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

Takateru Yamagishi(1) Yoshimasa Matsumura(2) Hiroyasu Hasumi(3)

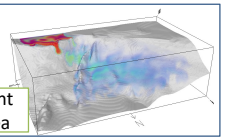
1: Research Organization for Information Science and Technology, *yamagishi@rist.jp* 2: Atmosphere and Ocean Research Institute, The University of Tokyo, *ymatsu@aori.u-tokyo.ac.jp* 3: Atmosphere and Ocean Research Institute, The University of Tokyo, *hasumi@aori.u-tokyo.ac.jp*

Introduction

- The objective is to accelerate high-spatial-resolution numerical ocean simulations with multiple NVIDIA GPUs.
- We implemented and optimized our "kinaco" numerical ocean model on multiple NVIDIA P100 GPUs.
- We optimized the inter-GPU communications of MGCG (multigrid preconditioned conjugate gradient method) solver by overlapping of communication with computation and modifying aggregation level of information in a multigrid method.
- We achieved 3.9 times speedup compared to CPUs with good weak scaling up to 64 GPUs.
- We learned how communications depended on the number of GPUs and the aggregation level of information in a multigrid method.

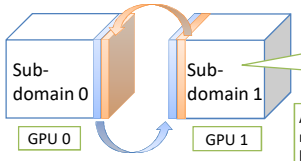
"Kinaco" non-hydrostatic ocean model

- Kinaco [1] resolves vertical convection and eddy mixing with non-hydrostatic approximation on ~1 km scale.
- Pressure field is solved using MGCG (multigrid preconditioned conjugate gradient method) solver, which can be scaled up to 8k processors [2].



Deep convection experiment in the Southern Weddell Sea

Multi-GPU implementation



Implemented domains on GPU with CUDA Fortran [3]

All components are on GPUs, no redundant data transfer between CPUs and GPUs.

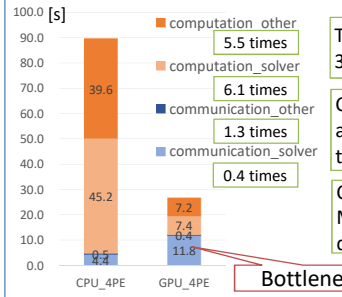
Halo region data required by adjacent domain passed to additional layer by CUDA-aware MPI using GPUDirect RDMA transfer

Specifications of Tokyo Univ.'s Reedbush and experiments

Hardware	Reedbush-H	SGI Rackable C1102-GP8
	CPU	Xeon E5-2695v4, [605 GF, 2.1 GHz, 18 core] × 2
	GPU	Tesla P100 with NVLink, [5.3 TF, 16 GB] × 2
	Interconnect	InfiniBand FDR 4x [56 Gbps] x 2
Software	Fortran	PGI 18.7, CUDA Fortran
	CUDA-aware MPI	OpenMPI 2.1.2 GDR (GPUDirect)
	CUDA	CUDA 9.1.85

Performance Evaluation

CPU vs GPU: elapsed time[s] in our SC18 study [3]



The overall speedup is 3.3 times faster [3].

Computations are accelerated by 5.5 to 6.1 times.

Communications of MGCG solver are deteriorated (0.4 times).

Bottleneck!

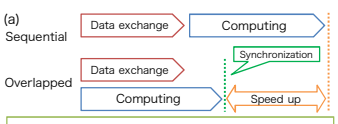
We used two techniques to solve this problem

- overlapping of communication with computation [4, 5]
- modification of multigrid method aggregation levels.

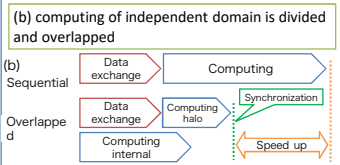
Communication optimization on fine grid: Overlapping with computation

Overlapping of communication with computation

(a) independent computing is overlapped



(b) computing of independent domain is divided and overlapped



Enough number of grids are required to overlap with communication, and we applied this method to only fine grids of MGCG solver.

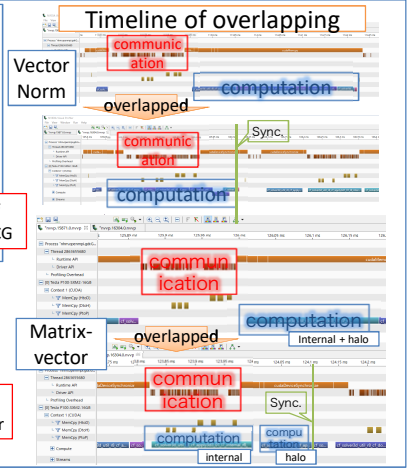
```
define preconditioner matrix of L as M
r_0 = q - L*p_0; u_0 = M*r_0; rho_0 = (M*r_0, r_0)
DO n = 0,1,2,,,
  alpha = rho_n/(u_n, L*u_n)
   // Dot Products
  p_n+1 = p_n + alpha*p_n
  r_n+1 = r_n - alpha*L*u_n
  if r_n+1 accurate enough then quit
   // Vector Norm
  e_n+1 = M*r_n+1  // preconditioner, CALL mgsolve
   //solve for e_n+1 in L*e_n+1 = r_n+1
  rho_n+1 = (M*r_n+1, r_n+1)
   // Dot Products
  beta = rho_n+1/rho_n
  u_n+1 = e_n+1 + beta*u_n
END DO
```

