

# ALE3D and BlueGene-L: Early Plans & Ideas

Presented at BGL Workshop

Reno NV

10/13-10/14 2003

Rob Neely

DCOM / B-Div / LLNL

ALE3D Project

Lawrence Livermore National Laboratory, PO Box 808, Livermore CA 94551



This work performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

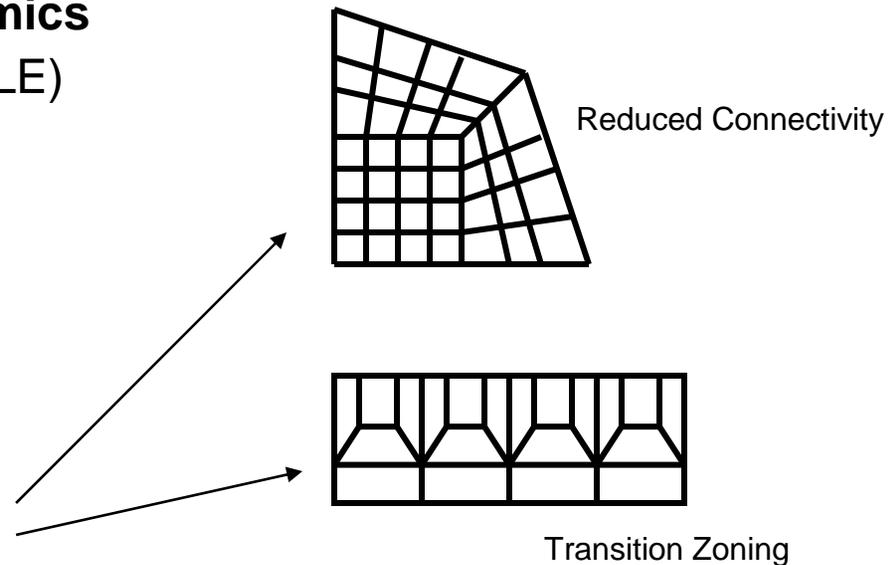


- Overview of ALE3D
- Applications we have in mind for BG/L
- Anticipated Challenges
- Questions

# Overview of ALE3D



- **ALE3D is a large, multi-physics, finite element, application**
- **Explicit and Implicit Hydrodynamics**
  - Arbitrary Lagrange-Eulerian (ALE)
  - Slide Surfaces (Contact)
  - Advanced Material Models
  - Thermal Transport
  - Chemical Reactions
  - Incompressible Flow
  - etc...
- **Unstructured hexahedral mesh**
- **Mixed material elements**
  - Material interfaces are tracked independent of mesh boundaries



## ALE3D Overview (con't)



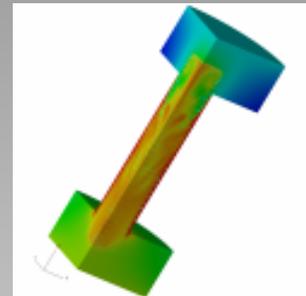
- Mixture of C / C++ / F77/ F90 (75/15/5/5%)
- Originally developed on vector architectures (late 80's - 90's)
- Ported to MPP architectures 1995-1998
- Scaled to several thousand processors with many of the physics packages
- “Production code” with ~10 active developers and dozens of users’
- Uses a separate generator and main code
  - Generator creates a 0 time restart file, given:
    - mesh geometry
    - input file (materials + physics)
    - domain decomposition
    - shape generation
  - Main code always starts from a restart file

# We have several applications in mind for BGL right now



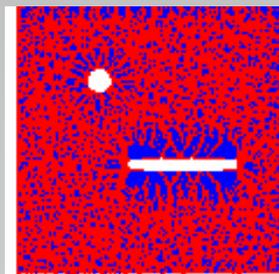
- All require a huge amount of resources
- All are primarily limited by memory requirements and resource availability
- All can use (physics) models which are currently either:
  - Implemented in the code
  - In the plan to be implemented in same time frame
  - Easy to add
- All are important to “the program”

HE (High Explosive)  
Grain Scale  
Modelling

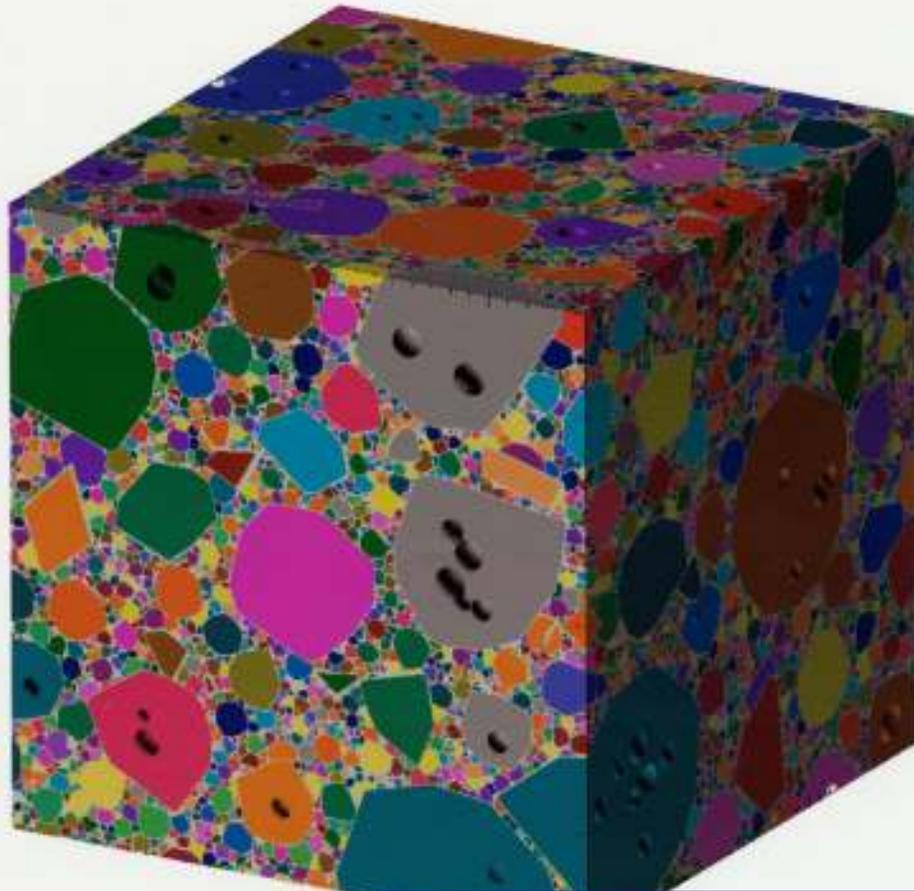


Thermal  
Cookoff  
of High  
Explosive

Fracture  
modeling and  
advanced  
material modeling



## HE Grain Scale Modeling Has Been the Focus of Prior ASCI Milestones



- 216,000,000 elements
- 100,000 crystals
- 10,000,000 planes used to define crystals
- 1075 input files containing shape geometry planes
- 50 ASCII white nodes (800 p)
- 2 GB memory per processor
- 1600 domains
- 3.5 hours to generate

