A new bridging model for high-performance programming

Chong LI (Doctorant CIFRE)

LACL, Université Paris-Est
EXQIM S.A.S.

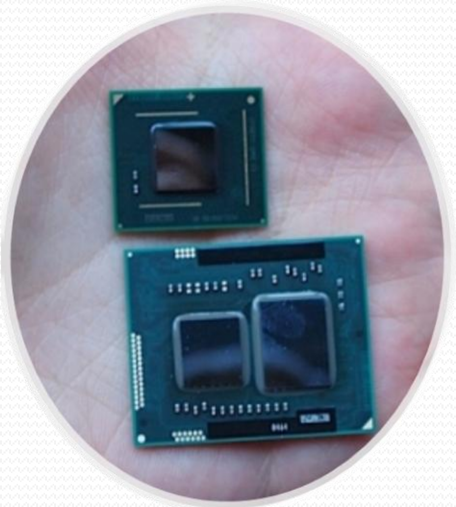
2nd Journée de LaMHA
Dec 14, 2010 - LIFO

PLAN

- Bridging Models
- Scatter-Gather Language
- Operational Semantics

Parallel Computing & Bridging Model

Various Computers



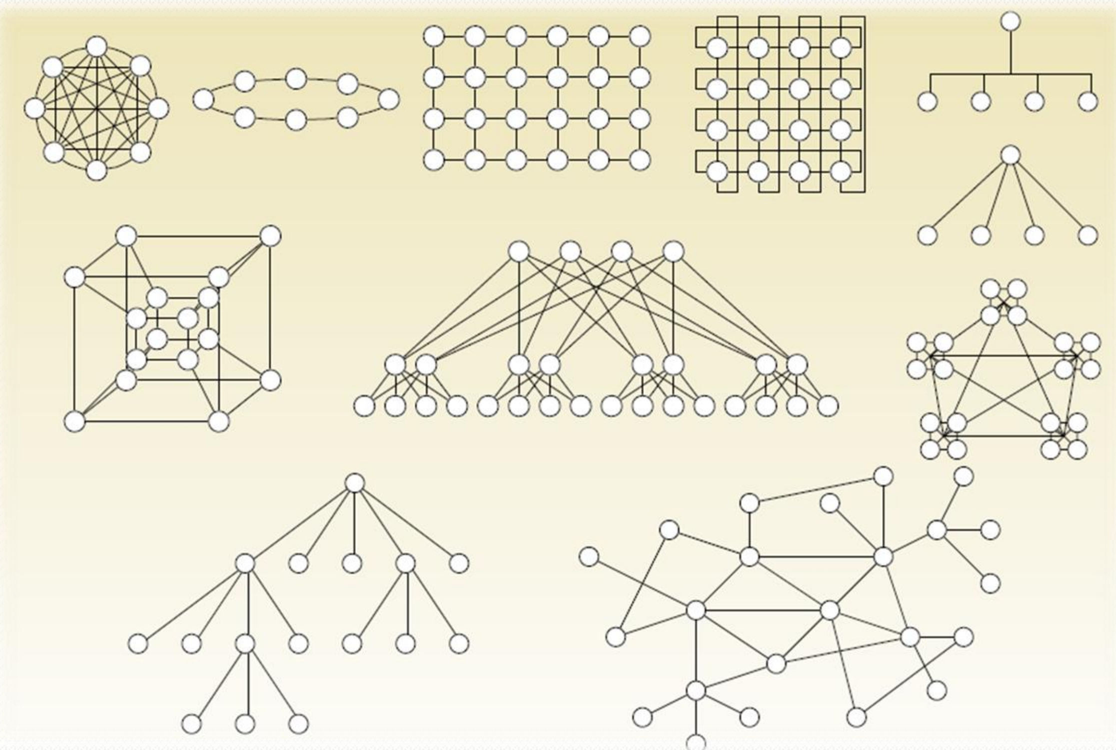
- Multi-core computing

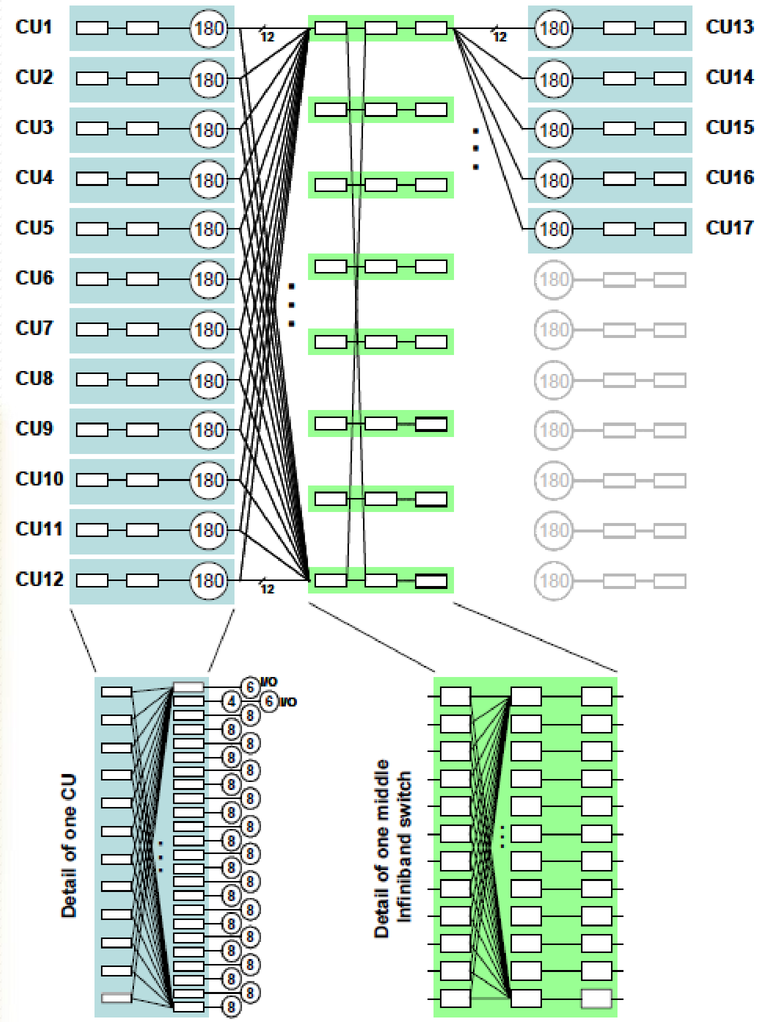
  e.g. Intel i5, STI cell
- Symmetric multiprocessing (SMP)

  iden proc, share memory, connect via a bus
- Specialized parallel computers
  - FPGA
  - GP-GPU
  - ASIC
  - Vector processors

- Distributed computing
  - Cluster computing
  - Massive parallel processing (MPP) ≈ big cluster
  - Grid computing distributed form, over Internet

Various Topologies





- An overview of the Roadrunner system interconnection

Top 500 (November 2010)

