LARGE-SCALE ELASTIC ARCHITECTURE FOR DATA AS A SERVICE

FERARI

Streaming Big Data Analytics

Minos Garofalakis

ATHENA Research & Innovation Centre

and Technical University of Crete

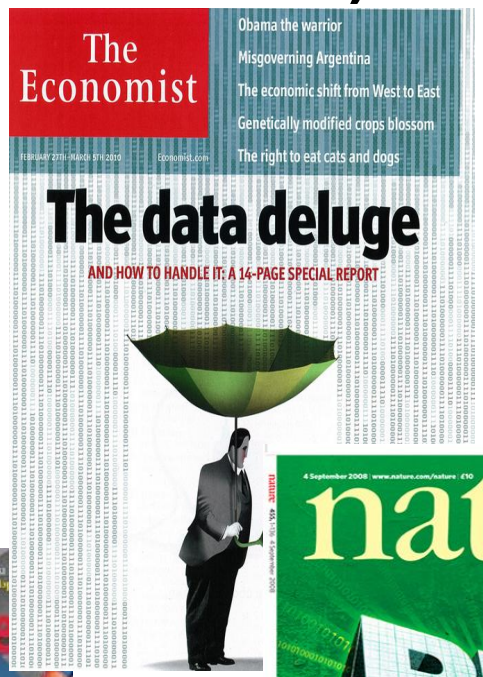
http://www.softnet.tuc.gr/~minos/

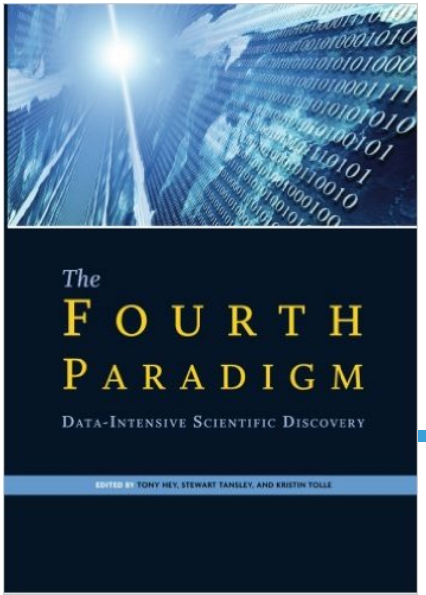
ATHENA Research & Innovation Information Technologies

Big Data is Big News (and Big Business…)

- Rapid growth due to several information-generating technologies, such as mobile computing, sensornets, and social networks

- How can we cost-effectively manage and analyze all this data…?

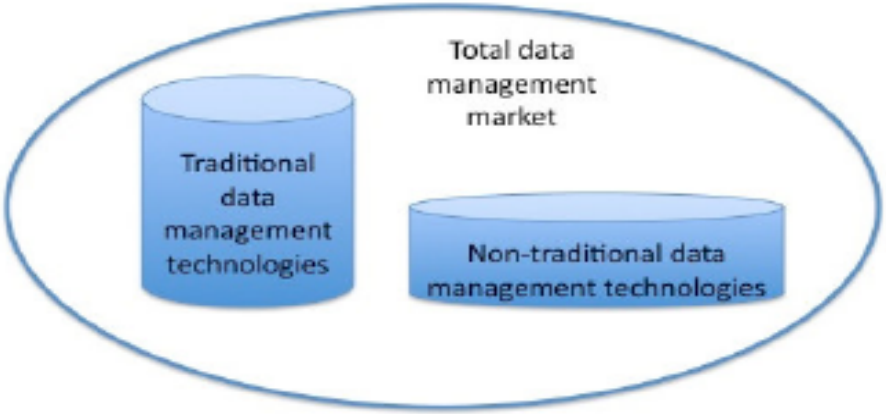


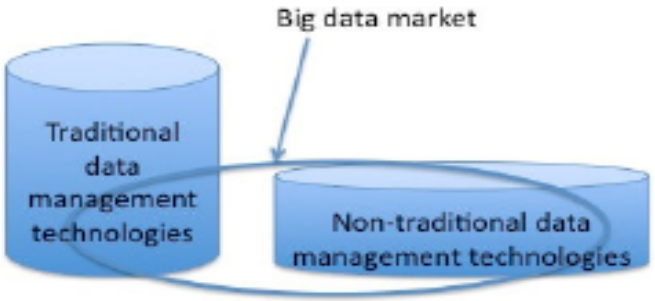


What is "Big Data"? - 451 Group - Definition

"Big data is a term applied to data sets that are large, complex and dynamic (or a combination thereof) and for which there is a requirement to capture, manage and process the data set in its entirety, such that it is not possible to process the data using traditional software tools and analytic techniques within tolerable time frames ."

451 Group, Sizing the Big Data Problem





Big Data Challenges: The Four V's (and one D)…

- **Volume:** Scaling from Terabytes to Exa/Zettabytes
- **Velocity:** Processing massive amounts of streaming data
- **Variety:** Managing the complexity of multiple relational and non-relational data types and schemas
- **Veracity:** Handling the inherent uncertainty and noise in the data
- **Distribution:** Dealing with massively distributed information
- ***Our focus : Volume, Velocity, Distribution…***
  - **Basic algorithmic tools for Centralized and Distributed Streams**

LARGE-SCALE ELASTIC ARCHITECTURE FOR DATA AS A SERVICE

FERARI

PART I: Centralized Data Streaming

ATHENA Research & Innovation Information Technologies

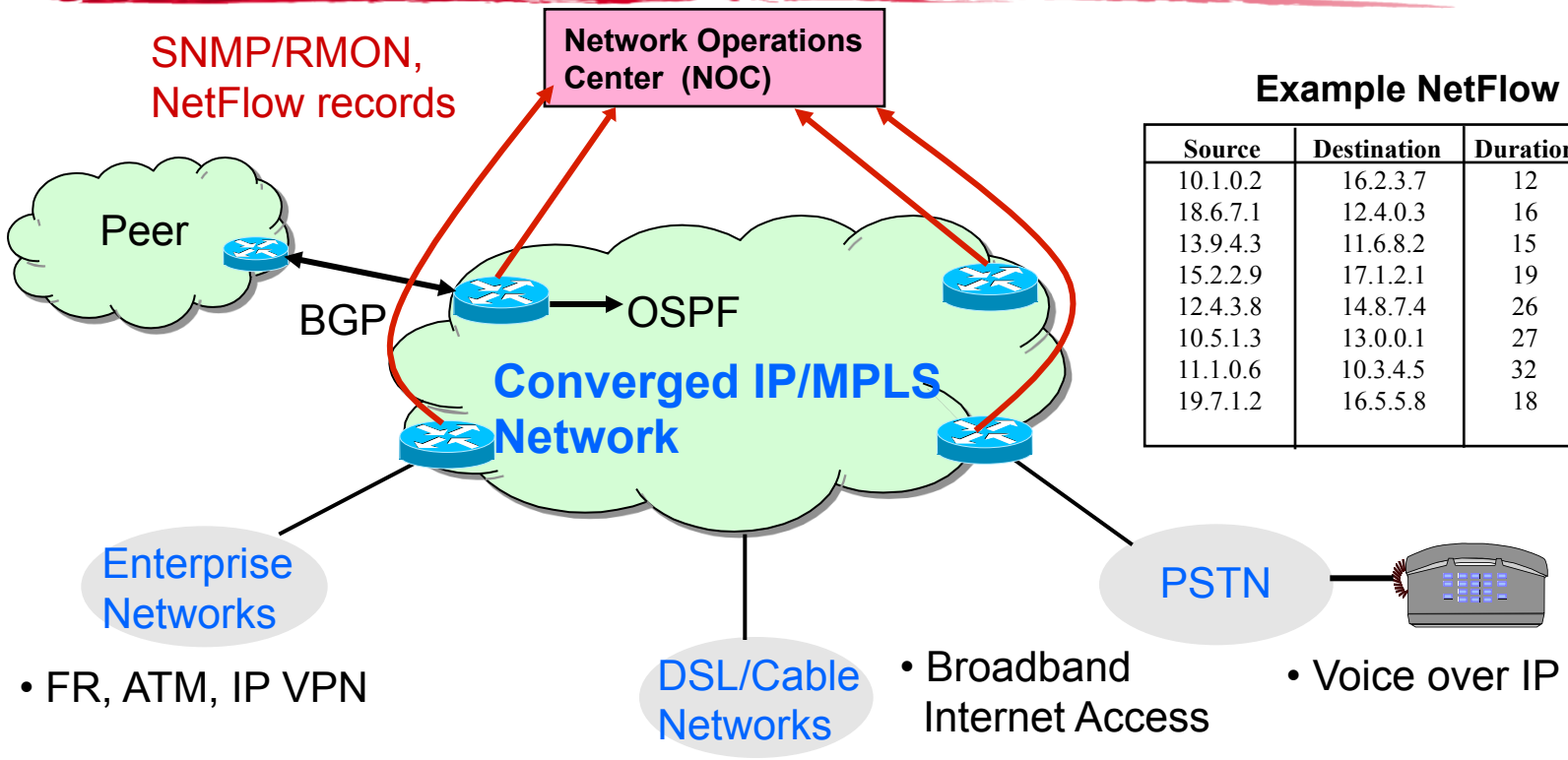
Data Stream Management

- Traditional DBMS – data stored in finite, persistent data sets

- Data Streams – distributed, continuous, unbounded, rapid, time varying, noisy, . . .

- Data-Stream Management – variety of modern applications
  - Network monitoring and traffic engineering
  - Telecom call-detail records
  - Network security
  - Financial applications
  - Sensor networks
  - Manufacturing processes
  - Web logs and clickstreams
  - Massive data sets

Networks Generate Massive Data Streams



Example NetFlow IP Session Data

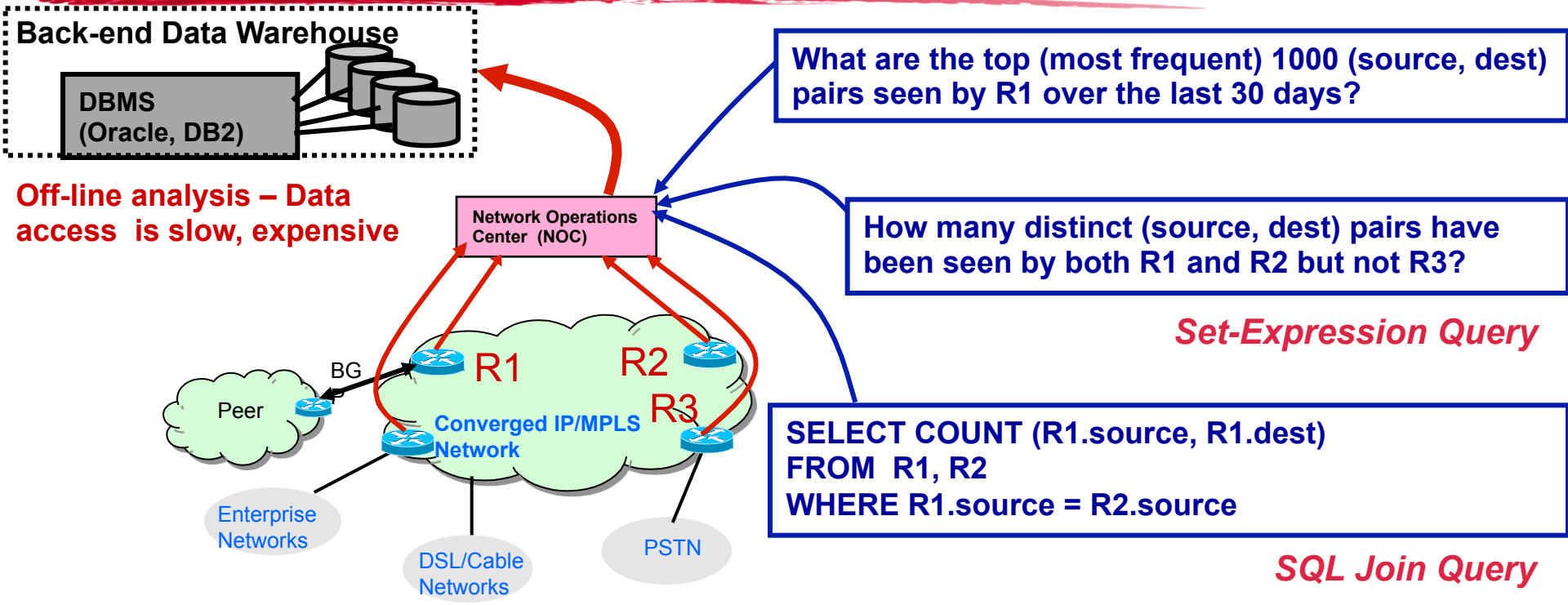
Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

- SNMP/RMON/NetFlow data records arrive 24x7 from different parts of the network
- Truly massive streams arriving at rapid rates
  - AT&T collects several Terabytes of NetFlow data each day!
- Typically shipped to a back-end data warehouse for off-line analysis

Packet-Level Data Streams

- Single 2Gb/sec link; say avg packet size is 50bytes
- Number of packets/sec = 5 million
- Time per packet = 0.2 µsec
- If we only capture header information per packet: src/dest IP, time, no. of bytes, etc. – at least 10bytes.
  - Space per second is 50Mb
  - Space per day is 4.5Tb per link
  - ISPs typically have hundreds of links!
- Analyzing packet content streams – whole different ballgame!!

Real-Time Data-Stream Analysis

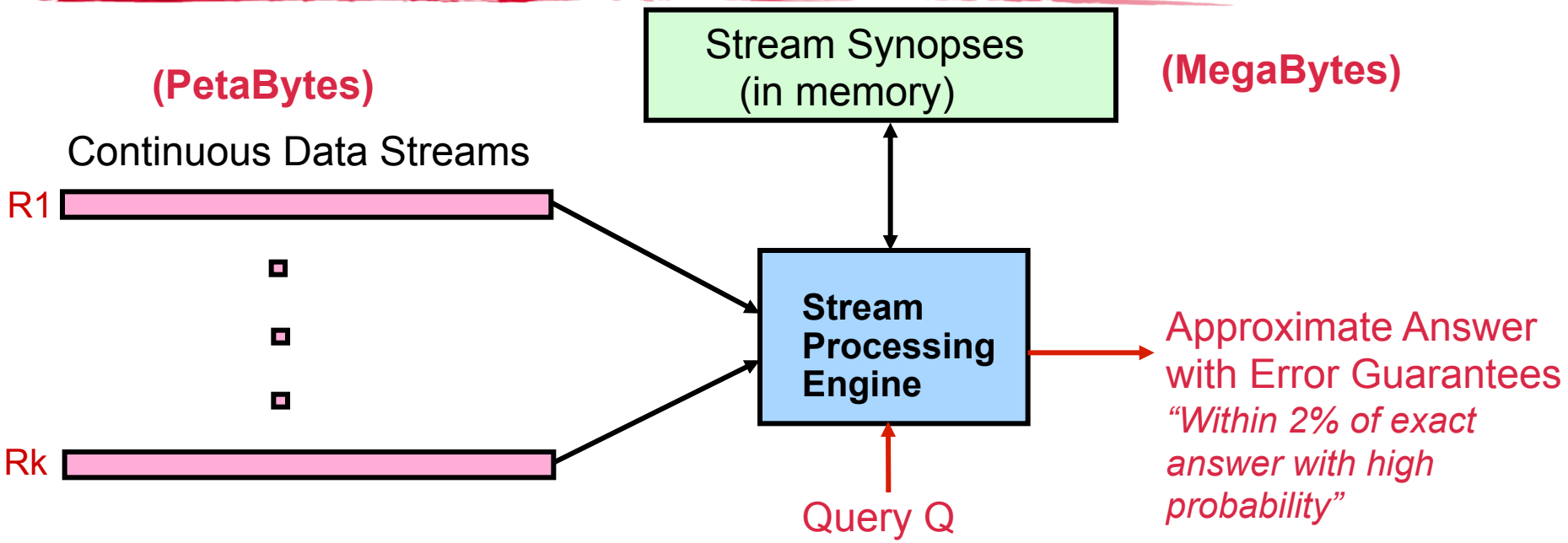


- Need ability to process/analyze network-data streams *in real-time*
  - As records stream in: look at records *only once in arrival order!*
  - Within resource (CPU, memory) limitations of the NOC
- Critical to important NM tasks
  - Detect and react to Fraud, DoS attacks, SLA violations
  - Real-time traffic engineering for load-balancing and utilization

Outline – Part I

- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
  - Sampling
  - Sketching Multisets: Linear-Projection Sketches -- *AMS, CountMin*
    - *Applications:* Joins, Top-k, Range sums, Wavelets
  - Sketching Sets: FM Sketches
    - *Applications:* Distinct Values, Distinct Sampling
  - *(Time permitting)* Sliding Window model and Exponential Histograms (EHs)

Data-Stream Processing Model



- Approximate answers often suffice, e.g., trend analysis, anomaly detection
- Requirements for stream synopses
  - *Single-Pass:* Each record is examined at most once, in (fixed) arrival order
  - *Small-Space:* Log or polylog in data stream size
  - *Small-Time:* Per-record processing time (to maintain synopses) must be low
  - *Delete-Proof:* Can handle record deletions as well as insertions
  - *Composable:* Built in a *distributed fashion* and combined later

Data Stream Processing Algorithms

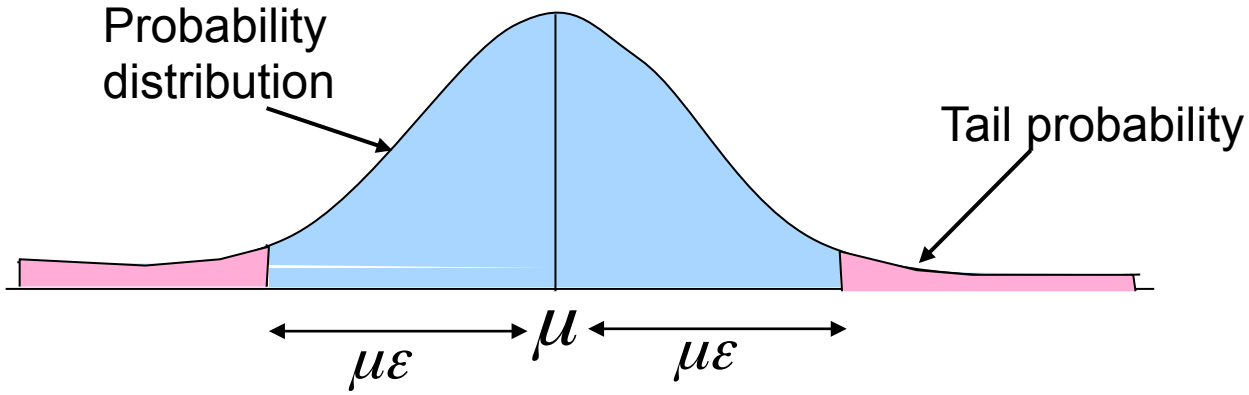
- Generally, algorithms compute approximate answers
  - Provably difficult to compute answers accurately with limited memory
- Approximate answers - Deterministic bounds
  - Algorithms only compute an approximate answer, but bounds on error
- Approximate answers - Probabilistic bounds
  - Algorithms compute an approximate answer with high probability
    - With probability at least $1-\delta$, the computed answer is within an error ε of the actual answer
- *Single-pass algorithms* for processing streams also applicable to (massive) databases!

Probabilistic Guarantees

- Example: Actual answer is within 5 ± 1 with prob ≥ 0.9
- Randomized algorithms: Answer returned is a specially-built random variable
- User-tunable (ε,δ)–**approximations**
  - Estimate is within (relative) error of ε with probability >= $1-\delta$
- Use Tail Inequalities to give probabilistic bounds on returned answer
  - Markov Inequality
  - Chebyshev's Inequality
  - Chernoff Bound
  - Hoeffding Bound

Basic Tools: Tail Inequalities

- General bounds on *tail probability* of a random variable (that is, probability that a random variable deviates far from its expectation)



- Basic Inequalities: Let X be a random variable with expectation μ and variance Var[X]. Then for any $\varepsilon > 0$

Markov: $$\Pr(X \geq \mu + \varepsilon) \leq \frac{\mu}{\mu + \varepsilon}$$

Chebyshev: $$\Pr(|X - \mu| \geq \varepsilon) \leq \frac{\mathrm{Var}[X]}{\varepsilon^2}$$

Tail Inequalities for Sums

- Possible to derive even stronger bounds on tail probabilities for the sum of independent *Bernoulli trials*

- **Chernoff Bound:**

  Let X1, …, Xm be independent Bernoulli trials such that Pr[Xi=1] = p (Pr[Xi=0] = 1-p). Let $X = \sum_i X_i$ and $\mu = mp$ be the expectation of X. Then, for any $\varepsilon > 0$,

$$\Pr(|X - \mu| \geq \mu\varepsilon) \leq 2\exp^{\frac{-\mu\varepsilon^2}{2}}$$

Outline – Part I

- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
  - Sampling
  - Sketching Multisets: Linear-Projection Sketches -- *AMS, CountMin*
    - *Applications:* Joins, Top-k, Range sums, Wavelets
  - Sketching Sets: FM Sketches
    - *Applications:* Distinct Values, Distinct Sampling
  - *(Time permitting)* Sliding Window model and Exponential Histograms (EHs)

Sampling: Basics

- Idea: A small random sample S of the data often well-represents all the data
  - For a fast approx answer, apply "modified" query to S
  - Example: select agg from U where U.e is odd

Data stream: 9 3 5 2 7 1 6 5 8 4 9 1 (n=12)

Sample S: 9 5 1 8

  - If agg is avg, return average of odd elements in S **answer: 5**
  - If agg is count, return scaled count of all odd elements e in S

answer: 3*|U|/|S| = 3*3 =9

Unbiased: For expressions involving count, sum, avg: the estimator is unbiased, i.e., the expected value of the answer is the actual answer

Sampling from a Stream

- Reservoir Sampling [Vit85]: Maintains a sample R of a fixed-size M
  - Add each new element to R with probability M/n, where n is the current number of stream elements
  - If add an element, evict a random element from R
  - *[Vitter] Instead of flipping a coin for each element, determine the number of elements to skip before the next to be added to R*

- Concise sampling [GM98]: Duplicates in sample R stored as <value, count> pairs (thus, potentially boosting actual sample size)
  - Add each new element to R with probability 1/T (simply increment count if element already in R)
  - If sample size exceeds M
    - Select new threshold T' > T
    - Evict each element (decrement count) from R with prob. 1-T/T'
  - Add subsequent elements to R with probability 1/T'

Reservoir Sampling Analysis

- R(n) = reservoir after the nth arrival x(n), size of reservoir = M
- Want to show that after nth arrival

$$\Pr[\, x(i) \text{ in } R(n) \,] = M/n \qquad \text{for all } n \text{ and } i<=n$$

- Proof **by induction** – assume true for n=k, show for n = k+1

For x(k+1): $\Pr[\, x(k+1) \text{ in } R(k+1) \,] = M/(k+1)$ (RS algorithm)

For x(i), i<k+1:

$$\Pr[\, x(i) \text{ in } R(k+1) \,] = \Pr[x(i) \text{ in } R(k)] * \Pr[x(i) \text{ in } R(k+1) \mid x(i) \text{ in } R(k)]$$

Reservoir Sampling Analysis

For x(i), i<k+1:

$$\Pr[\, x(i) \text{ in } R(k+1) \,] = \Pr[x(i) \text{ in } R(k)] * \Pr[x(i) \text{ in } R(k+1) \mid x(i) \text{ in } R(k)]$$

$$= \Pr[x(i) \text{ in } R(k)] * (\Pr[x(k+1) \text{ is NOT chosen}] + \Pr[x(k+1) \text{ is chosen}] * \Pr[x(i) \text{ not evicted}])$$

$$= M/k * ((1 - M/(k+1)) + M/(k+1) * (1 - 1/M))$$

$$= \ldots = M / (k+1)$$ **QED**

Simple, elegant *… and incomplete!!*

Reservoir Sampling Analysis

- Uniform Random Sampling: Probability of selecting a subset of size M out of N elements = 1 / (N choose M)

