Hadoop Application Architectures: Architecting a Next Generation Data Platform

Strata Data Conference, London 2018
Ted Malaska | @tedmalaska
Jonathan Seidman | @jseidman
Logistics

- Break at 3:00 – 3:30 PM
- Questions at the end of each section
- Slides at tiny.cloudera.com/app-arch-london-2018
- Code at https://github.com/hadooparchitecturebook/Taxi360
About the book

- @hadooparchbook
- hadooparchitecturebook.com
- github.com/hadooparchitecturebook
- slideshare.com/hadooparchbook
About the presenters

Ted Malaska

- Director of Engineering of Global Insights at Blizzard
- Cloudera Principal Solution Architect
- Architect at FINRA
- Contributor to:
  - Apache Spark,
  - Hadoop,
  - Hive,
  - Sqoop,
  - Yarn,
  - Flume,
  - Etc.
About the presenters

Jonathan Seidman

- Software Engineer at Cloudera
- Previously Technical Lead on the big data team at Orbitz
- Co-founder of the Chicago Hadoop User Group and Chicago Big Data
Case Study Overview
Internet of Things and Entity 360
Customer 360

Questions?
tiny.cloudera.com/ukquestions
Connected Cars

Questions?
tiny.cloudera.com/ukquestions
Entity (Taxi) 360 View

Streaming Vehicle Data
Geo-location/Traffic Data
Customer Data
Maintenance Data
Other Data Sources
In the old days

Data Warehouse → Extract → Transform (SQL) → Load → Data Mart
At the time of our Book

Questions?
tiny.cloudera.com/ukquestions
Today...

Data Sources

Producers

Streaming Pipes

Storage

Access

- Code
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

- File/Object
- RDBMS/MPP
- Time Series
- Reverse Indexed
- Memory
- Stream

- SQL
- Machine Learning
- Request Response
- Batch Processing

Questions?
tiny.cloudera.com/ukquestions
Enabling a Range of New Use Cases…

Fraud Detection
Market Transactions
Internet of Things
Network Security
Challenges

And so many more

Questions?
tiny.cloudera.com/ukquestions
Challenges – Architectural Considerations

- Reliable and scalable ingress of multiple data types and sources:
  - High volume event data? Batch data?

- Reliable and scalable storage to support multiple workloads and access patterns
  - Historical data? Real-time search? Analytics?

- Processing engines (for background processing):
  - Stream processing? Batch processing?

- Data Modeling
  - Modeling data for real-time random access? Analytic access? Batch access?

- Deployment Considerations
  - On-premise, cloud?
Case Study
Requirements
Overview
Requirements

- Allow users (technical and non-technical) to analyze and visualize data…
Requirements

- Provide analysts with query capabilities via a standard interface…
Requirements

- Provide developers the ability to perform batch processing on historical data…
Requirements

- To support all this, we need:
  - Reliable ingestion of streaming and batch data.
  - Ability to perform transformations on streaming data in flight.
  - Ability to perform sophisticated processing of historical data.
  - Reliable and scalable storage to support modeling and processing of multiple data formats.
High level architecture
Walkthrough
High level architecture

- **Data Sources**
  - Code
  - Agents
  - Log Aggregators

- **Producers**
  - Schema Validation
  - Enrichment
  - Stream Processing
  - Routing
  - Transport
  - Replication

- **Streaming Pipes**
  - Code
  - Agents
  - Log Aggregators

- **Storage**
  - File/Object
  - RDBMS/MPP
  - Time Series
  - Reverse Indexed
  - Memory
  - Stream

- **Access**
  - SQL
  - Machine Learning
  - Request Response
  - Batch Processing

Questions?
tiny.cloudera.com/ukquestions
Data Sources
And Data Acquisition
Considerations
High level architecture
Key to Customer 360 Success

Your project is only as good as the quality and variety of data sources

- Geo-location/Traffic Data
- Files: CSV, XML, JSON
- Database
- Maintenance Data
- Customer Data
- Salesforce?
- Other Data Sources
- Twitter? Mainframe?
- Streaming Vehicle Data
- MQTT

Questions?
tiny.cloudera.com/ukquestions
High Level Architecture

Data Sources
- Code
- Agents
- Log Aggregators

Producers
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Streaming Pipes

Storage
- File/Object
- RDBMS/MPP
- Time Series
- Reverse Indexed
- Memory
- Stream

Access
- SQL
- Machine Learning
- Request Response
- Batch Processing

Questions?
tiny.cloudera.com/ukquestions
Producers – Push and Pull

- Code
  - Agents
  - Log Aggregation
Producers – Push and Pull

- Code
  - Directly to pipeline
  - To aggregation layer
  - REST
  - Kafka Producer
Producers – Push and Pull

- Agents
  - Snap
  - Prometheus
  - Telegraf
Producers – Push and Pull

- Log Aggregations
  - Syslog
  - Logger Implementation
Kafka Clients

Apache Kafka Clients

Ecosystem Clients

Java
Scala
Clojure
Ruby
Perl
php
C
Python
Go
R
.NET
Node.js
Erlang

Questions?
tiny.cloudera.com/ukquestions
REST Proxy
Talking to Non-native Kafka Apps and Outside the Firewall

- Provides a RESTful interface to a Kafka cluster
- Simplifies message creation and consumption
- Simplifies administrative actions

Non-Java Applications

REST Proxy

Native Kafka Java Applications

Questions?
tiny.cloudera.com/ukquestions
Kafka Connect
Streaming Data Capture

- Fault tolerant
- Manage hundreds of data sources and sinks
- Preserves data schema
- Part of Apache Kafka project
- Includes simple transformations

Kafka Connect API

- Sources: JDBC, Logs, MQTT
- Connectors: Connector, Connector, Connector, Connector
- Sinks: RDBMS, Key/Value, HDFS, Sinks

Kafka
Ecosystem of Connectors

<table>
<thead>
<tr>
<th>Databases</th>
<th>Datastore/File Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC</td>
<td>hadoop</td>
</tr>
<tr>
<td>mongoDB</td>
<td>Ignite</td>
</tr>
<tr>
<td>Cassandra</td>
<td>FTP</td>
</tr>
<tr>
<td>ORACLE Golden Gate</td>
<td>apache</td>
</tr>
<tr>
<td>Just One Data</td>
<td>syslog-ng</td>
</tr>
<tr>
<td>Couchbase</td>
<td>hazelcast</td>
</tr>
<tr>
<td>DynamoDB</td>
<td>Kudu</td>
</tr>
<tr>
<td>InfluxDB</td>
<td></td>
</tr>
<tr>
<td>HBase</td>
<td></td>
</tr>
<tr>
<td>RethinkDB</td>
<td></td>
</tr>
<tr>
<td>Kudu</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analytics</th>
<th>Applications / Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERTICA</td>
<td>Bloomberg</td>
</tr>
<tr>
<td>elastic</td>
<td>twitter</td>
</tr>
<tr>
<td>mixpanel</td>
<td>syncsort</td>
</tr>
<tr>
<td>Solr</td>
<td>ATTUNITY</td>
</tr>
<tr>
<td>Apache Kafka</td>
<td></td>
</tr>
<tr>
<td>Kudu</td>
<td></td>
</tr>
</tbody>
</table>
High level architecture

