General Comments

The manuscript gmd-2023-222 by Kai Cao et al. describes the porting of the Piecewise Parabolic Method numerical solver as part of the CAMx air quality model on heterogeneous computing architectures, and in particular on the new Chinese domestic accelerators, in comparison to NVIDIA GPUs. The resulting speedup and scaling behaviour are investigated. The study is relevant for the audience of the GMD journal and is timely in the context of the advent of GPU technologies in geoscientific model development.

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

The use of English is poor has to be improved throughout the manuscript. The text has to be revised and edited for syntax and grammar before it is reconsidered for publication.

Response: Thanks for the constructive comment. In order to improve the English language of the manuscript, we will call for the professional English language services to polish the manuscript after the public discussion.

Specific Comments:

The title is too long and may be confusing to the reader. In particular, it is not clear what "higher" model accuracy is with respect to. Please consider revising to "GPU-HADVPPM4HIP V1.0: using the heterogeneous 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"

Response: Thanks for the constructive suggestion. We will modify the title according to the suggestion in the revised manuscript.

Lines 53-60: the details of specific supercomputers are not needed and may be supefluous for the

reader. Propose to remove these lines.

Response: Thanks for the constructive suggestion. We have deleted the details of specific supercomputers in lines 48-54, which are as follows:

Lines 48-54:

The 61st edition of the top 10 list, released in June 2023, reveals that 80% of advanced supercomputers adopt the heterogeneous architectures (Top500, 2023), and the Frontier system equipped with AMD Instinct MI250X GPU at the Oak Ridge National Laboratory remains the only true exascale machine with the High-Performance Linpack benchmark (HPL) score of 1.194 Exaflop/s (News, 2023). How to realize the large-scale parallel computing and improve the computational performance of geoscience numerical models on the GPU has become one of the significant directions for the future development of numerical models.

L143-154: Is there a reference to the architecture of the Chinese heterogeneous cluster? In what ways to generations A and B differ? How do they compare to the NVIDIA clusters? These details are necessary for the reader to understand the potential and achieved performance comparison.

Response: Thanks for the constructive suggestion. In this study, two China' s domestically heterogeneous clusters are conducted the experiments. Two domestic heterogeneous clusters both include thousands of computing nodes and each containing one China' s domestically CPU processor, four China' s domestically GPU-like accelerators, and 128 GB of DDR4 2666 memory. The domestic CPU has 32 X86 based processors. The accelerator adopts a GPU-like architecture consisting of a 16 GB HBM2 device memory and many compute units. The GPU-like accelerators connected to CPU with PCI-E, the peak bandwidth of the data transfer between main memory and device memory is 16 GB/s. The domestic cluster B is the next generation of domestic cluster A, which has been updated in three main aspects: the CPU clock speed has been increased from 2.0 GHz to 2.5 GHz, the number of GPU-like computing units has been increased from 3,840 to 8,192, and the peak bandwidth between main memory and video memory has been increased from 16 GB/s to 32 GB/s. The domestic GPU-like accelerator is similar in architecture to the NVIDIA GPU, and

the CUDA core of the NVIDIA GPU essentially corresponds to the computing unit of the domestic GPU. In addition, the NVIDA GPU is programmed using the CUDA toolkit, and the domestic GPU-like is programmed using the ROCm-HIP toolkit (ROCm, 2023). We have modified this part in **lines 122-141**, which are as follows:

Lines 122-141:

Table 1 listed four GPU clusters which are conducted the experiments, two NVIDIA heterogeneous clusters which have the same hardware configuration as Cao et al. (2023) and two China's domestically heterogeneous clusters (domestic clusters) newly used in this research. Two NVIDIA heterogeneous clusters are equipped with NVIDIA Tesla K40m and V100 GPU accelerators, respectively. Both two domestic clusters include thousands of computing nodes and each containing one China's domestically CPU processor, four China's domestically GPU-like accelerators, and 128 GB of DDR4 2666 memory. The domestic CPU has 32 X86 based processors. The accelerator adopts a GPU-like architecture consisting of a 16 GB HBM2 device memory and many compute units. The GPU-like accelerators connected to CPU with PCI-E, the peak bandwidth of the data transfer between main memory and device memory is 16 GB/s.

It is worth noting that the domestic cluster B is the next generation of domestic cluster A, which has been updated in three main aspects: the CPU clock speed has been increased from 2.0 GHz to 2.5 GHz, the number of GPU-like computing units has been increased from 3,840 to 8,192, and the peak bandwidth between main memory and video memory has been increased from 16 GB/s to 32 GB/s. The domestic GPU-like accelerator is very similar in architecture to the NVIDIA GPU, and the CUDA core of the NVIDIA GPU essentially corresponds to the computing unit of the domestic GPU. In addition, the NVIDA GPU is programmed using the CUDA toolkit, and the domestic GPU-like is programmed using the ROCm-HIP toolkit. More details about the hardware composition and software environment of the four heterogeneous clusters are presented in Table 1.

Table 1 is unnecessary and need not be reproduced here. It's enough to refer to the HIP API.

