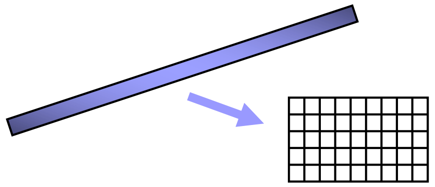
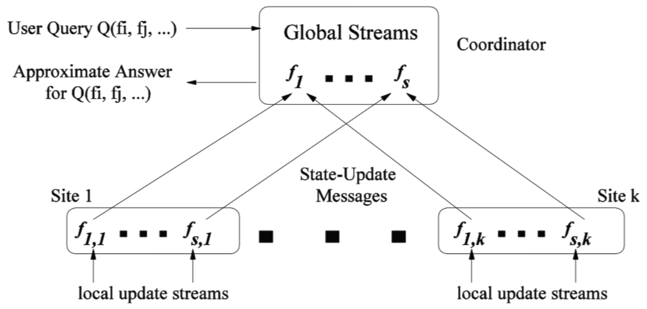
# A Quick Introduction to Data Stream Algorithmics





***Minos Garofalakis***

Yahoo! Research & UC Berkeley

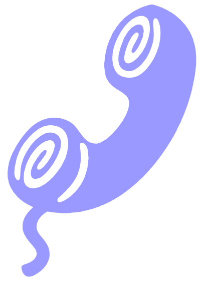
minos@acm.org

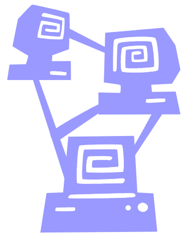
## Streams – A Brave New World

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

## Massive Data Streams

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



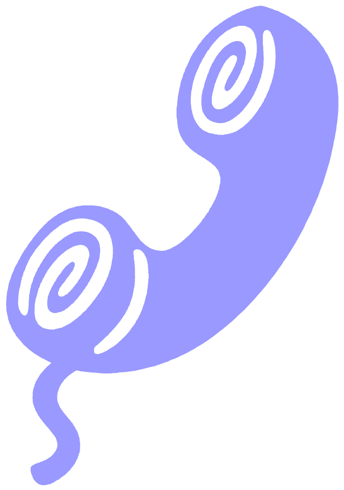


## Massive Data Stream Analysis

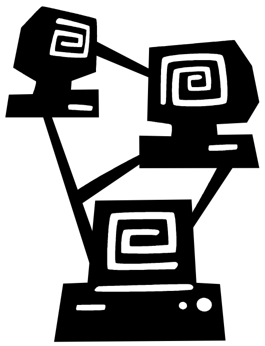
Must analyze this massive data:

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

Else, why even measure this data?

