Response to Reviewers' comments

We thank the two reviewers for their thoughtful and constructive comments that helped us substantially improve the manuscript. We have revised the manuscript accordingly. Listed below is our blue point-to-point response to each comment the reviewers offered.

Response to Reviewer #1

The authors have adequately addressed all comments and implemented a number of suggestions that have substantially improved the manuscript and presentation of results. I thus recommend that the manuscript is accepted for publication, subject to the call of the professional English language services for editortial review.

Response: Thanks a million for your precious time and your suggestion. To improve the English language of the manuscript, we have carefully corrected the grammatical errors in the paper. On the other hand, we will agree to apply the typesetting and language copy-editing service provided by Copernicus Publications during the production (https://www.geoscientific-model-development.net/policies/proofreading_guidelines.html). Moreover, two reviewers' point-to-point responses are listed below.

Response to Reviewer #2

The authors have addressed most of the points I raised.

Nevertheless, there are still two major not-fully-resolved points.

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

1- On the point of accuracy vs performance on NVIDIA hardware:

My main point of criticism was, in a nutshell, that the evidence provided (Figure 5) is not specific, thorough or deep enough to support the claim that the different range of errors in the species can be attributed to the hardware. The authors have included some comments on the tradeoffs between accuracy and computational performance, by invoking mixed precision techniques (which by their own account, they are not using), and now "speculate" that the differences compared to CPU computations can be explained by manufacturing processes resulting in unknown optimisation on GPU performance while renouncing accuracy. This is only a very minor toning-down of the previous claim. The authors also claim not to have sufficient expertise and knowledge to attribute this.

This is precisely the point. The claims are in my opinion still too bold. The authors seem to be sure that nothing else except hardware differences can explain the differences in the relative errors, which is a big leaf of faith from Figure 5. I would argue there are many things that can go wrong, and because there are basically two different codes at play, it is likely. At the very least, it cannot be simply assumed that nothing else except hardware is to blame.

An example: the authors point out that only the advection module is running on GPUs, and other modules on CPUs (including the chemistry). They also point out that for PNH4, PNO3 and PSO4 the initial concentrations are very low and they are generated via complex chemical reactions. I translate this as follows: the reactive part of the equations is very important. Presumably, reactions are computed in the chemistry module, which is always on the CPUs. Then, why would the GPUs be mostly responsible for these errors which are far away from machine precision? I find it hard to attribute such large errors to porting an advection solver to GPUs (in fact, my experience is quite the opposite, accuracy is very much comparable).

I unfortunately cannot take my own argument much further without making assumptions about the solvers and implementations. For example, if the chemical reactions lead to iterative linearisations and linear solvers, some convergence tolerances can play a role in there, which may be orders of magnitude above machine precision. This would not necessarily explain the difference between the Nvidia and the Chinese accelerators, but the errors could come from the different behaviours on the CPU.

I concede that all of this is speculation on my side, but I write it to illustrate the variety of thoughts one can have before attributing these differences to hardware. I do not intend to claim that it is NOT attributable to hardware, but proving that would require to define other types of tests, which offer much more control.

Response: Thanks for the constructive comment. To further detail the differences in the simulation results, we supplement the offline experimental results of the advection module on the NVIDIA K40m cluster and the Songshan supercomputer. First, we construct the Fortran programs to provide consistent input data for the advection module written in CUDA C code and HIP C code on NVIDIA Tesla K40m GPU and domestic GPU-like accelerator, respectively. The accuracy of the input data is kept at 12 decimal places. Then, the advection module outputs and prints the computing results after completing one integration operation on different accelerators. Finally, the results of the various accelerators were compared with those of the Fortran code on the Intel Xeon E5-2682 v4 CPU processor. The specific results are shown in the Fig.5 below. The difference in the computing results of the advection module written in HIP C code on the domestic GPU-like accelerator is smaller than that of the CUDA C code on the NVIDIA Tesla K40m GPU. The mean relative errors (REs) and AEs of the computing results on the NVIDIA Tesla K40m GPU are $1.3 \times 10^{-5}\%$ and 7.1×10^{-9}, respectively, while on the domestic GPU-like accelerator, the mean REs and AEs of the results are $5.4 \times 10^{-6}\%$ and 2.6×10^{-9}, respectively. The above codes related to offline tests are available online via ZENODO (https://zenodo.org/doi/10.5281/zenodo.10158214).

