Intuitive Visualizations for Performance Analysis at Scale

Charm++ Workshop

April 15 2013

Todd Gamblin
Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

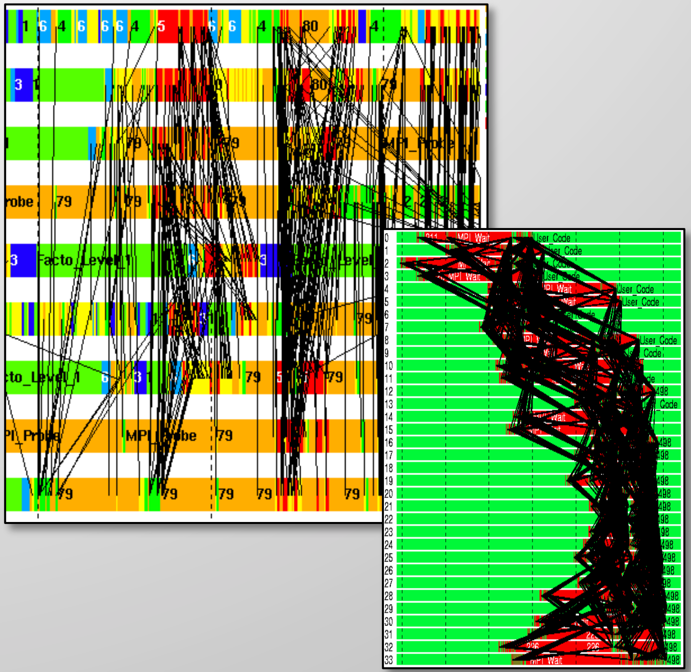
LLNL-PRES-635584
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

The PAVE Team

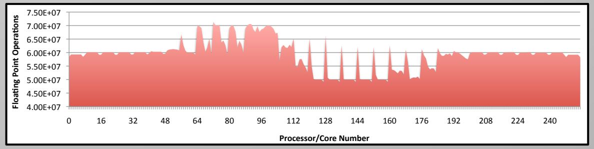
- **Lawrence Livermore National Laboratory**
  - Abhinav Bhatele, Peer-Timo Bremer, Todd Gamblin, Nikhil Jain (UIUC), Martin Schulz
- **University of Utah / SCI Institute**
  - Aaditya Landge, Valerio Pascucci
- **University of California Davis**
  - Bernd Hamann, Kate Isaacs
- **Clemson University**
  - Joshua Levine

Massive parallelism has made performance a data-rich field, but we lack the tools to interpret and understand the data

- **We can collect data at scale, but data volume is overwhelming**
  - Too many variables to measure
  - Difficult to write out data from 500 million cores, even if we do measure it
- **Information is highly categorical, discontinuous**
  - Profiles, traces
  - Hardware Performance Counters
    - — FP counts, cache misses, network traffic
  - Counts map to particular cores
- **MPI Process ID space is often unintuitive**
  - Rank offers little insight into underlying network
- **It is difficult to apply analysis techniques because this data lacks structure**



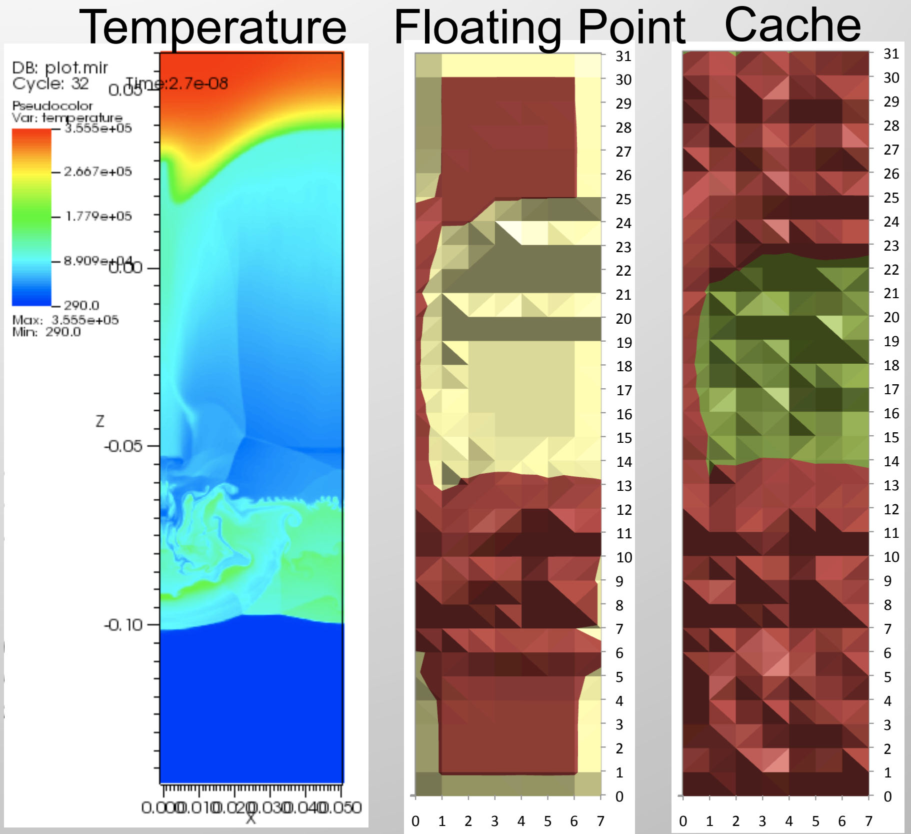
MPI Trace Data from runs with 16 and 34 processes

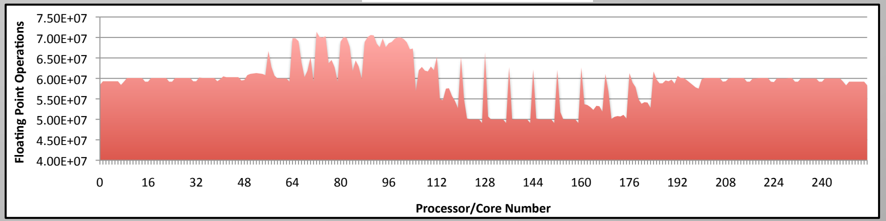


Floating Point Instruction Counts

Presenting performance data on more familiar, intuitive domains provides opportunities for new insights

- **How can we make raw performance data more useful?**
- **Application developers understand application data**
    - Temperature plot of Miranda data
- **What if we could map performance attributes onto application data structures?**
- **We constructed a simple example to show FP data in the application domain**
    - Can show cache misses similarly
- This is only the tip of the iceberg
    - We must extend this approach to more domains and more complex data
    - We will enable feature-based analysis and correlations



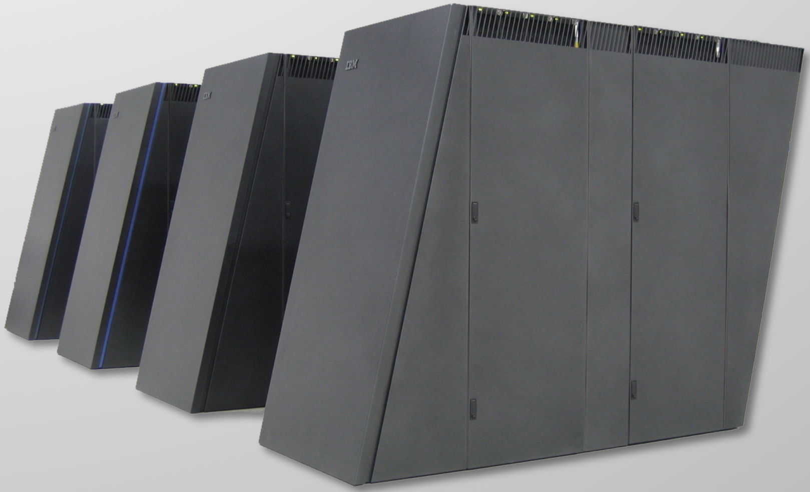


PAVE will develop methods to organize and analyze performance data in domains familiar to scientists

- **PAVE will develop new analysis techniques to:**
  - Attribute performance measurements to intuitive domains: *Physical, Hardware*, and *Communication*
  - Extract features within data domains
  - Correlate features between domains
  - Analyze mappings among domains

- **This work can only be accomplished by combining and extending state-of-the-art techniques from two fields:**
  - We will extend run-time performance analysis for application-semantic attribution at scale
  - We will restructure performance data so that it is amenable to analysis
  - We will develop analysis techniques to correlate and map features between these domains

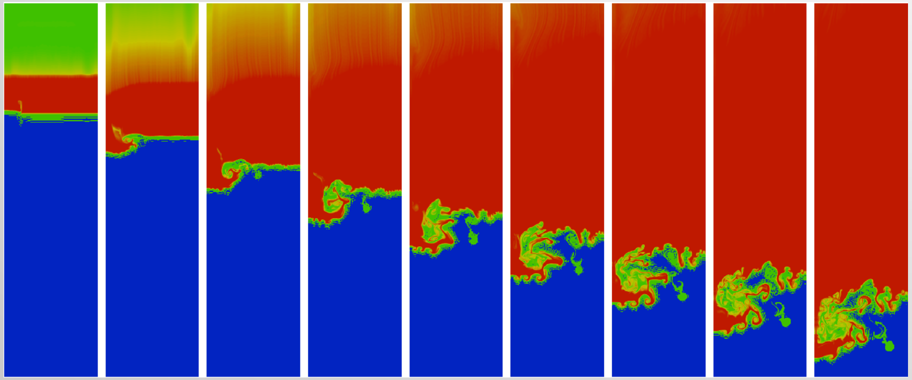
- **Data domains, the correlations between them, and the analysis of their mappings will provide new insights into application performance**



