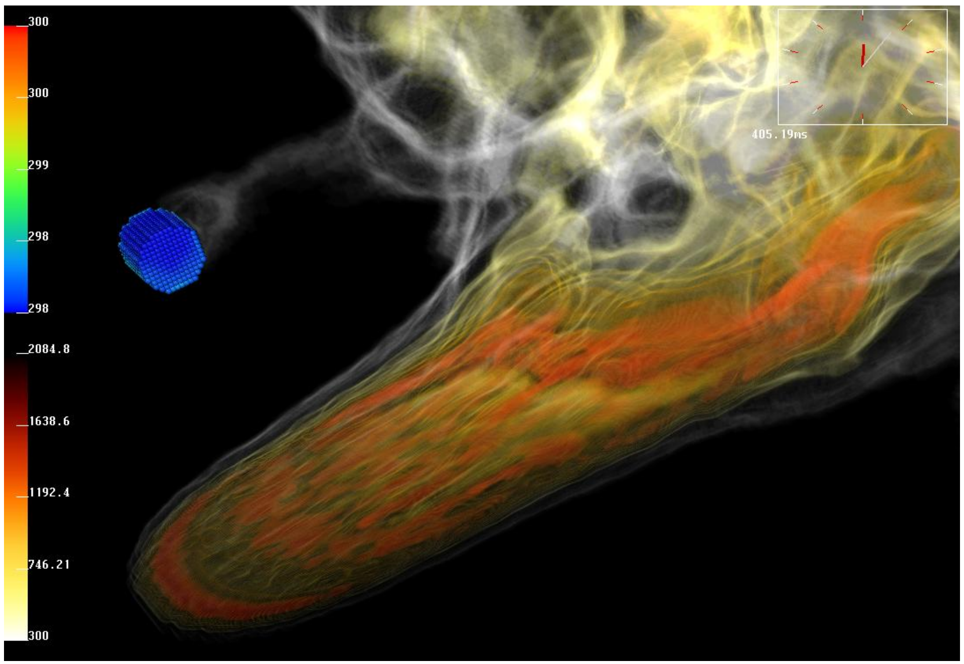
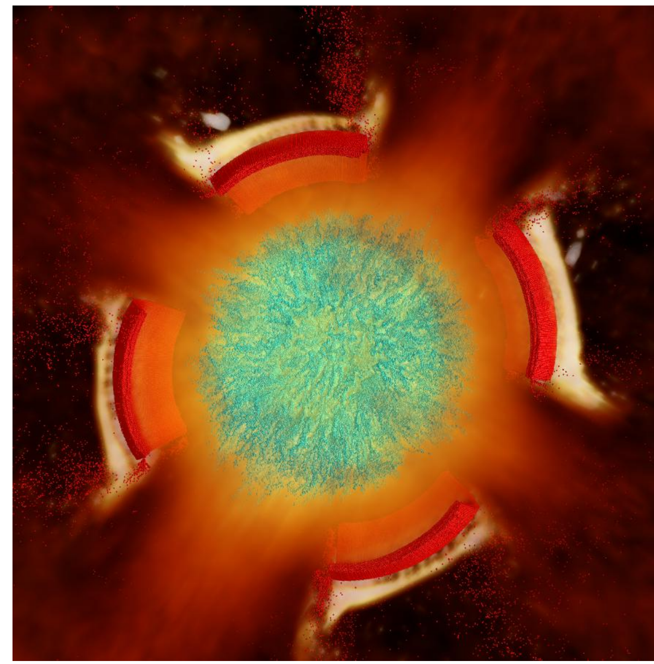
Cost Estimation Algorithms for Dynamic Load Balancing of AMR Simulations





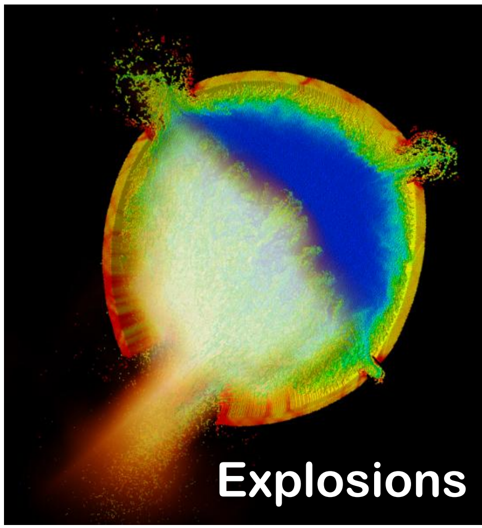
Justin Luitjens, Qingyu Meng, Martin Berzins, John Schmidt, et al.

