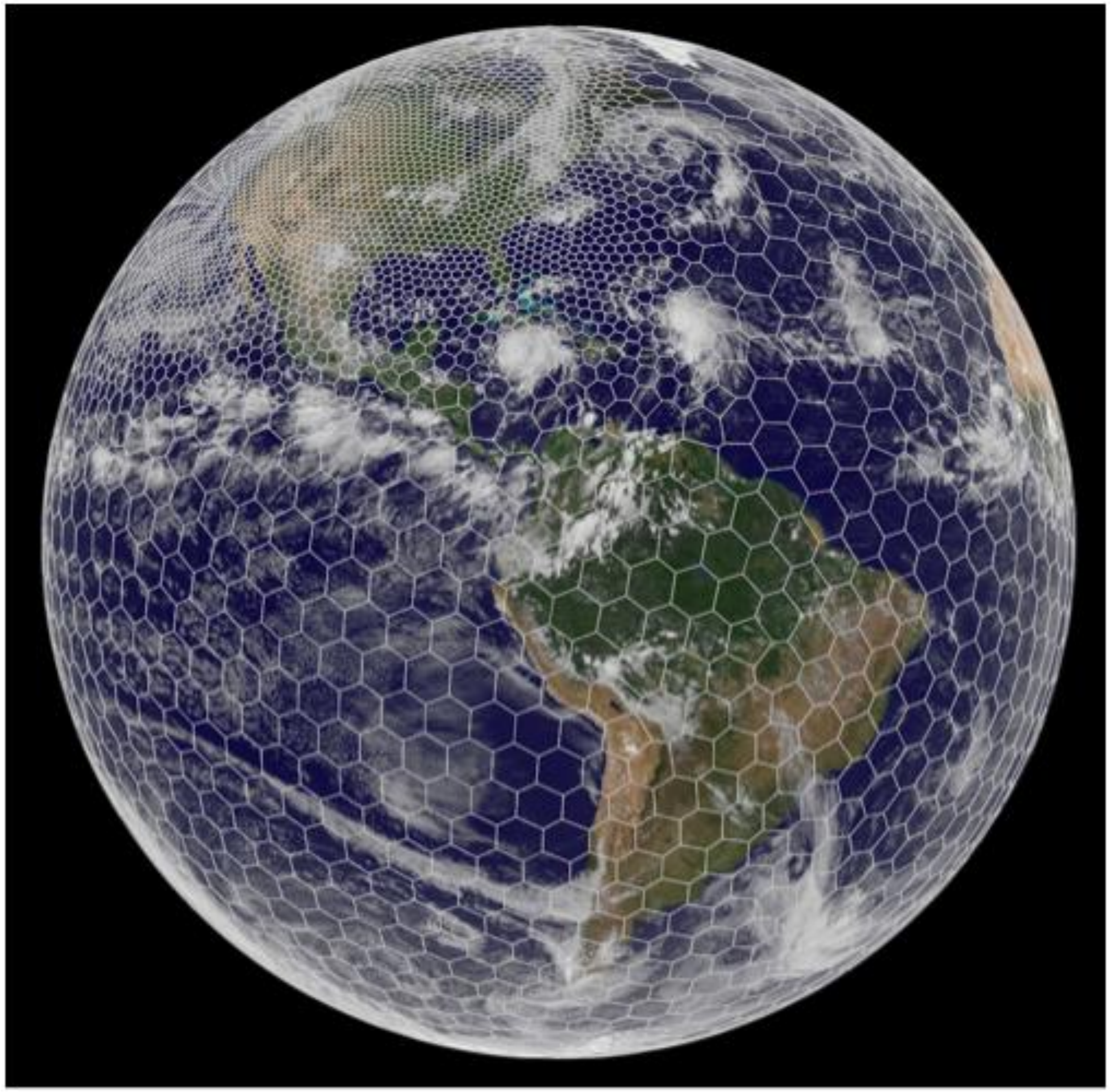
MPAS on GPUs Using OpenACC

Supreeth Suresh
Software Engineer II
Special Technical Projects (STP) Group
National Center for Atmospheric Research

26th September, 2019

Outline

- Team
- Introduction
- System and Software Specs
- Approach, Challenges & Performance
  - Dynamical core
    - Optimizations
    - Scalability
  - Physics
- Questions



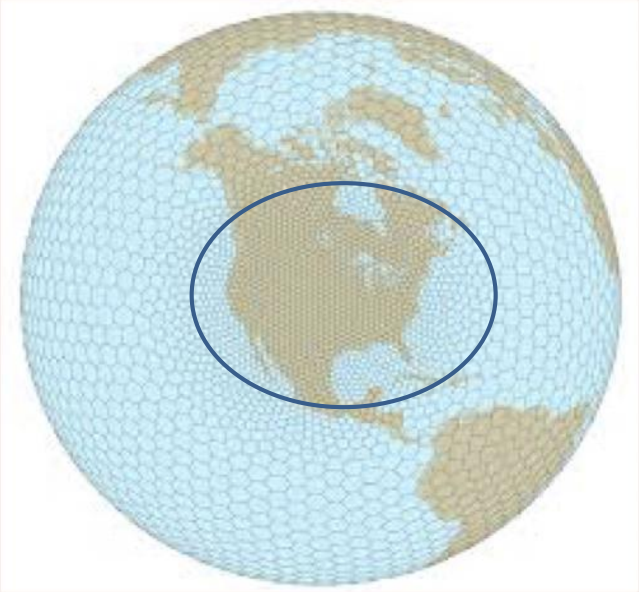
Our Team of Developers

- **NCAR**
  - Supreeth Suresh, Software Engineer, STP
  - Cena Miller, Software Engineer, STP
  - Dr. Michael Duda, Software Engineer, MMM
- **NVIDIA/PGI**
  - Dr. Raghu Raj Kumar, DevTech, NVIDIA
  - Dr. Carl Ponder, Senior Applications Engineer
  - Dr. Craig Tierney, Solutions Architect
  - Brent Leback, PGI Compiler Engineering Manager
- **University of Wyoming:**
  - GRAs: Pranay Kommera, Sumathi Lakshmiranganatha, Henry O’Meara, George Dylan
  - Undergrads: Brett Gilman, Briley James, Suzanne Piver
- **IBM/TWC**
- **Korean Institute of Science and Technology Information**
  - Jae Youp Kim, GRA

David Gutzwiller
Lead Software Developer
NUMECA
“Porting our unstructured C++ CFD solver FINE/Open to GPUs using OpenACC would have been impossible two or three years ago, but OpenACC has developed enough that we're now getting some really good results.”
MPAS-A
Richard Loft
Director, Technology Development NCAR
“Our team has been evaluating OpenACC as a pathway to performance portability for the Model for Prediction (MPAS) atmospheric model. Using this approach on the MPAS dynamical core, we have achieved performance on a single P100 GPU equivalent to 2.7 dual socketed Intel Xeon nodes on our new Cheyenne supercomputer.”

