Providing More Intuitive Performance Analysis through Scalable Visualizations

Martin Schulz

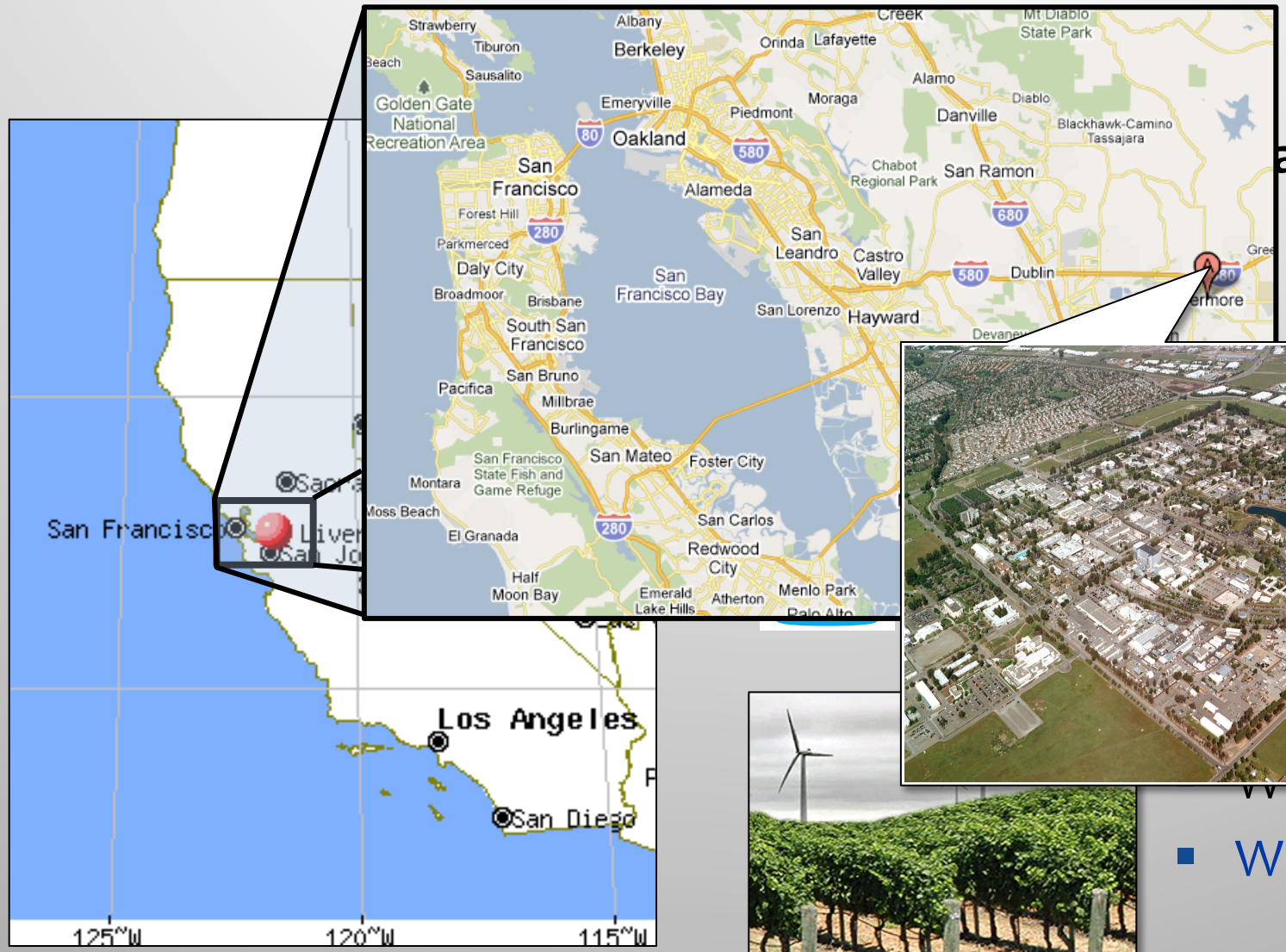
Lawrence Livermore National Laboratory

CHANGES Workshop ◆ Sept. 5th, 2012

LLNL-PRES-578932

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Where is Livermore?



- Windmills

Lawrence Livermore National Laboratory

- **2 km² site in Livermore, CA**
  - ~6,800 employees
  - Interdisciplinary
- **Primarily funded by the U.S. Department of Energy**
  - Budget: $1.6 billion
- **Mission:**
  Apply science and technology to:
  - National Security
  - Energy Security
  - Economic Competitiveness

Livermore Computing

IB Cluster

BlueGene/L

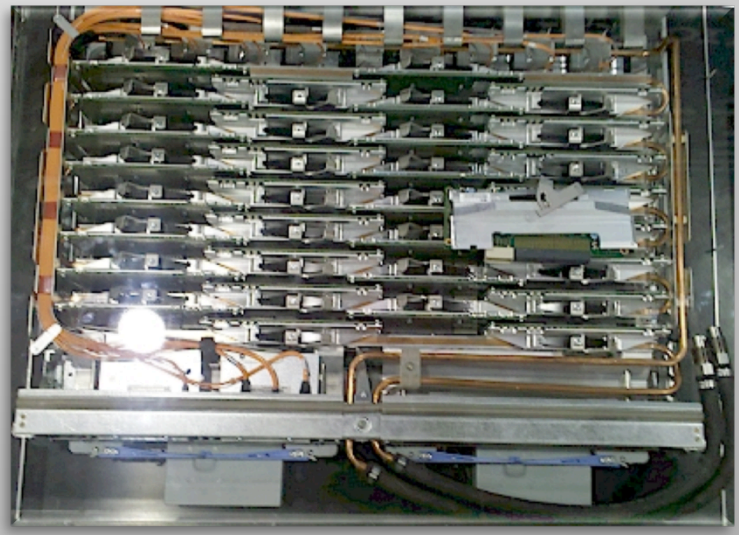
Dawn

- **Long supercomputing tradition**
  - Simulation in support of LLNL missions
- **BlueGene/L (~600TF) – 212,992 cores**
  - #1 machine from 2005-2007
  - #8 on the Top500 in June 2011
  - Last part decommissioned later this year
- **Some other current machines (as of 2011)**

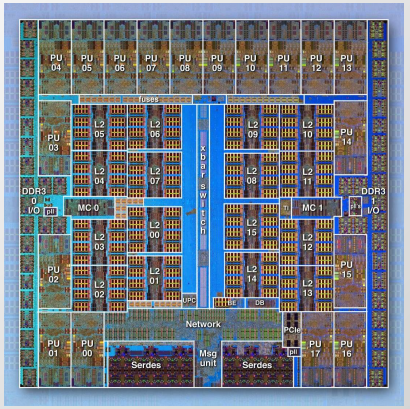
Zin	Sandy Bridge/IB	~1 PF
Dawn	BlueGene/P	~500 TF
Cab	Sandy Bridge/IB	~425 TF
Sierra	Nehalem / IB	~261 TF
Juno	Opteron / IB	~160 TF
Hera	Opteron / IB	~120 TF
Graph	Opteron / IB / GPU	~110 TF
Hyperion	Nehalem / IB	~90 TF

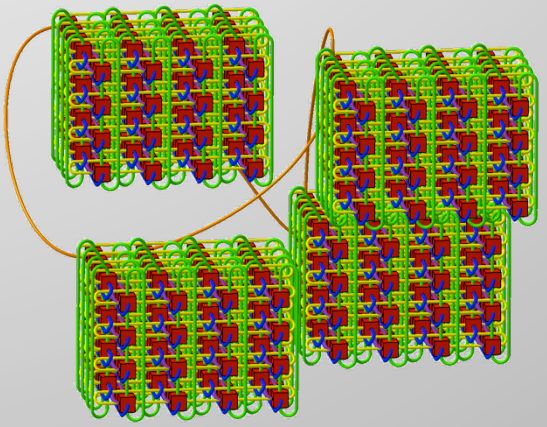
The Next Generation at LLNL: Sequoia

- **Blue Gene/Q:**
  - 20PF/s peak
  - 96 racks, 98,304 nodes
  - 1.5M cores/6M threads
  - 1.5 PB memory
  - Liquid cooled
  - 5D torus interconnect
  - New technologies like HW-TM



Complexity is on the Rise





- **Architectures are getting more complex**
  - Huge process and/or thread counts
  - Deep memory hierarchies
  - High dimensional network topologies
  - New accelerators and hardware features
- **More constraints**
  - Less memory per core
  - Power ceilings
  - Reliability
- **Applications are getting more complex**
  - Scale-bridging codes
  - Heterogeneous applications
  - Integration of UQ

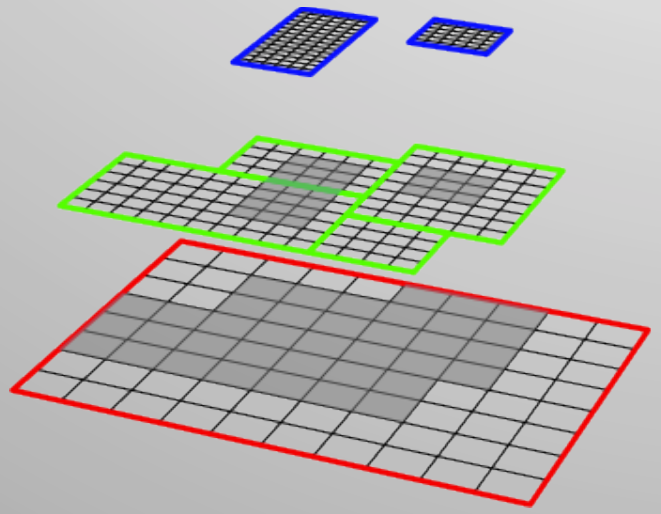
We need to understand the impact of these complexities and their interactions on application performance

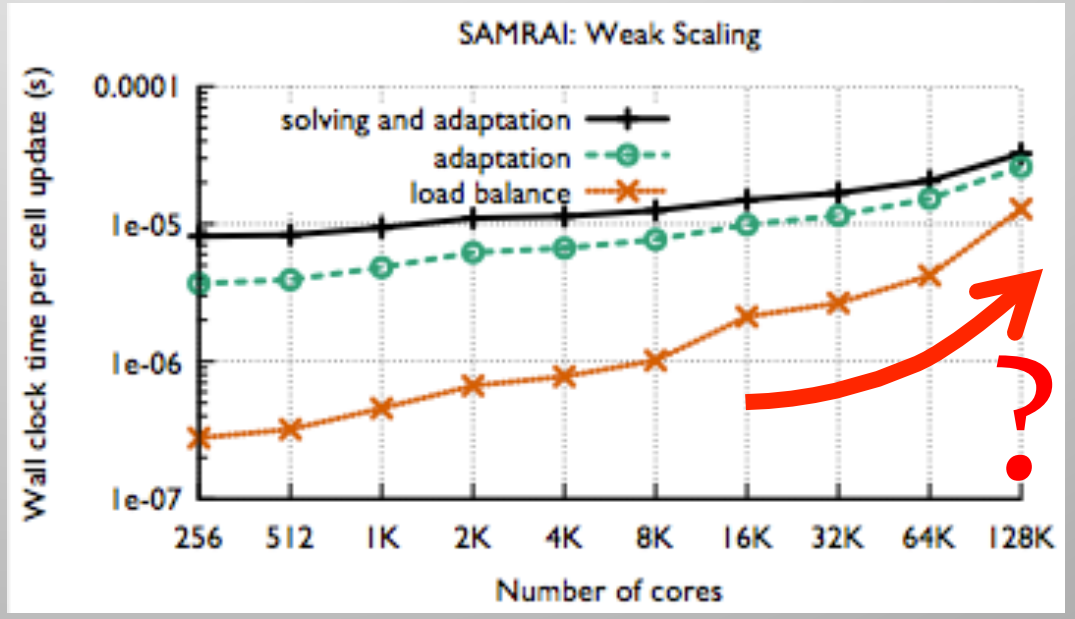
Complexity Directly Impacts Programmers

- **It will be a challenge to achieve efficiency**
  - Load balance will be key at billions of threads
  - Exploiting new hardware features
  - Reduction of data movements
  - Memory and network architectures will require layout optimizations
- **Definition of efficiency needs to be revisited**
  - Heterogeneous systems/nodes/chips/units/…
  - Multiple optimization targets (power vs. reliability vs. memory vs. speed)
  - Self-adaptive systems at all layers
  - Baselines are no longer obvious
  - Need to think about machine wide resource utilization
- **Programmers need tools and performance models more than ever**
  - Visualize/Illustrate code behavior
  - Distinguish “good” from “bad” behavior
  - Identify critical regions and bottlenecks in the code
  - Track down root causes of code behavior

Example: Scalable Load Balancing in AMR

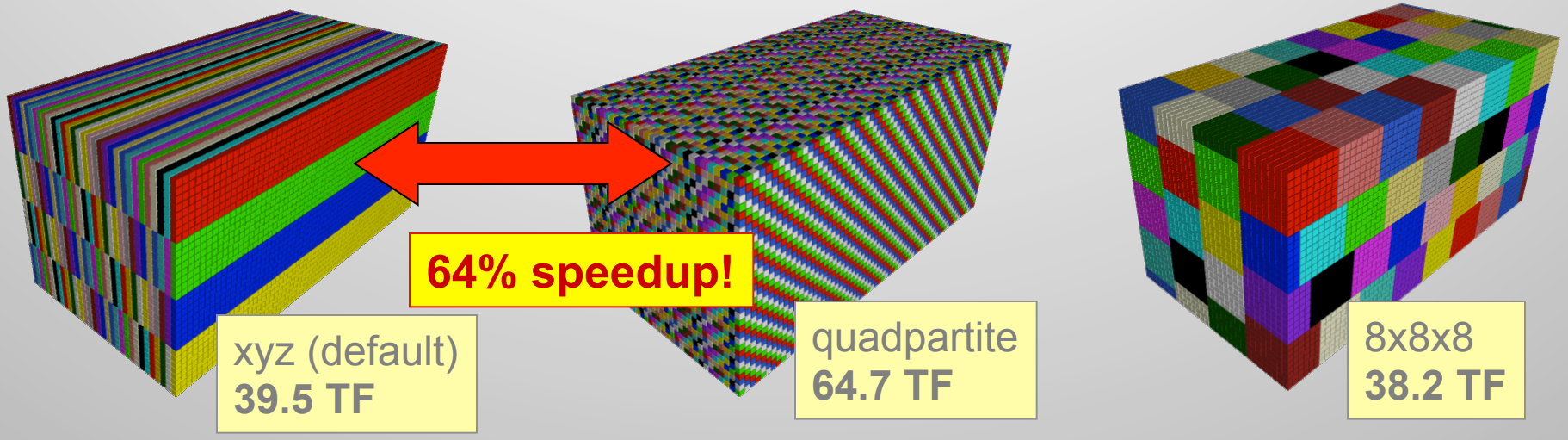
- **Adaptive Mesh Refinement (SAMRAI library)**
  - Different levels of patches to refine in areas of interest
  - Requires active load balancing
  - Load balancing shows bad scaling behavior
  - Dominates at large scale





Example: Node Mappings in Torus Networks

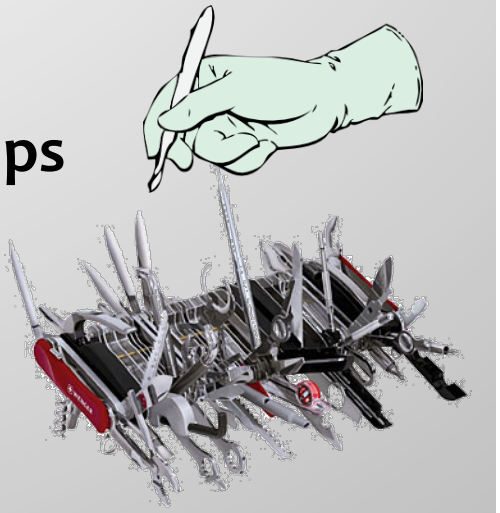
- **First principle molecular dynamics code**
  - Dense 2D matrix as the base data structure with X/Y communication
  - Need to map rows/columns onto 3D torus of BG/L



- **Understanding of performance is essential**
  - Why do certain mappings perform better?
  - How can we select and create mappings efficiently?
  - We must do this as scale!

