I have read the conflicting policies on open-source requirements between the authors and GMD. I hope this issue will be resolved soon.

Response: Thanks for your kind reminder. In our manuscript, we cited the model description about the EPICC-Model in "EPICC-Model Working Group.: Description and evaluation of the Emission and atmospheric Processes Integrated and Coupled Community (EPICC) Model version 1.0. Adv. Atmos. Sci., https://www.iapjournals.ac.cn/aas/article/doi/10.1007/s00376-025-4384-y", which the only author is "EPICC-Model Working Group" with the corresponding email, not the authors list or model developer members lists. In the user agreement of EPICC-Model, we are "strictly prohibited from transferring, selling, or sharing the EPICC-Model software or Zenodo account credentials, whether for profit or free of charge, to any third party", that is the reason why we can't upload the EPICC-Model codes into the zenodo with the model module "GPU-HADVPPM4HIP V1.0" (https://zenodo.org/records/16916413). And **the model users or anyone, who is interesting in this model, can email to Working-Group@EPICC-Model.cn and return the signed user agreement, and then he can get the model codes of EPICC-Model**.

Therefore, the model module "GPU-HADVPPM4HIP V1.0" we contributed had been uploaded to Zenodo (https://zenodo.org/records/16916413), which includes the codes and test datasets, and can be downloaded without restrictions, and provided the official download link (https://earthlab.iap.ac.cn/resdown/info_388.html) and the Zenodo download link (https://doi.org/10.5281/zenodo.17071574) for the EPICC-Model.

We believe this procedure aligns with GMD' s code and data policy, which states: "**Where the authors cannot, for reasons beyond their control, publicly archive part or all of the code and data associated with a paper, they must clearly state the restrictions. They must also provide confidential access to the code and data for the editor and reviewers in order to enable peer review.**" For details of the relevant policy, please refer to: https://www.geoscientific-model-development.net/policies/code_and_data_policy.html.

The manuscript presents a method to accelerate the EPICC air quality model (EPICC-Model v1.0) using China GPU-like accelerators. The authors port the advection module, one of the most computationally expensive components, onto the GPU to reduce the computational burden. They demonstrate that the offline module achieves several orders of

magnitude speed up relative to the CPU alone version and 1.5x faster when integrating GPU-HADVPPM4HP into the EPICC model. GPU acceleration of numerical models is in an early stage of development, and no mature GPU-enabled models are yet widely used in air quality forecasting. I highly appreciate the authors' technical effort, including rewriting the model from Fortran to C and then to a GPU language.

Response: We appreciate the editor for reviewing our manuscript and for the valuable suggestions, which we will address point by point in the following.

Major comments

- **Run the original model (CPU alone version) EPICC-GPU with the same inputs and configurations. Compare outputs (e.g., NOx, NH3, O3, PM2.5, and PM10) and calculate the relative differences between the two versions.**

Response: Thanks for the constructive comment. Accordingly, we have supplemented a set of comparative experiments. Using identical input data and model configurations, the differences in simulation results between the HIP-Opt2 version and the original Fortran version were evaluated in a real-world case study. The HIP-Opt2 version refers to the heterogeneous parallel version formed by coupling GPU-HADVPPM4HIP V1.0 into the EPICC-Model, along with optimizations in communication, thread and block coordinated indexing, and hybrid parallelization. In these comparative experiments, the model's input data and configurations were set as described in Section 4.1, with the simulation period covering 24 hours from 00:00 to 23:00 UTC on July 1, 2021.

Figures 1 and 2 present the simulated results of gases (e.g., HONO, SO_2, NO_2, NH_3) and aerosols (e.g., BC, $PM_{2.5}$, ASO_4, ANH_4) after a 24-hour integration by both the HIP-Opt2. and the original Fortran versions, along with the absolute errors (AEs) between the two model versions. As visually evident from the figures, the results from HIP-Opt2 after the 24-hour integration closely match those from the original Fortran version. For the vast majority of grid points, the AEs for gases and aerosols are within ±0.1 ppbV or ±0.1 $\mu g \cdot m^{-3}$.

To assess the scientific applicability of HIP-Opt2, we followed the methodologies of Wang et al. (2021) and Cao et al. (2024) by introducing two metrics: the root mean square error (RMSE) and the standard deviation (std). The ratio of RMSE to std was calculated to evaluate the scientific

usability of HIP-Opt2. Here, RMSE represents the error between HIP-Opt2 and the original Fortran version for different species, while std denotes the standard deviation of the respective species in the Fortran version. Taking NO_2 as an example, if the RMSE/std ratio is very small, it indicates that the computational deviation introduced by heterogeneous parallel acceleration is negligible compared to the intrinsic spatial variability of NO_2 itself. This suggests that such minor computational errors do not affect the model's utility in scientific research. Table 1 lists the RMSE, std, and their ratios for gases (e.g., HONO, SO_2, NO_2, NH_3) and aerosols (e.g., BC, $PM_{2.5}$, ASO_4, ANH_4). The RMSE/std ratios for these species range from 10^{-5}% to 10^{-2}%. Specifically, BC exhibits the smallest ratio at 4.8×10^{-5}%, while ASO_4 shows the largest ratio, yet only at 7.0×10^{-2}%. The RMSE/std ratios for all species in HIP-Opt2. are comparable to those reported by Cao et al. (2024), demonstrating that the simulation results from the HIP-Opt2 version are fully suitable for scientific research. We have modified this part in **lines 449-489**, which are as follows:

By coupling GPU-HADVPPM4HIP V1.0 into the EPICC-Model and implementing optimizations—such as reducing the communication frequency between CPUs and domestic GPUs, adopting thread and block coordinated indexing, and employing "MPI+HIP" hybrid parallelization—we developed the HIP-Opt2 version. Using identical input data and model configurations, the differences in simulation results between the HIP-Opt2 version and the original Fortran version were compared in a real-world case study. The model input data and configurations were set as described in Section 4.1, with the simulation period covering 24 hours from 00:00 to 23:00 UTC on July 1, 2021.