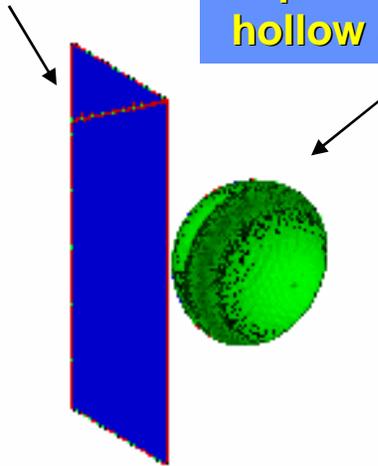
Although not the largest problem we've run, this Level 2 milestone calculation pushed code limits in several different areas, spawning new scaling features that will benefit future calculations

# HE Grain Scale Modeling - background



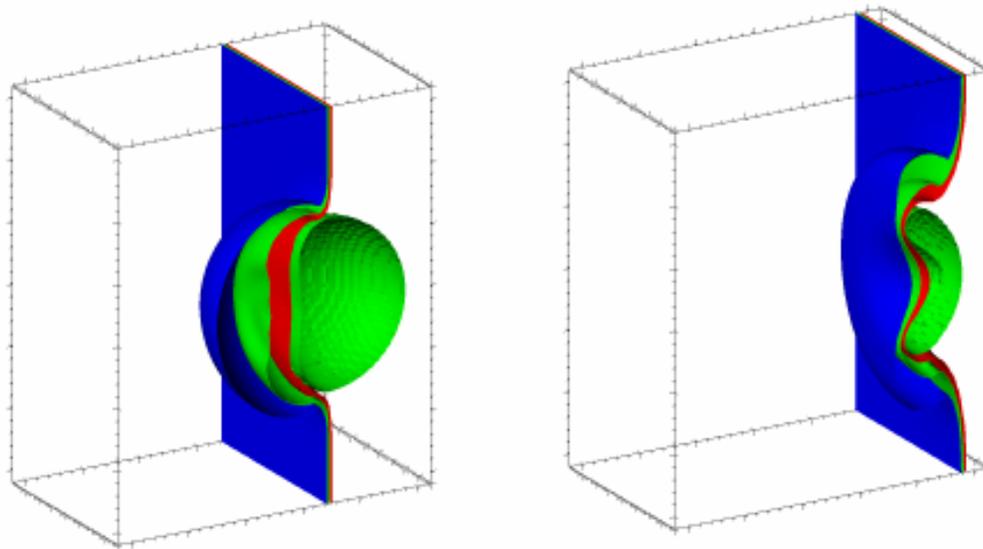
10 GPa shock

14  $\mu\text{m}$  diameter hollow sphere

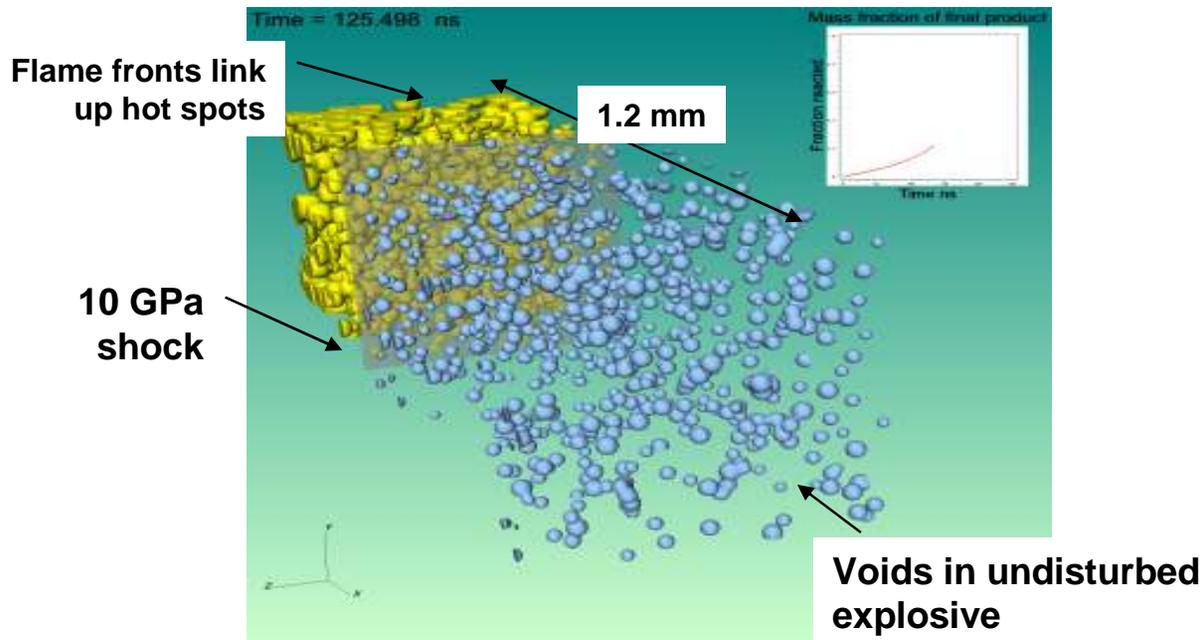


- We have performed analyses of the shock ignition of hollow or filled crystal imperfections
- If the shock is sharp relative to the imperfection, the cavity collapses asymmetrically

For strong shocks and weak (low strength/viscosity) solids, a jet forms, which impacts the opposite wall. This hot spot is an ignition site.



