A Performance Measurement Approach for Modeling Latency and Bandwidth for Load Balancing

Laércio Lima Pilla

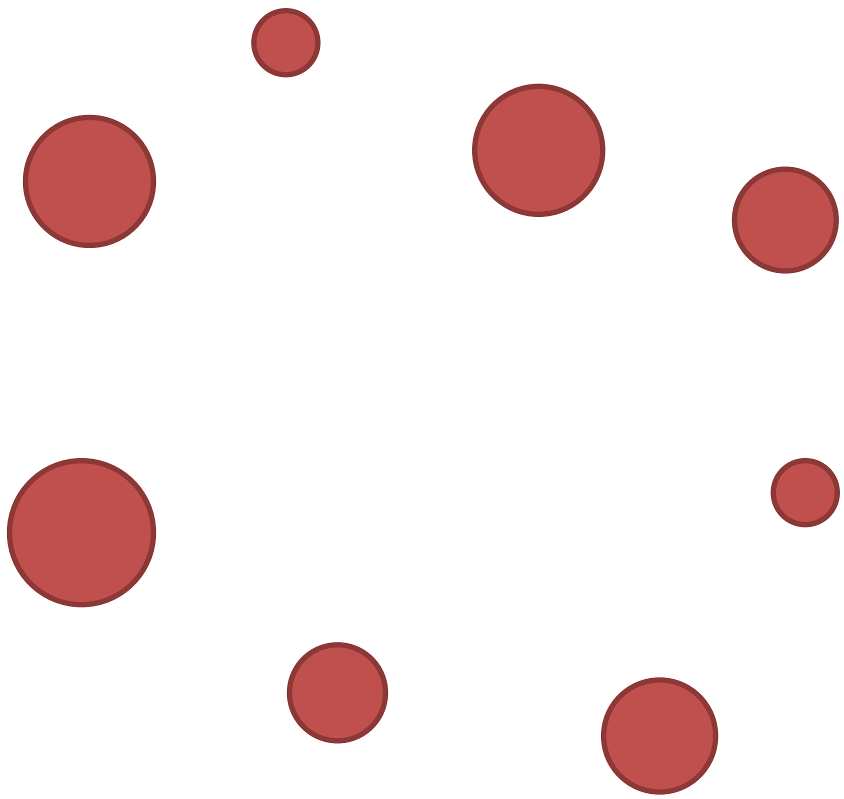
P.O.A. Navaux and J.F. Méhaut

What are we trying to solve here?

