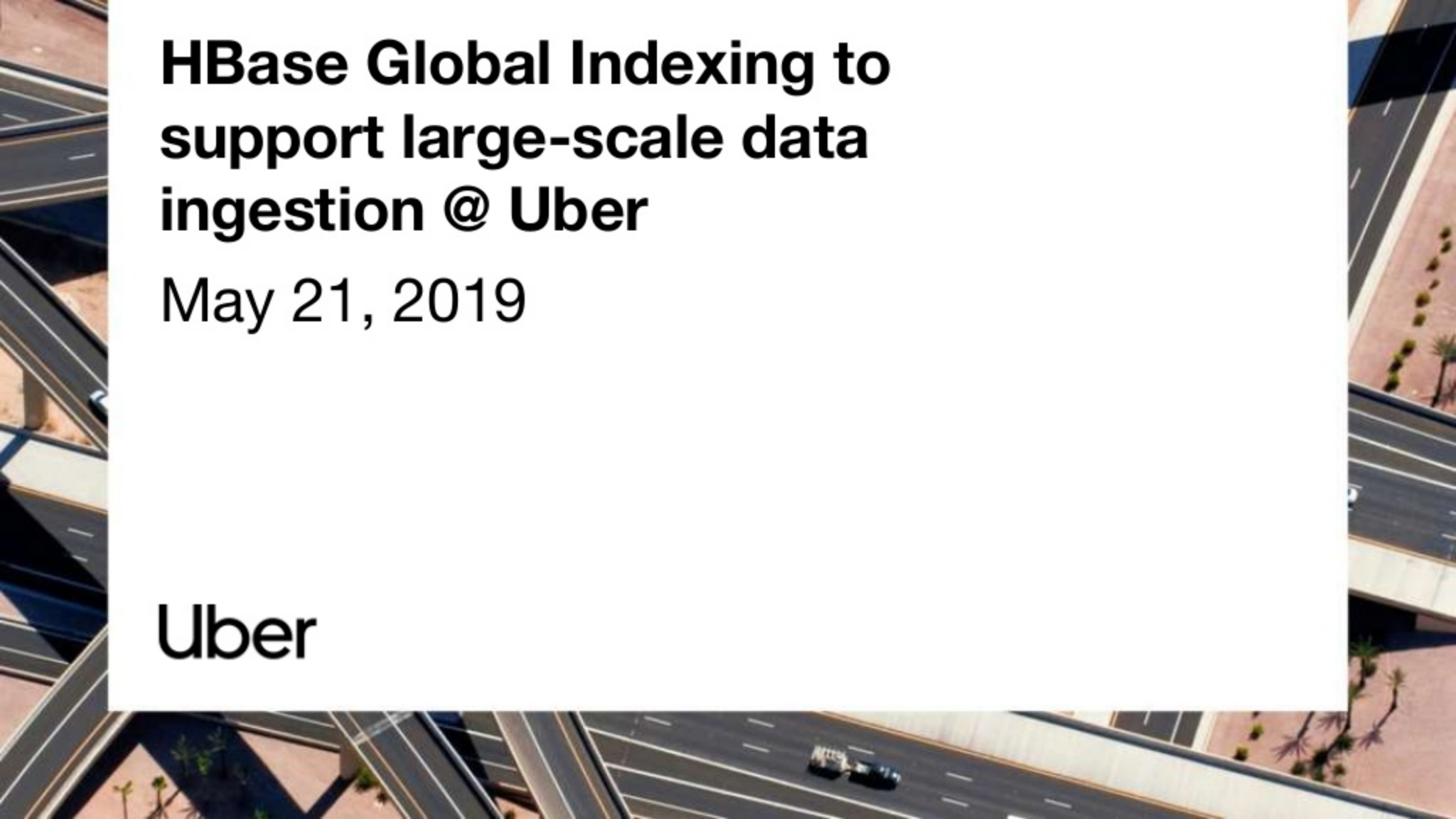


HBase Global Indexing to support large-scale data ingestion @ Uber

May 21, 2019

Uber

Danny Chen

dannyc@uber.com

- Engineering Manager on Hadoop Data Platform team
- Leading Data Ingestion team
- Previous worked @  on storage team (Manhattan)
- Enjoy playing basketball, biking, and spending time w/my kids.

