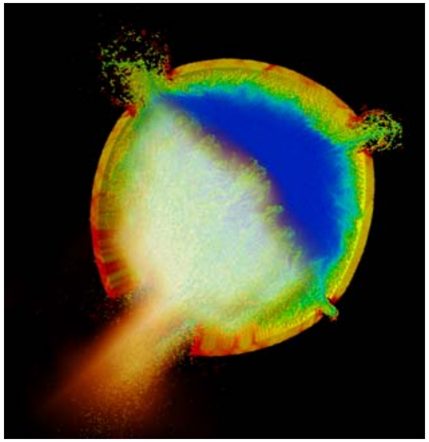
Software for Adaptive Meshing on Large Scale Fluid-Structure Interaction Problems



Martin Berzins

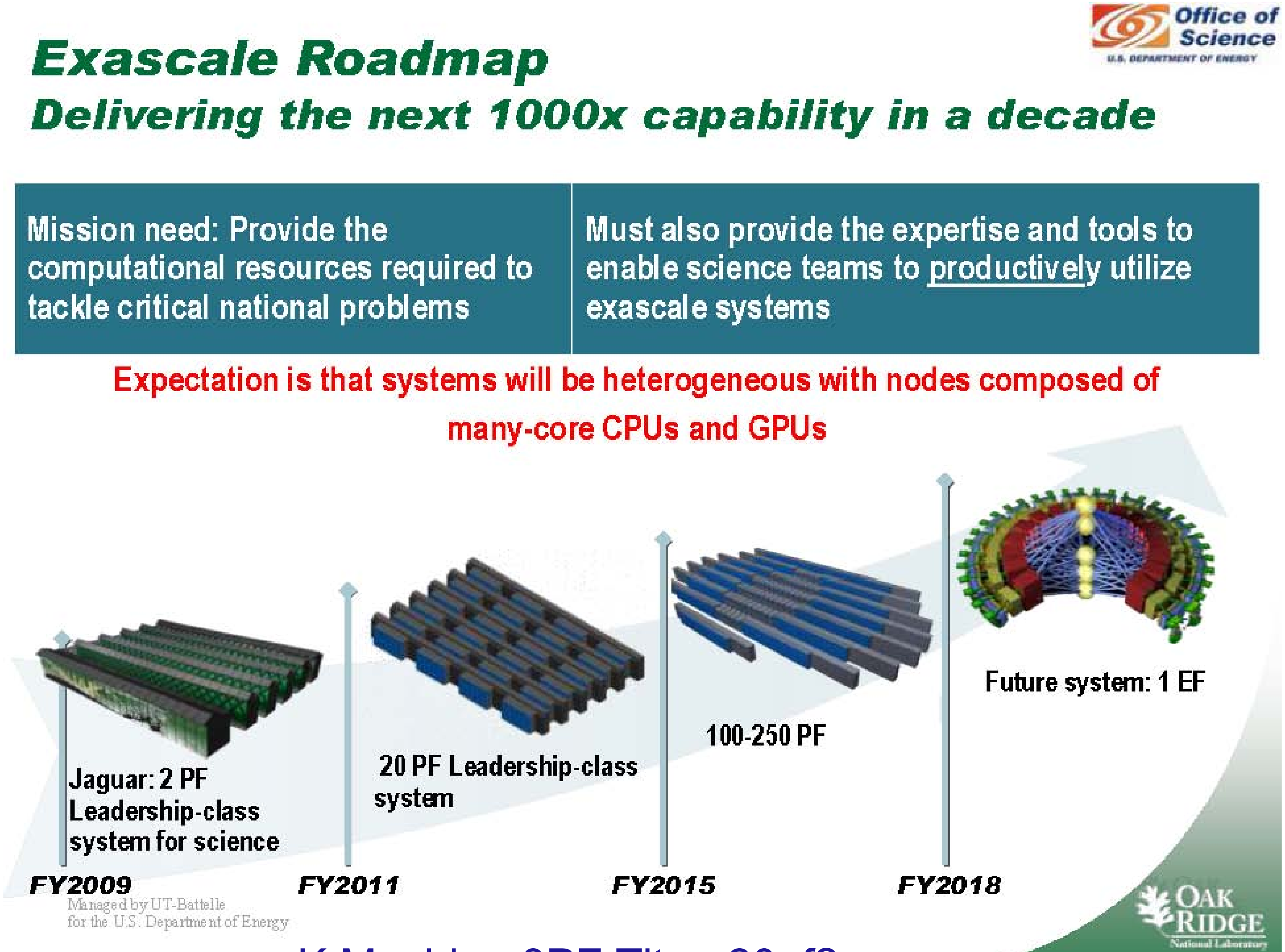
1. **Driving Problems and Technology**
2. **Uintah Software and Modeling Details**
3. **Scalable Adaptive Meshing Algorithms**
4. **Detonation Results and Future Work**

Thanks to:
Computer Science **Justin Luitjens, Qingyu Meng, John Schmidt**
Chemistry + Mech Eng. Todd Harman, Joseph Peterson, Chuck Wight
Steve Parker
DoE for funding the CSAFE project from 1997-2010, DOE NETL, INCITE
NSF for funding via SDCI and PetaApps (Abani Patra)
TACC NICS TRAC ORNL

http://www.uintah.utah.edu

THE UNIVERSITY OF UTAH

The path to exascale?



K Machine 8PF Titan 20pf?

Source Al. Geist

DARPA Exascale Hardware Study

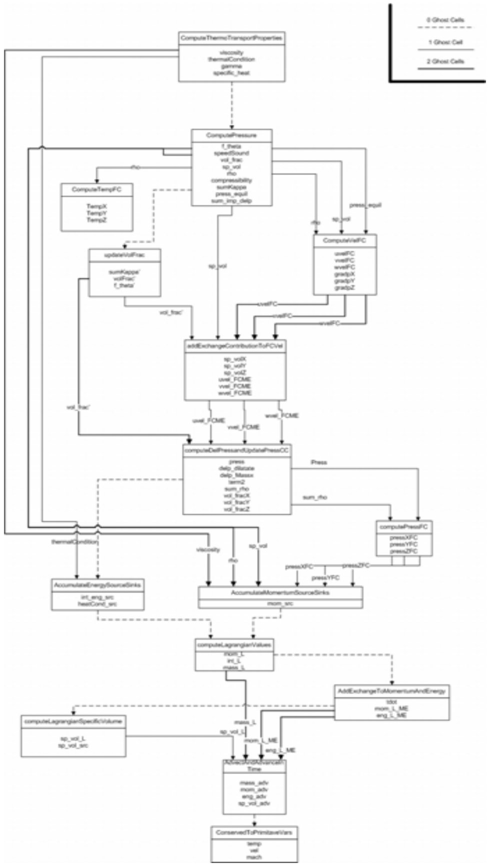
- DARPA public report (Peter Kogge et al.)
- Describes Challenges in going to Exascale at national level and petascale at University level.
- Exascale machine Aggressive Strawman:
  - 742 cores per socket, 12 sockets per node, 32 nodes per rack
  - 166,113,024 cores, 223,872 sockets
  - 4 flops per cycle per core @1.5Ghz, 1.029 PFlops
  - Power 67MW! DoE aims for 25MW
- Extraordinary concurrency is the only game in town
- Power, fault tolerance, programmability are key

IMPLICATION IS PETASCALE AT LOCAL LEVEL – terascale laptops!

DARPA Exascale Software Study

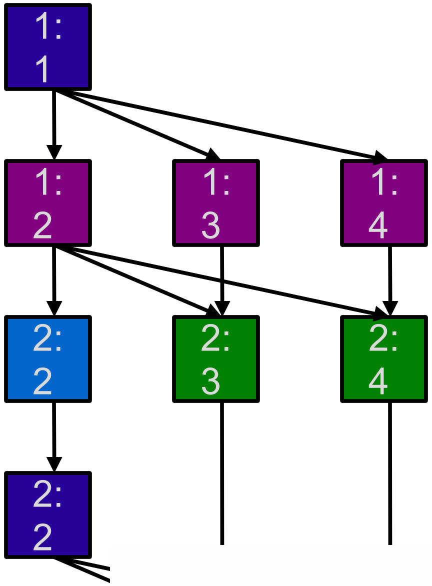
- DARPA public report by (Vivek Sarkar et al.)
- (Sterling) Silver model for exascale software:
  - Have abstraction for high degree of concurrency for directed dynamic graph structured calculations.
  - Enable latency hiding by overlapping computation and communications
  - Minimize synchronization and other overheads
  - Support adaptive resource scheduling
  - Unified approach to heterogeneous processing
- Silver model is a graph-based asynchronous-task work queue model.
- Some instances of this type of approach in use now. CnC, Charm++, Plasma, StarSS, Uintah Very disruptive technology - forces us to rethink programming model
- DOES IT WORK?

Graph Based Languages/frameworks

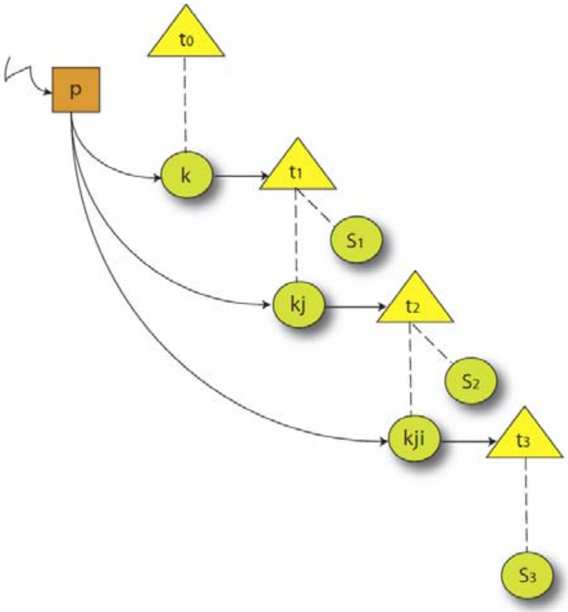


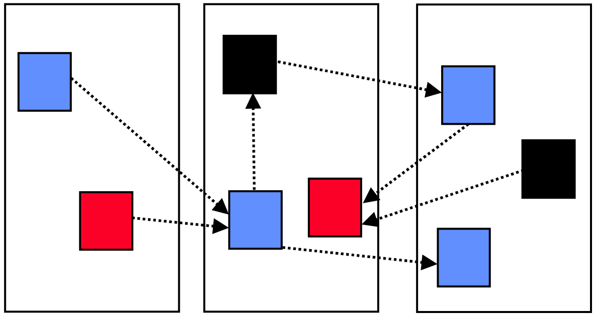
Uintah Taskgraph based PDE Solver

Plasma (Dongarra): DAG based Parallel linear algebra software



Intel CnC: new language for graph based parallelism





Charm++: Object-based Virtualization

Weak and Strong Scalability

Strong scalability $$T(n,p) = \frac{T(n,1)}{p}$$

Weak Scalability $$T(np,p) = T(n,1)$$

Constant time for larger problem on more cores

Both weak and strong scalability only if $T(n,1) = \alpha n$

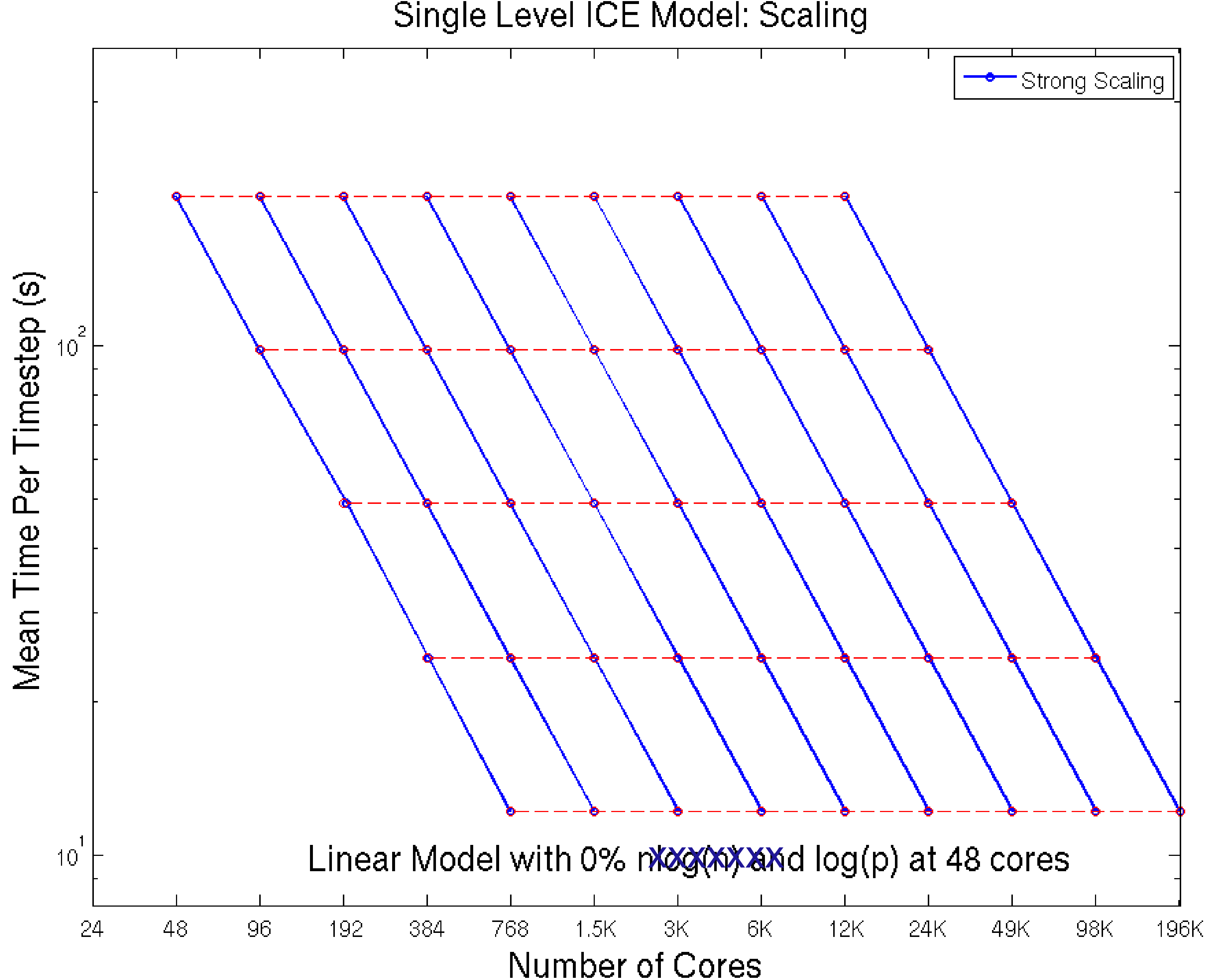
E.G. If $T(n,1) = \alpha n^k$ and $k = 2$

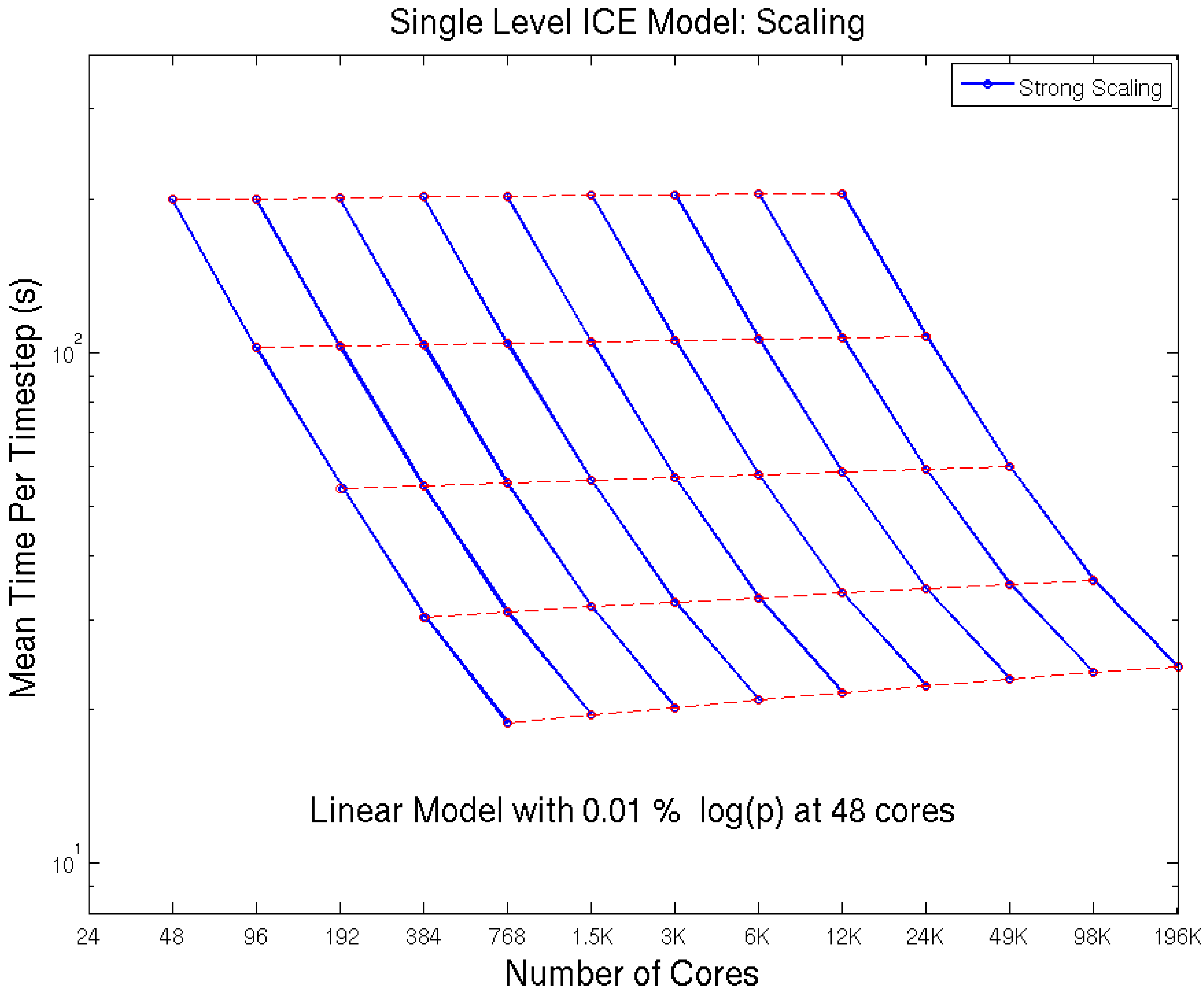
Then doubling the problem size gives four times as much work and hence twice runtime on 2x cores

More realistic model

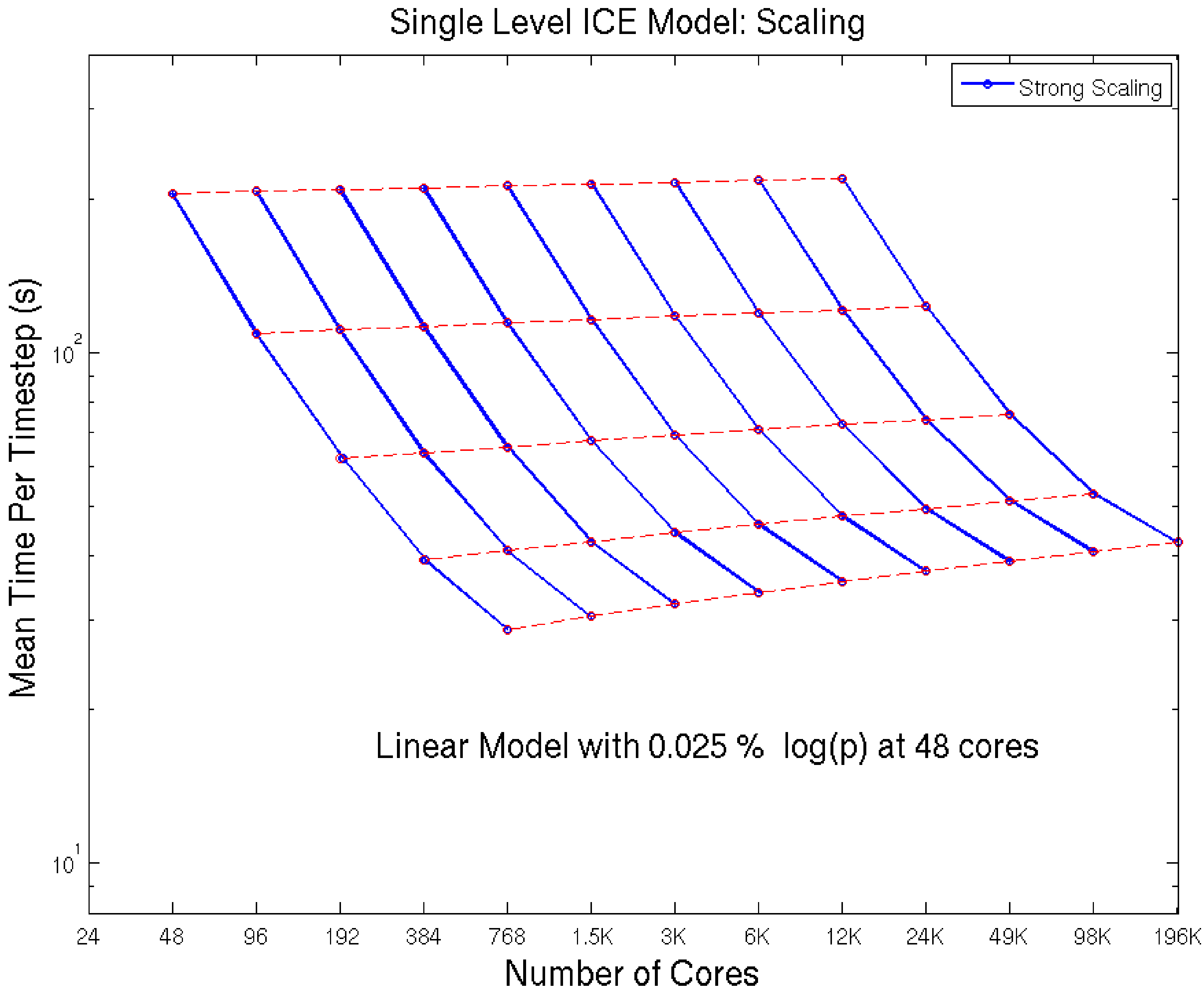
$$T(n,p) = \alpha n + \gamma \log(p)$$

$\log(p_0)\gamma / (\alpha n_0)$ Is fraction of time spent in global collectives at $n_0 p_0$

