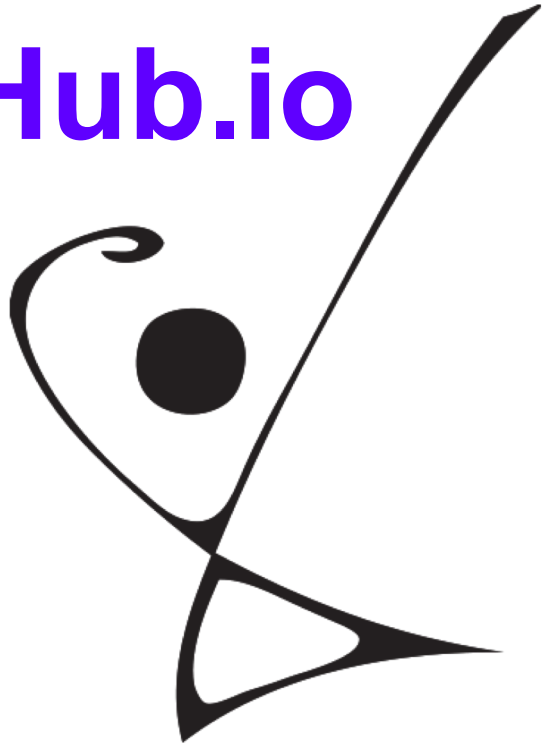
DataSketches.GitHub.io

A Required Toolkit for the Analysis of Big Data



Lee Rhodes
Oath Senior Tech Council 1 May 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.

Extended Team

- Graham Cormode, Ph.D., Professor, University of Warwick, Computer Science. Joined 2017.
- Srikanta Tirthapura, Ph.D., Professor, Iowa State University, Elec. & Computer Science. Joined 2017.
- Christopher Musco, Ph.D. Student, MIT, Theoretical Computer Science, Lin Algebra. Joined 2018.
- Pryce Bevan, Daniel Anderson, Ph.D. Students, Georgetown Univ. Joined 2017.

Our Challenge

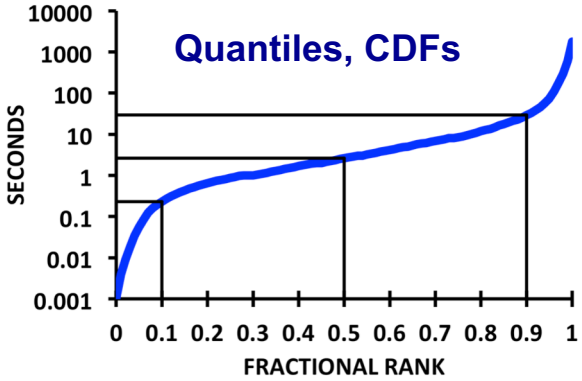
Example: Web Site Logs

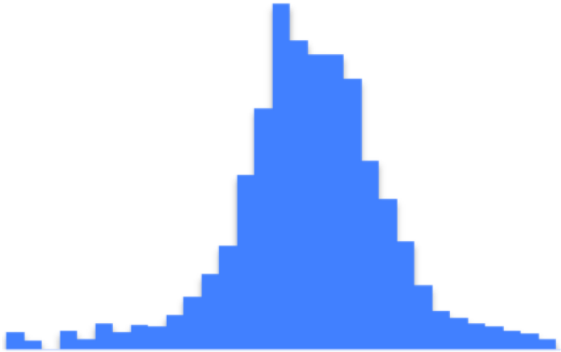
Time Stamp	User ID	Device ID	Site	Time Spent Sec	Items Viewed
9:00 AM	U1	D1	Apps	59	5
9:30 AM	U2	D2	Apps	179	15
10:00 AM	U3	D3	Music	29	3
1:00 PM	U1	D4	Music	89	10
Billions of *K,V* Pairs …					

Analyze This Data In Near-Real Time.

Some Very Common Queries …

Unique Identifiers

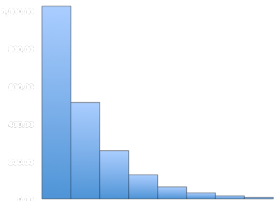




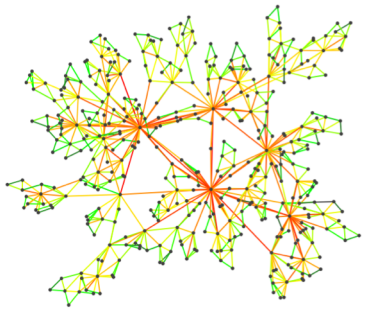
Histograms, PMFs

Vector & Matrix Operations: SVD, etc.

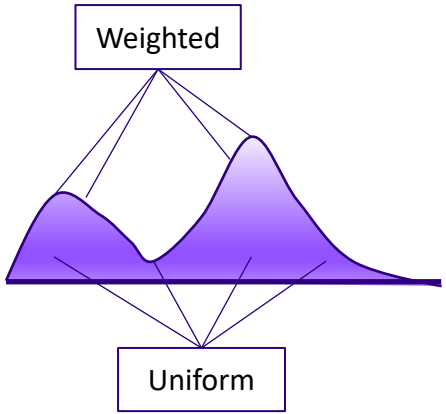
$$\begin{Bmatrix} 5 & \cdots & 2 \\ \vdots & \ddots & \vdots \\ 4 & \cdots & 3 \end{Bmatrix}$$



Frequent Items



Graphs



Reservoir Sampling

Are All Computationally Difficult

Our Mission…

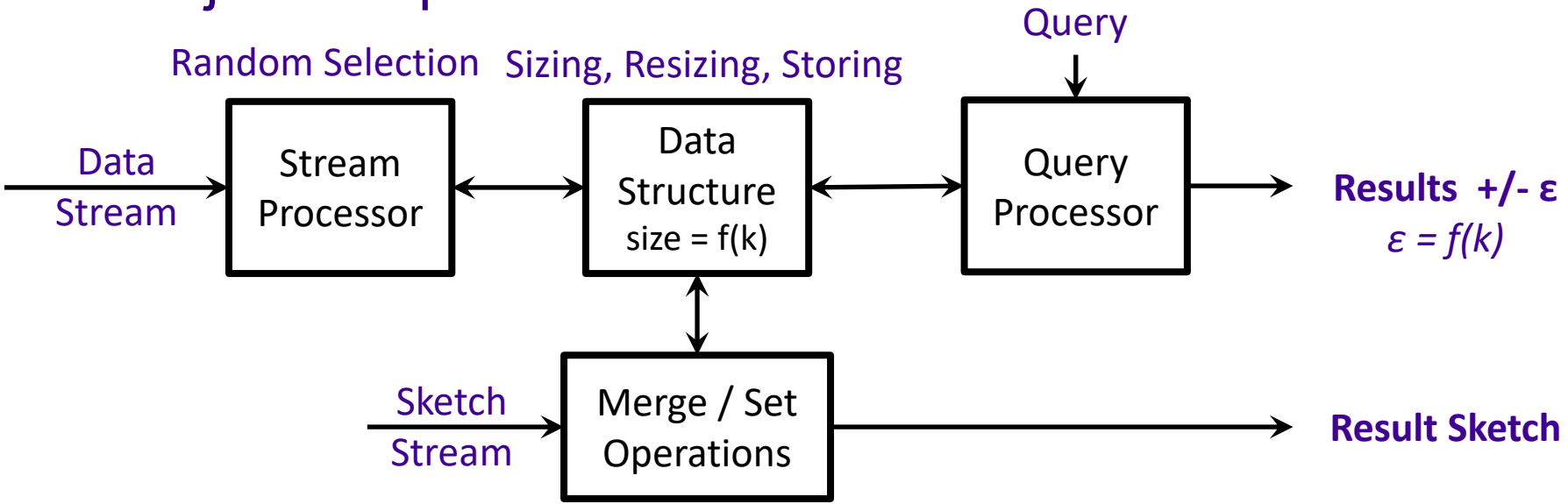
Develop **Production Quality**

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

to Address these Difficult Queries

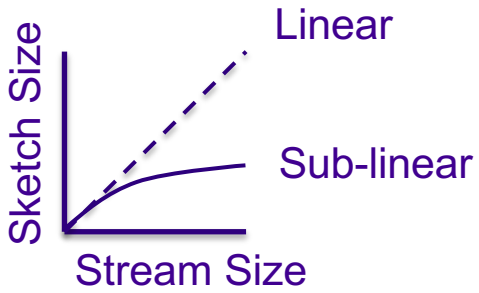
“Any sufficiently advanced technology is indistinguishable from magic” – Arthur C. Clarke

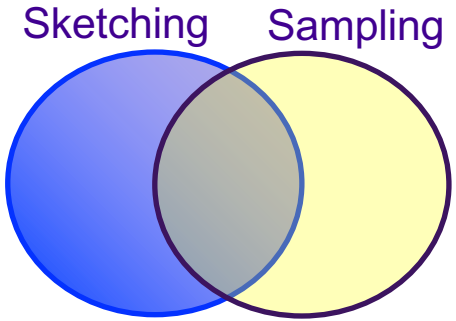
Four Major Components of a Sketch



4 Major Characteristics

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





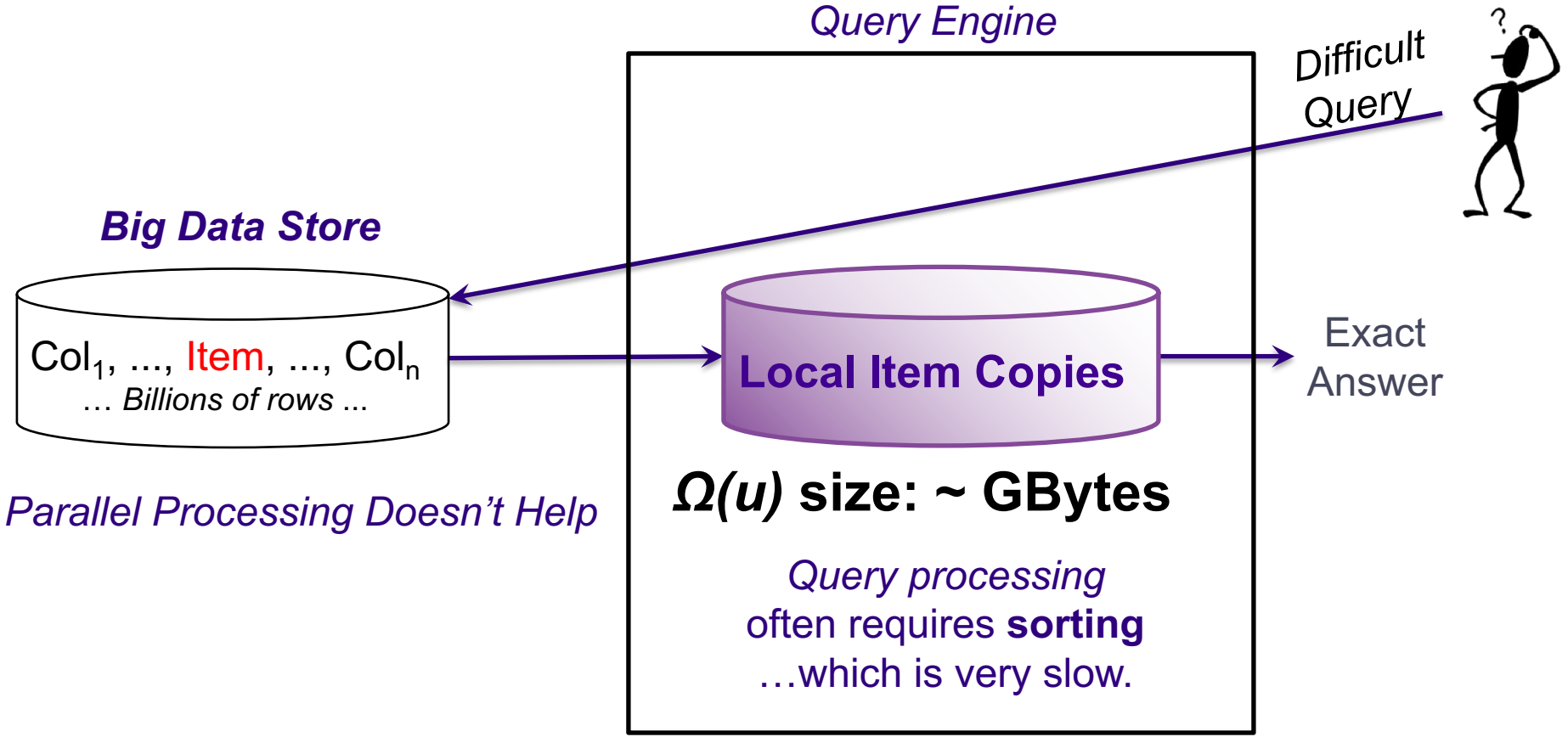
What Does “Production Quality” Mean?

