U.S. DEPARTMENT OF ENERGY

PIPER

Performance Insights for Programmers and Exascale Runtimes

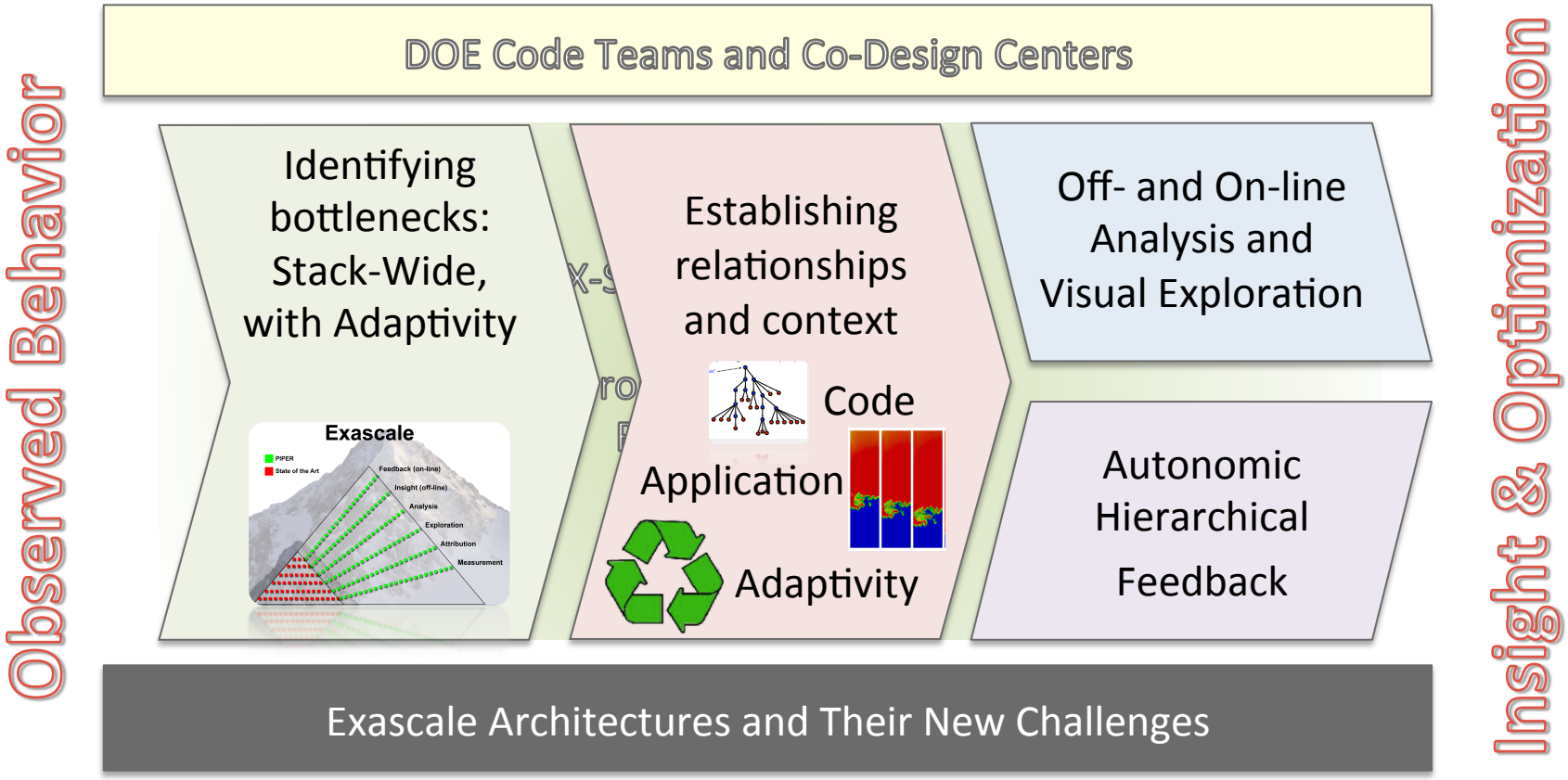
X-Stack 2 PI Meeting @ MIT, Cambridge, MA - May 28, 2014

Martin Schulz (lead PI)

Co-PIs: Peer-Timo Bremer (LLNL), Todd Gamblin (LLNL), Jeff Hollingsworth (UMD), John Mellor-Crummey (Rice), Bart Miller (UW), Valerio Pascucci (Utah), Nathan Tallent (PNNL)

PIPER Vision

- **Performance Analysis for the X-Stack**
  - Enable insights into application performance
  - Allow users and runtimes to act on insights



PIPER: Major Challenges and Focus Points

- **Performance Analysis as a Cross-Cut of the SW stack**
  - Support for legacy applications and environments
  - Support for new X-Stack 2 programming models
- **Stack wide performance measurements and attribution**
  - Interfaces into all layers of the system
  - Understanding of relationships between layers
  - Clearing house for performance interfaces
- **Ability to work on adaptive systems**
  - Applications, e.g., AMR or Multiphysics codes
  - Software stack, e.g., load balancing or dynamic scheduling
  - Hardware architecture, e.g., resilience or power scheduling
- **New techniques to exploit performance data**
  - Analysis and visualization of complex information
  - Automatic feedback and tuning
- **Central: common data model**

The PIPER Team

- **Lawrence Livermore National Laboratory**
  - Martin Schulz (lead PI), Peer-Timo Bremer, Todd Gamblin, Abhinav Bhatele, David Boehme (starting in July)

