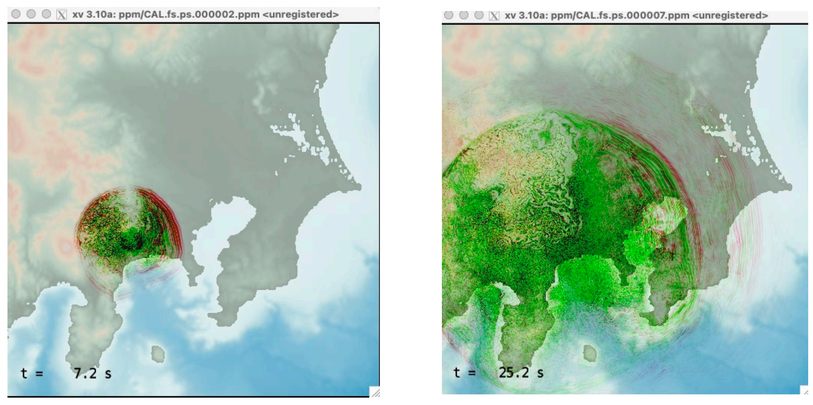
Performance Evaluations of OpenSWPC using Various GPU Programming Methods

Tatsumasa Seimi and Akira Nukada (University of Tsukuba, Japan)

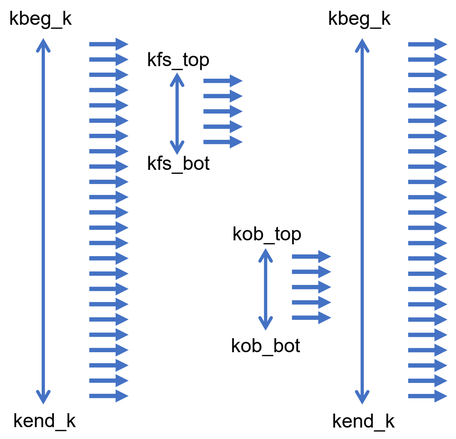
OpenSWPC (Seismic Wave Propagation Code)

OpenSWPC is an open-source seismic wave propagation code developed by Furumura, et al. The code is written in Fortran 90 and parallelized by OpenMP and MPI. We offload all OpenMP parallel regions in main loop to GPU.



Common optimization

In the main loop of the OpenSWPC code, velocity and stress values are updated alternately. The computation is like 3D-stencil, but it is not uniform around sea floor and surface. To exploit massive parallelism of GPU, we need to modify the complex nested loops.




```
do i=ibeg, iend
  do j=jbeg, jend
    do k=kbeg_k, kend_k
      d3Sx3(k) = …
    end do
    do k=kfs_top(i,j),kfs_bot(i,j)
      d3Sx3(k) = …
    end do
    do k=kob_top(i,j), kob_bot(i,j)
      d3Sx3(k) = …
    end do
    do k=kbeg_k, kend_k
      Vx(k,i,j) = …  d3Sx3(k) …
    end do
  end do
end do
```

```
do concurrent (i=ibeg:iend, j=jbeg:jend, k=kbeg_k:kend_k) local(d3Sx3)
  d3Sx3 = …
  if (k>=kfs_top(i,j) .and. k<=kfs_bot(i,j)) then
    d3Sx3 = …
  end if
  if (k>=kob_top(i,j) .and. k<=kob_bot(i,j)) then
    d3Sx3 = …
  end if
  Vx(k,i,j) = …  d3Sx3 …
end do
```

GPU Programming Methods for Fortran

NVIDIA CUDA was released in 2006, as the first programming environment for general-purpose computation using GPU. It can exploit full features of GPU; however, the coding is still difficult for many researchers in applied sciences.
As easy-to-use GPU programming methods, currently four methods are available.

