Response to Anonymous Referee #1

https://egusphere.copernicus.org/preprints/2026/egusphere-2026-539/#AC1

We sincerely thank the reviewer for the time and effort dedicated to evaluate our manuscript and for the constructive comments. In this document, we carefully address each point. Reviewer's comments appear highlighted in yellow, and proposed changes by the authors, in blue, alongside with the affected lines and the action performed (ADD, MODIFY, REPLACE or REMOVE). We are waiting for other reviewers to provide their comments before applying the changes in the manuscript.

General comments

The paper is not general in the sense that it does not provide a guideline for someone who would like to take autosubmit and apply this to a different workflow like a different numerical model to be run on yet another HPC system.

We thank the reviewer for sharing their understanding on this. While we have uploaded the workflows we have tested to WorkflowHub, we agree that we did not explain or provide examples of the configuration process in the manuscript.

After careful consideration, we have decided to add a note in the methodology section explaining that these configurations were made in accordance with the instructions on the specific pages in the Autosubmit documentation. Additionally, we have included an Autosubmit workflow configuration example in the new Appendix A. There, we have focused on the most relevant configurations, since the complete setup of an Autosubmit experiment is beyond the scope of this manuscript. It can be found attached at the end of this document for further reference.

[ADD] L 256: The experiment configuration presented in Table 1 was set up in accordance with the instructions in the Autosubmit documentation (Autosubmit Docs, 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.

"Autosubmit Docs" is a reference to the documentation website for the Autosubmit version used in the experimentation. "https://autosubmit.readthedocs.io/en/3.15.0-branch".

I well accept the authors choice of keeping the focus on task scheduling as such. Having said that, in this light I am missing some more in-depths discussion of the interrelation between

queue waiting times, shortcomings of SLURM parameter adjustments and its bypassing with the task aggregation approach.

We acknowledge the reviewer's view in the lack of discussion of the different actors in the queue time. Therefore, we have decided to extend the Fair Tree algorithm section to discuss how task aggregation affects the scheduling by including a comment in section 4.2 where we mention how task aggregation relates to fair share.

[ADD] L 195: 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.

Moreover, we have also extended the discussion section to mention the impact of such different actors. This further explanation comes from our previous work Marciani et al. (2025).

[MODIFY] L 323ff:

Furthermore, it was observed that the impact of the use of wrappers on the time-to-solution varied depending on ~~the level of~~ **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.

Finally, we have clarified in the conclusions.

[REPLACE] L 366-375:

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.

While I understand the underlying problem and the proposed solution I wonder if this can also be approached by simply making a single task running for longer time, e.g. using chunks of 10 years rather than 1 month. Thus I am left with a bit of a doubt that task aggregation is the only

solution for any Earth system model to reduce the time to solution.

The proposal to define longer chunks is interesting and valid. However, it is important to keep in mind that the workflow developer must have the flexibility to define the necessary granularity to divide the workflow into traceable atomic units. This division of the simulation allows, for example, setting up *checkpointing* to prevent the loss of results whenever an unexpected system error occurs. From a logical standpoint, performing checkpointing at the end of each chunk—when the model finishes—simplifies the workflow.

Task aggregation, beyond its benefits for scheduling, would not only allow this mechanism to be maintained but would also help adapt the workflow to the specific configurations of each machine, which may include restrictions on the number of jobs that can be submitted per Quality of Service (QoS).

On the other hand, although the experiments in this work do not cover them, there are other types of aggregation that allow concurrent—and even heterogeneous—tasks to be grouped within the same allocation. Therefore, increasing the size of the chunks, even while foregoing workflow-level mechanisms such as *checkpointing*, might not be enough in some cases.

Consequently, in certain scenarios, a developer might choose to configure chunks of one month or 10 years, depending on the workflow's specific needs and local platform constraints. In our specific case, selecting chunks of one month comes motivated by the fact that it is a typical configuration used in CMIP simulations. In any case, task aggregation is not the only way to reduce the time to solution for Earth System Models.

We understand how important it is for this to be clear to the reader, so we have reviewed the background to explain the potential of wrappers in scenarios where we have concurrent tasks with different resource requirements—even though this aspect is not tested in our work—to better illustrate that the goal is not merely to increase the size of the jobs processed by the machine, but to do so without sacrificing other mechanisms.

[MODIFY] L 125ff: 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.

Specific comments

Abstract

A matter of taste, but to me the abstract does not read as a short summary highlighting the

most important findings but already provides very detailed information (e.g. congested machines). Task aggregation and queue time are mentioned before they are defined.

