Model for Prediction Across Scales (MPAS) on GPUs

Supreeth Suresh
National Center for Atmospheric Research
JUNE 11, 2019

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Outline

- Goal of MPAS-G portability project
- Refactoring strategy
- Porting and Optimization strategies (talk by Dr. Raghu Raj Prasanna Kumar)
- Results
- Future work

Goals of MPAS-GPU Portability project

- Achieve portability across Xeon, Xeon phi and GPU architectures without sacrificing CPU performance
- Minimize use of architecture-specific code:

```
#ifdef __GPU__
:
elseif __CPU__
:
#endif
```

- Manage porting/optimization costs: Use OpenACC to enable CPU-GPU portability
- Use all the hardware (CPU and GPU) available

MPAS refactoring strategy

Default time integration

```
Call Physics

Do dynamics_split_steps

      Do rk3_step = 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
```

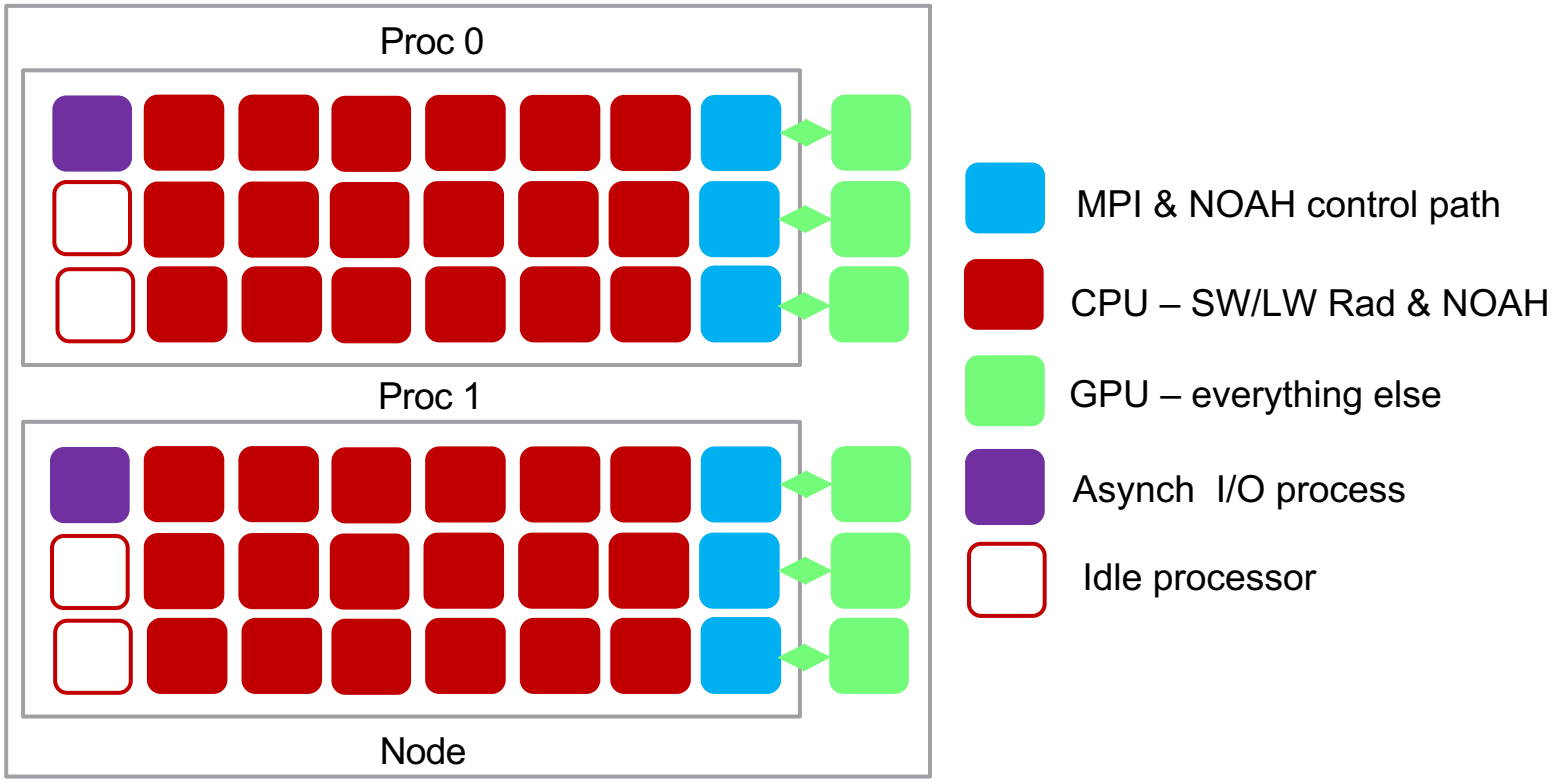
- **Dynamics Solver** ~10,000 SLOC
- **Physics** ~ 100,000 SLOC
  - Radiative Transport: ~37,000 SLOC
  - NOAH Land Surface Model: ~21,000 SLOC
  - Other physics code: ~42,000 SLOC
- **Time evenly split between dynamics and physics**

