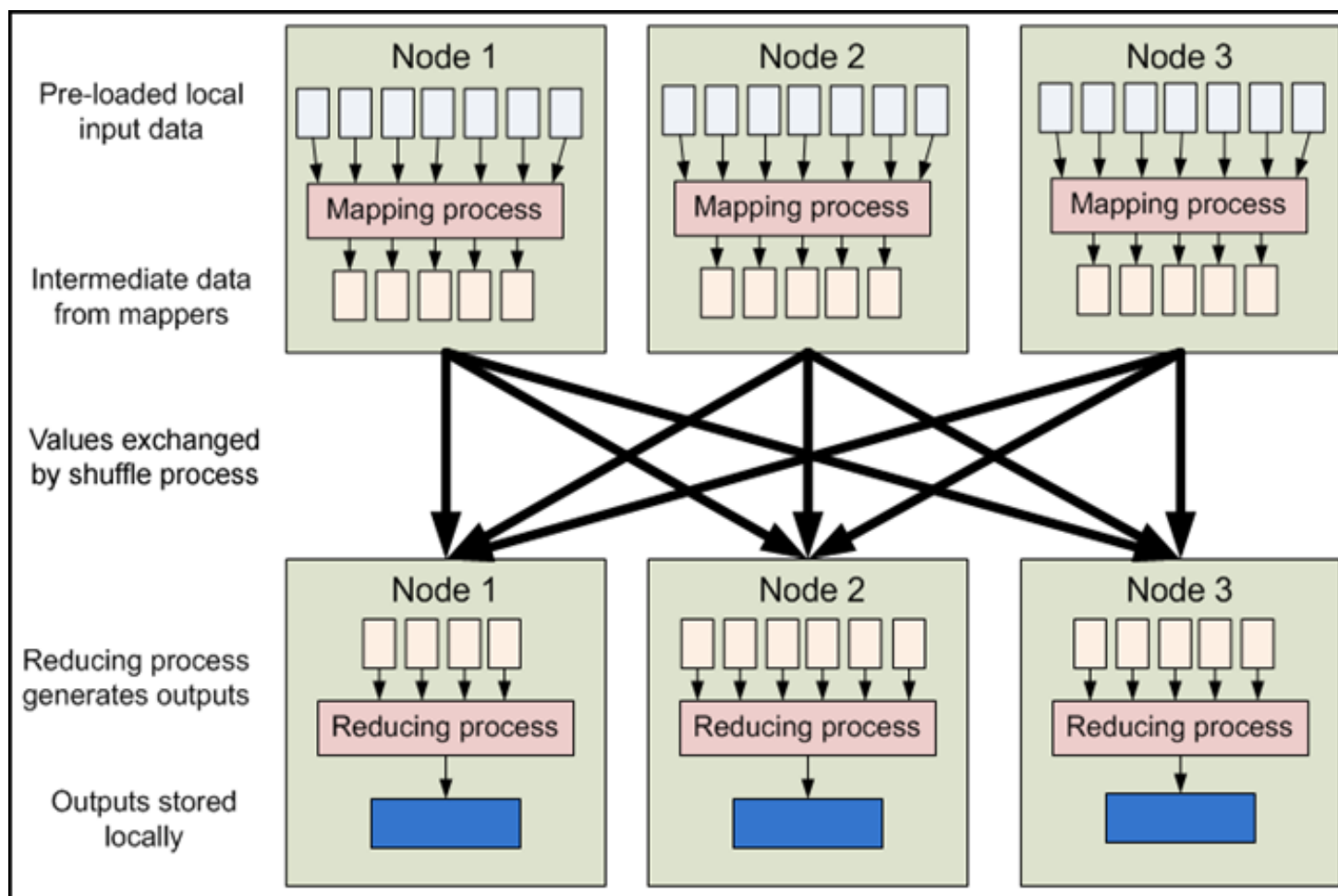


Computing Statistical Summaries over Massive Distributed Data

Ke Yi

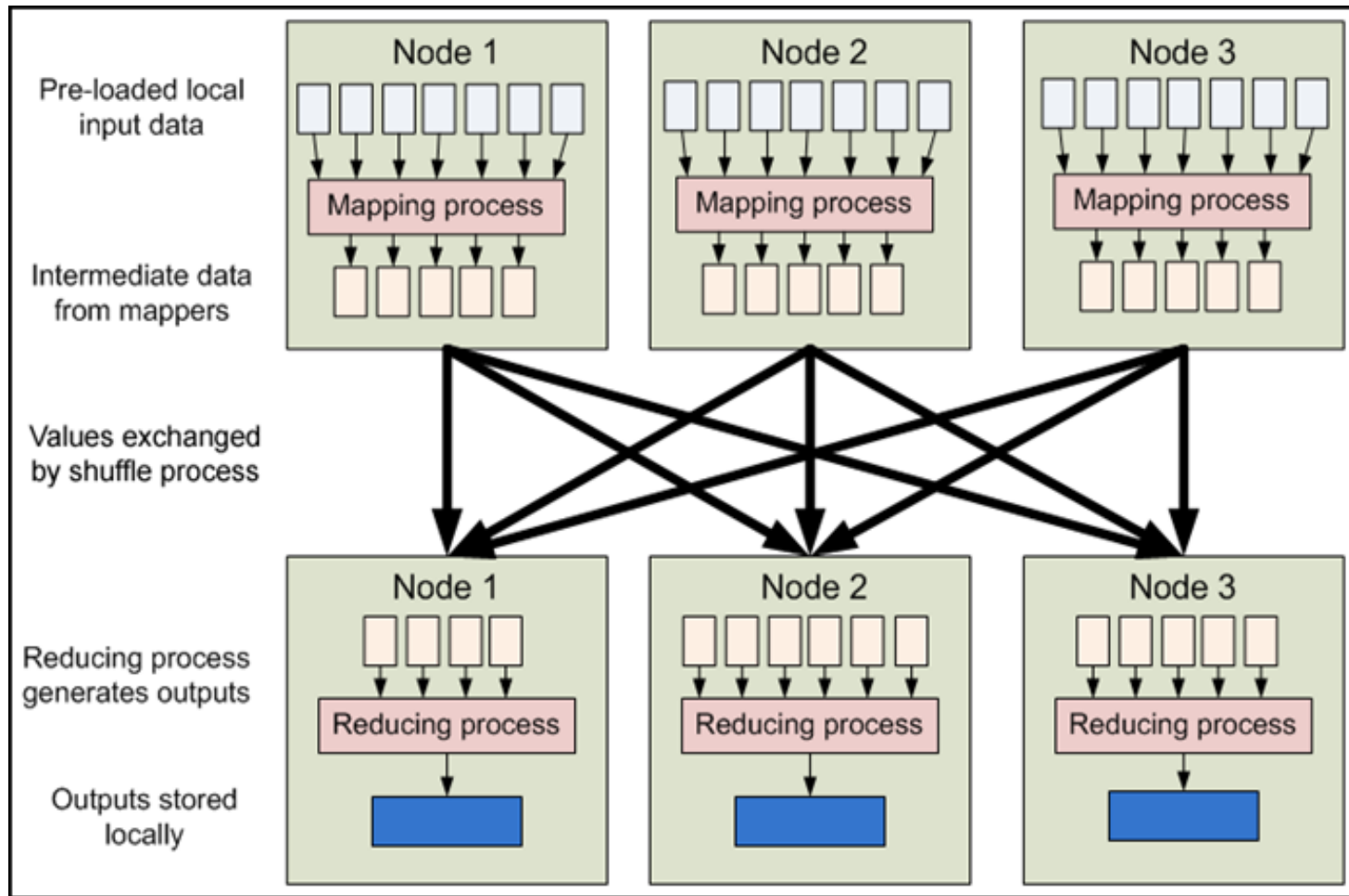
Hong Kong University of Science and Technology

Distributed Systems for Massive Data: MapReduce



Open source implementation: Hadoop

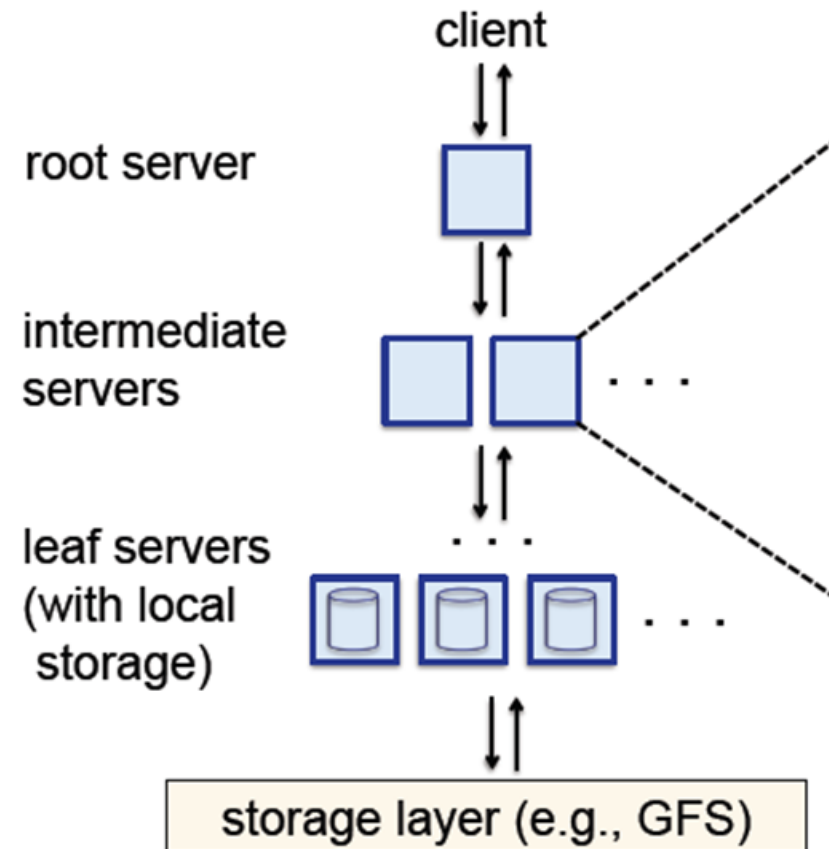
Distributed Systems for Massive Data: MapReduce



Open source implementation: Hadoop

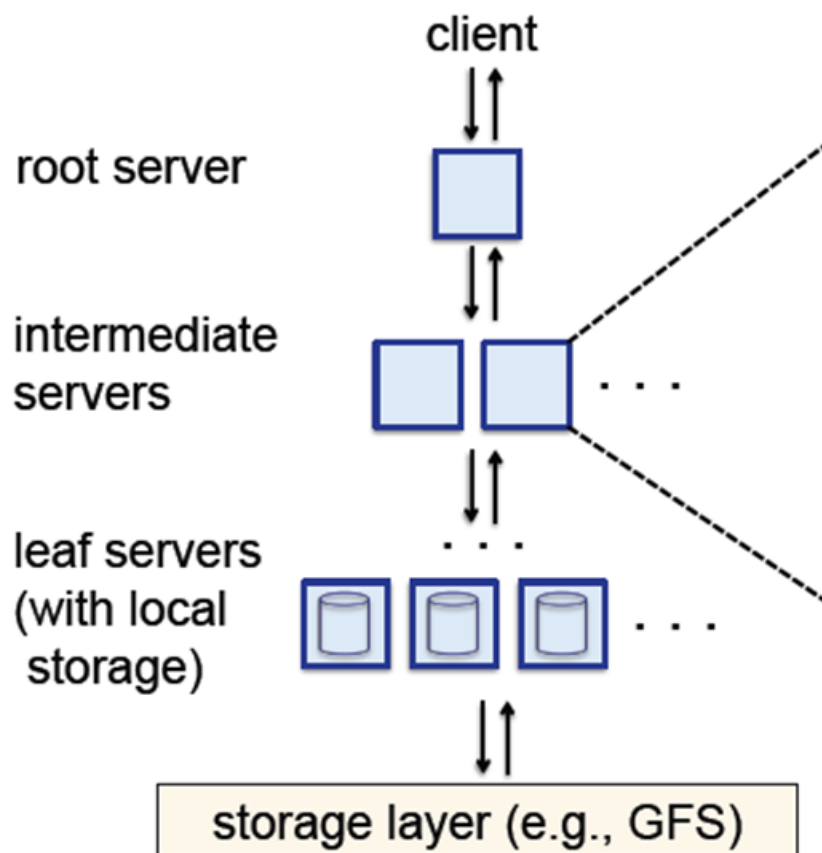
Suitable for batch processing (e.g., index construction)

Distributed Systems for Massive Data: Dremel



No open source implementation yet

Distributed Systems for Massive Data: Dremel

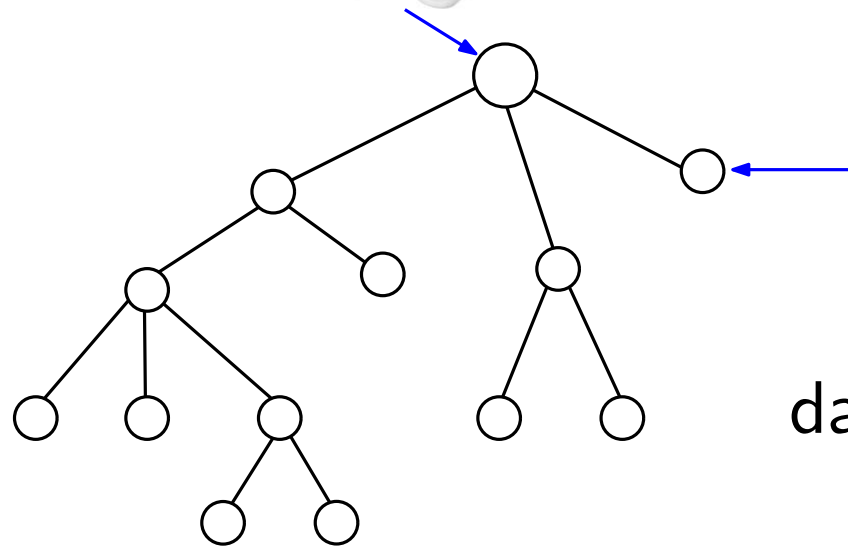


No open source implementation yet

Suitable for analytical queries (e.g., extracting a summary)

