Algorithms for Processing Massive Data at Network Line Speeds

Graham Cormode, DIMACS
graham@dimacs.rutgers.edu

Joint work with S. Muthukrishnan

Outline

- What's the problem?
- What's hot and what's not?
- What's new?
- What's next?

Data is Massive

Data is growing faster than our ability to store or process 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) of routers!

Massive Data Analysis

MassDAL
Massive Data Analysis Lab

Must analyze this massive data:

- System management (spot faults, drops, failures)
- Customer research (association rules, new offers)
- For revenue protection (phone fraud, service abuse)
- Scientific research (Climate Change, SETI etc.)

Else, why even measure this data?

Focus: Network Data

MassDAL
Massive Data Analysis Lab

- Networks are sources of massive data: the metadata per hour per router is gigabytes
- 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

Network Data Questions

MassDAL
Massive Data Analysis Lab

Network managers ask questions that often map onto "simple" functions of the data.

- How many distinct host addresses?
- Destinations using most bandwidth?
- Address with biggest change in traffic overnight?

The complexity comes from space and time restrictions.

Data Stream Algorithms

- Recent interest in "*data stream algorithms*“ from theory: small space, one pass approximations

- Alon, Matias, Szegedy 96: frequency moments
  Henzinger, Raghavan, Rajagopalan 98 graph streams

- In last few years:
  Counting distinct items, finding frequent items, quantiles, wavelet and Fourier representations, histograms...

The Gap

A big gap between theory and practice: many good theory results aren't yet ready for primetime.

Approximate within $1 \pm \varepsilon$ with probability $> 1-\delta$. Eg: AMS sketches for F_2 estimation, set $\varepsilon=1\%$, $\delta=1\%$

- Space $O(1/\varepsilon^2 \log 1/\delta)$ is approx 10^6 words = 4Mb
  Network device may have 100k-4Mb space *total*
- Each data item requires pass over whole space
  At network line speeds can afford a few dozen memory accesses, perhaps more with parallelization

Bridging the Gap

MassDAL
Massive Data Analysis Lab

My work sets out to bridge the gap: the Count-Min sketch and change detection data structures.

- Simple, small, fast data stream summaries which have been implemented to solve several problems
- Some subtlety: to beat $1/\varepsilon^2$ lower bounds, must explicitly avoid estimating frequency moments
- Here: Application to fundamental problems in networks and beyond, finding heavy hitters and large changes

Outline

- What's the problem?
- **What's hot and what's not?**
- What's new?
- What's next?

1. Heavy Hitters

MassDAL
Massive Data Analysis Lab

- Focus on the Heavy Hitters problem: Find users (IP addresses) consuming more than 1% of bandwidth
- In algorithms, "Frequent Items": Find items and their counts when count more than ϕN
- Two versions:
  a) arrivals only : models most network scenarios
  b) arrivals and departures : applicable to databases

Prior Work

Heavily studied problem (for arrivals only):

- Sampling, keep counts of certain items:
  Gibbons, Matias 1998
  Manku, Motwani 2002
  Demaine, Lopez-Ortiz, Munro 2002
  Karp, Papadimitriou, Shenker 2003

- Filter or sketch based:
  Fang, Shivakumar, Garcia-Molina, Motwani, Ullman 1998
  Charikar, Chen, Farach-Colton 2002
  Estan, Varghese 2002

No prior solutions for arrivals and departures before this.

