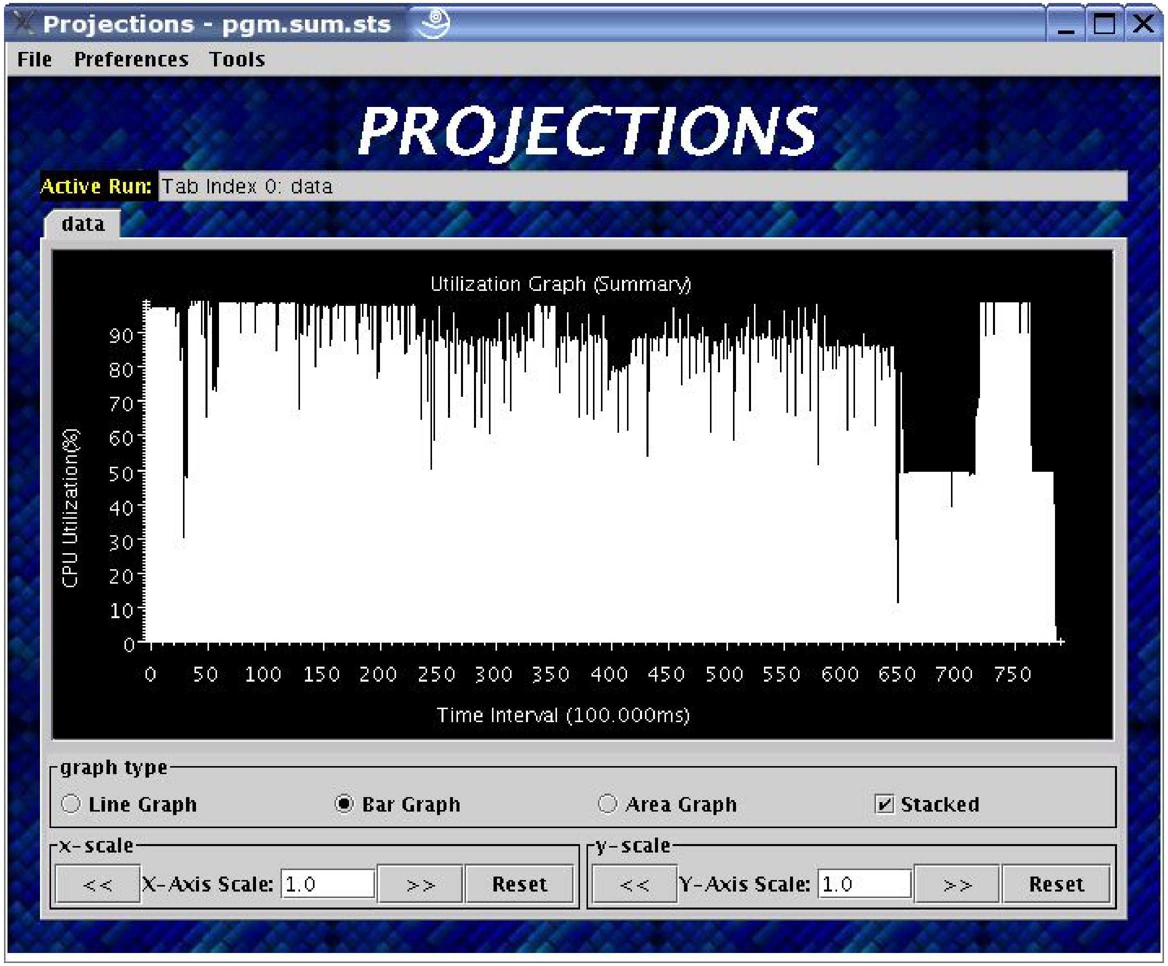
# Enhancing Scalability for Charm++ Performance Tools

Chee Wai Lee
PPL, UIUC

# Outline

- Projections: Performance Analysis for Charm++
- Studying Larger Applications on Larger Machines
- Data Scalability
- Visualization Scalability
- Future Work

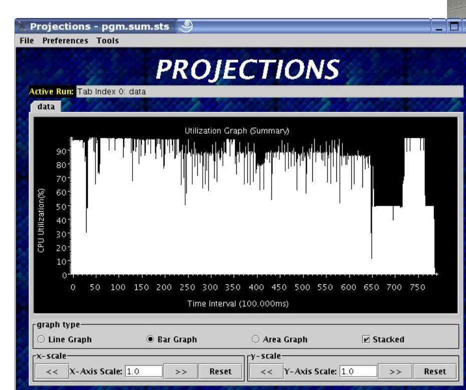
# Projections: A Performance Analysis Tool for Charm++ Applications



# Features

- Automatic Instrumentation
- Multiple levels of data resolution decided at application link-time.
- Options at run-time.
- Data from logs studied using Java visualizer in a human-centric approach.

