Practical Sketching for Production Systems

Sketching for Production Systems

- What makes a practical sketch?
- Sketch-based Architectures
  - Progression: Experimentation to Data cubes
  - Case Study
- Common Questions and Challenges
  - Implementation subtlety and challenges
  - Accepting approximation
  - Which Sketch to use?
  - System planning
- Examples
  - Apache DataSketches* Library
  - Demonstration

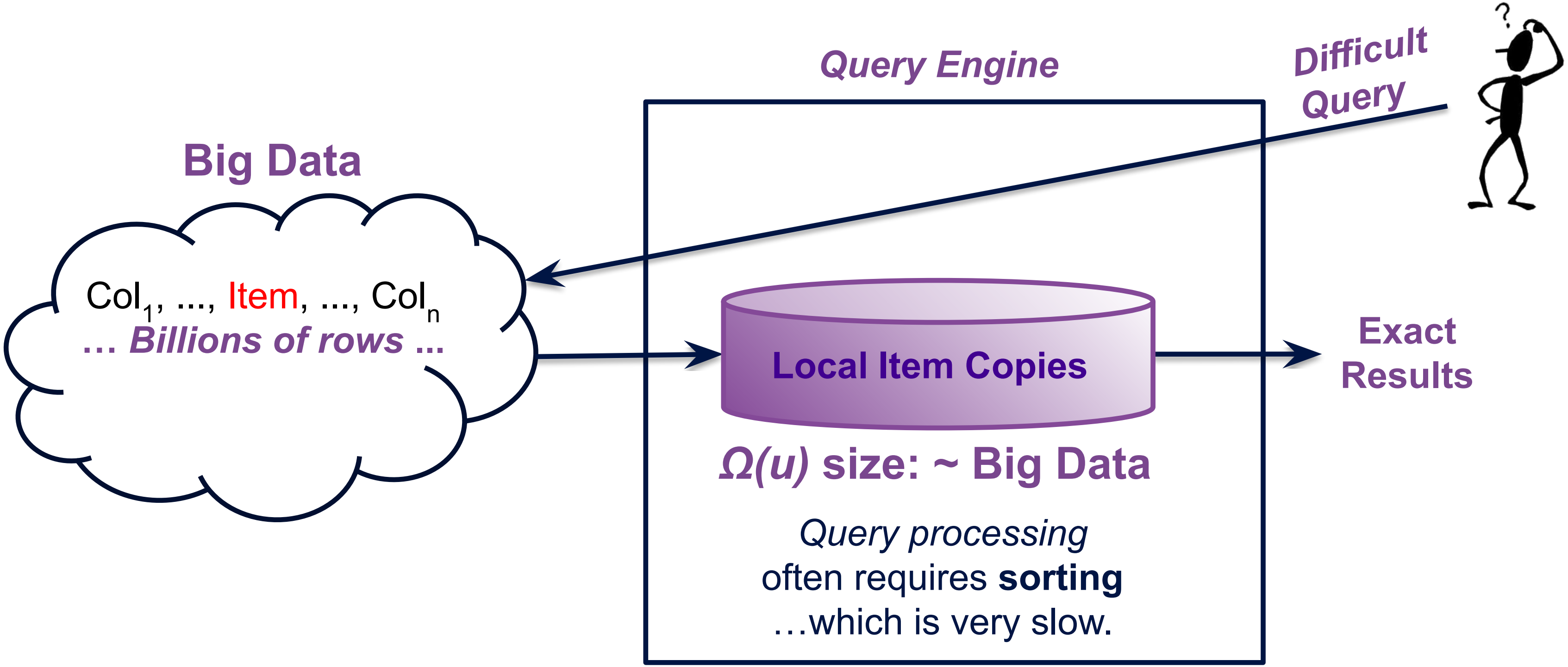
