

Dear Reviewer,

We sincerely thank you for your careful review and insightful comments on our manuscript. Your thoughtful suggestions are highly appreciated and have provided valuable guidance for improving the clarity, accuracy, and overall quality of our work. We have carefully examined each of your points. In the responses below, we provide detailed explanations and describe the specific revisions we will make to the manuscript to address your suggestions. We are committed to incorporating your feedback thoroughly and believe that these revisions will significantly enhance the manuscript.

General comment 1

Reviewer comment:

As pointed out also by Referee #1, the link to previous efforts to adapt CESM to Sunway systems is not clear. For instance, the authors cite Zhang et al. (2020), describing the porting of CESM1.3 to the previous Sunway supercomputer. Why is the optimized CESM1.3 not used as the starting point for optimizing CESM2? Does CESM2 benefit from any optimization strategy devised for CESM1.3? Why is a different optimization approach (based on OpenMP) pursued here? The authors might have very valid arguments, but these should be clearly stated in the text.

Response:

We thank the reviewer for the comment. As explained in our response to Reviewer 1, the main reasons why this work did not directly reuse the previous CESM1.3 porting results are as follows: (1) CESM2.2 underwent extensive refactoring in multiple core components compared with CESM1.3, making the finely-tuned code from the earlier version incompatible; (2) this study targets the new-generation Sunway supercomputer, whose architecture differs significantly from the previous platform, so optimization strategies tailored for the old system cannot be directly applied; (3) the CESM codebase is extremely large, and previous optimizations only covered limited portions, whereas this work leverages automated tools (O2ATH) to achieve broader, cross-component acceleration, representing an extension and upgrade rather than a repetition of prior work. In the revised manuscript, we have clarified these differences to help readers better understand the necessity and novelty of our work. Please see line 109-117.

General comment 2&3

Reviewer comment:

The title is misleading and vague. From my perspective, OpenMP cannot be deemed as an “automated” software adaptation tool, since compiler directives must be inserted into the code manually; it is rather a “high-level” programming paradigm. For this same reason and because of the fine-grained refactoring at the Athread-level presented in Section 3.4, the porting cannot be flagged as “non-intrusive” (as stated in Section 3.2). Moreover, while the title alludes to a generic porting and optimization approach, I believe that the proposed optimization toolchain is much tailored to the peculiar hardware architecture of the Sunway chips; I doubt the overall porting strategy can be applied to other machines and other models as-is. Finally, it is not clear what “highly-efficient” means: which is the metric with respect to the methodology is efficient? Energy-to-solution, time-to-solution, hardware resources utilization, ...?

The authors use the expression “ultra-high-resolution” to refer to the 5v3 model configuration, which I find exaggerated. Simply “high-resolution” or “kilometer-scale” would be more appropriate.

Response:

We thank the reviewer for the valuable comments regarding the title and related wording. We fully understand your concerns about terms such as “automated”, “non-intrusive”, “highly-efficient”, and “ultra-high-resolution”. In response, we will carefully reconsider and revise the manuscript title and the relevant expressions in the revised version.

General comment 4**Reviewer comment:**

The authors do not provide a well-defined baseline to assess the performance of the optimized code. The actual speed-up offered by the CPEs can only be measured by deploying the unoptimized code (i.e. the code executing on the MPEs only) and the optimized code on the same number of nodes. However, Fig. 1 only shows results up to 18000 processes, while the 5v3 simulation conducted with the optimized code is scaled up to the full machine. More comments will be provided in the next section.

Response:

We sincerely thank the reviewer for the valuable comments. We fully understand the importance of comparing the “unoptimized baseline” with the “optimized performance” at the same scale. The performance tests shown in Figure 1 only extend up to 18,000 processes as the unoptimized version exhibits very low efficiency in both the initialization and

computation phases on MPEs, which makes large-scale runs, including those on the full machine scale with over 600,000 processes, excessively time- and resource-consuming. Therefore, it is not feasible to perform complete and reproducible performance tests on the full machine for the unoptimized code. In contrast, the optimized version significantly improves the efficiency of both initialization and computation, enabling the successful execution of the 5v3 simulation at full machine scale. We have added explanations in the revised manuscript, please see line 254-259.

