Improving the Parallelism of CESM on GPU

Zehui Jin[1], Ming Dun[1], Xin You[1], Hailong Yang[1(✉)], Yunchun Li[1], Yingchun Lin[2], Zhongzhi Luan[1], and Depei Qian[1]

[1] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
hailong.yang@buaa.edu.cn
[2] Fourth Research Institute of Telecommunications Technology Corporation, Xi'an 710061, Shaanxi, China

Abstract. Community Earth System Model (CESM) is one of the most popular climatology research models. However, the computation of CESM is quite expensive and usually lasts for weeks even on high-performance clusters. In this paper, we propose several optimization strategies to improve the parallelism of three hotspots in CESM on GPU. Specifically, we analyze the performance bottleneck of CESM and propose corresponding GPU accelerations. The experiment results show that after applying our GPU optimizations, the kernels of the physical model achieve significant performance speedup respectively.

Keywords: CESM · GPU · Performance optimization

1 Introduction

Scientists have been striving to understand and even predict the earth's climate system since the early days. The effort led by the National Center for Atmosphere Research (NCAR) gave birth to the open-source simulation model named Community Climate Model (CCM) [4]. As the successor of CCM, NCAR released The Community Earth System Model (CESM), which becomes one of the cutting-edge climate simulation models nowadays.

The CESM simulation model is constituted of five geophysical components, including atmosphere, land, ocean, sea-ice and land-ice component. There is also a coupler in CESM to incorporate the above geophysical components. However, both the computation and memory cost when running CESM is overwhelmingly high on CPU due to the complex mathematical equations adopted in the simulation.

There have already been several works optimizing CESM by accelerating certain kernels in CESM on GPU. Korwar et al. [5] implement the long wave and short wave radiation kernels on GPU. Carpenter et al. [3] implement the spectral element based dynamical core (HOMME) on GPU. Sun et al. [10] optimize a solver in the chemistry procedure of the atmospheric component and implement

S. Wen et al. (Eds.): ICA3PP 2019, LNCS 11945, pp. 11–18, 2020.
https://doi.org/10.1007/978-3-030-38961-1_2

it on GPU for higher performance. However, none of the existing works simultaneously accelerates three hotspot kernels on GPU including aerosol masses conversion, longwave radiation, and solar radiation.

This work focuses on optimizing the performance of the atmospheric component, which is one of the most time-consuming components of CESM. First, we comprehensively analyze the performance bottlenecks of CESM. Then we optimize the three hotspot kernels on GPU. We demonstrate the effectiveness of our optimizations with experiment results.

In short, this paper makes the following contributions:

- We conduct a comprehensive performance analysis of CESM and identify three hotspot kernels in the atmospheric component.
- We implement the identified hotspot kernels on GPU and optimize the kernels for better performance.

The rest of the paper is organized as follows. The background of CESM and existing parallel methods are presented in Sect. 2. We perform the bottleneck analysis of CESM and present the performance optimization strategies in Sect. 3. In Sect. 4, the detailed experiment results are given. We present the related work in Sect. 5 and conclude this paper in Sect. 6.

2 Background

2.1 CESM Overview

CESM is mainly developed by the NCAR and its predecessor is known as the CCSM. CESM is comprised of several coupled components: atmosphere, ocean, land, land-ice, sea-ice and a central coupler component. Once the simulation began, the coupler will swap the two-dimensional data. The cutting-edge earth system simulation software provides support for plenty of options in different combinations of component configurations and resolutions [11].

2.2 The Atmosphere Component

The atmosphere component of CESM is called the Community Atmosphere Model (CAM). CAM 4.0 mainly constitutes four kinds of dynamical cores and a physical model which is an aggregation of the simulation of the physical process. The total parameter package of physical model can be indicated as Eq. 1, where M means (Moist) precipitation processes, R for radiation and clouds, S for the abbreviation of the Surface model, and T for Turbulent mixing [8].

$$P = \{M, R, S, T\} \tag{1}$$

2.3 Parallelization of CESM

Computation Parallelization. Within almost every geophysical part of CESM, the system is divided into hierarchical grids and the grids are allocated to different processes to compute. To the perspective of a standalone component, the atmosphere, ice, ocean, and land component support two levels of parallelization: the distributed memory parallelism(MPI) and the shared memory parallelism model (OpenMP) [13]. For the whole CESM, some geophysical components can run concurrently but there remain constraints. For instance, the atmosphere component cannot run with ice or land component.

