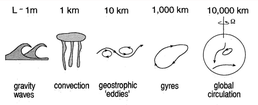
GPU Acceleration of a Non-Hydrostatic Ocean Model Using a Mixed Precision Multigrid Preconditioned Conjugate Gradient Method

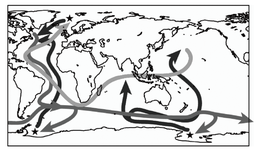
Takateru Yamagishi(1) Yoshimasa Matsumura(2)

1: Research Organization for Information Science and Technology, yamagishi@rist.jp 2: Institute of Low Temperature Science, Hokkaido University, ymatsu@lowtem.hokudai.ac.jp

Numerical Ocean Simulation, non-hydrostatic, high-resolution, to be accelarated

The important role of the ocean:
- Heat and water transport in climate system
- Natural resources like the marine products industry





- Ocean consists of various type and scale processes
- They interact with each other

- Domain is wide
- Local flow interacts with far flow

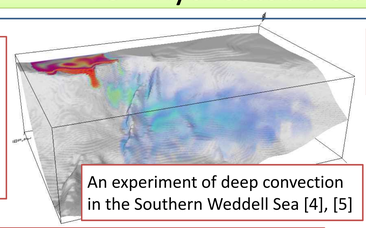
Requirement:
- Simulation with a large number of cells
- Highly parallelized, high performance/throughput
-> GPU must be a candidate to meet the requirement.

Previous studies:
- The applications of the GPU to ocean models are limited to a few previous studies [1, 2]
- Idealistic and simple experiments

Our objectives:
- Study the applicability of ocean model to the GPU
- Make the basic of the experiments to study small oceanic processes

Non-hydrostatic ocean model “kinaco”

kinaco [3] can resolve vertical convection, eddy mixing with non-hydrostatic approximation on ~1 km scale

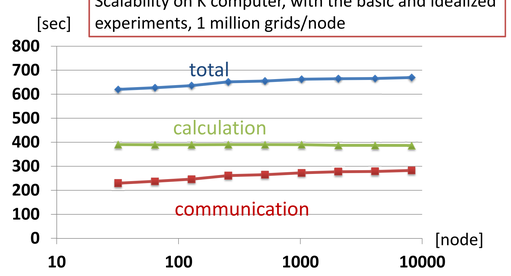


An experiment of deep convection in the Southern Weddell Sea [4], [5]

- 3D Navier-Stokes equation, diffusion-advection equation
- Dynamic LES (Germano et al., 1991) for subgrid scale eddies
- Structured grids, finite-difference method, systematic memory access to adjacent grids
- Poisson/Helmholtz solver to solve pressure and surface height
- Introduction of multigrid preconditioned conjugate gradient (MGCG) method, which is ideal for simulations with the large number of cells

High performance and linear scalability with 10 billion cells on K computer

Scalability on K computer, with the basic and idealized experiments, 1 million grids/node



-> kinaco can be expected to achieve high performance on huge GPU cluster like TSUBAME of Tokyo TEC.

Implementation to the GPU

- kinaco is originally written in Fortran 90 -> PGI CUDA Fortran
- Utilization of abundant data parallelism of ocean simulation

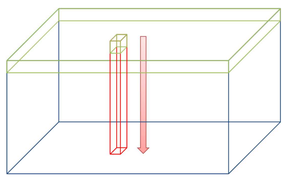
```
DO k=1, n3
  DO j=1, n2
    DO i=1, n1
      mg%out(i, j, k) = mg%out(i, j, k)                &
          + mg%a(-3, i, j, k) * mg%x(i  , j  , k-1)  &
          + mg%a(-2, i, j, k) * mg%x(i  , j-1, k  )  &
          + mg%a(-1, i, j, k) * mg%x(i-1, j  , k  )  &
          + mg%a( 0, i, j, k) * mg%x(i  , j  , k  )  &
          + mg%a( 1, i, j, k) * mg%x(i+1, j  , k  )  &
          + mg%a( 2, i, j, k) * mg%x(i  , j+1, k  )  &
          + mg%a( 3, i, j, k) * mg%x(i  , j  , k+1)
    END DO
  END DO
END DO
```

