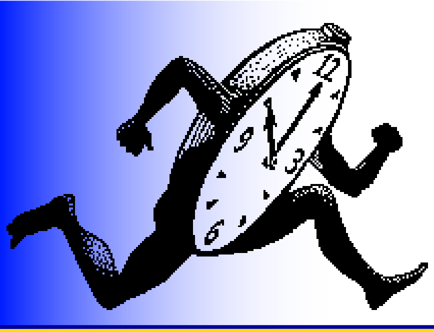
The ninth Workshop of the
INRIA-Illinois Joint Laboratory

Topology Management and MPI Implementations Improvements

Guillaume Mercier, Emmanuel Jeannot, François Tessier

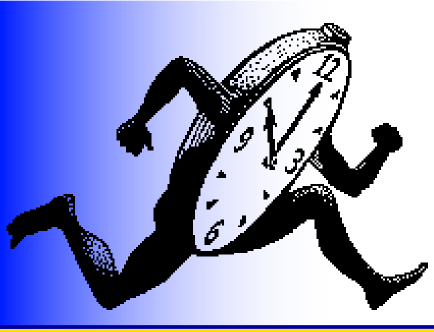
INRIA Bordeaux Sud-Ouest – Institut Polytechnique de Bordeaux

{mercier,tessier,jeannot}@inria.fr



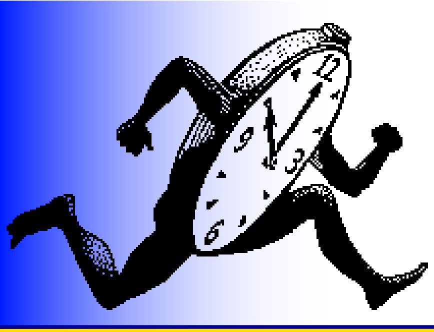
Outline

- Motivation
- Method
- Experimental Results
- Conclusion
- Future Works



Motivation

- **Multicore clusters are *heterogenous* architectures, *performance-wise***
  - Memory Hierarchy
  - Numa effects
- **An MPI application features a *communication pattern***
  - MPI processes do not necessarily exchange the same amount of data
  - An MPI process is likely to exchange data with a subset of the whole set of processes
- **A natural idea is to match the application communication pattern to the underlying hardware communication channels**

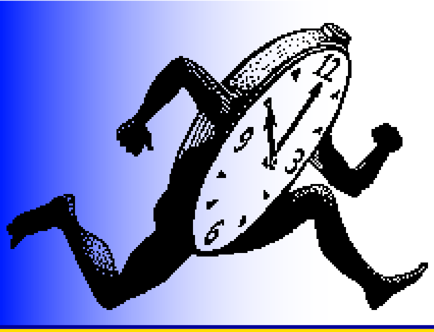


Core Binding Vs Rank Reordering

- **Core Binding: MPI processes are bound to physical cores in order to speed up communication**
    - No need to modify an existing MPI application
    - The user needs to understand the underlying hardware
    - Not portable
        - Process manager options might vary (if existing)
        - Numactl-like command
        - Difficult to change the binding during the execution
- **Rank Reordering: a new MPI communicator is created and the ranks are reorganized**
    - Legacy MPI applications have to be modified
    - No need understand hardware details
    - Portable : relies on standard MPI routines
        - Possibility to perform a reordering during the application execution
- **Both techniques yield the same performance improvements**

