Toward performance prediction for Multi-BSP programs in ML

Victor ALLOMBERT[1], Frédéric GAVA[2], Julien TESSON[2]

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

13 December 2018

Table of Contents

Table of Contents

The world of parallel computing

Simulations:

Fluid simulation
3D Visualisation

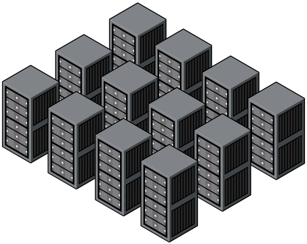
Big-Data:

IoT
Social Networking
Data science

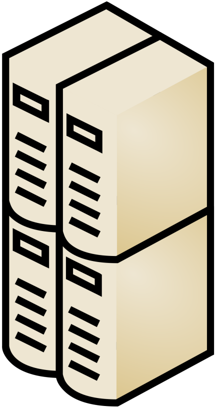
Symbolic computation:

Model-Checking
Formal computing

Super-computer

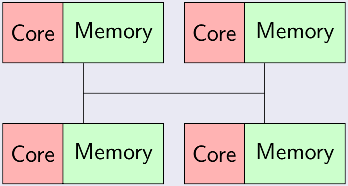


Distributed computing



Characterised by:

- Interconnected units
- Distributed memory
- Communication network
- MPI



Bulk **S**ynchronous **P**arallelism

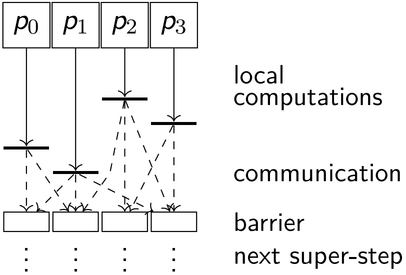
The BSP computer

Defined by:

- **p** pairs CPU/memory
- Communication network
- Synchronisation unit
- Super-steps execution

Properties:

- Confluent
- Deadlock-free
- Predictable performances



Bulk Synchronous Parallelism

BSP cost model

- The number of processors $\mathbf{p}$;
- The time $\mathbf{L}$ required for a barrier;
- The time $\mathbf{g}$ for collectively delivering a 1-relation.

(expressed as multiples the local processing speed r)

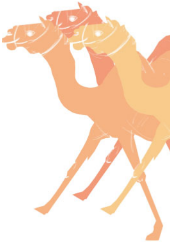
Superstep's cost formulae

$$\mathsf{Cost}(s) = \max_{0 \leq i < \mathbf{p}} w_i^s + \max_{0 \leq i < \mathbf{p}} h_i^s \times \mathbf{g} + \mathbf{L}$$

Bulk Synchronous ML

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

Bulk Synchronous ML

What is BSML?

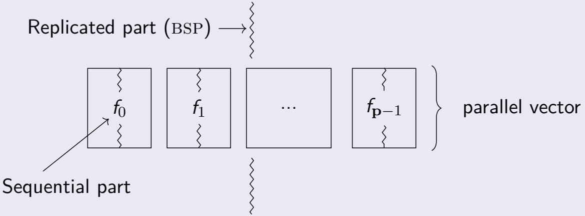
- Explicit BSP programming with a functional approach
- Based upon ML and implemented over OCAML
- Formal semantics → computer-assisted proofs (COQ)

What is BSML?

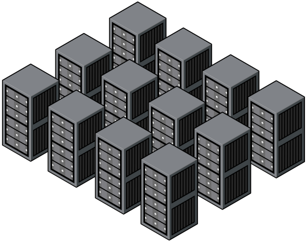
- Explicit BSP programming with a functional approach
- Based upon ML and implemented over OCAML
- Formal semantics → computer-assisted proofs (COQ)

Main idea

Parallel data structure ⇒ *parallel vector*:

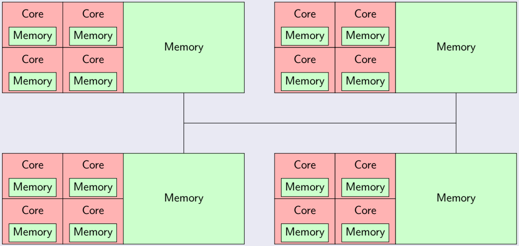


Hierarchical architectures

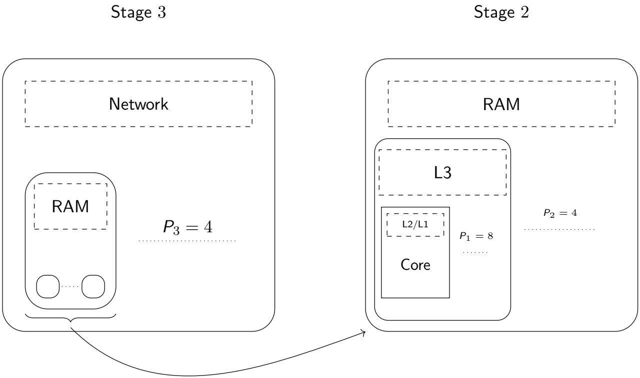


Characterised by:

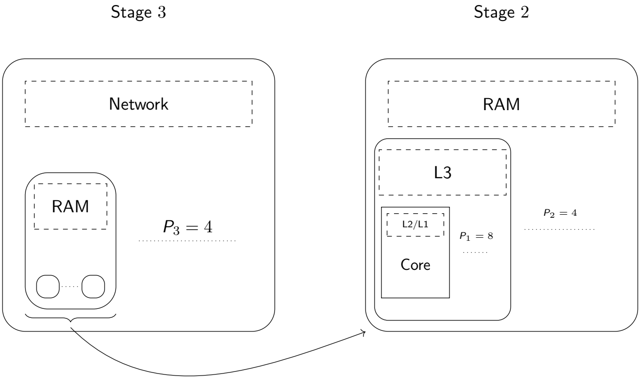
- Interconnected units
- Both shared and distributed memories
- Hierarchical memories



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

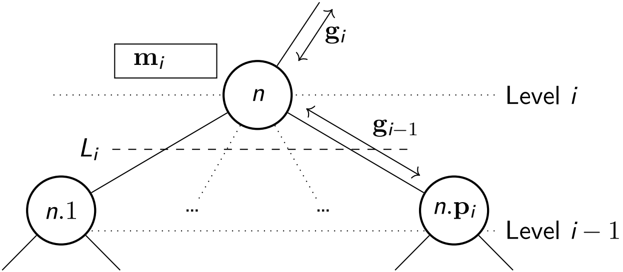


- Stage 3: 4 nodes with a network access
- Stage 2: one node has 4 chips plus RAM
- Stage 1: one chip has 8 cores plus L3 cache
- Stage 0: one core with L1/L2 caches



Execution model

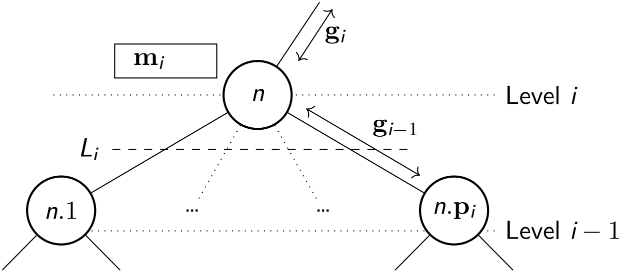
A level i superstep is:



Execution model

A level i superstep is:

- Level $i-1$ executes code independently

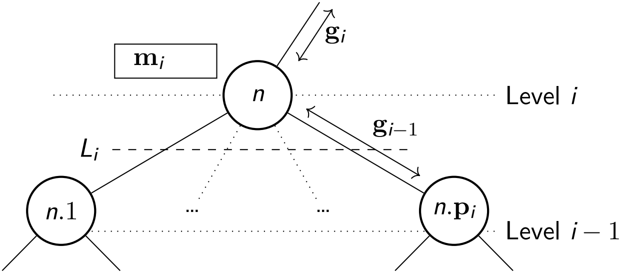


The MULTI-BSP model

Execution model

A level i superstep is:

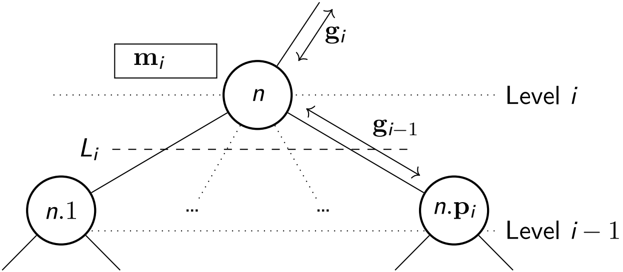
- Level $i-1$ executes code independently
- Exchanges information with the m_i memory



Execution model

A level i superstep is:

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



The MULTI-BSP model

MULTI-BSP cost model

4 parameters for the *d* levels: $(\mathbf{p_i}, \mathbf{g_i}, \mathbf{L_i}, \mathbf{m_i})$

Cost formulae

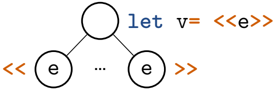
$$T = \sum_{i=0}^{\mathbf{d}-1} \left(\sum_{j=0}^{N_i - 1} w_j^i + C_j^i \right)$$

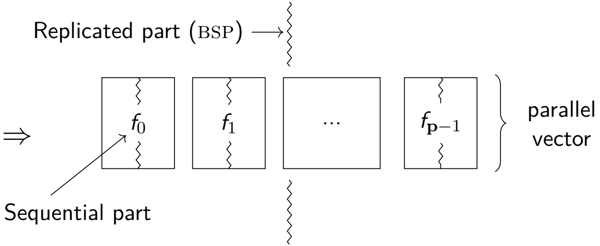
Basic ideas

The MULTI-ML language

Basic ideas

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

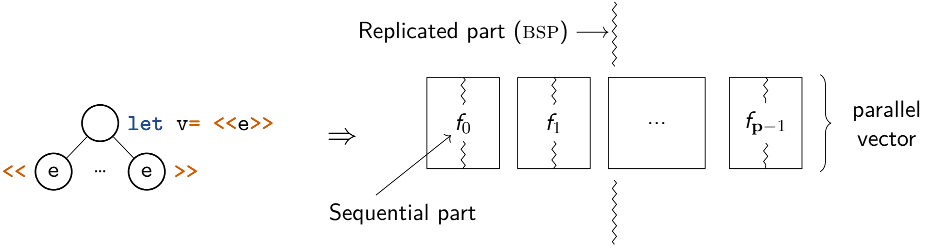




The MULTI-ML language

Basic ideas

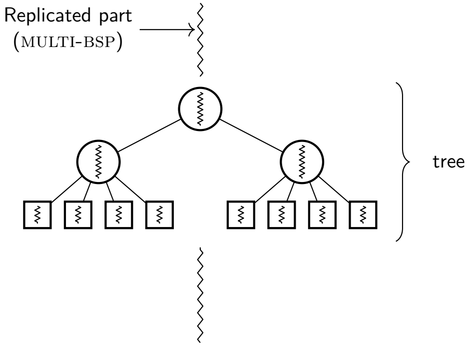
- BSML-like code on every stage of the MULTI-BSP architecture
- Specific syntax over ML: eases programming



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



MULTI-ML: Tree recursion

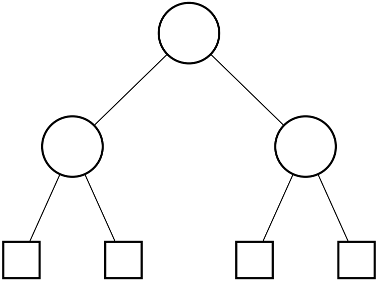
Recursion structure

```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```

MULTI-ML: Tree recursion

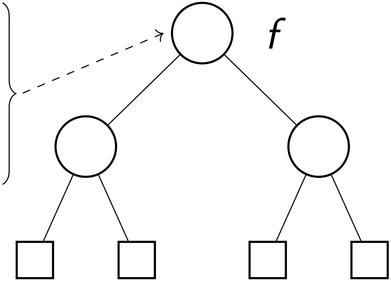
Recursion structure

