PARTITION DATA

timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	en	SF	USA		10	65
2011-01-01T00:03:63Z	Justin Bieber	en	SF	USA		15	62
2011-01-01T00:04:51Z	Justin Bieber	en	SF	USA		32	45

Segment 2011-01-01T00/2011-01-01T01

2011-01-01T01:00:00Z	Ke$ha	en	Calgary	CA		17	87

Segment 2011-01-01T01/2011-01-01T02

2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		43	99
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		12	53

Segment 2011-01-01T02/2011-01-01T03

- Shard data by time
- Immutable chunks of data called “segments”

IMMUTABLE SEGMENTS

- Fundamental storage unit in Druid
- No contention between reads and writes
- One thread scans one segment
- Multiple threads can access same underlying data

COLUMNAR STORAGE

timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	en	SF	USA		10	65
2011-01-01T00:03:63Z	Justin Bieber	en	SF	USA		15	62
2011-01-01T00:04:51Z	Justin Bieber	en	SF	USA		32	45
2011-01-01T01:00:00Z	Ke$ha	en	Calgary	CA		17	87
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		43	99
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		12	53
...							

- Scan/load only what you need
- Compression!
- Indexes!

COLUMN COMPRESSION - DICTIONARIES

timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	en	SF	USA		10	65
2011-01-01T00:03:63Z	Justin Bieber	en	SF	USA		15	62
2011-01-01T00:04:51Z	Justin Bieber	en	SF	USA		32	45
2011-01-01T01:00:00Z	Ke$ha	en	Calgary	CA		17	87
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		43	99
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		12	53
...							

- Create ids
  - Justin Bieber -> 0, Ke$ha -> 1
- Store
  - page -> **[0 0 0 1 1 1]**
  - language -> **[0 0 0 0 0 0]**

BITMAP INDICES

timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	en	SF	USA		10	65
2011-01-01T00:03:63Z	Justin Bieber	en	SF	USA		15	62
2011-01-01T00:04:51Z	Justin Bieber	en	SF	USA		32	45
2011-01-01T01:00:00Z	Ke$ha	en	Calgary	CA		17	87
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		43	99
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		12	53
...							

- Justin Bieber -> [0, 1, 2] -> [111000]
- Ke$ha -> [3, 4, 5] -> [000111]