PROBLEM CHARACTERIZATION

Problem Characterization

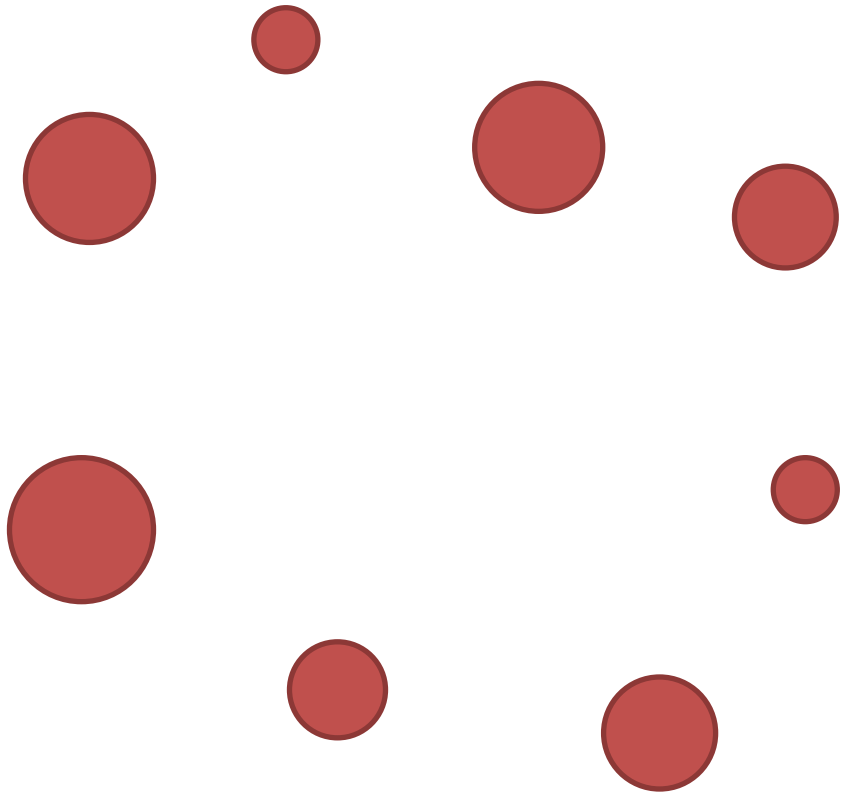
General idea



Problem Characterization

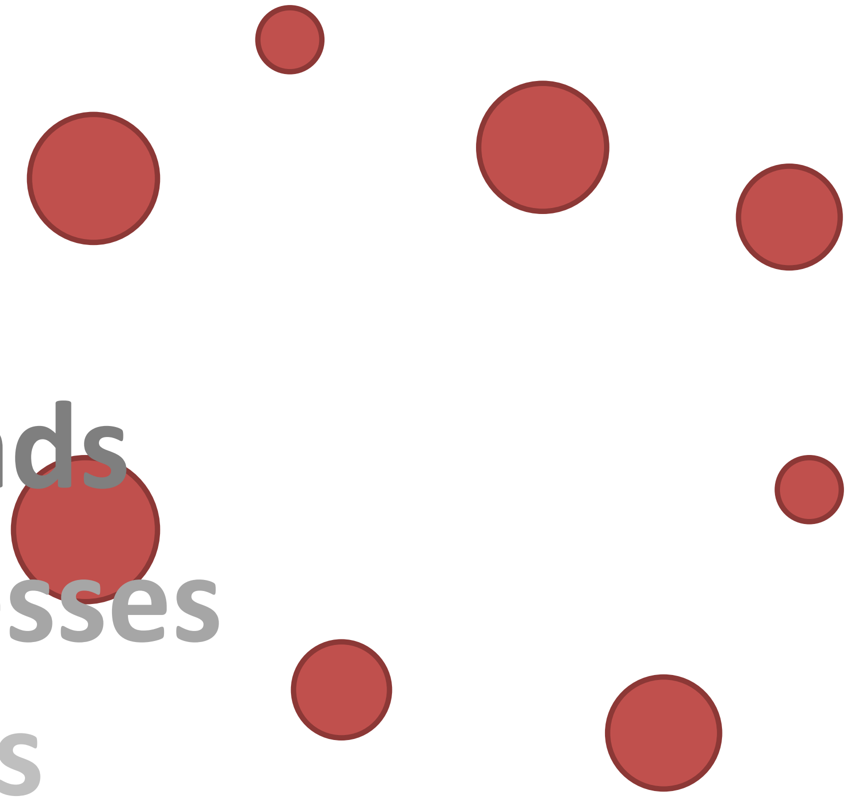
General idea

Tasks



Problem Characterization

General idea



Tasks

Threads

Processes

Actors

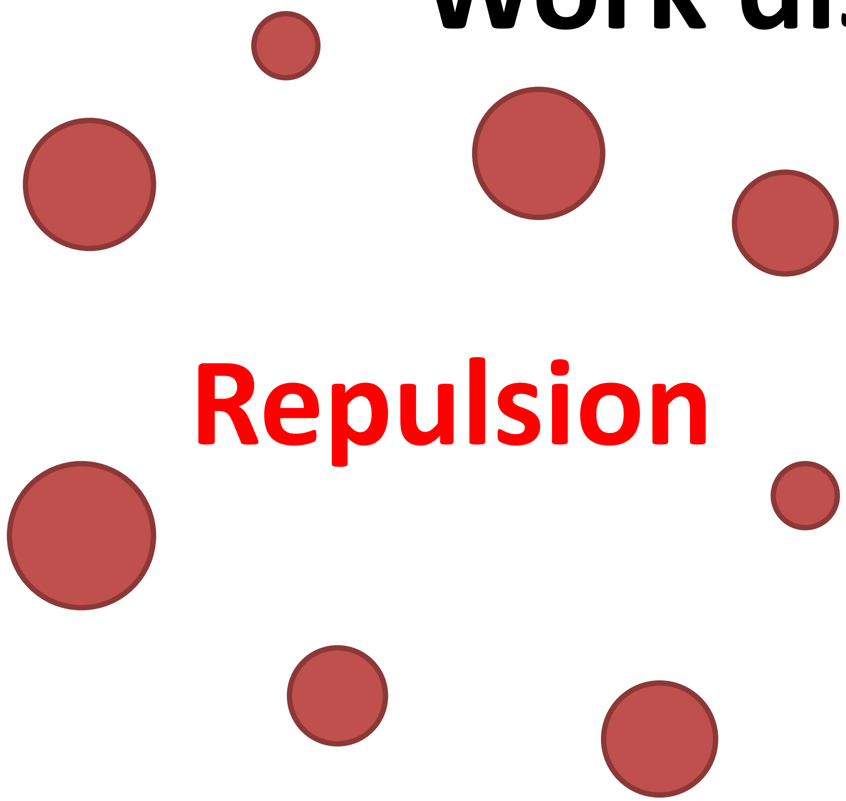
Objects

8th Workshop of the INRIA-ANL-Illinois Joint Laboratory on Petascale Computing, November 19-21, 2012

