Strata+Hadoop WORLD

PRESENTED BY

O'REILLY cloudera

strataconf.com

#strataconf

#hadoopworld

Stories from the Trenches

The Challenges of Building an Analytics Stack

Fangjin Yang · Xavier Léauté

Druid Committers

Software Engineers

Overview

- Demo
- Motivations
- Successes and Failures
- Lessons

DEMO

IN CASE THE INTERNET DIDN’T WORK,
PRETEND YOU SAW SOMETHING COOL

Motivations

- Interactive data warehouses
- Answer BI questions
  - How much revenue was generated last quarter broken down by a demographic
  - How many unique male visitors my website last month?
  - Not dumping an entire data set
  - Not querying for an individual event
- Cost effective (we are a startup after all)

Technical Challenges

- Ad-hoc queries
- Arbitrarily slice ’n dice, and drill into data
- Immediate insights
- Scalability
- Availability
- Low operational overhead

Where We Stand Today

- Over 10 trillion events
- ~40PB of raw data
- Over 200TB of compressed query-able data
- Ingesting over 300,000 events/second on average
- Average query time 500ms
- 90% queries under 1 second
- 99% queries under 10 seconds

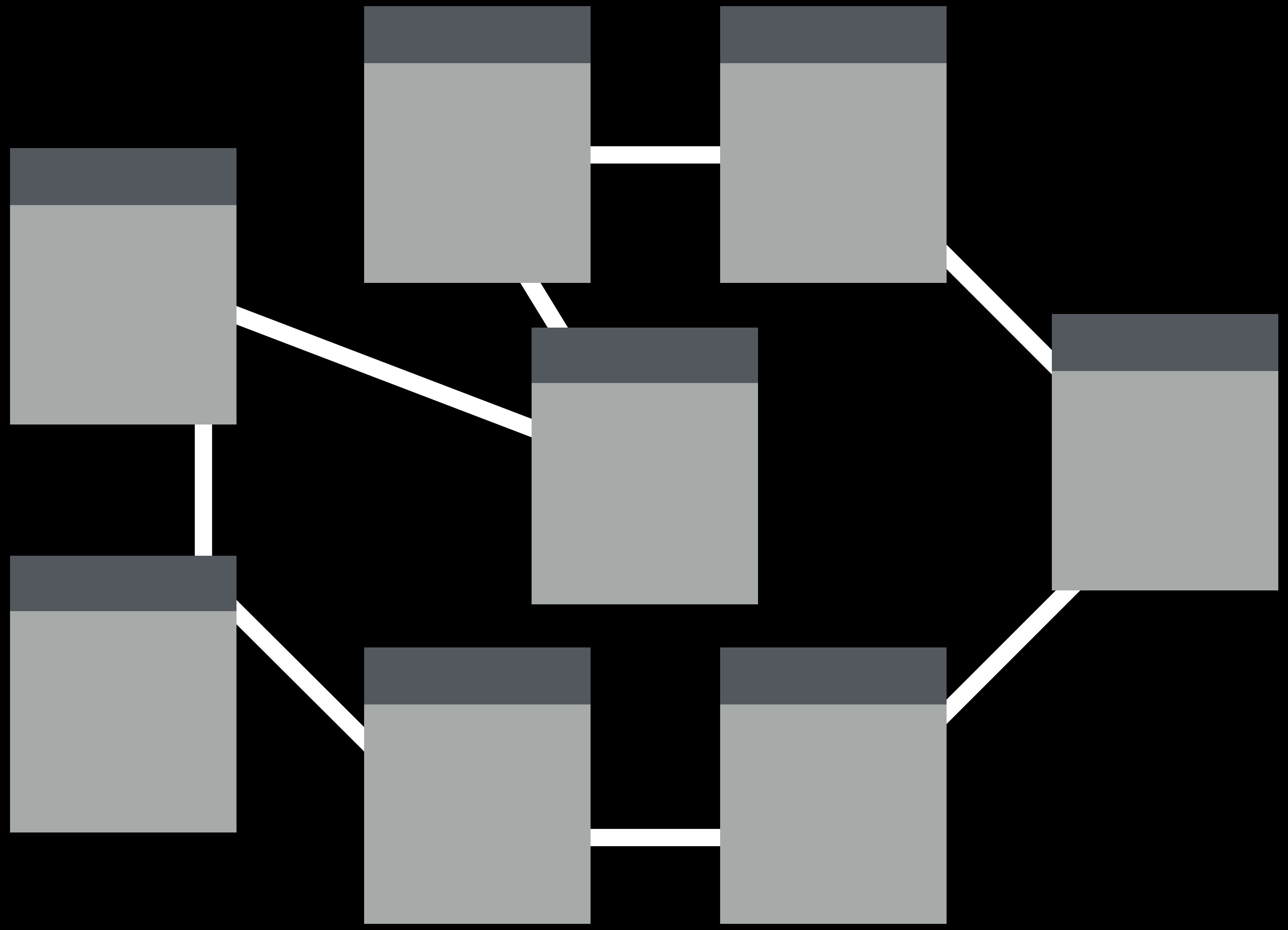
How Did We Get There?

