Massively Parallel 1

Eva Rotenberg

Massively Parallel Computation.

- P processors each with space S.
- Typically $S = N^{\varepsilon}$, and $P = N^{1-\varepsilon}$
- Synchronous computation in rounds: computation+communication
- Total communication into each processor $\leq S$.

Again: local computation is free. We count rounds.

Example implementation: the map-reduce framework.

Warm-up: Assume $S = \Theta(\sqrt{N})$, and $P = \Theta(\sqrt{N})$.

N numbers are distributed arbitrarily. Compute the sum?

$\underline{7, 42, 3}$ $\underline{1, 18, 2}$ $\underline{9, 10, 11}$ $\underline{4, 51, 6}$ $\underline{3, 24, 92}$ $\underline{56, 19, 8}$ $\underline{5, 22, 33}$

- Round 1: Each processor computes its sum, sends to processor 0.
- Round 2: Processor 0 computes the whole sum. Done ✓.

Massively parallel sorting

Sorting as a massive problem?

Input: numbers. $7, 42, 3, 1, 18, 2, 9, 10, 11, 4, 51, 6, 3, 24, 92, 56, 19, 8, 5, 22, 33$

Output: number and rank. $(7, 8), (42, 18), (3, 3), (1, 1), (18, 13), (2, 2), (9, 10), (10, 11), (11, 12), (4, 5), (51, 19), (6, 7), (3, 4), (24, 16), (92, 21), (56, 20), (19, 14), (8, 9), (5, 6), (22, 15), (33, 17)$

- Input is distributed in any arbitrary way amongst processors
- Output we are allowed to distribute arbitrarily, too.

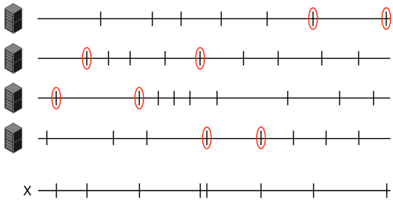
Assume S and P are $\Theta(\sqrt{N} \cdot (\log N)^c)$, aka. $\Theta(\sqrt{N}\,\mathrm{polylog}\, n)$

Idea? If only we could make each processor in charge of an interval.

Snag. We just don't know which intervals are relevant.

Remedy. Use sampling to get a grasp; coordinate in processor 0.

Massively Parallel Sorting of N elements



- Each processor samples each of its numbers with probability $\frac{2P \ln N}{N}$ and sends these sampled numbers to processor 0.
- Processor 0 broadcasts the set of samples to all processors.

What is the total sample size? Expected: $2P \ln N$

With high probability, $\leq 4P \ln N$.

Massively Parallel Sorting: Density Lemma

Let X be the samples from previous slide; i.e. every number is sampled with probability $2P \ln N$.

Lemma Consider the input numbers in their sorted order. Consider a partition of that order into P ranges, containing N/P numbers each. Then, w. high probability, all such ranges contain at ≥ 1 element from X.

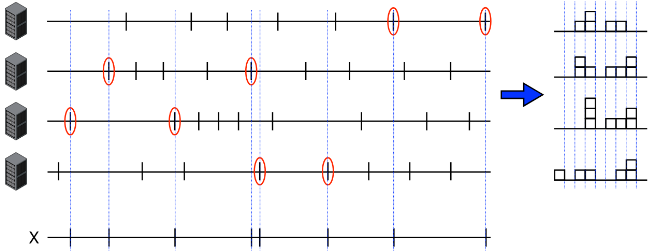
Why? Probability we do not sample some number: $1 - \frac{2P \ln N}{N}$.

There are $\frac{N}{P}$ in a range. Range empty pb: $\left(1 - \frac{2P \ln N}{N}\right)^{\frac{N}{P}} \leq e^{-2 \ln N} = \frac{1}{N^2}$

There are P ranges, and we fail if anyone is empty. Failpb: $\leq P \cdot \frac{1}{N^2} < \frac{1}{N}$.

I.e. we succeed with probability $1 - \frac{1}{N}$.

Massively Parallel Sorting: The Histogram Trick.



The set X partitions the universe of all numbers into ranges.

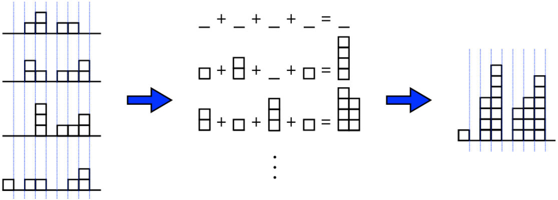
Each processor computes its histogram according to these ranges.

Space of histogram?

X numbers. $X \log N$ space. $\leq 4P \ln N \log N$ i.e. $O(P \log^2 n)$.

Challenge: We cannot just send all histograms to one processor.

Massively Parallel Sorting: Global Histogram

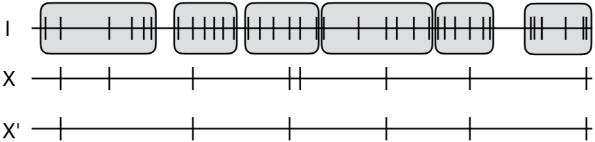


- Each processor sends count for range i to processor no. $i \mod P$.
- Processor i sums counts for its ranges and sends to processor 0.
- Processor 0 constructs global histogram.

How many ranges is Processor i responsible for? X/P. $O(\log N)$.

- Now, Processor 0 computes a set $X' \subseteq X$ with the property: Each range defined by X' has $O(N/P)$ numbers; X' has size P.
- Broadcast to all machines.
- (Exists whp by density lemma).

Massively Parallel Sorting



- Processor 0 computes a set $X' \subseteq X$ with the property: Each range defined by X' has $O(N/P)$ numbers; X' has size P.
- Broadcast to all machines.
- (Exists whp by density lemma).
- Assign each X'-range to a processor.
- Everyone sends numbers from range j to processor j.
- Locally sort your own range.
- Take global offset (from Processor 0) into account. ✓

How many rounds? $O(1)$.