Data Ingestion and Replication @ Twitter

By Praveen Killamsetti and Zhenzhao Wang

BEAM SUMMIT

Austin, 2022

Agenda

- Introduction
- Replication and Ingestion Architecture
- Batch Data Replication
- Real Time Log Ingestion

Data Lifecycle Team

- Ingestion
  - Offline ingestion to data lakes (HDFS, GCS)
  - Near Real time ingestion to Kafka, Pubsub and BigQuery
- Replication
  - Replication of data between data lakes (HDFS, GCS)
  - Replication of data between HDFS/GCS and BigQuery
  - Replicating data between KV Store and BigQuery
- Metadata Management
  - Data discovery, segment metadata
- Storage and Retention
  - HDFS, GCS
  - BigQuery (Retention only)

Data Ingestion / Replication Challenges

Data scrubbing breaks WORM data model

GDPR Compliance:

- Account deletion must remove all personal information
- User data deletion/modification must remove specified fields or rows associated with a user

Need for Real Time ingestion
Existing pipelines takes hours to ingest data to BigQuery

Engineering Velocity
Building and managing replication pipelines consumes lot of engineering time

Consolidate Replication/Retention Services
8+ different replication mechanisms based on source and destination combination

Data Ingestion / Replication Goals

Dimensions

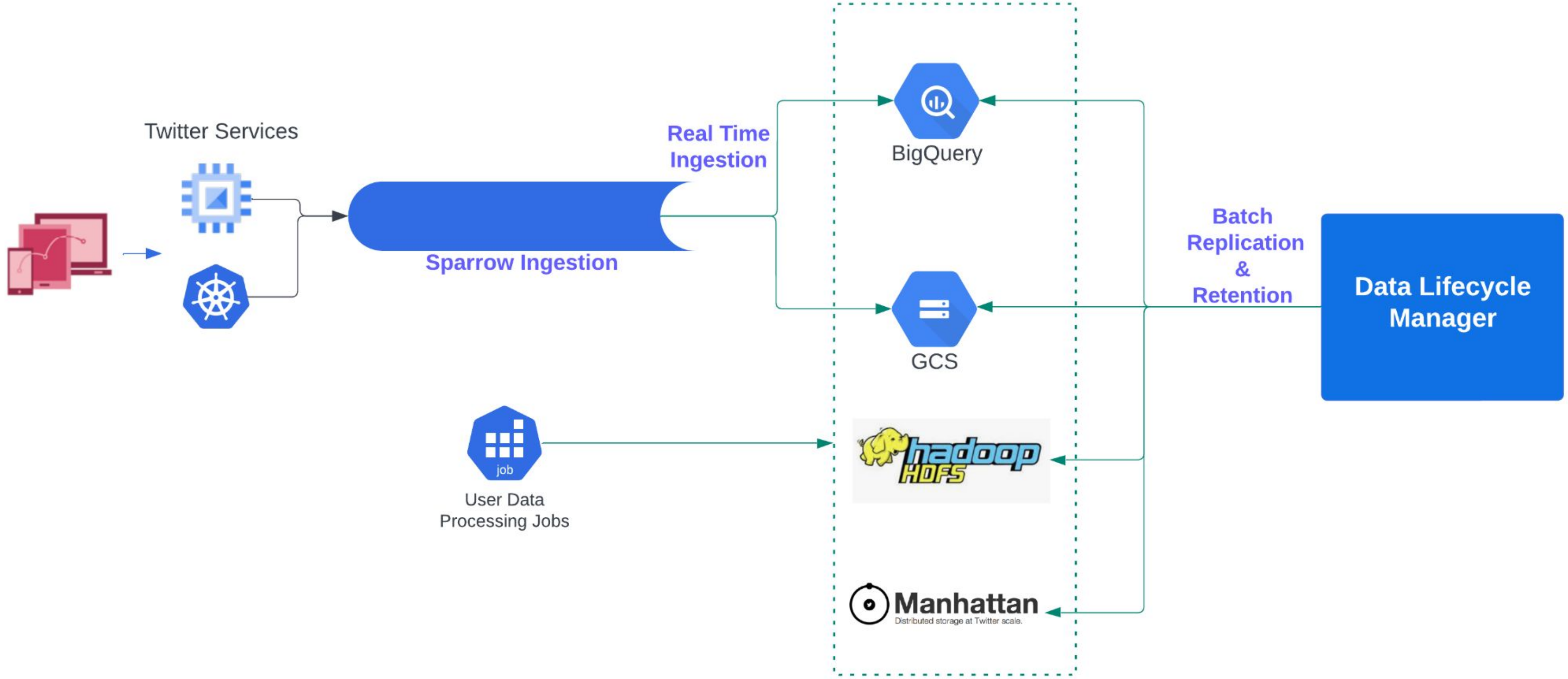
Simple to Use
Unified(Batch and Streaming) way to configure replication/ingestion across all analytics stores within few mins

Platform Offering
Managed Ingestion/Replication offering with production SLOs and multiple tiers of QoS. Automatic schema update handling

Platform Integration
Metadata driven replication, Scrubbing Aware, Authorization, Authentication, Chargeback, Unified Monitoring

