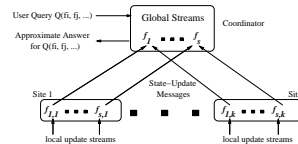
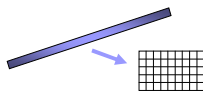


Processing Massive Data Streams



Minos Garofalakis

Yahoo! Research & UC Berkeley

minos@acm.org

(Thanks to: **Graham Cormode, Johannes Gehrke, Rajeev Rastogi**)



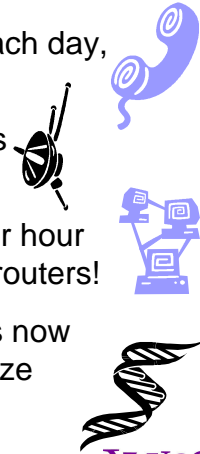
Streams – A Brave New World

- **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
 - Sensor networks
 - Telecom call-detail records
 - Network security
 - Financial applications
 - Manufacturing processes
 - Web logs and clickstreams
 - Other massive data sets...



Massive Data Streams

- Data is *continuously growing* faster than our ability to store or index it
- There are 3 Billion **Telephone Calls** in US each day, 30 Billion emails daily, 1 Billion SMS, IMs
- **Scientific data**: NASA's observation satellites generate billions of readings each per day
- **IP Network Traffic**: up to 1 Billion packets per hour per router. Each ISP has many (hundreds) routers!
- Whole **genome sequences** for many species now available: each megabytes to gigabytes in size



3

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Massive Data Stream Analysis

Must analyze this massive data:

- Scientific research (monitor environment, species)
- System management (spot faults, drops, failures)
- Business intelligence (marketing rules, new offers)
- For revenue protection (phone fraud, service abuse)

Else, why even measure this data?



4

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Example: IP Network Data



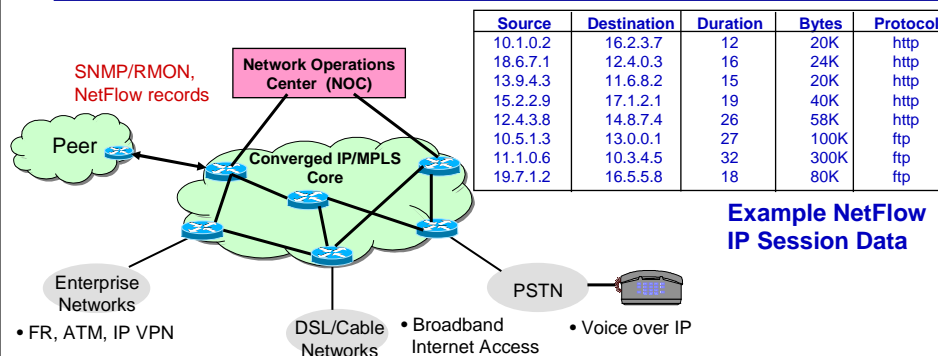
- Networks are sources of massive data: the **metadata** per hour per IP router is gigabytes
- Fundamental problem of data stream analysis:
*Too much information to **store** or **transmit***
- So process data as it arrives – *One pass, small space: the **data stream** approach*
- *Approximate answers* to many questions are OK, if there are guarantees of result quality

5

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

IP Network Monitoring Application



- 24x7 IP packet/flow data-streams at network elements
- Truly massive streams arriving at rapid rates
 - AT&T/Sprint collect ~1 Terabyte of NetFlow data each day
- Often shipped off-site to data warehouse for off-line analysis

6

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Packet-Level Data Streams

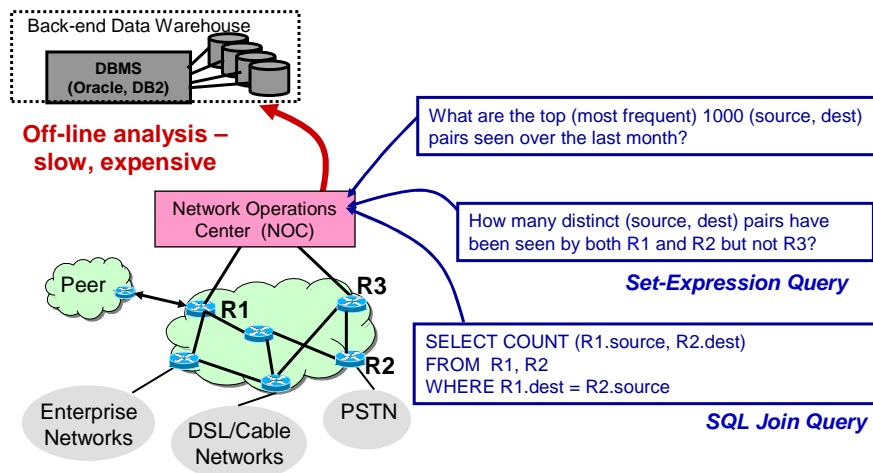
- Single 2Gb/sec link; say avg packet size is 50bytes
- Number of packets/sec = 5 million
- Time per packet = 0.2 microsec
- If we only capture **header information** per packet: src/dest IP, time, no. of bytes, etc. – at least 10bytes.
 - Space per second is 50Mb
 - Space per day is 4.5Tb per link
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

7

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Network Monitoring Queries



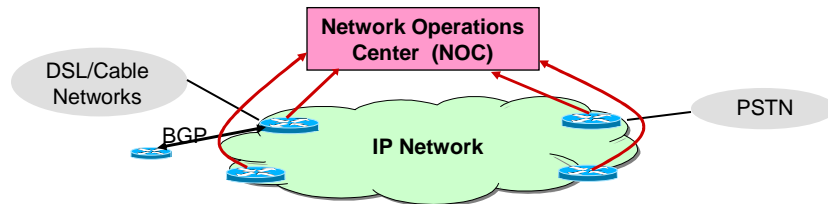
8

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

- Extra complexity comes from *limited space and time*
- Will introduce solutions for these and other problems

Real-Time Data-Stream Analysis



- Must process network streams in *real-time* and *one pass*
- Critical NM tasks: fraud, DoS attacks, SLA violations
 - Real-time traffic engineering to improve utilization
- *Tradeoff result accuracy vs. space/time/communication*
 - Fast responses, small space/time
 - Minimize use of communication resources

9

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sensor Networks

- Wireless sensor networks becoming ubiquitous in environmental monitoring, military applications, ...
- Many (100s, 10^3 , 10^6 ?) sensors scattered over terrain
- Sensors observe and process a local stream of readings:
 - Measure light, temperature, pressure...
 - Detect signals, movement, radiation...
 - Record audio, images, motion...



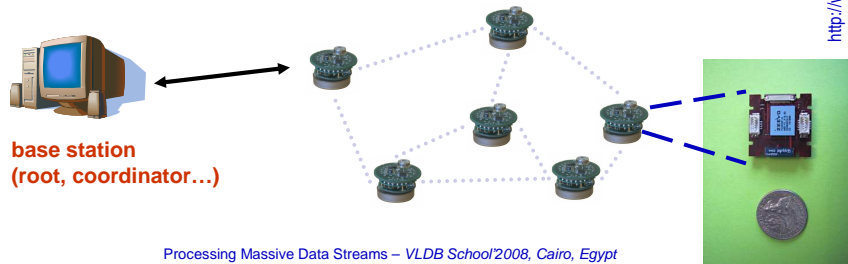
10

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sensornet Querying Application

- Query sensornet through a (remote) *base station*
- Sensor nodes have **severe resource constraints**
 - Limited battery power, memory, processor, radio range...
 - *Communication* is the major source of battery drain
 - “transmitting a single bit of data is equivalent to 800 instructions” [Madden et al.'02]



11

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

Tutorial Outline

- Motivation & Streaming Applications
- **Centralized Stream Processing**
 - Basic streaming models and tools
 - Stream synopses and applications
 - Sampling, sketches
 - The Sliding Window model
- Distributed Stream Processing
- Open Problems & Future Directions
- Conclusions

12

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Some Disclaimers...

- Fairly broad coverage, but still biased view of data-streaming world
 - Revolve around personal biases (line of work and interests)
 - Main focus on key algorithmic concepts, tools, and results – for both the *centralized and distributed settings*
 - Only minimal discussion of systems/prototypes
 - A lot more information out there
 - Sensornets [Madden'06]
 - Systems issues [Koudas, Srivastava'03], [Babcock et al.'02]
 - Theory/algorithms [Muthukrishnan'03]

13

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Data Streaming Model

- **Underlying signal:** One-dimensional array $A[1 \dots N]$ with values $A[i]$ all initially zero
 - Multi-dimensional arrays as well (e.g., row-major)
- Signal is implicitly represented via a *stream of update tuples*
 - j -th update is $\langle x, c[j] \rangle$ implying
 - $A[x] := A[x] + c[j]$ ($c[j]$ can be $>0, <0$)
- **Goal:** Compute functions on $A[]$ subject to
 - Small space
 - Fast processing of updates
 - Fast function computation
 - ...
- Complexity arises from massive length and domain size (N) of streams

14

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Example IP Network Signals



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

15

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Streaming Model: Special Cases

- **Time-Series Model**
 - Only x -th update updates $A[x]$ (i.e., $A[x] := c[x]$)
- **Cash-Register Model: Arrivals-Only Streams**
 - $c[x]$ is always > 0
 - Typically, $c[x]=1$, so we see a multi-set of items in one pass
 - Example: $\langle x, 3 \rangle, \langle y, 2 \rangle, \langle x, 2 \rangle$ encodes x 
the arrival of 3 copies of item x ,
 y 
2 copies of y , then 2 copies of x .
 - Could represent, e.g., packets on a network; power usage

16

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

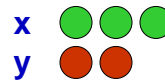
Streaming Model: Special Cases

■ Turnstile Model: Arrivals and Departures

- Most general streaming model
- $c[x]$ can be >0 or <0

■ Arrivals and departures:

- Example: $\langle x, 3 \rangle, \langle y, 2 \rangle, \langle x, -2 \rangle$ encodes final state of $\langle x, 1 \rangle, \langle y, 2 \rangle$.
- Can represent fluctuating quantities, or measure differences between two distributions



- *Problem difficulty varies depending on the model*
 - E.g., MIN/MAX in Time-Series vs. Turnstile!

17

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Approximation and Randomization

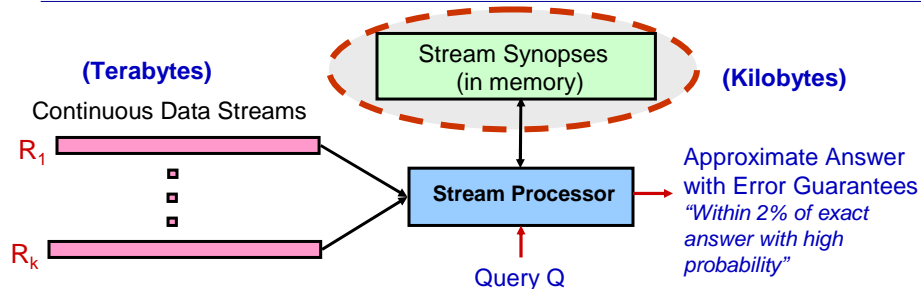
- Many things are hard to compute exactly over a stream
 - Is the count of all items the same in two different streams?
 - Requires linear space to compute exactly
- **Approximation**: find an answer correct within some factor
 - Find an answer that is within 10% of correct result
 - More generally, a $(1 \pm \epsilon)$ factor approximation
- **Randomization**: allow a small probability of failure
 - Answer is correct, except with probability 1 in 10,000
 - More generally, success probability $(1-\delta)$
- **Approximation and Randomization**: (ϵ, δ) -approximations

18

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Data-Stream Algorithmics Model



- *Approximate answers*– e.g. trend analysis, anomaly detection
- Requirements for stream synopses
 - *Single Pass*: Each record is examined at most once
 - *Small Space*: Log or polylog in data stream size
 - *Small-time*: Low per-record processing time (maintain synopses)
 - Also: *delete-proof*, *composable*, ...

19

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Probabilistic Guarantees

- User-tunable (ϵ, δ) -approximations
 - Example: Actual answer is within 5 ± 1 with prob ≥ 0.9
- **Randomized algorithms**: Answer returned is a specially-built *random variable*
 - *Unbiased* (correct on expectation)
 - Combine several *Independent Identically Distributed (iid)* instantiations (average/median)
- Use *Tail Inequalities* to give probabilistic bounds on returned answer
 - *Markov Inequality*
 - *Chebyshev Inequality*
 - *Chernoff Bound*
 - *Hoeffding Bound*

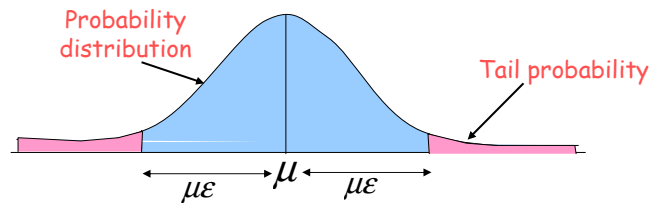
20

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

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 $\text{Var}[X]$. Then, for any $\varepsilon > 0$

Markov:

$$\Pr(X \geq (1 + \varepsilon)\mu) \leq \frac{1}{1 + \varepsilon}$$

Chebyshev:

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

21

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Tail Inequalities for Sums

- Possible to derive stronger bounds on tail probabilities for the *sum of independent random variables*
- Hoeffding Bound:** Let X_1, \dots, X_m be independent random variables with $0 \leq X_i \leq r$. Let $\bar{X} = \frac{1}{m} \sum_i X_i$ and μ be the expectation of \bar{X} . Then, for any $\varepsilon > 0$,

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

- Application:** Sample average \approx population average
 - See below...

22

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Tail Inequalities for Sums

- Possible to derive even stronger bounds on tail probabilities for the sum of *independent Bernoulli trials*
- **Chernoff Bound:** Let X_1, \dots, X_m be independent Bernoulli trials such that $\Pr[X_i=1] = p$ ($\Pr[X_i=0] = 1-p$). Let $X = \sum_i X_i$ and $\mu = mp$ be the expectation of X . Then, for any $\epsilon > 0$,

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

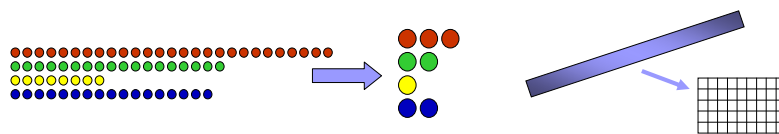
- **Application:** Sample selectivity \approx population selectivity
 - See below...
- **Remark:** Chernoff bound results in tighter bounds for *count queries* compared to Hoeffding bound

23

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sampling, Sketches and Applications



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 R where $R.e$ is odd

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

Sample S : 9 5 1 8

- If agg is avg, return average of odd elements in S **answer: 5**
- If agg is count, return average over all elements e in S of
 - n if e is odd
 - 0 if e is even**answer: $12 * 3/4 = 9$**

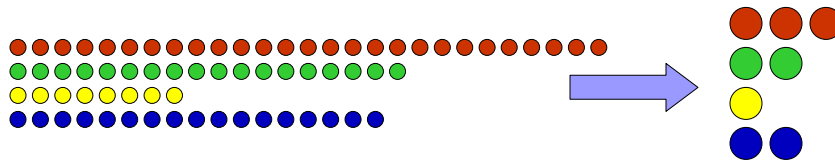
- Unbiased Estimator (for count, avg, sum, etc.)
 - Bound error using *Hoeffding* (sum, avg) or *Chernoff* (count)

25

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sampling from a Data Stream



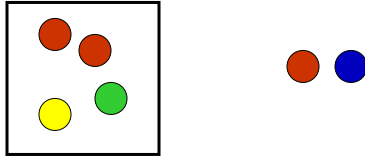
- Fundamental problem: sample m items uniformly from stream
 - Useful: approximate costly computation on small sample
- Challenge: don't know how long stream is
 - So when/how often to sample?
- Two solutions, apply to different situations:
 - Reservoir sampling (dates from 1980s?)
 - Min-wise sampling (dates from 1990s?)

26

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Reservoir Sampling



- Sample first m items
- Choose to sample the i 'th item ($i > m$) with probability m/i
- If sampled, randomly replace a previously sampled item
- **Optimization:** when i gets large, compute which item will be sampled next, skip over intervening items [Vitter'85]

27

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Reservoir Sampling - Analysis

- Analyze simple case: sample size $m = 1$
- Probability i 'th item is the sample from stream length n :
 - Prob. i is sampled on arrival \times prob. i survives to end

$$\frac{1}{i} \times \frac{i}{i-1} \times \frac{i-1}{i-2} \dots \frac{i-2}{i-1} \times \frac{1}{n}$$

$= 1/n$

- Case for $m > 1$ is similar, easy to show uniform probability
- Drawbacks of reservoir sampling: hard to parallelize

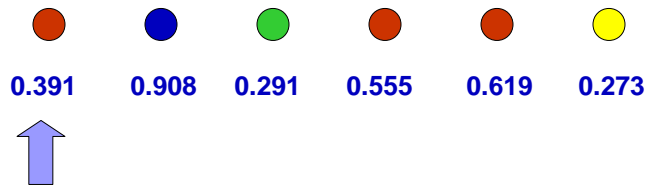
28

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Min-wise Sampling

- For each item, pick a random fraction between 0 and 1
- Store item(s) with the smallest random tag [Nath et al.'04]



- Each item has same chance of least tag, so uniform
- Can run on multiple streams separately, then merge

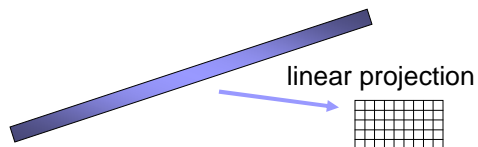
29

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketches

- Not every problem can be solved with sampling
 - **Example:** counting how many distinct items in the stream
 - If a large fraction of items aren't sampled, don't know if they are all same or all different
- Other techniques take advantage that the algorithm can “see” all the data even if it can't “remember” it all
- **“Sketch”:** essentially, a linear transform of the input
 - Model stream as defining a vector, sketch is result of multiplying stream vector by an (implicit) matrix

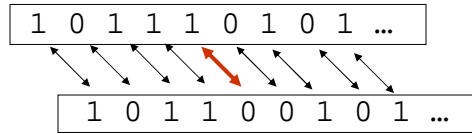


30

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Trivial Example of a Sketch



- Test if two (asynchronous) binary streams are equal
 $d_-(x,y) = 0$ iff $x=y$, 1 otherwise
- To test in small space: pick a random hash function h
- Test $h(x)=h(y)$: small chance of false positive, no chance of false negative.
- Compute $h(x)$, $h(y)$ incrementally as new bits arrive
 (Karp-Rabin: $h(x) = x_i 2^i \bmod p$)
 - Exercise: extend to real valued vectors in update model

31

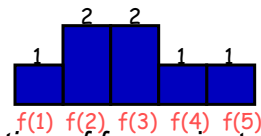
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

AMS Sketching

- **Goal:** Build small-space summary for distribution vector $f[v]$ ($v=1,\dots, N$) seen as a stream of v -values

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



- **Basic Construct:** Randomized Linear Projection of $f =$ project onto dot product of f -vector

$$X = \sum_v f[v] \xi_v \quad \text{where } \xi = \text{vector of random values from an appropriate distribution}$$

- Simple to compute: Add ξ_v whenever the value v is seen

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



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

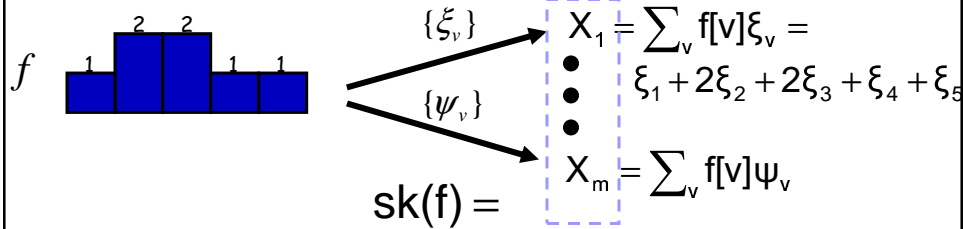
- Generate ξ_v 's in small ($\log N$) space using pseudo-random generators

32

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

AMS Sketching (contd.)