Uber Apache Hadoop Platform Team Mission

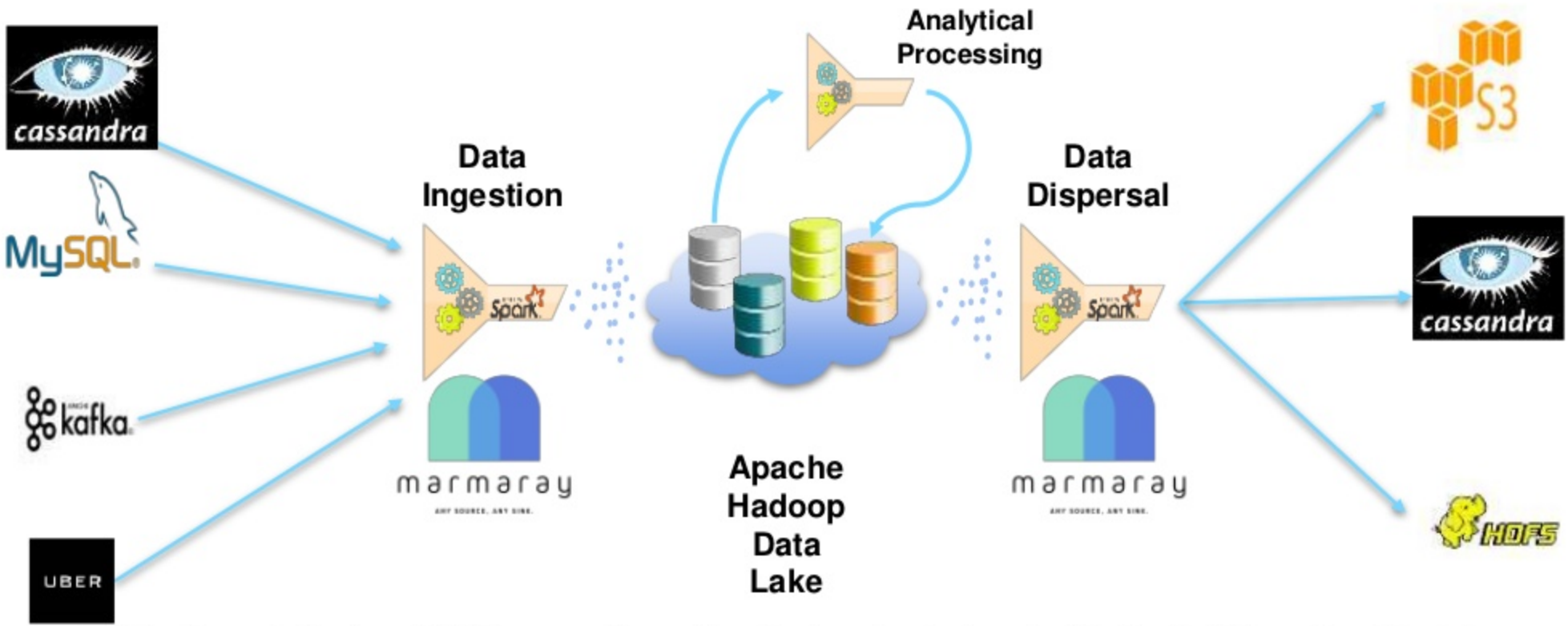
Build products to support reliable, scalable, easy-to-use, compliant, and efficient data transfer (both ingestion & dispersal) as well as data storage leveraging the Apache Hadoop ecosystem.

