

# **MPAS on GPUs Using OpenACC: Approach, Portability & Performance**

**Dr. Raghu Raj Kumar**

**Project Scientist I**

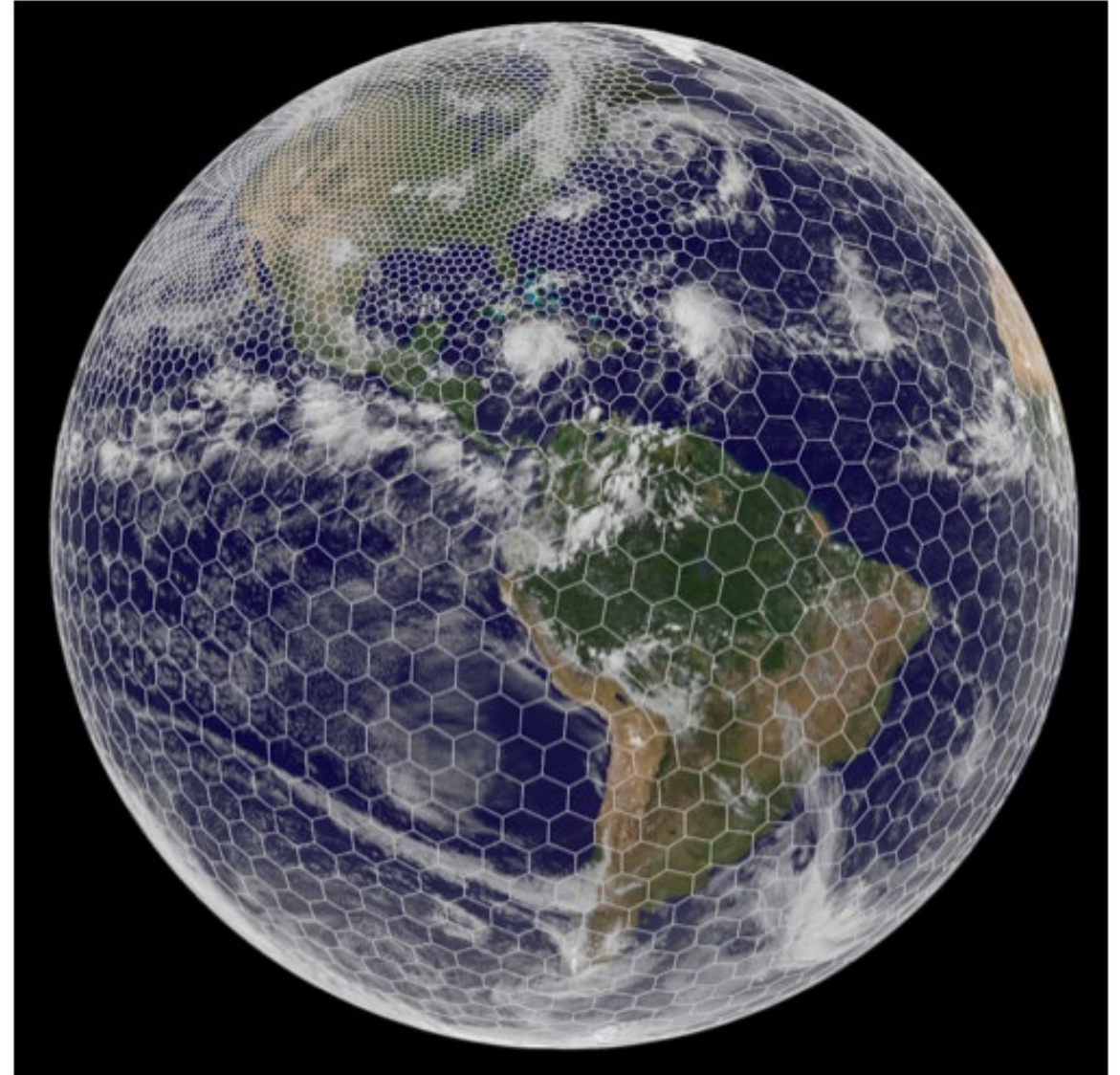
**Special Technical Projects (STP) Group**