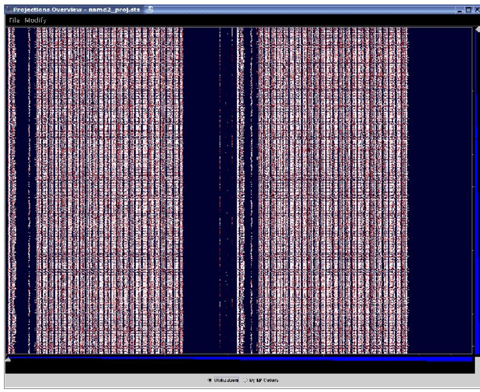
Submits Charm++ job with analysis modules linked into application.



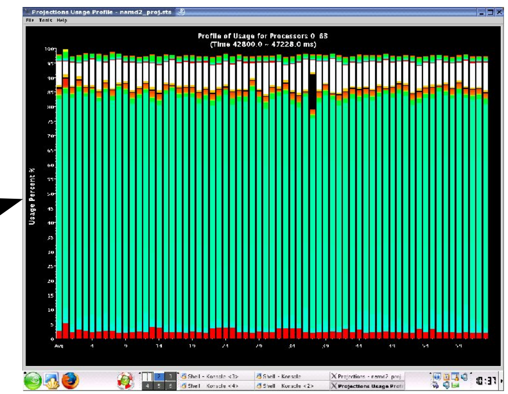

Trace Logs generated and studied.

# Rich Visualization Support

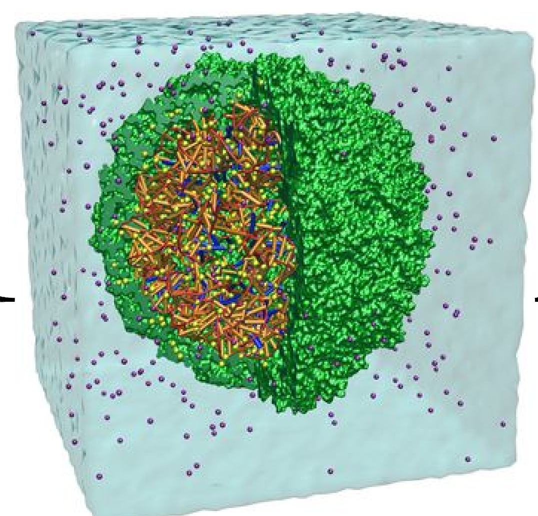
Highly detailed *Timeline* of selected processors



*Overview* of total workload per processor over time

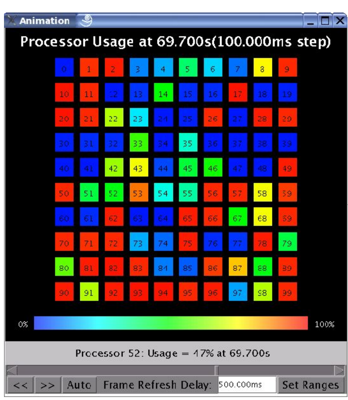


*Usage Profile* of each task per processor

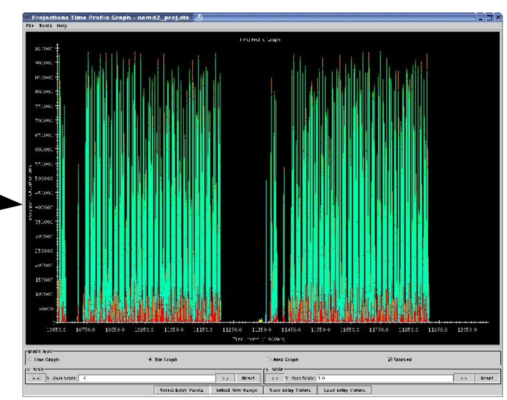


Theoretical and Computational Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

