# Communication-Efficient Computation on Distributed Noisy Datasets

Qin Zhang

Indiana University Bloomington

SPAA'15
June 15, 2015

# Model of computation

**The coordinator model**: $k$ sites and 1 coordinator.

– each site has a 2-way communication channel with the coordinator.

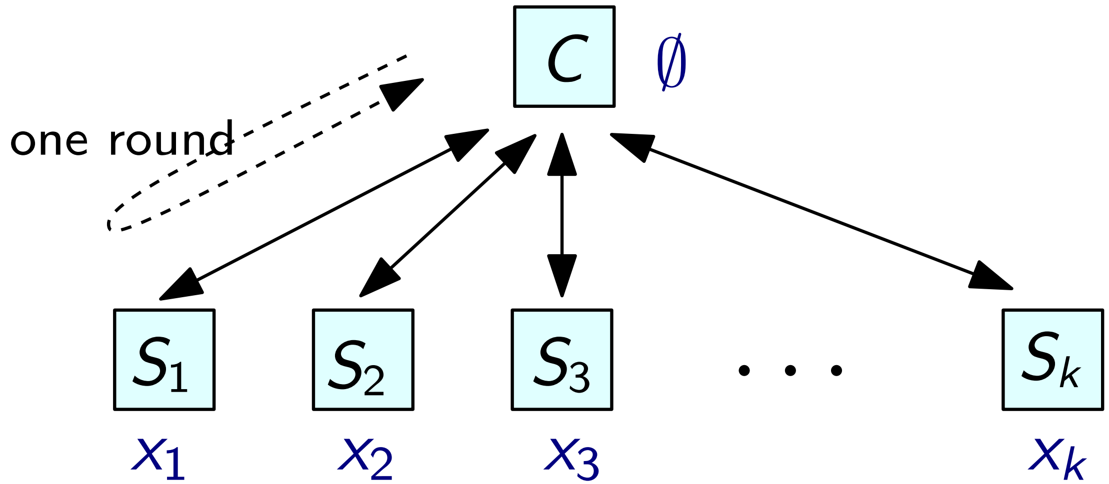
– each site $S_i$ has a piece of data $x_i$. The coordinator has $\emptyset$.

– **Task**: compute $f(x_1, \ldots, x_k)$ together via communication.
The coordinator reports the answer.

– computation is divided into rounds.

– **Goal**: minimize both

- total #bits of comm. (o(Input); best polylog(Input))
- and #rounds ($O(1)$ or polylog(Input)).



# Model of computation

**The coordinator model**: $k$ sites and 1 coordinator.

– each site has a 2-way communication channel with the coordinator.

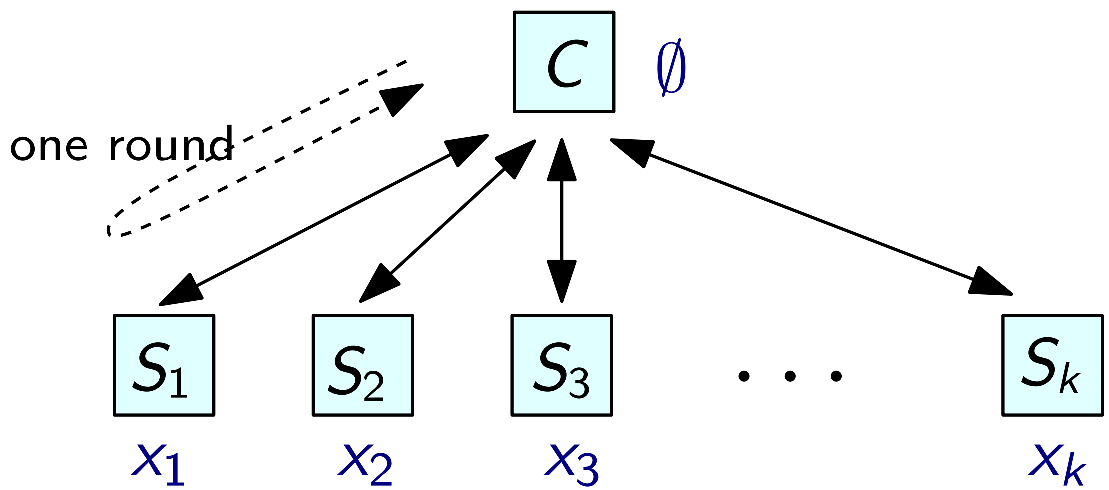
– each site $S_i$ has a piece of data $x_i$. The coordinator has $\emptyset$.

– **Task**: compute $f(x_1, \ldots, x_k)$ together via communication. The coordinator reports the answer.

– computation is divided into rounds.

– **Goal**: minimize both

- total #bits of comm. (o(Input); best polylog(Input))
- and #rounds ($O(1)$ or polylog(Input)).

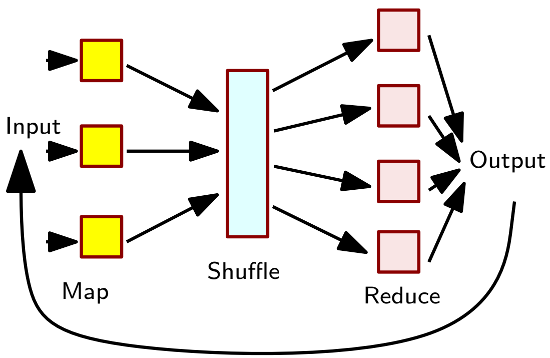



– no constraint on #bits can be sent by each site on each round. (usually balanced)

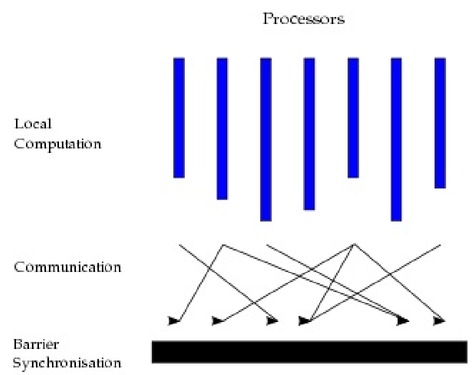
– do not count local computation (usually linear)

# The coordinator model (cont.)

**Communication → time, energy, bandwidth, . . .**

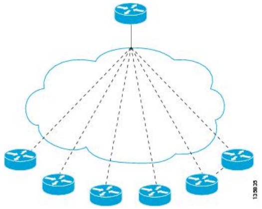



The MapReduce model.




The BSP model.

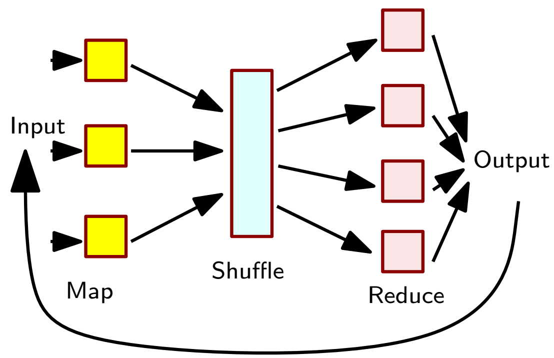
Also network monitoring, sensor networks, etc.



