Sandia National Laboratories

SAND2016-5044C

Performance Portable Assembly For Plasma Fluid Equations

Matthew Bettencourt, Eric Cyr, Richard Kramer, Roger Pawlowski, Edward Phillips, Alan Robinson and John Shadid
Sandia National Laboratories

ECCOMAS 2016

Tuesday June 7th, 2016

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

National Nuclear Security Administration

Problem Description

Prediction of plasma processes at moderate to high densities

- Multifluid plasma equations – Electrons, ions and neutrals
  - Six-dimensional Boltzmann equations too high of dimensionality
  - Particle methods too expensive due to plasma frequency
  - Sets of 5 or 13 coupled PDEs with stiff source terms need to be solved
- Equations need to be composible and maintainable

Analysis Beyond Forward Simulation

- Forward solves are not enough – we want to explore complex solution spaces:
  - Simultaneous analysis and design adds requirements (typically sensitivities)
  - Do not burden analysts/physics experts with analysis algorithm requirements: i.e. programming sensitivities for implicit solvers, optimization, stability, bifurcation analysis and UQ

Engine must be flexible, extensible, maintainable and EFFICIENT!

Problem Description

Prediction of plasma processes at moderate to high densities

- Multifluid plasma equations – Electrons, ions and neutrals
  - Six-dimensional Boltzmann equations too high of dimensionality

Directed Acyclic Graph-based Assembly

  - Sets of 5 or 13 coupled PDEs with stiff source terms need to be solved
- Equations need to be composible and maintainable

Analysis Beyond Forward Simulation

- Forward solves are not enough – we want to explore complex solution spaces:
  - Simultaneous analysis and design adds requirements (typically sensitivities)
  - Do not burden analysts/physics experts with analysis algorithm

Template-based Generic Programming

Engine must be flexible, extensible, maintainable and EFFICIENT!

Sandia National Laboratories

Drekar/Panzer: Algorithms and Software

Drekar: Algorithms research application

- Targets coupled multi-physics
- Large scale simulation (>100k cores)
- Advanced algorithm demonstration

A co-dependent development relationship

Panzer: Multi-physics assembly engine

- Finite element focused (currently)
- Embedded analysis (AD, Sensitivities)
- Technology sharing and deployment

Drekar drives **Panzer** requirements and design goals

Panzer provides **Drekar** flexible infrastructure and core technologies

John Shadid
Roger Pawlowski
Eric Cyr
Edward Phillips
Tim Wildey

Tom Smith
Paul Lin
David Sondak
Paula Weber
Richard Kramer

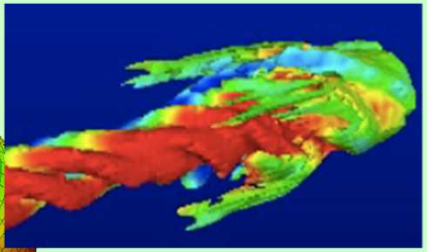
Allen Robinson
Matt Bettencourt
Sidafa Conde
Ben Seefeldt
Chris Siefert

