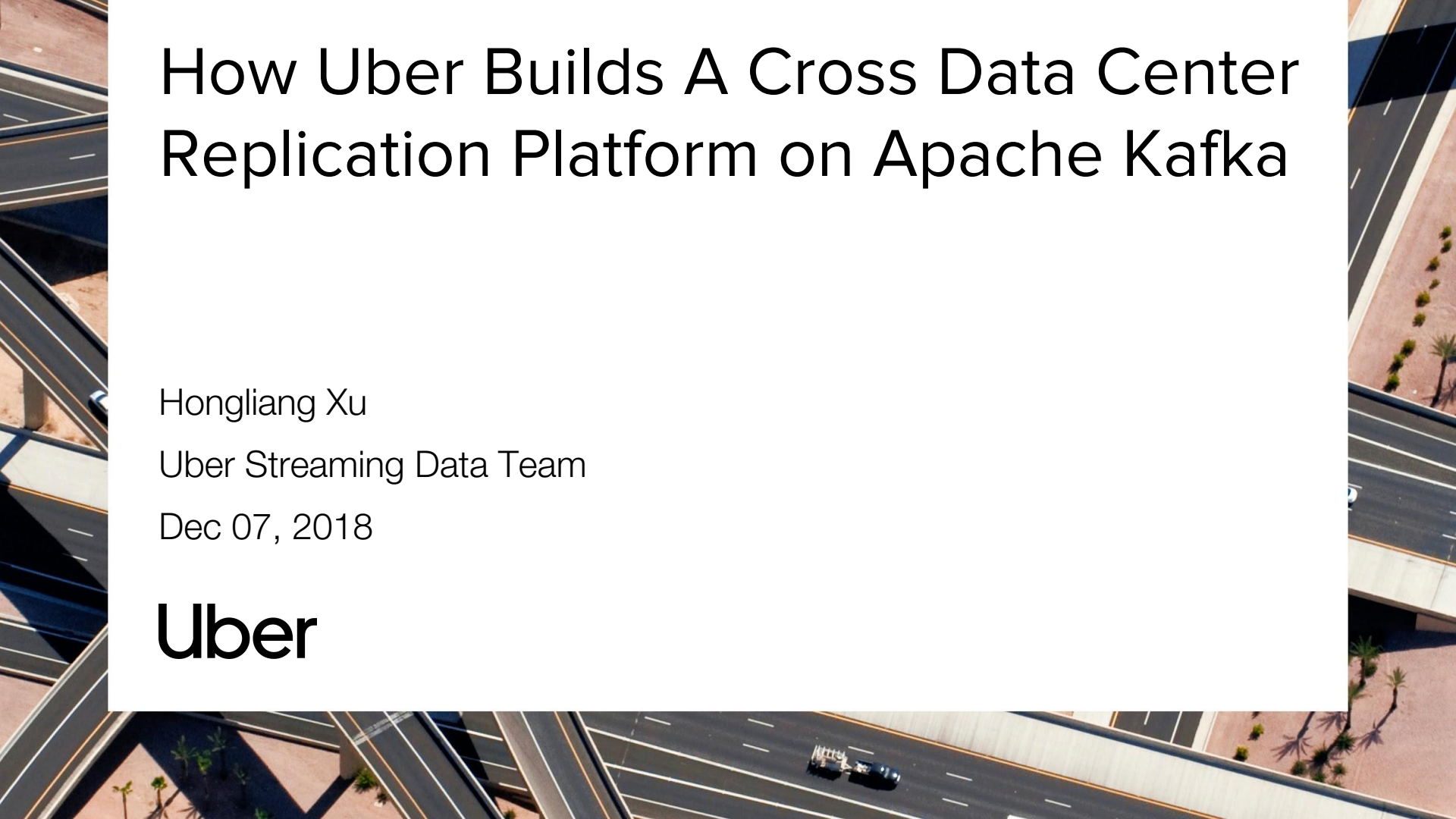


How Uber Builds A Cross Data Center Replication Platform on Apache Kafka

Hongliang Xu

Uber Streaming Data Team

Dec 07, 2018

Uber

Agenda

01 Apache Kafka at Uber
02 Apache Kafka pipeline & replication
03 uReplicator
