

# Sketching Techniques for Massive Data Streams

*Minos Garofalakis*

*Internet Management Research Department  
Bell Labs, Lucent Technologies*



## Disclaimers



- Personal, biased view of data-streaming world
  - Revolve around own line of work, interests, and results
  - Focus on a couple of basic algorithmic tools
    - A lot more out there . . .
    - 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], overview article by S. Muthukrishnan
  - Ask questions!
  - Talk to me afterwards

## Data-Stream Management

Lucent Technologies  
Bell Labs Innovations

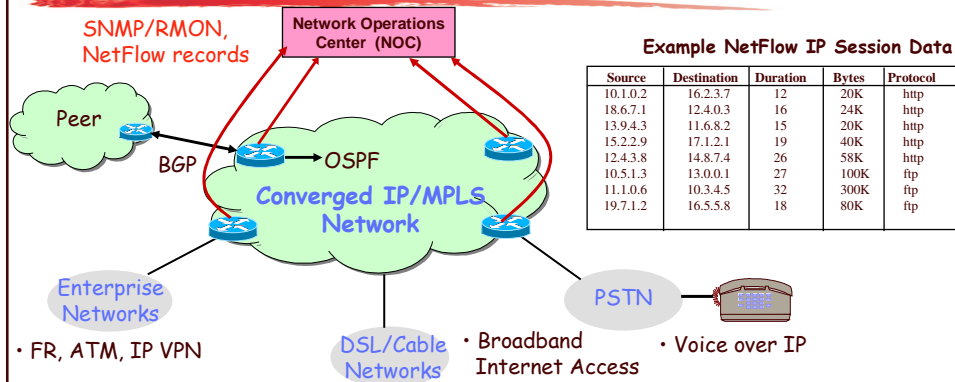


- **Traditional DBMS** - data stored in **finite, persistent data sets**
- **Data Streams** - distributed, continuous, unbounded, rapid, time varying, noisy, . . .
- **Data-Stream Management** - variety of modern applications
  - Network monitoring and traffic engineering
  - Telecom call-detail records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

3

## Networks Generate Massive Data Streams

Lucent Technologies  
Bell Labs Innovations



- 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

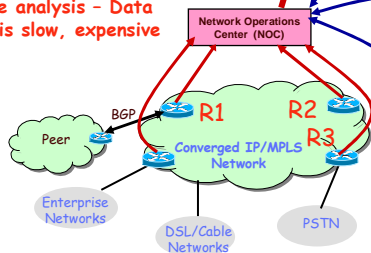
Lucent Technologies  
Bell Labs Innovations



### Back-end Data Warehouse

DBMS  
(Oracle, DB2)

Off-line analysis - Data access is slow, expensive



What are the top (most frequent) 1000 (source, dest) pairs seen by R1 over the last month?

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

*Set-Expression Query*

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

*SQL Join Query*

- 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

Lucent Technologies  
Bell Labs Innovations

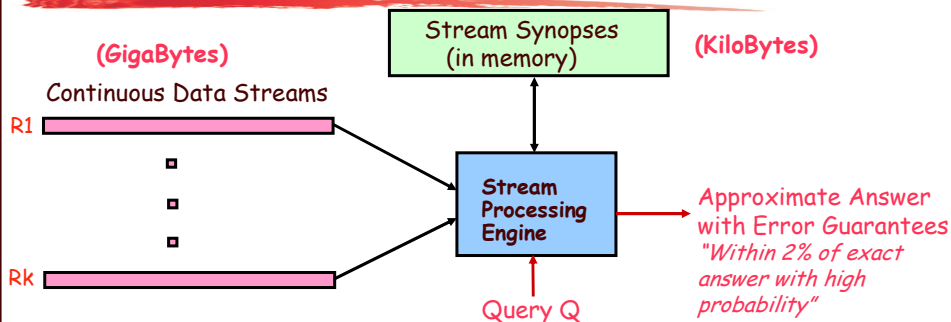


- Introduction & Motivation
- **Data Stream Computation Model**
- Two Basic Sketching Tools for Streams
  - Linear-Projection (aka AMS) Sketches
    - *Applications:* Join/Multi-Join Queries, Wavelets
  - Hash (aka FM) Sketches
    - *Applications:* Distinct Values, Set Expressions
- Extensions
  - Correlating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

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  $f(i)$  ( $i=1, \dots, N$ ) seen as a stream of  $i$ -values



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

$$\langle f, \xi \rangle = \sum f(i) \xi_i \quad \text{where } \xi = \text{vector of random values from an appropriate distribution}$$

- Simple to compute over the stream: Add  $\xi_i$  whenever the  $i$ -th value is seen



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

9

## Example: Binary-Join COUNT Query

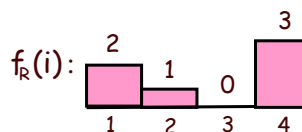
Lucent Technologies  
Bell Labs Innovations



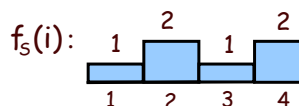
- **Problem:** Compute answer for the query  $\text{COUNT}(R \bowtie_A S)$

- **Example:**

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



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



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

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

10

## Basic AMS Sketching Technique [AMS96]

Lucent Technologies  
Bell Labs Innovations



- **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  $10 \pm 1$  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  $\xi_i$ ,  $E[\xi_i] = 0$

- Variables  $\xi_i$  are 4-wise independent

- Expected value of product of 4 distinct  $\xi_i = 0$

- Variables  $\xi_i$  can be generated using pseudo-random generator using only  $O(\log N)$  space (for seeding)!

11

## AMS Sketch Construction

Lucent Technologies  
Bell Labs Innovations



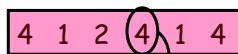
- Compute random variables:  $X_R = \sum_i f_R(i) \xi_i$  and  $X_S = \sum_i f_S(i) \xi_i$

- Simply add  $\xi_i$  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

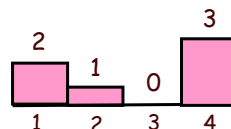
- **Example:**

Data stream R.A:



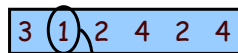
$$X_R = X_R + \xi_4$$

$f_R(i)$ :



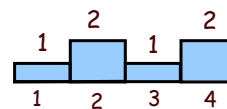
$$X_R = 2\xi_1 + \xi_2 + 3\xi_4$$

Data stream S.A:



$$X_S = X_S + \xi_1$$

$f_S(i)$ :



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

12

## Binary-Join AMS Sketching Analysis

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

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

1
0

- Using 4-wise independence, possible to show that

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

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

13

## Boosting Accuracy

- Chebyshev's Inequality:

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

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

$$s = \frac{8 \cdot (2 \cdot \text{SJ}(R) \cdot \text{SJ}(S))}{\epsilon^2 \text{COUNT}^2} \quad \text{copies} \quad \xrightarrow{\text{Average}} \quad Y \quad E[Y] = E[X] = \text{COUNT}(R \bowtie_A S)$$

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

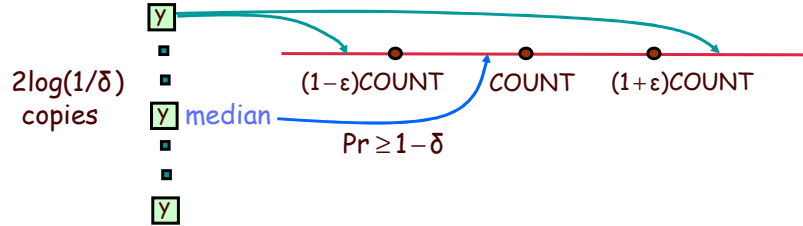
$$\Pr(|Y - \text{COUNT}| \geq \epsilon \cdot \text{COUNT}) \leq \frac{\text{Var}[Y]}{\epsilon^2 \text{COUNT}^2} \leq \frac{1}{8}$$

14

## Boosting Confidence

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

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



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

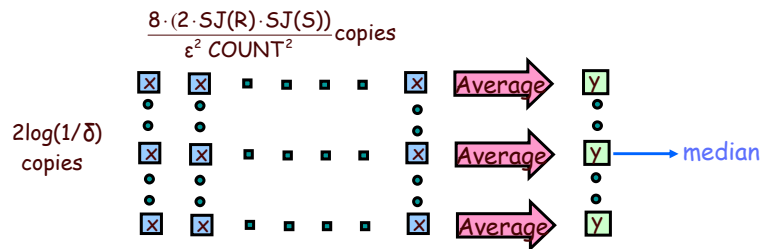
$$= \Pr[\# \text{ failures in } 2\log(1/\delta) \text{ trials} \geq \log(1/\delta)]$$

$$\leq \delta \quad (\text{By Chernoff Bound})$$

15

## Summary of Binary-Join AMS Sketching

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



- Main Theorem (AGMS99): Sketching approximates  $\text{COUNT}$  to within a relative error of  $\varepsilon$  with probability  $\geq 1-\delta$  using space

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

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

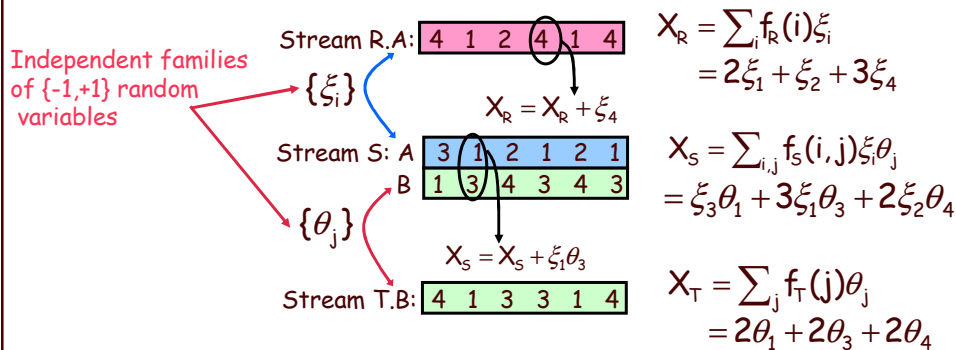
16



## AMS Sketching for Multi-Join Aggregates [DGGRO2]



- **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  $X_T$



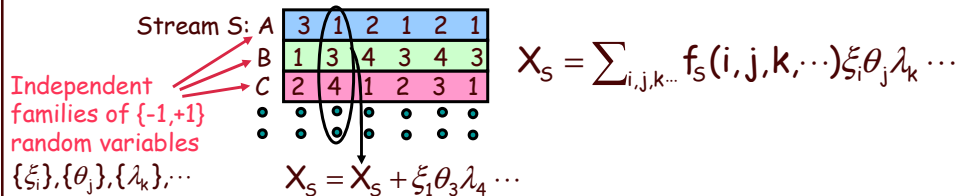
- Return  $X = X_R X_S X_T$  ( $E[X] = COUNT(R \bowtie_A S \bowtie_B T)$ )
- $E[f_R(i) \cdot f_S(i',j) \cdot f_T(j') \cdot \xi_i \xi_{i'} \theta_j \theta_{j'}] = 0$  if  $i \neq i'$  or  $j \neq j'$

17

## AMS Sketching for Multi-Join Aggregates



- 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  $X_R, X_S, X_T, \dots$



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

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

- Explosive increase with the number of joins!

18

## Boosting Accuracy by Sketch Partitioning: Basic Idea



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

$$s = \frac{8 \cdot (2^{2m} \text{SJ}(R) \cdot \text{SJ}(S) \dots)}{\epsilon^2 \text{COUNT}^2} \text{ copies}$$

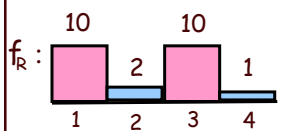
$$\text{Var}[Y] = \frac{\text{Var}[X]}{s} \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$$

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

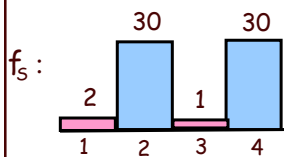
## Sketch Partitioning Example: Binary-Join COUNT Query



### Without Partitioning



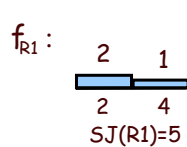
$$\text{SJ}(R) = 205$$



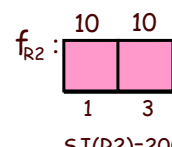
$$\text{SJ}(S) = 1805$$

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

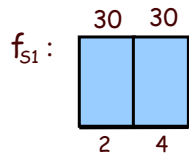
### With Partitioning ( $P1=\{2,4\}, P2=\{1,3\}$ )



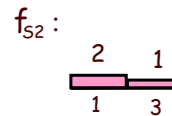
$$\text{SJ}(R1) = 5$$



$$\text{SJ}(R2) = 200$$



$$\text{SJ}(S1) = 1800$$



$$\text{SJ}(S2) = 5$$

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

$$\text{VAR}[X2] \approx 2 \cdot \text{SJ}(R2) \cdot \text{SJ}(S2) \approx 2\text{K}$$

$$X = X1 + X2, E[X] = \text{COUNT}(R \bowtie S)$$

$$\text{VAR}[X] = \text{VAR}[X1] + \text{VAR}[X2] \approx 20\text{K}$$

## Overview of Sketch Partitioning

- Maintain independent sketches for partitions of join-attribute space
- Improved error guarantees
  - $\text{Var}[X] = \sum \text{Var}[X_i]$  is smaller (by *intelligent domain partitioning*)
  - "Variance-aware" boosting: More space to higher-variance partitions
- **Problem:** Given total sketching space  $S$ , find domain partitions  $p_1, \dots, p_k$  and space allotments  $s_1, \dots, s_k$  such that  $\sum_j s_j \leq S$ , and the variance 
$$\frac{\text{Var}[X_1]}{s_1} + \frac{\text{Var}[X_2]}{s_2} + \dots + \frac{\text{Var}[X_k]}{s_k}$$
 is minimized
  - Solved optimal for binary-join case (using Dynamic-Programming)
  - NP-hard for  $\geq 2$  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