Figures 1 and 2 present the simulated concentrations of gases (e.g., HONO, SO_2, NO_2, NH_3) and aerosols (e.g., BC, $PM_{2.5}$, ASO_4, ANH_4) after a 24-hour integration by both the HIP-Opt2. and the original Fortran versions, along with the Absolute Errors (AEs) between the two model versions. As visually evident from the figures, the results from HIP-Opt2. after the 24-hour integration are in close agreement with those from the original Fortran version. For the vast majority of grid points, the AEs for gases and aerosols remain within ±0.1 ppbV or ±0.1 $\mu g \cdot m^{-3}$.

To further evaluate the suitability of HIP-Opt2. for scientific research, we followed the methodologies of Wang et al. (2021) and Cao et al. (2024) by introducing two metrics: the root mean square error (RMSE) and the standard deviation (std). The ratio of RMSE to std was calculated to quantify the scientific applicability of HIP-Opt2. Here, RMSE represents the error

between HIP-Opt2. and the original Fortran version for different species, while std denotes the standard deviation of each species in the Fortran version. Taking NO_2 as an example, a very small RMSE/std ratio indicates that the computational deviation introduced by heterogeneous parallel acceleration is negligible compared to the inherent spatial variability of NO_2. This implies that such minor computational errors do not compromise the model's utility in scientific research. Table 1 lists the RMSE, std, and their ratios for the aforementioned gases and aerosols. The RMSE/std ratios for these species range from 10^{-5}% to 10^{-2} %. Specifically, BC exhibits the smallest ratio at 4.8×10^{-5} %, while ASO_4 shows the largest ratio, yet only at 7.0×10^{-2} %. The RMSE/std ratios for all species in HIP-Opt2 are comparable to those reported by Cao et al. (2024), demonstrating that the simulation results from the HIP-Opt2 version are fully suitable for scientific applications.

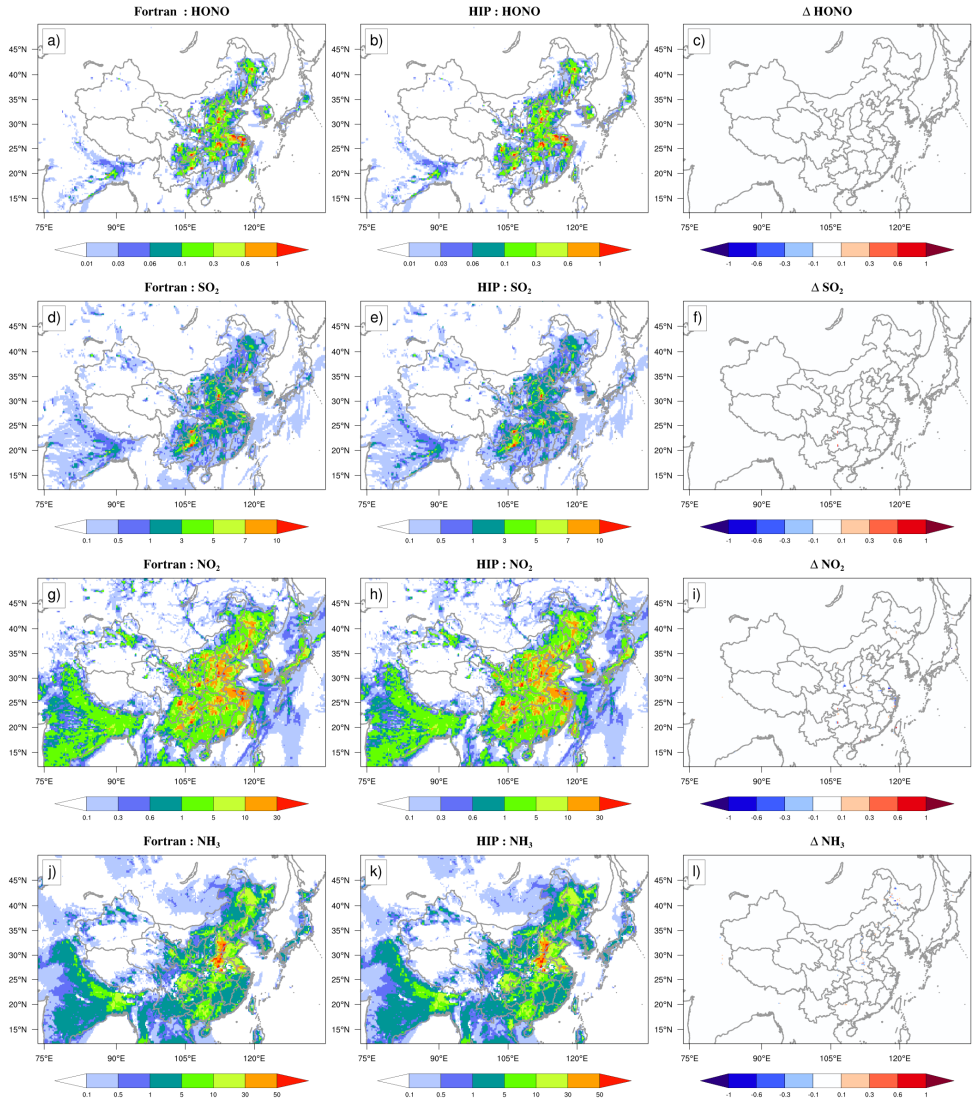

Figure 1. *HONO, SO_2, NO_2, and NH_3 concentrations outputted by the EPICC-Model for the*

Fortran and HIP-Opt2. versions. Panels (a), (d), (g), and (j) are from the Fortran version. Panels(b), (e), (h), and (k) are from the HIP-Opt2. version. Panels (c), (f), (i), and (l) are the absolute errors (AEs) between the Fortran and HIP-Opt2. Versions.