- **Pacific Northwest Laboratory**
  - Nathan Tallent

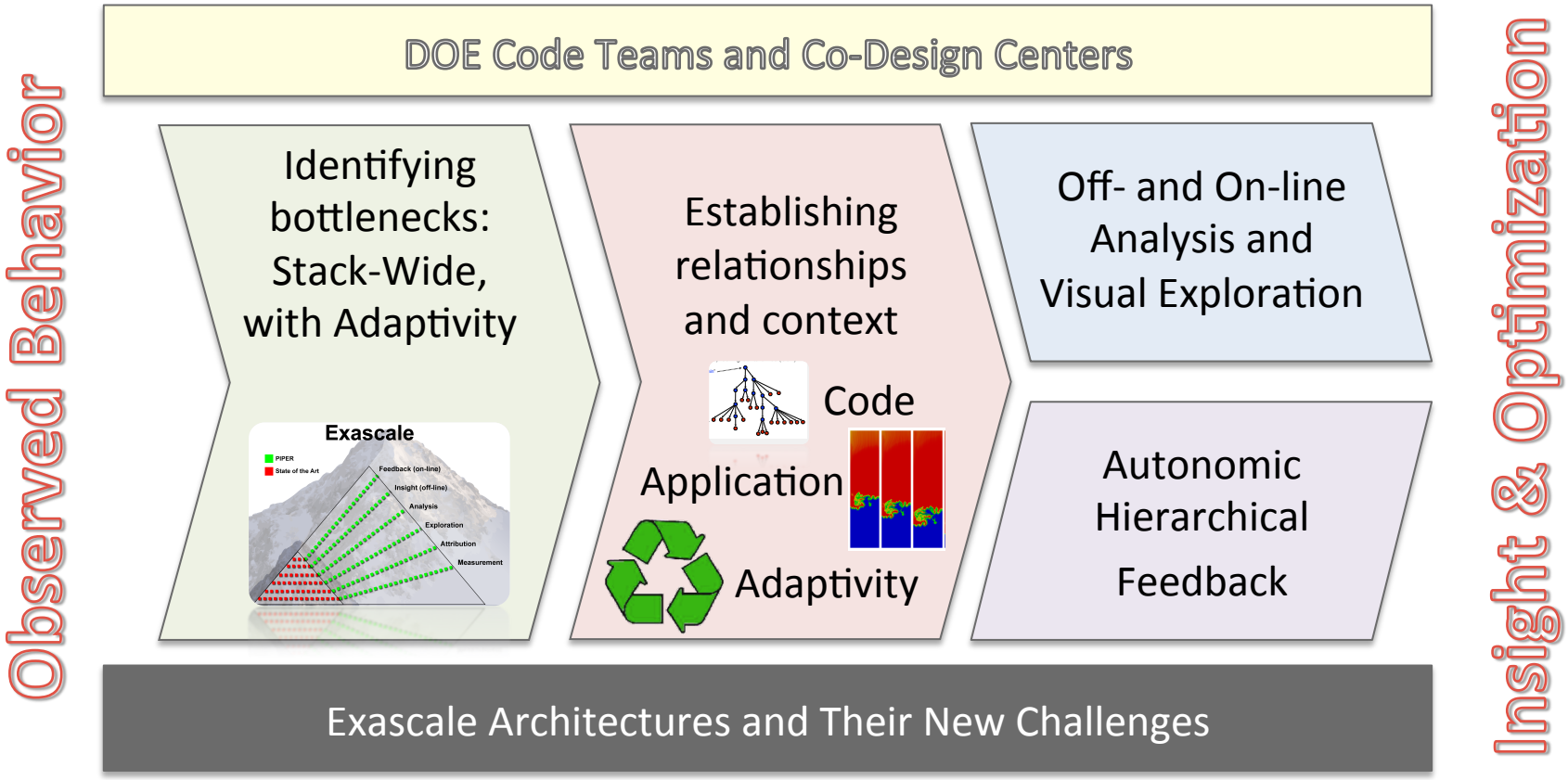
- **Rice University**
  - John Mellor-Crummey, Mark Krentel, Mike Fagan, Laksono Adhianto

- **University of Maryland**
  - Jeff Hollingsworth

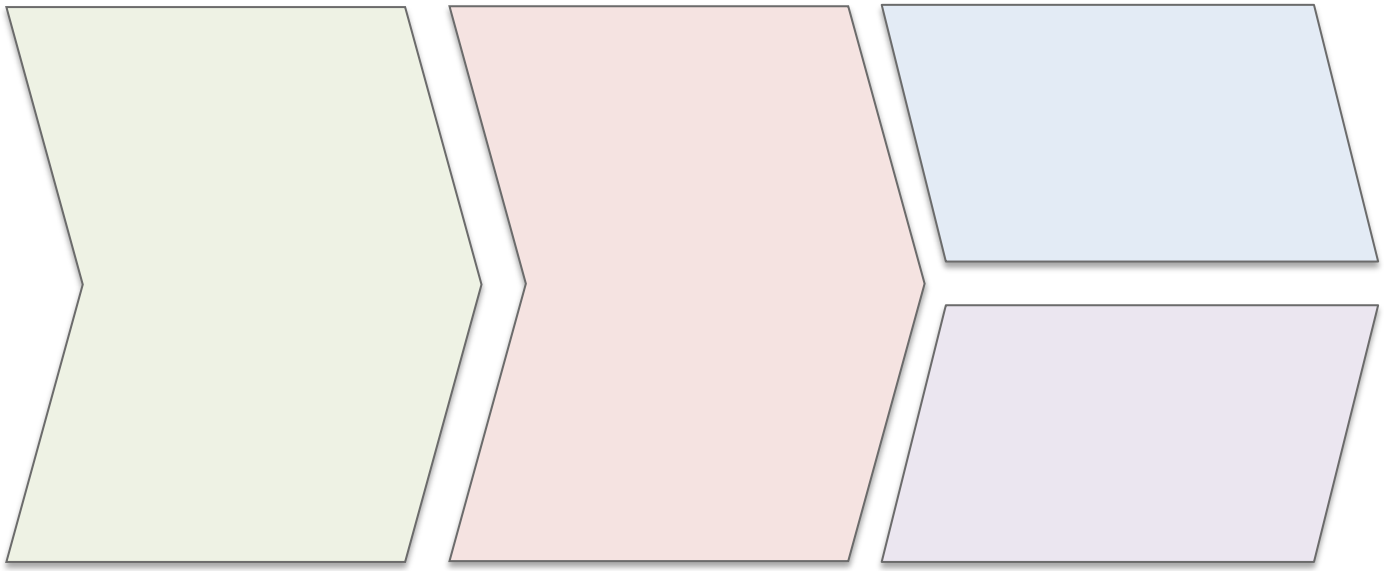
- **University of Utah**
  - Valerio Pascucci, Yarden Livnat