04 Data loss detection
05 Q&A

Real-time Dynamic Pricing

Uber

App Views

Vehicle Information

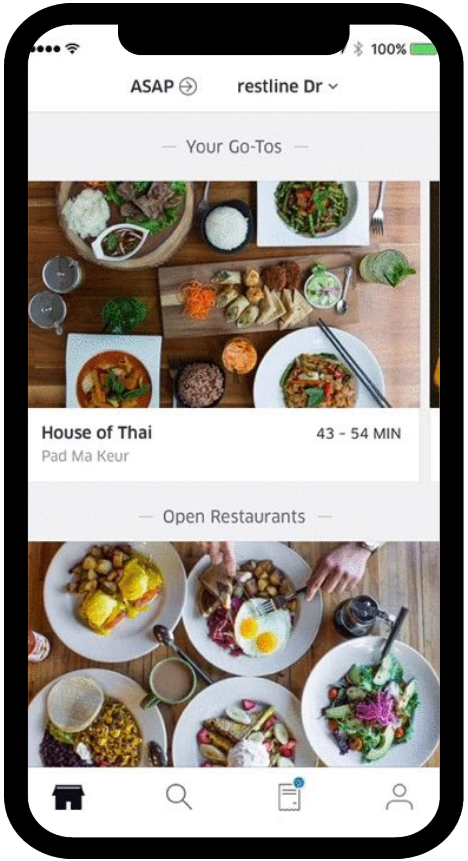
Apache Kafka

Stream Processing

Dynamic pricing

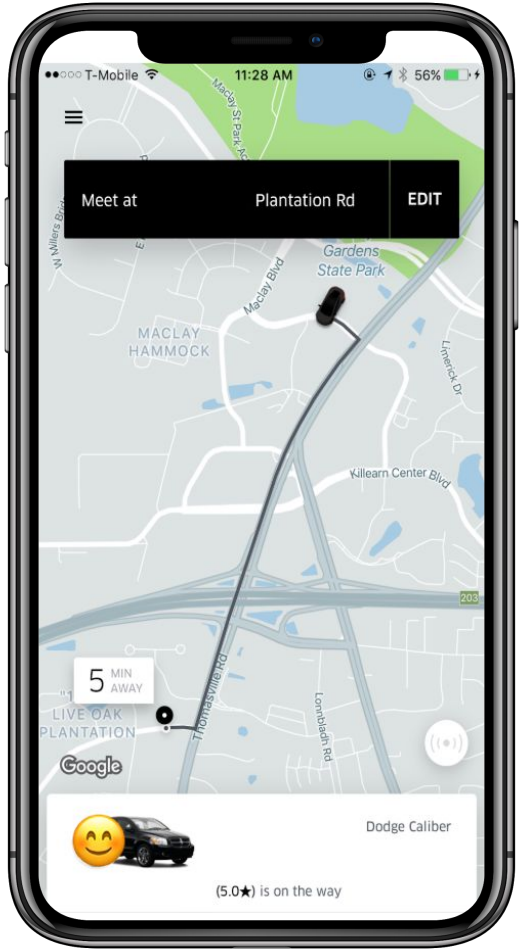
Uber Eats - Real-Time ETAs

Uber



A bunch more ...