Performance Tools are a Necessity !

- **Demand is increasing**
  - Traditionally little development time invested in optimization
  - Code teams are getting more interested in tools
- **Widely researched in many projects and groups**
  - From specialized tools to comprehensive toolkits
  - Different data acquisition (sampling and/or tracing)
  - Different instrumentation options
  - Commercial and open source options

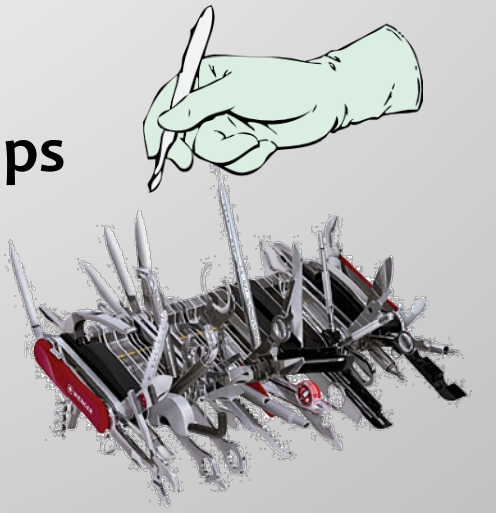


Wide Range of Performance Tools

- **Basic OS tools**
  - time, gprof, strace
- **Hardware counters**
  - PAPI API & tool set
  - hwctime (AIX)
- **Sampling tools**
  - Typically unmodified binaries
  - Callstack analysis
  - HPCToolkit (Rice U.)
- **Profiling/direct measurements**
  - MPI or OpenMP profiles
  - mpiP (LLNL&ORNL)
  - ompP (LMU Munich)
- **Tracing tool kits**
  - Capture all MPI events
  - Present as timeline
  - Vampir (TU-Dresden)
  - Jumpshot (ANL)
- **Trace Analysis**
  - Profile and trace capture
  - Automatic (parallel) trace analysis
  - Kojak/Scalasca (JSC)
  - Paraver (BSC)
- **Integrated tool kits**
  - Typically profiling and tracing
  - Combined workflow
  - Typically GUI/some vis. support
  - Binary: Open|SpeedShop (Krell/TriLab)
  - Source: TAU (U. of Oregon)
- **Specialized tools/techniques**
  - Libra (LLNL)
    Load balance analysis
  - Boxfish (LLNL/Utah/Davis)
    3D visualization of torus networks
  - Rubik (LLNL)
    Node mapping on torus architectures
- **Vendor Tools**

Performance Tools are a Necessity !

- **Demand is increasing**
  - Traditionally little development time invested in optimization
  - Code teams are getting more interested tools
- **Widely researched in many projects and groups**
  - From specialized tools to comprehensive toolkits
  - Different data acquisition (sampling and/or tracing)
  - Different instrumentation options
  - Commercial and open source options



- **Example: Open|SpeedShop (http://www.openspeedshop.org/)**
  - Developed by the Krell Institute in close collaboration with ASC
  - Performance analysis tool framework
  - Support for sampling and tracing on unmodified binaries
  - Support for Linux as well as BG/P&Q and Cray X? machines

Typical Tool Workflow (looking at O|SS)

Application

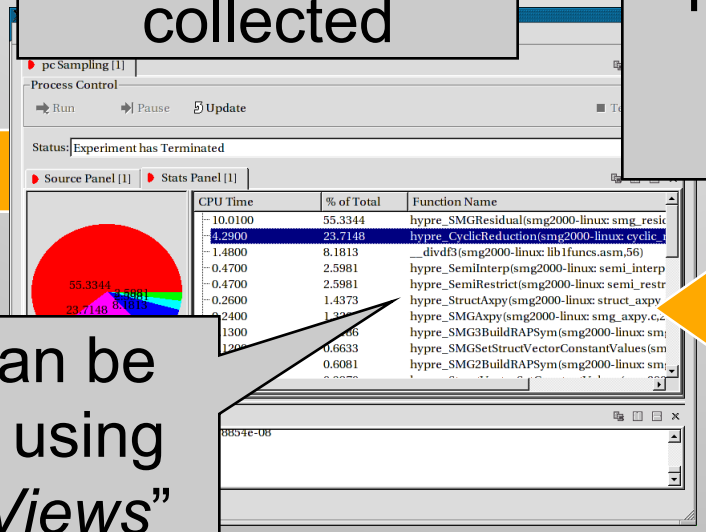
"Experiment"

Open | SspeedShop™

Run

Defines what and how data is collected

Process and Experiment Control



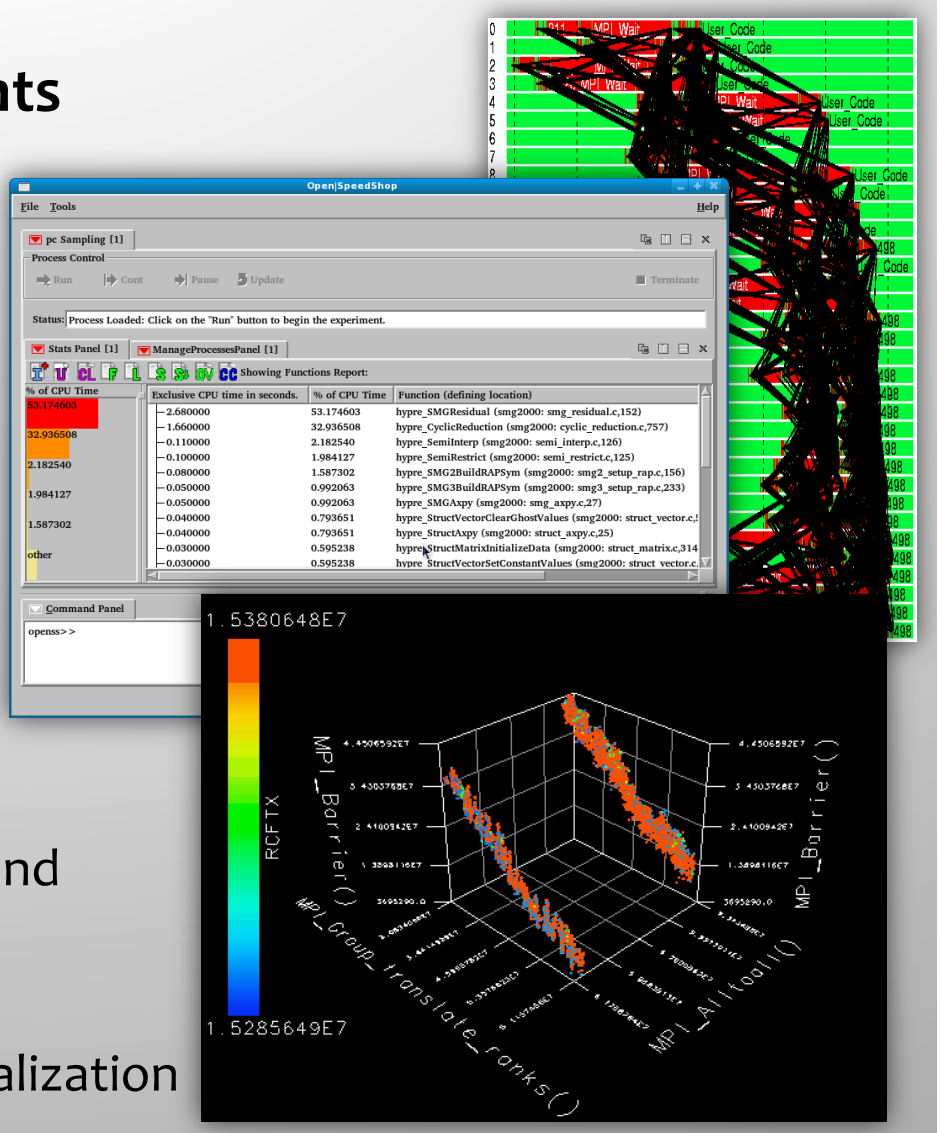
Results

Results can be displayed using several "*Views*"

Stored in a database

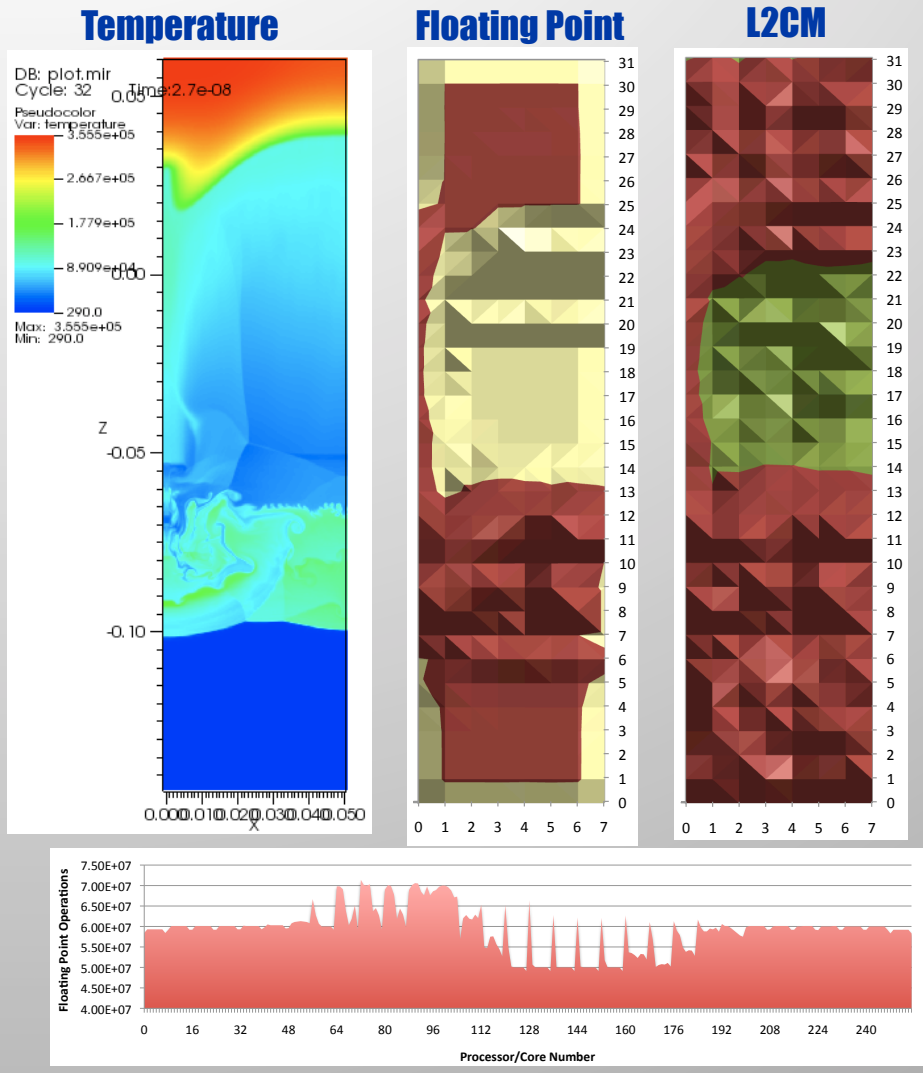
Existing Tools Enable Sophisticated Measurements

- **Large variety of measurements**
  - Attribution to source code
  - Multi-metric visualizations
- **Some tools provide analysis**
  - Scalasca detects trace patterns
  - Expert systems like PerfExpert
  - Limited to previously identified issues
- **Information often low level**
  - ➢ Need user's perspective
    - Developers need to understand their codes
    - Mapping to user's domain
  - ➢ Needed: Intuitive analysis & visualization

