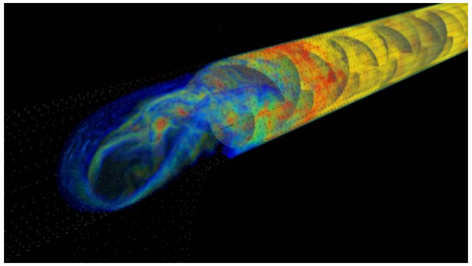
Multi-Scale and Multi-Physics Simulations on Present and Future Architectures



www.uintah.utah.edu

Martin Berzins

1. **Background and motivation**
2. **Uintah Software and Multicore Scalability**
3. **Runtime Systems for Heterogeneous Architectures**
4. **A Challenging Clean Coal Application**
5. **Conclusions and Portability for future Architectures Using DSLs and Kokkos**

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 teams in Utah

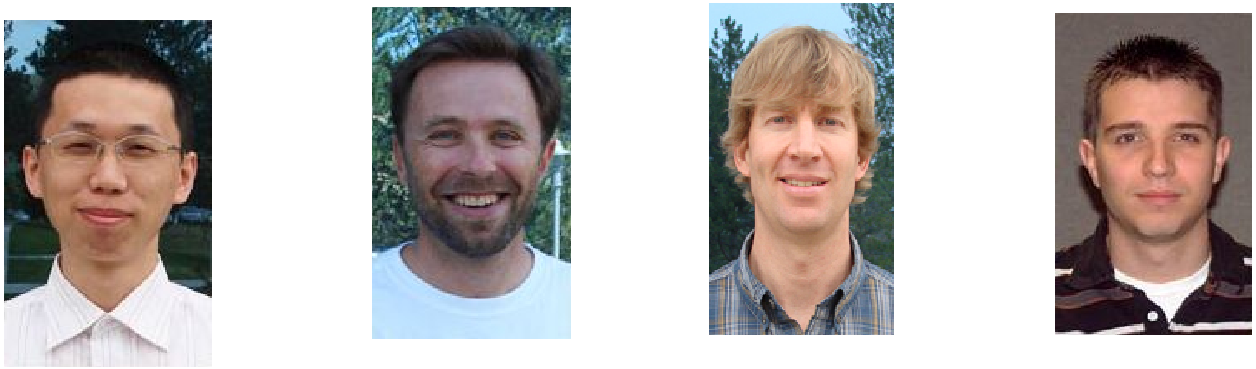
Energetic Materials: Chuck Wight, Jacqueline Beckvermit, Joseph Peterson, Todd Harman, Qingyu Meng NSF PetaApps 2009-2014 $1M, P.I. MB

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*,



* 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?

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

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….."

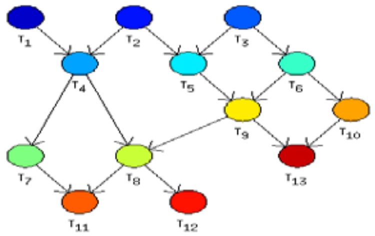
" DAG Task-based programming has always been a bad idea. It was a bad idea when it was introduced and it is a bad idea now " Parallel Proc. Award Winner

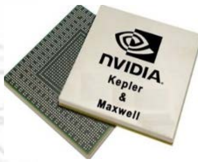
Much architectural uncertainty, many storage and power issues. Adaptive portable software needed

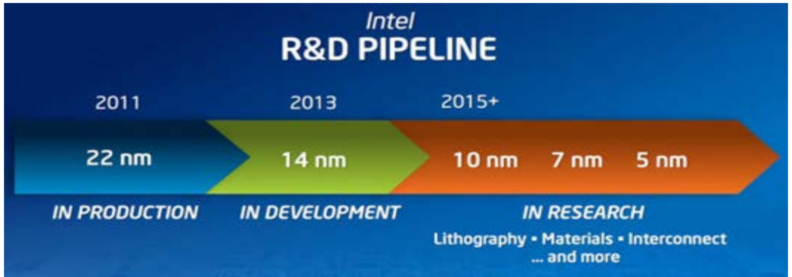
Compute

Communicate

Compute







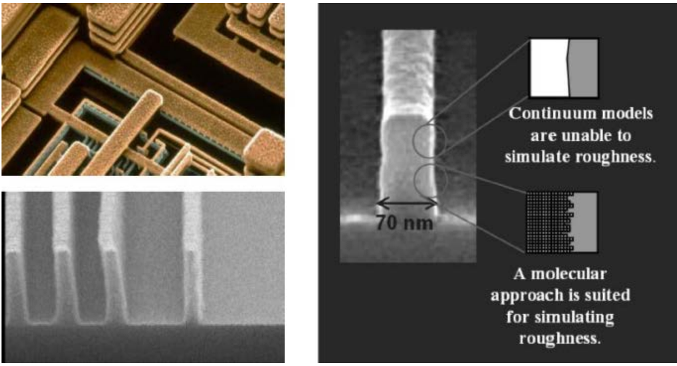
Intel
R&D PIPELINE

2011	2013	2015+		
22 nm	14 nm	10 nm	7 nm	5 nm
IN PRODUCTION	IN DEVELOPMENT	IN RESEARCH		

Lithography • Materials • Interconnect
... and more

Predictive Computational Science [Oden Karniadakis]

Predictive Computational (Materials) Science is changing e.g. nano-maufacturing

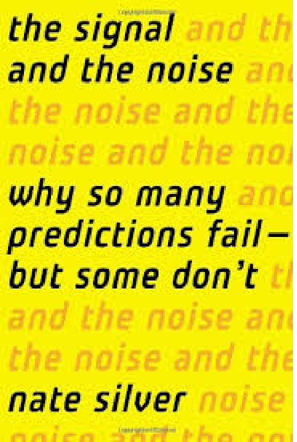


Science is based on subjective probability in which predictions must account for uncertainties in parameters, models, and experimental data . This involves many “experts” who are often wrong

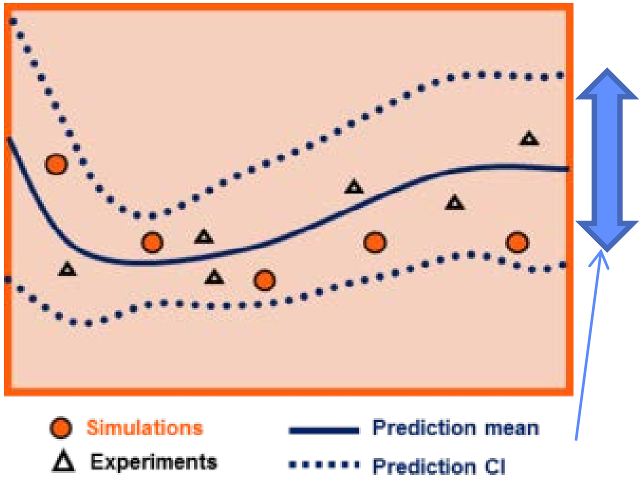
We cannot deliver predictive materials by design over the next decade without quantifying uncertainty

Predictive Computational Science:

Successful models are verified (codes) and validated (experiments) **(V&V)**. The uncertainty in computer predictions (the QoI’s) must be quantified if the predictions are used in important decisions. **(UQ)**



“Uncertainty is an essential and non-negotiable part of a forecast. Quantifying uncertainty carefully and explicitly is essential to scientific progress.” Nate Silver



Confidence interval

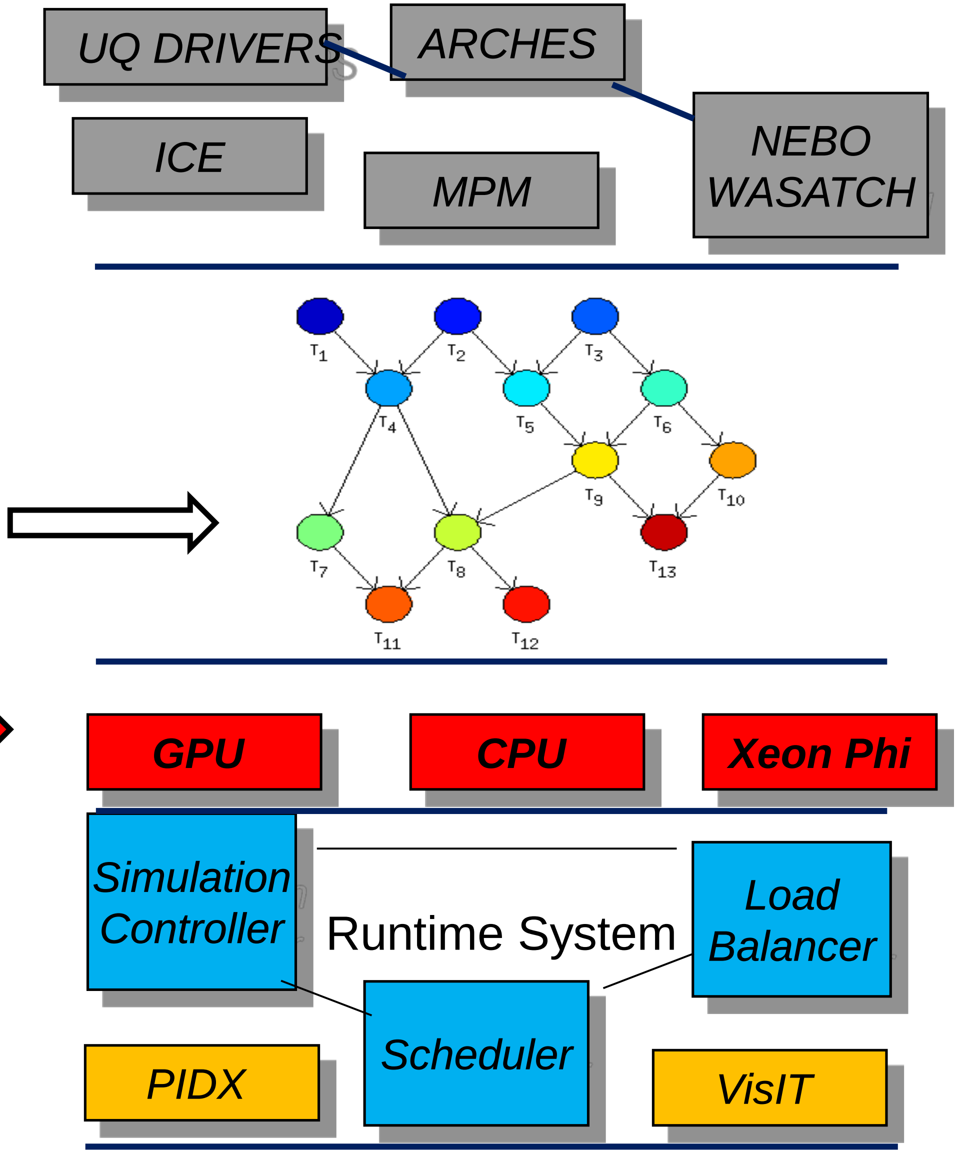
Some components have not changed as we have gone from 600 to 600K cores