(1) OpenACC

OpenACC is a directive-based extension to generate GPU code easily.

```
INTEGER, ALLOTABLE :: V(:)
INTEGER :: I, N

!$acc parallel loop copy(V)
DO I=1,N
   V(I)=V(I)+1
END DO
```

(2) OpenMP

OpenMP target directive generates GPU code easily. It works not only NVIDIA GPU but also AMD and Intel.

```
INTEGER, ALLOTABLE :: V(:)
INTEGER :: I, N

!$omp target loop map(tofrom:V)
DO I=1,N
   V(I)=V(I)+1
END DO
```

(3) Standard parallelism

Fortran 2008 includes standard parallelism for multithreading using DO CONCURRENT constructs. NVIDIA compiler can generate GPU code for those constructs.

```
INTEGER, ALLOTABLE :: V(:)
INTEGER :: I, N

DO CONCURRENT (I=1:N)
   V(I)=V(I)+1
END DO
```

(4) CUDA Fortran

CUDA Fortran is a low-level programming model like CUDA C/C++. Using CUDA managed memory and CUF kernels directives, we can develop GPU code in similar effort to the other directive-based methods.

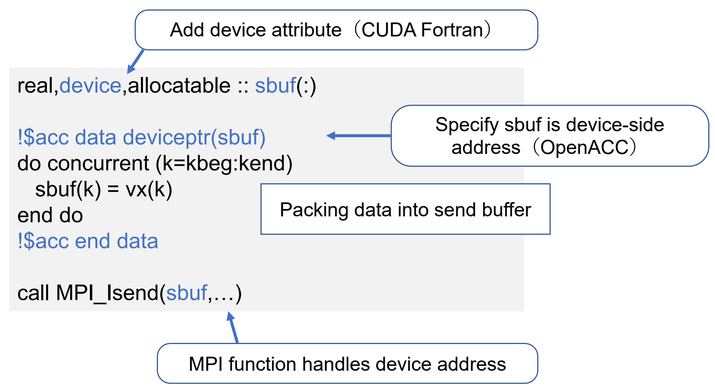
```
INTEGER, ALLOTABLE :: V(:)
INTEGER :: I, N

!$cuf kernel do <<<*,*>>>
DO I=1,N
   V(I)=V(I)+1
END DO
```

Code conversion

If you know the code in detail, the coding costs in the four methods are almost same. For GPU kernels, OpenACC, OpenMP and CUDA Fortran insert necessary directives. On the other hand, standard parallelism rewrite DO loops into DO CONCURRENT loops. We think inserting directives is easier.
Since standard parallelism and CUDA Fortran (in our method) use unified memory, we don't need to check which variables are used in GPU kernels. For OpenACC and OpenMP, without unified memory, we need to manually control the data transfers between host and device. This task will be hard for large-scale applications.