A 3-steps Method

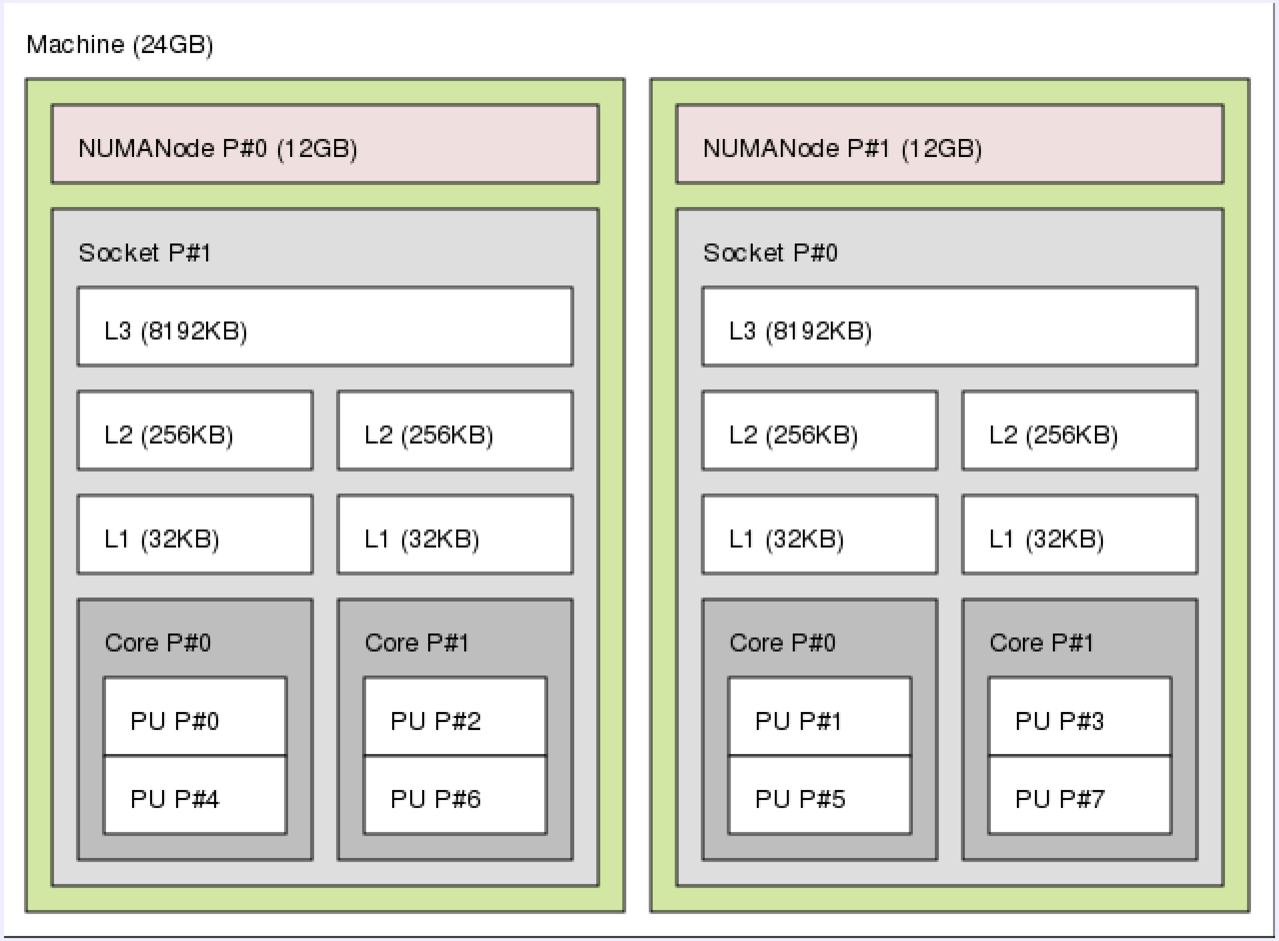
- Step 1
  - Gathering the hardware information (a graph)
- Step 2
  - Gathering the communication pattern (a graph)
- Step 3
  - Using an algorithm to solve the corresponding *graph embedding problem*
- Previously used for network topologies
  - We focus on the nodes architecture



Gathering the hardware information

- **Use of the HWLOC library**
  - Developed in our group
  - No other portable tool available
  - Hwloc data structures fit our needs
- **Two approches :**
  - Centralized
    - Independent from the host file
  - Partially Distributed
    - The host file matters

HWLOC's view of a Node



Machine (24GB)

NUMANode P#0 (12GB)

Socket P#1

L3 (8192KB)

L2 (256KB)

L2 (256KB)

L1 (32KB)

L1 (32KB)

Core P#0

PU P#0

PU P#4

Core P#1

PU P#2

PU P#6

NUMANode P#1 (12GB)

Socket P#0

L3 (8192KB)

L2 (256KB)

L2 (256KB)

L1 (32KB)

L1 (32KB)

Core P#0

PU P#1

PU P#5

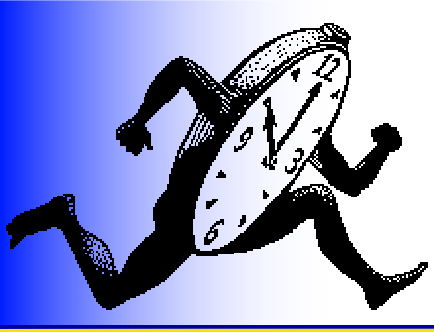
Core P#1

PU P#3

PU P#7

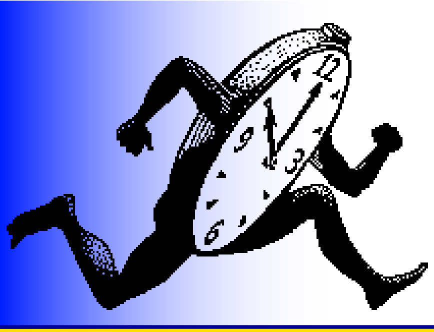
Gathering the Communication Pattern

- A prior run of the application is needed
  - The pattern might change
  - Issues with dynamic algorithms for collectives
- Use of a modified version of MPI
  - Need(ed) to know the amount of intra/internode communication
  - Possibility to trace collectives
    - Not fully MPI implementation-independent



