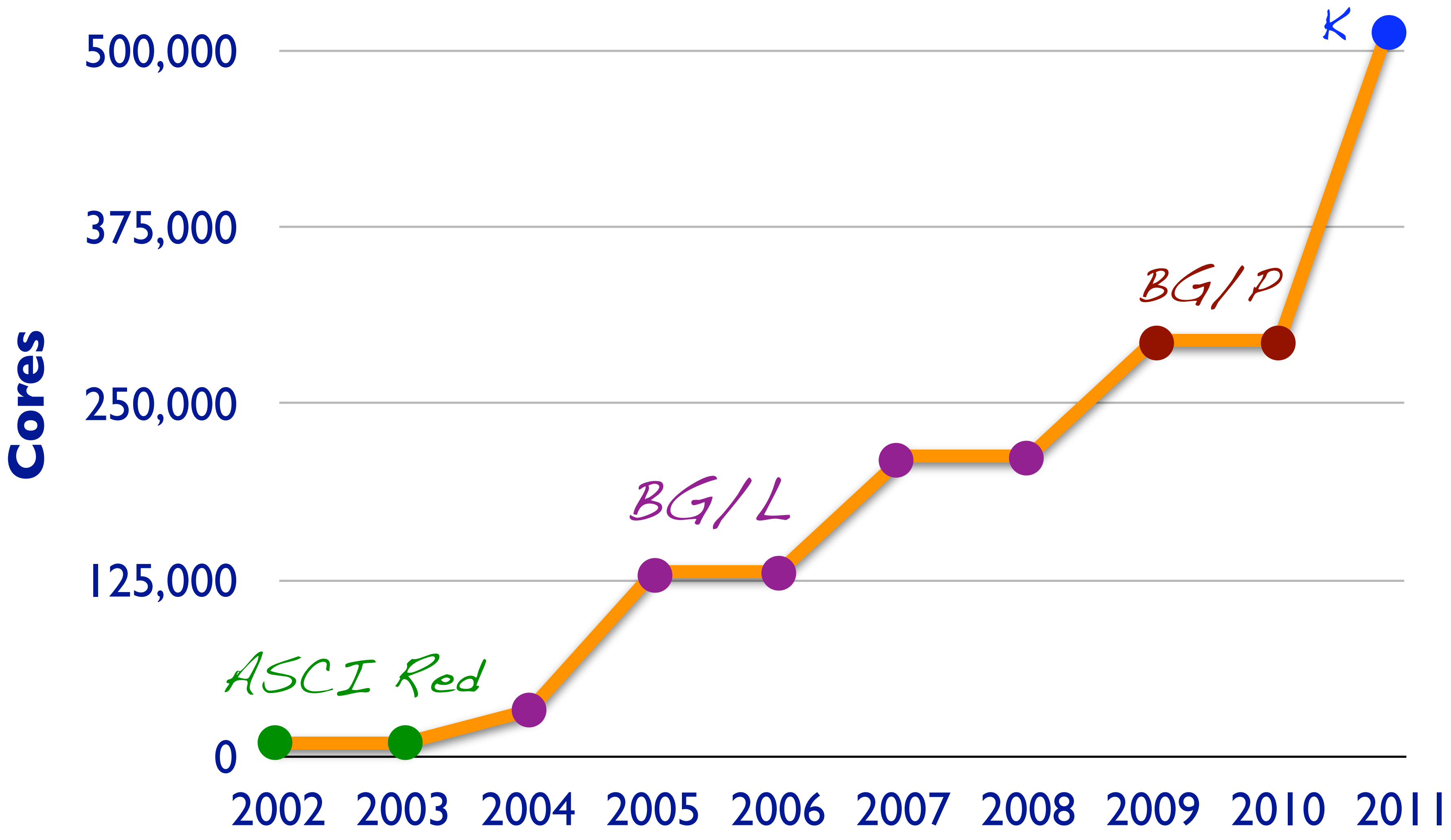
Dynamic Load Balance for Optimized Message Logging in Fault Tolerant HPC Applications

Esteban Meneses, Greg Bronevetsky and Laxmikant V. Kalé



Frequent Failures

Sequoia	Exascale
1.6 million cores	100 million cores?
2012	2018

A key challenge is to adapt to the unavoidable variability in time and space (processes/processors) of future applications and systems
International Exascale Software Project

→

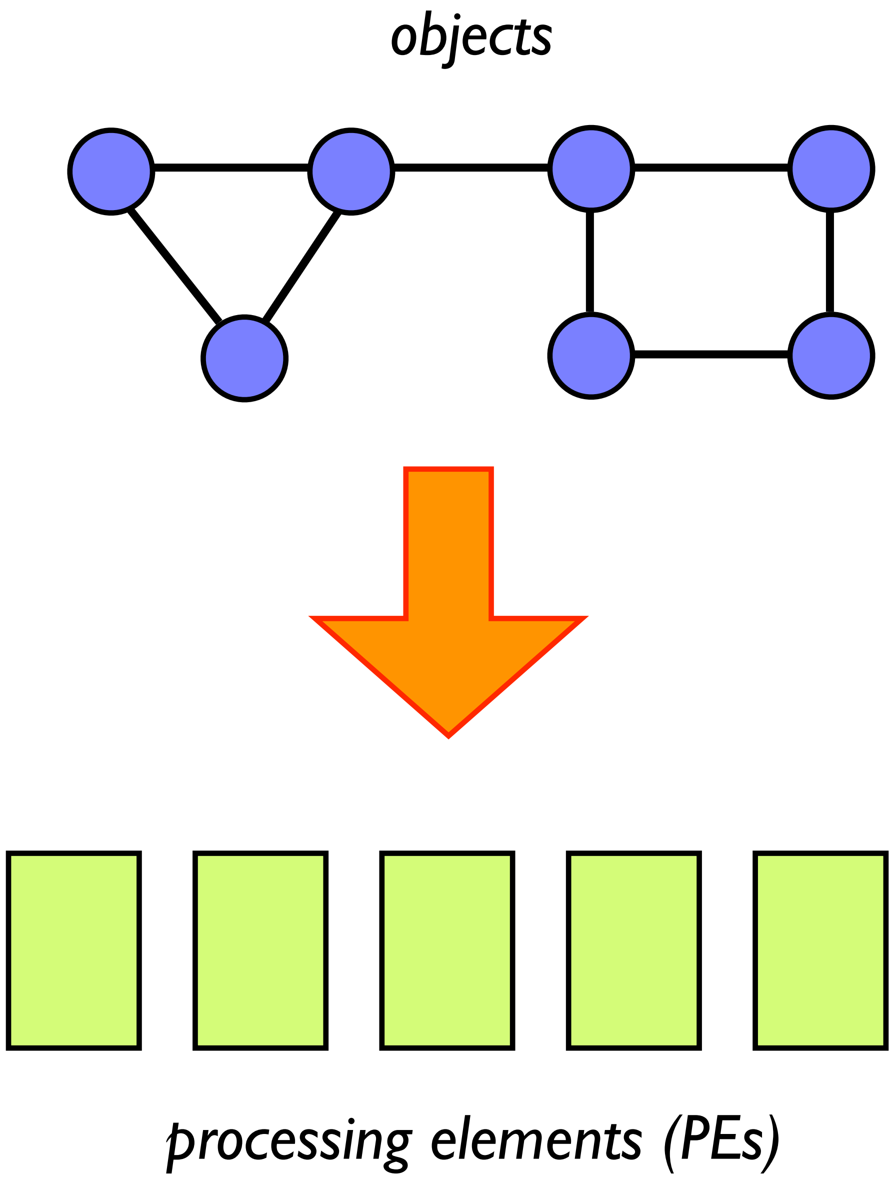
Leverage Load Balancing Infrastructure for Fault Tolerance