- **Fraud Detection**
- **Driver & Rider Sign-ups, etc.**



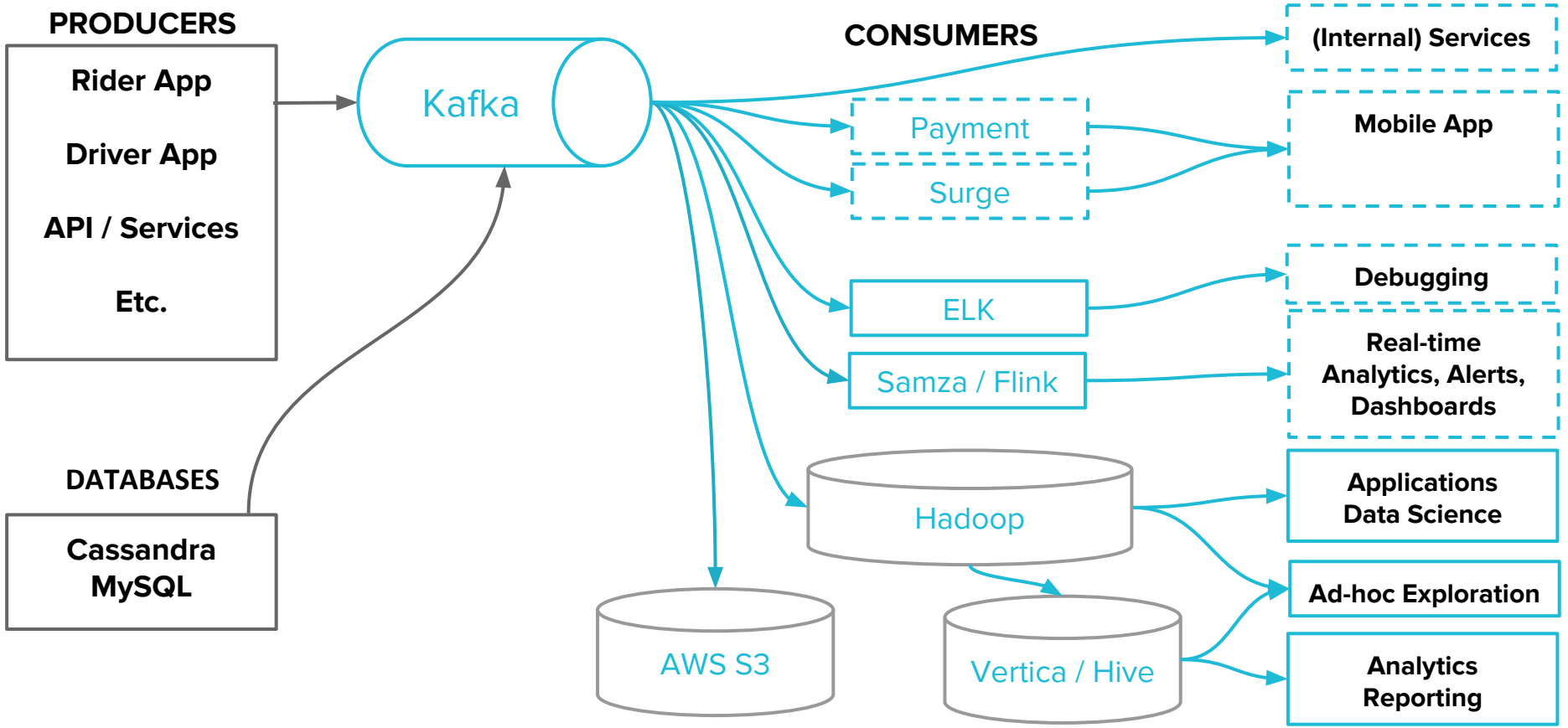
Apache Kafka - Use Cases

Uber

- General pub-sub, messaging queue
- Stream processing
  - AthenaX - self-service streaming analytics platform (Apache Samza & Apache Flink)
- Database changelog transport
  - Cassandra, MySQL, etc.
- Ingestion
  - HDFS, S3
- Logging