In the air quality model, the initial concentration of secondary fine particulate matter such as PSO_4, PNO_3, and PNH_4 is very low and is mainly generated by complex chemical reactions. The integration process of the advection module is ported from the CPU processor to the GPU accelerator, which will lead to minor differences in the results due to different hardware. The low initial concentration of secondary fine particulate matter is sensitive to these minor differences, which may eventually lead to a higher difference in the simulation results of secondary particulate matter than other species. We have modified this part in **lines 277-313**, which are as follows:

Lines 277-313:

To further detail the differences in the simulation results, we supplement the offline experimental results of the advection module on the NVIDIA K40m cluster and the Songshan supercomputer. First, we construct the Fortran programs to provide consistent input data for the advection module written in CUDA C code and HIP C code on NVIDIA Tesla K40m GPU and domestic GPU-like accelerator, respectively. The accuracy of the input data is kept at 12 decimal places. Then, the advection module outputs and prints the computing results after completing one integration operation on different accelerators. Finally, the results of the various accelerators were compared with those of the Fortran code on the Intel Xeon E5-2682 v4 CPU processor. The specific results are shown in the Fig.5. The difference in the computing results of the advection *module written in HIP C code on the domestic GPU-like accelerator is smaller than that of the CUDA C code on the NVIDIA Tesla K40m GPU. The mean relative errors (REs) and AEs of the computing results on the NVIDIA Tesla K40m GPU are* 1.3×10^{-5}*% and* 7.1×10^{-9}*, respectively, while on the domestic GPU-like accelerator, the mean REs and AEs of the results are* 5.4×10^{-6}*% and* 2.6×10^{-9}*, respectively.*

Fig.6 further presents the boxplot of the REs in all grid boxes for the PSO_4*,* PNO_3*,* PNH_4*,* O_3*, CO, and* NO_2 *during the 48-hour simulation under the BJ case. Statistically, the REs between the CUDA version on the NVIDIA K40m cluster and Fortran version on the Intel E5-2682 v4 CPU for the above six species are in the range of* $\pm$*0.006%,* $\pm$*0.01%,* $\pm$*0.008%,* $\pm$*0.002%,* $\pm$*0.002%, and* $\pm$*0.002%. In terms of REs between the HIP version on the "Songshan" supercomputer and the Fortran version on the Intel E5-2682 v4 CPU, the values are much smaller than REs between CUDA and Fortran versions which fall into the range of* $\pm$*0.0005%,* $\pm$*0.004%,* $\pm$*0.004%,* $\pm$*0.00006%,* $\pm$*0.00004%, and* $\pm$*0.00008%, respectively. In the air quality model, the initial concentration of secondary fine particulate matter such as* PSO_4*,* PNO_3*, and* PNH_4 *is very low and is mainly generated by complex chemical reactions. The integration process of the advection module is ported from the CPU processor to the GPU accelerator, which will lead to minor differences in the results due to different hardware. The low initial concentration of secondary fine particulate matter is sensitive to these minor differences, which may eventually lead to a higher difference in the simulation results of secondary particulate matter than other species.*