- **University of Wisconsin**
  - Bart Miller, Bill Williams

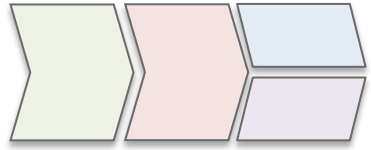
The PIPER Project Vision



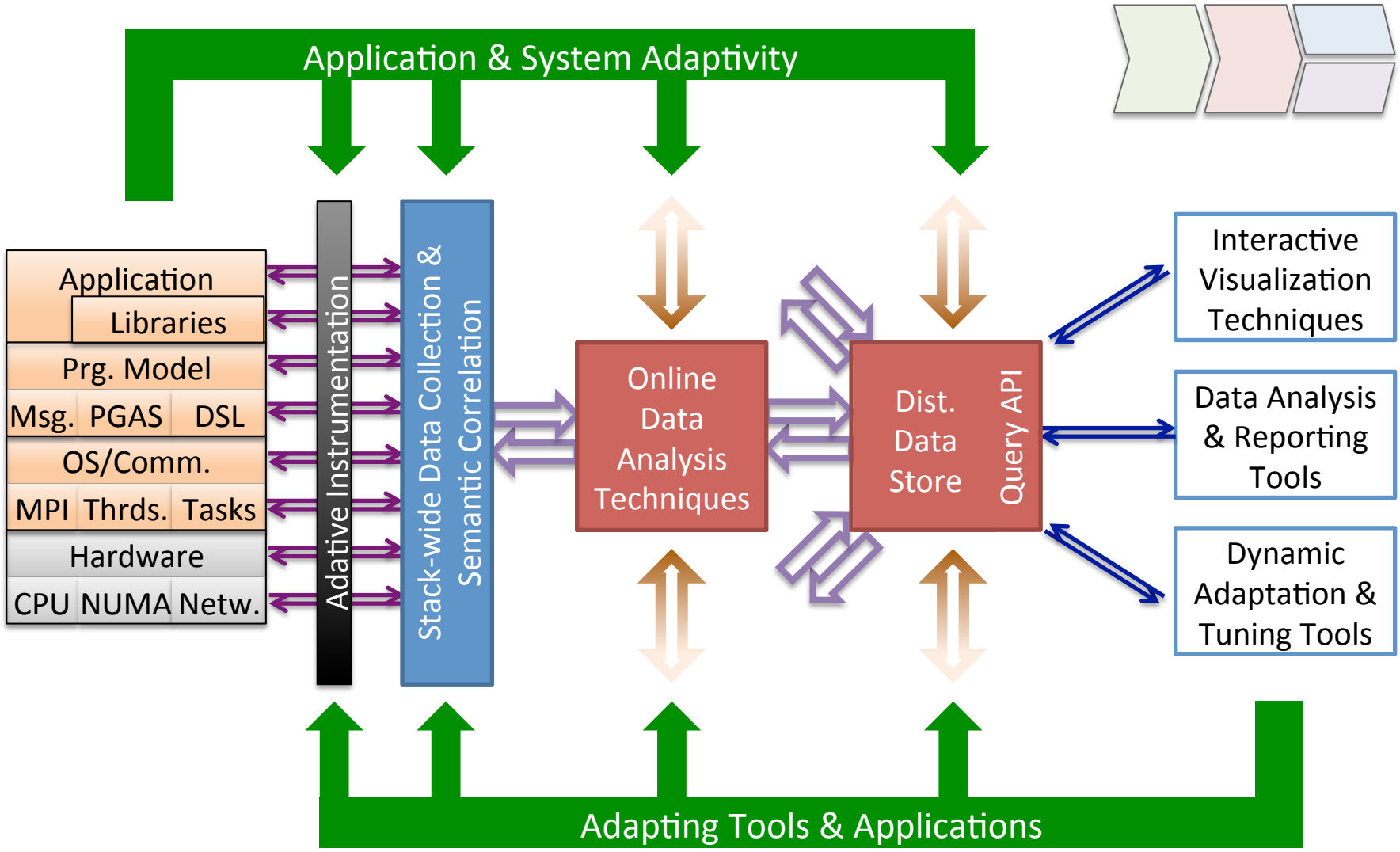
The PIPER Project Vision



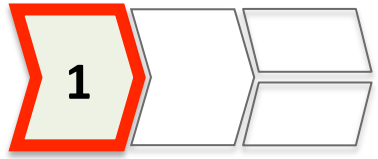
The PIPER Project Vision

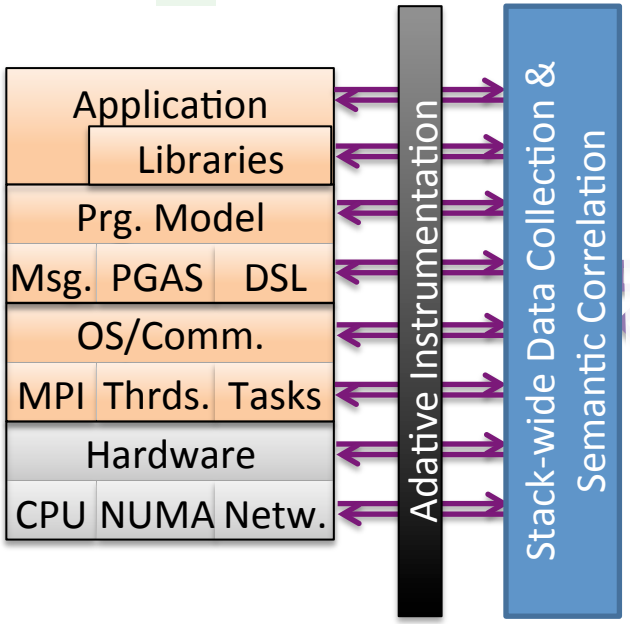


PIPER Architecture



Thrust 1: Measurement / Interfaces

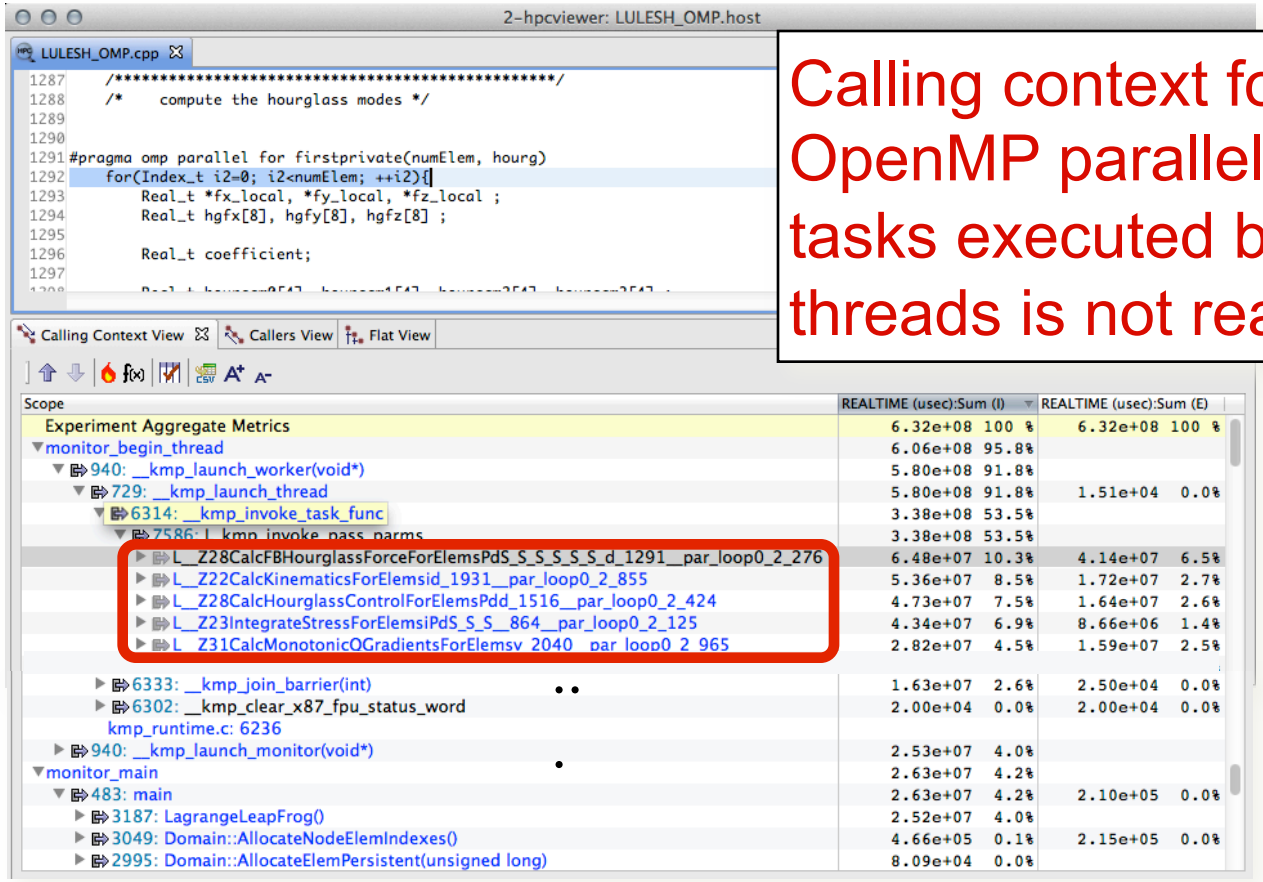




- **Stack-wide instrumentation**
  - Access to performance data from
    - Application
    - Programming Model/Runtime/OS
    - Hardware
  - Metadata to correlate information
- **Goals:**
  - New interfaces into system stack
  - New instrumentation technology
  - Techniques to deal with adaptivity
- **Learn from other interfaces**
  - PMPI function wrapping interface
  - MPI_T publish/subscriber model
  - OMPT callback and sampling model

Use Case: Threaded Runtimes

- **Large gap between between threaded programming models and their implementations**

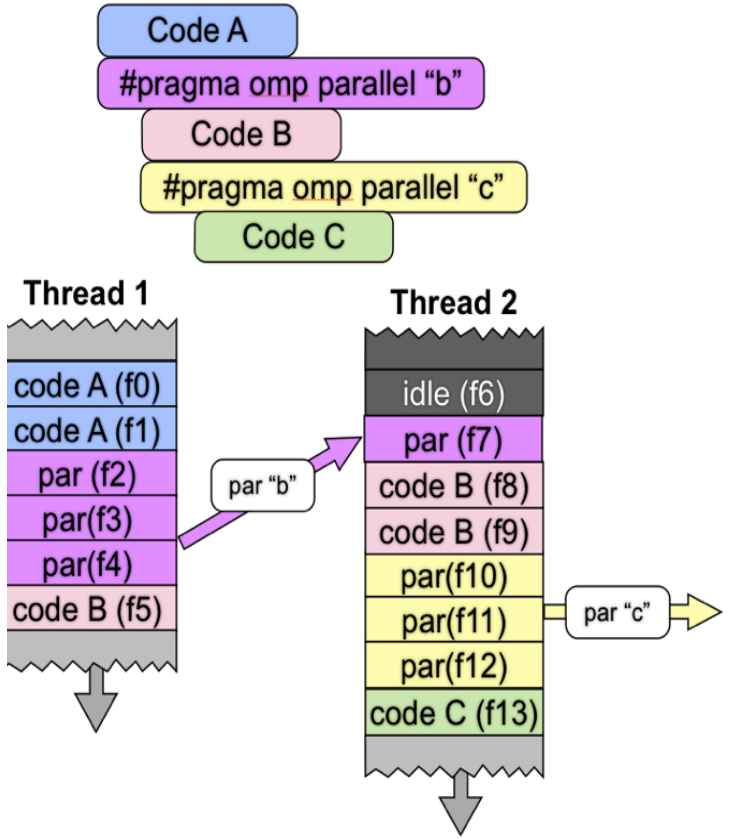


Calling context for code in OpenMP parallel regions and tasks executed by worker threads is not readily available

- **Instrumentation necessary for tools to bridge the gap**

OMPT: Instrumenting OpenMP

- **Goal: enable tools to gather information and associate costs with application source and runtime system**
  - Hooks for tracing and sampling
  - Minimal overhead
  - Low implementation complexity
  - Mandatory vs. optional parts
- **Call-stack stitching**
  - Create user level view
  - Hide runtime system details
- **Status:**
  - Initial API design complete
  - Ratified by ARB as white paper
  - Working on standardization
  - First runtimes and tools



Integrated View of MPI+OpenMP with OMPT

LLNL's luleshMPI_OMP (8 MPI x 3 OMP), 30, REALTIME@1000

