Druid

Power Interactive Applications at Scale

Fangjin Yang

Cofounder @ 

Overview

History & Motivation

Demo

Alternative Architectures

Druid Architecture

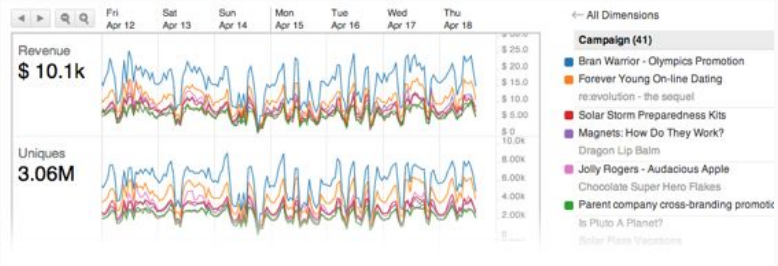
History & Motivation

First lines of Druid started in 2011

Initial use case: power ad-tech analytics product

Initial requirements:

- Scalable
- Flexible
- Interactive (low latency queries)



History & Motivation

Druid went open source in late 2012

- GPL license initially
- Part-time development until early 2014
- Apache v2 licensed in early 2015

More requirements:

- Scalable (trillions of events/day, petabytes of data)
- Multi-tenant (thousands of current users)
- “Real-time” (low latency data ingestion)

Demo

In case the internet didn’t work,
pretend you saw something cool

Powering a Data Application

Business intelligence/OLAP queries

- Time, dimensions, measures
- Filtering, grouping, and aggregating data
- Not dumping entire data set
- Not examining single events
- Result set < input set (aggregations)

Solution Space

Relational databases (MySQL, Postgres)

Key/value stores (HBase, Cassandra)

Column stores

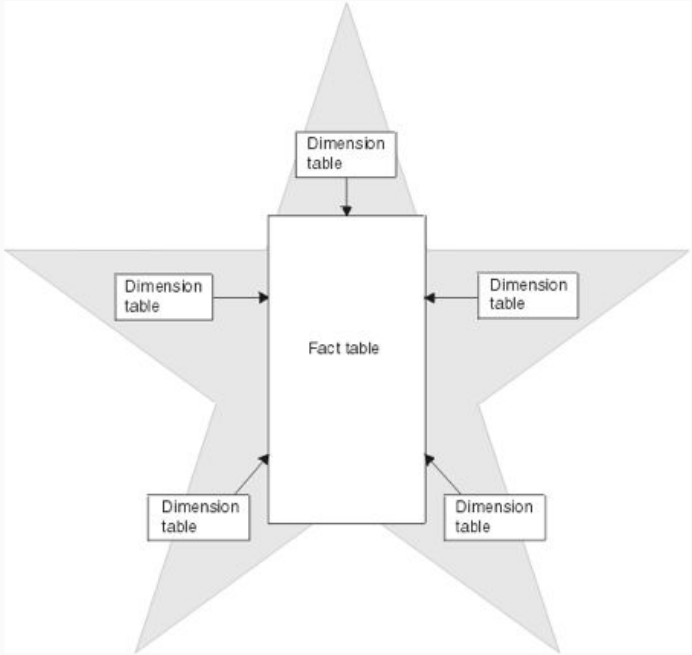
Relational Database

Traditional Data Warehouse

- Row store
- Star schema
- Aggregates tables & query caches

Fast becoming outdated

Slow!



Key/Value Stores

Pre-computation

- Pre-compute every possible query
- Pre-compute a set of queries
- Exponential scaling costs

ts	gender	age	revenue
1	M	18	$0.15
1	F	25	$1.03
1	F	18	$0.01

Key	Value
1	revenue=$1.19
1,M	revenue=$0.15
1,F	revenue=$1.04
1,18	revenue=$0.16
1,25	revenue=$1.03
1,M,18	revenue=$0.15
1,F,18	revenue=$0.01
1,F,25	revenue=$1.03

