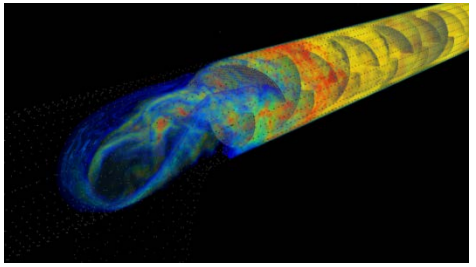
Developing Software Frameworks for Petascale and Beyond Using Dynamic Graph Based Approaches – Lessons and Achievements with Uintah



www.uintah.utah.edu

Martin Berzins

1. **Background and motivation**
2. **Uintah Software and Multicore Scalability**
3. **Runtime Systems for Heterogeneous Architectures**
4. **Conclusions Portability, DSLs and Kokkos**

Software team:
Qingyu Meng* John Schmidt, Alan Humphrey, Justin Luitjens*,

DSL team lead
James Sutherland

* Now in industry

Machines: Titan, Stampede, Mira, Vulcan, Blue Waters, local linux, local linux/GPU, MIC

SCI INSTITUTE

Thanks to DOE ASCI (97-10), NSF , DOE NETL+NNSA ARL NSF , INCITE, XSEDE, James, Carter and Dan

THE UNIVERSITY OF UTAH

Extreme Scale Research and Applications in Utah

Energetic Materials: Chuck Wight, Jacqueline Beckvermit, Joseph Peterson, Todd Harman, Qingyu Meng NSF PetaApps 2009-2014 $1M, P.I. MB

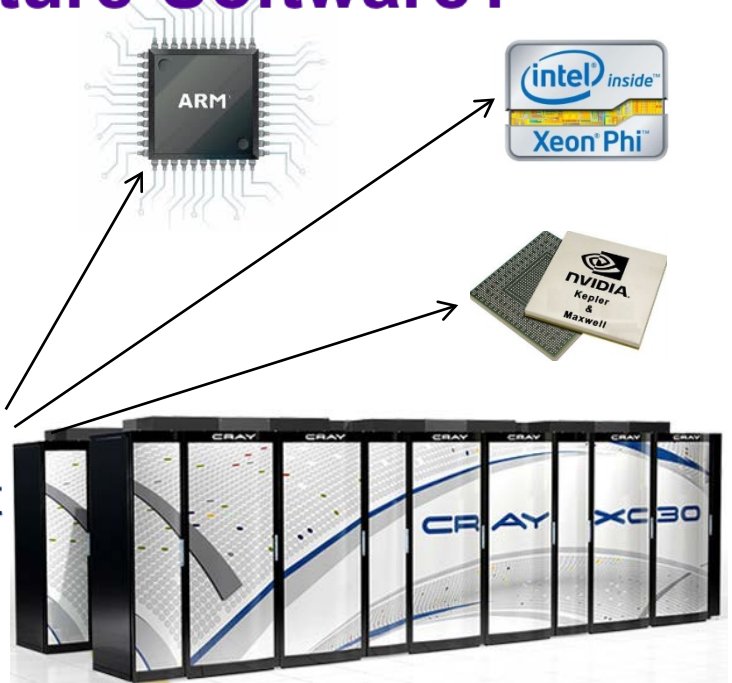
PSAAP Clean Coal Boilers: Phil Smith (P.I.), Jeremy Thornock James Sutherland etc Alan Humphrey John Schmidt DOE NNSA 2013-2018 $16M (MB CS lead)

Electronic Materials by Design: MB (PI) Dmitry Bedrov, Mike Kirby, Justin Hooper, Alan Humphrey Chris Gritton, + ARL TEAM 2011-2016 $12M

The Exascale challenge for Future Software?

202X Exascale “goal” requires 50 Petaflops per Megawatt, - not possible with existing hardware/software approaches.

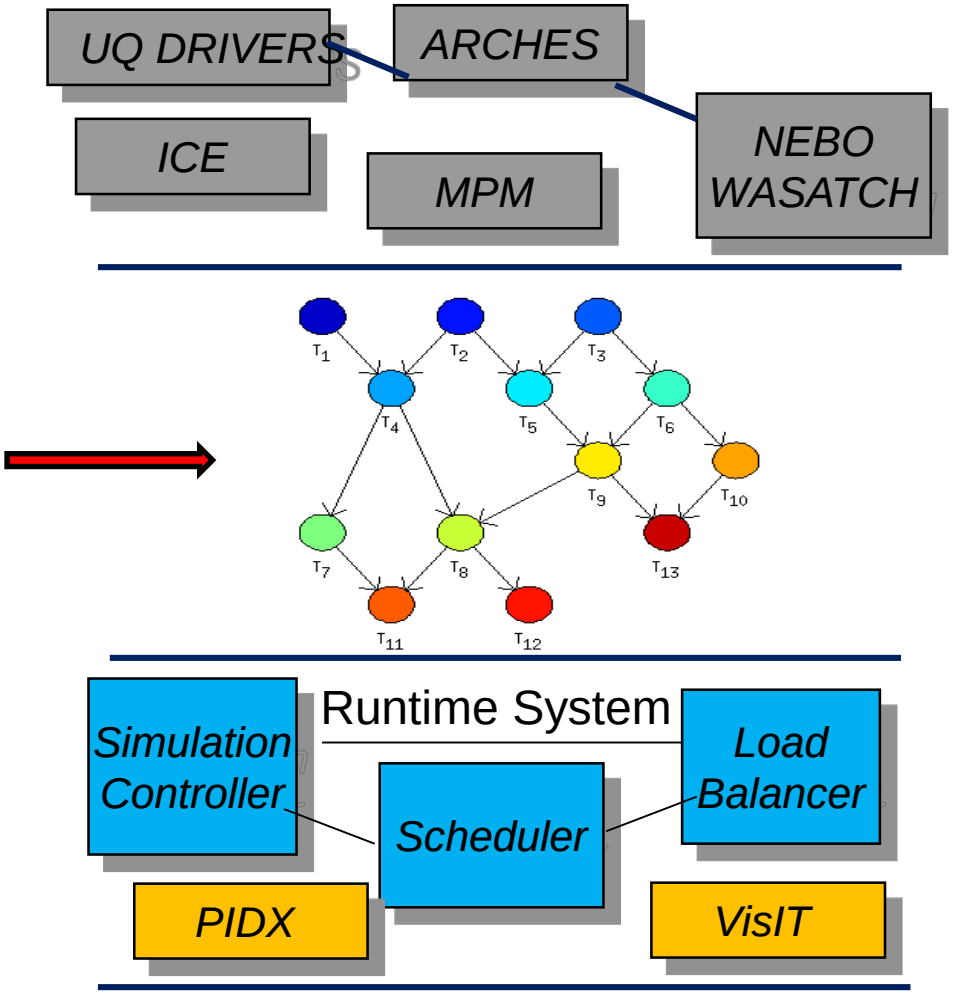
HPC software now has to take into account considerable uncertainty in architectures and run on accelerator-based machines that will be much more energy efficient.
Adaptive software needed



Exascale capable future software?