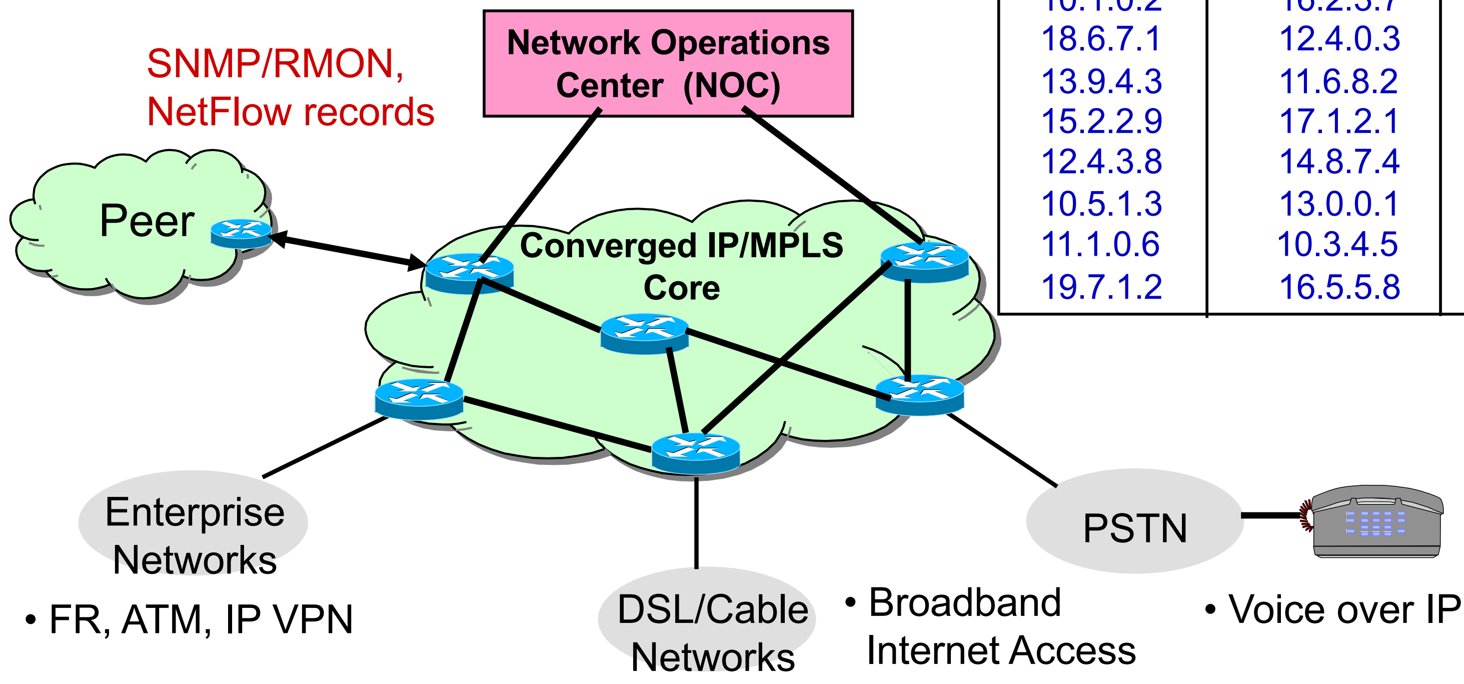


## Example: IP Network Data



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

# IP Network Monitoring Application



| 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 |

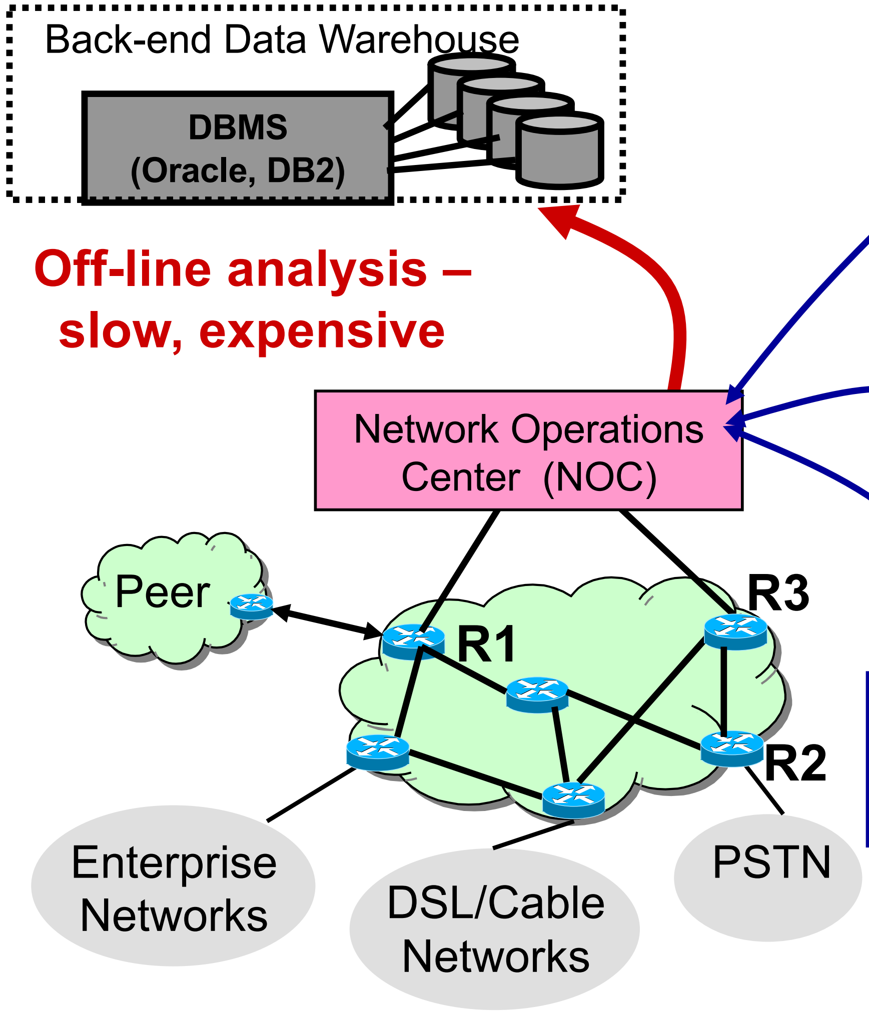
**Example NetFlow IP Session Data**

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

## Packet-Level Data Streams

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

# Network Monitoring Queries



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

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

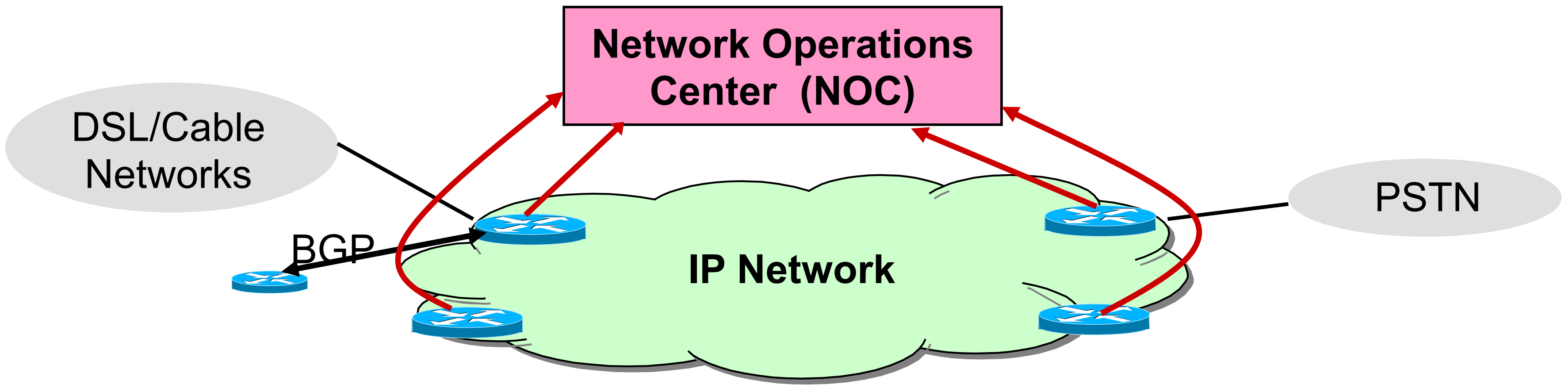
***Set-Expression Query***

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

***SQL Join Query***

- Extra complexity comes from *limited space and time*
- Solutions exist for these and other problems

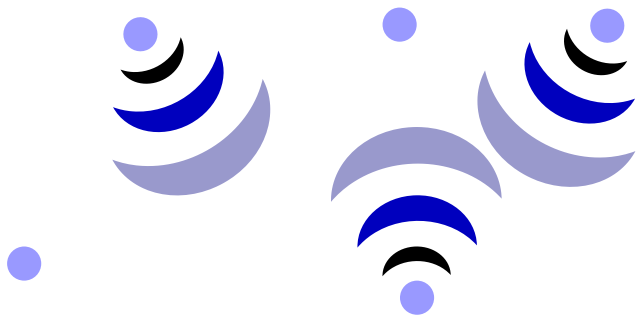
# Real-Time Data-Stream Analysis



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

## Sensor Networks

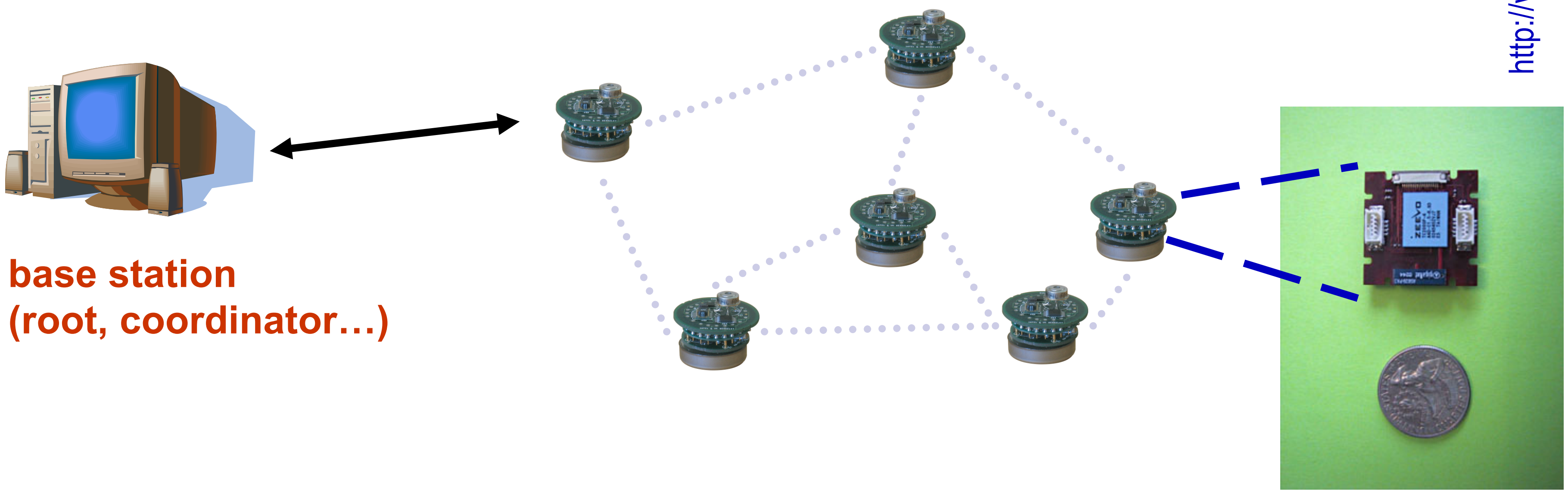
- Wireless sensor networks becoming ubiquitous in environmental monitoring, military applications, …
- Many (100s, $10^3$, $10^6$?) sensors scattered over terrain
- Sensors observe and process a local stream of readings:
  - Measure light, temperature, pressure…
  - Detect signals, movement, radiation…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



## Sensornet Querying Application

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

http://www.intel.com/research/exploratory/motes.htm



## Lecture Outline

- Motivation & Streaming Applications
- Centralized Stream Processing
  - Basic streaming models and tools
  - Stream synopses and applications
    - Sampling, sketches
- Conclusions

## Data Streaming Model

- Underlying signal: One-dimensional array A[1…N] with values A[i] all initially zero
  - Multi-dimensional arrays as well (e.g., row-major)
- Signal is implicitly represented via a *stream of update tuples*
  - j-th update is <x, c[j]> implying
    - A[x] := A[x] + c[j]     (c[j] can be >0, <0)