I/O Parallelization. CESM usually use netCDF [9] for reading and writing data. There are two ways to parallel the I/O procedure. One way is by enabling the parallelism version of netCDF and the other way is to replace the default netCDF with pnetCDF [6]. Paralleled netCDF can read input data through multiple processors which can contribute to I/O acceleration.

3 Performance Optimization Strategies

3.1 Bottleneck Analysis of CESM

We probe the hotspots of CESM by using HPCToolkit on the platform whose software and hardware details are shown in Sect. 4. The hotspot analysis is shown in Fig. 1. From the analysis result, the conclusion can be drawn that the atmosphere component costs most of the execution time, as the dynamical part spends approximately 48.8% of total running time while the physical part takes up about 33% of the overall execution time.

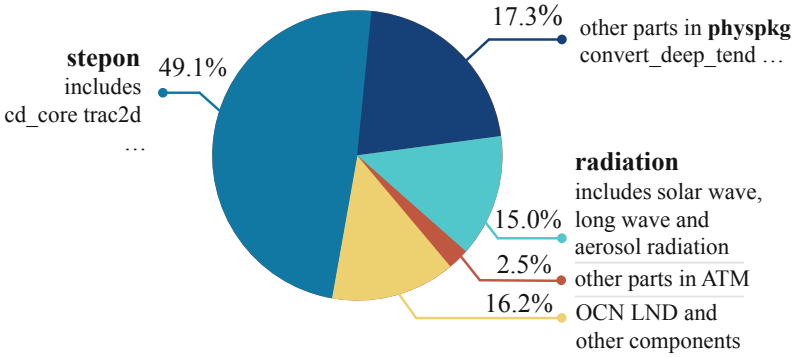

Fig. 1. The performance bottleneck analysis of CESM.

It can be figured out that the MPI communication and *cd_core* are two hotspots in the dynamic part. However, since the optimization of MPI can be

done by alternating MPI library which is much easier than algorithm optimization and *cd_core* consists of numerous loops that only cost about 0.1% of execution time each, we focus on optimizing the hotspots in physical part in this paper.

In the physical model, it can be revealed from the result that the radiation routine is a hotspot which takes up 15% of execution time approximately. The radiation routine includes solar radiation, longwave radiation and aerosol masses conversion, which contains computation-intense loops.

3.2 Solar Radiation Optimization

In the original solar radiation routine in CESM, there is a computation-intense triple cycle which contains $nswband \times (pver + 1) \times Nday$ iterations. To enhance the performance in its GPU implementation, we assign every iteration to a thread in GPU to improve parallelization, the stream number and the size of the grid are shown in Algorithm 1. Once all the threads finish liquid and ice valid computation, they get the boundaries of solar radiation and then compute radiative properties of layers. After all the computation is done, the results will be sent to the host.

Algorithm 1. The optimized solar radiation kernel on GPU

```
malloc memory for temporary vars on GPU
malloc memory for input vars on GPU and copy values
/ * stream number = 1 * /
/ * grid size = 4 × 4 × 1 * /
/ * block size = 64 × 4 × 1 * /
/ * total threads amount = Nday × (pver + 1) × nswbands * /
for each thread do
    initialize global constant parameters
    if (blockIdx.y × blockDim.y + threadIdx.y)%(pver + 1) ≠ 0 then
        calculate liquid and ice valid
    end if
    synchronize
    get boundaries of solar wave spectral
    compute layer radiative properties
end for
copy out results and free memory on GPU
```

3.3 Longwave Radiation Optimization

We focus on optimizing the absorptivities computation kernel when improving the performance of the longwave radiation routine. In the original kernel, there is a triple cycle which iterates $ncol \times (pverp - ntoplw + 1) \times (pverp - ntoplw + 1)$ times. To implement this kernel to GPU, we assign the threads to each iteration as shown in Algorithm 2. For each thread, it computes the band-dependent indices for non-window and window. Next, the threads calculate the absorptivity for non-nearest layers and finally calculate total absorptivity based on previous steps.

