Load Balancing for Parallel Multi-core Machines with Non-Uniform Communication Costs

Laércio Lima Pilla
laercio.lima-pilla@imag.fr

LIG Laboratory – Inria – Grenoble University
Grenoble, France

Institute of Informatics – Federal University of Rio Grande do Sul
Porto Alegre, Brazil

Load Balancing for Parallel Multi-core Machines with Non-Uniform Communication Costs

Laércio Lima Pilla, Christiane Pousa Ribeiro,
Daniel Cordeiro, Chao Mei, Abhinav Bhatele,
Philippe O. A. Navaux, François Broquedis,
Jean-François Méhaut, Laxmikant V. Kale,
Pierre Coucheney, Bruno Gaujal

Agenda

- **Introduction**
  - Applications
  - Parallel Machines
  - Issues and Objectives
- **Load Balancing Approach**
  - General Idea
  - Required Information
  - NucoLB
  - TopoAwareLB
- **Performance Evaluation**
  - Platforms
  - Load balancers
  - Benchmarks and Application Results
- **Concluding Remarks**

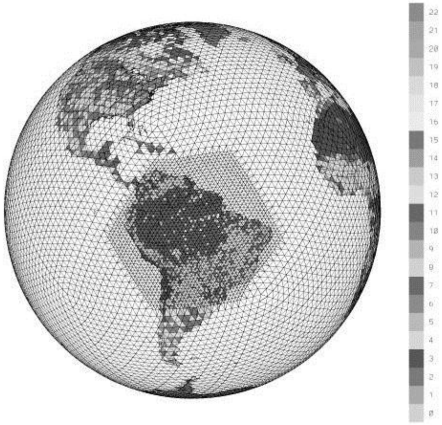
Agenda

- **Introduction**
  - Applications
  - Parallel Machines
  - Issues and Objectives
- **Load Balancing Approach**
  - General Idea
  - Required Information
  - NucoLB
  - TopoAwareLB
- **Performance Evaluation**
  - Platforms
  - Load balancers
  - Benchmarks and Application Results
- **Concluding Remarks**

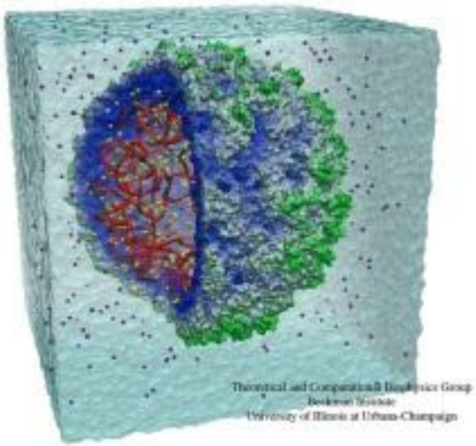
Introduction

Applications

- **Irregular Applications**
  - Unpredictable, dynamic behaviour
  - Input-dependent
  - Compute intensive (HPC)
  - Iterative simulations
  - **Load imbalance**
  - **Complex communication** patterns



Climatology



Molecular Dynamics

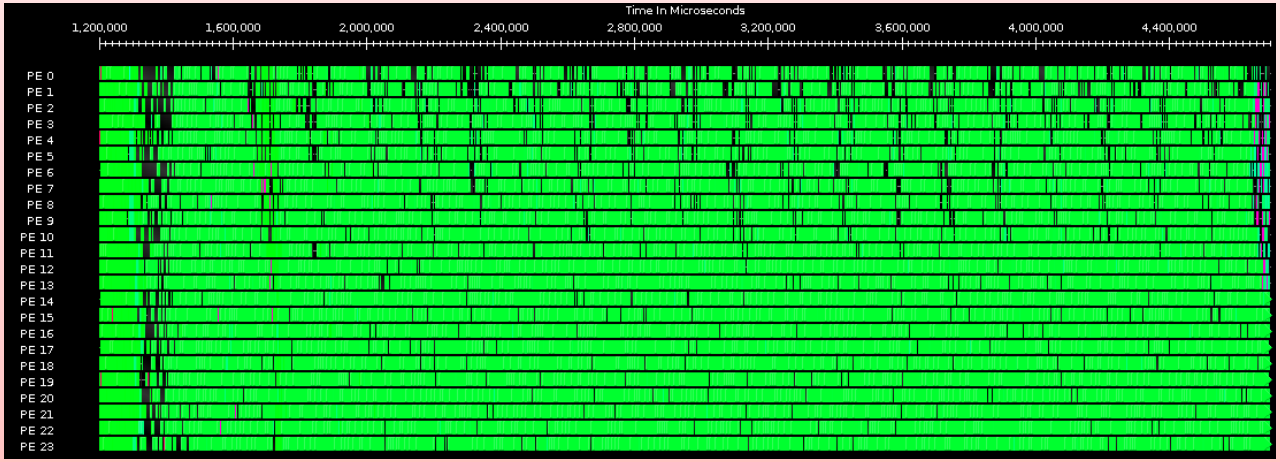
Introduction

Applications

- **Irregular Applications**

Time

Cores



Example: **LeanMD** on a 48 cores machine (24 cores shown)

- **Molecular dynamics application** written in Charm++
- 1875 tasks for 3.5 seconds (20 iterations)
- **Average core usage: 87.5%**
- Worst core: 65% usage

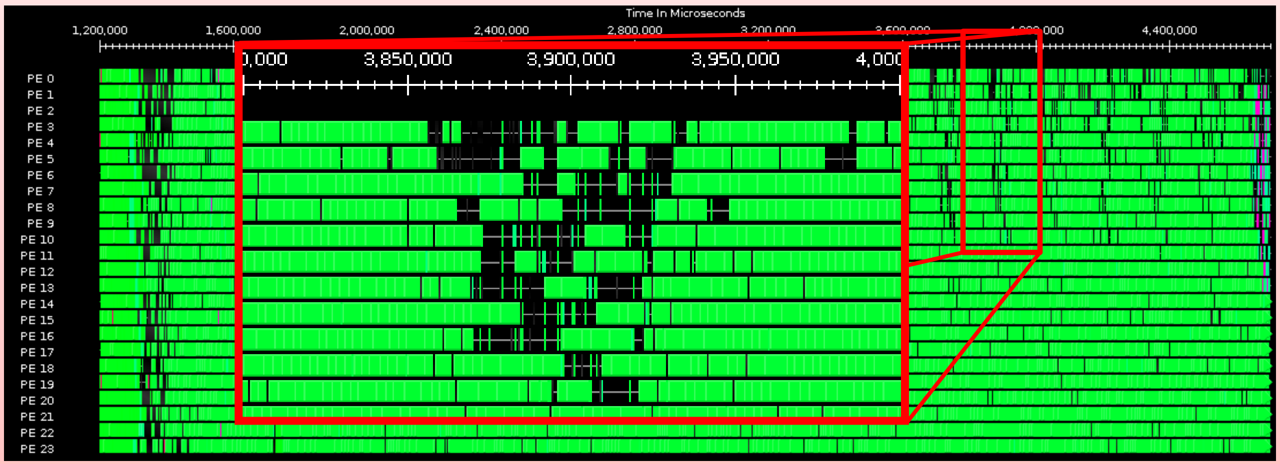
Introduction

Applications

- **Irregular Applications**

Time

Cores



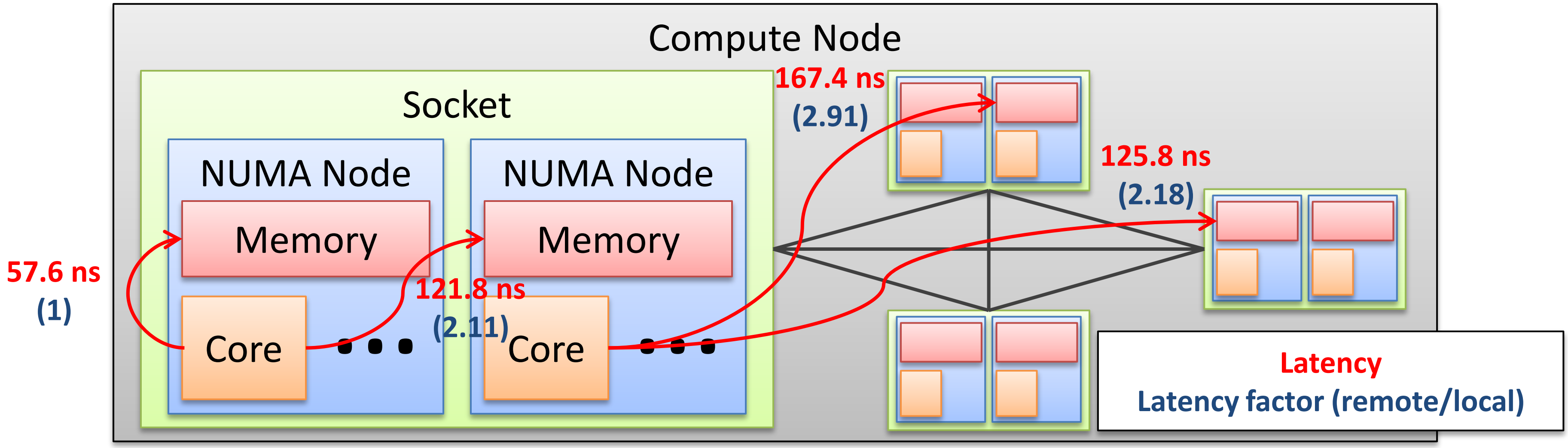
Example: **LeanMD** on a 48 cores machine (24 cores shown)

- **Molecular dynamics application** written in Charm++
- 1875 tasks for 3.5 seconds (20 iterations)
- **Average core usage: 87.5%**
- Worst core: 65% usage

