10th Workshop of the INRIA-Illinois-ANL Joint Laboratory on Petascale Computing

Using AMPI to improve the performance of the Ondes3D seismic wave simulator

Rafael Keller Tesser
rktesser@inf.ufrgs.br

Laércio Lima Pilla
llpilla@inf.ufrgs.br

Fabrice Dupros
f.dupros@brgm.fr

Philippe O. A. Navaux
navaux@inf.ufrgs.br

Jean-François Méhaut
Jean-Francois.Mehaut@imag.br

Celso Mendes
cmendes@ncsa.illinois.edu

Paper accepted for Euromicro PDP 2014
(Collaboration UFRGS, INRIA, BRGM, NCSA)

Research Context

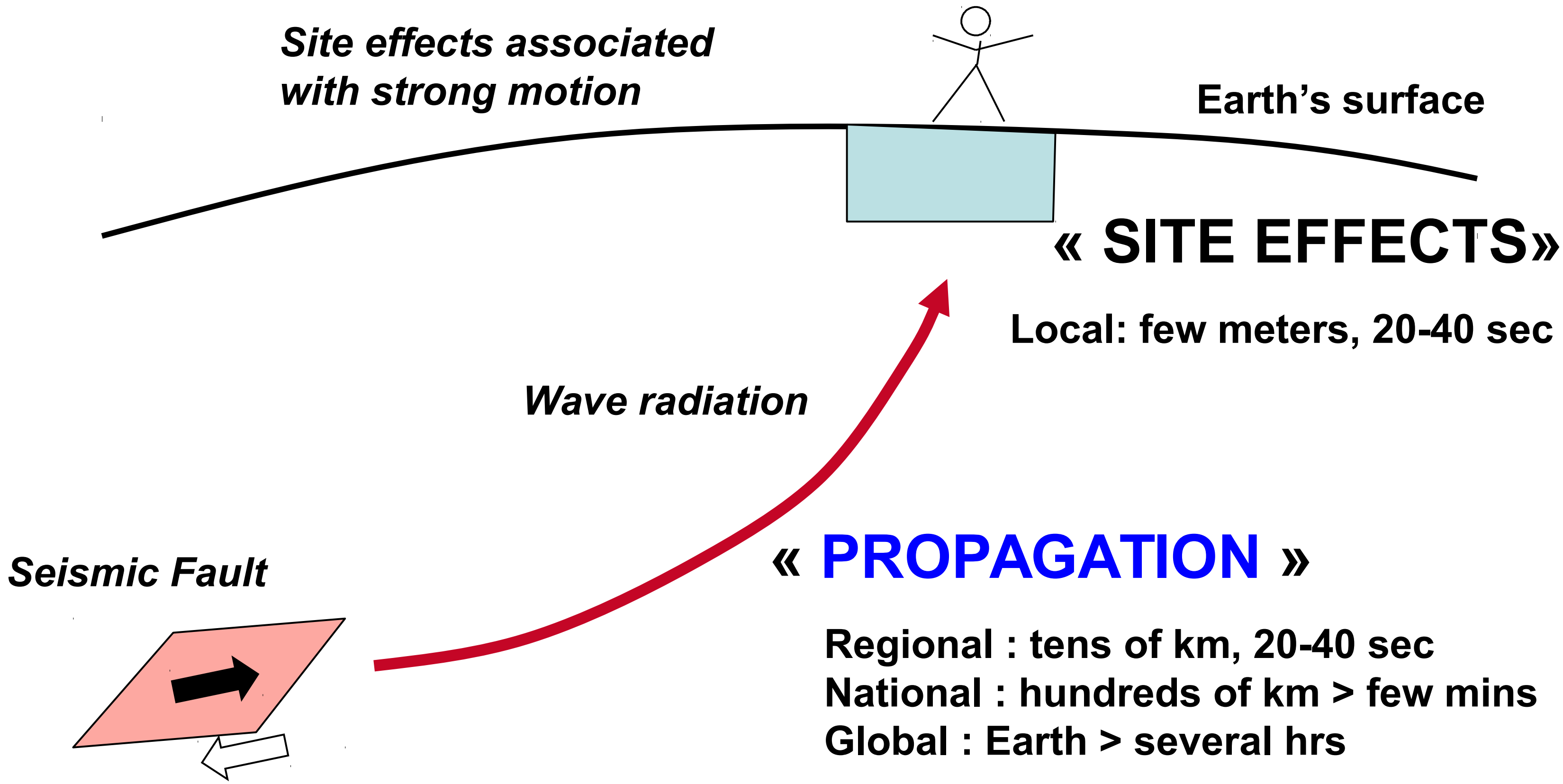
- **HPC-GA project**: High-Performance Computing for Geophysics Applications;
  - European Community's Seventh Framework Programme (FP7) IRSES Marie-Curie project;
  - International collaboration: UFRGS, INRIA, BCAM, UNAM, and BRGM;
- **LICIA** – Laboratoire International Associé;
  - Joint Computer Science Lab: Grenoble and Porto Alegre;
- **Collaboration with Urbana** as part of my PHD.

Outline

- Seismic Wave Propagation
- Modeling and Implementation
- Ondes3D
- Porting Ondes3D to AMPI
- Load Balancers with Charm++
- Overdecomposition Evaluation
- Performance Evaluation
- Conclusion

Seismic Wave Propagation

Seismic wave propagation



« SITE EFFECTS»

Local: few meters, 20-40 sec

« PROPAGATION »

Regional : tens of km, 20-40 sec
National : hundreds of km > few mins
Global : Earth > several hrs

« SOURCE »

Magnitude 6 : average of 10 km < 10 seconds

Modeling and Implementation

Seismic Wave Propagation Models

- Used to **predict the consequences** of future **earthquakes**;
- **Seismic waves** are represented by a set of **elastodynamics equations**;
  - Solved by implementing the **explicit finite difference method**;

Elastodynamics equations

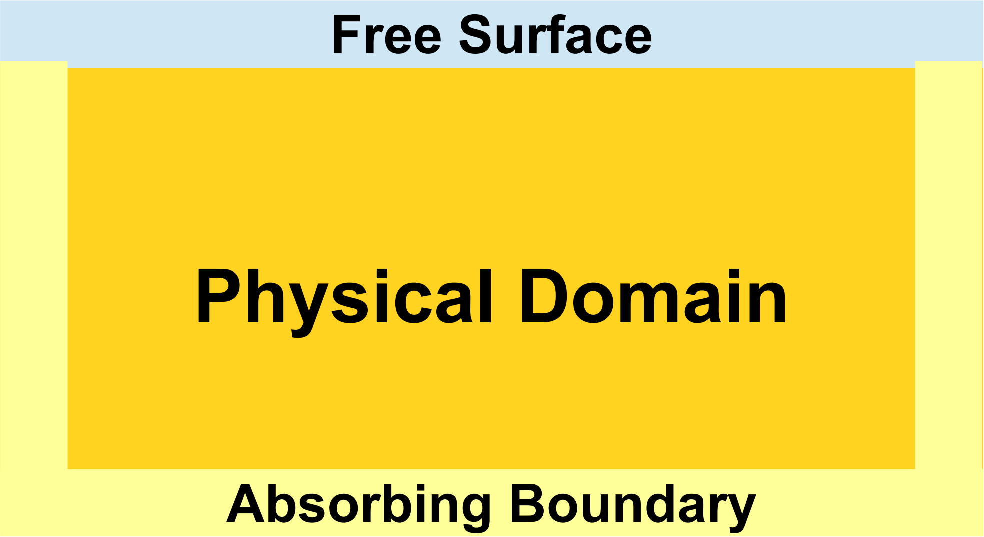
$$\begin{cases} \rho \frac{\partial}{\partial t} v_x &= \frac{\partial}{\partial x}\sigma_{xx} + \frac{\partial}{\partial y}\sigma_{xy} + \frac{\partial}{\partial z}\sigma_{xz} + f_x \\ \rho \frac{\partial}{\partial t} v_y &= \frac{\partial}{\partial x}\sigma_{yx} + \frac{\partial}{\partial y}\sigma_{yy} + \frac{\partial}{\partial z}\sigma_{yz} + f_y \\ \rho \frac{\partial}{\partial t} v_z &= \frac{\partial}{\partial x}\sigma_{zx} + \frac{\partial}{\partial y}\sigma_{zy} + \frac{\partial}{\partial z}\sigma_{zz} + f_z \end{cases} \tag{1}$$

$$\begin{cases} \frac{\partial}{\partial t}\sigma_{xx} &= \lambda \left(\frac{\partial}{\partial x} v_x + \frac{\partial}{\partial y} v_y + \frac{\partial}{\partial z} v_z \right) + 2\mu \frac{\partial}{\partial x} v_x \\ \frac{\partial}{\partial t}\sigma_{yy} &= \lambda \left(\frac{\partial}{\partial x} v_x + \frac{\partial}{\partial y} v_y + \frac{\partial}{\partial z} v_z \right) + 2\mu \frac{\partial}{\partial y} v_y \\ \frac{\partial}{\partial t}\sigma_{zz} &= \lambda \left(\frac{\partial}{\partial x} v_x + \frac{\partial}{\partial y} v_y + \frac{\partial}{\partial z} v_z \right) + 2\mu \frac{\partial}{\partial z} v_z \\ \frac{\partial}{\partial t}\sigma_{xy} &= \mu \left(\frac{\partial}{\partial y} v_x + \frac{\partial}{\partial x} v_y \right) \\ \frac{\partial}{\partial t}\sigma_{xz} &= \mu \left(\frac{\partial}{\partial z} v_x + \frac{\partial}{\partial x} v_z \right) \\ \frac{\partial}{\partial t}\sigma_{yz} &= \mu \left(\frac{\partial}{\partial z} v_y + \frac{\partial}{\partial y} v_z \right). \end{cases} \tag{2}$$

v: velocity field;
σ: stress field;
f: a known external source force;
ρ: the material density;
λ and μ: elastic coefficients known as Lamé parameters.