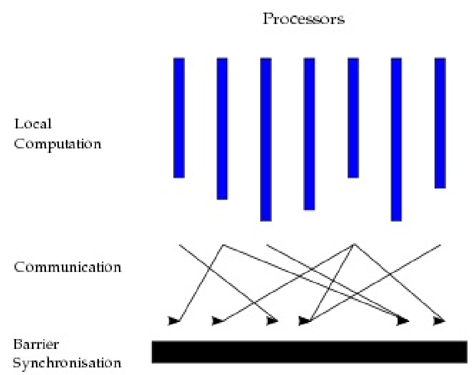
Abstraction

# The coordinator model (cont.)

**Communication $\rightarrow$ time, energy, bandwidth, ...**

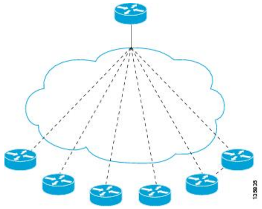



The MapReduce model.



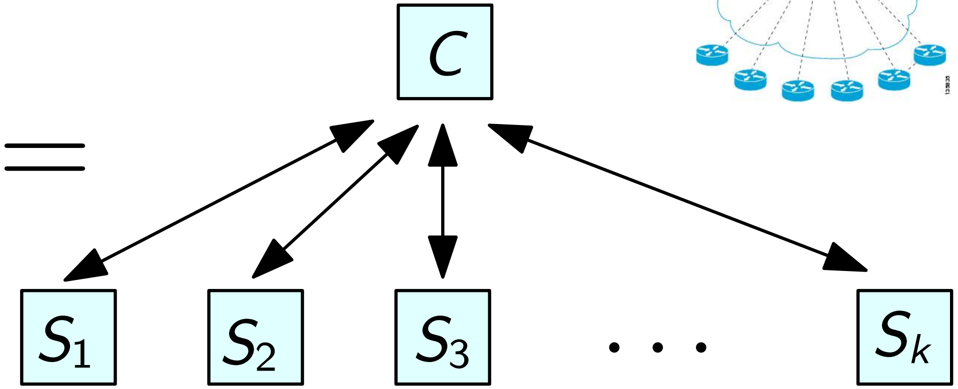

The BSP model.

Also network monitoring, sensor networks, etc.



Abstraction

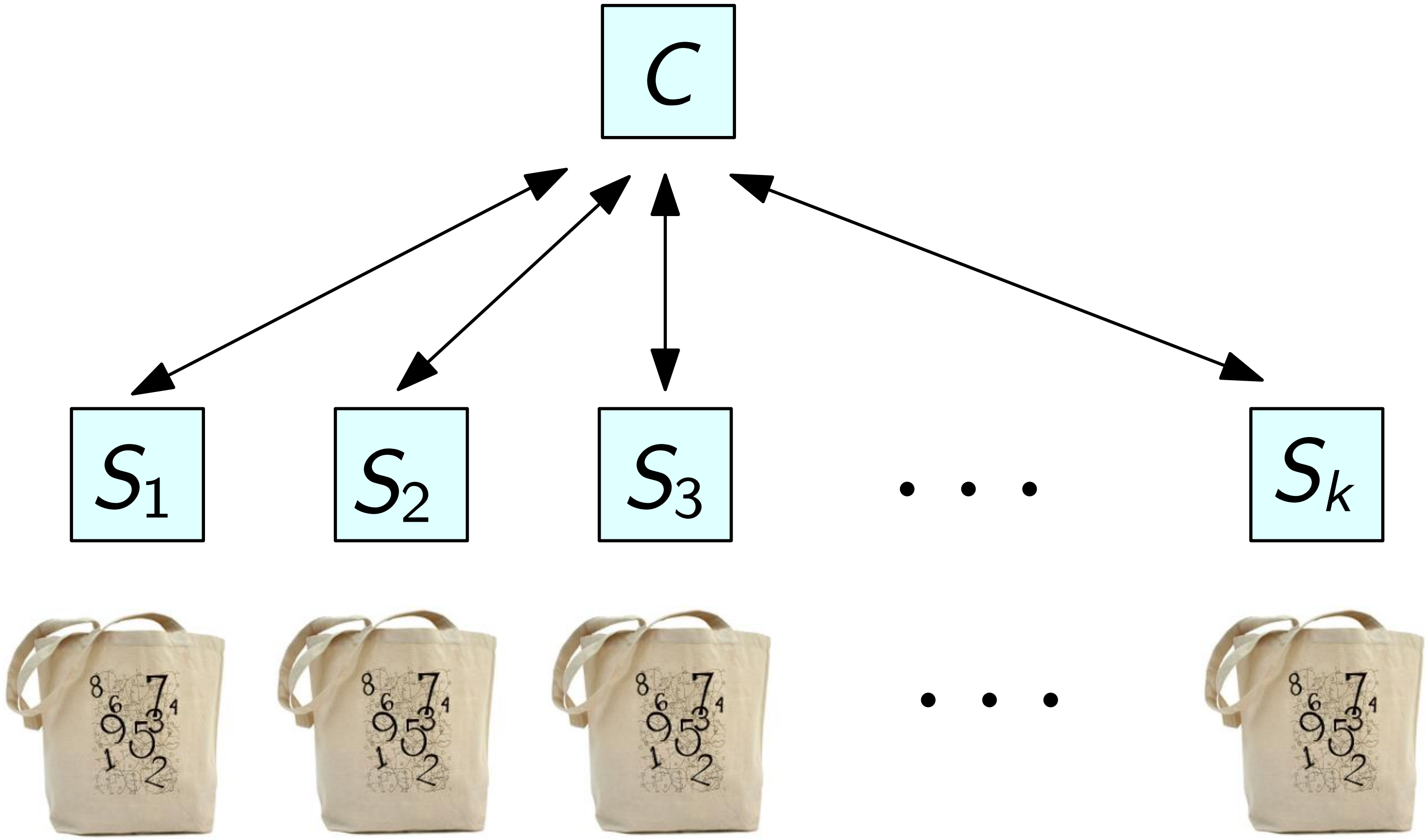
=



# The distributed distinct elements ($F_0$) problem

Function $f$ can be:

How many distinct elements ($F_0$) in the **union** of the $k$ bags?