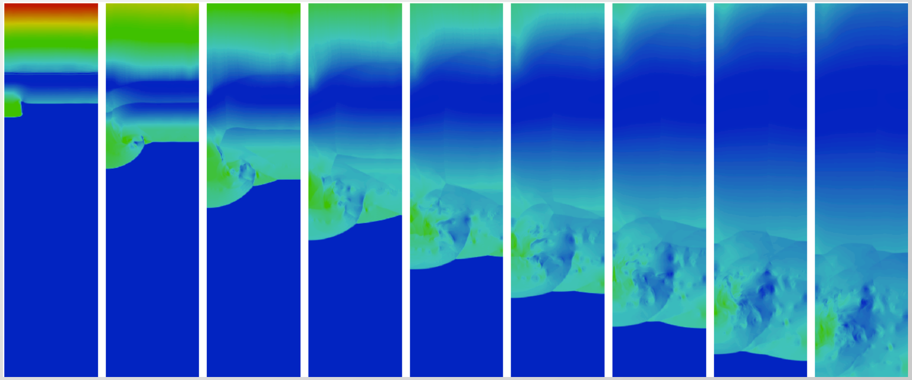
PAVE Overview

1. **Hardware to Application mapping**
   - Data-dependent computation in fluid dynamics simulations
2. **Communication Visualization for AMR**
   - Visualizing bottlenecks in
3. **Boxfish network visualization tool**
   - Plotting network counters on the network
4. **Future Directions**
   - Higher resolution hardware to application mapping
   - Adding structure back to parallel trace visualizations

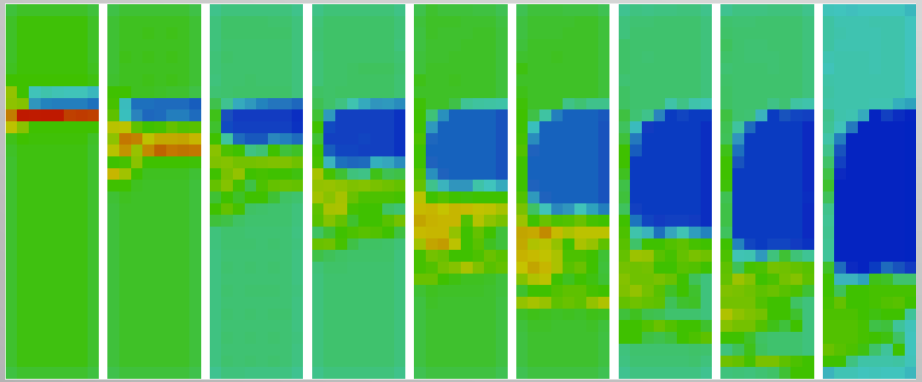
Projections on the app domain



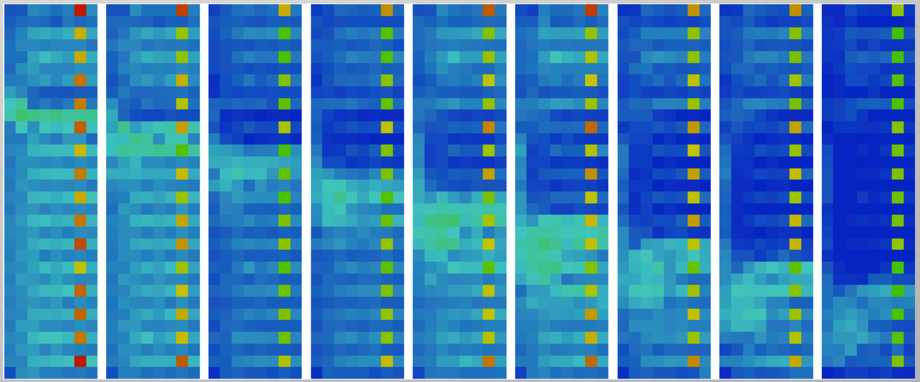
Aluminum distribution



Velocity distribution



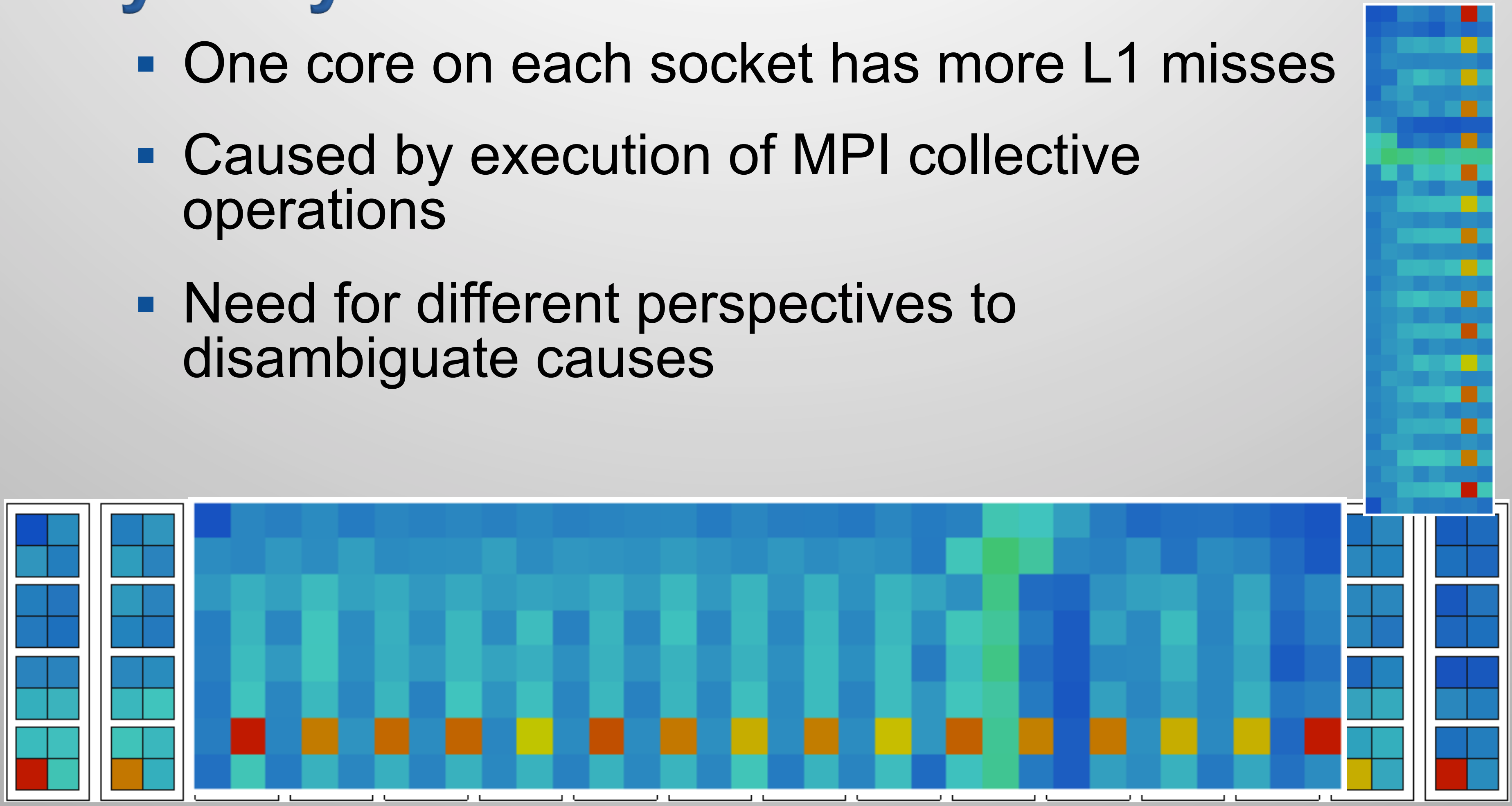
Floating point operations



L1 cache misses

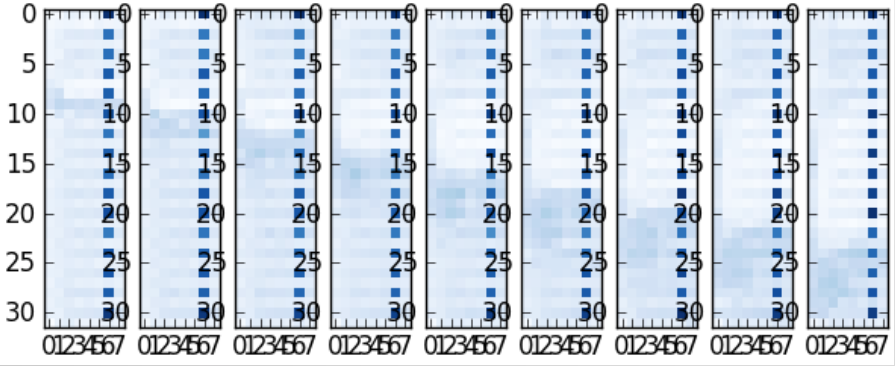
Mystery of L1 misses

- One core on each socket has more L1 misses
- Caused by execution of MPI collective operations
- Need for different perspectives to disambiguate causes