Key/Value Stores

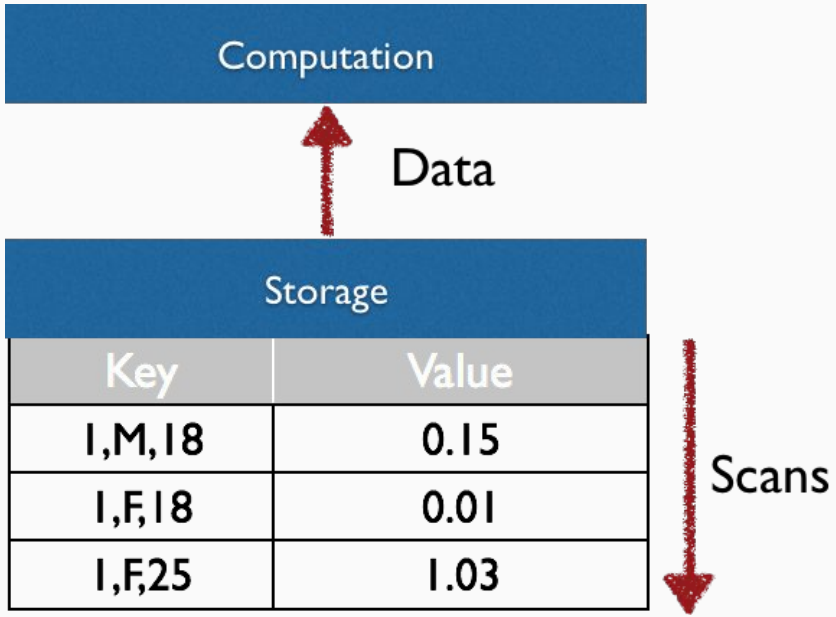
Range scans

- Primary key: dimensions/attributes
- Value: measures/metrics (things to aggregate)
- Still too slow!

ts	gender	age	revenue
1	M	18	$0.15
1	F	25	$1.03
1	F	18	$0.01

Key	Value
1,M,18	0.15
1,F,18	0.01
1,F,25	1.03

Key/Value Stores



Column stores

Load/scan exactly what you need for a query

Different compression algorithms for different columns

- Encoding for string columns
- Compression for measure columns

Different indexes for different columns

Druid

Druid

Custom column format optimized for event data and BI queries

Supports lots of concurrent reads

Streaming data ingestion

Supports extremely fast filters

Ideal for powering user-facing analytic applications

Storage Format

Raw data

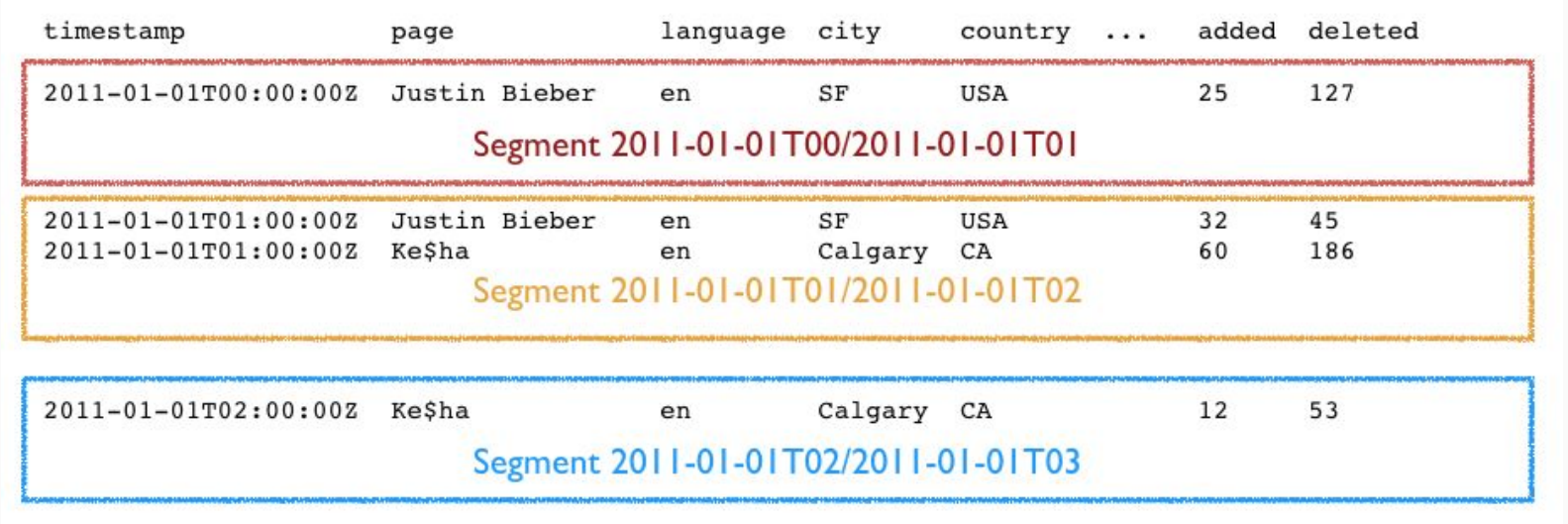
timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	en	SF	USA		10	65
2011-01-01T00:01:63Z	Justin Bieber	en	SF	USA		15	62
2011-01-01T01:02:51Z	Justin Bieber	en	SF	USA		32	45
2011-01-01T01:01:11Z	Ke$ha	en	Calgary	CA		17	87
2011-01-01T01:02:24Z	Ke$ha	en	Calgary	CA		43	99
2011-01-01T02:03:12Z	Ke$ha	en	Calgary	CA		12	53
...							

