Lessons Learned on the Development of a Flexible Software Infrastructure for ESMs for Modern Computing Architectures

Frank Giraldo

Department of Applied Mathematics

Naval Postgraduate School

Monterey CA 93943

fxgirald@nps.edu

http://frankgiraldo.wix.com/mysite

Acknowledgements

- **Funded by ONR, AFOSR, NSF, DOE**

- **Parts of this talk include discussions with:**

  - Jeremy Kozdon and Lucas Wilcox (NPS)
  - Michal Kopera (Boise State)
  - Simone Marras (NJIT)
  - Daniel Abdi (TempoQuest)
  - Emil Constantinescu (Argonne National Lab)
  - Andreas Mueller (ECMWF)

Talk Summary

- Caveats
- Lesson Learned via Development of GNuME
- Summary of Lessons Learned
- Where to next

Caveats

- This talk represents opinions based on my own experience building unstructured/adaptive fluids dynamics solvers (3 big codes so far).

- I am an applied mathematician and not a software engineer - although have learned some best practices.

- I am programming language agnostic - use what I need although have mainly used Fortran.

- Lessons learned will be applied to a new collaboration with Caltech, MIT, and JPL on Data-driven ESMs.

Talk Summary

- Caveats
- Lessons Learned via Development of GNuME
- Summary of Lessons Learned
- Where to next

What is GNuME

- GNuME (**G**alerkin **Nu**merical **M**odeling **E**nvironment) is a framework for developing flow solvers. GNuME contains a suite of high-order spatial discretization methods (CG/DG), time-integrators, adaptive mesh refinement, with an emphasis on targeting current and future HPC architectures. GNuME currently houses three components:

    1. NUMA[1] - nonhydrostatic (deep-planet) atmospheric model (global/limited-area)

    2. NUMO - nonhydrostatic ocean model (limited-area)

    3. Shallow water - coastal and global with wetting and drying

- GNuME is written in modern Fortran and C.
    - Fortran used primarily.
    - C used for interfacing with C-libraries and for many-core compute kernels.
- GNuME emerged from the desire to unify all the codes in my group under one umbrella.

[1] NUMA used in U.S. Navy's NEPTUNE NWP system

GNuME Components

p4est[1]
adaptive mesh manager

CG/DG
Element-based Galerkin Methods

Time-Integrators

Elliptic Solvers/
Preconditioners

OCCA[2]

MPI

NUMA
global or local compressible Navier-Stokes equation solver

NUMO
incompressible Navier-Stokes equation solver

SWE
shallow water equation solver

[1] Burstedde et al, SIAM J. Sci. Comput. 5 (2015)

[2] Medina et al, arXiv:1403.0968} (2014)

GNuME Components: Some Lessons

- Separating the core (**red**) components allowed us to unify all 10 codes for maintainability (e.g., various PDEs, 2d, 3d, CG, DG, etc.)

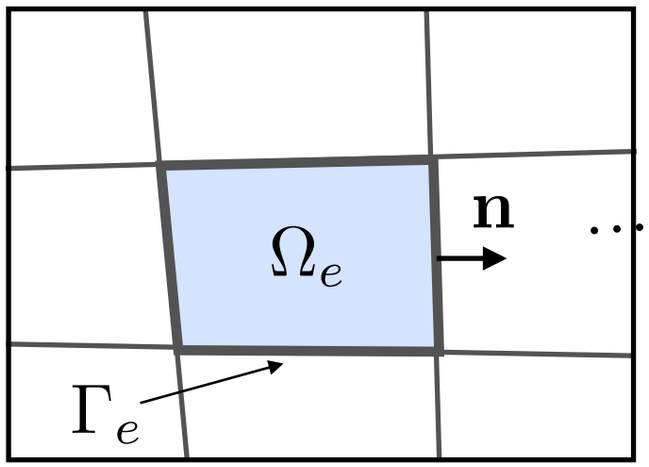
- The communicators (**brown**) are radically different. MPI is quite general (array-based). Many-core is quite specific (e.g., Structure of Arrays -> Array of Structures).

- The solvers (**blue**) are independent and users can add new components safely. Ideally, they should have their own subdirectories and distinct builds.