Thanks to DOE for funding since 1997, NSF since 2008, TACC, NICS

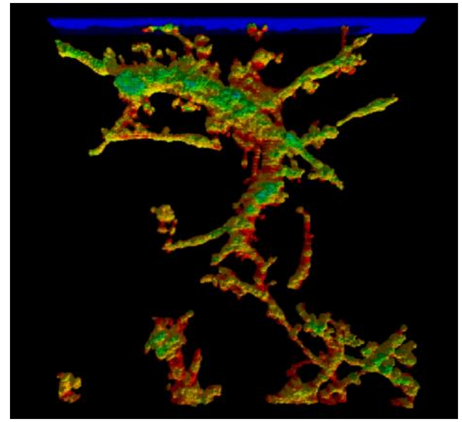
Uintah Parallel Computing Framework

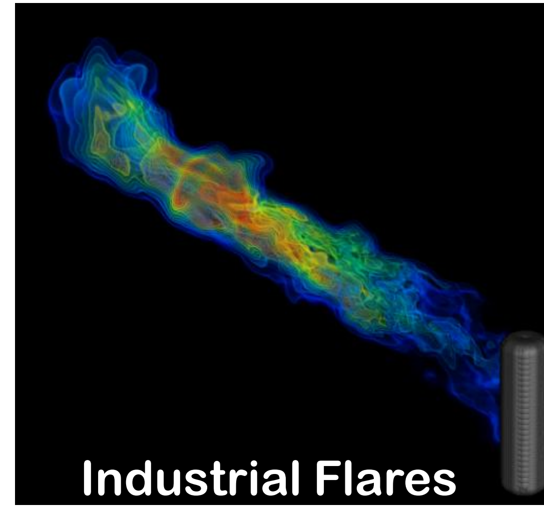
- Uintah - **far-sighted design by <u>Steve Parker</u>** :
  - **Automated parallelism**
    - **Engineer only writes “serial” code for a hexahedral patch**
    - **Complete separation of user code and parallelism**
    - **Asynchronous communication, message coalescing**
  - **Multiple Simulation Components**
    - **ICE, MPM, Arches, MPMICE, et al.**
  - **Supports AMR with a ICE and MPMICE**
  - **Automated load balancing & regridding**
  - **Simulation of a broad class of fluid-structure interaction problems**