Data Infrastructure @ Uber

Uber



Scale

Uber

Trillions

Messages / Day

PBs

Data

Tens of Thousands

Topics

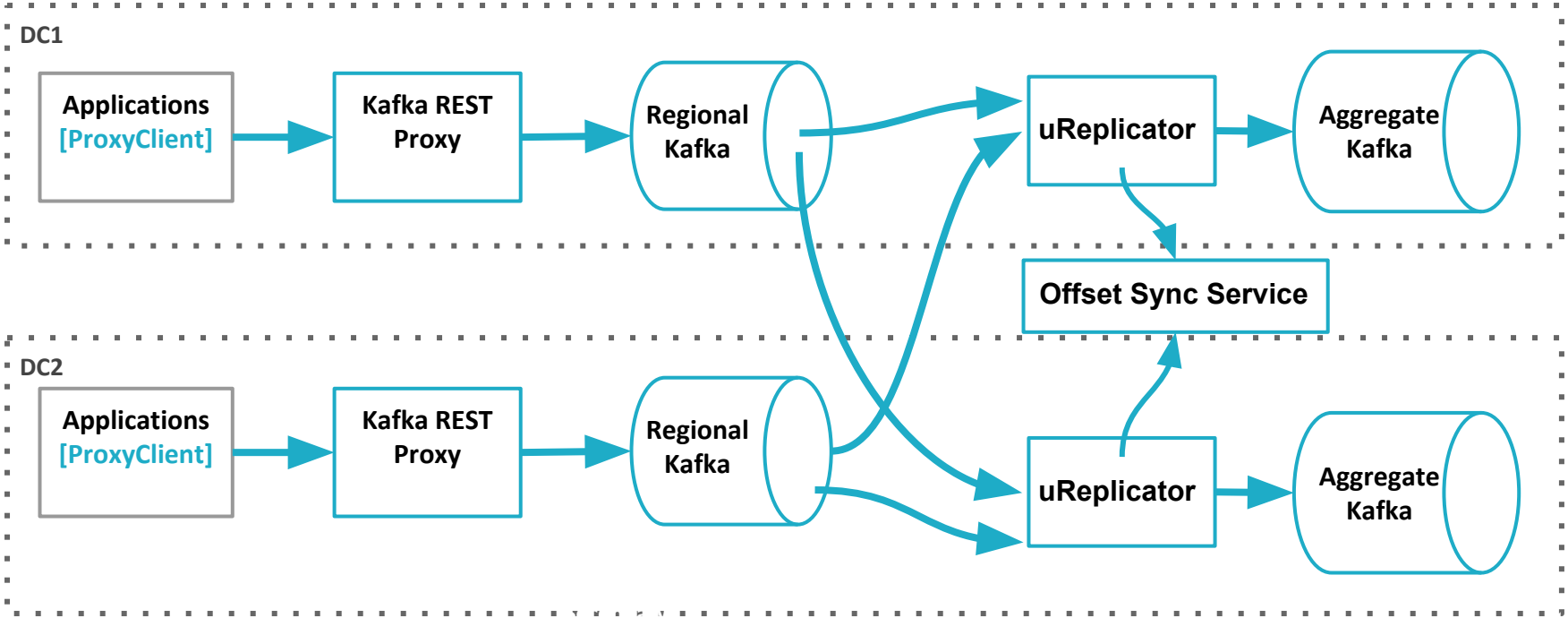
excluding replication

Agenda

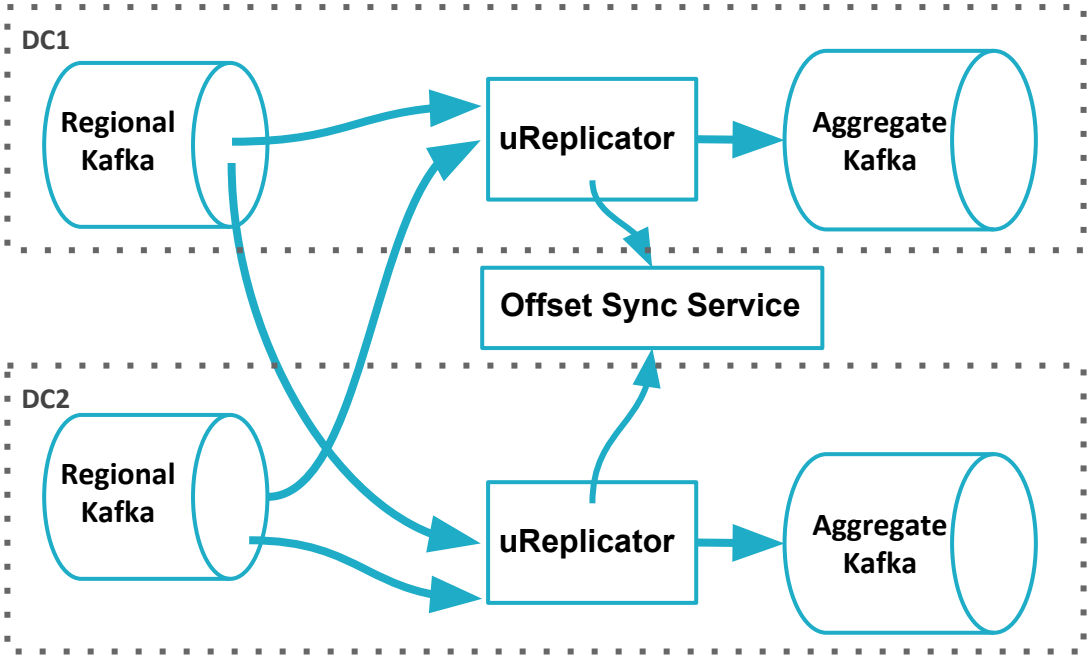
01 Apache Kafka at Uber
02 Apache Kafka pipeline & replication
03 uReplicator
04 Data loss detection
05 Q&A