Matching both Informations

- **The hardware topology is a tree**
- **The communication pattern is a random graph**
    - We extract a tree-structure graph from the communication matrix
- **We developed a dedicated matching algorithm called *TreeMatch***
    - Affinity metrics
        - Amount of data exchanged (Data Size)
        - Number of messages (Msg Num)
    - Could use other metrics
        - Energy Consumption



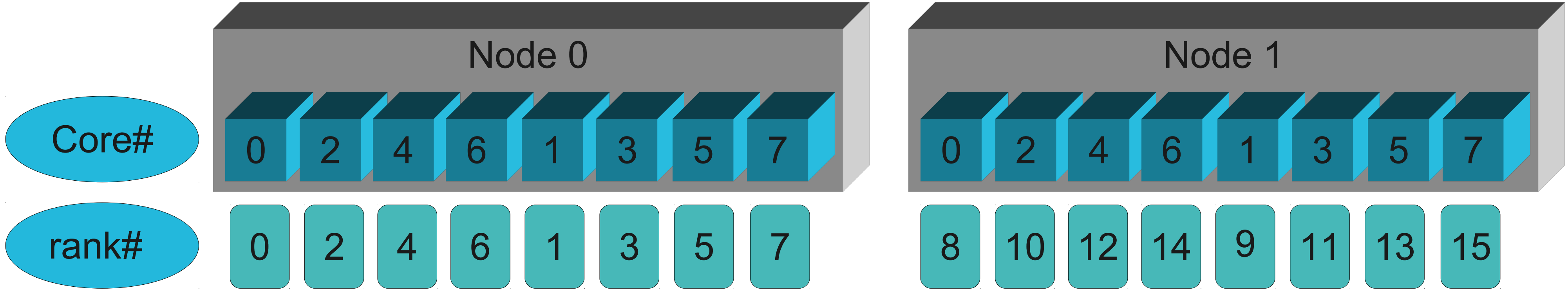
An enhanced implementation of MPI_Dist_Graph_create

- **A new function in the standard (MPI 2.2)**
  - Addresses scalability issues in the interface
  - Features a parameter that triggers reordering
- **Modified implementations of MPI-2**
  - MPICH2-Nemesis
  - MVAPICH2
  - Open MPI
- **Reordering computed at the begining**
  - Done only once
  - No need to move data between processes

Experimental Setup

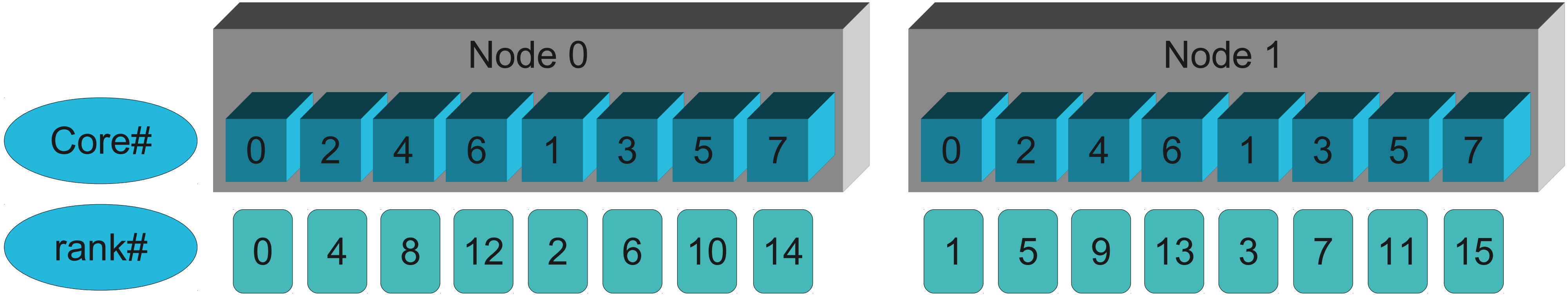
- Tests made with 64 processes
  - 8 nodes featuring 8 cores
- All nodes are connected to a single IB switch
  - The Network topology is not taken into account (flat vision)
- Two real applications shown
  - RSA-768
  - ZeusMP/2

Placement policies



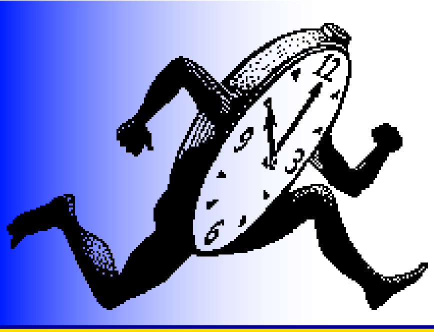
	Node 0								Node 1							
Core#	0	2	4	6	1	3	5	7	0	2	4	6	1	3	5	7
rank#	0	2	4	6	1	3	5	7	8	10	12	14	9	11	13	15