- 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.
- World-Class Performance
  - Best accuracy, space utilization and fastest performance, of anything out there!
  - Defensively programmed to minimize exceptions & problems in systems applications

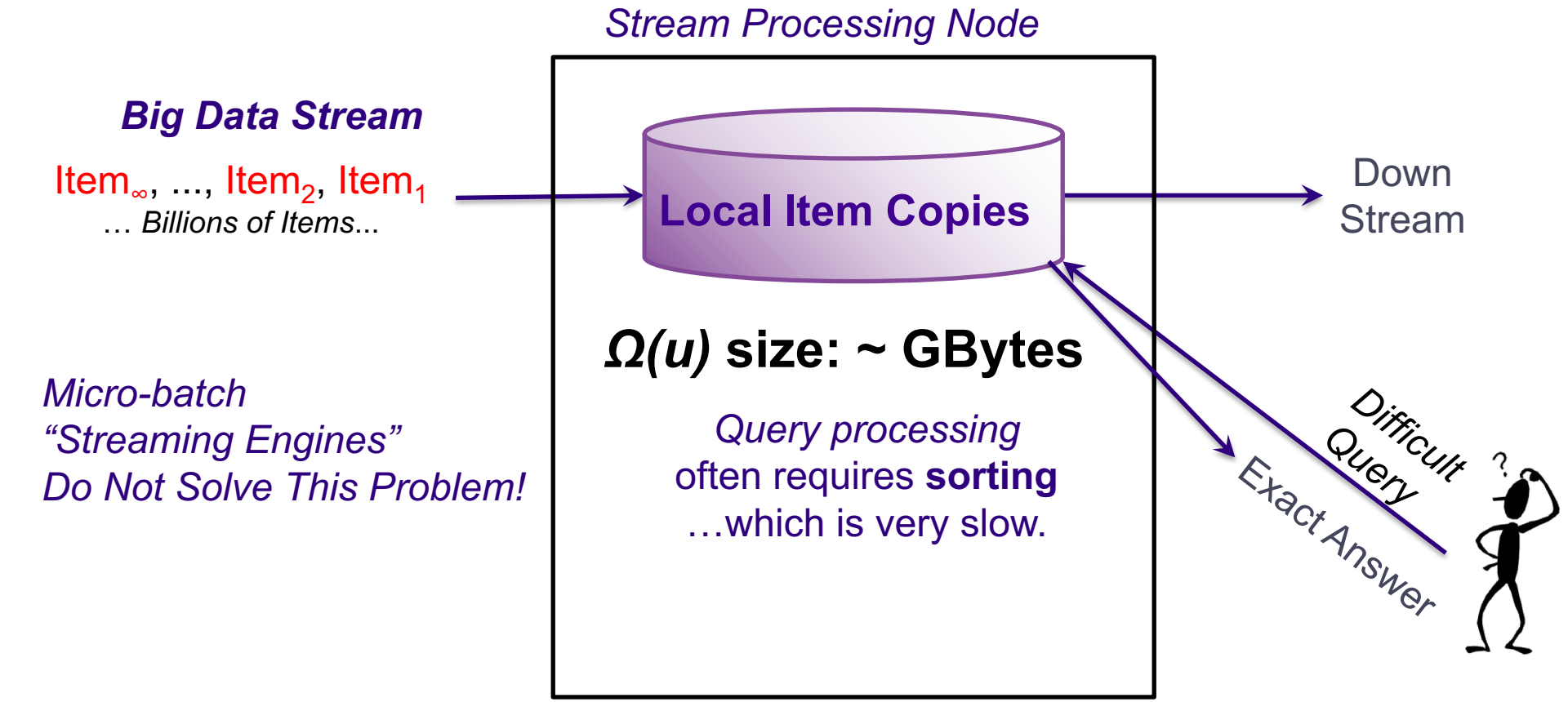
Systems Architecture Using Sketches

How do we use Sketches, and Why?

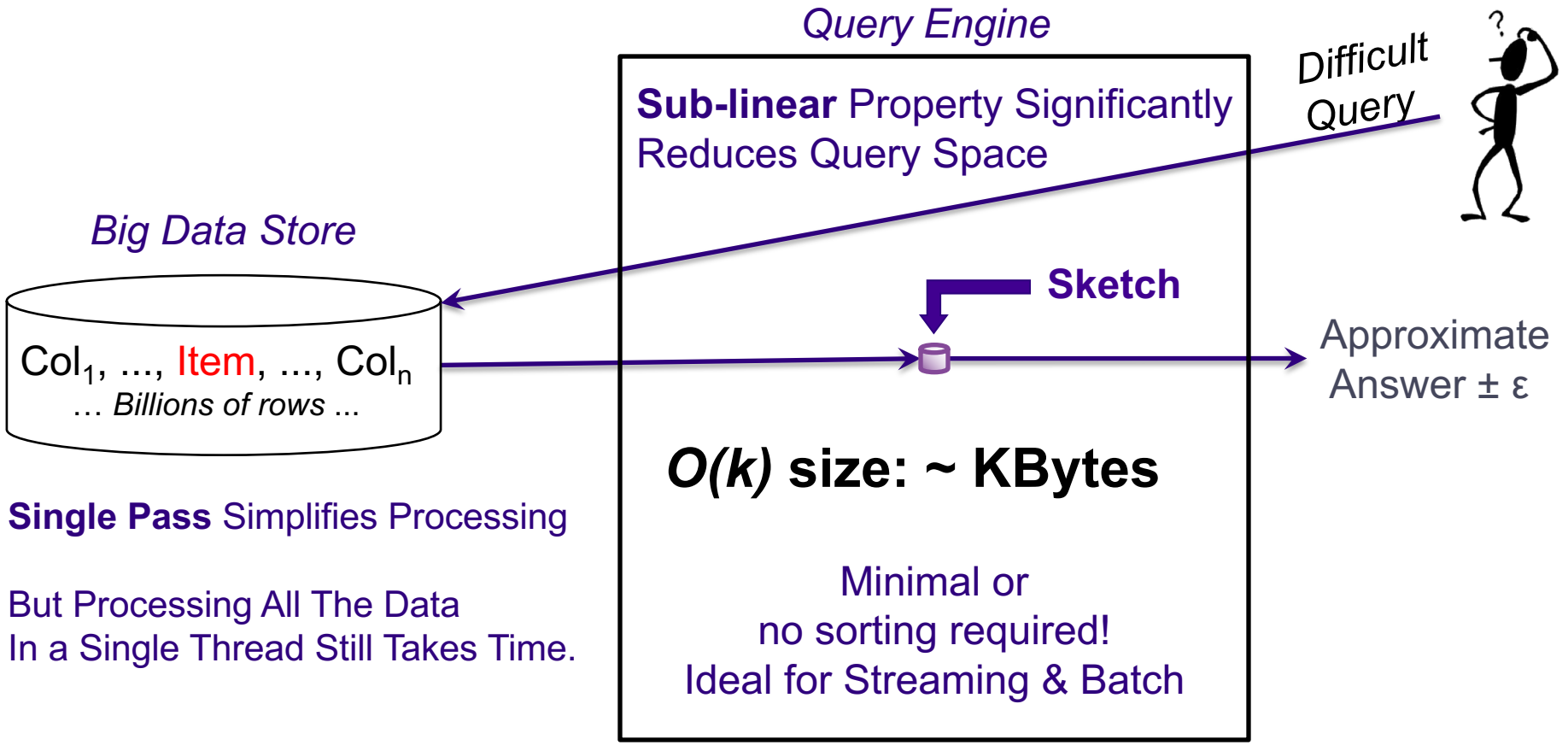
Exact Approach From Stored Data



Exact Approach From Streamed Data (e.g. Storm)



Sketches First Big Win: Query Space

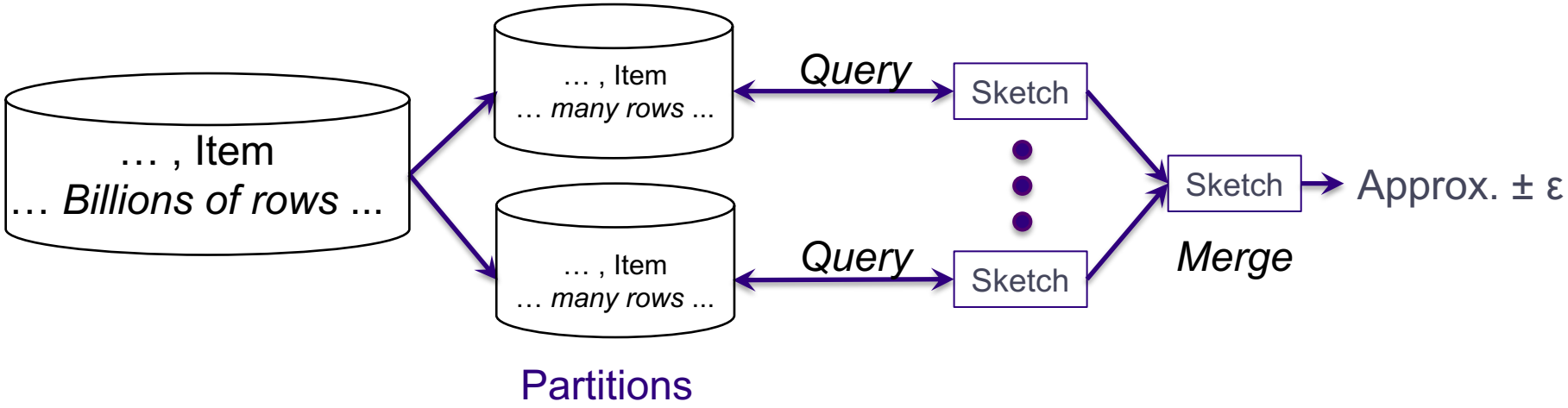


Single Pass Simplifies Processing

But Processing All The Data
In a Single Thread Still Takes Time.

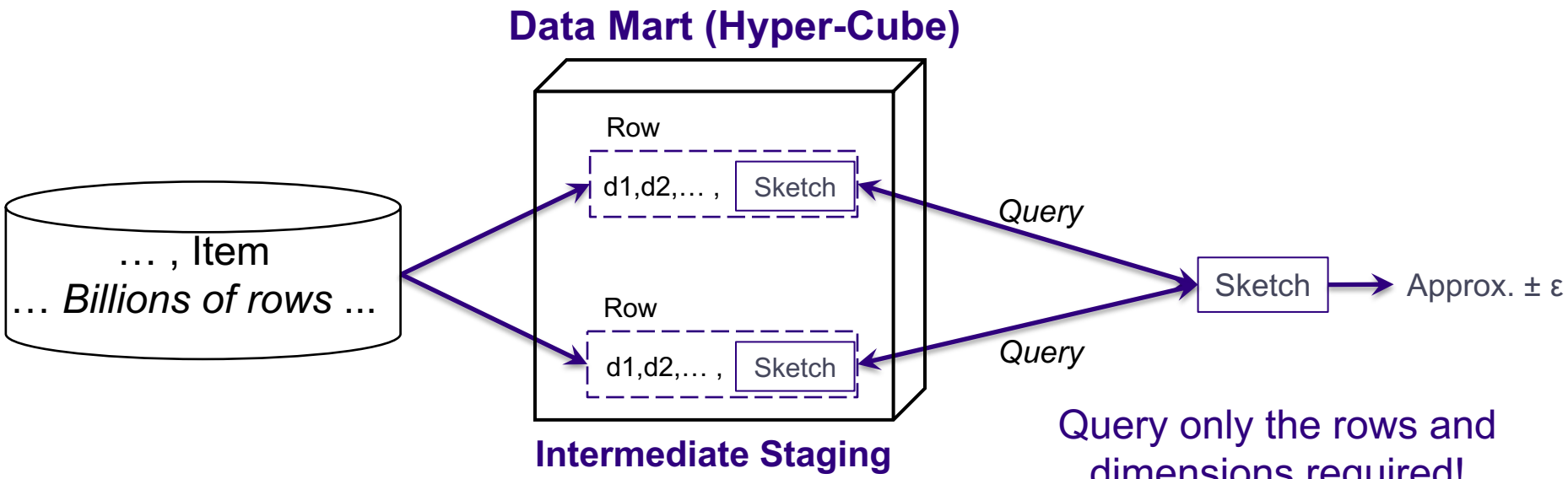
The Second Big Win: Mergeability

- **Mergeability** Enables Parallelism
- With No Additional Loss of Accuracy!
- But Data Skew Across Partitions Can Still Be A Systems Challenge



