The MOM6 dynamic core is successfully transitioning to GPUs while maintaining bit-for-bit reproducibility. By leveraging Fortran intrinsics and OpenMP offload for GPU execution, significant performance gains are achieved with reduced code refactoring. This hybrid approach allows the code to remain readable and portable, avoiding vendor-specific frameworks.

ACCESS National Research Infrastructure
access-nri.org.au

Porting MOM6 to GPUs with Fortran Intrinsics and OpenMP

Edward Yang[1], Marshall L. Ward[2], Jorge L. Galvez Vallejo[3], Micael J. T. Oliveira[1]

[1] ACCESS-NRI, [2] NOAA-GFDL, [3] National Computing Infrastructure (NCI)

Introduction

MOM6 is the latest edition in the MOM series of ocean simulation Fortran codes and is used in Australia's ACCESS climate models and weather forecasting. Despite its importance, it still relies on CPUs only. MOM6 GPU acceleration is crucial for future ACCESS models to leverage the GPU-heavy, next-generation Australian HPCs.

Fortran remains a useful language for physical scientists and engineers, but historically, has had few options for GPU acceleration that is performance portable and doesn't require a high maintenance burden.

In recent years, the GPU vendors have improved GPU support for Fortran's "do concurrent" loop construct and it now presents as a good candidate for GPU acceleration with reduced changes to existing code. Similarly, all major GPU vendors support OpenMP, thus paving a path for porting large Fortran codebases, like MOM6, while minimizing changes, ensuring maintainability and portability.

Porting Strategy

We use Fortran's intrinsic do concurrent to offload compute to GPU. However, for good performance on discrete GPUs, CPU-GPU data transfers are facilitated by OpenMP directives. NVIDIA, AMD, Intel GPUs support this workflow.

```
! Transfer data from CPU to GPU
!$omp target enter data &
!$omp   map(to: array1, array2) map(alloc: tmparray)

do concurrent (i = istart:iend, j = jstart:jend)
! ... do some compute on GPU
enddo

! Transfer results back to CPU
!$omp target exit data &
!$omp   map(from: array1, array2) map(release: tmparray)
```

Some loops are offloaded using OpenMP because nvfortran doesn't yet support private array variables with do concurrent, but its OpenMP implementation does.

```
!$omp target teams loop private(priv_arr) ...
do i = istart, iend
! ... do some compute on GPU
enddo
```

GPU-to-GPU MPI communications are performed using OpenMP directives and an MPI library built with GPU support.

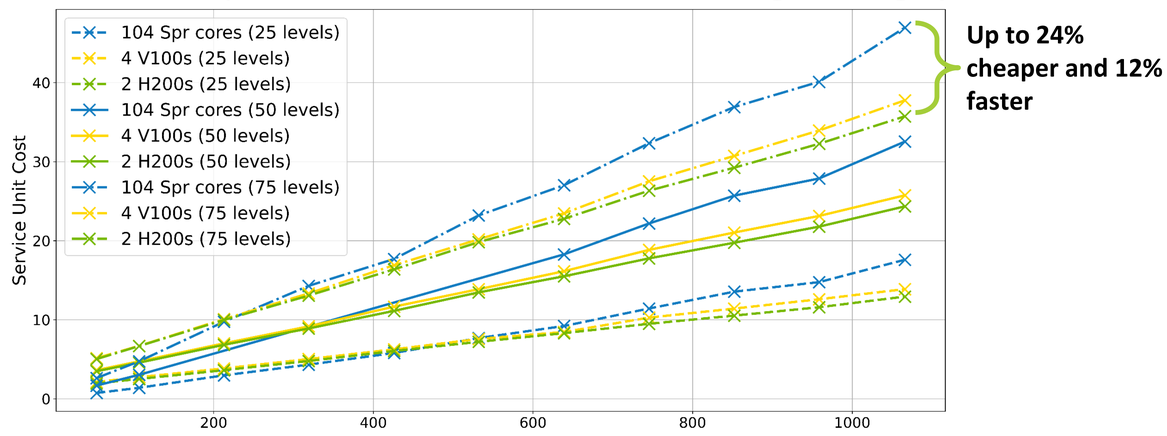
```
!$omp target data use_device_addr(buffer)
call MPI_XXX(buffer, ...)
!$omp end target data
```