Contents

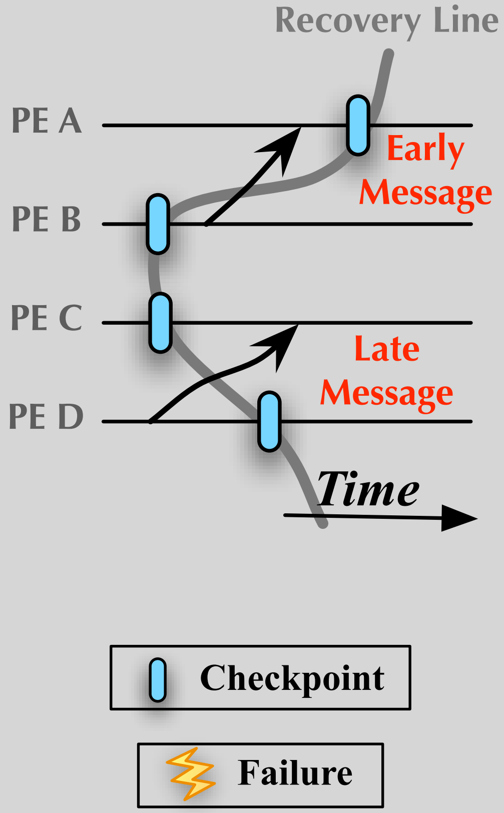
- Rollback-Recovery
- Load Balancing
- Optimized Message Logging
- Experiments
- Conclusions and Future Work

Model

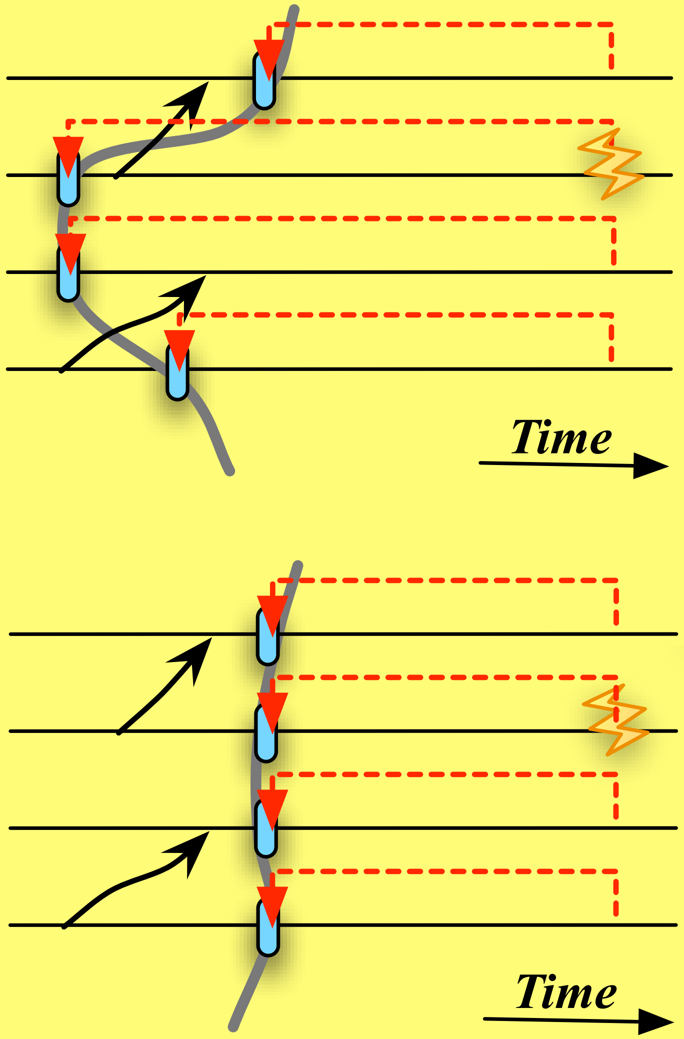
- Object-based over-decomposition.
- Asynchronous method invocation.
- Migratable-objects runtime system.
- Non-FIFO channels.
- Charm++ and Adaptive MPI.
- Fail-stop crashes, user-level checkpoint.



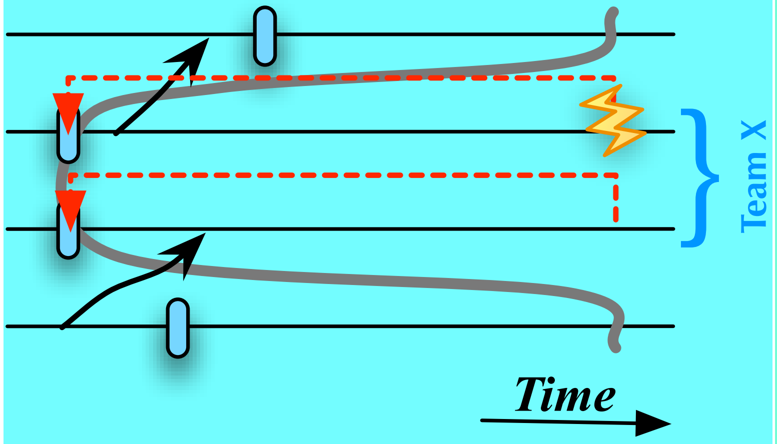
Rollback-Recovery



Checkpoint/Restart

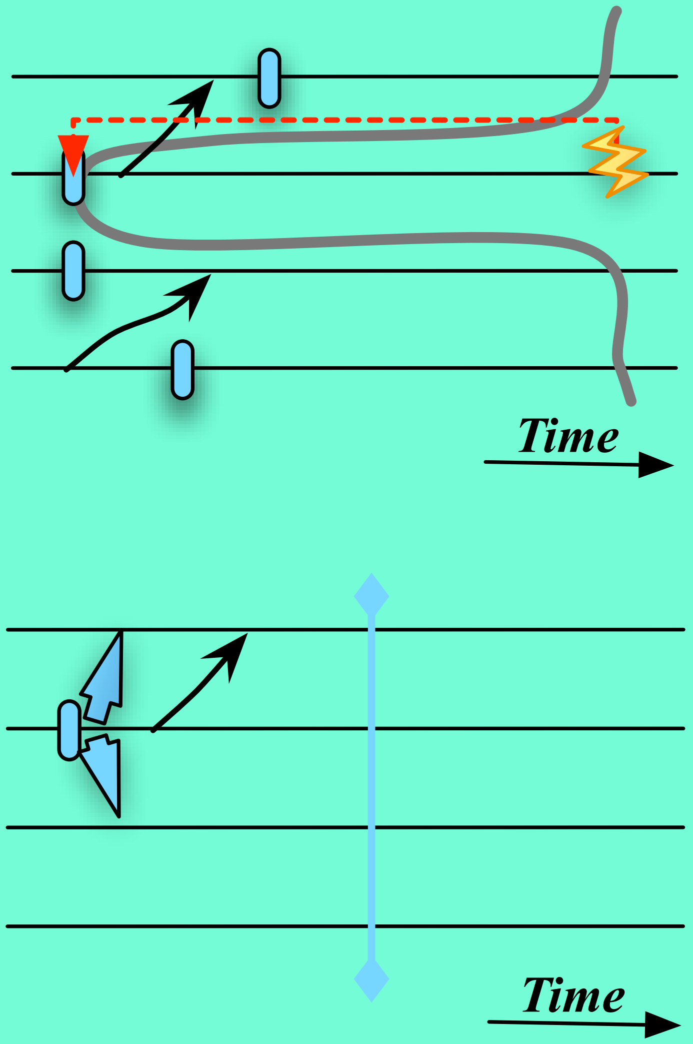


Team-based Message Logging



team = group of PEs

Message Logging



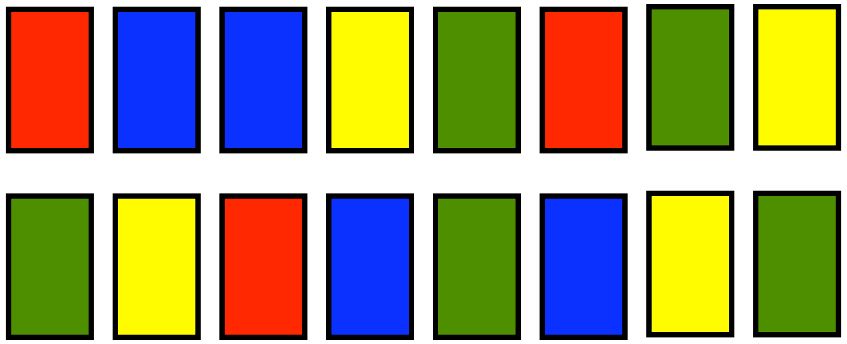
Expensive Recovery

How do you form teams?