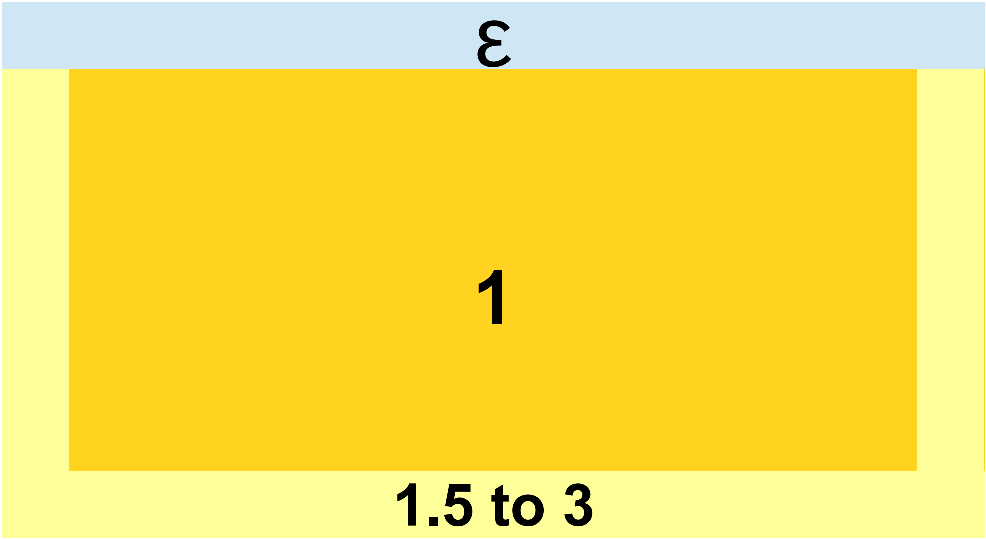
Boundary Conditions

- The **model** considers a **finite computing domain**;
- But the **physical problem** is **unbounded**;
- Need for **artificial boundary conditions** to a**bsorb the outgoing energy;**



- Specific set of **equations at the edges of the three dimensional geometry**;

Absorbing condition: C-PML



Free Surface

Physical Domain

Absorbing boundaries

- ABC → **C-PML** method (Berenger 1995, Komatitsch 2007) ;
  - **Variable CPU cost (incidence angle)**.

Parallel implementation

- The **domain** is represented by a **three dimensional grid**;
- **2D** Cartesian **decomposition**;
- **Problem**: **Boundary condition causes unbalanced load**;
  - Tasks at the **borders** perform **more computation**;
- Other **sources of load imbalance**:
  - Variation in the **constitution laws** of different **geological layers**;
  - **Wave propagation**.

Ondes3D

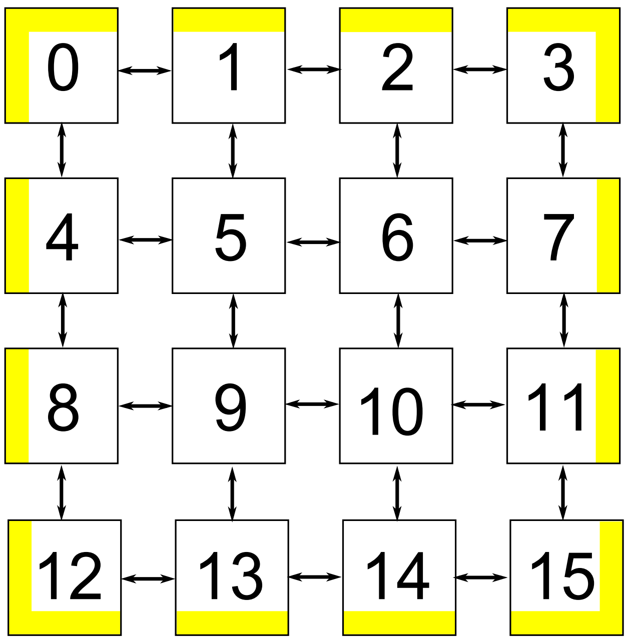
Ondes3D

- **Ondes3D** is a **seismic wave propagation simulator**;
- Developed by BRGM;
- Follows the implementation scheme from the previous slides;
- Our work is based on an **MPI Implementation**;

Ondes3D: MPI Implementation

- **Communication/Computation overlap**:
  - **Compute** the **points in the borders** of the subdomain;
  - **Send the borders to neighbor subdomains**;
    - Using **non-blocking communication**;
  - **Compute the center** of the subdomain.
- **Boundary Condition**: **Convolutional Perfectly Matched Layer (C-PML)**;
    - Standard **thickness** of **10** grid points.

Load Imbalance with MPI



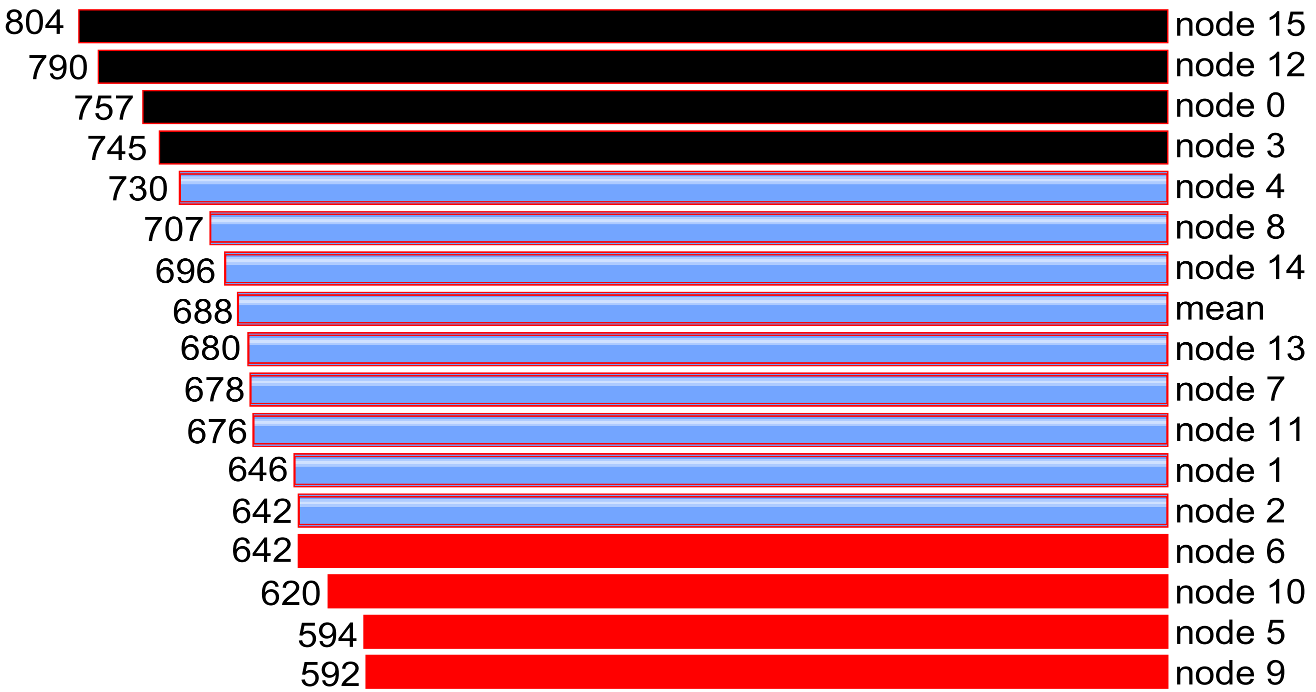


Figure: Load distribution of Ondes3D with a 4x4 decomposition

- *Results for an execution of the MPI implementation;*
  - *72 million grid points;*
- *The data was analyzed with TAU.*

Previous attempts

- The **MPI implementation** is **unbalanced**;
- **Previous attempts** to solve the problem:
  - **mesh partitioning** techniques;
  - **quasi-static load balancing algorithm** based on zone costs;
- Problem: **difficulties** to accurately **predict** the **execution time** of various parts of the program:
  - cache effects;
  - arithmetic considerations;
  - compiler behavior.

Proposal

Evaluate the use of dynamic load balancing to improve the performance of Ondes3D.

- **Port** of **Ondes3D** to **AMPI**:
  - A mature **dynamic load-balancing infrastructure**;
    - **Domain overdecomposition (virtual processors)**;
    - **Migration**;
  - **MPI-like** programming model;

Proposal

- **Evaluation** of the **performance** of the **AMPI version**;
  - **Compared** to the **MPI** implementation;
  - **Four load balancers distributed with Charm++**;
  - **Two topology aware load balancers**: **NucoLB** and **HwTopoLB**;

Port to AMPI

Port of Ondes3D to AMPI

- **Virtual processors support:**
  - Removal of global an static variables;
    - due to the **use of user-level threads in place of processes**;
    - Fortunately, **most global variables in Ondes3D are constants**;

MPI_Migrate

- **Support to process migration:**
  - **Implementation** of functions for **data serialization**;
    - **PUP functions**: **P**acking and **U**n**p**acking;
  - Destruction and creation of MPI_Request variables;
  - **Register the Pack and Unpacking function** (MPI_Register);
  - **Call MPI_Migrate()** every N time-steps:
    - N is defined at compiling time.

