# Estimating Join-Distinct Aggregates over Update Streams

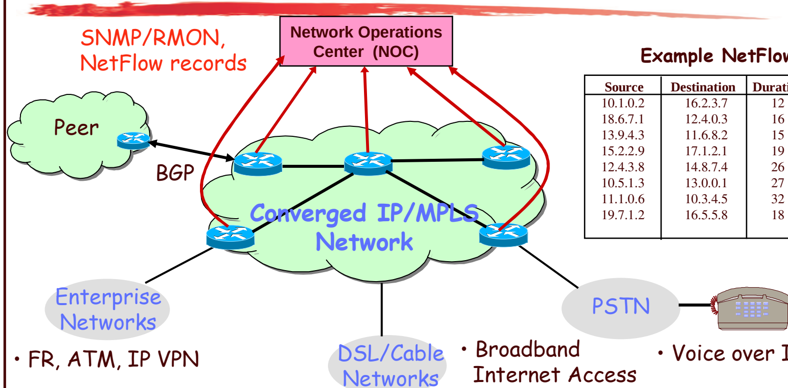

***Minos Garofalakis***

*Bell Labs, Lucent Technologies*

*(Joint work with Sumit Ganguly, Amit Kumar, Rajeev Rastogi)*



## Motivation: Massive Network-Data Streams




**Example NetFlow IP Session Data**

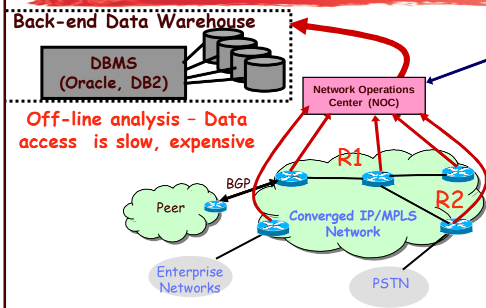
| Source | Destination | Duration | Bytes | Protocol |
|---|---|---|---|---|
| 10.1.0.2 | 16.2.3.7 | 12 | 20K | http |
| 18.6.7.1 | 12.4.0.3 | 16 | 24K | http |
| 13.9.4.3 | 11.6.8.2 | 15 | 20K | http |
| 15.2.2.9 | 17.1.2.1 | 19 | 40K | http |
| 12.4.3.8 | 14.8.7.4 | 26 | 58K | http |
| 10.5.1.3 | 13.0.0.1 | 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 for off-line analysis

2

# Real-Time Data-Stream Querying






How many distinct (source,dest) pairs are seen by R1 where "source" is also seen by R2?

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

- 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
  - Different classes of analysis queries: *top-k, quantiles, joins, …*
- ***Our focus: Join-Distinct (JD) aggregate queries***
  - Estimating cardinality of duplicate-eliminating projection over a join
- Critical to important NM tasks
  - Denial-of-Service attacks, SLA violations, real-time traffic engineering,…

3

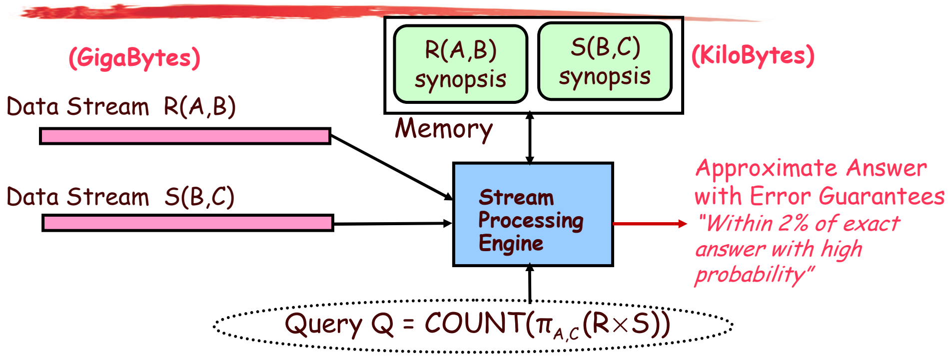
# Talk Outline

Lucent Technologies
Bell Labs Innovations

- Introduction & Motivation
- Data Stream Computation Model
- Key Prior Work
  - FM sketches for distinct counting
  - 2-level hash sketches for set-expression cardinalities
- Our Solution: ***JD-Sketch Synopses***
  - The basic structure
  - JD-sketch composition algorithm & JD estimator
  - Extensions
- Experimental Results
- Conclusions