Response: Thanks for the constructive suggestion. We have removed the Table 1 and modified this part in **lines 113-120**, which are as follows:

Lines 113-120:

In general, ROCm for the AMD GPU is equivalent to CUDA for NVIDIA GPU. On the ROCm software platform, it uses the AMD's HIP interface which is a C++ runtime API to allows developers to run programs on AMD GPUs. In general, it is very similar between the CUDA and HIP programming and their code can be converted directly by replacing the character "cuda" with "hip" in the most cases. More information about HIP API can be available on the AMD ROCm website (ROCm, 2023). Similar to AMD GPU, developers can also use ROCm-HIP programming interface to implement programs running on the China's domestically GPU-like accelerator.

Figure 1: It is not clear if the trancode happened from CUDA C to HIP C directly ("hipify" in the diagram) or the standard C code was changes (HIP arrow). The section does not document if any changes in the memory size/number of threads/blocks/kernels offloaded were necessary to support the GPU-like accelerator.

Response: Sorry for not being able to explain it clearly. There are two ways to implement the conversion of standard C code to HIP C code. Firstly, HIP technology is directly used to convert C code to HIP C code by adding related built-in functions (such as hipMalloc, hipMemcpy, hipFree, etc.). Secondly, based on the existing CUDA C code, the hipify toolkit was used to automatically replace the function names in the CUDA C code with the corresponding names in the HIP C code. The hipify toolkit is essentially a simple script written in the Perl language, and its function is text replacement, which replaces the function name in CUDA C code with the corresponding name HIP C code according to certain rules. For example, for the memory allocation function cudaMalloc in CUDA, the hipify toolkit can automatically recognize and replace it with hipMalloc. Therefore, the memory size/number of threads/blocks/kernels offloaded do not change when transcoding with the hipify toolkit. We have modified this part in **lines 151-167**, which are as follows:

Lines 151-167:

Fig. 1 shows the heterogeneous porting process of HADVPPM from CPU to NVIDIA GPU and domestic GPU-like accelerator. First, the original Fortran code was refactored using standard C

language. And then the CUDA and HIP technology were used to convert the standard C code into CUDA C and HIP C code to make it computable on the NIVIDA GPU and domestic GPU-like accelerator. Similar to CUDA technology, the HIP technology is implemented to convert the standard C code to HIP C code by adding related built-in functions (such as hipMalloc, hipMemcpy, hipFree, etc.). To facilitate the portability of applications across different GPU platforms, ROCm provides hipify toolkits to help transcode. The hipify toolkit is essentially a simple script written in the Perl language, and its function is text replacement, which replaces the function name in CUDA C code with the corresponding name in HIP C code according to certain rules. For example, for the memory allocation function cudaMalloc in CUDA, the hipify toolkit can automatically recognize and replace it with hipMalloc. Therefore, the thread and block configuration of GPU remain unchanged due to the simple text substitution during the transcoding. In this studying, the ROCm HIP technology was used to implement the operation of GPU-HADVPPM on domestic GPU-like accelerator based on the CUDA version of GPU-HADVPPM which was developed by Cao et al. (2023). During the compiling, the HIP code was compiled using the "hipcc" compiler driver with the library flag "-lamdhip64".

In Fig 2, the colour scale for the last two columns is generally too coarse and unsuitable to show any differences, other than a few points. Please consider revising. Are the concentrations for a specific time (for intance at the end of the run when any errors presumably accumulate)? Please give more detail.

Response: Thanks for the constructive suggestion. We reduced the color scale in the last two columns of Figure 2, and the absolute errors (AEs) between the Fortran version and CUDA version is more obvious than the HIP and Fortran version. In addition, considering the AEs accumulation and growth, Figure 3 highlights the time series of AEs between Fortran and CUDA versions and between Fortran and HIP versions after grid averaging. As is shown in Figure 3, the AEs of O_3, PSO_4, CO, and NO_2 between the Fortran version and the CUDA version are -0.0002 to 0.0001 ppbV, -0.00003 to 0.00001 $\mu g \cdot m^{-3}$, -0.0004 to 0.0004 ppbV, and -0.0002 to 0.0002 ppbV, respectively, and fluctuate. Although the AEs of the above four species between the Fortran and the HIP version also fluctuates, the fluctuation range is much smaller than that of the CUDA version. Importantly,

the AEs between Fortran and CUDA versions and between Fortran and HIP versions both do not accumulate and grow over prolonged simulation periods. We have modified this part in **lines 224-243**, which are as follows:

Lines 224-243:

Fig. 2 present the four major species simulation results of three CAMx version, including Fortran version on the Intel E5-2682 v4 CPU, CUDA version on the NVIDIA K40m cluster and HIP version on the domestic cluster A, after 48 hours integration, as well as the absolute errors (AEs) of their concentrations. The species' spatial pattern of three CAMx versions on different platform are visually very consistent, and the AEs between the HIP and Fortran version is much smaller than the CUDA and Fortran version. For example, the AEs between the CUDA and Fortran version for O_3*,* PSO_4*, and* NO_2 *are in the range of* ± 0.04 *ppbV,* ± 0.02 $\mu g \cdot m^{-3}$*, and* ± 0.04 *ppbV. And the AEs between the HIP and Fortran version for above the three species are fall into the range of* ± 0.01 *ppbV,* ± 0.005 $\mu g \cdot m^{-3}$*, and* ± 0.01 *ppbV. For CO, AEs is relatively large due to its high background concentration. However, the AEs between the HIP and Fortran versions is also less than that between the CUDA and Fortran versions where were in the range of* ± 0.4 *ppbV and* ± 0.1 *ppbV, respectively. Considering the situation of AEs accumulate and grow, Figure 3 highlights the time series of AEs between Fortran and CUDA versions and between Fortran and HIP versions after grid averaging. As is shown in Figure 3, the AEs of* O_3*,* PSO_4*, CO, and* NO_2 *between the Fortran version and the CUDA version are -0.0002 to 0.0001 ppbV, -0.00003 to 0.00001* $\mu g \cdot m^{-3}$*, -0.0004 to 0.0004 ppbV, and -0.0002 to 0.0002 ppbV, respectively, and fluctuate. Although the AEs of the above four species between the Fortran and the HIP version also fluctuates, the fluctuation range is much smaller than that of the CUDA version. Importantly, the AEs between Fortran and CUDA versions and between Fortran and HIP versions both do not accumulate and grow over prolonged simulation periods.*

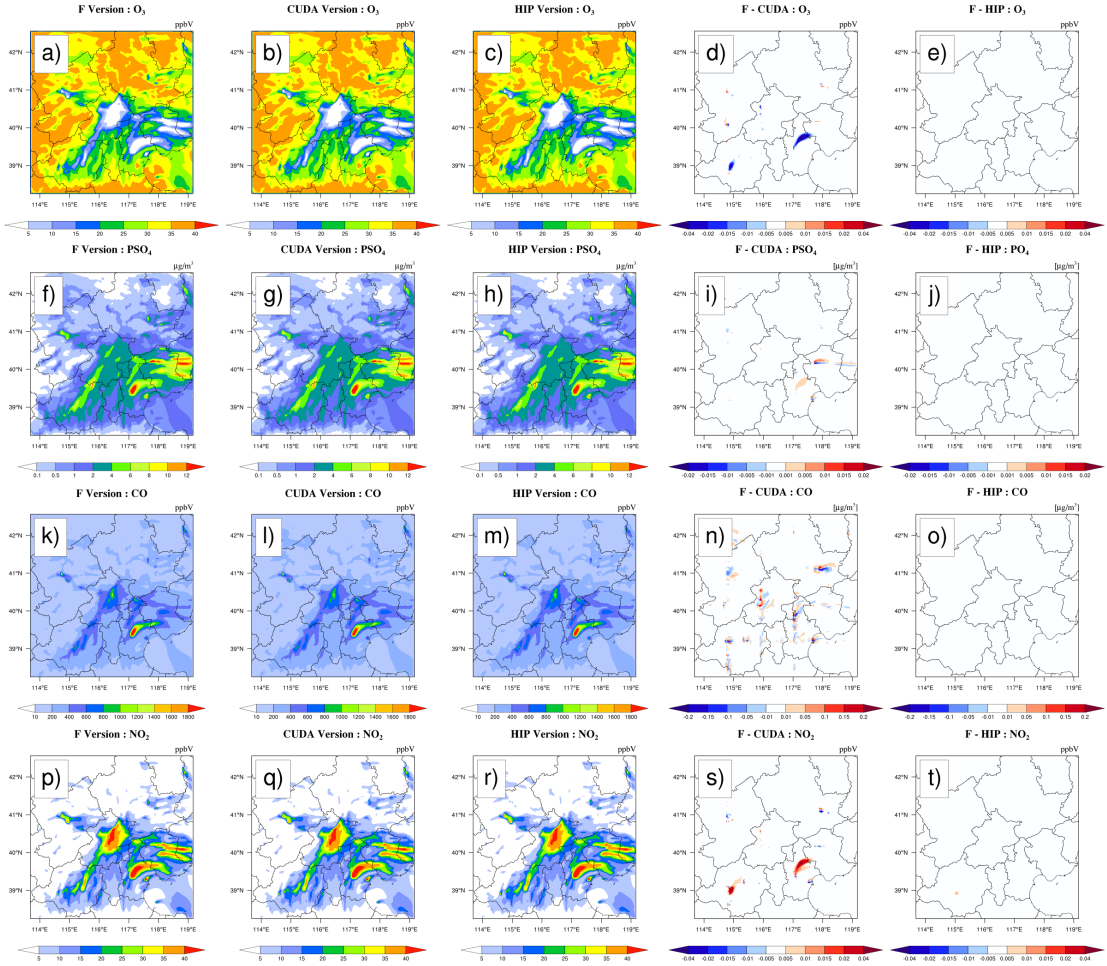

Figure 2. O_3, PSO_4, CO, and NO_2 concentrations outputted by the CAMx Fortran version on the Intel E5-2682 v4 CPU, CUDA version on the NVIDIA K40m cluster and HIP version on the domestic cluster A under the BJ case. Panels (a), (f), (k), and (p) are from the Fortran version of simulation results for four species. Panels (b), (g), (l), and (q) are from the CUDA version of simulation results for four species. Panels (c), (h), (m), and (r) are from the HIP version of simulation results for four species. Panels (d), (i), (n), and (s) are the AEs between the Fortran and CUDA versions. Panels (e), (j), (o), and (t) are the AEs between the Fortran and HIP versions.