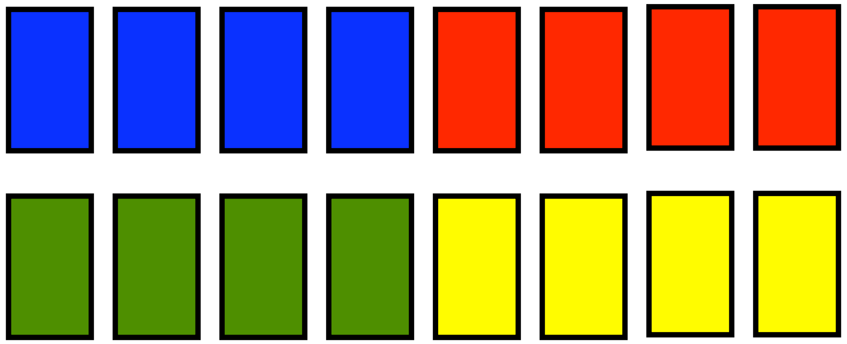
Memory Overhead

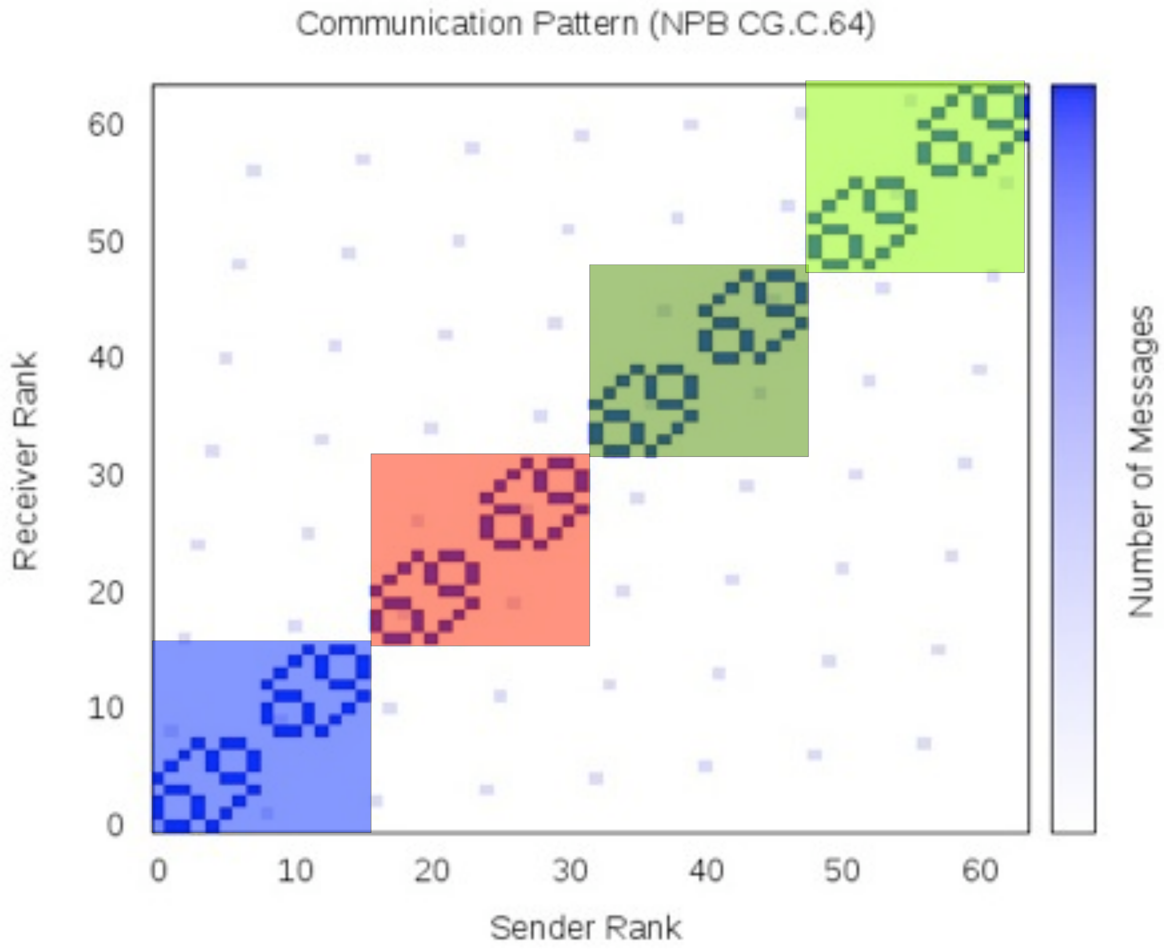
Forming Teams

Random

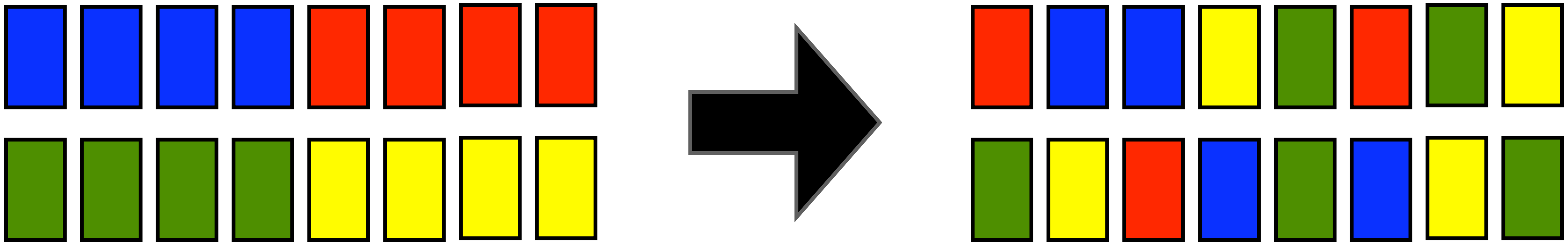


Static

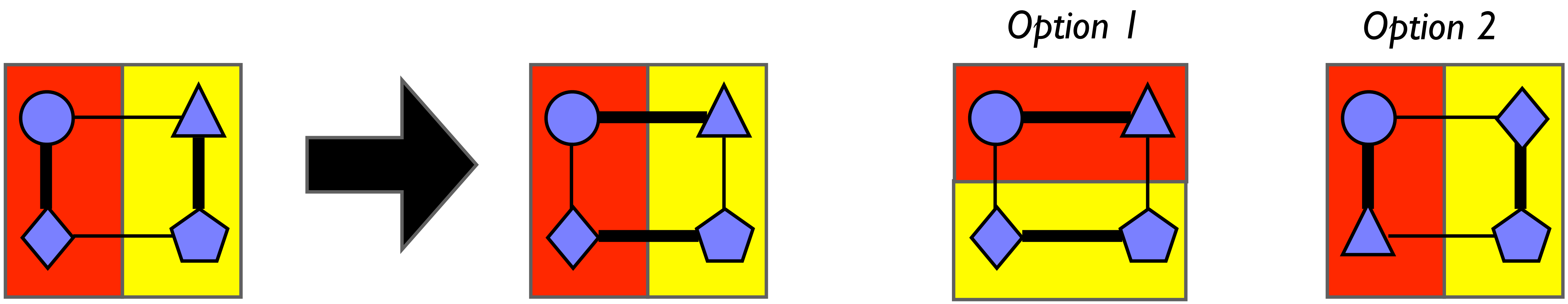




