

# **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.**



Apache Hadoop is either a registered trademark or trademark of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of this mark.

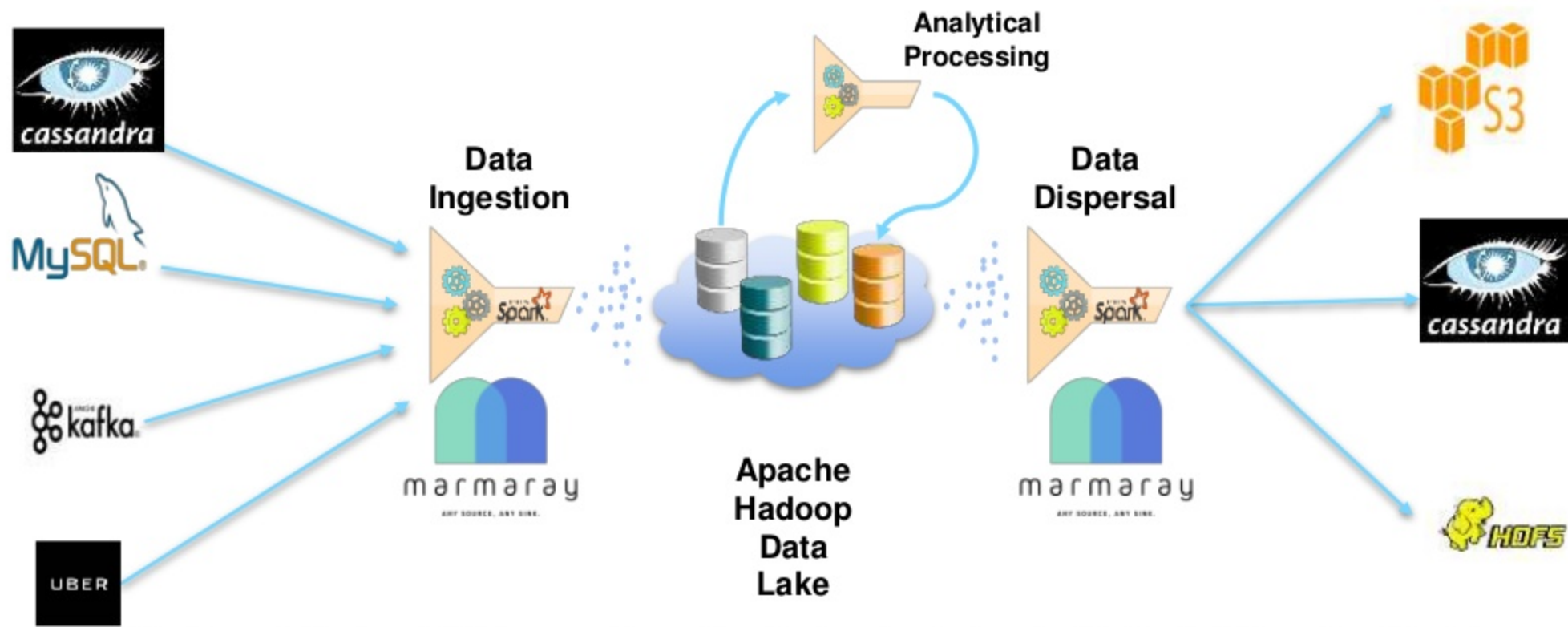
# Overview

- **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

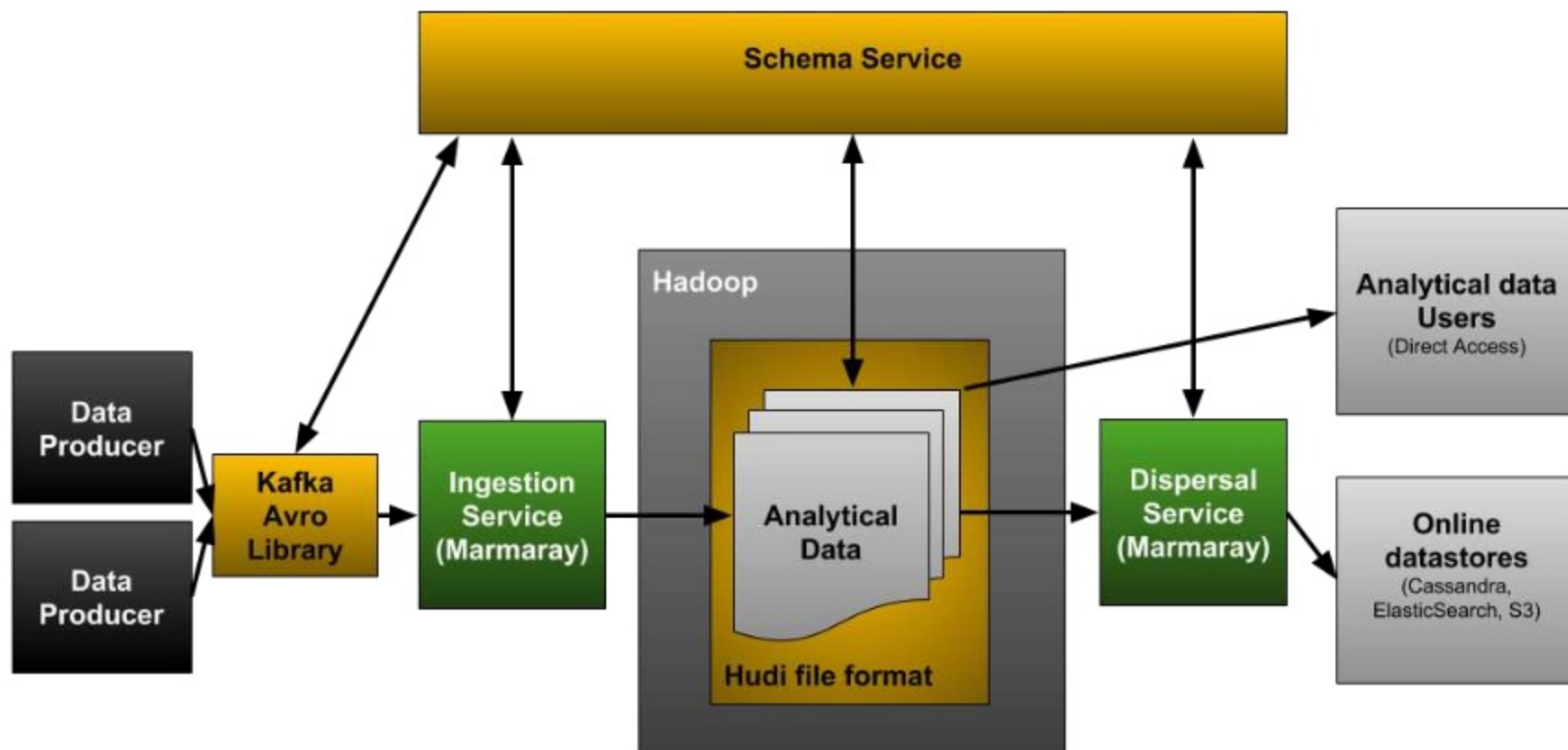
# Hadoop Data Ecosystem at Uber



Apache Kafka, Cassandra, Spark, and HDFS logos are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.



# 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

Changelogs received from upstream datastore:

- 1) (Key:k1, col1:a, col2:b)
- 2) (Key:k2, col1:c, col2:d)
- 3) (Key:k3, col1:e, col2:f)

(batch\_No:1, time:t0)

Changelogs received from upstream datastore:

- 1) (Key:k1, col1:g, col2:h)
- 2) (Key:k4, col1:i, col2:j)

(batch\_No:2, time:t1)

Ingestion Service

Hadoop raw source table:

Key	Col1	Col2	Date_Partition
k1	<del>a</del> g	<del>b</del> h	t0
k2	c	d	t0
k3	e	f	t0
k4	i	j	t1

“Incremental mode” view at time t2 given checkpoint timestamp of t0:

Key	Col1	Col2	Date_Partition
k1	g	h	t0
k4	i	j	t1

“Latest mode” view at time t2:

Key	Col1	Col2	Date_Partition
k1	g	h	t0
k2	c	d	t0
k3	e	f	t0
k4	i	j	t1

# 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





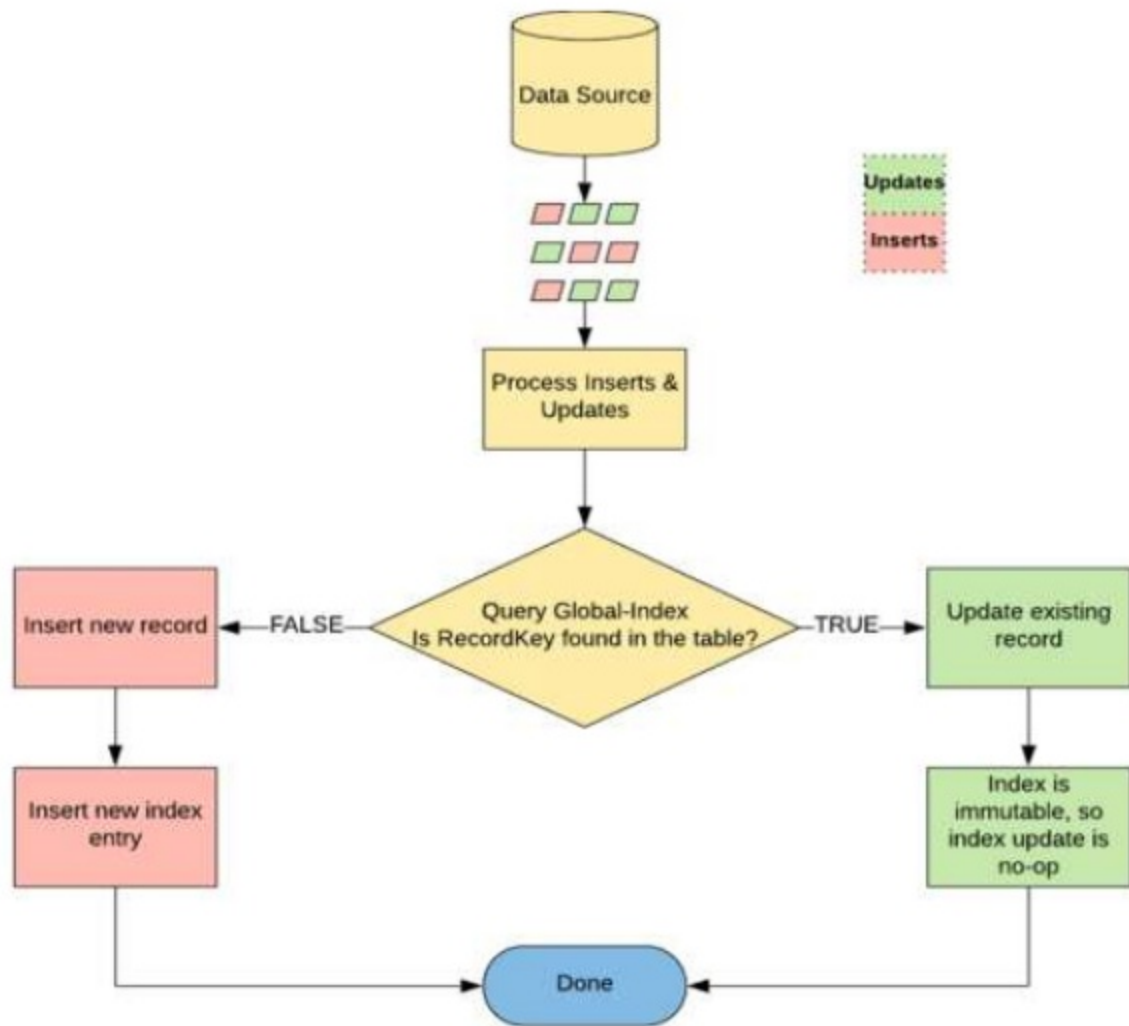
**VS**

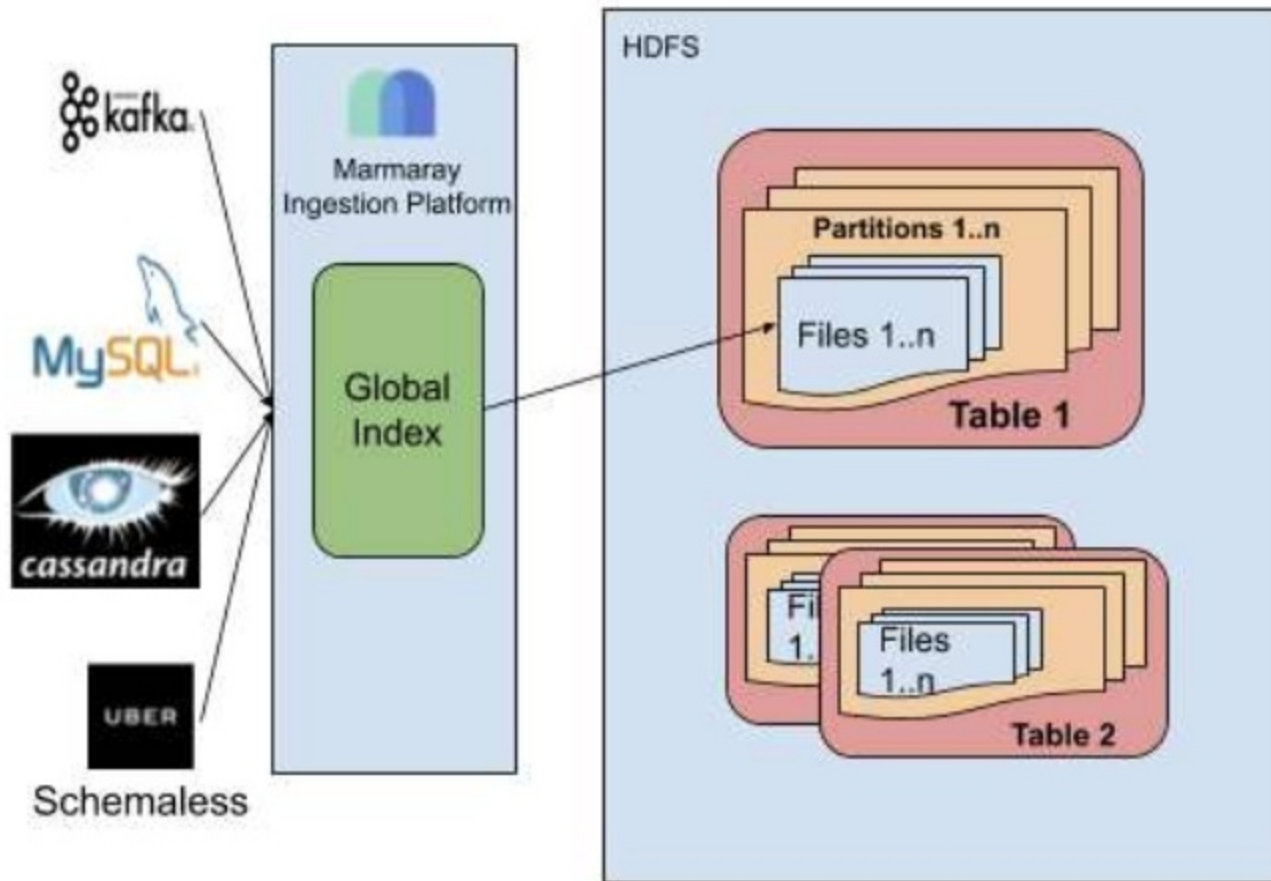


- **Decision was to use HBase**
- **Trade Availability for Consistency**
- **Automatic Rebalancing of HBase tables via region splitting**
- **Global view of dataset via master/slave architecture**

