Scaling Applications to Massively Parallel Machines Using Projections Performance Analysis Tool

Laxmikant V. Kale´, Gengbin Zheng, Chee Wai Lee, Sameer Kumar

Minghui Liu **Feb 18 2021**

Motivation

Scale complex applications to large number of processors

- How to understand and optimize performance of complex applications running on thousands of cores
- The authors' approach:
    - Use visual and analytical feedbacks
    - Created a visualization component called Projections for Charm++

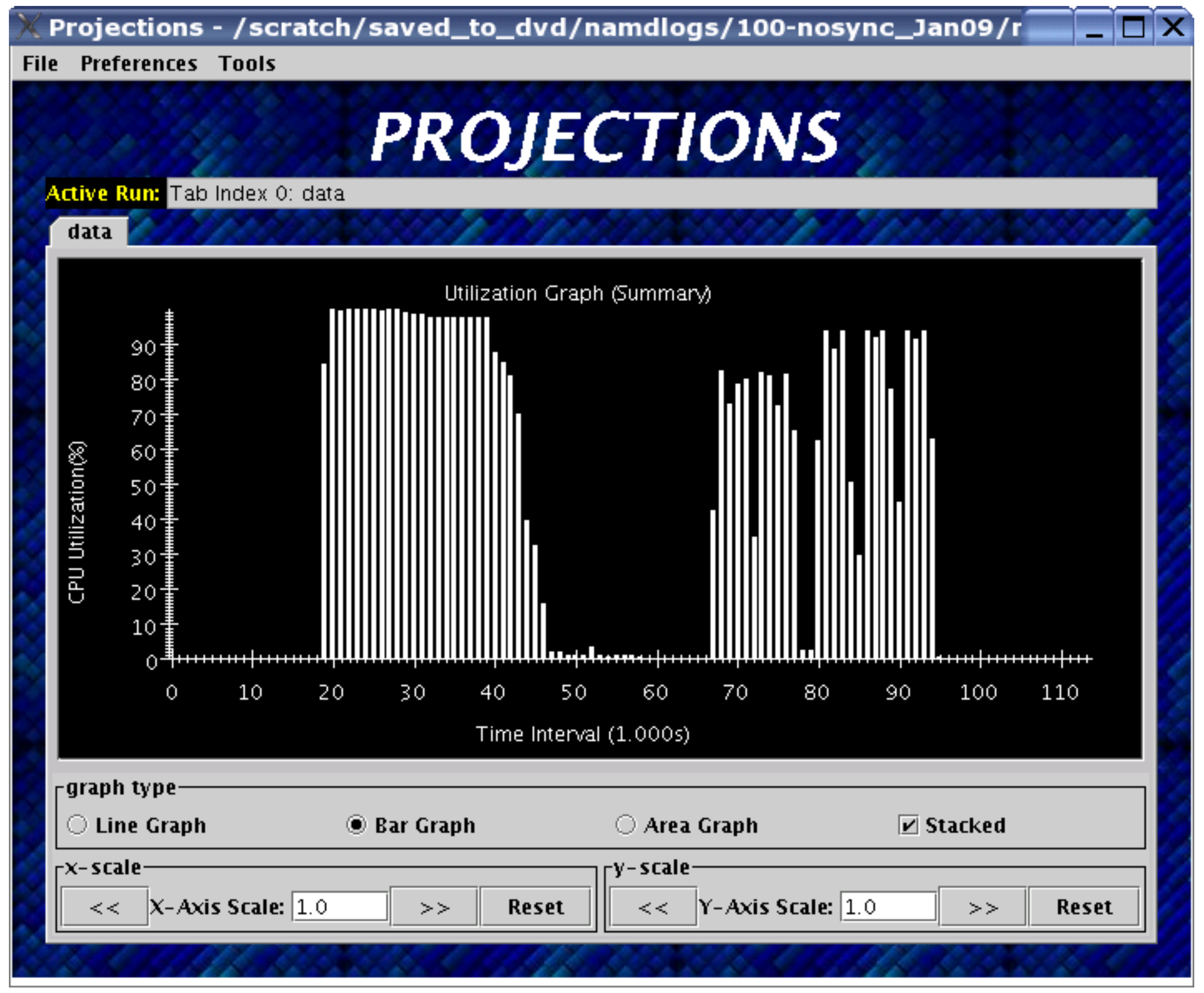
Charm++ Runtime

Two tracing modes

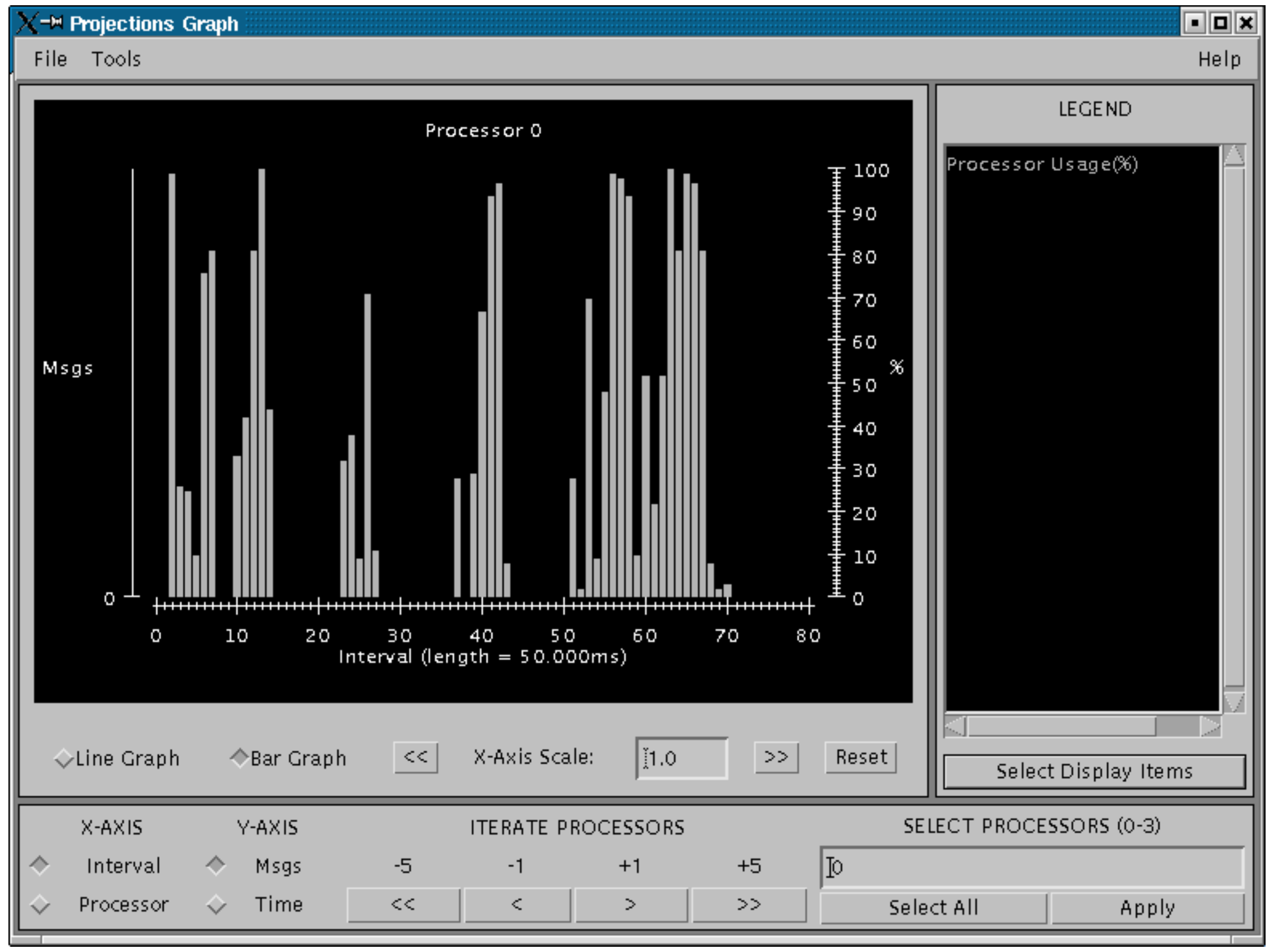
- Log mode
    - Each event is recorded in full detail, including timestamp
    - An “event” is entry method call, or message packing, unpacking, etc.
- Summary mode
    - A few lines of information per processor
    - Sum, max, avg execution time, number of times called, etc. for each entry method