**National Center for Atmospheric Research**



# Outline

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



# System Specs

- **San Diego Supercomputer Center**

- **COMET:** Dual socket Haswell (28 cores/node) with 4 P100/node, 36 nodes (only 8 accessible), intra-node PCIe connected, inter-node Infiniband

- **NCAR-Wyoming Supercomputing Center**

- **Cheyenne:** Dual socket Broadwell (36 cores/node), 4,032 nodes
- **Casper:** Two sets of nodes
  - Dual socket, Skylake (36 cores/node), 4 V100/node, 2 nodes, intra-node PCIe connect between Host-Device, NVLink connect between GPUs, EDR infiniband for internode
  - Dual socket, Skylake (36 cores/node), 8 V100/node, 2 nodes, intra-node PCIe connect between Host-Device, NVLink connect between GPUs, EDR infiniband for internode

# Software Spec: MPAS Dynamical Core

- **Software**
  - MPAS 6
  - Intel Compiler 17.0, PGI Compiler 17.10
- **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)
  - Number of levels = 56
  - Single precision (SP)
  - Simulation executed for 16 days, **performance shown for 1 timestep**

# Software Spec: MPAS

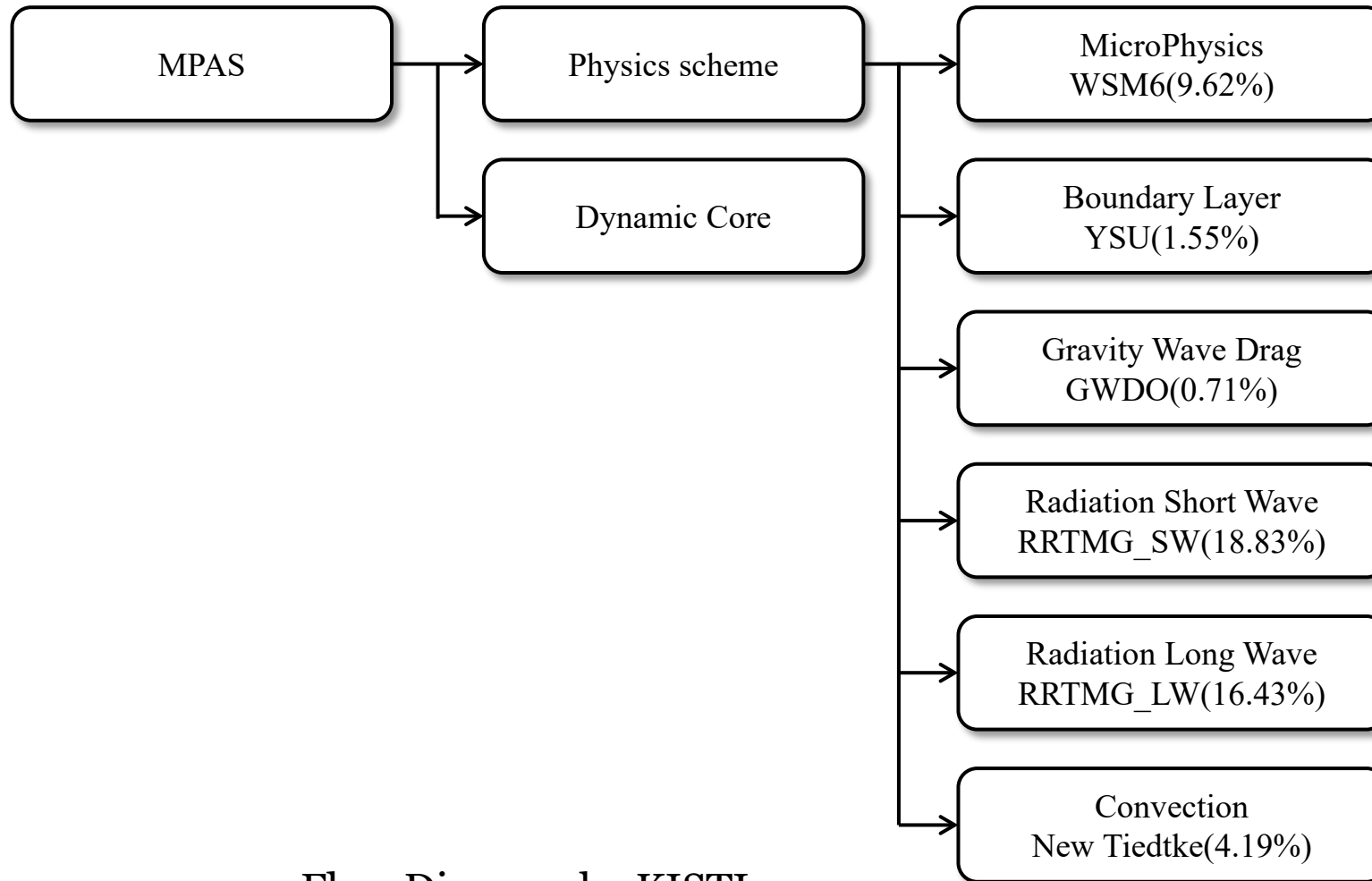
- **Software**
  - MPAS 6
  - Intel Compiler 17.0, PGI Compiler 17.10
- **Full physics suite**
  - Scale-aware Ntiedtke Convection, WSM 6 Microphysics, Noah Land surface, YSU Boundary Layer, Monin-Obhukov Surface layer, RRTMG radiation, Xu Randall Cloud Fraction
  - Radiation interval: 30 minutes
  - Single precision (SP)
  - Verification in progress, **performance shown for 1 timestep**

