NOAA Exascale Computing Project

Mark Govett

Christina Bonfanti, Bryan Flynt, Chris Harrop, Isidora Jankov,

Ed Hartnett, Jacques Middlecoff, Duane Rosenberg,

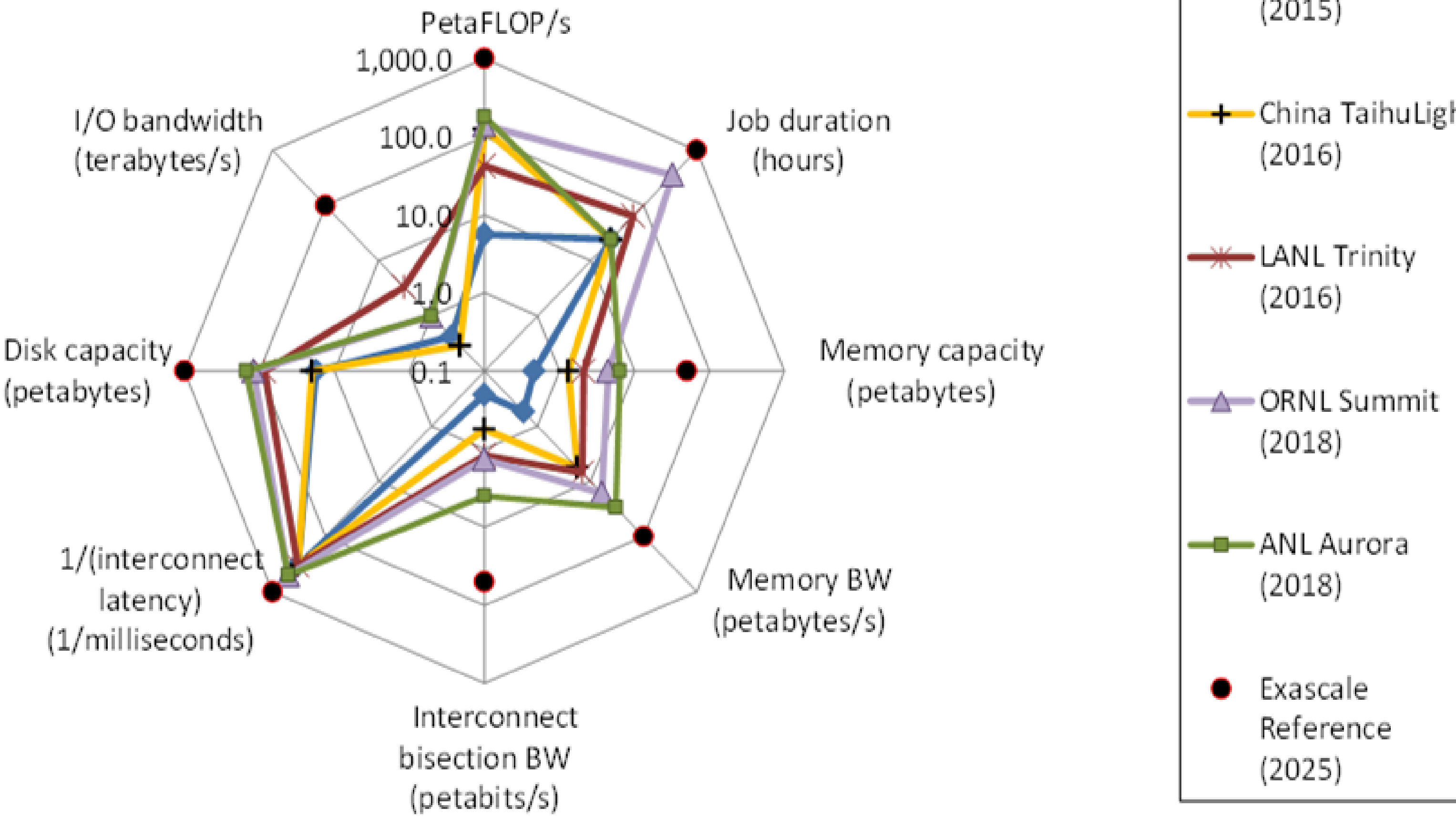
Jebb Stewart, Lynd Stringer, Lidia Trailovic, Yonggang Yu

Background

- Project began in November 2017
- To address challenges in future computing
  - Increasing diversity
  - More cores
  - I/O and inter-process communications concerns
- Key questions
  - How to expose sufficient parallelism for systems with millions of cores?
  - What is the best path to good performance and scalability?
  - Is performance – portability achievable?
  - Do we need to rewrite our applications?
    - Optimize ---> Refactor ---> Rewrite?
  - Can model and data assimilation be more tightly linked?

Hardware & Appli

- Diverse processor, systems
  - x86, ARM, GPU, Power+GPU
  - Powerful processors / nodes
  - Slow I/O, inter-process communications
  - Massive parallelism
- Applications
  - Complex, diverse
  - Low compute intensity
  - Limited parallelism



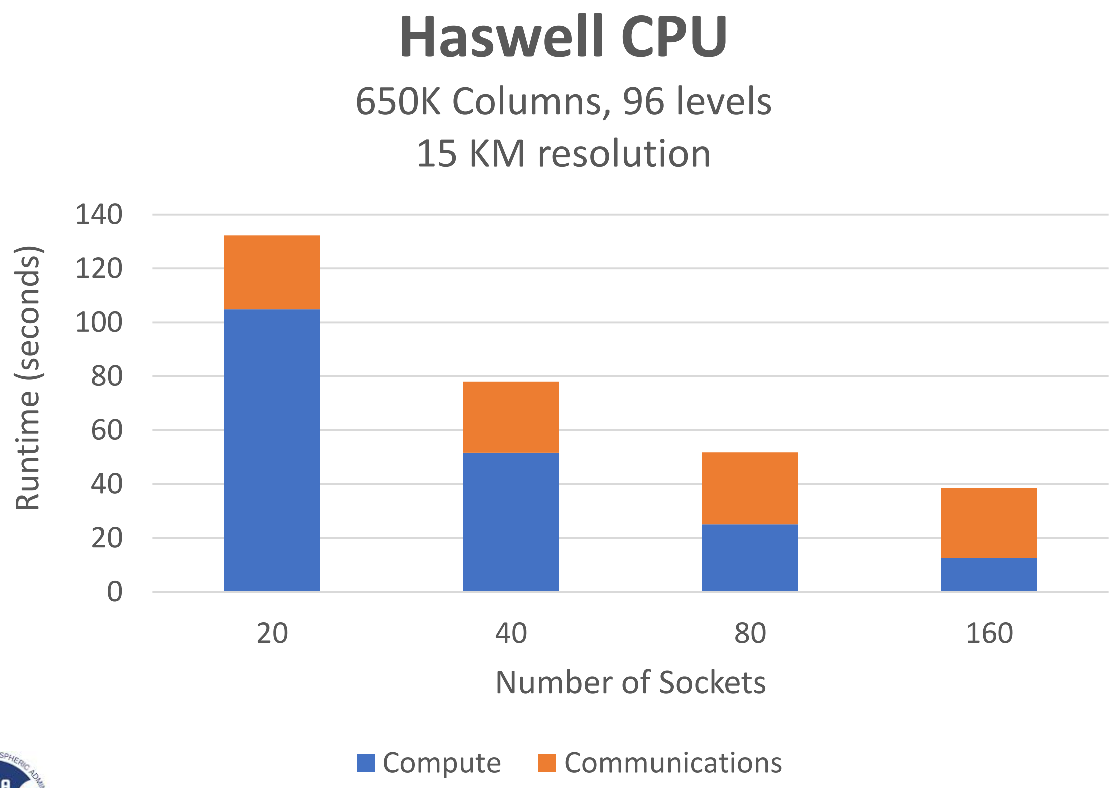
Credit: HPCMP Architectural Trends - Global to Corporate View, DOD HPC Modernization, February 2017