GNuME Components: CG/DG Numerics

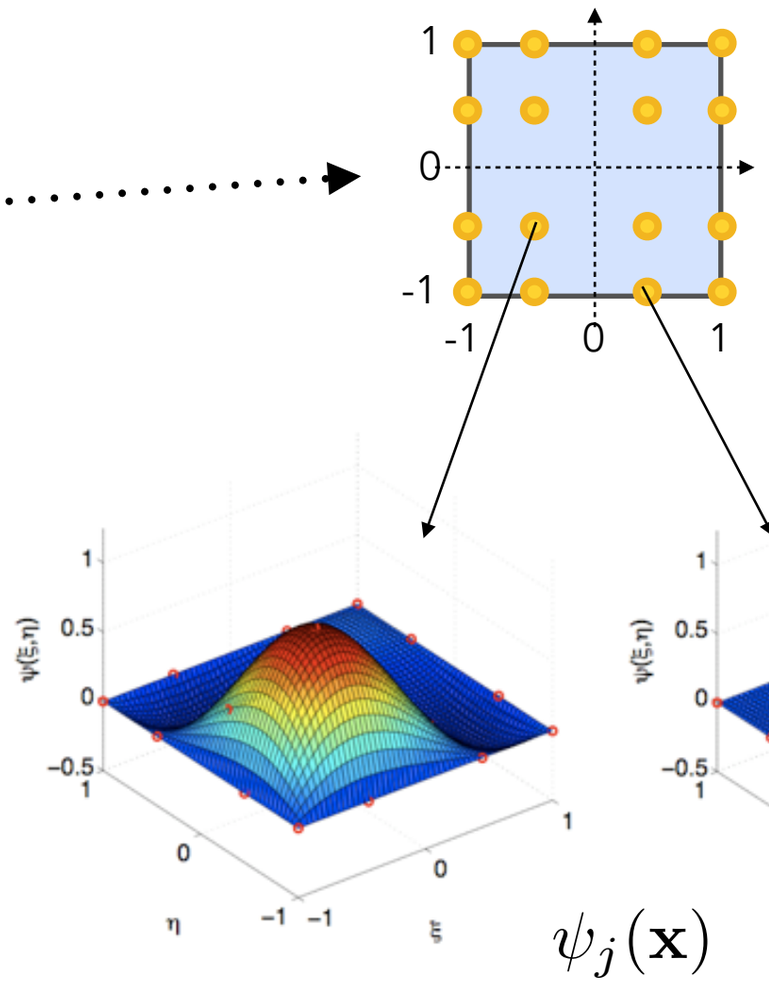
Domain decomposition

$$\Omega = \bigcup_{e=1}^{N_e} \Omega_e$$



Reference element

- Legendre-Gauss-Lobatto points



Basis functions - Lagrange polynomials

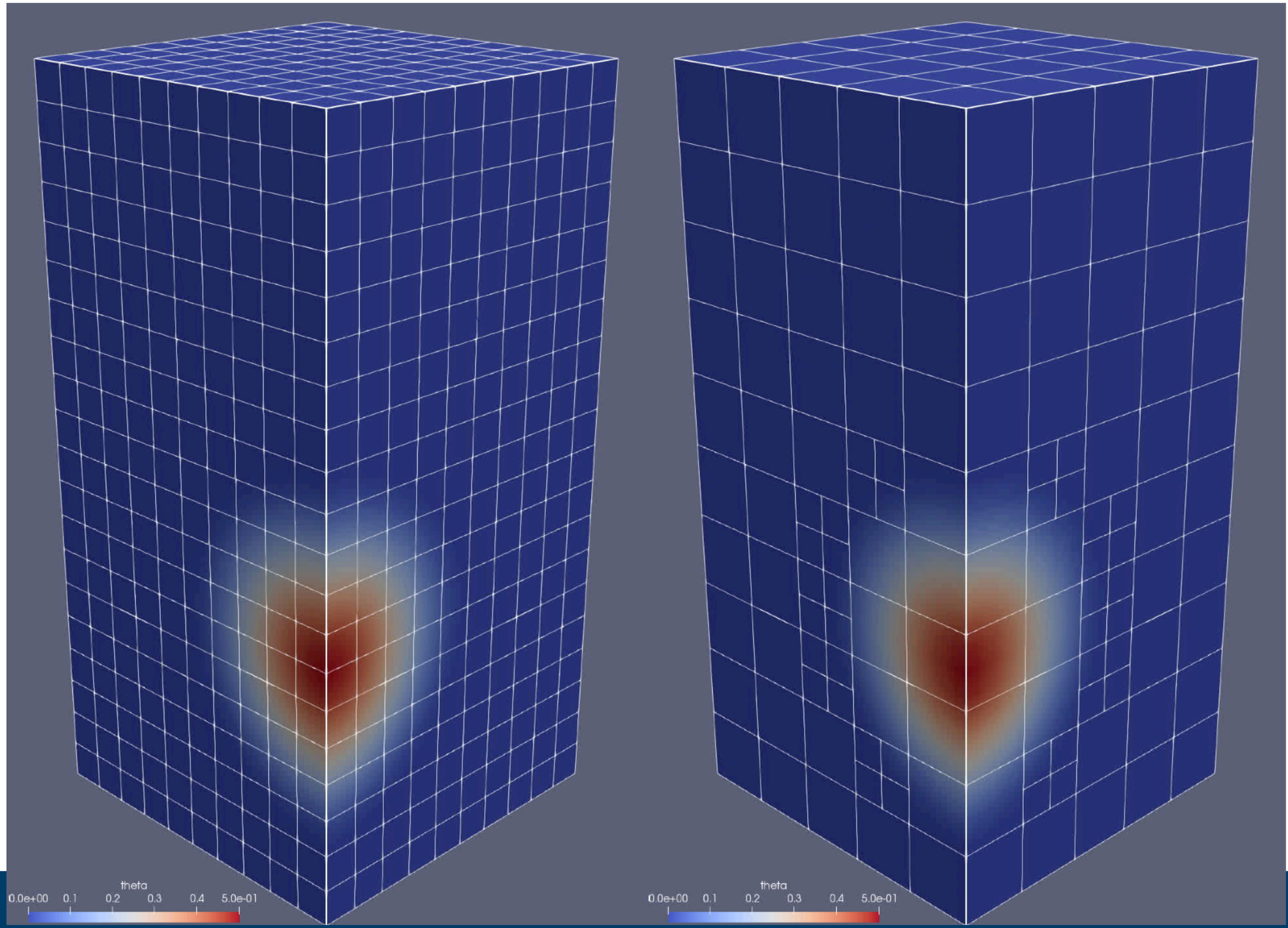
Approximate local solution as:

$$\mathbf{q}_N^{(e)}(\mathbf{x},t) = \sum_{j=1}^{M} \psi_j(\mathbf{x}) \mathbf{q}_j^{(e)}(t)$$