MPAS refactoring strategy

- **Use all the hardware (CPU & GPU) available**
- **CPU resident**
  - NOAH LSM is large, branchy and inexpensive -> CPU
  - RT is large, expensive but can run asynchronously -> CPU
  - I/O should be asynchronous -> CPU
- **GPU resident**
  - Dry/moist dynamics
  - All other physics

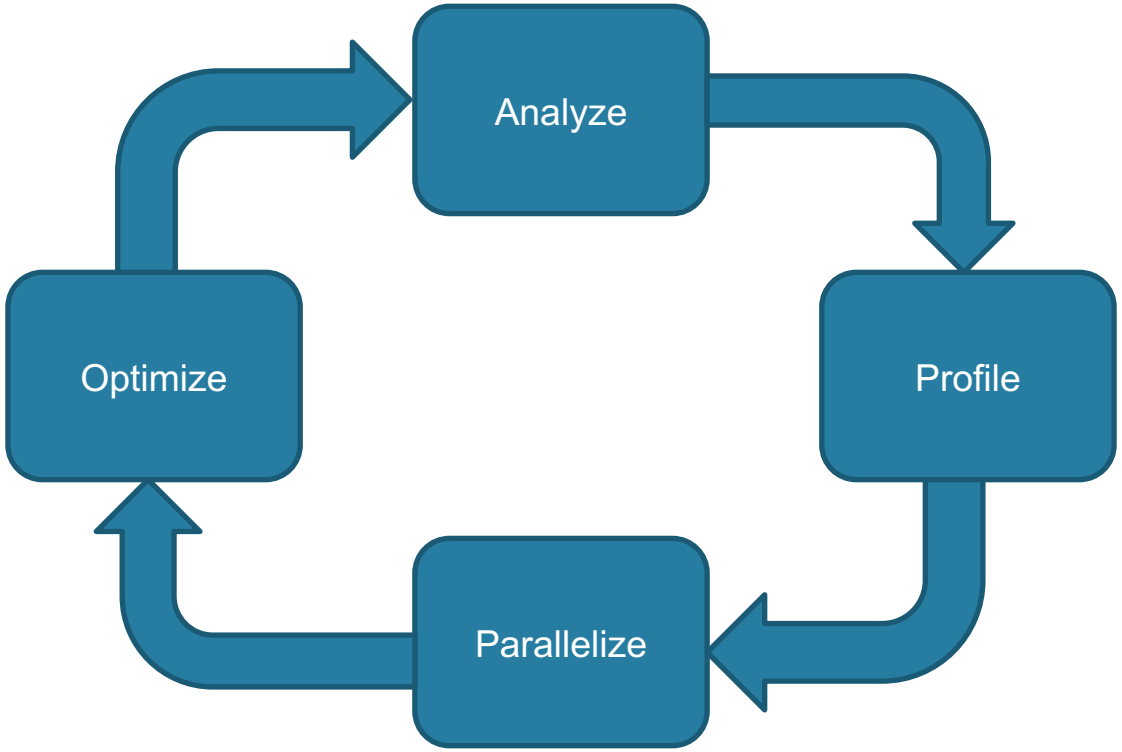
Part of our team: University of Wyoming students and NVIDIA/PGI.