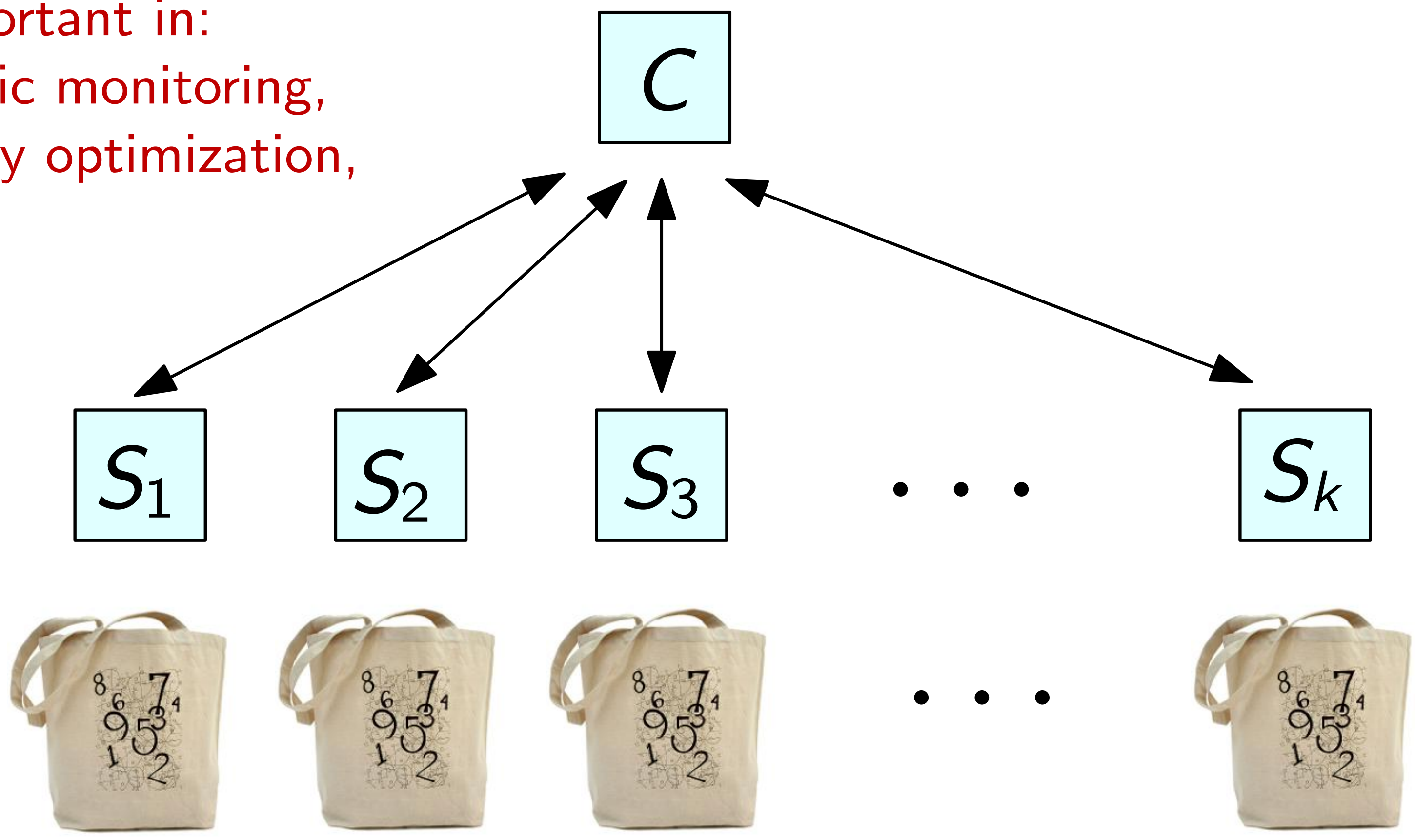


# The distributed distinct elements ($F_0$) problem

Function $f$ can be:

How many distinct elements ($F_0$) in the **union** of the $k$ bags?

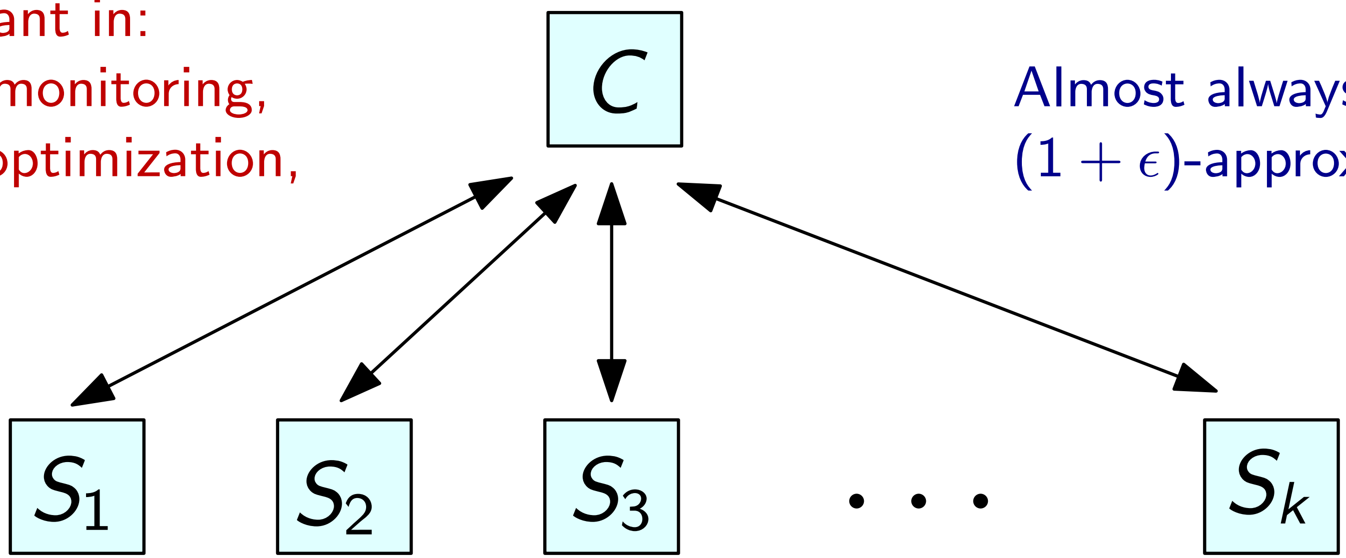
Important in:
traffic monitoring,
query optimization,
...



# The distributed distinct elements ($F_0$) problem

Function $f$ can be:
How many distinct elements ($F_0$) in the **union** of the $k$ bags?

