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

1 Introduction

2 Semantics with costs

3 Experiments

4 Conclusion
Table of Contents

1 Introduction
   Structured parallel computing
   BSP and BSML
   MULTI-BSP and MULTI-ML

2 Semantics with costs

3 Experiments

4 Conclusion
The world of parallel computing

Simulations:
- Fluid simulation
- 3D Visualisation

Big-Data:
- IoT
- Social Networking
- Data science

Symbolic computation:
- Model-Checking
- Formal computing
Distributed computing

Characterised by:

- Interconnected units
- Distributed memory
- Communication network
- MPI

![Distributed Computing Diagram]
Bulk Synchronous Parallelism

The BSP computer

Defined by:
- \( p \) pairs CPU/memory
- Communication network
- Synchronisation unit
- Super-steps execution

Properties:
- Confluent
- Deadlock-free
- Predictable performances

\[ p_0 \quad p_1 \quad p_2 \quad p_3 \]

local computations
communication barrier
next super-step
## Bulk Synchronous Parallelism

### BSP cost model

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

(expressed as multiples the local processing speed $r$)

### Superstep’s cost formulae

$$\text{Cost}(s) = \max_{0 \leq i < p} w_i^s + \max_{0 \leq i < p} h_i^s \times g + L$$
What is BSML?

- Explicit BSP programming with a functional approach
**Bulk Synchronous ML**

**What is BSML?**

- Explicit BSP programming with a functional approach
- Based upon ML and implemented over OCAMLO
What is BSML?

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

---

**Main idea**

Parallel data structure ⇒ *parallel vector*:

![Diagram](#)

**Replicated part (BSP):**

\[ f_0, f_1, \ldots, f_{p-1} \]

**Sequential part**

---

V. Allombert, F. Gava, J. Tesson

LaMHA 2018
Hierarchical architectures

Characterised by:

- Interconnected units
- Both shared and distributed memories
- Hierarchical memories
A tree structure with nested components
Where nodes have a storage capacity
And leaves are processors
With sub-synchronisation capabilities
MULTI-BSP

- 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

V. Allombert, F. Gava, J. Tesson
LaMHA 2018
The **MULTI-BSP** model

**Execution model**

A level $i$ superstep is:

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

Diagram:

- Node $n$ represents the current level $i$.
- Nodes $n.1$ and $n.p_i$ represent levels $i-1$.
- Arrows indicate communication and synchronization paths.
- $g_i$ and $g_{i-1}$ are communication paths.
- $m_i$ is the memory.

V. Allombert, F. Gava, J. Tesson  
LaMHA 2018
The **MULTI-BSP** model

**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
The MULTI-BSP model

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: $(p_i, g_i, L_i, m_i)$

**Cost formulae**

\[
T = \sum_{i=0}^{d-1} \left( \sum_{j=0}^{N_i-1} w^i_j + C^i_j \right)
\]
The MULTI-ML language

Basic ideas
The MULTI-ML language

Basic ideas

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

\[
\text{let } v = \langle\langle e \rangle\rangle
\]

\[
\langle\langle e \rangle\rangle \rightarrow \langle f_0, f_1, \ldots, f_p \rangle
\]

Sequential part

Replicated part (BSP)

Parallel vector
The MULTI-ML language

Basic ideas

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

```
let v = <<<e>>>
```

Replicated part (BSP) → `{ f_0, f_1, ..., f_{p-1} }` parallel vector

Sequential part

V. Allombert, F. Gava, J. Tesson
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

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

where leaf =
  (* OCaml code *)
  ...
  in v
```
MULTI-ML: Tree recursion

Recursion structure

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

Result

v 0 . 0 . 0
v 0 . 1 . 0
v 0 . 1 . 1
```
MULTI-ML: Tree recursion

Recursion structure

```plaintext
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

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

```
result v 0
```

```
V. Allombert, F. Gava, J. Tesson
```
**MULTI-ML: Tree recursion**

Recursion structure

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

![Tree recursion diagram](image)
MULTI-ML: Tree recursion

Recursion structure

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

---

Result

