Load balacing and affinities between processes with TreeMatch in Charm++ : preliminary results and prospects

The seventh workshop of the Joint Laboratory for Petascale Computing, Rennes

Emmanuel Jeannot Guillaume Mercier François Tessier
In collaboration with the Charm++ Team from the PPL : Sanjay Kale, Esteban Meneses-Rojas, Pritish Jetley and Gengbin Zheng

June 15, 2012

Introduction

State of Art

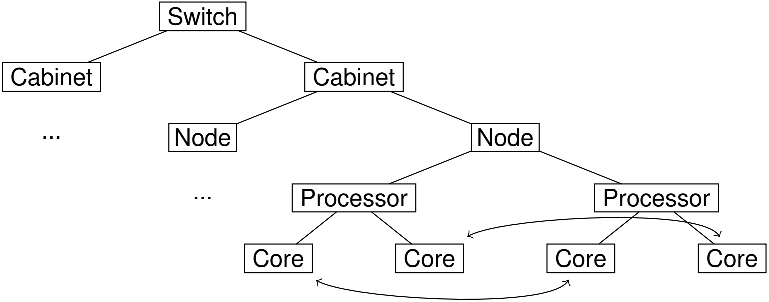
- Multi-node and multi-core architectures : Message passing paradigm
- Load balancing according to a flat topology

Problems

- Topology is not flat!
- Add the notion of processes affinity?
- Take into account the communication between processes?

Why we should consider it

- Plenty of current and future parallel platforms have several levels of hierarchy
- Application processes don't exchange the same amount of data (affinity)
- The process placement policy may have an impact on performance
  - Cache hierarchy, memory bus, high-performance network...



Problems

Given...

- ... The parallel machine topology
- The application communication pattern

Map application processes to physical resources (cores) to reduce the communication cost.

The TreeMatch Algorithm

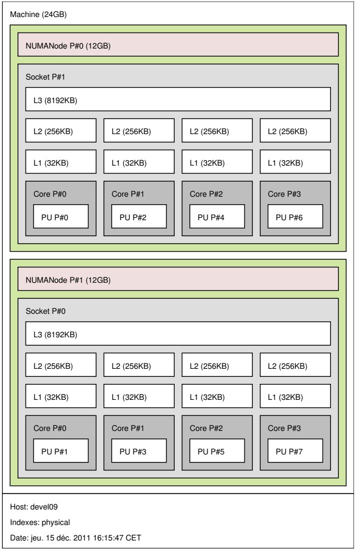
- Algorithm and tool to perform processes placement based on processes affinities and NUMA topology
- Given a process *i* of $\{1...p\}$, p the number of processes, and a topology tree composed of *n* leaves (cores) where $n \geq p$, try to find a permutation σ of $\{1...p\}$ such that σ_i is the core on which the process *i* has to be mapped to reduce communication cost.

Communication pattern

- Given as a *p x p* communication matrix (where *p* is the number of processes)
- Metrics : Amount of data, number of messages, average
- For MPI :
  - Need to modify the MPI implementation to monitor communication
- For Charm++ :
  - Communications between objects are natively monitored

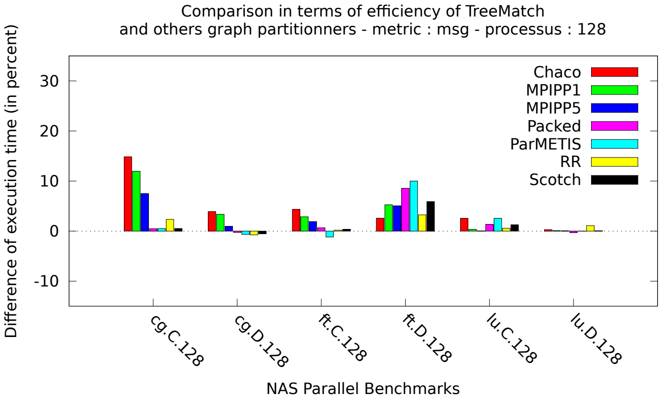
Topology

- Hwloc : library mainly developped at Inria
- Can provide us the topology (tools, C library)
- Portable abstraction, across OS, versions, architectures, ...
- Modern architectures (NUMA, cores, caches, ...)
- Can bind processes and threads to CPUs



NAS Parallel Benchmarks

- Static placement : Monitored execution → TreeMatch → affinity-aware execution
- CG (irregular memory access and communication), FT (all-to-all communication), LU (irregular communication)



What about charm++?

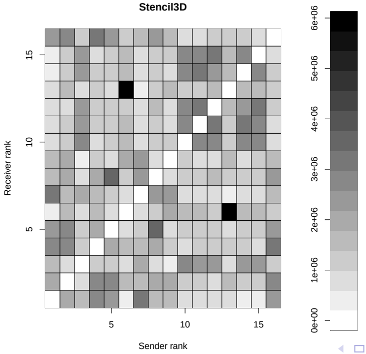
Not so easy...

