SnappyData

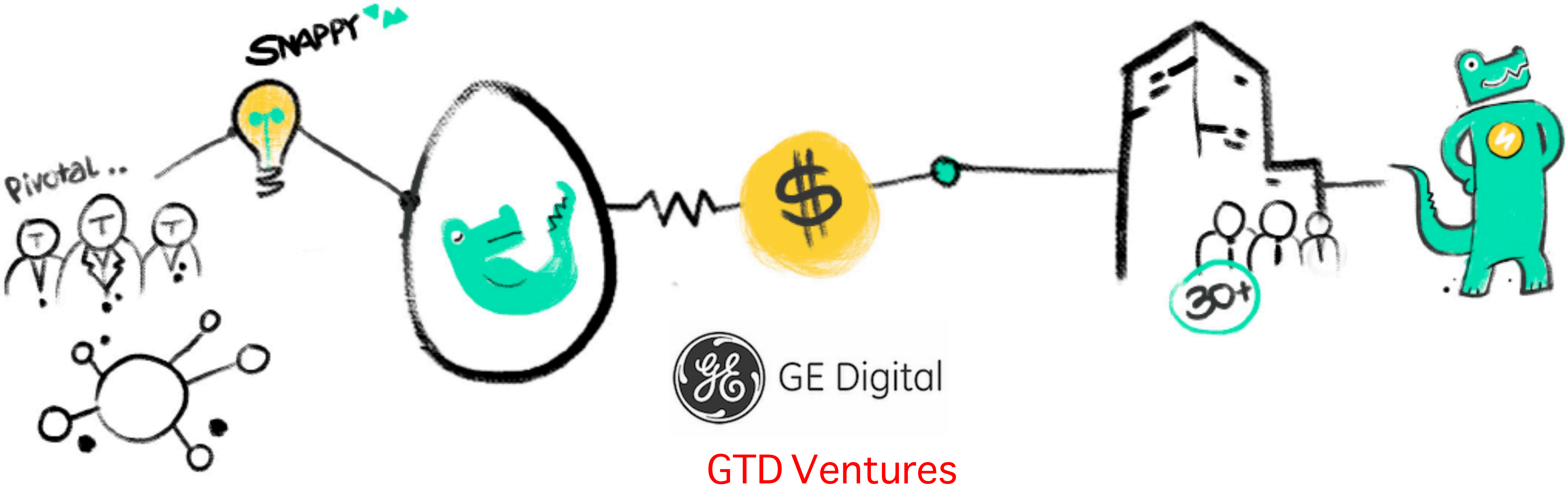
A Unified Cluster for Streaming, Transactions, & Interactive Analytics

Version 0.7 |

Barzan Mozafari | Jags Ramnarayan | Sudhir Menon
Yogesh Mahajan | Soubhik Chakraborty | Hemant Bhanawat | Kishor Bachhav