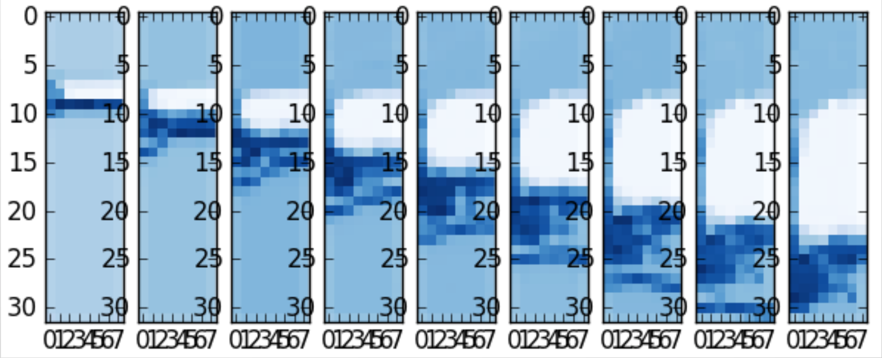


We are developing techniques to automatically segment features in performance data

Same data with linear color map



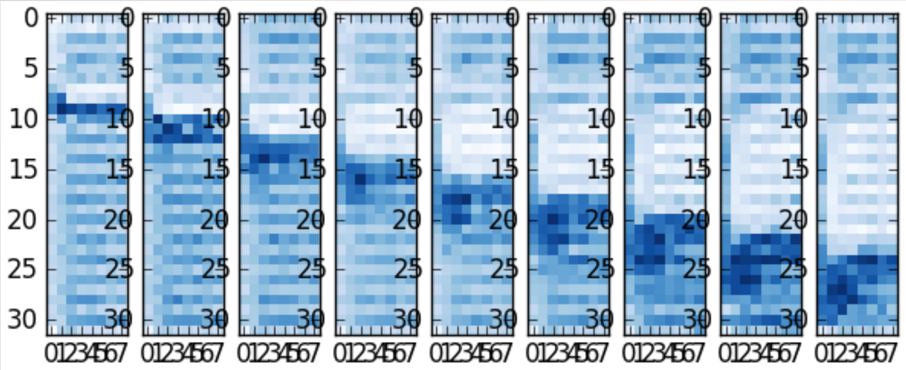
L1 Cache Misses



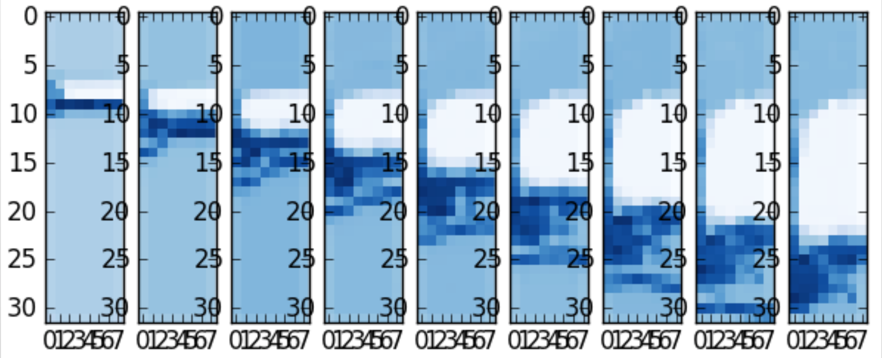
FP Operations

We are developing techniques to automatically segment features in performance data

Same data with linear color map



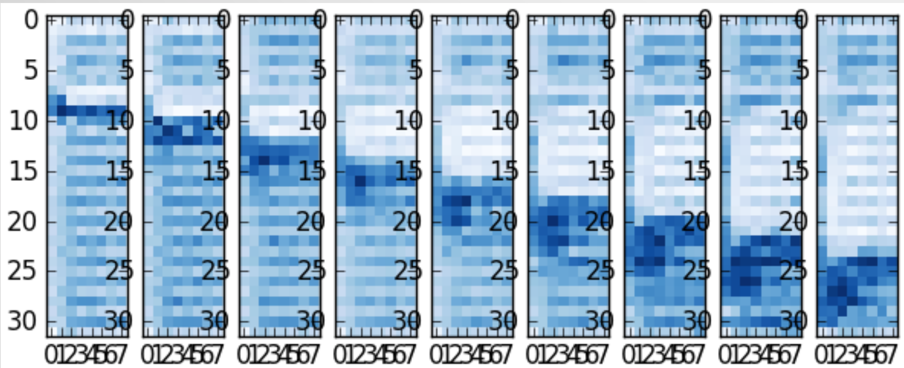
L1 Cache Misses with MPI worker filtered



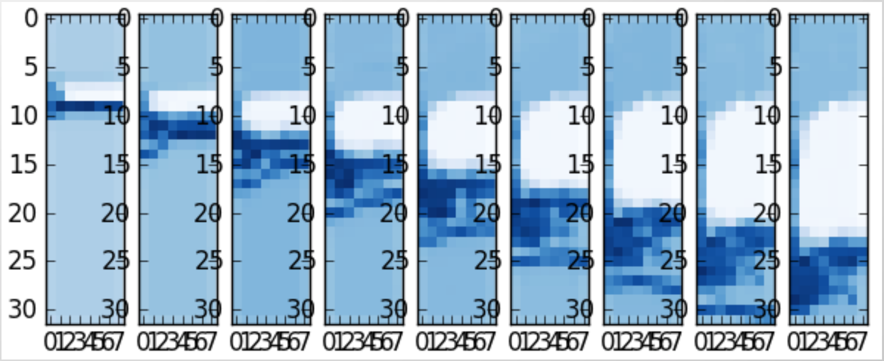
FP Operations

We are developing techniques to automatically segment features in performance data

