Joint Laboratory for Petascale Computation

Topology-aware load balancing for parallel applications on multi-core systems and beyond

Laércio Lima Pilla

and Christiane P. Ribeiro, François Broquedis,

Pierre Coucheney, Bruno Gaujal, Jean-François Méhaut,

Philippe O. A. Navaux, Laxmikant Kale

INF INSTITUTO DE INFORMÁTICA UFRGS

CAPES

GRENOBLE UNIVERSITÉ

INRIA

LIG

Overview

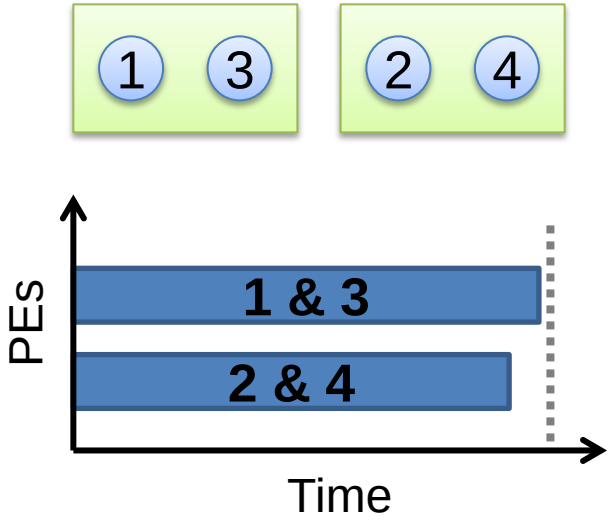
Asymptotically optimal load balancing algorithm for multi-core machines.

Motivation

- Example of environment: **CHARM++**
  - Parallel task overdecomposition
    - Chares 1 2 3
  - Platform independent
    - Processing elements 
  - Dynamic load balancing
    - Chare migration

Motivation: example

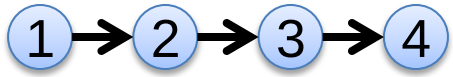
- Initial chare mapping
- Apply a load balancing algorithm
  - Based on **data provided by CHARM++**



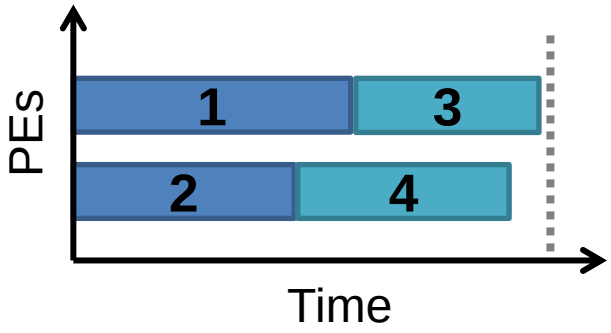
Chare

Processing Element (PE)

Motivation: example



- CHARM++ load balancing data
  - **Communication graph**
  - Current **chare mapping**
  - **Chares' load**

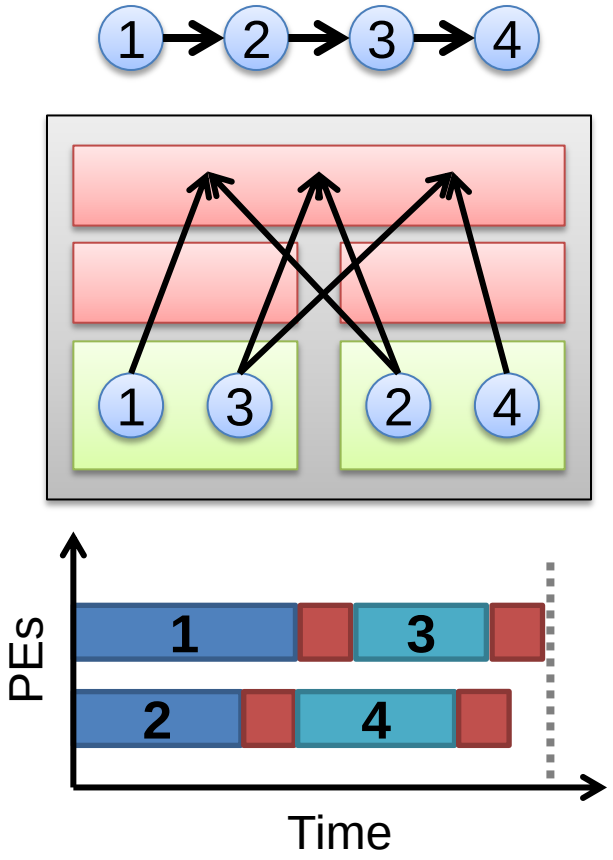


Chare

Processing Element (PE)

Motivation: example

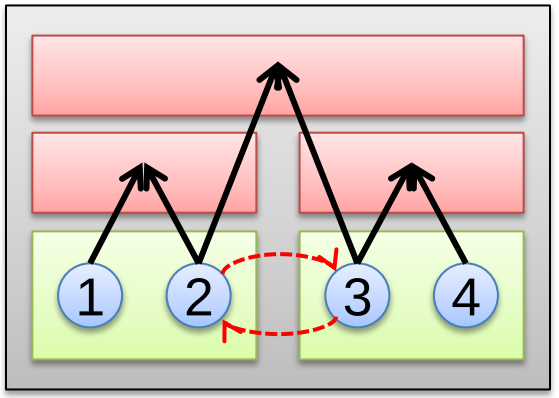
- Missing information
  - Communication costs
  - **Architectural information**
    - **Memory hierarchy**

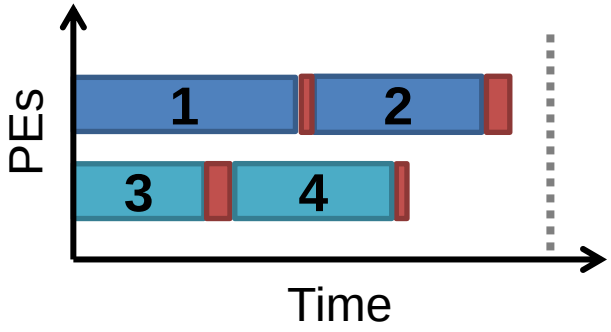


Chare Processing Element (PE) Cache

Motivation: example

- Knowledge about the **memory hierarchy**
  - Memory access costs
    - Reduce communication costs
  - **Highly hierarchical systems**
    - Even more on many-core





Chare Processing Element (PE) Cache

Motivation

- Our approach: **TopoLB**
  - Load balancing algorithm implemented on Charm++
  - Combines *application information* and the *machine topology*
  - Works on UMA and NUMA machines
  - Is asymptotically optimal