# Our Team of Developers

- **NCAR**
  - Dr. Raghu Raj Kumar, Project Scientist, STP
  - Supreeth Suresh, Software Engineer, STP
  - Clint Olsen, Student Assistant, STP
  - Dr. Michael Duda, Software Engineer, MMM
- **NVIDIA/PGI**
  - Dr. Carl Ponder, Senior Applications Engineer
  - Dr. Craig Tierney, Solutions Architect
  - Brent Leback, PGI Compiler Engineering Manager
- **University of Wyoming:**
  - GRAs: Pranay Kommera, Sumathi Lakshmiranganathan, Cena Miller, Bradley Riotto
  - Undergrads: Aisha Mohamed, Brett Gilman, Briley James, Suzanne Piver
- **Korean Institute of Science and Technology Information**
  - Jae Youp Kim, GRA
- **IBM/TWC**



# Where to begin?



## Execution time-

Physics: 45-50%

DyCore: 50-55%

## Lines of Code-

**Physics: 110,000**

**DyCore: 10,000**

Flow Diagram by KISTI

# Approach to Port Dycore: Simplest of all

```
do iCell=cellSolveStart,cellSolveEnd
  do i=1,nEdgesOnCell(iCell)
    iEdge = edgesOnCell(i,iCell)
    !DIR$ IVDEP
    do k = 2, nVertLevels
      flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
      w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
    end do
  end do
  !DIR$ IVDEP
  do k = 2, nVertLevels
    w_tend(k,iCell) = ( fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
  end do
end do
```

```
!$acc data present(w_tend, &
!$acc edgesoncell, edgesoncell_sign, fzm, fzp,nedgesoncell, u_tend, &
!$acc zb3_cell, zb_cell, zz)
!$acc parallel num_workers(8) vector_length(32)
!$acc loop gang worker private(iEdge, flux)
```

```
do iCell=cellSolveStart,cellSolveEnd
  do i=1,nEdgesOnCell(iCell)
    iEdge = edgesOnCell(i,iCell)
    !DIR$ IVDEP
    do k = 2, nVertLevels
      flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
      w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
    end do
  end do

  !DIR$ IVDEP
  do k = 2, nVertLevels
    w_tend(k,iCell) = ( fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
  end do
end do
!$acc end parallel
!$acc end data
```

```
!$acc data copy(w_tend, &
!$acc edgesoncell, edgesoncell_sign, fzm, fzp,nedgesoncell, u_tend, &
!$acc zb3_cell, zb_cell, zz)
!$acc kernel
```

```
do iCell=cellSolveStart,cellSolveEnd
  do i=1,nEdgesOnCell(iCell)
    iEdge = edgesOnCell(i,iCell)
    !DIR$ IVDEP
    do k = 2, nVertLevels
      flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
      w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
    end do
  end do

  !DIR$ IVDEP
  do k = 2, nVertLevels
    w_tend(k,iCell) = ( fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
  end do
end do
!$acc end kernel
!$acc end data
```



