FEBRUARY 6, 2025

Status update on the software developments of the TURBO project

John Dennis, Alper Altuntas, Mike Levy, Jian Sun, Lalo Torres, Michael Waxmonsky

Supported through the Cyberinfrastructure for Sustained Scientific Innovation (CSSI), and the Community Software Facility (CSF)

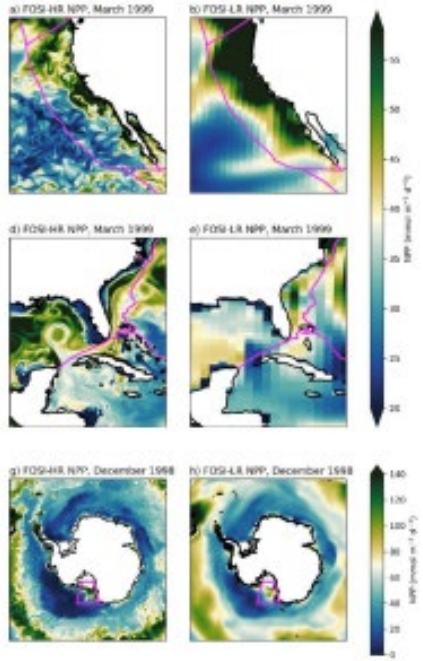
This material is based upon work supported by the NSF National Center for Atmospheric Research, a major facility sponsored by the U.S. National Science Foundation and managed by the University Corporation for Atmospheric Research. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF.

TURBO: Towards Ultra-High Resolution Community Earth System Model (CESM) with MOM6 and Ocean Biogeochemistry

- Overall goals:
  - Enable MOM6 to efficiently utilize both CPU and GPU-based systems
  - Enable cutting edge oceanography at 1/12° and 1/36° global resolution
  - Enable ocean eddy-permitting/resolving coupled CESM simulations
  - Gain access to **all** leadership class systems at NSF and DOE
- 5-year NSF Cyberinfrastructure for Sustained Scientific Innovation (CSSI)
- Collaboration: CGD, CISL, and Texas A&M University (TAMU), and GFDL
- **7** months into 60 month project

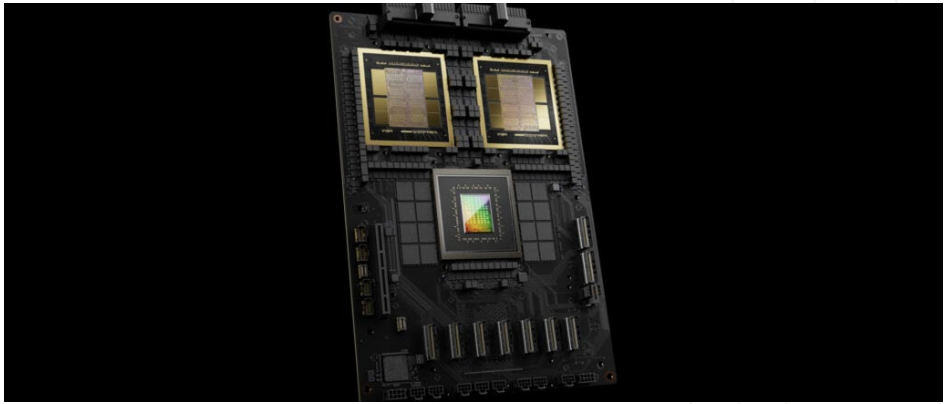
The TURBO Team

- Principle Investigator: Gokhan Danabasoglu
- CGD: Oceanography and community software support/development
  - Alper Altuntas, Frederic Castruccio, Gokhan Danabasoglu, Mike Levy, Keith Lindsay, Gustavo Marques
- CISL: GPU-enablement and optimization
  - John Dennis, Jian Sun, Lalo Torres, Michael Waxmonsky
- Texas A&M University (TAMU): Educational training
  - Ping Chang, Jian Tao



NSF Horizon system

- $250 Million investment
- Located at Texas Advanced Computing Center
- 4752 nodes of Vera-Vera, 176 cores-per-node of ARM
  - 836,352 cores [~20% of system capability]
- 2000 nodes of Grace-Blackwell 72 cores-per-node of ARM
  - 144,000 cores & 4000 GPUs [~80% of system capability]



Motivating use-case

1. Idealized Ocean-Only Simulations using Neverworld2 (1/4°, 1/8°, 1/16°, 1/32°)
2. Forced Ocean – Sea-ice simulation high-resolution (1/12°)
3. Fully coupled high-resolution (0.25° ATM and 1/12° ocean)
4. Forced Ocean - Sea-ice simulation ultra-high-resolution (1/36°)
5. Fully coupled ultra-high-resolution (3 km atmosphere and 1/36° ocean)

