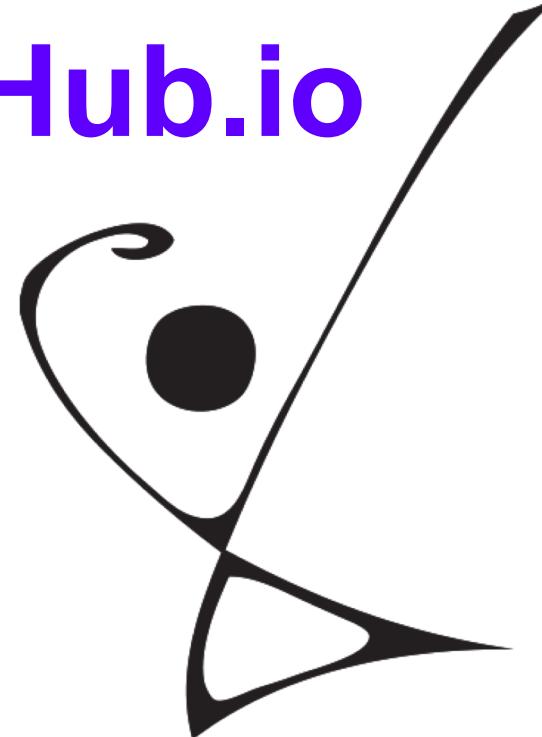


# DataSketches.GitHub.io

A Required Toolkit for the  
Analysis of Big Data



Lee Rhodes  
9 July 2018

# Who are we? -- Deep Science, Exceptional Engineering

## Core Team

- Lee Rhodes, Oath/Yahoo, Distinguished Architect. Started internal project 2011.
- Kevin Lang, Ph.D., Oath/Yahoo, Chief Scientist. Joined 2012.
- Alex Saydakov, Sr. Systems Developer, Oath/Yahoo. Joined 2015.
- Jon Malkin, Ph.D., Oath/Yahoo, Scientist. Joined 2016.
- Edo Liberty, Ph.D., Principal Scientist at Amazon Web Services and Head of Amazon AI Labs. Joined 2015.
- Justin Thaler, Ph.D., Assistant Professor, Georgetown University, Computer Science. Joined 2015.
- Christopher Musco, Ph.D. MIT→NYU, Theoretical Computer Science, Lin Algebra. Joined 2018.
- Mina Ghashami, Ph.D., Research Scientist, Visa, Inc. Joined 2018.

## Extended Team

- Graham Cormode, Ph.D., Professor, University of Warwick, UK, Computer Science. Joined 2017.
- Srikanta Tirthapura, Ph.D., Professor, Iowa State University, Elec. & Computer Science. Joined 2017.
- Pryce Bevan, Daniel Anderson, Ph.D. Students, Georgetown Univ. Joined 2017.

# The Data Analysis Challenges

Example: Web Site Logs

Time Stamp	User ID	Device ID	Site	Time Spent Sec	Items Viewed
9:00 AM	U1	D1	Apps	59	5
9:30 AM	U2	D2	Apps	179	15
10:00 AM	U3	D3	Music	29	3
1:00 PM	U1	D4	Music	89	10

Billions of K,V Pairs ...

Example: ML Training Data

ID	Attr 1	...	Attr N
U1	V <sub>1,1</sub>	...	V <sub>1,N</sub>
	:	:	:
U <sub>M</sub>	V <sub>M,1</sub>	...	V <sub>M,N</sub>

*Millions of Rows ...  
Thousands of Columns*

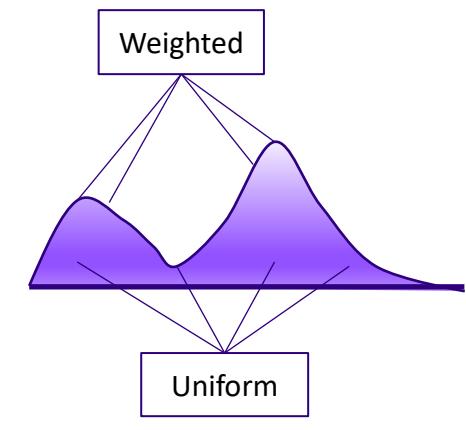
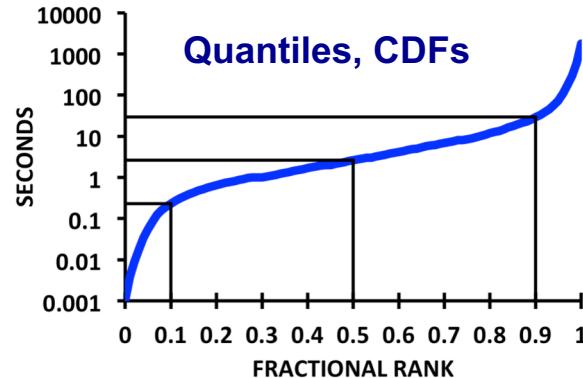
In Near-Real Time.

*Fast!*

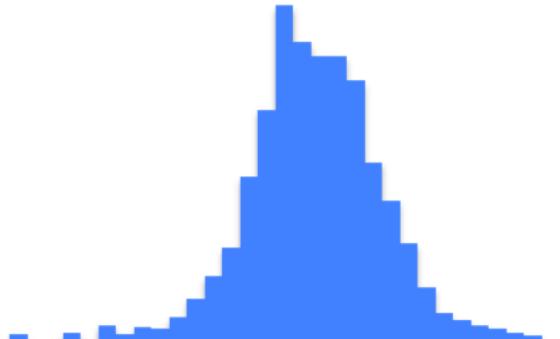
# Some Very Common Queries ...



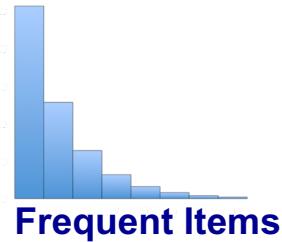
Counting Unique  
Identifiers



Reservoir Sampling



Histograms, PMFs, CDFs

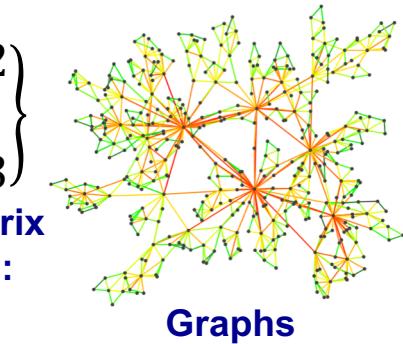


Frequent Items



$\left\{ \begin{matrix} 5 & \dots & 2 \\ \vdots & \ddots & \vdots \\ 4 & \dots & 3 \end{matrix} \right\}$

Vector & Matrix  
Operations:  
SVD, etc.



Graphs

... All Are Computationally Difficult

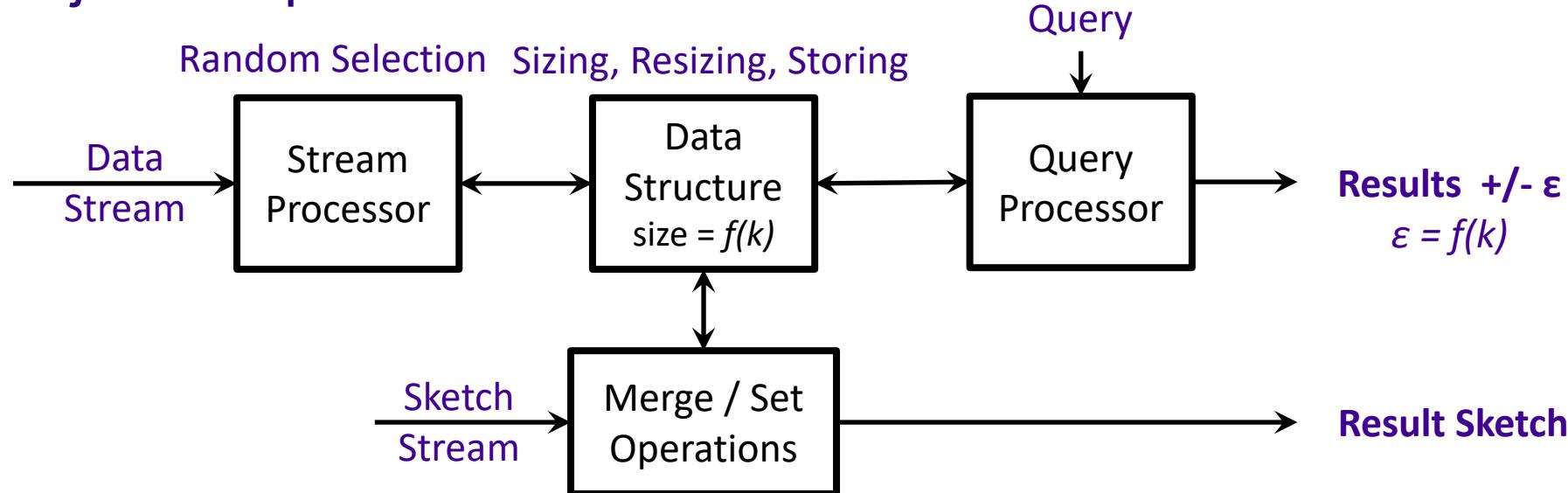
# Our Mission...

Develop **Production Quality**

Stochastic Streaming Algorithms a.k.a. **Sketches**

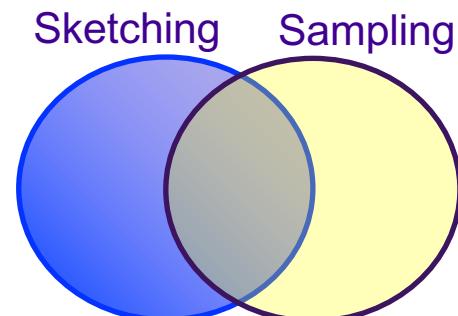
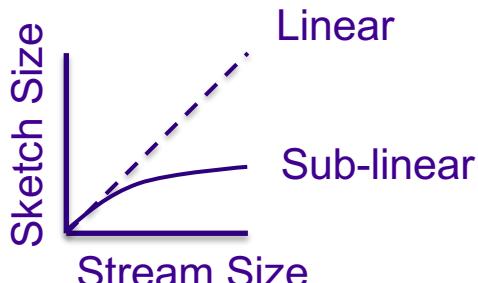
to Address these Difficult Queries

# Major Components of a Sketch



## 5 Major Characteristics

- Small Size, Sub-linear in Space →
- Single-pass
- Mergeable
- Approximate, Probabilistic
- Mathematically Proven Error Bounds



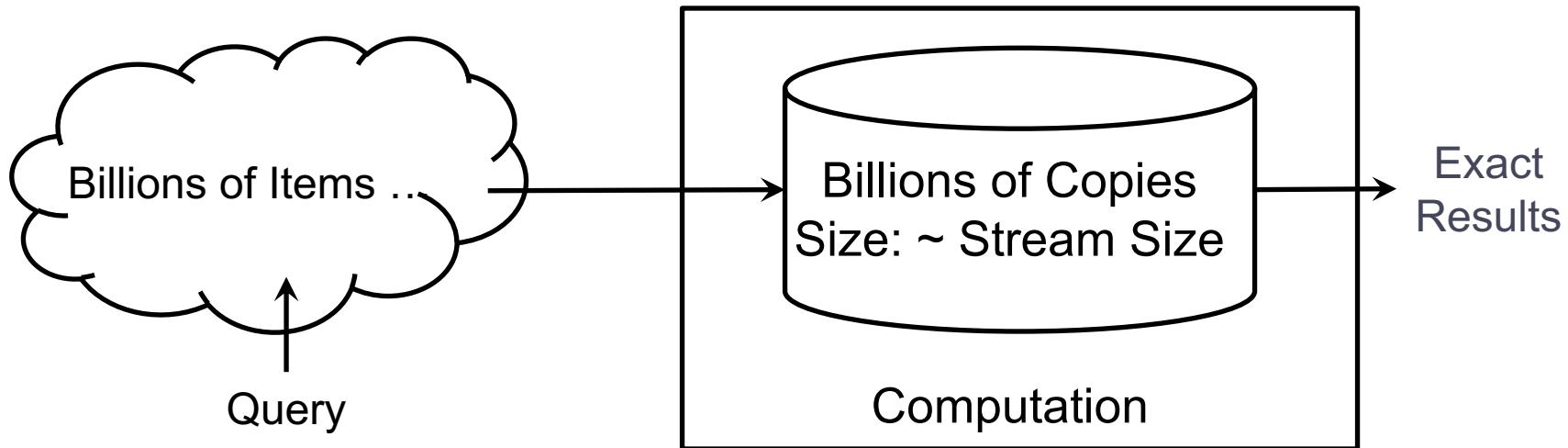
# What Does “Production Quality” Mean?

- World-Class Performance for Large Systems
  - Best accuracy, space utilization and fastest performance, of anything out there!
  - Specifically designed for large systems
- Robust
  - Extensive Unit-tests with > 90% code coverage
  - Comprehensive Accuracy and Speed Characterization Studies, most available on-line, with code!
  - “Battle-hardened” – Used extensively in our back-end and reporting systems
- Easy to Use – Designed with the Systems Developer in mind!
  - Comprehensive API, Javadocs, Web-site documentation & examples.
  - Minimal External Dependencies – The “core” library has Zero dependencies!
  - Mergeable with different size-accuracy parameters: (e.g.,  $k$ ) -- Change your mind over time!
  - Operations on Stored Sketches are Backward Compatible – From 2012!
  - Specific features targeted for large, distributed systems. (Off-heap, p-sampling modes, etc.)
  - Command-line application for experimentation.

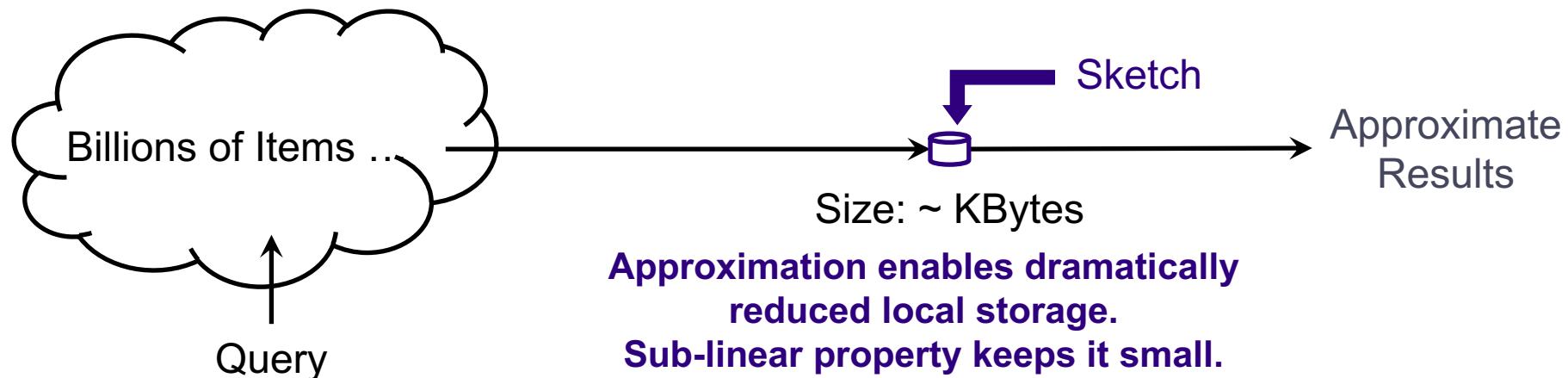
# Systems Architecture Using Sketches

How do we use Sketches, and Why?