General comment 5

Reviewer comment:

The syntax requires a comprehensive review. Many sentences are hard to follow and their meaning is not always clear. Moreover, while some acronyms are expanded multiple times (e.g., “CESM” at L378-379), others are not explained at all (e.g., “gld/gst” at L135). Given the target audience of the journal, any terminology belonging to the technical jargon must be explained.

Response:

We thank the reviewer for the valuable comments regarding language expression and terminology standardization. In the revised manuscript, we will carefully review and polish the text throughout to ensure clarity and logical coherence. Additionally, we have standardized the use of abbreviations, providing full explanations for all technical terms (including gld/gst, etc.) at their first occurrence. We sincerely appreciate the reviewer’s reminder.

Specific comment 1

Reviewer comment:

Section 2.1: Since the target Sunway supercomputer is rather new and not much information can be found on the internet, it would be useful to have more details about, e.g., clock frequency, system interconnect, software stack.

Response:

We thank the reviewer for the suggestion. In the revised manuscript, we have provided additional background information on the new-generation Sunway supercomputer in Section 2.1 in the revised manuscript.

Specific comment 2.1

Reviewer comment:

Table 1: I assume the table reports the total number of grid points for each model component. If so, there must be a typo, since the atmosphere component appear to feature the same number of grid points for 9v5 as for 25v10.

Response:

We appreciate the reviewer's careful observation admit that there is a typographical error in Table 1 regarding the number of atmospheric grid points. For the 9v5 case, the correct number of atmospheric grid points should be 6,998,400, rather than the value currently shown in the table. We have corrected this in the revised manuscript.

Specific comment 2.2

Reviewer comment:

How do you compute the throughput of each individual component? It seems to me that the SDPD of each component is proportional to its runtime fraction, which does not make any sense to me.

Response:

We thank the reviewer for raising this important point. The method of estimating module-level throughput by proportionally scaling from the overall SDPD runtime is indeed inappropriate. The "Total SDPD" labeled in the figure actually represents the cumulative runtime of individual modules, intended only for internal calculation of each module's runtime proportion and for generating visualization. Due to an error in the data aggregation process, this cumulative value was mistakenly plotted in the figure and therefore does not have actual physical significance. We sincerely apologize for any confusion caused by this labeling error and have corrected Figure 1 in the revised manuscript.

Specific comment 2.3&2.4

Reviewer comment:

L246-248: The reported increase factors in computational cost assume the time-step is kept fixed, i.e., a time-step satisfying the CFL condition at high resolution is also used at the coarse resolution. Please clarify.

L248-251: This is too vague. The statement that the increase in compute cost is inversely

proportional to the square of the grid spacing only holds if both the time-step and the number of processes are kept constant. The second condition is clearly not met here. Ultimately, could you explain how you choose the number of processes for each configuration.

Response:

We thank the reviewer for the detailed comments regarding the relationship between resolution changes and computational efficiency. The efficiency decline factor reported in the manuscript is based solely on the increase in computational cost per single iteration step due to the larger number of grid points, and does not take into account adjustments in time step or CFL constraints caused by higher resolution. Therefore, the statement that “the simulation speed is inversely proportional to the square of the grid resolution” refers only to the per-step computational cost and does not directly apply to the full-time integration.

For the selection of the number of processes for each configuration, we first run serial benchmark cases to evaluate the computational load and performance proportion of each component, then plan the PE layout based on the benchmark results, and finally allocate specific process counts according to grid resolution and component computational demands. As the number of grid points or resolution increases, the number of grids assigned to each process also increases, resulting in longer computation time per step.

We have clarified this point in the revised manuscript, please see line 271-279.