Use case (4) and (5) requires Leadership Class Computing

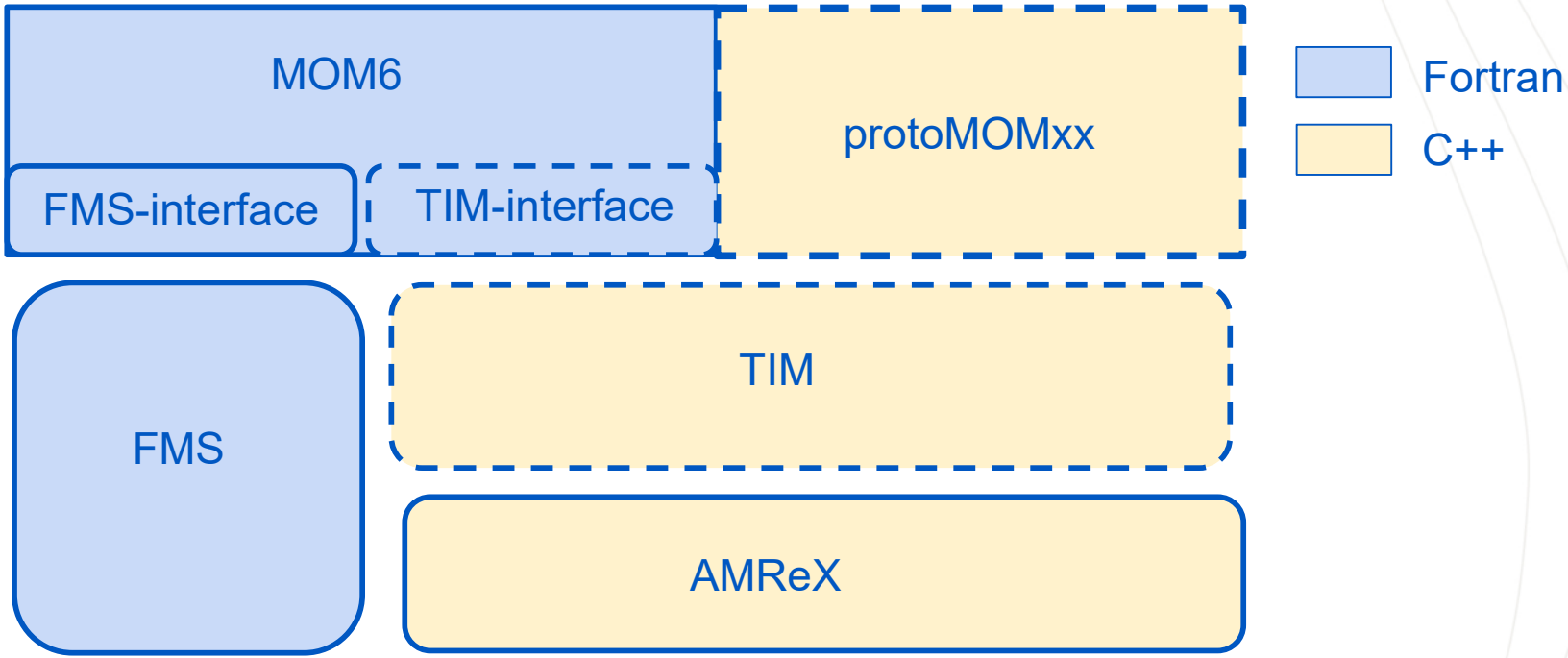
TURBO Software approach

- Evolve MOM6 code base such that it can efficiently utilize CPU and GPU based systems
- Create an alternative for Flexible Model System (FMS) that is GPU-enabled.
- Evaluate the viability of a pivot away from Fortran towards an alternative language approach for earth system modeling
  - Decide path forward at end of year -2 of project
- Why pivot away from Fortran?
  - Limited/lower quality vendor support versus C++/Python
  - Buggy support for Fortran on GPU → Delayed access to new hardware
  - Limited software tooling
  - Workforce challenges associated with Fortran

If not Fortran than what?

- C++ productivity framework: AMReX
  - AMReX is a software framework for block structured AMR (LBNL)
  - Exascale Computing Project (ECP) framework used by a number of Applications
    - Combustion, Astrophysics, Cosmology, Multiphase flow, Accelerators
  - Supports CPU and GPU
  - Assumes logically rectangular grid
  - Provide extensive infrastructure for I/O & message passing
  - Long history of Fortran compatibility
- AMReX provides more infrastructure versus Kokkos

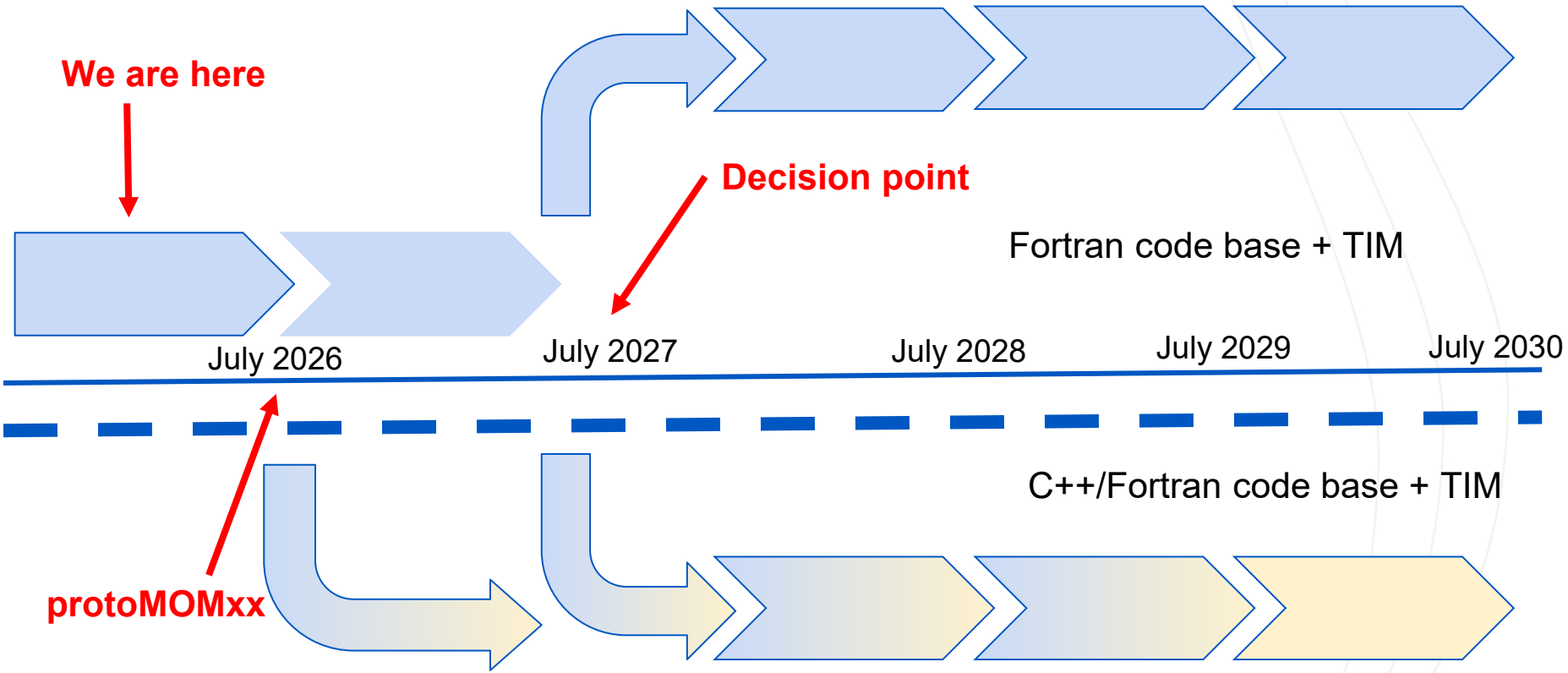
TURBO Infrastructure for MOM6 (TIM)