Stream of Packets

- Packets arrive in a stream. Extract from header:
  Identifier, i: Source or destination IP address
  Count: connections / packets / bytes

- Stream defines a vector a[1..U], initially all 0
  Each packet increases one entry, a[i].
  In networks $U = 2^{32}$ or 2^{64}, too big to store

- Heavy Hitters are those i's where $a[i] > \phi N$
  Maintain N = sum of counts

Arrivals Only Solution

MassDAL
Massive Data Analysis Lab

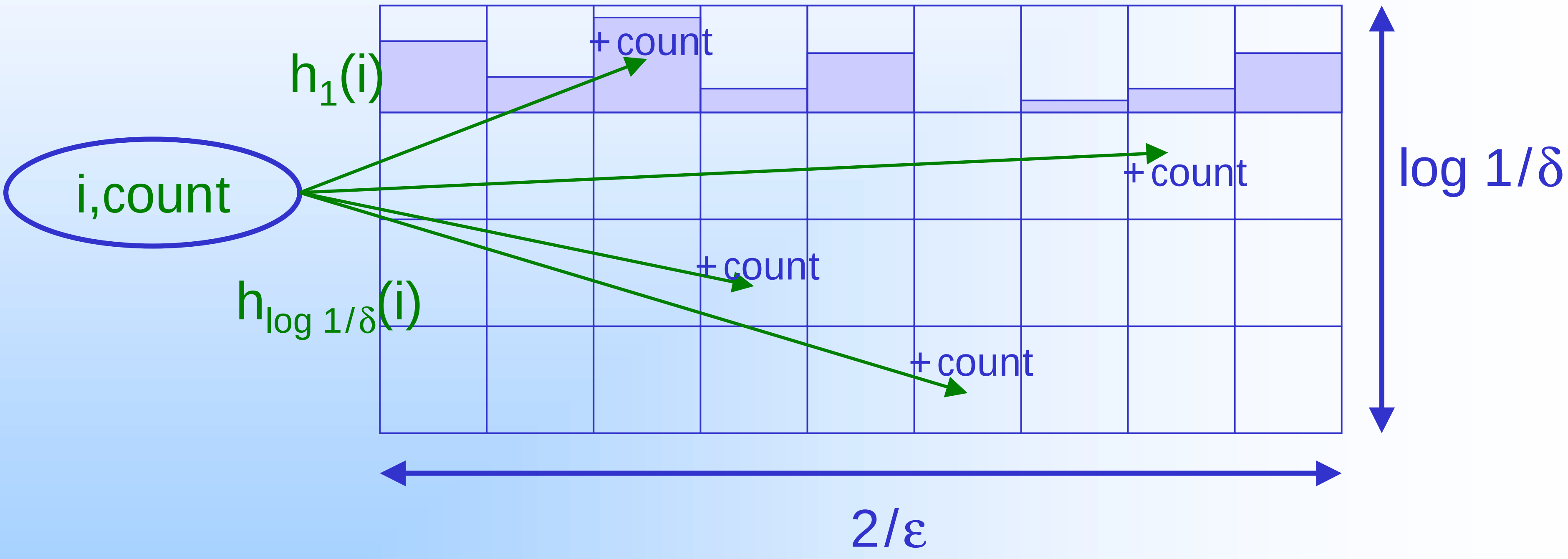
Naive solution: keep the array a and for every item in stream, test if $a[i] > \phi N$. Keep heap of items that pass since item can only become a HH following insertion.

Solution here: replace $a[i]$ with a small data structure which approximates all $a[i]$ upto εN with prob $1-\delta$

Ingredients:

- Universal hash fns $h_1..h_{\log 1/\delta}$ $\{1..U\} \rightarrow \{1..2/\varepsilon\}$
- Array of counters $CM[1..2/\varepsilon, 1..\log_2 1/\delta]$

Update Algorithm



Count-Min Sketch

Approximation

Approximate $\hat{a}[i] = \min_j CM[h_j(i),j]$

Analysis: In j'th row, $CM[h_j(i),j] = a[i] + X_{i,j}$

$$X_{i,j} = \Sigma\, a[k] \mid h_j(i) = h_j(k)$$

$$\begin{aligned} E(X_{i,j}) &= \Sigma\, a[k]*Pr[h_j(i)=h_j(k)] \\ &\leq Pr[h_j(i)=h_j(k)] * \Sigma\, a[k] \\ &= \varepsilon N/2 \text{ by pairwise independence of } h \end{aligned}$$

