ESMO
Earth System Modelling and Observations

WCRP
World Climate Research Programme

HPC Update: Climate/Weather Modeling in the Exascale Era

Sarat Sreepathi and numerous collaborators from WGNE, E3SM

Nov 5, 2025

WGNE40 Meeting, Beijing, China

Common HPC Themes

Identified from input provided by WGNE members (backup slides)

- Deployment of new supercomputers
  - Model porting and performance tuning
- Mixed Precision
  - Significant gains for some centers, further efforts planned
- GPU development
  - Directives (OpenACC), Source to source transformation tools
  - Data structure design
- Performance Portability
    - Retaining good CPU performance
- I/O throughput
- AI – competing for focus and resources

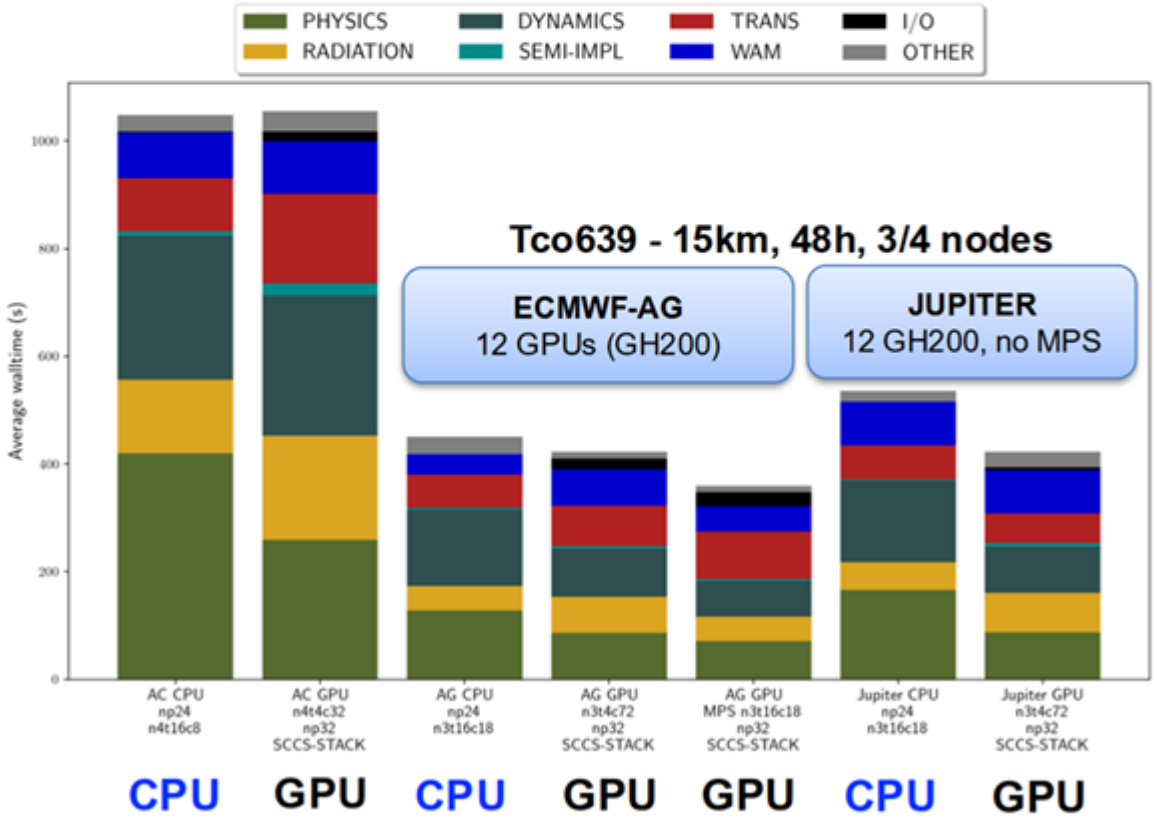
GPU-adapted IFS benchmark - putting it all together

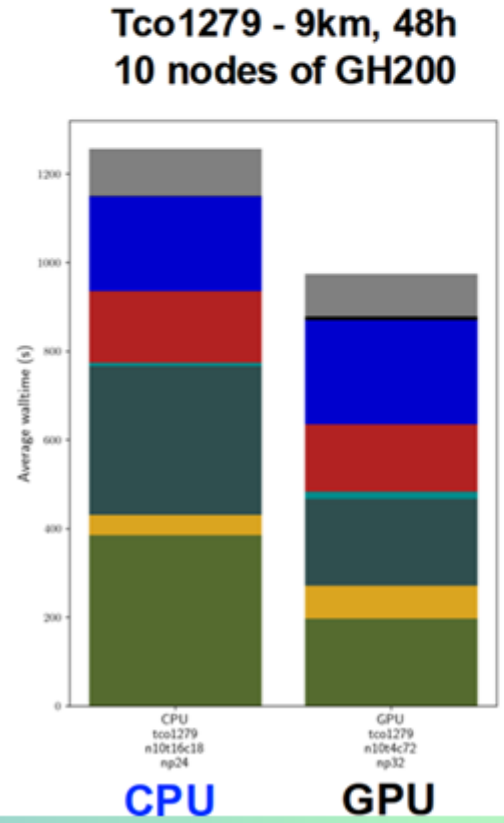
Latest RAPS release – internal cycle Cy49r3

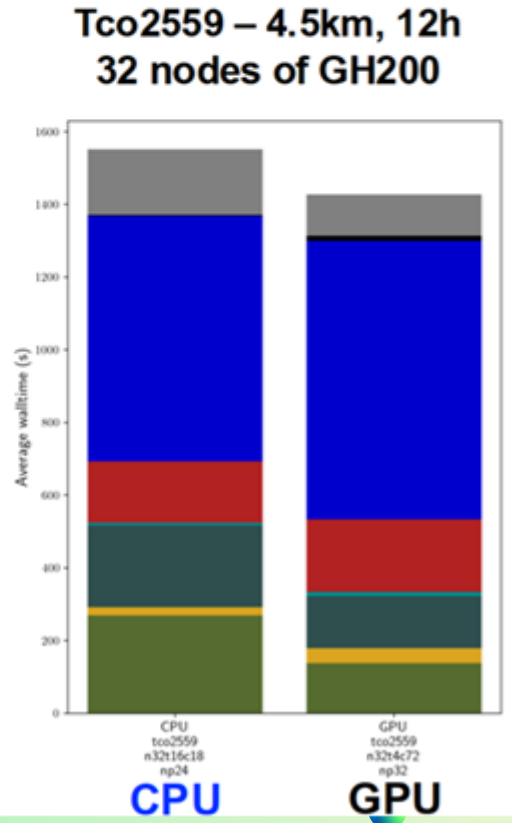
- High device residence (>85%), optimization ongoing

GPU-enabled IFS
- Competitive on Grace-Hopper
- Under-optimised data transfers
- Derived from optimised CPU
- **Latest science cycle (Cy50r1)**

Checkout the Gordon Bell Prize for Climate Modeling Finalist - paper released later this month



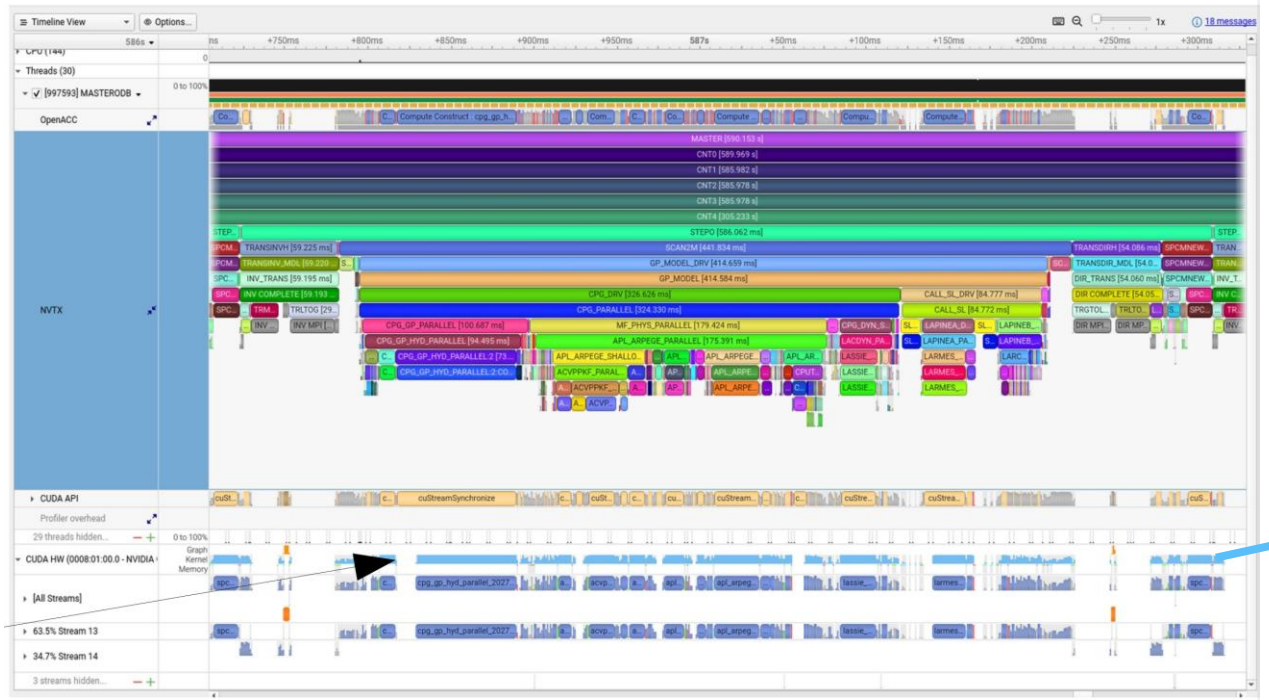




ARPEGE forecast prototype on accelerators

Achievements

- Full time step on GPU device
  - Grid point computations refactored and transformed using source-to-source software
  - Use ECTRANS and ECRAD versions ported by ECMWF
- Communications using MPI CUDA aware
  - GPU ↔ GPU: ECTRANS, semi-Lagrangian, semi-implicit
  - GPU → CPU: IO
- No transfer of field data owing to Field API
- Computations on GPU device: 90% of elapsed time



ARPEGE timestep flow on CPU/GPU

Computations on GPUs

Credits: NVIDIA

Input from F. Bouyssel

JMA: Impacts of GPU optimization for GSM

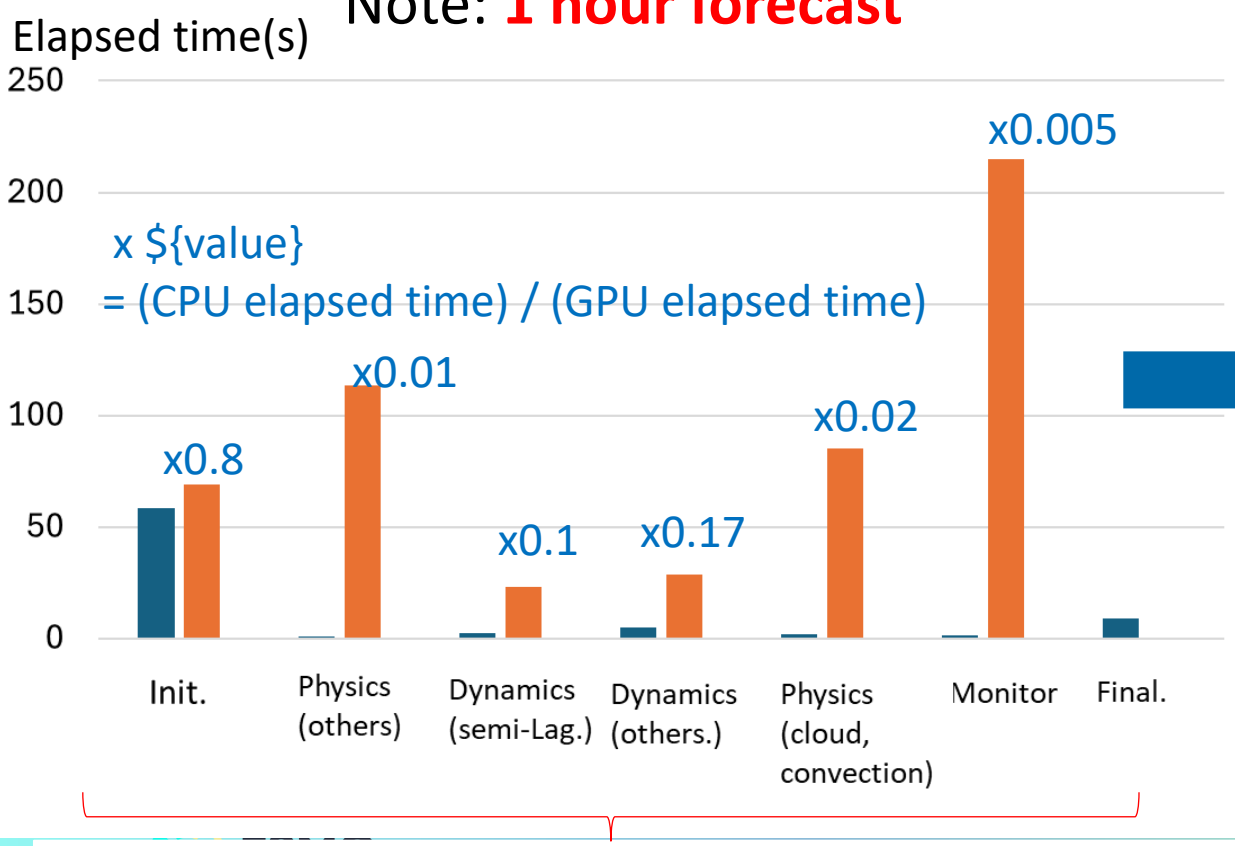
Horizontal resolution:13km(Tq1279L128)

CPU: IntelR XeonR CPU Max 9480 (3.405TFLOPS) x 2/node x 49nodes (incl. 1 I/O node)

