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.**