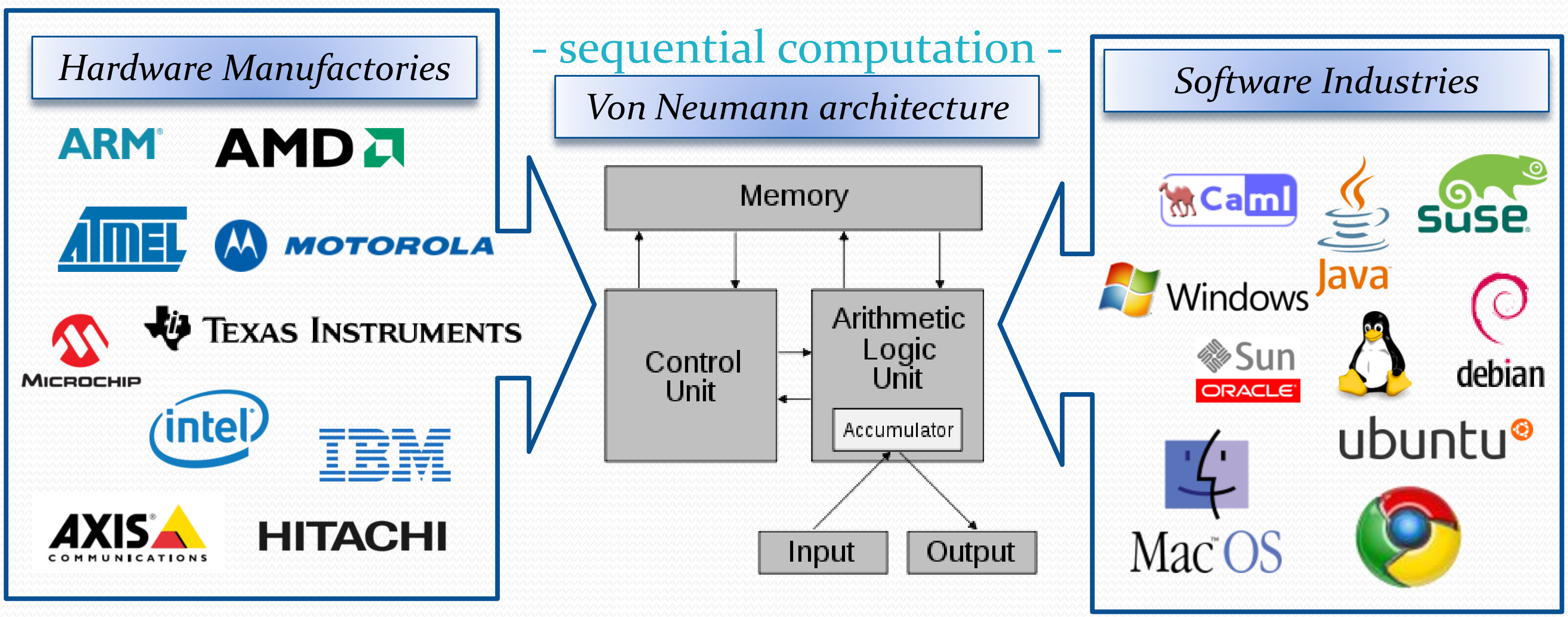
Rank	Rmax Rpeak (Tflops)	Name	Computer Processor cores	Vendor	Site Country, Year
1	2566.00 4701.00	*Tianhe-1A*	**NUDT YH Cluster** 14,336x6 Xeon + 7168×14 Fermi, Proprietary	NUDT	National Supercomputing Center in Tianjin China, 2010
2	1759.00 2331.00	*Jaguar*	**Cray XT5** 224,162 Opteron	Cray	Oak Ridge National Laboratory United States, 2009
3	1271.00 2984.30	*Nebulae*	**Dawning TC3600 Blade** 55,680 Xeon + 64,960 Tesla, InfiniBand	Dawning	National Supercomputing Center in Shenzhen (NSCS) China, 2010
4	1192.00 2287.63	*TSUBAME 2.0*	**HP Cluster Platform 3000SL** 73,278 Xeon, Fermi	NEC/HP	GSIC Center, Tokyo Institute of Technology Japan, 2010
5	1054.00 1288.63	*Hopper*	**Cray XE6** 153,408 Opteron	Cray	DOE/SC/LBNL/NERSC United States, 2010
6	1050.00 1254.55	*Tera 100*	**Bull Bullx** 138,368 Xeon, InfiniBand	Bull SA	Commissariat à l'énergie atomique (CEA) France, 2010
7	1042.00 1375.78	*Roadrunner*	**BladeCenter QS22/LS21** 122,400 Cell/Opteron	IBM	Los Alamos National Laboratory United States, 2009
8	831.70 1028.85	*Kraken*	**Cray XT5** 98,928 Opteron	Cray	National Institute for Computational Sciences United States, 2009
9	825.50 1002.70	*JUGENE*	**Blue Gene/P Solution** 294,912 Power	IBM	Forschungszentrum Jülich Germany, 2009
10	816.60 1028.66	*Cielo*	**Cray XE6** 107,152 Opteron	Cray	DOE/NNSA/LANL/SNL United States, 2010

Source: *http://en.wikipedia.org/wiki/TOP500*

Bridging Model for Computation

- *The success of the von Neumann model of sequential computation is attributable to the fact that it is an efficient bridge between software and hardware: high-level languages can be efficiently compiled on to this model; yet it can be efficiently implemented in hardware.*