What We Tried

- RDBMS (MySQL, Postgres)

RDBMS - The Setup

- Common setup for data warehousing
  - Star Schema
  - Aggregate Tables
  - Query Caches



RDBMS - Results

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

- ~~RDBMS (MySQL, Postgres)~~

What We Tried

- ~~RDBMS (MySQL, Postgres)~~
- NoSQL Key/Value stores (HBase, Cassandra)

NoSQL - The Setup

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

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

NoSQL - Results

- Queries were fast
  - range scan on primary key
- Inflexible
  - not aggregated, not available
- Not continuously updated
- Processing scales exponentially
  - Example: ~500k records
    - 11 dimensions : 4.5 hours on a 15-node Hadoop cluster
    - 14 dimensions: 9 hours on a 25-node Hadoop cluster

What We Tried

- ~~RDBMS (MySQL, Postgres)~~
- ~~NoSQL Key/Value stores (HBase, Cassandra)~~

What We Tried

- ~~RDBMS (MySQL, Postgres)~~
- ~~NoSQL Key/Value stores (HBase, Cassandra)~~
- ???

What We Learned

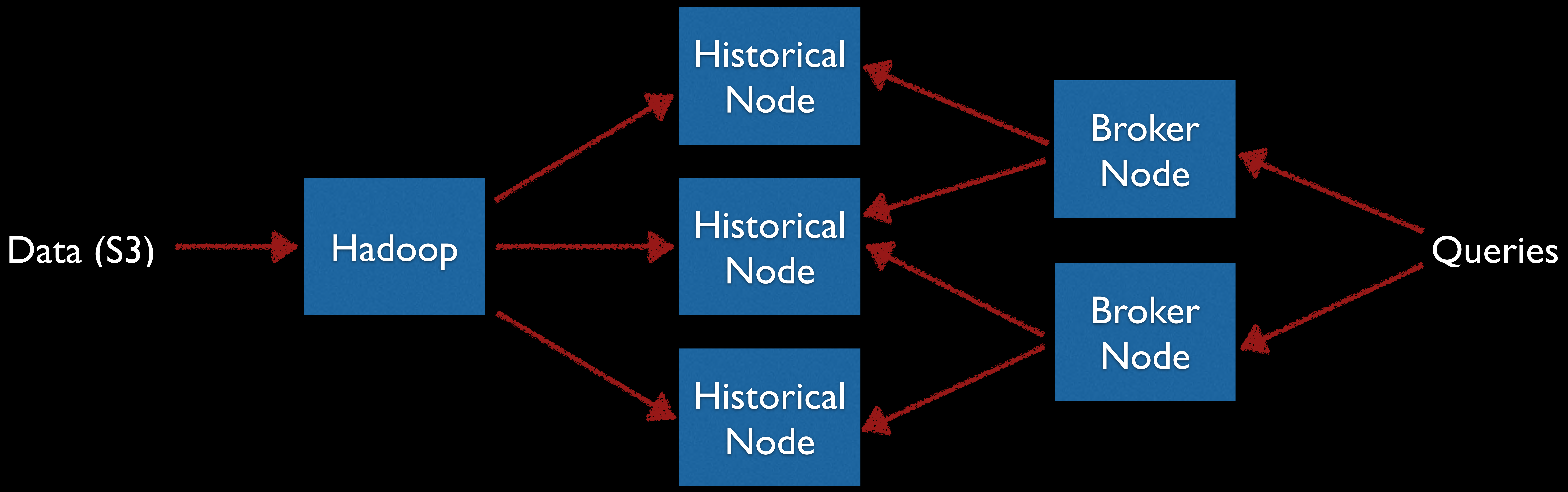
- Problem with RDBMS: scans are slow
- Problem with NoSQL: computationally intractable

- Tackling the RDBMS issue seems easier

What is Druid?

