Multi-GPU Optimization of a Non-hydrostatic Numerical Ocean Model with Multigrid Preconditioned Conjugate Gradient Method

Takateru Yamagishi
Research Organization for Information Science and Technology
Minato, Tokyo, Japan
yamagishi@rist.jp

Yoshimasa Matsumura
Atmosphere and Ocean Research Institute, The University of Tokyo
Kashiwa, Chiba, Japan
ymatsu@aori.u-tokyo.ac.jp

Hiroyasu Hasumi
Atmosphere and Ocean Research Institute, The University of Tokyo
Kashiwa, Chiba, Japan
hasumi@aori.u-tokyo.ac.jp

ABSTRACT

The conjugate gradient method with multigrid preconditioners (MGCG) is used in scientific applications because of its high performance and scalability with many computational nodes. GPUs are thought to be good candidates for accelerating such applications with many meshes where an MGCG solver could show high performance. No previous studies have evaluated and discussed the numerical character of an MGCG solver on GPUs. Consequently, we have implemented and optimized our "kinaco" numerical ocean model with an MGCG solver on GPUs. We evaluated its performance and discussed inter-GPU communications on a coarse grid on which GPUs could be intrinsically problematic. We achieved 3.9 times speedup compared to CPUs and learned how inter-GPU communications depended on the number of GPUs and the aggregation level of information in a multigrid method.

1 Introduction

Numerical ocean models are expected to play important roles in predicting climate change and investigating marine resources under various climatic conditions. To conduct realistic simulations, we therefore must resolve and simulate very small-scale dynamics over an extensive ocean. GPUs have substantial computational power and wide memory bandwidth, making them suitable for numerical simulations with many grids. This model includes an MGCG solver used in several HPC applications because of its high performance and scalability with many computational nodes. Its optimization and numerical character on CPUs have been discussed in previous studies [1], but no studies have evaluated and discussed MGCG solvers on GPUs in detail. GPUs are not efficient at both computation and inter-GPU communication on coarse grids, hence GPUs implemented on MGCG solvers on coarse grids require evaluation and discussion. In this study, we optimize our numerical ocean model using an MGCG solver and discuss inter-GPU communications on coarse grids.

2 Model description

Our model, kinaco, simulates ocean dynamics using the three-dimensional non-hydrostatic Navier–Stokes equation in an orthogonal curvilinear coordinate system [2]. All of the equations are discretized on structured grids and approximated using Poisson/Helmholtz equations, solved using the MGCG solver. This model is parallelized by domain decomposition using the MPI communication library. In the multigrid method that we apply to preconditioning, matrix and/or vector calculations are executed on several resolution grids. The level of the finest grid is set to zero, and the levels are numbered from the finest to coarsest grid on which the number of meshes is one for each MPI process. In this experiment, we assigned one GPU to one MPI process. At the coarsest level, the information of multiple GPUs is gathered to one GPU through collective MPI. The gathered meshes are executed on one GPU following the same idea, where no communication is required (Fig. 1). Especially, communication of a few grids can be more inefficient compared to communication on CPUs owing to the overhead of CUDA-Aware MPI resulting from the intrinsic structure of a GPU-CPU system.

3 Implementation and Optimization on GPUs

Our previous study in SC18 [3] implemented and optimized kinaco on as many as 16 GPUs using CUDA, which showed 3.3 times speedup compared to CPU-based executions. In this study, we optimized the communication of kinaco to achieve more acceleration and expanded to 64 GPUs.

We evaluated our GPU-implemented kinaco performance and found that communications in the MGCG solver were performance bottlenecks. We used two techniques to solve this problem, overlapping of communication with computation [4, 5] and modification of multigrid method aggregation levels [1]. We introduced two kinds of overlapping method. In the first, some values can be executed independently of others, and their communications can overlap with other variables (Fig. 2a). As for kinaco, calculations of dot-products and norms of error vectors could overlap with data exchange. The second method relies on data independence within a single variable. As for most of kinaco's variables, only the halo region is exchanged with neighbors. Fig. 2b shows an example in which exchange of the halo areas is required for sequential computing. This calculation in the inner area is not related to previous communication and hence can overlap with communication. As for kinaco, the matrix-vector multiplication kernel is divided into internal and halo domains. In our original multigrid method, operations for aggregation of MPI processes are performed at the finest level (Fig. 1). Coarse grid aggregation (CGA) was proposed in [3]. Fig. 1 shows CGA procedures, where information from each MPI

process is gathered in a single processor at level = 7([4, 4, 1]). Even though the size of the coarse grid problem is larger than that of the original configuration and the cost of gathering all information to one GPU would increase, we can expect that gathering all information to one GPU removes small grid communication, and also improves computational efficiency.

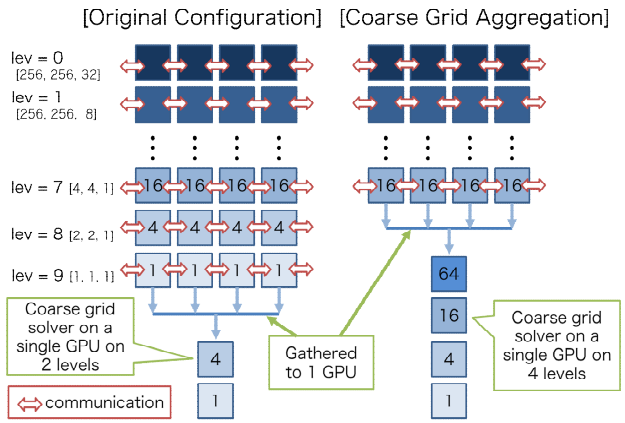

Figure 1: Procedures of original multigrid method and coarse grid aggregation with divided domain assigned to four GPUs.

