Sketching Massive Distributed Data Streams

Minos Garofalakis

Internet Management Research Department
Bell Labs, Lucent Technologies

Disclaimers

Lucent Technologies
Bell Labs Innovations

- Personal, biased view of data-streaming world
  - Revolve around own line of work, interests, and results
  - Focus on one basic algorithmic tool
    - A lot more out there (e.g., different sketch types) . . .
    - Interesting research prototypes and systems work not covered
      - Aurora, STREAM, Telegraph, . . .
- Discussion necessarily short and fairly high-level
  - More detailed overviews
    - 3-hour tutorial at VLDB'02, Motwani et al. [PODS'02], upcoming edited book (Springer-Verlag). . .
  - Ask questions!
  - Talk to me afterwards

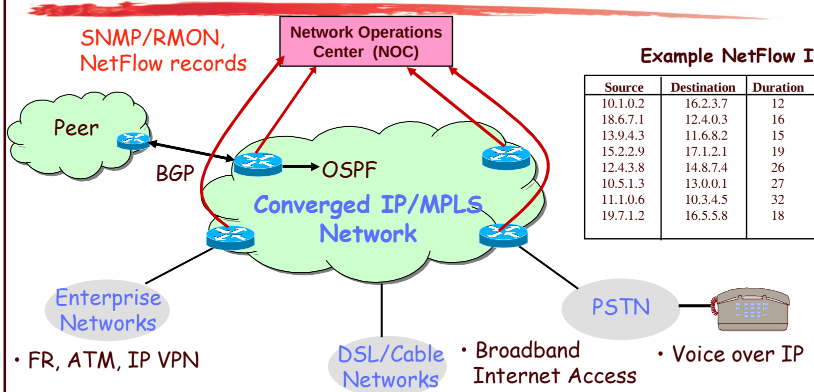
2

Data-Stream Management

- Traditional DBMS – data stored in finite, persistent data sets
- Data Streams – distributed, continuous, unbounded, rapid, time varying, noisy, . . .
- Data-Stream Management – variety of modern applications
  - Network monitoring and traffic engineering
  - Telecom call-detail records
  - Network security
  - Financial applications
  - Sensor networks
  - Manufacturing processes
  - Web logs and clickstreams
  - Massive data sets

3

Networks Generate Massive Data Streams



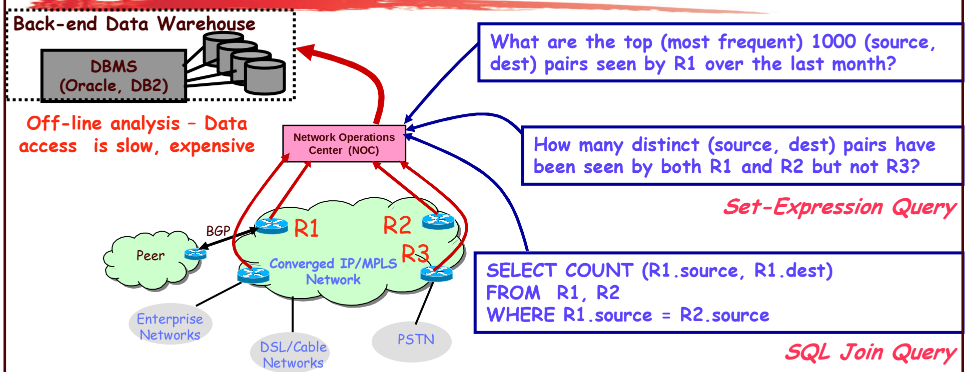
Example NetFlow IP Session Data

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

- SNMP/RMON/NetFlow data records arrive 24x7 from different parts of the network
- Truly massive streams arriving at rapid rates
  - AT&T collects 600-800 GigaBytes of NetFlow data each day!
- Typically shipped to a back-end data warehouse (off site) for off-line analysis

4

Real-Time Data-Stream Analysis



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

5

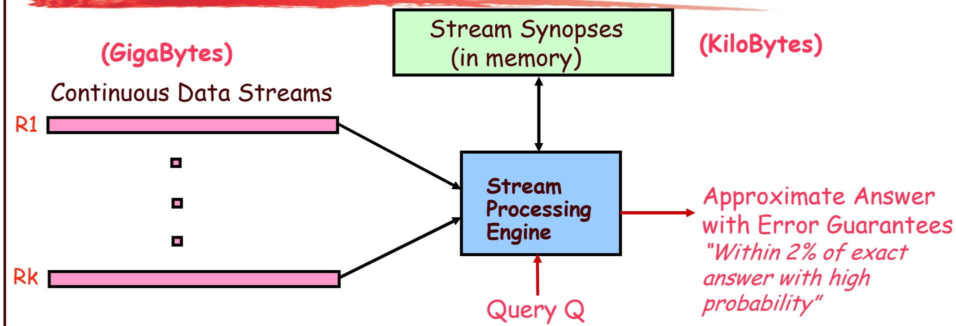
Talk Outline

- Introduction & Motivation
- Data Stream Computation Model
- A Basic Sketching Tool for Streams
  - Linear-Projection (aka AMS) Sketches
    - *Applications:* Join/Multi-Join Queries, Wavelets, Correlating XML streams
- Tracking Queries over *Distributed* Streams
  - Communication-efficient, sketch-based approach
- Conclusions & Future Research Directions

6

Data-Stream Processing Model

Lucent Technologies
Bell Labs Innovations



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

7

Synopses for Relational Streams

Lucent Technologies
Bell Labs Innovations

- Conventional data summaries fall short
  - Quantiles and 1-d histograms [MRL98,99], [GK01], [GKMS02]
    - Cannot capture attribute correlations
    - Little support for approximation guarantees
  - Samples (e.g., using Reservoir Sampling)
    - Perform poorly for joins [AGMS99] or distinct values [CCMN00]
    - Cannot handle deletion of records
  - Multi-d histograms/wavelets
    - Construction requires multiple passes over the data
- Different approach: *Pseudo-random sketch synopses*
  - Only logarithmic space
  - *Probabilistic guarantees* on the quality of the approximate answer
  - Support insertion as well as deletion of records

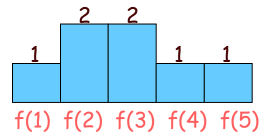
8

Linear-Projection (aka AMS) Sketch Synopses

Lucent Technologies
Bell Labs Innovations

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

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



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

$$< f, \xi >= \sum f(i)\xi_i$$ where ξ = vector of random values from an appropriate distribution

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

Data stream: 3, 1, 2, 4, 2, 3, 5, . . . → $\xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5$

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

9

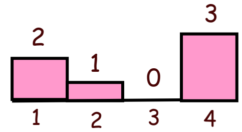
Example: Binary-Join COUNT Query

Lucent Technologies
Bell Labs Innovations

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

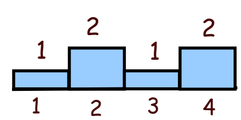
Data stream R.A: 4 1 2 4 1 4

$f_R(i)$:



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

$f_S(i)$:



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

= 10 (2 + 2 + 0 + 6)

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

10

Basic AMS Sketching Technique [AMS96]

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

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

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

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

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

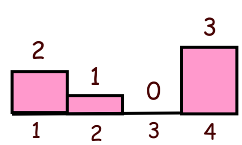
AMS Sketch Construction

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

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

$X_R = X_R + \xi_4$

$f_R(i)$:

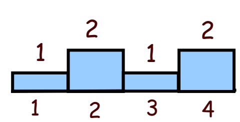


$X_R = 2\xi_1 + \xi_2 + 3\xi_4$

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

$X_S = X_S + \xi_1$

$f_S(i)$:



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

Binary-Join AMS Sketching Analysis

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

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

1   0

- Using 4-wise independence, possible to show that

$$Var[X] \leq 2 \cdot SJ(R) \cdot SJ(S)$$

- $SJ(R) = \sum_i f_R(i)^2$ is *self-join size of R*

13

Boosting Accuracy

- Chebyshev's Inequality:

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

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

X X . . . . X → Average → Y

$$s = \frac{8 \cdot (2 \cdot SJ(R) \cdot SJ(S))}{\varepsilon^2\, COUNT^2}$$ copies

$$E[Y] = E[X] = COUNT(R \bowtie S)$$

- By Chebyshev: $$Var[Y] = \frac{Var[X]}{s} \leq \frac{\varepsilon^2\, COUNT^2}{8}$$

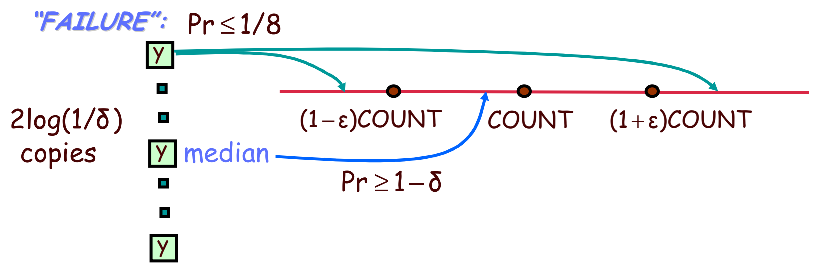
$$Pr(|Y - COUNT| \geq \varepsilon \cdot COUNT) \leq \frac{Var[Y]}{\varepsilon^2\, COUNT^2} \leq \frac{1}{8}$$

