NCAR UCAR

air • planet • people

Combining Machine Learning and Numerical Modeling to Transform Atmospheric Science

Dr. Richard Loft*
Director, Technology Development
Computational and Information Systems Laboratory
National Center for Atmospheric Research

***with special thanks to Dr. Raghu Kumar, NVIDIA; Supreeth Suresh, NCAR; the PGI team; and students and faculty at the University of Wyoming**

**GTC San Jose, CA
March 19, 2018**

Talk Summary

- **Science 3.0: HPC + ML**
  - Apply GPUs to accelerate models where physics is rigorous.
  - Replace parameterizations with Machine Learning emulators where the physics is phenomenological.

- Initial results are encouraging…

- But much more work needs to be done to prove these ideas out!

What's driving future of prediction? ESP!

- **Then:**
  - Weather prediction(5-10 days)
  - **GAP**
  - Climate projections (decades-centuries)
- **Divisions between meteorology and climate are breaking down!**
  - Discoveries of predictability driven by the ocean and land surface
- **Now: Earth System Prediction (ESP) filling that GAP**
  - Sub-seasonal (Weeks)
  - Seasonal (Months)
  - Climate predictions (years to decades)
- **Making these predictions will require significantly more computing power.**

Earth System Modeling Catch 22

- Due to insufficient computing power ESMs can’t resolve key phenomena.

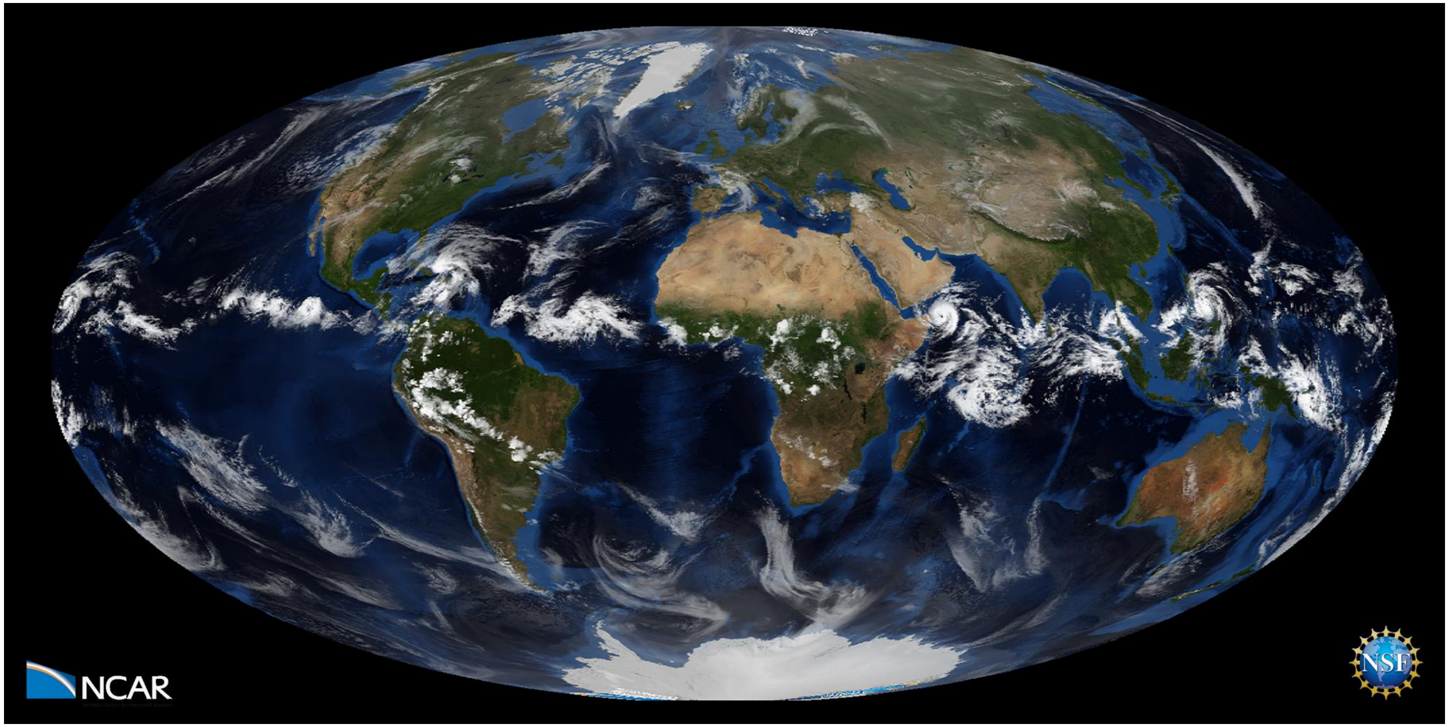
- Scientists try to describe the unresolved scales using human-crafted physics parameterizations.

- ESM’s *software complexity* grows, driven by the increasing complexity of these parameterizations.

- Growing *architectural complexity* hinders the ability to port and optimize ESM codes on new architectures.

- Due to insufficient computing power ESMs can’t resolve key phenomena.

Model for Prediction Across Scales - Atmosphere (MPAS-A)
A Global Meteorological Model & Future ESP Component

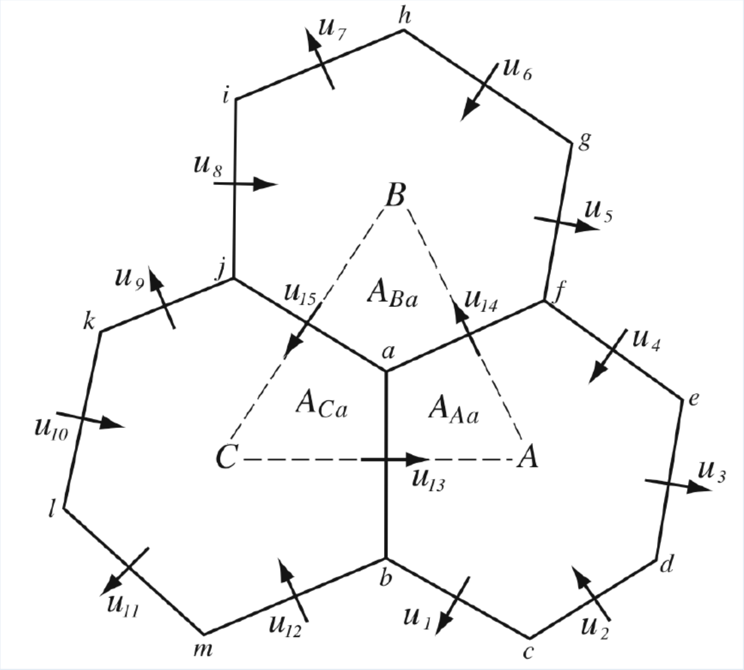


Simulation of 2012 Tropical Cyclones at 4 km resolution – Courtesy of Falko Judt, NCAR

MPAS: the algorithmic description

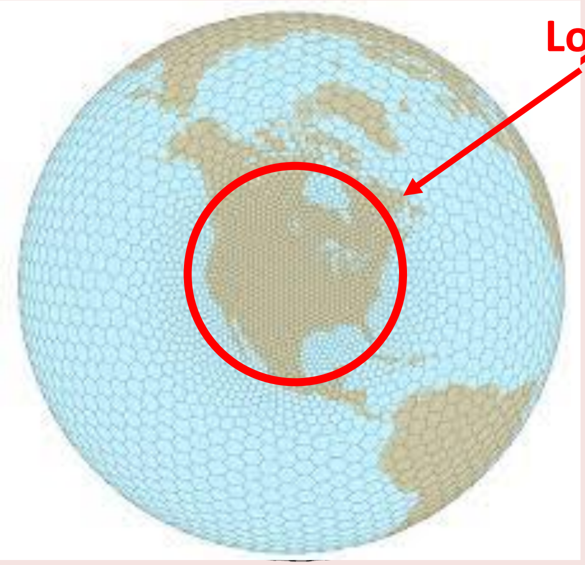
- **Fully compressible non-hydrostatic equations written in flux form**
- **Finite Volume Method on staggered grid**
  - The horizontal momentum normal to the cell edge (u) is sits at the **cell edges**.
  - Scalars sit at the **cell centers**
- **Split-Explicit timestepping scheme**
  - Time integration 3rd order Runge-Kutta
  - Fast horizontal waves are sub-cycled

MPAS is based on unstructured centroidal Voronoi (hexagonal) meshes using C-grid staggering and selective grid refinement.



MPAS Grids…

Horizontal



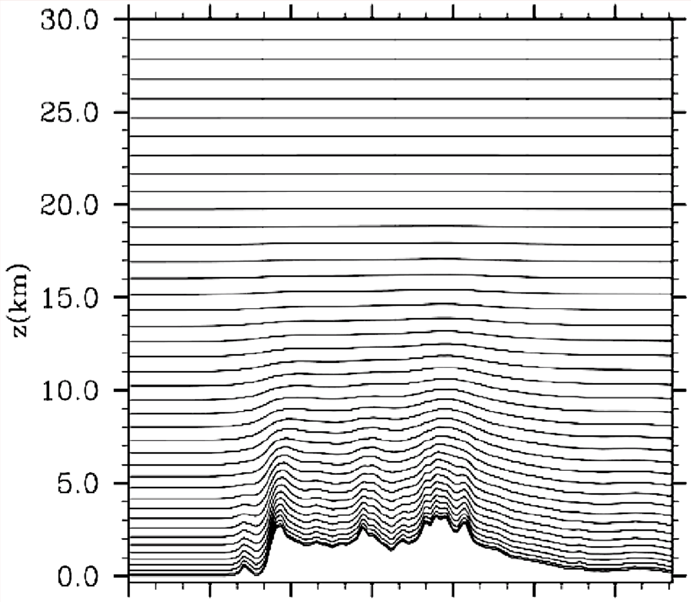
Sneaky pentagons

Local Refinement

MPAS
Unstructured Voronoi
(hexagonal) grid

- Good scaling on massively parallel computers
- No pole problems

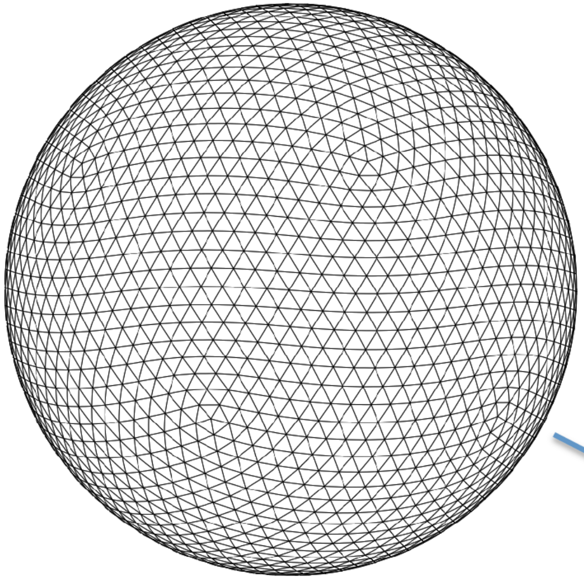
Vertical



MPAS
Height-based hybrid smoothed
terrain-following vertical coordinate

- Improved numerical accuracy

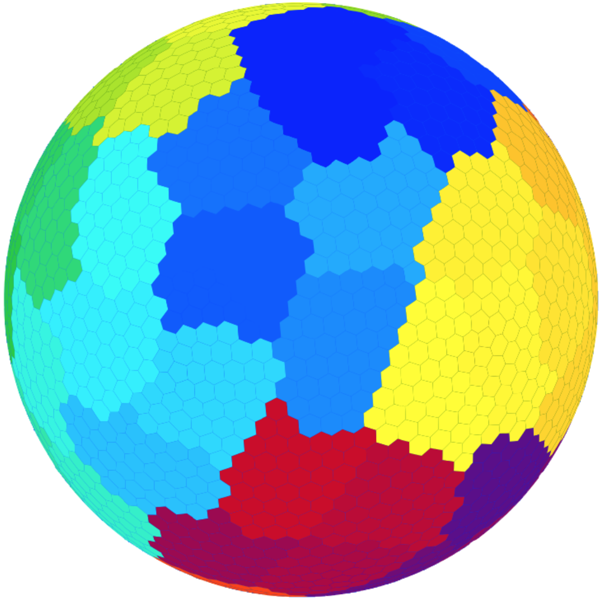
Parallel Decomposition via Metis



The *dual* mesh of a Voronoi tessellation is a Delaunay triangulation – essentially the connectivity graph of the cells