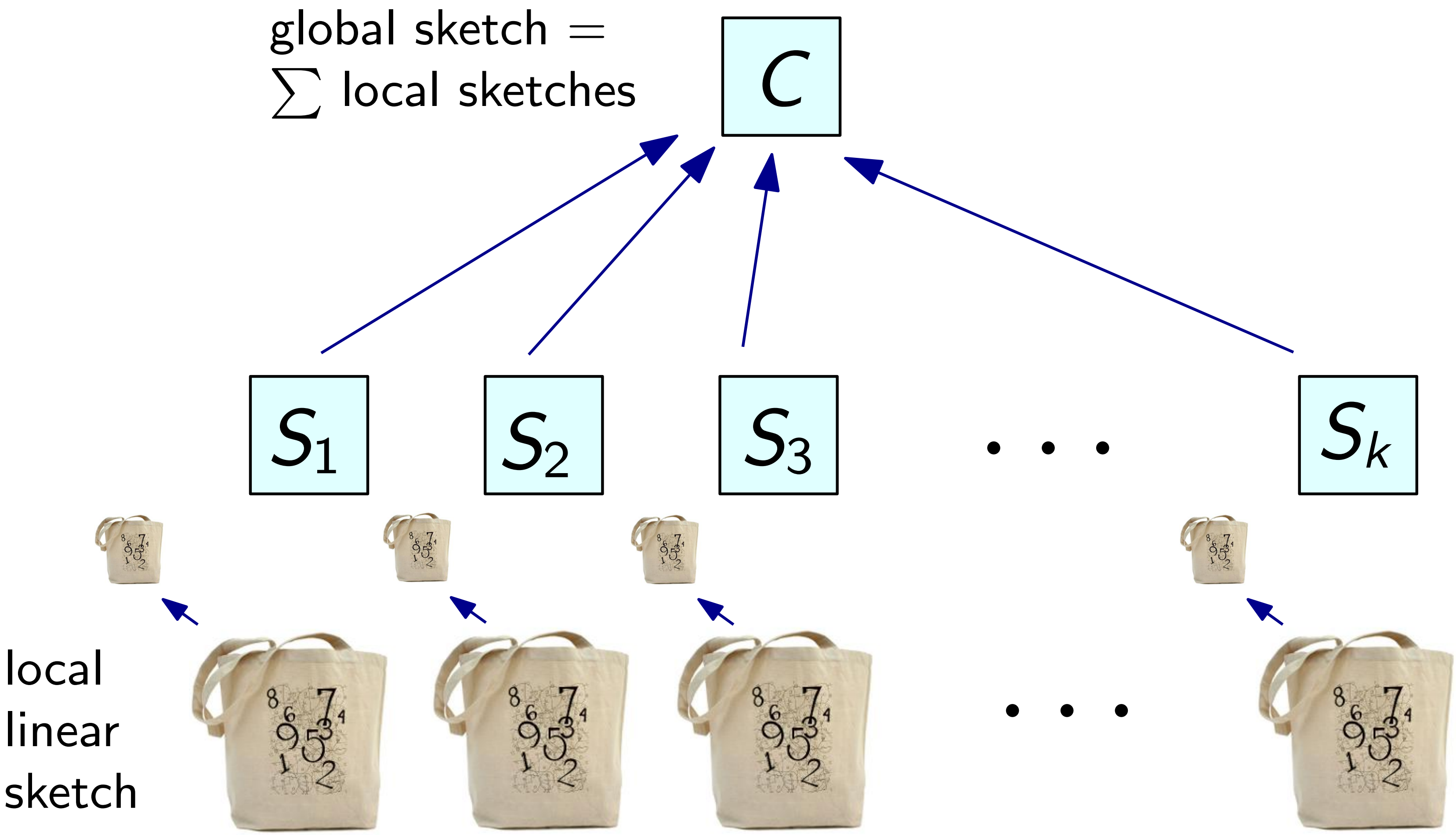
Important in:
traffic monitoring,
query optimization,
...




Almost always allow a $(1+\epsilon)$-approximation

# Existing solution – linear sketches

How many distinct elements ($F_0$) in the **union** of the $k$ bags?



# Linear sketches

- **Random linear mapping** $M : R^n \to R^k$ where $k \ll n$.

$$\underbrace{\begin{bmatrix} \quad & M & \quad \end{bmatrix}}_{\text{linear mapping}} \begin{bmatrix} x \\ \\ \\ \end{bmatrix} = \underbrace{\begin{bmatrix} Mx \end{bmatrix}}_{\text{sketching vector}} \longrightarrow \text{(approximate) } f(x)$$

The data. e.g.,
a frequency vector

# Linear sketches

- **Random linear mapping** $M : R^n \to R^k$ where $k \ll n$.

$$\begin{bmatrix} \quad & M & \quad \end{bmatrix} \begin{bmatrix} \\ x \\ \\ \end{bmatrix} = \begin{bmatrix} Mx \end{bmatrix} \longrightarrow \text{(approximate) } f(x)$$

linear mapping

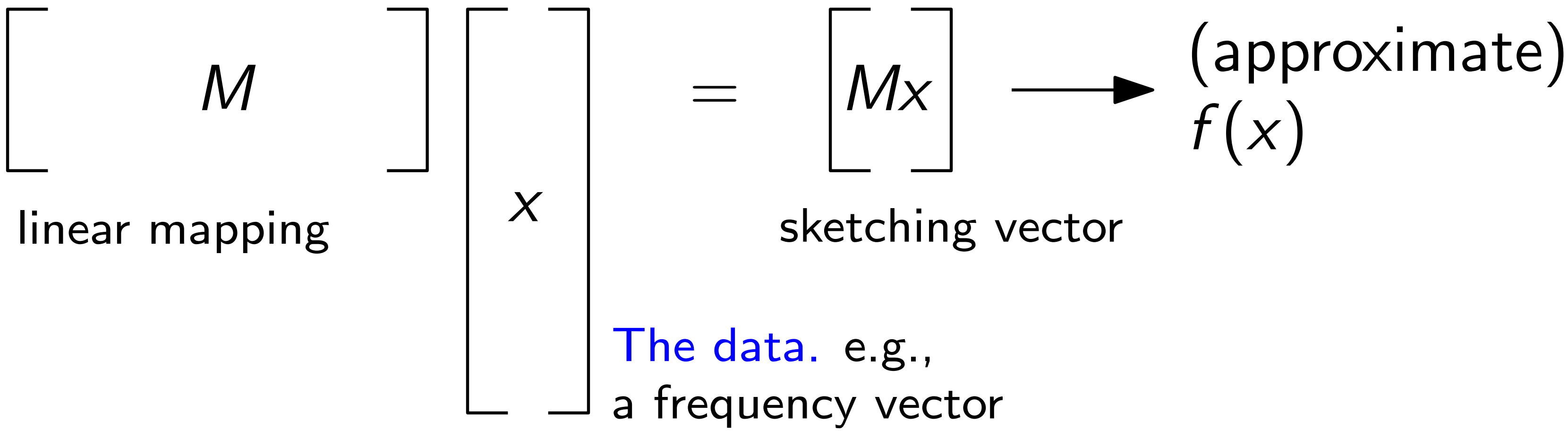
sketching vector

The data. e.g.,
a frequency vector

- **Simple and useful**: Statistical/graph/algebraic problems in data streams, compressive sensing, . . .

# Linear sketches

- **Random linear mapping** $M : R^n \to R^k$ where $k \ll n$.