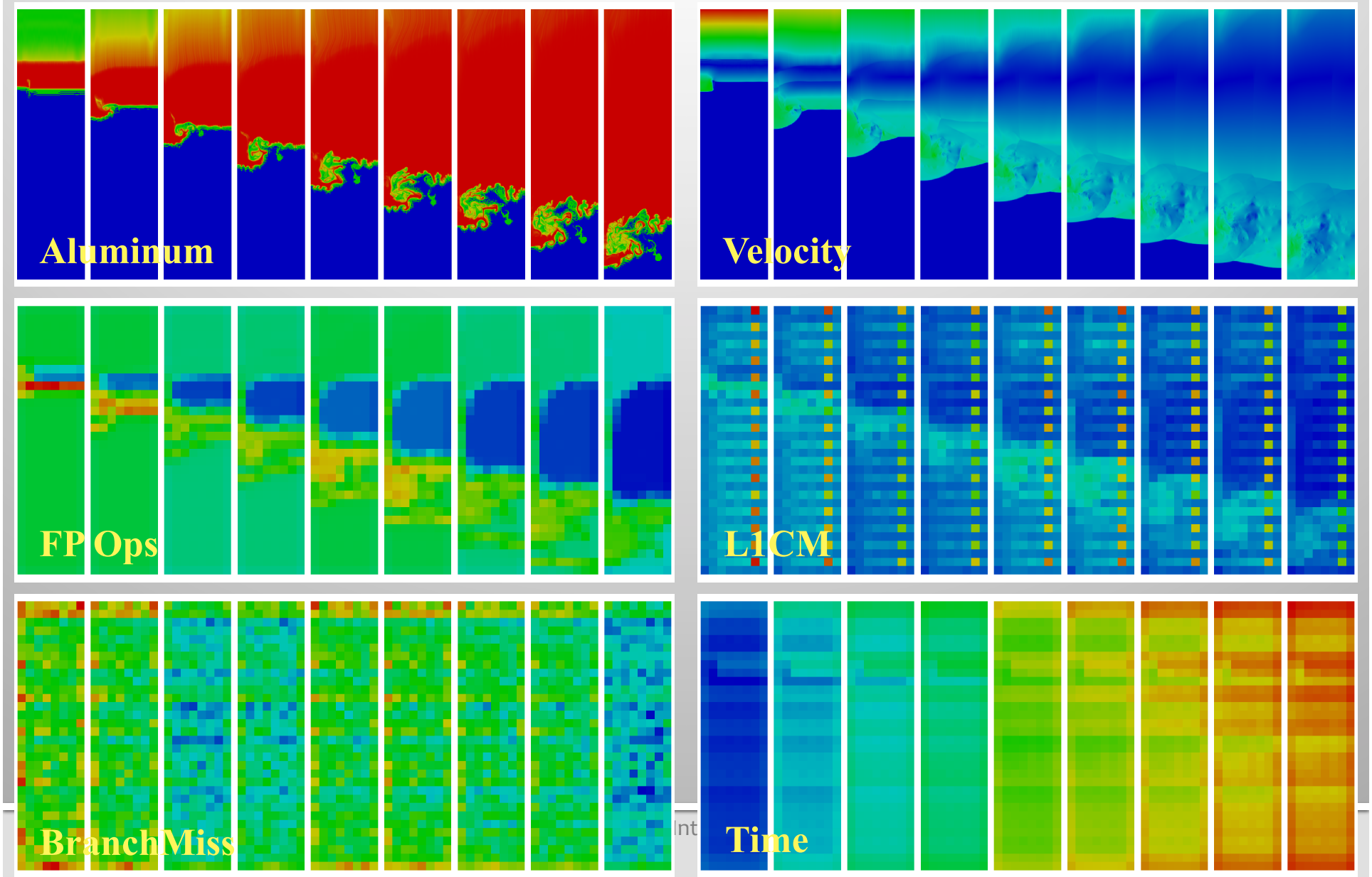


Bridging the Gap to the Application Domain

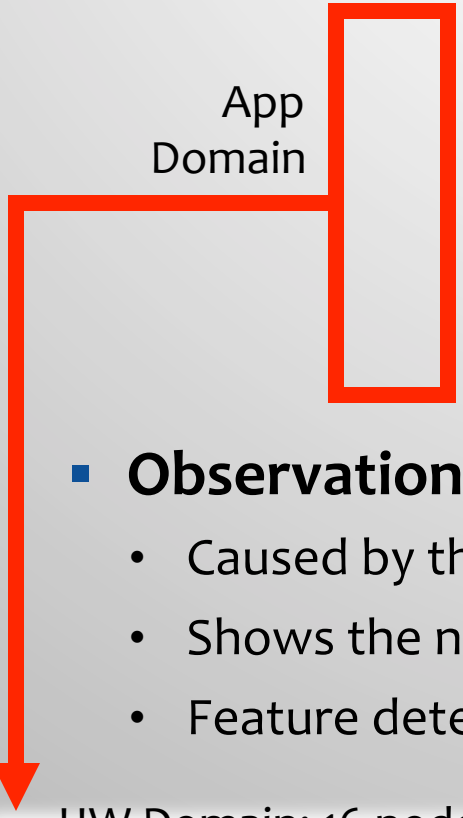
- **Example: 256 core run of a CFD application**
  - Floating point operations
- **Application developers think in the app domain**
- **Simple step:**
  - Map floating point ops onto the application domain
  - Similar L2 cache misses
- **Clear correlations**
  - Explains performance
  - Helps establish a baseline



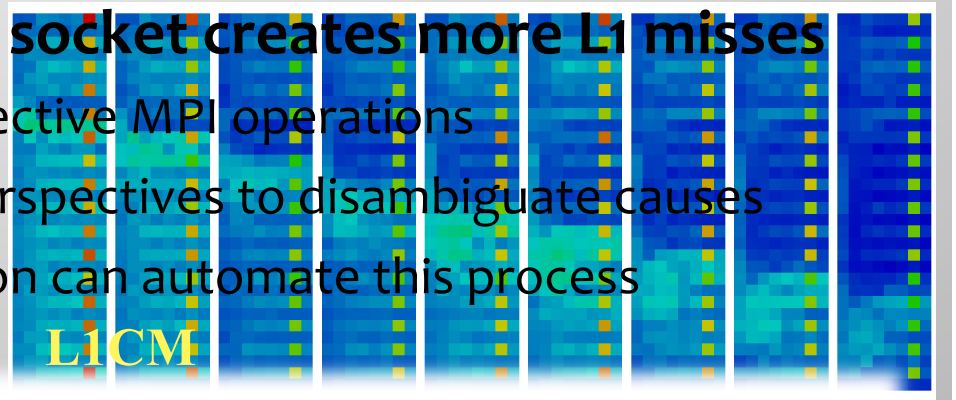
Mapping Measurements into the Application Domain



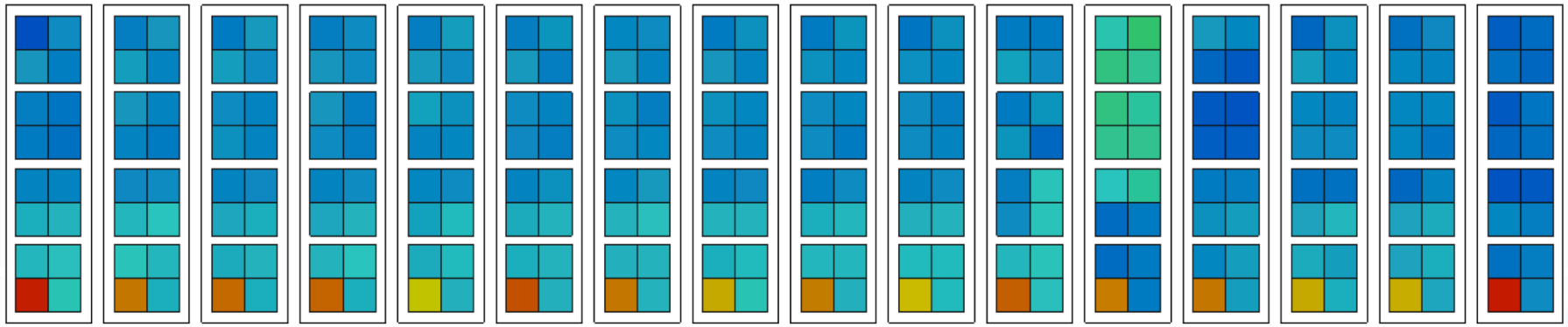
Multiple Views Can Help Disambiguate Effects



- **Observation: single core per socket creates more L1 misses**
  - Caused by the execution of collective MPI operations
  - Shows the need for different perspectives to disambiguate causes
  - Feature detection and correlation can automate this process

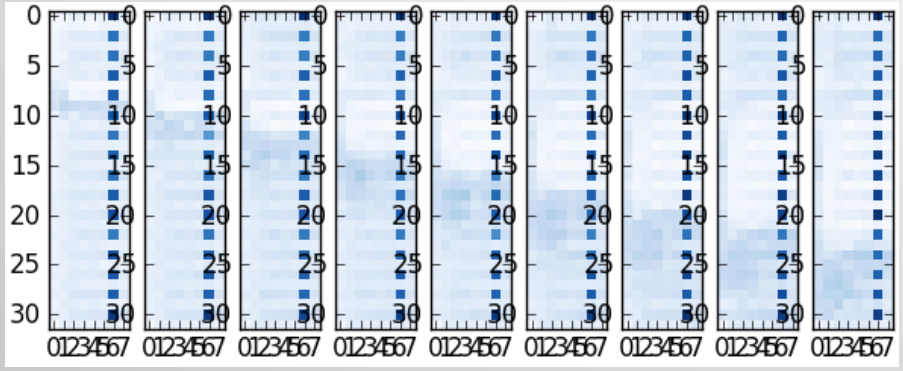


HW Domain: 16 nodes with 4x4 cores

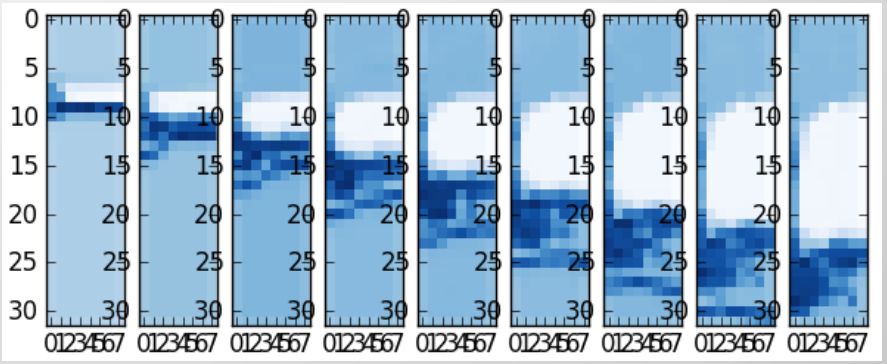


Feature Detection and Isolation

Same data with linear color map



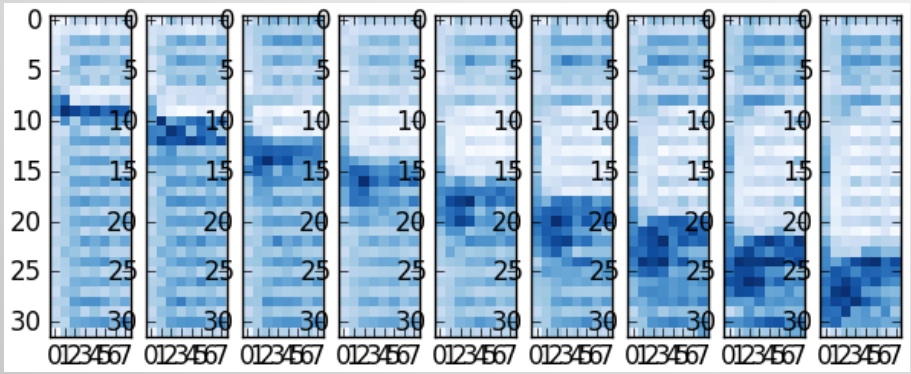
L1 Cache Misses



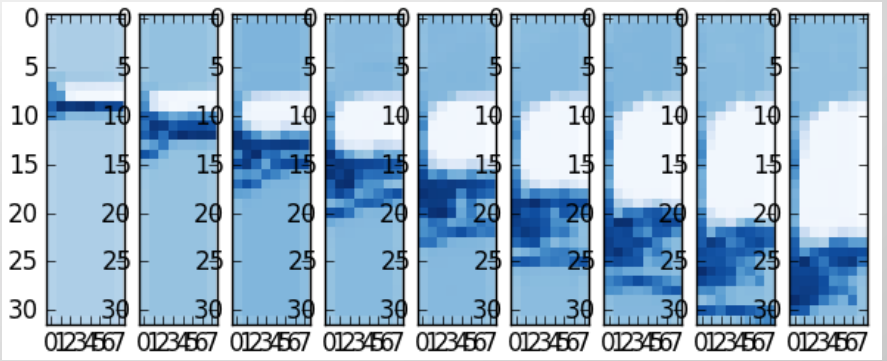
FP Operations

Feature Detection and Isolation

Same data with linear color map



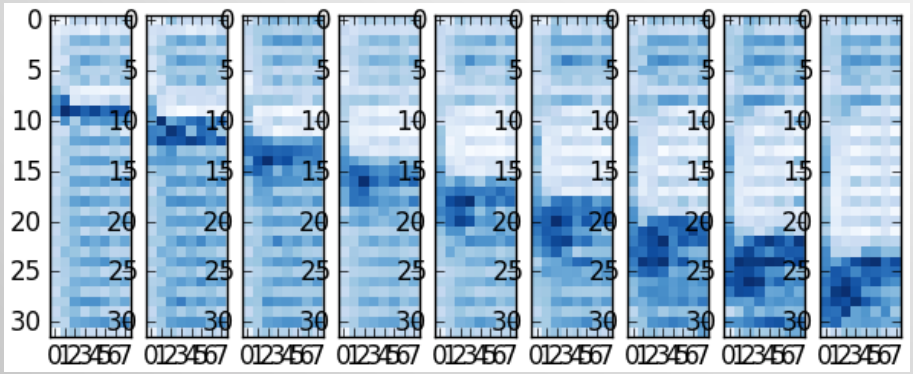
L1 Cache Misses with MPI worker filtered



