Programming BSP and Multi-BSP algorithms in ML

Victor ALLOMBERT[1], Frédéric GAVA[2]

[1]LIFO - Université d'Orléans, France
[2]LACL - Université Paris Est, France

13 July 2018

Table of Contents

Table of Contents

The world of parallel computing

Simulations:

Fluid simulation
3D Visualisation

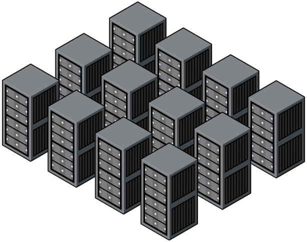
Big-Data:

IoT
Social Networking
Data science

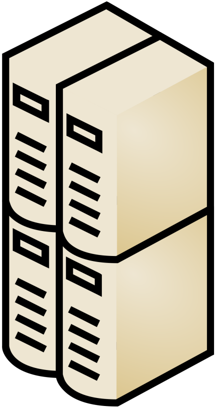
Symbolic computation:

Model-Checking
Formal computing

Super-computer

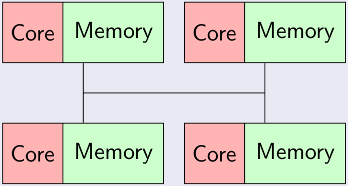


Distributed computing



Characterised by:

- Interconnected units
- Distributed memory
- Communication network
- MPI



Bulk Synchronous Parallelism

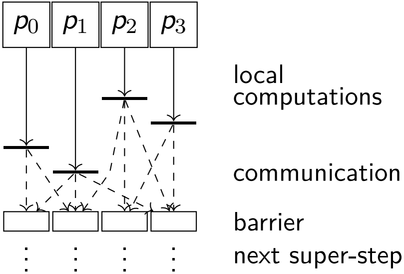
The BSP computer

Defined by:

- $\mathbf{p}$ pairs CPU/memory
- Communication network
- Synchronisation unit
- Super-steps execution

Properties:

- Deadlock-free
- Predictable performances



What is BSML?

- Explicit BSP programming with a functional approach

What is BSML?

- Explicit BSP programming with a functional approach
- Based upon ML and implemented over OCAML

What is BSML?

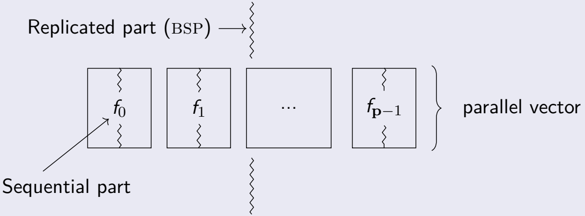
- Explicit BSP programming with a functional approach
- Based upon ML and implemented over OCAML
- Formal semantics → computer-assisted proofs (COQ)

What is BSML?

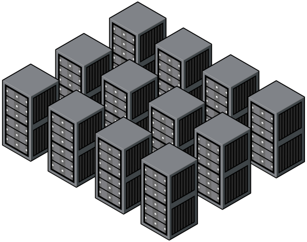
- Explicit BSP programming with a functional approach
- Based upon ML and implemented over OCAML
- Formal semantics → computer-assisted proofs (COQ)

Main idea

Parallel data structure ⇒ *parallel vector*:

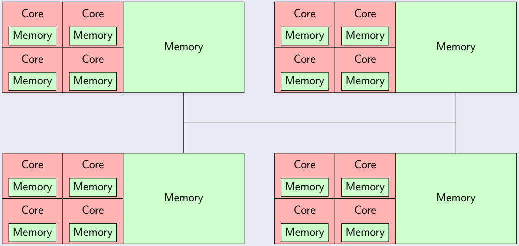


Hierarchical architectures

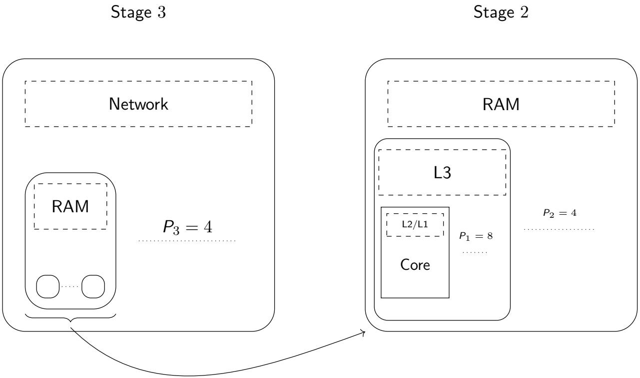


Characterised by:

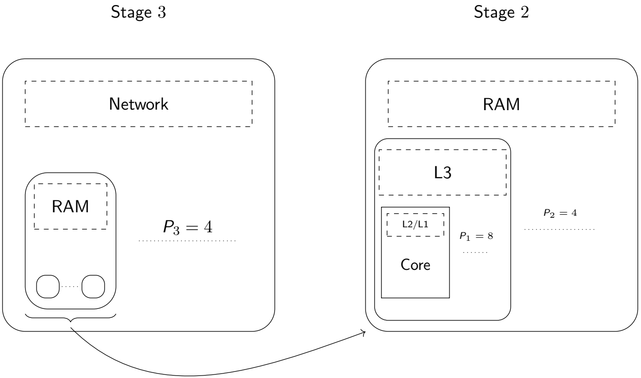
- Interconnected units
- Both shared and distributed memories
- Hierarchical memories



1. A tree structure with nested components
2. Where nodes have a storage capacity
3. And leaves are processors
4. With sub-synchronisation capabilities

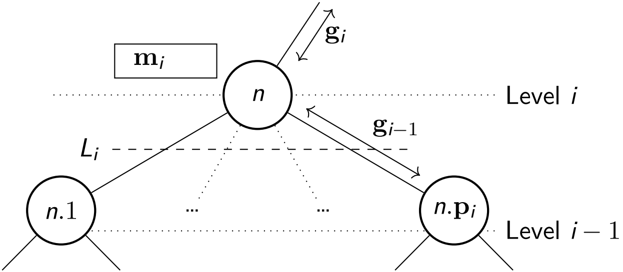


- Stage 3: 4 nodes with a network access
- Stage 2: one node has 4 chips plus RAM
- Stage 1: one chip has 8 cores plus L3 cache
- Stage 0: one core with L1/L2 caches



Execution model

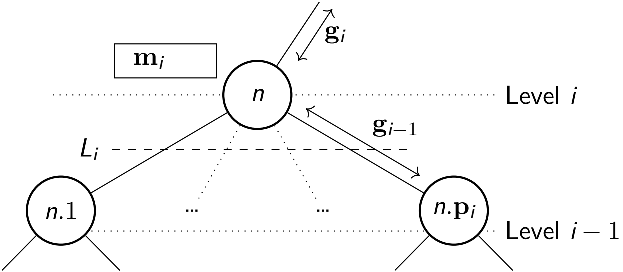
A level i superstep is:



Execution model

A level i superstep is:

- Level $i-1$ executes code independently

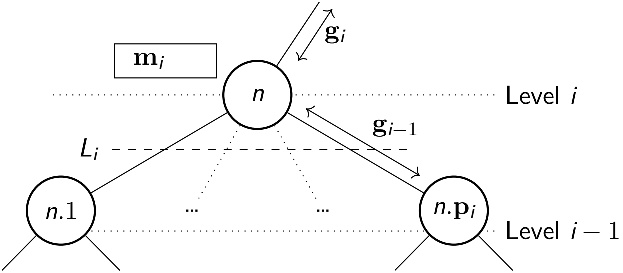


The MULTI-BSP model

Execution model

A level i superstep is:

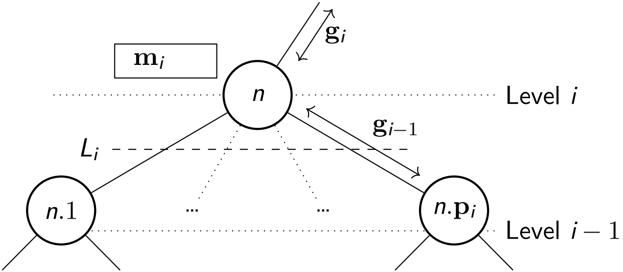
- Level $i - 1$ executes code independently
- Exchanges information with the m_i memory



Execution model

A level i superstep is:

- Level $i-1$ executes code independently
- Exchanges information with the m_i memory
- Synchronises