Projections

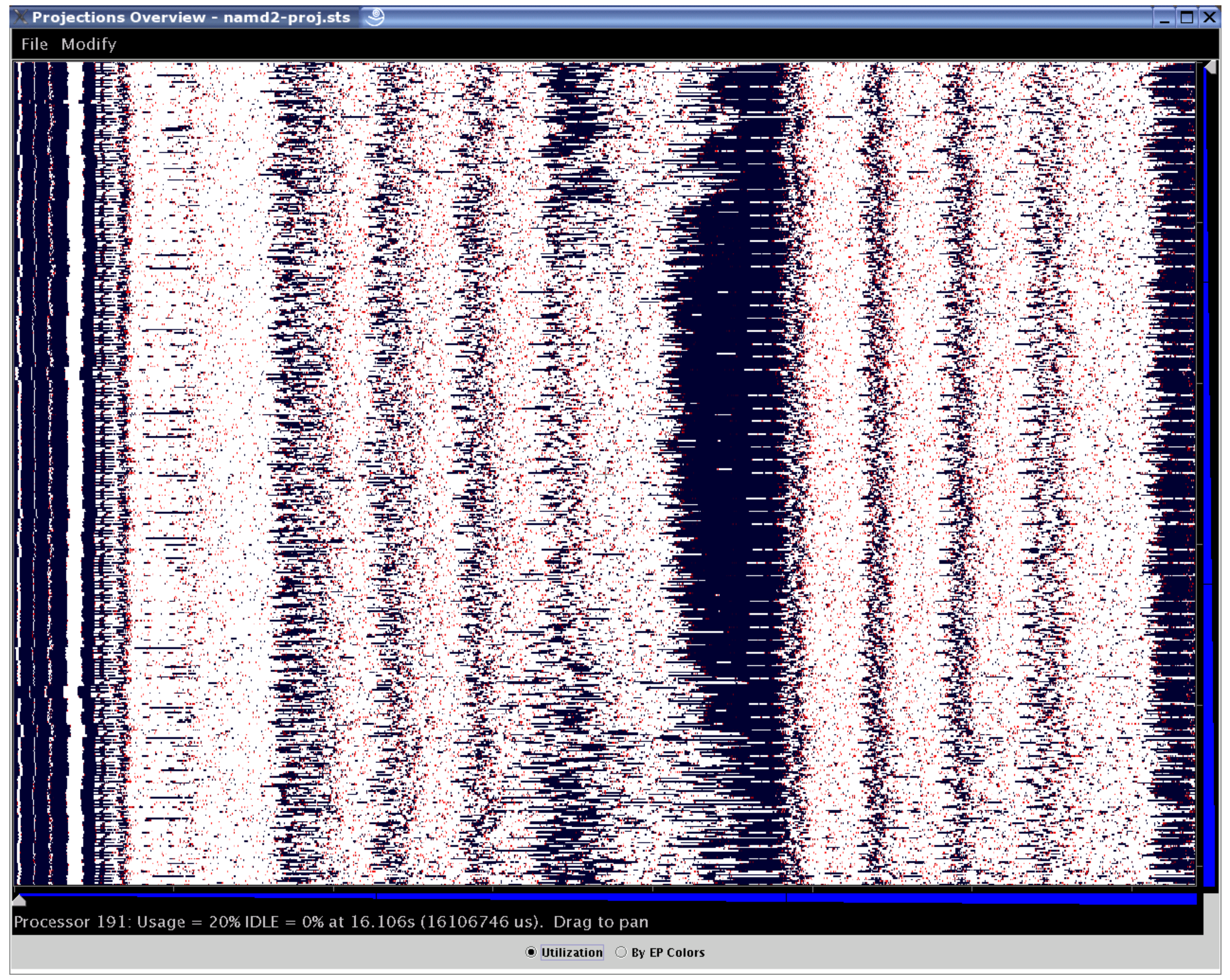
Visualization



Summary View
Average utilization over time



Graph View

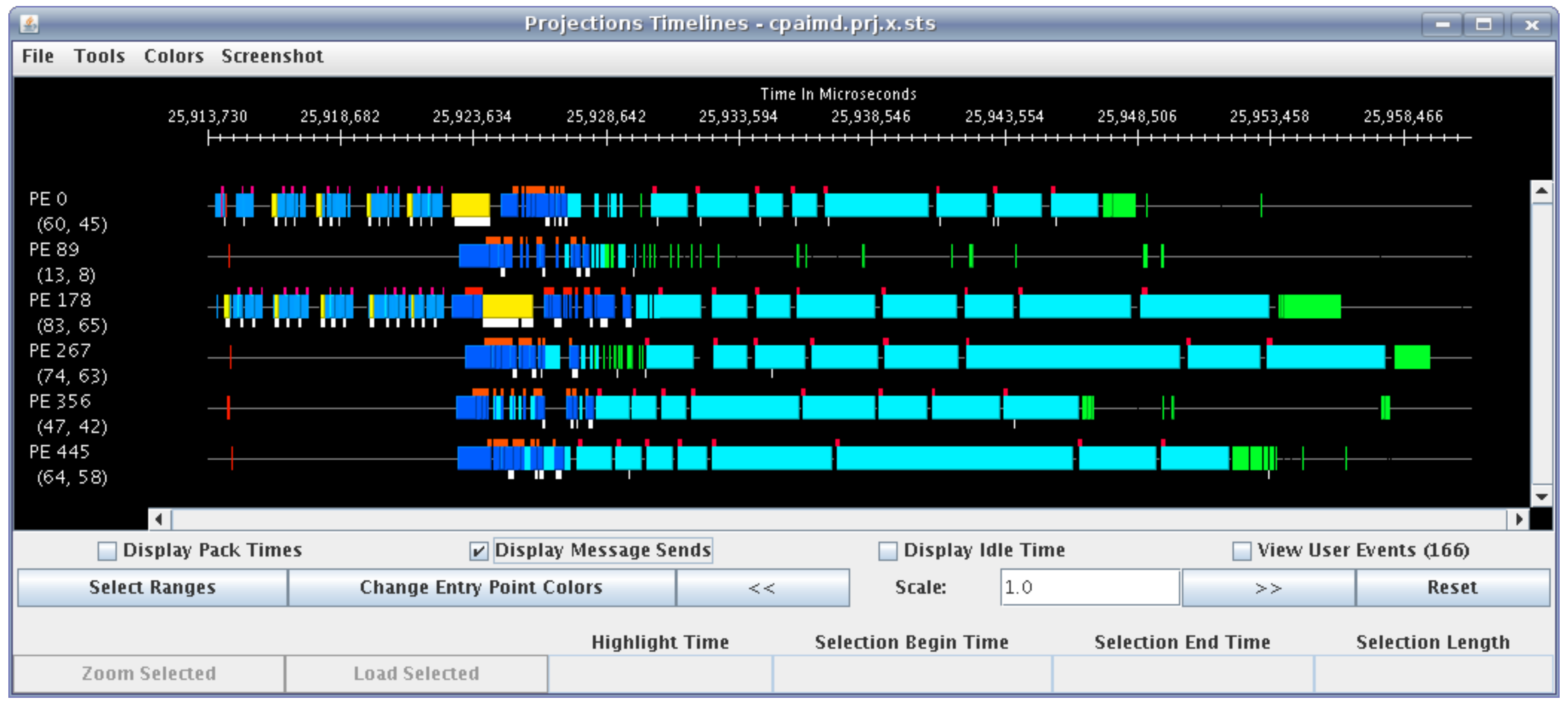


Processor 191: Usage = 20% IDLE = 0% at 16.106s (16106746 us). Drag to pan

Overview
Color intensity utilization over selected time interval

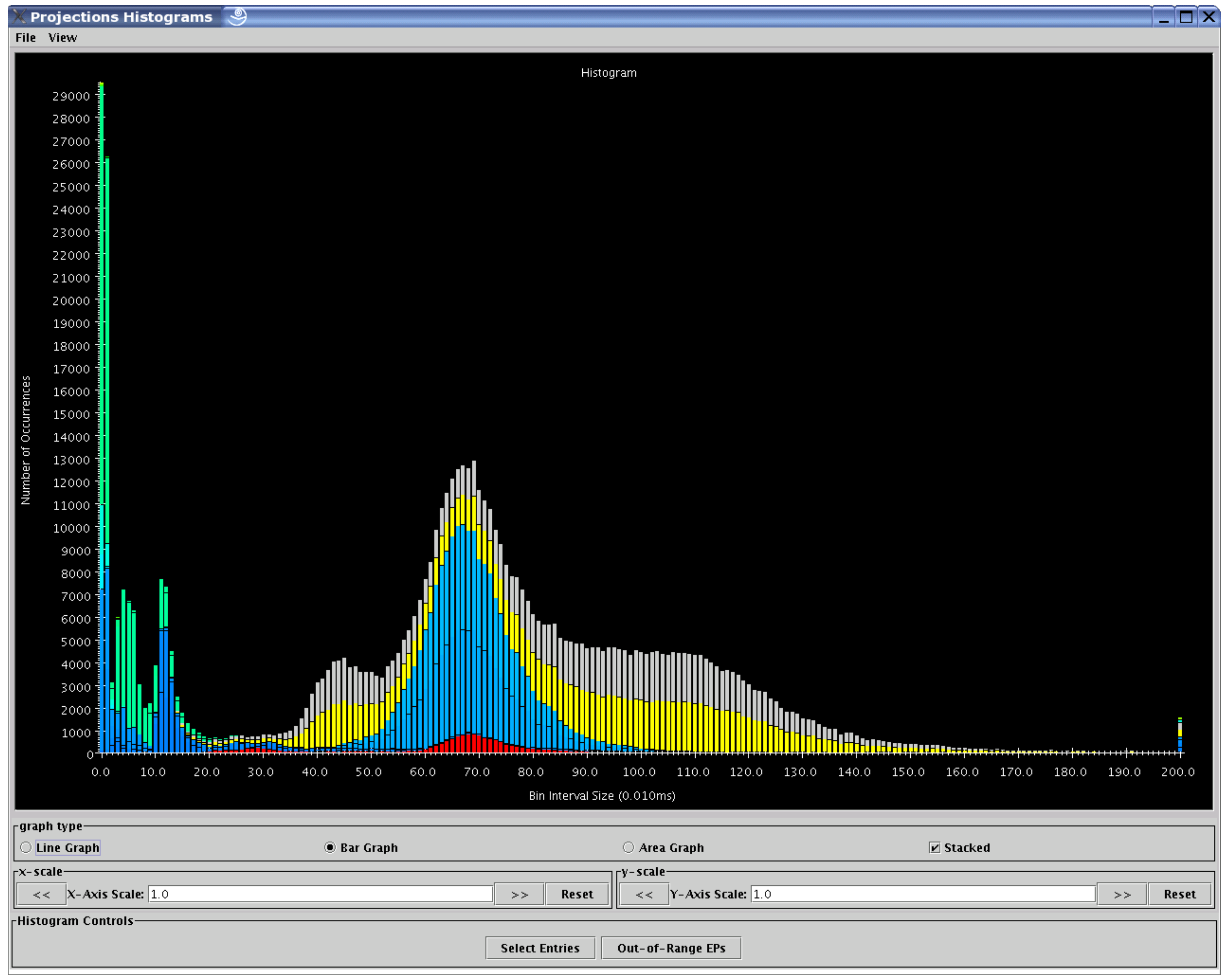
Projections

Visualization



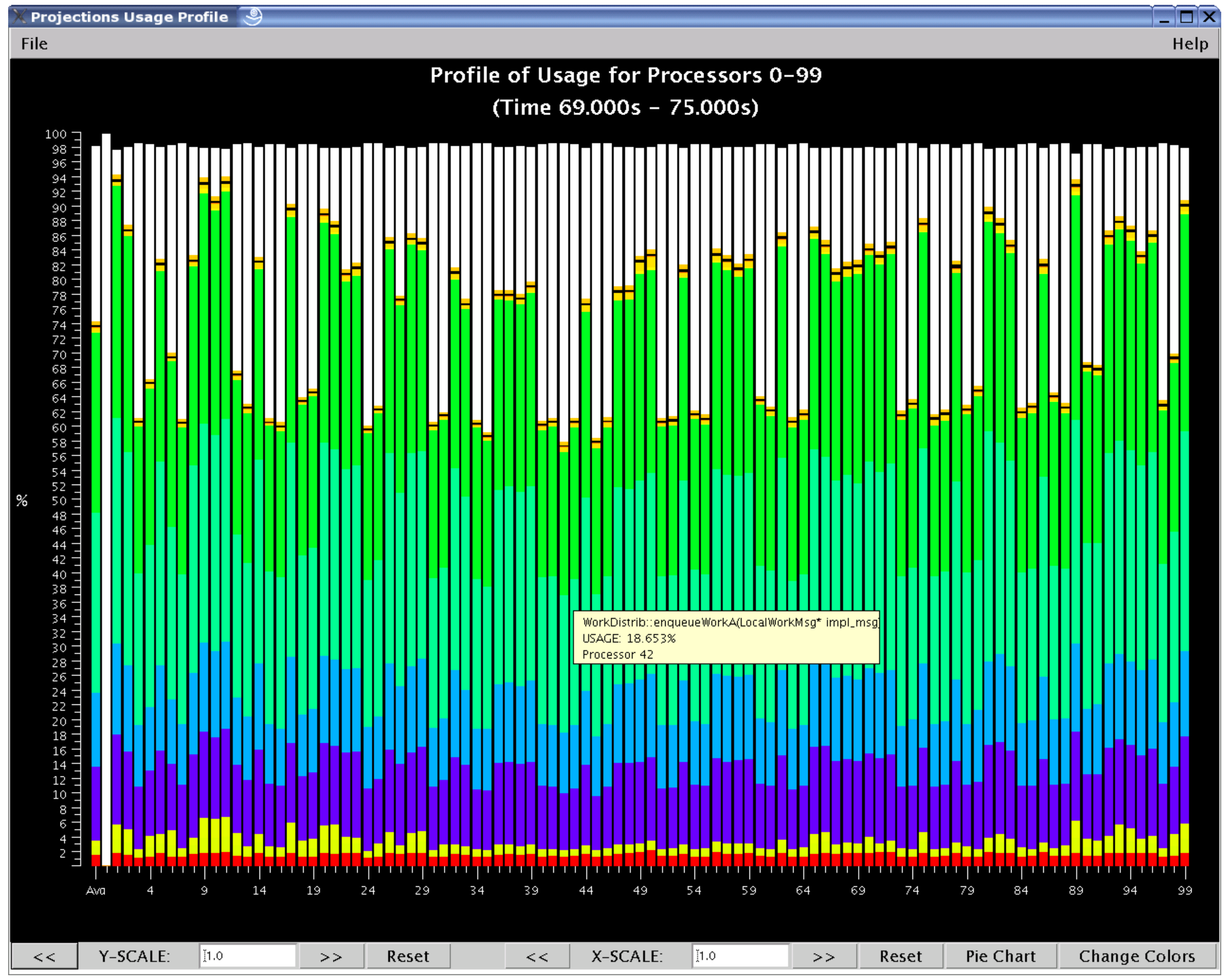
Timeline

A sequence of entry methods for each processor



Histogram

frequency of entry methods or messages



Usage Profile View

