Moving the needle of the Pin:

Streaming 100 TB of pins from MySQL to S3 / Hadoop continuously @ Pinterest

Henry Cai
www.linkedin.com/in/hecai

Yi Yin
www.linkedin.com/in/niyiy

March 7, 2018
Pinterest is the visual discovery engine.

Mission
Help people discover and do what they love.
75% of signups are from outside the U.S.

80% of Pinners use Pinterest from mobile

100B Pins and 2B Boards

>200M monthly active users
Data-driven products

- Personalized recommendation
- Spam Control
- Search Quality
- A/B Experiments
- Related Pins
- ...
Data ingestion stats

- > 150 billion messages / day
- > 200 TB data / day
- > 1,000 kafka brokers
- > tens of thousand of hosts
Data ingestion types

- Online logging
- Database snapshots
2016 pipeline
Data ingestion @ Pinterest 2016
Data ingestion @ Pinterest 2016

Pinterest Services

Singer

Kafka

events
Data ingestion @ Pinterest  2016

Pinterest Services → Singer → Kafka (events) → Real-time consumers

Pinterest Services

Spark Streaming

Merced

S3

Databases

Tracker

TensorFlow

presto

Apache Spark

Hadoop
Data ingestion @ Pinterest 2016

Pinterest Services → Kafka

Kafka → Real-time consumers

Pinterest Services → Singer

Singer → Merced

Merced → Databases

Databases → S3

S3 → hadoop, Spark, TensorFlow, presto
DB ingestion @ Pinterest

**Version 1**
- Shard 1 Master
- Shard 1 Slave
- Shard 1 Dr Slave
- Shard 2 Master
- Shard 2 Slave
- Shard 2 Dr Slave

- Mysqldump
- Hadoop Streaming

**Version 2**
- Databases
- logical csv backup

**Tracker**
Pain points

Constraints

- Reliability caused by mysql hosts hiccup
- Pulling over 100TB data daily but only a few TB changed every day
- Long latency > 24 hour

Future: DB Change Streams

- Truly captures db transactions
- Across-region cache invalidation
- Realtime search index building
- Realtime Recommendation Engine
The new pipeline
Data ingestion @ Pinterest now
Data ingestion @ Pinterest now

Pinterest Services → Singer → Kafka

Pinterest Services:
- Pinterest Services
- Singer
- Kafka
- Databases
- DB/Kafka Bridge
- events
Data ingestion @ Pinterest now

Pinterest Services

Singer

Kafka

DB/Kafka Bridge

Merced

Watermill

Databases

S3
Data ingestion @ Pinterest now

Data ingestion process:

1. Pinterest Services generate Singer events.
2. Singer events are ingested into Kafka.
3. Kafka events are processed by Spark Streaming.
4. Real-time consumers consume Kafka events.
5. Databases are integrated with a DB/Kafka Bridge.
6. Merced and Watermill facilitate data movement between different systems.
7. S3 is used for storage and integration with other tools like Hadoop, Presto, and TensorFlow.
DB/Kafka Bridge

Replica-Set Node
MySQL Processes and Schemas

- User Tables
  - Shard 1
  - Shard 2
  - Shard 3

- Maxwell Tables
  - Maxwell_position
  - Maxwell_schema

Binlog File
**DB/Kafka Bridge**

**Replica-Set Node**

MySQL Processes and Schemas

- **User Tables**
  - Shard 1
  - Shard 2
  - Shard 3
- **Maxwell Tables**
  - Maxwell_position
  - Maxwell_schema

**MySQL Processes (Co-located with MySQL Process)**

- **BinLog Tailer Thread**
- **InMemory Queue**
- **Async Kafka Producer Thread**

**Kafka User Topic**

- Based on Maxwell / Binlog-Connector
- Add GTID support
- Add handling for retry/out-of-order messages
- Co-locate with mysql
- Listens on master/slave

**Kafka Pin Topic**
Watermill compaction

Databases

DB/Kafka Bridge

Singer

Pinterest Services

Kafka events

Merced

Watermill

Real-time consumers

Spark Streaming

Amazon web services
Compaction For One Shard

- Hash Join between snapshot and delta
- Delta loaded in memory first as side lookup
- Base snapshot was piped through the mapper node and compare against lookup table
  - Lookup fail, snapshot record emit to output
  - Lookup succeed, but snapshot record old, skip the snapshot
  - Lookup succeed, but snapshot record newer, remove lookup record
- At the end, append the remaining lookup records to output
Incremental DB ingestion sequence
Incremental DB ingestion sequence
Incremental DB ingestion sequence

