The Bitonic Sort Algorithm

Mark Greenstreet and Ian M. Mitchell

CpSc 418 – February 5, 2018

Motivation

- Merge sort is a great sequential sorting algorithm
  - But, the final merge step(s) is (are) a sequential bottleneck.
  - Lower bound on run time of $\mathcal{O}(N)$ implies an upper bound on speed-up of $\sim\log N$.
- Merging and the 0-1 principle
  - We'll see that parallel merge is easy for some special cases.
- Bitonic sequences
  - How to exploit the special cases.
- Bitonic sort
  - Merge sort with a parallel merge step.
  - Run times is $O(\frac{N}{P}(\log N + (\log P)^2)) + \lambda \log N$.
  - Useful in practice. Key ideas from bitonic sort are used in other, faster parallel sorting algorithms.

0-1 Review

`x`	`-4 ≤ x`	`1 ≤ x`	`2 ≤ x`	`3 ≤ x`	`6 ≤ x`	`8 ≤ x`	`9 ≤ x`	`12 ≤ x`	`12 < x`
3	1	1	1	1	0	0	0	0	0
12	1	1	1	1	1	1	1	1	0
-4	1	0	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0	0
8	1	1	1	1	1	1	0	0	0
9	1	1	1	1	1	1	1	0	0
6	1	1	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0

- If a sorting network correctly sorts the original data, `x`.
  - Then by the `monotonicity lemma`, it correctly sorts all sequences that we can get by thresholding `x`.
  - Conversely, by showing that it sorts all of the thresholded sequences correctly, we can conclude that it sorts `x` correctly.
- Intuitively, a 0-1 argument considers **all** of the thresholded sequences at the same time.

Monotonic sequences

- A sequence, $X_0, X_1, \ldots, X_{N-1}$ is **monotonically increasing** if

$$X_0 \leq X_1 \leq \cdots \leq X_{N-1}$$

- A sequence, $X_0, X_1, \ldots, X_{N-1}$ is **monotonically decreasing** if

$$X_0 \geq X_1 \geq \cdots \geq X_{N-1}$$

- A sequence is **monotonic** if it is either monotonically increasing or monotonically decreasing.
- A sequence is **strictly** monotonically increasing if

$$X_0 < X_1 < \cdots < X_{N-1}$$

  - Likewise for strictly monotonically decreasing or strictly monotonic.
  - We won't use the "strict" versions very much – they aren't very useful with 0-1 sequences. ☺

A handy lemma

- Let X be a monotonically increasing sequence of 0s and 1s of length N. Let Y be a monotonically decreasing sequence of 0s and 1s of length N.
- Let Z be the sequence of length $2N$ with

$$\begin{array}{rcll} Z_i & = & \min(X_i, Y_i), & 0 \leq i < N \\ & = & \max(X_{i-N}, Y_{i-N}), & N \leq i < 2N \end{array}$$

- Then, either $Z_0, Z_1, \ldots, Z_{N-1}$ are all 0s, or $Z_N, Z_{N+1}, \ldots Z_{2N-1}$ are all 1s.
- Proof (details on the whiteboard):
  - Let $\texttt{zcount}(X)$ denote the number of 0s in X.
  - If $\texttt{zcount}(X) + \texttt{zcount}(Y) \geq N$, then $Z_0, \ldots, Z_{N-1}$ are all 0s.
  - If $\texttt{zcount}(X) + \texttt{zcount}(Y) \leq N$, then $Z_N, \ldots, Z_{2N-1}$ are all 1s.
  - $\square$
- What about the other half?
  - It's either $0^*1^*0^*$ or $1^*0^*1^*$.

Bitonic Sequences

- A sequence is **bitonic** if it consists of a monotonically increasing sequence followed by a monotonically 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 A be a bitonic sequence consisting of **0s** and **1s**. Let A_0 and A_1 be the even- and odd-indexed subsequences of A.
  - The number of **1s** in A_0 and A_1 differ by at most 1.
    - ⋆ We'll examine the number of **0s** on slide **??**.