hpcviewer: luleshMPI_OMP.host

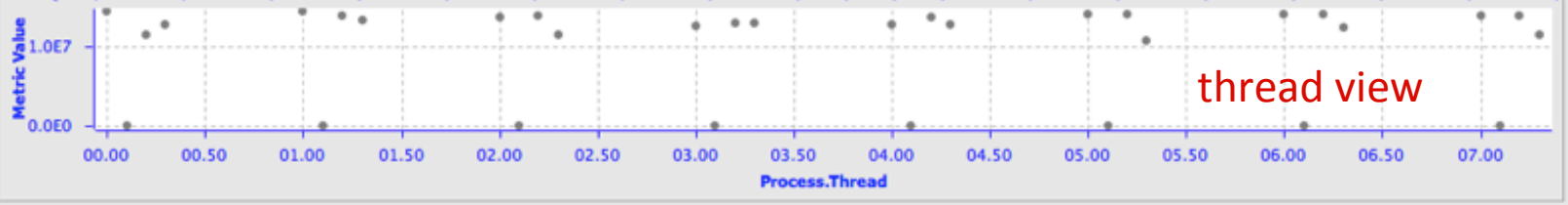
luleshMPI_OMP.cc

```
3214   }
3215
3216   {
3217 #pragma omp parallel for firstprivate(numNode)
3218     for( Index_t gnode=0 ; gnode<numNode ; ++gnode )
3219     {
3220        Index_t count = nodeElemCount[gnode] ;
3221        Index_t start = nodeElemStart[gnode] ;
3222        Real_t fx_tmp = Real_t(0.0) ;
```

source view

[Plot graph] CalcFBHourglassForceForElems(int*, double*, double*, double*, double*, double*, double*, double*, double*, int*, int*, int*, double*, double*, double*, double*, double*, doub...

lems(int*, double*, double*, double*, double*, double*, double*, double*, double*, int*, int*, int*, double*, double*, double*, double*, double*, double*, double*,



thread view

Calling Context View | Callers View | Flat View

Scope	REALTIME (usec):Sum (I)		REALTIME (usec):Sum (E)	
Experiment Aggregate Metrics	3.55e+10	100 %	3.55e+10	100 %
monitor_main	2.58e+10	72.8%		
483: main	2.58e+10	72.8%	7.02e+03	0.0%
loop at luleshMPI_OMP.cc: 5625	2.58e+10	72.8%	4.01e+03	0.0%
5626: LagrangeLeapFrog(Domain*)	2.53e+10	71.2%	1.50e+04	0.0%
4796: LagrangeNodal(Domain*)	1.68e+10	47.5%	5.02e+04	0.0%
s(Domain*)	1.60e+10	45.0%	1.18e+07	0.0%
ceForElems(Domain*)	1.44e+10	40.7%	1.56e+07	0.0%
ssControlForElems(Domain*, double*, double)	1.09e+10	30.7%	1.82e+08	0.5%
rglassForceForElems(int*, double*, double*, double*, double*, double*, d	7.86e+09	22.1%	2.41e+08	0.7%
lourglassForceForElems(int*, double*, double*, double*, double*, double*,	3.66e+09	10.3%	2.55e+09	7.2%
fork_barrier(int, int)	3.08e+09	8.7%	5.01e+03	0.0%
fork_barrier(int, int)	5.44e+08	1.5%	1.00e+04	0.0%
lourglassForceForElems(int*, double*, double*, double*, double*, double*,	3.16e+08	0.9%	3.15e+08	0.9%