Andrew Bradley
Greg von Winckel
David Hensinger

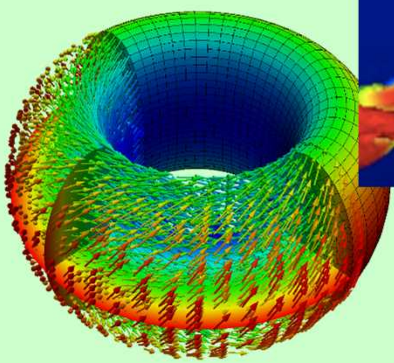
Sandia National Laboratories

Drekar/Panzer: Capabilities

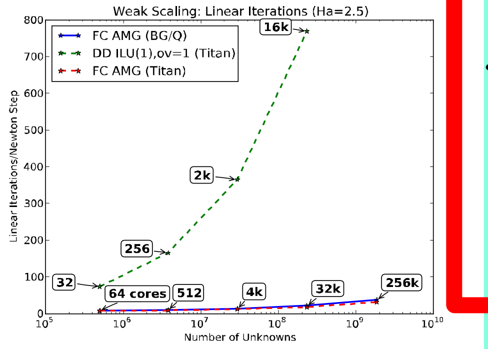
Applications



Turbulent CFD



Magnetohydrodynamics



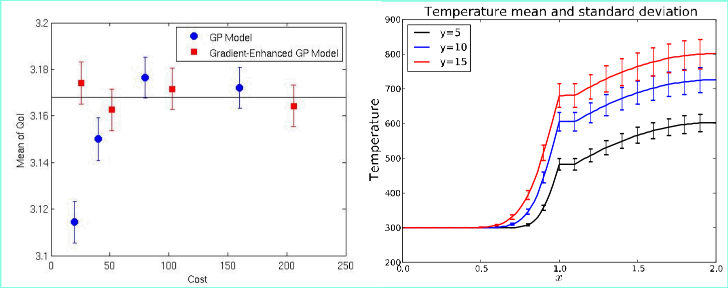
Algebraic Multigrid (>100k cores)

$$\mathcal{A} = \begin{bmatrix} I & \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & B^T \\ & S \end{bmatrix}$$

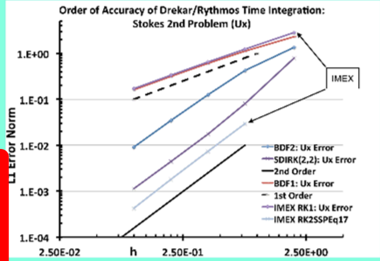
$$S = C - BF^{-1}B^T$$

Block Preconditioning

Discretizations & Algorithms

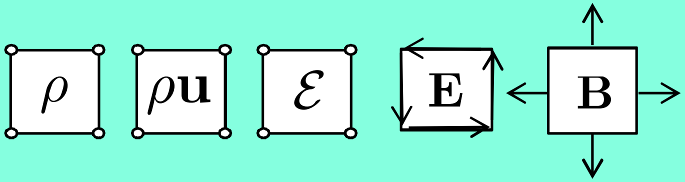


Uncertainty Quantification



IMEX

PDE Constrained Optimization

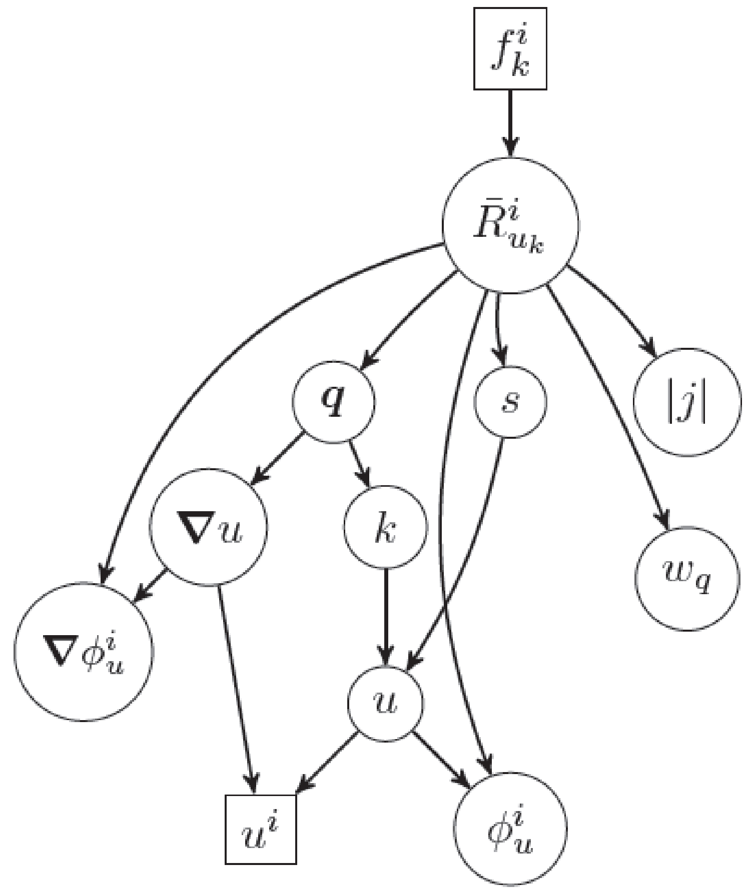


Compatible Discretizations

Lightweight DAG-based Expression Evaluation

- Decompose a complex model into a graph of simple kernels (functors)
  - Decomposition is NOT unique

$$R_u^i = \int_\Omega \left[\phi_u^i \dot{u} - \boldsymbol{\nabla} \phi_u^i \cdot \boldsymbol{q} + \phi_u^i s \right] \mathrm{d}\Omega$$



- Supports rapid development, separation of concerns and extensibility.

