DataSketches

A Required Toolkit for the Analysis of Big Data

Lee Rhodes, Architect, Distinguished
Jon Malkin, Research Scientist, Sr.
Alexander Saydakov, Software Engineer, Sr
Kevin Lang, Chief Scientist

YAHOO!

The Challenge

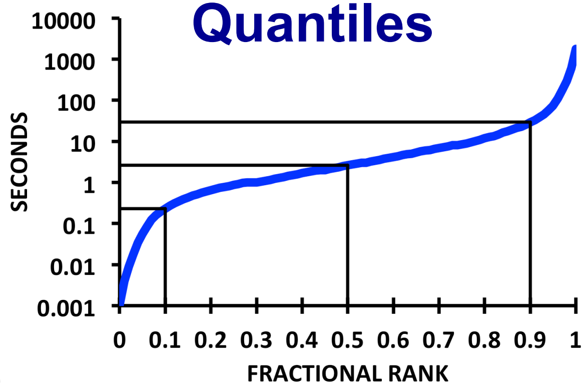
Web Site Logs

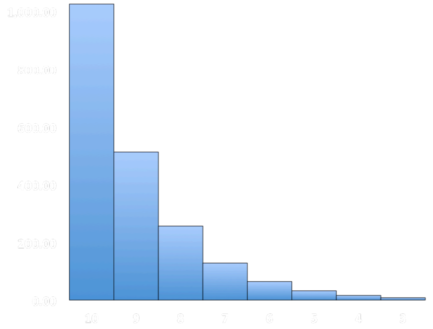
Time Stamp	User ID	Device ID	Site	Time Spent Sec	Items Viewed
9:00 AM	U1	D1	Apps	59	5
9:30 AM	U2	D2	Apps	179	15
10:00 AM	U3	D3	Music	29	3
1:00 PM	U1	D4	Music	89	10
Billions of rows …					

YAHOO!

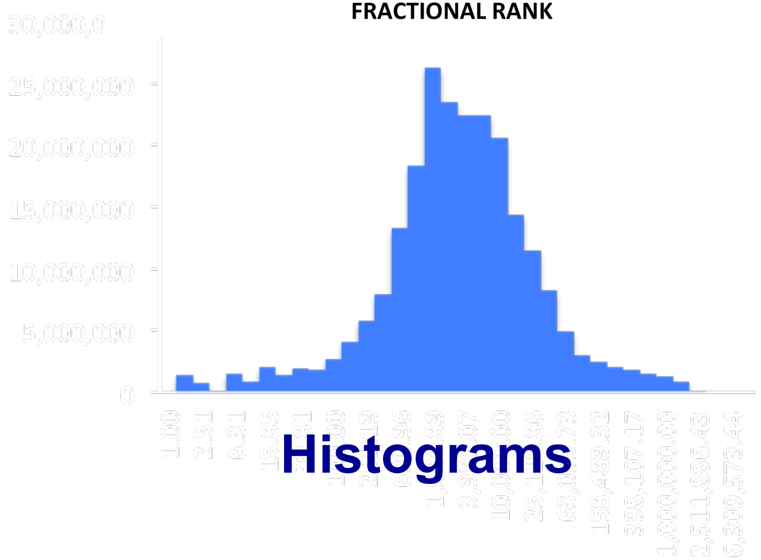
Some Very Common Queries …

Unique Users





Most Frequent Occurrences



Histograms

YAHOO!

Why are ***these*** operations so difficult with Big Data?