- **Application Specification** via ICE MPM ARCHES or NEBO/WASATCH DSL

- **Abstract task-graph** program that

- Is compiled for

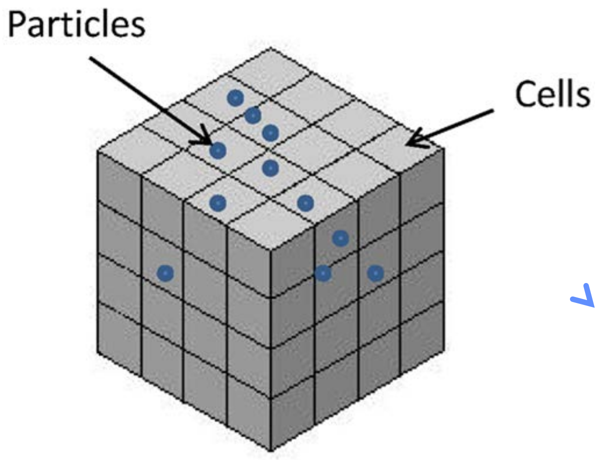
- Executes on: **Runtime System** with: asynchronous out-of-order execution, work stealing, Overlap communication & computation.Tasks running on cores and accelerators
- **Scalable I/O via Visus PIDX**



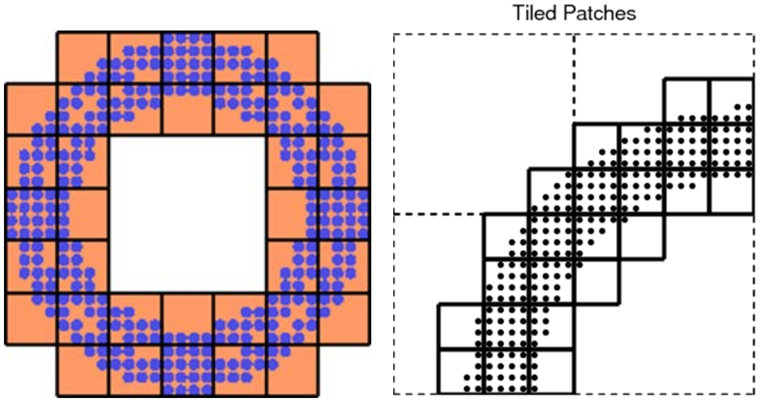
Uintah(X) Architecture Decomposition

Uintah Patch, Variables and AMR Outline

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



Uintah Patch



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

- **Structured Grid + Unstructured Points**
- **Patch-based Domain Decomposition**
- **Regular Local Adaptive Mesh Refinement**
- **Dynamic Load Balancing**
  - **Profiling + Forecasting Model**
  - **Parallel Space Filling Curves**
- **Works on MPI and/or thread level**

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

Uintah Directed Acyclic (Task) Graph-Based Computational Framework

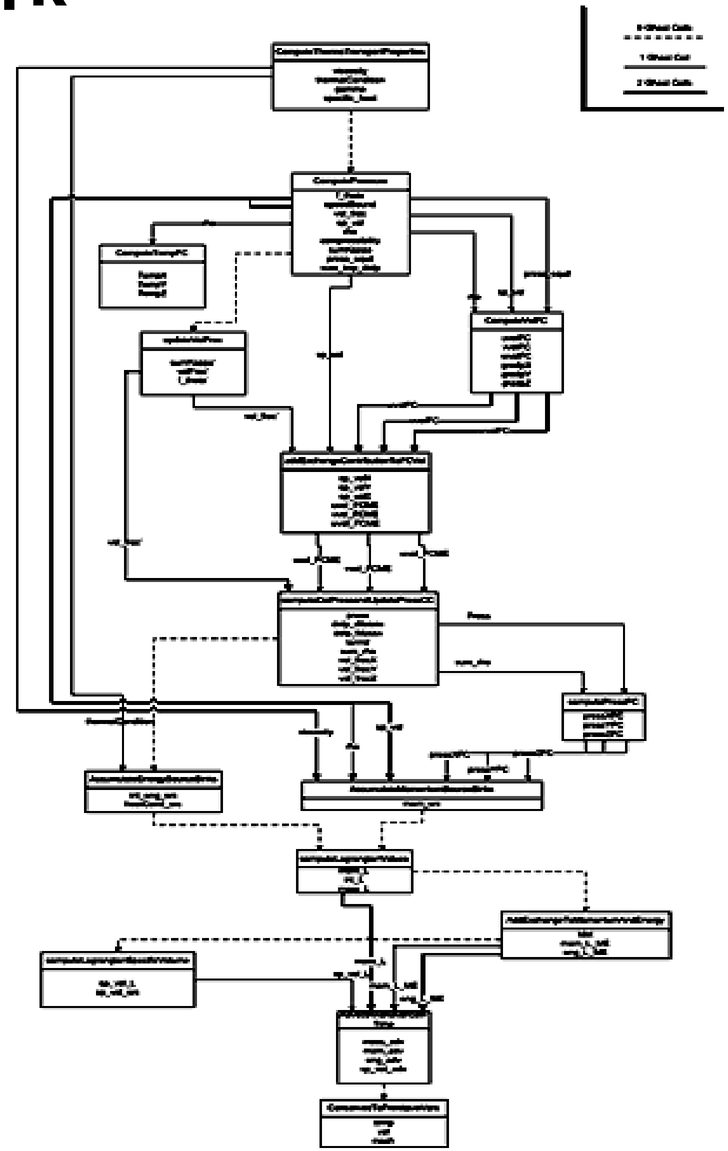
Each task defines its computation with required inputs and outputs

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

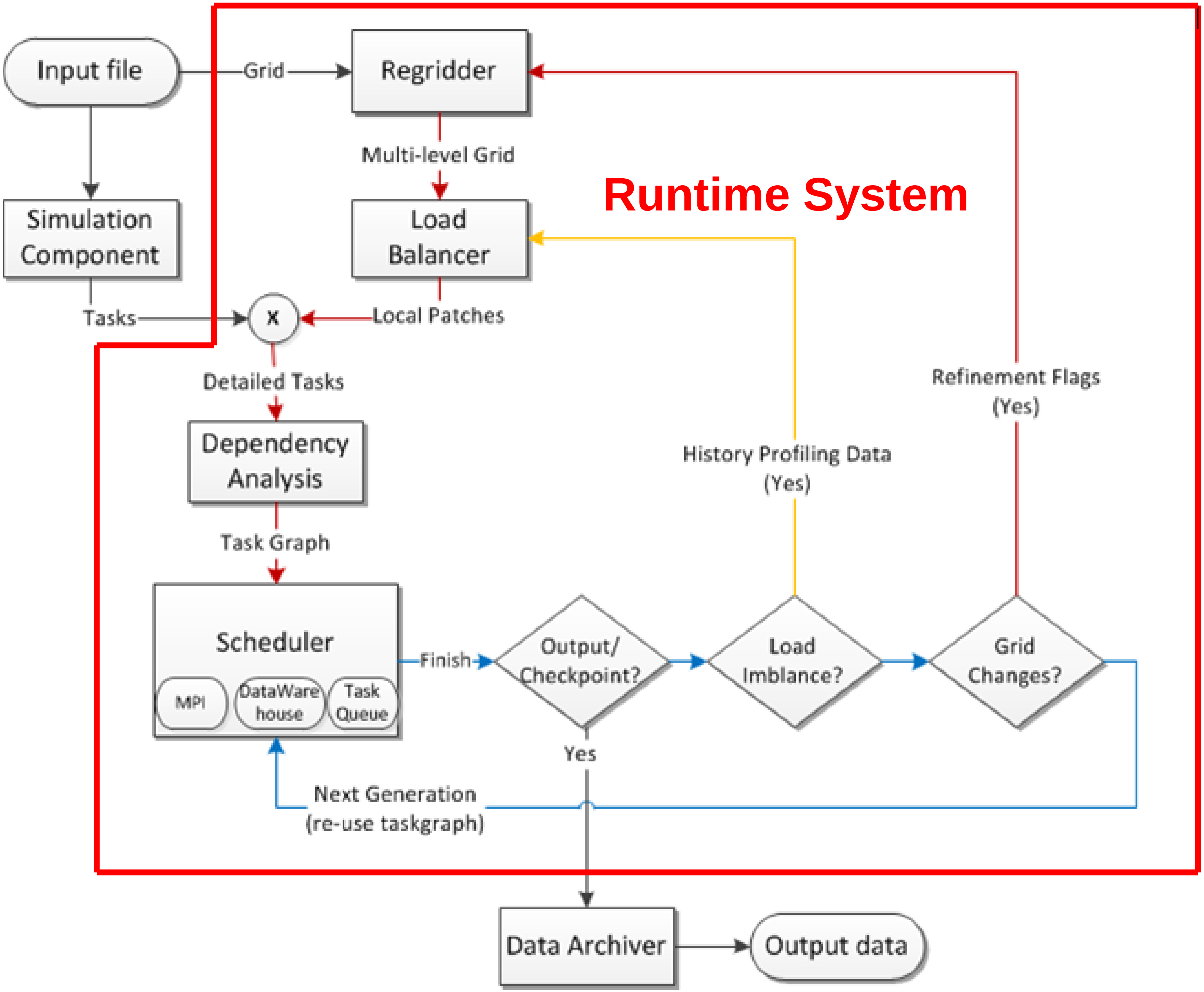
Tasks do not explicitly define communications but only what inputs they need from a data warehouse and which tasks need to execute before each other.

Communication is overlapped with computation

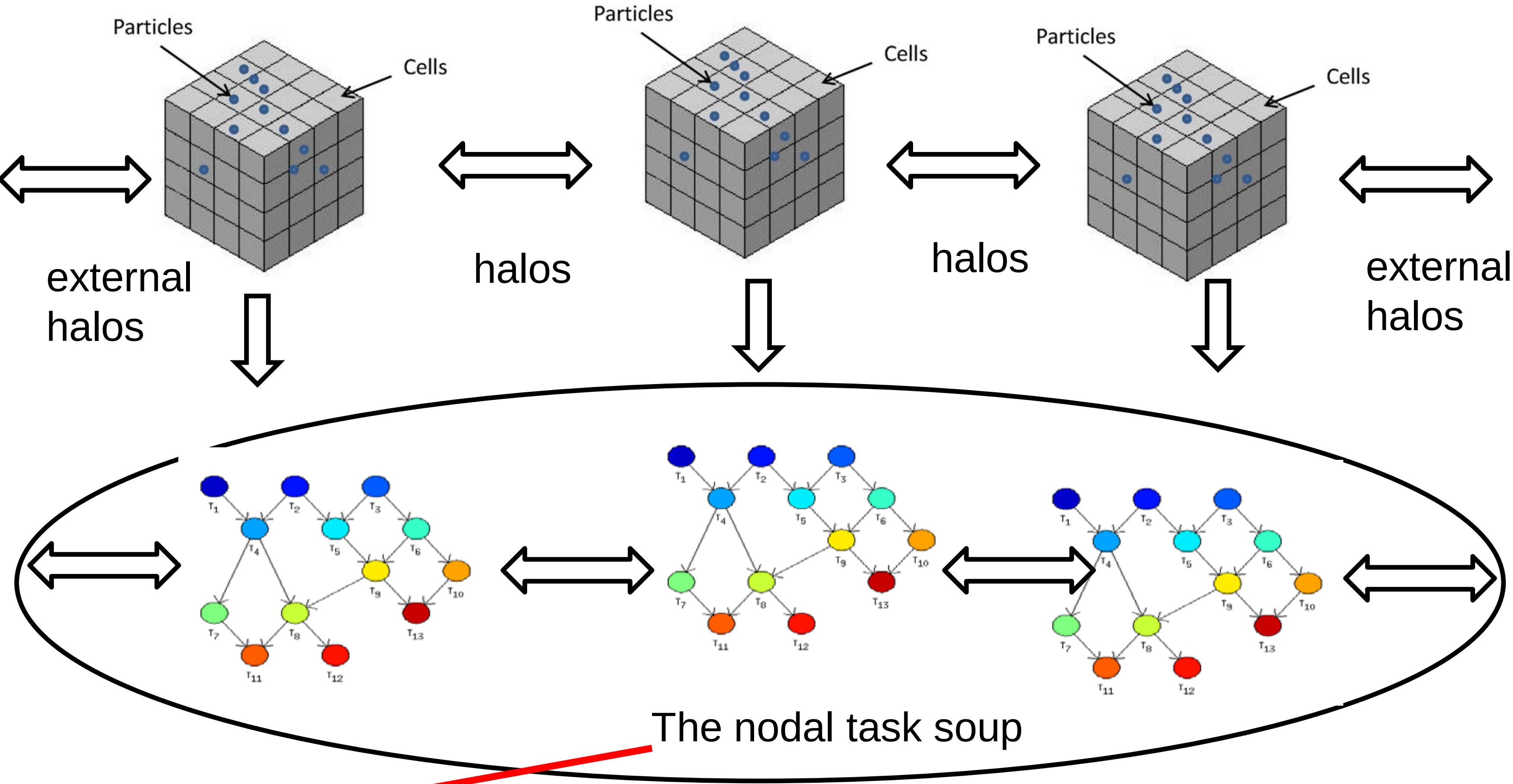
Taskgraph is executed adaptively and sometimes out of order, inputs to tasks are saved