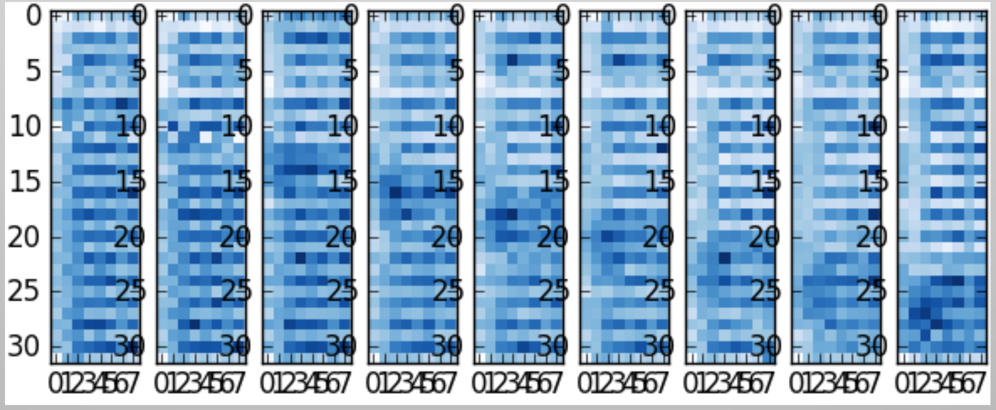
Same data with linear color map



L1 Cache Misses with MPI worker filtered



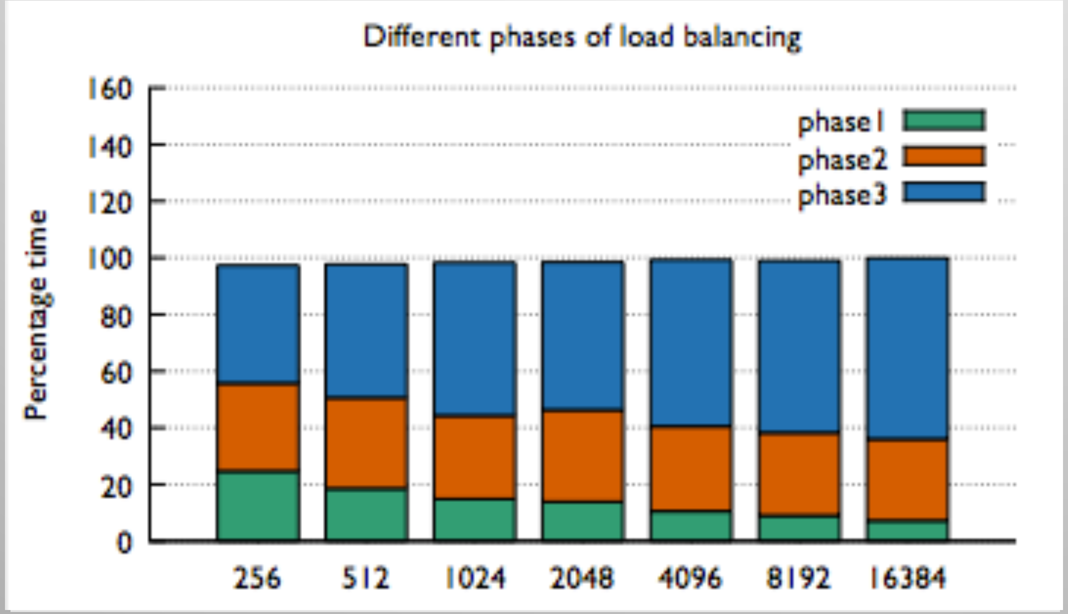
FP Operations

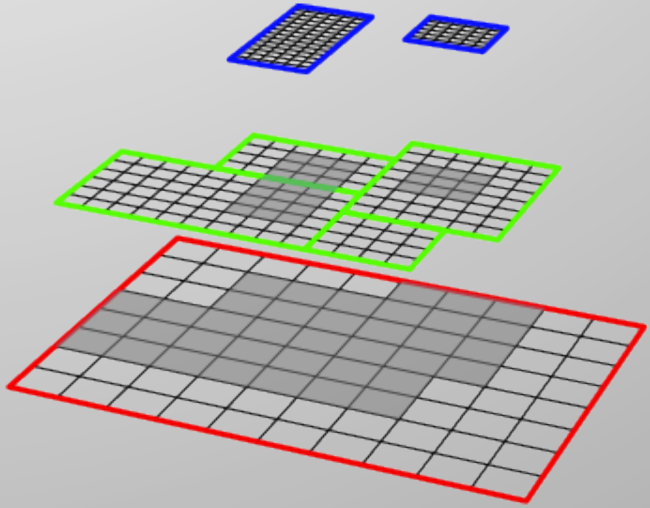


L1 Misses per FP operation: Proxy for efficiency

Projections on the comm domain

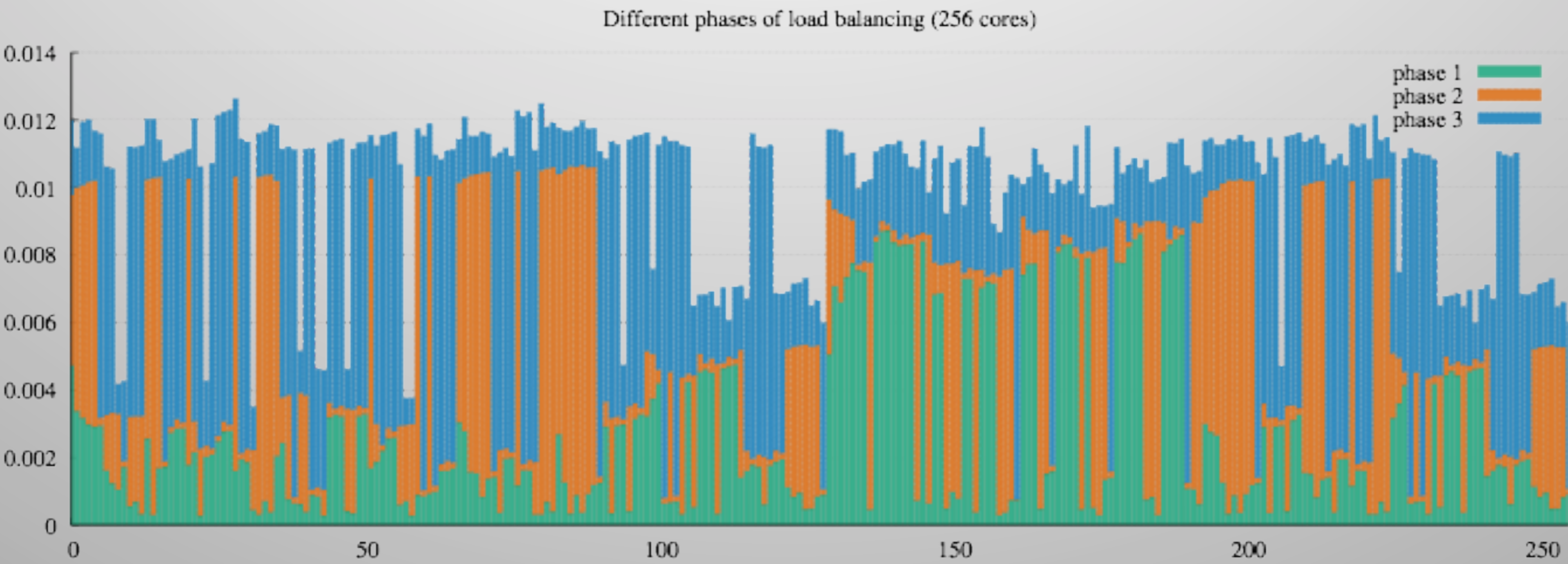
- Case study: SAMRAI, structured adaptive mesh refinement
  - Blue Gene/P at Argonne (256 to 128K cores)





Timings in MPI rank space

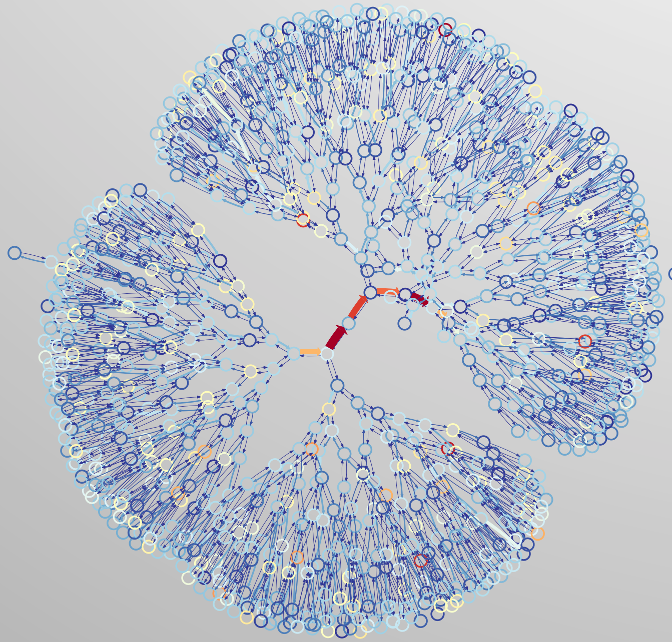
- Bottleneck is in phase 1 and not phase 3



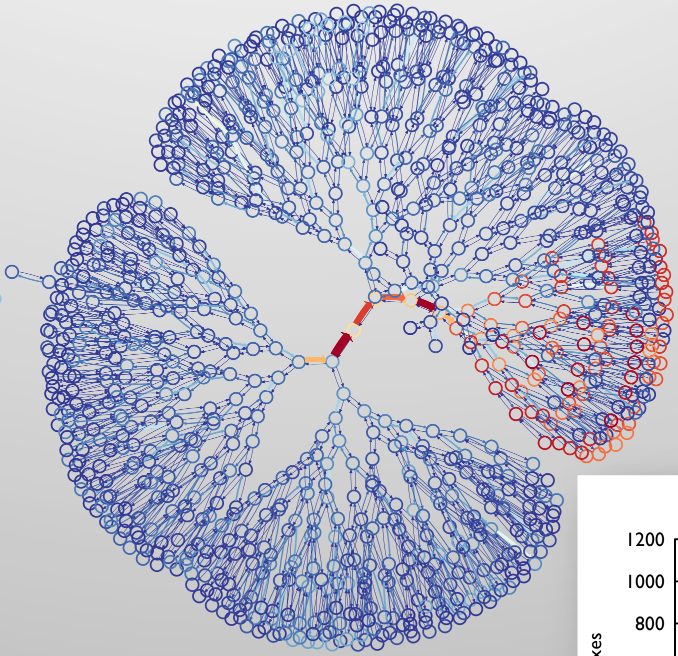
MPI rank

We plotted performance metrics on the SAMRAI communication graph

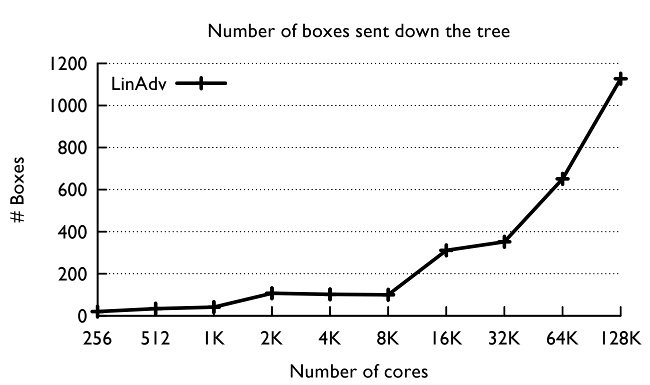
Load (Cells per process) Before balancing