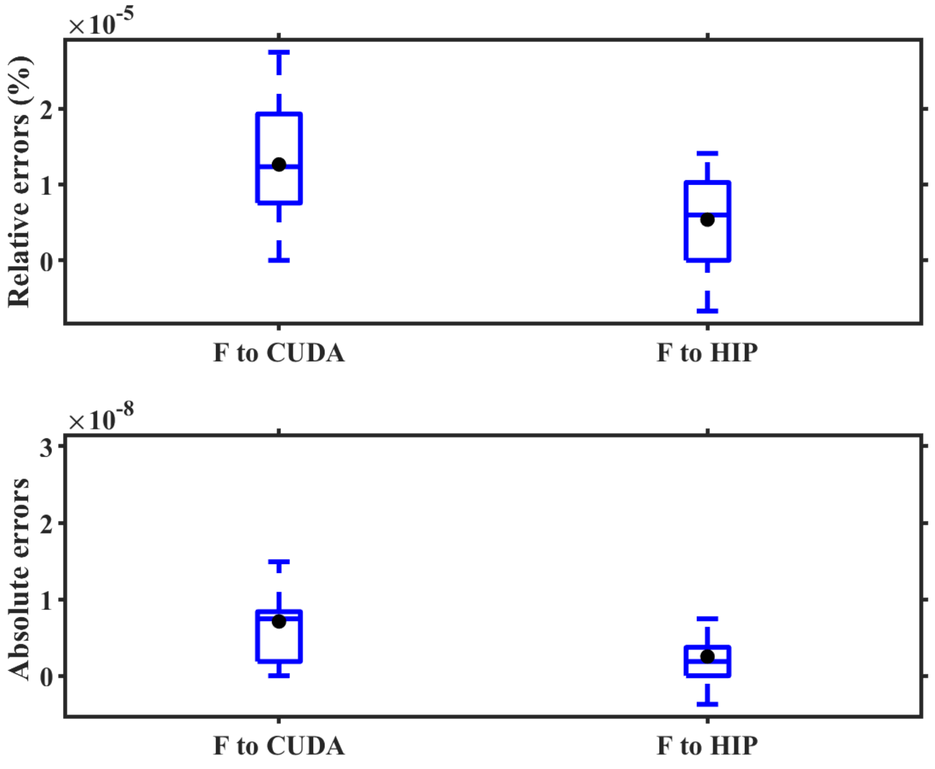

Figure 5. *The boxplots of REs and AEs between the Fortran code on Intel Xeon E5-2682 v4 CPU and CUDA C code on NVIDIA Tesla K40m GPU, and between HIP C code on domestic GPU-like accelerator, respectively, in the case of offline testing.*

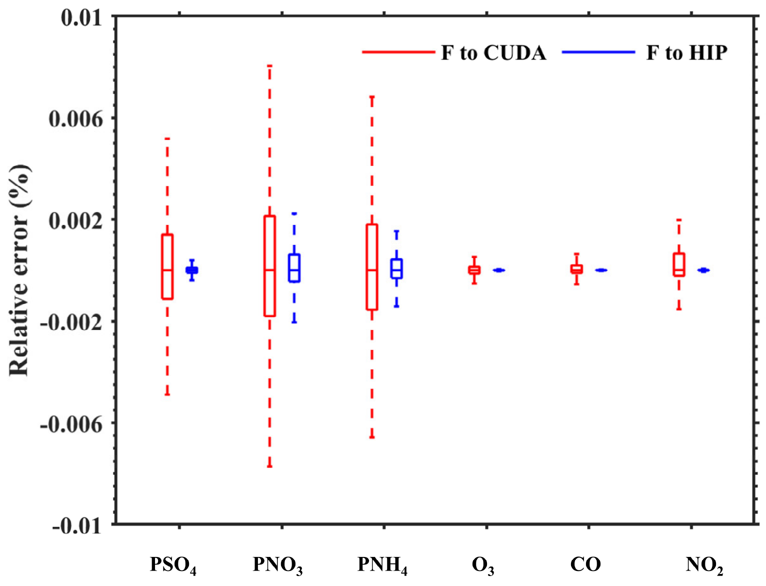

Figure 6. *The REs distribution in all grid boxes for the PSO_4, PNO_3, PNH_4, O_3, CO, and NO_2 under the BJ case. The red boxplot represents the REs between the CUDA version on the NVIDIA K40m cluster and the Fortran version on the Intel E5-2682 v4 CPU, and the blue boxplot represents the REs between the HIP version on the*

"Songshan" supercomputer and the Fortran version on the Intel E5-2682 v4 CPU.