Tasks get data from OLD Data Warehouse and put results into NEW Data Warehouse


Runtime System
Input file
Grid
Regridder
Multi-level Grid
Load Balancer
Simulation Component
Tasks
X
Local Patches
Detailed Tasks
Dependency Analysis
Task Graph
Scheduler
MPI
DataWarehouse
Task Queue
Finish
Output/ Checkpoint?
Load Imblance?
Grid Changes?
History Profiling Data (Yes)
Refinement Flags (Yes)
Yes
Next Generation (re-use taskgraph)
Data Archiver
Output data

Task Graph Structure on a Multicore Node with multiple patches



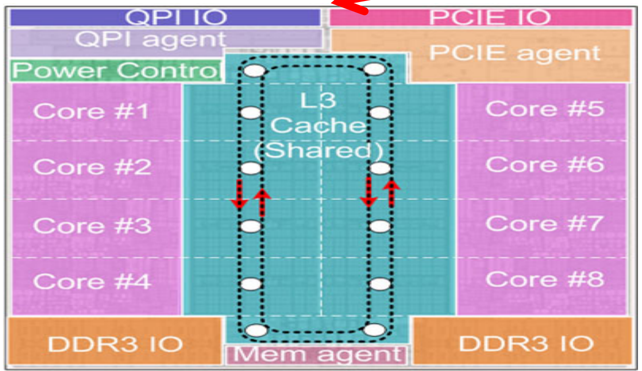
external halos

halos

halos

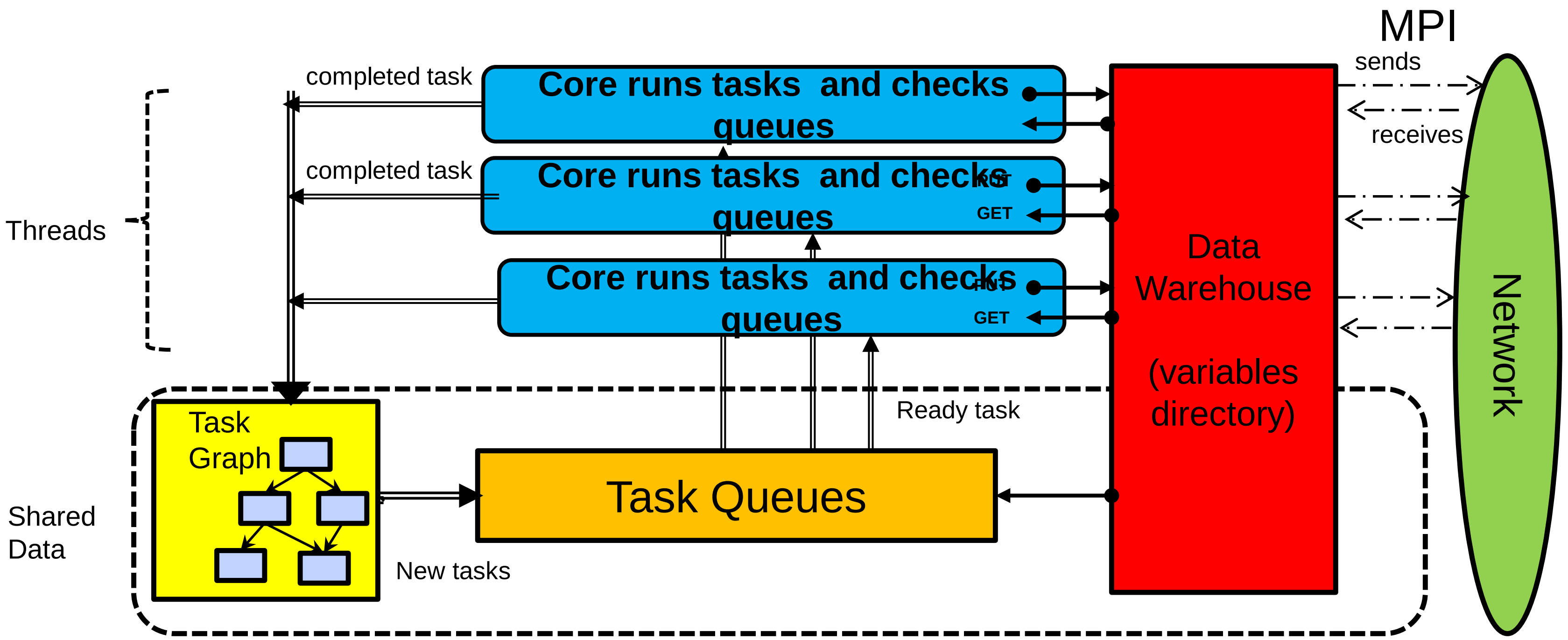
external halos

The nodal task soup



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)

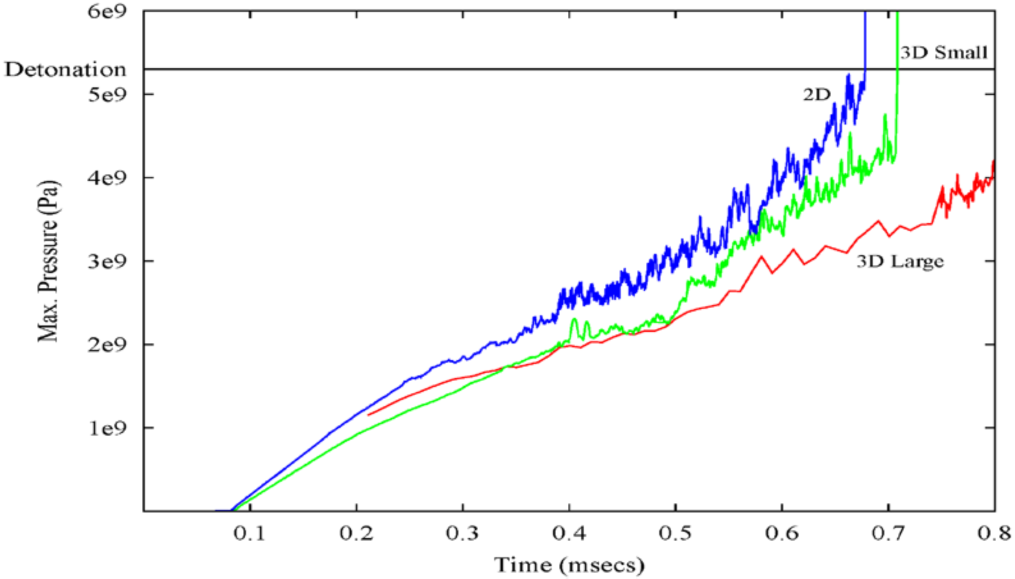


- One MPI Process per Multicore node
- All threads directly pull tasks from task queues execute tasks and process MPI sends/receives
- Tasks for one patch may run on different cores
- One data warehouse and task queue per multicore node
- Lock-free data warehouse enables all cores to access memory quickly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

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

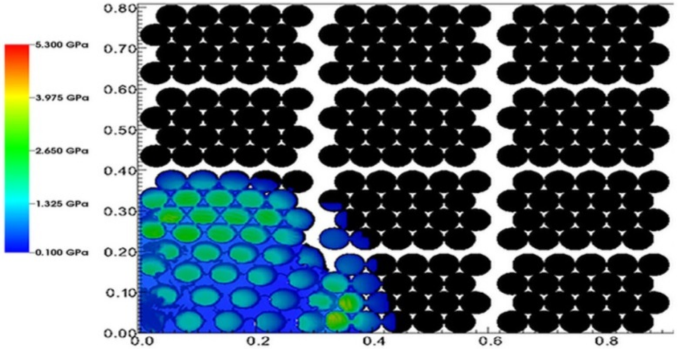
Speeding truck with 8000 explosive boosters each with 2.5-5.5 lbs of explosive overturned and caught fire

Experimental evidence for a transition from deflagration to detonation?

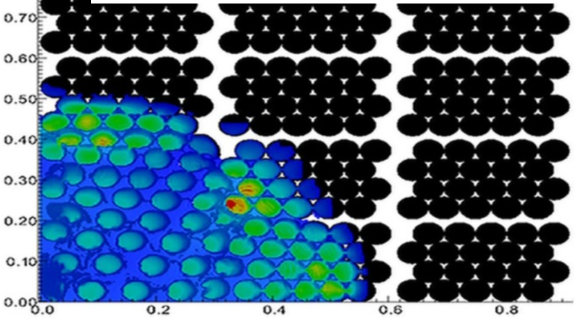


Deflagration wave moves at ~400m/s not all explosive consumed. Detonation wave moves 8500m/s all explosive consumed.