4

## Data-Stream Processing Model




- Approximate answers often suffice, e.g., trend analysis, anomaly detection
  - Exact solution requires linear space (SET-DISJOINTNESS)
- 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



## Existing Synopses for Relational Streams?

- Conventional data summaries fall short
  - Samples (e.g., using Reservoir Sampling)
    - Bad for joins and DISTINCT counting, cannot handle deletions
  - Multi-d histograms/wavelets
    - Construction requires multiple passes, not useful for DISTINCT clauses
- Combine existing stream-sketching solutions?
  - Hash (aka FM) sketches for distinct-value counting
  - AMS sketches for join-size estimation
  - *Fundamentally different* : Hashing vs. Random linear projections
    - Effective combination seems difficult
- Our Solution: *JD-Sketch stream synopses*
  - Novel, hash-based, log-sized stream summaries
  - Built *independently* over R, S streams, then *composed* to give JD estimate
  - Strong probabilistic accuracy guarantees

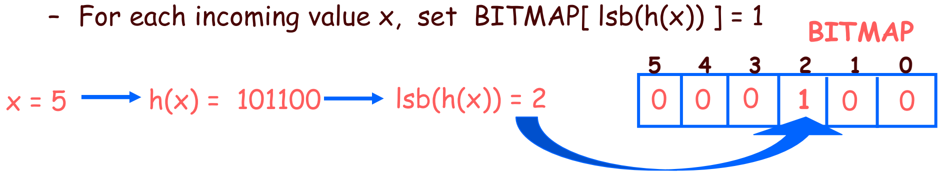
## Hash (aka FM) Sketches for Distinct Value Counting [FM85]

- **Problem:** Estimate the number of distinct items in a stream of values from [0,…, M-1]

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

*Number of distinct values: 5*

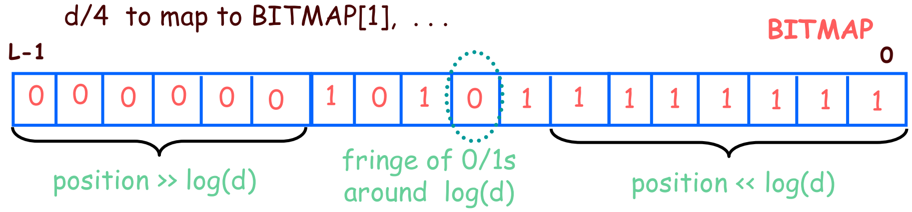
- Assume a hash function h(x) that maps incoming values x in [0,…, M-1] *uniformly* across [0,…, 2^L-1], where L = O(logM)
- Let lsb(y) denote the position of the least-significant 1 bit in the binary representation of y
  - A value x is mapped to lsb(h(x))
- Maintain *FM Sketch* = BITMAP array of L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




7

## Hash (aka FM) Sketches for Distinct Value Counting [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], . . .



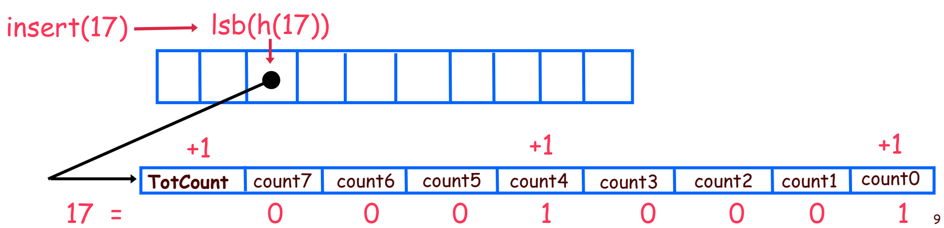

- Let R = position of rightmost zero in BITMAP
  - Use as indicator of log(d)
  - Estimate d = $2^R/\phi$
  - Average several iid instances (different hash functions) to reduce estimator variance