Pesudocode of Preconditioned CG

```
SUBROUTINE mgsolve(l, r, e)
  e_l = M_l * r_l
  CALL Halo_Data_Exchange
  IF l = l_c RETURN
  r_l+1 = Coarse(r_l - L * e_l)
  CALL Halo_Data_Exchange
  CALL mgsolve(l+1)
  e_l = e_l + Fine(e_l+1)
  CALL Halo_Data_Exchange
  t_l = r_l - L_l * e_l
  CALL Halo_Data_Exchange
  e_l = e_l + M_l * t_l
END SUBROUTINE
```

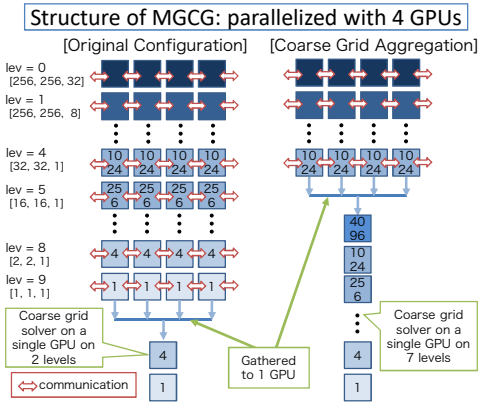
Multigrid Preconditioner

Timeline of overlapping



Communication optimization on coarse grid: Coarse Grid Aggregation of MGCG solver

Structure of MGCG: parallelized with 4 GPUs



In original configuration, 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.

Coarse grid aggregation (CGA) was proposed in [6]. In CGA procedures, where information from each MPI process is gathered in a single processor at level = 4 ([32, 32, 1] = 1024 grids/GPU).

CGA's pros:

- Halo communication in coarse grids, which includes severe bottlenecks, are removed.
- Computational efficiency is improved (problem of latency and imbalance).

CGA's cons:

- The size of the coarse grid problem is larger than that of the original configuration
- Cost of gathering all information to 1 GPU via collective MPI are increased.

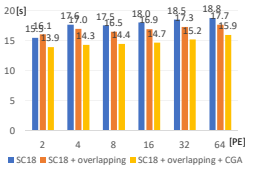
We changed the level on which all information is gather to one GPU, from lev = 4 ([32, 32, 1] = 1024 grids) to lev = 9 ([1, 1, 1] = 1 grids).

Experiment and Evaluation

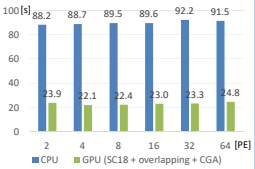
Experimental Settings:

- Each GPU's domain was set to a size of (256, 256, 32)
- two, four, eight, 16, 32, or 64 GPUs or CPUs (weak scaling)
- Idealistic and systematic forcing assuming baloclinic instability

Overlapping and CGA contributed 4% and 12% speedup of the MGCG solver, respectively.
These optimizations are equivalent to 28% reduction of communication of the MGCG solver.



Effects of communication optimization on a MGCG solver

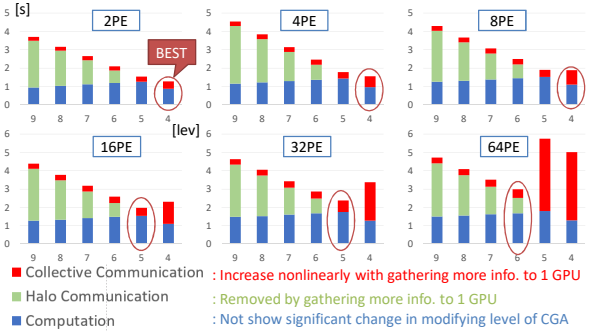


Entire kinaco's performance comparison between CPUs and GPUs.

The entire application is 3.9 times faster than CPUs, with good weak scaling up to 64 GPUs. 3.3 times (SC18) -> 3.9 times

Detail in CGA

Cost distribution in MGCG solver (from lev 4 to 9), Effects of CGA



: Increase nonlinearly with gathering more info. to 1 GPU
: Removed by gathering more info. to 1 GPU
: Not show significant change in modifying level of CGA

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.

Summary and future work

- We implemented and optimized our "kinaco" numerical ocean model with a MGCG solver on multiple NVIDIA P100 GPUs.
- We used two optimization techniques, overlapping of communication with computation and CGA for modification of multigrid aggregation levels.
- The speedup of MGCG solver is 16% faster, which is equivalent to 28% reduction of communication of the MGCG solver.
- We achieved speedup of kinaco 3.9 times compared to CPUs with good weak scaling up to 64 GPUs.
- We discussed inter-GPU communications on a coarse grid on which GPUs could be intrinsically problematic.
- We learned how inter-GPU communications depended on the number of GPUs and the aggregation level of information in a multigrid method.
- In future work, we will adopt the following optimization techniques for detailed experiments with a large number of cells with thousands of GPUs.
  - Hierarchical coarse grid aggregation (hCGA) proposed for CPUs [6] 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.
  - To reduce the MPI communication cost, we will attempt to apply hCGA to hundreds of GPUs and evaluate and analyze in detail.

References

[1] 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
[2] T. Yamagishi, Y. Matsumura and H. Hasumi (2018). Multi-GPU Accelerated Non-Hydrostatic Numerical Ocean Model with GPUDirect RDMA Transfers. Supercomputing Conference 2018.
[3] 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.
[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] K. Nakajima (2014). Optimization of serial and parallel communications for parallel geometric multigrid method. 2014 20th IEEE International.