14

Boosting Confidence

Lucent Technologies Bell Labs Innovations

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



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

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

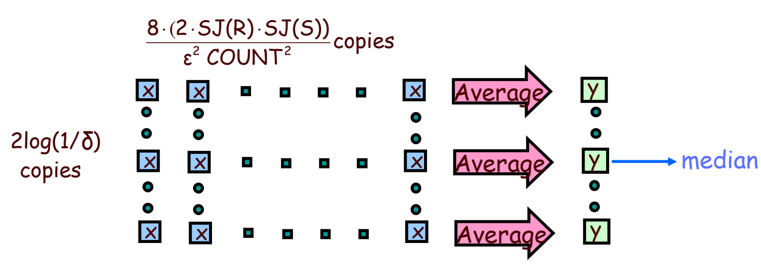
$\le \delta$ (By Chernoff Bound)

15

Summary of Binary-Join AMS Sketching

Lucent Technologies Bell Labs Innovations

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



- Main Theorem (AGMS99): Sketching approximates COUNT to within a relative error of ε with probability $\ge 1-\delta$ using space

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

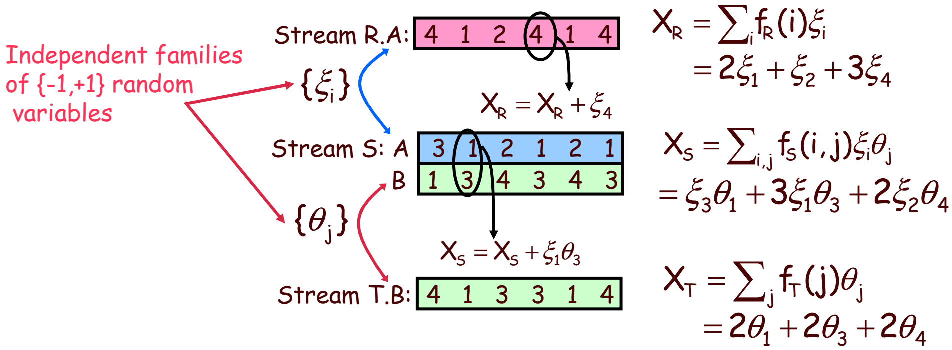
– Remember: O(log N) space for "seeding" the construction of each X

16

AMS Sketching for Multi-Join Aggregates [DGGR02]

Lucent Technologies Bell Labs Innovations

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



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

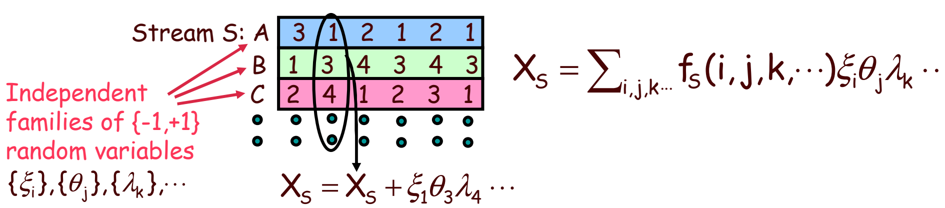
$$E[f_R(i) \cdot f_S(i',j) \cdot f_T(j')\xi_i\xi_{i'}\theta_j\theta_{j'}] = 0 \text{ if } i \neq i' \text{ or } j \neq j'$$

17

AMS Sketching for Multi-Join Aggregates

Lucent Technologies Bell Labs Innovations

- Sketches can be used to compute answers 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, ...



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

$$Var[X] \leq 2^{2m} \cdot SJ(R) \cdot SJ(S) \cdot SJ(T) \cdots$$

  - Explosive increase with the number of joins!

18

Boosting Accuracy by Sketch Partitioning: Basic Idea

- For ε error, need $Var[Y] \le \frac{\varepsilon^2 \, COUNT^2}{8}$

$$s = \frac{8 \cdot (2^{2m} SJ(R) \cdot SJ(S) \cdots)}{\varepsilon^2 \, COUNT^2} \quad \text{copies}$$

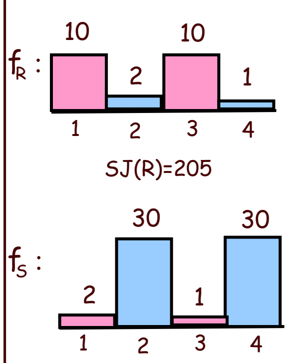
$$Var[Y] = \frac{Var[X]}{s} \le \frac{\varepsilon^2 \, COUNT^2}{8}$$

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

19

Sketch Partitioning Example: Binary-Join COUNT Query

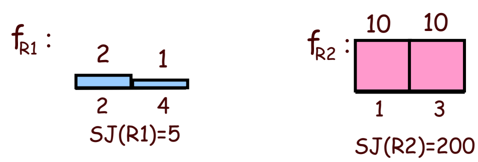
Without Partitioning

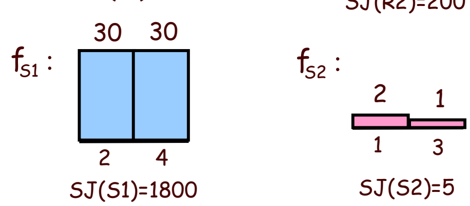


SJ(S)=1805

$VAR[X] \approx 2 \cdot SJ(R) \cdot SJ(S)$ ≈720K

With Partitioning (P1={2,4}, P2={1,3})





$VAR[X1] \approx 2 \cdot SJ(R1) \cdot SJ(S1)$ ≈18K

$VAR[X2] \approx 2 \cdot SJ(R2) \cdot SJ(S2)$ ≈2K

X = X1+X2, E[X] = COUNT(R ⋈ S)

$VAR[X] = VAR[X1] + VAR[X2]$ ≈20K

20

Overview of Sketch Partitioning

- Maintain independent sketches for partitions of join-attribute space
- Improved error guarantees
  - $Var[X] = \sum Var[Xi]$ is smaller (by *intelligent domain partitioning*)
  - *"Variance-aware" boosting:* More space to higher-variance partitions
- *Problem:* Given total sketching space S, find domain partitions p1,…, pk and space allotments s1,…,sk such that $\sum_j sj \leq S$, and the variance

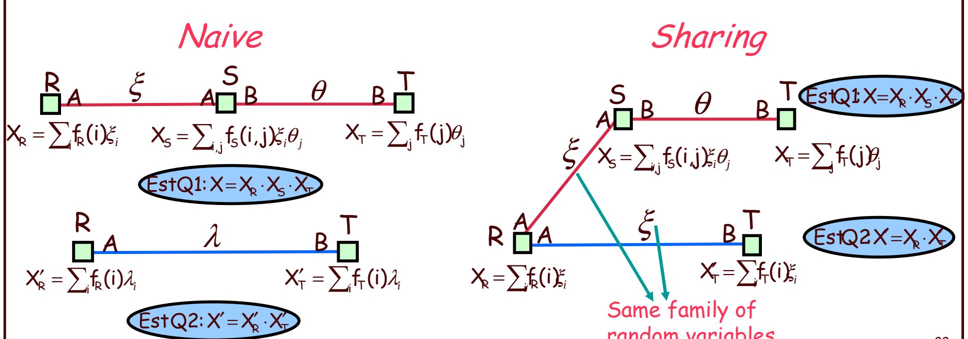
$$\frac{Var[X1]}{s1} + \frac{Var[X2]}{s2} + \cdots + \frac{Var[Xk]}{sk} \quad \text{is minimized}$$

  - Solved optimal for binary-join case (using Dynamic-Programming)
  - NP-hard for ≥ 2 joins
    - Extension of our DP algorithm is an effective heuristic -- *optimal* for independent join attributes
- Significant accuracy benefits for small number (2-4) of partitions

21

More Recent Results on Stream Joins

- Better accuracy using *"skimmed sketches"* [GGR04]
  - "Skim" dense items (i.e., large frequencies) from the AMS sketches
  - Use the "skimmed" sketch only for sparse element representation
  - Stronger worst-case guarantees, and much better in practice
    - Same effect as sketch partitioning with *no apriori knowledge!*
- Sharing sketch space/computation among *multiple queries* [DGGR04]

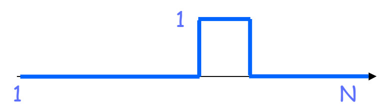


22

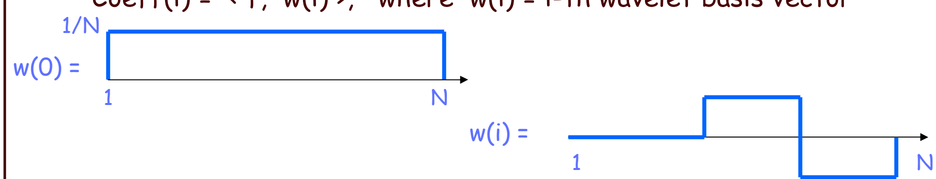
Other Applications of AMS Stream Sketching

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

- Other streaming inner products of interest
  - Top-k frequencies [CCF02]
    - Item frequency = < f, "unit_pulse" >

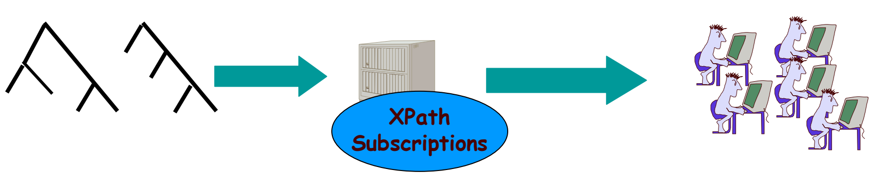


  - Large wavelet coefficients [GKMS01]
    - Coeff(i) = < f, w(i) >, where w(i) = i-th wavelet basis vector



Processing XML Data Streams

- XML: Much richer, (semi)structured data model
  - Ordered, node-labeled data trees
- Bulk of work on XML streaming: *Content-based filtering of XML documents* (publish/subscribe systems)
  - Quickly match incoming documents against standing XPath subscriptions

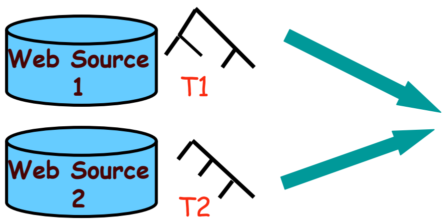


(X/Yfilter, Xtrie, etc.)