Specific comment 2.5

Reviewer comment:

L256: The authors say that “the load balancing [...] is particularly important for POP”. Could you please elaborate on this point. How does the domain decomposition work? Which are the criteria guiding the automatic decomposition? Why is load balancing so problematic for POP?

Response:

We thank the reviewer for raising this important issue regarding load balancing in the POP model. Taking the ts015 grid with 7,200 processes as an example, this grid can be partitioned in two ways: (1) Multiple physical blocks with specified sub-block grid partitioning: OCN_NX = 2400, OCN_NY = 1680, POP_BLCKX = 10, POP_BLCKY = 14, POP_MXBLCKS = 4; (2) Single physical block with specified sub-block grid partitioning: OCN_NX = 2400, OCN_NY = 1680, POP_BLCKX = 10, POP_BLCKY = 56, POP_MXBLCKS = 1.

When parallelizing on the SW26010P processor, POP acceleration can adopt two strategies:

first, parallelization based on the number of blocks. However, in order to maintain relatively balanced load and communication, the number of grid points per block in both directions is usually 20 to 40, resulting in a block count far fewer than 64 and insufficient to fully utilize the 64 CPEs in each Core Group (CG). Moreover, each CPE's Local Data Memory (LDM) is only 256 KB, so block-based parallelization may lead to LDM shortage. The second strategy is many-core parallelization based on grids within each block, which can better utilize the computational capacity of the CPEs. For our hardware architecture, we typically perform many-core parallelization along one dimension of the physical block or along the vertical dimension, and set POP_BLKCY close to but not exceeding 64 or its multiples, to ensure load balance while maximizing many-core acceleration performance.

The load balancing challenge of POP mainly arises from its latitude-longitude grid and land-sea distribution, where different processors are assigned varying numbers of active ocean grid points, resulting in load imbalance.

We have added explanations in the revised manuscript. Please see line 286-293.

Specific comment 2.6

Reviewer comment:

L257-261: This is not clear to me.

Response:

We thank the reviewer for raising this question. Regarding the PE layout method described in lines L257-261, we follow the load balancing procedure recommended in the CESM official manual. The manual notes that one of CESM's key advantages is allowing users to flexibly assign different components (such as CAM, POP, CICE, etc.) to different subsets of processors and control their sequential or concurrent execution to optimize overall efficiency. Our procedure is based on this guideline: first, an initial run is performed across all processors to evaluate the computational load and performance characteristics of each component; then, based on the performance data of computationally intensive components such as the atmosphere and ocean, the PE layout is adjusted to assign each component to appropriate processor subsets, forming a load-balanced configuration that maximizes overall throughput. We have clarified this method in the revised manuscript, please see line 294-304.

Specific comment 3

Reviewer comment:

Section 3.1:

L271-273: Is it correct that the dynamical core does not entail any vertical dependency?

L298-300: I don't have much experience with land-surface schemes, but the schemes I know involve many parameters and fields, but calculations are fast nonetheless since only a few layers into the soil are needed. Does this hold true for your surface parametrization as well?

Response:

Thank you for raising this important point. Indeed, the dynamical core has data dependencies in the vertical direction. Our original wording was not sufficiently precise, which may have caused some misunderstanding. Our intention was not to suggest that the dynamical core lacks vertical dependencies, but rather to highlight the differences in computational characteristics between the dynamical and physical components, emphasizing that the physical part tends to be computed more independently on a column-by-column basis. We will clarify this description in the revised manuscript to avoid any potential ambiguity.

Regarding the land surface parameterization, its computational cost accounts for a very small fraction of the overall simulation time and has negligible impact on the total computational performance, which is consistent with the characteristics mentioned by the reviewer. We have added content in the revised manuscript. Please see line 316-322.

Specific comment 4

Reviewer comment:

Section 3.3:

The meaning of technical terms like “swgcc”, “hybrid-LTO” and “SWACC” should be explained to engage a broader audience.

It is not clear to me how the xfort.py script is invoked. Since it takes the compiler arguments, does it act as a compiler wrapper invoked by the user, or it's rather used by the compiler under-the-hood?