Introduction

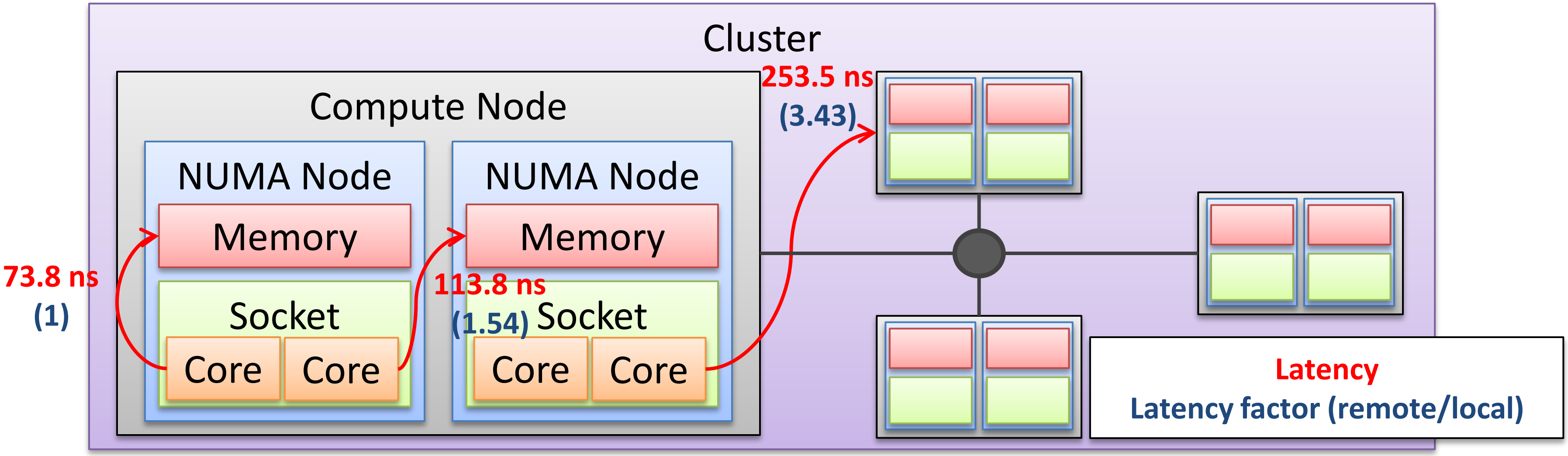
Parallel Machines

- **[Highly] Hierarchical Architectures**
  - **Memory and network** hierarchies
  - **Asymmetric communication costs**
    - Latency, bandwidth
  - Component sharing



Introduction

- **[Highly] Hierarchical Architectures**
  - **Memory and network** hierarchies
  - **Asymmetric communication costs**
    - Latency, bandwidth
  - Component sharing



Introduction

Issues and Objectives

- **Issues**
  - **Imbalanced** processors (idleness)
  - **Communication overheads** (lack of affinity)
- **Opposing problems**
  - Improving one may worsen the other

Introduction

- **Objectives**
  - Improve performance
  - Optimize resource usage
    - **Reduce** processor **idleness**
    - **Reduce communication costs**
    - Find the best **trade-off**
  - **Performance portability**
    - Different platforms, different applications

Agenda

- **Introduction**
  - Applications
  - Parallel Machines
  - Issues and Objectives
- **Load Balancing Approach**
  - General Idea
  - Required Information
  - NucoLB
  - TopoAwareLB
- **Performance Evaluation**
  - Platforms
  - Load balancers
  - Benchmarks and Application Results
- **Concluding Remarks**

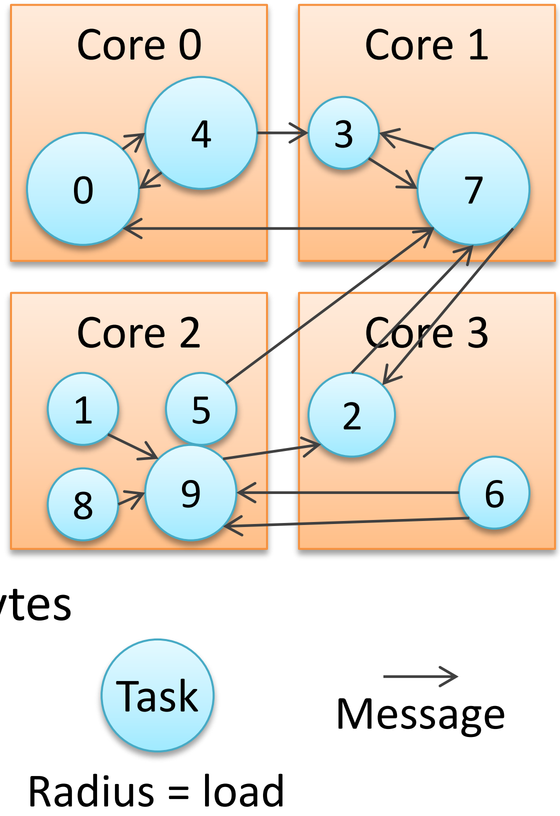
Load Balancing Approach

- **General Ideas**
  - **Combine** knowledge about
    - **Application**
    - **Machine topology**
  - **Load balancing**
    - Change task mapping dynamically
  - Avoid task migrations
    - Data movement costs

Load Balancing Approach

Required Information

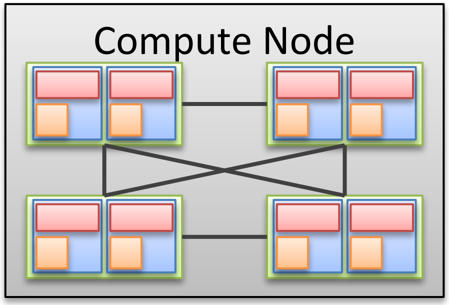
- **Discover the Application Behaviour**
  - **Profile the** parallel **execution**
    - Runtime system level
  - Collect
    - Task **computational loads**
      - Execution times
    - **Communication graph**
      - Number of messages between tasks, bytes
    - Current task mapping



Load Balancing Approach

Required Information

- **Explore Platform Topology**
  - **Cluster configuration**
    - Network map, routers, nodes
  - **Intra-node hierarchy**
    - Sockets, cores, NUMA nodes, caches, memory
  - **Benchmarked communication costs**
    - lmbench, stream, MPI ping-pong
    - Latency, bandwidth, derivatives



Latency matrix for NUMA nodes (ns)

	0	1	2	3	4	5	6	7
0	57.67	**121.8**	126.9	167.4	126.7	166.3	125.8	165.2
1	**122.4**	57.22	126.7	166.8	167.6	126.6	166.2	124.3
2	127.0	125.9	57.86	121.8	125.2	124.3	127.1	166.0
3	167.6	165.5	121.6	57.05	126.0	124.9	167.5	125.6
4	126.4	165.5	124.9	125.3	57.46	121.3	126.3	126.1
5	167.6	126.2	124.7	125.1	122.2	57.18	167.2	166.0
6	125.6	165.4	127.0	165.6	126.4	165.9	58.40	121.7
7	165.4	124.3	168.5	126.2	127.2	168.0	122.6	57.17

Load Balancing Approach

Algorithms

- **Load Balancing Algorithm**
  - **Dynamically** correct imbalance
  - Improve communications
  - Current work
    - **NucoLB**
      - **Load balancer for machines with non-uniform communication costs**
    - MultiCoreLB
      - Considers cache hierarchy on its decisions
    - **TopoAwareLB**
      - Provides optimality guarantees

Load Balancing Approach

- **NucoLB**
  - Non-uniform communication costs load balancer
  - **Greedy list scheduling algorithm**
    - Map the task with the biggest execution time to the core with the smallest cost
  - Integrated on **Charm++**
    - Charm++ Parallel System – UIUC
    - C++-based parallel programming model and RTS
    - **Mature load balancing framework**
      - Load balancing plugins
      - **Measurement-based**
    - Tasks = objects = chares

Load Balancing Approach

- **NucoLB**
  - Application information:
    - Current task mapping
    - Task computational load
    - Communication graph (number of messages)
  - Machine topology model:
    - Hierarchy
    - **NUCO factor (latency factor)**
      - NUMA factor inside a machine
      - Network factor in clusters
      - Pre-computed in the target platform

Load Balancing Approach

NucoLB

- **NucoLB: algorithm**

Input: T set of tasks, C set of cores,
M mapping of tasks to cores
Output: M' mapping of tasks to cores

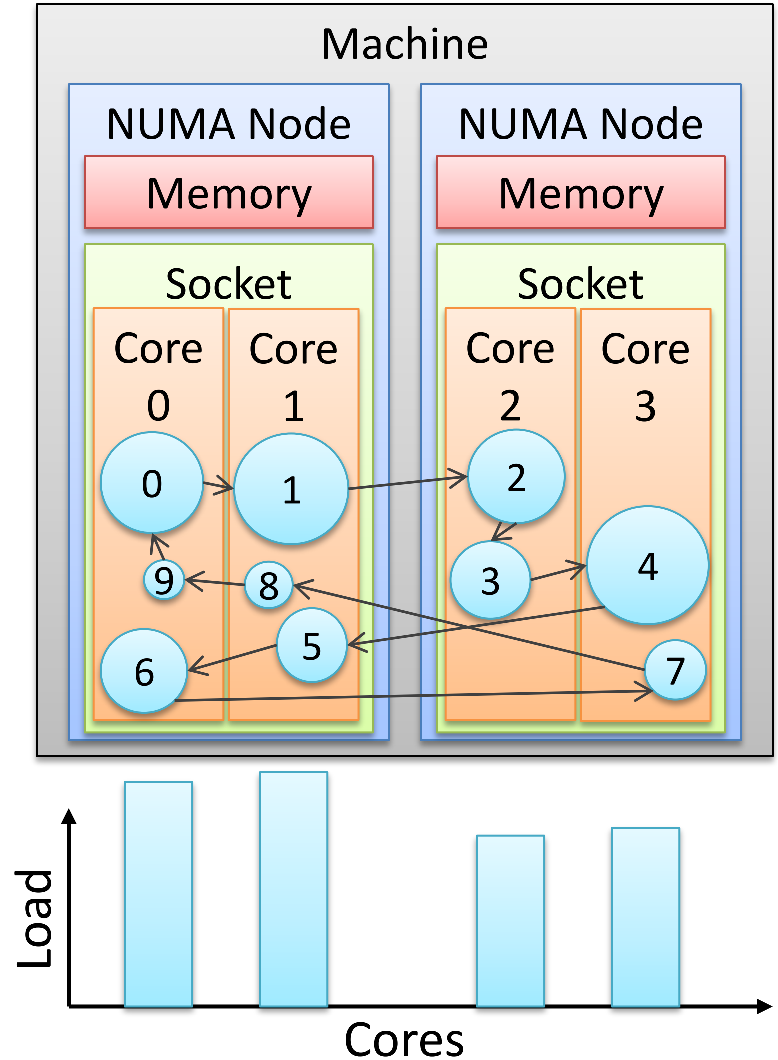
1. $M' \leftarrow M$
2. while $T \neq \emptyset$ do
3. $\quad t \leftarrow k \mid k \in \arg\max_{l \in T} \mathrm{load}(l)$
4. $\quad T \leftarrow T \setminus \{t\}$
5. $\quad c \leftarrow p, p \in C \wedge \{t,p\} \in M$
6. $\quad \mathrm{load}(c) \leftarrow \mathrm{load}(c) - \mathrm{load}(t)$
7. $\quad M' \leftarrow M' \setminus \{(t, c)\}$
8. $\quad c' \leftarrow p \mid p \in \arg\min_{q \in C} \mathrm{cost}(q, t)$
9. $\quad \mathrm{load}(c') \leftarrow \mathrm{load}(c') + \mathrm{load}(t)$
10. $M' \leftarrow M' \cup \{(t, c')\}$

where:

- t, k, and l are tasks;
- c, c', p, and q are cores;
- $\mathrm{load}(t)$ is the execution time of t in seconds;
- $\mathrm{load}(c)$ is the sum of the execution times of the tasks mapped to c in seconds;
- $\mathrm{cost}(q, t)$ is the cost to map ***t*** to ***q***, as
  $\mathrm{cost}(q, t) = \mathrm{load}(q) + \alpha * \lceil \mathrm{r_{comm}}(t, q)$
  $*\mathrm{NucoFactor}(\mathrm{comm}(t), \mathrm{node}(q)) - \mathrm{l_{comm}}(t, q) \rceil$
- $\mathrm{l_{comm}}(t, q)$ is the number of local messages sent to t by tasks on the NUMA node of q;
- $\mathrm{r_{comm}}(t, q)$ is the number of remote messages sent to t by tasks on other NUMA nodes; and
- $\mathrm{NucoFactor}(\mathrm{comm}(t), \mathrm{node}(q))$ represents latency factors between the NUMA node of q and other NUMA nodes of tasks that communicate with t.
- α is a constant to weight the communications

Load Balancing Approach

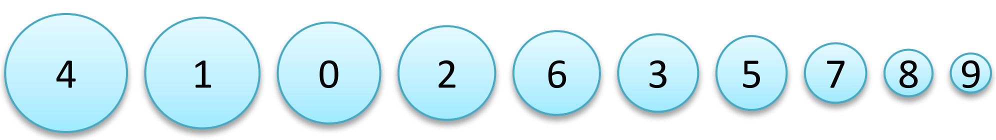
- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - Evaluate the cost function for each core
  - Choose the core with the smallest cost

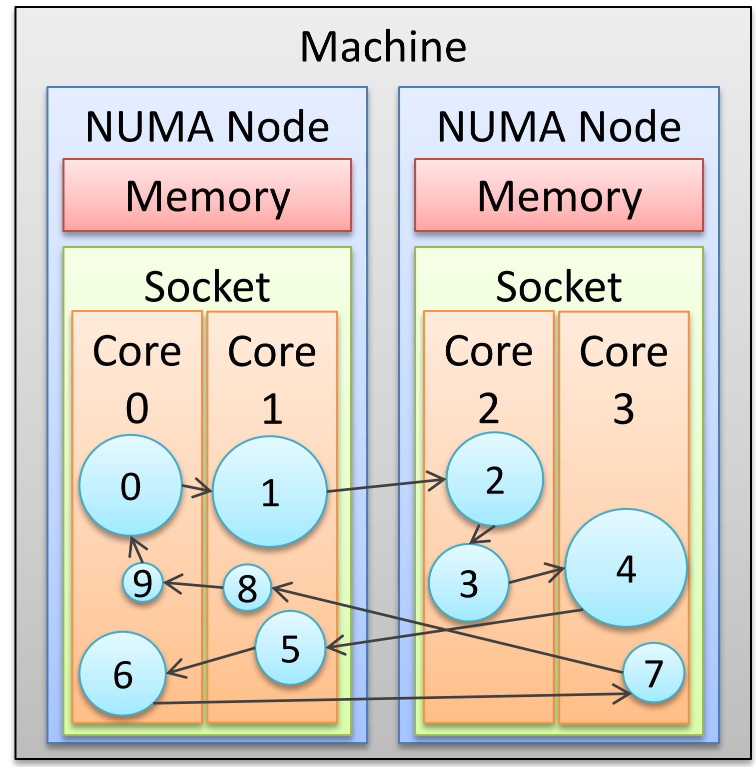


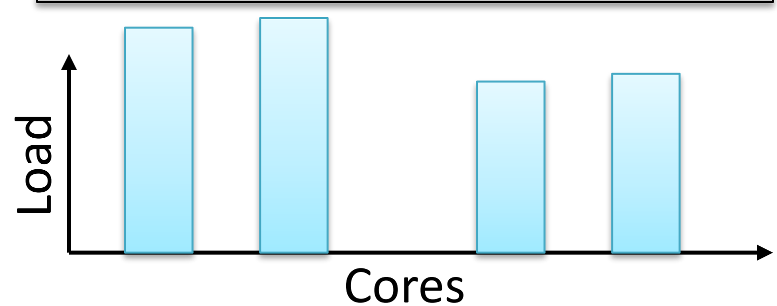
Load Balancing Approach

NucoLB

- **NucoLB: algorithm**
  - **Order tasks (chares) by load**
  - Evaluate the cost function for each core
  - Choose the core with the smallest cost

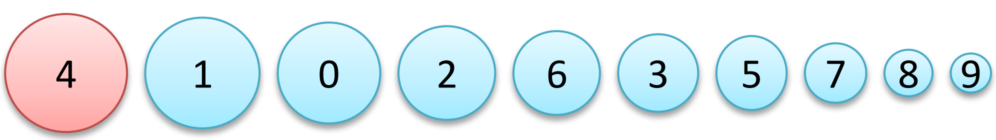


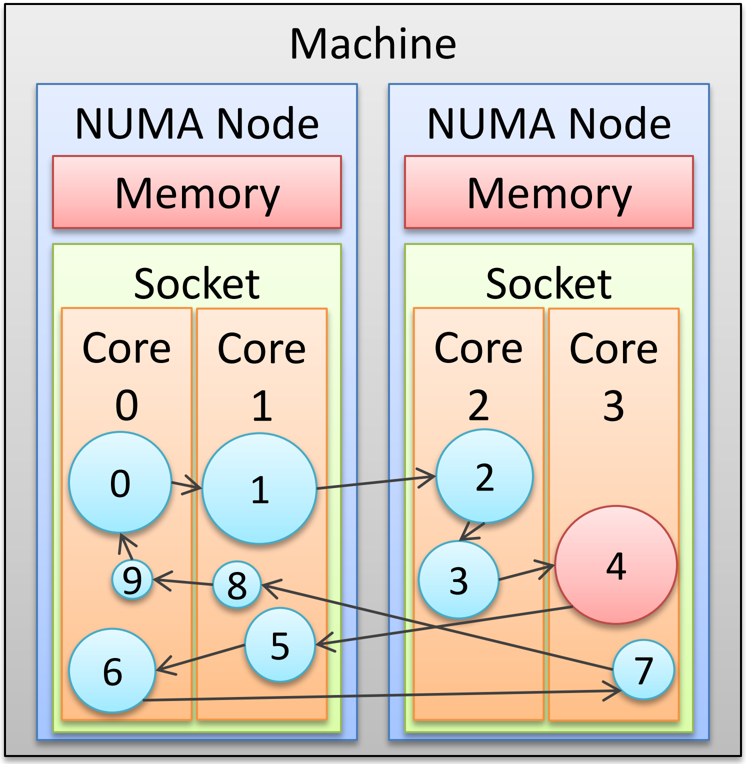


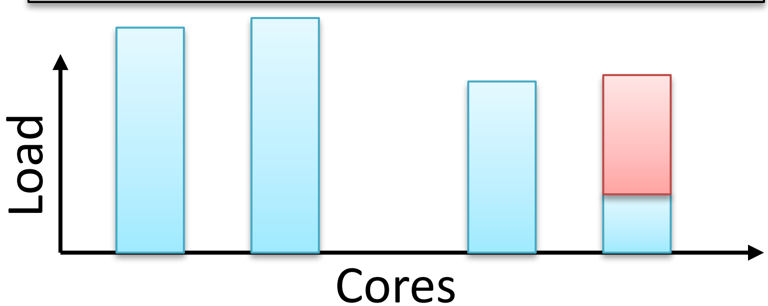


Load Balancing Approach

- **NucoLB: algorithm**
  - **Order tasks (chares) by load**
  - Evaluate the cost function for each core
  - Choose the core with the smallest cost