GPU: NVIDIA H100 (66.9TFLOP(FP32)) x 74 (incl 2 GPUs for I/O ranks)

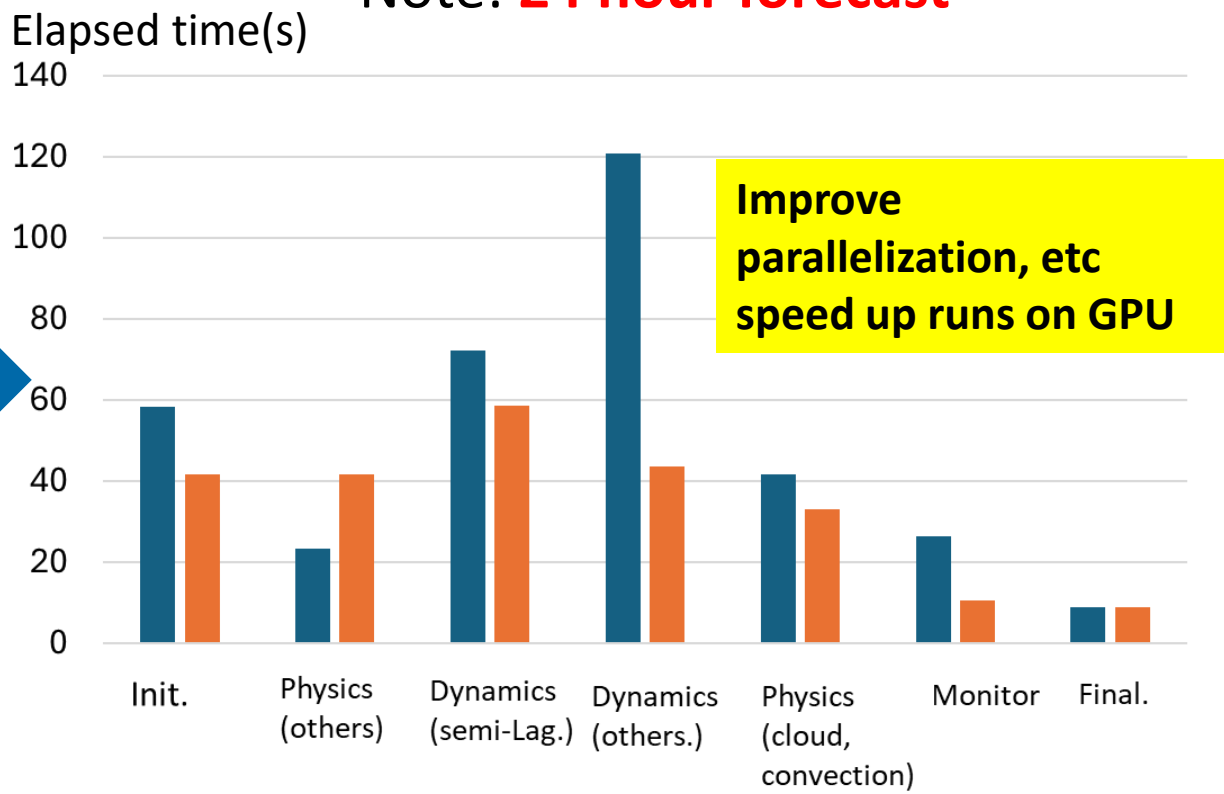
Before GPU optimization

Note: **1 hour forecast**



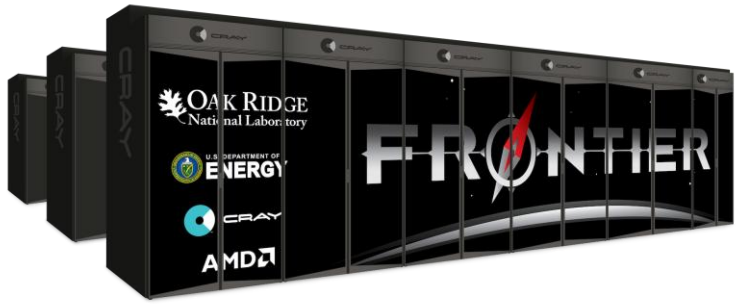
After GPU optimization

Note: **24 hour forecast**



Bottlenecks(un-parallelized/inefficiently parallelized loops) in every process

State of Exascale



- 3 U.S. Department of Energy supercomputers
  - #1 El Capitan – AMD MI300A APUs – classified use
  - #2 Frontier – AMD MI250X GPUs
  - #3 Aurora – Intel PVC GPUs
- China: 2+ exascale machines?
- Europe
  - #4 Jupiter – NVIDIA GH200
    - Confirmation of exascale – likely in new Top500list on Nov 19
- Context:
  - 1M node hours – typical allocation for E3SM on each machine
  - Facilitates decadal runs at 3km resolution

Programming Models

- C++ with templates (Kokkos)
  - Robust support across multiple GPU and CPU architectures
  - Requires minimal vendor support
- Fortran with OpenACC or OpenMP offload
  - Relies heavily on (lagging) vendor compiler support
  - Challenges with portability and performance
  - Good performance requires major code refactoring
- Domain Specific Languages
  - Promising approach (e.g. GT4Py/GridTools, PSyclone)
  - Needs sustained investment to maintain and add support for new architectures

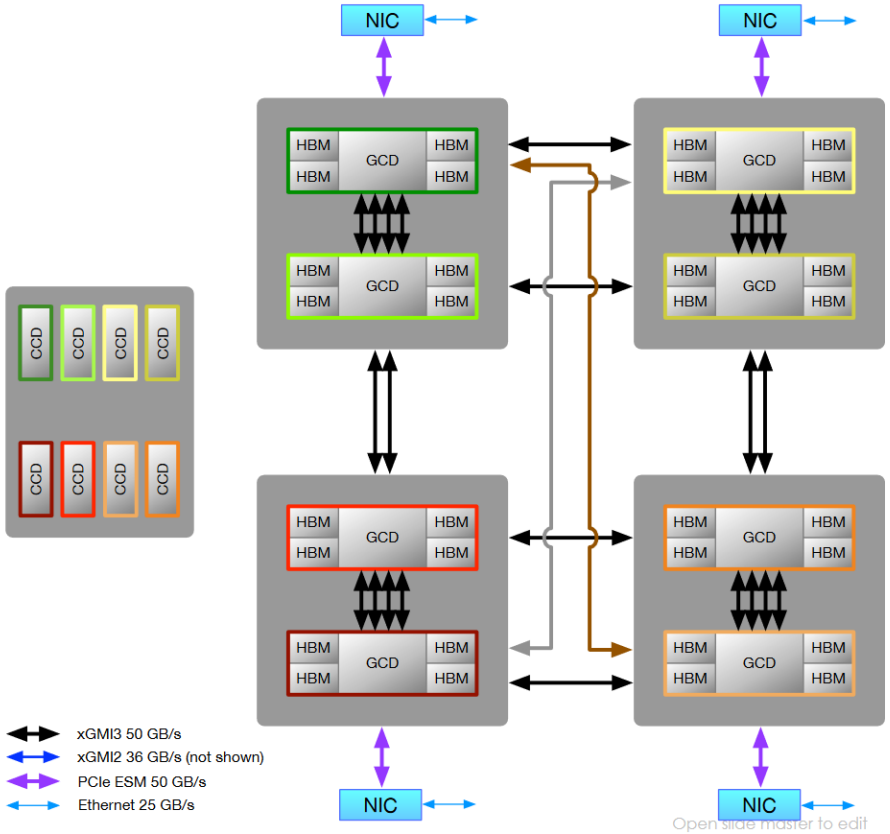
Today's Exascale: Frontier & Aurora

Frontier – First Exascale supercomputer

- 1.2 Exaflops (FP64 – HPL Benchmark)
- 29 MW, 4,000 ft^2, 9,408 nodes
- Node
  - 4 AMD MI250X GPUs/node ~ **8 logical GPU*s/node**
  - 1 AMD Trento CPU (64 cores)
  - 512 GiB DDR4 (CPU) + **512 GiB HBM2e** (GPU)
  - **GPU Mem B/W**: 8x 1,635 GB/s (**13,080 GB/s Total**)
  - **Concentration of mem b/w (98.5%) and compute flops (98.7%) on GPU.**
- GPUs directly connected to high-speed interconnect

Aurora

- **10,624 compute blades**
- **63,744 Ponte Vecchio GPUs**
- **21,248 Sapphire Rapids CPU (with HBM)**
- **Slingshot 11**



Frontier Compute Node Architecture
1 CPU, 8 GPU*s

One cabinet of Frontier (24 ft^2) has higher HPL than all of Titan (4,500 ft^2) while using lower power (309 kW vs. 7 MW)

#1 (2022-2024), succeeded by El Capitan (2024)

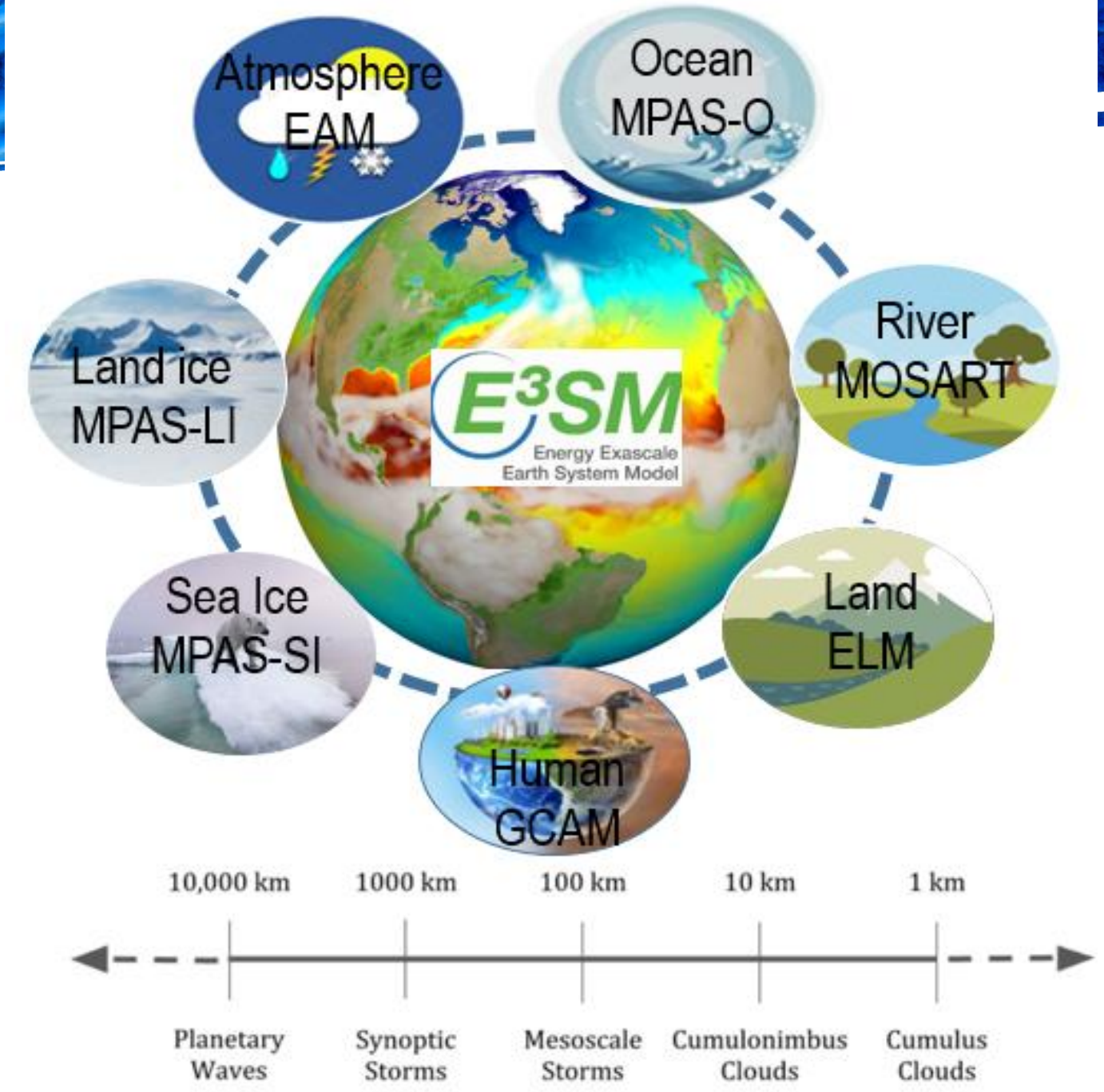
The E3SM Mission: Use exascale computing to carry out high-resolution Earth system modeling of natural, managed and man-made systems, to answer pressing problems for the DOE.*

"A DOE Model for DOE's Mission on DOE Computers"

***The E3SM project's long-term goal is to assert and maintain international scientific leadership in the development of Earth system models that address the grand challenge of actionable modeling and projections of Earth system variability and change, with an emphasis on addressing the most critical challenges facing the nation and DOE.**

E3SM Energy Exascale Earth System Model

- Global Earth System Model
- Atmosphere, Land, Ocean, Ice, … component models
- 8 DOE labs, 12 university partners,… ~$30+ M/year
- Started in Oct 2014
- Development driven by DOE mission interests: Energy/water issues looking out 40 years
- **Key computational goal: Ensure E3SM effectively utilizes DOE exascale supercomputers**
- E3SM is open source / open development
  - Website: www.e3sm.org
  - Github: https://github.com/E3SM-Project



E3SM Energy Exascale Earth System Model

Modeling across scales in three versions over a decade

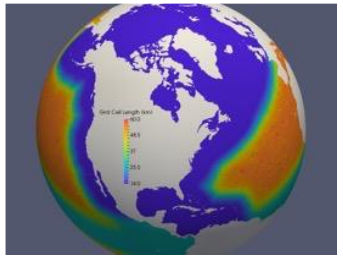
Beyond v3: unification

Model component	Lower resolution (LR)	High resolution (HR)	Cloud-resolving (SCREAM)	Regional refined model (RRM)
Atmosphere & Land	100 km	25 km	3 km	variable
Ocean & Ice	30-60 km	6-18 km	prescribed	variable
River	50 km	12 km	3 – 12 km	variable
	CMIP6 DECK, C4MIP LENS, DAMIP	**HighResMIP**	**DYAMOND**	**CMIP6 DECK (NARRM)**