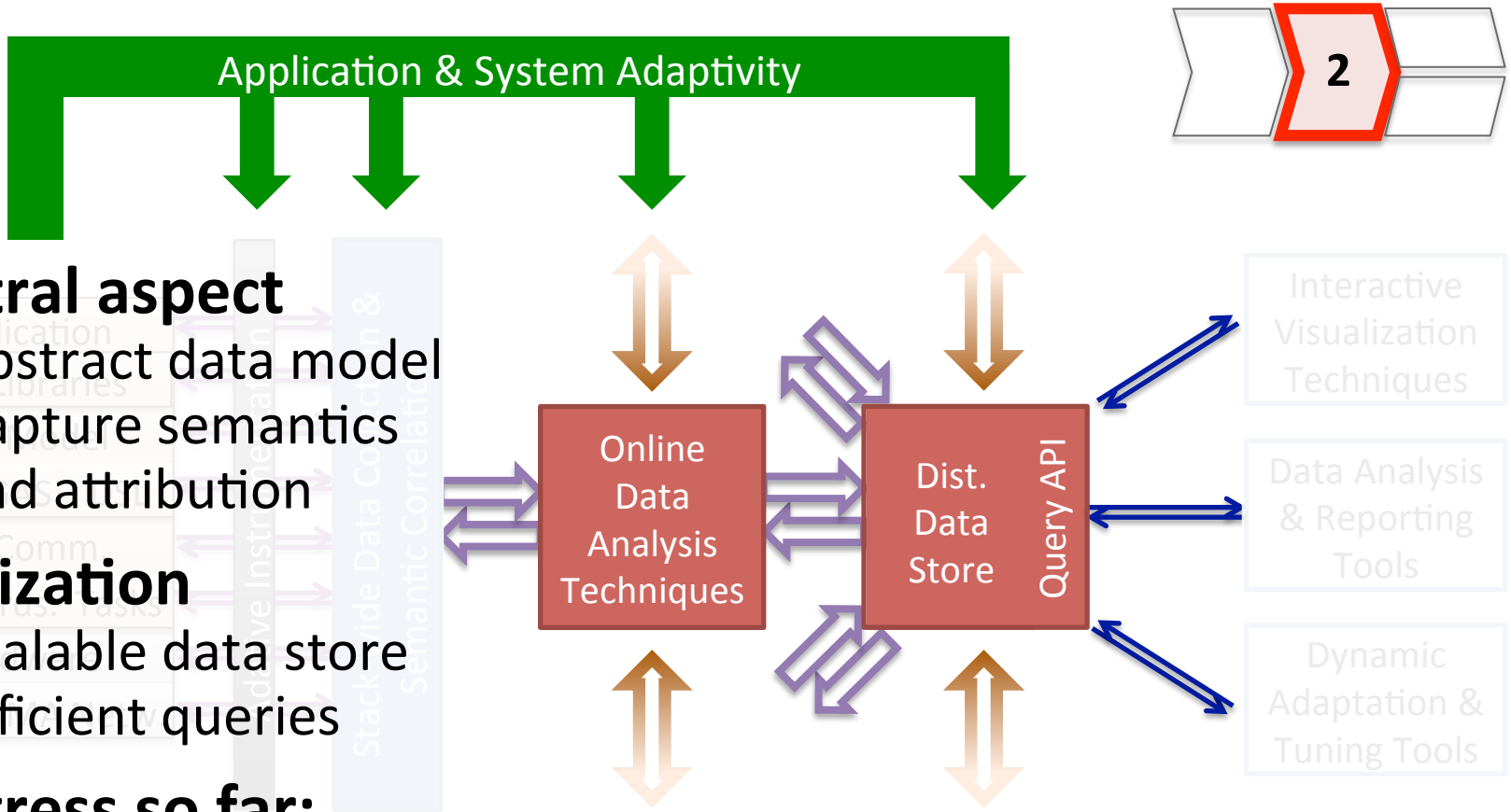
metric view

Look for HPCToolkit at tomorrow's Technology Marketplace

Next Steps and Other Efforts

- **Expanding to new programming/execution models**
  - OpenMP as initial case study
  - Principle applicable to any thread/task system
  - Discussions on integration and generalization necessary
  - Initial discussions: HPX and OmpSs
- **Network performance analysis**
  - Pinpoint and quantify network contention
  - Attribute contention to user communication
- **Infrastructure**
  - Dynist 8.2 release (dynamic instrumentation toolkit)
    - Functionality, speed, and stability have all noticeably improved
    - Included new in the Redhat distribution
  - MRNet 4.1 release (scalable tool infrastructure)
    - Reduced latency, improved startup, health monitoring

Thrust 2: Data Model and Attribution

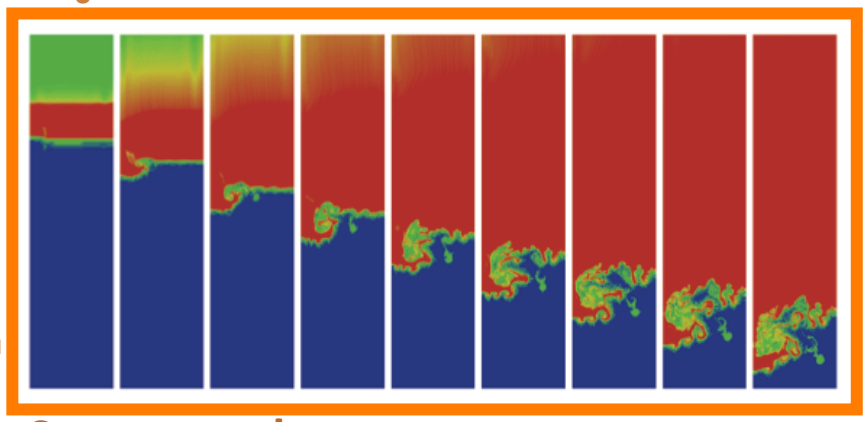


- **Central aspect**
  - Abstract data model
  - Capture semantics and attribution
- **Realization**
  - Scalable data store
  - Efficient queries
- **Progress so far:**
  - Discussion with visualization and performance community
  - Community agreement of a model strawman
  - Formal write-up in progress