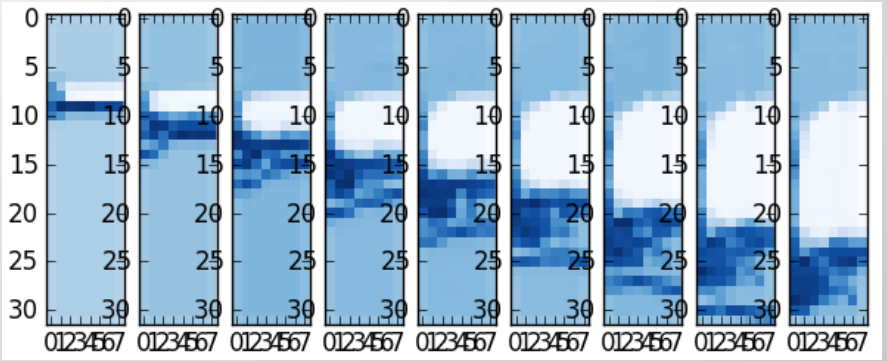
FP Operations

Feature Detection and Isolation

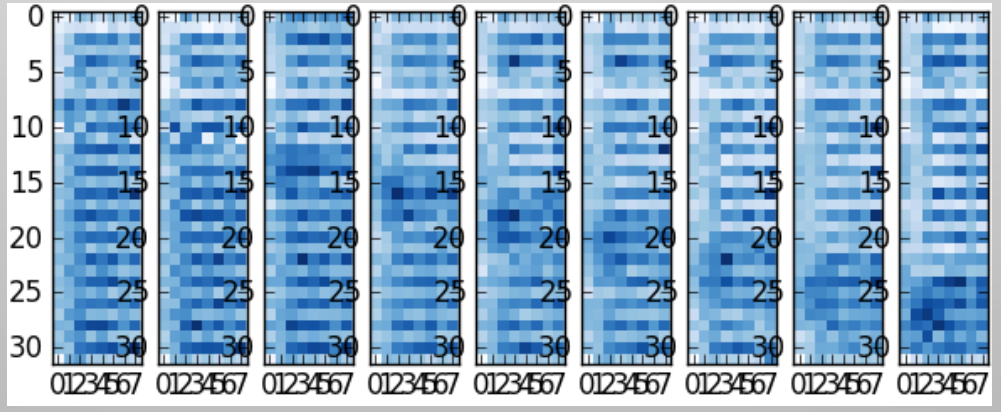
Same data with linear color map



L1 Cache Misses with MPI worker filtered



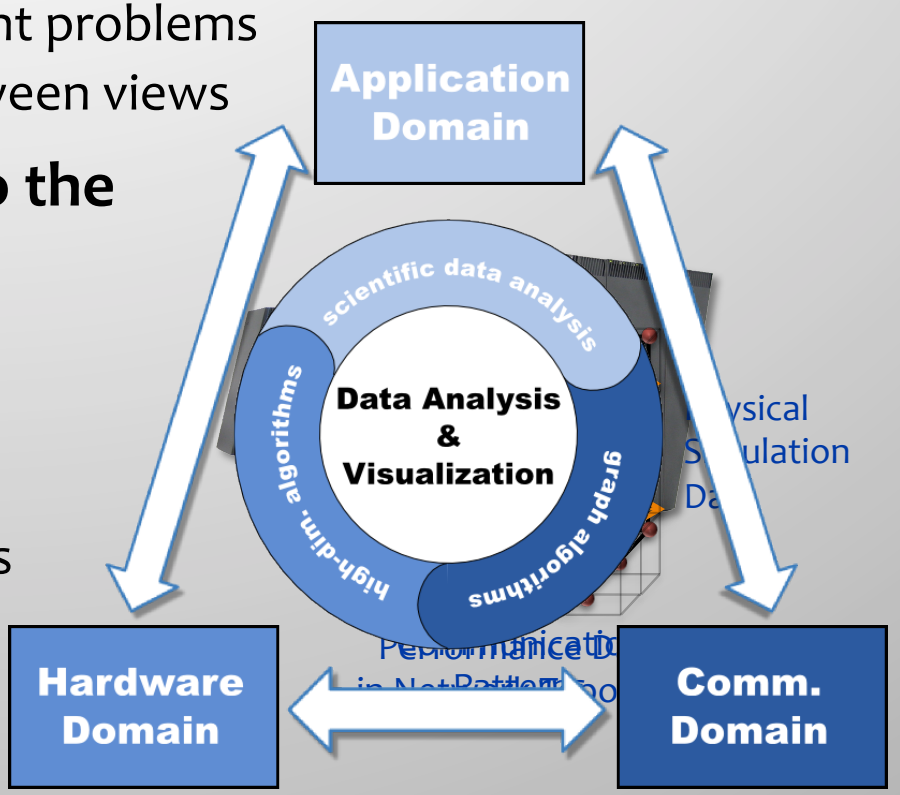
FP Operations



L1 Misses per FP operation: Proxy for efficiency

Correlating Performance Domains

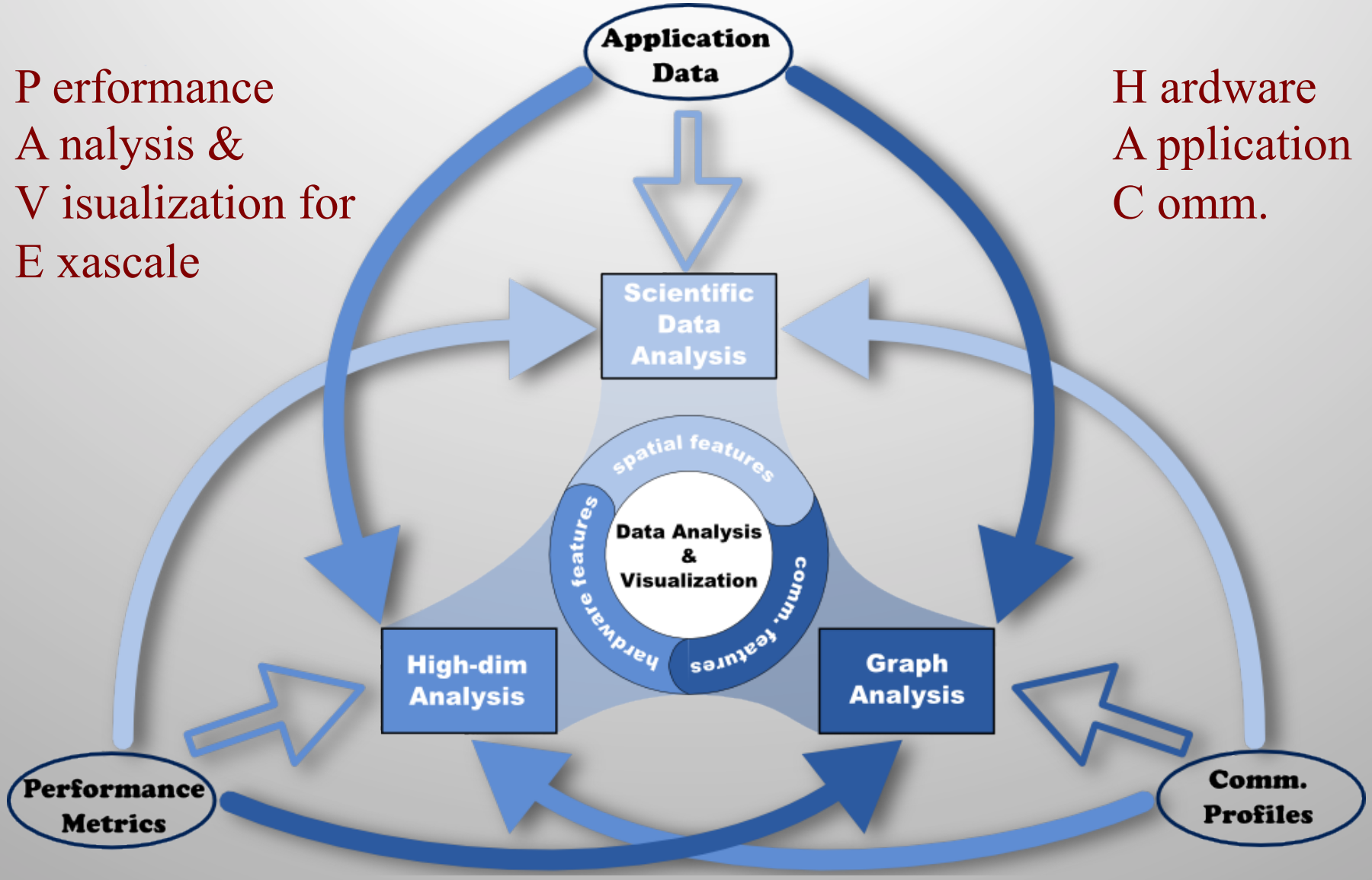
- **Single view on data is insufficient**
  - Different perspectives for different problems
  - Need to support correlation between views
- **Map data from one domain to the one of the other domains**
  - Comparable data
  - Enable correlation
  - Understand interactions
  - Access to visualization techniques
- **Increase intuition for users**
  - Display data in domains familiar to users
  - Make abstract measurements tangible



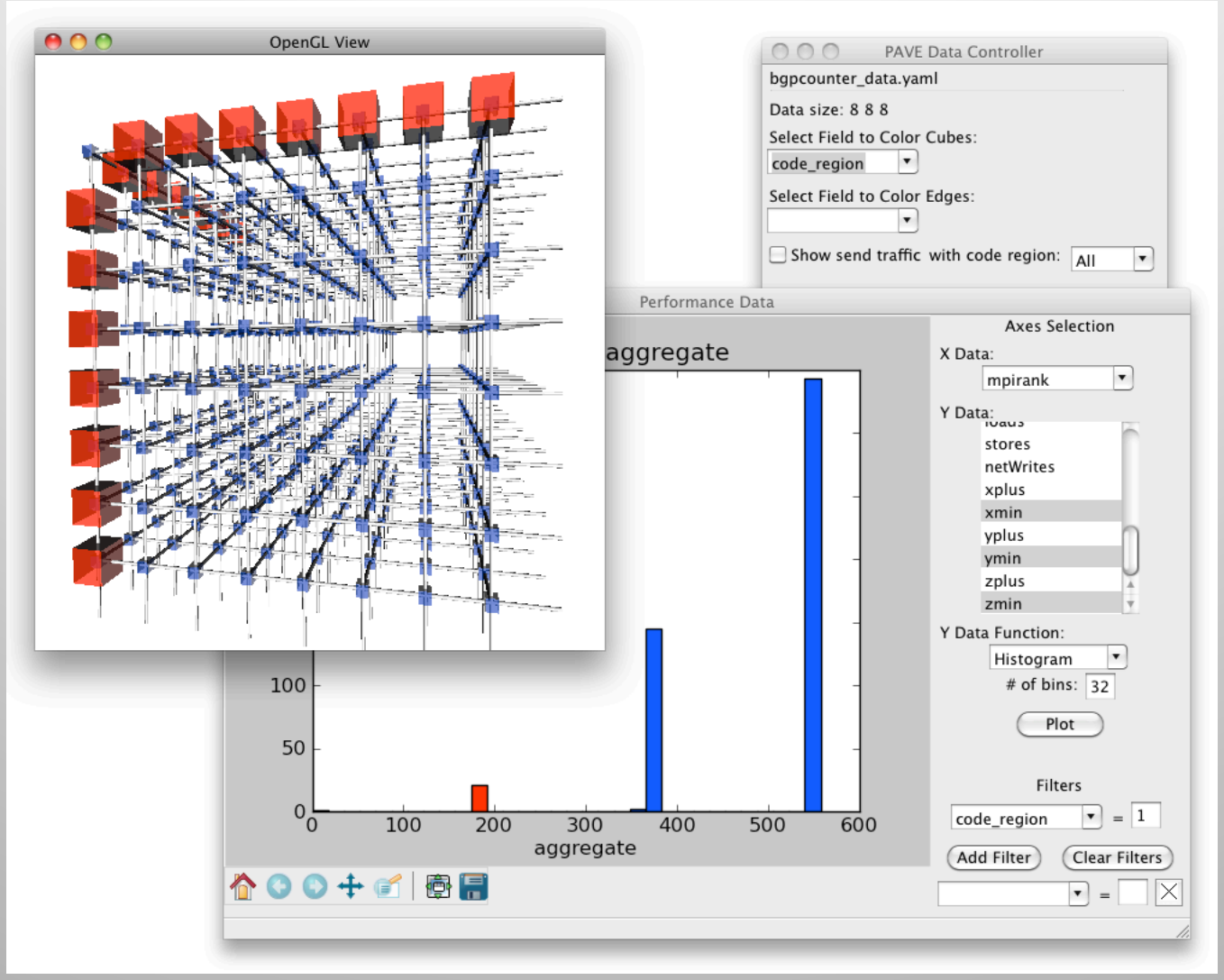
The PAVE/HAC Model

P erformance
A nalysis &
V isualization for
E xascale

H ardware
A pplication
C omm.

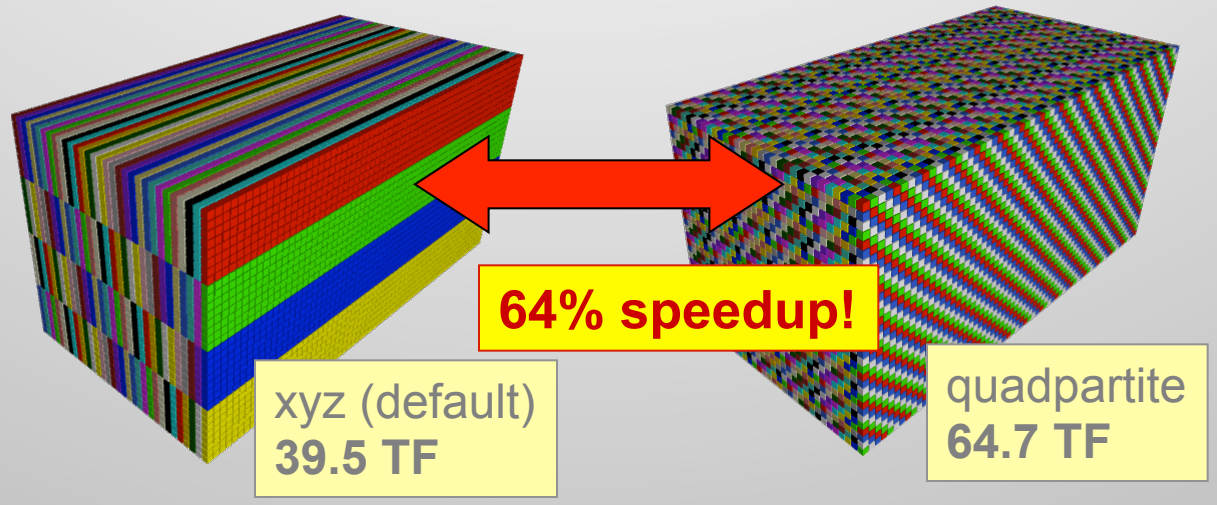


Boxfish: Interactively Visualizing Across Domains



Target: Optimize Node Mappings

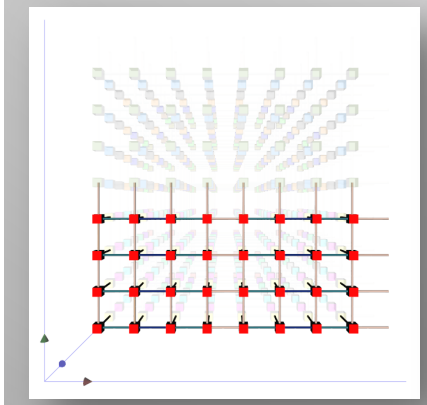
- **Network topologies getting more complex**
  - Interactions with communication topology non-trivial
  - Node placement has huge impact on performance