- **Application Specification** via ICE MPM ARCHES or NEBO/WASATCH DSL
- **Abstract task-graph** program that executes on:
- **Runtime System** with: asynchronous out-of-order execution, work stealing
- Overlap communication & computation
- Tasks running on cores and accelerators

- **Scalable I/O via Visus PIDX**

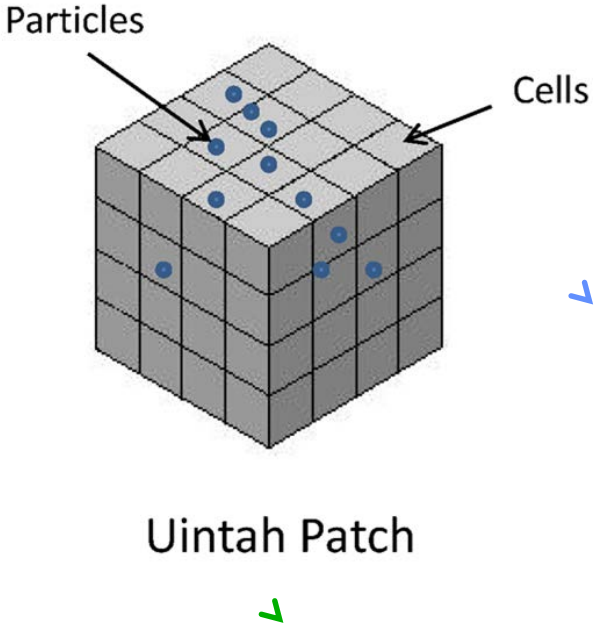


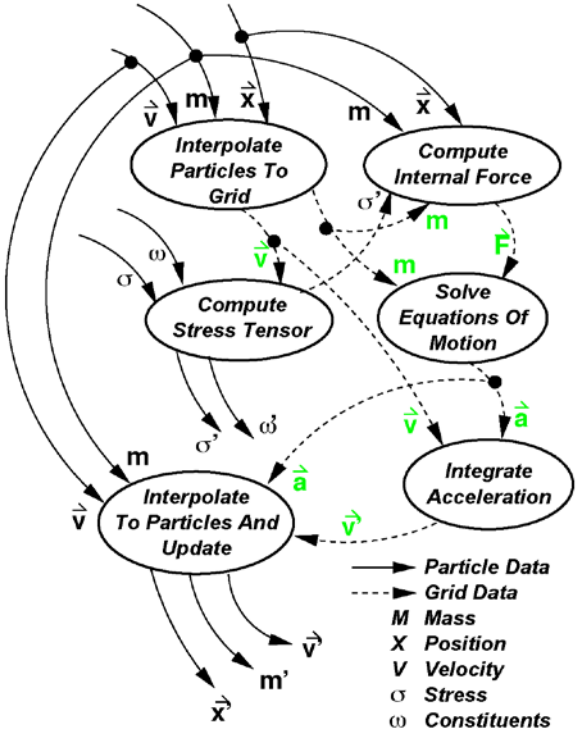
Uintah(X) Architecture Decomposition

The problem specs for some components have not changed as we have gone from 600 to 600K cores it is the Runtime System that changed

Uintah Patch, Variables and Task Graph

ICE is a cell-centered finite volume method for Navier Stokes equations





- **ICE** Structured Grid Variable (for Flows) are Cell Centered Nodes, Face Centered Nodes.
- Unstructured Points (for Solids) are **MPM** Particles

ARCHES is a combustion code using several different radiation models and linear solvers