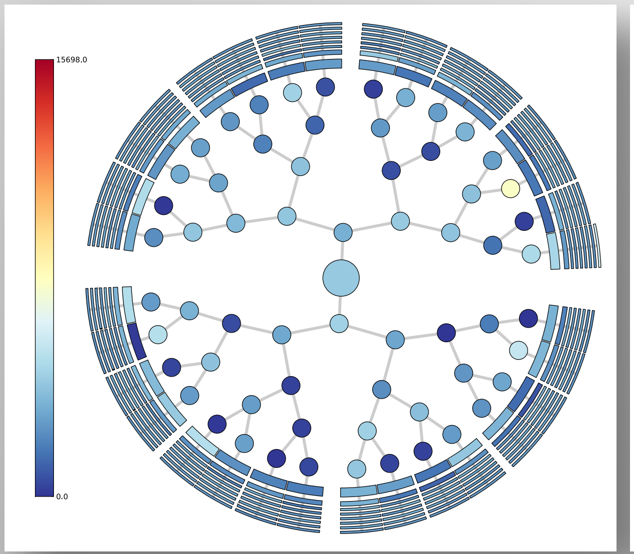
Time spent redistributing boxes



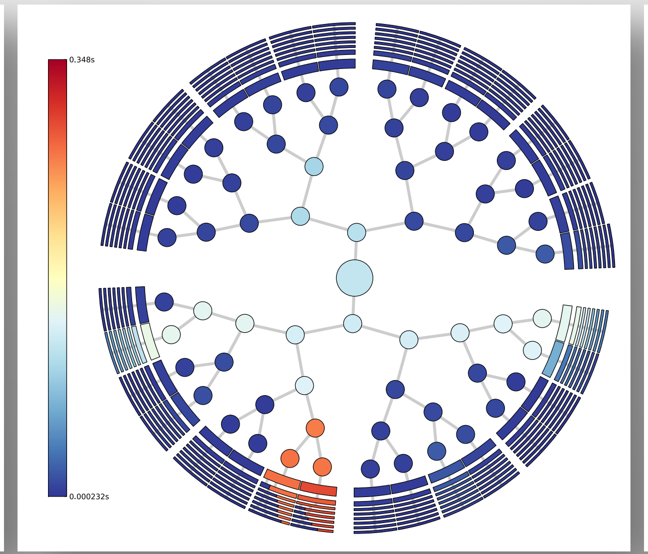
1024 processes



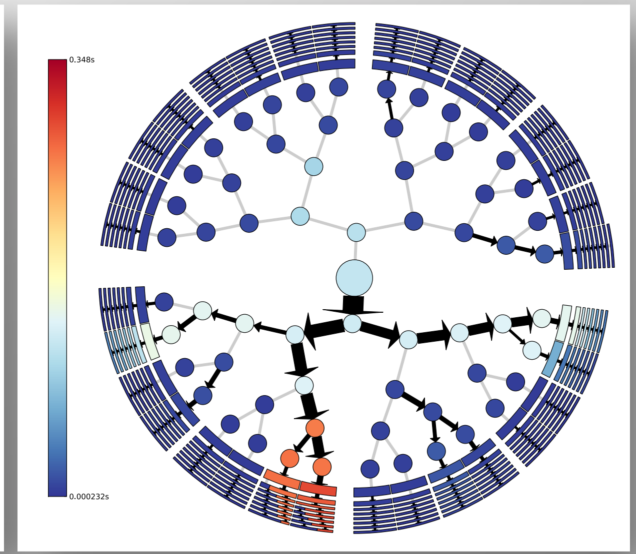
We developed a scalable visualization for large communication graphs



Load on 16k cores



Wait time for box distribution

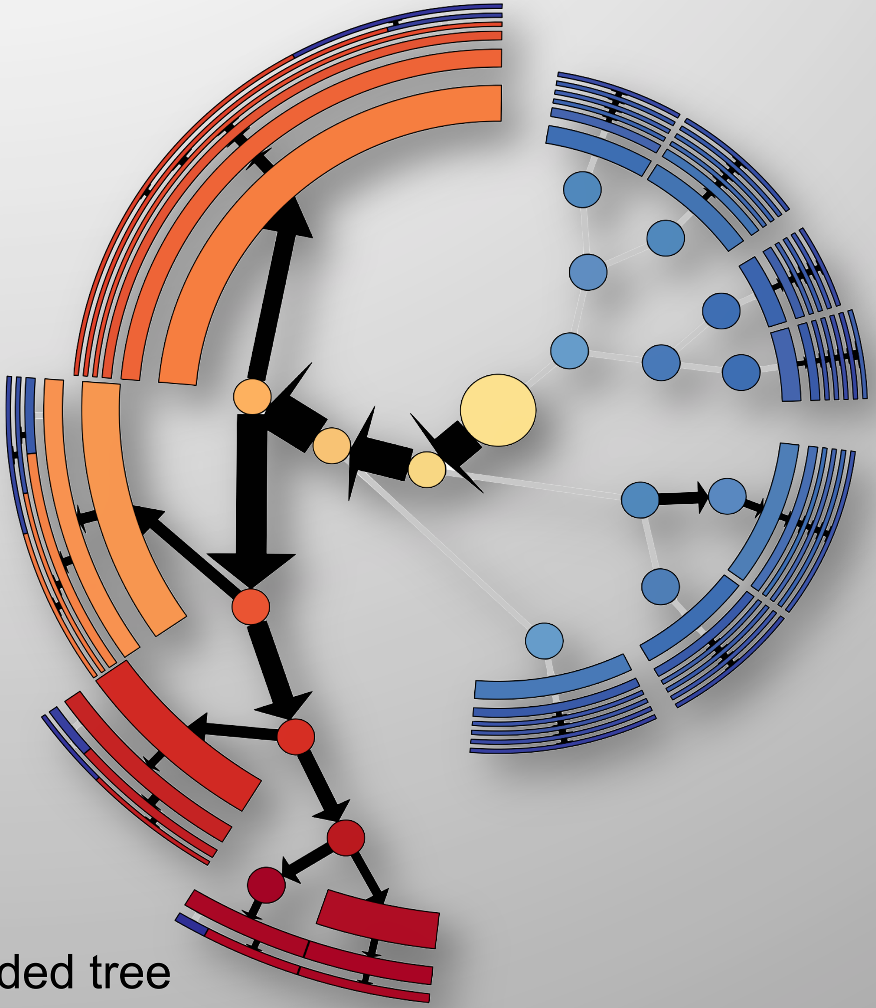


Wait time with flow information

This shows data for 16,384 cores

Our visualization can be adaptively refined for more detail

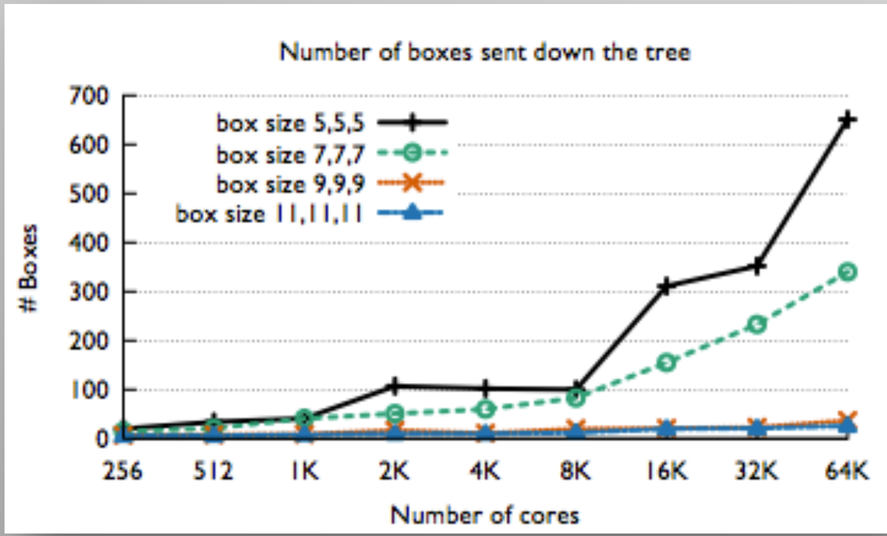
- Angles are apportioned by flow in the subtree
- Heavier trees are expanded to a deeper level
- Can see flow problems at any level of the tree

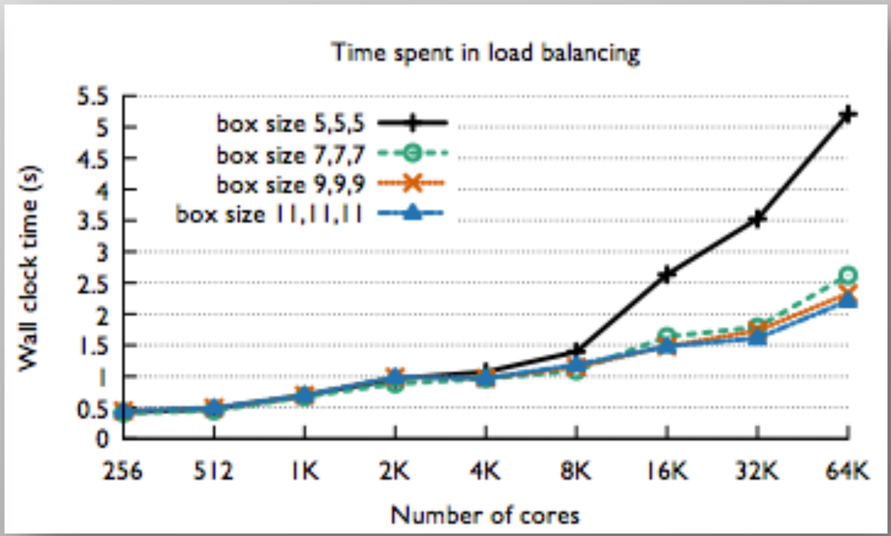


1024-process expanded tree

Performance improvements

- Mitigate the problem by reducing the size of box metadata
- Trade off slightly increased imbalance for coarser boxes
- Leads to 50% reduction in load balancing time at 64k cores
  - **22% reduction in overall time on 65k cores**