Summarization

timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:01:35Z	Justin Bieber	en	SF	USA		10	65
2011-01-01T00:01:63Z	Justin Bieber	en	SF	USA		15	62
2011-01-01T01:02:51Z	Justin Bieber	en	SF	USA		32	45
2011-01-01T01:01:11Z	Ke$ha	en	Calgary	CA		17	87
2011-01-01T01:02:24Z	Ke$ha	en	Calgary	CA		43	99
2011-01-01T02:03:12Z	Ke$ha	en	Calgary	CA		12	53
...							

timestamp	page	language	city	country	...	added	deleted
2011-01-01T00:00:00Z	Justin Bieber	en	SF	USA		25	127
2011-01-01T01:00:00Z	Justin Bieber	en	SF	USA		32	45
2011-01-01T01:00:00Z	Ke$ha	en	Calgary	CA		60	186
2011-01-01T02:00:00Z	Ke$ha	en	Calgary	CA		12	53
...							

Summarization



Data Partitioning

First level sharding done on time

- Done so for query optimization

Shards are called “segments” in Druid

Segments are immutable

Immutable Segments

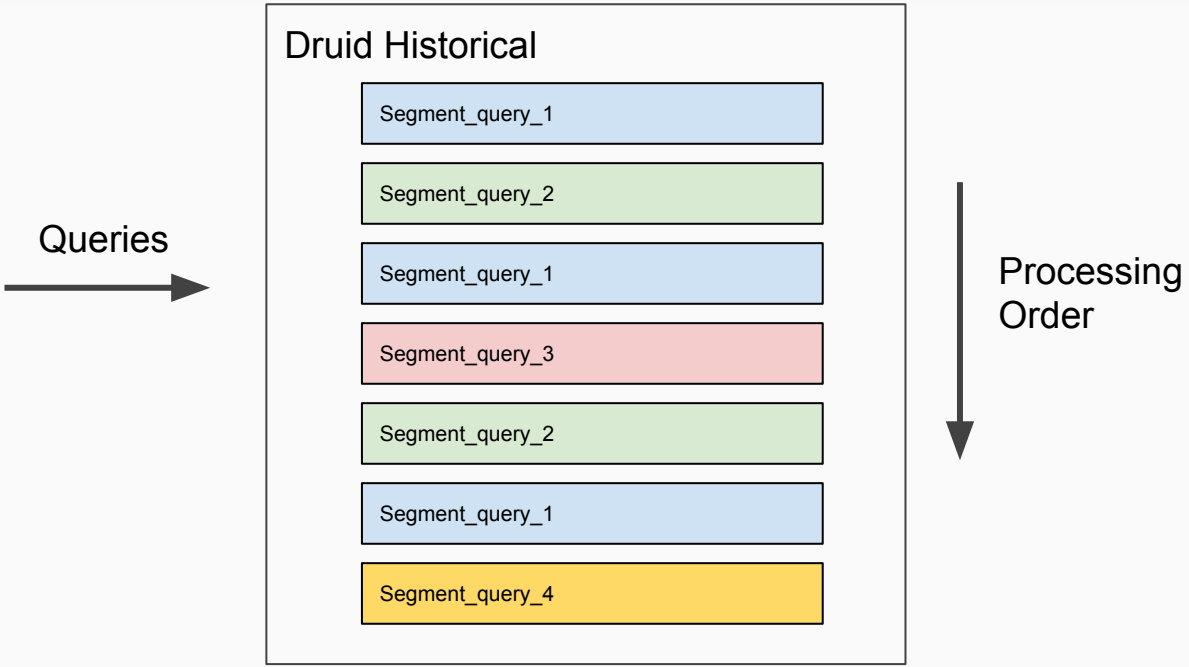
Fundamental storage unit in Druid

No contention between reads and writes

One thread scans one segment

Multiple threads can access same underlying data

Druid Multi-tenancy



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
...							

Create IDs

- Justin Bieber -> 0, Ke$ha -> 1

Store

- page → **[0 0 0 1 1 1]**
- language → **[0 0 0 0 0 0]**

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
...							

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

Justin Bieber OR Ke$ha → [111111]

Compression!