Agenda

~~Motivation~~

TopoLB

Experiments

And beyond

TopoLB: idea

- **Heuristic**
  - Load balancing is NP-Hard
  - No initial assumption about the application
- Improve performance (reduce makespan)
  - By reducing unbalance
  - By reducing communication costs
  - While avoiding migrations (data movement costs)

TopoLB: data

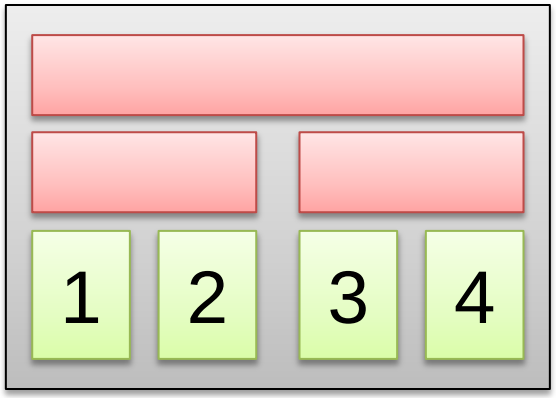
- ***Application information*** – provided by CHARM++
  - Chares' load
    - Reduce the makespan
  - Communication graph
    - Reduce the communication overhead
    - Bring together communicating chares
  - Current chare mapping
    - Avoid migration overheads

TopoLB: data

- ***Machine topology*** – our library
  - **Memory hierarchy**
    - Cache and memory sharing among cores
  - **Memory access latencies**
    - Estimate the communication time of each message through the memory hierarchy

Core

Cache

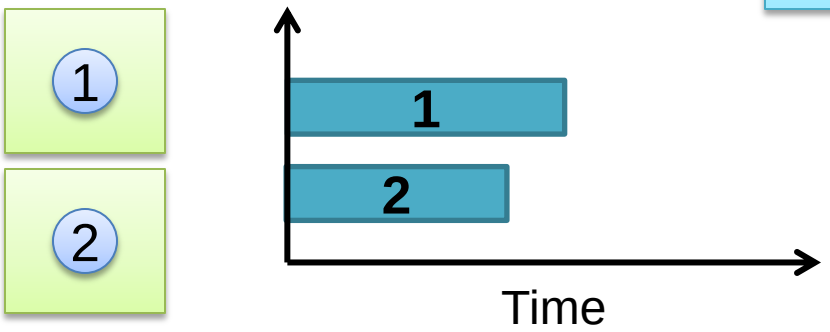


$$\begin{bmatrix} 1.63 & 1.63 & 6.67 & 6.67 \\ 1.63 & 1.63 & 6.67 & 6.67 \\ 6.67 & 6.67 & 1.63 & 1.63 \\ 6.67 & 6.67 & 1.63 & 1.63 \end{bmatrix}$$

TopoLB: algorithm

- Each chare has a **load** based on
  - Computation time
  - Communication time
    - **Number of messages** exchanged
    - Memory **latencies**

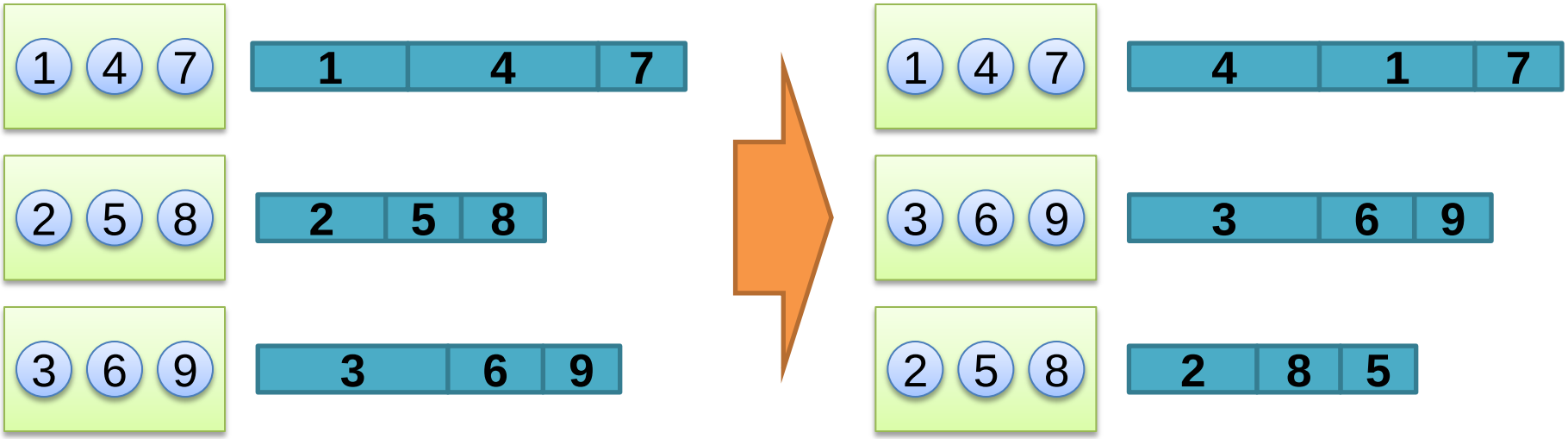
Communicating chares have a higher memory affinity



Chare Processing Element (PE)

TopoLB: algorithm

- Initial mapping
- **Sort** PEs (processors) **based on** their **load**

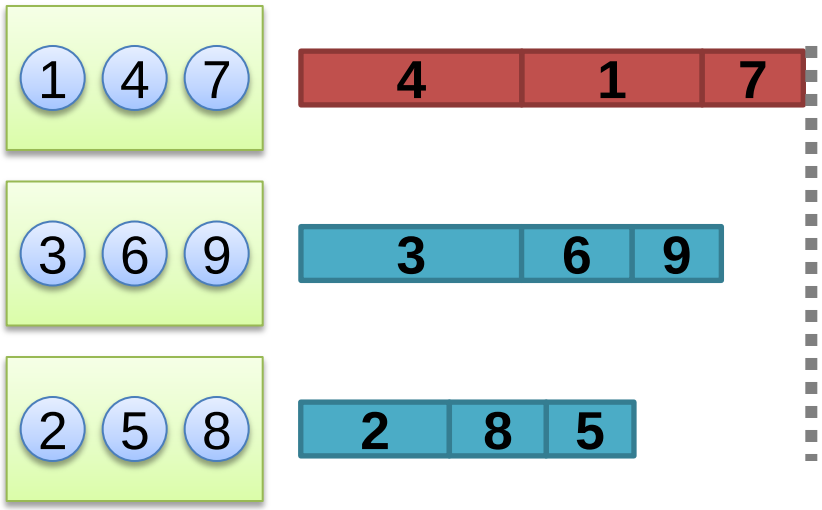


Chare

Processing Element (PE)