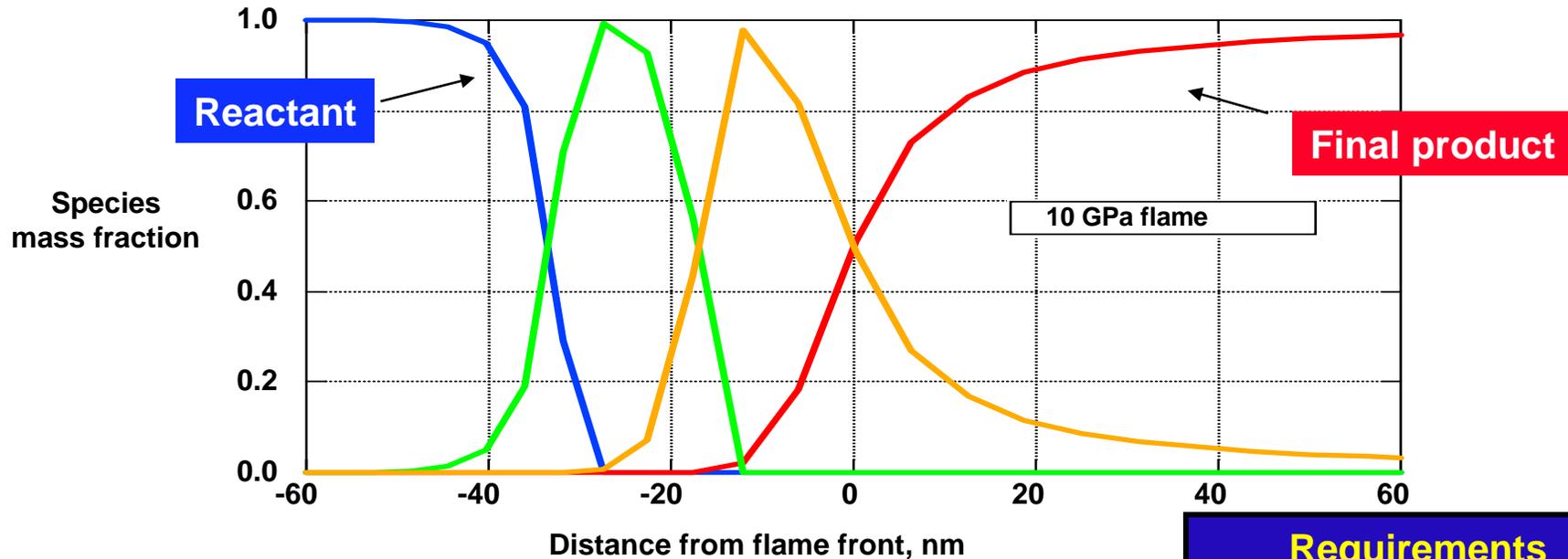
# PBX Studies



Requirements	
# Elements	216 M
Mem / elem	3.5 Kb
Elem / proc	50 k
# Procs	5000
# cycles	25,000
time per cycle	15s
Time to sol'n	100 hours

- We propose studies of PBX that are subject to lower input pressure
  - Physical extent is larger to examine the longer run-up distance to detonation
  - Past simulations were one-dimensional. We propose additional 3-dimensional simulations to examine edge effects

# Direct simulation of high pressure flames requires 4 nm mesh size



- We are proposing simulations of the collapse of a 3  $\mu\text{m}$  cavity that resolve the flame front
  - One isolated and two interacting cavities
- Currently approximated using level sets
- Coupled hydrodynamics, heat transfer and chemistry

Requirements	
# Elements	~ 3 Billion
Mem / elem	2.5 Kb
Elem / proc	40,000
# Procs	65,000
# cycles	~ 200,000
time per cycle	~ 15 s
Time to sol'n	~ 35 days

# Implicit hydrodynamic simulations of the densification of HE grains is proposed

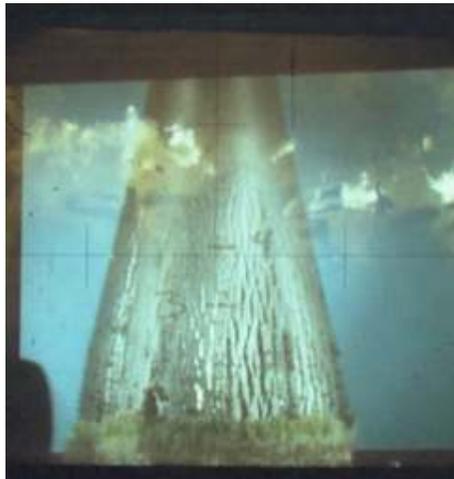


- Each color represents a different crystal orientation
- Impose boundary conditions to slowly crush the grain
- The densification is accompanied by crystal flow and fracture

Requirements	
# Elements	200 M
Mem / elem	8 kb
Elem / proc	20,000
# Procs	10,000
# cycles	600
time per cycle	300 s
TTS	~ 50 hours



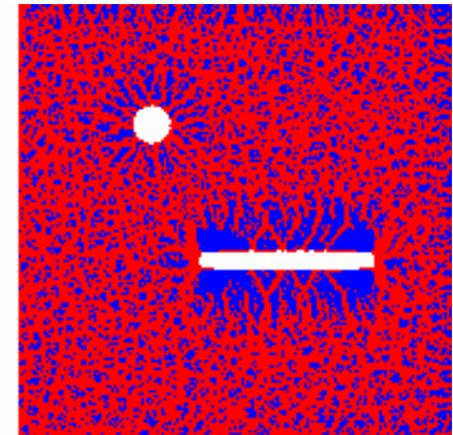
# Simulations of explosively driven fracture require a fine spatial discretization to capture fragmentation



Pipe bomb at 25 $\mu$ s

Fragmentation is triggered by spatially inhomogeneous material properties which are captured statistically by the model

Simulations have demonstrated spatial convergence of fracture patterns and fragment size.



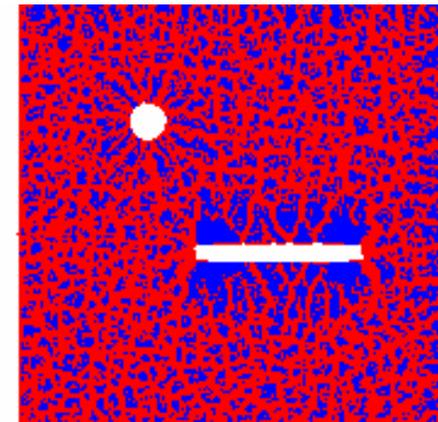
600 X 600 elements



Fine scale fracture at < 1 mm requires element sizes to be 100  $\mu$ m or less.