Tasks define their I/O

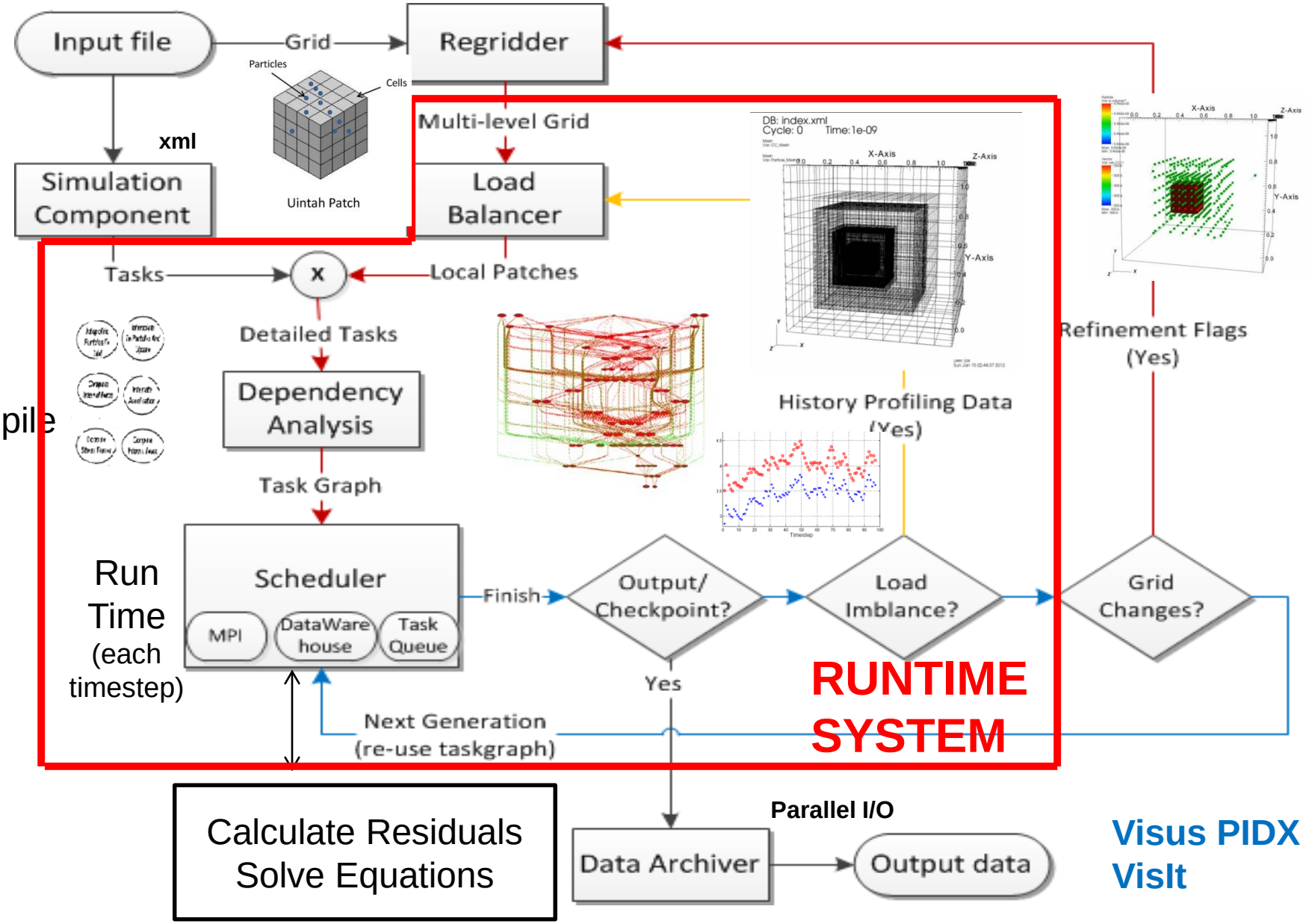
Uintah creates graph

Data comes from nodal warehouse via MPI when needed

Adaptive execution

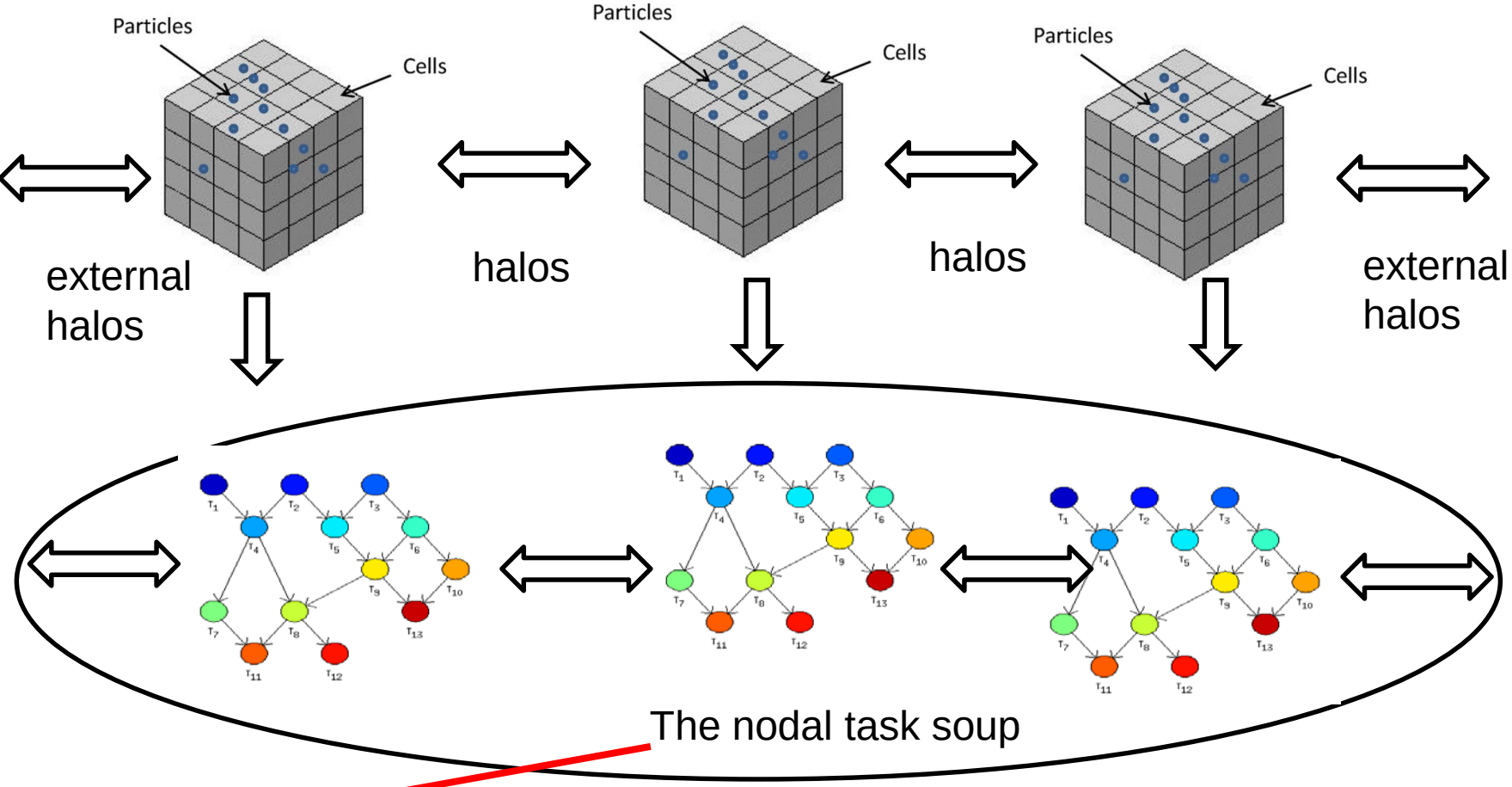
Uintah:MD based on Lucretius is a new molecular dynamics component

