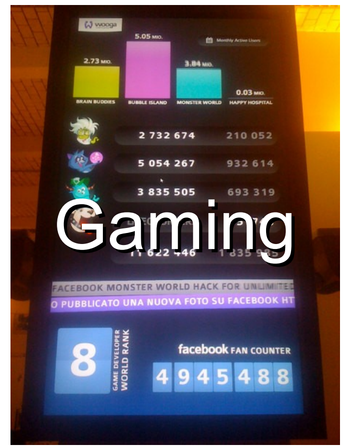
Beyond Scaling: Real-Time Event Processing with Stream Mining

Mikio L. Braun, @mikiobraun
http://blog.mikiobraun.de

TWIMPACT
http://twimpact.com

Event Data

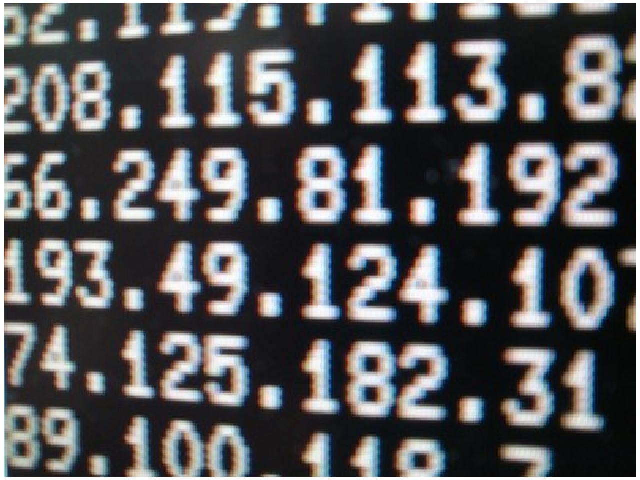


Attribution: flickr users kenteegardin, fguillen, torkildr, Docklandsboy, brewbooks, ellbrown, JasonAHowie

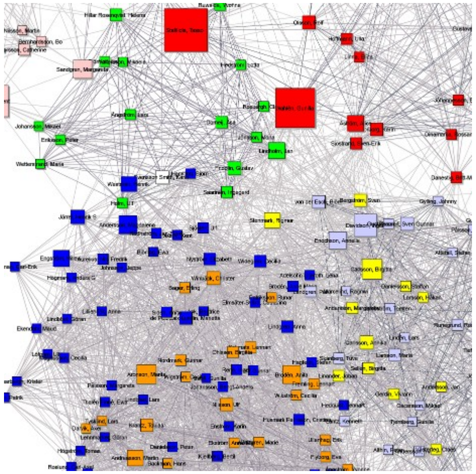
Event Data is Huge: Volume

- The problem: You easily get A LOT OF DATA!
  - 100 events per second
  - 360k events per hour
  - 8.6M events per day
  - 260M events per month
  - 3.2B events per year