Serial Ranking Policy



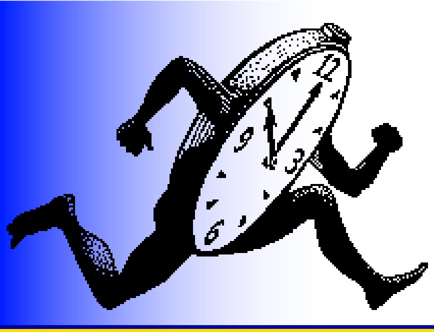
	Node 0								Node 1							
Core#	0	2	4	6	1	3	5	7	0	2	4	6	1	3	5	7
rank#	0	4	8	12	2	6	10	14	1	5	9	13	3	7	11	15

Round Robin Policy

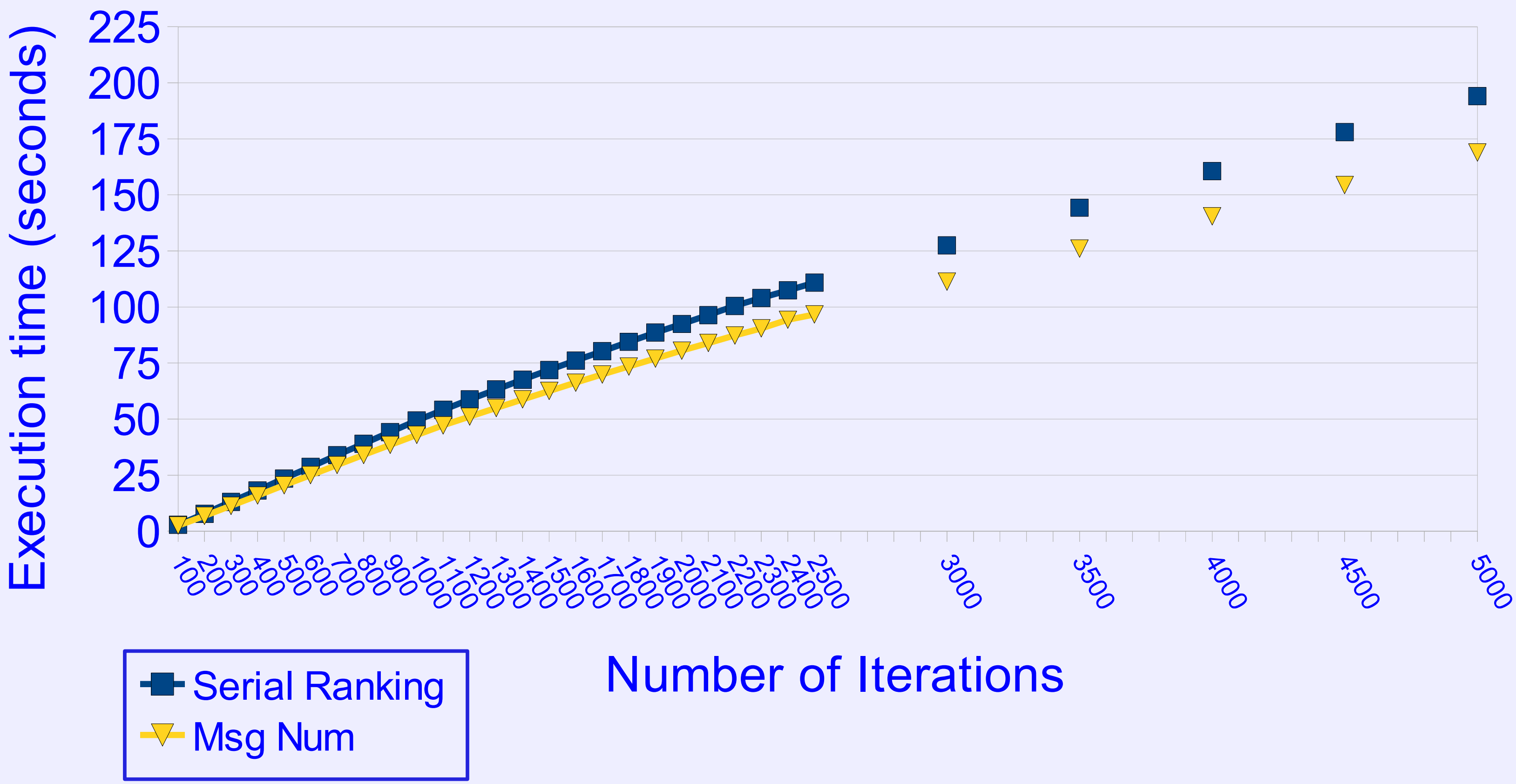


Real Applications

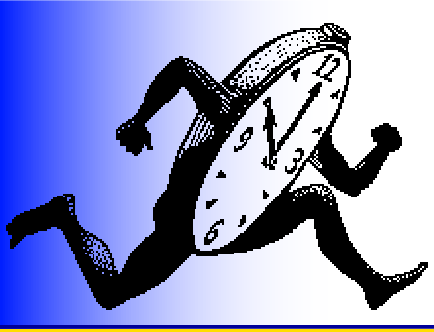
- Zeus-MP
  - Computational Fluid Dynamics application
  - Not optimized for the hardware
- RSA-768
  - Block Wiedemann step
  - Communication-bound application
  - Underlying hardware taken into account