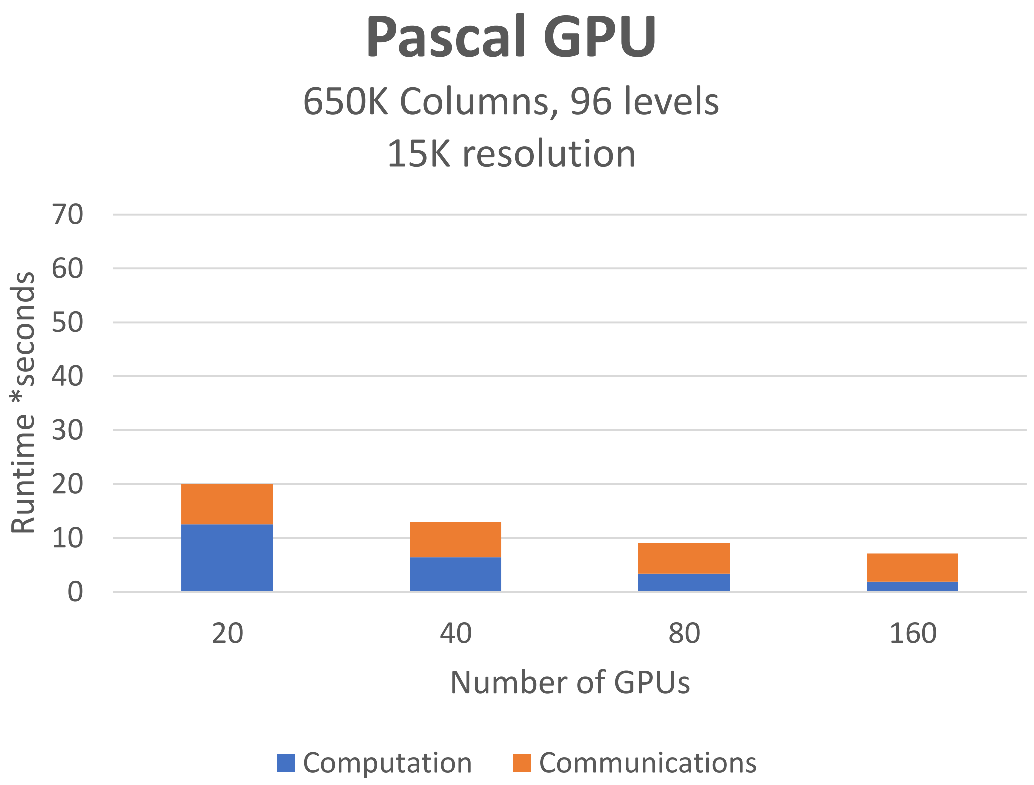
Application Performance Measures

- Percentage-of-peak
- Speedup
- Scalability
- Time-to-solution
  - Weather Forecast
    - 1 hour to produce a 10 day forecast === > 6 minutes / day ==== > 0.65 YPD
    - 1 hour to produce an 8 day forecast === > 7.5 minutes / day ==== > 0.52 YPD
  - Climate Prediction
    - 5 years per wall clock day is reasonable today
  - How much time should we spend on assimilation versus forecast?
    - NCEP requires DA run in 20 minutes, ECMWF allows 40 minutes

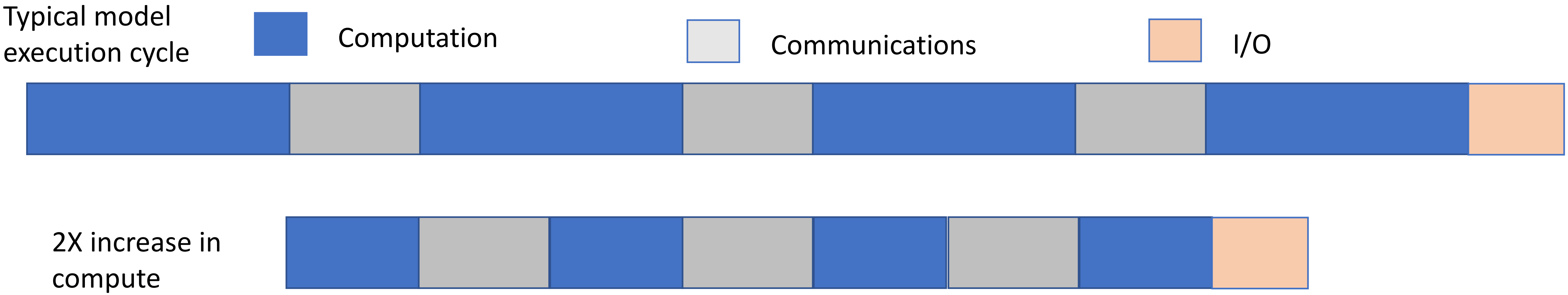
Strong Scaling: CPU, GPU

- CPU socket (2 per node) versus Pascal GPU (2 per node)
  - Identical system, interconnect, data movement per MPI task
  - Different communications CPU: impi, GPU: mvapich), affinity (CPU: range, GPU: pinned)



