# Nimble Algorithms for Cloud Computing

Ravi Kannan, Santosh Vempala and David Woodruff

# Cloud computing

Data is distributed arbitrarily on many servers

Parallel algorithms: time

Streaming algorithms: sublinear space

Cloud Complexity: time, space *and communication*

[Cormode-Muthukrishnan-Ye 2008]

**Nimble algorithm**: polynomial time/space (as usual) and sublinear (ideally polylog) communication between servers.

# Cloud vs Streaming

- Streaming algorithms make small "sketches" of data
- Nimble algorithms must communicate small "sketches"
- Are they equivalent?

Simple observation:

- Communication in cloud = O(memory in streaming)

[Daume-Philips-Saha-Venkatasubramanian12]

- Is cloud computing more powerful?

# Basic Problems on large data sets

- Frequency moments
- Counting copies of subgraphs (homomorphisms)
- Low-rank approximation
- Clustering

- …
- Matchings
- Flows
- Linear programs

# Streaming Lower Bounds

Frequency moments: Given a vector of frequencies $f=(f_1, f_2, \ldots, f_n)$ presented as a set of increments, estimate $\|f\|_k = \sum_i f_i^k$ to relative error $\epsilon$.

[Alon-Matias-Szegedy99, Indyk-Woodruff05]:

$\theta(n^{1-2/k})$ space (k =1,2 by random projection)

Counting homomorphisms: Estimate #triangles, $\#C_4$, $\#K_{r,r}$ … in a large graph G.

$\Omega(n^2)$ space lower bounds in streaming.

# Streaming Lower Bounds

Low-rank approximation: Given n x d matrix A, find $A$ of rank k s.t.

$$\|A - A\|_F \leq (1+\epsilon)\|A - A_k\|_F$$

[Clarkson-Woodruff09]

Any streaming algorithm needs $\Omega((n+d)k \log nd)$ space.

# Frequency moments in the cloud

- Lower bound via multi-player set disjointness.

- t players have sets $S_1, S_2, \ldots, S_t$, subsets of [n]
- Problem: determine if sets are disjoint or have one element in common.

- Thm: Communication needed = $\Omega(n/t \log t)$ bits.

# Frequency moments in the cloud

Thm. Communication needed to determine set disjointness of t sets is $\Omega(n/t\log t)$ bits.

Consider s sets being either

(i) completely disjoint or (ii) with one common element

(each set is on one server)

Then k'th frequency moment is either $n$ or $n-1+s^k$

Suppose we have a factor 2 approximation for the k'th moment. With $s^k = n+1$, then we can distinguish these cases. Therefore, communication needed is $\Omega(s^{k-1})$.

# Frequency moments in the cloud

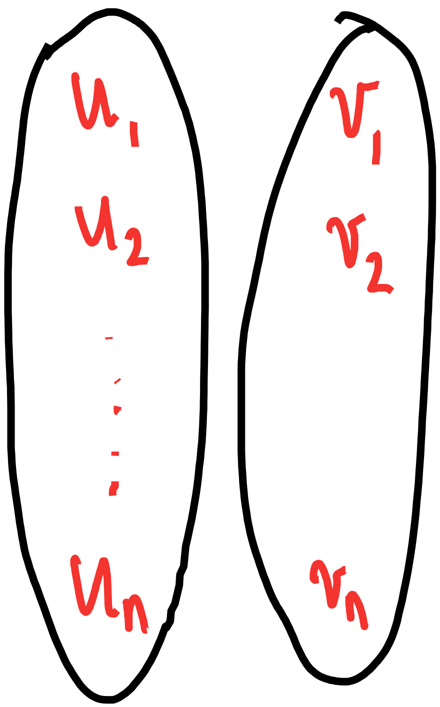
Thm. [Kannan-V.-Woodruff13]

Estimating k'th frequency moment on s servers takes $O(s^k/\epsilon^2)$ words of communication, with $O(b+\log n)$ bits per word.

- Lower bound is $s^{k-1}$
- Previous bound: $s^{k-1} (\log n/\epsilon)^{O(k)}$ [Woodruff-Zhang12]
- streaming space complexity is $n^{1-2/k}$

Main idea of algorithm: sample elements within a server according to higher moments.

# Warm-up: 2 servers, third moment




Goal: estimate $\sum_i (u_i + v_i)^3$

1. Estimate $\sum_i u_i^3$
2. Sample j w.p. $p_j = u_j^3 / \sum_i u_i^3$ ; announce
3. Second server computes $X = u_j^2 v_j / p_j$
4. Average over many samples.