Relevant TURBO software

- AMReX (Adaptive mesh refinement for the Exascale) [Exists]
- FMS (Flexible Modeling System) [Exists]
- FMS-interface (MOM6/config_src/infra/FMS2) [Exists]
  - Interface between MOM6 and FMS library
- TIM-interface (MOM6/config_src/infra/TIM) [To be written]
  - Interface between MOM6 and TIM library
  - Provides same interface to MOM6 as FMS-interface
  - MOM6 can be linked against TIM-interface or FMS-interface
- TIM [To be written]
  - Alternative to FMS
  - Based on AMReX
- protoMOMxx: MOM6 C++-based prototype [To be written]
  - Implements the double-gyre testcase (with rewritten MOM6 initialization)

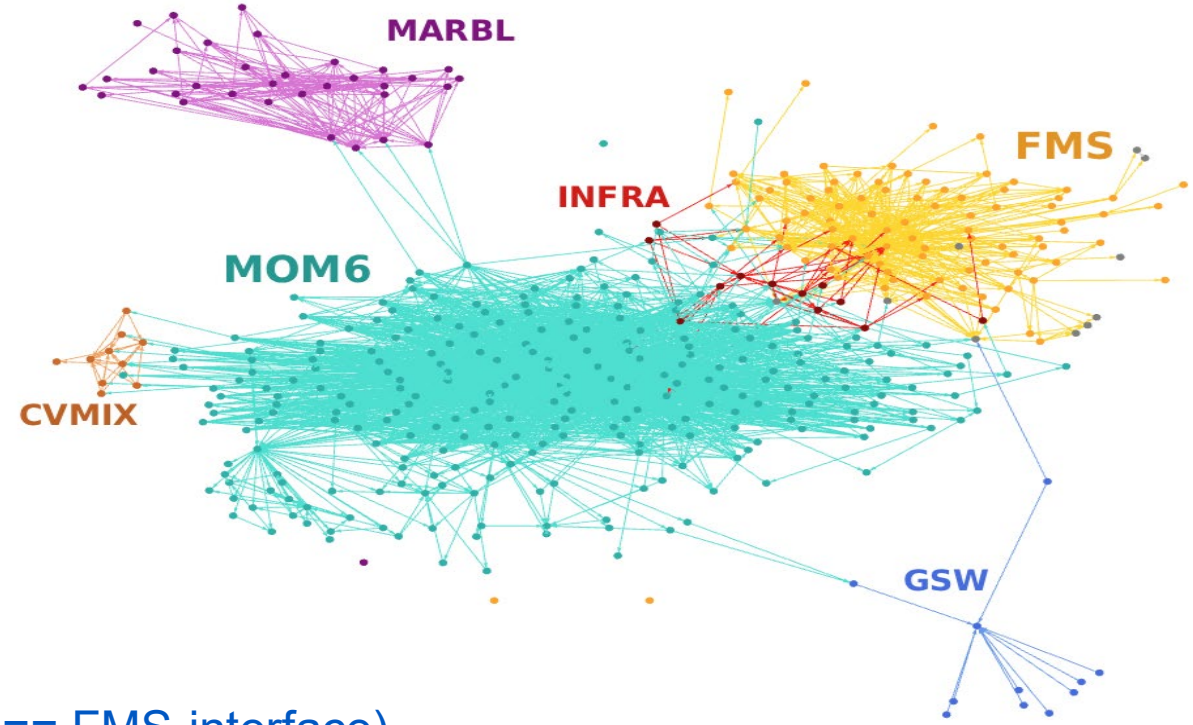
Timeline of 5 year project



Current status of project

- Multiple different MOM6 configurations of increasing complexity defined
- Integration of existing MOM6 build with CMake
- Robust CI
  - MOM6
  - {FMS,TIM}-interface
  - C++ prototype
- pFUnit for {FMS,TIM} - interface [18 unit tests]
- GTest for C++ prototype [15 unit tests]
- Understanding of MOM6 module dependencies
- Reduce complexity version of FMS2

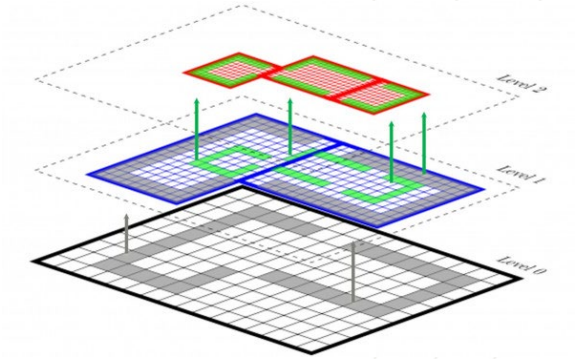
MOM6 + dependent software (modules)



(INFRA == FMS-interface)

MOM6 C++-based prototype (protoMOMxx)

- Small prototype that solves the double-gyre testcase
- Will utilize a C++ driver based on AMReX
- Will include
  - Partial implementation of TIM using AMReX
  - Fortran functions to test multi-language code base
  - C++ functions to illustrate potential future code base
- CPU and GPU enabled
- May include static mesh refinement capabilities
  - Block structured (Berger–Colella)



https://cs.lbl.gov/news-and-events/news/2023/amrex-a-performance-portable-framework-for-block-structured-adaptive-mesh-refinement-applications/

Questions?