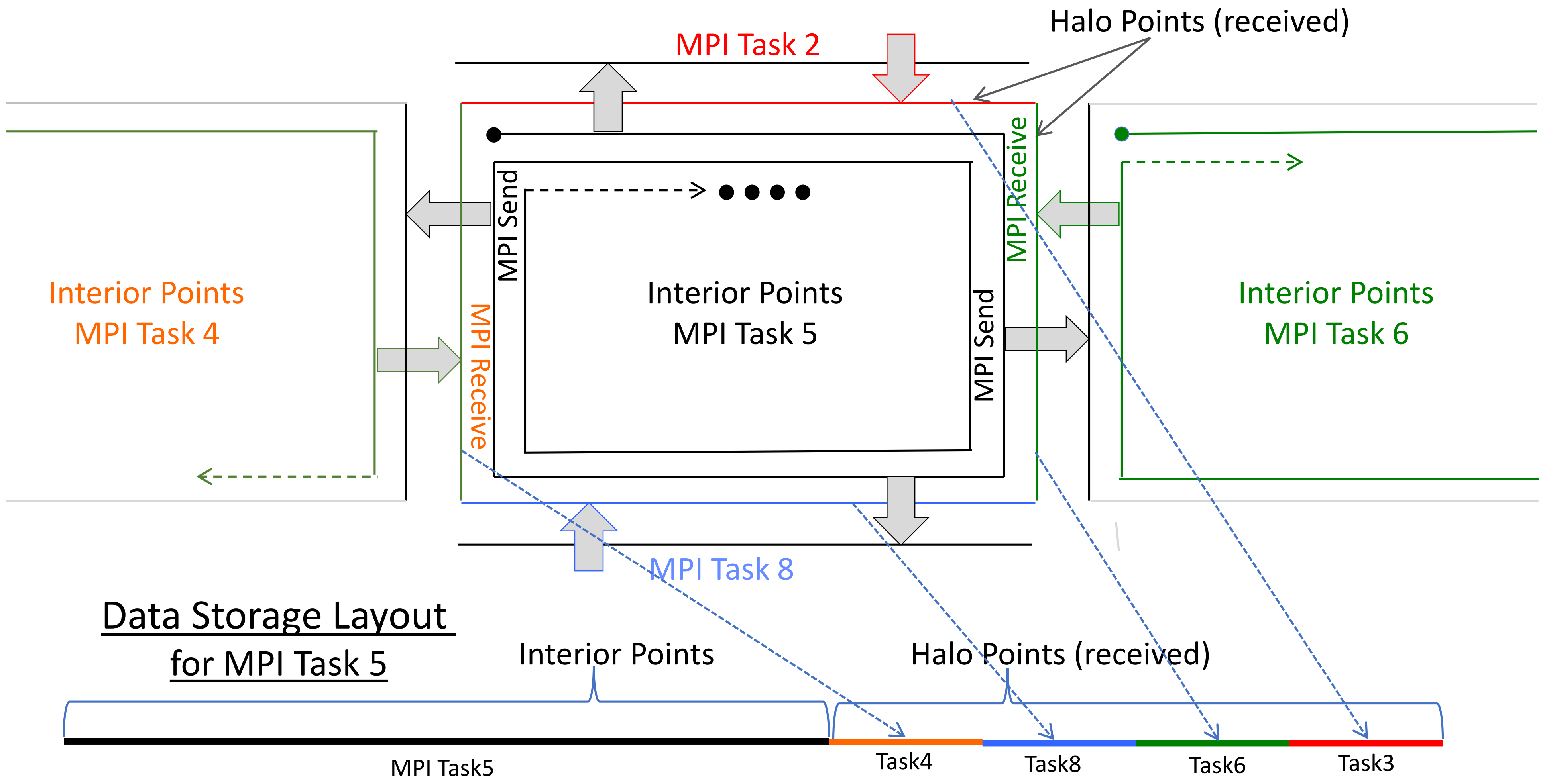


Communications Performance & Scalability



- Inter-process communications biggest factor affecting scalability
- Common techniques to reduce impact of MPI communications
  - Overlap communications with computation
  - Aggregation
  - Reduce frequency & volume of data
  - Reorder points to avoid MPI pack / unpack (Middlecoff, 2015)

Spiral Grid Ordering

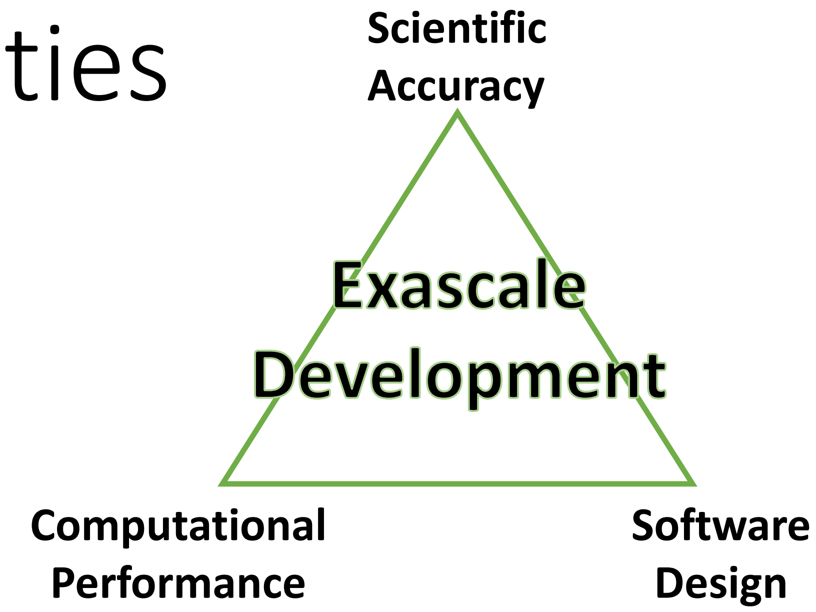


Parallelization and Performance of the NIM on CPU, GPU and MIC Processors, Bulletin of the AMS, Govett, et al, 2017

Performance - Portability

- Goal to have same code for x86, ARM, GPU, etc
- Languages
  - Fortran, C, C++, CUDA
  - Domain-Specific Languages (DSLs)
    - Source to source, high level to machine abstractions
  - Libraries: MPI, SMS, RSL
- Directives
  - OpenMP, OpenACC, SMS

Exascale Development Activities



- Model Dwarfs
  - Advection
  - Grid staggering
- Data Assimilation
  - Prototypes
  - TL & ADJ generation
  - Optimization
- Machine Learning
- I/O
- Software Design

- Evaluate scientific accuracy and computational efficiency
- Improve software process
- Incorporate science & computational aspects into design

Advection Dwarf Development & Beyond

Duane Rosenberg, Bryan Flynt

AGU, December 10-14, 2018

Characterizing and Improving Scientific Algorithms and I/O for Exascale: Nimble Dwarfs,

Bryan T. Flynt, Duane Rosenberg, Yonggang Yu and Mark Govett

Scientific Motivation & Scope

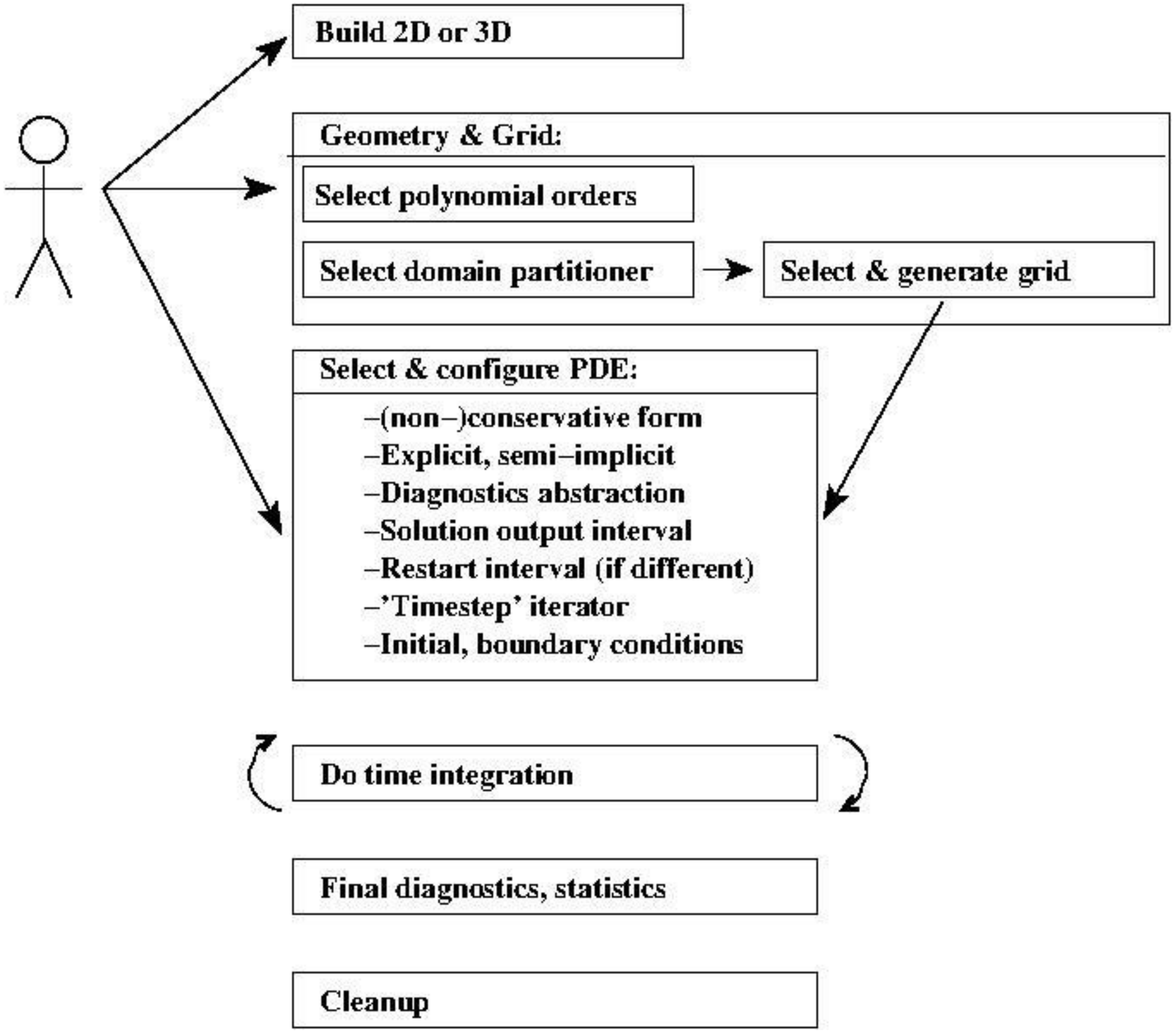
To develop highly accurate, high performance models for atmosphere (and ocean)