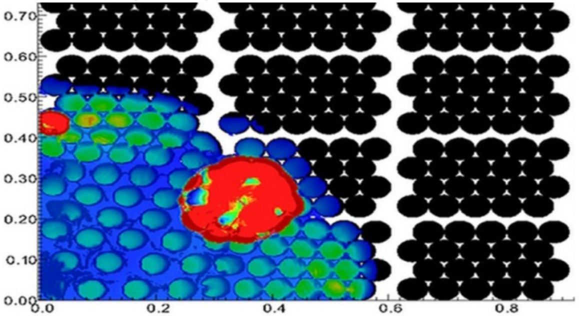
2013 Incite 200m cpu hrs



0.534 msec



0.634 msec

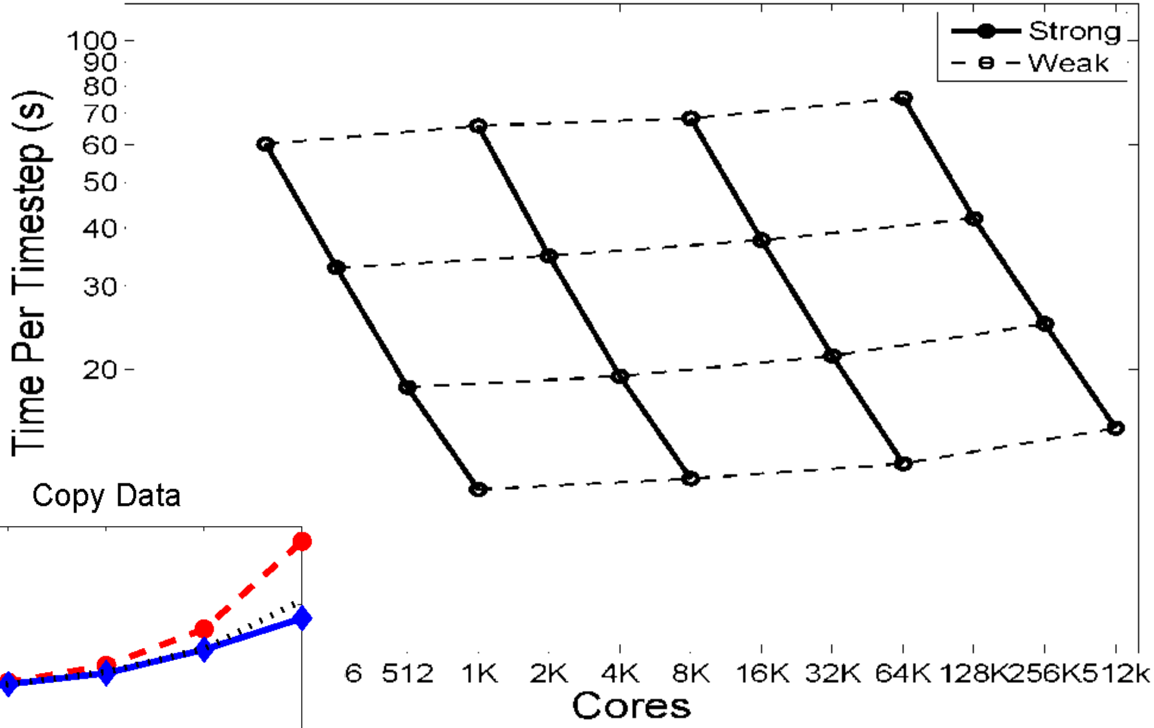


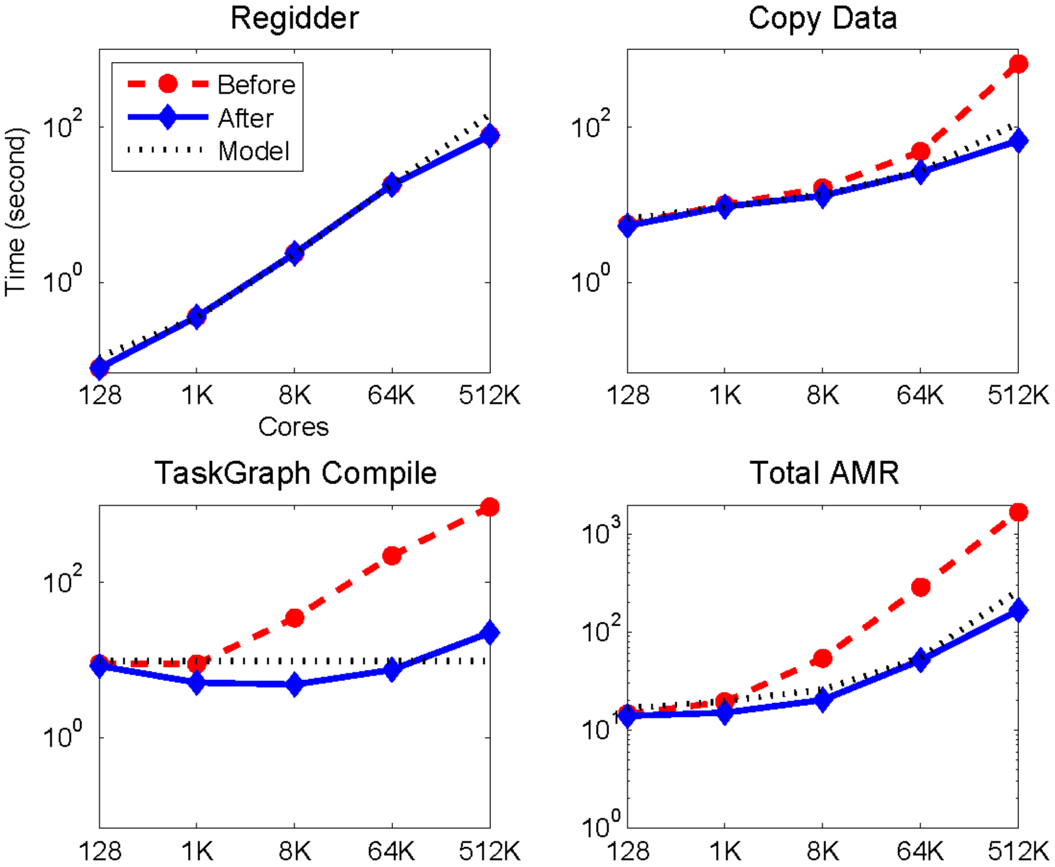
0.644 msec

Spanish Fork Accident

500K mesh patches
1.3 Billion mesh cells
7.8 Billion particles

Detonation MPMICE: Scaling on Mira BGQ

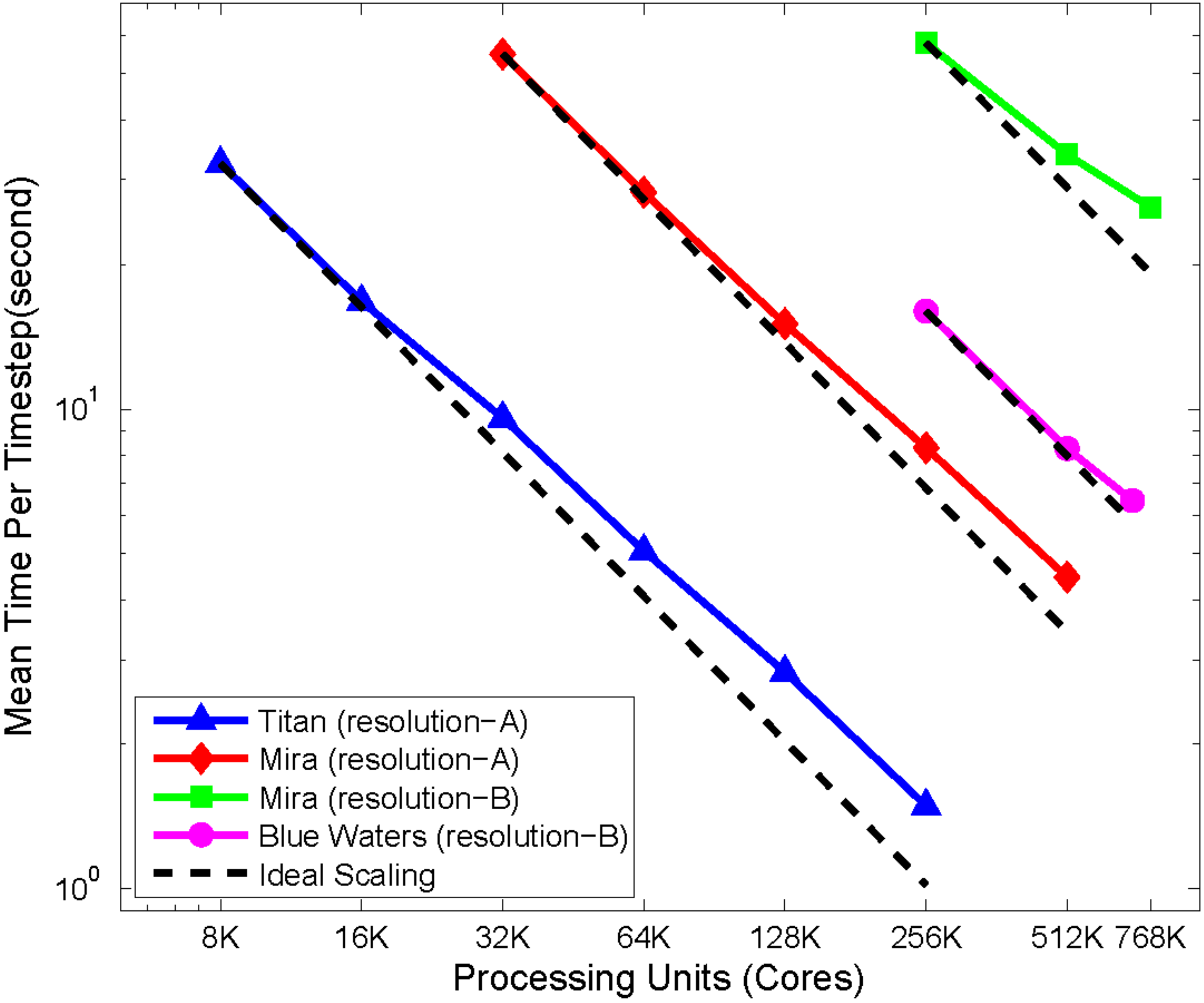




At every stage when we move
to the next generation of problems
Some of the algorithms and data
structures need to be replaced .

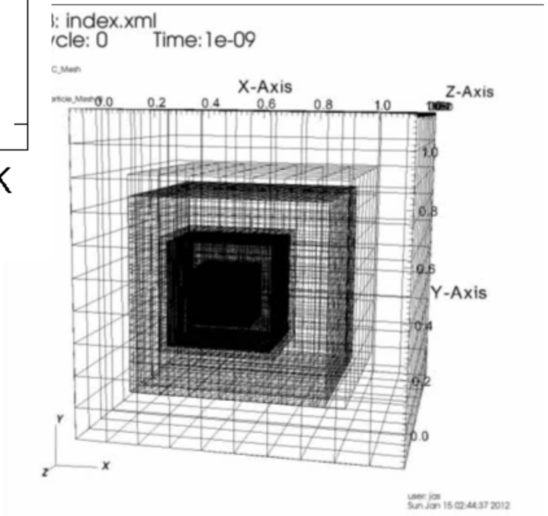
Scalability at one level is no certain
Indicator fro problems or machines
An order of magnitude larger

MPM AMR ICE Strong Scaling



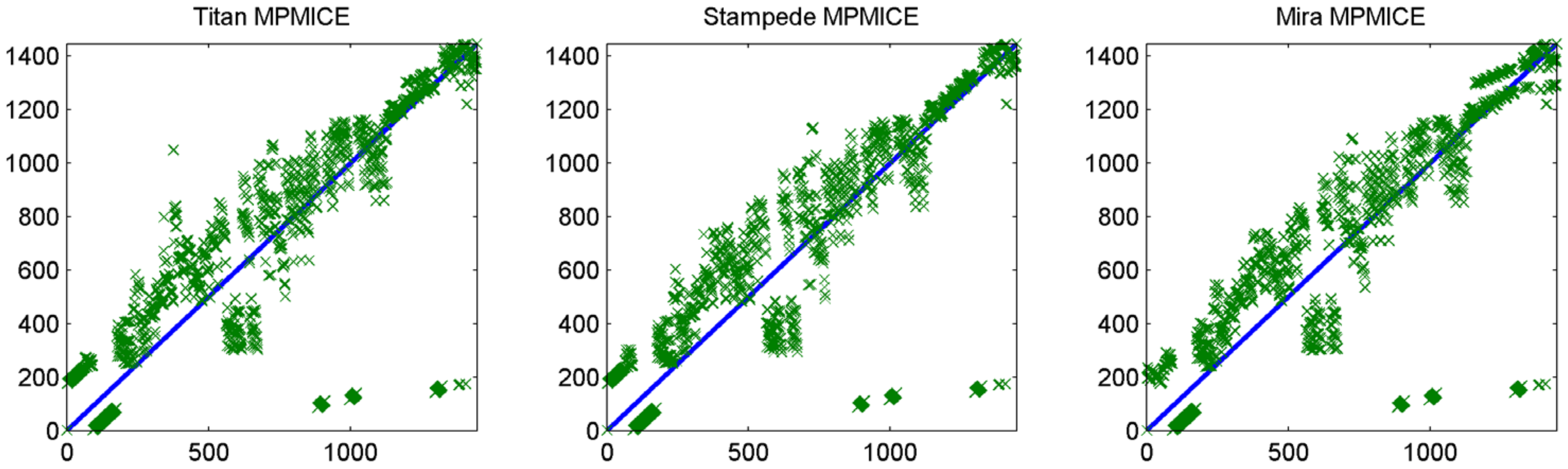
Mira DOE BG/Q 768K cores
Blue Waters Cray XE6/XK7 700K+ cores

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.