- Proof for RS based on the correctness of the *"Fisher-Yates Shuffle"*
  - Construct a provably random permutation of S[1…N]

    For i := 1 to N do

    Generate random x in [1…i]

    Swap S[i] with S[x]

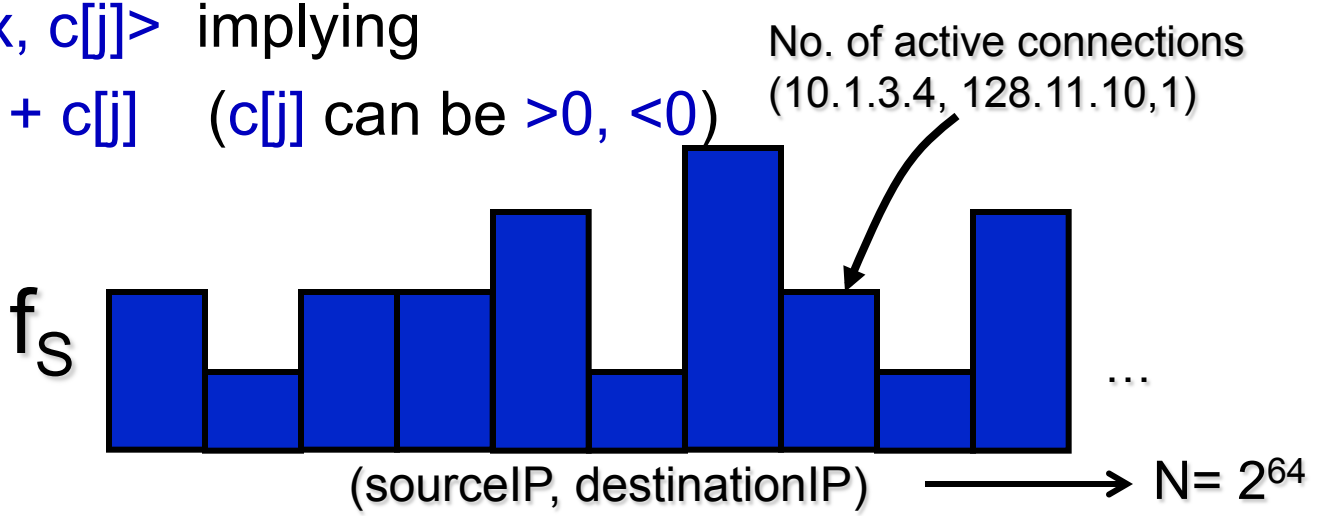
  - Each possible permutation with probability 1/N!
  - Essentially the RS algorithm! (Reservoir = S[1…M)

Outline – Part I

- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
  - Sampling
  - Sketching Multisets: Linear-Projection Sketches -- *AMS, CountMin*
    - *Applications:* Joins, Top-k, Range sums, Wavelets
  - Sketching Sets: FM Sketches
    - *Applications:* Distinct Values, Distinct Sampling
  - *(Time permitting)* Sliding Window model and Exponential Histograms (EHs)

General Model of a Relational Stream

- Relation "signal": *Large* array $f_S[1…N]$ with values $f_S[i]$ all initially zero
  - Frequency-distribution array of S
  - Multi-dimensional arrays as well (e.g., row-major)
- Relation implicitly rendered via a *stream of updates*
  - j-th update <x, c[j]> implying
    - $f_S[x] := f_S[x] + c[j]$ (c[j] can be >0, <0)



- ***Goal:*** **Compute queries (functions) on such dynamic vectors in "small" space and time (<< N)**

Example IP Network Signals

- Number of bytes (packets) sent by a source IP address during the day
  - 2^(32) sized vector; increment only
- Number of flows between a source-IP, destination-IP address pair during the day
  - 2^(64) sized vector; increment only, aggregate packets into flows
- Number of active flows per source-IP, destination-IP address pair
  - 2^(64) sized vector; increment and decrement

Streaming Model: Special Cases

- Time-Series Model
  - Only j-th update updates f[j] (i.e., f[j] := c[j])
- Cash-Register Model
  - c[j] is always >= 0 (i.e., increment-only)
  - Typically, c[j]=1, so we see a multi-set of items in one pass
- Turnstile Model
  - Most general streaming model
  - c[j] can be >0 or <0 (i.e., increment or decrement)
- *Problem difficulty varies depending on the model*
  - E.g., MIN/MAX in Time-Series vs. Turnstile!

Example Queries/Functions

- Vector Lp-norms $Q = \|f\|_p = \left(\sum_i (f[i])^p\right)^{1/p}$
  - Special cases: L0 (no. of distinct values), L2 (Euclidean norm, Second moment, "self-join"), L1 (simple sum), …

- Inner product queries $Q = \sum_i f[i]g[i]$
  - Join and multi-join aggregates

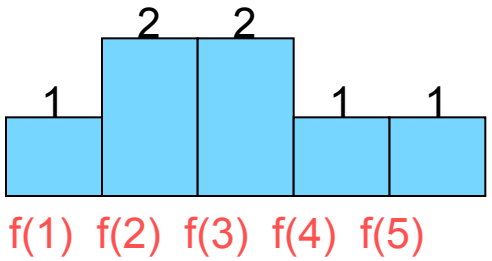
- Topk-entries, "Heavy-hitters"
  - Track entries that are "large" (i.e., f[i] > threshold)

- Other transforms/functions of f[]
  - Histograms, Wavelets, Range aggregates, Entropy, …

AMS Sketch Synopses

- *Goal:* Build small-space summary for distribution vector f(i) (i=1,..., N) seen as a stream of i-values

Data stream: 3, 1, 2, 4, 2, 3, 5, . . .



- *Basic Construct:* *Randomized Linear Projection of f()* = project onto inner/dot product of f-vector

$$\langle f, \xi \rangle = \sum f(i)\xi_i$$

where ξ= vector of random values from an appropriate distribution

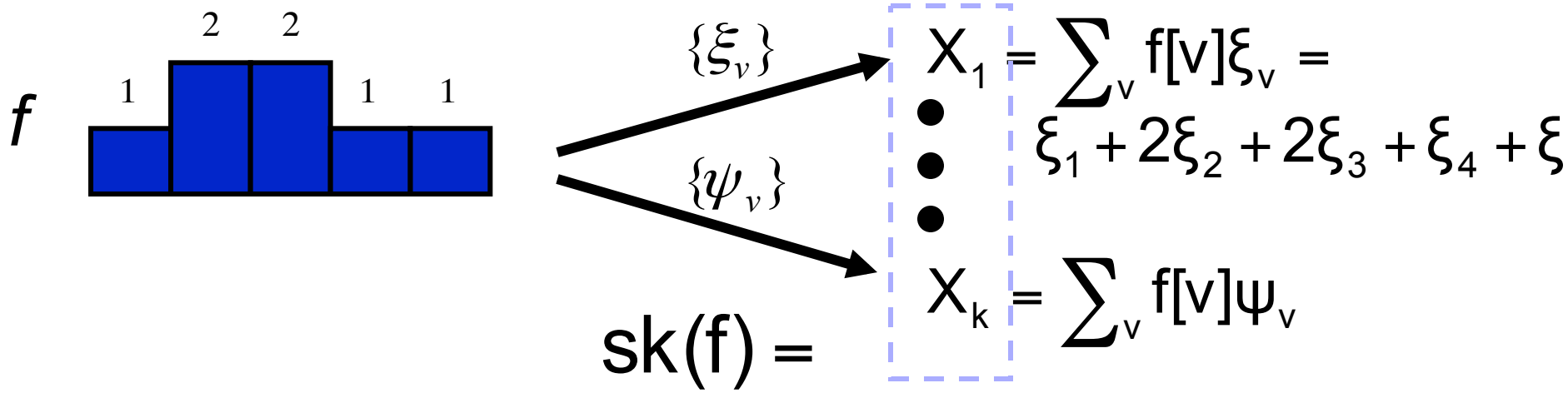
  - Simple to compute over the stream: Add ξ_i whenever the i-th value is seen

Data stream: 3, 1, 2, 4, 2, 3, 5, . . .

$$\xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5$$

  - Generate ξ_i's in small (logN) space using pseudo-random generators
  - *Tunable probabilistic guarantees* on approximation error
  - *Delete-Proof:* Just subtract ξ_i to delete an i-th value occurrence
  - *Composable:* Simply *add* independently-built projections

AMS Sketching 101



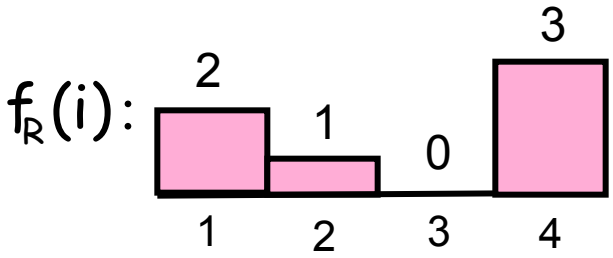
- Simple randomized linear projections of data distribution
  - Easily computed over stream using logarithmic space
  - *Linear:* Compose through simple addition
- *Theorem[AGMS]:* Given sketches of size k= $O(\frac{\log(1/\delta)}{\varepsilon^2})$

$$sk(f_R) \cdot sk(f_S) \in f_R \cdot f_S \pm \varepsilon \| f_R \|_2 \| f_S \|_2$$

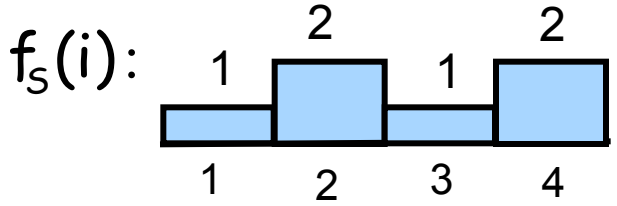
Example: Binary-Join COUNT Query

- Problem: Compute answer for the query COUNT($R \bowtie_A S$)
- Example:

Data stream R.A: 4 1 2 4 1 4



Data stream S.A: 3 1 2 4 2 4



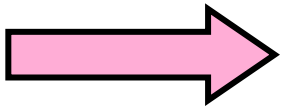
$$COUNT(R \bowtie_A S) = \sum_i f_R(i) \cdot f_S(i)$$

$$= 10 \quad (2 + 2 + 0 + 6)$$

- Exact solution: too expensive, requires O(N) space!
  - N = sizeof(domain(A))

Basic AMS Sketching Technique [AMS96]

- Key Intuition: Use randomized linear projections of f() to define random variable X such that
  - X is easily computed over the stream (in small space)
  - E[X] = COUNT(R $\bowtie_A$ S)
  - Var[X] is small



Probabilistic error guarantees (e.g., actual answer is 10±1 with probability 0.9)

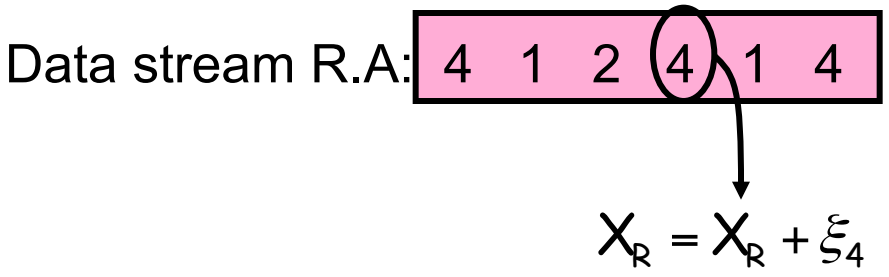
- Basic Idea:
  - Define a family of 4-wise independent {-1, +1} random variables

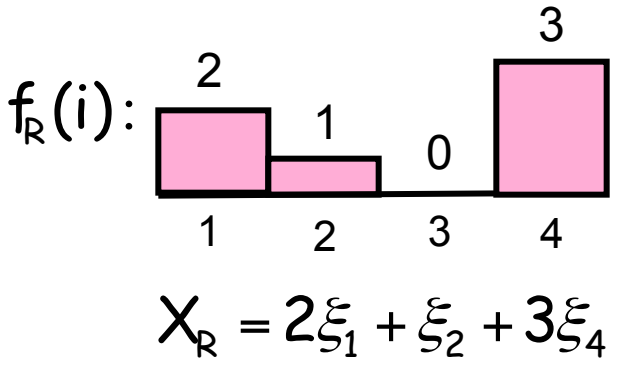
$$\{\xi_i : i = 1,\ldots,N\}$$

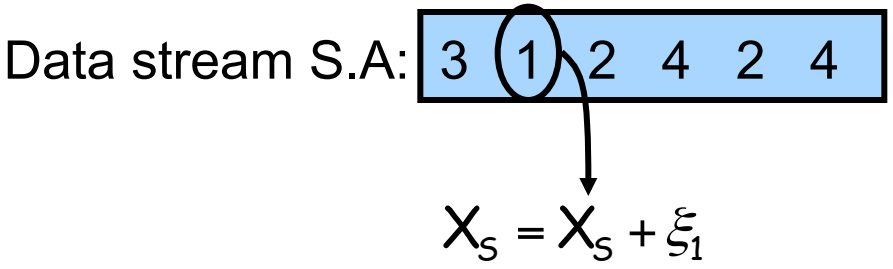
  - Pr[ξ_i = +1] = Pr[ξ_i = -1] = 1/2
    - Expected value of each ξ_i, E[ξ_i] = 0
  - Variables ξ_i are 4-wise independent
    - Expected value of product of 4 distinct ξ_i = 0
  - Variables ξ_i can be generated using pseudo-random generator using only O(log N) space (for seeding)!

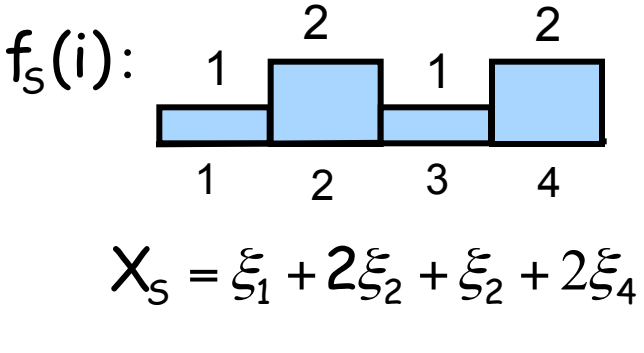
AMS Sketch Construction

- Compute random variables: $X_R = \sum_i f_R(i)\xi_i$ and $X_S = \sum_i f_S(i)\xi_i$
  - Simply add ξ_i to $X_R(X_S)$ whenever the i-th value is observed in the R.A (S.A) stream
- Define $X = X_R X_S$ to be estimate of COUNT query
- Example:









Binary-Join AMS Sketching Analysis

- Expected value of X = COUNT($R \bowtie_A S$)

$$E[X] = E[X_R \cdot X_S]$$

$$= E[\sum_i f_R(i)\xi_i \cdot \sum_i f_S(i)\xi_i]$$

$$= E[\sum_i f_R(i) f_S(i) \xi_i^2] + E[\sum_{i \neq i'} f_R(i) f_S(i') \xi_i \xi_{i'}]$$

$$= \sum_i f_R(i) \cdot f_S(i)$$

$\xi_i^2 = 1$

$E[\sum_{i \neq i'} f_R(i) f_S(i') \xi_i \xi_{i'}] = 0$

- Using 4-wise independence, possible to show that

$$Var[X] \leq 2 \|f_R\|_2^2 \|f_S\|_2^2$$

Boosting Accuracy

- Chebyshev's Inequality:

$$\Pr(|X - E[X]| \geq k) \leq \frac{Var[X]}{k^2}$$

- Boost accuracy to ε by averaging over several independent copies of X (reduces variance)

$$s = \frac{16}{\varepsilon^2} \quad \text{copies} \qquad E[Y] = E[X] = COUNT(R \bowtie S)$$

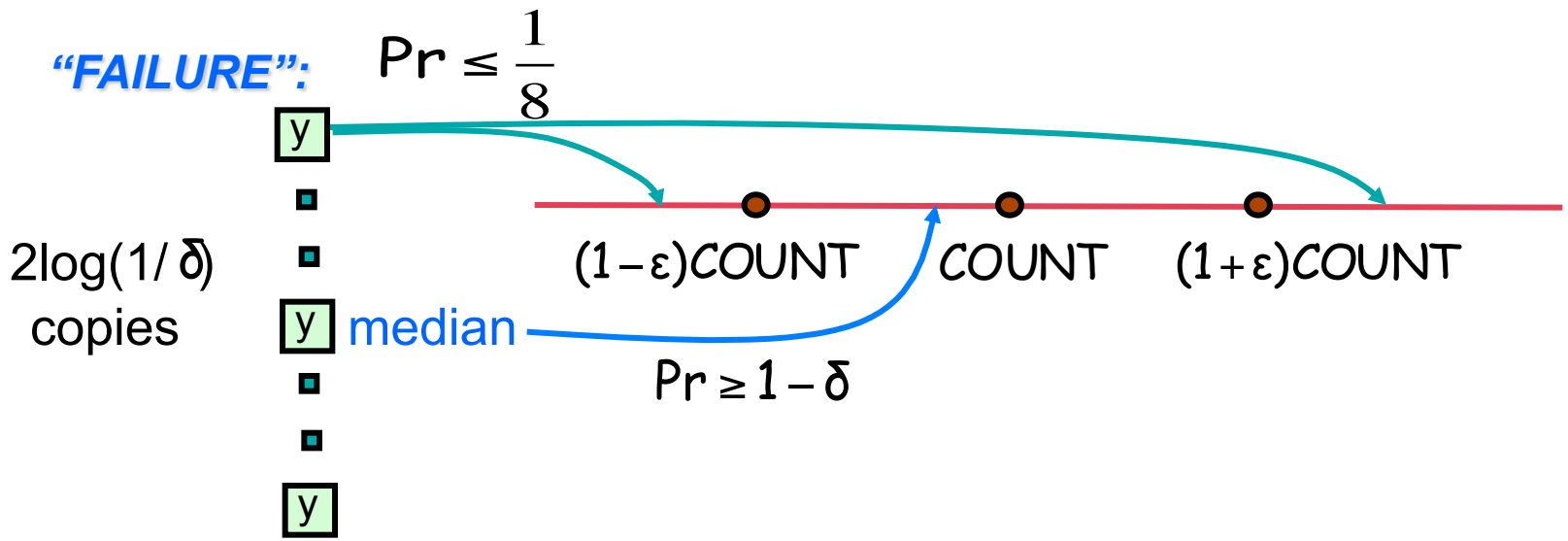
- By Chebyshev:

$$Var[Y] = \frac{Var[X]}{s} \leq \frac{\varepsilon^2 \|f_R\|_2^2 \|f_S\|_2^2}{8}$$

$$\Pr(|Y - COUNT| \geq \varepsilon \cdot \|f_R\|_2 \|f_S\|_2) \leq \frac{Var[Y]}{\varepsilon^2 \|f_R\|_2^2 \|f_S\|_2^2} \leq \frac{1}{8}$$

Boosting Confidence

- Boost confidence to $1-\delta$ by taking median of $2\log(1/\delta)$ independent copies of Y
- Each Y = ***Bernoulli Trial***



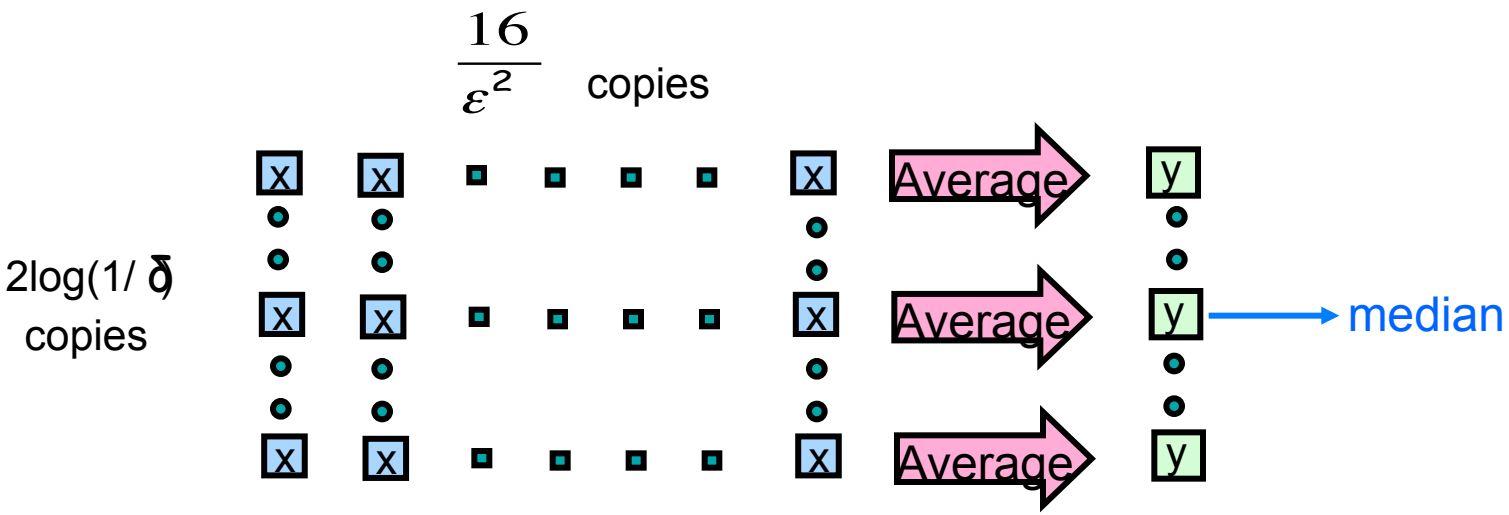
Pr[|median(Y)-COUNT| ≥ ε · COUNT]

= Pr[# failures in $2\log(1/\delta)$ trials >= $\log(1/\delta)$]

≤ δ (By Chernoff Bound)

Summary of Binary-Join AMS Sketching

- Step 1: Compute random variables: $X_R = \sum_i f_R(i)\xi_i$ and $X_S = \sum_i f_S(i)\xi_i$
- Step 2: Define $X = X_R X_S$
- Steps 3 & 4: Average independent copies of X; Return median of averages



- Main Theorem (AGMS99): Sketching approximates COUNT to within an error $\varepsilon \|f_R\|_2 \|f_S\|_2$ with probability $\geq 1-\delta$ using $O(\frac{\log(1/\delta)}{\varepsilon^2}\log N)$ bits
  - Remember: O(log N) bits for "seeding" the construction of each X

A Special Case: Self-join Size

- Estimate COUNT($R \bowtie_A R$) = $\sum_i f_R^2(i)$ *(original AMS paper)*
- Second (L2) moment of data distribution, Gini index of heterogeneity, measure of skew in the data

In this case, COUNT = SJ(R), so we get a RELATIVE ERROR (ε,δ)–estimate using space only

$$O\left(\frac{\log(1/\delta)\cdot \log N}{\varepsilon^2}\right)$$

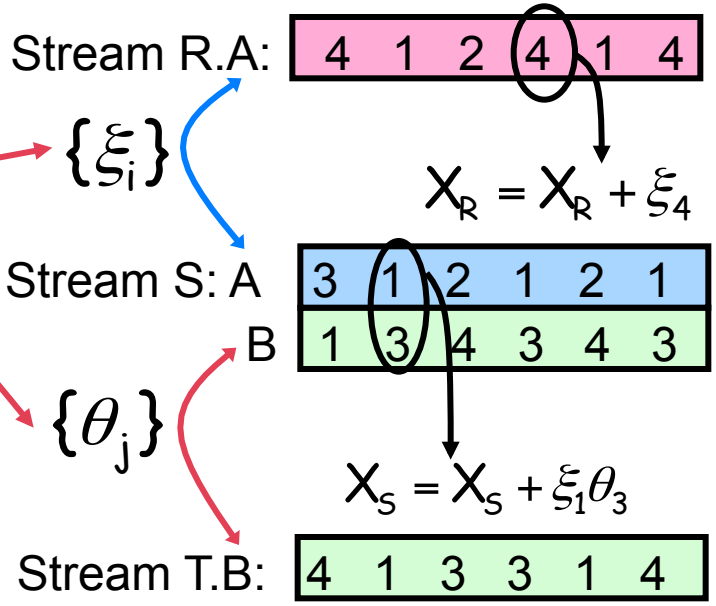
Best-case for AMS streaming join-size estimation

What's the worst case??

