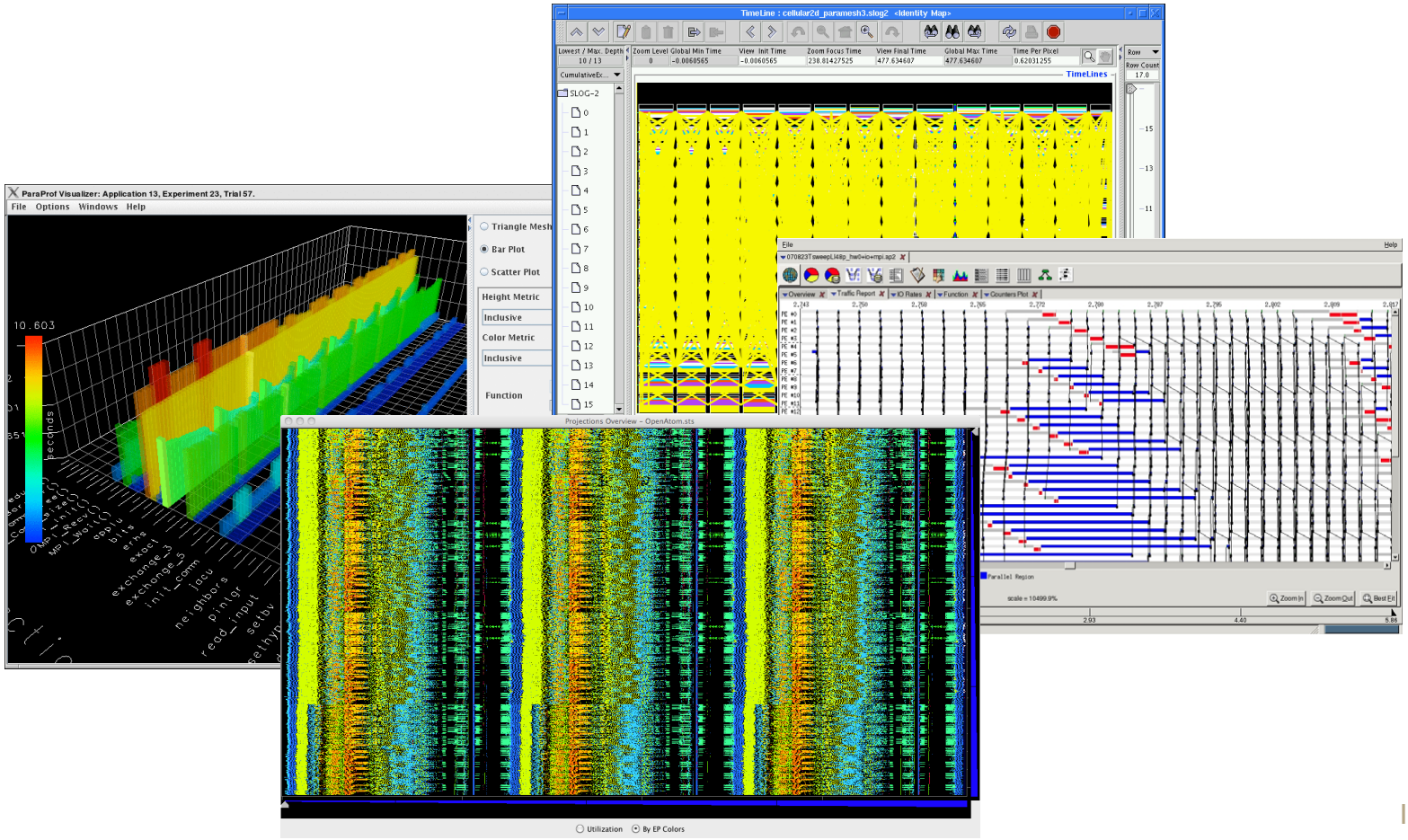
Scalable performance analysis with *Projections*

Sanjay Kale, http://charm.cs.illinois.edu

Based on Thesis defense slides By Chee Wai Lee



Effects of Application Scaling

- Enlarged performance-space.
- Increased performance data volume.
- Reduces accessibility to machines and increases resource costs
  - Time to queue.
  - CPU resource consumption.

Overview

- Introduction.
- Scalable Techniques:
  - Support for Analysis Idioms
  - Data Reduction
  - Live Streaming
  - Hypothesis Testing

Scalable Tool Features: Motivations

- Performance analysis idioms need to be effectively supported by tool features.
- Idioms must avoid using tool features that become ineffectual at large processor counts.
- We want to catalog common idioms and match these with scalable features.

Scalable Tool Feature Support (1/2)

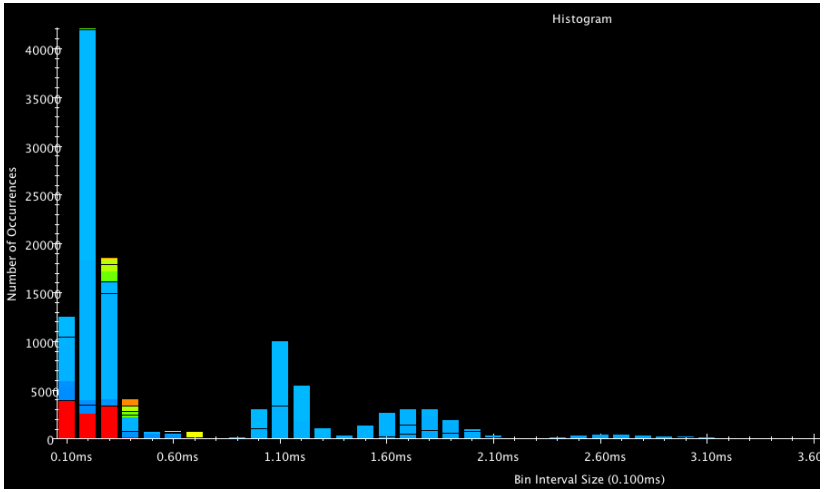
- Non-scalable tool features require analysts to **scan** for visual cues over the processor domain.
- How do we avoid this requirement on analysts?

Scalable Tool Feature Support (2/2)

- Aggregation across processor domain:
  - Histograms.
  - High resolution Time Profiles.
- Processor selection:
  - Extrema Tool.

Histogram as a Scalable Tool Feature

- Bins represent time spent by activities.
- Counts of activities across all processors are added to appropriate bins.
- Total counts for each activity are displayed as different colored bars.