# Single Node Performance: MPAS Dycore

- **Timers**
  - MPAS GPTL timers reported in log files
- **GPU Timing : Has no updates from device to host**
  - Host updates maybe needed for printing values on screen
  - Host updates maybe needed for netcdf file output

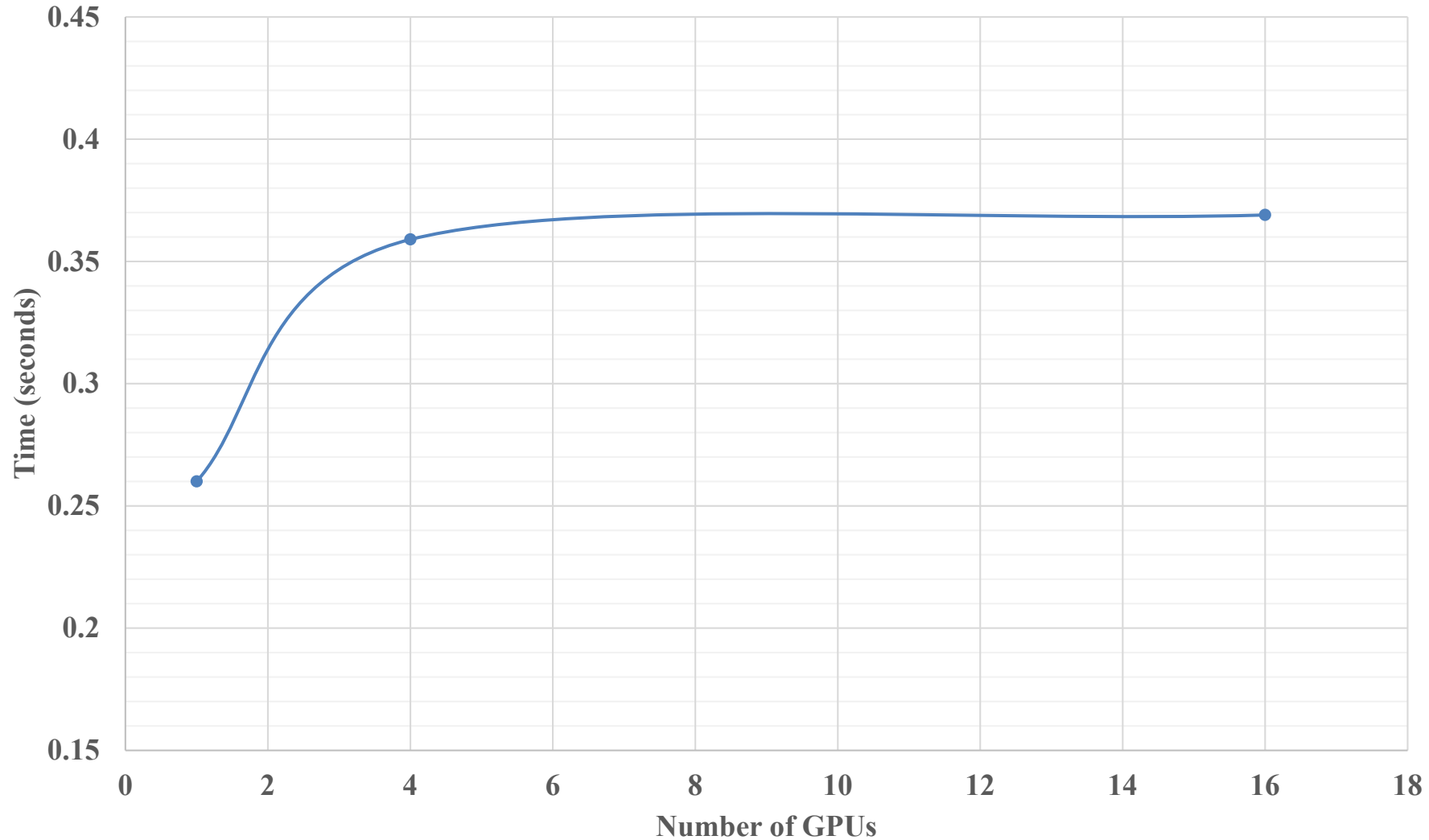
Dataset		Broadwell (Fully Subscribed, OpenMP Enabled, Intel compiled, Base code)	P100 with Haswell(1 GPU, PGI compiled, OpenACC code)	V100 with Skylake (1 GPU, PGI compiled, OpenACC code)	Speedup Broadwell vs P100	Speedup Broadwell vs V100	Speedup P100 vs V100
120 Km (40K)	SP	0.46	0.25	0.16	1.84	2.95	1.60
	DP	0.94	0.42	0.23	2.24	4.09	1.83