Parallel decomposition of an MPAS mesh then becomes a graph partitioning problem*: **equally distribute nodes among partitions (give each process equal work) while minimizing the edge cut (minimizing parallel communication)***

Graph partitioning



We use the Metis package for parallel graph decomposition

- Currently done as a pre-processing step, but could be done “on-line”

Metis also handles weighted graph partitioning

- Given *a priori* estimates for the computational costs of each grid cell, we can better balance the load among processes

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 ~350 halo exchanges /timestep!

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

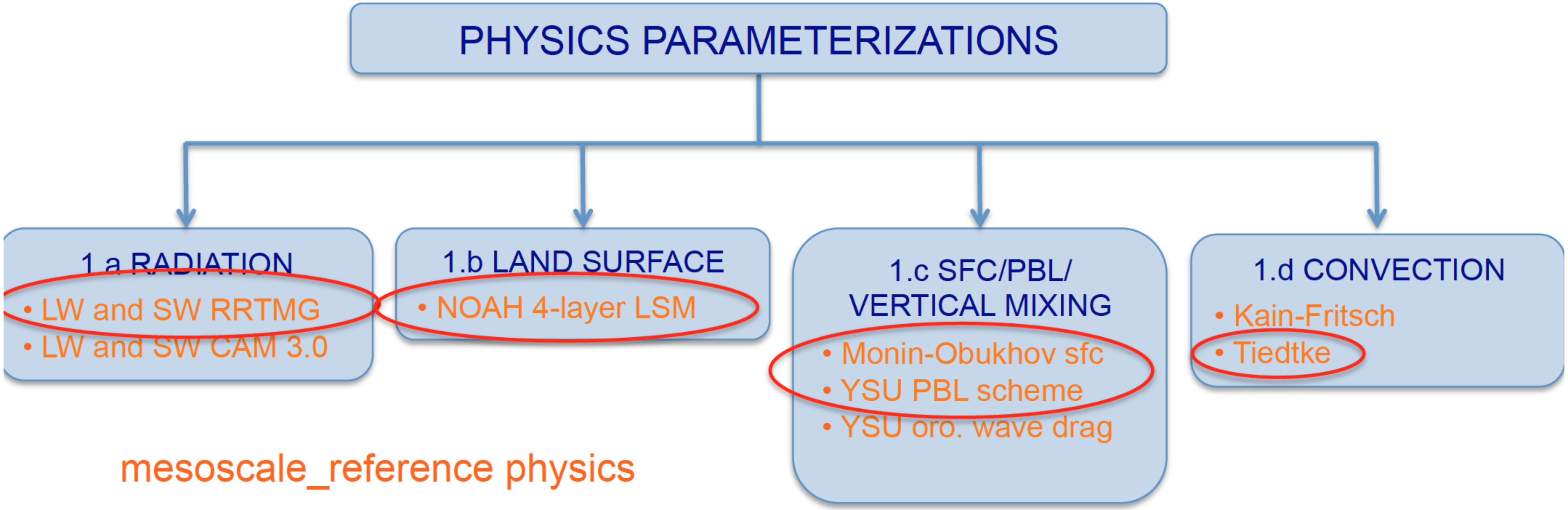
We can use any FV scheme here (we are not tied to RK3)

Scalar transport and physics are the expensive pieces in most applications.

Physics (Called before dynamics)

MPAS Timestep: 1. PHYSICS CALLED BEFORE DYNAMICAL CORE

- The physics parameterizations are called *sequentially*, i.e. they all use the same input surface and atmospheric conditions.
- They update the surface and top-of-the-atmosphere energy budgets.
- They calculates physics tendencies of temperature, momentum, water vapor, and water species (cloud water, cloud ice, rain, snow, graupel) for the dynamical core.



PHYSICS PARAMETERIZATIONS

1.a RADIATION
- [✓ LW and SW RRTMG]
- LW and SW CAM 3.0

1.b LAND SURFACE
- [✓ NOAH 4-layer LSM]

1.c SFC/PBL/ VERTICAL MIXING
- [✓ Monin-Obukhov sfc]
- [✓ YSU PBL scheme]
- YSU oro. wave drag

1.d CONVECTION
- Kain-Fritsch
- [✓ Tiedtke]

mesoscale_reference physics

Microphysics (called after dynamics)

MPAS Timestep: 1. PHYSICS CALLED BEFORE DYNAMICAL CORE

MPAS Timestep: 2. DYNAMICS and SCALAR TRANSPORT

MPS Timestep: 3. CLOUD MICROPHYSICS AFTER DYNAMICAL CORE

CLOUD MICROPHYSICS

WSM6 ($T,q_v,q_c,q_r,q_i,q_s,q_g$)

Kessler(T,q_v,q_c,q_r)

- Unlike the other physics parameterization schemes, the cloud microphysics scheme directly updates the temperature, water vapor, and other water/ice species instead of calculating cloud microphysics tendencies.
- Calling the cloud microphysics parameterization after the dynamics integration ensures that the relative humidity does not exceed 100% at the end of the model time-step.

MPAS: The Code inventory

MPAS Component	SLOC	Where it runs
Dynamics	10,000	GPU
Radiative Transport	37,000	CPU
Land Surface Model	21,000	CPU
Other physics	42,000	GPU
Total	110,000	

Goals of MPAS-GPU Portability Project

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

```
#ifdef _GPU_
   :
#endif
```

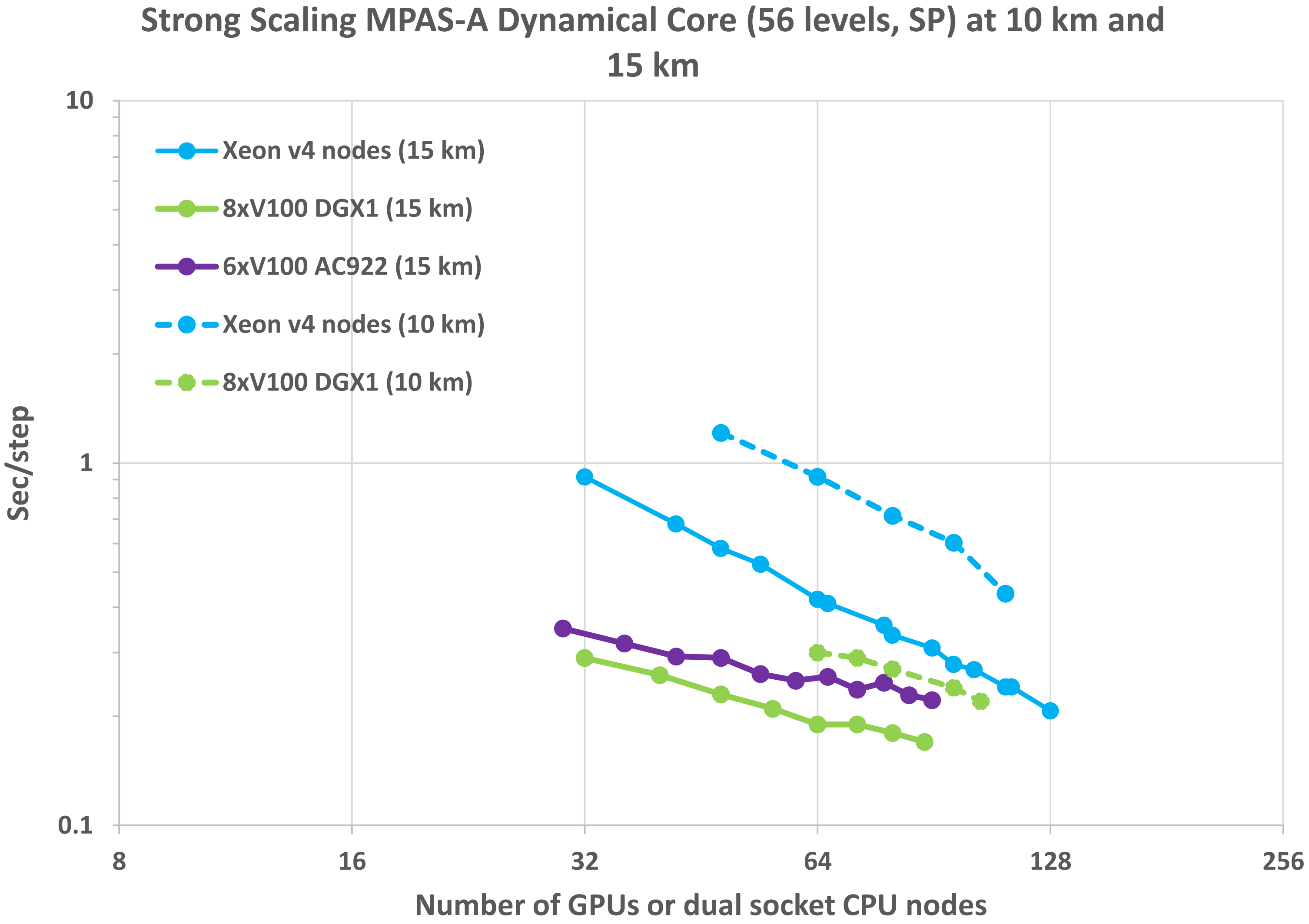
- Manage porting/optimization costs
  - Use OpenACC to enable CPU-GPU portability
- Use all the hardware (CPU & GPU) available
  - After all we paid for it!

Part of our team: UW students and PGI experts.