MySQL -> Maxwell -> Kafka -> Merced -> Delta -> Periodic Compaction -> Snapshot 1 -> Snapshot 2
Incremental DB ingestion sequence

MySQL -> Tracker Batch Backup -> Backup Snapshot

Backup Snapshot -> Maxwelle

Maxwell -> Kafka

Kafka -> Merced

Merced -> Delta

Delta -> Periodic Compaction

Periodic Compaction -> Snapshot 2

Snapshot 2 -> Snapshot 1

Snapshot 1 -> Bootstrapper
Incremental DB ingestion sequence

MySQL → Tracker Batch Backup

Tracker Batch Backup → Backup Snapshot → Bootstrapper

Bootstrapper → Snapshot 1

Snapshot 1 → Kafka → Merced → Delta → Periodic Compaction

Delta → Periodic File GC

Periodic File GC → Snapshot 2 → Differ

Differ → Periodic Compaction
Incremental DB ingestion sequence

MySQL

Tracker Batch Backup

Backup Snapshot

Bootstrapper

Snapshot 1

Maxwell

Kafka

Merced

Delta

Periodic Compaction

Periodic File GC

Custom Input Format

SELECT FROM rt_users

Differ
Data Lifecycle and Timeline Management

Timeline

11:30

Daily Dump

11:55

Bootstrap Snapshot
Data Lifecycle and Timeline Management

Timeline:
- Daily Dump: 11:30
- Bootstrap Snapshot: 11:55
- Kafka
- Merced
- Delta: 12:01
Data Lifecycle and Timeline Management

- Daily Dump: 11:30
- Bootstrap: 11:55
- Kafka
- Merced
- Delta: 12:01
- Compaction
- Snapshot: 12:10 AM

Timeline:
- 11:30
- 11:55
- 12:01
- 12:10
Data Lifecycle and Timeline Management

Daily Dump 11:30 → Bootstrap Snapshot 11:55 → Merced → Delta 12:01 → Compaction → Snapshot 12:10AM → 12:15 Select

Timeline:
11:30
11:55
12:01
12:10
12:15
Data Lifecycle and Timeline Management

- **Daily Dump 11:30**
  - **Bootstrap Snapshot 11:55**
  - **Merced**
  - **Delta 12:01**
  - **Compaction**
  - **Snapshot 12:10AM**
  - **12:15 Select**

Timeline:
- **11:30**
- **11:55**
- **12:01**
- **12:10**
- **12:15**

- **Processed UpTo**
- **Current Snapshot**
Data Lifecycle and Timeline Management

Timeline

11:30 11:55 12:01 12:10 12:20 11:45 12:20

Daily Dump 11:30 → Bootstrap Snapshot 11:55 → Kafka → Merced → Delta 12:01 → Compaction → Snapshot 12:10AM → 12:15 Select

Daily Dump 11:45 → Bootstrap Snapshot 12:20

Bootstrap Snapshot 11:55 → Daily Dump 11:30 → Bootstrap Snapshot 12:20

Kafka
Data Lifecycle and Timeline Management

Timeline:

- Daily Dump 11:30
- Bootstrap Snapshot 11:55
- Delta 12:01
- Compaction
- Snapshot 12:10AM
- Daily Dump 11:45
- Bootstrap Snapshot 12:20

Kafka

- 12:25 Select
- 12:15 Select

Current Snapshot: 12:20
Processed Upto: 11:45
Data Lifecycle and Timeline Management

Timeline:
- Daily Dump 11:30
- Bootstrap Snapshot 11:55
- Kafka
- Merced
- Delta 12:01
- Compaction
- Snapshot 12:10 AM
- Daily Dump 11:45
- Bootstrap Snapshot 12:20
- 12:25 Select

Processed Upto:
- Daily Dump 11:45
- Bootstrap Snapshot 12:20
- ... Next Compaction ...
Data Lifecycle and Timeline Management

Timeline:

11:30 11:55 12:01 12:10 11:45 12:20

Possible Rewind

Periodic GC

Compaction

Delta

Snapshot

12:01

12:10AM

11:45

12:20

Bootstrap

Snapshot

11:55

11:55

Snapshot

12:0000

12:00

1:3

1:5

1:2

1:1

1:0

Daily Dump

Daily Dump

Kafka

Merced

Compaction

Possible Rewind
Consistency

- MySQL Master/Slave Failover, Shard Migration
- MySQL Transactions:
  - Split between tables, split between Kafka messages
- Ordering
  - Between INSERT and UPDATE
  - Between UPDATE and DELETE
- Soft DELETE vs. Hard DELETE
- Consistency between multiple bootstrap and incremental streams
- Duplicate Records
Scalability

• Partitioning
  • Sharded MySQL
    - Shard based db snapshot and delta files
    - Two level sharing in the case that original shards are not balanced
  • UnSharded dataset
    - Use hash + mod to partition the data on both snapshot and delta file
• File filtering using predicate pushdown:
  • On shard/partition level
  • On S3 directory, file and record level
Kafka Nuances

• Message Ordering:
  • Async producer but still need to maintain message order
  • Maintain order between S3 file and within S3 file

• At-least-once delivery
  • Duplicate messages
  • MySQL GTID not always increasing

• Deal with Kafka cluster hiccup:
  • producer acks = 2
  • clean leader election
S3 Nuances

- Eventual Consistency
  - Read-after-write is OK, but not PUT followed by LIST

- Directory listing is slow
  - Shorter SLA → More smaller files
  - In early iterations, directly listing >> file content reading

- Rate Limit:
  - Launching thousands of mappers would quickly hit S3 rate limit
PII Processing

• username, email address etc needs to be filtered out
• ip address needs to be filtered out
PII Processing

- username, email address etc needs to be filtered out
- ip address needs to be filtered out

Columnar Layout and Incremental Processing

- Use parquet format to support fast queries on subset of columns
- ingest_time as new column to get the incremental result since the last processing;
Bootstrap, synchronize & rewind

MySQL

Tracker Batch Backup

Backup Snapshot

Bootstrapper

Snapshot 1

Snapshot 2

Maxwell

Kafka

Merced

Delta

Periodic Compaction
We have the ability to synchronize and rewind:
- In case of software bugs or network glitches
- Snapshot(s) onto Bootstrap to synchronize
- Ability to rewind via the Snapshots/Bootstrap mechanism
Schema Management and Schema Change

- **Schema is Used for**
  - Identify the primary key of the row
  - Drive the parquet file generation

- **Dealing With Schema change**
  - Will issue a new bootstrap on offline table schema
  - Compaction will still use the snapshot schema (which might be old)

<table>
<thead>
<tr>
<th>ID</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>....</td>
<td>...</td>
</tr>
<tr>
<td>124</td>
<td>...</td>
<td>....</td>
</tr>
<tr>
<td>125</td>
<td>....</td>
<td>...</td>
</tr>
<tr>
<td>126</td>
<td>...</td>
<td>....</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>new_column</th>
</tr>
</thead>
<tbody>
<tr>
<td>....</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>....</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Validation

- Validation
  - Creating compaction based on from and to GTID range
  - Compaction output vs batch backup output
- Monitoring
  - Error, failure, stall
  - Latency on compaction
Comparison to other technologies

- Uber Hudi (Hoodie)
  - Not supporting S3, Only support Java 8+, Avro
Comparison to other technologies

- Uber Hudi (Hoodie)
  - Not supporting S3, Only support Java 8+, Avro
- Kafka Connect
  - Only ingestion, no compacting, synchronize between bootstrap/incremental
Comparison to other technologies

- Uber Hudi (Hoodie)
  - Not supporting S3, Only support Java 8+, Avro
- Kafka Connect
  - Only ingestion, no compacting, synchronize between bootstrap/incremental
- Apache Sqoop
  - Based on Batch Mode
Takeaway

• Scalability
  • support 100TB of database data
  • E2E latency of 15 minutes

• Reliability
  • Strong database consistency on global transactions, message ordering, duplicate message handling
  • Validation and Monitoring

• Operability
  • Bootstrap, re-synchronize
  • Schema management
Future work

• Adopting Kafka Exact-Once Processing Model
• Kafka as the database change stream
  • Cache invalidation across data centers
  • Building Materialized Views for MySQL
  • Generating Incremental Recommendation Signals
• Open Source
Acknowledgement

- Joint work from many engineering, including Yu Yang, Chunyan Wang, Indy Prentice, Shawn Nguyen, Yinian Qi, and many others
Thanks!