AMS Sketching for Multi-Join Aggregates [DGGR02]

- Problem: Compute answer for COUNT($R \bowtie_A S \bowtie_B T$) = $\sum_{i,j} f_R(i) f_S(i,j) f_T(j)$
- Sketch-based solution
  - Compute random variables X_R, X_S and X_T

Independent families of {-1,+1} random variables



$$X_R = \sum_i f_R(i)\xi_i = 2\xi_1 + \xi_2 + 3\xi_4$$

$$X_S = \sum_{i,j} f_S(i,j)\xi_i\theta_j = \xi_3\theta_1 + 3\xi_1\theta_3 + 2\xi_2\theta_4$$

$$X_T = \sum_j f_T(j)\theta_j = 2\theta_1 + 2\theta_3 + 2\theta_4$$

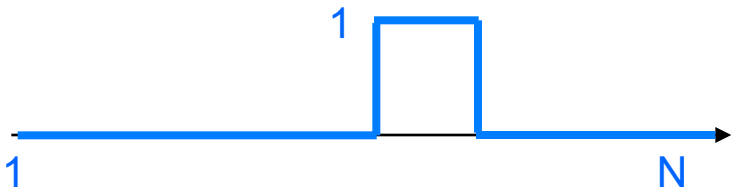
- Return $X = X_R X_S X_T$ (E[X]= COUNT($R \bowtie_A S \bowtie_B T$))

$$E[f_R(i)\cdot f_S(i',j)\cdot f_T(j')\xi_i\xi_{i'}\theta_j\theta_{j'}] = 0 \text{ if } i \neq i' \text{ or } j \neq j'$$

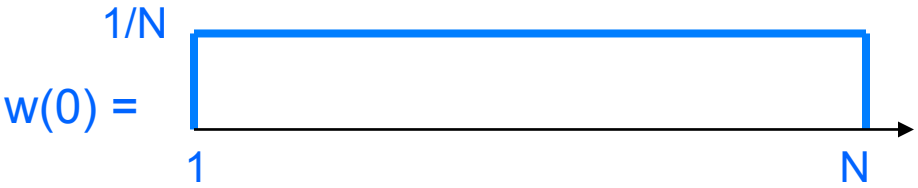
Other Applications of AMS Stream Sketching

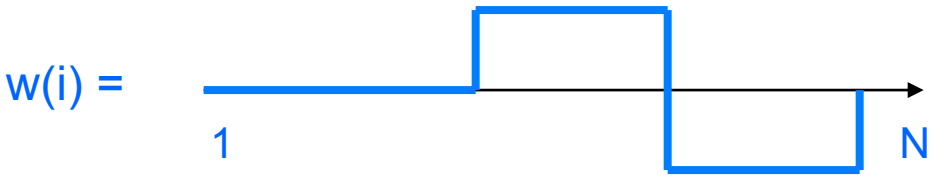
- Key Observation: $|R1 \bowtie R2| = \sum f_1(i) f_2(i) = < f_1, f_2 > =$ *inner product!*
- *General result:* Streaming (ε, δ) estimation of "large" inner products using AMS sketching

- Other streaming inner products of interest
  - Topk/Heavy-hitter entries [CCF02]
    - f[i] = < f, "unit_pulse at i" >
  - Range Sums…?



  - Large wavelet coefficients [GKMS01,CGS06]
    - Coeff(i) = < f, w(i) >, where w(i) = i-th wavelet basis vector

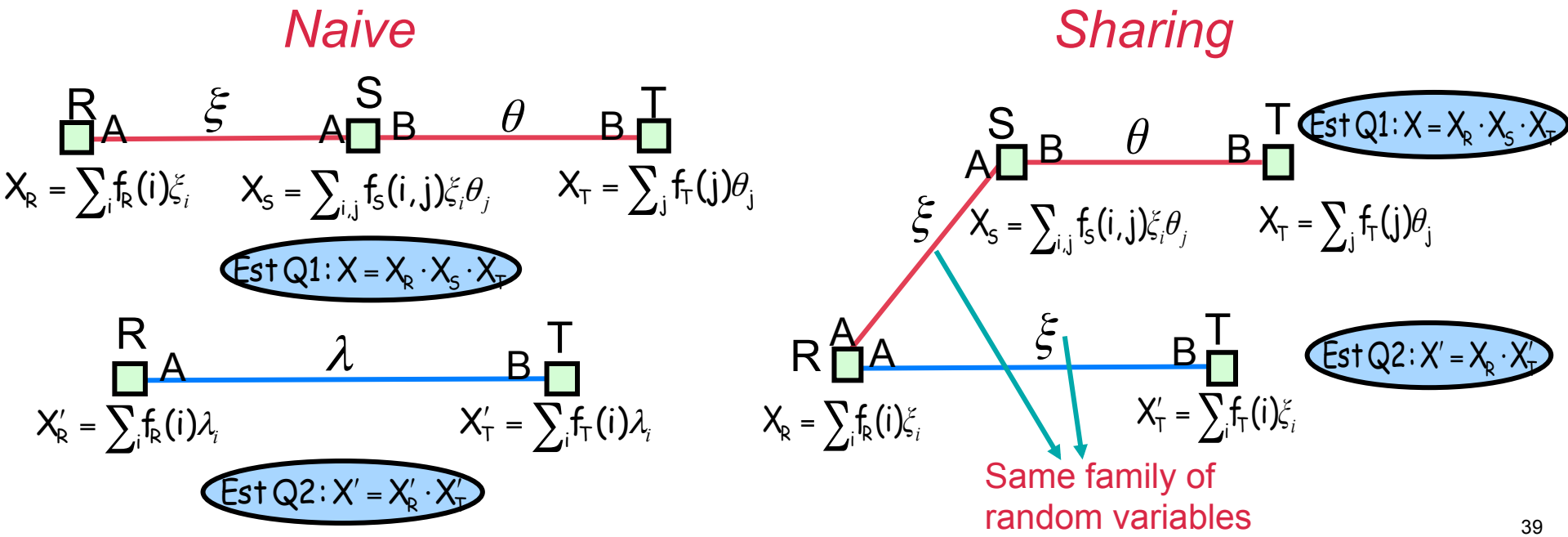




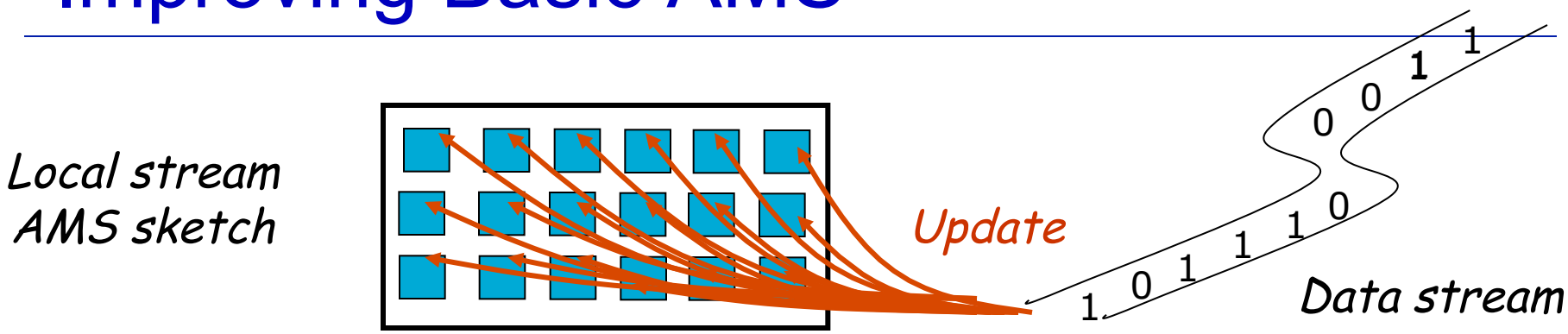
More Results on Sketches for Stream Joins

- Better accuracy using *"skimmed sketches"* [GGR04]
  - "Skim" dense items (i.e., large frequencies) from the AMS sketches
  - Use the "skimmed" sketch only for sparse element representation
  - Stronger worst-case guarantees, and much better in practice

- Sharing sketch space/computation among *multiple queries* [DGGR04]



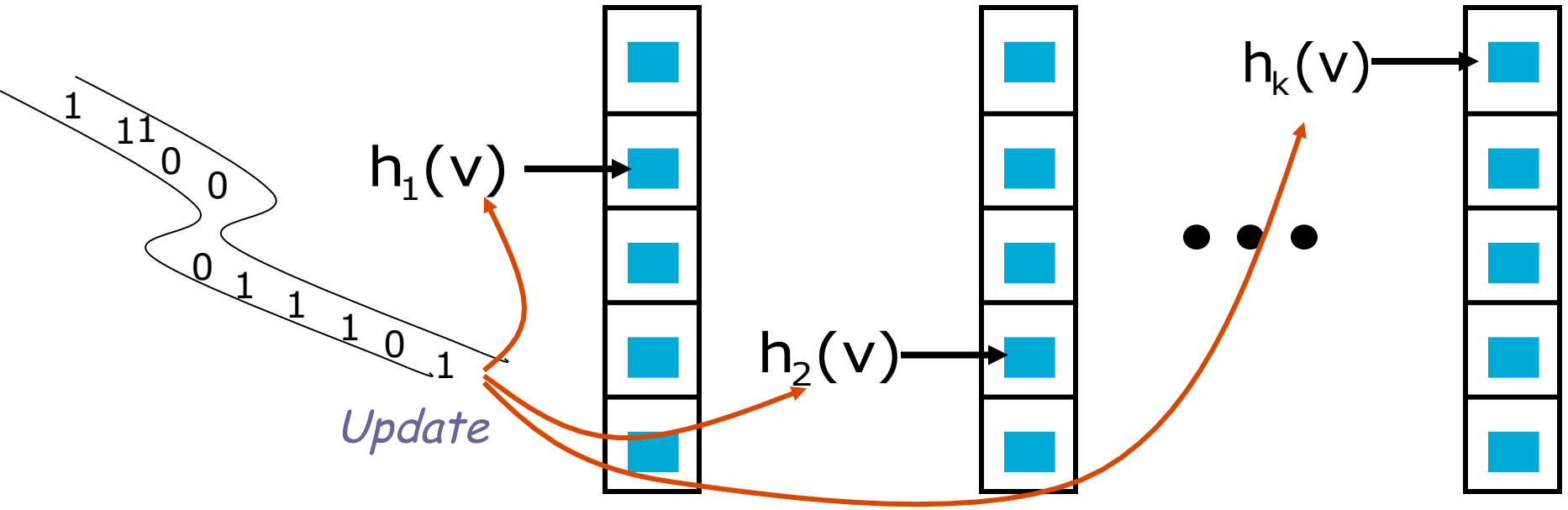
Improving Basic AMS



- Update time for basic AMS sketch is $\Omega(|sketch|)$

- *BUT…*
  - Sketches can get large – cannot afford to touch every counter for rapid-rate streams!
    - Complex queries, stringent error guarantees, …
  - Sketch size may not be the limiting factor (PCs with GBs of RAM)

Key Idea [MG+05'08]: The Fast AMS Sketch



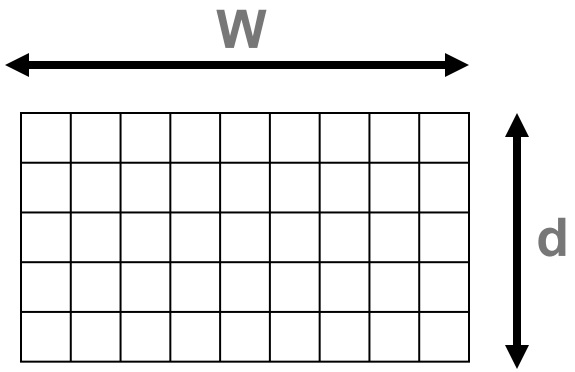
- *Fast AMS Sketch:* Organize the atomic AMS counters into hash-table buckets
  - Each update touches only a few counters (one per table)
  - Same space/accuracy tradeoff as basic AMS (in fact, better☺)
  - *BUT, guaranteed logarithmic update times (regardless of sketch size)!!*

Outline – Part I

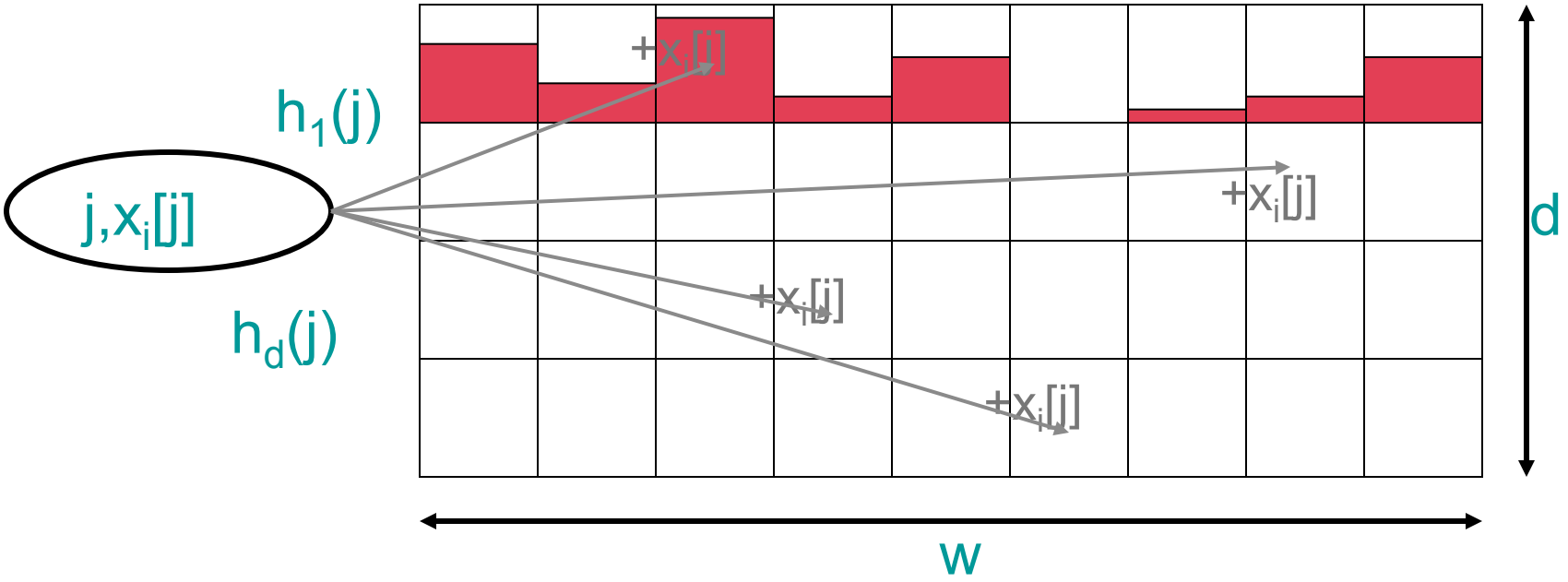
- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
  - Sampling
  - Sketching Multisets: Linear-Projection Sketches -- *AMS, CountMin*
    - *Applications:* Joins, Top-k, Range sums, Wavelets
  - Sketching Sets: FM Sketches
    - *Applications:* Distinct Values, Distinct Sampling
  - *(Time permitting)* Sliding Window model and Exponential Histograms (EHs)

The CountMin (CM) Sketch [Cormode,Muthu'05]

- Simple sketch idea, can be used for point/range queries, heavy hitters, quantiles, join size estimation, …
- Model input at each node as a vector f of dimension N, where N is large
- Creates a small summary as an array of w × d in size
- Use d hash functions to map vector entries to [1..w]



CM Sketch Structure



- Each entry in vector f is mapped to one bucket per row
- Merge two sketches by entry-wise summation
- Estimate f[j] by taking $\min_k$ sketch[k,h_k(j)]

[Cormode, Muthukrishnan '05]

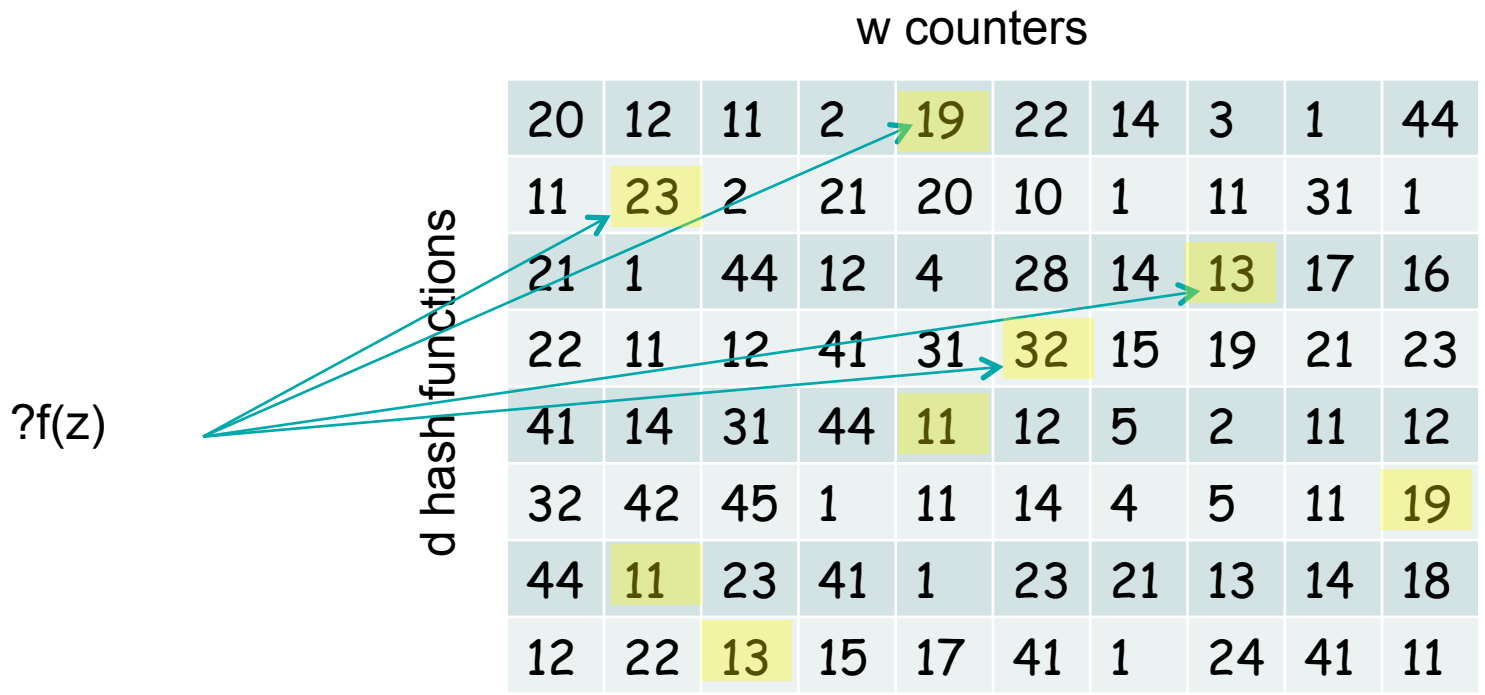
CM Sketch Structure - Insertion

Add ‘z’ 10 times (e.g. z is an ip address)

- Each entry in vector f is mapped to one bucket per row
- Estimate f[j] by taking $\min_k$ sketch[k,h_k(j)]

[Cormode, Muthukrishnan ’05]

CM Sketch Structure - Querying



- Each entry in vector f is mapped to one bucket per row
- Estimate f[j] by taking $\min_k$ sketch[k,h_k(j)]

[Cormode, Muthukrishnan '05]

CM Estimation error

- Answers can be overestimates due to hashing collisions
- E.g.

  ?f(z)

  - $hf_0(z)=hf_0(x)$, $hf_1(z)=hf_1(y)$, $hf_2(z)=\ldots$

w counters

20	12	11	2	19	22	14	3	1	44
11	23	2	21	20	10	1	11	31	1
21	1	44	12	4	28	14	13	17	16
22	11	12	41	31	32	15	19	21	23
41	14	31	44	11	12	5	2	11	12
32	42	45	1	11	14	4	5	11	19
44	11	23	41	1	23	21	13	14	18
12	22	13	15	17	41	1	24	41	11

d hash functions

- Biased error, depends on stream length
- Taking min reduces the error!
- Can this error be bounded?

[Cormode, Muthukrishnan '05]

CM Estimation error - Analysis

$$\hat{f}(x) = \min_j \text{sketch}[j, h_j(x)]$$

$\| f \|_1$: #elements in the sketch

For j th row:

$$\text{sketch}[j, h_j(x)] = f(x) + X_{j,x}, \text{ where } X_{j,x} = \sum_{z \neq x} f(z) \mid h_j(z) = h_j(x)$$

$$\Pr[h_j(z) = h_j(x)] = 1/w$$

By setting $w = e/\varepsilon$ we have $\Pr[h_j(z) = h_j(x)] = \varepsilon/e$

$$E[X_{j,x}] = \sum_{z \neq x} f(z) \times \Pr[h_j(z) = h_j(x)] = \sum_{z \neq x} f(z) \times \varepsilon/e \leq \varepsilon \| f \|_1 / e$$

[Cormode, Muthukrishnan '05]

CM Estimation error - Analysis

$$\hat{f}(x) = \min_j sketch[j, h_j(x)]$$

$\| f \|_1$: #elements in the sketch

For j th row, expected overestimate:

$$E[X_{j,x}] \leq \varepsilon \| f \|_1 / e$$

By Markov Inequality:

$$\Pr[X_{j,x} > \varepsilon \| f \|_1] \leq \frac{E[X_{j,x}]}{\varepsilon \| f \|_1} \leq 1/e$$

And, since the estimate $\hat{f}(x)$ is the min over d independent rows

(set $d = \ln 1/\delta$)

$$\Pr[\hat{f}(x) - f(x) > \varepsilon \| f \|_1] \leq (1/e)^d = \delta$$

[Cormode, Muthukrishnan '05]

CM Estimation error

- Answers can be overestimates due to hashing collisions
- E.g.

?f(z)

  - $hf_0(z)=hf_0(x)$, $hf_1(z)=hf_1(y)$, $hf_2(z)=\ldots$

- Taking min reduces the error!

w counters

20	12	11	2	19	22	14	3	1	44
11	23	2	21	20	10	1	11	31	1
21	1	44	12	4	28	14	13	17	16
22	11	12	41	31	32	15	19	21	23
41	14	31	44	11	12	5	2	11	12
32	42	45	1	11	14	4	5	11	19
44	11	23	41	1	23	21	13	14	18
12	22	13	15	17	41	1	24	41	11

d hash functions

- $w = e/\varepsilon, \quad d = \ln(1/\delta)$ → approximation error on point queries less than $\varepsilon||f||_1$, with probability at least $1-\delta$
- Size=w * d = $O(1/\varepsilon \log 1/\delta)$

[Cormode, Muthukrishnan '05]

CM-sketch - Inner product size

- Inner product (join size) queries:
  - Consider two streams a and b

$$f_a \otimes f_b = \sum_x f_a(x) * f_b(x)$$

  - Estimation using Count-min sketches

$$f_a \hat{\otimes} f_b = \min_j \sum_{i=1}^{w} sketch_a[j,i] \times sketch_b[j,i]$$

  - Accuracy guarantees

$$Pr[f_a \hat{\otimes} f_b - f_a \otimes f_b > \varepsilon \| f_a \|_1 \| f_b \|_1] \leq \delta$$

[Cormode, Muthukrishnan '05]

CM Sketch Summary

- Inaccuracies due to hashing collisions
  - Add more rows (increase d)
  - Increase counters per row (increase w)
- CM sketch guarantees approximation error on point queries less than $\varepsilon||f||_1$, with probability at least $1-\delta$
  - $w = e/\varepsilon, \quad d = \log\frac{1}{\delta}$
  - Size=w * d = $O(1/\varepsilon \log 1/\delta)$
  - Similar guarantees for range queries, quantiles, join size
  - Biased error, depends on stream length (L1 norm)!
- Sketch merging: counter-wise summation
- *Food for thought:* Compare CM and AMS…???