Stacked column bar showing time spent in different activities for selected processors

NAMD

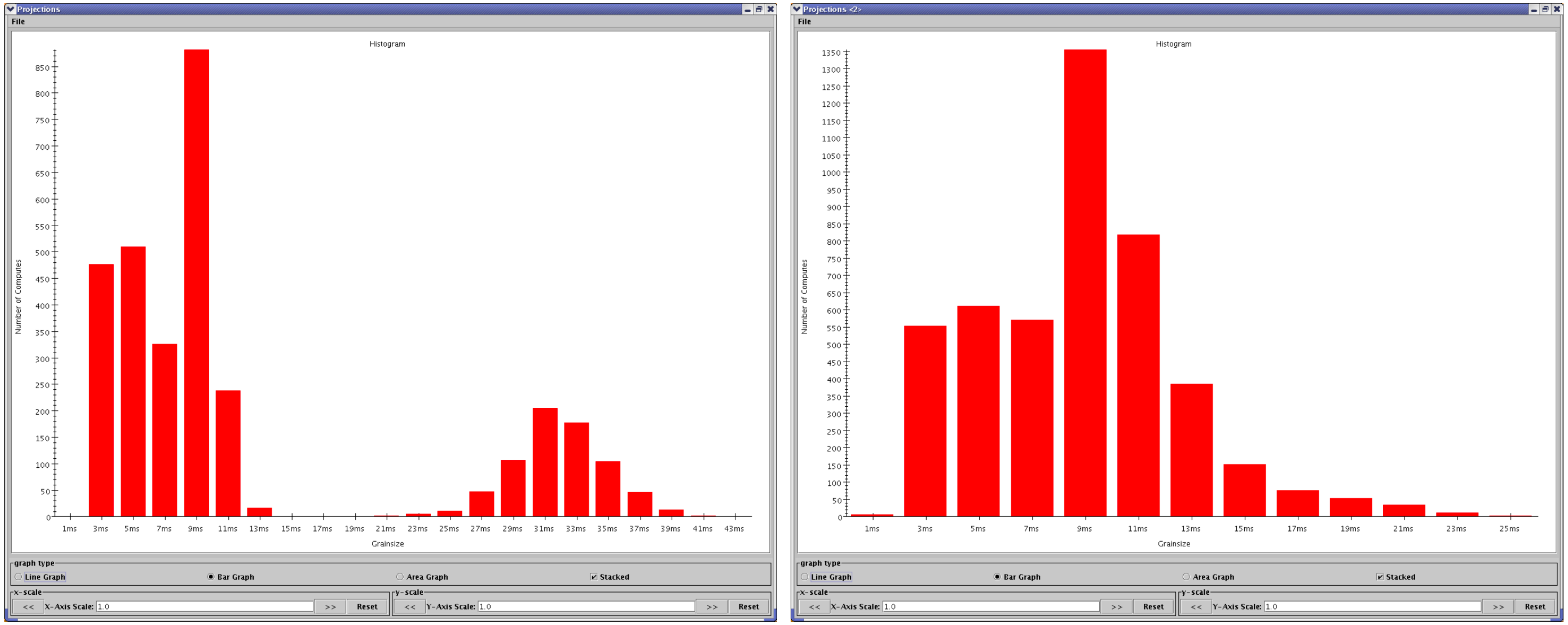
Nanoscale Molecular Dynamics

- Simulates large biomolecular systems
- Each time step:
  - Compute forces on each atom
  - Integrate forces to update atom positions (cutoff radius)
- Atoms are partitioned into cubes (dimension slightly larger than cutoff radius) called home patches
- Create a force computation object for each pair of neighboring cubes
- Each processor receives a number of neighboring cubes/patches, compute objects are distributed to a processor owning at least one home patch