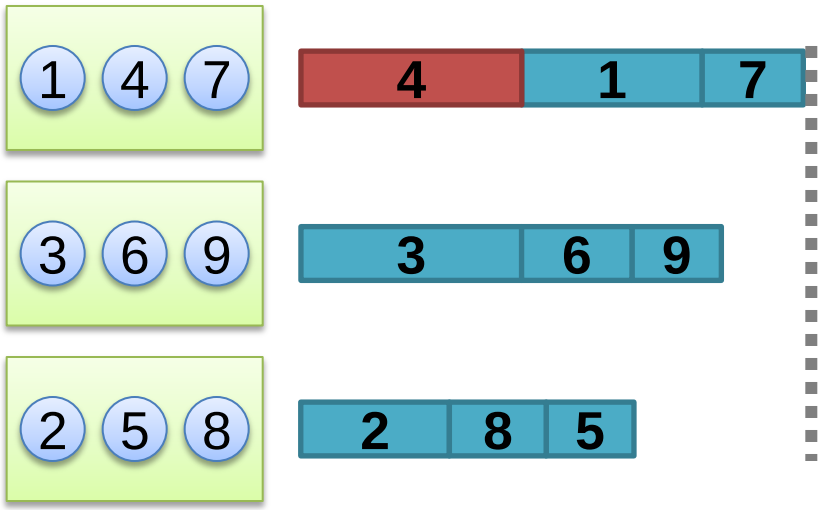
TopoLB: algorithm

- Tries to **migrate chares from the most loaded processor** with a **probability α**
  - Or chooses another processor uniformly



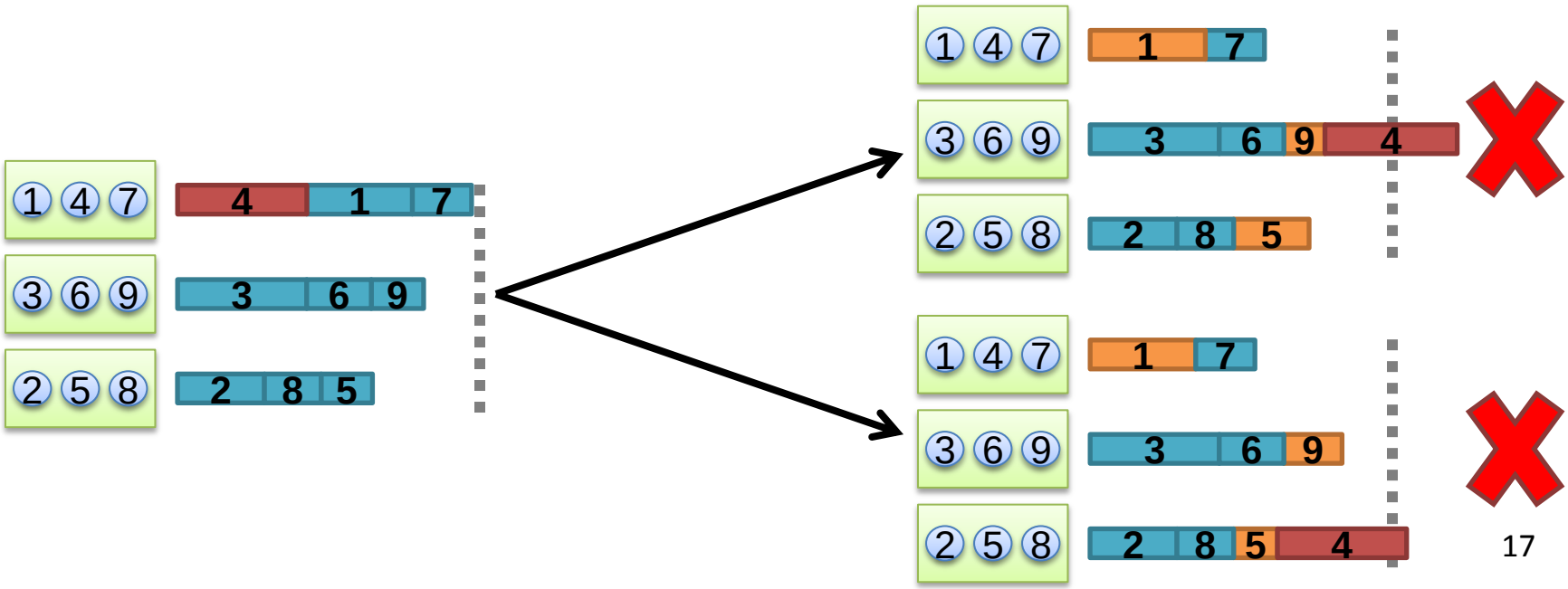
TopoLB: algorithm

- Starts from the **heaviest chare** with a **probability β**
  - Or chooses another chare uniformly
  - From heaviest chare to lightest chare