# How Global Index Works

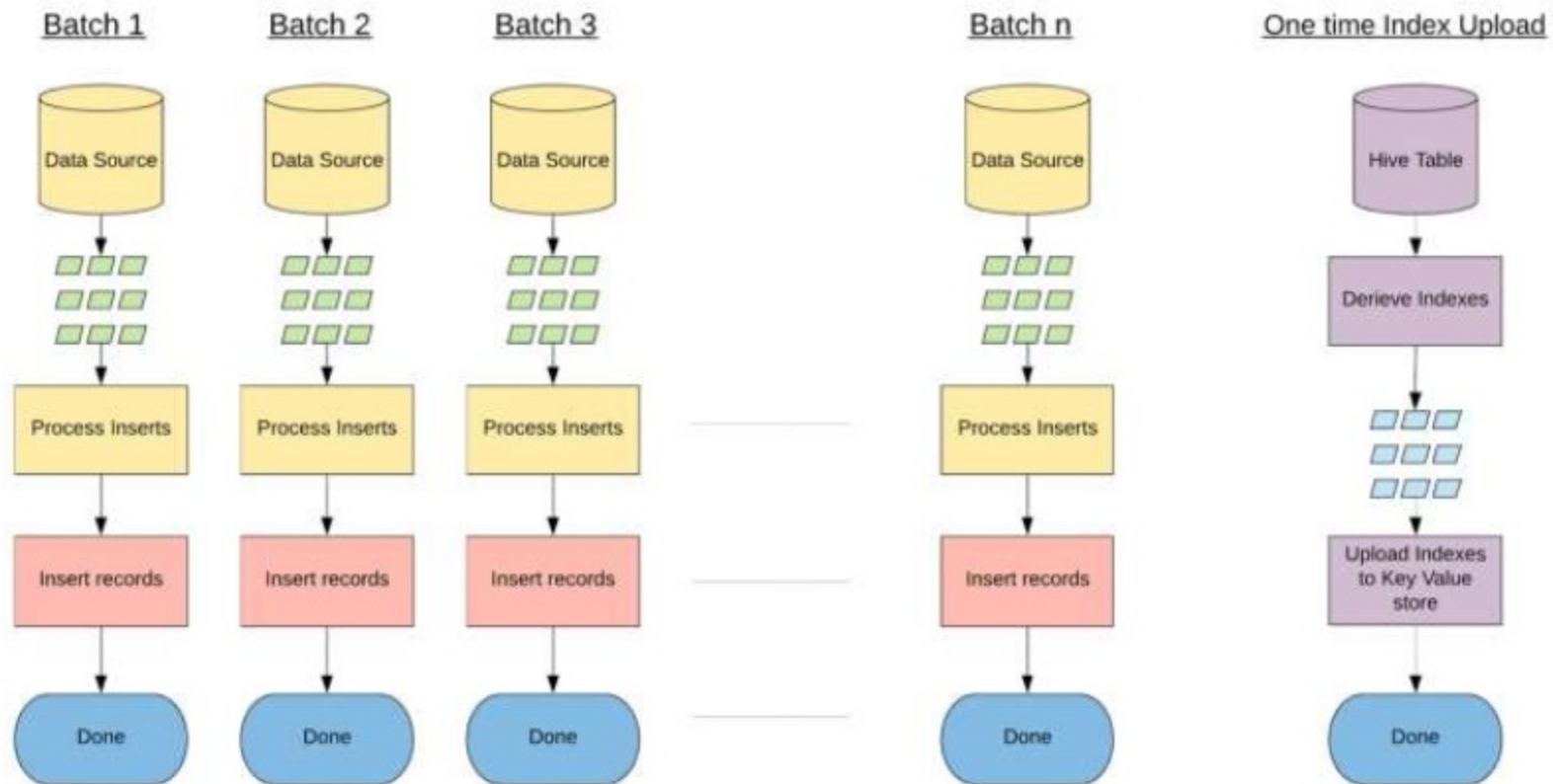






# Generating Global Indexes

# Batch and One Time Index Upload

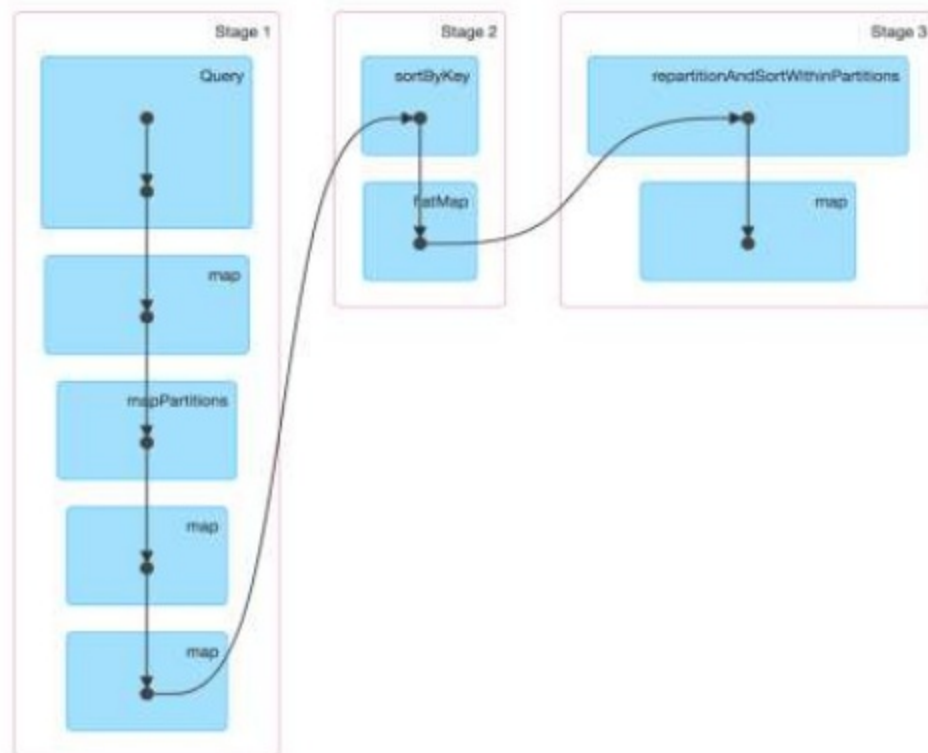


# Data Model For Global Index

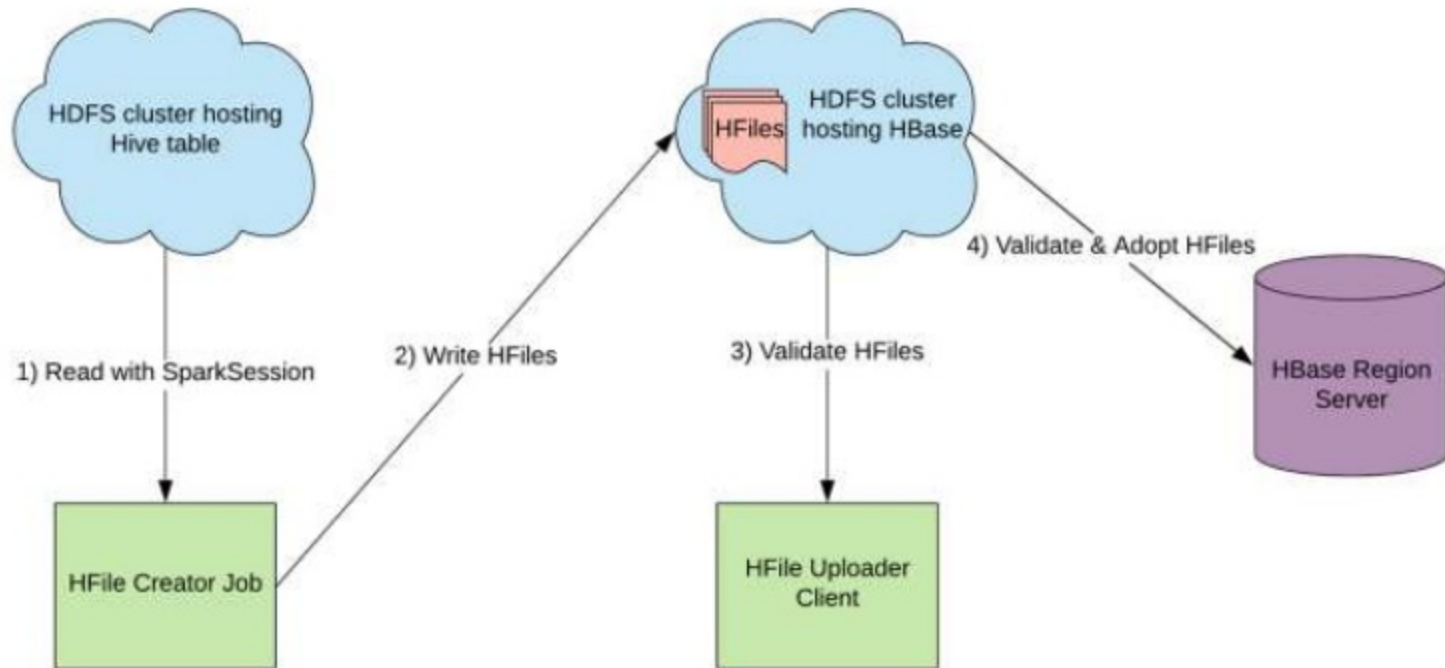
Key: RecordRowKey	ColumnFamily: v	Column: HivePartition	Column: FileName	Column: CommitId
-------------------	-----------------	-----------------------	------------------	------------------