NAMD

Optimization 1 - Grainsize Analysis

- Benchmark: 92,000 atoms, 57 seconds on one processor
- Cannot scale beyond 1,000 processors
- Analysis using projection revealed:
  - Most computation time was spent in force-computation objects, not uniform, range from 1 - 41 microseconds
  - Ideal should be 28 microseconds on 2,000 processors
  - Culprit: electrostatic force computations between cubes that have a common face
  - Solution: split these objects into multiple pieces



graph type: Line Graph [✓ Bar Graph] Area Graph [✓ Stacked]

x-scale: X-Axis Scale: 1.0

y-scale: Y-Axis Scale: 1.0

graph type: Line Graph [✓ Bar Graph] Area Graph [✓ Stacked]

x-scale: X-Axis Scale: 1.0

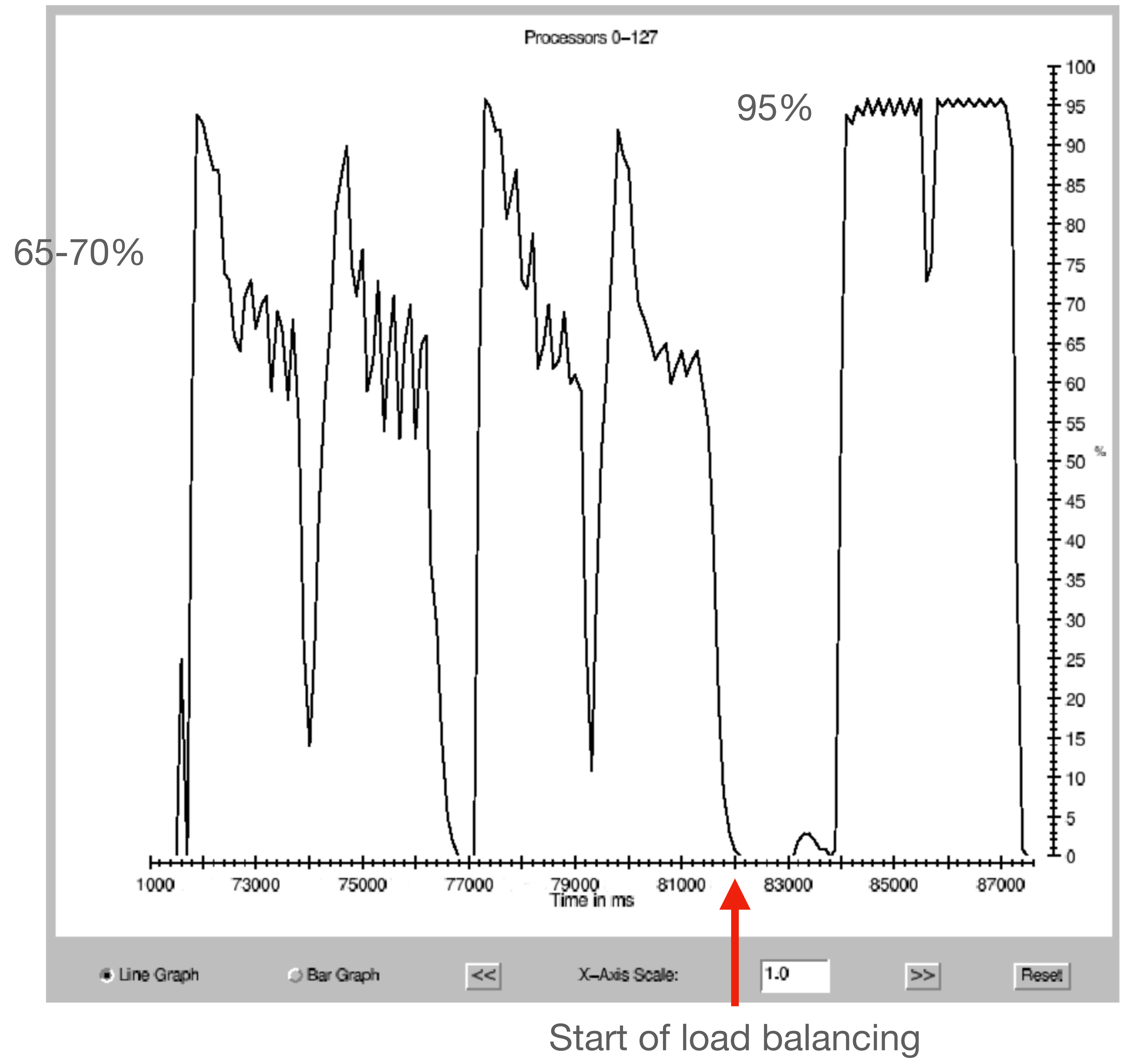
y-scale: Y-Axis Scale: 1.0

Fig. 2. Grainsize Distribution on ASCI Red

NAMD

Optimization 2 - Load Balancing

- Distribution of atoms over space is relatively non-uniform
- Was using Charm++’s measurement-based load balancing framework that supports runtime load and communication tracing
- Can admit different strategies as plugins during a single run
- Greedy strategy



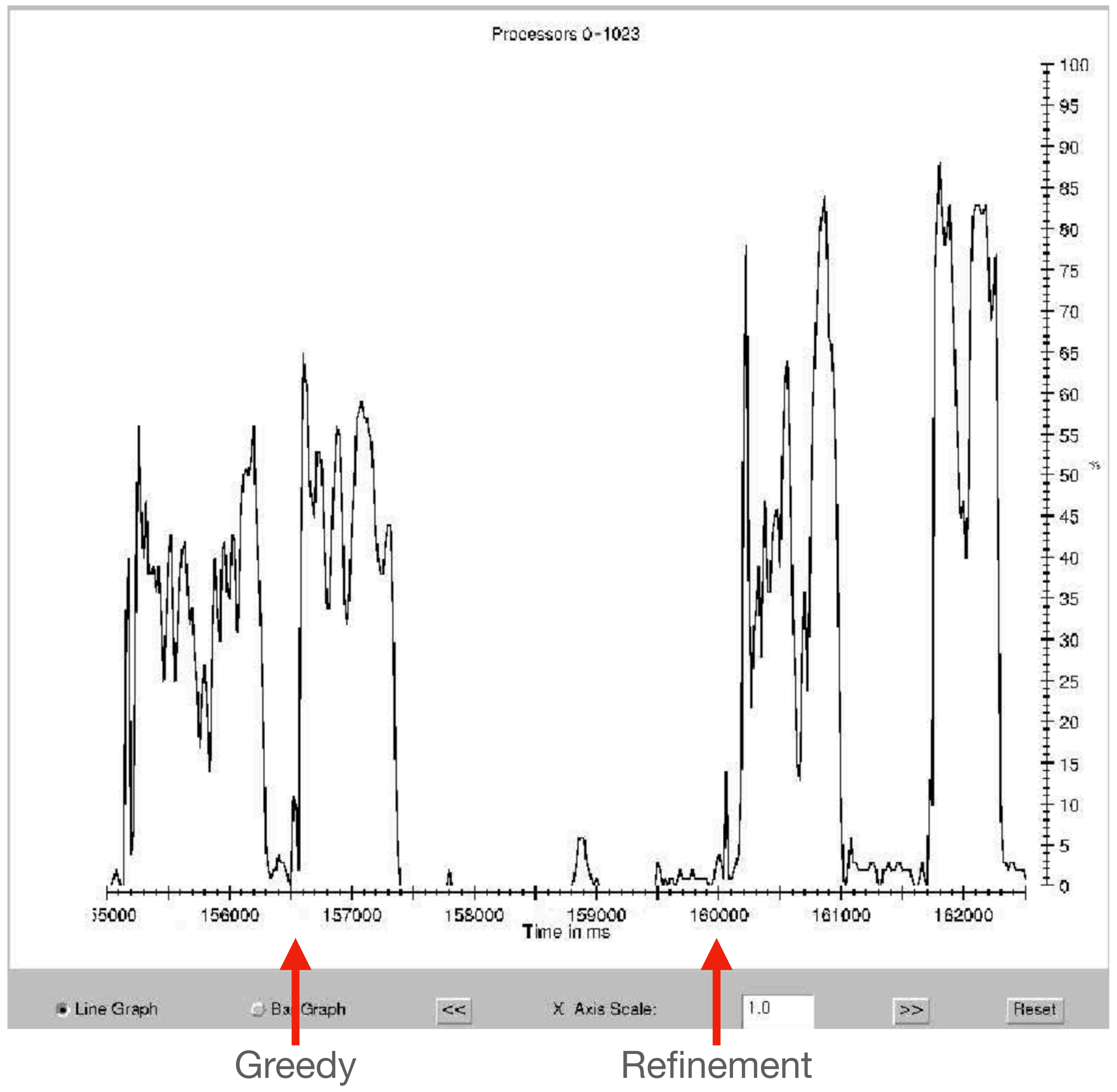


Fig. 3. Processor Utilization against Time on (a) 128 (b) 1024 processors

NAMD

Optimization 2 - Load Balancing

- Result on 1024 processors were not satisfying, the load on processors was different that what the load balancer had predicted
- The greedy strategy ignored existing placement of objects entirely to achieve close to optimal mapping, background load and cache performance were very different after massive object migration
- Solution: add another load balancing phase immediately after the greedy reallocation, which used a simpler “refinement” strategy: only move processors significantly above avg load (5%) so not to disturb the performance context