Problem Characterization

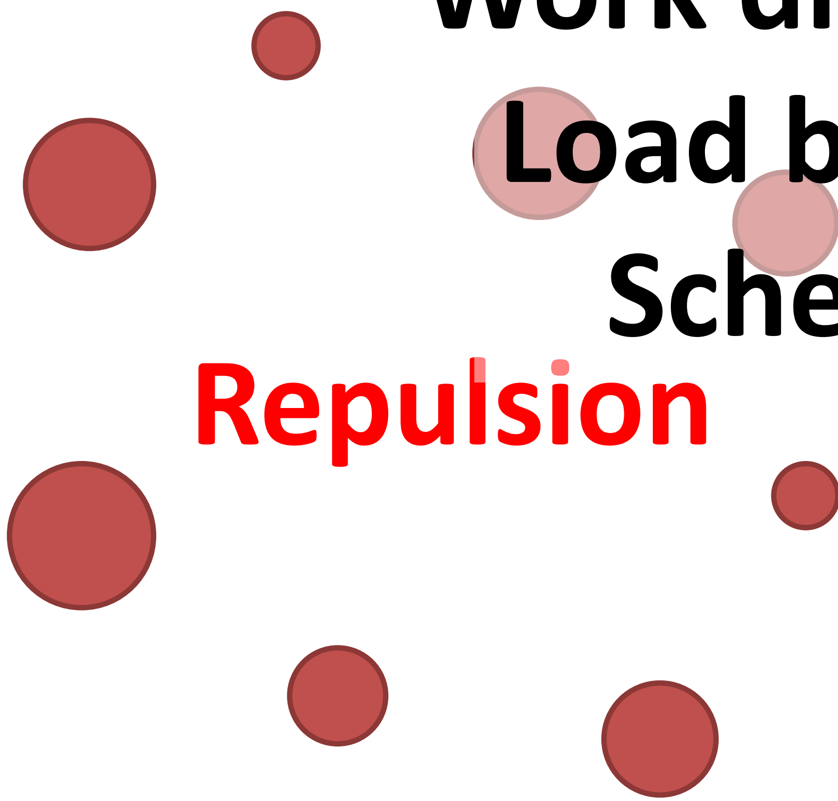
General idea

Work distribution



Problem Characterization

General idea



Work distribution

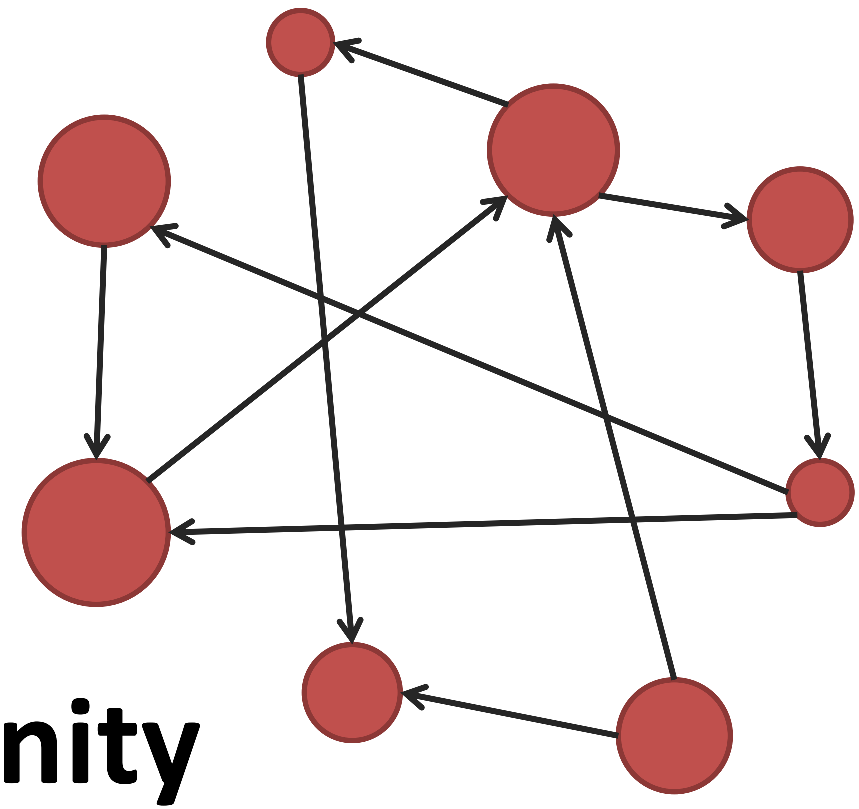
Load balancing

Scheduling

Repulsion

Problem Characterization

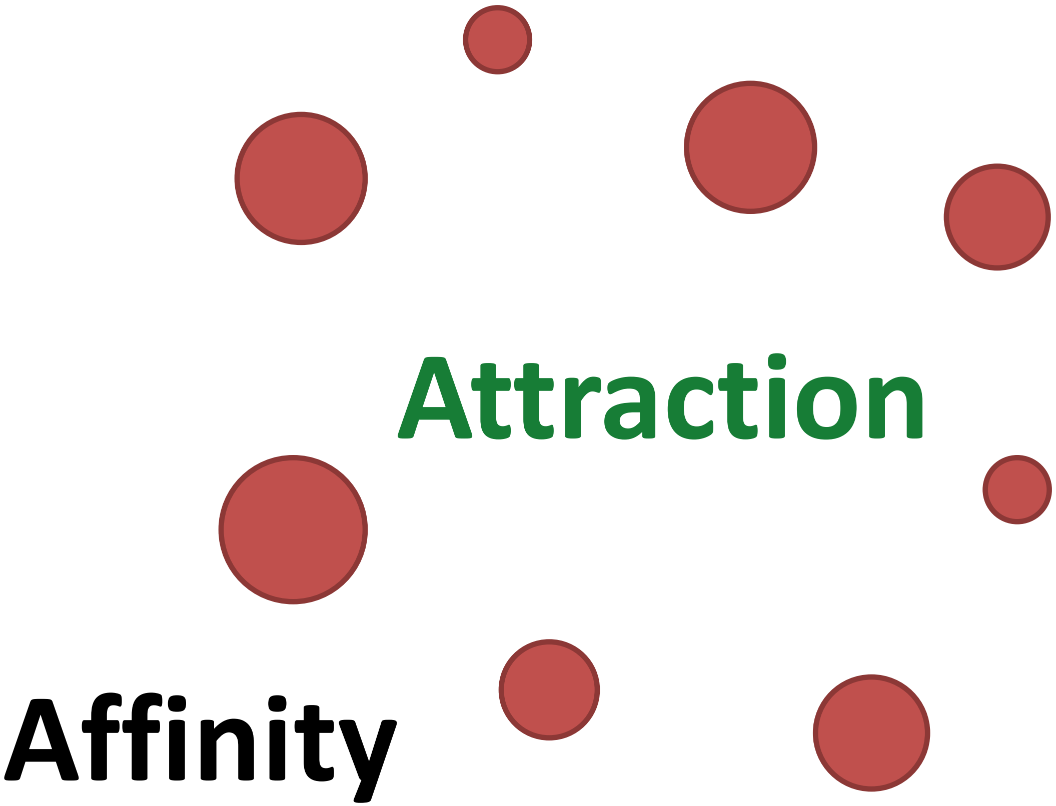
General idea



Affinity

Problem Characterization

General idea



Problem Characterization

General idea

Process mapping

Memory management

Attraction

Affinity

Problem Characterization