Evaluation

Hardware Description

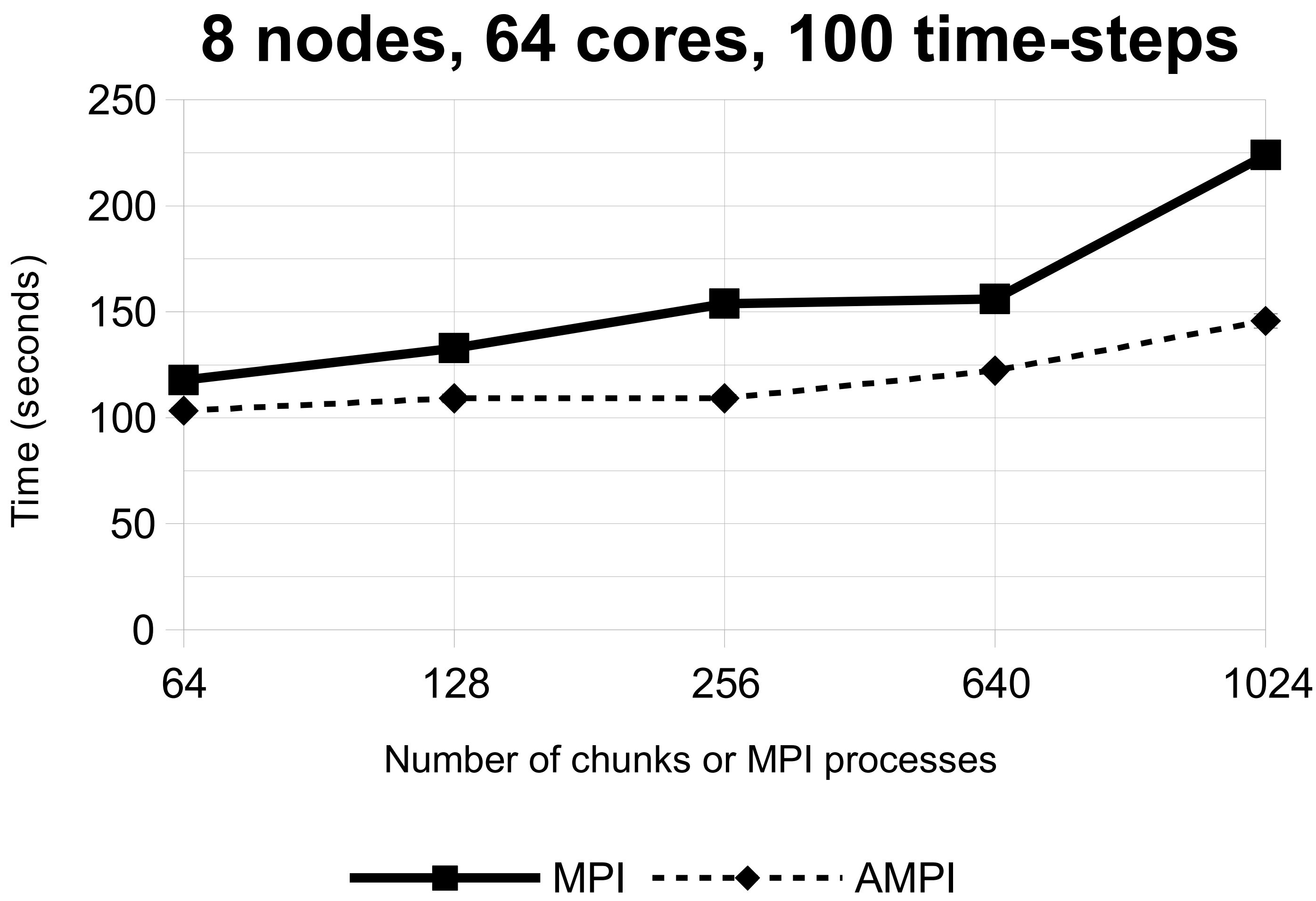
- Cluster Adonis from Grenoble (Grid'5000);
- CPU: Intel Xeon E5520 (Nehalem), 2. 27 GHz:
- 4 cores x 2 CPUs x 8 nodes = 64 cores
- Last level cache: 8 MB;Memory: 24 GB;
- InfiniBand 40G (Mellanox ConnectX IB 4X QDR MT26428).

Simulation

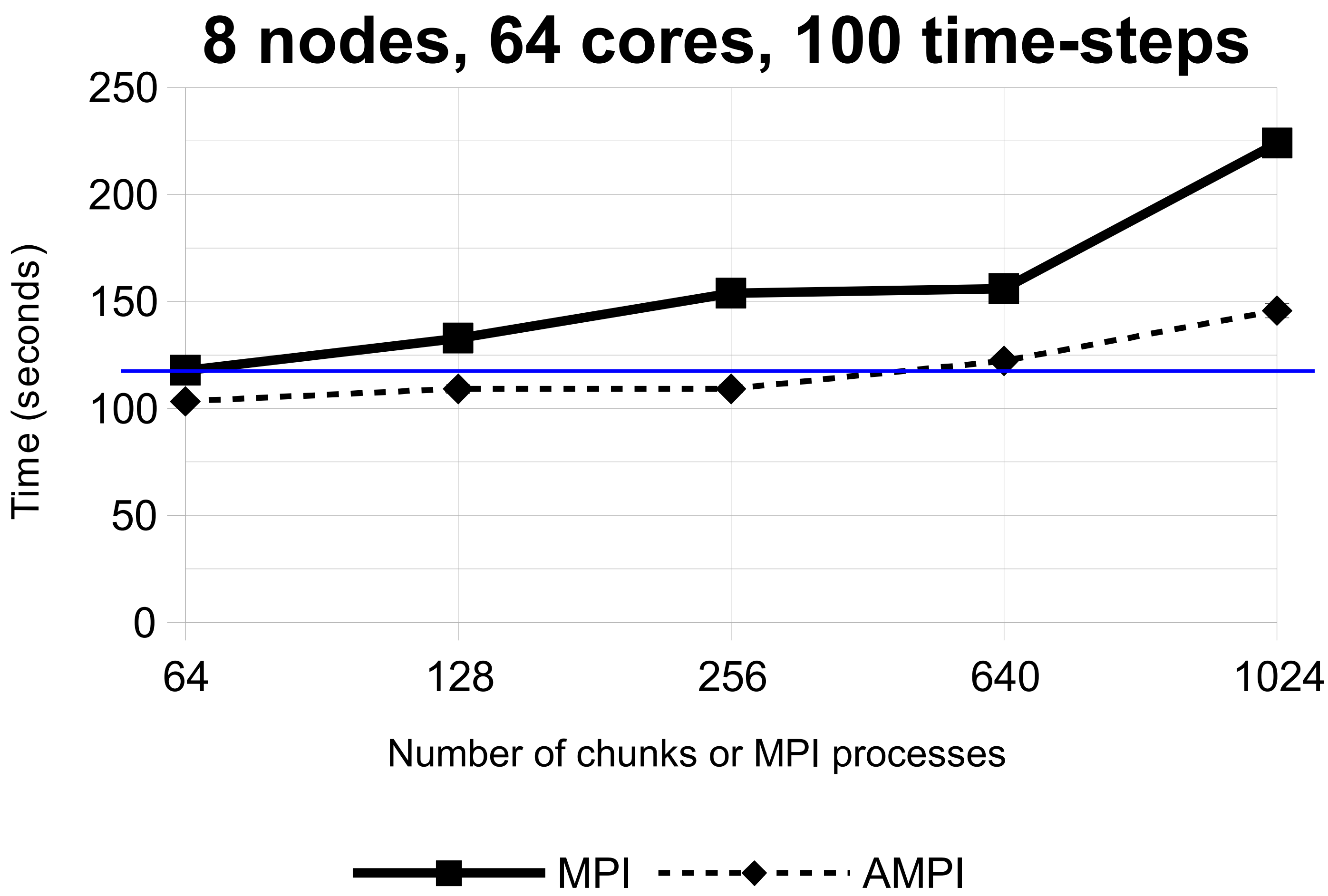
- Based on Mw6.6, 2007 Niigata Chuetsu-Oki, Japan, earthquake (Aochi et.al ICCS 2013) ;
  - Full problem (6000 time steps) → 162 minutes on 32 nodes (Intel Hapertown processors).
- Resolution : **122 million of grid points**;