- A node in the graph evaluates one or more **fields**:
  - Declare fields to evaluate
  - Declare dependent fields
  - Function to perform evaluation

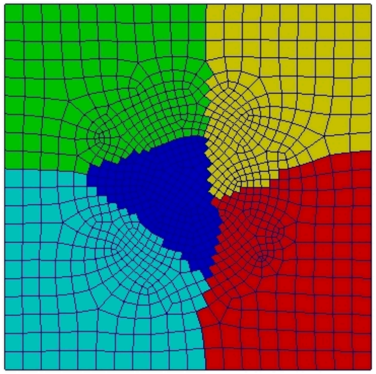
- Separation of data (Fields) and kernels (Expressions) that operate on the data
  - Fields are accessed via multidimensional array interface (shards or kokkos)

dia National Laboratories

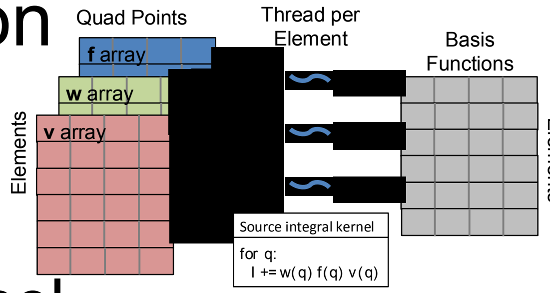
DAG Evaluation

- The problem domain is broken into worksets
  - Smaller number of elements which fit into cache or other memory structures
  - The entire DAG is evaluated for a single workset
  - Each math kernel can be threaded over the workset elements
- Worksets are filled with a gather operation
- The workset’s contributions are accrued in a matrix with a scatter operation
- All intermediate calculations are independent of if they are computing a Jacobian or residual or sensitivity

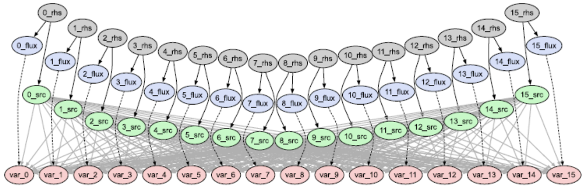
Parallelization Strategies



- MPI – Domain decomposition
  - This is the traditional approach
  - Not covered in this talk
- Threading – On node parallelization
  - More scalable for local cores
  - Extensible to GPUs/Cuda
  - Threads team up to accelerate a kernel