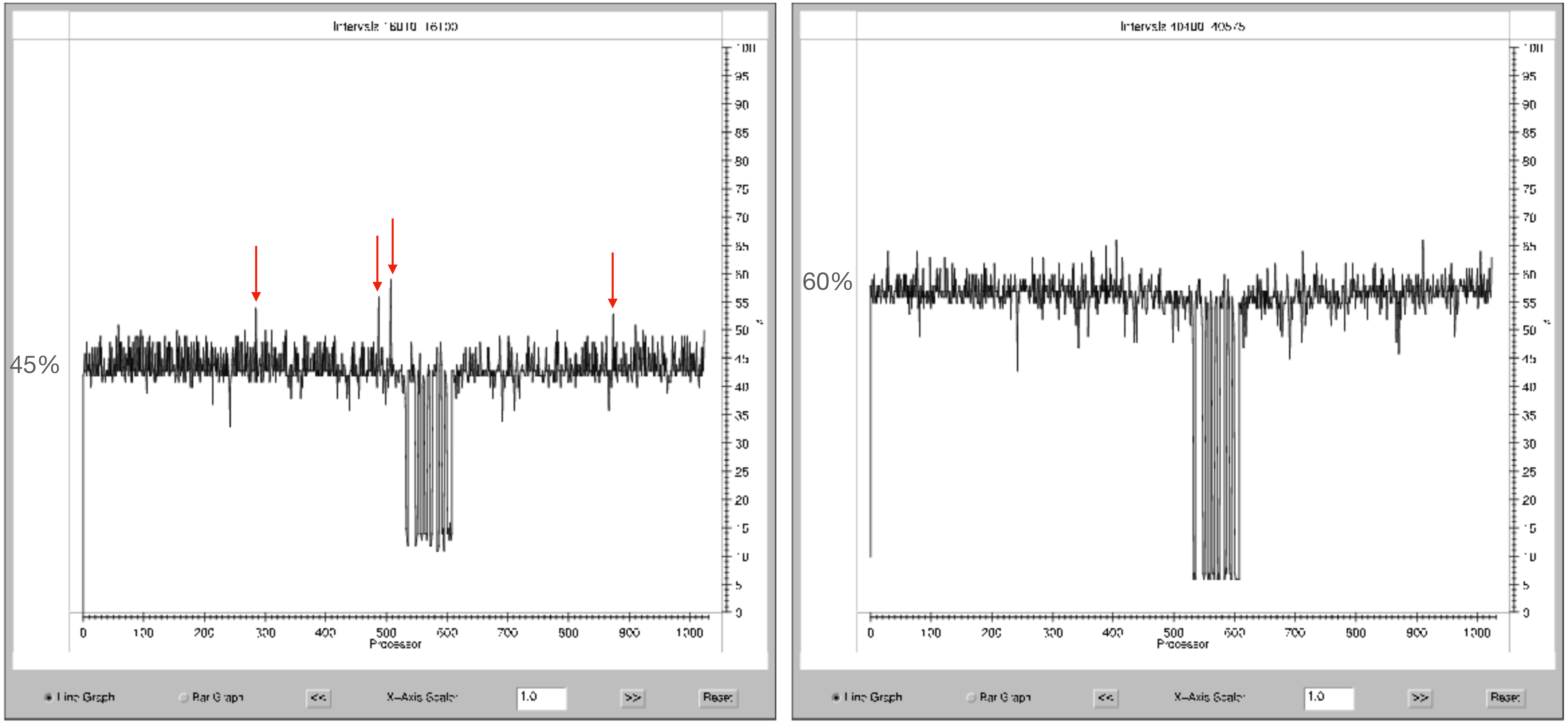


Fig. 4. Processor Utilization after (a) greedy load balancing and (b) refining

NAMD

Optimization 2 - Load Balancing

- Due to quirks in the background load, processor 500 - 600 underloaded
- Does not impact performance much, but overloaded processors do
- The refinement strategy did not change this, but improved overall utilization

NAMD

Optimization 3 - Stretched Entry Method

- Identified the “stretched” entry method problem using timeline view
- Occur on PSC Lemieux while running on large number of processors
- Process 900, 933 methods took 20 - 30 ms (usually 2-3ms or less)
- Entry method blocked on send operation, caused by mistuned library
- OS daemon interference problem