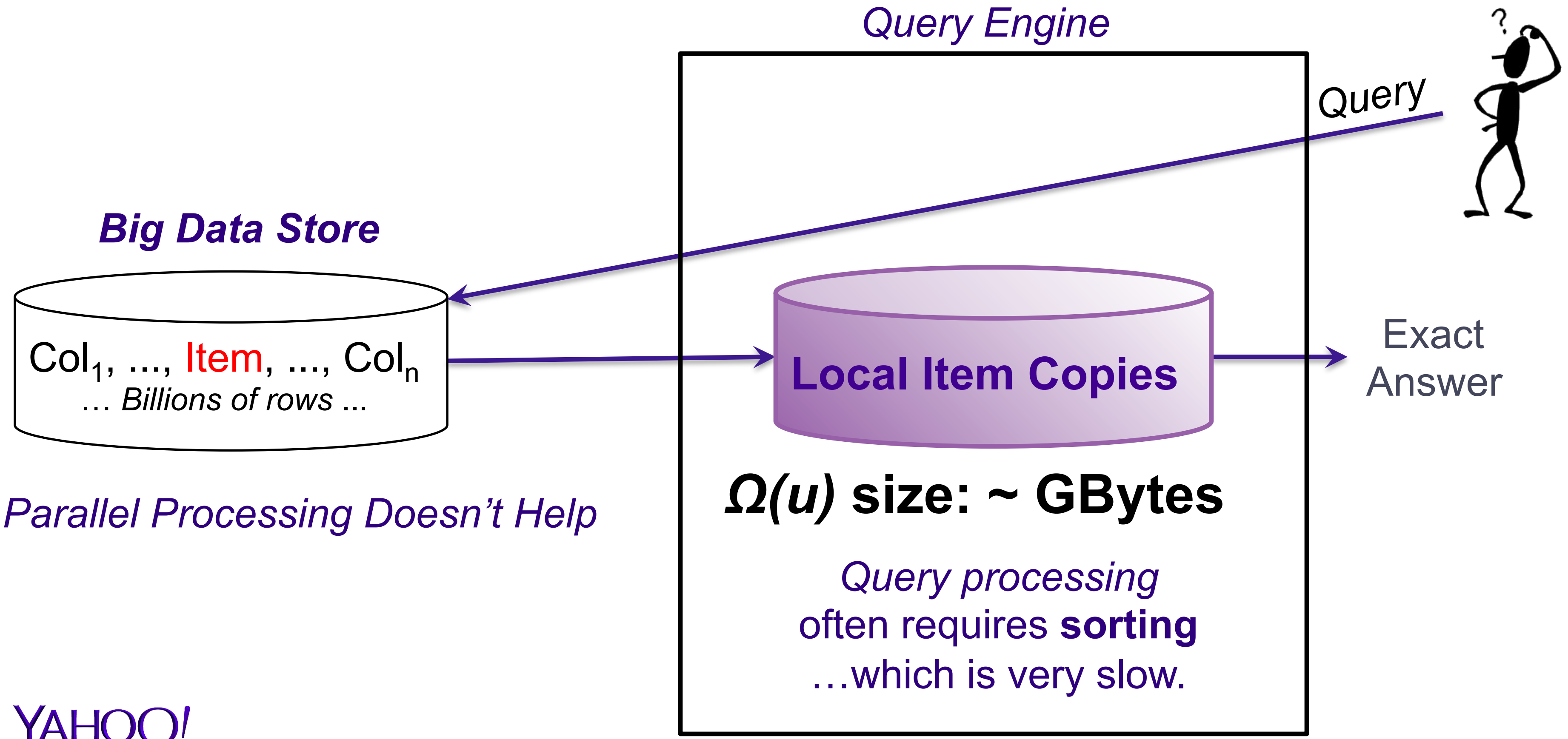
Mathematically Proven Lower Space Bound = $\Omega(u)$:

With no prior knowledge of the data …
There does not exist an algorithm that can produce an exact result with