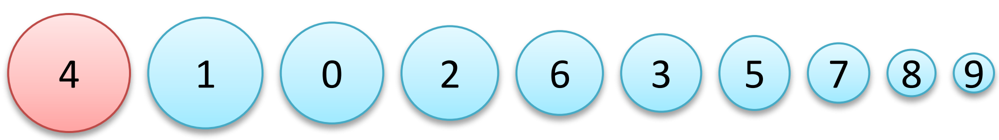


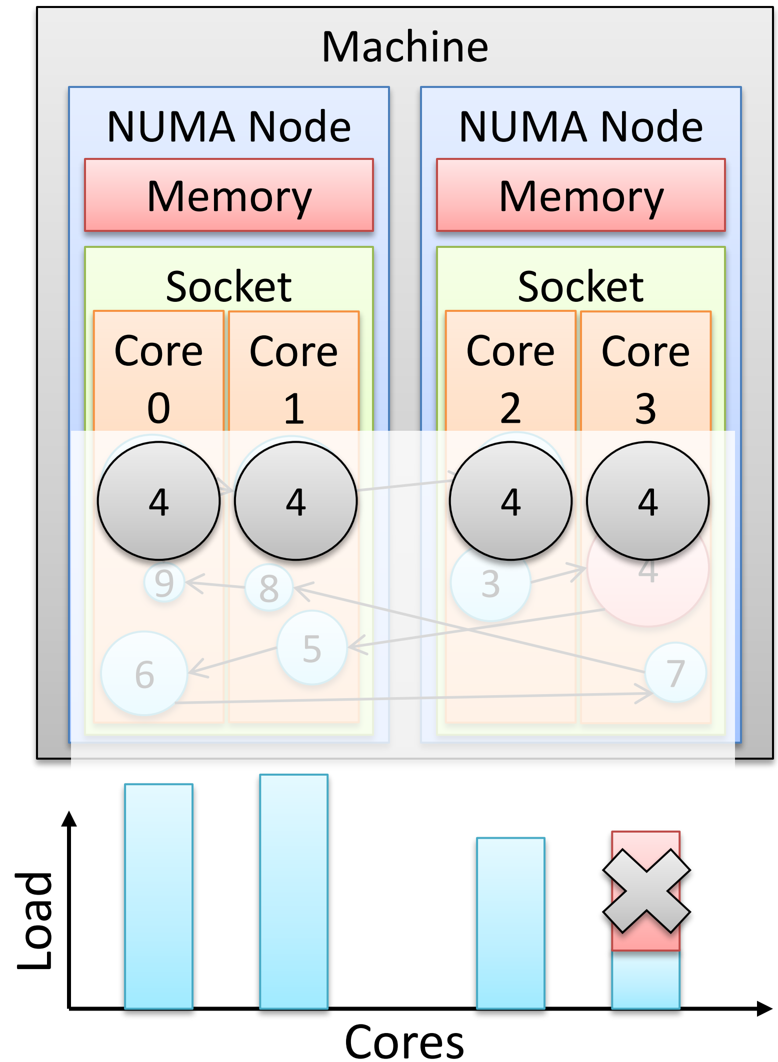


Load Balancing Approach

NucoLB

- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - **Evaluate the cost function for each core**
  - Choose the core with the smallest cost

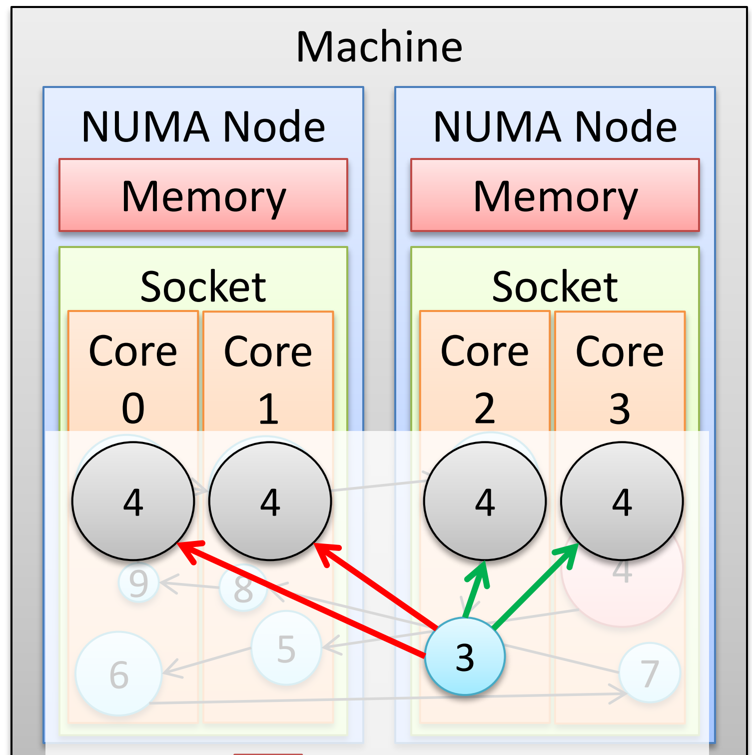


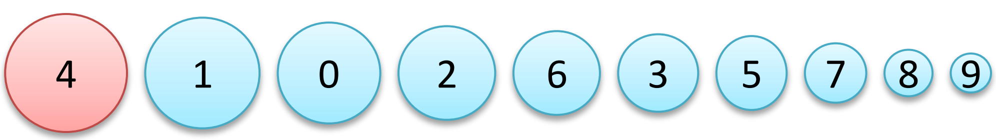


Load Balancing Approach

NucoLB

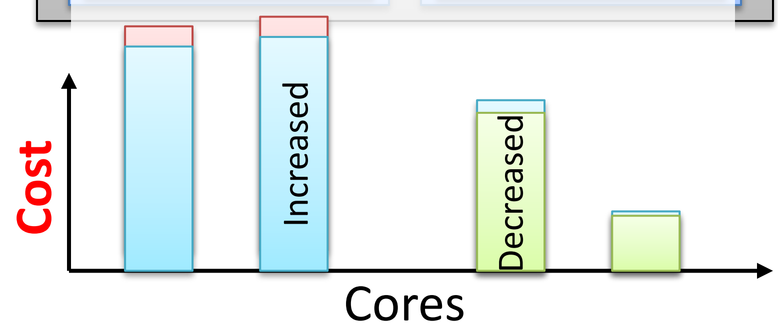
- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - **Evaluate the cost function for each core**
  - Choose the core with the smallest cost





Local communication (- #messages)

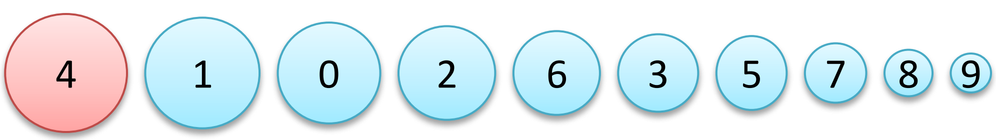
Remote communication (+ #messages*NUCO factor)

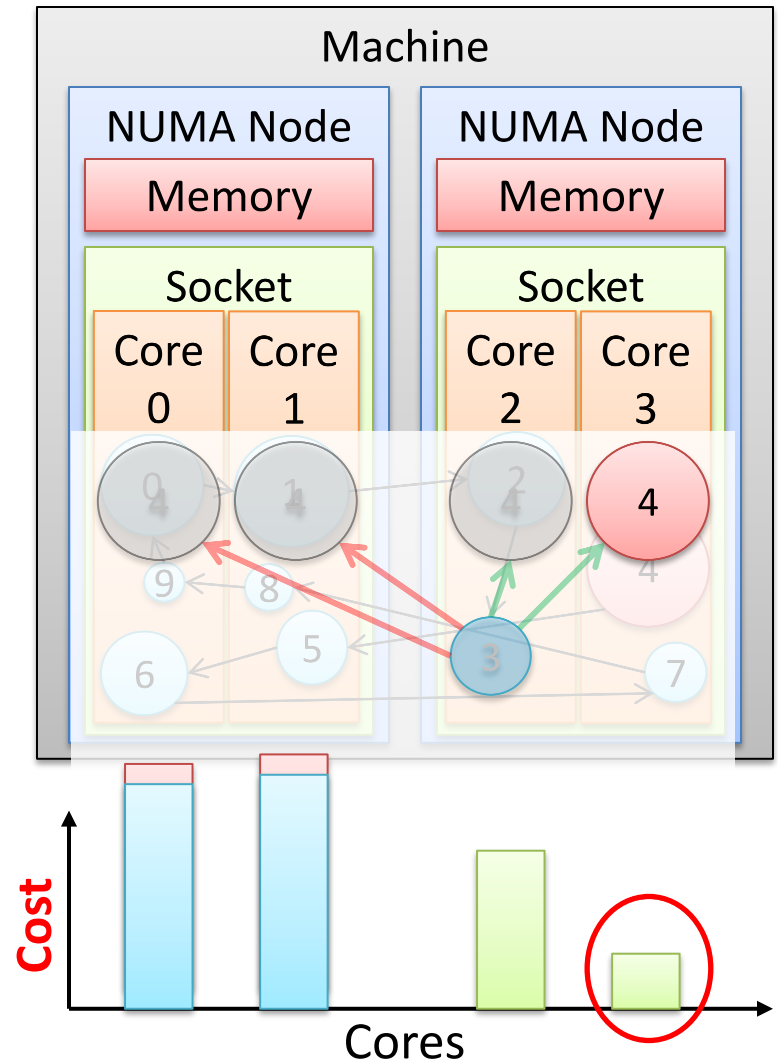


Load Balancing Approach

NucoLB

- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - Evaluate the cost function for each core
  - **Choose the core with the smallest cost**