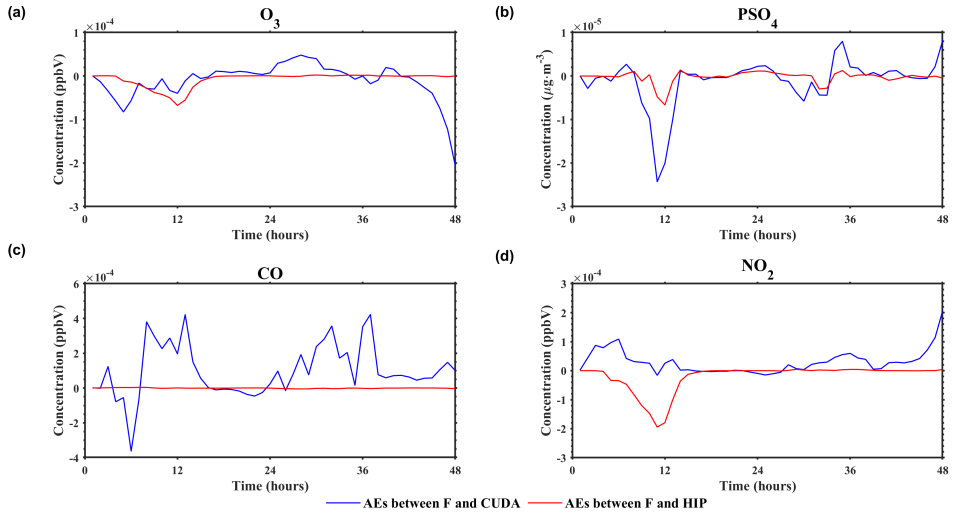

Figure 3. The time series of AEs between Fortran and CUDA versions (solid blue line) and between Fortran and HIP versions (solid red line) after grid averaging. Panel (a)~(d) represent the AEs of O_3, PSO_4, CO, and NO_2, respectively.

L. 297-290: What does it mean that the NVIDIA GPU sacrifices part of the accuracy for improved computing performance? Is this for floating point representation and/or arithmetic? Is there no user option (e.g. optimisation levels at compile time) to control the accuracy? Please elaborate.

Response: Sorry for not being able to explain it clearly. We have modified the above statement as "NVIDIA GPU loss part of the precision to improve the computing performance". There is a strong correlation between processor performance and precision. In the field of geoscience numerical models, many model optimization works improve the computational performance by reducing the precision of the data, such as Váňa et al. (2017) changed some variables precision in the atmospheric model from double precision to single precision, which increased the overall computational efficiency by 40%, and Wang et al. (2019) improved the computational efficiency of the gas-phase chemistry module in the air quality mode by 25%~28% by modifying the floating-point precision compile flag. In this study, GPU-HADVPPM4HIP has less bias on the domestic GPU-like accelerator. We infer that this may be related to the manufacturing process of NVIDIA GPUs, especially NIVIDA Tesla V100 series GPUs, which may use unknown optimizations to improve GPU performance efficiency by losing part of the accuracy. We also want to know the specific reasons for this, but we are not professional GPU research and development designers after all, so

we can only present our experimental results in the air pollution model to you, and discuss with each other to jointly promote the application of GPU in the field of geoscience numerical models. We have modified this part in **lines 282-298**, which are as follows:

Lines 282-298:

From AEs, REs, and ratio of RMSE and std between different CAMx versions, it can be identified that the GPU-HADVPPM4HIP program runs on domestic cluster A with less difference, and the reason for this difference may be related to the fact that the NVIDIA GPU loss part of the precision to improve the computing performance. There is a strong correlation between processor performance and precision. In the field of geoscience numerical models, many model optimization works improve the computational performance by reducing the precision of the data, such as Váňa et al. (2017) changed some variables precision in the atmospheric model from double precision to single precision, which increased the overall computational efficiency by 40%, and Wang et al. (2019) improved the computational efficiency of the gas-phase chemistry module in the air quality mode by 25%~28% by modifying the floating-point precision compile flag. In this study, GPU-HADVPPM4HIP has less bias on the domestic GPU-like accelerator. We infer that this may be related to the manufacturing process of NVIDIA GPUs, especially NIVIDA Tesla V100 series GPUs, which may use unknown optimizations to improve GPU performance efficiency by losing part of the accuracy. We also want to know the specific reasons for this, but we are not professional GPU research and development designers after all, so we can only present our experimental results in the air pollution model to you, and discuss with each other to jointly promote the application of GPU in the field of geoscience numerical models.

Sec. 4.3.1: Given the large time spend to transfer the memory to and from the accelerator, wouldn't a more relevant comparison for any real-world application be of the total required time for the completion of the air quality simulation rather than the compute time of the numerical kernel? Was there any consideration on how to limit the required memory size/bandwidth?

