Mapping Asynchronous Iterative Applications on Heterogeneous Distributed Architectures

Raphaël Couturier and David Laiymani and **Sébastien Miquée**

2010/04/23

AND Team
LIFC

University of
Franche-Comté

Outline

Outline

Parallel computing
Solving methods

Objectives

Solving large scale numerical problems

- **Direct methods**: solve the problem in a finite operations number
- **Iterative methods**: solve the problem by successive iterations and give the desired result's approximation

Solving methods
Why choosing iterative methods?

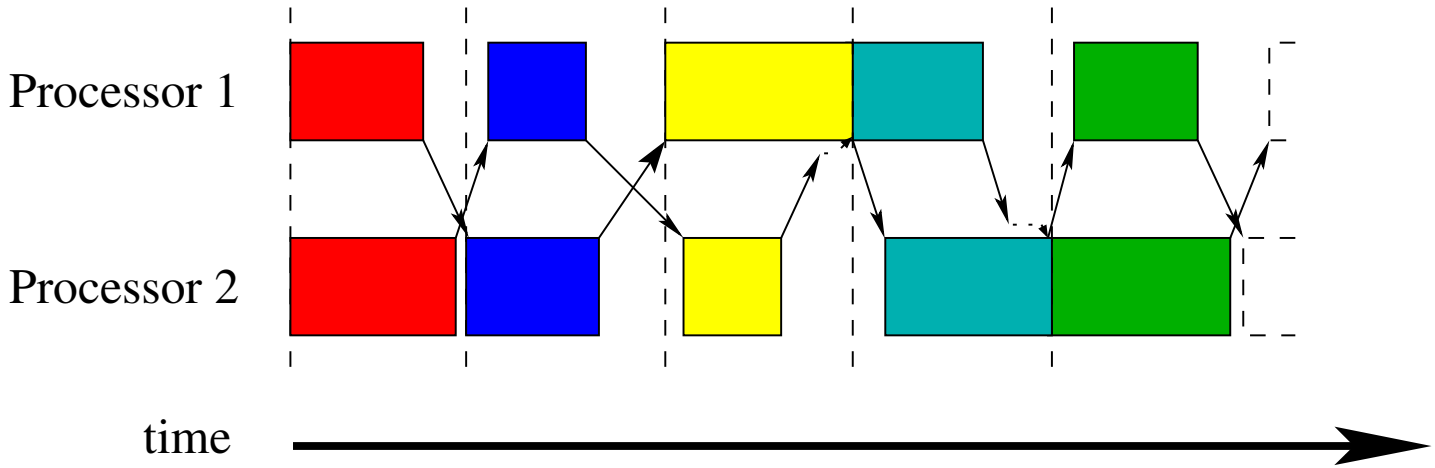
Advantages

- Some problems could not be solved by direct methods
- More simple to parallelize
- Could solve problems of a larger size