General Idea

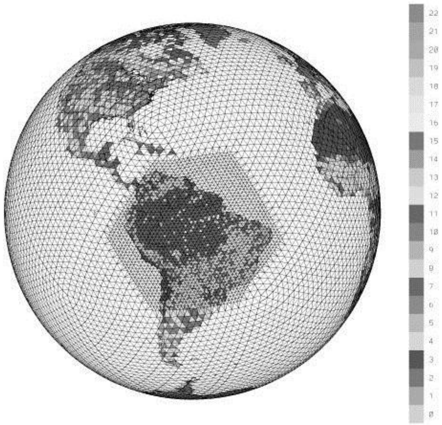
Where is the sweet spot for performance?

Problem Characterization

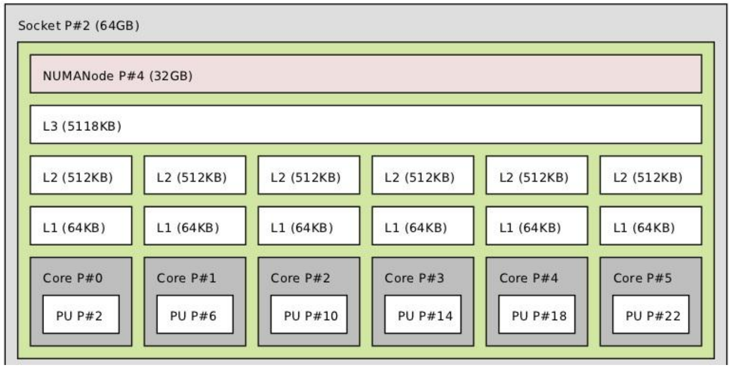
- **Objectives**
  - Improve performance
  - Optimize resource usage
    - **Reduce** processor **idleness**
    - **Reduce communication costs**
    - Find the best **trade-off**
  - **Performance portability**
    - Different platforms, different applications

Problem Characterization

- **Irregular Applications**
  - **Load imbalance**