- 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 $O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$

$$sk(f_i) \cdot sk(f_j) \in f_i \cdot f_j \pm \epsilon \|f_i\|_2 \|f_j\|_2$$

33 **Inner Product**

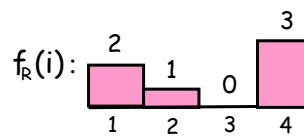
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

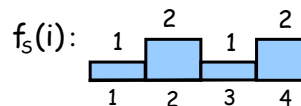
Application: Binary-Join COUNT Query

- 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



$$\begin{aligned}
 COUNT(R \bowtie_A S) &= \sum_i f_R(i) \cdot f_S(i) \\
 &= 10 \quad (2 + 2 + 0 + 6)
 \end{aligned}$$

Inner Product

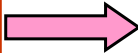
- Exact solution:** too expensive, requires $O(N)$ space!
 - $N = \text{sizeof}(\text{domain}(A))$

34

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Basic AMS Sketching Technique

- 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)
 - $E[X] = \text{COUNT}(R \bowtie_A S)$
 - $\text{Var}[X]$ 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 $\{\xi_i : i = 1, \dots, N\}$
 - $\Pr[\xi_i = +1] = \Pr[\xi_i = -1] = 1/2$
 - Expected value of each ξ_i , $E[\xi_i] = 0$
 - Variables ξ_i are 4-wise independent
 - Expected value of product of 4 distinct $\xi_i = 0$
 - Variables ξ_i can be generated using pseudo-random generator using only $O(\log N)$ space (for seeding)!

35

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH


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:

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

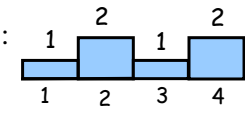
$X_R = X_R + \xi_4$

$f_R(i)$: 

$X_R = 2\xi_1 + \xi_2 + 3\xi_4$

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

$X_S = X_S + \xi_1$

$f_S(i)$: 

$X_S = \xi_1 + 2\xi_2 + \xi_3 + 2\xi_4$

36

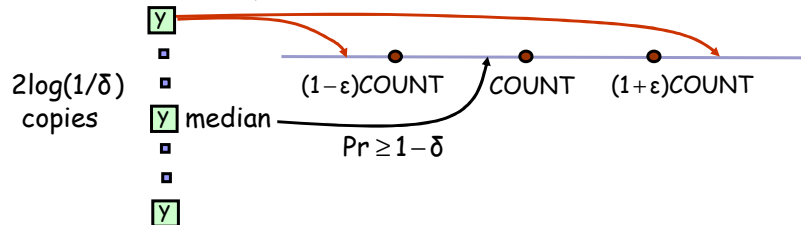
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Boosting Confidence

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

"FAILURE": $\Pr \leq 1/8$



$$\Pr[|\text{median}(Y) - \text{COUNT}| \geq \varepsilon \cdot \text{COUNT}]$$

$$= \Pr[\# \text{ failures in } 2 \log(1/\delta) \text{ trials} \geq \log(1/\delta)] \leq \delta \quad (\text{by Chernoff Bound})$$

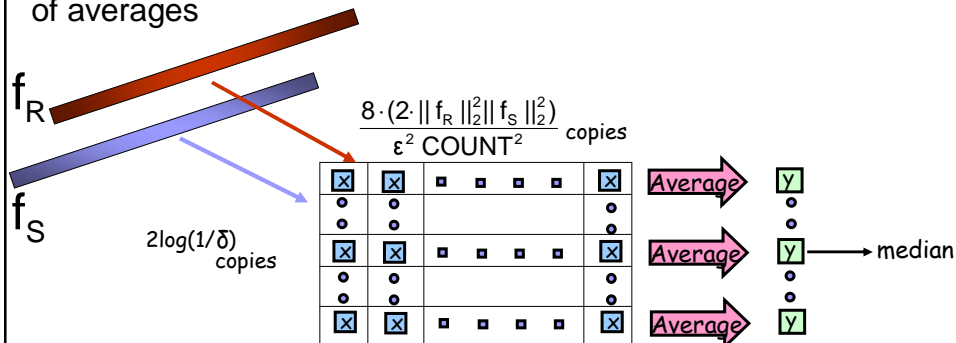
39

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

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



40

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Summary of Binary-Join AMS Sketching

- **Main Theorem [AGMS99]:** Sketching approximates COUNT to within a *relative error of ϵ* with probability $\geq 1-\delta$ using space

$$O\left(\frac{\|f_R\|_2^2 \|f_S\|_2^2 \cdot \log(1/\delta) \log N}{\epsilon^2 \text{COUNT}^2}\right)$$

– Remember: $O(\log N)$ space for “seeding” the construction of each X

- **Special Case – Self-join size:** $\text{COUNT}(R \bowtie_A R) = \|f_R\|_2^2$
 - Gini index of heterogeneity, measure of skew in the data
 - (ϵ, δ) -estimate using space only $O\left(\frac{\log(1/\delta) \log N}{\epsilon^2}\right)$
 - Best-case for AMS streaming join-size estimation...
 - **Q: What's the worst case??**

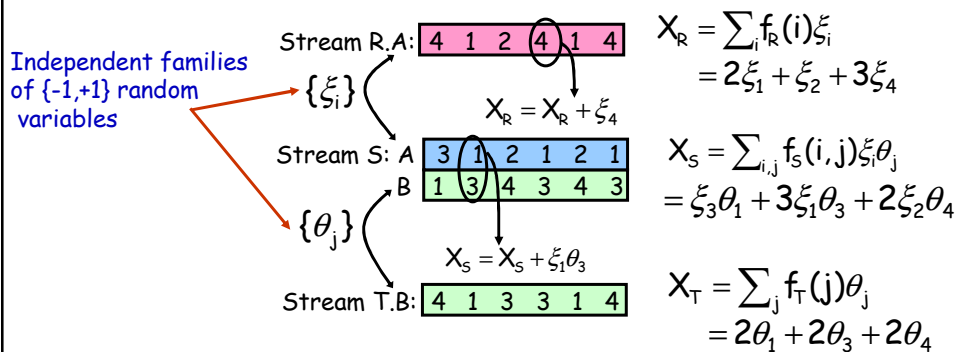
41

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

AMS Sketching for Multi-Joins [Dobra et al.02]

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



– Return $X = X_R X_S X_T$

$$(E[X] = \text{COUNT}(R \bowtie_A S \bowtie_B T))$$

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

42

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

AMS Sketching for Multi-Joins [Dobra et al.02]

- Sketches for general *multi-join COUNT queries* (over streams R, S, T, ...)

- For each pair of attributes in equality join constraint, use independent family of $\{-1, +1\}$ random variables
- Compute random variables X_R, X_S, X_T, \dots

Stream S:

A	3	1	2	1	2	1
B	1	3	4	3	4	3
C	2	4	1	2	3	1

Independent families of $\{-1, +1\}$ random variables $\{\xi_i\}, \{\theta_j\}, \{\lambda_k\}, \dots$

$$X_S = \sum_{i,j,k,\dots} f_S(i,j,k,\dots) \xi_i \theta_j \lambda_k \dots$$

$$X_S = X_S + \xi_1 \theta_3 \lambda_4 \dots$$

- Return $X = X_R X_S X_T \dots$ ($E[X] = \text{COUNT}(R \bowtie S \bowtie T \bowtie \dots)$)

$$\text{Var}[X] \leq 2^{2m} \cdot \|f_R\|_2^2 \|f_S\|_2^2 \|f_T\|_2^2 \dots$$

- Explosive increase with the number of joins!*

43

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Boosting Accuracy by Sketch Partitioning

- For ϵ error, need $\text{Var}[Y] \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$

$$s = \frac{8 \cdot (2^{2m} \|f_R\|_2^2 \|f_S\|_2^2 \dots)}{\epsilon^2 \text{COUNT}^2} \text{ copies}$$

X X □ □ □ □ X ➔ Y

Average

$$\text{Var}[Y] = \frac{\text{Var}[X]}{s} \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$$

- Key Observation:** Product of self-join sizes for *partitions of streams* can be *much smaller* than product of self-join sizes

- Reduce space requirements by partitioning join attribute domains
 - Overall join size = **sum of join size estimates for partitions**
- Exploit coarse statistics (e.g., histograms) based on historical data or collected in an initial pass, to compute the *best partitioning*

44

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

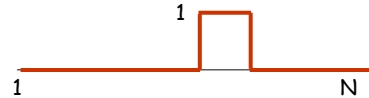
Other Applications of AMS Sketching

- **General result:** Streaming (ϵ, δ) estimation of “large” inner products using AMS sketches

- Other streaming inner products of interest

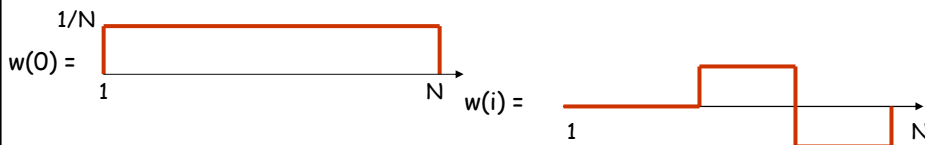
- Top-k frequencies [Charikar et al.'02]

- Item frequency = $\langle f, \text{“unit_pulse”} \rangle$



- Large wavelet coefficients [Gilbert et al.'01], [Cormode et al.'06]

- $\text{Coeff}(i) = \langle f, w(i) \rangle$, where $w(i)$ = i-th wavelet basis vector



47

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

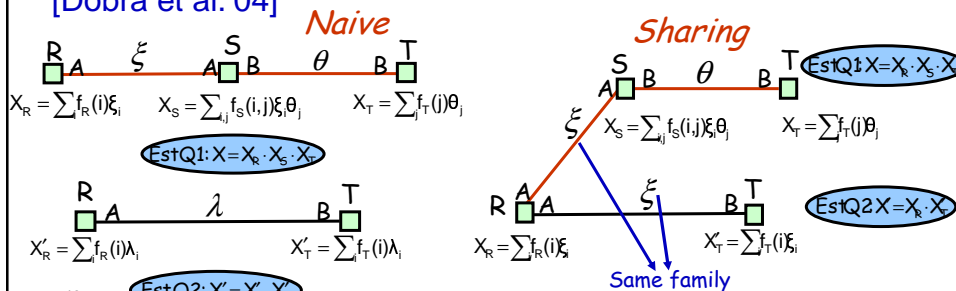
YAHOO! RESEARCH

Recent Results on Stream Joins

- Better accuracy using “skimmed sketches” [Ganguly et al.'04]

- “Skim” dense items (i.e., large freqs) from the AMS sketches
- Use the “skimmed” sketch only for sparse elements
- Stronger worst-case guarantees, and much better in practice
 - Same effect as sketch partitioning with *no a priori knowledge!*

- Sharing sketch space/computation among multiple queries [Dobra et al.'04]

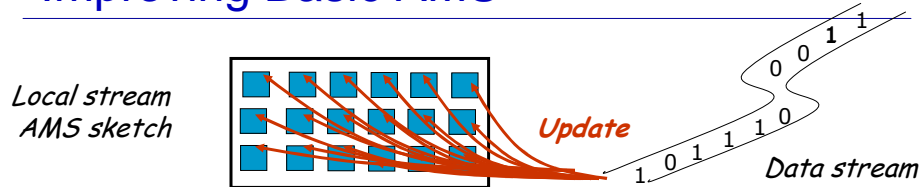


48

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Improving Basic AMS



■ Update time for basic AGMS sketch is $\Omega(|\text{sketch}|)$

■ **BUT...**

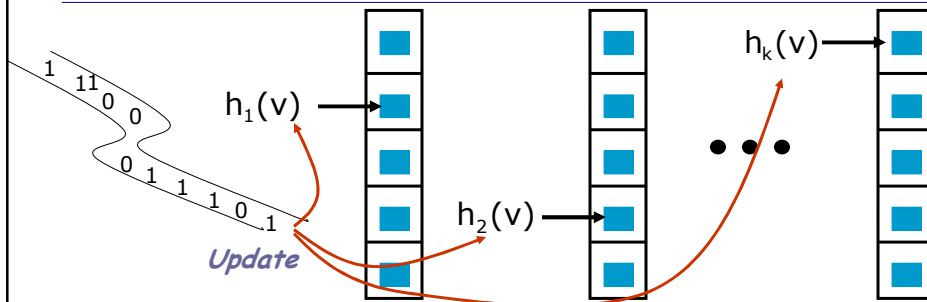
- Sketches can get large – cannot afford to touch every counter for rapid-rate streams!
 - Complex queries, stringent error guarantees, ...
- Sketch size may not be the limiting factor (PCs with GBs of RAM)

49

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt



The Fast AMS Sketch [Cormode, Garofalakis'05]



■ **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)!!

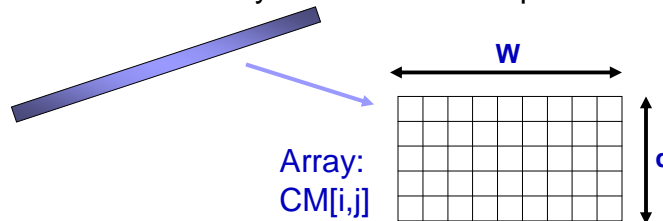
50

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt



Count-Min Sketch [Cormode, Muthukrishnan'04]

- Simple sketch idea, can be used for as the basis of many different stream mining tasks
 - Join aggregates, range queries, moments, ...
- Model input stream as a vector A of dimension N
- Creates a small summary as an array of $w \times d$ in size
- Use d hash functions to map vector entries to $[1..w]$
- Works on arrivals only and arrivals & departures streams

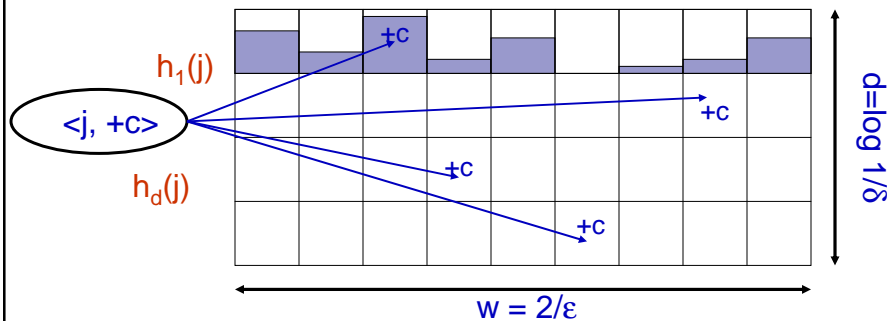


51

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

CM Sketch Structure



- Each entry in input vector $A[j]$ is mapped to one bucket per row
 - $h()$'s are *pairwise independent*
- Merge two sketches by entry-wise summation
- Estimate $A[j]$ by taking $\min_k \{ CM[k, h_k(j)] \}$

52

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

CM Sketch Guarantees

- [Cormode, Muthukrishnan'04] CM sketch guarantees approximation error on point queries less than $\epsilon \|A\|_1$ in space $O(1/\epsilon \log 1/\delta)$
 - Probability of more error is less than $1-\delta$
 - Similar guarantees for range queries, quantiles, join size,...
- Hints
 - Counts are *biased (overestimates)* due to collisions
 - Limit the expected amount of extra “mass” at each bucket? (Use Markov)
 - Use Chernoff-like argument to boost the confidence for the $\min\{\}$ estimate
 - Based on independence of row hashes

53

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

CM Sketch Analysis

Estimate $A'[j] = \min_k \{ CM[k, h_k(j)] \}$

- Analysis: In k 'th row, $CM[k, h_k(j)] = A[j] + X_{k,j}$
 - $X_{k,j} = \sum A[i] \mid h_k(i) = h_k(j)$
 - $E[X_{k,j}] = \sum A[i] * \Pr[h_k(i)=h_k(j)] \leq (\epsilon/2) * \sum A[i] = \epsilon \|A\|_1 / 2$ (pairwise independence of h)
 - $\Pr[X_{k,j} \geq \epsilon \|A\|_1] = \Pr[X_{k,j} \geq 2E[X_{k,j}]] \leq 1/2$ by Markov inequality
- So, $\Pr[A'[j] \geq A[j] + \epsilon \|A\|_1] = \Pr[\forall k. X_{k,j} > \epsilon \|A\|_1] \leq 1/2^{\log 1/\delta} = \delta$
- Final result: with certainty $A[j] \leq A'[j]$ and with probability at least $1-\delta$, $A'[j] < A[j] + \epsilon \|A\|_1$
- Q: How do CM sketch guarantees compare to AMS??

