BSP (1)

Bulk-Synchronous Processing Model
Models of parallel computation

- **Shared-memory model**
  - Implicit communication
    - algorithm design and analysis relatively simple
    - but implementation issues shine through
      - caches, distribution of data in memories, consistency, synchronization costs, ….
    - limits to scaling in practice

- **Distributed-memory model**
  - explicit communication (message passing)
    - design and analysis takes into account interconnection network and is complex
    - results not easily transferred to different networks

- **“Bridging” model**
  - simplified communication costs
    - balance realism with tractability of analysis
    - independent of detailed network characteristics (topology, routing, etc.)
    - cost model relies on average or “expected” network behavior
Bridging model of parallel computation

- $p$ (processor-memory) pairs
  - $p$ separate address spaces (distributed memory)

- Memory references
  - segregated into local and remote references
  - remote references
    - are explicit, typically in the form (proc, addr)
    - carry communication cost

- Global barrier synchronization
  - has large cost
BSP - Bulk Synchronous Parallel programming model

- BSP algorithm consists of a sequence of *supersteps*
- Superstep \( i \) consists of
  - local work: processors compute asynchronously
    - access values in local memory
    - record remote reads & writes to be performed
  - global communication
    - let \( Out^j_i \) be the set of values leaving proc \( j \) in step \( i \)
    - let \( In^j_{i+1} \) be the set of values arriving at proc \( j \) at the start of step \( i + 1 \)
    - the relation \( Out^j_i \leftrightarrow In^j_{i+1} \) over all processors specifies the communication pattern
  - global synchronization
    - ensure communication phase is complete
    - ensure memory incorporates all updates
BSP communication cost

- **Definition**
  - the *communication size* in step $i$ (measured in 8-byte *words*) is
    \[
    h_i = \max \left( \max_{0 \leq j < p} (|Out_i^j|, |In_{i+1}^j|) \right)
    \]

  - the *communication cost* for superstep $i$ is $h_i \cdot g + L$
    - $g$ and $L$ are machine-specific parameters of the cost model where
      - $g$ (bandwidth$^{-1}$ i.e. time per word) is the per-processor full-load permeability of the network
      - $L$ (latency) is the transit time across the network plus any additional time for barrier synchronization of the processors

```plaintext
Source Proc  | Dest Proc
0            | 0
1            | 1
2            | 2
3            | 3
```
Basic communication operations (1)

– Send n values from proc 1 to proc 3

Source Proc

Dest Proc

h =

BSP communication cost =
Basic communication operations (2)

- Exchange n values between proc 1 and proc 3

\[
h = \text{BSP communication cost} = \]

Source Proc

\[
0 \quad 1 \\
2 \quad 3
\]

Dest Proc

\[
0 \quad 1 \\
2 \quad 3
\]
Basic communication operations (3)

- Exchange \( n \) values between proc \( i \) and proc \( H(i) \) for all \( 0 \leq i < p \), with \( H \) a permutation of \( 0:p-1 \)

\[
\begin{array}{c|c}
\text{Source Proc} & \text{Dest Proc} \\
0 & 0 \\
1 & 1 \\
2 & 2 \\
3 & 3 \\
\end{array}
\]

\[ h = \]

BSP communication cost =
Basic communication operations (4)

– Distribute \( n = kp \) values in proc 0 over \( p \) procs. Each proc receives \( k \) values from proc 0

\[
\begin{align*}
\text{Source Proc} & : 0 \quad 1 \quad 2 \quad 3 \\
\text{Dest Proc} & : 0 \quad 1 \quad 2 \quad 3
\end{align*}
\]

\[
h = \text{BSP communication cost} =
\]
Basic communication operations (5)

– Combine $n = kp$ values into proc 0. Each proc sends $k$ values

<table>
<thead>
<tr>
<th>Source Proc</th>
<th>Dest Proc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

$h = \text{BSP communication cost } = \quad$
Basic communication operations (6)

- **Total exchange** (all-to-all exchange) of \( n = kp \) values among \( p \) processors. Each processor receives \( k \) values from every other processor.

\[
\begin{array}{ccc}
\text{Source Proc} & \text{Dest Proc} \\
0 & 0 \\
0 & 1 \\
0 & 2 \\
0 & 3 \\
1 & 0 \\
1 & 1 \\
1 & 2 \\
1 & 3 \\
2 & 0 \\
2 & 1 \\
2 & 2 \\
2 & 3 \\
3 & 0 \\
3 & 1 \\
3 & 2 \\
3 & 3 \\
\end{array}
\]

\[
h = \text{BSP communication cost} =
\]
Basic communication operations (7)

– Broadcast n values from proc 0 to all other processors

<table>
<thead>
<tr>
<th>Source Proc</th>
<th>Dest Proc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
h = \text{BSP communication cost} = \]
BSP programming and execution model

• Basic presentation style is processor-centric
  – not like WT programs
    • explicit processor id \( j \)
    • number of processors \( p \)

• Single-Program Multiple-Datastream (SPMD) execution model
  – all processors execute same sequential program asynchronously
  – explicitly specify distribution of data over processors
  – specify supersteps
  – for each superstep specify
    • work to be performed by each processor
    • h-relation to be communicated
BSP cost