MPAS Grids…

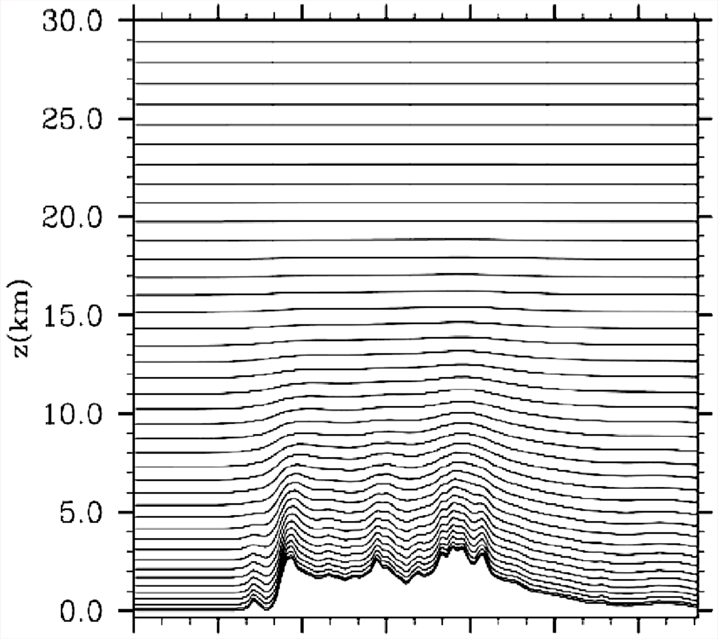
Horizontal



MPAS
Unstructured Voronoi
(hexagonal) grid

- Good scaling on massively parallel computers
- No pole problems

Vertical



MPAS
Height-based hybrid smoothed
terrain-following vertical coordinate

- Improved numerical accuracy

MPAS Time-Integration Design

Default time integration

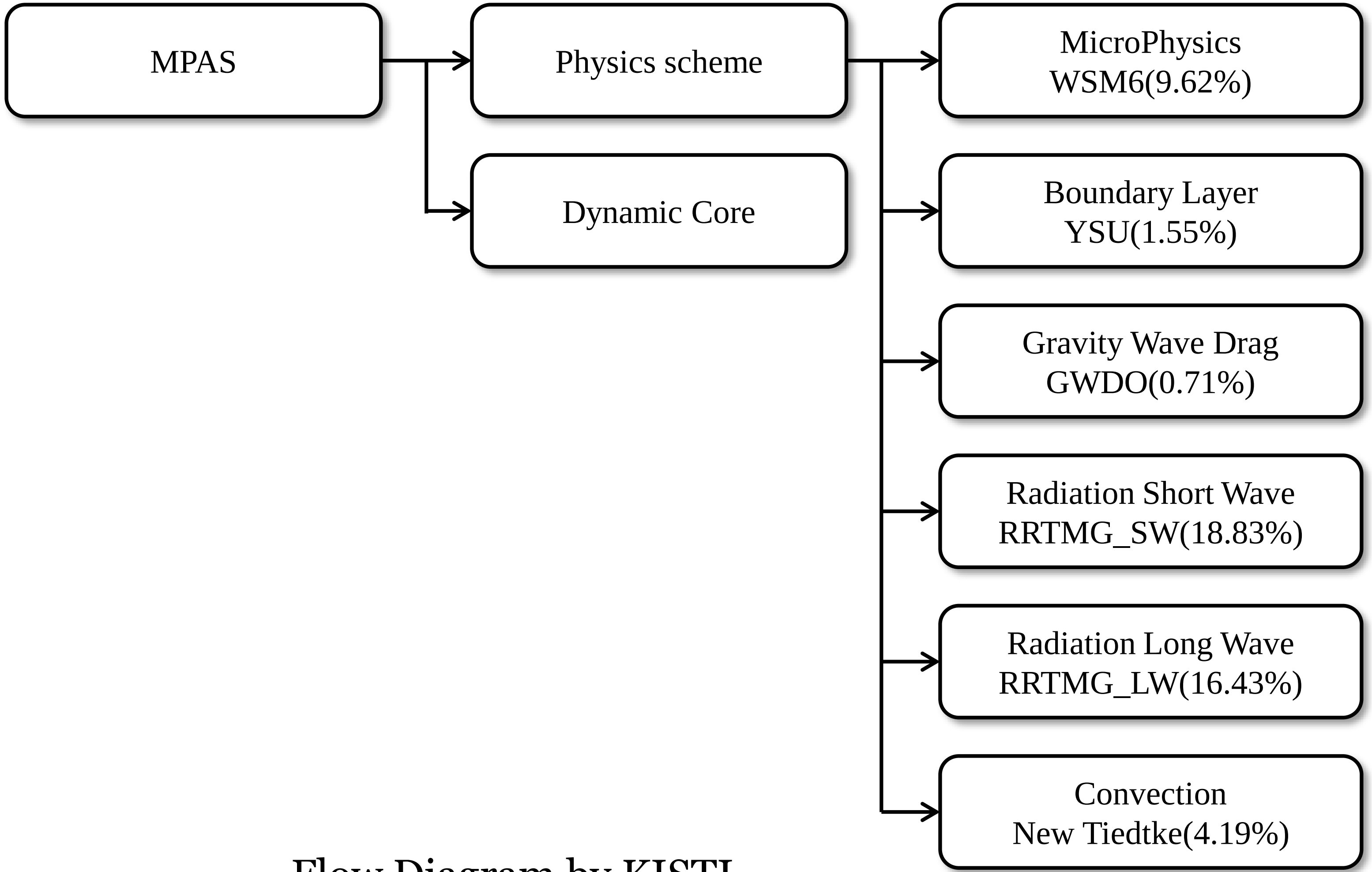
```
Call physics

Do dynamics_split_steps
    Do step_rk3 = 1, 3
        compute large-time-step tendency
        Do acoustic_steps
            update u
            update rho, theta and w
        End acoustic_steps
    End rk3 step
End dynamics_split_steps

Do scalar step_rk3 = 1, 3
    scalar RK3 transport
End scalar rk3 step

Call microphysics
```

There are 100s of halo exchanges /timestep!

Allows for smaller dynamics timesteps relative to scalar transport timestep and main physics timestep.

We can use any FV scheme here (we are not tied to RK3)
Scalar transport and physics are the expensive pieces in most applications.

Where to begin?