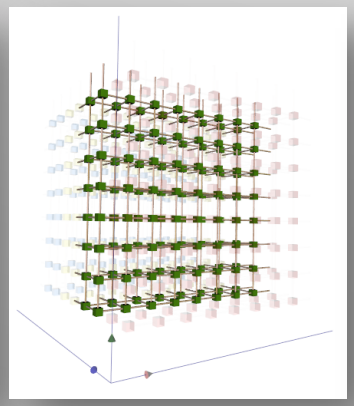
- **Require tools to help with defining and evaluating layouts**
  - Easier specification and visualization of layouts
  - Capture and compare network traffic

Comparing Communicator Layouts with the Boxfish Tool

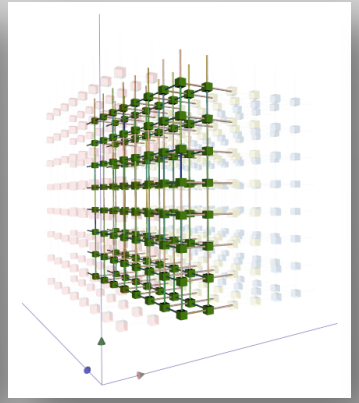
- **Mapping a single plane into a torus**
  - Multiple options (through BG/P mapfiles)
  - Combination of mapping and tilting
- **Layouts can get complicated and need to be visualized**
  - Boxfish can be used to visually confirm mappings
  - Example: single X/Y plane of a 3D problem
- **3D view can be tricky to visualize link utilization**



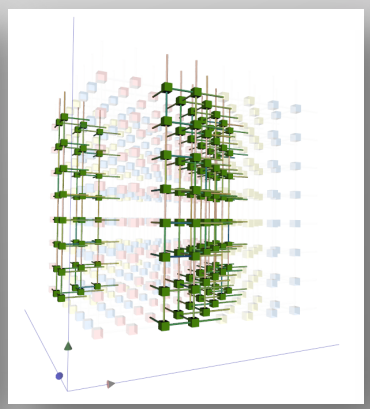
TXYZ



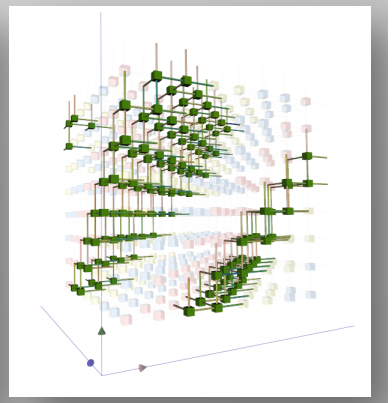
XYZT



Tile



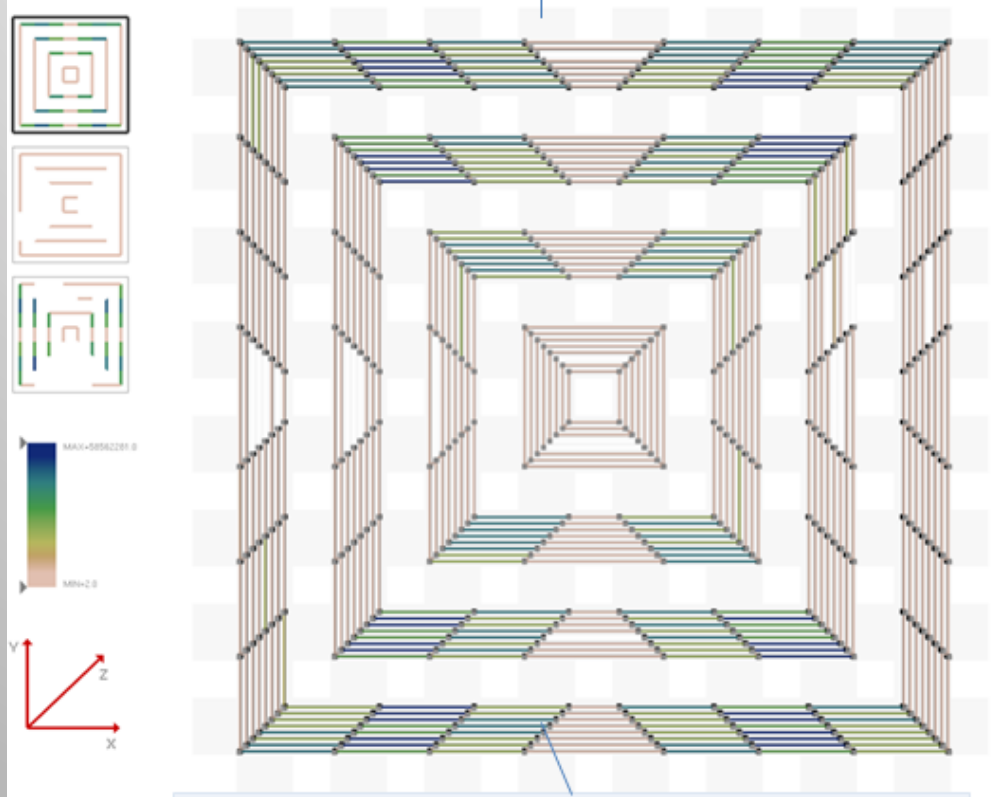
Tilt Z



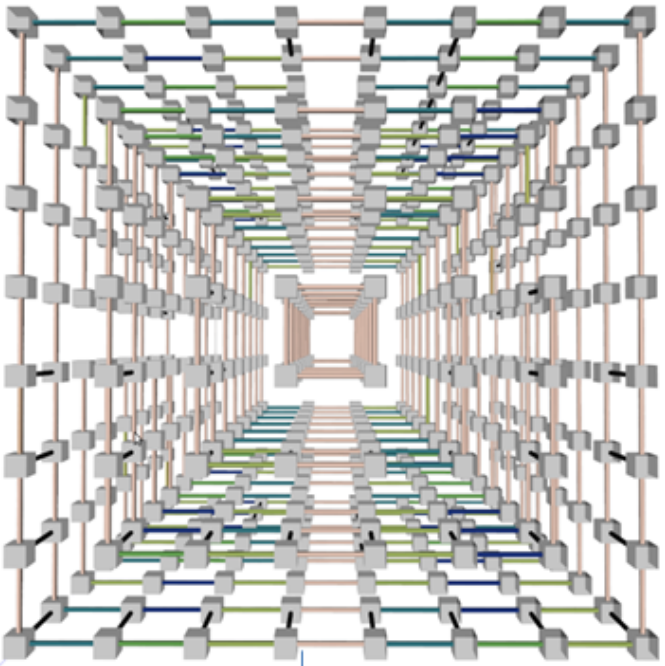
Tilt ZY

Flattening Network Traffic to 2D

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



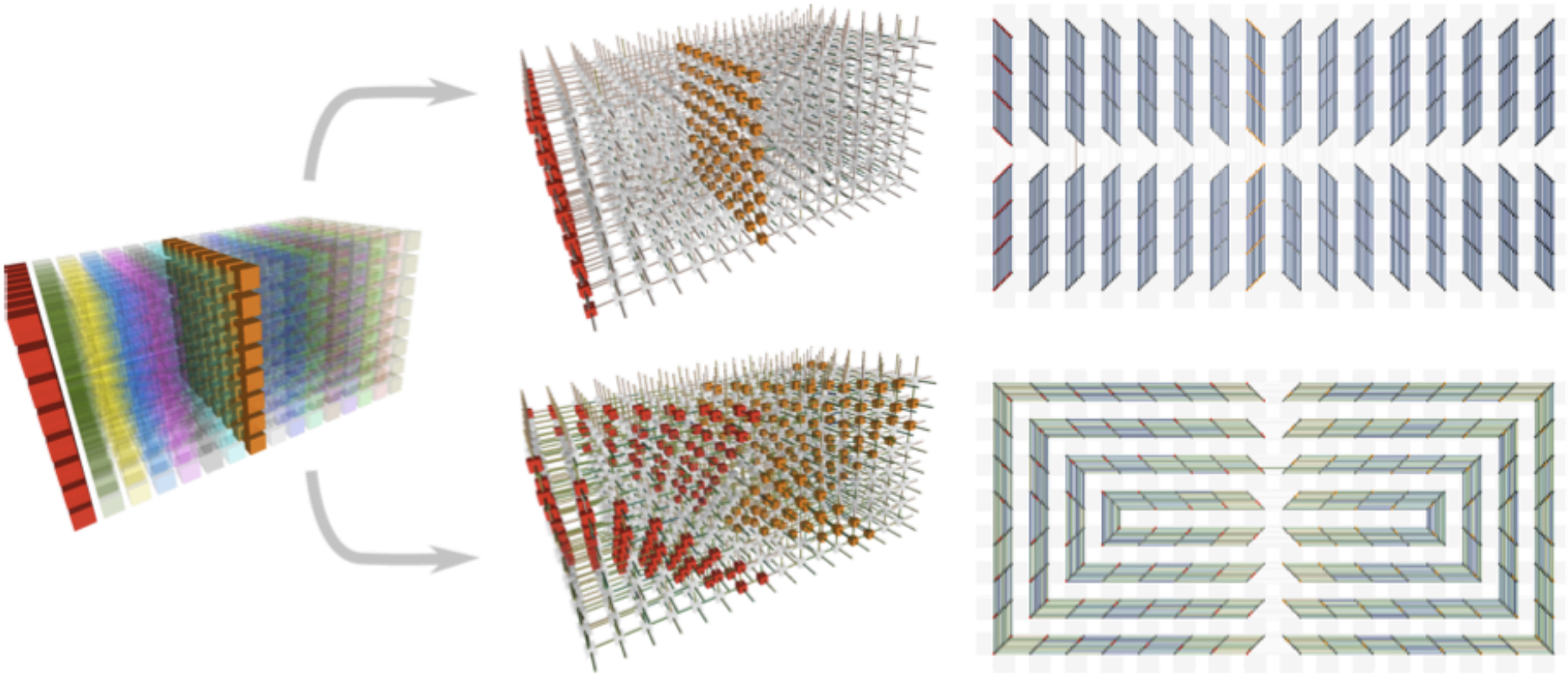
The 2D view can display all nodes without any occlusion. Only half of X links and half of Y links and all of Z links are shown. The Z links are along the diagonals.



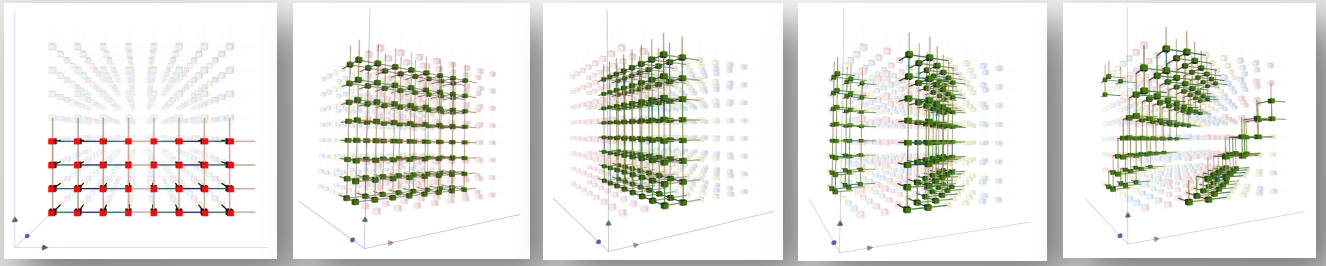
3D view supports interactions like **zooming, rotation and translation.**

Views into an LLNL Laser Code

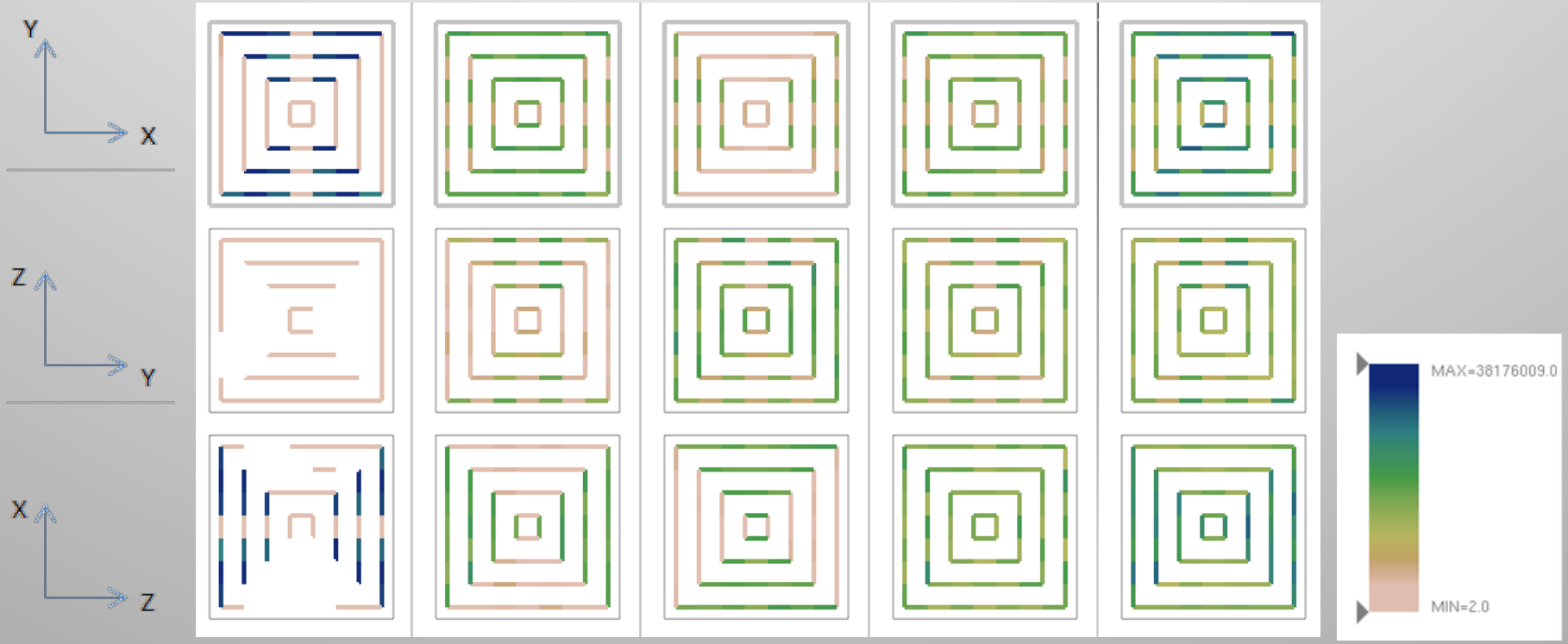
- **Problems setup as a series of 2D slabs**
  - During each step: X/Y phases within a slab
  - Looking at performance for each step