$E(X) = \sum_i u_i^2 v_i$

# Warm-up: 2 servers, third moment

Goal: estimate $\sum_{i} (u_i + v_i)^3$

$p_j = u_j^3 / \sum_{i} u_i^3$ $\quad X = u_j^2 v_j / p_j$ $\quad E(X) = \sum_{i} u_i^2 v_i$

$Var(X) \leq \sum_{i: v_i > 0} (u_i^2 v_i)^2 / p_i$

$\leq \sum_{i} u_i^3 \sum_{i} u_i v_i^2$

$\leq (\sum_{i} u_i^3 + v_i^3)^2$

So, $O(1/\epsilon^2)$ samples suffice.

# Many servers, k'th moment

GOAL: $\sum_{i=1}^{n} \left( \sum_{j=1}^{s} f_{ij} \right)^{k}$

$$= \sum_{i} \sum_{\substack{r_1 \ldots r_s \\ \sum r_j = k}} \binom{k}{r_1\ r_2\ \ r_s} \prod_{j} f_{ij}^{r_j}$$

$$= \sum_{r_1 \ldots r_s} \binom{k}{r_1 \ldots r_s} \underbrace{\sum_{i} \prod_{j} f_{ij}^{r_j}}$$

# Many servers, k'th moment

GOAL: $$\sum_{i} f_{i_1}^{r_1} \cdots f_{i_m}^{r_m} \quad \left(\sum r_j = k\right)$$

Each server j:

- Sample i w. prob $p_i = f_{ij}^k / \sum_t f_{tj}^k$ according to k'th moment.
- Every j' sends $f_{ij'}$ if j' < j and $f_{ij'} < f_{ij}$

  or j'>j and $f_{ij'} \le f_{ij}$
- Server j computes $X_i = \prod_{j=1}^{s} f_{ij'}^{r_j} / p_i$

# Many servers, k'th moment

Each server j:

- Sample i w. prob $p_i = f_{ij}^k / \sum_t f_{tj}^k$ according to k'th moment.
- Every j' sends $f_{ij'}$ if j' < j and $f_{ij'} < f_{ij}$
  or j'>j and $f_{ij'} \leq f_{ij}$
- Server j computes $X_i = \prod_{j=1}^{s} f_{ij'}^{r_j} / p_i$

Lemma. $E(X) = \sum_{R_j} \prod_j f_{ij}^{r_j}$ and $Var(X) \leq (\sum_i f_{ij}^k)^2$

Theorem follows as there are $< s^k$ terms in total.

# Counting homomorphisms

- How many copies of graph H in large graph G?
- E.g., H = triangle, 4-cycle, complete bipartite etc.
- Linear lower bounds for counting 4-cycles, triangles.
- We assume an (arbitrary) partition of the vertices among servers.

# Counting homomorphisms

- To count number of paths of length 2, in a graph with degrees $d_1$, $d_2$, …, $d_n$, we need:

$$t(K_{1,2}, G) = \sum_{i=1}^{n} \binom{d_i}{2}$$

This is a polynomial in frequency moments!

- #stars is $t(K_{1,r}, G) = \sum_{i=1}^{n} \binom{d_i}{r}$
- #C4's: let $d_{ij}$ is the number of common neighbors of i and j. Then,

$$t(C_4, G) = \sum_{i=1}^{n} \binom{d_{ij}}{2}$$

- #$K_{a,b}$: let $d_S$ be the number of common neighbors of a set of vertices S. Then,

$$t(K_{a,b}, G) = \sum_{S \subset V, |S| = a} \binom{d_S}{b}$$

# Low-rank approximation

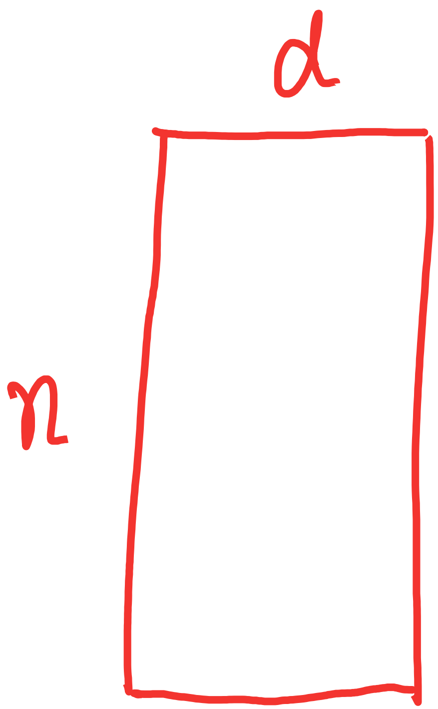
Given n x d matrix A partitioned arbitrarily as $A = A_1 + A_2 + \ldots + A_s$ among s servers, find $A$ of rank k s.t. $||A - A||_F \leq (1+\epsilon) OPT$.

