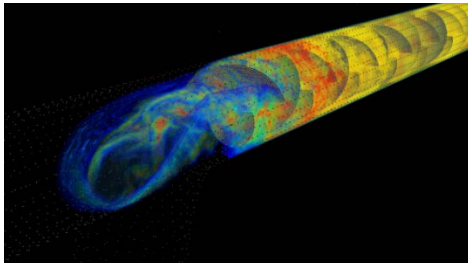
Large Scale Engineering Simulations on Multicore and Heterogeneous Architectures using the Uintah computational Frameworks



www.uintah.utah.edu

Martin Berzins

1. **Background and motivation**
2. **Uintah Software and Multicore Scalability**
3. **Runtime Systems for Heterogeneous Architectures**
4. **Portability for future Architectures Using DSLs(#) and Kokkos (*)**
5. **Conclusions**

#slides from James Sutherland, * slides from Carter Edwards and Dan Sunderland

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

Extreme Scale Research and teams in Utah

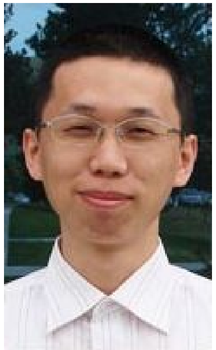
Energetic Materials: Chuck Wight, Jacqueline Beckvermit, Joseph Peterson, Todd Harman, Qingyu Meng NSF PetaApps 2009-2014 $1M, P.I. MB

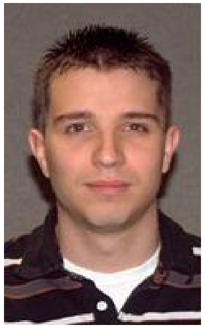
PSAAP Clean Coal Boilers: Phil Smith (P.I.), Jeremy Thornock James Sutherland etc Alan Humphrey John Schmidt DOE NNSA 2013-2018 $16M (MB CS lead)

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

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

DSL team lead
James Sutherland





* Now at Google

* Now at NVIDIA

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

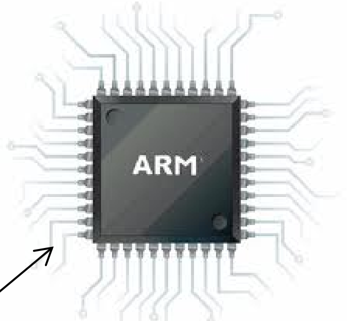
The Exascale challenge for Future Software?

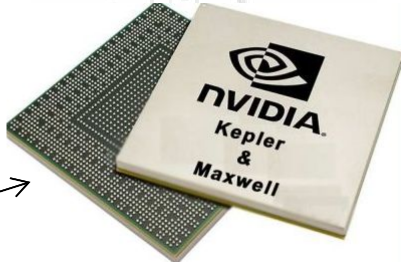
2013 Titan, Blue Gene Q - 2 Petaflops per MegaWatt
300K cpus 5M gpu cores

202X Exascale “goal” requires 50 Petaflops per Megawatt, 1B cores - not possible with existing hardware/software approaches.

Many more cores (majority on “accelerators”), variable Power consumption. Communication delays.
Many more component failures.

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 also means Petascale in a cabinet

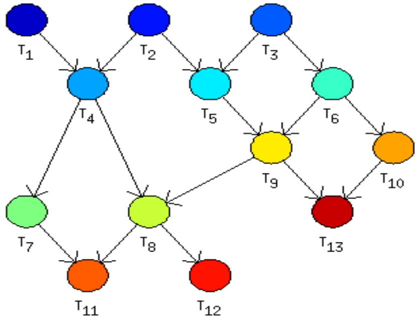
The Exascale challenge for Future Software?

Harrod SC12: “today’s bulk synchronous (BSP), distributed memory, execution model is approaching an efficiency, scalability, and power wall.”

Compute

Communicate

Compute

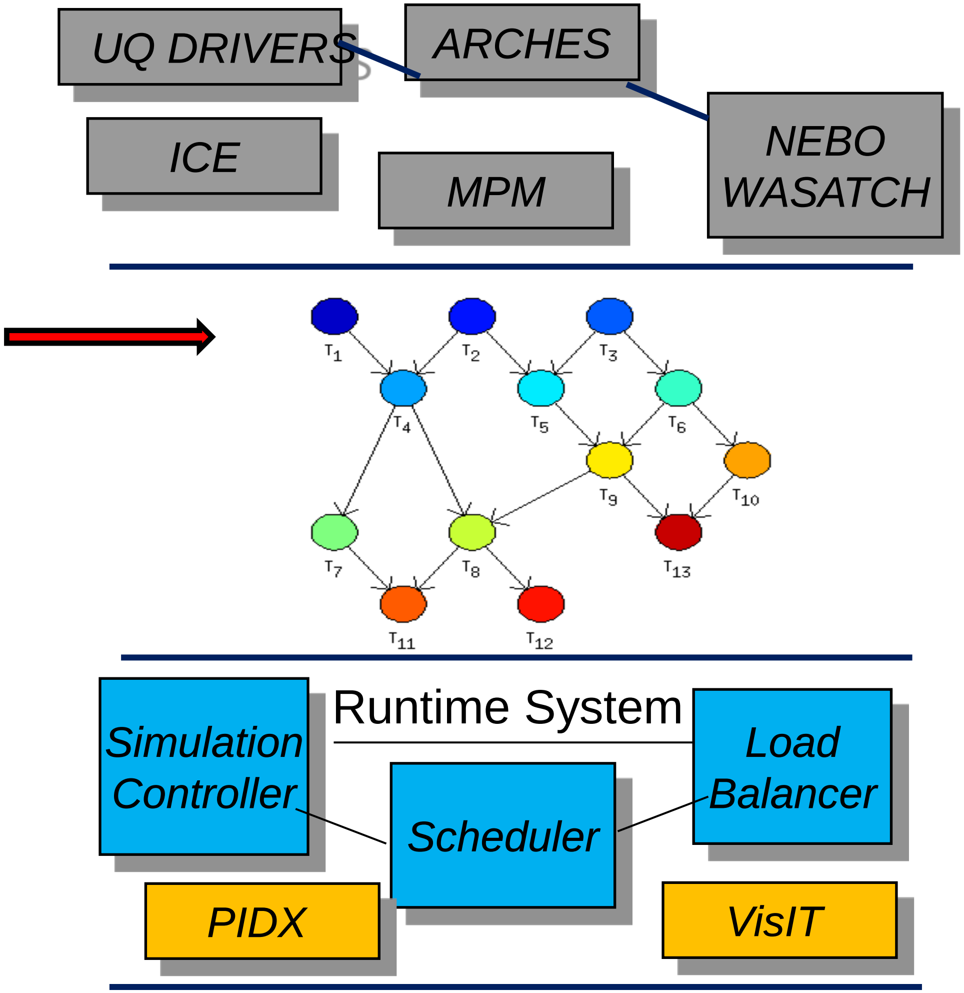


Sarkar et al. “Exascale programming will require prioritization of critical-path and non-critical path tasks, adaptive directed acyclic graph scheduling of critical-path tasks, and adaptive rebalancing of all tasks……”

“ DAG Task-based programming has always been a bad idea. It was a bad idea when it was introduced and it is a bad idea now “ Parallel Processing Award Winner

- **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**

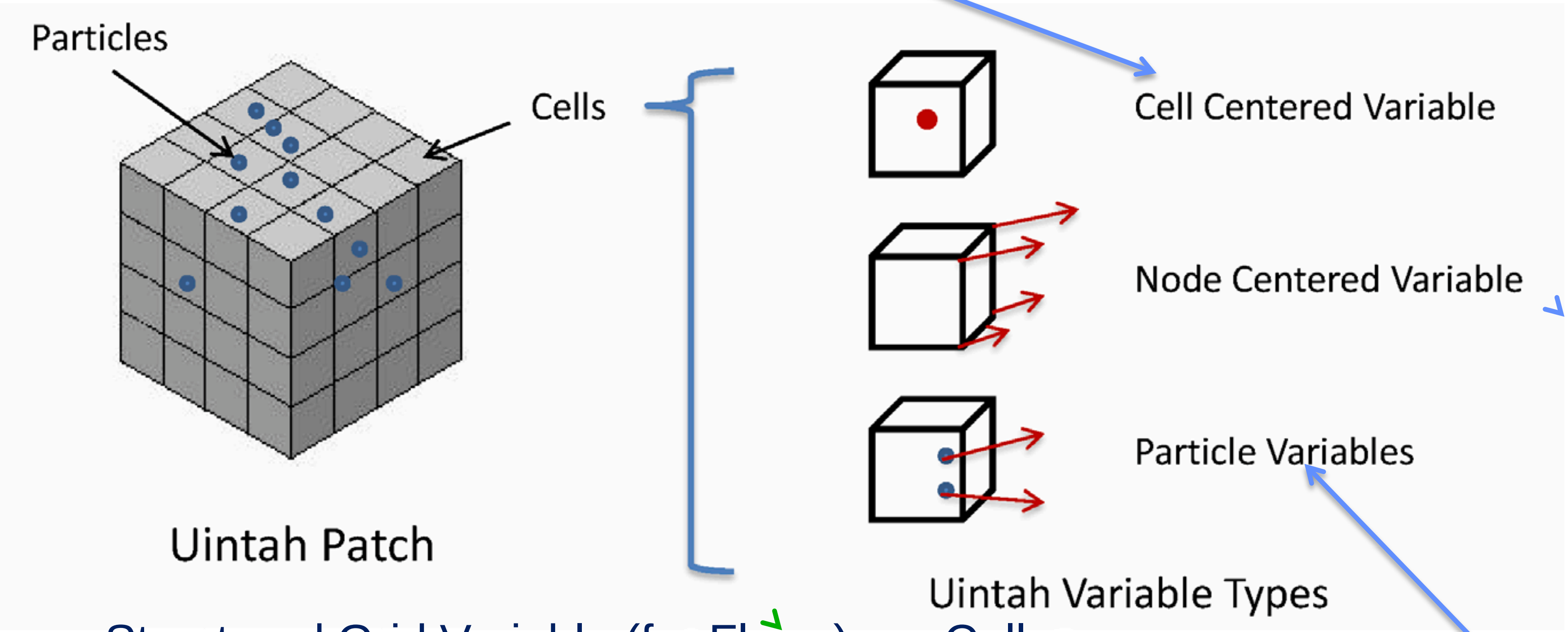


Uintah(X) Architecture Decomposition

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 and Variables

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



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

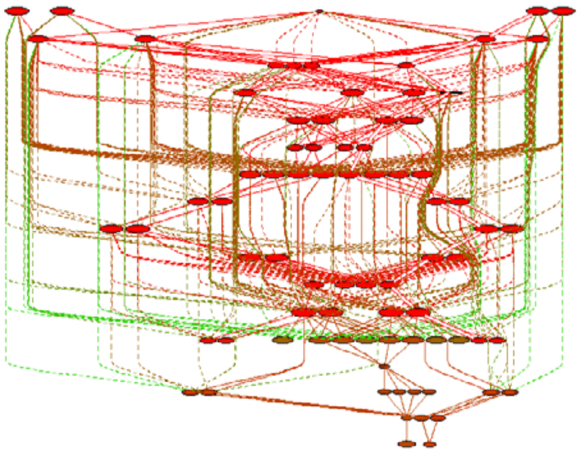
ARCHES is a combustion code using several different radiation models and linear solvers