- Goal: Compute functions on A[] subject to
  - Small space
  - Fast processing of updates
  - Fast function computation
  - …
- Complexity arises from massive length and domain size (N) of streams

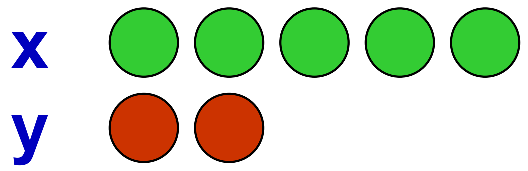
## Example IP Network Signals

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

## Streaming Model: Special Cases

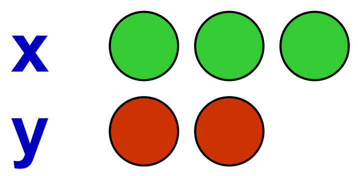
- Time-Series Model
  - Only x-th update updates A[x] (i.e., A[x] := c[x])

- Cash-Register Model: Arrivals-Only Streams
  - c[x] is always > 0
  - Typically, c[x]=1, so we see a multi-set of items in one pass
  - Example: <x, 3>, <y, 2>, <x, 2> encodes the arrival of 3 copies of item x, 2 copies of y, then 2 copies of x.
  - Could represent, e.g., packets on a network; power usage



# Streaming Model: Special Cases

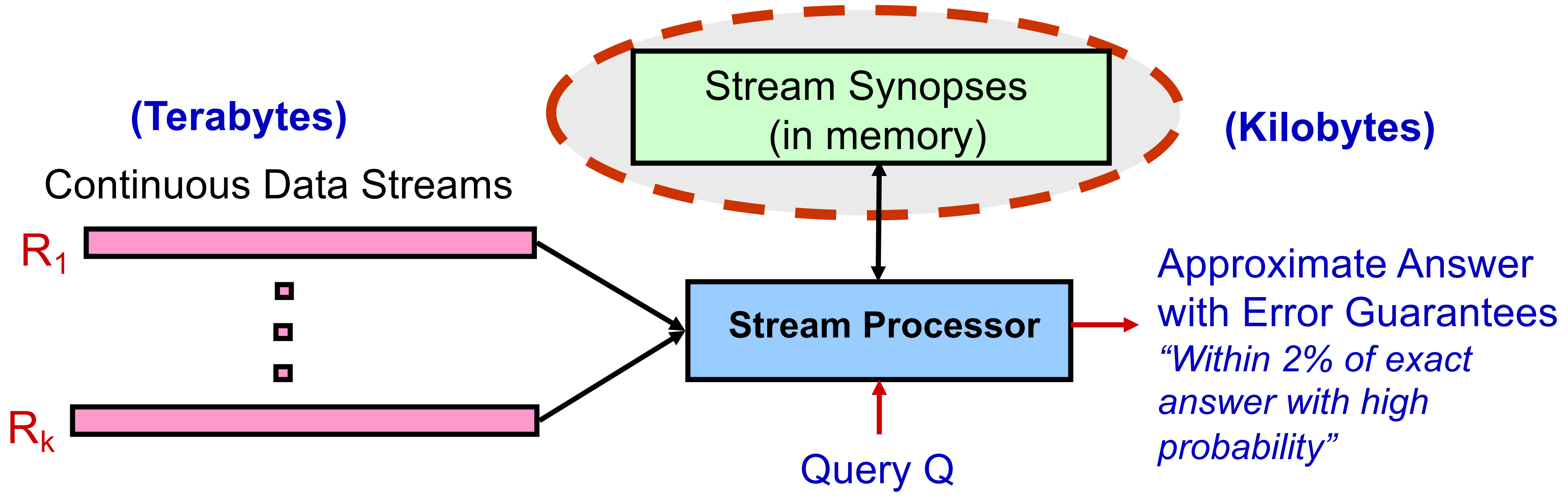
- Turnstile Model: Arrivals and Departures
  - Most general streaming model
  - c[x] can be >0 or <0
- Arrivals and departures:
  - Example: <x, 3>, <y,2>, <x, -2> encodes final state of <x, 1>, <y, 2>.
  - Can represent fluctuating quantities, or measure differences between two distributions



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

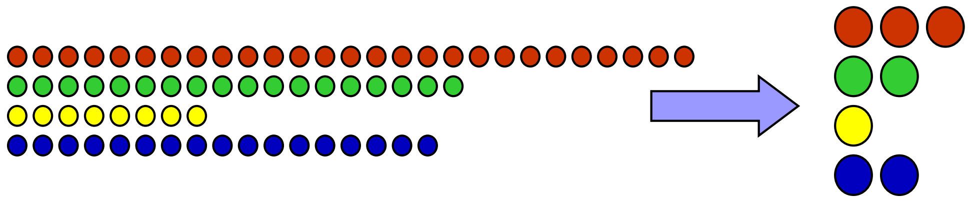
YAHOO! RESEARCH

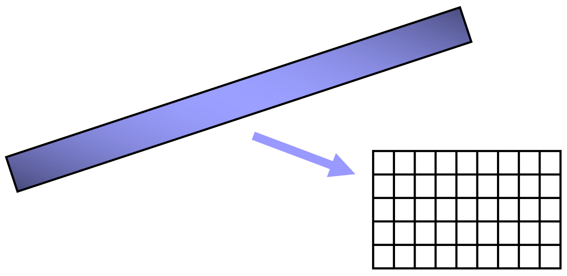
# Data-Stream Algorithmics Model


(Terabytes)
Continuous Data Streams
$R_1$
$R_k$
Stream Synopses (in memory)
(Kilobytes)
Stream Processor
Query Q
Approximate Answer with Error Guarantees
*"Within 2% of exact answer with high probability"*