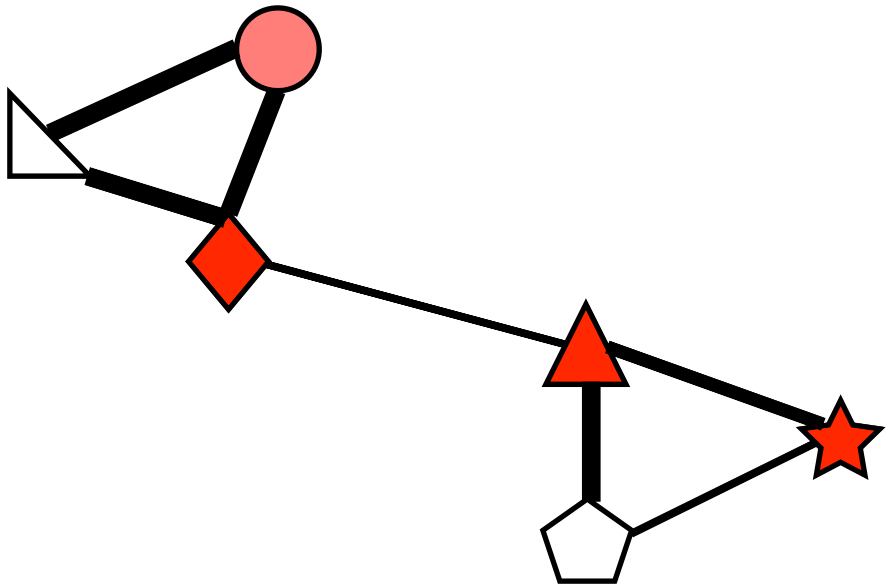
Dynamic



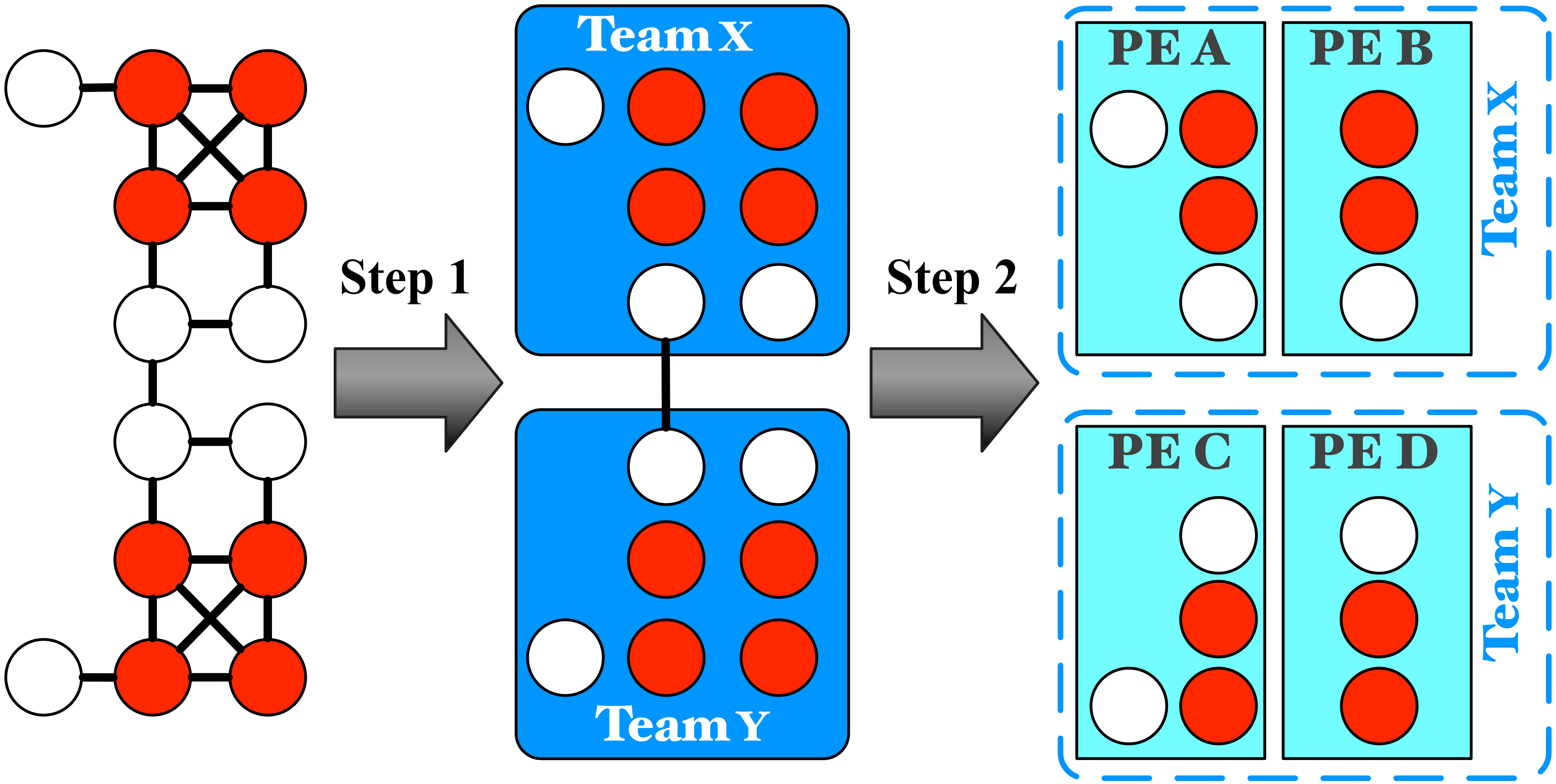
Load Balancing



- Runtime system collects statistics about objects:
  - Computational load.
  - Communication graph.