```ocaml```

---
**MULTI-ML: Tree recursion**

**Recursion structure**

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

Result

```
v_0
```

Diagram of recursion structure.
Table of Contents

1. Introduction
2. Semantics with costs
3. Experiments
4. Conclusion
Sequential semantics

\[ \ell \]
\[ \mathcal{P} \]
\[ \mathcal{E} \models e \Downarrow v \]
Sequential semantics

\[ \frac{P}{\mathcal{E} \vdash e \Downarrow v} \]
Sequential semantics

\[ \overline{E \vdash e \downarrow v} \]
Sequential semantics

\[ \mathcal{P} \quad \frac{\mathcal{E} \vdash e \downarrow v}{\mathcal{P}} \]

- Environment
- Expression
- Value
Sequential semantics

\[ \mathcal{P} \]

Environment \[ \mathcal{E} \vdash e \downarrow v \]

Premises

Value

Expression

Definitions

\[ v ::= \text{op} \] \[ j \] \[ \text{cst} \] \[ j \]

\[ (\text{fun} \ x ! e) \mathcal{E} \]

\[ (\text{rec} \ f x ! e) \mathcal{E} \]

V. Allombert, F. Gava, J. Tesson

LaMHA 2018

13 / 23
Sequential semantics

**Definitions**

\[
\begin{align*}
    v & ::= \text{op} \mid \text{cst} \mid (\text{fun } x \to e)[\mathcal{E}] \mid (\text{rec } f x \to e)[\mathcal{E}] \\
    \mathcal{E} & ::= \{x_1 \mapsto v_1, \ldots, x_n \mapsto x_n\}
\end{align*}
\]
Sequential semantics with cost

\[ \mathcal{P} \]

\[ \mathcal{E} \vdash e \Downarrow v \leadsto C \]
Sequential semantics with cost

\[ P \quad \text{Premises} \]

\[ \mathcal{E} \vdash e \downarrow v \leadsto C \quad \text{Environment} \]

\[ \text{Expression} \quad \text{Value} \]

\[ \text{Definitions} \]

\[ C \text{ is: } \sum c_{2C_n} T_c \]
Sequential semantics with cost

Definitions

Cost $C$ is: $\sum_{c \in C} n_c \times T_c$
Sequential semantics with cost

Examples

\[
\begin{align*}
\text{CSTS} & \quad \frac{E \mid \text{cst} \Downarrow \text{cst} \leadsto^s 0}{E \mid \text{cst}} \\
\text{LET} & \quad \frac{E \mid \text{e}_1 \Downarrow \text{v}_1 \leadsto^{s_1} \text{c}_1 \quad E \uplus \{ x \mapsto \text{v} \} \mid \text{e}_2 \Downarrow \text{v}_2 \leadsto^{s_2} \text{c}_2}{E \mid \text{let } x = \text{e}_1 \text{ in } \text{e}_2 \Downarrow \text{v}_2 \leadsto^{s_2} \text{c}_1 \oplus \text{c}_2 \oplus T_{let}}
\end{align*}
\]
Definitions

\( \nu ::= \cdots | < \nu_1, \ldots, \nu_p > \)

Cost: \( < c_1, \ldots, c_p >_s | n \times g | L \)

Cost algebra:

\( < c_1, \ldots, c_p >_s \oplus < c'_1, \ldots, c'_p >_s \equiv < c_1 \oplus c'_1, \ldots, c_p \oplus c'_p >_s \)

\( < T_{op} \oplus c_1, \ldots, T_{op} \oplus c_p >_s \equiv T_{op} \oplus < c_1, \ldots, c_p >_s \)

