DRUID: REAL-TIME QUERIES MEET REAL-TIME DATA

ERIC TSCHETTER - LEAD ARCHITECT
@ METAMARKETS

DEMO

WHAT WE TRIED

WHAT WE TRIED

I. RDMBS - Relational Database

I. RDBMS - THE SETUP

- Star Schema
- Aggregate Tables
- Query Caching

I. RDBMS - THE RESULTS

- Queries that were cached
  - fast
- Queries against aggregate tables
  - fast to acceptable
- Queries against base fact table
  - generally unacceptable

I. RDBMS - PERFORMANCE

Naive benchmark scan rate	~5.5M rows / second / core
1 day of summarized aggregates	60M+ rows
1 query over 1 week, 16 cores	~5 seconds
Page load with 20 queries over a week of data	long time

WHAT WE TRIED

I. RDMBS - Relational Database

WHAT WE TRIED

I. ~~RDMBS - Relational Database~~

WHAT WE TRIED

I. ~~RDMBS - Relational Database~~

II. NoSQL - Key/Value Store

II. NOSQL - THE SETUP

- Pre-aggregate all dimensional combinations
- Store results in a NoSQL store

II. NOSQL - THE RESULTS

- Queries were fast
  - range scan on primary key
- Inflexible
  - not aggregated, not available
- Not continuously updated
  - aggregate first, then display
- Processing scales exponentially

II. NOSQL - PERFORMANCE

- Dimensional combinations => exponential increase
- Tried limiting dimensional depth
  - still expands exponentially
- Example: ~500k records
  - 11 dimensions, 5-deep
    - 4.5 hours on a 15-node Hadoop cluster
  - 14 dimensions, 5-deep
    - 9 hours on a 25-node Hadoop cluster

WHAT WE TRIED

I. ~~RDMBS - Relational Database~~

II. NoSQL - Key/Value Store

WHAT WE TRIED

I. ~~RDMBS - Relational Database~~

II. ~~NoSQL - Key/Value Store~~

WHAT WE TRIED

~~I. RDMBS - Relational Database~~

~~II. NoSQL - Key/Value Store~~

III. ???

WHAT WE LEARNED

- Problem with RDBMS: scans are slow
- Problem with NoSQL: computationally intractable

WHAT WE LEARNED

- Problem with RDBMS: scans are slow
- Problem with NoSQL: computationally intractable

- Tackling RDBMS issue seems easier

INTRODUCING DRUID


Honey

DRUID – FOUR KEY FEATURES

1.Distributed Column-Store

2.Fast Data Scans

3.Fast Filtering

4.Real-time Ingestion

FEATURE 1 – DISTRIBUTED COLUMN-STORE

- Data chunked into “segments”
  - MVCC swapping protocol
- Converted to columnar format
- Coordinator oversees cluster
  - Data replication and balance
- Per-segment replication
  - Increase replication on hot spots

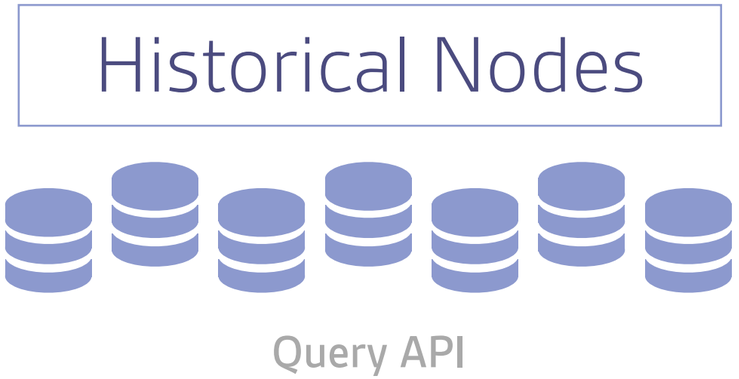
FEATURE 2 – FAST DATA SCANS

- Column orientation
  - Only load/scan what you need
- Compression
  - Decrease size of data in storage (RAM/disk)

