Massively Parallel Communication (MPC)

Inspired by frameworks such as MapReduce, Hadoop, Dryad, Spark

Clean algorithmic framework for designing algos. that can be parallelized

[KSV '10] MapReduce Class MRC

[BKS '13] MPC

Input data size : N
\# machines : M
Memory per machine: S

$S \geq N$: single machine suffices

typically $S = N^c$ $c < 1$

- Computation proceeds in synchronous rounds
- In each round, each machine operates on local data then sends/receives messages to/from other machines
- Communication sent/received $\leq S$ words per round

— ignore local computation
— Focus on # rounds

Other parameters : Replication factor : $\frac{M \cdot S}{N}$

Total work / Work Efficiency

* ignore asynchronous communication, fault tolerance

<u>Graph Problems</u> :

$n = |V|$, $m = |E|$

input size $N = m$

\# machines $M = \mathcal{O}\left(\frac{N}{S}\right)$

Three memory regimes:

① Strongly superlinear : $S = n^{1+\varepsilon}$ $\varepsilon > 0$

② Near linear $S = \tilde{O}(n)$

③ Strongly sublinear $S = n^{\alpha}$ $\alpha \in (0, 1)$

Initial data distrib^{n}: split across machines arbitrarily

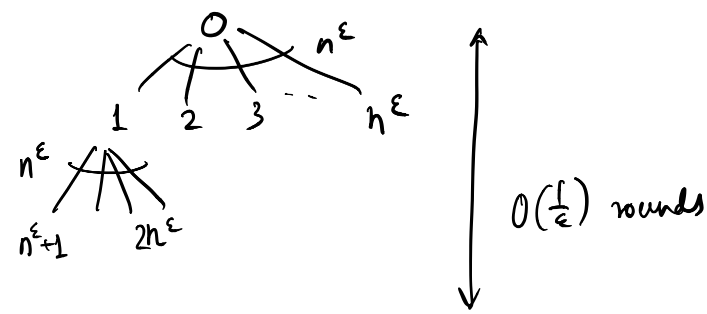
- can "load balance" using hashing in $O(1)$ rounds

S limits communication between machines in single round

How do we send / receive messages to all machines?

Say $S = n^{1+\varepsilon}$, one machine needs to send n words to all M machines

$S = n \cdot n^{\varepsilon}$



broadcast :

Converge Cast : All machines can send n word message to root with combination rule : union / intersection / sum

Computation output:

Output stored (in distributed fashion) on M machines

Sorting : machine holding item x knows rank/pos^{n} of x

Matching: machine holding vertex v knows if v is an end-pt of matched edge (and if so, other end pt.)

Connectivity: machine holding vertex v knows id. of its connected component

If S large enough, entire output on one machine

Dense graph problems : $N = m = n^{1+c}$

Superlinear memory : $S = n^{1+\varepsilon}$

[LMSV'11]: Filtering technique

MST , Matching

Minimum Spanning Tree

Idea : Partition edges into subsets of size $\leq S = n^{1+\varepsilon}$ and send each subgraph to its own machine

- Each machine computes MST of local subgraph
- throw out edge if it is heaviest edge on some cycle in local subgraph
- Any such edge is not in global MST either

Algorithm MST (V, E)

If $|E| < S$ then
compute $T^* = MST(E)$
return T^*

$\ell = 2|E|/S$

Partition E into $E_1 \ldots E_\ell$ where $|E_i| < S$
using hash f^n $h : E \longrightarrow \{1, 2 \ldots \ell\}$

In parallel : compute T_i : min. spanning tree on $G(V, E_i)$

return $MST(V, \cup T_i)$

Each iteration reduces input size by n^{ε}

Lemma: Algo MST (V, E) terminates after $\lceil \frac{c}{\varepsilon} \rceil$ iterations & returns MST

Pf: Correctness (follows from earlier discussion)

random partitioning $\Rightarrow$ w.h.p. each machine gets S edges

$\mathbb{E}[|E_i|] = \frac{S}{2}$. whp $|E_i| < S$

(Chernoff)

$$|\cup T_i| \leq l.(n-1) \leq \frac{2|E|}{S} \cdot n = O\left(\frac{|E|}{n^{\varepsilon}}\right)$$

Terminates in $\lceil \frac{c}{\varepsilon} \rceil$ rounds.

Matching:

Maximum Matching

Maximal Matching: $|\text{Maximal Matching}| \geq \frac{1}{2} |\text{Maximum Matching}|$

Idea: put subset $E' \subseteq E$ on single machine

Find maximal matching

Remove matched vertices from graph

Continue till remaining edges fit on single machine

E' : randomly selected : $|E'| \leq S$

edges decreases by $\sim n^{\varepsilon}$ in each round

Algorithm

Machine 0 is free

Edges distributed amongst machines $1 \ldots M$

$G_r(V, E_r)$: graph at round $r \in \{0, \ldots R\}$

G_0 : input graph

In each round r:

1. $m = |E_r|$
2. For $i \in \{1, \ldots M\}$ machine i marks each local edge indptly w. prob. $p = n^{1+\varepsilon}/2m$

3. For $i \in \{1 -- M\}$ machine i sends marked edges to machine 0

4. Machine 0 computes maximal matching M_r on marked edges, announces matched vertices to machines 1 to M

5. For $i \in \{1, -- M\}$, machine i discards any local edge that has a matched vertex as an end pt.

<u>Thm</u>: Algorithm terminates in $O\left(\frac{c}{\varepsilon}\right)$ rounds

<u>Lemma</u>: W.h.p. #marked edges fit onto single machine

Edges sampled w. prob $p = \frac{n^{1+\varepsilon}}{2m}$

$E[\#\text{marked edges}] = m \cdot p = \frac{n^{1+\varepsilon}}{2}$

Apply Chernoff.

<u>Lemma</u>: W.h.p. # remaining edges $\leq \frac{4m}{n^{\varepsilon}}$

<u>Pf</u>: I : unmatched vertices at end of round

No marked edge between any pair of vertices in I

If $\exists$ marked edge $\{u, v\}$, at least one of u, v will be matched. Contradiction

Consider arbitrary subset J of vertices with $\geq \frac{4m}{n^{\varepsilon}}$ induced edges

$P[\text{All induced edges in J unmarked}] \leq (1-p)^{\frac{4m}{n^{\varepsilon}}}$ $\left[\text{Recall } p = \frac{n^{1+\varepsilon}}{2m}\right]$

$\leq e^{-p \cdot \frac{4m}{n^{\varepsilon}}} = e^{-2n}$

Union bound over all such sets J (at most 2^n such)

$\Pr[\exists$ set with $\geq \frac{4m}{n^\varepsilon}$ induced edges, s.t. all unmarked$]$

$\leq 2^n \cdot e^{-2n}$

$\Rightarrow$ At most $\frac{4m}{n^\varepsilon}$ remaining edges at end of round w.h.p.

<u>Lemma:</u> Algo terminates in $R = O(\frac{c}{\varepsilon})$ rounds

$O(\log n)$ rounds with near linear memory

[Gheffari et al '18] : $O(1)$ approx in $O(\log\log n)$ rounds w. near linear memory

[Gheffari, Uitto '19] : $O(1)$ approx in $\tilde{O}(\sqrt{\log \Delta})$ with n^α memory $\alpha \in (0, 1)$

$O(1)$ approx can be turned into $(1+\varepsilon)$ approx using ideas of [McGregor '05]

$\sim (\frac{1}{\varepsilon})^{1/\varepsilon}$ overhead.