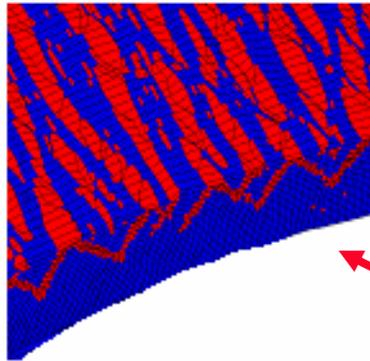


Through-thickness shear fracture

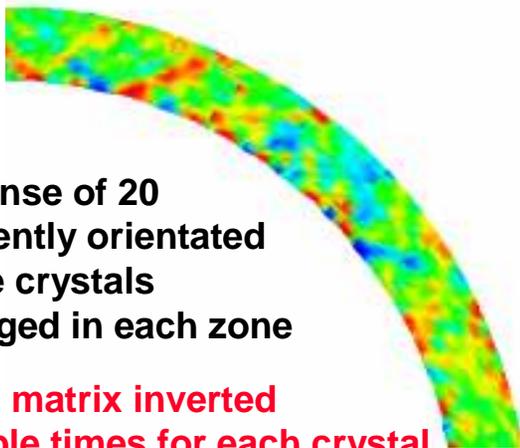


300 X 300 elements

# Computational requirements are further affected by model sophistication needed for shear fracture



Element aspect ratios should be kept near 1:1 to capture 45° shear fracture



response of 20 differently orientated single crystals averaged in each zone

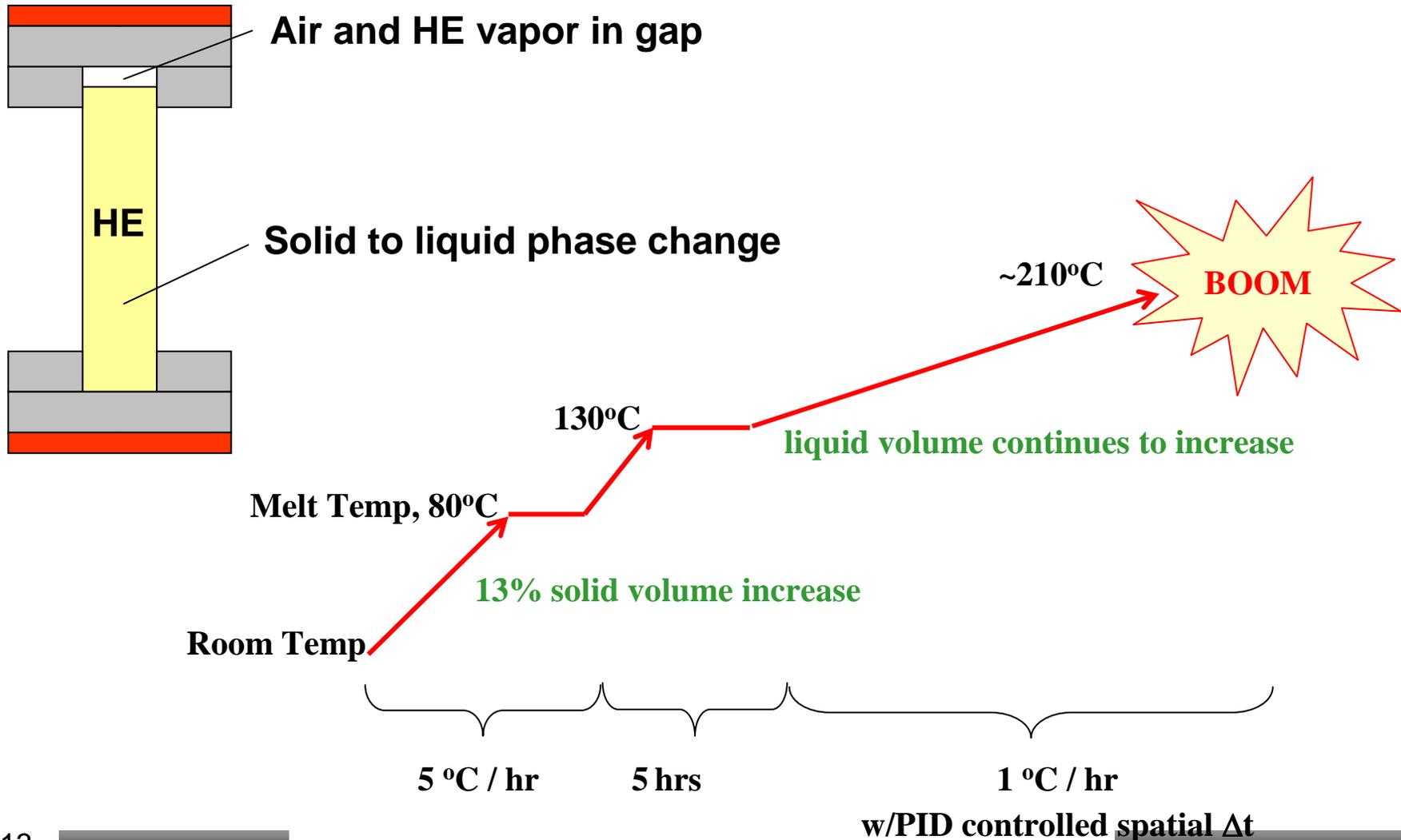
12x12 matrix inverted multiple times for each crystal

Requirements	
# Elements	500 M
Mem / elem	10 kb
Elem / proc	8 k
# Procs	62,500
# cycles	30,000
time per cycle	120 s
TTS	~ 40 days

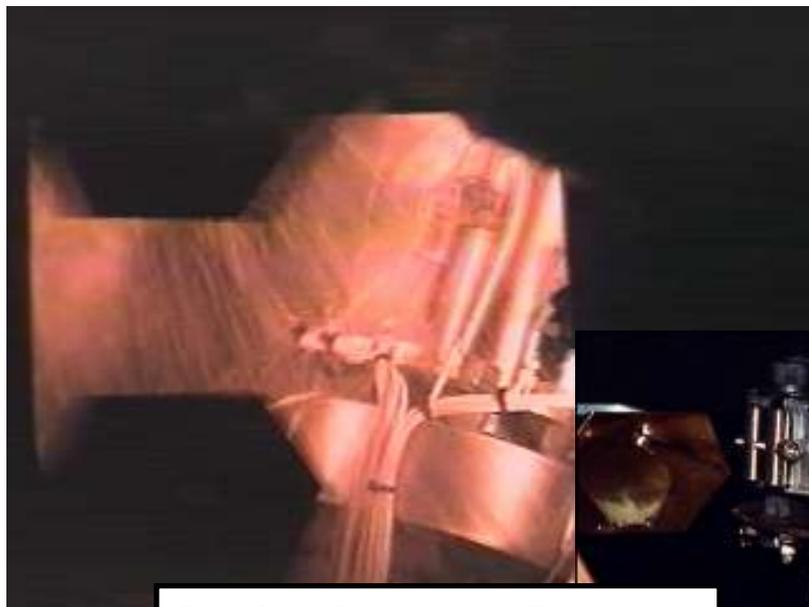
Approximately 20% of the elements in this problem will be in the pipe material. The above estimates are naïve in that they don't take into account load balance issues properly

Material models based on crystal plasticity capture shear localization prior to fracture