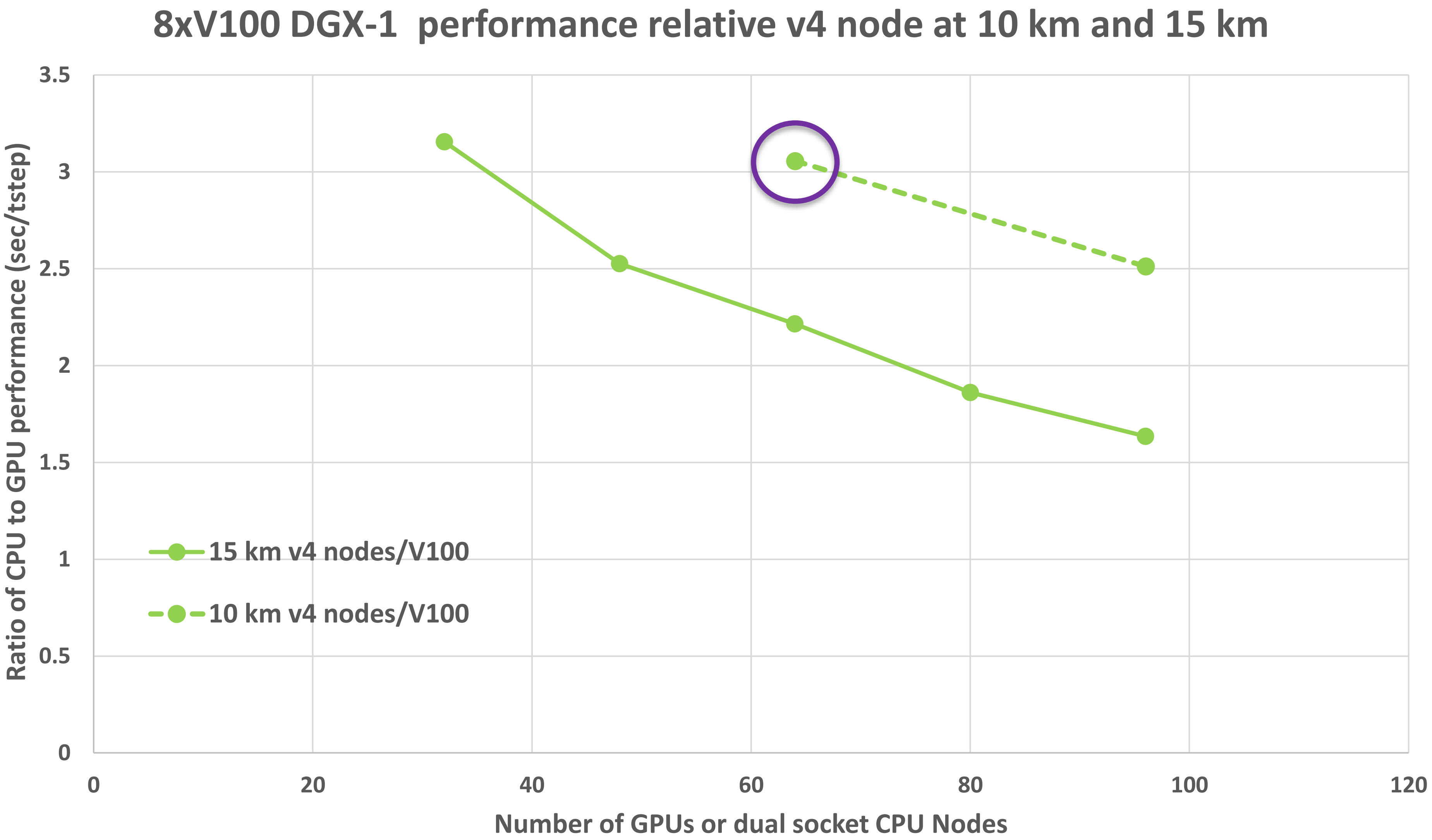
Scaling Benchmark Test Systems

- **Test case: MPAS-A dry dynamical core**
- **System 1: IBM "*WSC*" supercomputer**
  - AC922 node with 6, 16 GB V100 GPUs;
  - 2x 22-core IBM Power-9 CPUs;
  - Compiler: PGI 18.10
  - 2x IB interconnect; IBM Spectrum MPI
- **System 2: NVIDIA "Prometheus" supercomputer**
  - DGX-1 node with 8, 16 GB V100 GPUs;
  - 2x 18-core Intel Xeon v4 (BWL) CPUs;
  - Compiler: PGI 18.10
  - 4x IB interconnect; OpenMPI 3.1.3
- **System 3: NCAR Cheyenne supercomputer**
  - 2x 18-core Intel Xeon v4 (BWL)
  - Intel compiler 17.0.1
  - 1x EDR IB interconnect; HPE MPT 2.16 MPI

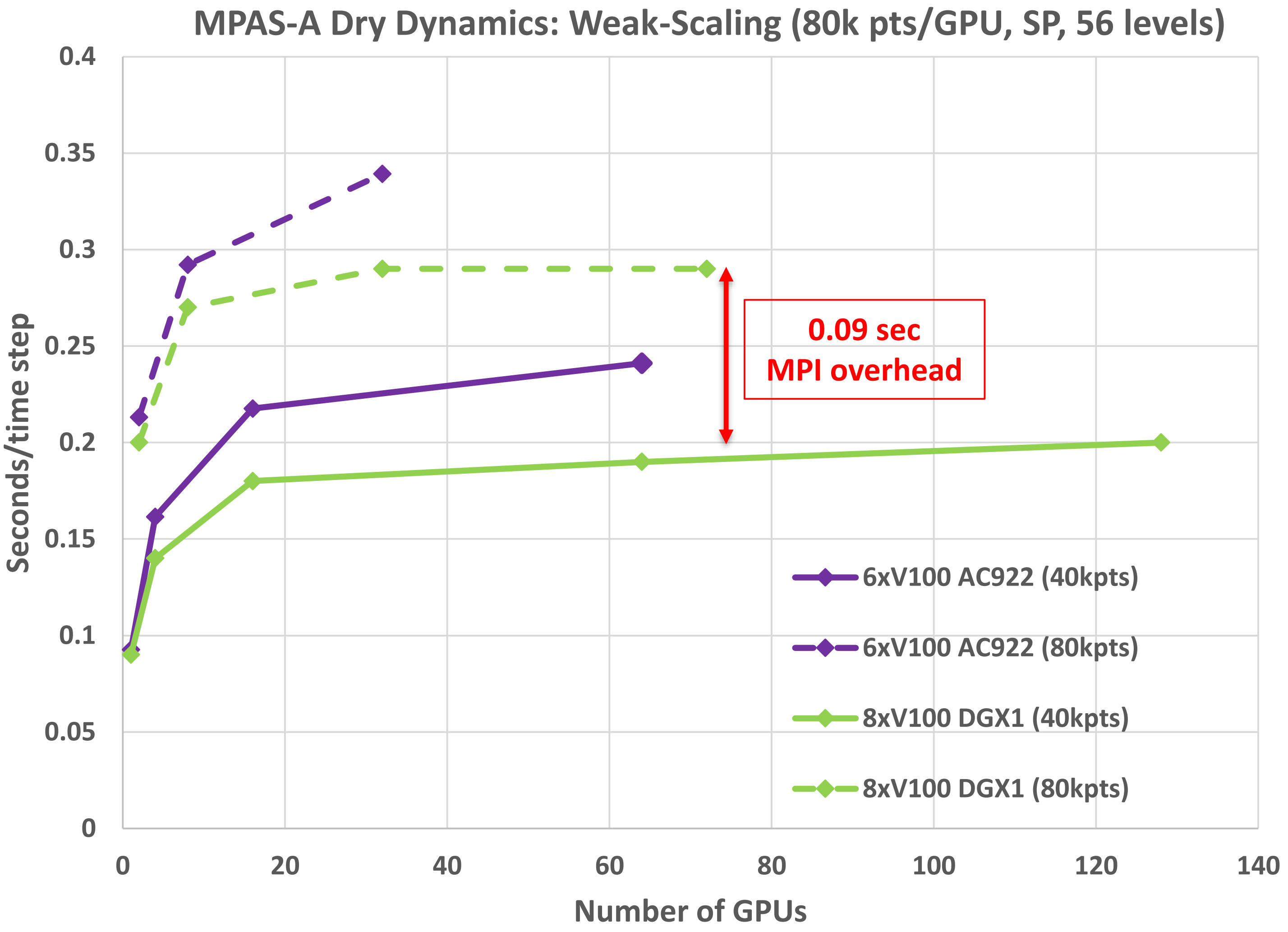
Strong Scaling V100 vs v4 Xeon at 10 km and 15 km