Overview

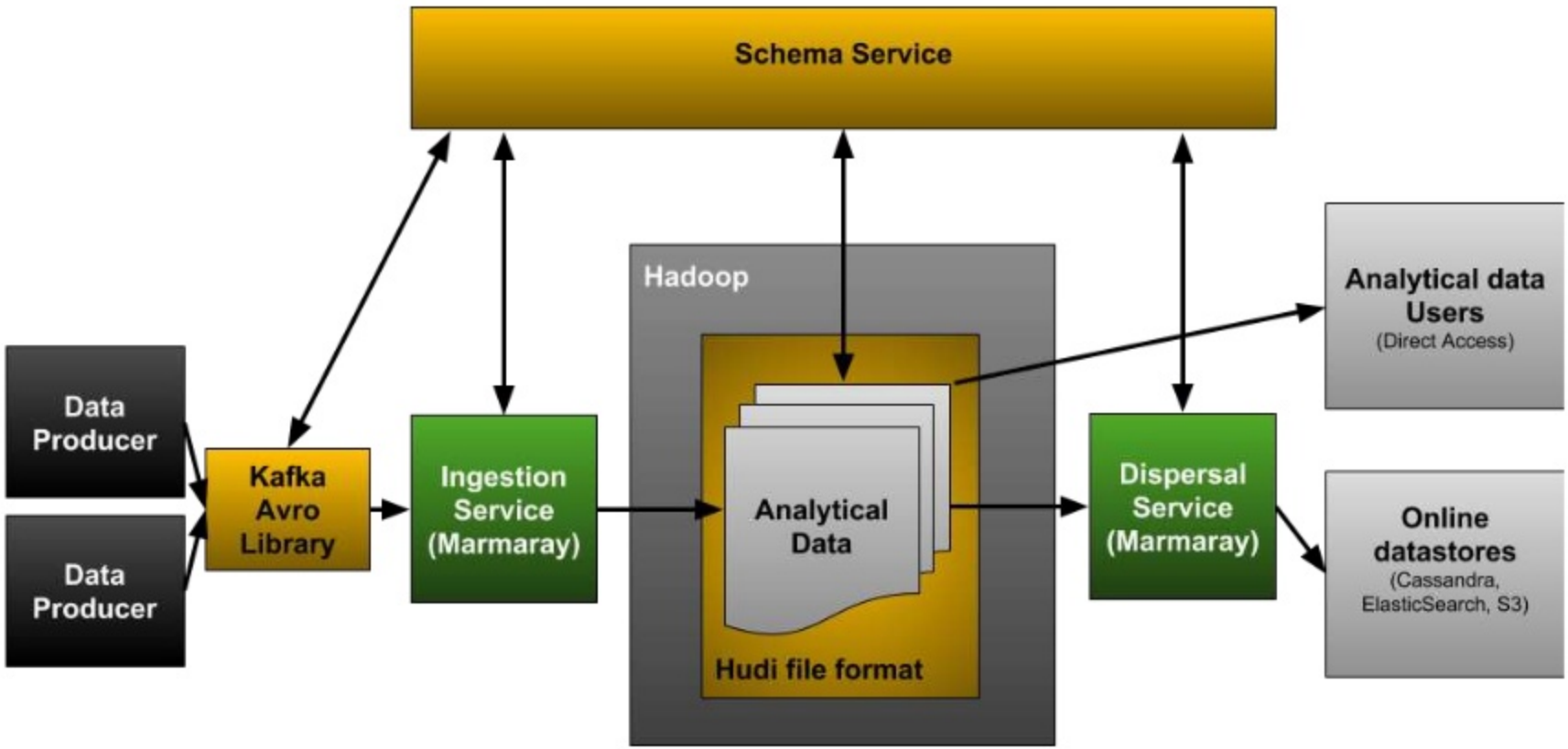
- High-Level Ingestion & Dispersal introduction
- Different types of workloads
- Need for Global Index
- How Global Index Works
- Generating Global Indexes with HFiles
- Throttling HBase Access
- Next Steps

High Level Ingestion/Dispersal Introduction

Hadoop Data Ecosystem at Uber



Hadoop Data Ecosystem at Uber



Different Types of Workloads

Bootstrap

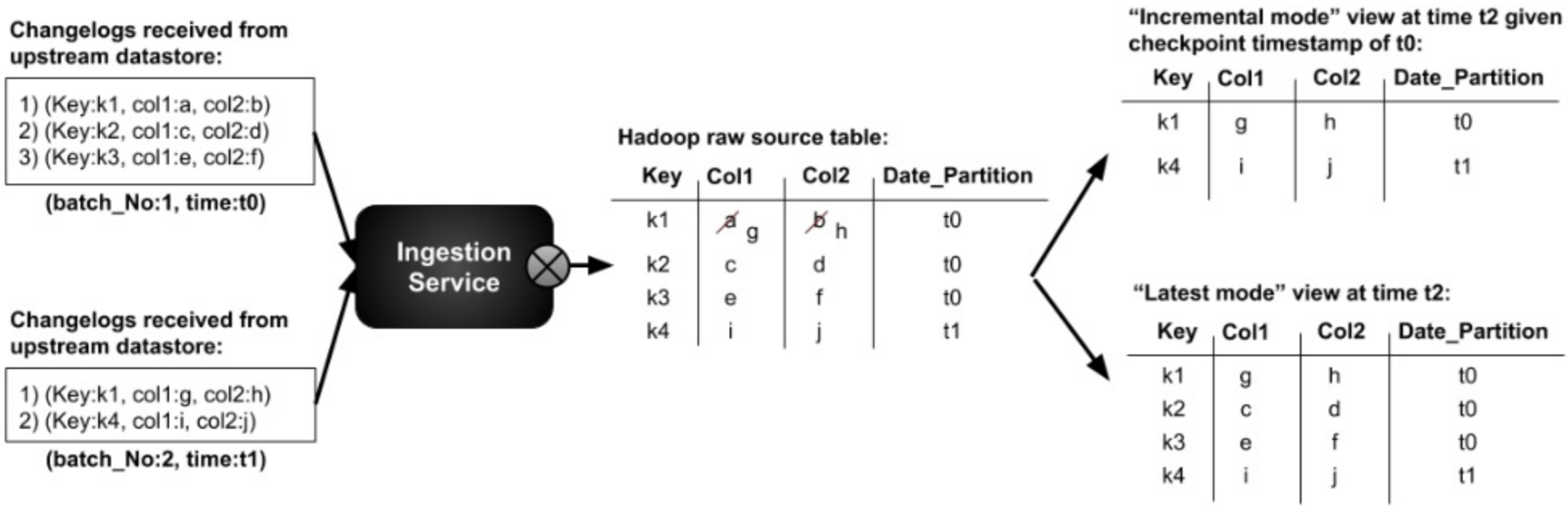
- One time only at beginning of lifecycle
- Large amounts of data
- Millions of QPS throughput
- Need to finish in a matter of hours
- NoSQL stores cannot keep up

