Solving Petascale Turbulent Combustion Problems with the Uintah Software

Martin Berzins

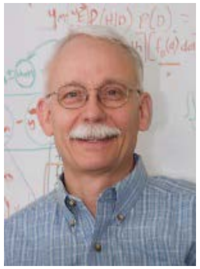
DOE NNSA PSAAP2 Center

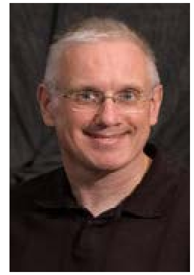
Thanks to DOE ASCI (97-10), NSF , DOE NETL+NNSA, NSF , INCITE, XSEDE, ALCC, ORNL, ALCF for funding and cpu hours

This work is part of our NNSA PSSAP2 Center using INCITE + ALCC awards

Part of Utah PSAAP Center

Phil Smith(PI) Dave Pershing MB





NSF RESILIENCE

Sahithi Chaganti Aditya Pakki

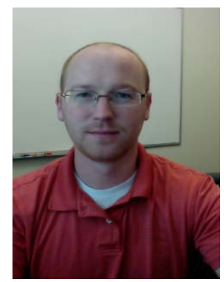
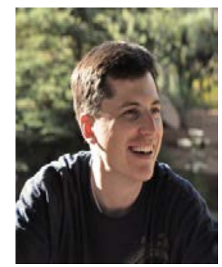

PSAAP2 Applications Team

Todd Harman Jeremy Thornock Derek Harris Ben Issac

PSAAP DSL Team

James Sutherland Tony Saad



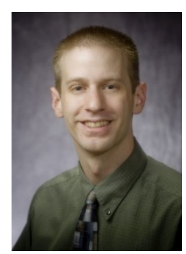
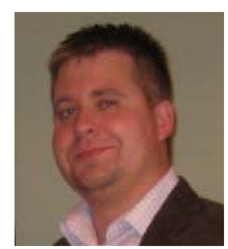

PSAAP Extreme Scaling team

John Schmidt Alan Humphrey John Holmen

SANDIA

Brad Peterson Dan Sunderland



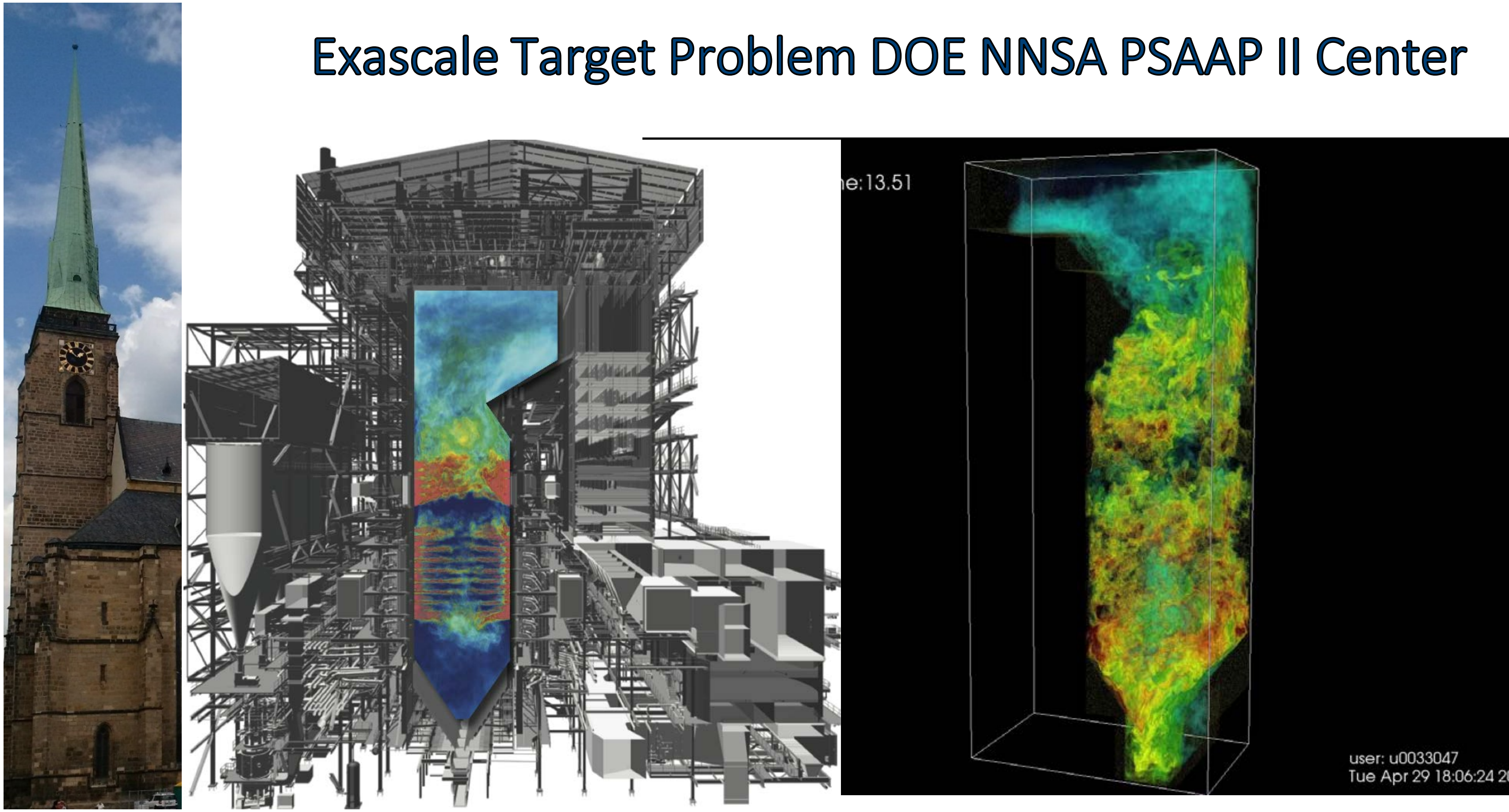




Seven abstractions for applications post- petascale

1.A task-based formulation of problems at scale
PSAAP GE/Alstom Clean Coal Boiler
2. A programming model to write these tasks as code Uintah tasks specify halos; Read from /Write to local data warehouse
3.A runtime system to execute these tasks
Uintah Runtime System continues to evolve
4. A low-level portability layer to allow tasks to run on different architectures Kokkos
5.Domain specific language to ease problem solving
Nebo Wasatch (not discussed here)
6 A Resilience model AMR based duplication
7. Scalable components I/O, in-situ Vis, Solvers PIDX, Visit, hypre.