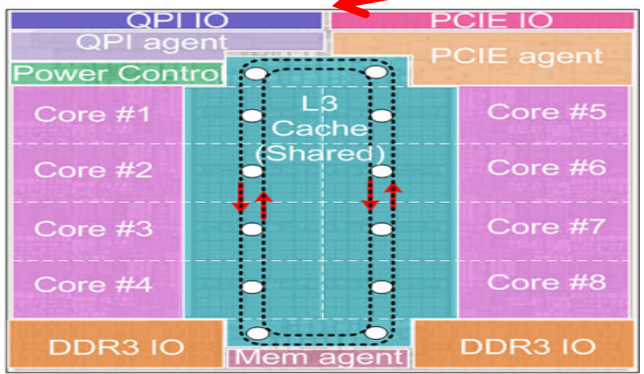
ARCHES or WASATCH/NEBO



UINTAH ARCHITECTURE

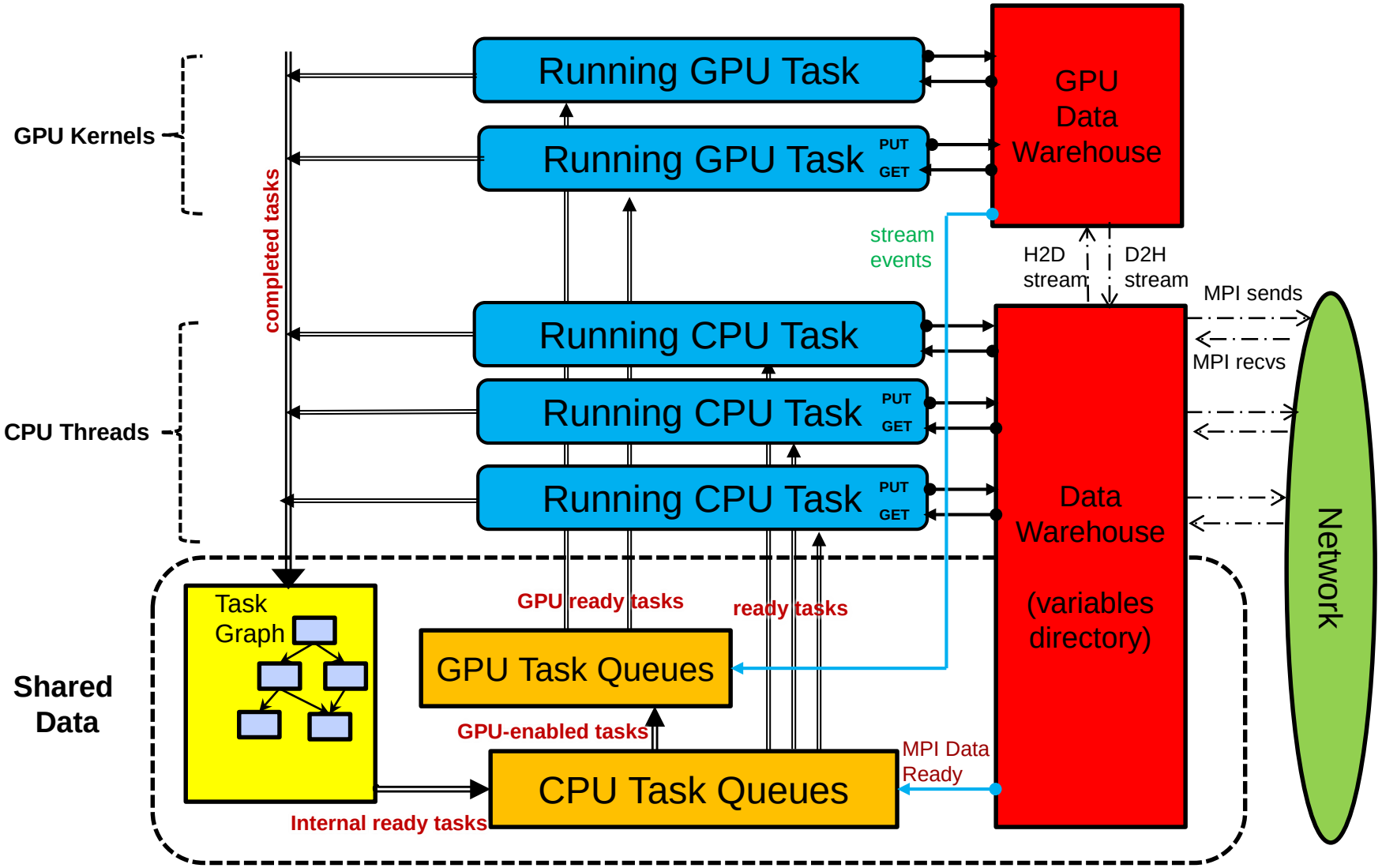
Task graph structure on a multicore node with multiple patches





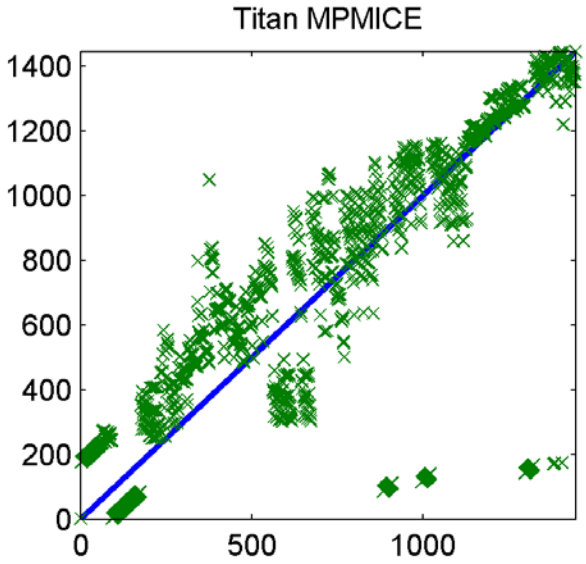
This is not a single graph. Multiscale and Multi-Physics merely add flavor to the "soup". There are many adaptive strategies and tricks that are used in the execution of this graph soup.

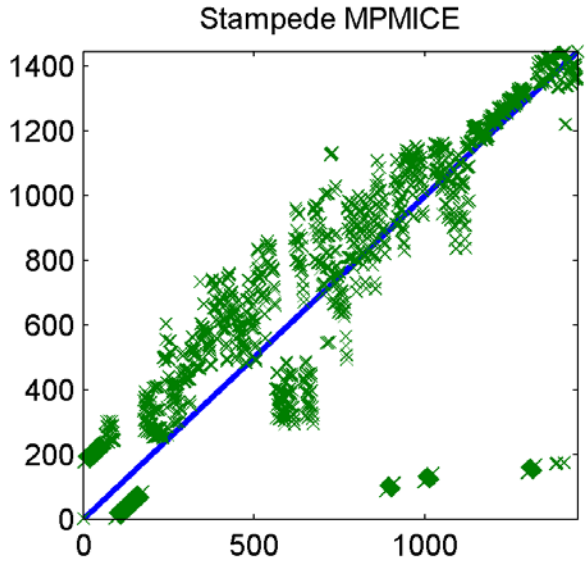
Unified Heterogeneous Scheduler & Runtime node

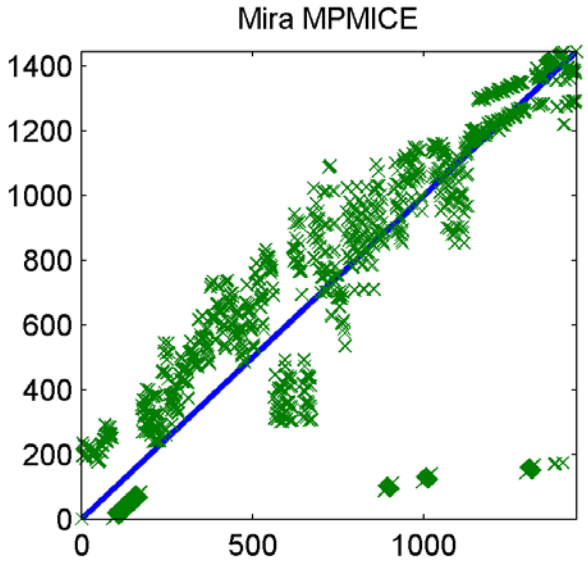


No MPI inside node, lock free DW , cores and GPUs pull work

Scalability is at least partially achieved by not executing tasks in order e.g. AMR fluid-structure interaction







Straight line represents given order of tasks Green X shows when a task is actually executed.
Above the line means late execution while below the line means early execution took place. More "late" tasks than "early" ones as e.g.