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

# Sampling & Sketches





## Sampling: Basics

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

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

Sample S: 9 5 1 8

  - If agg is avg, return average of odd elements in S

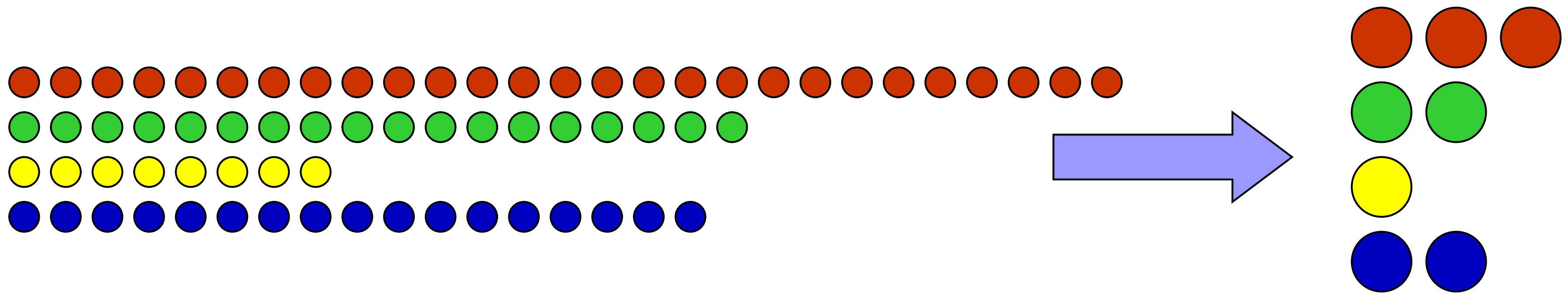
answer: 5

  - If agg is count, return average over all elements e in S of
    - n if e is odd
    - 0 if e is even

answer: 12*3/4 =9

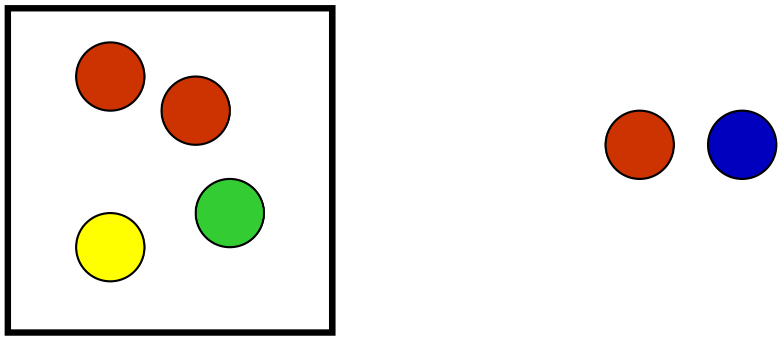
- ***Unbiased Estimator*** (for count, avg, sum, etc.)
  - Bound error using *Hoeffding* (sum, avg) or *Chernoff* (count)

## Sampling from a Data Stream



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

## Reservoir Sampling



- Sample first m items
- Choose to sample the i'th item (i>m) with probability m/i
- If sampled, randomly replace a previously sampled item

- Optimization: when i gets large, compute which item will be sampled next, skip over intervening items [Vitter'85]

## Reservoir Sampling - Analysis

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

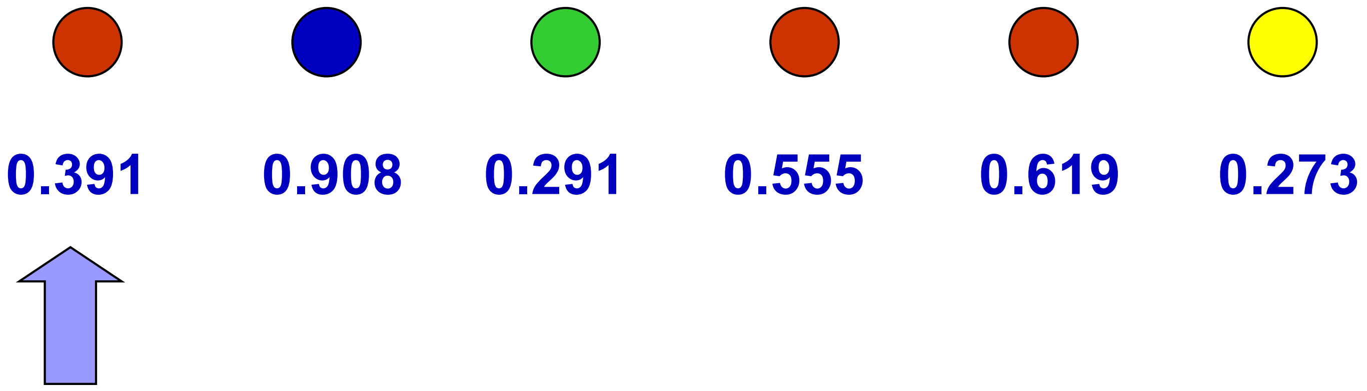
$$\frac{1}{\cancel{i}} \times \frac{\cancel{i}}{\cancel{i+1}} \times \frac{\cancel{i+1}}{\cancel{i+2}} \cdots \frac{\cancel{n-2}}{\cancel{n-1}} \times \frac{\cancel{n-1}}{n}$$

