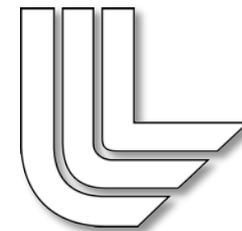


ASC's Tool Needs

Todd Gamblin

Lawrence Livermore National Laboratory



ECI Runtime Systems Workshop ♦ March 12 2015

LLNL-PRES-668497

<http://scalability.llnl.gov/>

Contributors:

David Beckingsale
Abhinav Bhatele
Peer-Timo Bremer
Alfredo Gimenez
Katherine Isaacs
Adam Kunen
Martin Schulz

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



ASC codes face two unprecedented performance challenges that tools can help to address.

■ Performance portability

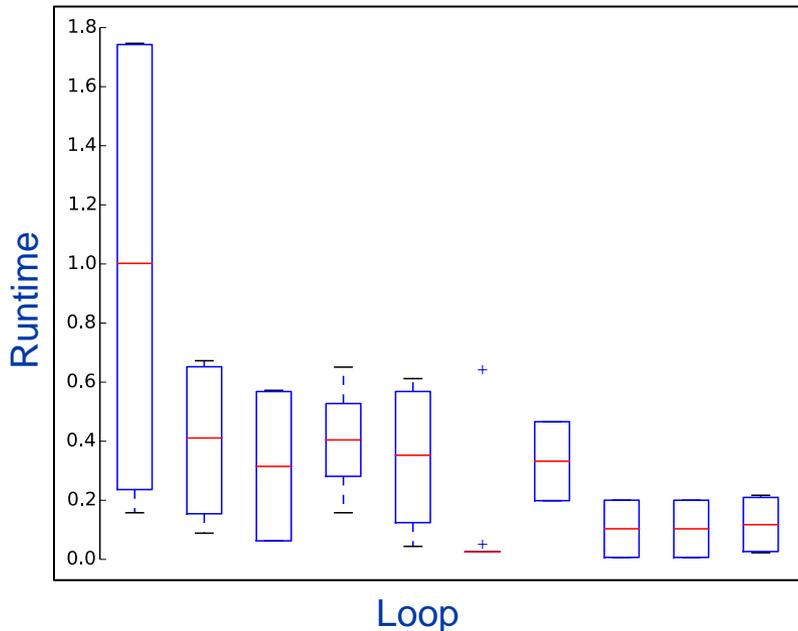
- What abstractions will allow code to run well across architectures?
- How do we choose the right:
 - Scheduling policy?
 - Granularity of parallelism?
 - Data access order?

■ Performance variability

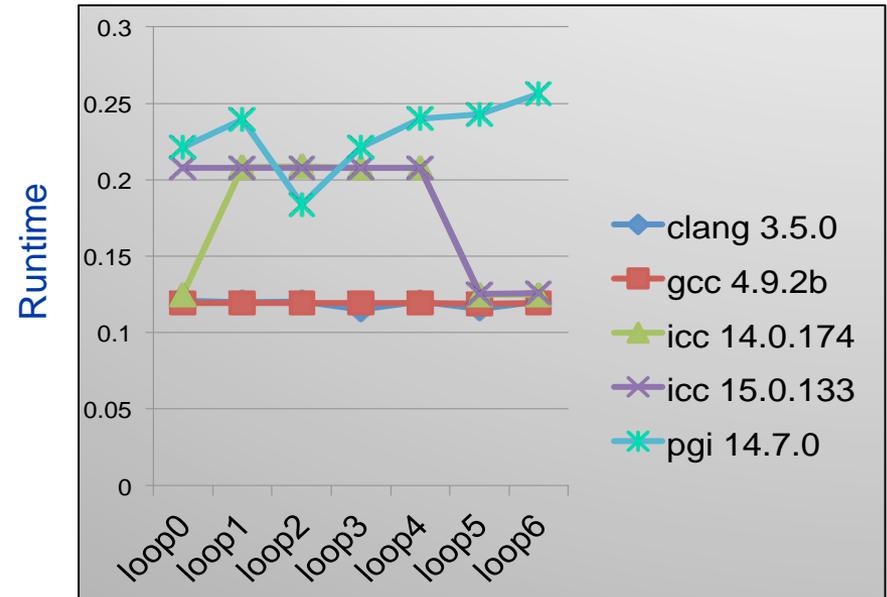
- Code performance can vary wildly between runs.
- Can be attributed to:
 - Resource contention (memory, network congestion)
 - Dynamic, nondeterministic scheduling
 - Data-dependent/adaptive applications (depends on input deck)

Current abstractions provide tunability, not portability.