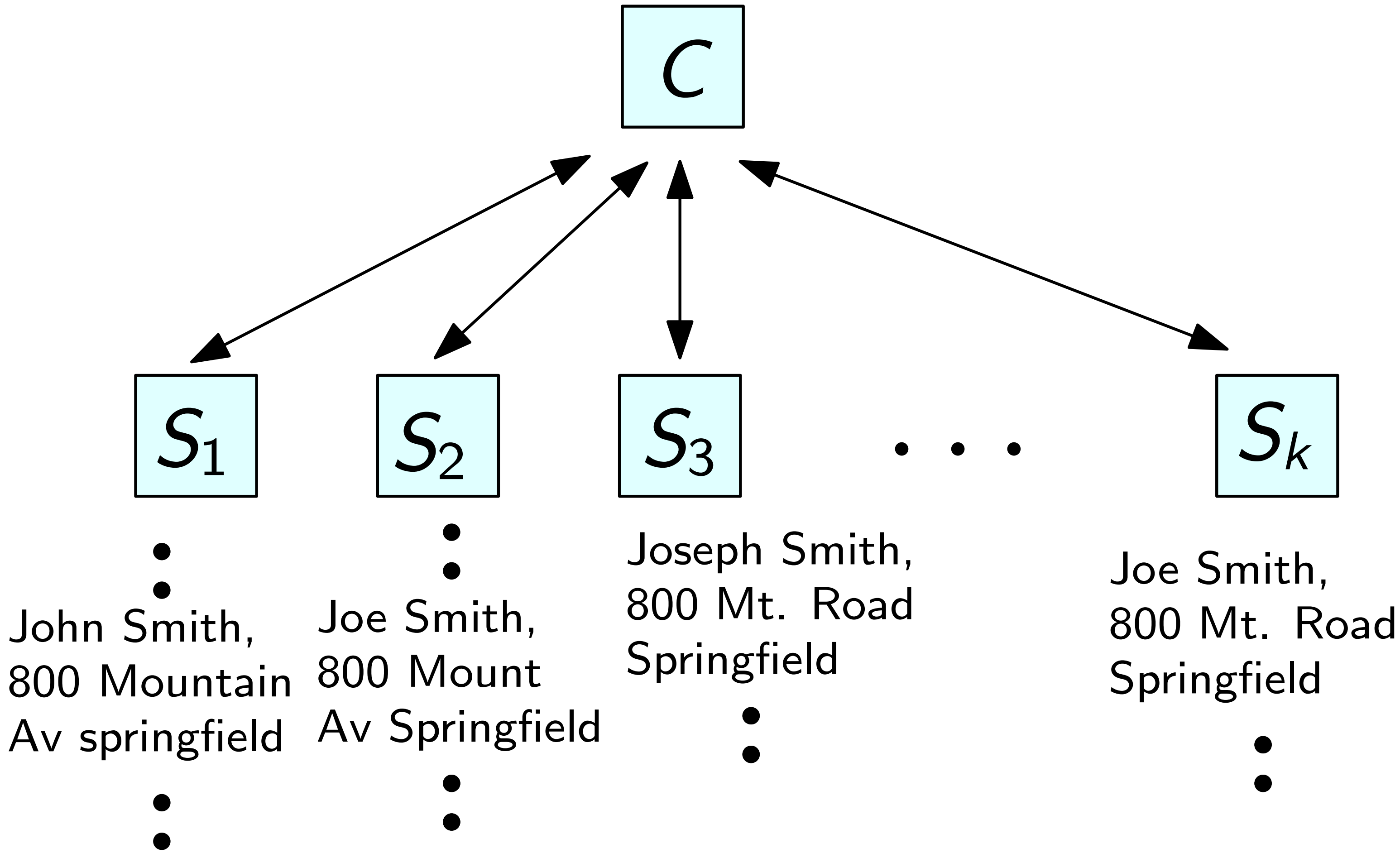



- **Simple and useful**: Statistical/graph/algebraic problems in data streams, compressive sensing, ...

- **Perfect for distributed computation**
  The data is distributed as $x = x_1 + \ldots + x_k$; $x_i$ on site $i$.
  Merge using linearity: $Mx_1 + \ldots + Mx_k = M(x_1 + \ldots + x_k)$

# Linear sketches cannot work for noisy datasets

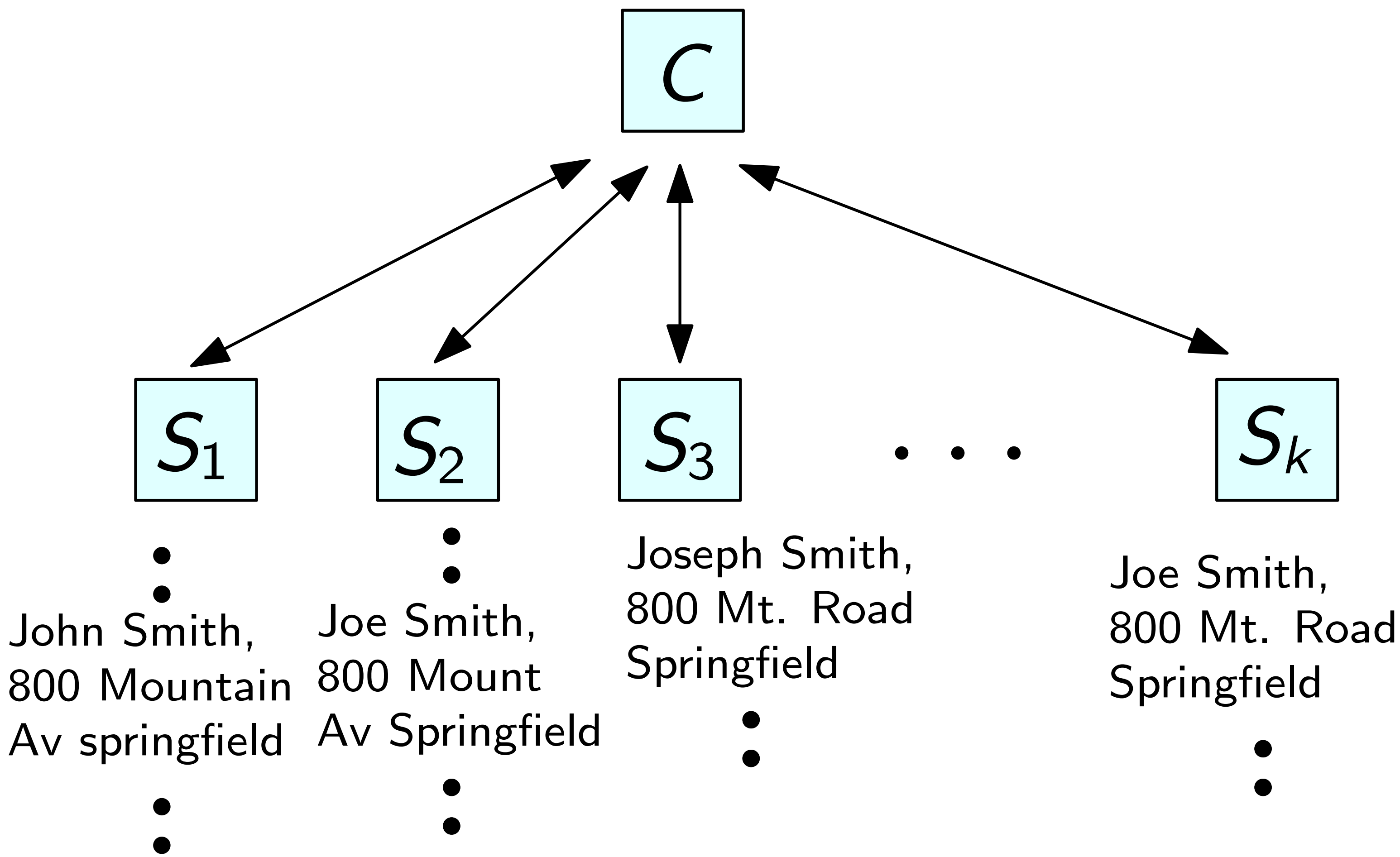
Real world distributed datasets are often noisy!



# Linear sketches cannot work for noisy datasets

Real world distributed datasets are often noisy!

We (have to) consider similar items as one element. Then how to compute $F_0$?