We agree with the reviewer that the abstract reads technical. We have improved it by omitting unexplained domain-specific concepts. Regarding the specific observation for the term "congestion", we retain some of the references to it in order to ensure the accuracy of the explanation, but we have also added a brief comment with the reason behind this phenomenon.

[MODIFY] L 1ff:

Abstract

Earth System Models (ESMs) are commonly executed as complex workflows consisting of numerous interdependent tasks~~—the atomic unit of computation within the workflow—~~ which 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~~, complicating~~**. This complicates** the execution of long ensemble simulation workflows~~ due to the accumulated queue time, because~~**, where** each successive simulation task can only be submitted once the preceding one finishes~~. These queue times are~~**, causing queue time to accumulate—**the duration for which the jobs~~—the compute units sent to be executed remotely—~~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 ~~durations~~ **intervals** from the first submission to the completion of the final task, by applying ~~task aggregation~~ **a solution** to reduce subsequent requests for resources and, consequently, reducing queue times. **This solution, known as** ~~T~~task aggregation**,** ~~is a strategy that~~ consists of grouping multiple tasks and submitting them as a single job, respecting their dependencies, and without altering their underlying logic.

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 ~~time~~ **gains**, we compute the impact of aggregation by ~~obtaining the differences between the Simulated Years Per Day (SYPD) and the Actual Simulated Years Per Day (ASYPD), two~~ **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 ~~in the ASYPD~~ of up to 23~~,04~~% **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 ~~congestion~~ **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.

Introduction

The introduction already provides details of the methods. I propose to shift these to your section 3 (Methods).

We accept the reviewer's suggestion, although we also prefer to keep a reduced description of the methodology we followed.

[REPLACE] L 91-99:

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 running with the Autosubmit workflow manager.

[MODIFY] L 237ff:

The workflows we ran were based on **testing configurations of** Auto-EC-Earth3~~ testing configurations~~ (Garcia Lopez et al., 2025)~~, varying~~ **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 according to their availability on the respective platforms, as well as the computational resources available for each. Since we ~~are~~ **were** not interested in the output of the models [...]

L 68ff I propose to move the introduction to autosubmit after the introduction of other work

(as state of the art) and thus address the mentioned problems mentioned rather than jumping back and forth between your approach and other work.

We agree that modifying the narrative flow can make the text easier to read. We have opted for a slight restructuring to, first, introduce the Autosubmit's wrappers solution—the one used in this work—explaining that it has been used without validation until now, and then continue with a state-of-the-art by explaining the theoretical study on task aggregation and listing similar solutions in other areas.

Changes in Lines 68-88:

- Paragraph starting on L 68 remains the same.
- The two paragraphs starting on lines 74 and 80 have been swapped.

[MODIFY] L 89: ~~Then~~**Building on these findings and addressing the lack of practical validation**, our objective is to measure the impact of task aggregation in real scenarios [...]

L 95ff The political impact of simulations with EC-Earth3 is not really relevant in this context.

With the changes made in response to the first comment on the "Introduction" section, we consider this matter addressed.

L 99 I have had a hard time understanding why your solution is transversal (see my remark for your sec. 3)

Task aggregation is a transversal solution in the sense that any scientific workflow consisting of many homogeneous sequential tasks can benefit from its application in situations of congestion on the HPC machine. However, we do not propose a transversal mechanism beyond explaining the Autosubmit mechanism—since it is the workflow manager used in our experiment—and mentioning others.

Based on these results, any user in the HPC community could find inspiration to implement an equivalent solution in applications from other areas with a similar workload. We propose adding a clarification in the conclusions to highlight the importance of the workload over job logic.

[MODIFY] L 377/378: [...] on shared HPC platforms**, since the scheduler is agnostic regarding the internal operation of the jobs.**

L 102/103 Isn't this sentence better placed in the next and final paragraph of the introduction? You assume that the reader is already familiar with "fair share". Thus I wonder if its mentioning here is too early.

We agree with the reviewer and the sentence will be moved to the last paragraph. It is possible to keep it since the *fair share* had been already defined in lines 83-84 as "the Slurm scheduling factor that balances the utilization of computational resources between users by prioritizing less served users over high-usage ones".

[REMOVE] L 102/103: ~~Finally, we explain why the fair share is relevant for queue time and how it impacts the performance of task aggregation~~.

[MODIFY] L 108: In Section 4 we compile, analyze, and discuss the results of the previous experiments~~ and~~**, 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 [...]