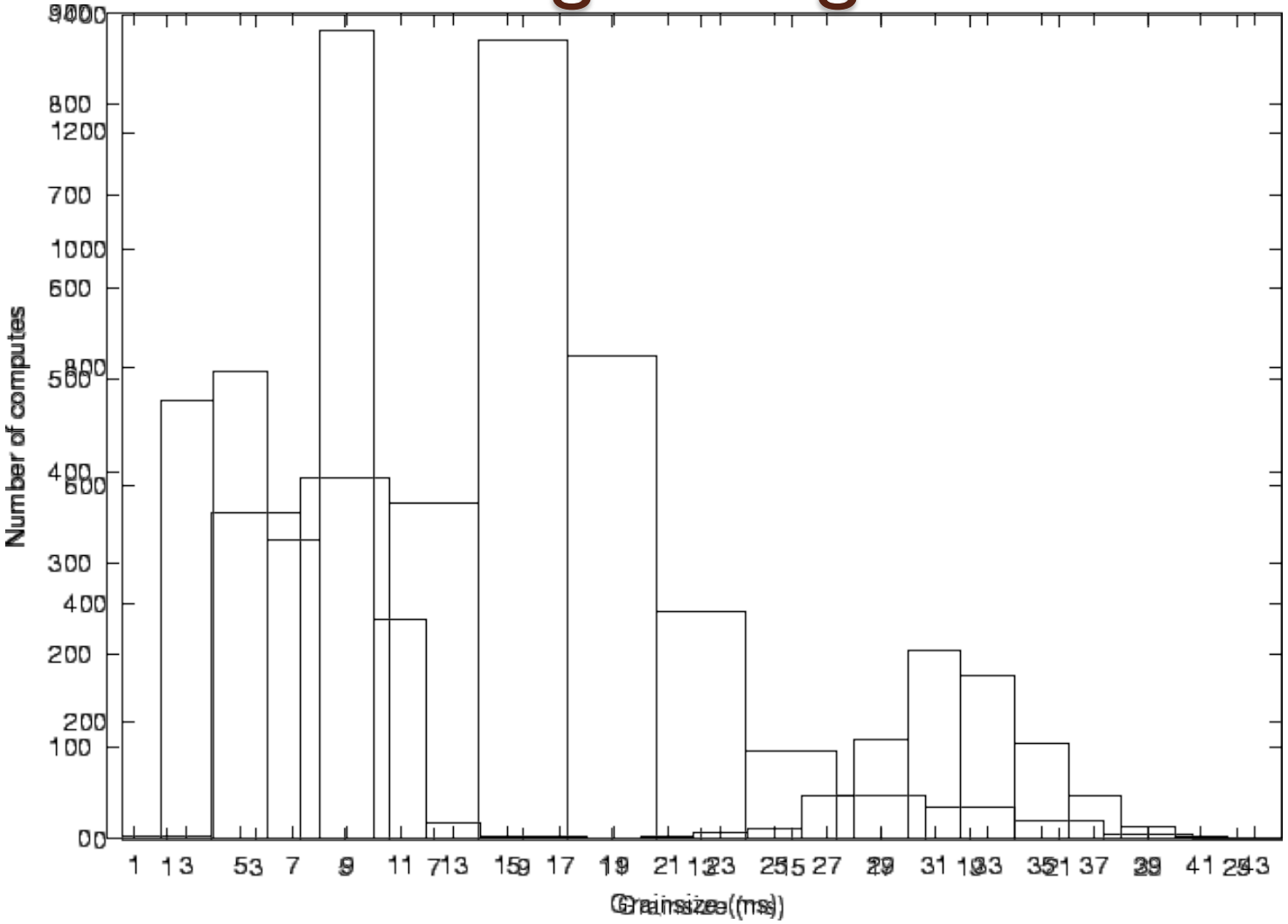


Case Study:

- Apparent load imbalance.
- No strategy appeared to solve imbalance.
- Picked overloaded processor timelines.*
- Found longer-than-expected activities.
- Longer activities associated with specific objects.
- Possible work grainsize distribution problems.

*As we will see later, not effective with large numbers of processors.

Case Study: Validation using Histograms



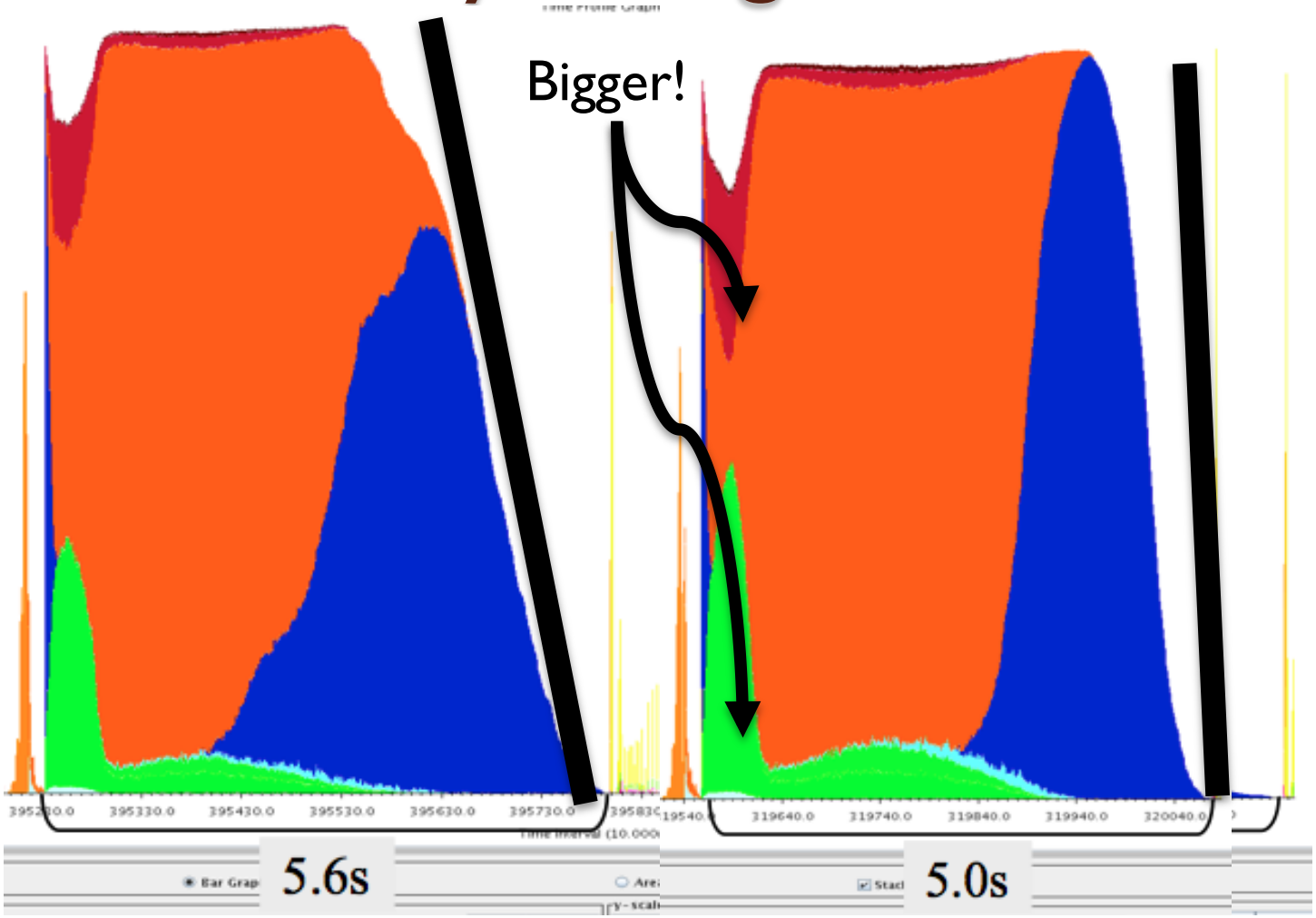
Effectiveness of Idiom

- Need to find way to pick out overloaded processors. Not scalable!
- Finding out if work grainsize was a problem simply required the histogram feature.