- Essentially, simple selection queries over a stream of XML records!
- No work on more complex XML stream queries
  - For example, queries trying to ***correlate*** different XML data streams

Processing XML Data Streams

- Example XML stream correlation query: *Similarity-Join*

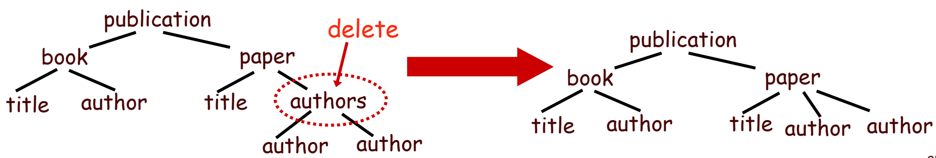


$$|SimJoin(S1, S2)| = |\{(T1,T2) \in S1 \times S2:\ dist(T1,T2) \le \theta\}|$$

Degree of content similarity between streaming XML sources

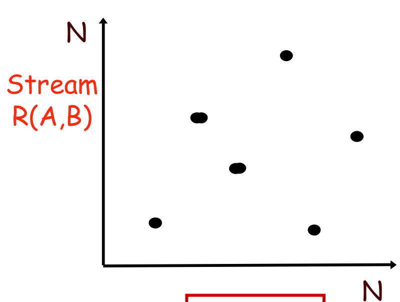
Different data representation for same information (DTDs, optional elements)

- Correlation metric: *Tree-edit distance ed(T1,T2)*
  - Node relabels, inserts, deletes - also, allow for *subtree moves*



How About AMS Sketches?

- Randomized linear projections (aka AMS sketches) are useful for points over a numeric vector space
  - *Not* structured objects over a complex metric space (tree-edit distance)



$O(N^2)$

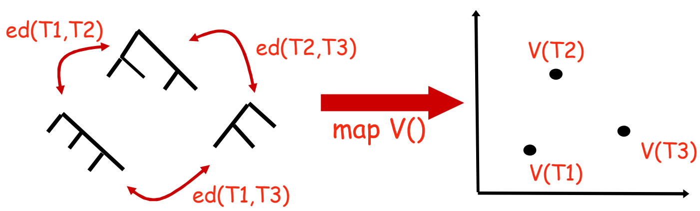
Atomic Sketch

$$X_R = \sum_{(i,j)} freq(i,j) \cdot \xi_{ij}$$

$O(\log N)$

Our Approach [GK03]

- *Key idea:* Build a *low-distortion embedding* of streaming XML and the tree-edit distance metric in a multi-d normed vector space



Distances are (approximately) preserved in the image vector space!!

- Given such an embedding, sketching techniques now become relevant in the streaming XML context!
  - E.g., use AMS sketches to produce synopses of the data distribution in the image vector space

Our Approach [GK03] *(cont.)*

- Construct low-distortion embedding for tree-edit distance over *streaming* XML documents -- Requirements:
  - *Small space/time*
  - *Oblivious:* Can compute V(T) independent of other trees in the stream(s)
  - Bourgain's Lemma is inapplicable!
- *First* algorithm for low-distortion, oblivious embedding of the tree-edit distance metric in small space/time
  - Fully deterministic, embed into L1 vector space
  - Bound of $O(\log^2 n \log^* n)$ on distance distortion for trees with $\leq$ n nodes

$$\| V(S) - V(T) \|_1 = \sum | V(S)[i] - V(T)[i] | = O(\log^2 n \log^* n) \cdot ed(S,T)$$

- *Worst-case bound!* Distortions *much smaller* over real-life data
  - See TODS'05 paper…

Our Approach [GK03] *(cont.)*

- Applications in XML stream query processing
  - Combine our embedding with existing pseudo-random linear-projection sketching techniques
  - Build a small-space sketch synopsis for a massive, streaming XML data tree
    - Concise surrogate for tree-edit distance computations
  - Approximating tree-edit distance similarity joins over XML streams in small space/time
- *First* algorithmic results on correlating XML data in the streaming model
- Other important algorithmic applications for our embedding result
  - Approximate tree-edit distance in (near-linear) $O(n \log^* n)$ time

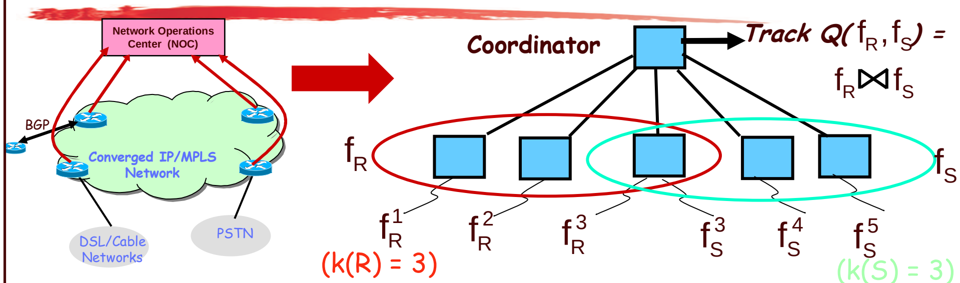
29

Talk Outline

- Introduction & Motivation
- Data Stream Computation Model
- A Basic Sketching Tool for Streams
  - Linear-Projection (aka AMS) Sketches
    - *Applications:* Join/Multi-Join Queries, Wavelets, Correlating XML streams
- Tracking Queries over *Distributed* Streams
  - Communication-efficient, sketch-based approach
- Conclusions & Future Research Directions

30

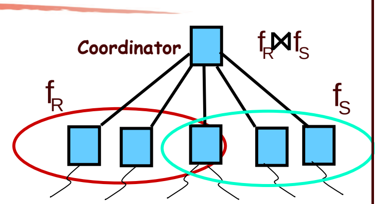
Tracking *Distributed* Data Streams



- Large-scale event monitoring: *Inherently distributed!*
  - Each stream f_R is distributed across a (sub)set of remote sites
    - $f_R = f_R^i \bigcup f_R^j \bigcup f_R^k \bigcup \ldots$, $k(R) = |sites(R)|$
    - E.g., stream of UDP packets through a subset of edge routers
- *Goal is "Holistic" Monitoring:* Effective tracking of a *global quantity/query* over the union of a *distributed* collection of streams
  - *Composability* of sketches makes them ideal for distributed computation
  - *BUT,* interested in *continuous query tracking* rather than "one-shot" computation
    - Provide approximate answer (+ guarantees) at coordinator *at all times!*

Tracking *Distributed* Data Streams

- Need solutions that are not only space/time-efficient, but also *communication-efficient!*
  - Optimize communication overhead involved for a given accuracy guarantee
  - Minimize monitoring overhead on network; Maximize node battery life (e.g., sensornets)
  - Can use sketches at remote sites, but naïve schemes that "push" to coordinator on every update won't work



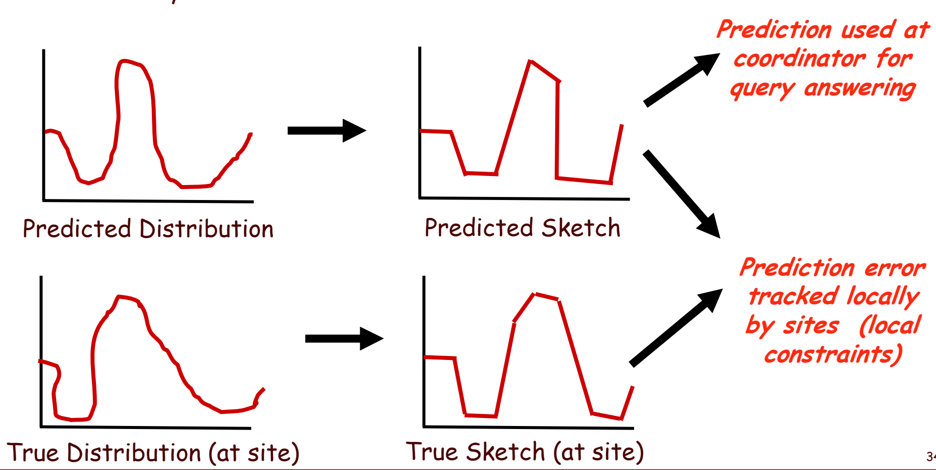
- ***Key Design Desiderata***
  - *Minimal global information exchanges*
    - Avoid sharing global information and expensive "global resync" steps
  - *Summary-based information exhange*
    - Remote sites only communicate *concise sketches* on locally-observed streams
  - *Stability*
    - No communication as long as local-stream behavior is stable, or *predictable*

Our Query-Tracking Solution [CG05]