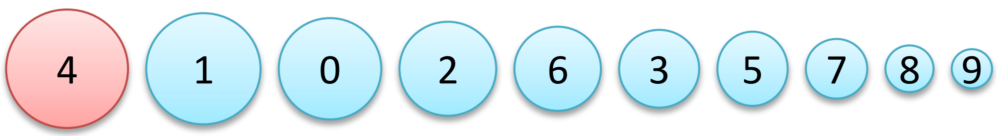


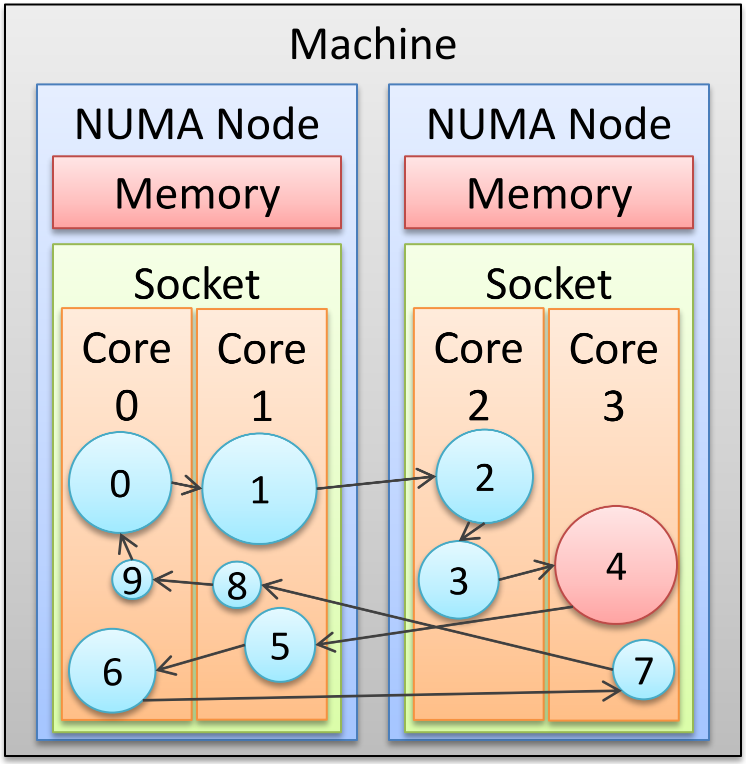


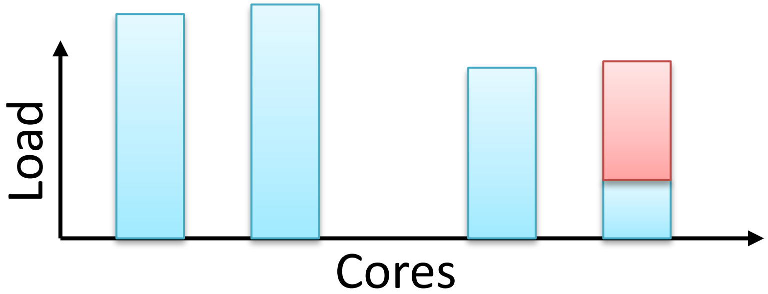
Load Balancing Approach

NucoLB

- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - Evaluate the cost function for each core
  - **Choose the core with the smallest cost**

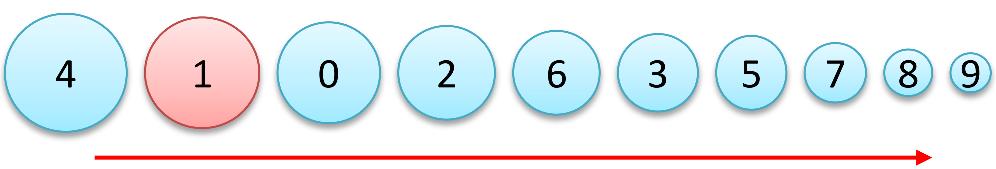


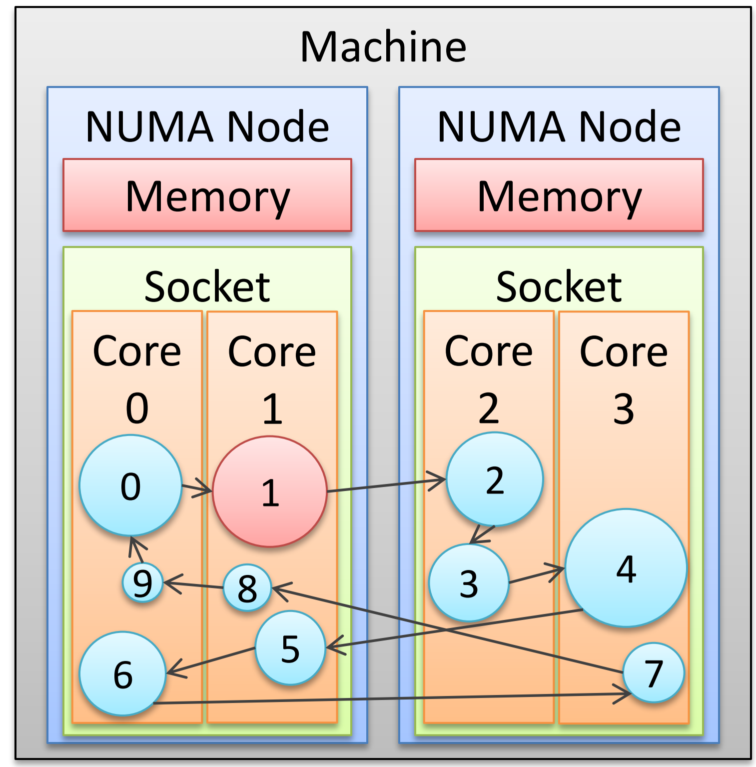


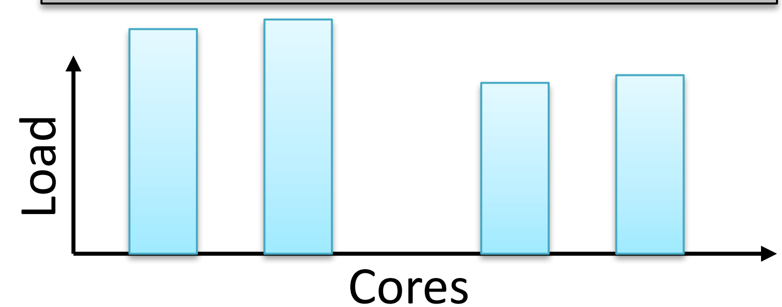


Load Balancing Approach

- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - Evaluate the cost function for each core
  - Choose the core with the smallest cost



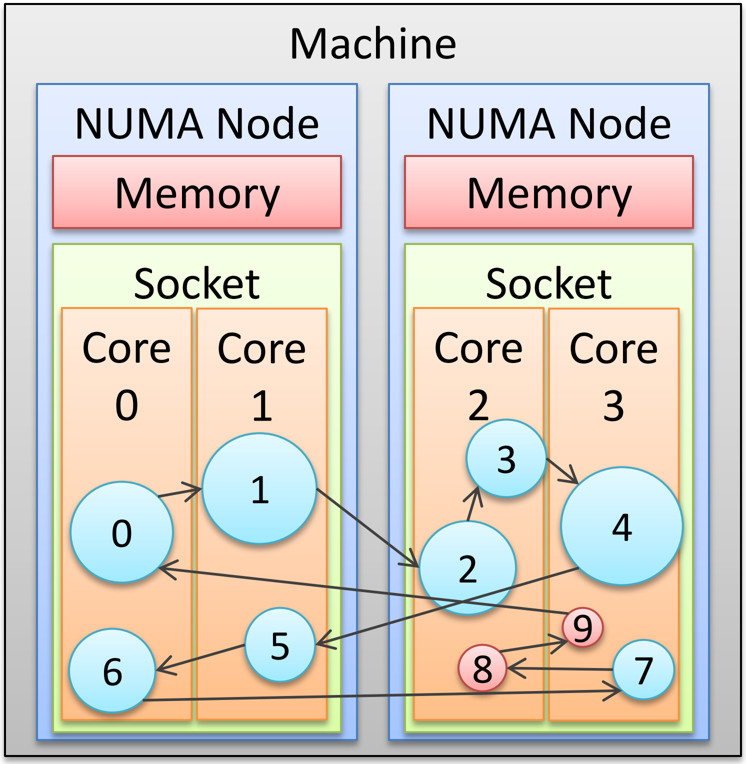




Load Balancing Approach

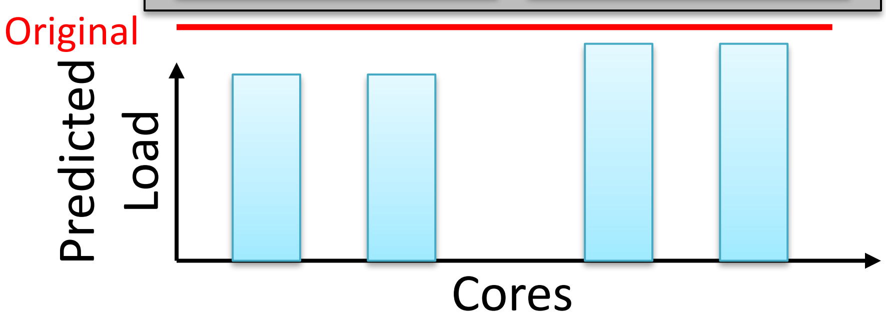
NucoLB

- **NucoLB: algorithm**
  - Order tasks (chares) by load
  - Evaluate the cost function for each core
  - Choose the core with the smallest cost



The new mapping may change the load of the tasks due to a reduction of communication costs

We usually have more parallelism to work with



Load Balancing Approach

- **TopoAwareLB**
  - Load balancer for **multi-core** machines
  - UMA and NUMA
  - Considers cache and memory **latencies**
  - **Proved asymptotically optimal**
  - 24% average improvement over other LBs with benchmarks

Load Balancing Approach

TopoAwareLB

- **TopoAwareLB: algorithm**
  - Choose most loaded core with probability α
    - Choose a random core otherwise
  - Choose heaviest task in the core with probability β
    - Choose a random task otherwise
  - Choose a new mapping according to a Gibbs distribution with temperature T

Agenda

- **Introduction**
  - Applications
  - Parallel Machines
  - Issues and Objectives
- **Load Balancing Approach**
  - General Idea
  - Required Information
  - NucoLB
  - TopoAwareLB
- **Performance Evaluation**
  - Platforms
  - Load balancers
  - Benchmarks and Application Results
- **Concluding Remarks**

Performance Evaluation

- **Platforms**
  - **NUMA32** (UJF-CEA)
    - 4 Intel Xeon processors, 8 cores each
    - 1 NUMA node per socket
  - **NUMA48** (UJF-CEA)
    - 4 AMD Opteron processors, 12 cores each
    - 2 NUMA nodes per socket
  - **20xNUMA4** (Grid'5000)
    - 2 AMD Opteron processors, 2 cores each
    - 20 machines on a **cluster**