www.Snappydata.io

Our Pedigree

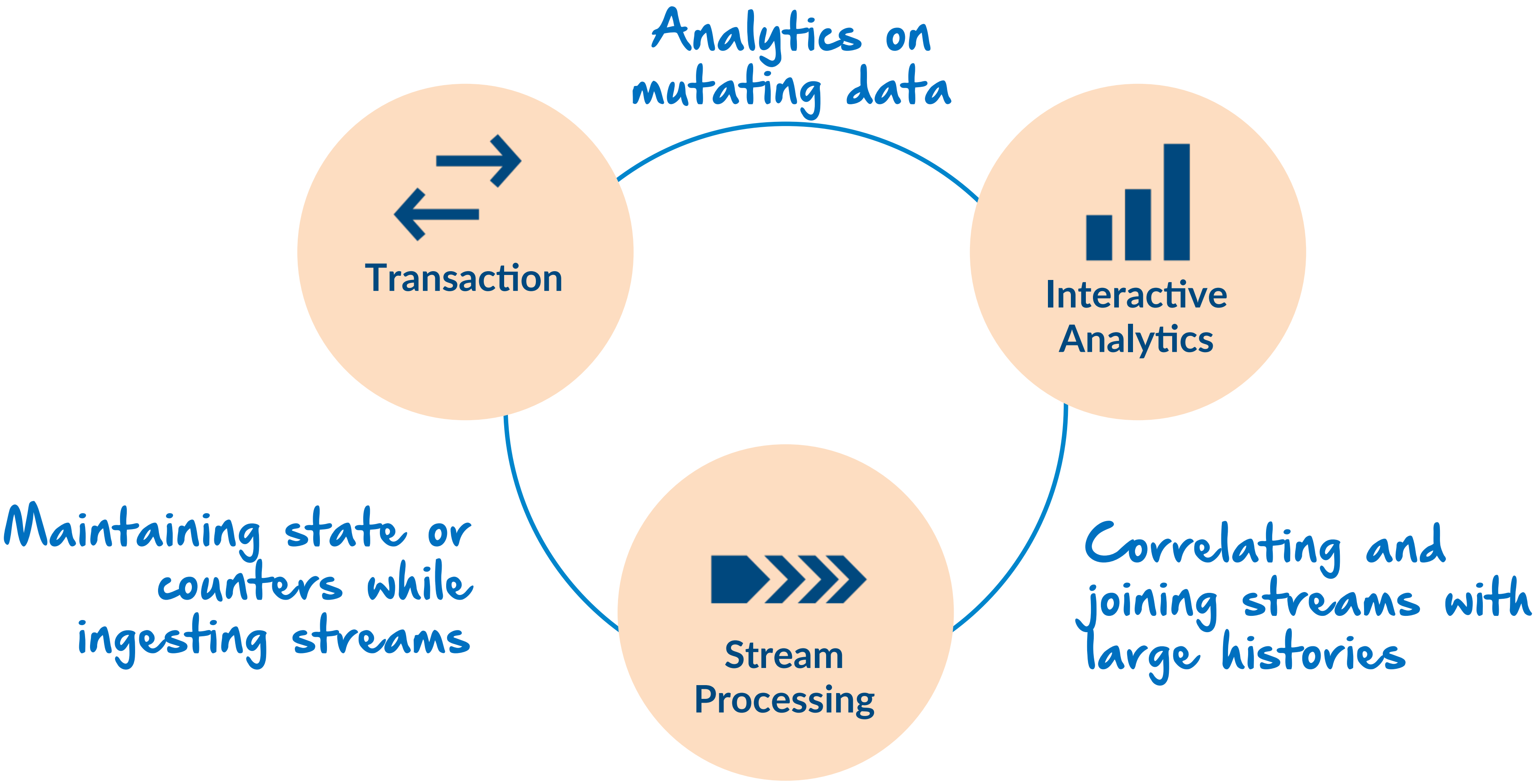


GE Digital

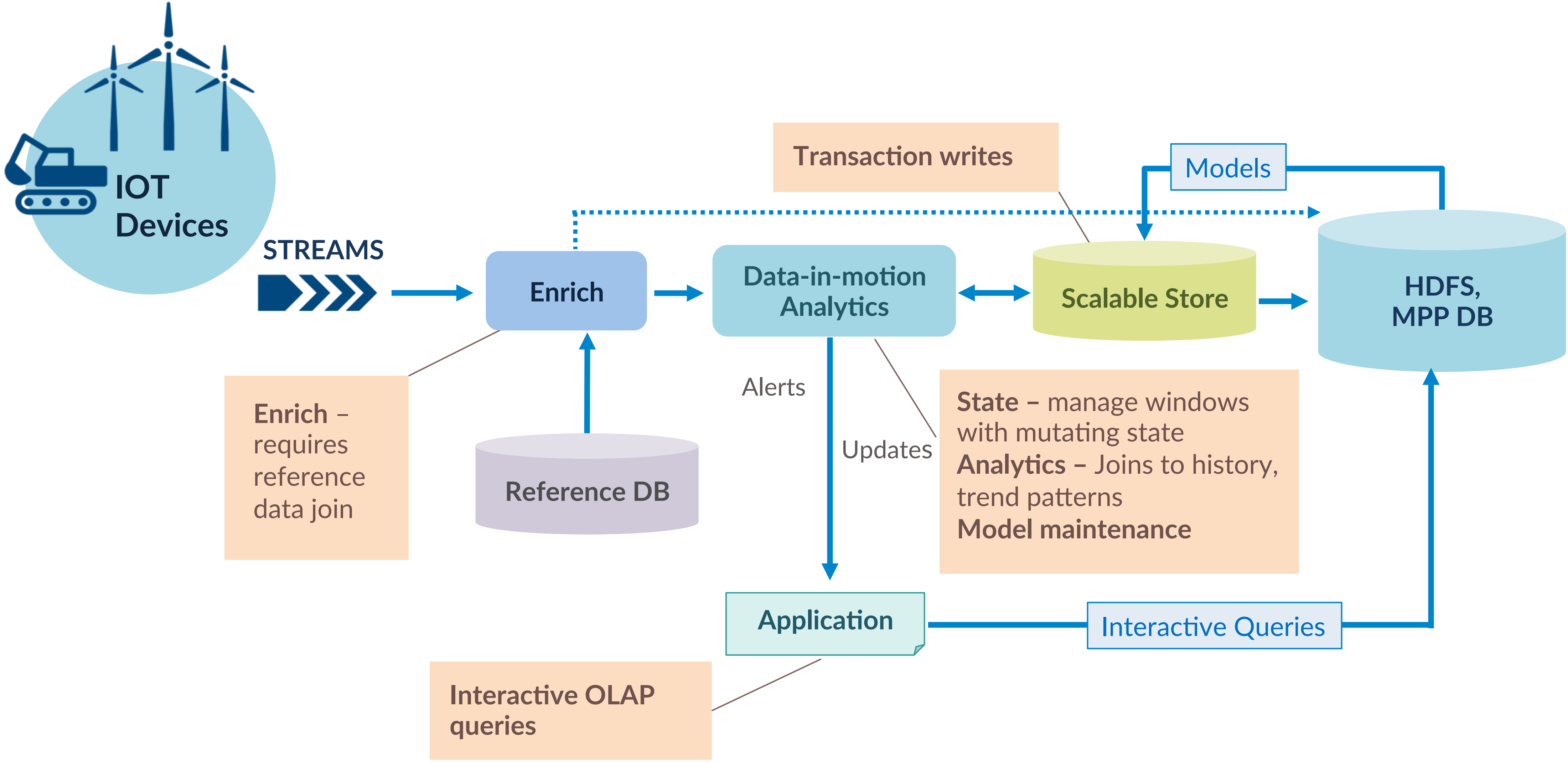
GTD Ventures

Pivotal

Mixed Workloads Are Everywhere



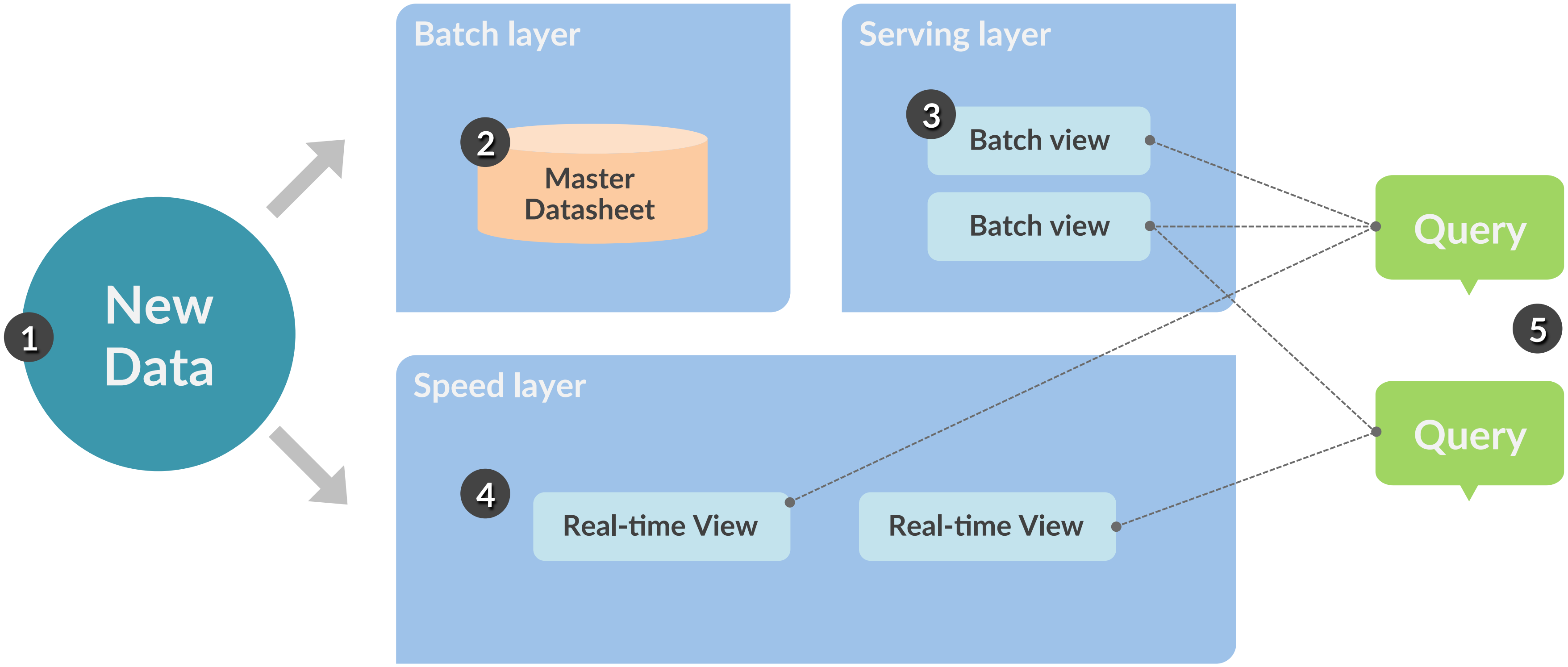
Mixed Workloads Are Everywhere



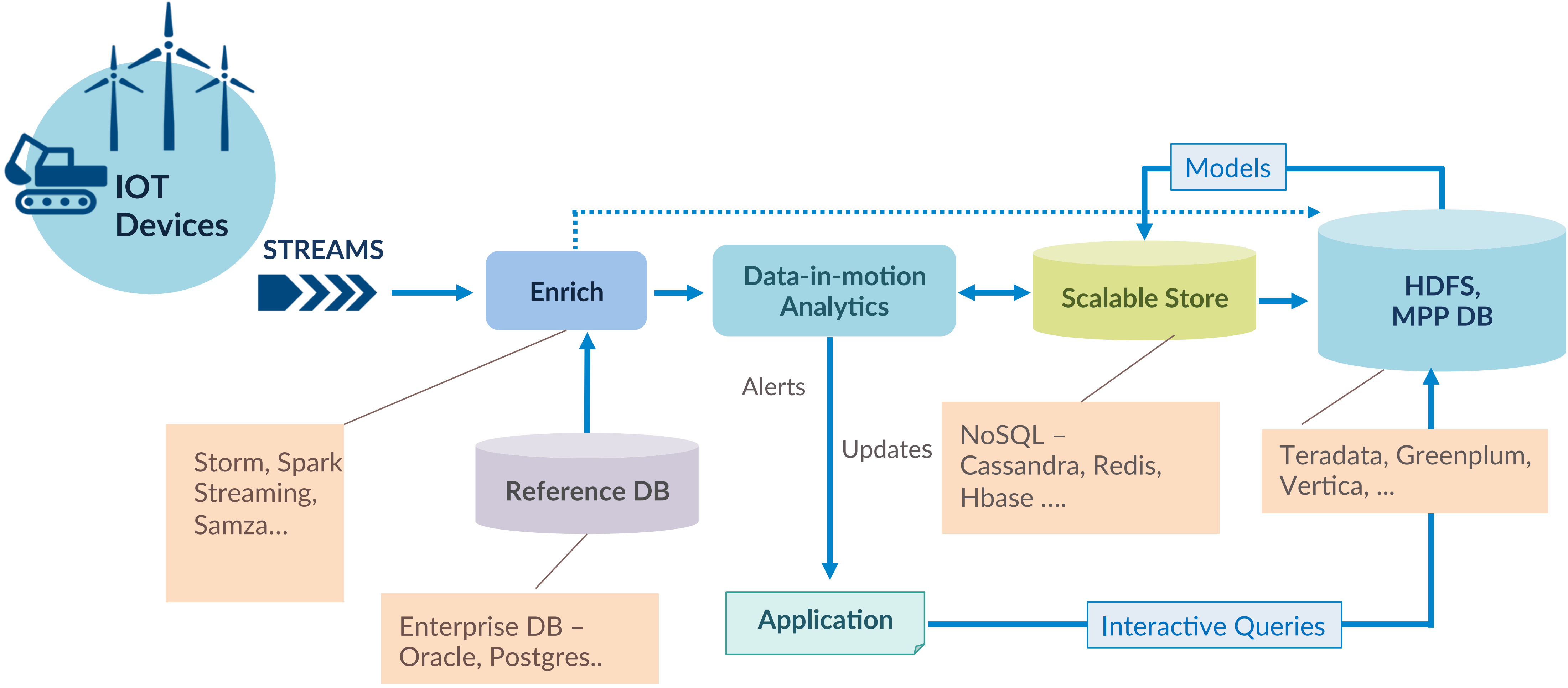
Why Supporting Mixed Workloads is Difficult?

Data Structures	Columnar	Row stores	Sketches
Query Processing Paradigm	Batch Processing	Point Lookups	Delta / Incremental
Scheduling & Provisioning	Long-running	Short-lived	Bursty

Lambda Architecture



Lambda Architecture is Complex



Lambda Architecture is Complex

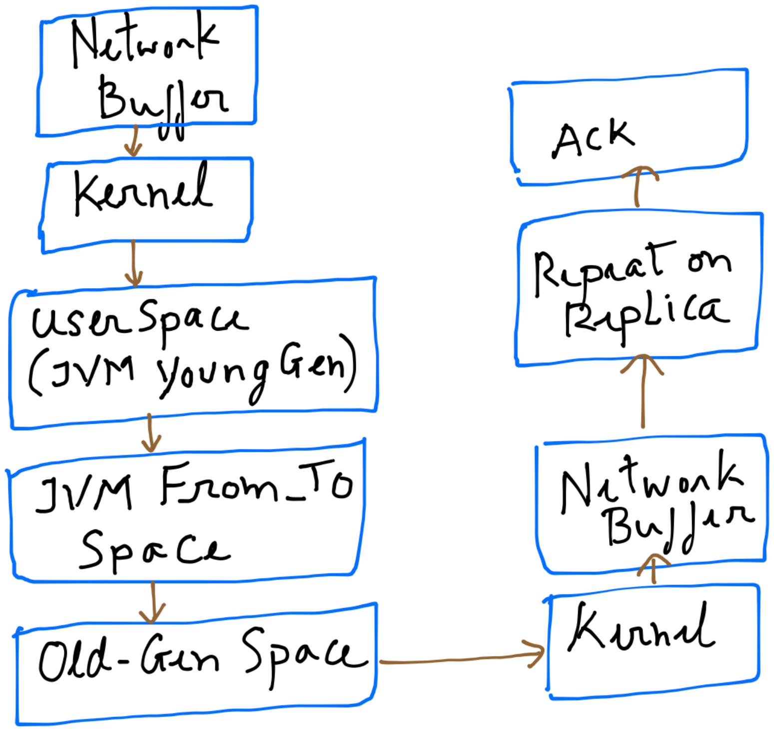
- **Complexity**

  - Learn and master multiple products, data models, disparate APIs & configs

- **Wasted resources**

- **Slower**

  - Excessive copying, serialization, shuffles
  - Impossible to achieve interactive-speed analytics on large or mutating data



Can We Simplify & Optimize?