54

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Distinct Value Estimation

- **Problem:** Find the *number of distinct values* in a stream of values with domain $[1, \dots, N]$
 - 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	2	5	3	2	1	7	5	1	2	3	7
---	---	---	---	---	---	---	---	---	---	---	---

Number of distinct values: 5
- Hard problem for random sampling! [Charikar et al.'00]
 - 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!
- AMS and CM only good for *multiset semantics*

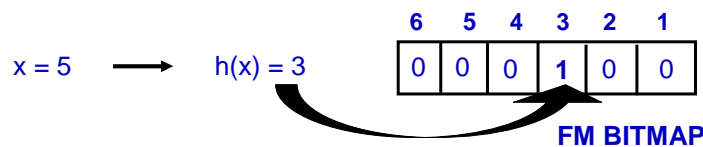
55

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

FM Sketch [Flajolet, Martin'85]

- Estimates number of distinct inputs (**count distinct**)
- Uses hash function mapping input items to i with prob 2^{-i}
 - i.e. $\Pr[h(x) = 1] = 1/2$, $\Pr[h(x) = 2] = 1/4$, $\Pr[h(x)=3] = 1/8 \dots$
 - Easy to construct $h()$ from a uniform hash function by counting trailing zeros
- Maintain FM Sketch = bitmap array of $L = \log N$ bits
 - Initialize bitmap to all 0s
 - For each incoming value x , set $FM[h(x)] = 1$



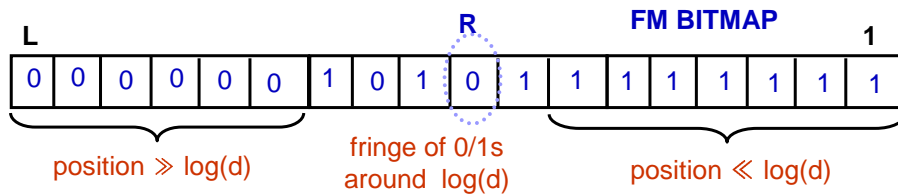
56

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

FM Sketch Analysis

- If d distinct values, expect $d/2$ map to FM[1], $d/4$ to FM[2]...



- Let R = position of rightmost zero in FM, indicator of $\log(d)$
- Basic estimate $d = c2^R$ for scaling constant $c \approx 1.3$
- Average many copies (different hash fns) improves accuracy

57

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

FM Sketch Properties

- With $O(1/\epsilon^2 \log 1/\delta)$ copies, get $(1 \pm \epsilon)$ accuracy with probability at least $1 - \delta$ [Bar-Yossef et al'02], [Ganguly et al.'04]
 - 10 copies gets $\approx 30\%$ error, 100 copies $< 10\%$ error
- *Delete-Proof*: Use counters instead of bits in sketch locations
 - +1 for inserts, -1 for deletes
- *Composable*: Component-wise OR/add distributed sketches together

$$\begin{array}{cccccc} 6 & 5 & 4 & 3 & 2 & 1 \\ \hline 0 & 0 & 1 & 0 & 1 & 1 \end{array} + \begin{array}{cccccc} 6 & 5 & 4 & 3 & 2 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 1 \end{array} = \begin{array}{cccccc} 6 & 5 & 4 & 3 & 2 & 1 \\ \hline 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

- Estimate $|S_1 \cup \dots \cup S_k|$ = set union cardinality

58

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Generalization: Distinct Values Queries

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

Template

```
SELECT COUNT( DISTINCT o_custkey )  
FROM orders  
WHERE o_orderdate >= '2008-01-01'
```

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?*

59

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Distinct Sampling [Gibbons'01]

Key Ideas

- Use FM-like technique to collect a specially-tailored sample over the *distinct values in the stream*
 - **Use hash function 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

60

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Building a Distinct Sample [Gibbons'01]

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

```
DistinctSampling( B , r )
// B = space bound, r = tuple-reservoir size for each distinct value
level = 1; S =  $\phi$ 
for each new tuple t do
  let x = value of DISTINCT target attribute in t
  if  $h(x) \geq \text{level}$  then // x belongs in the distinct sample
    use t to update the reservoir sample of tuples for x
  if  $|S| \geq B$  then // out of space
    evict from S all tuples with  $h(\text{target-attribute-value}) = \text{level}$ 
  set level = level + 1
```

61

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Using the Distinct Sample

- If $\text{level} = k$ for our sample, then we have selected all distinct values x such that $h(x) \geq k$
 - $\text{Prob}[h(x) \geq k] = 1/2^{k-1}$
 - By $h()$'s randomizing properties, we have uniformly sampled a $2^{-(k-1)}$ fraction of the distinct values in our stream
- **Query Answering:** Run distinct-values query on the distinct sample and scale the result up by 2^{k-1}
- **Distinct-value estimation:** Guarantee ϵ relative error with probability $\geq 1 - \delta$ using $O(\log(1/\delta)/\epsilon^2)$ 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

Our sampling rate!

62

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketching and Sampling Summary

- Sampling and sketching ideas are at the heart of many stream mining algorithms
 - Moments/join aggregates, histograms, wavelets, top-k, frequent items, other mining problems, ...
- A sample is a quite **general representative** of the data set; sketches tend to be **specific to a particular purpose**
 - FM sketch for count distinct, AMS sketch for joins/ L_2 estimation, ...
- Traditional sampling does not work in the **turnstile (arrivals & departures) model**
 - BUT... see recent generalizations of distinct sampling [Ganguly et al.'04], [Cormode et al.'05]; as well as [Gemulla et al.'08]

63

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Practicality

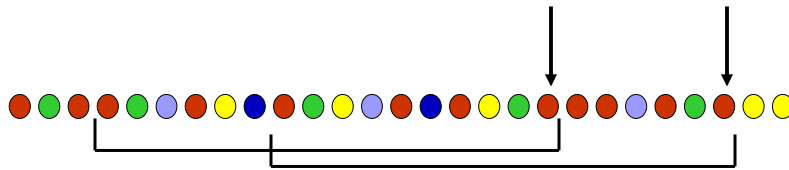
- Algorithms discussed here are quite simple and very fast
 - Sketches can easily process millions of updates per second on standard hardware
 - Limiting factor in practice is often I/O related
- Implemented in several practical systems:
 - AT&T's Gigascope system on live network streams
 - Sprint's CMON system on live streams
 - Google's log analysis
- Sample implementations available on the web
 - <http://www.cs.rutgers.edu/~muthu/massdal-code-index.html>
 - or web search for 'massdal'

64

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

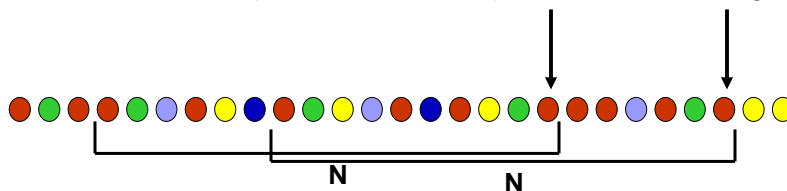
The Sliding Window Model



Sliding Window Streaming Model

■ Model

- At every time t , a data record arrives
- The record "expires" at time $t+N$ (N is the window length)



■ When is it useful?

- Make decisions based on "recently observed" data
- Stock data
- Sensor networks

Time in Data Stream Models

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

- Function $f(X,t,NOW)$
 - Input at time t : $f(X_1,1,t), f(X_2,2,t), f(X_3,3,t), \dots, f(X_t,t,t)$
 - Input at time $t+1$: $f(X_1,1,t+1), f(X_2,2,t+1), f(X_3,3,t+1), \dots, f(X_{t+1},t+1,t+1)$
- Full history: $f == \text{identity}$
- Partial history: Decay
 - **Exponential decay**: $f(X,t, NOW) = 2^{-(NOW-t)*X}$
 - Input at time t : $2^{-(t-1)*X_1}, 2^{-(t-2)*X_2}, \dots, \frac{1}{2} * X_{t-1}, X_t$
 - Input at time $t+1$: $2^{-t*X_1}, 2^{-(t-1)*X_2}, \dots, \frac{1}{4} * X_{t-1}, \frac{1}{2} * X_t, X_{t+1}$
 - **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 : $X_1, X_2, X_3, \dots, X_t$
 - Input at time $t+1$: $X_2, X_3, \dots, X_t, X_{t+1}$

67

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Simple Statistics over Sliding Windows

- Bitstream input – Count the number of ones [Datar et al.'02]
 - Exact solution: $\Theta(N)$ bits
 - Algorithm **BasicCounting**
 - $(1 \pm \epsilon)$ relative error approximation
 - Space: $O(1/\epsilon (\log^2 N))$ bits
 - Time: $O(\log N)$ worst case, $O(1)$ amortized per record
 - Lower Bound:
 - Space: $\Omega(1/\epsilon (\log^2 N))$ bits

68

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

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): $C_m, C_{m-1}, \dots, C_2, C_1$
- Absolute error $\leq C_m/2$
- Answer $\geq C_{m-1} + \dots + C_2 + C_1 + 1$.

■ Relative error $\leq C_m / (2(C_{m-1} + \dots + C_2 + C_1 + 1))$

- Maintain: $C_m / (2(C_{m-1} + \dots + C_2 + C_1 + 1)) \leq \epsilon$ ($=1/k$)

69

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Naïve: Equi-Width Histograms

■ Goal: Maintain $C_m/2 \leq \epsilon (C_{m-1} + \dots + C_2 + C_1 + 1)$

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 $C_i/2 \leq \epsilon (C_{i-1} + \dots + C_2 + C_1 + 1)$

70

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

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 $C_i=2^i$, then
$$C_{i-1}+\dots+C_2+C_1+1 \geq k/2 * (\Sigma(1+2+4+\dots+2^{i-1})) \geq k*2^i / 2 \geq k/2 * C_i$$
 - Setting $k = 1/\epsilon$ implies the required error guarantee!

71

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Space Complexity

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

72

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

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: $\text{TOTAL} - (\text{LAST}/2)$

73

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

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

74

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

The Power of EHs

- Counter for N items = $O(\log N)$ space
- EH = ϵ -approximate counter over sliding window of N items that requires $O(1/\epsilon * \log^2 N)$ space
 - $O(1/\epsilon \log N)$ penalty for (approximate) sliding-window counting
 - *Deterministic error guarantee!*
- Can plug-in 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

75

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Tutorial Outline

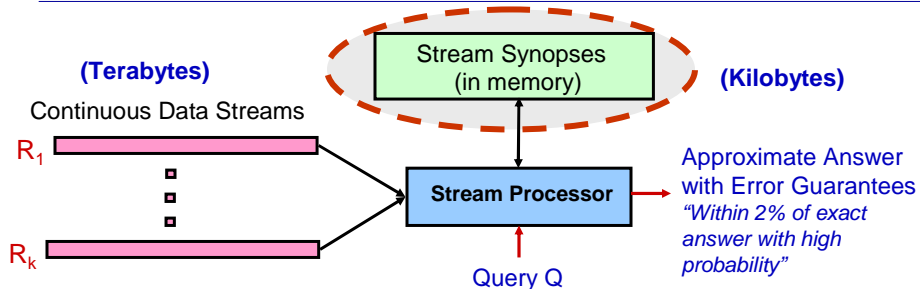
- Motivation & Streaming Applications
- Centralized Stream Processing
- **Distributed Stream Processing**
 - Basic model and problem setup
 - One-shot distributed-stream querying
 - Continuous distributed-stream tracking
 - Probabilistic distributed data acquisition
- Open Problems & Future Directions
- Conclusions

76

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Data-Stream Algorithmics Model



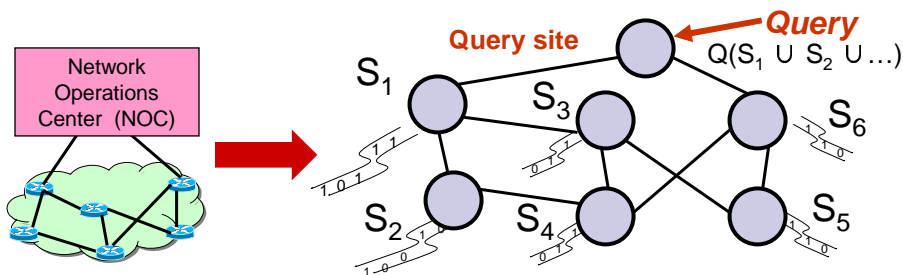
- *Approximate answers*– e.g. trend analysis, anomaly detection
- Requirements for stream synopses
 - *Single Pass*: Each record is examined at most once
 - *Small Space*: Log or polylog in data stream size
 - *Small-time*: Low per-record processing time (maintain synopses)
 - Also: *delete-proof*, *composable*, ...

77

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed Streams Model



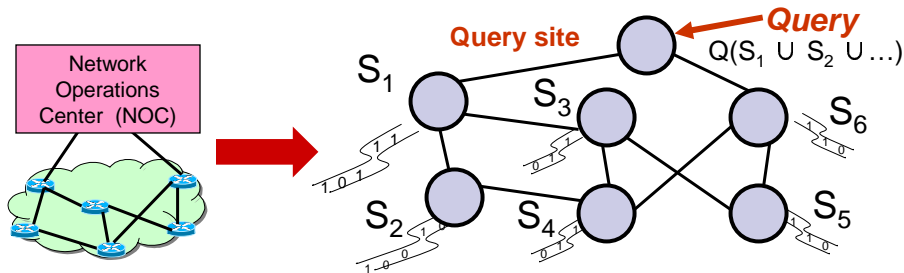
- Large-scale querying/monitoring: *Inherently distributed!*
 - Streams physically distributed across remote sites
E.g., stream of UDP packets through subset of edge routers
- *Challenge is "holistic" querying/monitoring*
 - Queries over the *union of distributed streams* $Q(S_1 \cup S_2 \cup \dots)$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

78

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed Streams Model



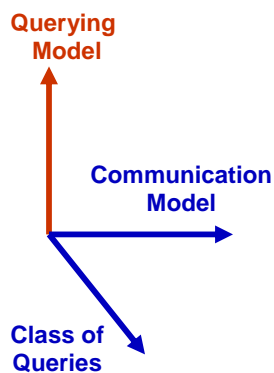
- Need timely, accurate, and efficient query answers
- Additional complexity over centralized data streaming!
- Need space/time- *and communication-efficient* solutions
 - Minimize network overhead
 - Maximize network lifetime (e.g., sensor battery life)
 - Cannot afford to “centralize” all streaming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

79

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed Stream Querying Space



“One-shot” vs. Continuous Querying

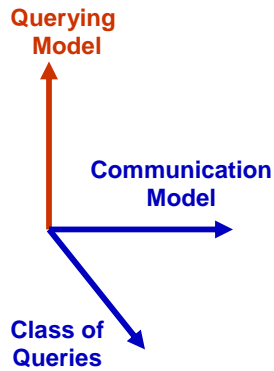
- **One-shot queries:** On-demand “pull” query answer from network
 - One or few rounds of communication
 - Nodes may prepare for a class of queries
- **Continuous queries:** Track/monitor answer at query site *at all times*
 - Detect anomalous/outlier behavior *in (near) real-time*, i.e., “Distributed triggers”
 - Challenge is to minimize communication
 - Use “push-based” techniques
 - May use one-shot algs as subroutines

80

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed Stream Querying Space



Minimizing communication often needs **approximation** and **randomization**

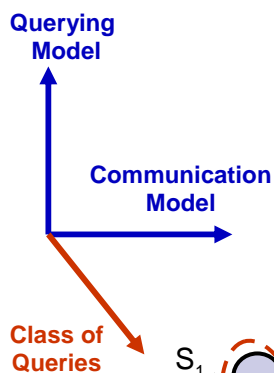
- E.g., Continuously monitor average value
 - Must send every change for exact answer
 - Only need 'significant' changes for approx (def. of "significant" specifies an algorithm)
- Probability sometimes vital to reduce communication
 - **count distinct** in one shot model needs randomness
 - Else **must** send complete data

81

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

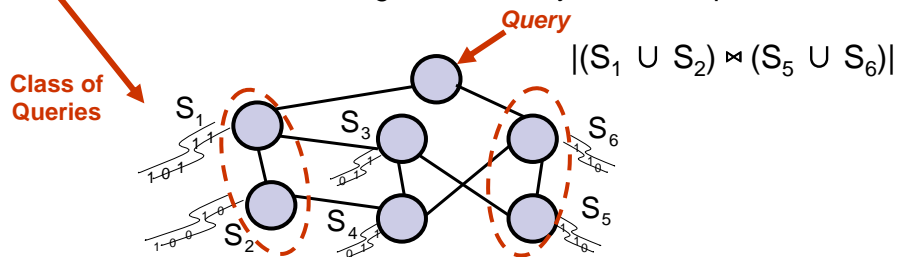
YAHOO! RESEARCH

Distributed Stream Querying Space



Class of Queries of Interest

- Simple algebraic vs. holistic aggregates
 - E.g., **count**/**max** vs. quantiles/top-k
- Duplicate-sensitive vs. duplicate-insensitive
 - "Bag" vs. "set" semantics
- Complex correlation queries
 - E.g., distributed joins, set expressions, ...