Main drawback

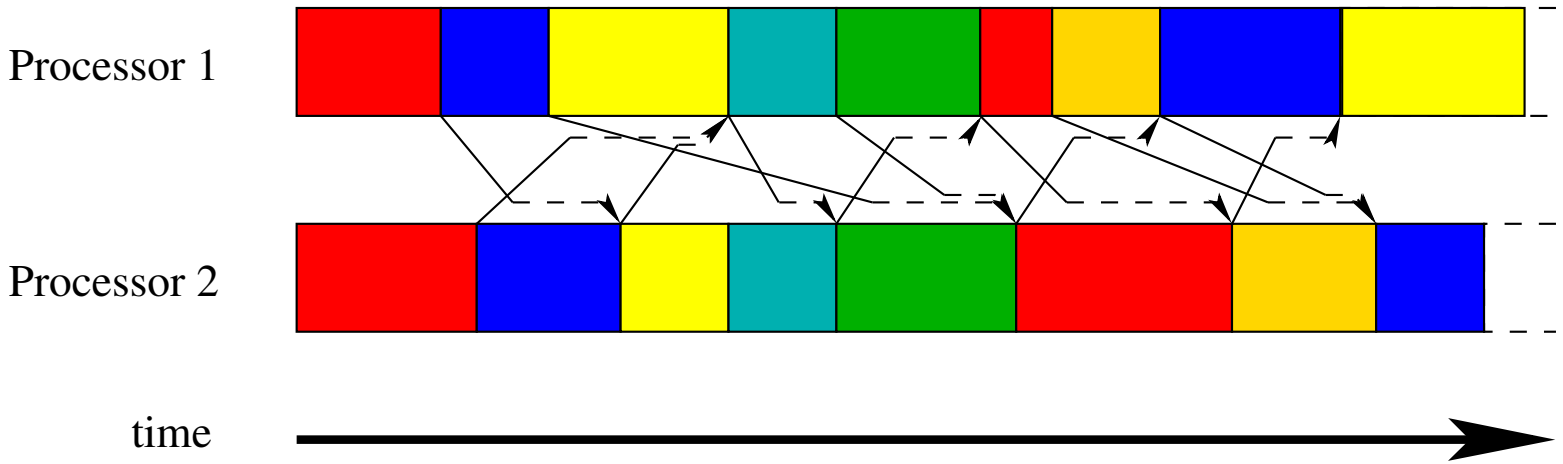
- Convergence study

Iterative computing
The synchronous model



- Iterations synchronizations
- Communications synchronizations (eventually)

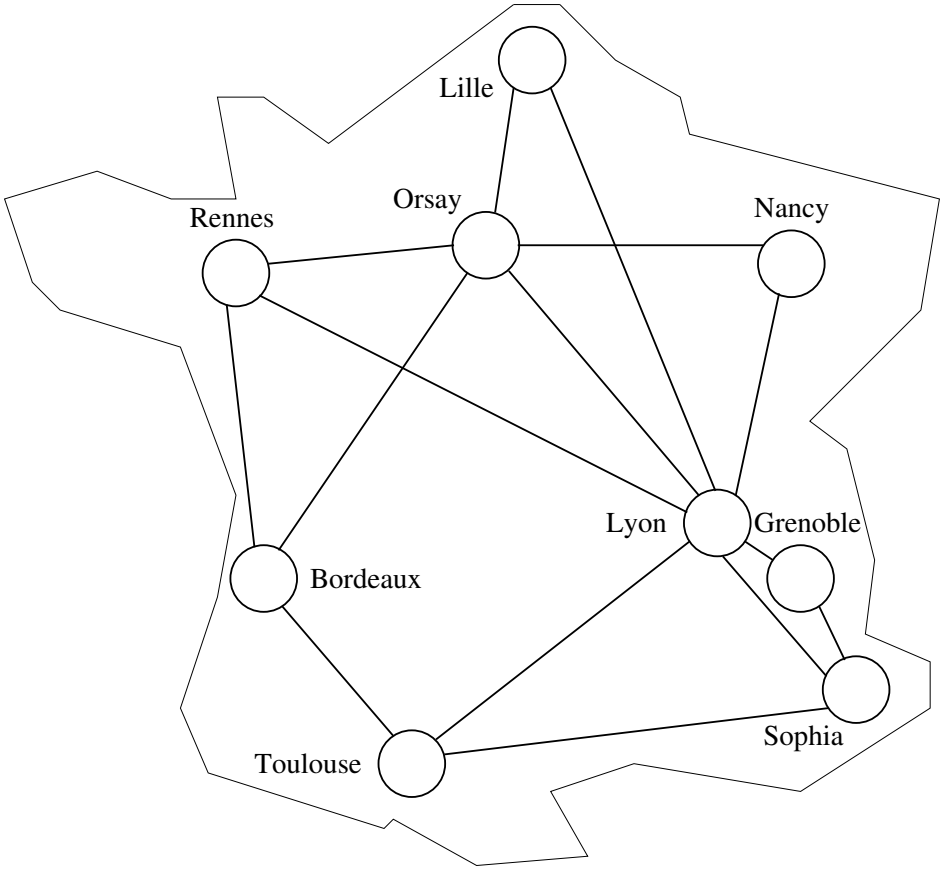
Iterative computing
The asynchronous model – AIAC



- No synchronization between iterations nor communications
- More iterations than synchronous model
- Not all algorithms can be transformed in asynchronous form