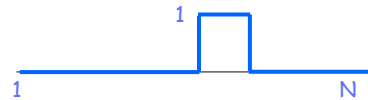
## Other Applications of AMS Stream Sketching

- Key Observation:  $|R_1 \bowtie R_2| = \sum f_1(i) f_2(i) = \langle f_1, f_2 \rangle = \text{inner product!}$
- **General result:** Streaming  $(\epsilon, \delta)$  estimation of "large" inner products using AMS sketching

- Other streaming inner products of interest

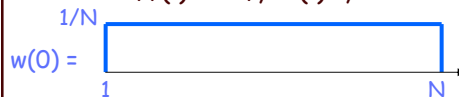
- Top-k frequencies [CCF02]

- Item frequency =  $\langle f, \text{"unit\_pulse"} \rangle$



- Large wavelet coefficients [GKMS01]

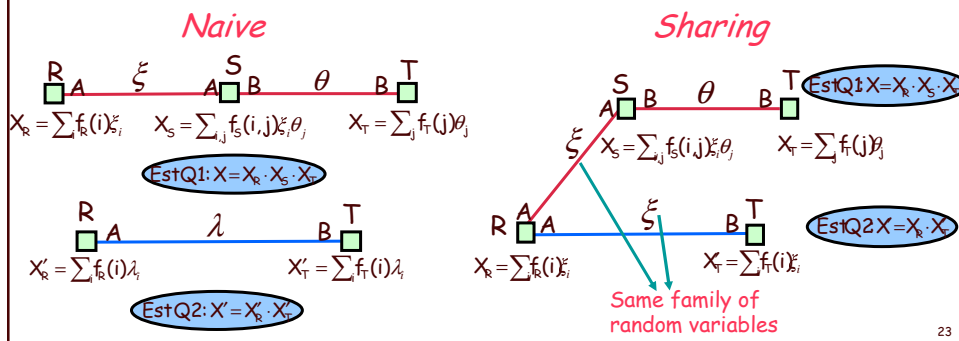
- $\text{Coeff}(i) = \langle f, w(i) \rangle$ , where  $w(i)$  =  $i$ -th wavelet basis vector



22

## 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]



23

## 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

24

## Distinct Value Estimation



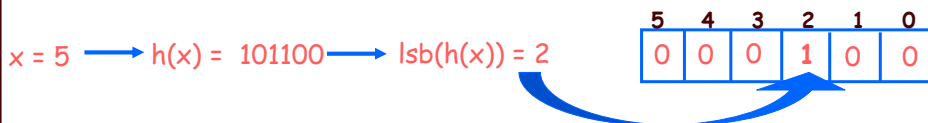
- **Problem:** Find the *number of distinct values* in a stream of values with domain  $[0, \dots, N-1]$ 
  - Zeroth frequency moment  $F_0$ , LO (Hamming) stream norm
  - Statistics: number of *species or classes* in a population
  - Important for query optimizers
  - *Network monitoring:* distinct 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!

25

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



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



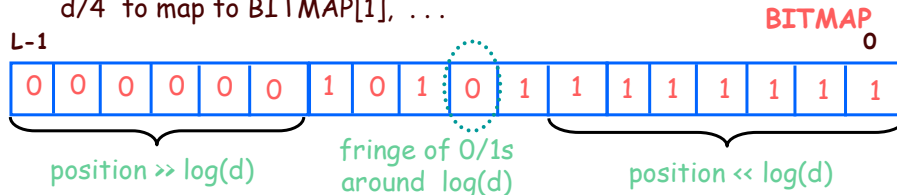
26

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

Lucent Technologies  
Bell Labs Innovations



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



- 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

27

## Hash Sketches for Distinct Value Estimation

Lucent Technologies  
Bell Labs Innovations



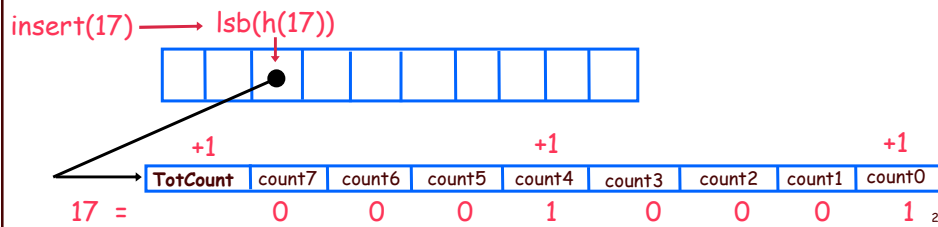
- [FM85] assume "ideal" hash functions  $h(x)$  (N-wise independence)
  - [AMS96]: pairwise independence is sufficient
    - $h(x) = (a \cdot x + b) \bmod N$ , where  $a, b$  are random binary vectors in  $[0, \dots, 2^L - 1]$
  - Small-space  $(\epsilon, \delta)$  estimates for distinct values proposed 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  $|S_1 \cup S_2 \cup \dots \cup S_k| = \text{set-union cardinality}$

28

## Processing Set Expressions over Update Streams [GGR03]



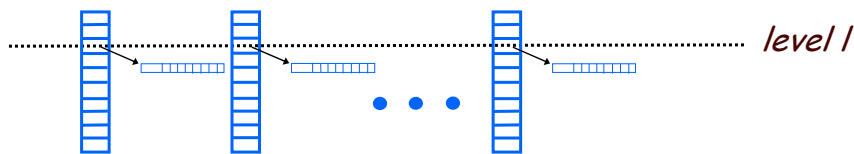
- Estimate cardinality of *general set expressions* over streams of updates
  - E.g., number of distinct (source,dest) pairs seen at both R1 and R2 but not R3?  $| (R1 \cap R2) - R3 |$
- *2-Level Hash-Sketch (2LHS) stream synopsis*: Generalizes FM sketch
  - *First level*:  $\Theta(\log N)$  buckets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



## 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



- 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  $\frac{1}{2^{l+1}}$
- 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, $|A-B|$

Lucent Technologies  
Bell Labs Innovations



- 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  $|A-B| = \frac{\# \text{witnesses for } A-B}{\# \text{singleton buckets}} \times \hat{u}$