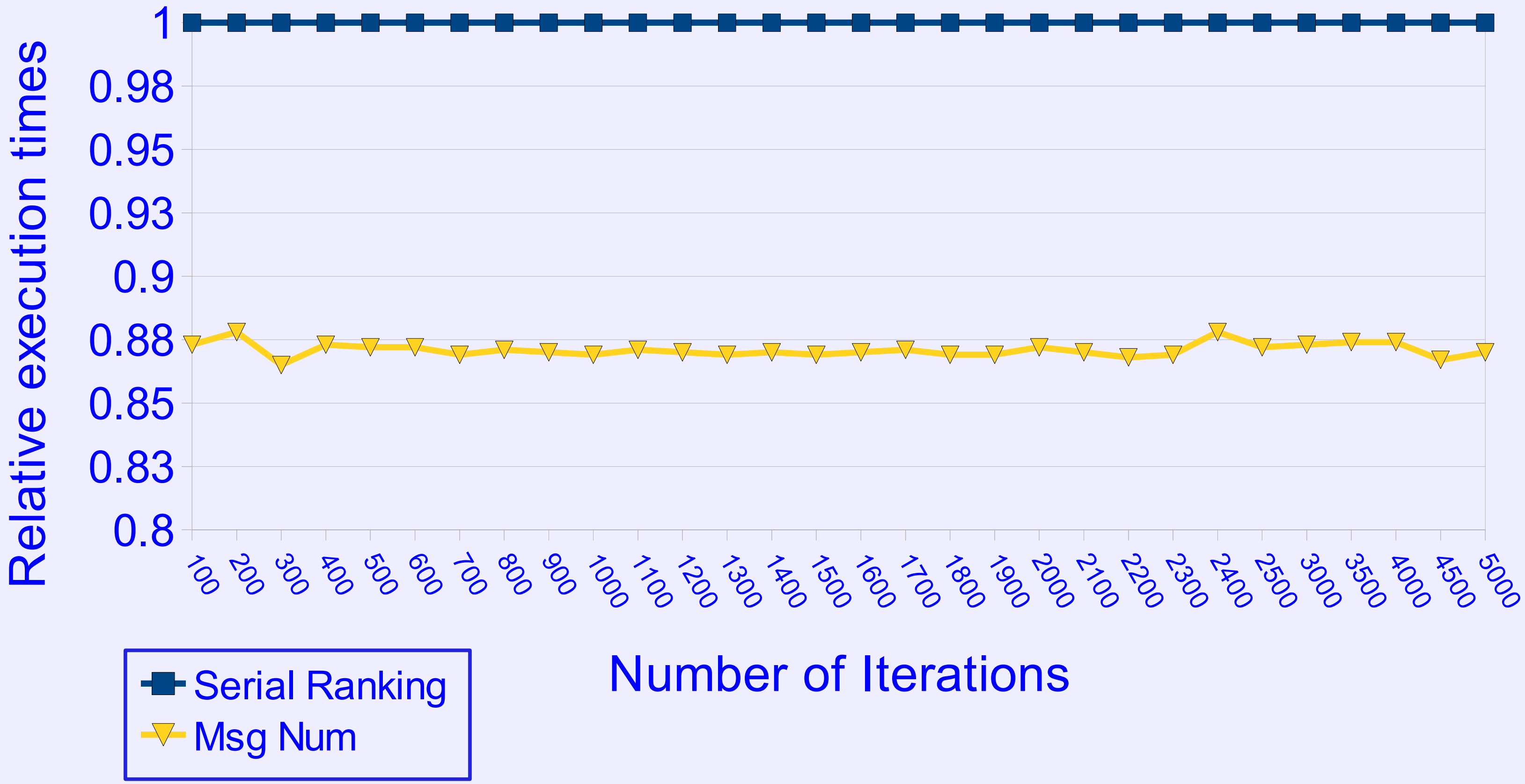
Experimental Results



Zeus-MP Results

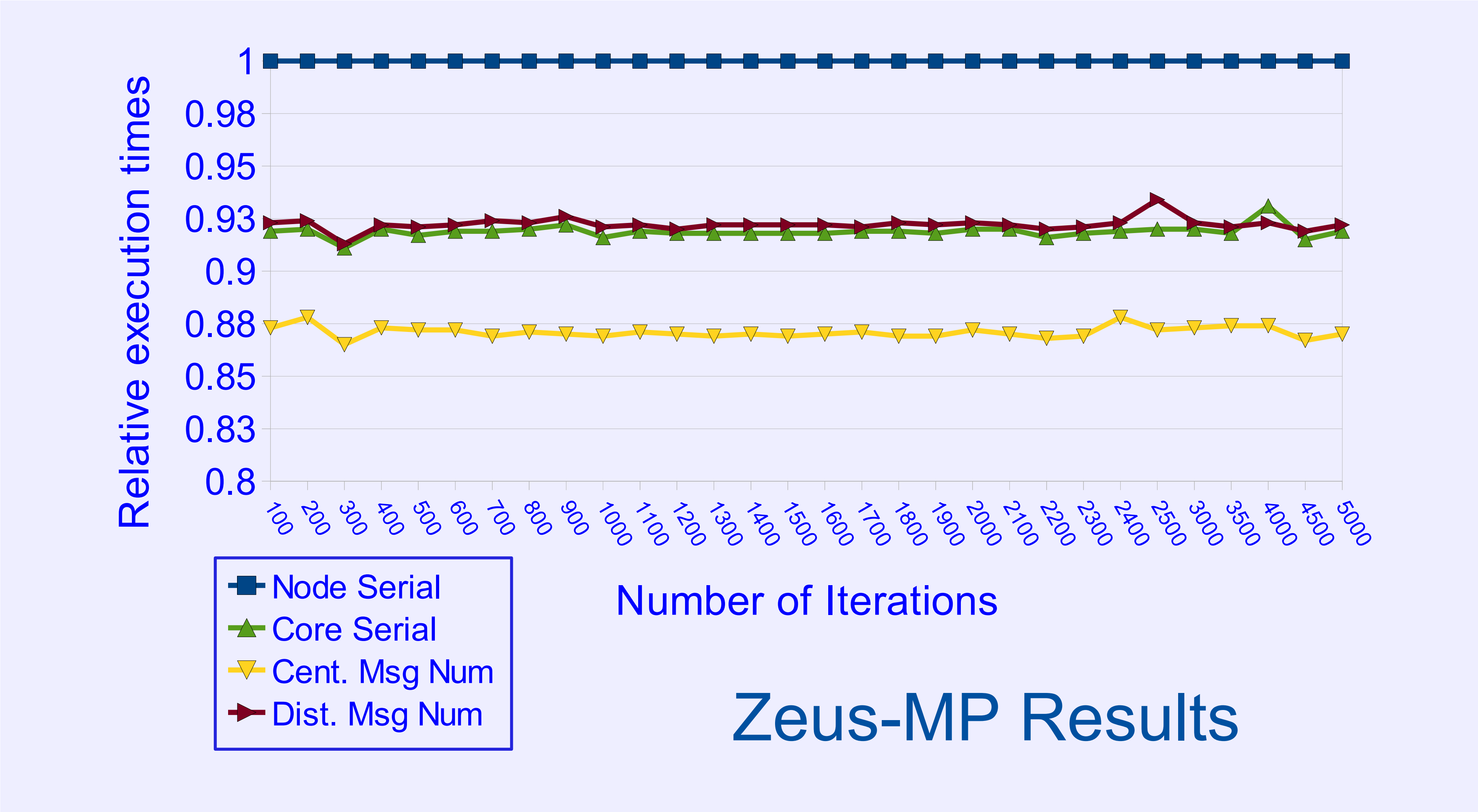


Experimental Results

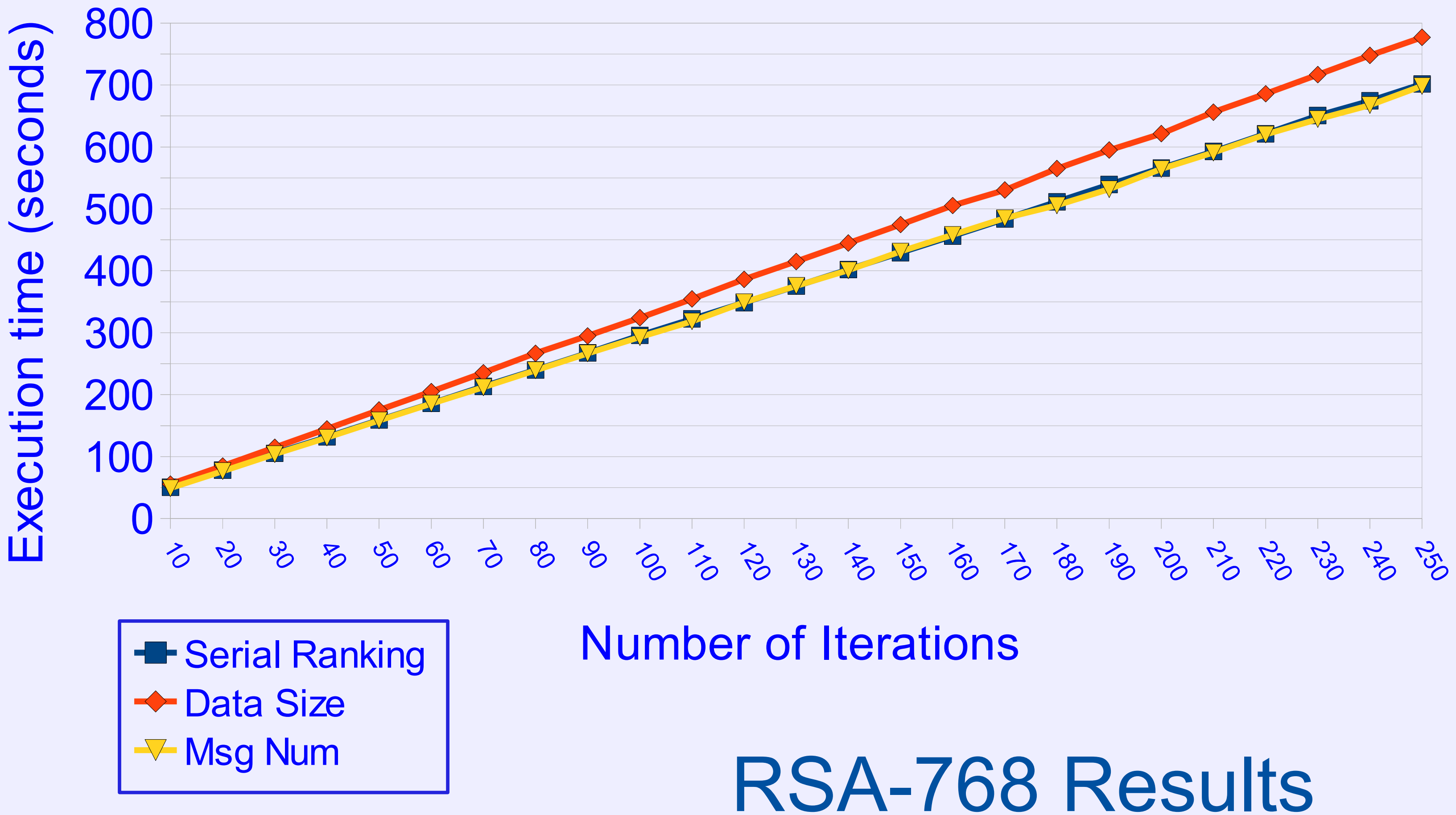


Zeus-MP Results

Zeus-MP Results : Partially Distributed version

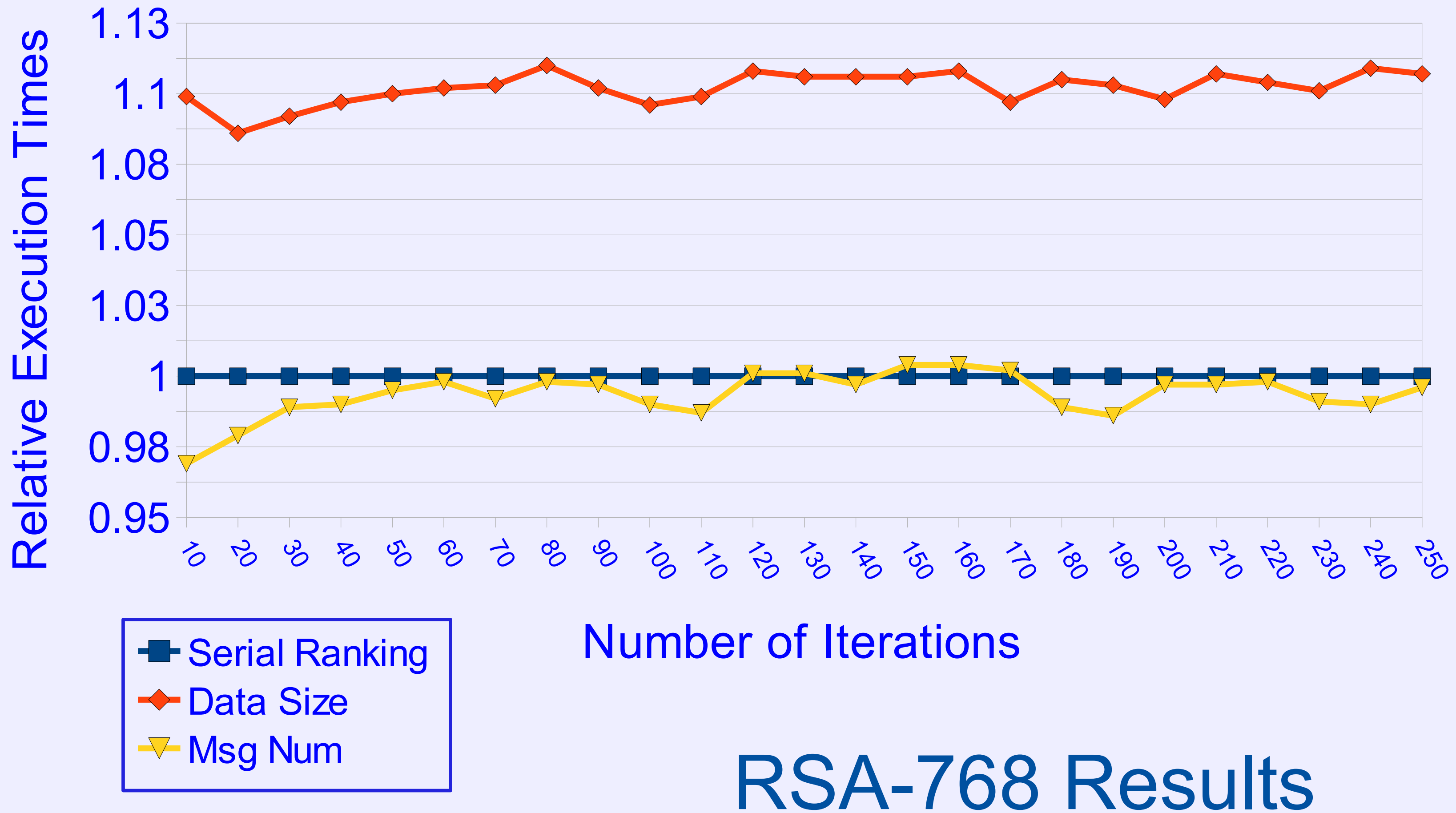


Experimental Results



RSA-768 Results

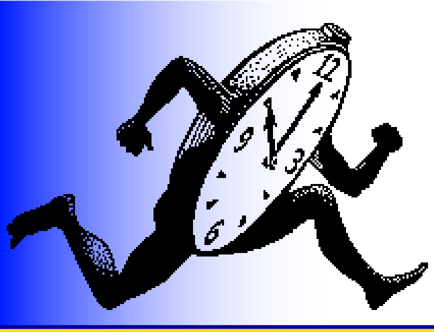
Experimental Results



RSA-768 Results

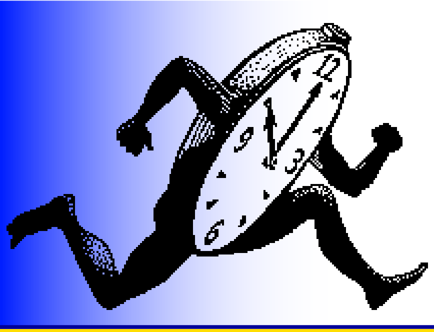
Conclusion