North America RRM

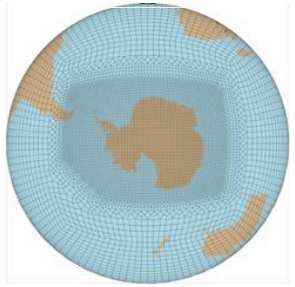
25 km → 100 km

14 km → 60 km

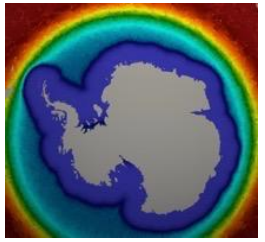


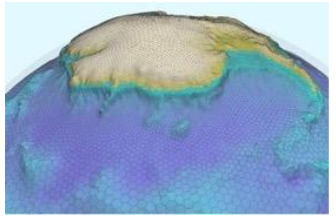
Southern Ocean RRM

25 km → 100 km

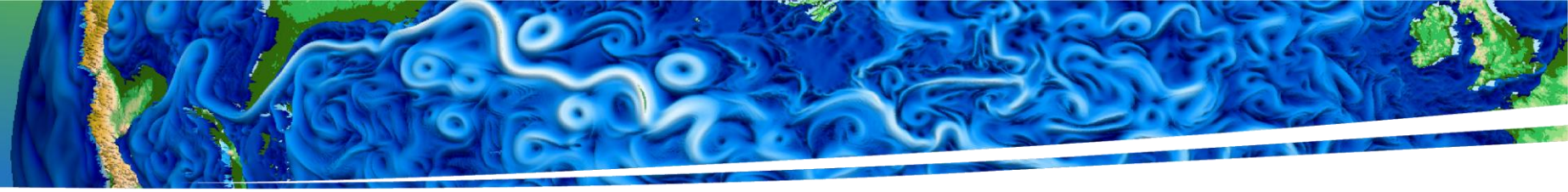


12 km in the Antarctic, 30-60 km elsewhere





~1.3 years of global coupled configuration 3km atmosphere, 18to6km ocean on Aurora



Scream Summary

- Resolution: 3.25km horizontal, 128 vertical levels
- dynamics: HOMME non-hydrostatic dycore (Taylor, JAMES, 2020)
- microphysics: Predicted Particle Properties (Morrison, Milbrandt, J.Atm.Sci. 2015)
- macrophysics: Simple High Order Closure (Bogenschutz, Kruger, JAMES 2015)
- radiation: RTE+RRTMGP package (Pincus et al, JAMES 2019)
- aerosol: prescribed

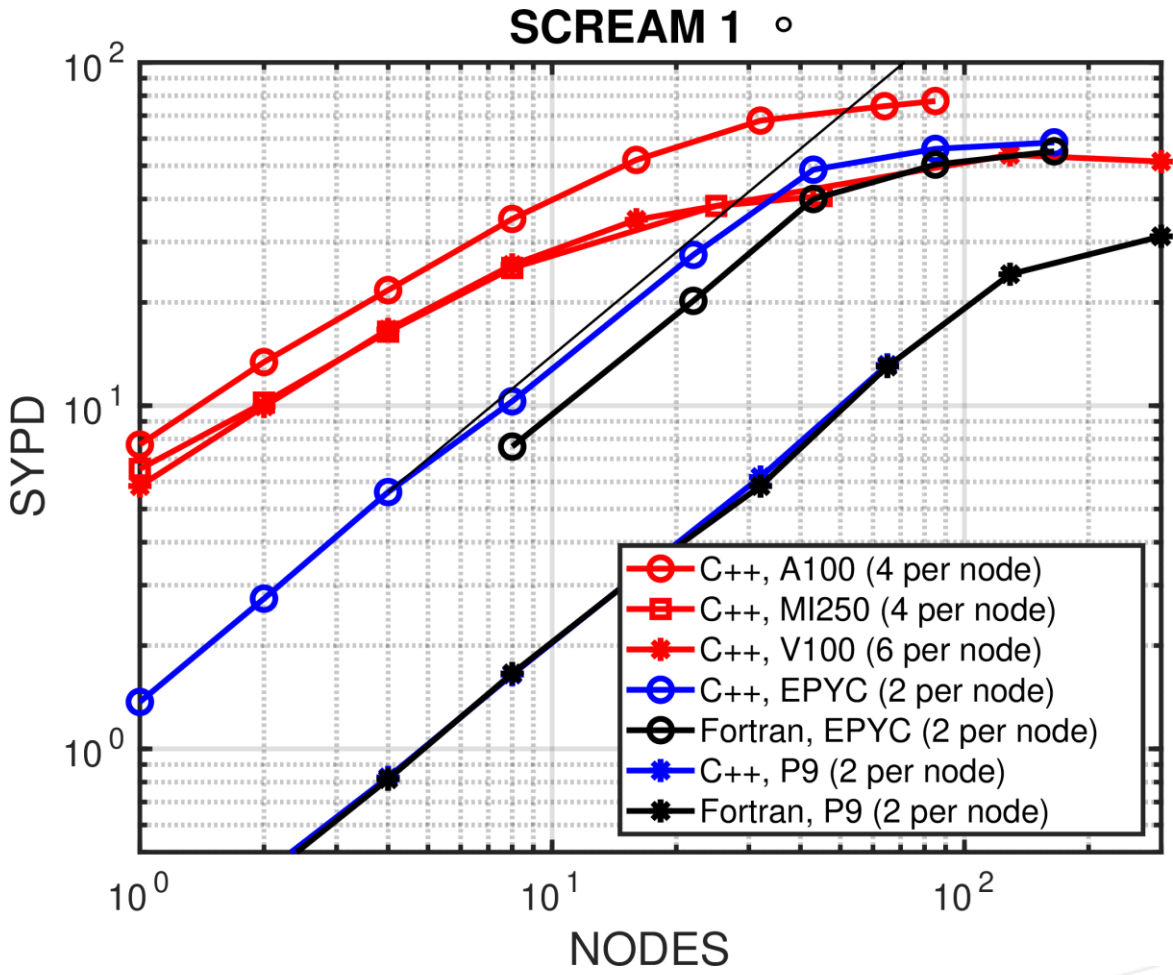
Horizontal resolution	Vertical resolution	No. of $p = 3$ spectral elements	timestep dynamics	timestep physics	dof dynamics	dof physics
110 km	128 Layers	5400	300s	1800s	6.2M	2.8M
3.25 km	128 Layers	6.3M	8.33s	100s	7.2B	3.2B

C++/Kokkos: Performance Portability

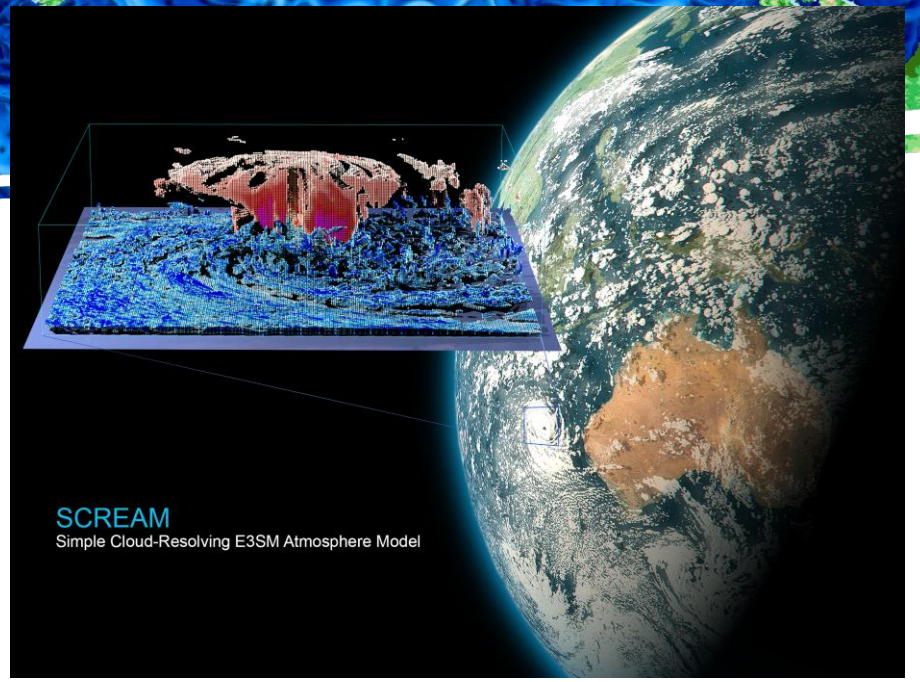
E3SM's Atmosphere model (EAMXX in "SCREAM" configuration)

1 degree resolution (~110km): 128 vertical levels, nonhydrostatic (NH) dycore, 10 tracers, P3/SHOC physics with prescribed aerosols,
no convective parameterization

- Performance portability
  - IBM P9, AMD EYPC
  - NVIDIA V100, A100
  - AMD MI250
- CPU performance:
  - C++/Kokkos as fast or faster than Fortran
- GPU performance:
  - Large scaling range where GPU nodes are 4-10x faster than CPU nodes