Incremental

- Dominates lifecycle of Hive table ingestion
- Incremental upstream changes from Kafka or other data sources.
- 1000's QPS per dataset
- Reasonable throughput requirements for NoSQL stores

Cell vs Row Changes

"Latest mode" vs "Incremental mode" for reading a Hive table that is being updated



Need for Global Index

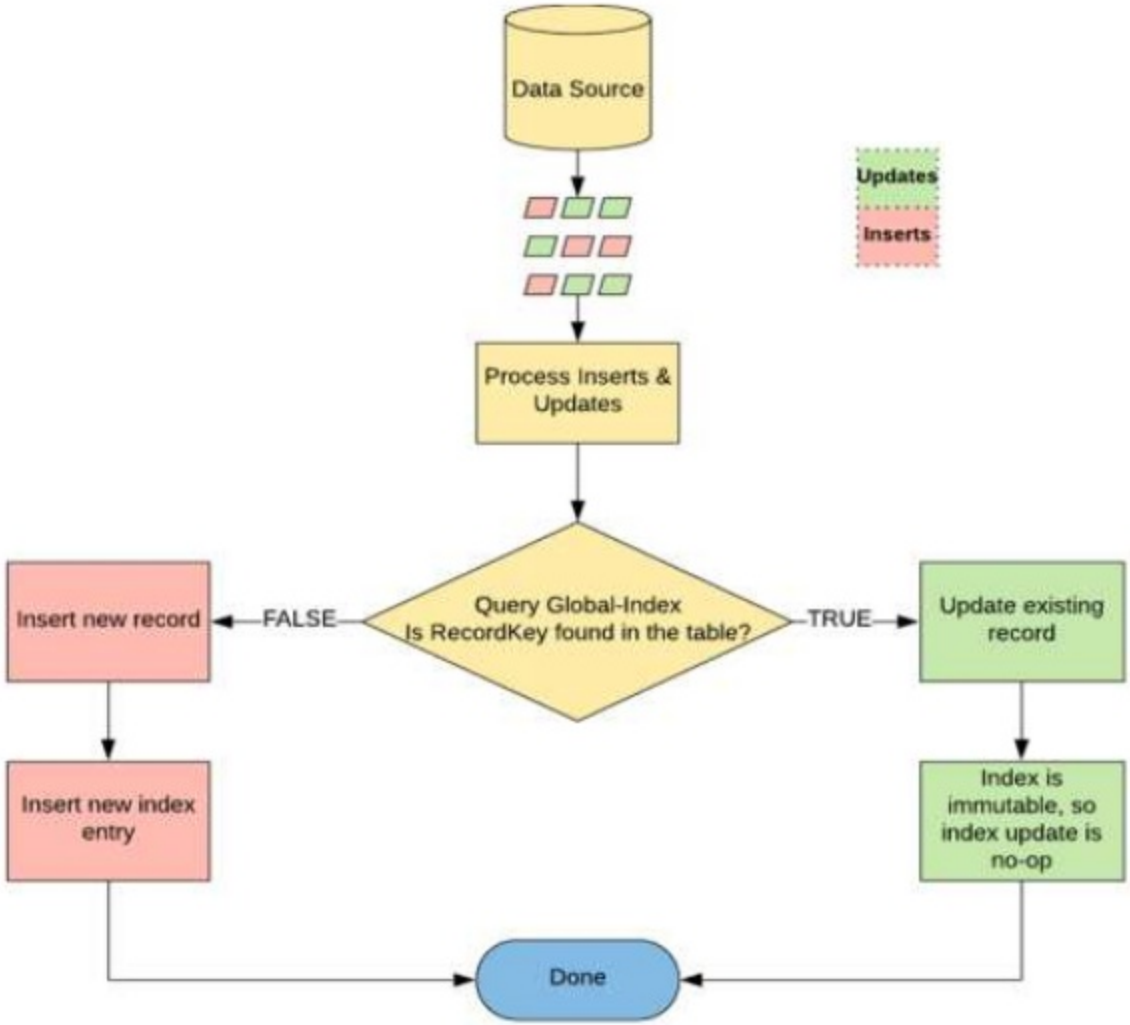
Requirements for Global Index

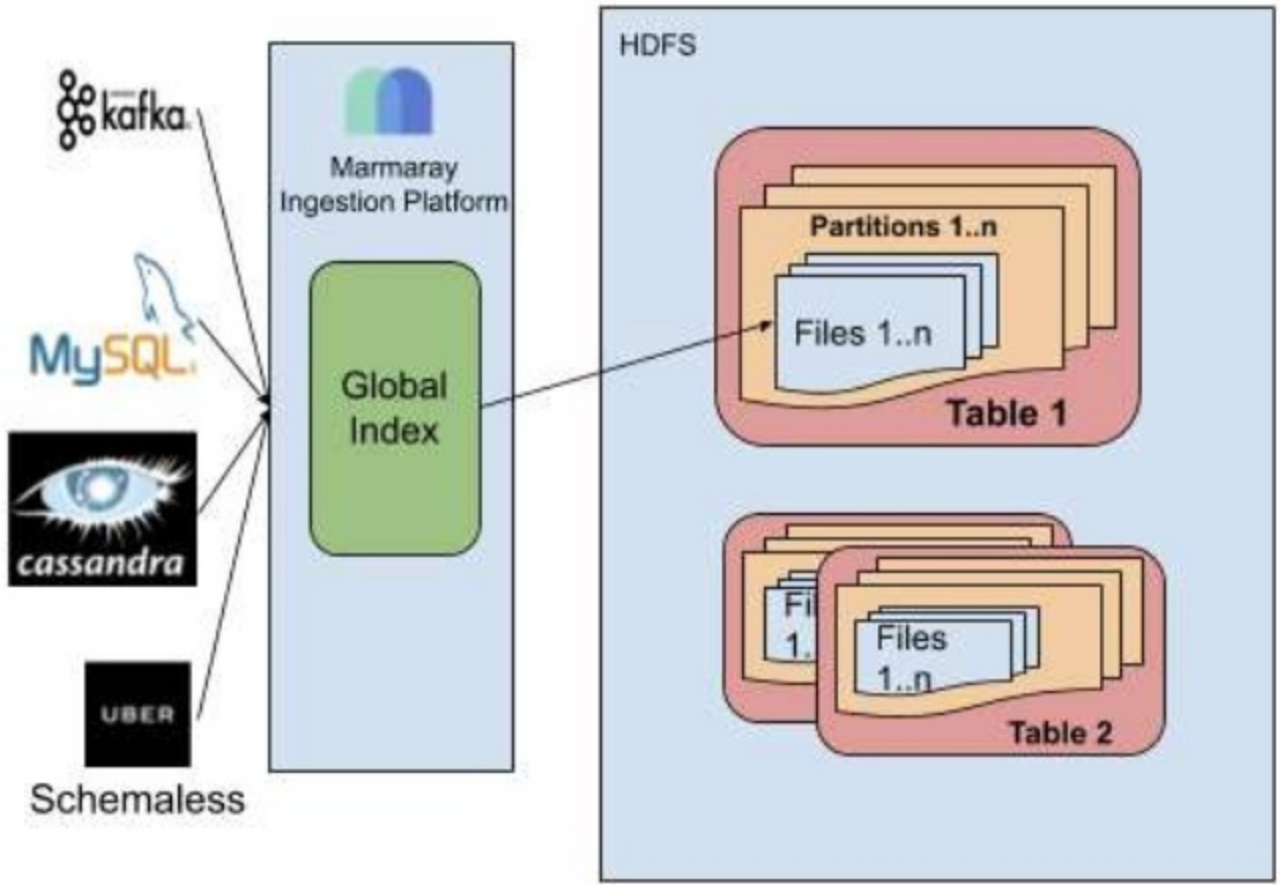
- Large amounts of historical data ingested in short amount of time
- Append only vs Append-plus-update
- Data layout and partitioning
- Bookkeeping for data layout
- Strong consistency
- High Throughput
- Horizontally scalable
- Required a NoSQL store

VS

- Decision was to use HBase
- Trade Availability for Consistency
- Automatic Rebalancing of HBase tables via region splitting
- Global view of dataset via master/slave architecture