- Several issues raised!
- Scalability of TreeMatch
- Need to find a better compromise between processes affinities and load balancing
- Impact of migration time?

The next slides will present what we tried, the encountered problems and what we plan to do to get around them.

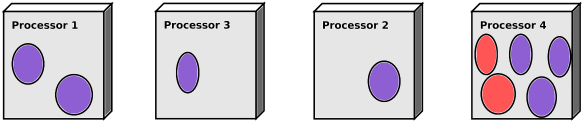
First Strategy

- GreedyLB (or any other LB) to perform load balancing
- Create a communication matrix of groups of chares (communicating objects responsible for performing some task) on each processor
- Run TreeMatch on this pattern and the corresponding topology
- Remap each group of chares on processors



First Strategy

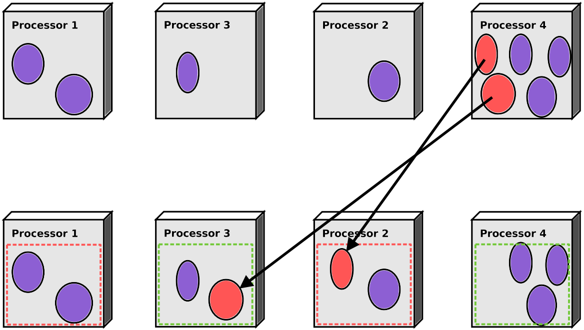
Initial state



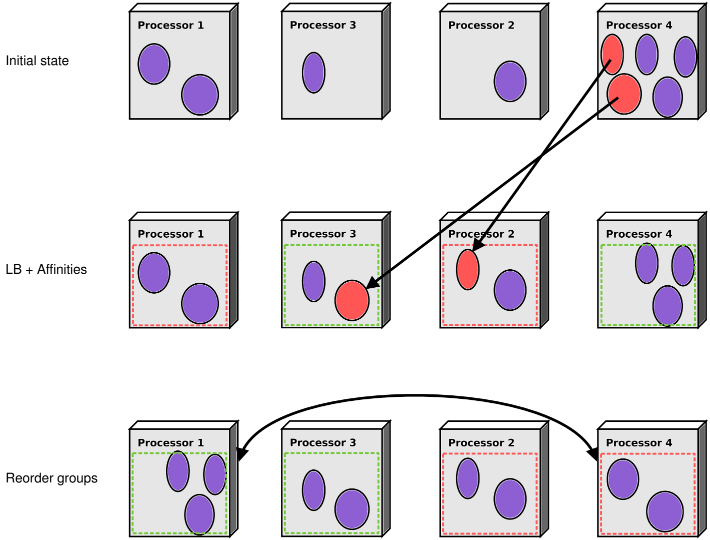
First Strategy

Initial state

LB + Affinities



First Strategy

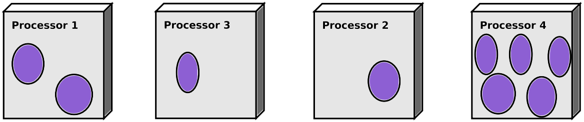


Second Strategy

- Create a communication matrix of chares
- Generate a fake topology, featuring as many leaf as chares (integer factorization)
- Run TreeMatch to find chares affinity
- Map chares to physical processors, taking into account the load and the affinity