- General approach: *"In-Network" Processing*
  - "Push" part of the query tracking to individual remote sites
  - Each site tracks *local constraints* on the deviation of its locally-observed stream(s) from the corresponding picture at the coordinator
  - Contact coordinator with updated sketch information only if local constraints are violated
- Overall error guarantee at coordinator is a function $g(\varepsilon, \theta)$
  - ε = error in local sketch summaries at each remote site
    - Each site maintains ε-approximate AMS sketches on local stream(s)
  - θ = upper bound on the local-stream deviation from the picture at the coordinator
    - Local constraints tracked at each site specify a θ-deviation (in L2 norm) of the local stream sketch wrt the sketch at the coordinator
- Exact form of $g(\varepsilon, \theta)$ depends on the specific query Q being tracked
    - *BUT,* local site constraints remain the same (i.e., L2 deviation of local sketches)

Sketch-Prediction Models [CG05]

- To ensure *Stability*, our solution relies on concise *sketch-prediction models*
  - Try to predict how the local-stream distribution (and its sketch) will evolve over time
  - Built locally at remote sites and communicated to coordinator

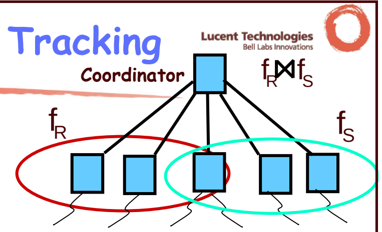


Sketch-Prediction Models

- Sketch-prediction models are *simple, concise* models of local-stream behavior
  - Information communicated to coordinator (in addition to local stream sketches) so that both site & coordinator are "in sync"
- Different alternatives
  - *Empty model*: Local stream does not change since last exchange (no additional information necessary)
  - *Velocity/Acceleration Models:* Predict change in the distribution using "velocity" and "acceleration" vectors built on recent history
    - Velocity model: $f^i(t) = f^i(t_{prev}) + \Delta t \cdot v^i$
    - Velocity vector v^i based on window of recent updates at site i
    - By *linearity* of sketching we only need to communicate a sketch of v^i to the coordinator
      - Only concise, sketch information is exchanged

35

Sketch-Based Distributed Join Tracking

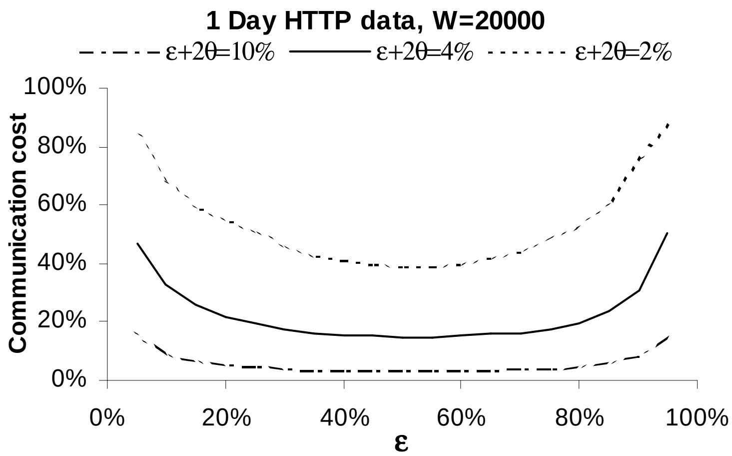


- Continuously track $Q = f_R \bowtie f_S$ (within specified error guarantees) at coordinator
- ***Protocol***
  - *Each remote site $i \in$ sites(R)*
    - Build and maintain ε-approximate AMS sketch on f^i_R
    - On each update, check that the relative L2-norm deviation of the *true local sketch* from the *predicted sketch* is bounded by $\theta / \sqrt{k(R)}$
    - If not, send current sketch and (perhaps) model info to coordinator
  - *Similarly for each remote site $j \in$ sites(S)*
  - *Coordinator:* Compute approximate answer using the *predicted sketches*
- ***Key Result:*** Protocol guarantees a $g(\varepsilon, \theta) \approx (\varepsilon + 2\theta)$ -approximate join size estimate at the coordinator *at all times*
- Approach and results extend naturally to other queries and streaming models
  - Distributed multi-join aggregates, wavelet/histogram maintenance, . . .
  - Sliding window model, exponential-decay model

36

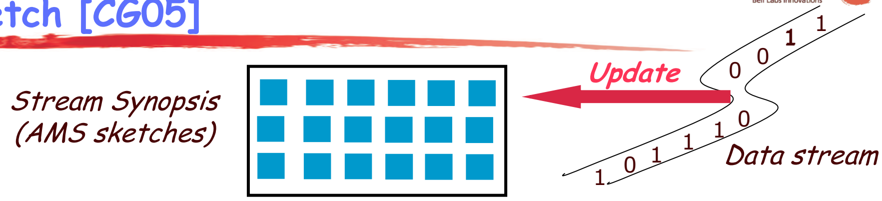
Accuracy Tradeoffs

- Given an overall error requirement $\varepsilon + 2\theta$ decide optimal "split" between ε, θ
  - E.g., larger ε, smaller θ => smaller sketches but communicate more frequently (smaller local-deviation tolerance)
  - Analysis of optimal settings in restricted cases (e.g., *Empty* model)



37

Small-Time Updates: The Fast-AMS Sketch [CG05]



- Synopsis-update times are typically $\Omega(|\text{synopsis}|)$ -- fine as long as synopsis sizes are small (polylog), *BUT* . . .
  - Small synopses are often *impossible* (strong communication-complexity lower bounds)
    - E.g., join/multi-join aggregates
- Synopsis size may not be the crucial limiting factor (PCs with Gigabytes of RAM)

- *Fast-AMS Sketch:* Organize the atomic AMS counters into hash-table buckets
  - Same space/accuracy tradeoff as basic AMS (in fact, slightly better☺)
  - *BUT, guaranteed logarithmic update times* (regardless of synopsis size)
    - Only touch a few counters per update

38

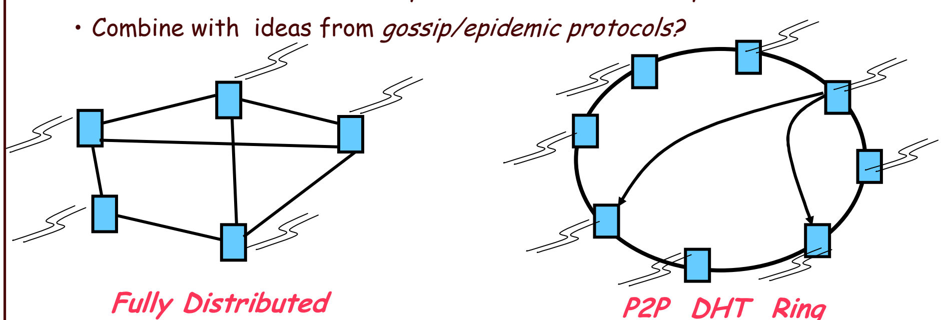
Conclusions

- Analyzing massive data streams: Real problem with several real-world applications
  - Fundamentally rethink data management under stringent constraints
  - Single-pass algorithms with limited memory/CPU-time resources
- *Sketches based on pseudo-random linear projections* are a viable technique for a variety of streaming tasks
  - Limited-space randomized approximations
  - *Probabilistic guarantees* on the quality of the approximate answer
  - *Delete-proof* (supports insertion *and* deletion of records)
  - *Composable* (computed remotely and combined later)
  - *General-purpose* (join/multi-join aggregates, wavelets, histograms, XML similarity joins, . . .)
  - *Effective Query Tracking over Distributed-Streams*

39

Future Work on Distributed-Stream Tracking

- *Stream tracking over different distributed architectures*
  - Our techniques and results extend naturally to *general, multi-level hierarchies* (tree structures, coordinator at the root)
  - Other distributed models: fully distributed, P2P overlays, . . .
    - Combine with ideas from *gossip/epidemic protocols?*



- *Foundations of distributed-streams model*
  - Lower bounds for different distributed-tracking problems?

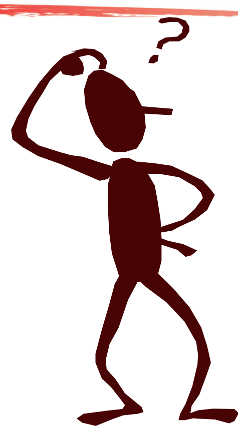
40

Future Research Directions Laundry List

- Sketches/synopses for richer types of streaming data and queries
  - Spatial/temporal data streams, mining/querying streaming graphs, . . .
- Other metric-space embeddings in the streaming model
- Stream-data processing architectures and query languages
  - Progress: Aurora, STREAM, Telegraph, . . .
- Integration of streams and static relations
  - Effect on DBMS components (e.g., query optimizer)
- Novel, important application domains
  - Sensor networks, financial analysis, security (network DDoS, virus/worm detection), . . .

