are executed on shared HPC platforms, since the scheduler is agnostic regarding the internal operation of the jobs. Furthermore, we have evidence that the longer the simulations, the more noticeable the positive impact is. Our solution is preferable to manually increasing the workload of each job in order to preserve flexibility, including task-level checkpoints, provenance, selective retries, and adaptation to platform constraints by using the capabilities of the workflow managers.

Code and data availability. All the Autosubmit workflows that we have created and executed have been archived on WorkflowHub (Gustafsson et al., 2025). Wrapped and unwrapped workflows, respectively, can be found in (https://doi.org/10.48546/workflowhub.workflow.2067.1, Goitia et al., 2025c) and (https://doi.org/10.48546/workflowhub.workflow.2066.1, Goitia et al., 2025d) for MareNostrum 4, (https://doi.org/10.48546/workflowhub.workflow.2065.1, Goitia et al., 2025e) and (https://doi.org/10.48546/workflowhub.workflow.2064.1, Goitia et al., 2025f) for MareNostrum 5, and (https://doi.org/10.48546/workflowhub.workflow.2063.1, Goitia et al., 2025a) and (https://doi.org/10.48546/workflowhub.workflow.2062.1, Goitia et al., 2025b) for MeluXina.

The scripts used to collect machine utilization data while executing the workflows have also been archived on Zenodo (https://doi.org/10.5281/zenodo.15673462, Goitia and Giménez de Castro, 2025).

The datasets with machine utilization records and statistics from the executions of the EC-Earth3 workflow on MareNostrum 4, MareNostrum 5, and MeluXina, are archived on Zenodo (https://doi.org/10.5281/zenodo.15292084, Goitia, 2025).

Appendix A: Configuring an Auto-EC-Earth3 experiment in MareNostrum 5

The experimentation stage of this research consisted of running two experiments simultaneously, one with the wrappers enabled and the other without. In this section we provide a guide to properly set up the wrapped experiment in the MareNostrum 5 platform. All workflows used throughout this study can be accessed in WorkflowHub. To do so, please check section "Code and data availability".

A1 Experiment overview

The experiment was based on a test case which operated with a T255L91-ORCA1L75-LIM3 configuration. According to Table 1, the workflow was configured with 40 chunks, each simulating 12 months-each, organized into 4 wrappers of 10 jobs.

In order to apply these modifications and configure the wrappers, the configuration files were edited as Table A1 indicates. This configuration files contain structured information about the workflow, in `INI` format. The experiment identifier was `a7d4`.

The files mentioned in the table, with all configurations already set, can be found in the `conf` directories of the experiments.

Table A1. Changes applied to the configuration files of the `a7d4` (wrapped) experiment in MareNostrum 5.

Configuration file	**Modifications**
autosubmit_a7d4.conf	Added the wrapping policy in Fig. A1. 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, as the Fig. A2 indicates.
jobs_a7d4.conf	Deleted unnecessary steps in the workflow and cleaned their dependencies. The `EXTENDED_HEADER` key for the `SIM` tasks was filled with the path to the header script that collected the Slurm *share* metrics described in the methodology.
platforms_a7d4.conf	Placed the platform details, such as the *hostname*, *username*, *scheduler*, and *account*.
proj_a7d4.conf	Disabled several flags associated with the jobs previously deleted.

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

Figure A1. The wrapping configuration for the wrapped workflow in MareNostrum 5.

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

Figure A2. The experiment configuration for the wrapped workflow in MareNostrum 5.

Author contributions. PG has developed the methodology, carried out the experiments, gathered and analyzed the data, and wrote the manuscript. MGM supervised the work, conceptualized the experiments, and reviewed the manuscript. MC conceptualized the experiments and reviewed the manuscript. MCA reviewed the manuscript.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. Manuel G. Marciani is supported by grant CEX2021-001148-S-20-5 funded by MICIU/AEI/10.13039/501100011033 and by FSE+. Mario Acosta is supported from the Spanish National Research Council through OEMES (PID2020-116324RA-I00).