Our Solution: SnappyData

Single Unified HA Cluster

OLTP + OLAP + Streaming for Real-time Analytics

Spark

Batch design, high throughput

Rapidly Maturing

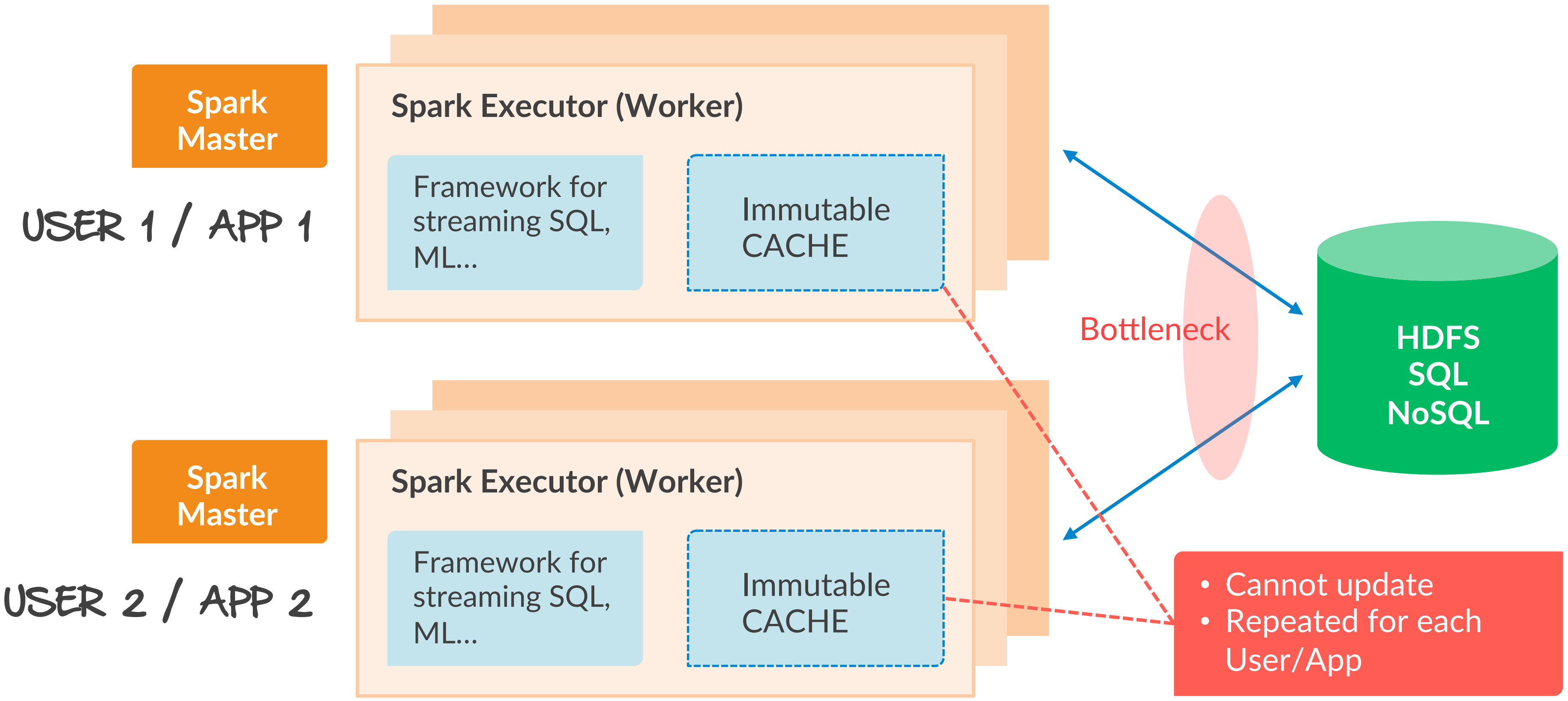
GemFire

Real-time design

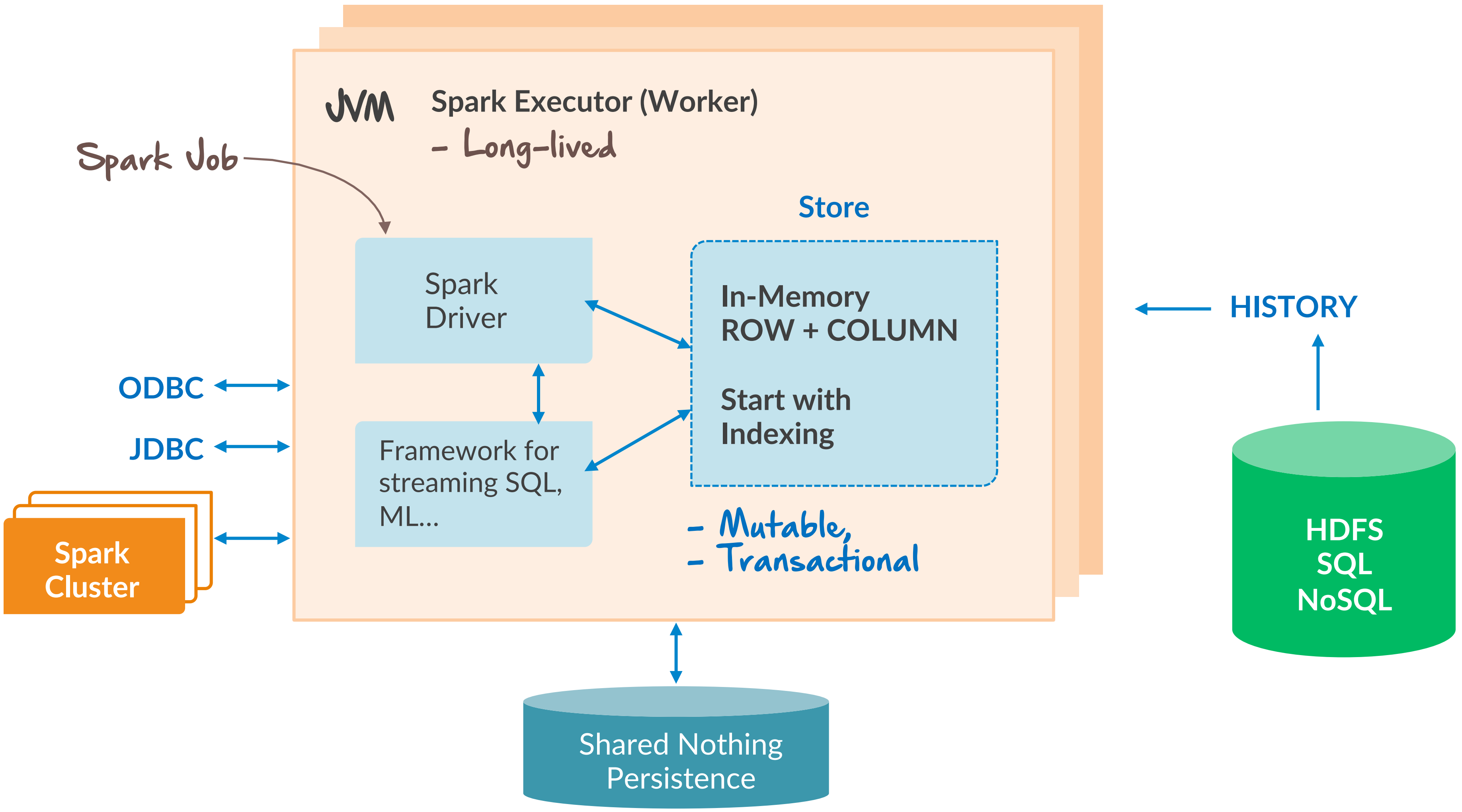
Low latency, HA, concurrency

Matured over 13 years

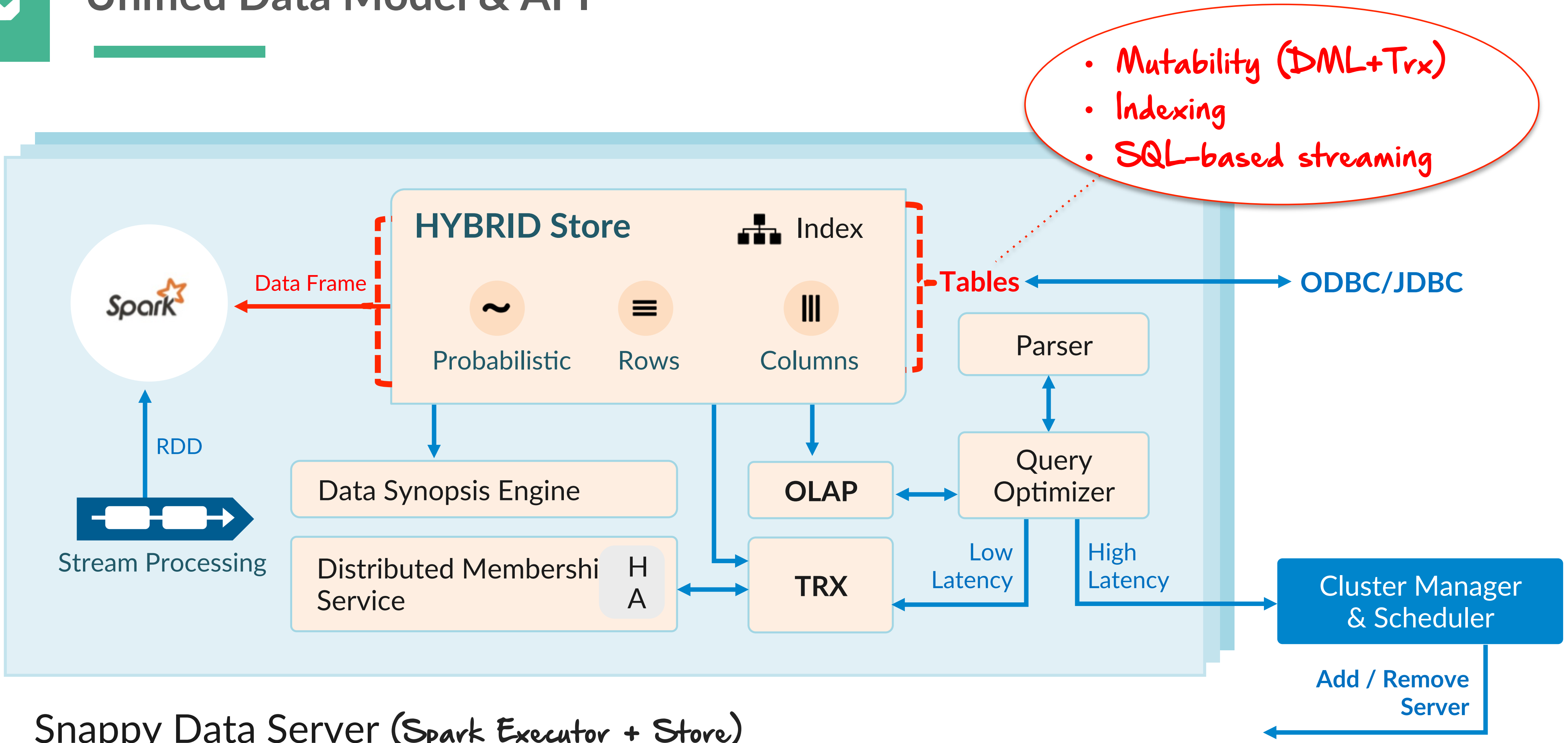
We Transform Spark from This …



... Into an "Always-On" Hybrid Database !