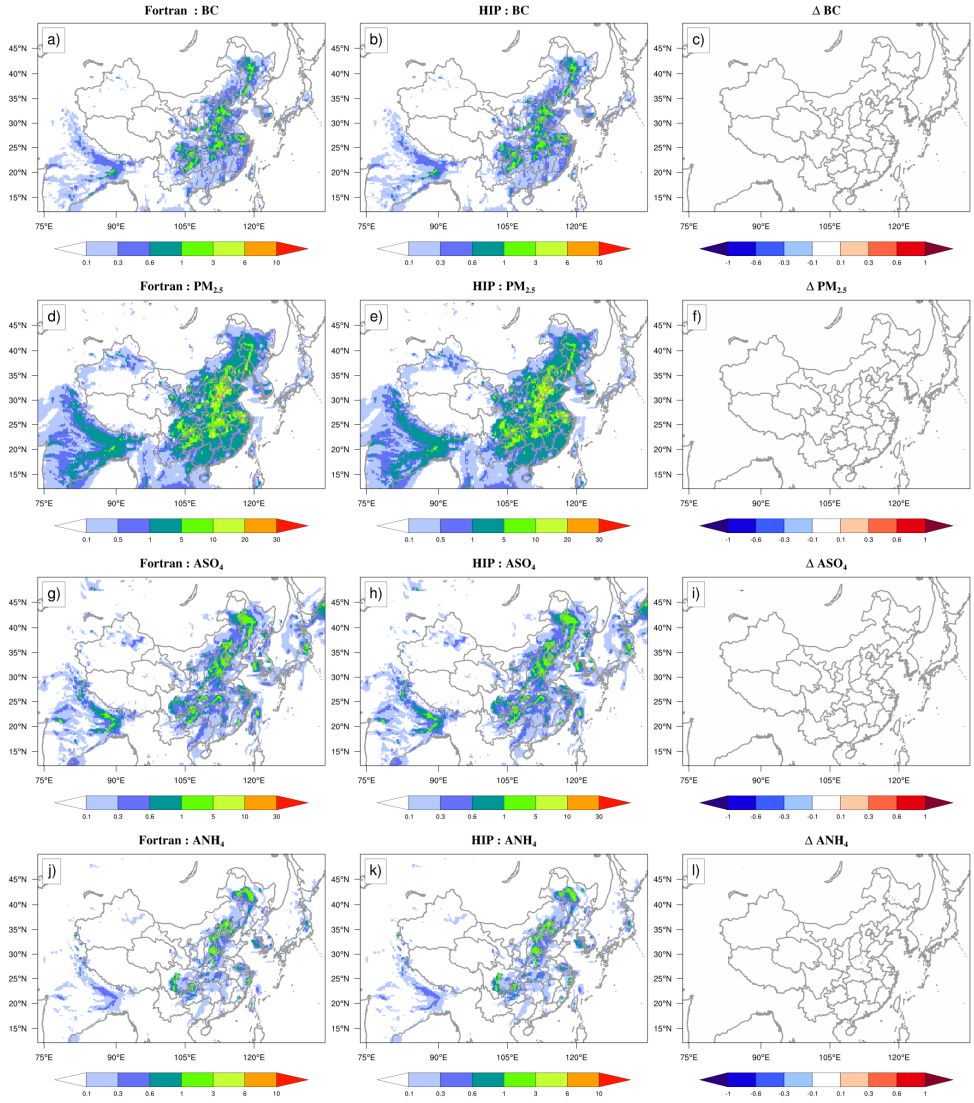

Figure 2. *BC, $PM_{2.5}$, ASO_4, and ANH_4 concentrations outputted by the EPICC-Model for the Fortran and HIP-Opt2. versions. Panels (a), (d), (g), and (j) are from the Fortran version. Panels(b), (e), (h), and (k) are from the HIP-Opt2. version. Panels (c), (f), (i), and (l) are the absolute errors (AEs) between the Fortran and HIP-Opt2 versions.*

Table 1. *The root mean square error (RMSE) between the Fortran and HIP-Opt2. versions, standard deviation (std) of the Fortran version, and the RMSE and std ratio.*

	RMSE	**std**	**RMSE/std (%)**
HONO (ppbV)	2.1×10^{-5}	0.1	2.1×10^{-2}

SO_2 (ppbV)	3.3×10^{-5}	0.7	4.5×10^{-3}
NO_2 (ppbV)	2.8×10^{-4}	3.4	8.3×10^{-3}
NH_3 (ppbV)	9.6×10^{-5}	4.1	2.4×10^{-3}
BC ($\mu g \cdot m^{-3}$)	9.7×10^{-8}	0.2	4.8×10^{-5}
$PM_{2.5}$ ($\mu g \cdot m^{-3}$)	4.4×10^{-6}	1.9	2.3×10^{-4}
ASO_4 ($\mu g \cdot m^{-3}$)	1.7×10^{-4}	0.2	7.0×10^{-2}
ANH_4 ($\mu g \cdot m^{-3}$)	1.1×10^{-4}	0.2	5.8×10^{-2}

- **Figure 5 shows that each CPU process and GPU handles a specific number of rows and columns. This raises concerns that the model may be hardcoded to a fixed domain. How can the current design be generalized to different grid definitions.**