Computing environments

- Clusters
- Distributed clusters
- Computing grids



Outline

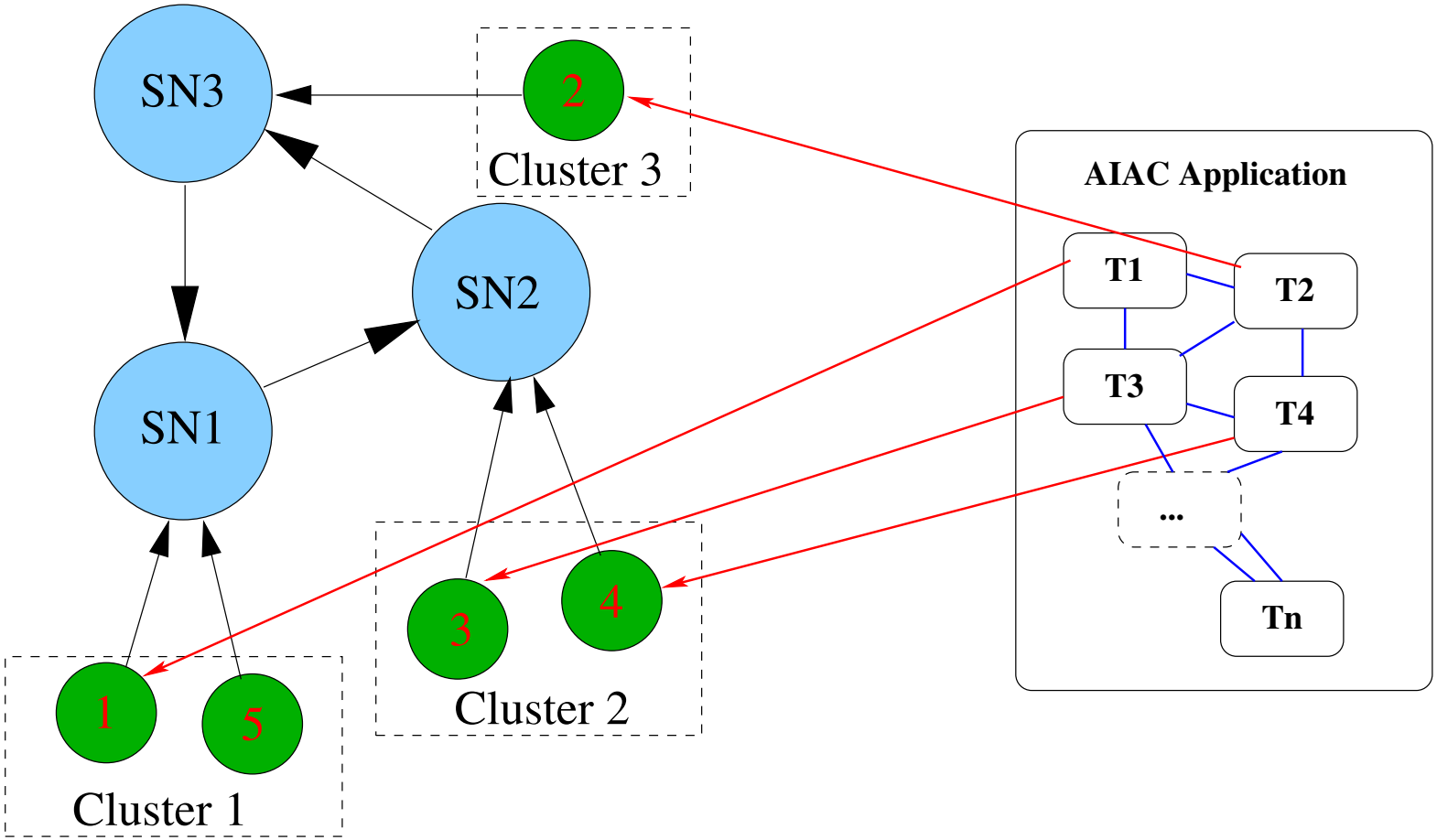
The JaceP2P-V2 platform

- Execution platform and programming environment
  - Platform: daemons, super-nodes, spawners
  - Functions library
- Designed for AIAC model
- Fully fault tolerant

- "Multi-threaded"
- Java language (portability)
- AND Team (*Jean-Claude Charr*)

The JaceP2P-V2 platform

Default "mapping"