A Central Data Model to Enable Mappings

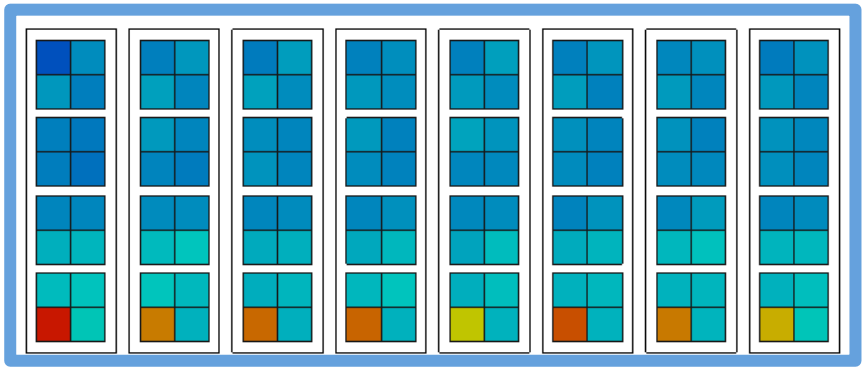
M_0 = Density

S_2 = y-coord



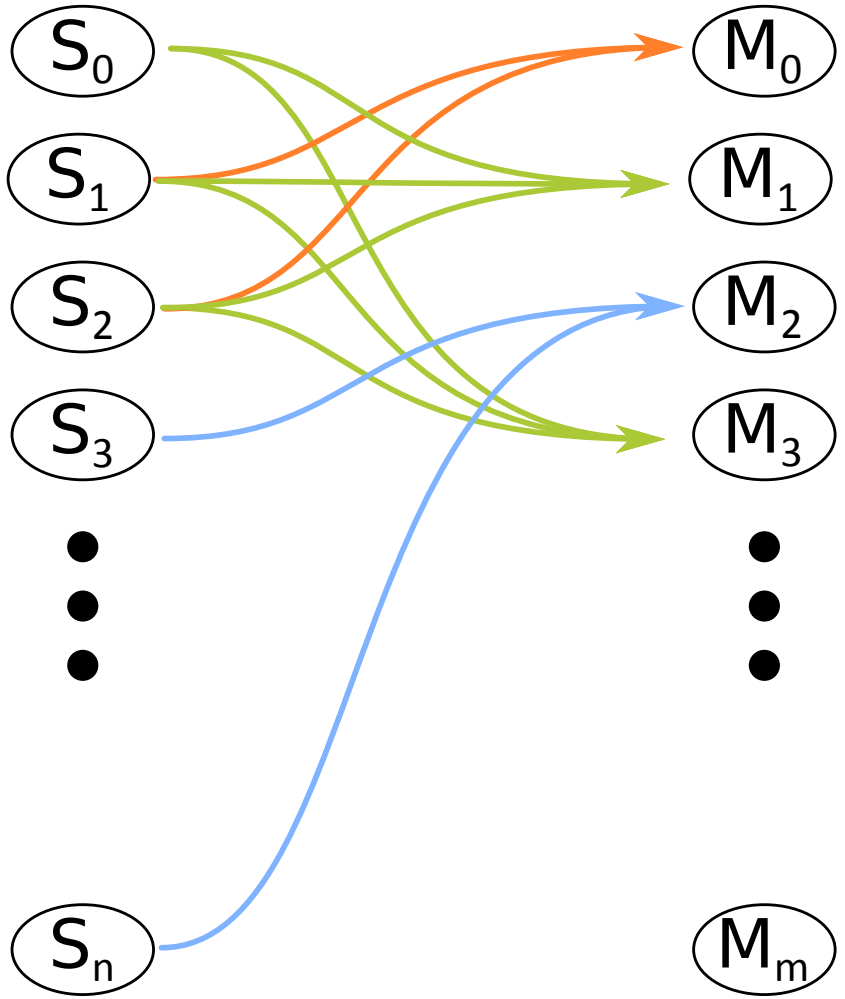
S_1 = x-coord

M_2 = Branch misses



S_3 = core-id, S_n = node-id

Spaces **Metrics**

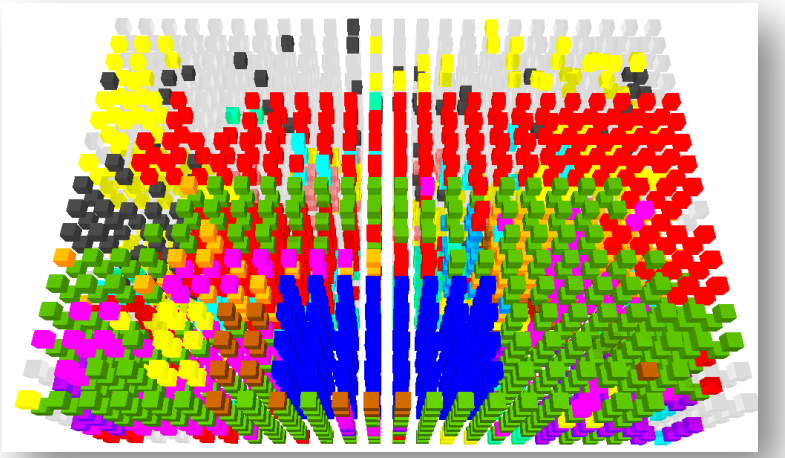


Thrust 3: Visual Performance Analysis

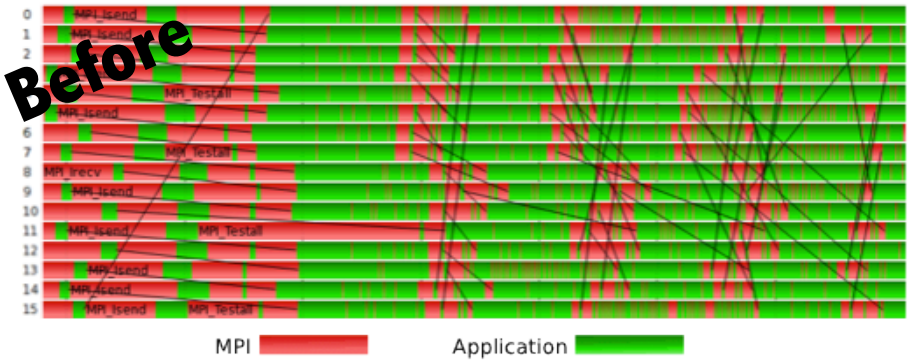
- **Holistic analysis**
  - Capture relationship between measurements
  - Adaptive behavior
  - Mappings and correlations through data model
  - Integrate meta data to capture adaptivity
- **Goal: understand performance behavior**
  - Enable users to understand their performance
  - Enable users to act on newly gained insight
- **Main challenge**
  - Map information to domains that are intuitive for users
    - Physical simulation domain, data structures, high-level constructs
    - Mask runtime constructs, resilience features, hidden adaptivity
  - Provide multiple perspectives