Flow Diagram by KISTI

Execution time-

Physics: 45-50%

DyCore: 50-55%

Lines of Code-

Physics: 110,000

DyCore: 10,000

System Specs

- **NCAR Cheyenne supercomputer**
  - 2x 18-core Intel Xeon v4 (BWL)
  - Intel compiler 19
  - 1x EDR IB interconnect; HPE MPT MPI
- **Summit and IBM "*WSC*" supercomputer**
  - AC922 with IB interconnect
  - 6 GPUs per node; 2x 22-core IBM Power-9
  - 2x EDR IB interconnect; IBM Spectrum MPI

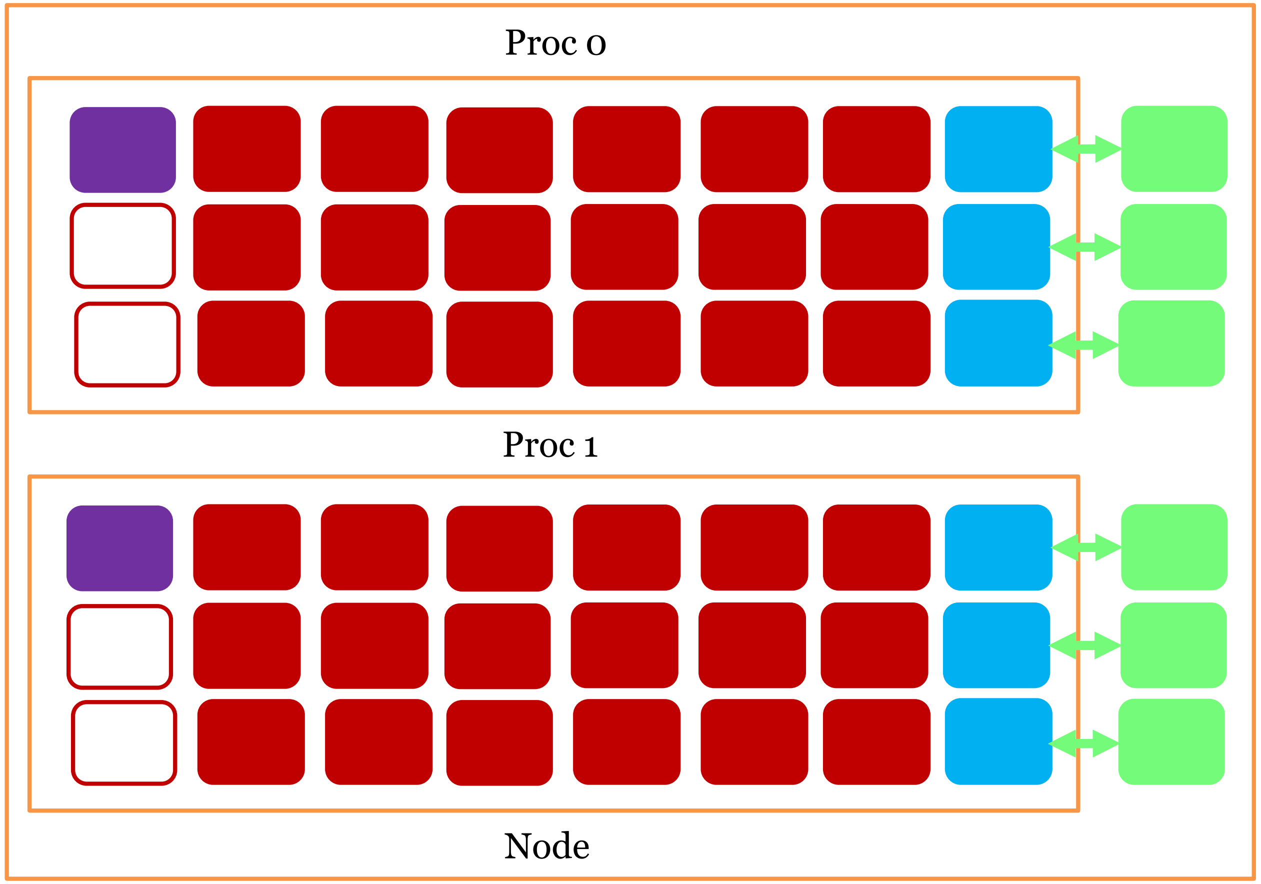
Software Spec: MPAS Dynamical Core

- **Software**
  - MPAS 6.x
  - PGI Compiler 19.4, Intel Compiler 19
- **Moist Baroclinic Instability Test- No physics**
  - Moist dynamics test-case produces baroclinic storms from analytic initial conditions
  - Split Dynamics: 2 sub-steps, 3 split steps
  - 120 km (40k grid points, dt=720s) , 60 km resolution (163k grid points, dt=300s), 30 km resolution (655k grid points, dt=150s) , 15 km resolution (2.6M grid points, dt=90s), 10 km resolution (5.8M grid points, dt=60s) , 5 km resolution (23M grid points, dt=30s)
  - Number of levels = 56, Single precision (SP)
  - Simulation executed for 16 days, **performance shown for 1 timestep**

Software Spec: MPAS

- **Software**
  - MPAS 6.x
  - PGI Compiler 19.4, Intel Compiler 19
- **Full physics suite**
  - Scale-aware Ntiedtke Convection, WSM 6 Microphysics, Noah Land surface, YSU Boundary Layer, Monin-Obhukov Surface layer, RRTMG radiation, Xu Randall Cloud Fraction
  - Radiation interval: 30 minutes
  - Single precision (SP)
  - Optimization and Integration in progress, **performance shown for 1 timestep**