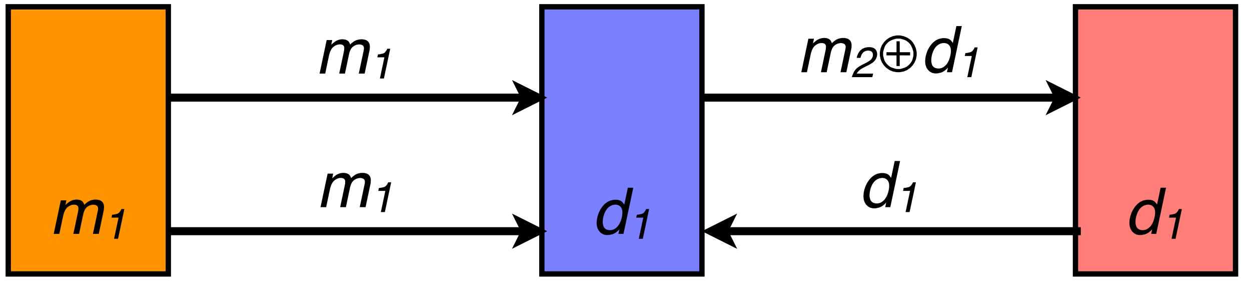


Load Balancer



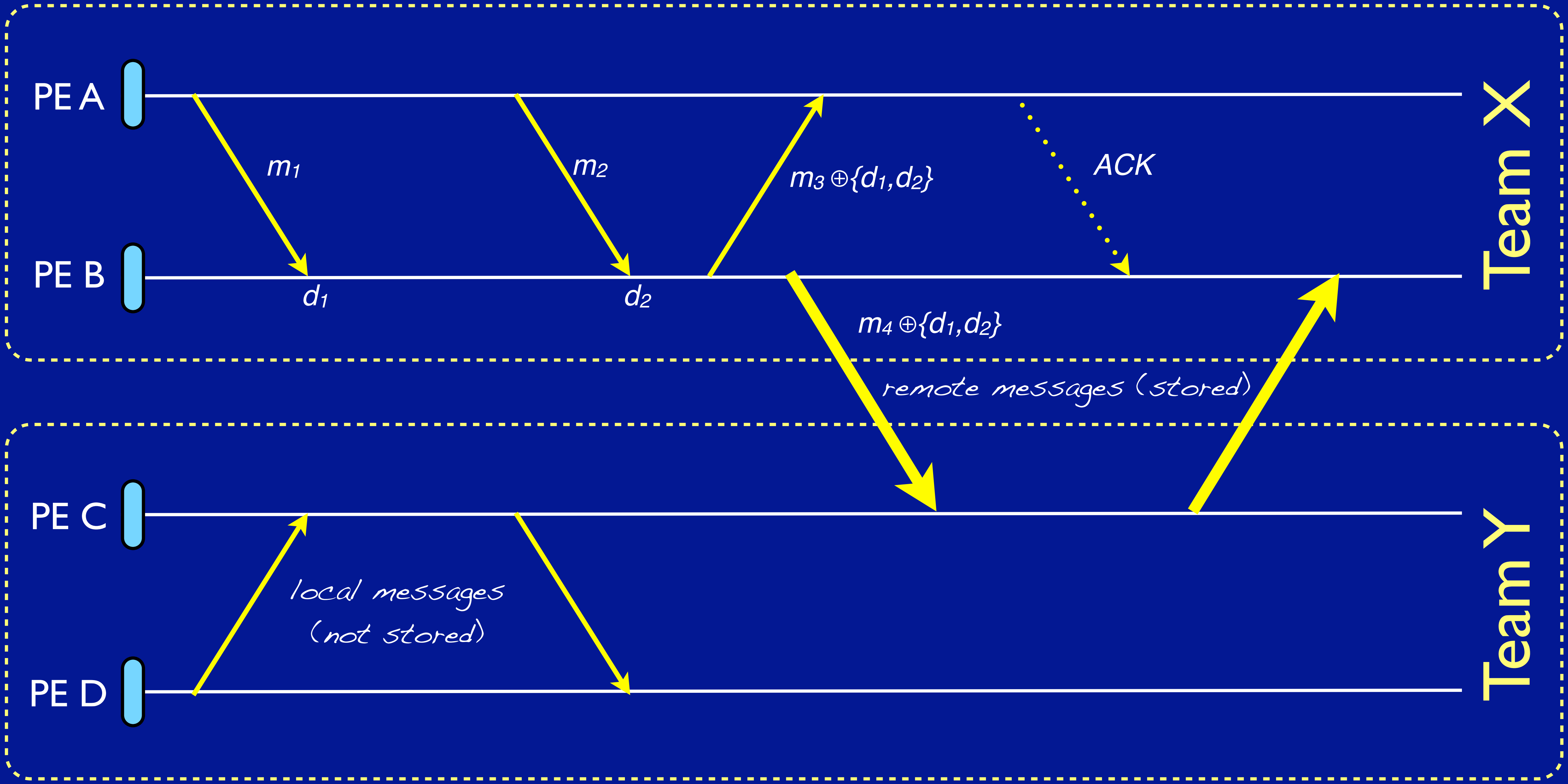
Optimized Message Logging

- Every non-deterministic event generates a determinant (message reception).
- Simple Causal Message Logging:
  - Messages stored at sender's memory.
  - Determinants stored at receivers (at least once).



- Optimization: synchronized checkpoint.

Protocol


PE A
PE B
PE C
PE D
m1
m2
m3 ⊕{d1,d2}
ACK
d1
d2
m4 ⊕{d1,d2}
remote messages (stored)
local messages
(not stored)
Team X
Team Y

Experiments

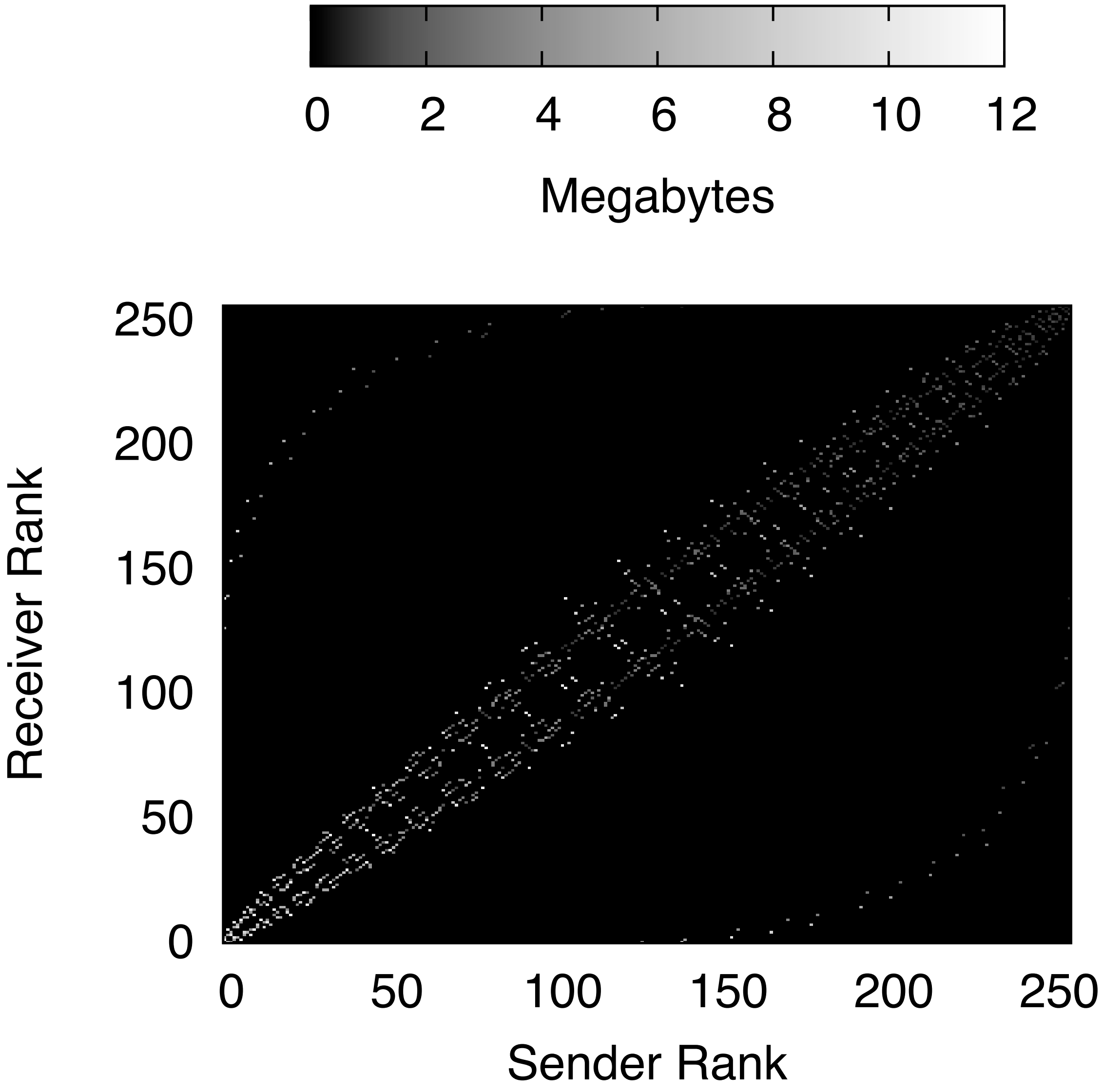
- Overhead of load balancer with team formation.
- Smart team formation benefits.
- Team size vs memory overhead.

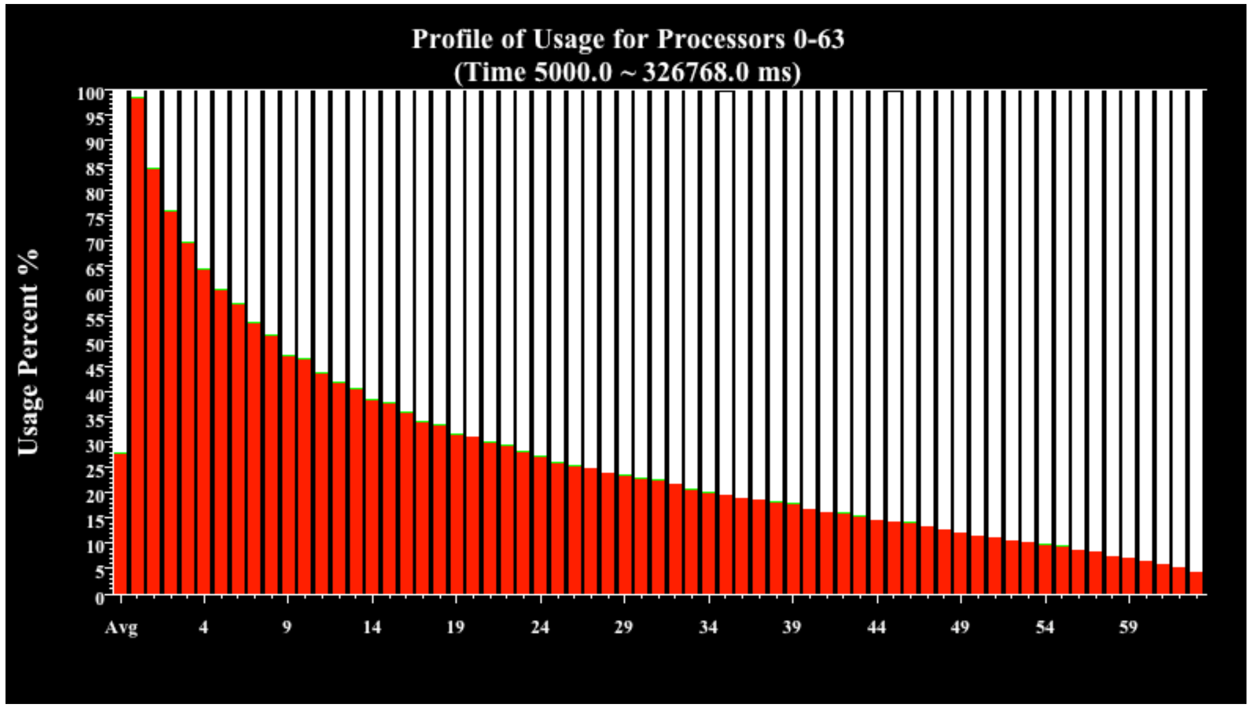
Application	Language	Load Imbalance
NPB-BT-MZ	MPI	Static
Mol3D	Charm++	Dynamic