FAST AND FLEXIBLE QUERIES

row	page
0	Justin Bieber
1	Justin Bieber
2	Ke$ha
3	Ke$ha

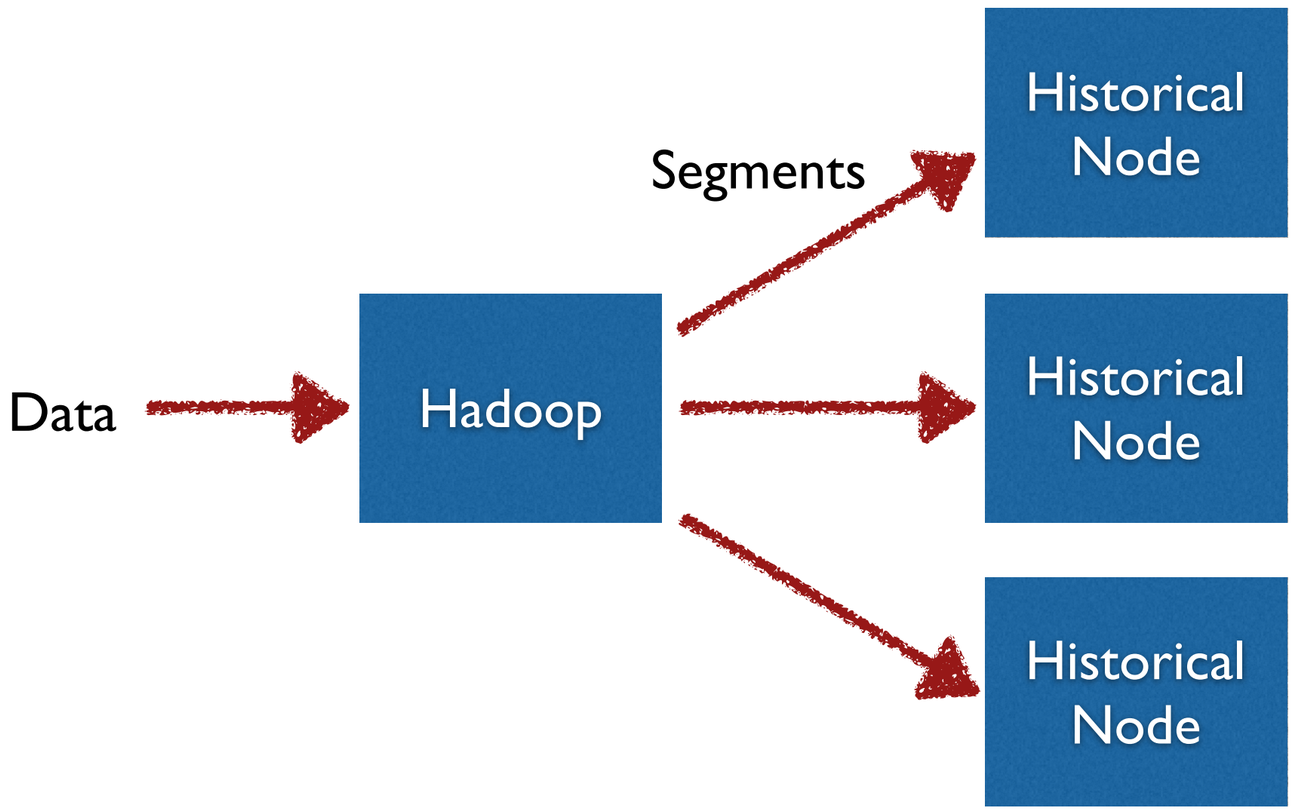
JUSTIN BIEBER
[1, 1, 0, 0]

KE$HA
[0, 0, 1, 1]

JUSTIN BIEBER
OR
KE$HA
[1, 1, 1, 1]

ARCHITECTURE

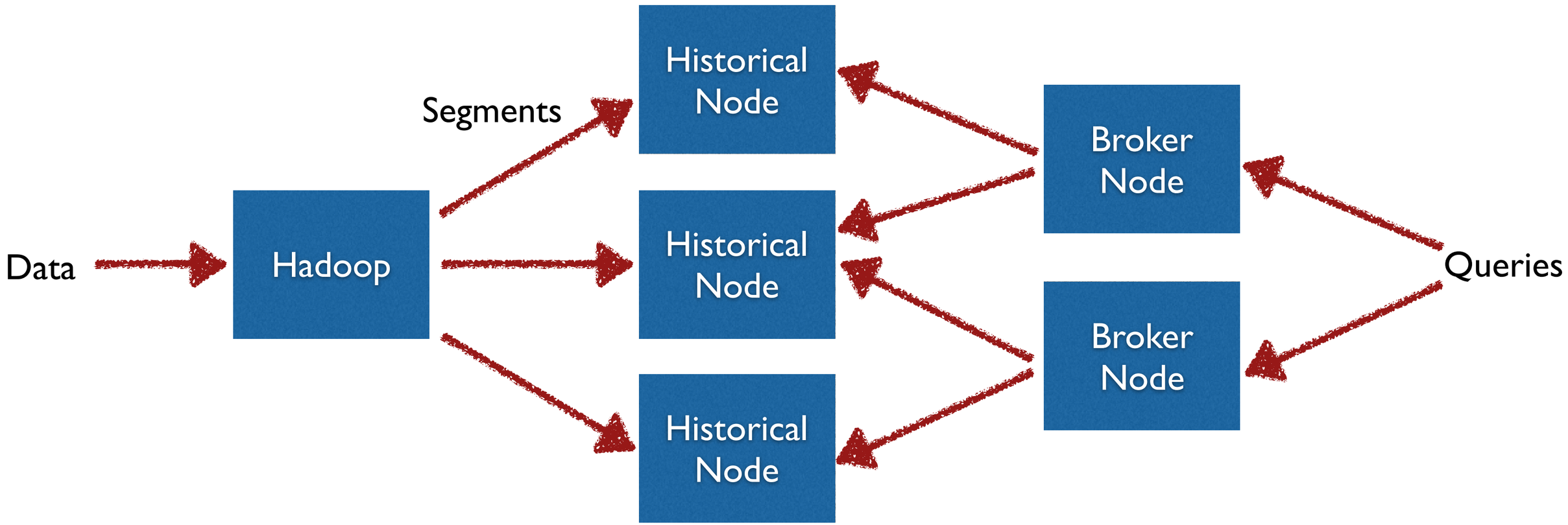
ARCHITECTURE (BATCH ONLY)