GPU speed relative to dual socket Intel Xeon v4 nodes



Weak scaling of MPAS-A dry dycore (56 level, SP) on GPUs

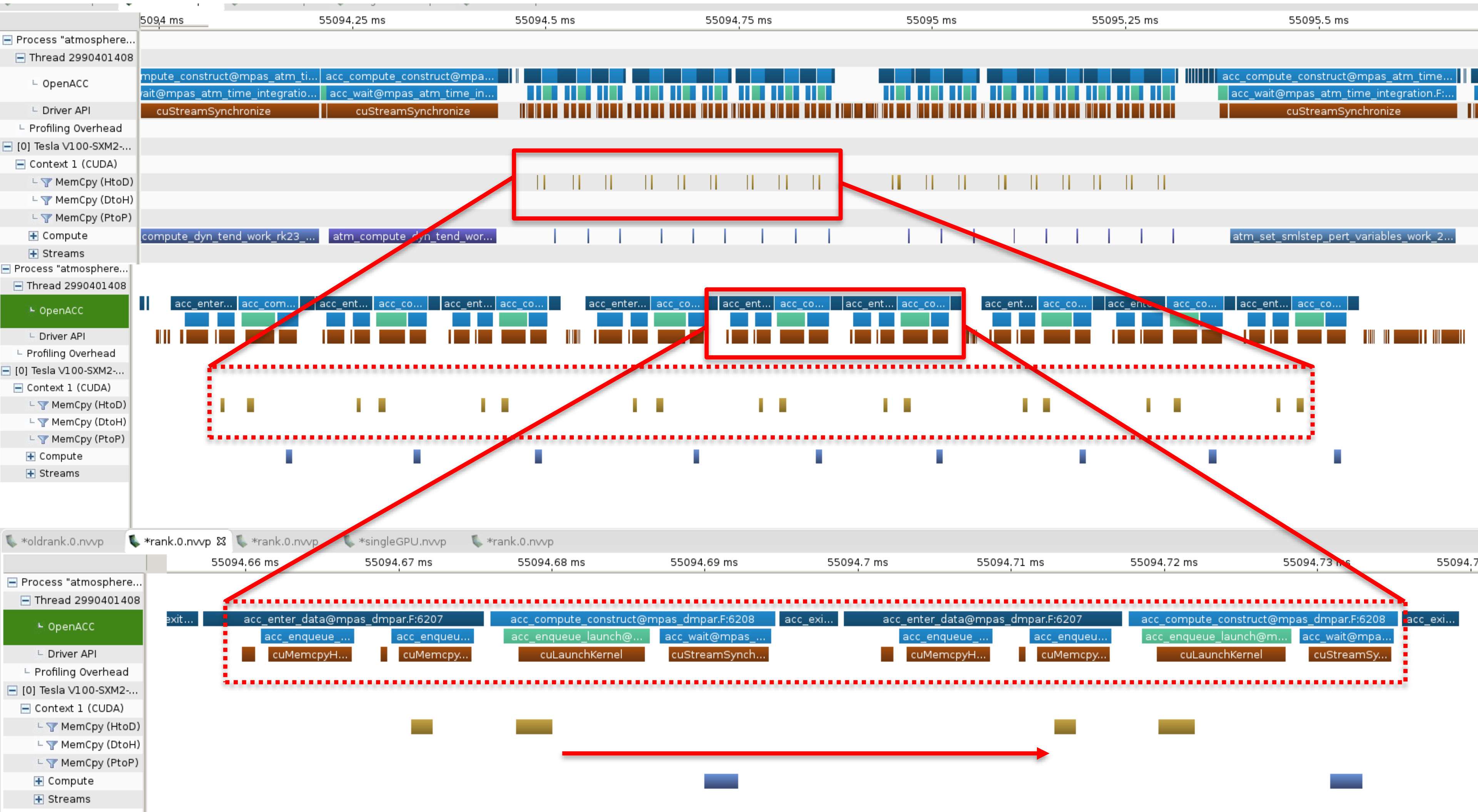


0.09 sec
MPI overhead

Optimizing MPAS-A dynamical core: Lessons Learned

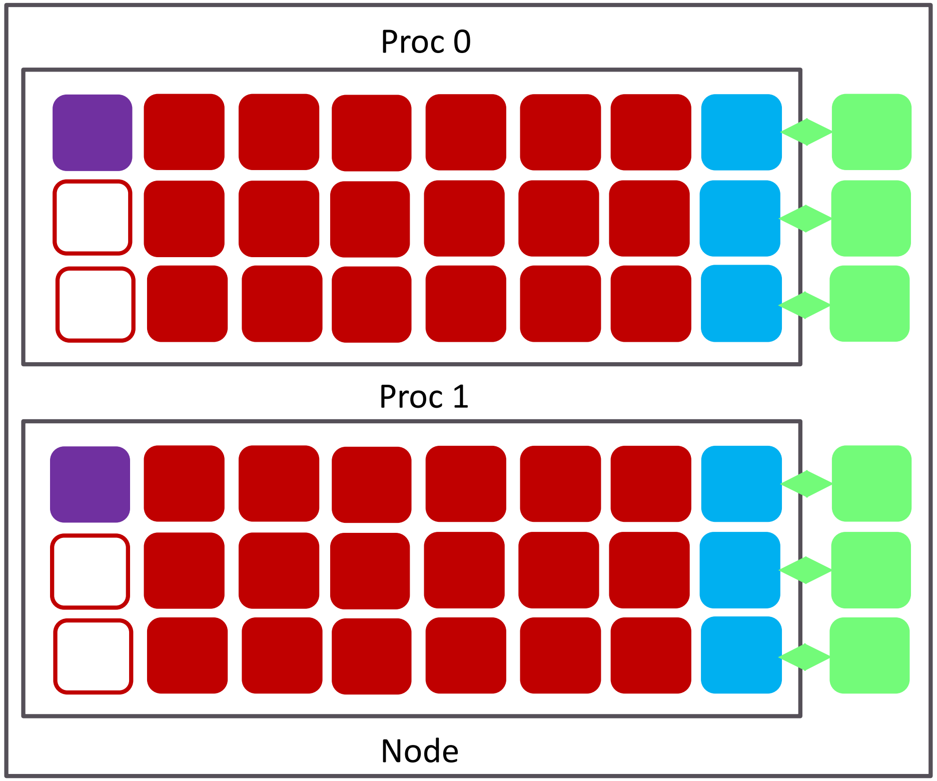
- Module level allocatable variables (20 in number) were unnecessarily being copied by compiler from host to device to initialize them with zeroes. Moved the initialization to GPUs.
- dyn_tend: eliminated dynamic allocation and deallocation of variables that introduced H<->D data copies. It's now statically 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.

Improving MPAS-A halo exchange performance: coalescing kernels



Coalescing these 9 kernels should drop MPI overhead by 50%

Overlapping Radiation Calculation: Process Layout (Example)



🟦 MPI & NOAH control path

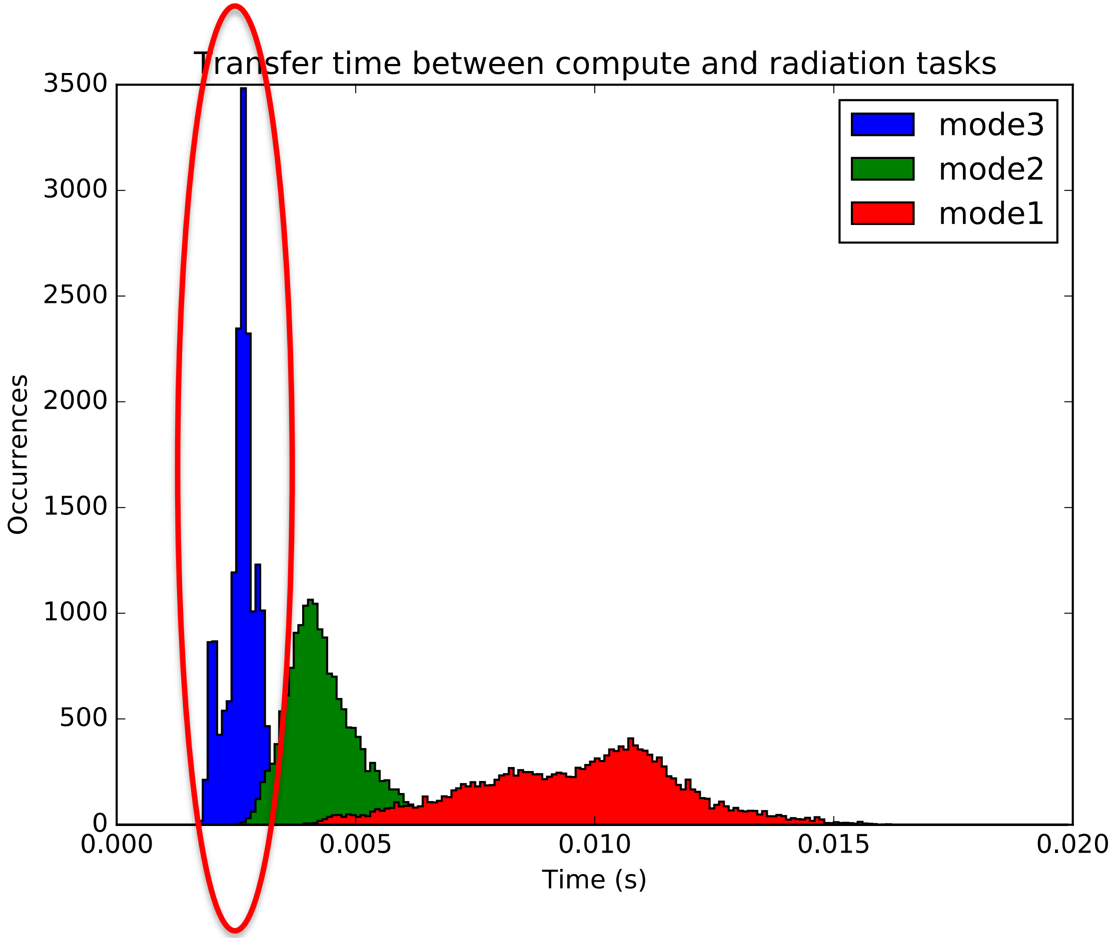
🟥 CPU – SW/LW Rad & NOAH

🟩 GPU – everything else

🟪 Asynch I/O process

□ Idle processor

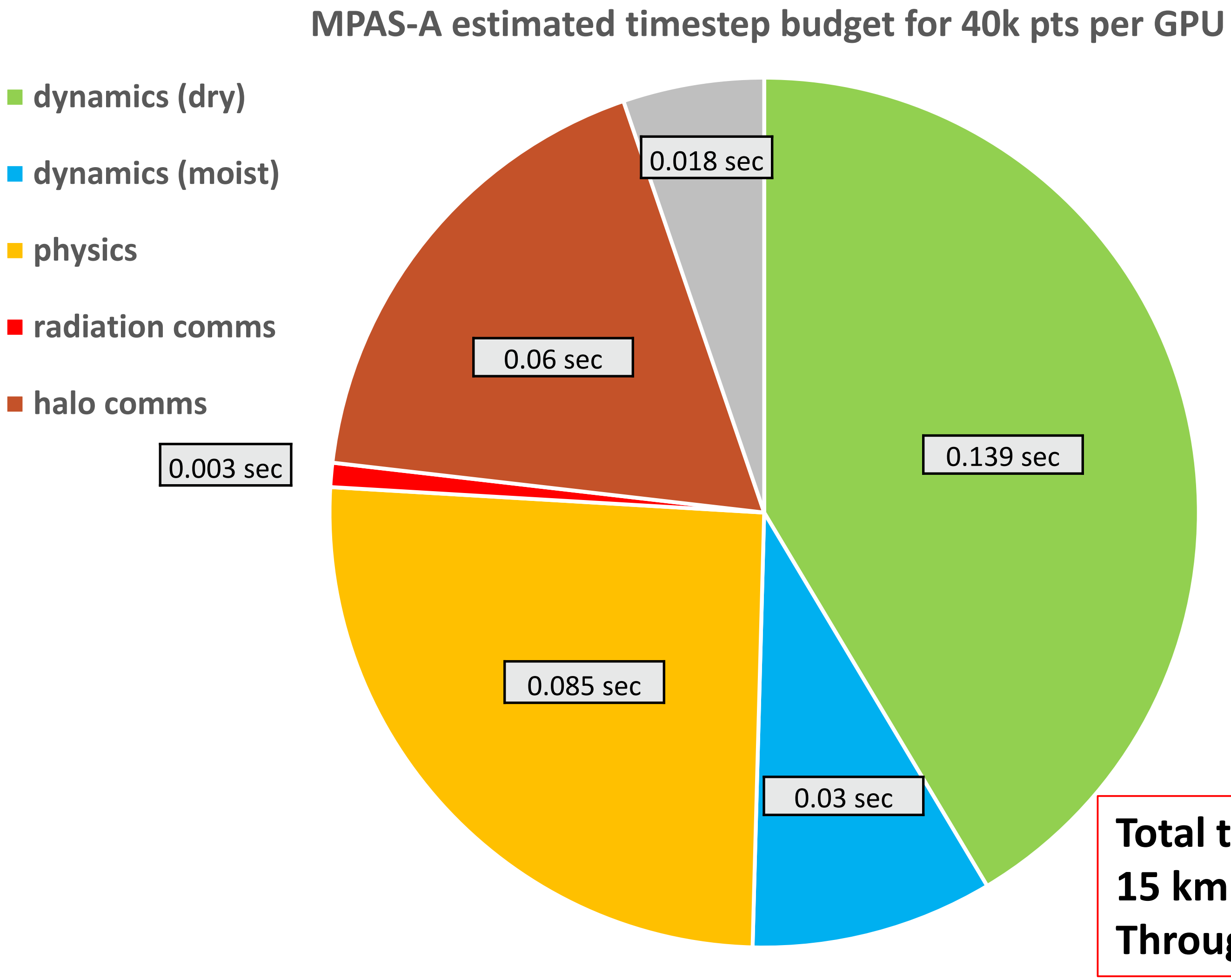
Co-locating radiation and integration tasks



Distribution of times to transfer general physics input fields from integration to radiation tasks for the 60-km uniform mesh on Cheyenne.