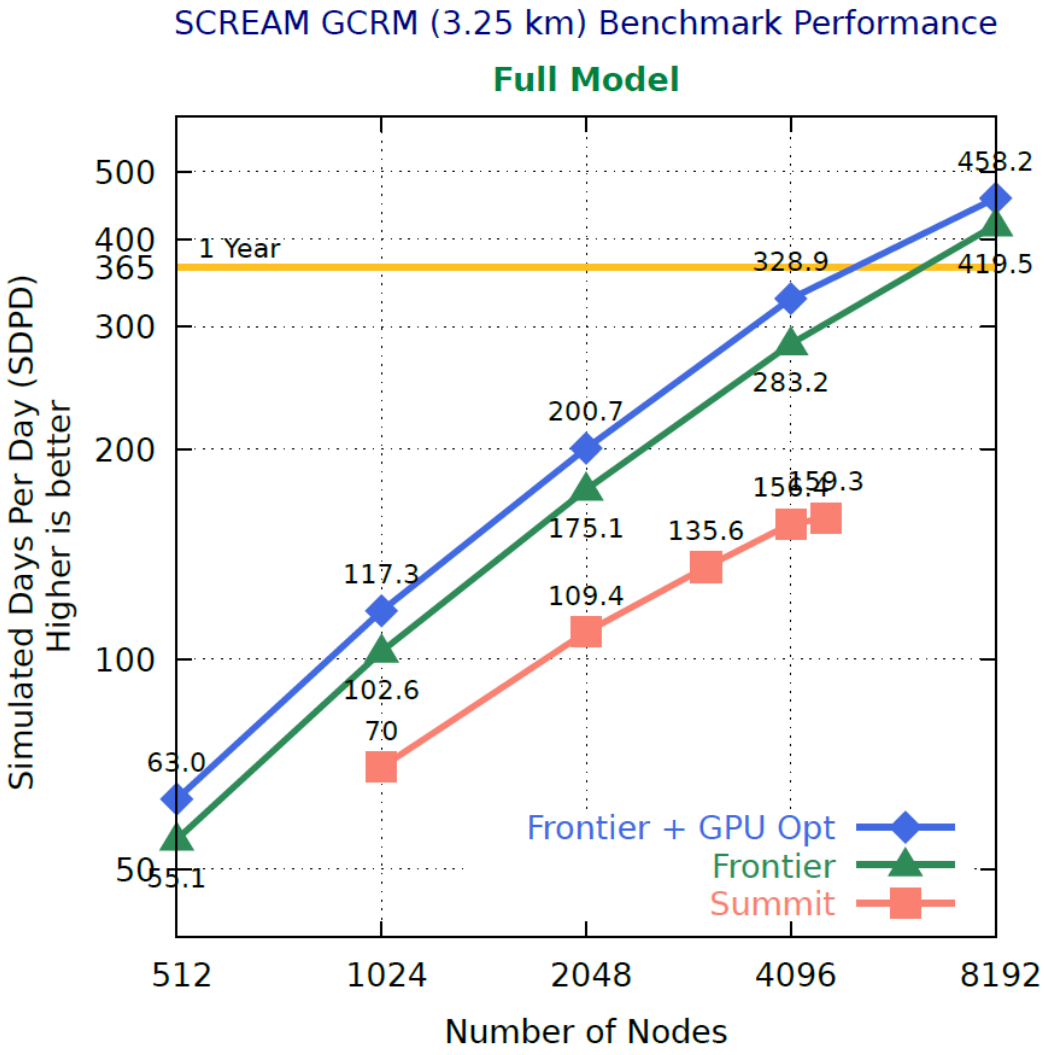


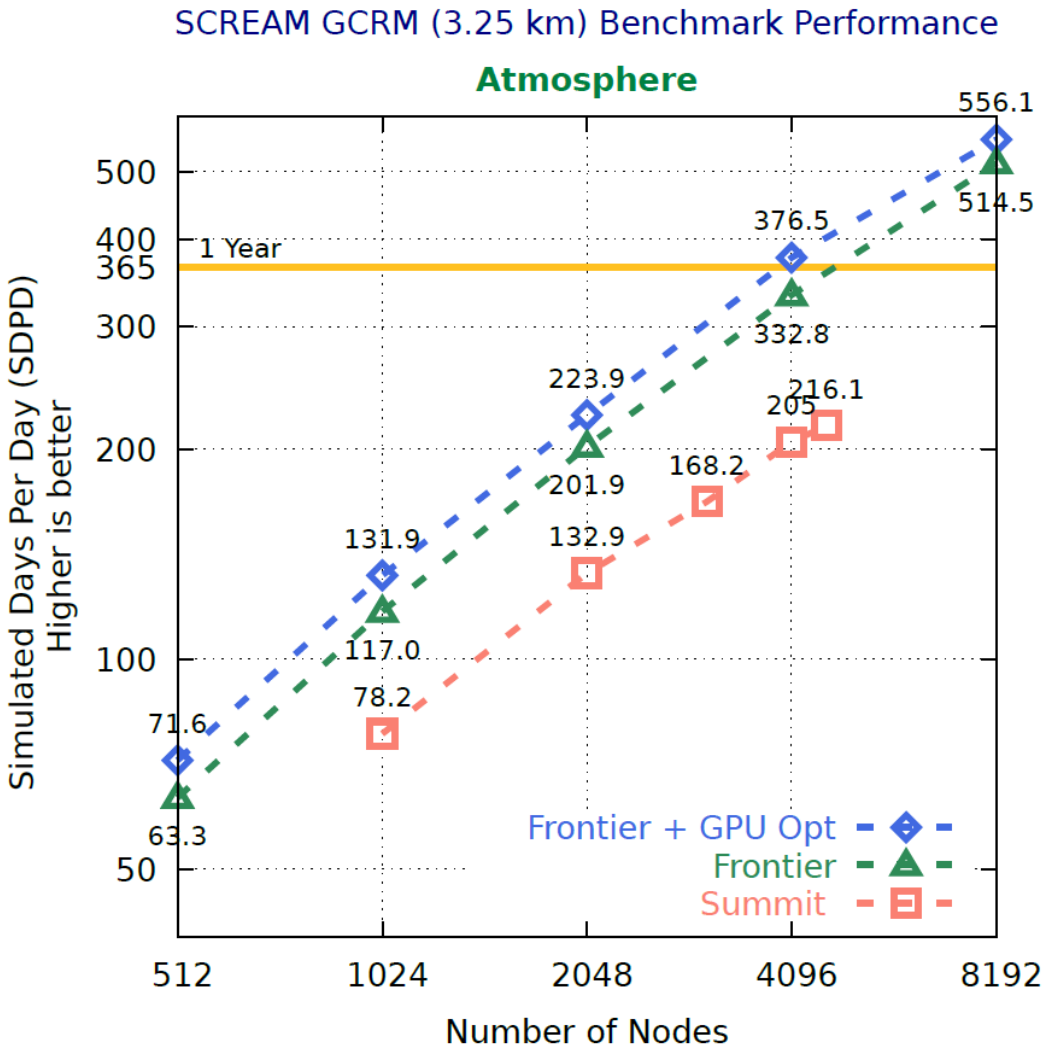
SCREAM



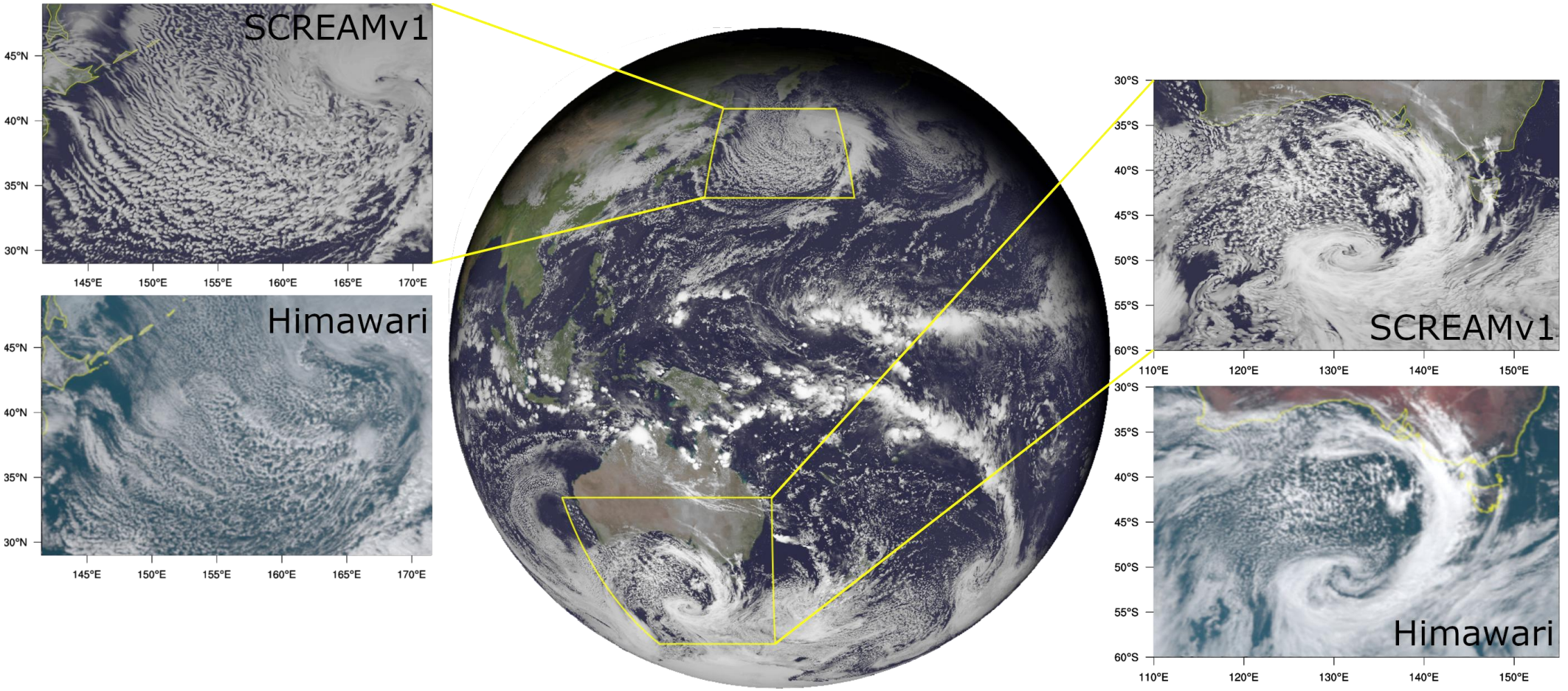
- SCREAM
  - Demonstrated true performance portability:
  - Competitive performance on CPUs compared to Fortran code
  - Excellent results on NVIDIA GPUs (V100s, A100s) and AMD GPUs (MI250s). Performance optimization ongoing on Intel PVC GPUs.
- First-to-Exascale opening up new science:
  - Broke the long standing “1 SYPD” goal for a global cloud resolving model
  - Multi-decadal length simulations at cloud resolving resolutions completed in 2024!
  - 2023 Gordon Bell Prize in Climate Modelling for innovative parallel computing contributions toward solving the global climate crises.

Taylor, Caldwell, Bertagna, Clevenger, Donahue, Foucar, Guba, Hillman, Keen, Krishna, Norman, Sreepathi, Terai, White, Wu, Salinger, McCoy, Leung, Bader, *The Simple Cloud-Resolving E3SM Atmosphere Model Running on the Frontier Exascale System* SC23: International Conference for High Performance Computing, Networking, Storage and Analysis (2023)



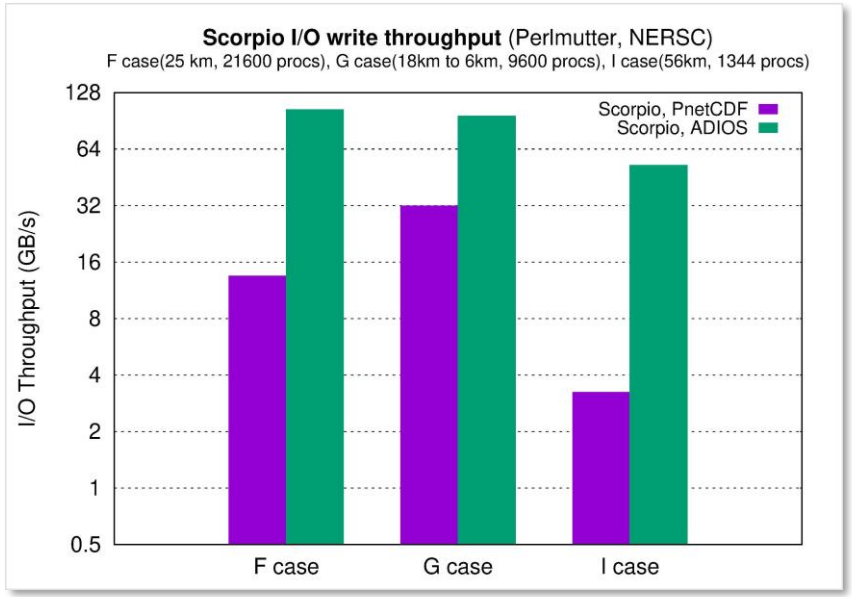


54% improvement (from initial paper) at 8k nodes



Snapshots of outgoing shortwave radiative flux at the model top from a January SCREAM simulation, taken two days into the simulation (2020-01-22 at 02:00:00 UTC). The insets show comparisons against Himawari-8 visible satellite imagery for two scenes: a cold air outbreak event near Siberia (left), and a cyclone south of Australia (right).

I/O performance : *E3SM High-Resolution Cases*



- ADIOS gives an order of magnitude higher performance by offloading some of the data rearrangement to the NetCDF conversion tool (No conv for restarts)

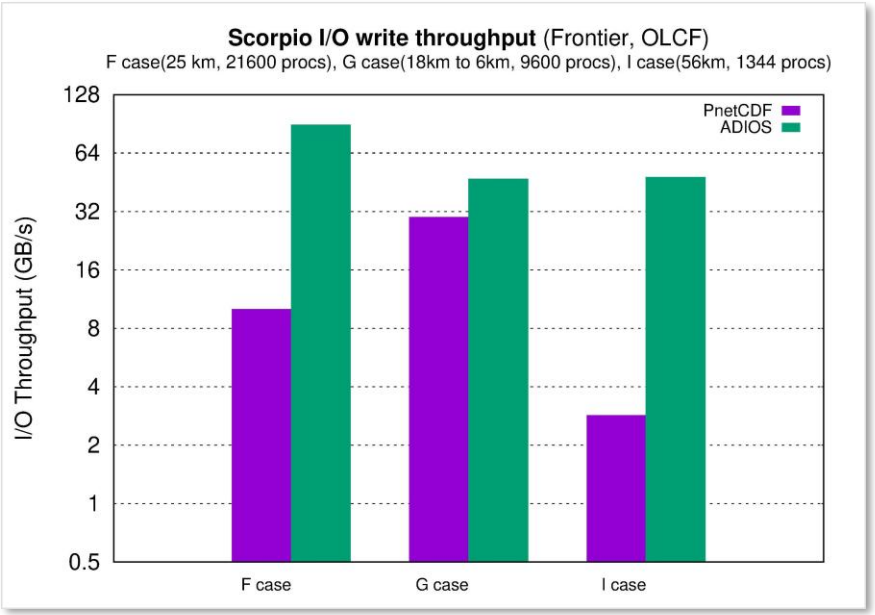
- No restarts, F/G cases 1 day, I case 10 days
- Fcase: 296 GB, Gcase: 80 GB, Icase : 360 GB
- Fcase : 338 nodes (21600 procs, no threading, 64 procs/node)
- G case : 150 nodes, I case : 21 nodes
- Perlmutter@NERSC: 3K nodes, 128 cores/node, 512 GB/node, HPE Slingshot, Lustre

F case : Active Atmosphere and Land

G case : Active Ocean and Sea Ice

I case : Active Land and River

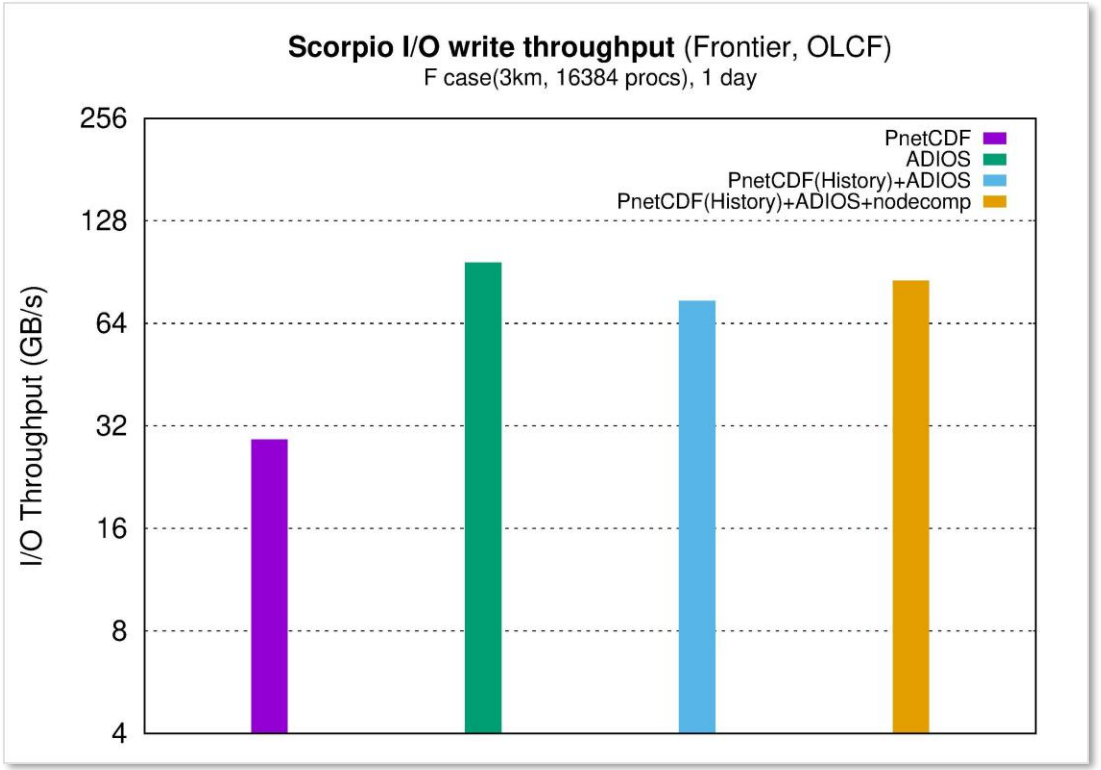
>100 GB/s write throughput for atmosphere model runs



- Comparable perf across Perlmutter and Frontier with PnetCDF
- ADIOS write for I Case is 6X of PnetCDF

I/O Performance : *E3SM Ultra High-resolution Cases*

- E3SM/SCREAM decadal production run configuration
- 1 day, ne1024pg2, 3km
- 2048 nodes (8 MPI x 7 threads/node), Frontier
- ATM Restart file : 2.3 TB (dominates the I/O cost), ELM Restart file : 400 GB, CPL restart : 48 GB
- ATM history : ~20 GB, ELM history : ~100 GB
- (ADIOS for all output) vs (ADIOS for restarts & PnetCDF for history output)
- Overall performance not impacted significantly by using PnetCDF for history writes
- Lustre



F case : Active Atmosphere and Land

Summary

- Exascale is here!
- Increasing heterogeneity in compute architectures
- Performance portability is critical
- Currently, GPUs are the only pathway for leadership-class supercomputing due to power constraints
- Top end of supercomputing offers unique capabilities
  - Need creative model configurations for useful science
  - May not fit traditional climate modeling scenarios (throughput of 100s of years)
- Strategic planning for scientific models need to prepare for this future

ESMO
Earth System Modelling and Observations

WCRP
World Climate Research Programme

Thank you!