Distributed Systems for Massive Data: Sensor Net

base station

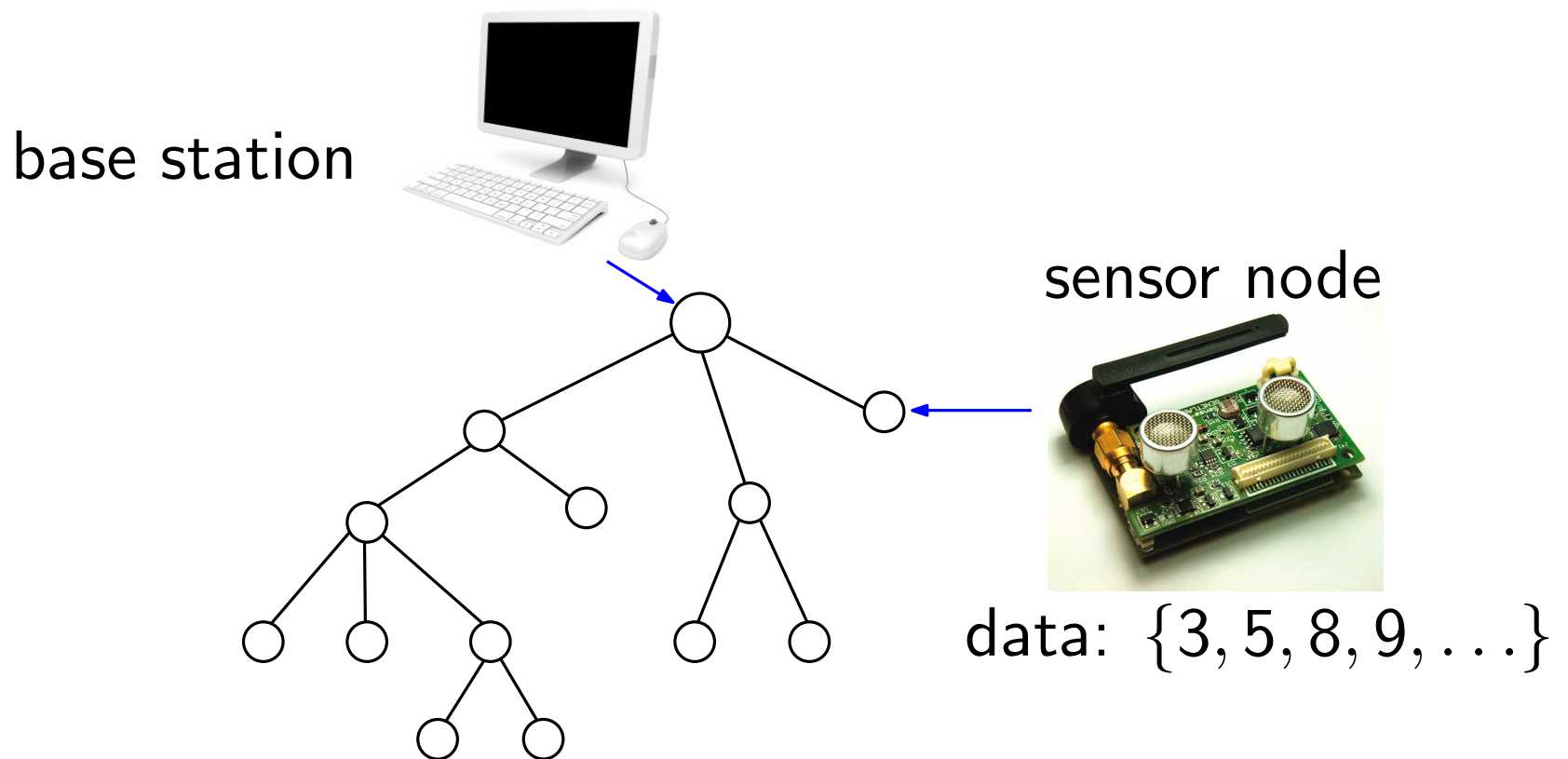


sensor node



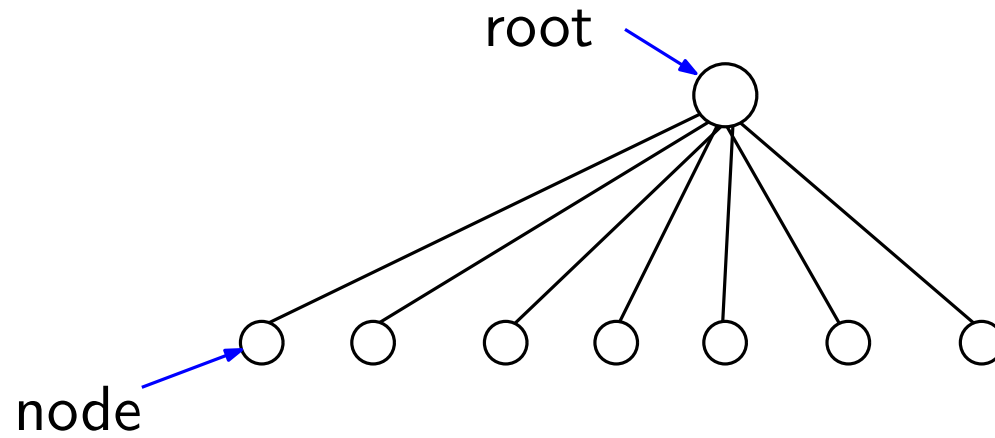
data: {3, 5, 8, 9, ...}

Distributed Systems for Massive Data: Sensor Net

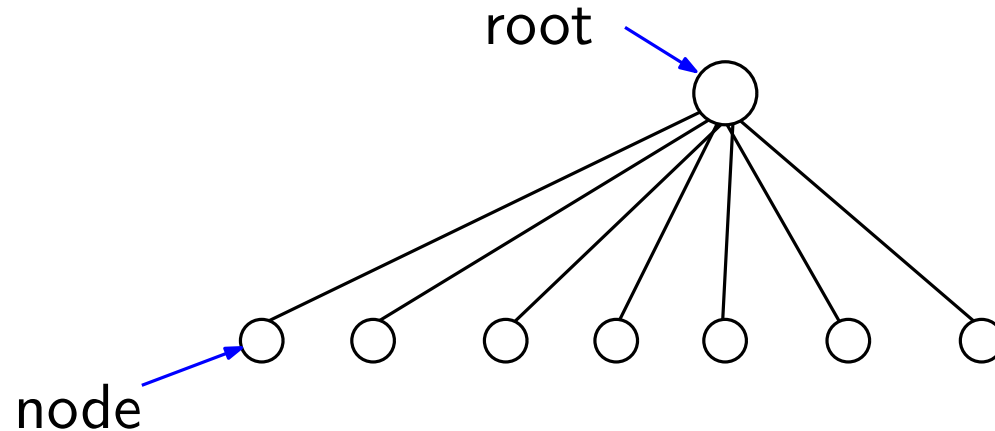


Data aggregation: Extracting a summary of the data

(Simplified) Model of Computation

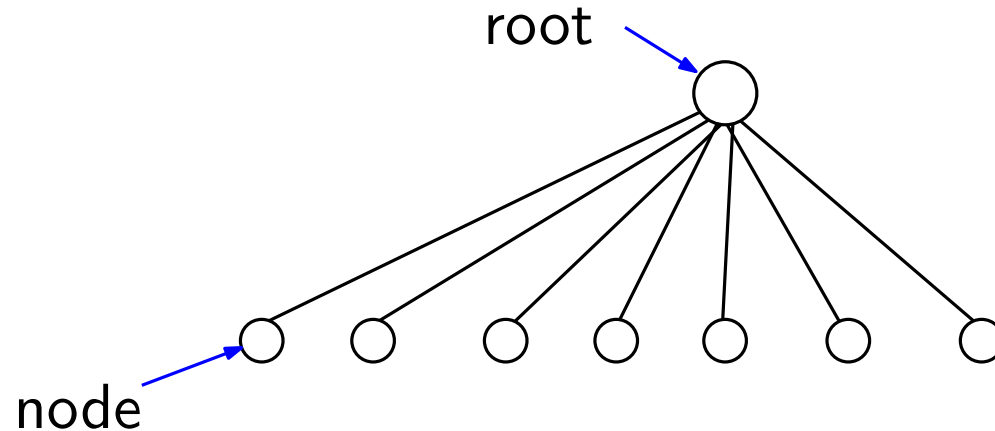


(Simplified) Model of Computation



- The root broadcasts a message to initialize computation
- Each node computes a summary on its local data
- The root combines the summaries to produce a global summary

(Simplified) Model of Computation

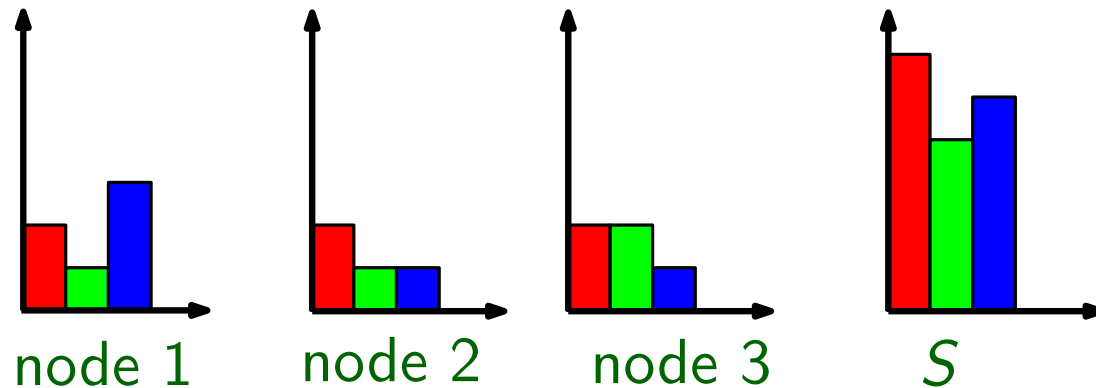


- The root broadcasts a message to initialize computation
- Each node computes a summary on its local data
- The root combines the summaries to produce a global summary
- Using minimum communication (and load balancing)

Outline

- Model of computation
- **Frequency estimation (heavy hitters)**
- Quantiles (order statistics)
- Other problems

Problem: Frequency Estimation



- Input: Multiset S of N items drawn from the universe $[u] = \{1 \dots u\}$
For example, all IP addresses
- Each node $j \in [k]$ holds a subset of S
For any item $i \in [u]$
 x_{ij} : total number of i 's in node j (local count)
 $y_i = \sum_{j=1}^n x_{ij}$ (global count)
- Compute y_i for each i

Frequency Estimation: Possible Solutions

- Compute exactly: **send everything**

Frequency Estimation: Possible Solutions

- Compute exactly: **send everything**
- Approximate each y_i within additive error ϵN

Frequency Estimation: Possible Solutions

- Compute exactly: **send everything**
- Approximate each y_i within additive error ϵN
- Sketching: Each node computes a sketch of its own data and sends it to the coordinator.

Count-min sketch, MG sketch, Space saving, etc.

Sketch size: $O(1/\epsilon)$

Communication cost: $O(k/\epsilon)$

Frequency Estimation: Possible Solutions

- Compute exactly: **send everything**
- Approximate each y_i within additive error ϵN
- Sketching: Each node computes a sketch of its own data and sends it to the coordinator.

Count-min sketch, MG sketch, Space saving, etc.

Sketch size: $O(1/\epsilon)$

Communication cost: $O(k/\epsilon)$

- Random sampling

Uniformly randomly sample a subset of size $O(1/\epsilon^2)$

Frequency Estimation: Possible Solutions

- Compute exactly: **send everything**
- Approximate each y_i within additive error ϵN
- Sketching: Each node computes a sketch of its own data and sends it to the coordinator.

Count-min sketch, MG sketch, Space saving, etc.

Sketch size: $O(1/\epsilon)$

Communication cost: $O(k/\epsilon)$

- Random sampling

Uniformly randomly sample a subset of size $O(1/\epsilon^2)$

- We can achieve: $O(\sqrt{k}/\epsilon)$

Typical values of $\epsilon = 10^{-3} \sim 10^{-6}$, $k = 10^2 \sim 10^4$

We assume $k < 1/\epsilon^2$

HT estimator [Horvitz and Thompson 56]

Each node holds a set of (item, count) pairs

(item, count)

(1, 20)
(2, 13)
(3, 35)
(4, 12)
(5, 5)
(6, 22)

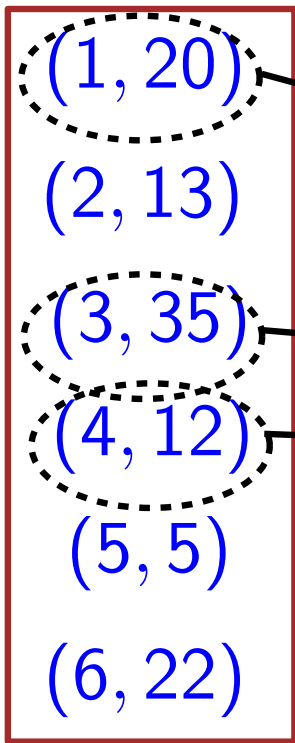
node j

send each pair (i, x_{ij}) with probability $g(x_{ij})$

HT estimator [Horvitz and Thompson 56]

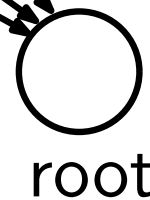
Each node holds a set of (item, count) pairs

(item, count)



node j

send each pair (i, x_{ij}) with probability $g(x_{ij})$



HT estimator [Horvitz and Thompson 56]

HT estimator for x_{ij} :

$$Y_{i,j} = \frac{x_{i,j}}{g(x_{i,j})} \text{ if it is sampled, otherwise } 0$$

This is an unbiased estimator

Estimator for y_i :

$$Y_i = Y_{i,1} + \cdots + Y_{i,n}$$

HT estimator [Horvitz and Thompson 56]

HT estimator for x_{ij} :

$$Y_{i,j} = \frac{x_{i,j}}{g(x_{i,j})} \text{ if it is sampled, otherwise } 0$$

This is an unbiased estimator

Estimator for y_i :

$$Y_i = Y_{i,1} + \cdots + Y_{i,n}$$

$$\begin{aligned} \text{Var}[Y_{i,j}] &= \left(\frac{x_{i,j}}{g(x_{i,j})} - x_{i,j} \right)^2 g(x_{i,j}) + (x_{i,j})^2 (1 - g(x_{i,j})) \\ &= \frac{x_{i,j}^2 (1 - g(x_{i,j}))}{g(x_{i,j})} \end{aligned}$$

$$\text{Var}[Y_i] = \sum_{j=1}^n \text{Var}[Y_{ij}] = \sum_{j=1}^n \frac{x_{i,j}^2 (1 - g(x_{i,j}))}{g(x_{i,j})}$$

Sampling Function

Question: What sampling function $g(x)$ should we use

Sampling Function

Question: What sampling function $g(x)$ should we use

Accuracy: standard deviation less than ϵN

A function is **valid**, if $\text{Var}[Y_i] \leq (\epsilon N)^2$ for all items i

Sampling Function

Question: What sampling function $g(x)$ should we use

Accuracy: standard deviation less than ϵN

A function is **valid**, if $\text{Var}[Y_i] \leq (\epsilon N)^2$ for all items i

Communication cost: $\sum_{i,j} g(x_{ij})$

Sampling Function

Question: What sampling function $g(x)$ should we use

Accuracy: standard deviation less than ϵN

A function is **valid**, if $\text{Var}[Y_i] \leq (\epsilon N)^2$ for all items i

Communication cost: $\sum_{i,j} g(x_{ij})$

Optimal valid $g(x)$?