Custom Columns

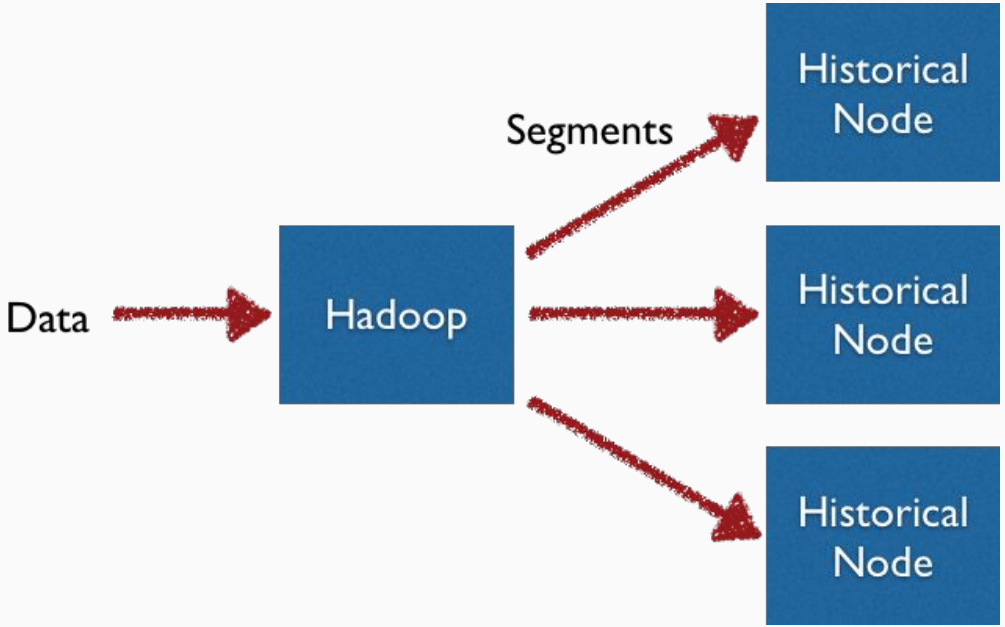
Create approximate sketches

- Hyperloglog
- Approximate Histograms
- Theta sketches

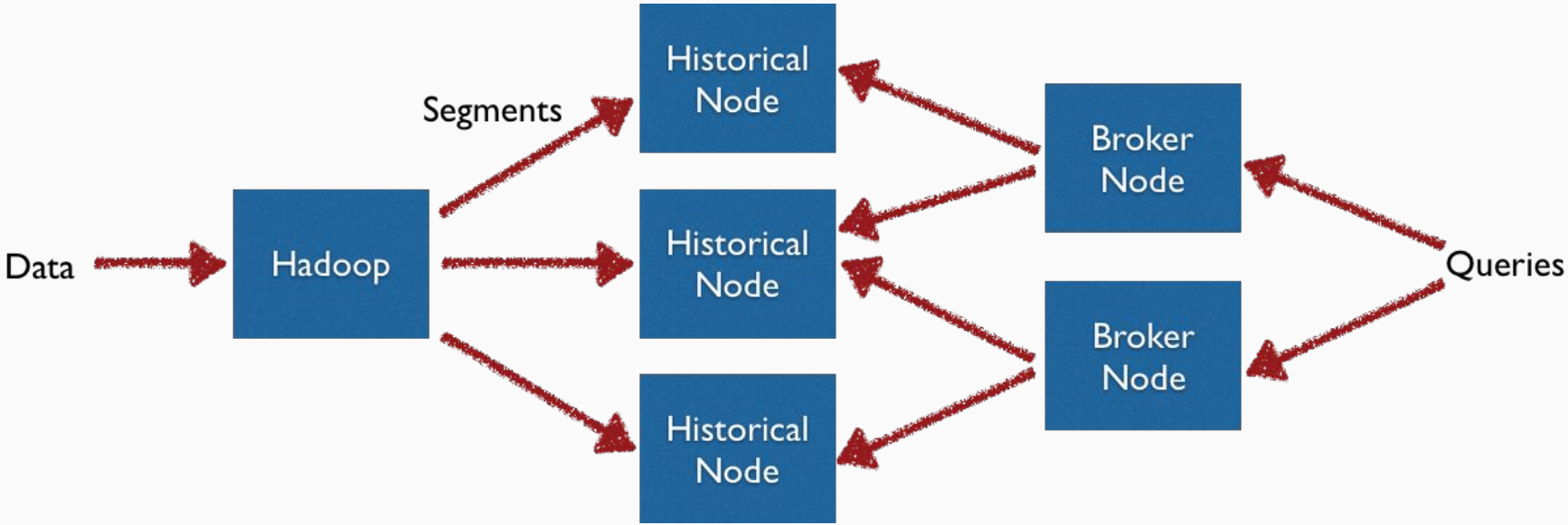
Approximate algorithms are very powerful for fast queries

Architecture

Architecture (Batch Ingestion)



Architecture (Batch Ingestion)