$$\text{Space} < C * \#\textit{Unique Items}$$

C = A constant factor ≥ 1

Exact Approach From Stored Data



Parallel Processing Doesn't Help

$\Omega(u)$ size: ~ GBytes

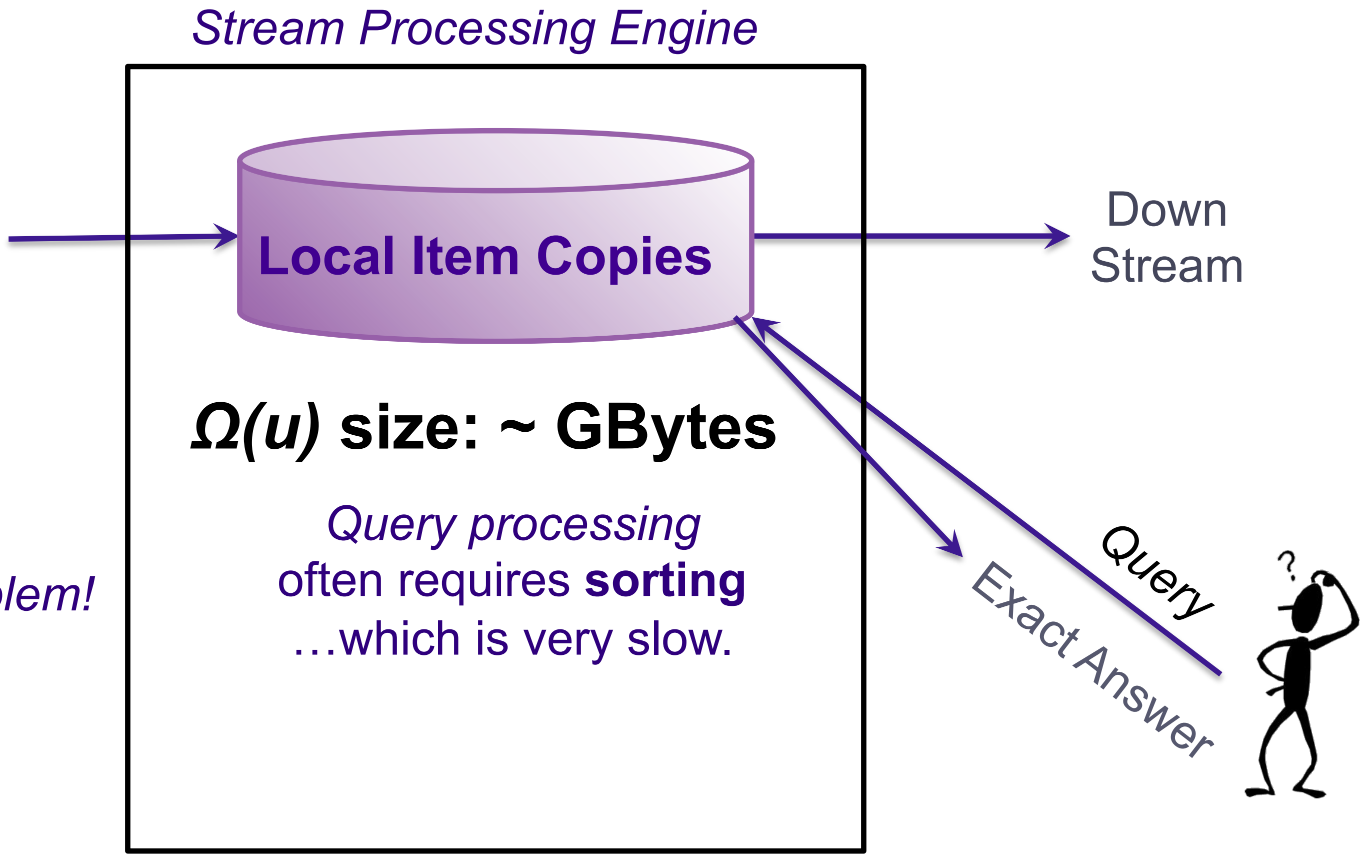
Query processing
often requires **sorting**
…which is very slow.

Exact Approach From Streamed Data

Big Data Stream

Item_∞, ..., Item_2, Item_1

… *Billions of Items...*



$\Omega(u)$ size: ~ GBytes

Query processing
often requires **sorting**
…which is very slow.

Micro-batch
"Streaming Engines"
Do Not Solve This Problem!

YAHOO!

If Approximate Results are Acceptable …
We Can Reduce the Query Size Substantially
… by *Sketching*!

Sketch Type	Sketch K	Sketch Size	Sketch Error
Distinct Count	k = 4096	32 KB (or 2 KB HLL)	1.6% Relative
Frequent Items	k = 256	4 KB	(1.4% * W) Absolute
Quantiles	k = 128, N = 1B	25 KB	1.7% Absolute Rank

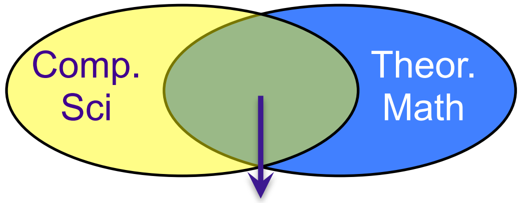
k = A configuration parameter that affects size and accuracy

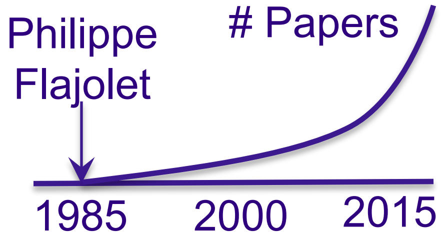
W = Sum of all count weights

N = Stream size

YAHOO!