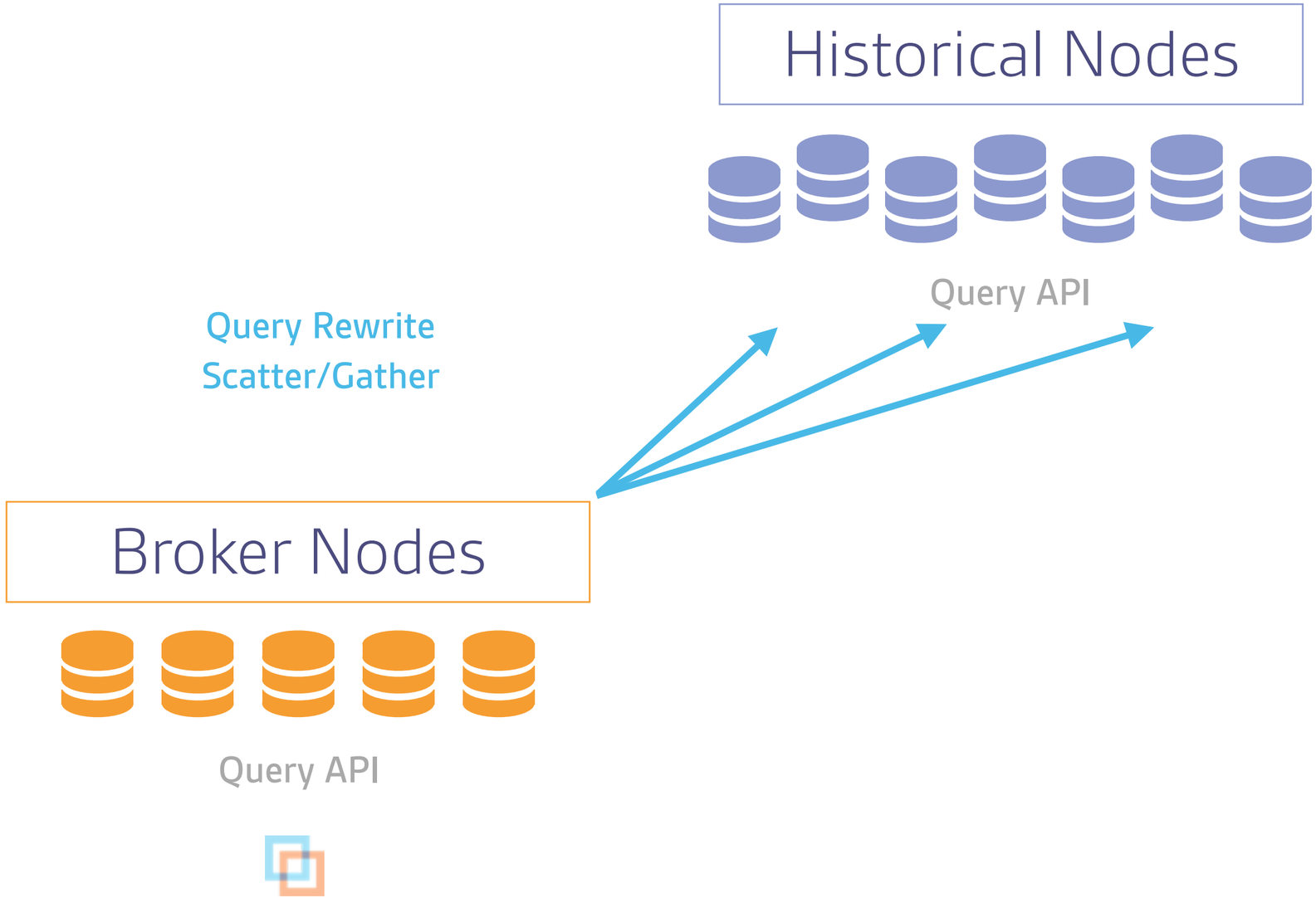
FEATURE 3 – FAST FILTERING

- Indexed
  - Bitmap indices
- Compressed Bitmaps
  - CONCISE compression
  - Operate on compressed form
  - Resolve filters without looking at data