Mapping relevance

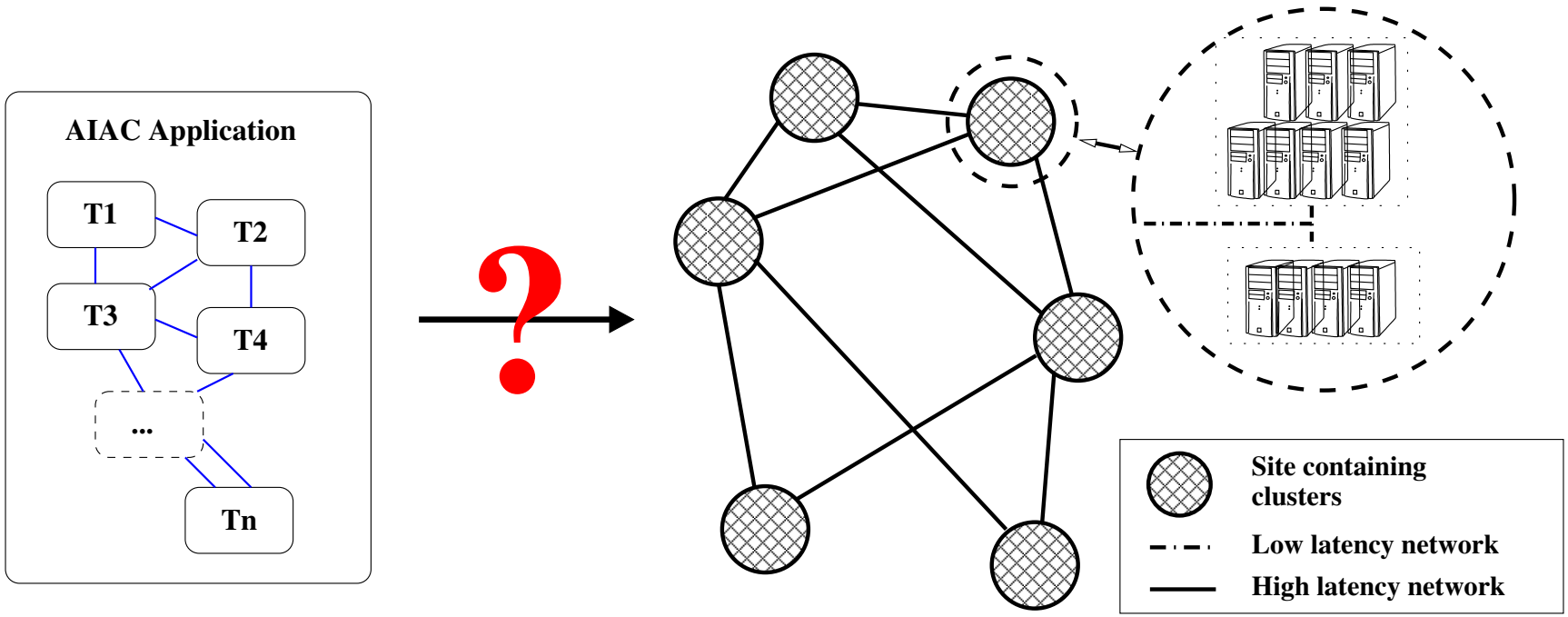
Conditions

- Grid'5000 platform – 200 computing nodes
- Simple Mapping algorithm (SMa)
- Application using the Multisplitting CG
  - Problems sizes : 550'000 and 5'000'000

Results – Gains in execution time

- For 550'000: 30% faster than without mapping algorithm
- For 5'000'000: 40% faster than without mapping algorithm