41

Thank you!



http://www.bell-labs.com/~minos/

minos@research.bell-labs.com

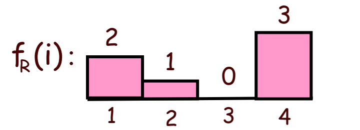
42

Using Sketches to Answer SUM Queries

- Problem: Compute answer for query $SUM_B(R \bowtie_A S) = \sum_i f_R(i) \cdot SUM_S(i)$
  - $SUM_S(i)$ is sum of B attribute values for records in S for whom S.A = i
- Sketch-based solution
  - Compute random variables X_R and X_S

Stream R.A: 4 1 2 4 1 4

$X_R = X_R + \xi_4$

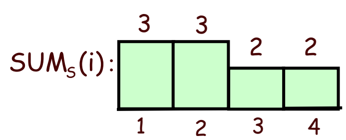


$X_R = \sum_i f_R(i)\xi_i = 2\xi_1 + \xi_2 + 3\xi_4$

Stream S: A 3 1 2 4 2 3

B 1 3 2 2 1 1

$X_S = X_S + 3\xi_1$



$X_S = \sum_i SUM_S(i)\xi_i = 3\xi_1 + 3\xi_2 + 2\xi_2 + 2\xi_4$

  - Return $X = X_R X_S$ $(E[X] = SUM_B(R \bowtie_A S))$

43

Stream Wavelet Approximation using AMS Sketches [GKMS01]

- Single-join approximation with sketches [AGMS99]
  - Construct approximation to $|R1 \bowtie R2| = \sum f_1(i) f_2(i)$ within a relative error of ε with probability $\geq 1-\delta$ using space $O(\log N \cdot \log(1/\delta)/(\varepsilon^2 \lambda^2))$, where

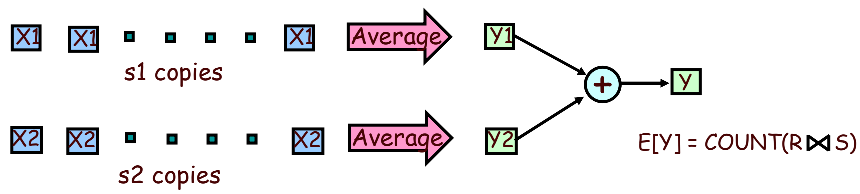
$$\lambda \leq \frac{|\sum f_1(i) f_2(i)|}{\sqrt{\sum f_1^2(i) \cdot \sum f_2^2(i)}}$$ = |R1 $\bowtie$ R2| / Sqrt($\prod$ self-join sizes)

- Observation: $|R1 \bowtie R2| = \sum f_1(i) f_2(i) = \langle f_1, f_2 \rangle$ *= inner product!!*
  - General result for inner-product approximation using sketches
- Other inner products of interest: *Haar wavelet coefficients!*
  - Haar wavelet decomposition = inner products of signal/distribution with specialized (wavelet basis) vectors

44

Space Allocation Among Partitions

- Key Idea: Allocate more space to sketches for partitions with higher variance



$$Var[Y] = \frac{Var[X1]}{s1} + \frac{Var[X2]}{s2} \le \frac{\varepsilon^2\, COUNT^2}{8}$$

- Example: Var[X1]=20K, Var[X2]=2K
  - For s1=s2=20K, Var[Y] = 1.0 + 0.1 = 1.1
  - For s1=25K, s2=8K, Var[Y] = 0.8 + 0.25 = 1.05

45

Sketch Partitioning Problems

- Problem 1: Given sketches X1, ..., Xk for partitions P1, ..., Pk of the join attribute domain, what is the space sj that must be allocated to Pj (for sj copies of Xj) so that

$$Var[Y] = \frac{Var[X1]}{s1} + \frac{Var[X2]}{s2} + \cdots + \frac{Var[Xk]}{sk} \le \frac{\varepsilon^2\, COUNT^2}{8}$$

and $\sum_j sj$ is minimum

- Problem 2: Compute a partitioning P1, ..., Pk of the join attribute domain, and space sj allocated to each Pj (for sj copies of Xj) such that

$$Var[Y] = \frac{Var[X1]}{s1} + \frac{Var[X2]}{s2} + \cdots + \frac{Var[Xk]}{sk} \le \frac{\varepsilon^2\, COUNT^2}{8}$$

and $\sum_j sj$ is minimum

- Solutions also apply to *dual problem* (Min. variance for fixed space)

46

Optimal Space Allocation Among Partitions

- Key Result (Problem 1): Let X1, ..., Xk be sketches for partitions P1, ..., Pk of the join attribute domain. Then, allocating space

$$sj = \frac{8\sqrt{Var(Xj)}(\sum_j \sqrt{Var(Xj)})}{\varepsilon^2\ COUNT^2}$$

to each Pj (for sj copies of Xj) ensures that $Var[Y] \le \frac{\varepsilon^2\ COUNT^2}{8}$ and $\sum_j sj$ is minimum

- Total sketch space required: $\sum_j sj = \frac{8(\sum_j \sqrt{Var(Xj)})^2}{\varepsilon^2\ COUNT^2}$

- Problem 2 (Restated): Compute a partitioning P1, ..., Pk of the join attribute domain such that $\sum_j \sqrt{Var(Xj)}$ is minimum
  - Optimal partitioning P1, ..., Pk minimizes total sketch space

47

Binary-Join Queries: Binary Space Partitioning

- Problem: For COUNT(R $\bowtie_A$ S), compute a partitioning P1, P2 of A's domain {1, 2, ..., N} such that $\sqrt{Var(X1)} + \sqrt{Var(X2)}$ is minimum
  - Note: $Var(Xj) \approx 2\sum_{i \in Pj} f_R(i)^2 \sum_{i \in Pj} f_S(i)^2$

- Key Result (due to Breiman): For an optimal partitioning P1, P2,

$$\forall i1 \in P1, \forall i2 \in P2, \quad \frac{f_R(i1)}{f_S(i1)} < \frac{f_R(i2)}{f_S(i2)}$$

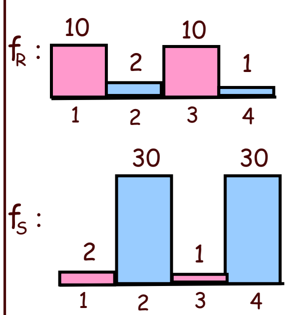
- Algorithm
  - Sort values i in A's domain in increasing value of $\frac{f_R(i)}{f_S(i)}$
  - Choose partitioning point that minimizes

$$\sqrt{2\sum_{i \in P1} f_R(i)^2 \sum_{i \in P1} f_S(i)^2} + \sqrt{2\sum_{i \in P2} f_R(i)^2 \sum_{i \in P2} f_S(i)^2}$$

48

Binary Sketch Partitioning Example

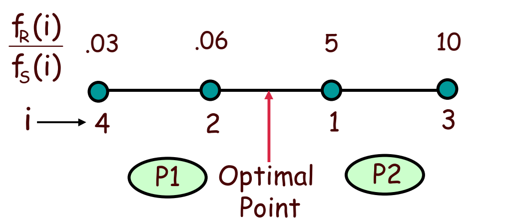
<u>Without Partitioning</u>



$$\text{Space} = \frac{8 \cdot Var[X]}{\varepsilon^2\ COUNT^2}$$

$Var[X] \approx$ 720K

<u>With Optimal Partitioning</u>



$$\text{Space} = \frac{8(\sum_j \sqrt{Var(Xj)})^2}{\varepsilon^2\ COUNT^2}$$

$$(\sum_j \sqrt{Var(Xj)})^2 \approx (\sqrt{18000} + \sqrt{2000})^2 \approx 32K$$

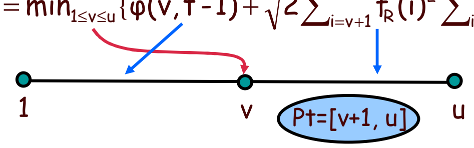
$(Var[X1] \approx 18K,\ Var[X2] \approx 2K)$

49

Binary-Join Queries: K-ary Sketch Partitioning

- <u>Problem:</u> For COUNT($R \bowtie_A S$), compute a partitioning P1, P2, ..., Pk of A's domain such that $\sum_j \sqrt{Var(Xj)}$ is minimum
- Previous result (for 2 partitions) generalizes to k partitions
- Optimal k partitions can be computed using Dynamic Programming
  - Sort values i in A's domain in increasing value of $\frac{f_R(i)}{f_S(i)}$
  - Let $\varphi(u,t)$ be the value of $\sum_j \sqrt{2\sum_{i \in Pj} f_R(i)^2 \sum_{i \in Pj} f_S(i)^2}$ when [1,u] is split optimally into t partitions P1, P2, ..., Pt

$$\varphi(u,t) = \min_{1 \le v \le u}\{\varphi(v,t-1) + \sqrt{2\sum_{i=v+1}^{u} f_R(i)^2 \sum_{i=v+1}^{u} f_R(i)^2}\}$$



- Time complexity: $O(kN^2)$

50

Sketch Partitioning for Multi-Join Queries

- Problem: For COUNT($R \bowtie_A S \bowtie_B T$), compute a partitioning $P_1^A, P_2^A, \ldots, P_{k_A}^A$ ($P_1^B, P_2^B, \ldots, P_{k_B}^B$) of A(B)'s domain such that $k_A k_B < k$, and the following is minimum

$$\psi = \sum_{P_i^A} \sum_{P_j^B} \sqrt{Var(X_{(P_i^A, P_j^B)})}$$

- Partitioning problem is NP-hard for more than 1 join attribute
- If join attributes are independent, then possible to compute optimal partitioning
  - Choose k1 such that allocating k1 partitions to attribute A and k/k1 to remaining attributes minimizes ψ
  - Compute optimal k1 partitions for A using previous dynamic programming algorithm