Contact:

Sarat Sreepathi

sarat@sarats.com

HPC/efficiency efforts at Météo-France

Towards a general use of single-precision (32 bits) in operational NWP systems.

1. Operational use in all AROME[1] operational systems (forecast component only)
2. Operational use in all ARPEGE forecasts (upcoming e-suite)
3. Next steps: soon operational use
   i. in all trajectories within the assimilation cycle
   ii. later, in parts of assimilation whenever possible

Adaptation to hybrid processors with accelerators:

- Full time step of ARPEGE model ported on GPU (except the surface model SURFEX). Used as a benchmark for the call of tender of our future HPC system
- Plans:
  - Further optimize ARPEGE, finalize refactoring
  - Progress on AROME regional model
  - progress on source-to-source scripts and integration in our compiling environment,
  - Port our surface model (SURFEX)
- Main objective: have an ARPEGE configuration running on GPU for testing in operations
- Work done in collaboration with ECMWF and ACCORD[2] partners (DestinE On Demand project phase 2), and within TRACCS[3] French national programme (WP on new computing paradigms).

[1] Météo-France LAM NWP operational system
[2] A COnsortium for convective-scale modelling Research and Development

Input from F. Bouyssel

ARPEGE forecast prototype on accelerators

Objectives

- Increase computing power, while keeping power consumption as of today
- Increase competition between HPC and processor manufacturers
  - ➢ Important for Météo-France call for tender
  - ➢ Porting of ARPEGE started in 2021

Approach

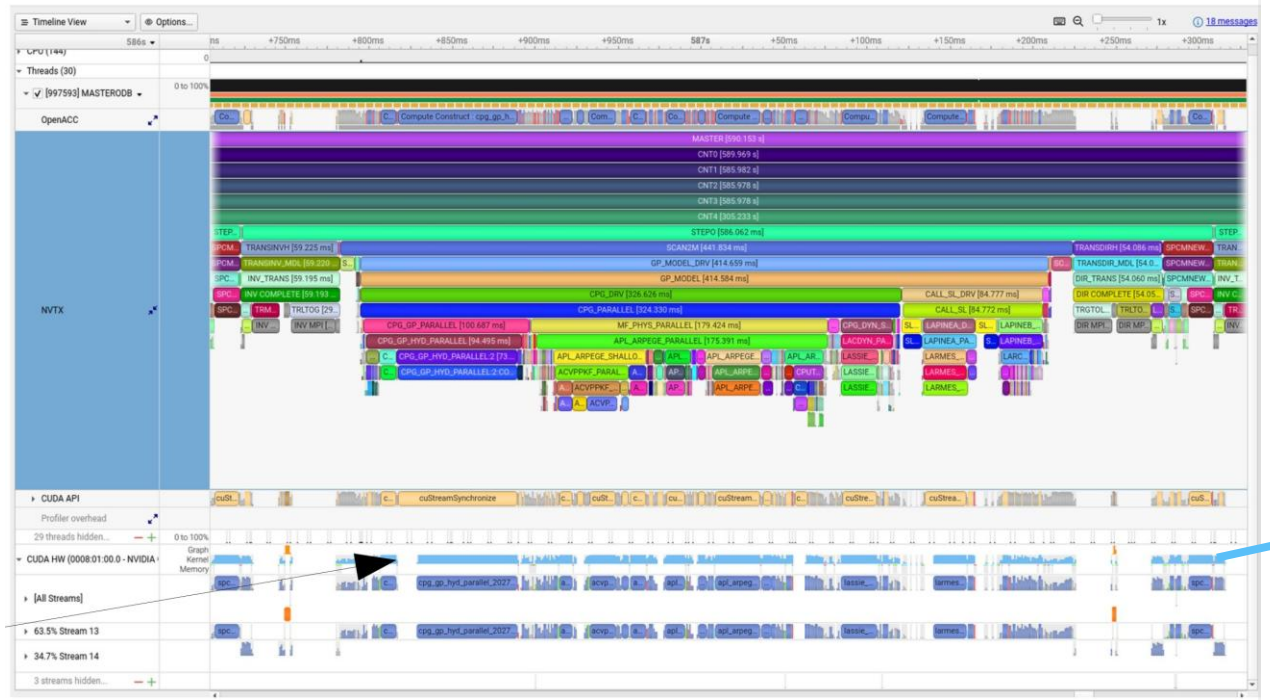
- Refactor the code
- Manage field data with Field API
- Apply coding norms
- Use source-to-source translation tools (fxtran/loki)
- Keep the code vendor-agnostic
- Work in a collaborative framework with our partners (ECMWF, ACCORD)

Input from F. Bouyssel

ARPEGE forecast prototype on accelerators

Achievements

- Full time step on GPU device
  - Grid point computations refactored and transformed using source-to-source software
  - Use ECTRANS and ECRAD versions ported by ECMWF
- Communications using MPI CUDA aware
  - GPU ↔ GPU: ECTRANS, semi-Lagrangian, semi-implicit
  - GPU → CPU: IO
- No transfer of field data owing to Field API
- Computations on GPU device: 90% of elapsed time



ARPEGE timestep flow on CPU/GPU

Credits: NVIDIA

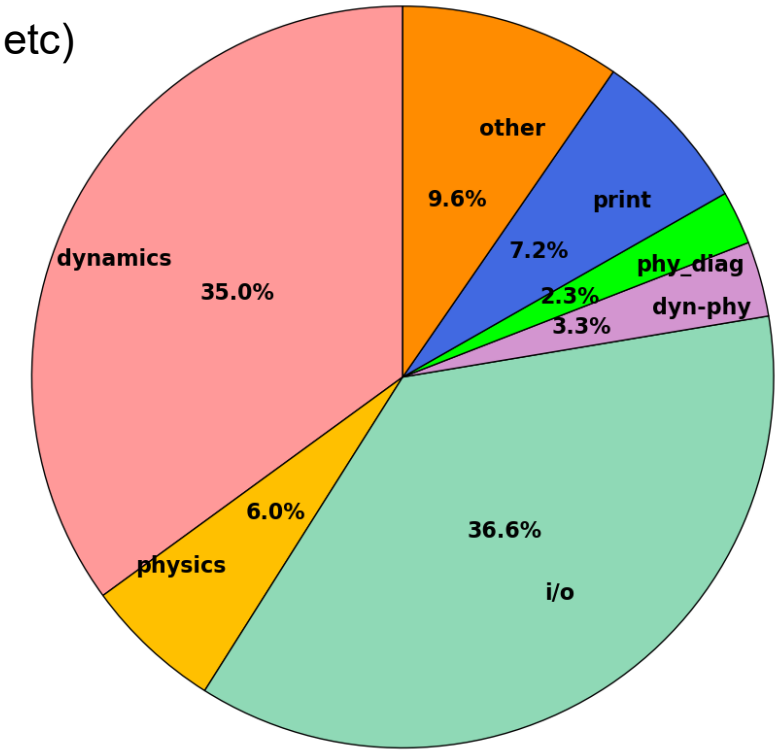
Input from F. Bouyssel

HPC/Exascale efforts at KIAPS

KIAPS

The Operational 8-km KIM at KMA: Updated Version Active Since May 14, 2025

- **Optimization includes:**
  - Optimization for the KMA's 5th supercomputer (compiler, I/O, environment, etc)
  - "Stripe" partitioning for the cubed-sphere grid
  - High performance parallel I/O method (grouping, rearrangement)
- **MPI rank average elapsed time ratio**
  - 1,500 nodes (Intel Ice-Lake 8368Q 2.6GHz 38Cx2)
  - 15 day time integration, Total elapse time: 160 minute
  - File write elapse time ratio: 36.6% (14.7 TB output file)



MPI rank average elapse time ratio for operational KIM

Ongoing work to improve computational efficiency

- Development of an adaptive time-stepping method
- Exploring the use of large time step

In the upcoming five year plan (2027- 2031), KIAPS aims to further improve the model's performance by utilizing advanced numerical methods, CPU optimization, I/O node development, and GPU porting.

ECMWF Input

- Nils talk on Day 1.
- Prior talk: https://events.ecmwf.int/event/460/contributions/5298/attachments/3210/5354/HPC-WS_Wedi.pdf

HPC readiness: Input from JMA

Japan Meteorological Agency

Outline

- Three main JMA models are being prepared for adopting to future HPCs (e.g. reduced precision, GPU porting etc)
  - GSM (JMA's operational global model)
  - ASUCA (JMA's operational regional model)
  - MRI.COM (JMA/MRI's operational/research ocean model)
- This ppt reports recent experience of GPU porting and optimization at JMA

Status of GPU porting

- Porting using OpenACC
- Status of each model
  - GSM , ASUCA: Almost all of time integration parts are ported to GPU
    - Full GPU approach rather than partial GPU approach for bottlenecks, due to data transfer costs and different data structure suitable for CPUs/GPUs
  - ASUCA-Var: Tangent linear / Adjoint models are ported to GPU
  - MRI.COM: Under porting
- GSM, ASUCA and ASUCA-Var are in the stage of optimization

Steps for GPU porting and optimization

- **(0) Preparation :** e.g. flexible data structure (size of inner/outer most loops adjustable) suitable both for CPU and GPU
- **(1)Porting （several years including familiarization with GPU）**
  - Safe port to GPU first : ensuring bit-identical or difference between rounding error are desirable **-> important process for quality assurance**
    - Compile options for consistent mathmatical functions between CPU and GPU as possible can be a useful method .
  - Data on GPU memory as possible, make data transfer between CPU-GPU minimum
- **(2)Construct common develop environment so that new developers / scientists get started to GPU optimization quickly （several months ~ a year）**
  - Common branch for running target models (with timers for each model process) on GPU **-> important preparation / infrastructure for optimization**
- **(3)Optimization (several months <u>if (0)-(2) are satisfied</u>)**
  - **Parallelize “loops”, and parallelize “development”**

Tactics of GPU optimization for GSM

- **Parallelize “loops”**
  - Find parallelization-suppressed loops by compile logs, and bottlenecks by timers
  - **Make loops parallelizable even if these increase computational amount, memory size, or inefficient memory access by:**
    - Removing loop-carried dependency
    - Replacing scalar variables with arrays in inner-most loops
    - Different tactics of optimization on CPU/scalar machines
  - **Specify parallelism (“gang” and/or “vector”) explicitly** using OpenACC directives
  - Reduce overheads of launching “acc kernels”
- **Parallelize “development”**
  - Those who are familiar with a model process optimize the process
    - dynamics, physics parameterization, data assimilation, EPS, etc
  - Use the common develop environment & cases
- **Modification within the range of mathematically identical (bit-comparison is not always guaranteed) for safe optimization**

An examples parallelization

Before

```
!$acc kernels &
!$acc present(rupd,rdnd,tdir,rupb,upflx,dnflx,tdnb, &
!$acc&        upcs,dncs,up,dn,aclm_r                 )
!!$acc update host(rupd,rupb,rdnd,tdir,tdnb,aclm_r)

      do k = 2, km1

        do i = 1, im

          !   Flux in each sub-column
          do iclm = 0, NCMAX
            rrupdn = 1.0_WPRR / (1.0_WPRR - rupd(iclm,i,k) *
rdnd(iclm,i,k))
            dirrup = tdir(iclm,i,k) * rupb(iclm,i,k) * rrupdn
            upflx(iclm) = tdnb(iclm,i,k) * rupd(iclm,i,k) * rrupdn
+ dirrup
            dnflx(iclm) = tdnb(iclm,i,k) * rrupdn + rdnd(iclm,i,k)
* dirrup + tdir(iclm,i,k)
          end do

          !   Clear-sky
          upcs(i,k) = upflx(0)
          dncs(i,k) = dnflx(0)

          !   All-sky
          up(i,k) = 0.0_WPRR
          dn(i,k) = 0.0_WPRR
          do iclm = 1, NCMAX
            up(i,k) = up(i,k) + upflx(iclm) * aclm_r(iclm,i)
            dn(i,k) = dn(i,k) + dnflx(iclm) * aclm_r(iclm,i)
          end do
```