Response: We apologize for any confusion caused by the imprecise expression. In the original Fortran version of EPICC-Model, the entire simulation domain is decomposed into different sub-domains using MPI, with each CPU process responsible for computing one sub-domain—including the integrations of the advection module. In this study, our work involves offloading the advection module computations, originally performed on the CPU after MPI decomposition, to China's domestic GPU-like accelerators through heterogeneous porting, thereby enabling parallel computation of the advection module on the GPUs. When different simulation domains or varying numbers of MPI processes are configured for the same domain, the size of the sub-domain handled by each CPU process after MPI decomposition differs. Consequently, the dimensions (i.e., the number of rows and columns) of the region for which the GPU is responsible in performing advection computations will also vary. To avoid ambiguity, we have revised the relevant description in **lines 341–366** as follows:

In the original Fortran version of EPICC-Model, the entire simulation domain is decomposed into subdomains using MPI, with each CPU process responsible for computations—including those of the advection module—within one subdomain. In this study, after MPI domain decomposition, we offloaded the advection module computations originally performed on the CPU to China's domestic GPUs through heterogeneous porting, thereby enabling parallel execution of the advection module on GPUs. Considering future high-resolution applications with large-scale data, assigning a single GPU to multiple CPU processes could lead to data-transfer contention and potential GPU memory overflow. Therefore, we adopted an "MPI+HIP" hybrid

parallelization approach, in which each participating CPU process is assigned a dedicated GPU accelerator. This design expands the parallel computing capacity of the model on heterogeneous clusters and makes full use of GPU resources. The implementation can be summarized in three main steps: (1) Obtain the MPI process rank information, as well as the number and indices of GPUs available on each compute node; (2) Based on the MPI process rank and using the remainder function in standard C, determine the number and indices of GPUs to be launched; (3) Map each MPI process to a specific GPU index, thereby realizing the "one CPU process – one GPU" configuration in the MPI+HIP hybrid parallel scheme.

As illustrated in Figure 5, taking the d02 domain configured in Section 4.1 with 8 CPU processes and 8 China's domestic GPU-like accelerators as an example: the EPICC model decomposes the simulation domain into 8 subdomains using the MPI software standard, with each CPU process handling its assigned subdomain. During the execution of the advection module, the "MPI+HIP" hybrid parallel scheme allocates one GPU accelerator to each CPU process. Computational tasks originally performed by the CPU are offloaded to the corresponding GPU; after the advection computations are completed, the results are returned to the CPU.

- **My main concern is the practical performance of the model. While the GPU port version clearly shows gains in offline and coupled tests. Its real world usefulness is questionable. In this paper, the authors use 10 CPU processes with 10 GPUs and report an overall performance improvement of only 1.5x. The cost of one GPU can exceed that of a 64 core CPU. Why not increase the number of CPUs instead, rather than pairing them with GPUs for such limited gain.**

Response: Your concern is highly valid and necessary. Indeed, there remains considerable room for improving the computational performance of the advection module on China's domestic GPU-like accelerators in this study. However, the task of porting and adapting key computational modules of numerical models to GPUs and accelerating them through heterogeneous parallelism is both urgent and imperative. First, limited by the heat dissipation capacity of transistors, the improvement in computational performance of CPU processors has slowed and is gradually approaching its physical limits. In recent years, however, GPU processors have continued to achieve significant performance gains due to their architectural advantages, and heterogeneous

supercomputing centered on "CPU+GPU" has become absolutely dominant among the world's leading high-performance computing systems. Taking the 66th TOP500 list published in November 2025 as an example, 9 of the top 10 supercomputers adopt heterogeneous architectures. Therefore, heterogeneous computing represents the primary direction for the future development of supercomputing. Nevertheless, the advancement of numerical models relies heavily on the support of supercomputing. The team that won the Gordon Bell Prize for Climate Modeling in November 2025 leveraged the powerful computational capabilities of NVIDIA GH200 GPUs to achieve the world's first 1.25 km ultra-high-resolution Earth system simulation (Klocke et al., 2025). To realize kilometer-scale ultra-high-resolution simulations of atmospheric pollution, it is both necessary and imperative for us to adapt air quality models to heterogeneous supercomputers and implement efficient parallel computing. Furthermore, both the offline performance tests in this study and the coupled performance tests in Cao et al. (2024) demonstrate that the larger the computational scale, the more pronounced the GPU acceleration becomes. This sufficiently proves the potential of GPUs in large-scale, high-resolution application scenarios. The simulation case configured in this study has a resolution of only 15 km over the whole of China, which may not yet fully demonstrate the parallel computing advantages of GPUs. In the future, higher-resolution simulation cases will be designed to better highlight the performance benefits of GPUs. Finally, in future work, we will employ additional optimization techniques to further enhance the computational performance of the advection module and other critical modules on China's domestic GPU-like accelerators. These include adopting asynchronous communication strategies to hide the communication overhead between CPUs and GPUs, configuring thread and block coordinated indexing to enable parallel computation of the model's three-dimensional grids, utilizing unified memory access, and implementing mixed-precision schemes. We have modified this part in **lines 732-781**, which are as follows:

There remains considerable room for improving the computational performance of GPU-HADVPPM4HIP V1.0 within the EPICC-Model in this study. First, constrained by the thermal dissipation limits of transistors, the growth in computational performance of CPU processors has slowed and is gradually approaching its physical limits. In recent years, however, GPU processors have continued to achieve substantial performance gains due to their architectural advantages, and heterogeneous supercomputing architectures centered on "CPU+GPU" now dominate the

landscape of advanced high-performance computing systems worldwide. Taking the 66th TOP500 list released in November 2025 as an example, 9 out of the top 10 supercomputers adopt heterogeneous architectures. Hence, heterogeneous computing represents the primary direction for the future development of supercomputing. That said, the advancement of numerical models relies fundamentally on the support of supercomputing capabilities. The team awarded the Gordon Bell Prize for Climate Modeling in November 2025 leveraged the powerful computational capacity of NVIDIA GH200 GPUs to accomplish the world's first 1.25 km ultra-high-resolution Earth system simulation (Klocke et al., 2025). To achieve kilometer-scale, ultra-high-resolution simulations of atmospheric pollution, it is both essential and imperative to adapt air quality models to heterogeneous supercomputing environments and enable efficient parallel computing.