Overdecomposition Evaluation

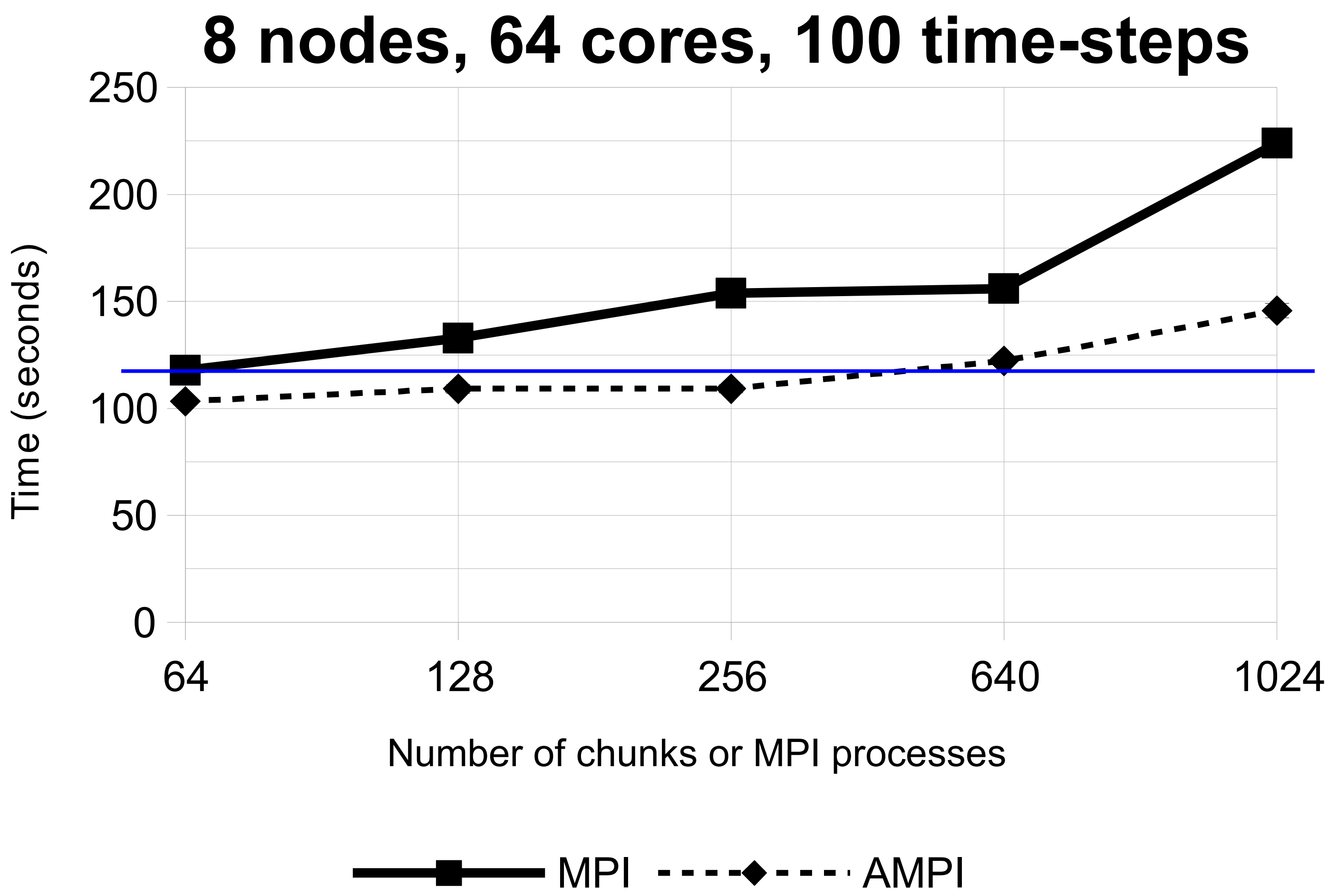
Overdecomposition evaluation: MPI vs. AMPI



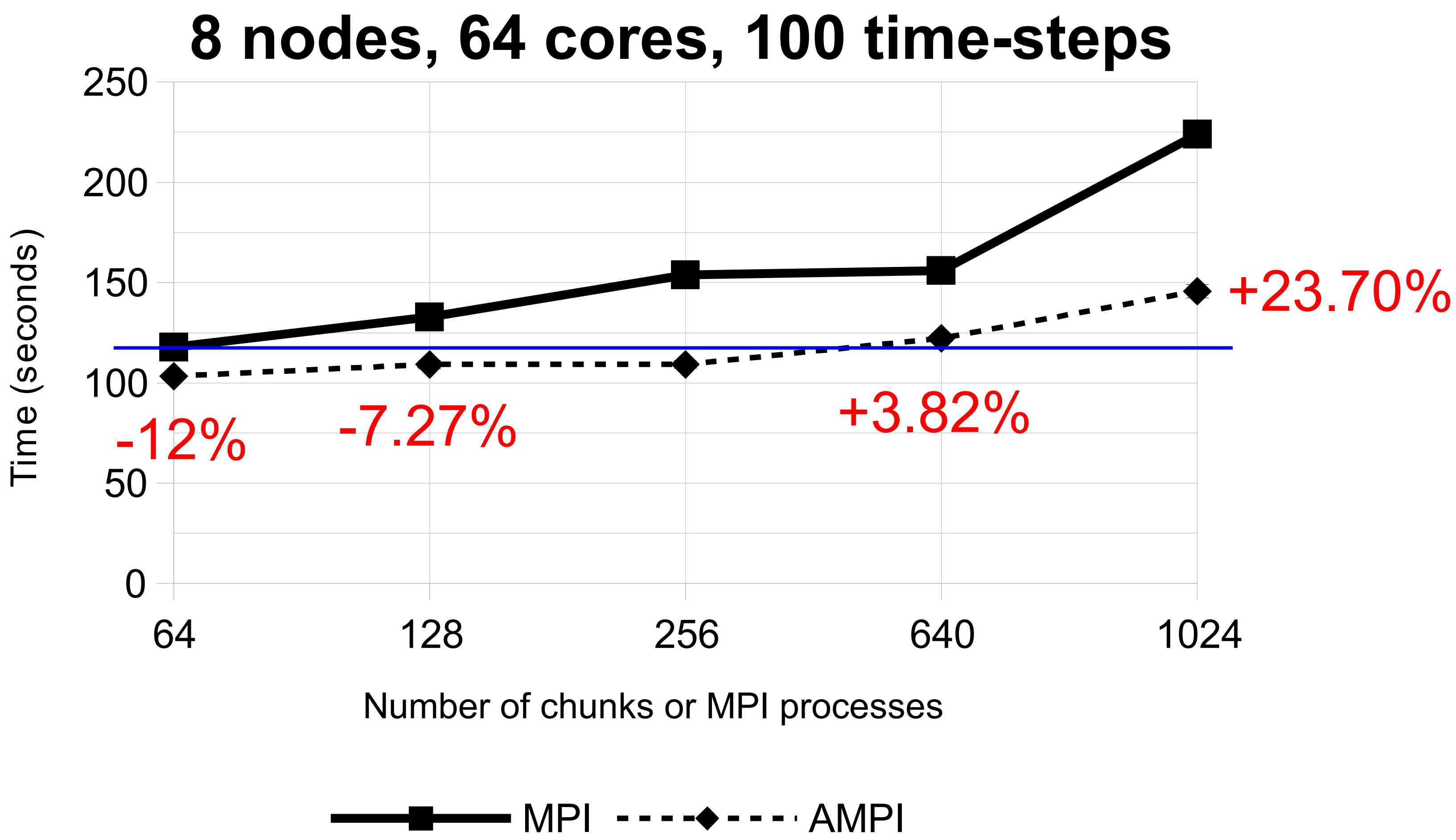
Overdecomposition evaluation: MPI vs. AMPI



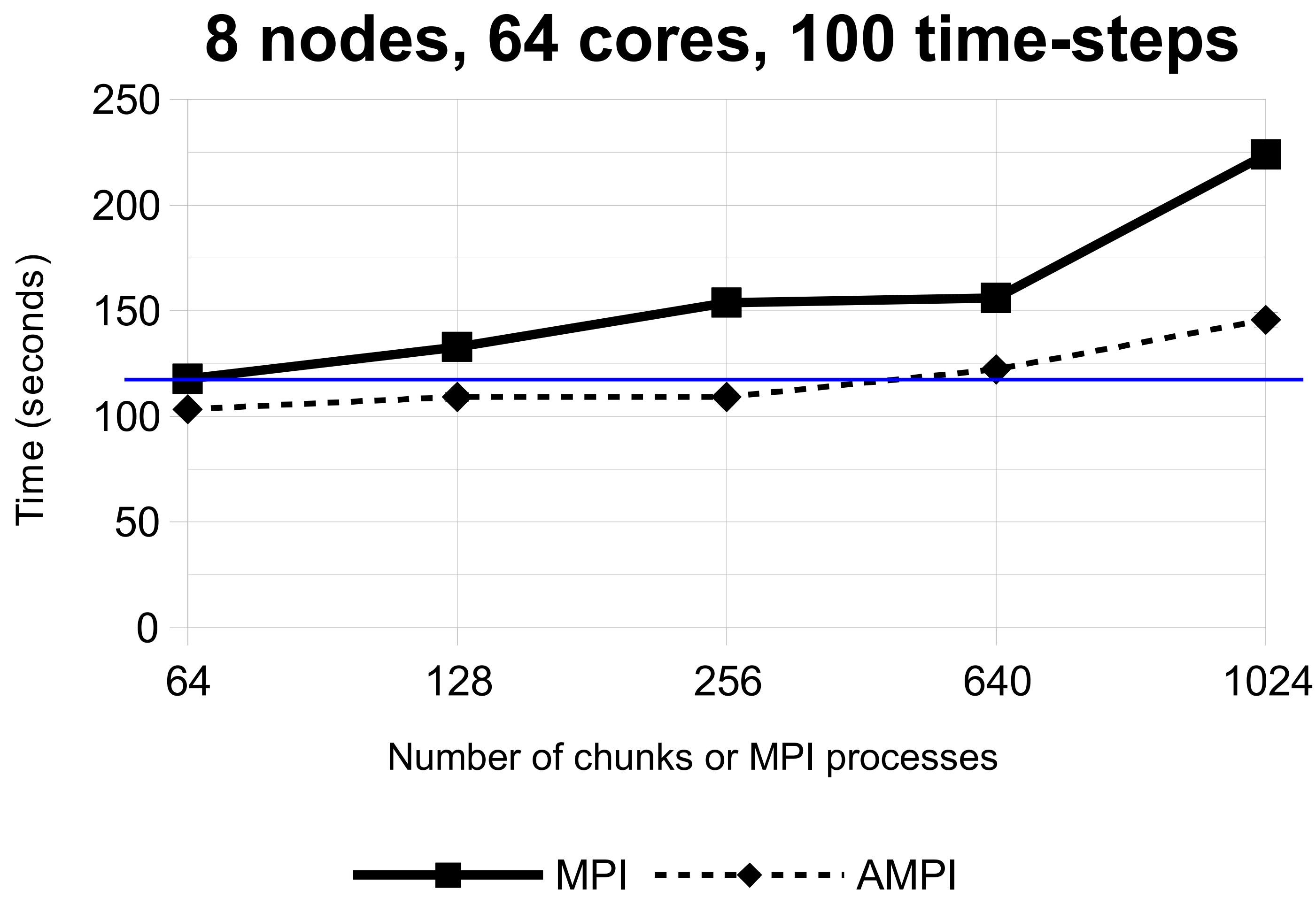
Overdecomposition evaluation: MPI vs. AMPI



Overdecomposition evaluation: MPI vs. AMPI



Overdecomposition evaluation: MPI vs. AMPI



Low overhead compared to our best MPI result.