Algorithm 2. The optimized longwave radiation kernel on GPU

```
Initialize constants as local parameters
/ * stream number : 1 * /
/ * grid size : 5 × 4 × 1 * /
/ * block size : 4 × 64 × 1 * /
/ * total threads amount : ncol × (pverp − ntoplw + 1) × (pverp − ntoplw + 1) * /
for each thread do
    / * H2O Continuum path for 0 − 800 and 1200 − 2200 cm^−1 * /
    Band − dependent indices for non − window
    / * Line transmission in 800 − 1000 and 1000 − 1200 cm^−1 intervals * /
    Get O3 9.6 micrometer band
    Get CO2 15 micrometer band system
    Calculate absorptivity for non nearest layers
    Calculate total absorptivity based on previous five steps
end for
```

3.4 Aerosol Masses Conversion Optimization

In the GPU implementation of aerosol masses conversion routine whose processing logic is shown in Algorithm 3. After the stream and thread are set up by GPU, the device will get *opticstype* which indicates the type of aerosol to be computed. There are four types of aerosols: hygroscopic aerosols, non-hygroscopic aerosols, volcanic aerosols, and volcanic radius aerosols. Once the *opticstype* is received, the threads will calculate the corresponding optical properties for those aerosols.

Algorithm 3. The optimized aerosol masses conversion kernel on GPU

```
Malloc memory for temporary variables on GPU
Create CUDA stream
/ * total threads = pver × pcols × nswbands * /
/ * stream number = 1
/ * grid size = 8 × 4 × 1 * /
/ * block size = 64 × 4 × 1 * /
for iaerosol = 1 → numaersols do
    Get opticstype : optics ← StringToInt(optistype)
    if optics == 1 then
        for each thread, calculate optical properties for hygroscopic aerosols
    else if optics == 2 then
        for each thread, calculate optical properties for non − hygroscopic aerosols
    else if opitcs == 3 then
        for each thread, calculate optical properties for volcanic aerosols
    else if optics == 4 then
        for each thread, calculate optical properties for volcanic radius aerosols
    else
        return error message / * default * /
    end if
    copy results to host's memory
    free memory on GPU
end for
```

4 Evaluation

4.1 Experiment Setup

The experiments are conducted on a single server. The server is equipped with 2× Intel Xeon E5-2680v4 2.40 GHz 14 cores and two NVIDIA Volta 32 GB V100. The operating system installed is CentOS v7.6 with Linux kernel v3.10.0-957.el7.x86_64. We use CESM v1.2.2 compiled with Intel compiler v2018.5.274 as our baseline. CUDA version is 10.1, and netCDF is v4.6.2 enabled parallelization. For the simulation input, we choose two representative datasets, *E1850CN* and *F*, with corresponding simulation setups. For each dataset, we run CESM on two commonly used resolutions, 0.47 × 0.63 and 0.23 × 0.31.

4.2 Performance Analysis

In Fig. 2(a), the orange bars represent the average running time of the aerosol masses conversion kernel in the original CESM, whereas the purple ones represent the average running time of our GPU optimization. It is obvious that our GPU optimization of the kernel reduces the execution time significantly. The average performance speedup on dataset *E1850CN* and *F* is 6.91× and 9.14× respectively.