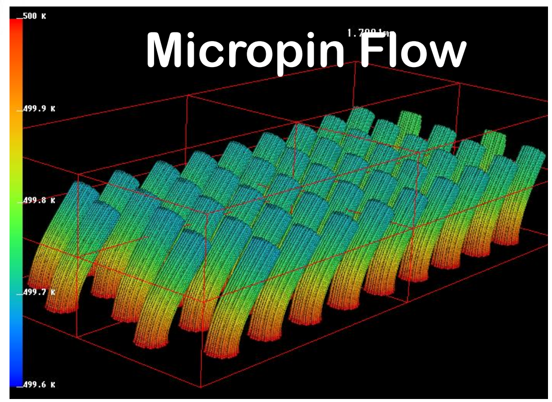
Uintah Applications

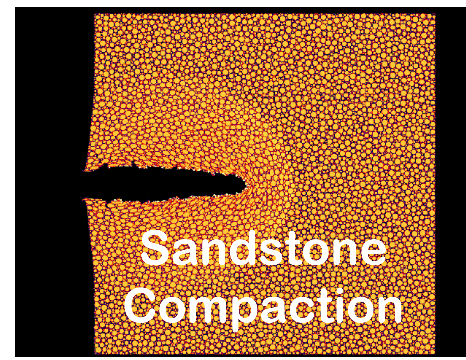

Explosions

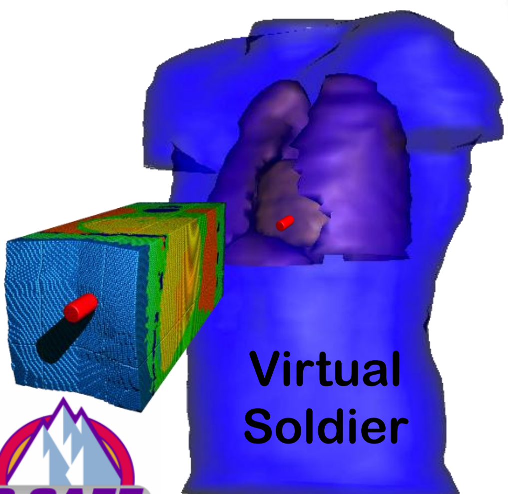
Plume Fires

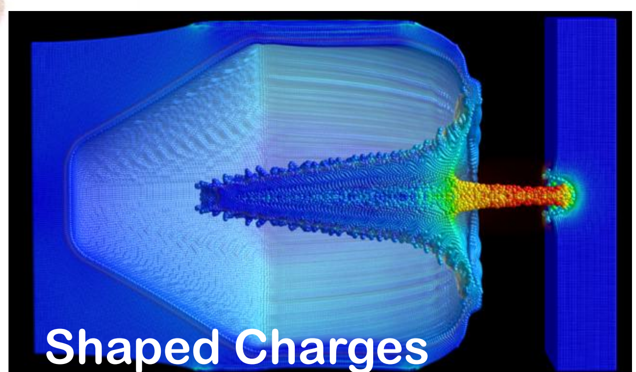


Angiogenesis