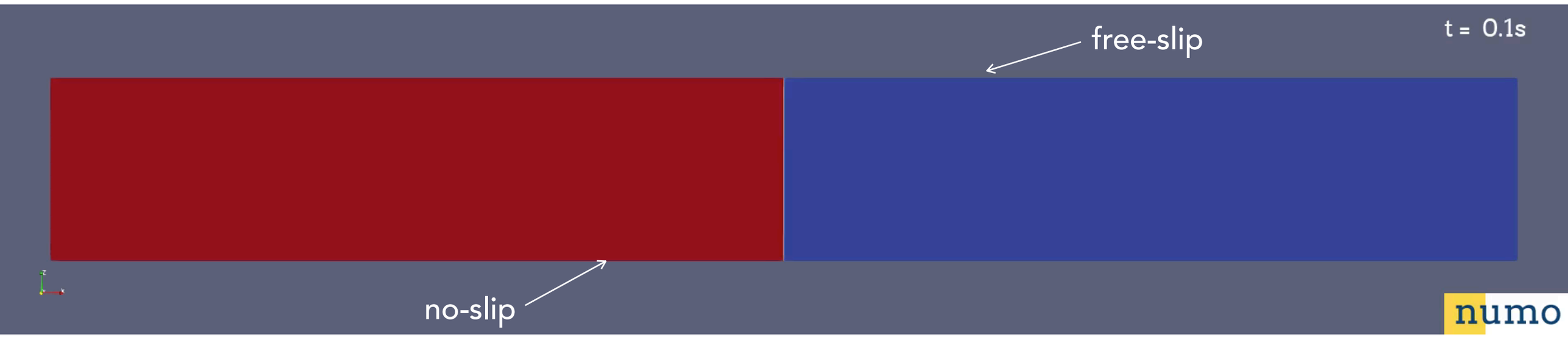
- Bottom line: choose methodologies that extend the shelf-life of the model (e.g., offer new capabilities).

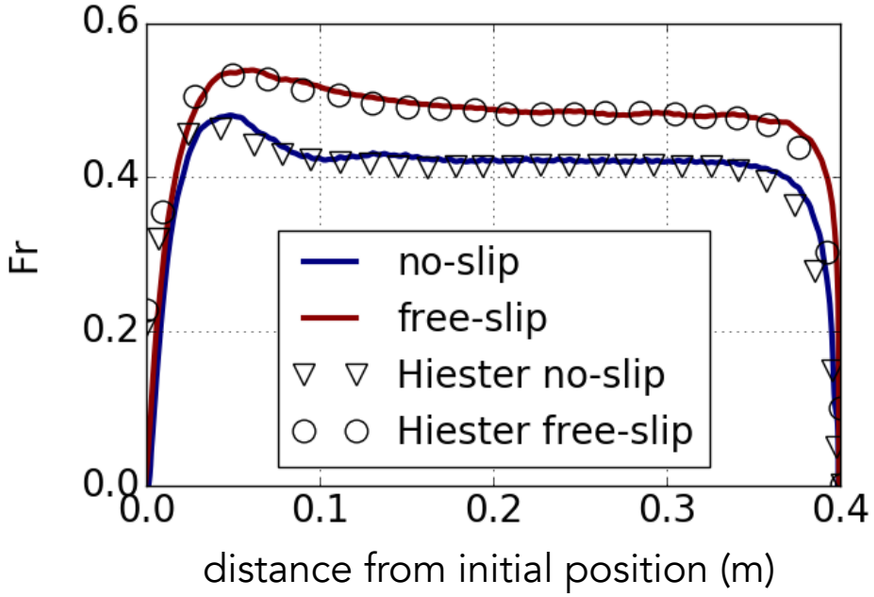
GNuME Numerics: Future Capabilities

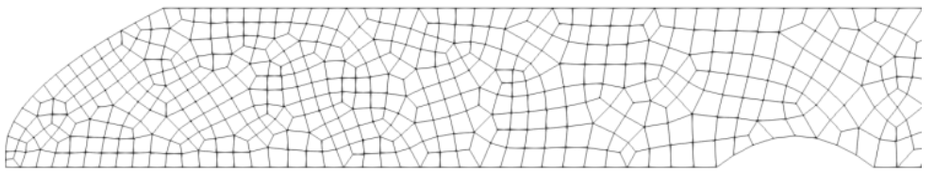
- NUMA with dynamically adaptive mesh refinement: Uniform simulation (left) is 2-4x slower than AMR simulation (right) on 200 processors with 1 million DOF