NAMD Simulation Executed

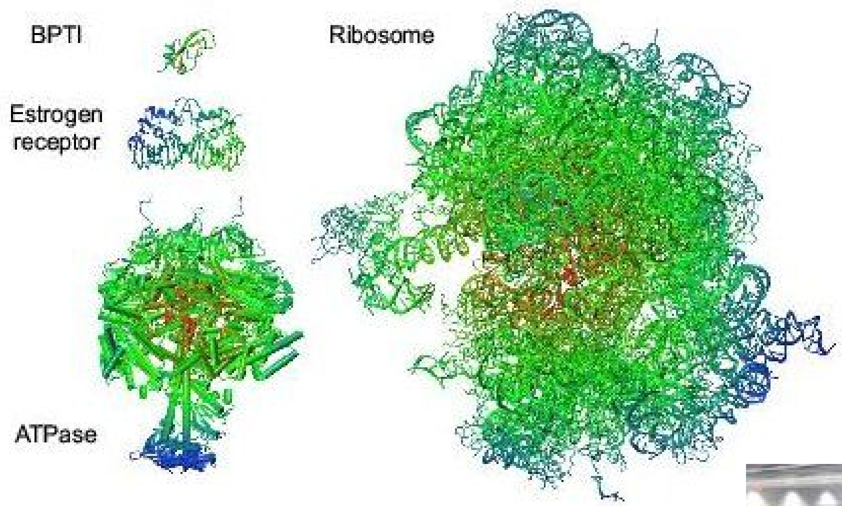



*Animation* of Proccessor Loads



*Time Profile* of task activity over time summed across processors

# Studying Larger Applications on Larger Machines




Full-atom molecular systems simulated have grown from 3K atoms (BPTI) to 2.8M atoms (Ribosome).

Cray XT3 at PSC

SGI Altix at NCSA

Blue Gene/L at LLNL

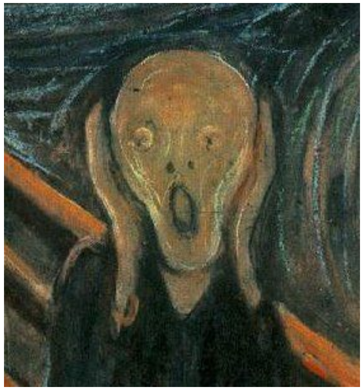
# Data Scalability

# Performance Data Volume Growth

- Total data collected grows when we:
  - study longer-running applications.
    - this is usually unnecessary as will be explained later
  - study larger scientific simulation systems.
    - due to an increase to the total amount of work done
  - increase the number of processors studied.
    - usually due to increased communication events