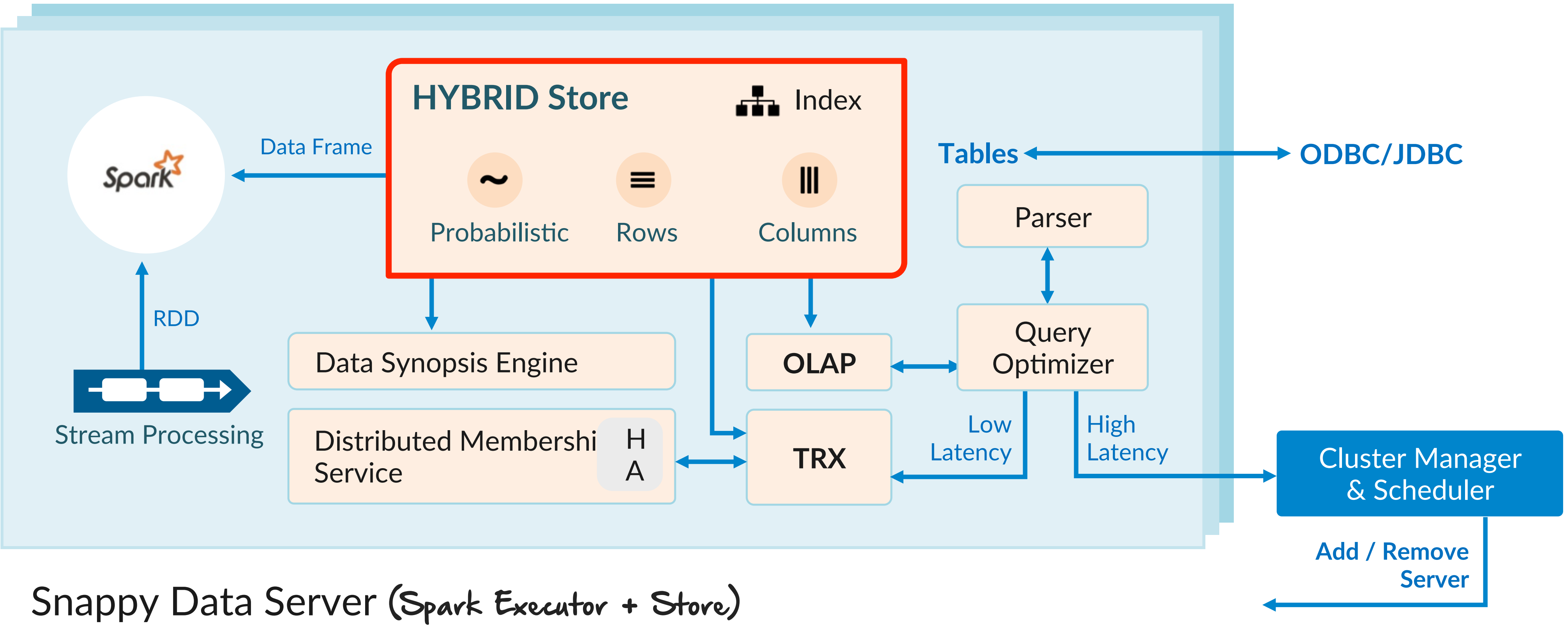


Unified Data Model & API



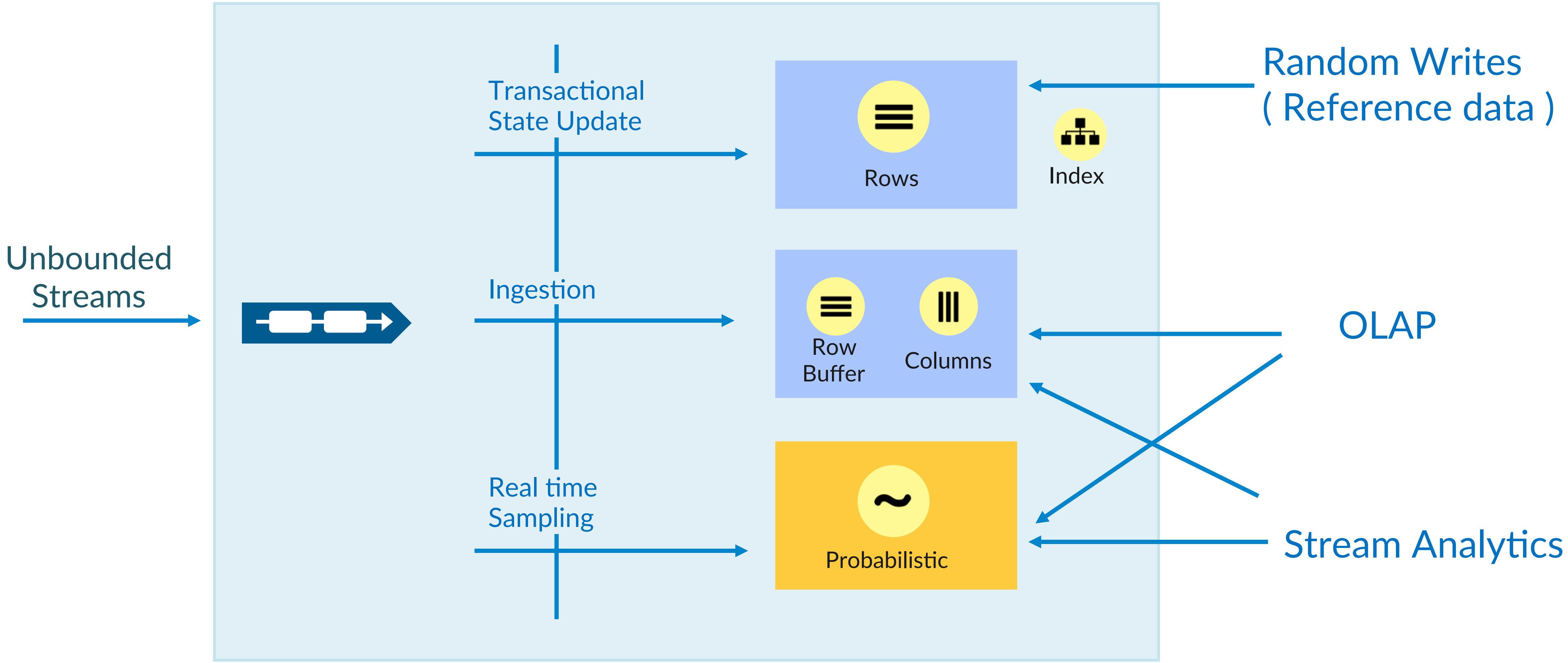
Snappy Data Server (Spark Executor + Store)

Overview

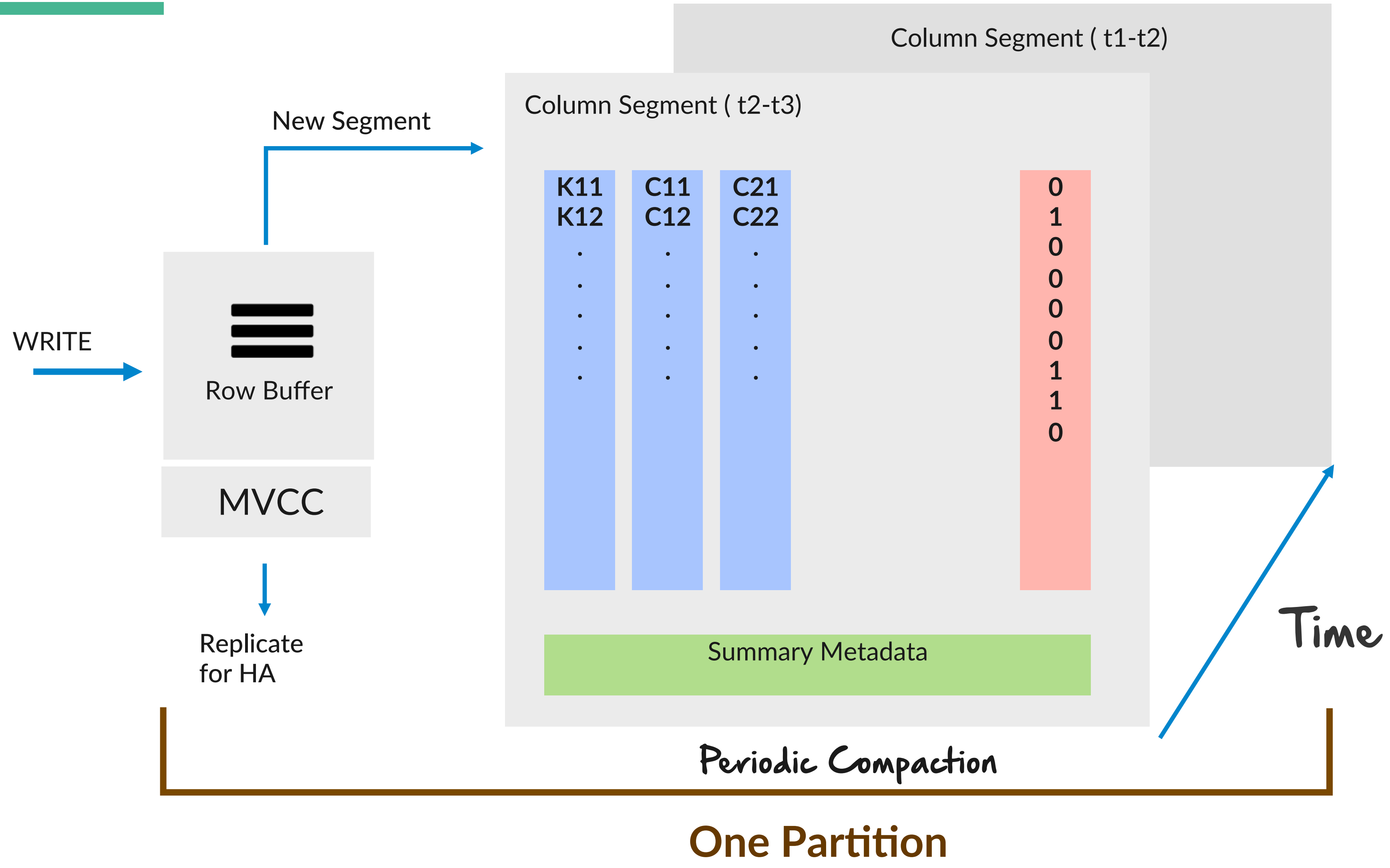


HYBRID Store
Index
Probabilistic
Rows
Columns
Data Frame
Spark
RDD
Stream Processing
Data Synopsis Engine
Distributed Membershi
Service
H
A
OLAP
TRX
Tables
ODBC/JDBC
Parser
Query
Optimizer
Low
Latency
High
Latency
Cluster Manager
& Scheduler
Add / Remove
Server
Snappy Data Server (Spark Executor + Store)