L 104 tells me what I have just read and can be deleted.

We agree that it can sound repetitive, so we propose directly removing the first section explanation.

[REMOVE] L 104: This document is structured into five sections, the present one being Section 1~~, which introduces the project outline~~.

L 111ff This is just a plain repetition of what was said in line 105 and 106.

We acknowledge that the description of the background provided in the introduction is pretty similar to the one already present in that section. We consider the background needs a richer and more meaningful introduction.

[REPLACE] L 111ff:

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 workflow, alongside the metrics we used to evaluate its performance.

Background

L 155 I do not get why the backfill algorithm plays a negative role. It is just designed to fill gaps if possible to guarantee an efficient use of the whole system.

We acknowledge that we have not been entirely clear in explaining the impact of the backfill algorithm on task aggregation. The algorithm does not directly penalize aggregated tasks. The problem is that, by increasing the time window required by each job sent to the scheduler, it becomes less likely that the backfill algorithm will find a slot that meets its resource and walltime requirements. In such cases, the advantage that the backfill algorithm provides for the smaller tasks in the machine is lost.

Despite this, curiously, we conclude later that "We observed that single jobs stood about the same time, or less, in queue as their longer grouped counterparts despite the backfill algorithm" (L 363,364). These are the changes we have made to clarify this issue:

[MODIFY] L 155: Therefore, the backfill algorithm ~~plays a negative role with respect to task aggregation~~ **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.**

L 187 is of an account: please rephrase.

[REPLACE] L 187/188: Because the lowest LF belongs to an account, the algorithm recurses.

L 215ff Is the detailed description of EC-Earth3 really necessary in this context here?

We agree that it is not necessary to go into as much detail about the model as it is about the workflow, so we propose replacing the section on EC-Earth3 with a brief description of Auto-EC-Earth3.

[REPLACE] L 213ff:

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, and diagnostics.

Below, in the Methodology section, we will include references to the articles on the specific models as they are mentioned. This way, we also avoid discussing the coupled EC-Earth3 models that are not involved in our experiments.

[MODIFY] L 245ff:

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.

L 226, 227 Repetition from the introduction

We agree that the full names for the performance metrics can be removed since they have been explained before.

[MODIFY] L 226/227:

Out of all the metrics that the authors introduce, we highlight the ~~Simulated Years Per Day (~~SYPD~~)~~ and **the** ~~Actual Simulated Years Per Day (~~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.

L 236 we tried: Did you succeed?

It is true that this causes confusion. In the "Limitations" section of our study, in L 329, we stated that in the case of MareNostrum 5 this was impossible to achieve because one of the users had a prior usage that could not be restored according to the scheduler's policy. We believe that the following modification makes the methodology clearer:

L 235: [...] 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**.
L 236: ~~We tried to have the same usage on both users so that they had the same initial fair share.~~

Methods

Why does a chunk size has to be 12 months? Why can't it be 120? This would allow one to run 5 single jobs without the need of wrapping? Is there some limitation set by EC-Earth3 or any of the internal components which stand in the way? Are there any other advantages using chunk sizes of 12 and thus a wrapper size > 1 to reach the maximum allowed queue time?

At the workflow configuration level, it has been decided that the chunk size should be 12 months because this is a common configuration in the previous experiments with this model that we have been able to analyze.

Regardless of this, defining smaller chunks has the advantage that post-processing applications—which often run in parallel with the simulation—can receive the simulation output in smaller batches, thereby reducing the time these tasks spend waiting for the corresponding simulations to finish, since Autosubmit is capable of resolving dependencies between the simulations and the post-processing tasks, even if the simulation tasks are wrapped.

Furthermore, in the event of an error on the computing node, there is a risk of losing the simulation progress, so having smaller chunks would reduce the data loss.

At the task aggregation level, although defining a single 120-month chunk is possible in EC-Earth3 and is also equivalent to defining wrappers consisting of 10 chunks of 12 months each (in terms of simulated period, resource requirements, and computation time), we would lose any logic that may have been implemented at the workflow level, such as the checkpointing mechanism. To the best of our knowledge, not all models have this type of mechanism built in.

We believe that the response to the third comment in this document may provide greater context.

How and where in the manuscript do you make use of Slurm parameters fair share, Level Fair Share, and raw usage that you collected? Earlier on, you explained Level Fair Share. How is this related to Fair Share and Raw usage mentioned in the figures. I am afraid that the reader gets a bit lost here. A bit more guidance would be nice.