Sketch Origins





Research Disciplines

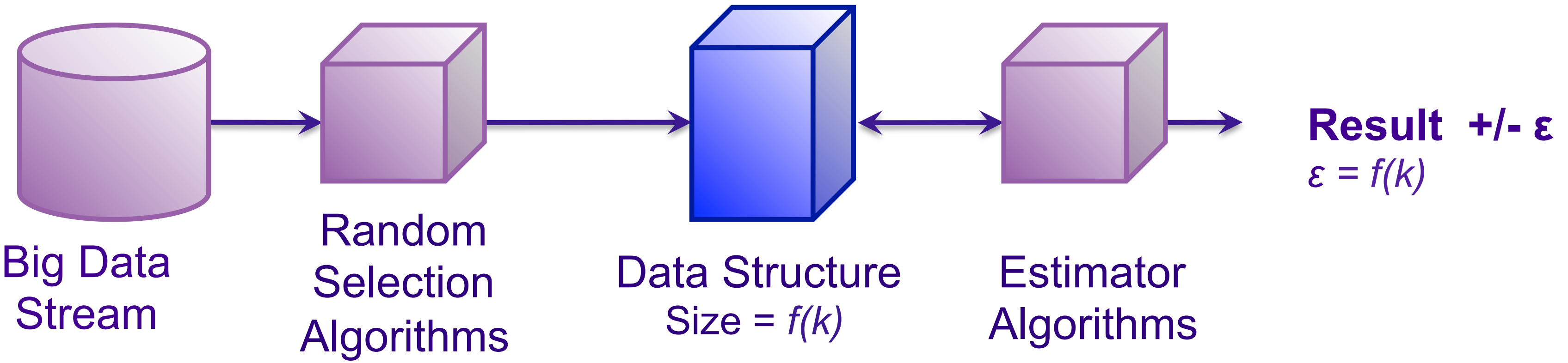
- Stochastic Streaming Algorithms
- Sub-linear Algorithms

"Sketch"

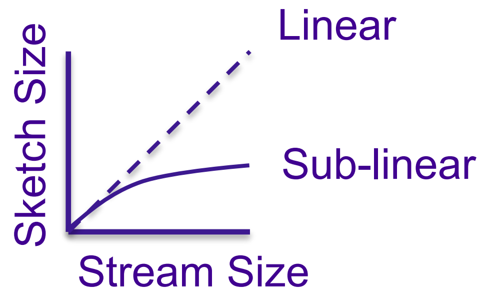
The Common Term for a Broad Range of These Algorithms

YAHOO!