Evaluate primary components of geofluid models with spectral element/DG and finite-volume approaches to examine parallel performance and scientific accuracy

Dwarf Development: Requirements and Initial Focus

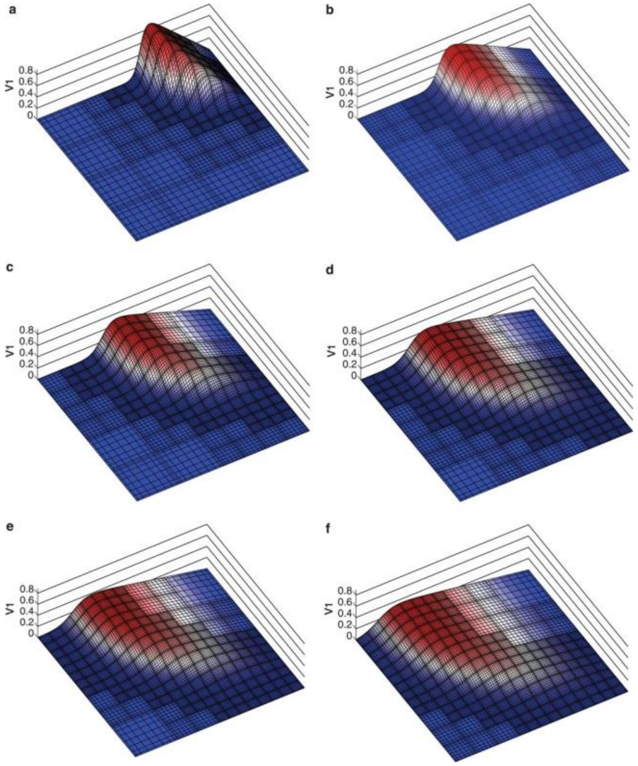
Requirements	Initial Focus
Simplest framework to *test* linear, nonlinear, (non-)conservative, transport in idealized setting Extensible to allow different grid types, PDEs	Advection-diffusion: Burgers equation, DG and CG forms: $\partial_t \mathbf{u} + \nabla \cdot (\mathbf{u}\mathbf{u}) = \mathbf{u}\nabla \cdot \mathbf{u} + \nu \nabla^2 \mathbf{u} \quad (1)$ $\partial_t \mathbf{u} + \mathbf{c}(\mathbf{u}) \cdot \nabla \mathbf{u} = \mathbf{f} + \nu \nabla^2 \mathbf{u} \quad (2)$
Must be discretized on the sphere, 2D and 3D	Icosahedral (2D) and spherical-polar (3D)
Must allow different time stepping methods, explicit & semi-implicit	Explicit: Runga-Kutta
Relatively small, self-contained & *manageable;* compiler, platform portable	Object-oriented; test-driven development
Must accommodate 'coarse' and 'fine'-grain parallelization	MPI-ready, building in OpenACC/OpenMP offloading

Configuration Framework



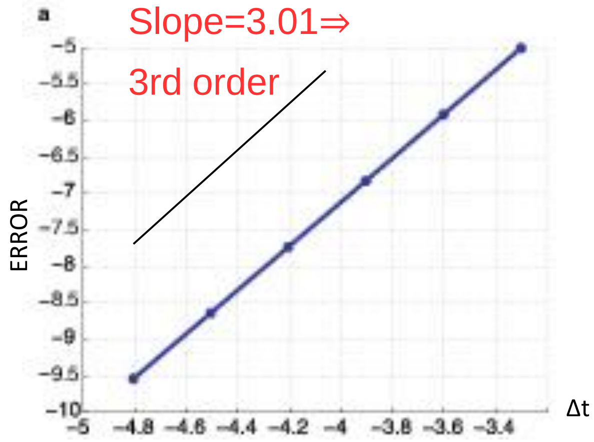
Evaluation of Accuracy and Complexity

Select initial conditions, e.g. N-wave, and evolve:

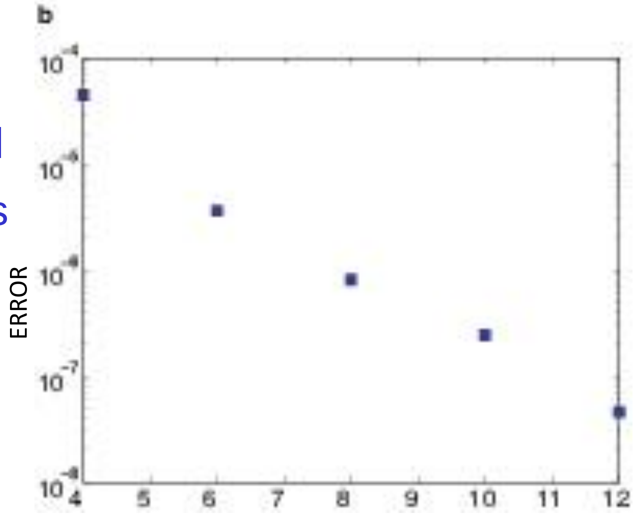


Use (semi-)analytic solution to find:

Error vs **Δ**t yields temporal truncation error

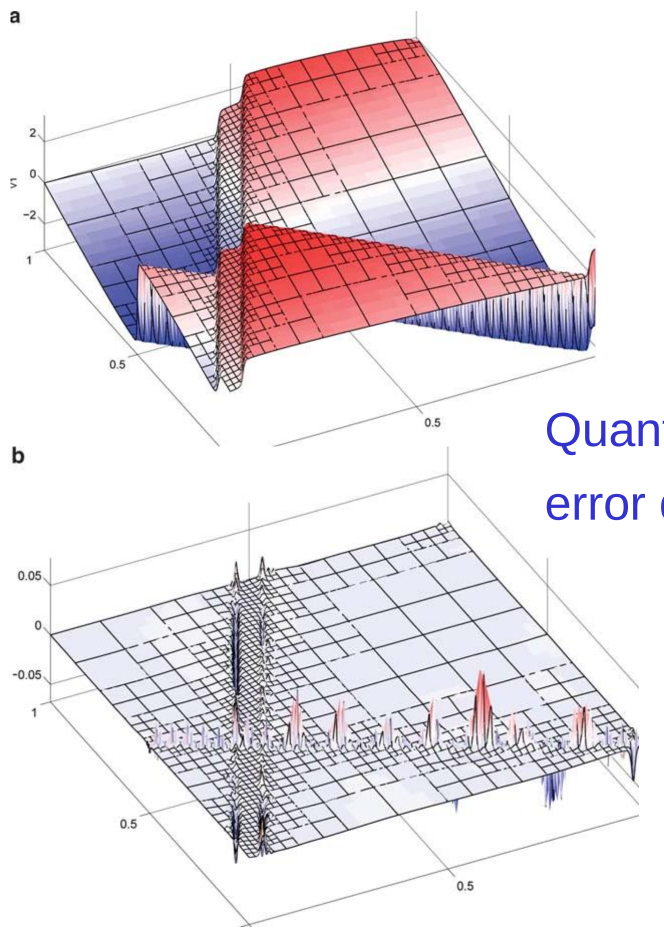


Error vs polynomial order demonstrates spatial spectral convergence



Accuracy and Complexity

Interacting fronts:



Quantify spatial error distribution

Quantify via:

Communication complexity:

Comm. volume/Flops (per elem)

'Accuracy efficiency':

F[T log(Error) / T_0 log(Error_0)]

(T is solution time)

Compare with low order:

$\log N_{cells} \sim p \log (pE)$

($p \equiv$ poly. order; $E \equiv$ # elements)

Grid Staggering

Evaluate the Scientific Accuracy and Computational Performance of the A-grid and C-grid staggering

Yonggang Yu

AMS, January 6 -10

Session: Developing and Preparing Models for Exascale Computers

Comparison of A-grid and C-grid Shallow Water Model Solver on Icosahedral Grids

Yonggang Yu, Ning Wang and Mark Govett

Design and performance testing for A-grid and C-grid Shallow Water Model Solvers for Exascale, Jacques Middlecoff, Yonggang Yu, and Mark Govett

Scope

Small software solving shallow water model on sphere to test computational cost versus scientific accuracy using different spatial staggering schemes

Design principles for this code:

- ❖ Algorithmic Versatility
  - Solving PDE by spatial discretization on A-grid, C-grid, etc
  - Support icosahedral grid and the associated Voronoi cell
- ❖ Support HPC enabling technologies
  - MPI, OpenMP, OpenACC
- ❖ Small and self-contained codes
  - Icosahedral grid generation (Ning Wang)
- ❖ Suitable for profiling and tests

Advantages:

- ❖ Fair comparison among numerical schemes
  - Fixed time integration method (Runge-Kutta 4^{th} order)
  - Avoid bias from different icos grid generators (NCAR, CSU, ESRL)
  - Same level of MPI optimization (e.g. using SMS for message [un]packing, Spiral grid optimization (Jacques Middlecoff))
- ❖ Compilation and Run time consistency
  - Keeping the same compiler option (on same hardware)

Solving Shallow Water Model on Icosahedral Grids using A-grid and C-grid Staggering Schemes

Dynamic equations:

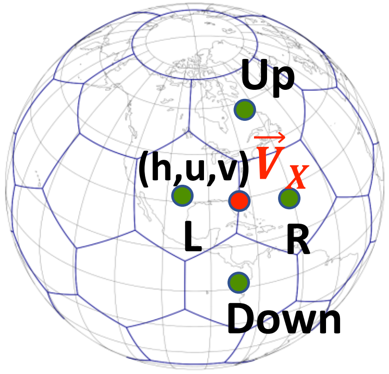
$$\frac{\partial h}{\partial t} + \nabla \cdot \left(h\vec{V}\right) = 0$$

$$\frac{\partial \vec{V}}{\partial t} + \left(2\vec{\Omega} + \nabla \times \vec{V}\right) \times \vec{V} + \nabla\left(\frac{1}{2}\vec{V}\cdot\vec{V} + gh\right) = 0$$

Reference:

- Tomita, H., M. Tsugawa, M. Satoh, and K. Goto, 2001: Shallow Water Model on a Modified Icosahedral Geodesic Grid by Using Spring Dynamics. J. Comput. Phys., 174, 579–613, doi: 10.1006/jcph.2001.6897.
- Thuburn, J., T. Ringler, J. Klemp and W. Skamarock, 2009: Numerical representation of geostrophic modes on arbitrarily structured C-grids, J. Comp. Phys., 2009: 228 (22), 8321
- Ringler, T. D., J. Thuburn, J. B. Klemp, and W. C. Skamarock, 2010: A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids. J. Comput. Phys., 229, 3065–3090, doi:10.1016/j.jcp.2009.12.007.
- Arakawa and Lamb: Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model, in Methods in Computational Physics: Advances in Research and Applications, Vol. 17, p 173 (1977)

A-grid (NICAM method)



$$\vec{V}_X = \frac{1}{2}\cdot\begin{bmatrix}\left(\alpha_1\vec{V}_L + \beta_1\vec{V}_R + \gamma_1\vec{V}_U\right) + \\ \left(\alpha_2\vec{V}_L + \beta_2\vec{V}_R + \gamma_2\vec{V}_D\right)\end{bmatrix}$$

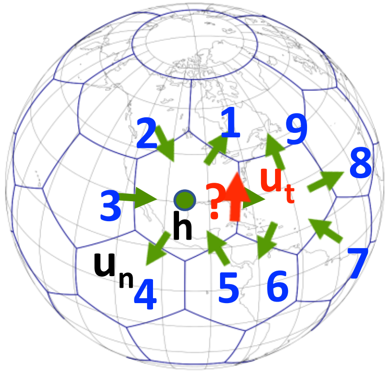
$\alpha_1 + \beta_1 + \gamma_1 = 1$

$\alpha_2 + \beta_2 + \gamma_2 = 1$

⋮

Finite volume approach

C-grid (MPAS method)



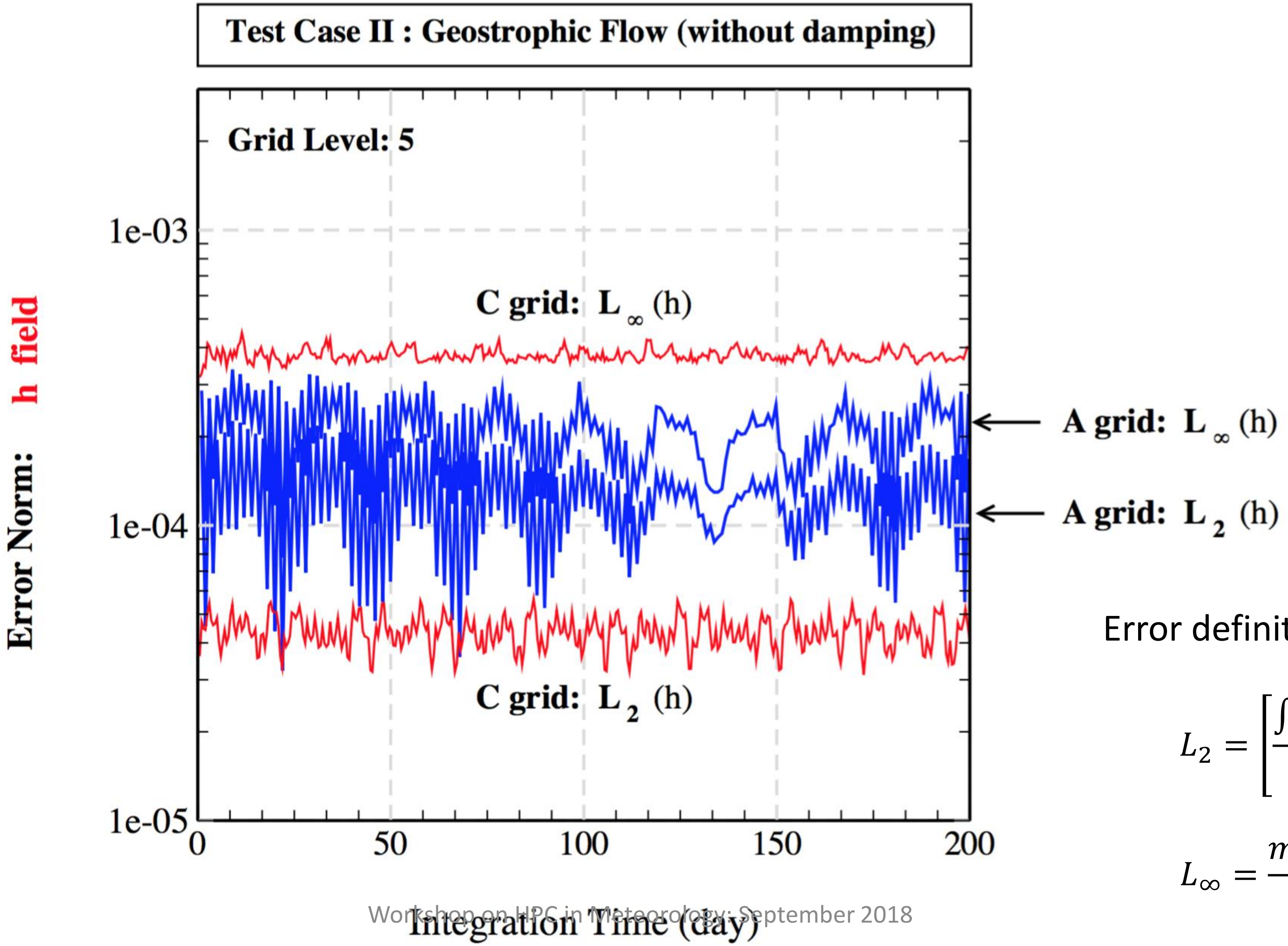
$$U_t^e = \frac{1}{d_e}\sum_{e'=1}^{9\text{ or }10} W_{ee'}\, l_{e'}\, U_n^{e'}$$