Apache Kafka Pipeline @ Uber

Uber



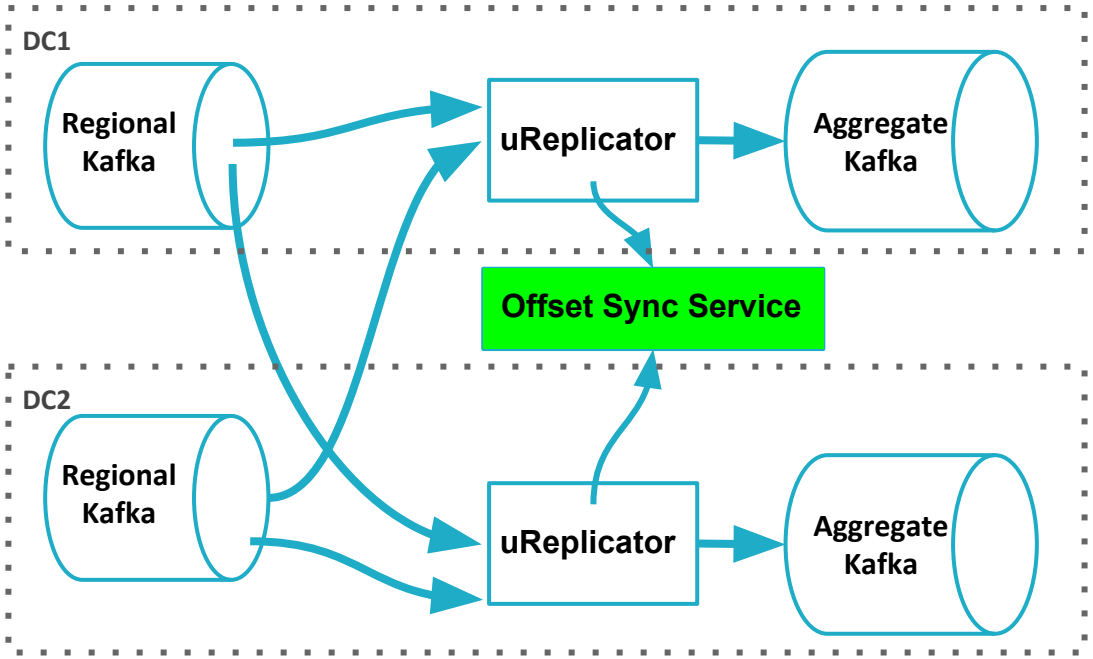
Aggregation



- Global view

Cross-Data Center Failover

Uber



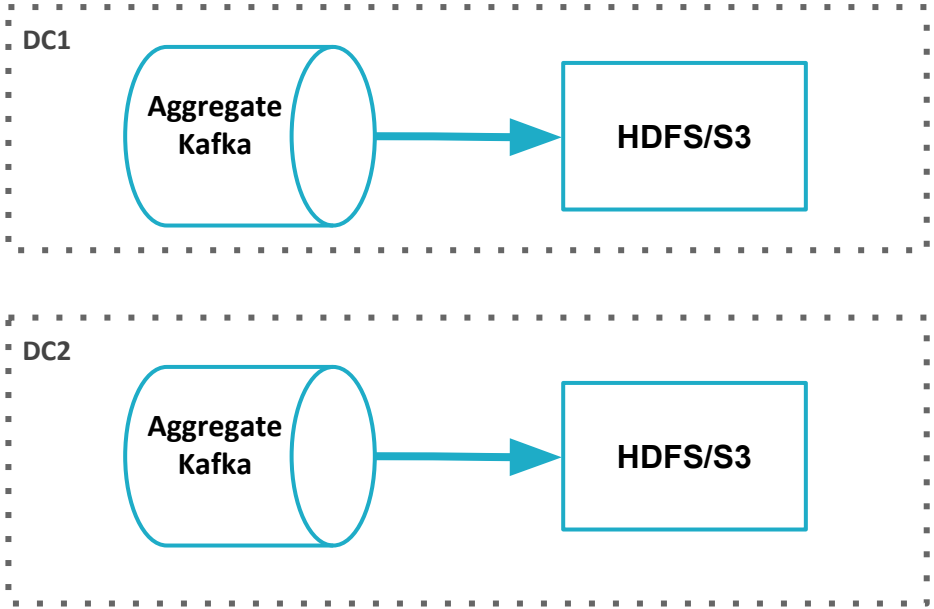
During runtime

- uReplicator reports offset mapping to offset sync service