We appreciate the point of view. The fourth section (results) analyzes how the fair share changes as our use of the machine increases, quantified as raw usage. Although we have also calculated the Level Fair Share, these two metrics already allowed us to present conclusive results. Therefore, we will remove the statement that we need to calculate it for the analysis:

[REMOVE] L 260: [...] we gathered some Slurm parameters, including fair share~~, Level Fair Share~~ and raw usage [...]

In any case, the Level Fair Share values will remain available for readers interested in their examination.

Regarding the "raw usage", it is true that the term is not explicitly mentioned in the explanation of the Fair Tree algorithm. We therefore propose the following modification:

[MODIFY] L 181ff: 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**.

Results and discussion

I interpret the differences in SYPD for the wrapped vs unwrapped cases as system fluctuations. The percentage and speedup with a precision of up to two decimals (even though true for these single cases) suggest an exactness which is questionable. I assume that these numbers are rather volatile and depend on the overall load of the system at a given time.

We share the interpretation of these variations, given that the workflow pairs were run under equivalent workload conditions, as suggested by the almost identical trends in fair share over the course of execution. In this regard, we propose the following changes so that the reader can understand this more directly:

[MODIFY] L 270/271: 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**.

As for the decimal precision of the gain, it is currently not possible to quantify the impact of such fluctuations on the final result. For this reason, we propose reducing the precision to one decimal place to minimize the influence of this variability on our conclusions.

[MODIFY] L 24/25: [...] representing an improvement in the ASYPD of up to ~~23,04%~~**23,0%** in the case of the platform with the highest congestion.

[MODIFY] Table 2:

Performance metrics						
SYPD	~~7.62~~ **7.6**	~~7.78~~ **7.8**	~~14.30~~ **14.3**	~~14.34~~ **14.3**	~~19.92~~ **19.9**	~~21.08~~ **21.1**
ASYPD	~~7.38~~ **7.4**	~~5.68~~ **5.7**	~~14.30~~ **14.3**	~~14.22~~ **14.2**	~~19.91~~ **19.9**	~~20.83~~ **20.8**

[MODIFY] L 275: This also translates to an ASYPD of ~~7.38~~**7.4** for the wrapped experiment and ~~5.68~~**5.7** for the reference.

[MODIFY] L 289ff: We observe in MareNostrum 4 that the difference between the SYPD and ASYPD of the wrapped experiment is of ~~0.24~~**0.2** whereas in the unwrapped case it is ~~2.10~~**2.1**. This equates to a relative difference of ~~3.15%~~**2,6%** and ~~26.99%~~**26.9%**, respectively. The latter value is in line with the overhead observed by Acosta et al. (2024).

[MODIFY] L 304/305: This is also reflected in the low spread of ~~0.12~~**0.1** between SYPD and ASYPD in the reference simulation.

[MODIFY] L 367ff: For instance, although all platforms had similar queue time reduction rates, in MareNostrum 4, the improvement in the ASYPD—a standard performance metric of the field—was of ~~23.04%~~**23.0%**, whereas in MareNostrum 5, it was only of ~~0.56%~~**0.7%**, or even negligible in the case of MeluXina.

Figures 4, 5 and 6: The figure title is a repetition of the figure caption.

We have removed the titles from the three figures mentioned and updated their captions to include the *chunk size* that was originally included in the titles:

[REPLACE]

- **Caption of 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.
- **Caption of 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.
- **Caption of 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.

Conclusions

You described task aggregation of rather homogenous tasks. If other tasks like postprocessing were added would you expect similar benefits?

If post-processing tasks were added, we would expect similar results as long as a vertical—or sequential—structure is maintained, such as the one we have tested in this study.

However, as mentioned previously, post-processing is typically run in parallel with the simulation. While it is true that it could be added alongside the simulation using other structures—such as the hybrid structures we briefly introduced in this study—we have no evidence that this would yield any benefits in terms of scheduling beyond meeting the limits defined for the Quality of Service.

Would there still be a benefit if all users on a congested system opted for task aggregation? How much could be compensated by adjusting the SLURM parameters? Wouldn't it make more sense to use reservations to run those long experiments.

We understand the question and recognize that wrapping could reduce the throughput of the system. But when we consider the climate simulations, with their strong temporal dependency, we see that the simulations are considerably slowed down (by a factor of ten, as the results of this paper show).

One possibility is the one the reviewer mentions, to request a reservation. But this is hardly ever granted in the European research HPC infrastructure, where most climate simulations are executed.