Specify parallelism by directives

Use 3d arrays for making “i” and “k” loops parallelizable

After

```
!$acc kernels &
!$acc present(rupd,rdnd,tdir,rupb,upflx3d,dnflx3d,tdnb, &
!$acc&        upcs,dncs,up,dn,aclm_r                    )

!$acc loop gang
do k = 2, km1
  !$acc loop vector
  do i = 1, im

    !   Flux in each sub-column
    do iclm = 0, NCMAX
      rrupdn = 1.0_WPRR / (1.0_WPRR - rupd(iclm,i,k) *
rdnd(iclm,i,k))
      dirrup = tdir(iclm,i,k) * rupb(iclm,i,k) * rrupdn
      upflx3d(iclm,i,k) = tdnb(iclm,i,k) * rupd(iclm,i,k) *
rrupdn + dirrup
      dnflx3d(iclm,i,k) = tdnb(iclm,i,k) * rrupdn +
rdnd(iclm,i,k) * dirrup + tdir(iclm,i,k)
    end do

    !   Clear-sky
    upcs(i,k) = upflx3d(0,i,k)
    dncs(i,k) = dnflx3d(0,i,k)

    !   All-sky
    up(i,k) = 0.0_WPRR
    dn(i,k) = 0.0_WPRR
    do iclm = 1, NCMAX
      up(i,k) = up(i,k) + upflx3d(iclm,i,k) * aclm_r(iclm,i)
      dn(i,k) = dn(i,k) + dnflx3d(iclm,i,k) * aclm_r(iclm,i)
    end do
```

x 1000 speed-up in this section (on NVIDIA A100)

Note: size of loops

km1: 129
im: O(10^3)
NCMAX: 15

Compile log:

```
743, Generating present(rupb(:,:,:),dnflx(:),rupd(:,:,:),rdnd(:,:,:),upflx(:),upcs(
747, Complex loop carried dependence of upflx,dnflx prevents parallelization
     Parallelization would require privatization of array dnflx(0:15),upflx(0:15)
748, Parallelization would require privatization of array upflx(0:15),dnflx(0:15)
     Generating NVIDIA GPU code
    747, !$acc loop seq
    748, !$acc loop seq
    751, !$acc loop vector(32) ! threadidx%x
    765, !$acc loop seq
751, Loop is parallelizable
```

```
748, Generating present(rupb(:,:,:),dnflx3d(:,:,:),rupd(:,:,:),rdnd(:,:,:),upflx3d(:,:
753, Loop is parallelizable
755, Loop is parallelizable
     Generating NVIDIA GPU code
    753, !$acc loop gang, vector(4) ! blockidx%y threadidx%y
    755, !$acc loop gang, vector(32) ! blockidx%x threadidx%x
    758, !$acc loop seq
    772, !$acc loop seq
758, Loop is parallelizable
```

Impacts of GPU optimization for GSM

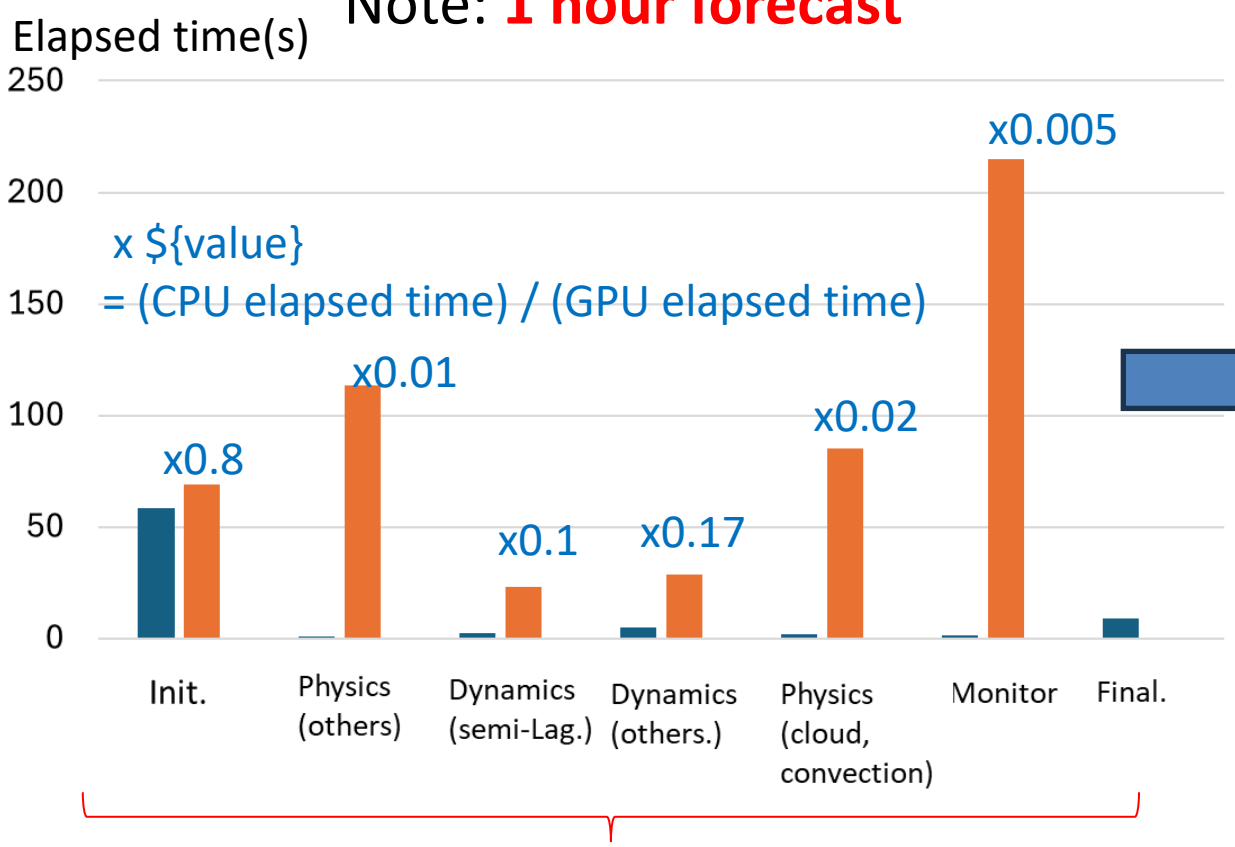
Horizontal resolution:13km(Tq1279L128)

CPU: IntelR XeonR CPU Max 9480 (3.405TFLOPS) x 2/node x 49nodes (incl. 1 I/O node)

GPU: NVIDIA H100 (66.9TFLOP(FP32)) x 74 (incl 2 GPUs for I/O ranks)

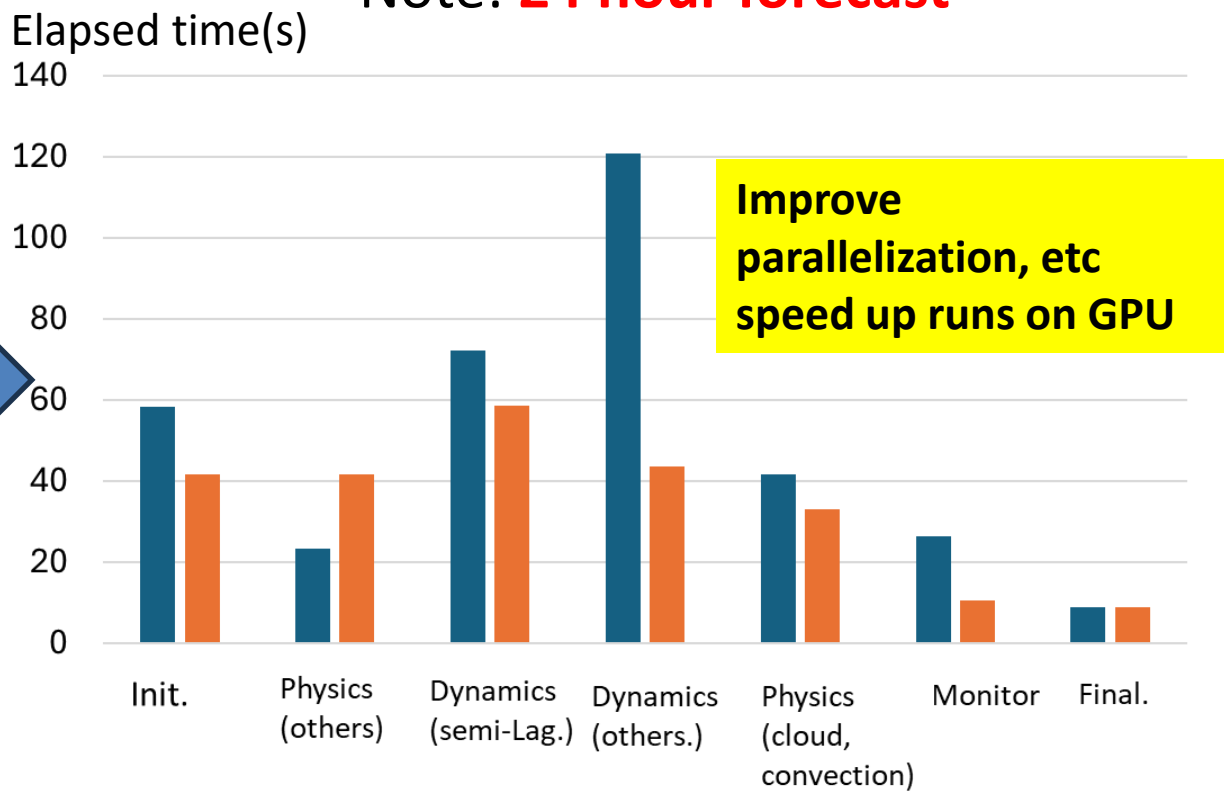
Before GPU optimization

Note: **1 hour forecast**



After GPU optimization

Note: **24 hour forecast**



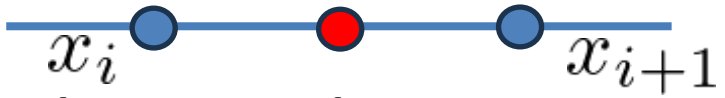
Bottlenecks(un-parallelized/inefficiently parallelized loops) in every process

GPU-parallelizable adjoint code in DA : an example of interpolation

Forward operator:

$$\begin{pmatrix}\vec{y}\\ \vec{x}\end{pmatrix} = \begin{pmatrix}\mathbf{O} & \mathbf{M}\\ \mathbf{O} & \mathbf{I}\end{pmatrix}\begin{pmatrix}\vec{y}\\ \vec{x}\end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0.5 & 0.5 \end{pmatrix}$$

$$y_i = 0.5\left(x_i + x_{i+1}\right)$$



Adjoint (transpose)operator:

$$\begin{pmatrix}\vec{y}\\ \vec{x}\end{pmatrix} = \begin{pmatrix}\mathbf{O} & \mathbf{O}\\ \mathbf{M}^T & \mathbf{I}\end{pmatrix}\begin{pmatrix}\vec{y}\\ \vec{x}\end{pmatrix}$$

Original adjoint code
(can not be parallelized) on GPU

```
!$acc parallel loop seq
do i = 1, nx
  x(i)   =  x(i)   + 0.5 * y(i)
  x(i+1) =  x(i+1) + 0.5 * y(i)
  y(i)   =  0.0
end do
```

Loop carried dependency prevents GPU parallelization

GPU parallelizable adjoint code

x O(100) faster than the original code on NVIDIA A100

```
!$acc parallel loop
do i = 0, nx+1
  x_p(i) = 0.d0
end do

!$acc parallel loop
do i = 1, nx
  x(i)   =  x(i)   + 0.5 * y(i)
  x_p(i+1) =  x_p(i+1) + 0.5 * y(i)
  y(i)   =  0.0
end do

!$acc parallel loop
do i = 0, nx+1
  x(i) = x(i) + x_p(i); x_p(i) = 0.0
end do
```

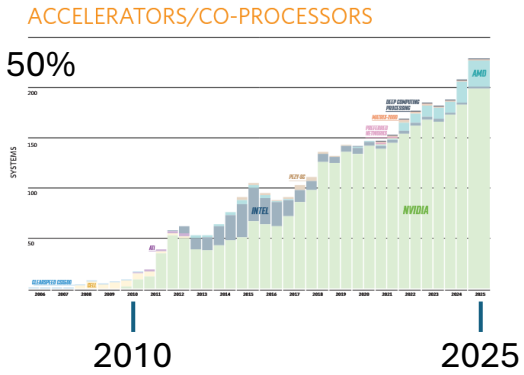
Avoid loop carried dependency using temporary arrays

YAMAGUCHI Junpei, YOKOTA Sho and BANNO Takahiro

CMC HPC/Exascale Background

178	**Underhill** - ThinkSystem SD650 V2, Xeon Platinum 8380 40C 2.3GHz, Infiniband HDR, Lenovo Shared Services Canada Canada	148,320	7.76	10.92	1,295
179	**Robert** - ThinkSystem SD650-N V2, Xeon Platinum 8380 40C 2.3GHz, Infiniband HDR, Lenovo Shared Services Canada Canada	148,320	7.76	10.92	1,295

- The move to consider exascale models has slowed because of increasing investments (both human and computational) in AI:
  - More than half of the developers of the GEM dynamical core have been moved to full-time AI model development
- Given expected computing growth, the CMC is very unlikely to hit “exascxale” in the next 15 years, making any associated planning very uncertain
- One thing that does seem clear is that if we hit exascale – even with physically based models – it will be using GPUs rather than CPUs