Data Sources
- Code
- Agents
- Log Aggregators

Producers
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Streaming Pipes
- Coding
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Storage
- File/Object
- RDBMS/MPP
- Time Series
- Reverse Indexed
- Memory
- Stream

Access
- SQL
- Machine Learning
- Request Response
- Batch Processing

Questions?
tiny.cloudera.com/ukquestions
Why Schema Validation?

▪ Schema declaration
  - Regulation (GDPR)
  - Consistency
▪ Common formats
▪ Additional Information
  - Routing
  - Filtering
  - Transformations
How to Implement Schema Validation

- Kafka Schema Registry
- Custom implementation
- Other solutions
High level architecture

Data Sources
- Code
- Agents
- Log Aggregators

Producers
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Streaming Pipes
- Long Term SQL
- Speed Layer SQL
- Time Series
- Reverse Indexed
- Stream

Storage
- SQL
- Machine Learning
- Request Response
- Batch Processing

Access

Questions?
tiny.cloudera.com/ukquestions
Goals for our Transport Layer

- Buffering
- Partitioning
- Speed
- Replay ability
- High throughput
- Reliable
Goals for our Transport Layer

- To meet these goals we want some kind of publish-subscribe queue:
  - Kafka
  - Kinesis
  - RabbitMQ
  - Azure Queues
  - Azure Service Bus
  - Google Pub/Sub
  - etc…
Goals for our Transport Layer

▪ In general you have two decisions:
  - Cloud service or your own managed component
  - Guarantee Levels
Buffering Data

- What do we mean by “buffering” and why do we need it?

- Network partitions happen
- Producers and Consumers work at different rates
- Reliable storage is hard
  Stream processing is hard
- Lets do one at a time

This is bad!
Buffering Data – Message Brokers

Diagram showing publishers and subscribers connected to a message queue.

Questions?
tiny.cloudera.com/ukquestions
What is Kafka?

- It’s like a message queue, right?
  - Actually, it’s a “distributed commit log”
  - Or “streaming data platform”
Topics and Partitions

- Messages are organized into topics, and each topic is split into partitions.
  - Each partition is an immutable, time-sequenced log of messages on disk.
  - Note that time ordering is guaranteed within, but not across, partitions.
In Our Architecture

Producer

Kafka

Topic

taxi-trip-input

Consumers

Stream Processing (Analytic)

Stream Processing (Lookup)

Stream Processing (Search)

Stream Processing (Long Term)

In Our Architecture

Producer

Kafka

Topic

taxi-trip-input

Consumers

Stream Processing (Analytic)

Stream Processing (Lookup)

Stream Processing (Search)

Stream Processing (Long Term)
Input Events

cmt,2009-01-05 08:31:55,2009-01-05 8:37:50,1,0.90000000000000002,-73.977936999999997,40.745919000000001,,,-
73.983609000000001,40.755051000000002,Credit,5.2999999999999998,0,,0.79000000000000004,0,6.0
8999999999999999
Kafka Considerations – Reliability

- But remember there are tradeoffs…
Kafka Considerations – Reliability

- Different reliability levels for topics:

  - Taxi Trip Data
  - Taxi-trip-input
  - Customer sentiment

100% – dups are ok (“At least once”)

<=100% (“At most once”)

News Flash: Kafka’s Exactly Once Producer is on the way
Kafka Reliability – Replication
Kafka Reliability – Replication

Producer

- Partition 1
- Partition 2
- Partition 3
Kafka Reliability – Replication

Questions?
tiny.cloudera.com/ukquestions
Kafka Reliability – Replication

Questions?
tiny.cloudera.com/ukquestions
Kafka Reliability – Replication

Questions?
tiny.cloudera.com/ukquestions
View from the Consumer

Kafka Broker Cluster

Partition A

Partition B

Partition C

Consumer Group A

Consumer

Consumer

Consumer

Consumer Group B

Consumer

Consumer

Consumer

Questions?
tiny.cloudera.com/ukquestions
View from the Consumer
Kafka Reliability – Replication

- So how does this relate to our application?

```
kafka-topics --zookeeper ZKHOST:ZKPORT -partition 2 --replication-factor 3 \   --create --topic taxi-trip-input
```

```
kafka-topics --zookeeper ZKHOST:ZKPORT -partition 2 --replication-factor 1 \   --create --topic customer-sentiment
```
Kafka Reliability – Producers

Producer

Taxi Trip Data

acks=all

Message failure?

Resend message

Kafka

taxi_trip_input

Partition 1

Partition 2

Partition 3

Topic B

Partition 1

Partition 2

Partition 3

Questions?
tiny.cloudera.com/ukquestions
Kafka Reliability – Producers

▪ What about duplicates?

<table>
<thead>
<tr>
<th>ID</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2009-01-04 03:02:00,1,2.629,...</td>
</tr>
<tr>
<td>1001</td>
<td>2009-01-04 03:03:00,3,4.549,...</td>
</tr>
<tr>
<td>1001</td>
<td>2009-01-04 03:38:00,3,4.549,...</td>
</tr>
</tbody>
</table>

Producer
Taxi Trip Data

Kafka

- taxi_trip_input
  - Partition 1
  - Partition 2
  - Partition 3

- Topic B
  - Partition 1
  - Partition 2
  - Partition 3
Kafka Scaling – Partitions

Producer → Kafka → taxi-trip-input → Partition 1, Partition 2, Partition 3 → Consumer Group

Consumer
Consumer
Consumer
Kafka Scaling – Partitions

Producer -> Higher throughput -> Kafka -> taxi-trip-input -> Partition 1, Partition 2, Partition 3, Partition 4, Partition 5 -> Higher throughput -> Consumer Group

- Higher throughput
- More resources (memory)
- More resources (file handles)

Questions?
tiny.cloudera.com/ukquestions
How many partitions?