To avoid linear communication, on each server t, we leave a matrix $A_t$, s.t. $A = A_1 + A_2 + \ldots + A_s$ and is of rank k.

How to compute these matrices?

# Low-rank approximation in the cloud

Thm. [KVW13]. Low-rank approximation of n x d matrix A partitioned arbitrarily among s servers takes $O\uparrow* (skd)$ communication.



# Warm-up: row partition

- Full matrix A is n x d with n >> d.
- Each server j has a subset of rows $A_j$

- Computes $A_j^T A_j$ and sends to server 1.
- Server 1 computes $B = \sum_{j=1}^{s} A_j^T A_j$ and announces V, the top k eigenvectors of B.
- Now each server j can compute $A_j V V^T$.

- Total communication = $O(sd^2)$.

# Low-rank approximation: arbitrary partition

- To extend this to arbitrary partitions, we use limited-independence random projection.

- Subspace embedding: matrix P of size $O(d/\epsilon^2) \times n$ s.t. for any $x \in R^d$, $\|PAx\| = (1 \pm \epsilon)\|Ax\|$.

- Agree on projection $P$ via a random seed
- Each server computes $PA_t$, sends to server 1.
- Server 1 computes $PA = \sum_t PA_t$ and its top k right singular vectors V.
- Project rows of A to V.

- Total communication = $O(sd^2/\epsilon^2)$.

# Low-rank approximation: arbitrary partition

- Agree on projection $P$ via a random seed
- Each server computes $PA_t$, sends to server 1.
- Server 1 computes $PA = \sum_t PA_t$ and its top k right singular vectors V.
- Project rows of A to V.

Thm. $\|A - AVV^T\| \le (1 + O(\epsilon)) OPT$.

Pf. Extend V to a basis $v_1, v_2, \ldots, v_d$. Then,

$$\|A - AVV^T\|_F^2 = \sum_{i=k+1}^{d} \|Av_i\|^2 \le (1+\epsilon)^2 \sum_{i=k+1}^{d} \|PAv_i\|^2.$$

And, with $u_1, u_2, \ldots, u_d$ singular vectors of A,

$$\sum_{i=k+1}^{d} \|PAv_i\|^2 \le \sum_{i=k+1}^{d} \|PAu_i\|^2 \le (1+\epsilon)^2 \sum_{i=k+1}^{d} \|Au_i\|^2$$

$$= (1 + O(\epsilon)) OPT^2.$$

# Low-rank approximation in the cloud

To improve to O(skd), we use a subspace embedding up front, and observe that O(k)-wise independence suffices for the random projection matrix.

- Agree on $O(k/\epsilon) \times n$ matrix S and $O(k/\epsilon^2) \times n$ matrix P.
- Each server computes $SA_t$ and sends to server 1.
- S1 computes $SA = \sum_t SA_t$ and an orthonormal basis $U^T$ for its row space.
- Apply previous algorithm to $AU$.

# K-means clustering

- Find a set of k centers $c_1, c_2, \ldots, c_k$ that minimize
$\sum_{i \in S} \text{Min}_{j=1}^{k} ||A_i - c_j||^2$

- A near-optimal (i.e. $1+\epsilon$) solution could be very different!

- So, cannot project up front to reduce dimension and approximately preserve distances.

# K-means clustering

- Kannan-Kumar condition:
- Every pair of cluster centers are f(k) standard deviations apart.
- "variance": maximum over 1-d projections, of the average squared distance of a point to its center.

(e.g. for Gaussian mixtures, max directional variance)

- Thm. [Kannan-Kumar10]. Under this condition, projection to the top k principal components followed by the k-means iteration starting at an approximately optimal set of centers finds a nearly correct clustering.

- Finds centers close to the optimal ones, so that the induced clustering is same for most point.

# K-means clustering in the cloud

- Points (rows) are partitioned among servers
- Low-rank approximation to project to SVD space.

- How to find a good starting set of centers?
- Need a constant-factor approximation.

- Thm [Chen]. There exists a small subset ("core") s.t. the k-means value of this set (weighted) is within a constant factor of the k-means value of the full set of points (for any set of centers!).
- Chen's algorithm can be made nimble.

**Thm.** K-means clustering in the cloud achieves the Kannan-Kumar guarantee with $O(d^2 + k^4)$ communication on s = O(1) servers.

# Cloud computing: What problems have nimble algorithms?

- Approximate flow/matching?
- Linear programs
- Which graph properties/parameters can be checked/estimated in the cloud?

  (e.g., planarity? expansion? small diameter?)
- Other Optimization/Clustering/
- Learning problems

  [Balcan-Blum-Fine-Mansour12,

  Daume-Philips-Saha-Venkatasubramaian12]

- Random partition of data?

- Connection to property testing?