Exascale Target Problem DOE NNSA PSAAP II Center

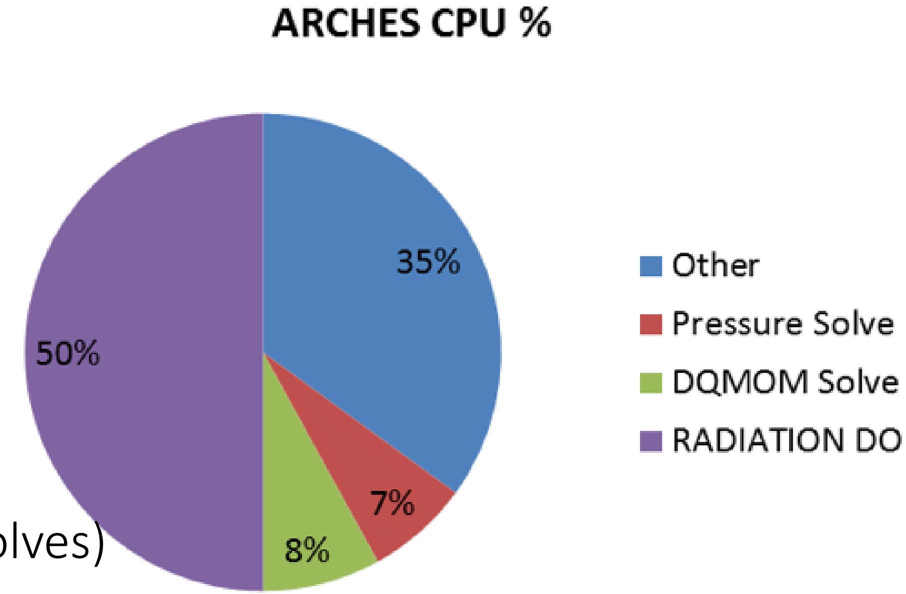
92 meters



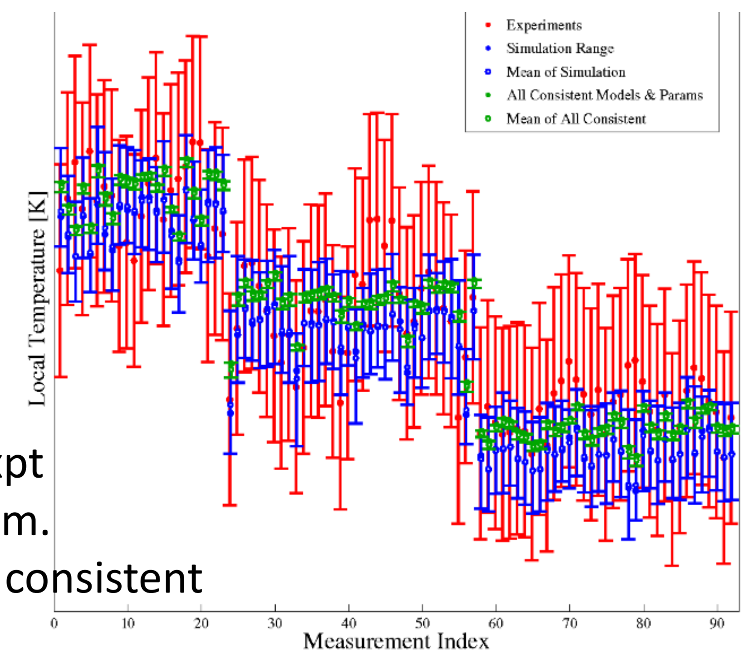
O_2 concentrations boiler simulation

- Alstom Power 1000MWe “Twin Fireball” boiler
- Supply power for 1M people
- 1mm grid resolution = 9 x 10^{12} cells
- 100x > largest problems solved today
- AMR, linear systems, thermal radiation
- Turbulent combustion LES

Simulations of Clean coal Boilers using ARCHES in Uintah



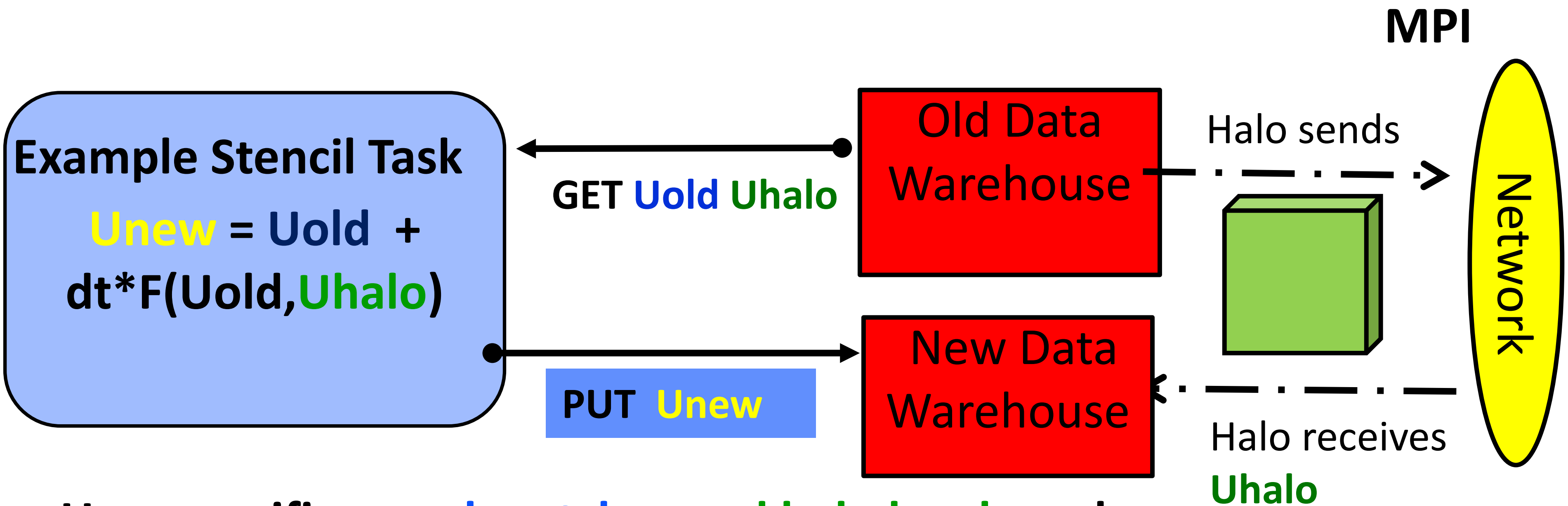
- Traditional Lagrangian/RANS approaches do not address well particle effects so use **Large Eddy Simulation** has potential to be an important design tool
- **Structured, high order finite-volume** Mass, momentum, energy conservation
- **Particles via DQMOM** (many small linear solves)
- Low Mach number approx. (**pressure Poisson solve up to 10^{12} variables** hypre GMG + RB GS
- **Radiation** via Discrete Ordinates – massive
- solves 20+ every few steps of Radiation Transfer Equation with hypre
- **Radiation Ray tracing** .
- **Uncertainty quantification**