```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```



Recursion structure

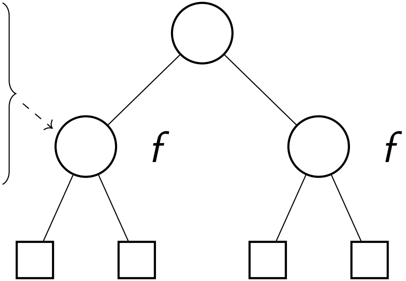
```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```



MULTI-ML: Tree recursion

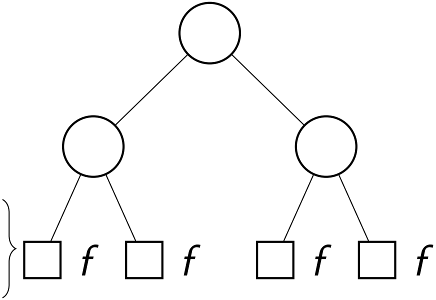
Recursion structure

```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```



Recursion structure

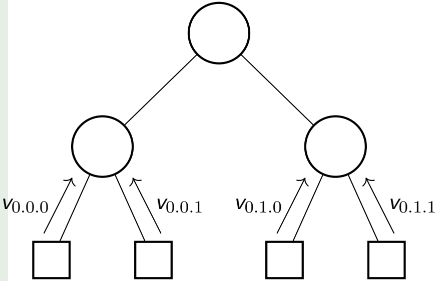
```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```



MULTI-ML: Tree recursion

Recursion structure

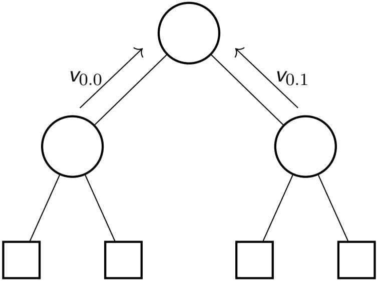
```
let multi f [args]=
  where node =
    (* BSML code *)
    ...
    << f [args] >>
    ... in v
  where leaf =
    (* OCaml code *)
    ... in v
```



MULTI-ML: Tree recursion

Recursion structure

```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```



Recursion structure

```
let multi f [args]=
  where node =
   (* BSML code *)
   ...
   << f [args] >>
   ... in v
  where leaf =
   (* OCaml code *)
   ... in v
```

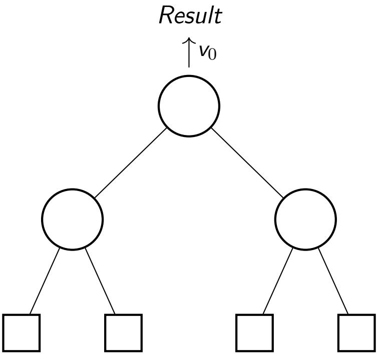


Table of Contents

Sequential semantics

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v}$$

Sequential semantics

Environment

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v}$$

Sequential semantics

Environment

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v}$$

Expression

Sequential semantics

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v}$$

Environment

Value

Expression

Sequential semantics

Premises

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v}$$

Environment

Value

Expression

Sequential semantics

Premises

Environment $$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v}$$ Value

Expression

Definitions

$$\begin{array}{lcl} \mathbf{v} & ::= & \mathbf{op} \quad | \quad \mathbf{cst} \quad | \quad \overline{(\mathbf{fun}\ \ x \to e\ )[\mathcal{E}]} \quad | \quad \overline{(\mathbf{rec}\ \ f\ x \to e\ )[\mathcal{E}]} \\ \mathcal{E} & ::= & \{x_1 \mapsto v_1, ...x_n \mapsto x_n\} \end{array}$$

Sequential semantics with cost

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v \rightsquigarrow C}$$

Sequential semantics with cost

Premises

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v \rightsquigarrow C}$$

Environment

Expression

Value

Sequential semantics with cost

Premises

Environment

$$\frac{\mathcal{P}}{\mathcal{E} \vdash e \Downarrow v \rightsquigarrow C}$$

Cost

Expression

Value

Definitions

Cost C is: $\sum_{\mathbf{c}\in\mathcal{C}} n_{\mathbf{c}} \times T_{\mathbf{c}}$