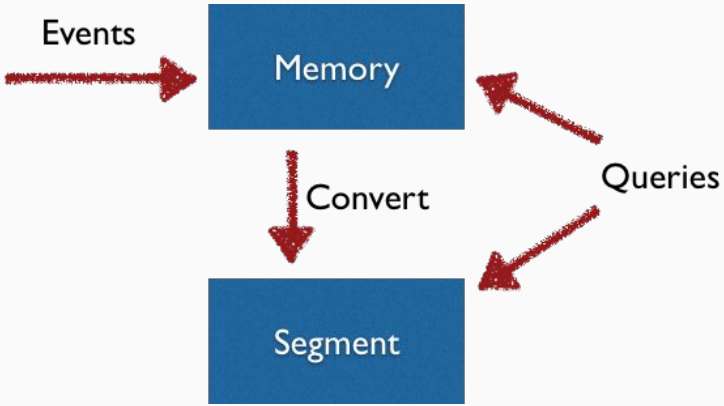
Real-time Nodes

Write-optimized data structure: hash map in heap

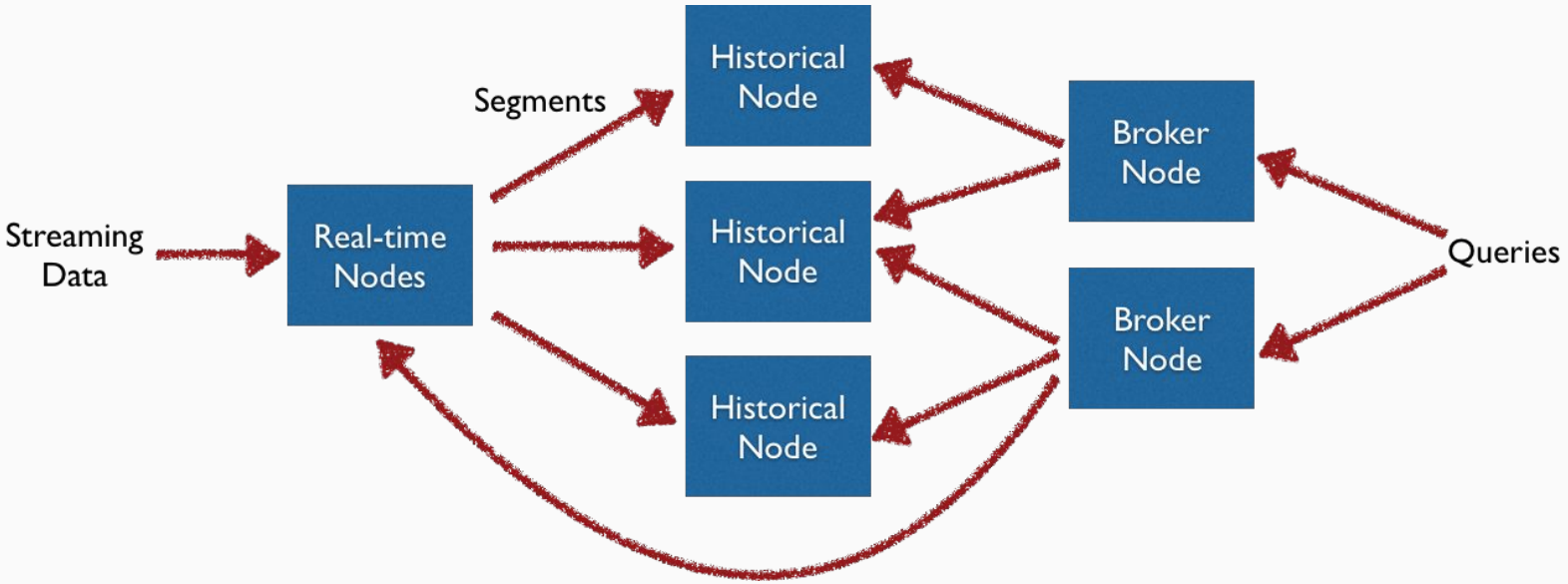
Convert write optimized -> read optimized

Read-optimized data structure: Druid segments

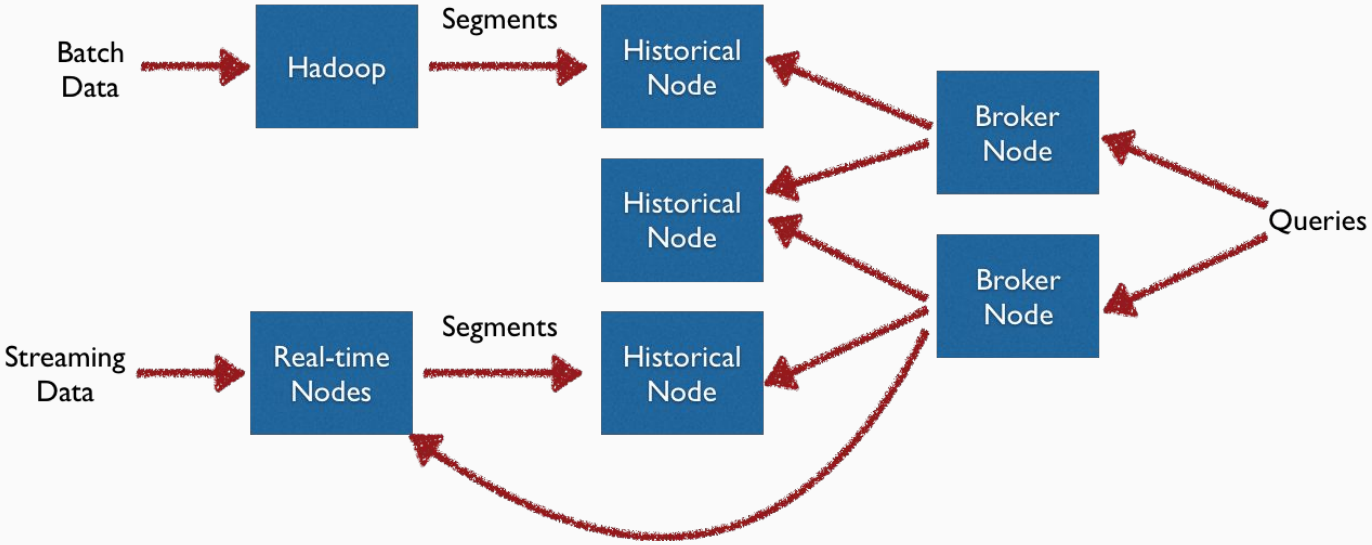
Query data immediately



Architecture (Streaming Ingestion)



Architecture (Lambda)



Querying

Query libraries:

- JSON over HTTP
- SQL
- R
- Python
- Ruby

Open source UIs

- Pivot
- Grafana
- Panoramix

Druid in Production

Community

Growing Community

- 140+ contributors from many different companies

In production at many different companies, we're hoping for more!