MPAS-GPU Process Layout on IBM node



Porting onto GPU

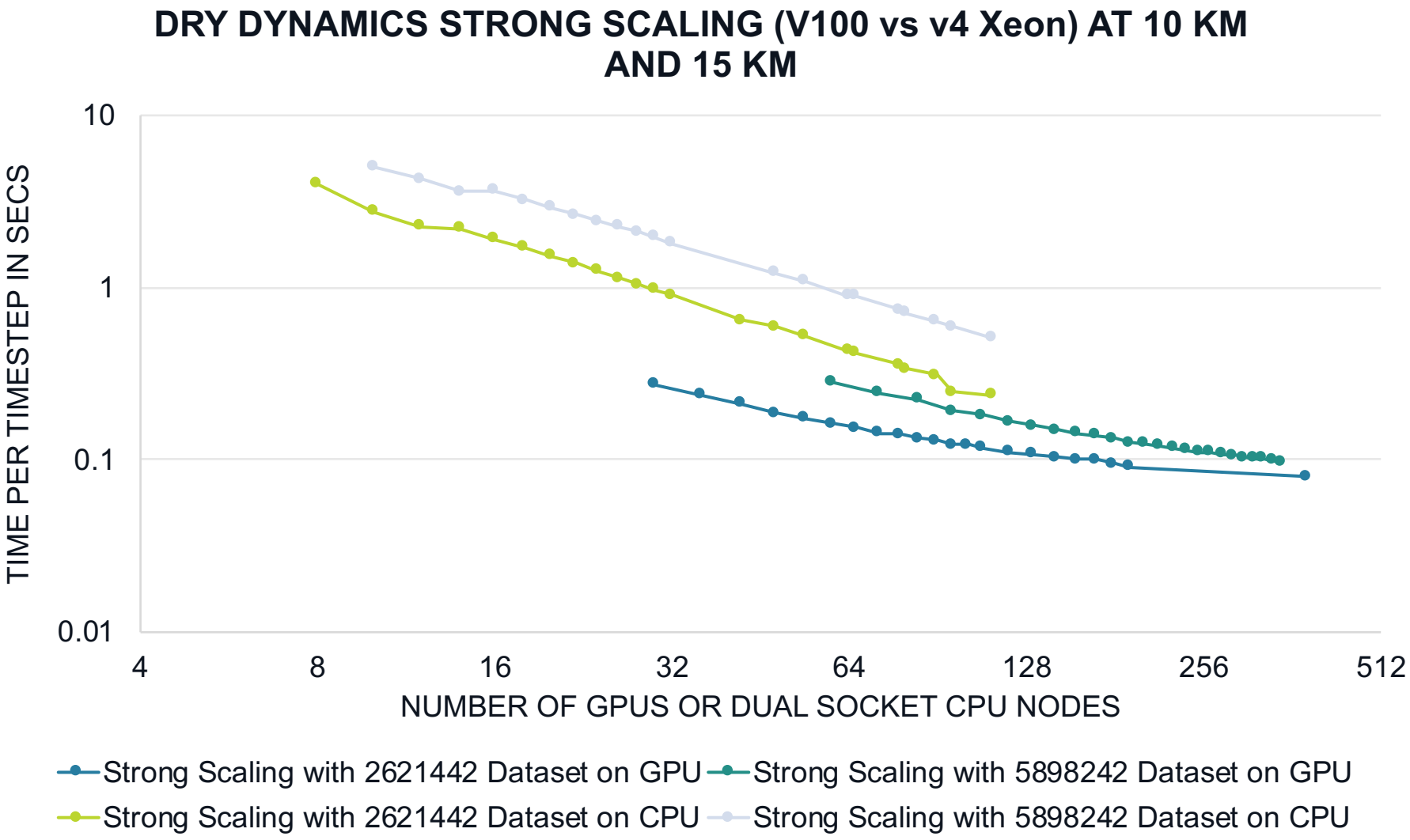
OpenACC development cycle



Strong scaling benchmark test setup

- **MPAS-A Version 6.x**
- **Test case: Dry dynamics (no physics)**
- **Compiler:** GPU **-** PGI 19.4, CPU - Intel 18
- **MPI:** GPU **-** IBM spectrum, CPU - Intel MPI
- **CPU:** 2 socket Broadwell node with 36 cores
- **GPU:** NVIDIA Volta V100
- **15, 10 km problem**
  - Timestep: 90, 60 sec
  - Horizontal points/rank: 2621442 points, 5898242 points(uniform grid)
  - Vertical: 56 levels