8

## 2-Level Hash Sketches for Set Expression Cardinalities [GGR03]

Lucent Technologies
Bell Labs Innovations

- Estimate cardinality of *general set expressions* over streams of updates
  - E.g., number of distinct (source,dest) pairs seen at both R1 and R2? $|R1 \cap R2|$
- *2-Level Hash-Sketch (2LHS) stream synopsis:* Generalizes FM sketch
  - *First level:* $\Theta(\log M)$ buckets with exponentially-decreasing probabilities (using lsb(h(x)), as in FM)
  - *Second level:* Count-signature array (logM+1 counters)
    - One "total count" for elements in first-level bucket
    - logM "bit-location counts" for 1-bits of incoming elements

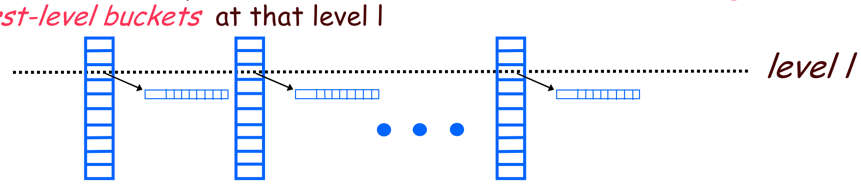



9

## Processing Set Expressions over Update Streams: Key Ideas

Lucent Technologies
Bell Labs Innovations

- 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|

10

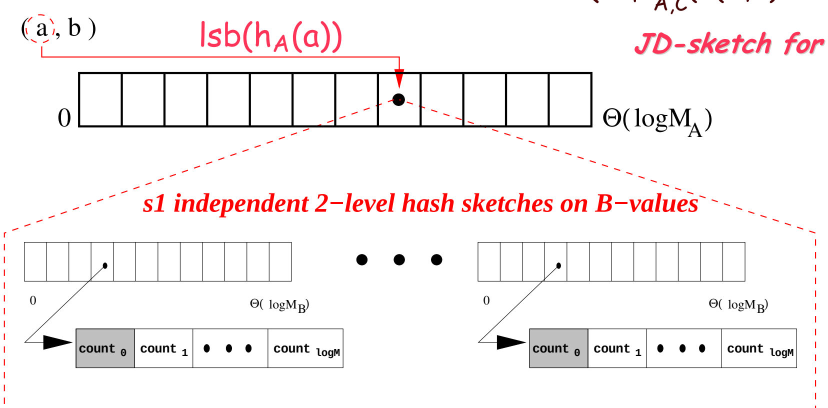
# The JD-Sketch Synopsis: Basic Structure

Lucent Technologies Bell Labs Innovations

- First level of hashing (hash fn+lsb) on the *projected stream attribute*
- Second level of hashing (collection of independent 2LHS) on the *join stream attribute*
- Maintenance: straightforward (based on 2LHS)
  - Composable, delete-proof, …

$Q = |\pi_{A,C}(R(A,B) \bowtie S(B,C))|$

***JD-sketch for R(A,B)***




11

# Our JD Estimator: *Composing* JD-Sketch Synopses

Lucent Technologies Bell Labs Innovations

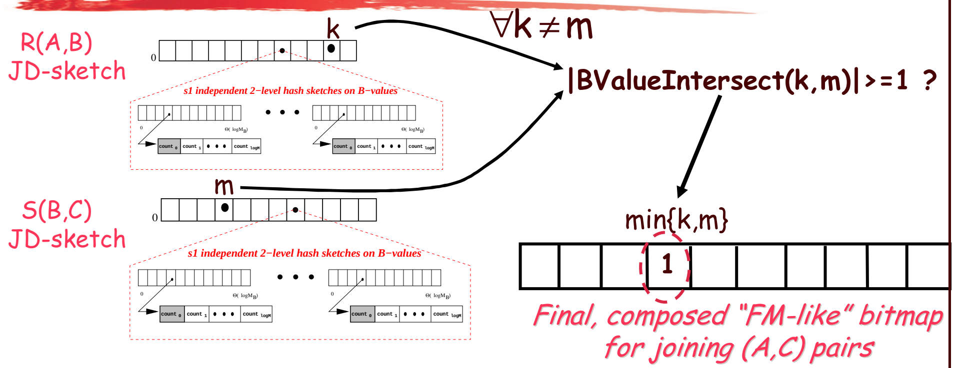
- **Input:** Pair of (independently-built) *parallel* JD-sketches on the R(A,B) and S(B,C) streams
  - Same hash functions for corresponding 2LHS pairs
- **Output:** FM-like summary (bitmap) for estimating the number of distinct *joining* (A,C) pairs