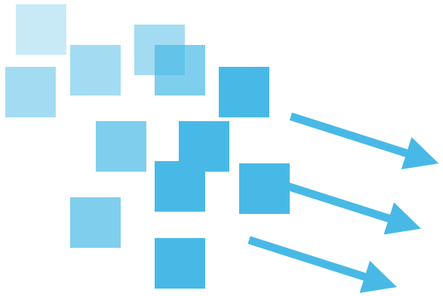
FEATURE 4 – REAL-TIME INGESTION



FEATURE 4 – REAL-TIME INGESTION



FEATURE 4 – REAL-TIME INGESTION



Real-time Nodes

Query API

Query Rewrite
Scatter/Gather

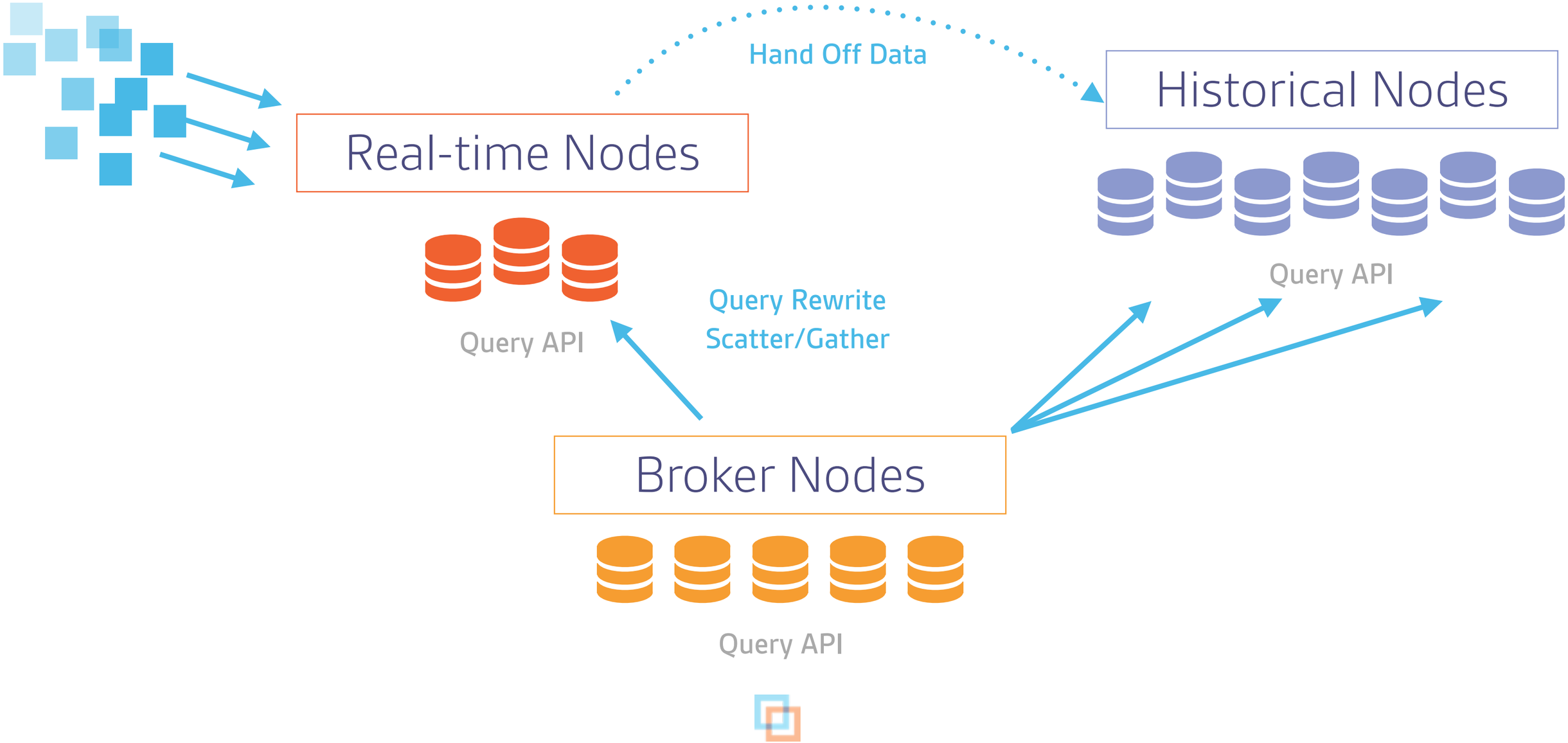
Historical Nodes

Query API

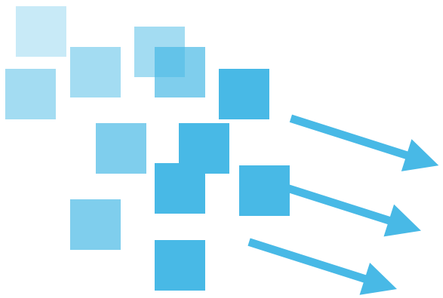
Broker Nodes