Hybrid Store



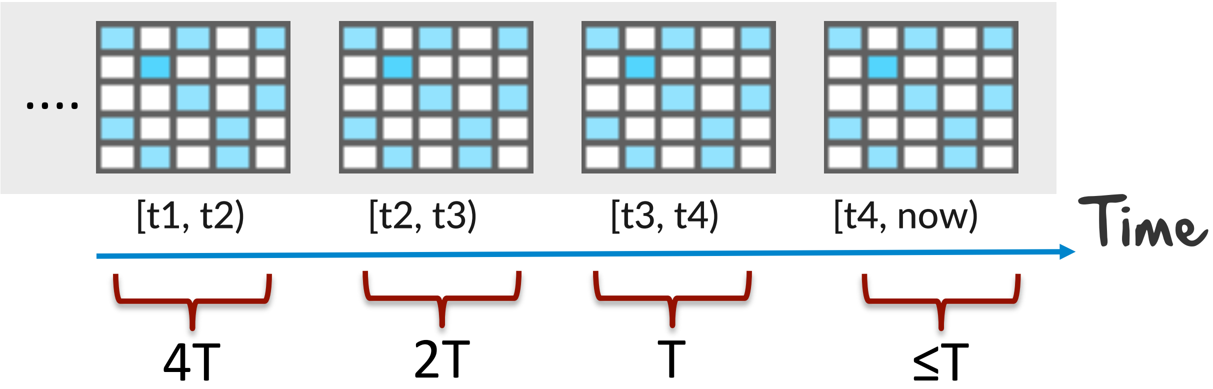
Updates & Deletes on Column Tables



Probabilistic Store: Synopses + Uniform & Stratified Samples

1. Streaming CMS (Count-Min-Sketch)

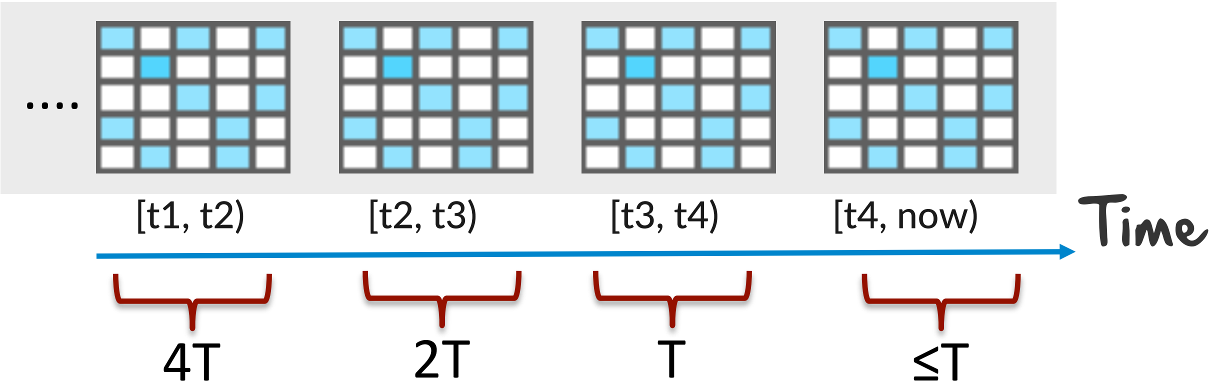
Higher resolution for more recent time ranges

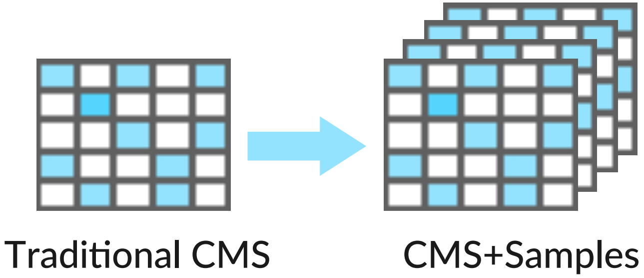


Probabilistic Store: Synopses + Uniform & Stratified Samples

1. Streaming CMS (Count-Min-Sketch)

Higher resolution for more recent time ranges





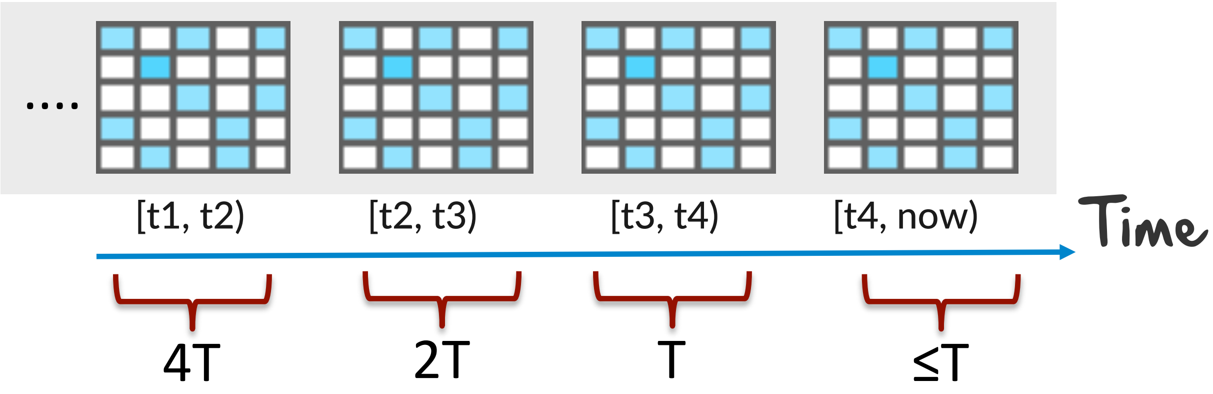
2. Top-K Queries w/ Arbitrary Filters

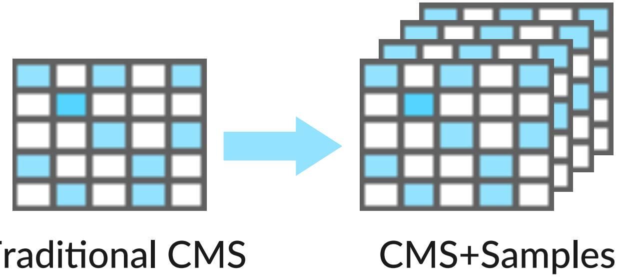
Maintain a small sample at each CMS cell

Probabilistic Store: Synopses + Uniform & Stratified Samples

1. Streaming CMS (Count-Min-Sketch)

Higher resolution for more recent time ranges



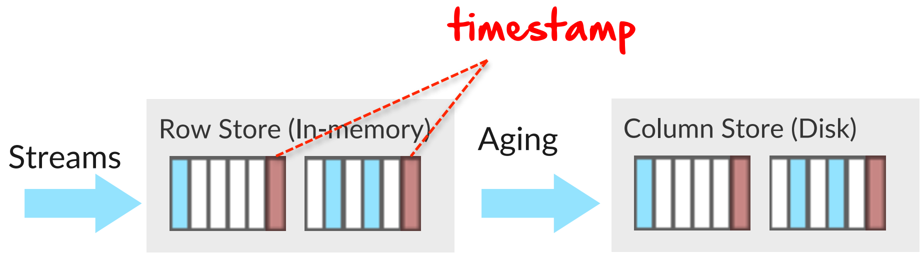


2. Top-K Queries w/ Arbitrary Filters

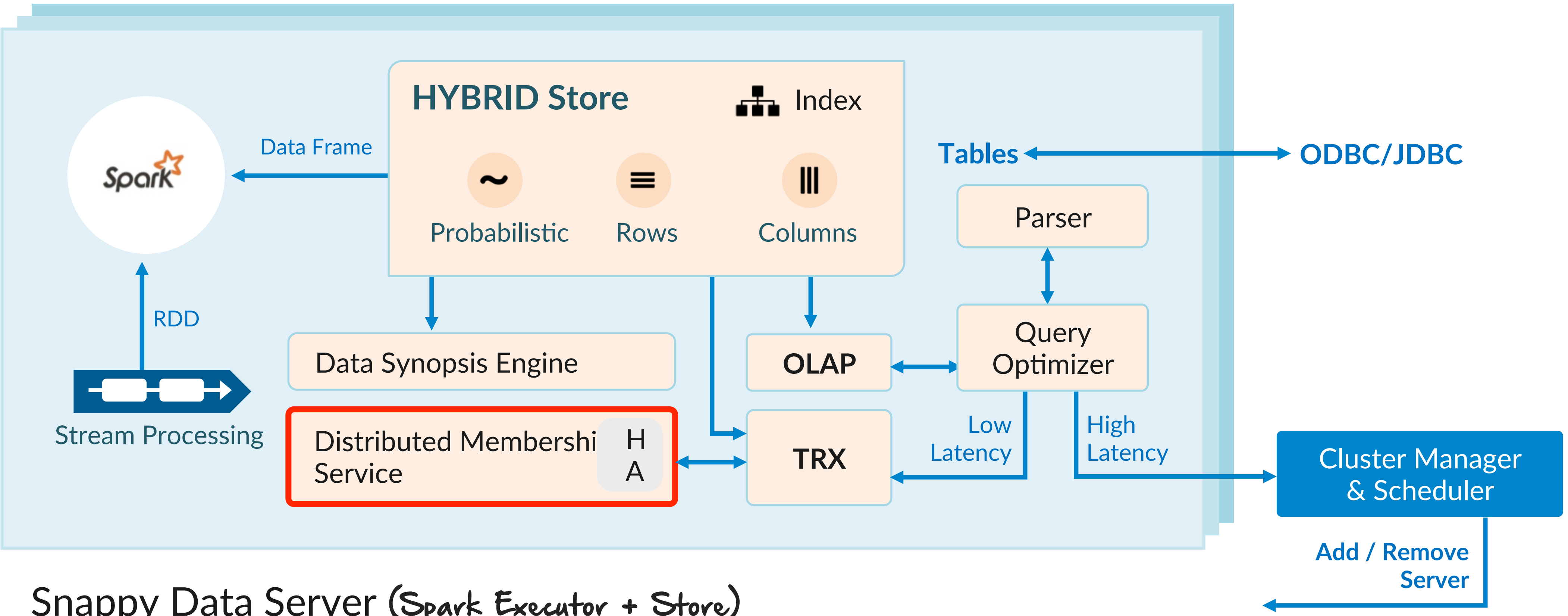
Maintain a small sample at each CMS cell