Outline – Part I

- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
  - Sampling
  - Sketching Multisets: Linear-Projection Sketches -- *AMS, CountMin*
    - *Applications:* Joins, Top-k, Range sums, Wavelets
  - Sketching Sets: FM Sketches
    - *Applications:* Distinct Values, Distinct Sampling
  - *(Time permitting)* Sliding Window model and Exponential Histograms (EHs)

Distinct Value Estimation

- Problem: Find the *number of distinct values* in a stream of values with domain [0,...,N-1]
  - Zeroth frequency moment F_0, L0 (Hamming) stream norm
  - Statistics: number of *species or classes* in a population
  - Important for query optimizers
  - *Network monitoring:* distinct destination IP addresses, source/destination pairs, requested URLs, etc.

- Example (N=64)

Data stream: 3 0 5 3 0 1 7 5 1 0 3 7

Number of distinct values: 5

- Hard problem for random sampling! [CCMN00]
  - Must sample almost the entire table to guarantee the estimate is within a factor of 10 with probability > 1/2, regardless of the estimator used!

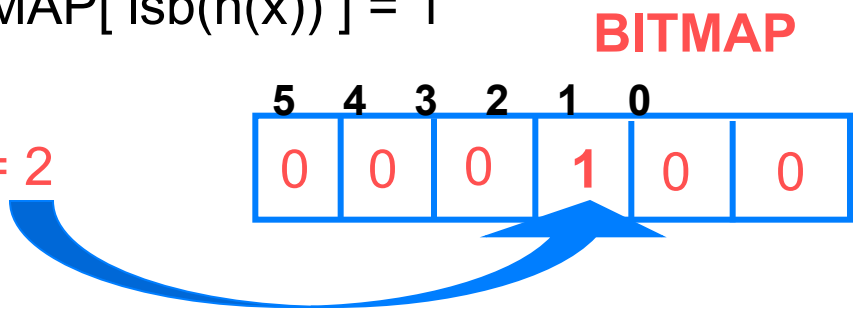
FM Sketches for Distinct Value Estimation [FM85]

- Assume a hash function h(x) that maps incoming values x in [0,…, N-1] *uniformly* across [0,…, 2^L-1], where L = O(logN)

- Let lsb(y) denote the position of the least-significant 1 bit in the binary representation of y
  - A value x is mapped to lsb(h(x))

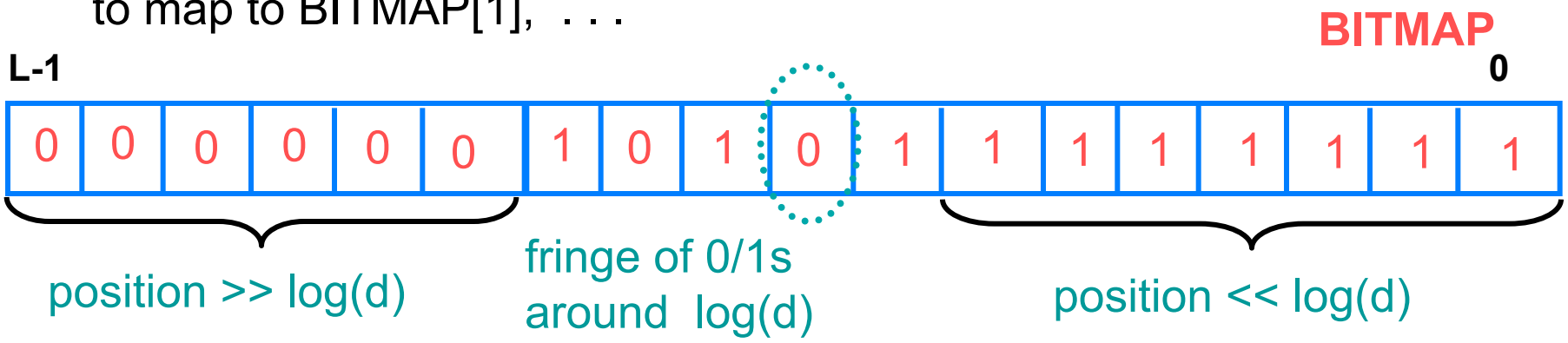
- Maintain *Hash Sketch* = BITMAP array of L bits, initialized to 0
  - For each incoming value x, set BITMAP[lsb(h(x))] = 1

x = 5 → h(x) = 101100 → lsb(h(x)) = 2



FM Sketches for Distinct Value Estimation [FM85]

- By uniformity through h(x): Prob[BITMAP[k]=1] = Prob[10^k] = $\frac{1}{2^{k+1}}$
  - Assuming d distinct values: expect d/2 to map to BITMAP[0] , d/4 to map to BITMAP[1], . . .



- Let R = position of rightmost zero in BITMAP
  - Use as indicator of log(d)
- [FM85] prove that E[R] = $\log(\phi d)$, where $\phi = .7735$
  - Estimate d = $2^R / \phi$
  - Average several iid instances (different hash functions) to reduce estimator variance

FM Sketches for Distinct Value Estimation

- [FM85] assume "ideal" hash functions h(x) (N-wise independence)
  - [AMS96]: pairwise independence is sufficient
    - h(x) = $(a \cdot x + b) \bmod N$, where a, b are random binary vectors in [0,…,2^L-1]
  - Small-space (ε, δ) estimates for distinct values proposed based on FM ideas
- *Delete-Proof:* Just use counters instead of bits in the sketch locations
  - +1 for inserts, -1 for deletes
- *Composable:* Component-wise OR/add distributed sketches together
  - Estimate |S1 ∪ S2 ∪… ∪ Sk| = *set-union cardinality*

Generalization: Distinct Values Queries

- SELECT COUNT(DISTINCT target-attr)
- FROM relation
- WHERE predicate

Template

- SELECT COUNT(DISTINCT o_custkey)
- FROM orders
- WHERE o_orderdate >= '2016-01-01'

TPC-H example

  - "How many distinct customers have placed orders this year?"
  - Predicate not necessarily only on the DISTINCT target attribute

- *Approximate answers with error guarantees over a stream of tuples?*

Distinct Sampling [Gib01]

Key Ideas

- Use FM-like technique to collect a specially-tailored sample over the *distinct values in the stream*
  - ***Use hash function mapping to sample values from the data domain!!***
  - Uniform random sample of the distinct values
  - Very different from traditional random sample: each distinct value is chosen uniformly regardless of its frequency
  - DISTINCT query answers: simply scale up sample answer by sampling rate

- To handle additional predicates
  - *Reservoir sampling* of tuples for each distinct value in the sample
  - Use reservoir sample to evaluate predicates

Building a Distinct Sample [Gib01]

- Use FM-like hash function h() for each streaming value x
  - Prob[h(x) = k] = $\frac{1}{2^{k+1}}$
- *Key Invariant:* *"All values with h(x) >= level (and only these) are in the distinct sample"*

DistinctSampling(B , r)

// B = space bound, r = tuple-reservoir size for each distinct value

level = 0; S = ϕ

for each new tuple t do

    let x = value of DISTINCT target attribute in t

    if h(x) >= level then // x belongs in the distinct sample

        use t to update the reservoir sample of tuples for x

    if |S| >= B then // out of space

        evict from S all tuples with h(target-attribute-value) = level

        set level = level + 1

Using the Distinct Sample [Gib01]

- If level = l for our sample, then we have selected all distinct values x such that h(x) >= l
  - Prob[h(x) >= l] = $\frac{1}{2^l}$
  - By h()'s randomizing properties, we have uniformly sampled a 2^{-l} fraction of the distinct values in our stream

Our sampling rate!

- *Query Answering:* Run distinct-values query on the distinct sample and scale the result up by 2^l

- *Distinct-value estimation:* Guarantee ε relative error with probability $1 - \delta$ using $O(\log(1/\delta)/\varepsilon^2)$ space
  - For q% selectivity predicates the space goes up inversely with q

- *Experimental results:* 0-10% error vs. 50-250% error for previous best approaches, using 0.2% to 10% synopses

Distinct Sampling Example

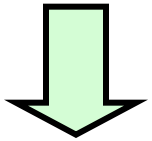
- B=3, N=8 (r = 0 to simplify example)

Data stream: 3 0 5 3 0 1 7 5 1 0 3 7

hash:	0	1	3	5	7
	0	1	0	1	0

Data stream: 1 7 5 1 0 3 7

S={3,0,5}, level = 0



S={1,5}, level = 1

- Computed value: 4

Outline – Part I

- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
  - Sampling
  - Sketching Multisets: Linear-Projection Sketches -- *AMS, CountMin*
    - *Applications:* Joins, Top-k, Range sums, Wavelets
  - Sketching Sets: FM Sketches
    - *Applications:* Distinct Values, Distinct Sampling
  - *(Time permitting)* Sliding Window model and Exponential Histograms (EHs)

Sliding Window Streaming Model

- Model
  - At time t, a data record arrives
  - The record "expires" at time t+N (N is the window length)
  - Time-based or Arrival-based Window
  - Window size is the limiting factor for memory, CPU, …
- When is it useful?
  - Make decisions based on "recently observed" data
  - Stock data
  - Sensor networks

Time in Data Stream Models

Tuples arrive $X_1, X_2, X_3, \ldots, X_t, \ldots$

- Function f(X,t,NOW)
  - Input at time t: $f(X_1,1,t), f(X_2,2,t). f(X_3,3,t), \ldots, f(X_t,t,t)$
  - Input at time t+1: $f(X_1,1,t+1), f(X_2,2,t+). f(X_3,3,t+1), \ldots, f(X_{t+1},t+1,t+1)$
- Full history (Landmark): f == identity
- Partial history: Decay
  - Exponential decay: $f(X,t, NOW) = 2^{-(NOW-t)}*X$
    - Input at time t: $2^{-(t-1)}*X_1, 2^{-(t-2)}*X_{2,}, \ldots, \frac{1}{2} * X_{t-1}, X_t$
    - Input at time t+1: $2^{-t}*X_1, 2^{-(t-1)}*X_{2,}, \ldots, 1/4 * X_{t-1}, \frac{1}{2} *X_{t,} X_{t+1}$
  - Sliding window (special type of decay):
    - f(X,t,NOW) = X if NOW-t < N
    - f(X,t,NOW) = 0, otherwise
    - Input at time t: $X_1, X_2, X_3, \ldots, X_t$
    - Input at time t+1: $X_2, X_3, \ldots, X_{t,} X_{t+1,}$

Statistics Over Sliding Windows

- Bitstream: Count the number of ones [DGIM02]
  - Exact solution: **$\Theta(N)$ bits**
  - Exponential Histograms (EHs):
    - $1 + \varepsilon$ approximation (relative error!)
    - Space: $O(\log^2 N/\varepsilon)$ bits
    - Time: $O(\log N)$ worst case, $O(1)$ amortized per record
  - Lower Bound:
    - Space: $\Omega(\log^2 N/\varepsilon)$ bits

Approach: Temporal Histograms

Example: … 01101010011111110110 0101 …

Equi-width histogram:

… 0110 1010 0111 1111 0110 0101 …

- Issues:
  - Error is in the last (leftmost) bucket.
  - Bucket counts (left to right): $C_m, C_{m-1}, \ldots, C_2, C_1$
  - Absolute error <= $C_m/2$.
  - Answer >= $C_{m-1}+\ldots+C_2+C_1+1$.
  - Relative error <= $C_m/2(C_{m-1}+\ldots+C_2+C_1+1)$.
  - Maintain: $C_m/2(C_{m-1}+\ldots+C_2+C_1+1) <= \varepsilon\ (=1/k)$.

Naïve: Equi-Width Histograms

- Goal: Maintain $C_m/2 <= \varepsilon (C_{m-1}+...+C_2+C_1+1)$

Problem case:

... 0110 1010 0111 1111 0110 1111 0000 0000 0000 0000 ...

- Note:
  - Every Bucket will be the last bucket sometime!
  - New records may be all zeros ➔
    For **every** bucket i, require $C_i/2 <= \varepsilon (C_{i-1}+...+C_2+C_1+1)$

Exponential Histograms (EHs)

- Data structure invariant:
  - Bucket sizes are non-decreasing powers of 2
  - For every bucket size other than that of the last bucket, there are at least k/2 and at most k/2+1 buckets of that size
  - Example: k=4: (8,4,4,4,2,2,2,1,1..)
- Invariant implies:
  - Assume $C_i=2^j$, then
    - $C_{i-1}+...+C_2+C_1+1 >= k/2*(\Sigma(1+2+4+..+2^{j-1})) >= k*2^j/2 >= k/2*C_i$
  - Setting $k = 1/\varepsilon$ implies the required error guarantee!

Space Complexity

- Number of buckets m:
  - m <= [# of buckets of size j]*[# of different bucket sizes]
    <= (k/2 +1) * ((log(2N/k)+1) = O(k* log(N))
- Each bucket requires O(log N) bits.
- Total memory:
  $O(k \log^2 N) = O(1/\varepsilon * \log^2 N)$ bits
- Invariant (with k = $1/\varepsilon$) maintains error guarantee!

EH Maintenance Algorithm

Data structures:

- For each bucket: timestamp of most recent 1, size = #1's in bucket
- LAST: size of the last bucket
- TOTAL: Total size of the buckets

New element arrives at time t

- If last bucket expired, update LAST and TOTAL
- If (element == 1)
  Create new bucket with size 1; update TOTAL
- Merge buckets if there are more than k/2+2 buckets of the same size
- Update LAST if changed

Anytime estimate: TOTAL – (LAST/2)

Example Run

- If last bucket expired, update LAST and TOTAL
- If (element == 1)
  Create new bucket with size 1; update TOTAL
- Merge two oldest buckets if there are more than k/2+2 buckets of the same size
- Update LAST if changed

Example (k=2):

32,16,8,8,4,4,2,1,1

32,16,8,8,4,4,2,2,1

32,16,8,8,4,4,2,2,1,1

32,16,16,8,4,2,1

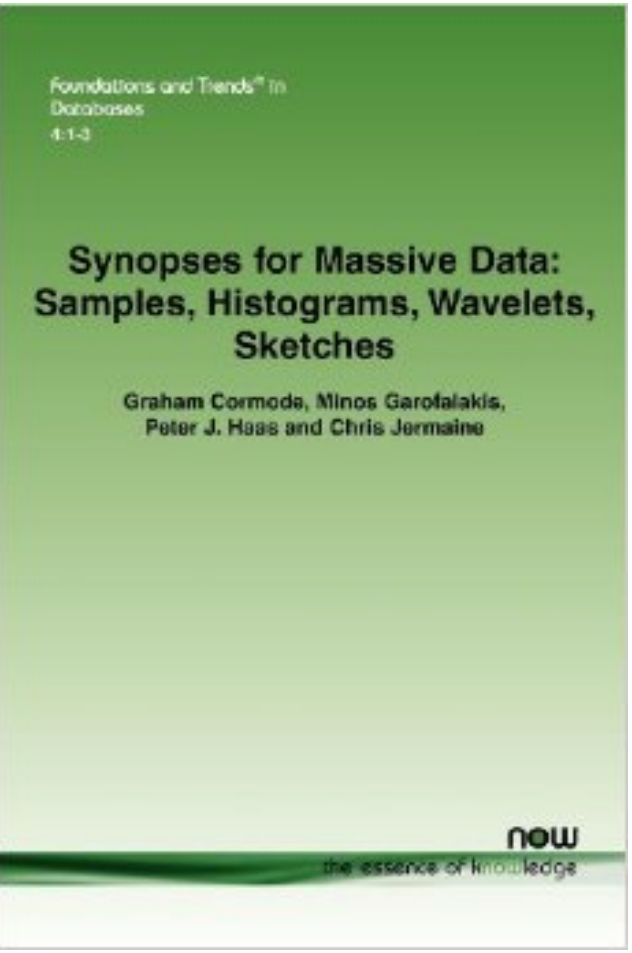
The Power of EHs

- Counter for N items = O(logN) space
- EH = ε–approximate counter *over sliding window* of N items that requires $O(\log^2 N/\varepsilon)$ space
  - O(logN/ε) penalty for (approximate) sliding-window counting
- Can plugin EH-counters to counter-based streaming methods ➔ work in sliding-window model!!
  - Examples: histograms, CM-sketches, …
- *Complication:* counting is now ε–approximate
  - Account for that in analysis
  - See *ECM Sketches* [PGD VLDB'12]

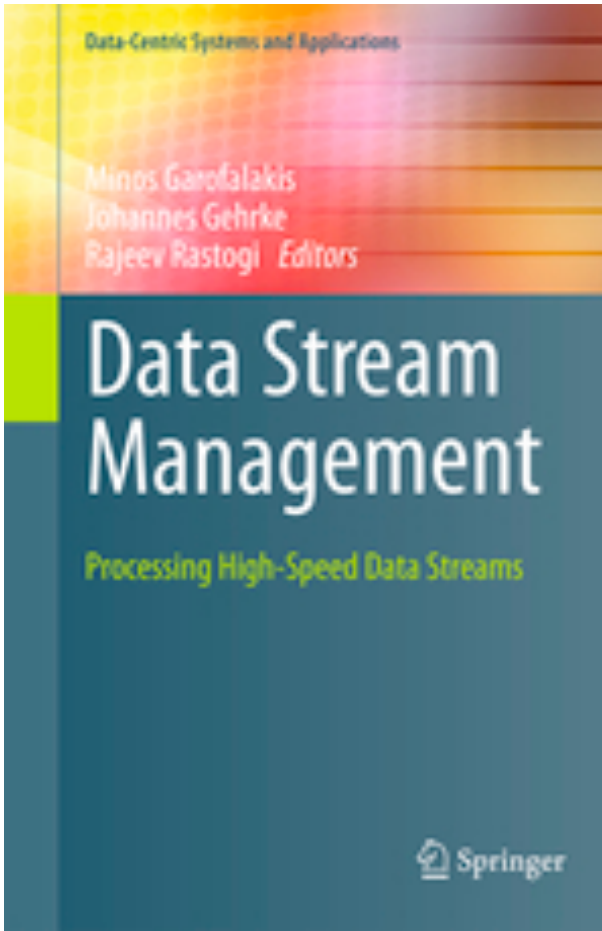
Conclusions – Part I

- Querying and finding patterns in massive streams is a real problem with many real-world applications
- Fundamentally rethink data-management issues under stringent constraints
  - Single-pass algorithms with limited memory resources
- A lot of progress in the last few years
  - Algorithms, system models & architectures
    - Academic prototypes: Aurora (Brandeis/Brown/MIT), Niagara (Wisconsin), STREAM (Stanford), Telegraph (Berkeley)
    - Several systems developed in-house for stream-processing, using synopses, etc.: GigaScope (AT&T), tools at Facebook, Twitter, …
    - Cluster platforms: Stream/Heron, Spark Streaming, Flink, …

Shameless plug...



300 pages on everything you ever wanted to know about data synopses

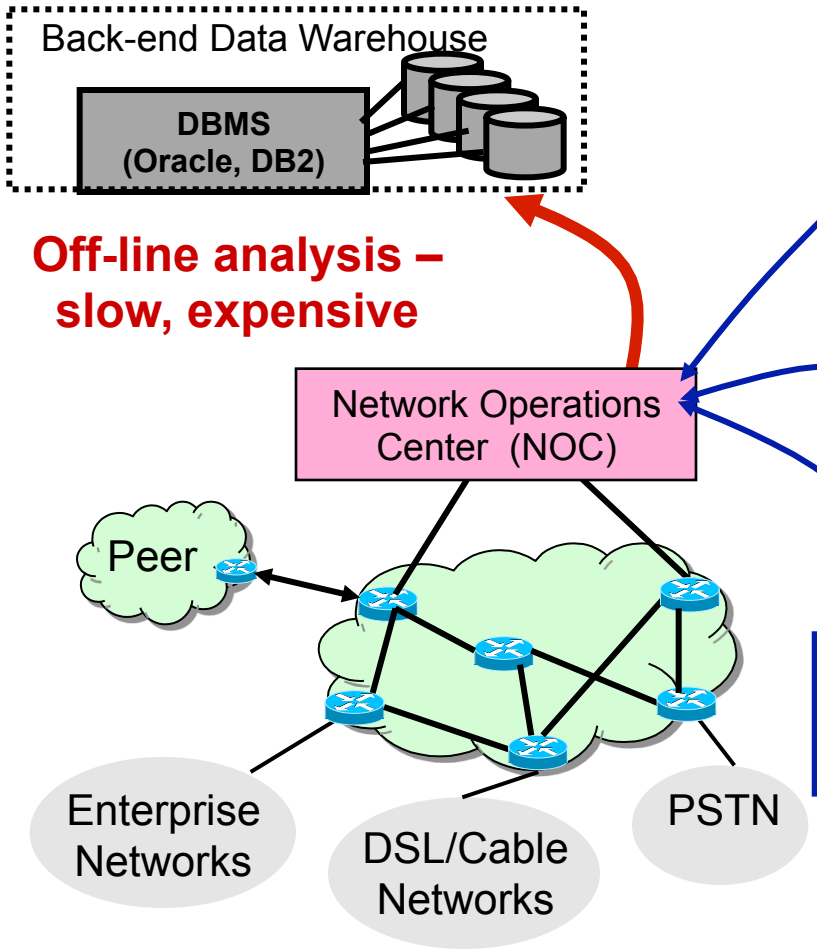


25 chapters by domain experts on algorithmic and systems aspects of data stream management

LARGE-SCALE ELASTIC ARCHITECTURE FOR DATA AS A SERVICE

PART II: Distributed Data Streaming

Network Monitoring Queries



What are the top (most frequent) 1000 (source, dest) pairs seen over the last 30 days?

How many distinct (source, dest) pairs have been seen by both R1 and R2 but not R3?