Query API

FEATURE 4 – REAL-TIME INGESTION



FEATURE 4 – REAL-TIME INGESTION



Real-time Nodes

Query API

Hand Off Data

Query Rewrite
Scatter/Gather

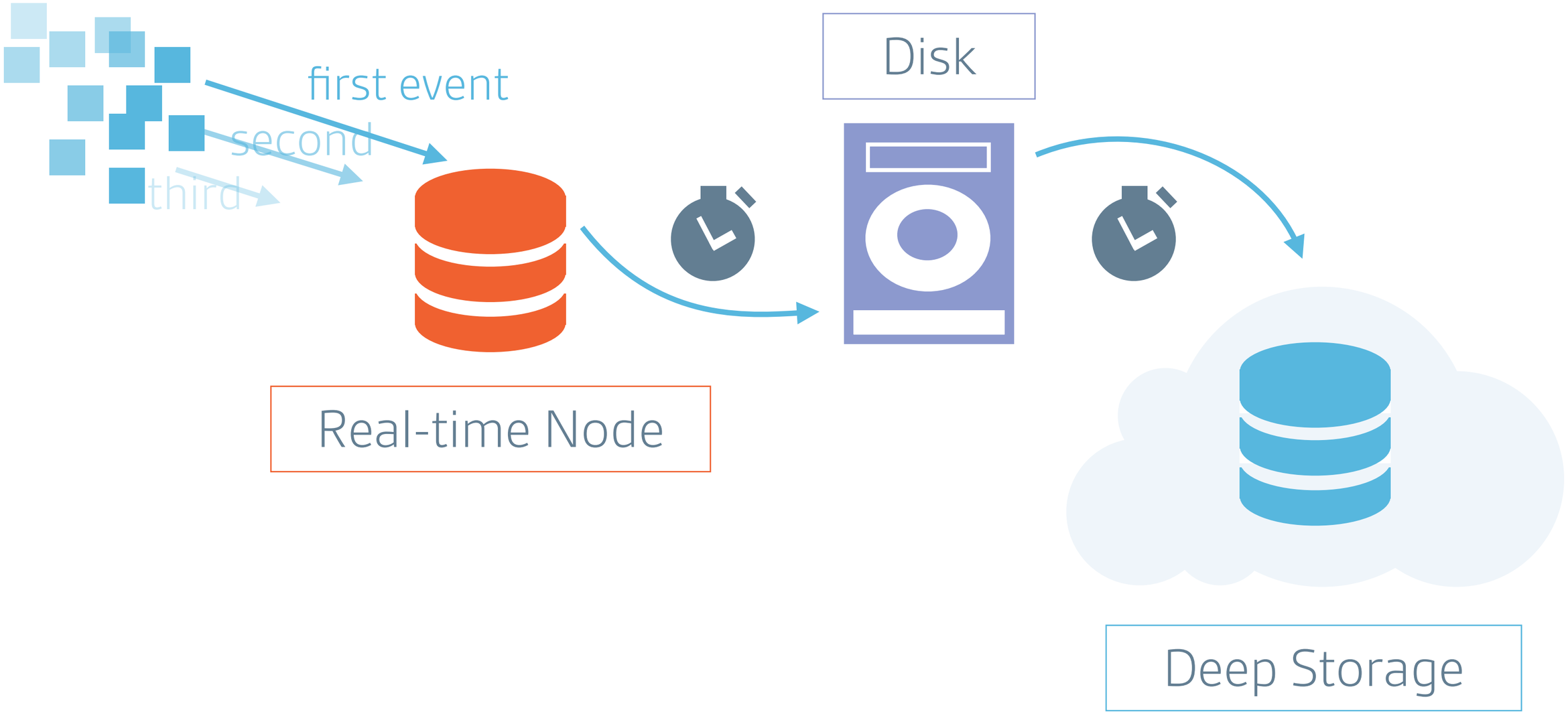
Historical Nodes

Query API

Broker Nodes

Query API

FEATURE 4: REAL-TIME INGESTION



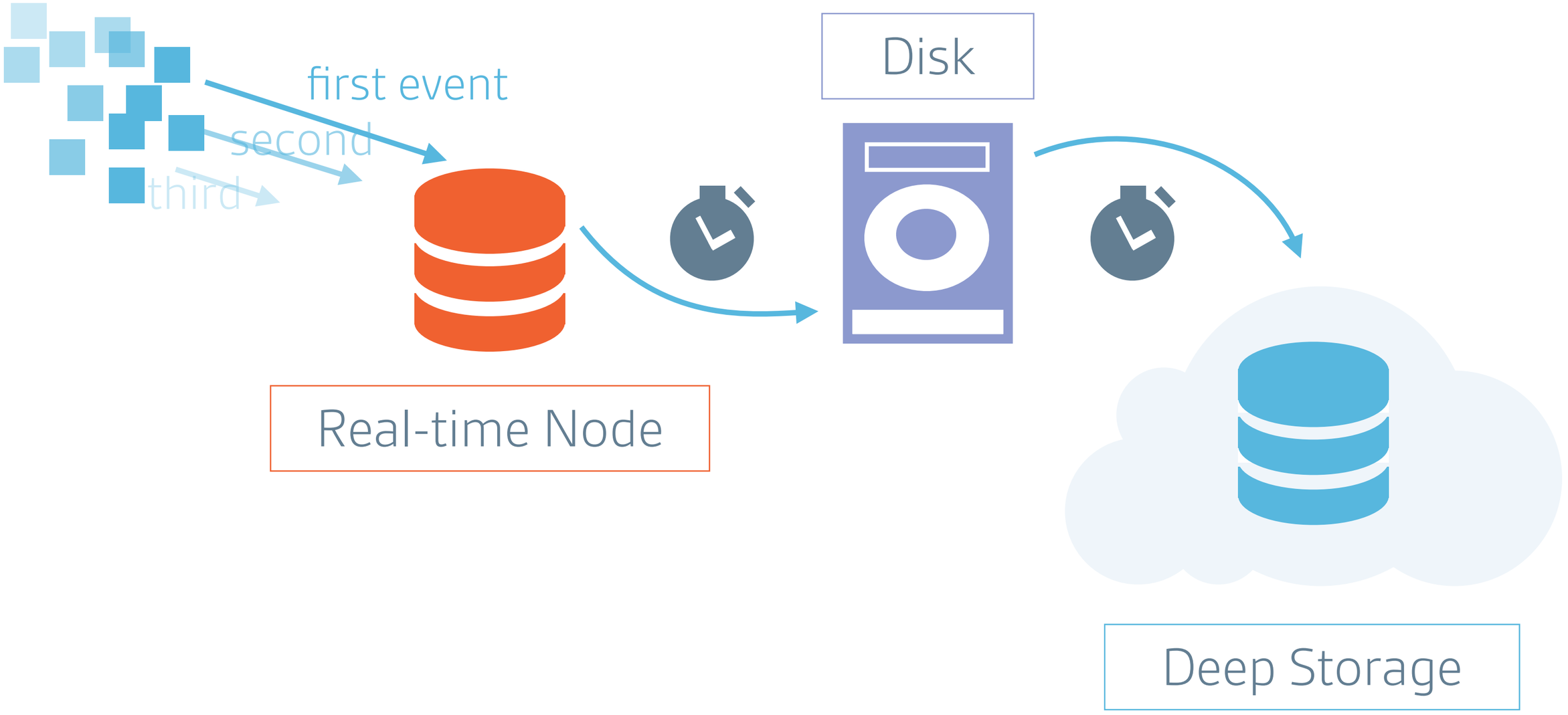
METAMARKETS

2013

REAL-TIME INGESTION: AVAILABILITY

- Real-time data availability
  - Persisted to local disk at configurable time window
  - Merged and persisted to deep storage at (wider) time window
- Failure Scenarios
  - Lose process
    - Start back up and re-loads data from disk
  - Lose machine and disks
    - Data might be lost