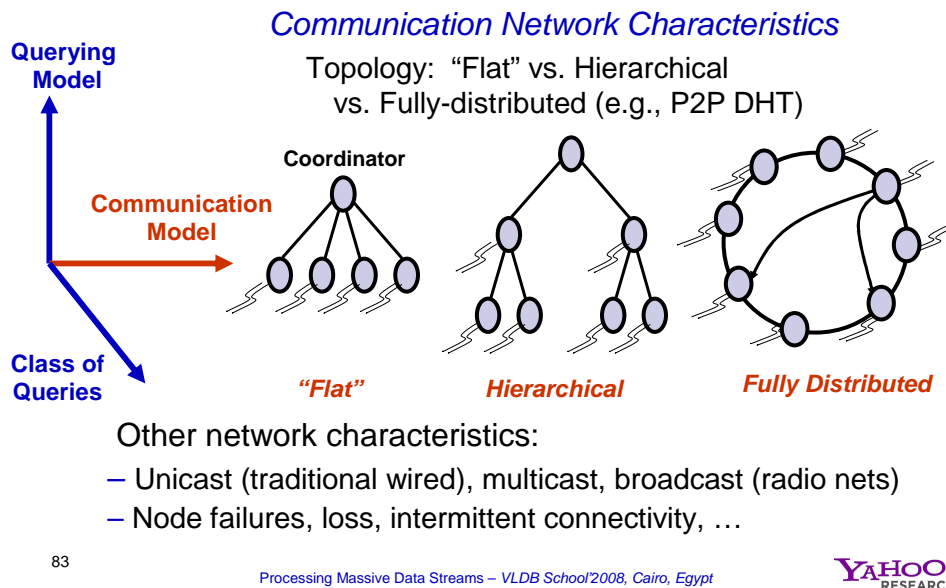


82

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed Stream Querying Space



Tutorial Outline

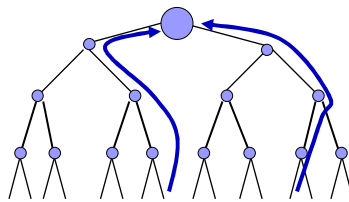
- Motivation & Streaming Applications
- Centralized Stream Processing
- Distributed Stream Processing
 - One-shot distributed-stream querying
 - Tree-based aggregation
 - Robustness and loss
 - Decentralized computation and gossiping
- Open Problems & Future Directions
- Conclusions

84

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

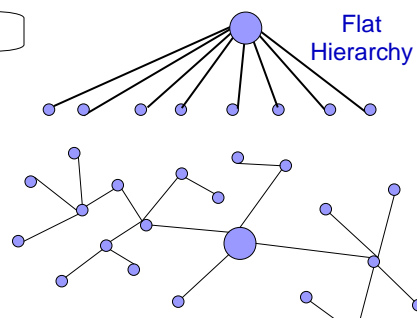
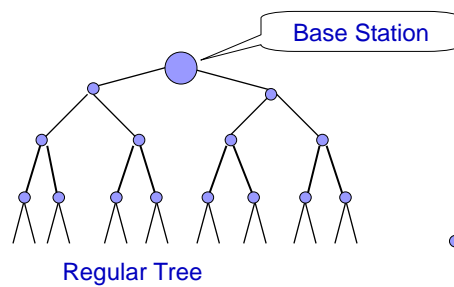
YAHOO! RESEARCH

Tree Based Aggregation



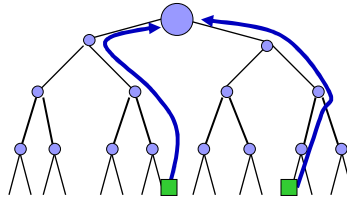
Network Trees

- Tree structured networks are a basic primitive
 - Much work in, e.g., sensor nets on building communication trees
 - We assume that tree has been built, focus on issues with a fixed tree



Computation in Trees

- Goal is for root to compute a function of data at leaves
- Trivial solution: push all data up tree and compute at base station
 - Strains nodes near root: batteries drain, disconnecting network
 - Very wasteful: no attempt at saving communication
- Can do much better by “In-network” query processing
 - Simple example: computing **max**
 - Each node hears from all children, computes max and sends to parent (each node sends only one item)



87

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Efficient In-network Computation

- What are aggregates of interest?
 - SQL Primitives: **min, max, sum, count, avg**
 - More complex: **count distinct**, point & range queries, quantiles, wavelets, histograms, sample
 - Data mining: association rules, clusterings etc.
- Some aggregates are easy – e.g., SQL primitives
- Can set up a formal framework for in-network aggregation

88

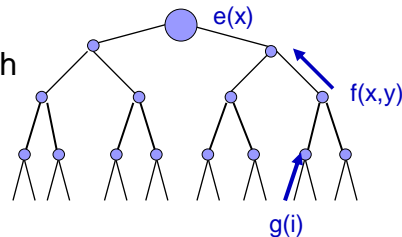
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Generate, Fuse, Evaluate Framework

- Abstract in-network aggregation. Define functions:
 - **Generate**, $g(i)$: take input, produce summary (at leaves)
 - **Fusion**, $f(x,y)$: merge two summaries (at internal nodes)
 - **Evaluate**, $e(x)$: output result (at root)
- E.g. **max**: $g(i) = i$ $f(x,y) = \max(x,y)$ $e(x) = x$
- E.g. **avg**: $g(i) = (i,1)$ $f((i,j),(k,l)) = (i+k,j+l)$ $e(i,j) = i/j$

- Can specify any function with
 $g(i) = \{i\}$, $f(x,y) = x \cup y$
 Want to bound $|f(x,y)|$



89

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Classification of Aggregates

- Different properties of aggregates
 (from TAG paper [Madden et al '02])
 - **Duplicate sensitive** – is answer same if multiple identical values are reported?
 - **Example or summary** – is result some value from input (**max**) or a small summary over the input (**sum**)
 - **Monotonicity** – is $F(X \cup Y)$ monotonic compared to $F(X)$ and $F(Y)$ (affects push down of selections)
 - **Partial state** – are $|g(x)|$, $|f(x,y)|$ constant size, or growing?
 Is the aggregate *algebraic*, or *holistic*?

90

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Classification of some aggregates

	Duplicate Sensitive	Example or summary	Monotonic	Partial State
min, max	No	Example	Yes	algebraic
sum, count	Yes	Summary	Yes	algebraic
average	Yes	Summary	No	algebraic
median, quantiles	Yes	Example	No	holistic
count distinct	No	Summary	Yes	holistic
sample	Yes	Example(s)	No	algebraic?
histogram	Yes	Summary	No	holistic

adapted from [Madden et al.'02]

91

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Cost of Different Aggregates

Slide adapted from <http://db.lcs.mit.edu/madden/html/jobtalk3.ppt>

Simulation Results

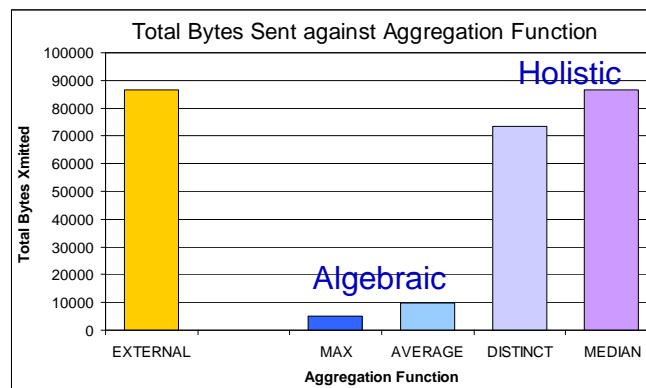
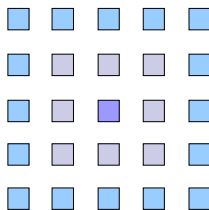
2500 Nodes

50x50 Grid

Depth = ~10

Neighbors = ~20

Uniform Dist.



92

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Holistic Aggregates

- Holistic aggregates need the whole input to compute (no summary suffices)
 - E.g., **count distinct**, need to remember all distinct items to tell if new item is distinct or not
- So, focus on **approximating** aggregates to limit data sent
 - Adopt ideas from sampling, data reduction, streams, etc.
- Many techniques for in-network aggregate approximation:
 - Sketch summaries
 - Other mergeable summaries
 - Building uniform samples, etc...

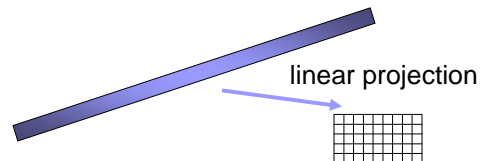
93

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketch Summaries

- Sketch summaries are typically pseudo-random linear projections of data. Fits generate/fuse/evaluate model:
 - Suppose input is vectors x_i and aggregate is $F(\sum_i x_i)$
 - Sketch of x_i , $g(x_i)$, is a matrix product Mx_i
 - Combination of two sketches is their summation:
 $f(g(x_i), g(x_j)) = M(x_i + x_j) = Mx_i + Mx_j = g(x_i) + g(x_j)$
 - Extraction function $e()$ depends on sketch, different sketches allow approximation of different aggregates



94

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketch Summary

- CM sketch guarantees approximation error on point queries less than $\epsilon \|x\|_1$ in size $O(1/\epsilon \log 1/\delta)$
 - Probability of more error is less than $1-\delta$
 - Similar guarantees for range queries, quantiles, join size
- AMS sketches approximate self-join and join size with error less than $\epsilon \|x\|_2 \|y\|_2$ in size $O(1/\epsilon^2 \log 1/\delta)$
 - [Alon, Matias, Szegedy '96, Alon, Gibbons, Matias, Szegedy '99]
- FM sketches approximate number of distinct items ($\|x\|_0$) with error less than $\epsilon \|x\|_0$ in size $O(1/\epsilon^2 \log 1/\delta)$
- Bloom filters: compactly encode sets in sketch like fashion

95

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other approaches: Careful Merging

- **Approach 1. Careful merging of summaries**
 - Small summaries of a large amount of data at each site
 - E.g., Greenwald-Khanna algorithm (GK) keeps a small data structure to allow quantile queries to be answered
 - Can sometimes carefully merge summaries up the tree
Problem: if not done properly, the merged summaries can grow very large as they approach root
 - Balance final quality of answer against number of merges by decreasing approximation quality (*precision gradient*)
 - See [Greenwald, Khanna '04; Manjhi et al.'05; Manjhi, Nath, Gibbons '05]

96

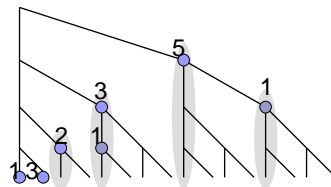
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other approaches: Domain Aware

■ Approach 2. Domain-aware Summaries

- Each site sees information drawn from discrete domain $[1 \dots N]$ – e.g., for IP addresses, $N = 2^{32}$
- Build summaries by imposing tree-structure on domain and keeping counts of nodes representing subtrees
- [Agrawal et al '04] show $O(1/\epsilon \log N)$ size summary for quantiles and range & point queries
- Can merge repeatedly without increasing error or summary size



97

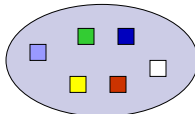
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other approaches: Random Samples

■ Approach 3. Uniform random samples

- As in centralized databases, a uniform random sample of size $O(1/\epsilon^2 \log 1/\delta)$ answers many queries
- Can collect a random sample of data from each node, and merge up the tree (will show algorithms later)
- Works for frequent items, quantile queries, histograms
- No good for count distinct, min, max, wavelets...



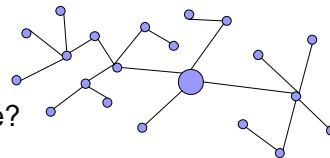
98

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Thoughts on Tree Aggregation

- Some methods too heavyweight for today's sensor nets, but as technology improves may soon be appropriate
- Most are well suited for, e.g., **wired network monitoring**
 - Trees in wired networks often treated as flat, i.e. send directly to root without modification along the way
- Techniques are fairly well-developed owing to work on data reduction/summarization and streams
- Open problems and challenges:
 - Improve size of larger summaries
 - Avoid randomized methods?
Or use randomness to reduce size?

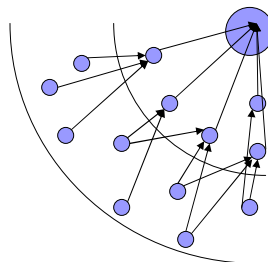


99

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

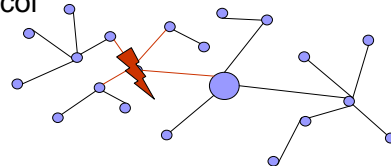
YAHOO!
RESEARCH

Robustness and Loss



Unreliability

- Tree aggregation techniques assumed a reliable network
 - we assumed no node failure, nor loss of any message
- Failure can dramatically affect the computation
 - E.g., **sum** – if a node near the root fails, then a whole subtree may be lost
- Clearly a particular problem in sensor networks
 - If messages are lost, maybe can detect and resend
 - If a node fails, may need to rebuild the whole tree and re-run protocol
 - Need to detect the failure, could cause high uncertainty



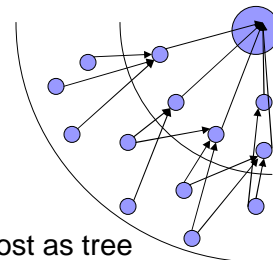
101

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sensor Network Issues

- Sensor nets typically based on radio communication
 - So broadcast (within range) cost the same as unicast
 - Use multi-path routing: improved reliability, reduced impact of failures, less need to repeat messages
- E.g., computation of **max**
 - structure network into rings of nodes in equal hop count from root
 - listen to all messages from ring below, then send max of all values heard
 - converges quickly, high path diversity
 - each node sends only once, so same cost as tree



102

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Order and Duplicate Insensitivity

- It works because **max** is *Order and Duplicate Insensitive (ODI)* [Nath et al.'04]
- Make use of the same $e()$, $f()$, $g()$ framework as before
- Can prove correct if $e()$, $f()$, $g()$ satisfy properties:
 - g gives same output for duplicates: $i=j \Rightarrow g(i) = g(j)$
 - f is associative and commutative:
 $f(x,y) = f(y,x)$; $f(x,f(y,z)) = f(f(x,y),z)$
 - f is same-synopsis idempotent: $f(x,x) = x$
- Easy to check **min**, **max** satisfy these requirements, **sum** does not

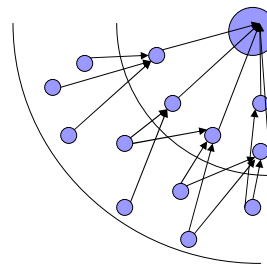
103

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Applying ODI idea

- Only **max** and **min** seem to be “naturally” ODI
- How to make ODI summaries for other aggregates?
- Will make use of duplicate insensitive primitives:
 - Flajolet-Martin Sketch (FM)
 - Min-wise hashing
 - Random labeling
 - Bloom Filter



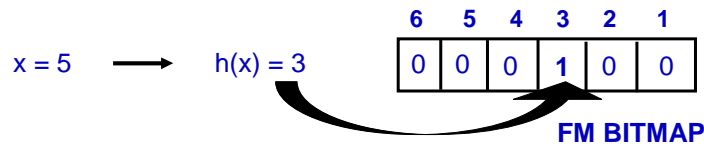
104

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

FM Sketch [Flajolet, Martin'85]

- Estimates number of distinct inputs (**count distinct**)
- Uses hash function mapping input items to i with prob 2^{-i}
 - i.e. $\Pr[h(x) = 1] = 1/2$, $\Pr[h(x) = 2] = 1/4$, $\Pr[h(x)=3] = 1/8 \dots$
 - Easy to construct $h()$ from a uniform hash function by counting trailing zeros
- Maintain FM Sketch = bitmap array of $L = \log N$ bits
 - Initialize bitmap to all 0s
 - For each incoming value x , set $FM[h(x)] = 1$

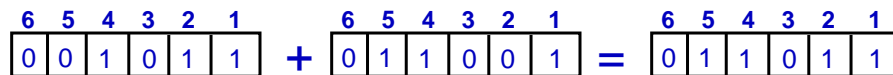


105

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

FM Sketch – ODI Properties



- Fits into the Generate, Fuse, Evaluate framework.
 - Can fuse multiple FM summaries (with same hash $h()$): take bitwise-OR of the summaries
- With $O(1/\epsilon^2 \log 1/\delta)$ copies, get $(1 \pm \epsilon)$ accuracy with probability at least $1 - \delta$
 - 10 copies gets $\approx 30\%$ error, 100 copies $< 10\%$ error
 - Can pack FM into e.g., 32 bits. Assume $h()$ is known to all.
- Similar ideas used in [Gibbons, Tirthapura '01]
 - improves time cost to create summary, simplifies analysis

106

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

FM within ODI

- What if we want to count, not count distinct?
 - E.g., each site i has a count c_i , we want $\sum_i c_i$
 - Tag each item with site ID, write in unary: $(i,1), (i,2)\dots (i,c_i)$
 - Run FM on the modified input, and run ODI protocol
- What if counts are large?
 - Writing in unary might be too slow, need to make efficient
 - [Considine et al.'05]: simulate a random variable that tells which entries in sketch are set
 - [Aduri, Tirthapura '05]: allow range updates, treat (i,c_i) as range.

107

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other applications of FM in ODI

- Can take sketches and other summaries and make them ODI by replacing counters with FM sketches
 - CM sketch + FM sketch = CMFM, ODI point queries etc.
[Cormode, Muthukrishnan '05]
 - Q-digest + FM sketch = ODI quantiles
[Hadjieleftheriou, Byers, Kollios '05]
 - Counts and sums
[Nath et al.'04, Considine et al.'05]

6	5	4	3	2	1
0	1	1	0	1	1

108

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Combining ODI and Tree

- *Tributaries and Deltas* idea [Manjhi, Nath, Gibbons '05]

- Combine small synopsis of tree-based aggregation with reliability of ODI

- Run tree synopsis at edge of network, where connectivity is limited (tributary)
- Convert to ODI summary in dense core of network (delta)
- Adjust crossover point adaptively

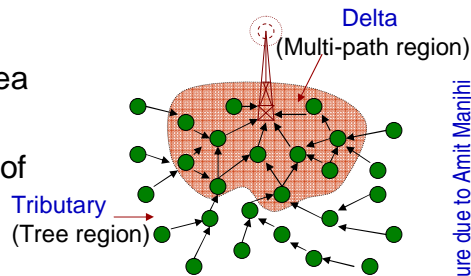


Figure due to Amit Manjhi

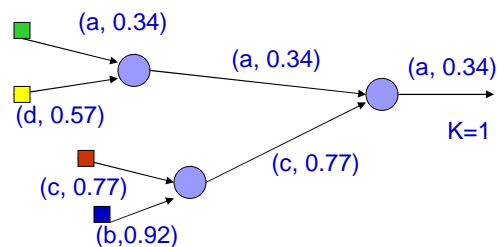
109

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Random Samples

- Suppose each node has a (multi)set of items.
- How to find a random sample of the union of all sets?
- Use a “random tagging” trick [Nath et al.'05]:
 - For each item, attach a random label in range $[0 \dots 1]$
 - Pick the items with the K smallest labels to send
 - Merge all received items, and pick K smallest labels



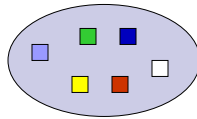
110

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Uniform Random Samples

- Result at the coordinator:
 - A sample of size K items from the input
 - Can show that the sample is chosen uniformly at random without replacement (could make “with replacement”)
- Related to **min-wise hashing**
 - Suppose we want to sample from distinct items
 - Then replace random tag with hash value on item name
 - Result: uniform sample from **set** of present items
- Sample can be used for quantiles, frequent items, etc.