Performance Evaluation

Usage profile with AMPI

Figure: Processor usage profile of Ondes3D with one process per core

- *For the period from 25s to 75s of a 103s execution;*
- *100 time steps*
- *122 million grid points;*
- ***Average usage among all processes: 81.72%.***

Load Balancers

- These are the load balancers we used in the experiments:
  - **GreedyLB**:
    - aggressive scheduling decisions;
    - It is a greedy algorithm that uses only VPs loads for its decisions;
    - iteratively maps the virtual processor with the biggest load to the least loaded core;
  - **GreedyCommLB:**
    - includes communication loads;
    - Instead of simply mapping the VP with the biggest load to the least loaded core to map, it considers all other cores that have VPs that communicate with it;

Load Balancers (cont.)

- **RefineLB**:
  - tries to improve load balance by incrementally adjusting the current scheduling;
  - checks all possible VP migrations from the most loaded core to cores below the average load;
  - migrates the VP that leaves its new core the closest to the average;
  - less migrations than GreedyLB and GreedyCommLB;
- **RefineCommLB:**
  - adds communications costs to RefineLB;
  - considers that a communication overhead is present whenever a VP is mapped to a different core than the ones that contain VPs that it communicates with.

Load Balancers (cont.)

- **NucoLB*:**
  - Developed for parallel platforms with non-uniform levels in their topologies (mainly NUMA nodes);
  - Assigns the VP with the largest load to the core that presents the smallest cost;
  - The cost is related to:
    - the current load on the core;
    - the communication cost of mapping such VP to it;

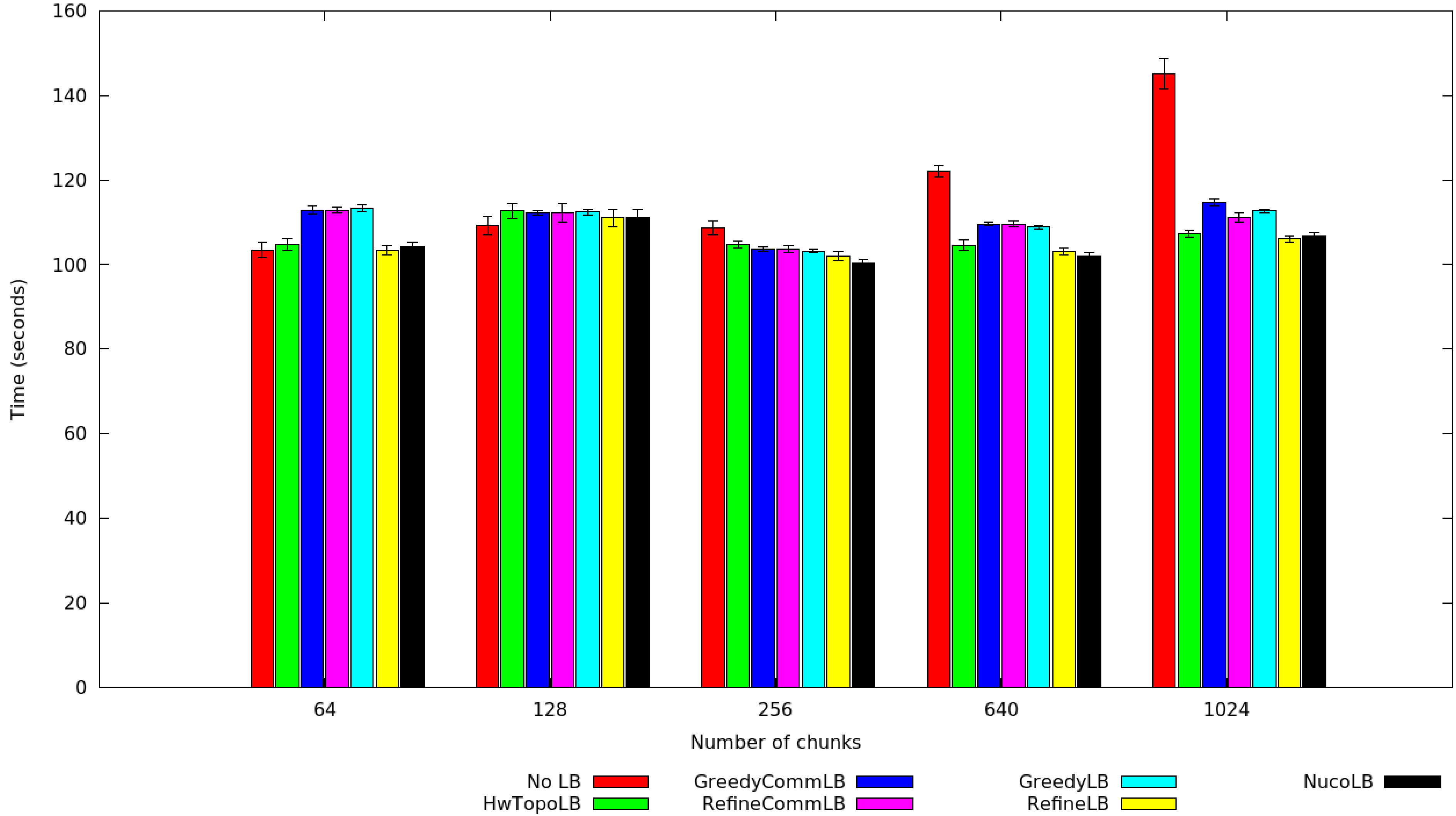
* L. L. Pilla, C. P. Ribeiro, D. Cordeiro, C. Mei, A. Bhatele, P. O. A. Navaux, F. Broquedis, J. Méhaut, and L. V. Kale *“A Hierarchical Approach for Load Balancing on Parallel Multi-core Systems”*, ICPP 2012

Load Balancers (cont.)

- **HwTopoLB*:**
  - Trade-off: map a VP to a more underutilized core or mapping it closer to the other VPs it communicates with**;**
  - considers the whole machine topology:
    - caches, memory and network;
    - chooses a core and a VP that is assigned to it;
    - evaluates all possible mappings;
    - chooses the one that has the highest probability of minimizing the makespan;
  - proven to asymptotically converge to an optimal solution.

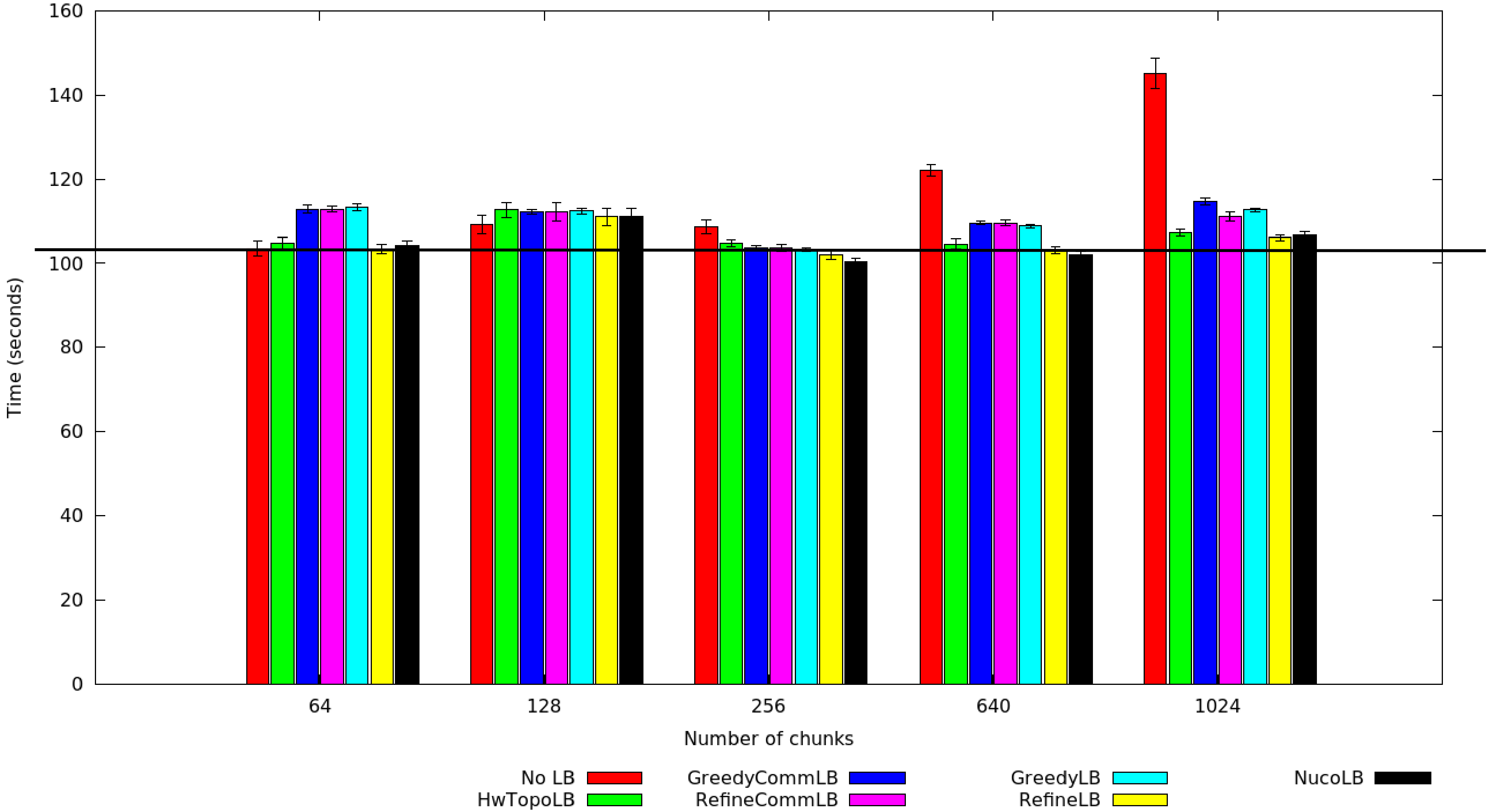
* L. L. Pilla, C. P. Ribeiro, P. Coucheney, F. Broquedis, B. Gaujal, P. O. A. Navaux, and J.-F. Méhaut, "*A Topology-Aware Load Balancing Algorithm for Clustered Hierarchical Multi-Core Machines*," Future Generation Computer Systems, 2013.