This takes our workflows to compete for the resources in Slurm. Ideally, the scheduler would take into consideration all the current running tasks, the available resources, and the follow up tasks, but it does not. And, looking into the parallel job scheduling literature, neither it seems that it is feasible due to that the current implementation is already demanding, even more if the scheduler has to look forward.

Therefore wrapping is the solution that the community has found to mitigate the issue with queue time accumulating that much [1].

[1] Lannon, Kevin, et al. "Reshaping Analysis for Fast Turnaround: Leveraging Concurrency to Reduce Latency in Late-Stage LHC Analysis Workflows." EPJ Web of Conferences. Vol. 337. EDP Sciences, 2025.

Response to Anonymous Referee #2

https://egusphere.copernicus.org/preprints/2026/egusphere-2026-539/#AC2

We thank the reviewer for this important comment. It has helped us identify two aspects that were not sufficiently clear in the original manuscript: the computational dimensions of the submitted jobs and the definition of CMIP experiments. We have therefore added the missing job information and revised the document accordingly.

At the same time, we respectfully clarify that the 10-year CESM job layout discussed by Mickelson et al. (2020) is one valid centre- and model-specific implementation, but it does not establish a common job segmentation strategy across CMIP. CMIP defines experiment periods and data requirements, while the segmentation of those experiments into scheduler jobs depends on the model throughput, configuration, workflow implementation, platform policy, and operational choices of each modelling centre.

We would also like to point out that, although configuring the simulation period to approach the wallclock limit may reduce the amount of queue submissions, we strongly recommended to perform fine-grained segmentation so that the workflow manager can adapt the execution to the workload conditions and the constraints of the machine during runtime. Especially as we increase the resolution, and therefore the parallelism, which increases the likelihood of failures. By employing task aggregation, the simulation and the produced data are secured through workflow and model mechanisms, with the retries and the restart files.

In the following, we provide a point-by-point response to each comment and describe the corresponding changes made to the manuscript.

Specific comments

A key element missing in the description is the size of the jobs and what percentage of the system they represent.

We agree. Although the corresponding configurations were available in the archived workflows, this information should have been reported directly in the manuscript. We have extended Table 1 to include the number of nodes requested for each simulation task, the percentage of the corresponding CPU partition, the requested walltime per task, and the maximum requested walltime of the vertical wrappers.

The individual simulation tasks used 11 of 3,456 nodes on MareNostrum 4, 7 of 6,192 DDR nodes on MareNostrum 5, and 10 of 573 CPU nodes on MeluXina. These values correspond to approximately 0.32%, 0.11%, and 1.75% of the corresponding CPU partitions, respectively. Since the tasks within a vertical wrapper execute sequentially, wrapping does not

multiply the number of requested nodes. It retains the same node allocation and increases the requested walltime.

[REPLACE] Table 1 (p. 10): Added columns “Nodes per chunk” and “Wallclock per chunk (hours), and restructured the table to avoid overflow.

	Chunks				Wrappers		Other information
Platform	Quantity	Size (months)	**Nodes**	**Wallclock**	Quantity	Size (chunks)	Additional coupled models
MareNostrum 4	50	12	**11**	**03:30**	5*	10*	PISCES-LPJG-TM5
MareNostrum 5	40	12	**7**	**02:30**	4	10	–
MeluXina	50	12	**10**	**01:30**	5	10	–

[ADD] L 234/235: 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. **These systems have 3,456, 6,192, and 573 nodes, respectively.**

Small jobs will bias the results. The total queue time was so small compared to the run time that even when aggregating reduced total queue time, it had little impact on the ASYPD for MareNostrum 5 and MeluXina (results in Table 2).

We acknowledge that the limited queue times observed on MareNostrum 5 and MeluXina naturally reduced the impact of task aggregation on the overall ASYPD. However, we believe that these less congested platforms enrich our results given the diversity of workloads of the tested environments, rather than biasing the results.

The results still show a clear and consistent improvement in the metric directly targeted by task aggregation. The cumulative queue time was reduced by factors of 11.33 and 12.33 on MareNostrum 5 and MeluXina, respectively, despite the aggregated jobs requesting longer walltimes and therefore potentially being less favourable for backfilling. On MareNostrum 5, this also produced an improvement in ASYPD. On MeluXina, variability in model runtime offset the queue-time reduction in the overall time to solution, but the reduction in the time spent waiting for resources remained clear.

MareNostrum 4 represented a congested environment in which queue time constitutes a substantial part of the workflow duration and aggregation produces a clear improvement in ASYPD. Conversely, MareNostrum 5 and MeluXina represented lightly congested environments

in which the same mechanism reduced cumulative queue time, but the small initial queue overhead limited its effect on end-to-end performance.