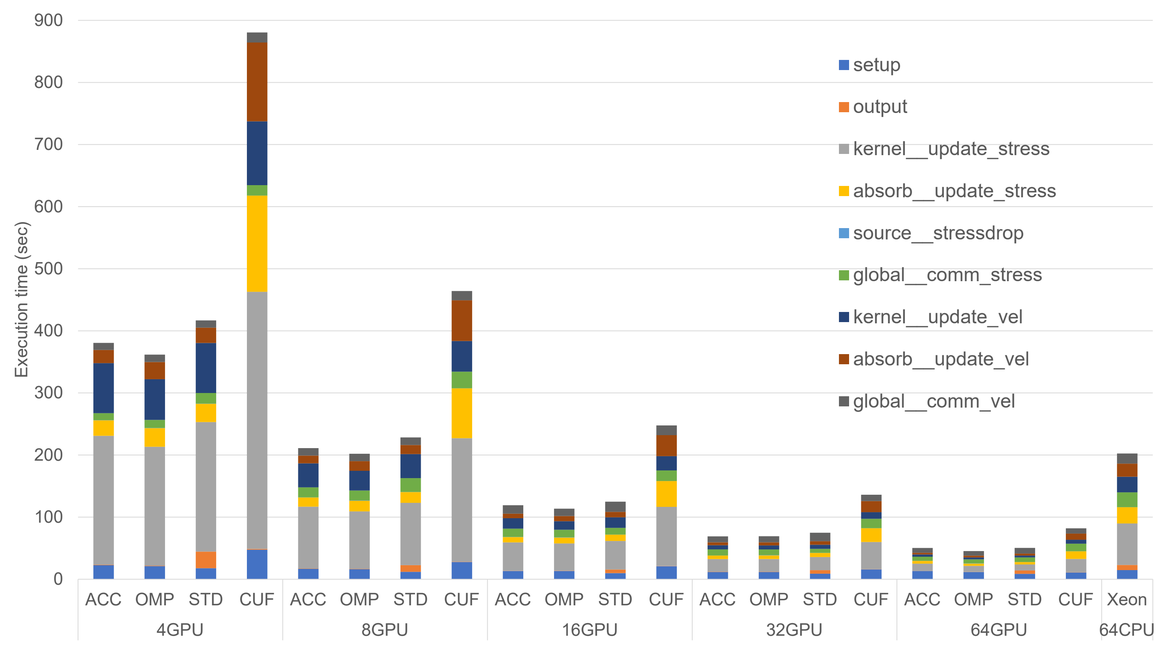
Communication buffer

As we are converting OpenMP + MPI hybrid program into GPU program, the communication between GPU is performed by the MPI library. For systems with GPUDirect RDMA support, it is better to use device memory for communication buffers. The unified memory is the worst, because it is impossible to pin down it for efficient data transfer.
OpenACC, OpenMP and CUDA Fortran can use device memory explicitly. In case of standard parallelism, we need to employ CUDA Fortran and OpenACC features to use device memory.



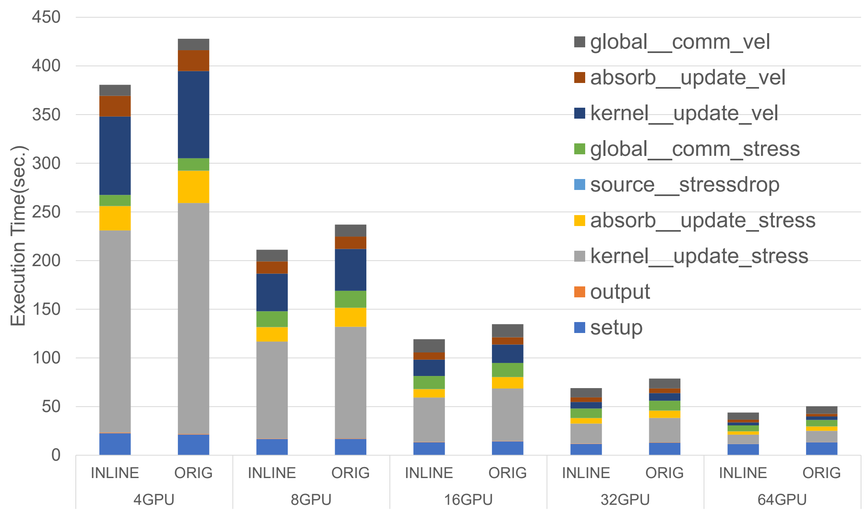
Performance Evaluations

We evaluate the performance of OpenSWPC using four GPU programming methods, and Pegasus supercomputer at CCS, U. Tsukuba (Sapphire Rapids + NVIDIA H100 PCIe, NVHPC 23.1). We use Odawara earthquake data (strong scaling). Performance of OpenACC, OpenMP and standard parallelism are almost comparable, but CUDA Fortran shows much lower. Although generated PTX kernels are similar to those of the other methods, number of registers becomes much larger, which may degrade occupancy and efficiency of GPU kernels.



Subroutine Calls

The original OpenSWPC requires subroutine calls in GPU code. Standard parallelism and CUDA Fortran need inlining of those subroutines. OpenACC and OpenMP allows subroutine calls by using directives, however it may degrade the performance. The subroutine calls are included in "output" phase which consumes negligible time. On the other hand, it extends the elapsed times of all the other kernels.



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.