Runtimes for loops in ARES, varying RAJA execution policy



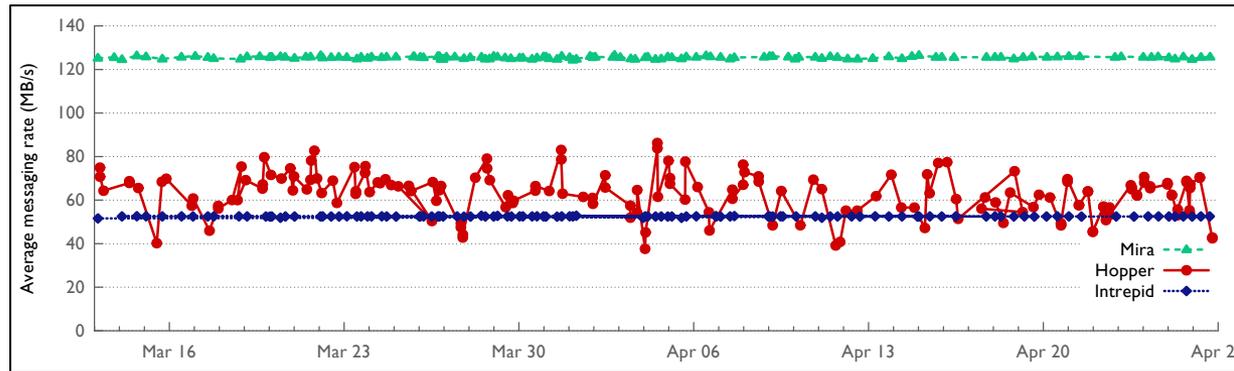
Runtime for simple $X=cY$ loop in Kripke, varying level of abstraction



Increasing abstraction (labmdas, templates)

- User can easily tune runtime parameters in RAJA and Kokkos, but doesn't know what execution model is best for what hardware
- Abstraction has a cost – compilers don't always optimize.
- Finding causes of problems like this can be very difficult.

Many factors contribute to performance variability



Communication in pF3D: Cray (red) vs. BG/P, BG/Q

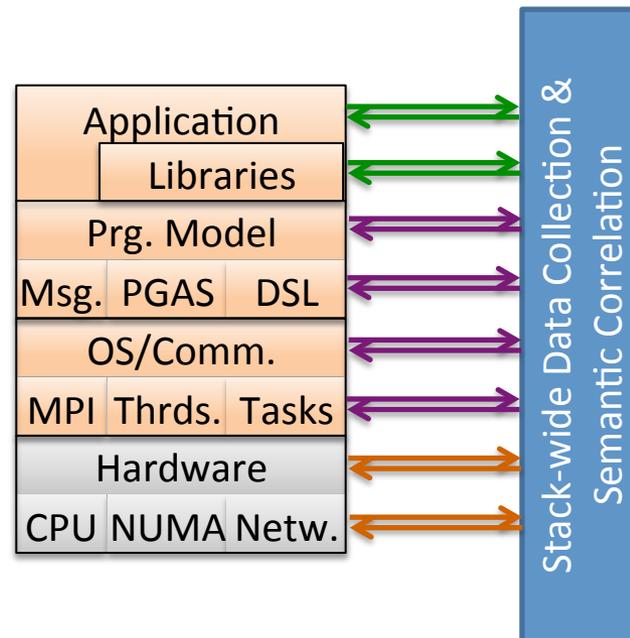
- Network congestion, topology
- OS noise
- Dynamic task scheduling, migration
- Memory topology
- Load imbalance
- Data dependent performance (varies by input deck)
 - Modeling many codes' performance requires modeling data, as well.
- And, moreso at exascale:
 - Power bounds?
 - Resilience issues?

Measurement tools must collect sufficient data to characterize performance variability.

- Single-trial characterizations don't reflect real-world performance.