100 time-steps
Average execution times

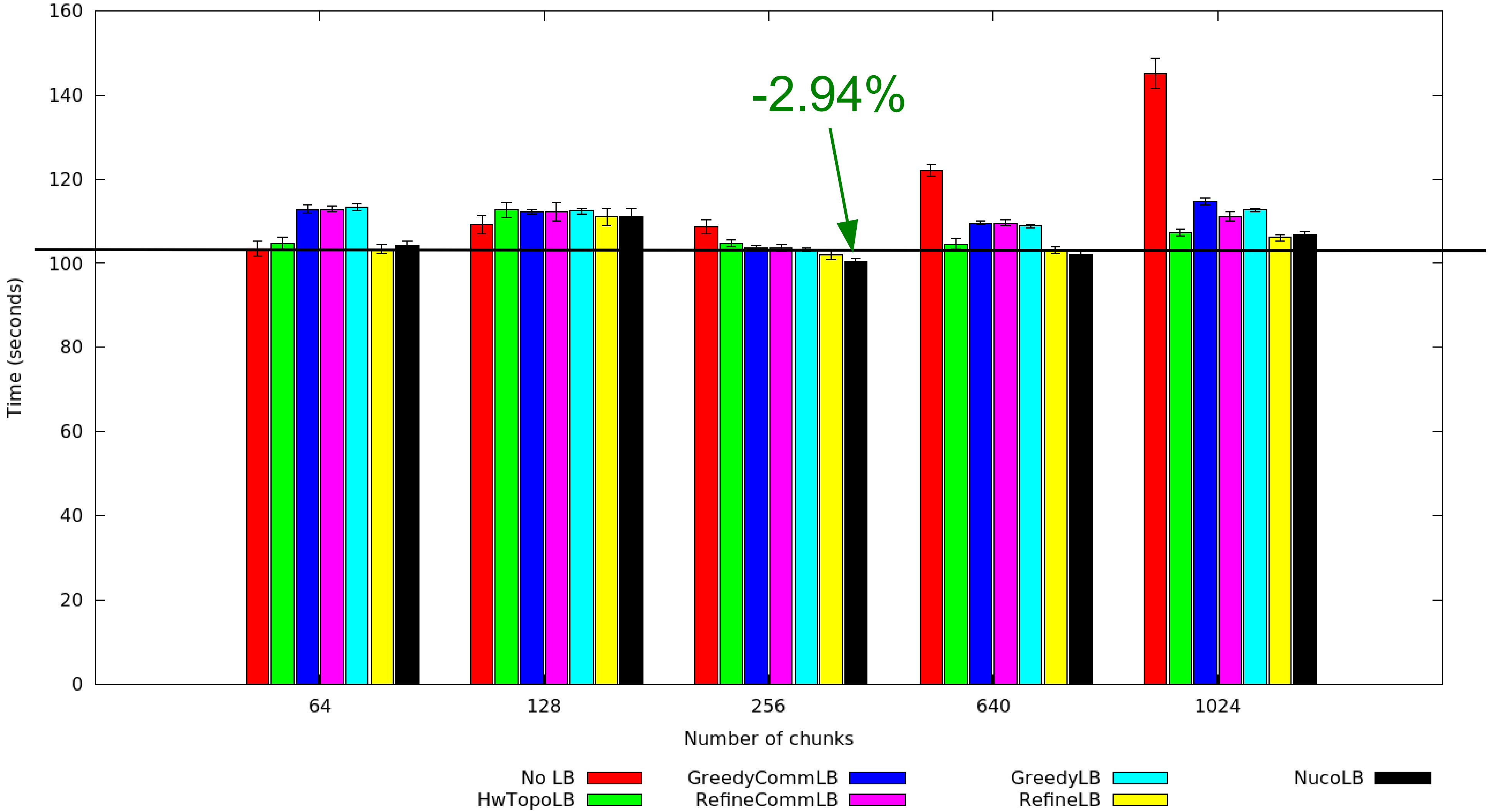


**With 95% confidence intervals.*

100 time-steps
Average execution times

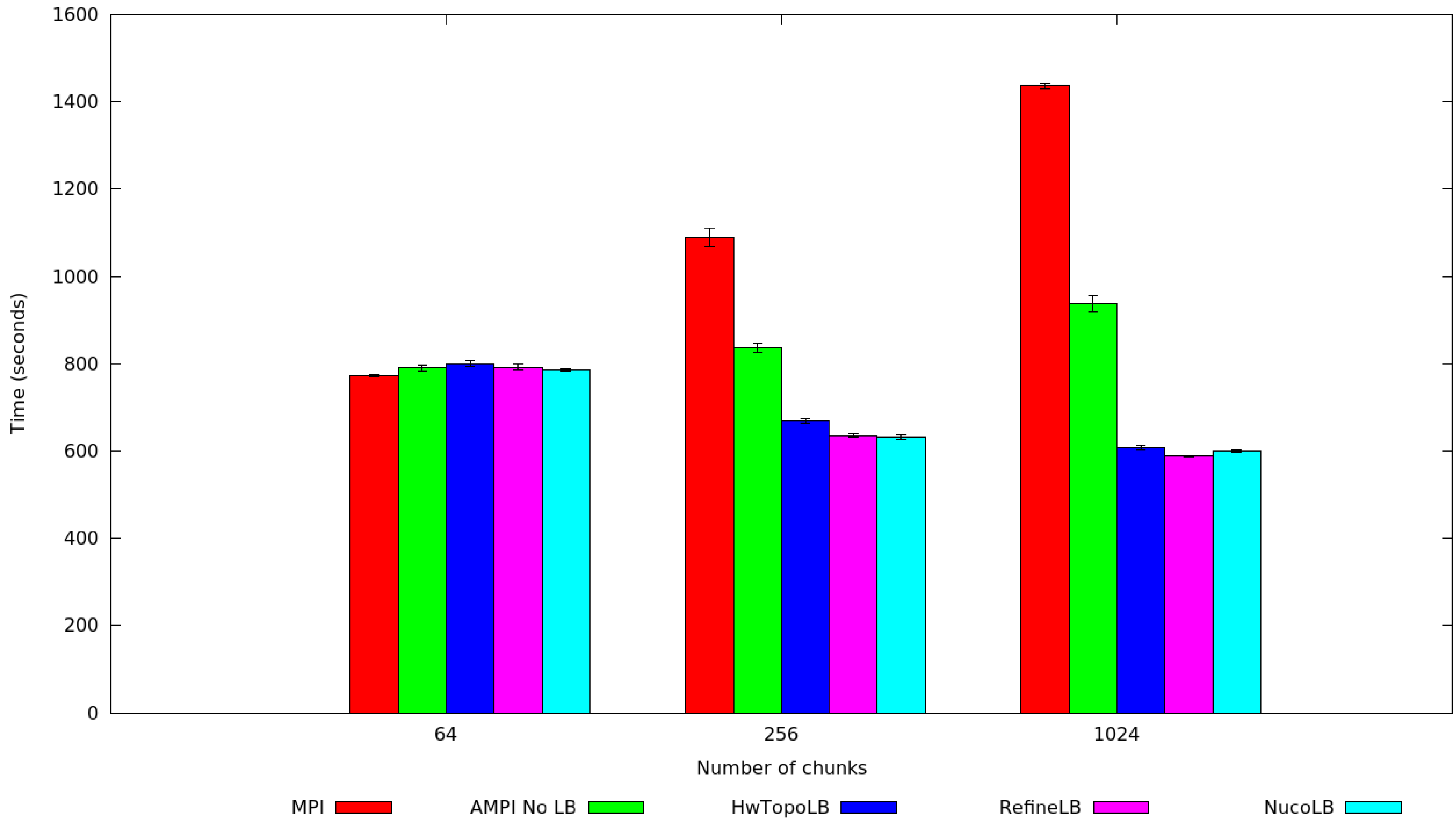


100 time-steps
Average execution times



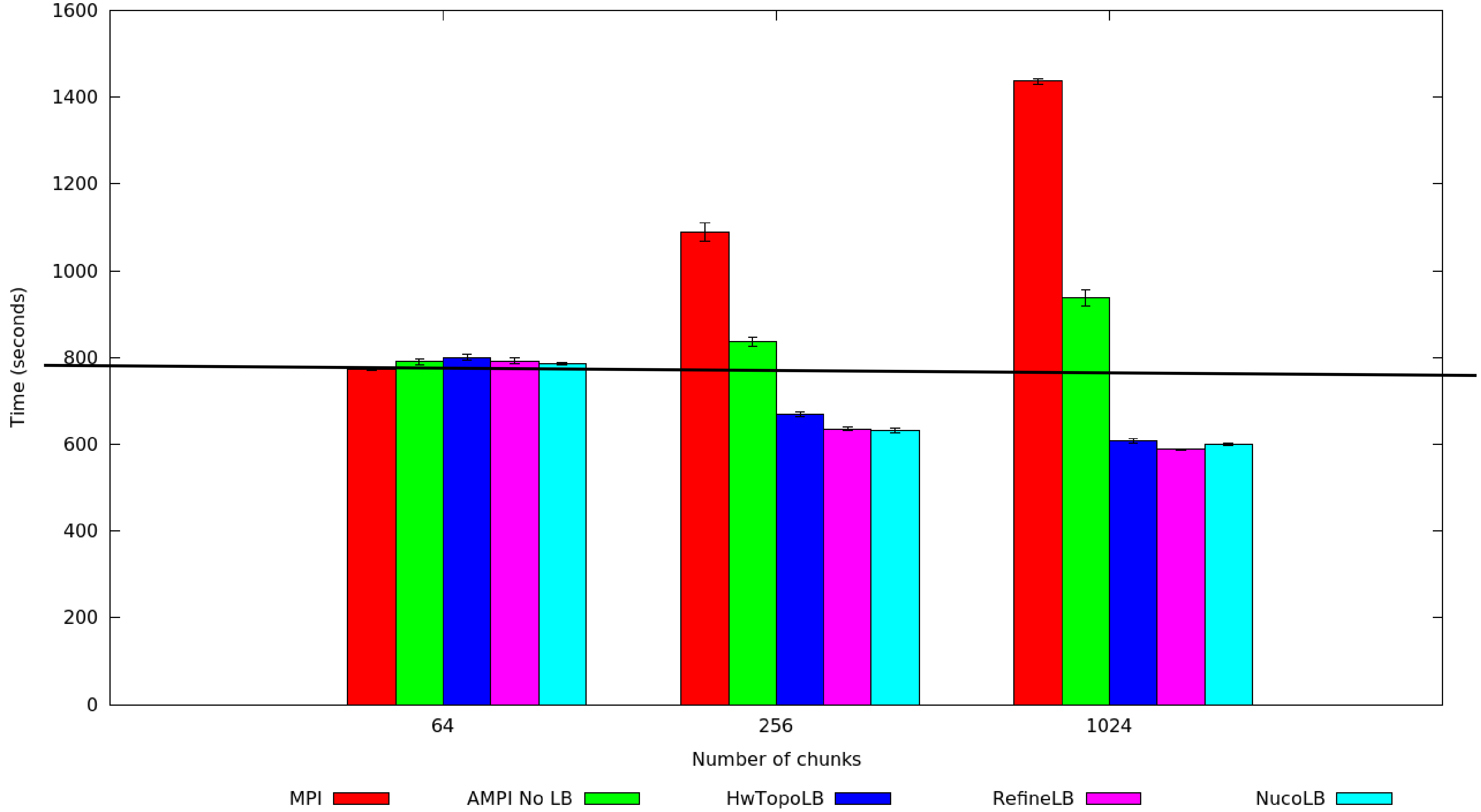
Very small performance gain.