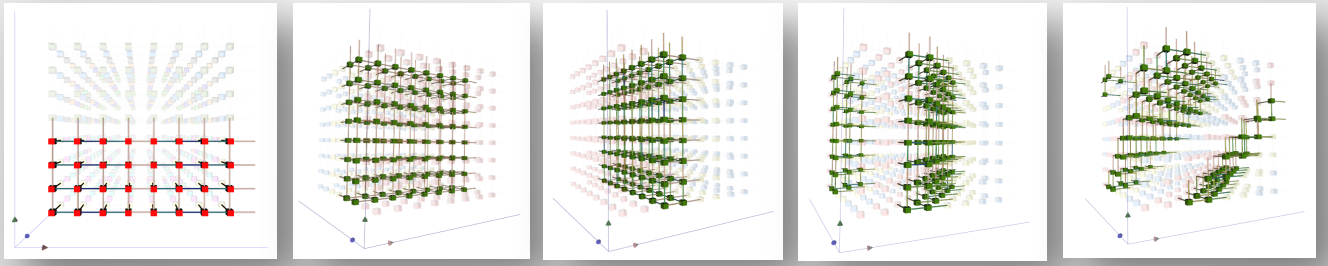
Boxfish's 2D mini-maps Summarize Bandwidth (x phase)



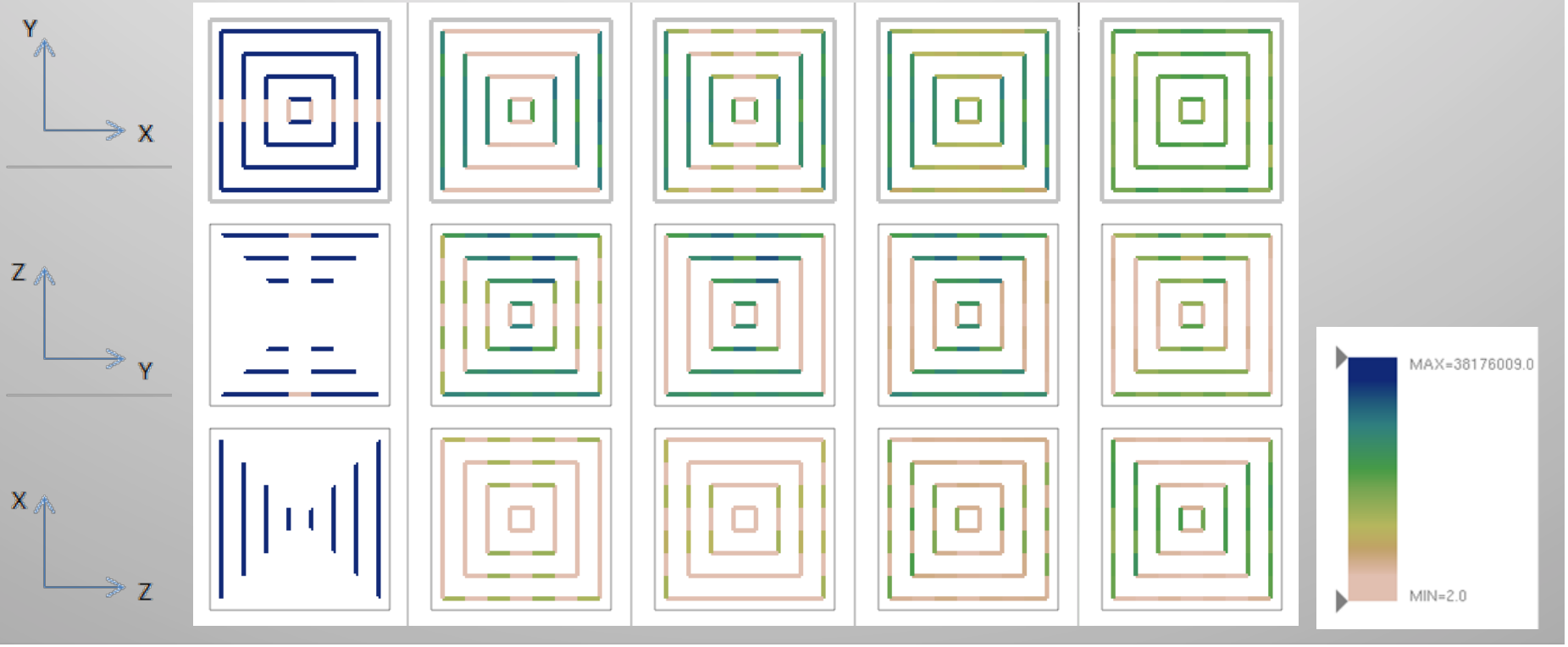
TXYZ	XYZT	Tile	Tilt Z	Tilt ZY
55 MB/s	129 MB/s	131 MB/s	174 MB/s	201 MB/s



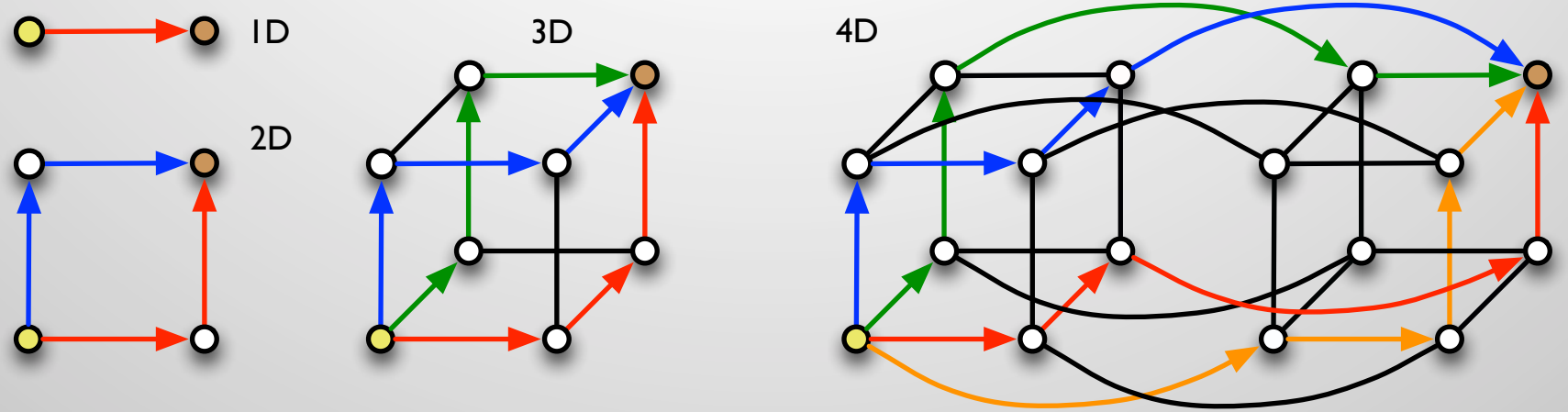
Boxfish's 2D mini-maps Summarize Bandwidth (y phase)



TXYZ	XYZT	Tile	Tilt Z	Tilt ZY
55 MB/s	129 MB/s	131 MB/s	174 MB/s	201 MB/s



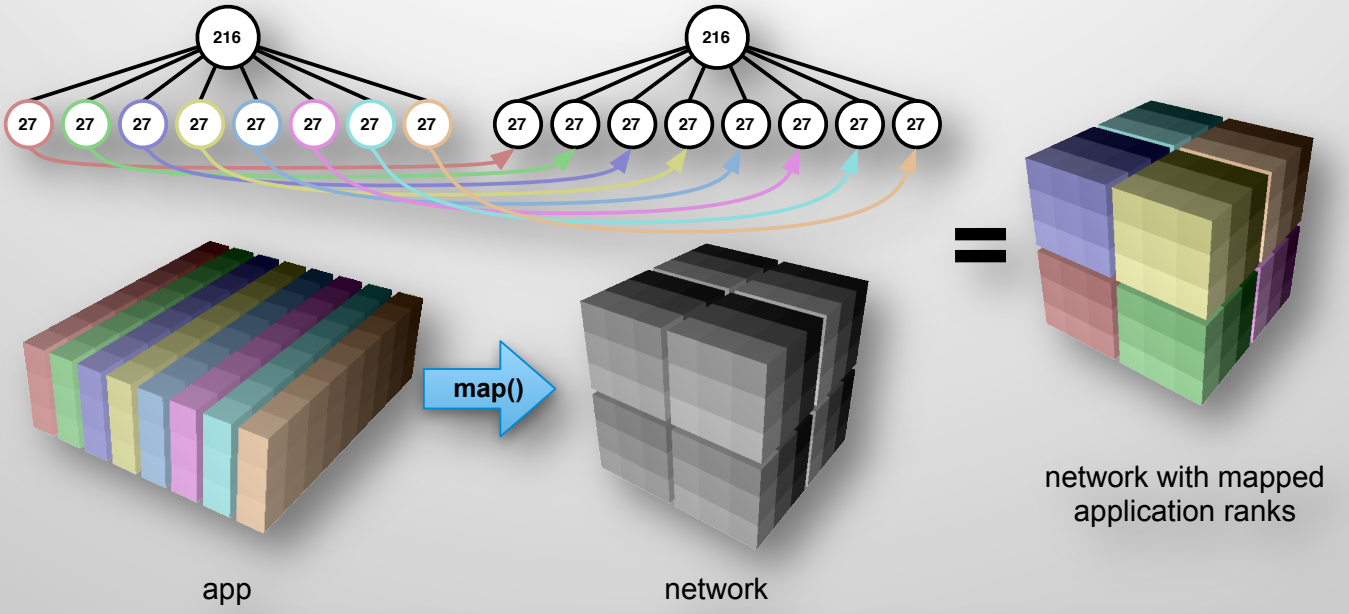
Utilizing the Full Capacity of the Torus



Black links are “spare” links that can handle extra traffic that comes through the cube.

- **Dimension independent transformations/tilting**
  - Tilting optimization allows higher bandwidth on torus links
  - Tilting is easily extended into higher dimensions (5D, etc.)

Rubik: Easy generation of BG mapping files



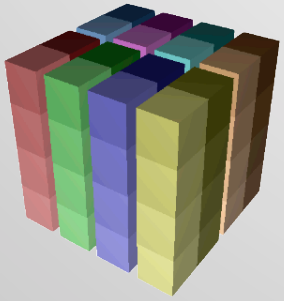

```
# Create app partition tree of 27-task planes
app = box([9,3,8])
app.tile([9,3,1])

# Create network partition tree of 27-task cubes
network = box([6,6,6])
network.tile([3,3,3])

network.map(app)   # Map plane tasks into cubes
```

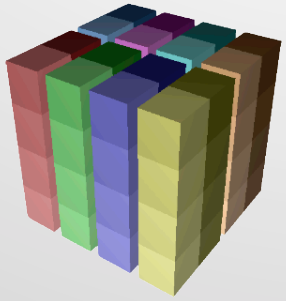
Additional Rubik Operations

div



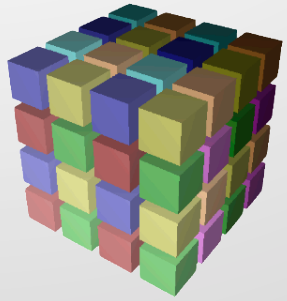
```
app = box([4,4,4])
app.div([2,1,4])
```

tile



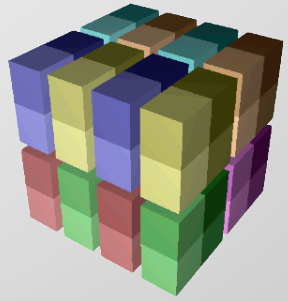
```
app = box([4,4,4])
app.tile([2,4,1])
```

mod



```
app = box([4,4,4])
app.mod([2,2,2])
```

cut

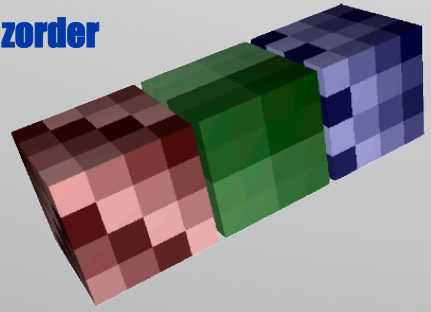


```
app = box([4,4,4])
app.cut([2,2,2],
        [div,div,mod])
```

zigzag

zorder

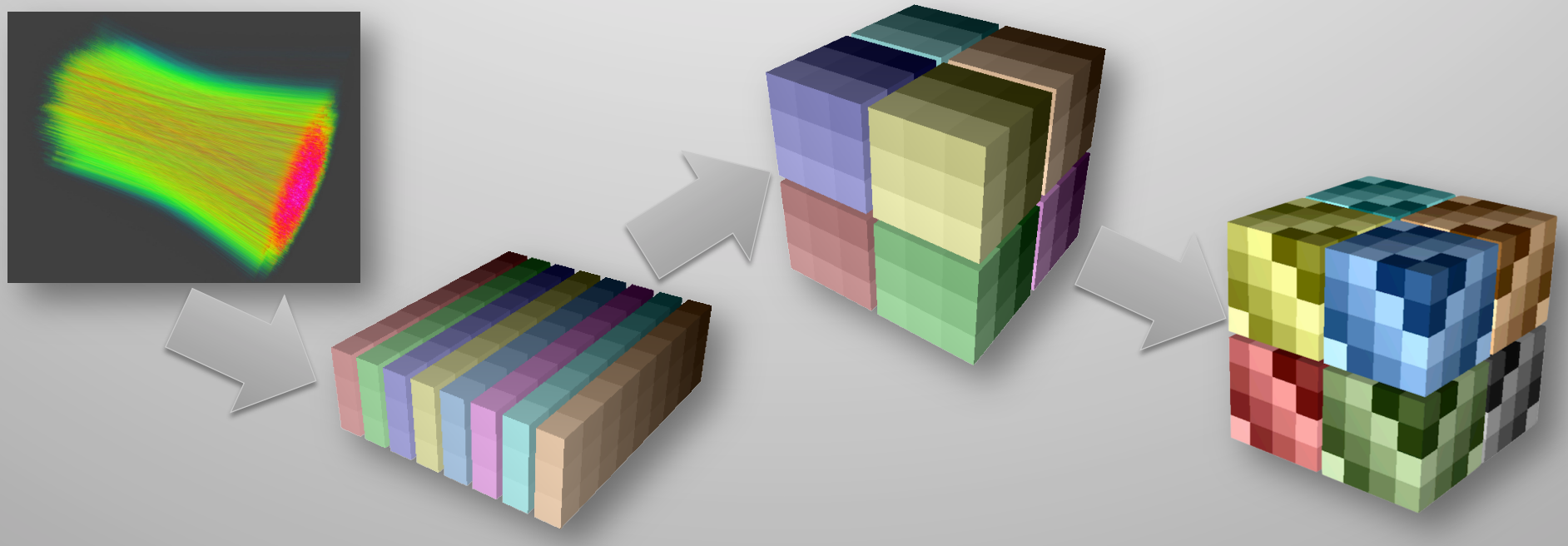
tilt



```
Z, Y, X = 0, 1, 2
net = box([12,4,4])
net.div([3,1,1])
net[0,0,0].tilt(Z,X,1)
net[0,0,0].tilt(X,Y,1)
net[1,0,0].zorder()
net[2,0,0].zigzag(Z,X,1)
net[2,0,0].zigzag(X,Y,1)
```