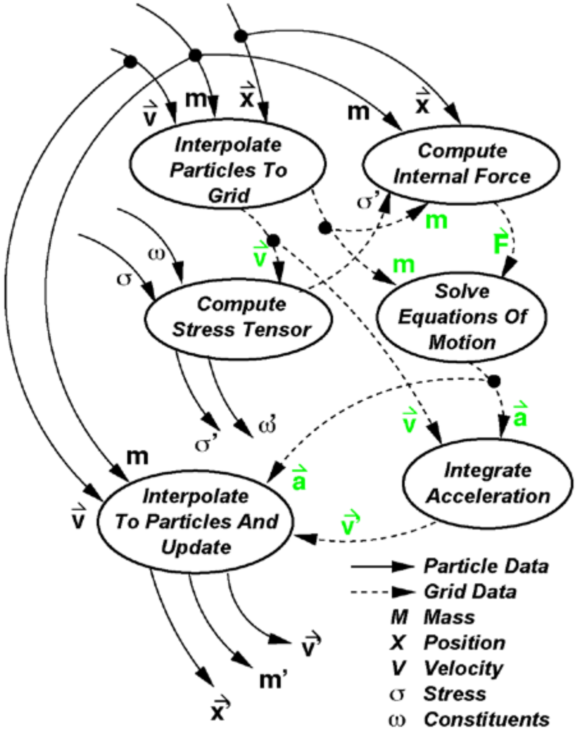
MPM is a novel method that uses particles and nodes

Exchange data with ICE, not just boundary condition

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

Uintah DAG :Directed Acyclic (Task) Graph-Based Computational Framework





Each task defines its computation with required inputs and outputs

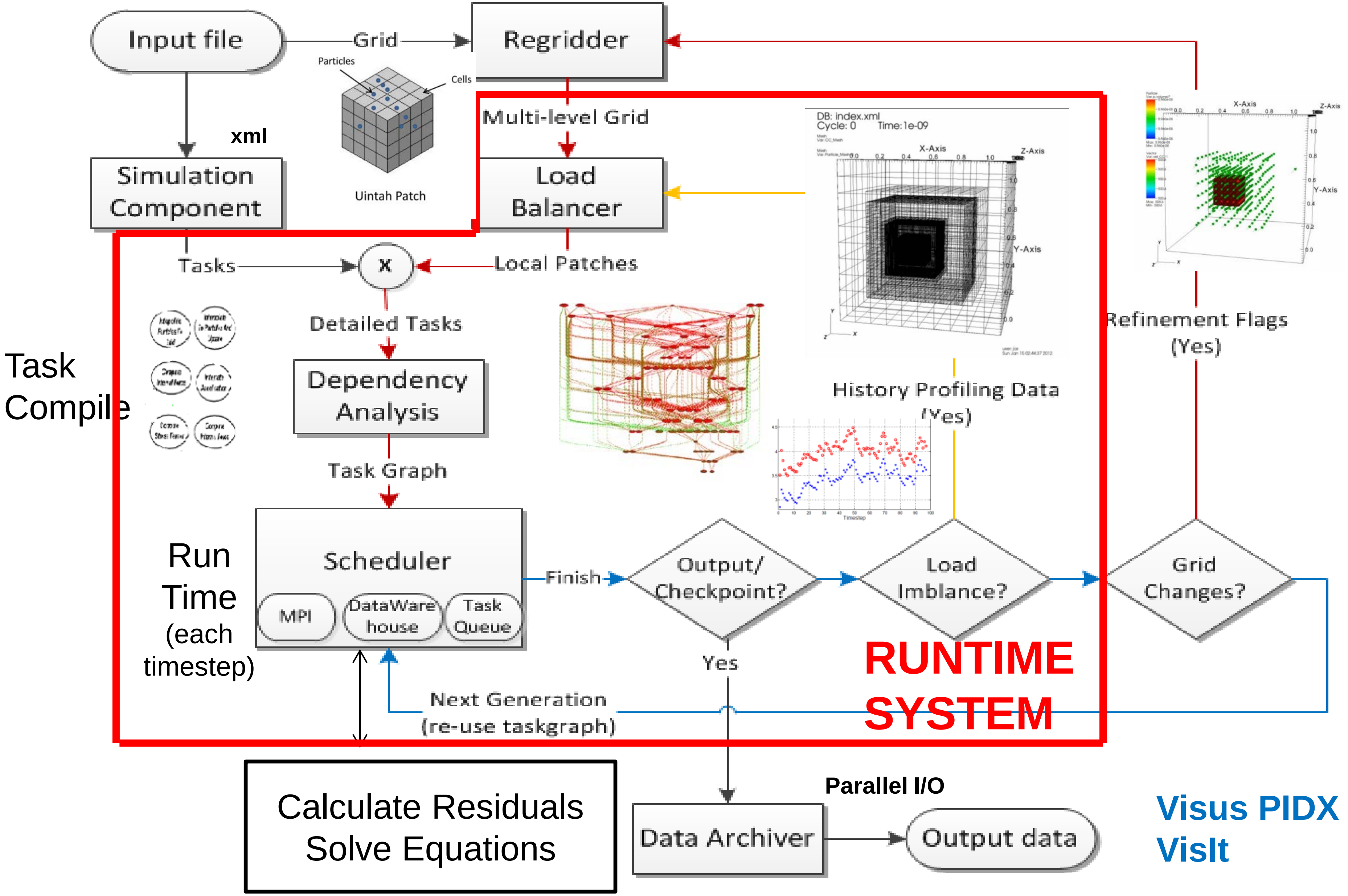
Uintah uses this information to create a task graph of computation (nodes) + communication (along edges)

Tasks do not explicitly define communications but only what inputs they need from a data warehouse and which tasks need to execute before each other.

Communication is overlapped with computation

Taskgraph is executed adaptively and sometimes out of order

ARCHES or WASATCH/NEBO



Input file

Grid

Regridder

Particles

Cells

Uintah Patch

xml

Simulation Component

Multi-level Grid

Load Balancer

DB: index.xml
Cycle: 0 Time:1e-09

Tasks

X

Local Patches

Detailed Tasks

Dependency Analysis

Task Graph

Task Compile

Refinement Flags (Yes)

History Profiling Data (Yes)

Run Time (each timestep)

Scheduler

MPI

DataWarehouse

Task Queue

Finish

Output/ Checkpoint?

Load Imblance?

Grid Changes?

Yes

Next Generation (re-use taskgraph)

