Lecture 8

Bitonic sorting
Word problems

Announcements

• Today’s readings include a handout on Bitonic sort (updated on Wednesday)
  www.cse.ucsd.edu/classes/sp03/cse160/Lectures/Lec07/bitonic.pdf
Bitonic sort

- Classic parallel sorting algorithm $O(\log^2 n)$ time on $n$ processors

**Definition:** A *bitonic sequence* is a sequence of numbers $a_0, a_1...a_{n-1}$ with the following properties

- There exists an index $i$ where $a_0 \leq a_1 \leq ... \leq a_i$ and $a_i \geq a_{i+1} \geq a_{i+2} ... \geq a_{n-1}$
- We may cyclically shift the $a_k$ while maintaining this relationship

```
1,2,4,7,6,0
1,2,4,7,6,0
```

```
7,6,0,1,2,4
```

Merge property of a bitonic sequence

- We may merge two bitonic sequences in much the same way as we merge two *monotonic* sequences
Splitting property of bitonic sequences

- We can split a bitonic sequence $y$ into two bitonic sequences $L(y)$ and $R(y)$
  
  $$L(y) = \langle \min\{a_0, a_{n/2}\}, \min\{a_1, a_{n/2+1}\}, \ldots, \min\{a_{n/2+1}, a_{n-1}\} \rangle$$

  $$R(y) = \langle \max\{a_0, a_{n/2}\}, \max\{a_1, a_{n/2+1}\}, \ldots, \max\{a_{n/2+1}, a_{n-1}\} \rangle$$

- See the notes for a proof

  All values in $L(y) < R(y)$

  $L(y): 3 \ 4 \ 2 \ 1$
  $R(y): 7 \ 5 \ 8 \ 9$

Sorting a bitonic sequence is easy

- Split the bitonic sequence $y$ into two bitonic subsequences $L(y)$ & $R(y)$
- Sort $L(y)$ and $R(y)$ recursively
- Merge the two sorted lists
  - Since all values in $L(y)$ are smaller than all values in $R(y)$ we don’t need to exchange values in $L(y)$ and $R(y)$
- When $|L(.)| < 3$, sorting is trivial
- We designate $S(n)$ to be sort on of an n-element bitonic sequence
Bitonic sort algorithm

- Create a bitonic sequence $y$ from an unsorted list
- Apply the previous algorithm to sort the bitonic sequence
- We need an algorithm to create the bitonic sequence $y$

Additional properties of bitonic sequences

- Any 2 element sequence is a bitonic sequence
- We can trivially construct a bitonic sequence from two monotonic sequences, one sorted in increasing order, the other in decreasing order
Inductive construction of the initial bitonic sequence

- Form matched pairs of 2-element bitonic sequences, pointing up and down \([B(2)]\)
- Trivially merge these into 4-element bitonic sequences
- Now form matched pairs of 4-element sequences \([B(4)]\)
- Apply \(S(4)\) to each sequence, sorting the first upward, the second downward
- Trivially merge into an 8-element bitonic sequence
- Continue until there is just one sequence

Implementing the bitonic sort algorithm

- Create a bitonic sequence \(y\) from an unsorted list, \(B(n)\)
- Apply the previous algorithm to sort the bitonic sequence, \(S(n)\)
- We use comparators to re-order data
- We use a shuffle exchange network to form \(L(y)\) and \(R(y)\)
  - This network shuffles an \(n\)-element sequence by interleaving \(x_0, x_{n/2}, x_1, x_{n/2+1}, \ldots\)
Comparators

- Given two values x & y, produce two outputs
  - For an increasing comparator, the output is \(\min[x,y], \max[x,y]\)
  - For a decreasing comparator, the output is \(\max[x,y], \min[x,y]\)

Bitonic merging network

- Converts a bitonic sequence into a sorted sequence

From Introduction to Parallel Computing, V. Kumar et al, Benjamin Cummings, 1994
Bitonic conversion network

Converts an unordered sequence into a bitonic sequence

\[ B(4) = S(4) + S(2) \]

From Introduction to Parallel Computing, V. Kumar et al, Benjamin Cummings, 1994

Word problems
K-ary d-cubes

- **Definition**: A k-ary d-cube is an interconnection network with \( k^d \) nodes
  - There are \( k \) nodes along all \( d \) axis
  - End around connections
  - A generalization of a mesh and hypercube
  - The hypercube is a special case with \( k=2 \)

- Derive the diameter, number of links, and bisection width of a k-ary d-cube with \( p \) nodes

---

End around connections on a network

- Derive the running time for the ring broadcast algorithm that can take advantage of the end around connection
- Compare with the variant that doesn’t take advantage of this extra connections
Prefix sum

- The prefix sum (also called a sum-scan) of a sequence of numbers $x_k$ is in turn a sequence of running sums $S_k$ defined as follows
  - $S_0 = 0$
  - $S_k = S_{k-1} + x_k$

- Thus, scan $(3,1,4,0,2) = (3,4,8,8,10)$

- Design an algorithm for prefix sum based on the ring interconnect (with end around connections), and give the accompanying performance model

Time constrained scaling

- We may express scaled speedup as $PW/(T_P(PW,P))$

- For the summation problem of adding up $N$ numbers of $P<N$ processors….

- Determine the largest problem that can be solved in time $T=512$ time units, where $\alpha = 10$ time units, and addition costs one unit of time, for $P=1, 16, 256, \text{ and } 4096$