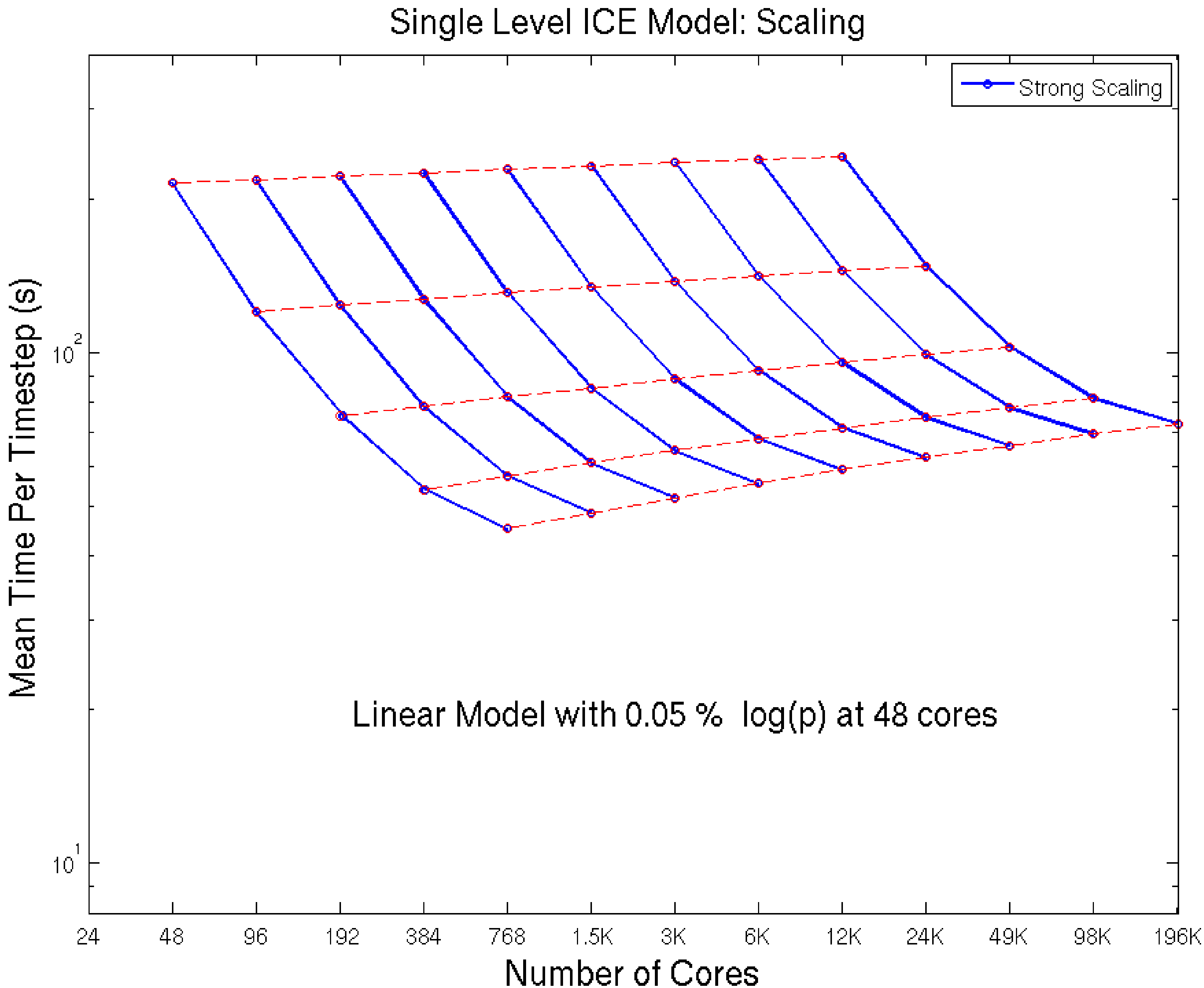

Single Level ICE Model: Scaling
Strong Scaling
Mean Time Per Timestep (s)
10^2
10^1
24
48
96
192
384
768
1.5K
3K
6K
12K
24K
49K
98K
196K
Number of Cores
Linear Model with 0% nlog(n) and log(p) at 48 cores



Single Level ICE Model: Scaling
Strong Scaling
Mean Time Per Timestep (s)
10^2
10^1
Linear Model with 0.01 % log(p) at 48 cores
24
48
96
192
384
768
1.5K
3K
6K
12K
24K
49K
98K
196K
Number of Cores



Single Level ICE Model: Scaling
Strong Scaling
Mean Time Per Timestep (s)
10^2
10^1
Linear Model with 0.025 % log(p) at 48 cores
24
48
96
192
384
768
1.5K
3K
6K
12K
24K
49K
98K
196K
Number of Cores

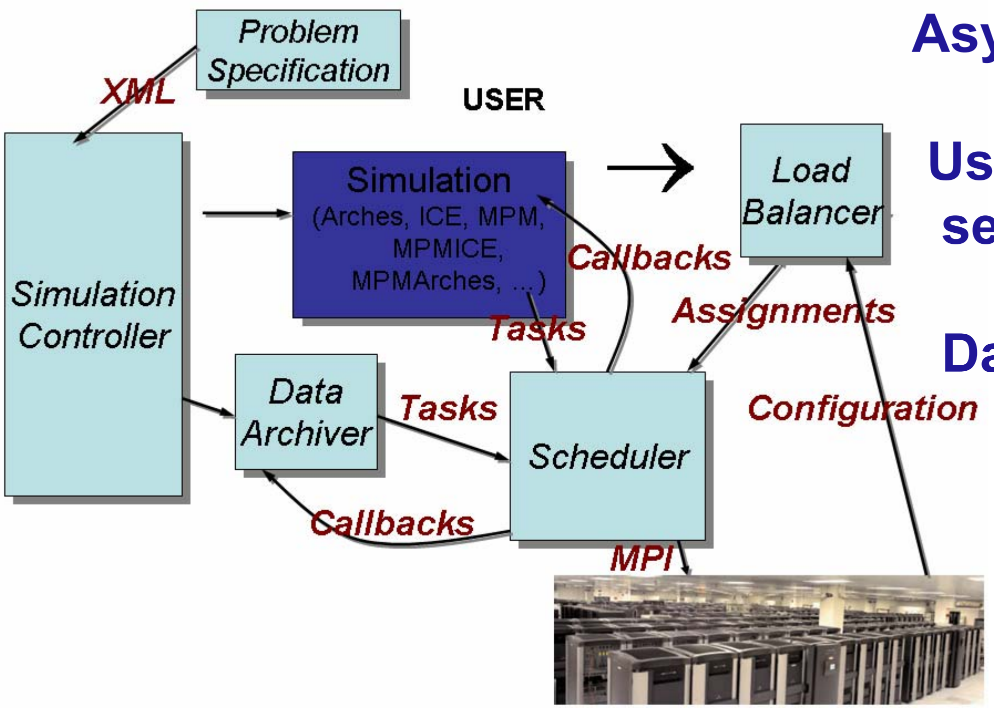


Single Level ICE Model: Scaling
Strong Scaling
Mean Time Per Timestep (s)
10^2
10^1
Linear Model with 0.05 % log(p) at 48 cores
24
48
96
192
384
768
1.5K
3K
6K
12K
24K
49K
98K
196K
Number of Cores

Uintah Parallel Computing Framework

- Uintah (1998-2005) used DOE parallel computers, typical run – 512 to 2K cores far-sighted design by Steve Parker:

Solution of broad class of fluid-structure interaction problems
Patch-based AMR using particles and mesh-based fluid solver
Automated task-graph generation for scheduling parallelism
Automated load balancing



Asynchronous communication

User only writes “serial” code for a serial mesh patch

Data is pulled from a data warehouse and results sent there

Uintah Parallel Computing Framework

- Uintah had "legacy" code aspects –original design sound
- MUCH OF THE CODE HAS BEEN REWRITTEN

- **New scalable AMR algorithm**
- **New measurement-based load balancer**
- **Dynamic execution (including out of order) of tasks**
- **New Hybrid MPI/Pthreads execution model**
- **Better use of Hypre solvers for time-dependent problems**
- **Much algorthmic development of discretisation methods**
-
- **Uintah now uses NSF (Ranger Kraken) DOE (Jaguar) computers, typical run – 2K to 196K cores**

- **How do we apply Uintah to model Developing Detonations? How do we start to think about scaling to beyond petascale**

Spanish Fork Accident 8/10/05

Speeding truck with 8000 explosive boosters each with 2.5-5.5 lbs of explosive

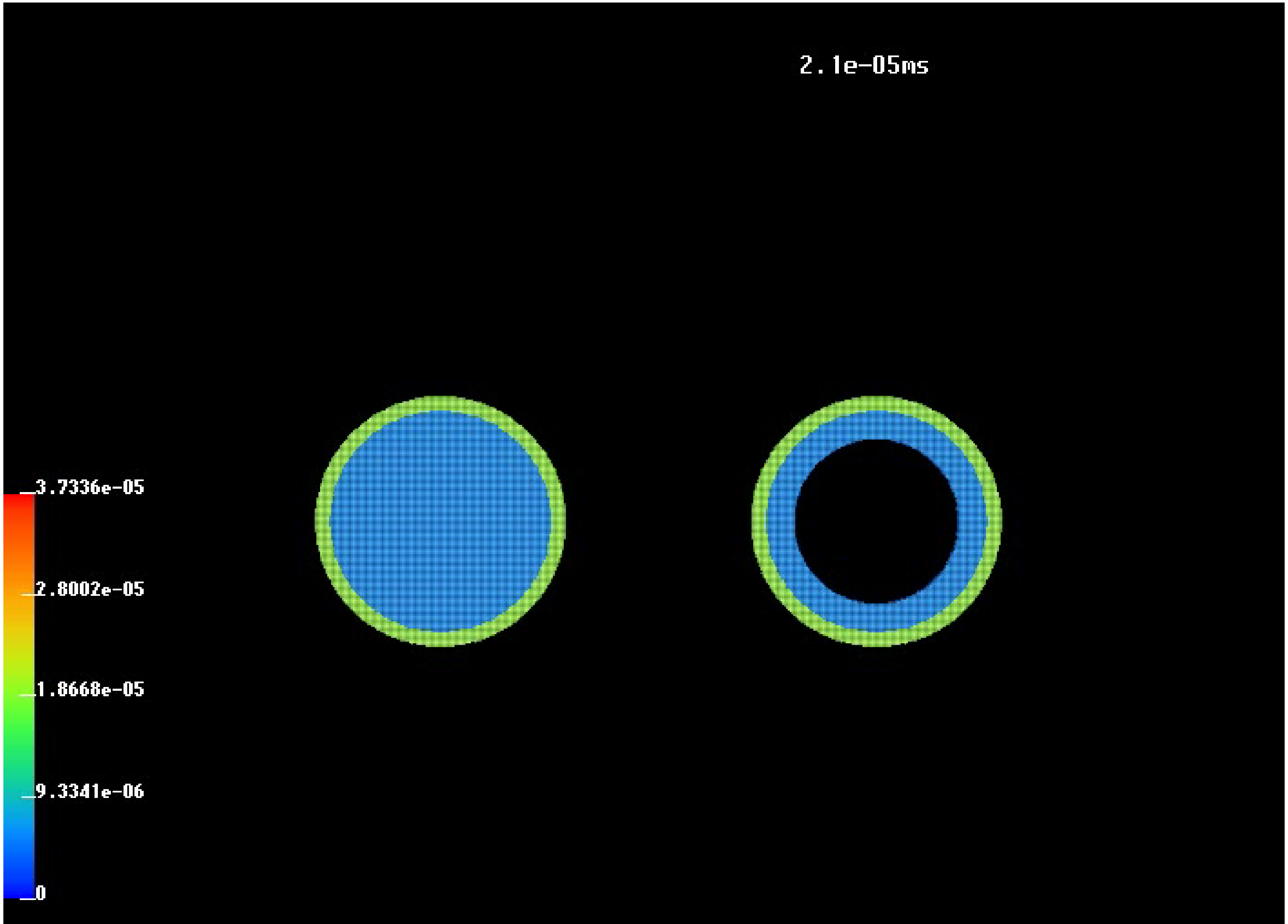
Experimental evidence suggests that a transition from deflagration to detonation took place. Why?

How can we prevent this?

Images from KUTV and Deseret News

Counterintuitive Dual Container Experiment

left – solid explosive, right- explosive with air has 4x energy release



Explosive Mass Burned Comparison

Mass

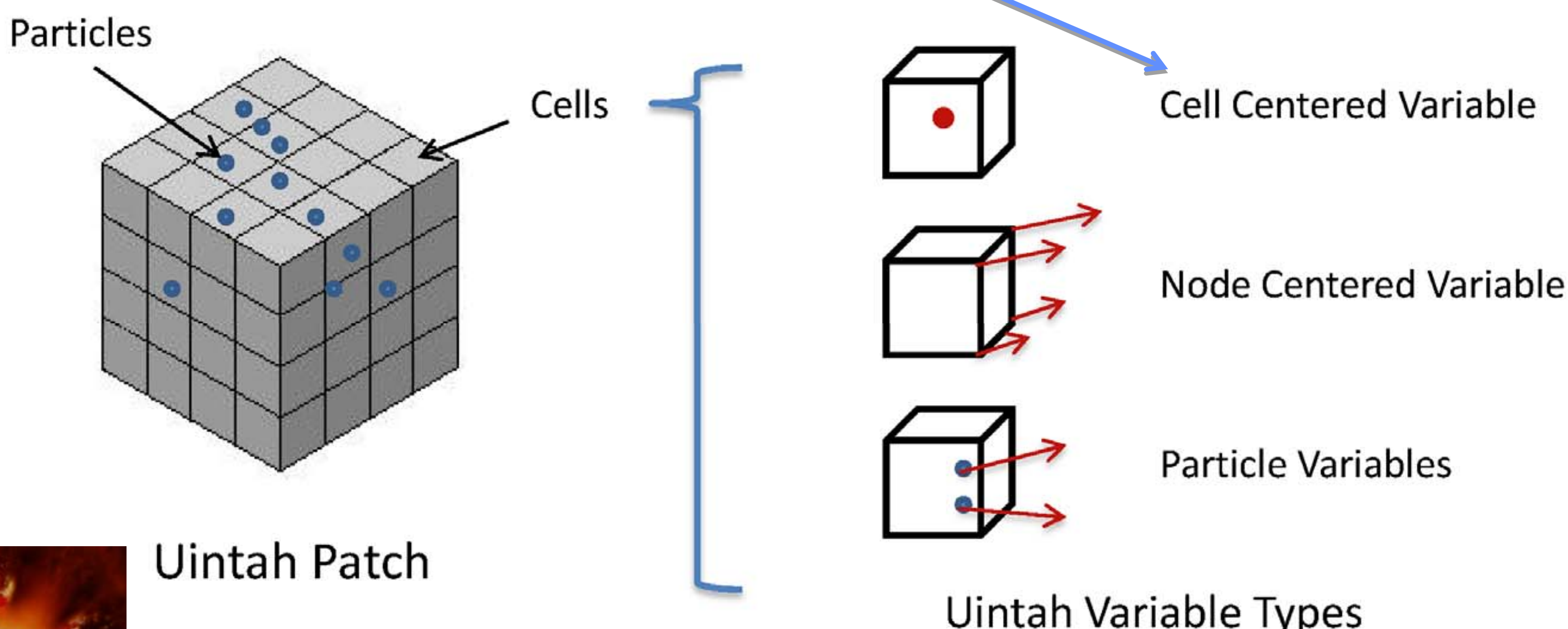
Time

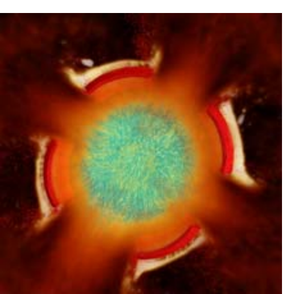
Large Bore Hollow case burns 4.5 x Mass of solid case

Uintah MPM-ICE-AMR Software

MPM (solids) and ICE (fluids) exchange data several times per timestep (not just boundary condition exchange

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





MPM is a novel method that uses particles and nodes

Cartesian grid used as a common frame of reference

ICE Algorithm

Consider convection-diffusion type form

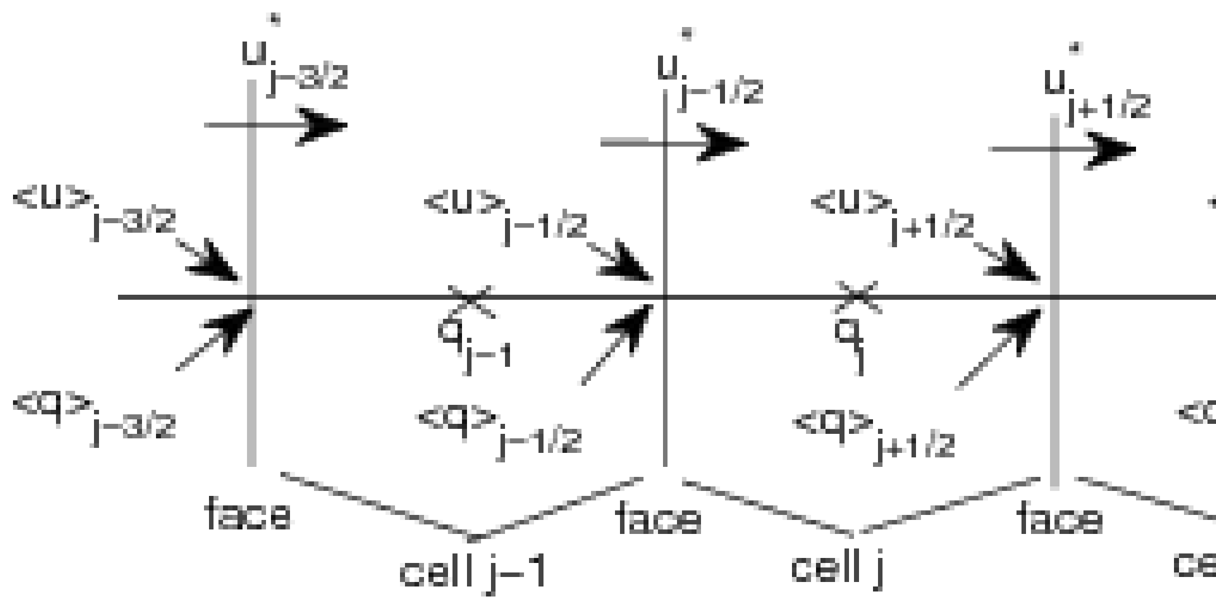
$$\frac{\partial q}{\partial t} + \frac{\partial (uq)}{\partial x} = -\frac{\partial f^{p}(q)}{\partial x}$$

Fluxing velocity $u^{*}_{j+1/2}$ **at** $t_n + \Delta t/2, x_j + \Delta x/2$

Lagrangian $$q^{l}_{j}V^{l}_{j} = q^{n}_{j}V^{n}_{j} - \Delta t(f^{p}(< q^{n+1/2}_{j+1/2} >) - f^{p}(< q^{n+1/2}_{j-1/2} >))$$

Eulerian $$q^{n+1}_{j}V^{n+1}_{j} = (q^{l}_{j}V^{l}_{j}) - \Delta t(< q >^{n}_{j+1/2} u^{*}_{j+1/2} - < q >^{n}_{j-1/2} u^{*}_{j-1/2})$$