# Exact Distinct Counting, Single Machine



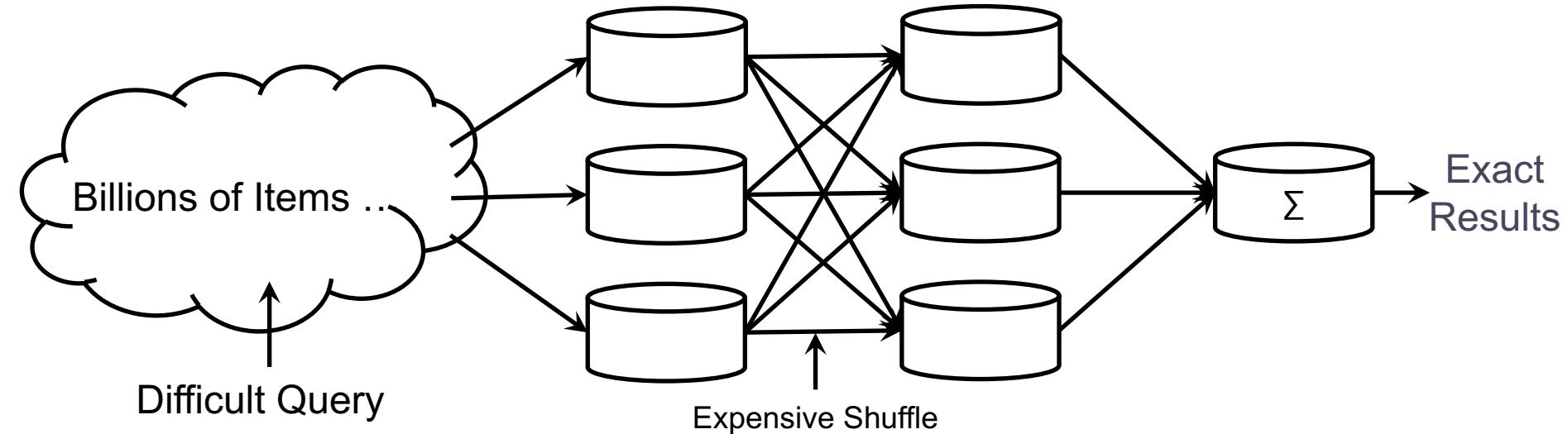
# Sketch Distinct Counting, Single Machine



# Exact Distinct Counting, Map Reduce

Each Partition:

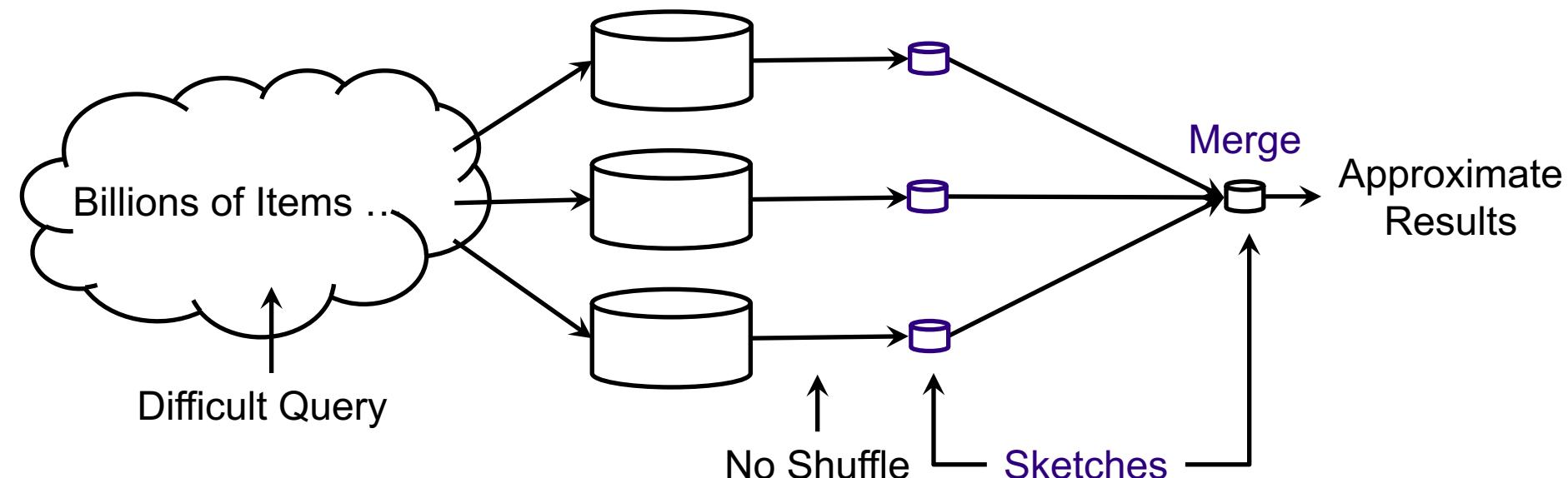
- Must Be Mutually Distinct WRT Cardinality
- Performs Traditional Count Distinct



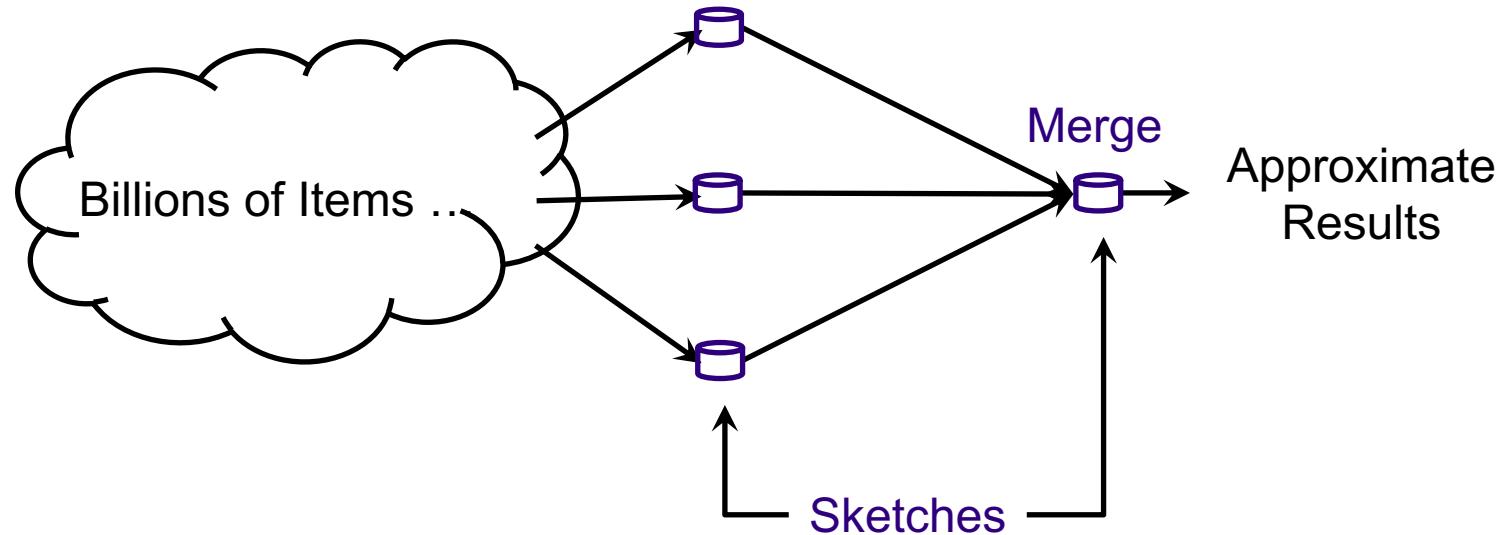
*Parallel Processing Doesn't Help Much.  
You have to keep the copies somewhere!*

# Sketch Distinct Counting, Multiple Machines

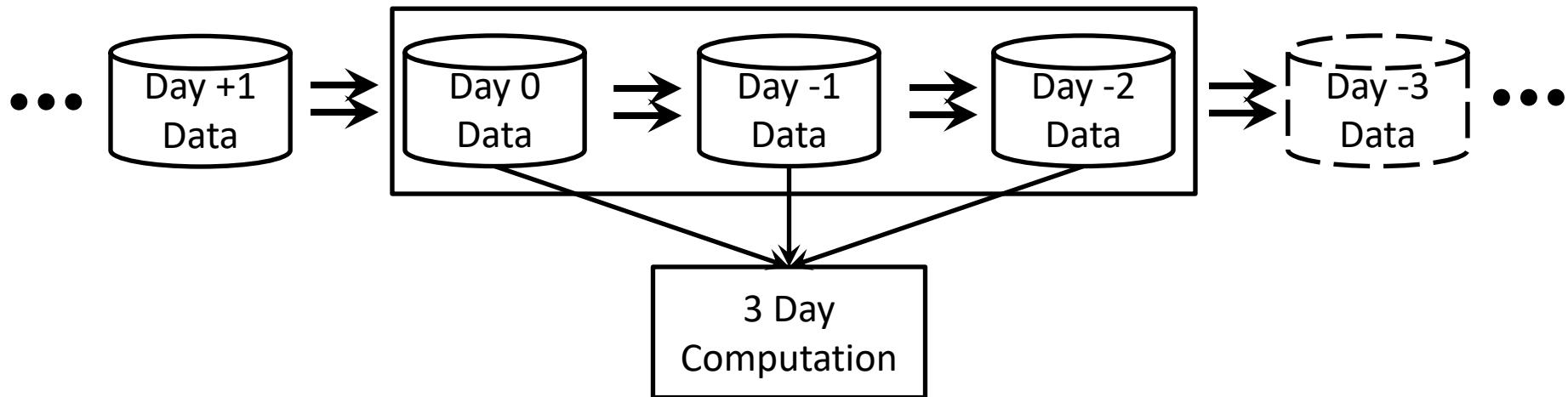
Mergeability enables significant speedup.



# Sketch Distinct Counting, Real Time

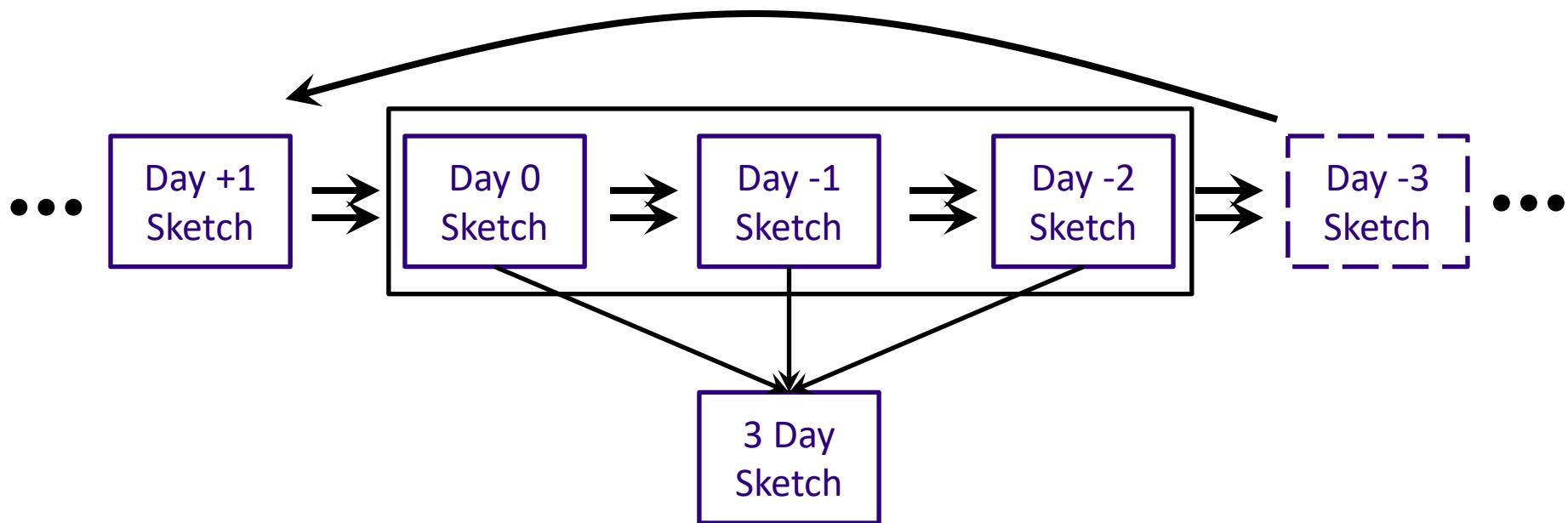


# Traditional Time Windowing



Every dataset is processed 3 times for a rolling 3 day window

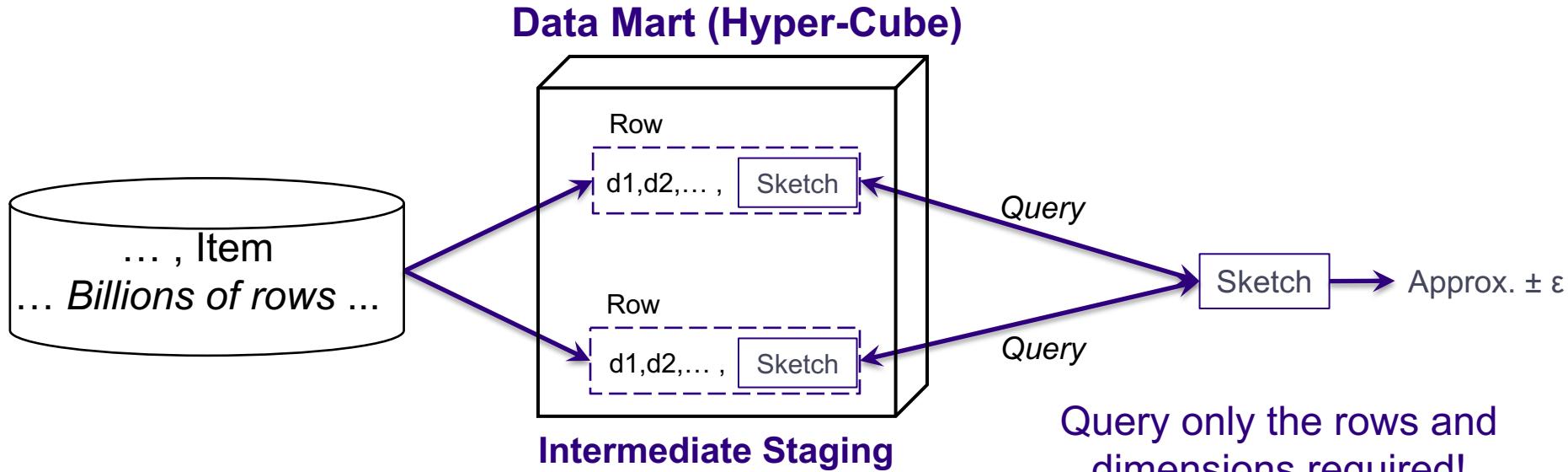
# Time Windowing with Sketches



Every data item is processed once for a rolling N day window.  
Late-data processing is now possible.