RUNTIME SYSTEM

Calculate Residuals
Solve Equations

Parallel I/O

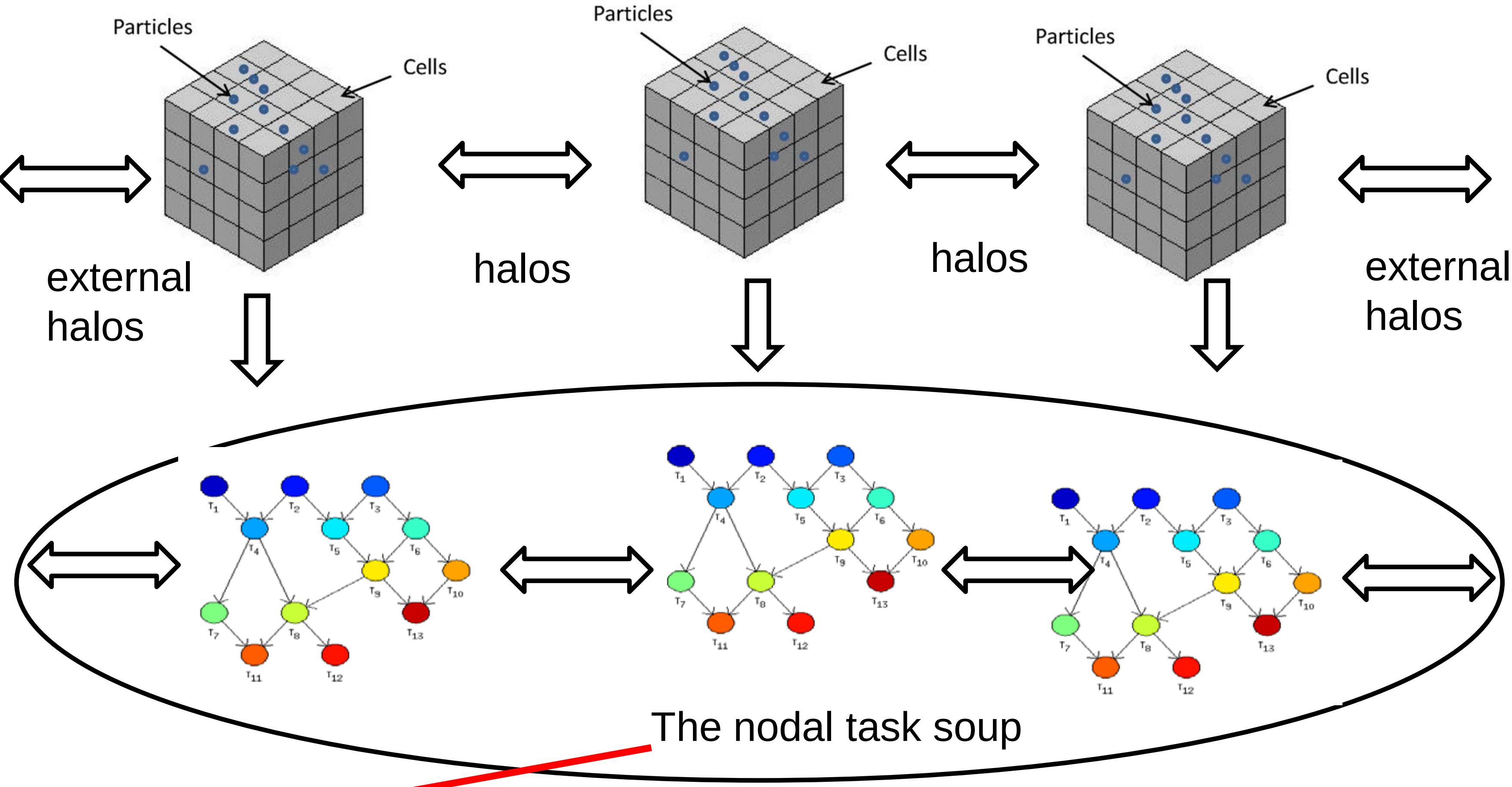
Data Archiver

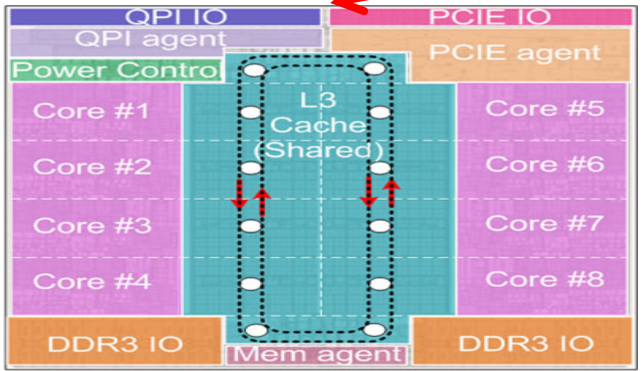
Output data

Visus PIDX
VisIt

UINTAH ARCHITECTURE

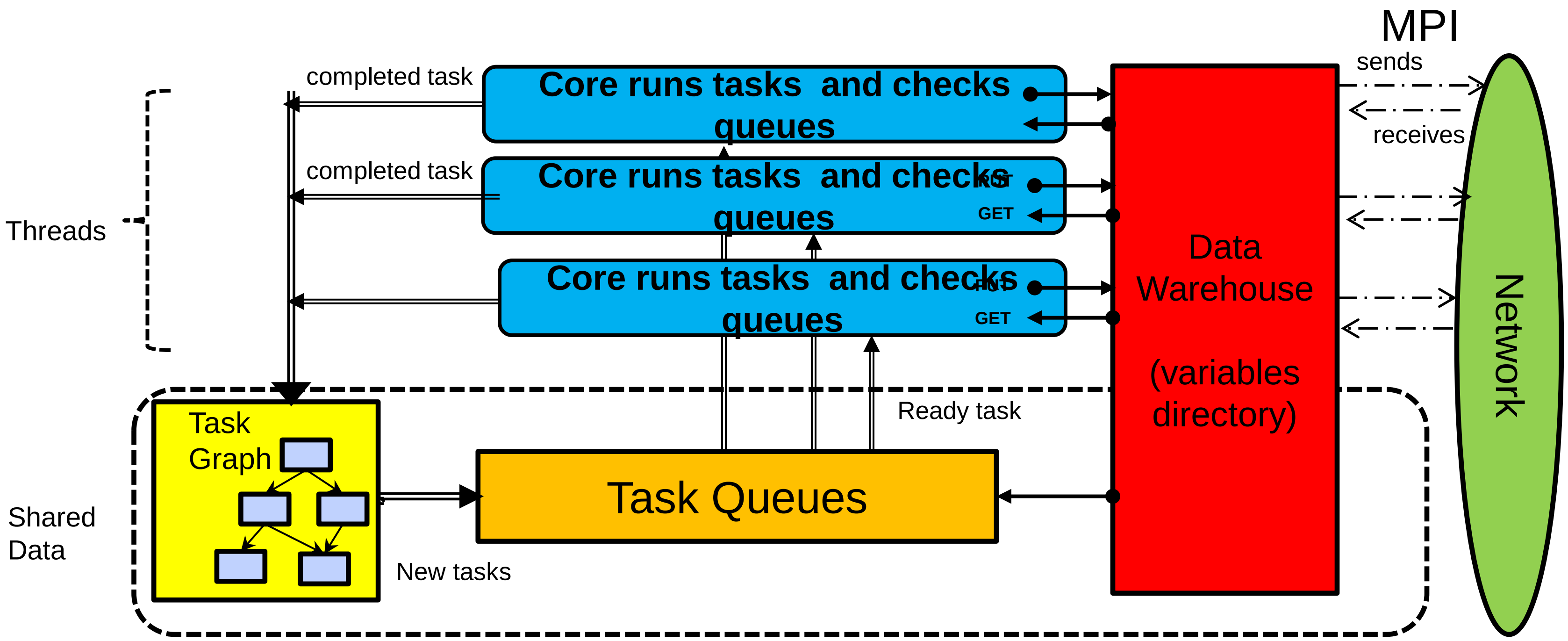
Task Graph Structure on a Multicore Node with multiple patches





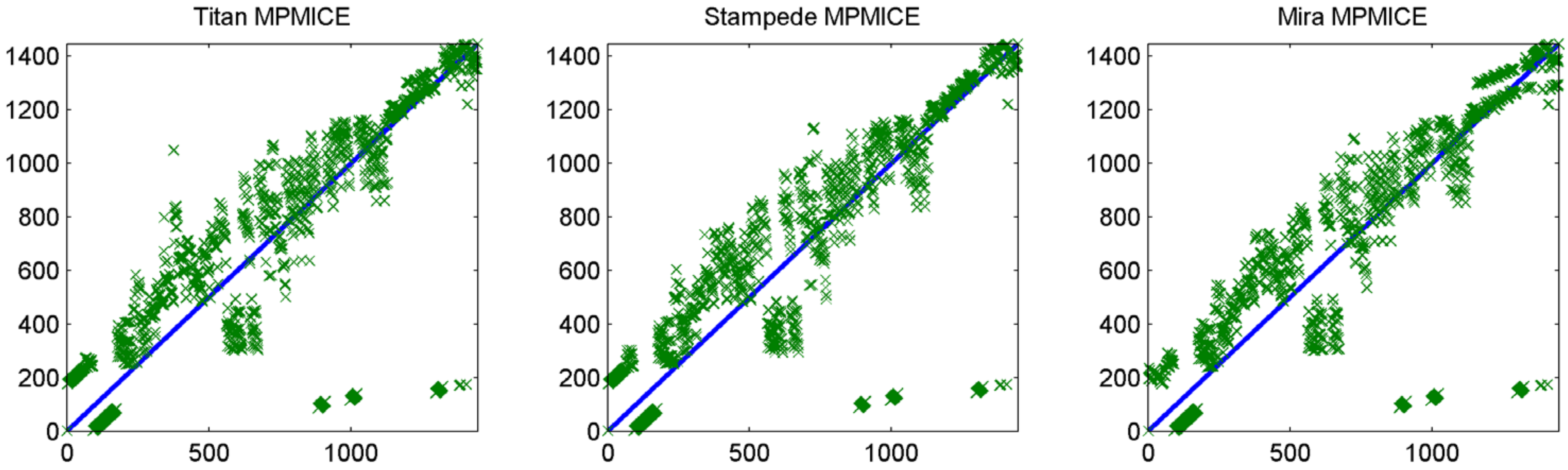
This is not a single graph. Multiscale and Multi-Physics merely add flavor to the “soup”. There are many adaptive strategies and tricks that are used in the execution of this graph soup.

Thread/MPI Scheduler (De-centralized)



- One MPI Process per Multicore node
- All threads directly pull tasks from task queues execute tasks and process MPI sends/receives
- Tasks for one patch may run on different cores
- One data warehouse and task queue per multicore node
- Lock-free data warehouse enables all cores to access memory quickly via atomic operations

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.

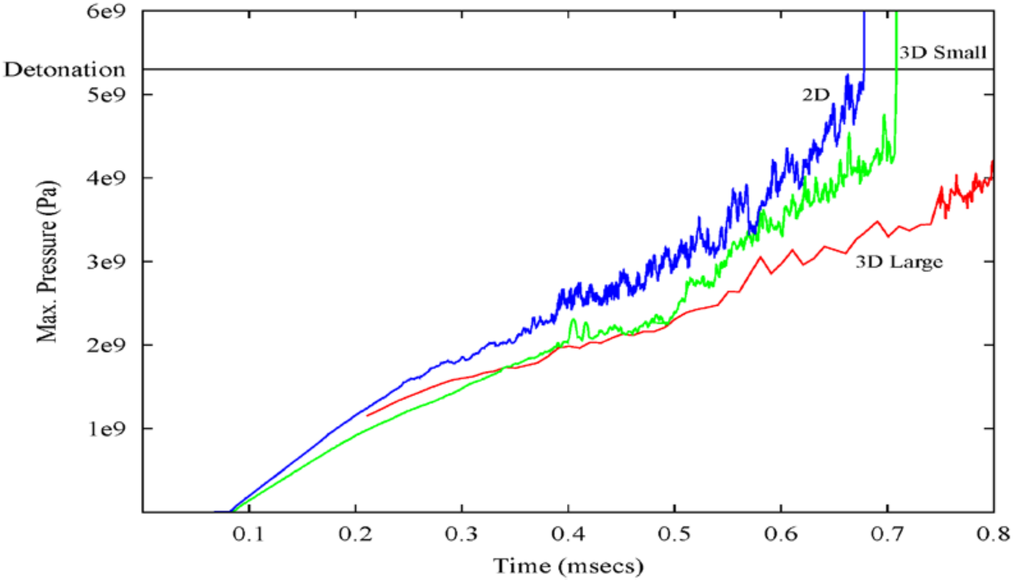
TASKS: 1 2 3 4 5 ⇒ 1 4 2 3 5

Early Late execution

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

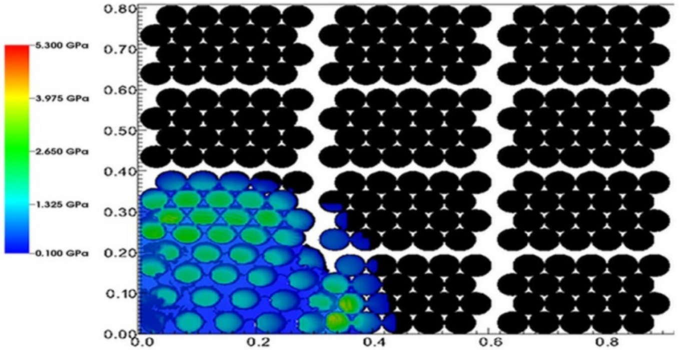
Speeding truck with 8000 explosive boosters each with 2.5-5.5 lbs of explosive overturned and caught fire

Experimental evidence for a transition from deflagration to detonation?