\( 0 \equiv < 0, \ldots, 0 >_s \)
BSML semantics with cost

Examples

\[
\begin{align*}
\text{RPL} & \quad \forall i \in \{1, \ldots, p\} \quad \mathcal{E} \vdash^s e \downarrow v_i \sim^s c_i \\
& \quad \mathcal{E} \vdash^s < e_1, \ldots, e_p > \downarrow < v_1, \ldots, v_p > \sim^s T_{\text{proj}} \oplus < c_1, \ldots, c_p >_s \\
\text{PROJ} & \quad \mathcal{E} \vdash^s e \downarrow < v_1, \ldots, v_p > \sim^{s'} c \quad \text{where } \forall i \in \{1, \ldots, p\} \quad \mathcal{E} \vdash (f \ i) \equiv v_i \\
& \quad \mathcal{E} \vdash^s (\text{proj } e) \downarrow f \sim^{s'+1} T_{\text{proj}} \oplus c \oplus HRelation(v_1, \ldots, v_p) \times g \oplus L
\end{align*}
\]
Multi-ML semantics with cost

Definitions

\[ \nu ::= \ldots \mid (\text{multi } f \times \rightarrow e \dagger e)[\mathcal{E}] \]

Cost:
\[
\max(n_1 \times T_1 \oplus \cdots \oplus n_t \times T_m \oplus < c_1, \ldots, c_{p_n} > s) \equiv \\
\max(n_1 \times T_1 \oplus \cdots \oplus n_t \times T_t, \max_{i=1..p_n}(c_i))
\]
Multi-ML semantics with cost

Example

\[
\begin{align*}
\text{MultiNode} & \quad \left\{ 
\begin{array}{l}
\mathcal{E} \vdash^s e_1 \downarrow_n^l (\text{multi } f x \rightarrow e'_1 \uparrow e'_2)[\mathcal{E'}] \rightsquigarrow^{s_1} c_1 \\
\mathcal{E} \vdash^{s_1} e_2 \downarrow_n^l v \rightsquigarrow^{s_2} c_2 \\
\mathcal{E'} \vdash^0 e'_1 \downarrow_{n+1}^b v' \rightsquigarrow^{s_3} c_3
\end{array}
\right.
\end{align*}
\]

\[
\mathcal{E} \vdash^s (e_1 e_2) \downarrow_n^l v' \rightsquigarrow^{s_3} T_{app} \oplus c_1 \oplus c_2 \oplus \max(c_3) \oplus L_n
\]
<table>
<thead>
<tr>
<th>#</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Semantics with costs</td>
</tr>
<tr>
<td>3</td>
<td>Experiments</td>
</tr>
<tr>
<td>4</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Matrix vector product algorithm

Estimate the cost of programs using the semantics

\[ S(0) \times T_{map} \oplus \sum_{i=1}^{d} (S(i-1) \times g_{i-1} \oplus L_{i-1}) \oplus S(i) \times T_{red} \]
Matrix vector product algorithm

Estimate the cost of programs using the semantics

\[ S(0) \times T_{\text{map}} \oplus \sum_{i=1}^{d} (S(i - 1) \times g_{i-1} \oplus L_{i-1}) \oplus S(i) \times T_{\text{red}}) \]

Benchmark the communication parameters

- \(g_0 = 1100\), \(g_1 = 1800\), \(g_2 = 1\)
- \(L_0 = 149000\), \(L_1 = 1100\), \(L_2 = 1800\)
Matrix vector product algorithm

Estimate the cost of programs using the semantics

\[ S(0) \times T_{map} \oplus \sum_{i=1}^{d} (S(i-1) \times g_{i-1} \oplus L_{i-1}) \oplus S(i) \times T_{red} \]

Benchmark the communication parameters

\[ g_0 = 1100, \ g_1 = 1800, \ g_2 = 1 \]
\[ L_0 = 149000, \ L_1 = 1100, \ L_2 = 1800 \]

Micro-Benchs the “real-time” of each construction

\[ T_{map} = 3 \times T_{get} \oplus T_{set} \oplus 2 \times T_{FloatMult} \oplus \cdots \oplus 10 \times T_{Var}. \] \[ \text{With:} \]
\[ T_{Set} = 1,778\mu s \]
\[ T_{Get} = 1,324\mu s \]
\[ T_{Var} = 0.619\mu s \]
Comparing predictions and benchmarks
Table of Contents

1 Introduction
2 Semantics with costs
3 Experiments
4 Conclusion
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 ?