Moreover, results from offline computational performance tests in this study and coupled performance tests in Cao et al. (2024) consistently show that the acceleration effect of GPUs becomes more pronounced as the computational scale increases, which sufficiently demonstrates the potential of GPUs in large-scale, high-resolution application scenarios. The simulation case configured in this study, with a resolution of only 15 km over the whole of China, may not yet fully reflect the parallel computing advantages of GPUs. In the future, higher-resolution simulation cases will be designed to better highlight the performance advantages of GPU acceleration.

Finally, in future work, we will employ further optimization techniques to enhance the computational performance of the advection module and other computationally intensive modules in the EPICC-Model on domestic heterogeneous clusters, including but not limited to:

(1) Firstly, priority should be given to optimizing CPU-GPU data transfer efficiency by reducing communication overhead through strategies such as unified memory architecture, asynchronous communication, mixed-precision methods, and minimizing non-essential variable I/O in air quality forecasting.

(2) Second, while GPU-accelerated modules including the gas-phase chemistry module (Cao et al., 2025) and advection module have been individually developed, their systematic integration into EPICC-Model requires architectural refinement to increase GPU code coverage. We will analyze the computational characteristics of the code in other modules of the model. For code segments involving iterative computations, we will first decouple the iterative computations by creating intermediate variables, thereby eliminating dependencies

between successive calculation steps. Subsequently, the computational code of other modules after iterative decoupling will be rewritten in the form of Kernel functions to increase the proportion of code executed on the GPU.

(3) *Finally, in current heterogeneous architecture supercomputing systems, the number of CPU processes within a computing node typically exceeds the number of GPUs. Employing the current matching scheme of one CPU process to one GPU accelerator results in the waste of remaining CPU computing resources. In the future, on one hand, while avoiding data transmission competition between the CPU and GPU, we will consider designing a more sophisticated mechanism for matching multiple CPU processes with a single GPU. On the other hand, drawing on Cao et al. (2024), we plan to introduce an OpenMP shared-memory parallel scheme into the EPICC-Model. Through multi-level hybrid parallelism, while porting computationally intensive modules to the GPU for parallel computing, other modules running on the CPU will utilize OpenMP multithreading parallelism, thereby fully leveraging CPU computing resources.*

■ **Currently there seems to be no straightforward solution.**

1. **The authors could load more computation (by porting more code) onto the GPU for a single memory allocation and copy. However, this approach introduces serial dependent computation within the kernel, reducing overall performance.**
2. **The authors could share one GPU across multiple processors, but this creates competition for host and device data transfer, which becomes a bottleneck.**

Response: Thanks for your constructive suggestions. Indeed, the efficiency of data transfer between the CPU and GPU is one of the most critical factors affecting the computational performance of numerical models on heterogeneous clusters. To improve the efficiency of data transfer between the CPU and GPU, this study reduces the communication frequency and increases the data transfer volume between them. However, relying solely on this optimization strategy is far from sufficient. In the future, on one hand, we will employ asynchronous communication strategies to hide part of the communication overhead; on the other hand, we will analyze the computational characteristics of other modules in the model. For code segments involving iterative computations, we will first decouple the iterative computations by introducing

intermediate variables, ensuring that successive calculation steps are independent of each other. Subsequently, the computational code of other modules after iterative decoupling will be restructured into Kernel functions. This approach not only increases the proportion of code executed on the GPU but also facilitates parallel computation on the GPU.

As for your mention of allocating one GPU accelerator to multiple CPU processes, in high-resolution application scenarios with large data scales, this approach may indeed lead to competition in data transfer between CPU and GPU and is prone to GPU memory overflow. Therefore, designing a more complex CPU-GPU matching mechanism is required to achieve pairing between multiple CPU processes and a single GPU. Currently, we opt for the relatively easier-to-implement scheme of matching a single CPU process with a single GPU. In the future, on one hand, while avoiding data transfer competition between CPU and GPU, we will consider designing a more sophisticated mechanism for matching multiple CPU processes with a single GPU. On the other hand, drawing on Cao et al. (2024), we plan to introduce an OpenMP shared-memory parallel scheme into the EPICC-Model. Through multi-level hybrid parallelism, while porting computationally intensive modules to the GPU for parallel computing, other modules running on the CPU will utilize OpenMP multithreading parallelism, thereby fully leveraging CPU computing resources. We have modified this part in **lines 762-781**, which are as follows:

Second, while GPU-accelerated modules including the gas-phase chemistry module (Cao et al., 2025) and advection module have been individually developed, their systematic integration into EPICC-Model requires architectural refinement to increase GPU code coverage. We will analyze the computational characteristics of the code in other modules of the model. For code segments involving iterative computations, we will first decouple the iterative computations by creating intermediate variables, thereby eliminating dependencies between successive calculation steps. Subsequently, the computational code of other modules after iterative decoupling will be rewritten in the form of Kernel functions to increase the proportion of code executed on the GPU.

Finally, in current heterogeneous architecture supercomputing systems, the number of CPU processes within a computing node typically exceeds the number of GPUs. Employing the current matching scheme of one CPU process to one GPU accelerator results in the waste of remaining CPU computing resources. In the future, on one hand, while avoiding data transmission competition between the CPU and GPU, we will consider designing a more sophisticated

mechanism for matching multiple CPU processes with a single GPU. On the other hand, drawing on Cao et al. (2024), we plan to introduce an OpenMP shared-memory parallel scheme into the EPICC-Model. Through multi-level hybrid parallelism, while porting computationally intensive modules to the GPU for parallel computing, other modules running on the CPU will utilize OpenMP multithreading parallelism, thereby fully leveraging CPU computing resources.