- Ad-tech, network traffic, cloud security, operations, activity streams, etc.

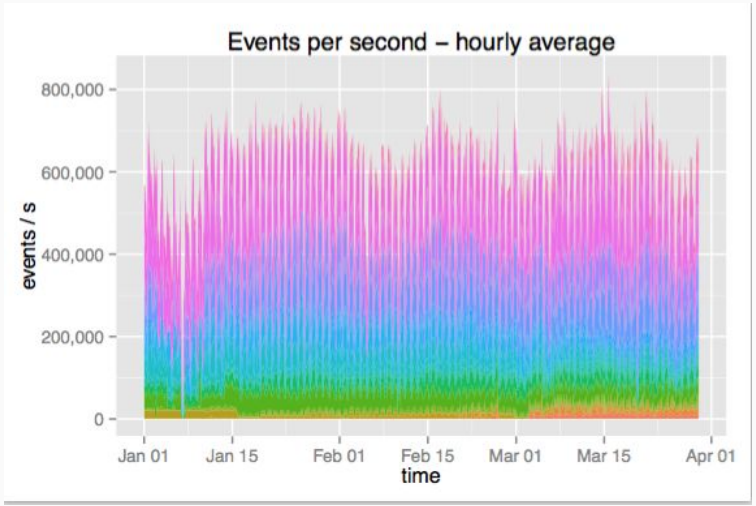
We love contributions!

Some Folks in Production

Ingestion

>3M events / second sustained (200B+ events/day)

10 – 100k events / second / core



Volume

Largest known cluster

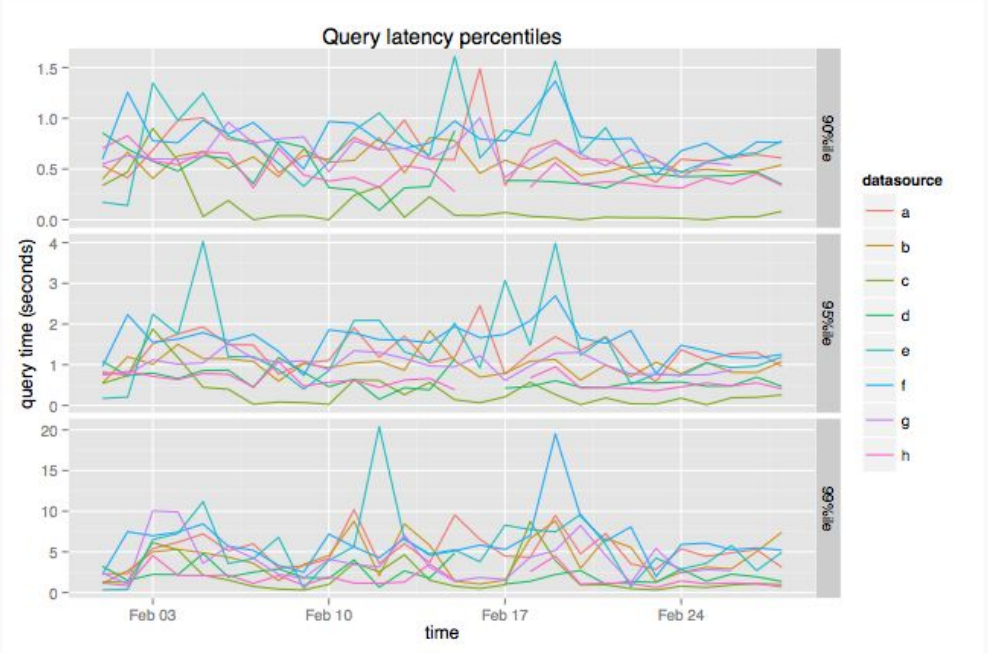
- >500 TB of segments (>50 trillion raw events, >50 PB raw data)