Analysis

MassDAL
Massive Data Analysis Lab

$$\Pr[X_{i,j} \geq \varepsilon N] = \Pr[X_{i,j} \geq 2E(X_{i,j})]$$
$$\leq 1/2$$ by Markov inequality

Hence, $\Pr[\hat{a}[i] \geq a[i] + \varepsilon N] = \Pr[\forall j.\ X_{i,j} > \varepsilon N]$
$$\leq 1/2^{\log 1/\delta} = \delta$$

Final result:

with certainty $a[i] \leq \hat{a}[i]$ and

with probability at least $1-\delta$, $\hat{a}[i] < a[i] + \varepsilon N$

Results for Heavy Hitters

MassDAL
Massive Data Analysis Lab

- Solve the arrivals only problem by remembering the largest estimated counts (in a heap).
- Every item with count $> \phi N$ is output and with prob $1-\delta$, each item in output has count $> (\phi-\varepsilon)N$
- Space $= 2/\varepsilon \log_2 1/\delta$ counters $+ \log_2 1/\delta$ hash fns
  Time per update $= \log_2 1/\delta$ hashes
  (Universal hash functions are fast and simple)
- Fast enough and lightweight enough for use in network implementations

Implementation Details

MassDAL
Massive Data Analysis Lab

Implementations work pretty well, better than theory suggests: 3 or so hash functions suffice in practice

Running in AT&T's Gigascope, on live 2.4Gbs streams

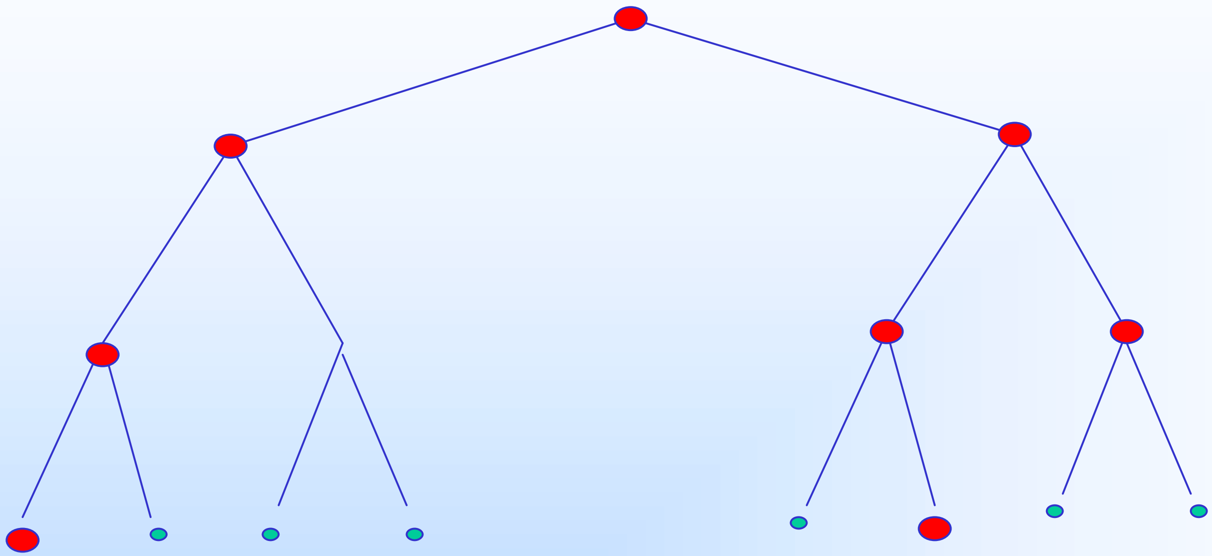
– Each query may fire many instantiations of CM sketch, how do they scale?
– Should sketching be done at low level (close to NIC) or at high level (after aggregation)?
– Always allocate space for a sketch, or run exact algorithm until count of distinct IPs is large?