The authors declare they used AI in order to improve readability of the manuscript. After that, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Acosta, M. C., Palomas, S., Paronuzzi Ticco, S. V., Utrera, G., Biercamp, J., Bretonniere, P.-A., Budich, R., Castrillo, M., Caubel, A., Doblas-Reyes, F., Epicoco, I., Fladrich, U., Joussaume, S., Kumar Gupta, A., Lawrence, B., Le Sager, P., Lister, G., Moine, M.-P., Rioual, J.-C., Valcke, S., Zadeh, N., and Balaji, V.: The computational and energy cost of simulation and storage for climate science: lessons from CMIP6, Geoscientific Model Development, 17, 3081–3098, https://doi.org/10.5194/gmd-17-3081-2024, 2024.

Ahn, D. H., Bass, N., Chu, A., Garlick, J., Grondona, M., Herbein, S., Ingólfsson, H. I., Koning, J., Patki, T., Scogland, T. R., Springmeyer, B., and Taufer, M.: Flux: Overcoming scheduling challenges for exascale workflows, Future Generation Computer Systems, 110, 202–213, https://doi.org/https://doi.org/10.1016/j.future.2020.04.006, 2020.

Aumont, O., Ethé, C., Tagliabue, A., Bopp, L., and Gehlen, M.: PISCES-v2: an ocean biogeochemical model for carbon and ecosystem studies, Geoscientific Model Development, 8, 2465–2513, https://doi.org/10.5194/gmd-8-2465-2015, 2015.

Autosubmit Team: Autosubmit v3.15 Documentation, https://autosubmit.readthedocs.io/en/3.15.0-branch/, last accessed: 2026-08-24, 2023.

Babuji, Y., Woodard, A., Li, Z., Katz, D. S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J. M., Foster, I., Wilde, M., and Chard, K.: Parsl: Pervasive Parallel Programming in Python, in: HPDC '19, p. 25–36, Association for Computing Machinery, New York, NY, USA, ISBN 9781450366700, https://doi.org/10.1145/3307681.3325400, 2019.

Balaji, V., Maisonnave, E., Zadeh, N., Lawrence, B. N., Biercamp, J., Fladrich, U., Aloisio, G., Benson, R., Caubel, A., Durachta, J., Foujols, M.-A., Lister, G., Mocavero, S., Underwood, S., and Wright, G.: CPMIP: measurements of real computational performance of Earth system models in CMIP6, Geoscientific Model Development, 10, 19–34, https://doi.org/10.5194/gmd-10-19-2017, 2017.

Banchelli, F., Garcia-Gasulla, M., Mantovani, F., Vinyals, J., Pocurull, J., Vicente, D., Eguzkitza, B., Galeazzo, F. C. C., Acosta, M. C., and Girona, S.: Introducing MareNostrum5: A European pre-exascale energy-efficient system designed to serve a broad spectrum of scientific workloads, https://arxiv.org/abs/2503.09917, 2025.

Beránek, J., Böhm, A., Palermo, G., Martinovič, J., and Jansík, B.: HyperQueue: Efficient and ergonomic task graphs on HPC clusters, SoftwareX, 27, 101 814, https://doi.org/10.1016/j.softx.2024.101814, 2024.

Buizza, R., Alonso-Balmaseda, M., Brown, A., English, S., Forbes, R., Geer, A., Haiden, T., Leutbecher, M., Magnusson, L., Rodwell, M., Sleigh, M., Stockdale, T., Vitart, F., and Wedi, N.: The development and evaluation process followed at ECMWF to upgrade the Integrated Forecasting System (IFS), European Centre for Medium Range Weather Forecasts, https://doi.org/10.21957/xzopnhty9, 2018.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., and Wenger, K.: Pegasus, a workflow management system for science automation, Future Generation Computer Systems, 46, 17–35, https://doi.org/https://doi.org/10.1016/j.future.2014.10.008, 2015.