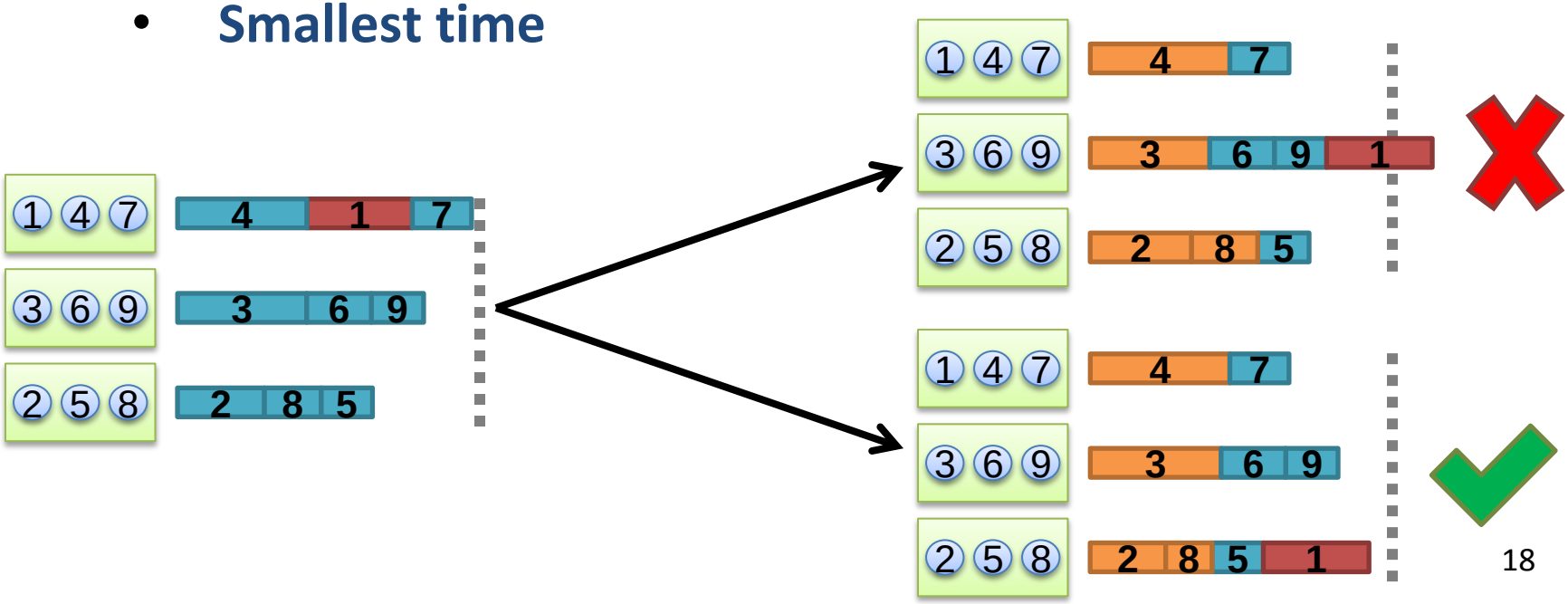
TopoLB: algorithm

- A migration might decrease the makespan
- A migration may affect other chares
  - Communication time may change



TopoLB: algorithm

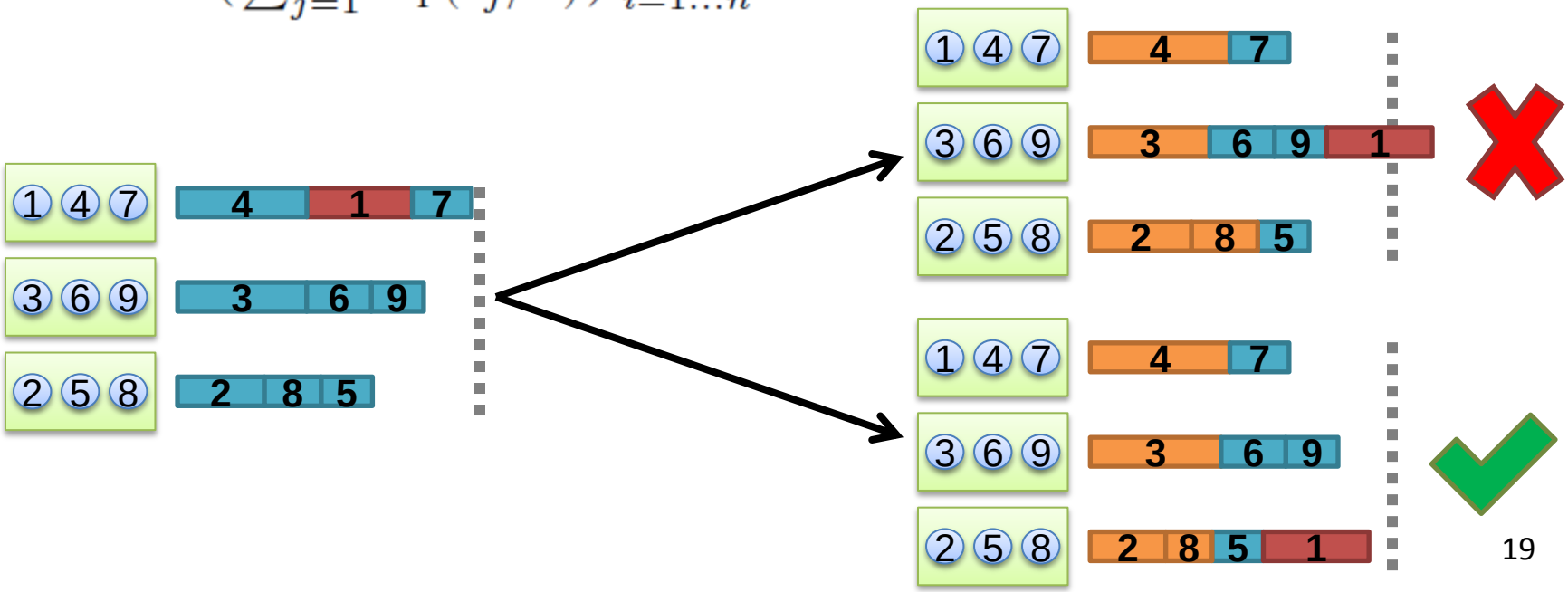
- Evaluates all possible processors
  - Migrates the chare to the **processor that minimizes the makespan** with high probability
    - **Smallest time**



TopoLB: algorithm

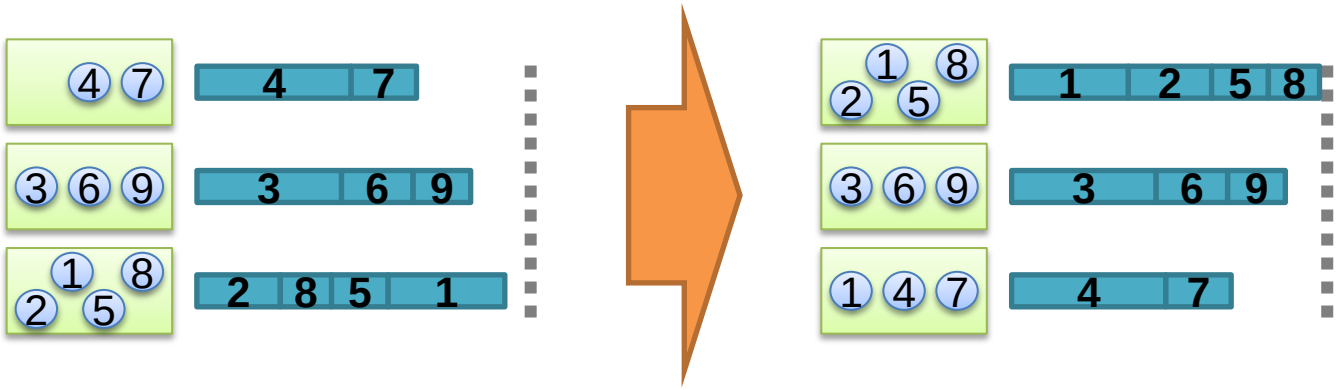
- **High probability**
  - Using a **Gibbs distribution** with a **temperature *T***