MPAS-GPU Process Layout on IBM node



🟦 MPI & NOAH control path

🟥 CPU – SW/LW Rad & NOAH

🟩 GPU – everything else

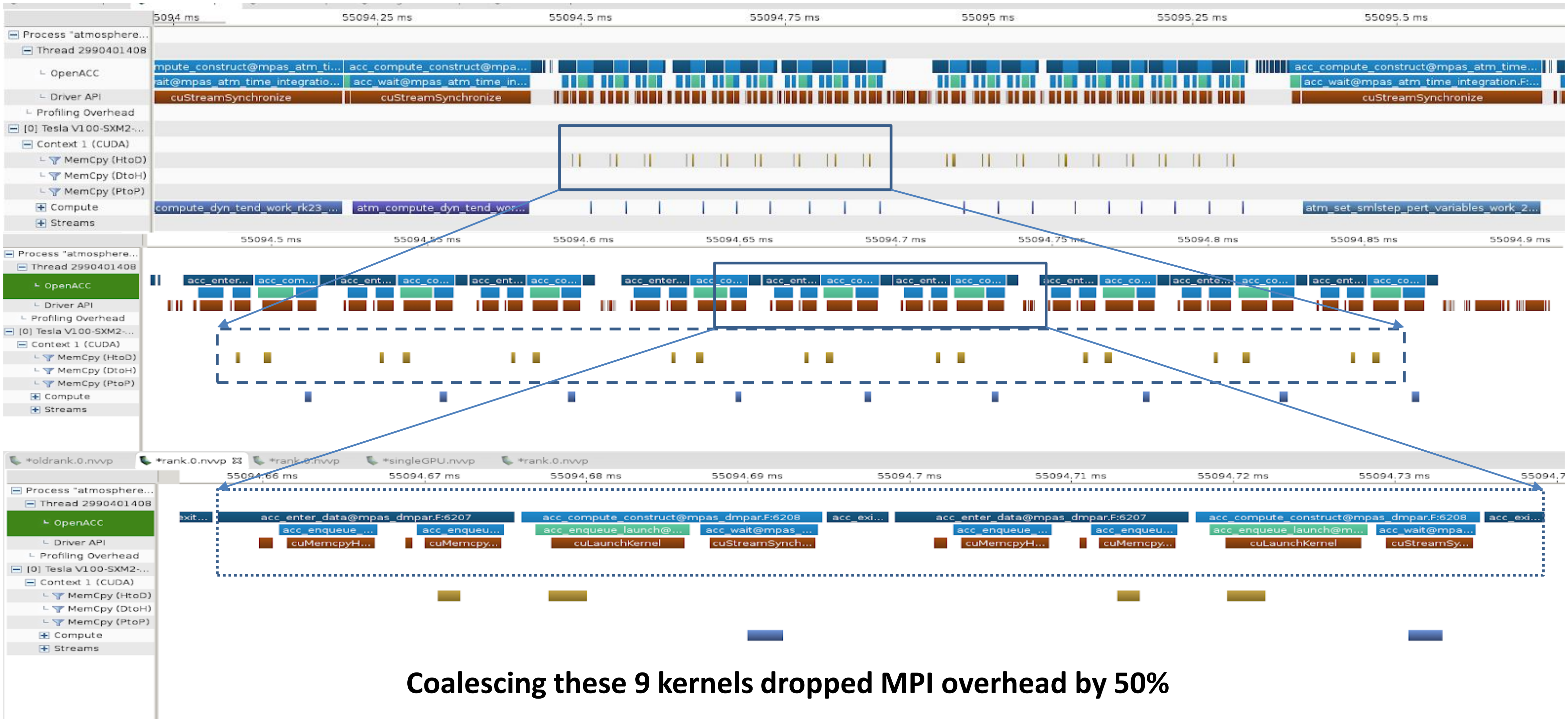
🟪 Asynch I/O process

□ Idle processor

MPAS dycore halo exchange

- **Approach**
  - Original halo exchange written with linked lists
    - OpenACC loved it!
  - MMM rewrote halo exchange with arrays
    - Worked with OpenACC, but huge overhead due to book keeping on CPU
    - Moved MPI book keeping on GPUs
      - Bottleneck was send/recv buffer allocations on CPU
  - MMM rewrote halo exchange with once per execution buffer allocation
    - No more CPU overheads
  - STP and NVIDIA rewrote the halo exchange to minimize the data transfers of the buffer

Improving MPAS-A halo exchange performance: coalescing kernels

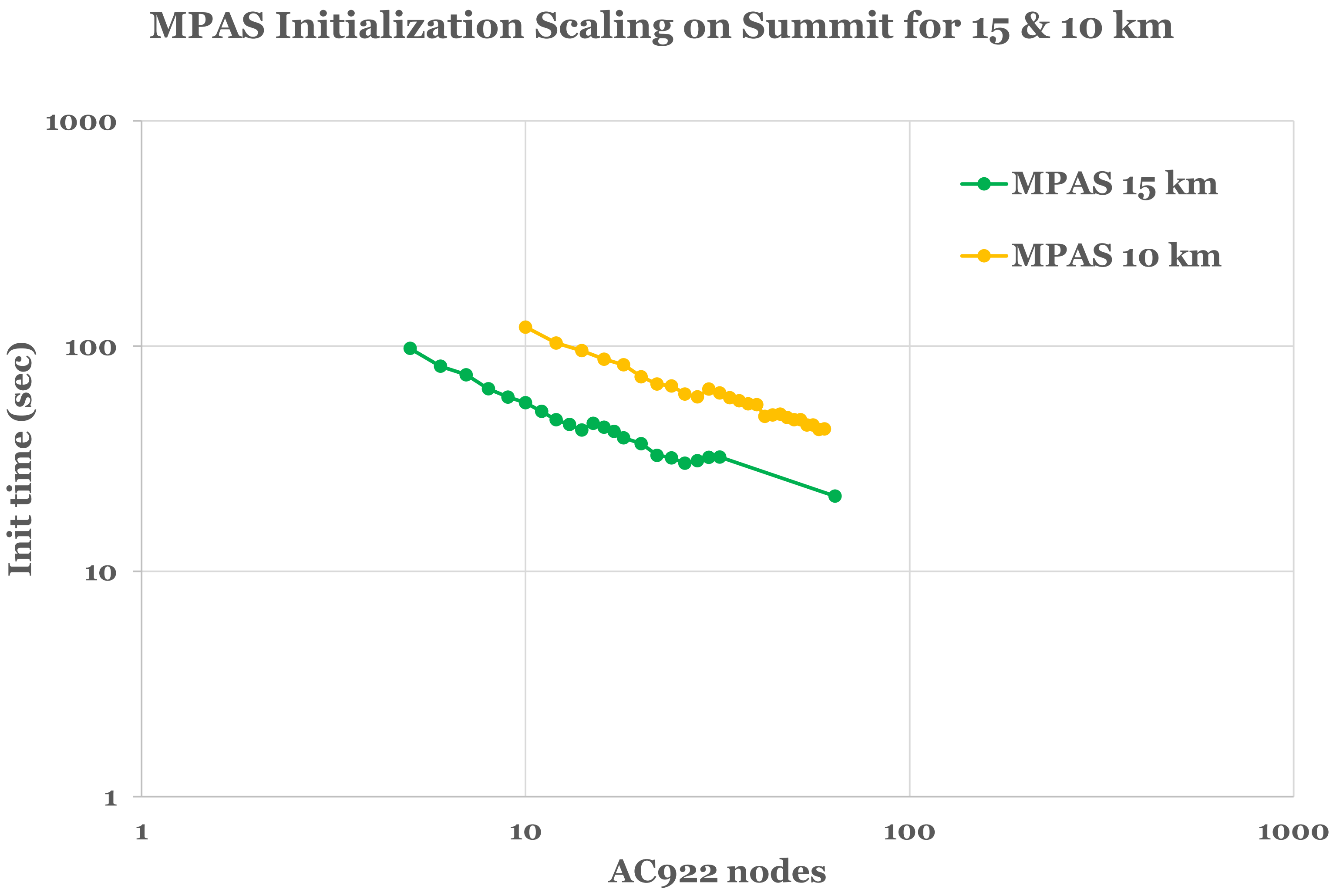


Coalescing these 9 kernels dropped MPI overhead by 50%

Optimizing MPAS-A dynamical core: Lessons Learned

- Module level allocatable variables (20 in number) were unnecessarily being copied by compiler from host to device to initialize them with zeroes. Moved the initialization to GPUs.
- dyn_tend: eliminated dynamic allocation and deallocation of variables that introduced H<->D data copies. It’s now statically created.
- MPAS_reconstruct: originally kept on CPU was ported to GPUs.
- MPAS_reconstruct: mixed F77 and F90 array syntax caused compiler to serialize the execution on GPUs. Rewrote with F90 constructs.
- Printing out summary info (by default) for every timestep consumed time. Turned into debug option.