Extremely cost effective at scale

Queries

500ms average query latency

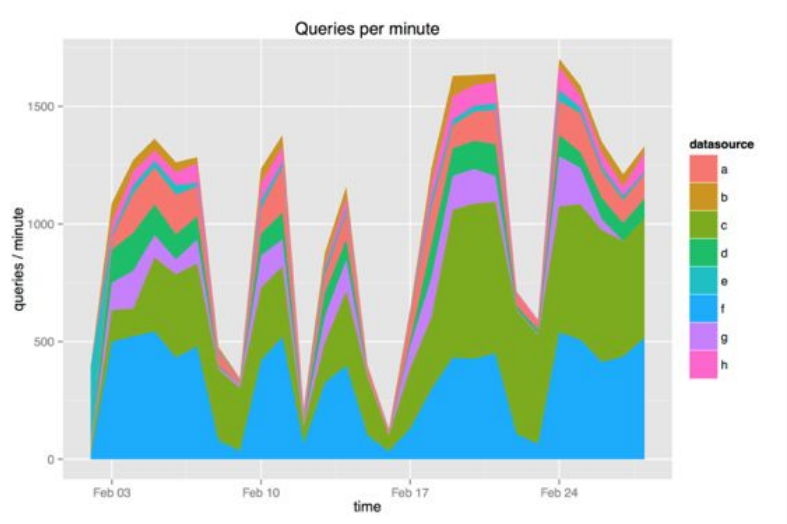
90% < 1s, 95% < 2S, 99% < 10s



Multi-tenancy

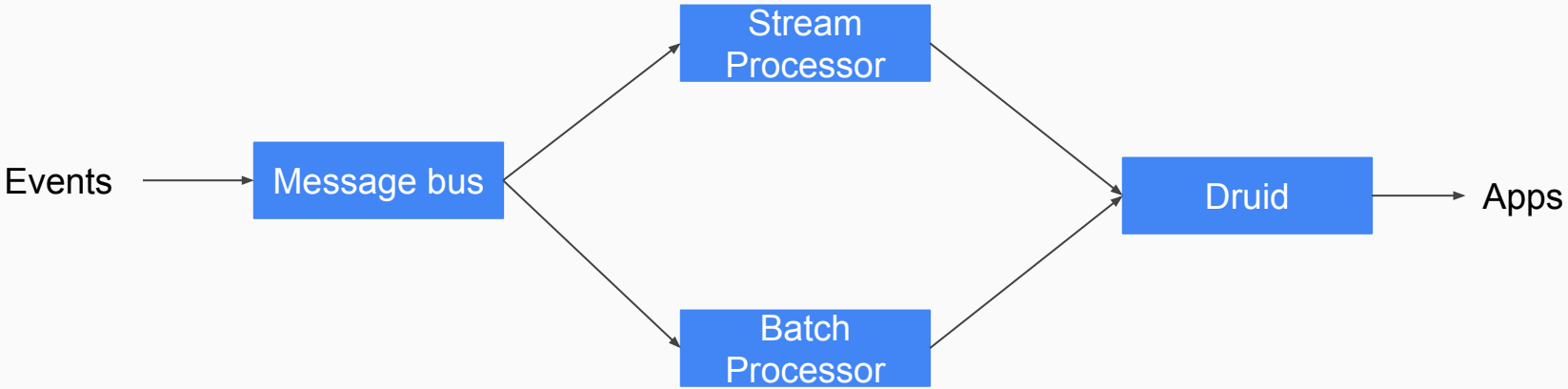
Several Hundred queries / second

Variety of group by & top-K queries



Druid & the Data Space

Architecture (Streaming Ingestion)



Integration

Druid is complementary to many solutions

- SQL-on-Hadoop (Hive, Impala, Spark SQL, Drill, Presto)
- Stream processors (Storm, Spark streaming, Flink, Samza)
- Batch processors (Spark, Hadoop, Flink)
- Messages buses (Kafka, RabbitMQ)

Takeaway

Druid is pretty good for analytic applications

Druid is pretty good at fast OLAP queries

Druid is pretty good at streaming ingestion

Druid works well with existing data infrastructure systems

Thanks!

@implydata
@druidio
@fangjin

imply.io
druid.io