Event Data is Huge: Diversity

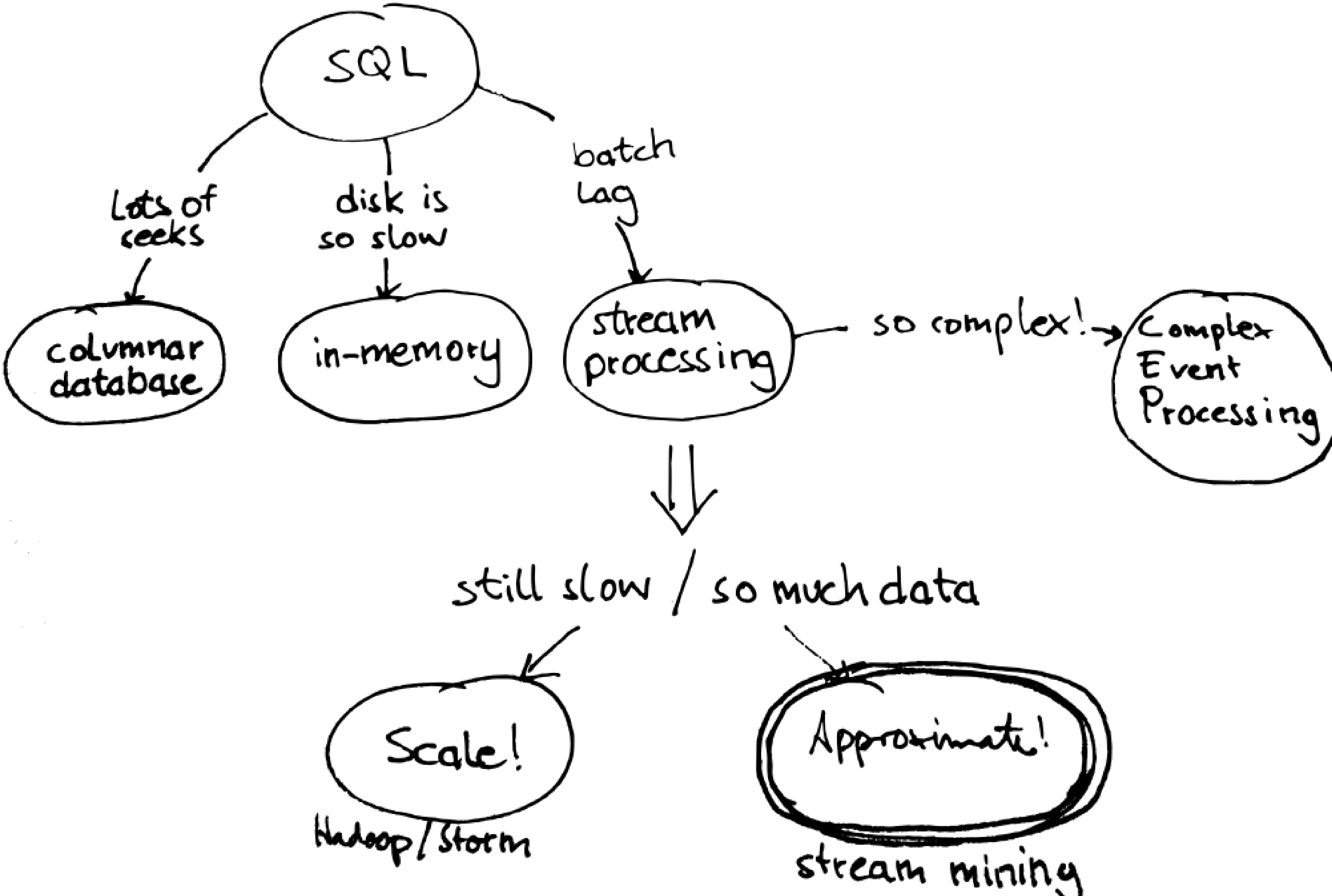


- Potentially large spaces:
  - distinct words: >100k
  - IP addresses: >100M
  - users in a social network: >10M



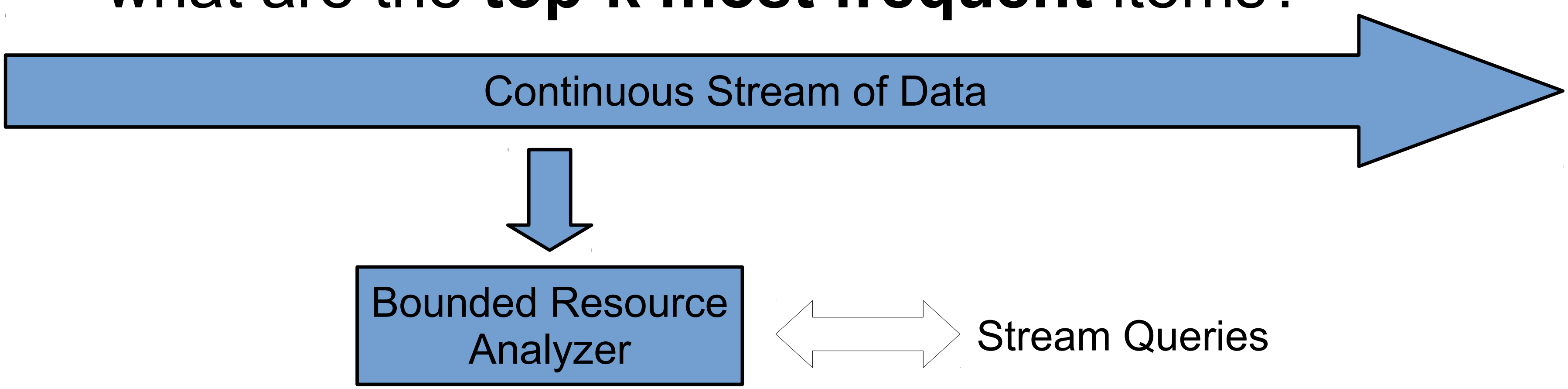
http://wordle.net http://www.flickr.com/photos/arenamontanus/269158554/

To Scale Or Not To Scale?

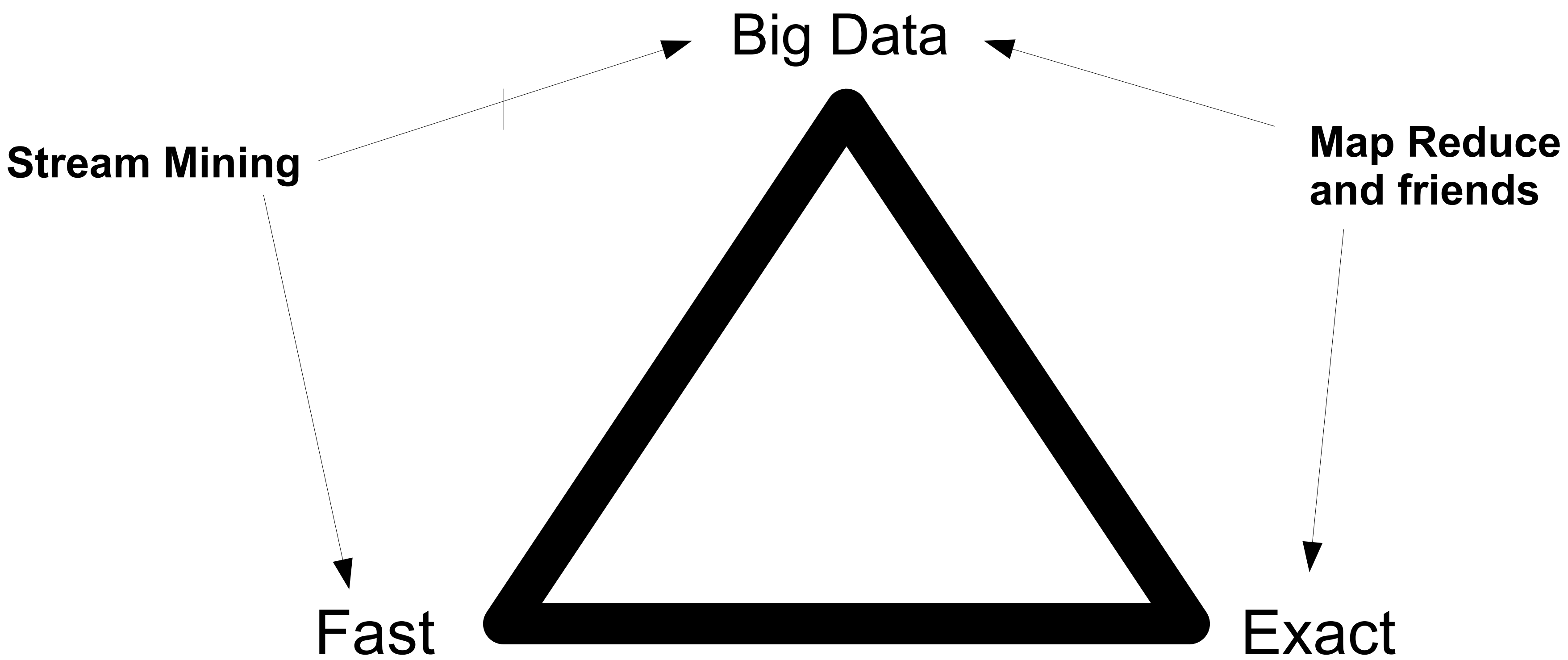


Stream Mining to the rescue

- Stream mining algorithms:
  - answer “stream queries” with finite resources
- Typical examples:
  - **how often** does an item appear in a stream?
  - **how many distinct** elements are in the stream?
  - what are the **top-k most frequent** items?