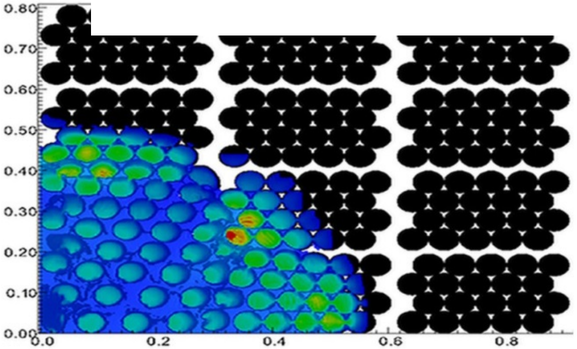


Deflagration wave moves at ~400m/s not all explosive consumed. Detonation wave moves 8500m/s all explosive consumed.

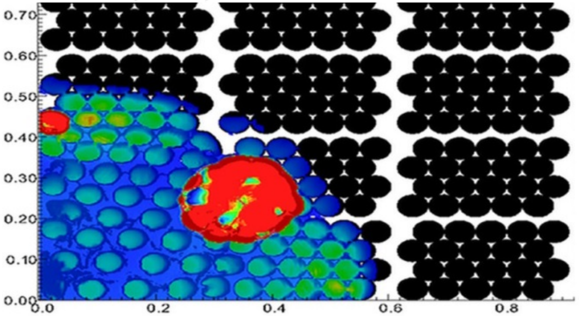
2013 Incite 200m cpu hrs



0.534 msec



0.634 msec

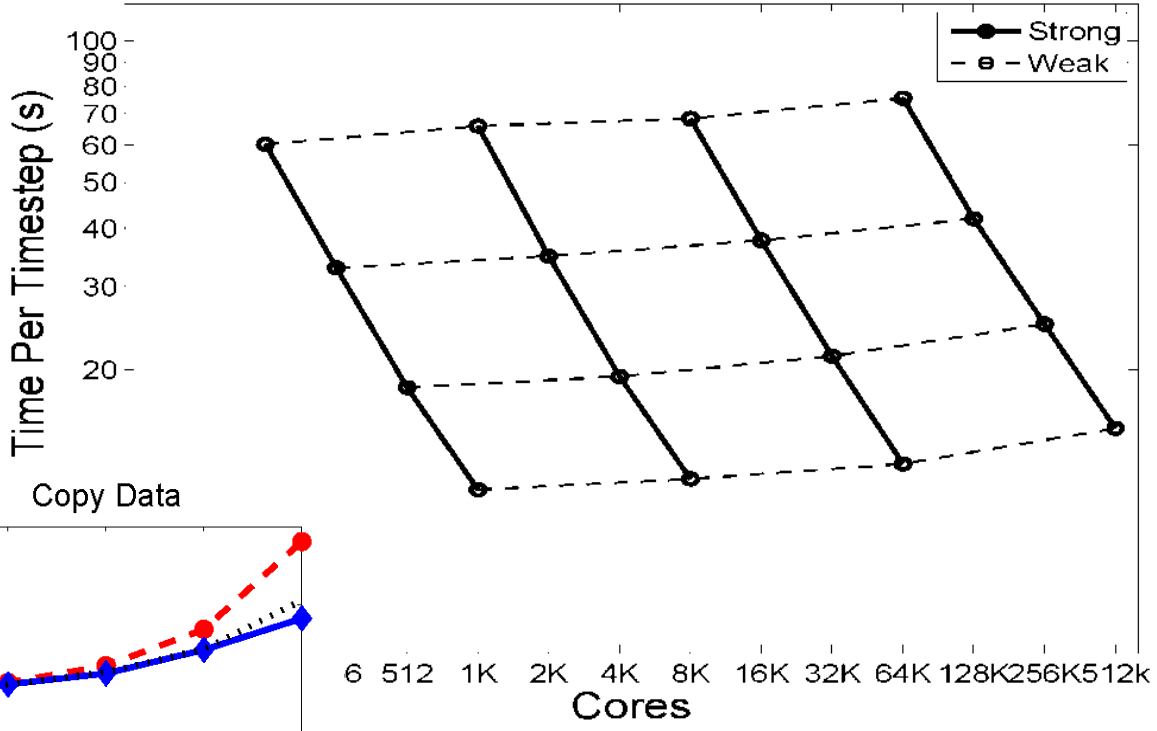


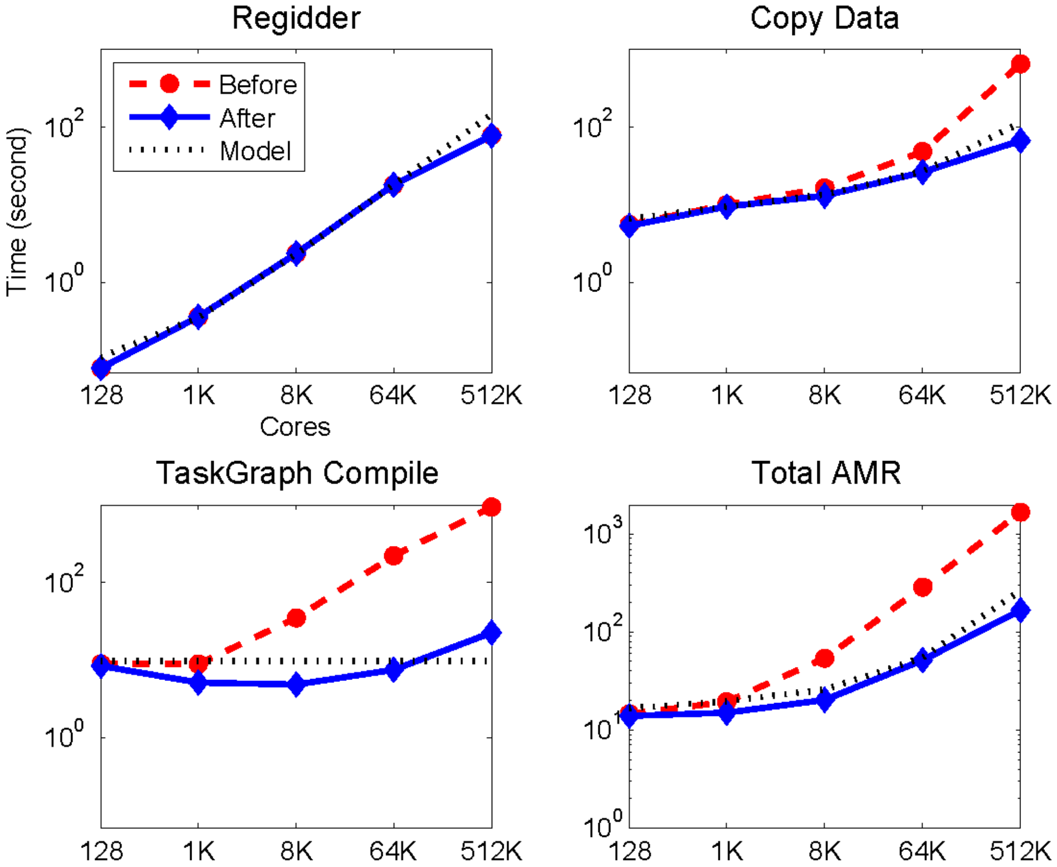
0.644 msec

Spanish Fork Accident

500K mesh patches
1.3 Billion mesh cells
7.8 Billion particles

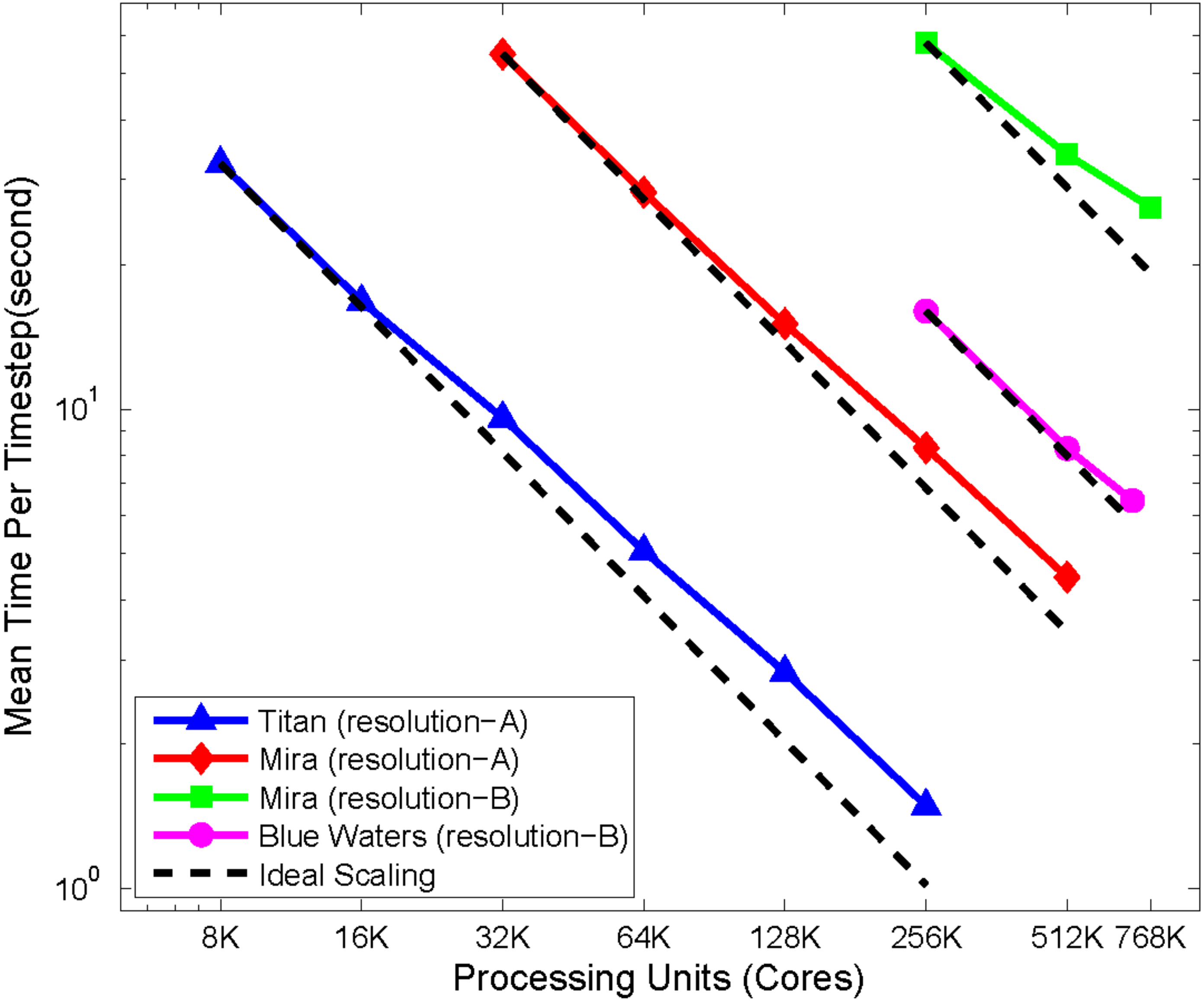
Detonation MPMICE: Scaling on Mira BGQ





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

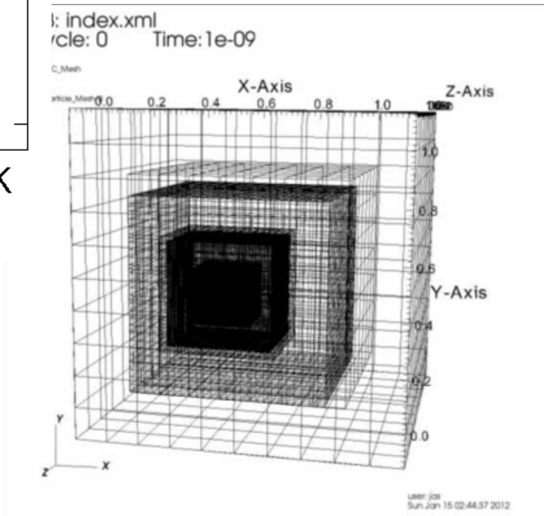
Scalability at one level is no certain
Indicator fro problems or machines
An order of magnitude larger



MPM AMR ICE Strong Scaling

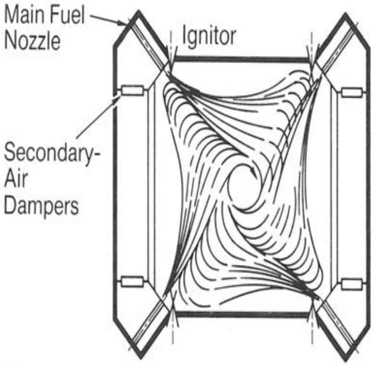
Mira DOE BG/Q 768K cores
Blue Waters Cray XE6/XK7 700K+ cores

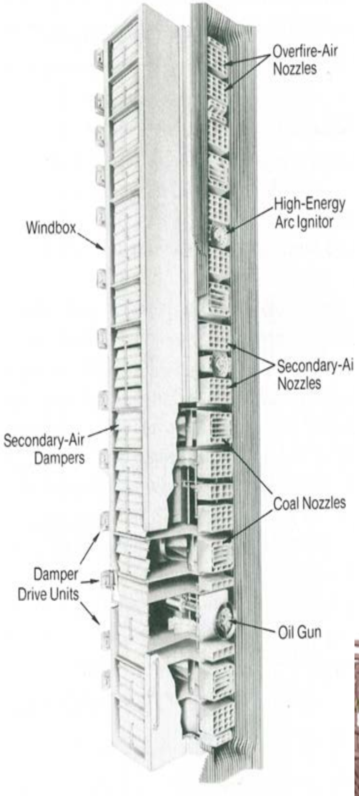