Döscher, R., Acosta, M., Alessandri, A., Anthoni, P., Arsouze, T., Bergman, T., Bernardello, R., Boussetta, S., Caron, L. P., Carver, G., Castrillo, M., Catalano, F., Cvijanovic, I., Davini, P., Dekker, E., Doblas-Reyes, F. J., Docquier, D., Echevarria, P., Fladrich, U., Fuentes-Franco, R., Gröger, M., Hardenberg, J. V., Hieronymus, J., Karami, M. P., Keskinen, J. P., Koenigk, T., Makkonen, R., Massonnet, F., Ménégoz, M., Miller, P. A., Moreno-Chamarro, E., Nieradzik, L., Van Noije, T., Nolan, P., O'donnell, D., Ollinaho, P., Van Den Oord, G., Ortega, P., Prims, O. T., Ramos, A., Reerink, T., Rousset, C., Ruprich-Robert, Y., Le Sager, P., Schmith, T., Schrödner, R., Serva, F., Sicardi, V., Sloth Madsen, M., Smith, B., Tian, T., Tourigny, E., Uotila, P., Vancoppenolle, M., Wang, S., Wårlind, D., Willén, U., Wyser, K., Yang, S., Yepes-Arbós, X., and Zhang, Q.: The EC-Earth3 Earth system model for the Coupled Model Intercomparison Project 6, Geoscientific Model Development, 15, 2973–3020, https://doi.org/10.5194/gmd-15-2973-2022, 2022.

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geoscientific Model Development, 9, 1937–1958, https://doi.org/10.5194/gmd-9-1937-2016, 2016.

Ferrer, E., Montane, G., Castrillo, M., and Garcia, A.: Auto-EC-Earth: An automatic workflow to manage climate modelling experiments using Autosubmit, in: EGU General Assembly 2025, Vienna, Austria, https://doi.org/10.5194/egusphere-egu25-8621, 2025.

Fujitsu: FUJITSU Software Technical Computing Suite V4.0L20, Fujitsu, available at https://software.fujitsu.com/jp/manual/manualindex/p21000155e.html, 2021.

Garcia Lopez, A., Arriola Meikle, L., Montane Pinto, G., Castrillo, M., de Paula Kinoshita, B., Ferrer Escuin, E., and Gaya Avila, A.: Enabling reliable workflow development with an advanced Testing Suite, in: EGU General Assembly 2025, Vienna, Austria, EGU25-8305, 2025.

Goitia, P.: Machine utilization and statistics of EC-Earth3 workflows running on MareNostrum 4, MareNostrum 5, and MeluXina, https://doi.org/10.5281/zenodo.15292084, 2025.

Goitia, P. and Giménez de Castro, M.: Header scripts to gather statistics and usage records of EC-Earth3 workflows from MareNostrum 4, MareNostrum 5, and Meluxina, https://doi.org/10.5281/zenodo.15673462, 2025.

Goitia, P., Ferrer, E., Garcia, A., Bonet, G., Montane, G., and Castrillo, M.: EC-Earth3 workflow with wrappers in MeluXina, WorkflowHub, https://doi.org/10.48546/workflowhub.workflow.2063.1, 2025a.

Goitia, P., Ferrer, E., Garcia, A., Bonet, G., Montane, G., and Castrillo, M.: EC-Earth3 workflow without wrappers in MeluXina, WorkflowHub, https://doi.org/10.48546/workflowhub.workflow.2062.1, 2025b.

Goitia, P., Ferrer, E., Garcia, A., Bonet, G., Montane, G., and Castrillo, M.: EC-Earth3 workflow with wrappers in MareNostrum 4, WorkflowHub, https://doi.org/10.48546/workflowhub.workflow.2067.1, 2025c.

Goitia, P., Ferrer, E., Garcia, A., Bonet, G., Montane, G., and Castrillo, M.: EC-Earth3 workflow without wrappers in MareNostrum 4, WorkflowHub, https://doi.org/10.48546/workflowhub.workflow.2066.1, 2025d.

Goitia, P., Ferrer, E., Garcia, A., Bonet, G., Montane, G., and Castrillo, M.: EC-Earth3 workflow with wrappers in MareNostrum 5, WorkflowHub, https://doi.org/10.48546/workflowhub.workflow.2065.1, 2025e.

Goitia, P., Ferrer, E., Garcia, A., Bonet, G., Montane, G., and Castrillo, M.: EC-Earth3 workflow without wrappers in MareNostrum 5, WorkflowHub, https://doi.org/10.48546/workflowhub.workflow.2064.1, 2025f.

Gustafsson, O. J. R., Wilkinson, S. R., Bacall, F., Soiland-Reyes, S., Leo, S., Pireddu, L., Owen, S., Juty, N., Fernández, J. M., Brown, T., Ménager, H., Grüning, B., Capella-Gutierrez, S., Coppens, F., and Goble, C.: WorkflowHub: a registry for computational workflows, Scientific Data, 12, 837, https://doi.org/10.1038/s41597-025-04786-3, 2025.