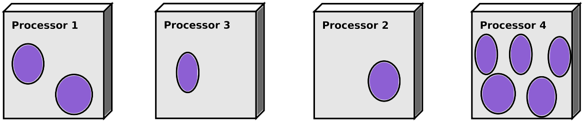
Example

Initial state

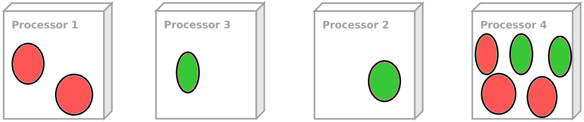


Example

Initial state



Find affinities

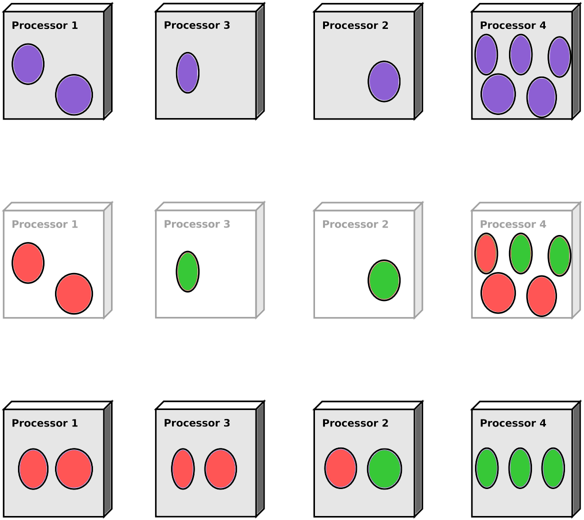


Example

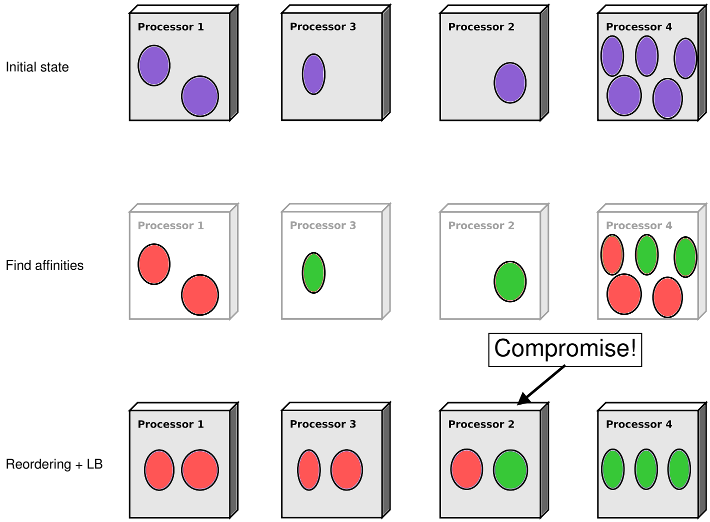
Initial state

Find affinities

Reordering + LB

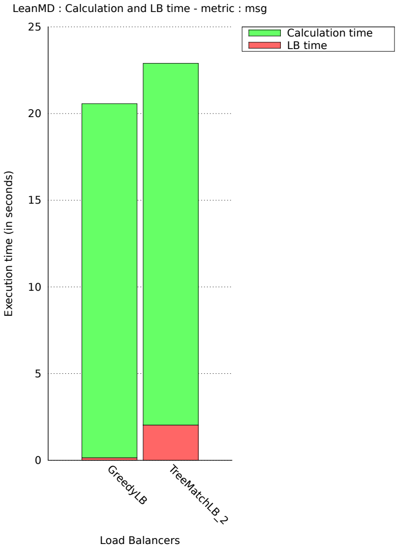


Example


Initial state
Processor 1
Processor 3
Processor 2
Processor 4
Find affinities
Processor 1
Processor 3
Processor 2
Processor 4
Compromise!
Reordering + LB
Processor 1
Processor 3
Processor 2
Processor 4

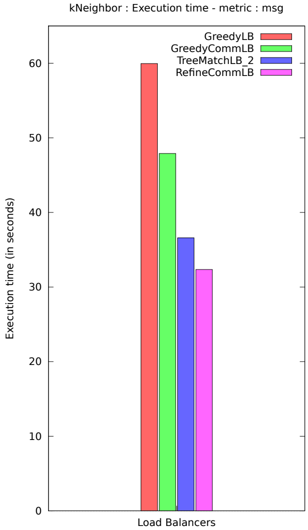
LeanMD

- Molecular Dynamics application
- Few communications
- Experiments on 4 nodes with 8 processors on each (Intel Xeon 5550)



kNeighbor

- Benchmarks application designed to simulate intensive communication between processes
- Experiments on 4 nodes with 8 processors on each (Intel Xeon 5550)
- Particularly compared to RefineCommLB
  - Take into account load and communication
  - minimize migrations



Barnes-hut Tree

- Cosmological algorithm for performing a n-body simulation
- Charm++ version
- Irregular communication (see next slide)

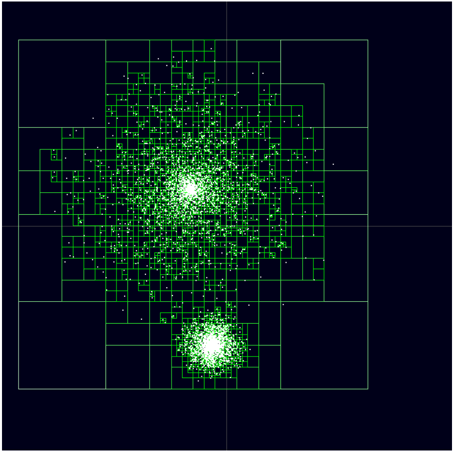


Figure: Complete Barnes-Hut tree [Wikipedia]