The handy lemma, bitonic-version

- Let X be a bitonic sequence of 0s and 1s. Let $N = \texttt{length}(X)$. Let

$$\begin{array}{rcll} Z_i & = & \min(X_i, X_{i+\frac{N}{2}}), & 0 \le i < \frac{N}{2} \\ Z_i & = & \max(X_{i-\frac{N}{2}}, X_i), & \frac{N}{2} \le i < N \end{array}$$

  - Then either $Z_0, \dots, Z_{\frac{N}{2}-1}$ is all 0s or $Z_{\frac{N}{2}}, \dots, Z_{N-1}$ is all 1s, and
  - The other half of Z is bitonic.
  - Note: this implies that element in the lower half is $\le$ every element in the upper half.
- Proof (the easy cases):
  - If $X_0, \dots, X_{\frac{N}{2}-1}$ is all 0s, then $Z = X$, $Z_0, \dots, Z_{\frac{N}{2}-1}$ is all 0s, and $Z_{\frac{N}{2}}, \dots, Z_{N-1}$ is bitonic – it's a subsequence of a bitonic sequence.
  - Likewise, if $X_0, \dots, X_{\frac{N}{2}-1}$ is all 1s, or if $X_{\frac{N}{2}}, \dots, X_{N-1}$ is all 0s or all 1s.
  - Need to consider the case when both $X_0, \dots, X_{\frac{N}{2}-1}$ and $X_{\frac{N}{2}}, \dots, X_{N-1}$ are mixed.

Case: both halves of X are mixed

Consider the case where $X \in 0^*1^*0^*$ – the other case is equivalent.

- Let i be the smallest integer with $0 \leq i < \frac{N}{2}$ such that $X_i = 1$.
- Let j be the smallest integer with $\frac{N}{2} \leq j < N$ such that $X_j = 0$.
- If $j - i \leq \frac{N}{2}$, then
  - $Z_0, \ldots, Z_{\frac{N}{2}-1}$ is all 0s, and
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1} \in 1^*0^*1^*$.
- If $j - i \geq \frac{N}{2}$, then
  - $Z_0, \ldots, Z_{\frac{N}{2}-1} \in 0^*1^*0^*$.
  - $Z_{\frac{N}{2}}, \ldots, Z_{N-1}$ is all 1s.
- $\square$

Bitonic Sort: The big picture

Sort N values

- Divide into two halves of size $\frac{N}{2}$.
  - **Parallel:** sort each half.
  - This is a typical, divide-and-conquer approach.
  - Now, we just need to merge the two halves.
- Combine the two, sorted halves into one **bitonic** sequence of length N.
- Use the method described on slide 7 to create a clean half of length $\frac{N}{2}$ and a bitonic half of length $\frac{N}{2}$.
- Recursively merge the two halves.
  - **Parallel:** merge each half.
  - The recursion works on sequences of length N, $\frac{N}{2}$, $\frac{N}{4}$, $\ldots$, 2.
  - Total parallel time: $\log_2 N$.
  - Total number of compare-and-swaps $\frac{N}{2}\log_2 N$.

Complexity of Bitonic Sort

- The whole algorithm:
  - Use $\frac{N}{2}$ compare-and-swap operations in parallel to sort pairs of elements.
  - Perform a 4-way bitonic merge for each pair of length-2 sorted sequences to obtain a length-4 sorted sequence.
  - Perform a 8-way bitonic merge for each pair of length-4 sorted sequences to obtain a length-8 sorted sequence.
  - ...
  - Perform a N-way bitonic merge for the two length-$\frac{N}{2}$ sorted sequences to obtain the length-N sorted sequence.
- Complexity
- Parallel time:

$$\sum_{k=1} \log_2 Nk \quad = \quad O(\log^2 N)$$

- Total number of compare and swaps: $O(N \log^2 N)$.