Performance Evaluation of OpenSWPC using Various GPU Programming Methods

Tatsumasa Seimi
s2200609@u.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Ibaraki, Japan

Akira Nukada*
nukada@ccs.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Ibaraki, Japan

GPU computing already becomes standard in high-end systems. However, the use of GPU accelerators requires additional programming effort and sometimes tuning effort. There are several easy-to-use ways of GPU programming. For NVIDIA GPU, OpenACC and OpenMP (target) are directive-based approaches. C++ and Fortran standard now includes parallelism (stdpar and do concurrent), and several compilers can generate GPU code for these parallel parts. CUDA Fortran also includes directive-based kernel generation (`!$cuf kernel do`), and can generate GPU code with minimum modifications in combination with unified memory.

We use OpenSWPC [1], a Seismic Wave Propagation Code, as a target application. The original Fortran code is designed for CPUs, and it is parallelized with MPI and OpenMP. The main computation of the OpenSWPC is stencil-based, but it is not uniform around the surface and sea floor. We converted the OpenMP parallel regions into GPU codes. The original OpenMP code parallelize only outer loop of nested loops. For GPU, we need to use inner loops to exploit further parallelism. We merge these nested multiple loops into single GPU kernel to reduce the overheads of kernel launches and memory accesses.

The standard parallelism and CUDA Fortran do not allow any subroutine and function calls in GPU kernels. We need to use inlining them.

The unified memory makes GPU programming easier, however MPI communication using unified memory has critical performance issue [2]. It is better to use the device memory for buffer memory if GPUDirect RDMA is supported. OpenACC, OpenMP, and CUDA Fortran can allocate device memory explicitly. Since the standard parallelism employs unified memory, it requires combination of CUDA Fortran and OpenACC features as shown in Figure 1.

```
real(MP),private,device,allocatable::sbuf_ip(:)

!$acc data deviceptr(sbuf_ip,sbuf_im)
do concurrent(j=jbeg:jend,k=kbeg:kend) local(ptr)
  ptr=(k-kbeg)*Nsl+(j-jbeg)*Nsl*(kend-kbeg+1)+1

  sbuf_ip(ptr:ptr+Nsl-1)=Vx(k,iend-Nsl+1:iend,j)
end do
!$acc end data

call mpi_isend(sbuf_ip,s_isize,mpi_precision,itbl(idx+1,idy),1,
     mpi_comm_world,ireq1(1),ierr)
```

Figure 1: Explicit use of device memory for MPI communication in standard parallelism

The porting costs in the four GPU programming methods are almost equivalent. Fortunately, inlining subroutines is easy by using preprocessor macro. Inserting directives for OpenACC and OpenMP are easier than replacing do loops by do concurrent constructs. OpenACC and OpenMP without unified memory require explicit control of data transfer between host and device. We need to make a list of variables accessed by GPU kernels.

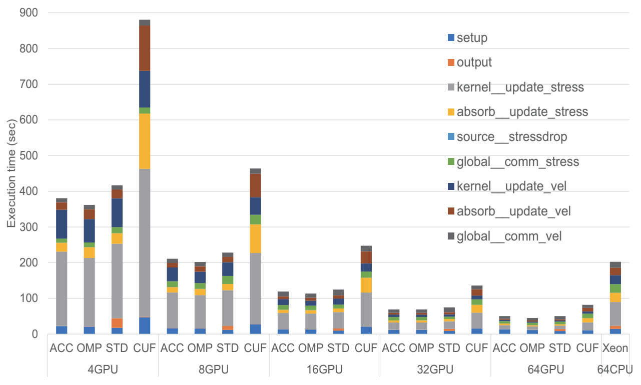

Figure 2: Execution times for Odawara earthquake data

Figure 2 shows the performance (execution times) using Pegasus supercomputer (Sapphire Rapids Xeon + NVIDIA H100 PCIe). Inlining subroutines is applied for all cases. The nvfortran compiler of NVIDIA HPC SDK 23.1 is used with OpenMPI 4.1.5. Performance with OpenACC (ACC), OpenMP (OMP) and standard parallelism (STD) are almost same. However, CUDA Fortran (CUF) increases kernel execution times. This requires further investigations. OpenACC and OpenMP do not require inlining of subroutines, however it increases kernel execution times by 10% since the subroutine accesses module variables directly.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number JP20K19807, and "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN)" and "High Performance Computing Infrastructure (HPCI)" in Japan (Project ID: jh230037). The authors would like to thank Mr. Naruse at NVIDIA for his valuable advice.

REFERENCES

[1] Takuto Maeda, Shunsuke Takemura, and Takashi Furumura. 2017. OpenSWPC: An open-source integrated parallel simulation code for modeling seismic wave propagation in 3D heterogeneous viscoelastic media. *Earth, Planets and Space* 69, 102 (2017), 1–20. https://doi.org/10.1186/s40623-017-0687-2

[2] Akira Naruse, James D. Trotter, Johannes Langguth, Xing Cai, and Kengo Nakajima. 2022. High resolution simulation of cardiac electrophysiology on realistic whole-heart geometries on Wisteria/BDEC-01 Aquarius. In *SIG Technical Reports (HPC)*, Vol. 2022-HPC-187. IPSJ, 1–7. http://id.nii.ac.jp/1001/00222549/