Key: RecordRowKey	Column: HivePartition
Key: RecordRowKey	Column: FileName
Key: RecordRowKey	Column: CommitId

# Spark & RDD Transformations for index generation



# HFile Upload Process





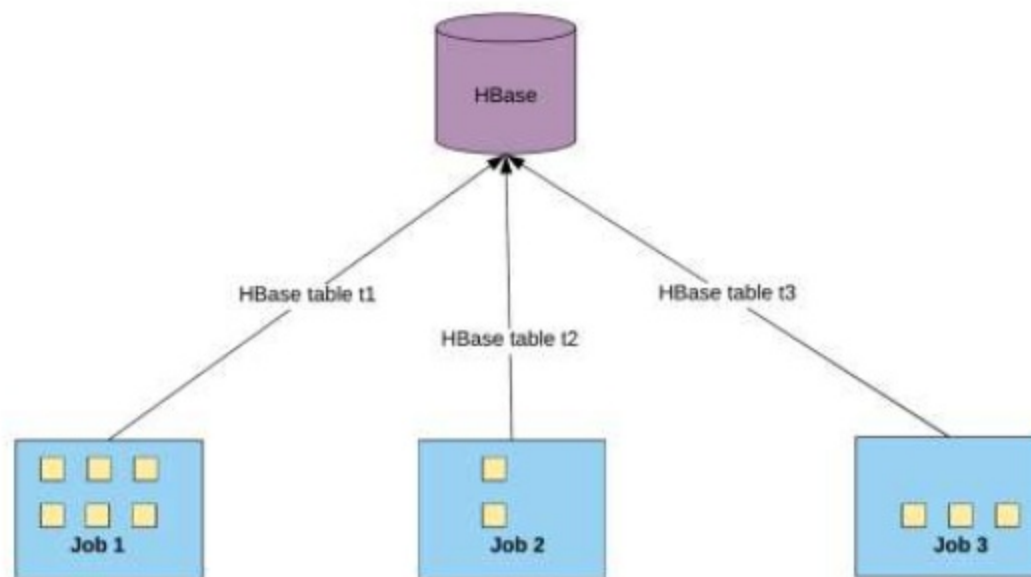
# HFile Index Job Tuning

- 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



# 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/>



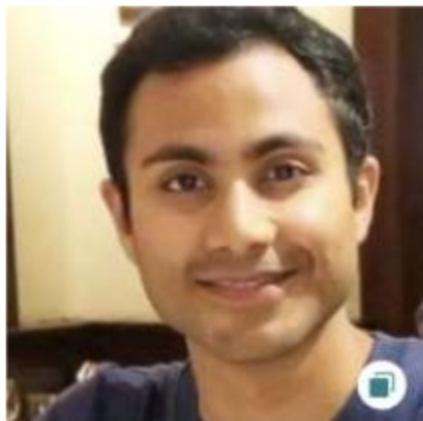
# Other Dataworks Summit Talks



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](mailto:hadoop-platform-jobs@uber.com)



# Thank you

Questions: email [ospo@uber.com](mailto:ospo@uber.com)

Follow our Facebook page: [www.facebook.com/uberopensource](http://www.facebook.com/uberopensource)

Proprietary © 2018 Uber Technologies, Inc. All rights reserved. No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval systems, without permission in writing from Uber. This document is intended only for the use of the individual or entity to whom it is addressed. All recipients of this document are notified that the information contained herein includes proprietary information of Uber, and recipient may not make use of, disseminate, or in any way disclose this document or any of the enclosed information to any person other than employees of addressee to the extent necessary for consultations with authorized personnel of Uber.