Hoffmann, J., Bauer, P., Sandu, I., Wedi, N., Geenen, T., and Thiemert, D.: Destination Earth – A digital twin in support of climate services, Climate Services, 30, 100 394, https://doi.org/10.1016/j.cliser.2023.100394, 2023.

Huber, S. P., Zoupanos, S., Uhrin, M., Talirz, L., Kahle, L., Häuselmann, R., Gresch, D., Müller, T., Yakutovich, A. V., Andersen, C. W., Ramirez, F. F., Adorf, C. S., Gargiulo, F., Kumbhar, S., Passaro, E., Johnston, C., Merkys, A., Cepellotti, A., Mounet, N., Marzari, N., Kozinsky, B., and Pizzi, G.: AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance, Scientific Data, 7, https://doi.org/10.1038/s41597-020-00638-4, 2020.

Irrmann, G., Masson, S., Maisonnave, E., Guibert, D., and Raffin, E.: Improving ocean modeling software NEMO 4.0 benchmarking and communication efficiency, Geoscientific Model Development, 15, 1567–1582, https://doi.org/10.5194/gmd-15-1567-2022, 2022.

Jette, M. A. and Wickberg, T.: Architecture of the Slurm Workload Manager, in: Job Scheduling Strategies for Parallel Processing. JSSPP 2023. Lecture Notes in Computer Science, vol 14283, edited by Klusáček Dalibor, Corbalán, J., and Rodrigo Gonzalo P, pp. 3–23, Springer Nature Switzerland, Cham, https://doi.org/10.1007/978-3-031-43943-8_1, 2023.

Jones, M. D., White, J. P., Innus, M., DeLeon, R. L., Simakov, N., Palmer, J. T., Gallo, S. M., Furlani, T. R., Showerman, M., Brunner, R., Kot, A., Bauer, G., Bode, B., Enos, J., and Kramer, W.: Workload Analysis of Blue Waters, https://arxiv.org/abs/1703.00924, 2017.

Madec, G. and the NEMO System Team: NEMO ocean engine, Scientific Notes of IPSL Climate Modelling Center, 27. Zenodo, https://doi.org/10.5281/zenodo.6334656, 2022.

Manubens-Gil, D., Vegas-Regidor, J., Prodhomme, C., Mula-Valls, O., and Doblas-Reyes, F. J.: Seamless management of ensemble climate prediction experiments on HPC platforms, in: 2016 International Conference on High Performance Computing & Simulation (HPCS), pp. 895–900, https://doi.org/10.1109/HPCSim.2016.7568429, 2016.

Marciani, M. G., Castrillo, M., Utrera, G., Acosta, M. C., Kinoshita, B. P., and Doblas-Reyes, F.: Evaluating the impact of task aggregation in workflows with shared resource environments: use case for the MONARCH application, Geoscientific Model Development, 18, 9709–9721, https://doi.org/10.5194/gmd-18-9709-2025, 2025.

Merzky, A., Turilli, M., Maldonado, M., Santcroos, M., and Jha, S.: Using Pilot Systems to Execute Many Task Workloads on Supercomputers, in: Job Scheduling Strategies for Parallel Processing, edited by Klusáček, D., Cirne, W., and Desai, N., pp. 61–82, Springer International Publishing, Cham, 2019.

Merzky, A., Turilli, M., Titov, M., Al-Saadi, A., and Jha, S.: Design and Performance Characterization of RADICAL-Pilot on Leadership-Class Platforms, IEEE Transactions on Parallel and Distributed Systems, 33, 818–829, https://doi.org/10.1109/TPDS.2021.3105994, 2022.

Mickelson, S., Bertini, A., Strand, G., Paul, K., Nienhouse, E., Dennis, J., and Vertenstein, M.: A new end-to-end workflow for the Community Earth System Model (version 2.0) for the Coupled Model Intercomparison Project Phase 6 (CMIP6), Geoscientific Model Development, 13, 5567–5581, https://doi.org/10.5194/gmd-13-5567-2020, 2020.

