GPU accelerators have great potential to improve the efficiency of air quality modeling. The authors have explained the steps taken to convert a PPM solver of horizontal advection for air quality model CAMx into a new Compute Unified Device Architecture C (CUDA C) code and how this optimization approach has resulted in significant improvement in computational efficiency. This paper presents a promising approach for using GPU accelerators in air quality modeling and provides valuable insights into the optimization measures that can be taken to improve the overall computing performance of model. The authors' findings can potentially benefit researchers in air quality modeling and related fields. My comments listed below.

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

1. The manuscript should be written more concisely, removing unnecessary material. For example, the information presented in Table 1 partly duplicates that presented in Figure 1b. Figure 9 doesn't provide too much important information and can be moved to Supplementary.

   Response: Thanks for the constructive comment. We have removed Table 1, and placed Figure 9 in the supplementary. The relevant expressions and contents in the paper have also been modified in lines 156-162 and 417-421, which are as follows:

   Lines 156-162: The VTune tool detects each module's runtime and the most time-consuming functions on P1. As shown in Figure 1b, the top four time-consuming modules are chemistry, diffusion, horizontal advection, and vertical advection in the CAMx model. In the above four modules, the top five most time-consuming programs are ebirate, hadvppm, tridiag, diffus, and ebisolv programs, and the total runtime of P1 is 325.1 seconds. Top1 and Top2's most time-consuming programs take 49.4 and 35.6 seconds, respectively.

   Lines 417-421: In terms of time series, the regionally averaged time series of the three versions are almost consistent (as is shown in Figure S2), and the maximum AEs for the above six species are 0.001ppbV, 0.005 ppbV, 0.002 ppbV, 0.03ppbV, 0.0001 ppbV and 0.0002 $\mu g \cdot m^{-3}$, respectively, between the Fortran and CUDA C versions.

2. The authors select the hadvppm program to implement the heterogeneous porting. It could benefit from providing some additional context about the motivation for selecting hadvppm

program.

Response: Thanks for the constructive comment. There are some reasons for selecting hadvppm for heterogeneous porting. Firstly, the advection module is one of the compulsory modules of the air quality model, which is mainly used to simulate the transport process of air pollutants, and it is also a hotspot module detected by the Intel VTune tool. Then, typical air quality models CAMx, CMAQ, and NAQPMS include advection modules and use the exact PPM advection solver. The heterogeneous version developed in this study can be directly applied to the above models. Furthermore, the weather model (e.g., WRF) also contains an advection module, so this study's heterogeneous porting method and experience can be used for reference. We have revised this part in lines 163-173, which are as follows:

By consideration, the hadvppm program was selected to carry out heterogeneous porting for some reasons. Firstly, the advection module is one of the compulsory modules of the air quality model, which is mainly used to simulate the transport process of air pollutants, and it is also a hotspot module detected by the Intel VTune tool. Then, typical air quality models CAMx, CMAQ, and NAQPMS include advection modules and use the exact PPM advection solver. The heterogeneous version developed in this study can be directly applied to the above models. Furthermore, the weather model (e.g., WRF) also contains an advection module, so this study's heterogeneous porting method and experience can be used for reference. Therefore, a GPU acceleration version of the HADVPPM scheme, namely GPU-HADVPPM, is built to improve CAMx performance.

3. In Figure 5-7, large absolute errors mainly occur over Zhangjiakou City for SO_2, NO_2 CO and PSO_4 while simulated O_3 and H_2O_2 show large errors over Baoding city. Please explain how do porting process cause such patterns.

Response: Thanks for the constructive comment. During the process of heterogeneous porting, the CUDA technology was used to convert the standard C code into CUDA C to make the hadvppm program computable on the GPU. However, due to the slight difference in data operation and accuracy between CPU and GPU(NVIDIA,2023), the concentration variable of hadvppm program appears to have minimal negative values (about -10^{-4}~-10^{-9}) when integrating on GPU. In order to allow the program to continue running, we forcibly replace these negative values with 10^{-9}. It is because these negative values are replaced by positive

values that the simulation results are biased. Furthermore, we adopted the evaluation method of Wang et al. (2021) to compute the ratio between the root-mean-square-error and standard deviation of six species, such as SO_2, O_3, NO_2, CO, H_2O_2, and PSO_4. The results were 0.004%, 0.003%, 0.003%, 0.006%, 0.015%, and 0.004%, respectively, which were far less than the 0.2% mentioned by Wang et al. (2021), so the absolute errors of simulation results in this study are within the acceptable range. We have revised this part in lines 375-383, which are as follows:

During the porting process, the primary error came from converting standard C to CUDA C, and the main reason was the hardware difference between the CPU and GPU. Due to the slight difference in data operation and accuracy between CPU and GPU(NVIDIA,2023), the concentration variable of hadvppm program appears to have minimal negative values (about -10^{-4}~-10^{-9}) when integrating on GPU. In order to allow the program to continue running, we forcibly replace these negative values with 10^{-9}. It is because these negative values are replaced by positive values that the simulation results are biased.