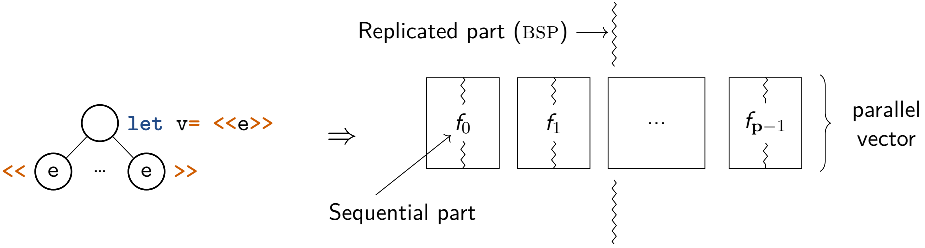


Basic ideas

The MULTI-ML language

Basic ideas

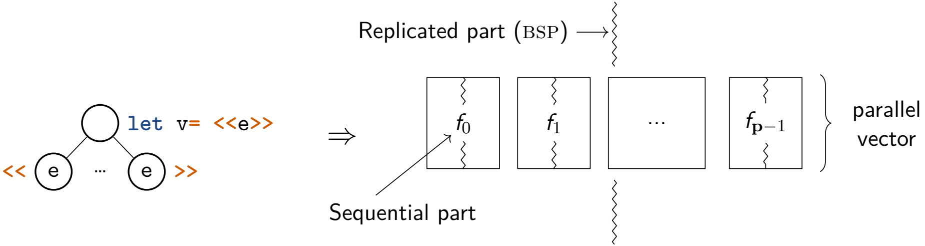
- BSML-like code on every stage of the MULTI-BSP architecture



The MULTI-ML language

Basic ideas

- BSML-like code on every stage of the MULTI-BSP architecture
- Specific syntax over ML: eases programming



The MULTI-ML language

Basic ideas

- BSML-like code on every stage of the MULTI-BSP architecture
- Specific syntax over ML: eases programming
- *Multi-functions* that recursively go through the MULTI-BSP tree

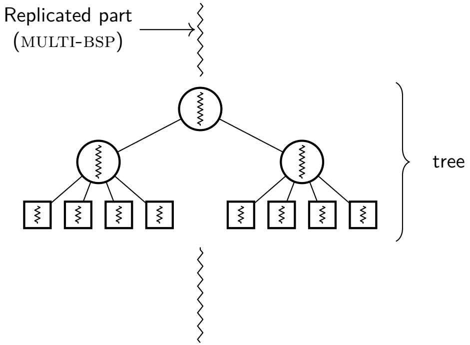


Table of Contents

How to compare ? I

"Incremental programming"

1. Write a sequential OCAML code
2. BSP extension using BSML
3. MULTI-BSP extension using MULTI-ML

How to compare ? II

Difficulty of writing a code

- Halsted Difficulty (HD)
- Halsted Effort (HE)
- McCabe Cyclomatic Complexity (CC)
- Maintainability Index (MI)

How to compare ? III

Overall performances

- Speedup (based on sequential algorithm)
- Execution time

Execution platforms

- Mirev2:
  - 8 nodes with 2 quad-cores (AMD 2376 at 2.3Ghz)
  - 16GB of memory per node
  - 1Gbit/s network
- Mirev3:
  - 4 nodes with 2 octo-cores (INTEL xeon $E5 - 2650$ at 2.6Ghz)
  - 64GB of memory per node
  - 10Gbit/s network

Application cases

- Model-checking (*Symbolic computation*)
- Skeleton based FFT (*Numerical computation*)
- All Pairs similarity search problem (*Big-data computation*)

Model-checking case

What is it ?

Exhaustively check if a model meets a given specification.