# Hyper-Cube Provides Fast, Simple Architecture

Intermediate Sketch Staging Enables Query Speed & Simpler Architecture



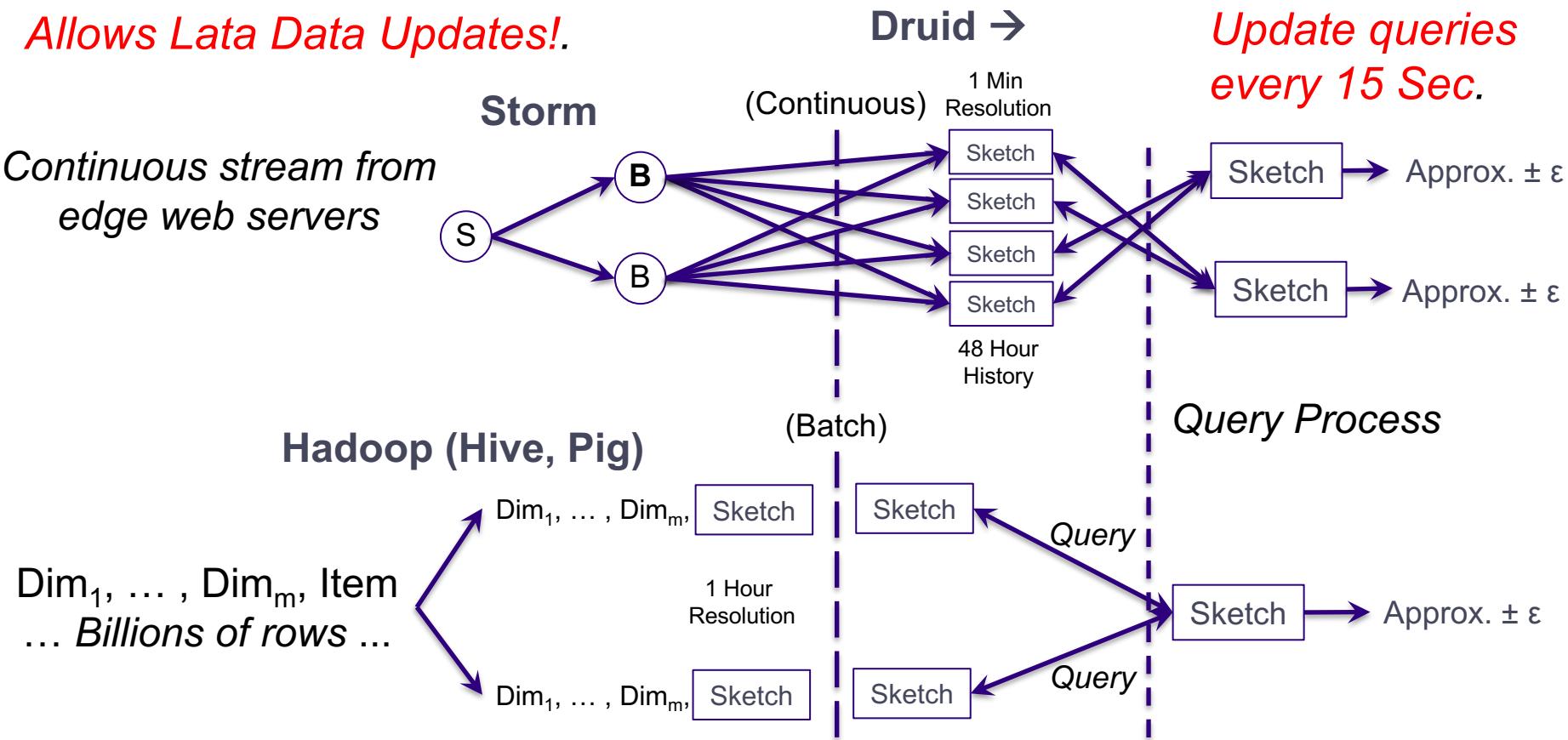
Stored Sketches Can Be Merged  
By Any Dimensions, Including Time!

Query only the rows and dimensions required!

# Combine Off-line, On-line for Real-time

## Case Study: Flurry/Druid Sketch Flow Architecture

*Allows Late Data Updates!*



# Case Study: Flurry, Before and After

- Customers: >250K Mobile App Developers
- Data: 40-50 TB per day
- Platform: 2 clusters X 80 Nodes = 160 Nodes
  - Node: 24 CPUs, 250GB RAM

Big Wins!  
Near-Real Time  
Lower System \$

	Before Sketches	After Sketches
VCS* / Mo.	~80B	~20B
Result Freshness	Daily: 2 to 8 hours; Weekly: ~3 days <b>Real-time Unique Counts Not Feasible</b>	<b>15 seconds!</b>

\* VCS: Virtual Core Seconds

# Advantages of Sketch-based System Design

- Sketches enable near-real time performance that would otherwise be infeasible with massive data and exact methods – no matter how much hardware you throw at it.
- Architectural simplicity: Non-additive data becomes “additive”
  - Fewer processing steps
  - Fewer intermediate tables
- Enables reporting on arbitrary dimension combinations
  - Fast merging
  - Simple Time-Windowing

# Major Sketch Families in DataSketches Library

## Cardinality, 3 Families

- Theta Sketches: Set Expressions (e.g., Union, Intersection, Difference), on-heap or off-heap
- HLL Sketches: Highly compact; HLL Map
- Tuple Sketches: Associative Theta Sketches

## Quantiles Sketches

- Quantiles, PMF's and CDF's of streams of comparable objects. On-heap or off-heap
- KLL, (so far, only on-heap doubles), highly optimized for accuracy-space.

## Frequent Items Sketches

- Heavy Hitters from a stream of weighted objects

## Reservoir and VarOpt Sketches

- Uniform and weighted sampling to fixed- $k$  sized buckets

## Sketches for Mobile (Android)

- Quantiles, PMF's and CDF's

## Vector & Matrix Sketches

- Frequent Directions ( Approximate SVD)

# Applications Suitable for Sketches ...

## Items (words, IDs, events, clicks, ...)

- Counting distinct elements
- Item frequencies (Heavy Hitters, etc.)
- Quantiles, Ranks, PMFs, CDFs
- Moment and entropy estimation
- Set Operations
- Sampling

## Graphs<sup>1</sup> (Social networks, communications, ...)

- Connectivity
- Cut sparsification
- Weighted matching
- ...

Vectors (text docs, images, features, ...) AND

## Matrices (text corpora, recommendations, ...)

- Covariance estimation
- Low rank approximation
- Sparsification
- Dimensionality reduction (SVD)
- Clustering (k-means, k-median, ...)
- Linear Regression
- Machine Learning (some of it at least)
- Density estimation

Areas we have implemented

Active areas of research (World-wide)

# <Pause>

“Any sufficiently advanced technology is indistinguishable from magic” – Arthur C. Clarke

DataSketches

Science Innovations

And Results

# Innovations for Unique Counting Sketches

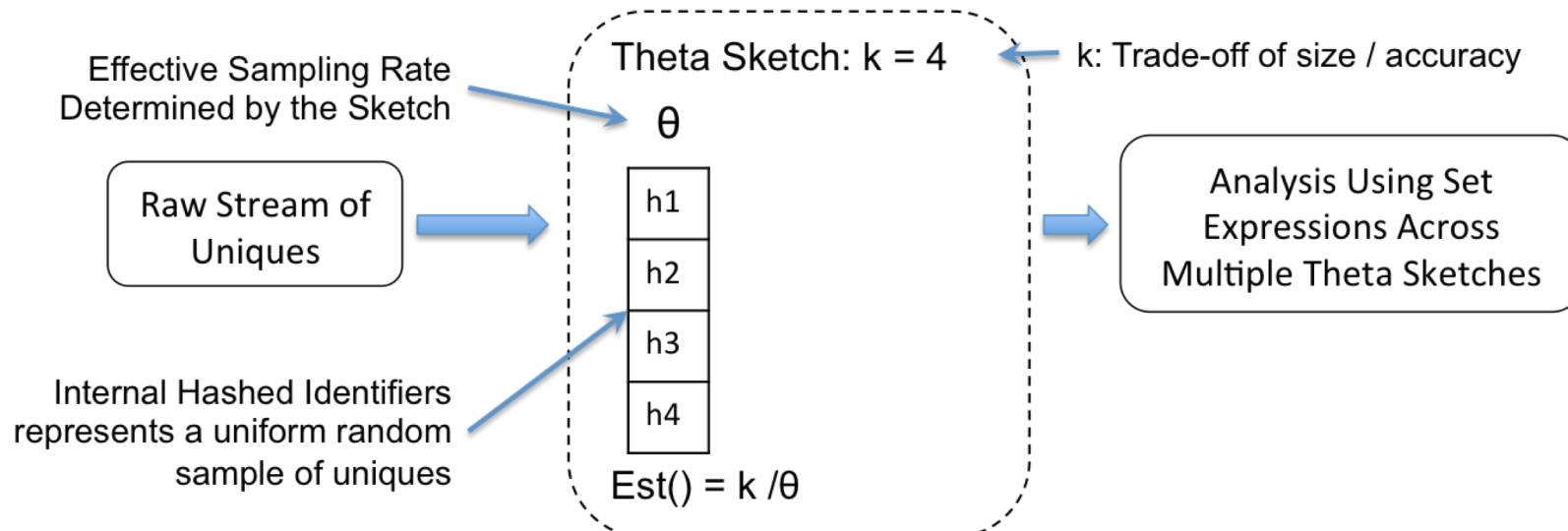
## Theta Sketch Framework (TSF):

A. Dasgupta, K. Lang, L. Rhodes, J. Thaler, A Framework for Estimating Stream Expression Cardinalities, ACM ICDT 2016

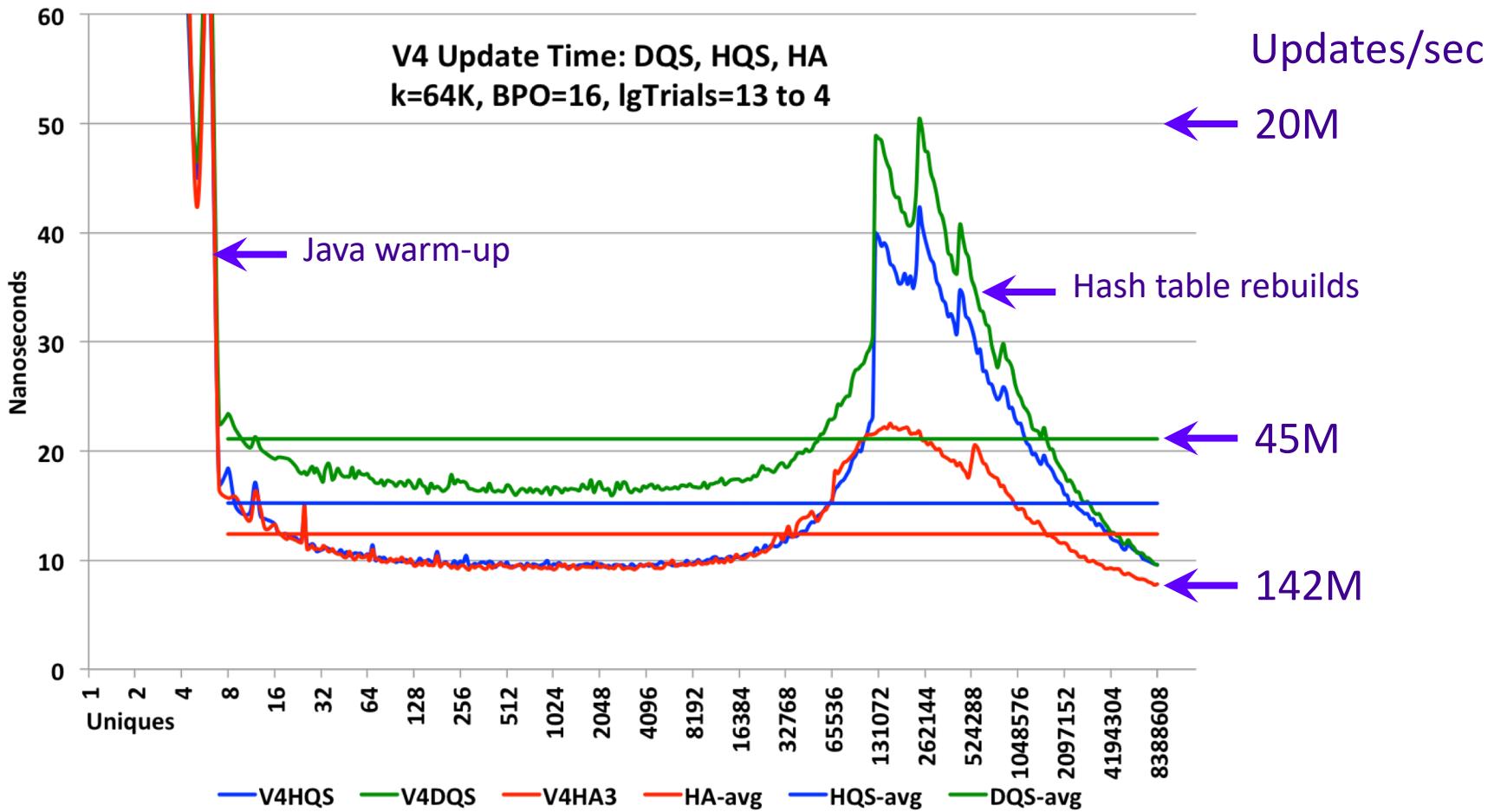
- Builds on Bar-Yossef, et al, 2002 “Counting Distinct Elements...”
- TSF applies to a whole family of sketches
- Enables simple methods for enabling set expressions and multiple-k merging
- Enables trivial up-front, ( $p$ KMV) sampling for tighter space usage in large systems
- Library Theta Sketch Framework:
  - Sketches: UpdateSketch, CompactSketch, AlphaSketch
  - Set Expressions: Union, Intersection, AnotB:  $(A \cup B) \cap (C \cup D) \setminus E$
  - Tuple Sketch (Update Sketch with User-defined attributes)