GNuME Numerics: NUMO Lock Exchange





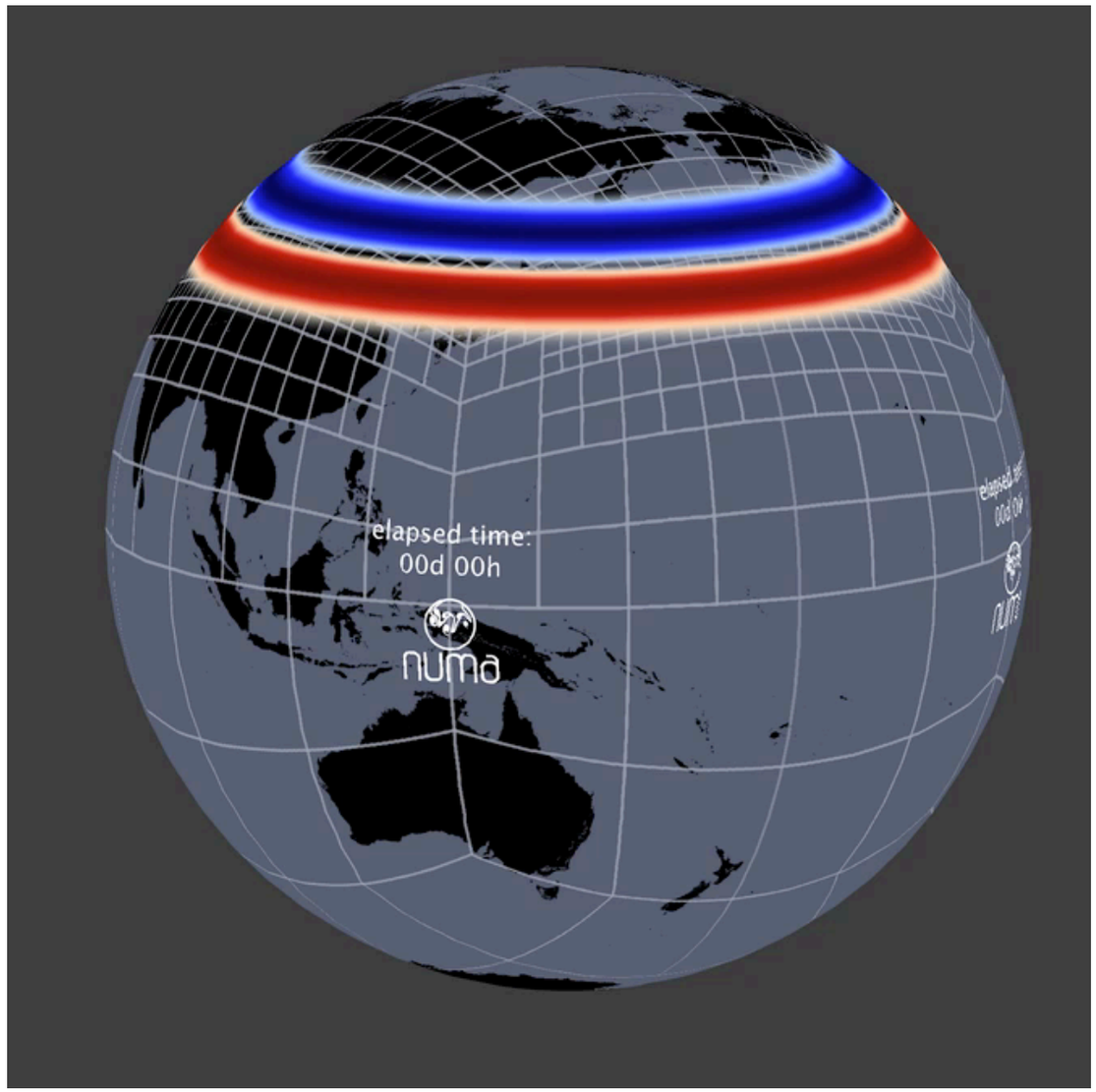


$$Gr = \frac{g\Delta\rho H^3}{\nu^2} = \frac{\text{buoyancy}}{\text{viscosity}} \qquad Sc = \frac{\nu}{\kappa_T} = \frac{\text{viscosity}}{\text{diffusivity}} \qquad Fr = \frac{\mathbf{u}}{\sqrt{\frac{\Delta\rho}{\rho_0} gH}} = \frac{\text{inertia}}{\text{gravity}}$$

		Gr	*Sc*	*Fr*	
				no-slip	free-slip
NUMO	**2D**	**1.25x10⁶**	**6.74**	**0.420**	**0.482**
Hiester et al. (2011)	2D	1.25x10⁶	-	0.417	0.482
Fringer et al. (2006)	2D	1.25x10⁶	-	0.396	0.428
Simpson & Britter (1979)	EXP	4.8x10⁶	~7	0.432	-
NUMO	**2D**	**1.25x10⁶**	**0.71**	**0.407**	**0.475**
Hartel et al. (2000)	2D	1.25x10⁶	0.71	0.406	0.477
Cantero et al. (2007)	3D	1.5x10⁶	0.71	0.407	-