* Currently in Incubating status

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 Rows or Key, Value Pairs …					

… Analyze This Data in Near-Real Time

Exact Analysis Methods Require Local Copies

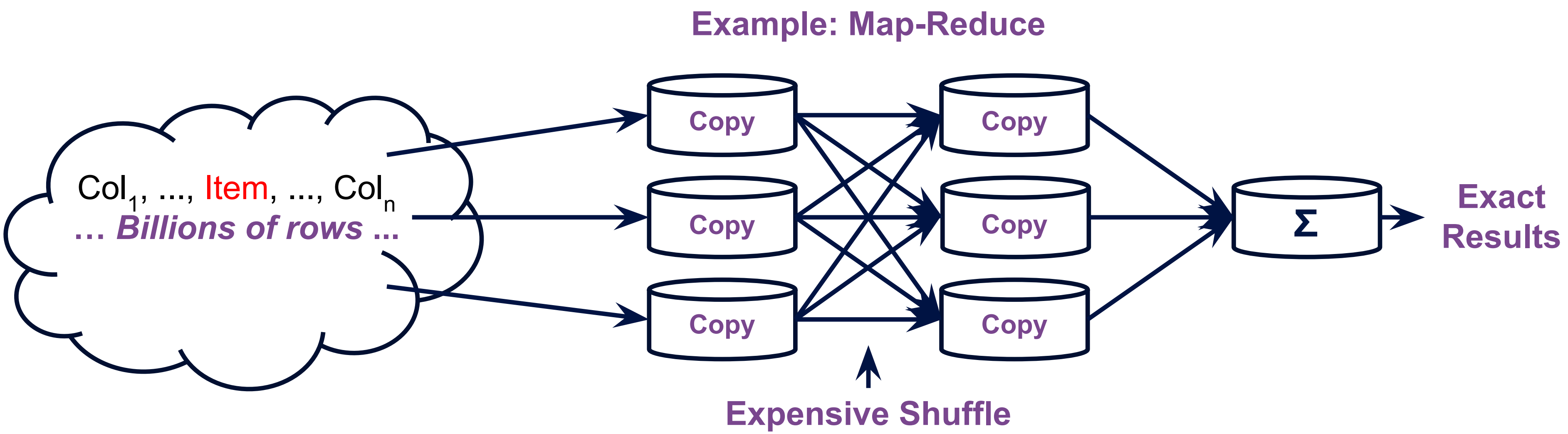


Note: Micro-batch “streaming platforms”, e.g., Storm, do not solve the fundamental problem for you!

Parallelization Does Not Help Much

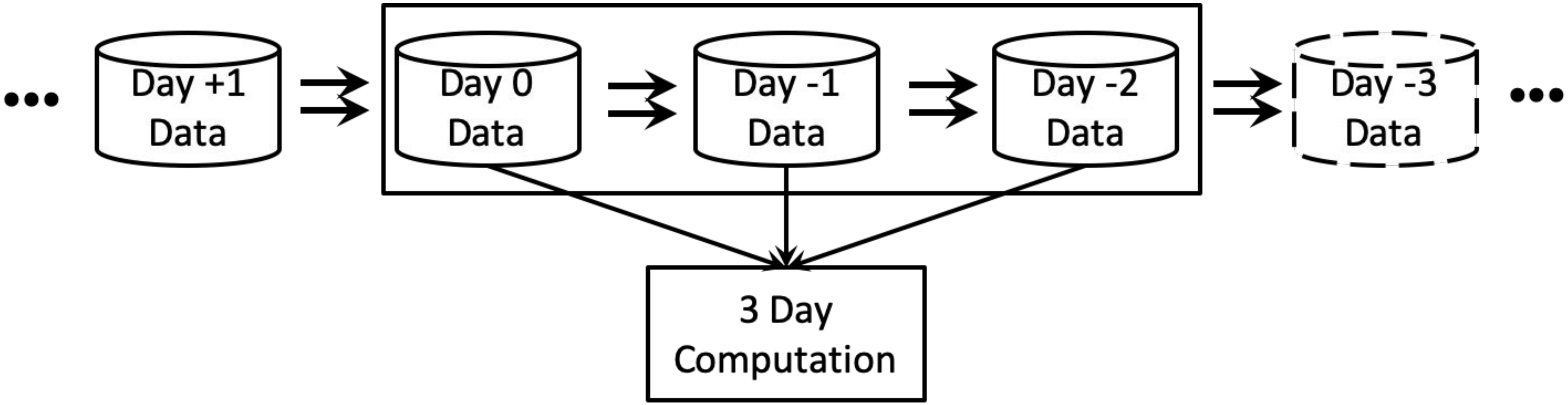
Because of Non-Additivity.