Response:

Thank you for your insightful comment. To make the manuscript more accessible to a broader audience, we will provide explanations for the technical terms appearing in the text in the revised version. xfort.py is a script that encapsulates the O2ATH tool and is responsible for initiating the entire compilation workflow of O2ATH. We have clarified it in our revised manuscript, please see line 402-408.

Specific comment 5

Reviewer comment:

Section 3.4 - Figure 5: The authors present the speed-up provided by OpenMP+OATH compared to the fine-grained refactoring at the Athread-level. But which is the number of grid points? How the speed-ups vary with the grid size? Did you apply the fine-grained refactoring to other stencils as well? Moreover, I would expect the Athread optimizations to be performance-neutral at worst, while they can lead to performance regression on some stencils. How do you explain this?

Response:

We appreciate the reviewer's valuable feedback. The speedups shown in Figure 5 are based on relatively small-scale grids and compare the acceleration performance of O2ATH with Athread. As the grid size increases, the number of grid points assigned to each slave core remains roughly the same as in the small-scale case, so the speedup is almost unaffected by changes in the total number of grid points. The speedup comparison in Figure 5 covers only a subset of the stencil functions. We have applied similar fine-grained reconstruction optimizations to other functions as well, and we will clarify this point in the revised manuscript. Finally, in theory, the performance under Athread-level optimization should remain unchanged in the worst-case scenario. However, performance decline is observed for some stencils because these functions have deeply nested code structures. Manual hotspot-specific optimizations cannot cover all details, as only hotspot code is modified, whereas the automated tool O2ATH addresses a broader set of details, which can sometimes lead to localized performance regressions. We have added content in the revised manuscript. Please see line 445-451.

Specific comment 6

Reviewer comment:

Section 4.1:

Figure 6: Do the panels refer to the model configuration 5v3?

L463-465: The authors here say that 5v3 running on MPEs only achieves 1.7 SDPD. However, Figure 1 shows that the throughput is already 4.34 SDPD when using only 18000 processes.

Response:

We thank the reviewer for the careful reading and helpful comments. We would like to clarify that the results shown in Figure 6 correspond to the 5v3 configuration with a size of 600,000, and we have made this clearer in the revised manuscript, please see line 505 in the revised manuscript.

The performance of 5v3 under the MPE-only configuration was obtained from our own independent tests using 18,000 processes, yielding 1.59 SDPD after double checking. Since no performance optimizations were applied before the MPE-only tests, running at larger scales would result in prohibitively long execution times. Therefore, the results at 18,000 processes are used as the baseline to facilitate subsequent performance comparisons and optimization analysis.

Specific comment 7

Reviewer comment:

Section 4.2 - 7: although the use of CPEs in POP2 leads to a small gain in terms of time-to-solution (comparing the two rightmost bars), the throughput leaps from 131 SDPD to 222 SDPD. Is there a mistake in the plot, or am I missing something?

Response:

Thank you for pointing out the issue in the figure. Upon verification, the inconsistency between the SDPD improvement and the wall seconds bar chart in Section 4.2 is indeed due to a plotting error. We have corrected the relevant data and redraw the figures in the revised manuscript to ensure that the line and bar charts are consistent.

Specific comment 8

Reviewer comment:

Figure 8: In the weak scaling plots (panels (a) and (b)), the choice of the number of processes is not clear to me. If NE30 is run with 1800 processes, shouldn't NE120 use 7200 processes, i.e. 4-times more processes? And is the time-step kept fixed?

Response:

Thank you for raising this important point. The weak-scaling performance comparisons in Figure 8 are all based on the average runtime per single time step. From NE30 to NE120, the horizontal grid is extended 4 times in each dimension, resulting in a total 16-fold increase. Therefore, we used 28,800 processes for the NE120 configuration to match the increased computational workload. We will further clarify the correspondence between the number of

processes and the time step in the revised manuscript to ensure readers can accurately understand.