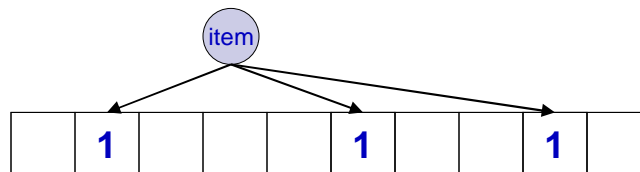
111

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Bloom Filters

- Bloom filters compactly encode set membership
 - k hash functions map items to bit vector k times
 - Set all k entries to **1** to indicate item is present
 - Can lookup items, store set of size n in $\sim 2n$ bits



- Bloom filters are ODI, and merge like FM sketches

112

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Open Questions and Extensions

- Characterize all queries – can everything be made ODI with small summaries?
- How practical for different sensor systems?
 - Few FM sketches are very small (10s of bytes)
 - Sketch with FMs for counters grow large (100s of KBs)
 - What about the computational cost for sensors?
- Amount of randomness required, and implicit coordination needed to agree hash functions, etc.?
- Other implicit requirements: unique sensor IDs?

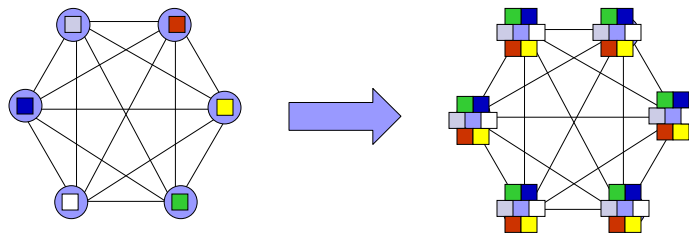
6	5	4	3	2	1
0	1	1	0	1	1

113

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

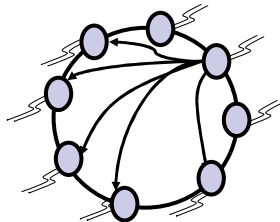
YAHOO!
RESEARCH

Decentralized Computation and Gossiping



Decentralized Computations

- All methods so far have a single point of failure: if the base station (root) dies, everything collapses
- An alternative is **Decentralized Computation**
 - Everyone participates in computation, all get the result
 - Somewhat resilient to failures / departures
- Initially, assume anyone can talk to anyone else directly



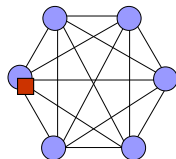
115

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Gossiping

- “Uniform Gossiping” is a well-studied protocol for spreading information
 - I know a secret, I tell two friends, who tell two friends ...
 - Formally, each round, everyone who knows the data sends it to one of the n participants chosen at random
 - After $O(\log n)$ rounds, all n participants know the information (with high probability) [Pittel 1987]



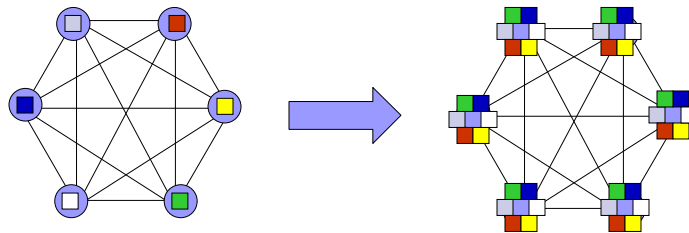
116

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Aggregate Computation via Gossip

- Naïve approach: use uniform gossip to share all the data, then everyone can compute the result.
 - Slightly different situation: gossiping to exchange n secrets
 - Need to store all results so far to avoid double counting
 - Messages grow large: end up sending whole input around



117

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

ODI Gossiping

- If we have an ODI summary, we can gossip with this
 - When new summary received, merge with current summary
 - ODI properties ensure repeated merging stays accurate
- Number of messages required is same as uniform gossip
 - After $O(\log n)$ rounds everyone knows the merged summary
 - Message size and storage space is a single summary
 - $O(n \log n)$ messages in total
 - So, this works for FM, FM-based sketches, samples, etc.

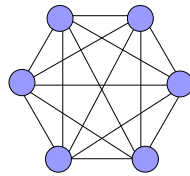
118

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Aggregate Gossiping

- ODI gossiping doesn't always work
 - May be too heavyweight for really restricted devices
 - Summaries may be too large in some cases
- An alternate approach due to [Kempe et al. '03]
 - A novel way to avoid double counting: split up the counts and use "conservation of mass"



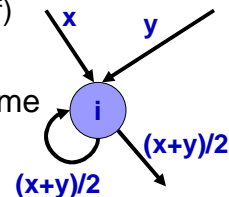
119

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Push-Sum

- Setting: all n participants have a value, want to compute average
- Define "Push-Sum" protocol
 - In round t , node i receives set of $(\text{sum}_j^{t-1}, \text{count}_j^{t-1})$ pairs
 - Compute $\text{sum}_i^t = \sum_j \text{sum}_j^{t-1}$, $\text{count}_i^t = \sum_j \text{count}_j$
 - Pick k uniformly from other nodes
 - Send $(\frac{1}{2} \text{sum}_i^t, \frac{1}{2} \text{count}_i^t)$ to k and to i (self)
- Round zero: send $(\text{value}, 1)$ to self
- Conservation of counts: $\sum_i \text{sum}_i^t$ stays same
- Estimate $\text{avg} = \text{sum}_i^t / \text{count}_i^t$

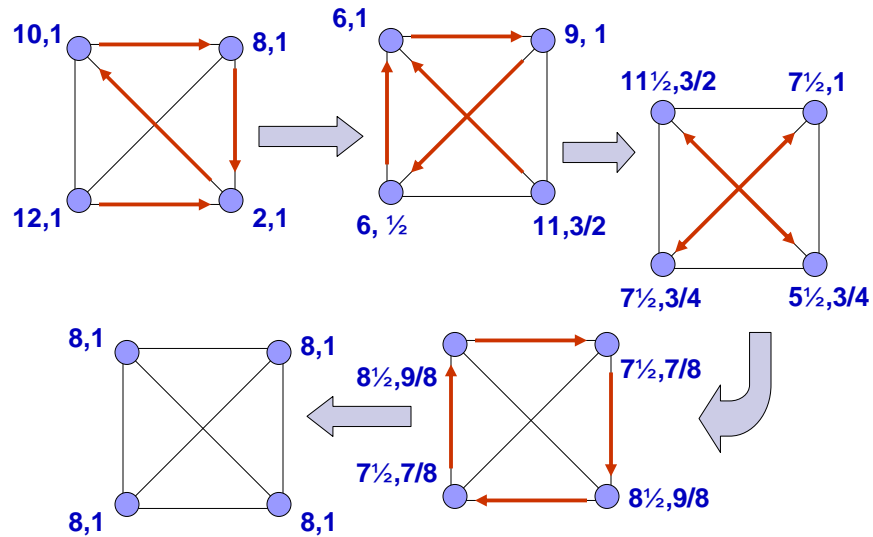


120

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Push-Sum Convergence



121

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Convergence Speed

- Can show that after $O(\log n + \log 1/\epsilon + \log 1/\delta)$ rounds, the protocol converges within ϵ
 - n = number of nodes
 - ϵ = (relative) error
 - δ = failure probability
- Correctness due in large part to conservation of counts
 - Sum of values remains constant throughout
 - (Assuming no loss or failure)

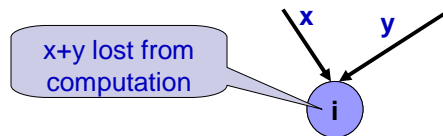
122

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Resilience to Loss and Failures

- Some resilience comes for “free”
 - If node detects message was not delivered, delay 1 round then choose a different target
 - Can show that this only increases number of rounds by a small constant factor, even with many losses
 - Deals with message loss, and “dead” nodes without error
- If a node fails during the protocol, some “mass” is lost, and count conservation does not hold
 - If the mass lost is not too large, error is bounded...



123

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Gossip on Vectors

- Can run **Push-Sum** independently on each entry of vector
- More strongly, generalize to **Push-Vector**:
 - Sum incoming vectors
 - Split sum: half for self, half for randomly chosen target
 - Can prove same conservation and convergence properties
- Generalize to sketches: a sketch is just a vector
 - But ϵ error on a sketch may have different impact on result
 - Require $O(\log n + \log 1/\epsilon + \log 1/\delta)$ rounds as before
 - Only store $O(1)$ sketches per site, send 1 per round

124

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Thoughts and Extensions

- How realistic is complete connectivity assumption?
 - In sensor nets, nodes only see a local subset
 - Variations: spatial gossip ensures nodes hear about local events with high probability [Kempe, Kleinberg, Demers '01]
- Can do better with more structured gossip, but impact of failure is higher [Kashyap et al.'06]
- Is it possible to do better when only a subset of nodes have relevant data and want to know the answer?

125

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Tutorial Outline

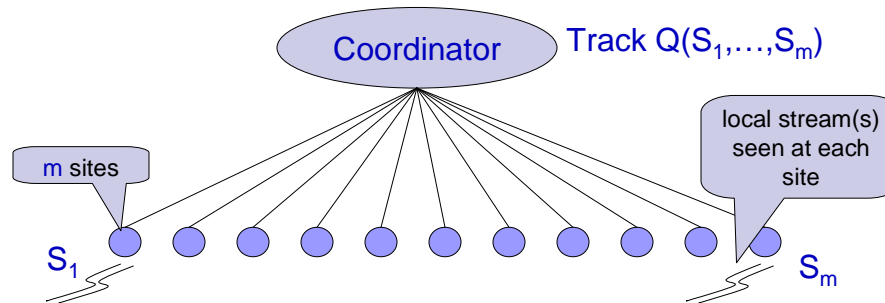
- Motivation & Streaming Applications
- Centralized Stream Processing
- Distributed Stream Processing
 - One-shot distributed-stream querying
 - Continuous distributed-stream tracking
 - Adaptive slack allocation
 - Predictive local-stream models
 - Distributed triggers
- Open Problems & Future Directions
- Conclusions

126

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Continuous Distributed Model



- Other structures possible (e.g., hierarchical)
- Could allow site-site communication, but mostly unneeded
- **Goal:** *Continuously track* (global) query over streams at the coordinator
 - Large-scale network-event monitoring, real-time anomaly/DDoS attack detection, power grid monitoring, ...

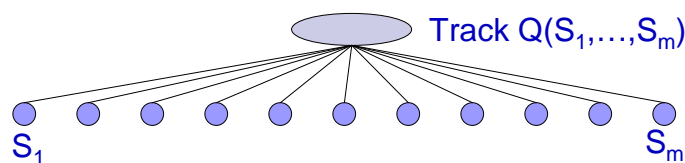
127

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Continuous Distributed Streams

- But... local site streams continuously change!
 - E.g., new readings are made, new data arrives
 - *Assumption:* Changes are somewhat smooth and gradual
- Need to guarantee an answer at the coordinator that is always correct, within some guaranteed accuracy bound
- Naïve solutions must *continuously* centralize all data
 - Enormous communication overhead!



128

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Challenges

- Monitoring is **Continuous...**
 - Real-time tracking, rather than one-shot query/response
- **...Distributed...**
 - Each remote site only observes part of the global stream(s)
 - *Communication constraints*: must minimize monitoring burden
- **...Streaming...**
 - Each site sees a high-speed local data stream and can be resource (CPU/memory) constrained
- **...Holistic...**
 - Challenge is to monitor the *complete global data distribution*
 - Simple aggregates (e.g., aggregate traffic) are easier

129

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

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



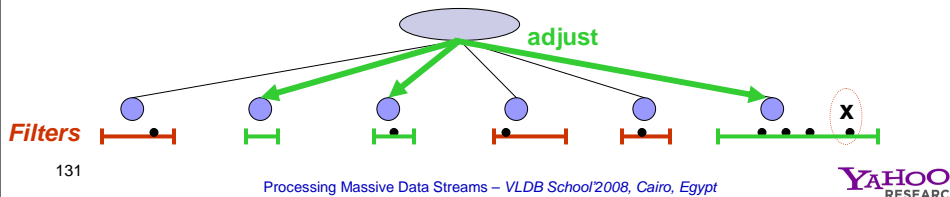
130

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Communication-Efficient Monitoring

- Exact answers are not needed
 - Approximations with accuracy guarantees suffice
 - Tradeoff *accuracy* and *communication/processing cost*
- Key Insight: *"Push-based" in-network 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
 - Coordinator sets/adjusts local filters to guarantee accuracy

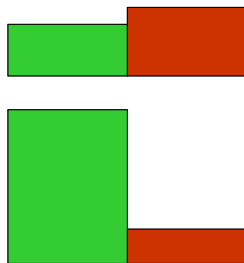


131

Processing Massive Data Streams – VLDB School/2008, Cairo, Egypt

YAHOO!
RESEARCH

Adaptive Slack Allocation



Slack Allocation

- A key idea is **Slack Allocation**
- Because we allow approximation, there is slack: the tolerance for error between computed answer and truth
 - May be absolute: $|Y - \hat{Y}| \leq \epsilon$: slack is ϵ
 - Or relative: $\hat{Y}/Y \leq (1 \pm \epsilon)$: slack is ϵY
- For a given aggregate, show that the slack can be divided between sites
- Will see different slack division heuristics

133

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Top-k Monitoring

- Influential work on monitoring [Babcock, Olston'03]
 - Introduces some basic heuristics for dividing slack
 - Use local offset parameters so that all local distributions look like the global distribution
 - Attempt to fix local slack violations by negotiation with coordinator before a global readjustment
 - Showed that message delay does not affect correctness

Billboard
Top 100

1	Justin Bieber featuring Travis Taylor	1:58	100%
2	Justin Bieber	3:53	95%
3	Ke\$ha	3:53	90%
4	Madonna featuring Nicki Minaj	3:53	85%
5	Justin Bieber	3:53	80%
6	Ke\$ha	3:53	75%

Images from <http://www.billboard.com>

134

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Top-k Scenario

- Each site monitors n objects with local counts $V_{i,j}$
 - item $i \in [n]$
 - site $j \in [m]$
- Values change over time with updates seen at site j
- Global count $V_i = \sum_j V_{i,j}$
- Want to find **topk**, an ϵ -approximation to true top-k set:
 - OK provided $i \in \text{topk}, l \notin \text{topk}, V_i + \epsilon \geq V_l$

gives a little
"wobble room"

135

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Adjustment Factors

- Define a set of 'adjustment factors', $\delta_{i,j}$
 - Make top-k of $V_{i,j} + \delta_{i,j}$ same as top-k of V_i



- Maintain invariants:
 - For item i , adjustment factors sum to zero
 - $\delta_{l,0}$ of non-topk item $l \leq \delta_{i,0} + \epsilon$ of topk item i
 - Invariants and local conditions used to prove correctness

136

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Local Conditions and Resolution

Local Conditions:

At each site j check adjusted topk counts dominate non-topk

$$\begin{array}{c} \delta_{i,j} \\ \hline V_{i,j} \\ i \in \text{topk} \end{array} \geq \begin{array}{c} \delta_{l,j} \\ \hline V_{l,j} \\ l \notin \text{topk} \end{array}$$

If any local condition violated at site j , resolution is triggered

- Local resolution: site j and coordinator only try to fix
 - Try to “borrow” from $\delta_{i,0}$ and $\delta_{l,0}$ to restore condition
- Global resolution: if local resolution fails, contact all sites
 - Collect all affected $V_{i,j}$ s – i.e., topk plus violated counts
 - Compute slacks for each count, and reallocate (next)
 - Send new adjustment factors $\delta'_{i,j}$, continue

137

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

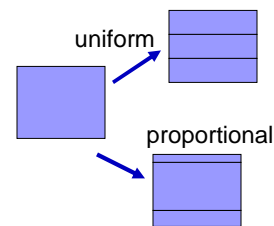
YAHOO! RESEARCH

Slack Division Strategies

- Define “slack” based on current counts and adjustments
- What fraction of slack to keep back for coordinator?
 - $\delta_{i,0} = 0$: No slack left to fix local violations
 - $\delta_{i,0} = 100\%$ of slack: Next violation will be soon
 - Empirical setting: $\delta_{i,0} = 50\%$ of slack when ϵ very small
 $\delta_{i,0} = 0$ when ϵ is large ($\epsilon > V_i/1000$)

- How to divide remainder of slack?

- Uniform: $1/m$ fraction to each site
- Proportional: $V_{i,j}/V_i$ fraction to site j for i



138

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Pros and Cons

- Result has many advantages:
 - Guaranteed correctness within approximation bounds
 - Can show convergence to correct results even with delays
 - Communication reduced by 1 order magnitude (compared to sending $V_{i,j}$ whenever it changes by ϵ/m)
- Disadvantages:
 - Reallocation gets complex: must check $O(km)$ conditions
 - Need $O(n)$ space at each site, $O(mn)$ at coordinator
 - Large ($\approx O(k)$) messages
 - Global resyncs are expensive: m messages to k sites

139

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other Problems: Aggregate Values

- **Problem 1: Single value tracking**
Each site has one value v_i , want to compute $f(v)$, e.g., **sum**
- Allow small bound of uncertainty in answer
 - Divide uncertainty (slack) between sites
 - If new value is outside bounds, re-center on new value
- Naïve solution: allocate equal bounds to all sites
 - Values change at different rates; queries may overlap
- Adaptive filters approach [Olston, Jiang, Widom '03]
 - Shrink all bounds and selectively grow others: moves slack from stable values to unstable ones
 - Base growth on frequency of bounds violation, optimize

140

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other Problems: Set Expressions

■ Problem 2: Set Expression Tracking

$A \cup (B \cap C)$ where A, B, C defined by distributed streams

■ Key ideas [Das et al.'04]:

- Use semantics of set expression: if b arrives in set B , but b already in set A , no need to send
- Use cardinalities: if many copies of b seen already, no need to send if new copy of b arrives or a copy is deleted
- Combine these to create a *charging scheme* for each update: if sum of charges is small, no need to send.
- Optimizing charging is NP-hard, heuristics work well.