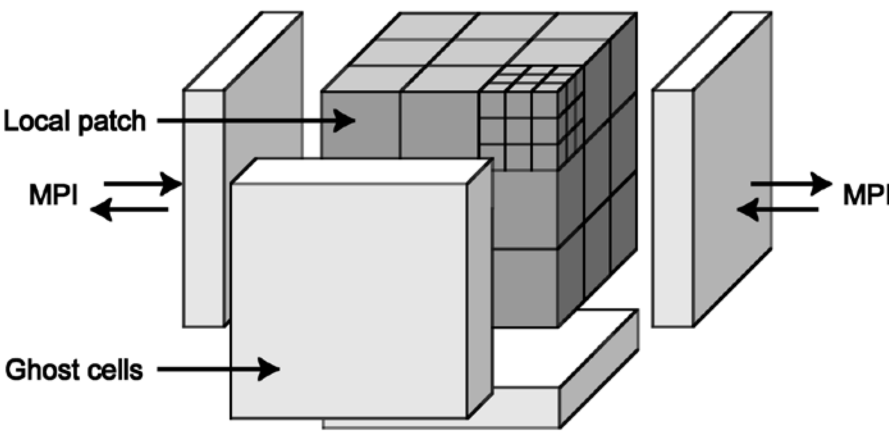
Red is expt
Blue is sim.
Green is consistent

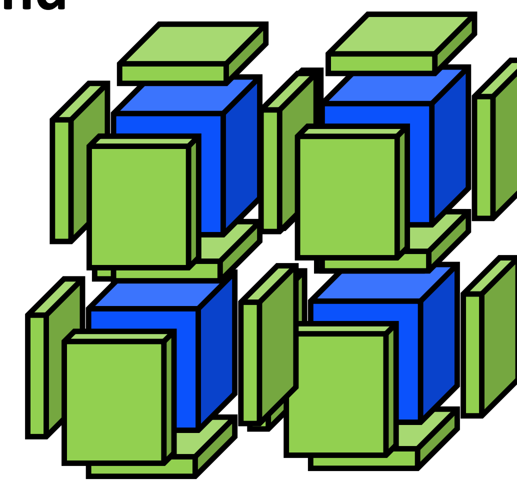
See [**Modest and Howarth]**

Uintah Programing Model for Stencil Timestep



User specifies mesh patches and halo levels and connections





Uintah Architecture

Applications code
Programing model

Components NOT architecture specific and do not change

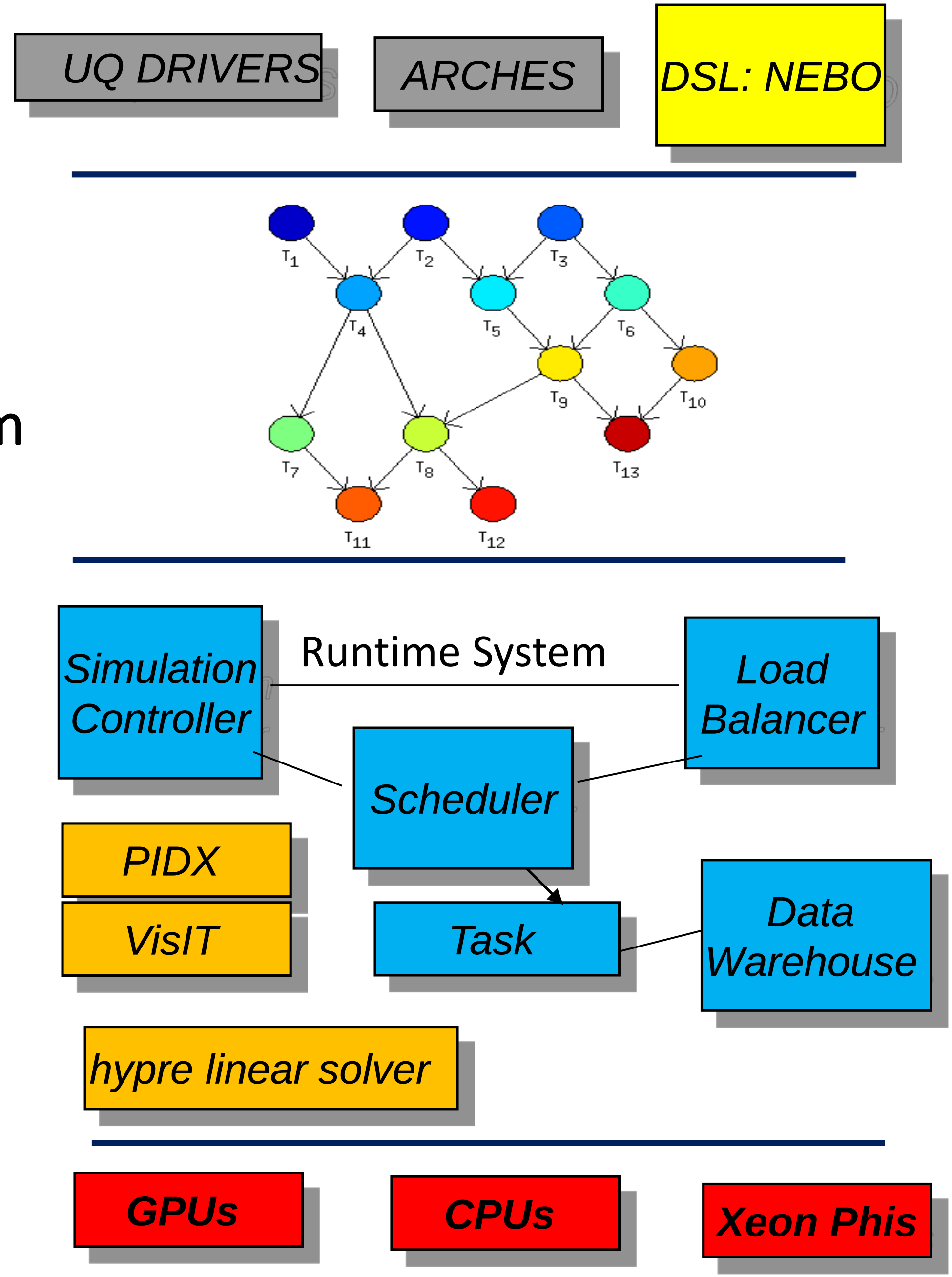
Automatically generated
Abstract C++ Task Graph Form