# Theta Sketch Framework

## Simple Theta Sketch

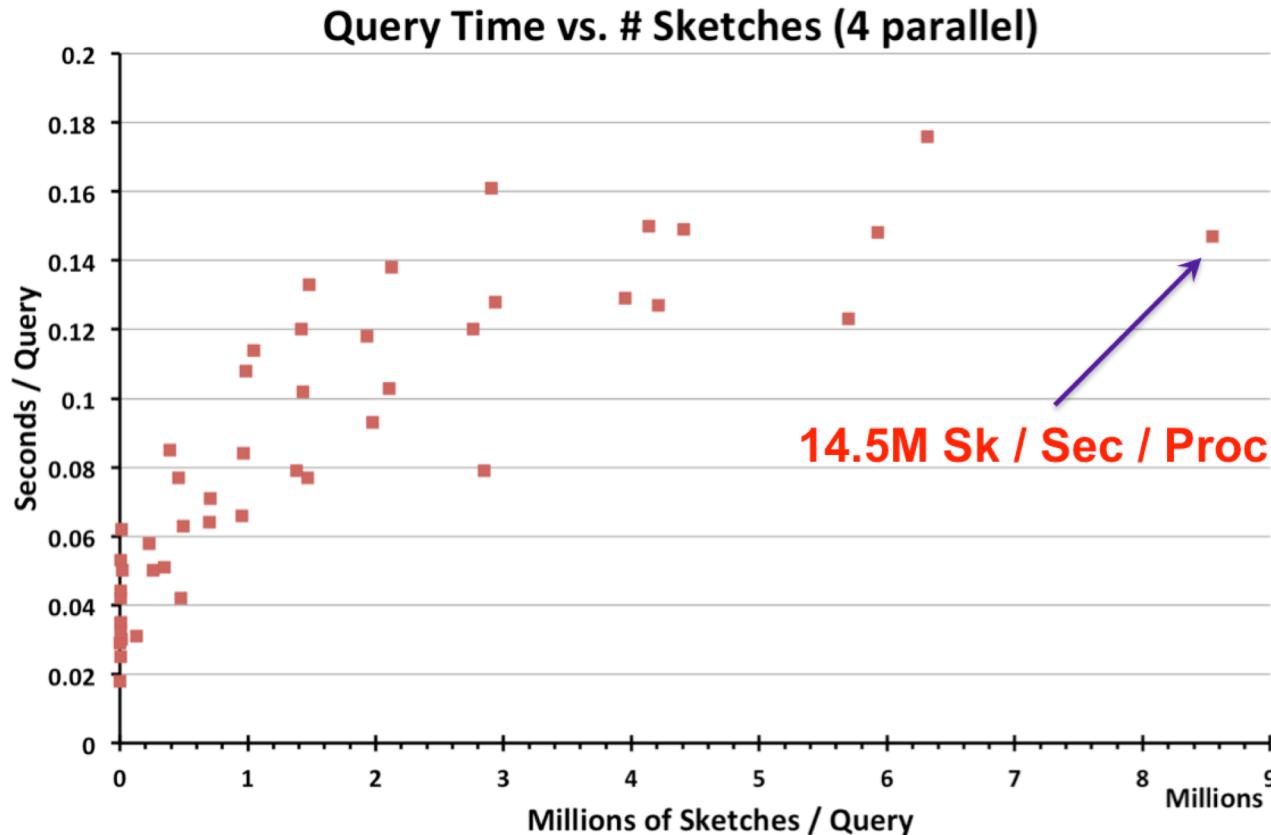


# TSF: Theta Sketch Update Speed, 64K

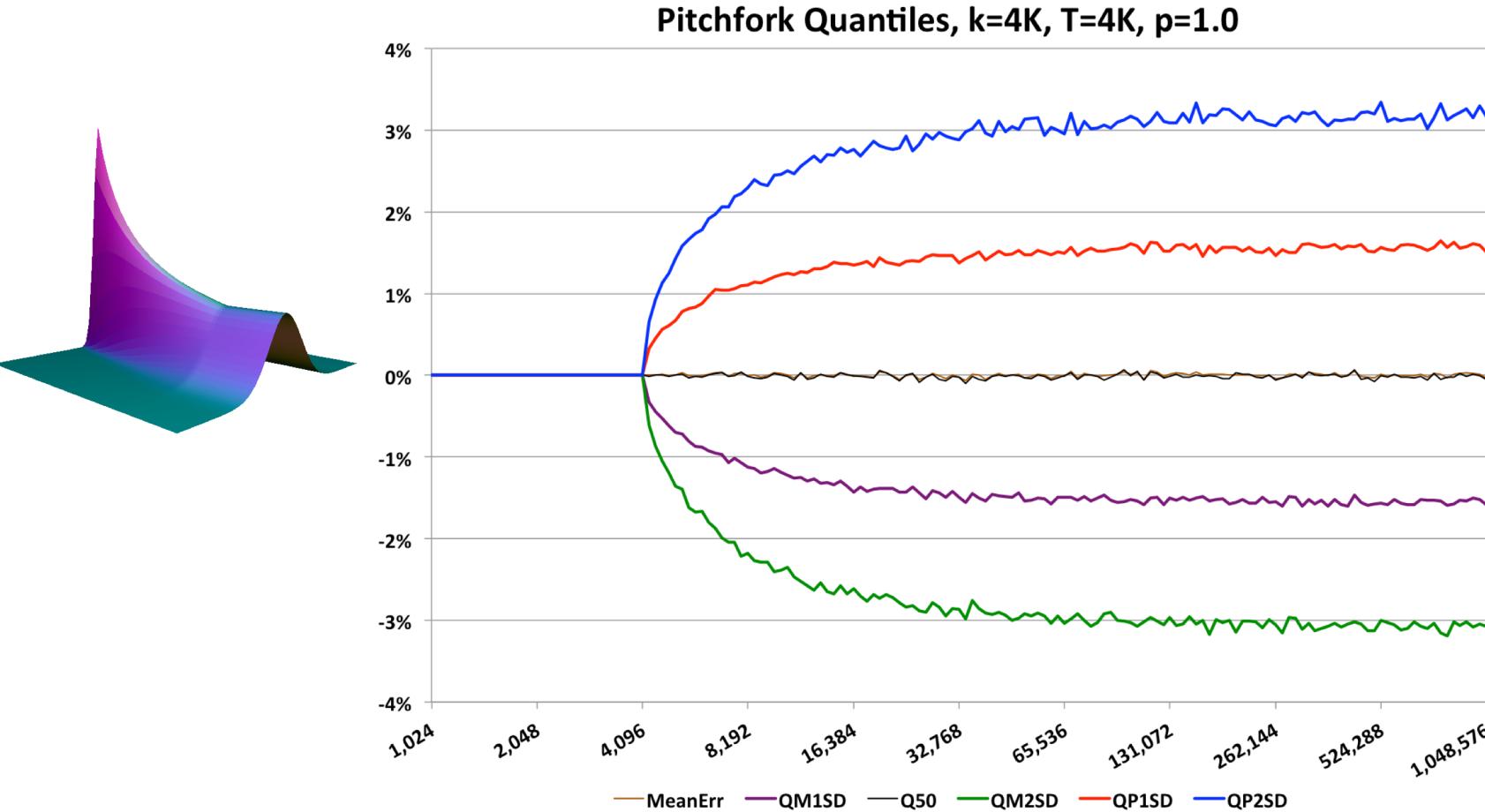


# Theta Sketch Framework

## Sketch Merge Time / Query

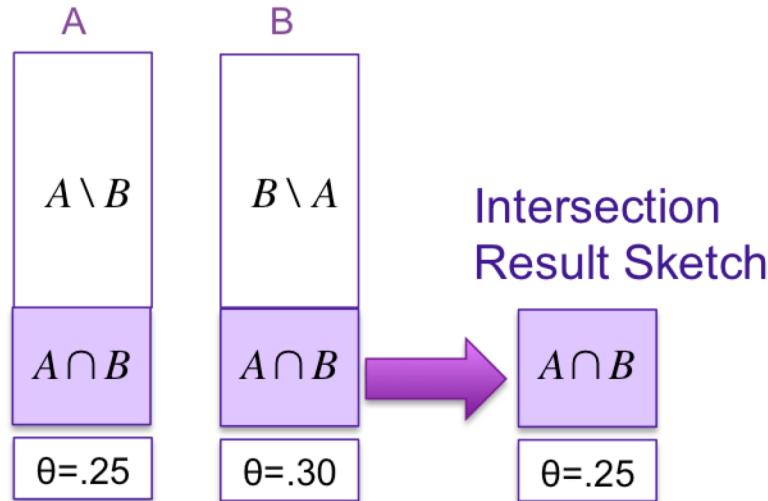


# Theta Sketch Framework: Theta Sketch Accuracy



# Theta Sketch Framework

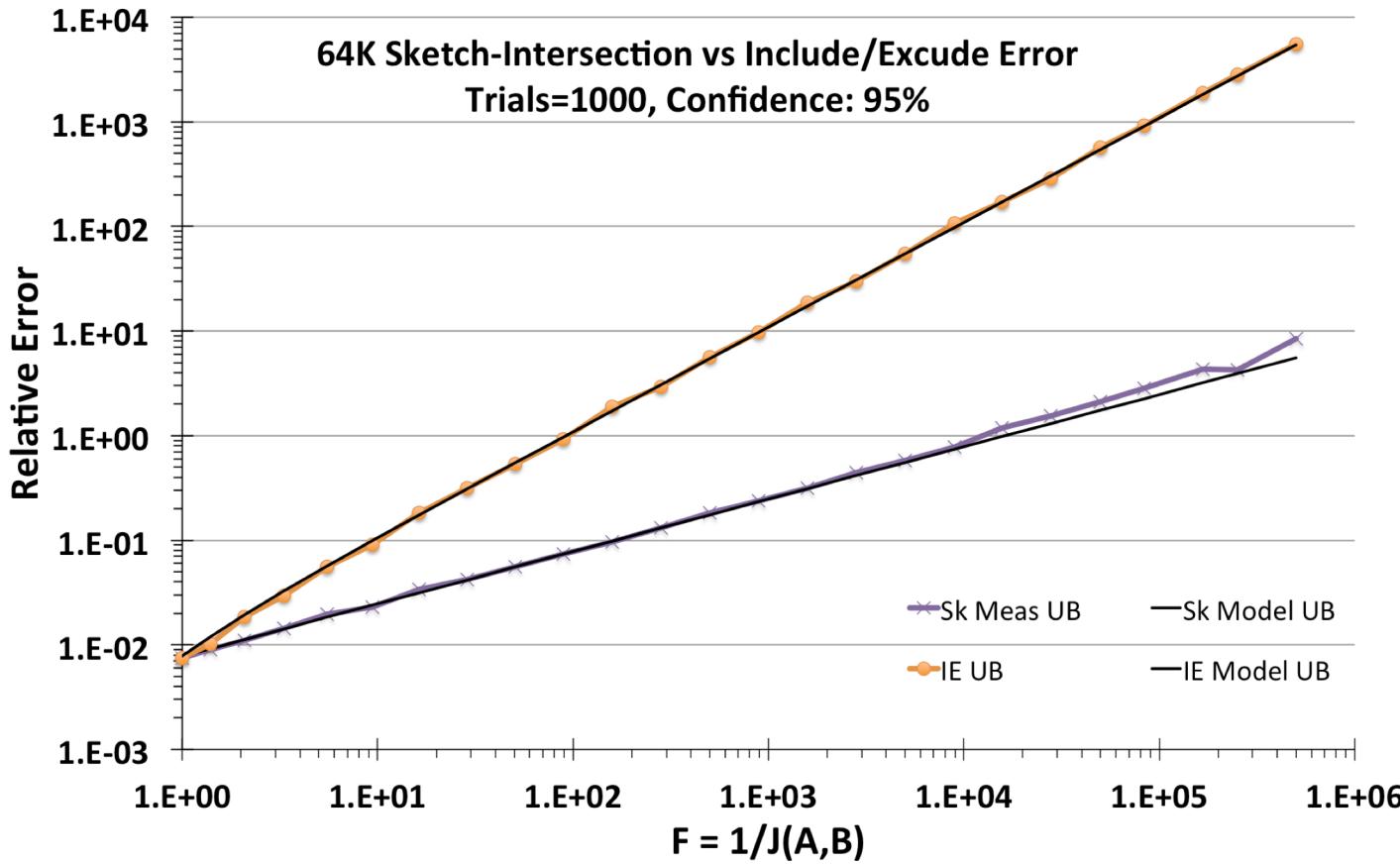
## Set Expressions



$$\Delta = \{\cup, \cap, \setminus\}; \quad \theta_{A \Delta B} = \min(\theta_A, \theta_B); \quad S_{A \Delta B} = \{x < \theta_{A \Delta B}; x \in (S_A \Delta S_B)\}$$

$$\text{est}(|A \Delta B|) = \frac{|S_{A \cup B}|}{\min(\theta_A, \theta_B)} \frac{|S_{A \Delta B}|}{|S_{A \cup B}|} = \frac{|S_{A \Delta B}|}{\min(\theta_A, \theta_B)}, \text{ Using "Broder Rule"}$$

