University of Wrocław: Algorithms for Big Data (Fall'19) 20/01/2020

Lecture 13: MPC

Lecturer: *Przemysław Uznański* Scribe: -

1 Massively Parallel Computing

Modern distributed computing model (captures map-reduce, hadoop, spark, etc.).

- input of size n
- M machines, each of space S, $S = n^{1-\delta}$, $M \cdot S = \mathcal{O}(n)$.
- output (might be too large, e.g. size n that does not fit on a single machine)

Computation:

- computation happens over R rounds
- each machine: near linear computation per round, so total computation cost $\mathcal{O}(n^{1+o(1)}R)$
- each machine communicates $\sim S$ bits per round, so total communication cost $\mathcal{O}(nR)$

goal: minimize R

1.1 Sorting (Tera-Sort)

Intuition: if we partition input onto machines, so each machine receives contiguous fragment of size S, then we are done in a single round (each machine sorts and outputs its own part, output == concatenation of outputs).

Idea:

- each machine receives input
- each machine samples randomly from its input
- each sample is sent to single machine (1 round)
- 1 machine gathers all the samples, sorts locally, and sents back to everyone approximate histogram
- machines use approximate histogram to decide how to partition locally their input and sent it to proper receivers
- then everyone sorts their parts

Total sample size $k = \mathcal{O}(S)$, and whp histogram is with $\pm\varepsilon n$ error, where $\frac{\log n}{\varepsilon^2} \leq k$. We are ok if error is $\mathcal{O}(S)$. This is satisfied when $S^{3/2} = \Omega(n\sqrt{\log n})$, or $S = \Omega(n^{2/3}(\log n)^{1/3})$ (thus whp we can sort in $\mathcal{O}(1)$ rounds).