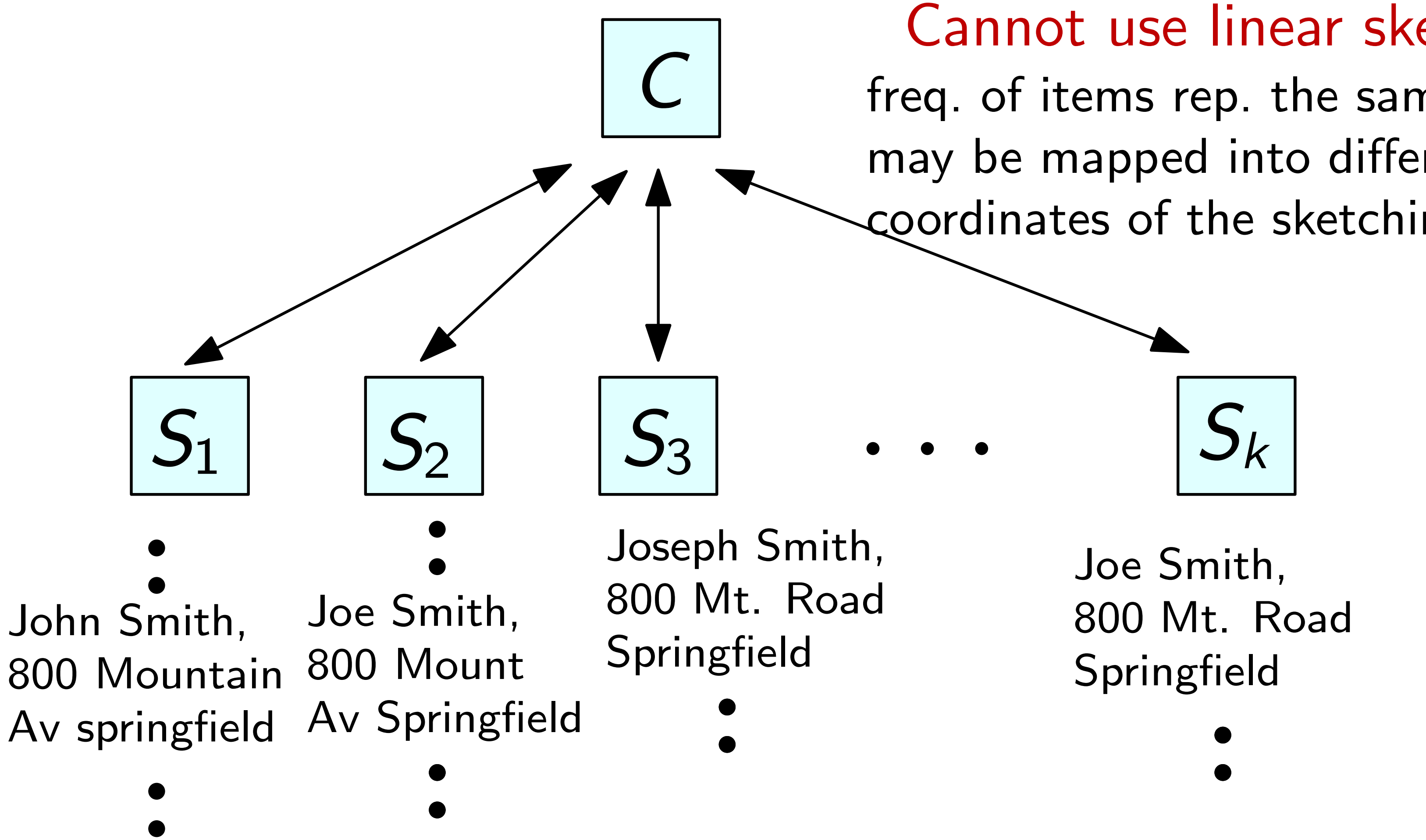
# Linear sketches cannot work for noisy datasets

Real world distributed datasets are often noisy!

We (have to) consider similar items as one element. Then how to compute $F_0$?

Cannot use linear sketches

freq. of items rep. the same entity may be mapped into different coordinates of the sketching vector



# Noisy data is universal

- Music, Images, ...
  After compressions, resize, reformat, etc.

# Noisy data is universal

- 


  Music, Images, ...
  After compressions, resize, reformat, etc.

- 


  "SPAA 2015"

  "27th ACM Symposium on Parallelism in Algorithms and Architectures"

  "ACM FCRC SPAA'15"

Queries of the same meaning sent to Google

# Related to Entity Resolution

- Related to Entity Resolution: Identify and link/group different manifestations of the same real world object.

  Very important in data cleaning / integration. Have been studied for 40 years in DB, also in AI, NT.

  E.g. [Gill& Goldacre'03, Koudas et al.'06, Elmagarmid et al.'07, Herzog et al.'07, Dong& Naumann'09, Willinger et al.'09, Christen'12] for introductions, and [Getoor and Machanavajjhala'12] for a toturial.

  Centralized, detect items representing the same entity, merge/output all distinct entities.

# Related to Entity Resolution

- Related to Entity Resolution: Identify and link/group different manifestations of the same real world object.

  Very important in data cleaning / integration. Have been studied for 40 years in DB, also in AI, NT.

  E.g. [Gill& Goldacre'03, Koudas et al.'06, Elmagarmid et al.'07, Herzog et al.'07, Dong& Naumann'09, Willinger et al.'09, Christen'12] for introductions, and [Getoor and Machanavajjhala'12] for a toturial.

  Centralized, detect items representing the same entity, merge/output all distinct entities.

- This work: distributed, statistical estimations,

# Related to Entity Resolution

- Related to Entity Resolution: Identify and link/group different manifestations of the same real world object.

  Very important in data cleaning / integration. Have been studied for 40 years in DB, also in AI, NT.

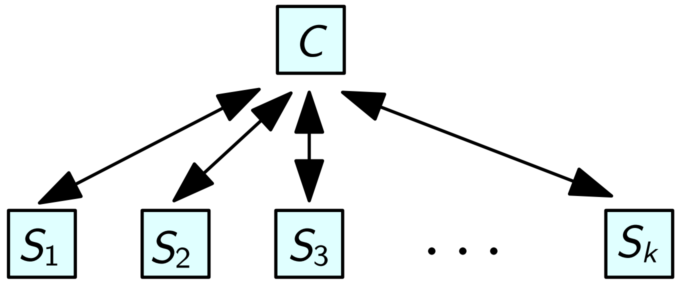
  E.g. [Gill& Goldacre'03, Koudas et al.'06, Elmagarmid et al.'07, Herzog et al.'07, Dong& Naumann'09, Willinger et al.'09, Christen'12] for introductions, and [Getoor and Machanavajjhala'12] for a toturial.

  Centralized, detect items representing the same entity, merge/output all distinct entities.

- This work: distributed, statistical estimations,

  We want more communication-efficient algorithms (o(input size)), without a comprehensive de-duplication.

# Our goal and problem



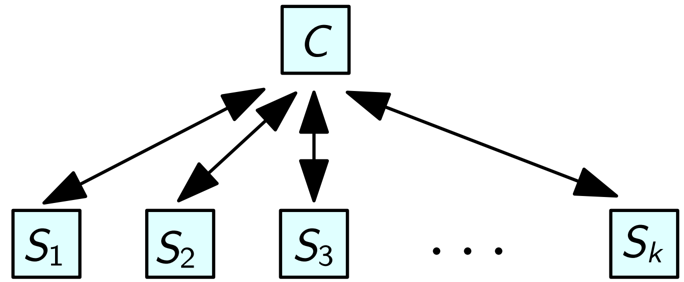

**Goal**: minimize communication & #rounds

**Problem**: how can we perform noise-resilient statistical estimation in the coordinator model comm. efficiently?

Assume all parties are provided with a pairwise distance metric and a threshold determining whether two items $u, v$ rep. the same entity (denoted by $u \sim v$) or not.