Big Wins 3, 4: Speed, Simpler Architecture

Intermediate Sketch Staging Enables Query Speed & Simpler Architecture

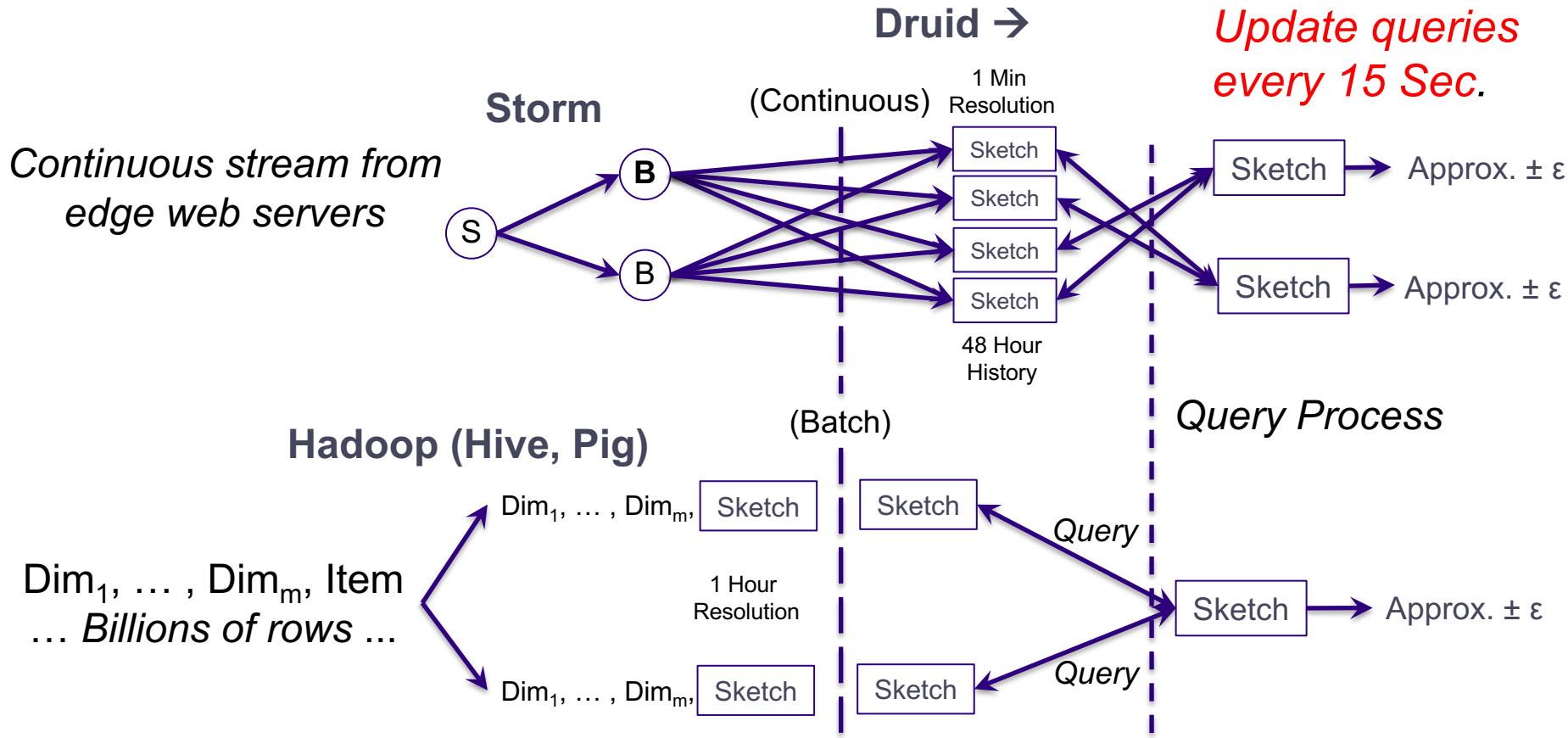


Query only the rows and dimensions required!

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

Big Wins 5, 6: Real-time, Late Data Updates

Case Study: Flurry/Druid Sketch Flow Architecture



Case Study: Real-time 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 Win 7:
Lower System $

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

* VCS: Virtual Core Seconds

Advantages of Sketch-based System Design

- Architectural simplicity
  - Fewer processing steps: Multiple sketches in parallel in one pass
  - Fewer intermediate tables: Store sketches vs. reprocessing raw data
  - Results stored in “additive” data cube instead of non-additive, one-time reports
- Enables reporting on arbitrary dimension combinations
  - Fast merging: Recombine sketches as needed
  - Rolling hour, day, week or month
  - Simple time zone adjustments
- Set operations are cheap
  - Unions (merging) – all sketches have this capability
  - Intersections, e.g. for user retention – Theta, Tuple Sketches
  - Set differences, e.g. for filtering – Theta, Tuple Sketches

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

Vector & Matrix Sketches

- Frequent Directions (Approximate SVD)

Sketches for Mobile (Android)

- Quantiles, PMF's and CDF's

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

Matrices (text corpora, recommendations, …)

- Covariance estimation
- Low rank approximation
- Sparsification

Vectors (text docs, images, features, …)

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

Graphs[1] (Social networks, communications, …)

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

Areas we have implemented

[1] Active area of research

<Pause>

DataSketches

Science Innovations

And Results

Innovations for Unique Counting Sketches

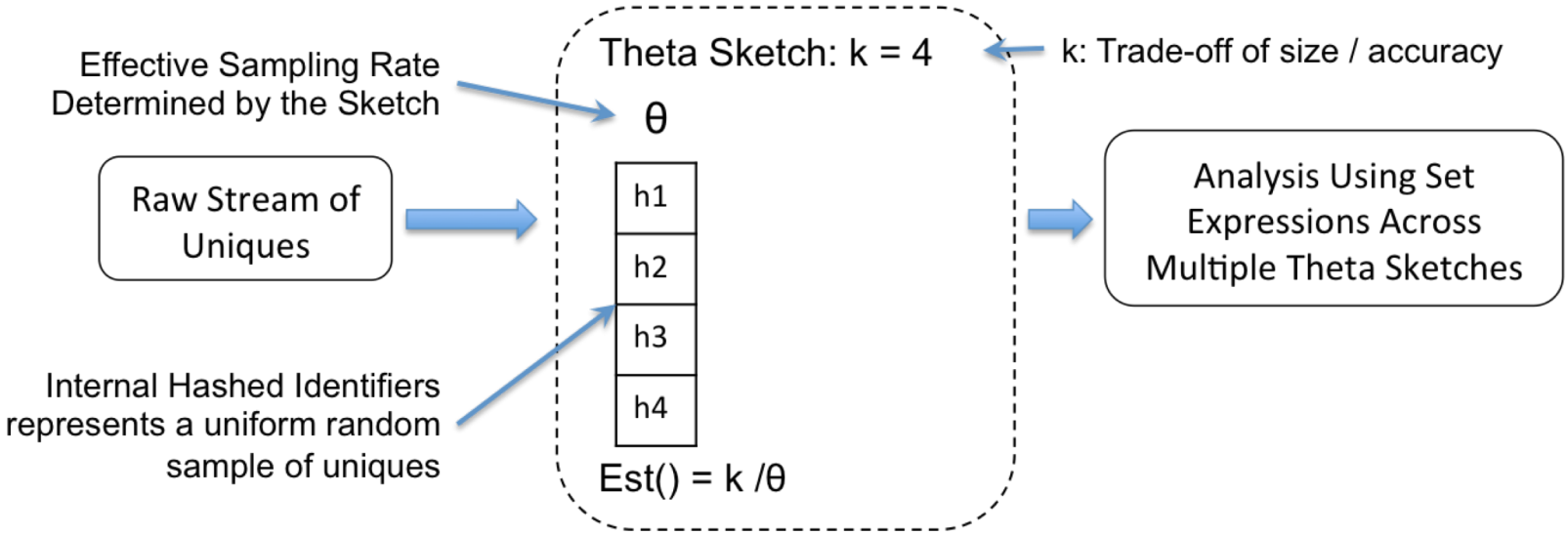
Theta Sketch Framework (TSF):

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

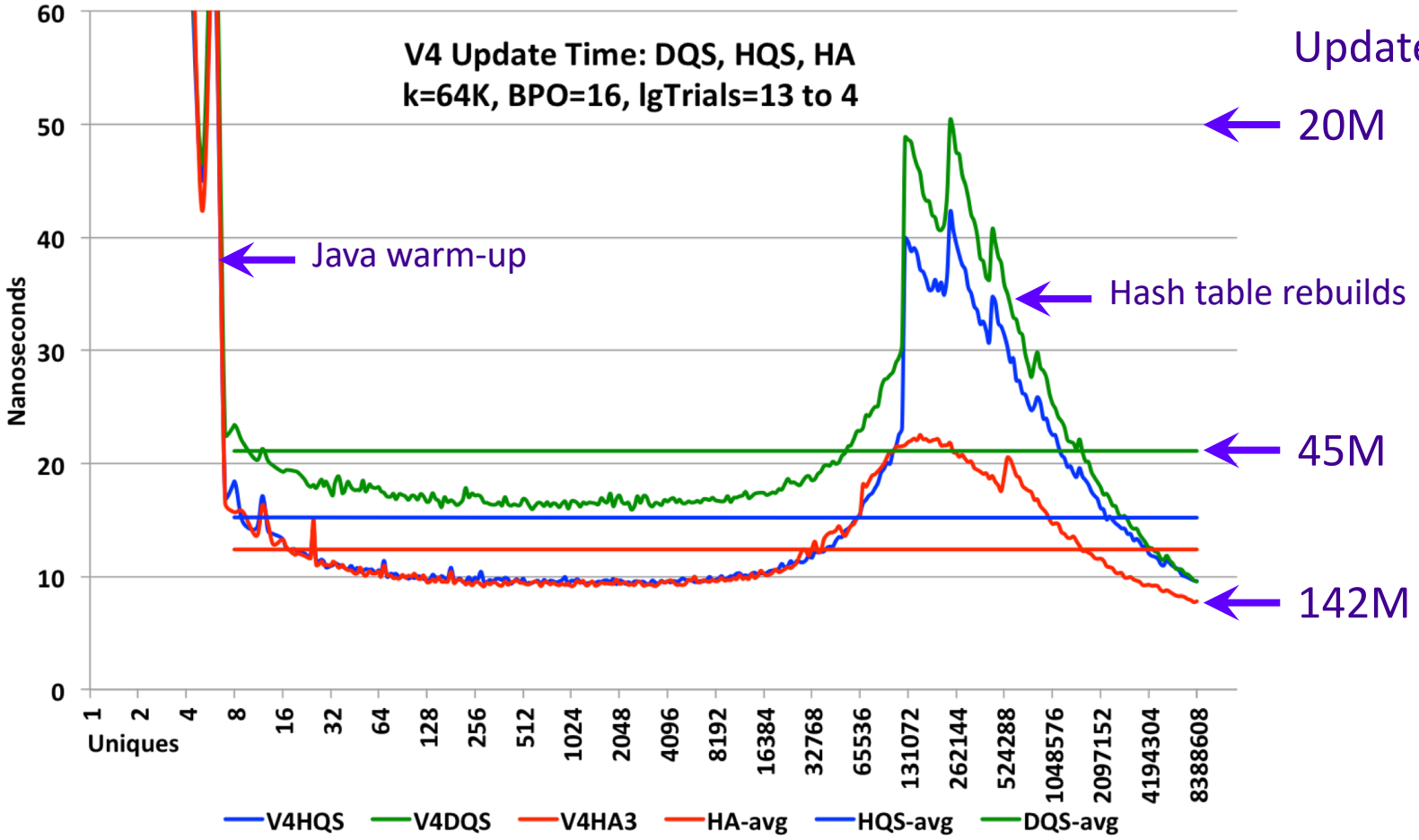
- Builds on Bar-Yossef, et al, 2002 “Counting Distinct Elements…”
- TSF applies to a whole family of sketches
- Enables simple methods for enabling set expressions and multiple-k merging
- Enables trivial up-front, (*p*KMV) sampling for tighter space usage in large systems
- Library Theta Sketch Framework: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