# What are the problems with data growth?

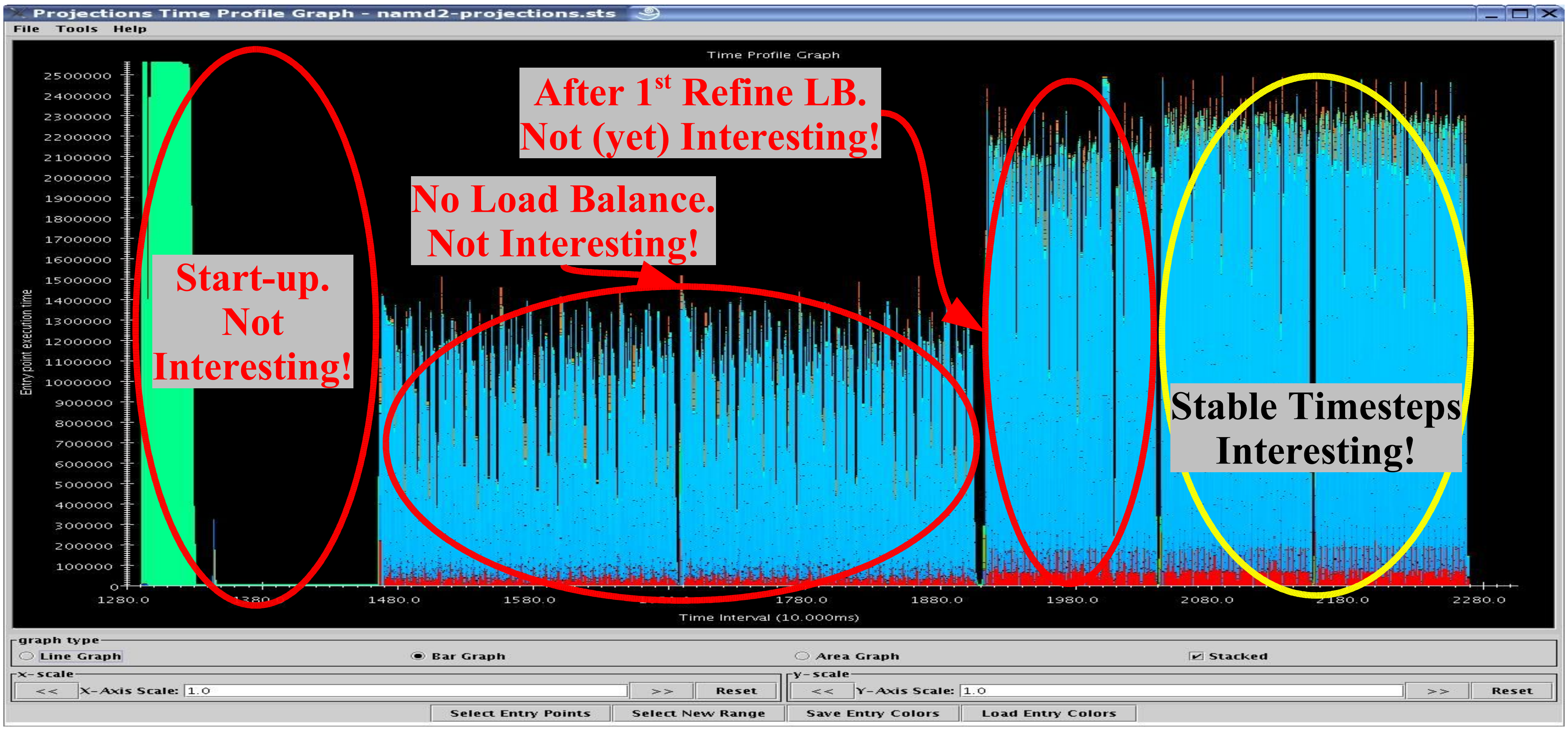
- Large data sets take time to:
  - compress/package.
  - transfer from the National Center machines to local machines for analysis. (e.g. it took about 4 min to transfer a 400 MB compressed file from PSC's Cray XT3 to my laptop)
- Visualization/Analysis tool has to deal with more files/data.
- Ability to interactively determine performance issues degraded.



# Longer Running Applications

# Do we need to study the entire execution?

- Generally, we want to study a portion of a run that reflects/impacts overall performance.



# Do we need to study the entire execution? (2)

- It is also generally not possible to trace the entire run of an application. Take molecular dynamics applications as an example:
  - molecular dynamics requires femto-second time resolutions.
  - meaningful biological processes is of the order of 100 nano-seconds and beyond.
  - 200 timesteps of a 327K-atom simulation generates 2.8GB of data on a 2048-processors run!
  - What would 100,000,000 timesteps generate?!

# Larger Simulation Systems
# and
# Larger Processor Counts

# Data Growth: How much?

Size
Growth
due to
processors

| nCPUs | apoa1 (92K atoms)* | | f1-atpase (327K atoms)* | |
|---|---|---|---|---|
| | max filesize (KB) | total datasize (MB) | max filesize (KB) | total datasize (MB) |
| 512 | 3,500 | 827 | 6,910 | 1,800 |
| 1024 | 3,400 | 938 | 5,580 | 2,200 |
| 2048 | 3,360 | 1,200 | 5,440 | 2,800 |

Size Growth
due to simulation system

* over 200 iterations

# Data Growth Due To Simulation Size

- Data Mining? Profiling? Data Aggregation?
- What are the problems?
  - larger amount of total work (events) per iteration (our "unit" of interest).
  - limit to how much data can be reduced before the selected time-range is no longer representative.

# Processor Growth: Our Approach

- Key Idea: Trim data by finding a suitable subset of processors from which to retain detailed event traces at *runtime*.
- How?
  - keep detailed log information about badly-behaved (extrema) processors through heuristics.
  - keep detailed log information of a representative* set of processors through *k-means clustering* algorithms.
  - all other processors just stores aggregated data (~50k per file).

* "representative" in the sense that the subset reflects overall application performance.

# Finding Extrema Processors

- Heuristics can be simple. (e.g. it is often useful to find processors in Charm++ with the *least idle time*.)
- We may also use more complex heuristics to help locate processors with significant deviant entry methods.