31

## Estimation Guarantees

Lucent Technologies  
Bell Labs Innovations



- Our set-difference cardinality estimate is within a relative error of  $\epsilon$  with probability  $\geq 1 - \delta$  when the number of 2LHS is  $O\left(\frac{|A \cup B| \log(1/\delta)}{|A-B| \epsilon^2}\right)$
- Lower bound of  $\Omega\left(\frac{|A \cup B|}{|A-B| \epsilon}\right)$  space, using communication-complexity arguments
- Natural generalization to arbitrary set expressions  $E = f(S_1, \dots, S_n)$ 
  - Build parallel, independent 2LHS for each  $S_1, \dots, S_n$
  - Generalize "witness" condition (inductively) based on  $E$ 's structure
  - $(\epsilon, \delta)$  estimate for  $|E|$  using  $O\left(\frac{|S_1 \cup \dots \cup S_n| \log(1/\delta)}{|E| \epsilon^2}\right)$  2LHS synopses
- *Worst-case bounds!* Performance in practice is much better [GGR03]

32

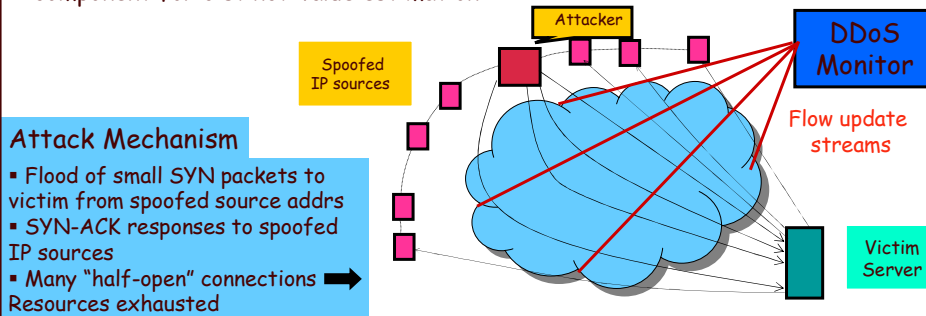


## Application: Detecting TCP-SYN-Flooding DDoS Attacks

Lucent Technologies  
Bell Labs Innovations



- 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



## Talk Outline

Lucent Technologies  
Bell Labs Innovations

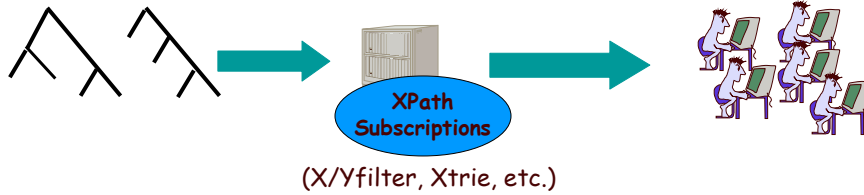


- Introduction & Motivation
- Data Stream Computation Model
- Two Basic Sketching Tools for Streams
  - Linear-Projection (aka AMS) Sketches
    - Applications: Join/Multi-Join Queries, Wavelets
  - Hash (aka FM) Sketches
    - Applications: Distinct Values, Set Expressions
- Extensions
  - Correlating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

34

# Processing XML Data Streams

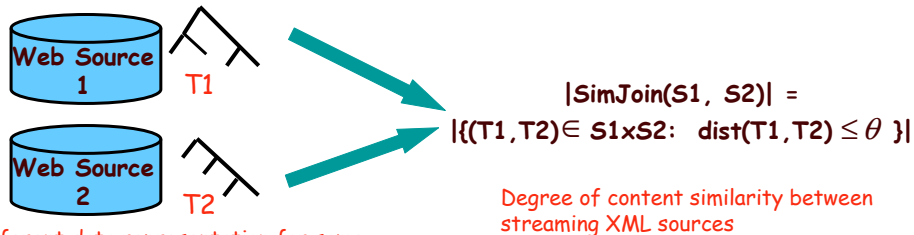
- XML: Much richer, (semi)structured data model
  - Ordered, node-labeled data trees
- Bulk of work on XML streaming: *Content-based filtering of XML documents* (publish/subscribe systems)
  - Quickly match incoming documents against standing XPath subscriptions



- 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

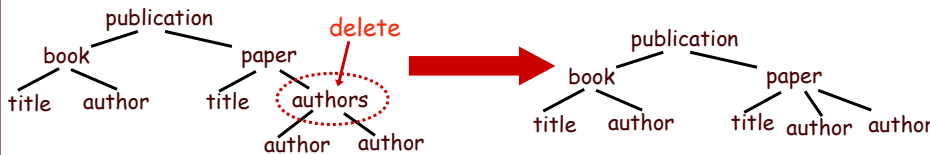
# Processing XML Data Streams

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



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

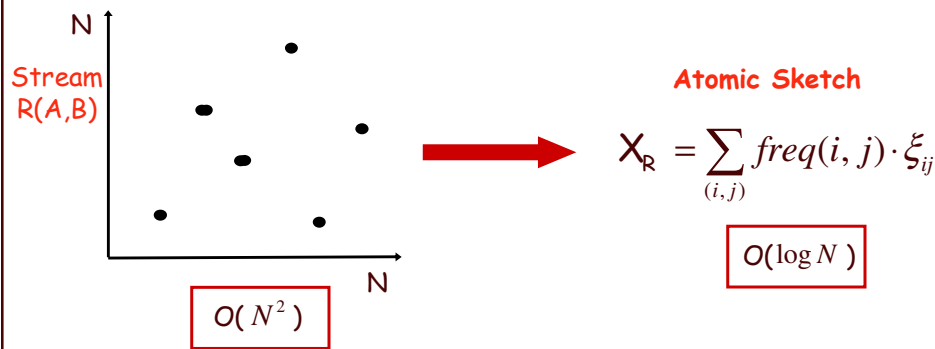
- Correlation metric: *Tree-edit distance*  $ed(T1, T2)$ 
  - Node relabels, inserts, deletes - also, allow for *subtree moves*



## 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)

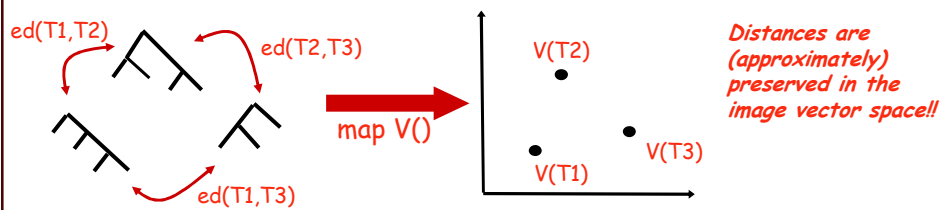


37

## 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



- 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

38

## 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
    - Factors of 5-10 for 15K-node trees, consistently overestimate