- Adding partitions late in the game is painful

- Basic formula:
  total desired throughput / throughput of slowest consumer or producer

- Or ~25GB disk space

- Not too many because:
  - Each partition takes broker heap memory and file handles
  - Each partition slows down node shutdown / recovery
  - 1000 – 4000 partitions per broker max
  - Producers will produce smaller batches – lower throughput
Kafka Scaling – Producers

Producer

Kafka

taxi-trip-input

Partition 1
Partition 2
Partition 3
Partition 4
Partition 5

Consumer Group

Consumer
Consumer
Consumer
Consumer
Consumer
Guarding Against Message Loss

- **Producer** – What happens if the producer loses connection to Kafka and the buffer overflows?
  - You get an exception. You can choose to… block? Write to file?

- **Source** – What happens if events are lost before getting sent to producer?
  - Once again use some kind of buffer to provide sufficient retention of data.
Stream Processing

Considerations
High level architecture

Data Sources
- Code
- Agents
- Log Aggregators

Producers
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Streaming Pipes
- Stream Processing

Storage
- File/Object
- RDBMS/MPP
- Time Series
- Reverse Indexed
- Memory
- Stream

Access
- SQL
- Machine Learning
- Request Response
- Batch Processing

Questions?
tiny.cloudera.com/ukquestions
Streaming agenda

- What do we mean by streaming?
- Streaming use-cases
- Streaming semantics
- Which streaming engine to choose?
- Streaming in our use-case
What do we mean by streaming?
What do we mean by streaming?

Constant low milliseconds & under

Low milliseconds to seconds, delay in case of failures

10s of seconds or more, re-run in case of failures

Real-time

Near real-time

Batch

Questions?
tiny.cloudera.com/ukquestions
What do we mean by streaming?

- **Constant low milliseconds & under**
- **Low milliseconds to seconds, delay in case of failures**
- **10s of seconds or more, re-run in case of failures**

- Real-time
- Near real-time
- Batch

Questions?
tiny.cloudera.com/ukquestions
But, there’s no free lunch

Constant low milliseconds & under
Low milliseconds to seconds, delay in case of failures
10s of seconds or more, re-run in case of failures

Real-time
Near real-time
Batch

“Difficult” architectures, lower latency
“Easier” architectures, higher latency

Questions?
tiny.cloudera.com/ukquestions
Streaming use-cases
What are Streaming Use Cases

- Atomic Enrichment
- Notifications
- Joining
- Partitioning
- Windowing
- Ingestion
- Aggregation & Counting
- Incremental Machine Learning
- Progressive Analytics
Atomic Enrichment

- Isolated Transforms
- Transform with context
- FlatMap & Filtering & Routing
Atomic Enrichment

- Isolated Transforms
Atomic Enrichment

- Transform with context
Atomic Enrichment

- Flat Map & Filtering
Notifications

- Making a decision on an event
- Send out a notification to an action taker
Notifications

Normal Queue → Processor → Notification Queue → Action Taker

Processor:
- Incoming Events
- Transform Logic
- Cached Context
  - Rules & Logic
- Cached Store

Notification Queue:
- Recording Decisions Queue
- Research & Machine Learning

Storage
Joining

- Joining big to small table
- Broadcast join
- Joining two large tables
- Getting latest on same key from different feeds
- Windowing
Windowing

- A table view of a stream
Windowing Lead and Lag
Windowing Lead and Lag
Windowing Lead and Lag

N<B and N<A => Valley
Windowing Lead and Lag

N>B and N>A => Peak
Tumbling Window
Tumbling Window
Sliding Window
Sliding Window
Sliding Window

Questions?
tiny.cloudera.com/ukquestions
What to do with Sessions?

- Window of events defined by a gap

- What to do with them
  - Length of sessions
  - Number of sessions with in a day
  - Average separation between events in a session
  - Common orders with in sessions
  - Counts of types of events with in sessions
What to do with Sessions?

- Start
- Length
- Min Session Gap Length
- Start
- New Event
- Start
- Length
What to do with Sessions?

- Start
- Length
- Min Session Gap Length
  - Start
  - New Event
- Start
- Length
- Close session
  - New Start

Questions?
tiny.cloudera.com/ukquestions
Respect to Infection Points

- Think about
  - Churn
  - Heart attacks
  - Death

- Infection point type
- Length before infection point
Inflection Point Windowing
Windowing

- **Processing Time**
  - Time the event reaches your system

- **Event Time**
  - Time the event was created
Ingestion

- Taking values from a stream into a data store
  - File System
  - Object Store
  - NoSQL
  - NoSQL Time Series
  - Document Store
  - Lucene based System
  - Spanner System
Ingestion

- **File Systems & Object Store**
  - Normally you want larger files
  - Normally you want high compression
  - Normally you want deduping
    - Window of deduping
    - 100% deduping in this case is difficult
      - Think about sequence numbering from source
Ingestion

- NoSQL, Lucene, and Spanner Systems
  - If you use the key deduping is given for free
  - Different levels of latency and through put
    - NoSQL
    - Document
    - Spanner
    - Lucene
  - Partitioning on the way in
Ingestion

- **NoSQL Time Series**
  - Dumb inserts
  - You may need Aggregation across metrics
  - To increase throughput you may want to buffer writes
  - **Context**
    - Row is a key time and data points There
      - Row1 Time1, DataPoint1
      - Row1 Time2, DataPoint2
    - Is slower then
      - Row1 Time1, DataPoint1, Time2, DataPoint2
Aggregation & Counting

- Counting value with respect to an entity ID
- Counting totals and with respect to window times
Aggregation & Counting

- This is where we talk about Lambda
- There are many definitions
  - Common but not correct: Jobs that involve both Batch & Streaming
  - Correct: Perfect count is not possible with streaming so we use a combination of streaming and batch to show the right value
Aggregation & Counting

- Why is streaming not perfect
Aggregation & Counting

- Why is streaming not perfect (more)
- Duping in general
Aggregation & Counting

- What is Lambda

So over time lots of errors
Aggregation & Counting

- What is Lambda

Less room for error. So in theory more correct.
Aggregation & Counting

- How is lambda Resolved with out batch?
- Failure problem
- Deduping problem
Aggregation & Counting

- Failure Problem Solved by Adding internal State
Aggregation & Counting

- Deduping Problem Solved by
  - Some deduping can happen with in a batch
  - But this is not good enough
- Source Sequence Numbers
  - With Order, Partitioning, and Internal State
Aggregation & Counting

- Deduping with Sequence Numbers

<table>
<thead>
<tr>
<th>Source</th>
<th>Sequence</th>
<th>Value</th>
<th>Seq of A</th>
<th>Value of A</th>
<th>Seq of B</th>
<th>Value of B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>100</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>20</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>100</td>
<td>2</td>
<td>20</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>30</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>100</td>
<td>2</td>
<td>30</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>100</td>
<td>3</td>
<td>30</td>
<td>3</td>
<td>300</td>
</tr>
</tbody>
</table>
Aggregation & Counting