**Broadwell Node vs one GPU device**

# 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
    - Waiting on right GPU configurations in a node + right PGI compiler + right OpenMPI/MVAPICH combo + right cluster to get benchmarks
  - Performance for 16 GPU runs on COMET are provided

# Weak Scaling for MPAS Dry Dycore (SP & DP) on P100 GPU



**Time per timestep, 4 GPUs per node, 1 MPI rank per GPU, Max of 4 MPI ranks per node, Intranode Affinity for MPI ranks, Uses OpenMPI, PCIe no NVLink, PGI 17.10**

# Up Next after halo exchange- Physics

- **Approach**

- Dealing with volume of code

- Use tools (PGI or KGen)

```
2346, Generating data copyin(rho_zz(:, :), rtheta_pp(:, :), rtheta_pp_old(:, :), ru_p(:, :), rdzw(:), rw(:, :), w(:, :), zz(:, :), zxu(:, :), alpha_tri(:, :), cofwz(:, :), cqu(:, :), edgesoncell_sign(:, :), invareacell(:), tend_rt(:, :), wwavg(:, :), tend_rho(:, :), rw_save(:, :), a_tri(:, :), cellsonedge(:, :), cofrz(:), coftz(:, :), cofwr(:, :), cofwt(:, :), dcedge(:), dss(:, :), dvedge(:), edgesoncell(:, :), exner(:, :), fzm(:), fzp(:), gamma_tri(:, :), invdcedge(:), nedgesoncell(:), rho_pp(:, :), ruavg(:, :), rw_p(:, :), tend_ru(:, :), tend_rw(:, :), theta_m(:, :))
```