  - **Complex communication** patterns
- **Hierarchical Architectures**
  - **Memory** hierarchy
  - **Network** hierarchy
  - **Asymmetric** communication **costs**



Climatology



How can we handle this performance dilemma?

APPROACH

Approach

- **Load balancing**
  - Combine **application information** with a **machine topology model**

Approach

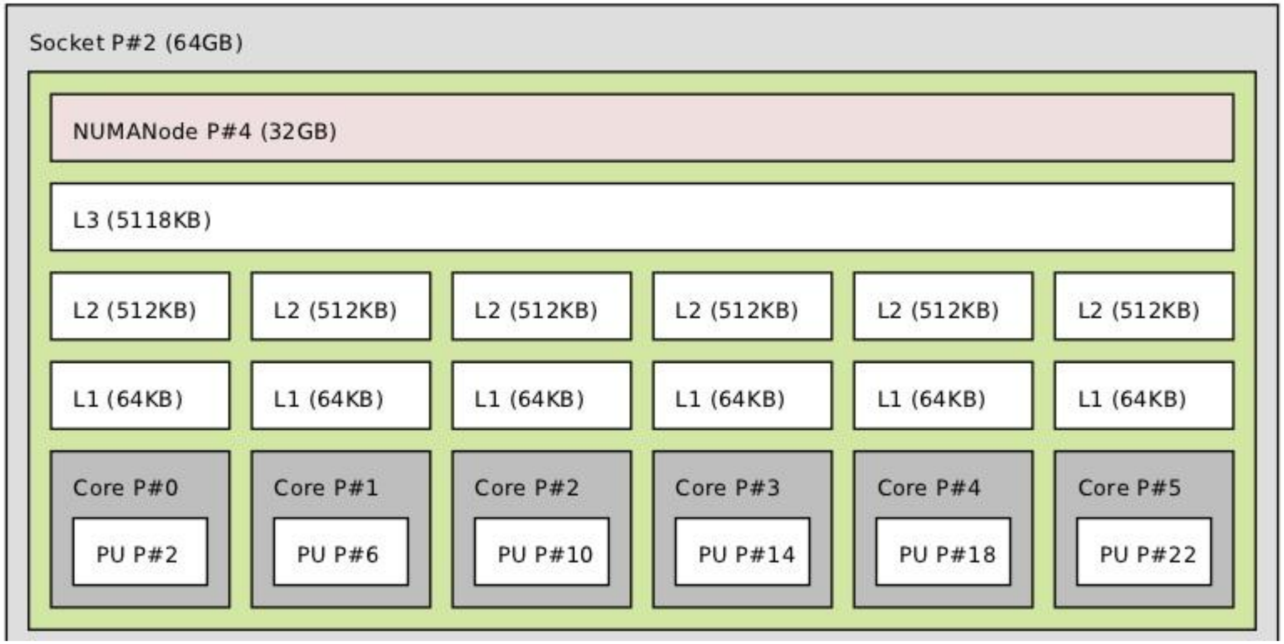
- **Application information**
  - **Execution time** of tasks (load)
  - **Communication graph**
  - Current task mapping