The Trade-Off



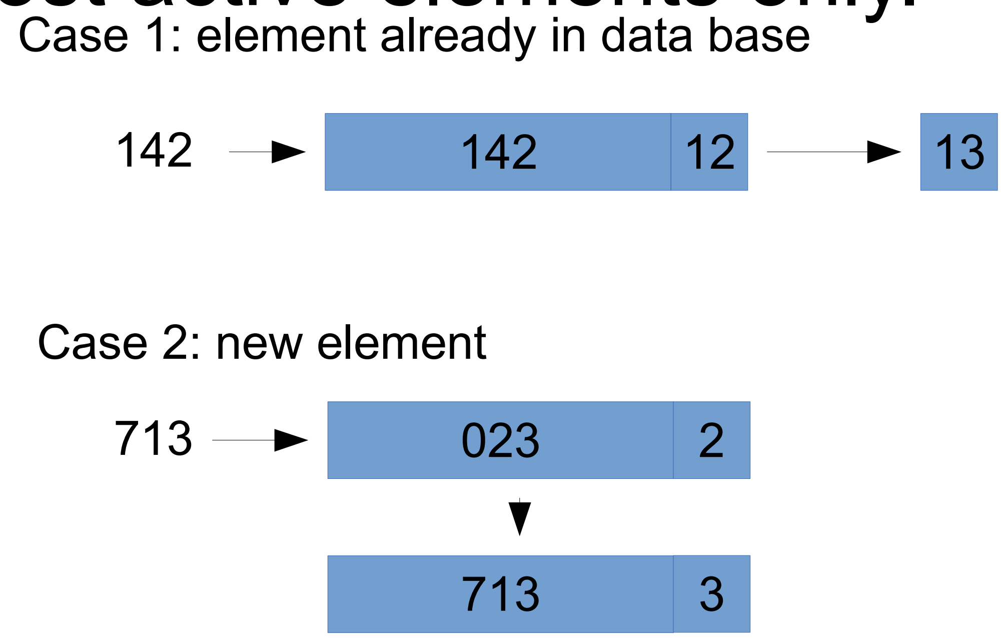
First seen here: http://www.slideshare.net/acunu/realtime-analytics-with-apache-cassandra

Heavy Hitters (a.k.a. Top-k)

- Count activities over large item sets (millions, even more, e.g. IP addresses, Twitter users)
- Interested in most active elements only.

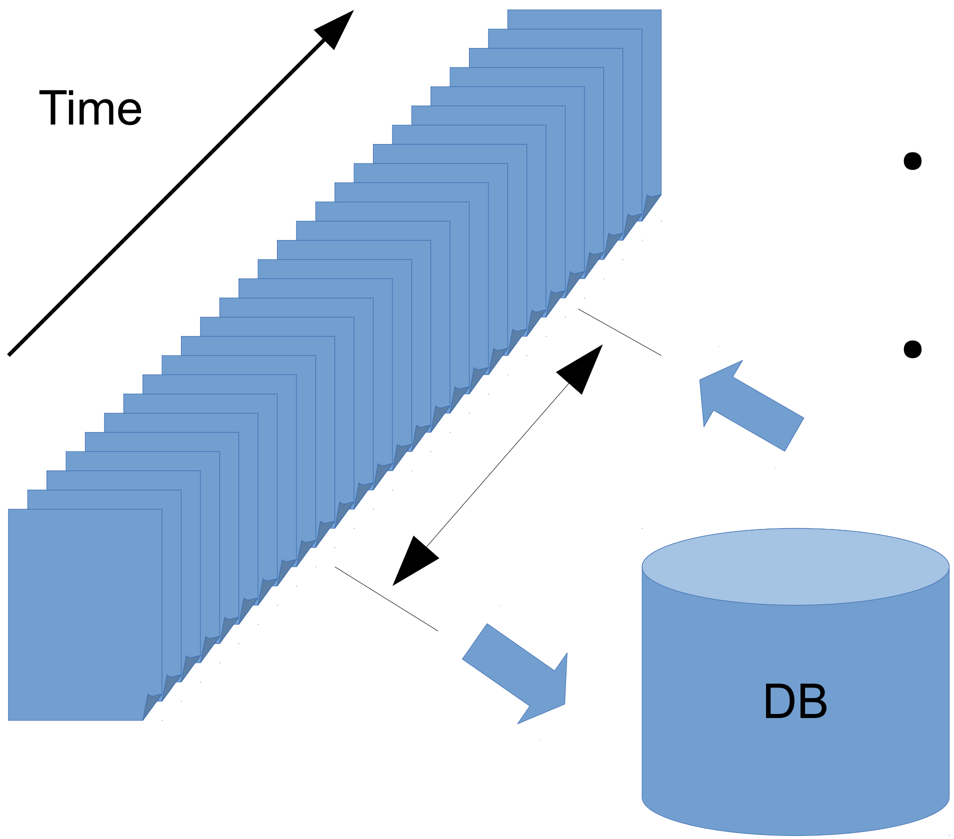
132	15
142	12
432	8
553	5
712	3
023	2

Fixed tables of counts

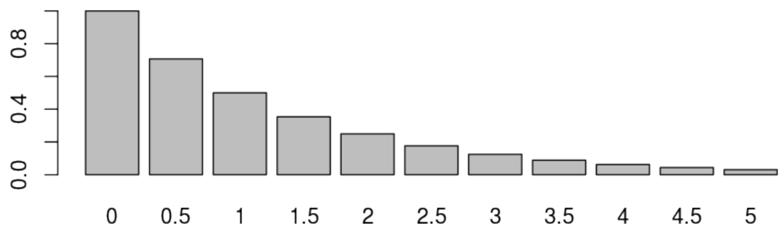


Metwally, Agrawal, Abbadi, *Efficient computation of Frequent and Top-k Elements in Data Streams, Internation Conference on Database Theory, 2005*