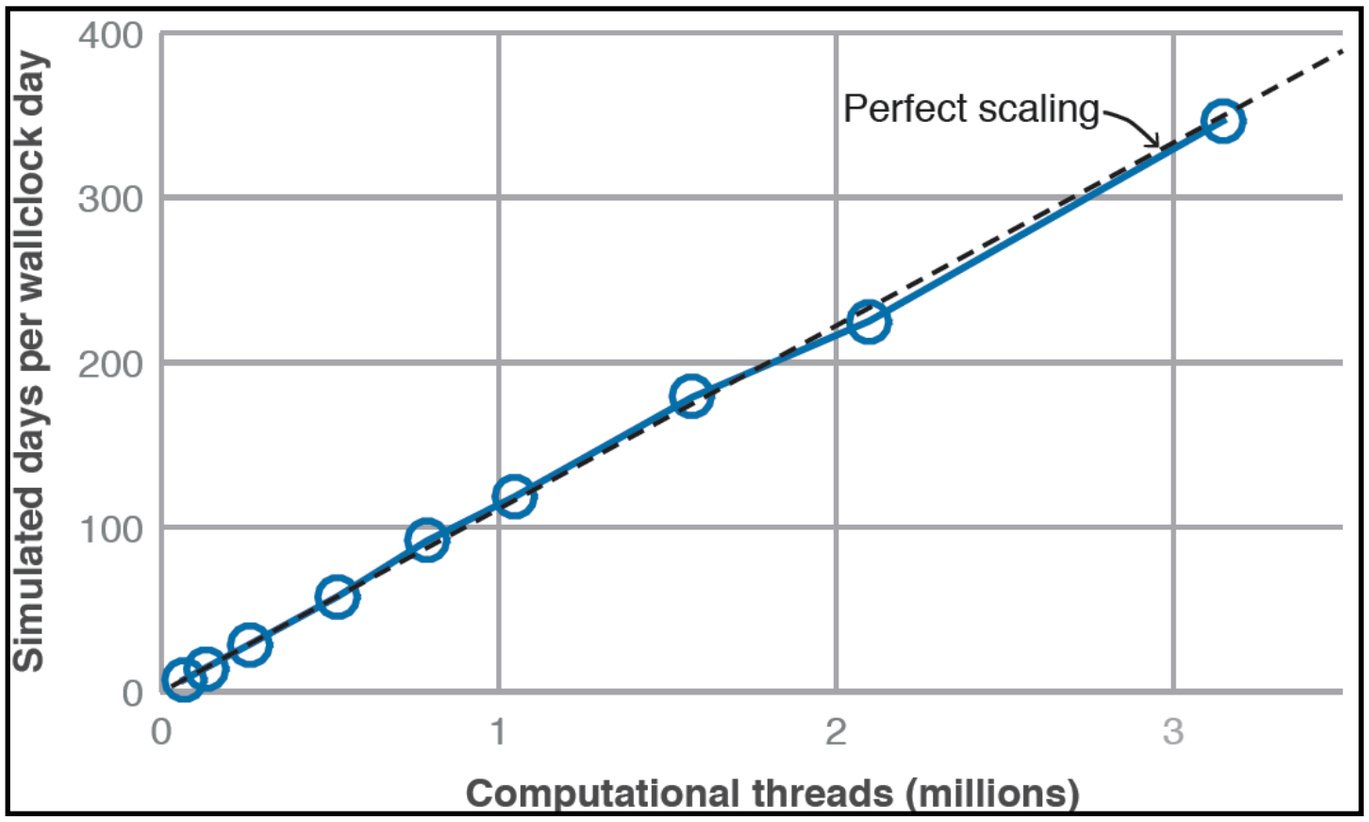
GNuME: Global Shallow Water

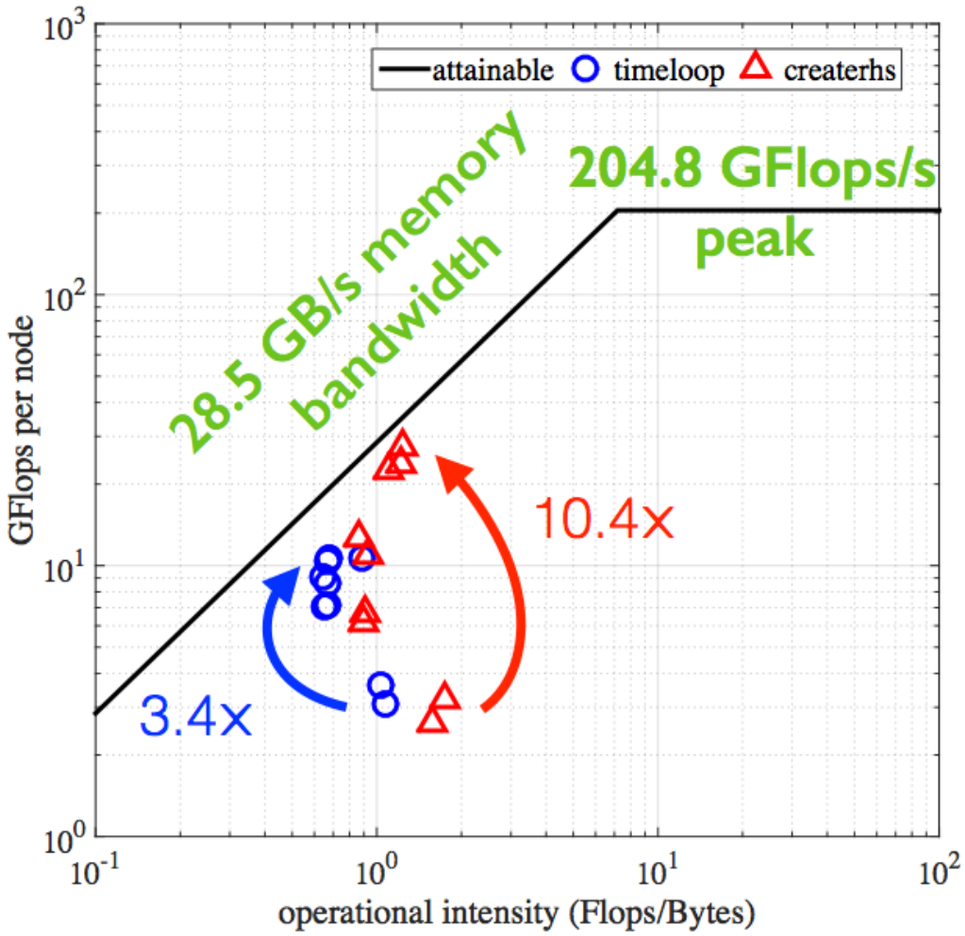
- Geometric flexibility (allows for unstructured/non-orthogonal grids, adaptivity/consistent nesting)
  - allows for focused regions (not just with grid spacing but also order of accuracy)



GNuME: HPC Landscape

- Current ESMs run adequately on CPU-only computers (X86).
  - E.g., NUMA on Mira at 3 km global resolution (dynamics only)
  - Mira (ALCF) is an IBM BG/Q with 786,432 cores each with 4 threads with a peak performance of 10 petaflops.





