Response to Reviewer 2 comments

Quantifying the scaling penalty of sequential coupling: The sequential coupling cost (Cseq) metric for Earth System Model components

by

Oriol Tintó Prims, Sergi Palomas, Simone Vacondio, and Mario C. Acosta

Reviewer (R)
Authors (A)

We thank the reviewer for their positive evaluation of our manuscript, for recognizing the value and practical utility of the sequential coupling cost (C_{seq}) metric, and for recommending publication after minor revisions. We appreciate the insightful suggestions provided, which helped us improve the clarity, context, and visual presentation of our work.

In response to your comments, we have thoroughly updated the manuscript. Below, we provide point-by-point responses detailing all specific changes incorporated into the revised version.

- - Line 26: Please define MPMD when it is first introduced. This will make the introduction more accessible to readers who are less familiar with HPC terminology.- We thank the reviewer for making us notice that error, It has been addressed in lines 28-29.

  > Traditionally, ESMs follow one of two coupling strategies (Mechoso et al., 2021): concurrent coupling using a Multiple Program Multiple Data (MPMD) approach or sequential coupling using a single-binary architecture.

- *- Introduction: The introduction would benefit from a short paragraph, or at least a few sentences, explaining why some modelling systems choose a single-executable/sequential design in the first place. For example, in the IFS–NEMO case, data assimilation applications and operational constraints are important practical motivations. This would help present the architectural choice as a trade-off rather than simply an inefficiency.*

  We thank the reviewer for this insightful suggestion and agree that framing sequential coupling as a design trade-off rather than purely an inefficiency improves the manuscript. We added a sentence incorporating the reviewer's suggestion, along with two supporting references, in lines 34–35:

```
In practice, the architectural choice is often dictated by
operational considerations and software engineering constraints
rather than raw parallel efficiency (Mogensen et al., 2012; Rackow et
al., 2025).
```

- *Lines 75–76: The text currently emphasises that Earth science models scale sub-linearly. I think an even more important point is that different Earth system components often scale differently. This is central to why sequential coupling can be inefficient: all components are forced to use the same resource pool, even when their optimal resource allocations differ substantially.*

  We thank the reviewer for this insight. We fully agree that a major advantage of concurrent coupling is the flexibility to independently tailor resource allocations to components with different scaling profiles.

  However, we wish to clarify that sequential coupling incurs a fundamental performance penalty even when components have identical scaling behaviors and optimal resource allocations. As illustrated in Figure 2, sequential execution forces components to run at twice the speed to achieve the same target throughput. Because of sub-linear scaling, doubling the required execution speed demands disproportionately more computational resources, inherently degrading efficiency and capping maximum speed.

  This behavior is confirmed in our OpenIFS and M7 chemistry case study (Section 4.3, Figure 7). Although both components exhibit very similar scalability profiles, the sequential setup still incurs a high overhead to reach 10 simulated years per day compared to the concurrent configuration. Thus, sub-linear scaling combined with sequential coupling inherently generates a sequential coupling cost, even in symmetric setups.

  To incorporate the reviewer's perspective, we have updated the text at the end of the conceptual section (lines 92-94) to explicitly acknowledge that real-world components differ and that concurrent coupling provides the flexibility to address this:

```
Although this simplified scenario assumes identical components, Earth
system model components typically show distinct scaling profiles.
While sequential coupling forces all components to share the exact
same resource pool, concurrent coupling enables users to adjust
resource allocations to match each component's specific demands.
```

- *- Figure 1: The figure is useful and the logic is clear. However, in the sequential panels, the labels T and >T appear above pairs of bars. Although this is logically correct, it may be visually misread as each individual bar having duration T or >T. I suggest labelling the individual sequential bars more explicitly, for example: linear sequential case as T/2+T/2=T, and sub-linear sequential case as >T/2+>T/2=>T.*

  We thank the reviewer for pointing this out. To remove any ambiguity regarding whether the labels apply to individual components or the total execution time, we have updated Figure 1 by adding brackets that span across both sequential bars. This visually clarifies that **T** and **>T** represent the total combined wall-clock time for the sequential execution.