Coalesced access to global memory, systematic memory access

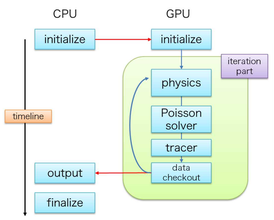
```
i = threadidx%x + blockdim%x * (blockidx%x-1)
j = threadidx%y + blockdim%y * (blockidx%y-1)

DO k=1, n3
  out_d(i, j, k) = out_d(i, j, k)            &
      + a_d(-3, i, j, k) * x_d(i,  j,  k-1)  &
      + a_d(-2, i, j, k) * x_d(i,  j-1,k  )  &
      + a_d(-1, i, j, k) * x_d(i-1,j,  k  )  &
      + a_d( 0, i, j, k) * x_d(i,  j,  k  )  &
      + a_d( 1, i, j, k) * x_d(i+1,j,  k  )  &
      + a_d( 2, i, j, k) * x_d(i,  j+1,k  )  &
      + a_d( 3, i, j, k) * x_d(i,  j,  k+1)
END DO
```

- The switch of the array of structures (AoS) to usual arrays
 -> coalesced access to the global memory
- The elimination of recursive description in multigrid kernel
 -> remove the overhead for launching multigrid kernel

Exploitation of a number of threads and enough instruction parallelism to hide latency



- (256, 256, 32) 3d cells per one GPU
- (256, 256, 1)threads to one GPU
- each 2D thread marching in z-direction



Minimization of data transfer between the CPU host and the GPU device

Application of mixed precision calculation to preconditioning

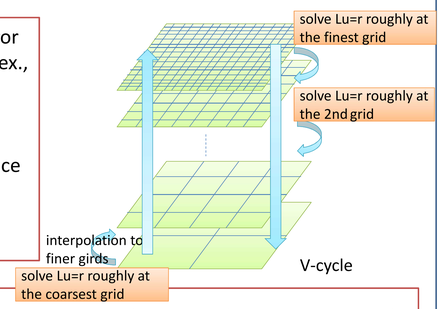
Poisson/Helmholtz equation, solved with multigrid preconditioned conjugate gradient (MGCG) method

MGCG method to solve Lp = q

```
set initial guess p(0)
r(0) = q - Lp(0)
Relax Lu(0) = r(0) using the multigrid method
ρ(0) = (u(0), r(0))
DO n = 0, 1, ⋯ WHILE |r(0)| > ε|q|
   α = ρ(n)/(u(n), Lu(n))
   p(n+1) = p(n) + αu(n)
   r(n+1) = r(n) - αLu(n)
   Relax Lu(n+1) = r(n+1) using the multigrid method
   ρ(n+1) = (u(n+1), r(n+1))
   β = ρ(n+1)/ρ(n)
   u(n+1) = u(n+1) + βu(n)
END DO
```

Dumping high frequency error efficiently in multiple grids (ex., [256x256x32], , , [4x4x1], [2x2x1])

It deteriorated in performance due to the small number of GPU threads in multigrid method.



- All arrays for multigrid preconditioning such as smoother matrix, residuals and temporal arrays are set as four bytes single precision
- The precision of the conjugate gradient method is kept as double precision

Results and Discussion

Settings of Experiment:
- NVIDIA K20C vs FUJITSU SPARC 64 VIIIfx
- PGI 14.10 Accelerator Fortran CUDA 6.5
- Idealized and simple boundary conditions (Visbeck et al. 1996)

General performance: elapsed time[sec]

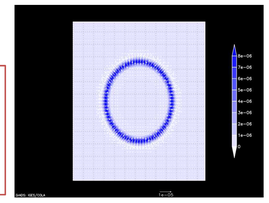
Application	GPU	CPU	Speedup
All	40.5	174.2	4.3
Poisson/Helmholtz solver	13.7	36.8	2.7
Others	26.9	137.4	5.1

basic performance profile

GFLOS of GPU is estimated by the ratio of elapsed time

Metrics	GPU	CPU
GFLOPS	33.0	7.7
GFLOPS/PEAK(%)	2.8	6.0
Mem throughput(GB/s)	n/a	22.2

Snapshot of velocity field , in the experiment on the GPU



No specific error on the GPU execution compared with that on the CPU
The accuracy of CG method is critical to the case.