You have to keep the copies somewhere!



Exact Time Windowing

Requires Multiple Touches of Every Item

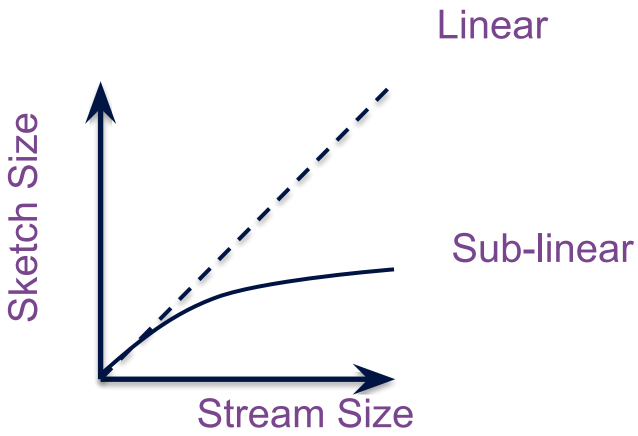


Every dataset is processed N times for a rolling N-day window!

Sketch Properties for Production Systems

(Not All Sketches Qualify)

- Small Stored Size
- Sub-linear in Space
- Single-pass, “One-Touch”
- Distribution Independent
- Order Independent
- Mergeable
- Approximate, Probabilistic
- **Mathematically Proven Error Properties**