Sketch: Common Elements & Properties

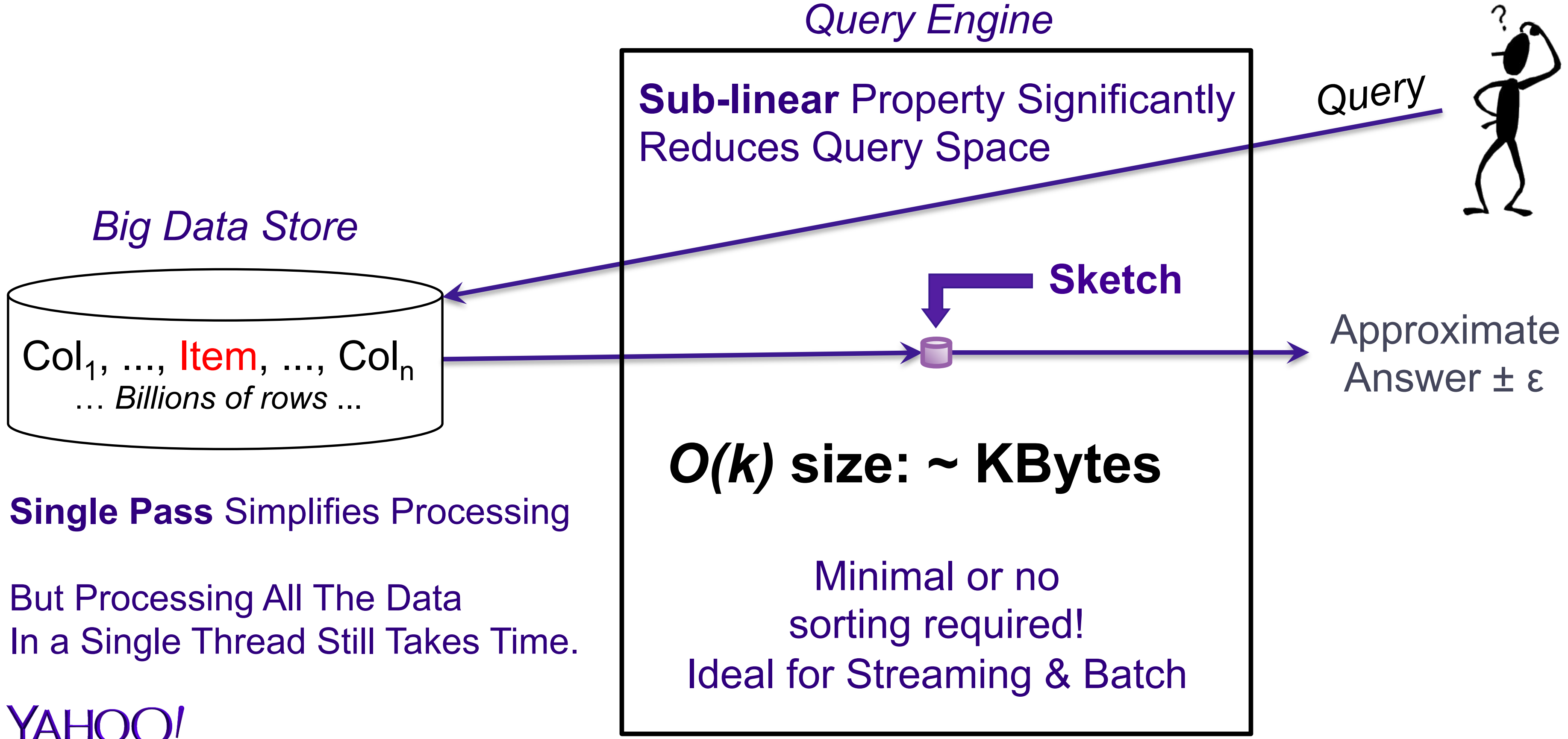


- Small Size, Sub-linear in Space
- Single-pass
- Mergeable
- Mathematically proven error bounds



YAHOO!

First Big Win: Query Space


Query Engine
Sub-linear Property Significantly Reduces Query Space
Query
Big Data Store
Col_1, ..., Item, ..., Col_n
… *Billions of rows* ...
Sketch
Approximate Answer ± ε
***O(k)* size: ~ KBytes**
Minimal or no sorting required!
Ideal for Streaming & Batch

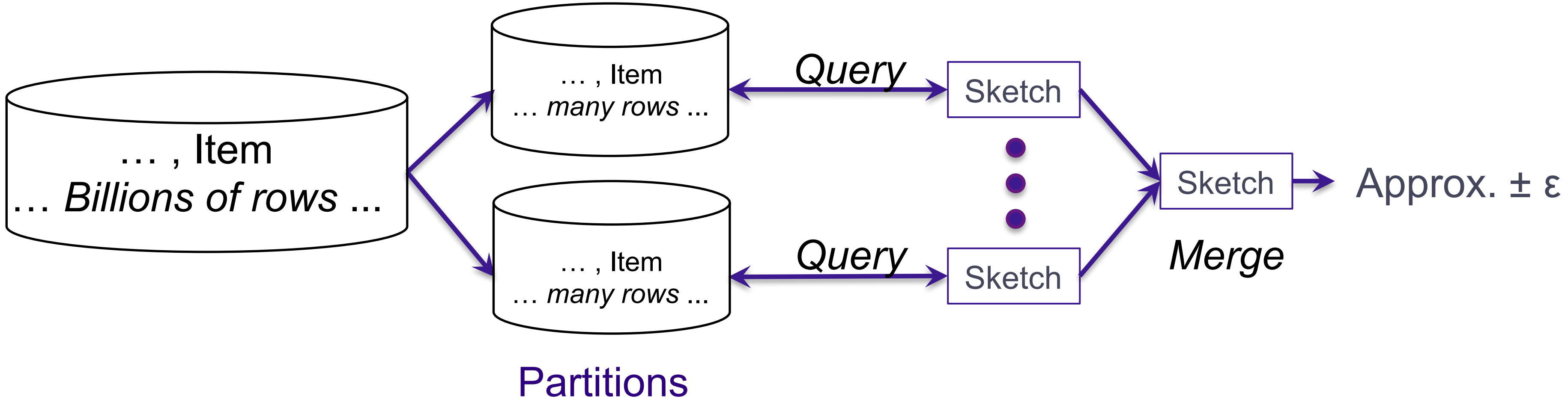
Single Pass Simplifies Processing

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

YAHOO!