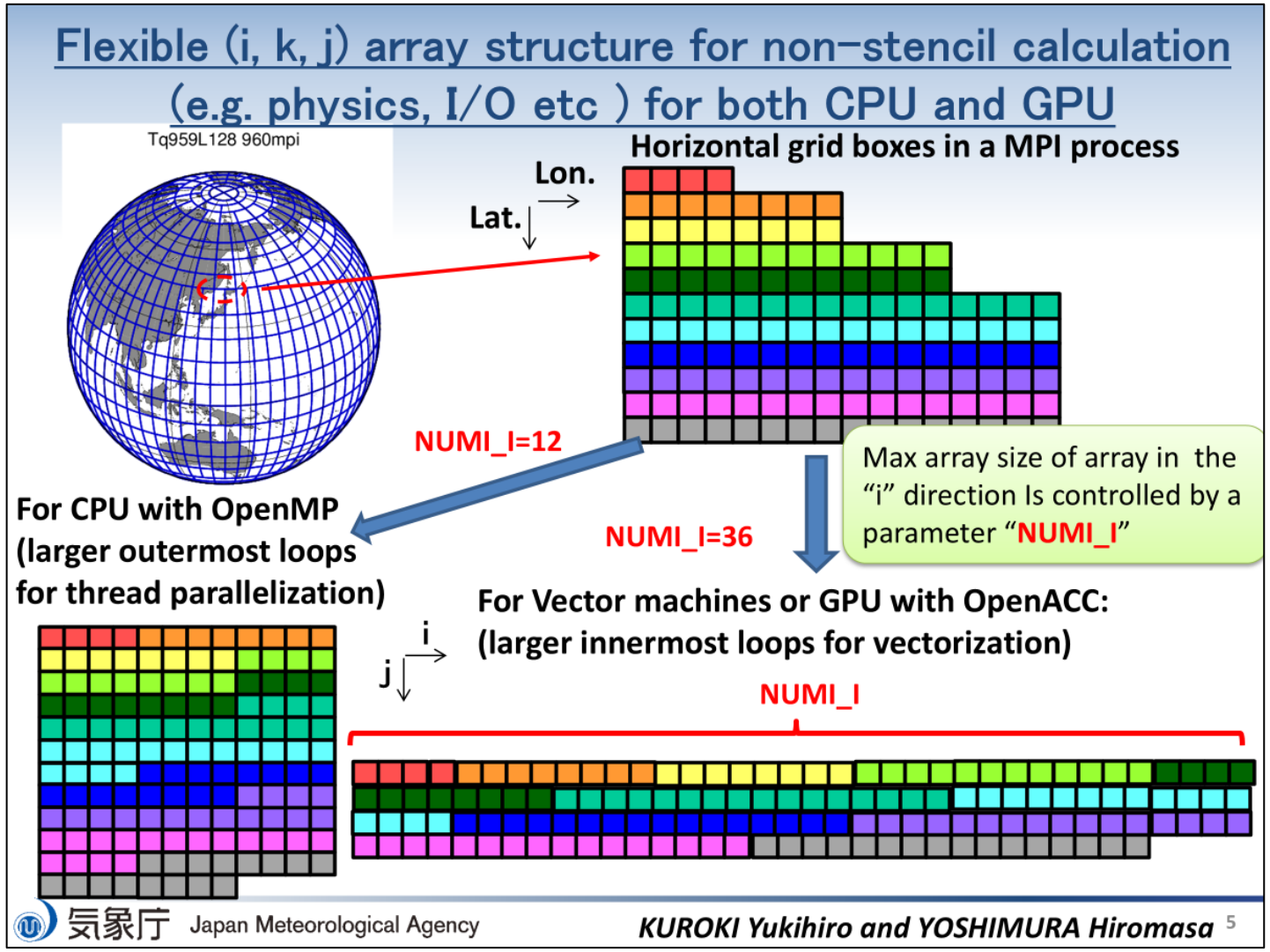
Canadian NWP / climate supercomputers from the Top500 list (top). Current prevalence of accelerators in the Top500 (bottom).

Preparing for GPU-based Architectures

- We have initiated a collaboration with the NOAA Global Systems Laboratory to investigate strategies for GPU porting:
  - GSL developers have made progress on individual parameterizations
  - ECCC does not have the resources to rewrite and maintain GEM (dynamics or physics) with a domain specific language while continuing development
  - Result needs to be able to run efficiently on both CPU and GPU
- Our current strategy is incremental
  1. Make use of vendor and open-source GPU-ready math libraries
  2. Refactor to improve memory management and loop structures
  3. Profile optimized code and introduce loop-level OpenMP offload to GPU
  4. Identify sections of code that can execute simultaneously on CPU / GPU
- Challenges with compiler support and performance
  - Collaboration with vendors might be necessary-but “lock-in” to be avoided

HPC Infrastructure at IITM / NCMRWF / MoES

- The IITM system is equipped with a capacity of **11.77 Peta FLOPS** and 33 petabytes of storage
- NCMRWF facility features **8.24 Peta FLOPS** with 24 petabytes of storage.
- Additionally, there is a dedicated standalone system for Artificial Intelligence and Machine Learning applications with a capacity of **1.9 Peta FLOPS**.
- With this augmentation, the Ministry of Earth Sciences will enhance its total computing power to **22 Peta FLOPS**, a substantial increase from the previous capacity of 6.8 Peta FLOPS



Typical source code in GSM

```
integer(4) :: WPR = kind(1.0d0) ! double
real(kind=WPR), dimension(NUMI_I, NUMFA_I, NUMJ_S) ::
array1(:,:,:),array2(:,:,:)

!$OMP PARALLEL default(SHARED), private(i,k,j)
!$OMP DO schedule(DYNAMIC)
 do j = 1, NUMJ_S
  !$acc kernels &
  !$acc present(NUMI,array1, array2,...)
  do k = 1, NUMFA_I
   do i = 1, NUMI(j)
     array1(i,k,j) = "parallell calculation using array2(I,k,j)"
   end do
  end do
  !$acc end kernels
 end do
 !$OMP END DO
!$OMP END PARALLEL
```

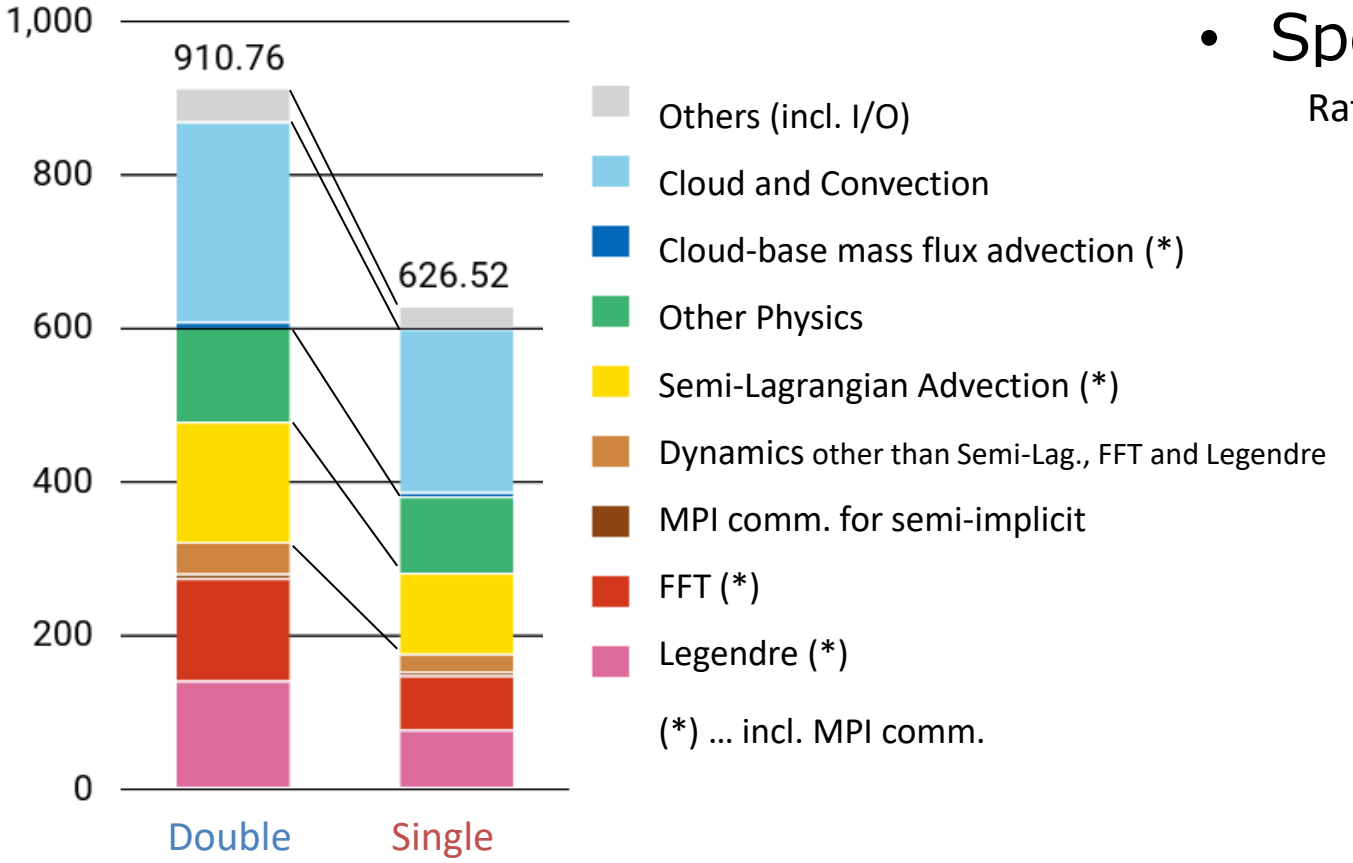
Development of reduced precision models

- GSM and ASUCA
  - Implemented switch functions between single/double precisions
  - Speed-up by 30% in the both models with the single precision mode, further speedup (~40%) is aimed.
  - Fixed model failure and significant numerical accuracy degradation issues, mainly due to loss of digits, information drop and overflow/underflow etc.…
    - TIPS and know-how accumulated
- MRI.COM
  - Mixed precision approach is tested
    - Several variables (e.g., mass and volume) need to be kept as double precision to obtain both speed up and accuracy

```
integer(4) :: WPR = kind(1.0d0) ! double  precision
!integer(4) :: WPR = kind(1.0e0) ! single precision
real(kind=WPR) :: some_data

some_data = 1.0_WPR
```

Single precision GSM



MPI rank average elapsed time [s] of Tq959 GSM (dx~13km) for 132hr time integration

49nodes, 390ranks (incl. 6 I/O ranks) and 14 OpenMP threads
on Fujitsu PRIMAGY CX2550 M7, Intel Fortran(2021.9.0) with "-O2" option

- Speed-up by 30%

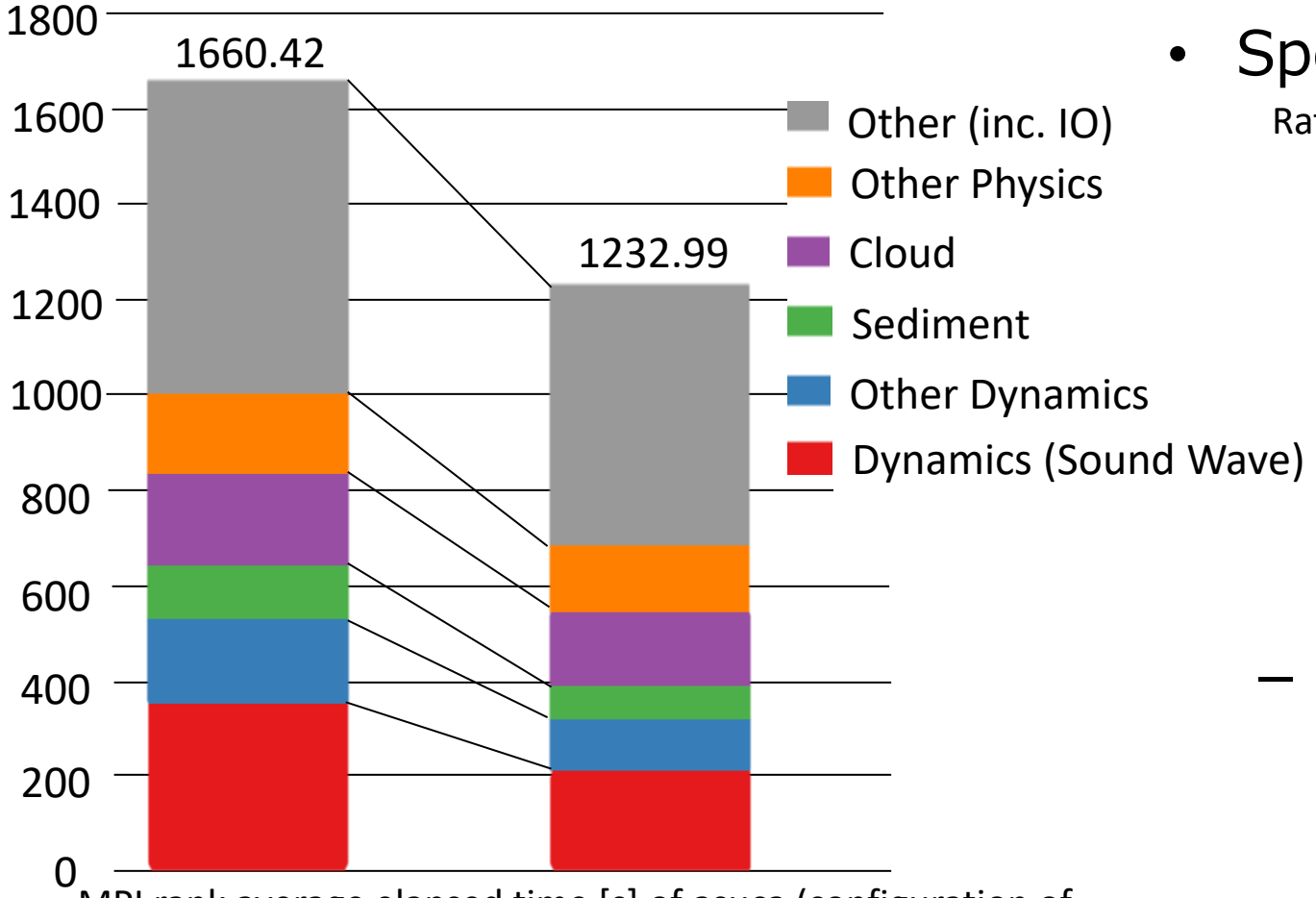
Ratio of elapsed time in single precision GSM to double precision

Process	Single/Double
Others (incl. I/O)	0.67
Cloud and Convection	0.82
Other physics	0.81
Semi-Lagrangian Advection	0.67
Other dynamics	0.54

Small speed-up rates in physics parameterization, particularly in specific subroutines presumably due to:

- Slow convergence in iterative algorithms
- SIMD suppression in loops with complex “if” branches