39

## 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

40

## 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  $i = 0, 1, \dots$ ) 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)

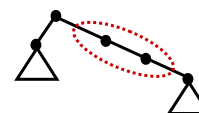
$V(T)[x] = \text{no. of times subtree } x \text{ appears in the parsing structure for } T$
- Our parsing guarantees
  - $O(\log|T|)$  parsing levels (constant-fraction reduction at each level)
  - $V(T)$  is *very sparse*: Only  $O(|T|)$  non-zero components in  $V(T)$ 
    - Even though dimensionality =  $O((4|\sigma|)^n)$  ( $\sigma$  = label alphabet)
    - Allows for effective sketching
  - $V(T)$  is constructed in time  $O(|T| \log^* |T|)$

41

## Embedding Algorithm (cont.)



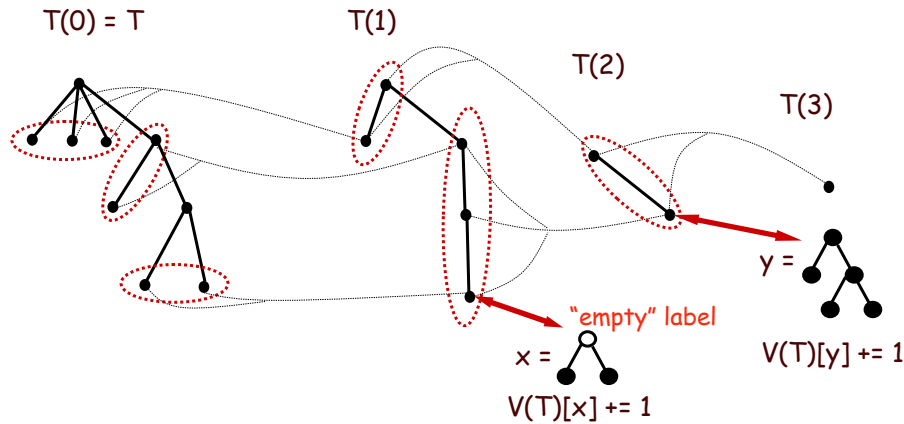
- Node grouping at a given parsing level  $T(i)$ : Create groups of 2 or 3 nodes of  $T(i)$  and merge them into a single node of  $T(i+1)$ 
  1. Group maximal sequence of contiguous leaf children of a node
  2. Group maximal sequence of contiguous nodes in a chain
  3. Fold leftmost lone leaf child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



42

## Embedding Algorithm (cont.)

- Example hierarchical tree parsing



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

43

## 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:

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

- 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  $O(|\text{influence region}|)$

44

## 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 ( $T(O)$ ), 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 . . .

45

## Sketching a Massive, Streaming XML Tree

- *Input*: Massive XML data tree  $T$  ( $n = |T| \gg$  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  $T$ ,  $\delta$  = probabilistic confidence in  $ed()$  approximation
  - XML trees are typically "bushy" ( $d \ll n$  or  $d = O(\text{polylog}(n))$ )

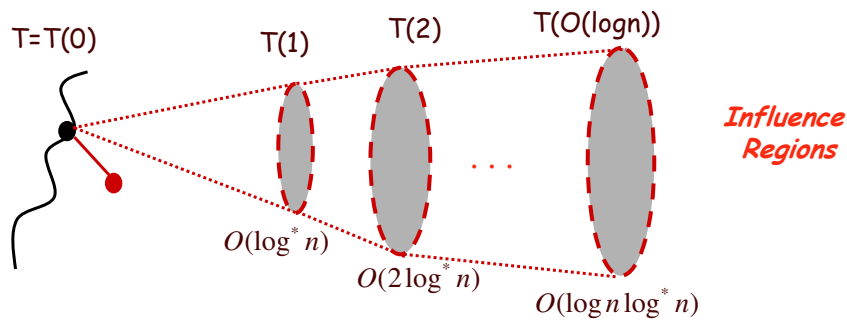
46

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



- **Key Ideas**

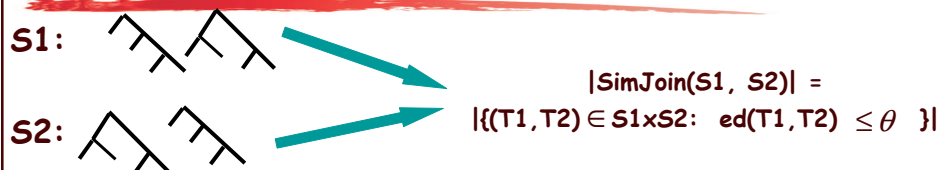
- Incrementally parse  $T$  to produce  $V(T)$  as elements stream in
- Just need to retain the *influence region* nodes for each parsing level and for each node in the current root-to-leaf path



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

47

## Approximate Similarity Joins over XML Streams



- **Input:** Long streams  $S_1, S_2$  of  $N$  (short) XML documents ( $\leq b$  nodes)
- **Output:** Estimate for  $|\text{SimJoin}(S_1, S_2)|$

- **Theorem:** Can build an atomic sketch-based estimate for  $|\text{SimJoin}(S_1, S_2)|$  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
  - $\delta$  = probabilistic confidence in distance estimates

48



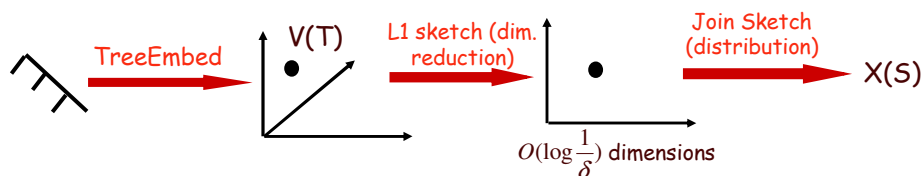
## Approximate Similarity Joins over XML Streams (cont.)

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 . . .