Contact: dennis@ucar.edu

Supported through the Cyberinfrastructure for Sustained Scientific innovation (CSSI) and the Community Software Facility (CSF)

pFunit test example

```
@test(npes=[4])
subroutine test_numPEs(this)
  class (MOM_comms_infra_test_case), intent(inout) :: this
  integer :: npesTIM
  integer :: npes
  if (.not. initialized) then
     call MOM_infra_init(this%getMpiCommunicator())
     initialized = .true.
  end if
  npesTIM = num_PEs()
  npes = this%getNumProcesses()
  @assertEqual(npes, npesTIM)
end subroutine test_numPEs
```

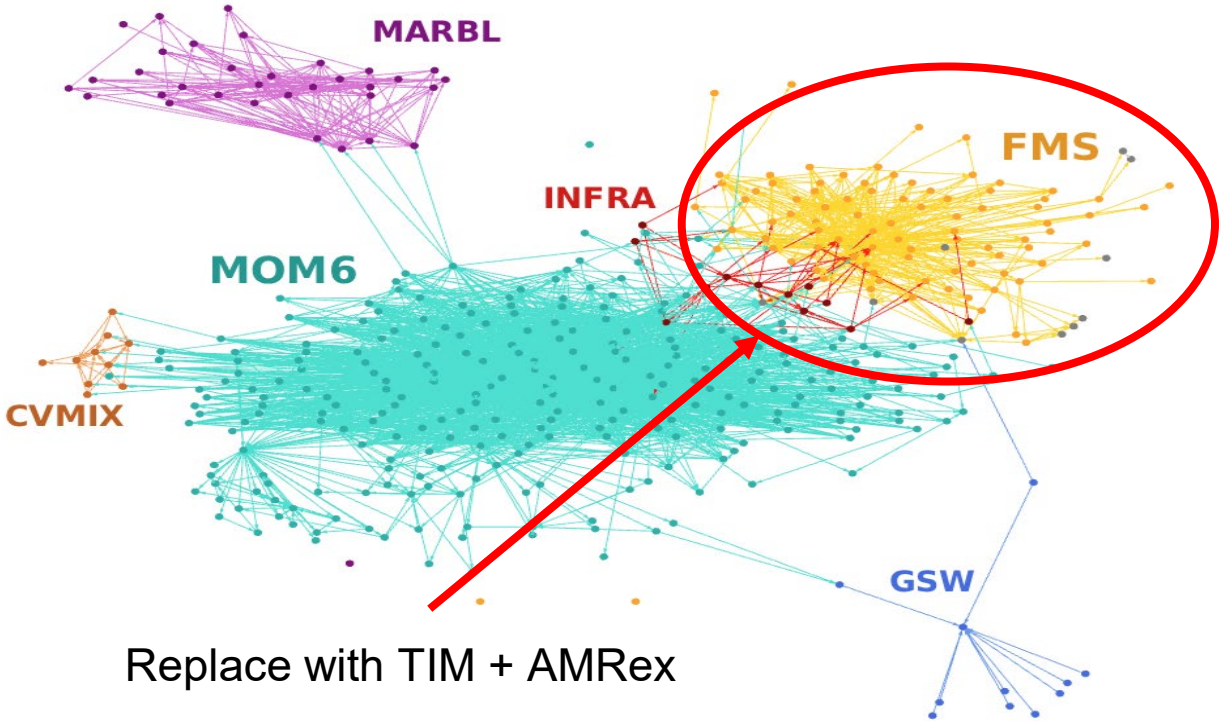
Run this test on 4 MPI ranks

Initialize FMS/TIM

Get number of MPI ranks

Make sure it matches the expected #

MOM6 + dependent software (modules)



Replace with TIM + AMRex

pFunit testing {FMS/TIM} - interface

				Passes	
	Routines	Ready	Unit-test	FMS	TIM
Overall	229	21	4	4	0
double-gyre	77	11	1	1	0

Continuous Integration for MOM6

	MPI	Compilers			
		nvhpc	oneapi	gcc14	clang
Ubuntu (CPU)	mpich	✅	✅	✅	✅
	openmpi		✅	✅	✅
Gha-runner (CPU)	mpich	✅	✅	✅	✅
	openmpi		✅	✅	✅

Continuous Integration for C++ prototype

	MPI	Compilers			
		nvhpc	oneapi	gcc14	clang
Ubuntu (CPU)	mpich		✅	✅	✅
	openmpi		✅	✅	✅
Gha-runner (CPU)	mpich		✅	✅	✅
	openmpi		✅	✅	✅

Continuous Integration for FMS/TIM - interface unit - tests

	MPI	Compilers			
		nvhpc	oneapi	gcc14	clang
Ubuntu (CPU)	mpich		✅	✅	
	openmpi				
Gha-runner (CPU)	mpich		✅	✅	
	openmpi				