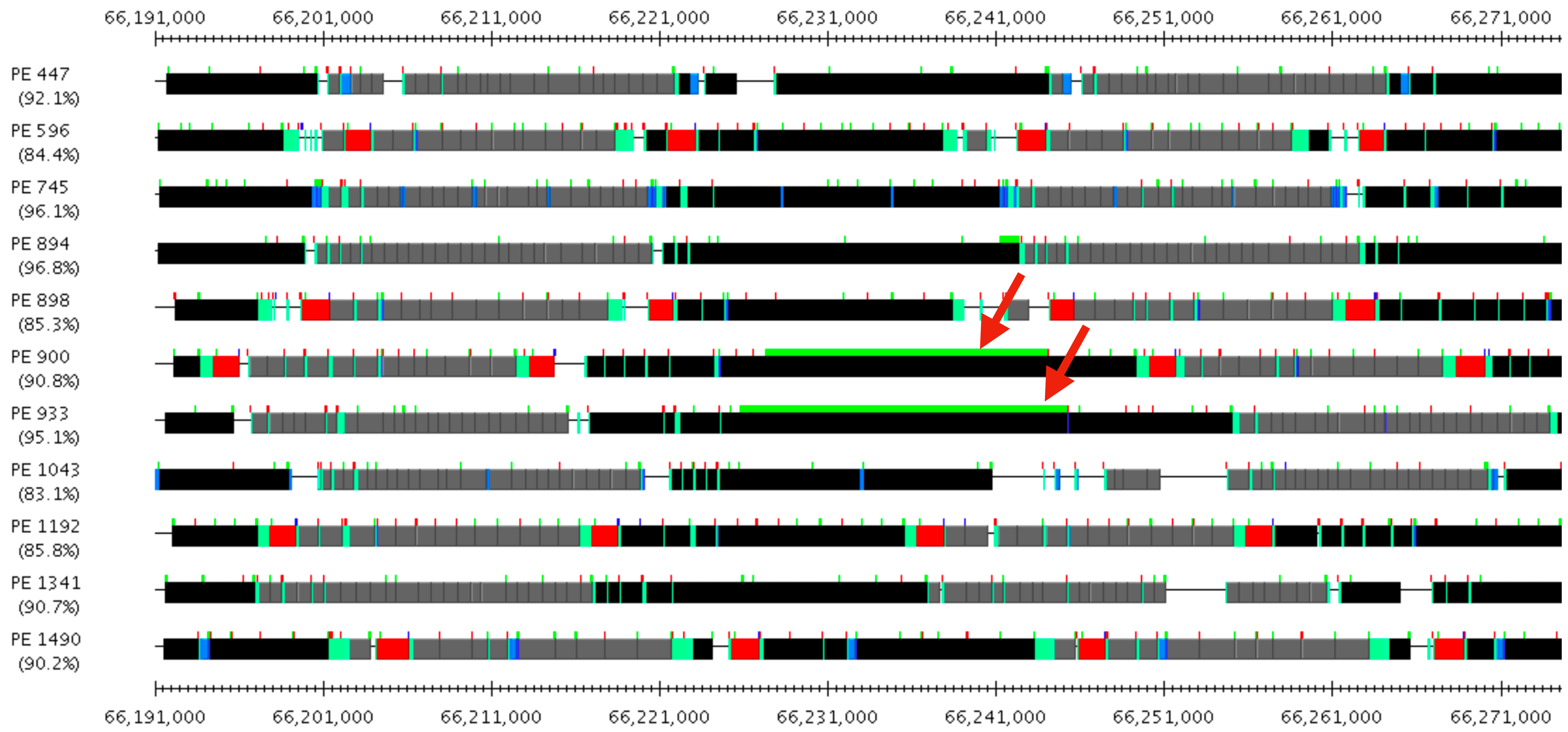
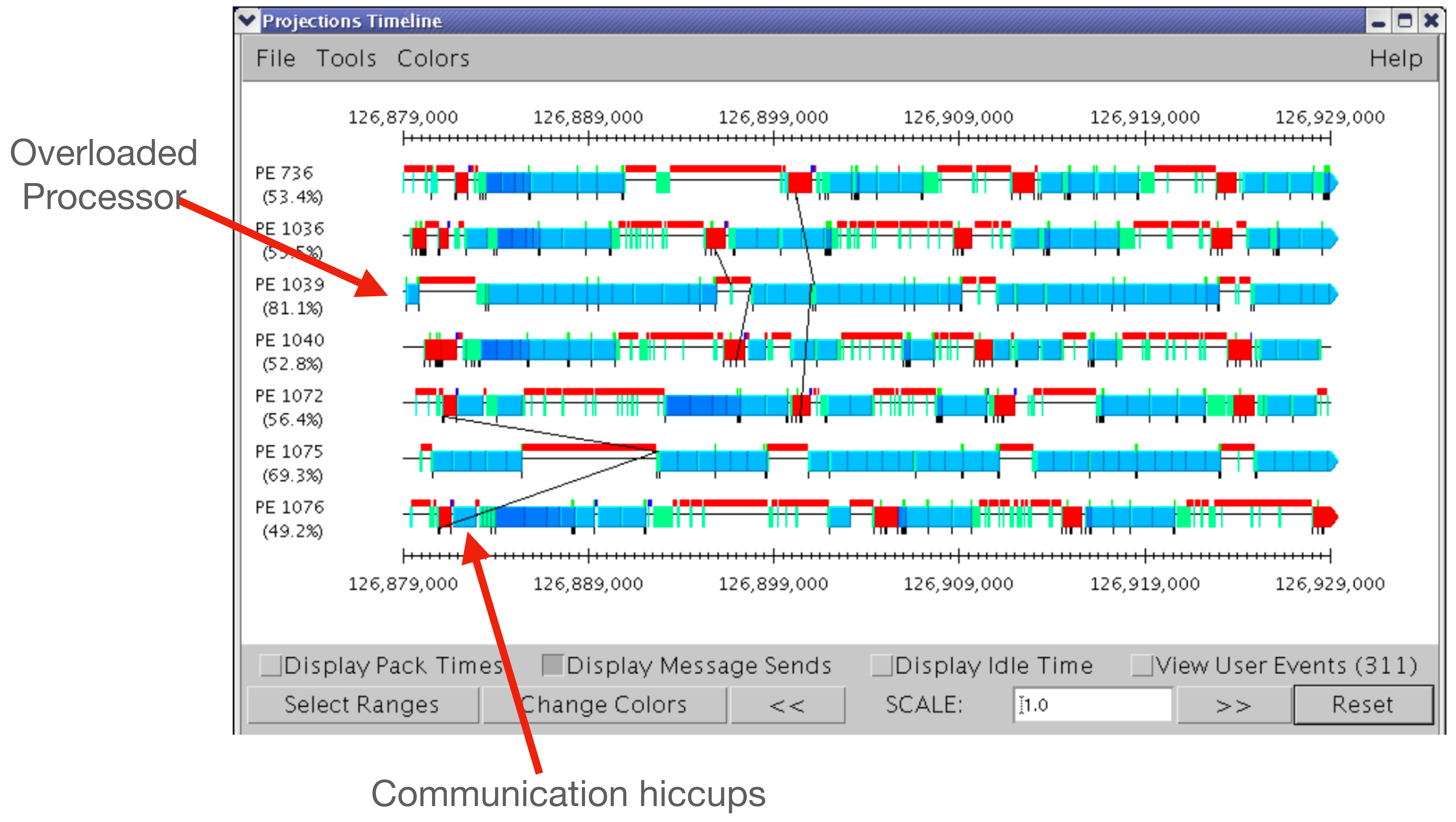
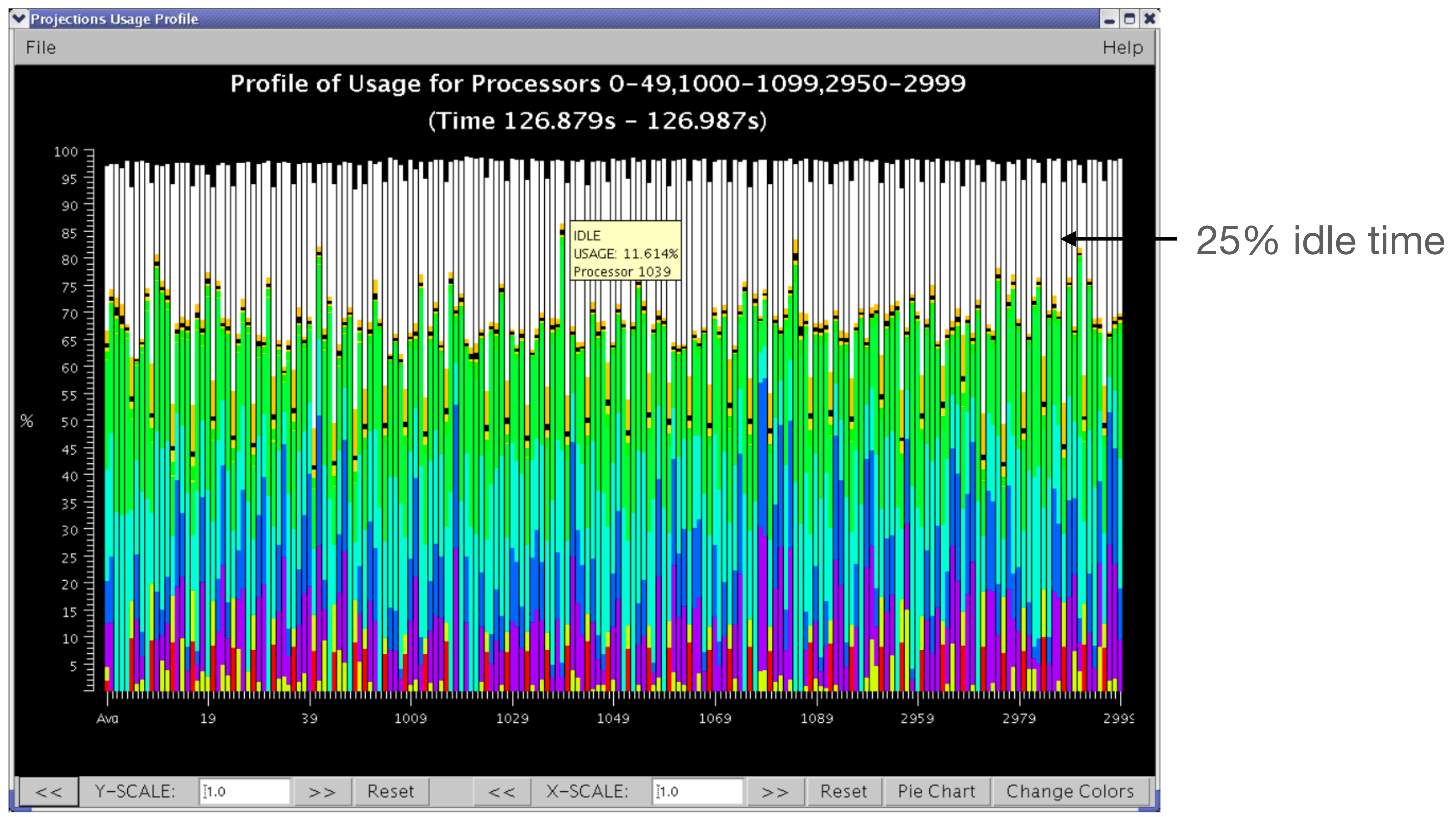