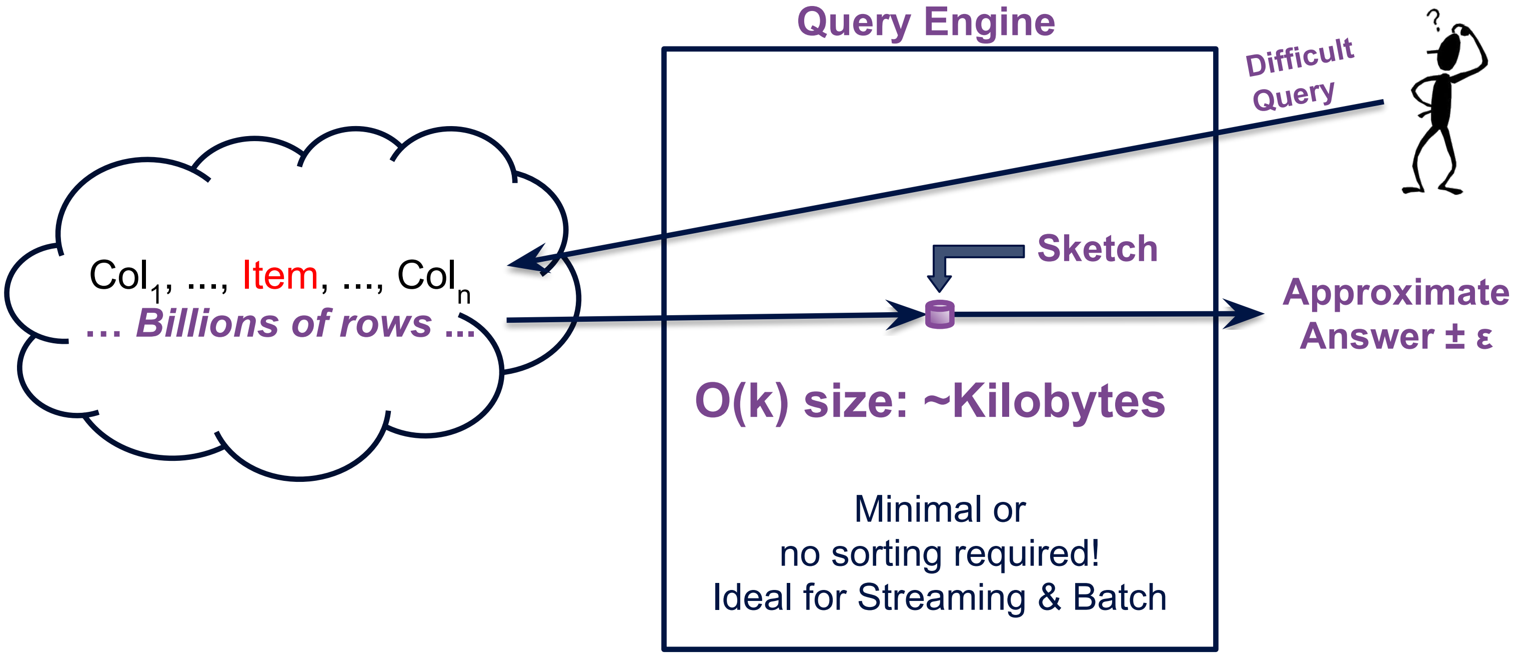


Sketch-Based Systems

- Common pattern while exploring sketches
  - Series of design wins from adopting sketches
- Faster, cheaper, enables new functionality
  - Not all desirable queries have sketching solutions
  - May still need to keep raw data

Win 1: Small Query Space

Sketches Start Small
Sublinear Means they Stay Small
Single Pass Simplifies Processing

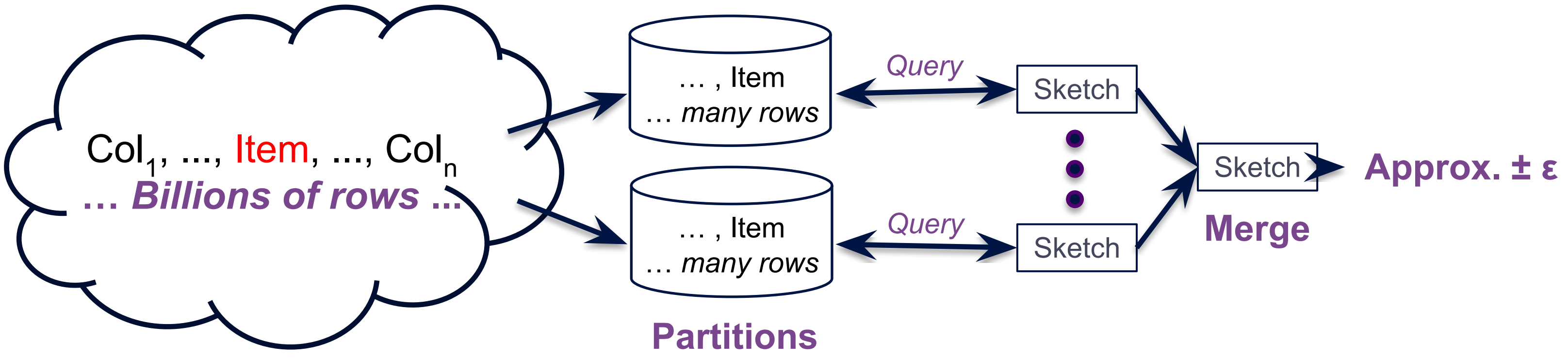


Win 2: Mergeability

Full Mergeability Enables Parallelism

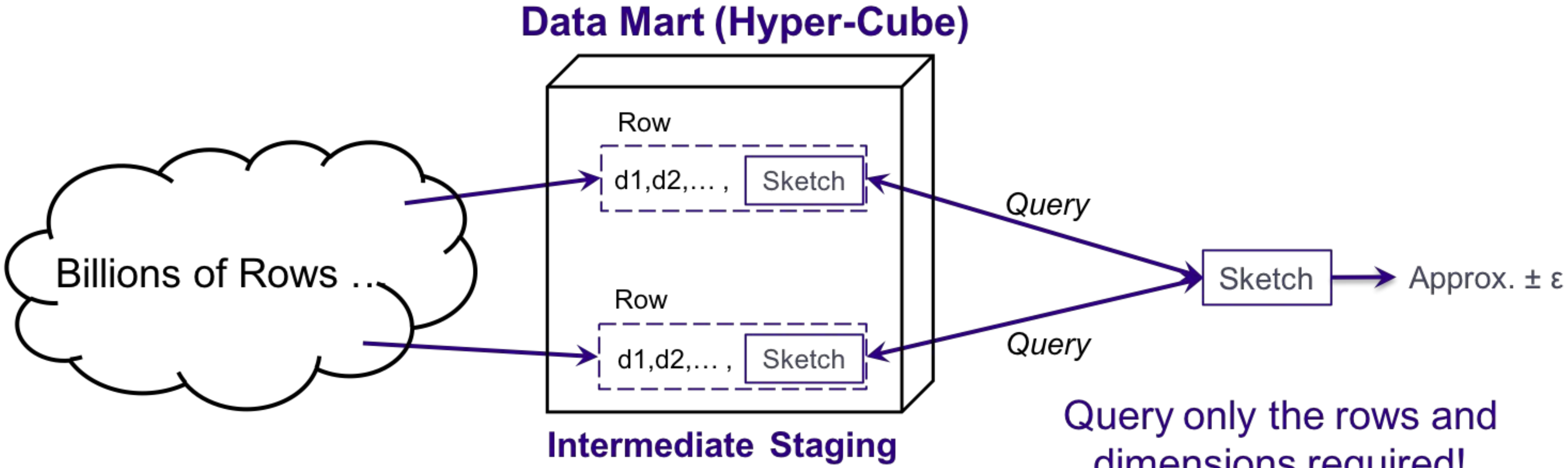
Non-Additive Metrics Act Like **Additive** Objects