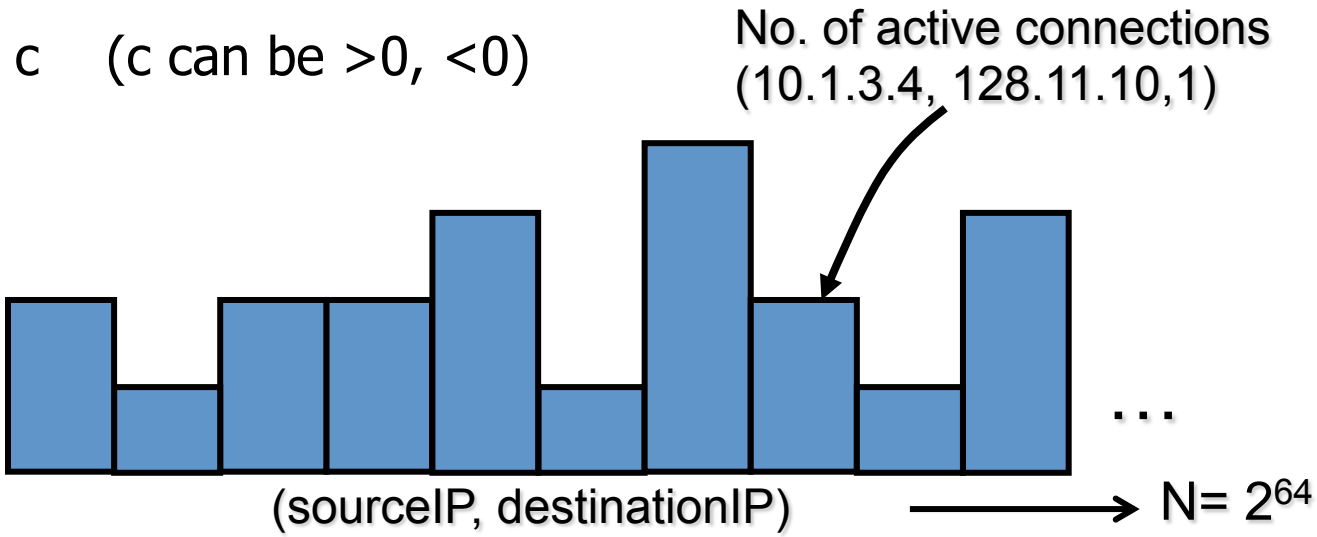
Set-Expression Query

```
SELECT COUNT (R1.source, R2.dest)
FROM  R1, R2
WHERE R1.dest = R2.source
```

SQL Join Query

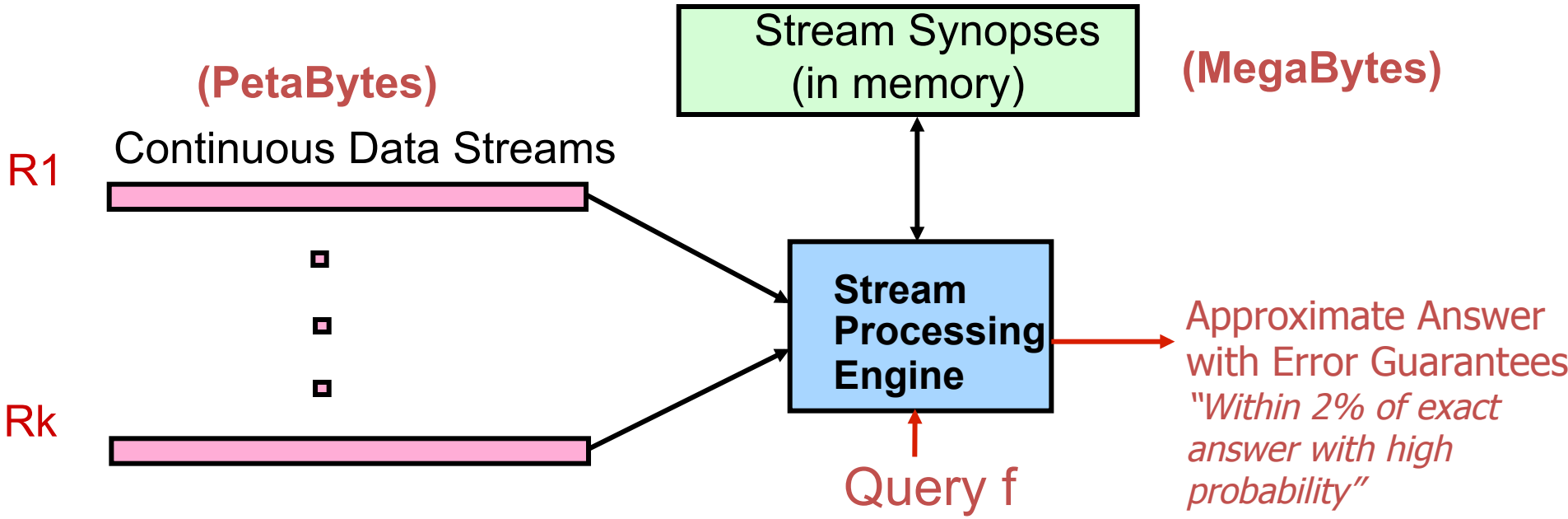
Model of a Relational Stream

- Relation "signal": *Large* array $v_S[1...N]$ with values $v_S[i]$ initially zero
  - Frequency-distribution array of **S**
  - Multi-dimensional arrays as well (e.g., row-major)
- Relation implicitly rendered via a *stream of updates*
  - Update <x, c> implying
    - $v_S[x] := v_S[x] + c$ (c can be >0, <0)



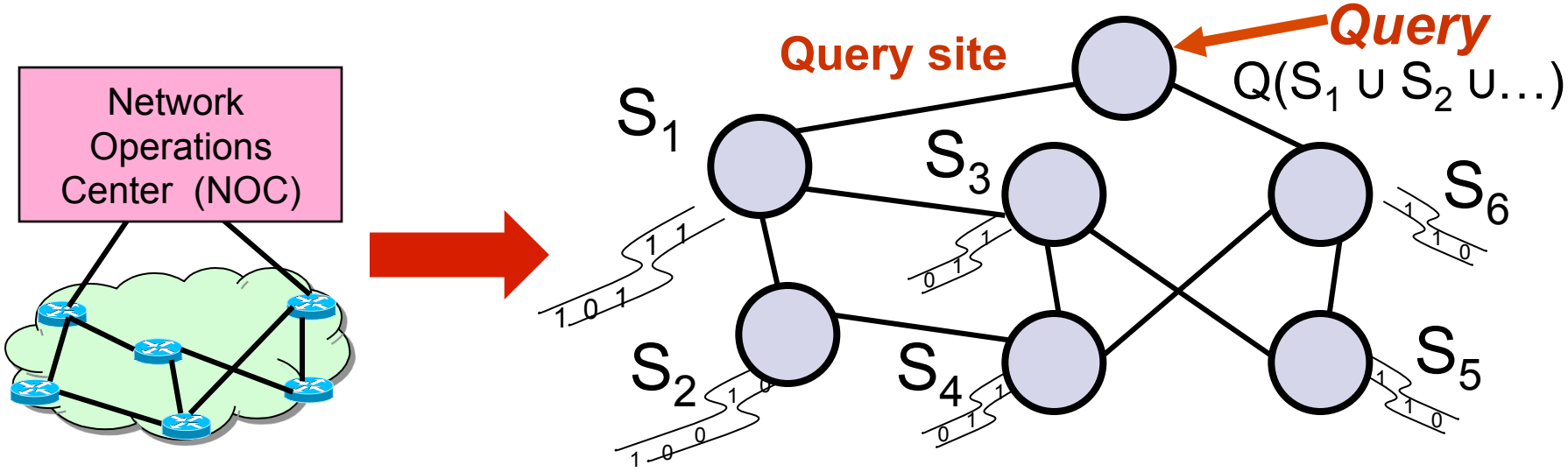
- *Goal:* Compute queries (functions) on such dynamic vectors in "small" space and time (<< N)

Stream Processing Model



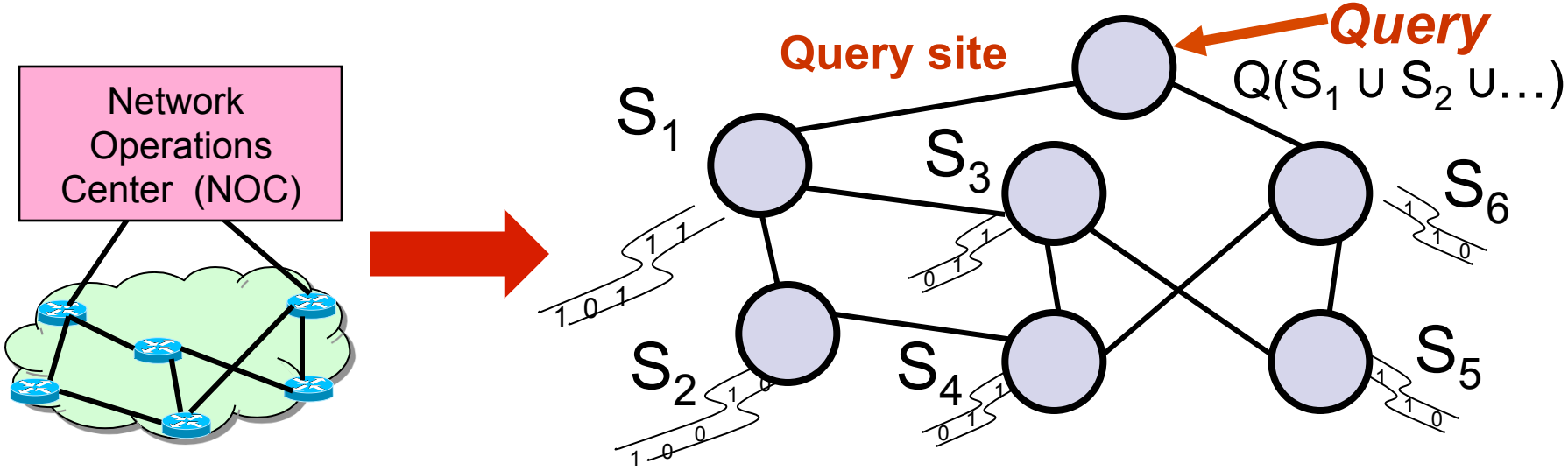
- Approximate answers often suffice, e.g., trends, anomalies
- Requirements for stream synopses
  - *Single Pass:* Each record examined at most once, in arrival order
  - *Small Space:* Log or polylog in data stream size
  - *Small Time:* Per-record processing time must be low
  - Also: *Delete-proof, Composable*, …

Distributed Streams Model



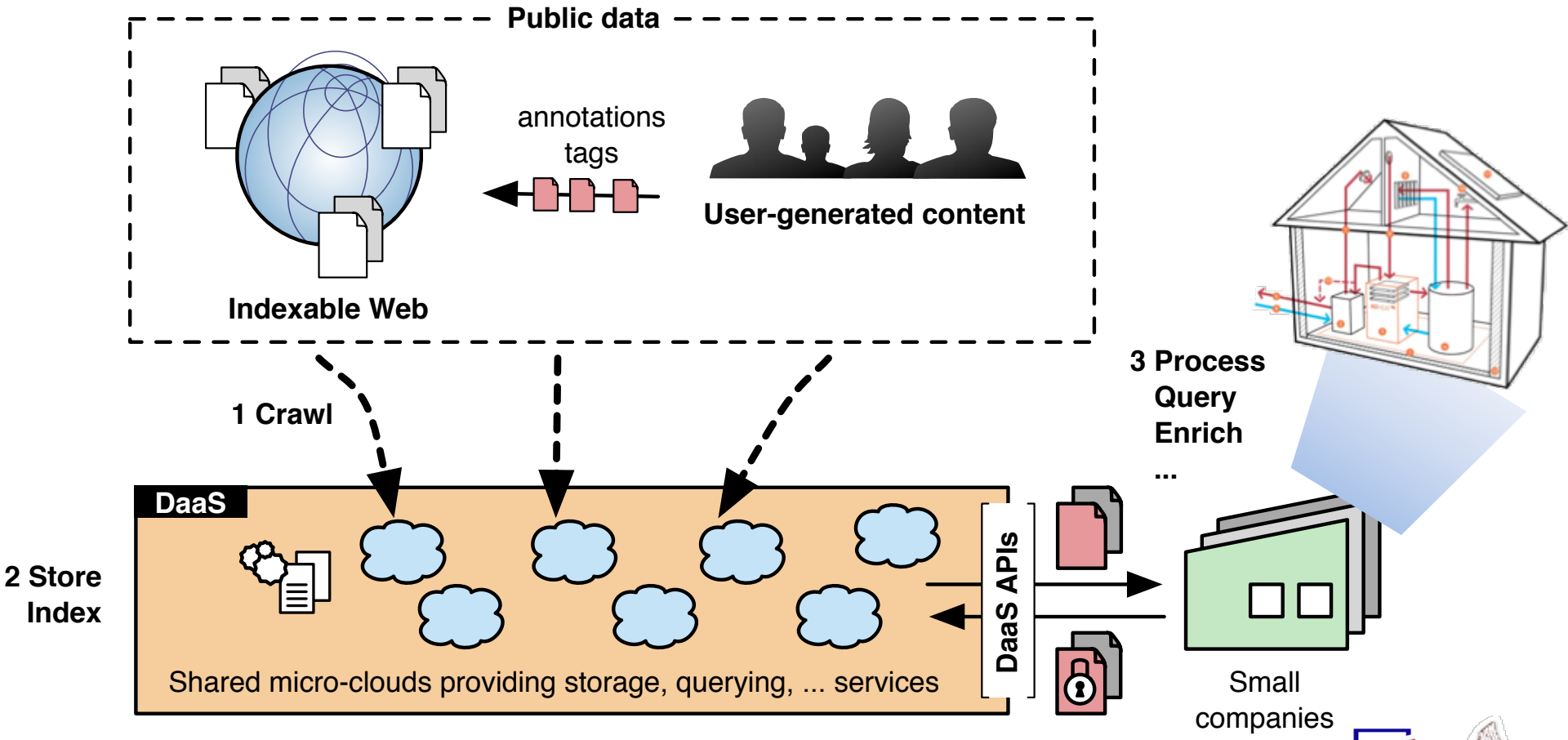
- Large-scale querying/monitoring: *Inherently distributed!*
  - Streams physically distributed across remote sites
    E.g., stream of UDP packets through subset of edge routers
- *Challenge is "holistic" querying/monitoring*
  - Queries over the *union of distributed streams* $Q(S_1 \cup S_2 \cup \ldots)$
  - Streaming data is spread throughout the network

Distributed Streams Model

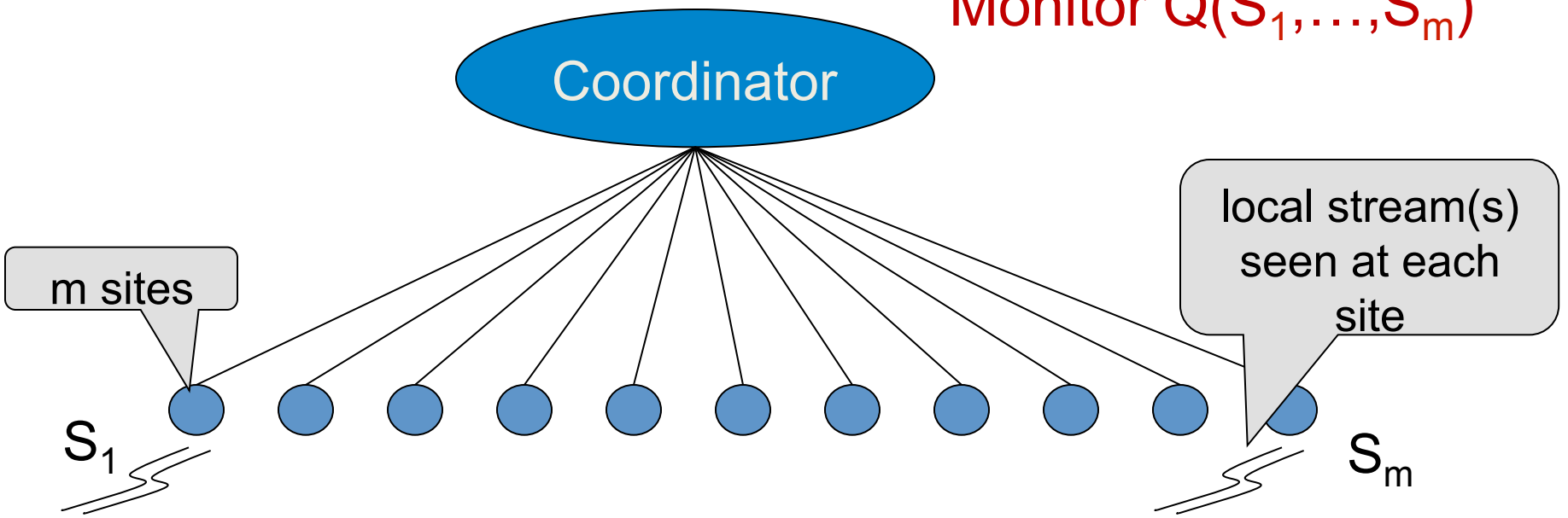


- Need timely, accurate, and efficient query answers
- Additional complexity over centralized data streaming!
- Need space/time- *and communication-efficient* solutions
  - Minimize network overhead
  - Maximize network lifetime (e.g., sensor battery life)
  - Cannot afford to “centralize” all streaming data

Example: LEADS Elastic μClouds Architecture *(http://leads-project.eu)*



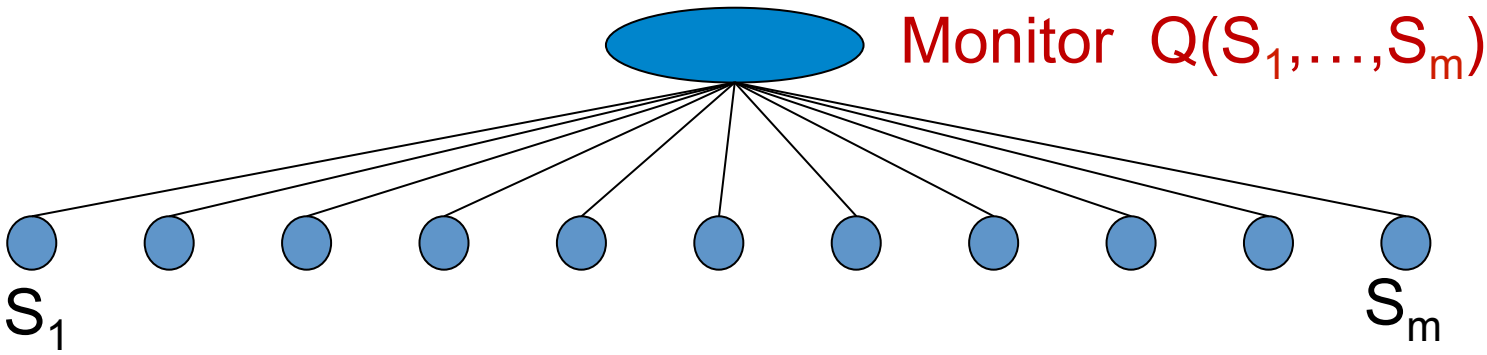
Velocity & Distribution: *Continuous Distributed Streaming*



- Other structures possible (e.g., hierarchical, P2P)
- Goal: *Continuously track* (global) query over streams at coordinator
  - Using small space, time, and ***communication***
  - Example queries:
    - Join aggregates, Variance, Entropy, Information Gain, …

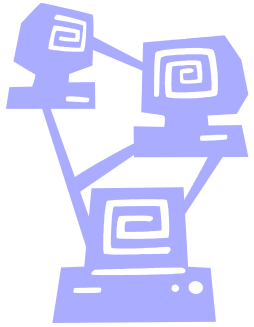
Continuous Distributed Streaming

- But… local site streams continuously change! New readings/data…
- Classes of monitoring problems
  - **Threshold Crossing**: Identify when $Q(S)>\tau$
  - **Approximate Tracking**: $Q(S)$ within **guaranteed accuracy bound** θ
    - Tradeoff *accuracy and communication / processing cost*
- Naïve solutions must *continuously* centralize all data
  - Enormous communication overhead!
- Instead, *in-situ* stream processing using *local constraints* !



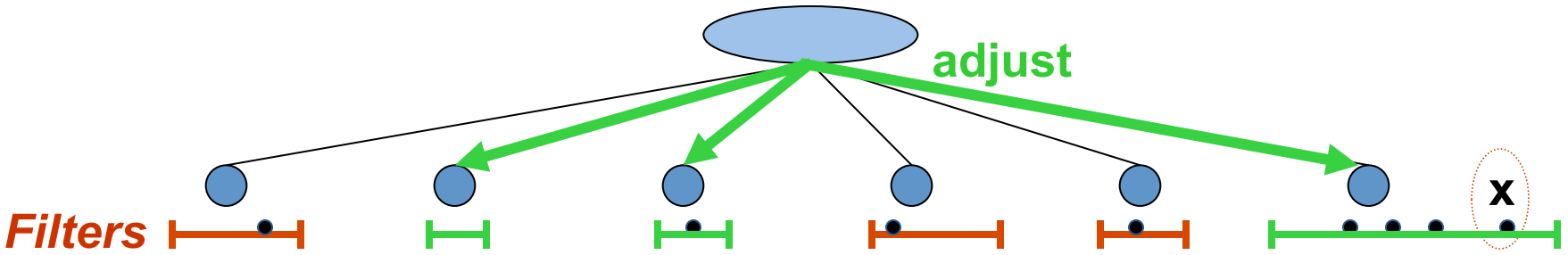
How about Periodic Polling?

- Sometimes periodic polling suffices for simple tasks
  - E.g., SNMP polls total traffic at coarse granularity
- Still need to deal with holistic nature of aggregates
- Must balance polling frequency against communication
  - Very frequent polling causes high communication, excess battery use in sensor networks
  - Infrequent polling means delays in observing events
- Need techniques to reduce communication while guaranteeing rapid response to events



Communication-Efficient Monitoring

- **Key Idea:** *"Push-based" in-situ processing*
  - *Local filters* installed at sites process local streaming updates
    - Offer bounds on local-stream behavior (at coordinator)
  - *"Push"* information to coordinator only when filter is violated
  - ***"Safe"!*** Coordinator sets/adjusts local filters to guarantee accuracy



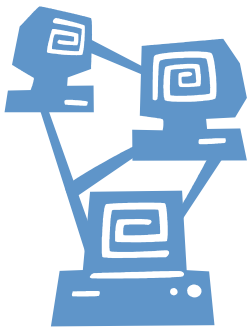
Local Slack/Filter Allocation

- Problem is relatively straightforward for tracking ***linear*** functions/queries
  - By *additivity*, simply divide the total slack across sites
  - Several early papers on tracking distributed counts, sums, top-k frequencies, etc.

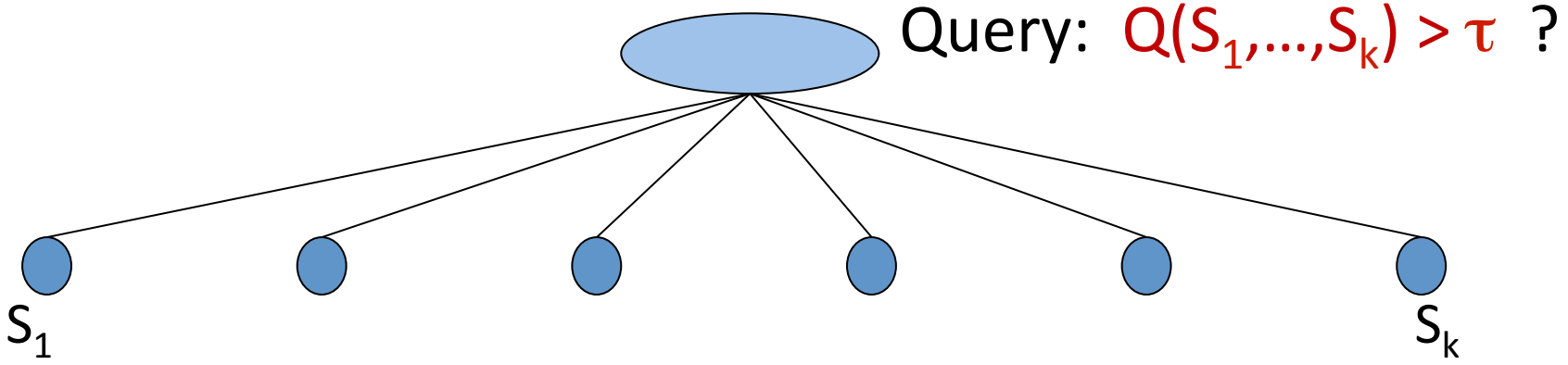
- Things get much more interesting for ***non-linear*** functions…!

Outline – Part II

- Introduction: Continuous Distributed Streaming
- The Geometric Method (GM)
- Recent Work: GM + Sketches, GM + Prediction Models
- Towards Convex Safe Zones (SZs)
- Future Directions & Conclusions



Monitoring General, Non-linear Functions



- For general, non-linear Q(), problem is a lot harder!
  - E.g., information gain over global data distribution
- Non-trivial to decompose the global threshold into "safe" local site constraints
  - E.g., consider $N=(N_1+N_2)/2$ and $Q(N) = 6N - N^2 > 1$
    Tricky to break into thresholds for $Q(N_1)$ and $Q(N_2)$

The Geometric Method

- A general purpose geometric approach [SKS SIGMOD'06]
  - Monitor **function domain** rather than the range of values!