- **Rank reordering allows the programmer to exploit the underlying architecture**
    - Transparently
        - No need to understand the hardware
    - In a portable fashion
        - The routines are part of the MPI standard
        - No need to dwelve into PM options
- **Rank reordering *may* lead to improvements**
    - Depends on how the application is written
    - But using it should not degrade performance either...
        - Depends on the metric used !



TreeMatch Improvements

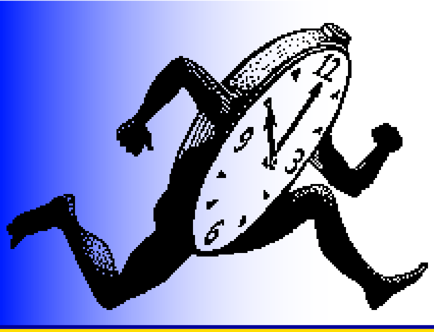
- Usable models
  - Finer Hardware models for NUMA nodes
    - New and more relevant metrics
  - Network models (topology, characteristics, etc.)
    - Complimentary works (e.g. LibTopoMap, etc.)
- Information (e.g comm patterns)
  - Static analysis
  - Simulation



TreeMatch Integration

TreeMatch could also be used in :

- All remaining topology routines
- Collective communications
- Parallel I/O
  - File accesses patterns
- Fault Tolerance
  - Hierarchical protocol
  - Need to create groups
  - Group affinity based on message logging



Future Works

What we plan to address :

- Understand metrics
- Our work focuses on the internal structure of the nodes
    - Need to integrate the network topology
- We want to take into account more information
    - Eg : Numa effects
- We need to implement a distributed version
    - Partially distributed version available