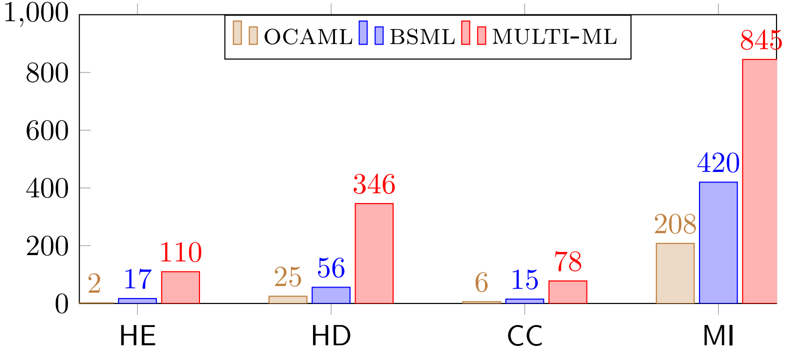
How ?

Exploring all the states accessible via a successor function from an initial state s_0

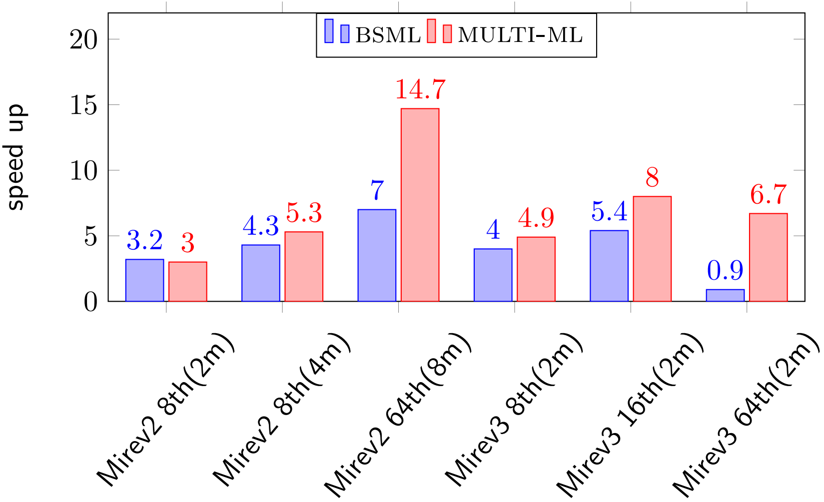
Algorithm characteristics

- Data intensive
- Task parallel
- Explicit load balancing

Difficulty of writing the code



Overall performances



Skeleton based FFT case

What is it ?

Converts a signal from its original domain to a representation in the frequency domain

How ?

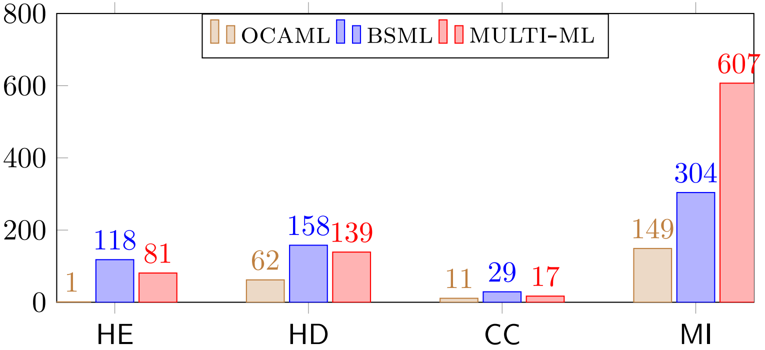
$(FFT\,x)_i = \sum_{k=0}^{n-1} x_k . {e^{2\pi\sqrt{-1}/n}}_n^{ki}$