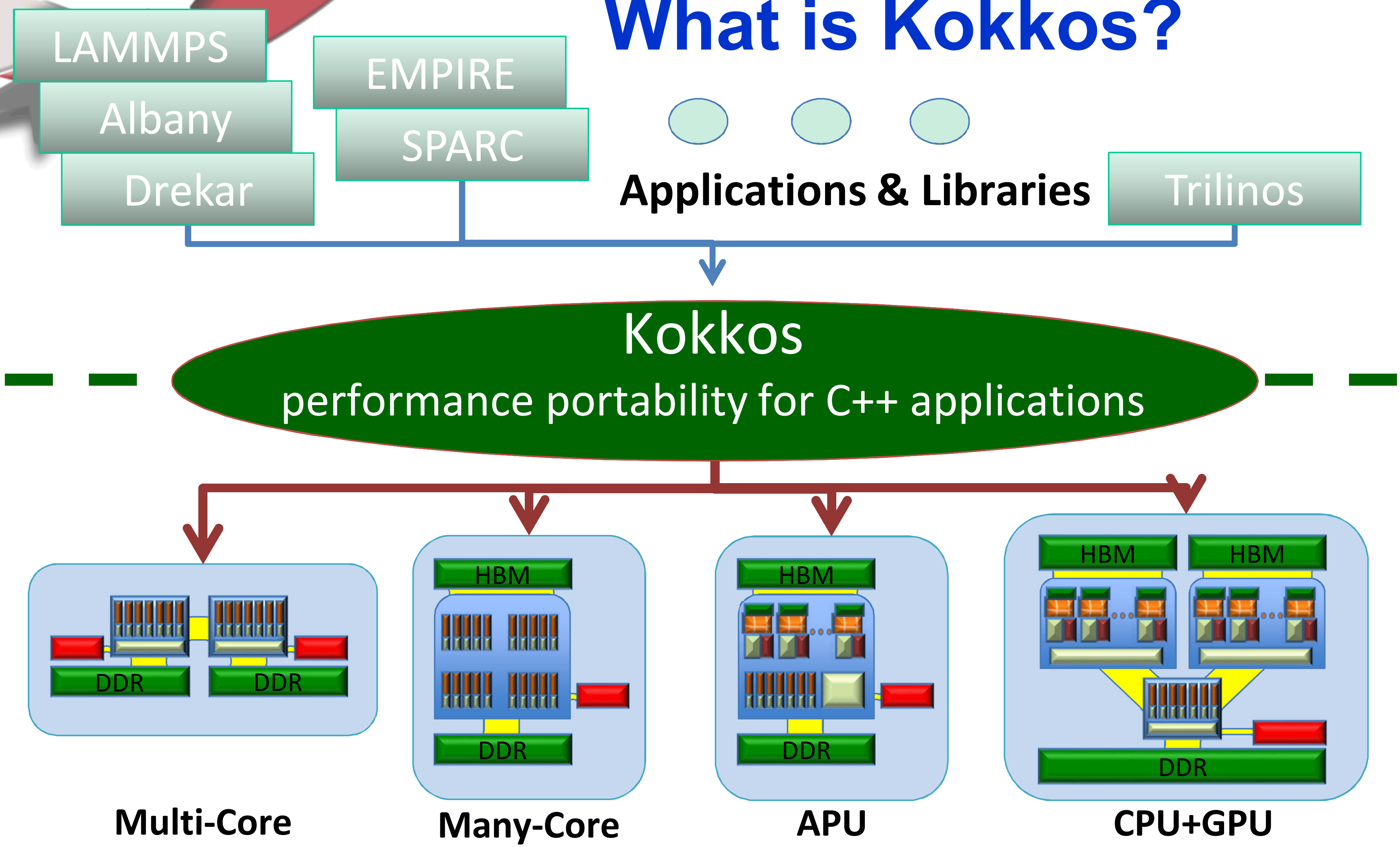


- Kernel parallelization
  - Different kernels at the same time
  - Asynchronous Many Tasking (AMT)



Sandia National Laboratories

What is Kokkos?



Cornerstone for performance portability across next generation HPC architectures at multiple DOE laboratories, and other organizations.

Sandia National Laboratories

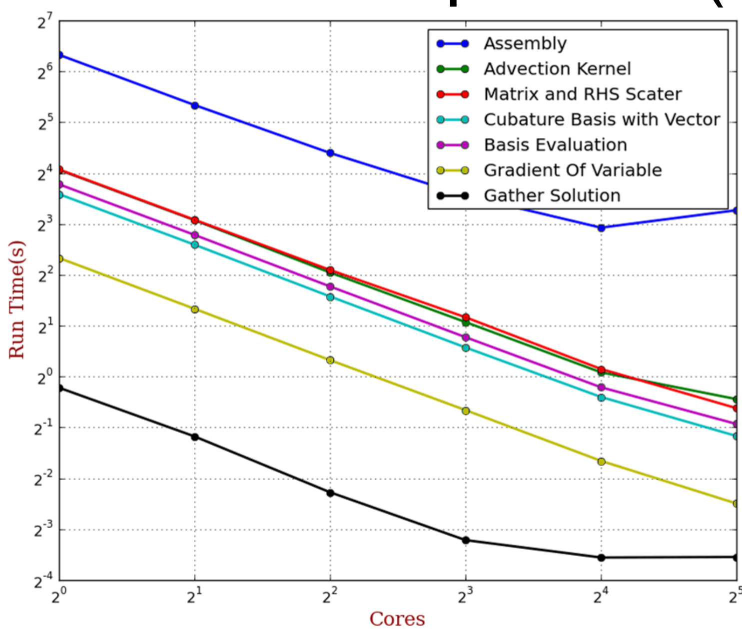
Abstractions
Patterns, Policies, and Spaces