$$= 1/n$$

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

## Min-wise Sampling

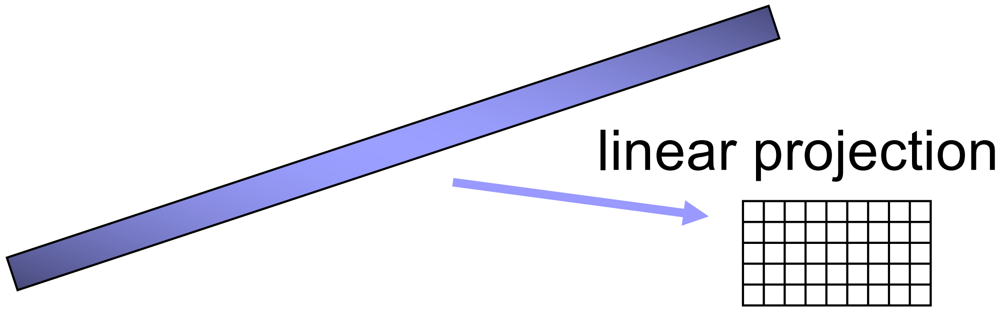
- For each item, pick a random fraction between 0 and 1
- Store item(s) with the smallest random tag [Nath et al.'04]



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

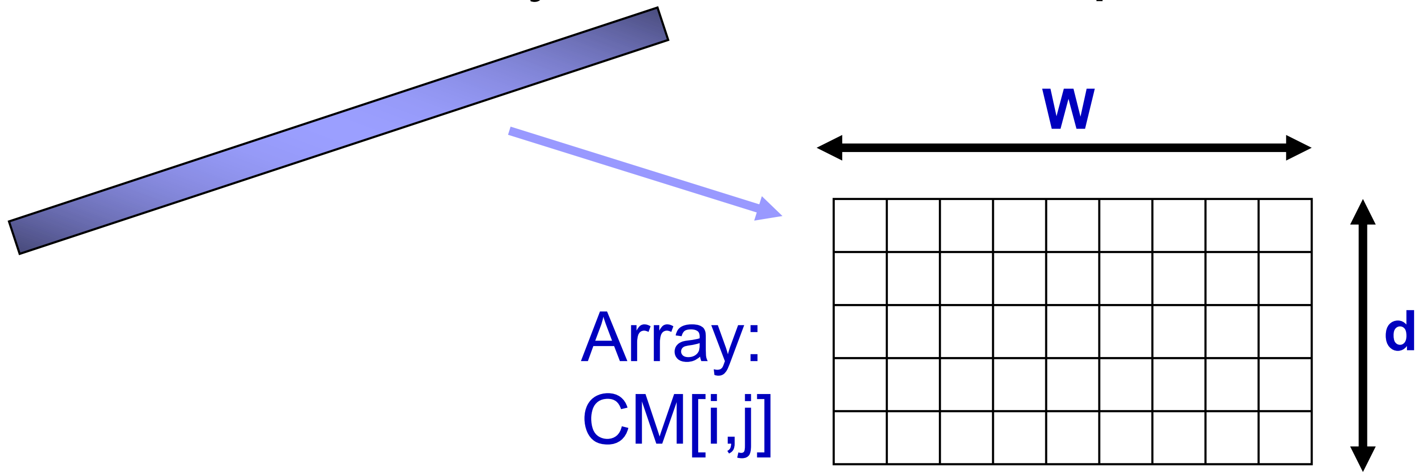
## Sketches

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

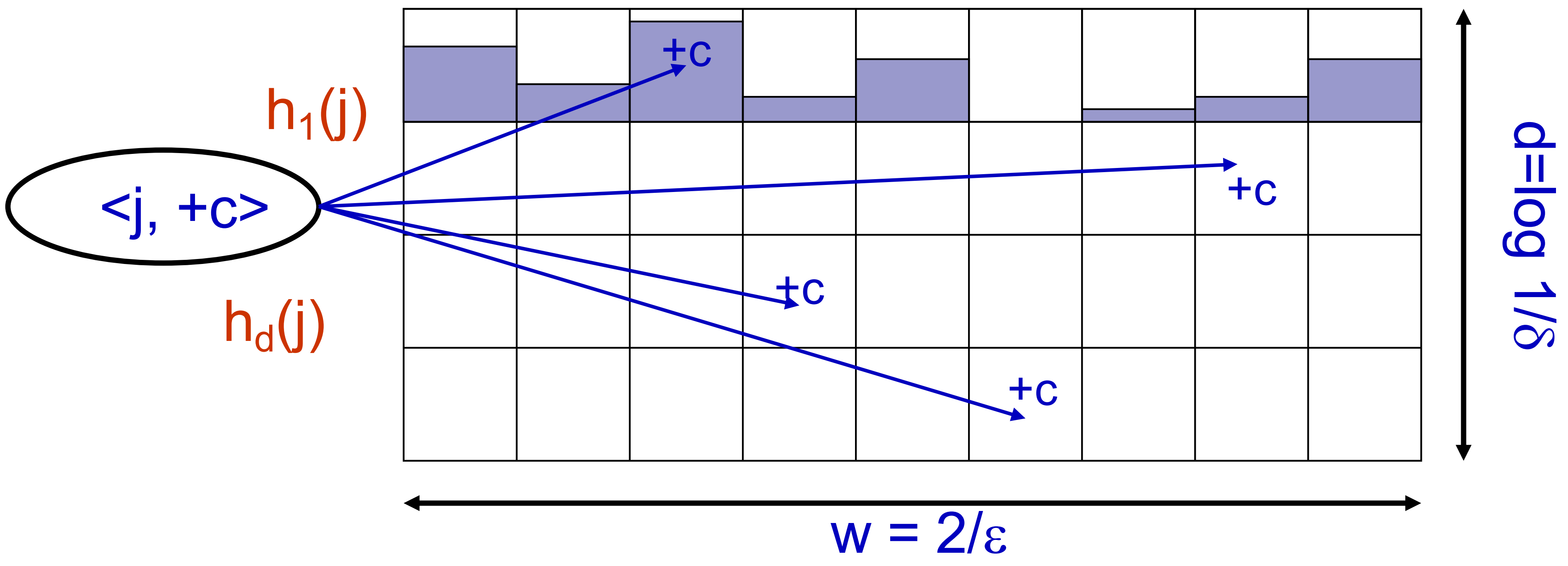


## Count-Min Sketch [Cormode, Muthukrishnan'04]

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



# CM Sketch Structure



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

## CM Sketch Guarantees

- *[Cormode, Muthukrishnan'04]* CM sketch guarantees approximation error on point queries less than $\varepsilon||A||_1$ in space $O(1/\varepsilon \log 1/\delta)$
  - Probability of more error is less than $1-\delta$