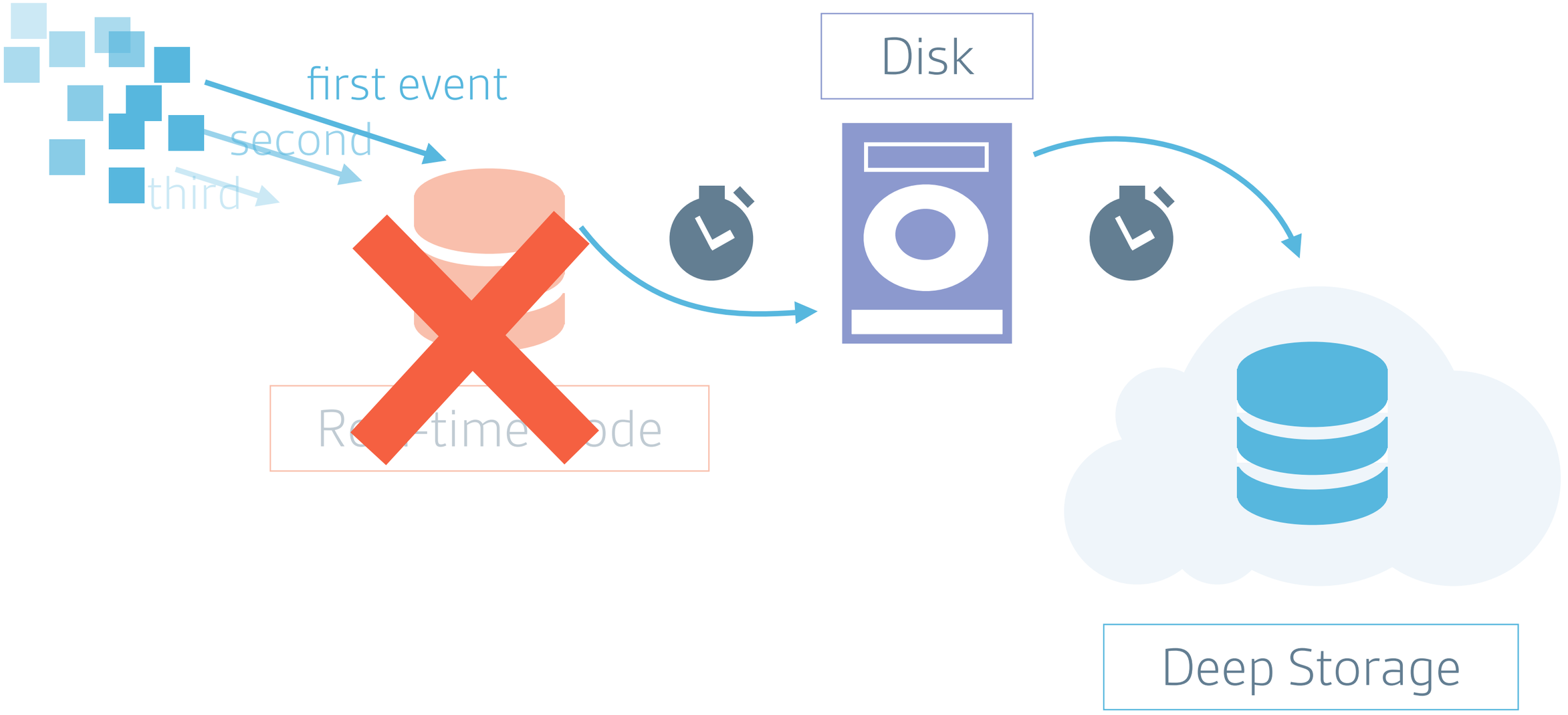
DRUID INGESTION LATENCY: AVAILABILITY



METAMARKETS

2013

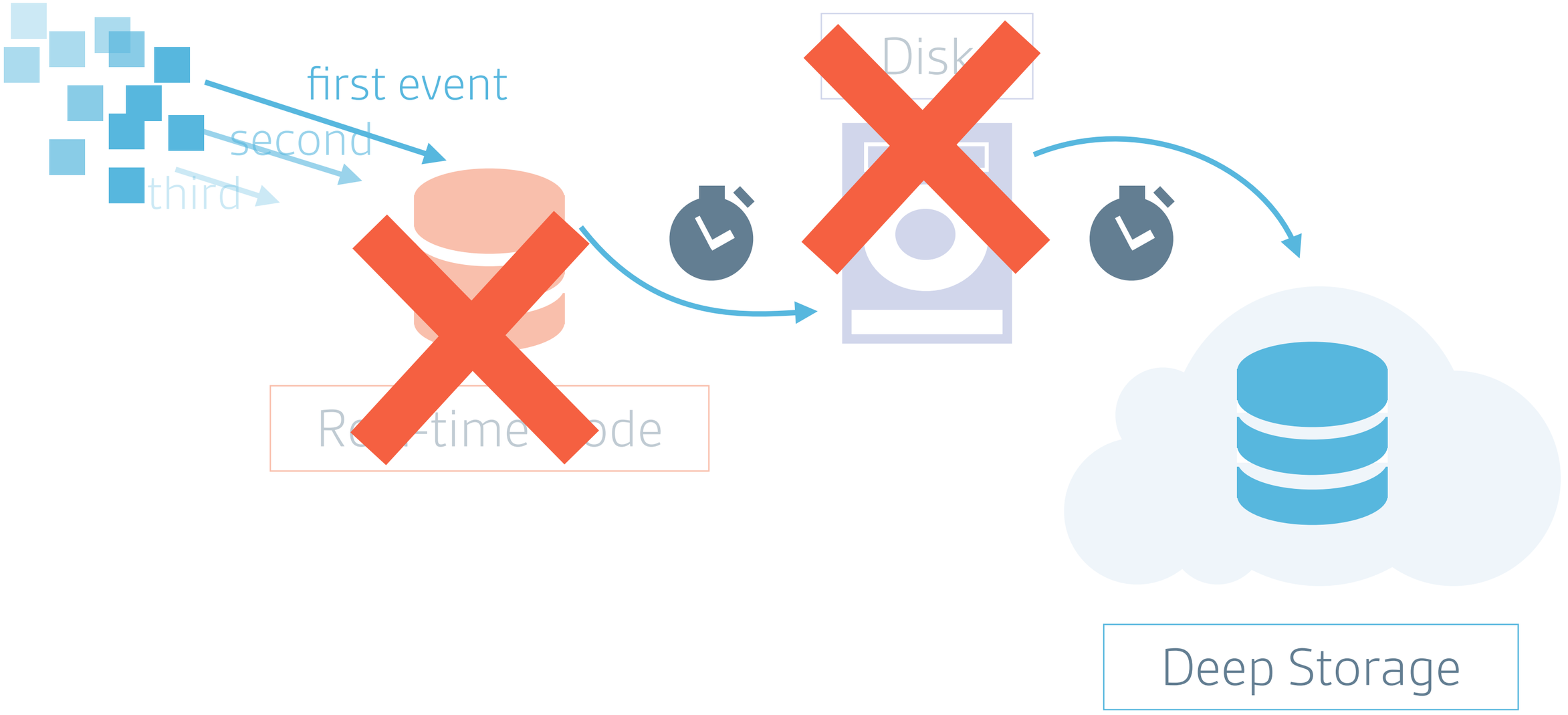
DRUID INGESTION LATENCY: AVAILABILITY



METAMARKETS

2013

DRUID INGESTION LATENCY: AVAILABILITY



METAMARKETS

2013

DRUID INGESTION LATENCY: MESSAGE BUS

- Key features of message bus
  - Message committing
  - Replay of messages
  - Consuming same feed multiple times (replication)

METAMARKETS

OPERATIONS

- Fault-tolerant
  - “Historical” Compute - Replication, Rebalancing
  - Real-time - Deployment-time configuration
- Rolling deployments/restarts
  - All node types have failover/replication
- Grow == start processes
- Shrink == kill processes

DRUID BENCHMARKS

- Scan speed
  - ~33M rows / second / core
- Realtime ingestion rate
  - ~10k records / second / node
- 100 nodes, 6 TB Memory:
  1.4 second in-memory query
- Partner cluster, real-time ingestion:
  150k events/s (17B events/day), 500MB/s (2 TB/h)

DRUID IS OPEN SOURCE

DRUID IS NOW OPEN SOURCE

Learn more:

METAMARKETS.COM/DRUID

GITHUB.COM/METAMX/DRUID

DRUID IS NOW OPEN SOURCE

Learn more:

METAMARKETS.COM/DRUID

GITHUB.COM/METAMX/DRUID

eric tschetter - cheddar@metamarkets.com

@ zedruid