Full Mergeability Enables Set Expressions for Selected Sketches



Win 3: Near Real-Time Queries
Win 4: Simplified Architecture

Intermediate Hyper-Cube Staging Enables Query Speed
Additivity Enables Simpler Architecture

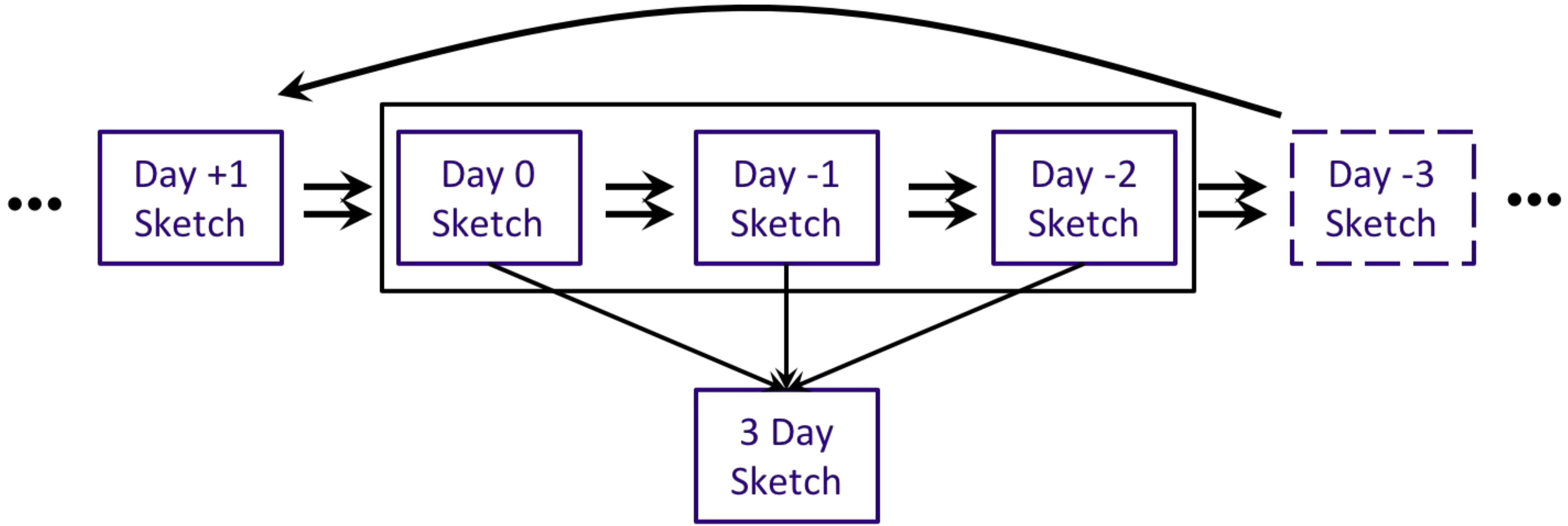


Query only the rows and dimensions required!