- Each site tracks a local statistics *vector* v_i (e.g., data distribution)

- Global condition is $Q(v) > \tau$, where $v = \sum_i \lambda_i v_i$ $(\sum_i \lambda_i = 1)$
  - E.g., v = *average* of local statistics vectors

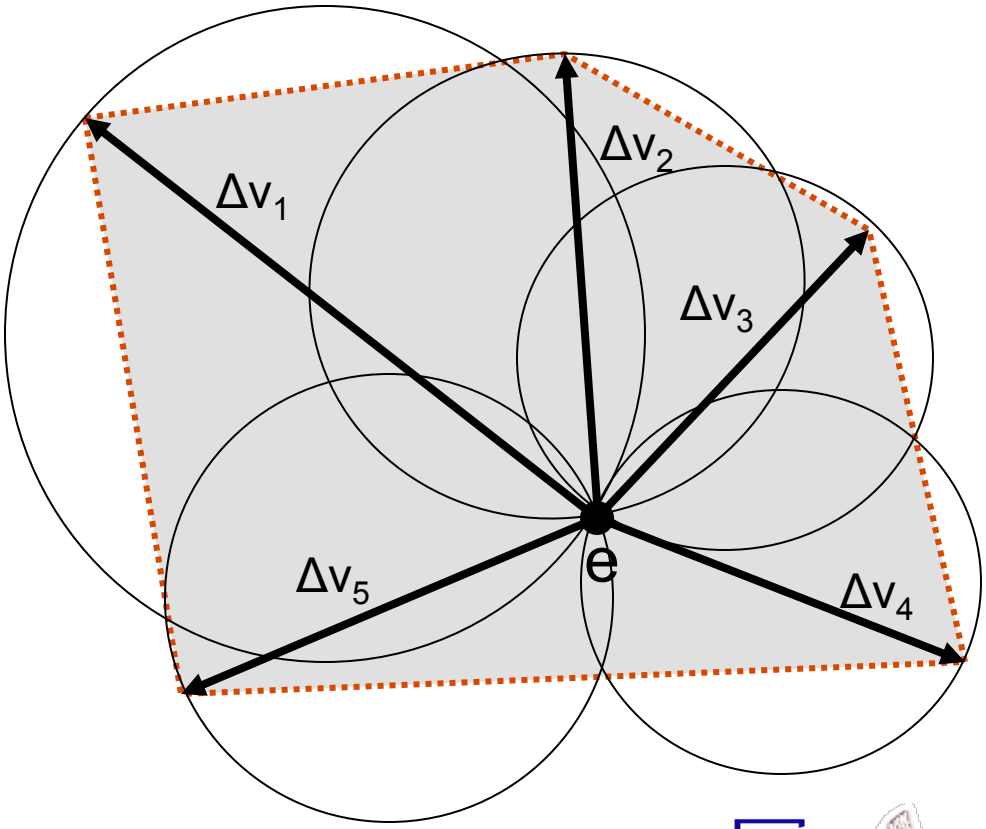
- All sites share estimate $e = \sum_i \lambda_i v_i'$ of v
  based on latest update v_i' from site i

- Each site i tracks its drift from its most recent update $\Delta v_i = v_i - v_i'$

Covering the Convex Hull

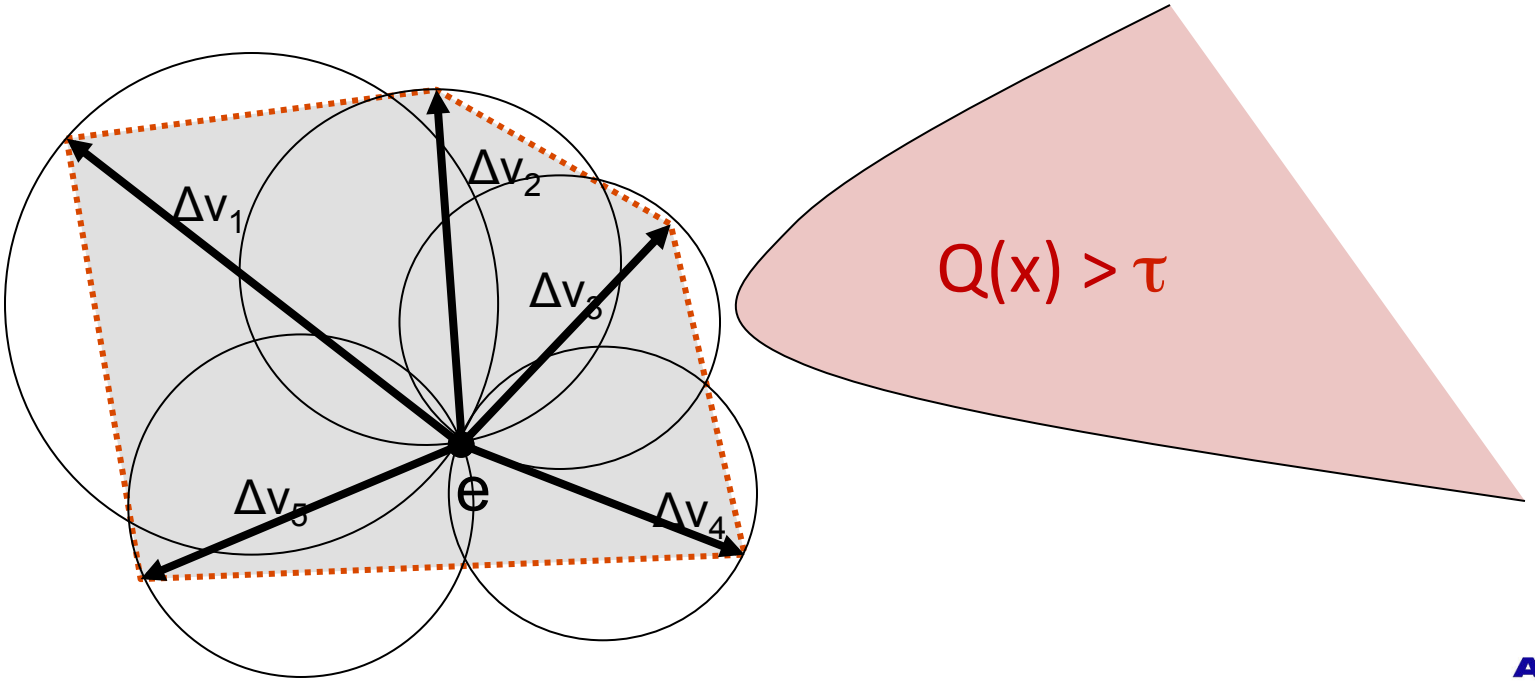
- Key observation: $v = \sum_i \lambda_i \cdot (e+\Delta v_i)$
  (a convex combination of "translated" local drifts)

- v lies in the convex hull of the $(e+\Delta v_i)$ vectors
- Convex hull is completely covered by spheres with radii $||\Delta v_i/2||_2$ centered at $e+\Delta v_i/2$
- Each such sphere can be constructed independently

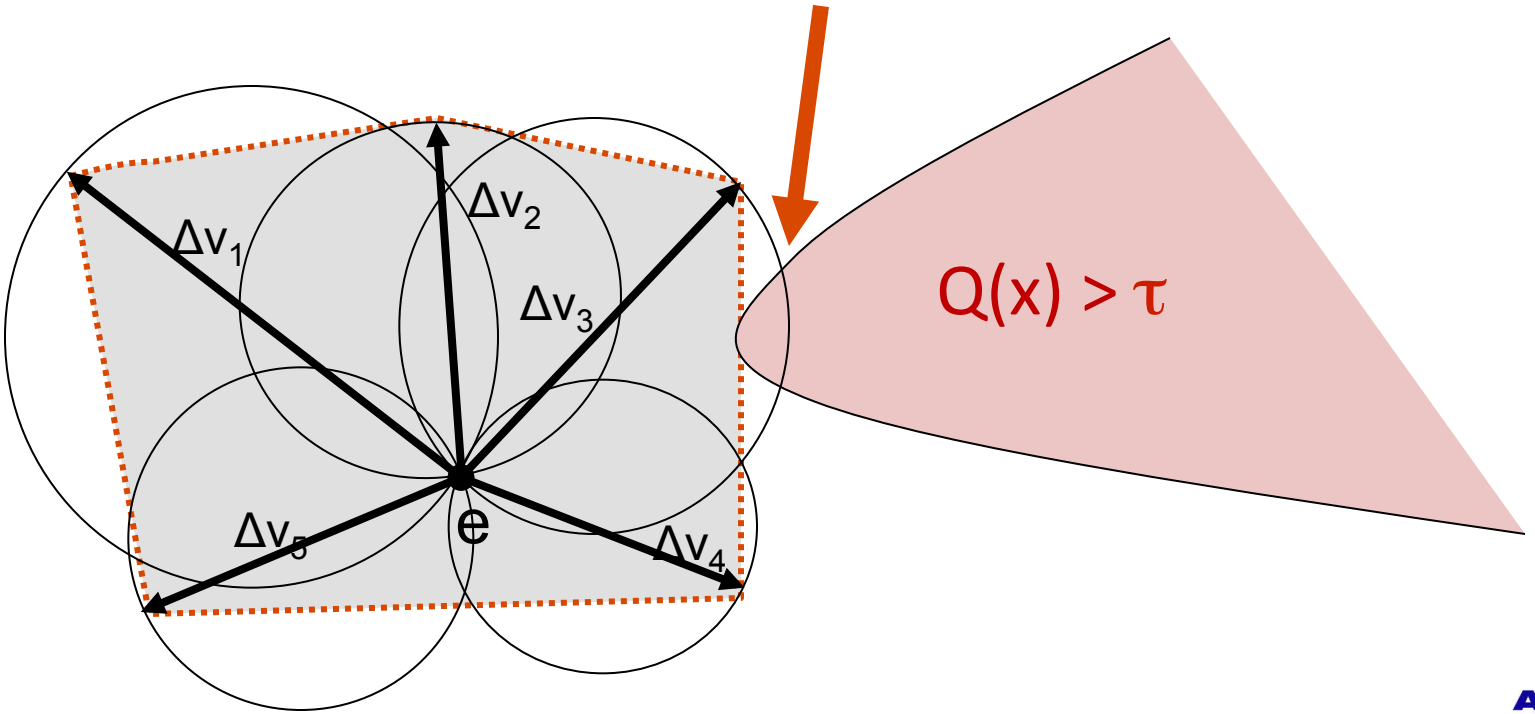


Monochromatic Regions

- Monochromatic Region: For all points x in the region, Q(x) is on the same side of the threshold ($Q(x) > \tau$ or $Q(x) \leq \tau$)
- Each site independently checks its sphere is monochromatic
  - Find max and min for Q() in local sphere region (may be costly)
  - Send updated value of v_i if not monochrome

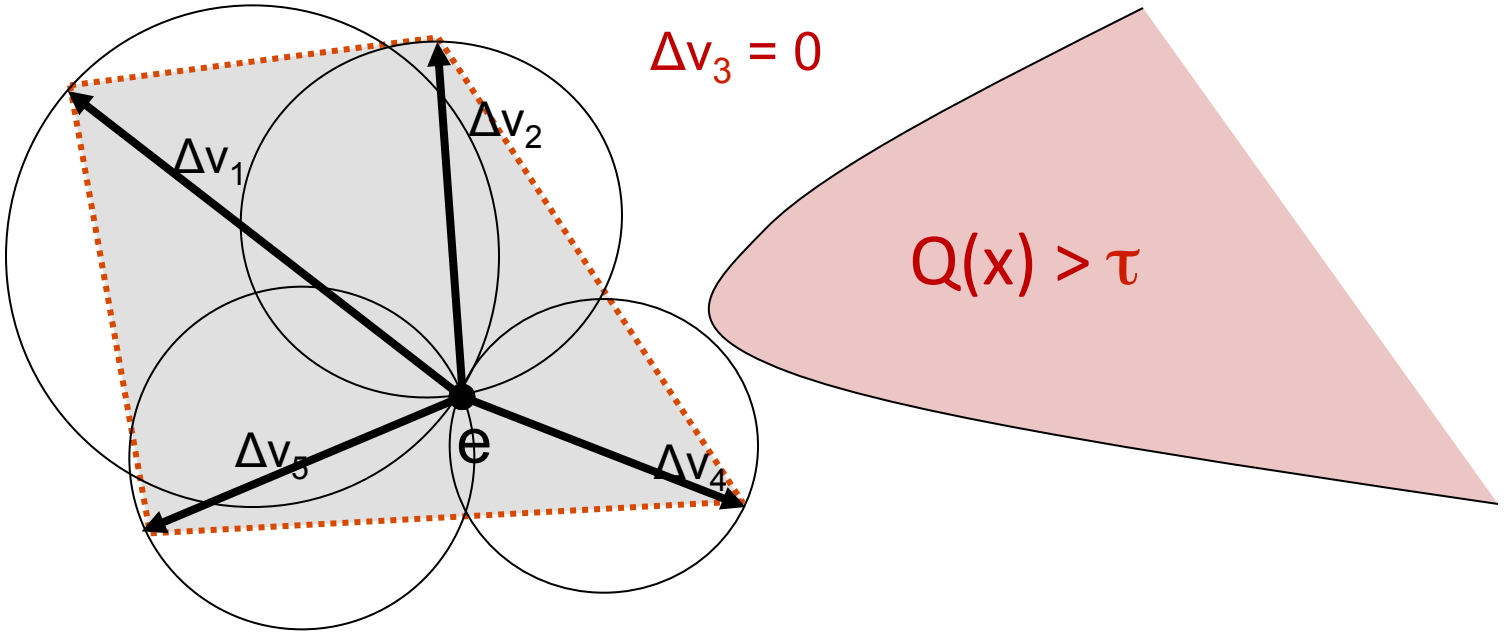


Restoring Monochromicity



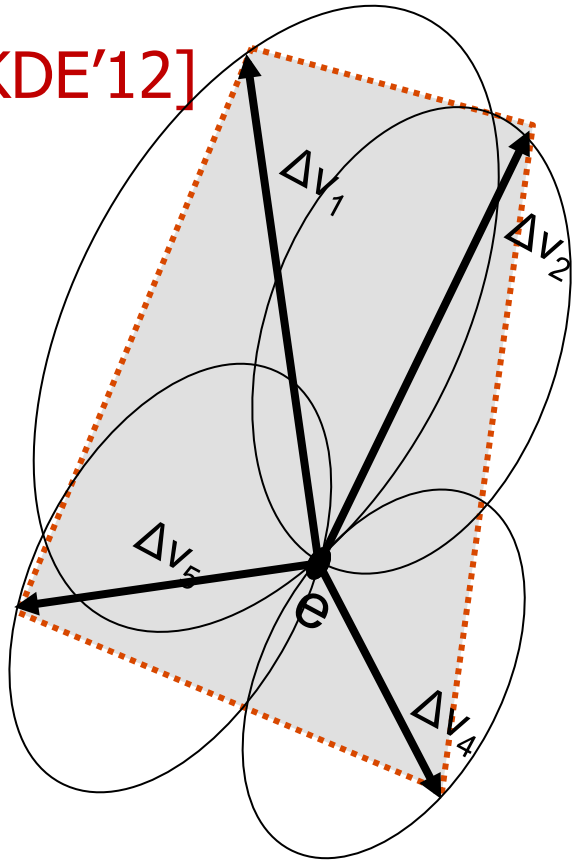
Restoring Monochromicity

- After update, $||\Delta v_i||_2 = 0 \Rightarrow$ Sphere at i is monochromatic
  - Global estimate e is updated, may cause more site updates
- Coordinator case*:* Can allocate local slack vectors to sites to enable "localized" resolutions
  - Drift (=radius) depends on slack (adjusted locally for subsets)



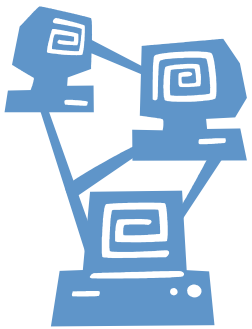
Extensions: Transforms, Shifts

- Subsequent developments [SKS TKDE'12]
  - Same analysis of correctness holds when spheres are allowed to be ellipsoids
  - Different reference vectors can be used to increase radius when close to threshold values
  - Combining these observations allows additional cost savings

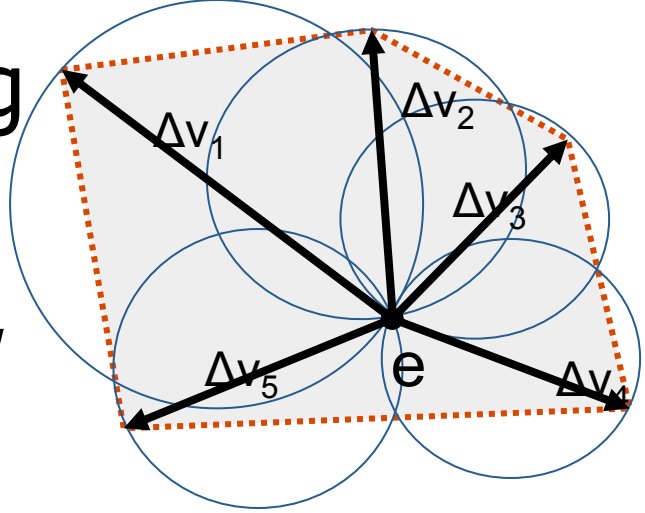


Outline – Part II

- Introduction: Continuous Distributed Streaming
- The Geometric Method (GM)
- Recent Work: GM + Sketches, GM + Prediction Models
- Towards Convex Safe Zones (SZs)
- Future Directions & Conclusions



Geometric Query Tracking using AMS Sketches [GKS VLDB'13]

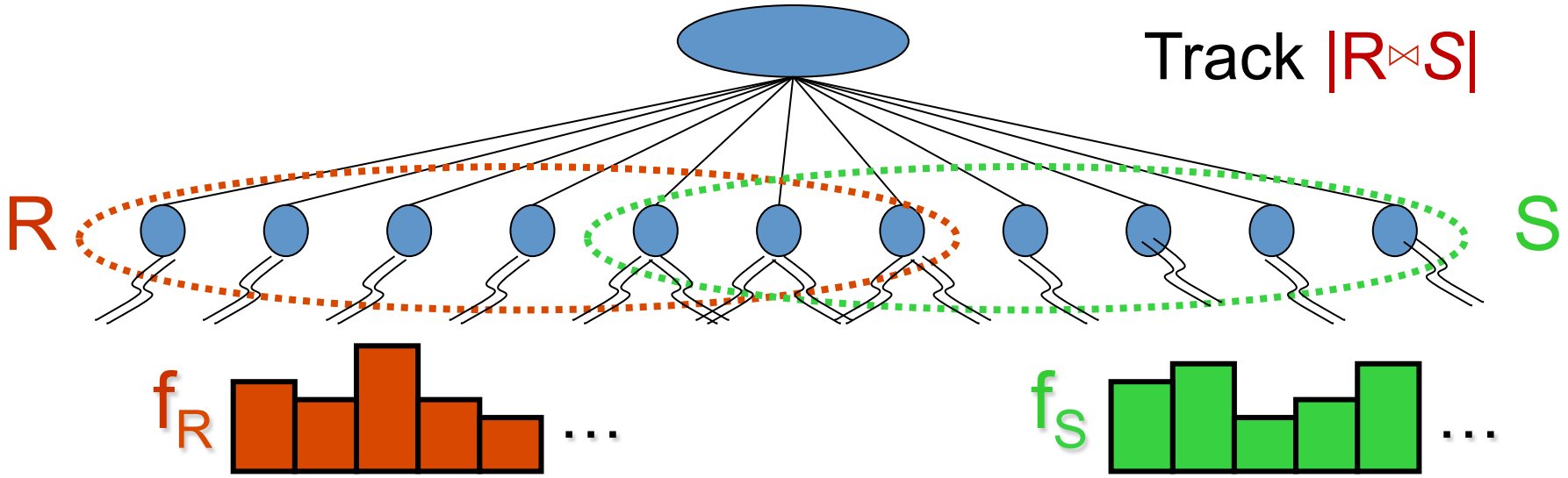


- *Continuous approximate monitoring*
  - Maintain the value of a function to within specified accuracy bound θ

- Too much local info ➔ *Local summaries at sites*
  - A form of dimensionality reduction
  - Bounding regions for the *lower-dimensional sketching space*
  - Function over sketch => Sketching error ε
    - Accounted for in the threshold checks (depend on both ε, θ)

- *Key Problems: (1) Minimize data exchange volume (2) Deal with highly-nonlinear AMS estimator*

Tracking Complex Aggregate Queries

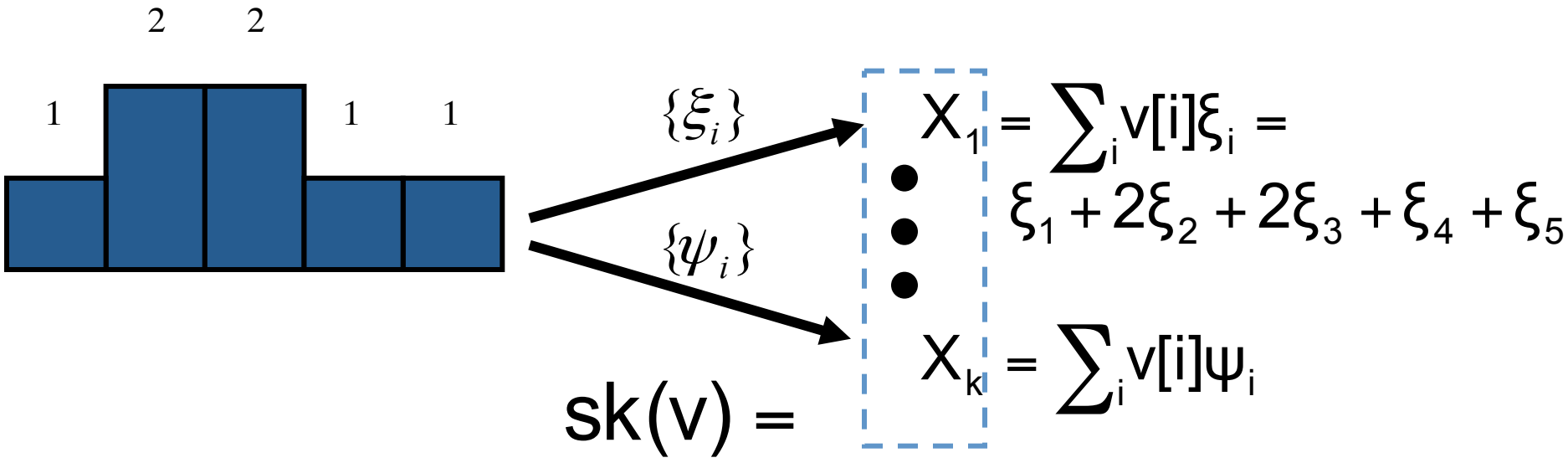


- *Class of queries:* Generalized inner products of streams

$$|R \bowtie S| = f_R \cdot f_S = \sum_v f_R[v]\, f_S[v]$$

  - Join/multi-join aggregates, range queries, heavy hitters, histograms, wavelets, …

AMS Sketches 101

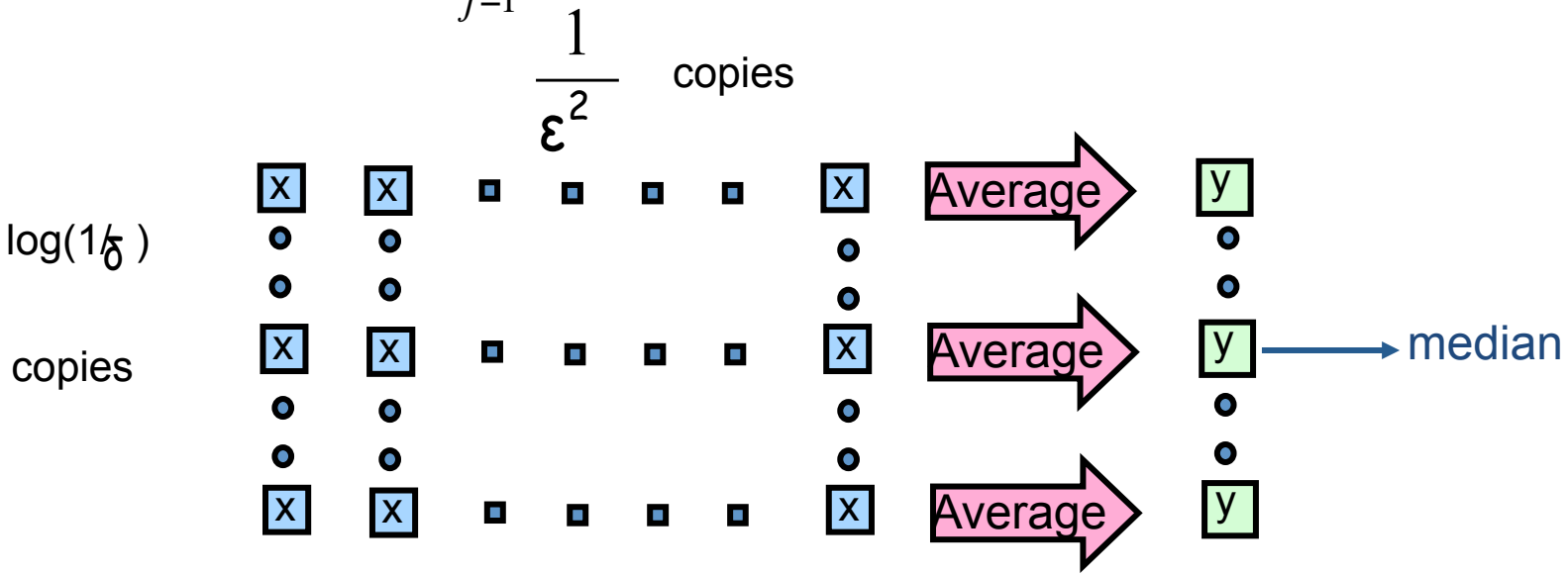


- Simple randomized linear projections of data distribution
  - Easily computed over stream using logarithmic space
  - *Linear:* Compose through simple vector addition

Monitored Function…?