Apply limiters at a number of stages To get a positivity preserving algorithm



ICE Algorithm

Original Algorithm for face value at t_n

$$< u >_{j+1/2} = \frac{\rho_j u_j + \rho_{j+1} u_{j+1}}{\rho_j + \rho_{j+1}}$$

Left Value

$$u^L_{j+1/2} = u_j + \frac{1}{2}\Phi(r_j)(u_j - u_{j-1}), r_j = \frac{u_{j+1} - u_j}{u_j - u_{j-1}}$$

Right Value

$$u^R_{j+1/2} = u_{j+1} - \frac{1}{2}\Phi(r_{j+1})(u_{j+2} - u_{j+1}), r_{j+1} = \frac{u_{j+1} - u_j}{u_{j+2} - u_{j+1}}$$

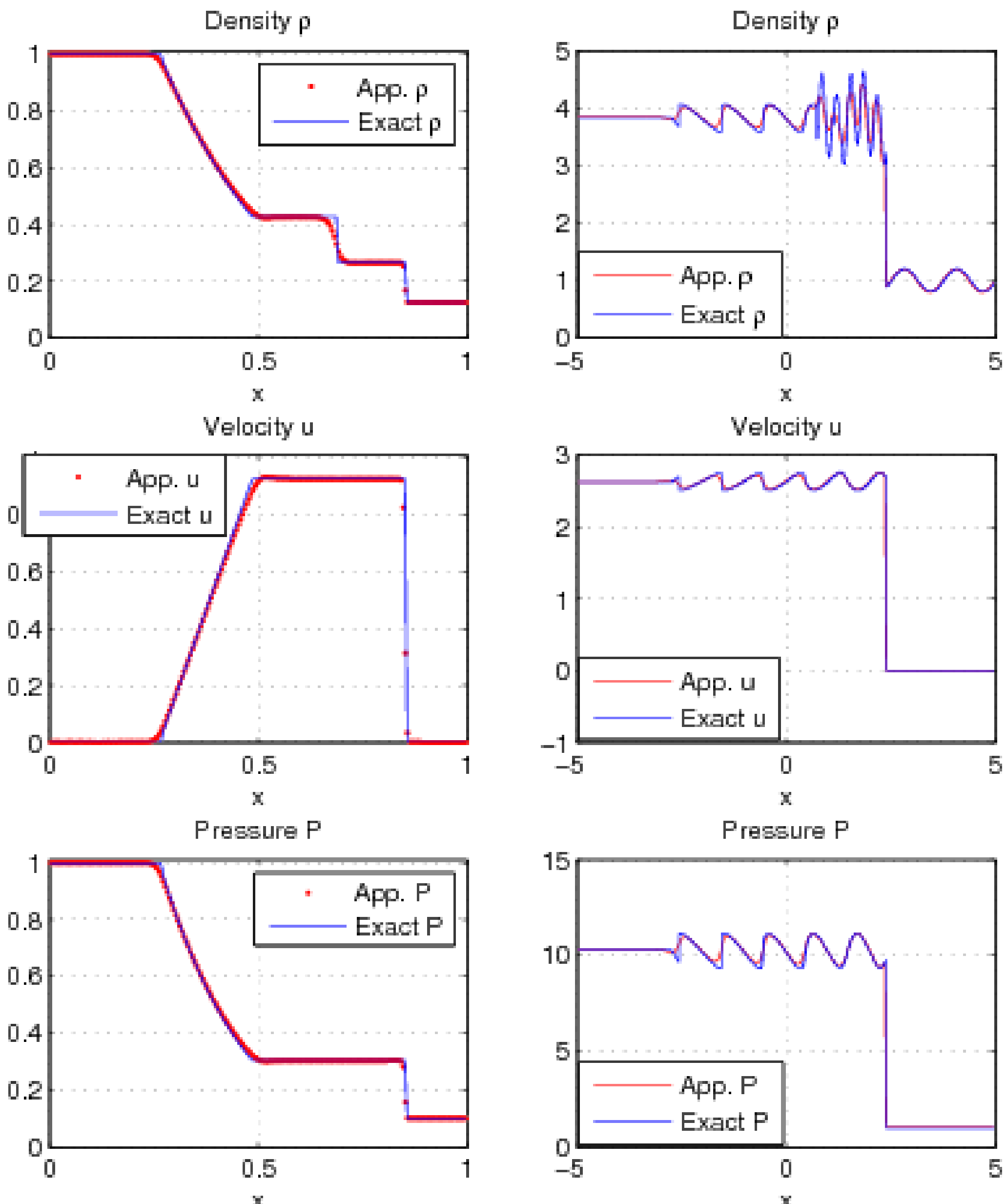
If $u^L_{j+1/2}\rho^l_{j+1/2} + u^R_{j+1/2}\rho^R_{j+1/2} > 0$

then

$< u >_{j+1/2} = u^L_{j+1/2}$ **else** $< u >_{j+1/2} = u^R_{j+1/2}$

$\Phi(r)$ **is a standard limiter as used for hyperbolic eqns**

ICE Navier Stokes Algorithm of Kashwa et al. Improved 08/09 for High Speed Flow Tran and Berzins



Models fluids in Uintah

Euler Equations
Elimination of oscillations and second order version

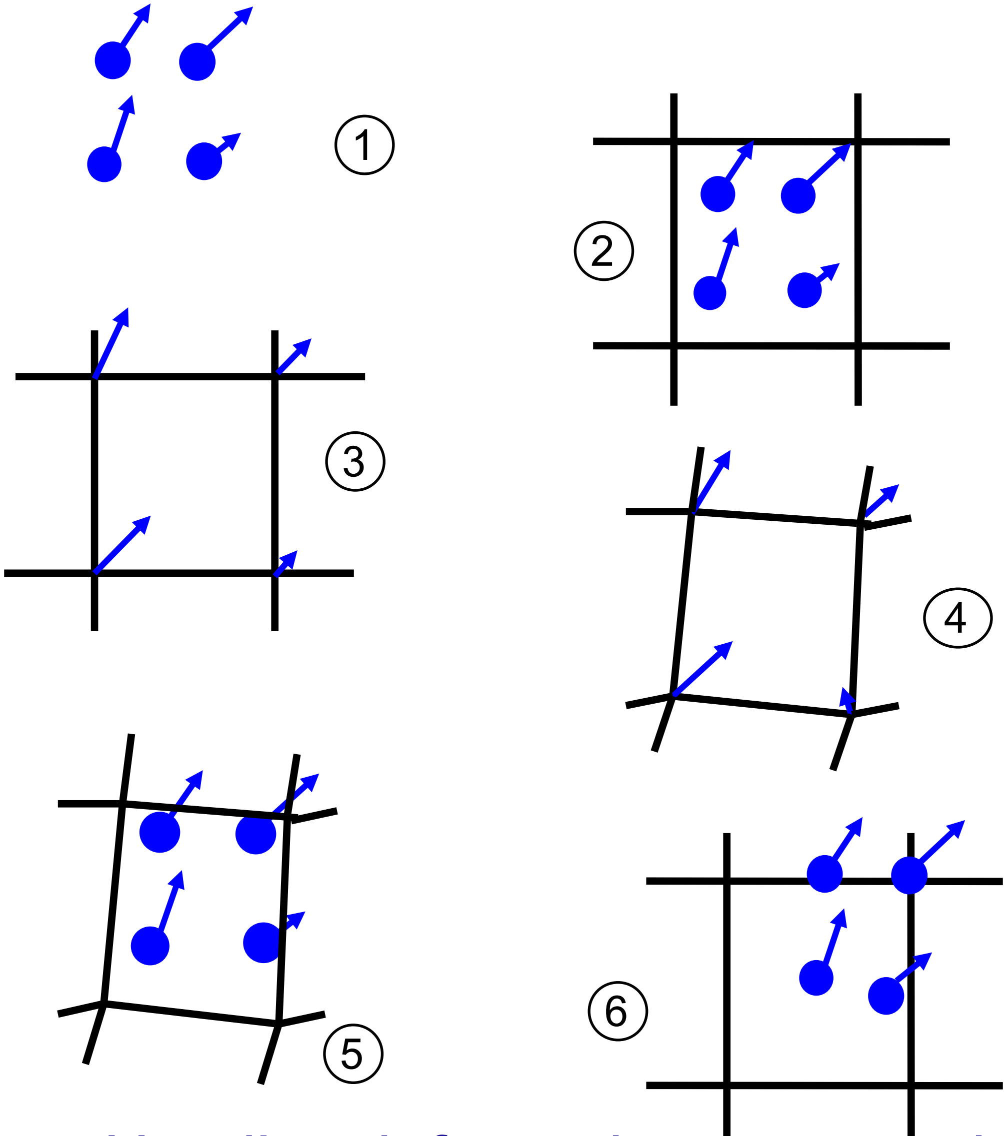
Examples:

Left : Sod Shock Tube with 200 points

Right: Shu-Osher Problem With 800 points

The Material Point Method (MPM)

Sulsky Guilkey Bardenhagen et al.



Particles with properties (velocity, mass etc) defined on a mesh

Particle properties mapped onto mesh points

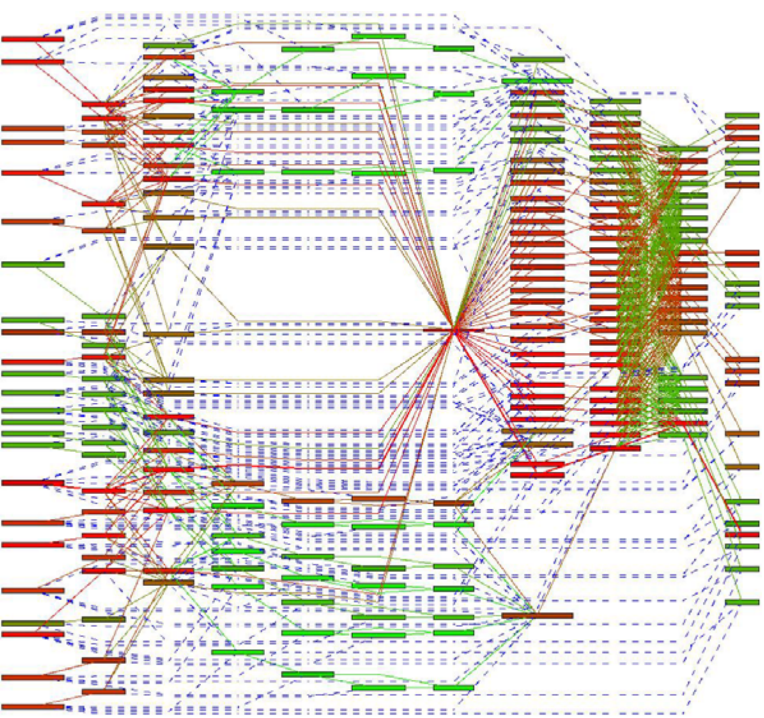
Forces, accelerations, velocities calculated on mesh points

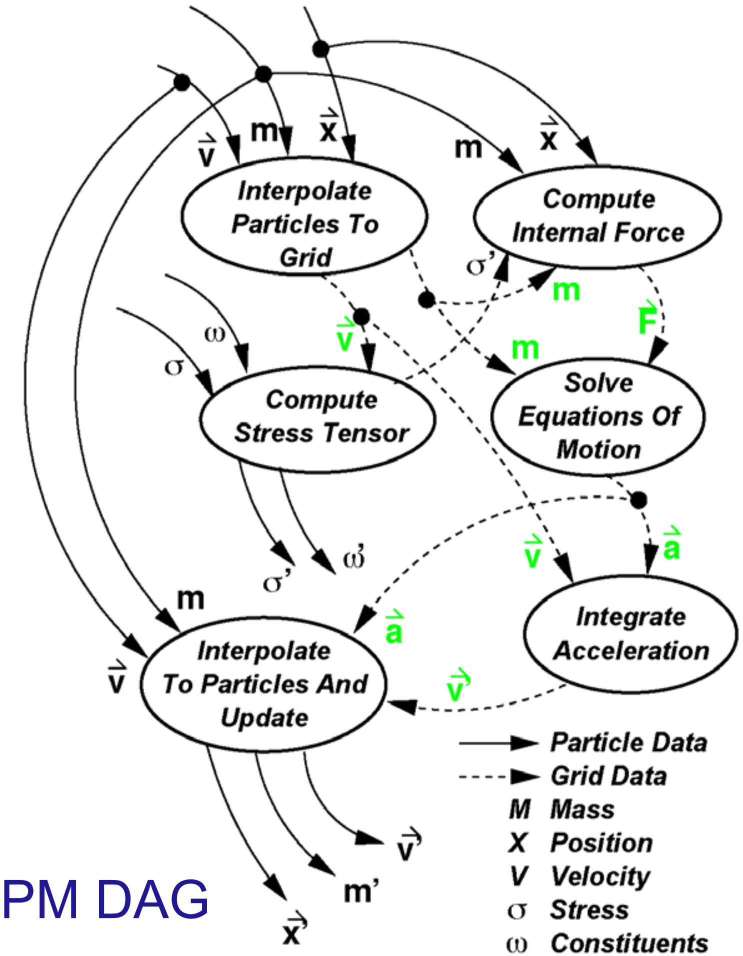
Mesh point motion calculated but only the particles moved by mapping velocities back to particles

Handles deformation, contact, high strain, fragmentation models solids in Uintah

Directed Acyclic Graphs (DAGs)

- 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)
- Similar to Charm++ TBlas, CnC DAG approach increasingly popular for efficient parallelism with irregular communications



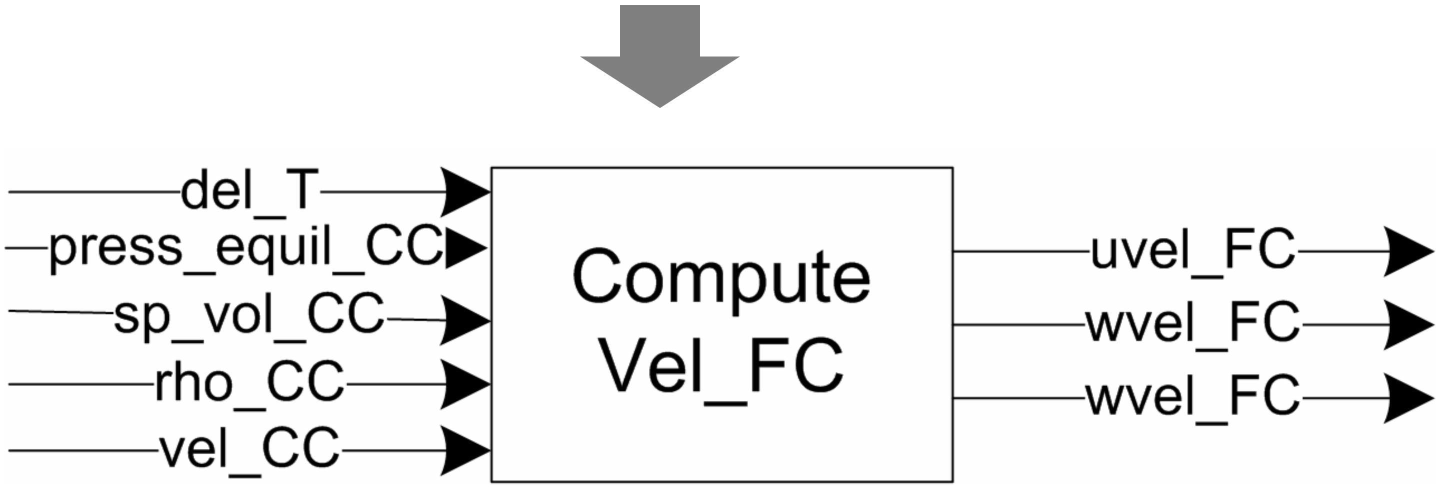


MPM DAG

Example Uintah Task from the ICE Algorithm

Compute face-centered Velocities:

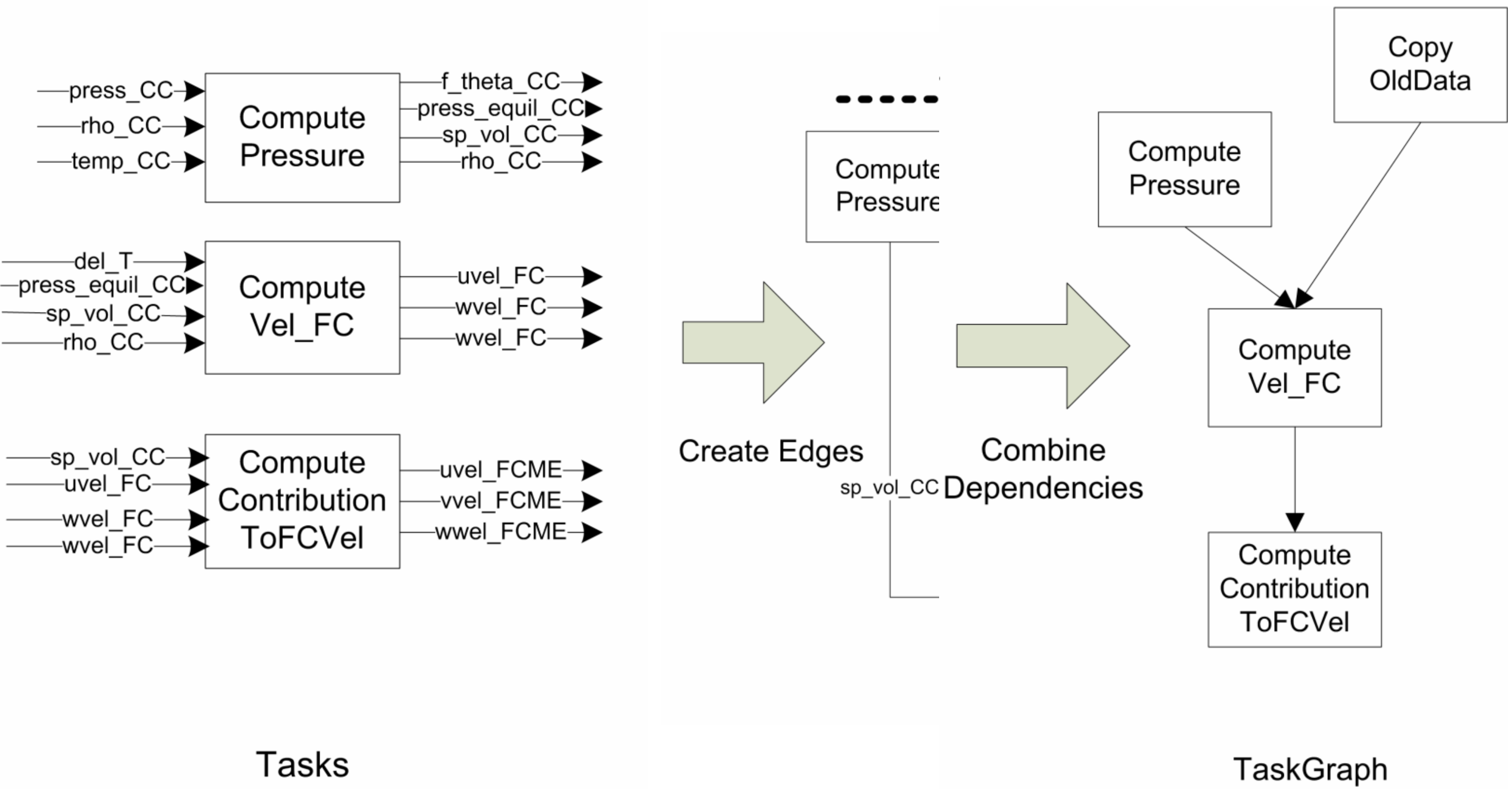
$$\vec{U}^{*f} = f(\Delta t, P_{eq}, \vec{g}, \rho, \vec{U})$$



Input variables
(include boundary conditions)