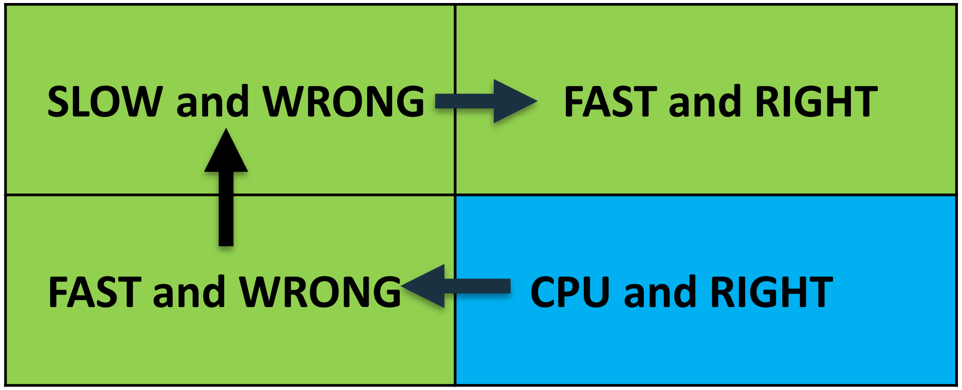
576 total tasks (16 nodes x 36 cores)
352 integration tasks
224 radiation tasks

Projected full MPAS-A model performance



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

Debugging MPAS-A: Tools



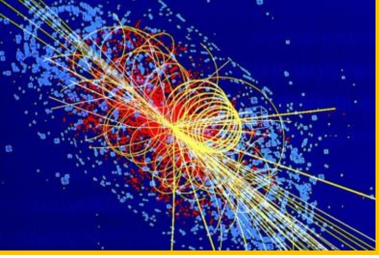
SLOW and WRONG

FAST and RIGHT

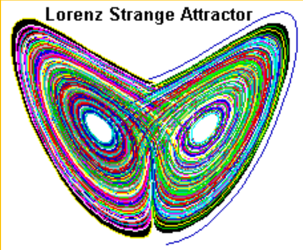
FAST and WRONG

CPU and RIGHT

- **PCAST:**
  - When do results first begin to differ between CPU and GPU?
- **MPAS Validation Tool**
  - When is different still right?



Debugging MPAS-A: PCAST



- **PGI Compiler Assisted Software Testing (PCAST)**
- Helps test for program correctness, and determine points of divergence.
- New in PGI 19.1 Compilers!
- Tells when CPU and GPU results diverge.
- There are three ways to invoke PCAST:
  - With the **autocompare** compiler flag
  - Through the **pgi_compare** run-time call
  - Through the **acc_compare run-time call**

```
PCAST sfclay1d:1008 Float
        idx: 3 FAIL ABS  act: 1.69916935e+01 exp: 1.69919109e+01 tol: 9.99999975e-05
        idx: 7 FAIL ABS  act: 2.56341431e+02 exp: 2.56343323e+02 tol: 9.99999975e-05
        idx: 9 FAIL ABS  act: 4.80718613e+01 exp: 4.80722618e+01 tol: 9.99999975e-05
        idx: 10 FAIL ABS  act: 1.20188065e+01 exp: 1.20190525e+01 tol: 9.99999975e-05
        idx: 11 FAIL ABS  act: 2.40540451e+02 exp: 2.40539322e+02 tol: 9.99999975e-05
        idx: 12 FAIL ABS  act: 3.09436970e+01 exp: 3.09440041e+01 tol: 9.99999975e-05
```

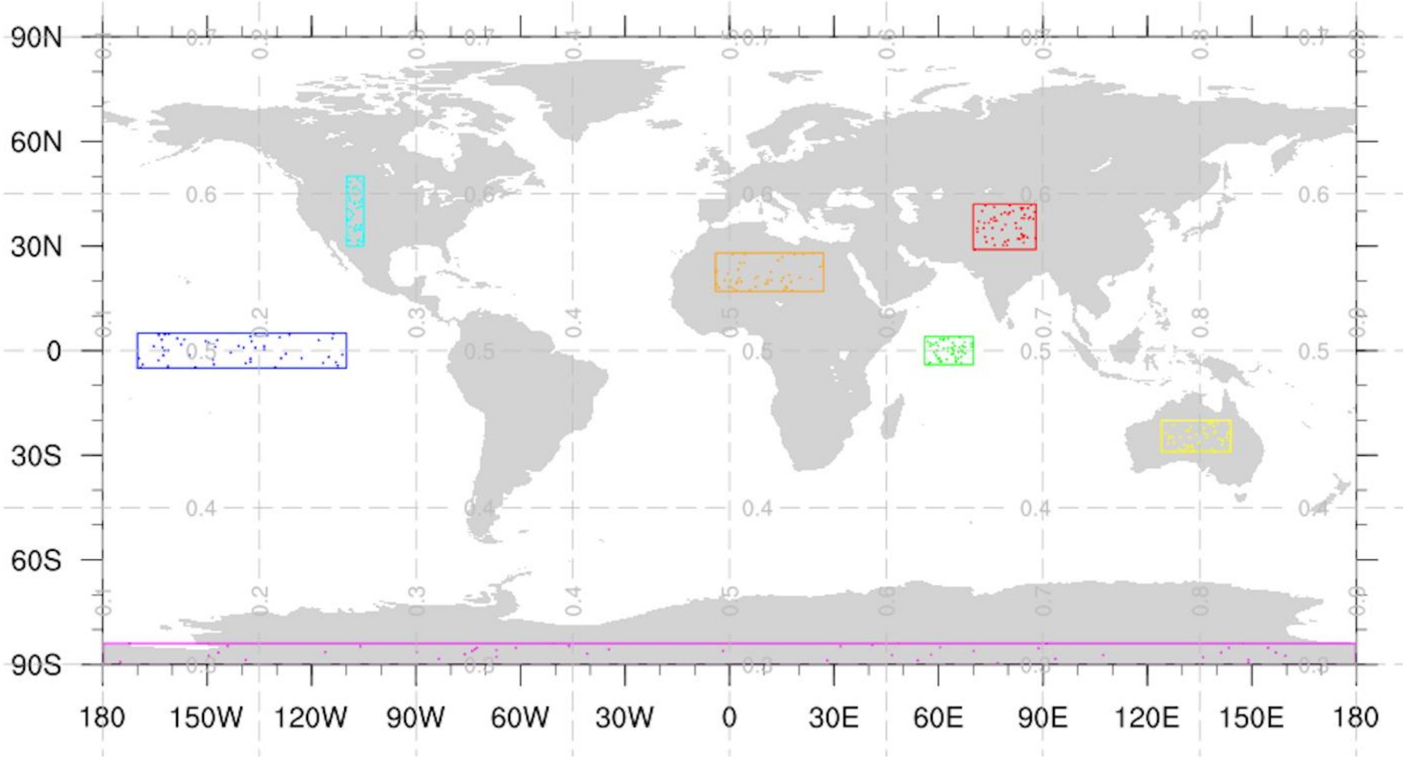
Is this numerical amplification of roundoff errors, or a bug?

Correctness: MPAS Validation Tool

- Identify physically based criteria to assist the validation (remove any “noise” in the data that can be attributed to realistic and anticipated considerations)

- Select different regions globally: mountains, deserts, oceans, ice-caps
  - Model imbalances over high terrain differ from domains over flat surfaces
  - Initial conditions routinely bias conditions differently between polar and tropical regions
  - Scatter domains around the globe so that day and night are considered

Correctness: MPAS Validation Tool



Six domains to cover extremes

MPAS Validation Script Output

Test Cases

- A = Standard case, -O3, r4
- B = Standard case, -O0, r8
- C = A + GPU-ized code, run on CPUs
- D = A + different physics

Probability Reject Null Hypothesis
0 => Same Data
> 0.95 => "Significant differences"

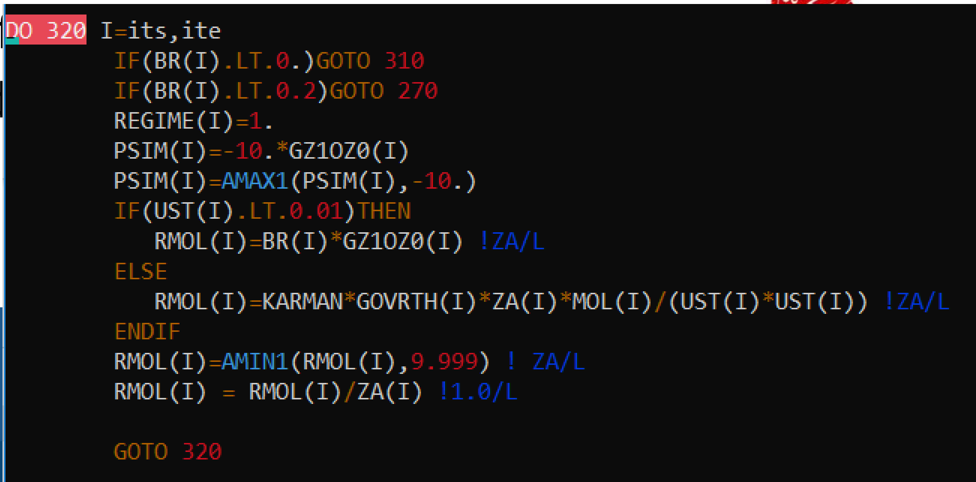
Test	Theta	Qv	U
A vs B	0.07	0.03	0.003
A vs C	0.000	0.02	0.000
A vs D	1.00	1.00	0.14

Bill and Ted’s *Excellent* Physics GPU Port

- Some differences PCAST detected were red-herrings.
  - Low-order bit numerical accuracy issues in single precision got amplified in the physics to O(10^{-3}).
  - Eventually we realized that these differences didn’t significantly influence core state variables.