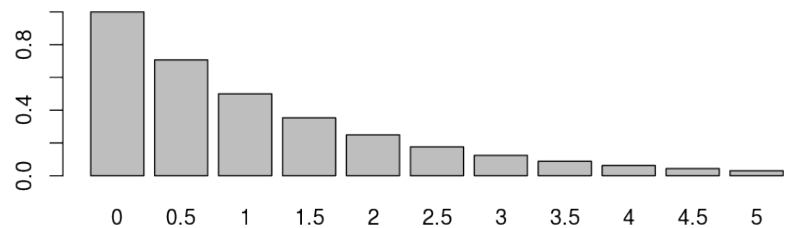
Heavy Hitters over Time-Window



- Keep quite a big log (a month?)
- Constant write/erase in database
- Alternative: Exponential decay



Exponential Decay



- Instead of a fixed window, use exponential decay

$$s(n) = \sum_{i=1}^{n} 2^{-\frac{t_n - t_i}{h}} c_i$$

timestamp

score

halftime

- The beauty: updates are recursive

$$s(n+1) = c_{n+1} + 2^{-\frac{t_{n+1} - t_n}{h}} s(n)$$

time shift term $w(t_{n+1}, t_n)$

Exponential Decay

- Collect stats by a table of expdecay counters

```
counters[item]          # counters
ts[item]                # last timestamp
```

- **update(C, item, timestamp, count)** – update counts

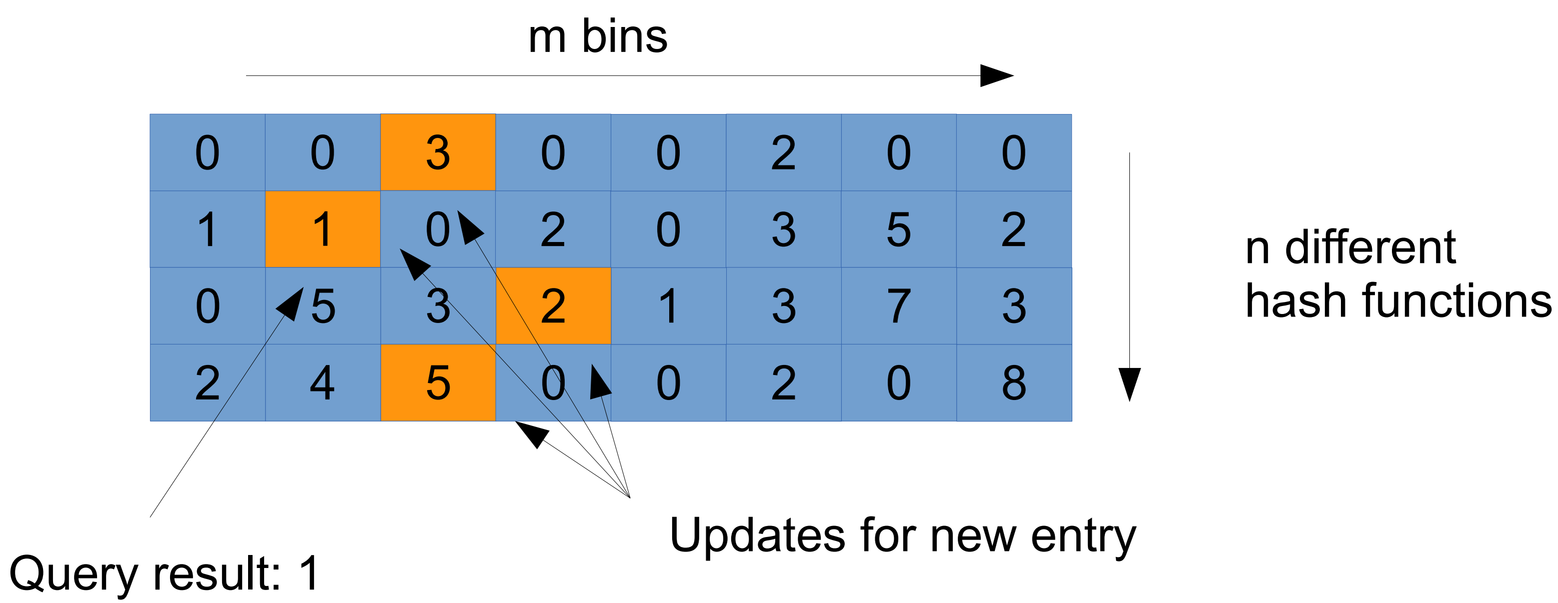
```
C.counters[item] = count +
       weight(timestamp, C.ts[item]) * C.counters[item]
C.ts[item] = timestamp
C.lastupdate = timestamp
```

- **score(C, item)** – return score

```
return weight(C.lastupdate, C.ts[item])
         * C.counters[item]
```

Count-Min Sketches

- Summarize histograms over large feature sets
- Like bloom filters, but better



- Query: Take minimum over all hash functions

G. Cormode and S. Muthukrishnan. *An improved data stream summary: The count-min sketch and its applications.* LATIN 2004, J. Algorithm 55(1): 58-75 (2005) .

Wait a minute? Only Counting?

- Well, getting the top most active items is already useful.
  - Web analytics, Users, Trending Topics
- Counting is statistics!