- Need to collect measurements with *semantic context* at many levels:

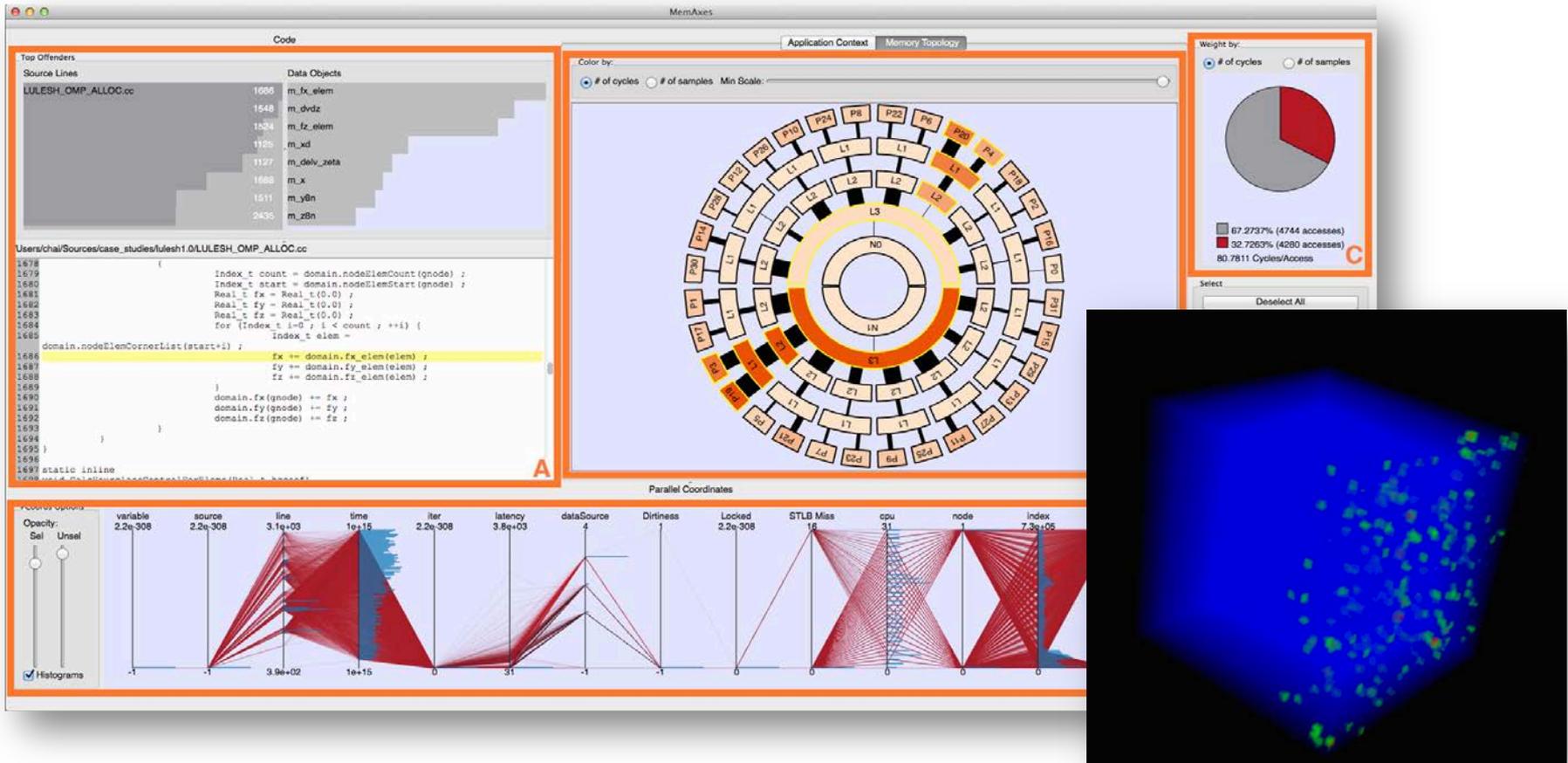
- Application
- Runtime
- Node
- Hardware
- Network
- OS
- Scheduler
- **Entire Facility?**