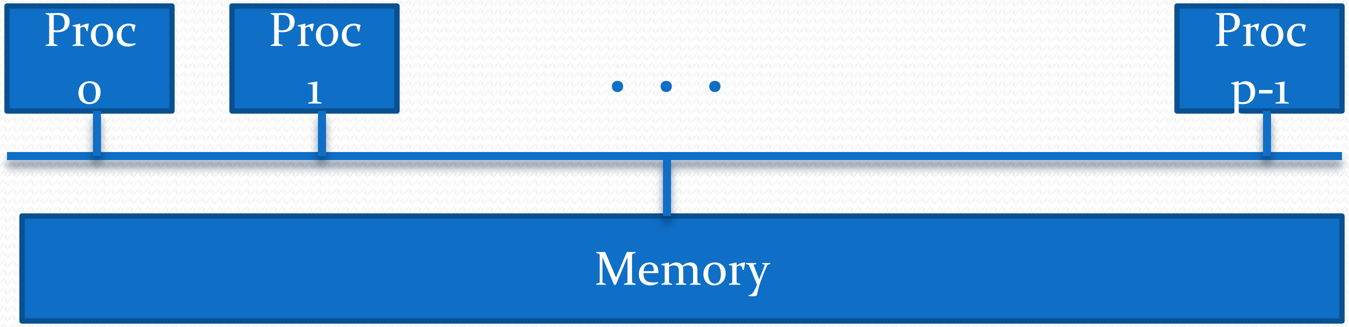
-- Leslie G. Valiant, 1990



PRAM Model

PRAM: Parallel Random-Access Machine (*Fortune et Wyllie, 1978*)

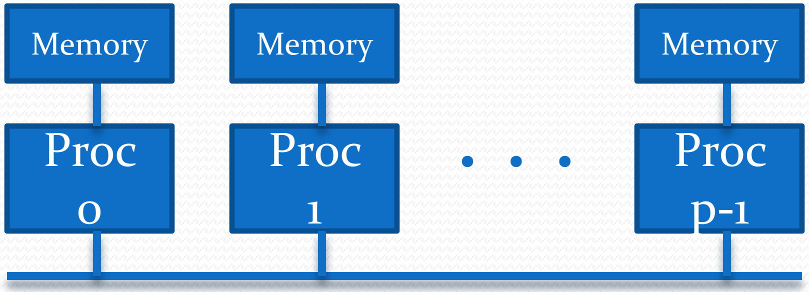
- *Shared-memory (random memory access)*
- *Synchronized-processor (same µProc clock)*

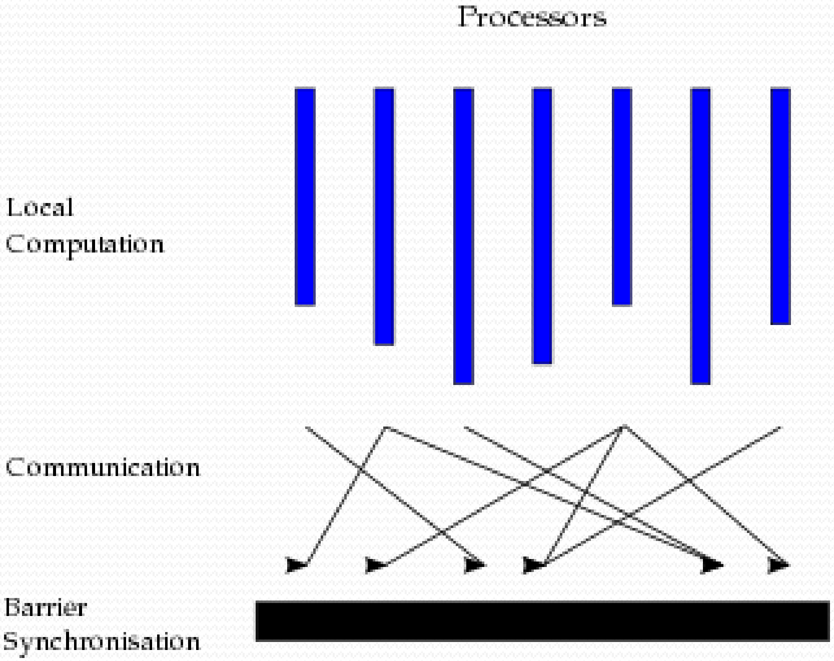


- Exclusive Read, Exclusive Write (EREW) PRAM
- Concurrent Read, Exclusive Write (CREW) PRAM
- Exclusive Read, Concurrent Write (ERCW) PRAM
- Concurrent Read, Concurrent Write (CRCW) PRAM

BSP Model

- **BSP: Bulk-Synchronous Parallel** (*Valiant, Aug 1990*)
  - Processor-memory pair
  - Synchronization barrier