4. I'd suggest the authors discussing in the Conclusions and Discussion section any limitations of the current approach, which may warrant future refinement.

Response: Thanks for the constructive comment. There are some limitations of the current approach.

1) We currently implemented thread and block co-indexing to compute horizontal grid points in parallel. Given the CAMx model 3-dimensional grid computing characteristics, 3-dimensional thread and block co-indexing will be considered to compute 3-dimensional grid points in parallel.
2) The communication bandwidth of data transfer is one of the main issues for restricting the computing performance of CUDA C codes on GPUs. In this study, data transmission efficiency between CPU and GPU is improved only by reducing communication frequency. In the future, more technologies, such as pinned memory (Wang et al.,2016), will be considered to solve the communication bottleneck between CPU and GPU.
3) In order to further improve the overall computational efficiency of the CAMx model, the heterogeneous porting scheme proposed in this study will be considered to carry out the heterogeneous porting of other CAMx modules in the future.

We have added this part in lines 600-614, which are as follows:

However, there are some limitations of the current approach which are as follows:

1) We currently implemented thread and block co-indexing to compute horizontal grid points in parallel. Given the CAMx model 3-dimensional grid computing characteristics, 3-dimensional thread and block co-indexing will be considered to compute 3-dimensional grid points in parallel.
2) The communication bandwidth of data transfer is one of the main issues for restricting the computing performance of CUDA C codes on GPUs. This restriction not only holds true for GPU-HADVPPM, but also WRF module as well (Mielikainen et al., 2012b; Mielikainen et al., 2013b; Huang et al., 2013). In this study, data transmission efficiency between CPU and GPU is improved only by reducing communication frequency. In the future, more technologies, such as pinned memory (Wang et al.,2016), will be considered to solve the communication bottleneck between CPU and GPU.
3) In order to further improve the overall computational efficiency of the CAMx model, the heterogeneous porting scheme proposed in this study will be considered to carry out the heterogeneous porting of other CAMx modules in the future.

5. Line 173-177, 337-342: Please unify the tense of the sentence.

Response: Sorry for these mistakes. We have revised this part in lines 180-187 and 342-349, which are as follows:

Lines 180-187: The heterogeneous scheme of CAMx-CUDA is shown in Figure 2. The second time-consuming hadvppm program in the CAMx model was selected to implement the heterogeneous porting. In order to map the hadvppm program to the GPU, the Fortran code was converted to standard C code. Then, CUDA programing language, which was tailor-made for NVIDIA, was added to convert the standard C code into CUDA C for data-parallel execution on GPU, as GPU-HADVPPM. It prepared the input data for GPU-HADVPPM by constructing random numbers and tested its offline performance on the GPU platform.

Lines 342-349: The validation and evaluation of porting the HADVPPM scheme from the CPU to the GPU platform were conducted using offline and coupling performance experiments. First, we validated the result between different CAMx versions, and then the offline performance of the GPU-HADVPPM on a single GPU was tested by offline experiment. Finally, the coupling performance experiments illustrate its potential in three dimensions with varying chemical

regimes. Sect.4.2 and Sect.4.4, the CAMx version of the HADVPPM scheme written by Fortran language, standard C, and CUDA C, is named F, C, and CUDA C, respectively.

6. Line 354: Please clarify “30-min spatial resolution”.

Response: Sorry for the confusion. Minute is the angular resolution. According to the conversion formula, 30-min is 0.5 degrees, and the corresponding range resolution is about 55.6 kilometers at mid-latitude. We have revised this part in line 361, which are as follows:

30-min (about 55.6km at mid-latitude) spatial resolution

Reference

NVIDIA: Floating Point and IEEE 754 Compliance for NVIDIA GPUs. Release 12.1, available at: https://docs.nvidia.com/cuda/floating-point/#differences-from-x86 (last access: 18 May 2023), 2023.

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, 10.5194/gmd-14-2781-2021, 2021.

Wang, Z., Wang, Y., Wang, X., Li, F., Zhou, C., Hu, H., and Jiang, J.: GPU-RRTMG_SW: Accelerating a Shortwave Radiative Transfer Scheme on GPU, IEEE Access, 9, 84231-84240, 10.1109/access.2021.3087507, 2016.