- This requires support from the OS/Runtime

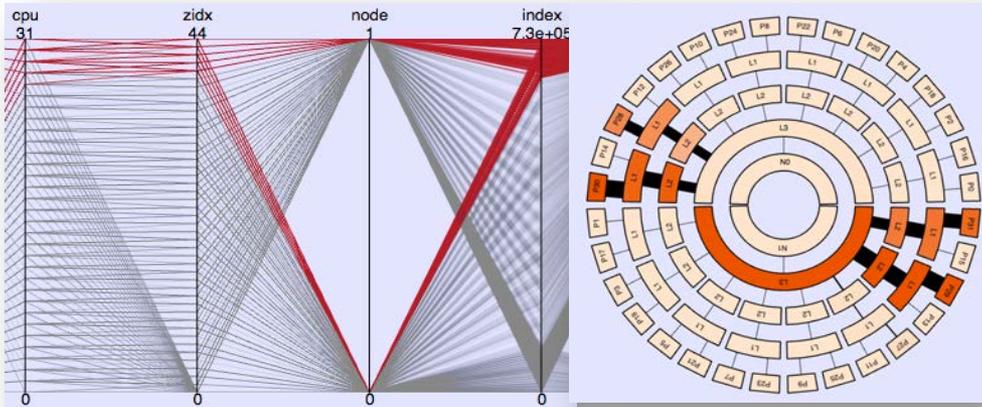
- We need to be able to add thread, task, event, scheduler context
- Interfaces like OMPT, HPX's performance measurement infrastructure are good examples of detailed measurements.
- OCR task view looks like it provides useful data (task graph position)

MemAxes: Visualizing High Dimensional Memory Data

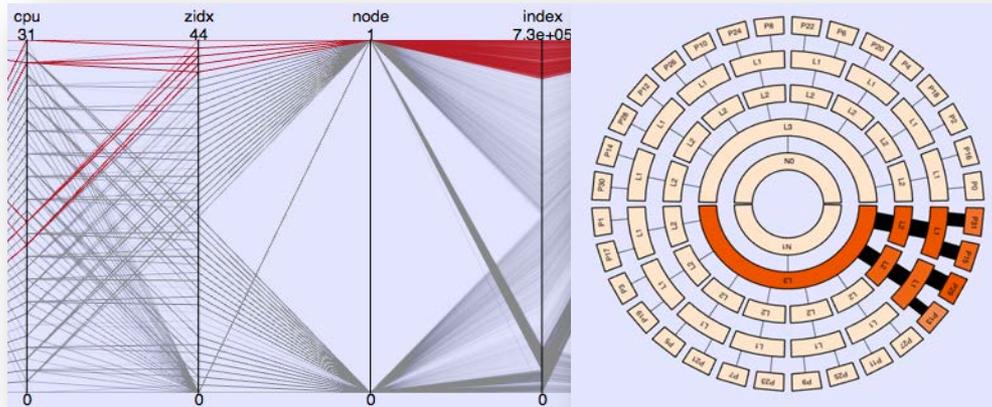


- Shows memory data mapped to code, machine, and application
- Parallel coordinates view (bottom) shows relationships between metrics for which there is no view.

