Weekplan: Massively Parallel Computation

Philip Bille Inge Li Gørtz Christian Wulff-Nilsen

References and Reading

[1] Massively Parallel Algorithms, M. Ghaffari, 2019.

[2] Parallel Algorithms, Chapter 6, M. Ghaffari, 2019.

[3] Scribe notes from "Algorithms for Massive Data Science", Ben Mosely, 2019.

We recommend reading:

- [1] until and including the first two lines of Section 1.2.
- [2] from the last two paragraphs of page 188 until and including Section 6.7.2.
- [3] until and including Section 3.

Exercises

1 Broadcast Tree Let $S = N^\epsilon$ and suppose some machine, call it machine 1, wants to send a message of size $m = O(\sqrt{S}) = O(N^{\epsilon/2})$ to all other machines. In the first round it sends the message to $k = S/m = \Theta(\sqrt{S})$ other machines. In the second round, those machines send the message to a total of k^2 new machines, and so on. This induces a k-ary tree (called a *broadcast tree*) on the machines where machine 1 is the root.

1.1 Argue that after $O(1/\epsilon)$ rounds, the message has reached every machine, and that no machine has sent or received more than S words in any round.

2 Converge-Cast Tree In Part 1 of the Sorting Algorithm in the slides, every machine counts the number of its elements in each set S_i. It then sends these counts to machine 1 which adds them together to compute the sizes of sets S_i.

However, if every machine sends its counts directly to machine 1 in the same round, machine 1 will receive close to N words in that round, significantly exceeding its space bound of $S = N^\epsilon$.

2.1 Show how machine 1 can obtain the size of each set S_i in $O(1/\epsilon)$ rounds such that no machine sends or receives more than S words in any round. *Hint:* Use the broadcast tree from the previous exercise in reverse. We call this a *converge-cast tree*.

3 Summing with other Parameters Consider the "Summing Numbers" problem from the lecture and the parameters M and S. Solve the following exercises.

3.1 Give an efficient algorithm when $S = \Theta(N^{3/4})$ and $M = \Theta(N^{1/4})$.

3.2 Give an efficient algorithm when $S = \Theta(N^{1/4})$ and $M = \Theta(N^{3/4})$.

3.3 Give an efficient algorithm when $S = \Theta(N^\epsilon)$ and $M = \Theta(N^{1-\epsilon})$.

4 Sorting Solve the following exercises.

4.1 Explain why the MPC algorithm for sorting can be viewed as a variant of QuickSort.

4.2 When the MPC sorting algorithm from the slides terminates, each element together with its rank is stored on some machine. This is not quite the format of [2] which requires that each (element, rank)-pair is stored on the machine that *initially* held that element. Show how to convert the output from the algorithm in the slides to the required format in $O(1)$ additional rounds.

5 Minimum Spanning Forest Let $E_1, \ldots, E_k$ be a partition of an edge set E. Show that a minimum spanning forest of E can be found in the union of minimum spanning forests of subsets E_i, i.e., show:

$$\mathrm{MSF}(E) = \mathrm{MSF}\left(\bigcup_{i=1}^{k} E(M_i)\right).$$

6 Prefix Sum, Distinct Elements, and Word Count Solve the following exercises. Assume $S = M = \Theta(\sqrt{N})$

6.1 Let A be an array of N integers distributed among machines. Each entry in A is stored as $(i, A[i])$. Show how to compute the prefix sum of A (the array $P[i] = \sum_{j \leq i} A[j]$) efficiently in the MPC model. The array P should be represented similar to A.

6.2 Let L be a list of N integers. Show how to compute the number of distinct elements in L in the MPC model.

6.3 Let W be a list of N strings each of constant length. The *word count* of W is the list of pairs of distinct words and their frequency in W. Computing the word count is a classic "hello world"-exercise for the MapReduce framework. Show how to implement it efficiently in the MPC model.

7 Dynamic Programming Let S and T be strings of length N and consider the classic $O(N^2)$ time solution for computing the longest common subsequence of S and T. Show how to implement the algorithm efficiently in the MPC model. Assume $S = M = \Theta(\sqrt{N})$.