# Our goal and problem

**Goal**: minimize communication & #rounds




**Problem**: how can we perform noise-resilient statistical estimation in the coordinator model comm. efficiently?

Assume all parties are provided with a pairwise distance metric and a threshold determining whether two items $u, v$ rep. the same entity (denoted by $u \sim v$) or not.

The distance metric design is a separate issue.
We will design a framework so that users can plug-in any "distance metric" at run time.

# Remarks

**Remark** 1. We do not specify the distance function in our algorithms, for two reasons:

(1) Allows our algorithms to work with *any* distance functions.

(2) Sometimes it is very hard to assume that similarities between items can be expressed by a well-known distance function: "AT&T Corporation" is closer to "IBM Corporation" than "AT&T Corp" under the edit distance!

# Remarks

**Remark** 1. We do not specify the distance function in our algorithms, for two reasons:

(1) Allows our algorithms to work with *any* distance functions.

(2) Sometimes it is very hard to assume that similarities between items can be expressed by a well-known distance function: "AT&T Corporation" is closer to "IBM Corporation" than "AT&T Corp" under the edit distance!

**Remark** 2. We assume transitivity: if $u \sim v$, $v \sim w$ then $u \sim w$. In other words, the noise is "well-shaped".

One may come up with the following problematic situation: we have $a \sim b$, $b \sim c$, $\ldots$, $y \sim z$, however, $a \not\sim z$.

Our algorithm still work if the number of "outliers" is small.

# Remarks (cont.)

**Remark** 3. Do exist approaches w/o assuming transitivity. E.g., assume so-called ICAR properties [BGM+09], or use clustering based approaches [ACN08].

Unlikely to have comm.-efficient algorithms in our setting.

# Remarks (cont.)

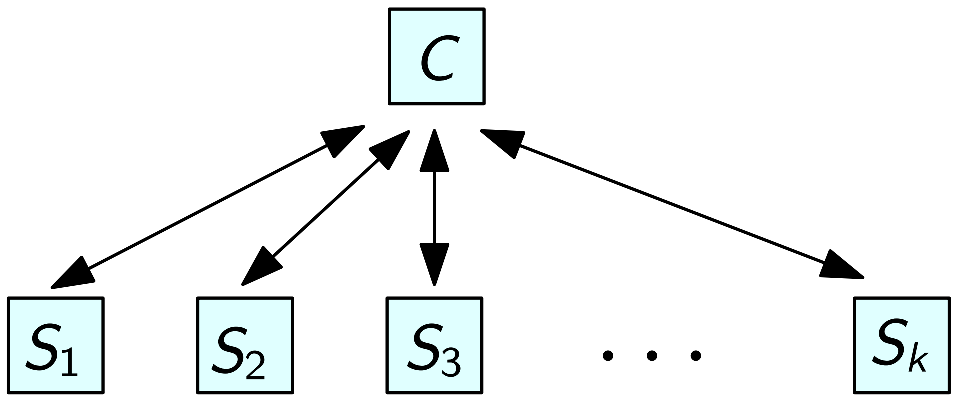
**Remark 3.** Do exist approaches w/o assuming transitivity. E.g., assume so-called ICAR properties [BGM+09], or use clustering based approaches [ACN08].

Unlikely to have comm.-efficient algorithms in our setting.

**Remark 4.** Whether there exists a magic hash function that can map (only) items in the same group into the same bucket **and** can be described succinctly?

Answer: NO

# A few notations




- We have $k$ sites (machines), each holding a multiset of items $S_i$.
- Let multiset $S = \bigcup_{i \in [k]} S_i$, let $m = |S|$.
- Under the transitivity assumption, $S$ can be partitioned into a set of groups $\mathcal{G} = \{G_1, \ldots, G_n\}$. Each group $G_i$ represents a distinct universe element.
- $\tilde{O}(\cdot)$ hides $\operatorname{poly}\log(m/\epsilon)$ factors.

# Our results

| | noisy data | | noise-free data |
|---|---|---|---|
| | (comm.) bits | rounds | bits |
| $F_0$ | $\tilde{O}(\min\{k/\epsilon^3, k^2/\epsilon^2\})$ | $\tilde{O}(1)$ | $\Omega(k/\epsilon^2)$ [WZ12,WZ14] |
| $L_0$-sampling | $\tilde{O}(k)$ | $\tilde{O}(1)$ | $\Omega(k)$ |
| $F_p$ $(p \geq 1)$ | $\tilde{O}((k^{p-1} + k^3)/\epsilon^3)$ | $O(1)$ | $\Omega(k^{p-1}/\epsilon^2)$ [WZ12] |
| $(\phi, \epsilon)$-HH | $\tilde{O}(\min\{k/\epsilon, 1/\epsilon^2\})$ | $O(1)$ | $\Omega(\min\{\frac{\sqrt{k}}{\epsilon}, \frac{1}{\epsilon^2}\})$ [HYZ12,WZ12] |
| Entropy | $\tilde{O}(k/\epsilon^2)$ | $O(1)$ | $\Omega(k/\epsilon^2)$ [WZ12] |

1. *p-th frequency moment* $F_p(S) = \sum_{i \in [n]} |G_i|^p$.

   We consider $F_0$ and $F_p$ $(p \geq 1)$, and allow a $(1+\epsilon)$-approximation.

2. *$L_0$-sampling* on $S$: return a group $G_i$ (or an arbitrary item in $G_i$) uniformly at random from $\mathcal{G}$.

3. *$(\phi, \epsilon)$-heavy-hitter* of $S$ $(0 < \epsilon \leq \phi \leq 1)$ (definition omitted)

4. *Empirical entropy*: $\text{Entropy}(S) = \sum_{i \in [n]} \frac{|G_i|}{m} \log \frac{m}{|G_i|}$.

   We allow a $(1+\epsilon)$-approximation.

# Take-home message:

In the distributed setting, we can handle well-shaped noise in statistical estimations almost for free!

# Rest of the talk: Algorithms for $F_0$

1. Simple Sampling

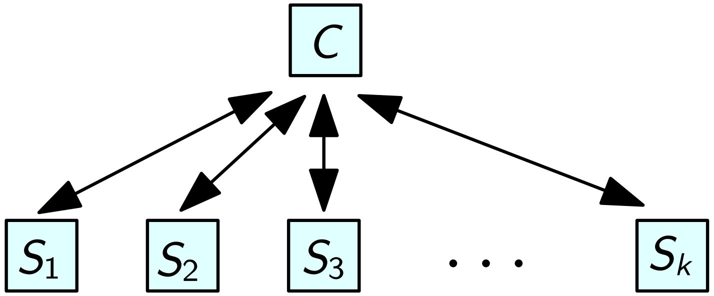
   Simple.
   $\tilde{O}(k^2/\epsilon^2)$ comm. 2 rounds.

2. Local Hierarchical Partition $+$
   Distributed Rejection Sampling

   Complicated.
   $\tilde{O}(k/\epsilon^3)$ comm. $\tilde{O}(1)$ rounds

Better than $\tilde{O}(k^2/\epsilon^2)$ bits because
(1) we want to scale on $k$
(2) later used in $\ell_0$-sampling with $\epsilon = \Theta(1)$

# Simple-Sampling




**Algorithm Simple-Sampling**

1. Let $m = |S| = \sum_{i \in [k]} |S_i|$.
2. For $j = 1, \ldots, \eta = \Theta(k/\epsilon^2)$
   (a) jointly sample a random item $u_j \in S$;
   Let $G_{u_j}$ be the group containing $u_j$.
   (b) jointly compute $|G_{u_j}|$, and set $X_j = 1/|G_{u_j}|$.
3. Output $\frac{m}{\eta} \sum_{j \in [k]} X_j$.