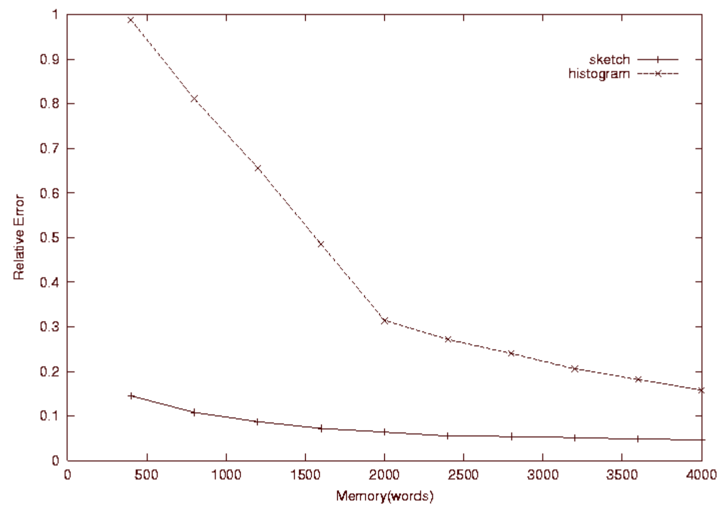
51

Experimental Study

- Summary of findings
  - Sketches are superior to 1-d (equi-depth) histograms for answering COUNT queries over data streams
  - Sketch partitioning is effective for reducing error
- Real-life Census Population Survey data sets (1999 and 2001)
  - Attributes considered:
    - Income (1:14)
    - Education (1:46)
    - Age (1:99)
    - Weekly Wage and Weekly Wage Overtime (0:288416)
- Error metric: relative error ($\frac{|actual - approx|}{actual}$)

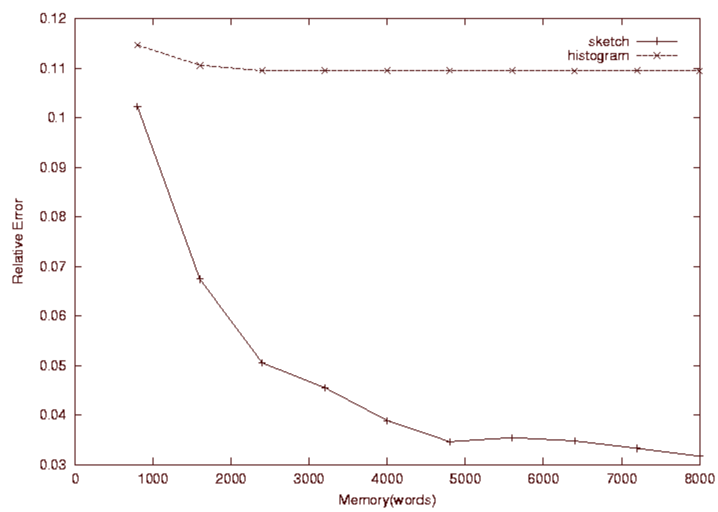
52

Join (Weekly Wage)



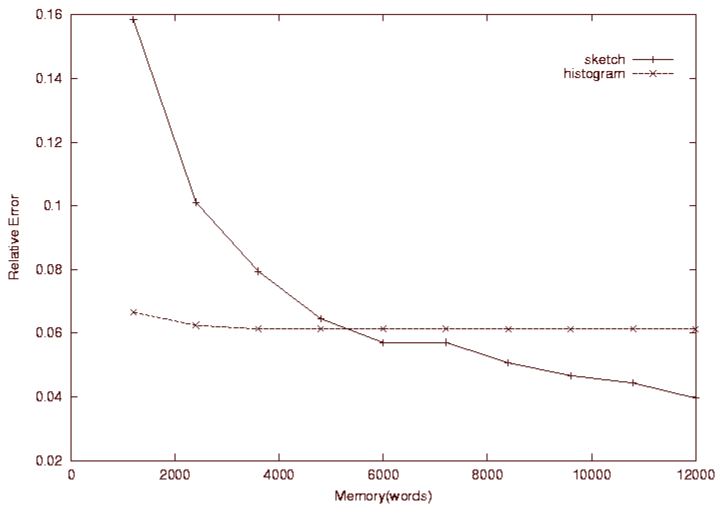
53

Join (Age, Education)



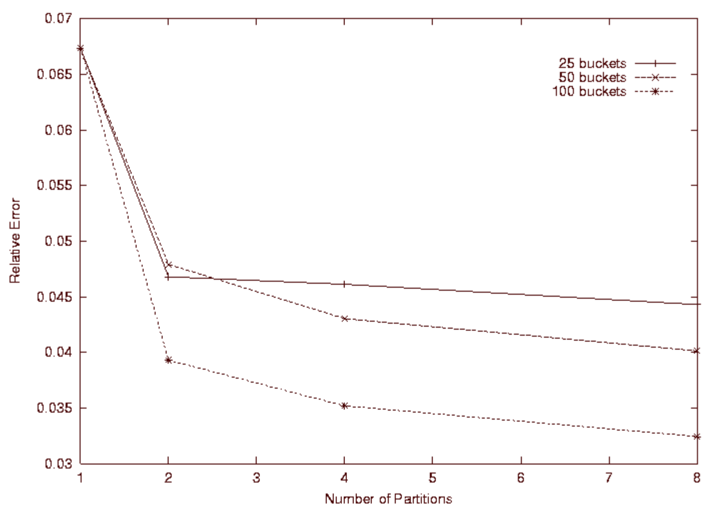
54

Star Join (Age, Education, Income)



55

Join (Weekly Wage Overtime = Weekly Wage)



56

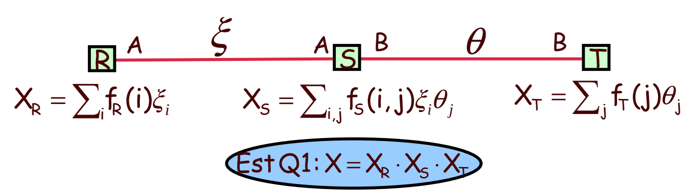
More work on Sketches...

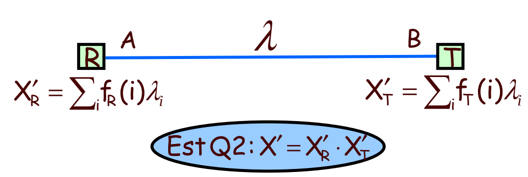
- Low-distortion vector-space embeddings (JL Lemma) [Ind01] and applications
  - E.g., approximate nearest neighbors [IM98]
- Wavelet and histogram extraction over data streams [GGI02, GIM02, GKMS01, TGIK02]
- Discovering patterns and periodicities in time-series databases [IKM00, CIK02]
- Quantile estimation over streams [GKMS02]
- Distinct value estimation over streams [CDI02]
- Maintaining *top-k* item frequencies over a stream [CCF02]
- Stream norm computation [FKS99, Ind00]
- Data cleaning [DJM02]

57

Sketching for Multiple Standing Queries

- Consider queries Q1 = COUNT(R $\bowtie_A$ S $\bowtie_B$ T) and Q2 = COUNT(R $\bowtie_{A=B}$ T)
- Naive approach: construct separate sketches for each join
  - ξ, θ, λ are independent families of pseudo-random variables

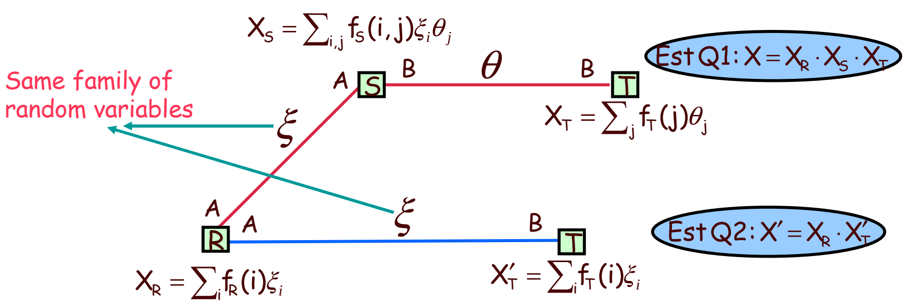




58

Sketch Sharing

- Key Idea: Share sketch for relation R between the two queries
  - Reduces space required to maintain sketches



- *BUT*, cannot also share the sketch for T !
  - Same family on the join edges of Q1

59

Sketching for Multiple Standing Queries

- Algorithms for sharing sketches and allocating space among the queries in the workload
  - Maximize sharing of sketch computations among queries
  - Minimize a cumulative error for the given synopsis space
- Novel, interesting combinatorial optimization problems
  - Several NP-hardness results :-)
- Designing effective heuristic solutions

60

Talk Outline

- Introduction & Motivation
- Data Stream Computation Model
- Two Basic Sketching Tools for Streams
  - Linear-Projection (aka AMS) Sketches
    - *Applications:* Join/Multi-Join Queries, Wavelets
  - Hash (aka FM) Sketches
    - *Applications:* Distinct Values, Set Expressions
- Extensions
  - Correlating XML data streams
- Conclusions & Future Research Directions

61

Distinct Value Estimation

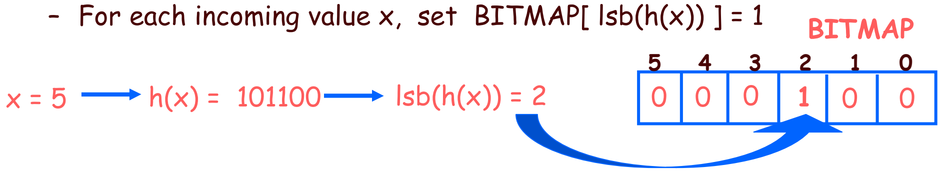
- Problem: Find the *number of distinct values* in a stream of values with domain [0,...,N-1]
  - Zeroth frequency moment F_0 , L0 (Hamming) stream norm
  - Statistics: number of *species or classes* in a population
  - Important for query optimizers
  - *Network monitoring:* distinct destination IP addresses, source/destination pairs, requested URLs, etc.
- Example (N=64) Data stream: 3 0 5 3 0 1 7 5 1 0 3 7

  Number of distinct values: 5
- Hard problem for random sampling! [CCMN00]
  - Must sample almost the entire table to guarantee the estimate is within a factor of 10 with probability > 1/2, regardless of the estimator used!