- Passing Fortran arrays between F90 (dynamics) and F77 (physics) styles confused the PGI compiler, especially deep in the physics call tree.
  - Some arrays became “new” instead of “present” and correct values were replaced with uninitialized arrays.

- MPAS-A physics had several loops with goto statements.
  - Made debugging extremely dif
  - Code was rewritten, in the end



```
DO 320 I=its,ite
  IF(BR(I).LT.0.)GOTO 310
  IF(BR(I).LT.0.2)GOTO 270
  REGIME(I)=1.
  PSIM(I)=-10.*GZ1OZ0(I)
  PSIM(I)=AMAX1(PSIM(I),-10.)
  IF(UST(I).LT.0.01)THEN
     RMOL(I)=BR(I)*GZ1OZ0(I) !ZA/L
  ELSE
     RMOL(I)=KARMAN*GOVRTH(I)*ZA(I)*MOL(I)/(UST(I)*UST(I)) !ZA/L
  ENDIF
  RMOL(I)=AMIN1(RMOL(I),9.999) ! ZA/L
  RMOL(I) = RMOL(I)/ZA(I) !1.0/L

  GOTO 320
```

Excellent Physics GPU Port (part 2)

- **Consider this code snippet:**

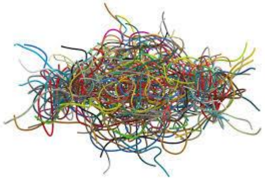
```
 subroutine foo

 real, allocatable :: a(:,:)
 :
 allocate(a(nx,ny,nz))
 call bar(a(1,1,1))
 deallocate(a)
:
 allocate(a(nx,ny,nz-1))
 call bar(a(1,1,1))
 deallocate(a)

 end subroutine foo
```

```
subroutine bar (a)
real a(nx,*)
:
end subroutine bar
```

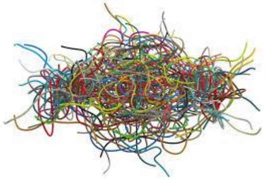
- The OpenACC compiler had a hard time determining the array size. This forced us to use several host copies to ensure that the OpenACC compiler got the right size.

Excellent Physics GPU Port (part 3)

- Junk data copied from host due to data scoping errors.
  - Root cause: model complexity + large team size
  - However, thanks to PCAST, this was the least of our problems.



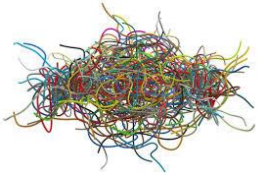
- Array transpositions between physics and dynamics were doing “sneak” computations.
  - We missed these, much to our misfortune.



- **Final score:**

2

2



2

Excellent Physics GPU Port: Lessons Learned

- **PCAST was a lifesaver throughout this process, single handedly converted months of work to weeks.**
  - providing a flag that can enable comparisons of all variables for every kernel instead of from “host directive” locations;
  - pointing the exact line number of the code where the deviation occurred.
- **MPAS Validation scripts have vital in sorting out red herrings.**
  - The scripts’ simplicity and versatility helped to narrow down numerical issues on GPU.
- **Other than the F77 issues, the PGI Fortran OpenACC compiler was robust w.r.t F90 code**.
- *It’s not what you don’t know that gets you into trouble it’s what you know for sure that just ain’t so.*

Earth System Modeling Catch 22 (reminder)

- Due to insufficient computing power ESMs can’t resolve key phenomena.

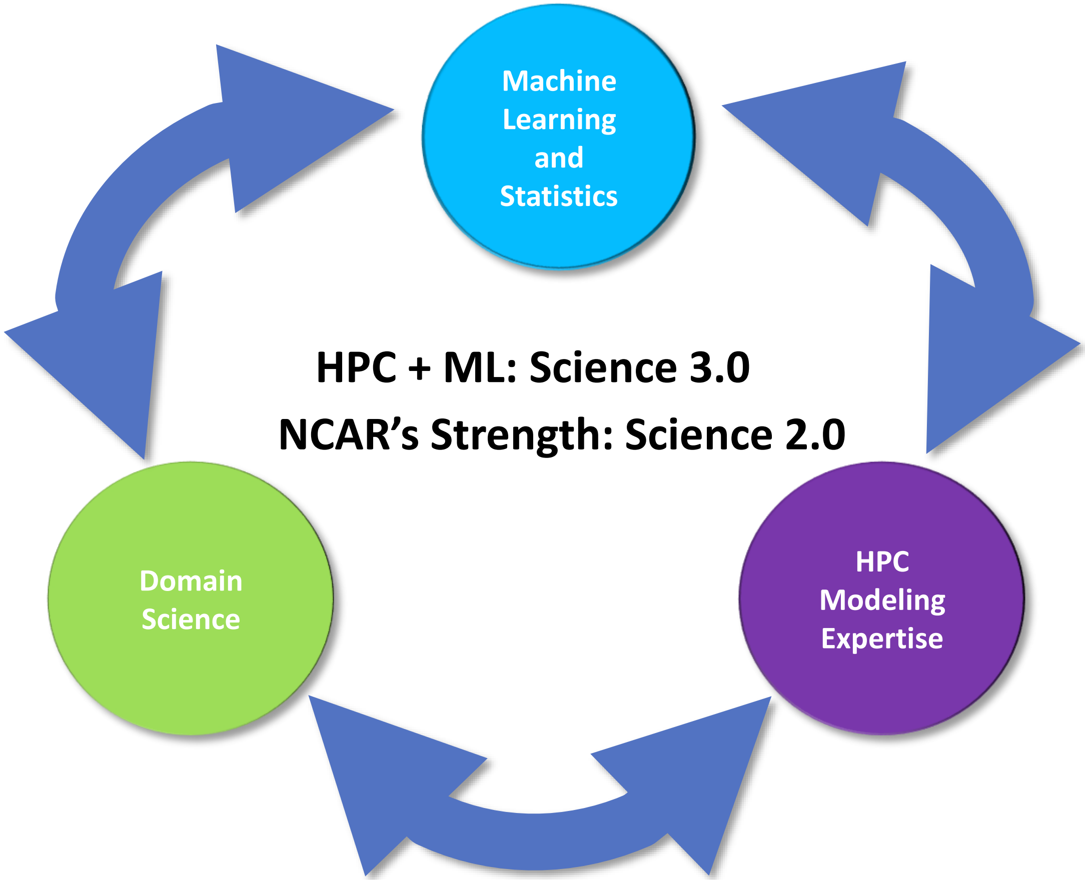
- Scientists try to describe the unresolved scales using human-crafted physics parameterizations.

- ESM’s *software complexity* grows, driven by the increasing complexity of these parameterizations.

- Growing *architectural complexity* hinders the ability to port and optimize ESM codes on new architectures.

- Due to insufficient computing power ESMs can’t resolve key phenomena.

Science 3.0: Blending Machine Learning and Traditional HPC



HPC + ML: Science 3.0

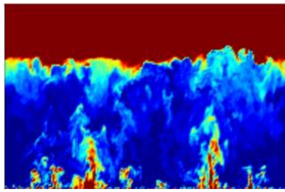
NCAR's Strength: Science 2.0

AIML: New Machine Learning Group at NCAR

AIML Founding Research Focus: model emulation

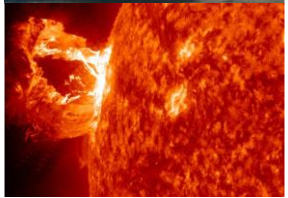
Why machine-learned emulation? The *per-core performance* of conventional computer architectures has stagnated, and models are getting *increasingly complex*. Replacing human-crafted parameterizations with machine learning algorithms may simplify, accelerate and improve models.

- **Sub-grid-scale turbulence -Drs. Kosovic & Haupt (RAL), Gagne (AIML)**
  - **improved representation of the surface layer in meteorological models**



- **Cloud microphysics - Drs. Gettelman (CGD), Gagne & Sobhani (AIML)**
  - **improved weather and climate modeling**

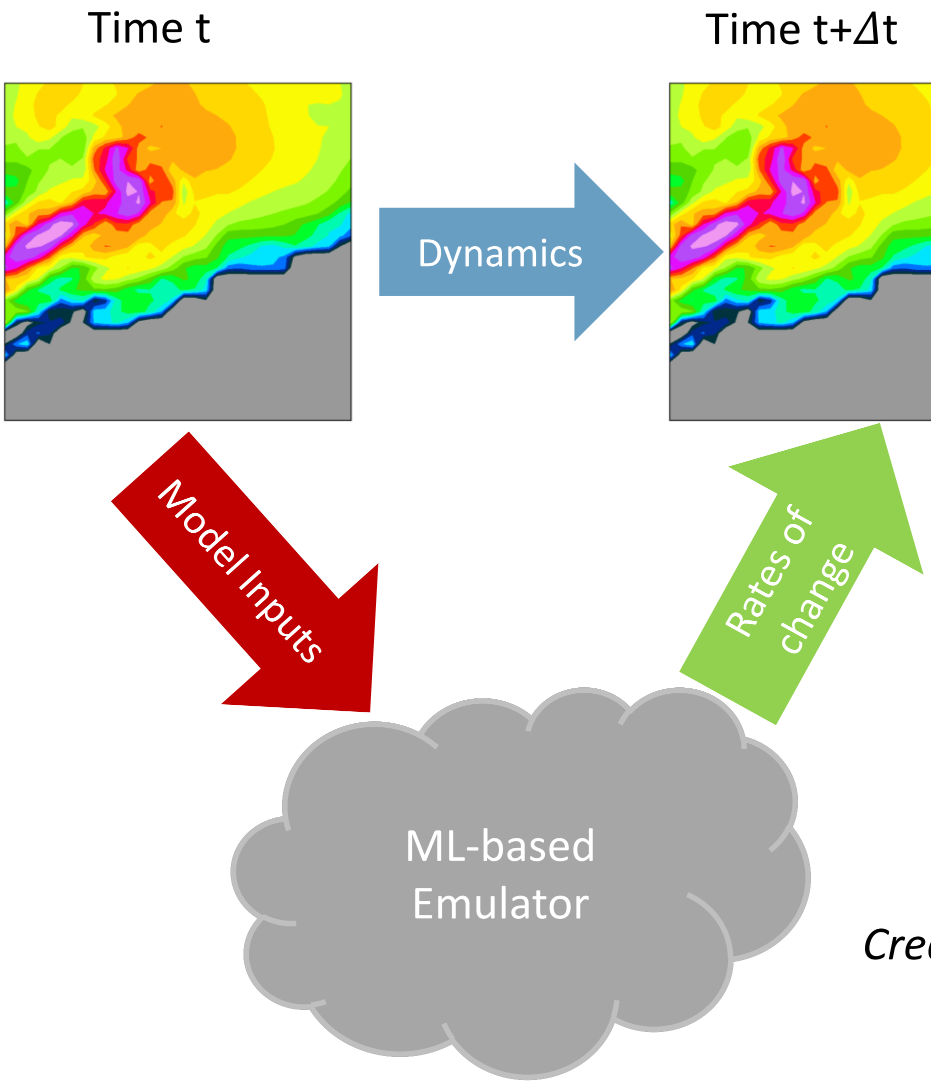
- **Interplanetary coronal mass ejection (CME) - Drs. Gibson (HAO), Flyer (AIML)**
  - **space weather prediction**