HISTORICAL NODES

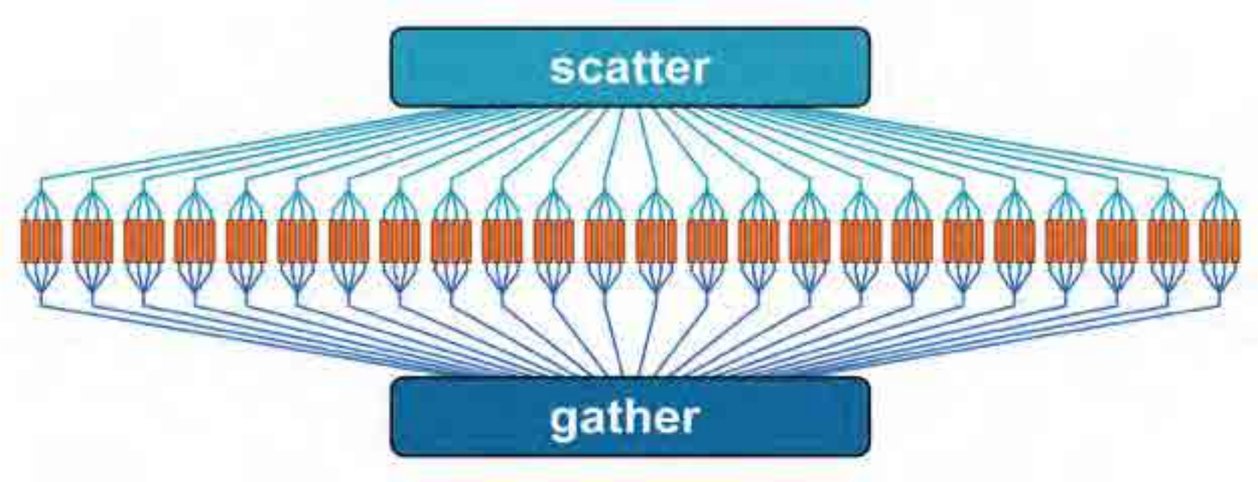
- Main workhorses of a Druid cluster
- Scan segments
- Shared-nothing architecture

ARCHITECTURE (BATCH ONLY)