The Second Big Win: Mergeability

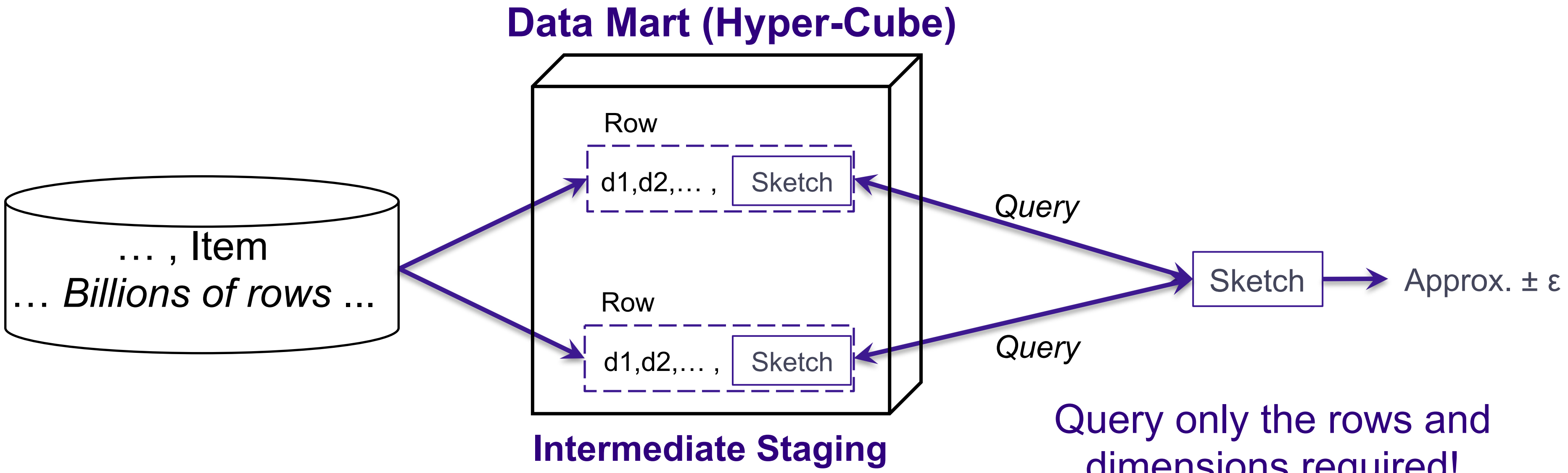
- **Mergeability** Enables Parallelism
- With No Additional Loss of Accuracy!
- But Data Skew Across Partitions Can Still Be A Systems Challenge



YAHOO!

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!

YAHOO!

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-leverageable reports
- Enables reporting on arbitrary dimension combinations
  - Fast merging: Recombine sketches as needed
    - Exact counting is not “additive”: Lots of tables, not a data cube
  - Rolling day, week or month
  - Simple time zone adjustments
- Set operations are cheap
  - Intersections, e.g. for user retention
  - Set differences, e.g. for filtering

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.

YAHOO!

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

YAHOO!

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

YAHOO!

