Bulk Synchronous Parallel ML

Julien Tesson

LACL, Université Paris-Est Créteil
Mail : julien.tesson@u-pec.fr

2015
Créteil

Bulk Synchronous Parallel ML

Principes de conception

- Un ensemble de primitives parallèles restreint
- Universel pour le parallélisme BSP
- Vue globale du programme
- Une sémantique formelle simple

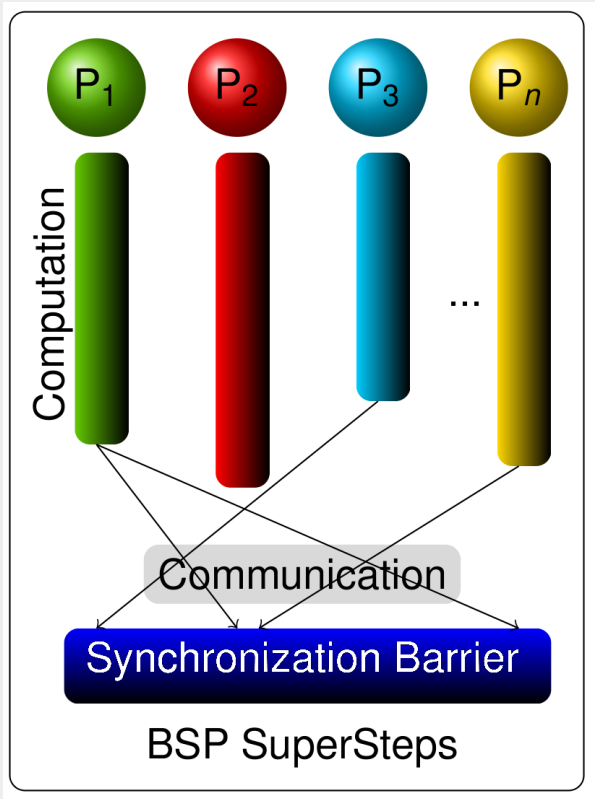
BSML

Un langage fonctionnel

\+ une structure de données parallèle (sans imbrication)

\+ des opérations parallèles sur cette structure

Modèle d'exécution BSP



Articles et logiciel :
http://traclifo.univ-orleans.fr/BSML

Implantation

Actuellement disponible

- Bibliothèque de programmation OCaml
- implantation Modulaire : MPI, TCP/IP, BSPlib, ...
- boucle interactive séquentielle

Rappels OCaml

Langage functionnel d'ordre supérieur

```
(** definition simple de type int → int → int *)
let my_addition a b = a + b

(* definition recursive, pattern matching, fonction en parametre *)
(* type : ( α → β ) → α list → β list *)
let rec my_map f l =
    match l with
    | [] → []
    | h::t → (f h)::(my_map f t)

(* fonction anonyme en parametre *)
(* type : int → int list → int list *)
let map_addition a l = my_map (fun b → my_addition a b) l
```

Bulk Synchronous Parallel ML

Vecteur Parallèle

- un type de donnée abstrait polymorphe : α **par**
- une taille fixe p : chaque processeur a une valeur de type α
- imbrication de vecteur interdite
- notation informelle : $\langle\ v_0\ , \ldots ,\ v_{p-1}\ \rangle$

Accès aux paramètres BSP

bsp_p: int
bsp_g: float
bsp_l: float

Bulk Synchronous Parallel ML - Primitives Parallèle

Création de vecteur parallèles

- **mkpar** : (int $\rightarrow$ α) $\rightarrow$ α **par**

(**mkpar** f)

$(f\ 0)$	$(f\ 1)$	$\cdots$	$(f\ (p-1))$

Exemple avec p=4

```
# let this =
 mkpar (fun i -> i) ;;

val this : int Bsml.par
= <0,1,2,3>
```

0	1	2	3

```
# mkpar (fun pid -> print_endline
    ("hello from "^(string_of_int pid)));;
hello from 3
hello from 1
hello from 0
hello from 2
- : unit Bsml.par
```

Bulk Synchronous Parallel ML - Primitives Parallèle

Application parallèle point à point

- **apply** : ($\alpha \rightarrow \beta$) **par** $\rightarrow$ α **par** $\rightarrow$ β **par**

$$\left(\textbf{apply} \begin{array}{|c|c|c|c|} \hline f_0 & f_1 & \cdots & f_{p-1} \\ \hline \end{array} \right.$$
$$\left. \begin{array}{|c|c|c|c|} \hline v_0 & v_1 & \cdots & v_{p-1} \\ \hline \end{array} \right)$$
$$= \begin{array}{|c|c|c|c|} \hline (f_0\ v_0) & (f_1\ v_1) & \cdots & (f_{p-1}\ v_{p-1}) \\ \hline \end{array}$$