- Closing Thoughts on Lambda
Progressive Analytics

- Running SQL on a stream
- The idea of streams and tables being one
Progressive Analytics

- Table and Streaming
- Tables are a set of columns and rows
  - They represent the current state of things
- Stream is a set of change longs
  - The combined total of the changed logs equals a table
Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum

**WAL**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

**Current State**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
</tbody>
</table>
Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>

Trigger Result:
A:3  
B:1  
C:1  

Current State

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
</tbody>
</table>
Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 messages
- Action => Group by Sum

<table>
<thead>
<tr>
<th>WAL</th>
<th>Current State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
</tbody>
</table>
Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 messages
- Action => Group by Sum

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

Evicted

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
</tbody>
</table>
Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum
Streaming semantics
Delivery Types

▪ **At most once**
  - Not good for many cases
  - Only where performance/SLA is more important than accuracy

▪ **Exactly once**
  - Expensive to achieve but desirable

▪ **At least once**
  - Easiest to achieve
Semantics of our architecture

Source System 1
Source System 2
Source System 3

Message broker

Streaming engine

At least once
Ordered
Partitioned

It depends

Ingest
Ingest
Ingest

Extract

Push

Destination system

Questions?
tiny.cloudera.com/ukquestions
Classification of storage systems

- File based
  - S3
  - HDFS

- NoSQL
  - HBase
  - Cassandra
  - MongoDB
  - Duid.io

- Document based
  - ES & Solr

- NoSQL-SQL
  - Kudu, CockRoachdb
Classification of storage systems

- **File based**
  - S3
  - HDFS

- **NoSQL**
  - HBase
  - Cassandra
  - MongoDB
  - Duid.io

- **Document based**
  - ES & Solr

- **NoSQL-SQL**
  - Kudu, CockRoachdb

De-duplication at file level

Semantics at key/record level

Questions?
  tiny.cloudera.com/ukquestions
Which streaming engine to choose?
Requirements

▪ Fault-tolerant and distributed
▪ Effectively once semantics
▪ Handle processing time vs. event time
▪ Allow stateful transformations
Spark Streaming

- Micro batch based architecture
- Allows stateful transformations
- Feature rich
  - Windowing
  - Sessionization
  - ML
  - SQL (Structured Streaming)
First Batch

Source -> Receiver -> DStream -> RDD

Source -> Receiver

RDD

Filter -> Count -> Print

Second Batch

Source -> Receiver

RDD

Filter -> Single Pass

Count -> Print
Spark Streaming - Gaps

- Not as low of a latency
  - Efforts towards reducing latency e.g. RISElab’s Drizzle

- Global consistent execution state
  - Stop overall execution of distributed computation
  - Eagerly persist records in transit meaning larger snapshots
Flink

- True “streaming” system, but not as feature rich as Spark
- Much better event time handling
- Good built-in backpressure support
- Allows stateful transformations
- Cleaner APIs
  - for (Triggering, Evicting, State management)
- Lower Latency
  - No Micro Batching
  - Asynchronous Barrier Snapshotting (ABS)
Flink - ABS
Flink - ABS

Questions?
tiny.cloudera.com/ukquestions
Storm

- Old school
- Didn’t manage state – had to use Trident
- No good support for batch processing
Samza

- Good integration with Kafka
- Doesn’t support batch
- Forked by Kafka Streams
Flume

- Well integrated with the Hadoop ecosystem
- Allowed interceptors (for simple transformations)
- Supports buffering
  - Memory
  - File
  - Kafka
- But no real fault-tolerance
- No state management
Kafka Streams

- Good integration with Kafka
- Light-weight library (not a framework)
- No micro-batching, uses Kafka as internal messaging layer
- Maintains local state per node (in RocksDB, or in memory hash map)
- Handles late events
- Stream-to-stream joins
Kafka Streams architecture

- Topic
  - Partition 1
  - Partition 2

- Task 1
- Task 2

- Re-partition topic
  - Partition 1
  - Partition 2

- Task 3
- Task 4
Storage Layer
Considerations
High level architecture

Data Sources
- Code
- Agents
- Log Aggregators

Producers
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Streaming Pipes
- Code
- Agents
- Log Aggregators

Storage
- File/Object
- RDBMS/MPP
- Time Series
- Reverse Indexed
- Memory
- Stream

Access
- SQL
- Machine Learning
- Request Response
- Batch Processing
Structured Landing Zones

- Long Term Storage
- Local - Remote
- Styles
  - Relational
  - Nested
- Speed Layer
- Time Series
- Reverse Index
- Graph
HDFS
Basic of GFS => HDFS

- NameNode
- Metadata of all the files/blocks
- Which data node they are assigned too
- Replication management
- Data Nodes
- Metadata for each block location on disk
Basic of GFS => HDFS

Write Path
A. Ask Name Node for Location to Write
B. Write to DataNode with NN Instructions
C. DataNode does replication
D. Confirms file is persisted to client
Basic of GFS => HDFS

- File are immutable
- File can be of any type
- Files are block up into Blocks (128MB -> 1GB)
  - Metadata cost is at the Block not the data size
- File may be splittable or may not be when reading
Object Stores
Object Store (Like and not like HDFS)

- Like a HDFS
  - Contains files
  - Break up large files

- Not like a HDFS
  - Not really a file system and is more Key value like a NoSQL
  - Doesn’t have any metadata limit problem
  - Traversing Folder directories is more work
  - There is no rename, only copy and delete
  - Eventually consist issues with listing files
    - (seen with things like MR and Spark)
    - Can be mostly addressed with EMRFS
Object Store (Many Ideas Continue)

- Because it is like a File System
- Almost everything you learned from HDFS stores true
- Hive works the same
- Files and file formats are the same
Object Store (Thinking Remote)

- Unlike HDFS the storage is always remote
  - Not on the same nodes as the execution
- Which allow you to save money in the cloud
  - Execution nodes are expensive vs storage only
- Network will be used to Read and Write
  - In fact you are normally throttled well before the network limit of your node
- You will want the highest rates of compression possible
  - To save money on storage
  - To read and write faster
Background Information
Compression Styles and Entropy

Columns

Rows

Questions?
tiny.cloudera.com/ukquestions
Compression Styles and Entropy
Compression Codecs

- **Snappy**: 2x-3x : Fast Read, Fast Write
- **Lzo**: 2x-3x : Fast Read, Fast Write
- **Gzip**: ~8x: ~Fast Read, Normal Write
- **Default**: ~8x: ~Fast Read, Normal Write
- **BZip2**: ~10x ~Fast Read, Slow Write
- **Others** ..