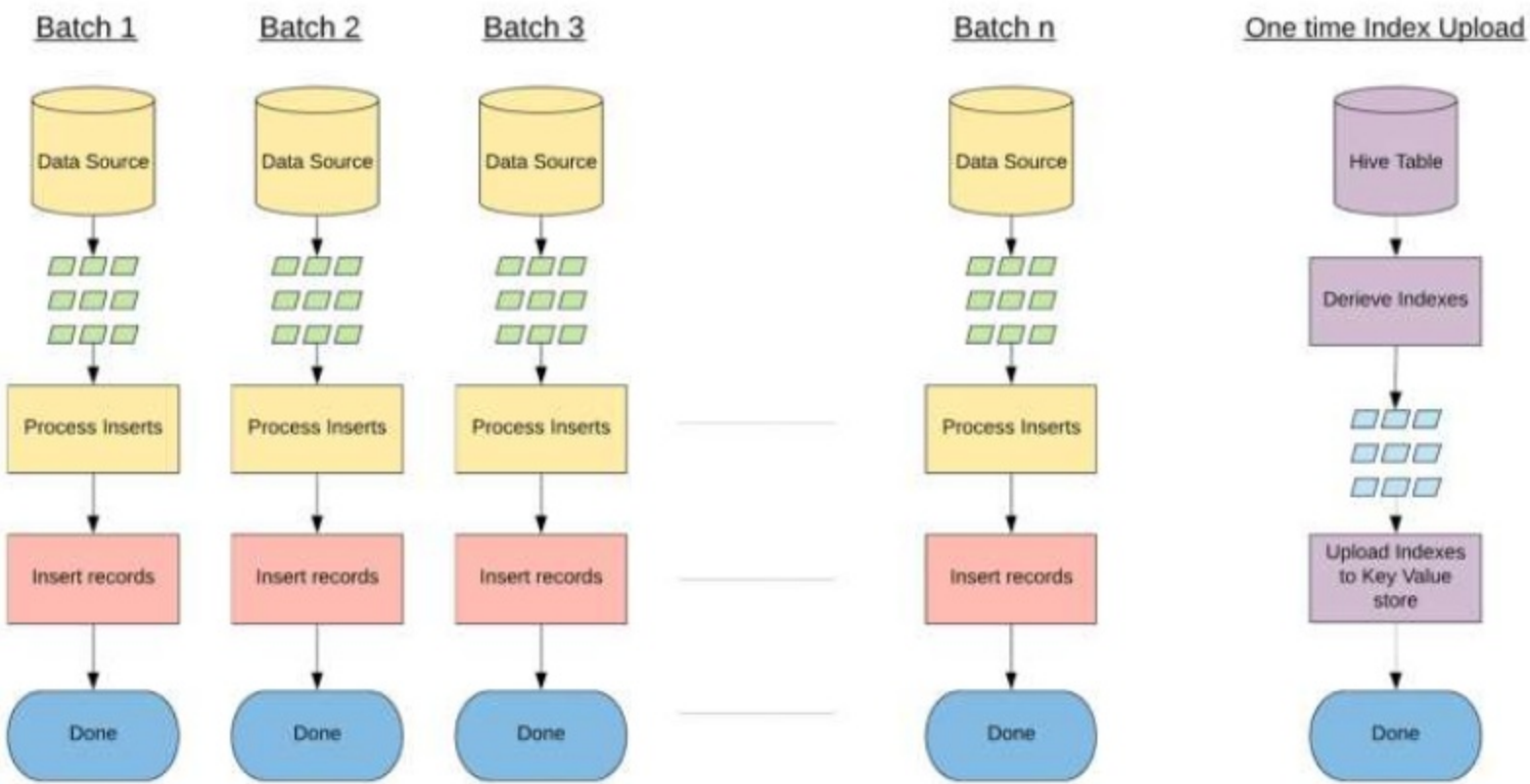
How Global Index Works


Data Source
Updates
Inserts
Process Inserts & Updates
Query Global-Index
Is RecordKey found in the table?
FALSE
TRUE
Insert new record
Update existing record
Insert new index entry
Index is immutable, so index update is no-op
Done

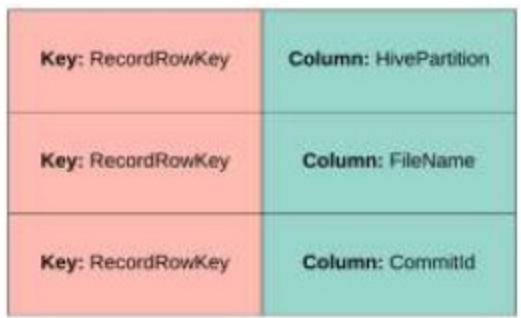

kafka
MySQL
cassandra
UBER
Schemaless
Marmaray
Ingestion Platform
Global
Index
HDFS
Partitions 1..n
Files 1..n
Table 1
Files
1..n
Table 2