Fig. 5. NAMD Run on 1536 processors



(a) Timeline showing Blocking Receives



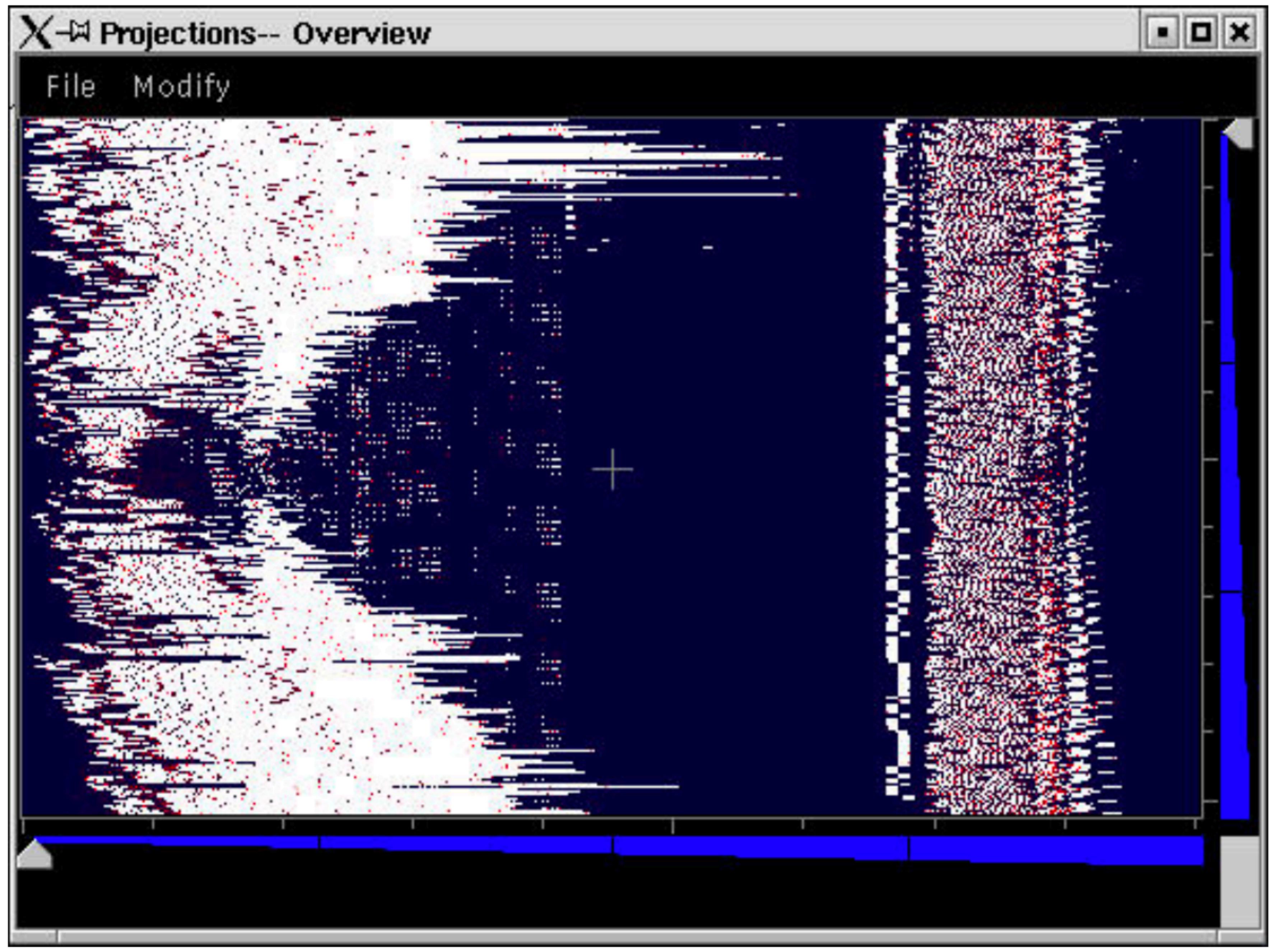
(b) Profile View

Fig. 6. Namd on 3000 processors

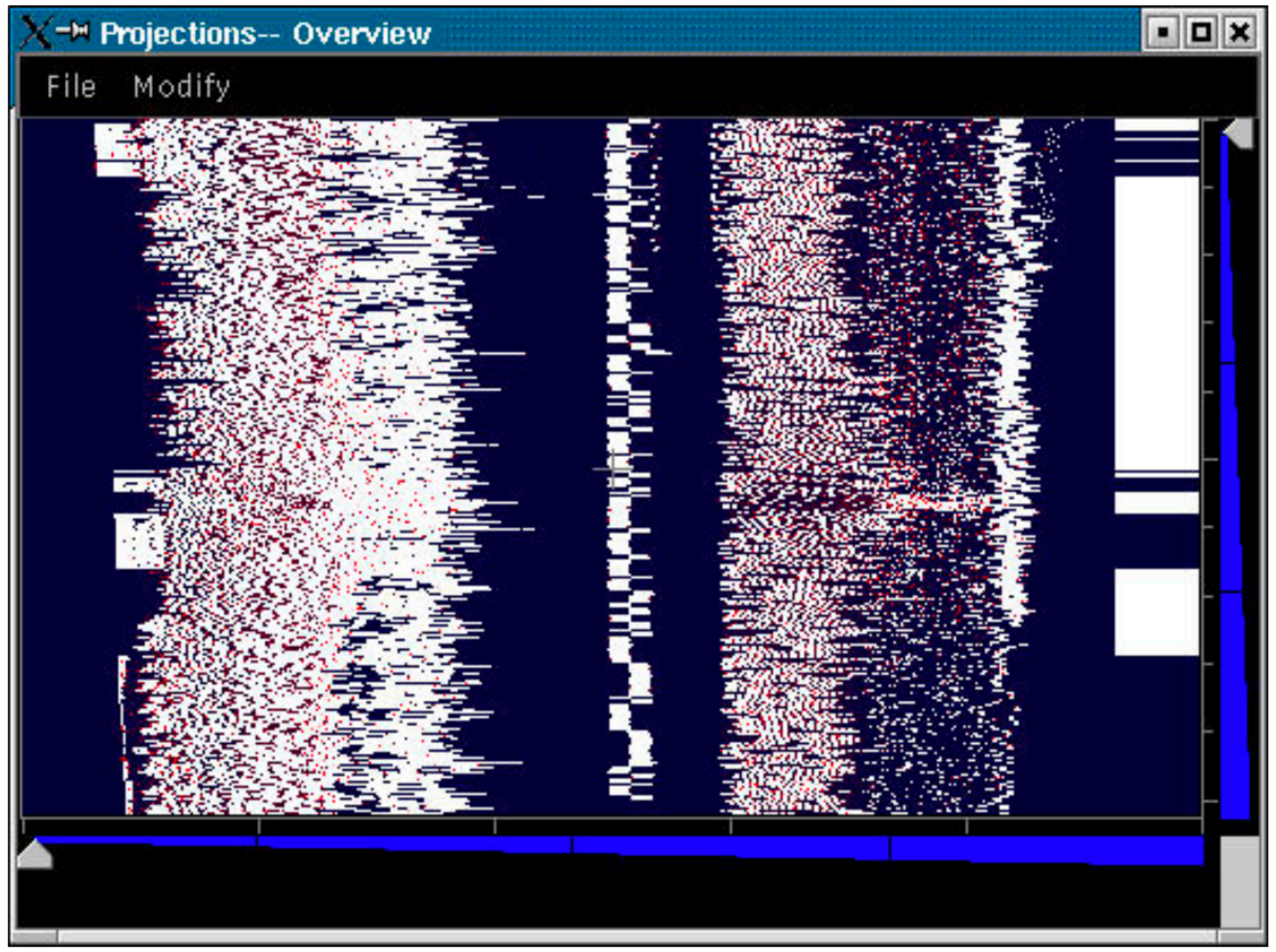
CPAIMD

Car-Parrinello ab initio molecurar dynamics

- Used to study key chemical and biological processes
- Restricted by the number of states, 3-D FFT is communication intensive, cannot scale to thousands of processors
- Objects are electron orbitals/states, each represent Fourier coefficients in 3D g-space
- Each virtual processor being a plane of g-space, which is not very dense, only a fraction of the cube is non-zero
- Initial mapping mapped planes uniformly across processors
- Solution: explicitly consider the load caused by each plane, result in better mapping



(a) Load imbalance in phases I and IX



(b) Final result with load-vectors

Fig. 7. Solving the problem of load imbalance on 1024 processors

Next Gen Supercomputers
2002

- IBM BlueGene/L: 64,000 dual-processor nodes, 360 teraflops peak performance
- IBM BlueGene/C (Cyclop), 1M floating point units fed by 8M instruction streams, 1 petaflops peak performance
- Challenges
  - Write parallel programs that exploits this power
  - Analyze their performance

Next Gen Supercomputers

BigSim

- To evaluate parallel applications and performance analysis tools on supercomputers, authors crated a parallel simulator: BigSim
- Employ the projections framework on BigSim
- Cannot generate 64,000 log files, I/O overheads and memory cost
    - Use summary mode, more compact trace data
    - Global reduction that collects and combines all trace data into one file
    - Generate detailed log, only for a specified range of processors