TASKS: 1 2 3 4 5 ⇨ 1 4 2 3 5

Early Late execution

Summary of Scalability Improvements

(i) Move to a one MPI process per multicore node reduces memory to less than 10% of previous for 100K+ cores

(ii) Use optimal size patches to balance overhead and granularity 16x16x 16 to 30x30x30.

(iii) Use only one data warehouse but allow all cores fast access to it, through the use of atomic operations.

(iv) Prioritize tasks with the most external communications

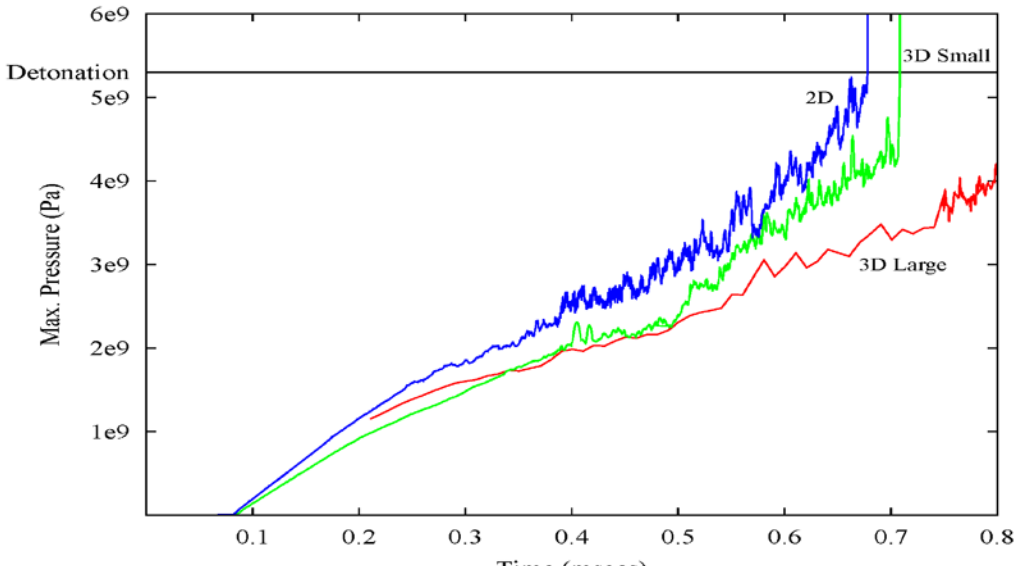
(v) Use out-of-order execution when possible

NSF funded modeling of Spanish Fork Accident 8/10/05

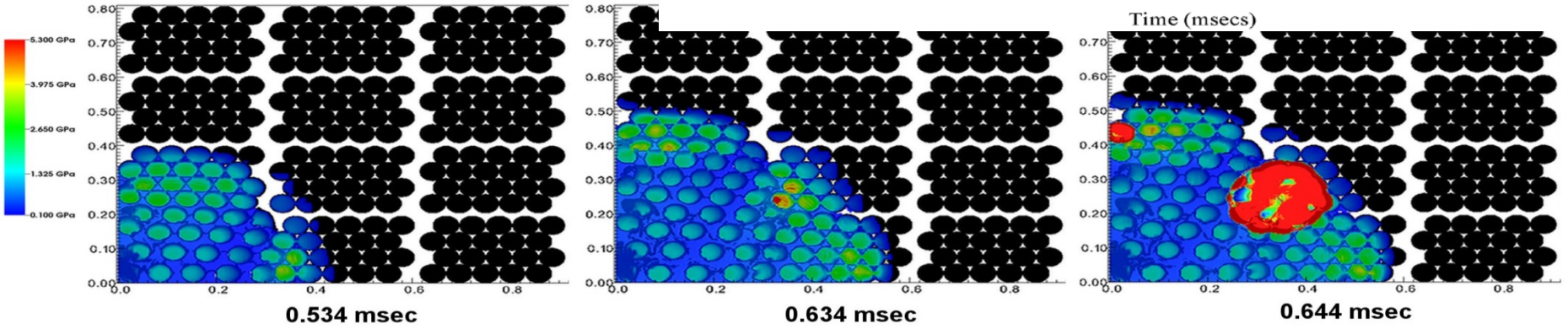
Speeding truck with 8000 explosive boosters each with 2.5-5.5 lbs of explosive overturned and caught fire

Experimental evidence for a transition from deflagration to detonation?

Deflagration wave moves at ~400m/s not all explosive consumed. Detonation wave moves 8500m/s all explosive consumed.