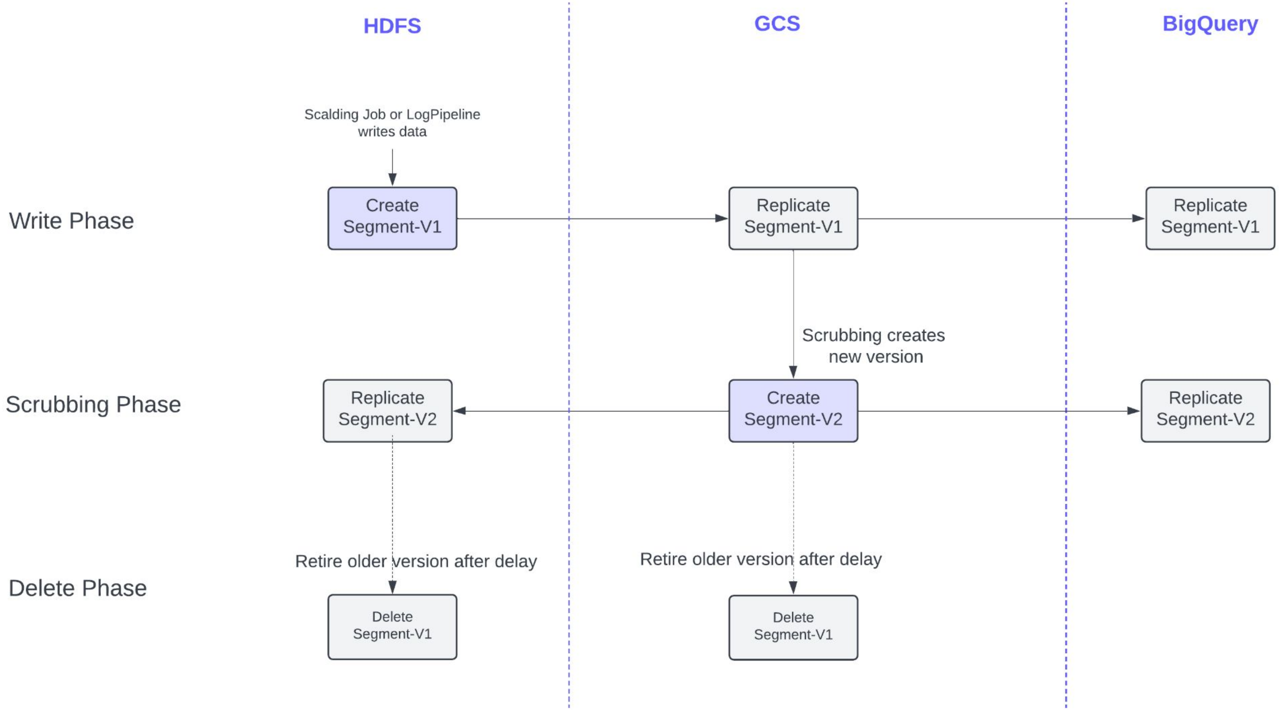
Extendable
Ability to add new storage systems support easily

Ingestion, Replication & Retention



Data Lifecycle Manager (DLM)

Data Scrubbing -> No more WORM



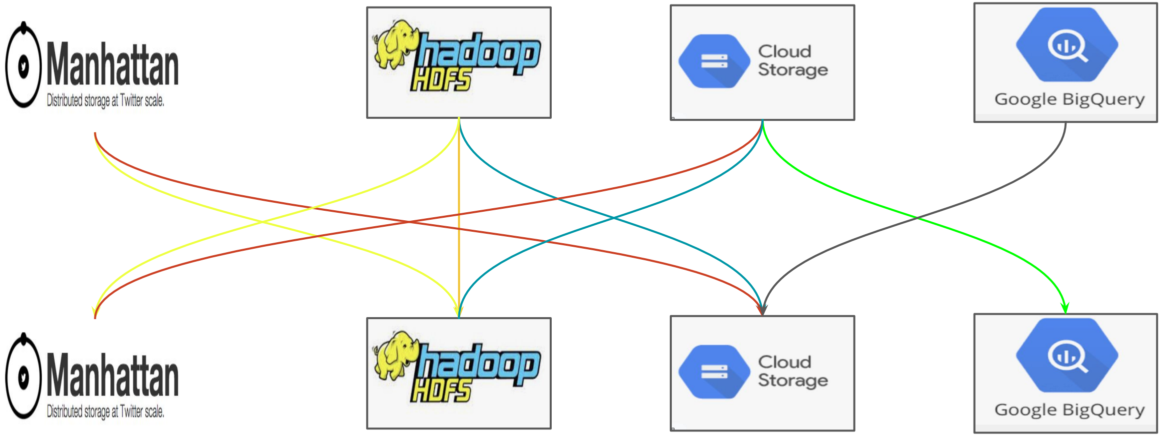
Need for Metadata Driven Data Lifecycle

- Traditional Batch Polling won’t work and not scalable either
- Similarly TTL based retention also does not work
- Versioning
  - Unit of metadata: **Segment** (Chunk of a dataset that maps to a time interval or version)
  - Increment version on segment rewrite/scrub/creation
  - Listen changes to metadata layer and trigger replication or retention
  - Change replication to version based replication with goal to keep replicate latest version across storage systems
  - Change retention architecture to do version based deletion along TTL