High Resolution Time Profiles

- Shows activity-overlap over time summed across all processors.
- Heuristics guide the search for visual cues for various potential problems:
  - Gradual downward slopes hint at possible load imbalance.
  - Gradual upward slopes hint at communication inefficiencies.
- At high resolution, gives insight into application sub-structure.

Case Study: Using Time Profiles



Possible Load Imbalance

After Greedy Load Balancing Strategy

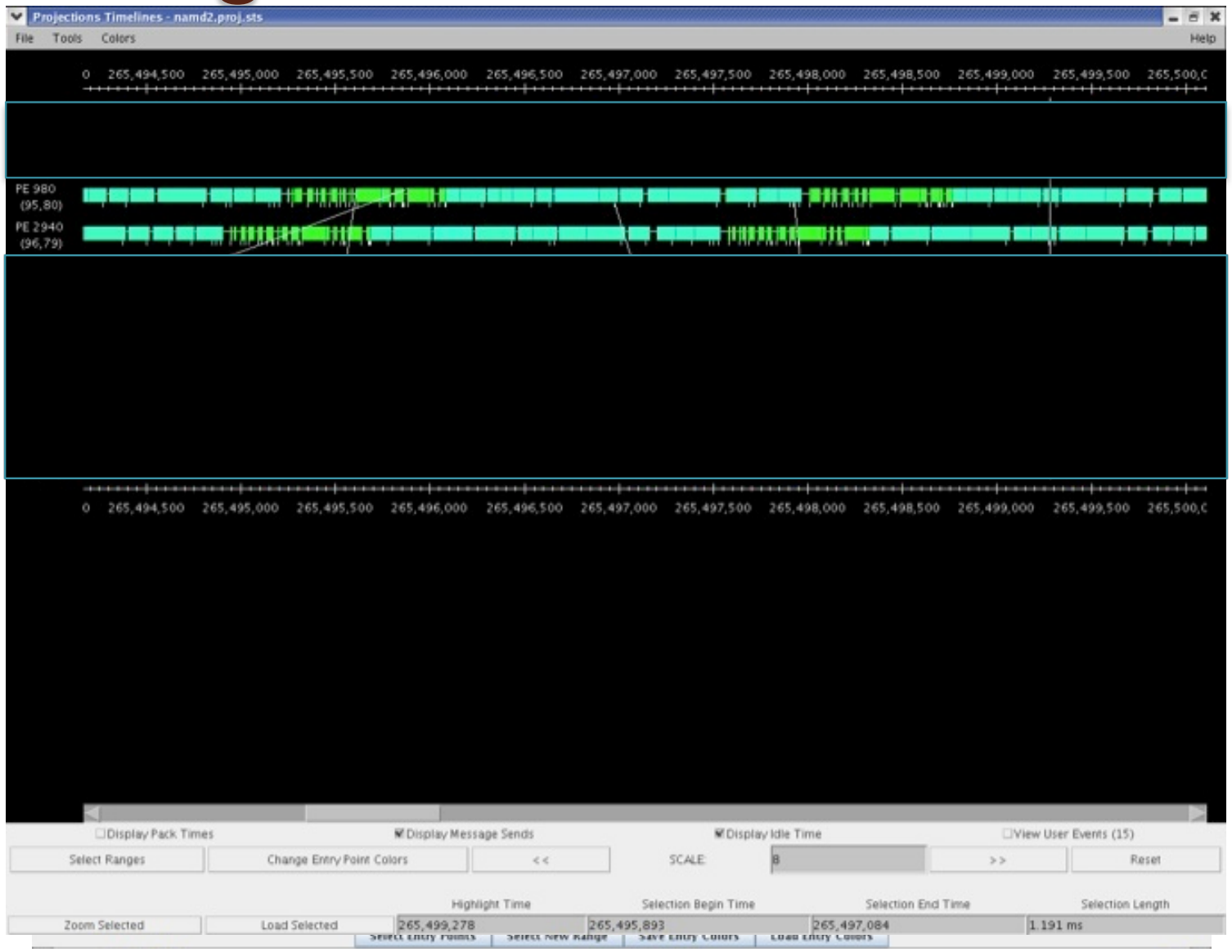
Finding Extreme or Unusual Processors

- A recurring theme in analysis idioms.
- Easy to pick out timelines in datasets with small numbers of processors.
- Examples of attributes and criteria:
  - Least idle processors.
  - Processors with late events.
  - Processors that behave very differently from the rest.