GNuME: HPC Landscape

- Current HPC Landscape: it has changed to hybrid

Rank	Name	Theoretical Peak PFlops	# of Cores	Hardware	Country
1	Sunway TaihuLight	125	10,649,600	260-core manycore	China
2	Tianhe-2	54	3,120,000	5-core Intel Xeon Phi (KNC)	China
3	Piz-Daint	25	361,760	Intel Xeon Phi+Nvidia P100	Switzerland
4	Titan	27	560,640	AMD Opteron + Nvidia K20	U.S.
5	Sequoia	27	1,572,864	IBM BG/Q	U.S.

[top500.org. June 2017]

GNuME: HPC Landscape

- Near Future HPC Landscape: it has changed to hybrid

Rank	Name	Theoretical Peak PFlops	# of Cores	Hardware	Country
1	Aurora	1000+	?	Not KNH	U.S.
2	Summit	207	230,000	IBM Power 9 + Nvidia Volta GPUs	U.S.
3	Sunway TaihuLight	125	10,649,600	260-core manycore	China
4	Tianhe-2	54	3,120,000	5-core Intel Xeon Phi (KNC)	China
5	Piz-Daint	25	361,760	Intel Xeon Phi+Nvidia P100	Switzerland
6	Titan	27	560,640	AMD Opteron + Nvidia K20	U.S.
7	Sequoia	27	1,572,864	IBM BG/Q	U.S.

GNuME: HPC Landscape

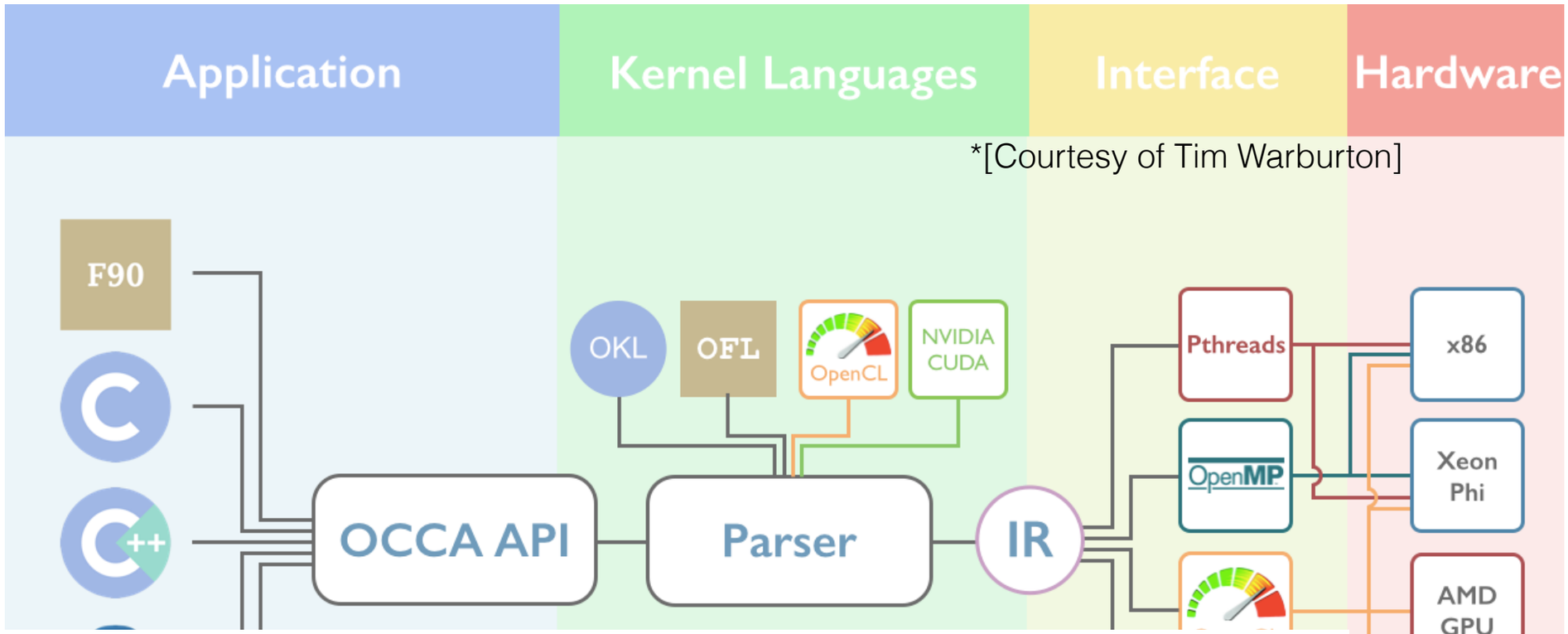
- Many APIs exist for accessing manycore hardware: OpenACC, OpenMP, CUDA, OpenCL, Kokkos, OCCA.

- As a domain scientist, perhaps easier to use DSLs (e.g., Gridtools, etc.).

- Our strategy has been to control as much as possible and deal directly with arrays (array of structures).