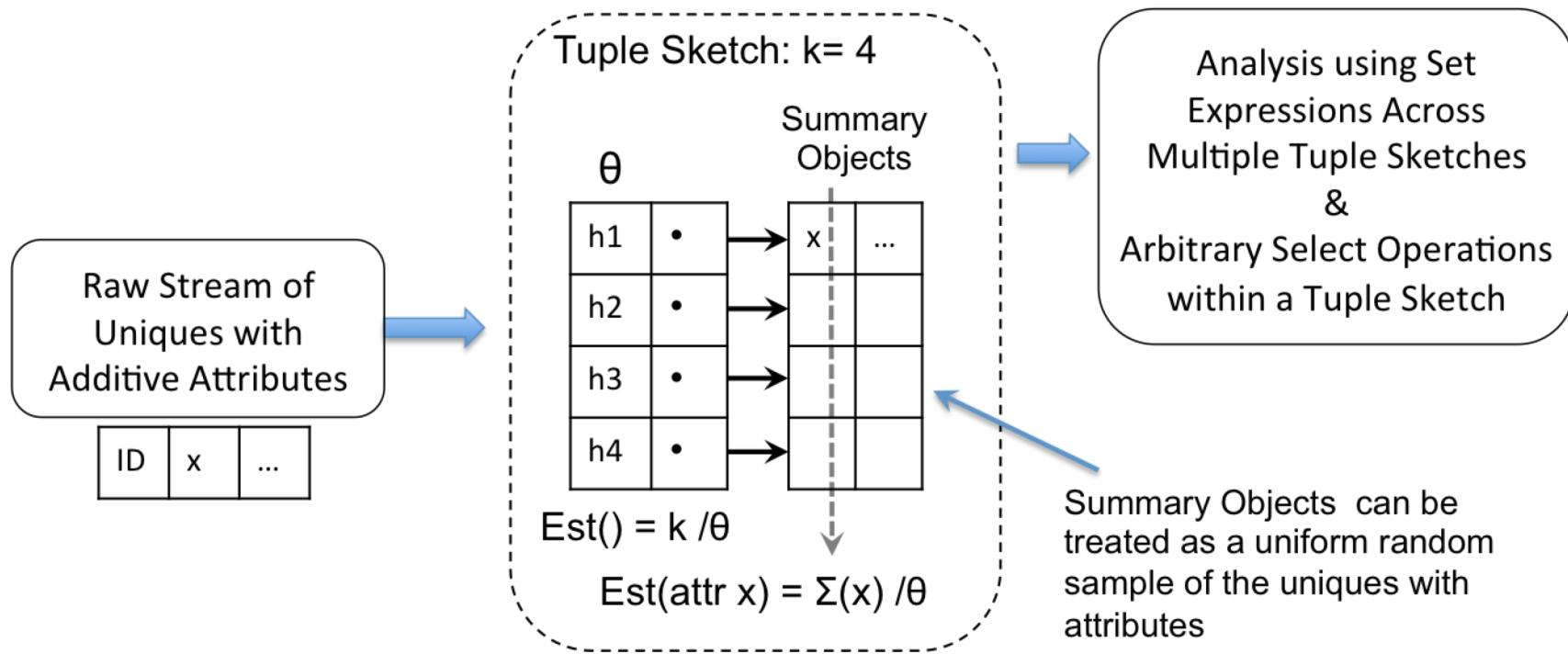
# Theta Sketch Framework: Intersection Accuracy



# Theta Sketch Framework: Tuple Sketch (cont.)

## Tuple Sketch: Adding Attributes to the Theta Sketch

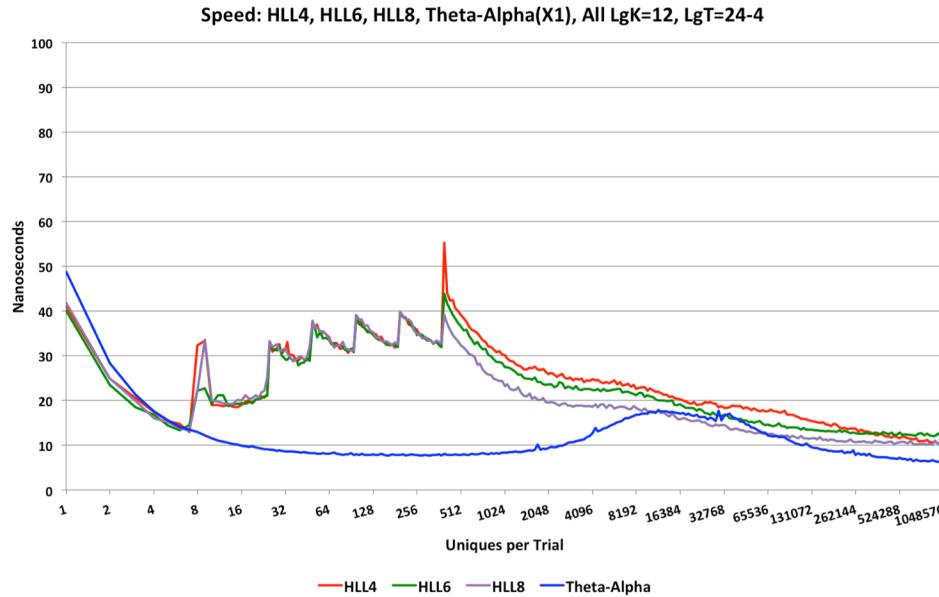
### Add User-defined Attributes



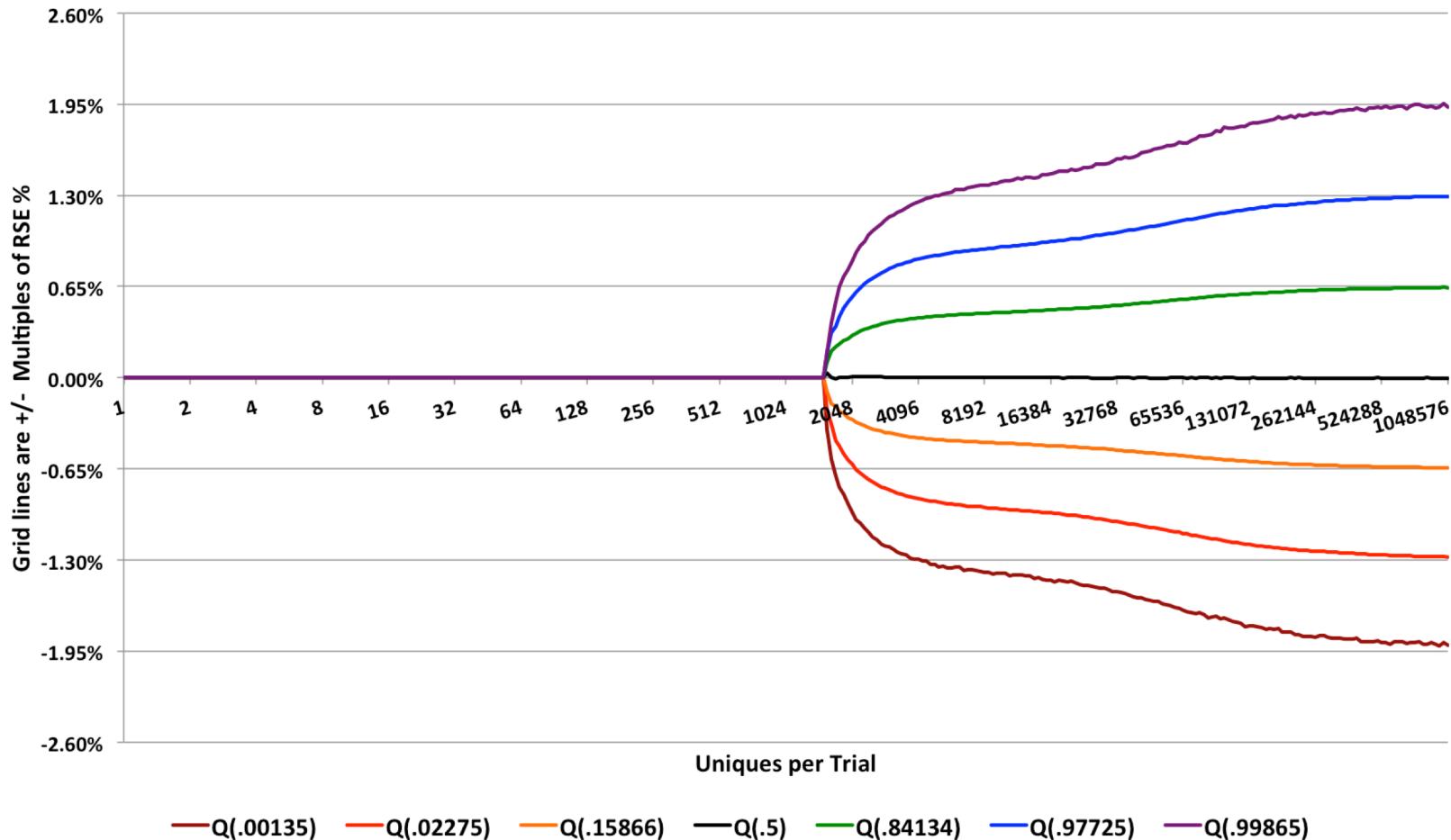
# Innovations for Hyper Log Log Sketches (cont.)

## HllSketch, The Fastest, Most Accurate HLL Sketch Out There

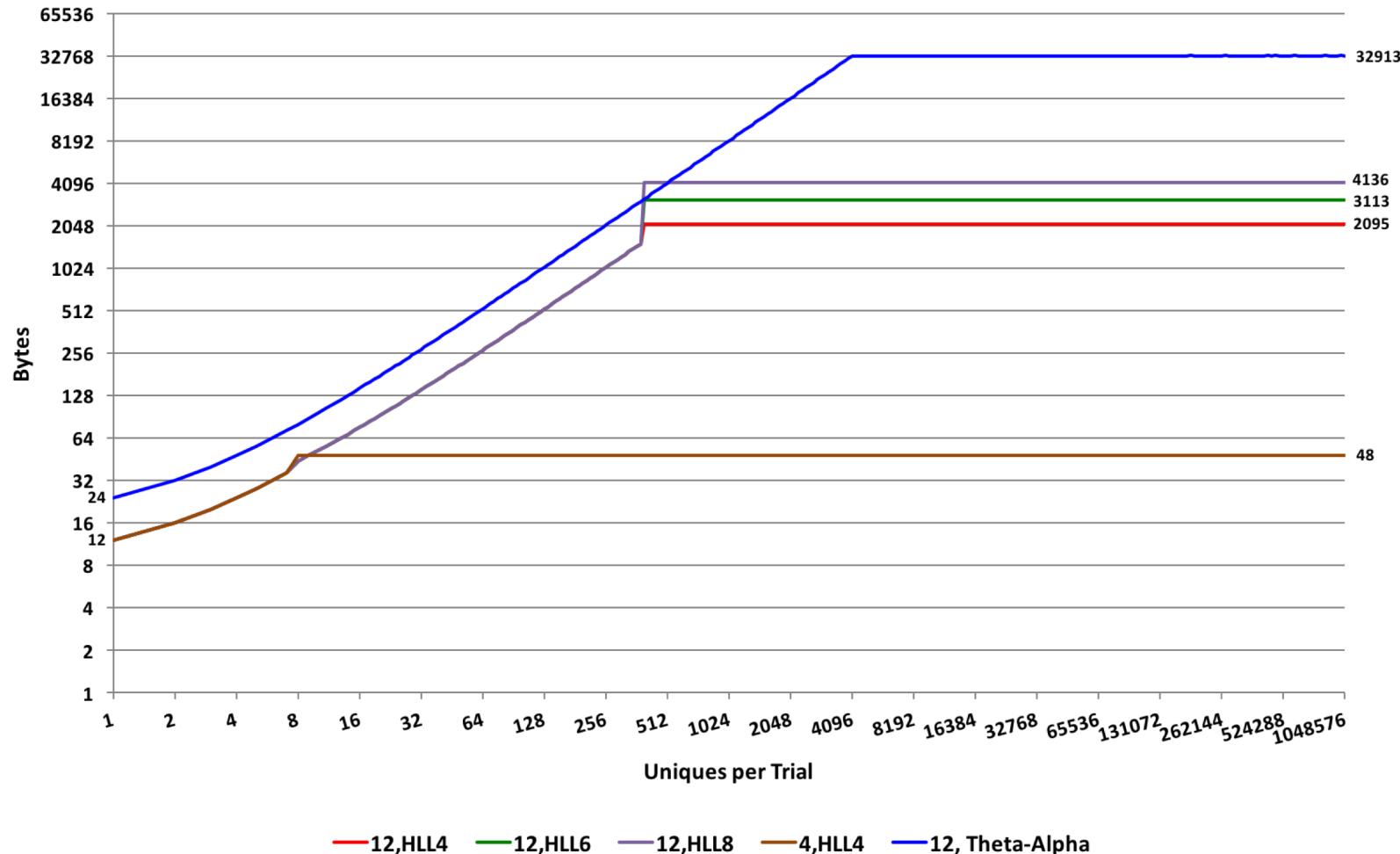
- Highly tuned for speed
- Simple-to-use API
- Operates either On-Heap or Off-Heap
- Leverages low-range estimators from the FM85 paper (mentioned below)



**HLL HIP Measured Quantiles vs RSE**  
**LgK=14, LgT=20, Factor=0.8326, RSE=0.0065**



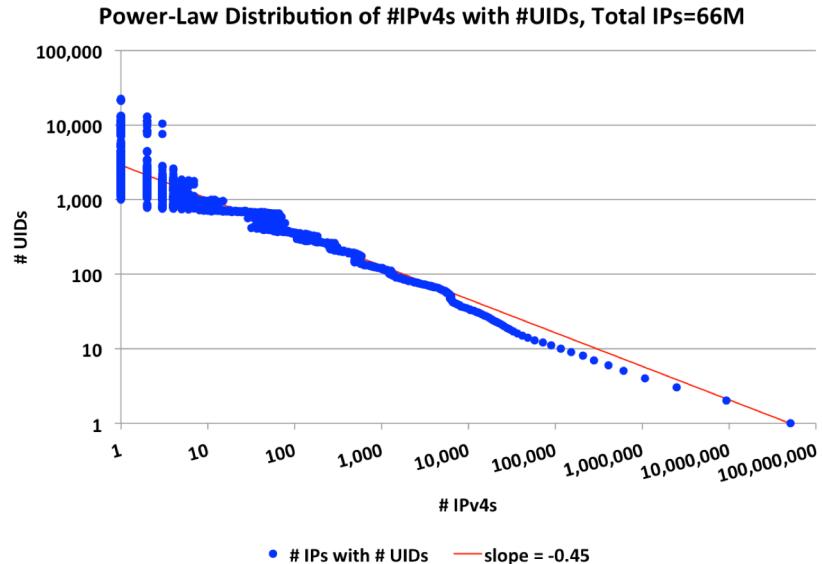
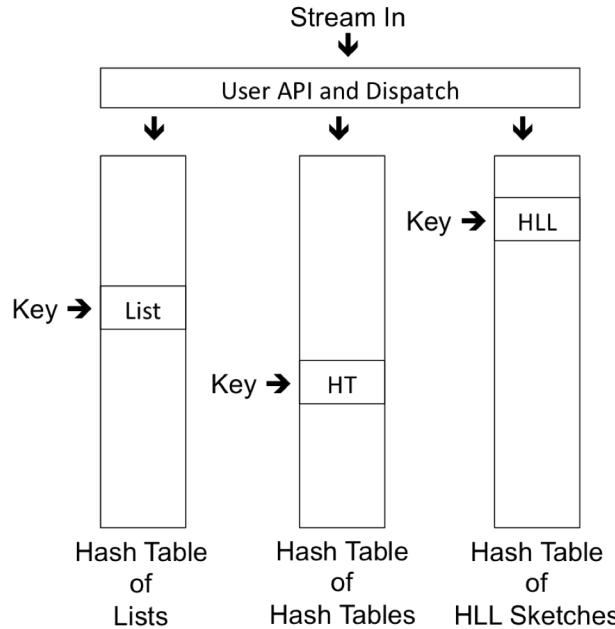
## Serialized Compact Sizes: LgK, Sketch Type