**Theorem**

Simple-Sampling gives a $(1+\epsilon)$ approximation of $F_0$ with probability $2/3$ using $\tilde{O}(k^2/\epsilon^2)$ bits and 2 rounds.

# Hierarchical partition + distributed rejection sampling

**Algorithm 3:** Estimating $F_0(\tilde{W}^\ell)$ for an $\ell \in \{0, 1, \ldots, L\}$

$cost \leftarrow 0, U \leftarrow \emptyset$;
$\eta_\tau \leftarrow c_\eta \cdot 16 \log^2 m/\epsilon^2 \cdot \log(200(L+1))$ /* set $\delta = 1/(200(L+1))$ */;
$t \leftarrow c_t \cdot k/\epsilon^3 \cdot \log^3 k \log^2 m \log N$ /* $c_t$ is a sufficiently large constant */;
**while** $(cost \leq t) \wedge (|U| < \eta_\tau)$ **do**
  the coordinator and sites generate a new sample (with replacement) $u \in W^\ell$;
  $s \leftarrow 0$ /* number of sites contacted */;
  $ct \leftarrow 0$ /* number of items in $G^\ell_{(u)}$ found in sampled sites */;
  **while** $s < k$ **do** /* test whether $u \in \tilde{W}^\ell$ */
    the coordinator samples (without replacement) a random site $I \in [k]$, and sends $u$ to site $I$ ;
    $s \leftarrow s + 1, cost \leftarrow cost + 1$ ;
    **if** $\exists v \in W^\ell_I$ *such that* $u \sim v$ **then**
      $ct \leftarrow ct + 1$;
      **if** $ct > \tau$ **then**
        mark $u$ bad;
        break /* do not satisfy item 3 in Definition 1 */;
    **if** $\exists v \in W^{\ell'}_I$ *such that* $\ell' > \ell$ *and* $u \sim v$ **then**
      mark $u$ bad;
      break /* do not satisfy item 2 in Definition 1 */;
  **if** $u$ *is not marked bad* **then**
    $U \leftarrow U \cup \{u\}$;
**if** $|U| \geq \eta_\tau$ **then**
  run Algorithm 2 on $U$ and output whatever Algorithm 2 outputs;
**else** output 0;

$\tilde{O}(k/\epsilon^3)$ bits

$\tilde{O}(1)$ rounds

# Hierarchical partition + distributed rejection sampling

**Main idea**: reduce the variance of $X_j$ in Simple-Sampling

– If we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_\ell = \{G \in \mathcal{G} \mid |G| \in (2^{\ell-1}, 2^\ell]\}$, and apply Algo Simple-Sampling on each class individually.

By doing this we can shave a factor of $k$ in the number of samples $X_j$ needed ( $\eta : k/\epsilon^2 \to 1/\epsilon^2$).

# Hierarchical partition + distributed rejection sampling

**Main idea**: reduce the variance of $X_j$ in Simple-Sampling

– If we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_\ell = \{G \in \mathcal{G} \mid |G| \in (2^{\ell-1}, 2^\ell]\}$, and apply Algo Simple-Sampling on each class individually.

By doing this we can shave a factor of $k$ in the number of samples $X_j$ needed ( $\eta : k/\epsilon^2 \to 1/\epsilon^2$).

– However, we cannot afford to partition the groups into classes in the distributed setting.

# Hierarchical partition + distributed rejection sampling

**Main idea**: reduce the variance of $X_j$ in Simple-Sampling

– If we can partition all groups in $\mathcal{G}$ into classes $\mathcal{G}_0, \ldots, \mathcal{G}_{\log k}$ such that $\mathcal{G}_\ell = \{G \in \mathcal{G} \mid |G| \in (2^{\ell-1}, 2^\ell]\}$, and apply Algo Simple-Sampling on each class individually.

By doing this we can shave a factor of $k$ in the number of samples $X_j$ needed ( $\eta : k/\epsilon^2 \rightarrow 1/\epsilon^2$).

– However, we cannot afford to partition the groups into classes in the distributed setting.

**Our techniques**:
local hierarchical partition (have inconsistency)
$+$ distributed rejection sampling (resolve the inconsistency)

Fairly complicated (use Algo Simple-Sampling as a subroutine).
See the paper for details.

# Other problems

1. $L_0$-**sampling**: $\tilde{O}(k)$ communication and $\tilde{O}(1)$ rounds.
   – Use Algorithm for $F_0$ as a subroutine
2. $p$-**th frequency moment**: $\tilde{O}((k^{p-1} + k^3)/\epsilon^3)$ comm. and $\tilde{O}(1)$ rounds.
   – Adapt a very algo by Kannan, Vempala and Woodruff. (COLT 2014)
3. $(\phi, \epsilon)$-**heavy-hitter**: $\tilde{O}(\min\{k/\epsilon, 1/\epsilon^2\})$ comm. and $O(1)$ rounds.
   – Easy
4. **Empirical entropy**: $\tilde{O}(k/\epsilon^2)$ comm. and $O(1)$ rounds.
   – Adapt an algo by Chakrabarti, Cormode and McGregor (SODA 2007) in data stream

# Open problems

- A number of bounds can possibly be improved. For example:
  – Can we get a (better) upper bound $\tilde{O}(k/\epsilon^2)$ for $F_0$?
  – Can we improve the round complexities of $F_0$ and $L_0$-sampling from $\tilde{O}(1)$ to $O(1)$?
  – Can we remove the $k^3$ factor in the communication cost for $F_p$?

# Open problems

- A number of bounds can possibly be improved. For example:
  – Can we get a (better) upper bound $\tilde{O}(k/\epsilon^2)$ for $F_0$?
  – Can we improve the round complexities of $F_0$ and $L_0$-sampling from $\tilde{O}(1)$ to $O(1)$?
  – Can we remove the $k^3$ factor in the communication cost for $F_p$?

- Can we obtain efficient algorithms for $L_2$-heavy-hitter and $L_p$-sampling?

# Open problems

- A number of bounds can possibly be improved. For example:
  – Can we get a (better) upper bound $\tilde{O}(k/\epsilon^2)$ for $F_0$?
  – Can we improve the round complexities of $F_0$ and $L_0$-sampling from $\tilde{O}(1)$ to $O(1)$?
  – Can we remove the $k^3$ factor in the communication cost for $F_p$?
- Can we obtain efficient algorithms for $L_2$-heavy-hitter and $L_p$-sampling?
- Lower bounds?

# Open problems

- A number of bounds can possibly be improved. For example:
  – Can we get a (better) upper bound $\tilde{O}(k/\epsilon^2)$ for $F_0$?
  – Can we improve the round complexities of $F_0$ and $L_0$-sampling from $\tilde{O}(1)$ to $O(1)$?
  – Can we remove the $k^3$ factor in the communication cost for $F_p$?
- Can we obtain efficient algorithms for $L_2$-heavy-hitter and $L_p$-sampling?
- Lower bounds?
- Relax/replace the transitivity assumption

Thank you!
Questions?