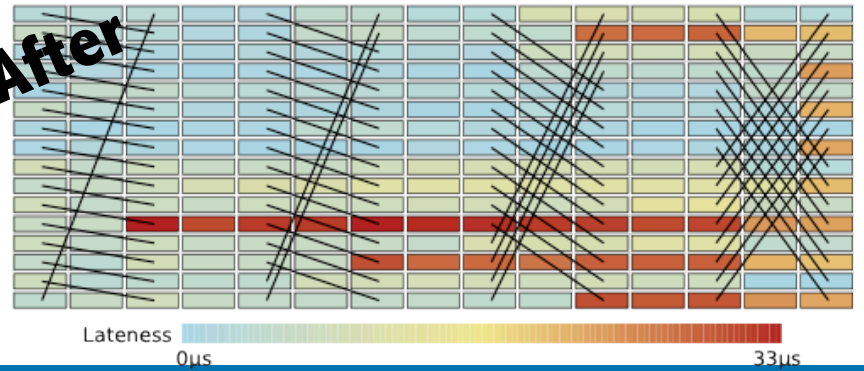
Visualization Help Create Insight

- **A picture can say more than 1000 words**
  - But it needs to be the right picture
  - New InfoVis techniques needed
    - Unstructured and discrete data
    - High dimensionality
    - Complex relationships

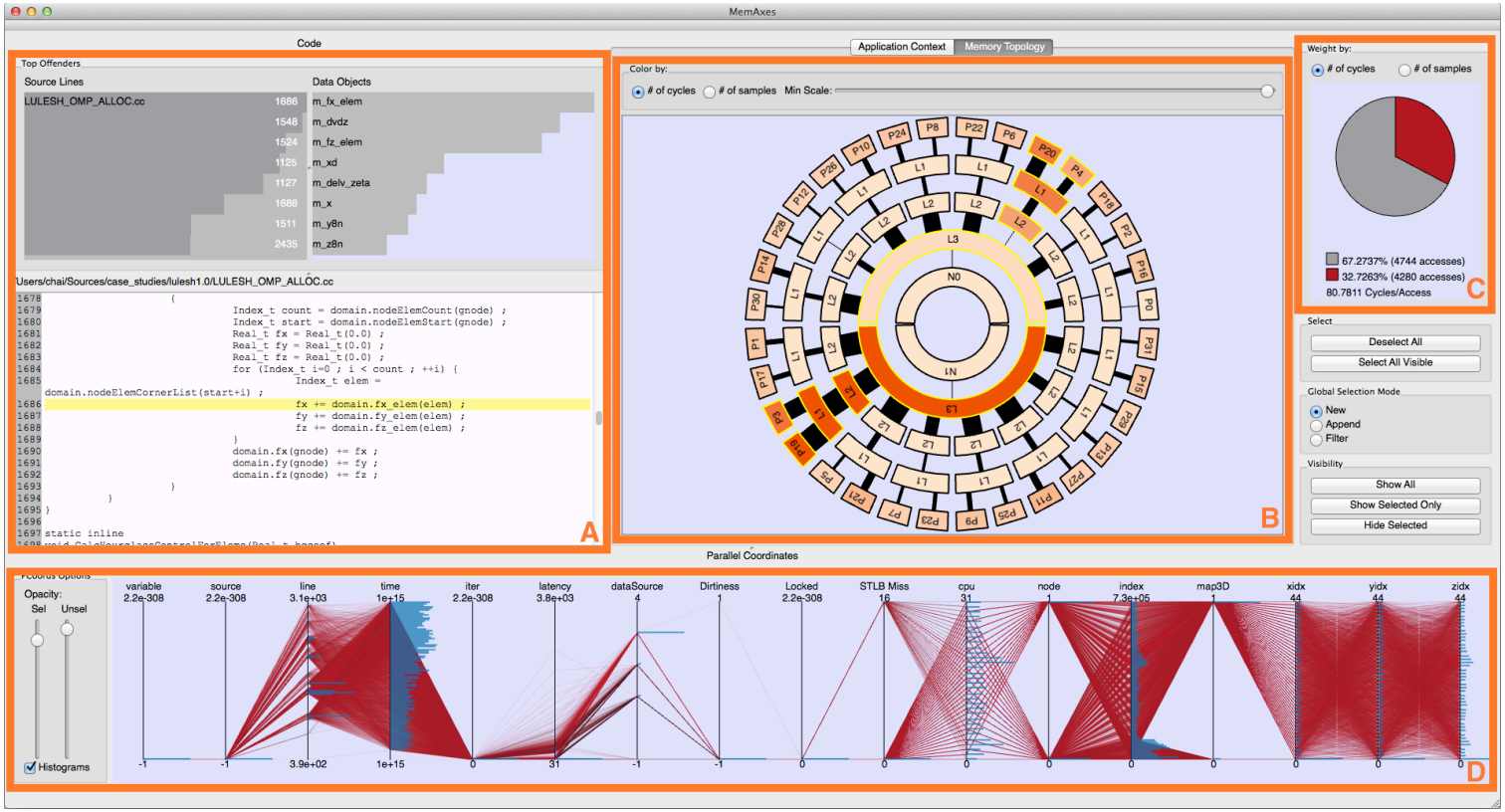


- **Example**
  - Job mappings on Cray systems
- **Example**
  - Messages in logical time with delay information