$$\left(\frac{\exp(v_i/T)}{\sum_{j=1}^{n} \exp(v_j/T)} \right)_{i=1\ldots n}$$



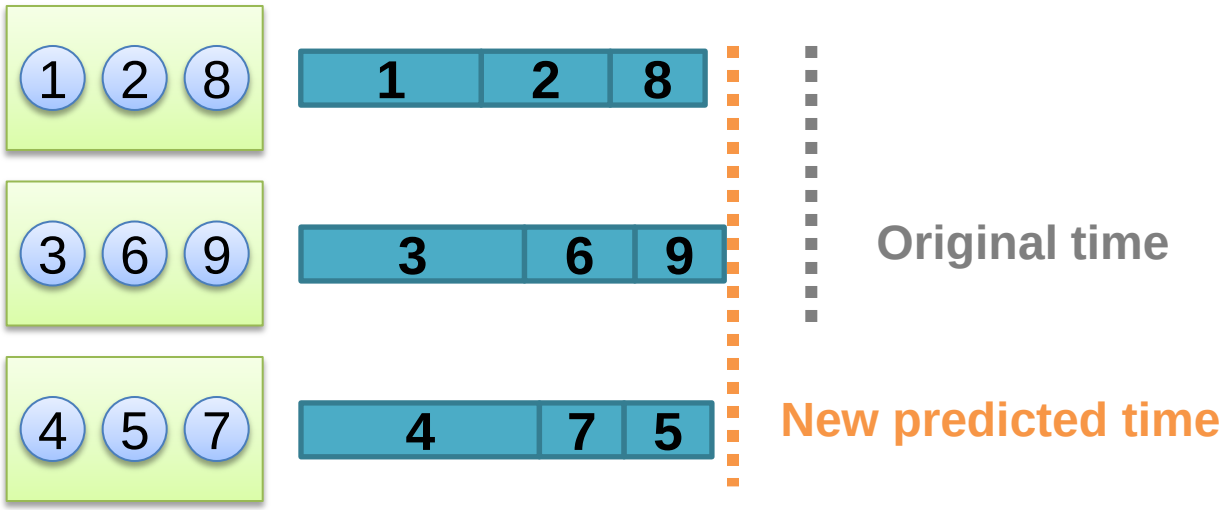
TopoLB: algorithm

- Continues **until no chare migrates from the heaviest processor**
  - Cannot reduce the makespan
    - Bound to the heaviest processor



TopoLB: algorithm

- Continues **until no chare migrates from the heaviest processor**
  - Cannot reduce the makespan
    - Bound to the heaviest processor



TopoLB: implementation details

- **Exponential backoff**
  - Reduce the load balancing overhead by not computing the algorithm on all LB calls
  - Launched when **no migrations happen**