Significant benefits from GPU computing for numerical models may only be recognized once PCI bandwidth is substantially improved, or unified memory becomes more widely supported.

Response: Indeed, technologies such as unified memory and enhanced data transfer bandwidth can significantly improve the computational performance of air quality models on GPUs. To this end, NVIDIA Corporation, a prominent GPU manufacturer, has incorporated unified memory access functionality into its CUDA and introduced dedicated GPU interconnect technologies. These technologies utilize high-bandwidth, low-latency point-to-point connections, replacing traditional PCIe buses. Consequently, achieving efficient parallel computation of air quality models on GPU accelerators necessitates collaboration between researchers well-versed in the intrinsic physicochemical mechanisms of the models and engineers with expertise in GPU hardware and software. Herein, we appeal for the participation of engineers specializing in GPU software and hardware development to jointly advance the progress of air quality modeling. We have added this part in **lines 782-786**, which are as follows:

Consequently, achieving efficient parallel computation of air quality models on GPU accelerators necessitates collaboration between researchers well-versed in the intrinsic physicochemical mechanisms of the models and engineers with expertise in GPU hardware and software. Herein, we appeal for the participation of engineers specializing in GPU software and hardware development to jointly advance the progress of air quality modeling.

General comments

- **The authors list many performance metrics, such as 556.5x, 17.0x, 1.5x, 20.5x, and 39.3% in the abstract, but it is unclear what each is being compared against. This is confusing for readers.**

Response: Thanks for your valuable feedback. We acknowledge that our initial abstract did not clearly specify the baseline for the performance comparisons. In this study, after refactoring the Fortran code into C and adapting the advection solver program (GPU-HADVPPM4HIP) to a domestic GPU-like accelerator using the HIP heterogeneous programming interface, we first conducted offline computational performance tests. Identically sized random arrays were provided to GPU-HADVPPM4HIP and the Fortran-version HADVPPM advection program to compare the offline computational performance of the advection module on the CPU versus the domestic GPU-like accelerator. The results show that the acceleration effect of the advection module on the domestic GPU-like accelerator becomes more pronounced as the data size increases. Compared to the Fortran version compiled with the -O0 baseline option, GPU-HADVPPM4HIP compiled with the -O3 optimization option achieves a speed-up of 556.5x on the domestic GPU-like accelerator for a data size of 10^8. After coupling GPU-HADVPPM4HIP into EPICC-Model without any further optimization, the baseline version (HIP-Ori) exhibited extremely low computational efficiency, requiring approximately 5.3 hours to simulate 1 hour of integration. Therefore, we performed the communication and thread optimizations described in Sections 3.2 and 3.3, resulting in the HIP-Opt1 and HIP-Opt2 versions. Compared to HIP-Ori, HIP-Opt1 with communication optimization achieved a 17.0x performance improvement. Building upon HIP-Opt1, HIP-Opt2 with additional thread optimization further improved computational performance by 1.5x relative to HIP-Opt1. Finally, we compared the computational performance of the advection module between the original Fortran version of EPICC-Model and the HIP-Opt2 version in a real word simulation case. The results indicate that, excluding data transfer overhead between the CPU and GPU, the advection module achieves a 20.5× speed-up on the domestic GPU-like accelerator compared to the CPU. When data transfer overhead is taken into account, the computational efficiency of the advection module on the domestic GPU is improved by 39.3%. We have added this part in **lines 28-39**, which are as follows:

Offline benchmark results demonstrated that GPU-HADVPPM4HIP V1.0 achieved a maximum speedup of 556.5x on a GPU-like accelerator using the compiler optimization option compared to the Fortran HADVPPM baseline compiled option for a data size of 10^8. Integrating GPU-HADVPPM4HIP V1.0 into EPICC-Model V1.0 yielded three distinct versions: the initial HIP-based version (HIP-Ori), a version optimized for CPU and GPU communication frequency

(HIP-Opt1), and a further-optimized version employing a thread-block coordinated indexing strategy (HIP-Opt2). Compared to the HIP-Ori version, HIP-Opt1 achieved a model-level computational efficiency improvement of 17.0x. Building upon HIP-Opt1, HIP-Opt2 delivered an additional 1.5x enhancement in computational efficiency. At the module level, including CPU and GPU data transfer overhead, the GPU implementation improves computational efficiency of the advection module by 39.3%; when communication cost is excluded, the advection module attains a 20.5× acceleration relative to its CPU counterpart.

- **Line 146: Typo "initial condition" should be IC.**

Response: We apologize for the oversight. The abbreviation for "initial conditions" has been standardized to "IC" in the revised manuscript.

- **A critical requirement for porting modules onto GPUs is that loops must be vectorized and independent, with each iteration not dependent on previous ones. What methods do the authors use to verify that the loops in the direction module meet this requirement?**

Response: Your question is highly professional and valuable. Indeed, GPUs are well-suited for large-scale matrix parallel computations without data dependencies. Before porting the advection module to a domestic GPU for parallel computing, we first analyzed the computational characteristics of its code. We identified that during the final step of concentration update, the module does involve iterative computation, where the updated concentration on the left side of the equation depends on the concentration value on the right side, i.e., con(i) = con(i) - delta(i), where delta represents the concentration change due to the advection process, i represents different grids. To address this, we decoupled the iterative computation by introducing an intermediate variable. Prior to the concentration update, the con variable is copied to a new variable, concpy. The final update is then performed using concpy: con(i) = concpy(i) - delta(i). This approach of decoupling the iteration not only ensures computational correctness but also fully leverages the multi-thread parallel computing capability of the GPU. We have added this part in **lines 324-330**, which are as follows:

Furthermore, given that GPUs are well-suited for large-scale matrix parallel computations without data dependencies, prior to implementing parallel computation on a two-dimensional grid

using thread and block coordinated indexing, we decoupled the iterative computations present in the advection module by introducing intermediate variables. This ensures computational independence at each step, meaning that the computation of the next step does not depend on the results of the previous step, thereby fully leveraging the multi-thread parallel computing capability of the GPU.