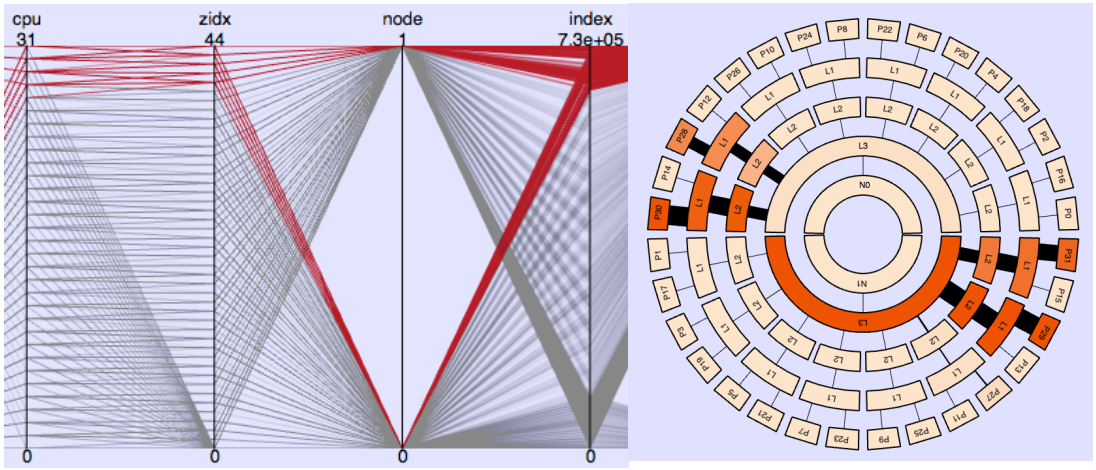


Use Case MemAxes: On-node Data Motion

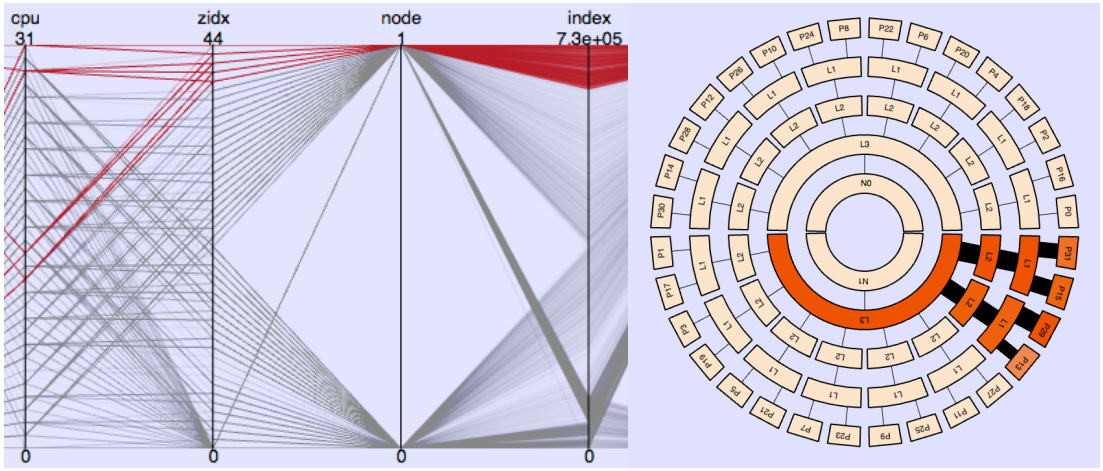


- **A measurement component samples memory instructions**
- **We map latency information onto source code and node topology**
- **Parallel coordinates view allows exploration of correlations**

Linked Views Show On-node Data Motion Problems



Default thread affinity with poor locality



Optimized thread affinity with good locality

- Parallel coordinates view shows correlation between array index and core id in LULESH
- Linked node topology view shows data motion for highlighted memory operations
- A contiguous chunk of an array is initially split between threads on four cores
- Using an optimized affinity scheme, we improve locality
- Performance improved by 10%
- Look for MemAxes at tomorrow's Technology Marketplace

Thrust 4: Feedback & Autotuning

- **Performance information as feedback**
  - Challenge 1: large search space
  - Challenge 2: need for distributed decisions
- **Goal: hierarchical feedback loop**
  - Local decisions where possible
  - Global decisions where necessary
- **Integration into all levels of software stack**
  - Inclusion of stack-wide information
  - Actors in all levels of the X-Stack

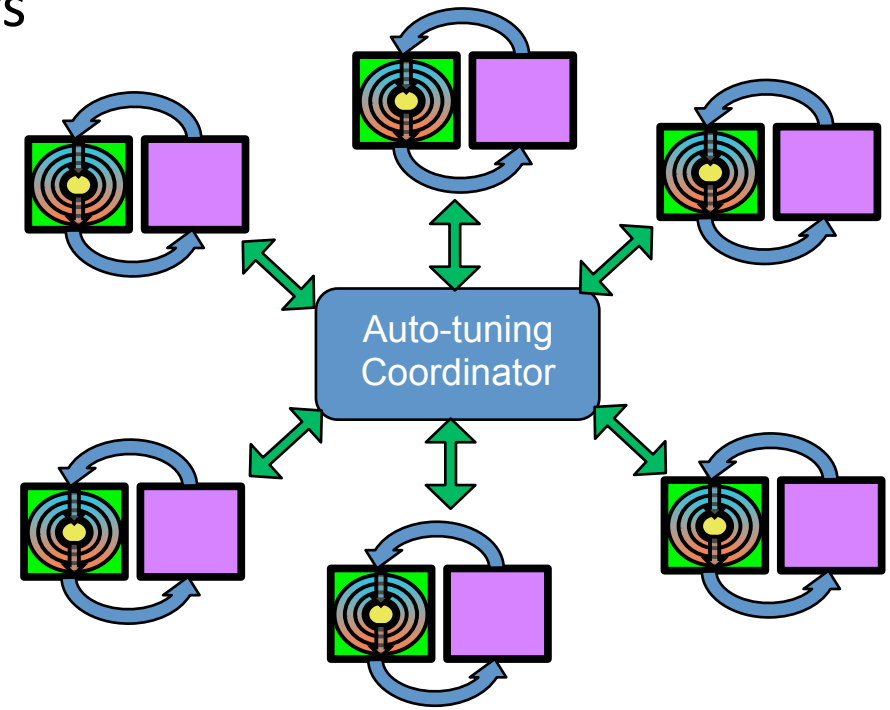
Interactive Visualization Techniques

Data Analysis & Reporting Tools

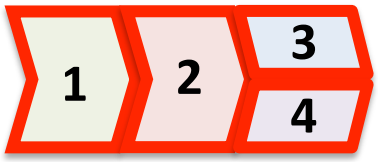
Dynamic Adaptation & Tuning Tools

Hierarchical Auto-Tuning

- **Challenge: high dimensional search spaces**
  - Separate into individual loops where possible
  - Initial results show significant search speed improvements
- **Challenge: high node counts**
  - Central auto-tuner no longer feasible
  - Multiple independent tuners often interact poorly
- **Goal: distributed, coordinated search**
  - Coordination service as mediator
  - Informs workers
  - Aggregates data



Status Summary



- **Thrust 1: Measurement / Interface**
  - Proof of concept for threaded RT interface
  - Prototype for OMPT use in HPCToolkit
  - Network contention analysis
  - Infrastructure improvements and releases (Dyninst & MRNet)
- **Thrust 2: Data Model and Attribution**
  - Initial data model complete and being written up
- **Thrust 3: Visual Performance Analysis**
  - New visualization techniques for message displays
  - MemAxes: on-node data motion visualization
- **Thrust 4: Feedback and Auto-Tuning**
  - Improved search for high dimensional spaces
  - First design of a distributed, coordinated tuning approach

Comparison to "Conventional" Techniques

- **Learning from conventional programming models**
  - Initial starting point: performance analysis for MPI+X
    - Existing infrastructure, experience, and interfaces
    - Expand to cross-stack correlation
  - Identify types of information and meta-data needed
- **Supporting emerging models**
  - Support emerging low-level OS and RT system (new interfaces)
  - Support emerging programming abstractions (new mappings)
  - Goal: compatible interfaces to avoid MxN scenario
  - Capture and properly represent system adaptivity
- **Final goal: new capabilities for the X-Stack**
  - New types of data and new presentation
  - Correlated and intuitive representation of information
  - Support users using the new X-Stack models
- **Will have to grow with the X-Stack projects**

Questions to Other Teams

- **Needs for new programming approaches**
  - What types of performance problems do you expect and/or want to measure and analyze?
  - What abstractions should a performance tool use to display?
    - Which source constructs and data structure information is useful?
    - How is this exposed to the tool?
  - What parts can benefit from auto-tuning?
    - Which knobs exist and how are they exposed?
- **Interfaces to runtimes, OS, and hardware**
  - What are the units managed by the respective layers?
  - How is information exposed?
  - What interfaces allow tools / tuners to set knobs
- **Interaction with the external environment / scheduler**
  - Request, initialize and control tool resources
- **Panel tomorrow:**
  Performance Tools and Their Interfaces