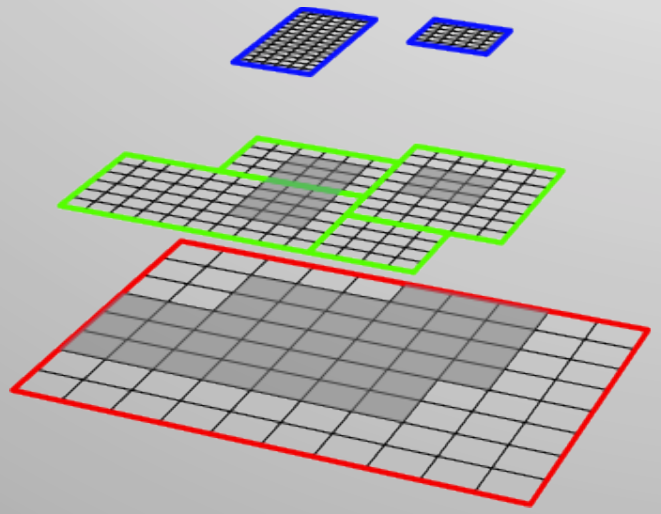
Mappings for the Laser Code

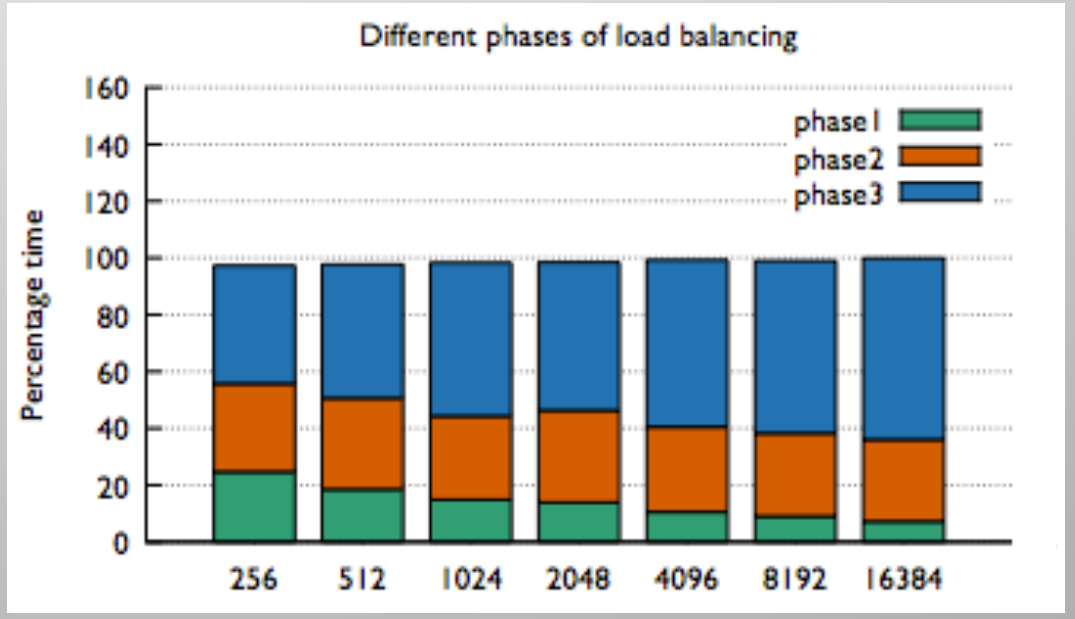
- Improved bandwidth from **50 MB/s to over 201 MB/s**
- Can be implemented as a single, short Python script
- Integrated visualization of mappings



Optimizing Load Balancing in AMR

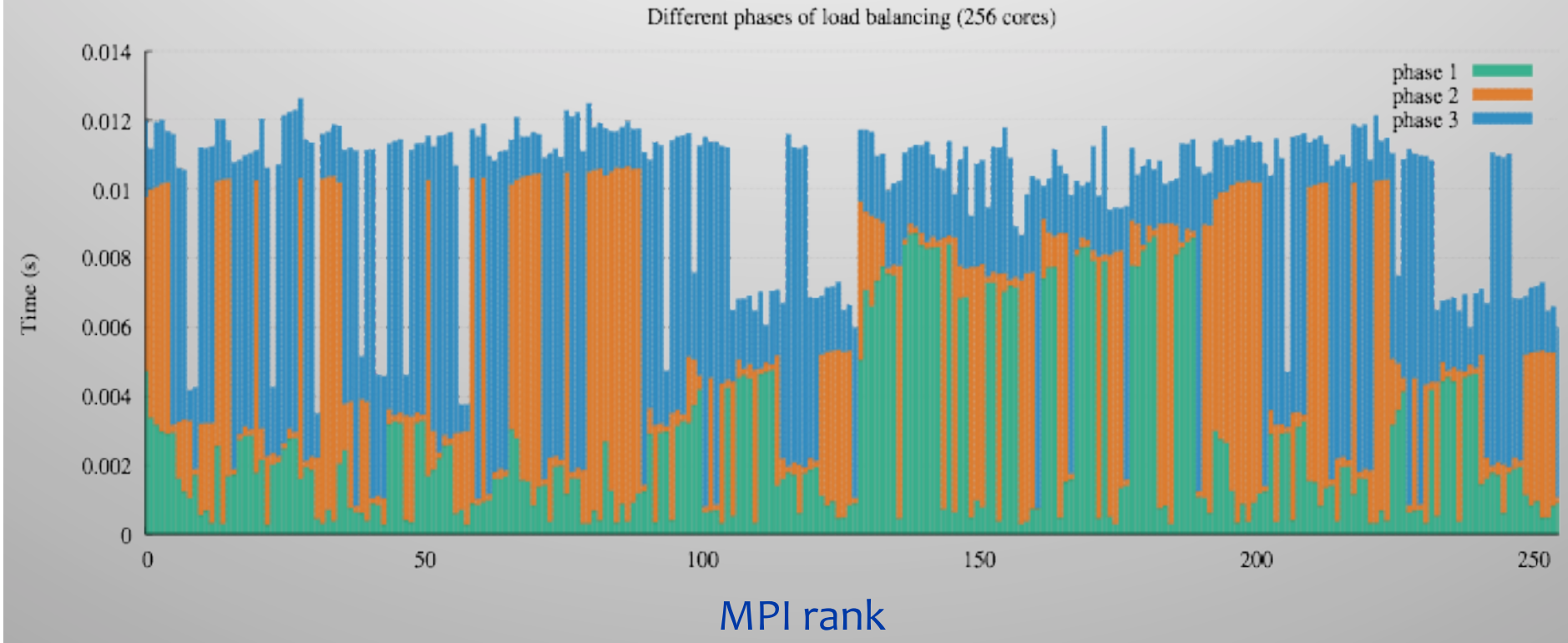
- Adaptive Mesh Refinement (SAMRAI library)
  - Different levels of patches to refine in areas of interest
  - Requires active load balancing
  - Load balancing shows bad scaling behavior
  - Dominates at large scale





Attempt 2: Timings in MPI rank space

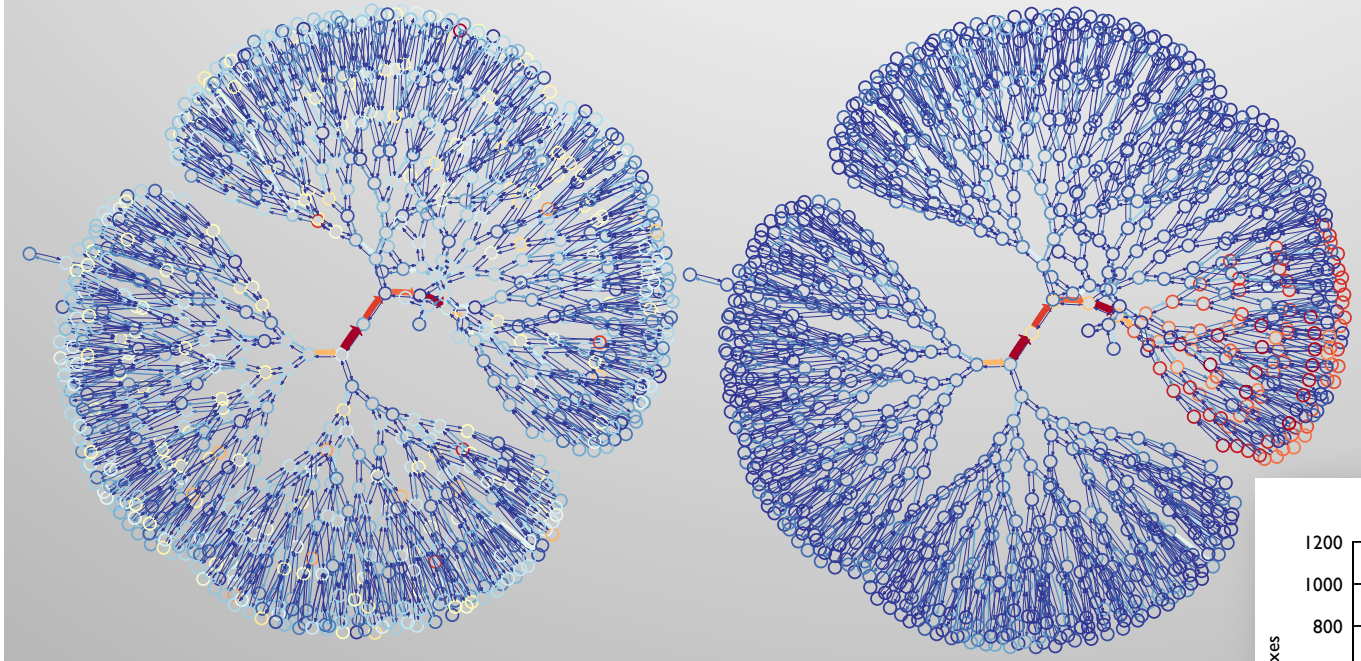
- **Per node timings for each phase**
  - Bottleneck is in phase 1 and not phase 3
  - Limited correlation based on rank space

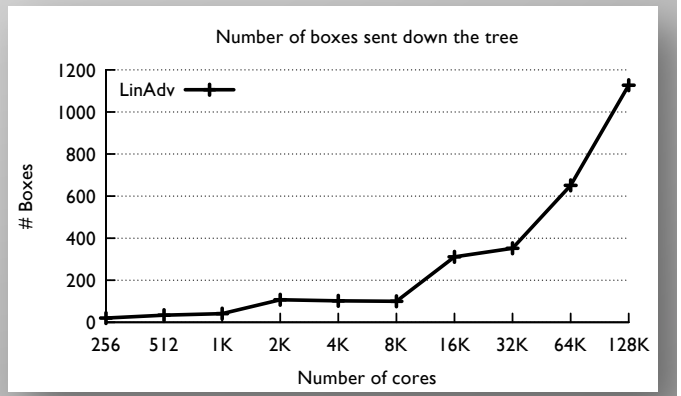


Alternative: Map Performance Metrics onto Underlying Communication Graph (1024 processes)

Load (Cells per process)
Before balancing

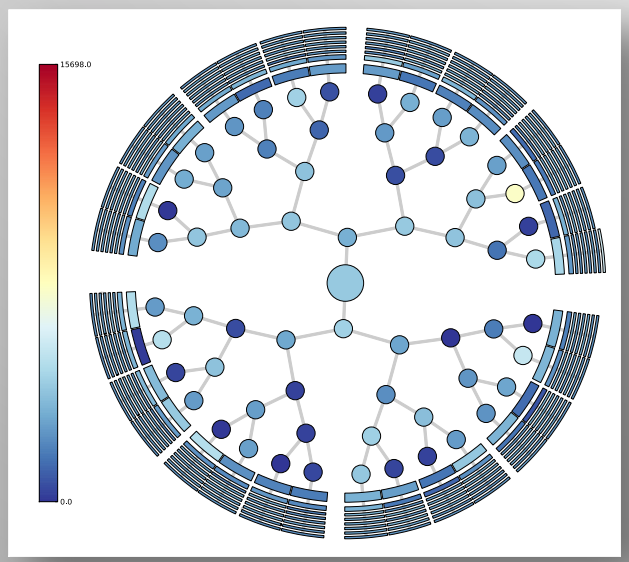
Time spent redistributing box information in the tree