General performance: elapsed time[sec]

Application	double precision[s]	mixed precision[s]	speedup
Poisson/Helmholtz solver	15.8	13.7	1.16

representative kernel performance: elapsed time[sec]

apply3d3: One of the matrix multiplication kernels in the preconditioning

Kernel	double precision[ms]	mixed precision[ms]	speedup
apply3d3	475.6	229.8	2.1

apply3d3: memory bound kernel, its performance is dependent on memory transfer

```
attributes(global) SUBROUTINE apply3d3_r8(n1, n2, n3, a_d, x_d, out_d)
   INTEGER, value , INTENT(IN)    :: n1, n2, n3
   REAL(8), device, INTENT(IN)    :: a_d (-3:3, 1:n1  , 1:n2  , 1:n3  )
   REAL(8), device, INTENT(IN)    :: x_d (      0:n1+1, 0:n2+1, 0:n3+1)
   REAL(8), device, INTENT(INOUT) :: out (      0:n1+1, 0:n2+1, 0:n3+1)
                                     switch to REAL(4)
i = threadidx%x + blockdim%x * (blockidx%x-1)
j = threadidx%y + blockdim%y * (blockidx%y-1)

DO k=1,n3
  out_d(i, j, k) = out_d(i, j, k)            &
      + a_d(-3, i, j, kk) * x_d(i,  j,  k-1) &
      + a_d(-2, i, j, kk) * x_d(i,  j-1,k  ) &
      + a_d(-1, i, j, kk) * x_d(i-1,j,  k  ) &
      + a_d( 0, i, j, kk) * x_d(i,  j,  k  ) &
      + a_d( 1, i, j, kk) * x_d(i+1,j,  k  ) &
      + a_d( 2, i, j, kk) * x_d(i,  j+1,k  ) &
      + a_d( 3, i, j, kk) * x_d(i,  j,  k+1)
END DO
END SUBROUTINE apply3d3_r8
```

double:
load : 124B
store: 8B
total: 132B

mixed:
load : 64B
store: 4B
total: 68B

B(double) vs B(mixed)= 132/68=1.9
-> consistent with the 2.1 times speedup

Conclusions

- GPU (K20C) implementation: x4.3 faster compared to CPU (SPARC 64 VIIIfx)
- Mixed precision calculation applied to solver: +16% acceleration and no specific error
- This study indicates that an ocean model is suitable to GPU implementation, and it has a potential for further improvement (ex. utilization of shared memory, storing temporal values to registers, kernel fusion and loop unrolling)
- Application of mixed precision to MGCG is an effective method
- Further study: to identify other applicable kernels and to verify them within both the computational science and geophysical disciplines

References

[1] Milakov, M. 2013. Accelerating NEMO with OpenACC. NVIDIA GTC 2013.
[2] van Werkhoven, B., Maassen, J., Kliphuis, M., Dijkstra, H. A., Brunnabend, S. E., van Meersbergen, M., Seinstra, F. J., Bal, H. E. 2013. A distributed computing approach to improve the performance of the Parallel Ocean Program (v2.1). Geoscientific Model Development Discussions. 6, 3, 4705-4744. DOI= http://dx.doi.org/10.5194/gmdd-6-4705-2013
[3] Matsumura, Y., Hasumi, H. 2008. A non-hydrostatic ocean model with a scalable multigrid Poisson solver. Ocean Model. 24, 15-28. DOI= http://dx.doi.org/10.1016/j.ocemod.2008.05.001
[4] Matsumura, Y., Hasumi, H. 2010. Modeling ice shelf water overflow and bottom water formation in the southern Weddell Sea. Journal of Geophysical Research: Oceans. 115, C10033. DOI= http://dx.doi.org/10.1029/2009JC005841
[5] Matsumura, Y., Hasumi, H. 2011. Dynamics of cross-isobath dense water transport induced by slope topography. J. Phys. Oceanogr., 41, 2402-2416, DOI= http://dx.doi.org/10.1175/JPO-D-10-05014.1