Output variables

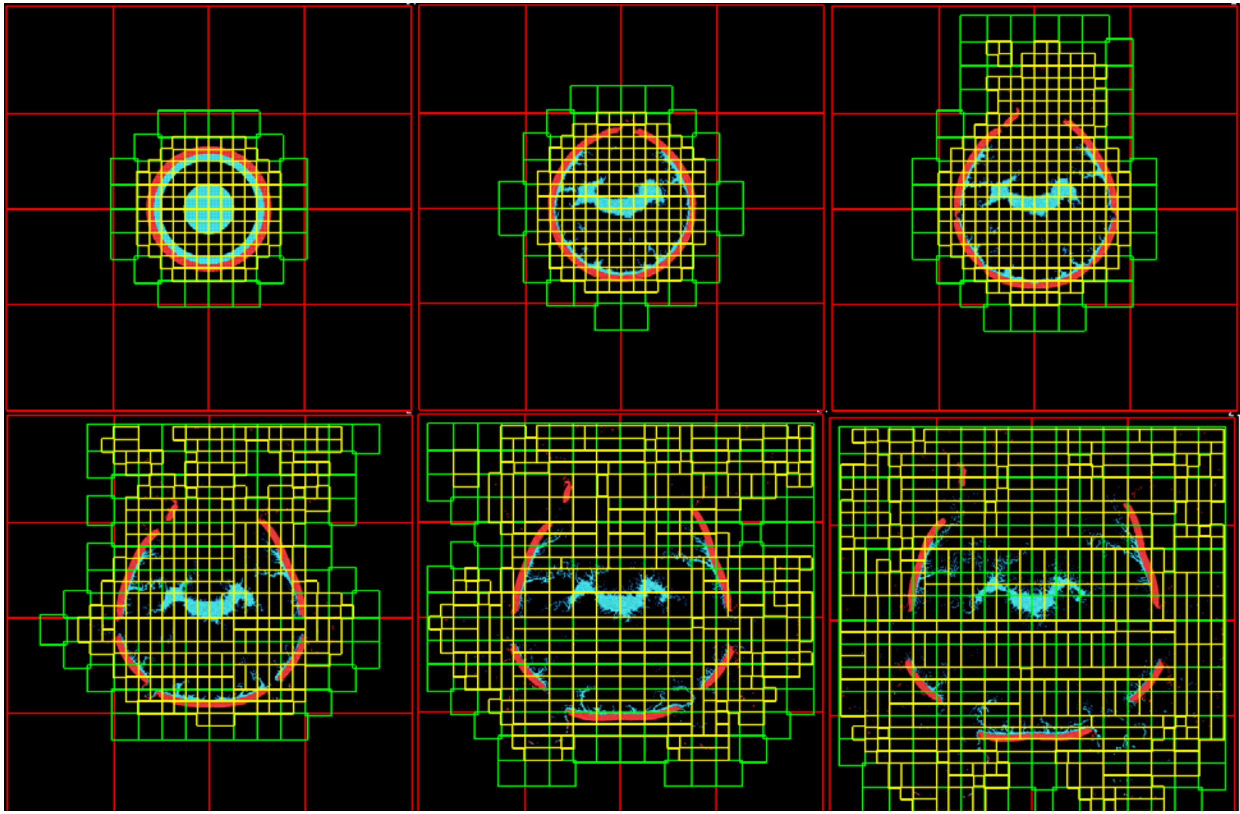
Task Graph Compiling



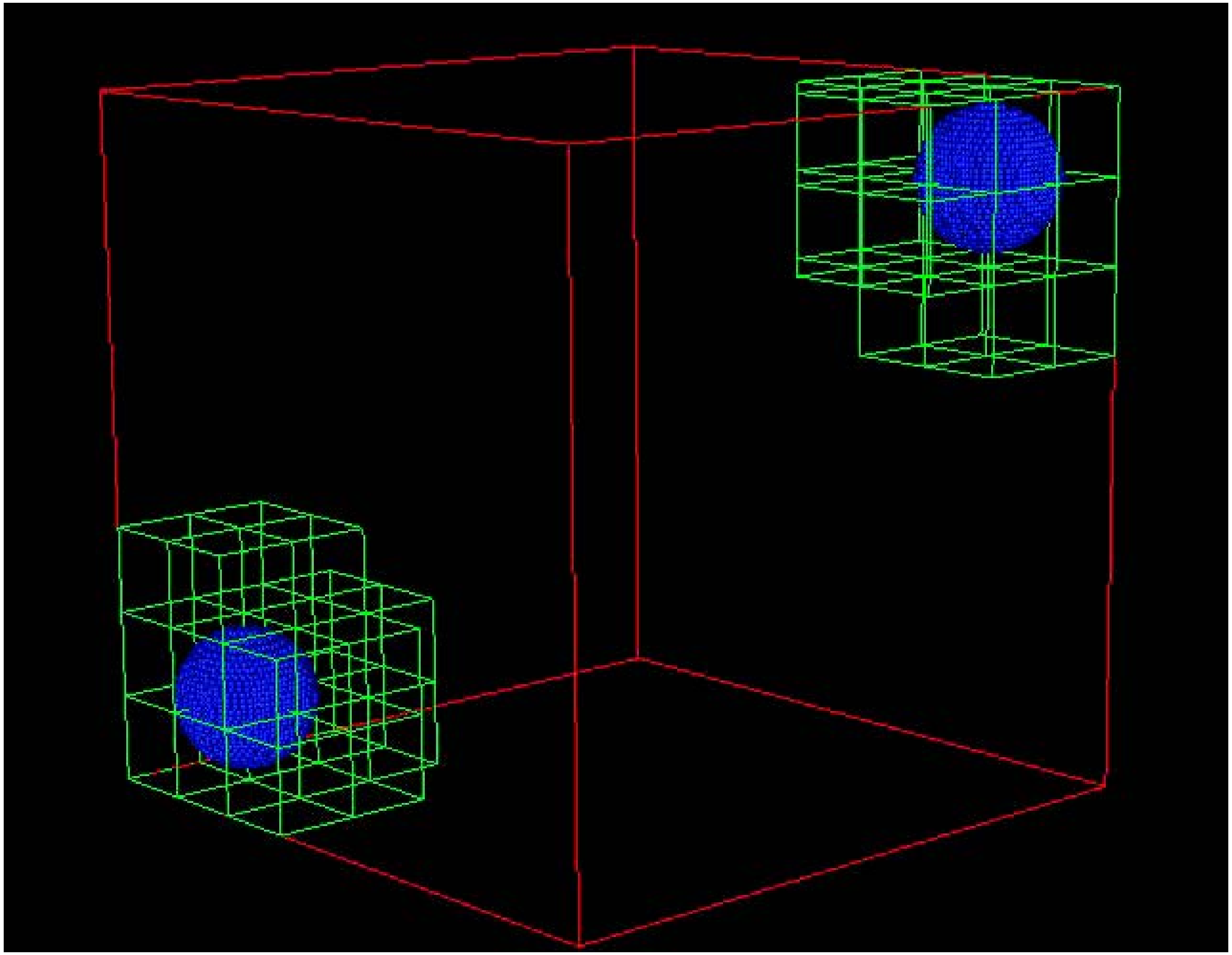
AMR for Multiple Space/Time Scales

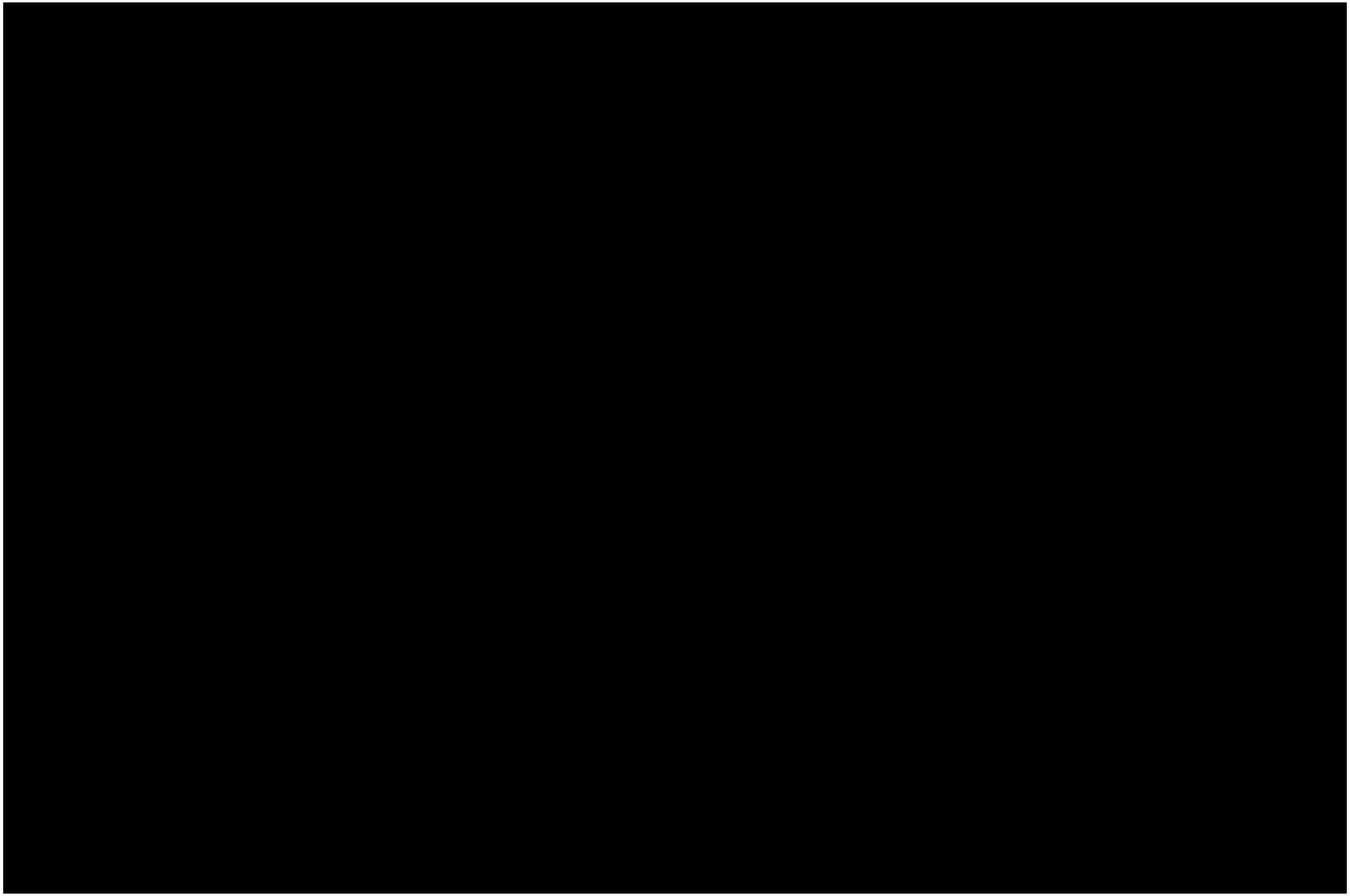
- Solvers that are designed to work together
- One mesh for all material phases

Refine/coarsen spatial mesh where gradients/ errors are large/small



EXAMPLE OF UINTAH MOVING MESHES WITH PARTICLES



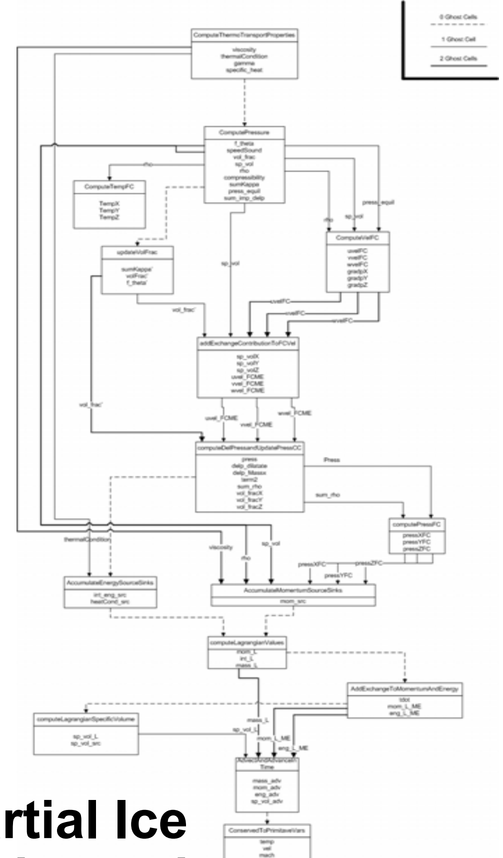


End to end simulation of container

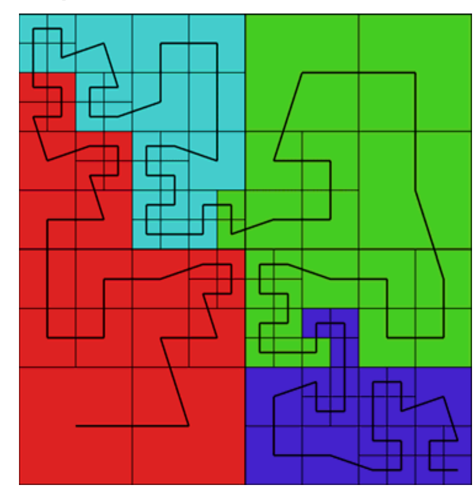
Performance Improvements

- Petascale apps require 300K cores and 1M+ AMR patches
- **Improve Algorithmic Complexity**
  - Identify & Eliminate O(Processors) O(Patches) via Tau, and hand profiling, memory usage analysis
- **Improve Task Graph Execution**
  - Out of order execution of task graph   e.g. ➔
  - Approach for multicores via multi-threading
- **Improve Load Balance**
  - Cost Estimation Algorithms based on data assimilation
  - Use load balancing algorithms based on patches and a new fast space filling curve algorithm ➔ [Concurrency] Complexity is:

$$c\frac{Nb}{P}\log(Nb)+(c+c\frac{Nb}{P})\log^2(P)$$



Partial Ice task graph



Load Balancing Weight Estimation

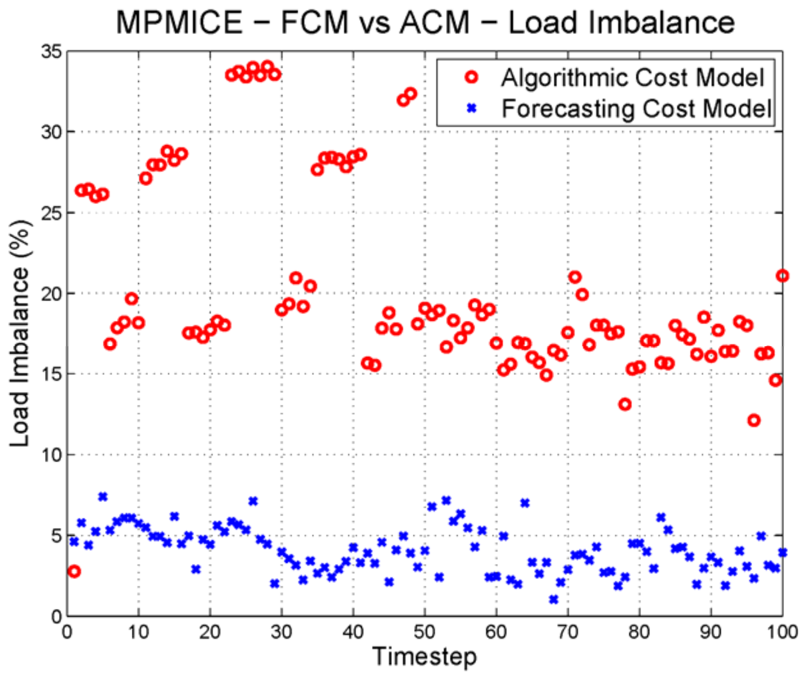
- Algorithmic Cost Models based on discretization method and machine, requires accurate information from the user
- Time Series Analysis used to forecast time for execution on each patch - automatically adjusts according to simulation and architecture with no user interaction

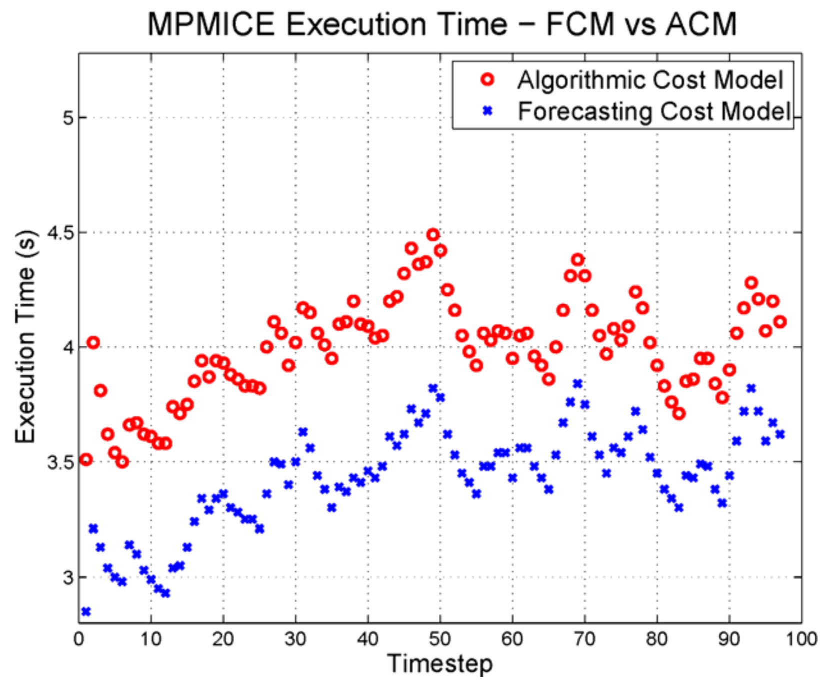
Simple Exponential Smoothing:

$E_{r,t}$: Estimated Time **$O_{r,t}$: Observed Time** **α: Decay Rate**

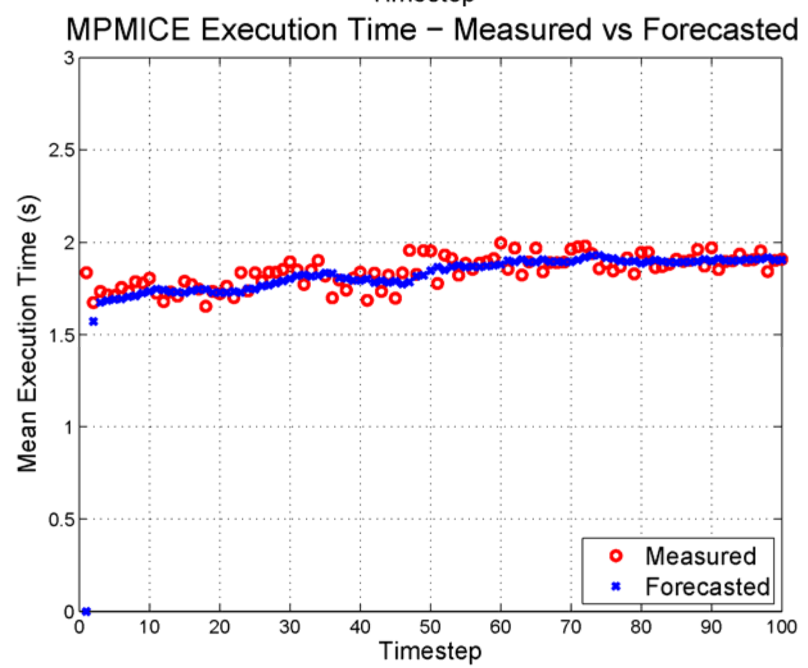
$$E_{r,t+1} = \alpha\, O_{r,t} + (1-\alpha)\, E_{r,t}$$
$$= \alpha\, \underbrace{(O_{r,t} - E_{r,t})}_{\text{Error in last prediction}} + E_{r,t}$$