Preliminary results with Charm++

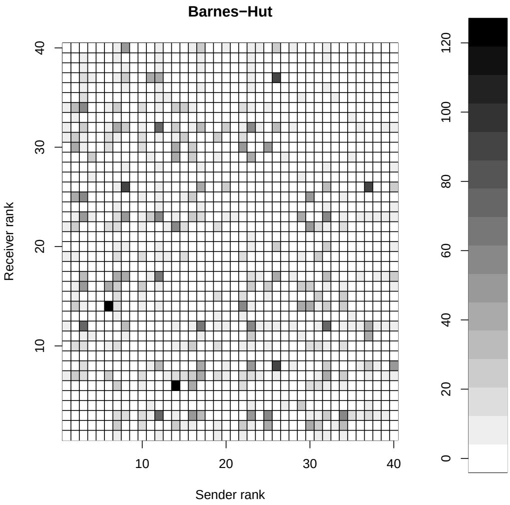
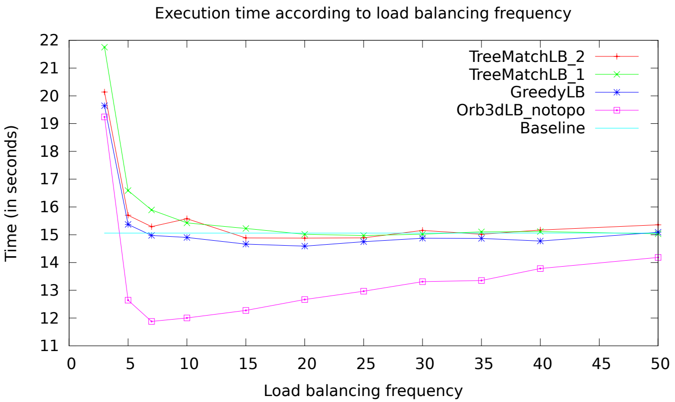


Figure: Barnes-Hut communication matrix

Preliminary results with Charm++

Barnes-Hut Tree

- No improvments for now...
- But it's a work in progress!



Preliminary results with Charm++

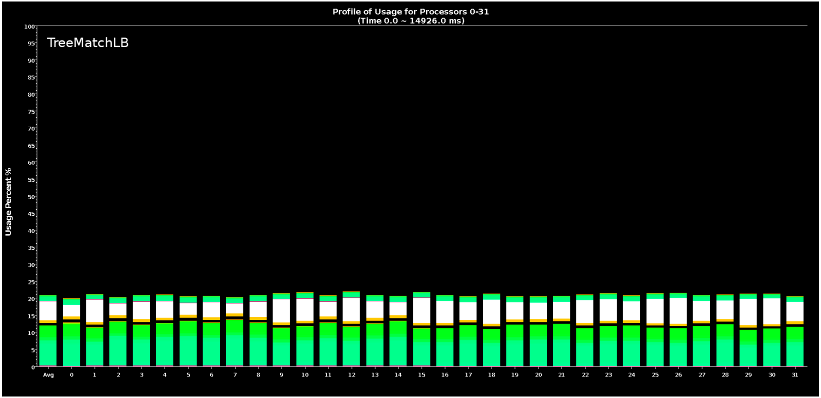


Figure: Barnes LB with TreeMatchLB

Preliminary results with Charm++

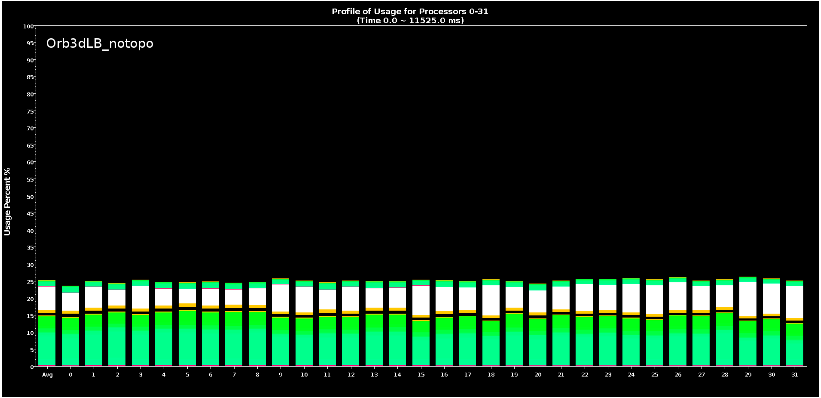


Figure: Barnes LB with Orb3d_notopo

Future works

Distributed algorithm

- Because of the lack of scalability of TreeMatch
- Divide the problem and run several instances of TreeMatch at the same time

Load balancing

- Work on an algorithm to find a better compromise between processes affinities and load balancing
- Estimate the migration impact and if necessary, include this constraint in the algorithm

Conclusion

The end

- Topology is not flat!
- Processes affinities are not uniform
- Take into account these informations to map chares could give us improvments
- Adapt our algorithm to large problems (Distributed)
- Continue collaborations with the PPL

The End

Thanks for your attention !

Any questions?