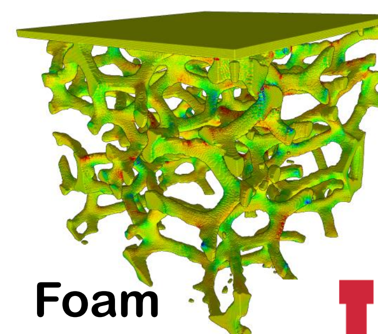

Industrial Flares


Micropin Flow

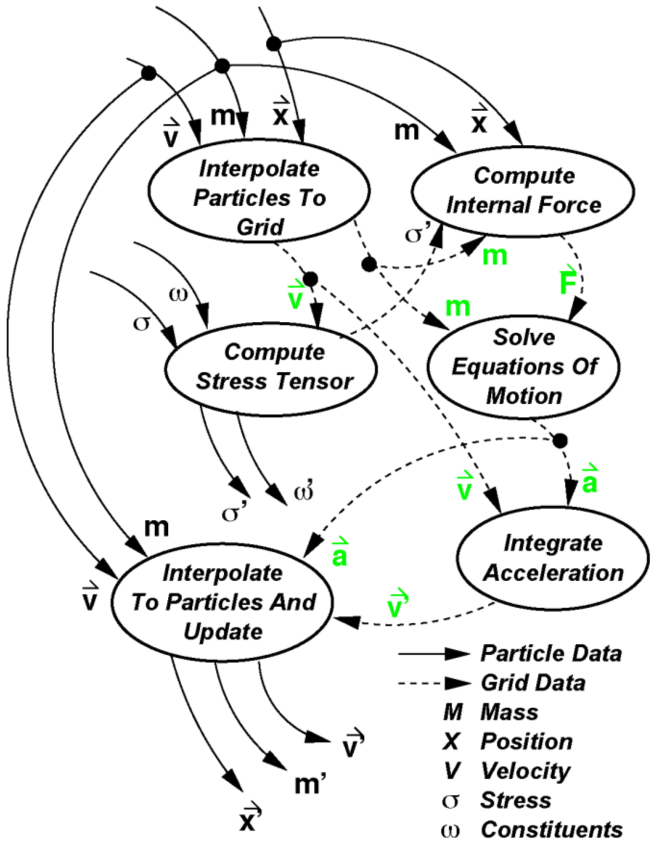

Sandstone Compaction

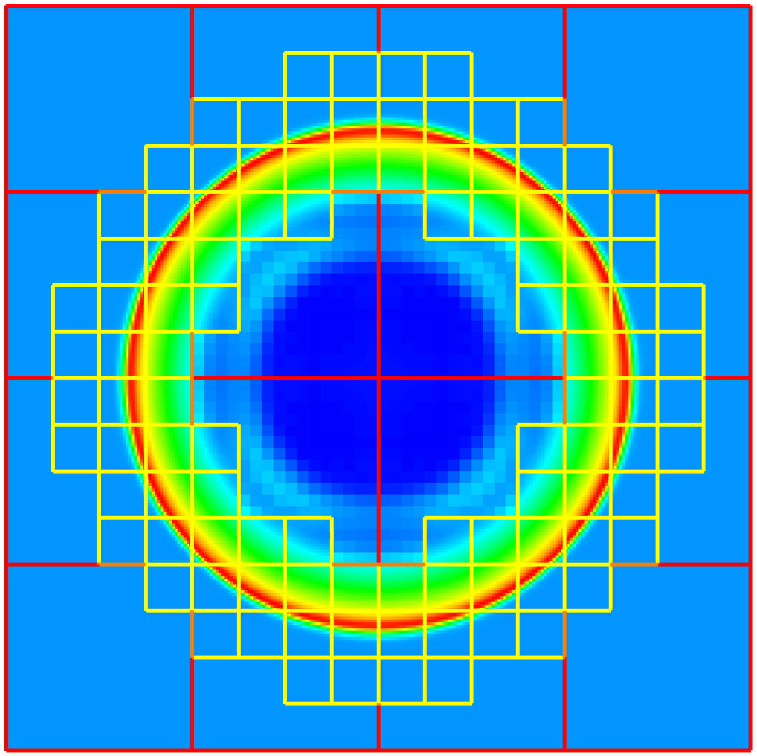

Virtual Soldier


Shaped Charges


Foam Compaction

How Does Uintah Work?