141

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other Problems: ODI Aggregates

■ Problem 3: ODI aggregates

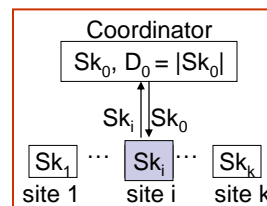
e.g., **count distinct** in continuous distributed model

■ Two important parameters emerge:

- How to divide the slack
- What the site sends to coordinator

■ In [Cormode et al.'06]:

- Share slack evenly: hard to do otherwise for this aggregate
- Sharing sketch of global distribution saves communication
- Better to be lazy: send sketch in reply, don't broadcast



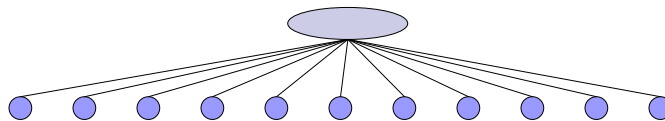
142

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

General Lessons

- Break a global (holistic) aggregate into “safe” local conditions, so local conditions \Rightarrow global correctness
- Set local parameters to help the tracking
- Use the approximation to define slack, divide slack between sites (and the coordinator)
- Avoid global reconciliation as much as possible, try to patch things up locally

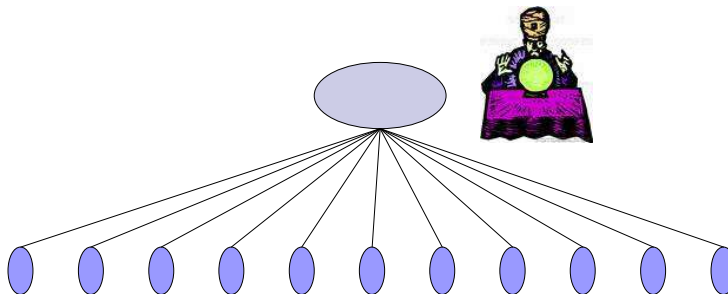


143

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

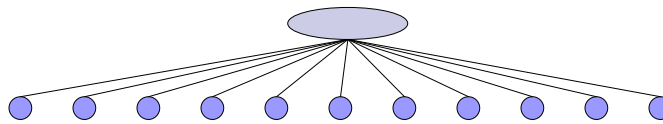
YAHOO!
RESEARCH

Predictive Local-Stream Models



More Sophisticated Local Predictors

- Slack allocation methods use simple “static” prediction
 - Site value implicitly assumed constant since last update
 - No update from site \Rightarrow last update (“predicted” value) is within required slack bounds \Rightarrow global error bound
- *Dynamic, more sophisticated prediction models* for local site behavior?
 - Model complex stream patterns, reduce number of updates to coordinator
 - **But...** more complex to maintain and communicate (to coordinator)

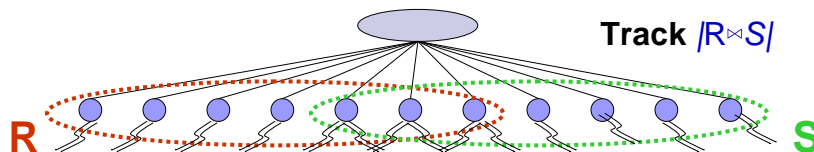


145

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Tracking Complex Aggregate Queries



- Continuous distributed tracking of complex aggregate queries using AMS sketches and local prediction models [Cormode, Garofalakis'05]
- *Class of queries:* Generalized inner products of streams

$$|R \bowtie S| = f_R \cdot f_S = \sum_v f_R[v] f_S[v] \quad (\pm \epsilon \|f_R\|_2 \|f_S\|_2)$$

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

146

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Local Sketches and Sketch Prediction

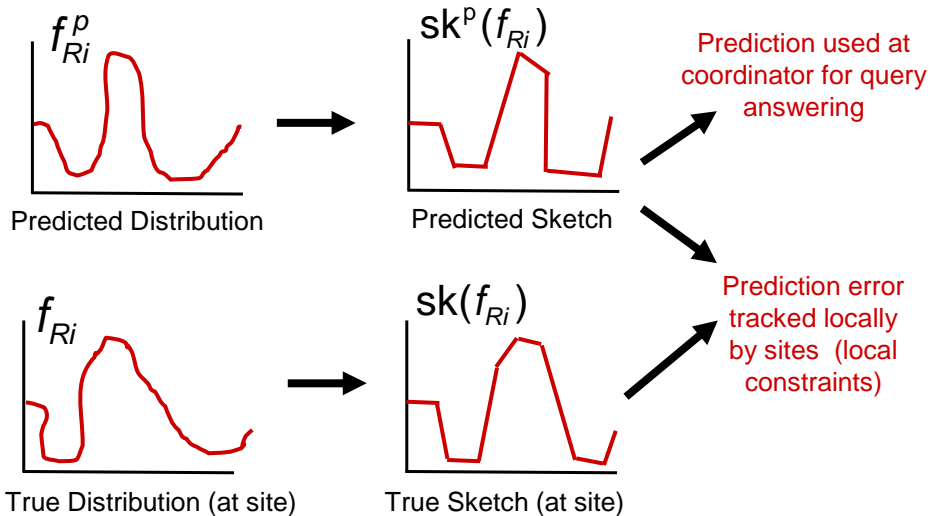
- Use (AMS) sketches to summarize local site distributions
 - Synopsis=small collection of random linear projections $sk(f_{R,i})$
 - *Linear transform*: Simply add to get global stream sketch
- Minimize updates to coordinator through *Sketch Prediction*
 - Try to predict how local-stream distributions (and their sketches) will evolve over time
 - Concise *sketch-prediction models*, built locally at remote sites and communicated to coordinator
 - *Shared knowledge* on expected stream behavior over time: Achieve “stability”

147

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketch Prediction



148

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Query Tracking Scheme

Tracking. At site j keep sketch of stream so far, $sk(f_{R,i})$

- Track local deviation between stream and prediction:

$$\| sk(f_{R,i}) - sk^p(f_{R,i}) \|_2 \leq \theta / \sqrt{k_i} \| sk(f_{R,i}) \|_2$$

- Send current sketch (and other info) if violated

Querying. At coordinator, query error $\leq (\epsilon + 2\theta) \|f_R\|_2 \|f_S\|_2$

- ϵ = local-sketch summarization error (at remote sites)
- θ = upper bound on local-stream deviation from prediction (“Lag” between remote-site and coordinator view)

- **Key Insight:** *With local deviations bounded, the predicted sketches at coordinator are **guaranteed accurate***

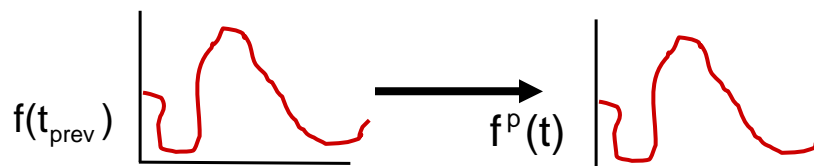
149

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketch-Prediction Models

- **Simple, concise** models of local-stream behavior
 - Sent to coordinator to keep site/coordinator “in-sync”
 - Many possible alternatives
- **Static model:** No change in distribution since last update
 - Naïve, “no change” assumption:
 - No model info sent to coordinator, $sk^p(f(t)) = sk(f(t_{prev}))$



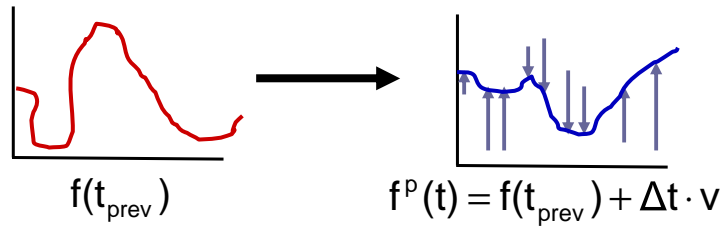
150

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Sketch-Prediction Models

- **Velocity model:** Predict change through “velocity” vectors from recent local history (simple linear model)
 - Velocity model: $f^p(t) = f(t_{prev}) + \Delta t \cdot v$
 - By sketch linearity, $sk^p(f(t)) = sk(f(t_{prev})) + \Delta t \cdot sk(v)$
 - Just need to communicate one extra sketch
 - Can extend with acceleration component



151

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Sketch-Prediction Models

Model	Info	Predicted Sketch
Static	\emptyset	$sk^p(f(t)) = sk(f(t_{prev}))$
Velocity	$sk(v)$	$sk^p(f(t)) = sk(f(t_{prev})) + \Delta t \cdot sk(v)$

- 1 – 2 orders of magnitude savings over sending all data

152

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Lessons, Thoughts, and Extensions

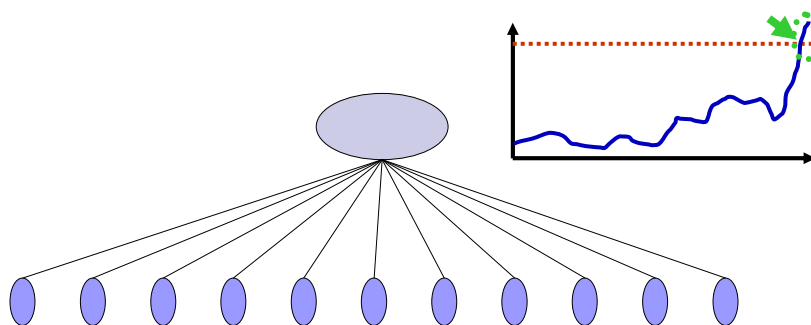
- Dynamic prediction models are a natural choice for continuous in-network processing
 - Can capture complex temporal (and spatial) patterns to reduce communication
- Many model choices possible
 - Need to **carefully balance power & conciseness**
 - Principled way for model selection?
- General-purpose solution (generality of AMS sketch)
 - Better solutions for special queries
E.g., continuous quantiles [Cormode et al.'05]

153

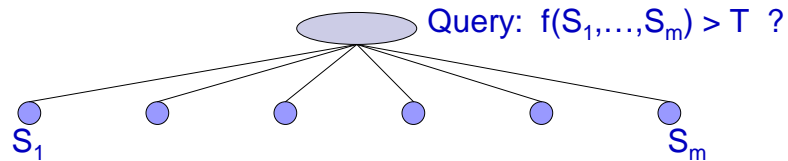
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Distributed Triggers



Tracking Distributed Triggers



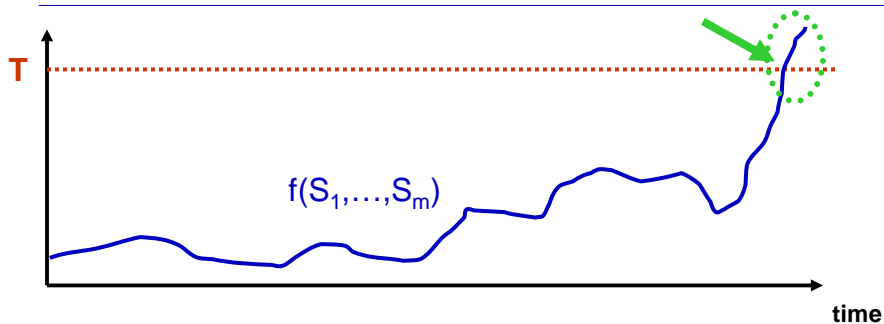
- Only interested in values of the “global query” above a certain threshold T
 - Network anomaly detection (e.g., DDoS attacks)
 - Total number of connections to a destination, “fire” when it exceeds a threshold
 - Air / water quality monitoring, total number of cars on highway
 - Fire when count/average exceeds a certain amount
- Introduced in HotNets paper [Jain, Hellerstein et al.’04]

155

Processing Massive Data Streams – VLDB School’2008, Cairo, Egypt

YAHOO!
RESEARCH

Tracking Distributed Triggers



- Problem “easier” than approximate query tracking
 - Only want accurate $f()$ values when they’re close to threshold
 - *Exploit threshold for intelligent slack allocation to sites*
- Push-based in-network operation even more relevant
 - Optimize operation for “common case”

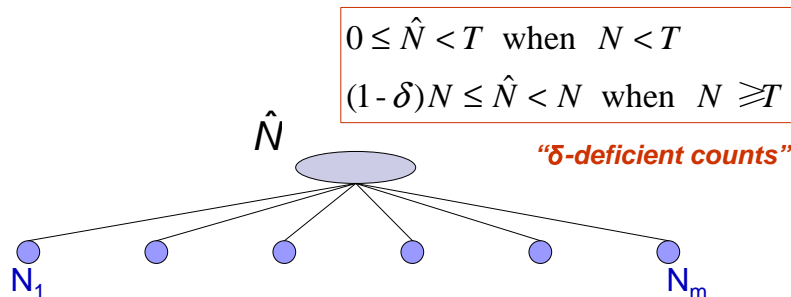
156

Processing Massive Data Streams – VLDB School’2008, Cairo, Egypt

YAHOO!
RESEARCH

Tracking Thresholded Counts

- Monitor a distributed aggregate count
- Guarantee a user-specified accuracy δ *only if the count exceeds a pre-specified threshold T* [Kerlapura et al.'06]
 - E.g., N_i = number of observed connections to 128.105.7.31
and $N = \sum_i N_i$



157

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Thresholded Counts Approach

- Site i maintains a set of local thresholds $t_{i,j}$, $j = 0, 1, 2, \dots$
- Local filter at site i : $t_{i,f(i)} \leq N_i < t_{i,f(i)+1}$
 - Local count between adjacent thresholds
 - Contact coordinator with new "level" $f(i)$ when violated
- Global estimate at coordinator $\hat{N} = \sum_i t_{i,f(i)}$
- For δ -deficient estimate, choose local threshold sequences $t_{i,j}$ such that

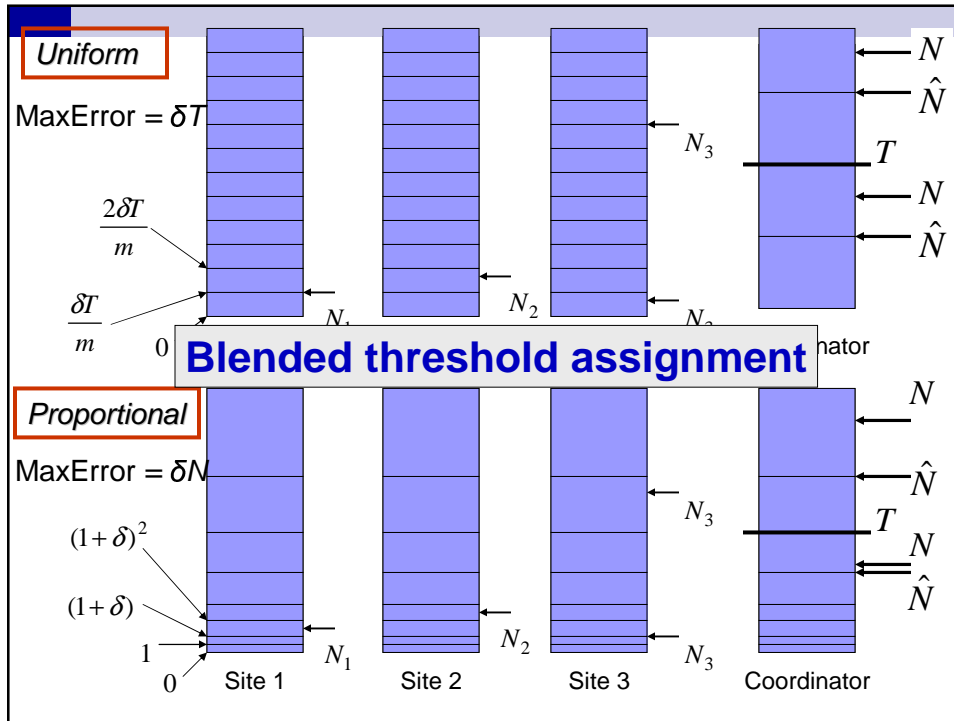
$$\sum_i (t_{i,f(i)+1} - t_{i,f(i)}) < \delta \sum_i t_{i,f(i)} \text{ whenever } \sum_i t_{i,f(i)+1} > T$$

"large" to minimize communication!
"small" to ensure global error bound!

158

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH



Blended Threshold Assignment

- Uniform: overly tight filters when $N > T$
- Proportional: overly tight filters when $N \ll T$
- **Blended Assignment**: combines best features of both:

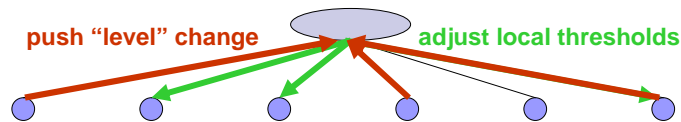
$$t_{i,j+1} = (1+\alpha\delta) \cdot t_{i,j} + (1-\alpha) \cdot \delta T / m \quad \text{where } \alpha \in [0,1]$$

- $\alpha = 0 \Rightarrow$ Uniform assignment
- $\alpha = 1 \Rightarrow$ Proportional assignment

- Optimal value of α exists for given N (expected or distribution)
 - Determined through, e.g., gradient descent

Adaptive Thresholding

- So far, *static* threshold sequences
 - Every site only has “local” view and just pushes updates to coordinator
- Coordinator has global view of current count estimate
 - Can *adaptively* adjust the local site thresholds (based on estimate and T)
 - E.g., dynamically switch from *uniform* to *proportional* growth strategy as estimate approaches/exceeds T

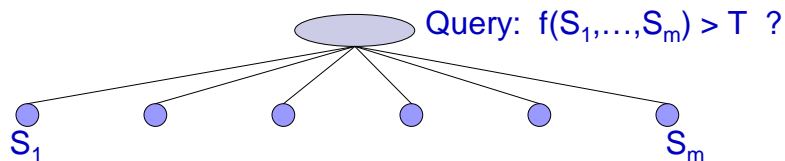


161

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

What about *Non-Linear* Functions?