Theta Sketch Framework

Simple Theta Sketch

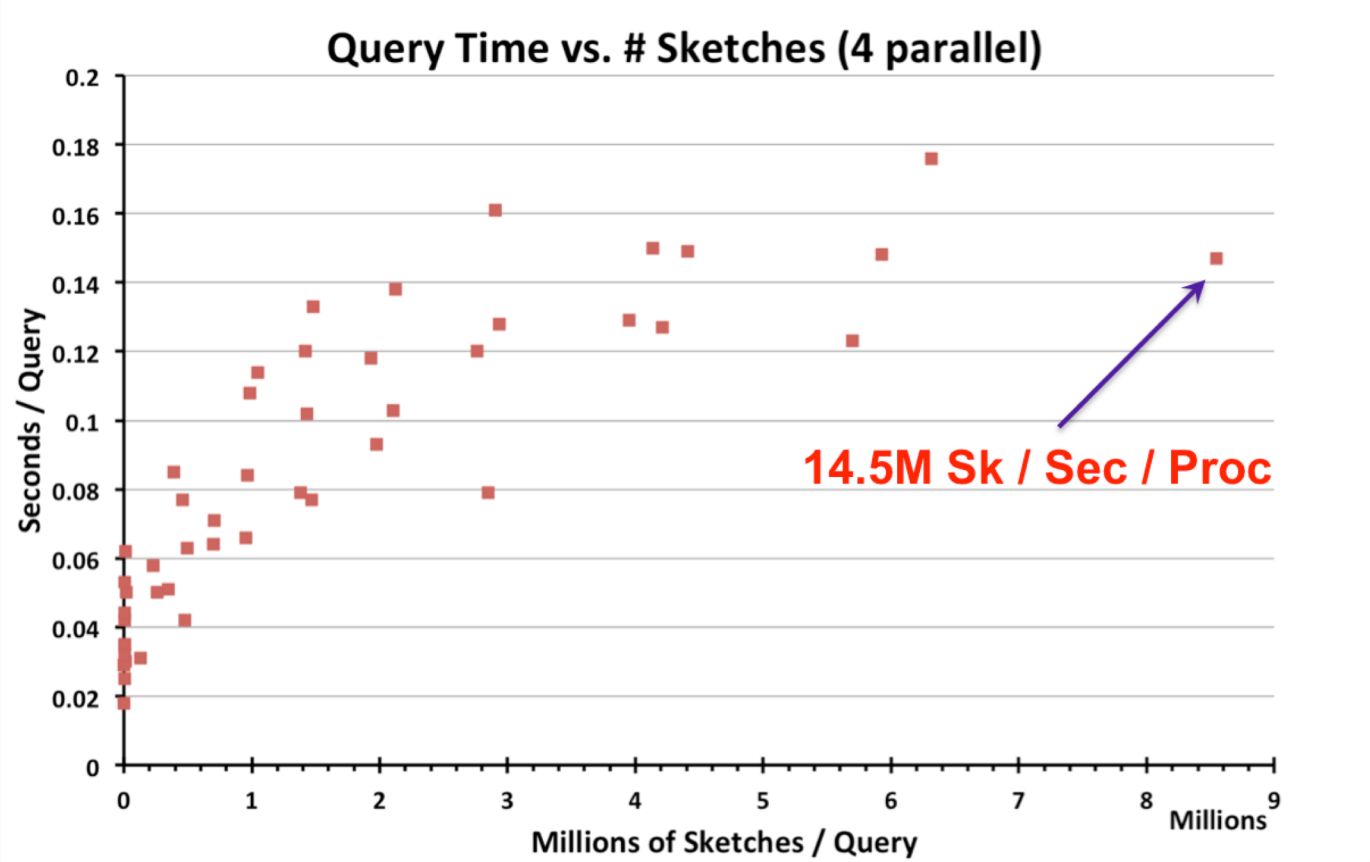


TSF: Theta Sketch Update Speed, 64K

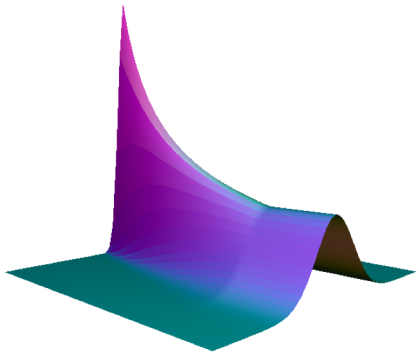


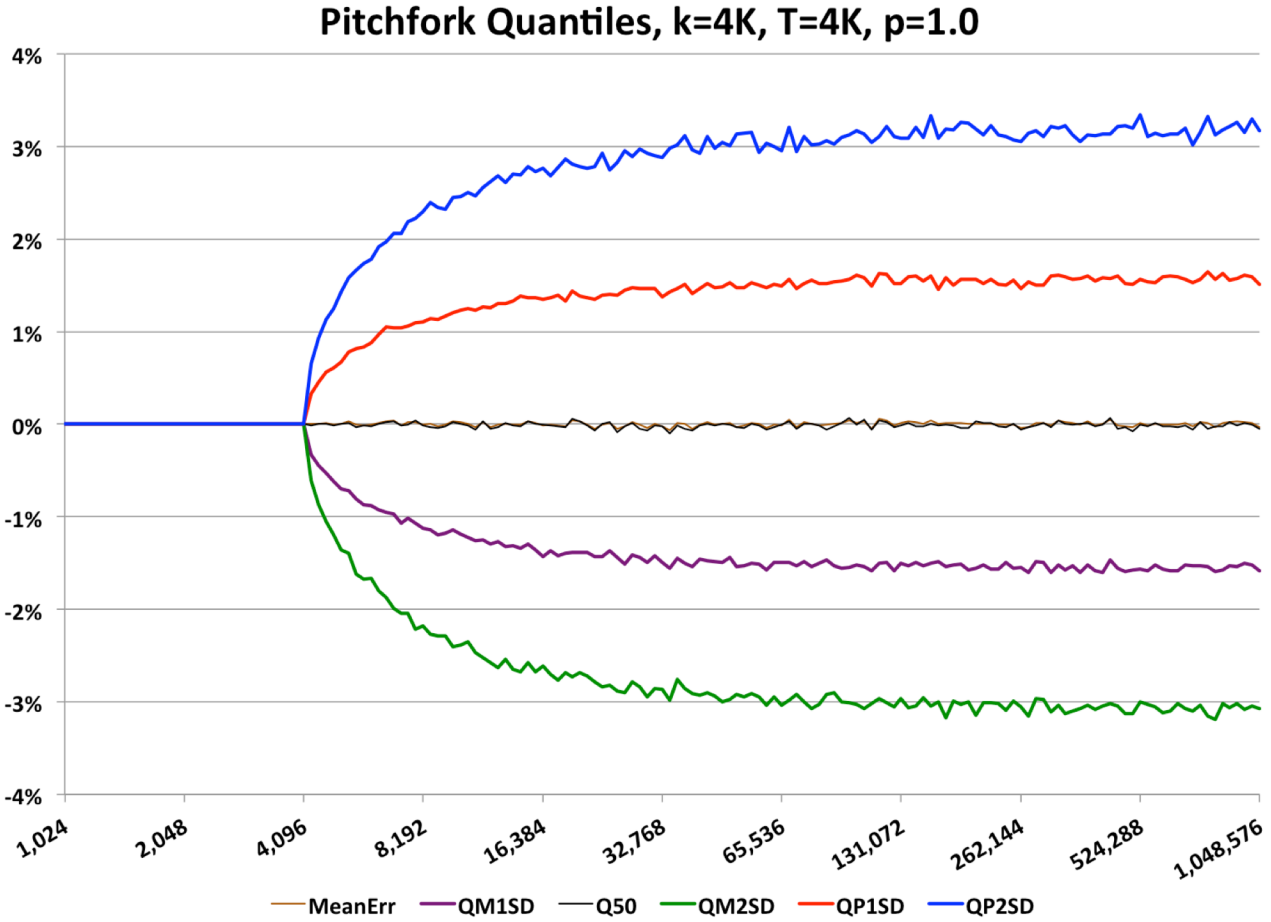
Theta Sketch Framework

Sketch Merge Time / Query



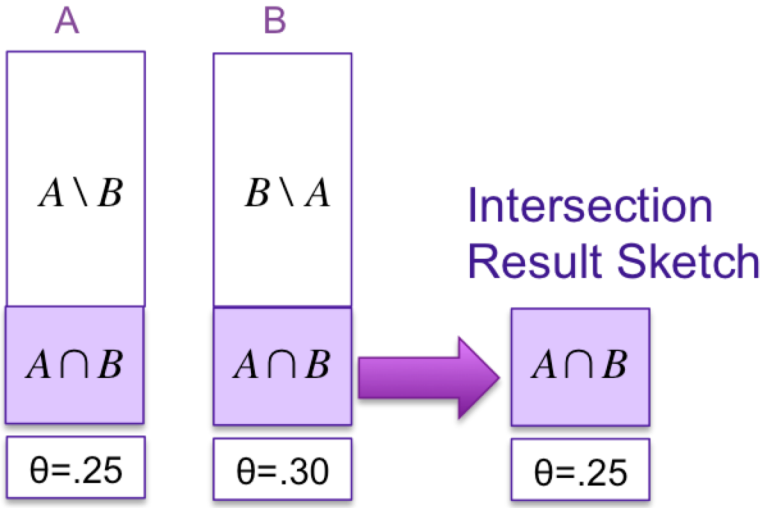
Theta Sketch Framework: Theta Sketch Accuracy





Theta Sketch Framework

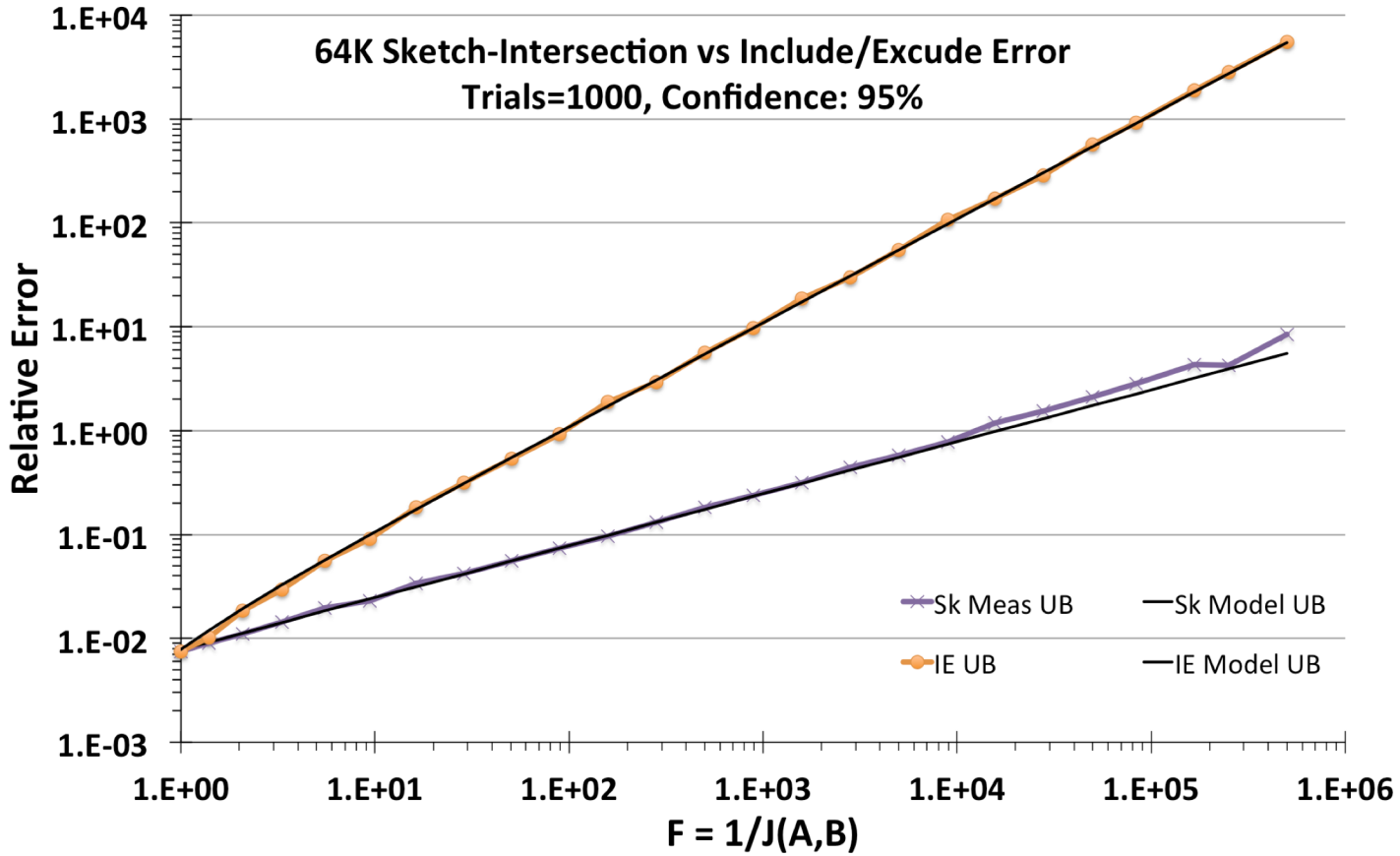
Set Expressions



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

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

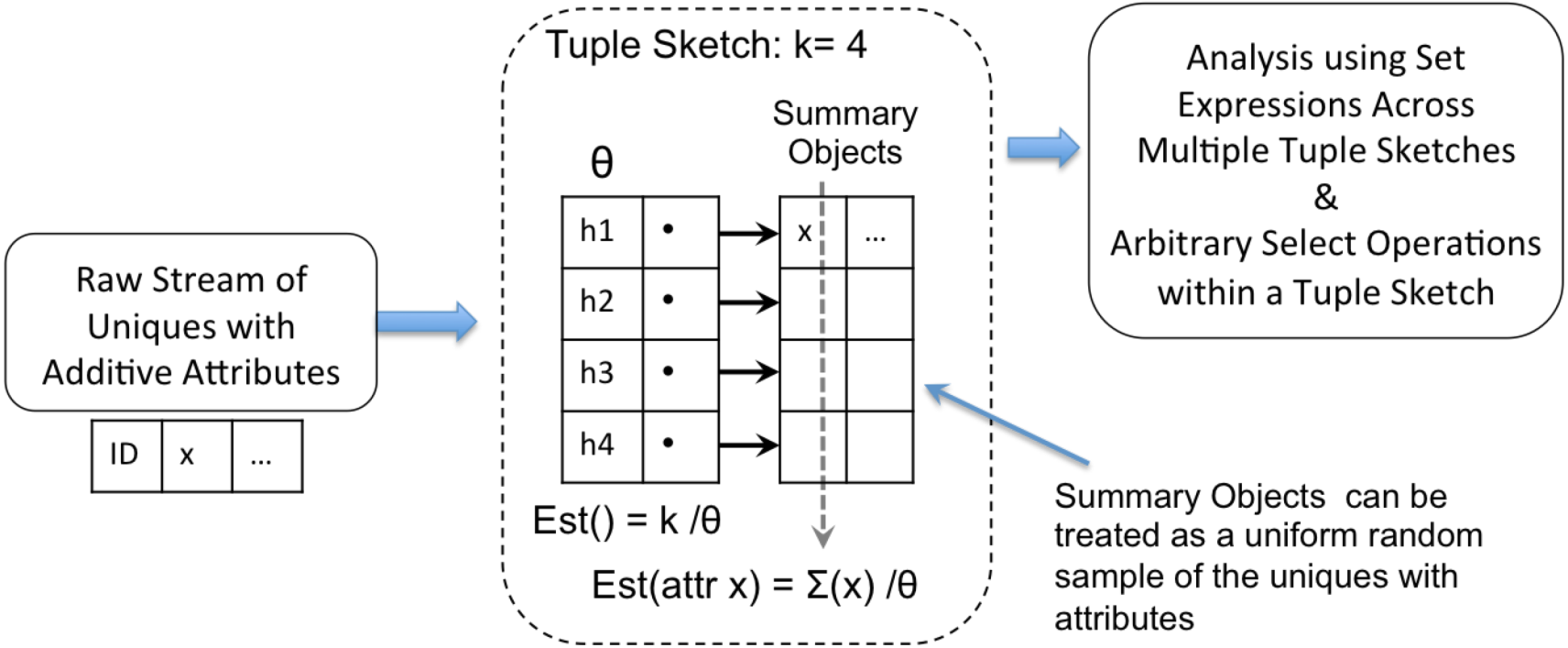
Theta Sketch Framework: Intersection Accuracy



Theta Sketch Framework: Tuple Sketch (cont.)