A Worst-Case Optimal Sampling Function

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

A Worst-Case Optimal Sampling Function

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

Can show:

$$\blacksquare \quad \text{Var}[Y_i] = - \left(\frac{y_i}{\sqrt{k}} - \frac{\varepsilon N}{2} \right)^2 + \frac{(\varepsilon N)^2}{4} \leq \frac{1}{4}(\varepsilon N)^2,$$

i.e., $g_1(x)$ is valid

- Communication cost of using $g_1(x)$ is $O(\sqrt{k}/\varepsilon)$
- Communication cost of any valid sampling function is $\Omega(\sqrt{k}/\varepsilon)$ in the worst case (i.e., on some input)

A Worst-Case Optimal Sampling Function

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

Can show:

$$\blacksquare \quad \text{Var}[Y_i] = - \left(\frac{y_i}{\sqrt{k}} - \frac{\varepsilon N}{2} \right)^2 + \frac{(\varepsilon N)^2}{4} \leq \frac{1}{4}(\varepsilon N)^2,$$

i.e., $g_1(x)$ is valid

- Communication cost of using $g_1(x)$ is $O(\sqrt{k}/\varepsilon)$
- Communication cost of any valid sampling function is $\Omega(\sqrt{k}/\varepsilon)$ in the worst case (i.e., on some input)
- A very recent result shows that **any** algorithm has to spend $\Omega(\sqrt{k}/\varepsilon)$ communication in the worst case [Verbin, Woodruff, Zhang, manuscript]

Another Sampling Function

$$g_2(x) = (g_1(x))^2$$

Another Sampling Function

$$g_2(x) = (g_1(x))^2$$

Can show:

- g_2 is also valid

Another Sampling Function

$$g_2(x) = (g_1(x))^2$$

Can show:

- g_2 is also valid
- Clearly, $g_1(x) \geq g_2(x)$

g_1 's communication cost is always $\Theta(\sqrt{k}/\varepsilon)$, while g_2 can be much better when there are many small local counts

Another Sampling Function

$$g_2(x) = (g_1(x))^2$$

Can show:

- g_2 is also valid
- Clearly, $g_1(x) \geq g_2(x)$

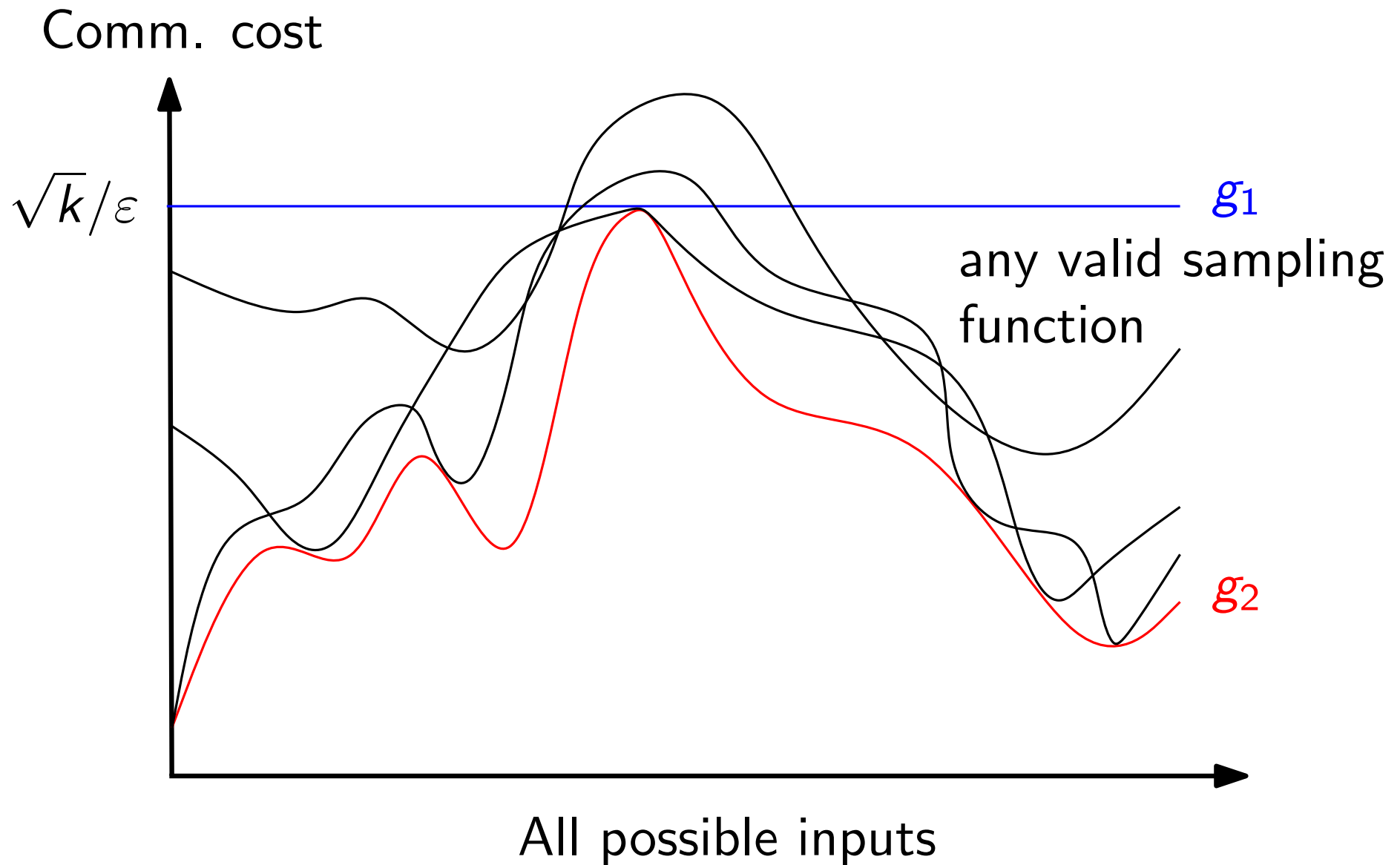
g_1 's communication cost is always $\Theta(\sqrt{k}/\varepsilon)$, while g_2 can be much better when there are many small local counts

- A stronger optimality: $g_2(x)$ is **instance-optimal**

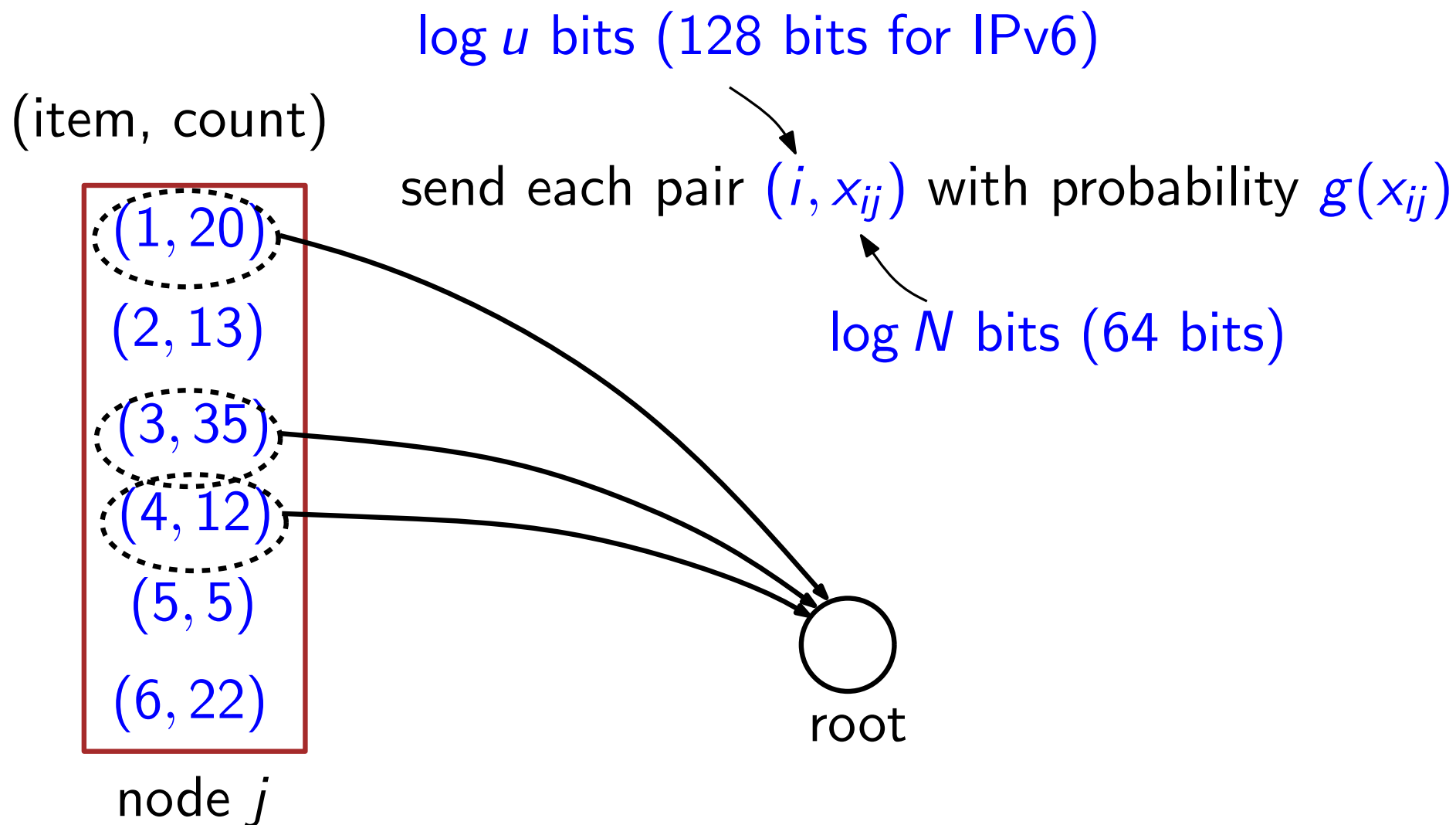
Define $opt(I) = \sum_{i,j} g_2(x_{i,j})$ on input $I : \{x_{i,j}\}$

Can show that on every input I , any valid sampling function must have cost $\Omega(opt(I))$

Instance Optimality



Further Reducing Communication Cost



Further Reducing Communication Cost

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

HT estimator for x_{ij} :

$$Y_{i,j} = \frac{x_{i,j}}{g(x_{i,j})} \text{ if it is sampled, otherwise } 0$$

Estimator for y_i :

$$Y_i = Y_{i,1} + \cdots + Y_{i,n}$$

Further Reducing Communication Cost

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

HT estimator for x_{ij} :

$$Y_{i,j} = \frac{x_{i,j}}{g(x_{i,j})} \text{ if it is sampled, otherwise } 0$$

Estimator for y_i :

$$Y_i = Y_{i,1} + \cdots + Y_{i,n}$$

$$Y_i = \frac{\varepsilon N}{\sqrt{k}}(1 + 0 + 1 + 1 + \cdots + 0 + 1)$$

Further Reducing Communication Cost

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

HT estimator for x_{ij} :

$$Y_{i,j} = \frac{x_{i,j}}{g(x_{i,j})} \text{ if it is sampled, otherwise } 0$$

Estimator for y_i :

$$Y_i = Y_{i,1} + \cdots + Y_{i,n}$$

$$Y_i = \frac{\varepsilon N}{\sqrt{k}}(1 + 0 + 1 + 1 + \cdots + 0 + 1)$$

Each site j just needs to tell whether i is sampled or not!

Further Reducing Communication Cost

$$g_1(x) = \min\left\{\frac{\sqrt{k}}{\varepsilon N}x, 1\right\}$$

HT estimator for x_{ij} :

$$Y_{i,j} = \frac{x_{i,j}}{g(x_{i,j})} \text{ if it is sampled, otherwise } 0$$

Estimator for y_i :

$$Y_i = Y_{i,1} + \cdots + Y_{i,n}$$

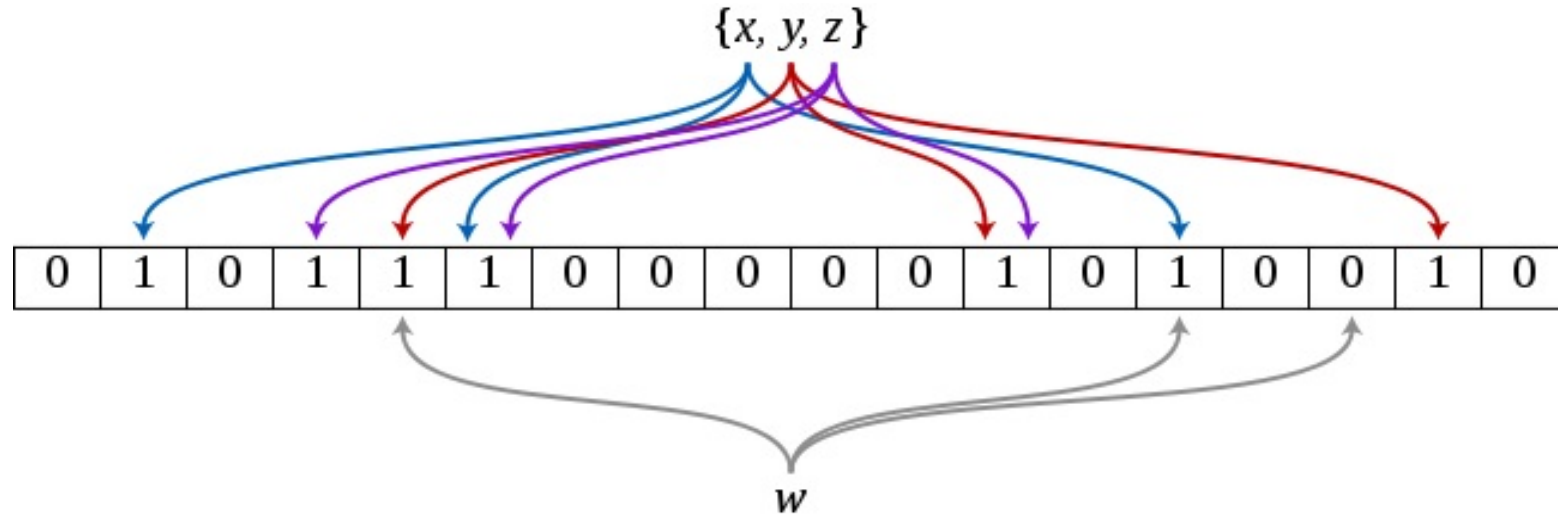
$$Y_i = \frac{\varepsilon N}{\sqrt{k}}(1 + 0 + 1 + 1 + \cdots + 0 + 1)$$

Each site j just needs to tell whether i is sampled or not!