- For *general, non-linear* $f()$, the problem becomes a lot harder!
 - E.g., information gain or entropy 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 $f(N) = 6N - N^2 > 1$
Impossible to break into thresholds for $f(N_1)$ and $f(N_2)$

162

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Monitoring General Threshold Functions

- Interesting *geometric* approach [Scharfman et al.'06]
- Each site tracks a *local statistics vector* v_i (e.g., data distribution)
- Global condition is $f(v) > T$, where $v = \sum_i \lambda_i v_i$ ($\sum_i \lambda_i = 1$)
 - v = convex combination of local statistics vectors
- All sites have an estimate $e = \sum_i \lambda_i v_i'$ of v based on latest update v_i' from site i
- Each site i continuously tracks its *drift* from its most recent update $\Delta v_i = v_i - v_i'$

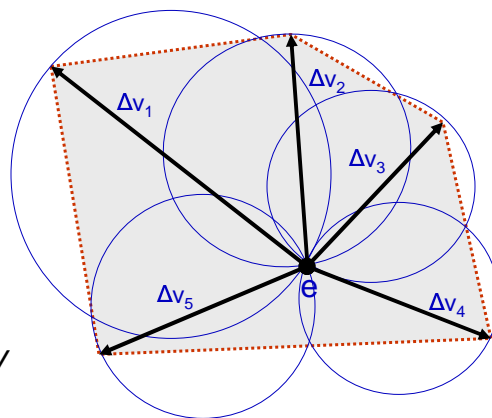
163

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Monitoring General Threshold Functions

- 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 the *balls* with radii $\|\Delta v_i / 2\|_2$ centered at $e + \Delta v_i / 2$
- Each such ball can be constructed *independently*



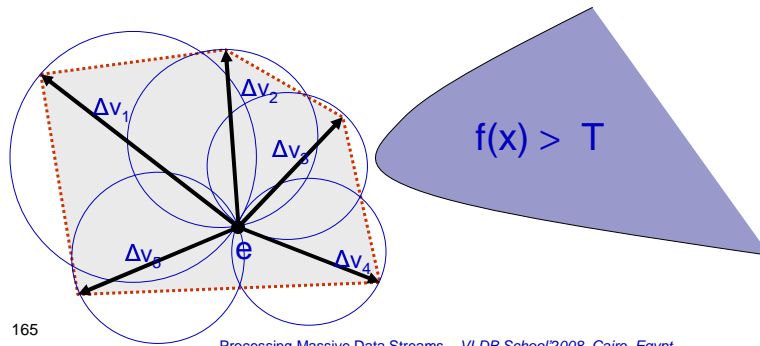
164

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

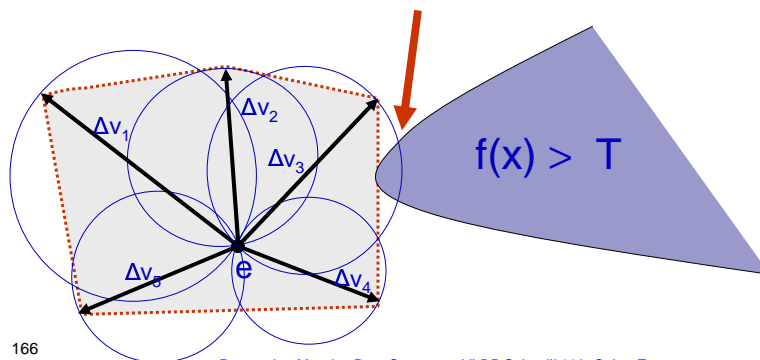
Monitoring General Threshold Functions

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



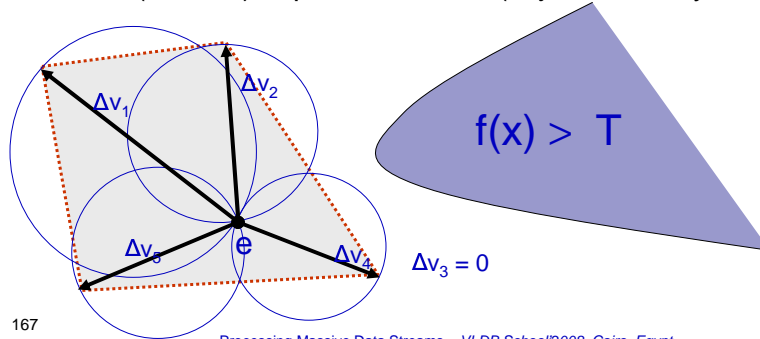
Monitoring General Threshold Functions

- After broadcast, $\|\Delta v_i\|_2 = 0 \Rightarrow$ Ball at i is monochromatic



Monitoring General Threshold Functions

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

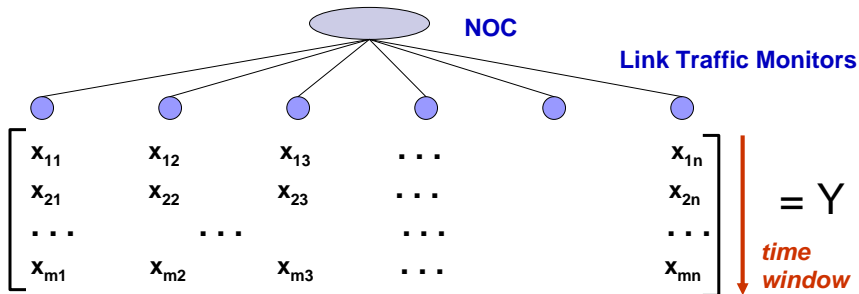


167

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Extension: Filtering for PCA Tracking



- Threshold total energy of the low PCA coefficients of $Y =$ Robust indicator of network-wide anomalies [Lakhina et al.'04]
 - Non-linear matrix operator over combined time-series
- Can combine local filtering ideas with *stochastic matrix perturbation theory* [Huang et al.'06]

168

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Lessons, Thoughts and Extensions

- Key idea in *trigger tracking*: The threshold is your friend!
 - Exploit for more intelligent (looser, yet “safe”) local filtering
- Also, optimize for the common case!
 - Threshold violations are typically “outside the norm”
 - “Push-based” model makes even more sense here
 - Local filters eliminate most/all of the “normal” traffic
- Use richer, dynamic prediction models for triggers?
 - Perhaps adapt depending on distance from threshold?
- More realistic network models?
- Geometric ideas for approximate query tracking?
 - Connections to approximate join-tracking scheme?

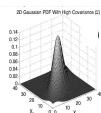
169

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Tutorial Outline

- Motivation & Streaming Applications
- Centralized Stream Processing
- Distributed Stream Processing
 - One-shot distributed-stream querying
 - Continuous distributed-stream tracking
 - Probabilistic distributed data acquisition
- Open Problems & Future Directions
- Conclusions



170

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Model-Driven Data Acquisition

- *Not only aggregates* – Approximate, bounded-error acquisition of individual sensor values [Deshpande et al. '04]
 - (ϵ, δ) -approximate acquisition: $|Y - \hat{Y}| \leq \epsilon$ with prob. $> 1 - \delta$
- Regular readings entails large amounts of data, noisy or incomplete data, inefficient, low battery life, ...
- *Intuition*: Sensors give (noisy, incomplete) samples of real-world processes
- Use *dynamic probabilistic model* of real-world process to
 - Robustly complement & interpret obtained readings
 - Drive efficient acquisitional query processing

171

Processing Massive Data Streams – VLDB School 2008, Cairo, Egypt

YAHOO! RESEARCH

Query Processing in TinyDB

Declarative Query
`select nodeID, temp
 where nodeID in {1..6}`

USER

Query Results

1, 22.73,
 ...
 6, 22.1.

Query Processor

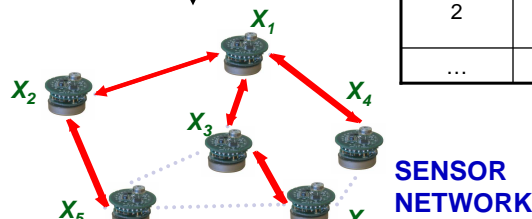
Virtual Table seen by the User

Observation Plan
`{[temp, 1], [temp, 2],
 ... , [temp, 6]}`

Data

1, temp
 ...
 6, temp

nodeID	Time	temp
1	10am	21
2	10am	22
...



172

Processing Massive Data Streams – VLDB School 2008, Cairo, Egypt

YAHOO! RESEARCH

Model-Based Data Acquisition: BBQ

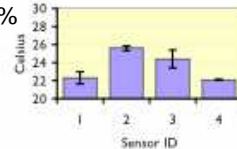
Declarative Query
 Select nodeID,
 temp ± .1C, conf(.95)
 where nodeID in {1..6}

USER

Query Results
 1, 22.73, 100%
 ...
 6, 22.1, 99%

Probabilistic Model

Query Processor



Observation Plan
 {[temp, 1],

Data
 1, temp = 22.73,

A *dynamic probabilistic model* of how the data (or the underlying physical process) behaves

- Models the evolution over time
- Captures inter-attribute correlations
- Domain-dependent

173

Processing massive Data Streams – VLDB School 2008, Cairo, Egypt

YAHOO! RESEARCH

BBQ Details

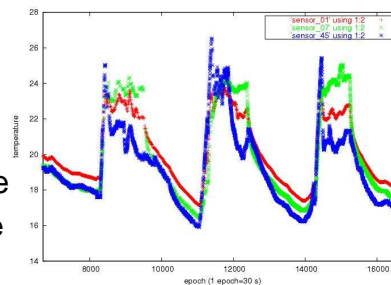
Probabilistic model captures the joint pdf $p(X_1, \dots, X_n)$

- *Spatial/temporal correlations*

- Sensor-to-sensor
- Attribute-to-attribute
 E.g., voltage & temperature

- *Dynamic*: pdf evolves over time

- BBQ: Time-varying multivariate Gaussians



- Given user query Q and accuracy guarantees (ϵ, δ)

- Try to answer Q directly from the current model
- If not possible, use model to find efficient *observation plan*
- Observations update the model & generate (ϵ, δ) answer

174

Processing Massive Data Streams – VLDB School 2008, Cairo, Egypt

YAHOO! RESEARCH

BBQ Probabilistic Queries

- Classes of probabilistic queries
 - *Range predicates*: Is $X_i \in [a_i, b_i]$ with prob. $> 1 - \delta$
 - *Value estimates*: Find X'_i such that $\Pr[|X_i - X'_i| < \epsilon] > 1 - \delta$
 - *Aggregate estimates*: (ϵ, δ) -estimate **avg/sum**($X_{i1}, X_{i2} \dots X_{ik}$)
 - Acquire readings if model cannot answer Q at δ conf. level
 - Key model operations are
 - *Marginalization*: $p(X_i) = \int p(X_1, \dots, X_n) dx$
 - *Conditioning*: $p(X_1, \dots, X_n | \text{observations})$
 - *Integration*: $\int_a^b p(X_1, \dots, X_n) dx$, also expectation $X'_i = E[X_i]$
- All significantly simplified for Gaussians!**

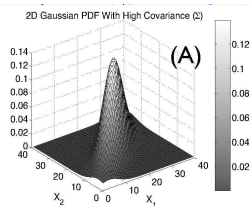
175

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

BBQ Query Processing

Joint pdf at time=t
 $p(X_1^t, \dots, X_n^t)$



Probabilistic query
 Value of $X_2 \pm \epsilon$
 with prob. $> 1 - \delta$

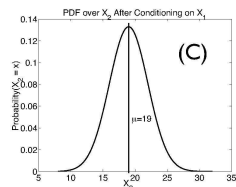
$$\text{Is } P(X_2 \in [\mu_2 - \epsilon, \mu_2 + \epsilon]) = \int_{\mu_2 - \epsilon}^{\mu_2 + \epsilon} P(x_2) dx_2$$

below $1 - \delta$?

Yes

**Higher prob.,
 can now
 answer query**

$P(X_2 | X_1 = 18)$



Return μ
Must sense more data
 Example: Observe $X_1 = 18$
 Incorporate into model

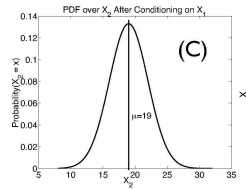
176

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

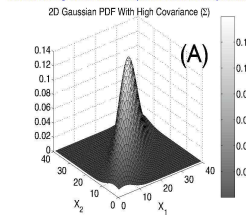
Evolving the Model over Time

Joint pdf at time= t
 $p(X_1^t, \dots, X_n^t | X_1^t=18)$



Use a (Markov)
Transition Model
 $P(X^{t+1} | X^t)$

Joint pdf at time= $t+1$
 $p(X_1^{t+1}, \dots, X_n^{t+1} | X_1^t=18)$



- In general, a two-step process:

$$p(X^t | obs^{1\dots t}) \xrightarrow{\text{Trans. Model}} p(X^{t+1} | obs^{1\dots t}) \xrightarrow{\text{Condition}} p(X^{t+1} | obs^{1\dots t+1})$$

- *Bayesian filtering* (for Gaussians this yields *Kalman filters*)

177

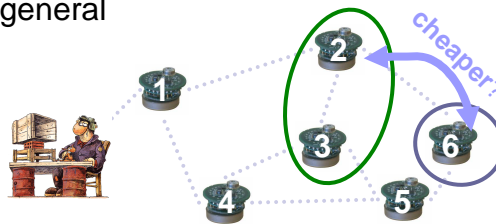
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Optimizing Data Acquisition

- Energy/communication-efficient observation plans
 - Non-uniform data acquisition costs and network communication costs
 - Exploit data correlations and knowledge of topology
- Minimize $Cost(obs)$ over all $obs \subseteq \{1, \dots, n\}$ so expected confidence in query answer given obs (from model) $> 1-\delta$
- **NP-hard** to optimize in general

Sensor	Energy per sample (mJ)
Solar Radiation	.525
Barometric Pressure	0.003
Humidity and Temp.	0.5
Voltage	0.00009



178

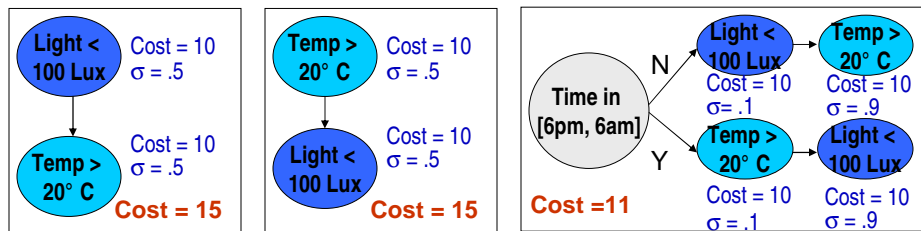
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Conditional Plans for Data Acquisition

- Observation plans ignore the attribute values observed
 - Attribute subset chosen is observed in its entirety
 - The observed attribute values give a lot more information
- *Conditional* observation plans (outlined in [Deshpande et al.'05])
 - Change the plan depending on observed attribute values (not necessarily in the query)
 - Not yet explored for *probabilistic* query answers

`SELECT * FROM sensors WHERE light < 100 Lux and temp > 20°C`



179

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

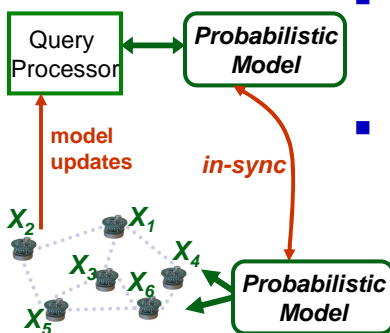
YAHOO! RESEARCH

Continuous Model-Driven Acquisition

```
select nodeID,
temp ± .1C, conf(.95)
where nodeID in {1..6}
epoch 2 min
```

Dynamic Replicated Prob Models (Ken)
[Chu et al.'06]

- Model *shared and sync'd* across base-station and sensornet
 - Push vs. Pull (BBQ)
- Nodes continuously check & maintain model accuracy based on ground truth
 - Push vs. Pull (BBQ)
- *Problem: In-network model maintenance*
 - Exploit *spatial data correlations*
 - Model updates decided in-network and sent to base-station
 - Always keep model (ϵ, δ) -approximate

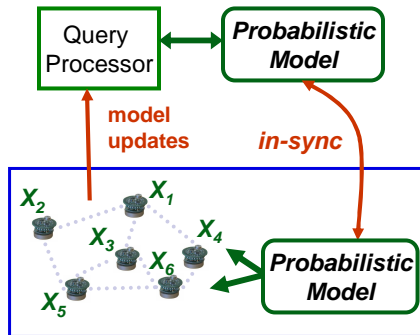


180

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

In-Network Model Maintenance



- Mapping model maintenance onto network topology
 - At each step, nodes check (ϵ, δ) accuracy, send updates to base
- Choice of model drastically affects communication cost
 - Must centralize correlated data for model check/update
 - Can be expensive!

■ Effect of *degree of spatial correlations*:

Single-node models $\prod p(X_i)$
 No spatial correlations
Cheap – check is local!



Full-network model $p(X_1, \dots, X_n)$
 Full spatial correlations
Expensive – centralize all data!

181

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

In-Network Model Maintenance

Single-node models $\prod p(X_i)$
 No spatial correlations
Cheap – check is local!



Full-network model $p(X_1, \dots, X_n)$
 Full spatial correlations
Expensive – centralize all data!