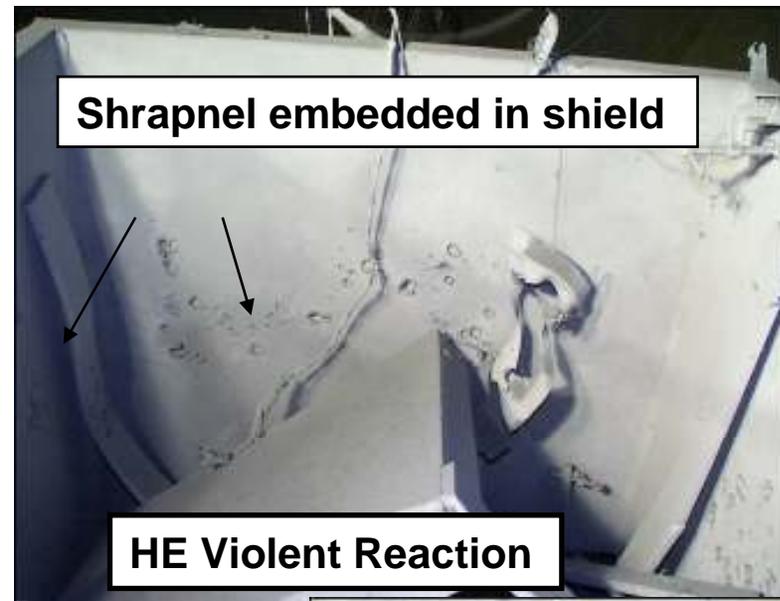
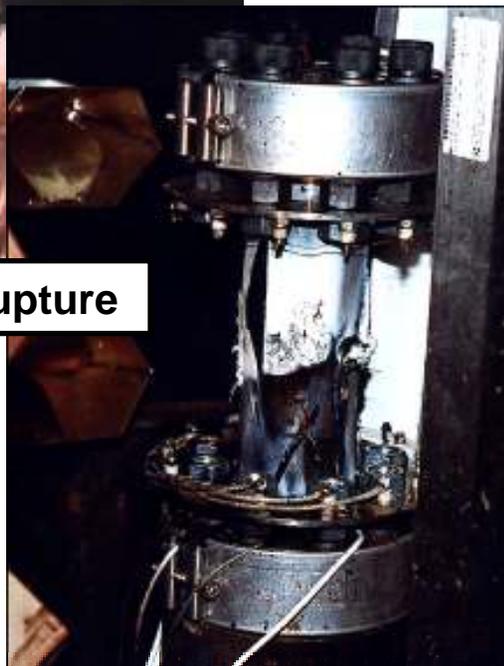
**Cookoff is a slow heating process so that long computational times are required.**



# Cookoffs Can Result in a Variety of Responses



**Benign Pressure Rupture**



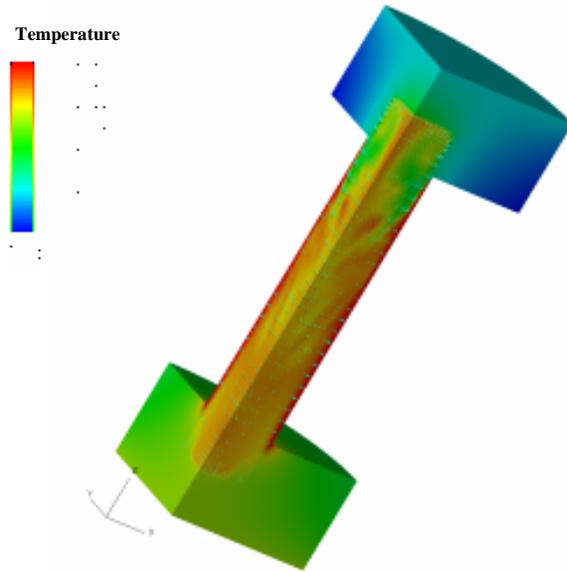
**Shrapnel embedded in shield**

**HE Violent Reaction**

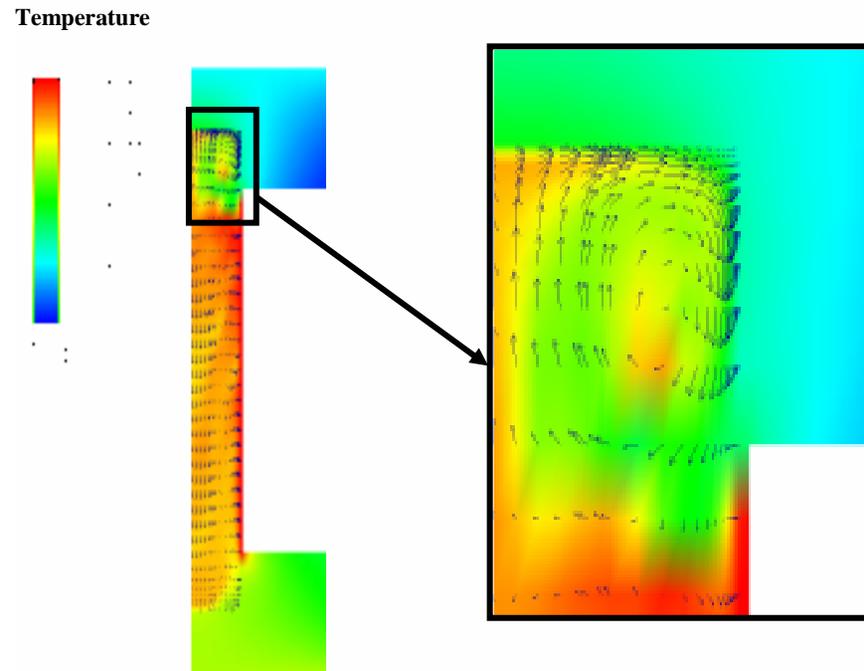
**2-inch hole punched in inch-thick end plate**



**ALE3D can simulate the thermal convection of liquid TNT for cookoff. The flow exhibits strong recirculating flow regions**



**The thermal gradient causes heated liquid TNT to travel up the sides of the tube and back down the middle.**

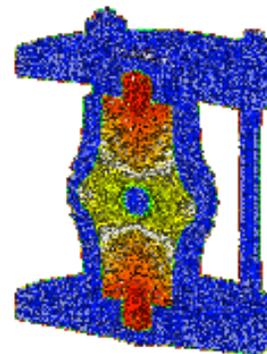
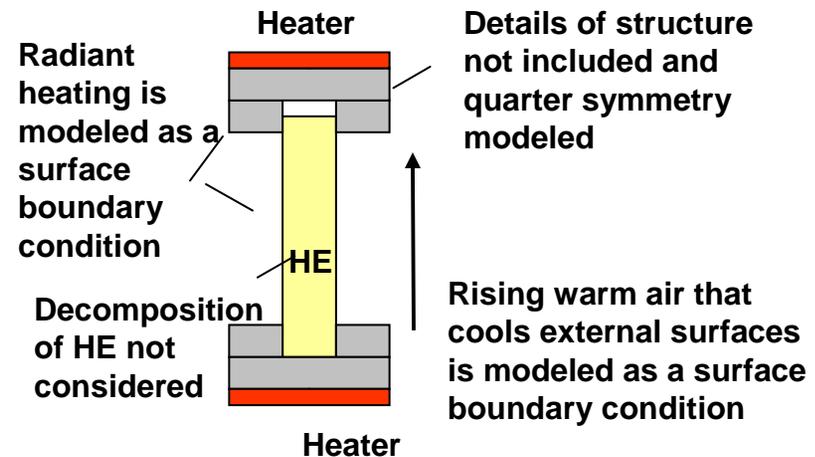