Tuple Sketch: Adding Attributes to the Theta Sketch

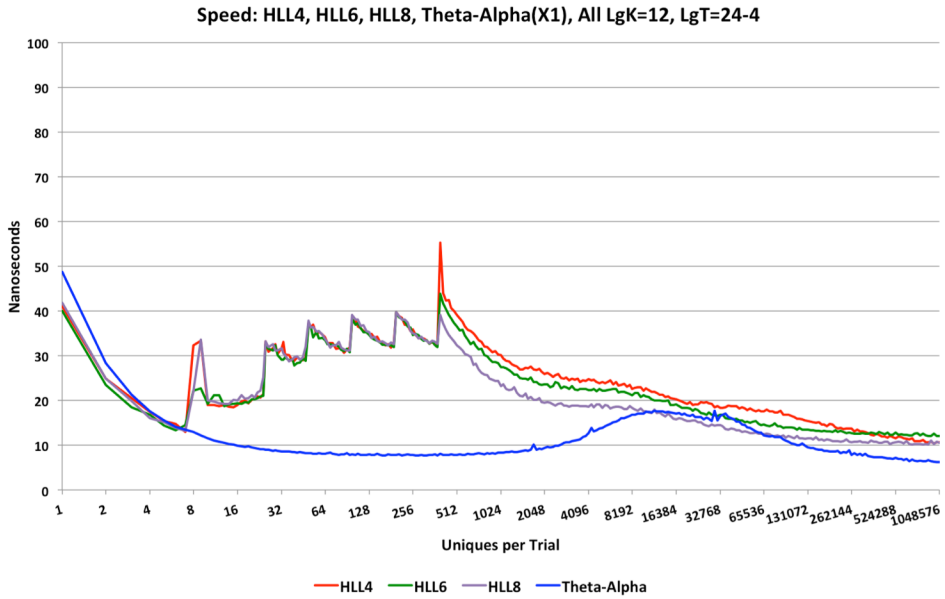
Add User-defined Attributes

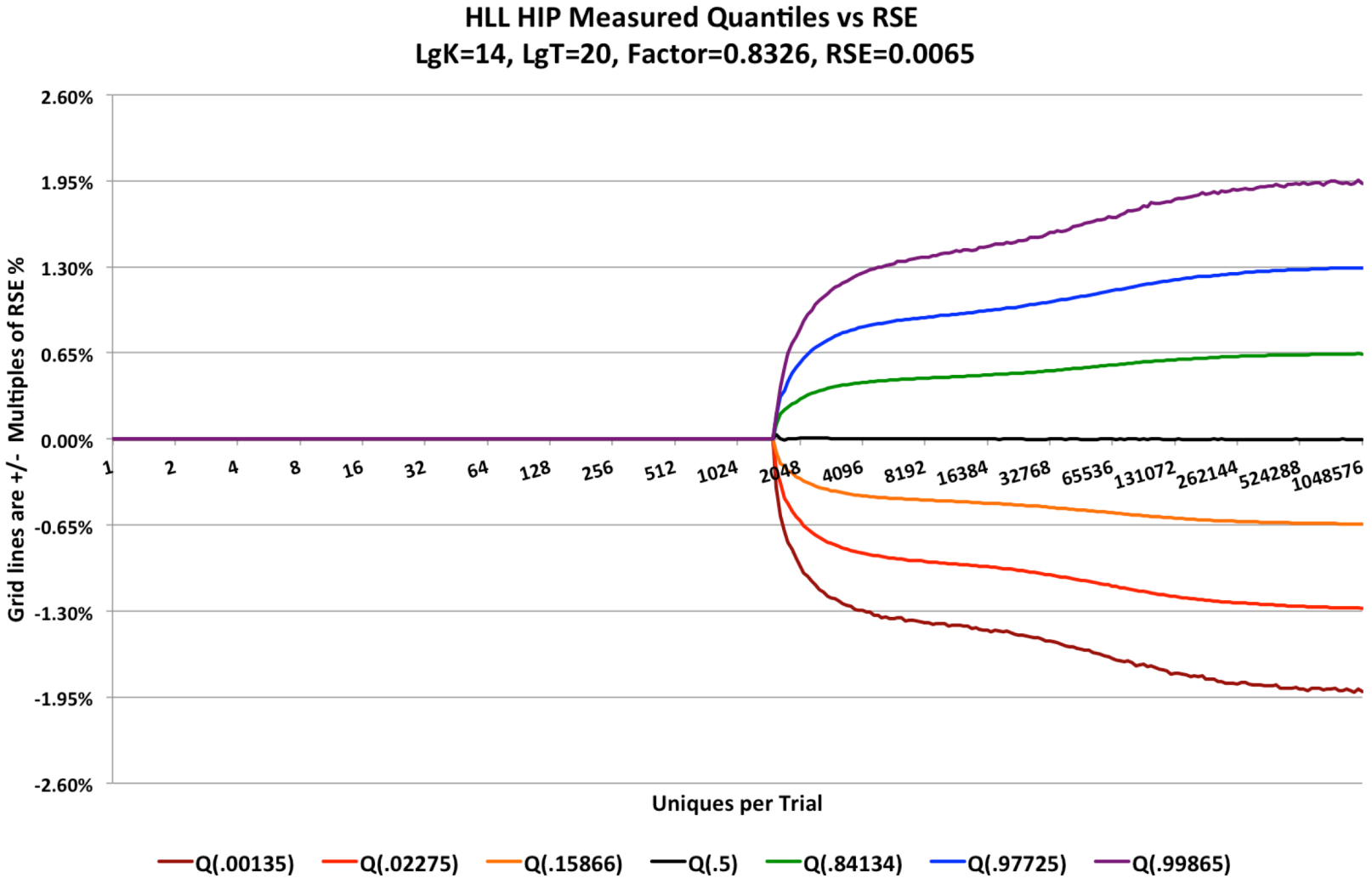


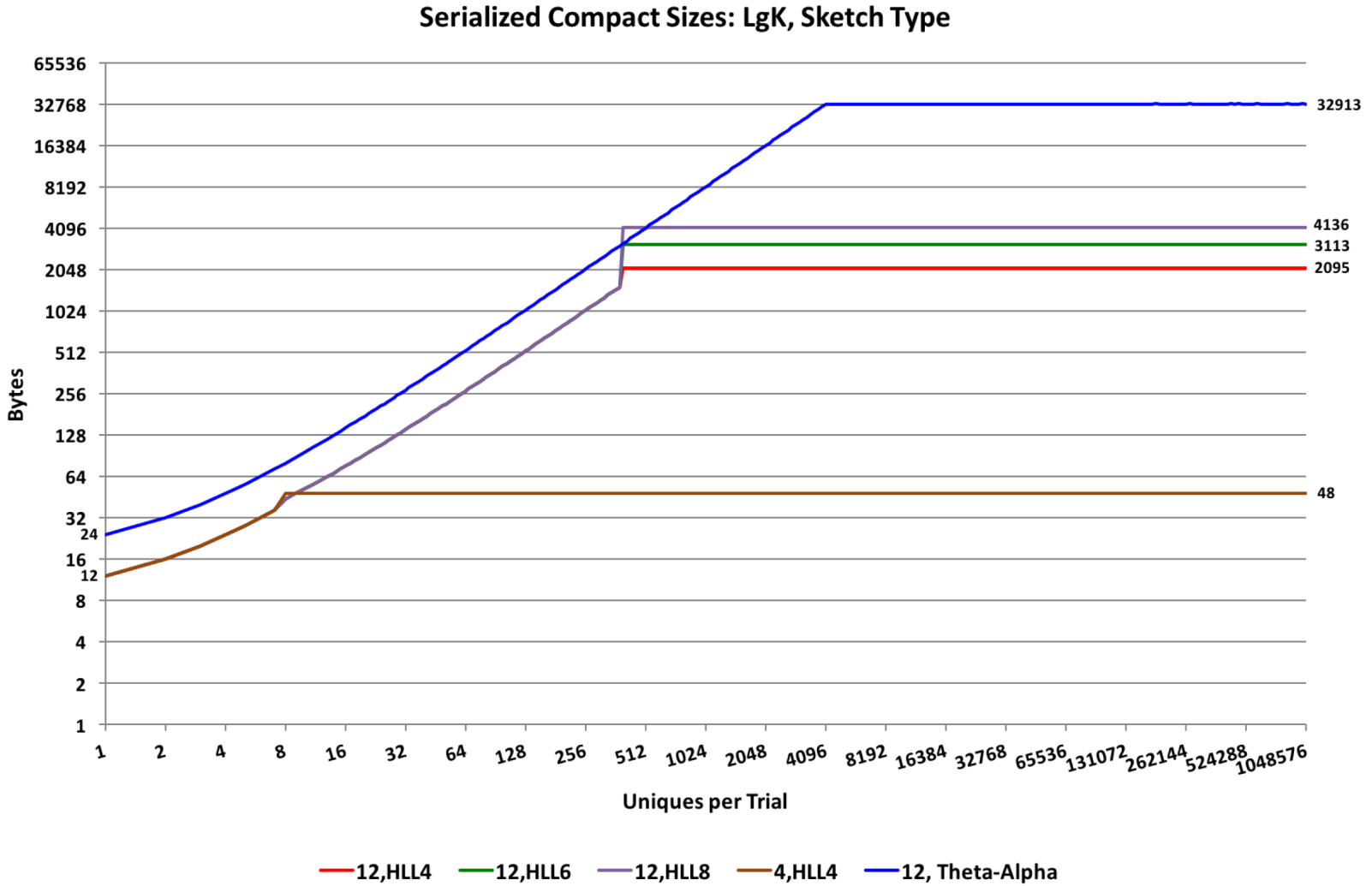
Innovations for Hyper Log Log Sketches (cont.)

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)



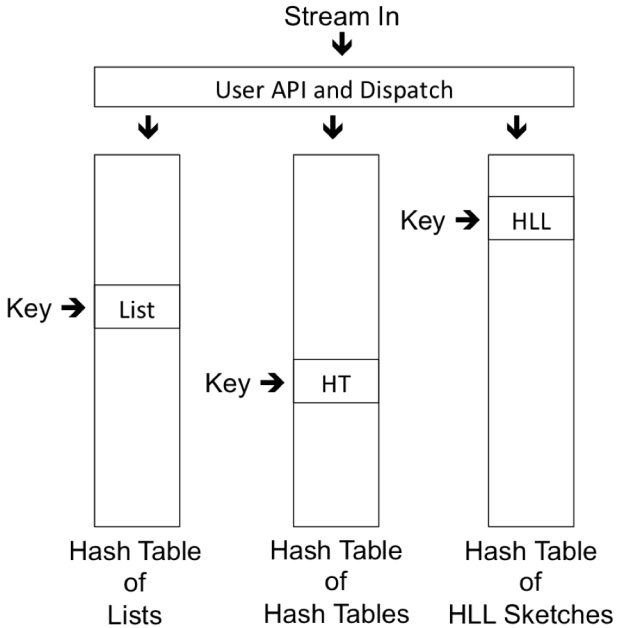

HLL HIP Measured Quantiles vs RSE
LgK=14, LgT=20, Factor=0.8326, RSE=0.0065
2.60%
1.95%
1.30%
0.65%
0.00%
-0.65%
-1.30%
-1.95%
-2.60%
Grid lines are +/- Multiples of RSE %
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
Uniques per Trial
Q(.00135)
Q(.02275)
Q(.15866)
Q(.5)
Q(.84134)
Q(.97725)
Q(.99865)

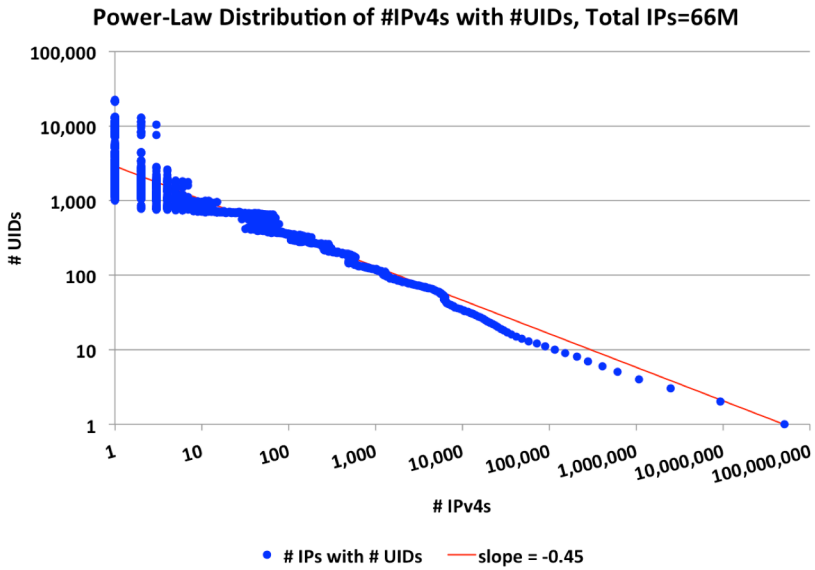

Serialized Compact Sizes: LgK, Sketch Type
65536
32768
16384
8192
4096
2048
1024
512
256
128
64
32
24
16
12
8
4
2
1
Bytes
32913
4136
3113
2095
48
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
Uniques per Trial
12,HLL4
12,HLL6
12,HLL8
4,HLL4
12, Theta-Alpha

Innovations for Hyper Log Log Sketches (cont.)