Adaptive Execution of tasks

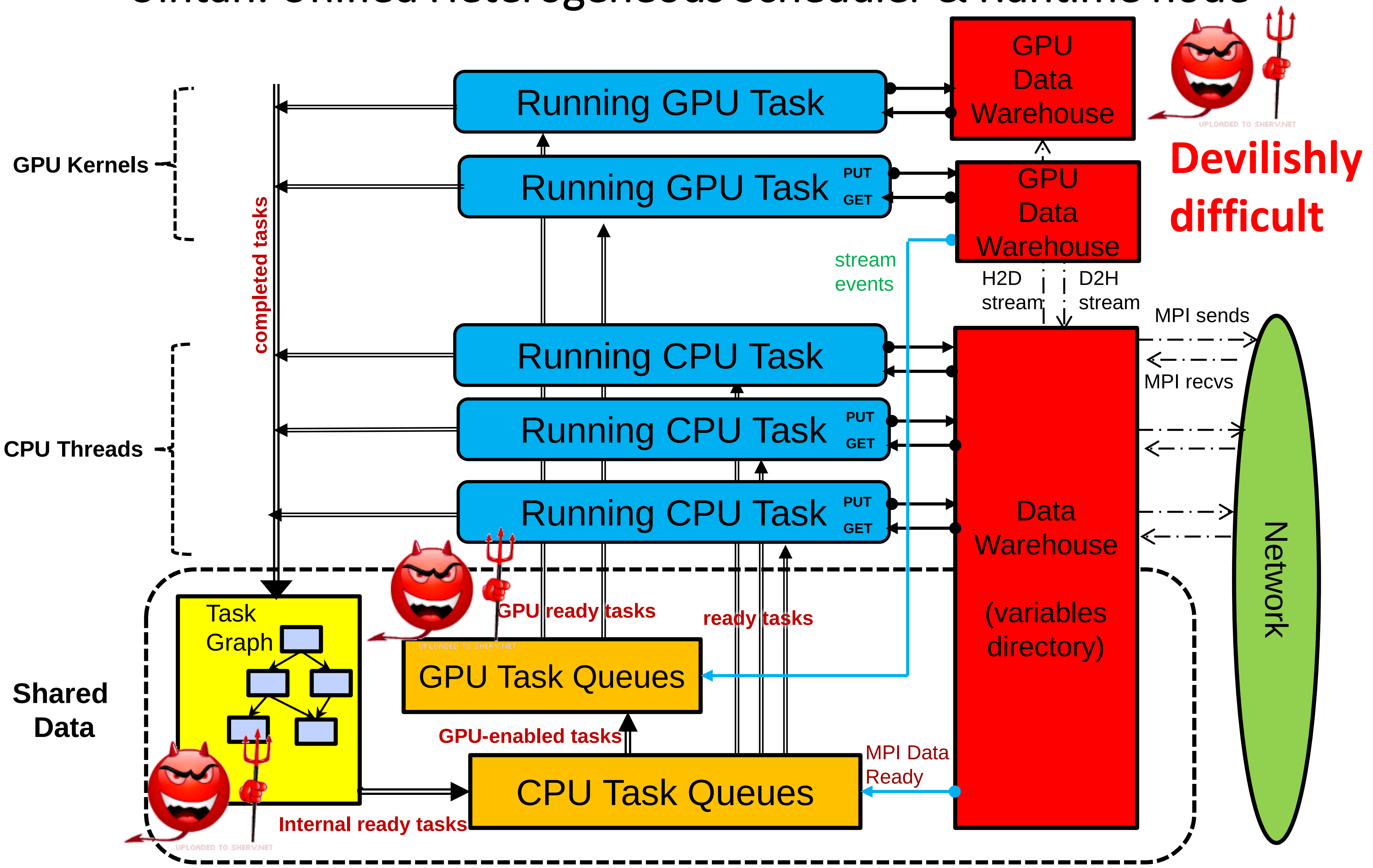
asynchronous out-of-order execution, work stealing, overlap communication & computation.