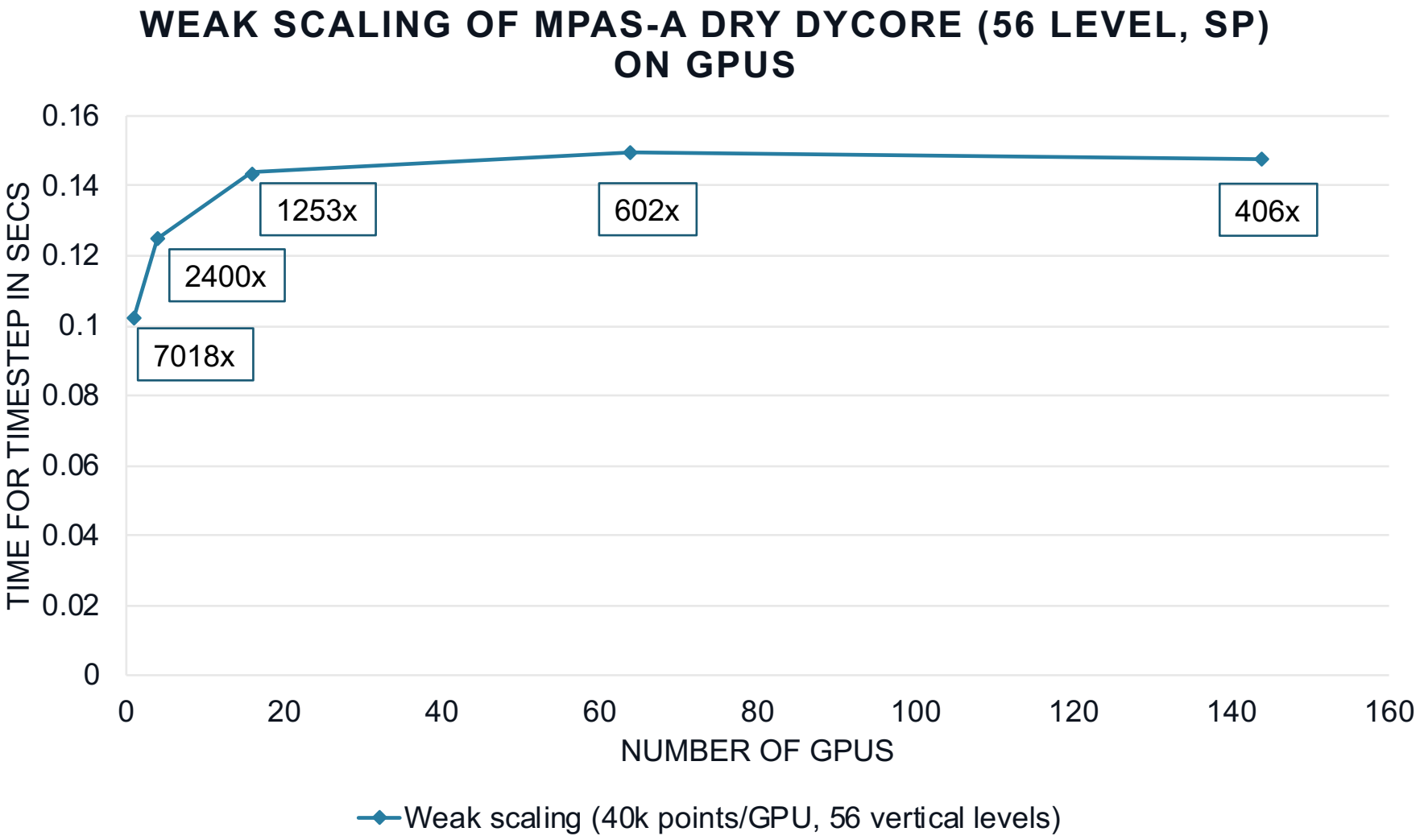
Strong scaling



Weak scaling benchmark test setup

- **MPAS-A Version 6.x**
- **Test case: Dry dynamics (no physics)**
- **Compiler:** GPU **-** PGI 19.4, CPU - Intel 18
- **MPI:** GPU **-** IBM spectrum, CPU - Intel MPI
- **CPU:** 2 socket Broadwell node with 36 cores
- **GPU:** NVIDIA Volta V100
- **120-60-30-15-10 km problem**
  - Timestep: 720, 300, 180, 90, 60 sec
  - Horizontal points/rank: 40,962 points (uniform grid)
  - Vertical: 56 levels