# Innovations for Hyper Log Log Sketches (cont.)

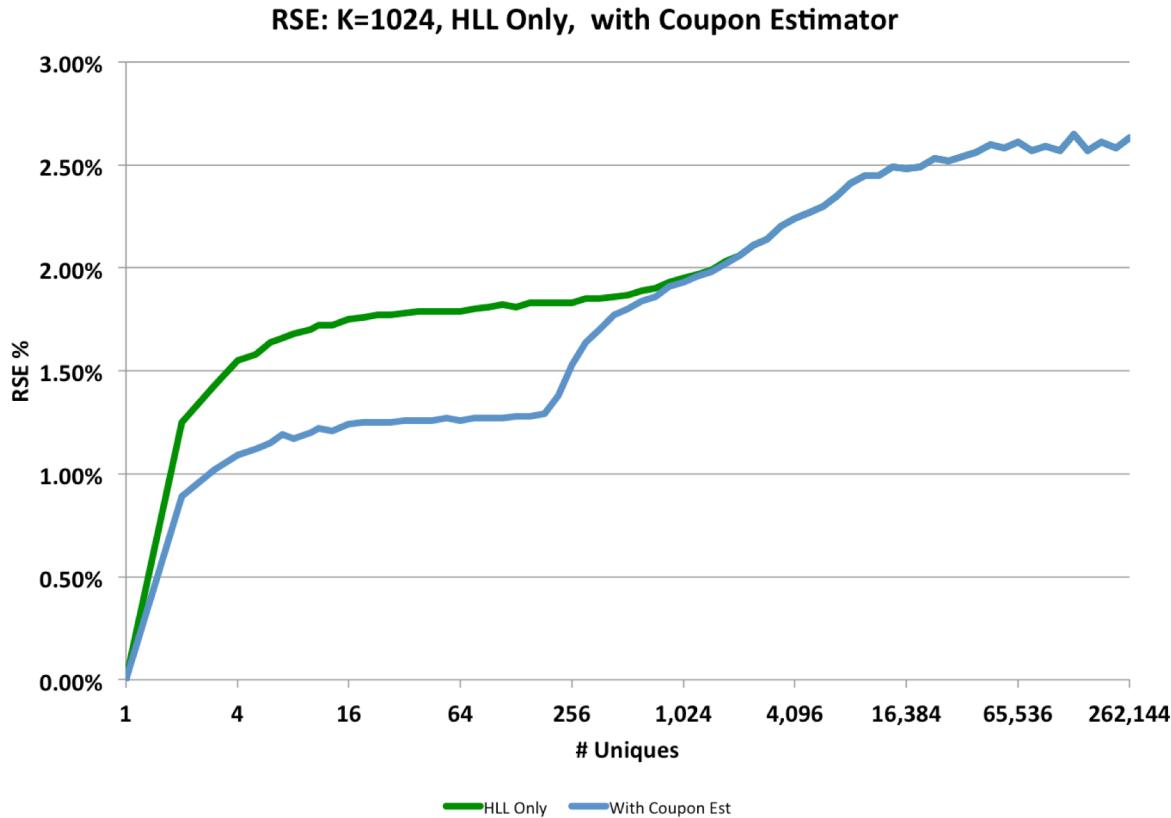
## UniqueCountMap (streams of millions of $K, V$ pairs)

- Real-time cardinality estimates of  $V$  per  $Key$
- Highly space-efficient (100M 4-byte keys require ~1.3GB: ~9 bytes / K for card.)
- Separate data structures manage different phases of sketching process
- Simple-to-use API



# Innovations for Hyper Log Log Sketches (cont.)

## UniqueCountMap Accuracy, K = 1024



# Innovations for Unique Counting Sketches (cont.)

## FM85 / ICON, The Next Generation: Better than HyperLogLog

K. Lang, Back to the Future: an Even More Nearly Optimal Cardinality Estimation Algorithm, arxiv.org/abs/1708.06839 (preparing for publication)

- Builds on Flajolet-Martin 1985 “Probabilistic Counting Algorithms For Data Base Applications”
- Three new estimators: all more accurate than original paper estimators
- More accurate per bit-of-entropy than Flajolet’s 2008 HLL sketches
- Simultaneously wins on all three dimensions of the time/space/accuracy tradeoff.
- Practical implementation is possible
- Already partially implemented in DataSketches HLL sketches

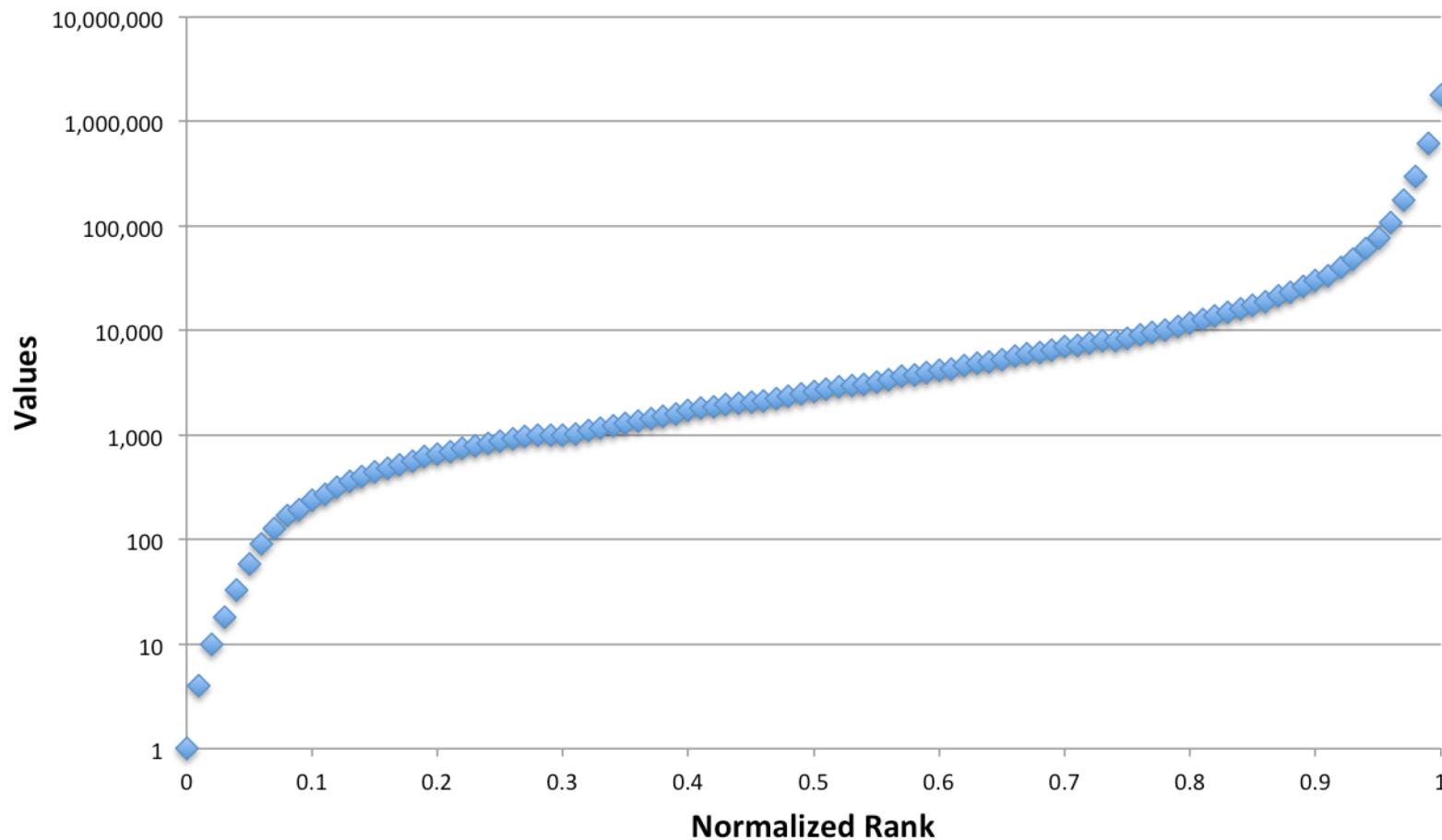
# Innovations for Quantiles / Histogram Sketches

Quantiles, PMF's and CDF's of streams of comparable objects

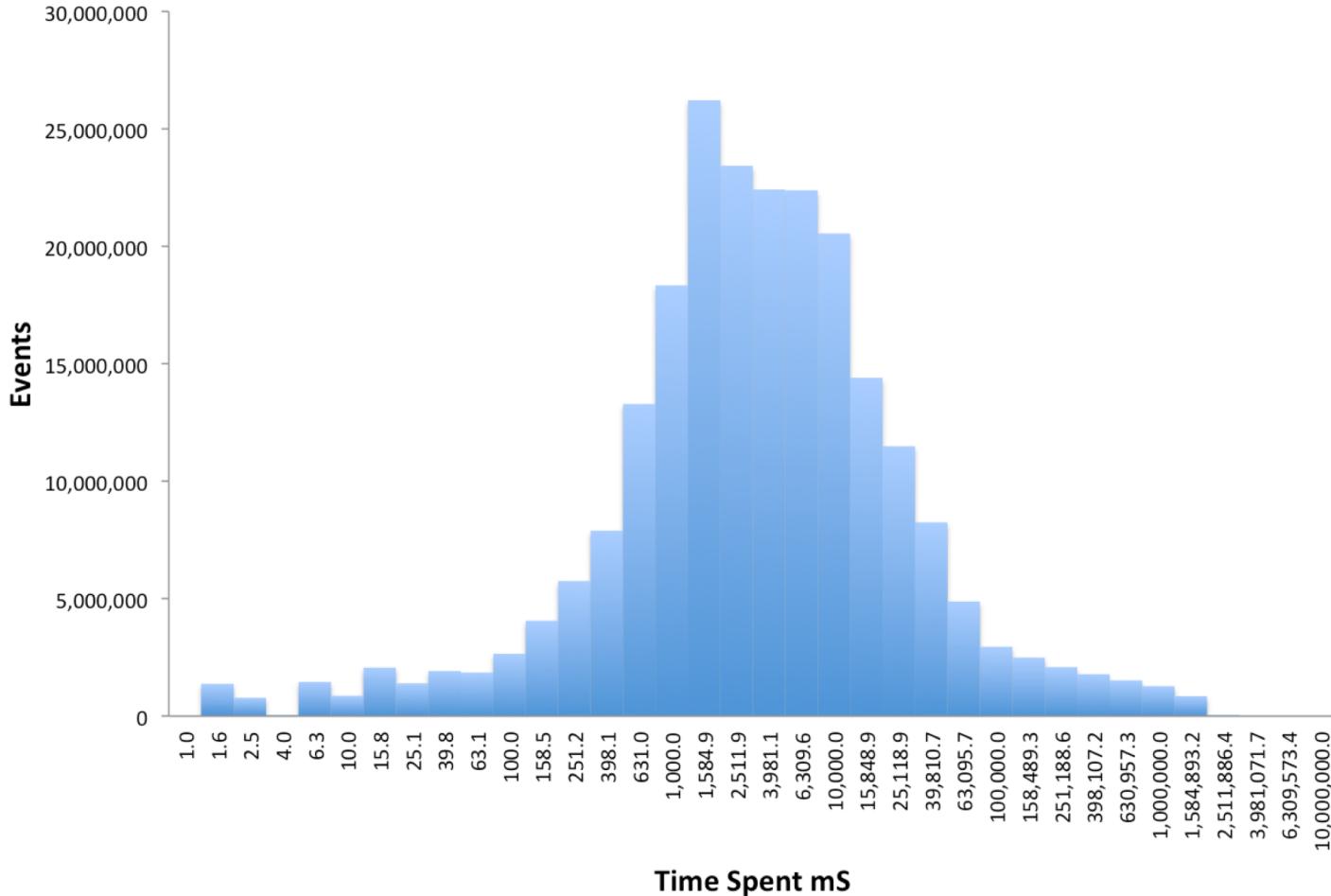
Z. Karnin, K. Lang, E. Liberty: Optimal Quantile Approximation in Streams,  
*IEEE FOCS*, 2016 (The KLL paper)

- Resolves one of the longest standing basic problems in the streaming computational model: The optimal construction of quantile sketches.
- Two implementations in the library
  - Quantiles Sketch: Operates both On-Heap and Off-Heap
  - KLL: Highly optimized for accuracy / space.

## Time Spent Quantiles



# Time Spent Histogram



# Innovations for Frequent Items Sketches

Frequent Items summaries for numerics and generic objects

E. Liberty, M. Mitzenmacher, J. Thaler, J. Ullman: Space Lower Bounds for Itemset Frequency Sketches, *ACM PODS*, 2016

D. Anderson, P. Bevan, K. Lang, E. Liberty, L. Rhodes, J. Thaler:  
A High Performance Algorithm for Identifying Frequent Items in Data Streams.  
*ACM IMC* 2017

- Handles weighted updates in amortized constant time
- Uses simple and fast method for merging sketches that improves on prior work.
- Currently implemented in our Library

# Innovations for Weighted Sampling Sketches

An extension of Edith Cohen's VarOpt Paper

E. Cohen, N. Duffield, H. Kaplan, C. Lund, M. Thorup: Stream sampling for variance-optimal estimation of subset sums. *ACM-SIAM Symposium on Discrete Algorithms*, 2009.

- Created an innovative and efficient implementation
- Extended concepts in the paper to achieve merging with multiple size parameters.
- Currently implemented in our Library

# Innovations for Vector / Matrix Sketches

## Frequent Directions is a New Family of Sketches

Edo Liberty. Simple and deterministic matrix sketching. In *ACM KDD 2013*

- Google Scholar: 152 citations

Mina Ghashami, Edo Liberty, Jeff Phillips. Efficient Frequent Directions Algorithm for Sparse Matrices, In *ACM KDD 2016*

- Google Scholar: 12 citations
- Approximate SVD for very large matrices.
- Created an innovative and efficient implementation
- Currently implemented in our Library.

# Invitation for Collaboration

Thank You!