Case Study: Optimization of On-node Locality Problems



Default thread affinity with poor locality

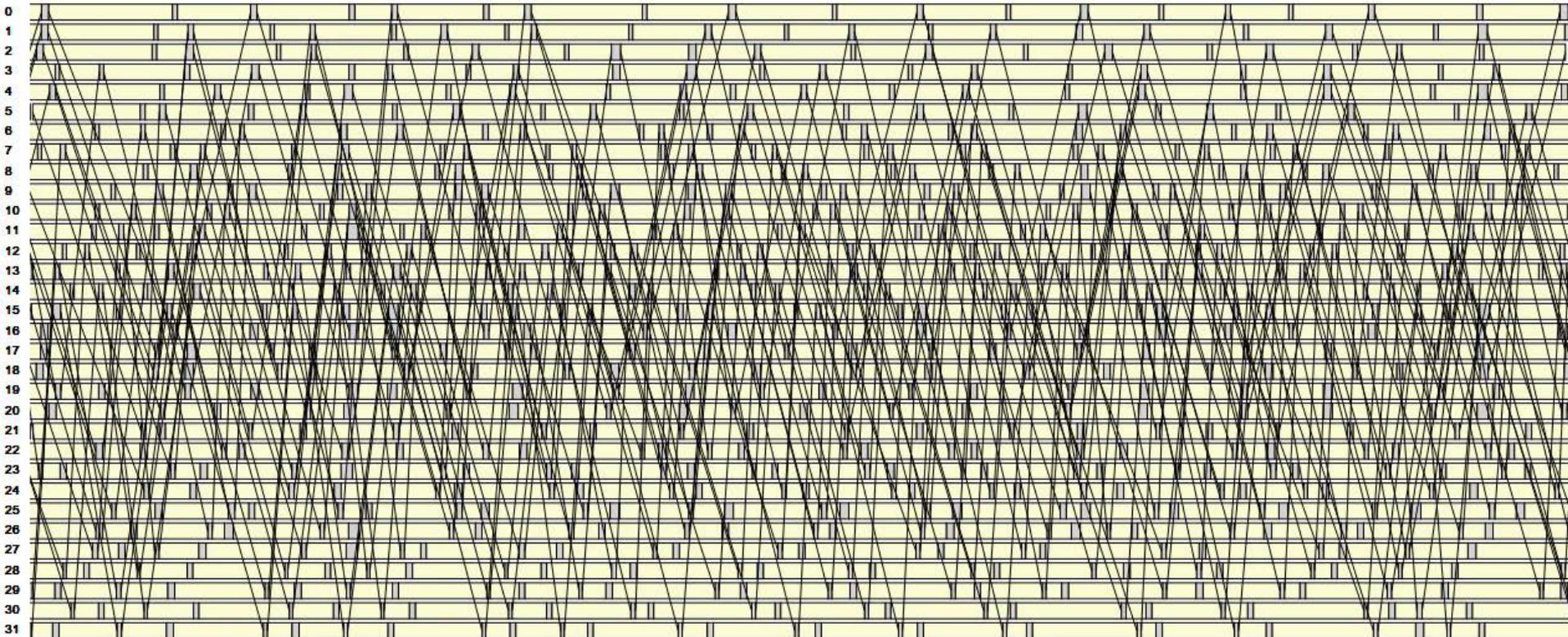


Optimized thread affinity with good locality

- 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%

Ravel: Making Message Traces Readable

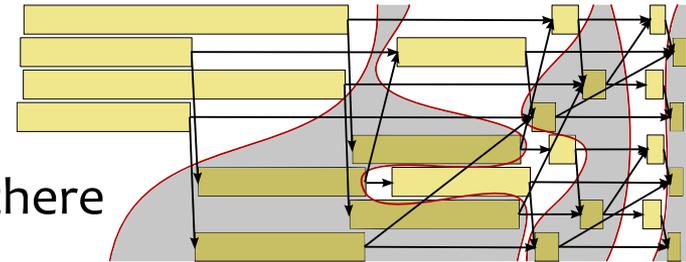
- **Noise and delays make traces unreadable**
- **Hard to understand logical structure of a message or task schedule.**
 - We need new techniques to unravel this hairball -> logical time



Extracting Virtual Time from MPI Traces

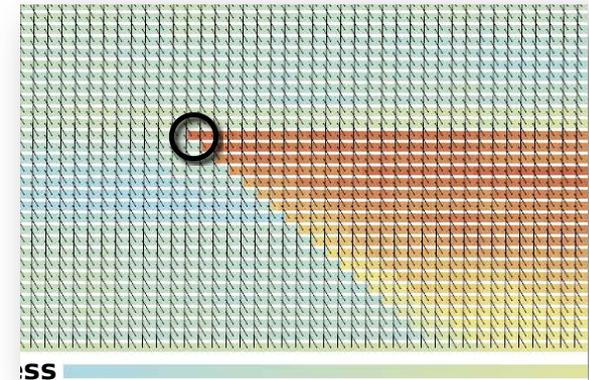
■ Step 1: Identifying time slices

- Start with Lamport *happened-before*
- Concept of connected components
- Start with send/recv pairs and grow from there
- Heuristics on when to stop growing