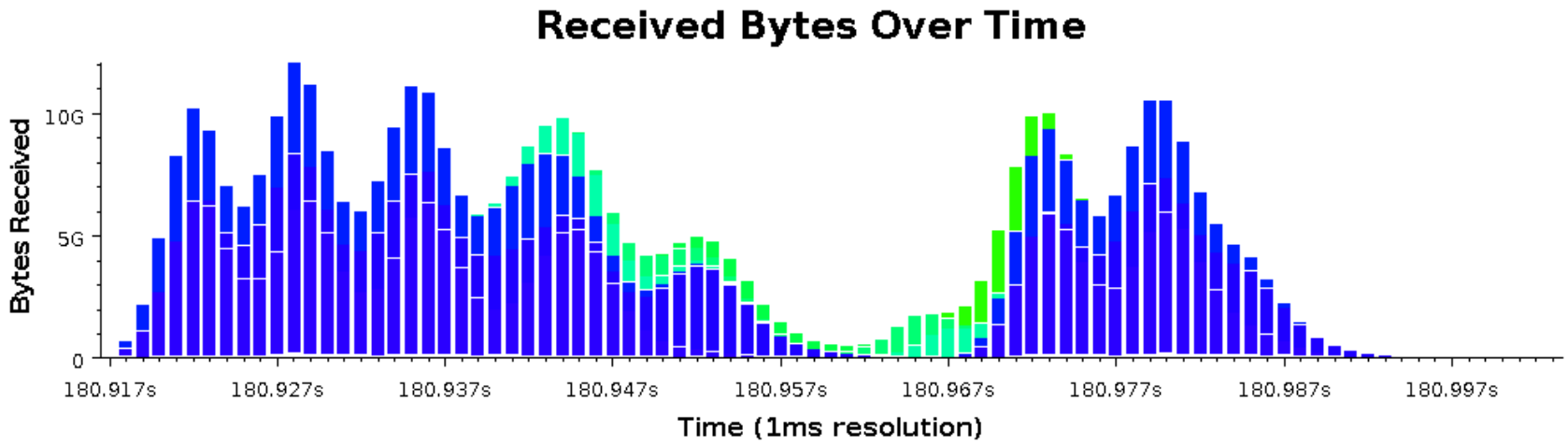
The Extrema Tool

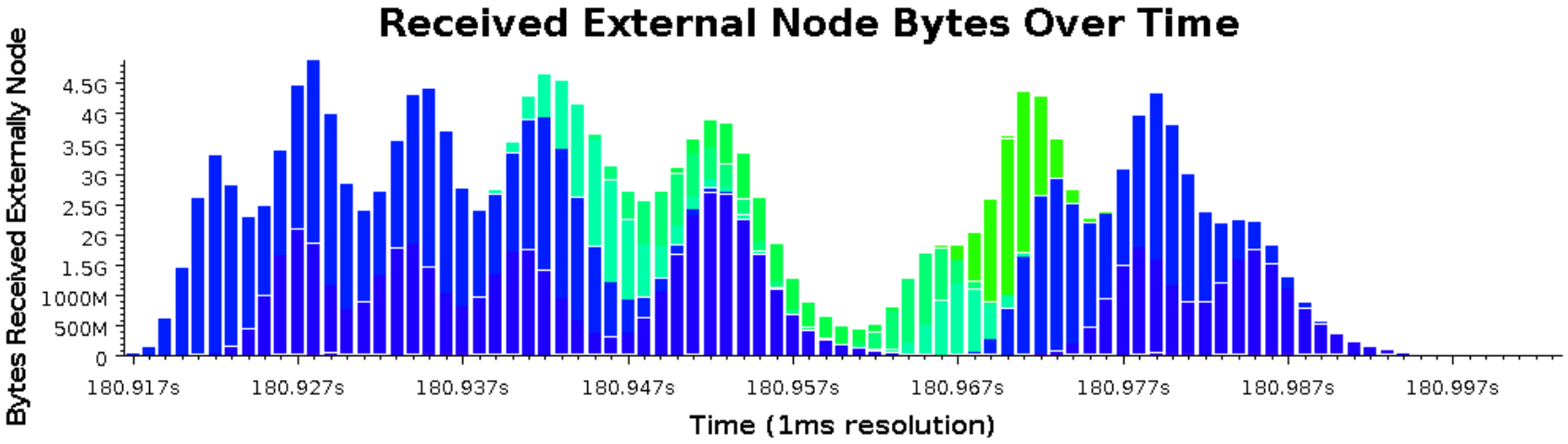
- Semi-automatically picks out interesting processors to display.
- Decisions based on analyst-specified criteria.
- Mouse-clicks on bars load interesting processors onto timeline.

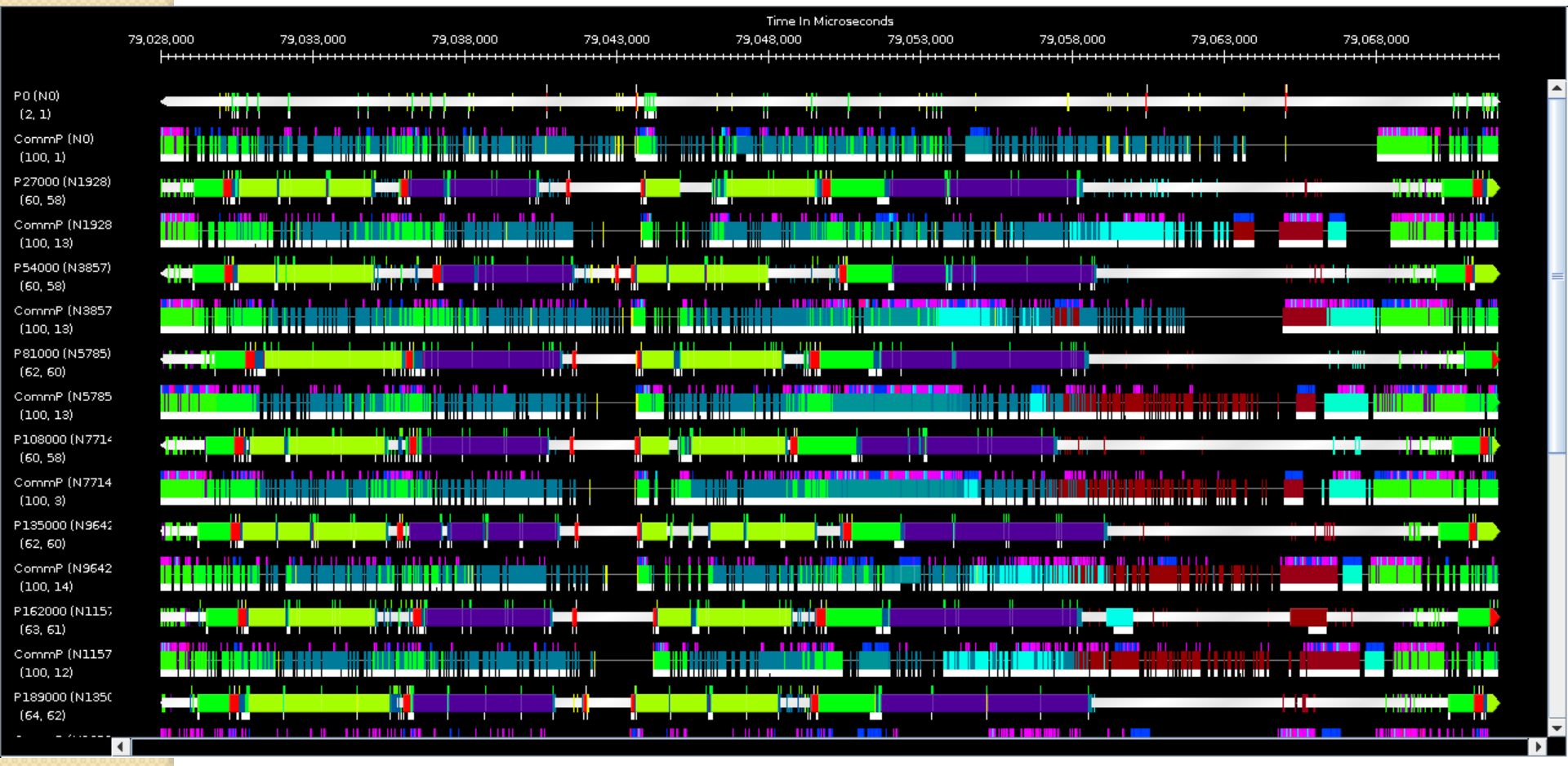
Using the Extrema Tool



Some recent examples: scalable views






Time In Microseconds
79,028,000
79,033,000
79,038,000
79,043,000
79,048,000
79,053,000
79,058,000
79,063,000
79,068,000
P0 (N0)
(2, 1)
CommP (N0)
(100, 1)
P27000 (N1928)
(60, 58)
CommP (N1928
(100, 13)
P54000 (N3857)
(60, 58)
CommP (N3857
(100, 13)
P81000 (N5785)
(62, 60)
CommP (N5785
(100, 13)
P108000 (N771
(60, 58)
CommP (N7714
(100, 3)
P135000 (N964
(62, 60)
CommP (N9642
(100, 14)
P162000 (N115
(63, 61)
CommP (N1157
(100, 12)
P189000 (N135
(64, 62)

Scalable Tool Features: Conclusions

- Effective analysis idioms must avoid non-scalable features.
- Histograms, Time Profiles and the Extrema Tool offer scalable features in support of idioms.

Data Reduction

- Normally, scalable tool features are used with full event traces.
- What happens if full event traces get too large?
- We can:
  - Choose to keep event traces for only a subset of processors.
  - Replace event traces of discarded processors with interval-based profiles.

Choosing Useful Processor Subset (1/2)

- What are the challenges?
  - No a priori information about performance problems in dataset.
  - Chosen processors need to capture details of performance problems.

Choosing Useful Processor Subsets (2/2)

- Observations:
  - Processors tend to form equivalence classes with respect to performance behavior.
  - Clustering can be used to discover equivalence classes in performance data.
  - Outliers in clusters may be good candidates for capturing performance problems.