Strong and weak scaling out to 800K cores for AMR Fluid structure interaction

Open source software
Worldwide distribution
Broad user base

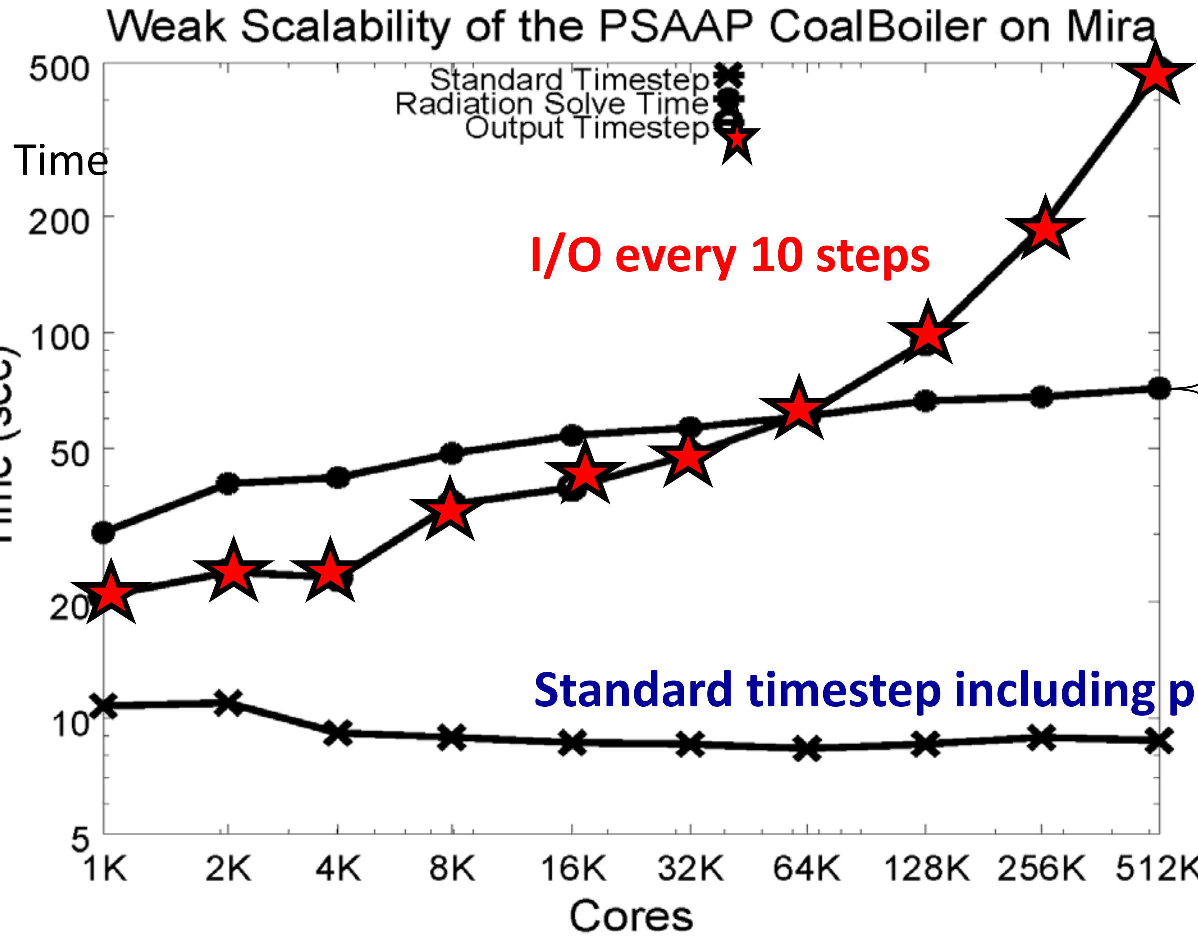


Uintah: Unified Heterogeneous Scheduler & Runtime node



No MPI inside node, lock free Data Warehouse , cores and GPUs pull work

Scaling Results
Mira 5/22



Standard timestep including pressure Poisson solve

Radiation solve Discrete Ordinates

Every 7 steps S_N 6 , 48 directions

hypre for each direction

One 12x12x12 patch per core, 10K variables per core, 31 timesteps

Largest case 5 Bn unknowns. Production runs use 250K cores

For I/O PIDX scales better and is being linked to Uintah

For radiation we have Raytracing working

Radiation Overview

Solving **energy** and **radiative heat transfer** equations simultaneously

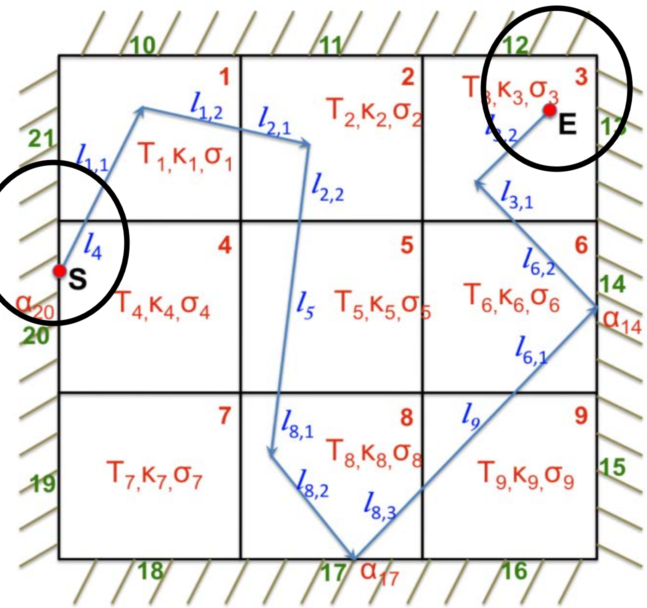
$$\frac{\partial T}{\partial t} = \text{Diffusion} - \text{Convection} + \text{Source/Sinks}$$

$$\nabla \cdot q$$

- Radiation-energy coupling incorporated by radiative source term
- Energy equation conventionally solved by ARCHES (finite volume)
- Temperature field, **T** used to compute **net radiative source term**
- $\nabla \cdot q$ requires integration of incoming intensity about a solid angle with reverse Monte Carlo ray tracing (RMCRT)

$$\int_{4\pi} I_{in} d\Omega \Rightarrow \sum_{ray=1}^{N} I_{ray} \frac{4\pi}{N}$$

Mutually exclusive Rays traced backwards from e.g. S to E *computational cell (cuda thread), eliminating the need to track rays that never reach that cell* S



Multi-Level AMR GPU RMCRT

Replicate mesh and use coarse representation of computational domain with multiple levels

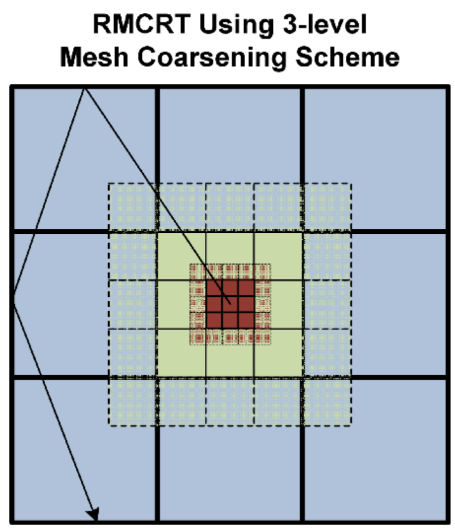
Define Region of Interest (**ROI**)

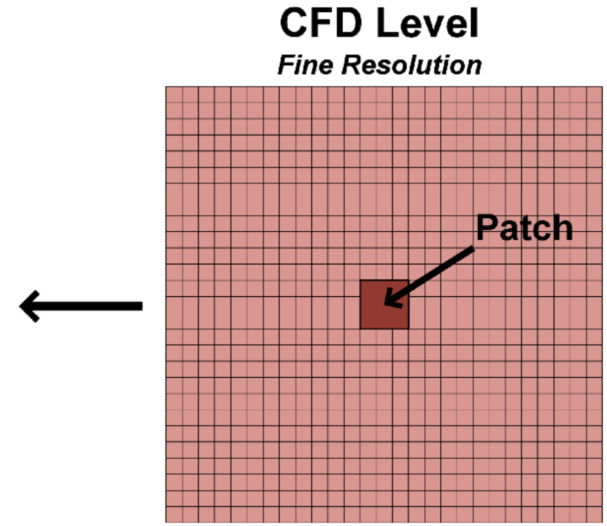
Surround **ROI** with coarser grids

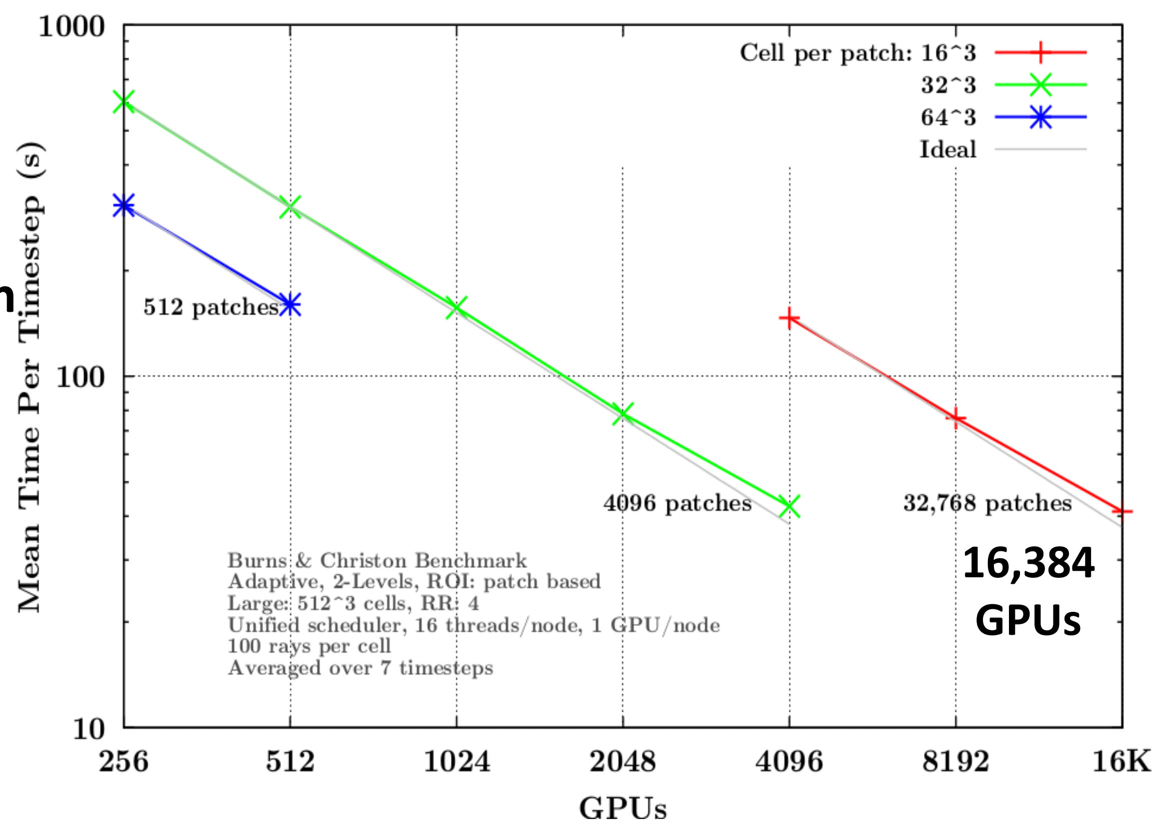
As rays travel further away from **ROI**, the mesh spacing becomes larger

Transmit new information relating to heat fluxes adsorption and scattering coeffs using same adaptive ideas

Reduces computational cost, memory and communications volume.



