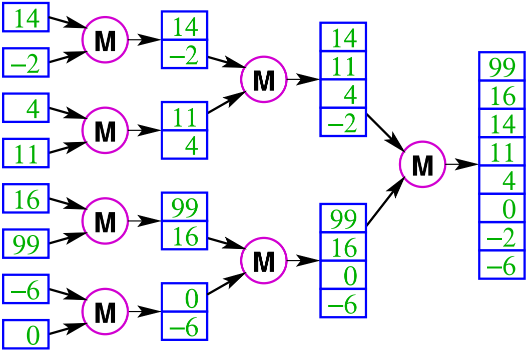
Bitonic Sort

Mark Greenstreet

CpSc 418 – Apr. 1, 2016

Parallelizing Mergesort



- We looked at this in the Mar. 30 lecture.
- The challenge is the merge step:
  - Can we make a parallel merge?

Merging and the 0-1 Principle

The main idea:

- Use divide-and-conquer.
  - Given two arrays, A_1 and A_2, divide them into smaller arrays that we can merge, and then easily combine the results.
  - What critereon should we use for dividing the arrays?
- Observation:
  - It's easy to merge two arrays of the same size, if they both have the same number of 1s.
  - If they have **nearly** the same number of 1s, that's easy as well.

Dividing the problem

- For simplicity, assume each array has an even number of elements.
  - As we go on, we'll assume that each array has an power-of-two number of elements.
  - That's the easiest way to explain bitonic sort.
  - Note: the algorithm works for arbitrary array sizes.
    - ★ See the lecture slides from 2013.
- Divide each array in the middle?
- Taking every other element?
- Other schemes?

Bitonic Sequences

- A sequence is **bitonic** if it consists of a monontonically increasing sequence followed by a monotonicially decreasing sequence.
  - Either of those sub-sequences can be empty.
  - We'll also consider a monotonically decreasing followed by monotonically increasing sequence to be bitonic.
- Properties of bitonic sequence
  - Any subsequence of a bitonic sequence is bitonic.
  - Let X be a bitonic sequence consisting of 0s and 1s. Let X_0 be the subsequence of the even-indexed elements of X, and let X_1 be odd-indexed subsequence.
  - The number of 0s in X_0 and X_1 differ by at most 1. Likewise for the number of 1s.

Bitonic Merge – big picture

- Given two sorted sequences, X_0 and X_1, note that

$$Y \;=\; X_0 \texttt{ ++ } \texttt{reverse}(X_1)$$

  is bitonic.
- Divide Y into Y_0 and Y_1, the even-indexed and odd-indexed subsequences.
  - Y_0 and Y_1 are both bitonic.
  - The number of 0s in Y_0 and Y_1 differ by at most 1. Likewise for 1s.
- Use bitonic merge (recursion) to sort Y_0 and Y_1 into ascending order to get Z_0 and Z_1.
  - The number of 0s in Z_0 and Z_1 differ by at most 1. Likewise for 1s.
- Shuffle Z_0 and Z_1 to get a list Z.
  - There can be at most one pair of out-of-order elements.
- Perform local compare-and-swap operations to get the sorted sequence, W.

Counting the 0s and 1s

The complexity of bitonic merge

Bitonic-Sort, and it's complexity

Shuffle and unshuffle

- Shuffle is like what you can do with a deck of cards:
  - Divide the deck in half
  - Select cards alternately from the two halves.
  - Shuffle is a circular-right-shift of the index bits.
    - Assuming the number of cards in the deck is a power of two.
- Unshuffle is the inverse of shuffle.
  - Unshuffling a deck of cards is dealing to two players.
  - Unshuffle is a circular-left-shift of the index bits.

Bitonic Merge and Unshuffle

The butterfly diagram – bitonic merge

The butterfly diagram – bitonic sort

Bitonic Sort in practice