$$t_{ev}\left(\hat{U}_n^{e'} \times \hat{U}_n^{e}\right)\cdot\hat{k}\, W_{ee'}$$

$$= \sum_{ip} R_{ip}^{(v)}\, \hat{U}_n^{e'} \cdot \hat{U}_{e'}^{out}$$

$$\frac{\partial U_n}{\partial t} + (\mathrm{f} + \xi)U_t + \hat{U}_n^e \cdot \hat{n}^{out}\frac{\partial(K+\Phi)}{\partial n} = 0$$

Computational Accuracy Study for A-grid v.s. C-grid Staggering



Error definition:

$$L_2 = \left[\frac{\int (h - h_0)^2 d\Omega}{\int {h_0}^2 d\Omega}\right]^{\frac{1}{2}}$$

$$L_\infty = \frac{max|h - h_0|}{max|h_0|}$$

Parallellization and Performance Measurement for A-grid v.s. C-grid Staggering

MPI parallelization schemes:

- Loop over basin centers
- domain decomposition and halos

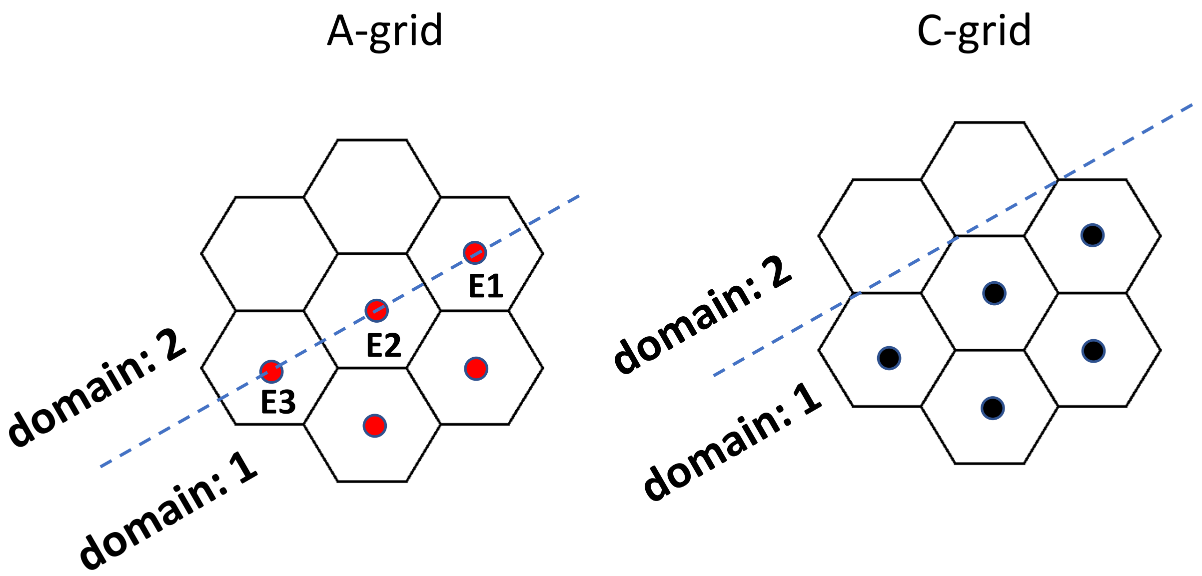
GPU parallelization schemes:

- Loop over basin centers and edges

Impose metrics for fair comparison:

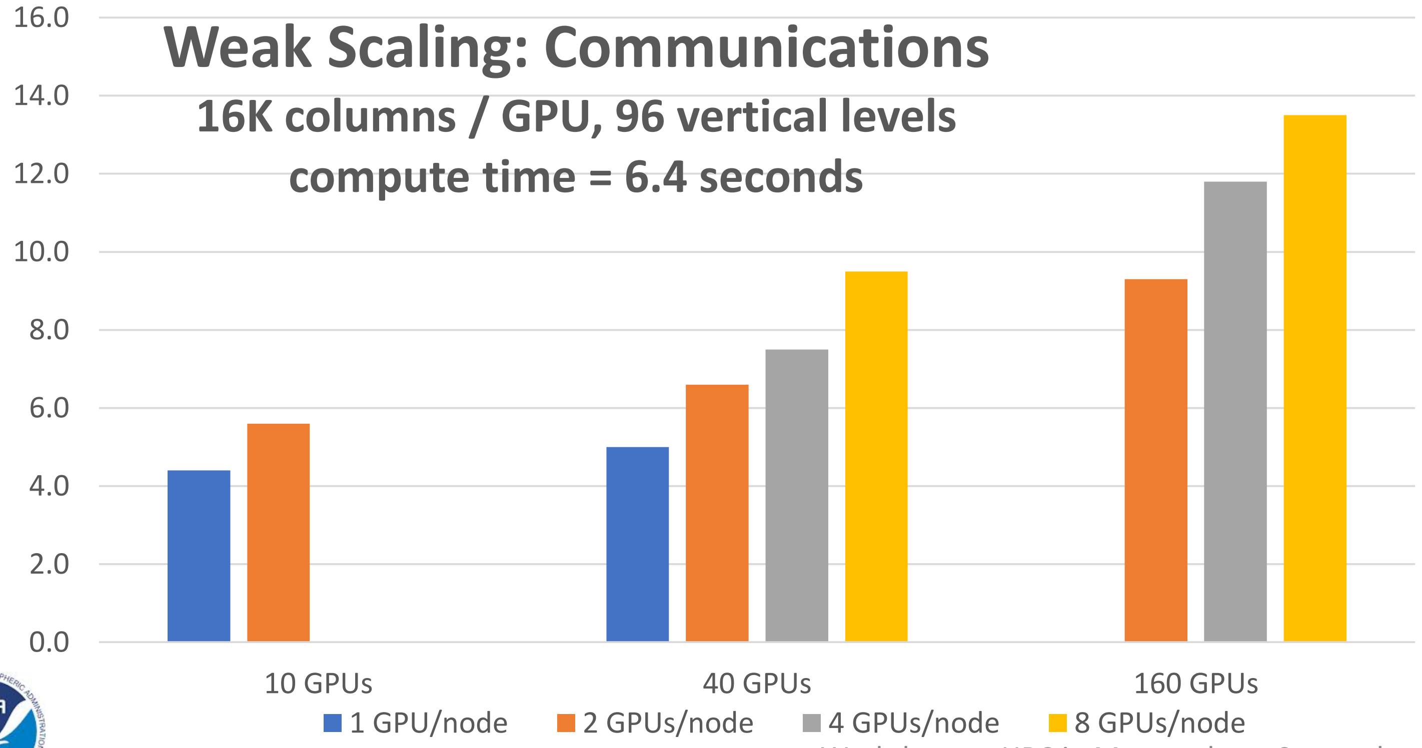
- Fair comparison among numerical schemes
  - Fixed time integration method (Runge-Kutta 4th order)
  - Disable numerical damping
  - Avoid bias from different icos grid generators (NCAR, CSU, ESRL)
  - Same level of MPI optimization