- **Always be skeptical**
  - All data compresses differently
  - Use your own data
Introducing the Hive Metastore

- **Hive Metastore**
  - Adds a table like metadata layer over a file system, block store, NoSql, or other
  - Allows for SQL access
  - Allows for greater security options
  - Allows for external metadata
  - Allows for partitioning
Typical Hive Table

- ParentFolder
  - TableFolder
    - Date=20171212
      - DataFiles
      - DataFiles
    - Date=20171211
      - DataFiles
      - DataFiles
Access Patterns

- Partitioning
- Filter push down
- Indexing should be considered poor
- Ideal for large scans
- Bucketing and Sorting
Relational Storage
Thinking about Object/Tables

1. Lets start off easy

1. Use Case: We are a Netflix type company and we have a log of users and movies watched that looks something like this:

<table>
<thead>
<tr>
<th>User ID</th>
<th>Age</th>
<th>Account Start Date</th>
<th>Category Of User</th>
<th>Movie Watched</th>
<th>Movie Category</th>
<th>Start Time</th>
<th>Events List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>42</td>
<td>12/12/2012</td>
<td>Basic</td>
<td>Die Hard</td>
<td>Action</td>
<td>5/4/2016 12:00</td>
<td>Play 0, pause at 15, FF at 40 to 55, E at 90</td>
</tr>
<tr>
<td>Kat</td>
<td>31</td>
<td>12/12/2012</td>
<td>Platum</td>
<td>Beauty and the Beast</td>
<td>Family</td>
<td>5/4/2016 12:00</td>
<td>Play 0, pause at 15, FF at 40 to 55, E at 90</td>
</tr>
</tbody>
</table>
Thinking about Object/Tables

1. To make this into objects we need to do some separation

- **User**
  - User_id
  - Age
  - St_dt
  - Category

- **Movie**
  - Movie_id
  - Title
  - Category

- **Category_Typ**
  - Category_id
  - Stream_rt
  - Is_feature_enabled

- **Watch_session**
  - Watch_id
  - St_dt
  - En_dt
  - User_id
  - Movie_id

- **Watch_Events**
  - Watch_id
  - St_dt
  - Type
  - Duration
Query Considerations

- Data is normally big so
- Partition respectively to access patterns
- Join with care
- Consider sampling or local testing before experimenting
- Data is files
- Latency to accessibility it high – seconds, minutes or more.
Look for big tables

```
<table>
<thead>
<tr>
<th>User</th>
<th>Watch_session</th>
<th>Movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>User_id</td>
<td>Watch_id</td>
<td>Movie_id</td>
</tr>
<tr>
<td>Age</td>
<td>St_dt</td>
<td>Title</td>
</tr>
<tr>
<td>St_dt</td>
<td>En_dt</td>
<td>Category</td>
</tr>
<tr>
<td>Category</td>
<td>User_id</td>
<td>Is_feature_enabled</td>
</tr>
<tr>
<td>Category_Typ</td>
<td>Movie_id</td>
<td></td>
</tr>
<tr>
<td>Category_id</td>
<td>Watch_id</td>
<td></td>
</tr>
<tr>
<td>Stream_rt</td>
<td>St_dt</td>
<td></td>
</tr>
<tr>
<td>Is_feature_enabled</td>
<td>Type</td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Mutation Patterns

- File is written once and can not be mutated
- Fine for append or snapshot use cases
- Mutation will require a compaction
## Compaction Recap

<table>
<thead>
<tr>
<th>Key</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>101</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>103</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>103</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>103</td>
</tr>
</tbody>
</table>

Questions?
[tiny.cloudera.com/ukquestions]
View Strategies

- **Hive Relational Model**
- **Hive Nested Model**
- **Hive Normal Views**
  - Use only for tables that filter records/columns or use for marking fields
- **Hive Materialized Table Views**
  - Use in the cases where the view requires a join that is done through a shuffle

Questions?
[ tiny.cloudera.com/ukquestions ]
Nested Structures
Nested

- Less Space than Denormalization
- Still have tables but the cost of joins is all but gone
- Also great for cartesian joins
  - N x M vs N + M
- Not really supported yet with Kudu or HBase with SQL
Nested Example

CREATE TABLE fact_contacts (id BIGINT, name STRING, address STRING) STORED AS PARQUET;
CREATE TABLE dim_phones
(
    contact_id BIGINT,
    category STRING,
    international_code STRING,
    area_code STRING,
    exchange STRING,
    extension STRING,
    mobile BOOLEAN,
    carrier STRING,
    current BOOLEAN,
    service_start_date TIMESTAMP,
    service_end_date TIMESTAMP
)
Nested Example

CREATE TABLE contacts_detailed_phones
(
  id BIGINT, name STRING, address STRING,
  phone ARRAY < STRUCT <
    category: STRING
  , international_code: STRING
  , area_code: STRING
  , exchange: STRING
  , extension: STRING
  , mobile: BOOLEAN
  , carrier: STRING
  , current: BOOLEAN
  , service_start_date: TIMESTAMP
  , service_end_date: TIMESTAMP
  >>
)
) STORED AS PARQUET;

https://www.cloudera.com/documentation/enterprise/latest/topics/impala_complex_types.html
De-normalized vs Nested

- **Nested Pros**
  - Co-location
  - Faster to group by
  - Faster to window
  - Joins are free
  - Less data
  - Better compression
  - Tables and Columns can be read without penalty from one not read
  - Great for limiting the effort are Cartesian Joins

- **Nested Cons**
  - Size limitation of parent row
  - Adding child requires the re-write the whole parent record
Options for appending Nested

- It is all about the parent record
- We can add more than one Partition key for the parent
- In our use case
  - User & watch month or day
Storage and In Memory

- Also don’t limit the idea of nested to just tables
- In Spark they can be used as in memory constructs to
  - conserve on networking
  - In memory cost
Nested Writing Example in Spark

```json
{
  "id": "0001",
  "type": "donut",
  "name": "Cake",
  "ppu": 0.55,
  "batters":
  {
    "batter":
    [  
      { "id": "1001", "type": "Regular" },
      { "id": "1002", "type": "Chocolate" },
      { "id": "1003", "type": "Blueberry" },
      { "id": "1004", "type": "Devil's Food" }
    ]
  },
  "topping":
  [  
    { "id": "5001", "type": "None" },
    { "id": "5002", "type": "Glazed" },
    { "id": "5005", "type": "Sugar" },
    { "id": "5007", "type": "Powdered Sugar" },
    { "id": "5006", "type": "Chocolate with Sprinkles" }
  ]
}
```
Nested Writing Example in Spark