  - Similar guarantees for range queries, quantiles, join size,…

- Hints
  - Counts are *biased (overestimates)* due to collisions
    - Limit the expected amount of extra "mass" at each bucket?
  - Use independence across rows to boost the confidence for the min{} estimate
    - Based on independence of row hashes

# CM Sketch Analysis

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

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

# Distinct Value Estimation

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

- Example (N=64)

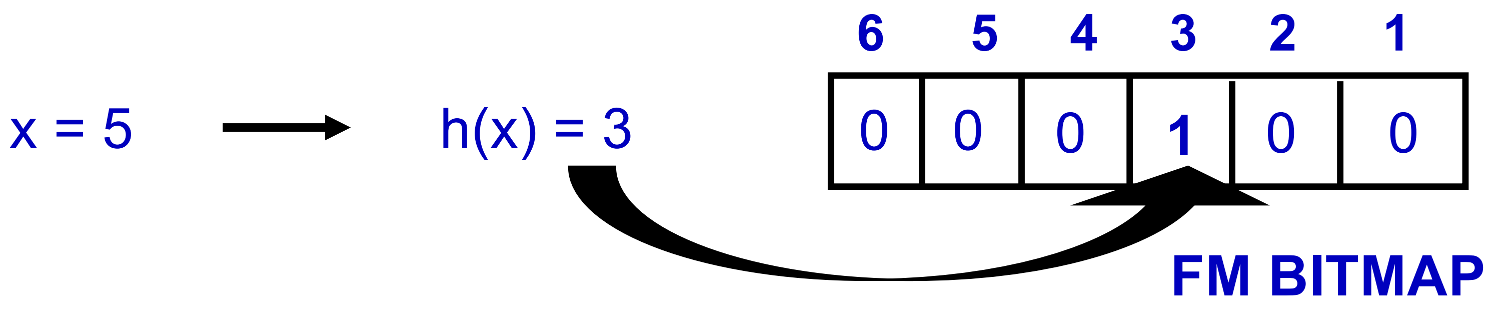
Data stream: 3 2 5 3 2 1 7 5 1 2 3 7

*Number of distinct values: 5*

- Hard problem for random sampling! [Charikar et al.'00]
  - Must sample almost the entire table to guarantee the estimate is within a factor of 10 with probability > 1/2, regardless of the estimator used!
- AMS and CM only good for *multiset semantics*