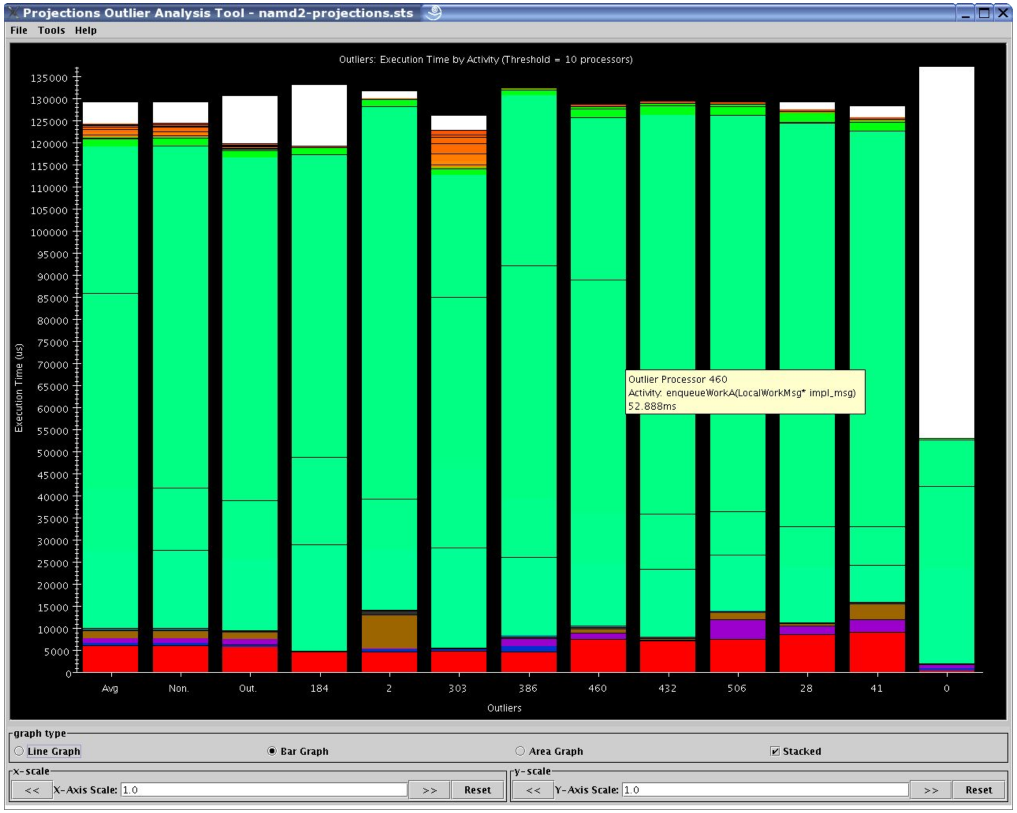
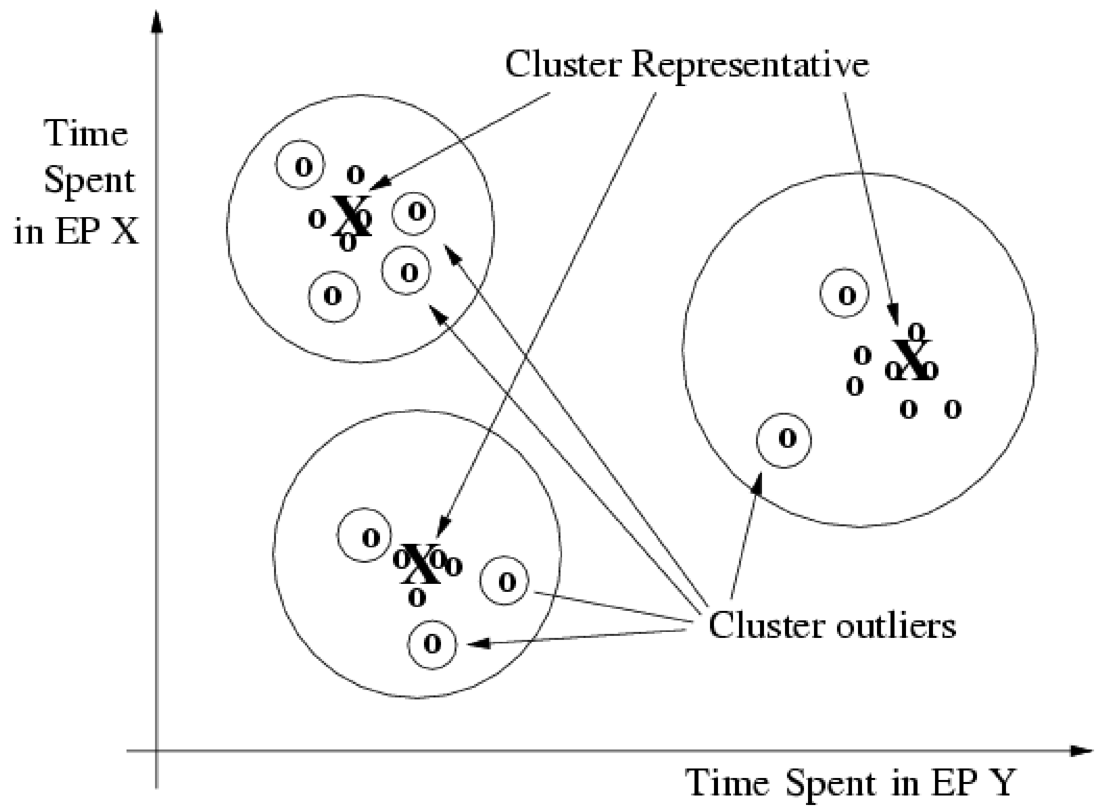