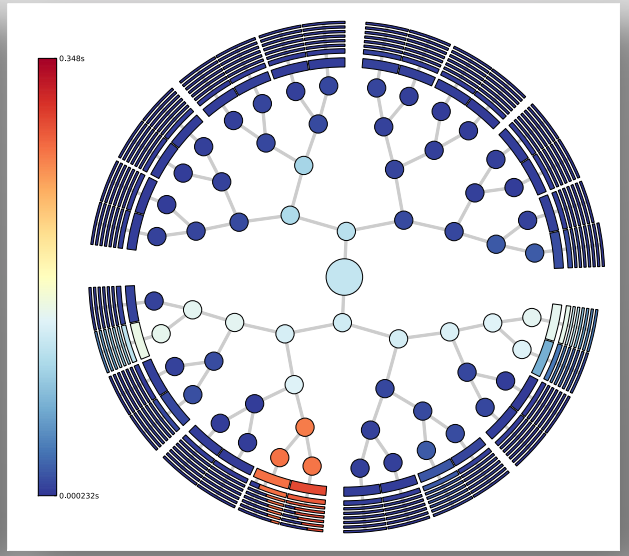


Visualizing Large Communication Graphs

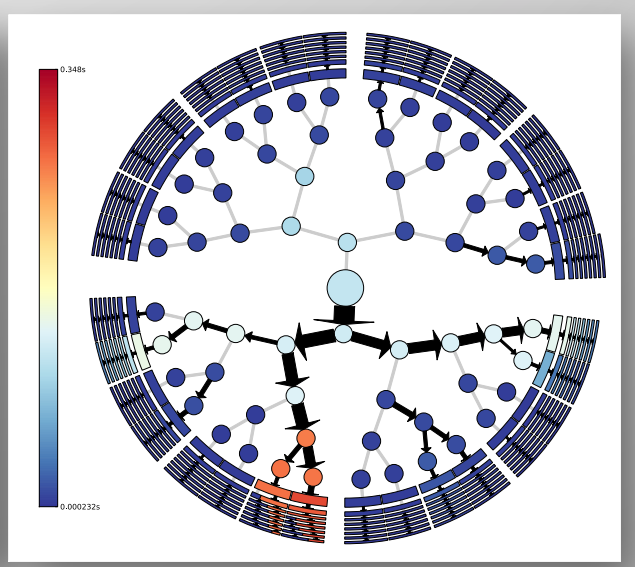
- **Display of individual nodes is not scalable**
  - Need to group nodes
  - Outer layers are best targets for this
  - Keep metric information / coloring



Load on 16k cores



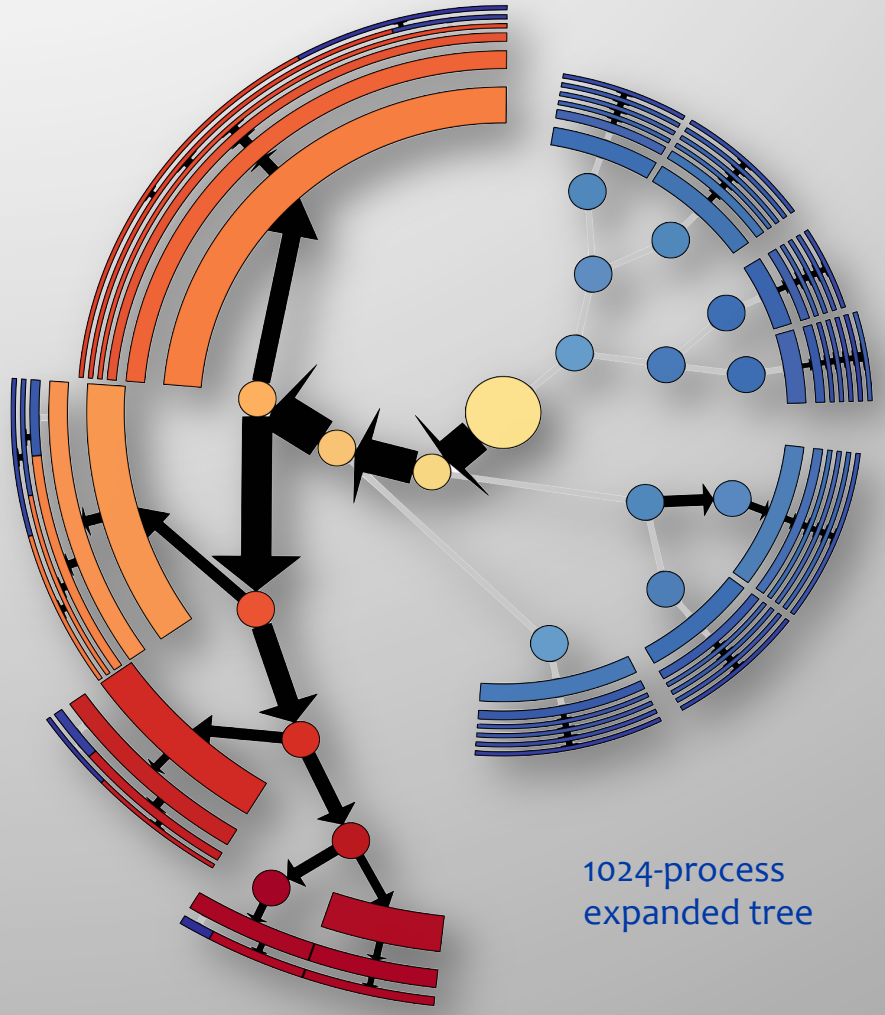
Wait time for box distribution



Wait time with flow information

Highlighting Areas of Interest in Deep Trees

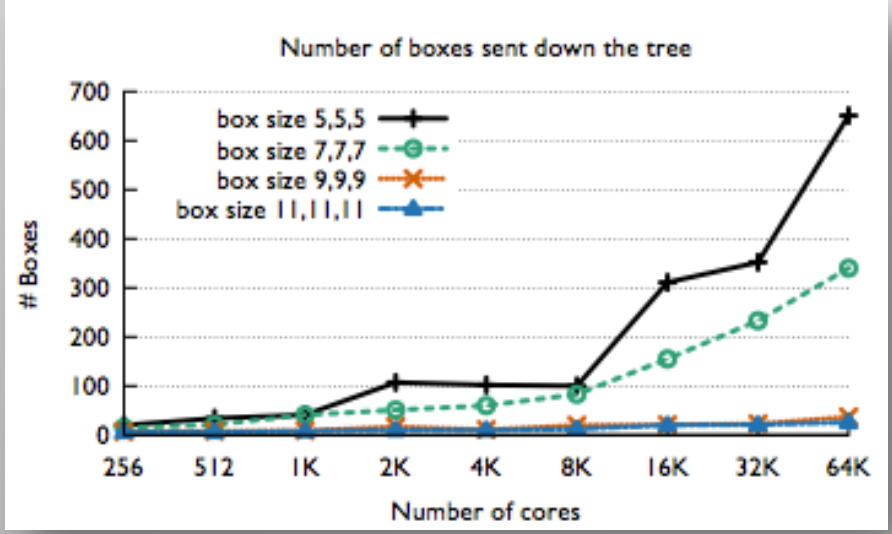
- Heavier trees are expanded to a deeper level
- Angles are apportioned by flow in the subtree
- Can see flow problems at any level of the tree

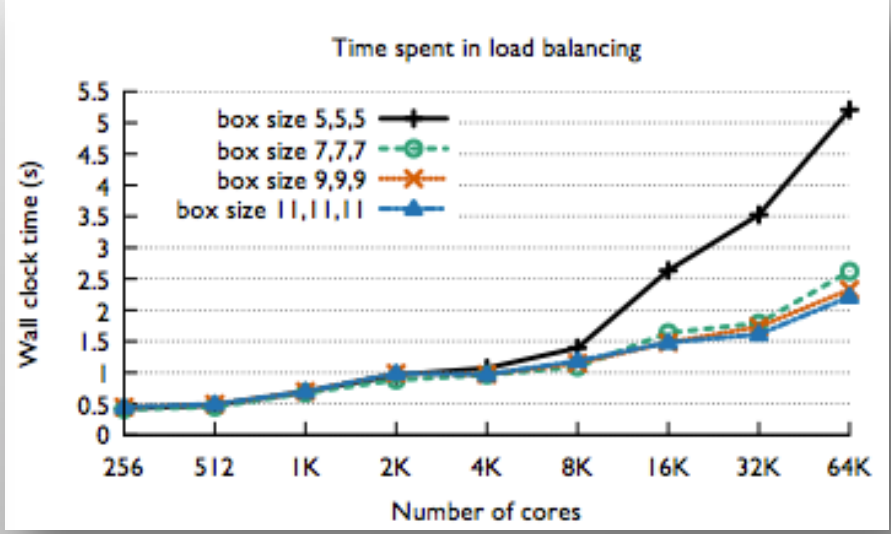


1024-process expanded tree

Performance Improvements

- **Need to address flow problem**
  - Reduce traffic through root
  - Box size / granularity is one of the knobs

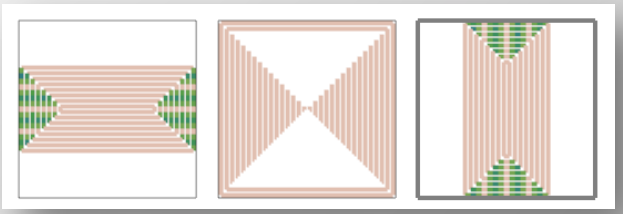


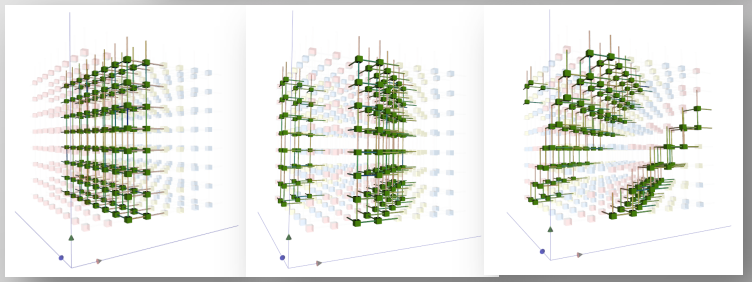


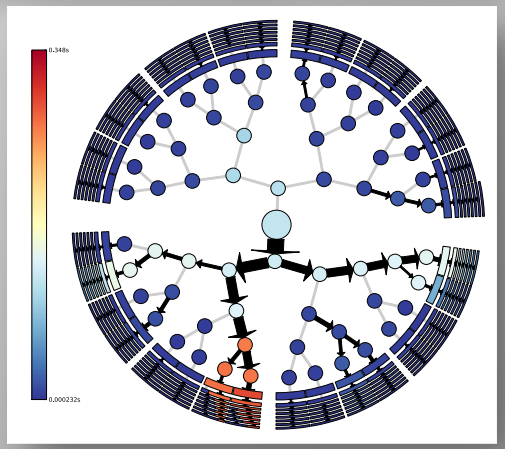
- **Ultimately need new communication/balancing algorithm**
  - Spread out load over multiple trees
  - Leads to a fat forest communication structure

Large Scale Visualization Can Help Performance Analysis

- **Tool support will be essential to exploit future machines**
  - Complex applications and architectures
  - Need intuitive insight for developers
- **Node mapping optimizations**
  - Compare different mappings
  - Boxfish minimaps
- **Optimizing AMR**
  - Map performance to underlying graph
  - Scaling imbalance visualizations
- **New generation of tools**
  - **More intuitive tools that help our users to understand performance at scale**
  - **Large scale visualization as driving instrument**





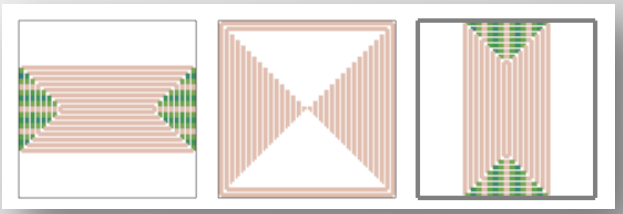


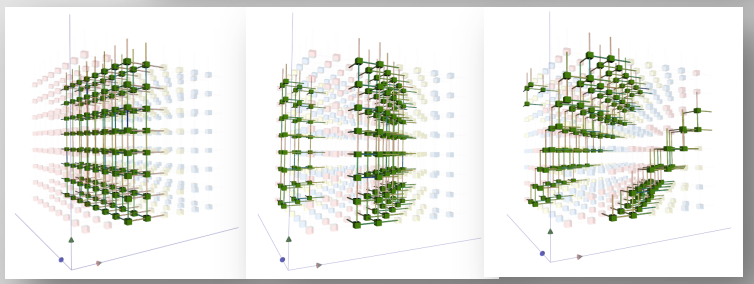
The PAVE Team

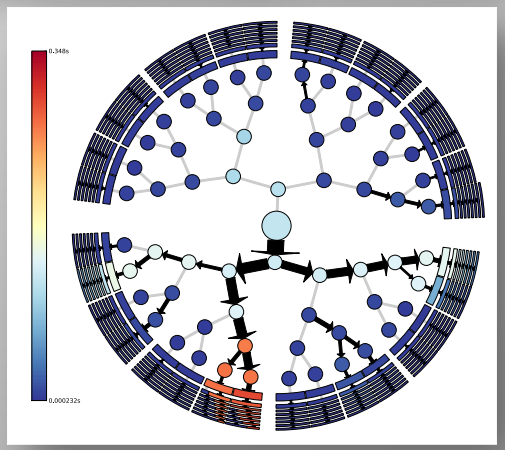
- Lawrence Livermore National Laboratory
  - Ahbinav Bhatele, Peer-Timo Bremer, Todd Gamblin, Nikhil Jain (UIUC), Martin Schulz
- University of Utah / SCI Institute
  - Aaditya Landge, Valerio Pascucci
- University of California Davis
  - Bernd Hamann, Kate Isaacs
- Clemson University
  - Joshua Levine

Large Scale Visualization Can Help Performance Analysis

- **Tool support will be essential to exploit future machines**
  - Complex applications and architectures
  - Need intuitive insight for developers
- **Node mapping optimizations**
  - Compare different mappings
  - Boxfish minimaps
- **Optimizing AMR**
  - Map performance to underlying graph
  - Scaling imbalance visualizations
- **New generation of tools**
  - **More intuitive tools that help our users to understand performance at scale**
  - **Large scale visualization as driving instrument**







http://scalability.llnl.gov/