Opportunity to Simplify

Sources

Destinations



Cloud Replicator

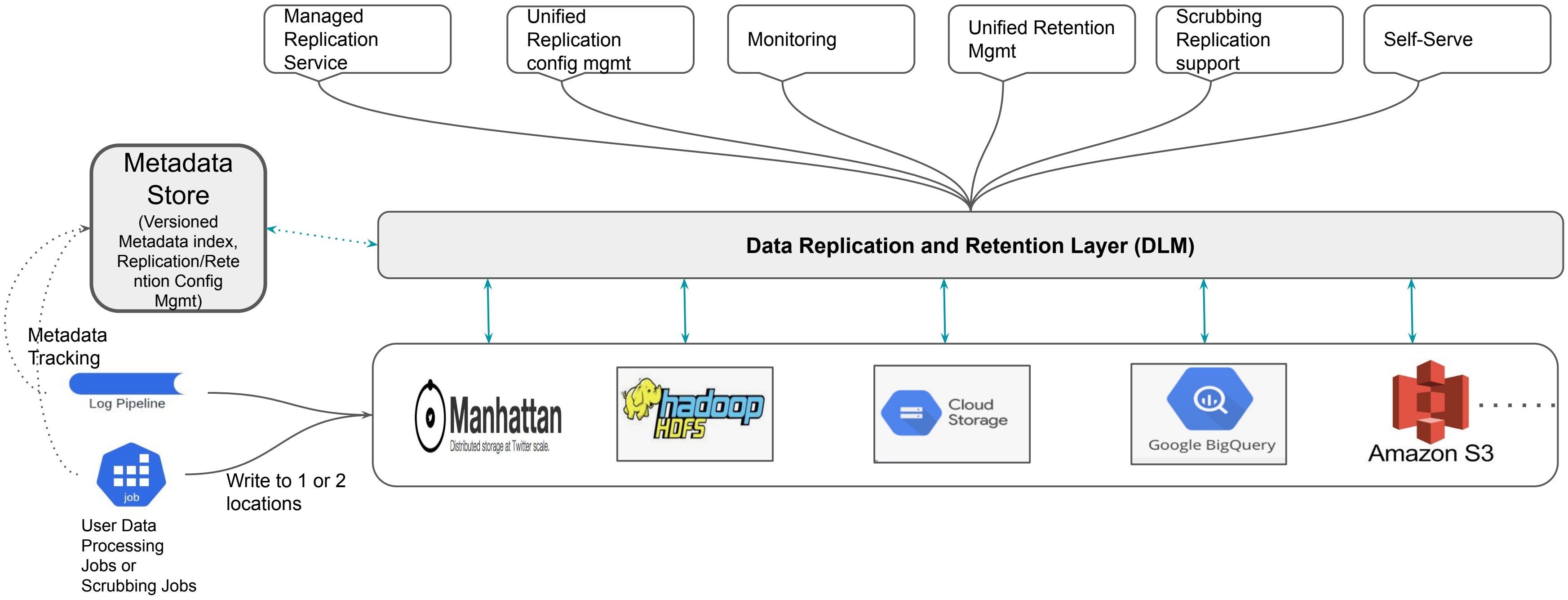
Roneo bird

Loris, BQ Blaster, BID

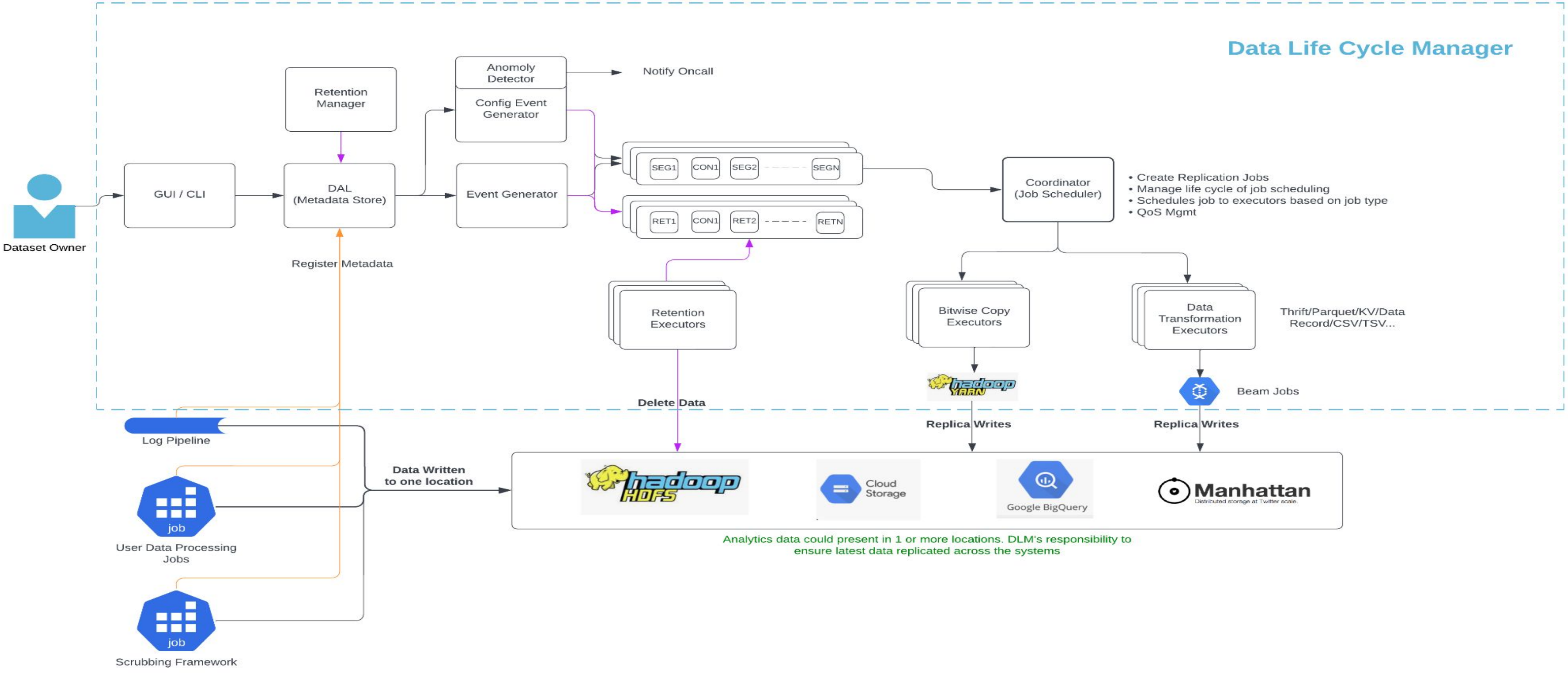
MH Import, Bifrost

Unsupported

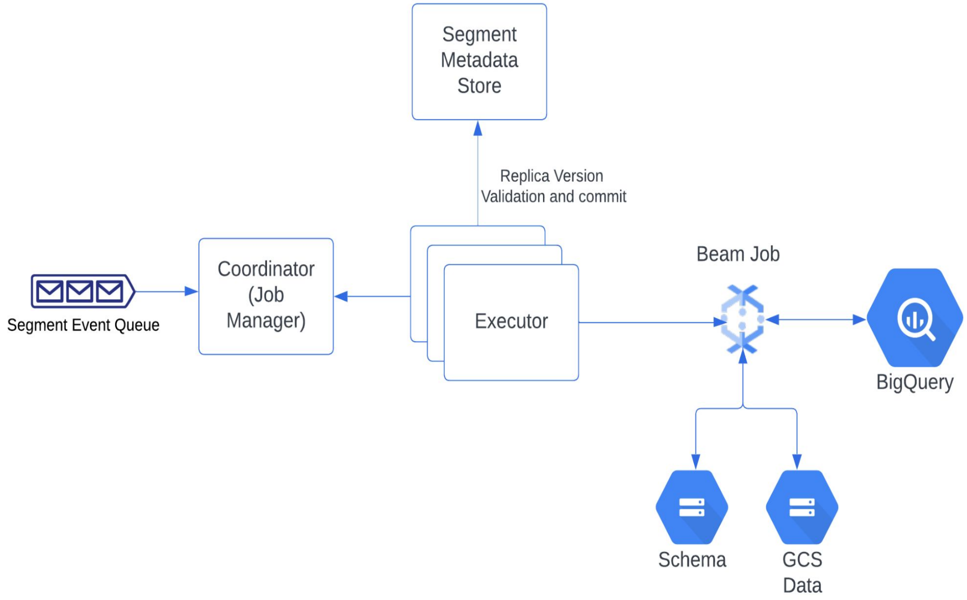
Batch Replication Vision



DLM Architecture



Beam Replicator



Dynamically deploy Beam Job

- Validate replica version
- Copy Replica
  - Load schema
  - Deserialize thrift data
  - Apply UDF (wip)
  - Convert to Avro
  - Write to BigQuery
    - BigQueryIO
    - BigQuery Load
- Commit replica Version

Data Formats

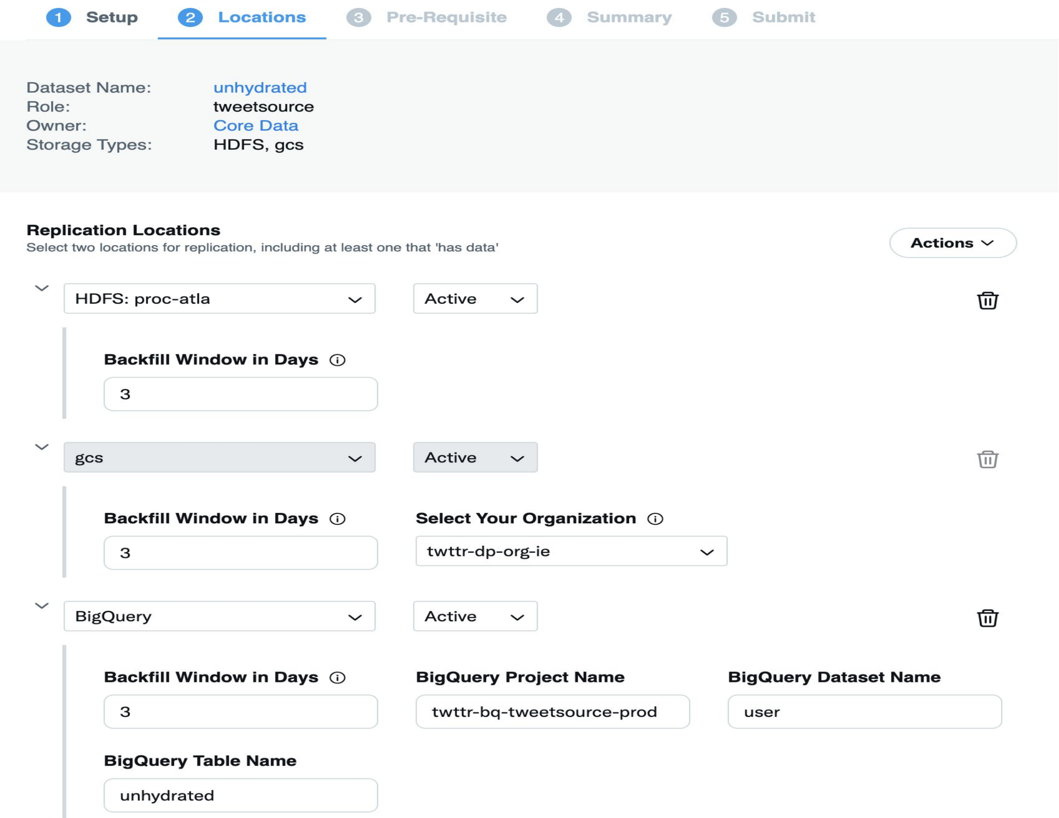
- Supported Data Formats
    - Thrift LZO, Thrift Parquet, Thrift Data Record, Key-Value data, CSV/TSV
- KV data
    - Annotation based deserialization spec embedded in schema

```
struct ManhattanDatasetPkey {
  1: required i64 userId;
}(persisted = "true", hasPersonalData = "true")

struct ManhattanDatasetValue {
  1: required embedding.TopSimClustersWithScore topSimClustersWithScore;
}(persisted = "true", hasPersonalData = "true")

struct ManhattanDatasetSchema {
  1: required ManhattanDatasetPkey manhattanPkey(MHPkeyCodec = "Injection.long2BigEndian");
  3: required ManhattanDatasetValue manhattanValue(MHValueCodec = "T_COMPACT");
}(persisted = "true", hasPersonalData = "true")
```