Counting is Statistics

- Empirical mean:

$$\hat{\mu} = \frac{1}{n} \boxed{\sum_{i=1}^{n}} X_i$$

- Correlations:

$$r = \frac{\boxed{\sum_{i=1}^{n}} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\boxed{\sum_{i=1}^{n}} (X_i - \bar{X})^2} \boxed{\sqrt{\sum_{i=1}^{n}}} (Y_i - \bar{Y})^2}$$

- Principal Component Analysis:

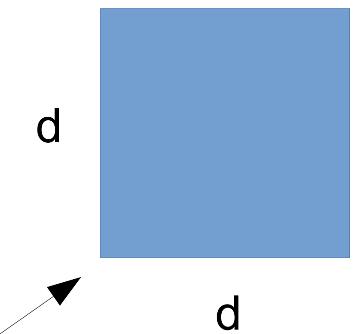
$$C_{ij} = \frac{1}{n} \boxed{\sum_{k=1}^{n}} X_{ik} X_{jk}$$

Example 1: Least Squares Regression

Idea: Batch method like least squares on recent portion of the data.

$$\hat{\beta} = (X^t X)^{-1} X^t Y$$

this could be huge!



with entries $[X^t X]_{ij} = \boxed{\sum_{k=1}^{n} X_{ki} X_{kj}}$

d x d is probably ok

But: It's just a sum!

Least Squares Regression

- Need to compute $\hat{\beta} = (X^t X)^{-1} X^t Y$
- For each X_i, Y_i do
  - $\mathrm{update}(C, (i,j), t, X_i X_j)$
  - $\mathrm{update}(S, i, t, X_i Y)$
- Then, reconstruct
  - $\hat{\beta} = C^{-1} S$

As a reminder:

$$[X^t X]_{ij} = \sum_{k=1}^{n} X_{ki} X_{kj}$$

$$[X^t Y] = \sum_{k=1}^{n} X_{ik} Y_k$$

Example 2: Maximum-Likelihood

- Estimate probabilistic models

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} X_i^2 - \left(\frac{1}{n} \sum_{i=1}^{n} X_i \right)^2$$

based on

$$\frac{1}{n} \left(\sum_{i=1}^{n} X_i - \frac{1}{n} \sum_{j=1}^{n} X_j \right)^2$$

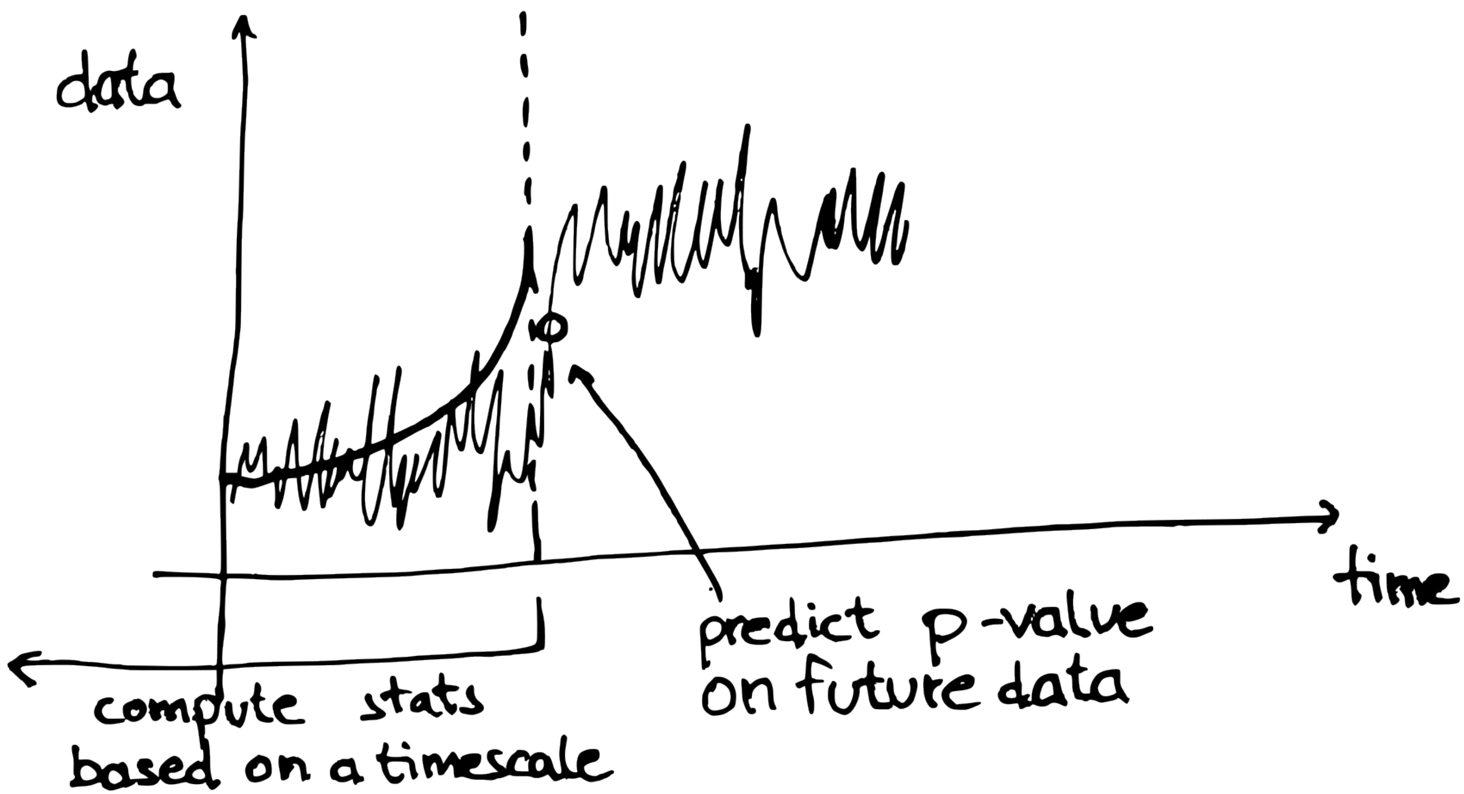
which is slightly biased, but simpler

- But wait, how do I "1/n" with randomly spaced events?