- ***Key Technical Challenges***
  - Want only *(A,C) pairs that join* to make it to our bitmap
    - *Idea:* Use 2LHS in the A- and C-buckets to determine (approximately) if the corresponding B-multisets *intersect*
  - A- and C-values are observed independently and in arbitrary order in their respective streams
    - Cannot directly hash arriving (A,C) pairs to a bitmap (traditional FM) -- all that we have are the JD-sketches for R, S!
    - *Idea:* Employ novel, *composable* hash functions $h_A()$, $h_C()$, and a sketch-composition algorithm that *guarantees FM-like properties*

12

## Our JD Estimator: *Composing* JD-Sketch Synopses

Lucent Technologies
Bell Labs Innovations



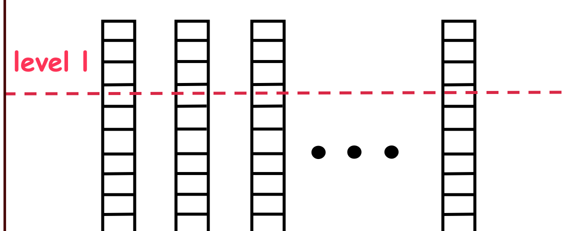

- **Theorem:** Using novel, composable linear hash functions, the above composition algorithm guarantees that
  - (A,C)-pairs map to final bitmap levels with exponentially-decreasing probabilities ($\approx 4^{-(l+1)}$)
  - (A,C)-pair mappings are *pairwise-independent*
- Both facts are crucial for our analysis…

13

## Our JD Estimator: Estimation Algorithm & Analysis

Lucent Technologies
Bell Labs Innovations

- Build and maintain s2 independent, parallel JD-sketch pairs over the R(A,B) and S(B,C) streams
- *At estimation time*
  - Compose each parallel JD-sketch pair, to obtain s2 "FM-like" bitmaps for joining (A,C) pairs
  - Find a level l in the composed bitmaps s.t. the fraction f of 1-bits lies in a certain range -- use f to estimate jd x Prob[level=l]
    - Return $jd \approx f \times 4^{l+1}$




$$\frac{count(1's)}{s2} \approx \frac{jd}{4^{l+1}}$$

14

## Our JD Estimator: Estimation Algorithm & Analysis

- **Theorem:** Our JD estimator returns an $(\varepsilon,\delta)$-estimate of JD cardinality using JD-sketches with a total space requirement of

$$O(\frac{U}{T}\frac{\log^2(1/\delta)}{\varepsilon^4}\log^3 M \log N)$$

  - U/T ≈ |B-value neighborhood|/ no. of joining B-values for randomly-chosen (A,C) pairs
    - JDs with low "support" are harder to estimate

- Lower bound based on information-theoretic arguments and Yao's lemma
  - Our space requirements are within constant and log factors of best possible