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

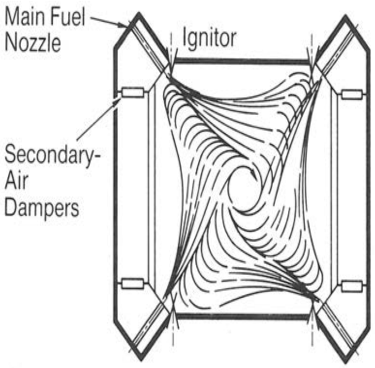
(ii) Use optimal size patches to balance overhead and granularity 16x16x 16 to 30x30x30.

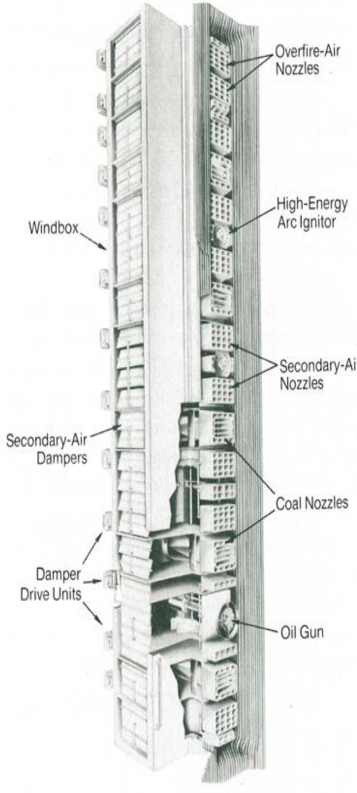
(iii) Use only one data warehouse but allow all cores fast access to it, through the use of atomic operations.

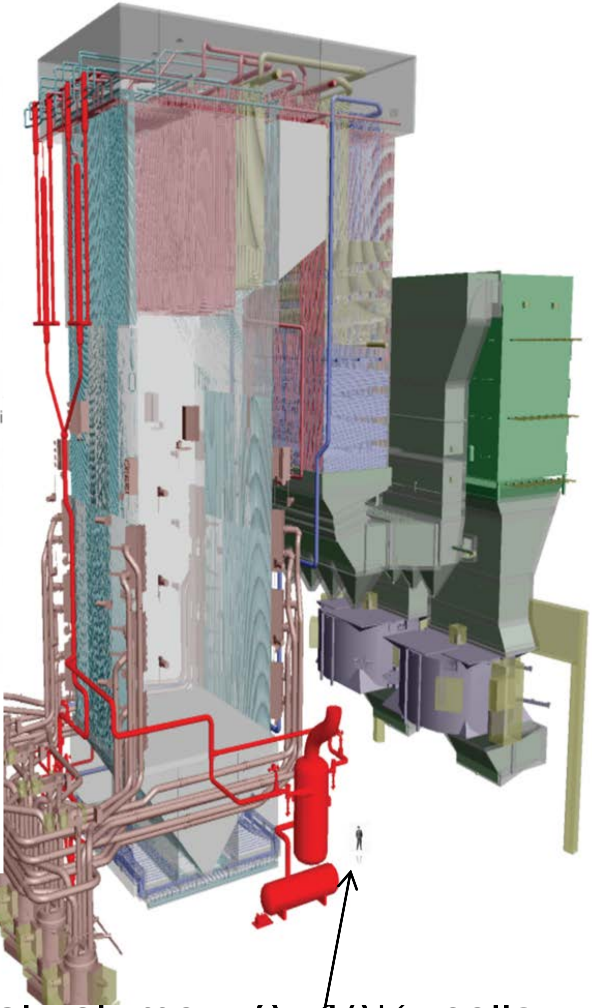
(iv) Prioritize tasks with the most external communications

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

An Exascale Design Problem - Alstom Clean Coal Boilers







For 350MWe boiler problem. LES resolu
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.

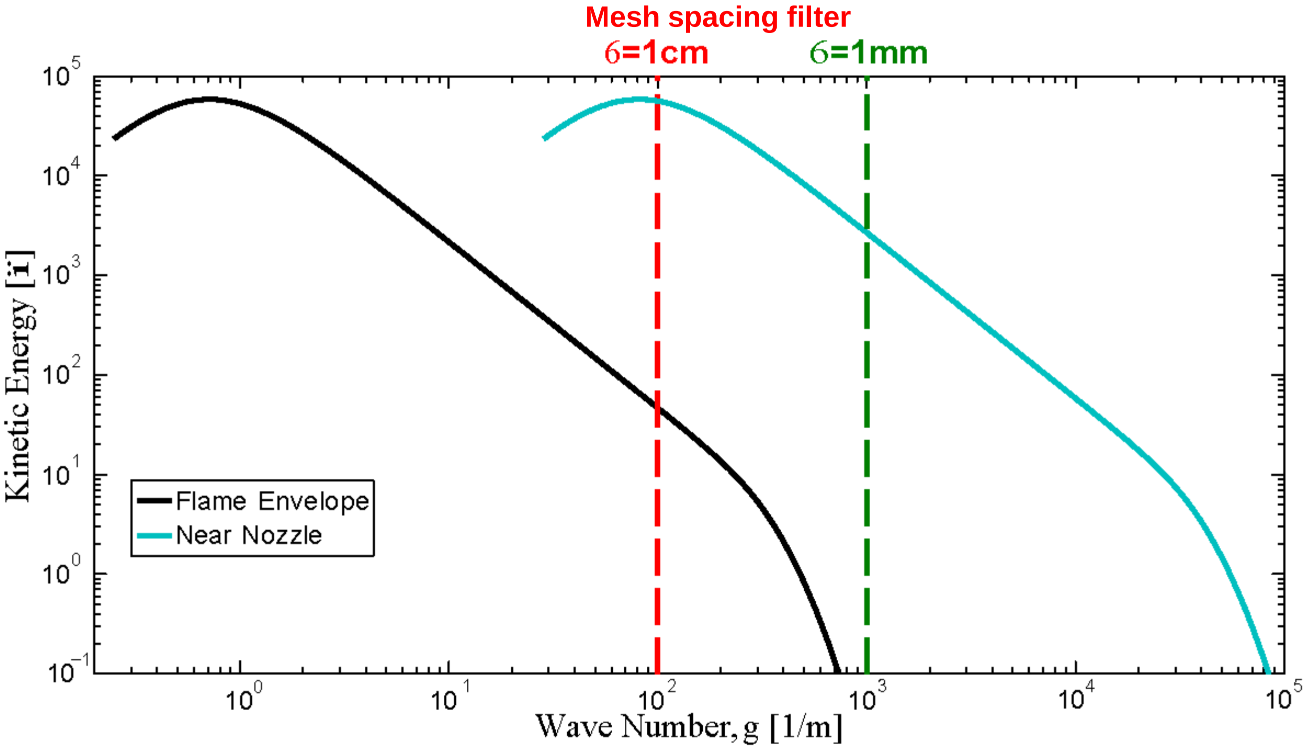
Prof. Phil Smith Dr Jeremy Thornock ICSE

Existing Simulations of Boilers using ARCHES in Uintah

(i) Traditional Lagrangian/RANS approaches do not address well particle effects

(ii) LES has potential to predict oxy---coal flames and to be an important design tool

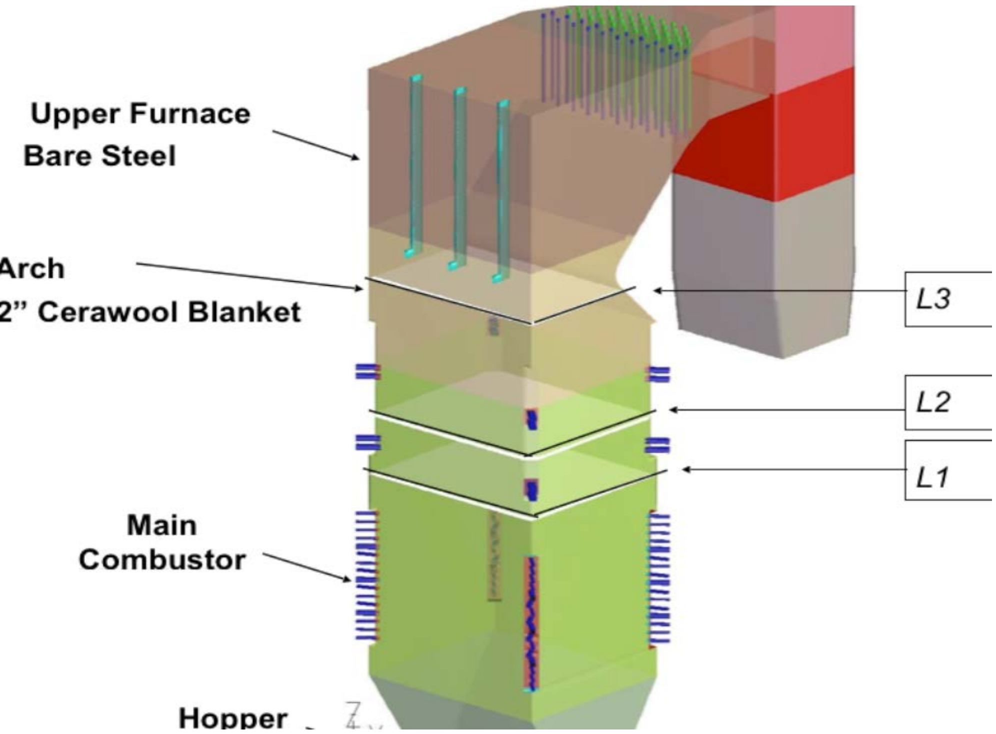
(iii) LES is “like DNS” for coal, but 1mm mesh needed to capture phenomena