Response: Thanks for the constructive suggestion. Table 5 further lists the total elapsed time of CAMx Fortran and HIP versions for BJ case in domestic clusters A and B, and the computing time

of advection module with or without data transfer. By coupling the GPU-HADVPPM4HIP to CAMx model and adopting a series of optimizations including communication optimization, memory access optimization, and 2D thread optimization (Cao et al.,2023), the overall computation time of CAMx-HIP model on a single domestic GPU-like accelerator is faster than that of the original Fortran version on a single domestic CPU core. For example, on domestic cluster A, one hour of simulation in CAMx-HIP model takes 469 seconds, and the Fortran version takes 481 seconds. In the domestic cluster B, the acceleration effect is more obvious due to the upgrade of hardware and network bandwidth, and the integration time of CAMx-HIP model is 433 seconds when maintaining the same software environment, and the integration time of the Fortran version is 453 seconds. Video memory and bandwidth are the two most significant factors affecting GPU performance, and high video memory and high bandwidth can better play the powerful computing performance of GPUs. Usually, the memory and bandwidth of the GPU are already given at the factory. In this case, the amount of data transferred to the GPU can be roughly estimated before the data is transferred to the GPU, and the amount of data transferred to the GPU can be adjusted according to the size of the GPU memory to ensure that the amount of data transferred to the GPU each time reaches the maximum GPU video memory, so as to give full play to the GPU performance more efficiently. We have modified this part in **lines 340-379**, which are as follows:

Lines 340-379:

Table 5 further lists the total elapsed time of CAMx Fortran and HIP versions for BJ case in domestic clusters A and B, and the computing time of advection module with or without data transfer. By coupling the GPU-HADVPPM4HIP to CAMx model and adopting a series of optimizations such as communication optimization, memory access optimization, and 2D thread optimization, the overall computation time of CAMx-HIP model on a single domestic GPU-like accelerator is faster than that of the original Fortran version on a single domestic CPU core. For example, on domestic cluster A, one hour of simulation in CAMx-HIP model takes 469 seconds, and the Fortran version takes 481 seconds. In the domestic cluster B, the acceleration effect is more obvious due to the upgrade of hardware and network bandwidth, and the integration time of CAMx-HIP model is 433 seconds when maintaining the same software environment, and the integration time of the Fortran version is 453 seconds.

The elapsed time of GPU-HADVPPM given in Table 4 on NVIDIA GPU and domestic GPU-like accelerator does not consider the data transfer time between CPU and GPU. However, the communication bandwidth of data transfer between the CPU and GPU is one of the most significant factors that restrict the performance of numerical model on the heterogeneous cluster (Mielikainen et al., 2012; Mielikainen et al., 2013; Huang et al., 2013). To exhibit the significant impact of CPU-GPU data transfer efficiency, the coupled computing performance of GPU-HADVPPM4HIP with and without data transfer time for the BJ case is tested on the domestic cluster A and B with the same DTK version 23.04 software environment and the results are further presented in Table 6. The elapsed time of GPU-HADVPPM4HIP on domestic GPU-like accelerator A with and without taking into account the data transfer time between CPU and GPU-like accelerator are 6.8 and 29.8 seconds, respectively. In other words, it only takes 6.8 seconds to complete the computation on the domestic accelerator, but it takes 23.0 seconds to complete the data transfer between the CPU and the domestic GPU-like accelerator, which is 3.4 times the computation time. The same problem exists in the more advanced domestic cluster B, where the GPU-HADVPPM4HIP takes only 5.7 seconds to complete the computation, while the data transmission takes 18.2 seconds, which is 3.2 times the computation time.

Video memory and bandwidth are the two most significant factors affecting GPU performance, and high video memory and high bandwidth can better play the powerful computing performance of GPUs. Usually, the memory and bandwidth of the GPU are already given at the factory. In this case, the amount of data transferred to the GPU can be roughly estimated before the data is transferred to the GPU, and the amount of data transferred to the GPU can be adjusted according to the size of the GPU memory to ensure that the amount of data transferred to the GPU each time reaches the maximum GPU video memory, so as to give full play to the GPU performance more efficiently. In addition, the computation time of advection module only accounts for about 10% of the total time of CAMx model (Cao et al., 2023). In the future work, the whole integral module of CAMx model except I/O can be further ported to GPU to minimize the frequency of data transfer between CPU-GPU and improve the efficiency of data transmission.

Table 5. The total elapsed time of CAMx Fortran and HIP versions for BJ case in domestic clusters A and B, and the computing time of advection module with or without data transfer. The unit of elapsed time is in seconds (s).

	China' s domestically cluster A		China' s domestically cluster B	
	Fortran version	HIP version	Fortran version	HIP version
Total elapsed time	481.0	469.0	453.0	433.0
Computing time of advection module without data transfer	57.8	6.8	47.8	5.7
Computing time of advection module with data transfer	57.8	29.8	47.8	23.9

Sec. 4.3.2: Are these results for the total model, or just the accelerated portion? How many timesteps is 1-hour of simulation? It would be generally interesting to see what the average speedup per timestep. It is hard to judge what the overall impact of the accelerator is, given that also the number of CPU cores/processes is also increasing. It would be good to conduct and add a scaling test purely on the CPU cores (ie. without acceleration) to isolate the speedup due to acceleration. Finally, where is the speedup performance saturation (e.g. for the BJ case) attributed for large core/cards counts?