- Low Latency Ingestion
- Fast Aggregations
- Arbitrary Slice-n-dice Capabilities
- Highly Available
- Approximate & Exact calculations

Early Druid Architecture

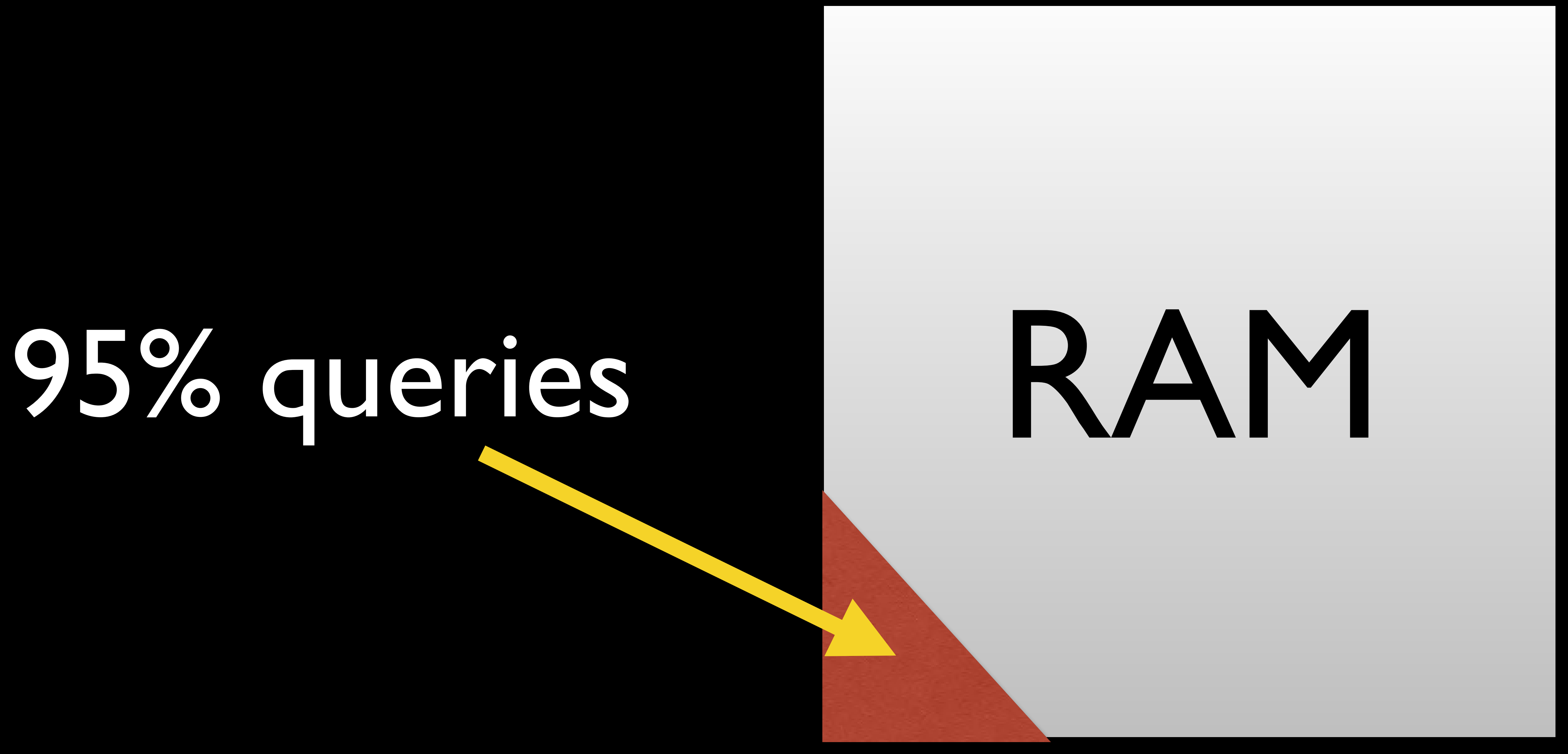


Why is Druid the Right Tool?