Generating Global Indexes

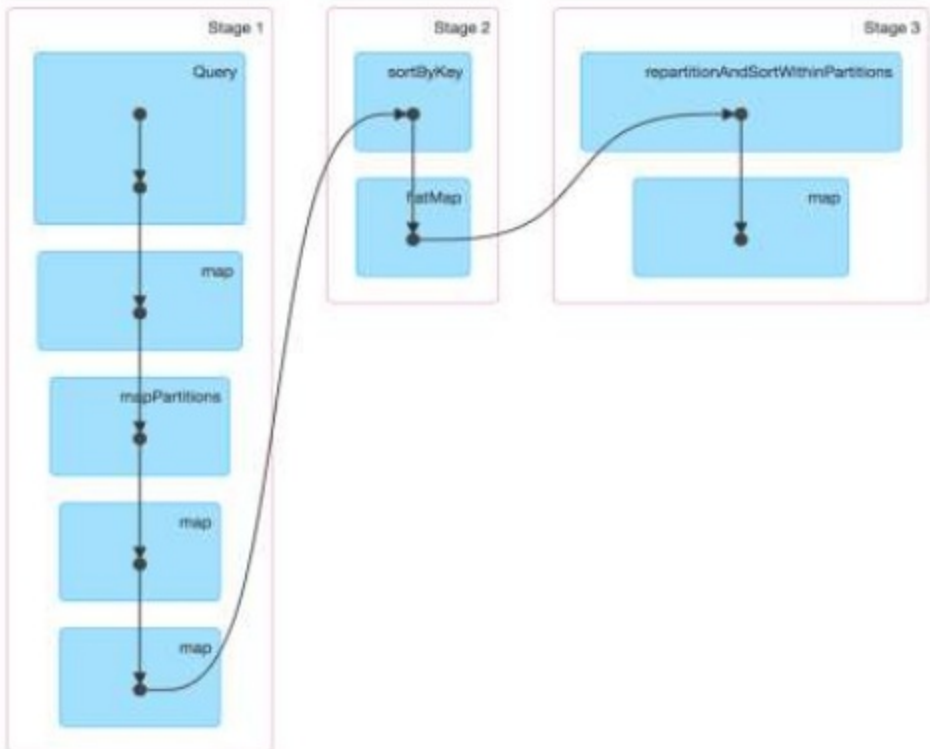
Batch and One Time Index Upload



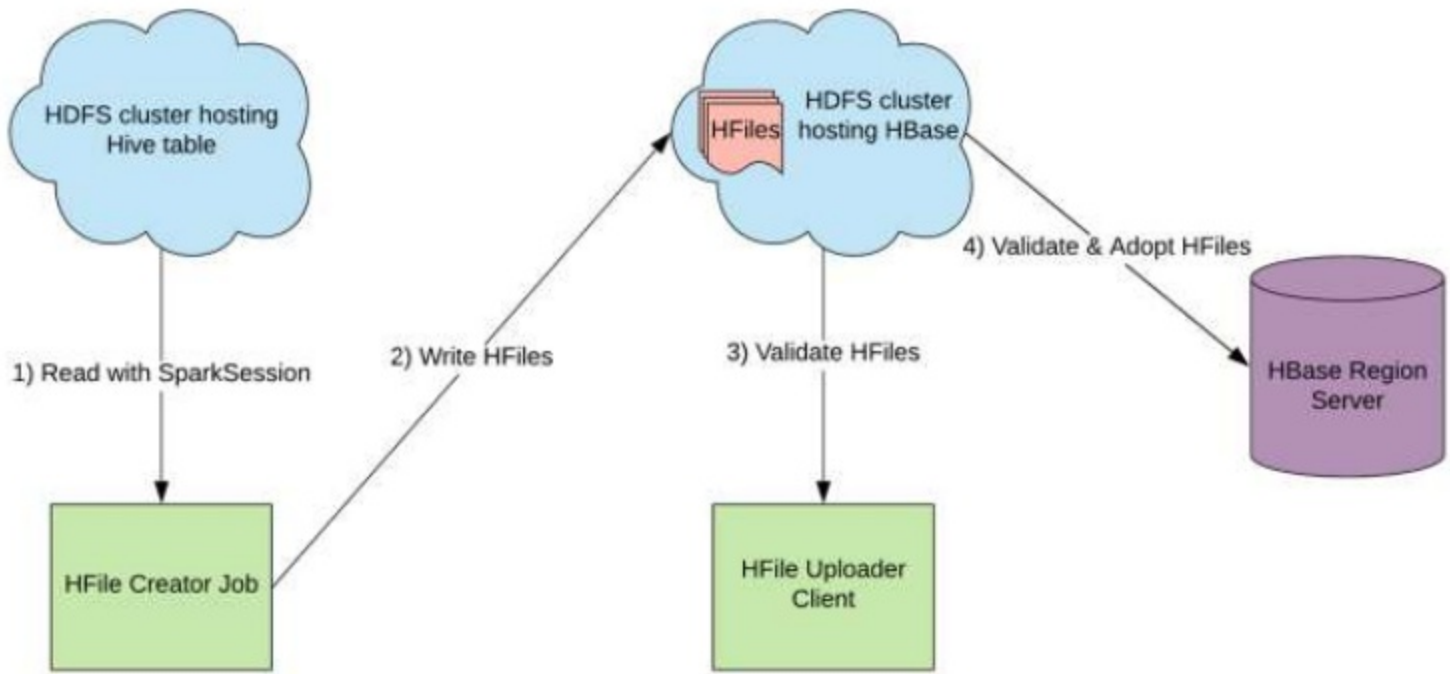
Data Model For Global Index



Spark & RDD Transformations for index generation



HFile Upload Process

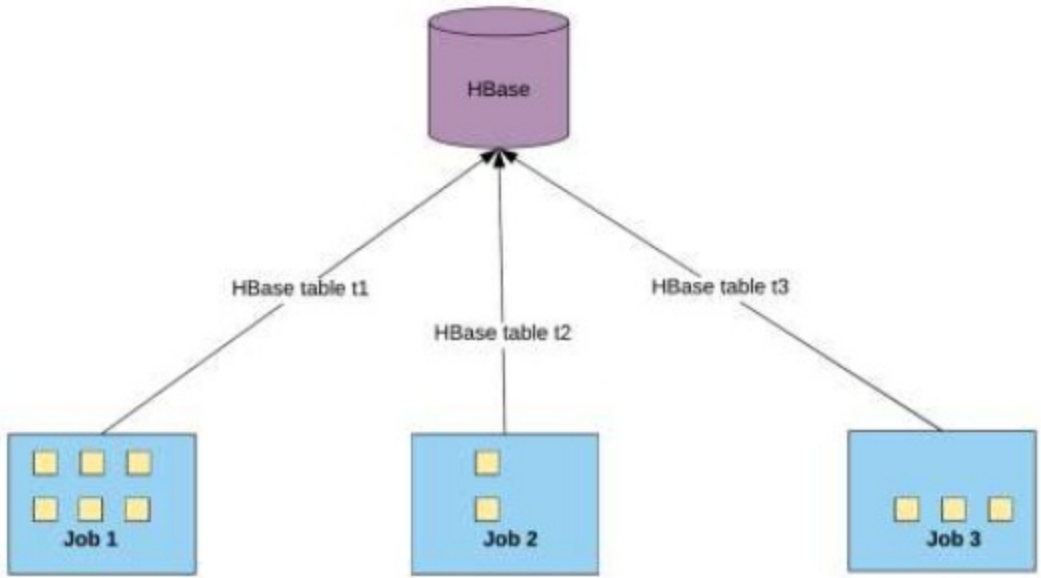


HFile Index Job Tuning

- Explicitly register classes with Kryo Serialization
- Reduce 3 shuffle stages to one
- Proper HFile Size
- Proper Partition Counting Size
- 13 TB index data with 54 billion indexes
    - 2 hours to generate indexes
    - 10 min to load

Throttling HBase Access

The need for throttling HBase Access



Horizontal Scalability & Throttling

Only 1 Region Server

Higher throughput with 5 Region Servers

Next Steps

Next Steps

- Handle non-append-only data during bootstrap
- Explore other indexing solutions

Useful Links

https://github.com/uber/marmaray

https://github.com/uber/hudi

https://eng.uber.com/data-partitioning-global-indexing/

https://eng.uber.com/uber-big-data-platform/

https://eng.uber.com/marmaray-hadoop-ingestion-open-source/

Other Dataworks Summit Talks

Marmaray: Uber's Open-sourced Generic Hadoop Data Ingestion and Dispersal Framework

Wednesday at 11 am

Attribution

Kaushik
Devarajaiah

Nishith
Agarwal

Jing
Li

We are hiring!

Positions available: **Seattle, Palo Alto** & **San Francisco**

email : hadoop-platform-jobs@uber.com

Thank you

Questions: email ospo@uber.com

Follow our Facebook page: www.facebook.com/uberopensource