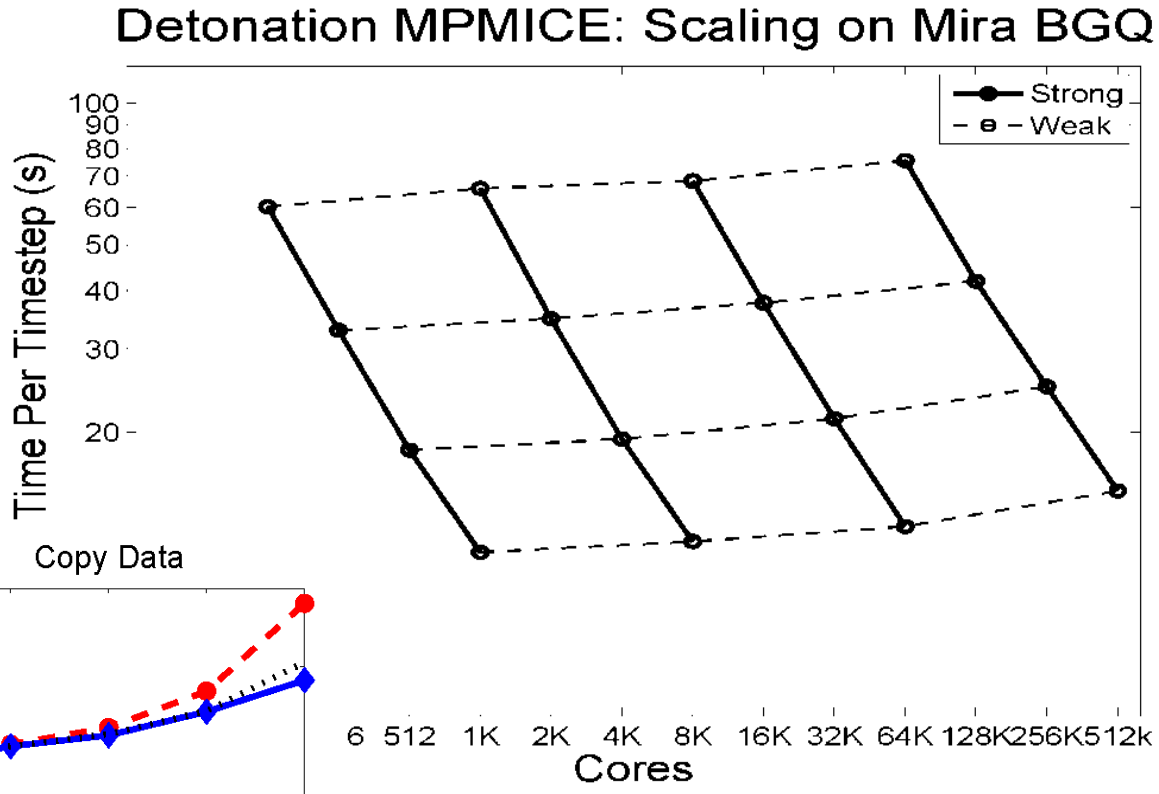


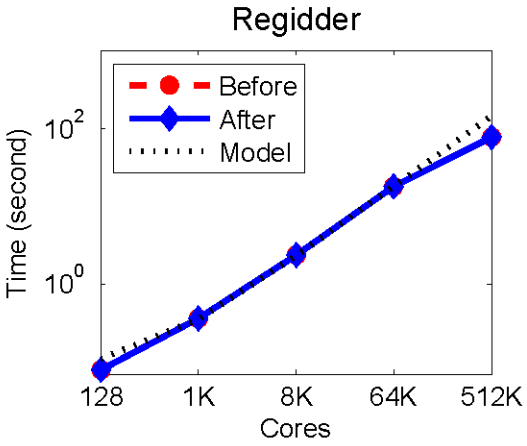
2013 Incite 200m cpu hrs

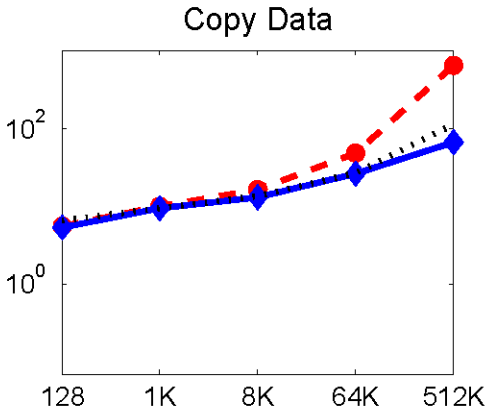


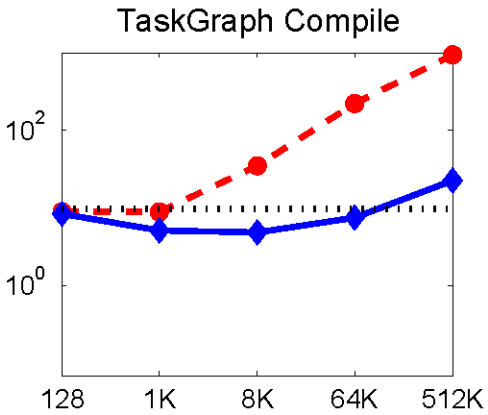
Spanish Fork Accident

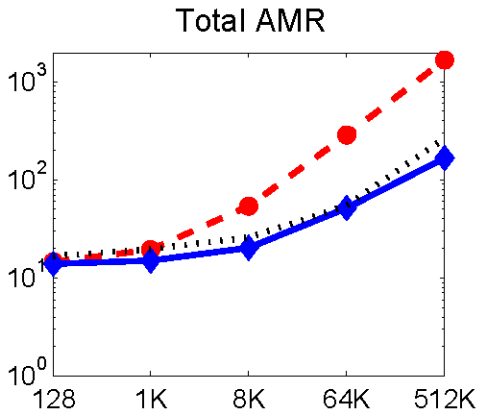
500K mesh patches
1.3 Billion mesh cells
7.8 Billion particles