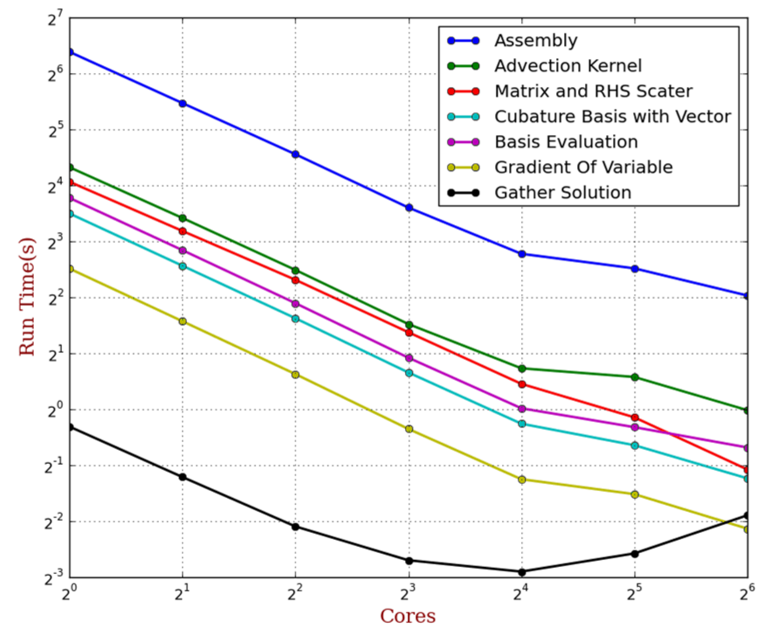
- **Parallel Pattern** of user's computations
  - parallel_for, parallel_reduce, parallel_scan, task-graph, ... *(extensible)*
- **Execution Policy** tells *how* user computation will execute
  - Static scheduling, dynamic scheduling, thread-teams, ... *(extensible)*
- **Execution Space** tells *where* computations will execute
  - Which cores, numa region, GPU, ... *(extensible)*
- **Memory Space** tells *where* user data resides
  - Host memory, GPU memory, high bandwidth memory, ... *(extensible)*
- **Layout** (policy) tells *how* user array data is laid out
  - Row-major, column-major, array-of-struct, struct-of-array … *(extensible)*
- Differentiating: Layout and Memory Space
  - Versus other programming models (OpenMP, OpenACC, …)
  - Critical for performance portability …

Sandia National Laboratories

Threading
Traditional Architectures

- The gather, math routines and scatter operations were all threaded
- MPI only (left) was compared to Kokkos::OpenMP (right) on 16000 elements