Mölder, F., Jablonski, K., Letcher, B., Hall, M., Tomkins-Tinch, C., Sochat, V., Forster, J., Lee, S., Twardziok, S., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., and Köster, J.: Sustainable data analysis with Snakemake, F1000Research, 10, https://doi.org/10.12688/f1000research.29032.2, 2021.

Oliver, H., Shin, M., Matthews, D., Sanders, O., Bartholomew, S., Clark, A., Fitzpatrick, B., van Haren, R., Hut, R., and Drost, N.: Workflow Automation for Cycling Systems, Computing in Science & Engineering, 21, 7–21, https://doi.org/10.1109/MCSE.2019.2906593, 2019.

Patel, T., Liu, Z., Kettimuthu, R., Rich, P., Allcock, W., and Tiwari, D.: Job Characteristics on Large-Scale Systems: Long-Term Analysis, Quantification, and Implications, in: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–17, https://doi.org/10.1109/SC41405.2020.00088, 2020.

Puiggros, A., Castrillo, M., de Paula Kinoshita, B., Bretonniere, P.-A., and Agudetse, V.: Enhancing Data Provenance in Workflow Management: Integrating FAIR Principles into Autosubmit and SUNSET, in: EGU General Assembly 2025, Vienna, Austria, https://doi.org/10.5194/egusphere-egu25-2142, 2025.

Rodrigo, G. P., Östberg, P.-O., Elmroth, E., Antypas, K., Gerber, R., and Ramakrishnan, L.: Towards understanding HPC users and systems: A NERSC case study, Journal of Parallel and Distributed Computing, 111, 206–221, https://doi.org/10.1016/j.jpdc.2017.09.002, 2018.

Rousset, C., Vancoppenolle, M., Madec, G., Fichefet, T., Flavoni, S., Barthélemy, A., Benshila, R., Chanut, J., Levy, C., Masson, S., and Vivier, F.: The Louvain-La-Neuve sea ice model LIM3.6: global and regional capabilities, Geoscientific Model Development, 8, 2991–3005, https://doi.org/10.5194/gmd-8-2991-2015, 2015.

SchedMD: Fair Tree Fairshare Algorithm, https://slurm.schedmd.com/fair_tree.html, last accessed: 2025-12-05, 2025a.

SchedMD: Multifactor Priority Plugin, https://slurm.schedmd.com/priority_multifactor.html, last accessed: 2025-12-05, 2025b.

Schlagkamp, S., Ferreira da Silva, R., Allcock, W., Deelman, E., and Schwiegelshohn, U.: Consecutive Job Submission Behavior at Mira Supercomputer, in: HPDC '16, p. 93–96, Association for Computing Machinery, New York, NY, USA, ISBN 9781450343145, https://doi.org/10.1145/2907294.2907314, 2016.

Smith, B., Wårlind, D., Arneth, A., Hickler, T., Leadley, P., Siltberg, J., and Zaehle, S.: Implications of incorporating N cycling and N limitations on primary production in an individual-based dynamic vegetation model, Biogeosciences, 11, 2027–2054, https://doi.org/10.5194/bg-11-2027-2014, 2014.

Suarez, E., Bockelmann, H., Eicker, N., Eitzinger, J., El Sayed, S., Fieseler, T., Frank, M., Frech, P., Giesselmann, P., Hackenberg, D., Hager, G., Herten, A., Ilsche, T., Koller, B., Laure, E., Manzano, C., Oeste, S., Ott, M., Reuter, K., Schneider, R., Thust, K., and von St. Vieth, B.: Energy-aware operation of HPC systems in Germany, Frontiers in High Performance Computing, Volume 3 - 2025, https://doi.org/10.3389/fhpcp.2025.1520207, 2025.

TOP500: https://top500.org/, last accessed: 2025-6-1, 2025.

Turilli, M., Santcroos, M., and Jha, S.: A Comprehensive Perspective on Pilot-Job Systems, ACM Comput. Surv., 51, https://doi.org/10.1145/3177851, 2018.

van Noije, T. P. C., Le Sager, P., Segers, A. J., van Velthoven, P. F. J., Krol, M. C., Hazeleger, W., Williams, A. G., and Chambers, S. D.: Simulation of tropospheric chemistry and aerosols with the climate model EC-Earth, Geoscientific Model Development, 7, 2435–2475, https://doi.org/10.5194/gmd-7-2435-2014, 2014.