Solutions with Departures

MassDAL
Massive Data Analysis Lab

- When items depart (eg deletions in a database relation), finding heavy hitters is more difficult.
- Items from the past may become heavy, following a deletion, so need to be able to recover item labels.
- Impose a (binary) tree structure on the universe, nodes correspond to sum of counts of leaves.
- Keep a sketch for nodes in each level and search the tree for frequent items with divide and conquer.

Search Structure



Find all items with count > ϕN by divide and conquer
(play off update and search time by changing degree)

Sketch structure is an oracle for *adaptive group testing*

Outline

MassDAL
Massive Data Analysis Lab

- What's the problem?
- What's hot and what's not?
- **What's new?**
- What's next?

2. Change Detection

- Find items with big change between streams x and y
  Find IP addresses with big change in traffic overnight
- "Change" could be absolute difference in counts, or large ratio, or large variance...
- Absolute difference: find large values in $|a(x) - a(y)|$
  Relative difference: find large values $a(x)[i]/a(y)[i]$
- CM sketch can approximate the differences, but how to find the items without testing everything?
  Divide and conquer (adaptive testing) won't work here!

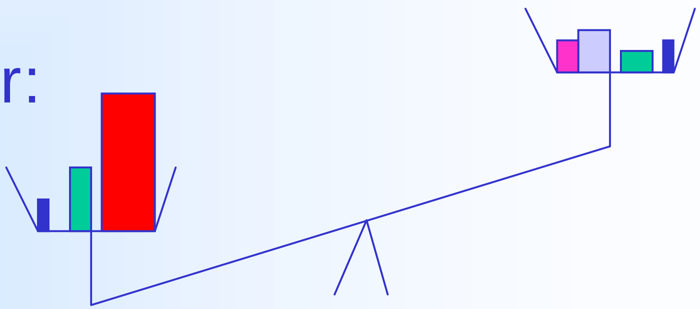
Change Detection

MassDAL
Massive Data Analysis Lab

- Use Non-Adaptive Group Testing: will pick groups of items in a randomized fashion
- Within each group, test for "deltoids": items that have shown a large change in behavior
- Must keep more information than just counts to recover identity of deltoids.
- We separate the structure of the groups from the tests, and consider each in turn.

Groups: Simple Case

- Suppose there is just one large item, i, whose "weight" is more than half the weight of all items.
- Use a pan-balance metaphor: this item will always be on the heavier side



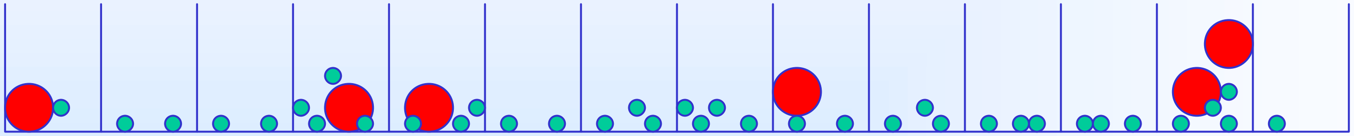
- Assume we have a test which tells us which group is heavy. The large item is always in that group.
- Arrange these tests to let us identify the deltoid.

Solving the simple case

- Keep a test of items whose identifier is odd, and for even: result of test tells whether i is odd or even
- Similarly, keep tests for every bit position.
- Then can just read off the index of the heavy item
- Now, turn original problem into this simple case…

Spread into Buckets

Allocate items into buckets:



- With enough buckets, we expect to achieve the simple case: each deltoid lands in a bucket where the rest of weight is small
- Repeat enough times independently to guarantee finding all deltoids

Group Structure

Formalize the scheme to find deltoids with weight at least $\phi - \varepsilon$ of total amount of change:

- Use a universal hash function to divide the universe into $2/\varepsilon$ groups, repeat $\log 1/\delta$ times.
- Keep a test for each group to determine if there is a deltoid within it. Keep $2\log U$ subgroups in each group based on the bit positions to identify deltoids.