**2D slice and snapshot in time of 3D transient simulation**

# Cookoff simulations are currently run underresolved in 2D



- Cookoff problems are initially run with implicit hydro and thermal to model slow heating
- At time of explosion, code switches to explicit hydro
- Deflagration model is used to study explosive properties
- Longer term, we hope to apply advanced fracture models to examine fragmentation of the case



Requirements	
# Elements	2 B
Mem / elem	3.5 kb
Elem / proc	50 k
# Procs	40,000
# cycles	1000 / 5000
time per cycle	300 / 5
TTS	100 hours

# Challenges to Scaling



- **Any approach we take will have to be evolutionary, not revolutionary**
  - Code architecture
  - Manpower resources
  - Other programmatic demands ...
- **Specialized optimizations are *typically* frowned upon**
  - Exceptions will be made, depending on overall effect on maintainability of the source code
- **No clear roadblocks currently exist. Yet.**
- **We have overcome scaling hurdles in the past**
  - Most were predictable, but also time consuming to solve
- **BG/L will be a good experiment to show if current SPMD algorithms can scale**

## Challenges – compile and link (porting)



- **Assuming a reasonable set of C/C++/F77 compilers, we don't anticipate (m)any porting problems**
  - xIC/xlf
  - However, any time we port to a new platform, it is a time consuming endeavor, primarily due to 3<sup>rd</sup> party libs
- **Absolute minimum required to run:**
  - Silo (I/O)
- **Required if we want to generate the problem on BGL**
  - SGEOS, LEOS, Parmetis, HDF5
- **Necessary to run implicit hydro, thermal, inc flow**
  - FEI, Hypre, Prometheus, LAPACK, ccode
- **Necessary if we want to build the entire code**
  - Petsc, xerces\*, FETI

\* within the year, xerces will be also be an absolute requirement

# Challenges – 32-bit ints

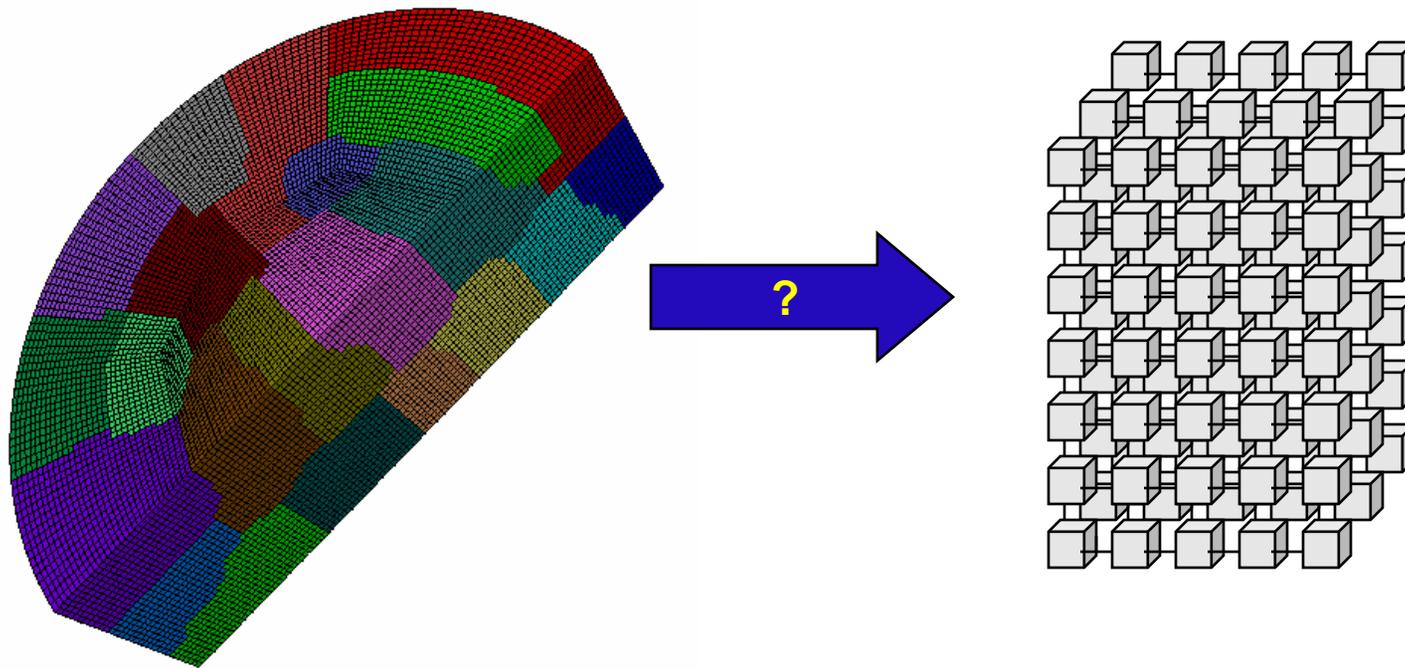


- What is `sizeof(int)` on BGL?
- In order to run on  $O(10^5)$  processors, we're going to end up with 2+ billion of *something*
- Although each processor uses local indexing, we have a handful of local->global maps that store unique global indices
- **Solution #1**
  - Ask for an “-i8” compiler switch, and build the entire code and all libraries with it
    - Pro: easy
    - Cons: Overkill. Hard to debug problems between libraries that were built differently
- **Solution #2**
  - Use long ints (or some sort of typedef) in our code
    - Pro: It's the right thing to do
    - Cons: A little bit harder. What if we miss one?

# Challenges – Domain Decomposition



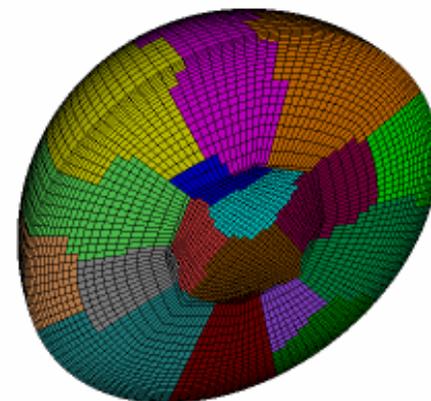
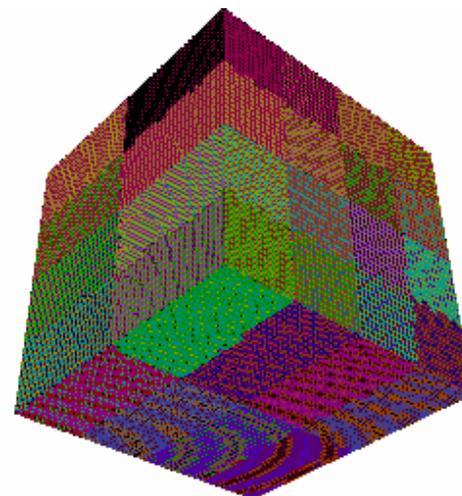
- Unstructured Mesh
- How to map this onto a 3D mesh where network locality matters?