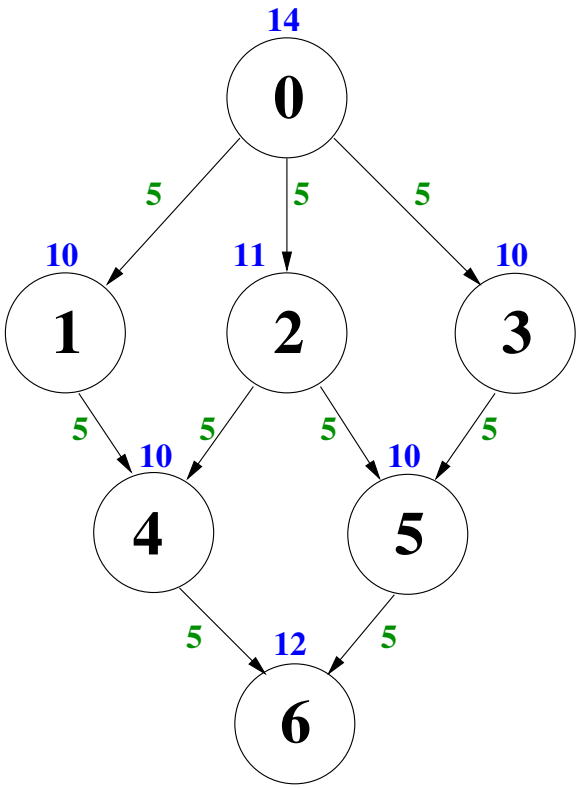
Problematic



Outline

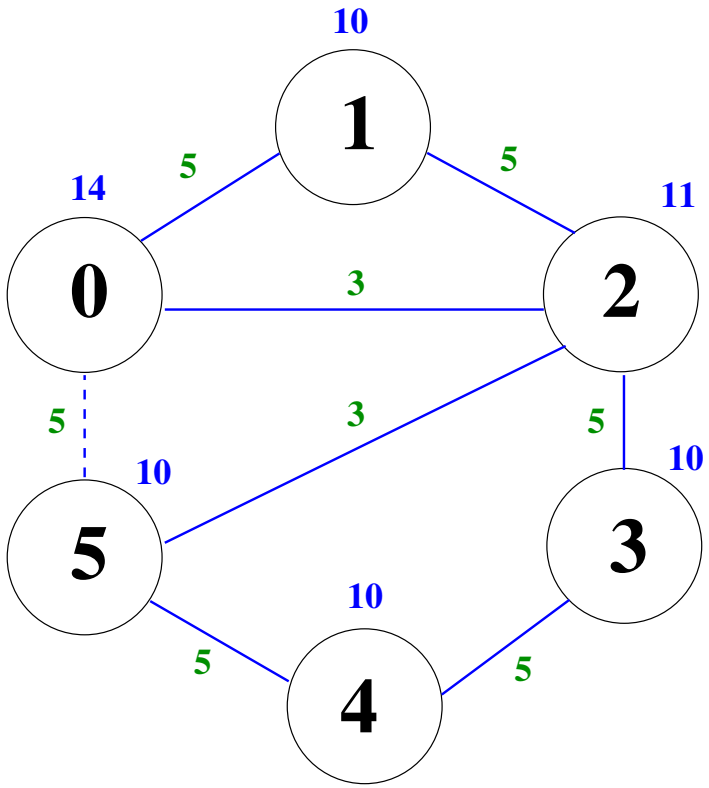
Application modeling

(fit with synchronous model)



DAG

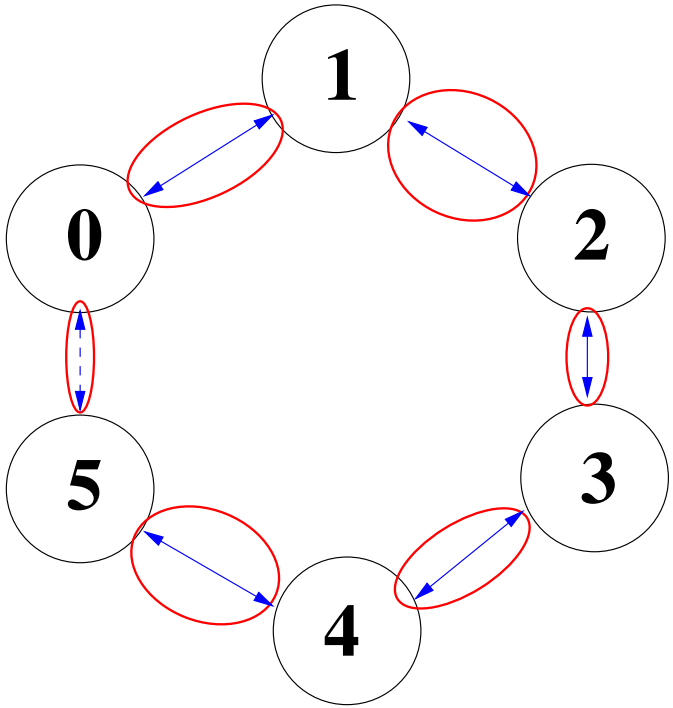
(fit with asynchronous model)



