APACHECON

Large Scale Migration to Apache Parquet in Uber

Huicheng Song, Xinli Shang

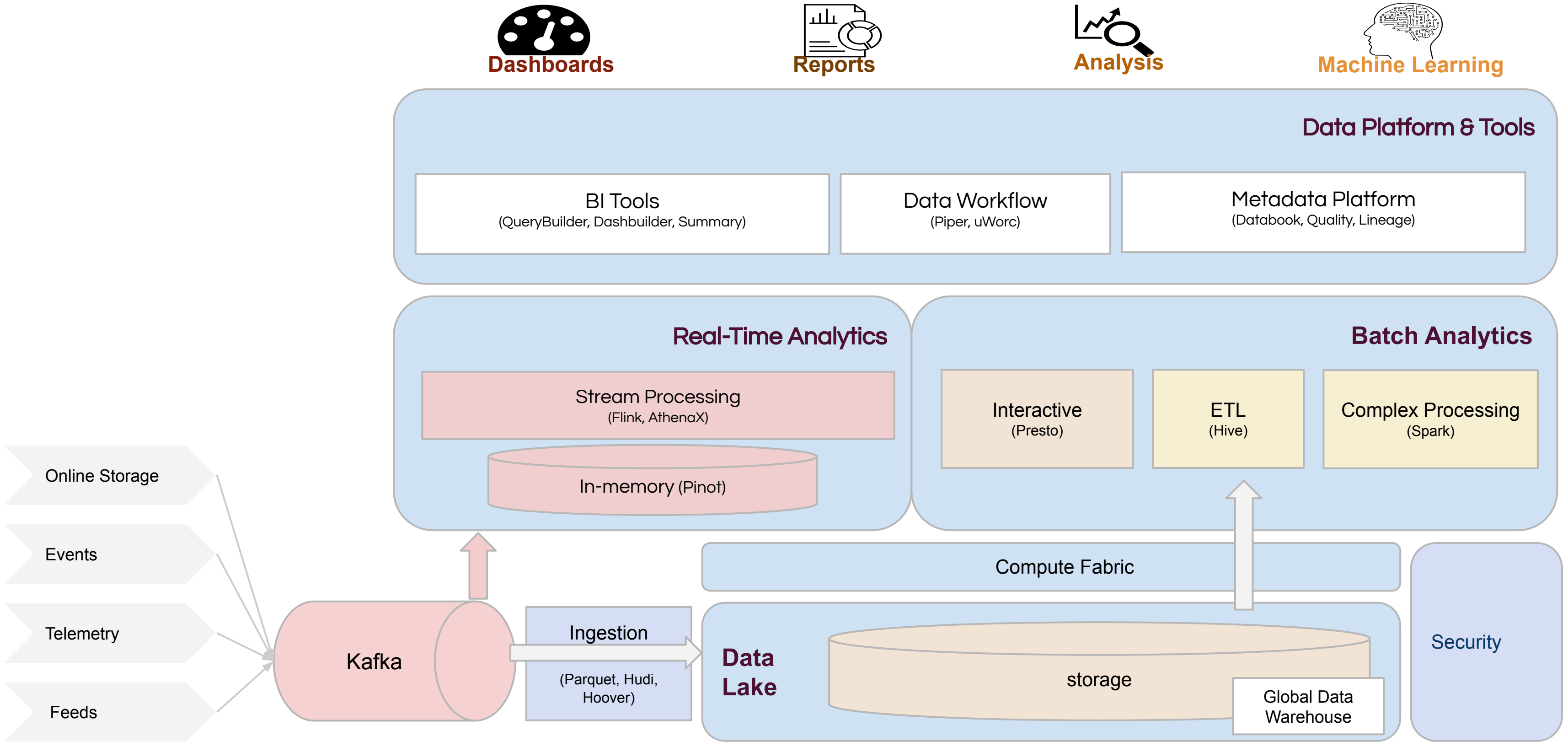
Speaker Intro

- Huicheng Song
  - Staff Software Engineer
- Xinli Shang
  - Apache Parquet PMC Chair
  - Manager at Uber Data

Agenda

- Uber data architecture
- Apache Parquet @ Uber
- Unification of big data file format
- Challenges and solutions
- Current status and futures

Uber Data Architecture



Apache Parquet Intro

- Widely used Big Data File Format
- Designed for efficiency, security & interoperability

Apache Parquet @ Uber Data Lake

- Most data is Parquet format while a small portion of ORC, Text…
- Security initiatives
  - Column level encryption
  - Cell level encryption
- Efficiency
  - Migration to ZSTD
  - Column Pruning to save storage

Challenges to solve

- Data safety: data + metadata
- Scale: 20% of 100k tables
- Zero downtime
- ETL pipeline diverse DDL/DML/replication