- Immutable data
  - Read consistency
  - Multiple threads can scan the same underlying data
  - Ideal for append-heavy, transactional data
- Column orientation
  - Load/scan only those columns needed for a query
- Search indexes (inverted indexes) to only scan what it needs

In-memory is Overrated

- All in-memory – fast and simple
- Keeping all data in memory is expensive
- Percentage of data queried at any given time is small



95% queries

RAM

Memory Map It

- Memory management is hard, let the OS handle paging
- Flexible configuration – control how much to page
- Use SSDs to mitigate the performance impact (still cheaper than RAM)
- Cost vs. Performance becomes a simple dial

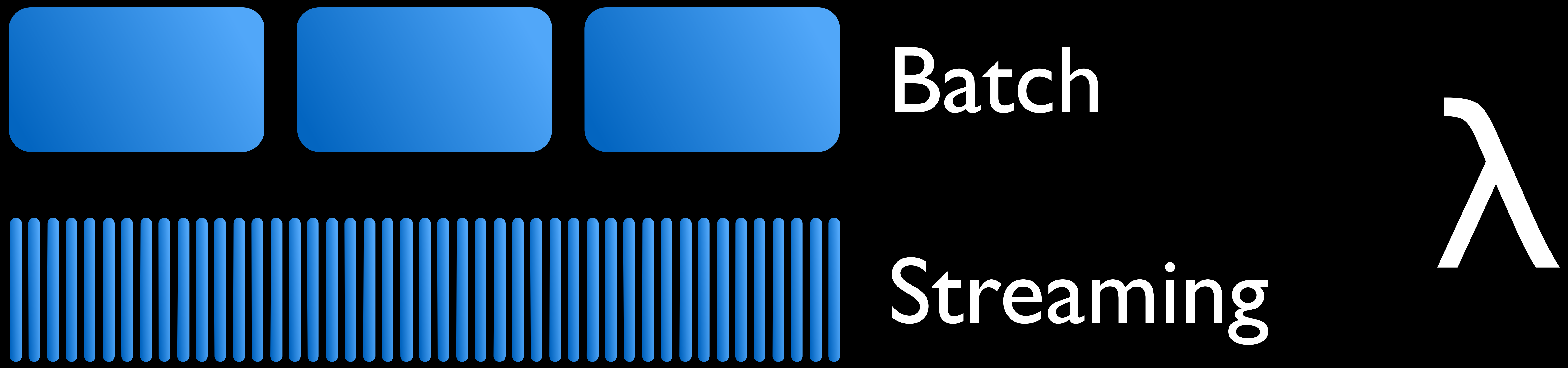
SSD

RAM

Compression is Your Friend

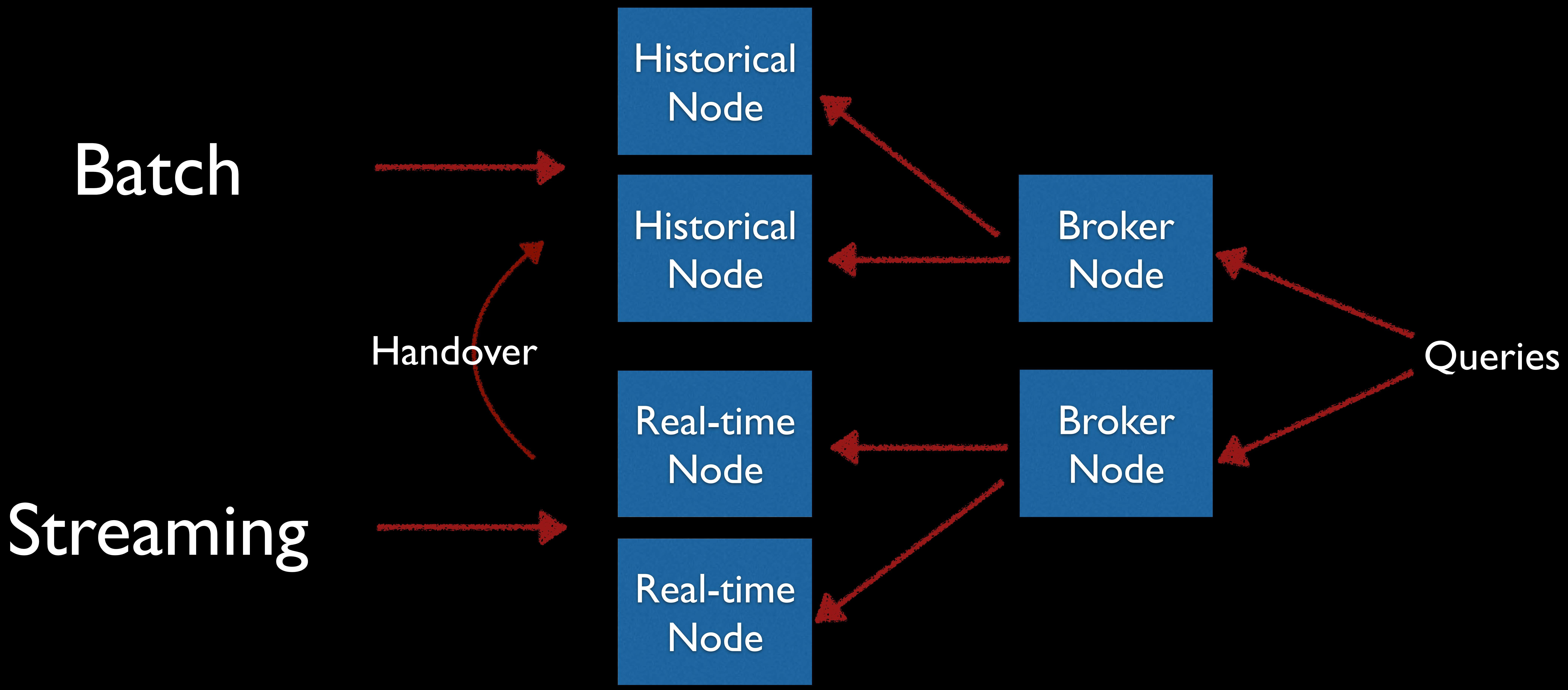
- Paging out data that isn't queried saves cost
- Memory is still critical for performance
- Cost of scaling CPU << cost of adding RAM
- On-the-fly decompression is fast with recent algorithms (LZF, Snappy, LZ4)

Low latency vs. High throughput



- Batch ingestion is accurate and efficient but slow
- Streaming ("real-time") ingestion is less accurate but fast
  - Reduces cost of frequent batch processing
- Immutable data made it easy to combine the two ingestion methods
- Now commonly referred to as lambda-architecture

Later Druid Architecture



Scaling is Hard

- Data doubles every 2 months
- More Data = More Nodes = More Failures
- Throwing money at the problem only a short term solution
- Some piece always fails to scale
- Startup means daily operations handled by dev team