```scala
val jsonDF = hiveContext.read.json(jsonRDD)
jsonDF.write.parquet("./parquet")
hiveContext.createExternalTable("jsonNestedTable", "/parquet")
```
Time Series
Time Series Options

- HBase
- Cassandra
- InfluxDB
- Data Dog
- DruidDB
- Many More
Entity Centric Time Series

- Partition by Entity ID
- Order by Time
- Allows for free windowing
- Allows for fetching of single time window of single entity at web scale
HBase Entity Time Series

- Cust-A, 10
- Cust-A, 20
- Cust-A, 40
- Cust-C, 10
- Cust-C, 20
- Cust-C, 30
- Cust-C, 40

- Cust-B, 10
- Cust-B, 20
- Cust-B, 30
- Cust-B, 40
- Cust-F, 20
- Cust-F, 30
- Cust-F, 40

- Cust-D, 10
- Cust-D, 20
- Cust-D, 40
- Cust-G, 10
- Cust-G, 20
- Cust-G, 30
- Cust-G, 40

Questions?
tiny.cloudera.com/ukquestions
HBase Entity Time Series

Rest Call → Short Scan

Cust-A, 10
Cust-A, 20
Cust-A, 40
Cust-C, 10
Cust-C, 20
Cust-C, 30
Cust-C, 40
Cust-B, 10
Cust-B, 20
Cust-B, 30
Cust-B, 40
Cust-F, 20
Cust-F, 30
Cust-F, 40
Cust-D, 10
Cust-D, 20
Cust-D, 40
Cust-G, 10
Cust-G, 20
Cust-G, 30
Cust-G, 40

Questions?
tiny.cloudera.com/ukquestions
HBase Entity Time Series

Mapper

Cust-A, 10
Cust-A, 20
Cust-A, 40
Cust-C, 10
Cust-C, 20
Cust-C, 30
Cust-C, 40

Mapper

Cust-B, 10
Cust-B, 20
Cust-B, 30
Cust-B, 40
Cust-F, 20
Cust-F, 30
Cust-F, 40

Mapper

Cust-D, 10
Cust-D, 20
Cust-D, 40
Cust-G, 10
Cust-G, 20
Cust-G, 30
Cust-G, 40
HBase Entity Time Series

Cust-A, 10
Cust-A, 20
Cust-A, 40
Cust-C, 10
Cust-C, 20
Cust-C, 30
Cust-C, 40

Cust-B, 10
Cust-B, 20
Cust-B, 30
Cust-B, 40
Cust-F, 20
Cust-F, 30
Cust-F, 40

Cust-D, 10
Cust-D, 20
Cust-D, 40
Cust-G, 10
Cust-G, 20
Cust-G, 30
Cust-G, 40

Questions?
tiny.cloudera.com/ukquestions
What is meant by Bucketing and Sorting

- Partitioning on a Key
- Then sorting on that key + another field(s)
- Example
  - User_id + Watch Event Time
# Example of Bucketed Sorted

<table>
<thead>
<tr>
<th>Cust-A</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust-A</td>
<td>20</td>
</tr>
<tr>
<td>Cust-A</td>
<td>40</td>
</tr>
<tr>
<td>Cust-C</td>
<td>10</td>
</tr>
<tr>
<td>Cust-C</td>
<td>20</td>
</tr>
<tr>
<td>Cust-C</td>
<td>30</td>
</tr>
<tr>
<td>Cust-C</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cust-B</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust-B</td>
<td>20</td>
</tr>
<tr>
<td>Cust-B</td>
<td>30</td>
</tr>
<tr>
<td>Cust-F</td>
<td>10</td>
</tr>
<tr>
<td>Cust-F</td>
<td>20</td>
</tr>
<tr>
<td>Cust-F</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cust-D</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cust-D</td>
<td>20</td>
</tr>
<tr>
<td>Cust-D</td>
<td>40</td>
</tr>
<tr>
<td>Cust-G</td>
<td>10</td>
</tr>
<tr>
<td>Cust-G</td>
<td>20</td>
</tr>
<tr>
<td>Cust-G</td>
<td>30</td>
</tr>
<tr>
<td>Cust-G</td>
<td>40</td>
</tr>
</tbody>
</table>
Good for Appending Nested

**New Data**
- Cust-A, 50
- Cust-A, 60
- Cust-B, 50
- Cust-B, 60
- Cust-C, 50
- Cust-D, 50
- Cust-G, 50

**Existing Data**
- Cust-A, 10
- Cust-A, 20
- Cust-A, 40
- Cust-C, 10
- Cust-C, 20
- Cust-C, 30
- Cust-C, 40
- Cust-B, 10
- Cust-B, 20
- Cust-B, 30
- Cust-B, 40
- Cust-F, 10
- Cust-F, 20
- Cust-F, 40
- Cust-D, 10
- Cust-D, 20
- Cust-D, 40
- Cust-G, 10
- Cust-G, 20
- Cust-G, 30
- Cust-G, 40

Questions?
tiny.cloudera.com/ukquestions
Good for Appending Nested

New Data

Existing Data

Cust-A, 50  Cust-A, 10  Cust-B, 10  Cust-D, 10
Cust-A, 60  Cust-A, 20  Cust-B, 20  Cust-D, 20
Cust-B, 50  Cust-A, 40  Cust-B, 30  Cust-D, 40
Cust-B, 60  Cust-C, 10  Cust-B, 40  Cust-G, 10
Cust-C, 50  Cust-C, 20  Cust-F, 10  Cust-G, 20
Cust-D, 50  Cust-C, 30  Cust-F, 20  Cust-G, 30
Cust-G, 50  Cust-C, 40  Cust-F, 40

Shuffle Join

Questions?
tiny.cloudera.com/ukquestions
Good for Appending Nested

New Data

Existing Data

Cust-A, 50
Cust-A, 60
Cust-C, 50

Cust-B, 50
Cust-D, 50
Cust-G, 50

Cust-A, 10
Cust-A, 20
Cust-A, 40
Cust-C, 10
Cust-C, 20
Cust-C, 30
Cust-C, 40
Cust-B, 10
Cust-B, 20
Cust-B, 30
Cust-B, 40
Cust-F, 10
Cust-F, 20
Cust-F, 40
Cust-D, 10
Cust-D, 20
Cust-D, 40
Cust-G, 10
Cust-G, 20
Cust-G, 30
Cust-G, 40
Cust-B, 50
Cust-B, 60

Merge Join

Questions?
tiny.cloudera.com/ukquestions
Good for Appending Nested
NoSQL
What is a NoSQL

- It’s not NO SQL
- It’s not a Database
- Think of it more like a
  - HashMap
  - Log
  - Bucketed and Ordered
Hash Map

- There is a Key and a Value
- It is really fast to grab a key/value
- It is really fast to add a key/value
- Iteration is also possible

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
</tr>
</tbody>
</table>

Client
Log with Compactions

- When new records come in they don’t rewrite the old
- They compact in

<table>
<thead>
<tr>
<th>Key</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>101</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>103</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>103</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>103</td>
</tr>
</tbody>
</table>
Log with Compactions

- **Write Path**
  - Get Local for Record (Cached)
  - First to WAL
  - Then to Memstore
  - Sorting & batching
  - Flush to New Hfile
  - Later Hfiles will be compacted
Ordered

- All Records Columns are ordered
- Ordering allows for simpler indexing
- Ordering allows for simpler compactions
- We will also use this ordering
  - Windowing
  - Time series
  - Local scanning
Bucketing or Partitions

- **HBase**
  - Out of the Box:
    - Range
  - Desired:
    - Salt
- **Cassandra**
  - Out of the Box:
    - HashMod
  - Bucketed HashMod
So what about SQL

- **Well SQL could totally work**
  - CQL for cassandra
  - Hive and SparkSQL on Hbase & Cassandra

- **Why is it not the best idea**
  - Built more for point look ups
  - Scans are not as fast as parquet
    - However the mutability may be more important than speed
  - Partitioning is not simple
    - It must be put into the key
Let’s talk about CAP for a Minute

- Strong Consistency
- HBase & Kudu
- Variable Consistency
- Cassandra
HBase Model

- Region Server owns range splits
- Region Server 1 fails
- Master needs to figure that out
- Master needs to assign new Region Server to own splits
- Region Server 2 has to get organized
- Region Server 2 is read to server reads and writes
Cassandra Model

Questions?
tiny.cloudera.com/ukquestions
Cassandra Model
Cassandra Model (Common Models)

3 Write - 1 Read

1 Write - 3 Read

1 Write - 1 Read
What do they share (Head Types)

- Master and a HA

- Leader Election
What do they share (Headless)
What do they share (CAP theorem)

- Consistency
- Availability
- Partitioned
What do they share (CAP theorem)

Consistency

Doesn’t Exist

Strong Consistence

Availability

Partitioned

Eventual Consistence
What do they share (CAP theorem)

- Cheating the CAP Theorem
- Cassandra is a good model
- Where they expand the definition of failure with variable consistence
- CAP still holds but …

![Diagram showing the CAP theorem with Consistency, Availability, and Eventual Consistency]

Questions?
tiny.cloudera.com/ukquestions
Indexed Search
Lucene Indexing (Features)

- We don’t have enough time in this whole class
- Ordering logic
- NGrams
- Weights
- Text Indexing
- Translations
- Facets *
Lucene Indexing (Facets)

- Facets are a side effect of our wonderful indexes
- It allows us to count all the documents that belong to given indexes to produce
  - Grouped Counts
  - Charts and Graphs (Kibana or Banana)
- People will also call this access pattern cubing a dataset
Lucene Indexing (Kibana & Banana)

Questions?
tiny.cloudera.com/ukquestions
## Lucene Indexing (Facets Example)

- **Time Series Example**

<table>
<thead>
<tr>
<th>Document ID</th>
<th>Hour of Day</th>
<th>User</th>
<th>State</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>4201</td>
<td>MD</td>
<td>click</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4202</td>
<td>VA</td>
<td>click</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4203</td>
<td>VA</td>
<td>click</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4201</td>
<td>MD</td>
<td>click</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4202</td>
<td>VA</td>
<td>view</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4204</td>
<td>CA</td>
<td>click</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>4205</td>
<td>VA</td>
<td>view</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4201</td>
<td>MD</td>
<td>click</td>
</tr>
</tbody>
</table>
# Lucene Indexing (Facets Example)

<table>
<thead>
<tr>
<th>Document ID</th>
<th>Hour of Day</th>
<th>User</th>
<th>State</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>4201</td>
<td>MD</td>
<td>click</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4202</td>
<td>VA</td>
<td>click</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4203</td>
<td>VA</td>
<td>click</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4201</td>
<td>MD</td>
<td>click</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4202</td>
<td>VA</td>
<td>view</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4204</td>
<td>CA</td>
<td>click</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>4205</td>
<td>VA</td>
<td>view</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4201</td>
<td>MD</td>
<td>click</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>4204</td>
<td>CA</td>
<td>click</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hour of Day</th>
<th>12</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>MD</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>VA</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>CA</th>
<th>6</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>click</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>view</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>4201</th>
<th>1</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4202</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4203</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4204</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4205</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lucene Indexing (Facets Example)

- Events per hour
- Simple array count

| Hour of Day | | | | |
|-------------|-----------|-----------|-----------|
| 12          | 1         | 2         | 3         |
| 1           | 4         | 5         |           |
| 2           | 6         | 7         | 8         |

Questions?
tiny.cloudera.com/ukquestions
## Lucene Indexing (Facets Example)

- Events per hour by State
- Simple array count

<table>
<thead>
<tr>
<th>Hour of Day</th>
<th>12</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>MD</th>
<th>VA</th>
<th>CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>VA</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>CA</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

### Events per hour by State

- **MD**: 1, 4, 8
- **VA**: 2, 3, 5, 7
- **CA**: 6, 9

Questions?
tiny.cloudera.com/ukquestions
Lucene Indexing (Facets Example)

- Note the bucketing and ordered pattern

<table>
<thead>
<tr>
<th>Hour of Day</th>
<th>12</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>MD</th>
<th>VA</th>
<th>CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>VA</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>CA</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hour of Day 2</th>
<th>State MD</th>
<th>State VA</th>
<th>State CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>
Lucene Indexing (Facets Example)

- Note the bucketing and ordered pattern

<table>
<thead>
<tr>
<th>Hour of Day 2</th>
<th>State MD</th>
<th>State VA</th>
<th>State CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hour of Day 2</th>
<th>State MD</th>
<th>State VA</th>
<th>State CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+1 CA
Lucene Indexing (Facets Example)

Note the bucketing and ordered pattern

<table>
<thead>
<tr>
<th>Hour of Day 2</th>
<th>State</th>
<th>State</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MD</td>
<td>VA</td>
<td>CA</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

+1 VA

<table>
<thead>
<tr>
<th>Hour of Day 2</th>
<th>State</th>
<th>State</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MD</td>
<td>VA</td>
<td>CA</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

+1 MD

+1 CA

Questions?
tiny.cloudera.com/ukquestions
Partitioning

- Solr and Elastic Search partition the document to land on all nodes
- This means
  - You have the power of the cluster when querying
  - This means you are accessing the cluster when querying
Writing Latency

- Lucene Indexing is more expensive than NoSQL work
- Think of it as micro batching
  - Larger batches ~= better throughput
- Compaction is also invalid
- Deletes impact storage and performance until they are compacted
Storage Cost

- TTL is your friend
- Think of Lucene based systems as great if
  - You dataset is manageable in size
  - You have a good TTL strategy
  - You have a boat load of money
Graphs
Thinking in terms of Graphs

- Nodes and Edges