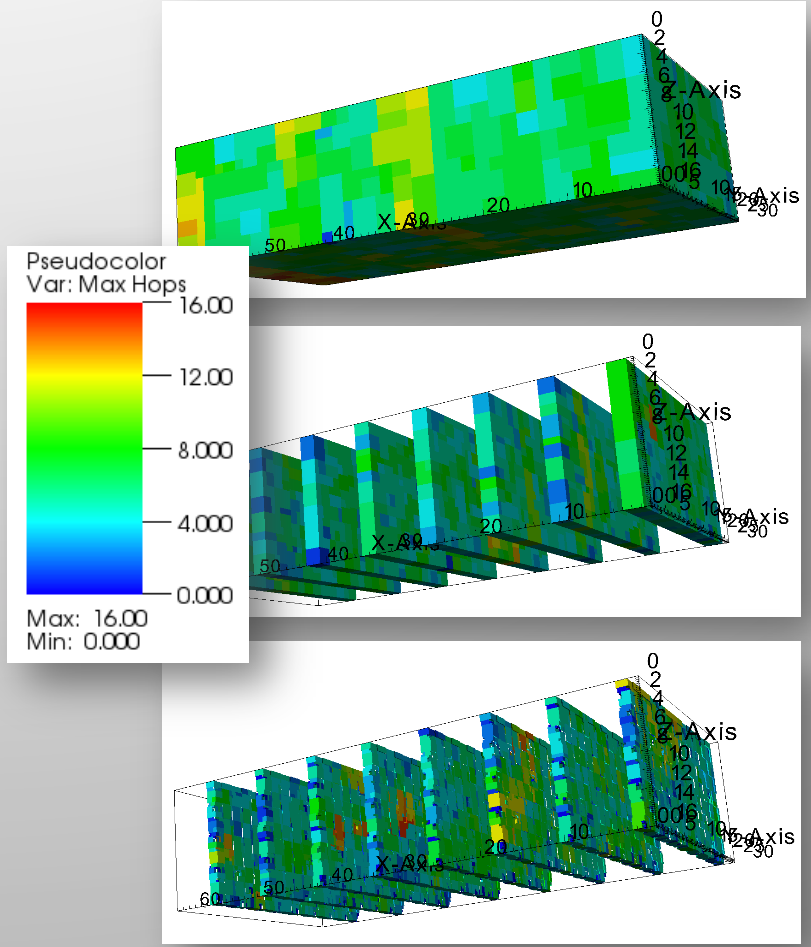


We anticipate that network topology will become a performance issue once load balance is *O(log(P))*

- We have mapped network measurements to SAMRAI patches

- Plots show patches colored by maximum hops *on the physical network* to any neighbor patch

- SAMRAI LinAdv benchmark does not appear to be affected by this imbalance, but other codes will be.

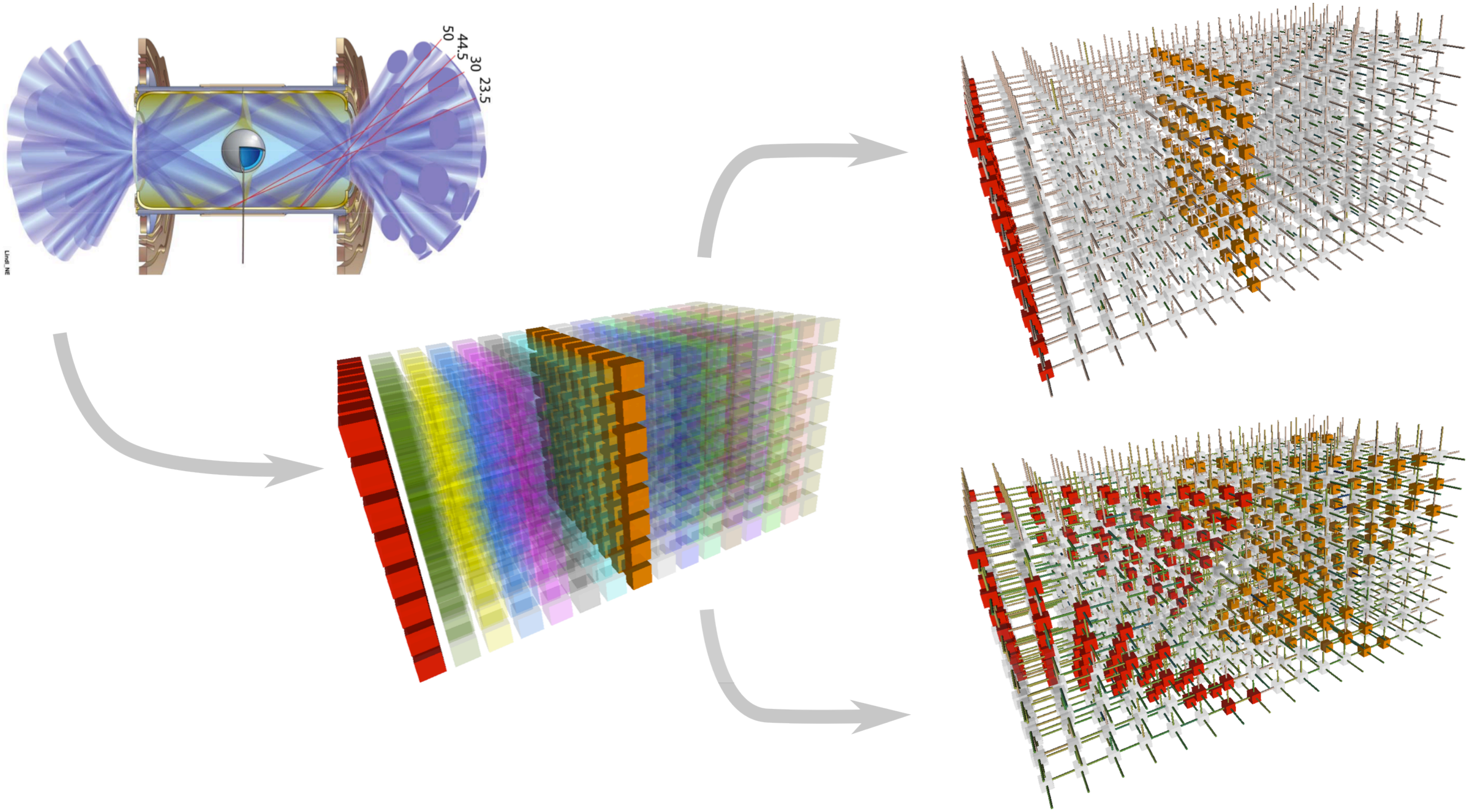


L1

L2

L3

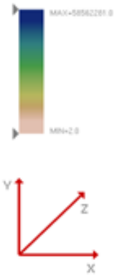
We have developed the Boxfish visualization tool to better understand network performance



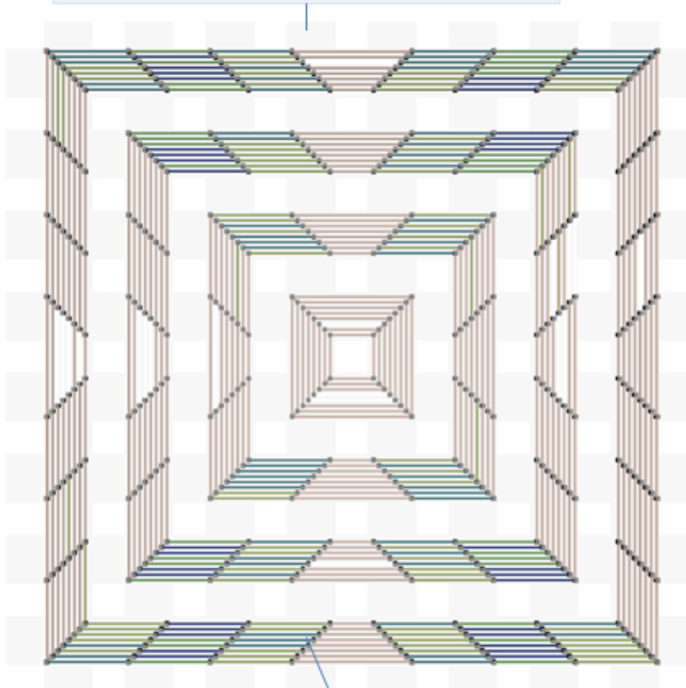
We have developed the Boxfish visualization tool to better understand network performance

Overview Minimaps for giving context once user zooms in. Color based on mean link values for the links in that plane. **Three different orientation options** as user can look into the 3D view in the X, Y or Z directions. These views also give an **overview** of the communication behavior

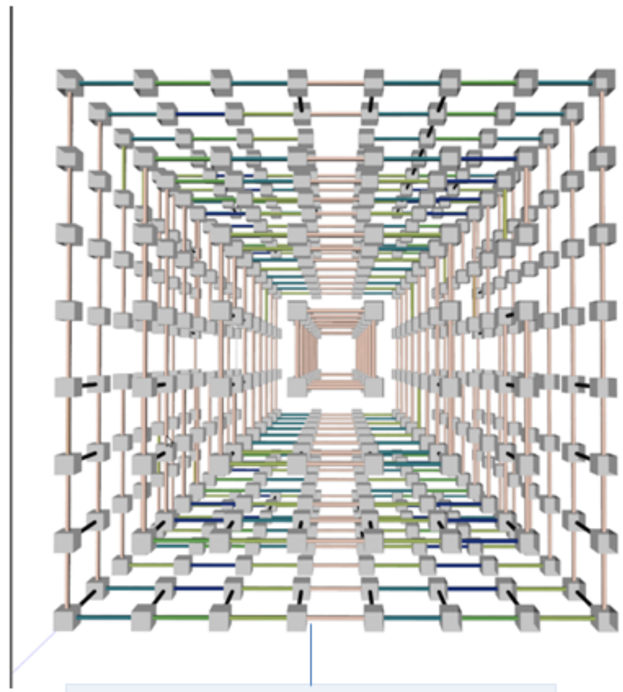
Interactive legend for selecting range of values to be viewed. **Axes** showing current orientation of the view. Clicking on any of the directions shows links in only that direction.



