BSP (2)
Parallel Sorting in the BSP model
Parallel sorting: problem definition

• Given
  – \( N \) values, each of size \( b \) bits
  – a total order \( \leq \) defined on the values

• Initial distribution
  – each processor holds \( n = N / p \) values

• Result

\[
\begin{array}{cccccc}
\text{proc}_0 & \text{proc}_1 & \text{proc}_2 & \ldots & \text{proc}_{p-1} \\
V_1 & V_{k_1+1} & V_{k_2+1} & \ldots & V_{k_{p-1}+1} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
V_{k_1} & V_{k_2} & V_{k_3} & \ldots & V_{k_p} \\
\end{array}
\]

– \( V_i \leq V_{i+1} \) for all \( 1 \leq i < N = k_p \)
– generally \( k_i = n \cdot i \), i.e. evenly distributed across processors
Parallel sorting: general remarks

• Typically concerned with case of $N \gg p$
  – Small $N$ problems don’t require parallel processing
  – Use algorithm cascading with efficient sequential sort of $n$ elements
    » sequential radix sort of $n$ values has $W_{SORT}(n) = \Omega(bn)$
    » sequential comparison-based sort has $W_{SORT}(n) = \Omega(n \lg n)$ and may be more appropriate when $b$ is large
  – Examine scalability in $N$ and $p$ using BSP model
    » three parallel algorithms considered
      • Bitonic sort, Radix sort, Sample sort

• What is the lower bound BSP cost for sorting?
  – Work bound
    » $(1/p) \times$ optimal sequential work $W_{SORT}(N)$
  – Communication bound
    » each value may have to move between processors from input to output
  – BSP lower bound
    $$C_p^{SORT}(N, p) \geq \frac{W_{SORT}(N)}{p} + \frac{N}{p} \cdot g + L$$
Background: Sorting networks for parallel sorting

- Basic component: the *comparator* module

```
  a  ---->  min(a,b)
   b  ---->  max(a,b)
```

- Comparator modules can be connected to form a sorting network
  - all inputs are presented in parallel
    - ex: sorting network for 4 values

```
  a  ---->  a'
   b  ---->  b'
   c  ---->  c'
   d  ---->  d'
```

**sorting network**  **schematic representation**
Sorting networks

• Sorting networks are **oblivious**
  – predetermined sequence of comparisons sorts any input sequence
  – the **depth** of a comparator is the maximum number of preceding comparators on any path to an input

• **A sorting network specifies a parallel sorting algorithm**
  – in step i, evaluate all comparators at depth i in parallel
    » each step permutes inputs to outputs (EREW)
    » at most n comparators evaluated in each step
  • let $d(n)$ be the depth of a network of size n, then $S(n) = d(n)$, $W(n) = O(n \cdot d(n))$

\[\begin{array}{c@{}c@{}c@{}c}
\text{step 1} & \text{step 2} & \text{step 3} \\
\hline
a & b & c & d \\
\hline
a' & b' & c' & d'
\end{array}\]
Bitonic Sequence

- **Definitions**
  - A sequence of values $w$ is **up-down** if $w = uv$ with $u$ increasing and $v$ decreasing
    - ex: $w = 1 \ 3 \ 5 \ 9 \ 6 \ 4 \ 3$
  - A sequence of values $w$ is **bitonic** if $w$ is a circular rotation of an up-down sequence
    - ex: $w = 5 \ 9 \ 6 \ 4 \ 3 \ 1 \ 3$
Bitonic sequence theorem

• Theorem
  – Suppose $w$ is a bitonic sequence of length $2n$ and we define sequences $r, s$ of length $n$ as follows
    $$r_i = \min(w_i, w_{n+i})$$
    $$s_i = \max(w_i, w_{n+i})$$
  then
  (1) $\forall 1 \leq i, j \leq n: \ r_i \leq s_j$ partitions the sorting problem!
  (2) $r, s$ are both bitonic sequences bitonic subproblems!

• Proof
  (by picture)
Bitonic merge

- A bitonic sequence of length \( n = 2^k \) can be sorted with a depth \( k \) sorting network
  - apply bitonic sequence theorem recursively

\[
\begin{align*}
  &w_0 &w_1 &w_2 &w_3 &w_4 &w_5 &w_6 &w_7 \\
  &\quad &\quad &\quad &\quad &\quad &\quad &\quad &\quad \\
  &\quad &\quad &\quad &\quad &\quad &\quad &\quad &\quad \\
  &\quad &\quad &\quad &\quad &\quad &\quad &\quad &\quad \\
  &\quad &\quad &\quad &\quad &\quad &\quad &\quad &\quad \\
  &\quad &\quad &\quad &\quad &\quad &\quad &\quad &\quad \\
\end{align*}
\]

one application of theorem with \( n = 8 \)

two applications of theorem with \( n = 4 \)

four applications of theorem with \( n = 2 \)
**Bitonic Sort**

- **Combine two length n bitonic merge sequences to form a length 2n bitonic sequence**
  - given two bitonic sequences $s, r$ of length $n$ let
    $w = (\text{bitonic merge } r) ++ (\text{reverse (bitonic merge } s))$
  - $w$ is a bitonic sequence of length $2n$

```
  r
  n

  s
  n

  w
  bitonic merge r
  reverse (bitonic merge s)
```

- **Bitonic sort of $n = 2^k$ values**
  - view input as $n/2$ bitonic sequences of length 2
  - combine bitonic sequences $k-1$ times to create a length $n$ bitonic sequence
  - apply final bitonic merge to yield sorted sequence

- **ex: $n = 8$**

```
  4 parallel merges of size 2

  2 parallel merges of size 4

  1 merge of size 8
```
Hypercube communication pattern