Sequential semantics with cost

Examples

$$\textsc{Csts}\quad \frac{}{\mathcal{E} \vdash^{s} \mathbf{cst} \Downarrow \mathbf{cst} \rightsquigarrow^{s} 0}$$

$$\textsc{Let}\quad \frac{\mathcal{E} \vdash^{s} e_1 \Downarrow v_1 \rightsquigarrow^{s_1} c_1 \quad \mathcal{E} \uplus \{x \mapsto v\} \vdash^{s_1} e_2 \Downarrow v_2 \rightsquigarrow^{s_2} c_2}{\mathcal{E} \vdash^{s} \mathbf{let}\ \ x = e_1\ \ \mathbf{in}\ \ e_2 \Downarrow v_2 \rightsquigarrow^{s_2} c_1 \oplus c_2 \oplus T_{let}}$$

BSML semantics with cost

Definitions

$v ::= \cdots \mid < v_1, \ldots, v_{\mathbf{p}} >$

Cost: $< c_1, \ldots, c_{\mathbf{p}} >_s \mid n \times \mathbf{g} \mid \mathbf{L}$

Cost algebra:

$< c_1, \ldots, c_{\mathbf{p}} >_s \oplus < c'_1, \ldots, c'_{\mathbf{p}} >_s \equiv < c_1 \oplus c'_1, \ldots, c_{\mathbf{p}} \oplus c'_{\mathbf{p}} >_s$

$< T_{op} \oplus c_1, \ldots, T_{op} \oplus c_{\mathbf{p}} >_s \equiv T_{op} \oplus < c_1, \ldots, c_{\mathbf{p}} >_s$

$0 \equiv < 0, \ldots, 0 >_s$

BSML semantics with cost

Examples

$$\textsc{Rpl}\quad \frac{\forall i \in \{1,\ldots,\mathbf{p}\} \quad \mathcal{E} \vdash^{s} e \Downarrow v_i \leadsto^{s} c_i}{\mathcal{E} \vdash^{s} < e_1,\ldots,e_p > \Downarrow < v_1,\ldots,v_{\mathbf{p}} > \leadsto^{s} T_{rpl} \oplus < c_1,\ldots,c_{\mathbf{p}} >_s}$$

$$\textsc{Proj}\quad \frac{\mathcal{E} \vdash^{s} e \Downarrow < v_1,\ldots,v_{\mathbf{p}} > \leadsto^{s'} c \quad \text{where } \forall i \in \{1,\ldots,\mathbf{p}\} \;\; \mathcal{E} \vdash (f\ i) \equiv v_i}{\mathcal{E} \vdash^{s} (\mathtt{proj}\ e) \Downarrow f \leadsto^{s'+1} T_{proj} \oplus c \oplus HRelation(v_1,\ldots,v_{\mathbf{p}}) \times \mathbf{g} \oplus \mathbf{L}}$$

Multi-ML semantics with cost

Definitions

$v ::= \ldots \ | \ \overline{(\mathbf{multi}\ \ f\ x \to e \dagger e)[\mathcal{E}]}$

Cost:

$\max(n_1 \times T_1 \oplus \cdots \oplus n_t \times T_m \oplus < c_1, \ldots, c_{\mathbf{p}_n} >_s) \equiv$
$\max(n_1 \times T_1 \oplus \cdots \oplus n_t \times T_t, \max_{i=1..\mathbf{p}_n}(c_i))$

Multi-ML semantics with cost

Example

$$\textsc{MultiNode}\quad \frac{\left\{\begin{array}{l} \mathcal{E} \vdash^{s} e_1 \Downarrow^{l}_{n} \overline{(\mathbf{multi}\ \ f\ x \to e'_1 \dagger e'_2)[\mathcal{E}']} \rightsquigarrow^{s_1} c_1 \\ \mathcal{E} \vdash^{s_1} e_2 \Downarrow^{l}_{n} v \rightsquigarrow^{s_2} c_2 \\ \mathcal{E}' \vdash^{0} e'_1 \Downarrow^{b}_{n+1} v' \rightsquigarrow^{s_3} c_3 \end{array}\right.}{\mathcal{E} \vdash^{s} (e_1\ \ e_2) \Downarrow^{l}_{n} v' \rightsquigarrow^{s_3} T_{app} \oplus c_1 \oplus c_2 \oplus \max(c_3) \oplus \mathbf{L}_n)}$$