- **Line 192: The text "as shown in Figure 2, the implementation of parallel computing ..." , does not align with the content of the figure.**

Response: We apologize for any confusion caused by our previous wording. In fact, regarding another computationally intensive module in the EPICC-Model for air quality—the gas-phase chemistry module—Cao et al. (2025) have implemented a 4th-order Rosenbrock solver for parallel computation on the China's domestic GPU-like accelerators. Performance tests show that the computational time fraction of the gas-phase chemistry module decreased from 45.7% in the original Fortran version to 29.9%. Along with this reduction in the computational share of the gas-phase chemistry module, the share of the advection module increased from 13.3% to 25.0%, making it the new computational hotspot. This shift is one of the key motivations for conducting GPU-based parallel acceleration of the advection module in this study. To avoid any inconsistency between the description and the figure content, we have updated Figure 2. Specifically, Figure 2(a) shows the computational time distribution among modules in the original Fortran version of EPICC-Model, where the gas-phase chemistry module accounts for 45.7% and the advection module for 13.3%. Figure 2(b) presents the distribution after Cao et al. (2025) implemented GPU parallelization of the gas-phase chemistry module: the share of the gas-phase chemistry module decreases from 45.7% to 29.9%, while the share of the advection module increases from 13.3% to 25.0%, establishing it as the new computational hotspot. We have modified this part in **lines 189-207**, which are as follows:

Figure 2 illustrates the changes in the computational time proportion among modules before and after the heterogeneous parallel implementation of the gas-phase chemistry module on a domestic GPU, as reported by Cao et al. (2025). Specifically, Figure 2(a) shows the time proportion of each module in the original Fortran version, while Figure 2(b) presents the corresponding proportion after the gas-phase chemistry module was ported for parallel

computing on the China's domestic GPU-like accelerator. As shown in Figure 2, the implementation of parallel computing for gas-phase chemistry modules on China's domestic GPU-like accelerator achieved significant efficiency improvements, reducing its computational time proportion from 45.7% to 29.9%. Notably, the computational time proportion of MPI_Barrier synchronization function has decreased from 23.9% to 13.4%. A critical observation emerged regarding the advection module, whose computational time proportion increased from 13.3% to 25.0%, establishing it as a new performance bottleneck comparable to the optimized chemistry module. This performance shift necessitates subsequent optimization efforts focusing on heterogeneous porting and parallel acceleration of the PPM scheme for China's domestic GPU-like architectures within the EPICC-Model V1.0 framework, aiming to enhance the computational efficiency of the advection module.

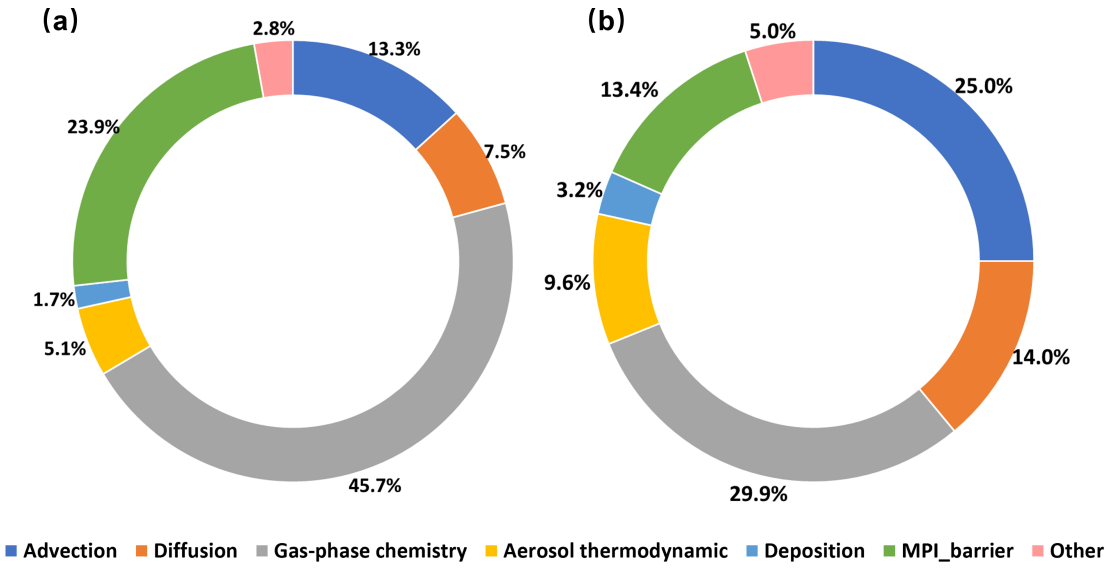

Figure 2. The computational time proportion among modules (a) for the Fortran version and (b) after implementing the gas-phase chemistry module on China' s domestic GPU-like accelerators for parallel computation.

■ **Line 360: The authors state that WRF is a state-of-the-art mesoscale numerical weather prediction. Note that NCAR has developed MPAS as the next generation model. Also, WRF v3.9.1 is used in this study, which is 8 years old. The latest version is 4.7.1.**

Response: Thanks for your valuable suggestion. Indeed, the Model for Prediction Across Scales

(MPAS) is a flagship next-generation numerical weather prediction model under active development at the National Center for Atmospheric Research (NCAR) in the United States. We have revised this part in **lines 383-387**, which are as follows:

Meteorological inputs are generated using the Weather Research and Forecasting (WRF, Skamarock et al., 2008) model, a mesoscale numerical weather prediction system, and is widely adopted in both theoretical research and operational forecasting. This study employed the WRF version 3.9.1, and the model domain configurations maintains identical nesting architecture and spatial coverage as the EPICC-Model.