62

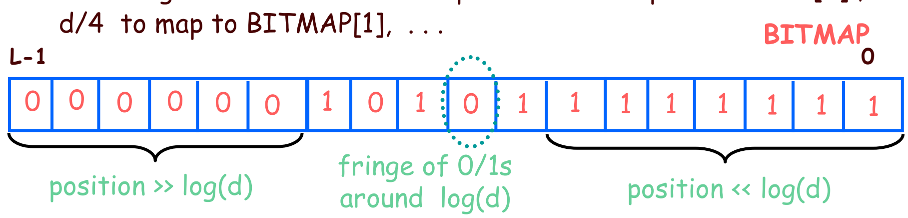
Hash (aka FM) Sketches for Distinct Value Estimation [FM85]

- Assume a hash function h(x) that maps incoming values x in [0,…, N-1] *uniformly* across [0,…, 2^L-1], where L = O(logN)
- Let lsb(y) denote the position of the least-significant 1 bit in the binary representation of y
  - A value x is mapped to lsb(h(x))
- Maintain *Hash Sketch* = BITMAP array of L bits, initialized to 0
  - For each incoming value x, set BITMAP[lsb(h(x))] = 1



Hash (aka FM) Sketches for Distinct Value Estimation [FM85]

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



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

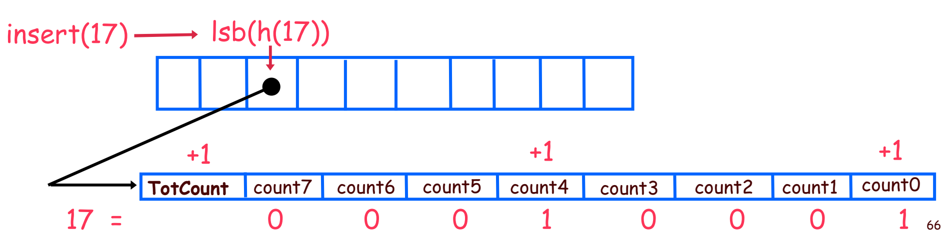
Hash Sketches for Distinct Value Estimation

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

65

Processing Set Expressions over Update Streams [GGR03]

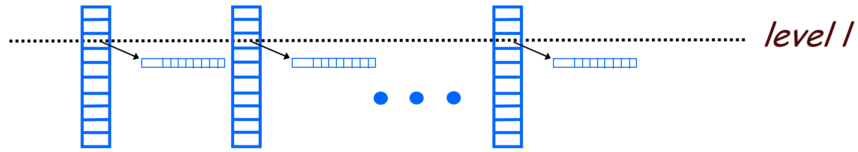
- Estimate cardinality of *general set expressions* over streams of updates
  - E.g., number of distinct (source,dest) pairs seen at both R1 and R2 but not R3? | (R1 ∩ R2) – R3 |
- *2-Level Hash-Sketch (2LHS) stream synopsis:* Generalizes FM sketch
  - *First level:* $\Theta(\log N)$ buckets with exponentially-decreasing probabilities (using lsb(h(x)), as in FM)
  - *Second level:* Count-signature array (logN+1 counters)
    - One "total count" for elements in first-level bucket
    - logN "bit-location counts" for 1-bits of incoming elements



66

Processing Set Expressions over Update Streams: Key Ideas

- Build several independent 2LHS, fix a level l, and look for *singleton first-level buckets* at that level l



- Singleton buckets and singleton element (in the bucket) are easily identified using the *count signature*

Singleton bucket count signature

Total=11	0	0	0	0	11	0	11	0

→ Singleton element = $1010_2 = 10$

- Singletons discovered form a *distinct-value sample* from the union of the streams
  - Frequency-independent, each value sampled with probability $1/2^{l+1}$
- Determine the fraction of *"witnesses"* for the set expression E in the sample, and scale-up to find the estimate for |E|

Example: Set Difference, |A-B|

- Parallel (same hash function), independent 2LHS synopses for input streams A, B
- Assume robust estimate $\hat{u}$ for $|A \cup B|$ (using known FM techniques)
- Look for buckets that are *singletons for* $A \cup B$ at level $l \approx \lceil \log \hat{u} \rceil$
  - Prob[singleton at level l] > constant (e.g., 1/4)
  - Number of singletons (i.e., *size of distinct sample*) is at least a constant fraction (e.g., > 1/6) of the number of 2LHS (w.h.p.)
- *"Witness" for set difference A-B:* Bucket is singleton for stream A and empty for stream B
  - Prob[witness | singleton] = $|A-B| / |A \cup B|$
- Estimate for |A-B| = $\frac{\#\text{witnesses for A-B}}{\#\text{singleton buckets}} \times \hat{u}$

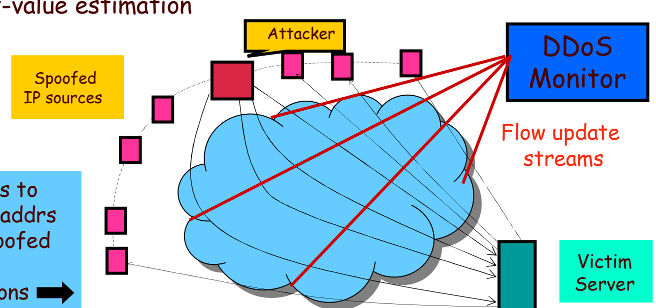
Estimation Guarantees

- Our set-difference cardinality estimate is within a relative error of ε with probability $\geq 1-\delta$ when the number of 2LHS is $O(\frac{|A \cup B| \log(1/\delta)}{|A-B| \varepsilon^2})$
- *Lower bound* of $\Omega(\frac{|A \cup B|}{|A-B| \varepsilon})$ space, using communication-complexity arguments
- Natural generalization to arbitrary set expressions E = f(S1,…,Sn)
  - Build parallel, independent 2LHS for each S1,…, Sn
  - Generalize "witness" condition (inductively) based on E's structure
  - (ε, δ) estimate for |E| using $O(\frac{|S1 \cup \ldots \cup Sn| \log(1/\delta)}{|E| \varepsilon^2})$ 2LHS synopses
- *Worst-case bounds!* Performance in practice is much better [GGR03]

69

Application: Detecting TCP-SYN-Flooding DDoS Attacks

- Monitor potential DDoS activity over large ISP network – cannot maintain state for each potential destination/victim
- *Top-k based on traffic volume* gives high traffic destinations (e.g., Yahoo!)
  - Attack traffic may not be high
  - Cannot distinguish attacks from "flash crowds"
- *"Right" metric:* Top-k destinations wrt number of distinct connecting sources
  - Deletions to remove legitimate TCP connections from synopses
- Novel, space/time efficient, hash-based streaming algorithm – 2LHS used as a component for distinct-value estimation



Attack Mechanism

- Flood of small SYN packets to victim from spoofed source addrs
- SYN-ACK responses to spoofed IP sources
- Many "half-open" connections ➡ Resources exhausted

70

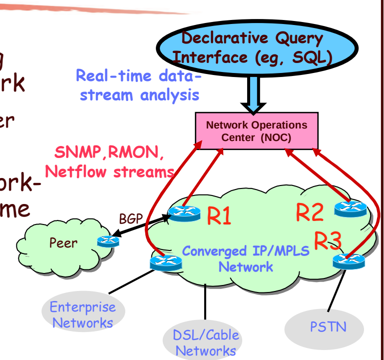
Set Expressions to Sketch Expressions

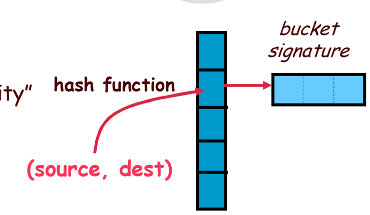
- Given set expression E = f(S1,…,Sn), level of inference l
  - Again, look for buckets that are singletons for the union of S1,…, Sn at level l
- *"Witness" Condition for E:* Create boolean expression B(E) over parallel sketches inductively
  1. Replace Si by isSingleton(sketch(Si), l)
  2. Replace E1 ⋂ E2 by B(E1) AND B(E2)
  3. Replace E1-E2 by B(E1) AND (NOT B(E2))
  4. Replace E1⋃ E2 by B(E1) OR B(E2)
- Then, Prob[witness | singleton] = |E| / |S1⋃ … ⋃ Sn|

71

Application: Robust, Real-Time DDoS Attack Detection

- ***Key Ideas***
  - Provide *declarative* interface for specifying DDoS/anomaly queries over large ISP network
    - E.g., *top-k* destinations with respect to number of distinct connecting sources
  - *Continuously track* these queries over network-measurement data streams in small space/time
- ***Innovations***
  - Small-footprint, hash-based synopses for approximate DDoS query tracking
  - Small update time per network-stream tuple
    - Log/poly-log space & time tracking
  - Strong, probabilistic approximation guarantees
    - "within 2% of exact answer with high probability"
  - Robust, real-time detection of DDoS anomaly conditions in the network
    - E.g., tracking "half-open" connections to distinguish DDoS attacks from flash-crowds





Hash-Sketch Synopsis

72

Embedding Algorithm

- *Key Idea:* Given an XML tree T, build a *hierarchical parsing structure* over T by intelligently grouping nodes and contracting edges in T
  - At parsing level i: T(i) is generated by grouping nodes of T(i-1) (T(0) = T)
  - Each node in the parsing structure (T(i), for all i = 0, 1, ...) corresponds to a connected subtree of T
  - Vector image V(T) is basically the *characteristic vector* of the resulting multiset of subtrees (in the entire parsing structure)