Update procedure: for each update, find the groups the items belongs to and update the corresponding tests.

Group Testing

- Searching: For each group whose test is positive, read results of tests of subgroups: if test j is positive, bit $j = 1$, test j' positive, bit $j=0$
- Avoid false positives: If test j and j' both positive, there are two deltoids in same group, so reject the group (also if j and j' both negative).
- Avoid false positives: Check the recovered item belongs to that group. If so, output it as a deltoid.
- Result: Find all deltoids, if tests gave correct results.

Tests

- How to construct a test for the presence of a deltoid?
- Naively, could keep sketch for each group, but space blows up ($1/\varepsilon^2$ or worse)
- For absolute change deltoids, keeping counts of items suffices, proof similar to CM sketch
- For relative change, appropriate counts also suffice, new proof needed.

Relative Change Test

Keep different information for each stream.

- For stream x, keep $T(x)[j] = \Sigma_{h(i) = j}\, a(x)[i]$
  sum counts of items in the group
- For stream y, keep $T(y)[j] = \Sigma_{h(i) = j}\, (1/a(y)[i])$
  sum reciprocal of counts of items in the group
- Test: if $T(x)[j]*T(y)[j] > \phi\, \Sigma\, (a(x)[i]/a(y)[i])$
  test if product of counts exceeds threshold
- Must be able to find $(1/a(y)[i])$ – open problem to remove this restriction

Relative Change Test

- Test has one-sided error, will always say yes if

  $$(a(x)[i]/a(y)[i]) > \phi \, \Sigma \, (a(x)[i]/a(y)[i])$$

- To bound false positives, and ensure true positives are not obscured by noise, need to argue that each test gives good enough estimate of $(a(x)[i]/a(y)[i])$

- In full paper, show that expected error is $\frac{1}{2} \varepsilon \, \|a(x)\|_1 \, \|1/a(y)\|_1$. So with constant probability this is good estimate of the change.