UniqueCountMap (streams of millions of *K,V* pairs)

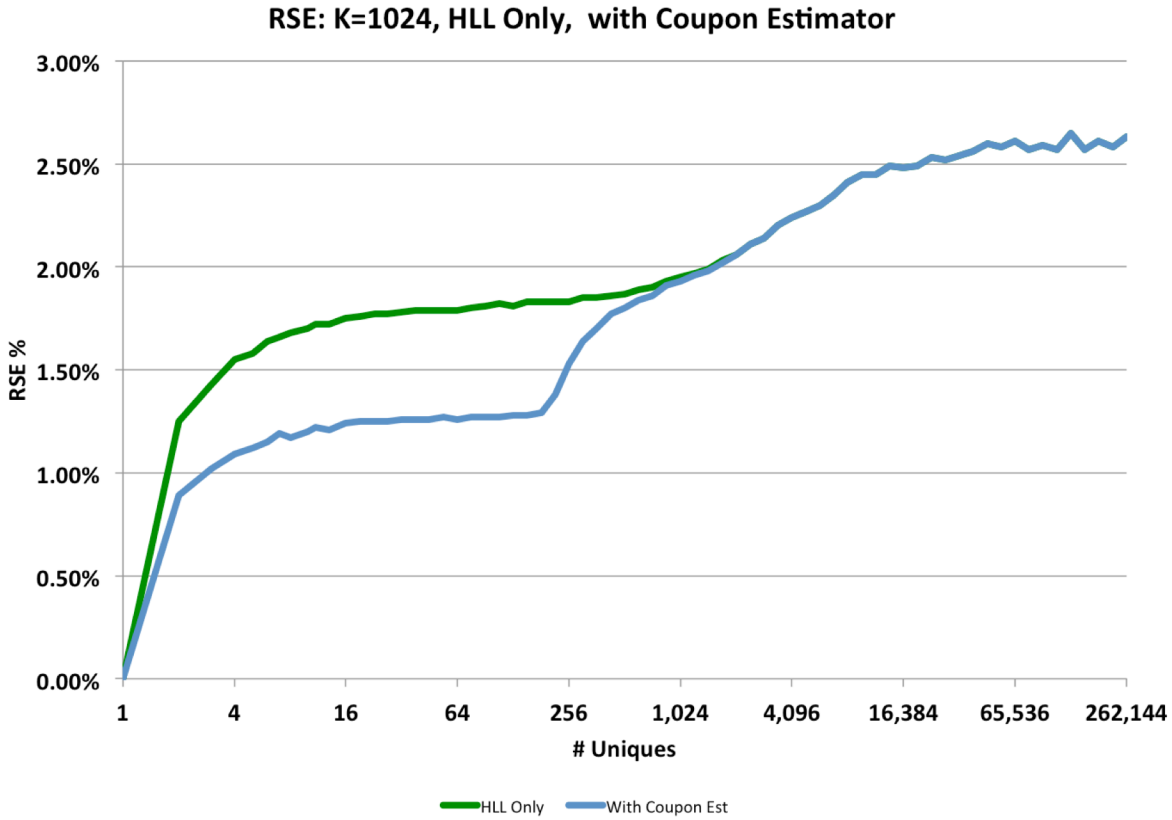
- Real-time cardinality estimates of *V per Key*
- Highly space-efficient (100M 4-byte keys require ~1.3GB: ~9 bytes / K for card.
- Separate data structures manage different phases of sketching process
- Simple-to-use API





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)

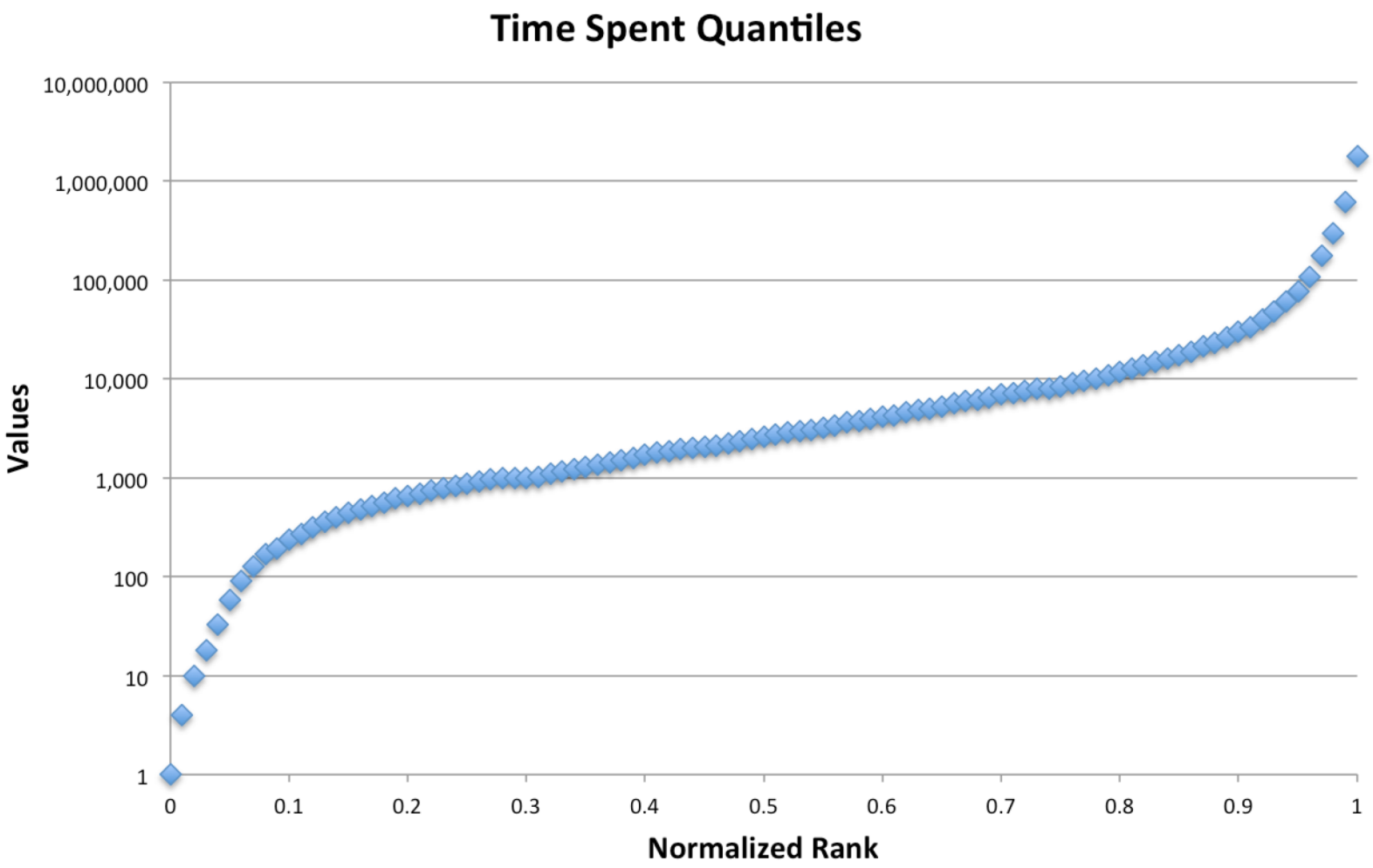
- Builds on Flajolet-Martin 1985 “Probabilistic Counting Algorithms For Data Base Applications”
- Three new estimators: all more accurate than original paper estimators
- More accurate per bit-of-entropy than Flajolet’s 2008 HLL sketches
- Simultaneously wins on all three dimensions of the time/space/accuracy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

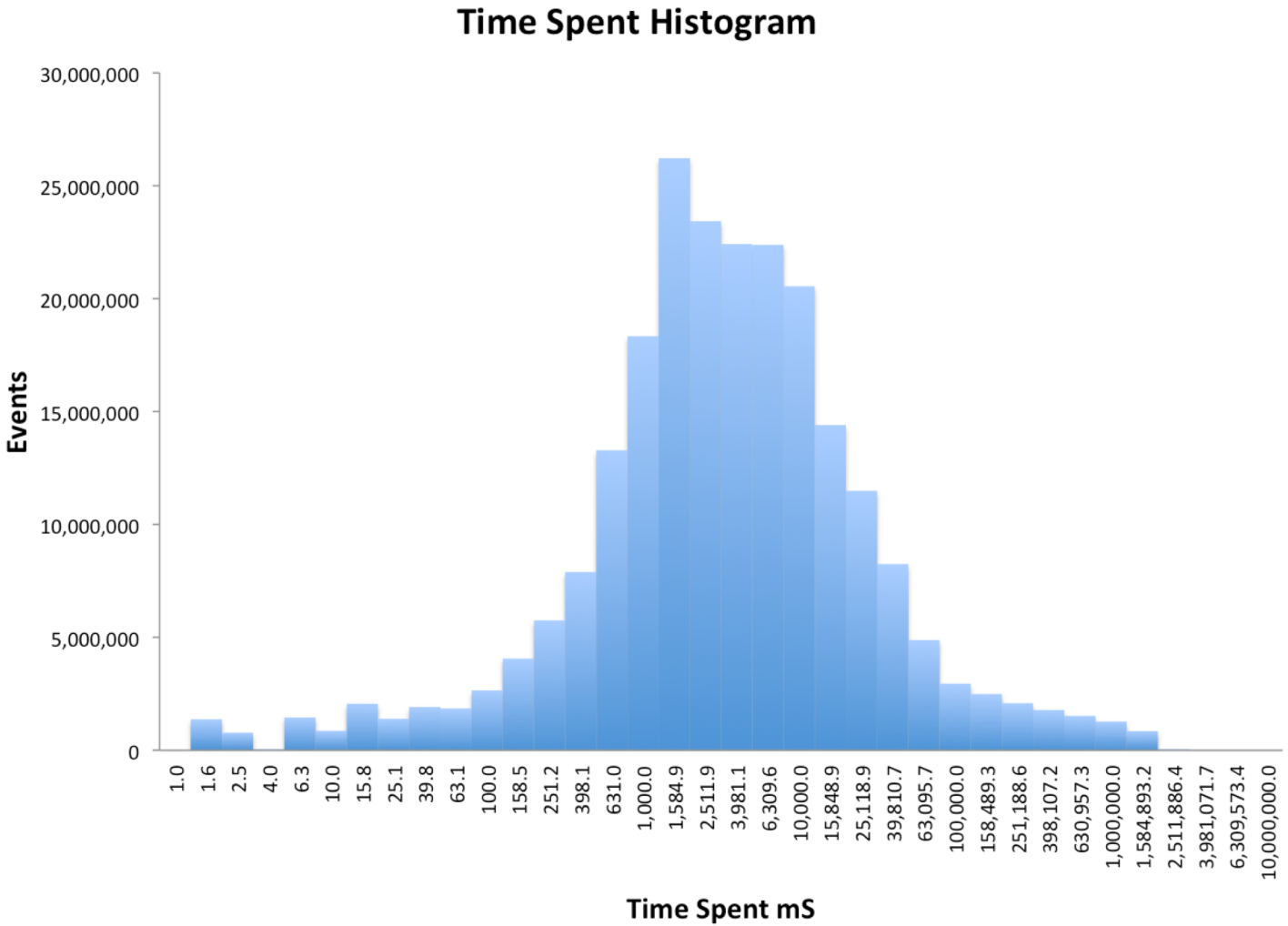
Innovations for Quantiles / Histogram Sketches

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
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
Values
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Rank


Time Spent Histogram
Events
30,000,000
25,000,000
20,000,000
15,000,000
10,000,000
5,000,000
0
1.0
1.6
2.5
4.0
6.3
10.0
15.8
25.1
39.8
63.1
100.0
158.5
251.2
398.1
631.0
1,000.0
1,584.9
2,511.9
3,981.1
6,309.6
10,000.0
15,848.9
25,118.9
39,810.7
63,095.7
100,000.0
158,489.3
251,188.6
398,107.2
630,957.3
1,000,000.0
1,584,893.2
2,511,886.4
3,981,071.7
6,309,573.4
10,000,000.0
Time Spent mS