49

## Conclusions

Lucent Technologies  
Bell Labs Innovations

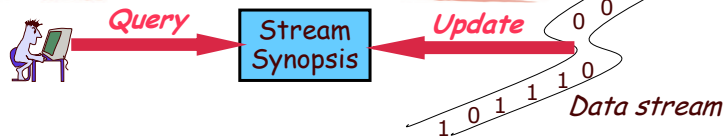


- Analyzing massive data streams: Real problem with several real-world applications
  - Fundamentally rethink data management under stringent constraints
  - Single-pass algorithms with limited memory/CPU-time resources
- *Pseudo-random sketching* is a viable technique for a variety of streaming tasks
  - Limited-space randomized approximations
  - *Probabilistic guarantees* on the quality of the approximate answer
  - *Delete-proof* (supports insertion *and* deletion of records)
  - *Composable* (ideal for distributed computation)

50

## Future Work: Tracking Continuous Streams in Small Time

Lucent Technologies  
Bell Labs Innovations

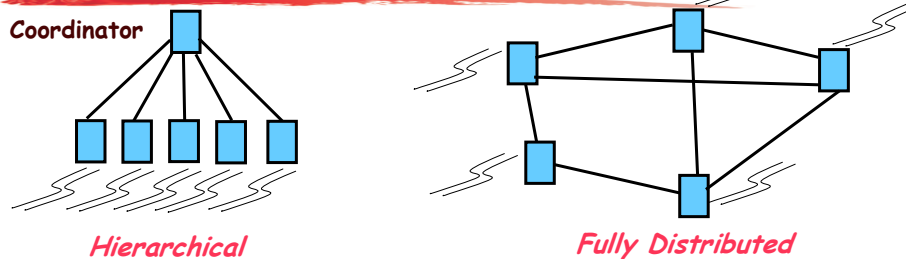


- Update/Query times are typically  $\Omega(|\text{synopsis}|)$  -- fine as long as synopsis sizes are small (polylog), BUT...
  - Small synopses are often *impossible* (strong communication-complexity lower bounds)
    - E.g., set expressions, joins, ...
- Synopsis size may not be the crucial limiting 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

51

## Future Work: Distributed Approximate Stream Tracking

Lucent Technologies  
Bell Labs Innovations



- **Goal:** 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
  - Additional concern: *Communication Efficiency*
    - Minimize message exchanges involved for a given accuracy guarantee
- Some initial results on distributed "top-k frequency" monitoring [BO03]
  - *Deterministic guarantees*, using *full space* -- no sketching/synopses employed
  - More complex distributed tracking problems (e.g., joins) are wide open!

52

## Other Future Research Directions

Lucent Technologies  
Bell Labs Innovations

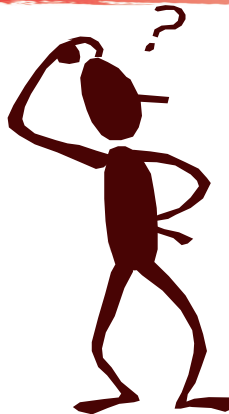


- Sketches/synopses for richer types of streaming data and queries
  - Spatial data streams, queries over sliding windows, mining/querying streaming graphs, . . .
- Other metric-space embeddings in the streaming model
- Stream-data processing architectures and query languages
  - Progress: Aurora, STREAM,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

53

## Thank you!

Lucent Technologies  
Bell Labs Innovations



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

[minos@research.bell-labs.com](mailto:minos@research.bell-labs.com)

54

## 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  $S.A = i$

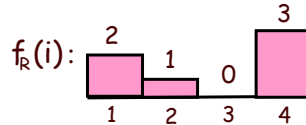
- Sketch-based solution

- Compute random variables  $X_R$  and  $X_S$

Stream R.A: 

4	1	2	4	1	4
---	---	---	---	---	---

$$X_R = X_R + \xi_4$$

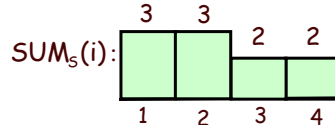


$$X_R = \sum_i f_R(i) \xi_i = 2\xi_1 + \xi_2 + 3\xi_4$$

Stream S: 

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

$$X_S = X_S + 3\xi_1$$



$$X_S = \sum_i SUM_S(i) \xi_i = 3\xi_1 + 3\xi_2 + 2\xi_3 + 2\xi_4$$

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

55

## 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  $\epsilon$  with probability  $\geq 1 - \delta$  using space  $O(\log N \cdot \log(1/\delta) / (\epsilon^2 \lambda^2))$ , where

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

- Observation:  $|R1 \bowtie R2| = \sum f_1(i) f_2(i) = \langle f_1, f_2 \rangle = \text{inner product!}$

- General result for inner-product approximation using sketches

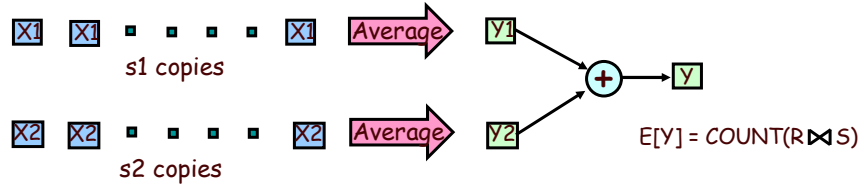
- Other inner products of interest: *Haar wavelet coefficients!*

- Haar wavelet decomposition = inner products of signal/distribution with specialized (wavelet basis) vectors

56

## Space Allocation Among Partitions

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



$$\text{Var}[Y] = \frac{\text{Var}[X1]}{s1} + \frac{\text{Var}[X2]}{s2} \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$$

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

57

## Sketch Partitioning Problems

- **Problem 1:** Given sketches  $X1, \dots, Xk$  for partitions  $P1, \dots, Pk$  of the join attribute domain, what is the space  $s_j$  that must be allocated to  $P_j$  (for  $s_j$  copies of  $X_j$ ) so that

$$\text{Var}[Y] = \frac{\text{Var}[X1]}{s1} + \frac{\text{Var}[X2]}{s2} + \dots + \frac{\text{Var}[Xk]}{sk} \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$$

and  $\sum_j s_j$  is minimum

- **Problem 2:** Compute a partitioning  $P1, \dots, Pk$  of the join attribute domain, and space  $s_j$  allocated to each  $P_j$  (for  $s_j$  copies of  $X_j$ ) such that

$$\text{Var}[Y] = \frac{\text{Var}[X1]}{s1} + \frac{\text{Var}[X2]}{s2} + \dots + \frac{\text{Var}[Xk]}{sk} \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$$

and  $\sum_j s_j$  is minimum

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

58

## Optimal Space Allocation Among Partitions



- **Key Result (Problem 1):** Let  $X_1, \dots, X_k$  be sketches for partitions  $P_1, \dots, P_k$  of the join attribute domain. Then, allocating space

$$s_j = \frac{8\sqrt{\text{Var}(X_j)}(\sum_j \sqrt{\text{Var}(X_j)})}{\epsilon^2 \text{COUNT}^2}$$

to each  $P_j$  (for  $s_j$  copies of  $X_j$ ) ensures that  $\text{Var}[Y] \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$   
and  $\sum_j s_j$  is minimum

- Total sketch space required:  $\sum_j s_j = \frac{8(\sum_j \sqrt{\text{Var}(X_j)})^2}{\epsilon^2 \text{COUNT}^2}$

- **Problem 2 (Restated):** Compute a partitioning  $P_1, \dots, P_k$  of the join attribute domain such that  $\sum_j \sqrt{\text{Var}(X_j)}$  is minimum
  - Optimal partitioning  $P_1, \dots, P_k$  minimizes total sketch space