Applying *k*-Means Clustering to Performance Data

- Treat the vector of recorded performance metric values on each processor as a data point for clustering.
- Measure similarity between two data points using the Euclidean Distance between the two metric vectors.
- Given *k* clusters to be found, the goal is to minimize similarity values between all data points and the centroids of the *k* clusters.

Choosing from Clusters

- Choosing Cluster Outliers.
  - Pick processors furthest from cluster centroid.
  - Number chosen by proportion of cluster size.
- Choosing Cluster Exemplars.
  - Pick a single processor closest to the cluster centroid.
- Outliers + Exemplars = Reduced Dataset.

Applying *k*-Means Clustering Online

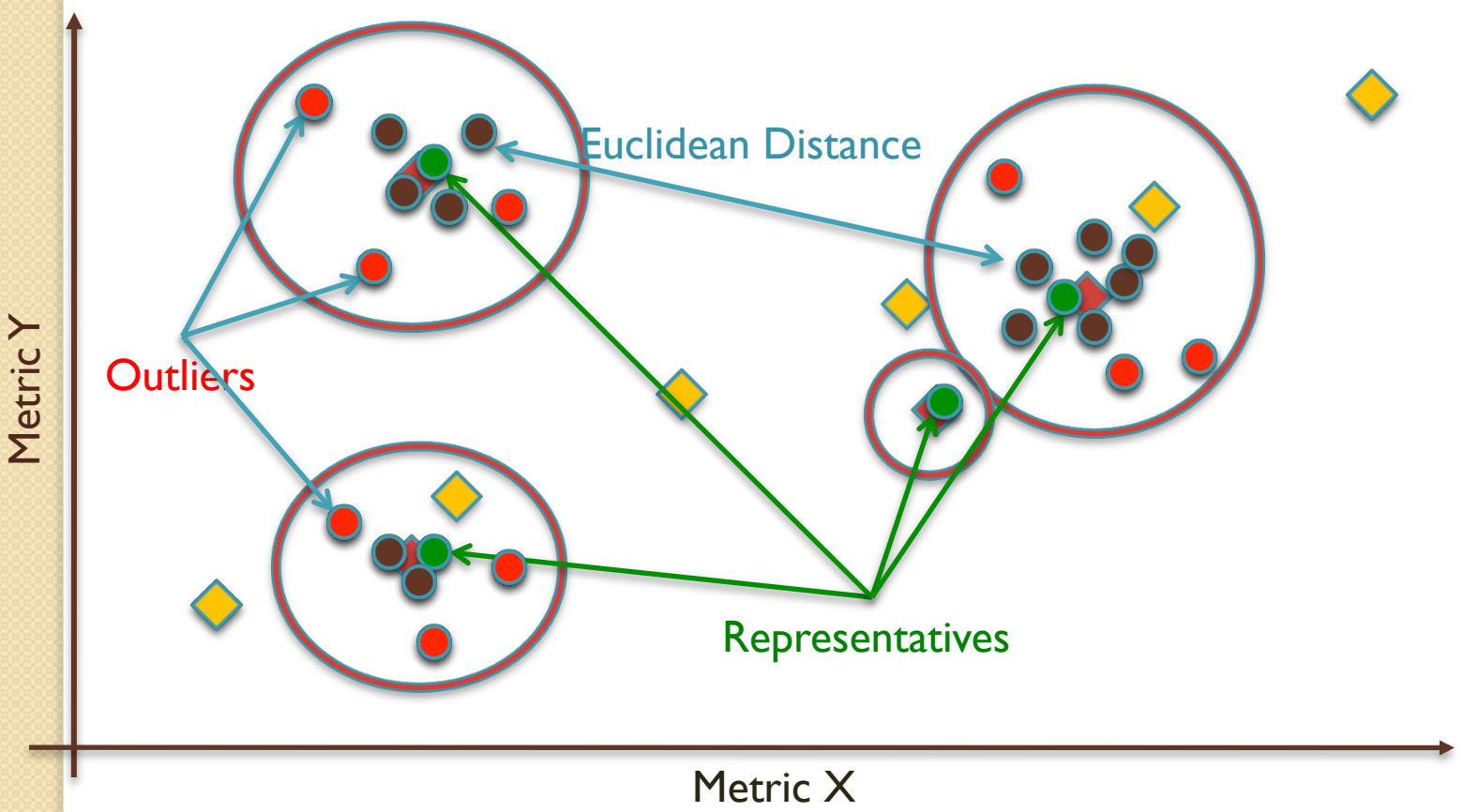
- "Death-bed" or "Moriens" analysis:
  - Just before the program terminates, we have all performance logs, and a huge parallel m/c
  - This is the simplest example of Moriens analysis
- Decisions on data retention are made before data is written to disk.
- Requires a low-overhead and scalable parallel *k*-Means algorithm

Important *k*-Means Parameters

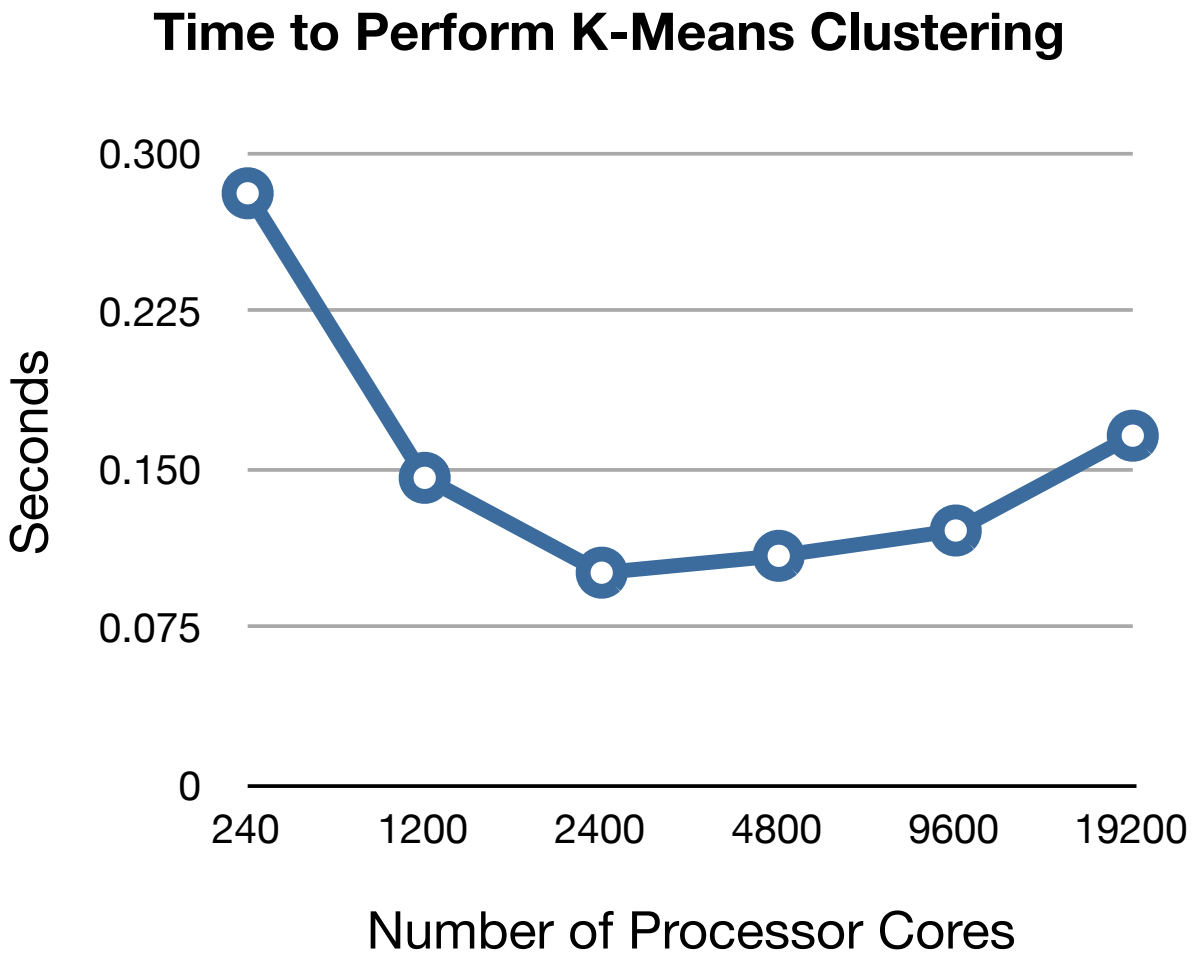
- Choice of metrics from domains:
  - Activity time.
  - Communication volume (bytes).
  - Communication (number of messages).