Self-Serve



1 Setup
2 Locations
3 Pre-Requisite
4 Summary
5 Submit
Dataset Name: unhydrated
Role: tweetsource
Owner: Core Data
Storage Types: HDFS, gcs
Replication Locations
Select two locations for replication, including at least one that 'has data'
Actions
HDFS: proc-atla
Active
Backfill Window in Days
3
gcs
Active
Backfill Window in Days
3
Select Your Organization
twttr-dp-org-ie
BigQuery
Active
Backfill Window in Days
3
BigQuery Project Name
twttr-bq-tweetsource-prod
BigQuery Dataset Name
user
BigQuery Table Name
unhydrated

Adoption

Data pipelines	**1200+**
Data Volume processed per day	**4PB+**
Number of Teams/Projects	**158**
Records Processed Per day	**4Tri+**
Jobs Per day	**22k+**

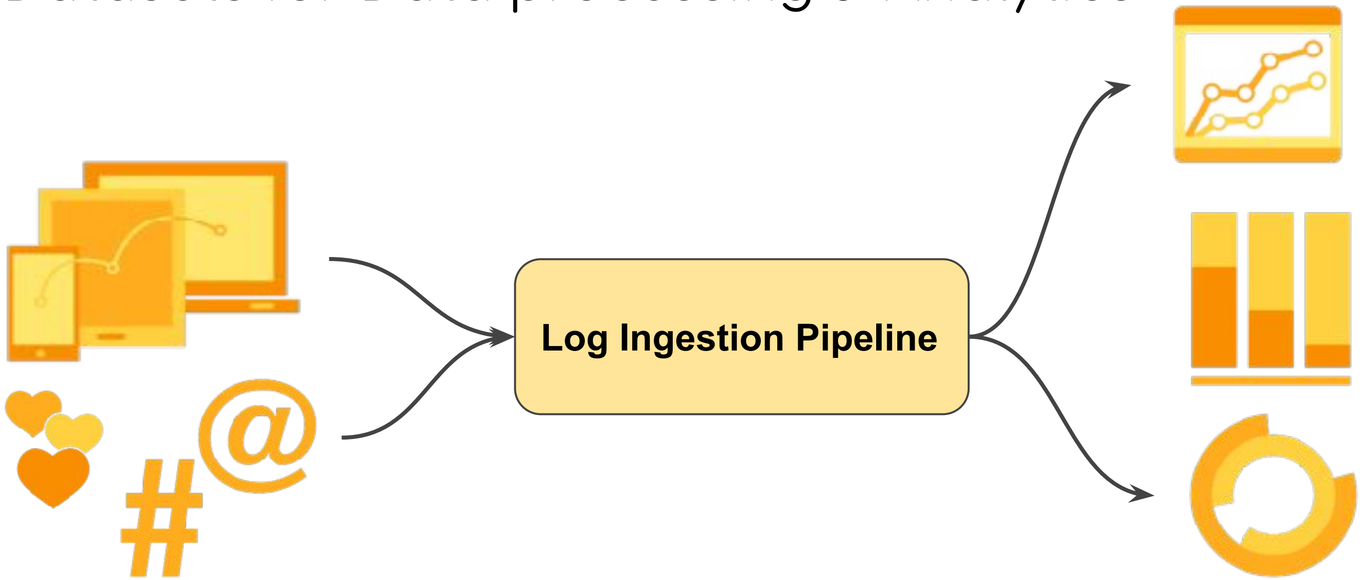
What's Next

- UDF support with SQL like expression for simple filtering
- Migrate existing pipelines to DLM
  - 600+ jobs maintained by 60+ teams
  - Migrate from older services and deprecate them
- Simplify replication between Manhattan KV store and BigQuery
- Cost/Perf Improvements
  - Migrating reflection based schema library to AST based schema library
  - Improve dataflow jobs to read from HDFS and write to BigQuery

Log Ingestion: Sparrow

What is Log Ingestion?

- User interactions & Internal service generate events
  - E.g. ads click, KV store write info
- Events are grouped as Datasets
- Datasets for Data processing & Analytics



History & Challenges at Twitter

- **Scalability**
  - 3~5 billion events per minute, 10~22 TB traffic per minute
  - Huge datasets could have 10~18 GB throughput and 35~43 millions per second
  - >1 million internal clients publishing data to the log ingestion systems

- **Historical Batch Solution**
  - It takes hours to deliver data to the user specified destination
  - Major components is on-premise, no support for data generated on cloud
  - Build on top of old tech such as HDFS, Tez, Mesos