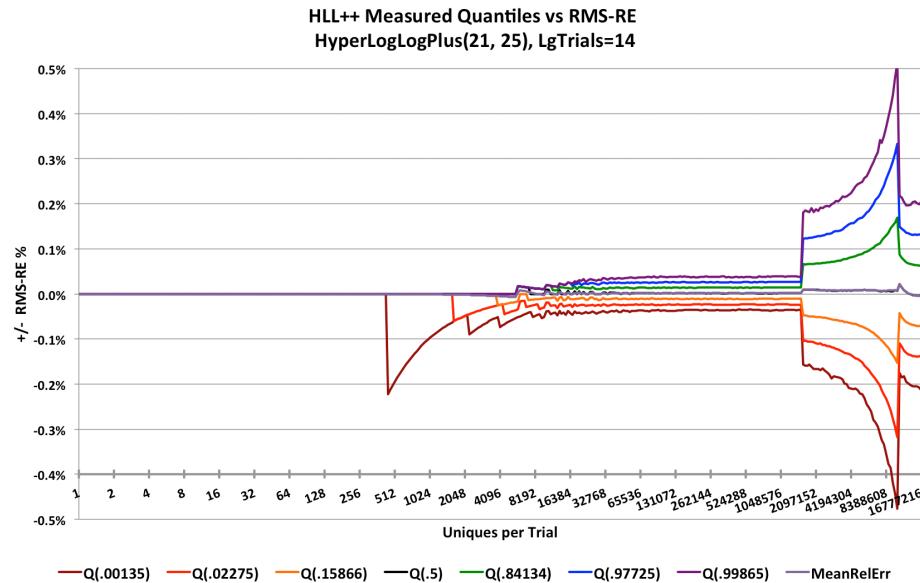
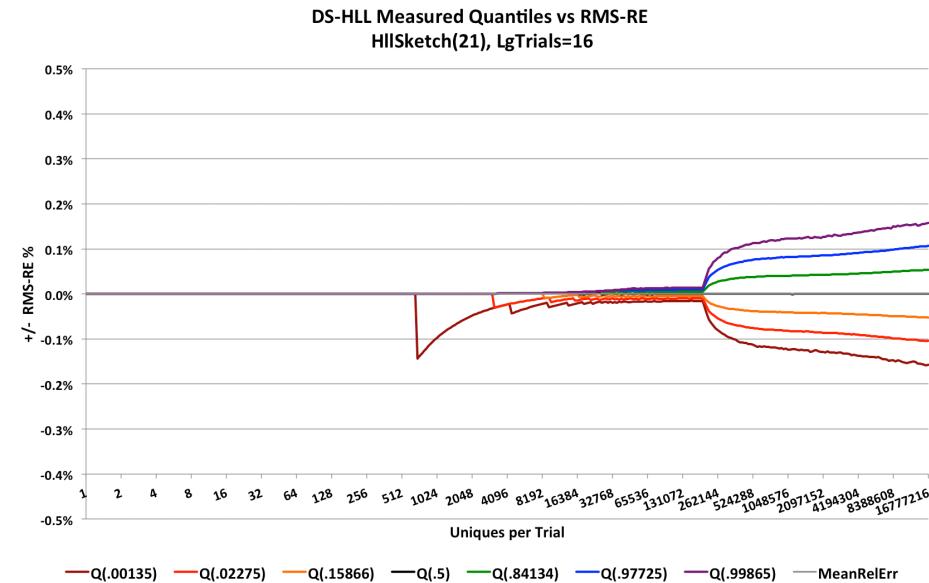
More material available on

*DataSketches.GitHub.io*

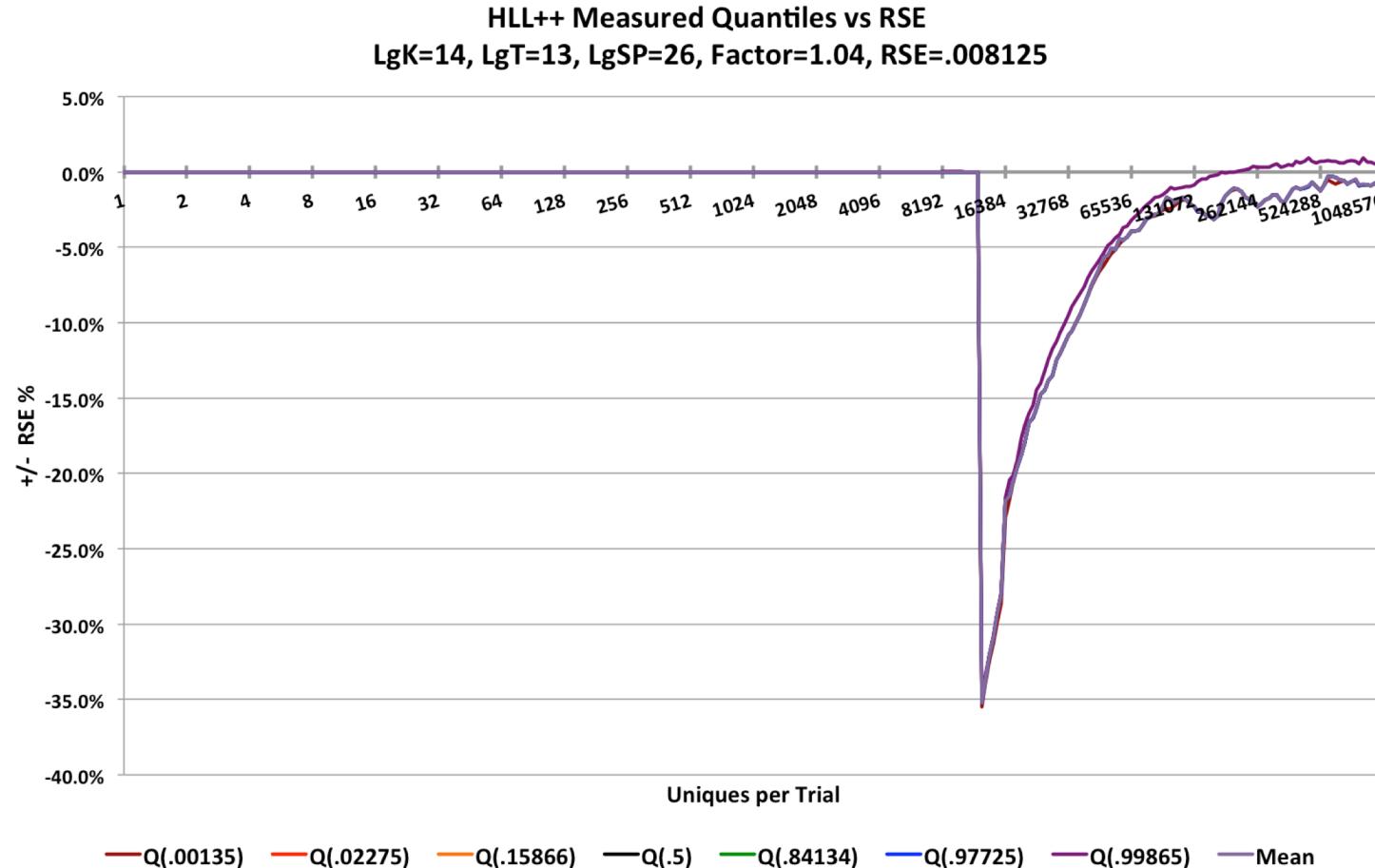
<Pause>

DataSketches HLL Sketch  
versus  
Clearspring implementation of Google HLL++

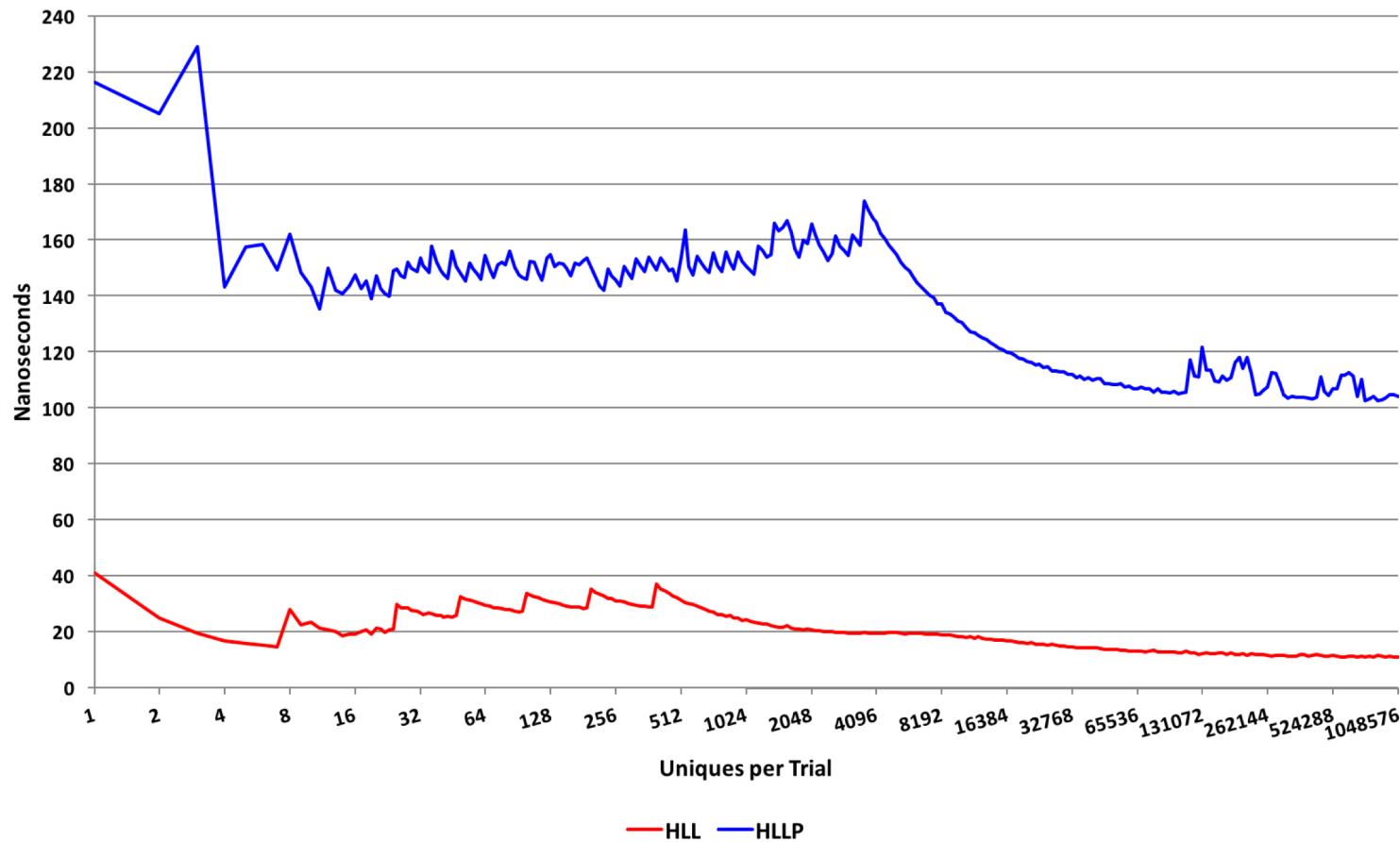
# DataSketches HllSketch vs Clearspring HyperLogLogPlus (Google HLL++)



# Catastrophic Failure of CS Google HLL++

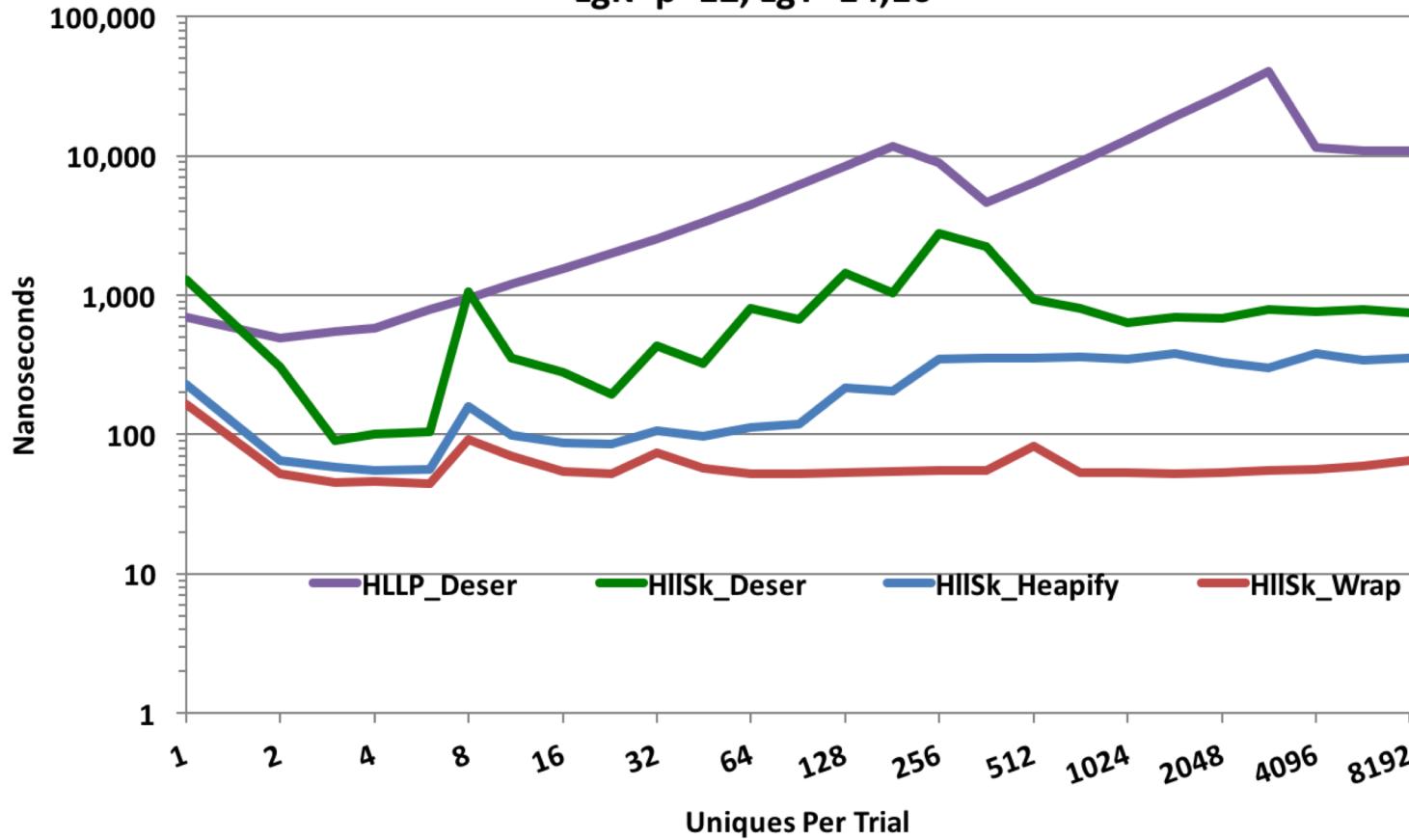


**Update Speed: HLL8 vs HLLP**  
 $\lg K=12, \lg T=23-4$



## HyperLogLogPlus & HllSketch in Different Deserialization Modes

LgK=p=12, LgT=14,16



# DataSketches HLLSketch vs ClearspringHLL++ Summary

Attribute	DataSketches HLL	Clearspring HLL++
Update Speed	10 ns	100 ns
Deserialization Speed	100 ns	10,000 ns
Accuracy given $k = 2^{21}$	0.1% ( $2 \sigma$ )	0.35% ( $2 \sigma$ ), 35% (failure)
Merge different $k$ 's	Yes	No
Error Bounds	Yes	No
Unit Tests	> 90% coverage	Minimal
Characterization Tests	Yes	No
Javadocs	Extensive	Almost none
Dependencies	None	Many
Active Developers	Yes	No
Stability	Very High	Buggy

<Pause>

# Case Study 1: Simple Batch Distinct Counting

- **Web Logs:**  $Dim1$ : PageID,  $Dim2$ : Time-Stamp,  $Id1$ : Browser Cookie,  $Id2$ : UserID  
( + many other fields)
- **Data Size:** ~245GB daily; ~7.6TB monthly
- **Task:** Report: Count Distinct  $Id1$  and  $Id2$  by PageID, and by hour, day, week, and month
- **Note:** This case study was run on Pig, Hive and Spark. The results below are from Pig. Hive and Spark showed similar results.