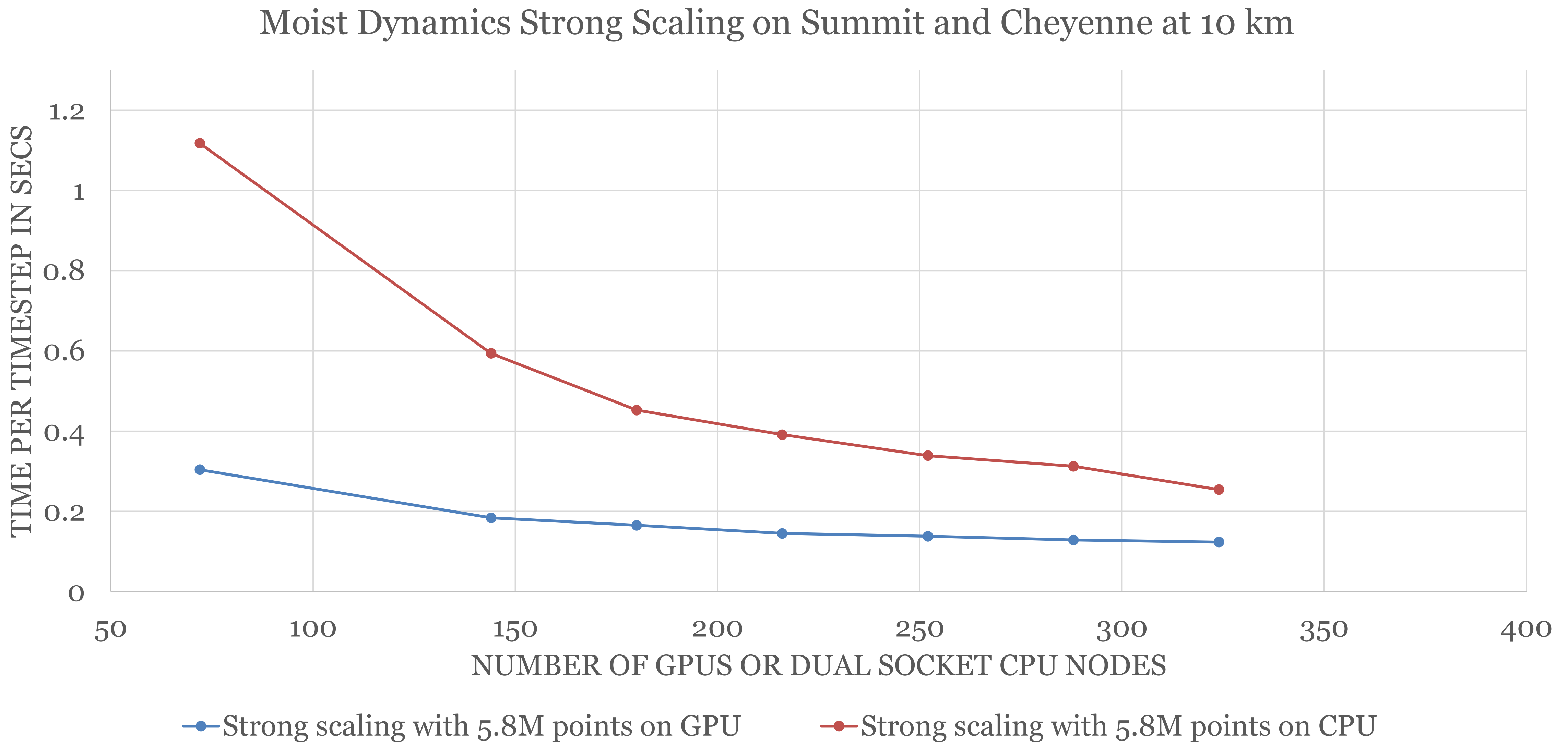
Scalable MPAS Initialization on Summit: CDF5 performance



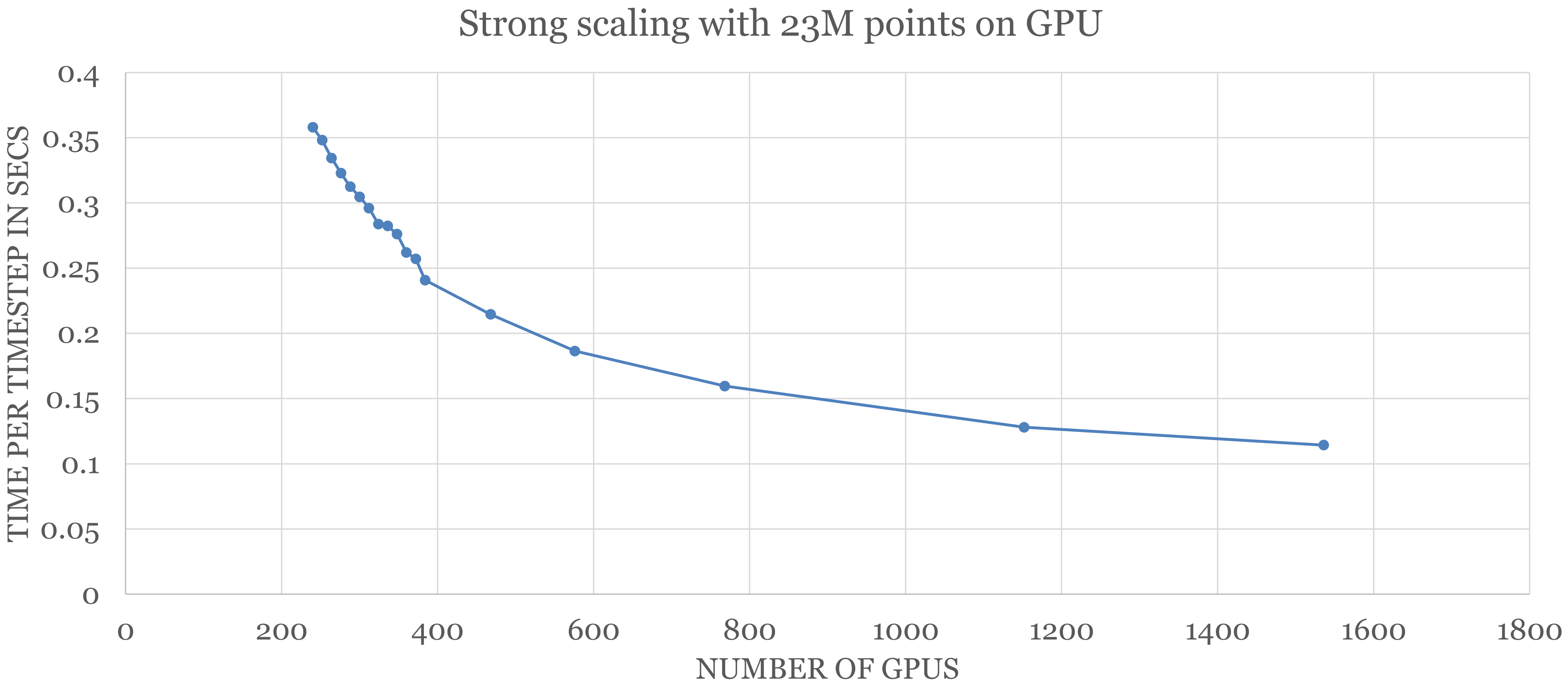
Strong scaling benchmark test setup

- **MPAS-A Version 6.x**
- **Test case: Moist dynamics**
- **Compiler:** GPU **-** PGI 19.4, CPU - Intel 19
- **MPI:** GPU **-** IBM spectrum, CPU - Intel MPI
- **CPU:** 2 socket Broadwell node with 36 cores
- **GPU:** NVIDIA Volta V100
- **10, 5 km problem**
  - Timestep: 60, 30 sec
  - Horizontal points/rank: 5,898,242 points, 23,592,962 points(uniform grid)
  - Vertical: 56 levels