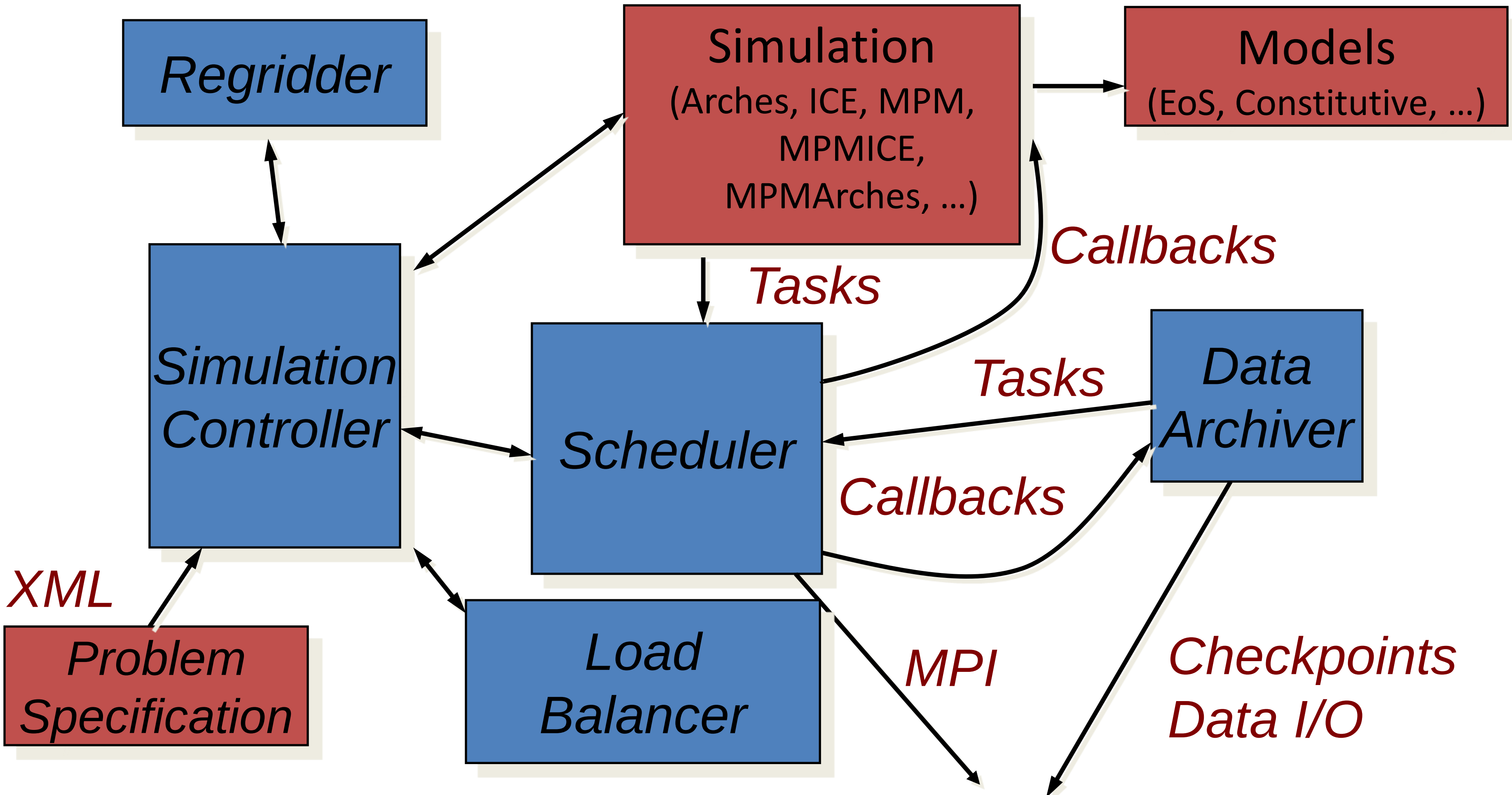


Task-Graph Specification

- Computes & Requries

Patch-Based Domain Decomposition

How Does Uintah Work?



Domain Expert

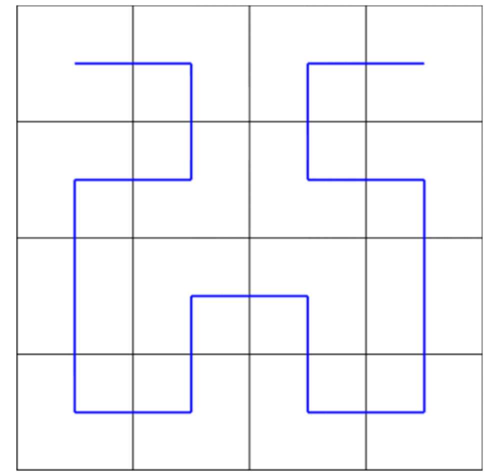
Tuning Expert

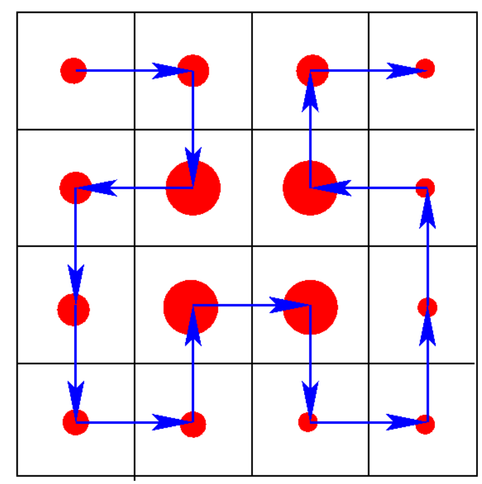
Legacy Issues

- Uintah is 12+ years old
- How do we scale to today's largest machines?
  - Identify and understand bottlenecks
    - TAU, hand profiling, complexity analysis
    - Reduce O(P) Dependencies
      - Look at memory footprint?
  - Redesigned components for O(100K) processors
    - Regridding, Load Balancing, Scheduling, etc

Uintah Load Balancing

- Assign Patches to Processors
  - Minimize Load Imbalance
  - Minimize Communication
  - Run Quickly in Parallel
- Uintah Default: Space-Filling Curves
- Support for Zoltan





In order to assign work evenly we must know how much work a patch requires

Cost Estimation: Performance Models

$E_{r,t}$: Estimated Time

G_r: Number of Grid Cells

P_r: Number of Particles

$$E_{r,t} = c_1 G_r + c_2 P_r + c_3$$

c_1, c_2, c_3 : Model Constants

- **Need to be proportionally accurate**
- **Vary with simulation component, sub models, compiler, material, physical state, etc.**

Can estimate constants using least squares at runtime

$$\begin{bmatrix} G_0 & P_0 & 1 \\ \dots & \dots & \dots \\ G_n & P_n & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} O_{0,t} \\ \dots \\ O_{n,t} \end{bmatrix}$$

$O_{r,t}$: Observed Time

What if the constants are not constant?