Response: Thanks for the constructive suggestion. All the test results in Sec. 4.3.2 are the total elapsed time for one hour simulation. Typically, super-large heterogeneous clusters contain thousands of compute nodes, and one or more GPU accelerators are configured for each compute node. In order to expand the parallel scale of CAMx-HIP model in domestic heterogeneous clusters and improve GPU-like accelerator utilization, we adopt the "MPI+HIP" hybrid parallel scheme, that is, a GPU-like accelerator is configured for each CPU core during the simulation, and the parallel scalability of the three test cases is shown in Figure 5. For the BJ and HN case, the parallel scalability is highest when configured with 24 CPU cores and 24 GPU-like accelerators, with speedup of 8.1x and 11.6x, respectively. In terms of the ZY case, due to its large number of grids, the parallel scalability is the highest when 32 CPU cores and 24 GPU cards are configured, and the acceleration ratio is 17.2x.

Data transfer between CPU and GPU takes several times more time than computation. Regardless of the CPU-GPU data transfer consumption, GPU-HADVPPM4HIP can achieve up to 28.9x speedup on a single domestic GPU-like accelerator. However, in terms of the total time

consumption, the CAMx-HIP model is only 10~20 seconds faster than the original Fortran version when one GPU-like accelerator is configured. And as the number of CPU cores and GPU-like accelerators increases, the overall computing performance of CAMx-HIP model is lower than that of the original Fortran version. The main reason is related to the amount of data transferred to GPU. As the number of MPI processes increases, the number of grids responsible for each process decreases, and the amount of data transmitted by the advection module from CPU to GPU decreases. However, GPUs are suitable for large-scale matrix computing. When the data scale is small, the performance of GPU is low, and the communication efficiency between CPU-GPU is the biggest bottleneck (Cao et al., 2023). Therefore, the computational performance of CAMx-HIP model is not as good as the original Fortran version when MPI processes increase. According to the characteristics of GPUs suitable for large-scale matrix computing, the model domain can be expanded and the model resolution can be increased in the future to ensure that the amount of data transferred to each GPU reaches the maximum video memory occupation, so as to make efficient use of GPU. In addition, the advection module only accounts for about 10% of the total time consumption in CAMx model (Cao et al., 2023), and in the future, it is considered to port the entire integration module except I/O to the GPU to minimize the communication frequency.

The timestep of BJ, HN and ZY case were 59, 47, and 61, respectively. Figure 6 shows the GPU-HADVPPM4HIP acceleration in each time step on a single GPU-like accelerator. It can be seen from the figure that all three cases have the smallest speedup of 8.2x, 11.2x, and 27.8x at the first timestep, which is related to the time required for GPU-like accelerator startup. When the GPU-like is started and operating normally, the speedup of the three cases tend to be stable in the following time steps, and stabilize around 8.5x, 11.5x and 28.0x respectively. We have modified this part in **lines 331-339** and **lines 382-414**, which are as follows:

Lines 331-339:

The timestep of BJ, HN and ZY case were 59, 47, and 61, respectively. Fig. 5 shows the GPU-HADVPPM4HIP acceleration in each time step on a single domestic GPU-like accelerator A. It can be seen from the figure that all three cases have the smallest speedup of 8.2x, 11.2x, and 27.8x at the first timestep, which is related to the time required for GPU-like accelerator startup. When the GPU-like is started and operating normally, the speedup of the three cases tend to be stable in

the following time steps, and stabilize around 8.5x, 11.5x and 28.0x respectively.

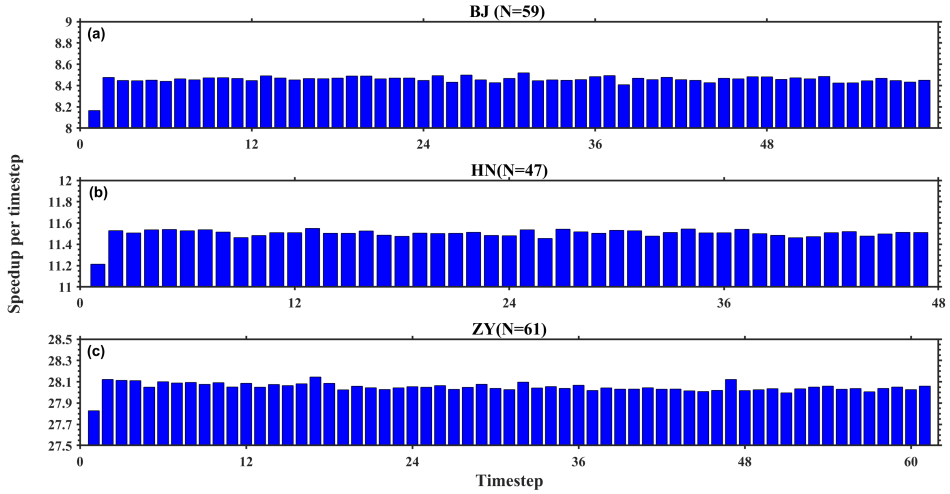

Figure 5. The GPU-HADVPPM4HIP acceleration in each time step on a single GPU-like accelerator for BJ, HN, and ZY case, and the timestep of three cases are 59, 47, and 61, respectively.