- The group structure amplifies this probability to $1-\delta$

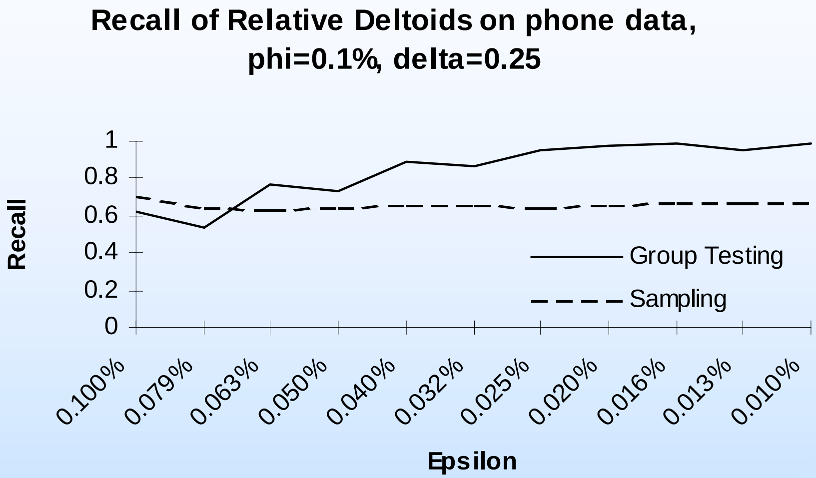
Results

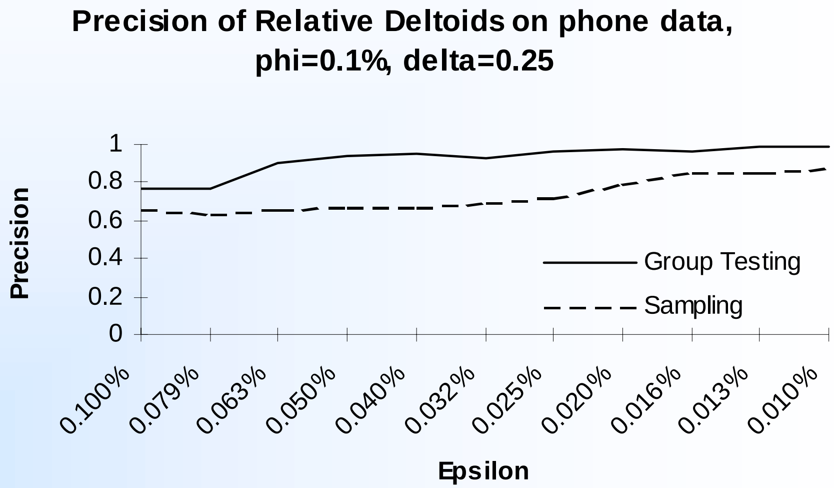
- With probability $1-\delta$, all deltoids are found, no items which are far from being deltoids

- Space is $O(1/\varepsilon \log U \log 1/\delta)$
  Update time is $O(\log U \log 1/\delta)$ per item
  Time to search is linear in the space used

- The same group structure works for different objective functions, if there is an efficient test.

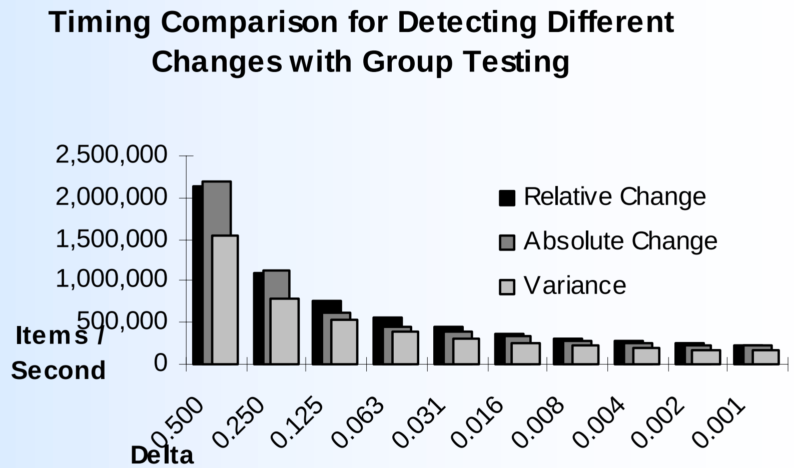
Experiments





Recall = fraction of deltoids found

Precision = fraction of returned items that are deltoids



Outline

MassDAL
Massive Data Analysis Lab

- What's the problem?
- What's hot and what's not?
- What's new?
- **What's next?**

Other Applications

These techniques can be applied to several other fundamental data analysis problems:

– Range Sum and Inner Product Estimation

– Finding Approximate Quantiles

– Wavelets and Histograms…

Limited (pairwise) independence suffices for all

Group testing approach is fundamental

Ongoing Work

Agenda: Move other data mining methods from the theoretical to the practical for massive data, in similar and new domains:

- Burst detection on many (large) texts
- Items in hierarchies, eg IP addresses, geographic data
- Massive geometric data — many points from mobile clients.
- Massive Graphs — eg call graphs, web graph

References

- "What's Hot and What's Not: Tracking Most Frequent Items Dynamically" Principles of Database Systems (PODS) 2003
- "An improved data stream summary: the Count-Min sketch and its applications" Journal of Algorithms, 2004
- "What's New: Finding Significant Differences in Network Data Streams" INFOCOM 2004

(all joint work with S. Muthukrishnan)

Code for these algorithms and others is publicly available
http://www.dimacs.rutgers.edu/~graham/code/

http://www.dimacs.rutgers.edu/~graham/
Or web search for "Graham Cormode"