AMS Estimator function for Self-Join

$$Q(sk(v)) = median_{i=1..n}\{\frac{1}{m}\sum_{j=1}^{m} sk(v)[i,j]^2\} = median_{i=1..n}\{\frac{1}{m}\| sk(v)[i]\|^2\}$$



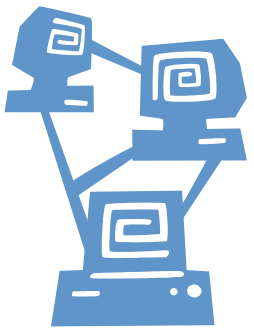
- Theorem(AMS96): Sketching approximates $\| v \|_2^2$ to within an error of $\pm \varepsilon \| v \|_2^2$ with probability $\geq 1-\delta$ using $O(\frac{1}{\varepsilon^2}\log(1/\delta))$ counters

Geometric Query Monitoring using AMS Sketches [GKS VLDB'13]

- Efficiently deciding ball monochromicity for median
  - Fast greedy algorithm for determining the distance to the inadmissible region
- *(Non-trivial!)* extension to *general inner product (join) queries*
- Minimizing volume of data exchanges
  - Sketches can still get pretty large!
  - Can reduce to monitoring in O(log(1/δ)) dimensions

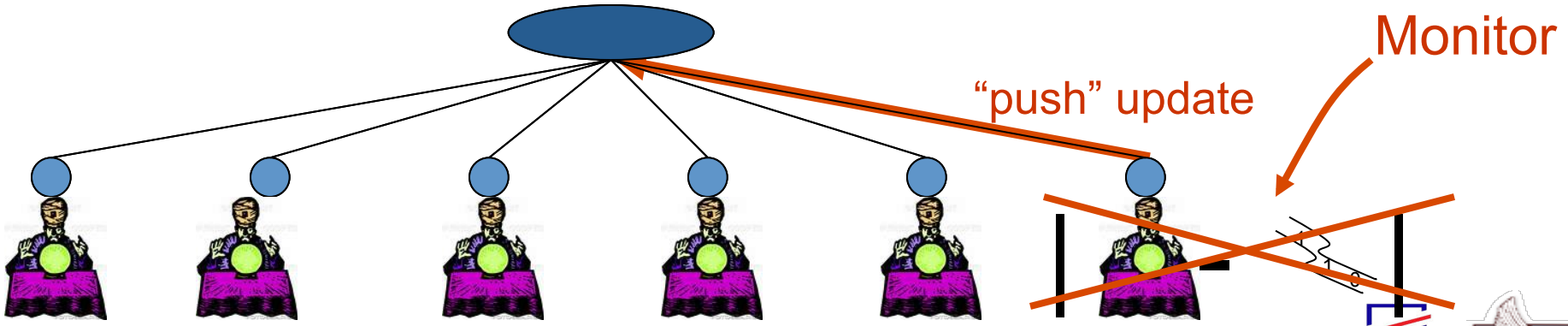
Outline – Part II

- Introduction: Continuous Distributed Streaming
- The Geometric Method (GM)
- Recent Work: GM + Sketches, GM + Prediction Models
- Towards Convex Safe Zones (SZs)
- Future Directions & Conclusions



Exploiting Shared Prediction Models

- Naïve *"static"* prediction: Local stream assumed "unchanged" since last update
  - No update from site ⇒ last update ("predicted" value) is unchanged ⇒ global estimate vector unchanged
- *Dynamic prediction models* of site behavior
  - Built locally at sites and *shared* with coordinator
  - Model complex stream patterns, reduce number of updates
  - But... more complex to maintain and communicate



Adopting Local Prediction Models

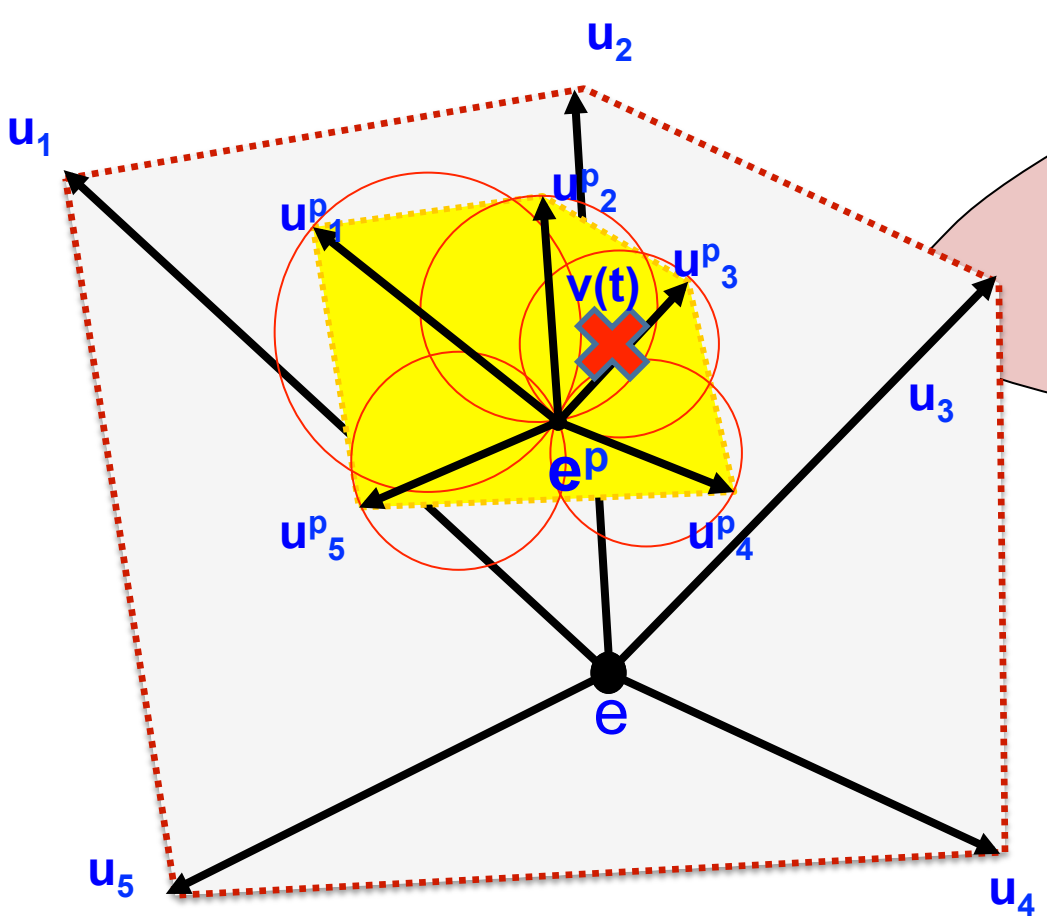
[CG VLDB'05, TODS'08]

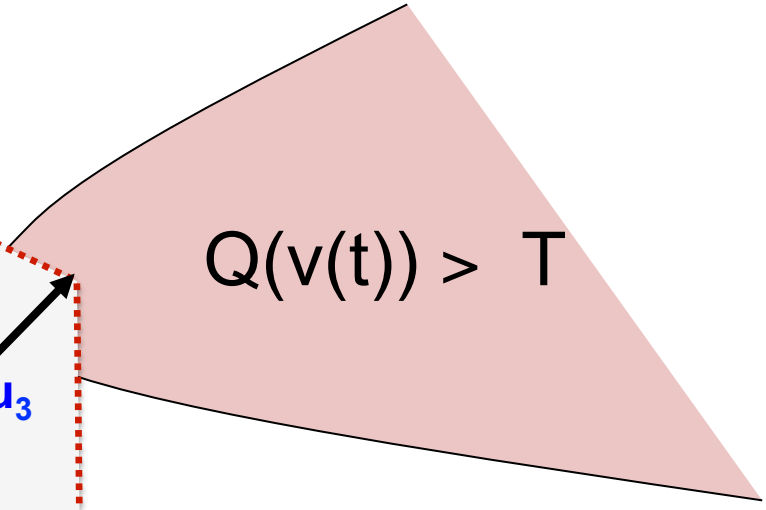
Model	Predicted v_i
Linear Growth	$v_i^p(t) = \frac{t}{t_s} v_i(t_s)$
Velocity/ Acceleration	$v_i^p(t) = v_i(t_s) + (t - t_s)vel_i + (t - t_s)^2 acc_i$
Static (Equivalent to the basic framework)	$v_i^p(t) = v_i(t_s)$

Predicted Global Vector: $e^p(t) = \sum \lambda_i v_i^p(t)$

Prediction-based Geometric Monitoring

[GDG SIGMOD'12, TODS'14]

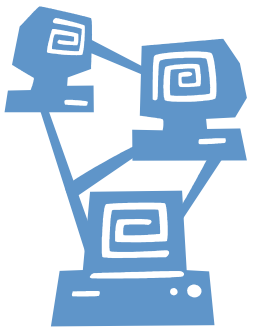




- Stricter local constraints if local predictions remain accurate
- Keep up with v(t) movement

Outline – Part II

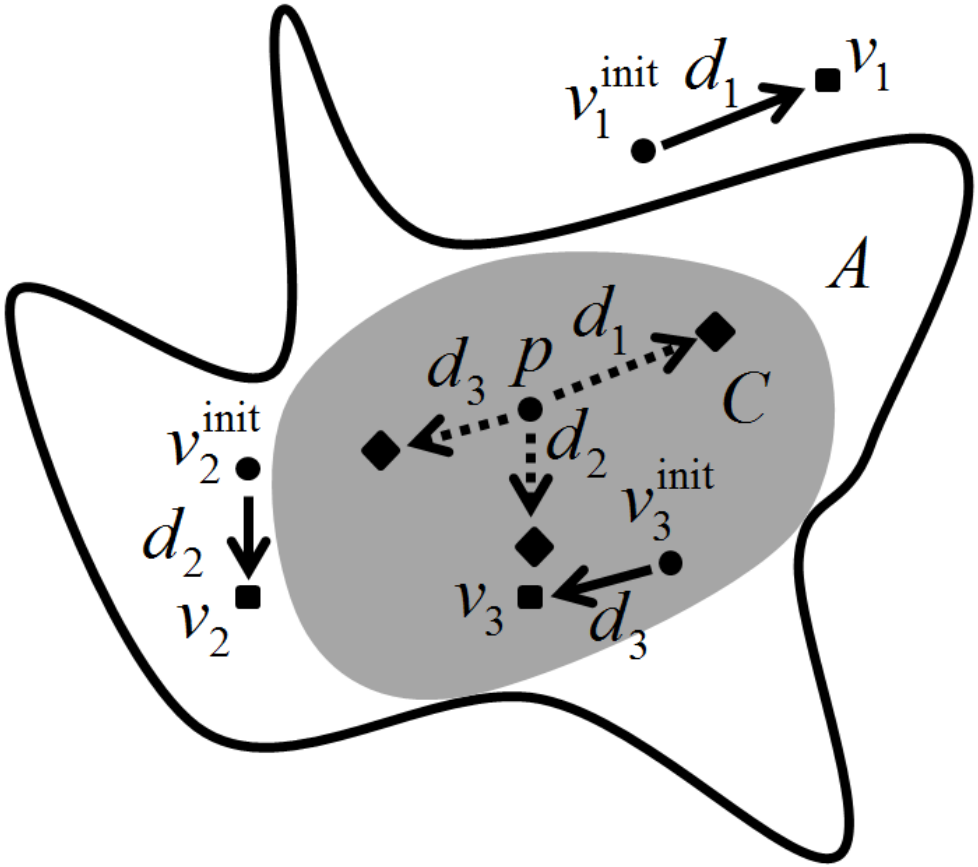
- Introduction: Continuous Distributed Streaming
- The Geometric Method (GM)
- Recent Work: GM + Sketches, GM + Prediction Models
- Towards Convex Safe Zones (SZs)
- Future Directions & Conclusions



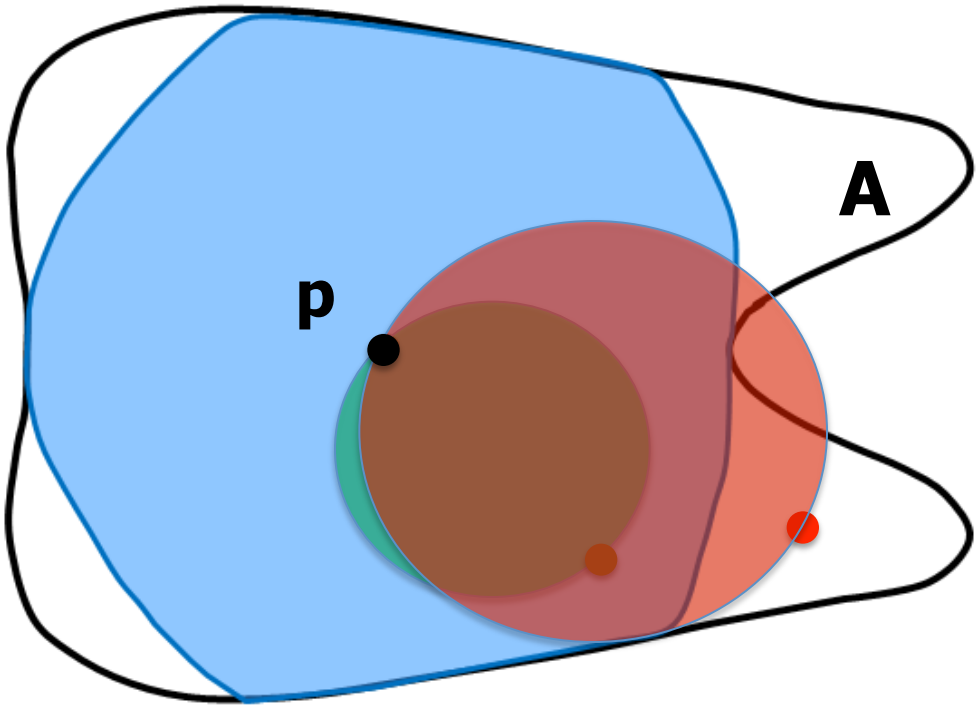
From Bounding Spheres to Safe Zones (SZs)

- **Safe Zone: Any convex subset of the Admissible Region**
  - As long as translated drifts stay within SZ, we are "safe"
    - By convexity

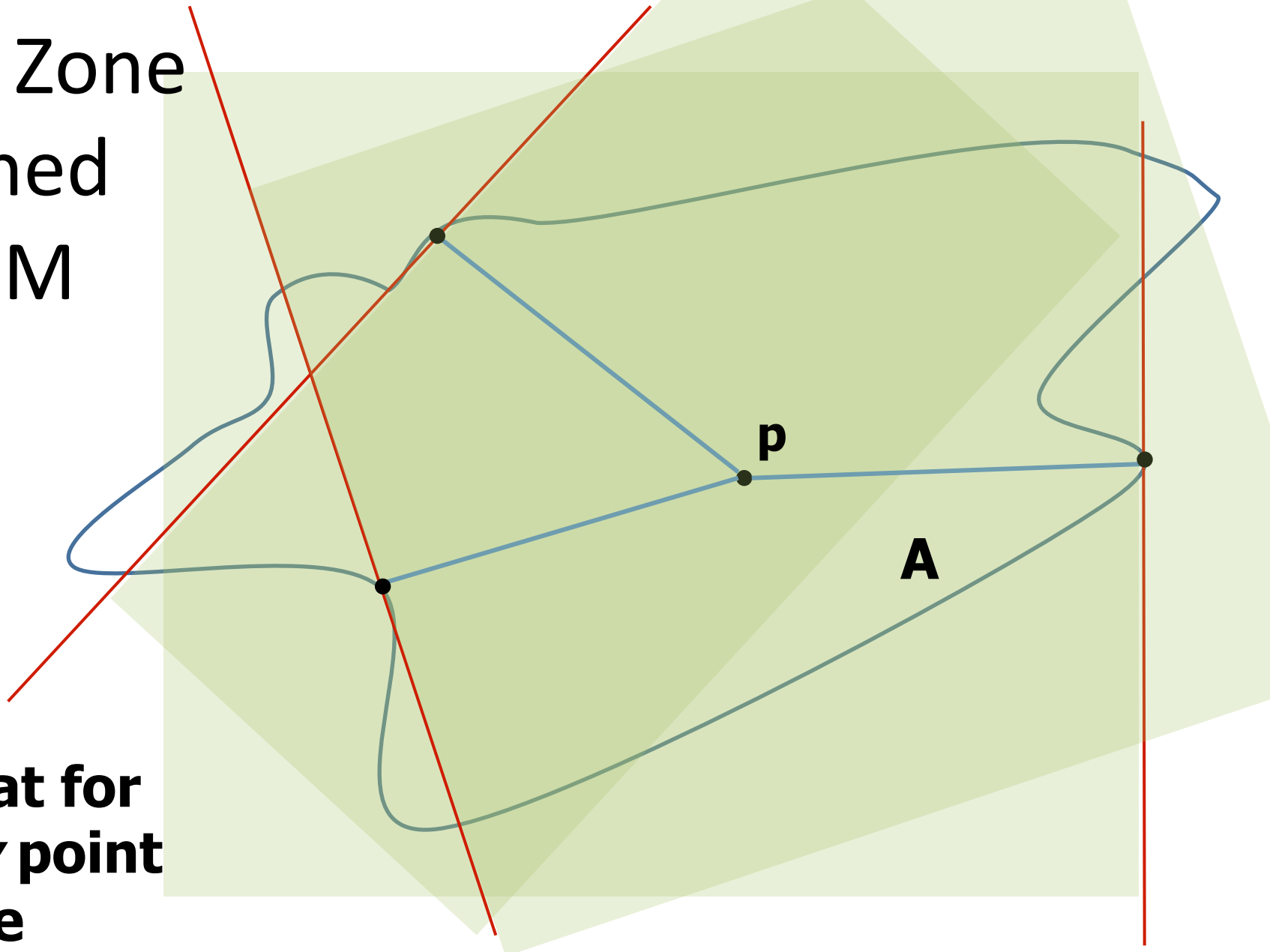
- Aim for large SZs, far from the boundary



Safe Zone defined by GM



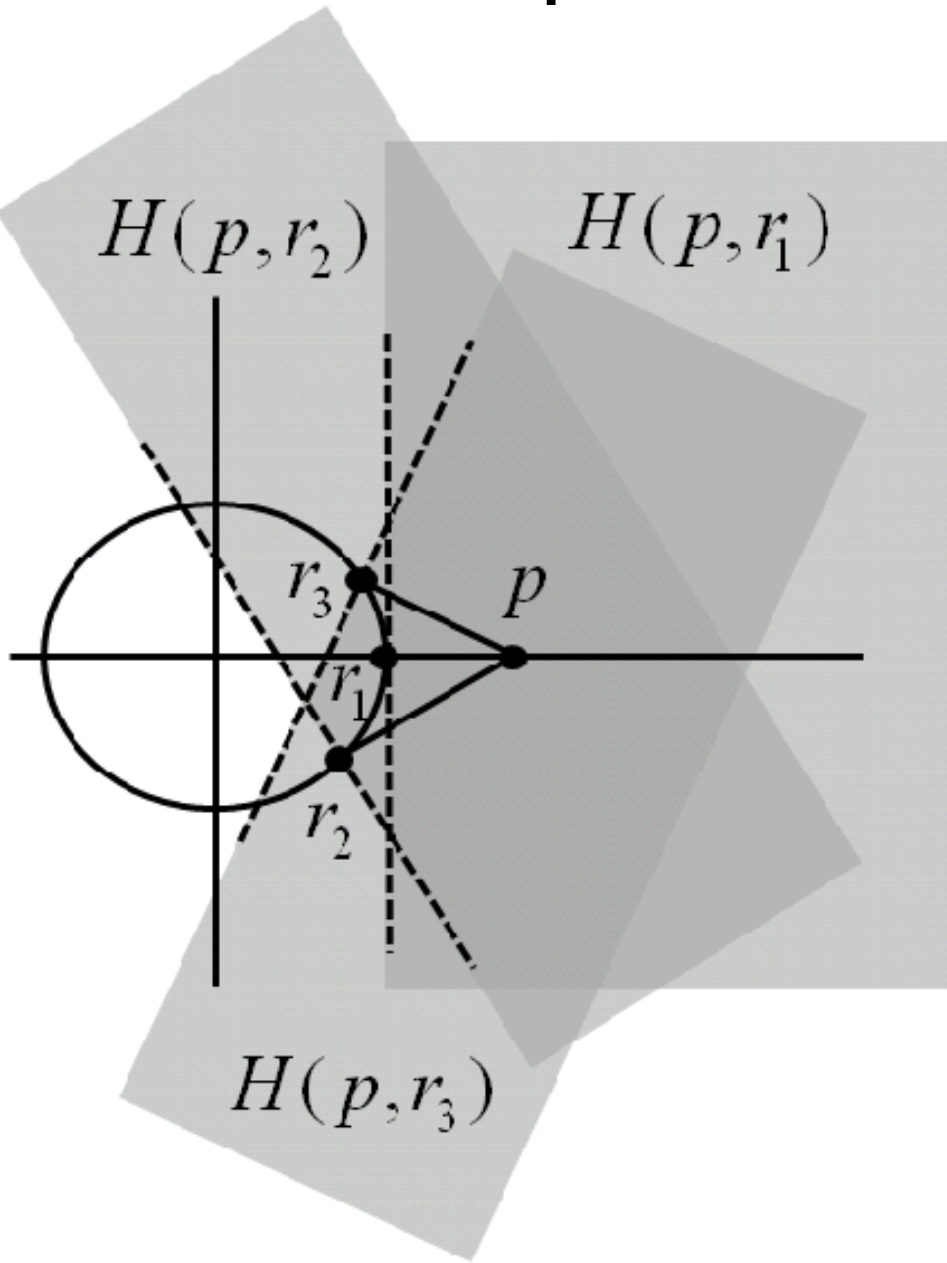
Safe Zone defined by GM



Repeat for *every* point on the boundary

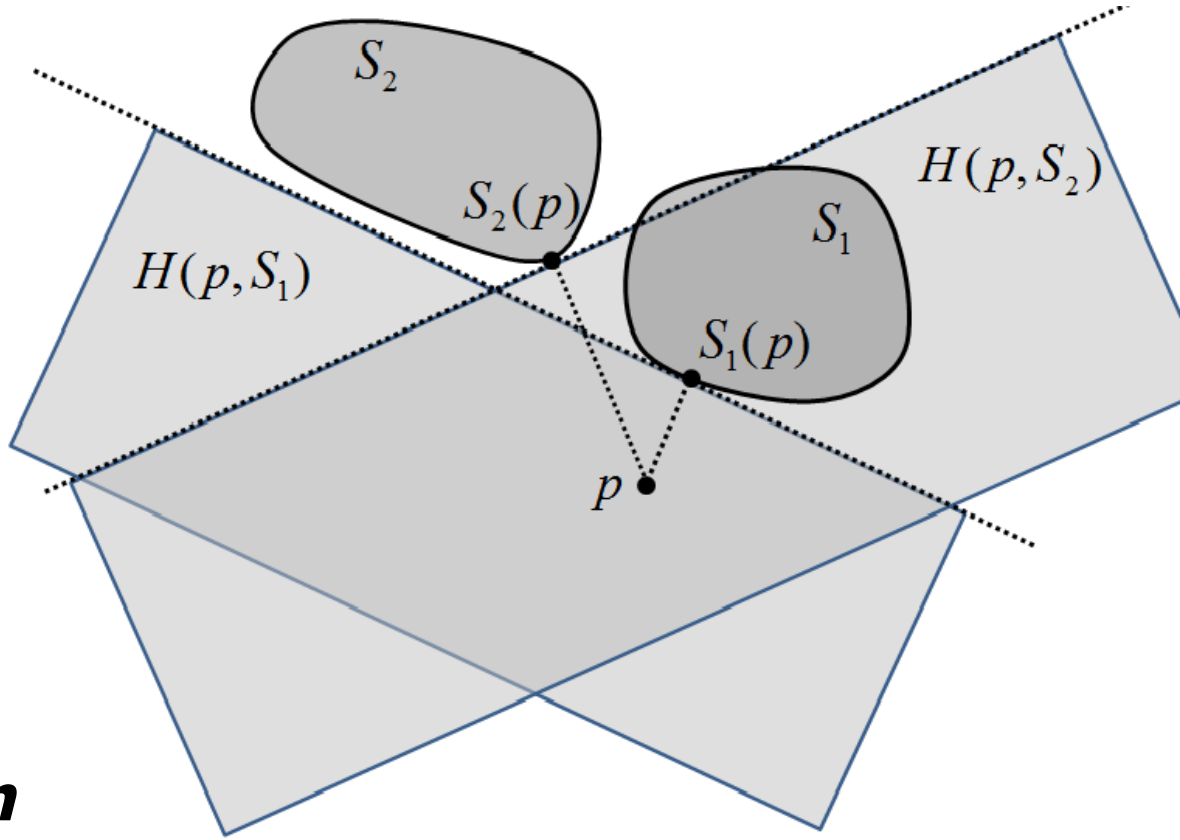
GM Safe Zones can be Far from Optimal!