Agenda

~~Motivation~~

~~TopoLB~~

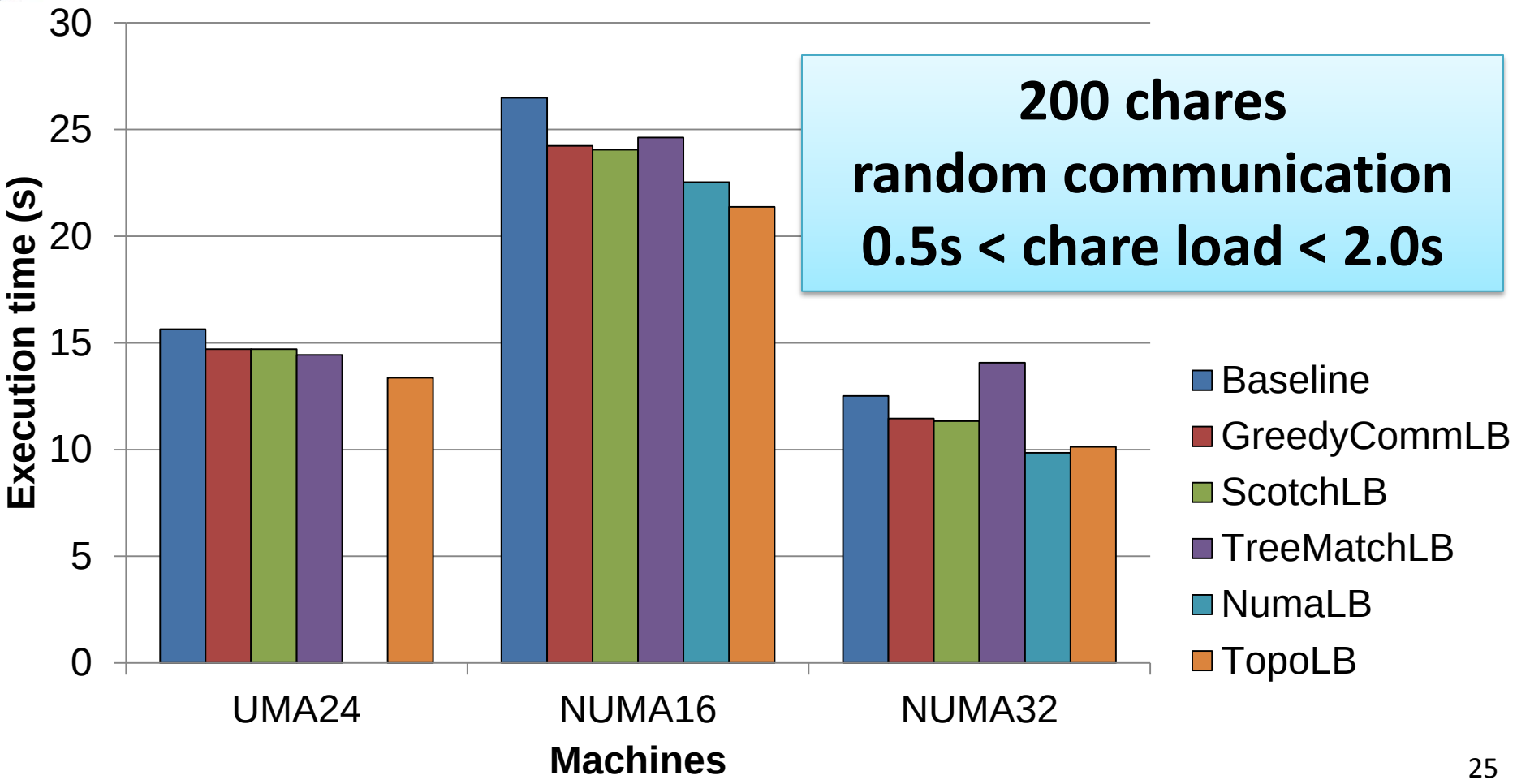
Experiments

And beyond

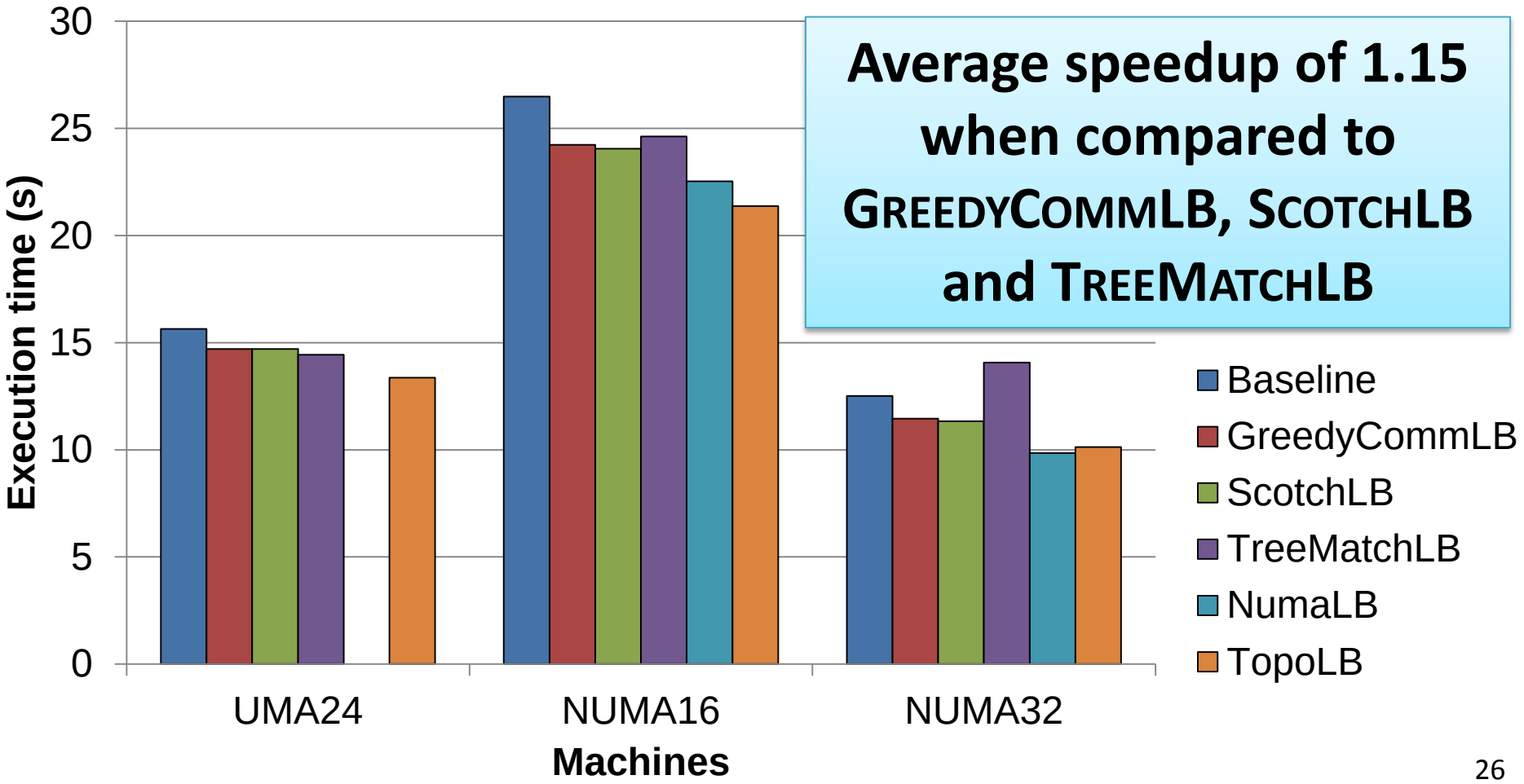
Experiments

- 3 different machines: UMA24, NUMA16 and NUMA32
- 4 other load balancers: GreedyCommLB, ScotchLB, TreeMatchLB and NumaLB
- 2 benchmarks: lb_test and mol3D
- α, β close to 1, *T* close to 0