Single-node Kalman filters
 [Jain et al. '04]

BBQ
 [Deshpande et al. '04]

- **Problem:** Find dynamic probabilistic model and in-network maintenance schedule to minimize overall communication
 - Map maintenance/update operations to network topology
- Key idea for “practical” in-network models
 - Exploit *limited-radius* spatial correlations of measurements
 - Localize model checks/updates to small regions

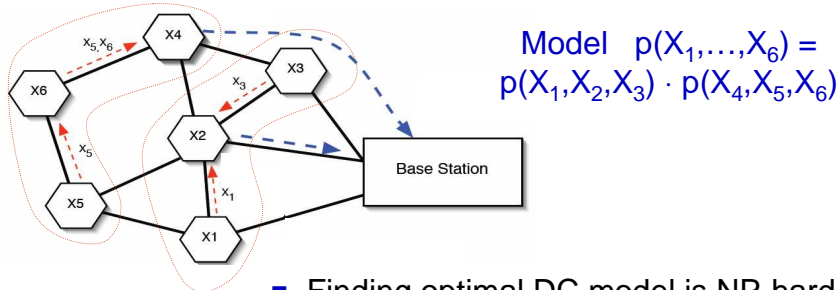
182

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Disjoint-Cliques Models

- *Idea*: Partition joint pdf into a set of small, localized “cliques” of random variables
 - Each clique maintained and updated *independently* at “clique root” nodes



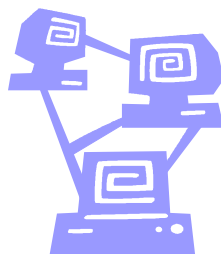
- Finding optimal DC model is NP-hard
 - Natural analogy to *Facility Location*

183

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

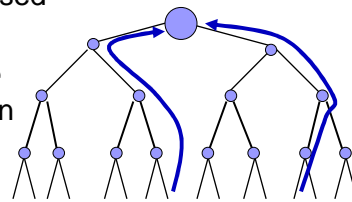
YAHOO!
RESEARCH

Distributed Data Stream Systems/Prototypes



Current Systems/Prototypes

- Main algorithmic idea in the tutorial: Trade-off space/time *and communication* with approximation quality
- Unfortunately, approximate query processing tools are still not widely adopted in current Stream Processing engines
 - Despite obvious relevance, *especially for streaming data*
- In the sensornet context
 - Simple in-network aggregation techniques (e.g., for **average**, **count**, etc.) are widely used
E.g., TAG/TinyDB [Madden et al '02]
 - More complex tools for *approximate* in-network data processing/collection have yet to gain wider acceptance



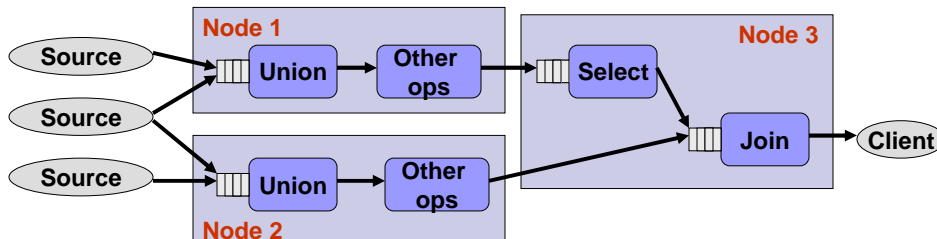
185

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed SP Engine Prototypes

- Telegraph/TelegraphCQ [Chandrasekaran et al.'03], Borealis/Medusa [Balazinska et al.'05], P2 [Loo et al.'06]
- Query processing typically viewed as a *large dataflow*
 - Network of connected, pipelined query operators



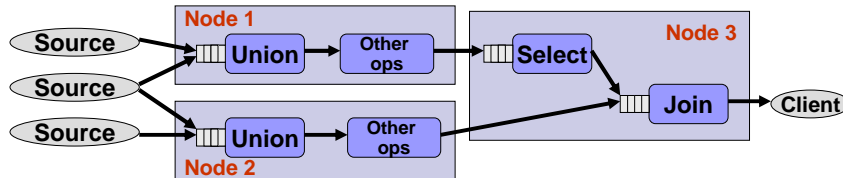
- Schedule a large dataflow over a distributed system
 - Objectives: Load-balancing, availability, early results, ...

186

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Distributed SP Engine Prototypes



- Approximate answers and error guarantees not considered
 - General relational queries, push/pull-ing tuples through the query network
 - **Load-shedding** techniques to manage overload
 - No hard error guarantees
- Network costs (bandwidth/latency) considered in some recent work [Pietzuch et al.'06]

187

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Other Systems & Prototypes

- **PIER** – Scaling to large, dynamic site populations using DHTs [Huebsch et al.'03]
 - See also the *Seaweed* paper [Narayanan et al.'06]
- **Gigaspice** – Streaming DB engine for large-scale network/ application monitoring
 - Optimized for high-rate data streams (“line speeds”)
 - Exploits approximate query processing tools (sampling, sketches, ...) for tracking streams at endpoints
 - Distribution issues not addressed (yet...)

188

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Tutorial Outline

- Motivation & Streaming Applications
- Centralized Stream Processing
- Distributed Stream Processing
- Open Problems & Future Directions
- Conclusions



189

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Basic Streaming Models & Issues

- Lots of other work on streaming data analysis problems
 - Stream mining (clustering, associations, classification, change detection,...)
 - XML stream filtering (pub/sub systems)
 - Geometric data/queries (location streams)
- Other emerging richer streaming models and problems
 - XML & text stream mining (beyond simple filtering)
 - Graph-data streams (e.g., stream of graph edges)
 - Both bound to gain importance with the proliferation of huge web data sets (e.g., WebGraph, social networks)
 - E.g., PageRank computation over a streaming WebGraph?

190

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Extensions for P2P Networks

- Much work focused on specifics of sensor and wired nets
- P2P and Grid computing present alternate models
 - Structure of multi-hop overlay networks
 - “Controlled failure” model: nodes explicitly leave and join
- Allows us to think beyond model of “highly resource constrained” sensors.
- Implementations such as OpenDHT over PlanetLab [Rhea et al.'05]



191

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Delay-Tolerant Networks

- How to cope when connectivity is *intermittent* ?
 - Roaming devices, exploring outer and inner space, network infrastructure for emerging regions (e.g., rural India), ...
 - Round trip times may be very long and varying
 - Radio to Mars is many minutes
 - Connectivity to remote villages varies [Jain, Fall, Patra '05]
- Goal is to minimize the **number** of communications and maximize **timeliness**
 - Size of communication is secondary

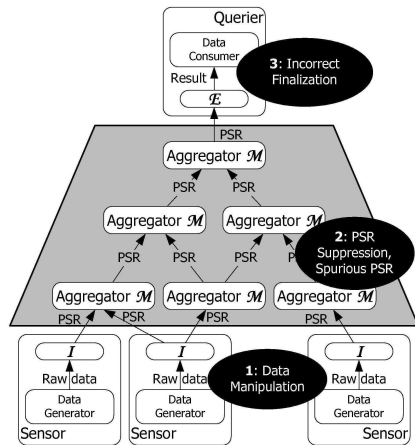


192

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Authenticated Stream Aggregation



- Wide-area query processing
 - Possible *malicious aggregators*
 - Can suppress or add spurious information
- Authenticate query results at the querier?
 - Perhaps, to within some approximation error
- Initial steps in [Garofalakis et al.'07]

193

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Other Classes of Queries

- Mostly talked about specific, well-defined aggregates
- What about *set-valued* query answers?
 - No principled, “universal” approximation error metric
- A general distributed query language (dist-streamSQL?)
 - Define a language so a query optimizer can find a plan that guarantees good performance, small communication?
- Other tasks, e.g., data mining, machine learning, over distributed streams?
 - ML/AI communities are already starting to consider communication-efficient distributed learning

194

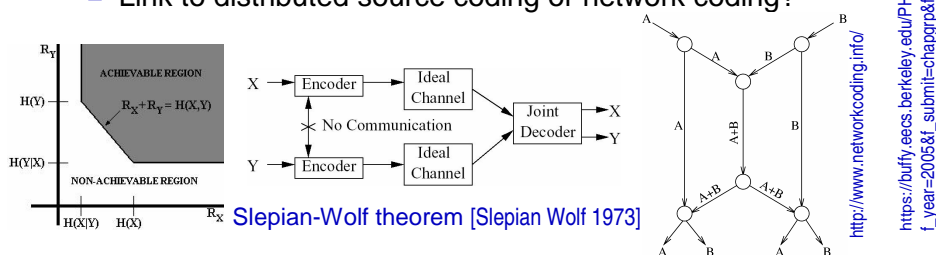
Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

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]

195

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

<http://www.networkcoding.info/>

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

Richer Prediction models

- The better we can capture and anticipate future stream direction, the less communication is needed
- So far, only look at predicting each stream alone
- Correlation/anti-correlation across streams should help?
 - But then, checking validity of model is expensive!
- Explore tradeoff-between power (expressiveness) of model and complexity (number of parameters)
 - Optimization via Minimum Description Length (MDL)? [Rissanen 1978]

196

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO! RESEARCH

Conclusions

- **Data Streaming:** Major departure from traditional persistent database paradigm
 - Fundamental re-thinking of models, assumptions, algorithms, system architectures, ...
- Many new streaming problems posed by developing technologies
- Simple tools from **approximation and/or randomization** play a critical role in effective solutions
 - Sampling, sketches (AMS, CM, FM, ...), Exponential histograms, ...
 - Simple, yet powerful, ideas with **great reach**
 - Can often “**mix & match**” for specific scenarios

197

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Conclusions

- **Distributed Streams:** Common features allow for general techniques/principles instead of “point” solutions
 - **In-network query processing**
Local filtering at sites, trading-off approximation with processing/network costs, ...
 - **Models of “normal” operation**
Static, dynamic (“predictive”), probabilistic, ...
 - **Exploiting network locality and avoiding global resyncs**
- Many new directions unstudied, more will emerge as new technologies arise
- *Lots of exciting research to be done!* ☺

198

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

Thank you!



<http://www.cs.berkeley.edu/~minos/>

minos@acm.org

199

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

References (1)

- [Aduri, Tirthapura '05] P. Aduri and S. Tirthapura. Range-efficient Counting of F_0 over Massive Data Streams. In IEEE International Conference on Data Engineering, 2005
- [Agrawal et al. '04] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In ACM SenSys, 2004
- [Alon, Gibbons, Matias, Szegedy '99] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In Proceedings of ACM Symposium on Principles of Database Systems, pages 10–20, 1999.
- [Alon, Matias, Szegedy '96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In Proceedings of the ACM Symposium on Theory of Computing, pages 20–29, 1996. Journal version in Journal of Computer and System Sciences, 58:137–147, 1999.
- [Babcock et al. '02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems In ACM Principles of Database Systems, 2002
- [Bar-Yossef et al.'02] Z. Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, Luca Trevisan: Counting Distinct Elements in a Data Stream. Proceedings of RANDOM 2002.
- [Chu et al'06] D. Chu, A. Deshpande, J. M. Hellerstein, W. Hong. Approximate Data Collection in Sensor Networks using Probabilistic Models. IEEE International Conference on Data Engineering 2006, p48
- [Considine, Kollios, Li, Byers '05] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In IEEE International Conference on Data Engineering, 2004.
- [Cormode, Garofalakis '05] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In Proceedings of the International Conference on Very Large Data Bases, 2005.
- [Cormode et al.'05] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2005.
- [Cormode, Muthukrishnan '04] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms, 55(1):58–75, 2004.
- [Cormode, Muthukrishnan '05] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In Proceedings of ACM Principles of Database Systems, 2005.

200

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

References (2)

- [Cormode et al.'05] G. Cormode, S. Muthukrishnan, I. Rozenbaum. Summarizing and Mining Inverse Distributions on Data Streams via Dynamic Inverse Sampling. In Proceedings of VLDB 2005.
- [Cormode et al.'06] Graham Cormode, Minos N. Garofalakis, Dimitris Sacharidis: Fast Approximate Wavelet Tracking on Streams. In Proceedings of EDBT 2006.
- [Das et al.'04] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed Set-Expression Cardinality Estimation. In Proceedings of VLDB, 2004.
- [Datar et al.'02] M. Datar, Aristides Gionis, Piotr Indyk, Rajeev Motwani. Maintaining stream statistics over sliding windows (extended abstract). In Proceedings of SODA 2002.
- [Deshpande et al.'04] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, W. Hong. Model-Driven Data Acquisition in Sensor Networks. In VLDB 2004, p 588-599
- [Deshpande et al.'05] A. Deshpande, C. Guestrin, W. Hong, S. Madden. Exploiting Correlated Attributes in Acquisitional Query Processing. In IEEE International Conference on Data Engineering 2005, p143-154
- [Dilman, Raz '01] M. Dilman, D. Raz. Efficient Reactive Monitoring. In IEEE Infocom, 2001.
- [Dobra et al.'02] A. Dobra, M. Garofalakis, J. Gehrke, R. Rastogi. Processing Complex Aggregate Queries over Data Streams. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2002.
- [Dobra et al.'04] A. Dobra, M. Garofalakis, J. Gehrke, R. Rastogi. Sketch-Based Multi-query Processing over Data Streams. In Proceedings of EDBT 2004.
- [Flajolet, Martin '83] P. Flajolet and G. N. Martin. Probabilistic counting. In IEEE Conference on Foundations of Computer Science, pages 76–82, 1983. Journal version in Journal of Computer and System Sciences, 31:182–209, 1985.
- [Ganguly et al.'04] S. Ganguly, M. Garofalakis, R. Rastogi. Tracking set-expression cardinalities over continuous update streams. The VLDB Journal, 2004
- [Ganguly et al.'04] S. Ganguly, M. Garofalakis, R. Rastogi. Processing Data-Stream Join Aggregates Using Skimmed Sketches. In Proceedings of EDBT 2004.
- [Garofalakis et al.'02] M. Garofalakis, J. Gehrke, R. Rastogi. Querying and Mining Data Streams: You Only Get One Look. Tutorial in ACM SIGMOD International Conference on Management of Data, 2002.
- [Garofalakis et al.'07] M. Garofalakis, J. Hellerstein, and P. Maniatis. Proof Sketches: Verifiable Multi-Party Aggregation. In Proceedings of ICDE 2007.

201

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt



References (3)

- [Gemulla et al.'08] Rainer Gemulla, Wolfgang Lehner, Peter J. Haas. Maintaining bounded-size sample synopses of evolving datasets. In The VLDB Journal, 2008.
- [Gibbons'01] P. Gibbons. Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports. Proceedings of VLDB'2001.
- [Gibbons, Tirthapura '01] P. Gibbons, S. Tirthapura. Estimating simple functions on the union of data streams. In ACM Symposium on Parallel Algorithms and Architectures, 2001.
- [Gilbert et al.'01] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, Martin Strauss. Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries. In Proceedings of VLDB 2001.
- [Greenwald, Khanna '01] M. Greenwald, S. Khanna. Space-efficient online computation of quantile summaries. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2001.
- [Greenwald, Khanna '04] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In Proceedings of ACM Principles of Database Systems, pages 275–285, 2004.
- [Hadjieleftheriou, Byers, Kollios '05] M. Hadjieleftheriou, J. W. Byers, and G. Kollios. Robust sketching and aggregation of distributed data streams. Technical Report 2005-11, Boston University Computer Science Department, 2005.
- [Huang et al.'06] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft. Distributed PCA and Network Anomaly Detection. In NIPS, 2006.
- [Huebsch et al.'03] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, I. Stoica. Querying the Internet with PIER. In VLDB, 2003.
- [Jain et al.'04] A. Jain, E. Y. Chang, Y-F. Wang. Adaptive stream resource management using Kalman Filters. In ACM SIGMOD International Conference on Management of Data, 2004.
- [Jain, Fall, Patra '05] S. Jain, K. Fall, R. Patra, Routing in a Delay Tolerant Network, In IEEE Infocom, 2005
- [Jain, Hellerstein et al.'04] A. Jain, J.M.Hellerstein, S. Ratnasamy, D. Wetherall. A Wakeup Call for Internet Monitoring Systems: The Case for Distributed Triggers. In Proceedings of HotNets-III, 2004.
- [Johnson et al.'05] T. Johnson, S. Muthukrishnan, V. Shkapenyuk, and O. Spateschek. A heartbeat mechanism and its application in Gigascope. In VLDB, 2005.

202

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt



References (4)

- [Kashyap et al. '06] S. Kashyap, S. Deb, K.V.M. Naidu, R. Rastogi, A. Srinivasan. Efficient Gossip-Based Aggregate Computation. In ACM Principles of Database Systems, 2006.
- [Kempe, Dobra, Gehrke '03] D. Kempe, A. Dobra, and J. Gehrke. Computing aggregates using gossip. In IEEE Conference on Foundations of Computer Science, 2003.
- [Kempe, Kleinberg, Demers '01] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In Proceedings of the ACM Symposium on Theory of Computing, 2001.
- [Kerlapura et al.'06] R. Kerlapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In ACM SIGMOD, 2006.
- [Koudas, Srivastava '03] N. Koudas and D. Srivastava. Data stream query processing: A tutorial. In VLDB, 2003.
- [Madden '06] S. Madden. Data management in sensor networks. In Proceedings of European Workshop on Sensor Networks, 2006.
- [Madden et al. '02] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. In Proceedings of Symposium on Operating System Design and Implementation, 2002.
- [Manjhi, Nath, Gibbons '05] A. Manjhi, S. Nath, and P. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2005.
- [Manjhi et al.'05] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In IEEE International Conference on Data Engineering, pages 767–778, 2005.
- [Muthukrishnan '03] S. Muthukrishnan. Data streams: algorithms and applications. In ACM-SIAM Symposium on Discrete Algorithms, 2003.
- [Narayanan et al.'06] D. Narayanan, A. Donnelly, R. Mortier, and A. Rowstron. Delay-aware querying with Seaweed. In VLDB, 2006.
- [Nath et al.'04] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In ACM SenSys, 2004.
- [Olston, Jiang, Widom '03] C. Olston, J. Jiang, J. Widom. Adaptive Filters for Continuous Queries over Distributed Data Streams. In ACM SIGMOD, 2003.

203

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH

References (5)

- [Pietzuch et al.'06] P. R. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, M. I. Seltzer. Network-Aware Operator Placement for Stream-Processing Systems. In IEEE ICDE, 2006.
- [Pittel '87] B. Pittel. On Spreading a Rumor. In SIAM Journal of Applied Mathematics, 47(1) 213-223, 1987.
- [Rhea et al. '05] S. Rhea, G. Brighten, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, Y. Harlan. OpenDHT: A public DHT service and its uses. In ACM SIGCOMM, 2005.
- [Rissanen '78] J. Rissanen. Modeling by shortest data description. Automatica, 14:465-471, 1978.
- [Sharfman et al.'06] Izchak Sharfman, Assaf Schuster, Daniel Keren: A geometric approach to monitoring threshold functions over distributed data streams. SIGMOD Conference 2006: 301-312
- [Slepian, Wolf '73] D. Slepian, J. Wolf. Noiseless coding of correlated information sources. IEEE Transactions on Information Theory, 19(4):471-480, July 1973.
- [Vitter'85] Jeffrey S. Vitter. Random Sampling with a reservoir. ACM Trans. on Math. Software, 11(1), 1985.

204

Processing Massive Data Streams – VLDB School'2008, Cairo, Egypt

YAHOO!
RESEARCH