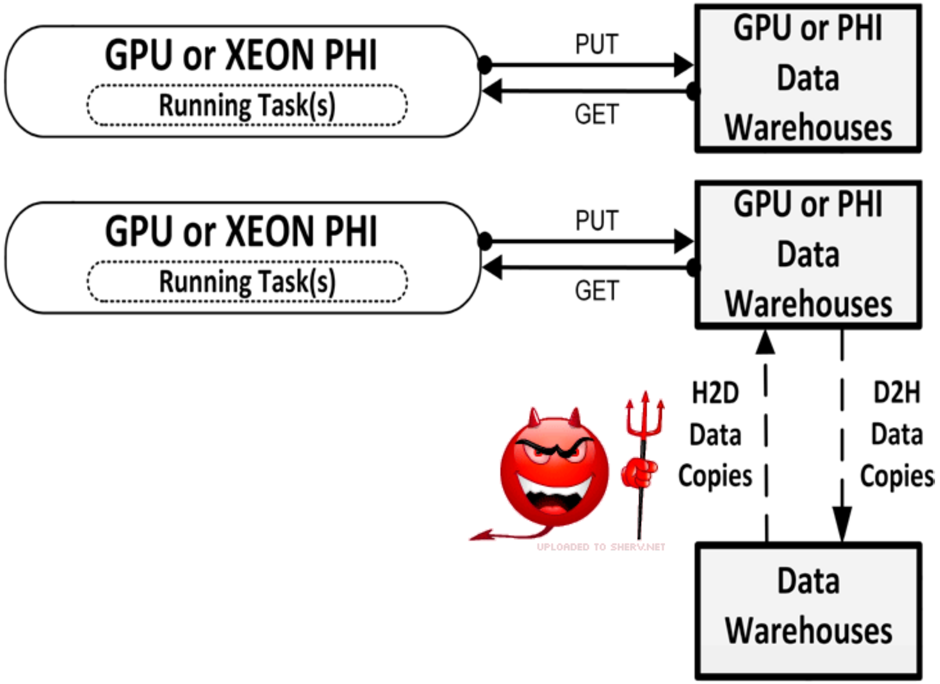




Todd Harman, Alan Humphrey

Better use of GPUs with Per Task GPU Datawarehouse

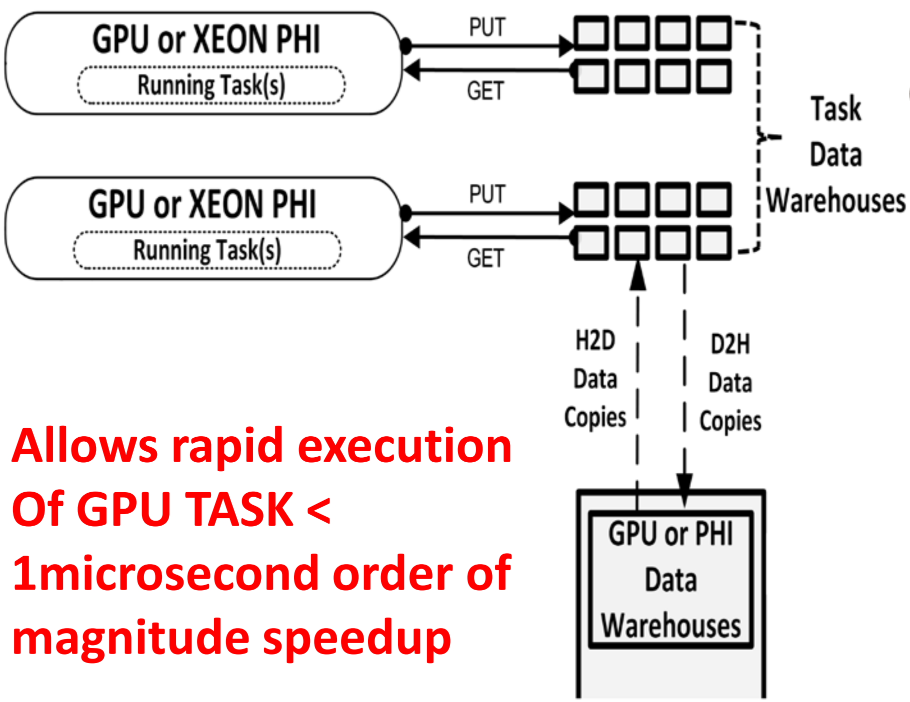
- Single, shared DataWarehouse does not scale with problem complexity
  - increasing DW size, meant more device synchronization
- Solution: per task DataWarehouses on GPU
  - no sharing or atomic operations required
  - can overlap comp and comm in a thread-safe manner