$$\hat{n} = \sum w(t_i)$$

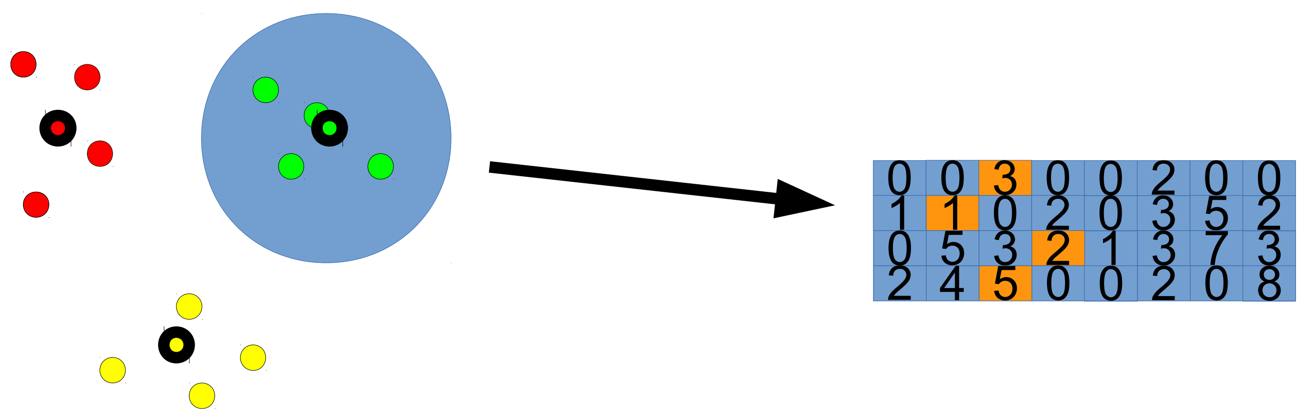
Example 3: Outlier detection

- Once you have a model, you can compute p-values (based on recent time frames!)



Example 4: Clustering

- Online clustering
  - For each data point:
    - Map to closest centroid (⇒ compute distances)
    - Update centroid
  - count-min sketches to represent sum over all vectors in a class



Aggarwal, *A Framework for Clustering Massive-Domain Data Streams, IEEE International Conference on Data Engineering , 2009*

Example 5: TF-IDF

- estimate word – document frequencies

$$DF(w) = \#\{\text{documents which contain word } w\}$$

- for each word: update(word, t, 1.0)
- for each document: update(“#docs”, t, 1.0)
- query: score(word) / score(“#docs”)

Example 6: Classification with Naïve Bayes

- Naive Bayes is also just counting, right?

frequency of word i in document

Number of times word i appears in class c

$$\ell_{MNB}(d) = \operatorname{argmax}_c \left[\log \hat{p}(\theta_c) + \sum_i f_i \log \frac{N_{ci} + \alpha_i}{N_c + \alpha} \right]$$

class priors

Priors

Multinomnial naïve Bayes

Total number of words in class c

Example 6: Classification with Naive Bayes

Tackling the Poor Assumptions of Naive Bayes Text Classifiers

Jason D. M. Rennie JRENNIE@MIT.EDU
Lawrence Shih KAI@MIT.EDU
Jaime Teevan TEEVAN@MIT.EDU
David R. Karger KARGER@MIT.EDU
Artificial Intelligence Laboratory; Massachusetts Institute of Technology; Cambridge, MA 02139

Abstract

Naive Bayes is often used as a baseline in text classification because it is fast and easy to implement. Its severe assumptions make such efficiency possible but also adversely affect the quality of its results. In this paper we propose simple, heuristic solutions to some of the problems with Naive Bayes classifiers, addressing both systemic issues as well as problems with ... amples. To balance the amount of training examples used per estimate, we introduce a "complement class" formulation of Naive Bayes.

Another systemic problem with Naive Bayes is that features are assumed to be independent. As a result, even when words are dependent, each word contributes evidence individually. Thus the magnitude of the weights for classes with strong word dependencies is larger than for classes with weak word dependencies.

ICML 2003

Example 6: Classification with Naive Bayes

- 7 Steps to improve NB:
  - transform TF to log(. + 1)
  - IDF-style normalization
  - square length normalization
  - use complement probability
  - another log
  - normalize those weights again
  - Predict linearly using those weights

What about non-parametric methods and Kernel Methods?