[IPDPS10 paper], Charm++ uses a similar idea without feedback



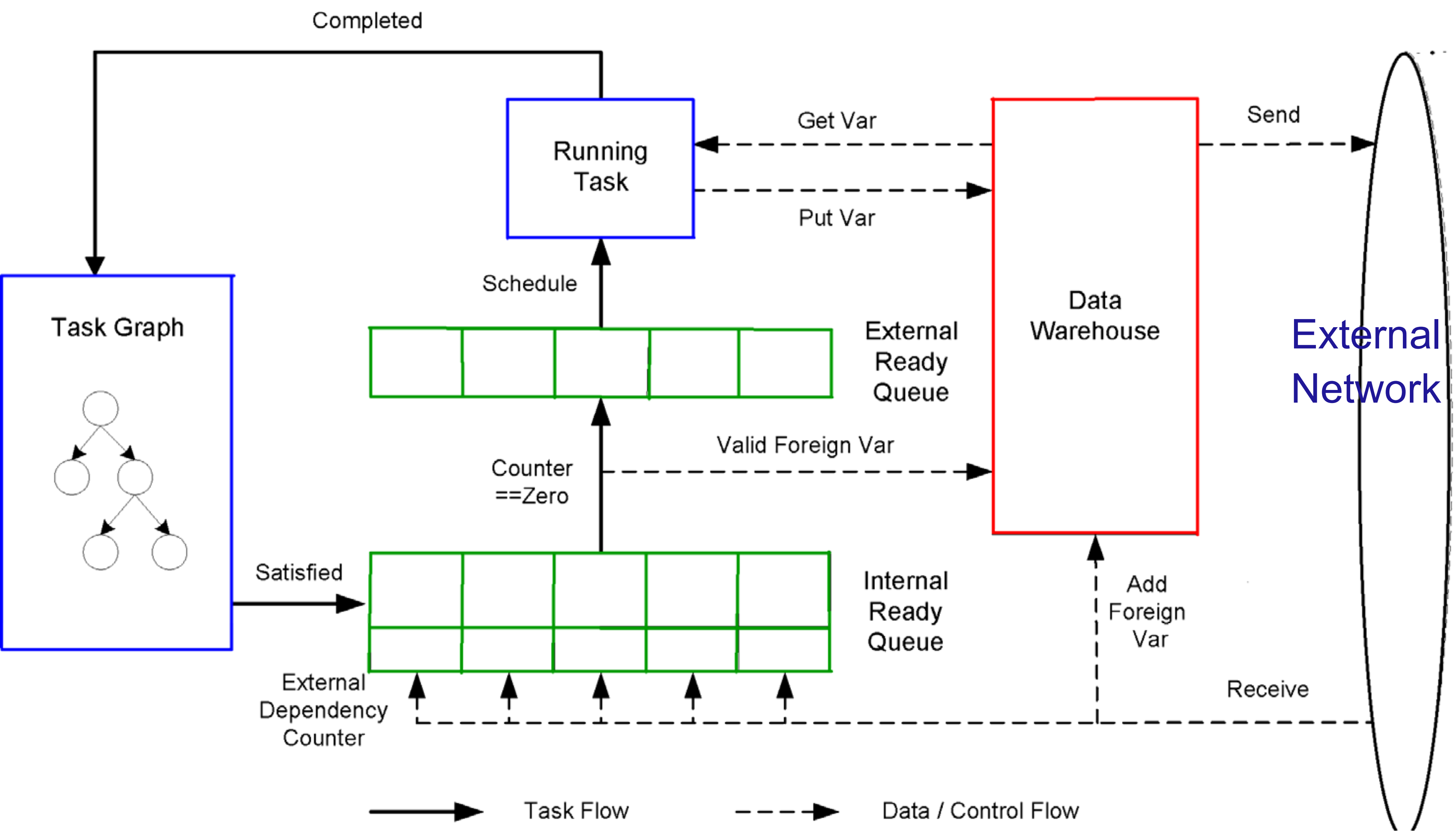


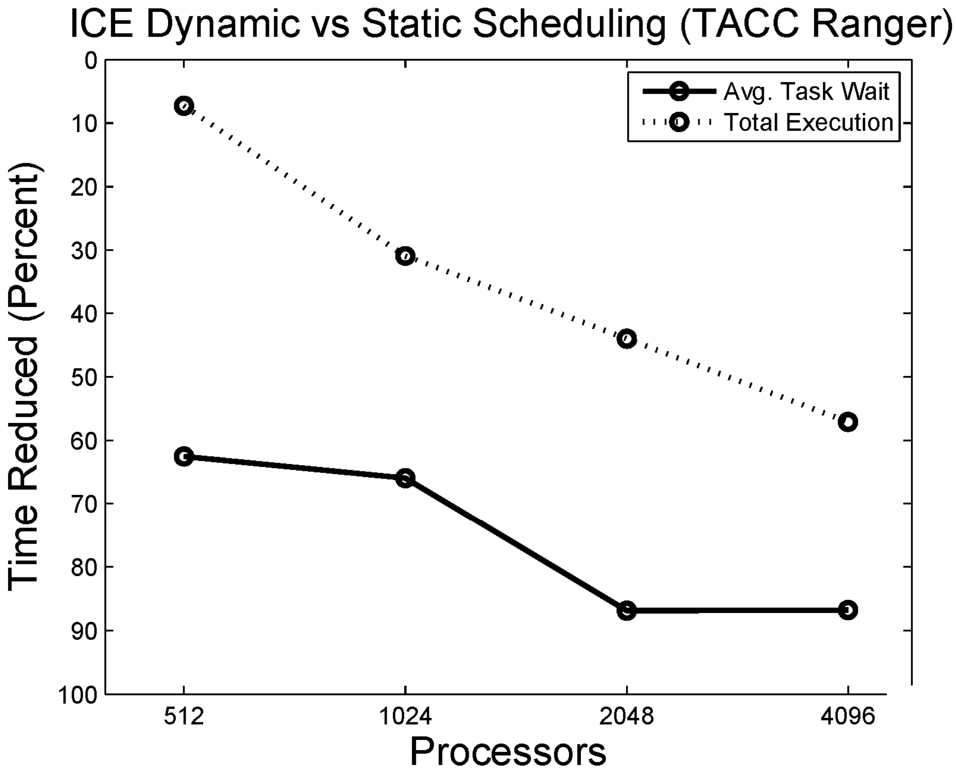
Comparison between
Forecast Cost Model FCM
& Algorithmic Cost Model
Particles + Fluid code
FULL SIMULATION



DYNAMIC TASK EXECUTION

When a task's external dependencies are satisfied it can execute

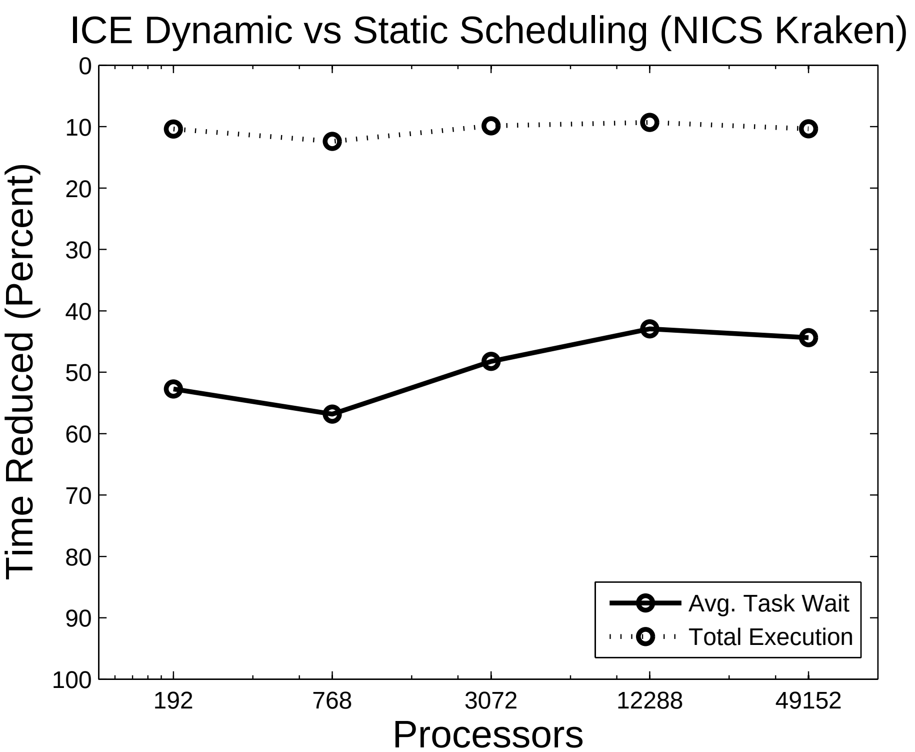




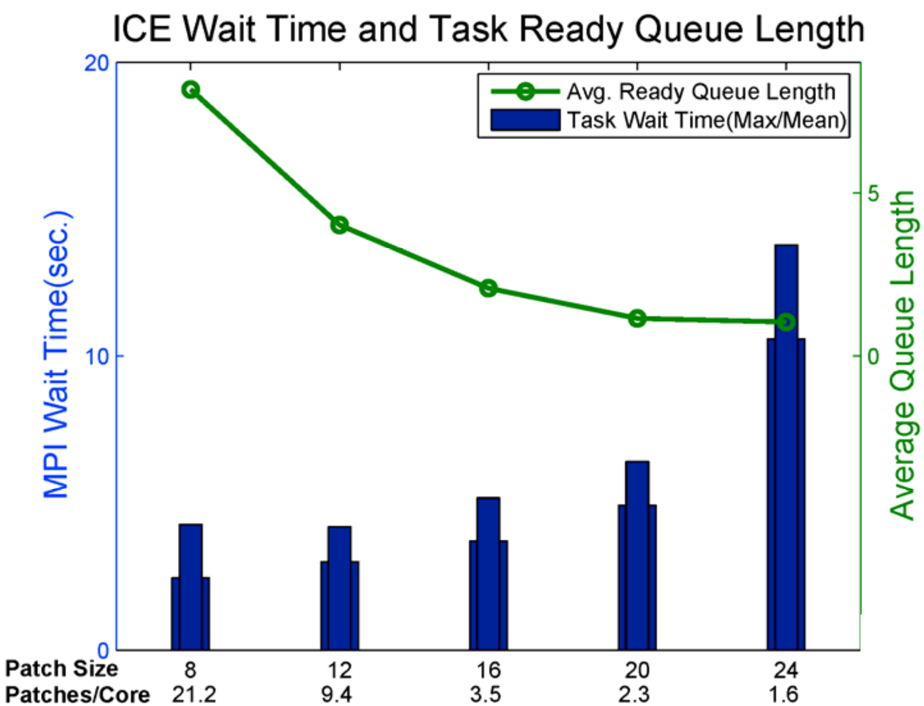
62K AMD coresSun infiniband

Static vs Dynamic Scheduling Improvements

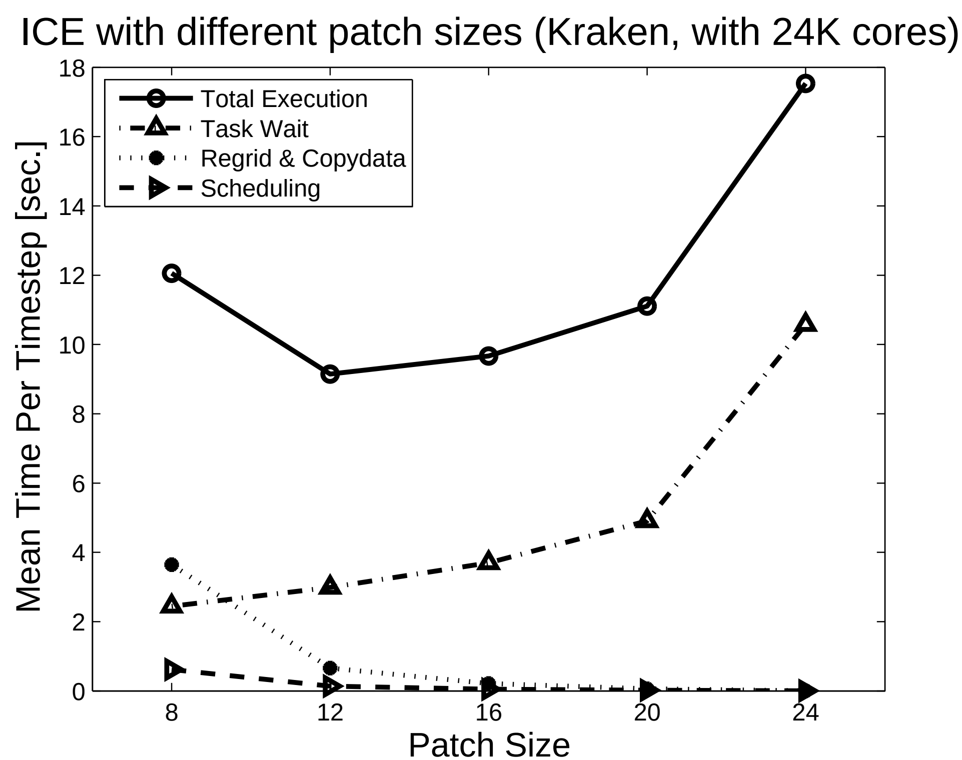
98K AMD 6 cores CRAY Seastar torus



Ranger has slower communications than Kraken and so we see less improvement in overall time through overlapping and out-of-order execution



Task queue length drops and wait time increases as patch sizes grow for a fixed mesh

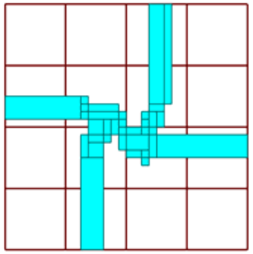


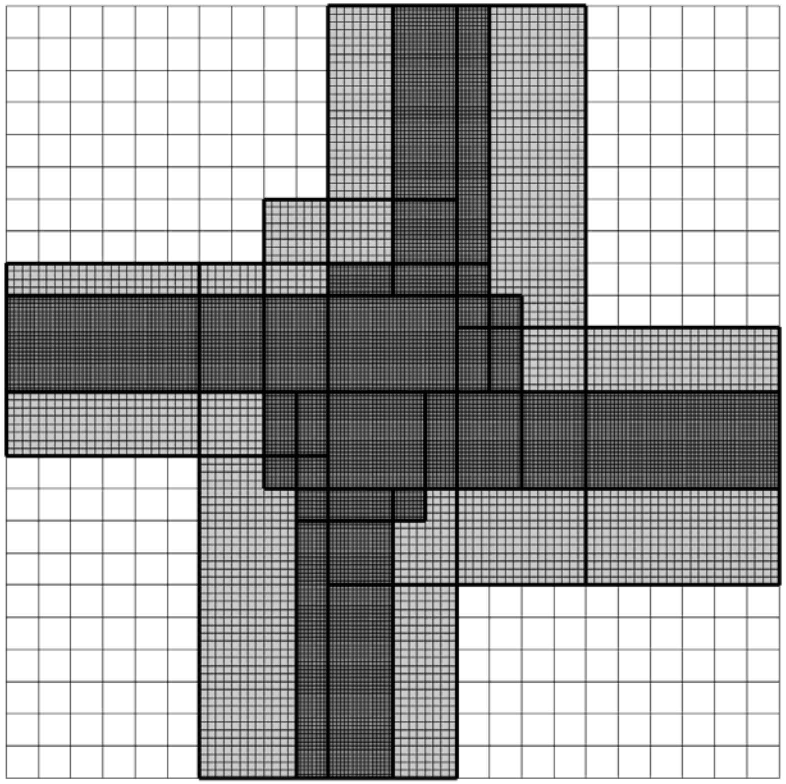
Execution time is a trade-off Between granularity improvements and overhead due to more patches

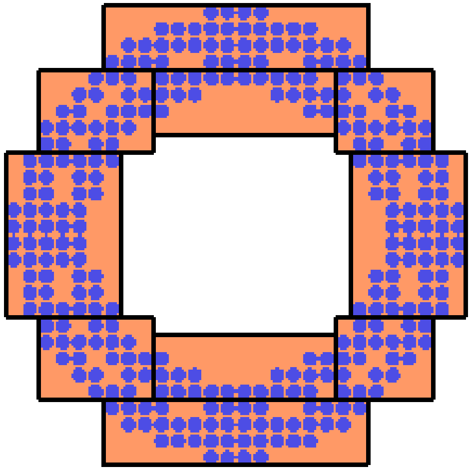
GRANULARITY EFFECTS

Scalable AMR Regridding Algorithm

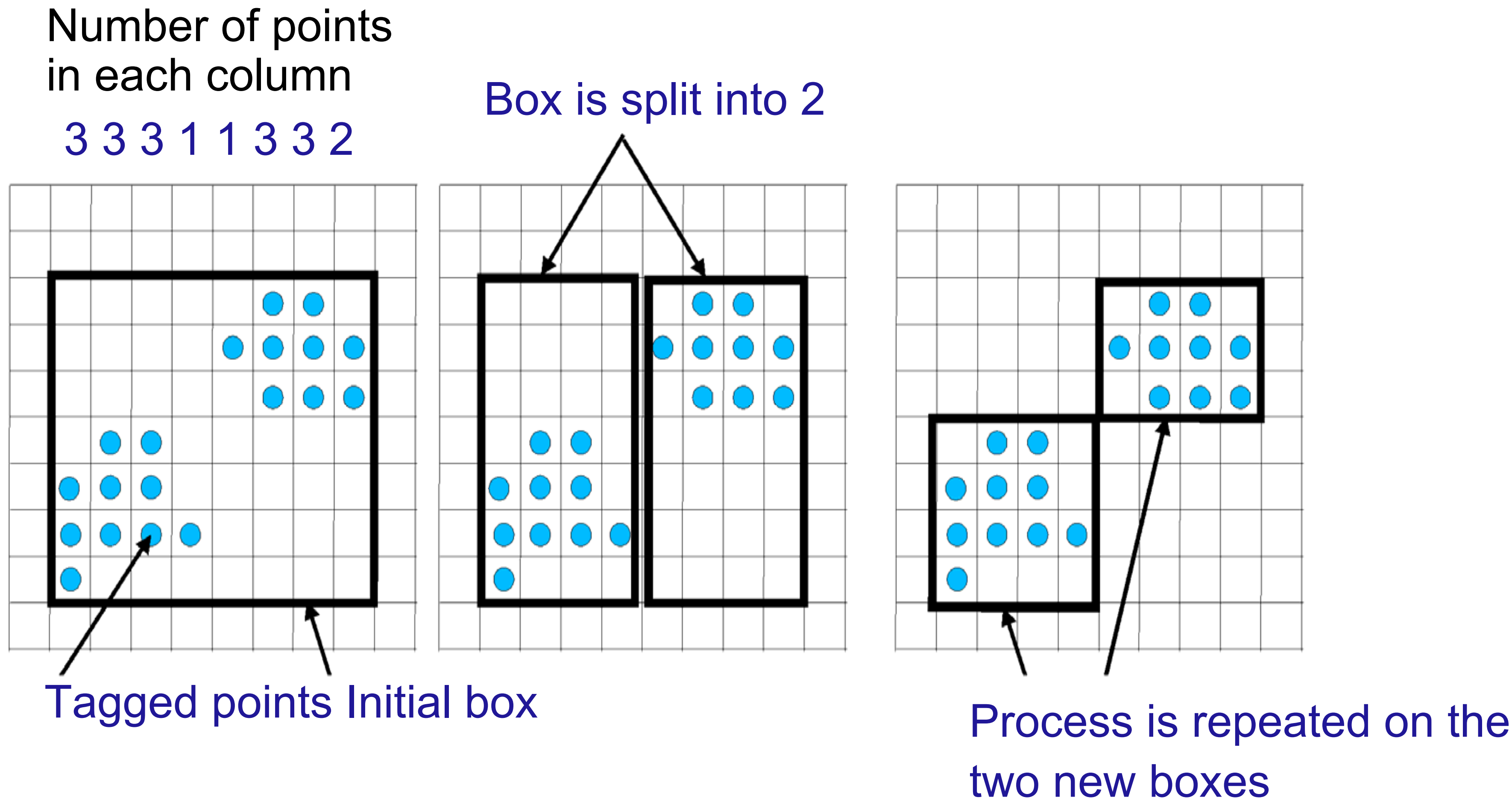
- Berger-Rigoutsos
  - Recursively split patches based on
    histograms
  - Histogram creation requires all-gathers O(Processors)
  - Complex and does not parallelize well
  - Irregular patch sets
  - Two versions – version 2 uses less cores in forming histogram







AMR regridding algorithm (Berger-Rigoutsos)



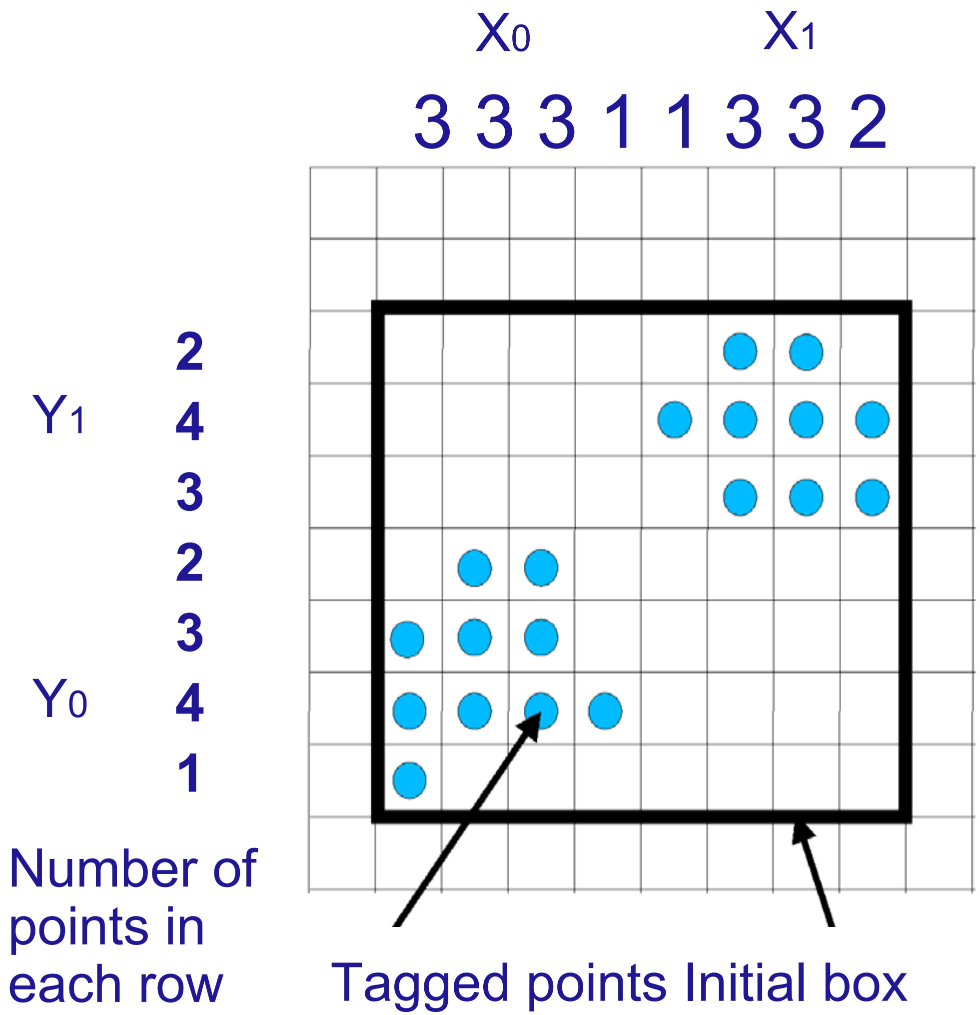
(1) tag cells where refinement is needed

(2) create a box to enclose tagged cells

(3) split the box along its long direction based on a histogram of tagged cells

(4) fit new boxes to each split box and repeat the steps as needed.