- Let \( p = 2^k \) for some \( k \geq 0 \). Processors are numbered \( 0 \leq h < p \). Let \( h^{(j)} \) be the \( j^{th} \) bit in the boolean representation of \( h \), where \( 1 \leq j \leq k \)

\[
\begin{align*}
\text{ex} & \quad p = 8, \ k = 3 \\
\text{h} &= 4 = & 1 \ 0 \ 0
\end{align*}
\]

- For \( 0 \leq h < p \), processor \( nb_j(h) \) is the neighbor of processor \( h \) in dimension \( j \). The bits of \( nb_j(h) \) are specified as follows, for \( 1 \leq r \leq k \)

\[
\{nb_j(h)\}^{(r)} = \begin{cases} 
 h^{(r)} & \text{if } r \neq j \\
 1 - h^{(r)} & \text{if } r = j 
\end{cases}
\]

\[
\text{dim 1} \quad \text{dim 2} \quad \text{dim 3}
\]
Bitonic sort of $A[0:p-1]$ using $p$ processors

- **Assumptions**
  - $p = 2^k$ and $A[h]$ is stored in variable $a$ on processor $h$
  - $CE(x,y) = (\min(x,y), \max(x,y))$

- **SPMD program for processor $h$**

```plaintext
for i := 1 to k do
    for j := i downto 1 do
        b := value of $a$ at $nb_j(h)$
        a,b := CE(a,b)
        if ($h(j) \neq h(i+1)$) then a,b := b,a
    end do
end do
```

- **BSP cost**

$$C(p) = \sum_{i=1,k} \sum_{j=1,i}(O(1) + 1 \cdot g + 2 \cdot L)$$

$$= (O(1) + 1 \cdot g + 2 \cdot L) \sum_{i=1,k} \sum_{j=1,i} 1 = (O(1) + 1 \cdot g + 2 \cdot L) \frac{k(k+1)}{2}$$

$$= O(\log^2 p)(1 + g + L)$$
Extending bitonic sort to $N > p$

- Simulate larger parallel machine
  - Let $N = np$ where $n = 2^q$ and $p = 2^k$ so $N = 2^{(k+q)}$
    
    for $i := 1$ to $k+q$ do
    for $j := i$ downto 1 do
      CE on dimension $j$

- BSP cost of CE on dimension $j$
  - lower dimensions in memory, higher dimensions across processors

  $$T_j(n) = \begin{cases} 
  O(n), & \text{if } j \leq q \\
  O(n) + n \cdot g + L, & \text{if } j > q 
  \end{cases}$$

- BSP cost for algorithm

  $$C(N, p) = \sum_{i=1}^{k+q} \sum_{j=1}^{i} T_j(N / p)$$

  $$= \left( \frac{(\lg N)(1 + \lg N)}{2} \right) \cdot O\left( \frac{N}{p} \right) + \sum_{i=q+1}^{k+q} \sum_{j=q+1}^{i} \left( \frac{N}{p} \cdot g + 2L \right)$$

  $$= \Theta(\lg^2 N) \cdot \frac{N}{p} + \Theta(\lg^2 p) \cdot \left( \frac{N}{p} \cdot g + 2L \right)$$
Improving work-efficiency

• What can be done?
  – first q iterations of outer loop create sorted sequences in processor memories
    » replace with efficient localsort (O(n) radix sort is assumed here for simplicity)
  – for each value $i > q$ in outer loop, last q iterations of inner loop perform a bitonic merge in processor memories
    » replace with efficient O(n) sequential algorithm for bitonic merge (sbmerge)

• Updated program

```plaintext
localsort(n)
for $i := q+1$ to $k+q$ do
  for $j := i$ downto $q+1$ do
    CE on dimension $j$
    sbmerge(n)
```

• BSP cost

$$C(N, p) = \Theta\left(\frac{N}{p}\right) + (\lg p)\left(\frac{1+\lg p}{2} \cdot O\left(\frac{N}{p}\right) + \frac{N}{p} \cdot g + 2L\right) + O\left(\frac{N}{p}\right)$$

$$= \Theta(\lg^2 p) \cdot \frac{N}{p} + \Theta(\lg^2 p) \cdot \left(\frac{N}{p} \cdot g + L\right)$$

Improving communication efficiency

• What can be done?
  – combine communication for up to \( \lg p \) successive CE operations

• Updated program
  localsort(n)
  for \( i := q+1 \) to \( k+q \) do
    transpose(n)
    \((i-q)\) successive CE(n) on local data
  transpose(n)
  sbmerge(n)

• BSP cost

\[
C(N, p) = \Theta\left(\frac{N}{p}\right) + (\lg p) \left(2\left(\frac{N}{p} \cdot g + L\right) + (1 + \lg p) \cdot \Theta\left(\frac{N}{p}\right) + \Theta\left(\frac{N}{p}\right)\right)
\]

\[
= \Theta(\lg^2 p) \cdot \frac{N}{p} + \Theta(\lg p) \cdot \left(\frac{N}{p} \cdot g + L\right)
\]
BSP predicted and measured times for bitonic sort

Figure 1.4  Predicted and measured execution time per key of bitonic sort on the CM-5. Times are shown to sort between 16K and 1M keys per processor on 32, 64, 128, 256 and 512 processors.
BSP breakdown of time in optimized bitonic sort

Figure 1.5  Predicted and measured execution times per key on 512 processors for the phases of bitonic sort. The time for the single gather is included in the time for the remap from a blocked to a cyclic layout; likewise, the time for the scatter is included in the time for the remap from a cyclic to a blocked layout.