```
Node:1 ...
   Friend
Node:2 ...
   Coach
Node:0 ...
   Wife
Node:3 ...
   Child
   Father
```
Thinking in terms of Graphs

- Use cases
- Querying
  - Cassandra with Sparkle
  - Neo4j
- Batch operations
  - Giraph
  - GraphX
  - GraphLab
BSP Bulk Synchronous Parallel

- Process every Node Atomically
  - Node gets all messages sent to it
  - Nodes can mutate themselves and their edges
  - Nodes can send messages to other nodes
    - But nothing is received yet
  - BSP waits until all the Node processing is done
- Then send messages to the right partition
- Repeat
Honorable Mentions

Kudu
CockroachDB
Druid.io
Why Honorable Mentions
Kudu

1. Replace the Region Servers with Tablet Servers
2. Replace block format HFile files with a parquet like TFiles
3. Replace the byte array focused HBase API with one that is more JDBC friendly
4. Tight integration with Spark SQL and Impala for SQL
5. Completely rewrite the compaction process to make for perfectly sized files with our having major compactions but always manageable micro compactions.
Kudu

Questions?
tiny.cloudera.com/ukquestions
CoackroachDB

1. Think about Kudu but
   1. Nested tables
   2. Transitions
Druid.IO

1. Not a general SQL type store
2. More focused on metric
   1. Where larger aggregations may be required
   2. Many variables for same metric tag combinations
Druid.IO

Druid.IO

Client

Query Planning and Response Preparation

Broker Cluster

Broker Node

Broker Node

Broker Node

ZooKeeper

Metadata Storage

House keeping services

Real Time Cluster

Real Time Node

Real Time Node

Real Time Node

Short TTL for hot memory cache

Main Ingestion Path

Batch Ingestion

Streaming Ingestion

Async large batching

History Cluster

History Node

History Node

History Node

Pluggable Storage

Try to optimize read path based on time request of the query

#StrataData

tiny.cloudera.com/ukquestions
Data Access
High level architecture

Data Sources
- Code
- Agents
- Log Aggregators

Producers
- Schema Validation
- Enrichment
- Stream Processing
- Routing
- Transport
- Replication

Streaming Pipes
- Long Term SQL
- Speed Layer SQL
- Time Series
- Reverse Indexed
- Stream

Storage
- SQL
- Machine Learning
- Request Response
- Batch Processing

Access
Data Access

Batch Processing
Why have batch processing?

- When you need a larger context
  - Say, to train a model
- Complex periodic job that does something
  - Convert data to a nested structure for reduced number of shuffles
- For example,
  - Kudu -> HDFS Nested is batch processing
  - KMeans calculation, etc.
Types of Processing

- ETL
- Analytics
- Graph Analytics
- Machine Learning
- Deep Learning
Batch processing options

- Spark (+ MLlib)
- MapReduce (+ Mahout)
- Flink (+ Flink ML)
- Distributed query engines
Spark

- Pretty popular
- Much faster than MapReduce
- Thriving community
MapReduce

- Sloooooow
Flink

- Pretty popular
- Batch is a special case of Streaming
- Developing community
Just SQL

- Apache Spark SQL
- Hive
  - MR, Spark, Tez
- Impala
- Presto
Distributed Machine Learning

- Apache Spark MLLib
- Apache Spark Custom
- H2o
- Data Robot
- Google Tensorflow
- Microsoft Cognitive
GPUs vs CPUs

- GPU are much better for floating point operations
  - Consider Gradient Descent
- Encourages hardware to be separated from storage
Data Access

Interactive Access
Types of data access

- REST server/APIs for querying entities and aggregates
- UI for displaying search facets
- SQL engine
REST servers

Considerations
REST Servers