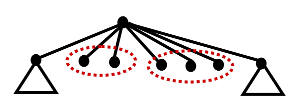
V(T)[x] = no. of times subtree x appears in the parsing structure for T

- Our parsing guarantees
  - O(log|T|) parsing levels (constant-fraction reduction at each level)
  - V(T) is *very sparse:* Only O(|T|) non-zero components in V(T)
    - Even though dimensionality = $O((4|\sigma|)^n)$ (σ = label alphabet)
    - Allows for effective sketching
  - V(T) is constructed in time $O(|T|\log^*|T|)$

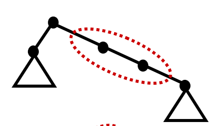
73

Embedding Algorithm *(cont.)*

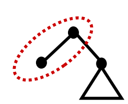
- Node grouping at a given parsing level T(i): Create groups of 2 or 3 nodes of T(i) and merge them into a single node of T(i+1)
  - 1. Group maximal sequence of contiguous leaf children of a node



  - 2. Group maximal sequence of contiguous nodes in a chain



  - 3. Fold leftmost lone leaf child into parent

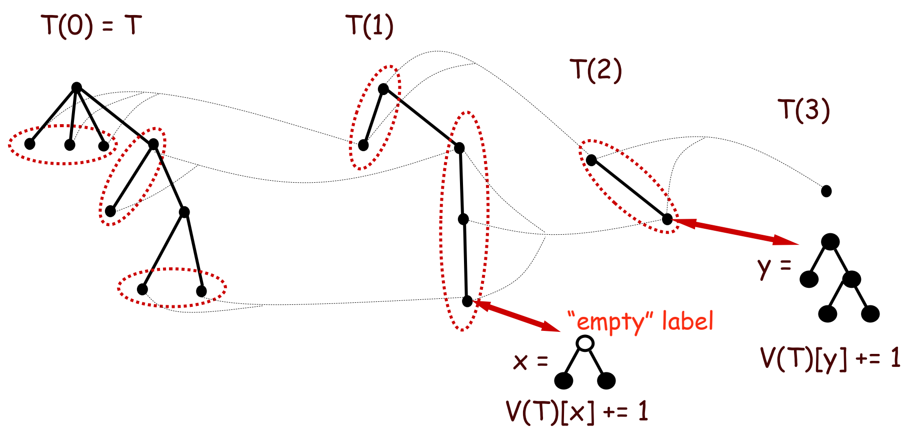


- Grouping for Cases 1,2: Deterministic coin-tossing process of Cormode and Muthukrishnan [SODA'02]
  - *Key property:* Insertion/deletion in a sequence of length k only affects the grouping of nodes in a radius of $\log^* k + 5$ from the point of change

74

Embedding Algorithm *(cont.)*

- Example hierarchical tree parsing



- O(log|T|) levels in the parsing, build V(T) in time $O(|T|\log^*|T|)$

75

Main Embedding Result

- *Theorem:* Our embedding algorithm builds a vector V(T) with O(|T|) non-zero components in time $O(|T|\log^*|T|)$; further, given trees T, S with n = max{|T|, |S|}, we have:

$$ed(S,T) \le 5 \cdot \|V(S) - V(T)\|_1 = O(\log^2 n \log^* n) \cdot ed(S,T)$$

- Upper-bound proof highlights
  - *Key Idea:* Bound the size of "influence region" (i.e., set of affected node groups) for a tree-edit operation on T (=T(0)) at each level of parsing
    - We show that this set is of size $O(i \log^* n)$ at level i
  - Then, it is simple to show that any tree-edit operation can change $\|V(T)\|_1$ by at most $O(\log^2 n \log^* n)$
    - L1 norm of subvector at level i changes by at most O(|influence region|)

76

Main Embedding Result *(cont.)*

- Lower-bound proof highlights
  - *Constructive:* "Budget" of at most $5 \cdot \| V(S) - V(T) \|_1$ tree-edit operations is sufficient to convert the parsing structure for S into that for T
    - Proceed bottom up, level-by-level
    - At bottom level (T(0)), use budget to insert/delete appropriate labeled nodes
    - At higher levels, use subtree moves to appropriately arrange nodes
- See PODS'03 paper for full details . . .

77

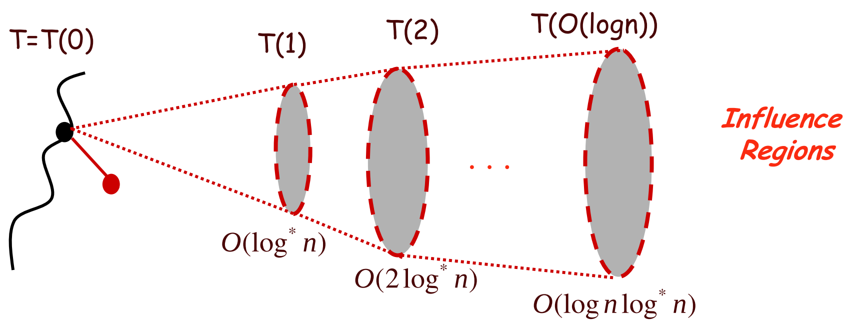
Sketching a Massive, Streaming XML Tree

- *Input:* Massive XML data tree T (n = |T| >> available memory), seen in preorder (e.g., SAX parser output)
- *Output:* Small space surrogate (vector) for high-probability, approximate tree-edit distance computations (to within our distortion bounds)
- *Theorem:* Can build a $O(\log \frac{1}{\delta})$ -size sketch vector of V(T) for approximate tree-edit distance computations in $O(d \log^2 n (\log^* n)^2)$ space and $O(\log d \log^2 n (\log^* n)^2)$ time per element
  - d = depth of T, δ = probabilistic confidence in ed() approximation
  - XML trees are typically "bushy" (d<<n or d = O(polylog(n)))

78

Sketching a Massive, Streaming XML Tree *(cont.)*

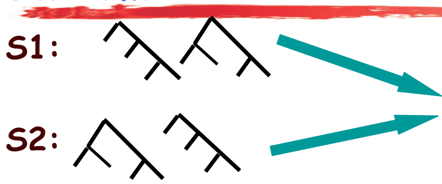
- *Key Ideas*
  - Incrementally parse T to produce V(T) as elements stream in
  - Just need to retain the *influence region* nodes for each parsing level and for each node in the current root-to-leaf path



  - While updating V(T), also produce an *L1 sketch* of the V(T) vector using the techniques of Indyk [FOCS'00]

79

Approximate Similarity Joins over XML Streams



|SimJoin(S1, S2)| =
|{(T1,T2)$\in$S1xS2: ed(T1,T2) $\leq \theta$ }|

- *Input:* Long streams S1, S2 of N (short) XML documents ($\leq$ b nodes)
- *Output:* Estimate for |SimJoin(S1, S2)|

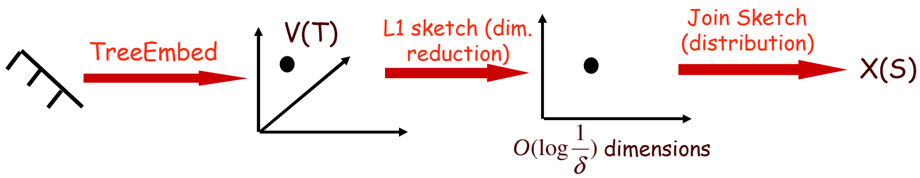
- *Theorem:* Can build an atomic sketch-based estimate for |SimJoin(S1, S2)| where distances are approximated to within $O(\log^2 b \log^* b)$ in space $O(b + \log\frac{1}{\delta}\log N)$ and $O(\frac{1}{\delta} + b\log^* b)$ time per document
  - δ = probabilistic confidence in distance estimates

80

Approximate Similarity Joins over XML Streams *(cont.)*

Lucent Technologies
Bell Labs Innovations

- *Key Ideas*
  - Our embedding of streaming document trees, plus two distinct levels of sketching
    - One to reduce L1 dimensionality, one to capture the data distribution (for joining)
    - Finally, similarity join in lower-dimensional L1 space

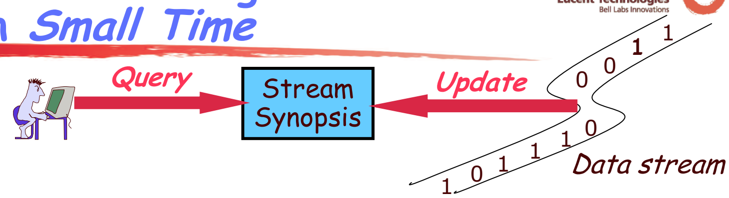


    - Some technical issues: high-probability L1 dimensionality reduction is not possible, sketching for L1 similarity joins
    - Details in the paper . . .

81

Future Work: Tracking Continuous Streams in *Small Time*

Lucent Technologies
Bell Labs Innovations



- Update/Query times are typically $\Omega(|\text{synopsis}|)$ -- fine as long as synopsis sizes are small (polylog), BUT…
  - Small synopses are often *impossible* (strong communication-complexity lower bounds)
    - E.g., set expressions, joins, . . .
- Synopsis size may not be the crucial limiting factor (PCs with Gigabytes of RAM)
- *Guaranteed small (polylog) update/query times* are critical for high-speed streams
  - *Time-efficient* streaming algorithms -- $\Omega(|\text{synopsis}|)$ times are not adequate!
  - Have some initial results for small-time tracking of set expressions and joins

82