- **Seasonal weather patterns - Drs. Sobhani (AIML) & DelVento (CISL)**
  - **Seasonal prediction of dangerous hot weather in the Eastern U.S.**

Combining numerical modeling and ML

Replacing Models with Emulation

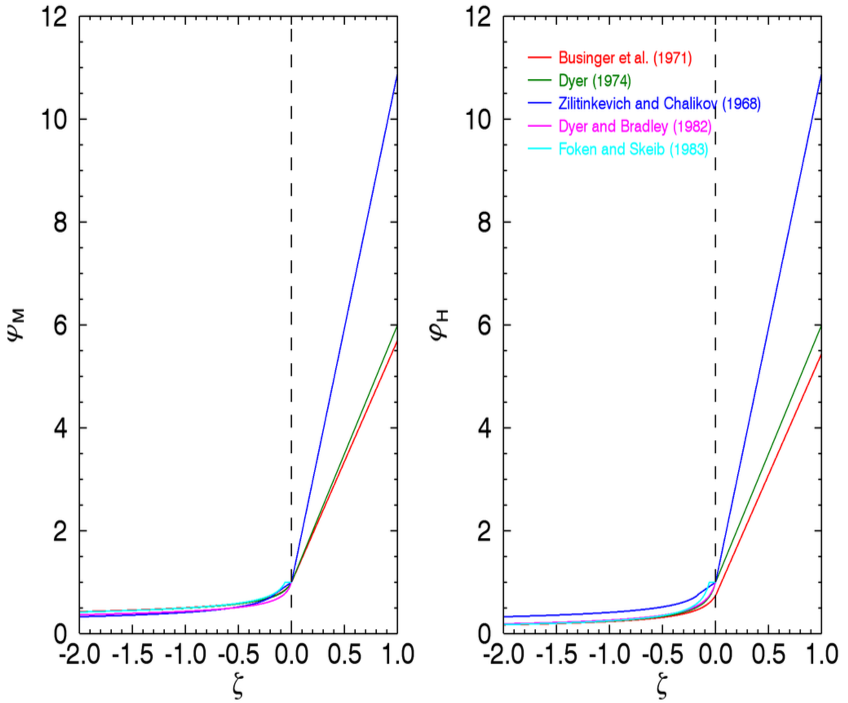


Credit: D.J. Gagne, NCAR

Machine Learning Research and Applications

Surface Layer Parameterization

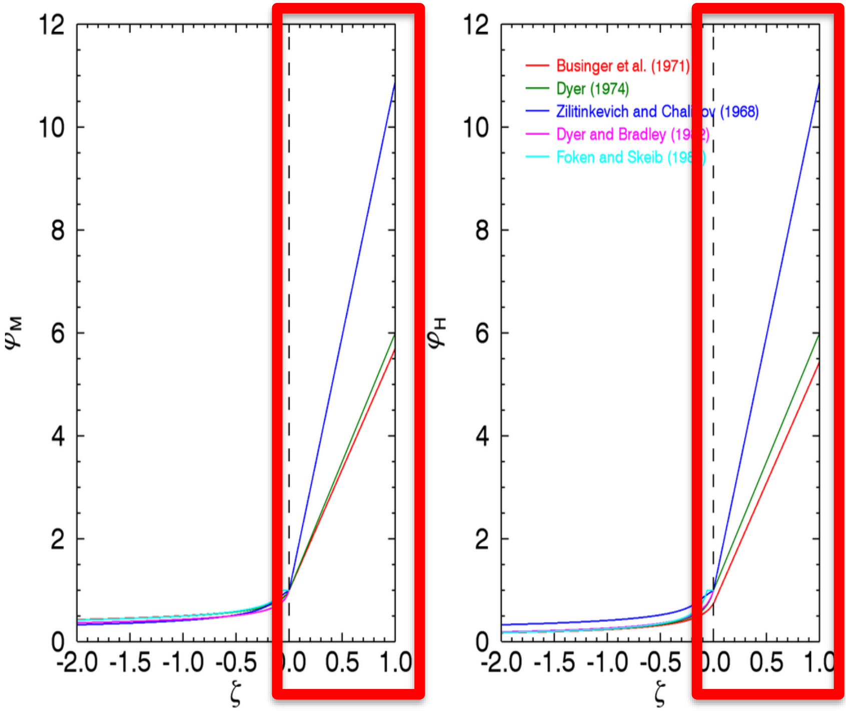
- In atmospheric models Monin-Obukhov similarity relations are used to determine surface fluxes and stresses
- Stability functions are determined experimentally from field studies under nearly ideal atmospheric flow conditions characterized by horizontally homogeneous flat terrain and stationarity



Machine Learning Research and Applications

Surface Layer Parameterization

- …but, even under such idealized conditions, in particular under stable stratification, there is large variation in stability functions determined from different field studies
- Goal: Use Machine Learning to replace M-O Similarity Theory in NWP Models



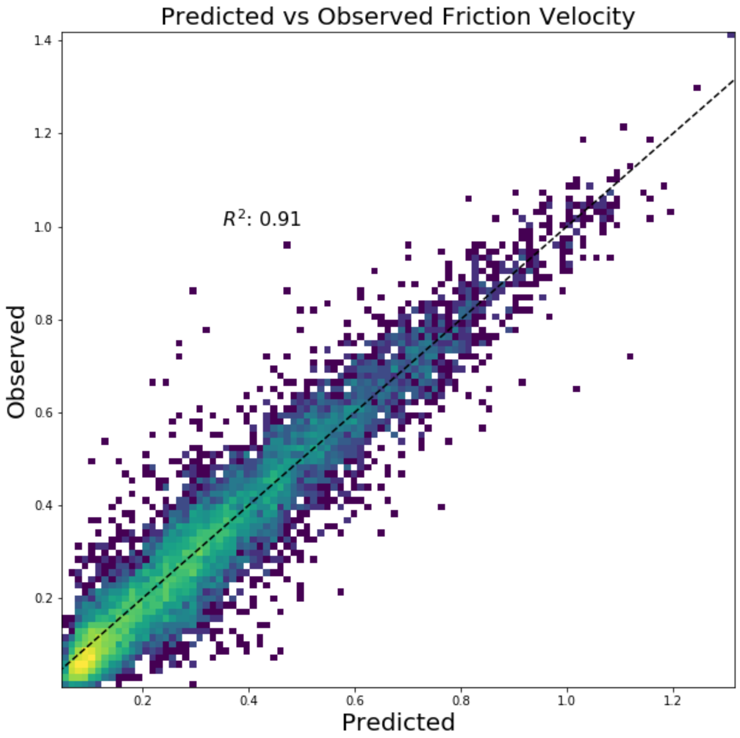
NCAR UCAR

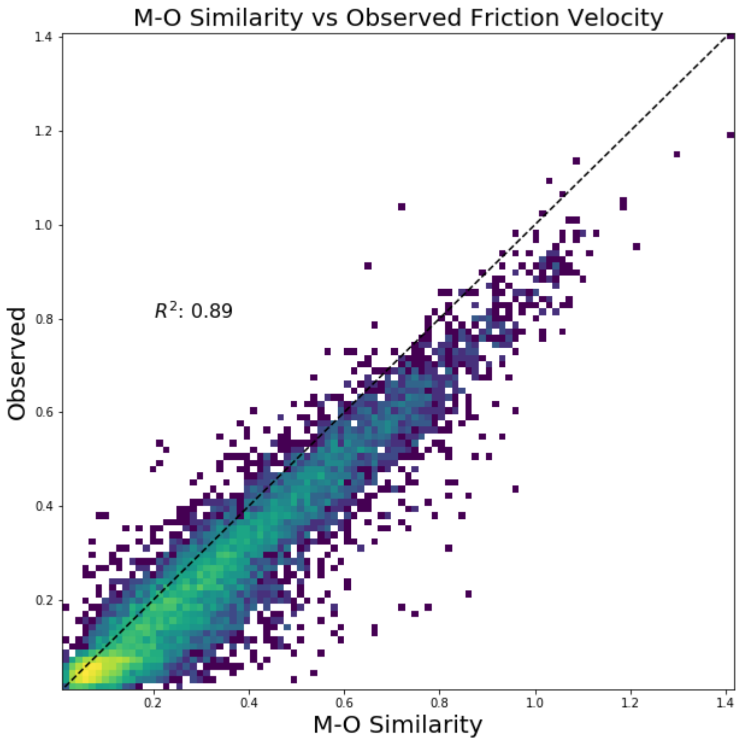
air • planet • people

Machine Learning Research and Applications

Surface Layer Parameterization

- Random Forest on Idaho Dataset (trained on 2016, 2017 and test results shown on 2015 data)
- Lower error and higher R^2 than M-O Similarity Theory for Friction Velocity