# Challenges – Domain Decomposition and MPI Process Topologies



- **Answer #1**
  - Use regular cubic shaped problem geometries
  - Use a specialized domain decomposer (e.g. RCB) that maintains regular decomposition
  - Use `MPI_Cart_create()`
- **Answer #2**
  - Live with irregular communication typical of unstructured geometries
  - Use `MPI_Graph_Create()`
- **Either solution will be dependent on a quality mapping of MPI virtual topologies to the underlying 3D mesh hardware**
- **Neither solution is currently implemented in ALE3D**
  - Should be relatively easy to add
- **`PMI_ranktorows()` and `PMI_rowstorank()` are a good start!**
  - How about:
    - `int PMI_distance(rank1, rank2)`



## Challenges - Solvers



- **Some of our applications that require solution of a large sparse ill-conditioned matrix will drive implicit solver scaling**
  - Implicit Hydrodynamics
  - Thermal transport
  - Incompressible Flow
  - etc...
- **We have typically outsourced solver technology to CASC (and others), and will rely on them to provide scalable solutions**
- **We are currently doing studies on ASCI White to about 4k processors**
  - AMG (Algebraic Multi-grid) preconditioners are showing great promise

# Dynamic Load Balancing



- **We currently don't perform dynamic load balancing**
  - Enormously difficult, and not yet shown to be entirely necessary
- **Redecomposition of unstructured mesh incurs great expense in (for example) recalculating ghost data**
- **Complex data structures have enormous inter-dependencies, which make redecomposition error-prone**
- **Ongoing changes to the underlying software architecture hopes to make process of DLB a more attractive option**

# Dynamic Load Balancing



- We do, however, compute a load imbalance metric ( $R$ ) by tracking “wasted time” ( $W$ ) on each proc
  - I.e. time spent waiting in MPI

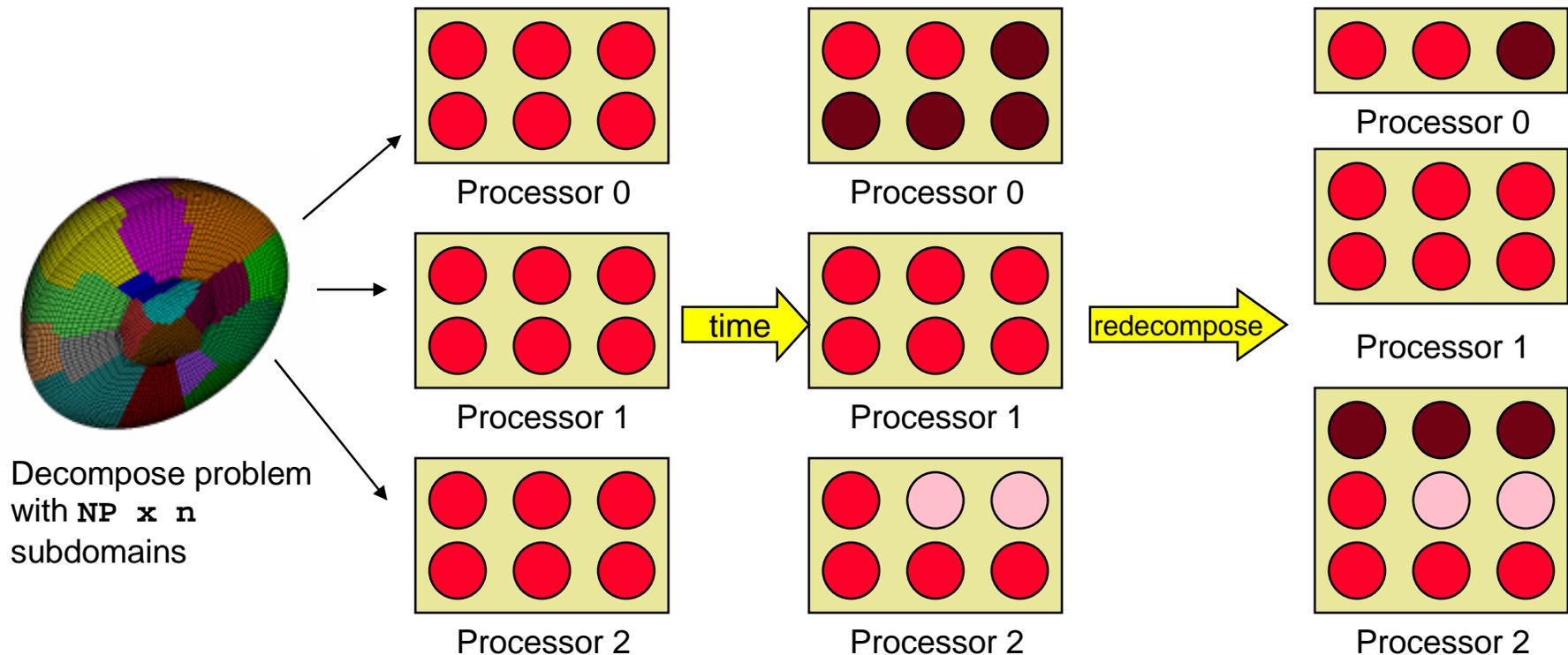
$$W = \sum_P (T_{\max} - T_i) \qquad R = \frac{W}{N_{proc} * T_{avg}}$$

- Many problems we currently run are relatively well balanced
  - $R \approx 10\text{-}30\%$
  - It’s not yet clear if our proposed problem set in combination with  $O(10^5)$  processors will entirely exacerbate this problem

# Dynamic Load Balancing by migrating entire subdomains



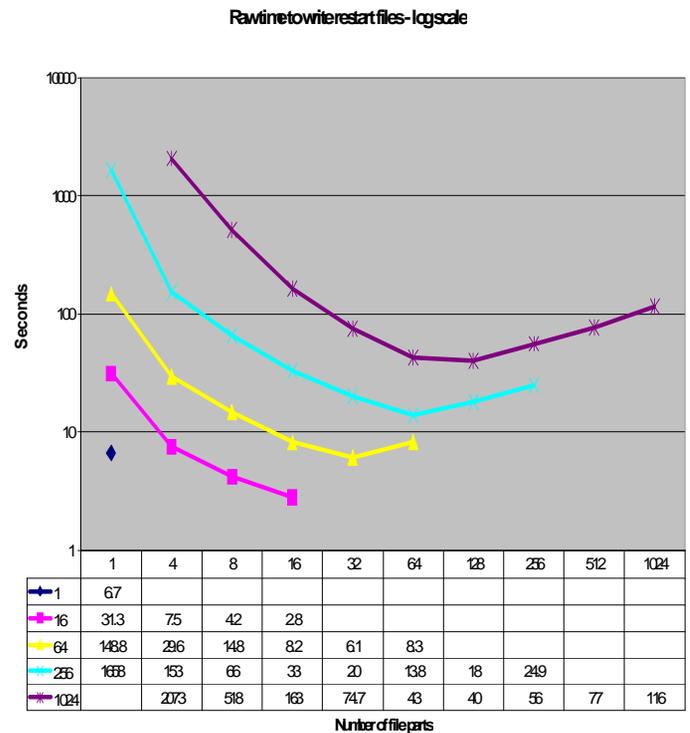
- **Another option for DLB: Migrate entire subdomains**
  - ie – base unit is a subdomain, instead of individual elements
  - Requires putting multiple (4-5) domains per processor
  - Incurs it's own overhead per processor
    - Subdomains on same processor still communicate via ghost elements