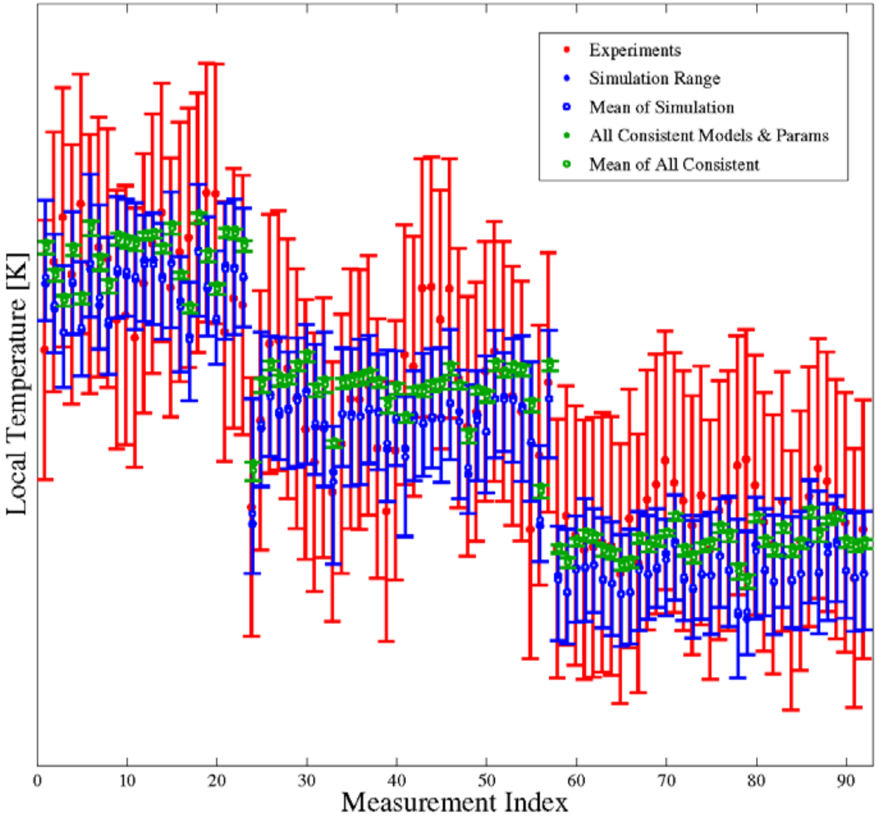


Structured, finite-volume method, Mass, momentum, energy with radiation

Higher-order temporal/spatial numerics, LES closure, Tabulated chemistry

Uncertainty Quantified Runs on a Small Prototype Boiler

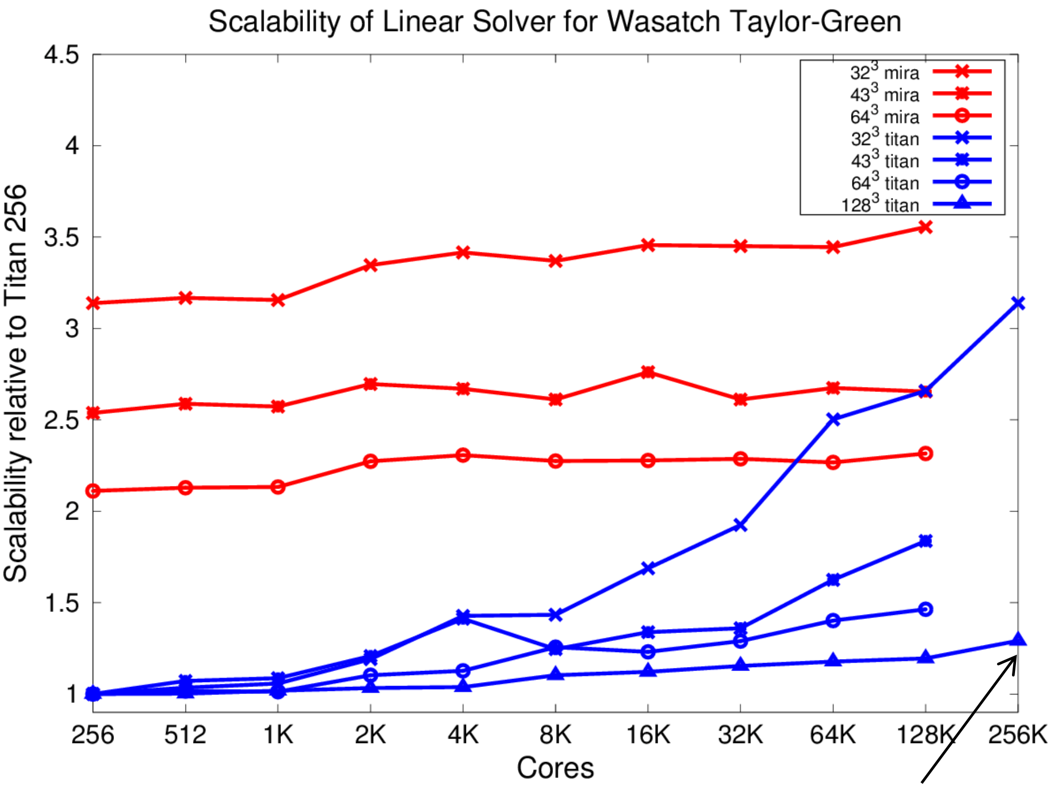




Red is experiment
Blue is simulation
Green is consistent

Absence of scales for commercial reasons

Linear Solves arises from Low Mach Number Navier –Stokes Equations

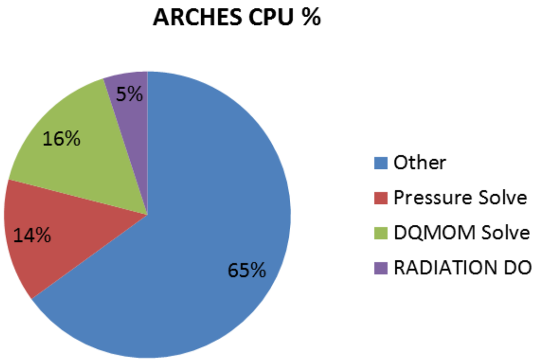


$$\nabla^2 p = R, \quad \text{where } R = \nabla . F + \frac{\partial^2 p}{\partial t^2}$$

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



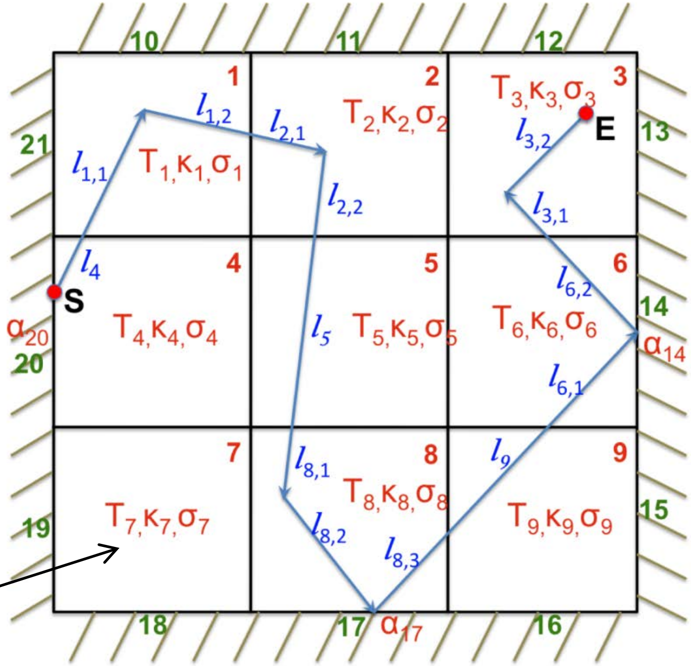
Each **Mira Run** is scaled wrt the **Titan Run at 256 cores**
Note these times are not the same for different patch sizes.

Weak Scalability of Hypre Code

One radiation solve
every 10 timesteps

GPU-RMCRT

- **Incorporate dominant physics**
  - *Emitting / Absorbing Media*
  - *Emitting and Reflective Walls*
  - *Ray Scattering*
- **User controls # rays per cell**
  - *Each cell has Temp Absorb and Scattering Coeffs*
- ***Radiative Heat Transfer key***
  - *Replicate Geometry on every node*
  - *Calculate heat fluxes on Geometry*
  - ***Transfer heat fluxes from all nodes to all nodes***



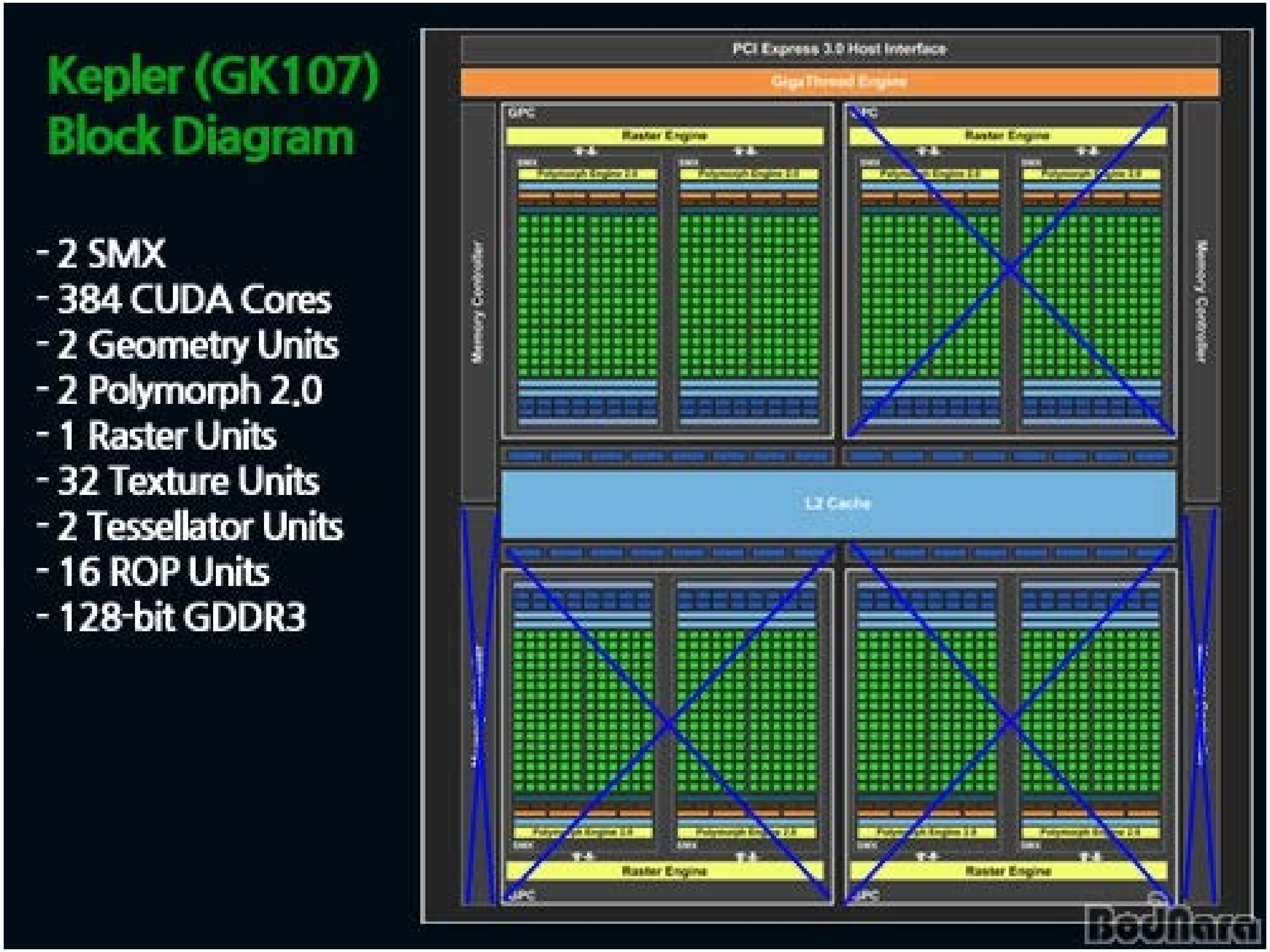
Reverse ray tracing back from Heat flux at walls to origin