2- this point is far simpler. It remains still unclear in section 4.3.1 exactly how many computational resources are used (e.g., how many cores for CPU computations, it seems only single GPUs). It might be good to simply state this very explicitly, very much how it is done in section 4.3.2. Nevertheless, in both sections explicitly stating how each module maps to which and how much hardware would be very much appreciated.

Response: Sorry for not being able to explain it clearly. We only used one CPU core when testing the computation performance of the advection module on the different GPU accelerator in section 4.3.1. For the MPI and HIP heterogeneous hybrid parallelization technology as described in section 4.3.2, we first use the ROCm-HIP library function hipGetDeviceCount to obtain the number of GPU accelerators configured for each compute node, and then the total number of accelerators to be launched and the ID number of accelerator cards in each node were determined according to the MPI process ID number and the remainder function in standard C language. Finally, the hipSetDevice library function in ROCm-HIP is used to configure an accelerator for each CPU core. During the simulation of the CAMx model, the computing of the emission, advection, dry deposition, diffusion, wet deposition, photolysis process and chemical process will be completed sequentially. In heterogeneous computing platforms, except for the advection process, the CPU processor completes the simulation of the rest of the processes, and the advection process is completed on the GPU accelerator. For example, using MPI and HIP hybrid parallel technology to launch four CPU processes and four GPU accelerators simultaneously, the advection process is completed on four GPU, and the other processes are still completed on four CPU processes. We have modified this part in **lines 182-192, lines 363-366 and lines 455-464**, which are as follows:

Linse 182-192:

As mentioned, the original CAMx model supports message passing interface (MPI) parallel technology running on the general-purpose CPU. The simulation domain is divided into several sub-regions by MPI, and each CPU process is responsible for the computation of its sub-region. To expand the heterogeneous parallel scale of the CAMx model on the Songshan supercomputer, a hybrid parallel architecture with an MPI and HIP was adopted to make full use of GPU *computing resources. Firstly, we use the ROCm-HIP library function hipGetDeviceCount to obtain the number of GPU accelerators configured for each compute node. Then, the total number of accelerators to be launched and the ID number of accelerator cards in each node were determined according to the MPI process ID number and the remainder function in standard C language. Finally, the hipSetDevice library function in ROCm-HIP is used to configure an accelerator for each CPU core.*

Lines 363-366:

When comparing the speedup on different GPU accelerators, the elapsed time of the advection module launched one CPU process (P1) on the domestic CPU processor A is taken as the benchmark; that is, the speedup is 1.0x. The runtime of the advection module on Intel CPU processor and different GPU accelerators is compared with the baseline to obtain the speedup.

Lines 455-464:

Generally, heterogeneous HPC systems have thousands of compute nodes equipped with one or more GPUs on each compute node. To fully use multiple GPUs, the hybrid parallelism with an MPI and HIP paradigm was used to implement the HIP version of GPU-HADVPPM run on multiple domestic GPU-like accelerators. During the simulation of the CAMx model, the emission, advection, dry deposition, diffusion, wet deposition, photolysis process, and chemical process will be computed sequentially. In heterogeneous computing platforms, except for the advection process, the CPU processor completes the simulation of the rest of the processes, and the advection process is completed on the GPU accelerator. For example, using MPI and HIP hybrid parallel technology to launch four CPU processes and four GPU accelerators simultaneously, the advection process is completed on four GPUs, and the other processes are still completed on four CPU processes.