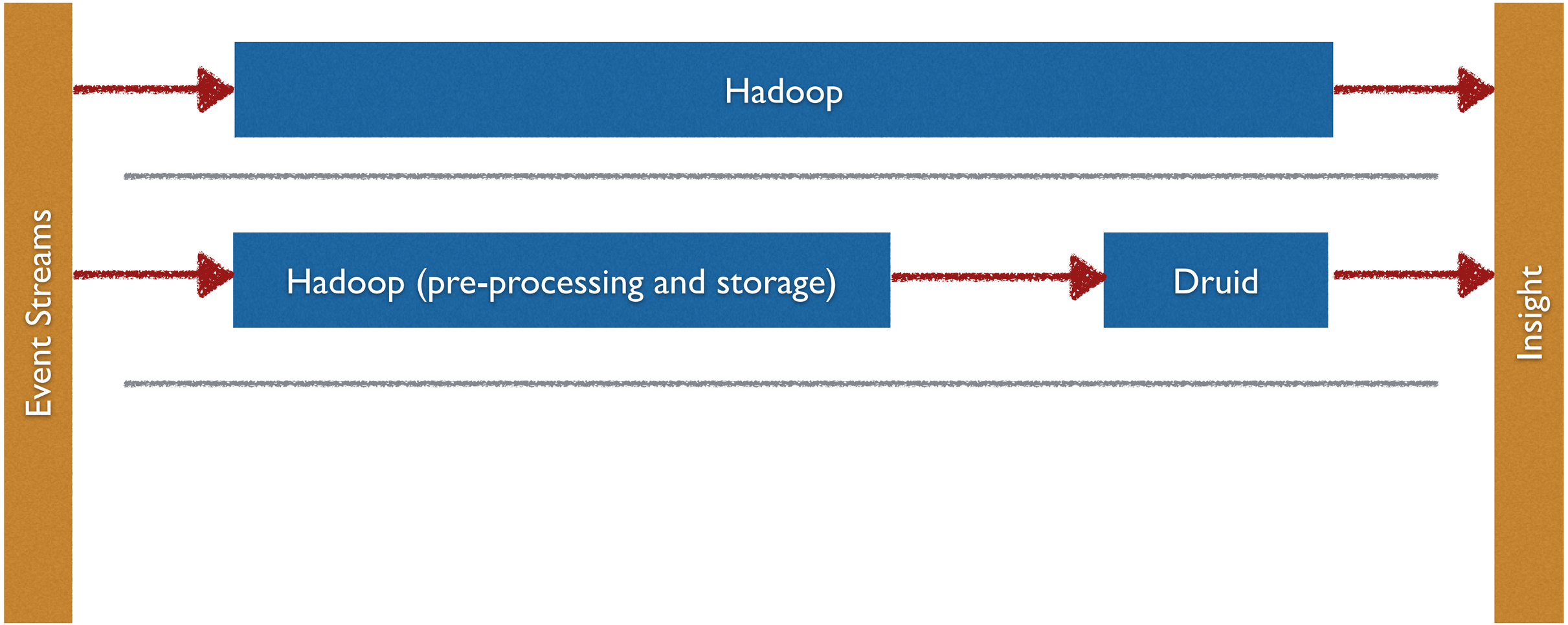


BROKER NODES

- Knows which nodes hold what data
- Query scatter/gather (send requests to nodes and merge results)
- Caching