More efficient than forward ray tracing

K20 and K40

Internal 200-300 GB/sec

External 8-16 GB/sec (the Dixie straw



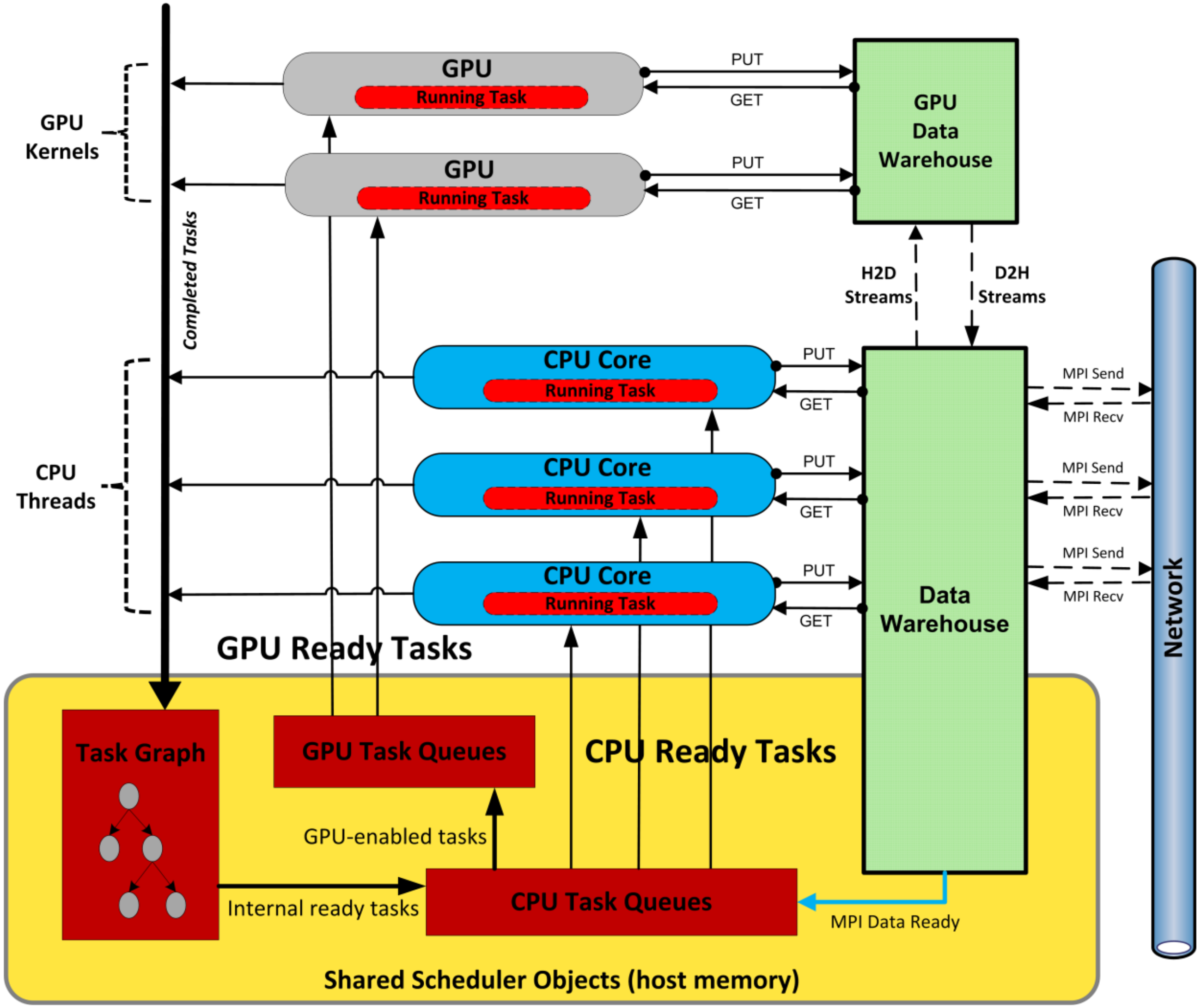
Kepler (GK107) Block Diagram

- 2 SMX
- 384 CUDA Cores
- 2 Geometry Units
- 2 Polymorph 2.0
- 1 Raster Units
- 32 Texture Units
- 2 Tessellator Units
- 16 ROP Units
- 128-bit GDDR3

NVIDIA K20m GPU ~order of magnitude speedup over 16 CPU cores

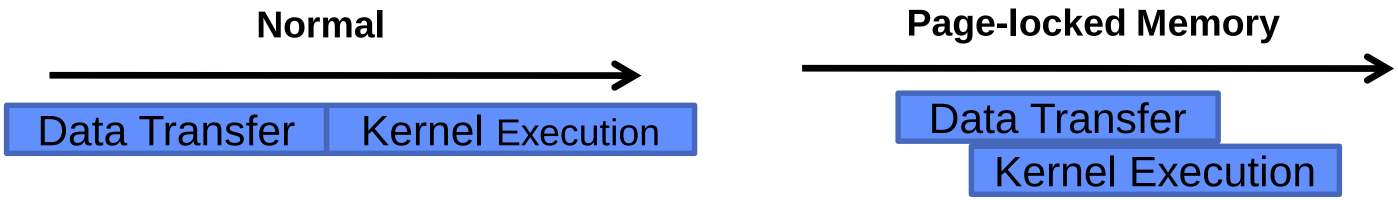
(Intel Xeon E5-2660 @2.20 GHz)

Uintah Heterogeneous Runtime System (GPU and Intel Xeon Phi (MIC)



GPU Task and Data Management

Framework Manages Data Movement
Host ← → Device



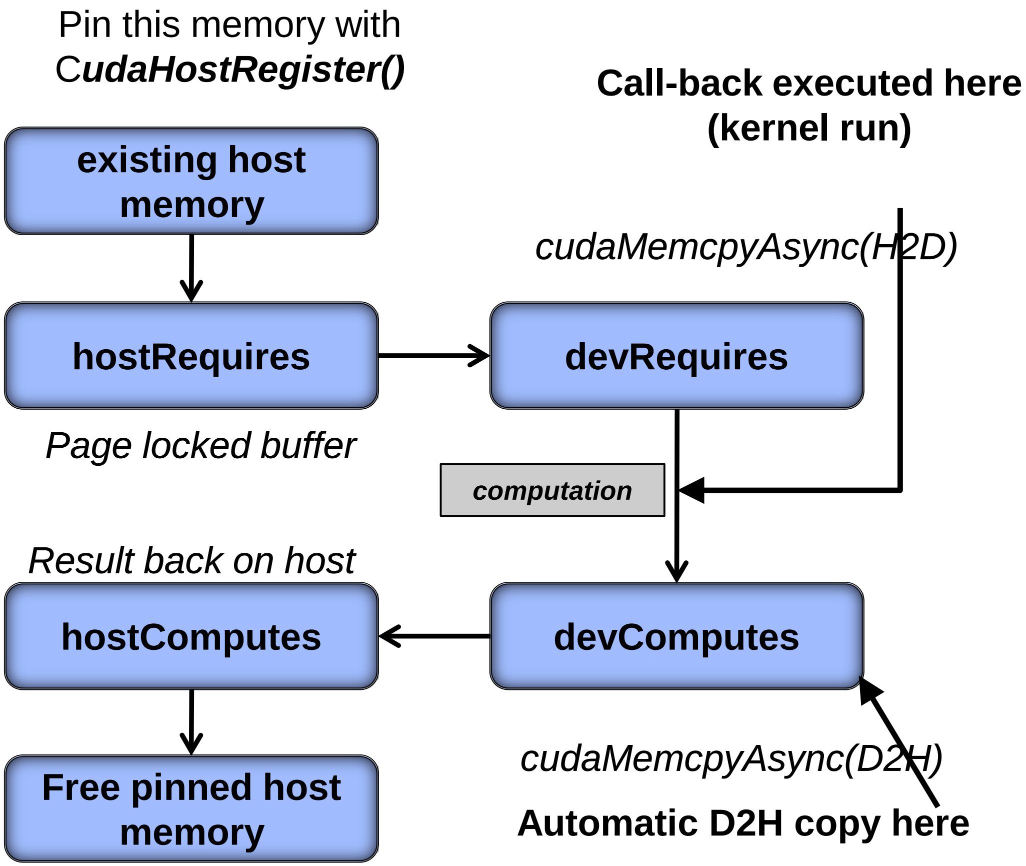
Normal

Data Transfer | Kernel Execution

Page-locked Memory

Data Transfer
Kernel Execution

- Use ***CUDA Asynchronous API***
- **Automatically** generate CUDA streams for task dependencies
- **Concurrently** execute kernels and memory copies
- **Preload** data before task kernel executes
- **Multi-GPU** support



Pin this memory with *CudaHostRegister()*

existing host memory

hostRequires

Page locked buffer

devRequires

cudaMemcpyAsync(H2D)

Call-back executed here (kernel run)

computation

Result back on host

hostComputes

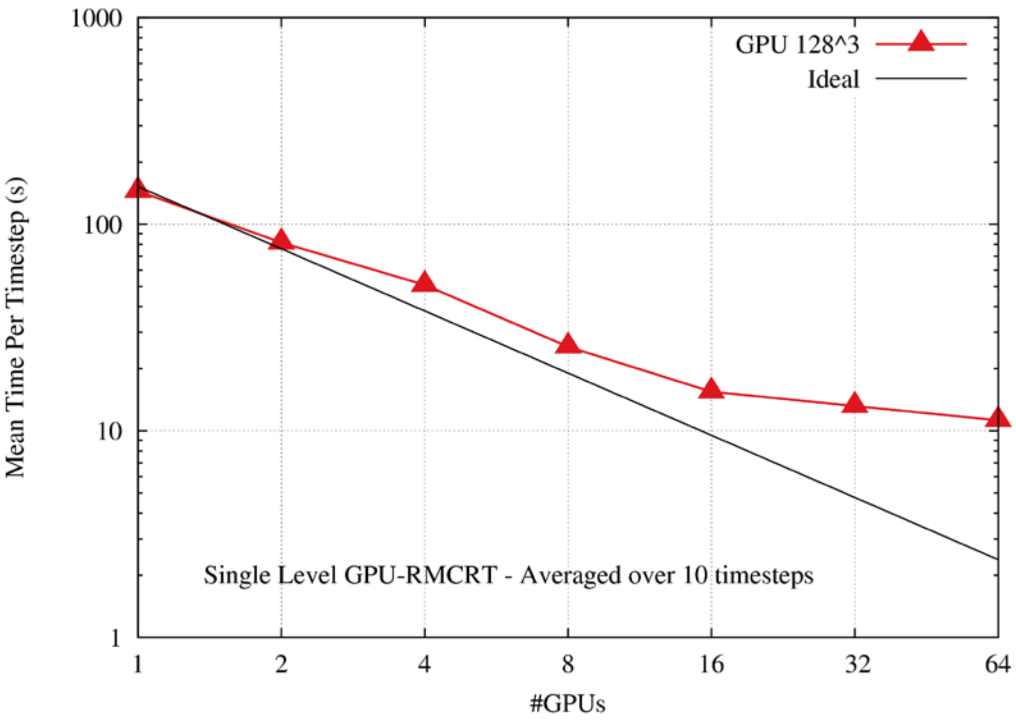
devComputes

Free pinned host memory

cudaMemcpyAsync(D2H)