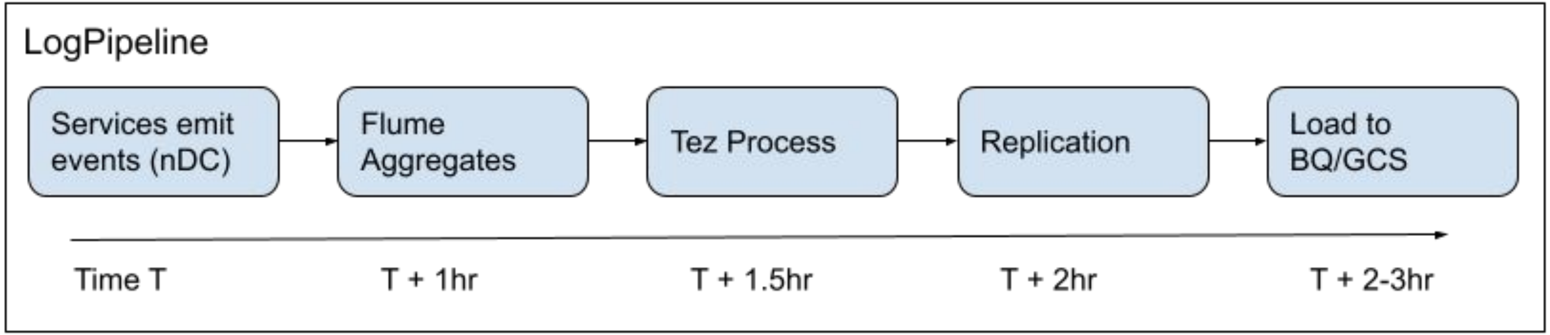
Evolve Log Pipeline - Goals

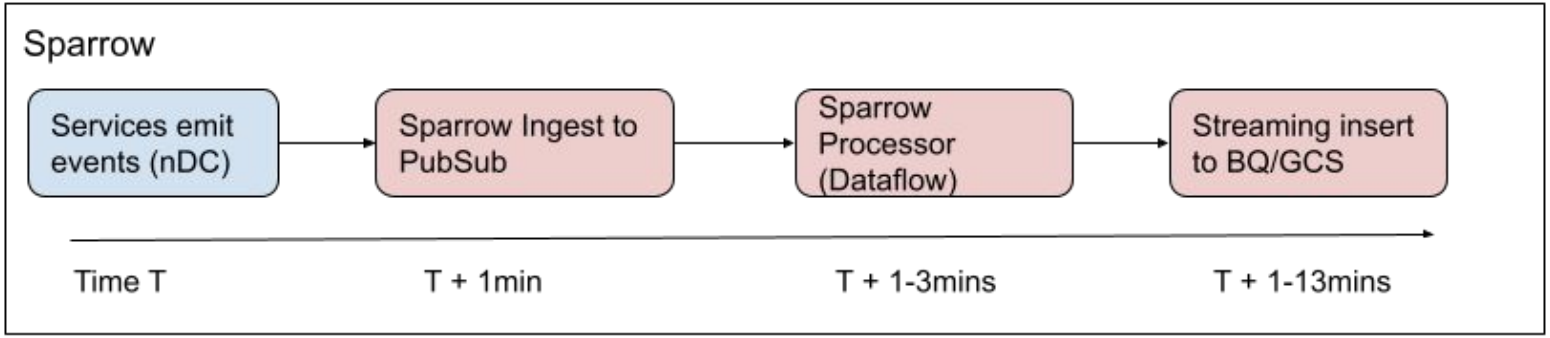
- ➔ **Scalability**
  - ◆ 3~5 billion events per minute with 50% traffic YoY growth target
  - ◆ 10~12GB/s in the largest dataset also with 50% traffic YoY growth traget
- ➔ **Streaming ability**
  - ◆ Provide streaming ability to deliver data in near real time
- ➔ **Cloud native**
  - ◆ Running on cloud environment
  - ◆ Adopt the cloud technologies
- ➔ **Compatibility with existing pipeline and Migration Friendness**
  - ◆ Produce compatibility - (near) transparent migration for existing consumers
  - ◆ Consumer compatibility - Scaldings, MR, etc
  - ◆ Data management compatibility - Data protocols/Layout
- ➔ **User Defined Function Support**
  - ◆ Compatible with on-prem UDF.
  - ◆ Empowers user to do light transformation light ETL serverlessly

What is Sparrow?

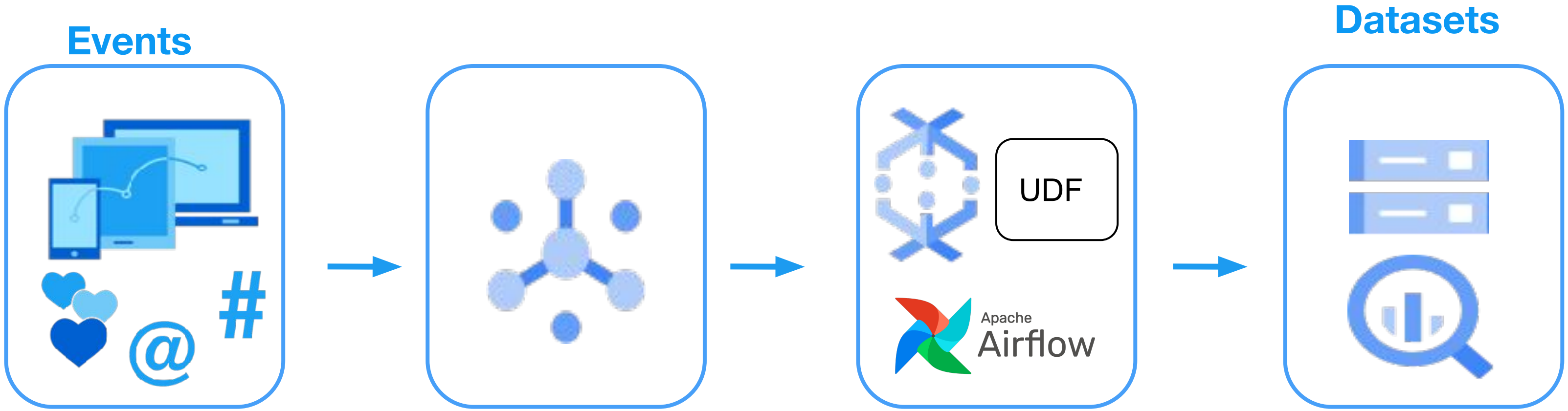
- **Sparrow: enable real-time analytics**
  - Ingest on-prem/GCP data to Pubsub/BigQuery/GCS in realtime.
  - Managed solution - no maintenance needed from users.
  - Cloud native
  - Transparent migration for existing customers
  - Transformation supported before ingestion via UDF(user defined function)
  - PDP(Private data protection) Compliance
  - Chargeback support, cost estimator etc