Weak scaling with 40k points per GPU

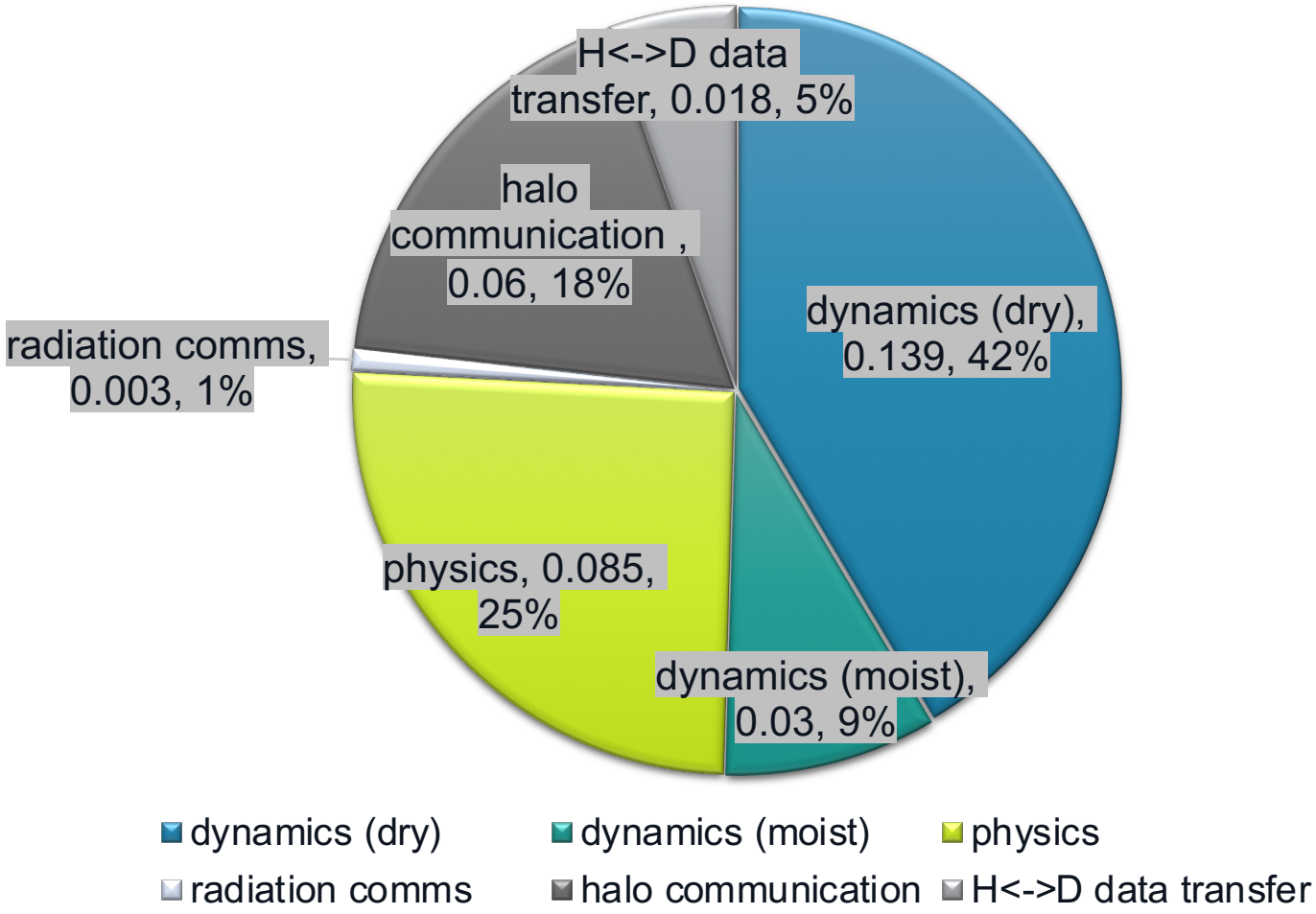


Future work

- Optimizing physics data transfers
- Porting Lagged radiation code to GPU
- Execute MPAS-A at a very high resolution 5km and 3km