GNuME: HPC Landscape

- How to harness different hardware
  - In the Hardware-agnostic approach, the idea is to write the compute-kernels once and offload them to different back-ends for varying architectures (figure shows the idea). This solution offers portability but no guarantee on performance.
  - The performance comes from tuning at the Kernel Language level (code snippet below).



*[Courtesy of Tim Warburton]

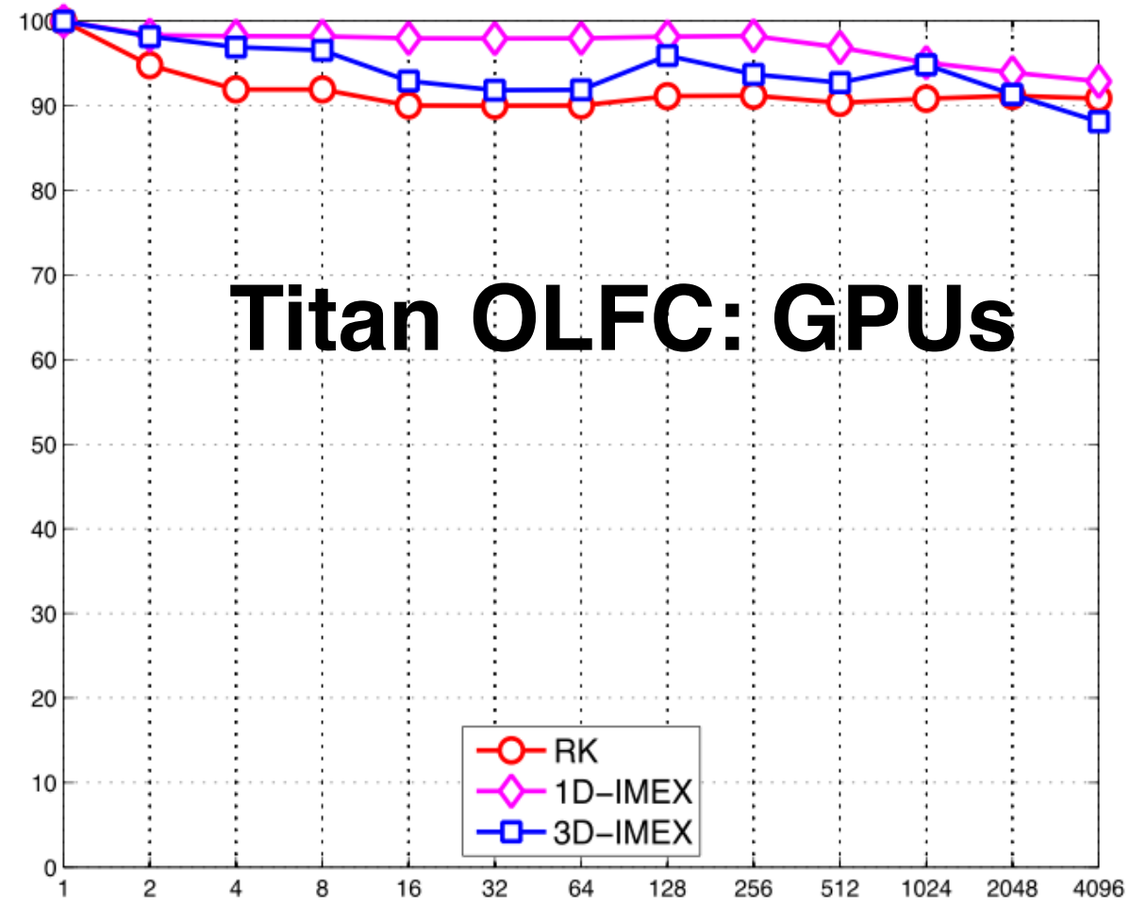
```
//--- OpenMP mode
#define myInnerFor  occaInnerFor

#if OCCA_USING_OPENMP

#ifdef Nthreads
#define PARALLEL_THREADS(x) omp parallel for num_threads(x) firstprivate(occaInnerId0,occaInnerId1,occaInnerId2)

#define myParallelFor     OCCA_PRAGMA(STRINGIZE(PARALLEL_THREADS(Nthreads)))
#else
#define myParallelFor     occaParallelFor
#endif
```