Approach

- **Machine topology model**
  - Topology (component sharing)
  - Actual distances between components
    - **Latency**
      - Time to start moving data
    - **Bandwidth**
      - Time moving data around
  - Obtained in feasible time

Approach

- **Machine topology model**
  - Topology (component sharing)
  - **Benchmarked communication costs**



Approach

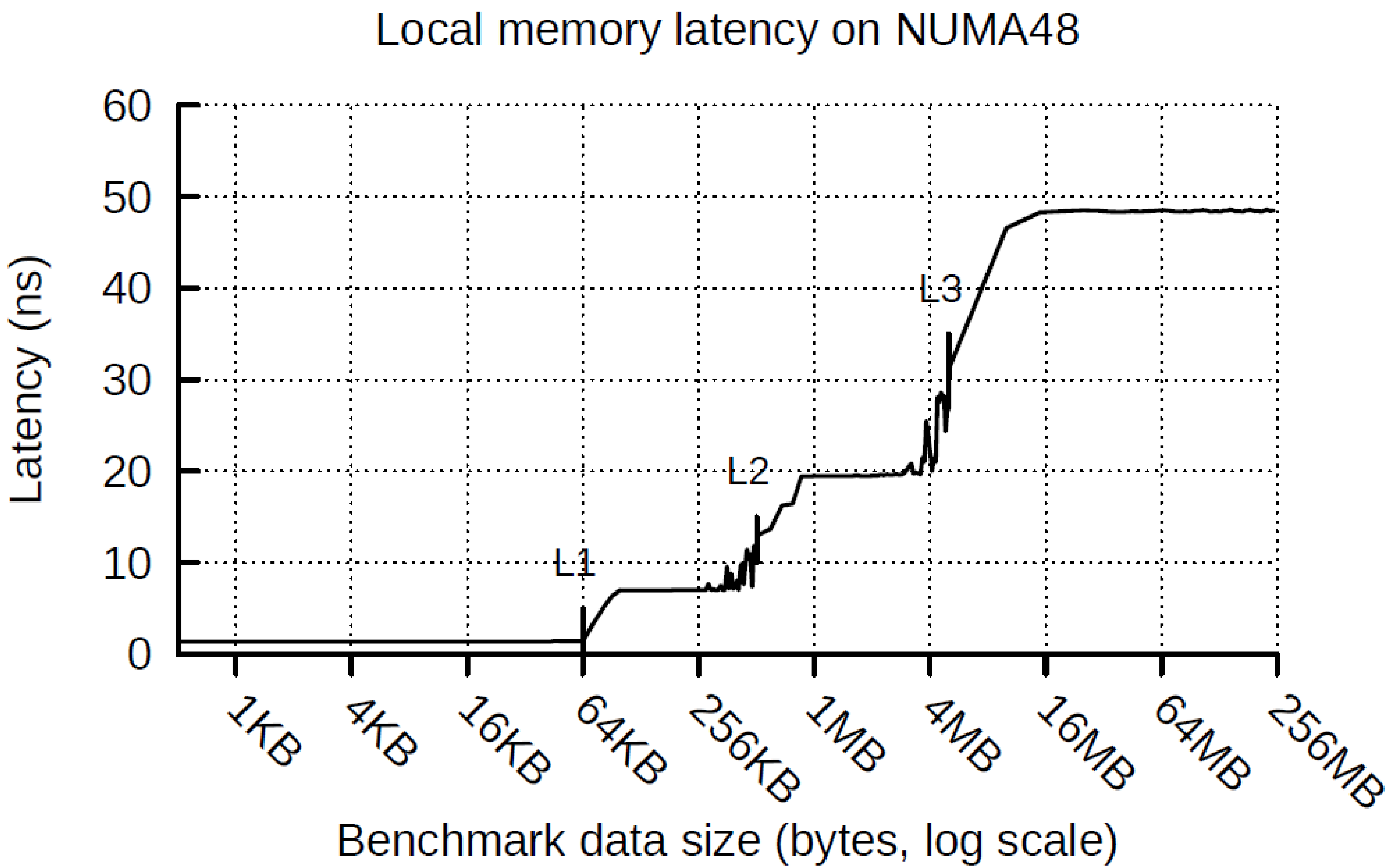
- **Benchmarked information**
  - **Memory**
    - **Latency**: lat_mem_rd (LMbench)
    - **Bandwidth**: bw_mem (LMbench)
  - **Network**
    - **Latency and bw**: MPI ping-pong (coNCePTuaL) + linear regression