Lines 382-414:

Generally, the super-large heterogeneous clusters have thousands of compute nodes which are equipped with one or more GPUs on each node. To make full use of multiple GPUs, a parallel architecture with an MPI and CUDA hybrid paradigm was implemented to improve the overall computational performance of CAMx-CUDA model (Cao et al., 2023). In this studying, 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. Fig.6 shows the total elapsed time and speedup of CAMx-HIP model which coupled with the HIP version GPU-HADVPPM on the domestic cluster A under the BJ, HN, and ZY cases. The simulation of above three cases for one hour took 488 seconds, 1135 seconds and 5691 seconds respectively when launching two domestic CPU processors and two GPU-like accelerators. For the BJ and HN case, the parallel scalability is highest when configured with 24 CPU cores and 24 GPU-like accelerators, with speedup of 8.1x and 11.6x, respectively. In terms of the ZY case, due to its large number of grids, the parallel scalability is the highest when 32 CPU cores and 24 GPU cards are configured, and the acceleration ratio is 17.2x.

Data transfer between CPU and GPU takes several times more time than computation.

Regardless of the CPU-GPU data transfer consumption, GPU-HADVPPM4HIP can achieve up to 28.9x speedup on a single domestic GPU-like accelerator. However, in terms of the total time consumption, the CAMx-HIP model is only 10~20 seconds faster than the original Fortran version when one GPU-like accelerator is configured. And as the number of CPU cores and GPU-like accelerators increases, the overall computing performance of CAMx-HIP model is lower than that of the original Fortran version. The main reason is related to the amount of data transferred to GPU. As the number of MPI processes increases, the number of grids responsible for each process decreases, and the amount of data transmitted by the advection module from CPU to GPU decreases. However, GPUs are suitable for large-scale matrix computing. When the data scale is small, the performance of GPU is low, and the communication efficiency between CPU-GPU is the biggest bottleneck (Cao et al., 2023). Therefore, the computational performance of CAMx-HIP model is not as good as the original Fortran version when MPI processes increase. According to the characteristics of GPUs suitable for large-scale matrix computing, the model domain can be expanded and the model resolution can be increased in the future to ensure that the amount of data transferred to each GPU reaches the maximum video memory occupation, so as to make efficient use of GPU. In addition, the advection module only accounts for about 10% of the total time consumption in CAMx model (Cao et al., 2023), and in the future, it is considered to port the entire integration module except I/O to the GPU to minimize the communication frequency.

Minor Comments:

L70: Kinesthetic PreProcessor: Accelerated (KPPA) -> It should read "Kinetic PreProcessor"

Response: Sorry for this mistake. We have modified the relevant statement in **line 62**, which are as follows:

Line 62:

Linford et al. (2011) presented the Kinetic PreProcessor (KPP) to generate the chemical mechanism code in CUDA language which can be implemented on NVIDIA Tesla C1060 GPU.

L. 115: CUDA is not only supported on Tesla, but all NVIDIA GPU architectures. Please rephrase

Response: Sorry for this mistake. We have modified the relevant statement in **lines 106-107**, which are as follows:

Lines 106-107:

CUDA is a proprietary application programming interface (API) and as such is supported on all NVIDIA GPU architectures.

L155-160: Most of the technical information is repeated in Table 2 and can be omitted from the text.

Response: Thanks for the constructive suggestion. We have omitted the repeated relevant description of Table 2 in **lines 122-141**, which are as follows:

Lines 122-141:

Table 1 listed four GPU clusters which are conducted the experiments, two NVIDIA heterogeneous clusters which have the same hardware configuration as Cao et al. (2023) and two China's domestically heterogeneous clusters (domestic clusters) newly used in this research. Two NVIDIA heterogeneous clusters are equipped with NVIDIA Tesla K40m and V100 GPU accelerators, respectively. Both two domestic clusters include thousands of computing nodes and each containing one China's domestically CPU processor, four China's domestically GPU-like accelerators, and 128 GB of DDR4 2666 memory. The domestic CPU has 32 X86 based processors. The accelerator adopts a GPU-like architecture consisting of a 16 GB HBM2 device memory and many compute units. The GPU-like accelerators connected to CPU with PCI-E, the peak bandwidth of the data transfer between main memory and device memory is 16 GB/s.

It is worth noting that the domestic cluster B is the next generation of domestic cluster A, which has been updated in three main aspects. The CPU clock speed has been increased from 2.0 GHz to 2.5 GHz, the number of GPU-like computing units has been increased from 3,840 to 8,192, and the peak bandwidth between main memory and video memory has been increased from 16 GB/s to 32 GB/s. The domestic GPU-like accelerator is very similar in architecture to the NVIDIA GPU, and

the CUDA core of the NVIDIA GPU essentially corresponds to the computing unit of the domestic GPU. In addition, the NVIDA GPU is programmed using the CUDA toolkit, and the domestic GPU-like is programmed using the ROCm-HIP toolkit. More details about the hardware composition and software environment of the four heterogeneous clusters are presented in Table 1.