This distinction is one of the relevant contributions of the paper: task aggregation consistently reduced cumulative queue time in the evaluated workflows, while the magnitude of the resulting improvement in ASYPD depended on the workload and congestion of the platform. The results therefore help identify the conditions under which the reduction in queue time translates into a substantial operational benefit, rather than suggesting that the same ASYPD improvement should be expected on every system.

But the biggest flaw in the experiment design is that a typical CMIP-class long (100 year) simulation will have each job use nearly the maximum allowed wall time, especially on a crowded machine. This makes it impossible to do any vertical aggregation of successive jobs since putting even 2 jobs together will exceed the maximum wall time. If the wrapper is creating job sizes that are effectively 10 years long that tells me the chunk should be 120 months, not 12. In the Mickelson 2020 paper, each CESM job submission is 10-years long. While it does make sense that fewer queue submissions will result in a shorter total queue time, this is not an option for most CMIP-class climate simulations.

We agree that vertical aggregation would not be applicable when an individual simulation—or any other—task already approaches the maximum walltime allowed by the platform without changing the segmentation. We have added this as an explicit applicability condition.

However, the available literature does not indicate that this condition characterises all, or necessarily most, CMIP workflows. The CMIP6 design encompasses a federated set of experiments and modelling centres rather than prescribing a common scheduler-job structure (Eyring et al., 2016). Balaji et al. (2017), when defining the CPMIP performance metrics, describe production-run segments as usually spanning at least one month and often one or more years, illustrating that production segmentation is not fixed at 10 years.

This flexibility is also explicit within the CESM ecosystem. CIME (Common Infrastructure for Modeling the Earth), the case-management and workflow infrastructure used by CESM, allows long simulations to be divided into a sequence of automatically resubmitted jobs. Its User Guide instructs users to select the length of each run segment according to the model throughput and the batch-queue limits of the computing platform. Segment lengths can be specified in days, months, or years, while restart files allow the simulation to continue in a bit-for-bit exact manner from the preceding segment. This demonstrates that CESM does not prescribe a fixed 10-year scheduler-job granularity; instead, the segmentation can be adapted to the model configuration, platform limits, and operational requirements (CIME User Guide).

More broadly, Acosta et al. (2024) report substantial variability across 33 CMIP6 experiments, 14 HPC systems, and 14 modelling institutions, and explicitly note that this diversity makes general operational rules difficult to formulate.

Mickelson et al. (2020) document a valid CESM workflow developed for the particular requirements of NCAR and Cheyenne. Our study addresses a different but also operationally relevant workflow class: segmented simulations in which several consecutive tasks fit within the walltime limit. This includes the annual EC-Earth3 configurations evaluated here and other workflows in which segmentation is chosen to preserve task-level checkpoints, provenance, selective retries, and adaptation to platform constraints through the capabilities of the workflow managers. Task aggregation preserves these task boundaries while changing how the tasks are submitted to the scheduler.

Moreover, for that reason, although it would be possible to make a chunk of 120 months instead of aggregating 10 chunks of 12 months, we would be losing these key features at the workflow management level and increasing the risk of wasting computational resources if the computing environment experiences intermittent failures. For this reason, we strongly recommend using segmentation whenever possible.

We have therefore revised the manuscript to define the scope more precisely and to clarify how task aggregation differs from just simulating for longer. Our results provide direct evidence for segmented vertical ESM workflows in which at least two consecutive tasks fit within one allocation and queueing overhead is non-negligible. We do not claim that vertical aggregation is applicable to configurations in which each individual task already consumes the available walltime. We also extended our conclusions by recommending the community to keep developing fine-grained workflows in order to preserve all these workflow management-level capabilities, thus allowing the workflow manager to aggregate.

[ADD] L 13/14: Task aggregation is a strategy that 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.**

[ADD] L 72/73: 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.**

[ADD] L 116-118: 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.

[ADD] L 121:

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.

[MODIFY] L 202/203: It is responsible for recording the experiments, their configuration, the execution of the workflow, **handling failures,** and ~~provenance~~ tracking **provenance**.

[MODIFY] L 220-222: The workflow implementation of EC-Earth3 in Autosubmit, also called Auto-EC-Earth3, 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, ~~and~~ 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.**

[MODIFY] L 353-355: 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 is called *task aggregation*.~~ **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*.**

[ADD] L 376-378: 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. 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.**