## Binary-Join Queries: Binary Space Partitioning



- **Problem:** For  $\text{COUNT}(R \bowtie_A S)$ , compute a partitioning  $P_1, P_2$  of  $A$ 's domain  $\{1, 2, \dots, N\}$  such that  $\sqrt{\text{Var}(X_1)} + \sqrt{\text{Var}(X_2)}$  is minimum

- Note:  $\text{Var}(X_j) \approx 2 \sum_{i \in P_j} f_R(i)^2 \sum_{i \in P_j} f_S(i)^2$

- **Key Result (due to Breiman):** For an optimal partitioning  $P_1, P_2$ ,

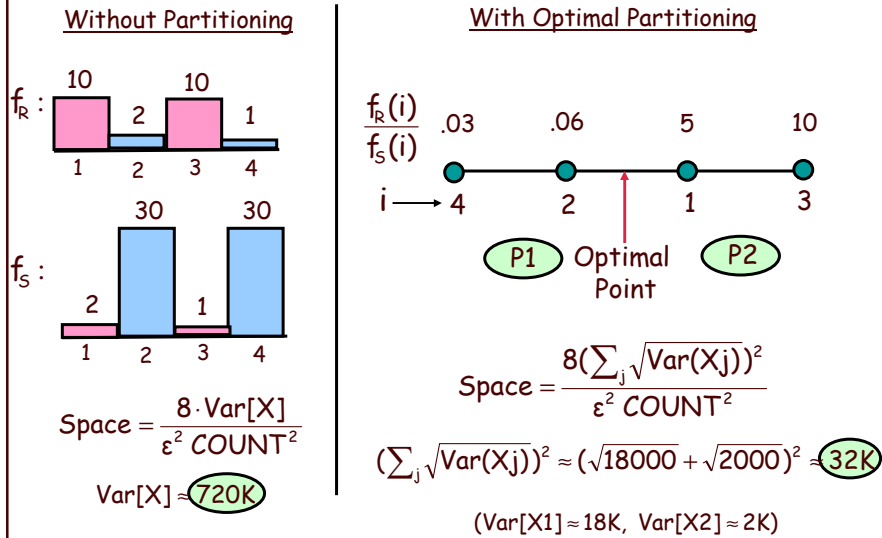
$$\forall i_1 \in P_1, \forall i_2 \in P_2, \frac{f_R(i_1)}{f_S(i_1)} < \frac{f_R(i_2)}{f_S(i_2)}$$

- **Algorithm**

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

$$\sqrt{2 \sum_{i \in P_1} f_R(i)^2 \sum_{i \in P_1} f_S(i)^2} + \sqrt{2 \sum_{i \in P_2} f_R(i)^2 \sum_{i \in P_2} f_S(i)^2}$$

## Binary Sketch Partitioning Example

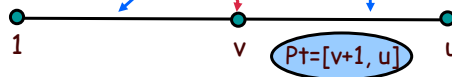


61

## Binary-Join Queries: K-ary Sketch Partitioning

- **Problem:** For  $\text{COUNT}(R \bowtie_A S)$ , compute a partitioning  $P_1, P_2, \dots, P_k$  of  $A$ 's domain such that  $\sum_j \sqrt{\text{Var}(X_j)}$  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  $\phi(u, t)$  be the value of  $\sum_j \sqrt{2 \sum_{i \in P_j} f_r(i)^2 \sum_{i \in P_j} f_s(i)^2}$  when  $[1, u]$  is split optimally into  $t$  partitions  $P_1, P_2, \dots, P_t$

$$\phi(u, t) = \min_{1 \leq v \leq u} \{ \phi(v, t-1) + \sqrt{2 \sum_{i=v+1}^u f_r(i)^2 \sum_{i=v+1}^u f_s(i)^2} \}$$



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

62

## Sketch Partitioning for Multi-Join Queries

Lucent Technologies  
Bell Labs Innovations



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

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

- Partitioning problem is NP-hard for more than 1 join attribute
- If join attributes are independent, then possible to compute optimal partitioning
  - Choose  $k_1$  such that allocating  $k_1$  partitions to attribute  $A$  and  $k/k_1$  to remaining attributes minimizes  $\Psi$
  - Compute optimal  $k_1$  partitions for  $A$  using previous dynamic programming algorithm

63

## Experimental Study

Lucent Technologies  
Bell Labs Innovations

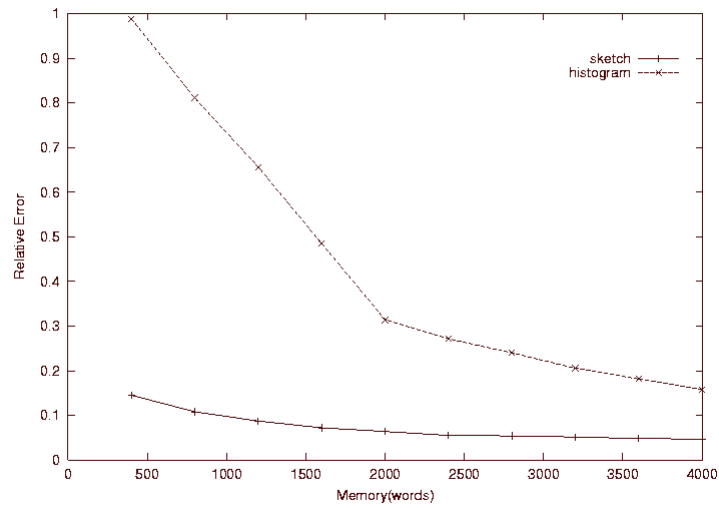


- Summary of findings
  - Sketches are superior to **1-d (equi-depth) histograms** for answering  $COUNT$  queries over data streams
  - Sketch partitioning is effective for reducing error
- Real-life Census Population Survey data sets (1999 and 2001)
  - Attributes considered:
    - Income (1:14)
 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