C-SAFE UNIVERSITY OF UTAH

THE UNIVERSITY OF UTAH

Cost Estimation: Fading Memory Filter

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

$$E_{r,t+1} = \alpha \, O_{r,t} + (1 - \alpha) \, E_{r,t}$$

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

- No model necessary
- Can track changing phenomena
- May react to system noise
- Also known as:
  - Simple Exponential Smoothing
  - Exponential Weighted Average

Compute per patch

C-SAFE UNIVERSITY OF UTAH

THE UNIVERSITY OF UTAH

Cost Estimation: Kalman Filter, 0th Order

$E_{r,t}$: Estimated Time $O_{r,t}$: Observed Time

Update Equation: $E_{r,t+1} = K_{r,t}\,(O_{r,t} - E_{r,t}) + E_{r,t}$

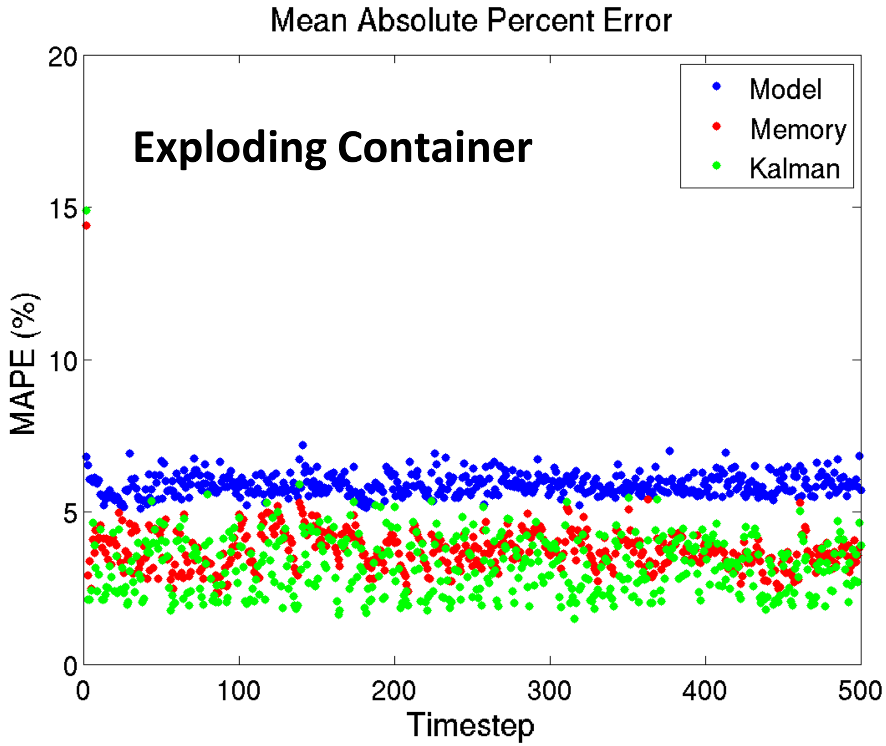
Gain: $K_{r,t} = M_{r,t} / (M_{r,t} + \sigma^2)$

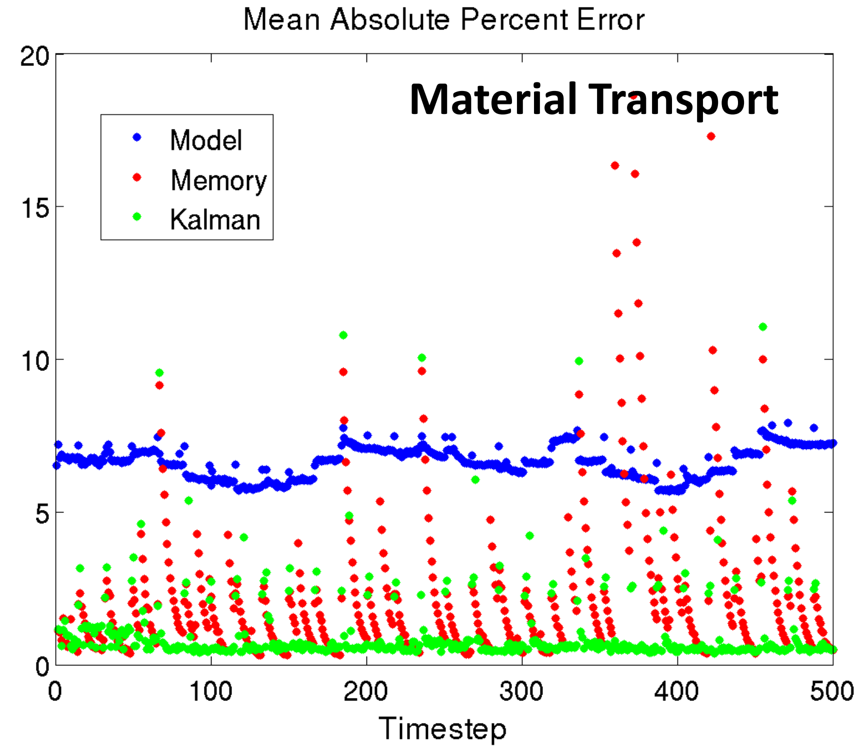
a priori cov: $M_{r,t} = P_{r,t-1} + \phi$

a posteri cov: $P_{r,t} = (1 - K_{r,t})\,M_{r,t}$ $P_0 = \infty$

- Accounts for uncertainty in the measurement: σ^2
- Accounts for uncertainty in the model: ϕ
- No model necessary
- Can track changing phenomena
- May react to system noise
- **Faster convergence than fading memory filter**