- Problem here, no real accumulation of information in statistics, e.g. SVMs

$$\hat{f}(x) = \sum_{i=1}^{n} k(X_i, X_j)\alpha_i + \alpha_0$$

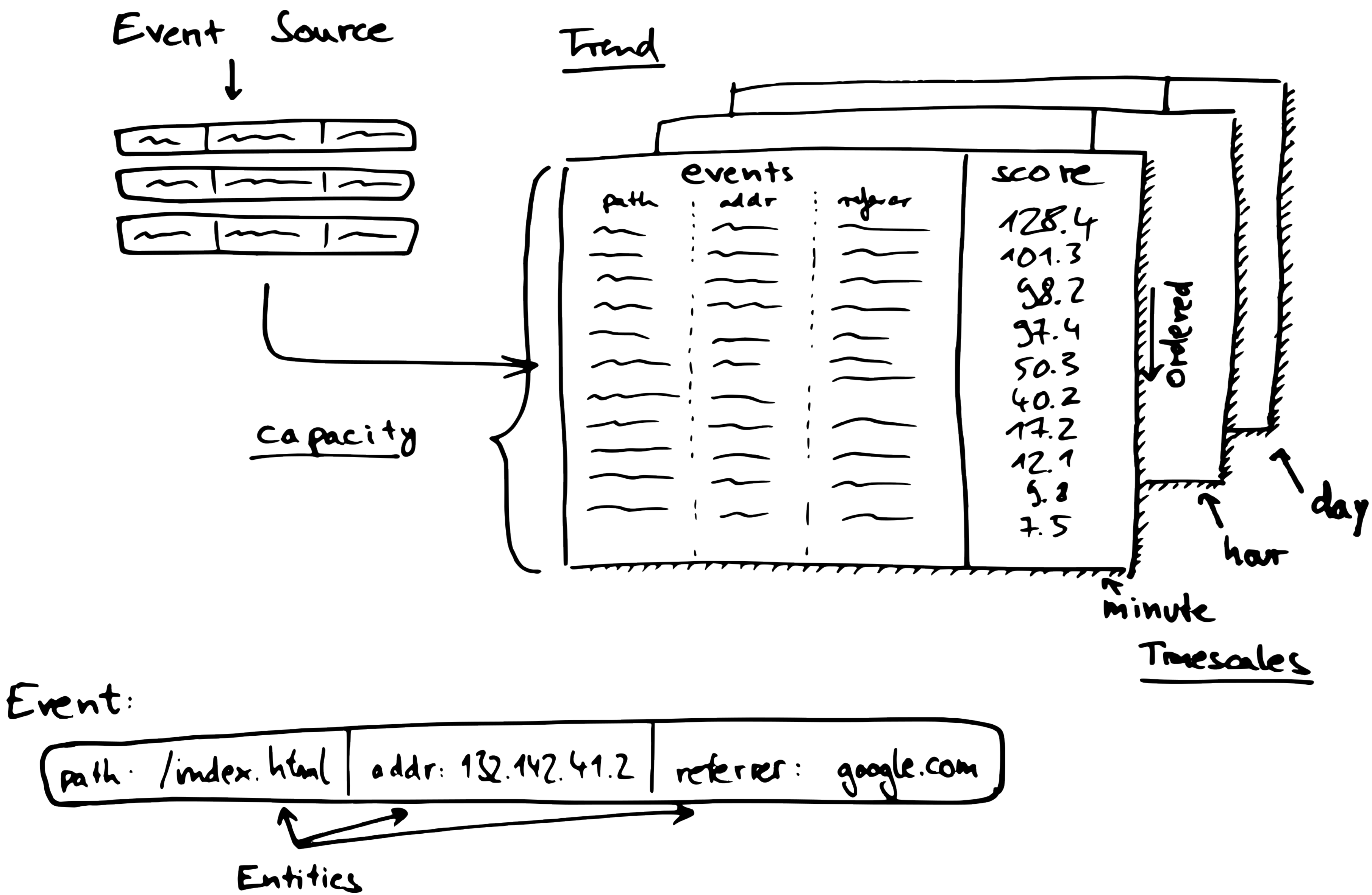
sum over all n^2 elements!

- Could still use streamdrill to extract a representative subset.

Streamdrill

- Heavy Hitters counting + exponential decay
- **Instant** counts & top-k results over time windows.
- Indices!
- Snapshots for historical analysis
- Beta demo available at http://streamdrill.com, launch imminent