Exemple avec p=4

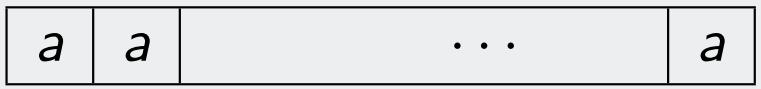
**let** plus_deux = (**mkpar** (**fun** pid $\rightarrow$ (**fun** x $\rightarrow$ x +2))) ;;
val plus_deux : (int $\rightarrow$ int) Bsml.**par** = <**fun**,**fun**,**fun**,**fun**>

**apply** plus_deux this
− : int Bsml.**par** = <2,3,4,5>

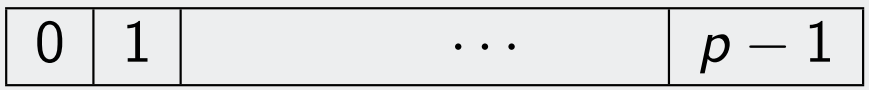
Bulk Synchronous Parallel ML - Primitives Parallèle

Syntaxe alternative

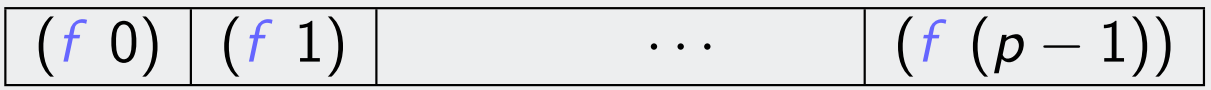
- $\ll$ a $\gg$



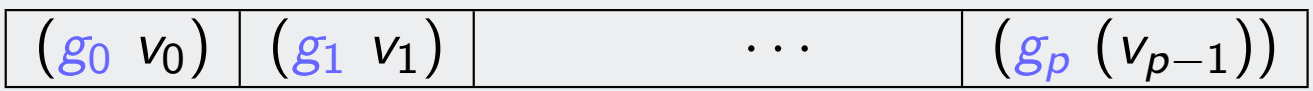
- $\ll$ $this$ $\gg$



- let mkpar f = $\ll$ f $this$ $\gg$



- **let apply** g v =$\ll$ g v $\gg$



Bulk Synchronous Parallel ML

Primitives Parallèle - communications

- **proj**: α **par** $\rightarrow$ (int $\rightarrow$ α)

$$\left(\textbf{proj}\ \boxed{v_0 \mid v_1 \mid \cdots \mid v_{p-1}}\right) = \begin{array}{lll} \textbf{function} & 0 & \rightarrow\ v_0 \\ \mid & 1 & \rightarrow\ v_1 \\ & \vdots & \\ \mid & p-1 & \rightarrow\ v_{p-1} \end{array}$$

Bulk Synchronous Parallel ML - Primitives Parallèle

Communications

- **put**: (int $\rightarrow$ α) **par** $\rightarrow$ (int $\rightarrow$ α) **par**

$$\left(\textbf{put} \quad \boxed{f_0 \mid f_1 \mid \cdots \mid f_{p-1}}\right) = \boxed{g_0 \mid g_1 \mid \cdots \mid g_{p-1}}$$

$g_i\ j = f_j\ i$

0	1	2	3
A	B	C	D
$(f_0\ 1)$	$(f_1\ 1)$	$(f_2\ 1)$	$(f_3\ 1)$
$(f_0\ 2)$	$(f_1\ 2)$	$(f_2\ 2)$	$(f_3\ 2)$
$(f_0\ 3)$	$(f_1\ 3)$	$(f_2\ 3)$	$(f_3\ 3)$

$\Rightarrow$

0	1	2	3
A	$(f_0\ 1)$	$(f_0\ 2)$	$(f_0\ 3)$
B	$(f_1\ 1)$	$(f_1\ 2)$	$(f_1\ 3)$
C	$(f_2\ 1)$	$(f_2\ 2)$	$(f_2\ 3)$
D	$(f_3\ 1)$	$(f_3\ 2)$	$(f_3\ 3)$

Bulk Synchronous Parallel ML en pratique

Boucle interactive (toplevel)

- S'assurer de l'existence d'un fichier `.bsmlrc` dans le home contenant le nombre de nœuds, pref réseau, latence, perf processeur de la machine parallèle simulée :
  $ **echo** '8,50.,5000.,400000.' > ∼/.bsmlrc
- lancer la boucle interactive :
  $ bsml
- Dans le boucle interactive, ouvrez le module contenant les primitives BSML
  # **open** Bsml;;
- Pour charger un fichier source `mon_code.ml` :
  # #use "mon_code.ml";;

Bulk Synchronous Parallel ML en pratique

Parallèle avec mpi

- (sous certain OS) $ module load mpi
- compilation $ bsmlopt.mpi hello.ml −o hello.mpi
- execution parallèle $ mpirun −np 16 ./hello.mpi