Strong scaling



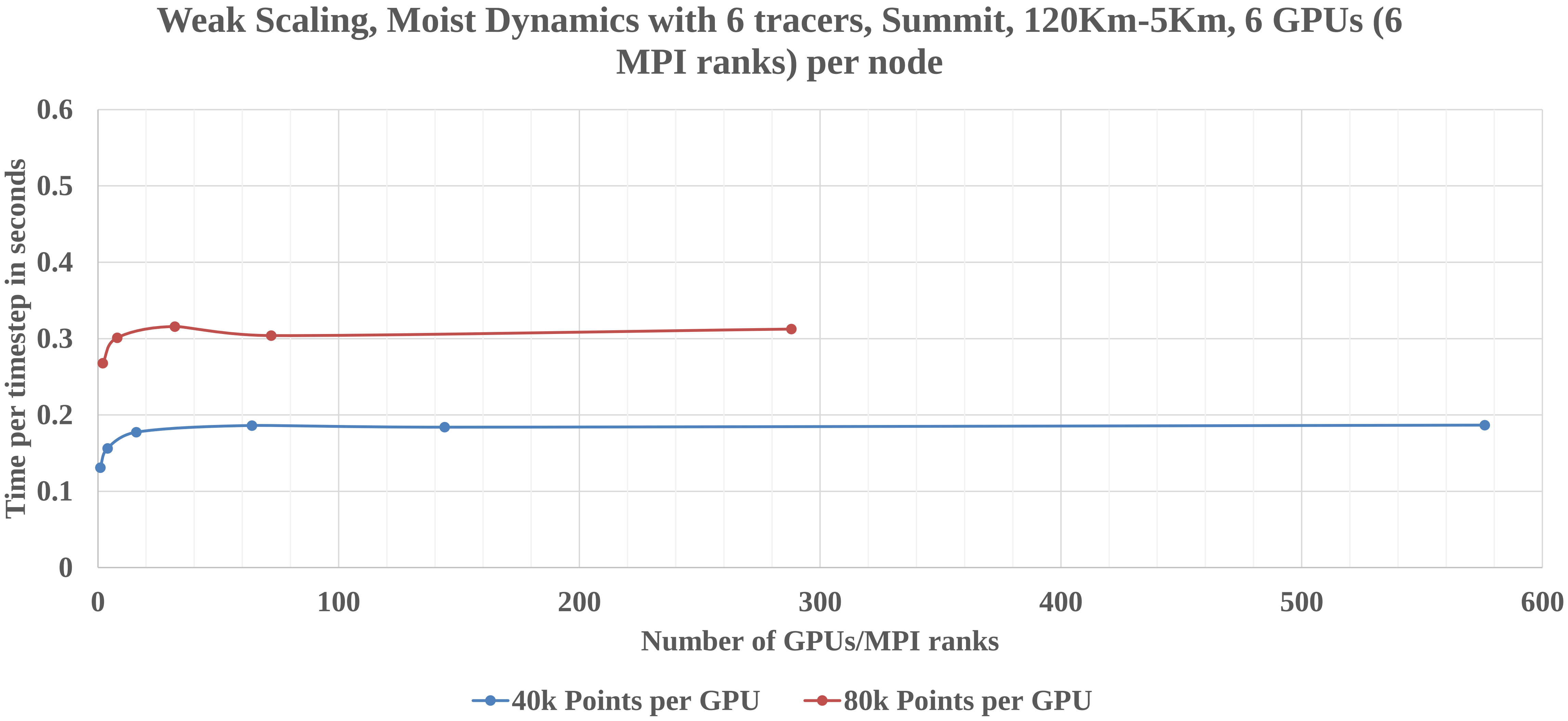
Moist dynamics strong scaling at 5km



Weak scaling benchmark test setup

- **MPAS-A Version 6.x**
- **Test case: Moist dynamics**
- **Compiler:** GPU **-** PGI 19.4, CPU - Intel 19
- **MPI:** GPU **-** IBM spectrum, CPU - Intel MPI
- **CPU:** 2 socket Broadwell node with 36 cores
- **GPU:** NVIDIA Volta V100
- **120-60-30-15-10-5 km problem**
  - Timestep: 720, 300, 180, 90, 60, 30 sec
  - Horizontal points/rank: 40,962 points, 81,921 points (uniform grid)
  - Vertical: 56 levels