TIG

Existing mapping algorithms

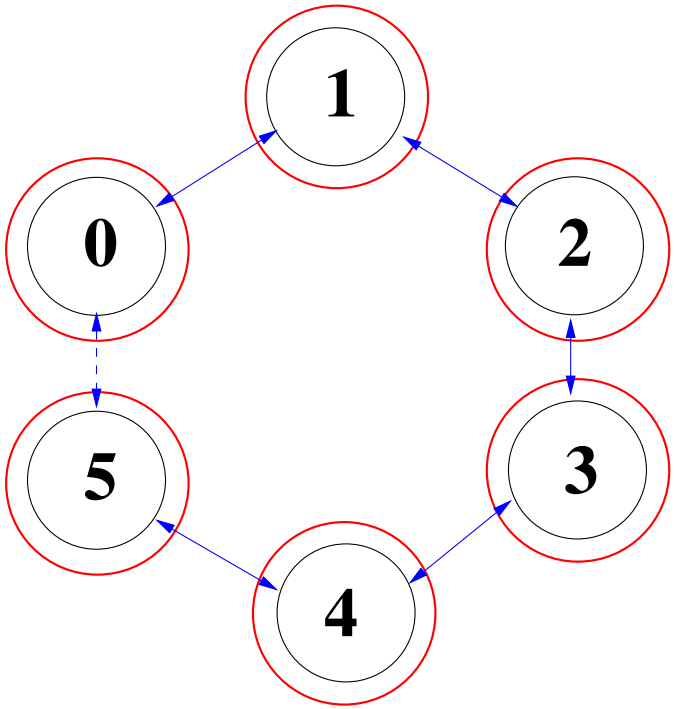
Minimization of external links – *edgecuts*



- Goal: avoiding penalizing communications
- Algorithms: Metis[1], Chaco[2]...

Existing mapping algorithms

Minimization of execution time



- Goal: reducing the application execution time
- Algorithms : QM[3], FastMap[4], MiniMax[5]...

Specificities of the AIAC model and JaceP2P-V2

Specificities due to the AIAC model and JaceP2P-V2

- Only one task per computing node
- *No task precedence*

Specificities due to the targeted architecture

- Computing node volatility
- Tasks state save
- Heterogeneity in computing nodes
- Heterogeneity in networks

AIAC-QM algorithm

Quick Quality Map (QM) [3]

- Optimization of the execution time of each task
- Execution time taking care of computation and communication
- Does not satisfy model's constraints

Evaluation of AIAC-QM

- Adaptation and implementation of QM in JaceP2P-V2 $\rightarrow$ AIAC-QM
- Introduction of a little part of *edge-cuts*