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

Mina Ghashami, E. Liberty, J. Phillips: Efficient Frequent Directions Algorithm for Sparse Matrices, *ACM KDD* 2016

- 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

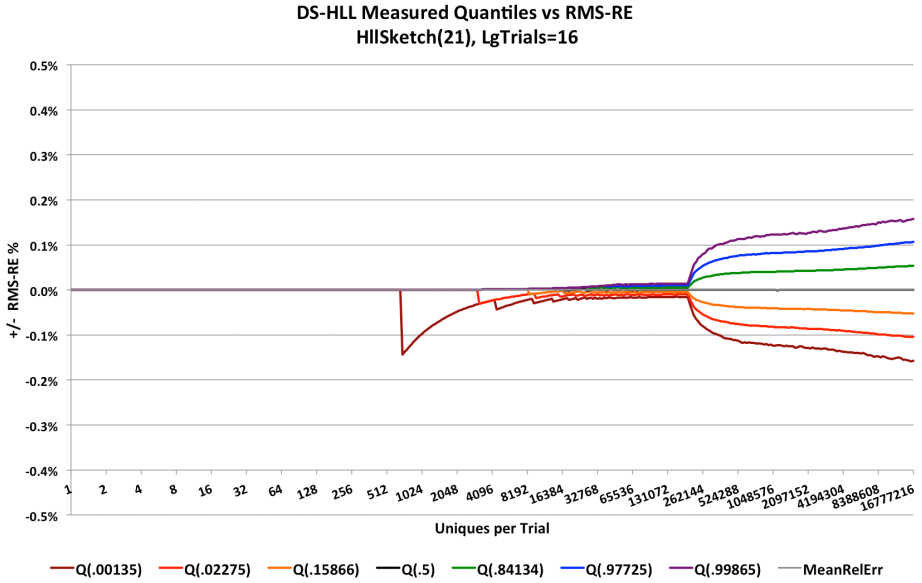
<Pause>

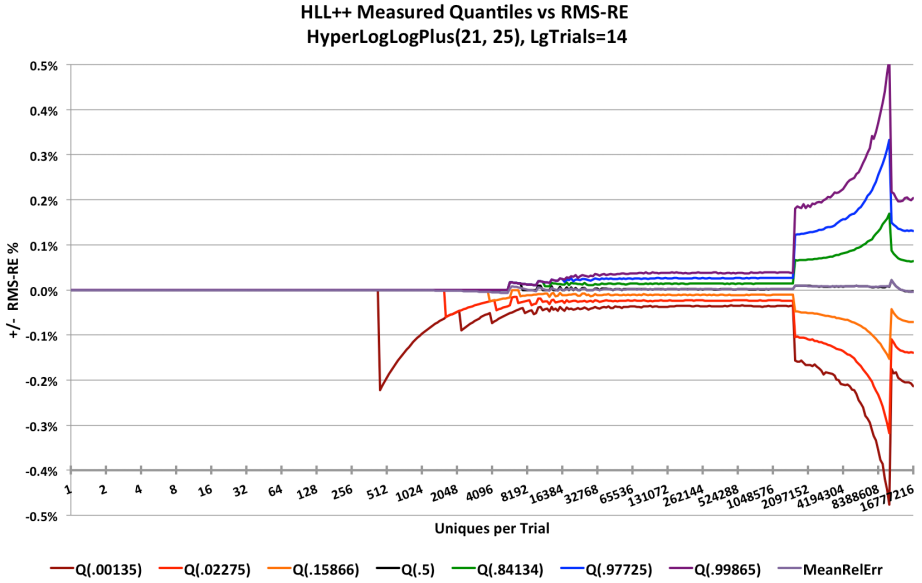
DataSketches HLL Sketch

versus

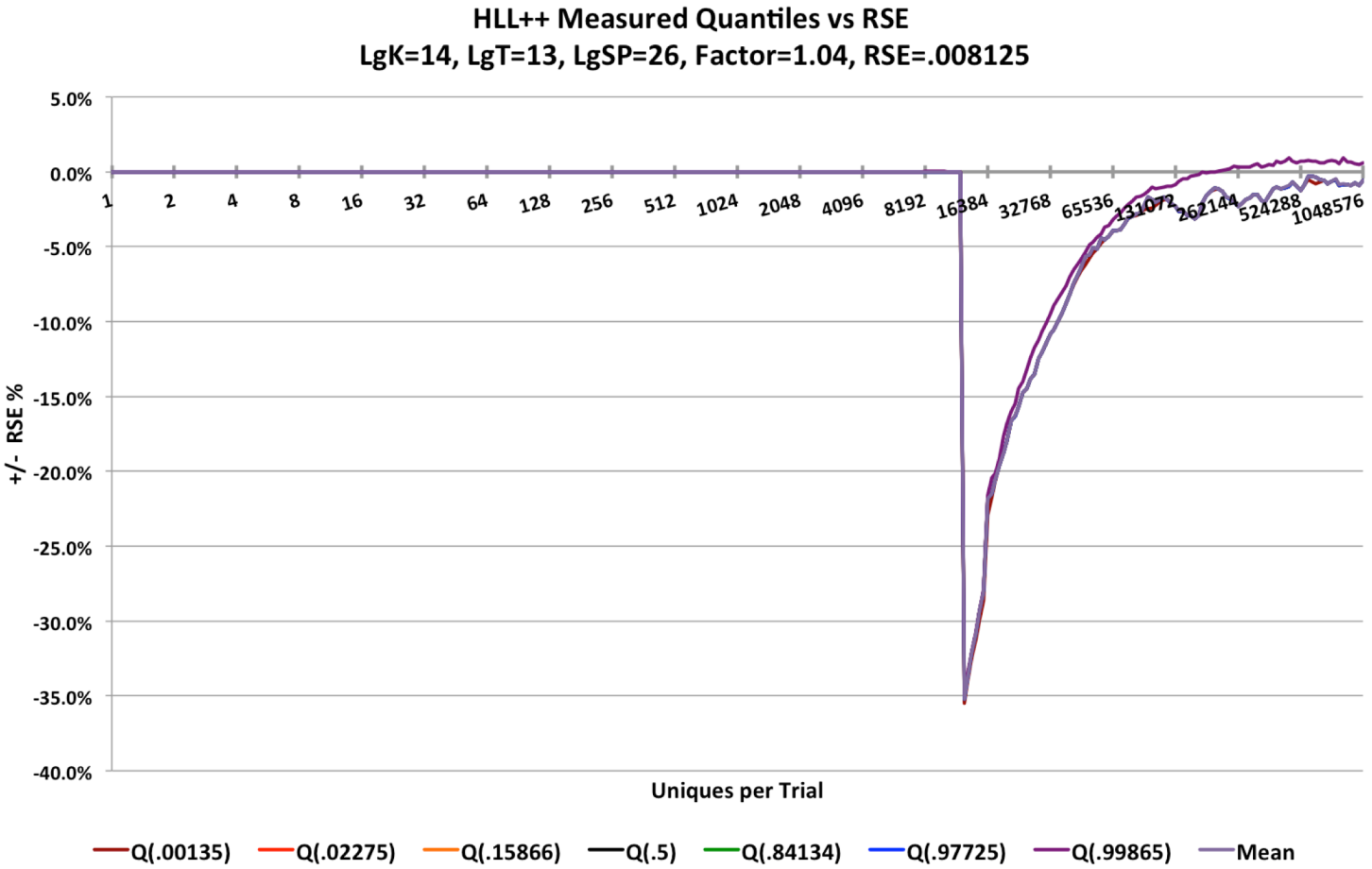
Clearspring implementation of Google HLL++

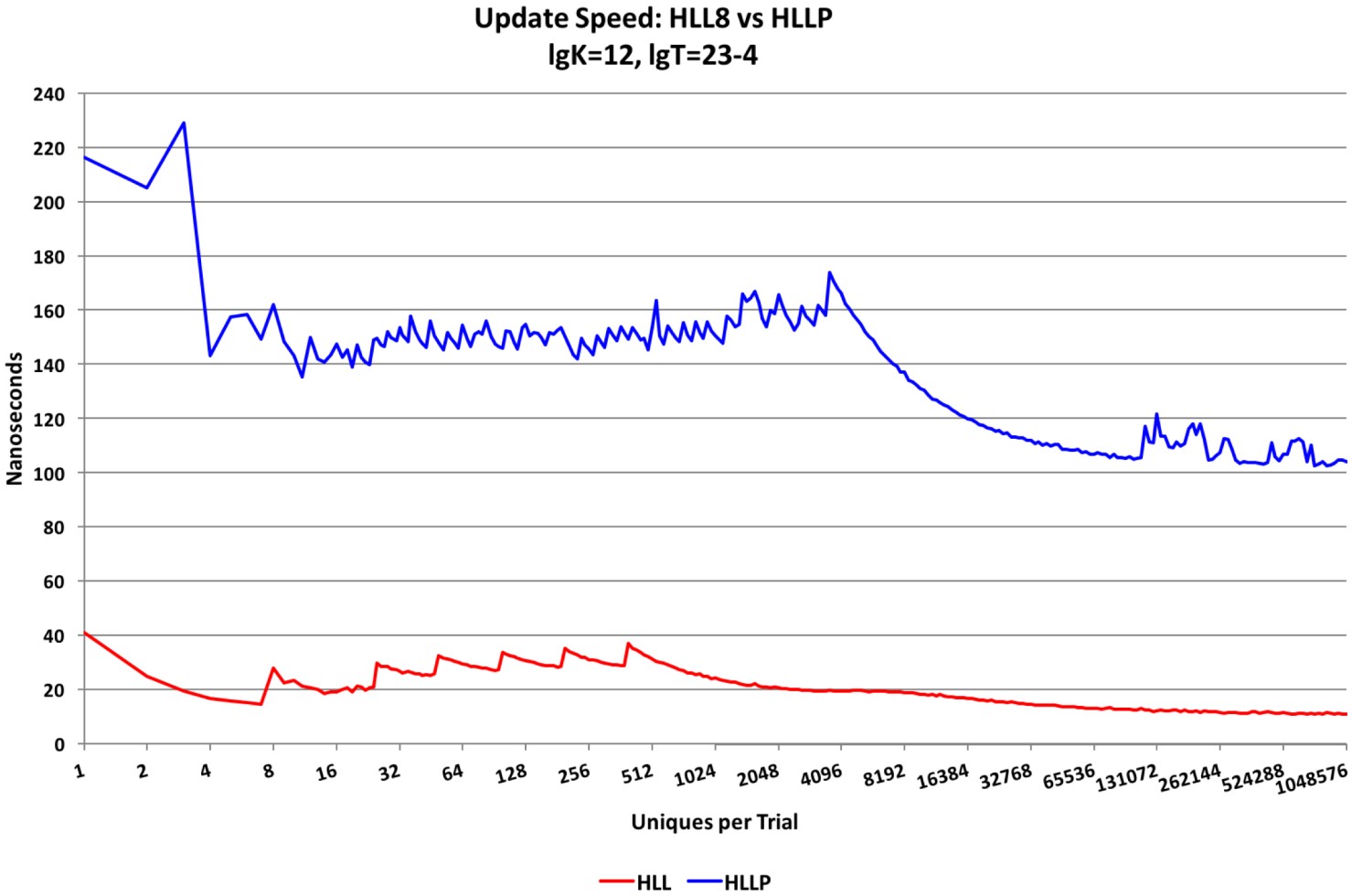
DataSketches HllSketch vs Clearspring HyperLogLogPlus (Google HLL++)