- For instance, when inadmissible region is convex
- Taking the intersection of all half-spaces is overly restrictive
- In this case, half-space H(p,r1) is clearly the optimal SZ!



SZs through Convex Decompositions [VLDB'15]

- Inadmissible region is (can be covered by) a union of convex sets
- Just intersect half-spaces that separate p from each set
  - Avoid *redundancy!*



- ***Provably better than GM!***
- Application in sketches and median monitoring

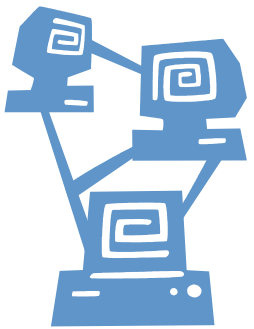
$S_1(p), S_2(p)$: "support vectors"

A "Cookbook" for Distributed Stream Monitoring?

- GM/bounding spheres is a generic, off-the-shelf technique
    - Any function, but can be far from optimal
- SZs: much better performance but must be designed for function/data at hand
  - Some initial progress on automated SZ construction (difficult optimization problem) **[TKDE'14]**
  - Generic mechanisms for composing SZs based on convex analysis **[ICDT'17]**

Outline – Part II

- Introduction: Continuous Distributed Streaming
- The Geometric Method (GM)
- Recent Work: GM + Sketches, GM + Prediction Models
- Towards Convex Safe Zones (SZs)
- Future Directions & Conclusions



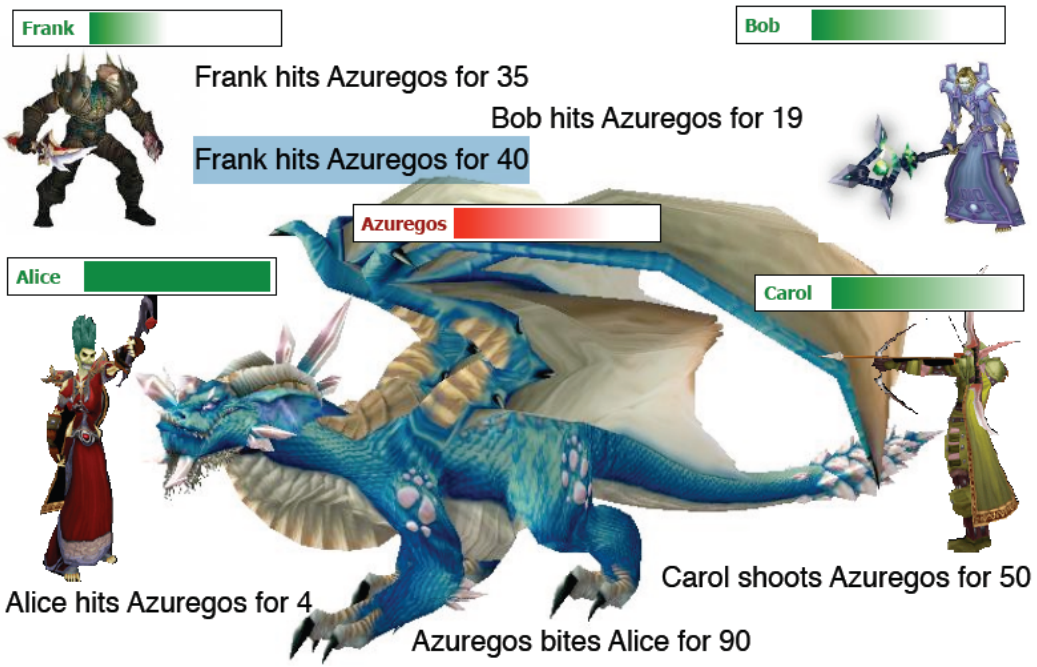
Work in CD Streaming

- Much interest in these problems in TCS and DB areas
- Many functions of (global) data distribution studied:
  - Set expressions [Das,Ganguly,G,Rastogi'04]
  - Quantiles and heavy hitters [Cormode,G, Muthukrishnan, Rastogi'05]
  - Number of distinct elements [Cormode et al.,'06]
  - Spectral properties of data matrix [Huang,G, et al.'06]
  - Anomaly detection in networks [Huang ,G, et al.'07]
  - Samples [Cormode et al.'10]
  - Counts, frequencies, ranks [Yi et al.,'12]
- See proceedings of recent NII Shonan meeting on Large-Scale Distributed Computation

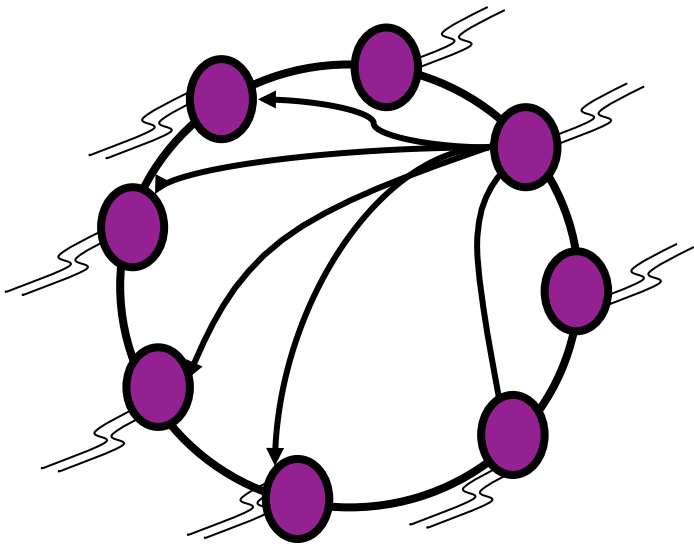
  http://www.nii.ac.jp/shonan/seminar011/

Monitoring Systems

- Much theory developed, but less progress on deployment
- Some empirical study in the lab, with recorded data
- Still, applications abound: Online Games [Heffner, Malecha'09]
  - Need to monitor many varying stats and bound communication
  - Also, Distributed CEP systems (FERARI project), **IoT (!)**
- Several steps to follow:
  - Build lib ms
  - Evolve t puted DBMSs?)
- Several qu
  - What fu specific?
  - What ke onitoring?



CD Monitoring in Scalable Network Architectures

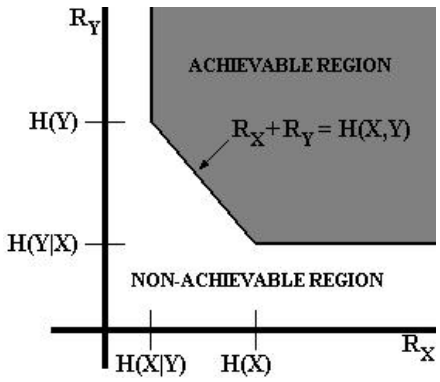


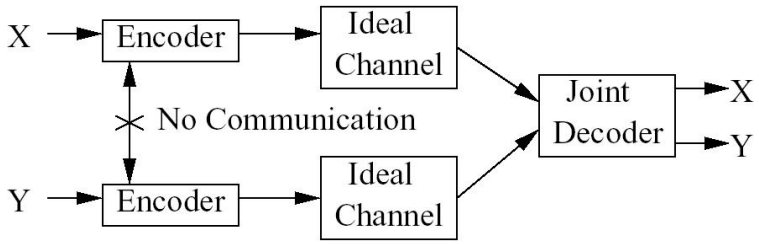
- E.g., DHT-based P2P networks
- Single query point
  - "Unfolding" the network gives hierarchy
  - But, single point of failure (i.e., root)
- Decentralized monitoring
  - Everyone participates in computation, all get the result
  - Exploit epidemics? Latency might be problematic…

Theoretical Foundations

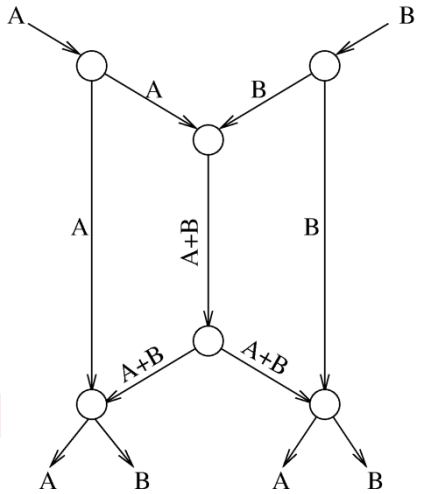
"Communication complexity" studies lower bounds of distributed one-shot computations

- Gives lower bounds for various problems, e.g., **count distinct** (via reduction to abstract problems)
- Need new theory for continuous computations
  - Based on info. theory and models of how streams evolve?
  - Link to distributed source coding or network coding?





Slepian-Wolf theorem [Slepian Wolf 1973]



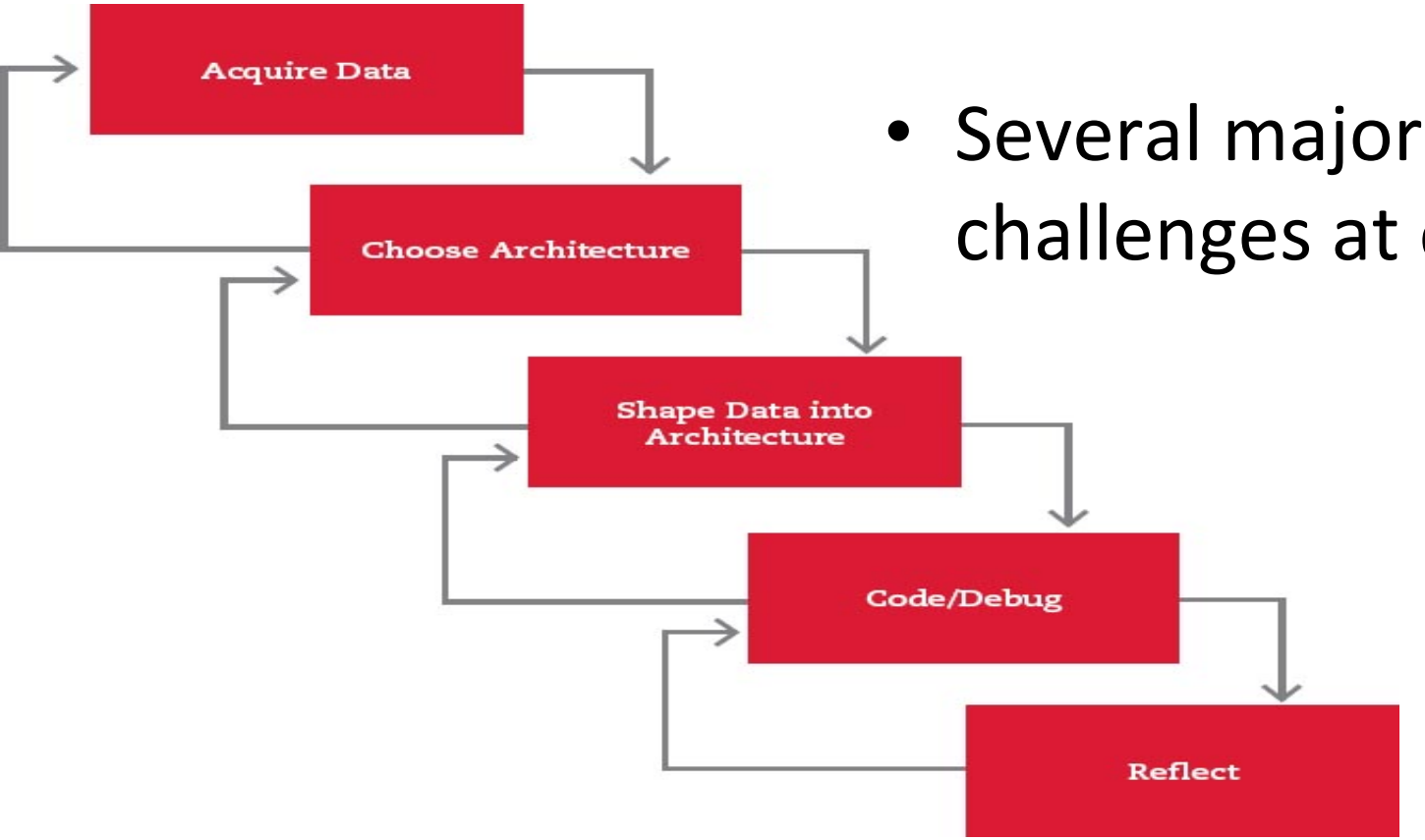
http://www.networkcoding.info/

https://buffy.eecs.berkeley.edu/PHP/resabs/resabs.php?f_year=2005&f_submit=chapgrp&f_chapter=1

Challenges, challenges, challenges…

- Guaranteeing privacy of sensitive data in μClouds?
- Geometric monitoring for Distributed CEP hierarchies?
  - Deal with uncertain events ("V" for Veracity)?
- Implementing GM ideas in scalable stream-processing engines (e.g., Storm)?
- CD machine learning to dynamically adapt to data/ workload conditions?
  - Communication just one of our concerns
- Scalable, adaptive analytics tools for massive, streaming *time series*?
- Streaming graph analytics functions (e.g., PageRank)?

The Big Data Pipeline



- Several major pain points/ challenges at each step

- Throwback to early batch computing of the 1960s!
  - No direct manipulation, interactivity, fast response
  - Processing is opaque, time consuming, costly
    - Typically, using a series of remote VMs

ATHENA Research & Innovation Information Technologies

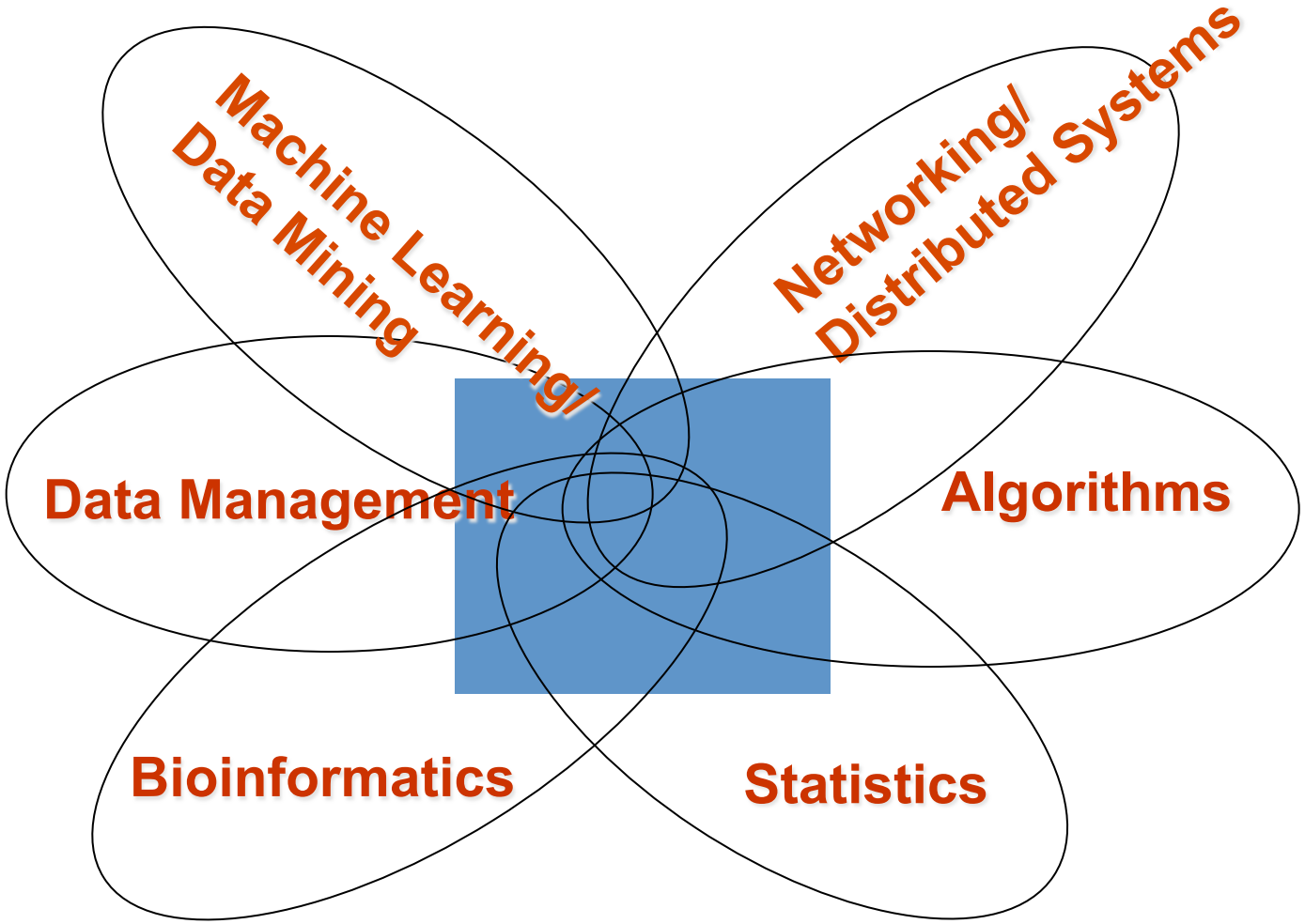
Vision: Interactive Big Data Science

- Can we support interactive exploration and rapid iteration over Big Data?
  - Data analytics is exploratory by nature!
  - Mimic versatility of local file handling with tools like Excel and scripts (e.g., R)

- Approach: **Scale-down + Scale-out**
  - Synopses and one-pass (streaming) algorithms **PLUS**
  - Parallelization/distribution over large computing infrastructures
    - Clusters (e.g., Spark, Storm/Heron, *Exareme*) or remote sensors (e.g., IoT)
  - Much to be done for complex analytics/ML tasks…

ATHENA Research & Innovation Information Technologies

Data Analytics Research in the 21st Century



Very exciting times!

Conclusions

- Continuous querying of distributed streams is a natural model
  - Interesting space/time/communication tradeoffs
  - Captures several real-world applications
- **Geometric Method** : Generic tool for monitoring complex, non-linear queries
  - Sketches [GKS VLDB'13], dynamic prediction models [GDG SIGMOD'12, TODS'14], Skyline Monitoring [PG ICDE'14]
- Much non-trivial algorithmic and theoretical work in CDS model
  - Intense research interest from DB and TCS communities
  - Deployment in real systems to come…
- ***Much interesting work to be done!***

Thank you!

http://www.softnet.tuc.gr/~minos/

ATHENA
Research & Innovation
Information Technologies

Some References…

1. Noga Alon, Yossi Matias, Mario Szegedy: *The Space Complexity of Approximating the Frequency Moments*. STOC 1996: 20-29
2. Noga Alon, Phillip B. Gibbons, Yossi Matias, Mario Szegedy: *Tracking Join and Self-Join Sizes in Limited Storage*. PODS 1999: 10-20
3. Graham Cormode, S. Muthukrishnan: *An Improved Data Stream Summary: The Count-Min Sketch and Its Applications*. LATIN 2004: 29-38
4. Phillip B. Gibbons: *Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports*. VLDB 2001: 541-550
5. Mayur Datar, Aristides Gionis, Piotr Indyk, Rajeev Motwani: *Maintaining Stream Statistics over Sliding Windows*. SIAM J. Comput. 31(6): 1794-1813, 2002
6. Graham Cormode, Minos N. Garofalakis: *Approximate continuous querying over distributed streams*. ACM Trans. Database Syst. 33(2), 2008
7. Izchak Sharfman, Assaf Schuster, Daniel Keren: *A geometric approach to monitoring threshold functions over distributed data streams*. SIGMOD Conference 2006: 301-312
8. Minos N. Garofalakis, Daniel Keren, Vasilis Samoladas: *Sketch-based Geometric Monitoring of Distributed Stream Queries*. PVLDB 6(10): 937-948, 2013
9. *Nikos Giatrakos, Antonios Deligiannakis, Minos Garofalakis, Izchak Sharfman, and Assaf Schuster. "Distributed Geometric Query Monitoring using Prediction Models", ACM Transactions on Database Systems, Vol. 39, No. 2, May 2014.*
10. Arnon Lazerson, Izchak Sharfman, Daniel Keren, Assaf Schuster, Minos Garofalakis, and Vasilis Samoladas. "Monitoring Distributed Streams using Convex Decompositions", VLDB'2015 (PVLDB 8(5)).
11. Minos Garofalakis and Vasilis Samoladas. "Distributed Query Monitoring through Convex Analysis: Towards Composable Safe Zones", ICDT'2017.

ATHENA Research & Innovation Information Technologies