- Generate a list of variables

- Use Kernels directive port

- Thanks to KISTI for the rapid port

- Round 2: Code refactoring & Optimization

- Only low hanging fruit

- Monin Obhukov is 7000+ lines takes less than 1% execution time

- Verify & Integrate

# Single Node Performance: MPAS Physics

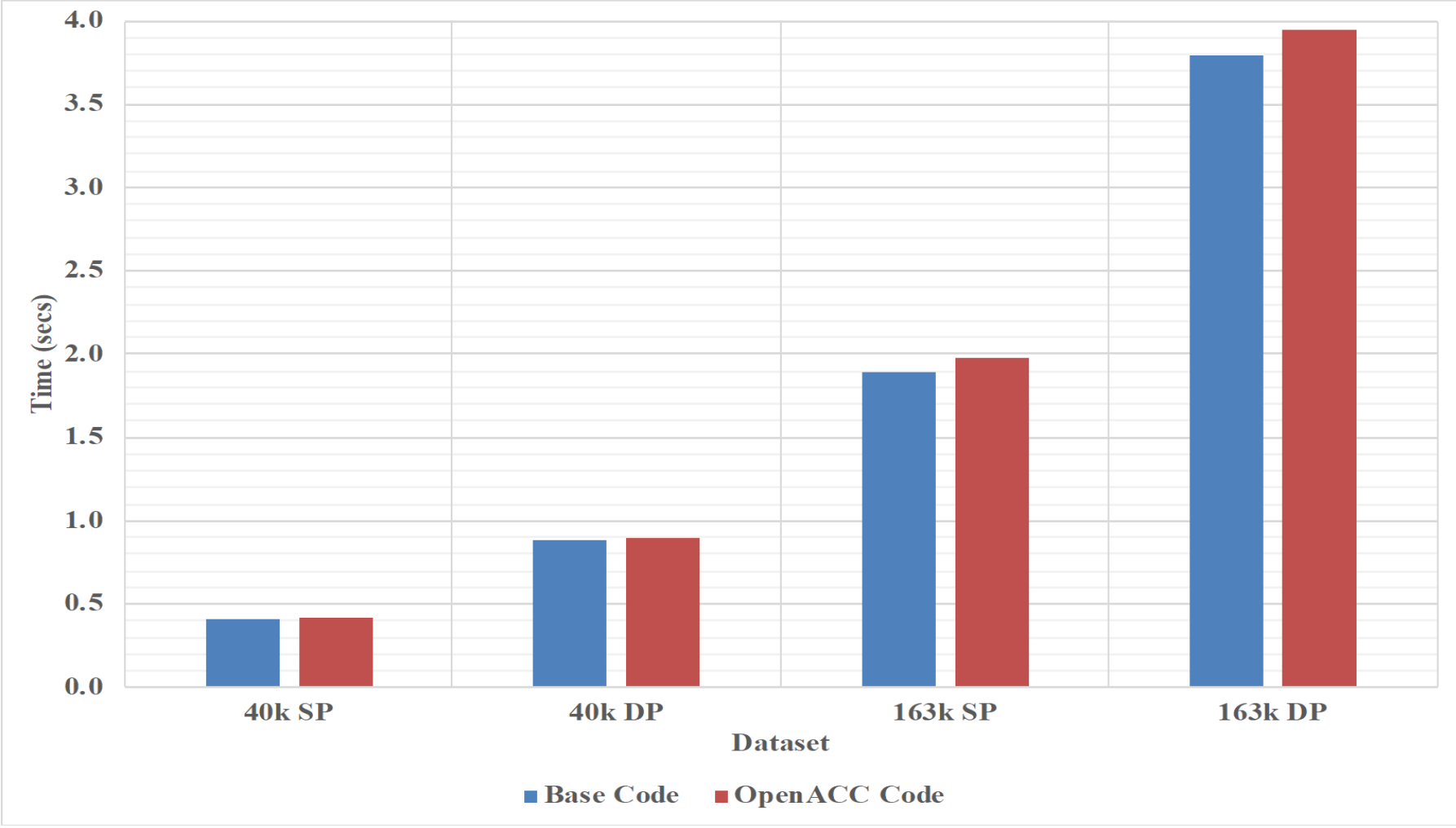
- **Timers**
  - MPAS GPTL timers reported in log files
- **GPU Timing : Excludes Host-Device-Host data transfer time**
  - Physics is yet to be verified, hence data copy directives are not removed

	Single node, Integrated Code Time step Time (seconds)			
Physics Module	Broadwell	P100	V100	Broadwell vs V100
WSM6	0.255	0.096	0.080	3.2
YSU	0.013	0.008	0.007	2.0
Monin Obukhov	0.001	0.001	0.001	0.9



Where do we stand regarding *Performance* Portability?

# Portability: Performance Comparison of Original Dycore code with Refactored (OpenACC) code on Fully Subscribed Broadwell single node




**For datasets up to 40k per node, the execution time is identical. For 40k & 163k, the variation is <1% & <4% respectively.**

# Future Work

- **MPAS Performance**
  - Benchmarking on MVAPICH –GDR, NVLink systems
  - GPU friendly Tridiagonal solver implementation
    - “Fast Tridiagonal Solvers on the GPU” by Cohen, Zhang, and Owens
    - Expecting 5%-10%
- **Physics**
  - Verification and Integration
  - Integrating Lagged Radiation





Thank you! Questions?

# Approach