- Compilation and Run time consistency
  - Keeping the same compiler option (on same hardware)

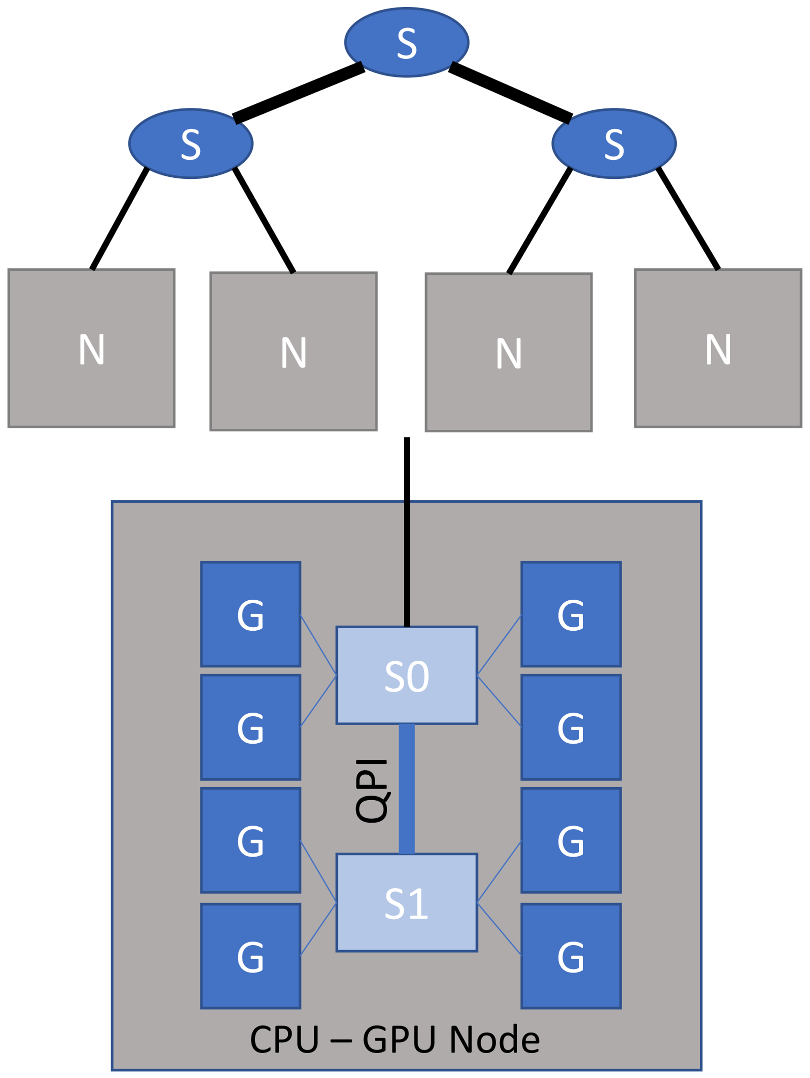


Scalability & Performance

- NOAA GPU System with 800 P100 GPUs
  - 20 core Haswell CPU and 8 P100 GPUs,
  - 100 nodes, mellanox QDR (upgrade?)
- DoE Summit



NOAA GPU System



Data Assimilation

Isidora Jankov, Lidia Trailovic, Chris Harrop

Shallow Water Model for Application in 4DVar

Shallow Water Equations: non-conservative form, simplified

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} = 0$$

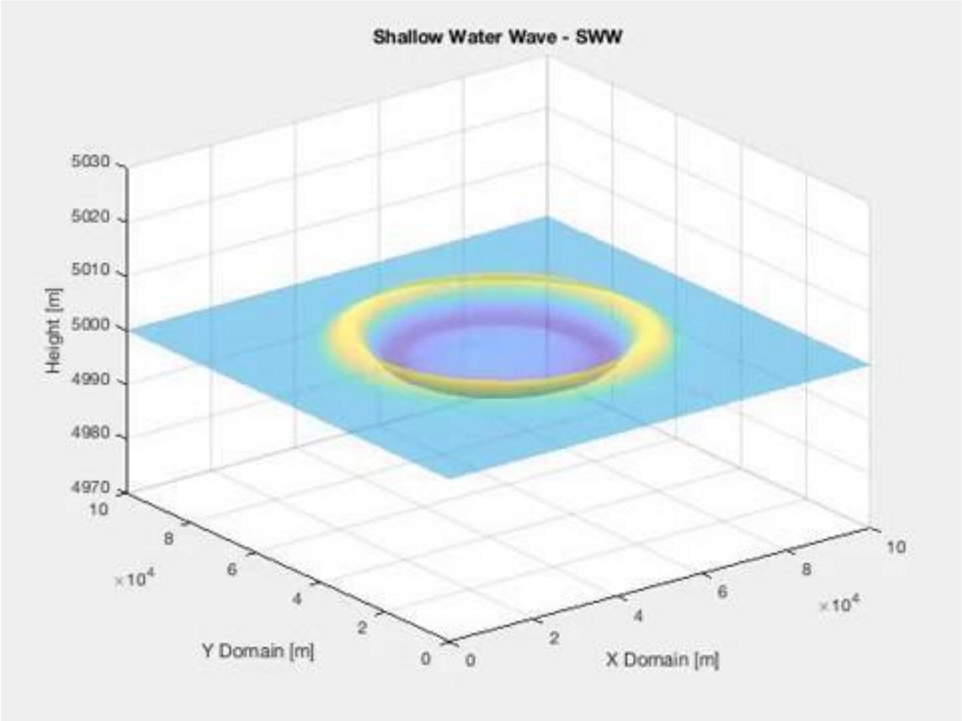
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial h}{\partial x} = 0$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial h}{\partial y} = 0$$

- Square grid, no viscosity, no bottom profile
- Initial condition h0: Gaussian pulse in the middle
- Boundary condition: reflective
- Proof of concept:
- Discretization & Linearization

Space (x,x+dx),(y,y+dy) grid (i,i+1),(j,j+1)

Time (t, t+dt) is (k, k+1)