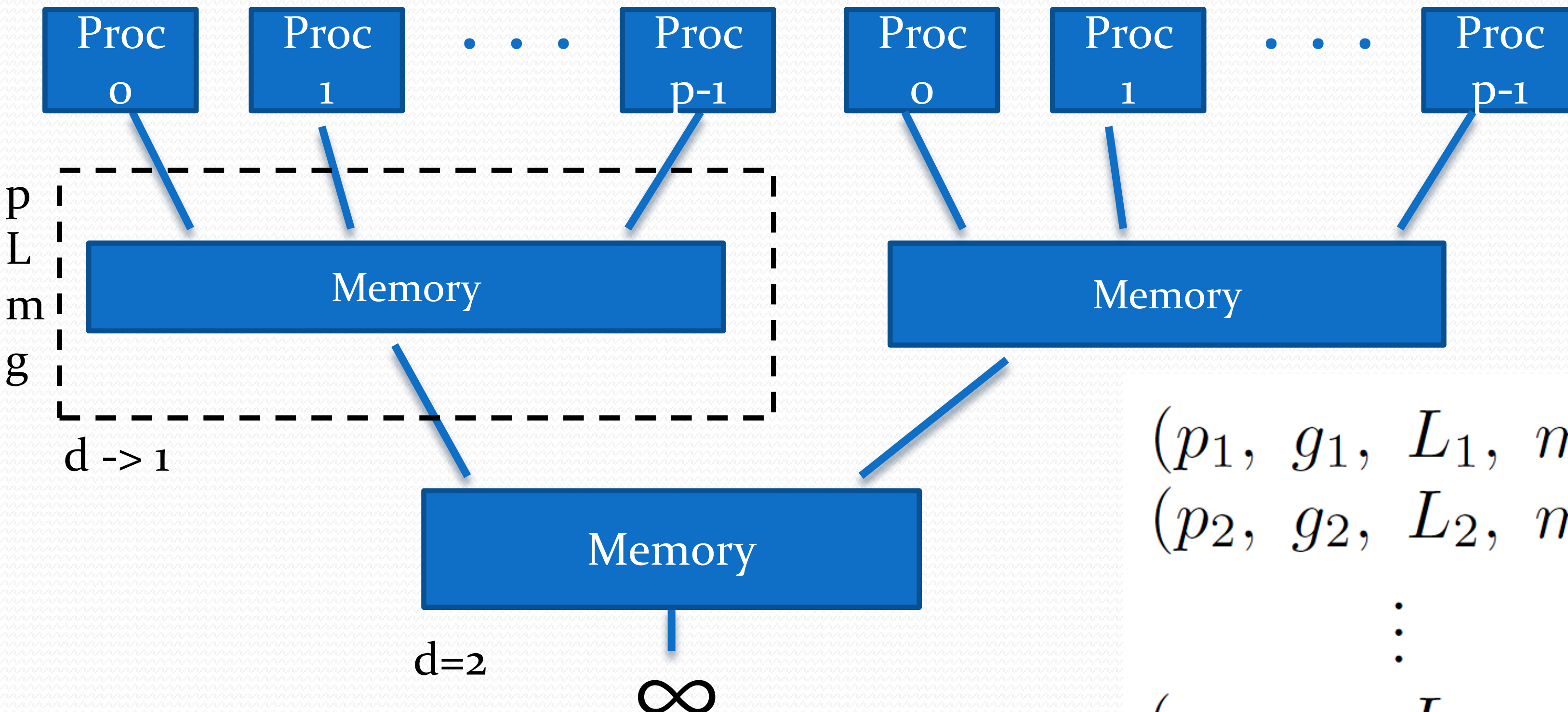




$$W + Hg + Sl = \sum_{s=1}^{S} w_s + g \sum_{s=1}^{S} h_s + Sl$$

Multi-BSP Model

- A bridging model for multi-core computing (*Valiant, 2008*)