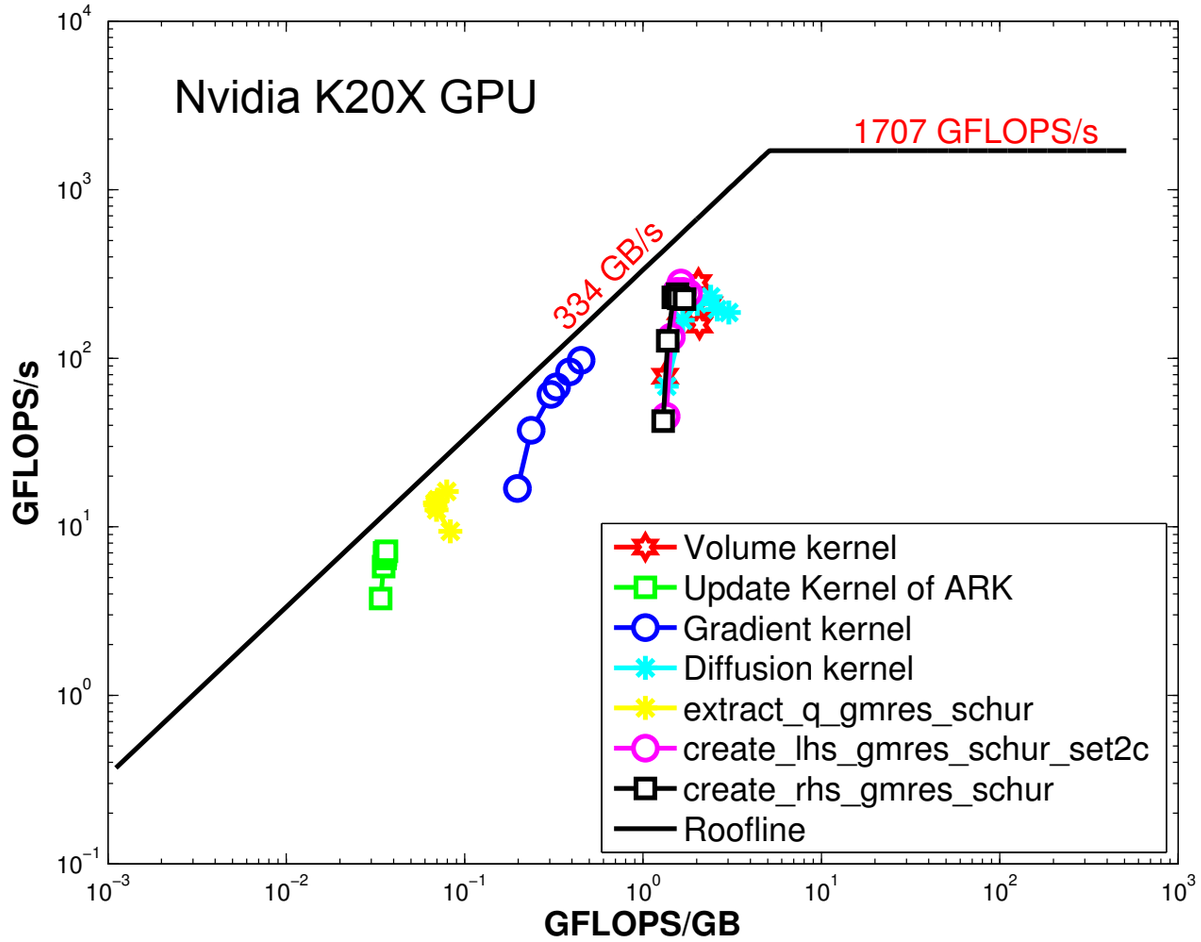
OpenMP

CUDA

GNuME: HPC Landscape

- How to access different hardware
  - Example of NUMA hardware-agnostic approach (with Occa[1]) on GPUs and Intel Xeon Phis. Same code-base achieves 90% (weak) scaling efficiency.
  - On GPU, we use a node-per-thread approach whereas on many-core an element-per thread approach (via myParallelFor)





Results with NUMA from [Abdi et al. IJHPCA 2017]

[1] Medina et al, arXiv:1403.0968} (2014)

Talk Summary

- Caveats
- Lessons Learned via Development of GNuME
- Summary of Lessons Learned
- Where to next

Summary

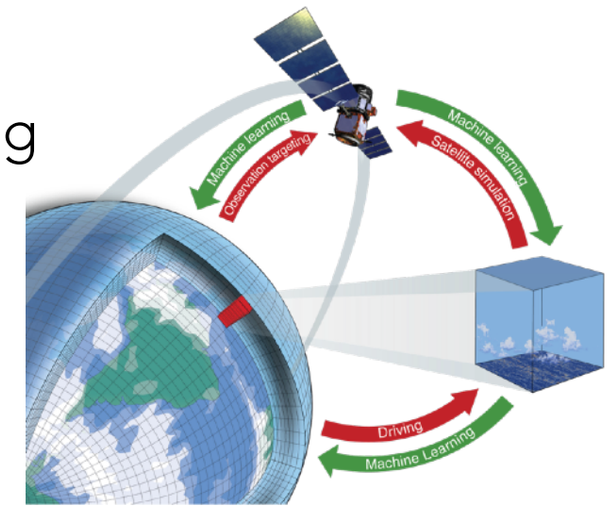
- Our driving principle has been simplicity (readability) over performance.
- Have also chosen approaches that extend the shelf-life of the model (e.g., numerics, hardware-agnosticism, etc.).
- Balance must be struck between rewriting code (a good thing) and reusing code as much as possible.
- Choice of Programming Language is important but a balance must be struck between maintainability (developer) and readability (user). Ideally would like one language for both prototyping and deployment (e.g., Firedrake, Julia).
- For OpenSource, need a simple enough language with a large user-base in ESM community.
- DSLs can insulate domain scientists from much of the software engineering complexities and speed-up progress (my group has not tried this approach, yet).

Talk Summary

- Caveats
- Development of GNuME
- Summary of Lessons Learned
- Where to next

Where to Next

- In collaboration with Caltech (Andrew Stuart's talk), MIT (Raffaele Ferrari's talk), and JPL, we will develop an infrastructure for running both global atmosphere and a multitude of hi-res LES domains (with data) for teaching the global model.



- To do so, we will redesign the code in a hardware agnostic approach (e.g., OCCA), with compute-kernels written in a variety of languages (use existing functions), managed by a Python/Julia workflow.

- Open source from the beginning (e.g., via Github).

- Numerical methods will stay the same (e.g., element-based Galerkin methods, time-integrators, non-conforming grids).

- Create a software infrastructure that allows for specific components (e.g., ESM vs CFD) with specific builds. To make it easier for collaborators to add components and streamline the workflow and increase (work) performance.