AIAC-QM algorithm

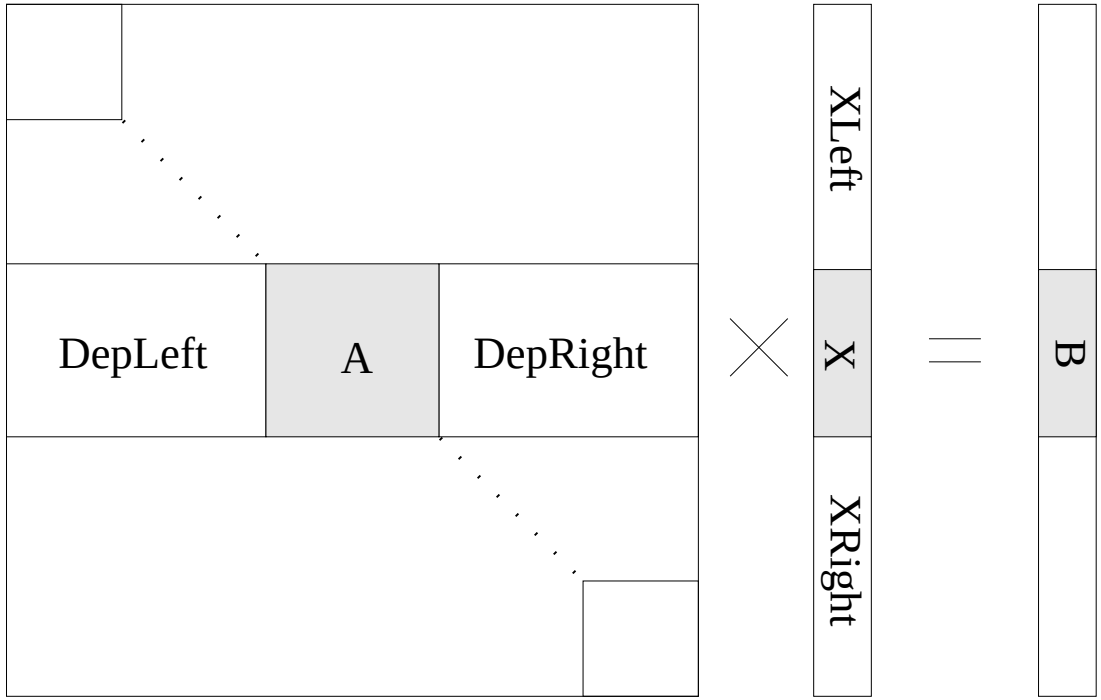
Principles

- Sort computing nodes by computation power
- Iterate on each task for searching a better node
- Limited number of rounds
- Execution time computation based on:
  - Computing node power
  - Locality with task's neighbors

Outline

NPB – Multisplitting Conjugate Gradient

- Nas Parallel Benchmark
- Conjugate Gradient matrix computation
- Asynchronous version using the multisplitting method[6]



Conditions of the evaluations

Application and architectures

- Multisplitting CG (E : 550'000 and F : 5'000'000)
- 64 (for E) and 128 (for F) computing nodes
- Homogeneous architectures
  - Architecture 1: 113 nodes – 440 cores
  - Architecture 2: 213 nodes – 840 cores
- Heterogeneous architectures
  - Architecture 3: 112 nodes – 394 cores
  - Architecture 4: 212 nodes – 754 cores
- No computing node failure

Results 1 – Homogeneous Architectures

Algorithm	No	SMa	AIAC QM	F-EC
Execution time	150s	110s	101s	90s
Gains	–	27%	33%	40%

Table: Multisplitting CG 550'000 – Architecture 1

Algorithm	No	SMa	AIAC QM	F-EC
Execution time	403s	265s	250s	218s
Gains	–	34%	38%	46%

Table: Multisplitting CG 5'000'000 – Architecture 2

Results 2 – Heterogeneous Architectures

Algorithm	No	SMa	AIAC QM	F-EC
Execution time	498s	341s	273s	385s
Gains	–	32%	45%	23%

Table: Multisplitting CG 550'000 – Architecture 3

Algorithm	No	SMa	AIAC QM	F-EC
Execution time	943s	594s	453s	660s
Gains	–	37%	52%	30%

Table: Multisplitting CG 5'000'000 – Architecture 4

Outline

Conclusion & Future works

Conclusion

- Mapping is essential for JaceP2P-V2
- Adaptation of existing mapping algorithms
- Considerable gains on application execution time

Future works

- Designing a hybrid algorithm
- Mapping tasks' saves
- Take care about nodes capacities
- Fault tolerance in mapping algorithms

References I


G. Karypis and V. Kumar.
A fast and high quality multilevel scheme for partioning irregular graphs.
SIAM Journal on Scientific Computing, 20(1):359–392, 1998.


B. Hendrickson and R. W. Leland.
The Chaco User's Guide.
Sandia National Laboratory, Albuquerque, 1995.

References II


P. Phinjaroenphan.
An Efficient, Pratical, Portable Mapping Technique on Computational Grids.
PhD thesis, School of Computer Science and Information technology Science, Engineering and Technology Portfolio, RMIT University, 2006.


S. Sanyal, A. Jain, S. K. Das, and Rupak Biswas.
A hierarchical and distributed approach for mapping large applications to heterogeneous grids using genetic algorithms.
In *CLUSTER*, pages 496–499, 2003.

References III


S. Kumar, S. K. Das, and Rupak Biswas.
Graph partitioning for parallel applications in heterogeneous grid environments.
In *IPDPS*, 2002.


J. Bahi, S. Contassot-Vivier, and R. Couturier.
Parallel Iterative Algorithms: from Sequential to Grid Computing, volume 1 of *Numerical Analysis & Scientific Computating*, chapter Asynchronous Iterations, pages 124–131.

Chapman & Hall/CRC, 2007.