- Total cost of a BSP algorithm
  - let $c$ be the number of supersteps
  - let $p$ be the number of processors
  - Define
    \begin{align*}
    w_i &= \max_{0 \leq j < p} \text{ (work done in FLOPS on superstep } i \text{ by processor } j) \\
    h_i &= \max_{0 \leq j < p} \left( \max \left( \left| \text{Out}_i^j \right|, \left| \text{In}_i^{j+1} \right| \right) \right)
    \end{align*}

  - then total cost (~ running time) $C(n, p)$ of a BSP algorithm is
    \begin{align*}
    C(n, p) &= \sum_{i=1}^{c} (w_i + h_i \cdot g + L) \\
    &= \sum_{i=1}^{c} w_i + \sum_{i=1}^{c} (h_i \cdot g + L)
    \end{align*}
BSP algorithm: Vector summation

- **Problem:** given $V^n$ distributed evenly over $p$ processors, find $s = \text{Sum}(V)$
  - for simplicity, assume $p = 2^k$ and $p$ divides $n$
  - let $0 \leq j < p$ be the processor id
  - initially processor $j$ holds $r = n/p$ values: $V[j \cdot r : (j + 1) \cdot r - 1]$
  - on completion, each processor holds the value of $s$

- **Algorithm**
  - **Superstep 1**
    - $s := \text{Sum} (V[j \cdot r : (j + 1) \cdot r - 1])$
    - read $s$ from proc $(j + 1) \mod p$ into $s'$
  - **Superstep $i = 2$ to $\log_2 p$**
    - $s := s + s'$
    - read $s$ in proc $(j + 2^{i-1}) \mod p$ into $s'$
  - **Superstep $1 + \log_2 p$**
    - $s := s + s'$

- **BSP cost**
BSP algorithm: Matrix * Vector

- Problem: given $M^{nxn}$, $V^n$ distributed evenly over $p$ processors, compute $R = M \cdot V$
  - for simplicity, assume $p$ divides $n$
  - initially each processor holds $n^2/p$ values of $M$, and $n/p$ values of $V$
  - on completion, each processor should hold $n/p$ values of $R$

- BSP algorithm
  - Let $0 \leq j < p$ be processor id, and let $r = n/p$
    - Superstep 1
      - get elements of $M$ from other processors so that local $M' = M[j \cdot r: (j+1) \cdot r-1, : ]$
      - get elements of $V$ from other processors so that local $V' = V$
    - Superstep 2
      - perform local computation of $R' = M' \cdot V'$ and observe that $R' = R[j \cdot r: (j+1) \cdot r-1]$
      - therefore each processor holds $r = n/p$ elements of the result

- BSP cost
**BSP algorithm: Matrix * Matrix**

- **Problem:** given $A, B \in \mathbb{R}^{n \times n}$ distributed evenly over $p$ processors, compute $C = A \cdot B$
  - assume $p^{1/2}$ integral and divides $n$
  - initially each proc holds $n^2/p$ values of $A$ and $B$
  - on completion, each proc should hold $n^2/p$ values of $C$

- **BSP algorithm**
  - Let $(i,j)$ in $(0.. p^{1/2} - 1, 0.. p^{1/2} - 1)$ be the processor id, and let $s = n/p^{1/2}$
  - **Superstep 1**
    - get elts of $A$ from other processors so that $A' = A[i\cdot s: (i+1)\cdot s-1, : ]$
    - get elts of $B$ from other processors so that $B' = B[ :, j\cdot s: (j+1)\cdot s-1]$
  - **Superstep 2**
    - perform local computation of $C' = A' \cdot B'$ to compute $s \times s$ portion of $C$

- **BSP cost**
BSP cost model: units

- Goal: architecture-independent performance analysis
  - \( g \) and \( L \) are expressed in FLOPS
  - \( h \) is expressed in words (8 bytes)
    - \( g = 10 \) means 10 FLOPS can be performed for every word communicated

- Relating BSP cost to running time
  - \( T_p(n,p) = s \cdot C(n,p) \)
    - parallel running time \( T_p(n,p) \)
    - BSP cost \( C(n,p) \)
    - \( s \) is a processor-specific constant in units of seconds per flop
      - typically \( s = 1/(\text{peak MFLOPS per second}) \)
      - tends to substantially underestimate true time on many machines
BSP metrics: normalized cost

• Normalized BSP cost
  – ratio of BSP cost to optimal parallel execution

\[ \overline{C}(n, p) = \frac{T_P^{BSP}(n, p)}{W(n)/p} = a + b \cdot g + c \cdot L \]

– work efficiency goal
  • \( a \approx 1 \)

– communication efficiency goal
  • \( b \ll 1/g \)
  • \( c \ll 1/L \)
More BSP metrics: asymptotic efficiency

- Recall \[ C(n, p) = \sum_{i=1}^{c} w_i + \sum_{i=1}^{c} (h_i \cdot g + L) \]

- Asymptotic efficiency measures
  - work efficiency \( \pi \)
    - also measures load-balance
    - goal \( \pi \) close to 1
  
  - communication overhead \( \mu \)
    - goal \( \mu < 1 \)

- Examples
  - Matrix * Vector
    - \( \pi = 1, \mu = g/2 \)
    - highly dependent on network performance at all problem sizes
  
  - Matrix * Matrix
    - \( \pi = 1, \mu = 0 \)
    - insensitive to network performance, for sufficiently large problems