lb_test

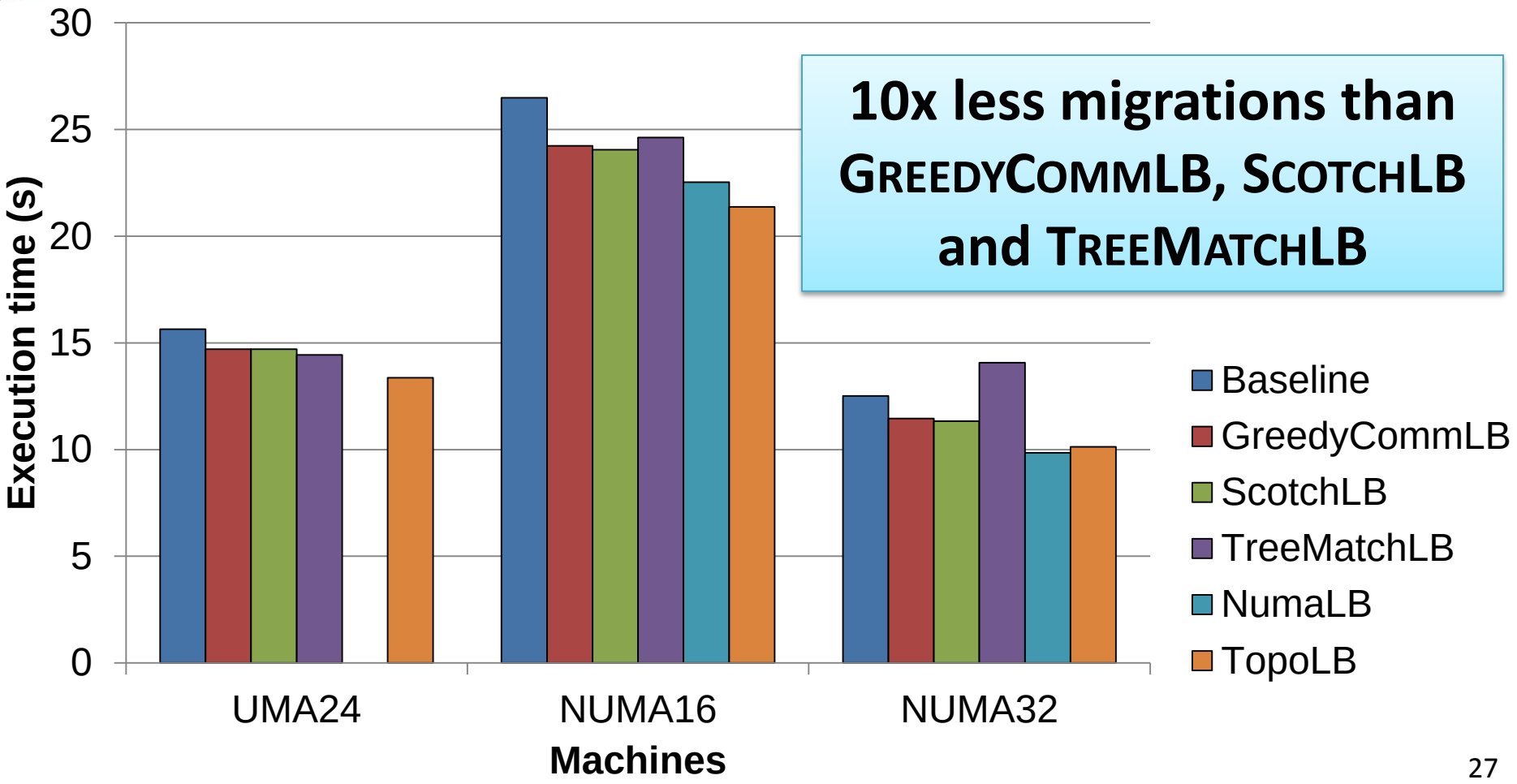

30
25
20
15
10
5
0
Execution time (s)
UMA24
NUMA16
NUMA32
Machines
200 chares
random communication
0.5s < chare load < 2.0s
Baseline
GreedyCommLB
ScotchLB
TreeMatchLB
NumaLB
TopoLB

lb_test



Average speedup of 1.15 when compared to GREEDYCOMMLB, SCOTCHLB and TREEMATCHLB

lb_test



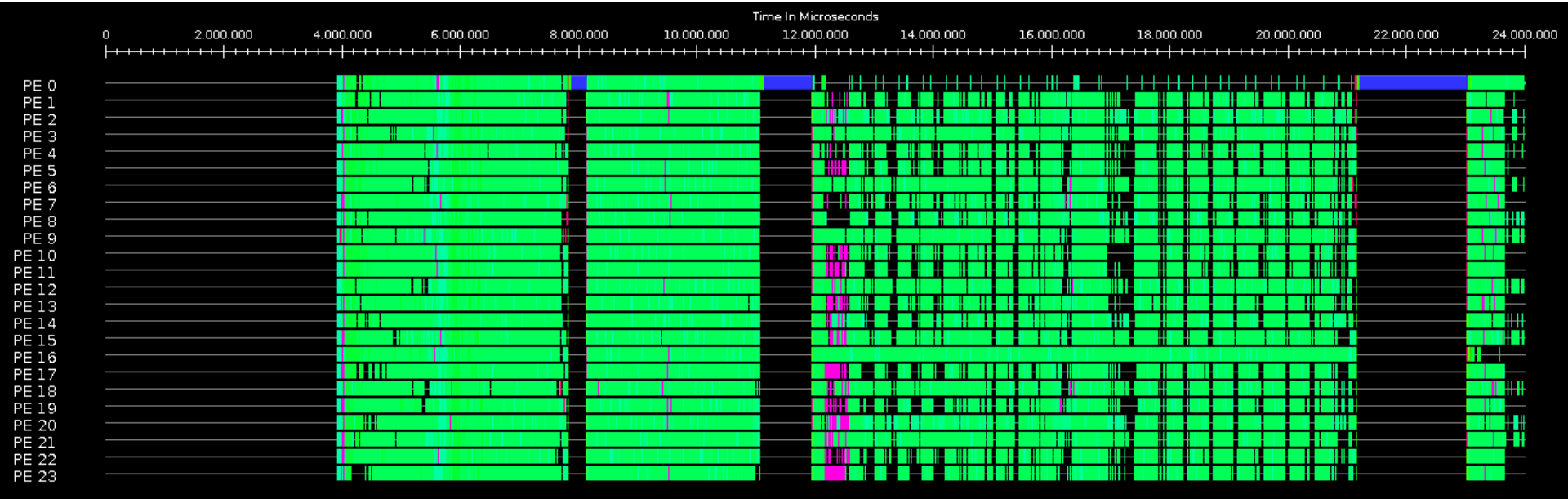
10x less migrations than GREEDYCOMMLB, SCOTCHLB and TREEMATCHLB

mol3D

- Apolipoprotein-A1
- **No load balancer improved performance**
- **TopoLB** performed **30x less migrations** than other load balancers
  - But took **2 to 3x more time on its load balancing decisions**