Resolution B
29 Billion particles
4 Billion mesh cells
1.2 Million mesh patches

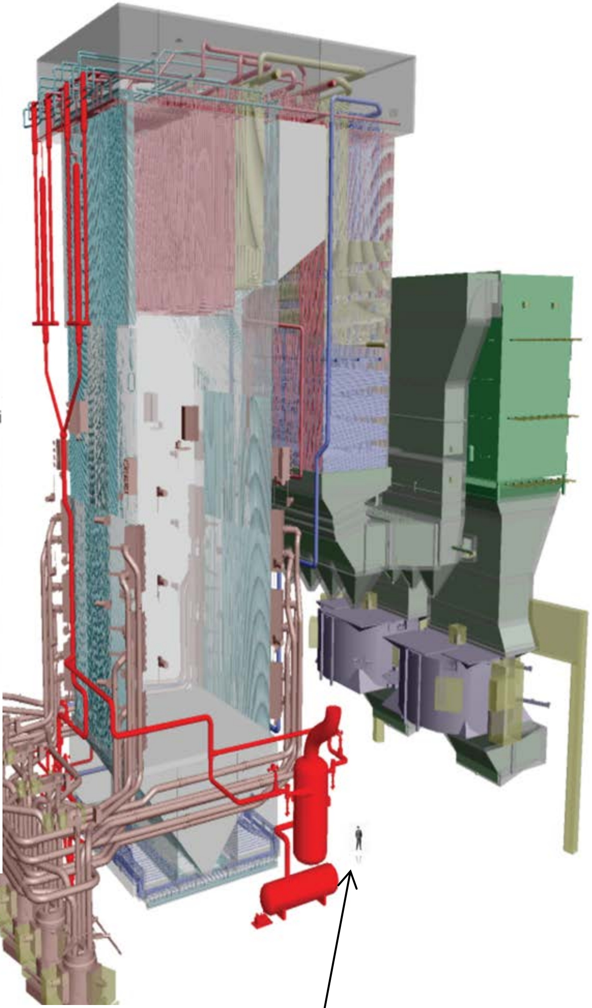


Complex fluid-structure interaction problem
with adaptive mesh refinement, see SC13/14 paper
NSF funding.

An Exascale Design Problem - Alstom Clean Coal Boilers







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 Navier –Stokes Equations

$$\frac{\partial \rho}{\partial t} + \nabla.\rho u = 0,$$

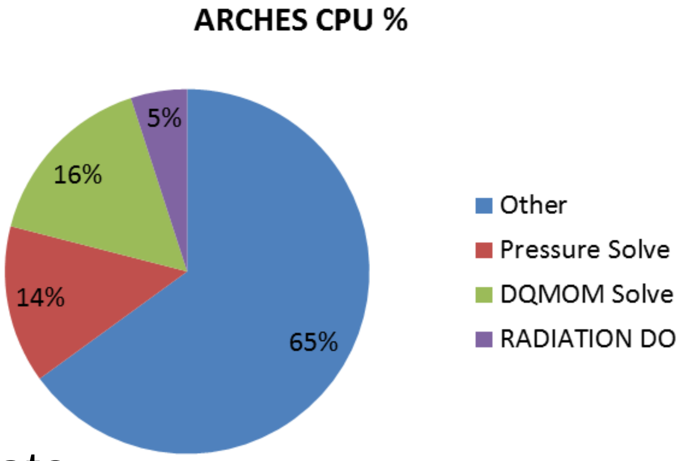
Full model includes turbulence, chemical reactions and radiation

where ρ is density, u is velocity vector and p is pressure

$$\frac{\partial \rho u}{\partial t} = F - \nabla p, \quad \text{where} \quad F = -\nabla.\rho uu + \nu\nabla^2 u + \rho g$$

Arrive at pressure Poisson equation to solve for *p*

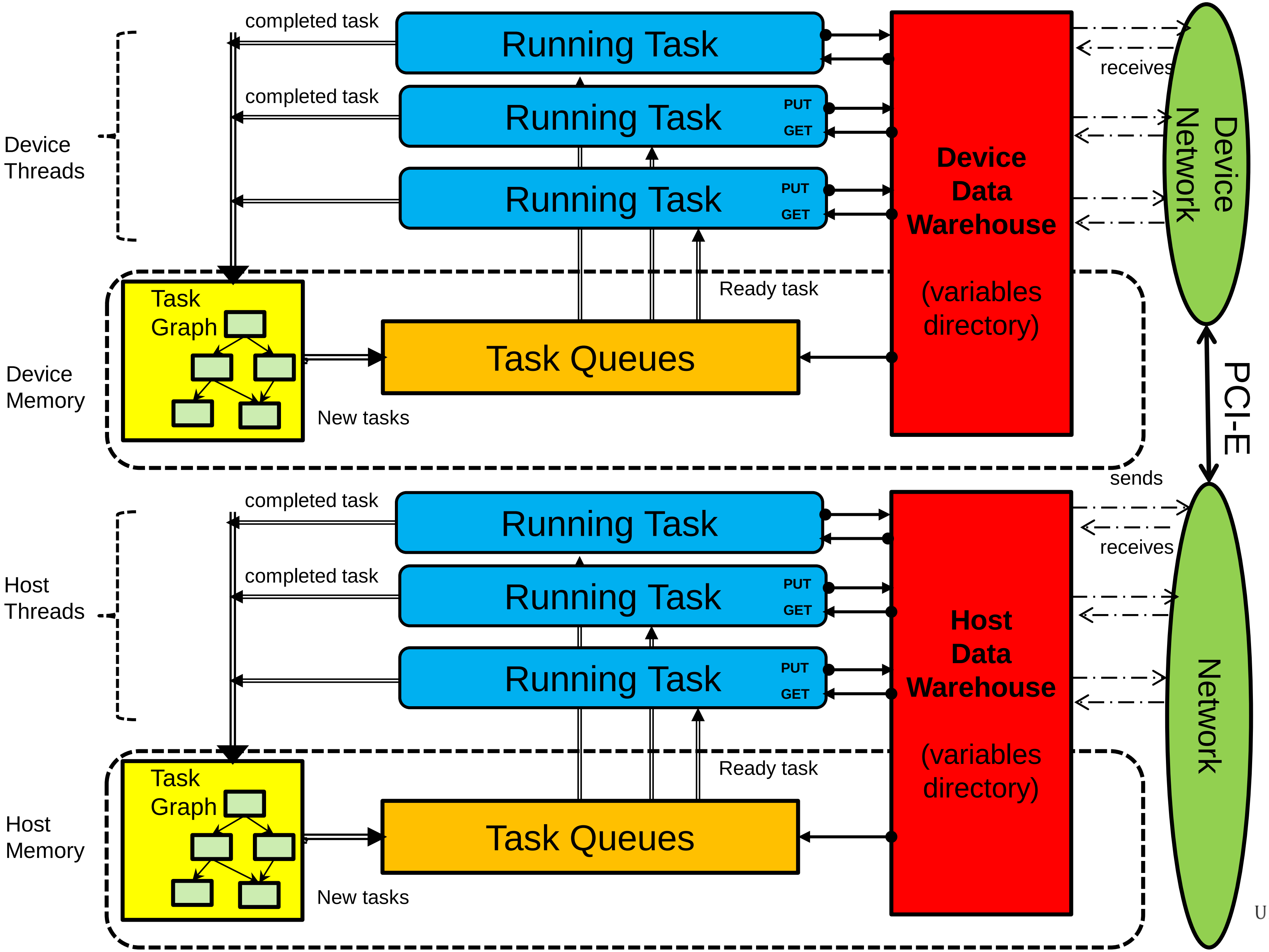
$$\nabla^2 p = R, \quad \text{where } R = \nabla.F + \frac{\partial^2 p}{\partial t^2}$$