Table 1. *Configurations of NVIDIA K40m cluster, NVIDIA V100 cluster, China's domestically cluster A, and China's domestically cluster B.*

	Hardware components	
	CPU	*GPU*
NVIDIA K40m cluster	*Intel Xeon E5-2682 v4 CPU @2.5 GHz, 16 cores*	*NVIDIA Tesla K40m GPU, 2880 CUDA cores, 12 GB video memory*
NVIDIA V100 cluster	*Intel Xeon Platinum 8168 CPU @2.7 GHz, 24 cores*	*NVIDIA Tesla V100 GPU, 5120 CUDA cores, 16 GB video memory*
China's domestically cluster A	*China's domestically CPU processor A, 2.0GHz, 32 cores*	*China's domestically GPU-like accelerator A, 3840 computing units, 16 GB memory*
China's domestically cluster B	*China's domestically CPU processor B, 2.5GHz, 32 cores*	*China's domestically GPU-like accelerator B, 8192 computing units, 16 GB memory*

	Software environment	
	Compiler and MPI	*Programming model*
NVIDIA K40m cluster	*Intel Toolkit 2021.4.0*	*CUDA-10.2*
NVIDIA V100 cluster	*Intel Toolkit 2019.1.144*	*CUDA-10.0*
China's domestically cluster A	*Intel Toolkit 2021.3.0*	*ROCm-4.0.1/ DTK-23.04*
China's domestically cluster B	*Intel Toolkit 2021.3.0*	*DTK-23.04*

Please move all links to websites to the references section

Response: Sorry for this mistake. We have moved all links to the references section, which are as follows:

Lines 48-52:

The 61st edition of the top 10 list, released in June 2023, reveals that 80% of advanced supercomputers adopt the heterogeneous architectures (Top500, 2023), and the Frontier system equipped with AMD Instinct MI250X GPU at the Oak Ridge National Laboratory remains the only true exascale machine with the High-Performance Linpack benchmark (HPL) score of 1.194 Exaflop/s (News, 2023).

Lines 84-87:

The Comprehensive Air Quality Model with Extensions version 6.10 (CAMx v6.10; ENVIRON, 2014) is a state-of-the-art air quality model which simulates the emission, dispersion, chemical reaction, and removal of the air pollutants on a system of nested three-dimensional grid boxes (CAMx, 2023).

Lines 117-118:

More information about HIP API can be available on the AMD ROCm website (ROCm, 2023).

Reference

CAMx, A multi-scale photochemical modeling system for gas and particulate air pollution, available at: https://www.camx.com/ (last access: 20 October 2023), 2023.

Cao, K., Wu, Q., Wang, L., Wang, N., Cheng, H., Tang, X., Li, D., and Wang, L.: GPU-HADVPPM V1.0: a high-efficiency parallel GPU design of the piecewise parabolic method (PPM) for horizontal advection in an air quality model (CAMx V6.10), Geosci. Model Dev., 16, 4367-4383, 10.5194/gmd-16-4367-2023, 2023.

Huang, M., Huang, B., Mielikainen, J., Huang, H. L. A., Goldberg, M. D., and Mehta, A.: Further Improvement on GPUBased Parallel Implementation of WRF 5-Layer Thermal Diffusion Scheme, in: 2013 International Conference on Parallel and Distributed Systems, Seoul, South Korea, 15–18 December 013, https://doi.org/10.1109/icpads.2013.126, 2013.

Linford, J. C., Michalakes, J., Vachharajani, M., and Sandu, A.: Automatic Generation of Multicore Chemical Kernels, IEEE Transactions on Parallel and Distributed Systems, 22, 119-131, 10.1109/tpds.2010.106, 2011.

Mielikainen, J., Huang, B., Huang, H.-L. A., and Goldberg, M. D.: GPU Implementation of Stony Brook University 5-Class Cloud Microphysics Scheme in the WRF, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 5, 625-633, 10.1109/jstars.2011.2175707, 2012.

Mielikainen, J., Huang, B., Wang, J., Allen Huang, H. L., and Goldberg, M. D.: Compute unified device architecture (CUDA)-based parallelization of WRF Kessler cloud microphysics scheme, Computers & Geosciences, 52, 292-299, 10.1016/j.cageo.2012.10.006, 2013.

News, Frontier Remains as Sole Exaflop Machine and Retains Top Spot, Improving Upon Its Previous HPL Score, available at: https://www.top500.org/news/frontier-remains-sole-exaflop-machine-and-retains-top-spot-improving-upon-its-previous-hpl-score/ (last access: 20 October 2023), 2023.

ROCm, AMD ROCm-HIP documentation, available at: https://rocm.docs.amd.com/projects/HIP/en/latest/index.html (last access: 20 October 2023), 2023.

Top500, Supercomputing Top500 list, available at: https://www.top500.org/lists/top500/2023/06/ (last access: 20 October 2023), 2023.

Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., and Carver, G.: Single Precision in Weather Forecasting Models: An Evaluation with the IFS, Mon. Weather Rev.,145, 495–502, https://doi.org/10.1175/mwr-d-16-0228.1, 2017.

Wang, H., Lin, J., Wu, Q., Chen, H., Tang, X., Wang, Z., Chen, X., Cheng, H., and Wang, L.: MP CBM-Z V1.0: design for a new Carbon Bond Mechanism Z (CBM-Z) gas-phase chemical mechanism architecture for next-generation processors, Geosci. Model Dev., 12, 749–764, https://doi.org/10.5194/gmd-12-749-2019, 2019.