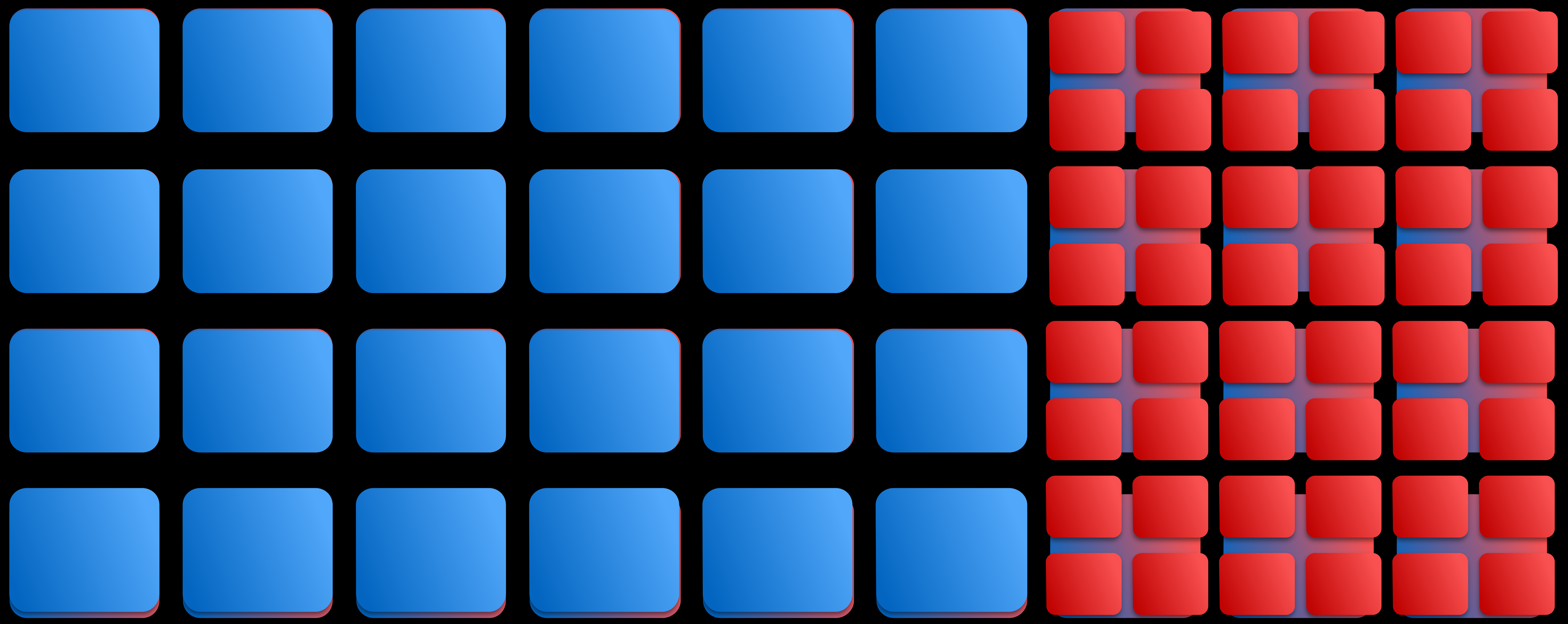
Not All Data is Created Equal

- Users really care about recent data
- Users still want to run quarterly reports
- Large queries create bottlenecks and resource contention

Smarter Rebalancing

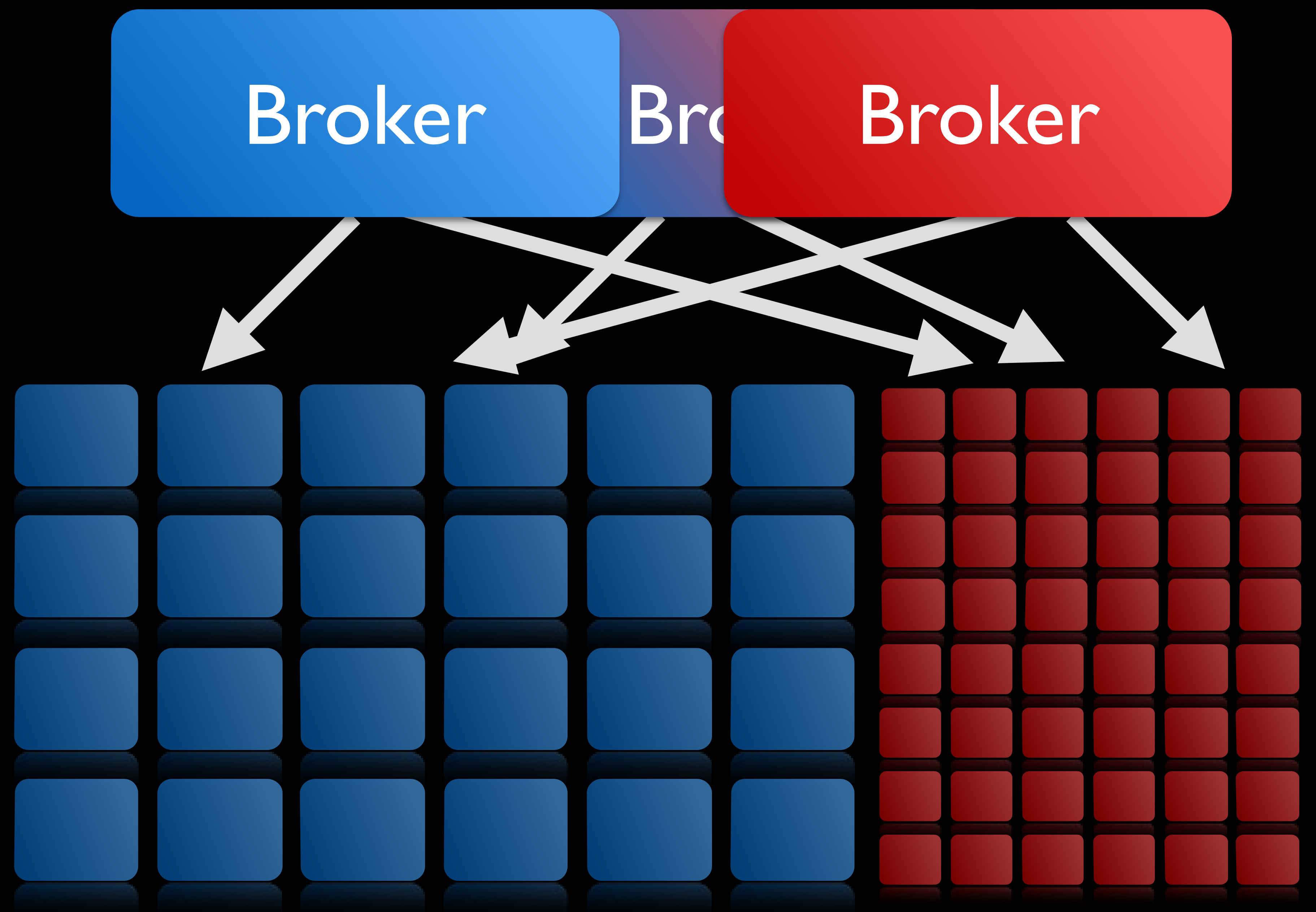
- Constantly rebalance to keep workload uniform
- Greedily rebalance based on cost heuristics
  - Avoid co-locating recent or overlapping data
  - Favor co-locating data for different customers
  - Distribute data likely to be queried simultaneously