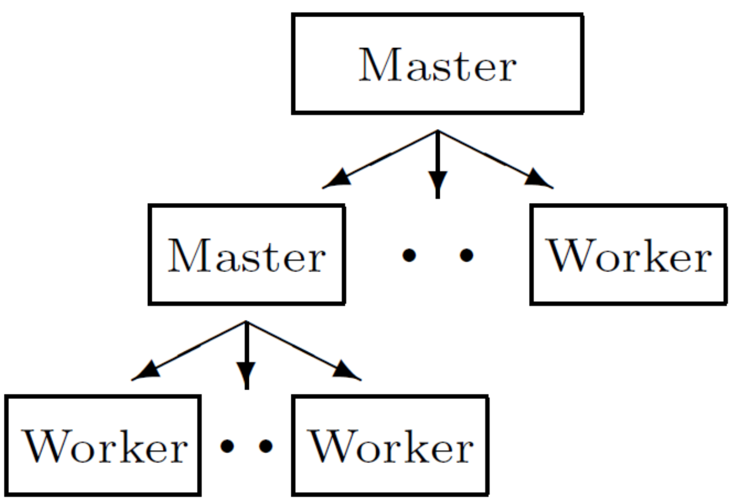
$$
\begin{array}{c}
(p_1,\ g_1,\ L_1,\ m_1) \\
(p_2,\ g_2,\ L_2,\ m_2) \\
\vdots \\
(p_d,\ g_d,\ L_d,\ m_d)
\end{array}
$$

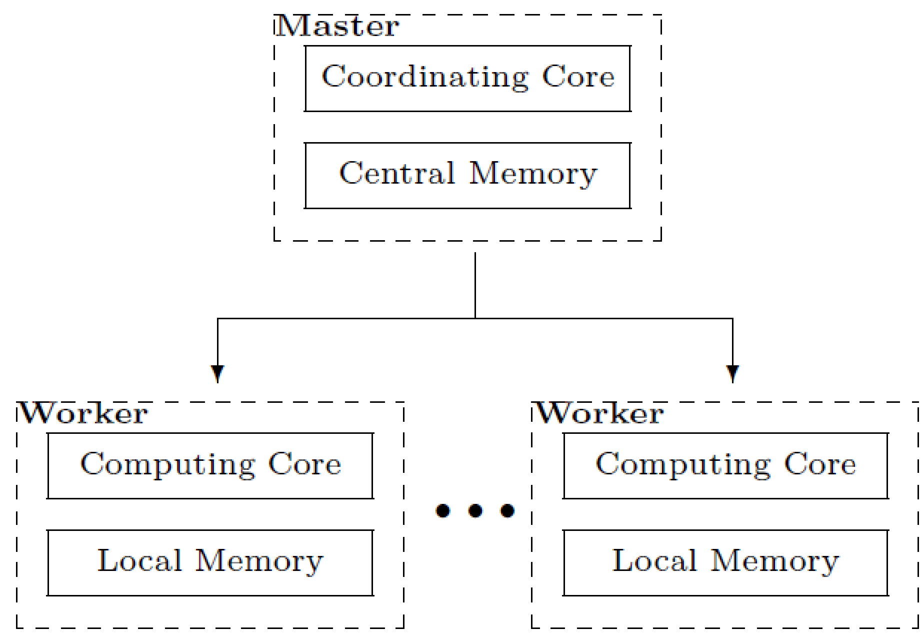
Scatter-Gather Language

Objectives

- Easy to learn, easy to use
- Portability of programmes
- Support heterogeneous machine
- Support hybrid architecture
- Performance prediction

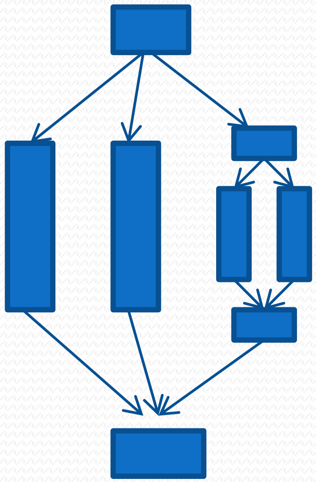
Abstract Machine





Execution Model

- A program is composed of a sequence of *supersteps*
- A *superstep* is composed of 4 phases:
  - Scatter communication phase
  - Children computation phase
    - A child computation phase can also be sub-*superstep*
  - Gather communication phase
  - Master computation phase



Cost Model

- l : latency of *scatter* or *gather*
- g : gap, minimum time interval for transmitting one word
- k : number of words to transmit
- w : works to be performed
- c : computation speed of processor
- p : number of children

$$\begin{aligned} Supstep_{par} &= 2l + k_{\downarrow} * g_{\downarrow} + \max_{i=1..p}(Supstep_{children}) + k_{\uparrow} * g_{\uparrow} + w * c \\ Supstep_{seq} &= w * c \end{aligned}$$

