COMP 790-033 Parallel Computing

Lecture 10
October 19, 2022

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 between 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 $(\text{proc}, \text{addr})$
    • 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 $Out_i^j$ be the set of values leaving proc $j$ in step $i$
    - let $In_{i+1}^j$ be the set of values arriving at proc $j$ at the start of step $i+1$
    - the relation $Out_i^j \leftrightarrow In_{i+1}^j$ over all processors specifies the communication pattern
  - global synchronization
    - ensure communication phase is complete
    - ensure memory incorporates all updates (consistency)
BSP communication cost

• Definition

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

  – 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

<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>
Basic communication operations (1)

- Send n values from proc 1 to proc 3

h =

BSP communication cost =
Basic communication operations (2)

– Exchange $n$ values between proc 1 and proc 3

<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>

$h =$

BSP communication cost $=$
Basic communication operations (3)

– Send \( n \) values between proc \( i \) and proc \( H(i) \) forall \( 0 \leq i < p \), with \( H \) a permutation of \( 0:p-1 \)

<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>

\[ h = \]

BSP communication cost =
Basic communication operations (4)

– Distribute $n = kp$ values in proc 0 among $p$ procs. Each proc receives $k$ values from proc 0

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

\[
h = \]

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>

$h =$

BSP communication cost $=$
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.

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

\[ h = \]

BSP communication cost =
Basic communication operations (7)

- Broadcast \( n \) values from proc 0 to all other processors

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

\[ h = \]

BSP communication cost =
BSP programs and execution model

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

• 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
    \[ w_i = \max_{0 \leq j < p} \left( \text{work done in FLOPS on superstep } i \text{ by processor } j \right) \]
    \[ h_i = \max_{0 \leq j < p} \left( \max(\lvert Out_i^j \rvert, \lvert In_{i+1}^j \rvert) \right) \]

  – then total cost (~ running time) \( C(n, p) \) of a BSP algorithm is
    \[
    C(n, p) = \sum_{i=1}^{c} (w_i + h_i \cdot g + L)
    \]
    \[
    = \sum_{i=1}^{c} w_i + \sum_{i=1}^{c} h_i \cdot g + c \cdot L
    \]
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 $\lg p$**
    - $s := s + s'$
    - read $s$ in proc $(j + 2^{i-1}) \mod p$ into $s'$
  - **Superstep $1 + \lg p$**
    - $s := s + s'$

- **BSP cost**
BSP algorithm: Vector summation

- Problem: given $V^n$ distributed evenly over $p$ processors, find $s = \text{Sum}(V)$
  - for simplicity, assume $p$ divides $n$
  - initially processor $i$ holds $r = n/p$ values: $V[i\cdot r: (i+1)\cdot r-1]$
  - on completion, each processor holds the value of $s$

- Algorithm
  - Let $0 \leq i < p$ be processor id
  - Superstep 1
    - $w_1 = \frac{n}{p} - 1$, $h_1 = 0$
    - $s := \text{Sum} ( V[i\cdot r: (i+1)\cdot r-1] )$
    - read $s$ in proc $(i+1) \mod p$ into $s'$
  - Superstep $j$ in $2 .. 1 + \lg p$
    - $w_j = 1$, $h_j = 1$
    - $s := s + s'$
    - read $s$ in proc $(i + 2^{j-1}) \mod p$ into $s'$

- BSP cost
  \[
  C^{\text{sum}}(n,p) = \sum_{j=1}^{1+\lg p} (w_j + h_j g + L) = \left(\frac{n}{p} - 1 + \lg p\right) + (1 + \lg p) \cdot (g + L)
  \approx \frac{n}{p} + (\lg p) \cdot (g + L)
  \]
BSP alternate vector summation algorithm

• Problem: given $V^n$ distributed evenly over $p$ processors, find $s = \text{Sum}(V)$
  • for simplicity, assume $p$ divides $n$
  • initially processor $i$ holds $r = n/p$ values: $V[i \cdot r: (i+1) \cdot r-1]$
  • on completion, each processor holds the value of $s$

• Algorithm
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 * 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 $w_1 = 0, \ h_1 = nr + n$
      - 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 $w_2 = \frac{2n^2}{p}, \ h_2 = 0$
      - 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
  $$C_{MV}^{(n, p)} = \frac{2n^2}{p} + \left(\frac{n^2}{p} + n\right) \cdot g + 2 \cdot L$$
BSP algorithm: Matrix * Matrix

- Problem: given $A, B \in \mathbb{R}^{nxn}$ 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 els of $A$ from other processors so that $A' = A[i•s: (i+1)•s-1 , : ]$
      - get els of $B$ from other processors so that $B' = B[ : , j•s: (j+1)•s-1]$
    - Superstep 2
      - perform local computation of $C' = A' \cdot B'$ to compute $s \times s$ portion of $C$

- BSP cost
BSP algorithm: Matrix * Matrix

- Problem: given $A, B \in \mathbb{R}^{n \times n}$ distributed evenly over $p$ procs, 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**
    \[ w_1 = 0, \quad h_1 = 2 \left( \frac{n}{\sqrt{p}} \right) n = \frac{2n^2}{\sqrt{p}} \]
    - 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**
    \[ w_1 = (2n) \left( \frac{n}{\sqrt{p}} \right)^2 = \frac{2n^3}{p}, \quad h_1 = 0 \]
    - perform local computation of $C' = A' \cdot B'$ to compute $s \times s$ portion of $C$

- BSP cost
  \[ C^{MM}(n, p) = \frac{2n^3}{p} + \left( \frac{2n^2}{\sqrt{p}} \right) \cdot g + 2 \cdot L \]
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
### g, L, s values for some (old) machines

<table>
<thead>
<tr>
<th>Machine</th>
<th>Network topology</th>
<th>$p_{max}$</th>
<th>Bisection b/w B (MB/s)</th>
<th>Peak rate r (Mflops)</th>
<th>$g = 8r/B$ (flops/wd)</th>
<th>L (flops)</th>
<th>s (sec/flop)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>bus</td>
<td>4</td>
<td>250</td>
<td>250p</td>
<td>8p</td>
<td>1200</td>
<td>4x10^{-9}</td>
</tr>
<tr>
<td>SGI O2000</td>
<td>hypercube</td>
<td>128</td>
<td>250p</td>
<td>500p</td>
<td>16</td>
<td>800</td>
<td>2x10^{-9}</td>
</tr>
<tr>
<td>Cray T3E</td>
<td>3D Torus</td>
<td>1024</td>
<td>$600p^{2/3}$</td>
<td>900p</td>
<td>$12p^{1/3}$</td>
<td>500</td>
<td>1.1x10^{-9}</td>
</tr>
<tr>
<td>NEC SX-5</td>
<td>crossbar</td>
<td>16</td>
<td>64000p</td>
<td>8000p</td>
<td>1</td>
<td>400</td>
<td>0.13x10^{-9}</td>
</tr>
</tbody>
</table>

**Notes**

- Bisection bandwidth is for the complete network and is measured in megabytes per second
- Peak computing rate is total for $p$ processor machine and is measured in megaflops per second
BSP metrics: normalized cost

- Normalized BSP cost
  - ratio of BSP cost to optimal parallel execution

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

- work efficiency goal
  - \( a \sim 1 \)

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

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

\[
\pi = \lim_{n \to \infty} \left( \frac{c(n, p)}{\sum_{i=1}^{W(n)/p}} \right)
\]

\[
\mu = \lim_{n \to \infty} \left( \frac{c(n, p)}{\sum_{i=1}^{W(n)/p}} \right)
\]