# Case Study 1: Hourly Process

**Exact:** For Hourly Reports and Basis for Daily Reports

Sub-Task	Data Stored	Sub-Task	Data Stored
Stage 1: <ul style="list-style-type: none"><li>• Read Raw Data</li><li>• -&gt; Hourly Tables</li></ul>	Create Table1: Group By {site, hour, id1}  Create Table2: Group by {site, hour, id2}	Stage 1: <ul style="list-style-type: none"><li>• Read Raw Data</li><li>• -&gt; Data Cube</li></ul>	Create Sketches Cube: By Dim Combination {site, hour, sketch(id1), sketch(id2)}
Intermediate Size	<b>33.4 GB 1 Month of Hourly</b>	Intermediate Size	<b>1.1 GB</b>
Stage 2a: <ul style="list-style-type: none"><li>• Read Hourly Tables</li><li>• Count Uniques</li></ul>	Group By {site, hour}, Count Id1 Group By {site, hour}, Count Id2	Stage 2 <ul style="list-style-type: none"><li>• Read Data Cube</li><li>• Produce Hourly Report</li></ul>	Merge Sketches across Chosen Dimensions
Stage 2b: <ul style="list-style-type: none"><li>• -&gt; Hourly Report</li></ul>	Join: {site, hour, count(id1), count(id2)}		
Total CPU Time	<b>1.39M Sec</b>	Total CPU Time	<b>1.06M Sec</b>

# Case Study 1: Daily Rollups

**Exact:** For Daily Reports and Basis for Weekly and Monthly

Sub-Task	Data Stored
Stage 1: <ul style="list-style-type: none"><li>• Read Hourly Intermediates</li><li>• -&gt; Daily Tables</li></ul>	Create Table1: Group By {site, day, id1}  Create Table2: Group by {site, day, id2}
Intermediate Size	16.0 GB just for Daily
Stage 2a: <ul style="list-style-type: none"><li>• Read Daily Intermediates</li><li>• Count Uniques</li></ul>	Group By {site, day}, Count Id1 Group By {site, day}, Count Id2
Stage 2b: <ul style="list-style-type: none"><li>• Produce Hourly Report</li></ul>	Join: {site, day, count(id1), count(id2)}
Total CPU Time	96,300 sec

**Sketches Cube:** For All Reports

Sub-Task	Data Stored
Stage 1: <ul style="list-style-type: none"><li>• Read Data Cube</li><li>• -&gt; Produce Daily Report</li></ul>	N/A
Intermediate Size	N/A
Total CPU Time	709 Sec

# Case Study 1: Weekly, Monthly Rollups

## Exact: For Wk/Mo Reports

Sub-Task	Data Stored
Stage 1: • Read Daily Tables	Create Temp Table1: Group By {site, wk/mo, id1}  Create Temp Table2: Group by {site, wk/mo, id2}
Stage 2a: • Read Temp Tables • Count Uniques	Group By {site, wk/mo}, Count Id1 Group By {site, wk/mo}, Count Id2
Stage 2b: • Produce Report	Join: {site, wk/mo, count(id1), count(id2)}
Total CPU Time	Week: 43,500 sec Month: 46,500 sec (via daily) Month: 70,900 sec (via hourly)

## Sketches Cube: For All Reports

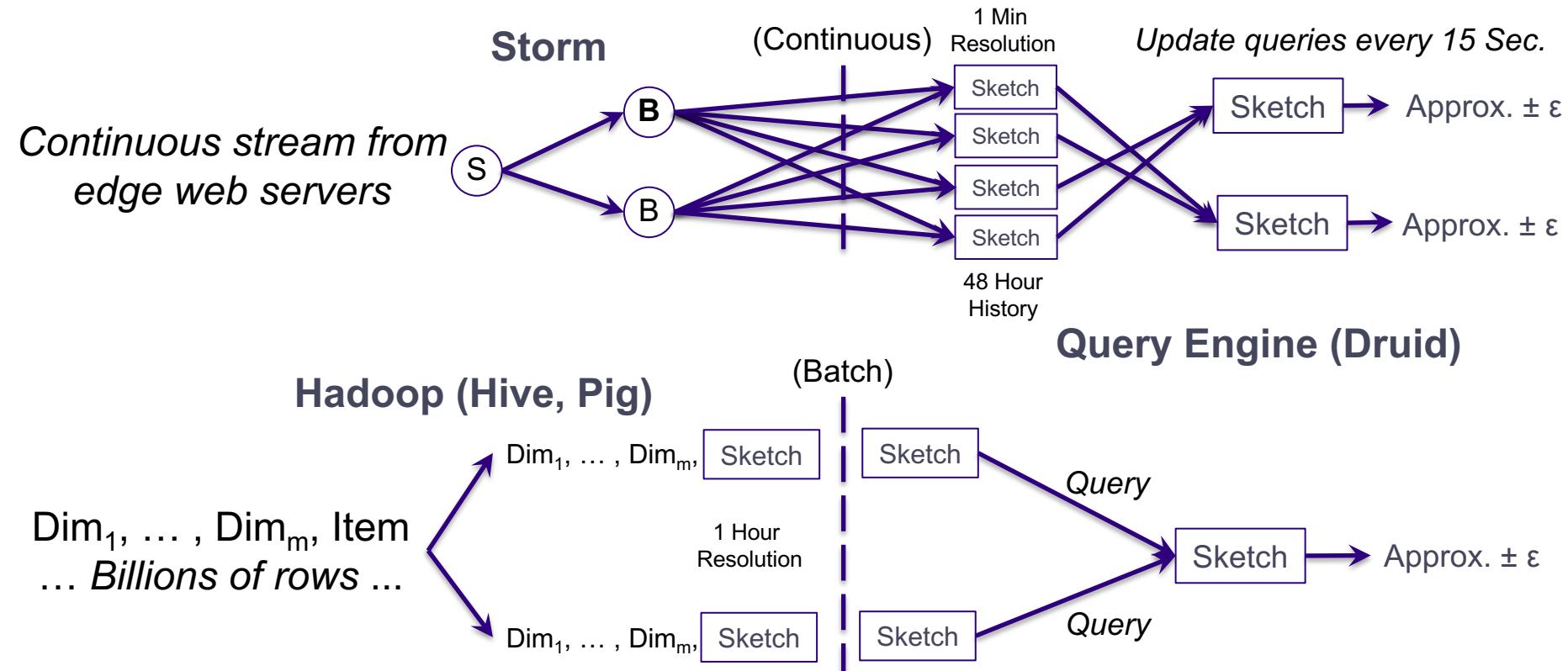
Sub-Task	Data Stored
Stage 1: • Read Data Cube • -> Produce Weekly or Monthly Reports	N/A
Total CPU Time	Week: 424 Sec Month: 466 Sec

# Case Study 1: Perspectives

- Only a few dimensions and metrics, moderate data size
  - Manageable with exact counting
  - However, sketching can still show substantial benefits, especially in real-time streaming
- Batch process (e.g. Pig, Hive)
  - Substantial job overhead penalizes the relative sketch compute time.
  - Contrast this to real-time reporting engines (e.g. Druid), where rollups can be computed in seconds.
- As the number of dimensions grows, the benefit of using sketches becomes even more dramatic

# Real-time, Late Data Updates

## Case Study: Flurry/Druid Sketch Flow Architecture



# Case Study: Flurry, Before and After

- Customers: >250K Mobile App Developers
- Data: 40-50 TB per day
- Platform: 2 clusters X 80 Nodes / cluster; each Node is 24 CPUs/48 hyper-threads, 250GB RAM

	Before Sketches	After Sketches
System environment	<b>Hadoop:</b> complete brute-force, pre-computation of uniques <b>HBase:</b> Query Engine	<b>Hadoop:</b> w/ Sketches <b>Storm:</b> Real-Time w/ Sketches <b>Druid:</b> Query Engine w/ Sketches
Cost (Virtual Core-Sec) (VCS)	<b>Mo Total: ~80B vcs</b>	<b>Mo. Total: ~20B vcs</b> <b>BIG WIN : Eco Friendly! Lower \$</b>
Result Freshness	<b>Daily: 2 to 8 hours</b> <b>Weekly: ~3 days</b> <b>Real-time Not Feasible</b>	<b>15 seconds!</b>
Flexibility	Uniques not additive, Must always compute from raw data, Not flexible.	Sketches are additive across time + other dim. Allow set expressions, Architecturally simpler

## Real Time

All Apps (337) ▾

## Usage

## Retention

## Active Devices

## Active Users

## Audience

## Events Summary

## Events Details

[Yahoo Mail++ - Production](#)[Yahoo Mail++ - Production](#)[Add Filter](#)

as of 9:36:20 PM

## Real Time Dashboard

ACTIVE DEVICES LAST 24 HOURS

**15,365,820**

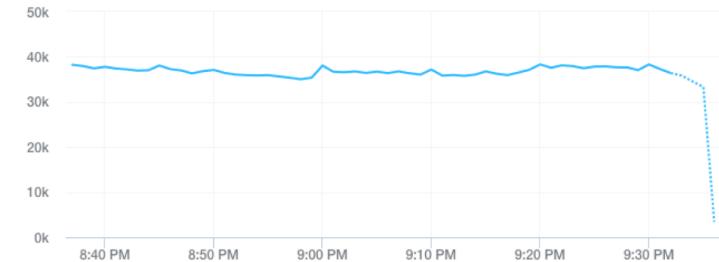
SESSIONS TODAY

**72,541,482**

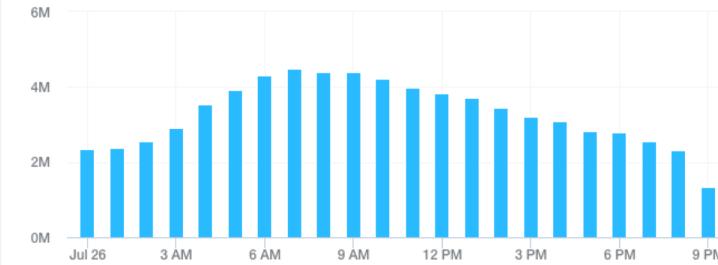
SESSIONS LAST 24 HOURS

**77,724,619**

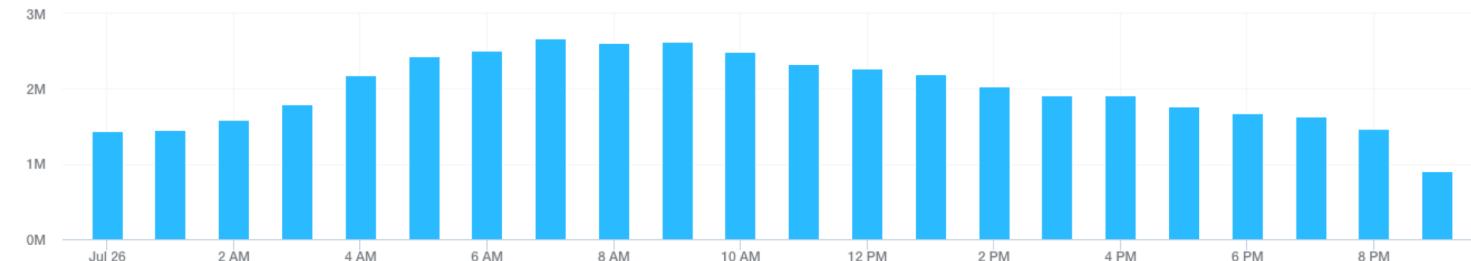
## SESSIONS - MINUTE



## SESSIONS - HOUR



## ACTIVE DEVICES - HOUR



Real Time

Usage

**Retention**

Active Devices

Active Users

Audience

Events Summary

Events Details

All Apps (337) ▾

Yahoo Mail++ - Production



7News - Dogfood



①Filtering coming soon

Jun 27, 2016 - Jul 26, 2016 ▾

## Retention Dashboard

AVG DAY 7 RETURN RATE

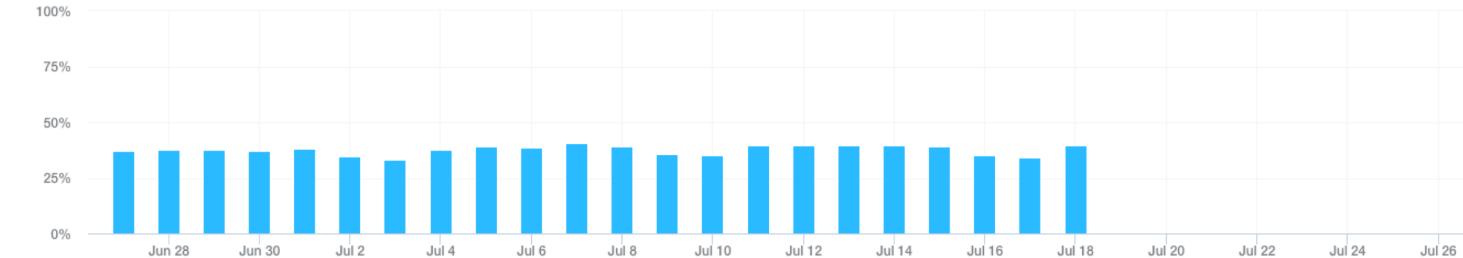
**37.63%**

AVG DAY 7 ROLLING RATE

**64.22%**

PERCENT RETURNING ON DAY 7 ▾ AFTER INSTALL

DAY ▾



ROLLING RETENTION

DAY ▾