Migration story V1

- High-level API based
  - Data / metadata loss
  - Hard to fix data issues: e.g. null map keys
  - ETL job issues: slow / OOM
- Pre-gen ETL pipelines + "crowdsourcing"
  - Data team manage infra; owners manage pipelines
  - Low partner engagement
  - Pipeline errors after migration, e.g. on swap DML

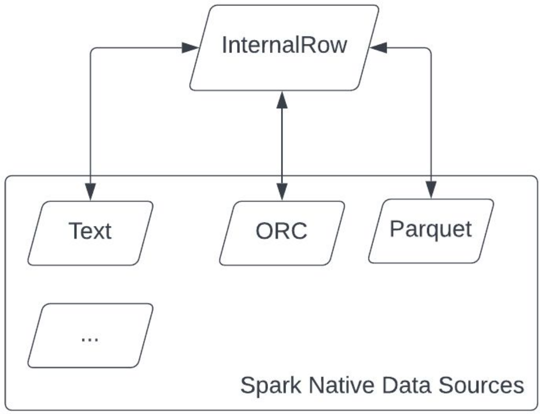
```
create table x_staging like x

insert into x_staging partition(datestr)
select * from x

alter table x rename as x_bak
alter table x_staging as x
```

Migration story V2

- File based rewriter
- Migration job management
- Mix format support
- Data driven ETL pipeline migration

File based rewriter

- Problems to solve
  - Safety / scale
- Rewriter based on Spark data source
- Rewrite => validate => swap
- Problems solved
  - Legacy ORC Schema
  - Spark timestamp resolution
  - Hash of Map type
  - Mitigate Spark HMS limitation
- Known issues
  - String order change
  - ORC +0/-0 bug



Rewriter job management

- Single Spark job issues
  - Slow with large tables
  - Higher chance of failure
  - Debuggability suffer
- Concurrent jobs among partitions
  - Naive approach: fix number of partitions per job
  - Final solution: split on partition count and file size
- Core# per job tuning: use smaller clusters
  - Reduce waiting time
- Scheduler: managing 100s jobs
  - prepare: turn-on mix-format, permissions etc.
  - rewrite: launch/poll/retry
  - clean-up: turn-off mix-format

Mixed format support

- Goal
  - incremental migration
  - zero-downtime for readers
  - Limited scope: per-table flag; only during migration
- Hive change
  - Schema-evolution on Parquet / ORC
  - Fix Serde issue causing NULL values
- Spark change
  - Force HadoopRDD over native data source
- Known issues
  - transient HMS / file format mismatch

ETL pipeline migration

- Figure out what / where / when
  - Pipeline update based on DDL / DML patterns
  - Select the running DC
  - Find the safe run time
- Data driven analysis
  - HMS / Hive / Spark audit logs
  - Job API for running state query

What to run

- Non-partition vs partitioned tables
- DDL patterns
  - create-table-if-not-exists: alter prod table
  - drop-n-create: update DDL
- DML patterns
  - insert-(overwrite): alter table format
  - swap locations: alter staging table format
  - dual-table sharing locations: data update on main table; alter format on both
- Data driven analysis
  - HMS / Hive audit logs
  - Manual analysis for Spark / Presto jobs

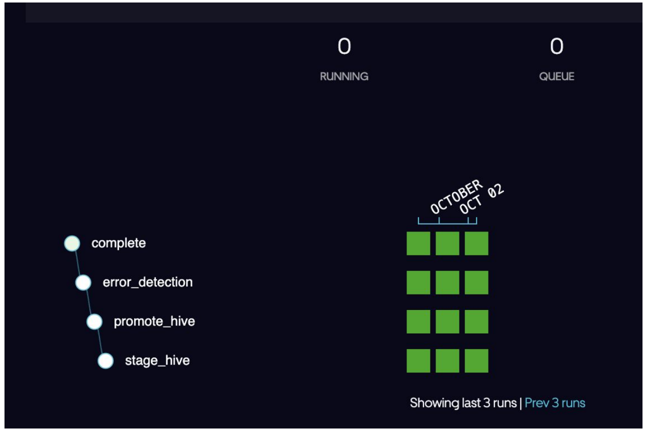
Where to run

- Two DCs
- ETL pipeline mode
  - one-dc
  - primary-dc
  - secondary-dc
  - both-dc
- Job API to track ETL location
- Replication handling

Replication	**Solution**
Double-single compute	No replication
Hive-Sync	Trigger Hive Metastore events
Data-Platinum	Explicit invoking RPC services

When to run

- Reduce migration and ETL conflict
- Use Job API to find run intervals
- Challenging cases
  - Job state may be unreliable
  - Non stopping jobs
  - Tables used in multiple jobs



Future work

- Full automation: from 1k -> 20k
- Optimize Spark app efficiency
  - Better error handling
  - Adapt to different file size / count

Learnings

- Flexible systems are hard to migrate
- Complexities hide in the details
- Audit logs across infra are valuable
- Make migration support a first class feature

Thanks