Grid layout to highlight planes in the 2D view with **mouse over.** The 2D view also supports interactions like **zooming** and **translations**

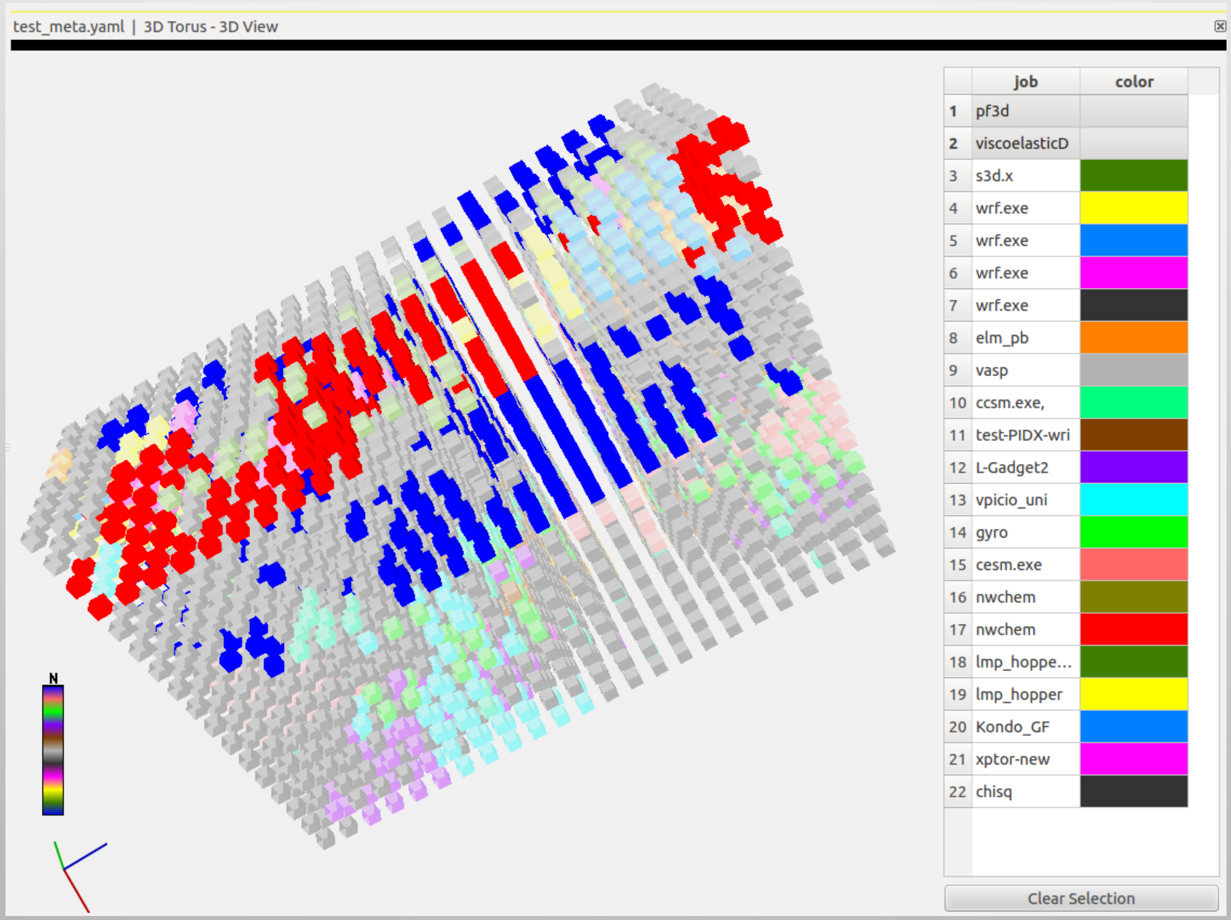


The 2D view can display all nodes without any occlusion. Only half of X links and half of Y links and all of Z links are shown. The Z links are along the diagonals.

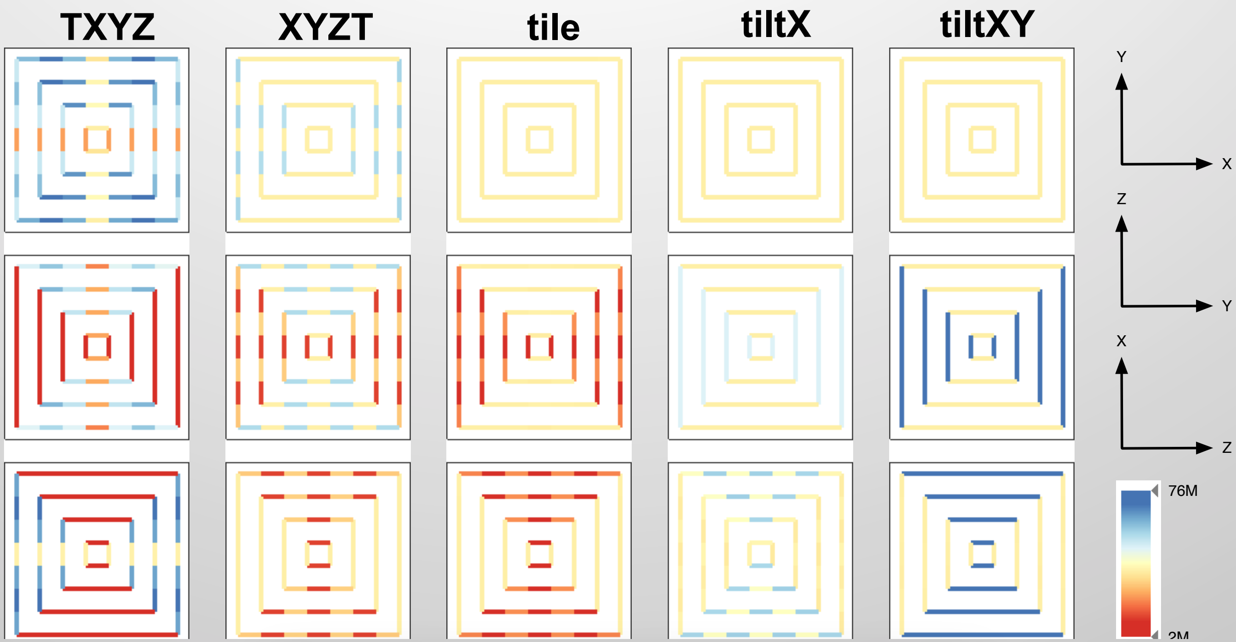


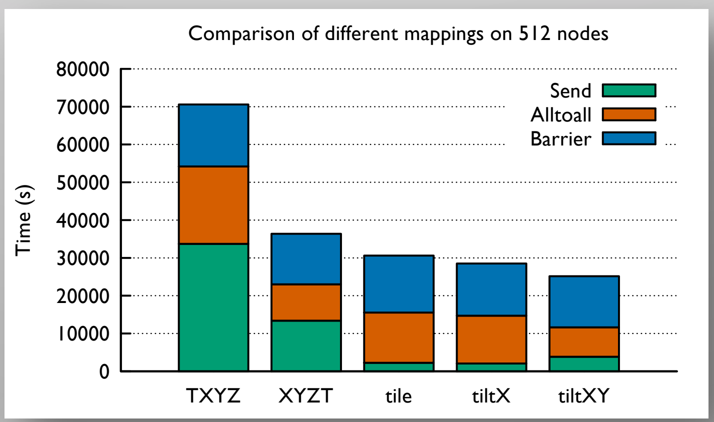
3D view supports interactions like **zooming, rotation and translation.**

We can use Boxfish to visualize job layout on Cray machines



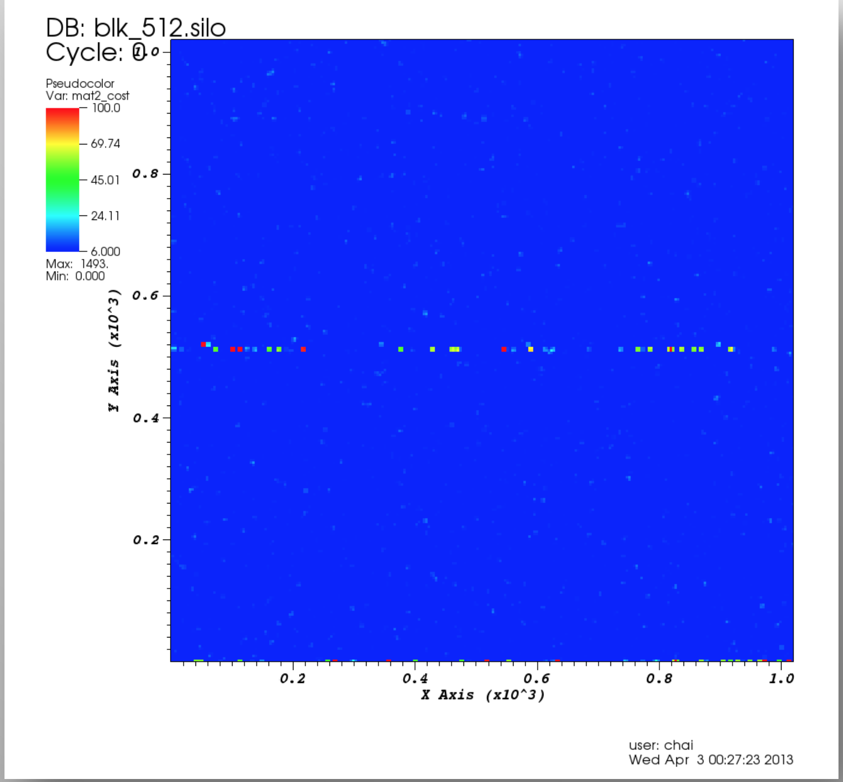
Boxfish's 2D mini-maps summarize bandwidth and help to explain performance differences