- Normalization of metrics:
  - Same metric domain = no normalization.
  - *Min-max* normalization across different metric domains to remove inter-domain bias.

Equivalence Class Discovery



Overhead of parallel k-Means



Data Reduction: Conclusions

- Showed combination of techniques for online data reduction is effective*.
- Choice of processors included in reduced datasets can be refined and improved
  - Include communicating processors.
  - Include processors on critical path.
- Consideration of application phases can further improve quality of reduced dataset.

*Chee Wai Lee, Celso Mendes and Laxmikant V. Kale. **Towards Scalable Performance Analysis and Visualization through Data Reduction.** 13th International Workshop on High-Level Parallel Programming Models and Supportive Environments, Miami, Florida, USA, April 2008.

Live Streaming of Performance Data

- Live Streaming mitigates need to store a large volume of performance data.
- Live Streaming enables analysis idioms that provide animated insight into the trends application behavior.
- Live Streaming also enables idioms for the observation of unanticipated problems, possibly over a long run.

Challenges to Live Streaming

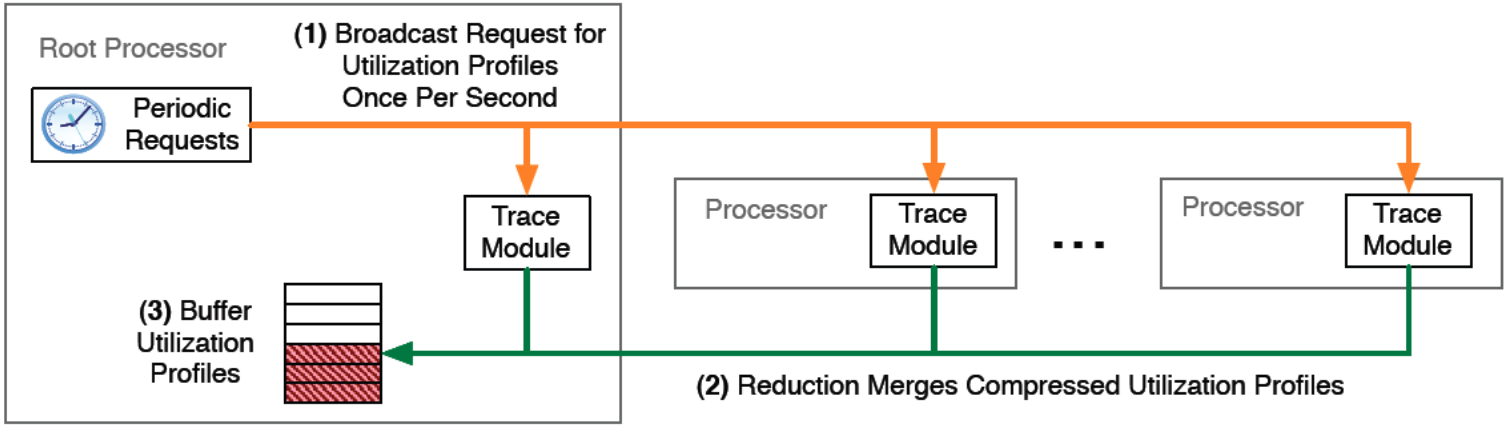
- Must maintain low overhead for performance data to be recorded, pre-processed and disposed-of.
- Need efficient mechanism for performance data to be sent via out-of-band channels to one (or a few) processors for delivery to a remote client.

Enabling Mechanisms

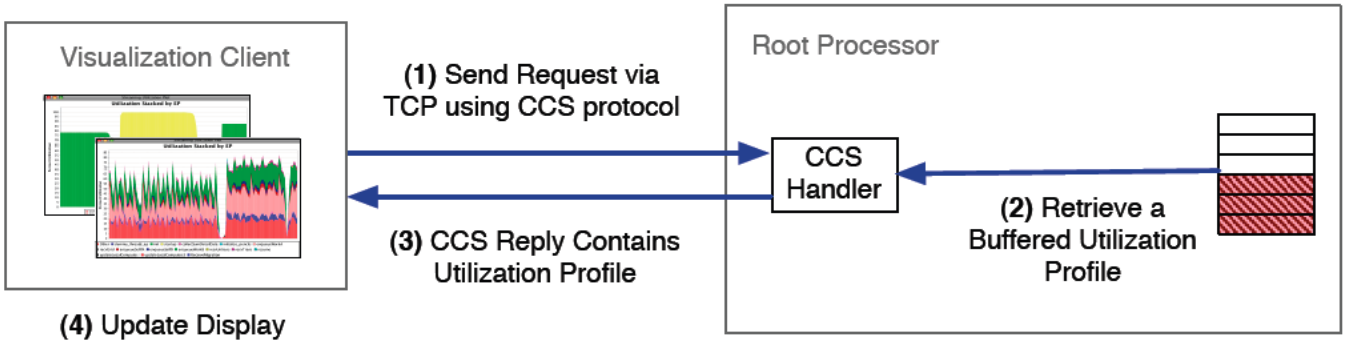
- Charm++ adaptive runtime as medium for scalable and efficient:
  - Control signal delivery.
  - Performance data capture and delivery.
- Converse Client-Server (CCS) enables remote interaction with running Charm++ application through a socket opened by the runtime.