Expressed using the Distributable Homomorphism (DH) skeleton

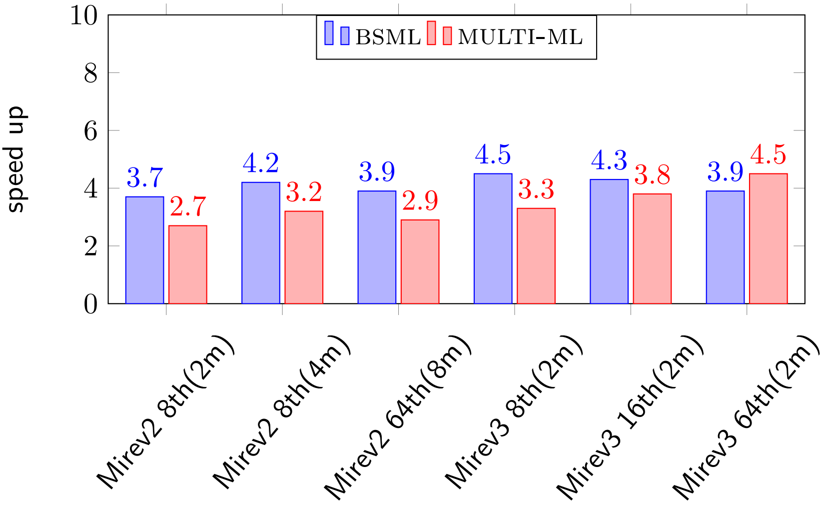
Algorithm characteristics

- Data intensive
- Data parallel
- Balanced communication scheme

Difficulty of writing the code



Overall performances



All Pairs similarity search problem case

What is it ?

Discover all the pairs of objects whose similarity is above a given threshold

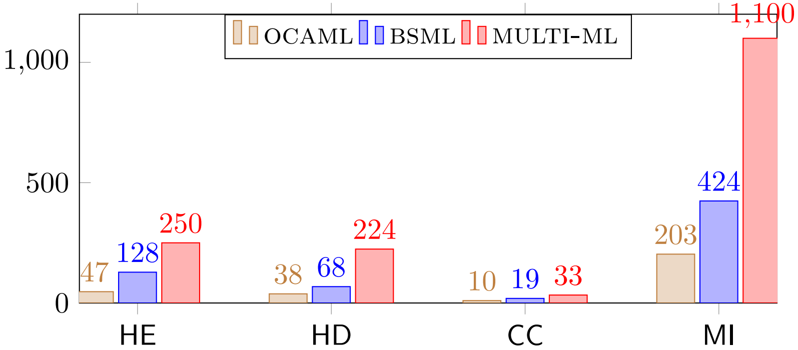
How ?

Compute local similarities of data sub-sets, then exchanges

Algorithm characteristics

- Data intensive
- Data parallel
- Balanced communication scheme

Difficulty of writing the code



Overall performances

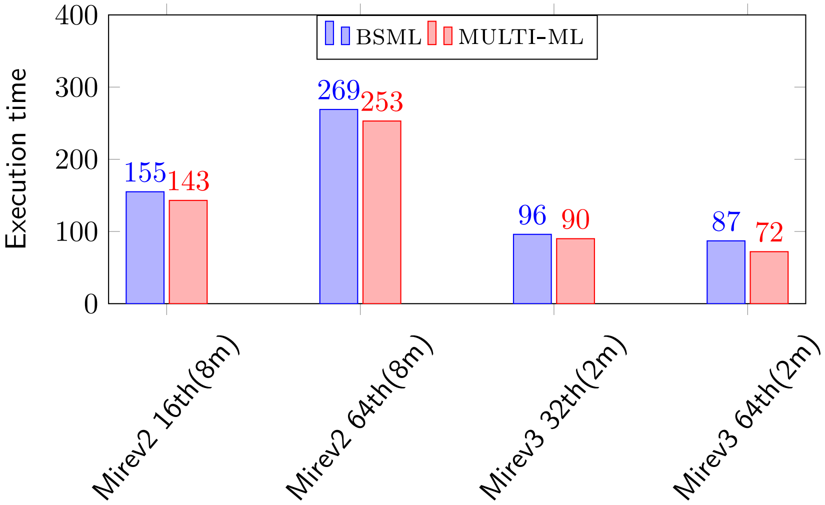


Table of Contents

To sum up

Model-checking

- Code complexity: OCAML $<$ BSML $<$ MULTI-ML
- Overall performances: OCAML $<$ BSML $\leq$ MULTI-ML

Skeleton based FFT

- Code complexity: OCAML $<$ BSML $\simeq$ MULTI-ML
- Overall performances: OCAML $<$ BSML $\simeq$ MULTI-ML

All Pairs similarity search problem

- Code complexity: OCAML $<$ BSML $<$ MULTI-ML
- Overall performances: OCAML $<$ BSML $<$ MULTI-ML

Conclusion

Hierarchical programming: is it worth it ?

\+ Performances

\- Hard to program

-> Harder to write *immortal* algorithms

Ongoing and future work

- Write MULTI-BSP algorithms
- Comparison with BSP and MULTI-BSP cost model
- Programming experiments

Thank you for your attention ☺

Questions ?