```scala
import org.mortbay.jetty.Server
import org.mortbay.jetty.servlet.{Context, ServletHolder}

val server = new Server(port)
val sh = new ServletHolder(classOf[ServletContainer])
sh.setInitParameter("com.sun.jersey.config.property.resourceConfigClass", "com.sun.jersey.api.core.PackagesResourceConfig")
sh.setInitParameter("com.sun.jersey.config.property.packages", "com.hadooparchitecturebook.taxi360.server.hbase")
sh.setInitParameter("com.sun.jersey.api.json.POJOMappingFeature", "true")
val context = new Context(server, "/", Context.SESSIONS)
context.addServlet(sh, "/*")
server.start()
server.join()
```
UI
Considerations
UI requirements

Something that can

▪ Represent search results really well
▪ Integrates with Apache Solr, query engines, etc.
UI options

- Hue
- Banana
- Kibana

Questions?
tiny.cloudera.com/ukquestions
Hue In Our Use Case

- Because it’s included
- Please look at the others
Data Access

SQL Engines
SQL engine criteria

- Low latency SQL access
- Allows for high concurrency
- JDBC/ODBC integration
- Capable of large scale aggregation
- Optionally integrates with Multiple Storage Systems for real-time updates to SQL tables
Apache Hive

- Good JDBC integration
- Not really low latency, even when using Tez
- Doesn’t integrate with Kudu
- Can run with MapReduce, Spark, or Tez
Presto

- Low latency SQL engine from Facebook
- Provides JDBC/ODBC access
- Is only in-memory, large aggregations can lead to OOM errors
- Doesn’t integrate with Kudu
Apache Impala

- Low latency SQL access
- Provides JDBC/ODBC access
- Excellent concurrency support
- Integrates with Kudu for real-time SQL
Apache Drill

- Similar in architecture to Impala
- Provides JDBC/ODBC access
- Doesn’t integrate with Kudu
Spark SQL

- Builds on top of Spark
- JDBC/ODBC access only via Spark Thrift Server
  - Doesn’t scale well with larger number of concurrent users
  - Doesn’t fully provide secure access.
Where else to find us?
Other Sessions

▪ Big Data at Speed (with Mark Grover) – Thursday, 12:05
Thank you!

@hadooparchbook
tiny.cloudera.com/app-arch-london
Gwen Shapira | @gwenshap
Jonathan Seidman | @jseidman