Live Streaming System Overview

A) Gathering Performance Data in Parallel Runtime System:

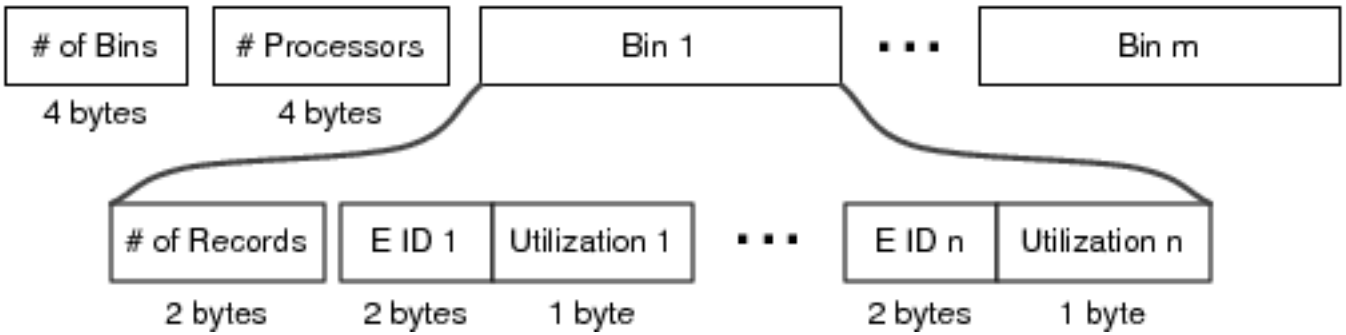


B) Visualizing Performance Data:

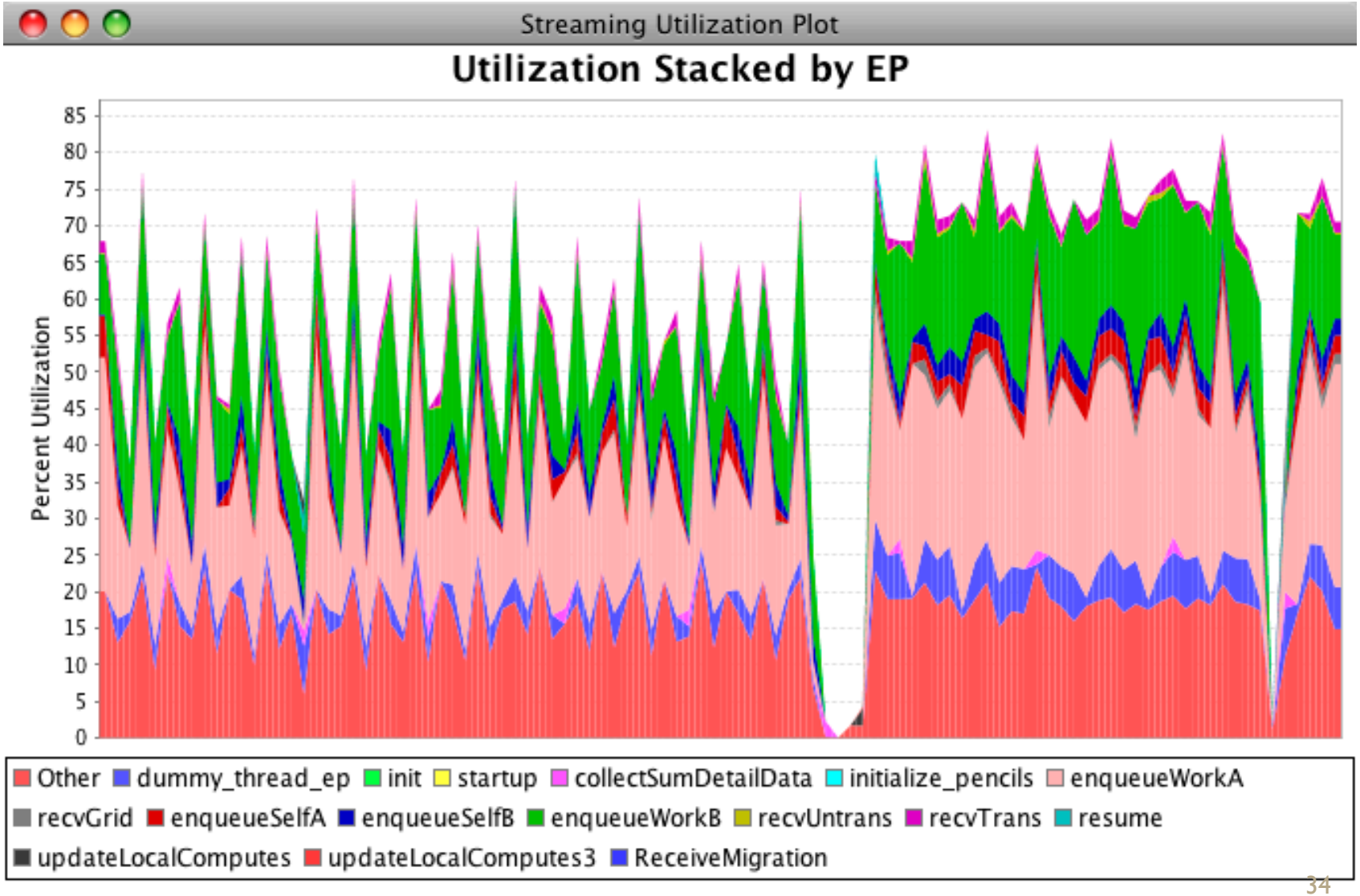


What is Streamed?

- A Utilization Profile similar to high resolution Time Profiles.
- Performance data is compressed by only considering significant metrics in a special format.
- Special reduction client merges data from multiple processors.



Visualization



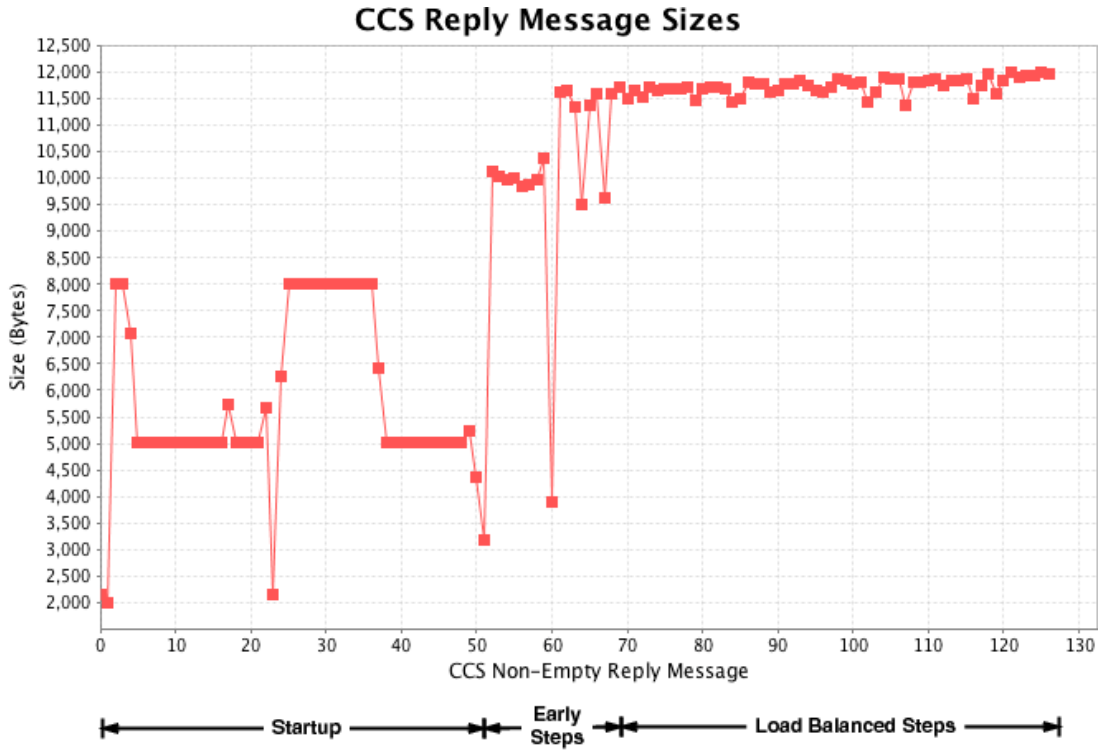
Overheads (1/2)

% Overhead when compared to baseline system: Same application with no performance instrumentation.