Create Data Tiers



- COLD high disk to cpu, and disk to ram ratio for old data
- HOT low disk to cpu and low disk to ram for new data

Create Query Tiers



- Separate query nodes for long and short running queries
- Prioritize shorter queries

Scaling Upgrades

🚫 DOWNTIME

- Make every piece of the system redundant
- Make components stateless
- Fail-over stateful components

Scaling Upgrades

🚫 DOWNTIME

- Shared nothing architecture
- Maintain backwards compatibility
- Allow upgrading components independently

It's OK to be Slow (sometimes)

- Replication can become expensive
- Not willing to sacrifice availability
- Tradeoff performance for cost during failures
  - Move replica to cold tier
  - Keep a single replica for hot

Simplify Operations

- Data migrations are painful
- Separate resources for
  - permanent data storage
  - data processing

- Machines become dispensable
- Easy to try out / upgrade to new hardware
- Smarter loading / unloading / archiving of data
- Reduced operational complexity

© Airbus S.A.S 2006, HCSGM - Computer Graphic Passion Graphic

Multitenancy is Harder

- Everyone wants a good experience
- Behavior is not uniform across customers
- 20% of customers take 80% of resources

Addressing Multitenancy

- Bound Resources
  - Keep units of computation small
  - Constantly yield resources
- Prefer fast approximate answers to slow exact ones
  - HyperLogLog sketches
  - Approximate top-k
  - Approximate histograms

Monitoring

- Collecting lots of data without having the tools to analyze it is useless
- Use Druid to monitor Druid!
- > 10TB of metrics data in Druid
- Often hard to tell where problems are coming from
- Interactive exploration of metrics allows us to pinpoints problems quickly
- Granularity down to the individual query or server level
- Gives both the big picture and the detailed breakdown
- Demo!

Flickr: morrosv7

Take Aways

- Pick the right tool
  - Pick the tool optimized for the types of queries you will make
- Tradeoffs are everywhere
  - Performance vs. cost (in-memory, tiering, compression)
  - Latency vs. throughput (streaming vs. batch ingestion)
  - Use cases should define engineering (understand query patterns)
- Monitor everything

More About Druid

- Open sourced 2 years ago
- 10+ Production Deployments
  - Ad-tech
  - Network traffic analysis
  - Operations Monitoring
  - Activity stream analysis

Thank You

@druidio druid.io #druid-dev

metamarkets.com