Illustration of extrema processors identified by complex heuristic.

# Finding a Representative Set of Processors

- We employ *k-means clustering* to do this.




The figure shows one possible way of interpreting cluster results

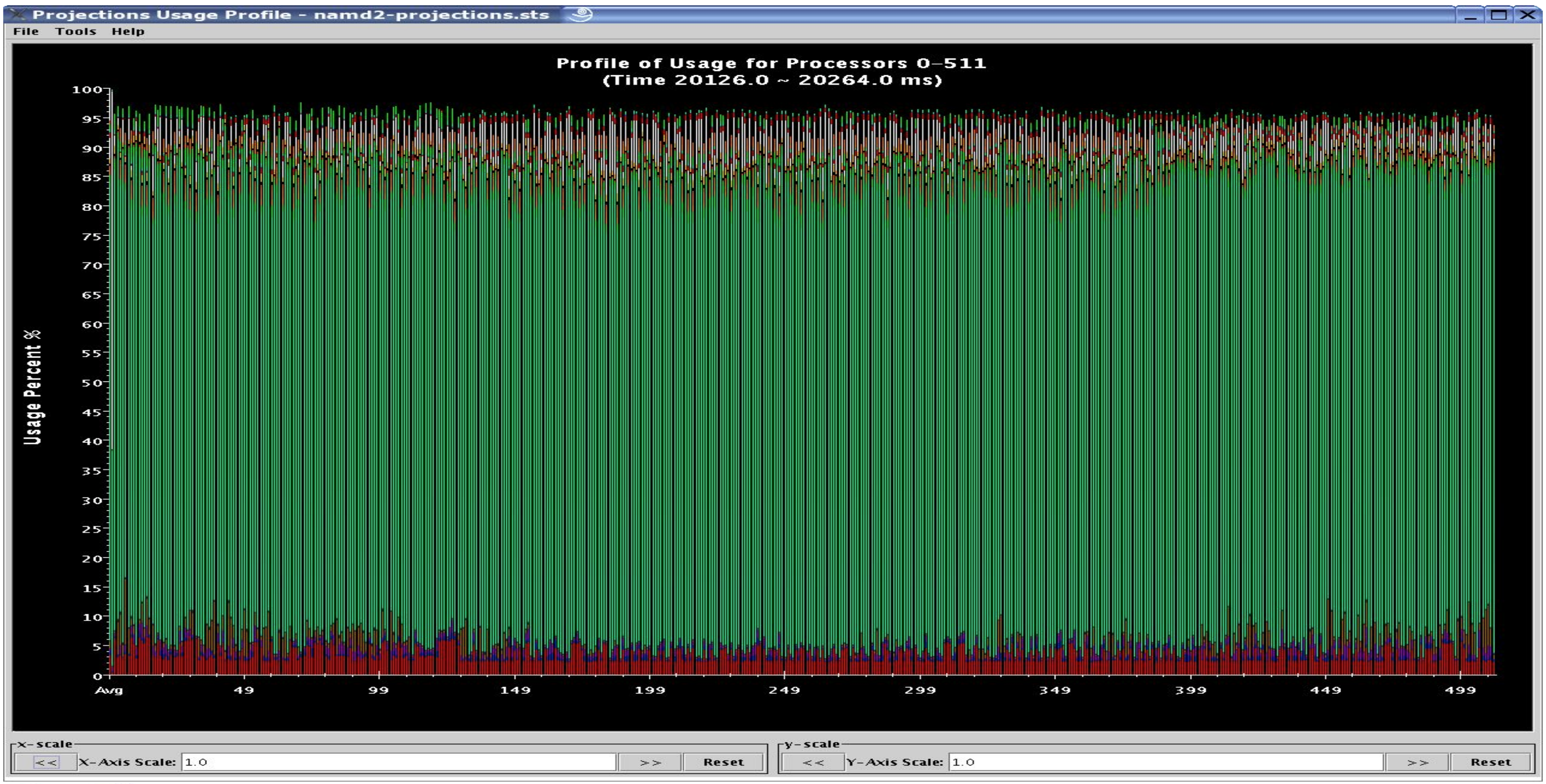
# Visualization/Analysis Scalability

# Challenges

- How not to annoy the analysts or waste their time. (e.g. waiting for 8000 log files to load is no fun!)
- How to keep the analysis process interactive and responsive.
- How to avoid overwhelming the analysts with too much information.