■ **Which variables are evaluated in Table 4?**

Response: We sincerely apologize for the lack of clarity in our previous description. In the process of implementing the parallel computation of the advection module on domestic GPUs, we first refactored the Fortran source code of the HADVPPM program into C, and then utilized the HIP heterogeneous programming interface to adapt the advection module for domestic GPUs, resulting in GPU-HADVPPM4HIP. In this study, we first compared the offline computational result errors between the Fortran and C-language versions of the HADVPPM program (F-to-C), between the C-language version of HADVPPM and the GPU-HADVPPM4HIP program (C-to-HIP), and between the Fortran version of HADVPPM and the GPU-HADVPPM4HIP program (F-to-HIP). For the offline result comparison, a dedicated Fortran program was developed to generate identical input datasets, each consisting of 100 double-precision floating-point numbers. Therefore, the absolute errors (AE) and relative errors (RE) presented in Table 4 represent the average values computed from the 100 double-precision floating-point results produced by the Fortran, C-language, and HIP versions of the HADVPPM program after performing the advection solution computation under the given input conditions. We have revised this part in **lines 402-409**, which are as follows:

To ensure input consistency, a dedicated Fortran program was developed to generate identical input datasets, including 100 double-precision floating-point numbers, for all three implementations. Each implementation executed a complete advection integration computation, with subsequent output recording and analysis. Therefore, the absolute errors (AE) and relative errors (RE) presented in Table 4 represent the average values computed from the 100 double-

precision floating-point results produced by the original Fortran code, restructured standard C code, and HIP-accelerated code of the HADVPPM program after performing the advection solution computation under the given input conditions.

- **The term "data scale" requires clarification.**

Response: Thanks for your constructive suggestions. To conduct the offline computational performance testing, we also developed a Fortran program to generate random arrays of varying sizes. These arrays were then input into both the Fortran version of the HADVPPM program and the GPU-HADVPPM4HIP program, in order to evaluate the offline computational performance of the advection module on the CPU and the domestic GPU. These random arrays are one-dimensional and contain 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, and 10^8 random numbers, corresponding to the different scales (from 10^2 to 10^8) mentioned in the manuscript. The offline testing code has been uploaded to ZENODO (https://doi.org/10.5281/zenodo.16916413, Cao and Wu, 2025) and is available for download should you be interested. We have revised this part in **lines 553-556**, which are as follows:

To achieve this, Fortran-based test programs were implemented to generate randomized input arrays with varying scales for both Fortran and HIP versions of the HADVPPM program. These random arrays are one-dimensional and contain 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, and 10^8 random numbers.

- **Lines 614 – 618: The authors state that eliminating non-critical variables such as sea salt aerosols could accelerate the model. A well designed GPU port should retain the integrity of the original model. Common chemical mechanisms involve hundreds of species with reaction pathways. Omitting components may compromise scientific completeness.**

Response: Thanks for your valuable comment. The third-generation air quality models are designed based on the "One Atmosphere" concept, which treats atmospheric physical and chemical processes as an integrated whole. It would be unscientific to arbitrarily remove non-critical variables, such as sea salt, solely to improve model computational efficiency. Accordingly, we have removed the related statements from the manuscript.

Reference

Cao, K., Wu, Q., Wang, L., Guo, H., Wang, N., Cheng, H., Tang, X., Li, D., Liu, L., Li, D., Wu, H., and Wang, L.: GPU-HADVPPM4HIP V1.0: using the heterogeneous-compute interface for portability (HIP) to speed up the piecewise parabolic method in the CAMx (v6.10) air quality model on China's domestic GPU-like accelerator, Geosci. Model Dev., 17, 6887-6901, 10.5194/gmd-17-6887-2024, 2024.

Cao, K., Tang, X., Chen, H., Ma, J., Wu, Q., Wang, W., Chen, X., Li J., Wang, Z.: Porting and Parallel Optimization of the Gas-phase Chemistry Module of the Air Quality Model EPICC-Model on China’s Domestic Accelerators, Frontiers of Data&Computing, 10.11871/jfdc.issn.2096-742X.2025.05.010, 2025(in Chinese).

Klocke, D., Frauen, C., Engels, J. F., Alexeev, D., Redler, R., Schnur, R., Haak, H., Kornblueh, L., Brüggemann, N., Chegini, F., Römmer, M., Hoffmann, L., Griessbach, S., Bode, M., Coles, J., Gila, M., Sawyer, W., Calotoiu, A., Budanaz, Y., Mazumder, P., Copik, M., Weber, B., Herten, A., Bockelmann, H., Hoefler, T., Hohenegger, C., and Stevens, B.: Computing the Full Earth System at 1km Resolution, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 10.1145/3712285.3771789, 2025.

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Barker, D.M., Duda, M. G., Huang, X. Y., Wang, W., and Powers, J. G.: A Description of the Advanced Research WRF Version3 (No.NCAR/TN-475CSTR), University Corporation for Atmospheric Research, NCAR, https://doi.org/10.5065/D68S4MVH, 2008.

Wang, P., Jiang, J., Lin, P., Ding, M., Wei, J., Zhang, F., Zhao, L., Li, Y., Yu, Z., Zheng, W., Yu, Y., Chi, X., and Liu, H.: The GPU version of LASG/IAP Climate System Ocean Model version 3 (LICOM3) under the heterogeneous-compute interface for portability (HIP) framework and its large-scale application, Geosci.Model Dev., 14, 2781–2799, https://doi.org/10.5194/gmd-14-2781-2021, 2021.