Cost Estimation Comparison





- Filters provide best estimate
- Filters spike when regridding

	Ex. Cont.	M. Trans.
Model LS	6.08	6.63
Memory	3.95	2.64
Kalman	3.44	1.21

C-SAFE University of Utah

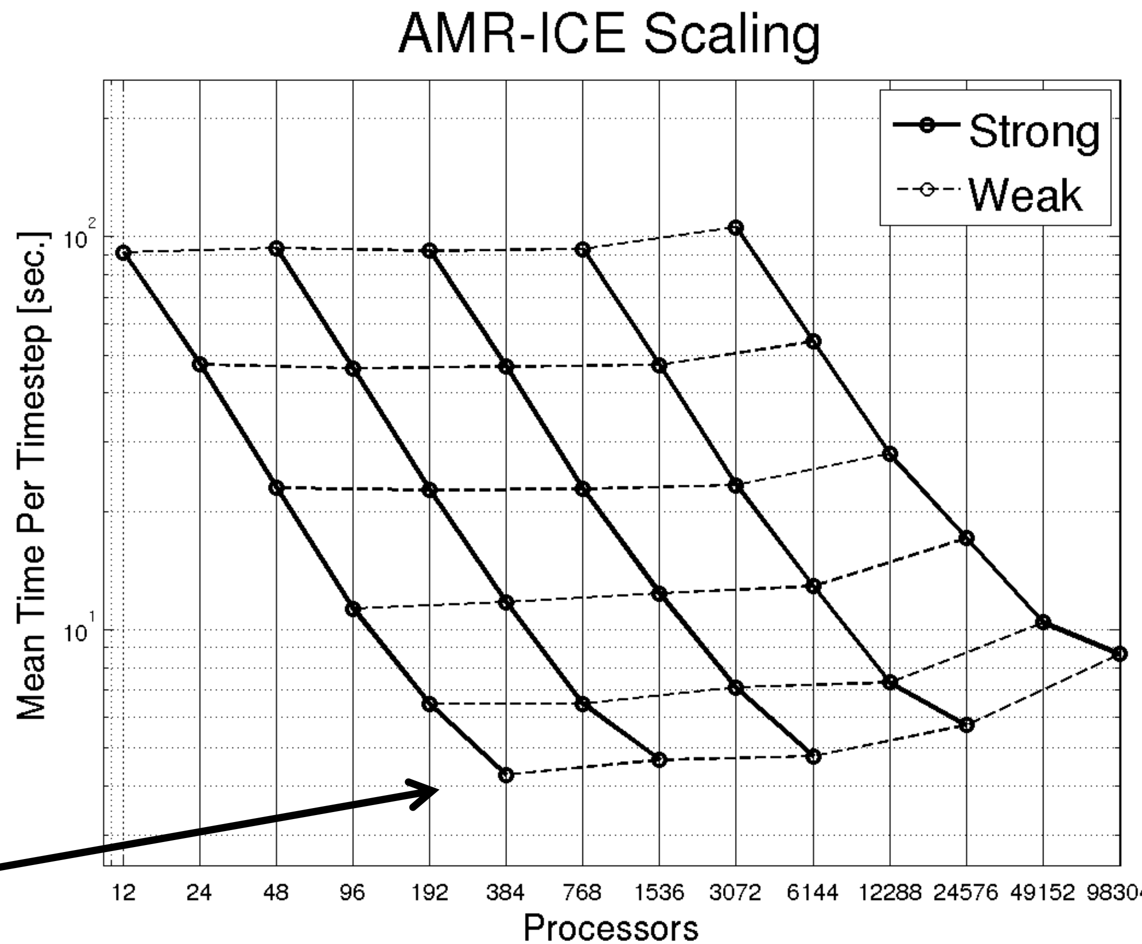
The University of Utah

AMR ICE Scalability

Highly Scalable AMR Framework

Even with small problem sizes

One 8^3 patch per processor



Problem: Compressible Navier-Stokes

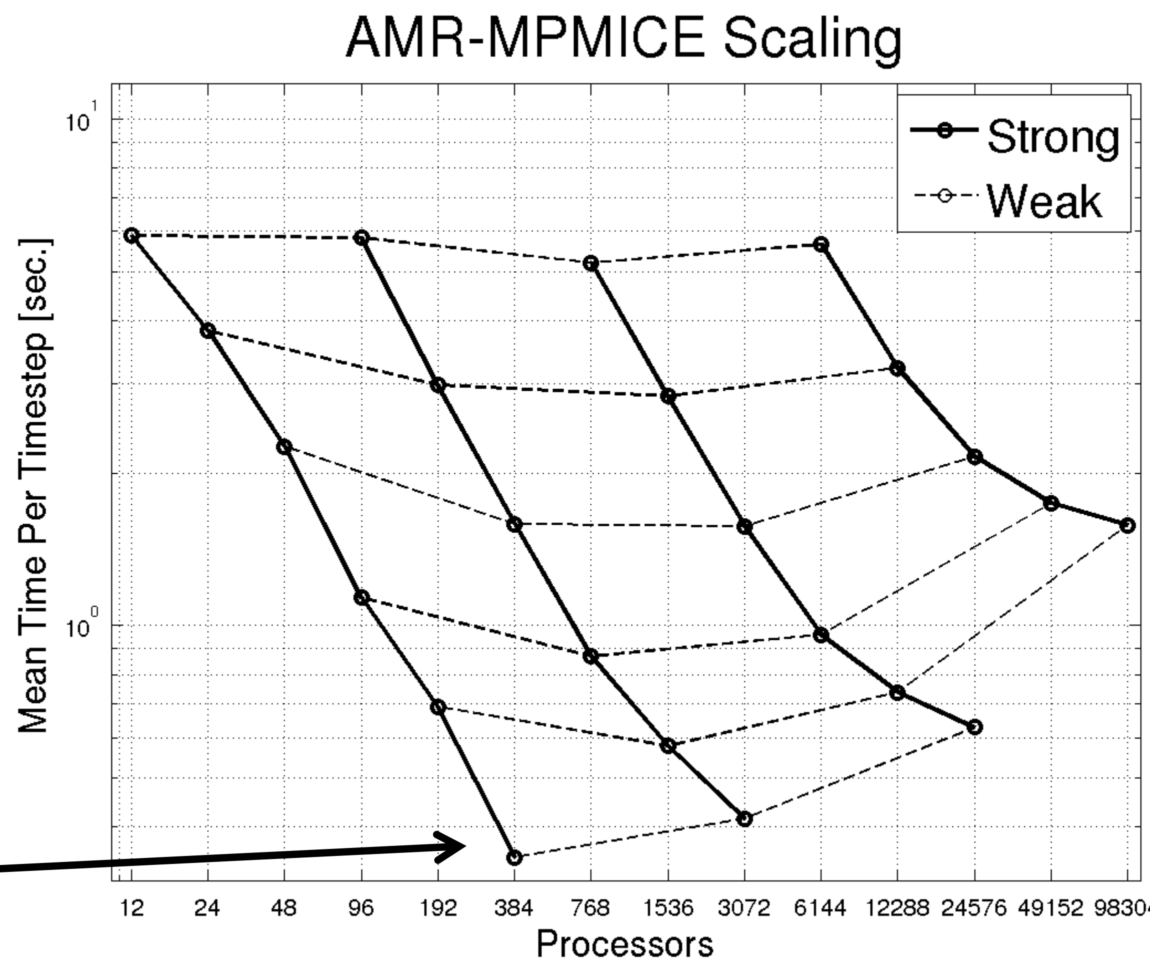
C-SAFE University of Utah

The University of Utah

AMR MPMICE Scalability

Decent MPMICE scaling

More work is needed



One 8^3 patch per processor

Problem: Exploding Container

Conclusions

- The complexity and range of applications within Uintah require an adaptable load balancer
- Profiling provides a good method to predict costs without burdening the user
- Large-Scale AMR requires that all portions of the algorithm scale well
- Through lots of work AMR within Uintah now scales to 100K processors
- A lot more work is needed to scale to O(200K-300K) processors

Questions?