AMR regridding algorithm (Berger-Rigoutsos)



Seed points for creation of boxes are (X_0 Y_0) and (X_1 Y_1)

The Laplacian of points In each row and column Is also used to help splitting

This step is repeated at least log(B) times where B is the number of patches

MPI_ ALLGATHER NEED

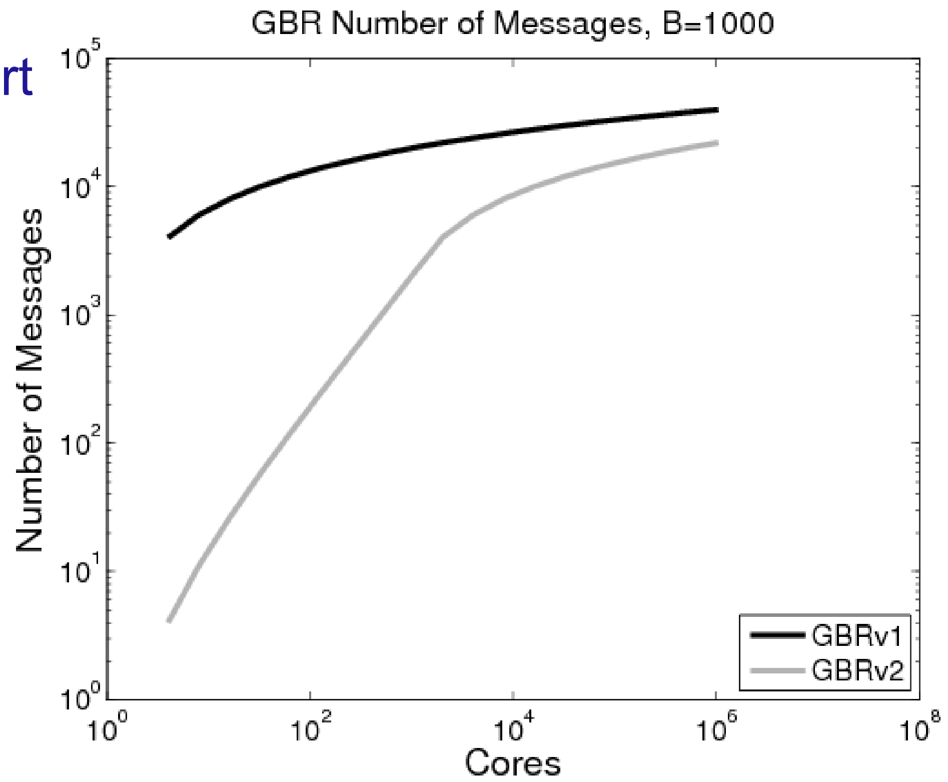
- Every processor has to pass its part of the histogram to every other so that the partitioning decision can be made
- The cost of an allgather is Log(P)

Version 1: (2B-1)log(P) messages

Version 2: Only certain cores take part [Gunney et al.]

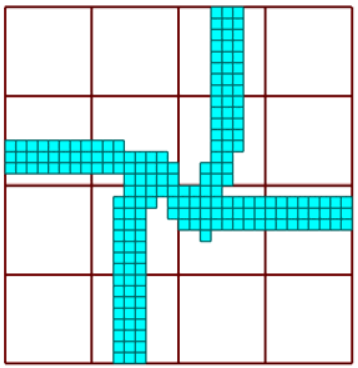
2P –log(P)-2 messages

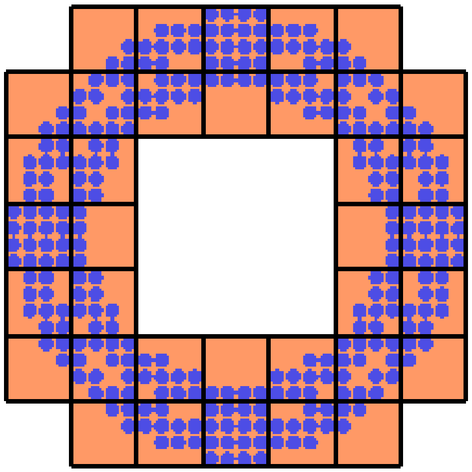
Alternative: use Berger Rigoutsos locally on each processor



Scalable AMR Regridding Algorithm

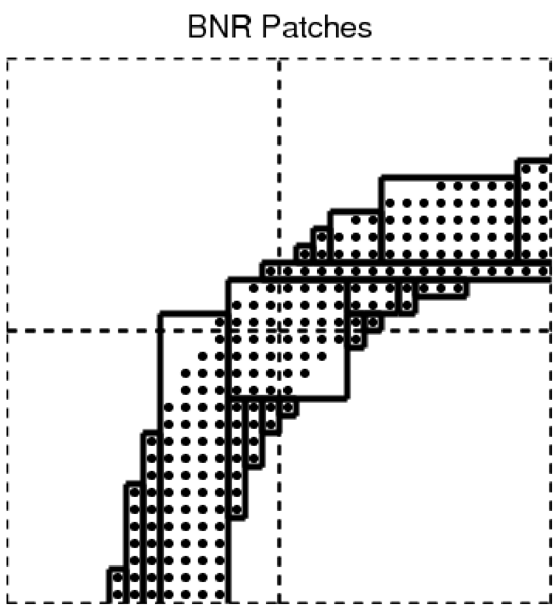
- Tiled Algorithm
  - Tiles that contain flags become patches
  - Simple and easy to parallelize
  - Semi-regular patch sets that can be exploited
    - **Example: Neighbor finding**
  - Each core processes subset of refinement flags in parallel-helps produce global patch set

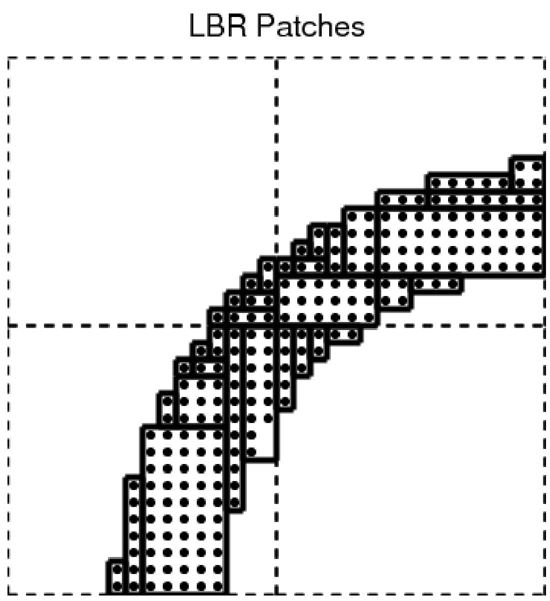


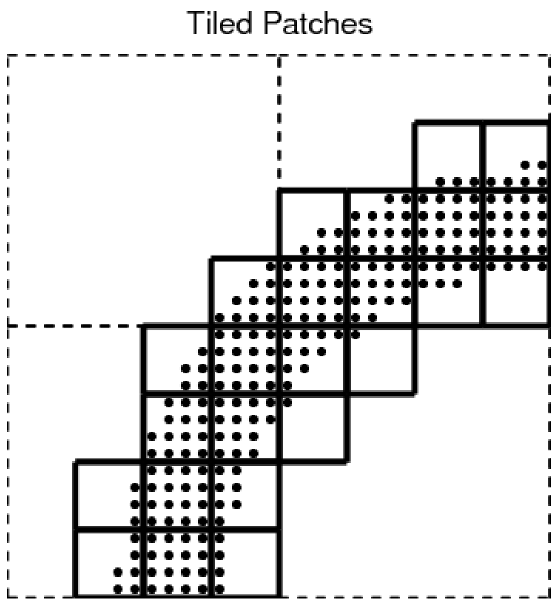


[Analysis to appear in Concurrency]

EXAMPLE – MESH REFINEMENT AROUND A CIRCULAR FRONT





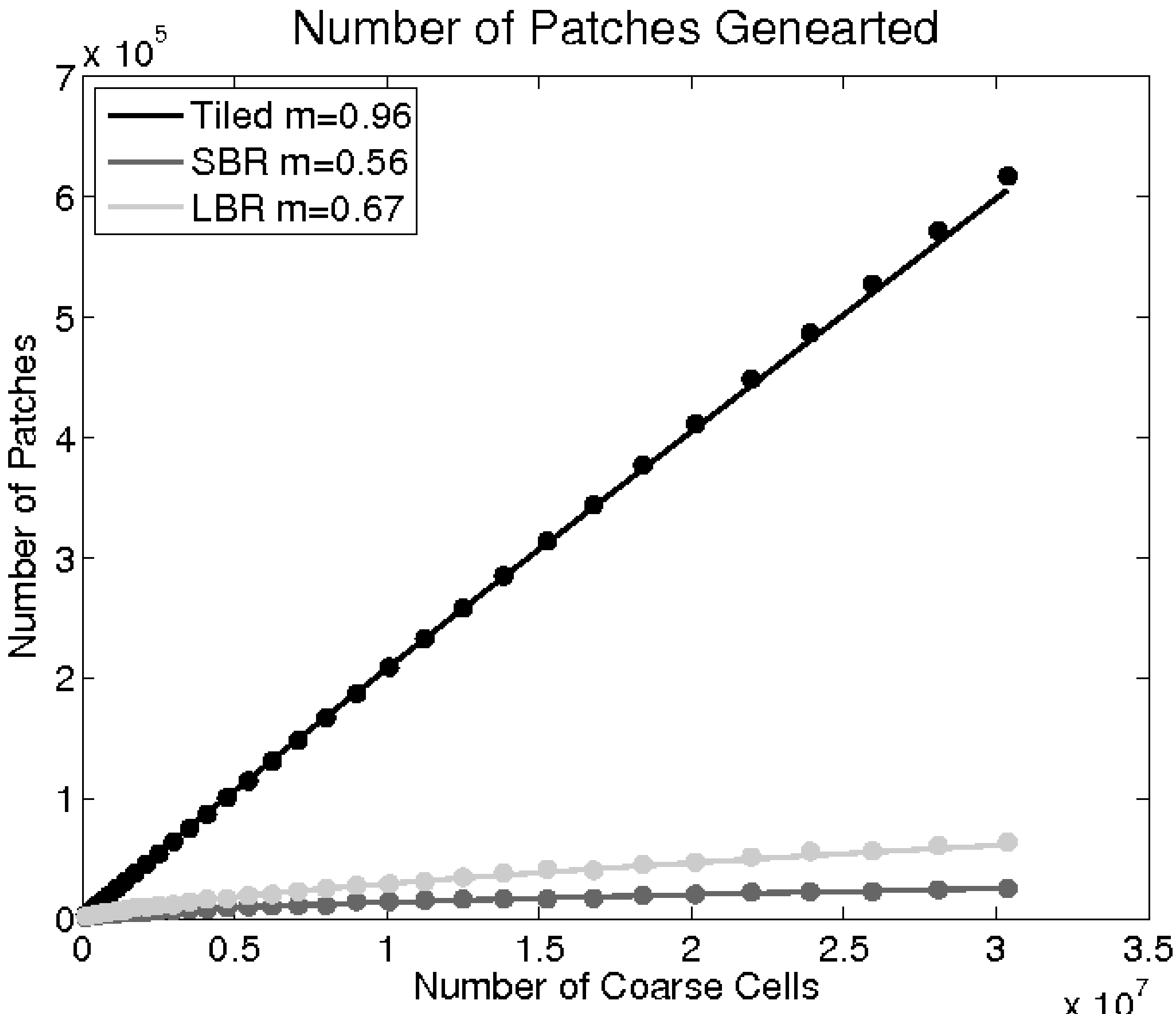


Global Berger-Rigoutsos

Local Berger-Rigoutsos

Tiled Algorithm

NUMBER OF PATCHES GENERATED



The number of patches needs to grow with the number of cells on a parallel machine – otherwise we need to partition patches

Theoretical Models of Refinement Algorithms

C = number of mesh cells in domain

F = number of refinement flags in domain

B = number of mesh patches in the domain

B_c = number of coarse mesh patches in the domain

P = number of processing cores

M = number of messages

T: **GBRv1** $= c_1 (F/P) \log(B) + c_2 M(\text{GBRv1})$

T: **GBRv2** $= c_1 (F/P) \log(B) + c_3 M(\text{GBRv2})$

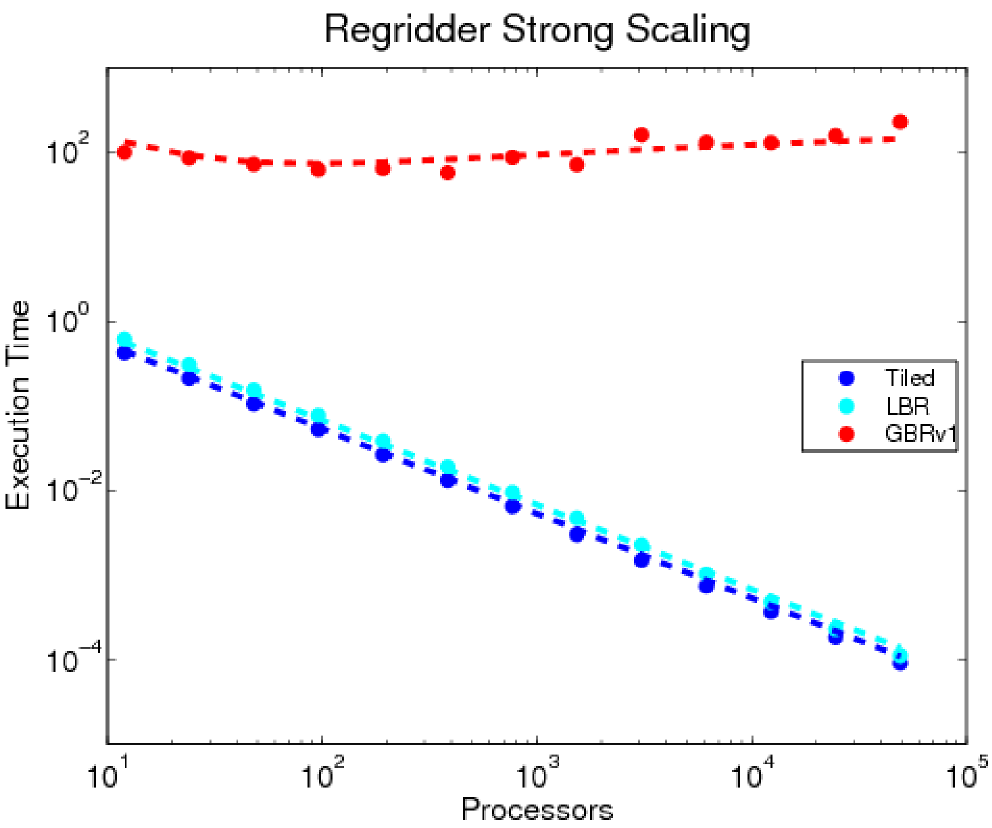
T: **LBR** $= c_4 (F/P) \log(B/B_c)$

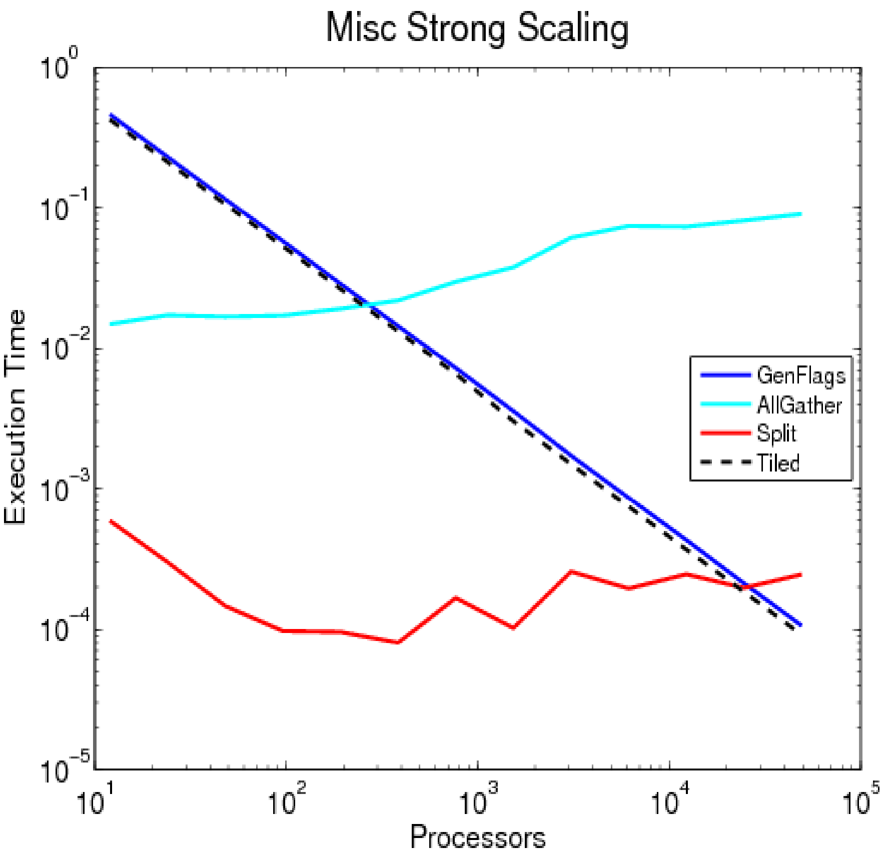
T: **Tiled** $= c_5 C/P$

T: **Split** $= c_6 B/P + c_7 \log(P)$ - is the time to split large patches

Strong Scaling of Algorithms and their Components

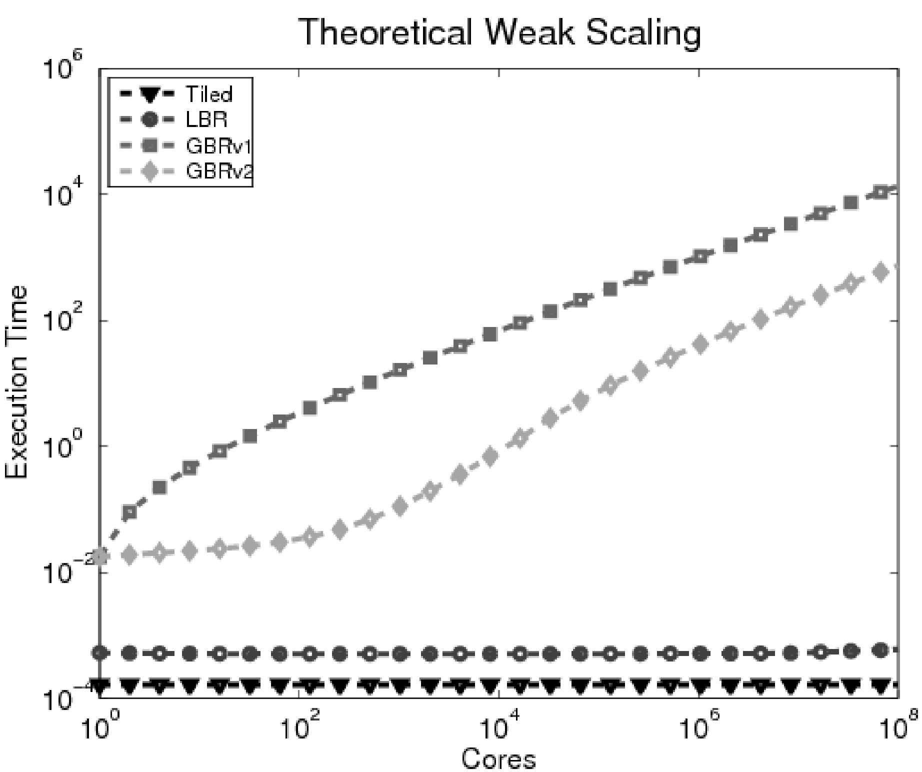
Dots are data lines are models

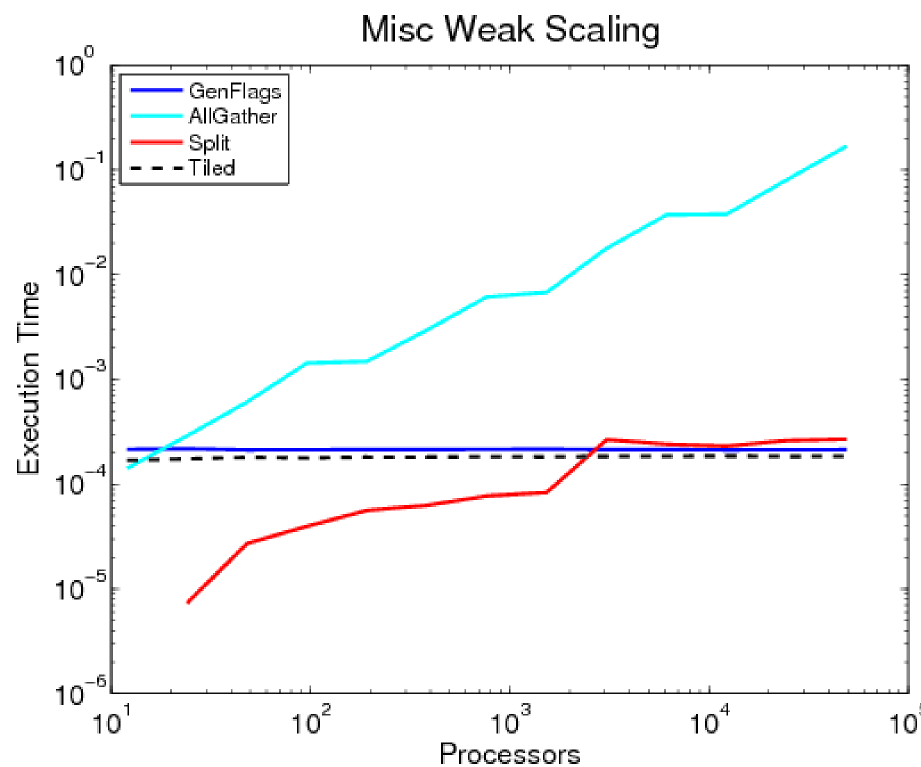




Strong Scaling – problem size fixed as number of cores increases should lead to decreasing execution time.