Programming Model

- Operational Semantics
  - Syntax
  - Evaluations
  - Examples

Syntax

- Values
  - $n \in$ numbers **Nat**
  - $< n_1, n_2, \ldots n_\ell > \in$ arrays of number **Vec**
  - $< v_1, v_2, \ldots v_\ell > \in$ arrays of array **VecVec**
- Locations
  - $X \in$ scalar locations **NatLoc**
    Here X *or* $X_0 \Rightarrow Master,\ X_{i=1..p} \Rightarrow Child$
  - $\overrightarrow{V} \in$ vectorial locations **VecLoc**
    Here $\overrightarrow{V}$ *or* $\overrightarrow{V}_0 \Rightarrow Master,\ \overrightarrow{V}_{i=1..p} \Rightarrow Child$
  - $\widetilde{W} \in$ vectorial vectorial locations **VVecLoc**

Syntax (cont.)

- Operations
  - $\odot \in$ binary operations $\mathbf{Op} ::= + \mid - \mid * \mid /$
- Expressions
  - $a \in$ scalar arithmetic expressions $\mathbf{Aexp} ::=$
    $n \mid X \mid a \odot a \mid \overrightarrow{V}[a]$
  - $b \in$ scalar boolean expressions $\mathbf{Bexp} ::=$
    $\mathbf{true} \mid \mathbf{false} \mid a = a \mid a \leq a \mid \neg b \mid b \wedge b \mid b \vee b$
  - $v \in$ vectorial expressions $\mathbf{Vexp} ::=$
    $< a_1, a_2, \ldots a_\ell > \mid \overrightarrow{V} \mid v \odot a \mid v \odot v \mid \widetilde{W}[a]$
  - $w \in$ vectorial vectorial expressions $\mathbf{VVexp} ::=$
    $< v_1, v_2, ..v_\ell > \mid \widetilde{W}$

Syntax (cont.)

- Commands
  - $c \in$ primitive commands $\mathbf{Com} ::=$

$$
\begin{aligned}
& X := a \mid \overrightarrow{V} := v \mid \widetilde{W} := w \mid c \; ; \; c \\
& \mid \textbf{skip} \mid \textbf{if } b \textbf{ then } c \textbf{ else } c \mid \textbf{for } X \textbf{ from } a \textbf{ to } a \textbf{ do } c \\
& \mid \textbf{scatter } w \textbf{ to } \overrightarrow{V} \mid \textbf{scatter } v \textbf{ to } X \\
& \mid \textbf{gather } \overrightarrow{V} \textbf{ to } \widetilde{W} \mid \textbf{gather } X \textbf{ to } \overrightarrow{V} \\
& \mid \textbf{pardo } c
\end{aligned}
$$

  - Auxiliary commands $\mathbf{Aux} ::=$

$$\textbf{numChd} \mid \textbf{len } \overrightarrow{V} \mid \textbf{len } \widetilde{W}$$

Evaluations

- State/Environment Functions

  - $\sigma : NatLoc \rightarrow Nat$, thus $\sigma(X) \in Nat$
  - $\sigma : VecLoc \rightarrow Vec$, thus $\sigma(\overrightarrow{V}) \in Ver$
  - $\sigma : VVecLoc \rightarrow VecVec$, thus $\sigma(\widetilde{W}) \in VecVec$

  Here $Pos \in Nat$ denotes a position of location:
  $0 \Rightarrow master$ (same as above), and $i \in \{1..p\} \Rightarrow i$-th child.

  - $\sigma : NatLoc \rightarrow Pos \rightarrow Nat$, thus $\sigma(X_{pos}) = \sigma_{pos}(X) \in Nat$
  - $\sigma : VecLoc \rightarrow Pos \rightarrow Vec$, thus $\sigma(\overrightarrow{V}_{pos}) = \sigma_{pos}(\overrightarrow{V}) \in Vec$

Evaluations (cont.)