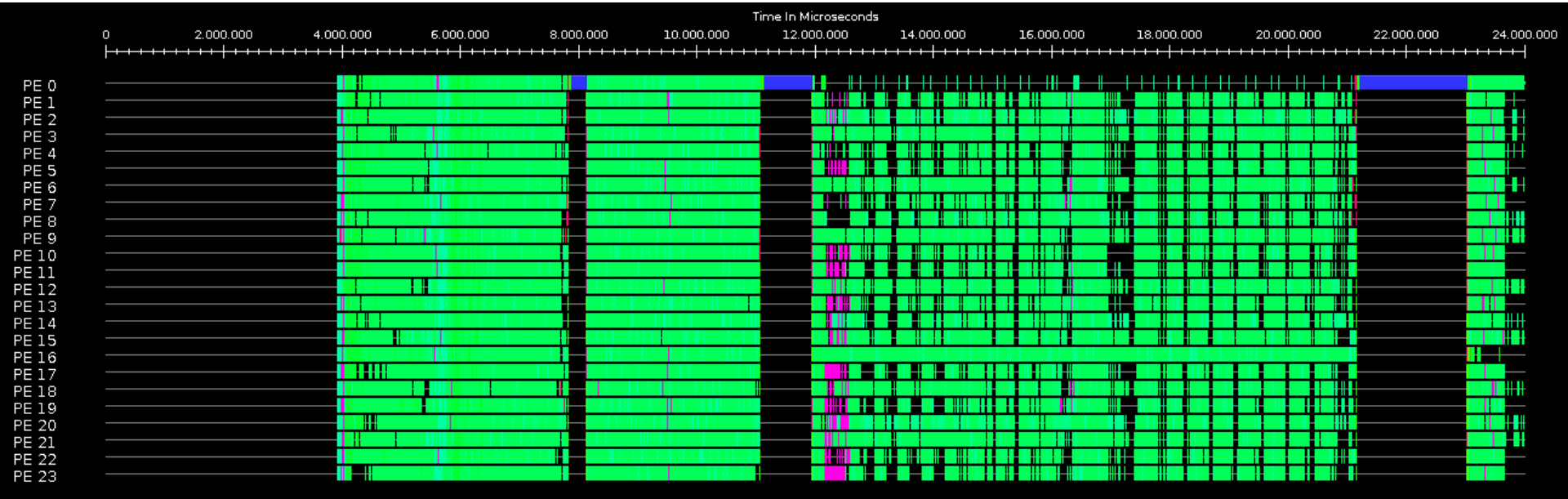
mol3D

- View using Projections
  - 3x20 iterations, 3 calls to TopoLB
  - ~7000 chares



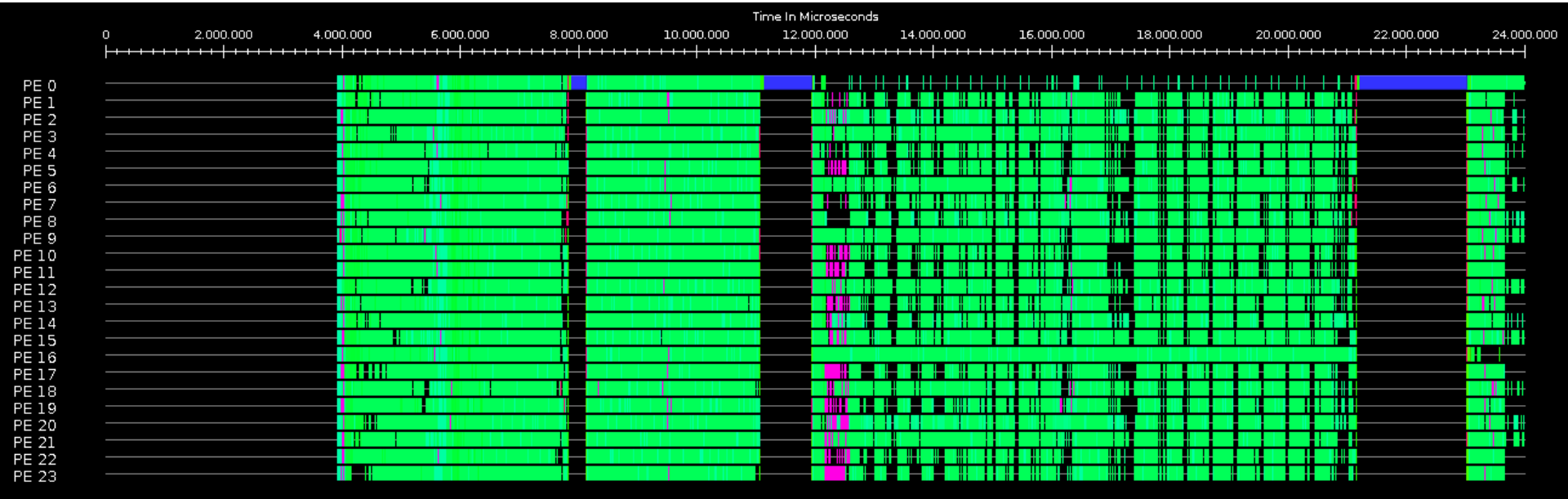
mol3D

Application starts balanced



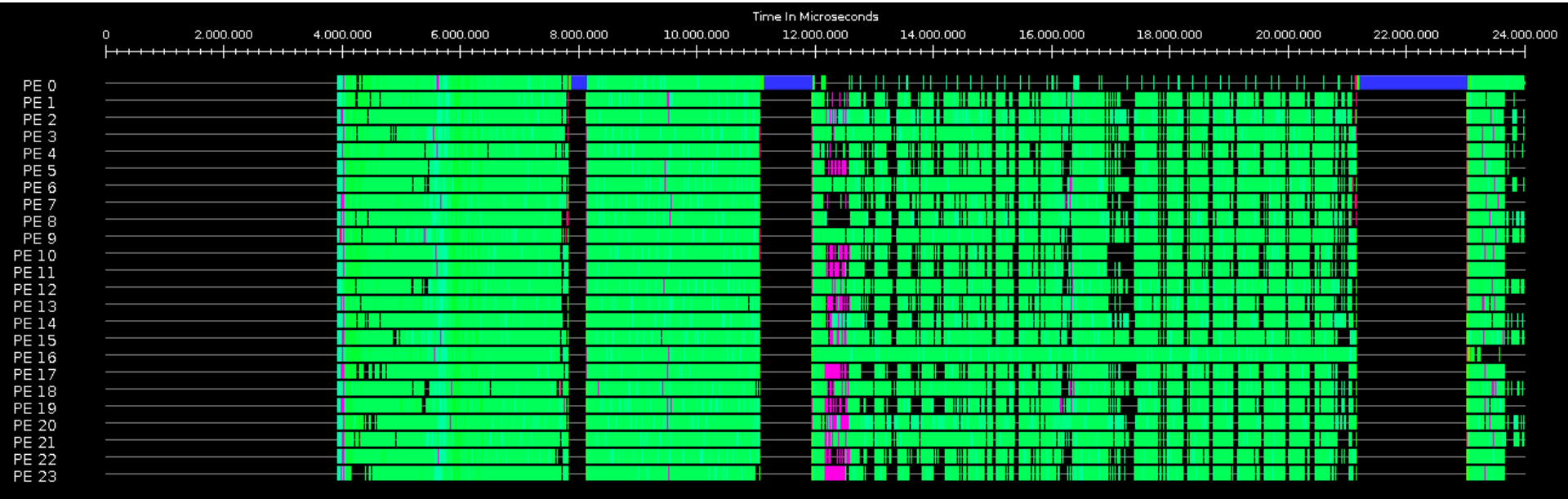
mol3D

TopoLB improves the iteration time



mol3D

Iteration time diverges strongly



Agenda

~~Motivation~~

~~TopoLB~~

~~Experiments~~

And beyond

And beyond
(concluding remarks)

- TopoLB presented a
  - Small number of migrations
  - Fast convergence
  - High computational cost (overhead)
    - Reduced by the exponential backoff

And beyond

- Working to extend TopoLB to **clusters of multi-core machines**
  - Two levels: Another LB for the cluster, TopoLB for the compute nodes
    - TopoLB can be **too costly** for large machines and applications

And beyond

- Working to **better understand and evaluate** the behavior of CHARM++ **load balancers**
  - Debugging library
  - Use real applications

- Added part of our machine model to HWLOC

Joint Laboratory for Petascale Computation

Topology-aware load balancing for parallel applications on multi-core systems and beyond

Thank you.

Laércio Lima Pilla

Contact: llpilla@inf.ufrgs.br

INSTITUTO DE INFORMÁTICA UFRGS

CAPES

GRENOBLE UNIVERSITÉ

INRIA

LIG