- Offset sync service is all-active and the offset info is replicated across data centers

During failover

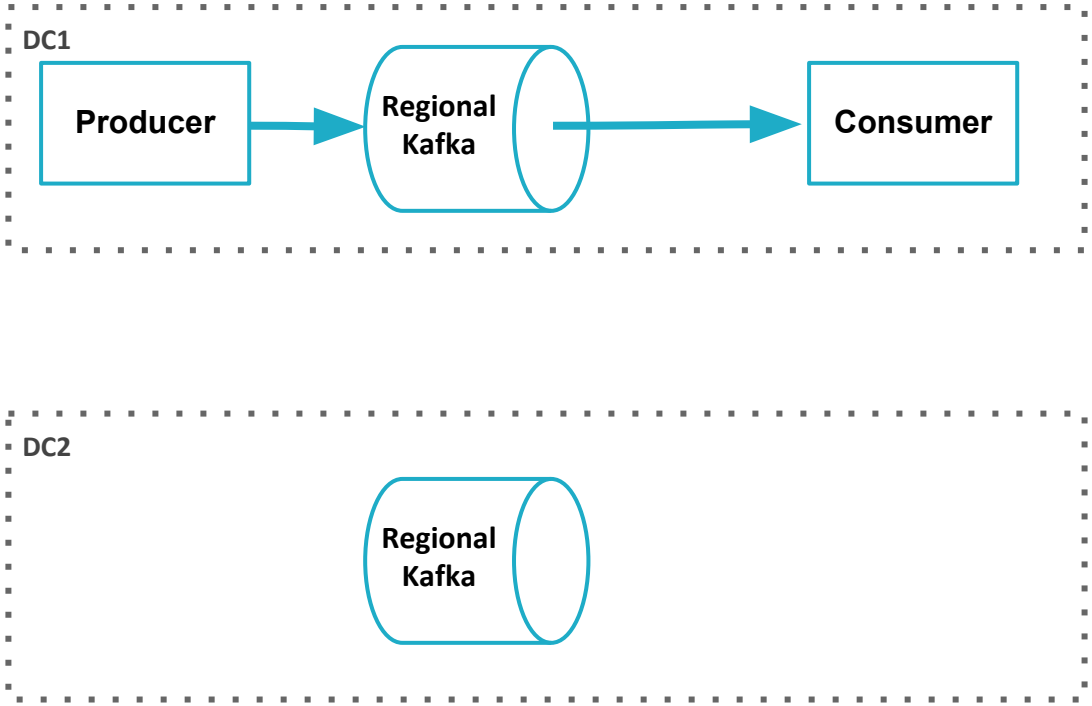
- Consumers ask offset sync service for offsets to resume consumption based on its last commit offsets
- Offset sync service translates offsets between aggregate clusters