Automatic D2H copy here

GPU-Based RMCRT Scalability

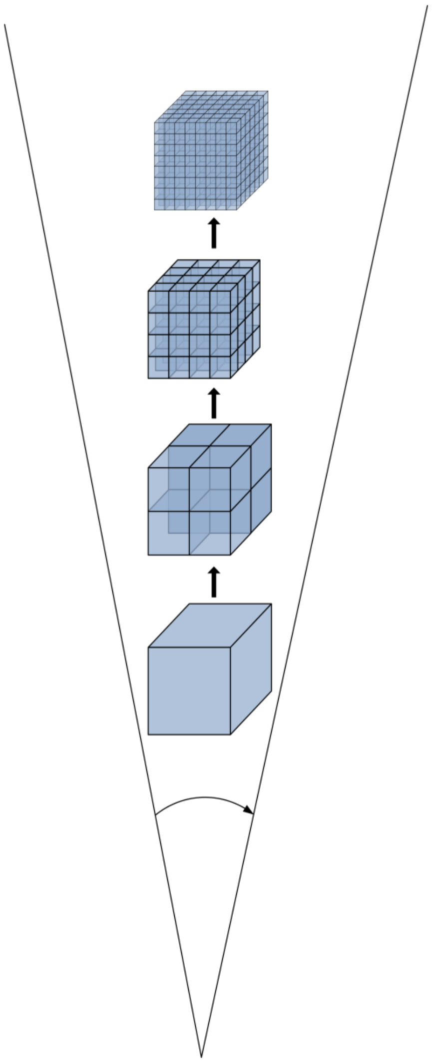


Strong scaling results for production GPU implementations of RMCRT
NVIDIA - K20 GPUs

- Mean time per timestep for GPU lower than CPU (up to 64 GPUs)
- GPU implementation quickly runs out of work
- **All-to-all** nature of problem limits size that can be computed due to memory and comm constraints with large, highly resolved physical domains

Adaptive RMCRT Approach

Use coarse patches
Further away



If we have N nodes all-to all complexity N log(N). Data sent is N log(N) FFpN (Fflux functions_per_Node)

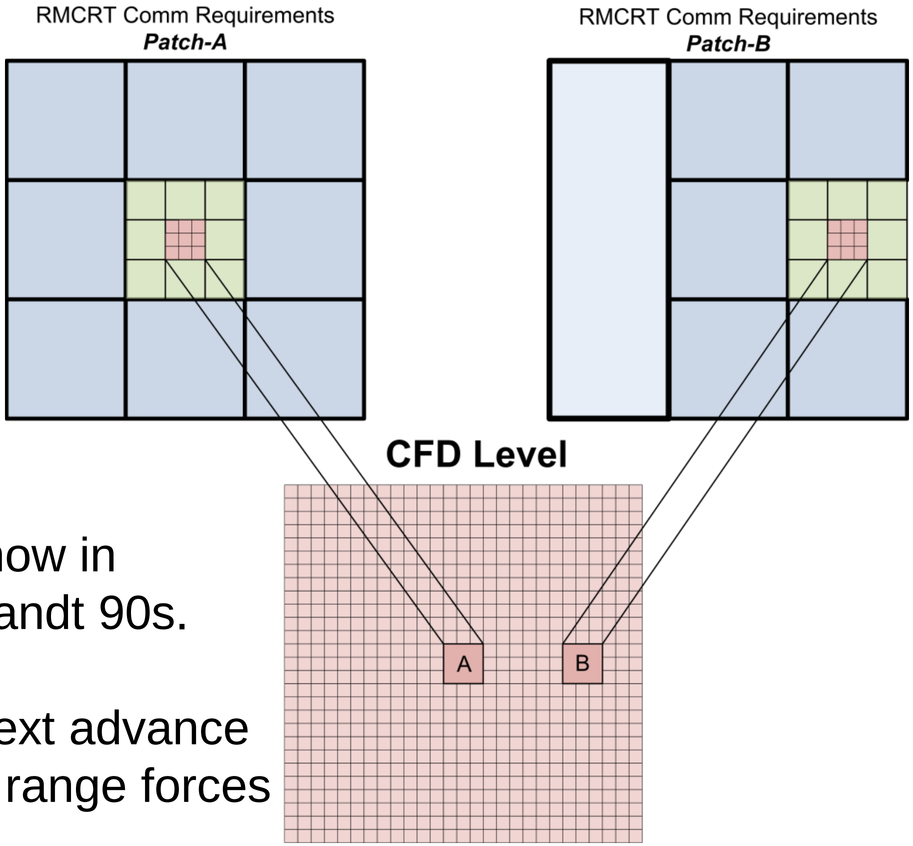
MPI buffers swamped on current machines

USE AMR to reduce data sent

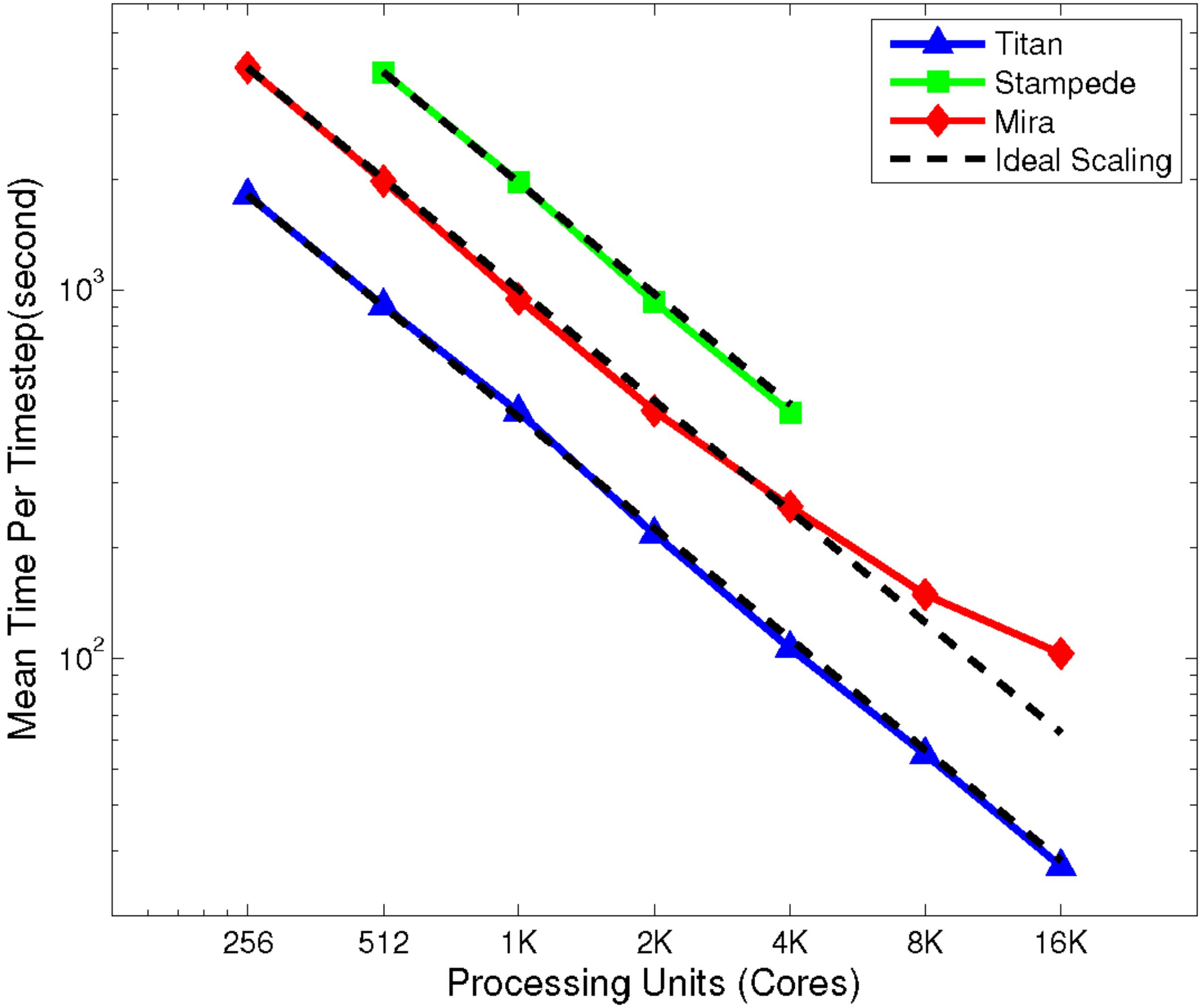
This is a well understood math paradigm.
Used in lubrication, now in MD going back to Brandt 90s.

Seen in MD as the next advance
In scalability for long range forces

4-Level Data Onion



Multi-Level RMCRT CPU Scalability

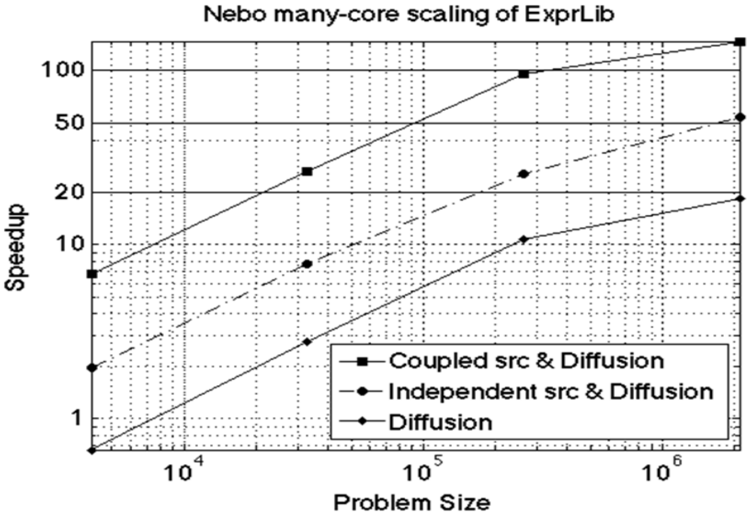


CPU Prototype in ARCHES

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: u**se Kokkos for rewriting legacy applications +Wasach/Nebo DSL for new code. MIC and GPU ongoing.
- **Linear Solvers Hypre and AMGX**

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



Kokkos: A Layered Collection of Libraries
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**