The set of sampled items can be encoded in a Bloom filter, taking $O(1)$ bits per item

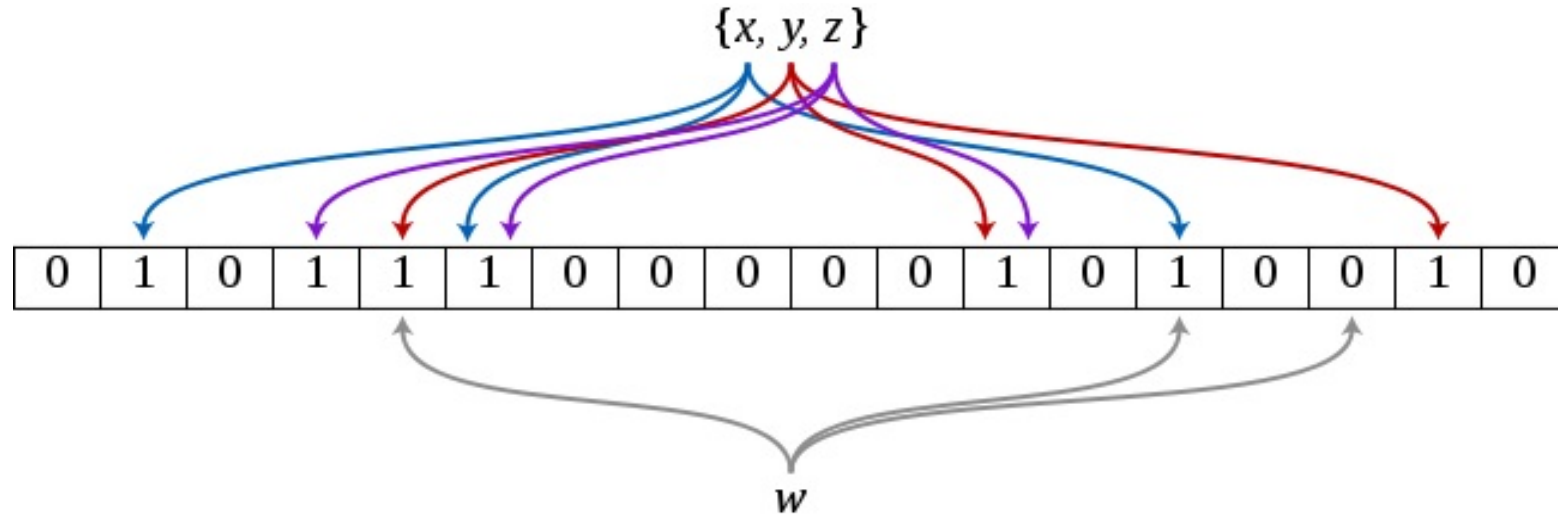
\Rightarrow total cost = $O(\sqrt{k}/\varepsilon)$ **bits**

Bloom Filters



- A Bloom filter needs $O(\log(1/q))$ bits per item
- No false negatives
- False positive probability = q

Bloom Filters



- A Bloom filter needs $O(\log(1/q))$ bits per item
- No false negatives
- False positive probability = q

Change the estimator to

$$Y_i = \frac{\epsilon N}{\sqrt{k}} \cdot \frac{Y_{i,1} + \dots + Y_{i,k} - kq}{1 - q}$$

Sampling with $g_2(x) = (g_1(x))^2$

- $g_2(x)$ samples $opt(I)$ (item, count) pairs, which may be much smaller than $O(\sqrt{k}/\varepsilon)$ on many inputs
- But it is a nonlinear sampling function

Estimator for y_i :

$$Y_i = \frac{x_{i,1}}{g_2(x_{i,1})} + 0 + 0 + \frac{x_{i,4}}{g_2(x_{i,4})} + \dots + 0 + \frac{x_{i,k}}{g_2(x_{i,k})}$$

Sampling with $g_2(x) = (g_1(x))^2$

- $g_2(x)$ samples $opt(I)$ (item, count) pairs, which may be much smaller than $O(\sqrt{k}/\varepsilon)$ on many inputs
- But it is a nonlinear sampling function

Estimator for y_i :

$opt(I)$ such terms

$$Y_i = \frac{x_{i,1}}{g_2(x_{i,1})} + 0 + 0 + \frac{x_{i,4}}{g_2(x_{i,4})} + \dots + 0 + \frac{x_{i,k}}{g_2(x_{i,k})}$$

Sampling with $g_2(x) = (g_1(x))^2$

- $g_2(x)$ samples $opt(I)$ (item, count) pairs, which may be much smaller than $O(\sqrt{k}/\epsilon)$ on many inputs
- But it is a nonlinear sampling function

Estimator for y_i :

$opt(I)$ such terms

$$Y_i = \frac{x_{i,1}}{g_2(x_{i,1})} + 0 + 0 + \frac{x_{i,4}}{g_2(x_{i,4})} + \dots + 0 + \frac{x_{i,k}}{g_2(x_{i,k})}$$

- Use $g_2(x)$ to perform the sampling locally
- Then use $g_1(x)$ + Bloom filters to sample the $\frac{x_{i,j}}{g_2(x_{i,j})}$'s

Sampling with $g_2(x) = (g_1(x))^2$

- $g_2(x)$ samples $opt(I)$ (item, count) pairs, which may be much smaller than $O(\sqrt{k}/\varepsilon)$ on many inputs
- But it is a nonlinear sampling function

Estimator for y_i :

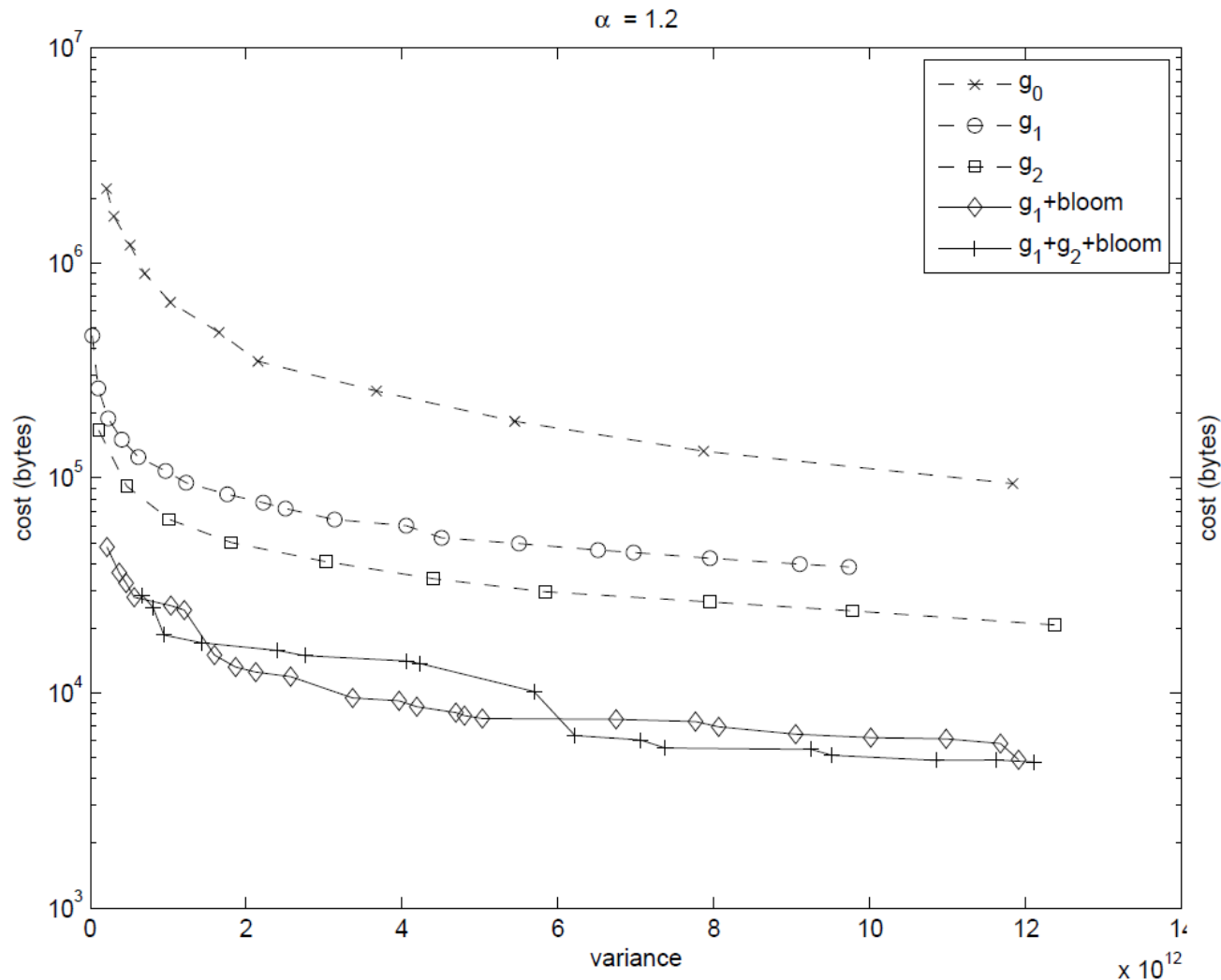
$opt(I)$ such terms

$$Y_i = \frac{x_{i,1}}{g_2(x_{i,1})} + 0 + 0 + \frac{x_{i,4}}{g_2(x_{i,4})} + \dots + 0 + \frac{x_{i,k}}{g_2(x_{i,k})}$$

- Use $g_2(x)$ to perform the sampling locally
- Then use $g_1(x)$ + Bloom filters to sample the $\frac{x_{i,j}}{g_2(x_{i,j})}$'s

Can show this takes $O\left(opt(I) \log^2\left(\frac{\sqrt{k}}{\varepsilon opt(I)}\right)\right)$ bits

Simulation Results



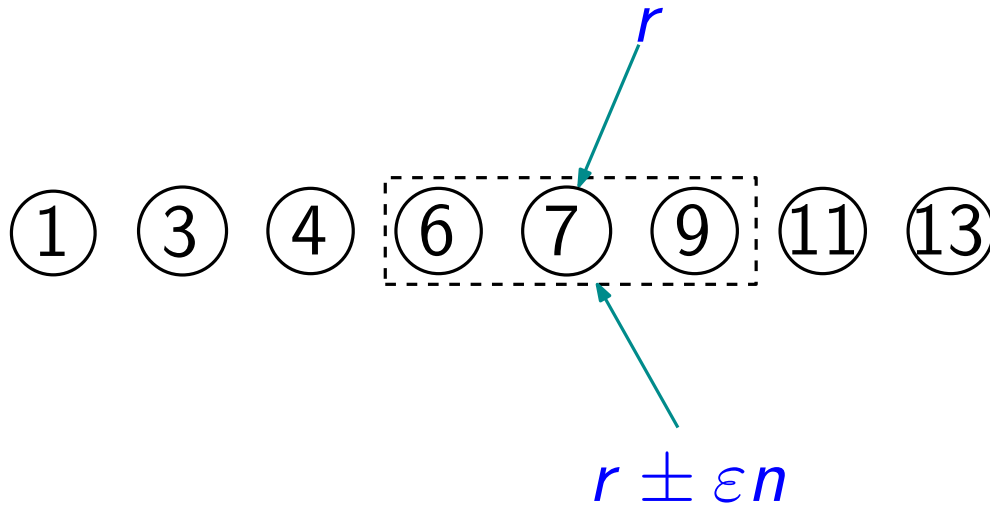
$k = 1000$, $N = 10^9$ following Zipf distribution with $\alpha = 1.2$.
Estimate the frequencies of the 100 most popular items. Variance
computed from 100 runs, and take the worst

Outline

- Model of computation
- Frequency estimation (heavy hitters)
- **Quantiles (order statistics)**
- Other problems

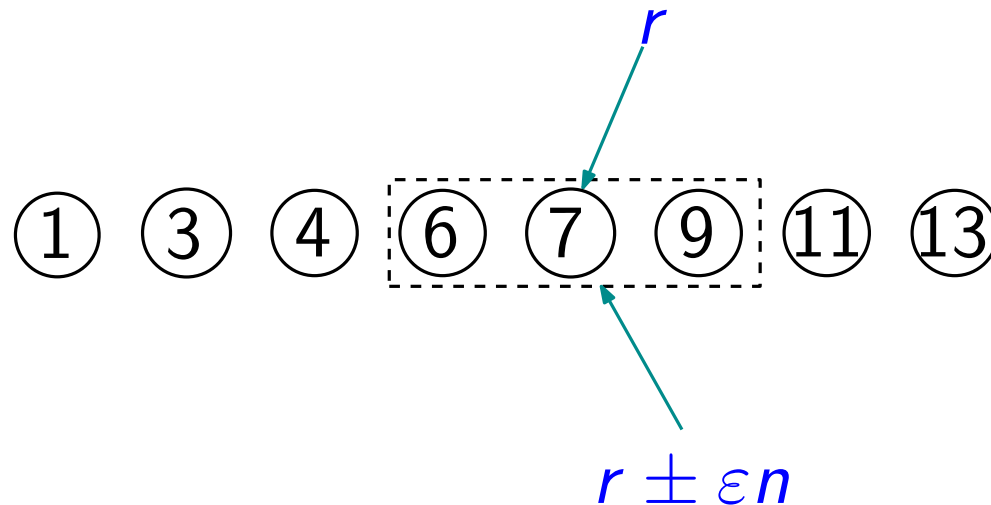
Quantiles

In a set of n values, the (r/n) -quantile is the value ranked at r .
The 0.5-quantile is the **median**.