Next Gen Supercomputers

NAMD on Blue Gene/L

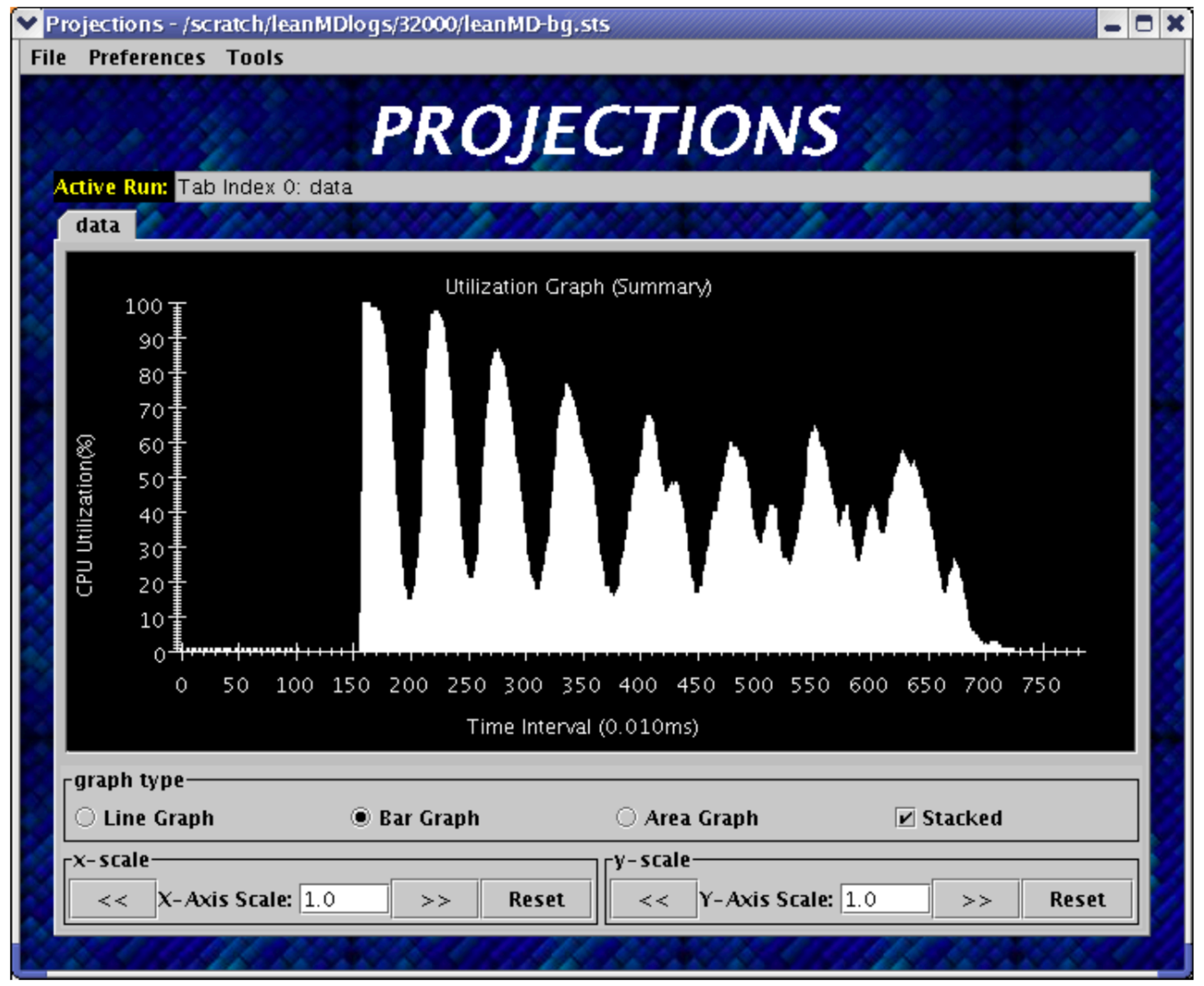
- NAMD shown to scale to 3,000 processors but not beyond
- ER-GRE benchmark, 36,573 atoms, simulates space of 92x92x92 $\mathring{A}^3$ (ångström)
- Using NAMD's one-way decomposition strategy: 8x8x8 number of cells given the cutoff distance of $12\mathring{A}^3$, 7168 cell to cell interactions to calculate
- Not enough work to distribute across 64,000 processors, some would be idle

Next Gen Supercomputers

NAMD on Blue Gene/L

- Three-away decomposition: three cells span the cutoff distance
- Every cell compute interactions with every cell that is three-away
- Produces 13,824 cells, more than 2 million cell-to-cell interactions
- Easily distributed across 64,000 nodes

- LeanMD: models the cutoff computation
- Run on PSC Lemieux simulated BlueGene/L using node size 1K - 64K
- On 32K processors utilization stabilizes at about 50%
- Speedup saturate starting from 16K processors
- Load imbalance: most processors have load of 2ms, some as high as 11ms

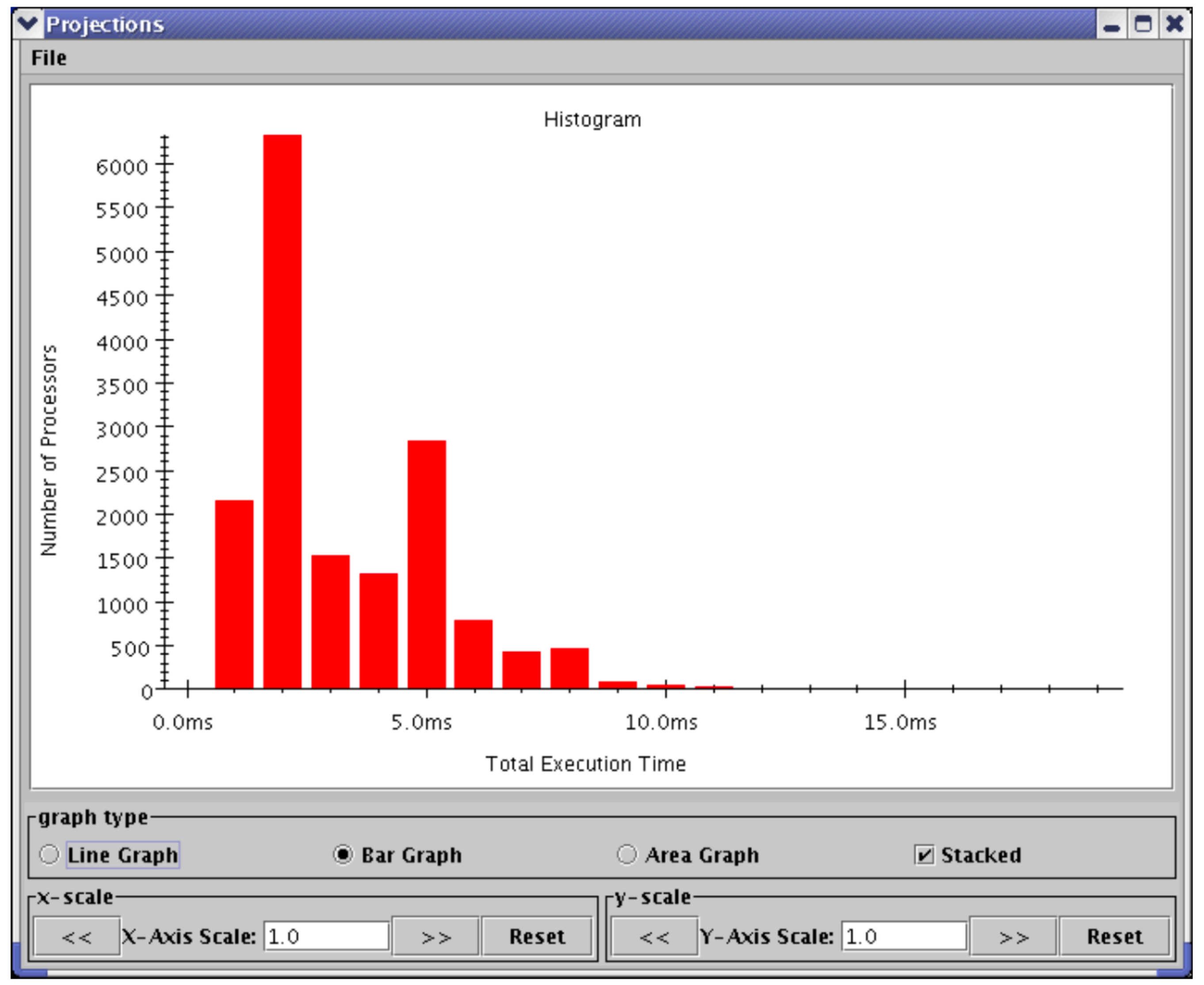


graph type: ( ) Line Graph [✓ Bar Graph] ( ) Area Graph [✓ Stacked]

X-Axis Scale: 1.0

Y-Axis Scale: 1.0

(a) Average utilization per interval for LeanMD on 32,000 processors



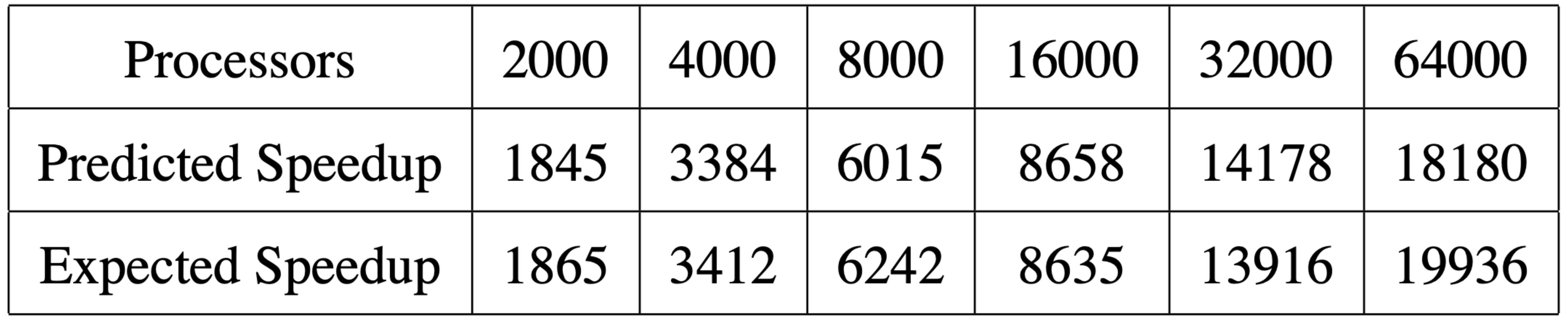
graph type: ( ) Line Graph [✓ Bar Graph] ( ) Area Graph [✓ Stacked]

X-Axis Scale: 1.0

Y-Axis Scale: 1.0

(b) Distribution of processors based on load in ms

Fig. 8. LeanMD Projections Views



Processors	2000	4000	8000	16000	32000	64000
Predicted Speedup	1845	3384	6015	8658	14178	18180
Expected Speedup	1865	3412	6242	8635	13916	19936

Table 1
Predicted vs. expected speedup, normalized based on 1000

- Calculate expected performance based on load imbalance alone
- Very close to the authors's predicted performance, indicating that load imbalance is the major performance issue
- This type of analysis is possible thanks to the rich trace data produced by the simulator