	512	1024	2048	4096	8192
With instrumentation, data reductions to root **with remote client** attached.	0.94%	0.17%	-0.26%	0.16%	0.83%
With instrumentation, data reductions to root but **no remote client** attached.	0.58%	-0.17%	0.37%	1.14%	0.99%

Overheads (2/2)

For bandwidth consumed when streaming performance data to the remote visualization client.



Live Streaming: Conclusions*

- Adaptive runtime allowed out-of-band collection of performance data while in user-space.
- Achieved with very low overhead and bandwidth requirements.

*Isaac Dooley, Chee Wai Lee, and Laxmikant V. Kale. **Continuous Performance Monitoring for Large-Scale Parallel Applications**. Accepted for publication at HiPC 2009, December-2009.

Repeated Large-Scale Hypothesis Testing

- Large-Scale runs are expensive:
  - Job submission of very wide jobs to supercomputing facilities.
  - CPU resources consumed by very wide jobs.
- How do we make repeated but inexpensive hypothesis testing experiments?

Trace-based Simulation

- Capture event dependency logs from a baseline application run.
- Simulation produces performance event traces from event dependency logs.

Advantages

- The time and memory requirements at simulation time are divorced from requirements at execution time.
- Simulation can be executed on fewer processors.
- Simulation can be executed on a cluster of workstations and still produce the same predictions.

Using the BigSim Framework (1/2)

- BigSim emulator captures:
  - Relative event time stamps.
  - Message dependencies.
  - Event dependencies.
- BigSim emulator produces event dependency logs.

Using the BigSim Framework (2/2)

- BigSim simulator uses a PDES engine to process event dependency logs to predict performance.
- BigSim simulator can generate performance event traces based on the predicted run.

Examples of Hypothesis Testing Possible

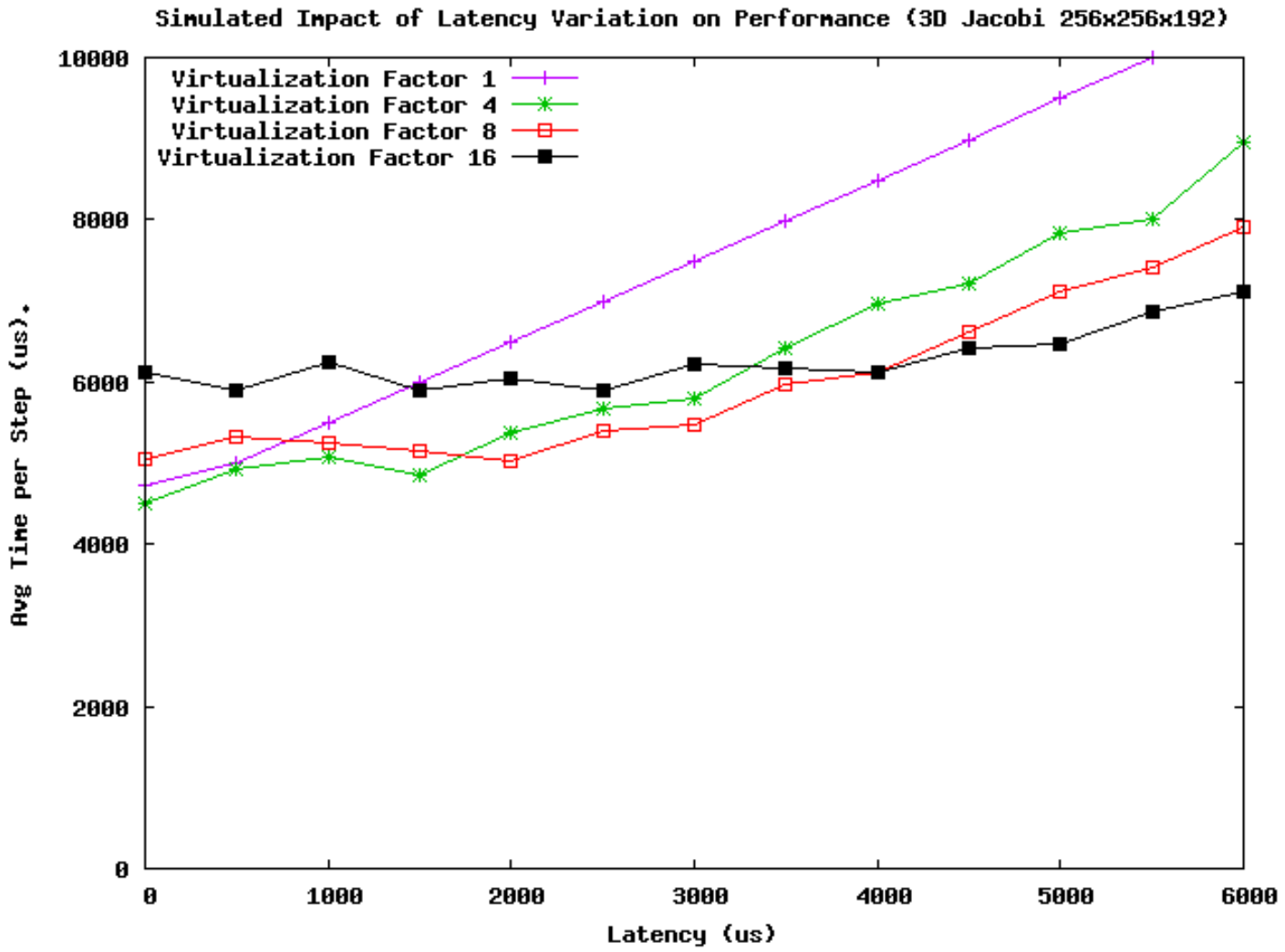
- Hypothetical Hardware changes:
  - Communication Latency.
  - Network properties.

- Hypothetical Software changes:
  - Different load balancing strategies.
  - Different initial object placement.
  - Different number of processors with the same object decomposition.

Example: Discovering Latency Trends

- Study the effects of network latency on performance of seven-point stencil computation
- For each of the data-points on the plots on next few slide/s
  - You have full traces
  - Can do projections analysis as if you ran it on the modified machine (with lower/higher latency)

Latency Trends – Jacobi 3d 256x256x192 on 48 pes



Summary

- Scalable views can be effective tools
  - Histograms, time-profiles, …
- Data reduction via on-line analysis
  - Parallel k-means, Sampling
- Live analysis: helped by message-driven execution
- Traces can be used for simulation:
  - what-if analysis via BigSim

Further Thoughts

- Future: a lot more emphasis on moriens (death-bed) analysis
  - Requires more automation of the analysis process
- More "Automated Expert Analysis"
  - Topic of 1994 thesis on projections
- Message-driven execution or communication layer integration for tools communication
  - No "out of band" issues
- Another grand challenge/s:
  - How to get grad students interested in tools research?
  - How to get funding in this area?
    - All our work has been (mostly) unfunded or only indirectly funded