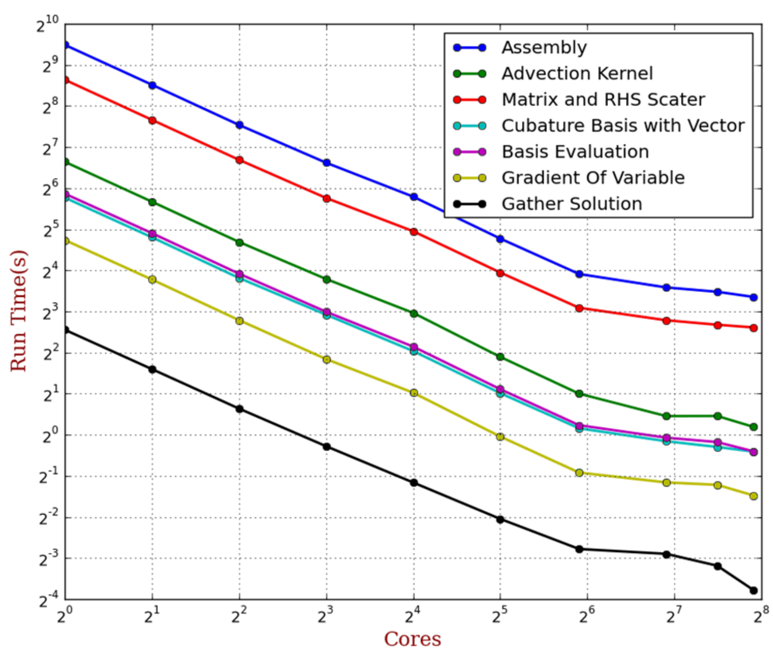


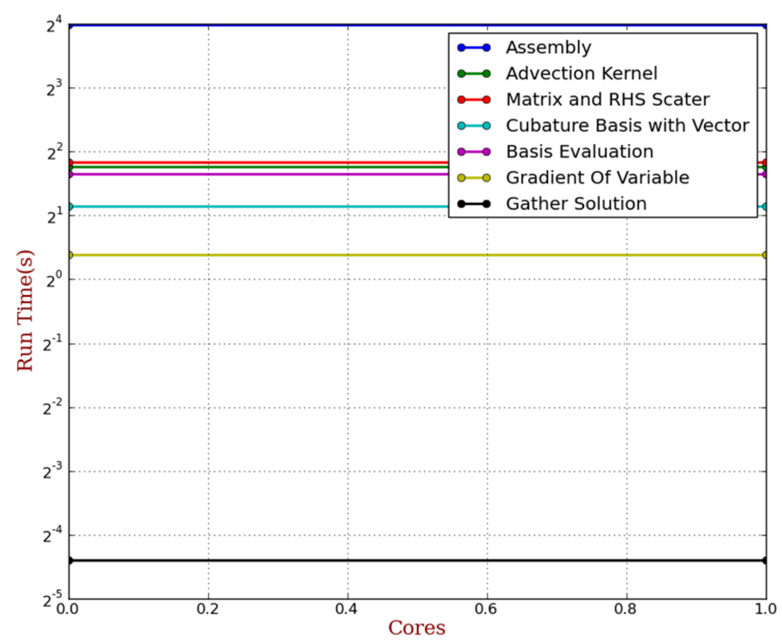
Run on a dual 16 core Haswell with hyperthreading. Left result uses one socket for 1 to 32 cores and MPI across sockets. MPI results show the average time across MPI ranks. 156k total degrees of freedom. Time is for Jacobian assembly only.