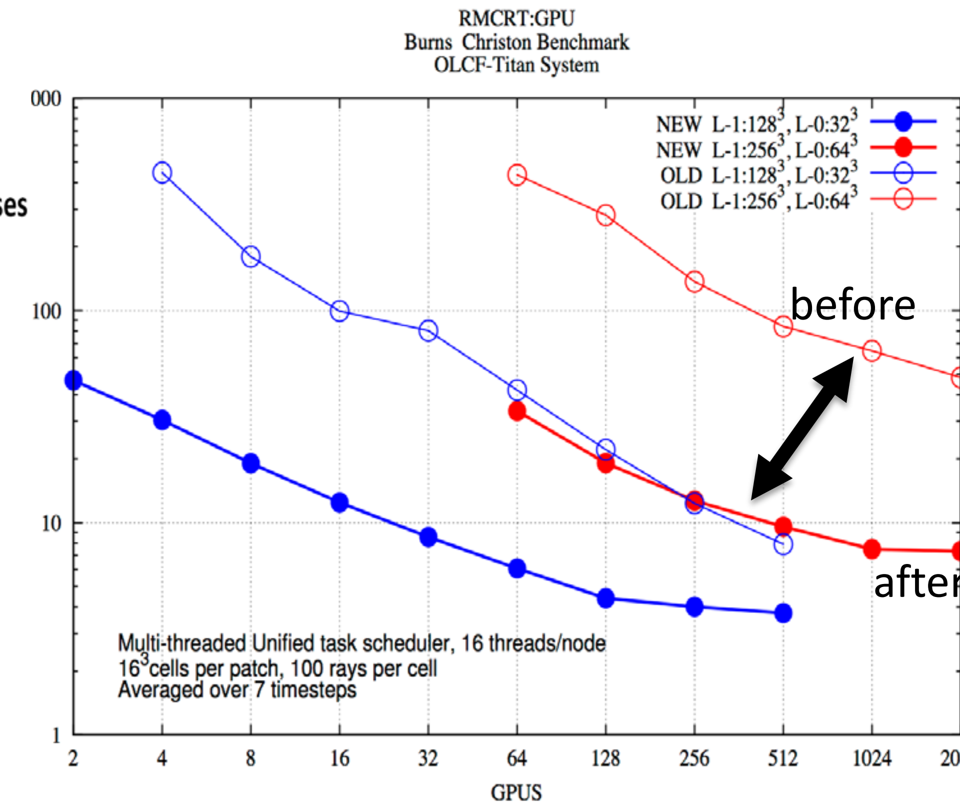
Brad Peterson

Better use of GPUs with Per Task GPU Datawarehouse

- Single, shared DataWarehouse does not scale with problem complexity
  - increasing DW size, meant more device synchronization
- Solution: per task DataWarehouses on GPU
  - no sharing or atomic operations required
  - can overlap comp and comm in a thread-safe manner



Allows rapid execution Of GPU TASK < 1microsecond order of magnitude speedup



Brad Peterson

Abstractions for Portability and Node Performance

- Use **Domain Specific Language Nebo -**weak scales to all of Titan 18K GPUs and 260K cpus
- Use **Kokkos abstraction layer** that maps loops onto machine efficiently **using cache aware memory models and vectorization / Openmp**

- Both use **C++ template metaprogramming** for compile time data structures and functions
- **While Nebo** allows users to solve problems within language framework, **Kokkos** allows users to modify code at loop level and to optimize loops and good memory placement

Kokkos – Uintah Infrastructure

Incremental refactor to Kokkos parallel patterns/views

Replace patch grid iterator loops

for (auto itr = patch.begin(); itr != patch.end(); ++itr) {
 IntVector iv = *itr;
 A[iv] = B[iv] + C[iv];}

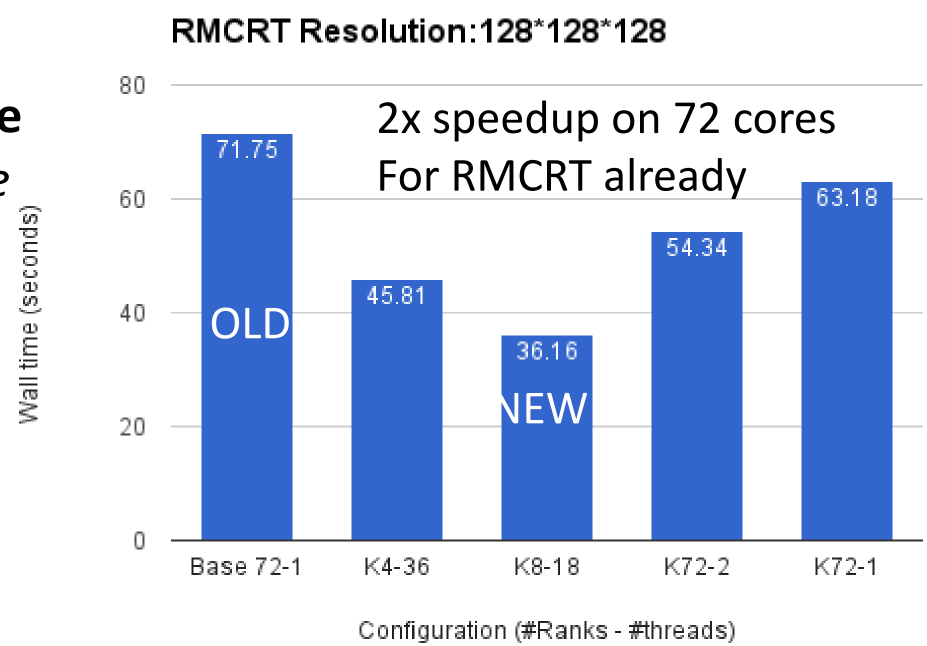
BECOMES

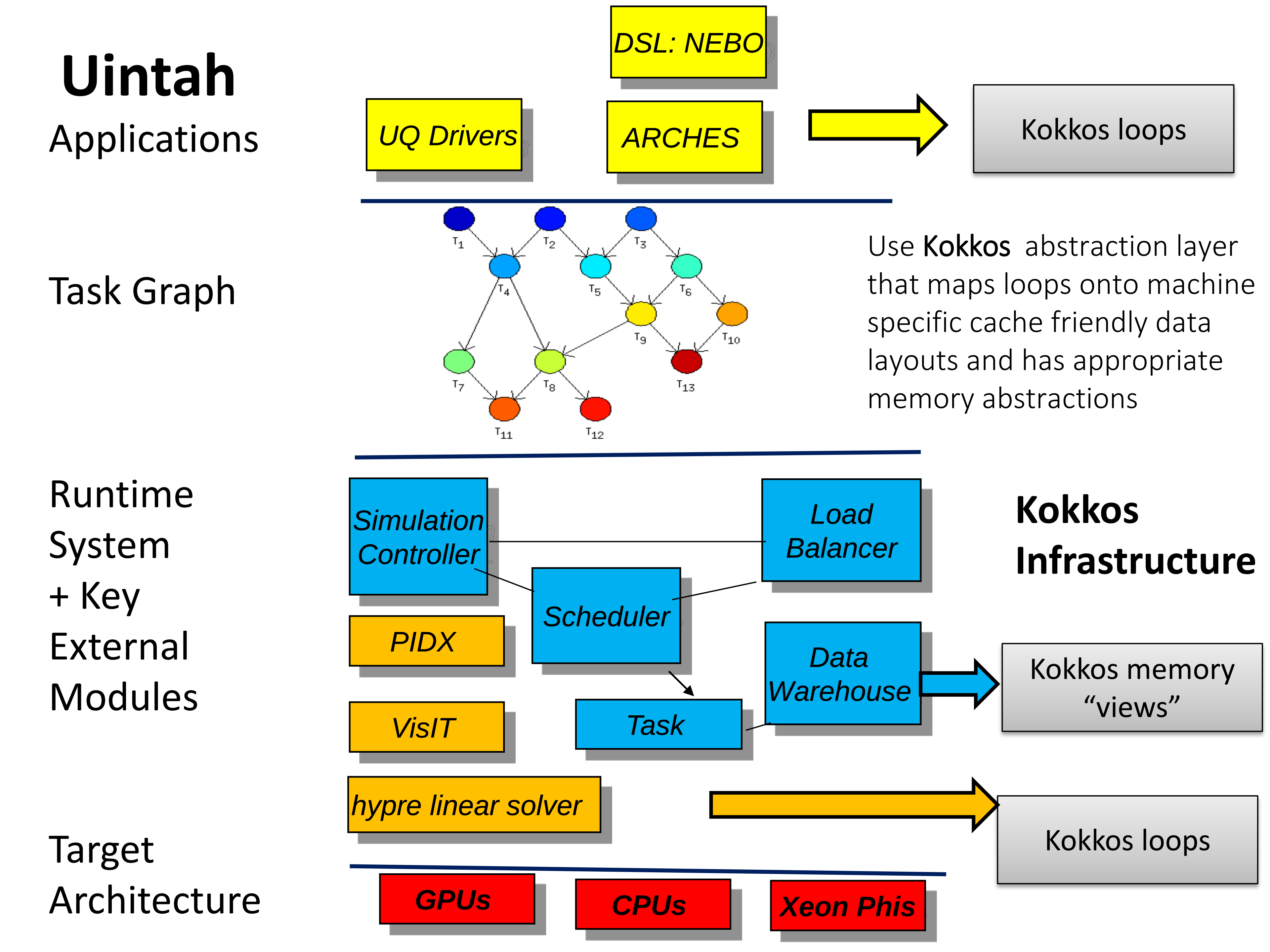
parallel_for(patch.range(), LAMBDA(int i, int j, int k) {
 A(i,j,k) = B(i,j,k) + C(i,j,k)});