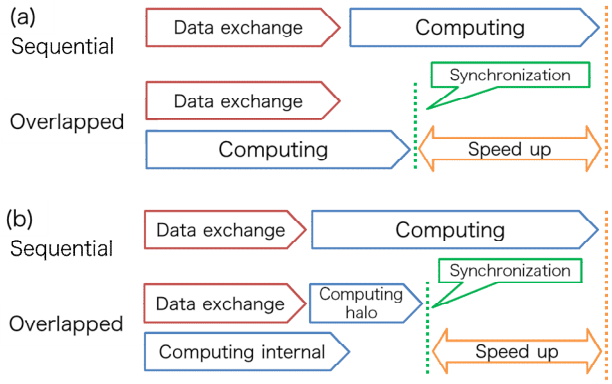

Figure 2: Overlapping methods of communication with computation. (a) independent computing is overlapped. (b) computing of independent domain is divided and overlapped.

4 Performance Evaluation

Each GPU's domain was set to a size of (256, 256, 32), and we evaluated the model using two, four, eight, 16, 32, or 64 GPUs or CPUs. Since the total number of grids increased linearly with the number of GPUs, this was an evaluation of the model's weak scalability. We considered a test case with idealistic and systematic forcing and with boundary conditions that induced baloclinic instability. We used NVIDIA Pascal P100 GPUs and Intel Xeon E5-2695v4 CPUs, both installed on Tokyo University's Reedbush supercomputer with GPUDirect technology. We used GPUDirect RDMA transfer for this evaluation.

As for communication optimizations, overlapping contributed 4% speedup of the MGCG solver (Fig. 3). As for the CGA, we changed the level on which all information is gathered to one GPU, from lev = 4 ([32, 32, 1]) to lev = 9 ([1, 1, 1]). As for two, four, and eight processors, the best case is lev = 4, which gathered the largest grids at the fine grid. 16 and 32 PEs were lev = 5, and 64 PEs was lev = 6. As for larger processors, the cost of gathering information was dominant in the case of finer grids. The improvement effects due to CGA are 12% for the case with two to 64 PEs. As a result of this communication optimization, the speed up of the MGCG solver is 16% faster, which is equivalent to 28% reduction of communication of the MGCG solver. The entire application is 3.9 times faster than CPUs, with good weak scaling up to 64 GPUs.

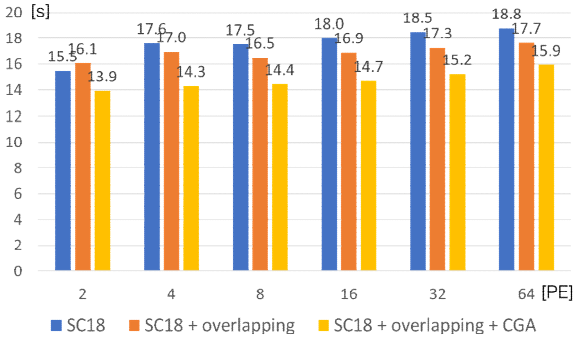

Figure 3: Effects of communication optimization on an MGCG solver. The vertical and horizontal axes represent elapsed time and number of processors, respectively.

5 Discussion and Conclusions

This study showed that multiple GPUs are good for both performance and scaling for numerical ocean modeling from two to 64 GPUs. MPI communication of an MGCG solver, which could include GPU-intrinsic bottlenecks, were improved using two techniques. We evaluated these optimizations and discussed both the reduction of halo data transfer and nonlinear increase in collective data transfer on coarse grids. We expect that these ideas can also be applied to other scientific applications using an MGCG solver.

An MGCG solver is known to be scalable, and kinaco has been shown to scale up to 8,000 CPUs [6]. As for the experiments with larger numbers of GPUs, the cost of MPI collective communication would increase nonlinearly, and the CGA method would work poorly. Hierarchical coarse grid aggregation (hCGA) proposed for CPUs [1] would be effective. In hCGA, the number of MPI processes are repartitioned at an intermediate level before the final coarse grid solver on a single MPI process. In future work, we plan to evaluate and optimize kinaco for thousands of GPUs. To reduce the MPI communication cost, we will attempt to apply hCGA to hundreds of GPUs and evaluate and analyze in detail.

REFERENCES

[1] K. Nakajima (2014). Optimization of serial and parallel communications for parallel geometric multigrid method. 2014 20th IEEE International.

[2] Y. Matsumura and H. Hasumi (2008). A non-hydrostatic ocean model with a scalable multigrid Poisson solver. Ocean Modelling 24(1–2): 15-28. http://dx.doi.org/10.1016/j.ocemod.2008.05.001

[3] T. Yamagishi, Y. Matsumura and H. Hasumi (2018). Multi-GPU Accelerated Non-Hydrostatic Numerical Ocean Model with GPUDirect RDMA Transfers. Supercomputing Conference 2018.

[4] P. Micikevicius (2009). 3D finite difference computation on GPUs using CUDA. Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. Washington, D.C., ACM: 79-84.

[5] T. Shimokawabe, T. Aoki, C. Muroi, J. Ishida, K. Kawano, T. Endo, A. Nukada, N. Maruyama and S. Matsuoka (2010). An 80-Fold Speedup, 15.0 TFlops Full GPU Acceleration of Non-Hydrostatic Weather Model ASUCA Production Code. Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Computer Society: 1-11.

[6] Y. Matsumura, H. Hasumi, E. Tomiyama, T. Yamagishi, T. Inoue, S. Inoue, K. Minami and K. I. Ohshima (2012). Numerical simulation of oceanic small scale processes by a non-hydrostatic ocean model. K computer symposium 2012.