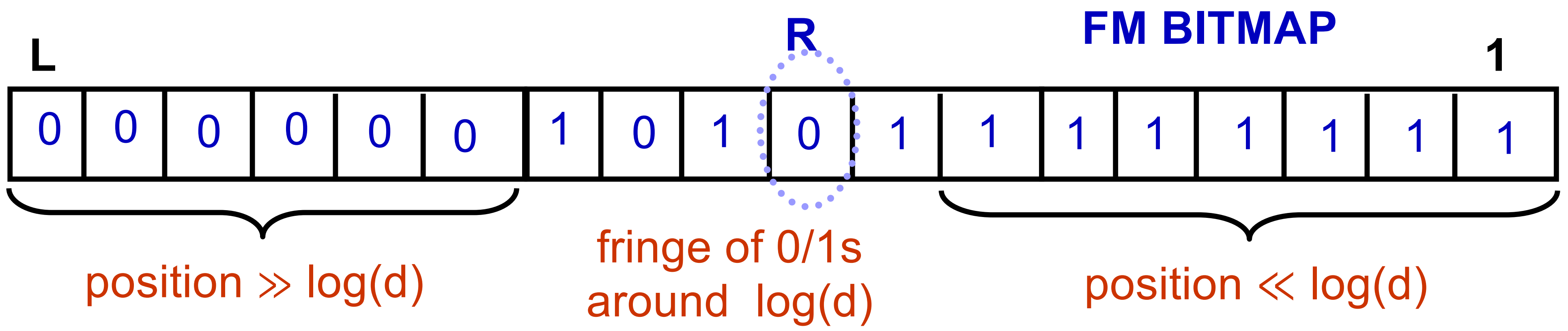
# FM Sketch [Flajolet, Martin'85]

- Estimates number of distinct inputs (`count distinct`)
- Uses hash function mapping input items to i with prob $2^{-i}$
  - i.e. Pr[h(x) = 1] = ½, Pr[h(x) = 2] = ¼, Pr[h(x)=3] = 1/8 …
  - Easy to construct h() from a uniform hash function by counting trailing zeros
- Maintain FM Sketch = bitmap array of L = log N bits
  - Initialize bitmap to all 0s
  - For each incoming value x, set FM[h(x)] = 1



# FM Sketch Analysis

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



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

## FM Sketch Properties

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

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

  - Estimate $|S_1 [\cdots[ S_k|$ = *set union cardinality*

## Sketching and Sampling Summary

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

## Practicality

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

## Conclusions

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

# Approximation and Randomization

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

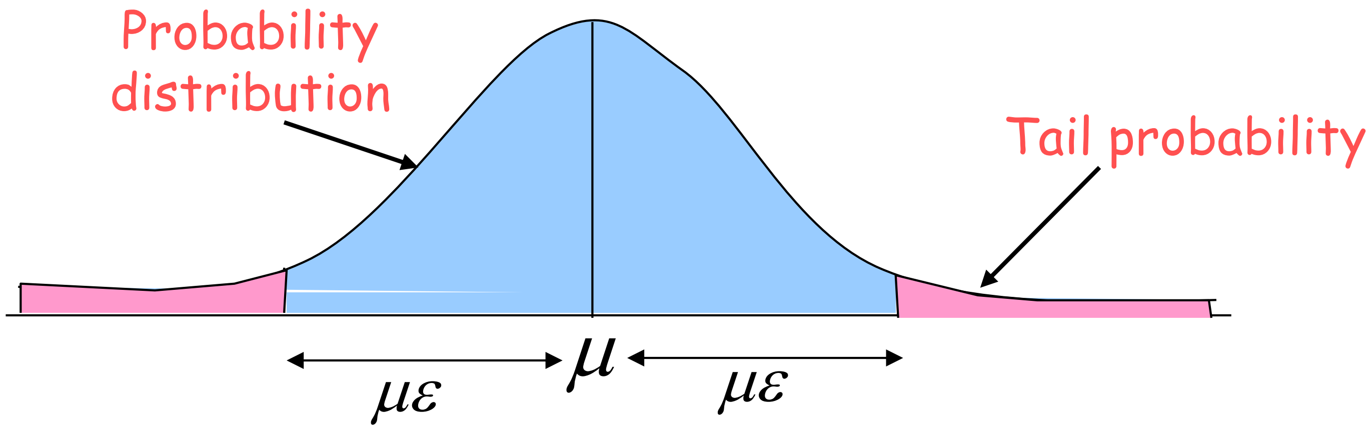
YAHOO! RESEARCH

# Probabilistic Guarantees

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

## Basic Tools: Tail Inequalities

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



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

***Markov:***

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

***Chebyshev:***

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

# Tail Inequalities for Sums

- Possible to derive stronger bounds on tail probabilities for the sum of independent random variables
- *Hoeffding Bound:* Let X1, ..., Xm be independent random variables with 0· Xi · r. Let $\bar{X} = \frac{1}{m}\sum_i X_i$ and $\mu$ be the expectation of $\bar{X}$. Then, for any $\varepsilon > 0$,

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

- *Application:* Sample average ¼ population average
  - See below…

## Tail Inequalities for Sums

- Possible to derive even stronger bounds on tail probabilities for the sum of independent *Bernoulli trials*
- *Chernoff Bound:* Let X1, ..., Xm be independent Bernoulli trials such that Pr[Xi=1] = p (Pr[Xi=0] = 1-p). Let $X = \sum_i X_i$ and $\mu = mp$ be the expectation of $X$. Then, for any $\varepsilon > 0$,

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

- *Application:* Sample selectivity ¼ population selectivity
  - See below…
- *Remark:* Chernoff bound results in tighter bounds for *count queries* compared to Hoeffding bound