Approach

- **hwloc**: Portable Hardware Locality
  - Machine topology
  - http://www.open-mpi.org/projects/hwloc/
- **HieSchella project**: extended model
  - Benchmark the memory hierarchy
  - https://forge.imag.fr/projects/hieschella/

Approach

Topology benchmarking



How do we glue those things together?

LOAD BALANCERS

Load Balancers

- **Charm++**
  - UIUC
  - Parallel programming language
  - **Load balancing framework**
  - http://charm.cs.uiuc.edu/

Load Balancers

- NucoLB
  - **Clusters** composed of NUMA nodes
  - NUCO factor
- **HwTopoLB**
  - Multicore machines
  - **Proved asymptotically optimal**

Load Balancers

- **HwTopoLB**
  - **Asymptotically optimal** algorithm
    - Choose most loaded core with probability α
    - Choose heaviest task with probability β
    - Choose a mapping according to a Gibbs distribution over the set of predicted makespans

The Gibbs distribution with temperature $T > 0$ over the set of real values $v_1 \ldots v_n$ is the probability vector on $\{1 \ldots n\}$:

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

Load Balancers

- **HwTopoLB**
  - Predicted makespans
    - Compute the load of all cores for each mapping
    - Take the **slowest core**
  - Tasks' loads change depending where their neighbors are
    - **Different latencies and bandwidths**
  - Communication cost
    - **#messages*latency + #bytes/bandwidth**
    - Depend on the first shared level of the topology