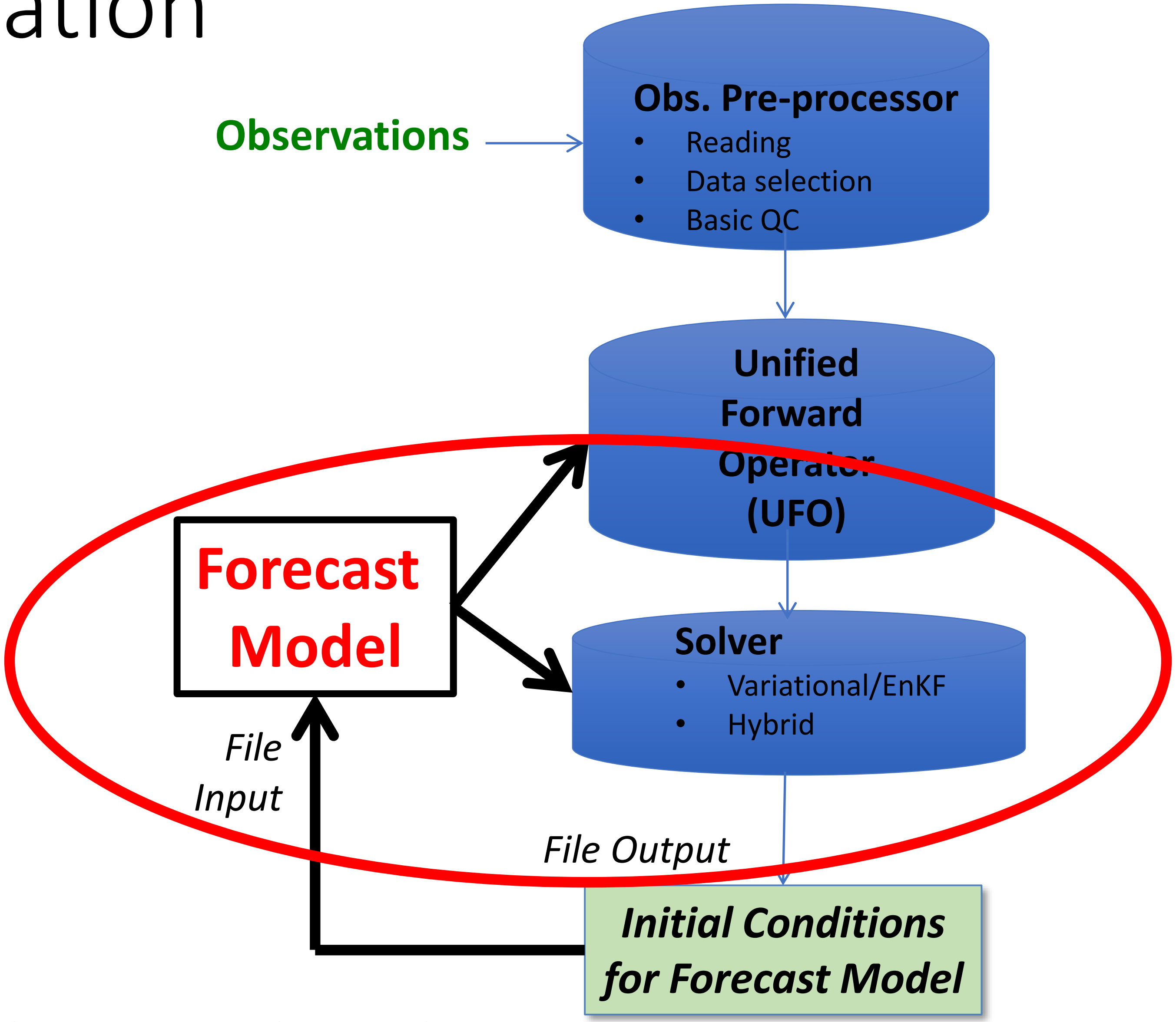
VIdeo: the first 1500 time steps

Next Steps:

- SW model with its TL and Adj will be made available in JEDI as an additional "toy" model (SW will include MPI, which will make it a unique "toy" model)
- Compare performance of code generators for Adj and TL (e.g. Tapanade) and manually produced & optimized models
- Add MPI option for B matrix preconditioning in JEDI and test different flavors of B
- Evaluate impact of various flavors of B and R matrices for application in existing operational systems (e.g. RAP, HRRR and UFS)

JEDI Data Assimilation

- Development
  - CRTM Optimization
  - MPI into JEDI
- Link Model and DA development and evaluation
  - Shallow water
  - More complex models



I/O Developments

Bryan Flynt, Ed Hartnett

AMS, January 6 -10

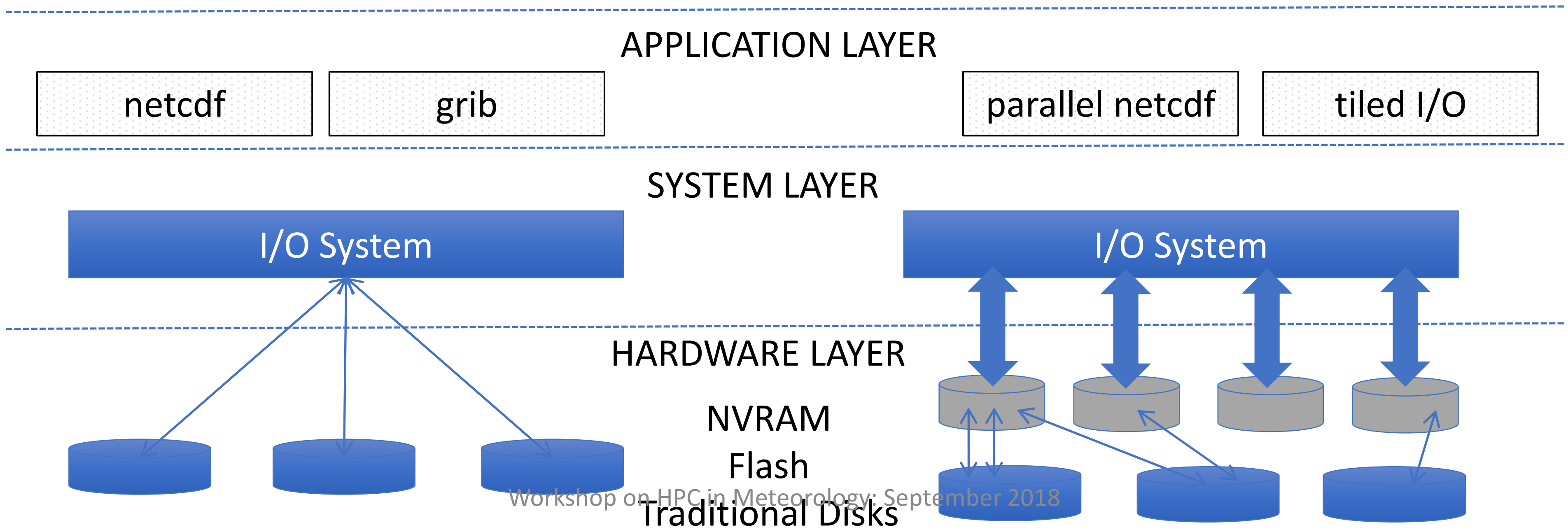
NetCDF-4 Performance Improvements Opening Complex Data Files, Ed Hartnett

Characterizing and Improving Scientific Algorithms and I/O for Exascale: Nimble Dwarfs,

Bryan T. Flynt, Duane Rosenberg, Yonggang Yu and Mark Govett

Scope

- To develop realistic I/O projections for exascale
  - 1 - 3KM global deterministic, 3 – 5 KM ensembles
  - hourly output?
- Test & tune on HPC systems
- Share with vendors, support procurements



Summary

- Exploring ways to more effectively design, develop and run modeling systems on exascale systems
- Models and HPC systems are tremendously complex which limits the ability to develop, run and maintain modeling systems (DA, Model)
- We are exploring a development strategy that strongly links scientific, computational and software development from inception
    - Development team is composed of scientists, software engineers, computer scientists who work **together** to design, build, test, optimize, evaluate, etc.

Scientific Algorithms

Model Code

Optimization

Refactoring

Integration