Weak Scaling of Algorithms and their Components



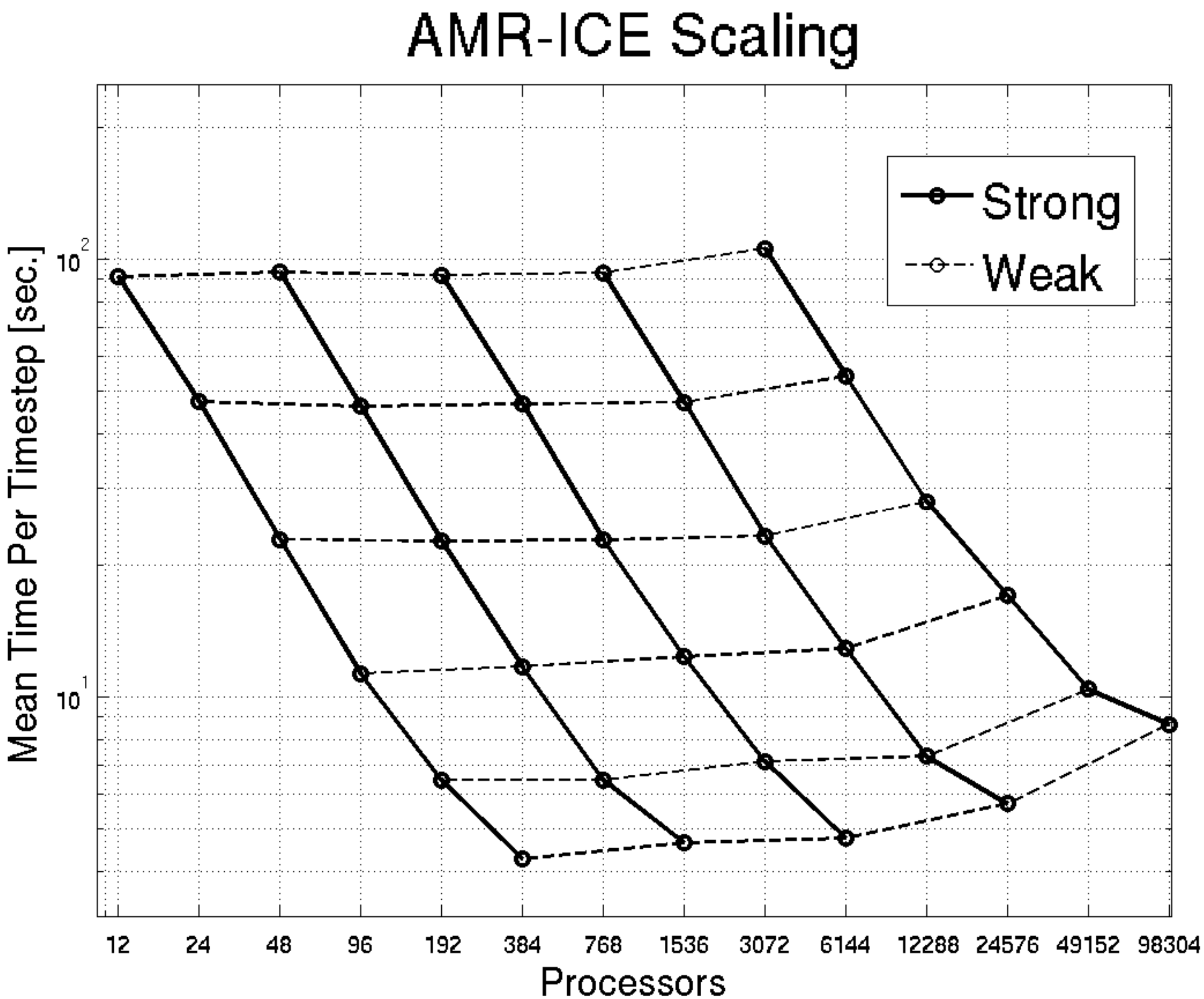


Weak Scaling – problem size per core fixed as number of cores increases should lead to constant execution time.

UINTAH SCALABILITY

**At 98k Proc
1 16x16x16
patch per
Core and so
Scalability fades**

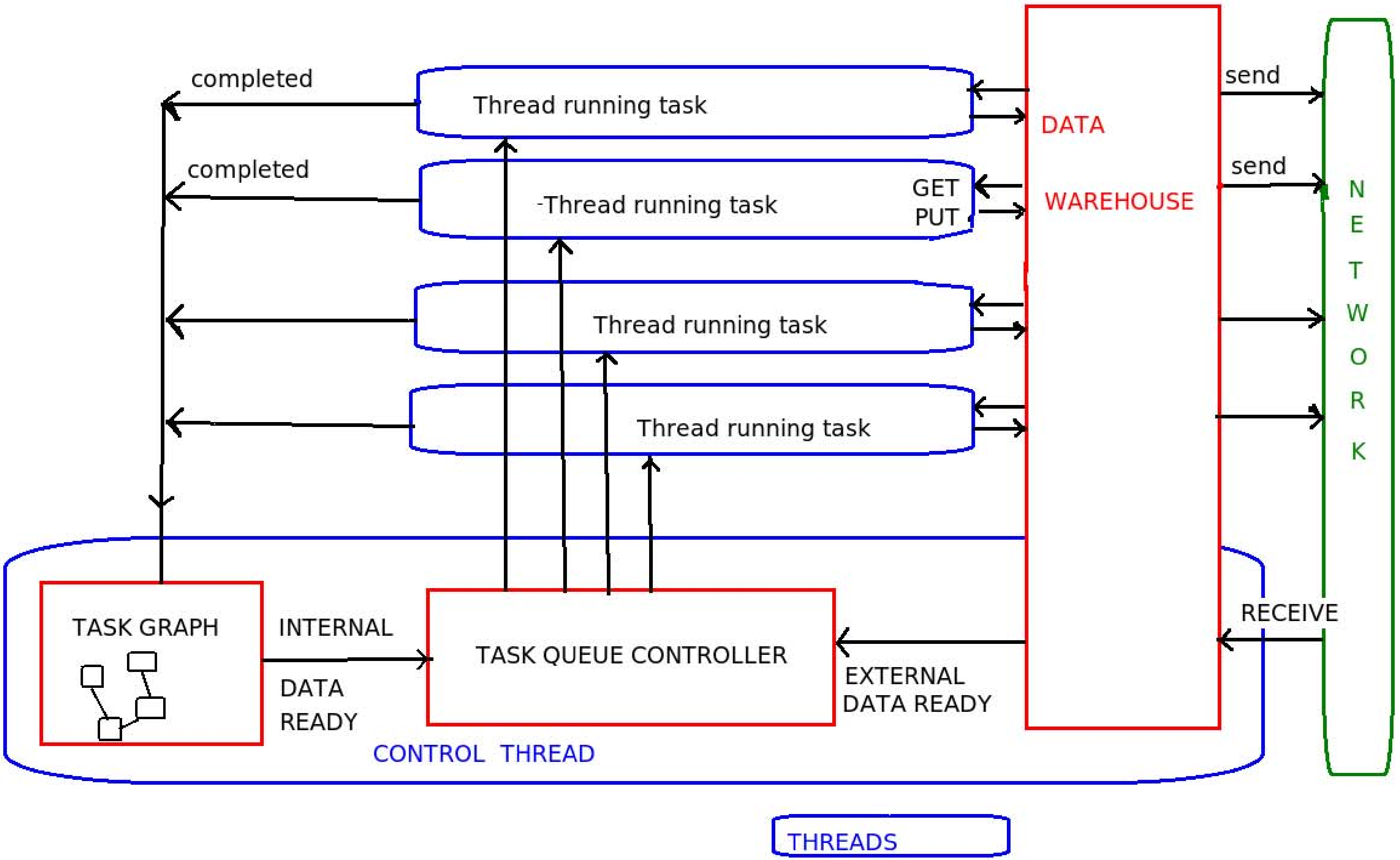
Problem is essentially an advected blob, but formulated as compress. Navier Stokes In 3D



NSF NICS Kraken 6-core AMD based machine

Uintah Hybrid MPI/Pthreads [TG11]

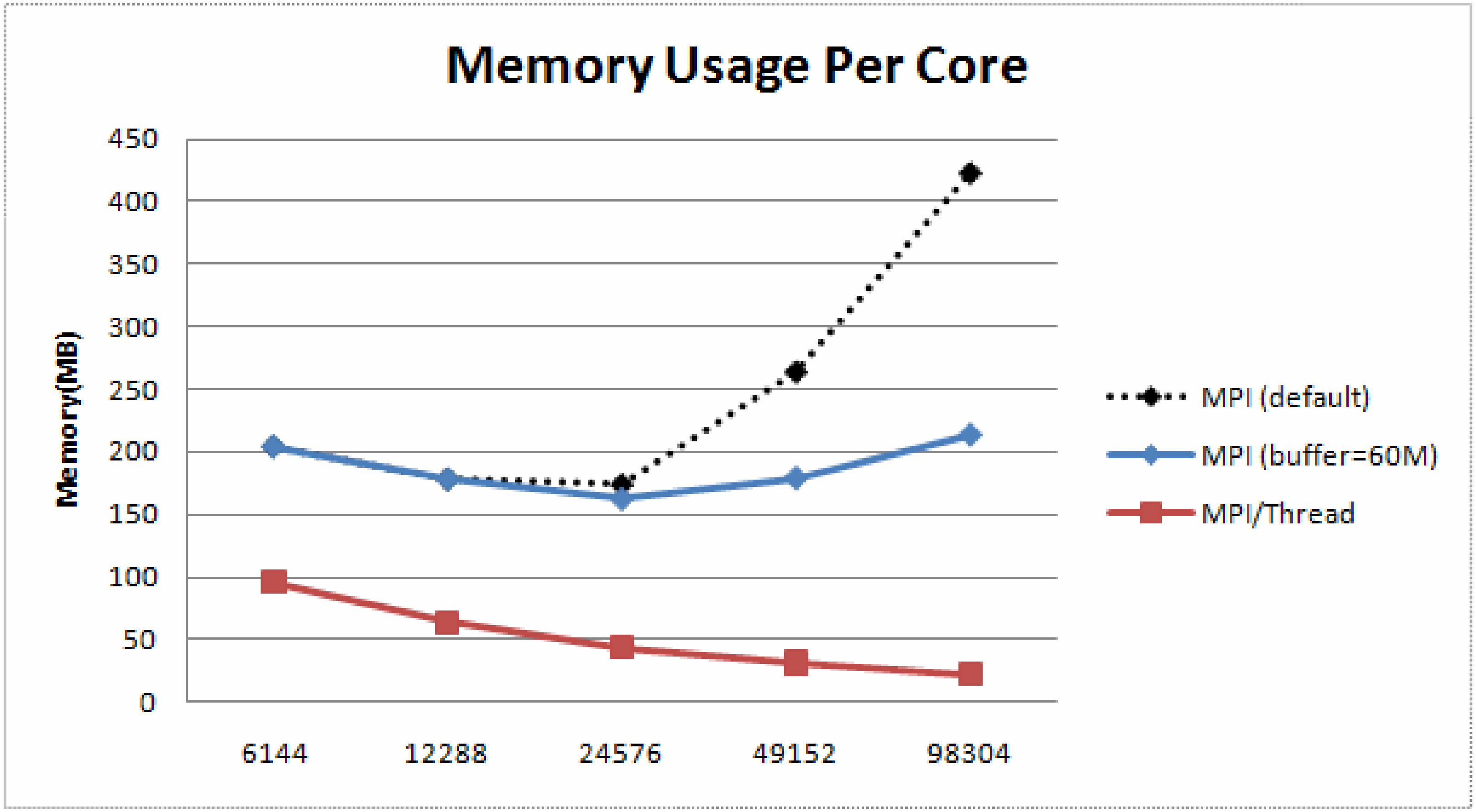
use only one copy of data warehouse per node as opposed to per MPI process



Uintah Memory Used Strong Scaling

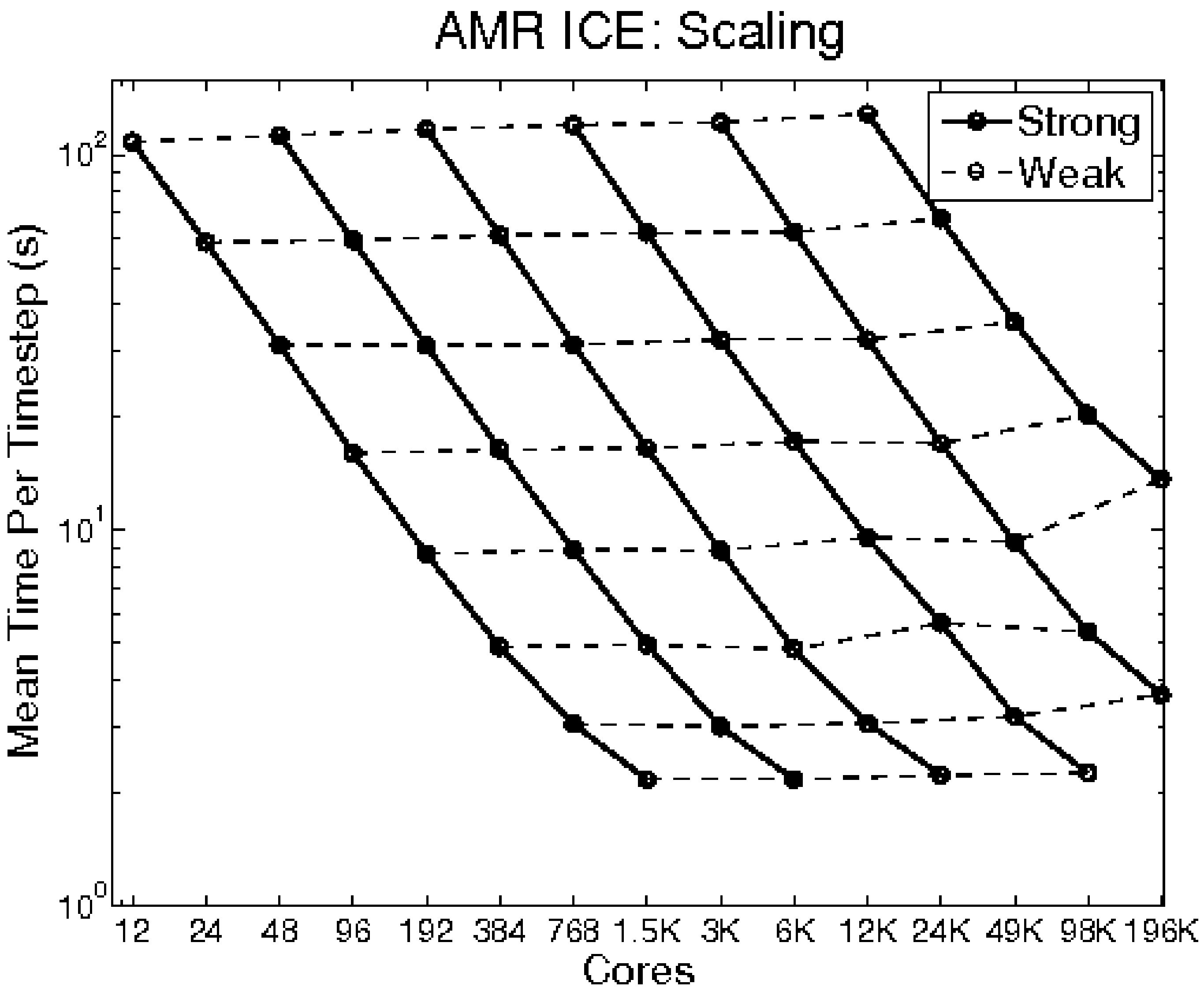
Can prove that global memory drops to (global memory)/(ncores)

Drop is more if Cray MPI buffers badly set….



Uintah now runs on 200K cores with only 10% of previous global memory per node and similar ~5% cpu times [TG11].

UINTAH SCALABILITY



Problem is essentially an advected blob, but formulated as compressible Navier Stokes Equations in 3D

DOE Jaguar 6-core AMD based machine

DEFLAGRATION

Wave moves at ~400m/s

540K value of T in corner

DETONATION

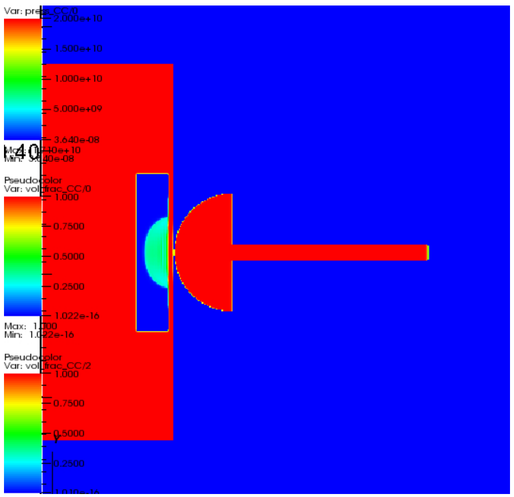
Wave moves 8500m/s

5Gpa P in corner

How do we go from this to this?

DDT: 3 Phase model

Stevens Test; data 72m/s
Simulation 65m/s



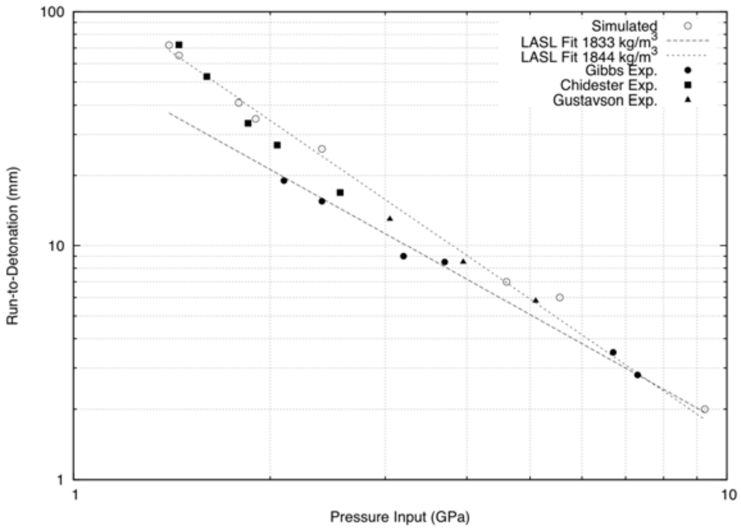
1 WSB two phase burning model

Two-phase thermal decomposition- reactants--MPM materials, products--ICE materials verified [Atwood]

2 Fast convective burning model

Impactor causes initial shock wave and hot spot/core collapse causing second shock which acts as a virtual piston which causes pressurization up to a critical threshold 5.0 GPa [Souers] activated by cracking [Berghout et al.] Pop plot shows correlation between LLNL experiments and computation.