Quantiles

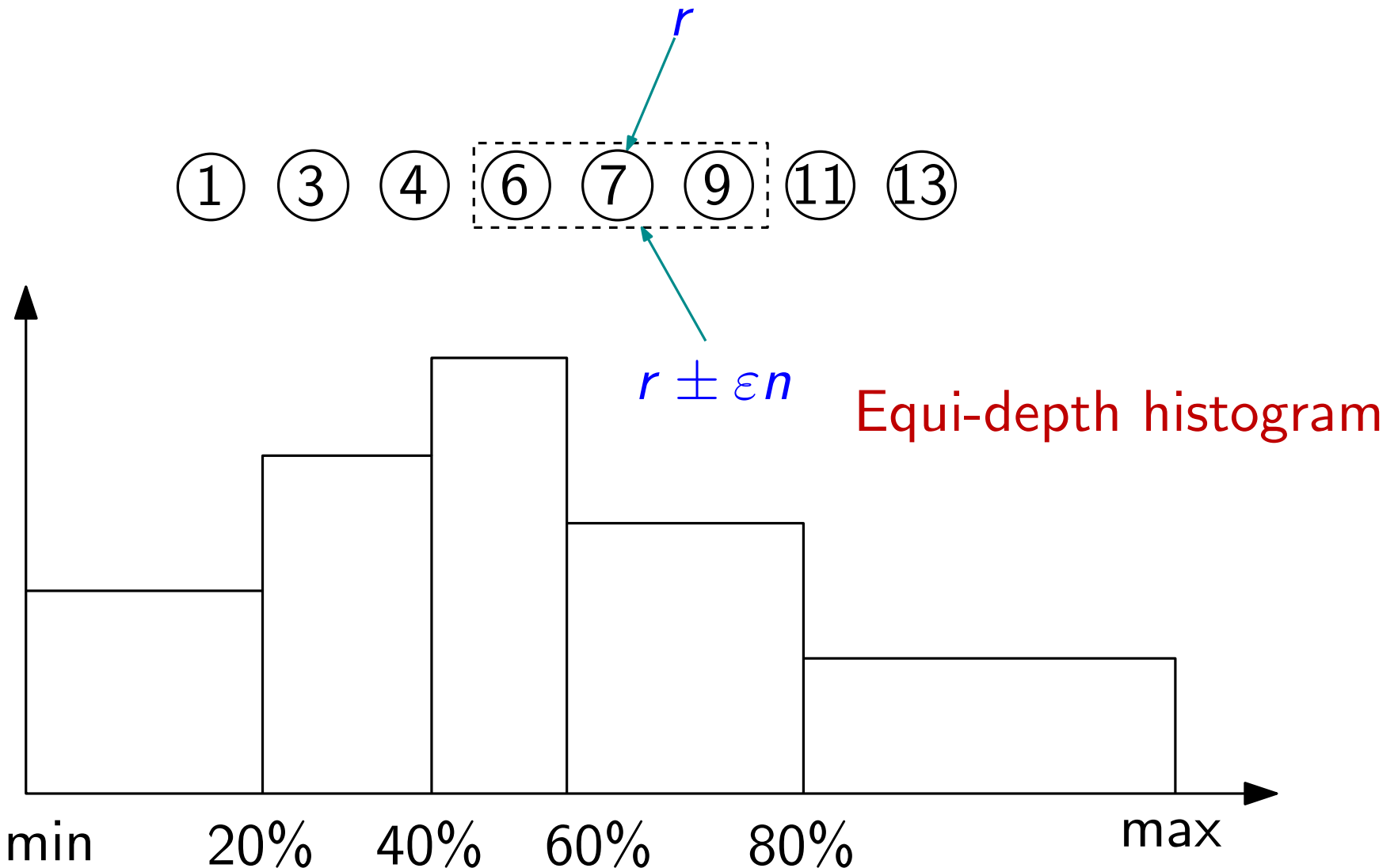
In a set of n values, the (r/n) -quantile is the value ranked at r .
The 0.5-quantile is the **median**.



An ϵ -approximate (r/n) -quantile is any value ranked between $[r - \epsilon n, r + \epsilon n]$.

Quantiles

In a set of n values, the (r/n) -quantile is the value ranked at r .
The 0.5-quantile is the **median**.



Quantile Summaries

Previous solution: Each site computes a **quantile summary** on its local data and sends it to the root, and the root combines them

	summary size	
q-digest [Shrivastava et al. '04] 272 citations	$O(\frac{1}{\epsilon} \log u)$	
GK [Greenwald, Khanna '04] 102 citations	$O(\frac{1}{\epsilon} \log^2 n)$	

n : total data size.

ϵ : error. $10^{-3} - 10^{-6}$

k : number of nodes. $100 \sim 10000$

u : size of universe. $\log u = 32$

Quantile Summaries

Previous solution: Each site computes a **quantile summary** on its local data and sends it to the root, and the root combines them

	summary size	total cost
q-digest [Shrivastava et al. '04] 272 citations	$O(\frac{1}{\epsilon} \log u)$	$O(\frac{k}{\epsilon} \log u)$
GK [Greenwald, Khanna '04] 102 citations	$O(\frac{1}{\epsilon} \log^2 n)$	$O(\frac{k}{\epsilon} \log^2 n)$

n : total data size.

ϵ : error. $10^{-3} - 10^{-6}$

k : number of nodes. $100 \sim 10000$

u : size of universe. $\log u = 32$

Quantile Summaries

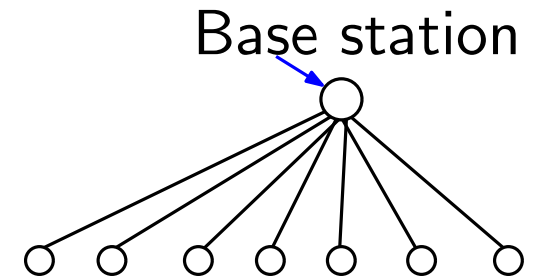
Previous solution: Each site computes a **quantile summary** on its local data and sends it to the root, and the root combines them

	summary size	total cost
q-digest [Shrivastava et al. '04] 272 citations	$O(\frac{1}{\epsilon} \log u)$	$O(\frac{k}{\epsilon} \log u)$
GK [Greenwald, Khanna '04] 102 citations	$O(\frac{1}{\epsilon} \log^2 n)$	$O(\frac{k}{\epsilon} \log^2 n)$
New	$O(\frac{1}{\epsilon} \log k)$	$O(\frac{\sqrt{k}}{\epsilon})$

n : total data size.
 ϵ : error. $10^{-3} - 10^{-6}$

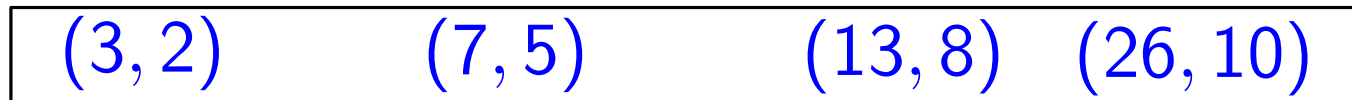
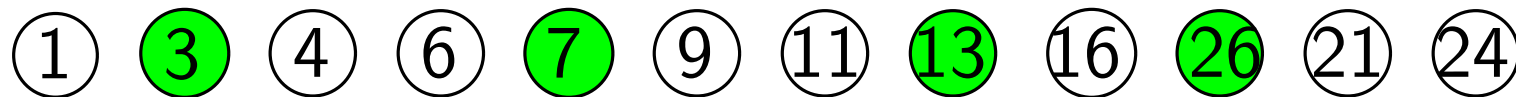
k : number of nodes. $100 \sim 10000$
 u : size of universe. $\log u = 32$

The Algorithm



The algorithm for each node

Sample each value with probability p



Compute local ranks

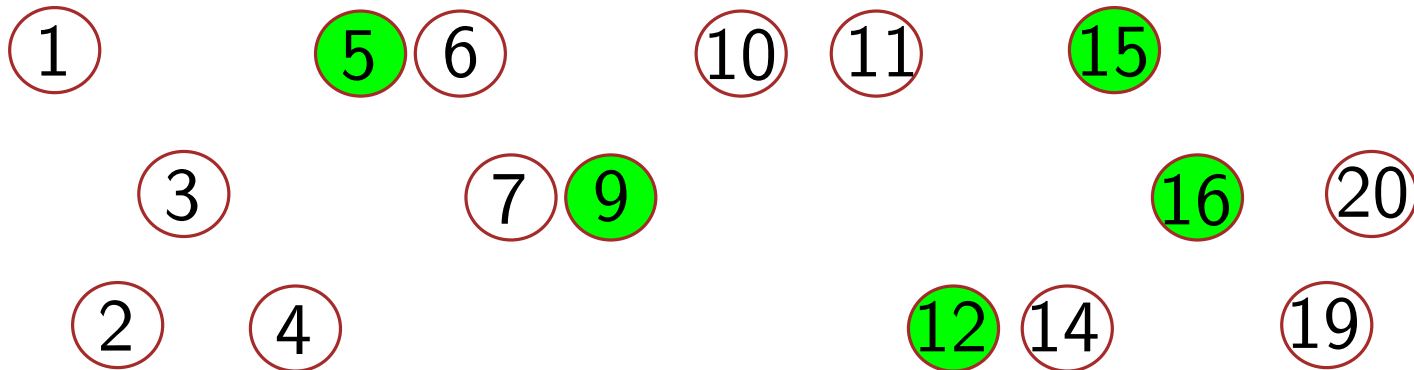
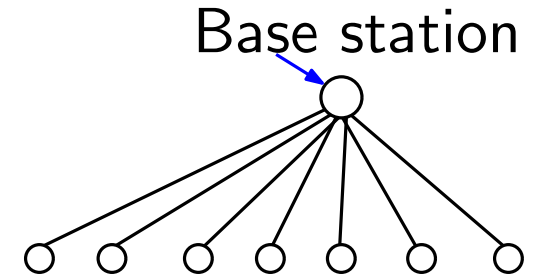


The Algorithm

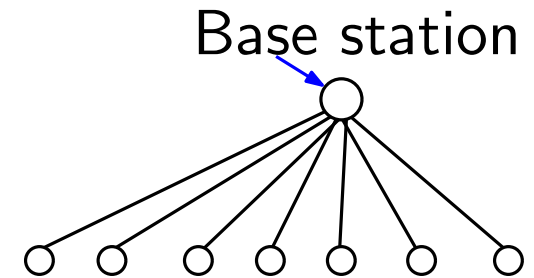
At the base station:

Answering **value-to-rank** query

Given any value x , estimates its rank $r(x)$



The Algorithm

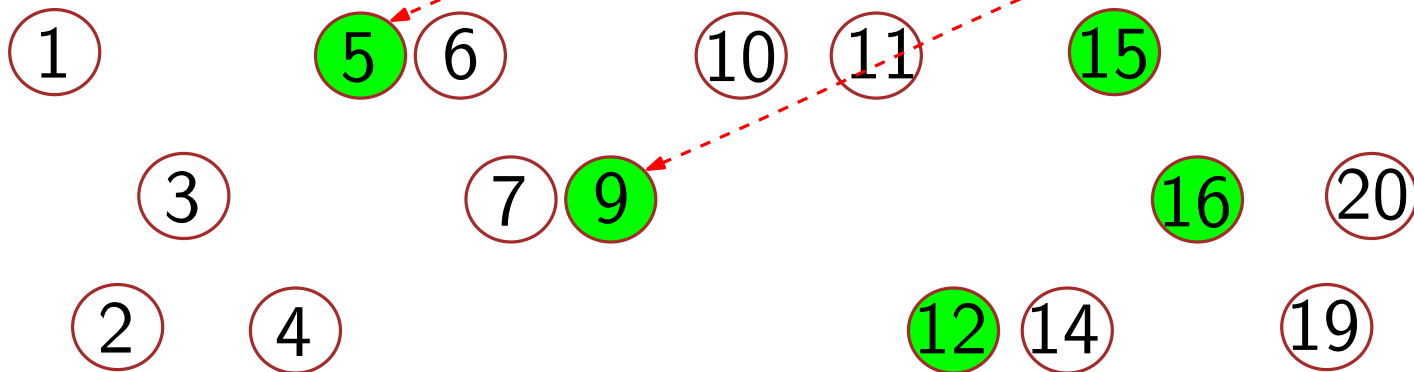


At the base station:

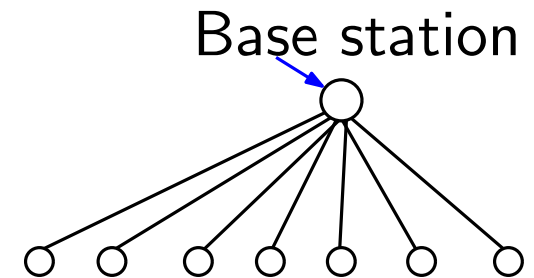
Answering **value-to-rank** query

Given any value x , estimates its rank $r(x)$

predecessor $r(10)?$



The Algorithm



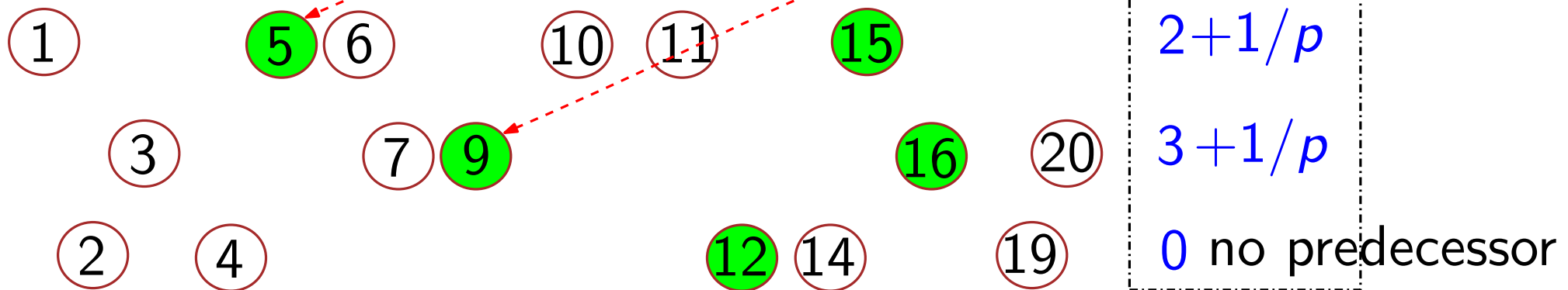
At the base station:

Answering **value-to-rank** query

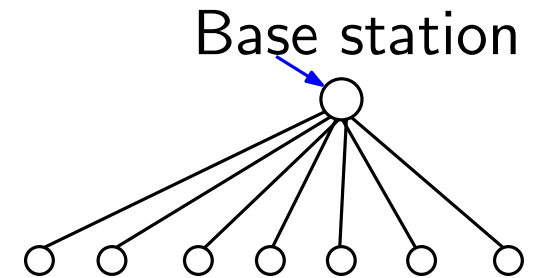
Given any value x , estimates its rank $r(x)$

predecessor

$r(10)?$



The Algorithm



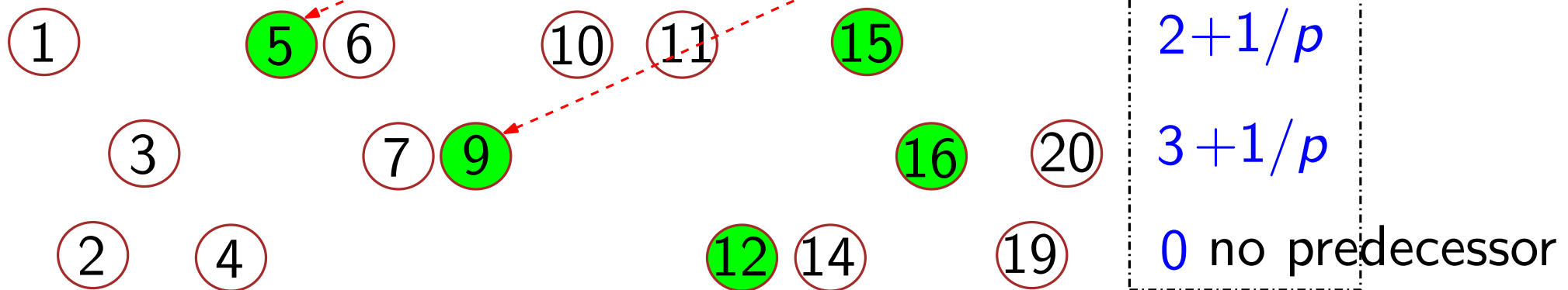
At the base station:

Answering **value-to-rank** query

Given any value x , estimates its rank $r(x)$

predecessor

$r(10)?$



$$\hat{r}(10) = 5 + 2/p$$

Correctness

Will show: $\hat{r}(x)$ is an unbiased estimator of $r(x)$ with standard deviation εn .

$r(10)?$

1

5

6

10

11

15

Correctness

Will show: $\hat{r}(x)$ is an unbiased estimator of $r(x)$ with standard deviation εn .

$r(10)$?

5

15

Correctness

Will show: $\hat{r}(x)$ is an unbiased estimator of $r(x)$ with standard deviation εn .

$r(10)?$

5

?

15

Correctness

Will show: $\hat{r}(x)$ is an unbiased estimator of $r(x)$ with standard deviation εn .

$r(10)?$

5

?

15



Follows a geometric distribution (almost)

$$E[?] = 1/p$$

$$\text{Var}[?] \leq 1/p^2$$

Correctness

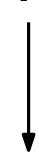
Will show: $\hat{r}(x)$ is an unbiased estimator of $r(x)$ with standard deviation εn .

$r(10)?$

5

?

15



Follows a geometric distribution (almost)

$$E[?] = 1/p$$

$$\text{Var}[?] \leq 1/p^2$$

$$\text{Set } p = \frac{\sqrt{k}}{\varepsilon n}$$

$$\text{Var}[\hat{r}(x)] \leq k/p^2 = (\varepsilon n)^2$$

Communication Cost

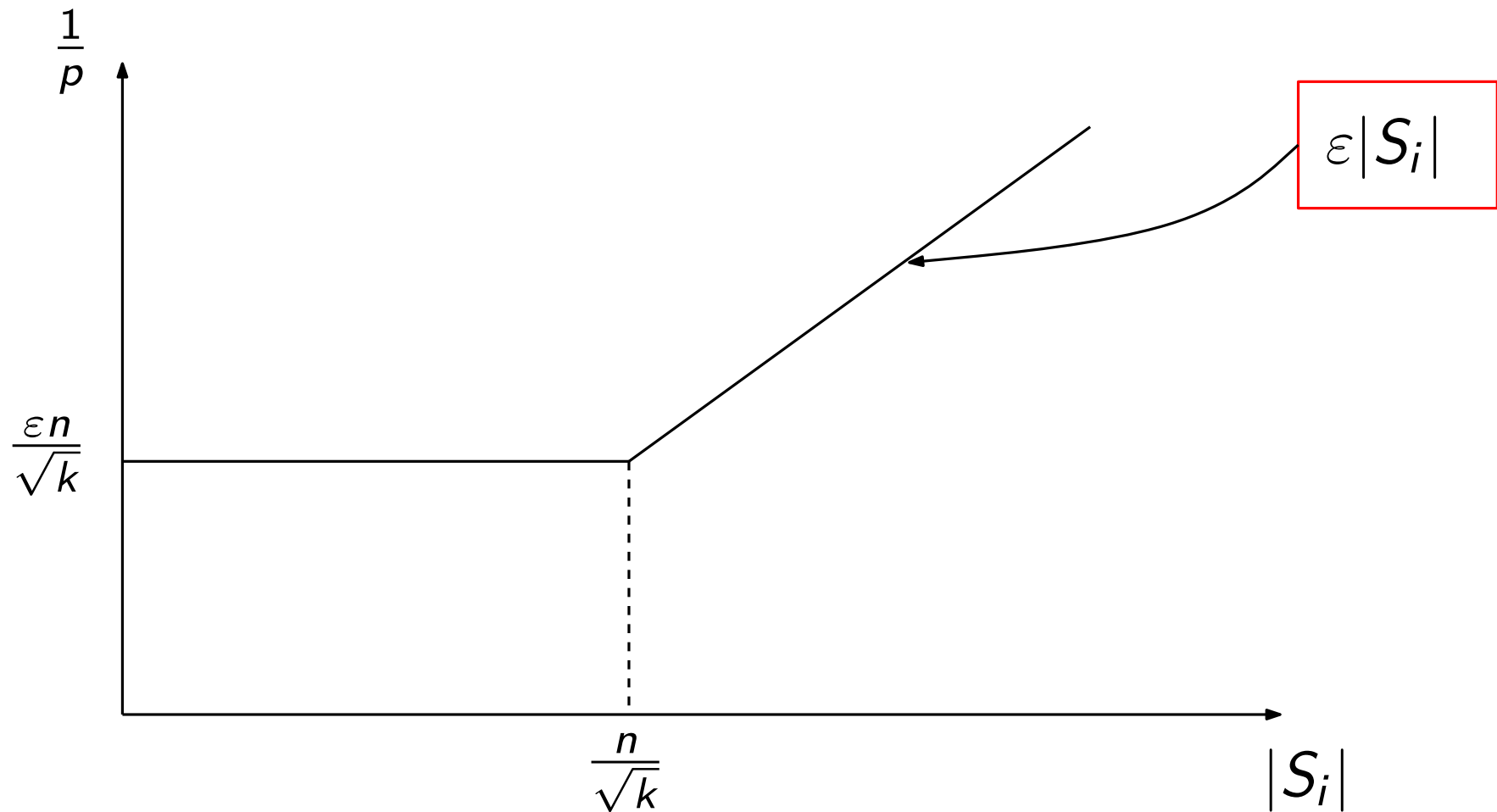
- Total cost: $np = \sqrt{k}/\varepsilon$ in expectation
- Max individual node cost: $O(\sqrt{k}/\varepsilon)$

Communication Cost

- Total cost: $np = \sqrt{k}/\varepsilon$ in expectation
- Max individual node cost: $O(\sqrt{k}/\varepsilon)$
Reduce to $O(1/\varepsilon)$

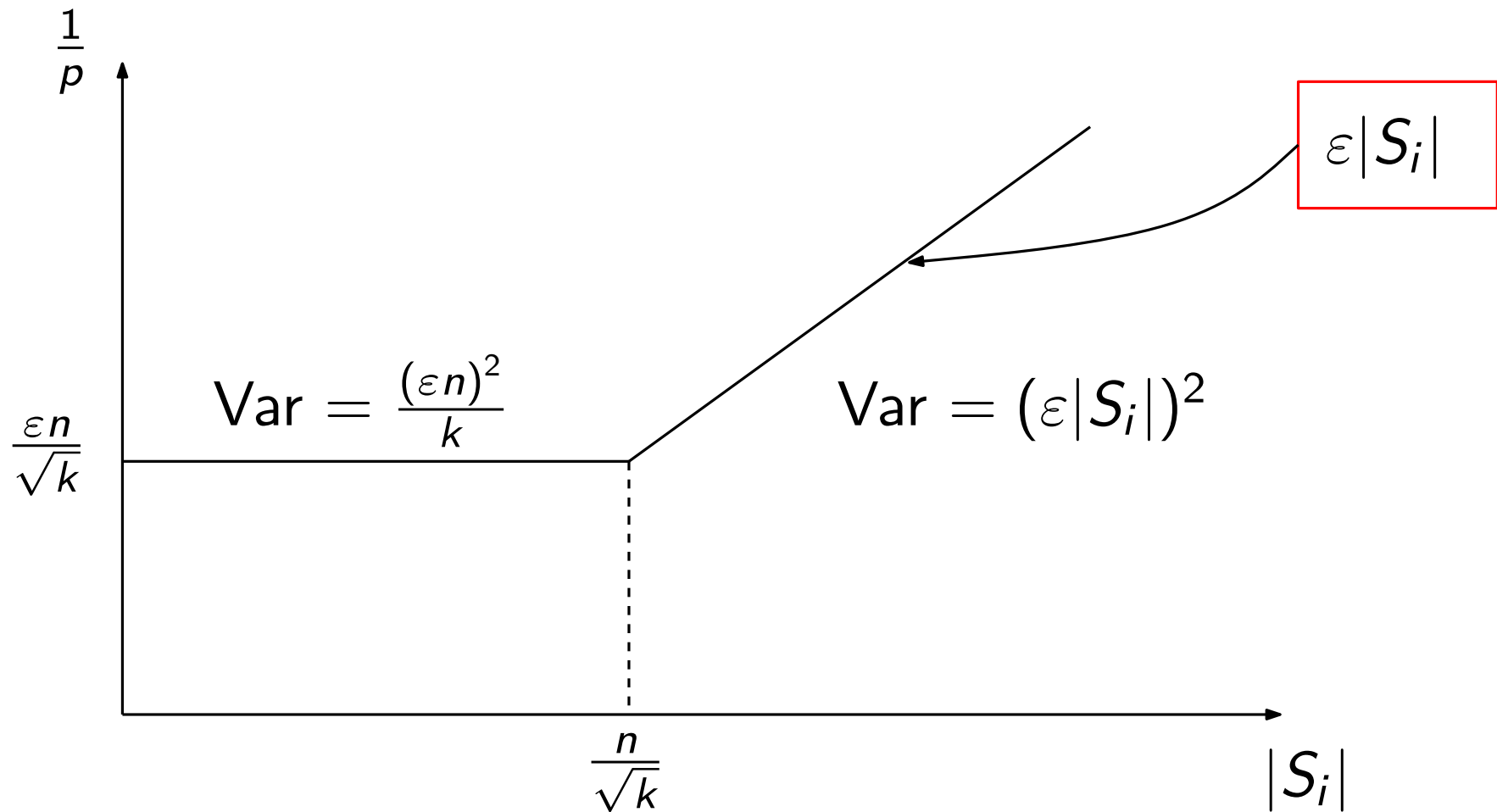
Communication Cost: Load Balancing

Let S_i be the set of data collected at node i



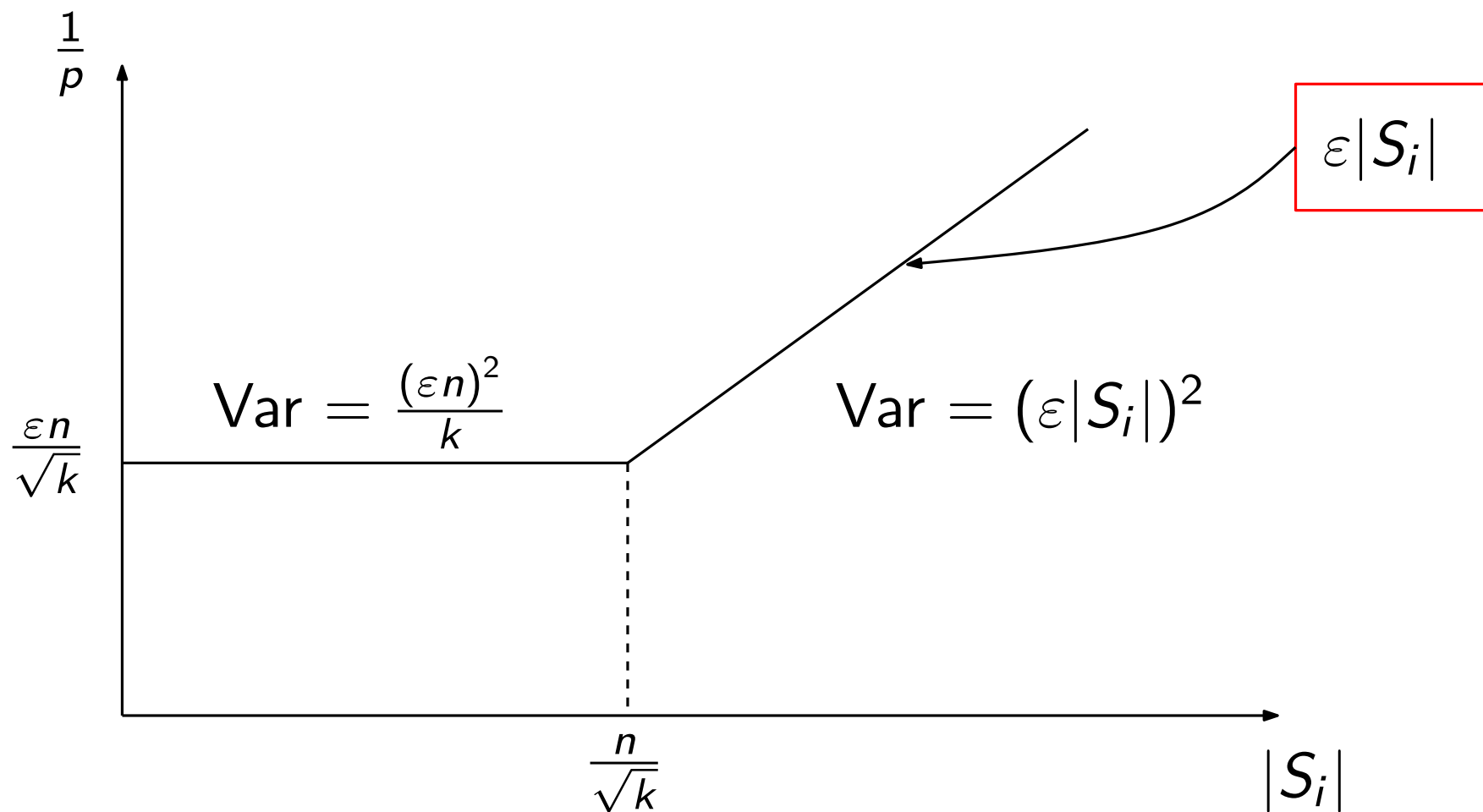
Communication Cost: Load Balancing

Let S_i be the set of data collected at node i



Communication Cost: Load Balancing

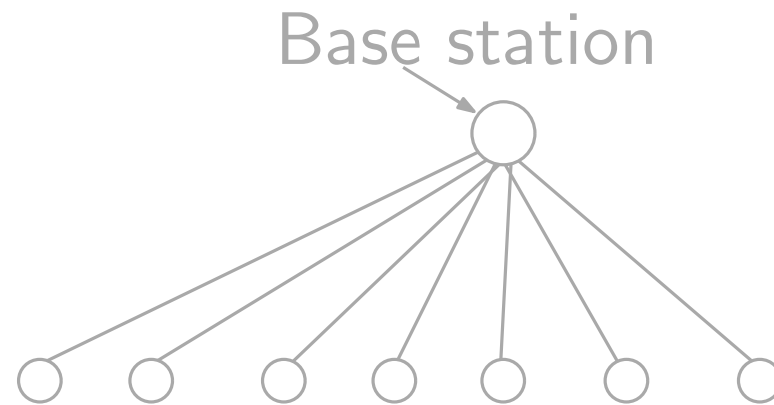
Let S_i be the set of data collected at node i