EVOLVING A SOLUTION



MORE PROBLEMS

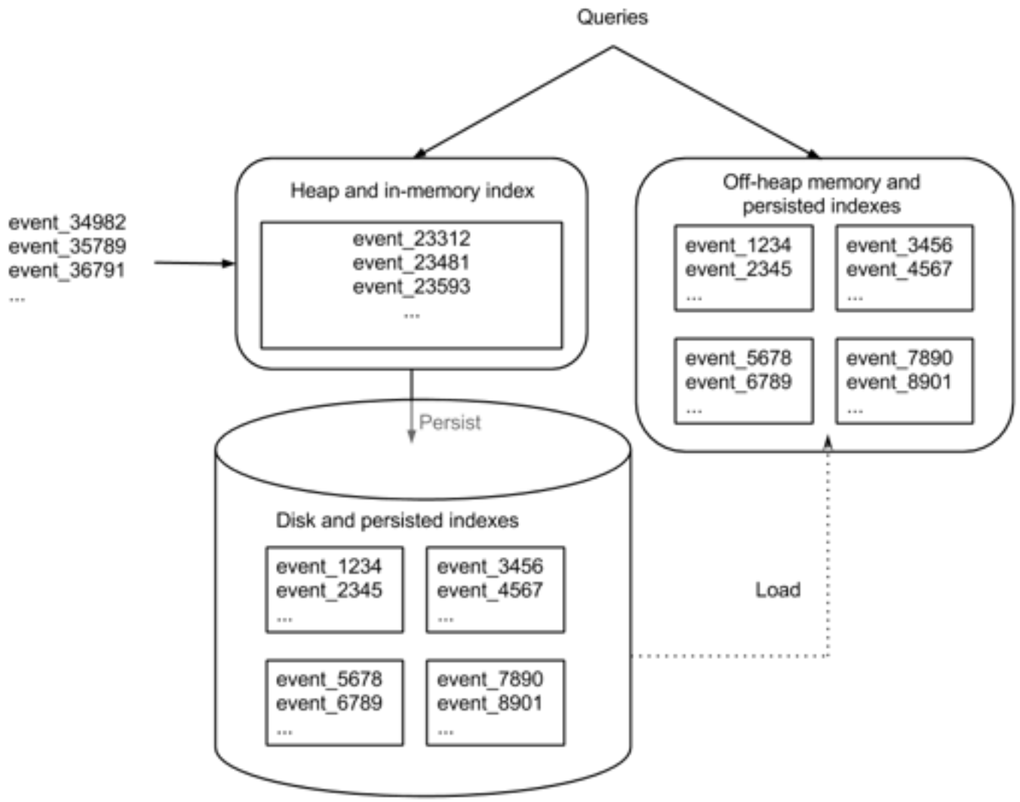
- We’ve solved the query problem
  - Druid gave us arbitrary data exploration & fast queries
- But what about data freshness?
  - Batch loading is slow!
  - We want “real-time”
  - Alerts, operational monitoring, etc.

FAST LOADING WITH DRUID

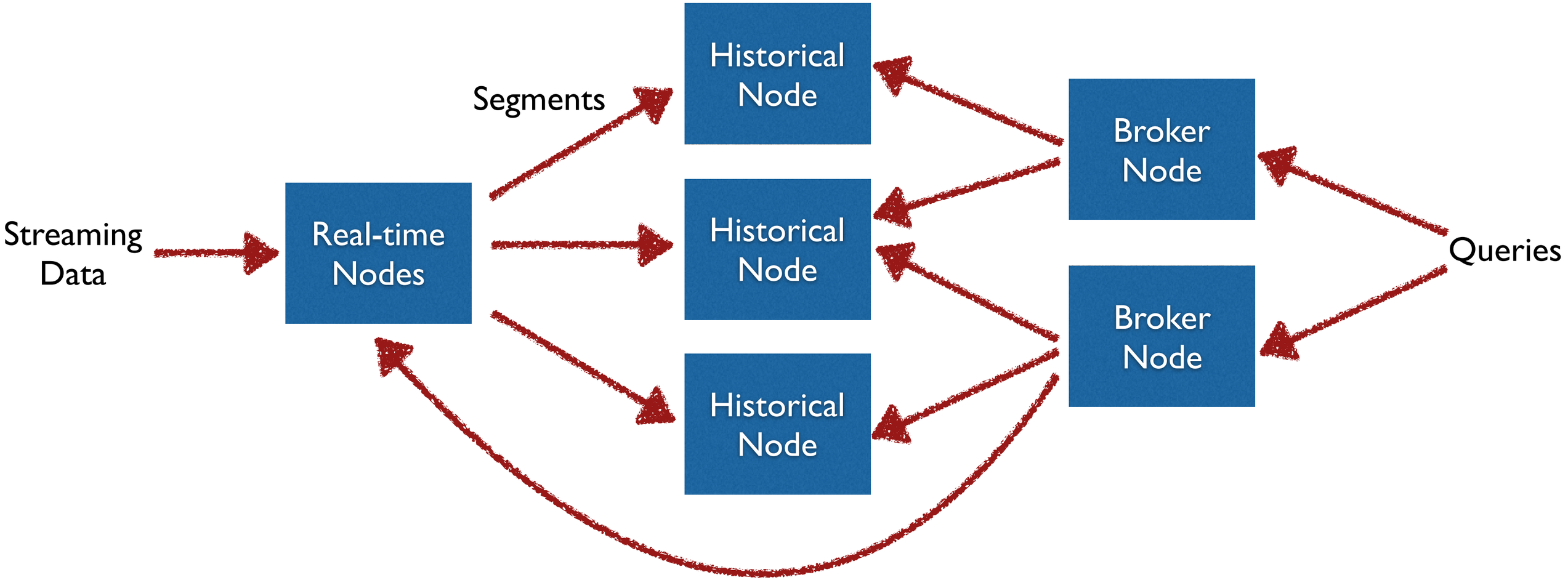
- We have an indexing system
- We have a serving system that runs queries on data
- We can serve queries while building indexes!
- Real-time indexing workers do this