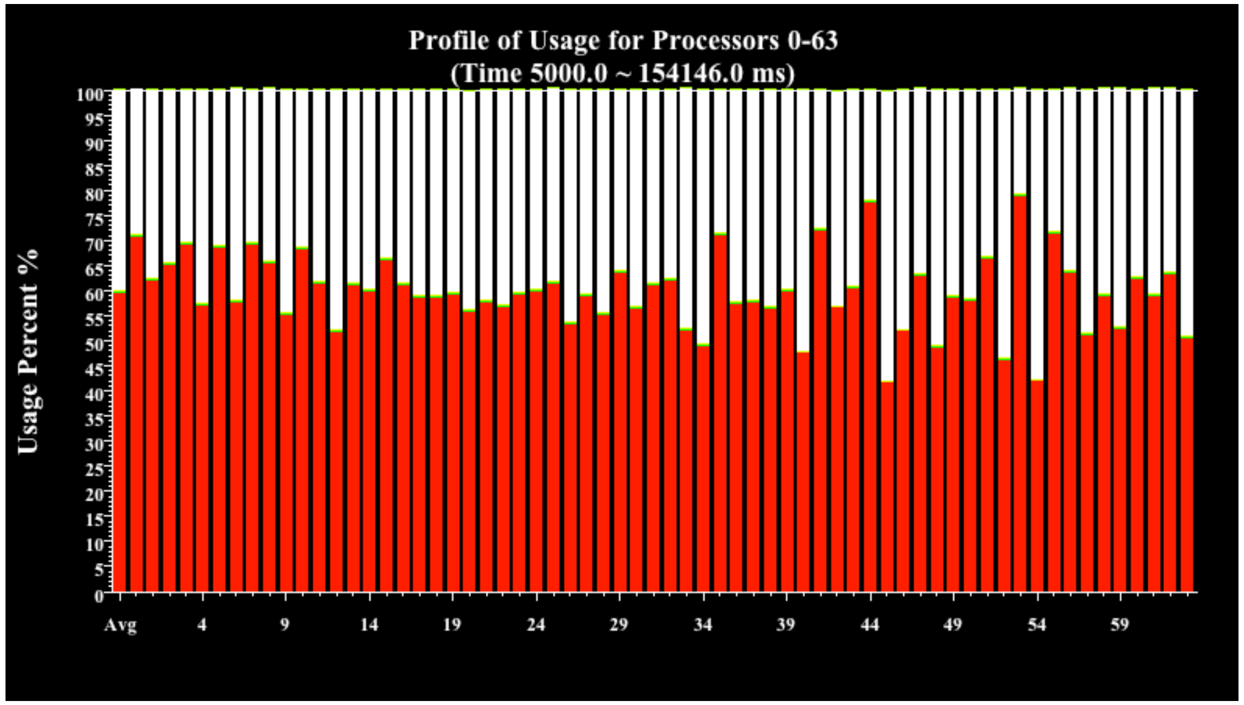
Testbed

- Steele@RCAC
- 893 8-core nodes
- 16 GB RAM per node
- Gigabit Ethernet

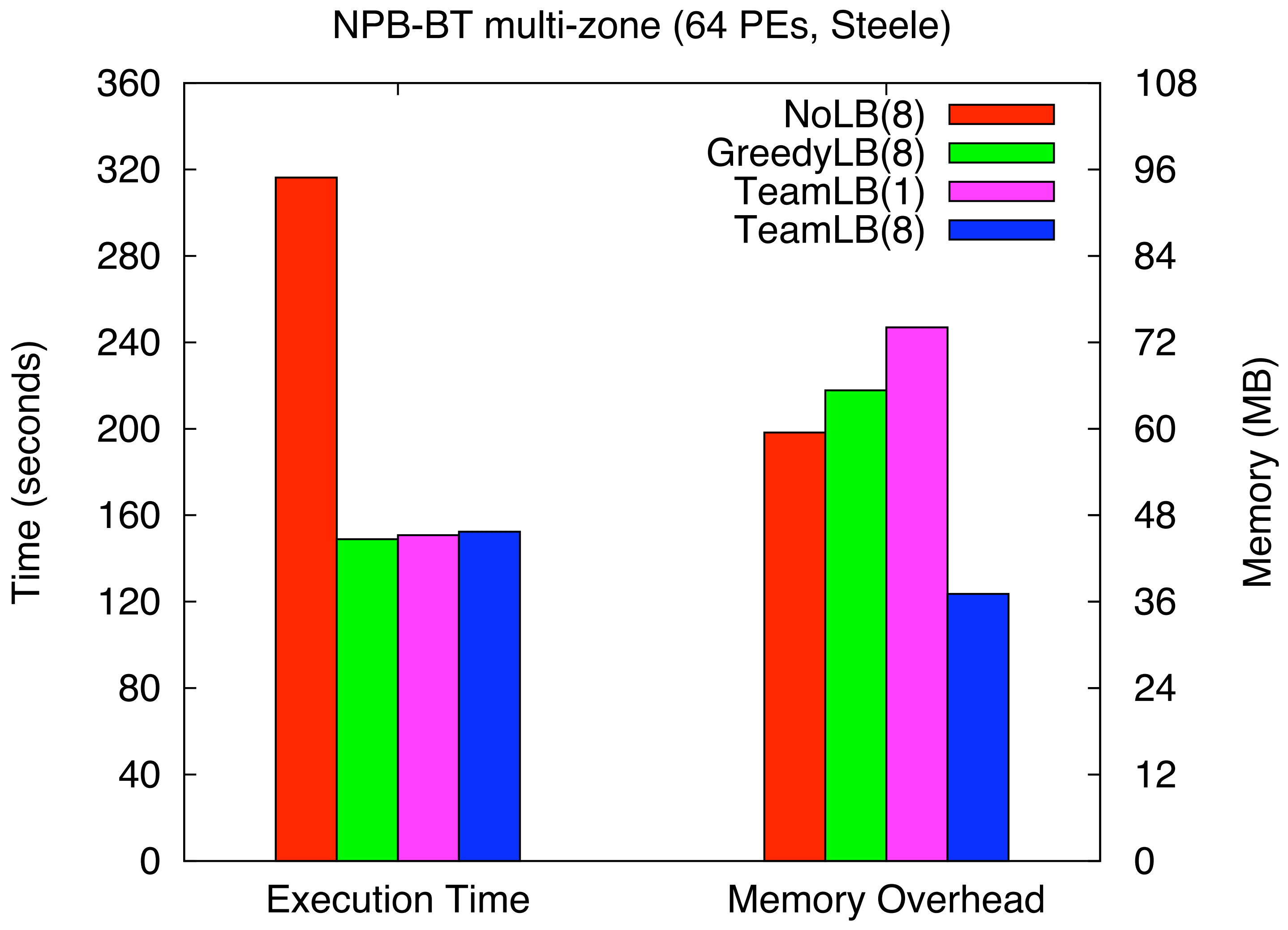
Result: Load Balance







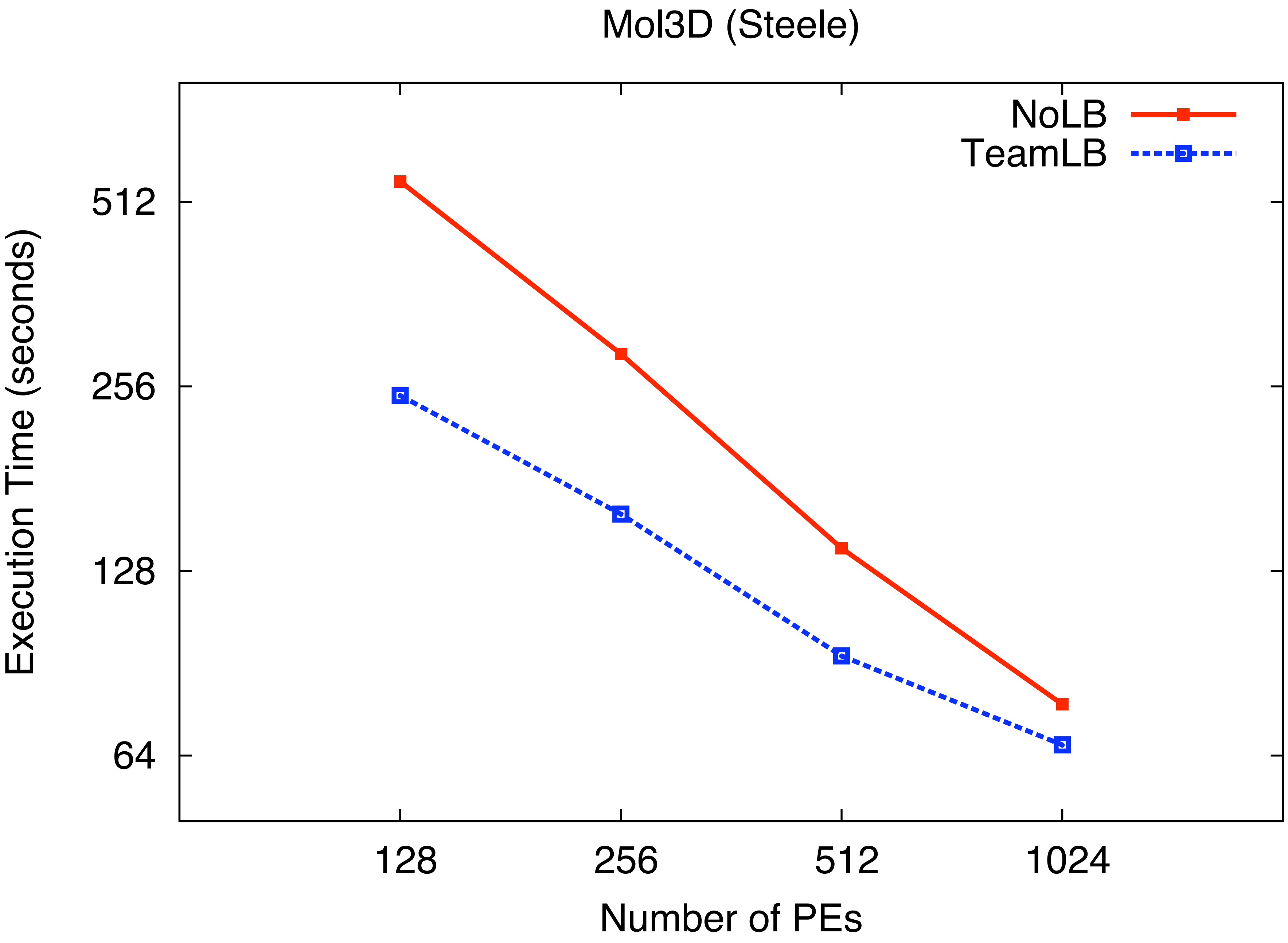
Result: Low Overhead



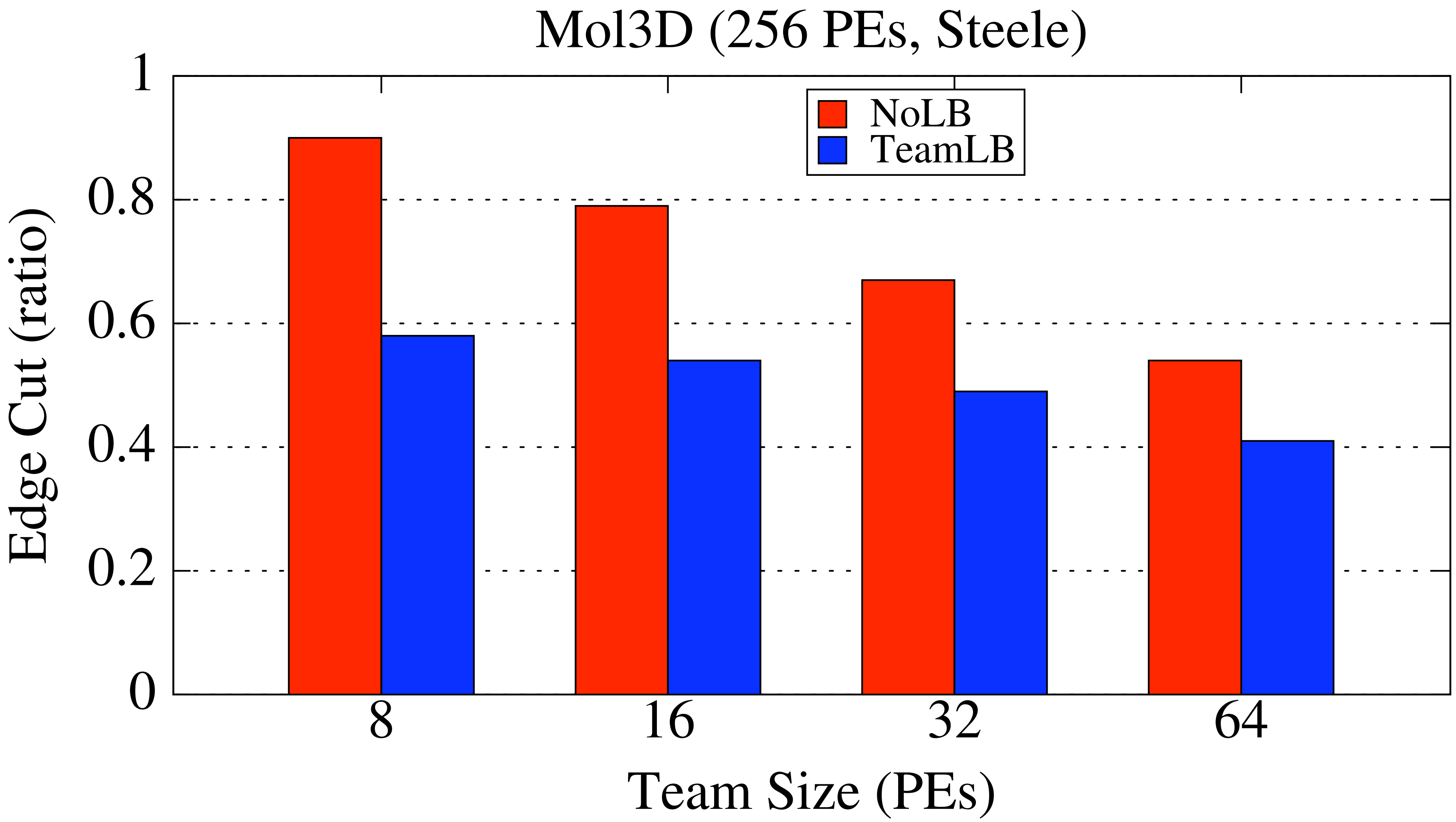
edge-cut ratio = 0.26

Clusters	
Number	8
Objects	256
Average Size	32
St. Dev. Size	13.76
Coeff. Variation	0.43
Max Size	53
Min Size	19

Result: Strong Scaling



Result: Team Size



Conclusions

- Migratable objects as a framework to provide fault tolerance and load balancing.
- Graph partitioning tools effective to drastically reduce memory overhead in message logging.
- Team size is a tradeoff parameter.

Contributions

- An extension to load balancing framework to dynamically form processor teams.
- A team-based algorithm for simple causal message logging.

Future Work

- More applications:
  - Adaptive mesh refinement.
  - Weather simulation.
- Incorporate team-based approach into SMP version.
- Test system with faster networks (Infiniband).
- Processor teams vs object teams.

Thank you!