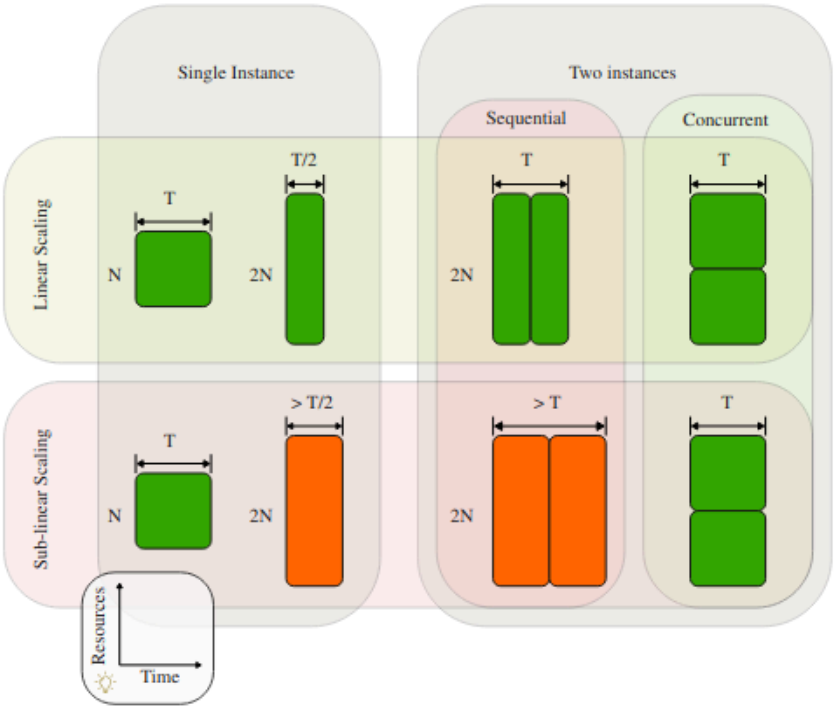

Figure 1. Comparison of total wall-clock time for different coupling strategies. While concurrent coupling maintains the baseline runtime when doubling the resources, when the models exhibit sub-linear scaling, the sequential setup results in a greater total execution time.

- *- Figure 2: I would avoid labelling the y-axis simply as "Speed". For the red and green curves, this is not the speed of an individual component, but rather the effective speed of completing the coupled system. A label such as "Effective coupled throughput", "Effective system speed", or simply "Coupled simulation speed" would reduce possible ambiguity.*

  We thank the reviewer for this suggestion. Because the y-axis plots both individual component performance (blue curve) and coupled system performance (red and green curves), using a coupled-specific label on the axis itself would be inaccurate for the single component.

To keep the axis label concise and applicable to all curves while resolving any ambiguity, we have explicitly clarified its meaning in the caption of Figure 2:

```
Speed denotes the completion rate (throughput) of the given model
setup, representing single-component execution alongside effective
coupled system performance.
```

- *- Lines 136–142: Around line 133 the paper states that no specialised performance analysis software is required, but shortly afterwards the Escalador library is introduced for the analysis workflow. This is slightly confusing. -*

  We thank the reviewer for highlighting this point. By "no specialized performance analysis software," we meant that the method requires no runtime code instrumentation, tracing, or dedicated profiling tools (e.g., Extrae, Scalasca, or VTune) to collect data, relying instead on standard execution logs.

  The Escalador library is simply a lightweight Python helper tool we used to automate the curve fitting and metric evaluations across our use cases. To clarify that the methodology itself is lightweight and tool-agnostic, we added Appendix A, which contains a complete standalone 20-line Python implementation using only numpy and scipy, and updated the text in Section 4 (lines 149-153) with the following:

  ```
  While the calculation can be performed using basic scientific
  software (a complete 20-line Python implementation using only numpy
  and scipy is provided in Appendix A), we used the Escalador library
  (Tintó Prims and Palomas, 2026) to standardize and automate this
  workflow across all test scenarios.
  ```

- *- Section 4.1 and later sections: Expressing node counts as decimal numbers, for example 30.6 nodes, feels somewhat artificial from a practical HPC perspective. Since real allocations use integer node counts, I suggest rounding these values, at least in the main text, or clearly stating that decimal node counts come from the analytical fits and represent idealised estimates. This would make the results easier to interpret operationally. -*

  We thank the reviewer for this practical observation. We agree that operational HPC allocations require integer node counts. To clarify why fractional values are presented without introducing rounding artifacts into our calculations, we have added the following explanatory note at the end of Section 4's introduction (lines 154-157):

  ```
  Because these optimal values are evaluated directly from the
  continuous analytical fits, the resulting resource requirements are
  ```

naturally expressed as fractional node counts (e.g., 30.6 nodes). While real-world HPC allocations might require integer numbers of whole nodes, retaining decimal values in our calculations avoids rounding artifacts and precisely captures the theoretical scaling behavior dictated by the analytical models.