Load Balancers

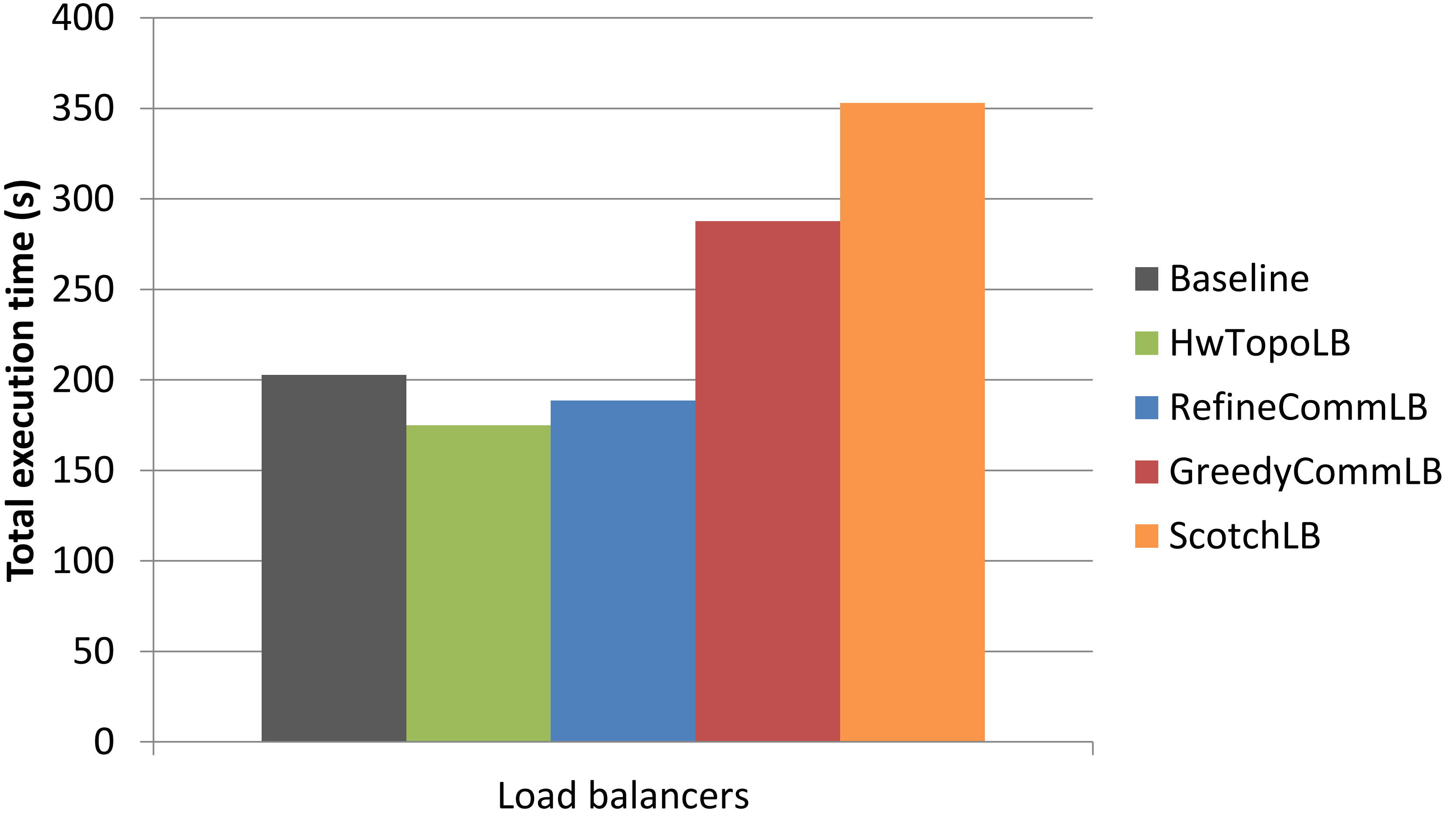
- **HwTopoLB**
  - Performance improvement of **24% in average** over other load balancers
  - *Asymptotically Optimal Load Balancing for Hierarchical Multi-Core Systems. To be published on ICPADS 2012.*
  - Working on an extend journal version

Load Balancers

- **Performance example**
  - **Initial results** on a **cluster**
  - **LeanMD** on 3 Cray XE6 nodes
    - Charm++ v6.4.0 mpi-crayxt-smt
    - 31 processing threads, 1 communication thread
    - 3024 computes
    - Cell array dimension: 6x6x6 of size 16x16x16
    - 1000 iterations, 10 load balancing calls
    - 20 runs

Performance Example

Initial results in a cluster



What can we take from this?

CLOSING

Closing

- Balance work distribution and affinity
- **Reduce idleness and comm. costs**
  - Irregular applications and hierarchical machines

Closing

- Balance work distribution and affinity
- **Reduce idleness and comm. costs**
  - Irregular applications and hierarchical machines
- **Load balancing**
  - Combine **application information** with a **machine topology model**

Closing

- **Future work**
  - Improve **network modeling**
  - Evaluate performance on clusters
- **Collaboration ideas**
  - **Charm++ with hwloc**
  - Charm++ over low power proc. (ARM)
  - Hardware counters information for LB
  - Distributed LB algorithms

A Performance Measurement Approach for Modeling Latency and Bandwidth for Load Balancing

Laércio Lima Pilla

laercio.pilla@inf.ufrgs.br

pilla@imag.fr

P.O.A. Navaux and J.F. Méhaut