Performance Evaluation

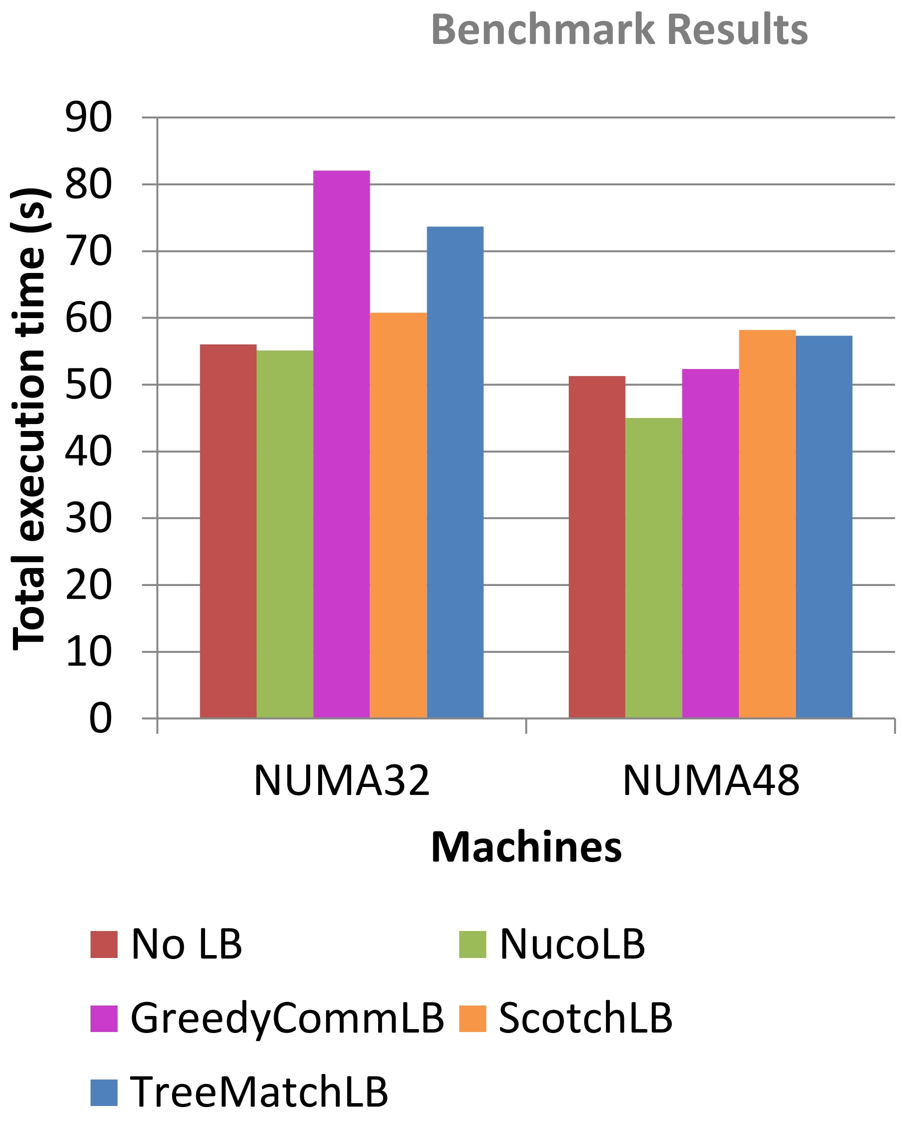
- **State-of-the-Art Load Balancers**
  - Used for performance comparisons
  - Available with Charm++
  - **GreedyCommLB**: greedy load balancer
  - **ScotchLB**: load balancer based on graph partitioning
  - **TreeMatchLB**: load balancer that maps the communication graph to the machine topology graph
  - All consider the load and communication behaviour of the tasks

Performance Evaluation

- 2 benchmarks
  - kNeighbor
  - stencil4d
- 1 application
  - LeanMD
- Average execution time
  - Minimum of 20 executions
  - 95% of statistical confidence
    - Student t-distribution
  - 5% relative error

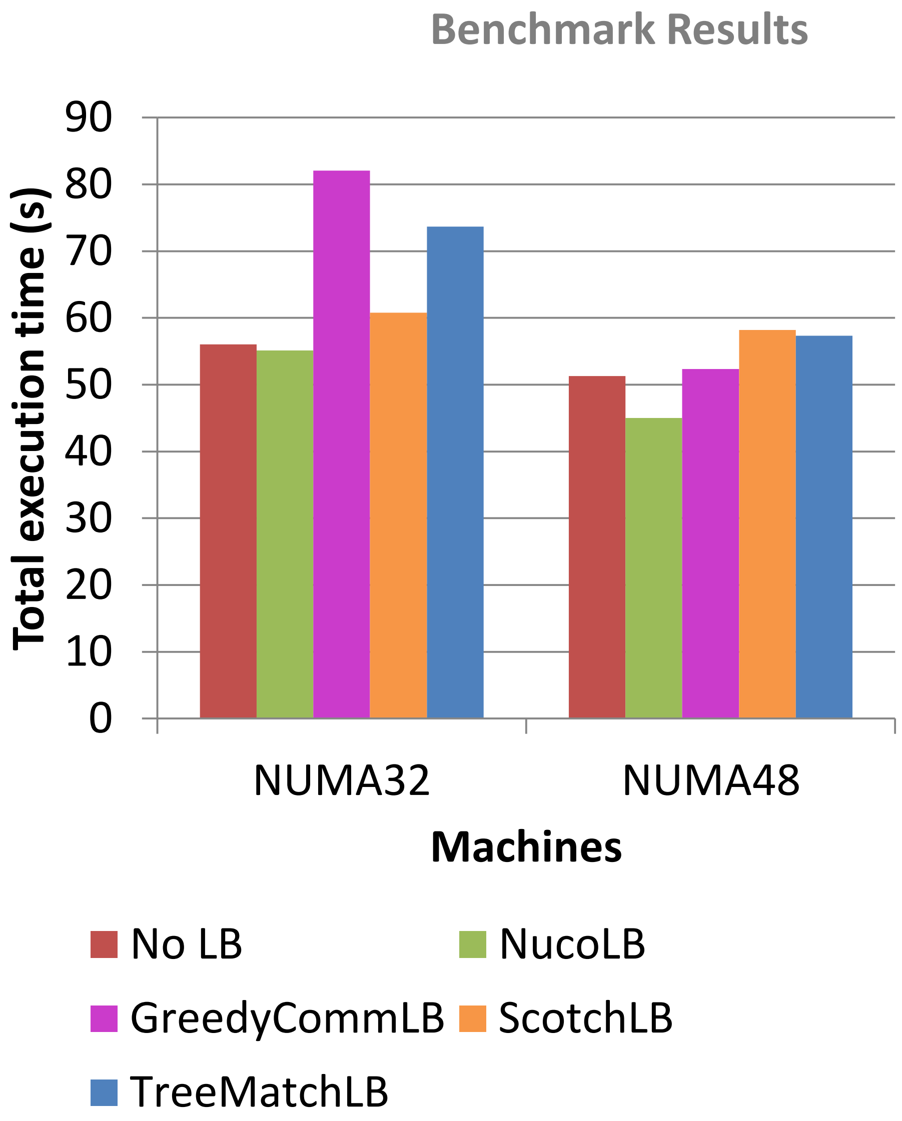
Performance Evaluation

- **kNeighbor**
  - Each task communicates with *k* other (*k*=7)
  - **Communication-bound**
  - 400 tasks, 8 KB messages, 50 iterations
  - **Regular**, load balanced
  - One load balancing call at each 10 iterations



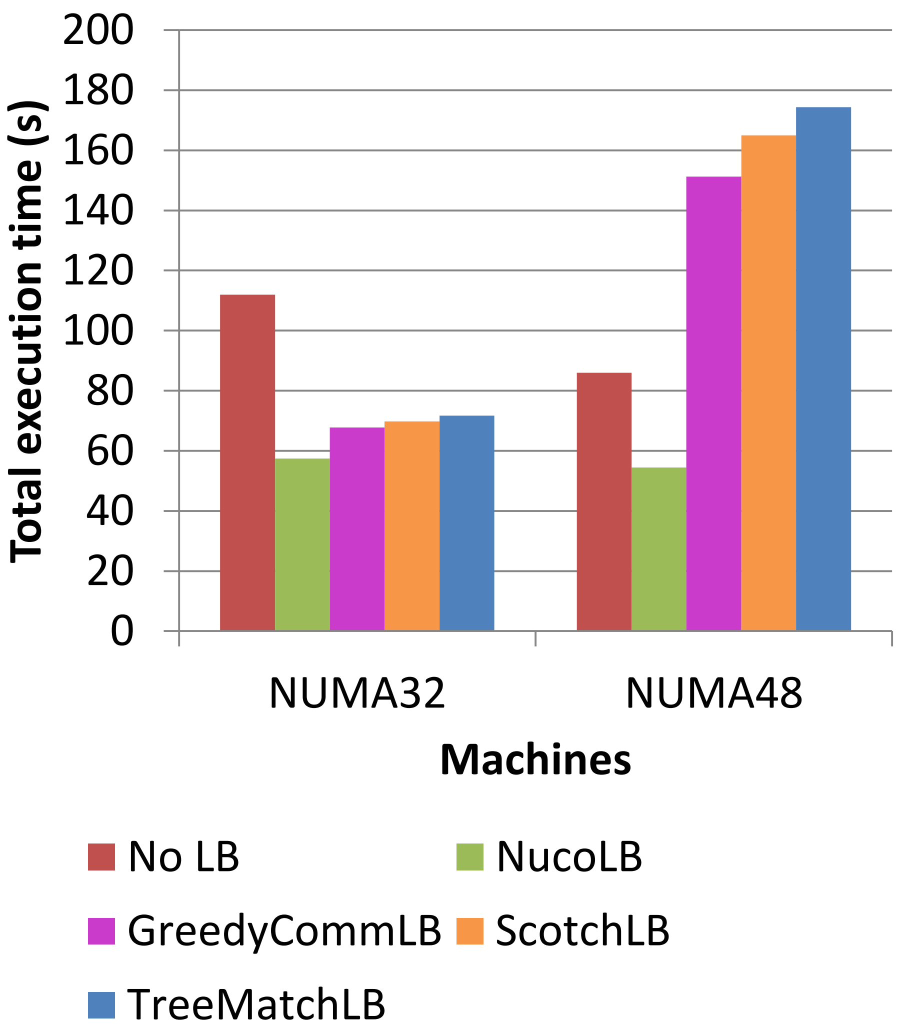
Performance Evaluation

- **kNeighbor**
  - **Low** load balancing **overhead**
    - 150 ms per LB call, other take 700 ms
  - Speedup of 1.14 over no LB on NUMA48
    - **Reduces the costs** of communication
    - **6% load reduction**