- Primitive Commands
  - Scatters

$$\frac{\langle w,\ \sigma\rangle \rightarrow < v_1, v_2, \ldots v_\ell > \quad \forall_{i=1..numChd}\langle \overrightarrow{V}_i := v_i,\ \sigma\rangle \rightarrow \sigma'_i}{\langle \textbf{scatter}\ w\ \textbf{to}\ \overrightarrow{V},\ \sigma\rangle \rightarrow \sigma'}$$

$$\frac{\langle v,\ \sigma\rangle \rightarrow < n_1, n_2, \ldots n_\ell > \quad \forall_{i=1..numChd}\langle X_i := n_i,\ \sigma\rangle \rightarrow \sigma'_i}{\langle \textbf{scatter}\ v\ \textbf{to}\ X,\ \sigma\rangle \rightarrow \sigma'}$$

  - Gathers

$$\frac{\langle \widetilde{W} :=< \overrightarrow{V}_1, \overrightarrow{V}_2, \ldots \overrightarrow{V}_{numChd} >,\ \sigma\rangle \rightarrow \sigma'}{\langle \textbf{gather}\ \overrightarrow{V}\ \textbf{to}\ \widetilde{W},\ \sigma\rangle \rightarrow \sigma'}$$

$$\frac{\langle \overrightarrow{V} :=< X_1, X_2, \ldots X_{numChd} >,\ \sigma\rangle \rightarrow \sigma'}{\langle \textbf{gather}\ X\ \textbf{to}\ \overrightarrow{V},\ \sigma\rangle \rightarrow \sigma'}$$

  - Parallel

$$\frac{\forall_{i=1..chdnum}\langle c,\ \sigma_i\rangle \rightarrow \sigma'_i}{\langle \textbf{pardo}\ c,\ \sigma\rangle \rightarrow \sigma'}$$

Example - Sum

Algorithm 1 Procedure: Sum(in Vec, out Nat)

```
if (numChd ≠ 0) and (use children) then
  W̃ := cut →Src to suit for children
  scatter W̃ to →V
  for all par do
    Sum(→V, LocRes)
  end for
  gather LocRes to →Res
  if parallel needed then
    Sum(→Res, Result)
  else
    Result := ∑ →Res
  end if
else
  Result := ∑ →Src
end if
return Result
```

$$
\begin{aligned}
Supstep_{par} &= 2l + (k_{in} + p) * g + \max_{i=1..p}(Supstep_{children}) + O(sum(p)) * c \\
Supstep_{seq} &= O(sum(k_{in})) * c
\end{aligned}
$$

Example - Sort

Algorithm 2 Procedure: Sort(in V, out V)

if $(numChd \neq 0)$ and (use children) **then**
  $\widetilde{W}$:= cut $\overrightarrow{Src}$ to suit for children
  scatter $\widetilde{W}$ **to** $\overrightarrow{V}$
  for all do
    $Sort(\overrightarrow{V}, \overrightarrow{V}')$
  end for
  gather $\overrightarrow{V}'$ **to** $\widetilde{List}$
  $\overrightarrow{Res}$:=merge $\widetilde{List}$
else
  $\overrightarrow{Res}$:=sort $\overrightarrow{Src}$
end if
return $\overrightarrow{Res}$

$$Supstep_{par} = 2l + (2 * k) * g + \max_{i=1..p}(Supstep_{children}) + O(merge(k)) * c$$
$$Supstep_{seq} = O(sort(k)) * c$$

Example - Scan

Algorithm 3 Procedure: Scan(in V, out V)

if $(numChd \neq 0)$ and (use children) **then**

$\widetilde{W}$:= cut $\overrightarrow{Src}$ to suit for children

scatter $\widetilde{W}$ **to** $\overrightarrow{V}$

for all do

$Scan(\overrightarrow{V}, \overrightarrow{V}')$

$X := \overrightarrow{V}'[\text{len } \overrightarrow{V}']$

end for

gather X **to** $\overrightarrow{Lst}$

for i **from** 2 **to** (len $\overrightarrow{Lst}$) **do** $\overrightarrow{Lst}[i] := \overrightarrow{Lst}[i-1]$

$\overrightarrow{Lst}[1] := 0$

$\overrightarrow{Lst}$:= scan $\overrightarrow{Lst}$

scatter $\overrightarrow{Lst}$ **to** X

for all do

$\overrightarrow{V}' := \overrightarrow{V}' + X$

end for

gather $\overrightarrow{V}'$ **to** $\widetilde{W}$

$\overrightarrow{Res}$:=concatenate $\widetilde{W}$

else

$\overrightarrow{Res}$:= scan $\overrightarrow{Src}$

end if

return $\overrightarrow{Res}$

Future works

- Implementation
- Experiments
- Optimization
  - Communication between children
  - Pipeline
  - …
- Load Balancing
- New Algorithms
- …

Thank you for your time

- Questions?