Sparrow & Historical Pipeline



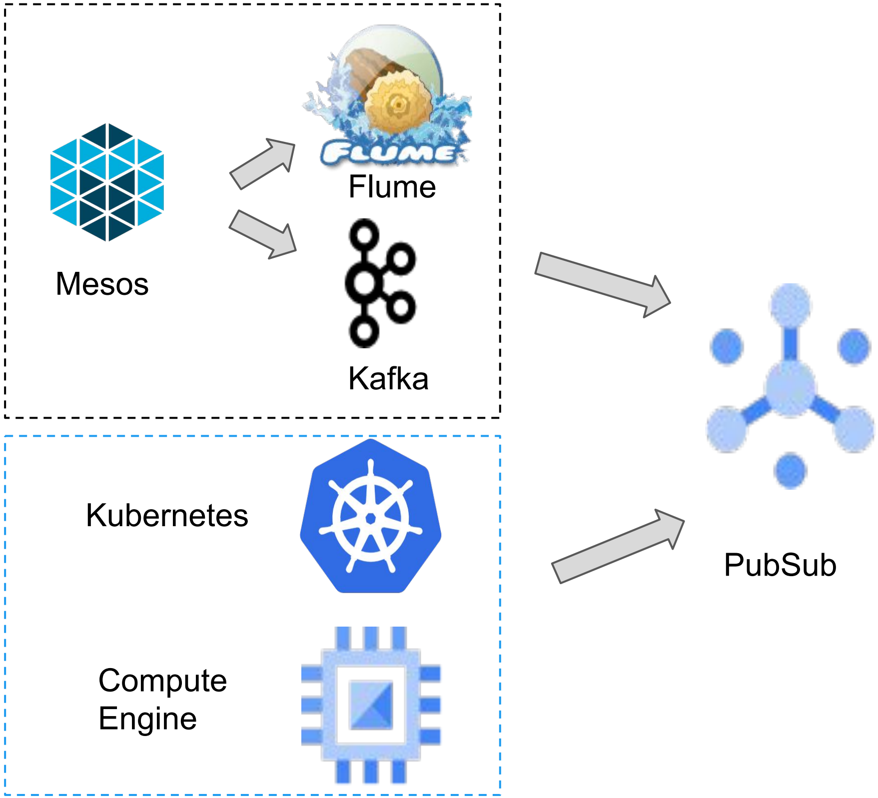


Architecture



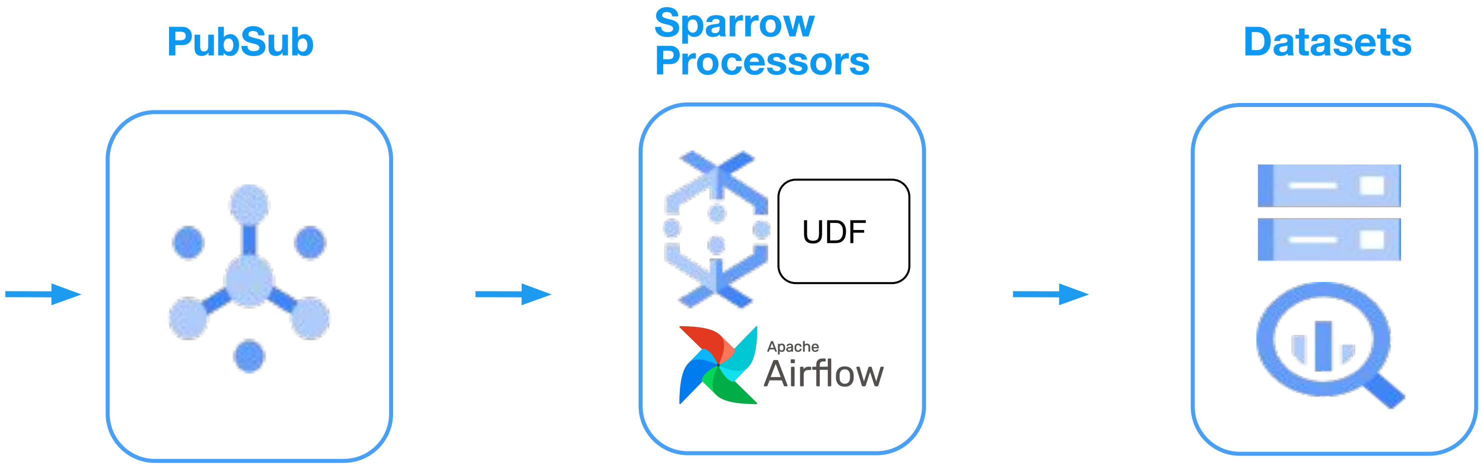
Clients	Streaming Aggregation	Streaming Processors	Real time insertion
K8S, Mesos, Compute engine	Flume, Kafka, Google PubSub	Apache Beam, Google Dataflow	BigQuery, GCS