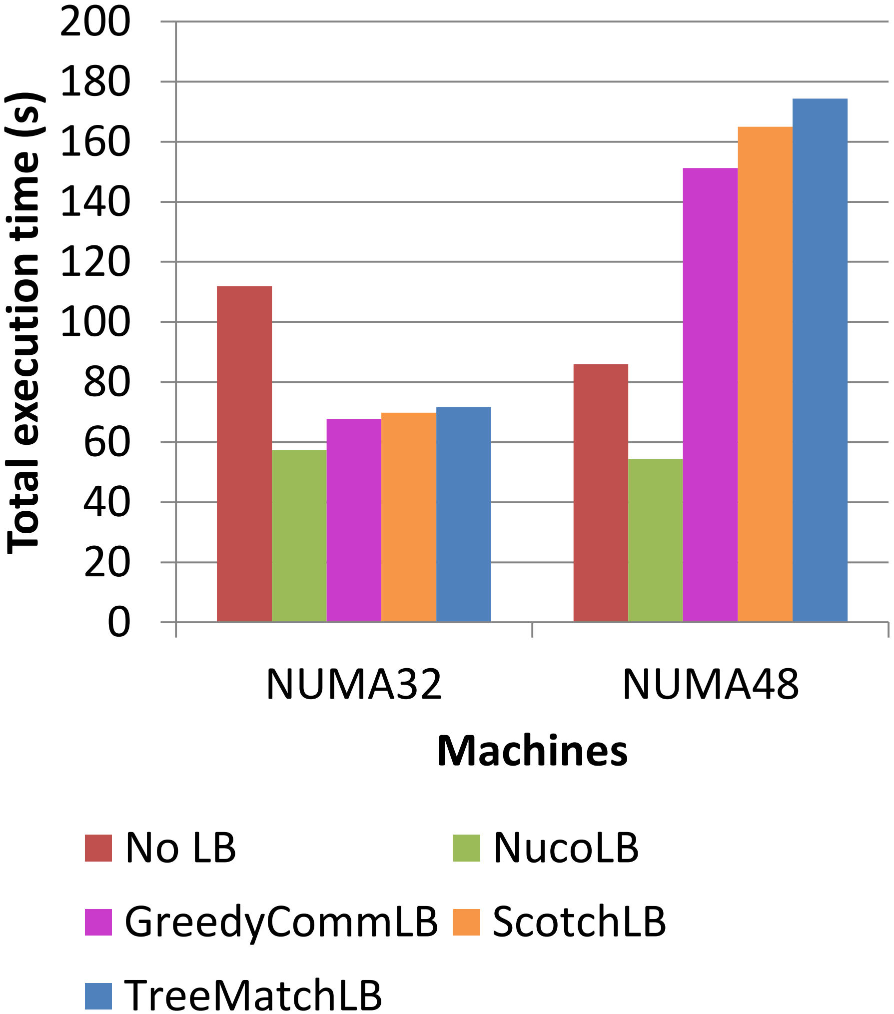
Performance Evaluation

Benchmark Results



- **stencil4d**
  - Four dimensional stencil
  - 256 tasks, 50 iterations
    - 4x4x4x4 stencil
  - **Compute-bound**
  - **Initially imbalanced**
  - **Large memory footprint**
  - One load balancing call at each 10 iterations

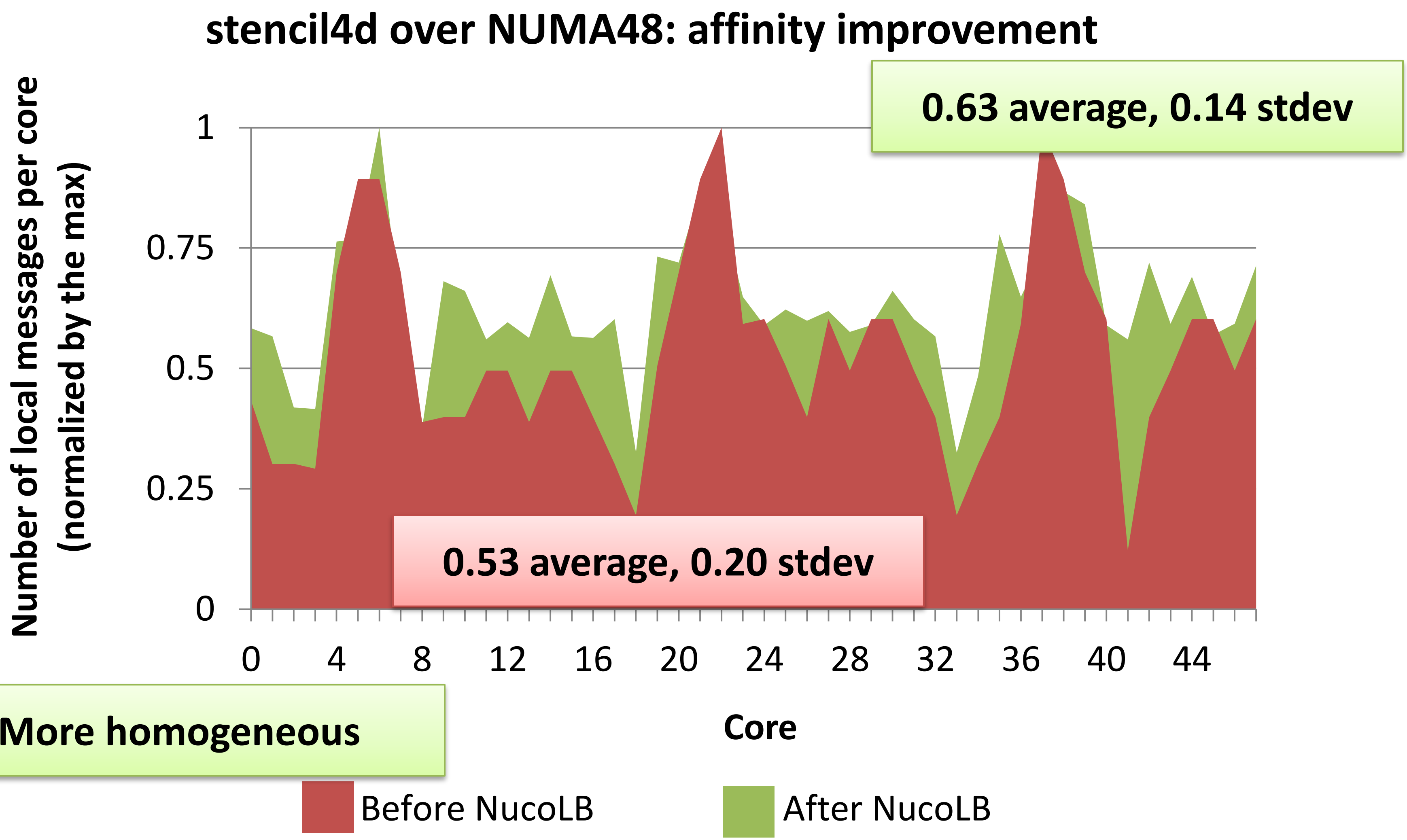
Performance Evaluation



- **stencil4d**
  - Speedup of 1.18 over the second best LB on NUMA32
  - Small LB overhead
    - **Does not increase cache misses and page faults**
    - Other LBs increase cache misses by 31% and page faults by 19%
  - Improves affinity