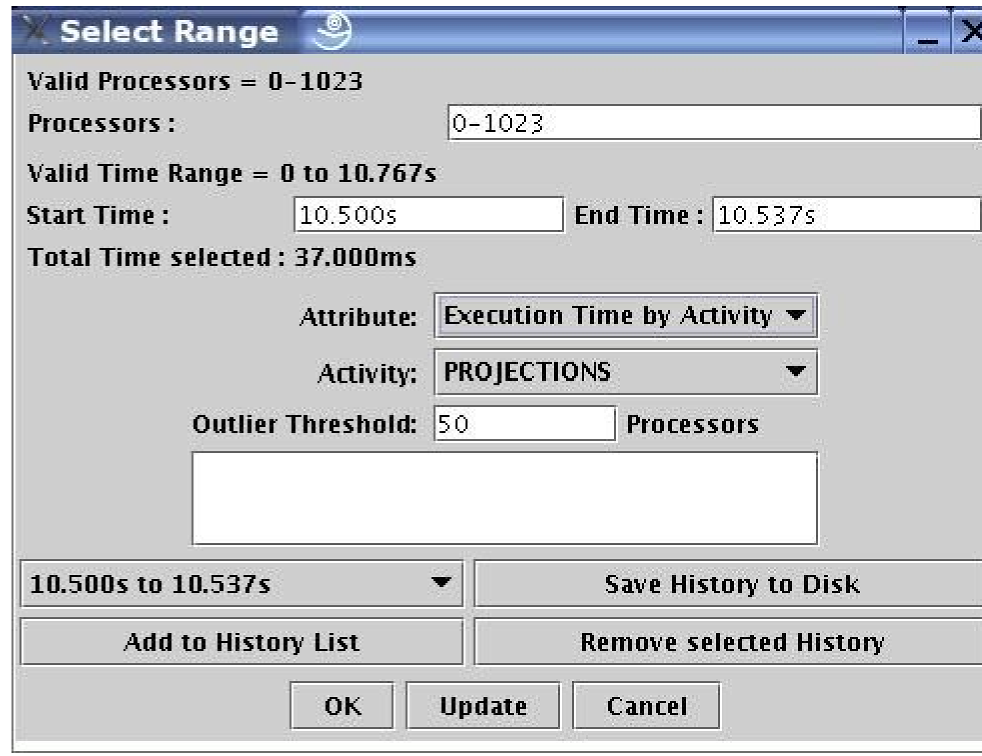
# EP utilization of 512 processors

**Challenge**: Find the top *k* mis-behaving processors to pull up on Timeline.

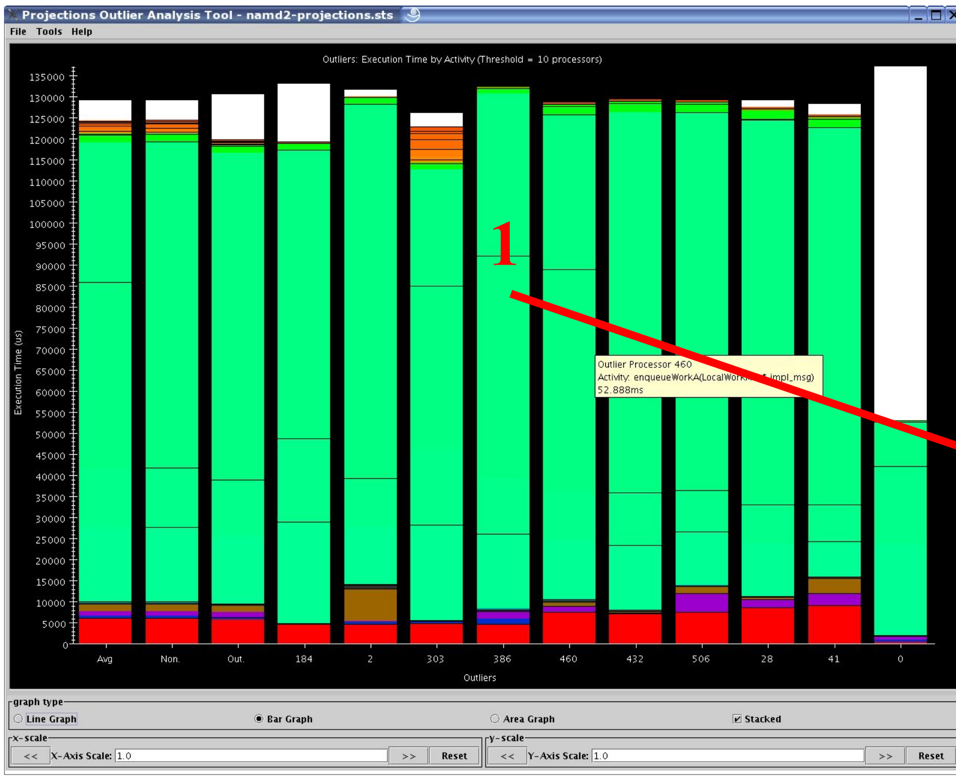


# Techniques

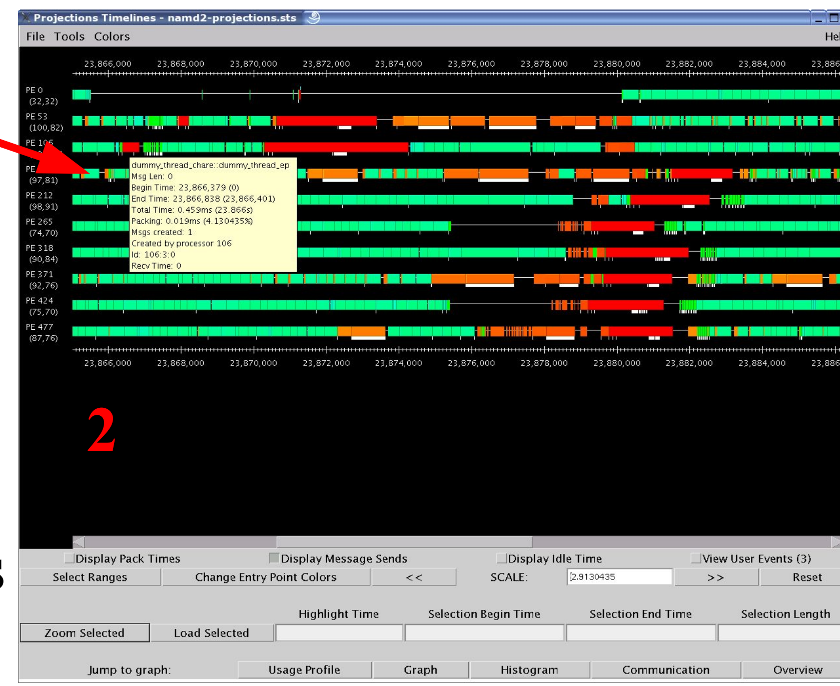
- We employ the same heuristics and k-means clustering *interactively* at visualization and analysis time.
- Works on full data sets or reduced data sets.



# Picking out Processors for Detailed Study




1) Click on a “badly-behaved” processor bar.




2) Timeline of the selected processor gets loaded into an existing set.

# Conclusion & Future Work

- New features recently used successfully for visualization/analysis.
- Applying the techniques to data reduction still requires refinement to clustering algorithms, heuristics.
- Parallel version of Projections.
- Tackling the problem of growing simulation sizes.