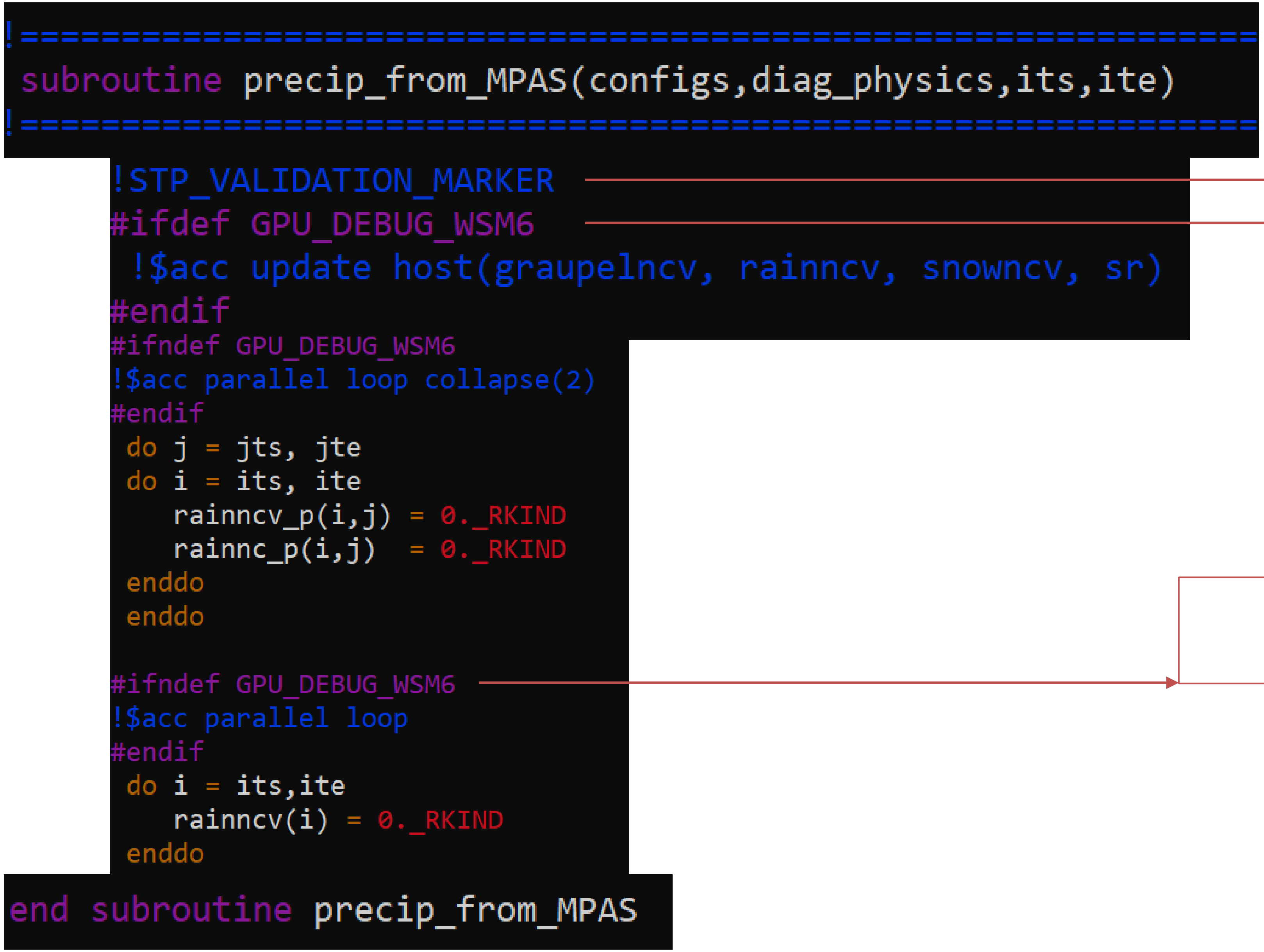
Weak scaling



MPAS Physics- Order of tasks

- Build a methodology that supports re-integration for all physics modules (50%)
  - Must be flexible to validate or integrate
  - Must be able to run individual portions on CPU/GPU as required
- Upgrade, Integrate, Validate & Optimize WSM6(20%)
- Benchmark Dycore-scalar-WSM6
- Upgrade, Integrate & Validate YSU and Gravity Wave Drag(15%)
- Benchmark Dycore-scalar-WSM6-YSU-GWDO
- Upgrade, Integrate & Validate Monin Obhukov (5%)
- Benchmark Dycore-scalar-WSM6-YSU- Monin Obhukov
- Upgrade, Integrate & Validate Ntiedtke (10%)
- Benchmark Full MPAS

What does a methodology look like?



```
!=============================================================
 subroutine precip_from_MPAS(configs,diag_physics,its,ite)
!=============================================================
     !STP_VALIDATION_MARKER
     #ifdef GPU_DEBUG_WSM6
      !$acc update host(graupelncv, rainncv, snowncv, sr)
     #endif
     #ifndef GPU_DEBUG_WSM6
     !$acc parallel loop collapse(2)
     #endif
      do j = jts, jte
      do i = its, ite
         rainncv_p(i,j) = 0._RKIND
         rainnc_p(i,j)  = 0._RKIND
      enddo
      enddo

     #ifndef GPU_DEBUG_WSM6
     !$acc parallel loop
     #endif
      do i = its,ite
         rainncv(i) = 0._RKIND
      enddo
end subroutine precip_from_MPAS
```

Grep search help string

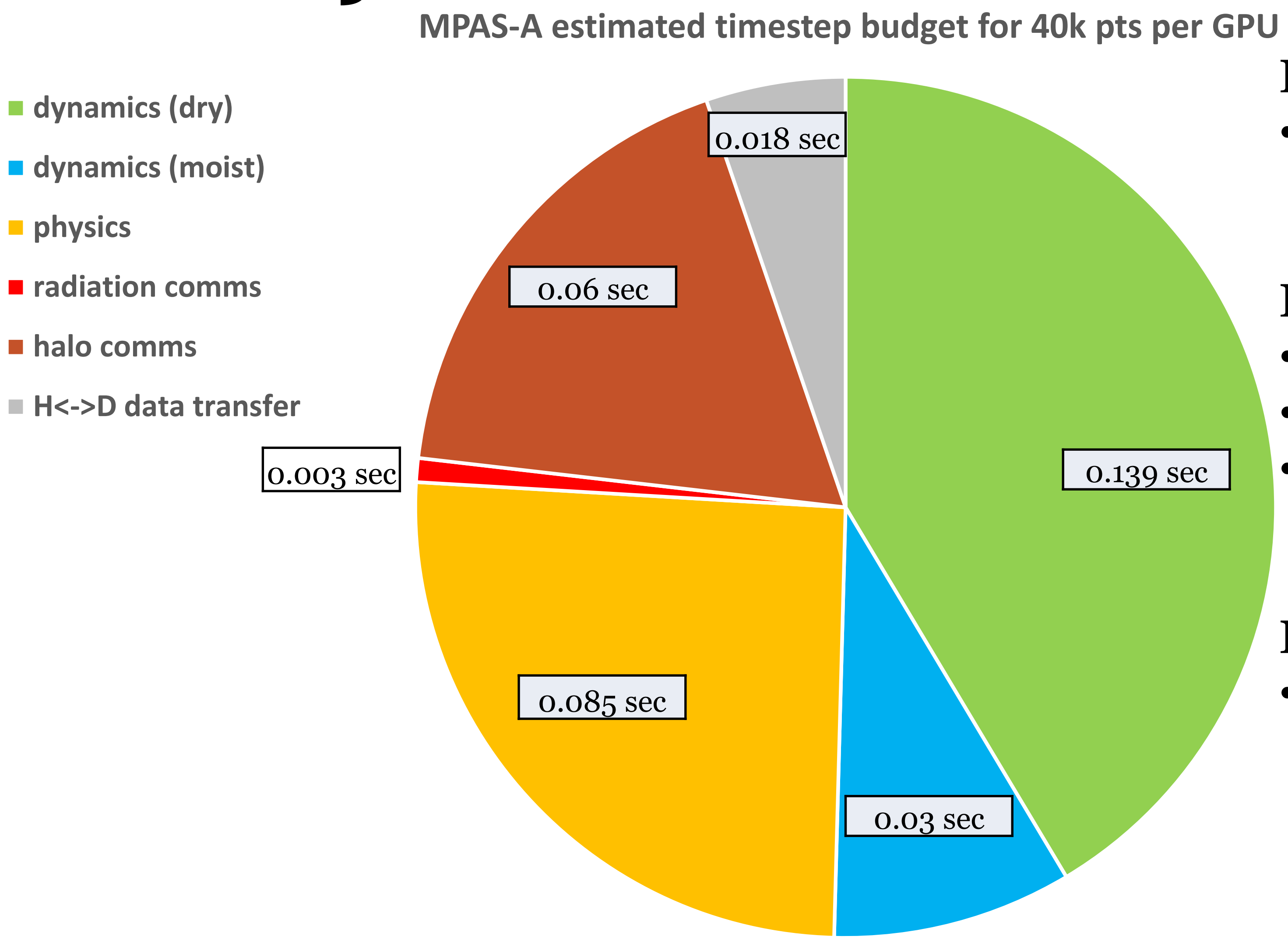
Preprocessor Directive to offload routine on CPU

Flip GPU/CPU based on requirement

Methodology description

- Repeat layout for all physics modules- Completes the framework
- The preprocessor directives will be removed after validation
- Methodology includes the required data directives
  - Noah & Radiation included

Projected Full MPAS Performance



Dynamics dry+moist+halo

- 0.18s instead of expected 0.22s

Physics- WSM6 + YSU

- 0.078s+0.008s = 0.086s
- Ntiedtke takes 0.04s on CPU
- Noah and MO together take less than 1msec on CPU