AMReX code example (Kernel approach)

```
for (MFIter mfi(p); mfi.isValid(); ++mfi)
{
      const Box& bx = mfi.validbox();
      ParallelFor(bx, [=] AMREX_GPU_DEVICE(int i, int j)
        {calc_cuK(i,j,p.const_array[mfi], u.const_array[mfi], cu.const_array[mfi]);});
}
```

AMReX C++ driver

```
AMREX_GPU_DEVICE AMREX_FORCE_INLINE
void calc_cuK( const int i, const int j,
        const Array2<Real const>& p,
        const Array2<Real const> & u,
        const Array2<Real const>& cu)
{
cu(i,j) = 0.5*(p(i,j) + p(i+1,j))*u(i,j);
}
```

AMReX C++ kernel

or

```
subroutine calc_cuK(i,j,p,u,cu)
   Integer :: i,j
   real, dimension(:,:), pointer :: p,u,cu
      cu(i,j) = 0.5*(p(i,j) + p(i+1,j))*u(i,j);
end subroutine calc_cucvKernel
```

Fortran kernel

AMReX code example (subroutine approach)

```
for (MFIter mfi(p); mfi.isValid(); ++mfi)
{
      const Box& bx = mfi.validbox();
      calc_cuS(BL_TO_FORTRAN_BOX(bx),
            BL_TO_FORTRAN_ANYD(p[mfi]),
            BL_TO_FORTRAN_ANYD(u[mfi]),

BL_TO_FORTRAN_ANYD(cu[mfi]));
}
```

AMReX driver

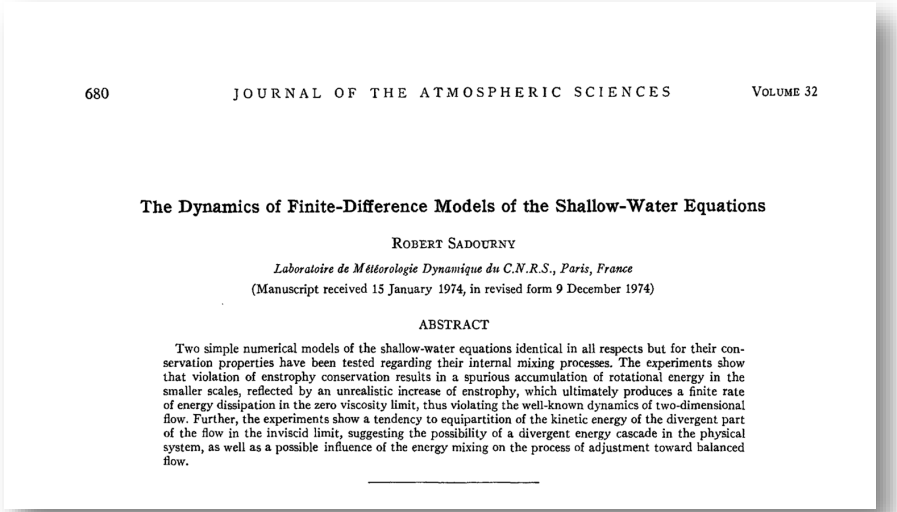
```
subroutine calc_cuS(lo,hi, p,p_lo,p_hi, u, u_lo,u_hi, &
                                    cu,cu_lo,cu_hi)
  integer, dimension(2) :: lo, hi, p_lo,p_hi,
  integer, dimension(2) :: u_lo,u_hi, cu_lo,cu_hi
  real :: p(p_lo(1):p_hi(1),p_lo(2):p_hi(2))
  real :: u(p_lo(1):p_hi(1),p_lo(2):p_hi(2))
  real :: cu(cu_lo(1):cu_hi(1),cu_lo(2):cu_hi(2))
  !$acc parallel deviceptr(p,u,cu)
  !$acc loop gang vector collapse(2)
  do j=lo(2), hi(2)
  do i=lo(1), hi(1)
     cu(i,j) = 0.5*(p(i,j) + p(i+1,j))*u(i,j);
  enddo
  enddo
  !$acc end parallel
end subroutine calc_cuS
```

OpenACC directives

Fortran subroutine

Same application written in:

- C w/OpenACC
- Fortran w/OpenACC
- Python Numpy
- Python gt4py
- C++ framework: AMReX



680 JOURNAL OF THE ATMOSPHERIC SCIENCES VOLUME 32

The Dynamics of Finite-Difference Models of the Shallow-Water Equations

ROBERT SADOURNY

Laboratoire de Météorologie Dynamique du C.N.R.S., Paris, France

(Manuscript received 15 January 1974, in revised form 9 December 1974)

ABSTRACT

Two simple numerical models of the shallow-water equations identical in all respects but for their conservation properties have been tested regarding their internal mixing processes. The experiments show that violation of enstrophy conservation results in a spurious accumulation of rotational energy in the smaller scales, reflected by an unrealistic increase of enstrophy, which ultimately produces a finite rate of energy dissipation in the zero viscosity limit, thus violating the well-known dynamics of two-dimensional flow. Further, the experiments show a tendency to equipartition of the kinetic energy of the divergent part of the flow in the inviscid limit, suggesting the possibility of a divergent energy cascade in the physical system, as well as a possible influence of the energy mixing on the process of adjustment toward balanced flow.