# Challenges – I/O



- The I/O system is a big question mark (at least to me) at this point
  - We spend more time writing than reading
- A typical ALE3D run will write a combination of restart and plot files
  - Restarts: 10Gb (+/-  $\infty$ ) each
  - Plots: 6 Gb each (x 250 to make a movie)
- We do have a way of doing “psuedo-parallel I/O” non-collectively
  - Allows us to “dial in” the degree of I/O parallelism
    - By specifying the number of file *parts* per plot/restart
- BGL may want MPI-IO, or some other collective I/O scheme
  - We have a prototype implementation of SAF in ALE3D

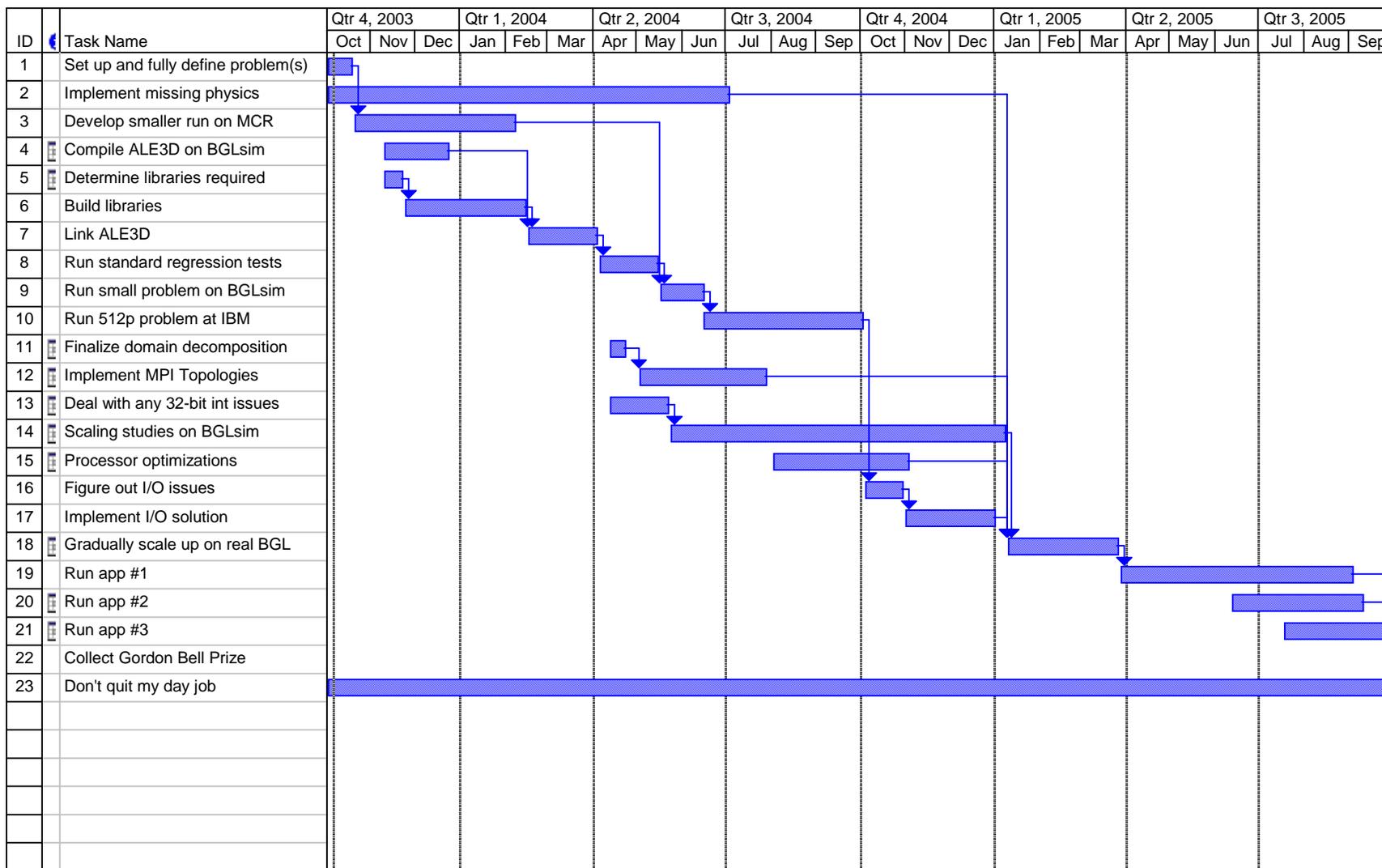


# Challenges - debugging



- **What are we going to do the first time we run 10,000 processors, and it seg-faults somewhere in initialization?**
  - Please don't say "printf"
    - We have 500,000+ lines of code
    - 3<sup>rd</sup> party libraries crash, too
- **Lightweight core files?**
  - Would be very helpful
- **Regular core files?**
  - Would surely cause problems (unless limited to processors that generate SIGSEGV or SIGBUS – and not SIGTERM)
- **A full debugger? You know it!**
  - Not always necessary for post-mortem analysis, but something we'll want
- **Being able to reproduce bugs at smaller scales is usually our only hope.**

# Detailed planning this early in the game is futile - but there are dependencies....



# Conclusions



- **There are several programmatically relevant ALE3D calculations ready to begin work on**
  - We plan to take full advantage of the 16 month lead time by running on existing platforms (e.g. Thunder, MCR, etc...) as well as BGLsim
- **Our definition of success will be based more on solving previously intractable problems – versus setting a Tflop record**
- **We don't anticipate any major roadblocks**
  - But there are the elusive “*unknown unknowns*”...
  - Running simpler problems at large scale will hopefully point out problems early
    - e.g. I/O, 32-bit, dom decomp, etc...
- **There's hopefully ample time and resources to work out issues before the final hardware arrives**

# Acknowledgements



- **Many others contributed to this talk.**
- **Special thanks to:**
  - Jack Reaugh (HE grain scale)
  - Rich Becker (material fragmentaion)
  - Rose McCallen, Al Nichols (thermal cookoff)
  
- **Questions?**