H<->D data transfer

- Pending

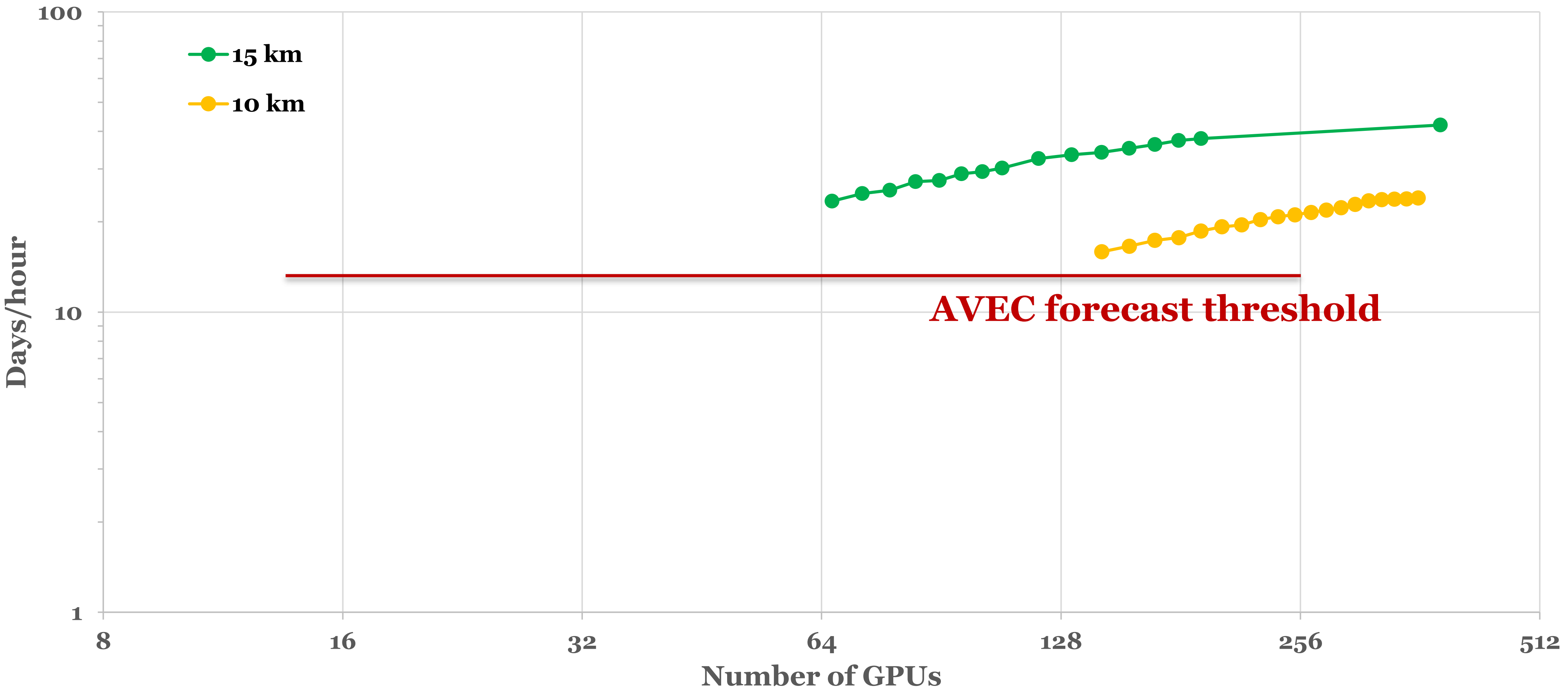
Total time: 0.275 sec/step
15 km -> 64 V100 GPUs
Throughput ~0.9 years/day

Future Work

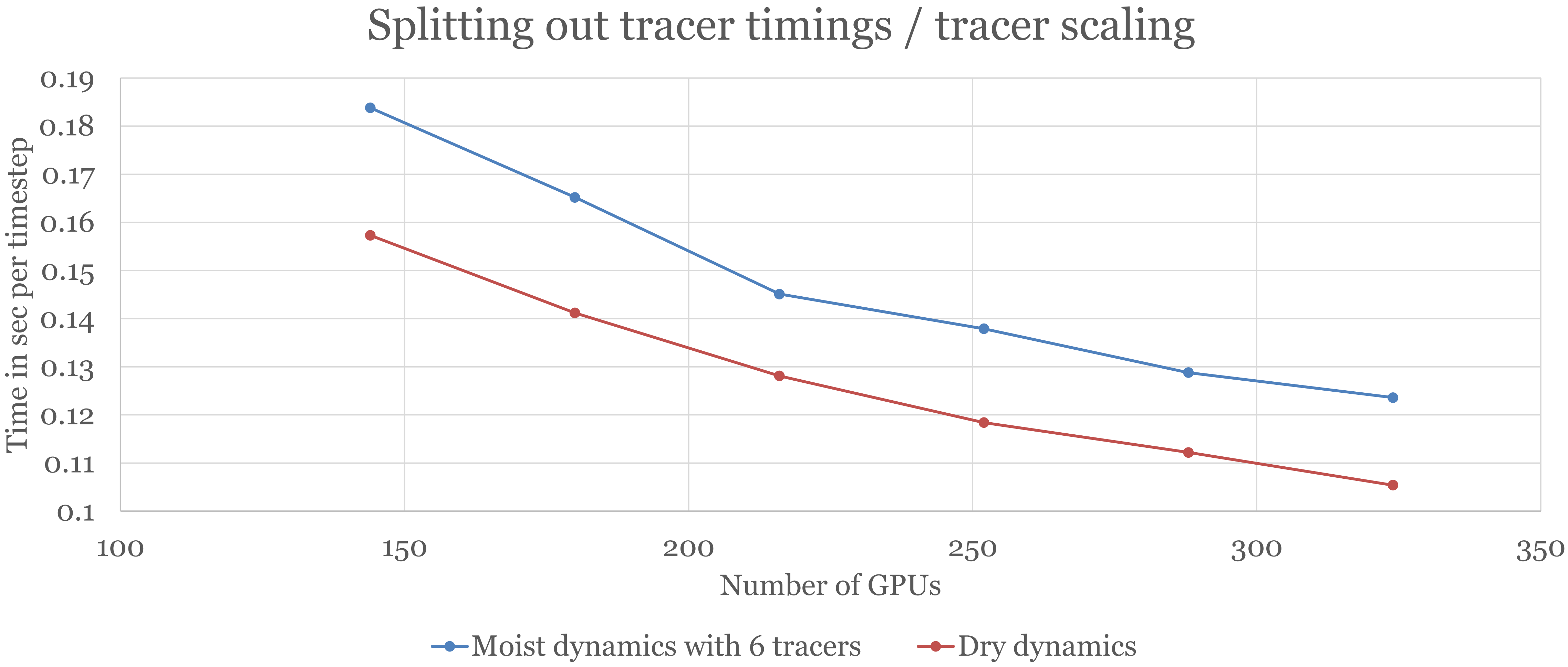
- **MPAS Performance**
  - Optimization of remaining physics schemes
  - Verification and Integration of remaining physics schemes
  - Integrating Lagged Radiation

Thank you! Questions?

Moist Dynamics Strong Scaling on Summit at 10 & 15 km



How does the scaling compare to dry dynamics?


Splitting out tracer timings / tracer scaling
Time in sec per timestep
Number of GPUs
Moist dynamics with 6 tracers
Dry dynamics