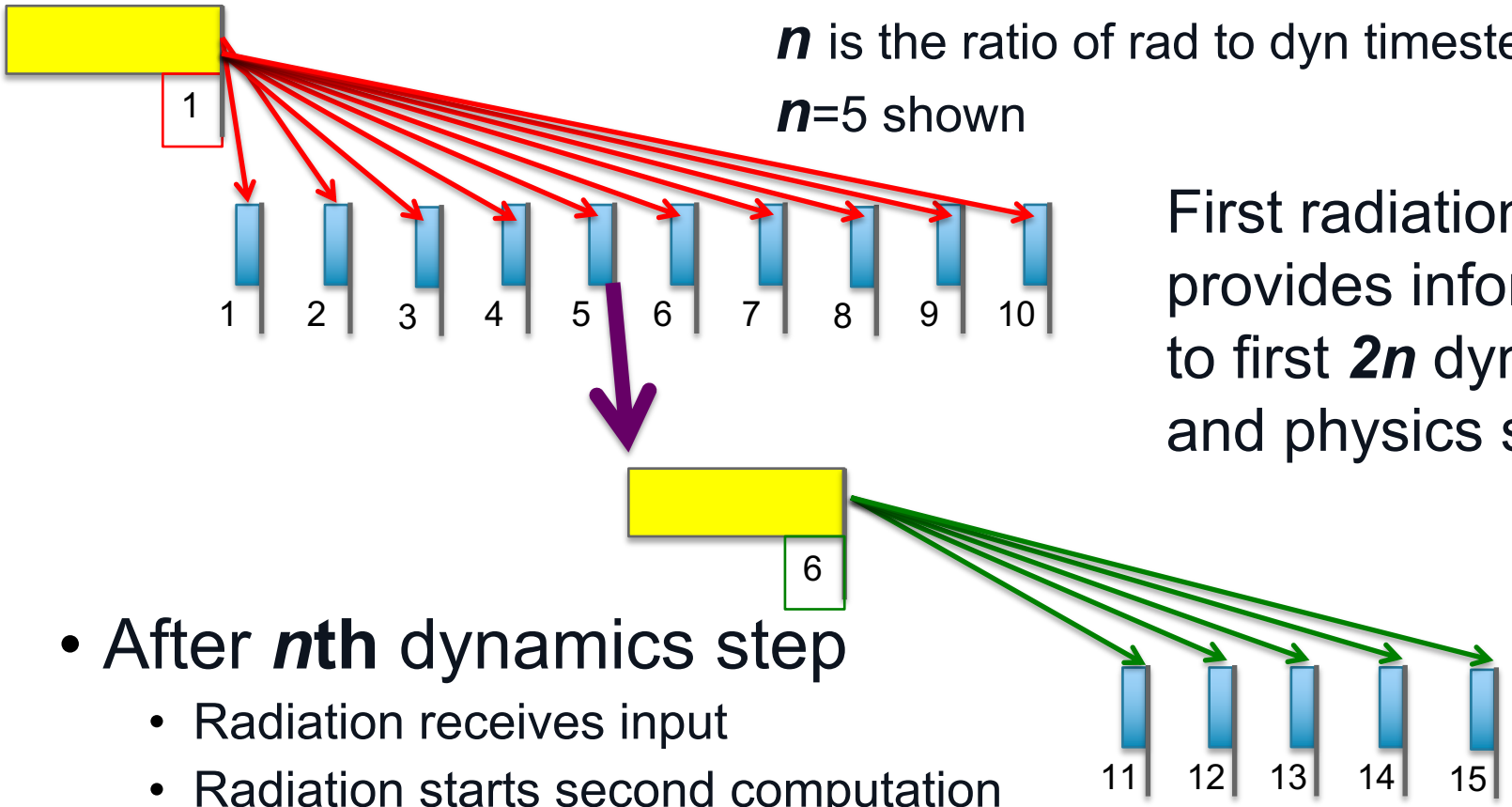
Projected full MPAS-A performance

MPAS-A estimated time budget for 40k points per GPU



Thank you
Questions??

Scheme to Overlap Radiation with Dynamics Solver Execution



n is the ratio of rad to dyn timesteps

n=5 shown

First radiation step provides information to first ***2n*** dynamics and physics steps

- After ***n*th** dynamics step
  - Radiation receives input
  - Radiation starts second computation
  - Dynamics keeps processing