National Laboratories

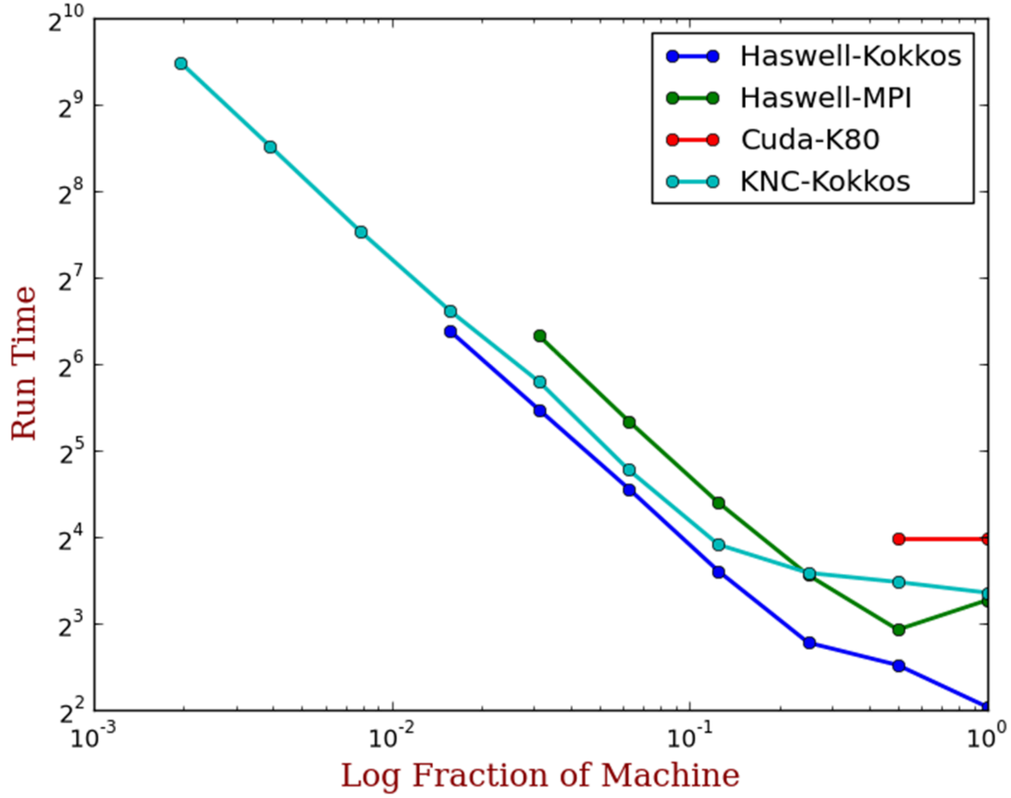
Threading
Modern Architectures

- Kokkos allows one to migrate to new machines and backends
  - Only recompilation is required to move to Intel's Knights Corner (left) or Nvidia K80 via Cuda (right)





Summary
Threading Results

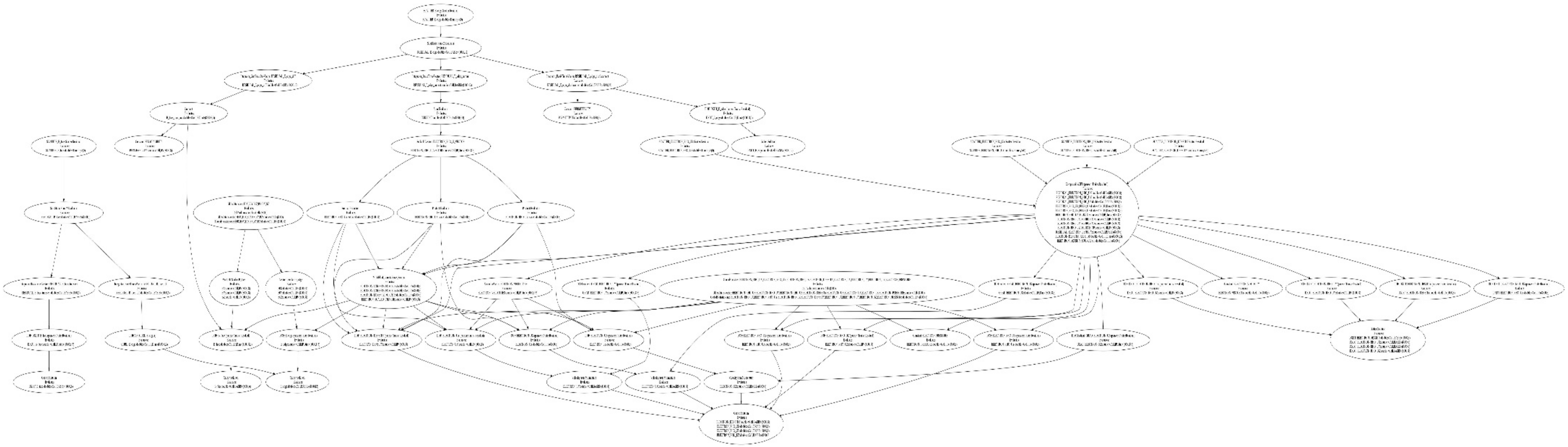


Beyond Threading

- Threading accelerates each individual math kernel
  - Eventually there is insufficient work in a single kernel to use all the threads efficiently
- Task parallelism
  - Chunks of threads solve different kernels
  - The wider the graph, the more task parallelism
  - Asynchronous Many Task (AMT) is an active research area in scheduling these tasks
    - HPX, DHARMA, Charm++, Legion, …

Task Parallelism
Single Fluid

Real DAG much more complex than the previous toy example



Theoretical speed up for this DAG

	1 Thread/Kernel	8 Threads/Kernel	16 Threads/Kernel
Jacobian	3.5	4.5	4.9
Residual	3.4	3.4	3.5

Summary
Next Steps

- Assembly performance can be improved with both threading and task parallelism
  - Threading shows up-to 64x speed up
  - Need to demonstrate these two approaches together
- Cuda parallelism performance lower than expected
  - Not enough parallelism per workset
    - Need bigger workset, however, limited memory
  - Need to implement hierarchical parallelism
- Kokkos makes thread parallelism easier
  - Need to incorporate Kokkos further