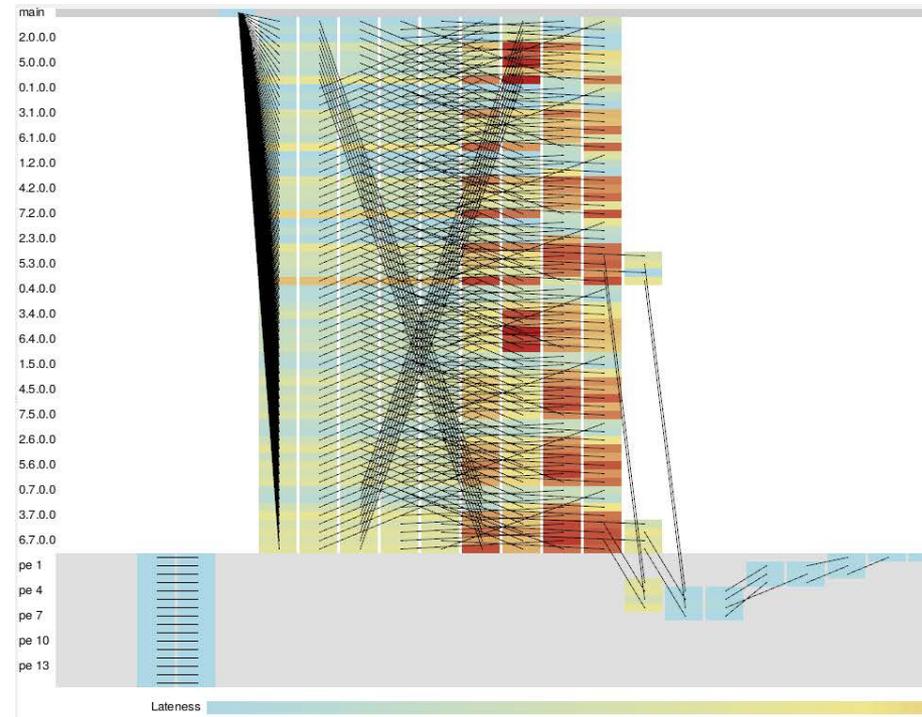
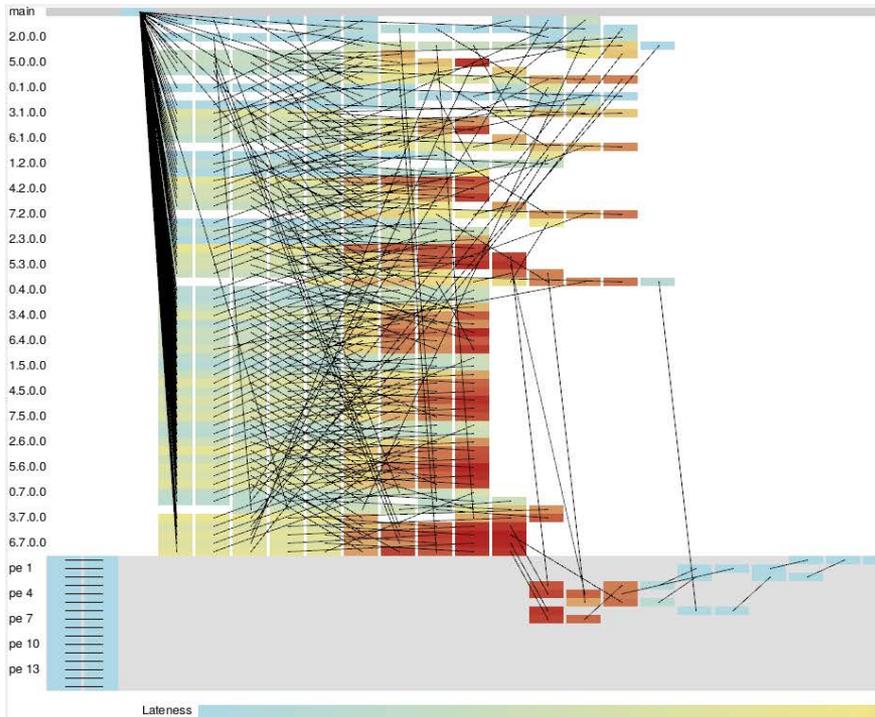
■ Step 2: Mapping timing metrics

- Mapping to virtual time loses physical time
- Reintroduction of time using lateness metric
 - Time difference to end of aligned phase
 - Shows propagations of delays