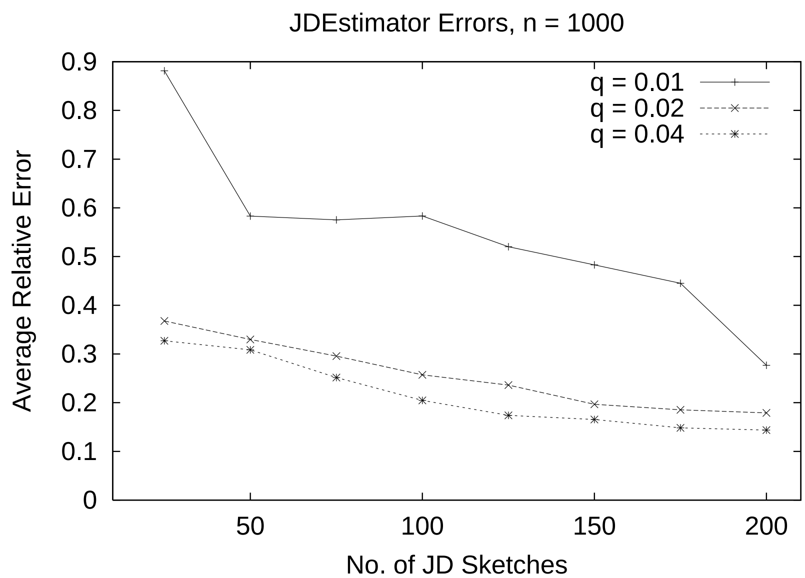
15

## Extensions

- Other forms of JD-cardinality queries are easy to handle with JD-sketches – for instance,
  - One-sided (semi)joins (e.g., $|\pi_{A,B}(R(A,B) \bowtie S(B,C))|$ )
  - "Full-projection" joins (e.g., $|\pi_{A,B,C}(R(A,B) \bowtie S(B,C))|$ )
  - Just choose the right stream attributes to hash on at the two levels of the JD-sketch
- Other JD-aggregates – e.g., estimating predicate selectivities over a JD operation
  - *Key observation:* Can use the JD-sketch to obtain a *distinct-value sample* of the JD result
- For cases where */B/ is small*, we propose a different, $\Theta(|B|)$-space JD synopsis and estimator
  - Based on simpler FM sketches built with *composable hash functions*
  - Conceptually simpler & easier to analyze, BUT requires at least linear space!

16

## Experimental Results: JD-Sketches on Random-Graph Data




17

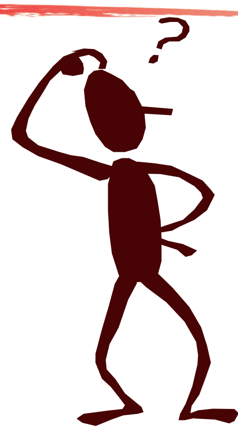
## Conclusions

- *First* space-efficient algorithmic techniques for estimating JD aggregates in the streaming model
- Novel, hash-based sketch synopses
  - Log-sized, delete-proof (general update streams)
  - Independently built over individual streams
  - Effectively *composed* at estimation time to provide approximate answers with strong probabilistic accuracy guarantees
- Verified effectiveness through preliminary experiments
- One key technical idea: *Composable Hash Functions*
  - Build hash-based sketches on individual attributes that can be *composed* into a sketch for *attribute combinations*
  - Powerful idea that could have applications in other streaming problems…

18

# Thank you!

Lucent Technologies
Bell Labs Innovations



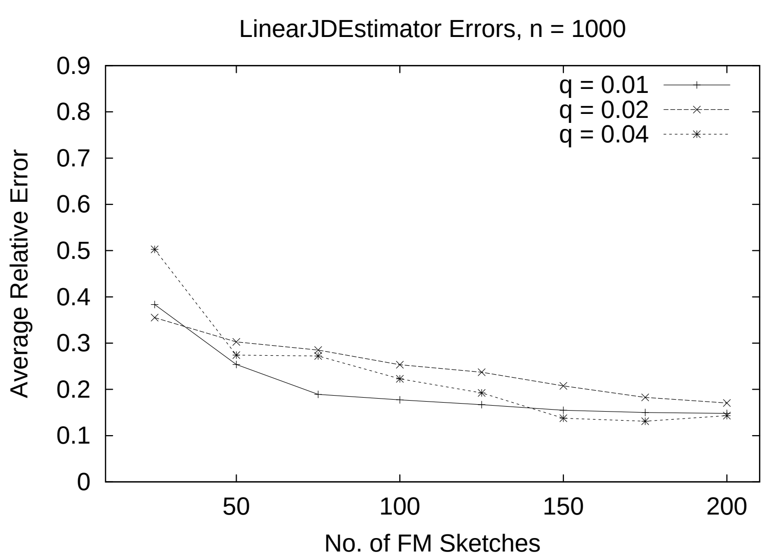
*http://www.bell-labs.com/~minos/*

*minos@research.bell-labs.com*

19

# Experimental Results: Linear-Space JD-Estimator on Random-Graph Data

Lucent Technologies
Bell Labs Innovations




20