Use Hype Solver distributed by LLNL
Many linear solvers inc. Preconditioned Conjugate Gradients on regular mesh patches used
Multi-grid pre-conditioner used
Careful adaptive strategies needed to get scalability
CCGrid13 paper.

One radiation solve
Every 10 timesteps

Unified Heterogeneous Scheduler (GPU or Phi symmetric)



THE UNIVERSITY OF UTAH

NEBO/Wasatch Example

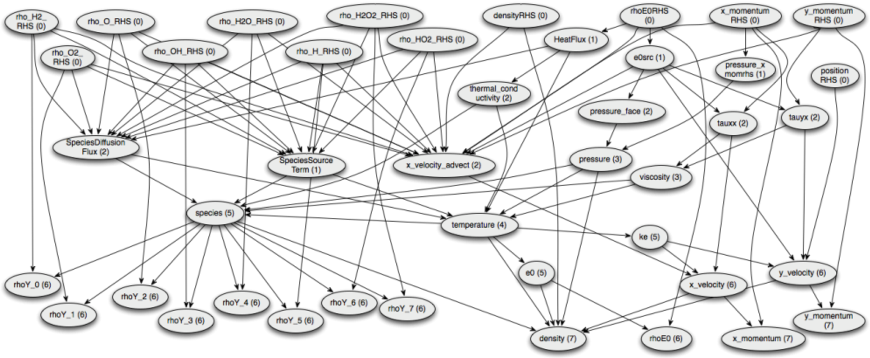
Express complex pde functions as DAG - automatically construct algorithms from expressions

Define field operations needed to execute tasks (fine grained vector parallelism on the mesh)

User writes only field operations code . Supports field & stencil operations directly - no more loops!

Strongly typed fields ensure valid operations at compile time. *Allows a variety of implementations to be tried without modifying application code.*

Scalability on a node - use **Uintah** infrastructure to get scalability across whole system



[Sutherland Earl Might]

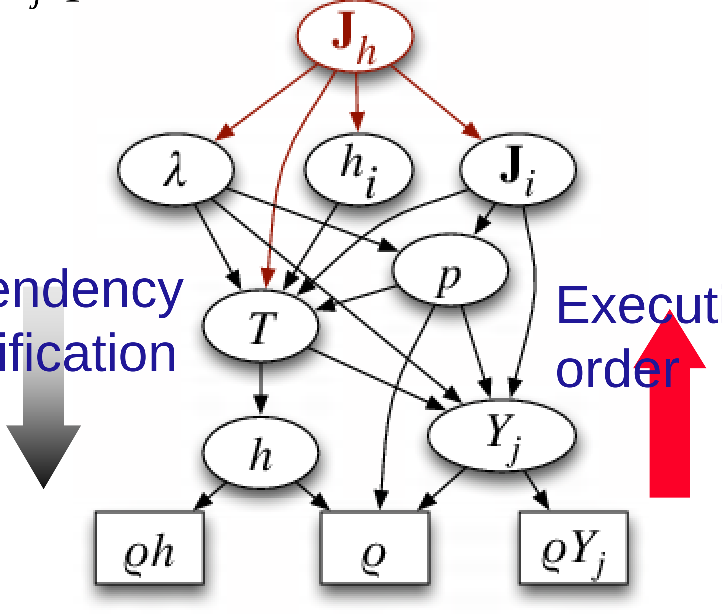
Energy equation

$$\frac{\partial \rho e}{\partial t} + \nabla.(\rho e \underline{u}) + \nabla.\underline{J}_h + terms = 0$$

Enthalpy diffusive flux

$$\underline{J}_h = -\lambda(T, Y_j)\nabla T - \sum_{i=1}^{n} h_i \underline{J}_i$$

$$\underline{J}_i = -\sum_{j=1}^{ns} D_{ij}(T, Y_j)\nabla Y_j - D_i^T(T, Y_j)\nabla T$$

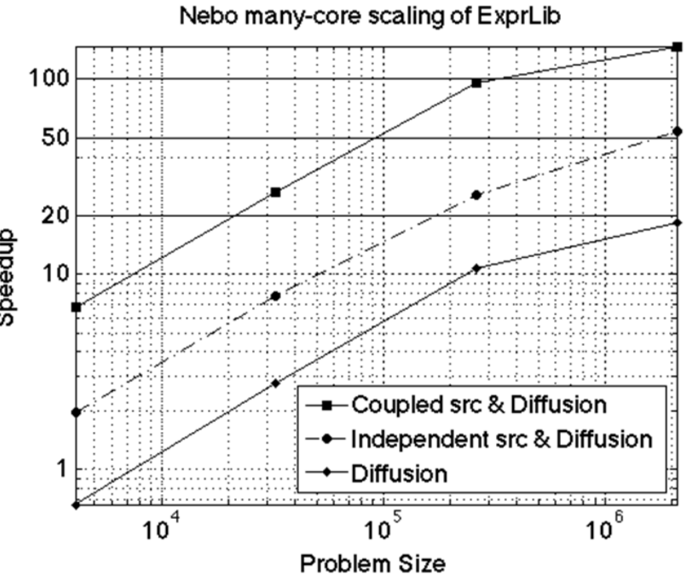


Dependency specification

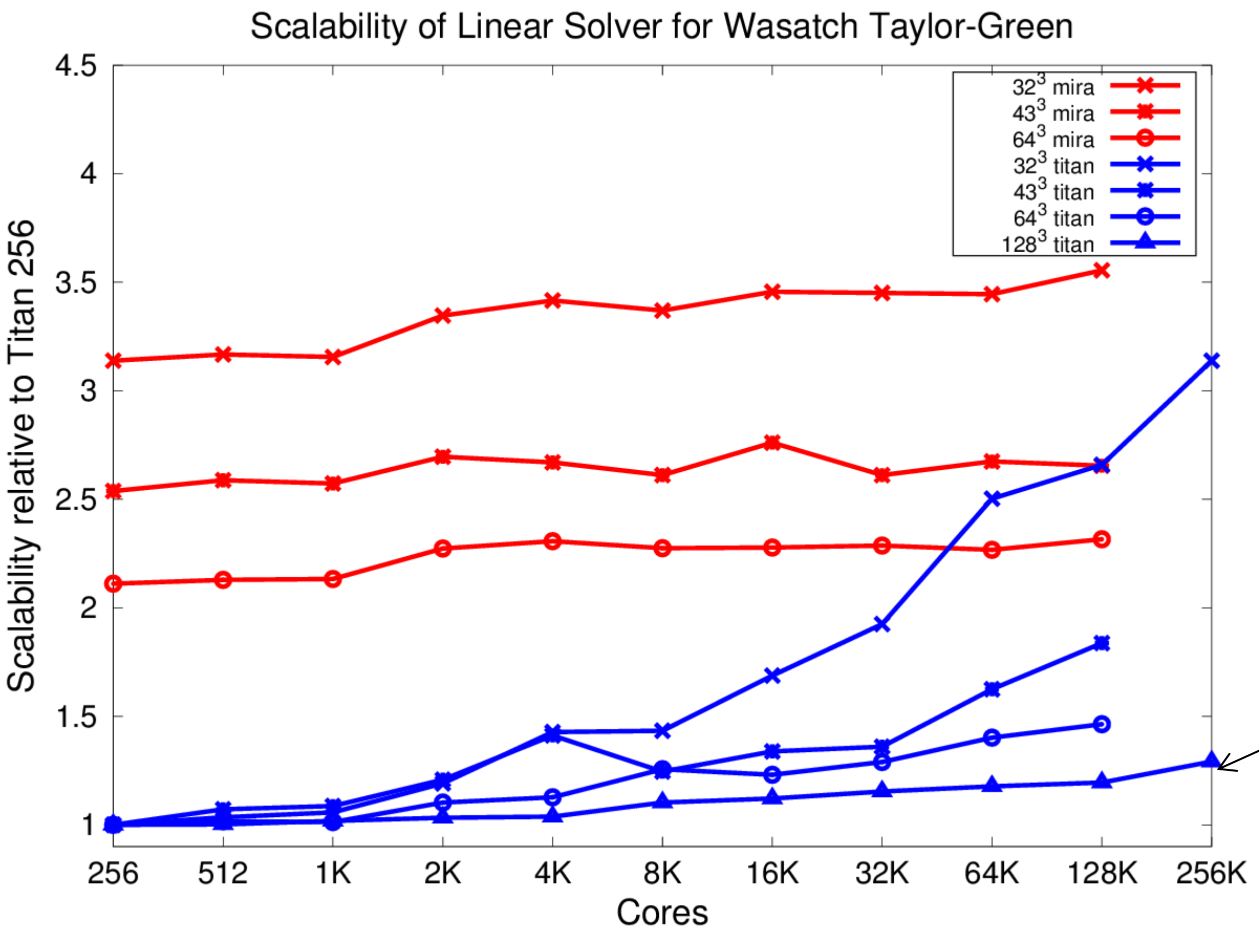
Execution order

Wasatch – Nebo Recent Milestones

- Wasatch is solving (nonreacting miniboiler~3-4x speedup over the non-DSL approach.
- New Nebo backend for CPU resultied in 20-30% speedup in the entire Wasatch code base.
- Much of the Wasatch code base is GPU-ready
- Arches plus SpatialOps & Nebo EDSL being scoped.