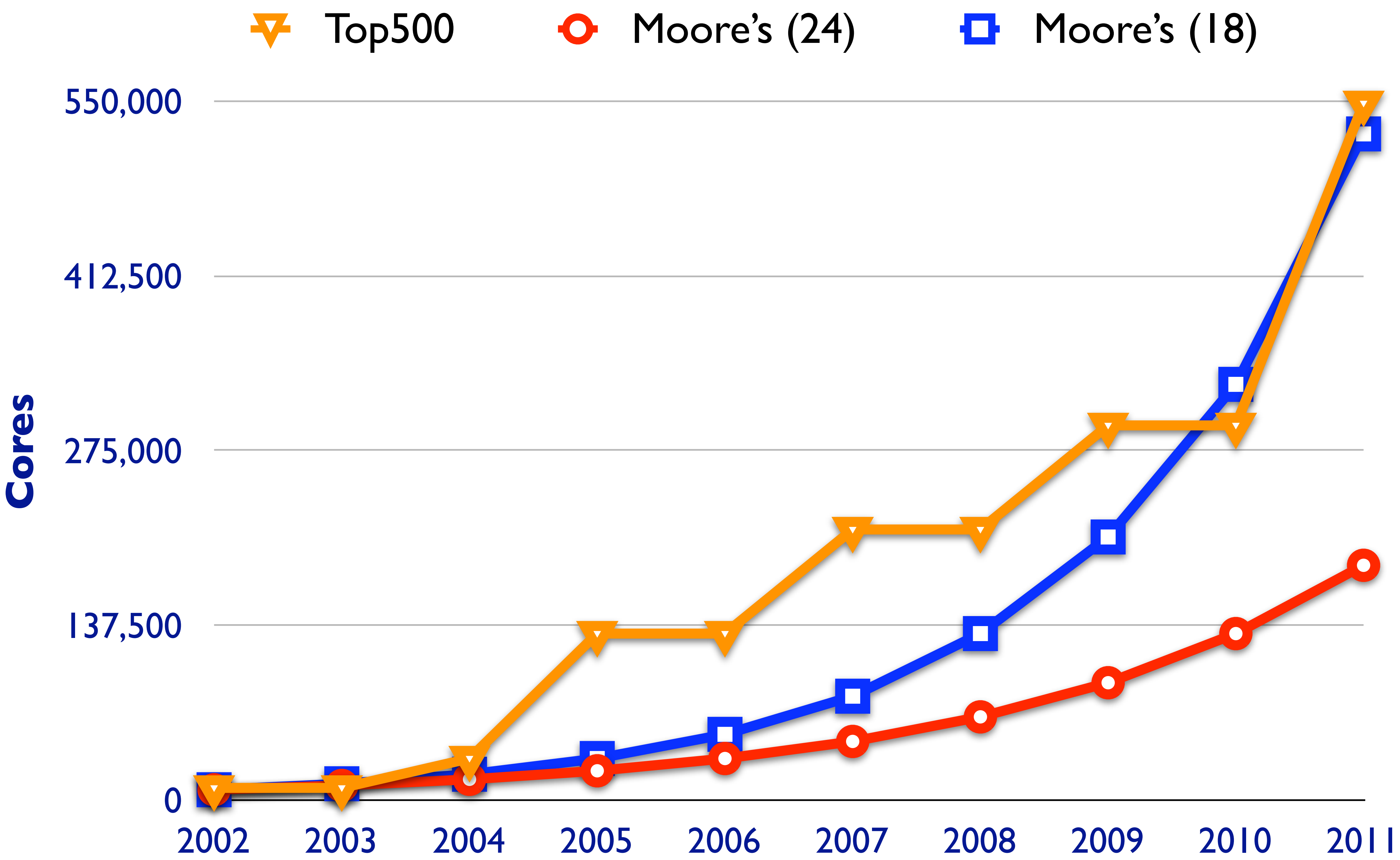
Q&A

Object Teams

- **Advantages**:
  - Natural to migratable-objects model.
  - Separation of concerns: load balance vs fault tolerance.
  - More flexibility for cluster formation.

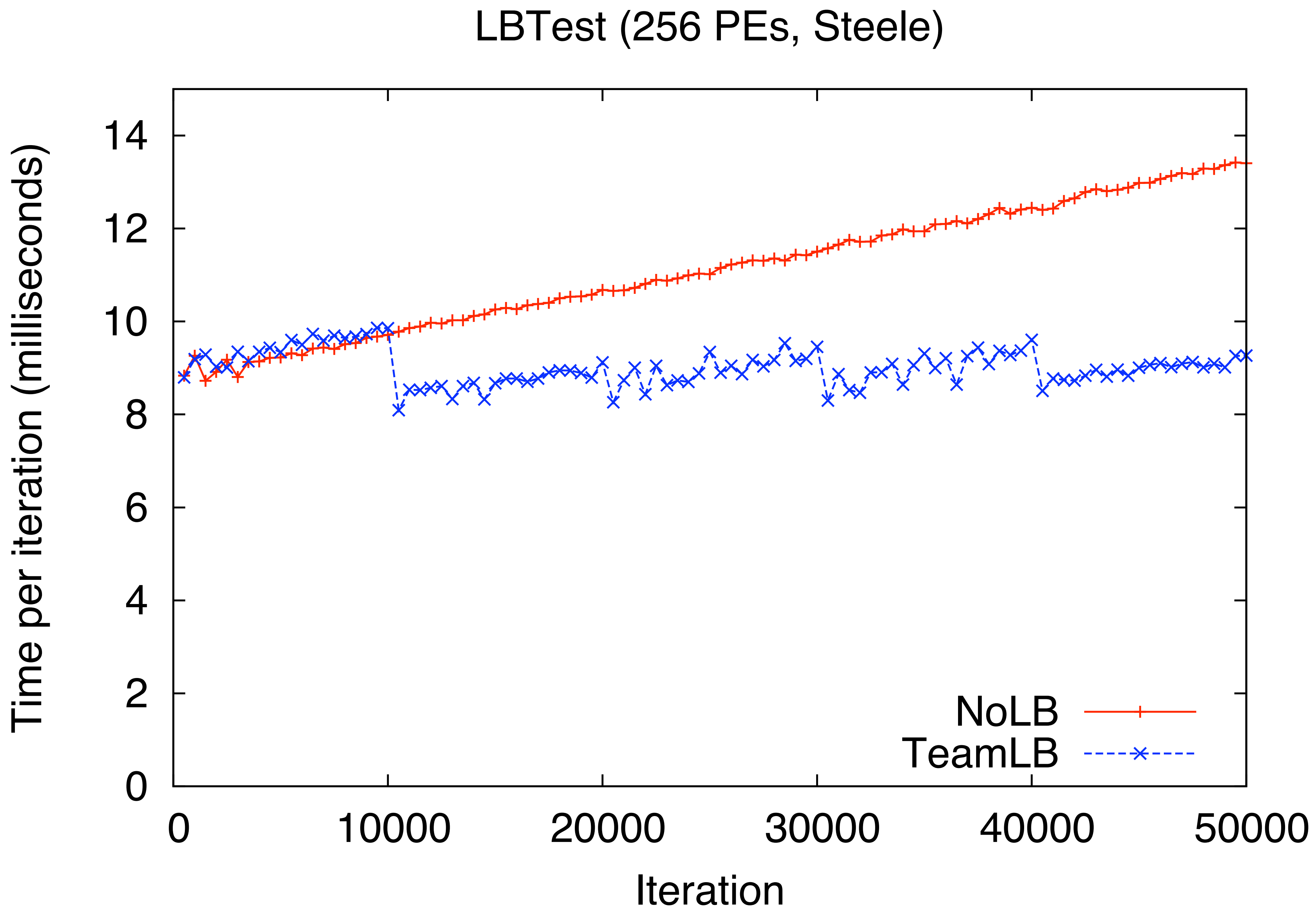
- **Disadvantages**:
  - In case of failure: unbounded number of teams to recover.
  - Some local messages have to be stored.
  - Load balancer involved in object distribution.

Moore's Law?



▽ Top500
○ Moore's (24)
□ Moore's (18)
550,000
412,500
275,000
137,500
0
Cores
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011

Result: Dynamic Teams


LBTest (256 PEs, Steele)
Time per iteration (milliseconds)
Iteration
NoLB
TeamLB