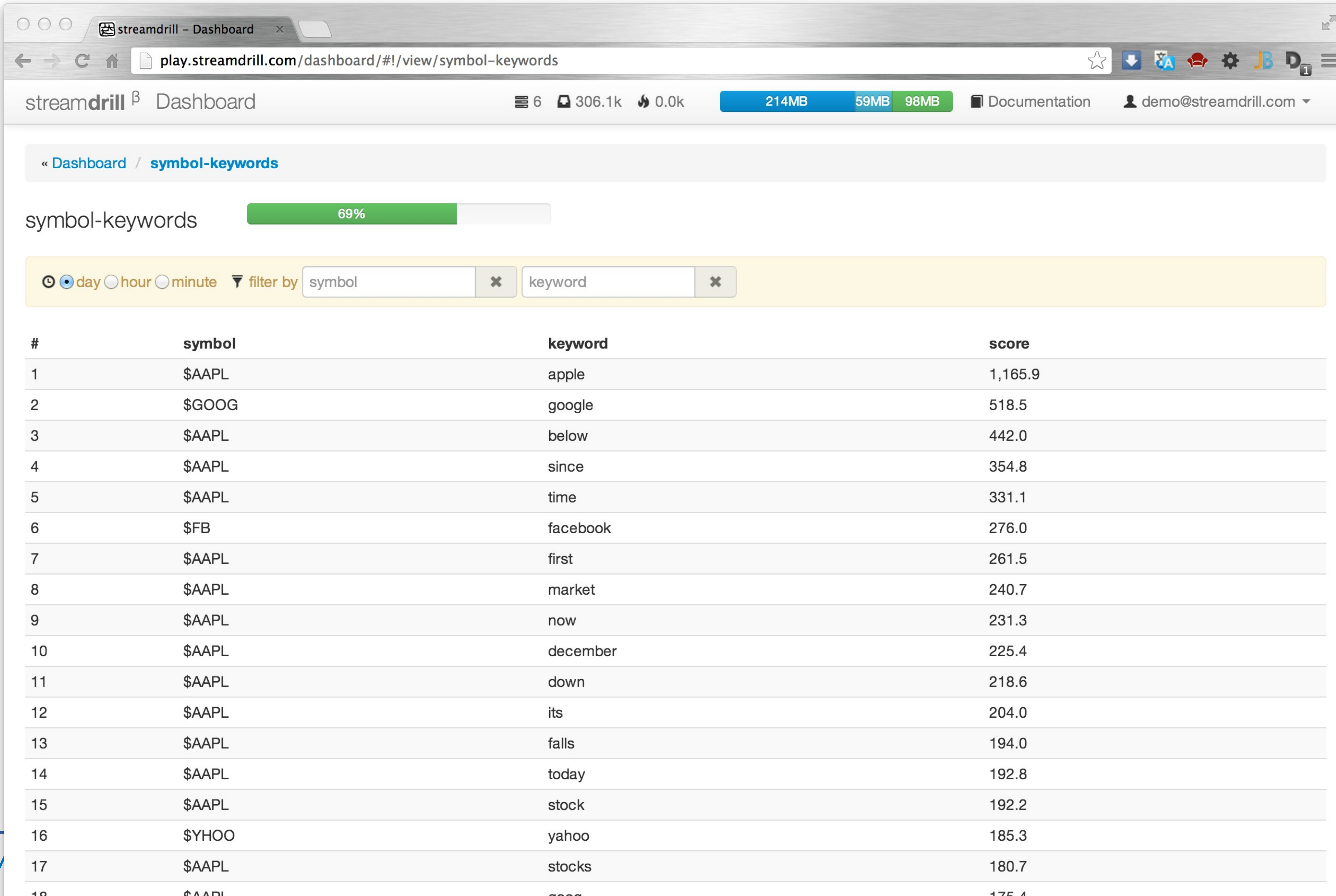
Architecture Overview



Example: Twitter Stock Analysis

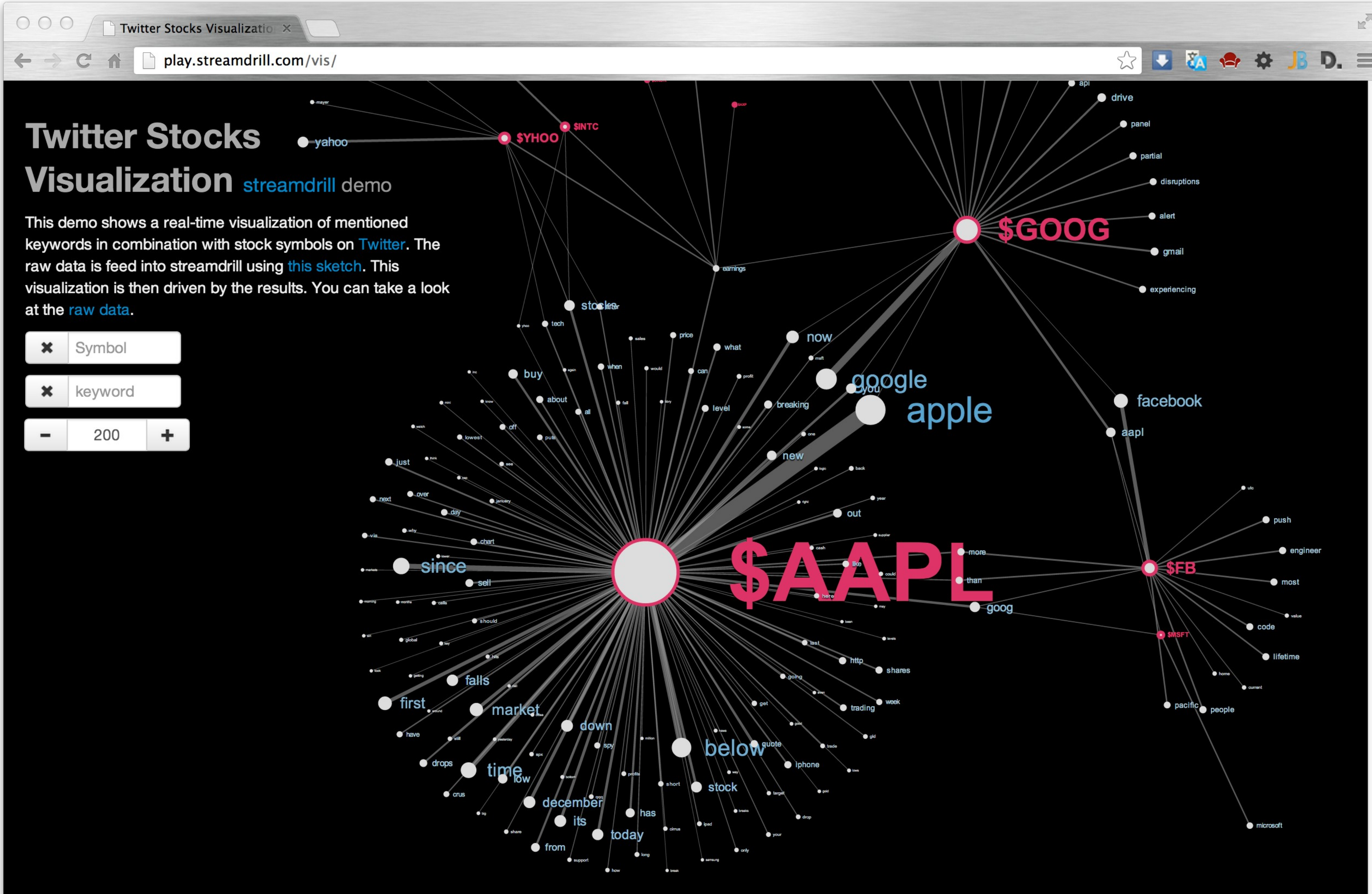
http://play.streamdrill.com/vis/

Example: Twitter Stock Analysis



#	symbol	keyword	score
1	$AAPL	apple	1,165.9
2	$GOOG	google	518.5
3	$AAPL	below	442.0
4	$AAPL	since	354.8
5	$AAPL	time	331.1
6	$FB	facebook	276.0
7	$AAPL	first	261.5
8	$AAPL	market	240.7
9	$AAPL	now	231.3
10	$AAPL	december	225.4
11	$AAPL	down	218.6
12	$AAPL	its	204.0
13	$AAPL	falls	194.0
14	$AAPL	today	192.8
15	$AAPL	stock	192.2
16	$YHOO	yahoo	185.3
17	$AAPL	stocks	180.7

Example: Twitter Stock Analysis



Summary

- Doesn't always have to be scaling!
- **Stream mining**: Approximate results with finite resources.
- stream**drill:** stream analysis engine