Good GPU scaling with (>32^3 per patch).
Loop fusion (heavy GPU kernels) needed e.g “coupled source & diffusion”

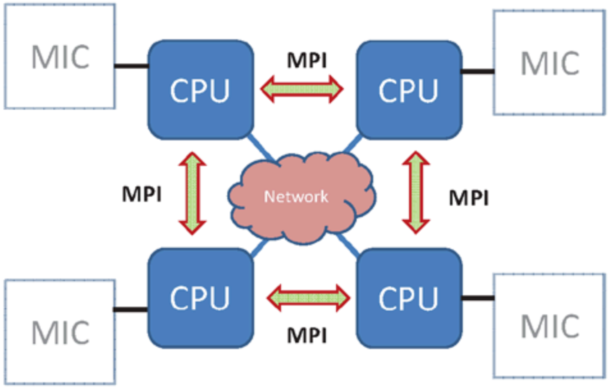


2.2 Trillion DOF

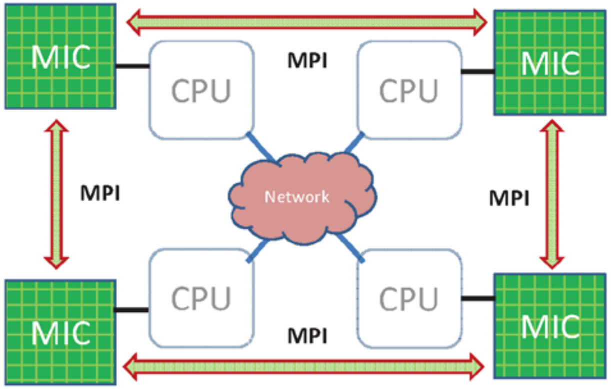
Each **Mira Run** is scaled wrt the **Titan Run at 256 cores**
Note these times are not the same for different patch sizes.

Weak Scalability of Hypre Code

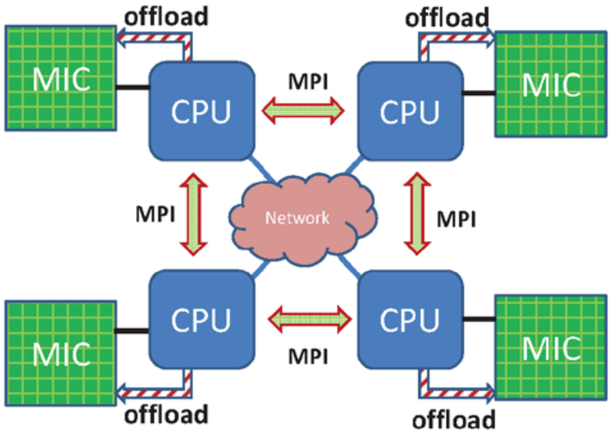
Xeon Phi Execution Models



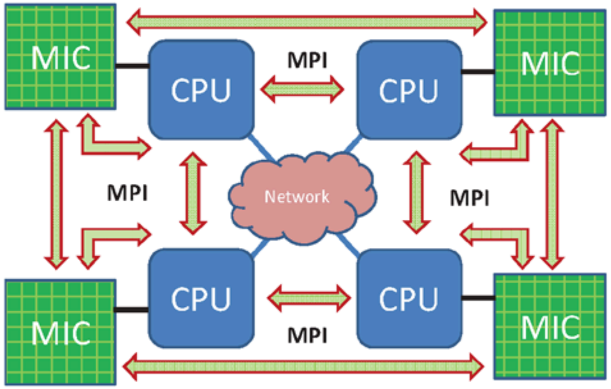
(1) Host-only Model



(2) MIC Native Model



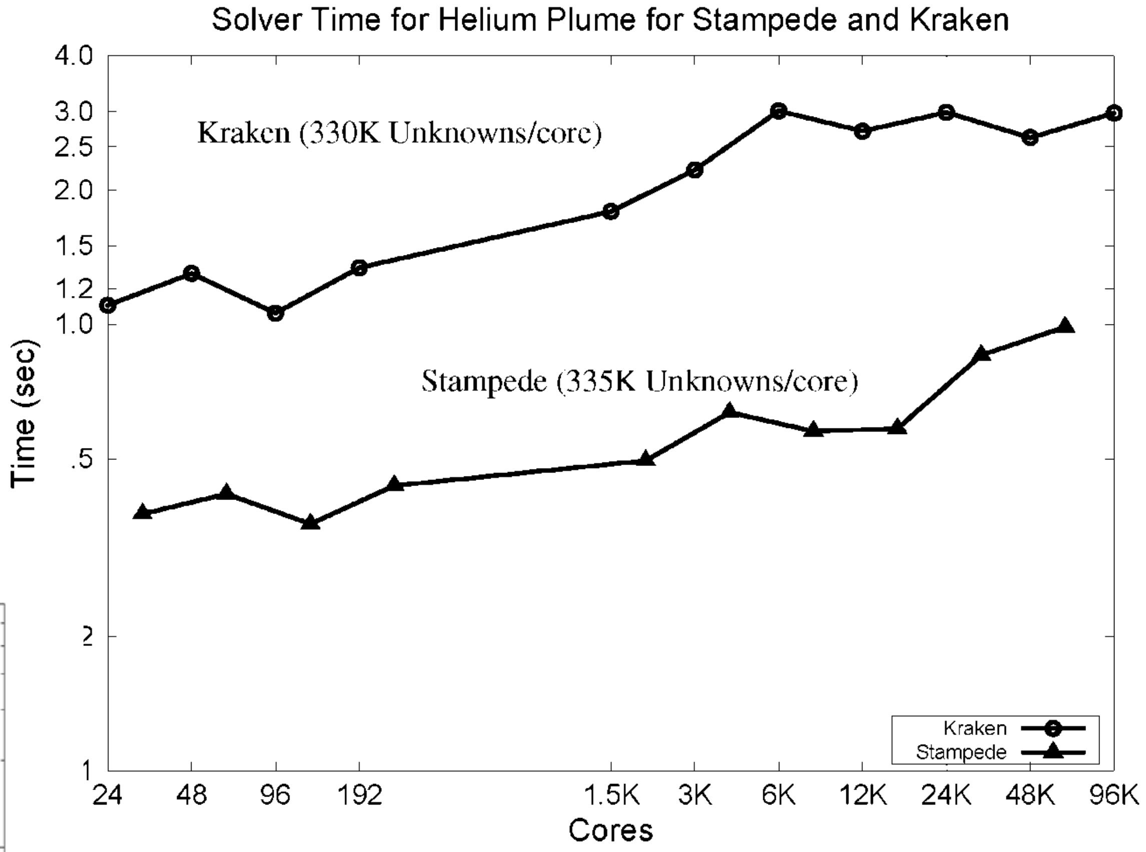
(3) Offload Model



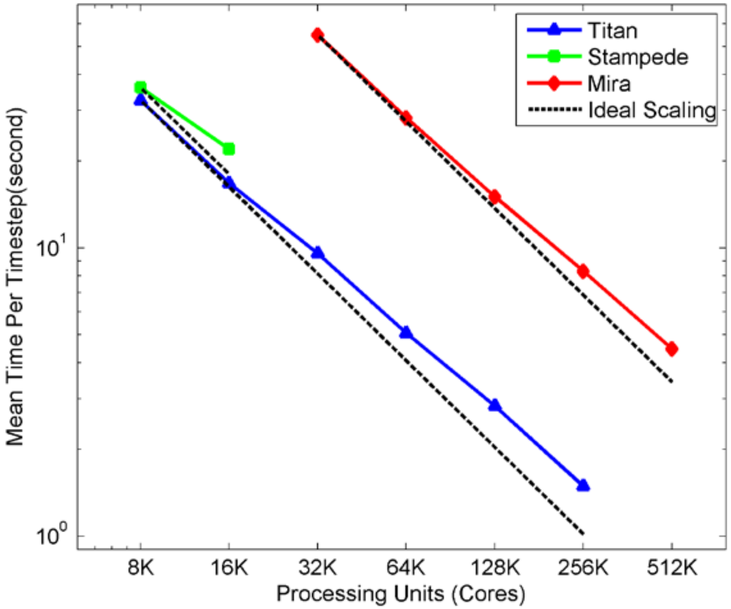
(4) Symmetric Model

THE UNIVERSITY OF UTAH

Uintah on Stampede: Host-only Model



Solver Time for Helium Plume for Stampede and Kraken
Kraken (330K Unknowns/core)
Stampede (335K Unknowns/core)
Time (sec)
4.0
3.0
2.5
2.0
1.5
1.2
1.0
.5
2
1
24
48
96
192
1.5K
3K
6K
12K
24K
48K
96K
Cores
Kraken
Stampede



- Using Hypre with a conjugate gradient solver
- Preconditioned with geometric multi-grid
- Red Black Gauss Seidel relaxation - each patch

THE UNIVERSITY OF UTAH

Uintah on Stampede: Offload Model

- Use compiler directives (#pragma)
  - Offload target: #pragma offload target(mic:0)
  - OpenMP: #pragma omp parallel
- Find copy in/out variables from task graph
- Functions called in MIC must be defined with __attribute__((target(mic)))
- Hard for Uintah to use offload mode
  - Rewrite highly templated C++ methods with simple C/C++ so they can be called on the Xeon Phi
  - Less effort than GPU port, but still significant work for complex code such as Uintah with 800K lines of code.