Progress and Challenges

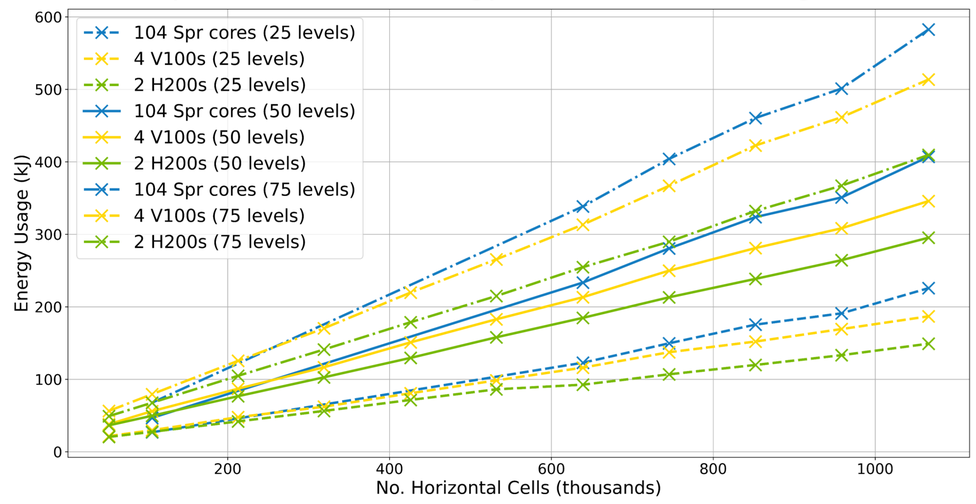
- **Enough of the dynamic core has been ported such that the "double gyre" example can be run on multiple GPUs using CUDA-aware MPI.**
  - Migration effort significantly lower than other porting strategies.
- **Some refactors were needed.**
  - Nested loops, where nesting crosses routine boundaries.
  - Loops that use type-bound procedures.
- **At the bleeding edge of compiler development**
  - Multiple GPU-related compiler bugs have been found
- **Incomplete support for modern Fortran constructs on GPU**
  - Derived types and pointers can cause segfaults

Results So Far

Now cheaper and faster to run on GPUs for large-enough problems



Up to 30% more energy efficient when using recent GPUs



All tests run on NCI Gadi HPC, where for 1h walltime, 104 "Spr" cores (1 node - 2x Intel 8470Q) cost 208 Service Units, 2 H200 NVIDIA GPUs cost 180, and 4 V100 NVIDIA GPUs cost 144.
The Service Unit (SU) is the charging unit that NCI uses to account for usage of compute services.
CPU tests using code prior to adding any GPU changes
CPU binary compiled with "ifx -O3 -flto -march=sapphirerapids -mtune=sapphirerapids"
GPU binary compiled with "nvfortran -O4 -Mnofma -Mnovect -mp=gpu -stdpar=gpu"
Times reported are from best of 5 runs
Energy usage measured using Intel RAPL and nvidia-smi and averaged across 5 runs
All tests run using double precision for all floating-point calculations

Conclusions and Future Work

- **OpenMP + do concurrent is enough for MOM6 to be worth running on GPUs, while maintaining bitwise reproducibility, and without needing to duplicate code.**
  - In some cases, there is a trade-off between CPU and GPU performance.
- **More of the MOM6 will be ported.**
  - Double gyre covers the dynamic core of MOM6, but other physics still must be ported.
  - Aiming to port the more complete "benchmark" example next.
- **Further optimisations are needed.**
  - GPU is underutilized when running tests
  - Halo exchange performance is limited by use of cray pointers, which is not well-supported by nvfortran.
- **Non-NVIDIA GPUs need to be tested and support improved.**
  - Have currently only tested on NVIDIA GPUs.

Australian National University · Australian Government Bureau of Meteorology · CSIRO · Monash University · University of Melbourne · UNSW Sydney · University of Tasmania

NCRIS National Research Infrastructure for Australia
An Australian Government Initiative