Dan Sunderland, Alan Humphrey

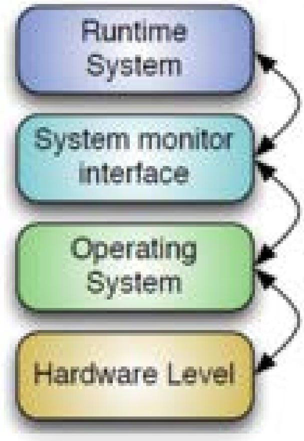
Refactored grid variables to expose unmanaged Kokkos views *Uses the existing memory allocations and layouts Removes many levels of indirection in existing implementation.*

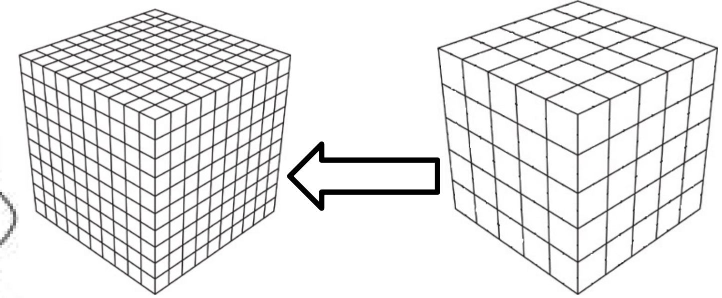
Future work using managed Kokkos views for portability all components benefit

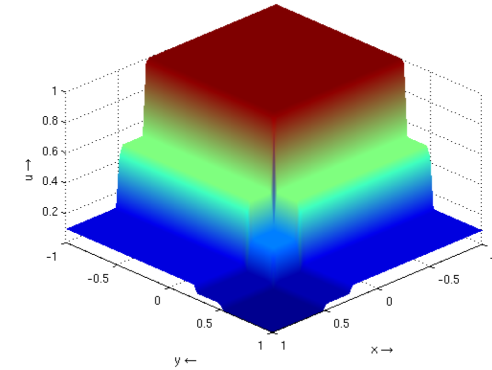



Uintah
Applications
DSL: NEBO
UQ Drivers
ARCHES
Kokkos loops
Task Graph
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
Use Kokkos abstraction layer that maps loops onto machine specific cache friendly data layouts and has appropriate memory abstractions
Runtime System + Key External Modules
Simulation Controller
Load Balancer
Scheduler
PIDX
Data Warehouse
Task
VisIT
Kokkos Infrastructure
Kokkos memory “views”
hypre linear solver
Kokkos loops
Target Architecture
GPUs
CPUs
Xeon Phis

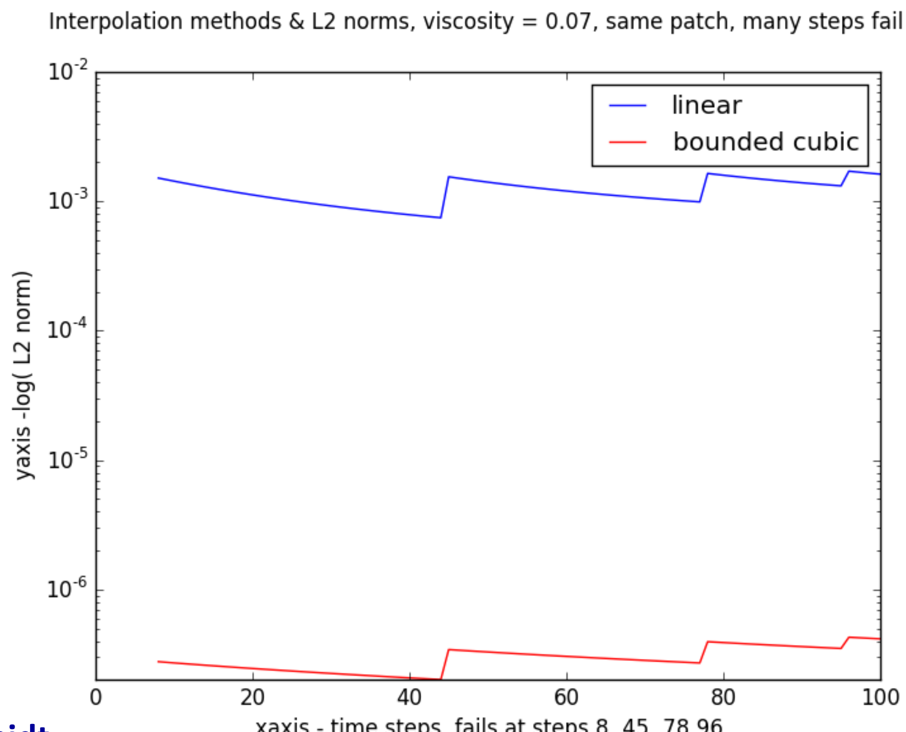
Resilience Joint Work With NSF XPS Project







- Need interfaces at system level to address :
- Core failure – reroute tasks
- Comms failure – reroute message
- Node failure – need to replicate patches use an AMR type approach in which a coarse patch is on another node. In 3D has 12.5% overhead Interpolation is key here
- Core slowdown - move tasaks elsewhere . 10% slowdown auto move Respa SC 2015 workshop paper
- Need to address possible MTBF of minutes ? Or do we?
- Early user program TACC Intel KNL



Aditya Pakki, Sahithi Chaganti, Alan Humphrey John Schmidt

Summary

- **Seven abstractions are all** important for portability, scaling and **for not needing to change applications code . Showing that this approach works at scale is a key outcome for our project**
- **Scalability** will still require tuning the runtime system.
- **Performance Portability: u**se Kokkos for rewriting legacy applications Phi and GPU ongoing. Aiming at Coral + Apex ++
- **Design Study using** 350M cpu hr INCITE award in 2016
- **Using packages** for scalable I/O (768K cores) Utah PIDX and linear algebra ongoing but GPUs problematic for linear solver community
- **Resilience** ongoing experiments **but perhaps not now expected to be such a problem?????**