REAL-TIME NODES

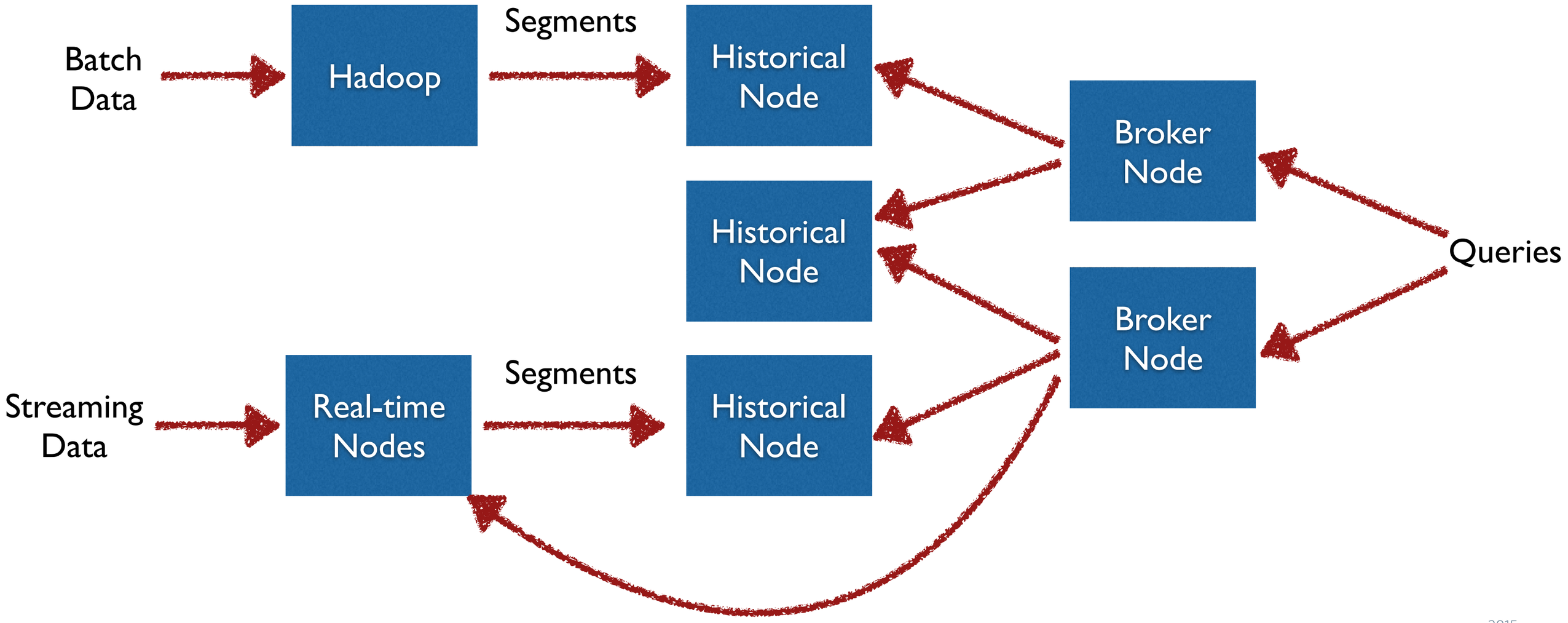
- Write-optimized data structure: hash map in heap
- Read-optimized data structure: Druid segments
- Convert write optimized -> read optimized
- Query data as soon as it is ingested
- Log-structured merge-tree