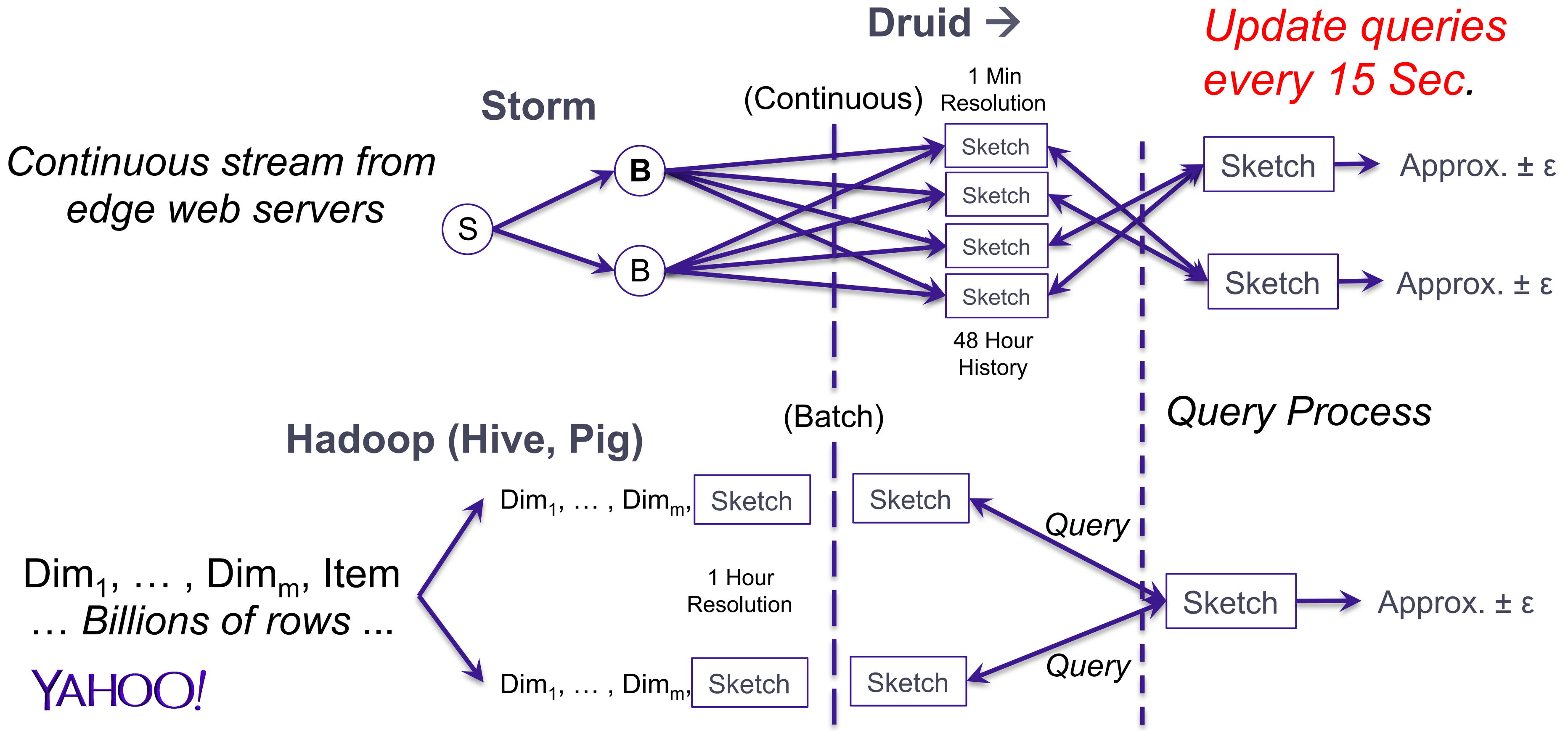
Case Study 1: Perspectives

- 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

YAHOO!

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

Case Study 2: Flurry/Druid Sketch Flow Architecture



YAHOO!

Case Study 2: 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

YAHOO!

Major Sketch Families in DataSketches Library

Cardinality: Theta & HyperLogLog (HLL) Sketches

- Theta: Includes Set Expressions (e.g., Union, Intersection, Difference)
- HLL & HLL Map: Highly compact
- Sample code for Java, Hive, Pig, Druid, Spark
- Adaptors for Hive, Pig, Druid
- Can Operate Off-Heap

Quantiles Sketches

- Quantiles, PMF's and CDF's of streams of comparable values.
- Sample code for Java, Hive, Pig
- Adaptors for Hive, Pig, Druid
- Can Operate Off-Heap

YAHOO!

Major Sketch Families in DataSketches Library

Frequent Items Sketches

- Heavy Hitters of arbitrary objects from a stream of objects
- Sample code for Java, Hive, Pig
- Adaptors for Hive, Pig

Associative: Tuple Sketches

- Theta Sketches with attributes
- Sample code for Java, Hive, Pig
- Adaptors for Hive, Pig

Sampling: Reservoir Sketches, Weighted and Unweighted.

- Uniform sampling to fixed-*k* sized buckets
- Sample code for Java, Pig
- Adaptors for Pig

YAHOO!

Thank You!

Please Visit:

DataSketches.GitHub.io

sketches-user@googlegroups.com

YAHOO!