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

Win 5: Time Windowing

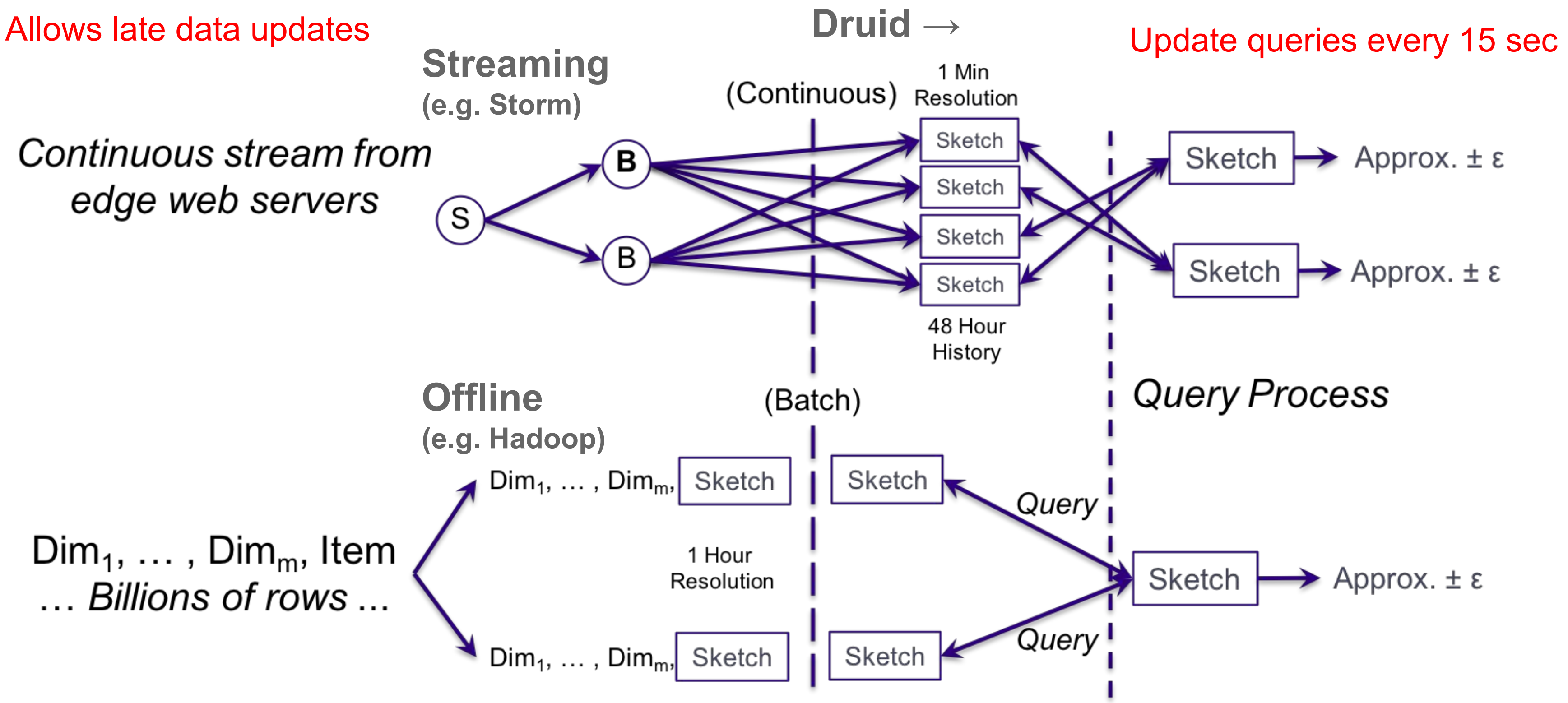
Late Data Processing Also Simplified



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

Case Study: Flurry/Druid

Offline + Online for Near Real-Time Results



Case Study: Real-Time Flurry, Before/After

Also, Win 6: Lower System Cost

- 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
Virtual Core Seconds (VCS) per Month	**~80B**	**~20B**
Result Freshness	**Daily: 2 to 8 hours; Weekly: ~3 days** **Real-time Unique Counts Not Feasible**	**15 seconds!**

Common Questions and Challenges

Implementation is subtle

- Treat like a math library: Don’t make your own
- Algorithms seem conceptually simple, but…
  - Lots of edge cases for robust implementations
  - Found significant bugs in well-known HLL distributions
- Simple mergability alone is insufficient!
  - System design requirements evolve, e.g. target sketch error
  - Need *correct* solutions for merging across sketch sizes

Accepting Approximation

- Different strategies for different roles
- Scientists/Engineers
  - Experiment to determine accuracy, see what else sketches provide
  - How does sketch error compare to other uncontrolled sources of error (e.g. missing/corrupt data or sampling error, whether implicit or explicit)
- Product Owners
  - Demonstrate new features
  - Speed gains and cost savings (including reprocessing)
  - Note configurable accuracy

Which Sketch Should I use?