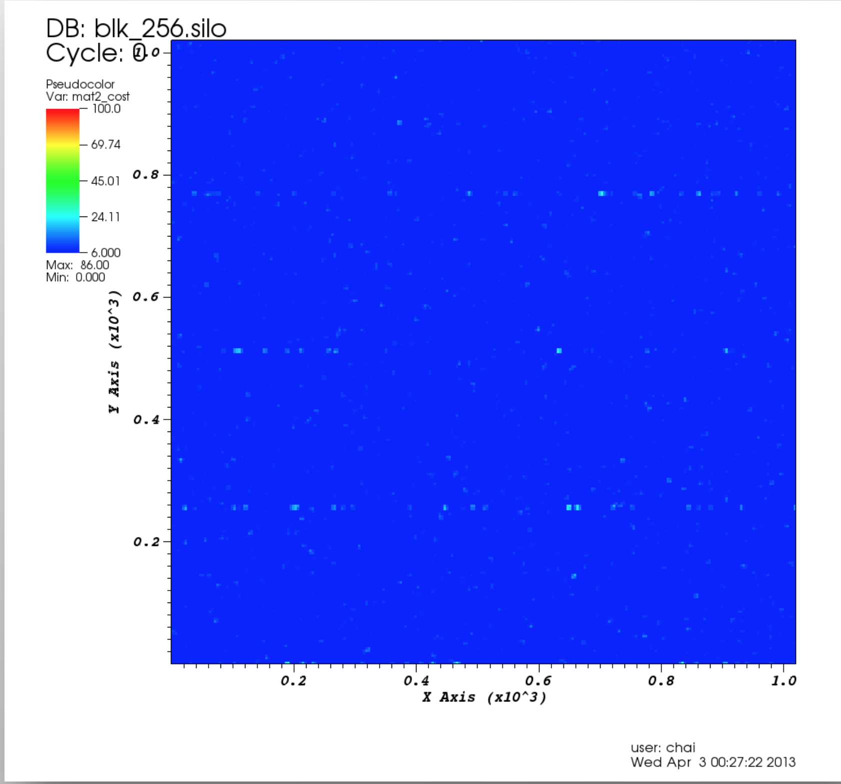


Future Directions: Fine-grained Application Mapping using PEBS

1000x1000 Matrix Multiply with different Blocking Optimizations



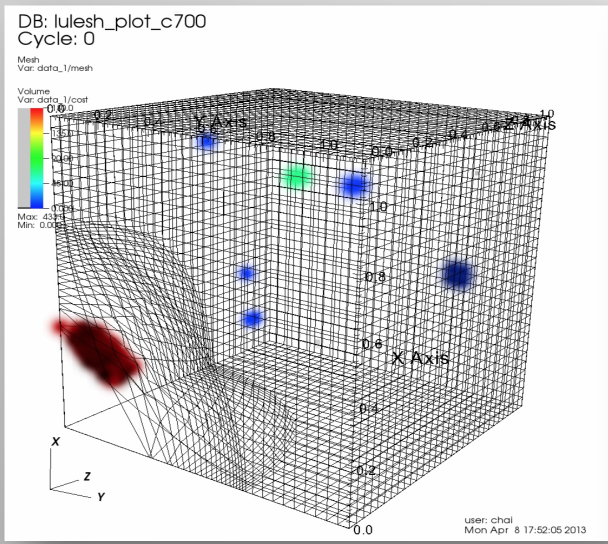
512x512 Blocks

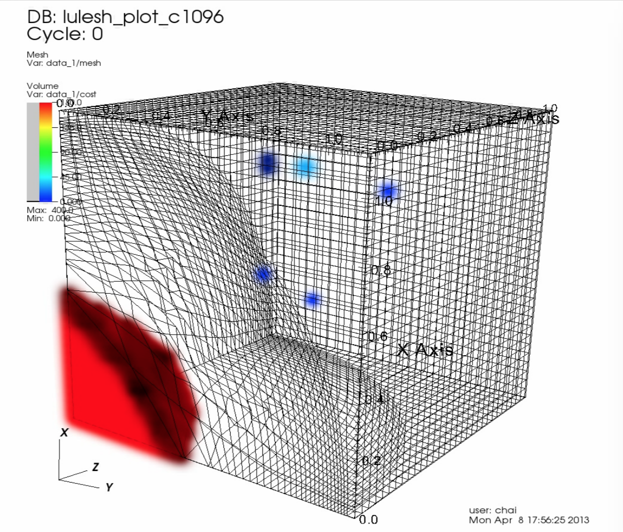


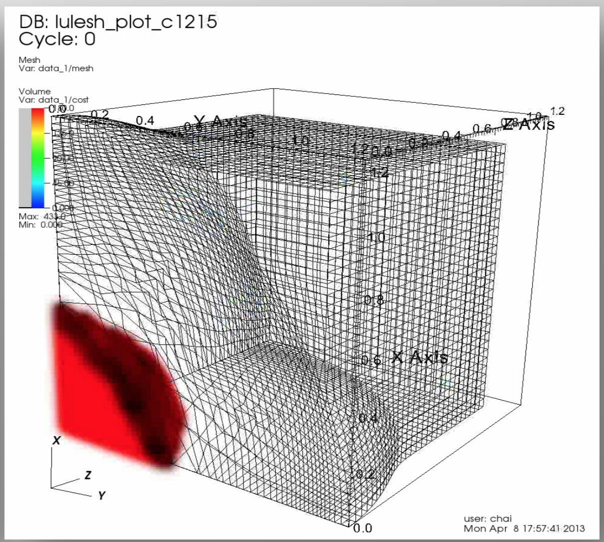
256x256 Blocks

Future Directions: Fine-grained Application Mapping using PEBS Counters

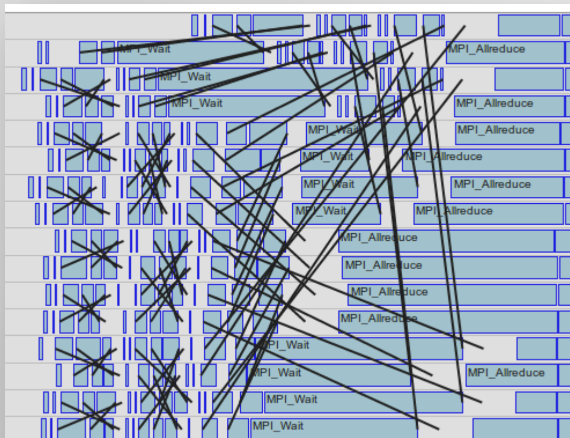
LULESH



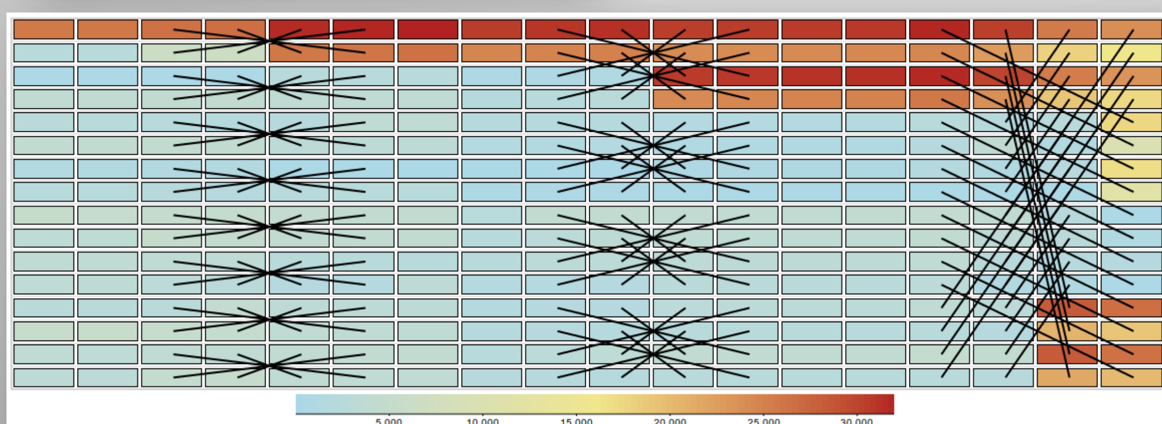




Future Directions: Adding Structure Back to Parallel Trace Visualization



Messy Multigrid Trace
- With Real Time



Same trace!
- With logical time steps
- Colors show lateness

Lawrence Livermore National Laboratory