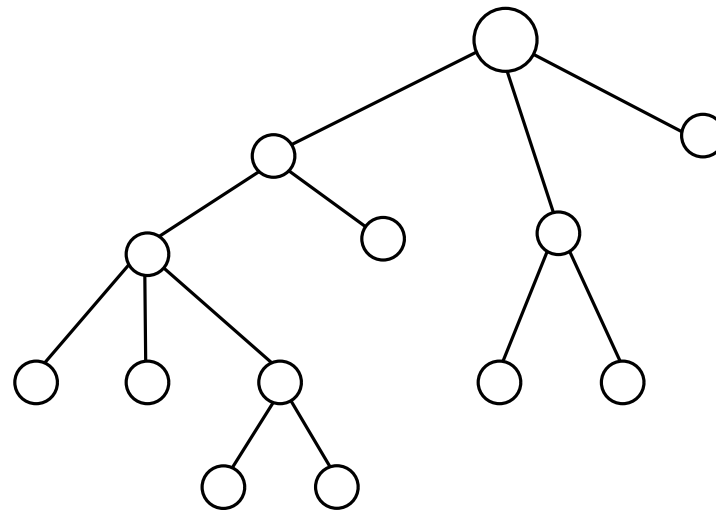
$$\text{Total variance} \leq \sum_i \left(\frac{(\epsilon n)^2}{k} + (\epsilon |S_i|)^2 \right) \leq 2(\epsilon n)^2$$

Roadmap

- Flat model

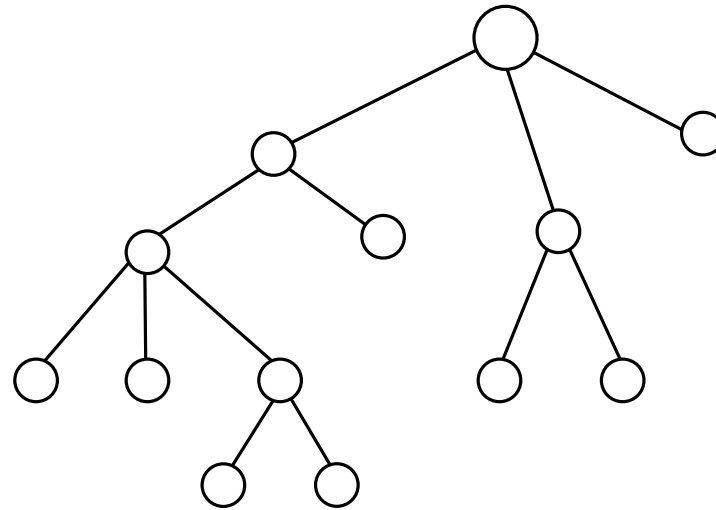


- Tree model



Tree Model: Naive Extension from Flat Model

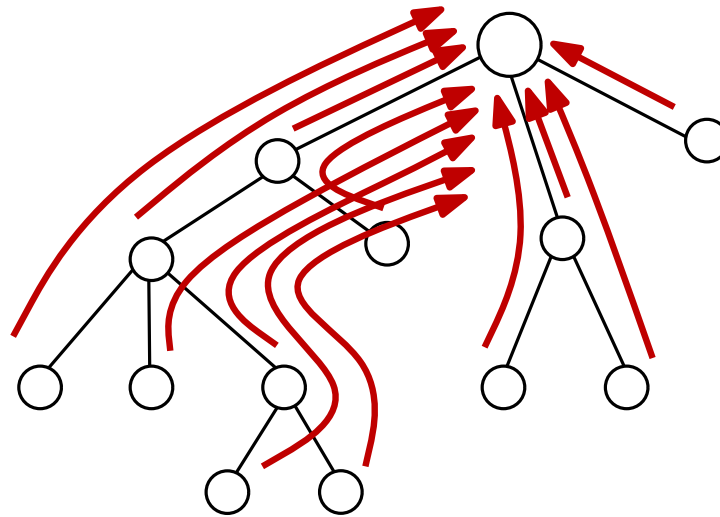
Total cost: $\frac{\sqrt{k}}{\epsilon} h$ (h is the height of the tree)



Tree Model: Naive Extension from Flat Model

Total cost: $\frac{\sqrt{k}}{\varepsilon} h$ (h is the height of the tree)

Max individual cost: $O\left(\frac{\sqrt{k}}{\varepsilon}\right)$



Tree Model: Merging Samples

S_i ① ③ ④ ⑥ ⑦ ⑨ ⑪ ⑬ ⑯ ⑰ ⑲ ⑳

p_i (3, 2) (7, 5) (13, 8) (26, 10)

$$p_i = \begin{cases} \frac{\sqrt{k}}{\varepsilon n}, & |S_i| \leq \frac{n}{\sqrt{k}} \\ \frac{1}{\varepsilon |S_i|}, & |S_i| \geq \frac{n}{\sqrt{k}} \end{cases}$$

Tree Model: Merging Samples

S_i	①	③	④	⑥	⑦	⑨	⑪	⑬	⑯	⑳	㉑	㉒
p_i		(3, 2)			(7, 5)			(13, 8)		(26, 10)		

$$p_i = \begin{cases} \frac{\sqrt{k}}{\varepsilon n}, & |S_i| \leq \frac{n}{\sqrt{k}} \\ \frac{1}{\varepsilon |S_i|}, & |S_i| \geq \frac{n}{\sqrt{k}} \end{cases}$$

$ S_1 $	p_1	(3, 2)	(7, 5)	(13, 8)	(26, 10)	Sample w.p. p/p_1
				+		
$ S_2 $	p_2	(5, 3)	(14, 6)	(18, 8)	(24, 11)	Sample w.p. p/p_2
				=		

$$|S_1| + |S_2| \quad p$$

$$\left(> \frac{n}{\sqrt{k}} \right)$$

Tree Model: Merging Samples

S_i	①	③	④	⑥	⑦	⑨	⑪	⑬	⑯	⑳	㉑	㉒
p_i		(3, 2)			(7, 5)			(13, 8)		(26, 10)		

$$p_i = \begin{cases} \frac{\sqrt{k}}{\varepsilon n}, & |S_i| \leq \frac{n}{\sqrt{k}} \\ \frac{1}{\varepsilon |S_i|}, & |S_i| \geq \frac{n}{\sqrt{k}} \end{cases}$$

$ S_1 $	p_1	(3, 2)	(7, 5)	(13, 8)	(26, 10)	Sample w.p. p/p_1
				+		
$ S_2 $	p_2	(5, 3)	(14, 6)	(18, 8)	(24, 11)	Sample w.p. p/p_2
				=		
$ S_1 + S_2 $	p			(13, 8+?)		

$$\left(> \frac{n}{\sqrt{k}} \right)$$

Tree Model: Merging Samples

S_i	①	③	④	⑥	⑦	⑨	⑪	⑬	⑬	⑯	⑲	⑳
p_i		(3, 2)			(7, 5)			(13, 8)		(26, 10)		

$$p_i = \begin{cases} \frac{\sqrt{k}}{\varepsilon n}, & |S_i| \leq \frac{n}{\sqrt{k}} \\ \frac{1}{\varepsilon |S_i|}, & |S_i| \geq \frac{n}{\sqrt{k}} \end{cases}$$

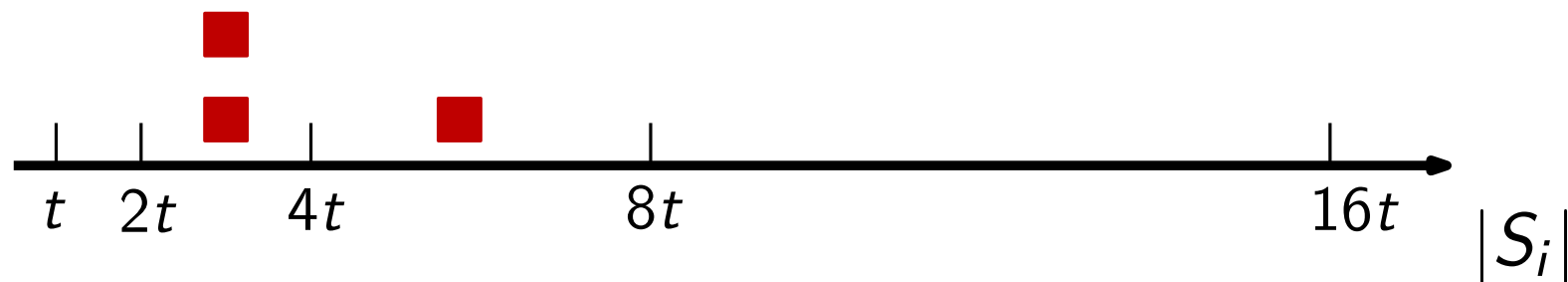
$ S_1 $	p_1	(3, 2)	(7, 5)	(13, 8)	(26, 10)	Sample w.p. p/p_1
				+		
$ S_2 $	p_2	(5, 3)	(14, 6)	(18, 8)	(24, 11)	Sample w.p. p/p_2
				=		
$ S_1 + S_2 $	p			(13, 8+?)		

This is a rank query in the other sample!

Tree Model: Merging Samples

Idea: Only merge samples with $p_1 \approx p_2$, i.e., $|S_1| \approx |S_2|$

$$t = \frac{n}{\sqrt{k}}$$

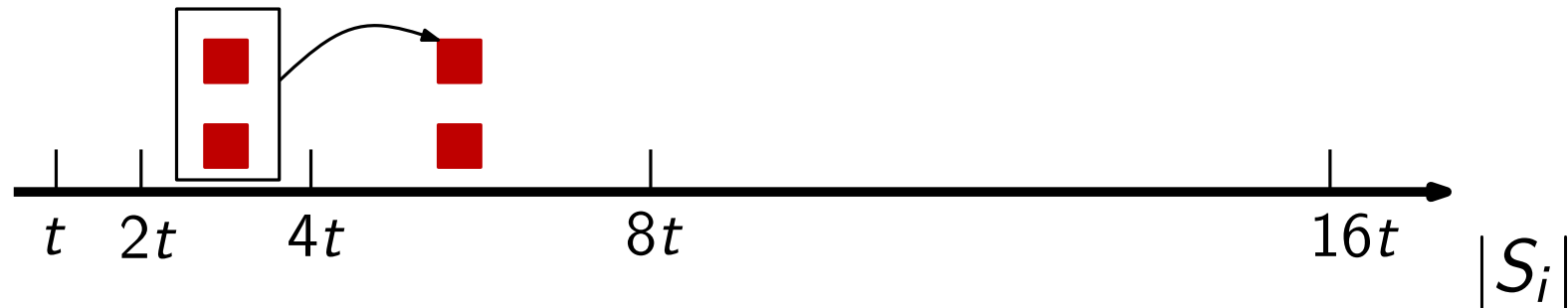


Merge samples in the same bucket

Tree Model: Merging Samples

Idea: Only merge samples with $p_1 \approx p_2$, i.e., $|S_1| \approx |S_2|$

$$t = \frac{n}{\sqrt{k}}$$

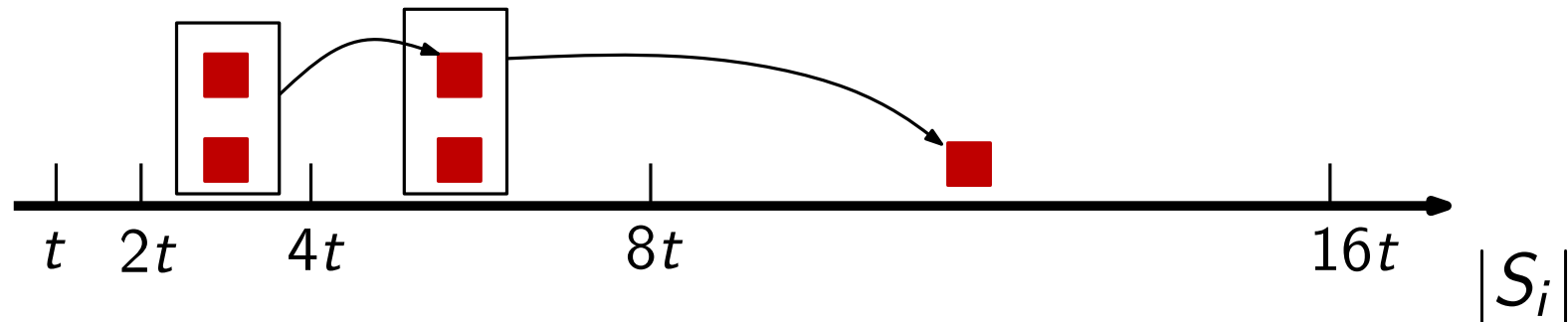


Merge samples in the same bucket

Tree Model: Merging Samples

Idea: Only merge samples with $p_1 \approx p_2$, i.e., $|S_1| \approx |S_2|$

$$t = \frac{n}{\sqrt{k}}$$



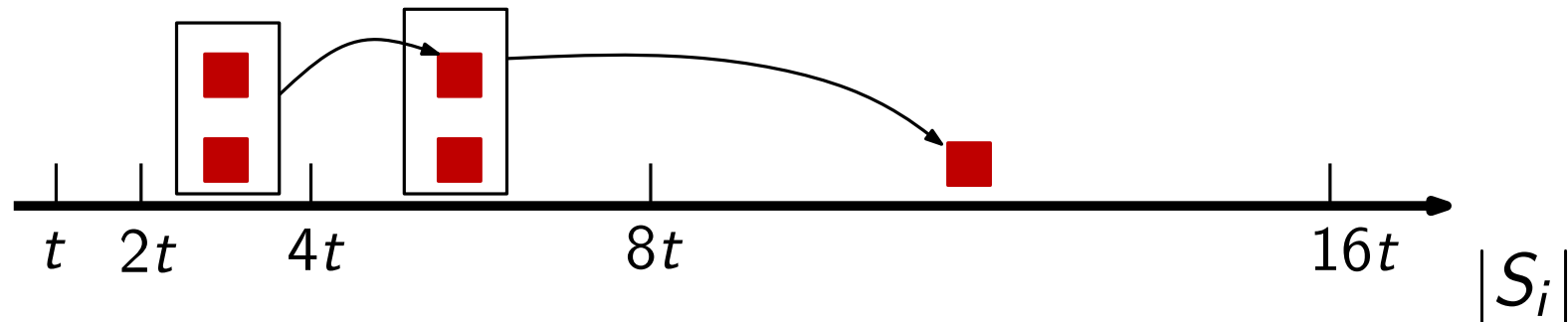
Merge samples in the same bucket

After merging, there are at most $\log \sqrt{k}$ samples
(so, the max individual cost is $O(\frac{1}{\epsilon} \log k)$)