Table of Contents

Matrix vector product algorithm

Estimate the cost of programs using the semantics

$\mathcal{S}(0) \times T_{map} \oplus \sum_{i=1}^{d} (\mathcal{S}(i-1) \times \mathbf{g}_{i-1} \oplus \mathbf{L}_{i-1}) \oplus \mathcal{S}(i) \times T_{red})$

Matrix vector product algorithm

Estimate the cost of programs using the semantics

$\mathcal{S}(0) \times T_{map} \oplus \sum_{i=1}^{d}(\mathcal{S}(i-1) \times \mathbf{g}_{i-1} \oplus \mathbf{L}_{i-1}) \oplus \mathcal{S}(i) \times T_{red})$

Benchmark the communication parameters

$\mathbf{g}_0 = 1100$, $\mathbf{g}_1 = 1800$, $\mathbf{g}_2 = 1$
$\mathbf{L}_0 = 149000$, $\mathbf{L}_1 = 1100$, $\mathbf{L}_2 = 1800$

Matrix vector product algorithm

Estimate the cost of programs using the semantics

$\mathcal{S}(0) \times T_{map} \oplus \sum_{i=1}^{d} (\mathcal{S}(i-1) \times \mathbf{g}_{i-1} \oplus \mathbf{L}_{i-1}) \oplus \mathcal{S}(i) \times T_{red})$

Benchmark the communication parameters

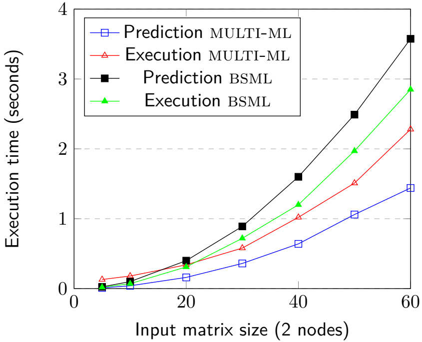
$\mathbf{g}_0 = 1100$, $\mathbf{g}_1 = 1800$, $\mathbf{g}_2 = 1$
$\mathbf{L}_0 = 149000$, $\mathbf{L}_1 = 1100$, $\mathbf{L}_2 = 1800$

Micro-Benchs the "real-time" of each construction

$T_{map} = 3 \times T_{get} \oplus T_{set} \oplus 2 \times T_{FloatMult} \oplus \cdots \oplus 10 \times T_{Var}$. With:

$$\begin{array}{rclrcl} T_{Set} & = & 1{,}778\mu s & T_{FloatMult} & = & 1{,}317\mu s \\ T_{Get} & = & 1{,}324\mu s & T_{Var} & = & 0.619\mu s \end{array}$$

Comparing predictions and benchmarks



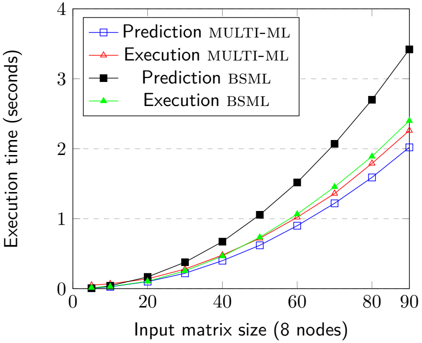


Table of Contents

Conclusion

Main results

- Formal cost semantics for ML, BSML and Multi-ML
- Semantics design incrementally
- Use of semantics for cost prediction
- Application to a skeleton based numerical example

Ongoing and future work

- Static and automatic analysis for cost prediction
- Need the use of annotations?
- Real world example

Thank you for your attention ☺

Questions ?