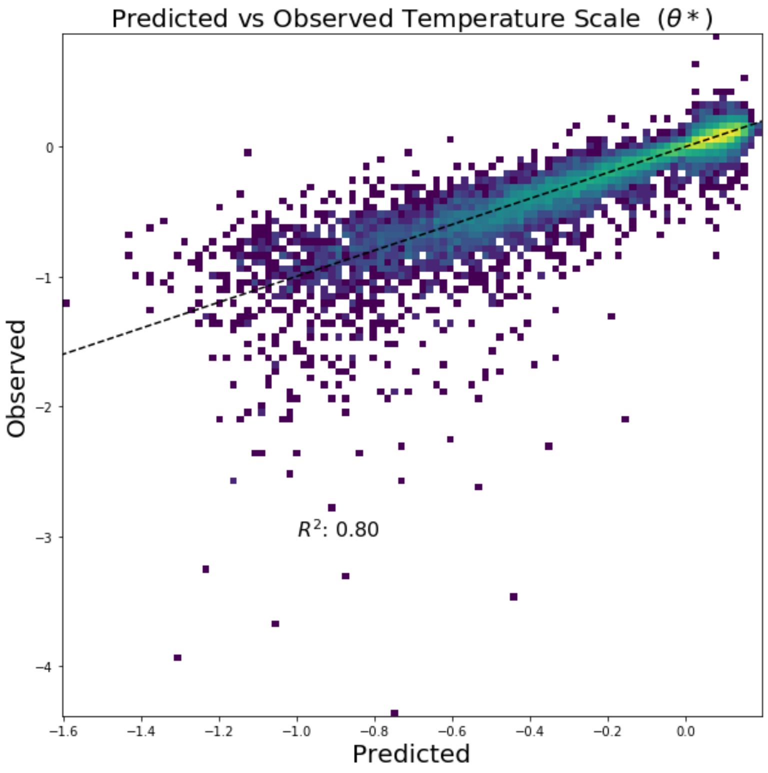


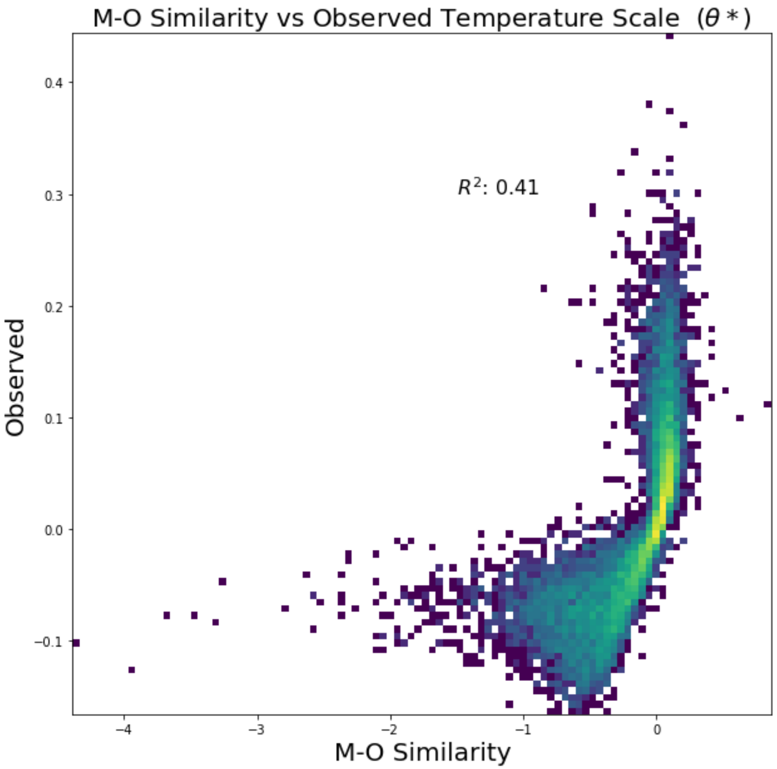


Machine Learning Research and Applications

Surface Layer Parameterization

- Random Forest on Idaho Dataset (trained on 2016, 2017 and test results shown on 2015 data)
- Lower error and higher R^2 than M-O Similarity Theory for Temperature Scale





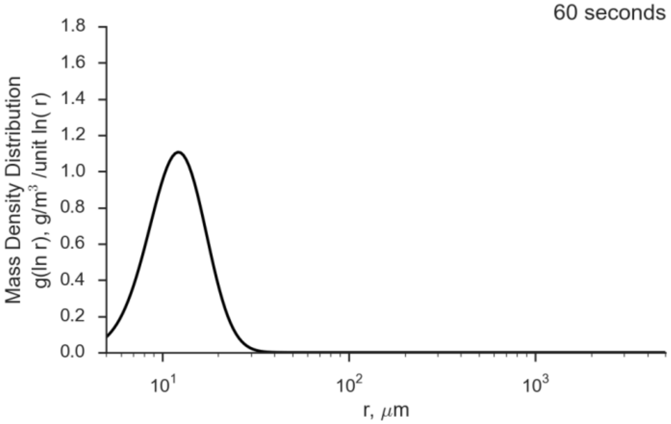
If you train an ML algorithm with data from one place, does it work in another?

	R2			MAE		
Idaho Test Dataset	Friction Velocity	Temperature Scale	Moisture Scale	Friction Velocity	Temperature Scale	Moisture Scale
MO Similarity	0.85	0.42		0.077	0.203	
RF Trained on Idaho	0.91	0.80	0.41	0.047	0.079	0.023
RF Trained on Cabauw	0.88	0.76	0.22	0.094	0.139	0.284
	R2			MAE		
Cabauw Test Dataset	Friction Velocity	Temperature Scale	Moisture Scale	Friction Velocity	Temperature Scale	Moisture Scale
MO Similarity	0.90	0.44		0.115	0.062	
RF Trained on Cabauw	0.93	0.82	0.73	0.031	0.030	0.055
RF Trained on Idaho	0.90	0.77	0.49	0.074	0.049	0.112

ML algorithm wins more “away games” than M-O theory

Emulating Cloud Microphysics: Motivation

- Precipitation formation is a critical uncertainty for weather and climate models.
- Different sizes of drops interact to evolve from small cloud drops to large precipitation drops.
- Detailed codes (right) are too expensive for large scale models, so empirical approaches are used.
- Let's emulate one (or more)
- Goal: put a detailed treatment into a global model and emulate it using ML techniques.
- Good test of ML approaches: can they reproduce a complex process, but with simple inputs/outputs?



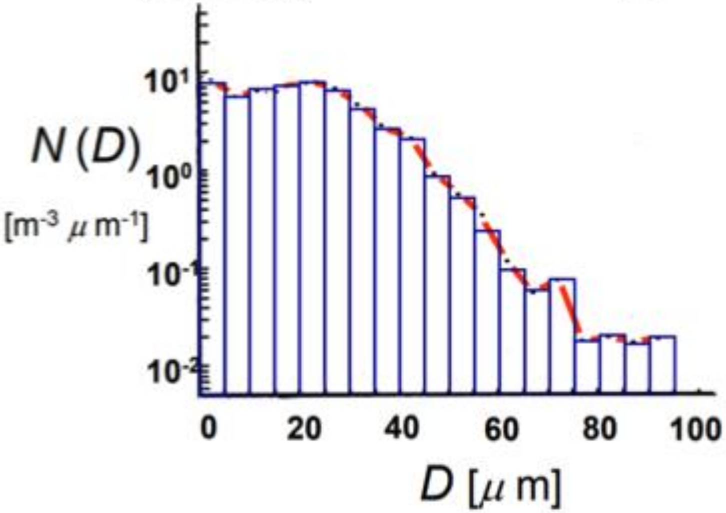
Sd-coal model output animation

Credit: Daniel Rothenberg

Ultimate Goal: Predict evolution of hydrometeor size distributions

Bin-resolving: $N(D) = \sum_{i=1}^{I} N_i$

(spectral)

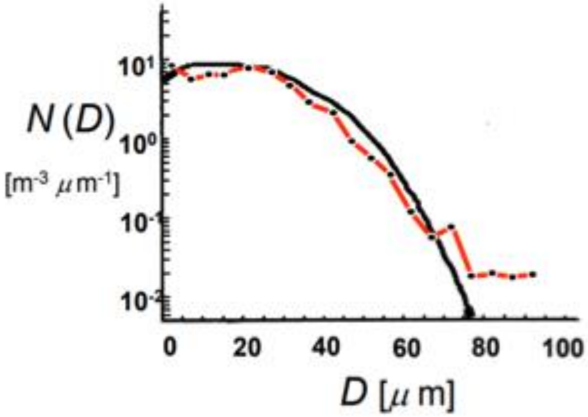


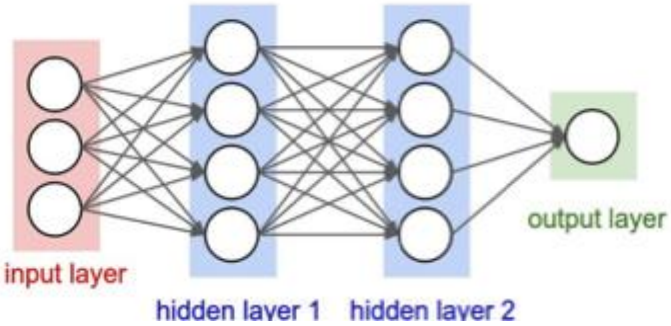
Divide microphysical particles into bins for different sizes, and compute evolution of each bin separately

Accurate but too expensive.

Bulk schemes calculate with a semi-empirical particle size distribution (PSD).

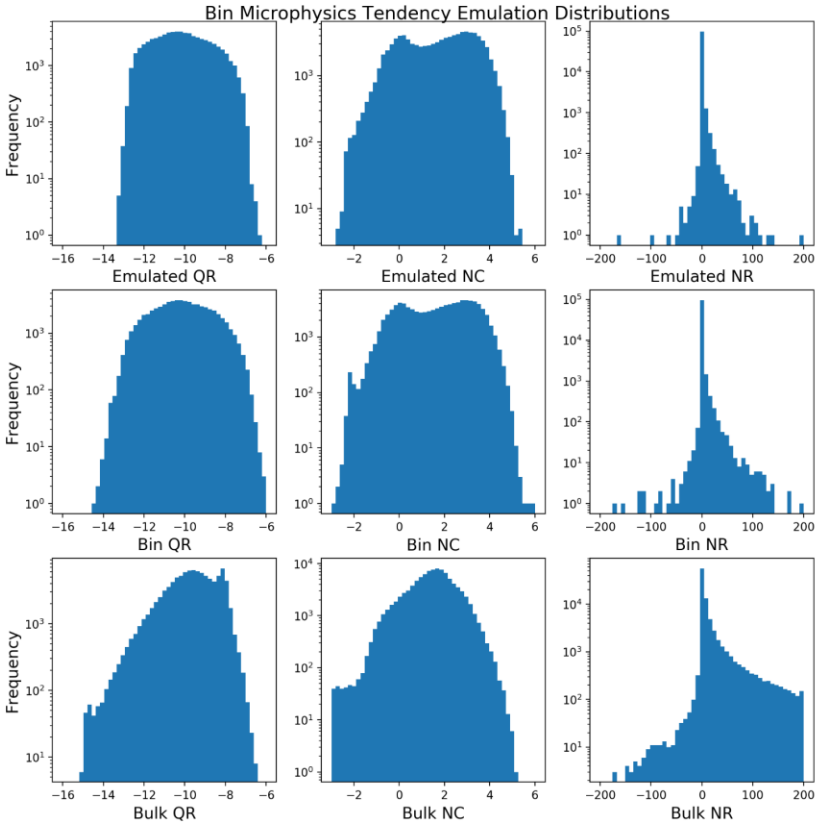
Bulk: $N(D) = N_0 D^{\alpha} e^{-\lambda D}$





Credit: Gagne & Gettelman, NCAR

Microphysics Emulator Results



Emulated ✓

Neural network microphysics emulates distribution and exact values of bin microphysics more closely than bulk microphysics

Bin - too expensive for climate

Bulk - affordable for climate

Credit: Gagne & Gettelman, NCAR

Outstanding emulator challenges

- Ensuring interpretability & reproducibility of ML emulator results.
- Conditioning/scaling inputs are critical to the successful formulation of a successful emulator.
- Tuning emulator hyper-parameters for optimal performance.
- Representing extreme/unusual events in the emulator’s training data.
- Getting ML emulators to respect constraints.
- Ensuring ML model robustness under iterative maps (time integration).

Talk Summary

- **Science 3.0: HPC + ML**
  - Apply GPUs to accelerate models where physics is rigorous.
  - Replace parameterizations with Machine Learning emulators where the physics is phenomenological.

- Initial results are encouraging…

- But much more work needs to be done to prove these ideas out!

Thanks!