https://github.com/NCAR/SWM

AMReX has good CPU performance

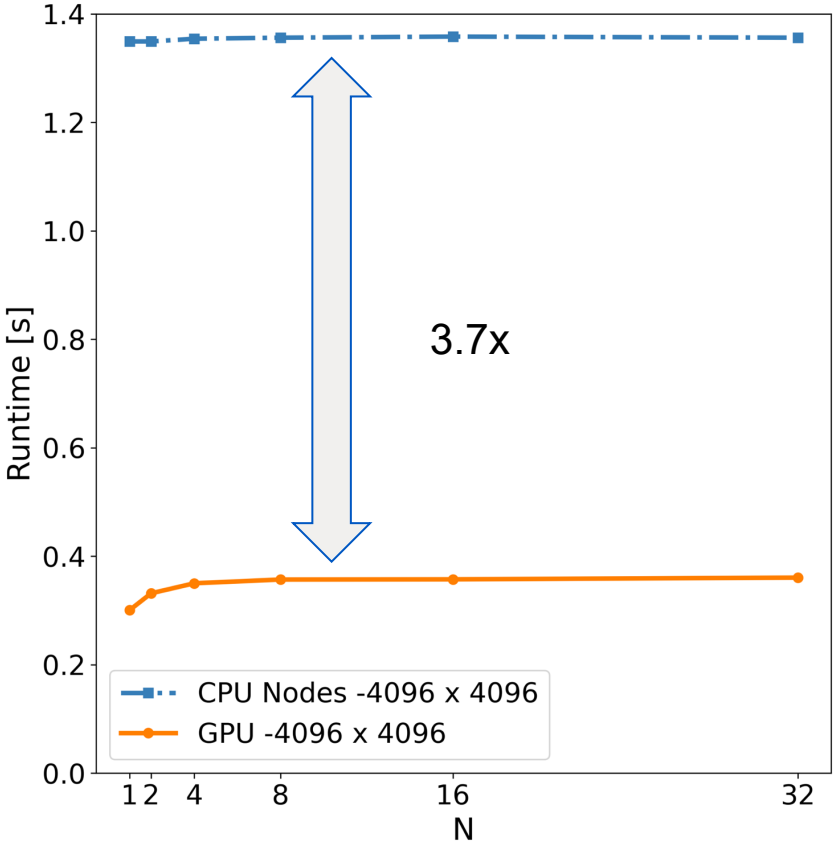
Version/problem size	64x64	128x128	256x256	512x512	1024x1024
C	0.08	0.44	1.44	14.16	59.51
Fortran	0.14	0.64	2.67	11.97	49.12
AMReX Ckernels	0.17	0.53	1.95	9.73	40.21
AMReX Fkernels	0.16	0.53	1.95	9.77	40.91
AMReX Fsubroutine	0.15	0.48	1.77	8.85	37.18
Python (numpy)	1.69	3.34	9.85	34.42	230.18
GT4Py (numpy)	3.59	5.07	12.38	40.21	236.39
GT4Py (gt:cpu_ifirst)	0.75	2.29	3.61	14.25	77.43
Julia	5.20	5.56	8.25	26.10	95.35

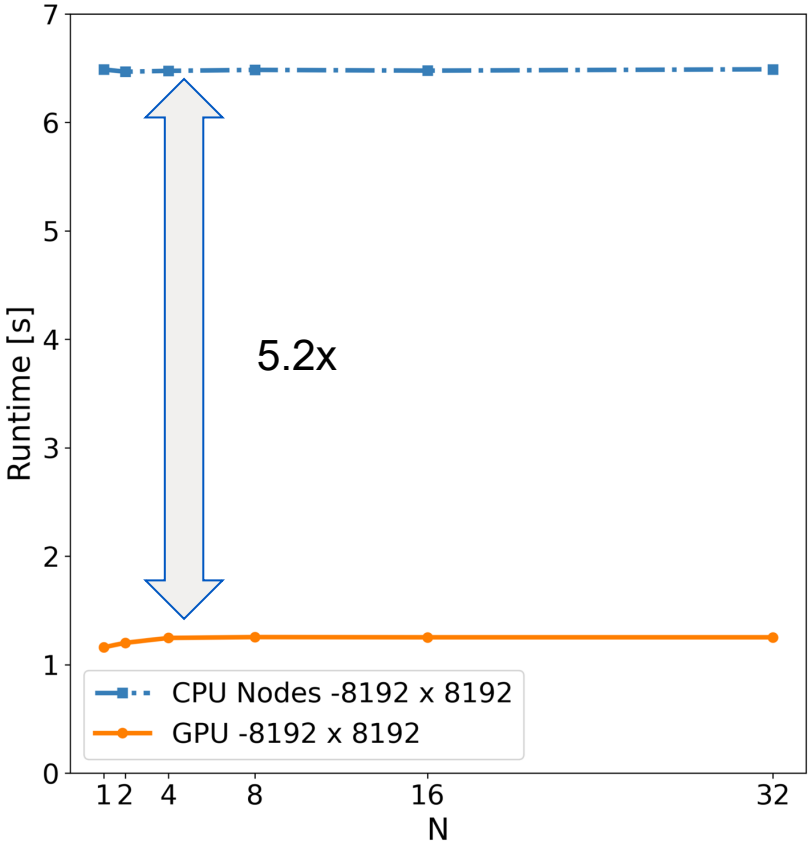
Includes supports for distributed memory

AMReX has good GPU performance

Version/problem size	64x64	128x128	256x256	512x512	1024x1024	2048x2048	4096x4096
C w/OpenACC	0.44	0.45	0.46	0.51	1.18	3.34	12.10
Fortran w/OpenACC	0.28	0.28	0.29	0.38	1.01	3.18	12.34
AMReX Ckernels	0.49	0.48	0.51	0.60	1.22	3.37	11.93
AMReX Fkernels	0.49	0.49	0.52	0.60	1.23	3.36	11.92
AMReX Fsubroutine w/OpenMP	0.83	0.82	0.89	0.93	1.55	3.77	12.65
AMReX Fsubroutine w/OpenACC	0.63	0.69	0.65	0.78	1.40	3.76	14.89
GT4Py (gt:cpu_ifirst)	2.96	2.91	3.00	3.21	3.80	6.29	16.74