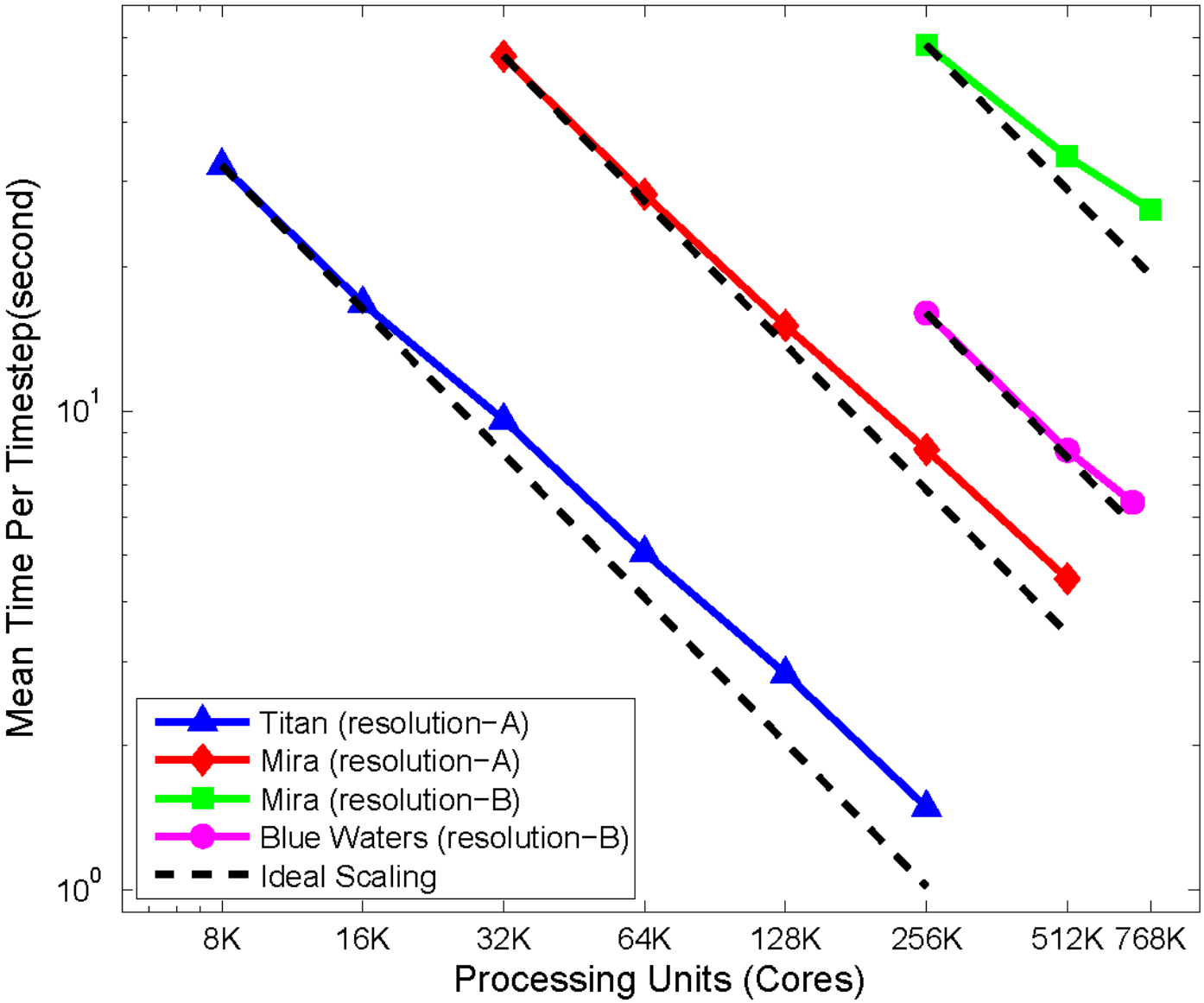




At every stage when we move to the next generation of problems Some of the algorithms and data structures need to be replaced .

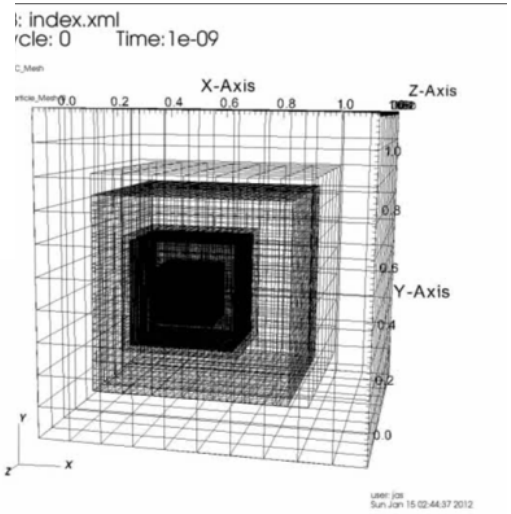
Scalability at one level is no certain Indicator fro problems or machines An order of magnitude larger

MPM AMR ICE Strong Scaling



Mira DOE BG/Q
768K cores
Blue Waters Cray XE6/XK7 700K+ cores

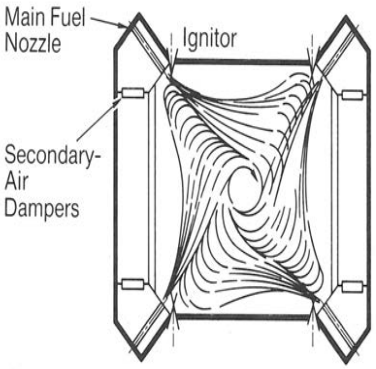
Resolution B
29 Billion particles
4 Billion mesh cells
1.2 Million mesh patches

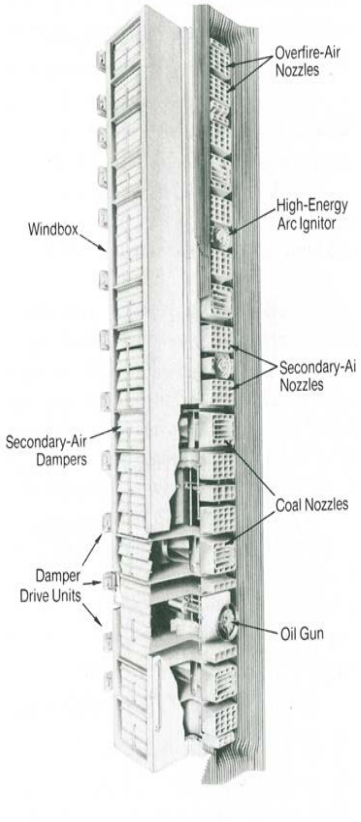


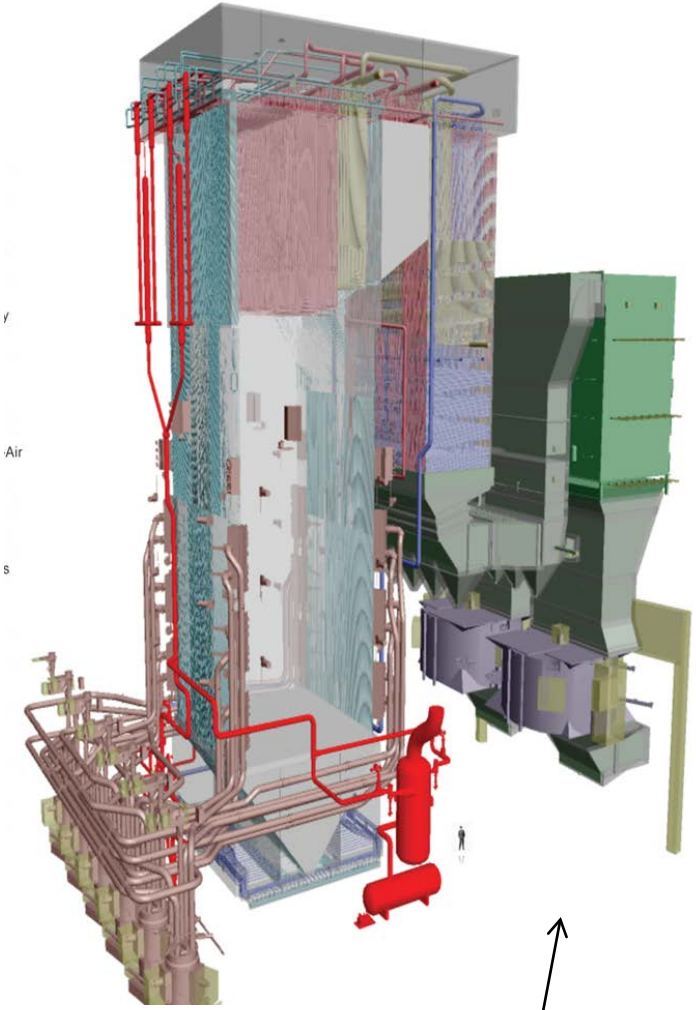
Complex fluid-structure interaction problem with adaptive mesh refinement, see SC13/14 paper

NSF funding.