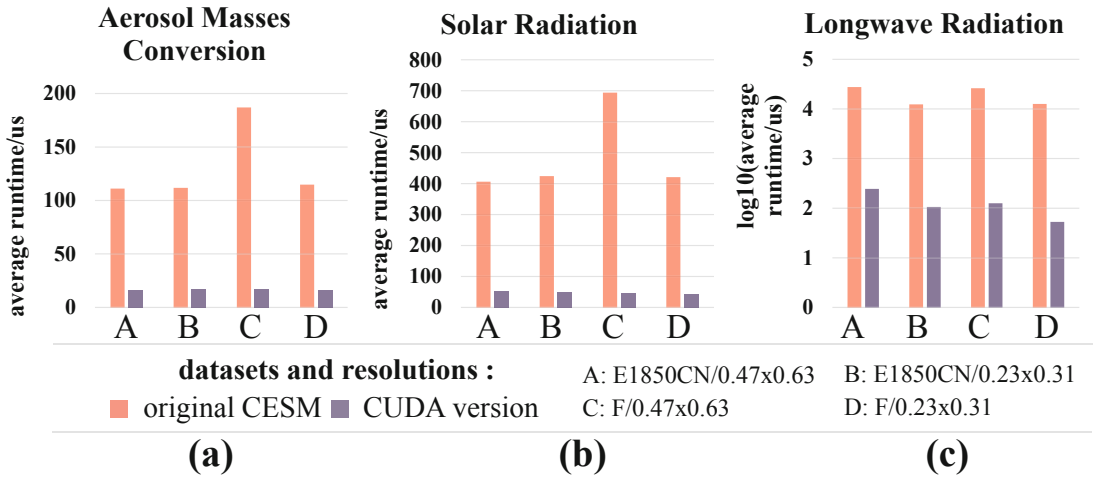

Fig. 2. The performance speedup of the three kernels. (a) for aerosol masses conversion kernel, (b) for solar radiation and (c) for longwave radiation kernel. (Color figure online)

We can see from Fig. 2(b) that the experiment results on solar radiation kernel show the similar trend. The average performance speedup on dataset *E1850CN* and *F* is 8.36× and 12.38× respectively. The higher performance speedup achieved by solar radiation kernel is due to its high computation intensity that benefits from the massive parallelism of GPU.

The performance speedup on longwave radiation kernel is much higher than the previous two kernels. As shown in Fig. 2(c), due to the computation-intensive

nature of the longwave radiation kernel, the original implementation takes more than 10^4 microseconds to complete, so we show the logarithmic results. The average performance speedup on dataset *E1850CN* and *F* is 114.95× and 222.4× respectively.

5 Related Work

Most of the relevant works port some of the kernels in CESM to GPU. Korwaret et al. [5] transfer the long-wave and solar radiation routines to GPU by OpenACC and develop a CPU-GPU asynchronous scheme. In the meantime, the spectral element based dynamical core(HOMME) is implemented to GPU by Carpenter et al. [3]. Later, Werkhoven et al. [12] modify the block-partitioning strategies in the ocean component of CESM named POP into a hierarchical scheme to optimize load balance and reduce MPI communication. Furthermore, Sun et al. [10] accelerate the second-order Rosenbrock solver by implementing it to GPU, and it turns out that a higher performance will be reached when the block size is equal to the warp size and memory is continuous. Moreover, as a part of the OpenACC based project Energy Exascale Earth System Model(E3SM) whose atmospheric component is a derivative of CAM5 in CESM1, Bertagna et al. [2] rewrite the HOMME with C++ and Kokkos which makes the core portable to GPU, which results in better performance. Since the inappropriate parameters can cause severe performance deterioration, an optimal execution configuration is desirable. Nan et al. [7] develop a framework named CESMTuner which can detect the best configuration automatically. Balaprakash et al. [1] develop a static parameter-probing model based on machine learning and apply it to the ice component of CESM.

6 Conclusion

In this paper, we optimize three hotspot kernels in CESM on GPU and achieve significant performance speedup. We use the HPC toolkit to analyze the performance of CESM and identify three hotspot kernels. Then we implement these hotspot kernels on GPU and optimize them correspondingly. The experiment results show that our GPU optimizations achieve 11.25×, 15.02× and 237× performance speedup for aerosol masses conversion, solar radiation and longwave radiation respectively.

Acknowledgement. This work is supported by National Key Research and Development Program of China (Grant No. 2016YFB1000304) and National Natural Science Foundation of China (Grant No. 61502019).