Single precision ASUCA



MPI rank average elapsed time [s] of asuca (configuration of LFM ,dx 2km) for 10.5hr time integration

19nodes, 304ranks (incl. 16 I/O ranks) and 14 OpenMP threads
on Fujitsu PRIMAGY CX2550 M7, Intel Fortran(2021.9.0) with "-O2" option

- Speed-up by 30%

Ratio of elapsed time in single precision asuca to double precision

Process	Single/Double
Others (incl. IO)	0.83
Other Physics	0.84
Cloud	0.81
Sediment	0.62
Other Dynamics	0.61
Dynamics (Sound Wave)	0.60

– Small speed-up rates in physics parameterization, particularly in specific subroutines presumably due to:
  - SIMD suppression in loops with complex “if” branches

Issues and remedies in transition from double to single precision

- **Issue:** zero-division at time-step mean solar zenith calculation (based on Hogan and Hirahara, 2016) : The issue occurs at sunrise at the end / sunset at the start of a timestep
- **Remedy:** use an approximated form without near zero-division
- Note that this issue is atmospheric state-independent (only dependent on date, spatial and time resolution), thus predictable whether / when / where a model fails

Before

$$\overline{\cos\theta} = \sin\delta\sin\phi + \frac{\cos\delta\cos\phi(\sin h_{\max} - \sin h_{\min})}{h_{\max} - h_{\min}}$$

In a model failure case:

$h_{\max} - h_{\min} = 0$ in single precision (zero due to loss of digit)

note: minimum value* in double precision: $h_{\max} - h_{\min} \sim O\left(10^{-12}\right)$

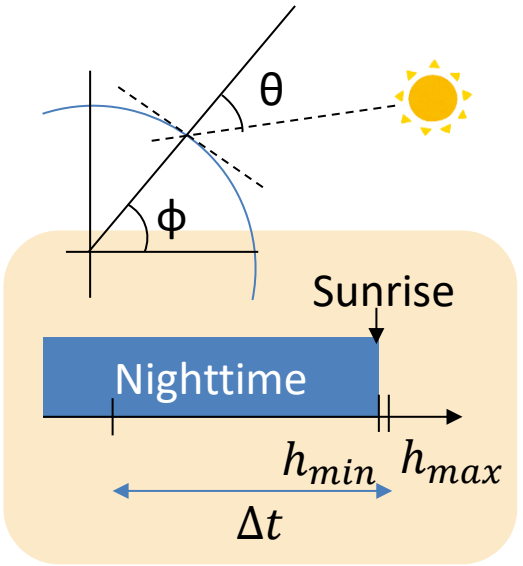
*… Tq959 (~13km), 24-hour forecast, cases for the next few years

After

$$\overline{\cos\theta} \sim \sin\delta\sin\phi + \cos\delta\cos\phi\cos\left(h_{\max} - h_{\min}\right)$$

$$\text{if } h_{\max} - h_{\min} < \epsilon$$

(ε is set by considering numerical safety and accuracy of the approximation, 10^{-7} for double precision, to be set for single precision)



δ : Declination ϕ : latitude
h_{min} *and* h_{min}: start and end of daytime in a time step

Issues and remedies in transition from double to single precision

- **Issue:** overflow in a tri-diagonal matrix solver for the boundary layer and land surface schemes due to multiplying diagonal components with large values
- **Remedy:** use mathematically identical but numerically safer procedures without multiplying large values in forward-elimination

$$\begin{pmatrix} b_{\phi,1} & c_{\phi,1} & & & 0 \\ a_{\phi,2} & b_{\phi,2} & c_{\phi,2} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ 0 & & & a_{\phi,n-1} & b_{\phi,n-1} \\ & & & a_{\phi,n} & b_{\phi,n} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{n-2} \\ \phi_{n-1} \\ \phi_n \end{pmatrix} = \begin{pmatrix} p_{\phi,1} \\ p_{\phi,2} \\ p_{\phi,3} \\ \vdots \\ p_{\phi,n-2} \\ p_{\phi,n-1} \\ p_{\phi,n} \end{pmatrix} \quad \text{where} \quad b_{\phi,k} > 1$$

Before

$$\begin{aligned} &do\ k = n, 1, -1 \\ &\quad b'_{\phi,k-1} = b_{\phi,k-1} b'_{\phi,k} - a'_{\phi,k} c_{\phi,k-1} \\ &\quad a'_{\phi,k-1} = a_{\phi,k-1} b'_{\phi,k} \\ &\quad p'_{\phi,k-1} = p_{\phi,k-1} b'_{\phi,k} - p'_{\phi,k} c_{\phi,k-1} \\ &end\ do \\ &do\ k = 1, n \\ &\quad \phi_k = (p'_{\phi,k} - a'_{\phi,k} \phi_{k-1}) / b'_{\phi,k} \\ &end\ do \end{aligned}$$

Forward elimination

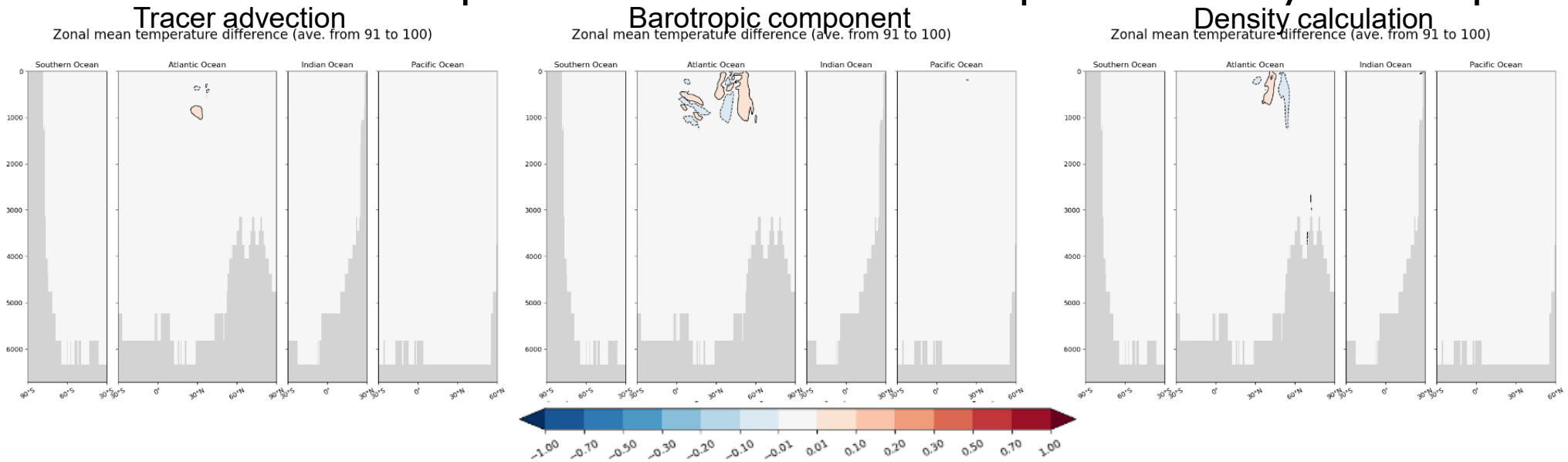
Backward substitution

After

$$\begin{aligned} &do\ k = n, 1, -1 \\ &\quad b'_{\phi,k-1} = b_{\phi,k-1} - a'_{\phi,k} c_{\phi,k-1} \\ &\quad a'_{\phi,k-1} = a_{\phi,k-1} / b'_{\phi,k-1} \\ &\quad p'_{\phi,k-1} = (p_{\phi,k-1} - p'_{\phi,k} c_{\phi,k-1}) / b'_{\phi,k-1} \\ &end\ do \\ &do\ k = 1, n \\ &\quad \phi_k = p'_{\phi,k} - a'_{\phi,k} \phi_{k-1} \\ &end\ do \end{aligned}$$

Mixed precision in MRI.COM

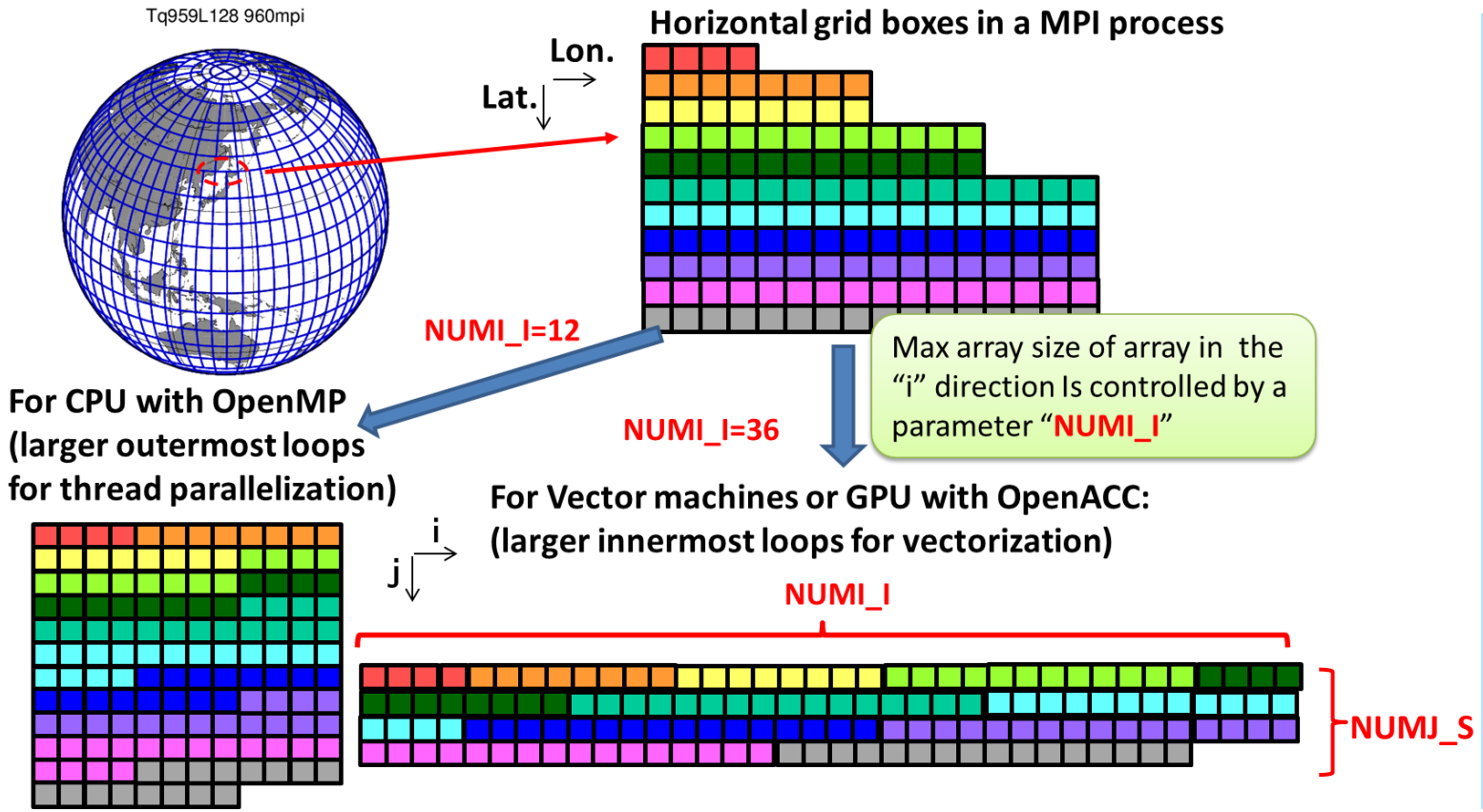
- Mixed precision approach: Several variables (e.g., mass and volume) need to be kept as double precision to obtain both speed up and accuracy
  - Tracer advection, momentum equation for barotropic component, density calculation
- To be finished implementation of mixed precision by next spring



Zonal mean temperature difference for the last 10-yr average of the 100-yr integration [K](mix - double)

GPU porting: Status and plans

- GSM
  - most of processes (parallelization, spectral transform, dynamics, cloud and convection) have been ported to GPU. To be completed porting by next spring
    - For GSM, full-GPU appears to be better rather than offloading only for bottlenecks, as optimum data structure for OpenMP (CPU) and OpenACC (GPU) parallelization is different.
- ASUCA
  - GPU porting completed (except I/O). Preliminary results were reported to WGNE-38.
  - Further evaluation and optimization are ongoing.
- MRI.COM
  - Several bottlenecks (tracer-advection, momentum equations for barotropic and baroclinic components) have been ported.
  - To be completed porting in a few years



Typical source code in GSM

```
integer(4) :: WPR = kind(1.0d0) ! double
real(kind=WPR), dimension(NUMI_I, NUMFA_I, NUMJ_S) ::
array1(:,:,:),array2(:,:,:)

!$OMP PARALLEL default(SHARED), private(i,k,j)
!$OMP DO schedule(DYNAMIC)
 do j = 1, NUMJ_S
  !$acc kernels &
  !$acc present(NUMI,array1, array2,...)
  do k = 1, NUMFA_I
   do i = 1, NUMI(j)
     array1(i,k,j) = "parallell calculation using array2(I,k,j)"
   end do
  end do
  !$acc end kernels
 end do
 !$OMP END DO
!$OMP END PARALLEL
```

Elapsed time [s] of Tl159L128 GSM (~110km) for 6hour time integration (1node, 8MPI, 14threads, 8GPU)

	All CPU (optimized outermost loop for CPU(OpenMP))	Semi-Lag. on GPU, others on CPU (optimized innermost loop for GPU(OpenACC))
Semi-Lagrangian advection	2.9413	1.5076
Cloud and convection	1.0733	13.881
Other physics	2.3951	20.433

Optimized for GPU, but awful performance in CPU

CPU: Intel Xeon Gold 6338 2.0GHz 32C/64T x2 with DDR4 memory (204GB/s)
GPU: NVIDIA A100 SXM2 80GB x8 with HBM2e memory (2039GB/s)