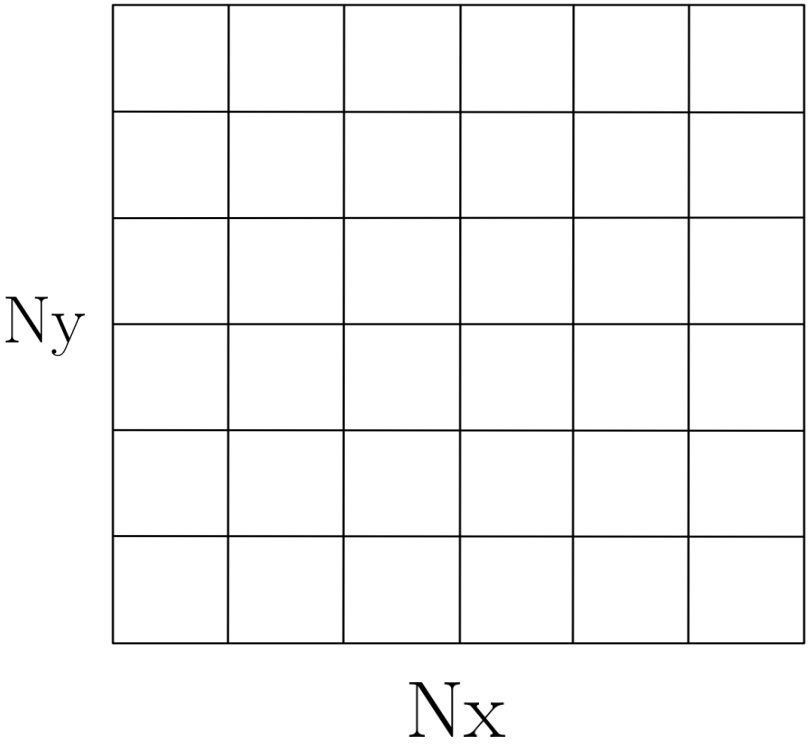
Includes supports for distributed memory


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Runtime [s]
3.7x
CPU Nodes -4096 x 4096
GPU -4096 x 4096
1 2
4
8
16
32
N

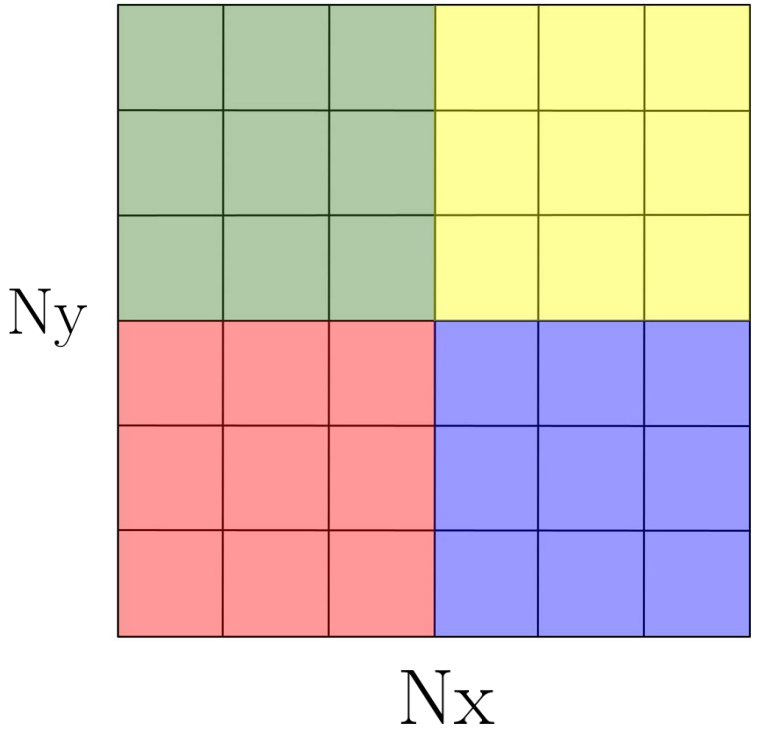

7
6
5
4
3
2
1
0
Runtime [s]
5.2x
CPU Nodes -8192 x 8192
GPU -8192 x 8192
1 2
4
8
16
32
N

AMReX - MultiFab - Parallel Data Structures

Box:

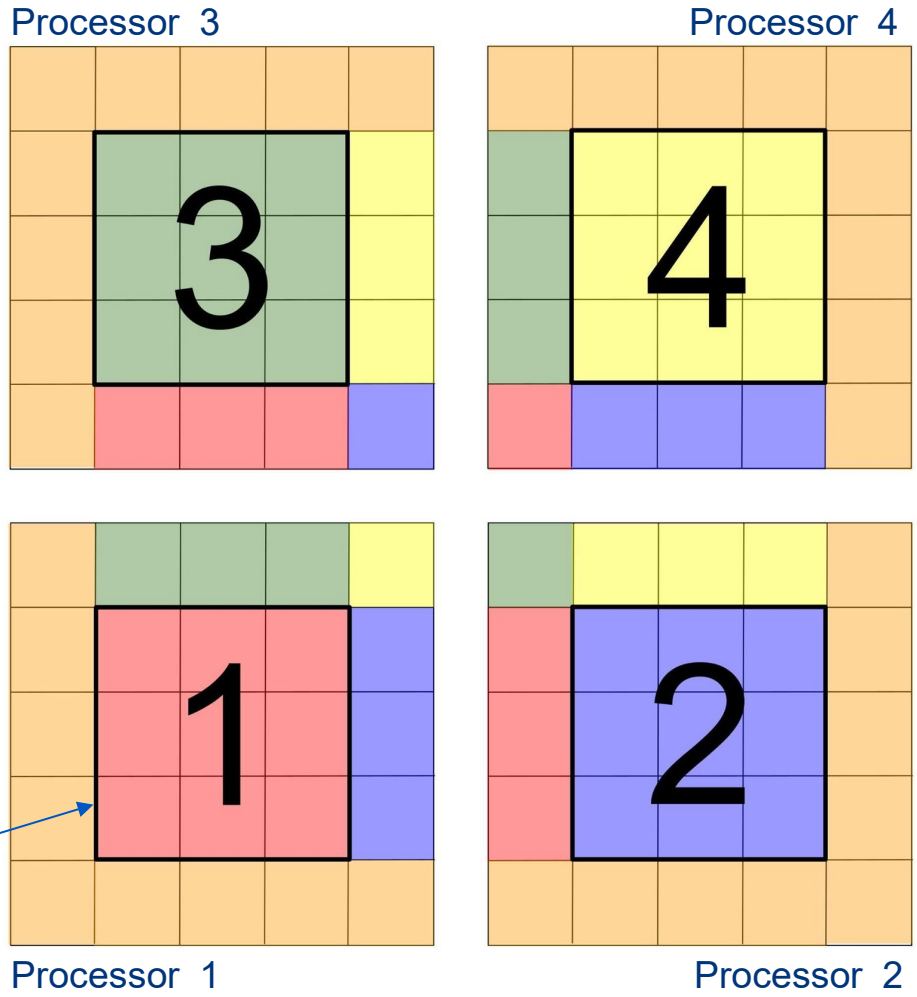


Box Array: Max grid size = 3

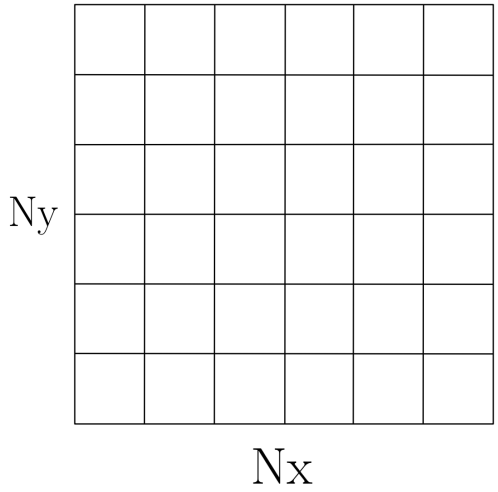


MultiFab:

N_grow = 1



Domain:



Governing Equations:

$$\frac{\partial \vec{V}}{\partial t} + \eta \hat{N} \times P\vec{V} + \nabla \left(P + \frac{1}{2}\vec{V} \cdot \vec{V} \right) = 0$$

$$\frac{\partial P}{\partial t} + \nabla \cdot \left(P\vec{V} \right) = 0$$

$$\eta = \frac{\nabla \times \vec{V}}{P}$$

Initial Conditions:

$$\vec{V}_0 \qquad P_0$$

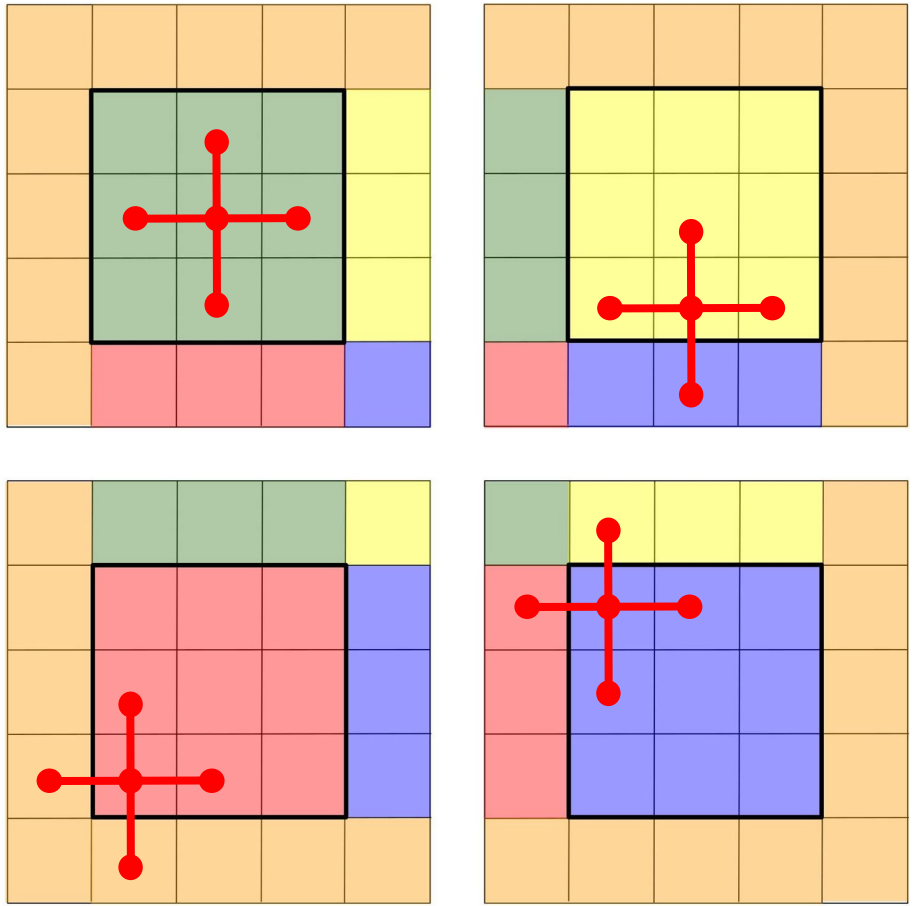
Boundary Conditions:

Periodic in both directions

AMReX Parallel Data Structures

Halo exchange and update boundary conditions:

```
u.FillBoundary(geom.periodicity())
```



- Loop over “valid” cell / face/ node indices
  - Apply same stencil kernel as interior