An Exascale Design Problem - Alstom Clean Coal Boilers





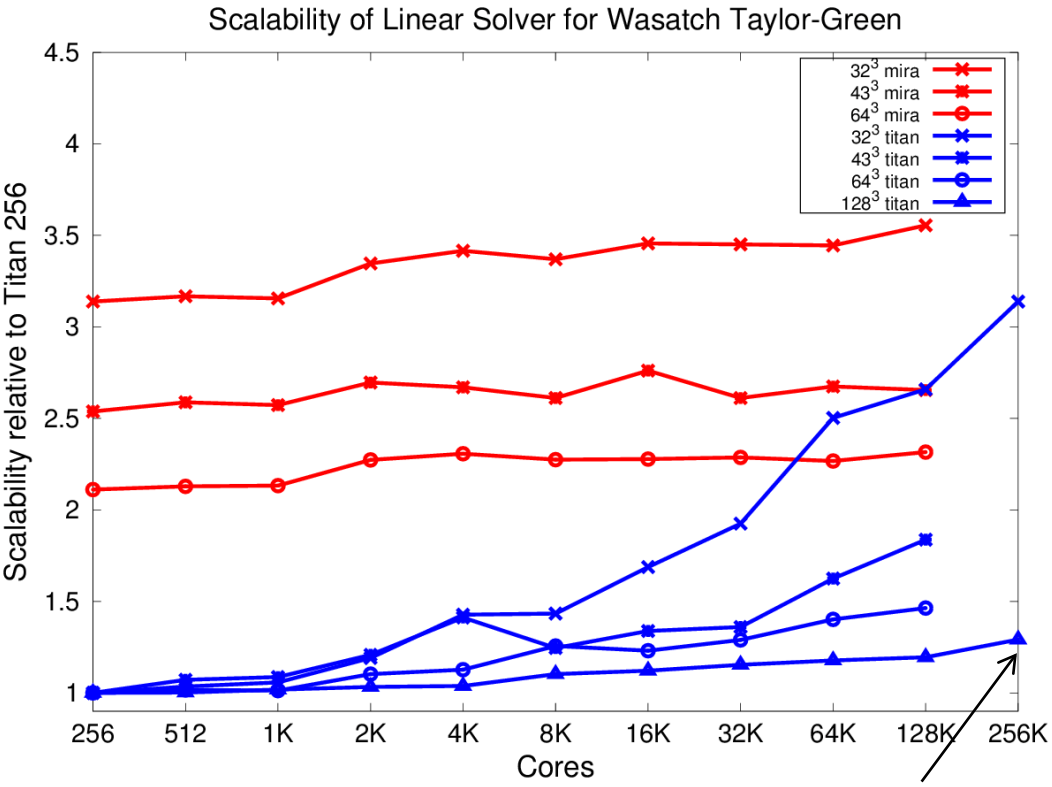


Temperature field

For 350MWe boiler problem. LES resolution needed: 1mm per side for each computational volume = 9x 10^{12} cells
This is one thousand times larger than the largest problems we solve today.

Prof. Phil Smith Dr Jeremy Thornock ICSE

Linear Solves arises from Low Mach Number Navier –Stokes Equations

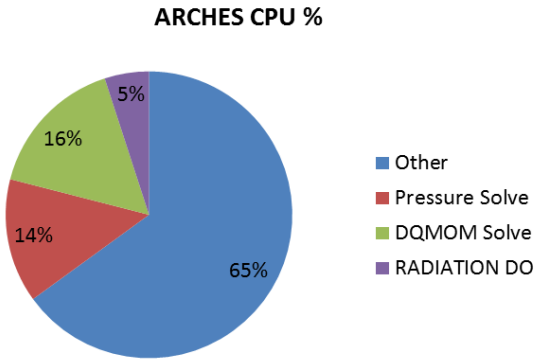


$$\nabla^2 p = R, \quad \text{where } R = \nabla . F + \frac{\partial^2 p}{\partial t^2}$$

Use Hypre Solver from LLNL Preconditioned Conjugate Gradients on regular mesh patches used

Multi-grid pre-conditioner used Careful adaptive strategies needed to get scalability

2.2 Trillion DOF



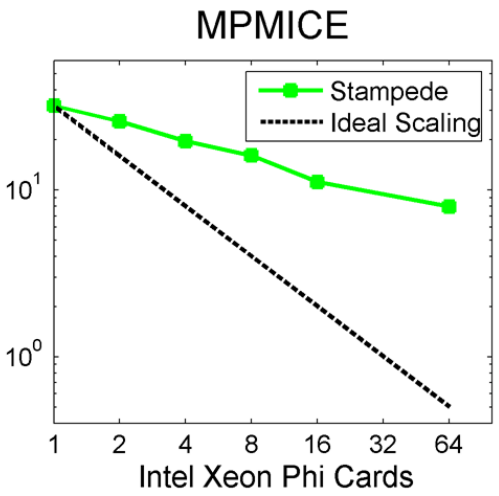
Each **Mira Run** is scaled wrt the **Titan Run at 256 cores**
Note these times are not the same for different patch sizes.

Weak Scalability of Hypre Code

One radiation solve every 10 timesteps

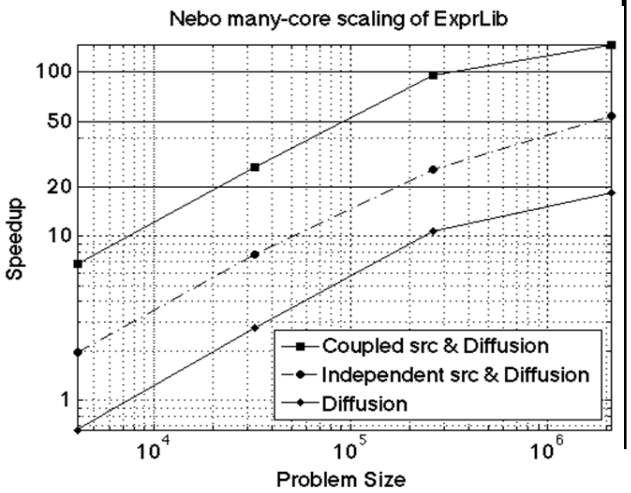
Summary

- **Layered DAG abstraction** important for scaling and for not needing to change applications code
- **Scalability** still requires tuning the runtime system. **Cannot develop nodal code in isolation.**
- **Future Portability** Kokkos for rewriting legacy applications +Wasach/Nebo DSL for new code. MIC and GPU ongoing.



DSL Wasatch (Sutherland) gives 3-4x speedup.
Nebo backend for CPU resulted in 20-30% speedup in the entire Wasatch code base. Much of the Wasatch code base is GPU-ready next is Arches

Good GPU scaling with (>32^3 per patch).Loop fusion for GPU kernels



Kokkos: A Layered Collection of Libraries
See [Carter Edwards and Dan Sunderland]

- **Standard C++, Not a language extension**
  - **In *spirit* of TBB, Thrust & CUSP, Uses C++ template meta-programming**
- **Multidimensional Arrays, *with a twist***
  - **Layout mapping: multi-index (i,j,k,...) ↔ memory location, invisble touse**
  - **Choose layout to satisfy device-specific memory access pattern**
  - **Good initial results on Xeon, Xeon Phi, CPUs**