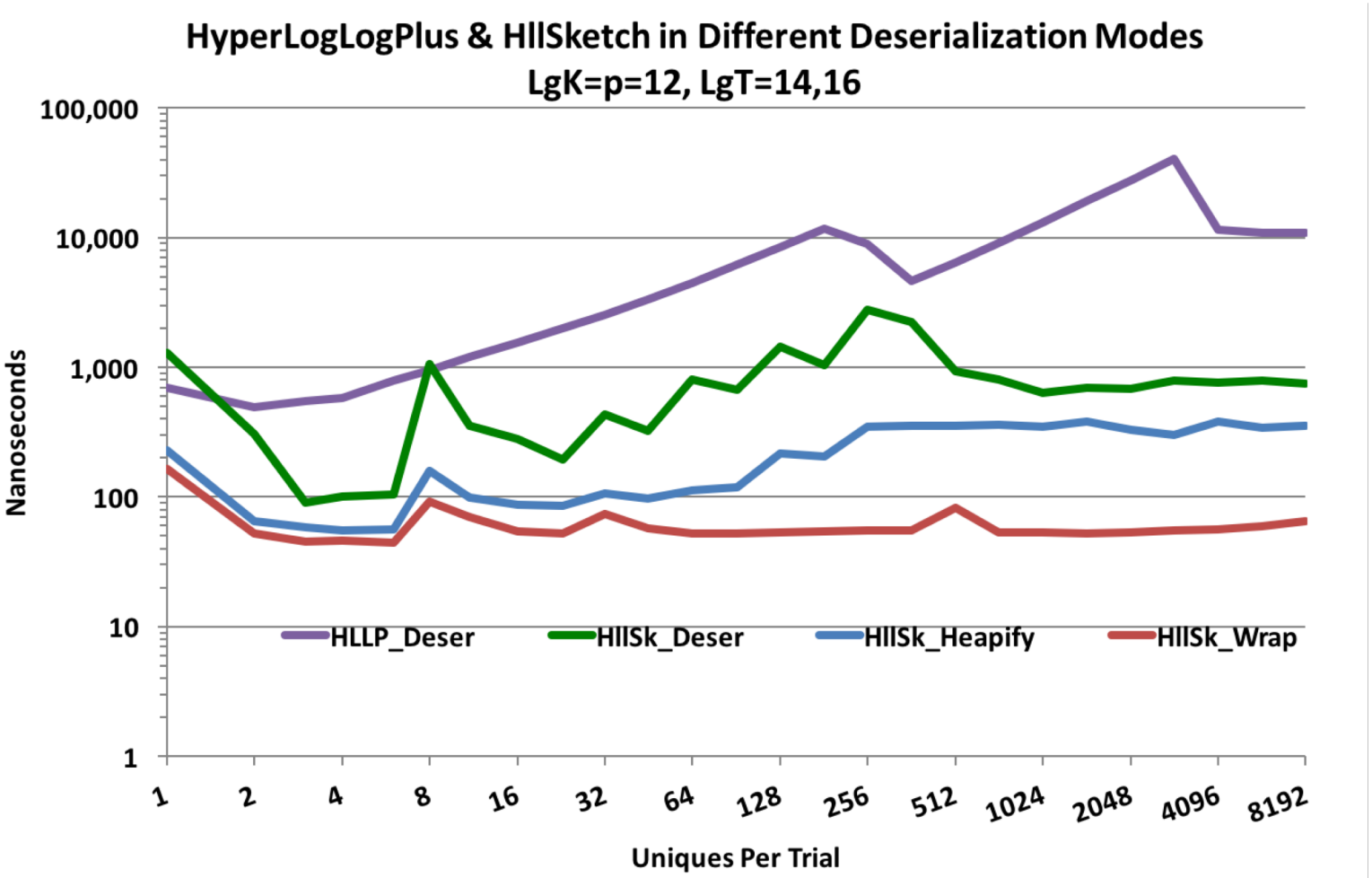




Catastrophic Failure of CS Google HLL++




Update Speed: HLL8 vs HLLP
lgK=12, lgT=23-4
Nanoseconds
240
220
200
180
160
140
120
100
80
60
40
20
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
Uniques per Trial
HLL
HLLP


HyperLogLogPlus & HllSketch in Different Deserialization Modes
LgK=p=12, LgT=14,16
100,000
10,000
1,000
100
10
1
Nanoseconds
HLLP_Deser
HllSk_Deser
HllSk_Heapify
HllSk_Wrap
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
Uniques Per Trial

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 σ)	0.35% (2 σ), 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
Stage 1: • Read Raw Data • -> Hourly Tables	Create Table1: Group By {site, hour, id1} Create Table2: Group by {site, hour, id2}
Intermediate Size	33.4 GB 1 Month of Hourly
Stage 2a: • Read Hourly Tables • Count Uniques	Group By {site, hour}, Count Id1 Group By {site, hour), Count Id2
Stage 2b: • -> Hourly Report	Join: {site, hour, count(id1), count(id2)}
Total CPU Time	1.39M Sec

Sketches Cube: For All Reports

Sub-Task	Data Stored
Stage 1: • Read Raw Data • -> Data Cube	Create Sketches Cube: By Dim Combination {site, hour, sketch(id1), sketch(id2)}
Intermediate Size	1.1 GB
Stage 2 • Read Data Cube • Produce Hourly Report	Merge Sketches across Chosen Dimensions
Total CPU Time	1.06M Sec

Case Study 1: Daily Rollups

Exact: For Daily Reports and Basis for Weekly and Monthly

Sub-Task	Data Stored
Stage 1: • Read Hourly Intermediates • -> Daily Tables	Create Table1: Group By {site, day, id1} Create Table2: Group by {site, day, id2}
Intermediate Size	16.0 GB just for Daily
Stage 2a: • Read Daily Intermediates • Count Uniques	Group By {site, day}, Count Id1 Group By {site, day), Count Id2
Stage 2b: • Produce Hourly Report	Join: {site, day, count(id1), count(id2)}
Total CPU Time	96,300 sec

Sketches Cube: For All Reports

Sub-Task	Data Stored
Stage 1: • Read Data Cube • -> Produce Daily Report	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

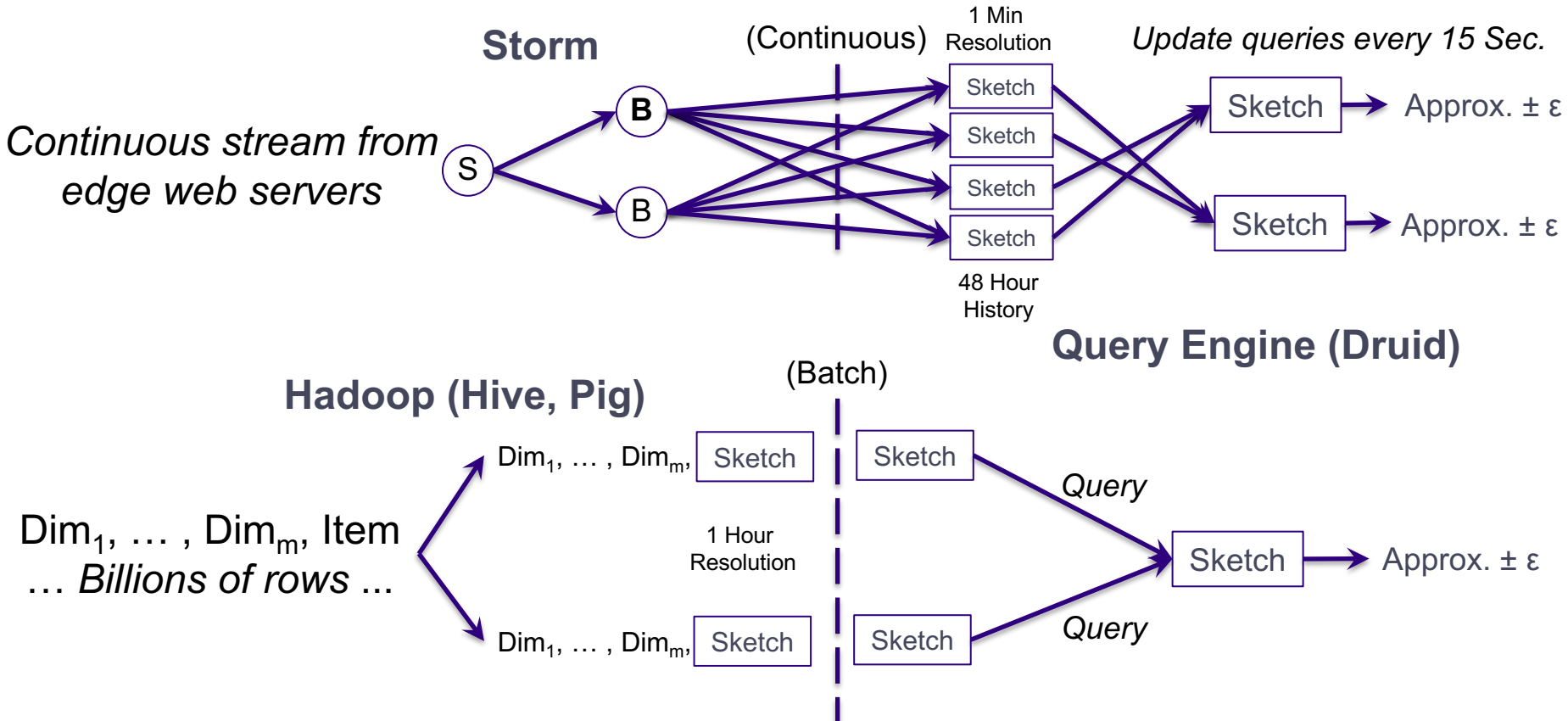
Sub-Task	Data Stored
Stage 1: • Read Data Cube • -> Produce Weekly or Monthly Reports	N/A
Total CPU Time	Week: 424 Sec Month: 466 Sec

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	**Hadoop:** complete brute-force, pre-computation of uniques **HBase:** Query Engine	**Hadoop:** w/ Sketches **Storm:** Real-Time w/ Sketches **Druid:** Query Engine w/ Sketches
Cost (Virtual Core-Sec) (VCS)	**Mo Total: ~80B vcs**	**Mo. Total: ~20B vcs** **BIG WIN : Eco Friendly! Lower $**
Result Freshness	**Daily: 2 to 8 hours** **Weekly: ~3 days** **Real-time Not Feasible**	**15 seconds!**
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
Usage
Retention
Active Devices
Active Users
Audience
Events Summary
Events Details

All Apps (337)

Yahoo Mail++ - Production | Yahoo Mail++ - Production

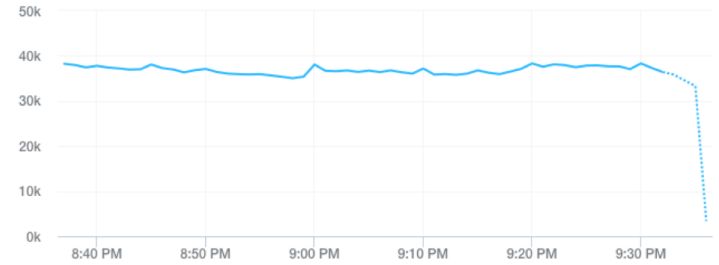
Add Filter

as of 9:36:20 PM

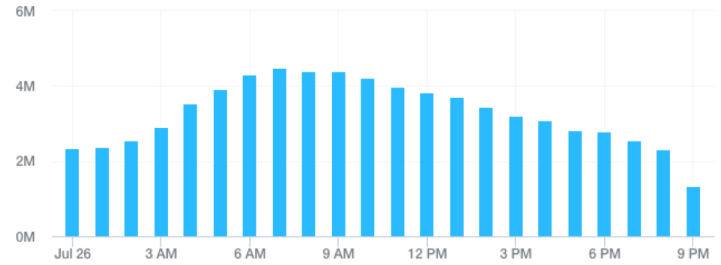
Real Time Dashboard

ACTIVE DEVICES LAST 24 HOURS	SESSIONS TODAY	SESSIONS LAST 24 HOURS
15,365,820	72,541,482	77,724,619

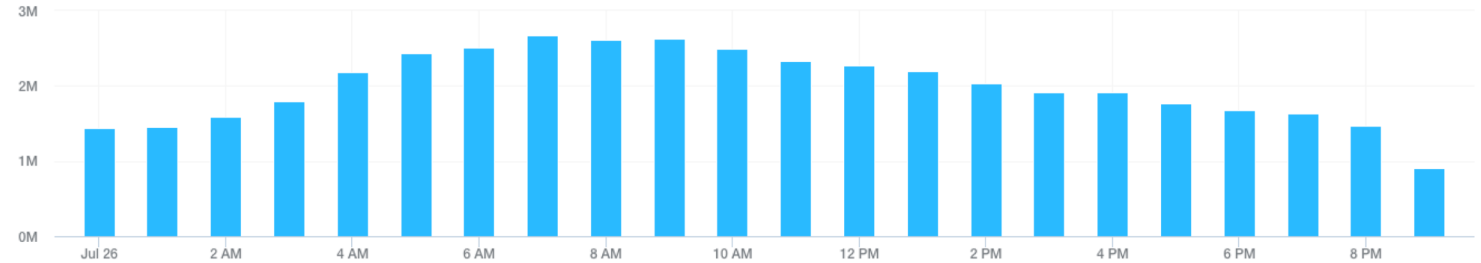
SESSIONS - MINUTE



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

Filtering coming soon

Jun 27, 2016 - Jul 26, 2016

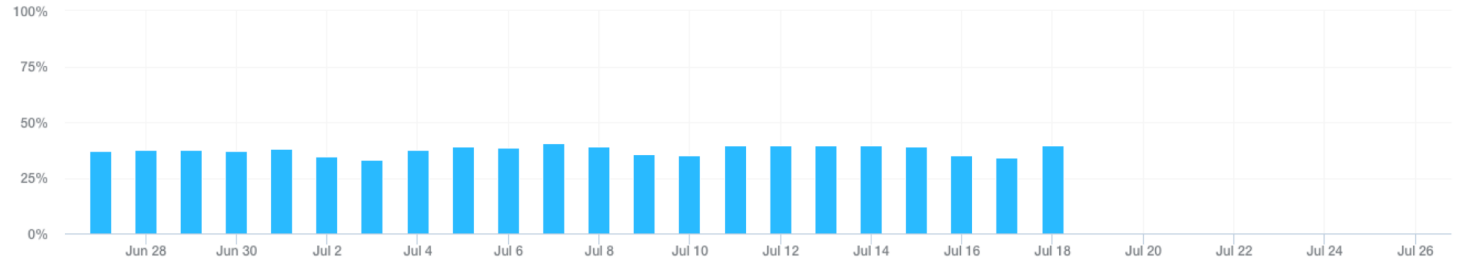
Retention Dashboard

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