3. Fully Distributed Stratified Samples

Always include timestamp as a stratified column for streams



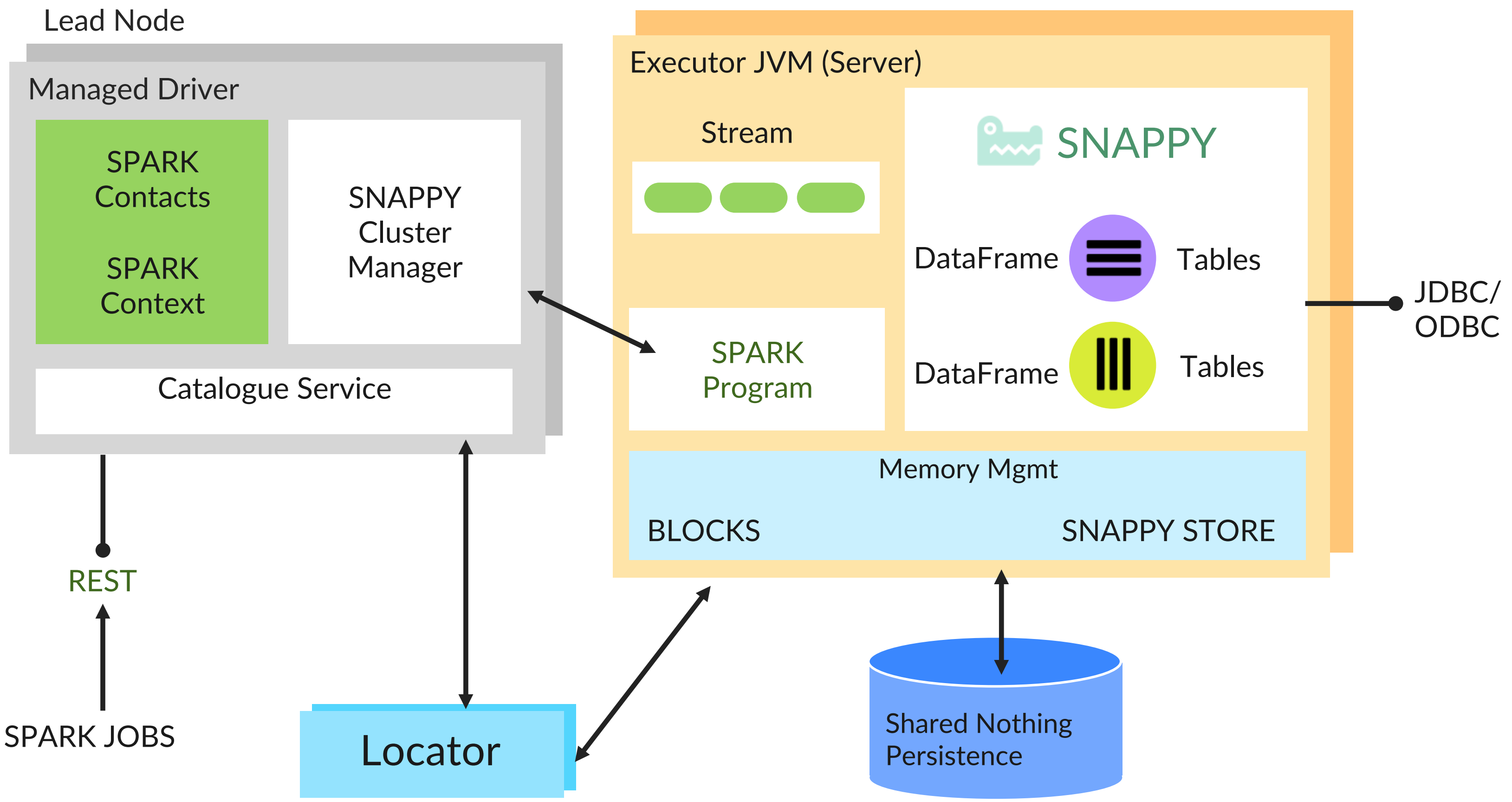
Overview



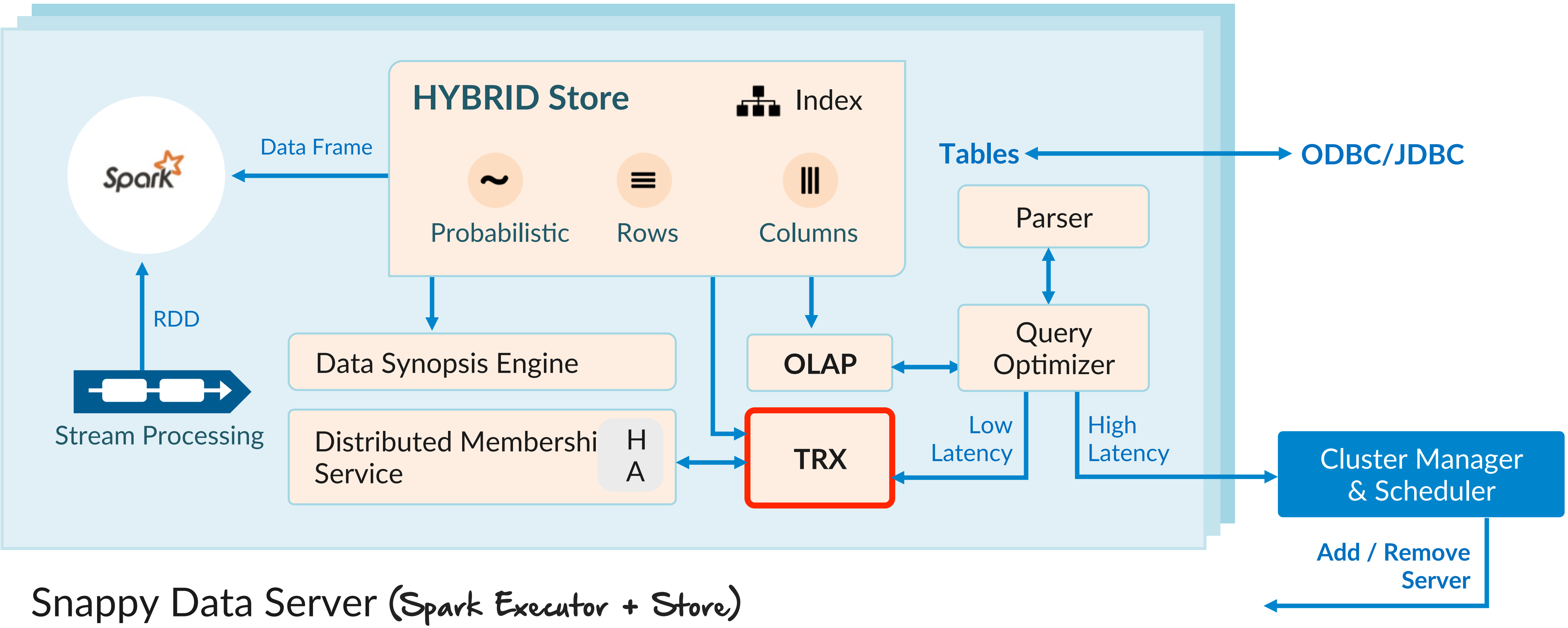
Snappy Data Server (Spark Executor + Store)

Supporting Real-time & HA

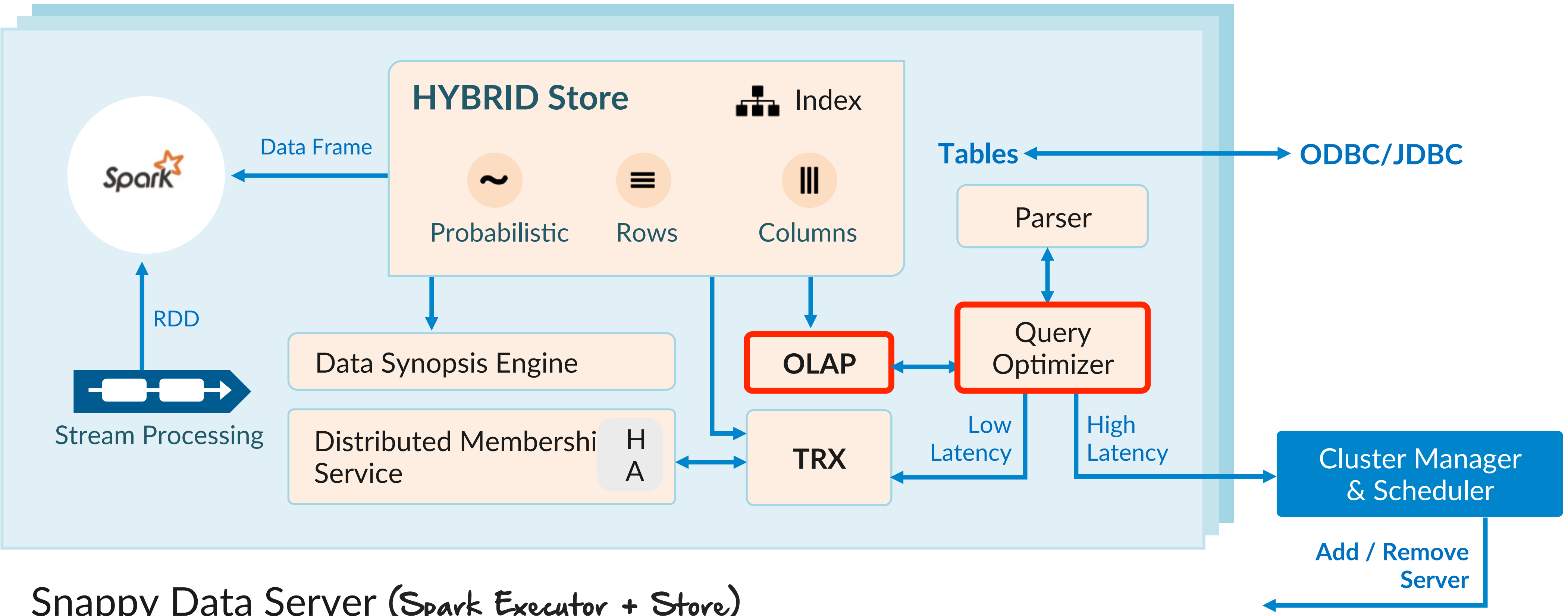
- **Spark Executors are long running.** Driver failure doesn't shutdown Executors
- **Driver HA** – Drivers are “Managed” by SnappyData with standby secondary
- **Data HA** – Consensus based clustering integrated for eager replication



Overview


HYBRID Store
Index
Data Frame
Probabilistic
Rows
Columns
Tables
ODBC/JDBC
Parser
RDD
Data Synopsis Engine
OLAP
Query Optimizer
Stream Processing
Distributed Membershi Service
H A
TRX
Low Latency
High Latency
Cluster Manager & Scheduler
Add / Remove Server
Snappy Data Server (Spark Executor + Store)