Uintah on Stampede: Symmetric Model

Xeon Phi directly calls MPI

Use Pthreads on both host CPU and Xeon Phi:

1 MPI process on host – 16 threads

1 MPI process on MIC – up to 120 threads

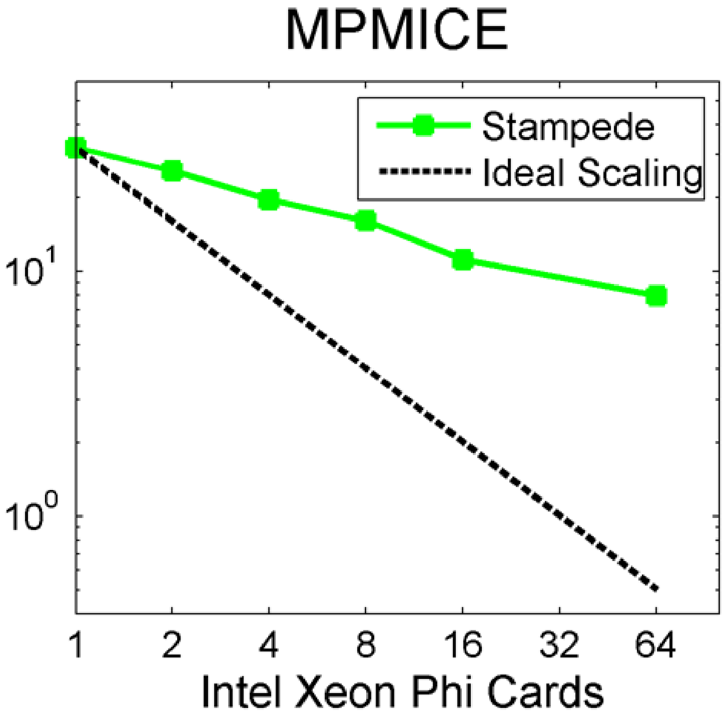
Same example as previously.

Multi MIC Cards (Symmetric Model)

Xeon Phi card: 60 threads per MPI process,   2 MPI processes

host CPU :16 threads per MPI process,   1 MPI process

Issue: load imbalance - profiling differently on host and Xeon Phi



DESIGNING FOR EXASCALE

Clear trend towards accelerators e.g. GPU but also Intel MIC – NSF "Stampede" Balance factor = flops/bandwidth – high.PORTABILITY IS THE KEY ISSUE:NEW CODE - use Wasatch to generate code for GPUs and MICs .How do we handle the challenge of existing code?

Kokkos: A Layered Collection of Libraries

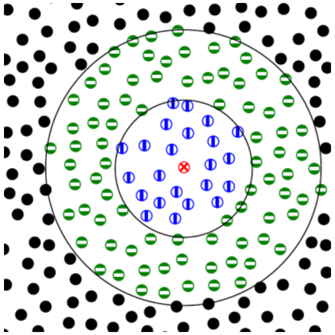
See [Carter Edwards and Dan Sunderland]

- **Standard C++, Not a language extension**
  - **In *spirit* of TBB, Thrust & CUSP, C++AMP, LLNL's RAJA, ...**
  - ***Not* a language extension like OpenMP, OpenACC, OpenCL, CUDA, ...**
- **Uses C++ template meta-programming**
- **Multidimensional Arrays, *with a twist***
  - **Layout mapping: multi-index (i,j,k,...) ↔ memory location**
  - ➢ **Choose layout to satisfy device-specific memory access pattern**
  - **Layout changes are invisible to the user code**

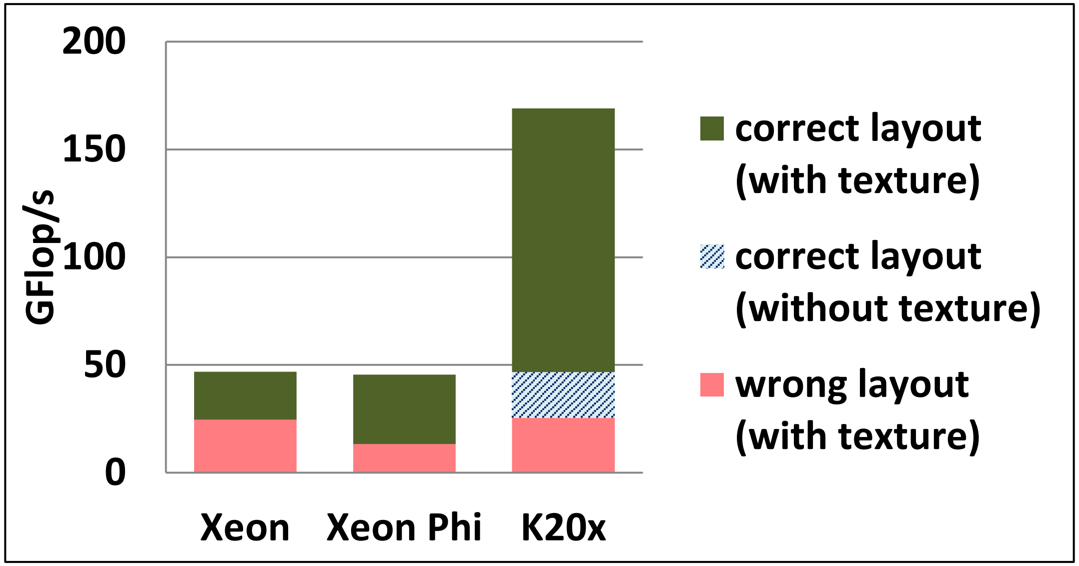
Evaluate Performance Impact of Array Layout [Edwards and Sunderland]

- Molecular dynamics computational kernel in miniMD
- Simple Lennard Jones force model: $F_i = \sum_{j,\, r_{ij} < r_{cut}} 6\varepsilon \left[\left(\frac{\varsigma}{r_{ij}}\right)^{7} - 2\left(\frac{\varsigma}{r_{ij}}\right)^{13} \right]$
- Atom neighbor list to avoid N^2 computations

```
pos_i = pos(i);
for( jj = 0; jj < num_neighbors(i); jj++) {
  j = neighbors(i,jj);
  r_ij = pos_i – pos(j); //random read 3 floats for pos
  if (|r_ij| < r_cut) f_i += 6*e*((s/r_ij)^7 – 2*(s/r_ij)^13)
}
f(i) = f_i;
```



- Test Problem
  - 864k atoms, ~77 neighbors
  - 2D neighbor array
  - Different layouts CPU vs GPU
  - Random read ‘pos’ through GPU texture cache
- **Large performance loss with wrong array layout**

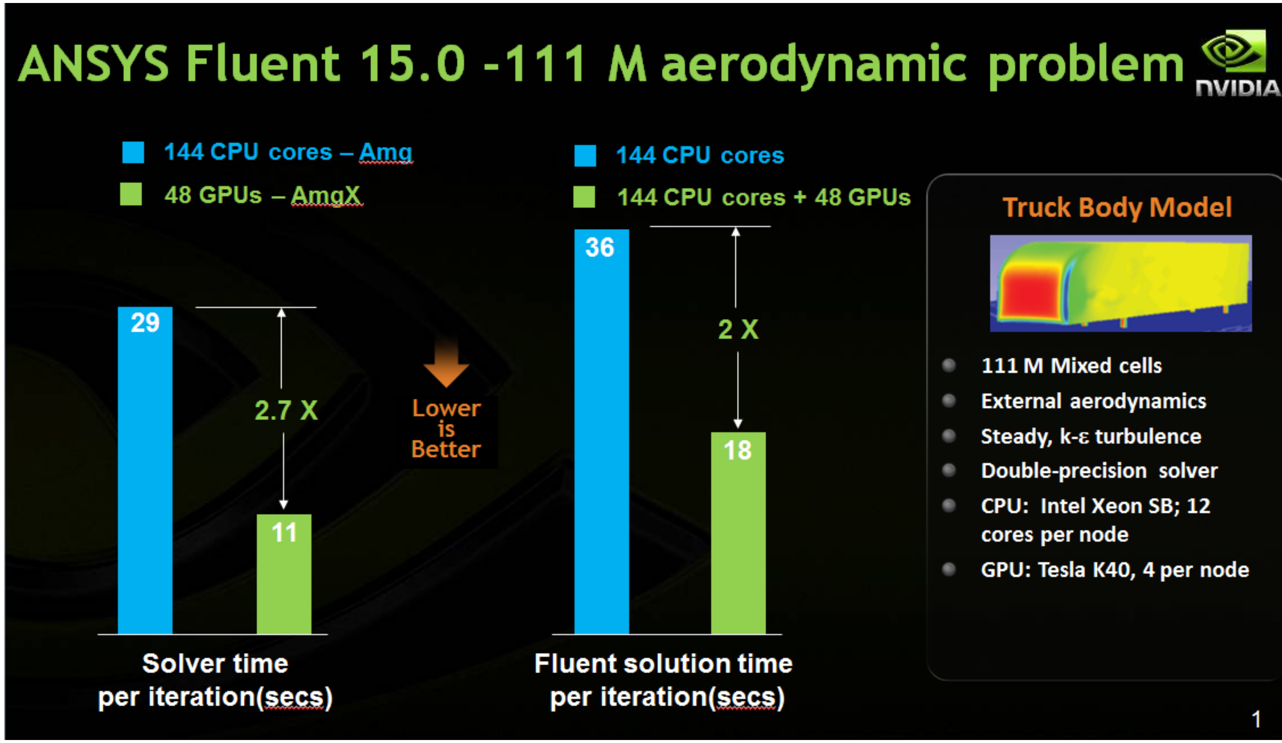


NVIDIA AMGX Linear Solvers on GPUs

- Fast, scalable iterative gpu linear solvers for packages e.g.,
- Flexible toolkit provides GPU accelerated Ax = b solver
- Simple API for multiple apps domains.
- Multiple GPUs (maybe thousands) with scaling

Key Features
Ruge-Steuben algebraic MG
Krylov methods: CG, GMRES, BiCGStab,
Smoothers and Solvers: Block- Jacobi, Gauss-Seidel, incomplete LU,

Flexible composition system
MPI support OpenMP support, Flexible and high level C API,



ANSYS Fluent 15.0 -111 M aerodynamic problem

Truck Body Model

- 111 M Mixed cells
- External aerodynamics
- Steady, k-ε turbulence
- Double-precision solver
- CPU: Intel Xeon SB; 12 cores per node
- GPU: Tesla K40, 4 per node

1

Free for non-commercial use
Utah access via Utah CUDA COE.

Summary

- DAG abstraction important for achieving scaling
- Layered approach very important for not needing to change applications code
- Scalability still requires much engineering of the runtime system.
- Obvious applicability to new architectures
- DSL approach very important for the future
- Kokkos very important for legacy codes
- MIC /GPU development ongoing
- The approach used here shows promise for very large core and MIC/GPU counts but using these architectures and future versions of them is an exciting challenge for our exascale problem. Future systems have mix of Intel Phi, GPU, IBM, Arm etc etc ?