Performance Evaluation

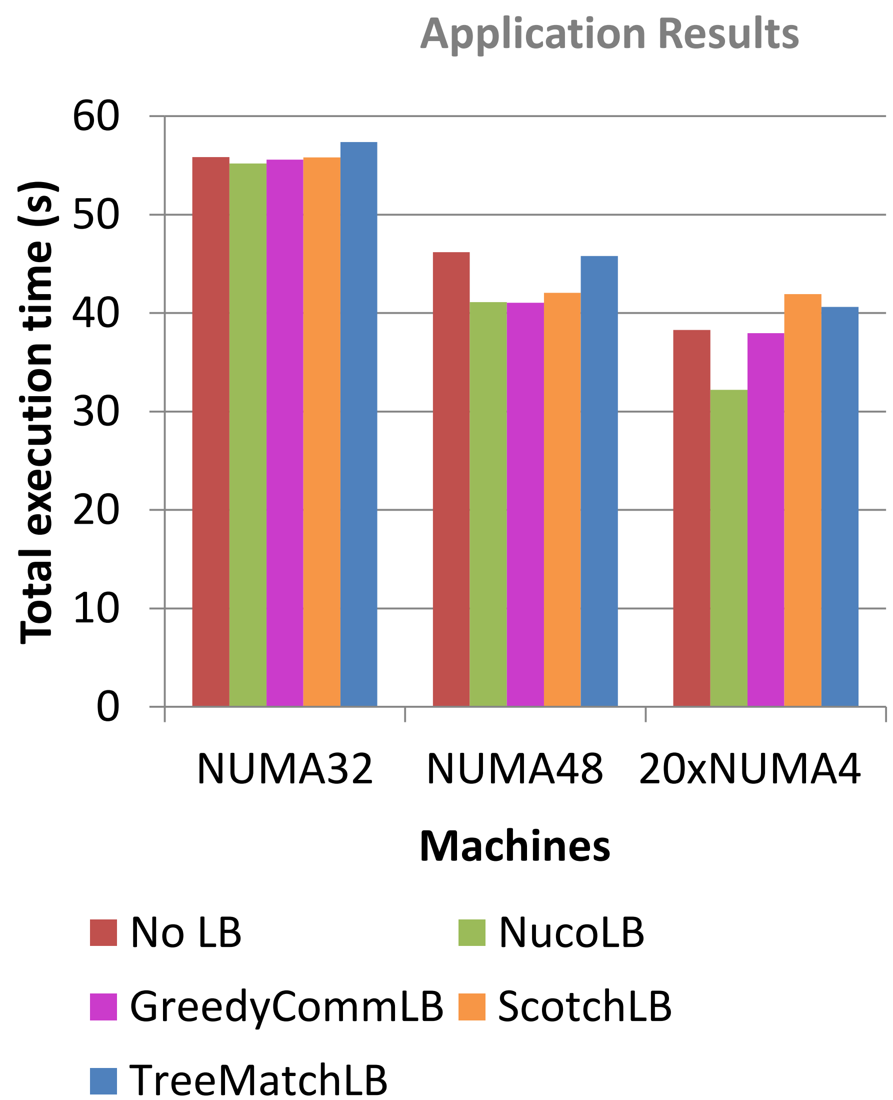
Benchmark Results



More homogeneous

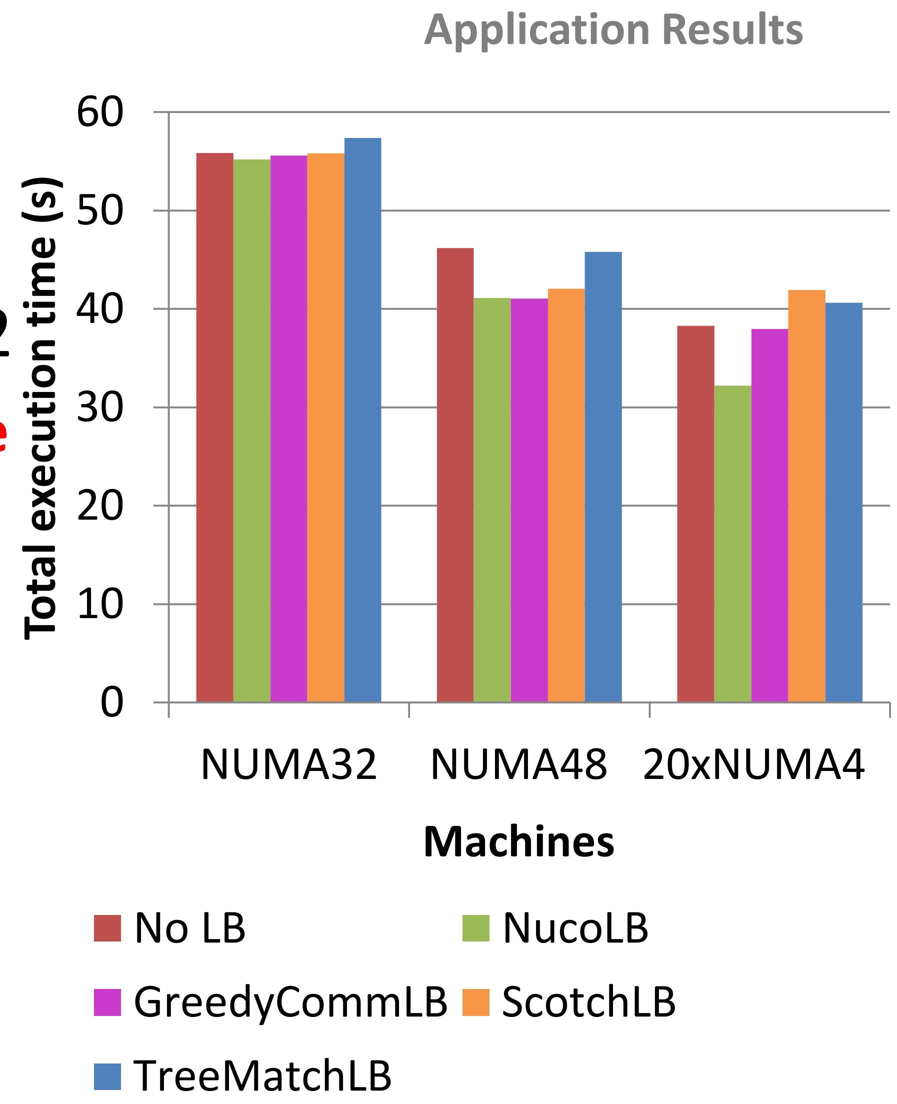
Performance Evaluation

- **LeanMD**
  - **Molecular dynamics application**
  - **Highly irregular**
  - Compute-bound
  - **1875 tasks**, 300 iterations
  - One load balancing call at each 60 iterations



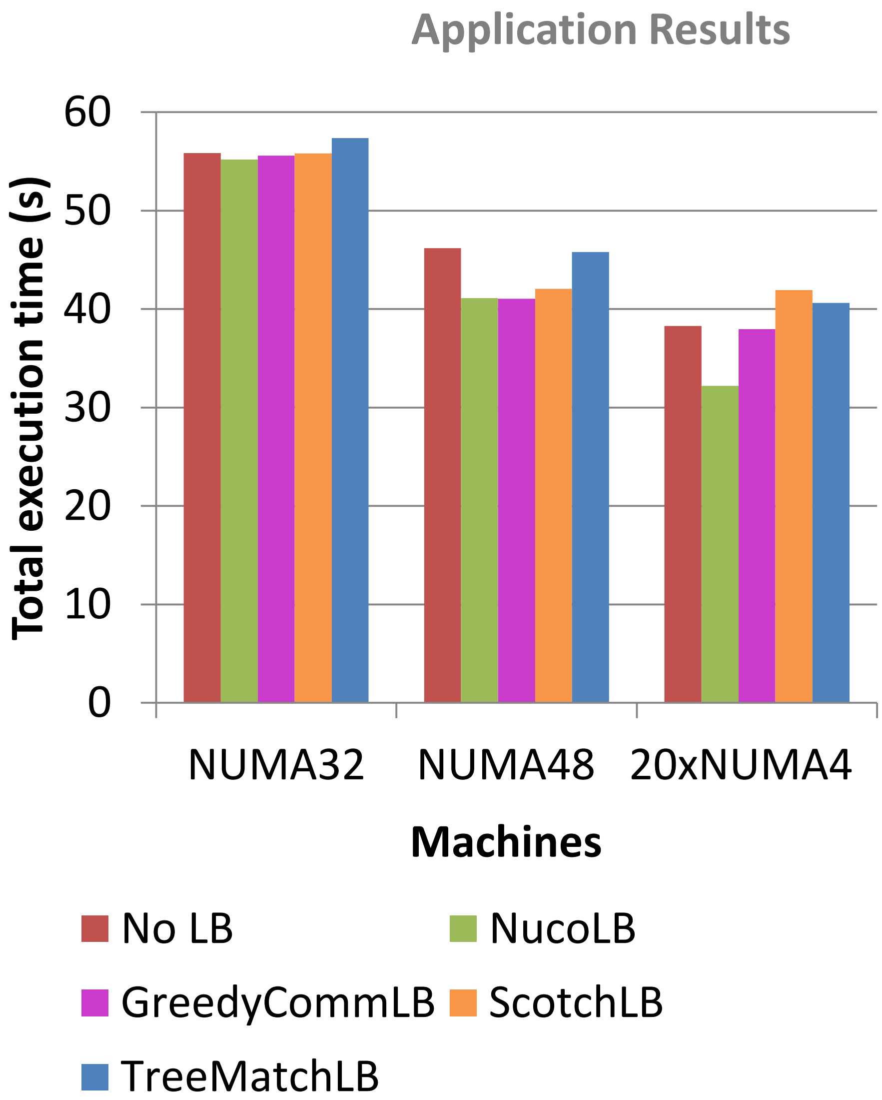
Performance Evaluation

- **LeanMD**
  - No changes on NUMA32
    - **95% core usage from the start**
    - Too many tasks per core
  - Speedup of 1.12 over the baseline on NUMA48
    - Negligible overhead to move tasks around



Performance Evaluation

- **LeanMD**
  - 19% reduction on total execution time on 20xNUMA4
    - **93% core usage**
    - **Iteration time reduction** from 114 ms to 90 ms
    - 11 to 18 times less migrations than other LBs
  - **Overloads some cores to reduce communication overhead**



Agenda

- **Introduction**
  - Applications
  - Parallel Machines
  - Issues and Objectives
- **Load Balancing Approach**
  - General Idea
  - Required Information
  - NucoLB
  - TopoAwareLB
- **Performance Evaluation**
  - Platforms
  - Load balancers
  - Benchmarks and Application Results
- **Concluding Remarks**

Concluding Remarks

Review

- **Context**
  - Handle **irregular applications** over non-uniform, **hierarchical architectures**
  - Reduce **communication costs** and **idleness**
- **Approach**
  - Combine information about the **application behaviour** and the **machine topology model** in **load balancing** algorithms
- **Results**
  - Low overheads, **affinity and efficiency** improvements
  - **Performance portability** over different environments

Concluding Remarks

Source Code

- **HieSchella Project**
  - *Hierarchical Scheduling for Large Scale Architectures*
  - Load balancers source code
  - https://forge.imag.fr/projects/hieschella/

Concluding Remarks

Publication

- Presentation on MMMM 2012, London
  - Memory Management for Many- and Multicore
- **Paper accepted on ICPP 2012**
  - A Hierarchical Approach for Load Balancing on Parallel Multi-core Systems
- Poster on ISC 2012

Concluding Remarks

- **Current and Future Work**
  - Improve/extend other load balancers
    - MultiCoreLB, TopoAwareLB
  - **Extend hwloc** to provide communication costs
    - hwloc: Portable Hardware Locality library
      - Open-MPI + Inria Bordeaux
    - Latency and bandwidth
    - Easier distribution

Concluding Remarks

- **Current and Future Work**
  - Consider additional information
    - **Bandwidth**
    - **Contention**
  - **Scalability**, distributed algorithm
  - Experiment on different platforms
    - **ARM processors**
  - Extend algorithms for **heterogeneous architectures**
    - GPUs, different processing powers

Load Balancing for Parallel Multi-core Machines with Non-Uniform Communication Costs

Laércio Lima Pilla

laercio.lima-pilla@imag.fr

Thank you.

LIG Laboratory – Inria – Grenoble University

Grenoble, France

Institute of Informatics – Federal University of Rio Grande do Sul

Porto Alegre, Brazil