Pop Plot

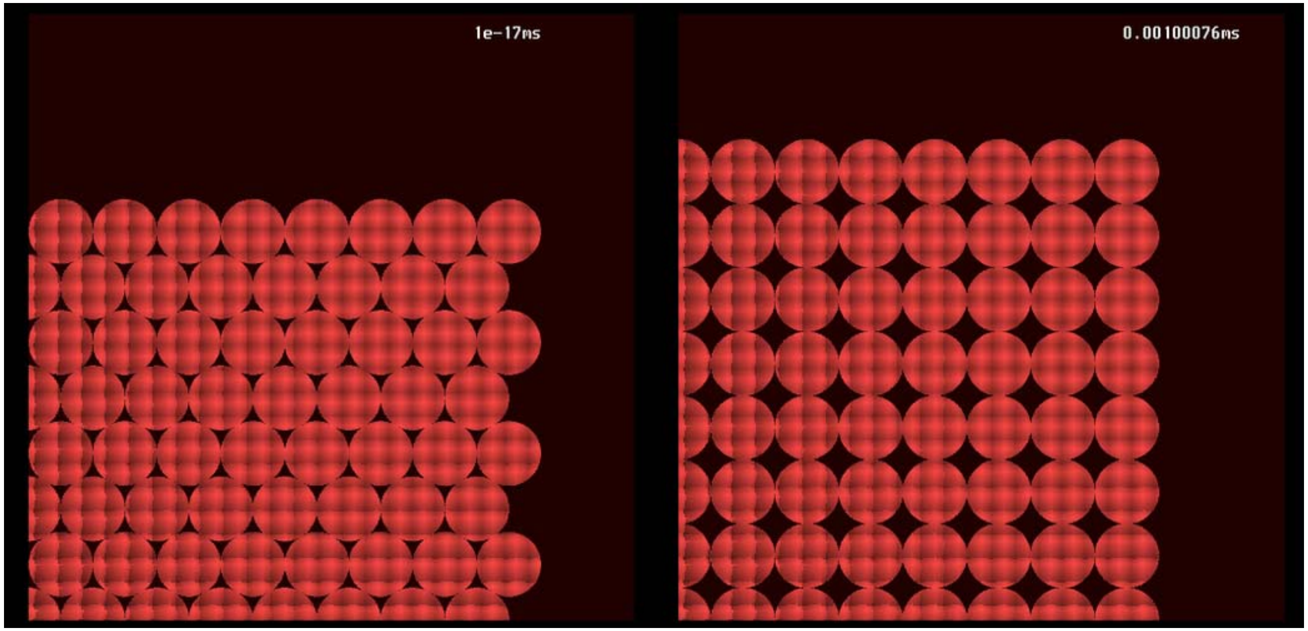


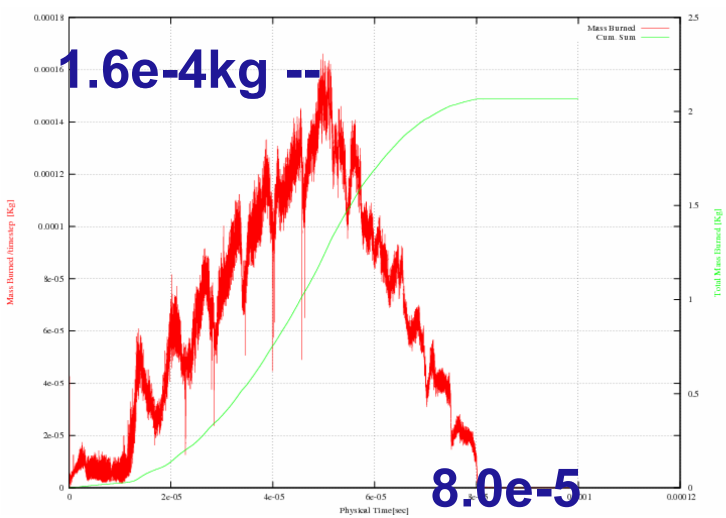
3 JWL++ detonation model

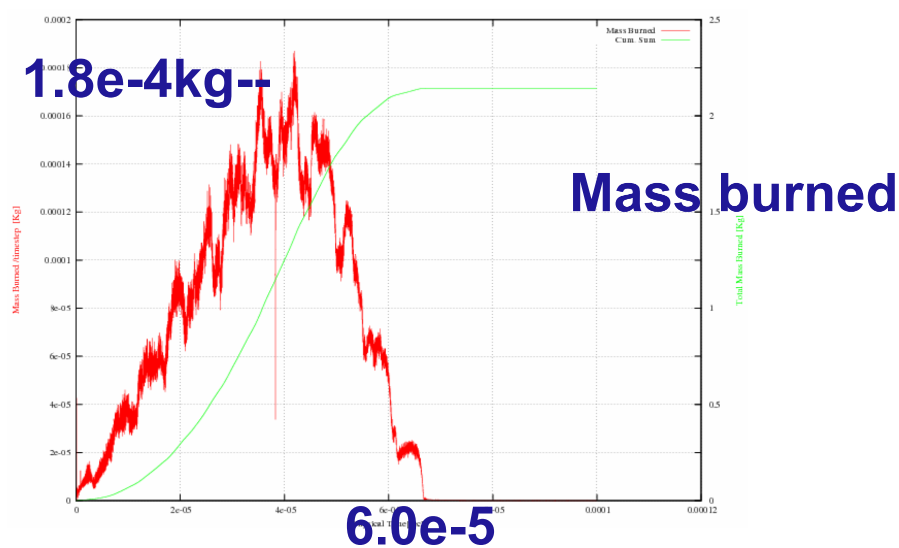
ρ=ρ0 × (P/K + (1 + (P/K)^2)½) ,K=11.4 GPa and ρ0=1840 kg/m3

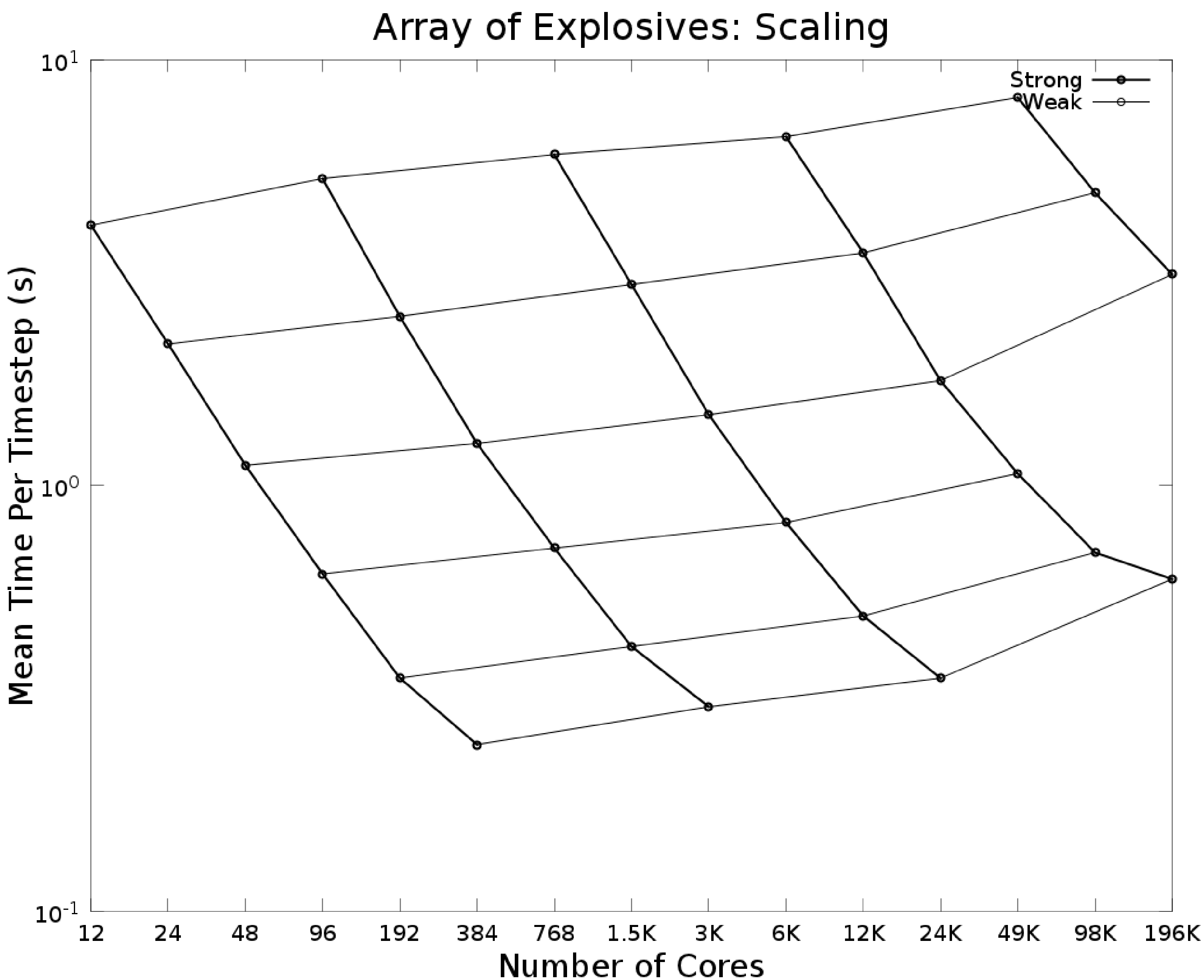
Gives good agreement with HMX explosive activation energy [German et al.]

Loose and Tightly packed Containers





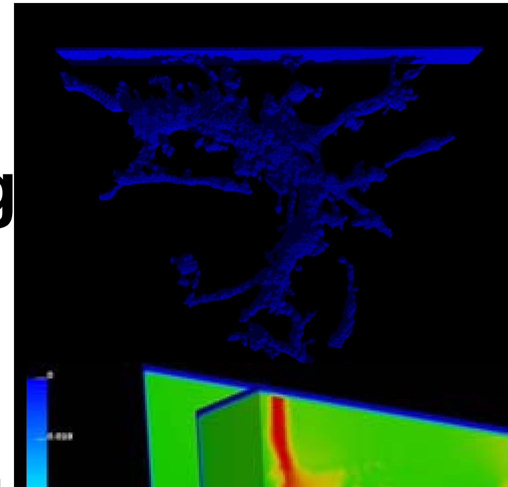


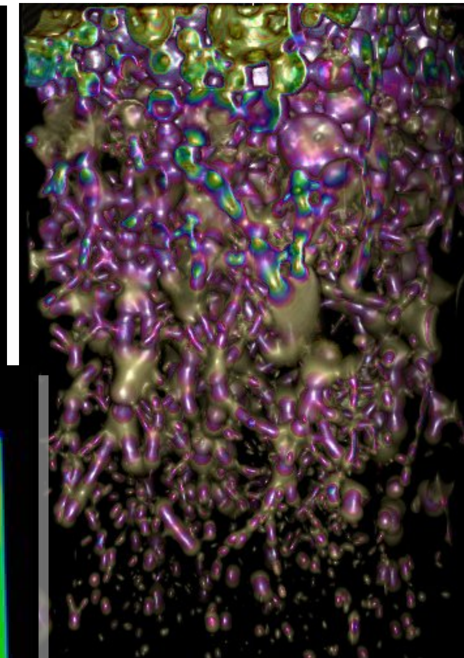


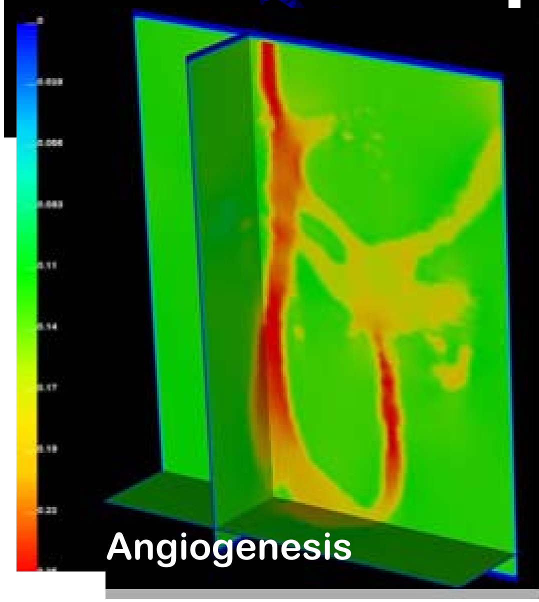
Five cases 64^3, 128^3, 256^3 512^3 1024^3 meshes

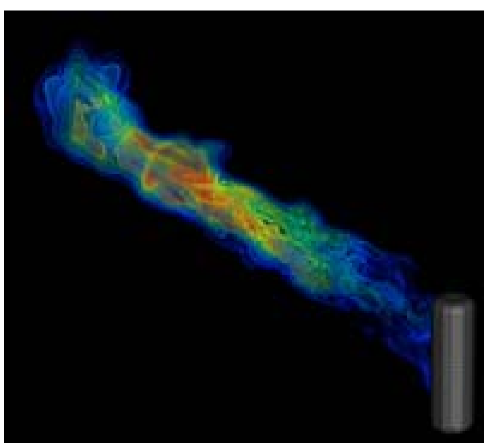
Uintah Applications

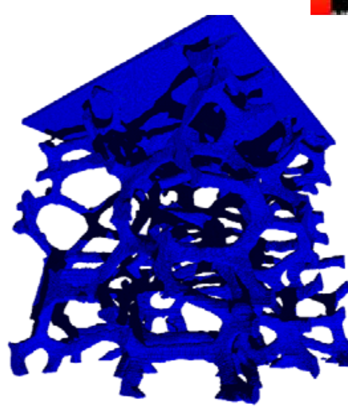
- Flare Simulation
- Angiogenesis
- Vocal chord modeling
- Rocket stage separation
- Heart injury modeling
- Foam properties
- Granular flow

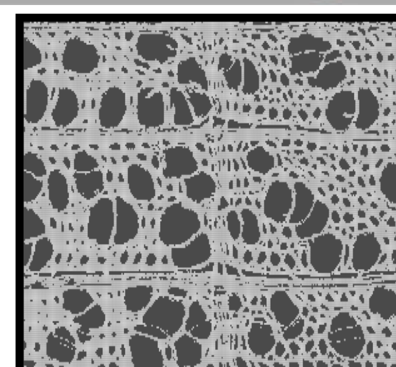






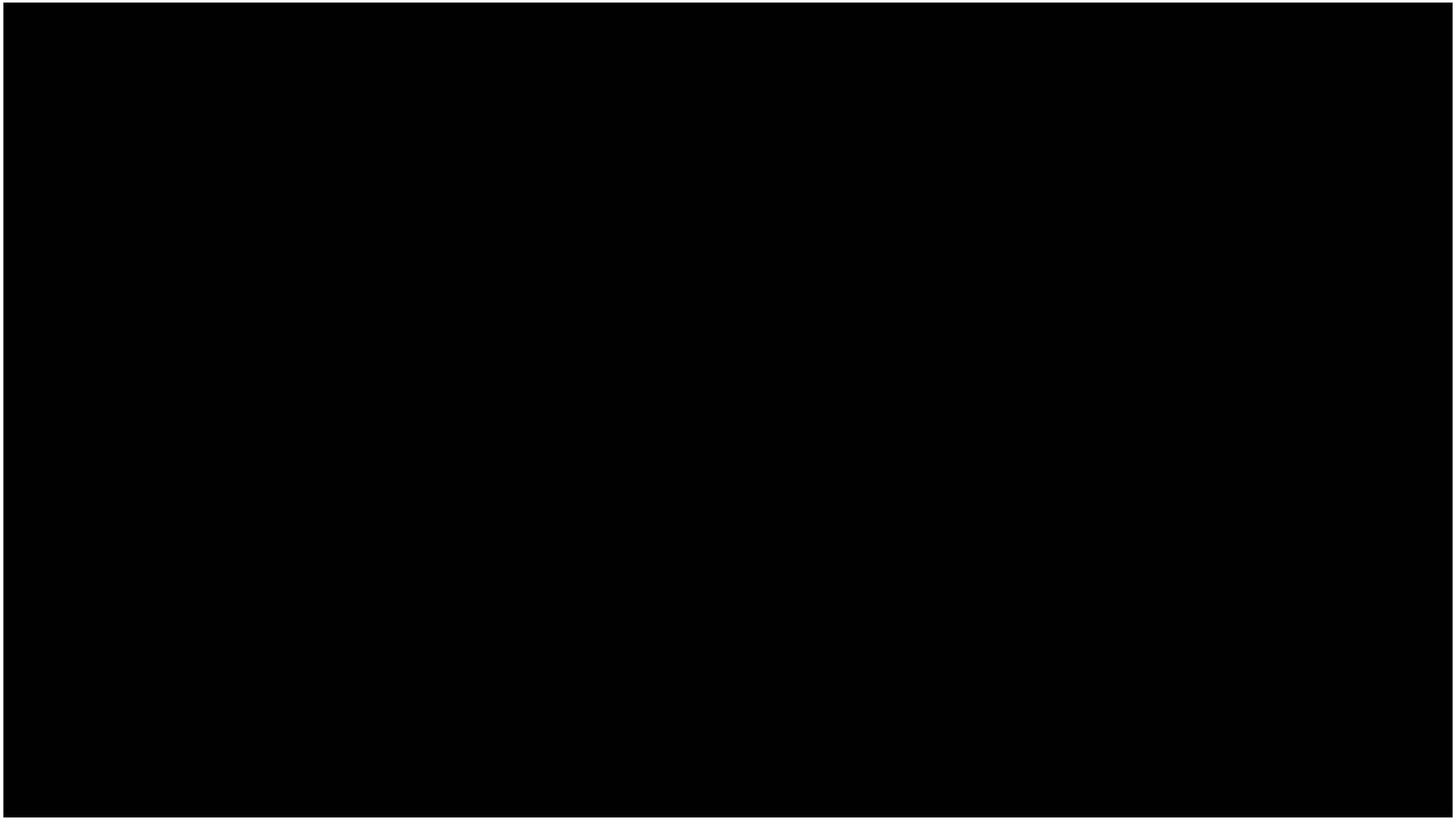






Shape Charge Simulation Guilkey Harman and Brannon

Metal casing copper liner deforms into penetrating hypersonic jet



The Algorithms will have to change

Consider stencil pde dx,dy,dz and dt with order p

Key metric is **<u>work per digits of accuracy in quantity of interest?</u>**

work = K p n_x n_y n_z n_t , K = stencil work constant

Pts in space and time

<u>Reducing the error by a factor of</u> f

Work$_{new}$ = **work**$_{old}$ $f^{\left(\frac{4}{p}\right)}$

If *p*=1 and *f* =2 first order, doubling accuracy requires 16x work

If *p*=4 and *f* =2 fourth order, doubling accuracy requires 2x work

The Algorithms will have to change

High order methods are widely used in some areas but…

Solution of hard problems sometimes suggests that they offer fewer advantages—

Non-oscillatory approximation of a front with order p requires (p+1) points in the front.

Mesh refinement needs to be combined with high-order methods

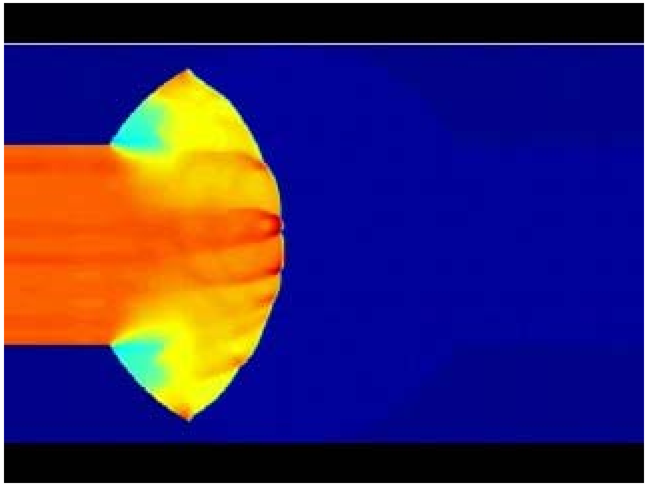
Error in quantity of interest requires adjoint-based remeshing methods not just gradient-based

Scalable Linear algebra solvers for exascale are a daunting challenge but one that is being tackled….

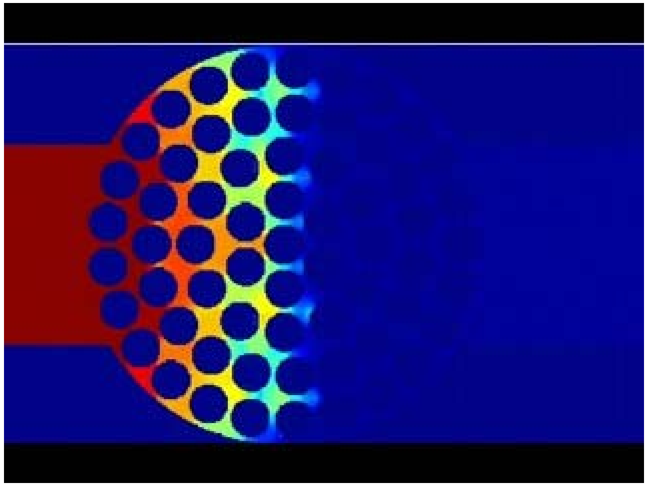
Summary

- Uintah version of silver model works
- Silver model is not a silver bullet
- UIntah runs on 196K cores in AMR and fixed grid modes
- Scalability aspects of Uintah investigated
- Work needed to move to full DDT model
- Need to start work on detonation reduction approaches

Detonation Diffracting into cylindrical volume



Detonation disrupted by cylindrical rods



Source DDT experts