Tree Model: Merging Samples

Idea: Only merge samples with $p_1 \approx p_2$, i.e., $|S_1| \approx |S_2|$

$$t = \frac{n}{\sqrt{k}}$$

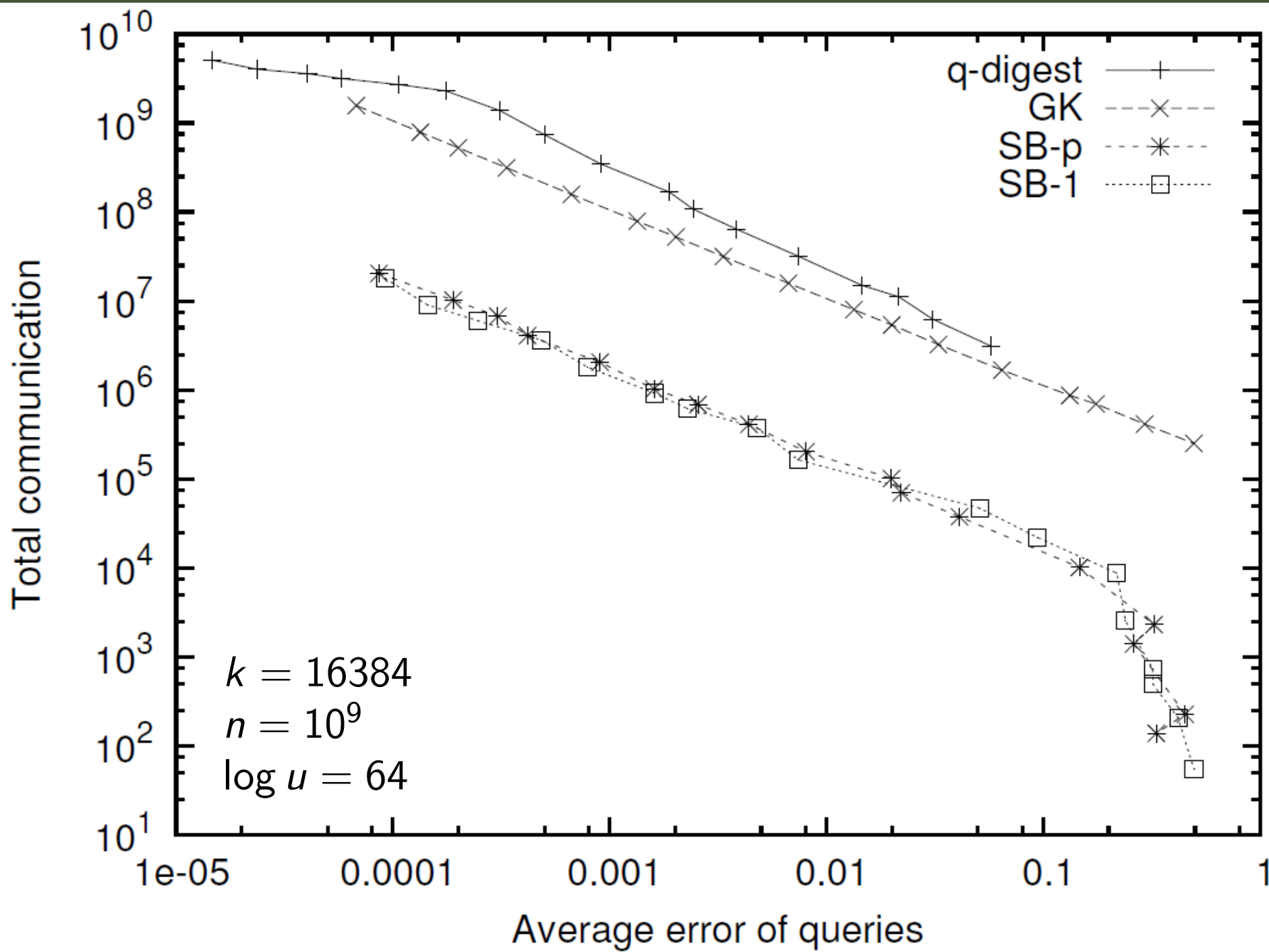


Merge samples in the same bucket

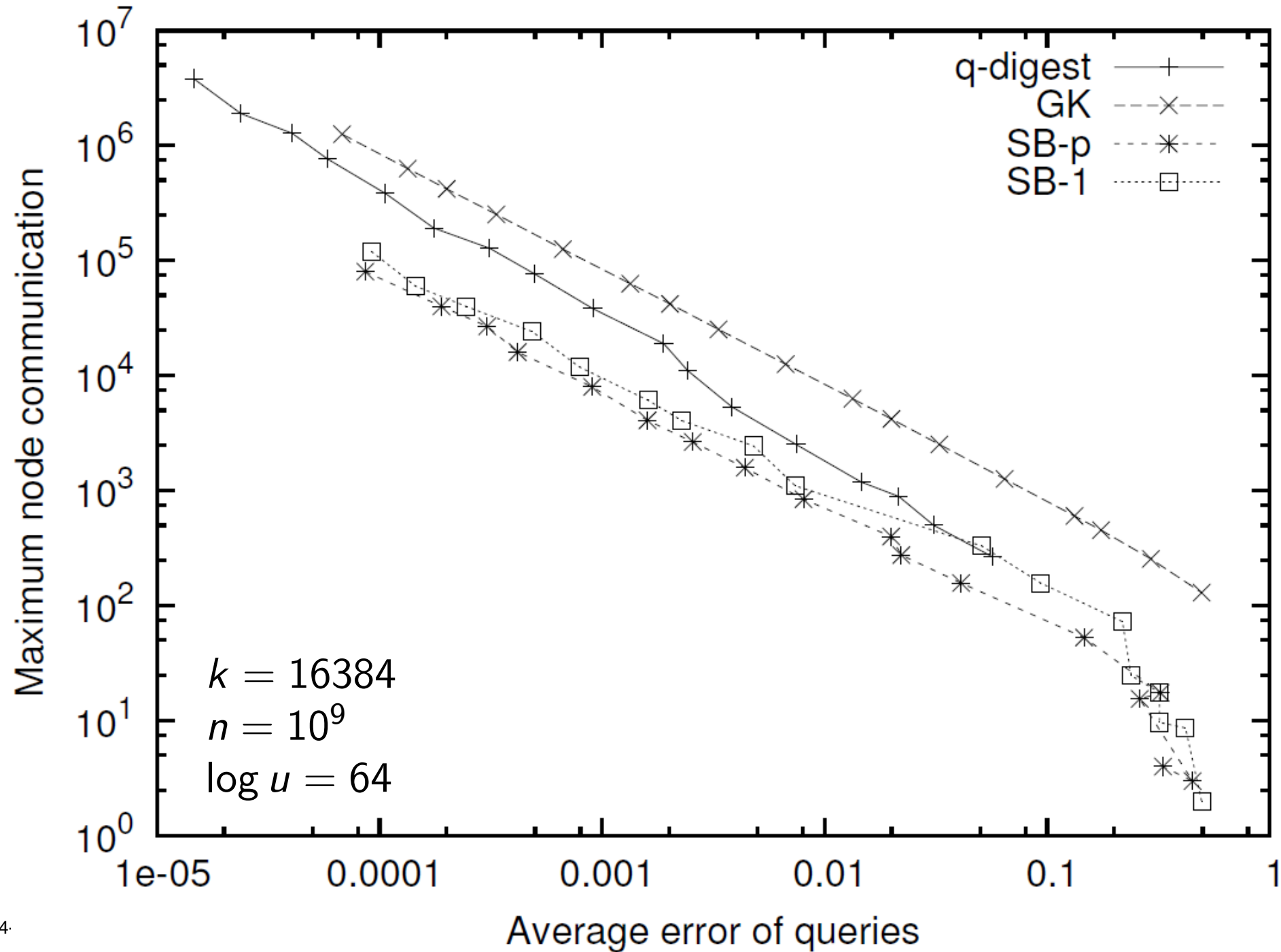
After merging, there are at most $\log \sqrt{k}$ samples
(so, the max individual cost is $O(\frac{1}{\epsilon} \log k)$)

Can show that the final variance is $O((\epsilon n)^2)$ (please see paper)

Experimental Results on Terrain Data



Experimental Results on Terrain Data



Outline

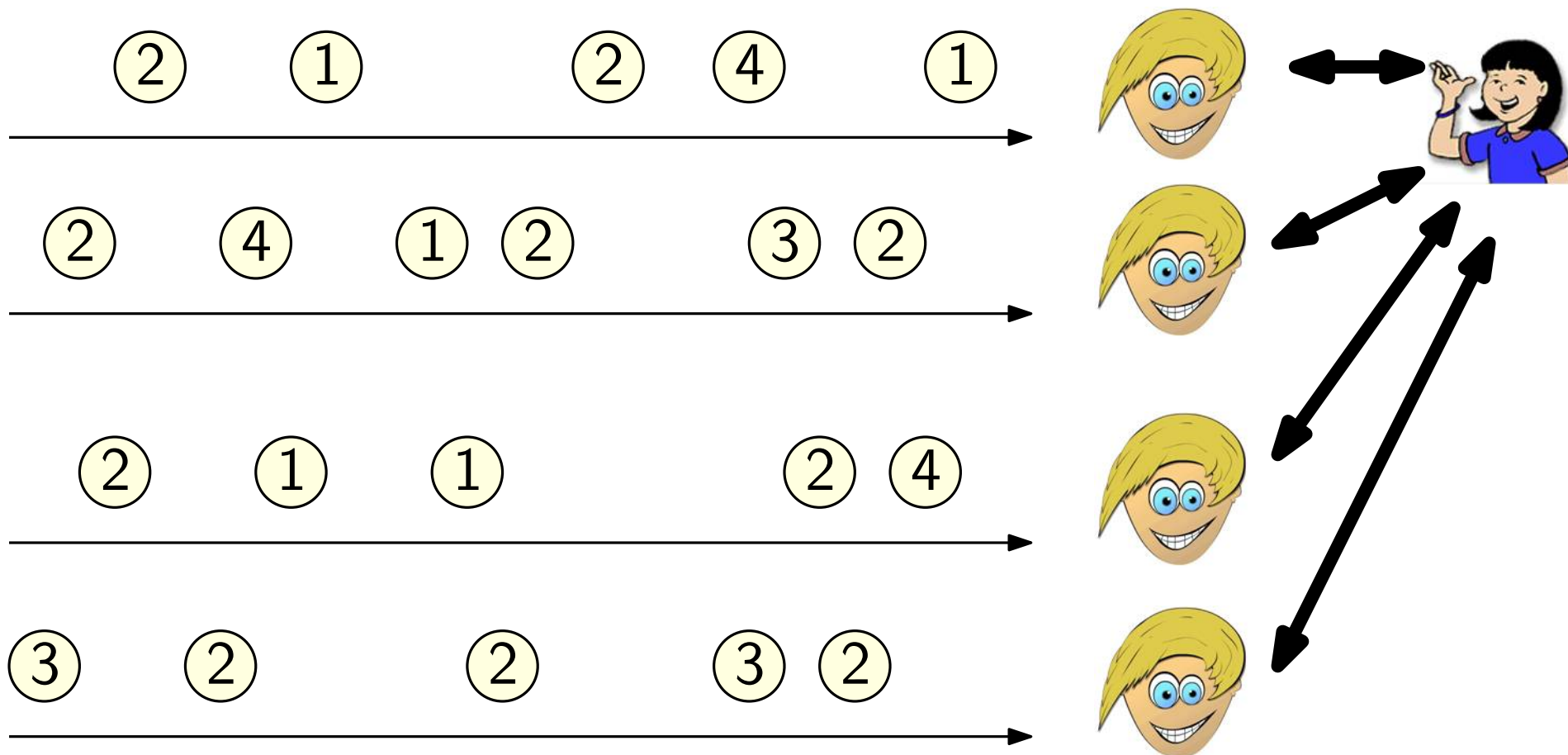
- Model of computation
- Frequency estimation (heavy hitters)
- Quantiles (order statistics)
- **Other problems**

Other Problems

- Wavelet histograms under MapReduce (VLDB'12)

Other Problems

- Wavelet histograms under MapReduce (VLDB'12)
- Continuous tracking (SODA'08, PODS'09, PODS'10)



Other Problems

- Wavelet histograms under MapReduce (VLDB'12)
- Continuous tracking (SODA'08, PODS'09, PODS'10)
sliding window