500 time-steps
Average execution times

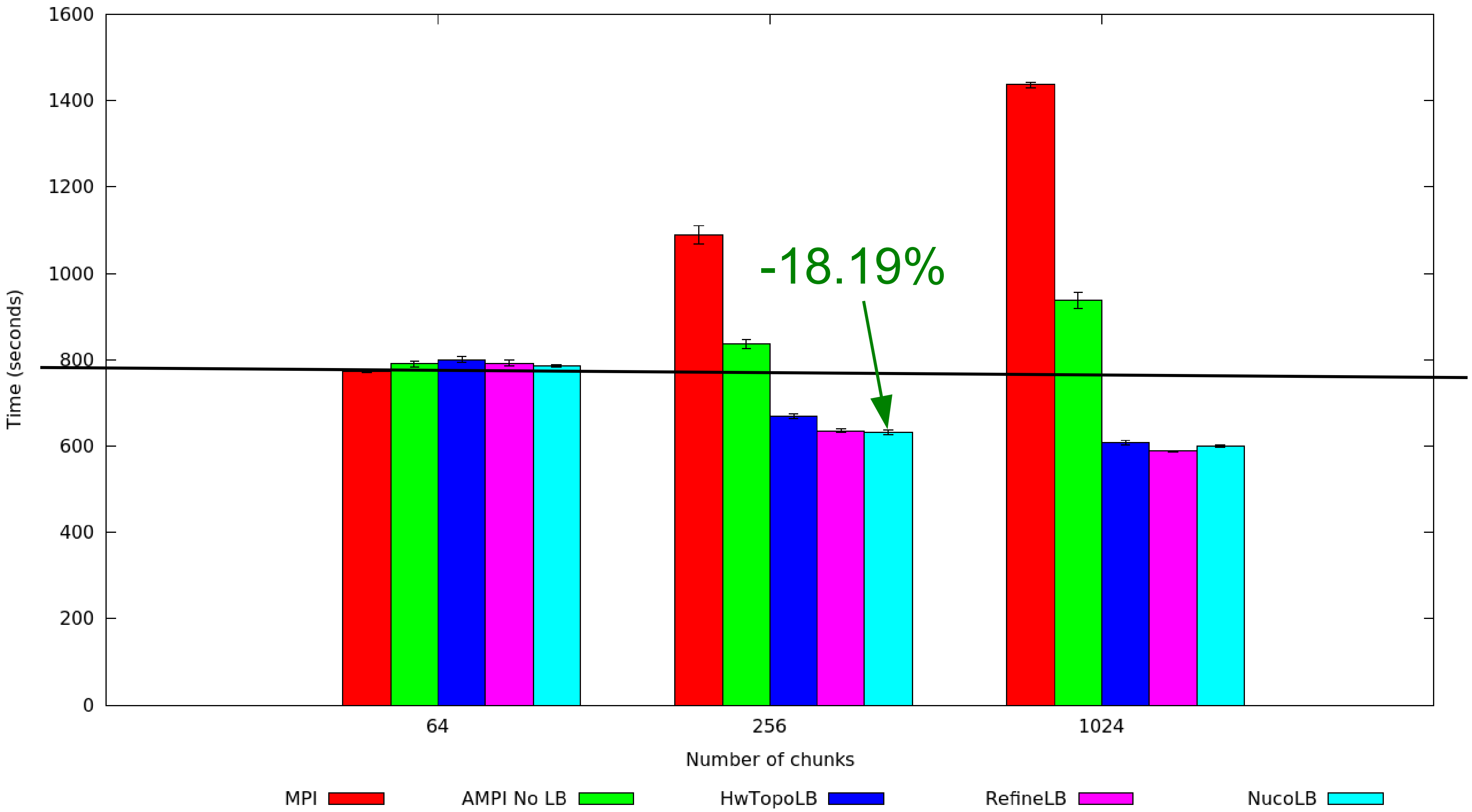


**With 95% confidence intervals.*

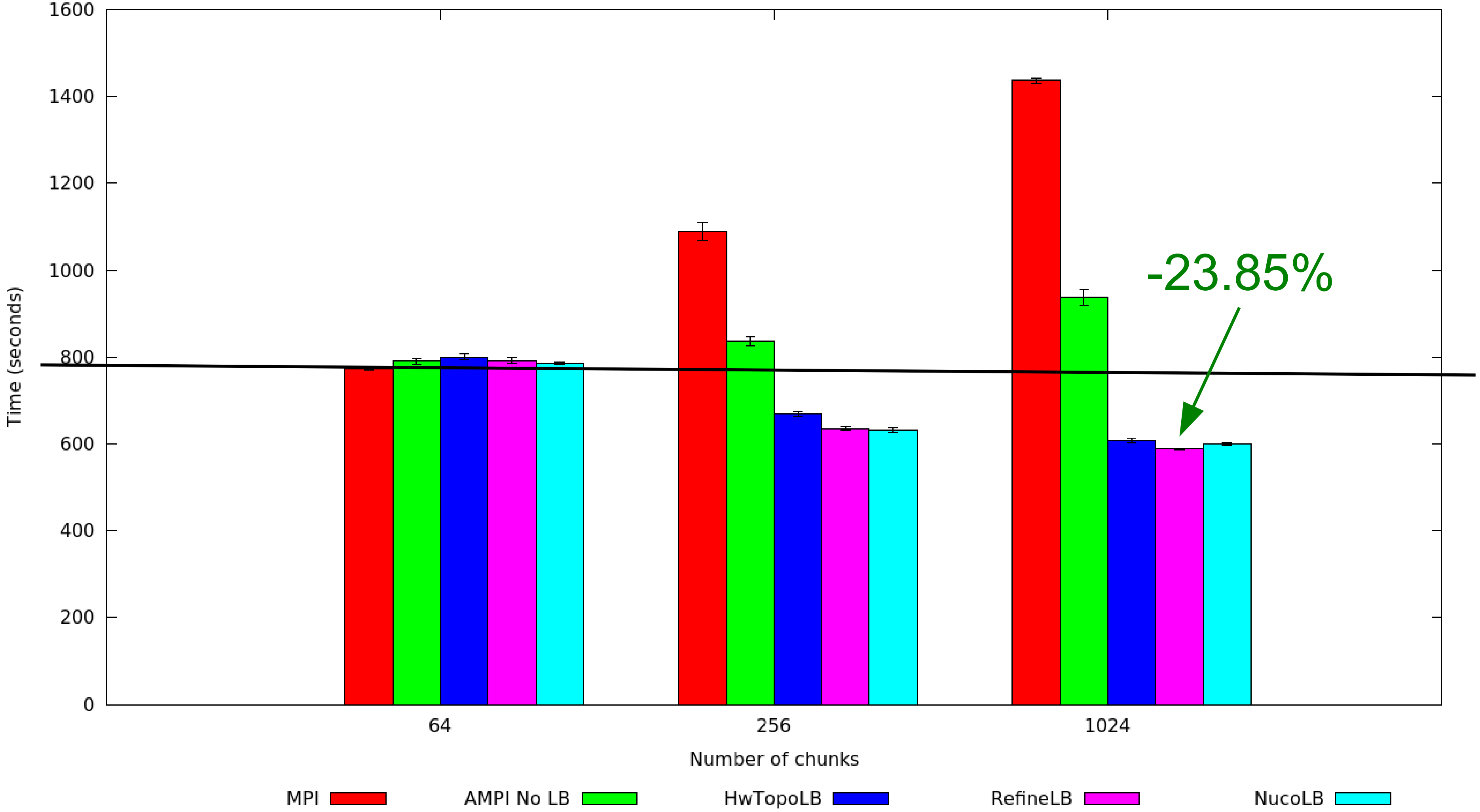
500 time-steps
Average execution times



500 time-steps
Average execution times

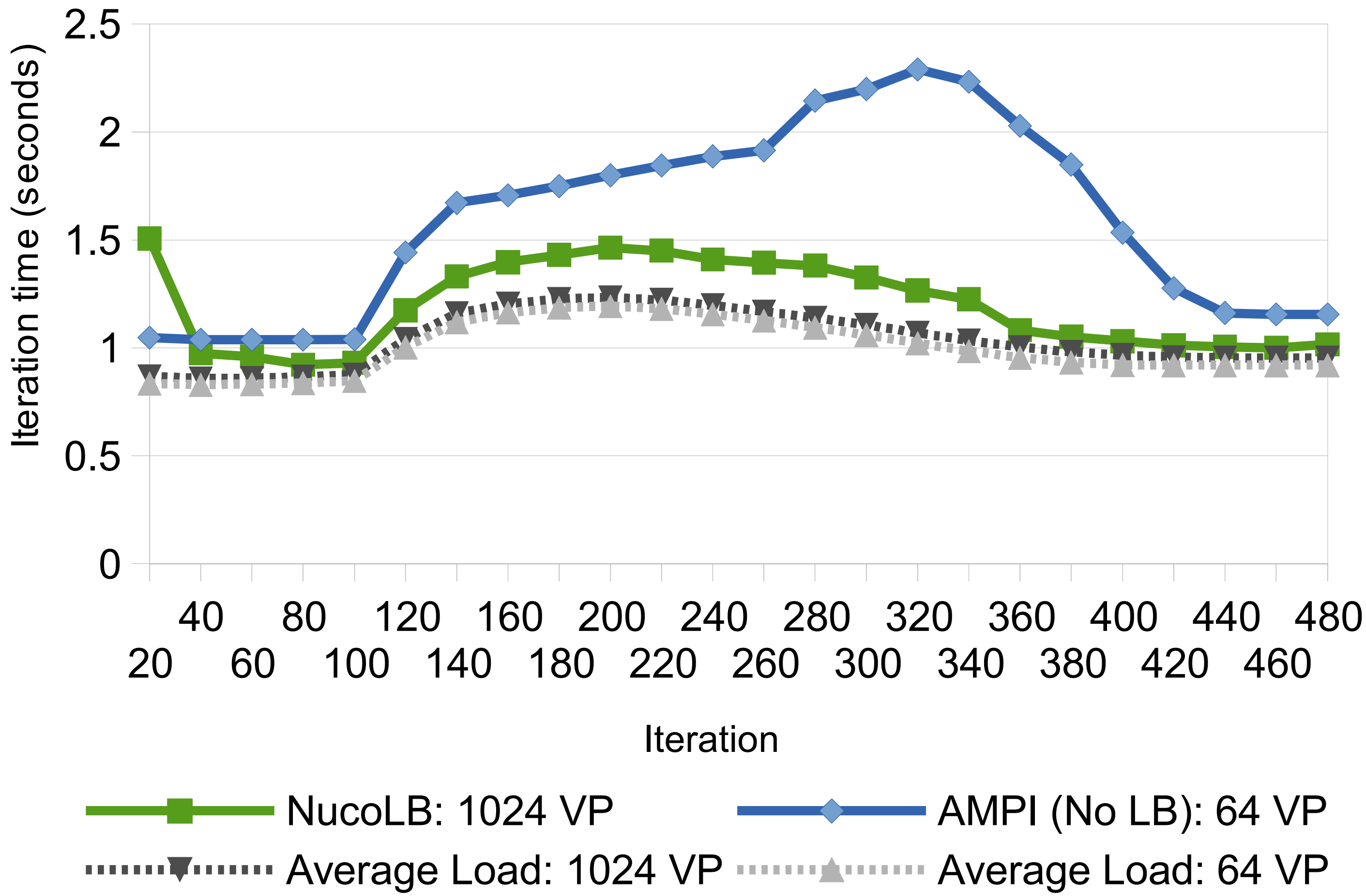


500 time-steps
Average execution times

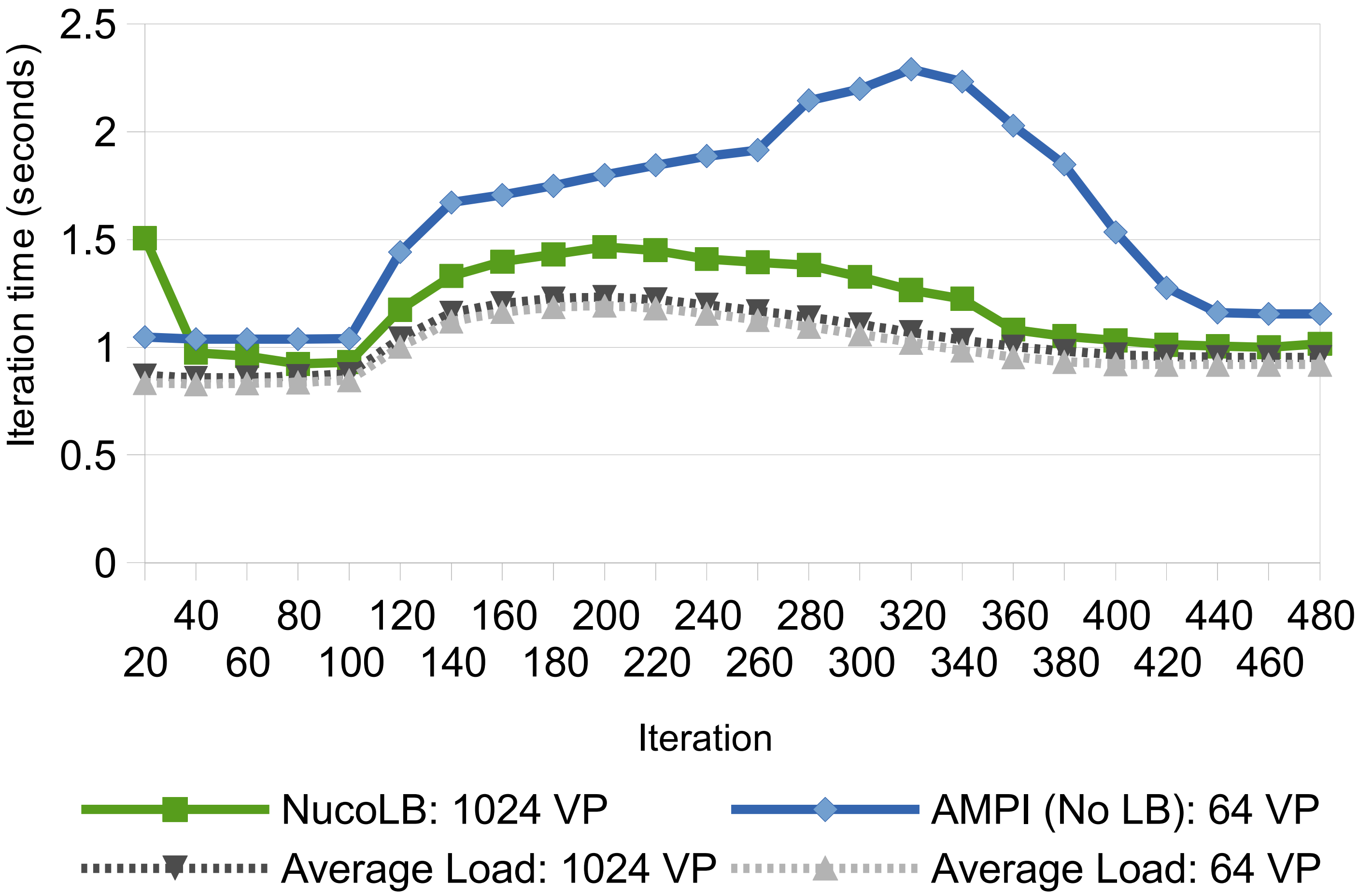


Up to 23.85% improvement.

Average and Maximum VP Loads



Average and Maximum VP Loads



- **The LB was able to maintain the maximum load close to the average;**
- **Even when the unbalanced load presented significant variation.**

Conclusion

Conclusion

- Load balancing is a real problem in the simulation of seismic wave propagation;
- Dynamic load-balancing with AMPI:
  - Up to 23.85% performance gain;
  - Load-balancer keeps the maximum load closer to the average;
  - Bonus: maintain a familiar programming model.

Future Work

- Ondes3D:
  - Larger scale;
  - Higher resolution;
  - Tune the frequency of load balancing calls;
  - Run a full simulation (6000 time-steps);
  - Tests with different simulations;
  - Instrumentation for simulation with BigSim;

Future Work

- GPU integration:
  - We are currently testing a GPU implementation on Tesla K20;
    - Still need to optimize the code for the architecture;
  - If possible, we intend to integrate the GPU kernels with our AMPI code.

Thank you!

Using AMPI to improve the performance of the Ondes3D seismic wave simulator

Rafael Keller Tesser
rktesser@inf.ufrgs.br

Laércio Lima Pilla
llpilla@inf.ufrgs.br

Fabrice Dupros
f.dupros@brgm.fr

Philippe O. A. Navaux
navaux@inf.ufrgs.br

Jean-François Méhaut
Jean-Francois.Mehaut@imag.br

Celso Mendes
cmendes@ncsa.illinois.edu

Paper accepted for Euromicro PDP 2014
(Collaboration UFRGS, INRIA, BRGM, NCSA)