Making sense of nondeterministic task schedules

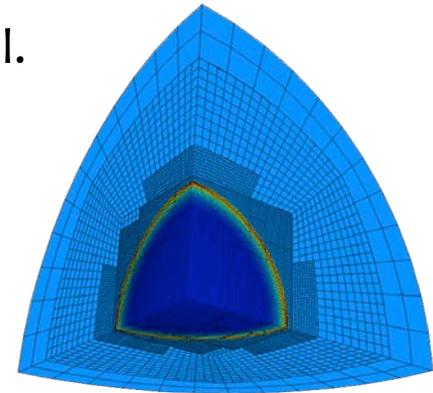
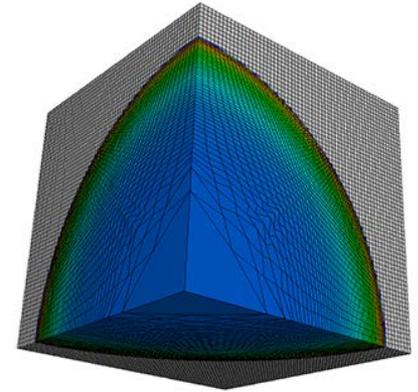
Jacobi2D (Charm++) with and without reordering for nondeterminism



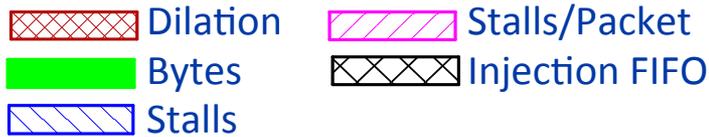
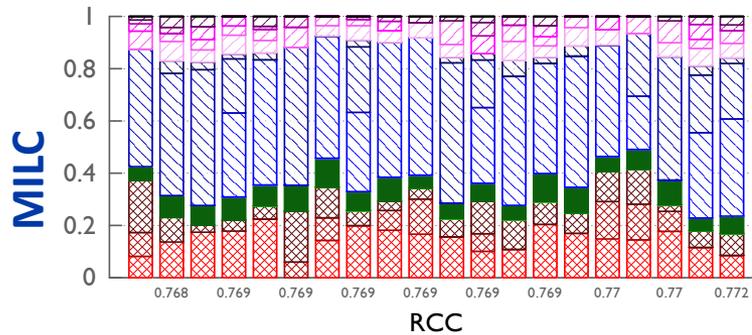
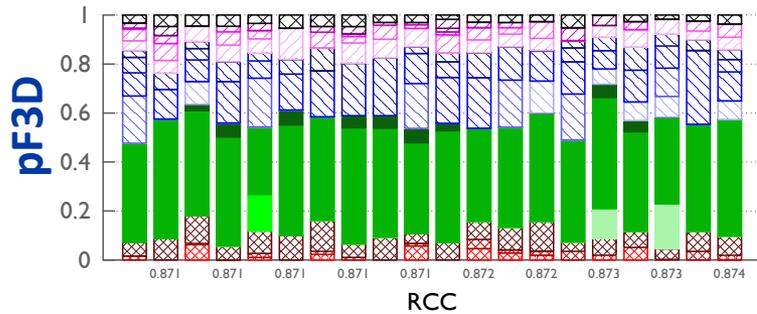
- Need to “unravel” nondeterministic behavior
- Left shows mess of tasks *considering* message receive order
- Right shows messages reordered to ignore nondeterminism, colored by lateness.

With enough data from tools, we can build models for policy decisions

- Using random forests, other regression models, can predict fastest RAJA policy for LULESH with:
 - **75% accuracy** given data size and type of loop.
 - **95% accuracy** if we include code-specific data (specific loop callsite)
- **ARES is more complicated (real physics)**
 - Very low accuracy with just loop type, data size
 - **95% accuracy** when instruction mix is included in the model.
 - More complex measurements are required for real codes vs. proxy applications.
- Performance analysis is big data analysis, but with limited measurements and huge feature space.



Predictive models can give insight into *causes* of performance problems.



- pF3D and MILC performance varies dramatically w/node mapping on BG/Q.
 - Ran 100s of trials w/many mappings.
 - Built models with different feature combinations
- Very different predictive features for the two apps:
 - pF3D: bytes (B/W)
 - MILC: network stalls (latency)
- Need system-wide measurement to understand performance on shared networks, other shared resources.
 - LC starting to use Sandia LDMS for ths.

So, what are ASC's tool needs?

0. Enough data for real-world empirical performance modeling.

- Needed for performance portability and for understanding performance variability.

How to do this:

1. Runtime systems must provide measurement mechanisms.

- Context for threads, tasks, data flow, and synchronization.

2. Continuous, system-wide monitoring.

- Cannot measure any code in isolation: cause may be external.

3. Tools that can correlate and associate *semantics* and other metadata with each measurement.

- Can be through analysis or direct measurement; often need HW support.