Sparrow Ingestion & Aggregation



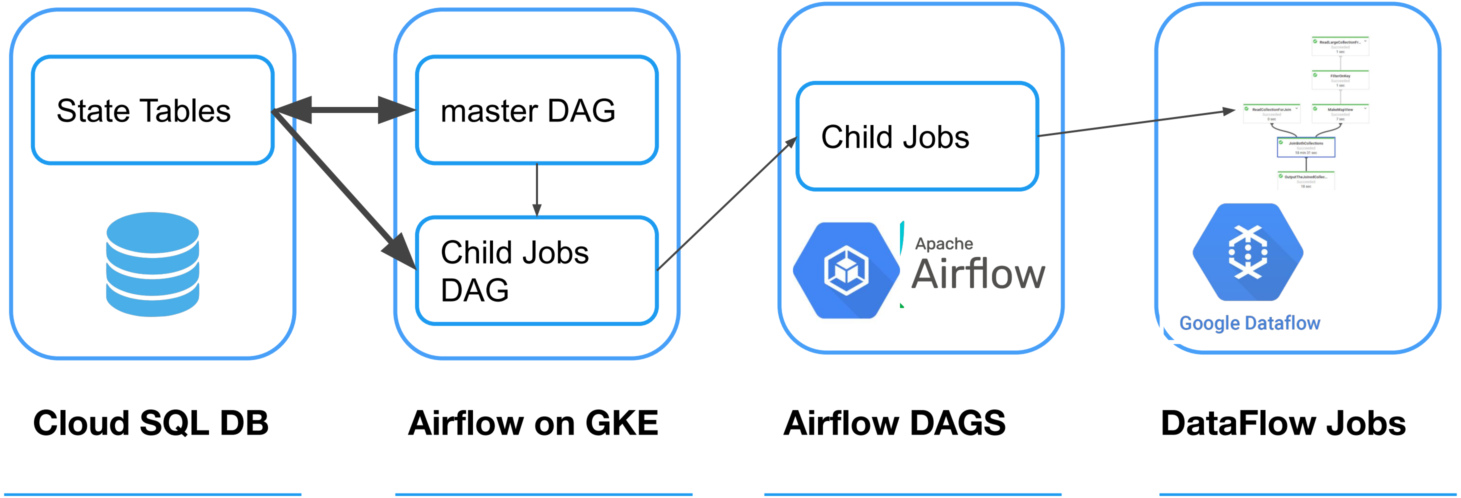
- Transparent support to existing on-prem traffic
- Unified client lib to provide consist API independent of environment.
- A metadata management system to make pubsub pluggable
- On the wire data compression before publish to PubSub

Sparrow Processors



- Metadata management
- One beam job+subscription per dataset for transformation
- User Defined Function Support
- Schema conversion & dynamic load schema for transformation
- Airflow based orchestration
- Up to 20GB/s single beam jobs

Sparrow Processors & Orchestration



Sparrow UDF(User defined Function)

Why?

- Users want to do light transformation before ingestion. E.g. filtering, enrich fields
- Writing a dataflow job/MR job is complex for simple ETL
- Maintain the ETL job is tedious and might be time consuming.

Goal

- Provide function based service and help user to focus on their core logic without worrying about schema transformation quota, PDP compliance, etc
- Provide a managed solution to make it maintenance free for users.
- SQL support

Sparrow UDF (User defined function)

- How to use it?
  - Implement in the interface and check in source.
- How it works?
  - Serverless to users
  - Managed solution
  - Every record will be feeded once
  - Zero or multiple output records support
- How to update user defined function?
  - As same as normal review process
  - Automatic update via version.

```
@LPContext(Dataset="DmEvents")
class AppEventToDirectMessageEventUf
  extends TBaseRecordUserFunction
  <AppEvent, DmEvent> {
  @Override
  public Record<DmEvent> processRecord(
     AppEvent appEvent) {
    if (appEvent.type != DirectMessage) {
      return Record.empty().
    }
    DmEvent event = extractDmField(appEvent);
    return enrichEvent(event);
  }
}
```

Beam Job Optimization

- Decreased the beam job resource by 80%~86% via removing shuffle in BigQuery IO connector
    - Collaborate with dataflow eng team and remove shuffle before ingestion
- Data compression on PubSub before processed by beam job
    - Reduce beam worker usage by ~20%
- Optimize schema conversion logic
    - Improve thrift=>avro=>TableRow schema conversion logic with nested schema

Future work for Log Ingestion

- Continuous Job optimization
  - Schema conversion optimization
- Performance enhancement
  - Long tail problem fix.
  - Better compression
- UDF enhancement
- User experience enhancement
  - UI support and improvement
  - Percentile metrics
- More destinations support
  - Druid, BigTable, etc

Recap

- Log Ingestion to start the data analytics journey
- Data replication bridges different analytics systems and online server databases.
- Beam’s Unified model helped us solve both streaming and Batch needs
- Managed solution to make it effortless for users & UDF to make easy to do customized ETL
- Metadata management system to take care of metadata management & data discovery

Questions?

https://www.linkedin.com/in/praveenklm/
https://www.linkedin.com/in/w-zhen-2a915062

Twitter Career
Twitter Engineer Blog
Twitter Open Source

Analytics Data @ Twitter

KV Store

Manhattan
Distributed storage at Twitter scale.

4k+ import / exports

Data Lakes

hadoop HDFS

Cloud Storage

14k+ datasets
50k+ jobs
200PB+ data

Data Warehouse

Google BigQuery

25k+ datasets
40k jobs/day
149PB+ data

Ingestion + Replication Volume	: 100PB+ day
Events Processed	: 7+ Trillion

Serves all

Data Lifecycle Team

- Ingestion
  - Offline ingestion to data lakes (HDFS, GCS)
  - Near Real time ingestion to Kafka, Pubsub and BigQuery
- Replication
  - Replication of data between data lakes (HDFS, GCS)
  - Ingestion of batch data from HDFS/GCS to BigQuery
  - Replicating data between KV Store and BigQuery
- Metadata Management
  - Data discovery, segment metadata
- Storage and Retention
  - HDFS, GCS
  - BigQuery (Retention only)

Data Volume processed	**100+PB**
Data across storage systems	**1+EB**
Events processed	**7+Tri**