- If multiple sketches seem appropriate, no general answer
  - Accuracy, in-memory size, stored size, update vs merge vs (de)serialize speed
  - Must decide in a systems context
- Examples
  - Network Router: Count distinct IPs to detect DDoS attack
    - Want small in-memory size, mergeability and set operations less critical
  - Web/App Analytics: Count distinct devices/people visiting
    - Different time windows and set operations likely key features

HLL vs CPC vs Theta

- HLL
  - Small serialized size, small in-memory footprint
  - Moderate merge speed
  - Terrible accuracy for intersections, no set difference
- CPC (Compressed Probabilistic Counting)
  - Best known compressed size/accuracy combination
  - Smallest serialized size, moderate in-memory footprint
  - Moderate merge speed
  - Terrible accuracy for intersections, no set difference
- Theta
  - Larger serialized size, in-memory footprint
  - Fast merge speed
  - Best accuracy for intersections, allows set differences
  - Relative size increase vs HLL or CPC depends on usage scenario

System Planning: Key Questions

- What types of queries do I need to support?
- What accuracy do I really need?
  - Ideally, pick library that lets you change your mind later!
- Do I need to support real-time data? Late data?
- With sketches available, what new functions will I want?

Examples

Examples Powered By:

Apache DataSketches

Currently in incubation status

datasketches.apache.org

Who are we?

Project Committers

- Lee Rhodes, Distinguished Architect, Verizon Media (project founder)
- Alex Saydakov, Systems Developer, Verizon Media
- Jon Malkin, Ph.D., Research Engineer, Verizon Media
- Edo Liberty, Ph.D., Founder, HyperCube Technologies
- Justin Thaler, Ph.D., Assistant Professor, Georgetown University, Computer Science
- Roman Leventov, Systems Developer for Druid, Metamarkets
- Eshcar Hillel, Ph.D., Sr Scientist, Verizon Media Israel

Extended Team/Consultants

- Graham Cormode, Ph.D., Professor, University of Warwick, Computer Science
- Jelani Nelson, Ph.D., Professor, U.C. Berkeley
- Daniel Ting, Ph.D., Sr Scientist, Tableau / Salesforce
- Dave Cromberge, Permutive

datasketches.apache.org

About the library

Mission: Deep science + quality engineering for **Production Quality** sketches

- Trustworthy sketches
- Robust implementations (8+ years of production use)
- Robust algorithms (see slide 7)
- Open source characterization code

Notable features for large-scale systems

- Backwards compatibility
- Merging across sketch sizes
- Binary compatibility across supported languages
- Consistent serialization formats

datasketches.apache.org

The Apache DataSketches Library

Cardinality, 4 Families

- **HLL:** A very high performing implementation of this well-known sketch
- **CPC:** The best accuracy per space
- **Theta Sketches**: Set Expressions (e.g., Union, Intersection, Difference), on/off Heap
- **Tuple Sketches**: Generic, Associative Theta Sketches, multiple derived sketches:

Quantiles Sketches, 2 Families

- **Quantiles**, Histograms, PMF's and CDF's of streams of comparable objects, on/off Heap.
  KLL, highly optimized for accuracy-space.
- **Relative Error Quantiles** (under development)

Frequent Items (Heavy-Hitters) Sketches, 2 Families

- **Frequent Items**: Weighted or Unweighted
- **Frequent Directions:** Approximate SVD (a Vector Sketch)

Sampling: Reservoir and Variance Optimal (VarOpt) Sketches, 2 Families

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

Languages Supported:

- **Java, C++, Python**
- **Binary Compatibility**

datasketches.apache.org

Thank you!