64

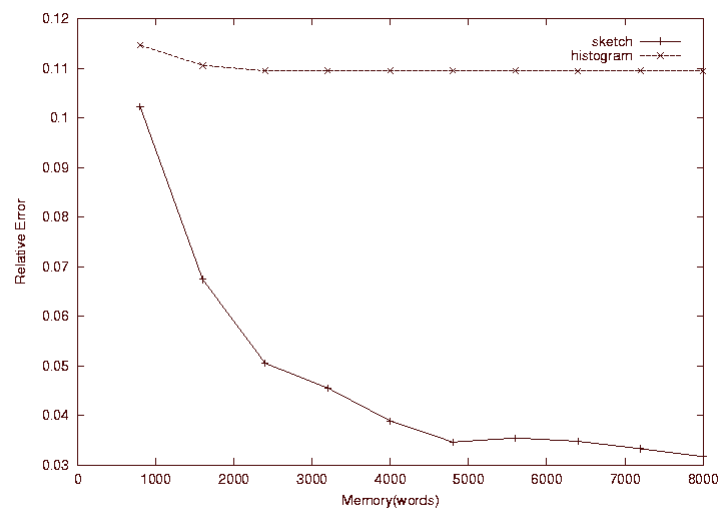


## Join (Weekly Wage)



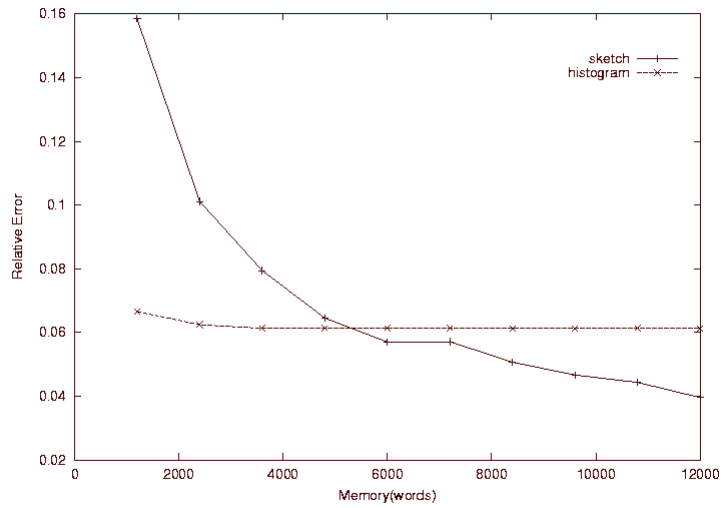
65

## Join (Age, Education)



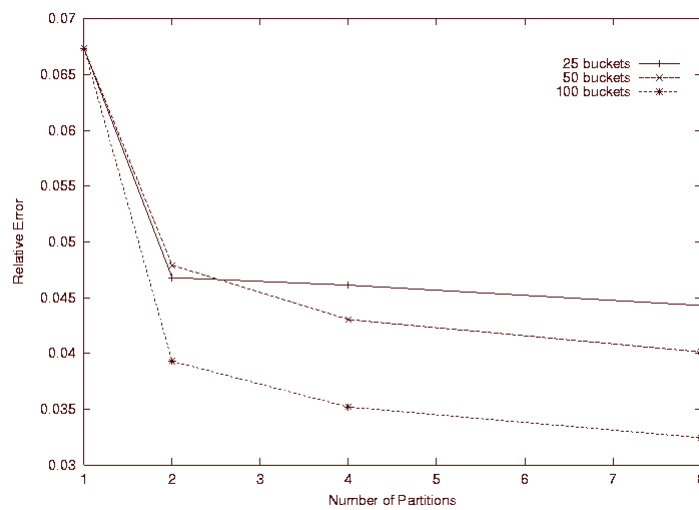
66

## Star Join (Age, Education, Income)



67

## Join (Weekly Wage Overtime = Weekly Wage)



68

## 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]

## Sketching for Multiple Standing Queries



- Consider queries  $Q_1 = \text{COUNT}(R \bowtie_A S \bowtie_B T)$  and  $Q_2 = \text{COUNT}(R \bowtie_{A=B} T)$
- Naive approach: construct separate sketches for each join
  - $\xi, \theta, \lambda$  are independent families of pseudo-random variables

$$\begin{array}{c}
 \boxed{R} \text{---} \overset{\xi}{A} \text{---} \boxed{S} \text{---} \overset{\theta}{B} \text{---} \boxed{T} \\
 X_R = \sum_i f_R(i) \xi_i \quad X_S = \sum_{i,j} f_S(i,j) \xi_i \theta_j \quad X_T = \sum_j f_T(j) \theta_j
 \end{array}$$

$$\text{Est } Q_1: X = X_R \cdot X_S \cdot X_T$$

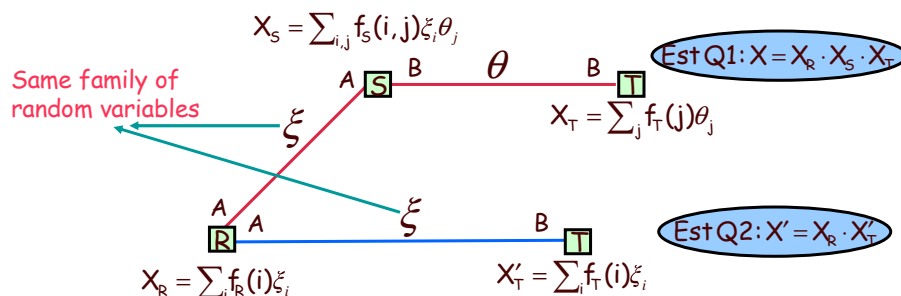
$$\begin{array}{c}
 \boxed{R} \text{---} \overset{\lambda}{A} \text{---} \boxed{T} \\
 X'_R = \sum_i f_R(i) \lambda_i \quad X'_T = \sum_i f_T(i) \lambda_i
 \end{array}$$

$$\text{Est } Q_2: X' = X'_R \cdot X'_T$$

## 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

71

## 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

72

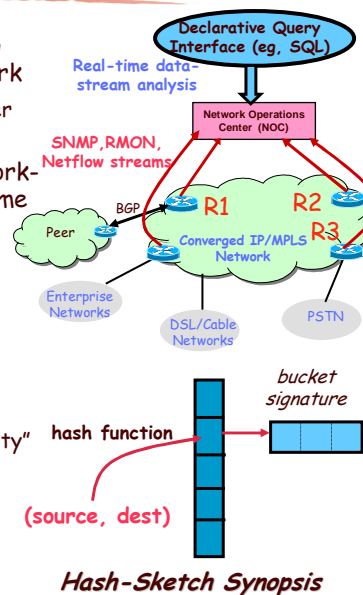
## Set Expressions to Sketch Expressions

- Given set expression  $E = f(S_1, \dots, S_n)$ , level of inference  $l$ 
  - Again, look for buckets that are singletons for the union of  $S_1, \dots, S_n$  at level  $l$
- "Witness" Condition for  $E$ : Create boolean expression  $B(E)$  over parallel sketches inductively
  - Replace  $S_i$  by  $\text{isSingleton}(\text{sketch}(S_i), l)$
  - Replace  $E_1 \cap E_2$  by  $B(E_1) \text{ AND } B(E_2)$
  - Replace  $E_1 - E_2$  by  $B(E_1) \text{ AND } (\text{NOT } B(E_2))$
  - Replace  $E_1 \cup E_2$  by  $B(E_1) \text{ OR } B(E_2)$
- Then,  $\text{Prob}[\text{witness} \mid \text{singleton}] = |E| / |S_1 \cup \dots \cup S_n|$

73

## 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



74