ARCHITECTURE (STREAMING-ONLY)



ARCHITECTURE (LAMBDA)



APPROXIMATE ANSWERS

- Drastically reduce storage space and compute time
  - Cardinality estimation
  - Histograms and Quantiles
  - Funnel analysis
  - Add your own proprietary modules

PRODUCTION READY

- High availability through replication
- Rolling restarts
  - 4 years of no down time for software updates and restarts
- Battle tested

DRUID TODAY

THE COMMUNITY

- Growing Community
  - 120+ contributors from many different companies
  - In production at many different companies, we're hoping for more!
    - Ad-tech, network traffic, operations, activity streams, etc.
  - We love contributions!

DRUID IN PRODUCTION

REALTIME INGESTION

>3M EVENTS / SECOND SUSTAINED (200B+ EVENTS/DAY)

10 – 100K EVENTS / SECOND / CORE

DRUID IN PRODUCTION

CLUSTER SIZE

>500TB OF SEGMENTS (>30 TRILLION RAW EVENTS)

>5000 CORES (>350 NODES, >100TB RAM)

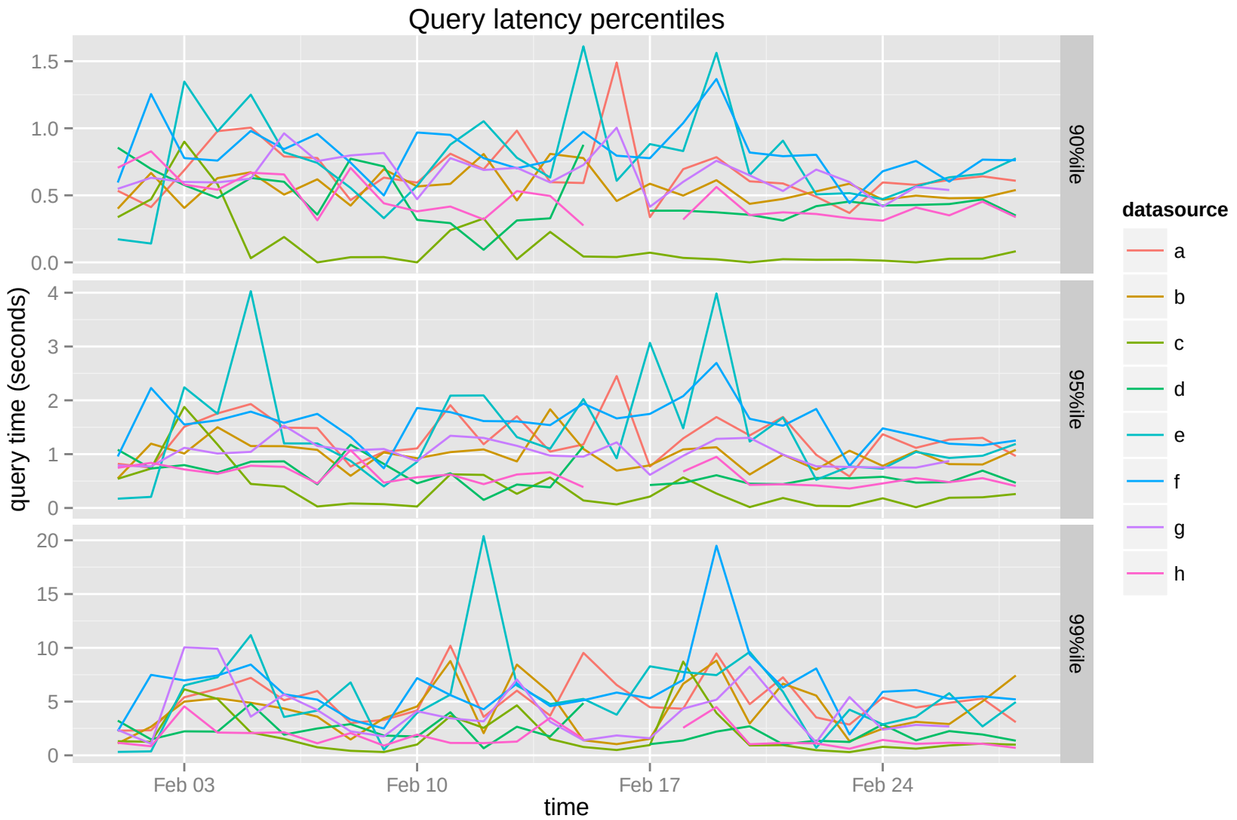
IT'S CHEAP

MOST COST EFFECTIVE AT THIS SCALE

DRUID IN PRODUCTION

QUERY LATENCY (500MS AVERAGE)

90% < 1S 95% < 2S 99% < 10S

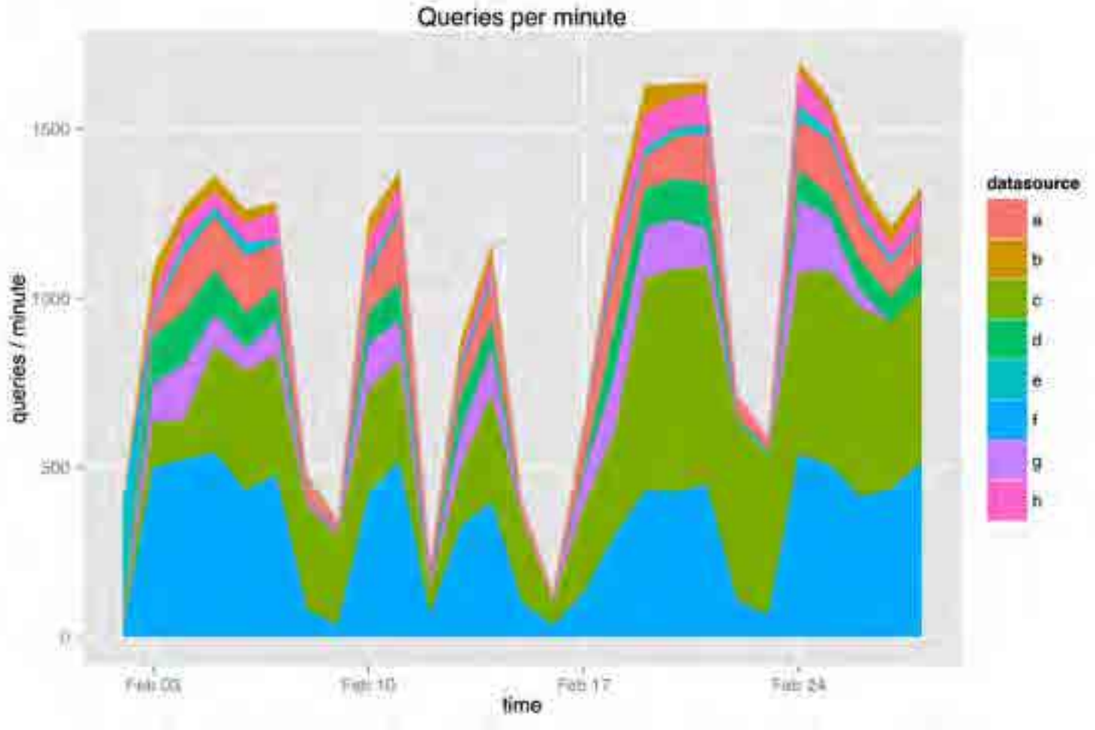


DRUID IN PRODUCTION

QUERY VOLUME

SEVERAL HUNDRED QUERIES / SECOND

VARIETY OF GROUP BY & TOP-K QUERIES



DRUID AND THE DATA INFRASTRUCTURE SPACE