Overview

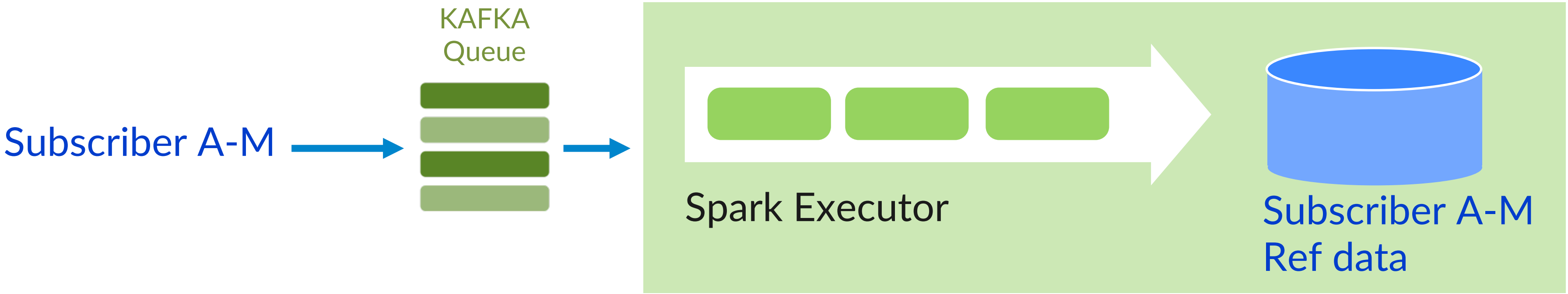


Snappy Data Server (Spark Executor + Store)

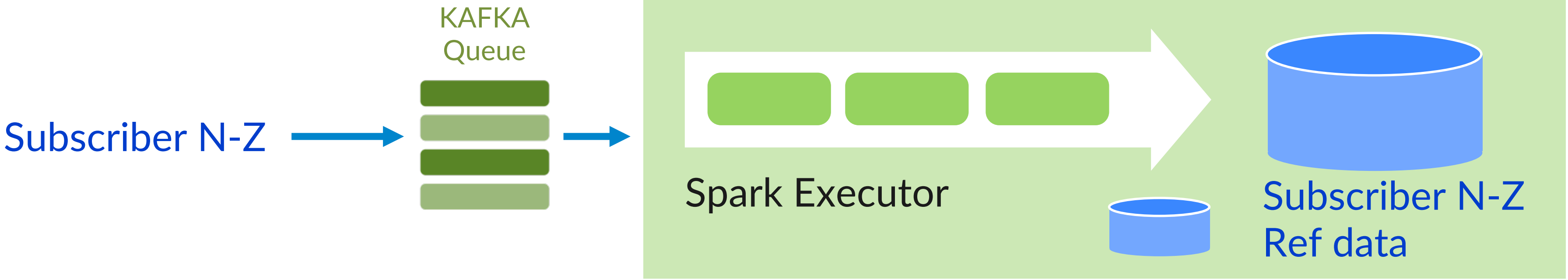
Query Optimization

- **Bypass the scheduler for transactions and low-latency jobs**
- **Minimize shuffles aggressively**
  - Dynamic replication for reference data
  - Retain ‘join indexes’ whenever possible
  - Collocate and co-partition related tables and streams
- **Optimized ‘Hash Join’, ‘Scan’, ‘GroupBy’ compared to Spark**
  - Use more variables (eliminate virtual funcs) to generate better code
  - Use vectorized structures
  - Avoid Spark’s single-node bottlenecks in broadcast joins
- **Column segment pruning through metadata**

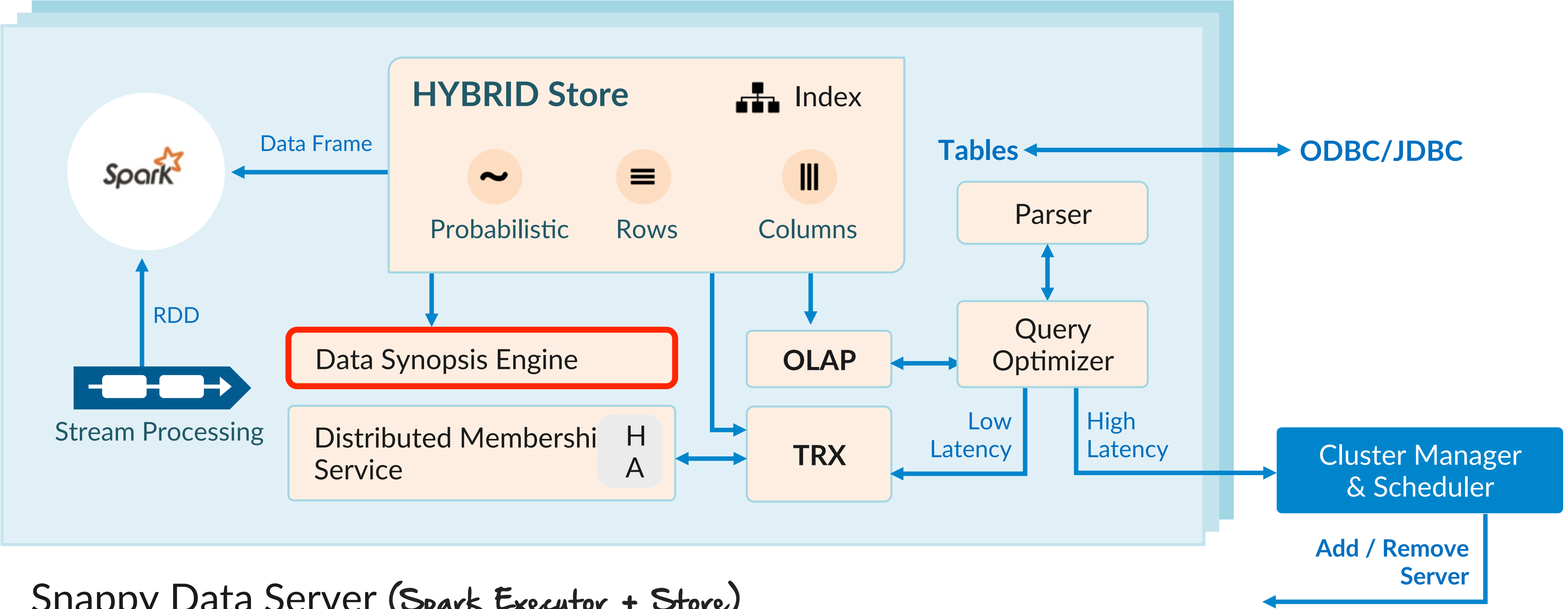
Co-partitioning & Co-location



Linearly scale with partition pruning



Overview



Snappy Data Server (Spark Executor + Store)

Approximate Query Processing (AQP): Academia vs. Industry

25+ yrs of successful research in academia

AQUA, Online Aggregation, MapReduce Online, STRAT, ABS, BlinkDB / G-OLA, ...

BUT:

User-facing AQP almost non-existent in commercial world!

Some approximate features in Infobright, Yahoo's Druid, Facebook's Presto, Oracle 12C, ...

WHY ?

Approximate Query Processing (AQP): Academia vs. Industry

25+ yrs of successful research in academia

AQUA, Online Aggregation, MapReduce Online, STRAT, ABS, BlinkDB / G-OLA, ...

BUT:

User-facing AQP almost non-existent in commercial world!

Some approximate features in Infobright, Yahoo's Druid, Facebook's Presto, Oracle 12C, ...

WHY ?

select geo, avg(bid)
from adImpressions
group by geo having avg(bid)>10
with error 0.05 at confidence 95

geo	avg(bid)	error	prob_existence
MI	21.5	± 0.4	0.99
CA	18.3	± 5.1	0.80
MA	15.6	± 2.4	0.81
...	...	...	....

Approximate Query Processing (AQP): Academia vs. Industry

25+ yrs of successful research in academia

AQUA, Online Aggregation, MapReduce Online, STRAT, ABS, BlinkDB / G-OLA, ...

BUT:

User-facing AQP almost non-existent in commercial world!

Some approximate features in Infobright, Yahoo's Druid, Facebook's Presto, Oracle 12C, ...

WHY ?

```
select geo, avg(bid)
from adImpressions
group by geo having avg(bid)>10
with error 0.05 at confidence 95
```

geo	avg(bid)	error	prob_existence
MI	21.5	± 0.4	0.99
CA	18.3	± 5.1	0.80
MA	15.6	± 2.4	0.81
...	...	...	....

1. Incompatible w/ BI tools
2. Complex semantics
3. Bad sales pitch!

A First Industrial-Grade AQP Engine

1. Highlevel Accuracy Contract (HAC)

- User picks a single number p, where 0≤p≤1 (by default p=0.95)
- Snappy guarantees that s/he only sees things that are at least p% accurate
- Snappy handles (and hides) everything else!

geo	avg(bid)
MI	21.5
WI	42.3
NY	65.6
…	…

A First Industrial-Grade AQP Engine

1. Highlevel Accuracy Contract (HAC)

- User picks a single number p, where $0 \leq p \leq 1$ (by default p=0.95)
- Snappy guarantees that s/he only sees things that are at least p% accurate
- Snappy handles (and hides) everything else!

geo	avg(bid)
MI	21.5
WI	42.3
NY	65.6
…	…

2. Fully compatible w/ BI tools

- Set HAC behavior at JDBC/ODBC connection level

tableau ZOOMDATA

A First Industrial-Grade AQP Engine

1. Highlevel Accuracy Contract (HAC)

- User picks a single number p, where 0≤p≤1 (by default p=0.95)
- Snappy guarantees that s/he only sees things that are at least p% accurate
- Snappy handles (and hides) everything else!

geo	avg(bid)
MI	21.5
WI	42.3
NY	65.6
…	…

2. Fully compatible w/ BI tools

- Set HAC behavior at JDBC/ODBC connection level

tableau

ZOOMDATA

3. Better marketing!

- Concurrency: 10's of queries in shared clusters
- Resource usage: everyone hates their AWS bill
- Network shuffles
- Immediate results while waiting for final results

iSight (Immediate inSight)

Benchmarks

Benchmarks

- Mixed Benchmark: Ad Analytics

Streaming
- Ad impressions arrive on a message bus
- Aggregate by publisher and geo
- Report avg bid, # of impressions, and # of uniques every few secs

Trx
- Write to a partitioned store
- Transactionally update the profiles during ingestion

OLAP
- Q1: Top-20 ads receiving most impressions per region
- Q2: Top-20 ads receiving largest bids per geo
- Q3: Top-20 publishers receiving largest sum of bids overall

- TPC - H

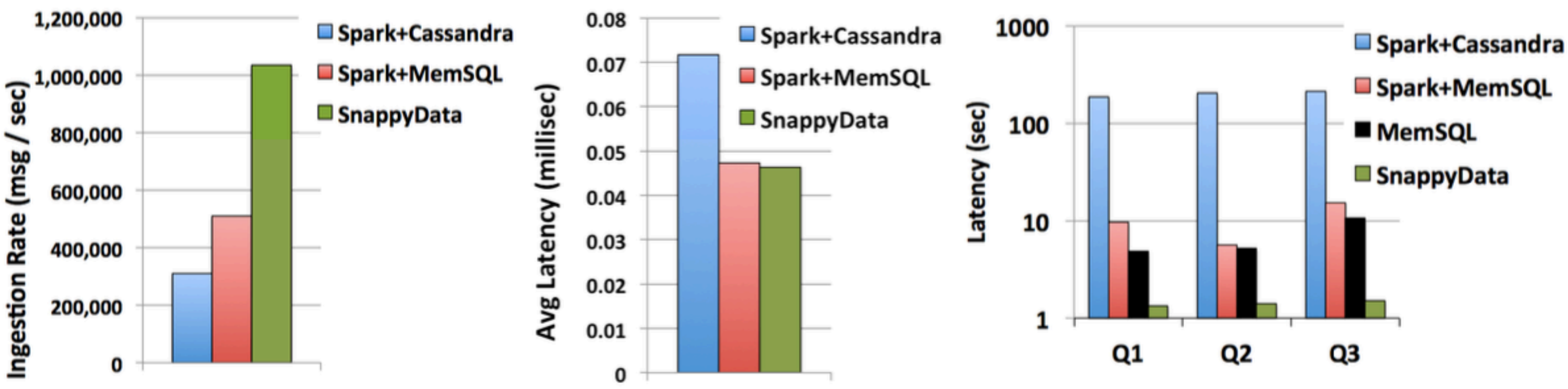
Setup

- HW: 5 c4.2xlarge EC2 instances
  - 1 coordinator + 4 workers
- Software: Latest GA versions avalable:
  - Kafka 2.10_0.8.2.2
  - Spark 2.0.0
  - Cassandra 3.9
    w/ Spark-Cassandra connector 2.0.0_M3
  - MemSQL Ops-5.5.10 Community Edition
    w/ Spark-MemSQL Connector 2.10_1.3.3
  - SnappyData 0.6.1

Ad Analytics

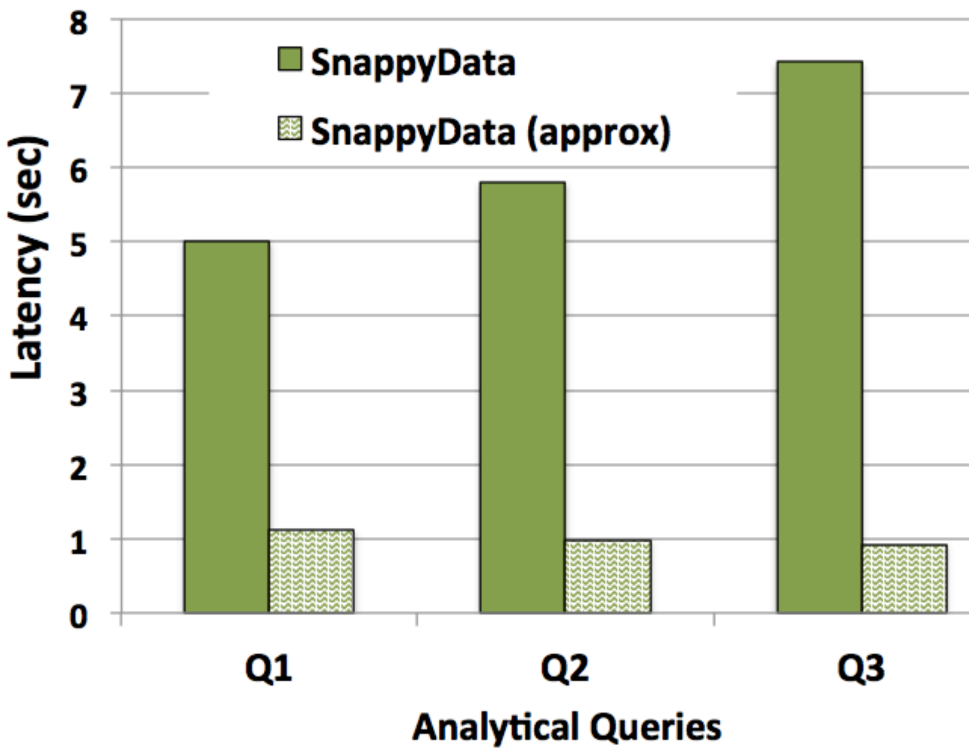
1.5-2x faster ingestion

7-142× faster analytics (at 300M records)



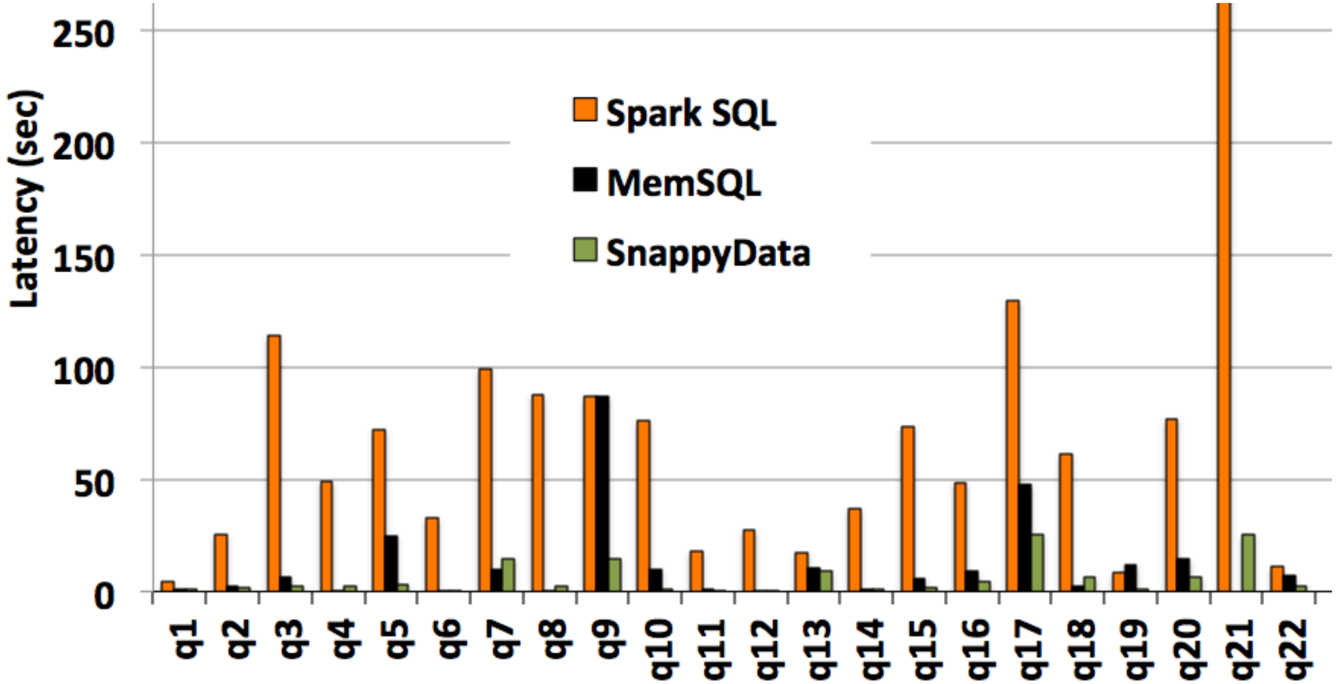
(a) Streaming (b) Transactions (c) Analytical Queries

Data Synopsis Engine



TPC-H

100 GB	Avg	Median
SnappyData	5.7s	2.5s
MemSQL	12.0s	6.4s
Spark	66.9s	55.1s



SnappyData was 2.6x faster than MemSQL & 22.4x faster than Spark 2.0

Conclusion

Where Are We Today ?

- Current customers
  - Investment banking, Industrial IoT, Telco, Ad Analytics, & healthcare
- Current release
  - 0.7 (GA 1.0 in Q1-2017)
- Next funding round
  - Q2-2017
- Upcoming features
  - Integration of Spark ML w/ our Data Synopsis Engine
  - Cost-based query optimizer
  - Physical designer & workload miner (http://CliffGuard.org)

Lessons Learned

1. A unique experience marryings two different breeds of distributed systems

lineage-based for high-throughput vs. (consensus-) replication-based for low-latency

Lessons Learned

1. A unique experience marryings two different breeds of distributed systems

lineage-based for high-throughput vs. (consensus-) replication-based for low-latency

2. A unified cluster is simpler, cheaper, and faster

- By sharing state across apps, we decouple apps from data servers and provide HA
- Save memory, data copying, serialization, and shuffles
- Co-partitioning and co-location for faster joins and stream analytics

Lessons Learned

1. A unique experience marryings two different breeds of distributed systems

lineage-based for high-throughput vs. (consensus-) replication-based for low-latency

2. A unified cluster is simpler, cheaper, and faster

- By sharing state across apps, we decouple apps from data servers and provide HA
- Save memory, data copying, serialization, and shuffles
- Co-partitioning and co-location for faster joins and stream analytics

3. Advantages over HTAP engines: Deep stream integration + AQP

- Stream processing ≠ stream analytics
- Top-k w/ almost arbitrary predicates + 1-pass stratified sampling over streams

Lessons Learned

1. A unique experience marryings two different breeds of distributed systems

lineage-based for high-throughput vs. (consensus-) replication-based for low-latency

2. A unified cluster is simpler, cheaper, and faster

- By sharing state across apps, we decouple apps from data servers and provide HA
- Save memory, data copying, serialization, and shuffles
- Co-partitioning and co-location for faster joins and stream analytics

3. Advantages over HTAP engines: Deep stream integration + AQP

- Stream processing ≠ stream analytics
- Top-k w/ almost arbitrary predicates + 1-pass stratified sampling over streams

4. Commercializing academic work is lots of work but also lots of fun

Try our iSight cloud for free:
http://snappydata.io/iSight

THANK YOU !