Ingestion



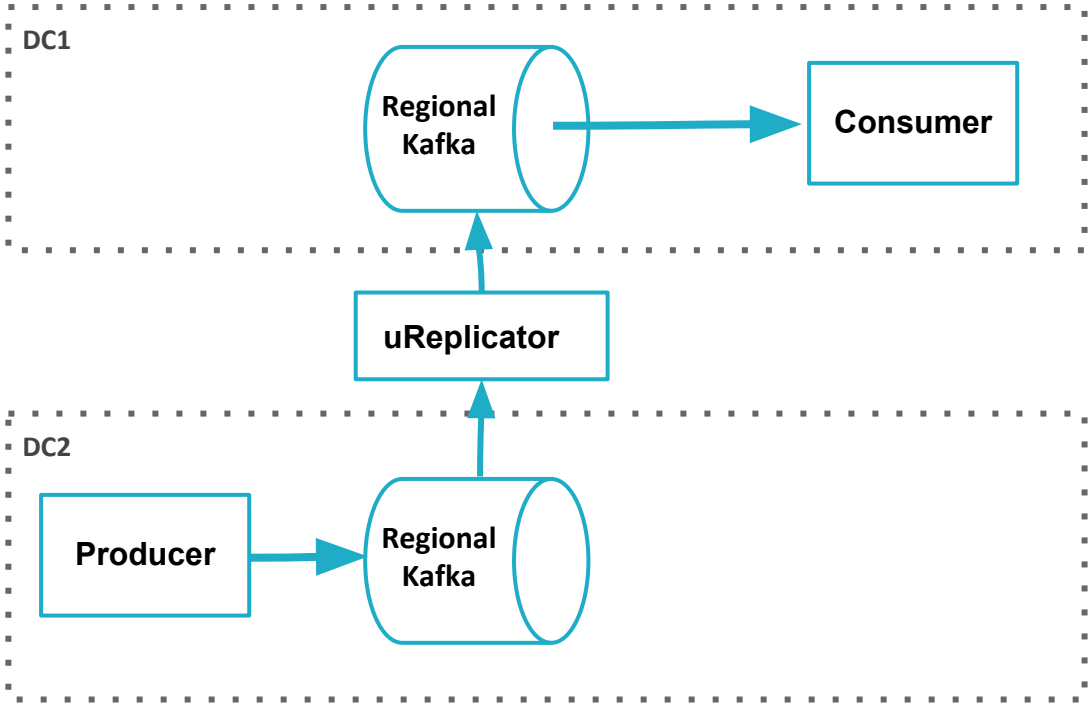
- HDFS
- S3

Topic Migration



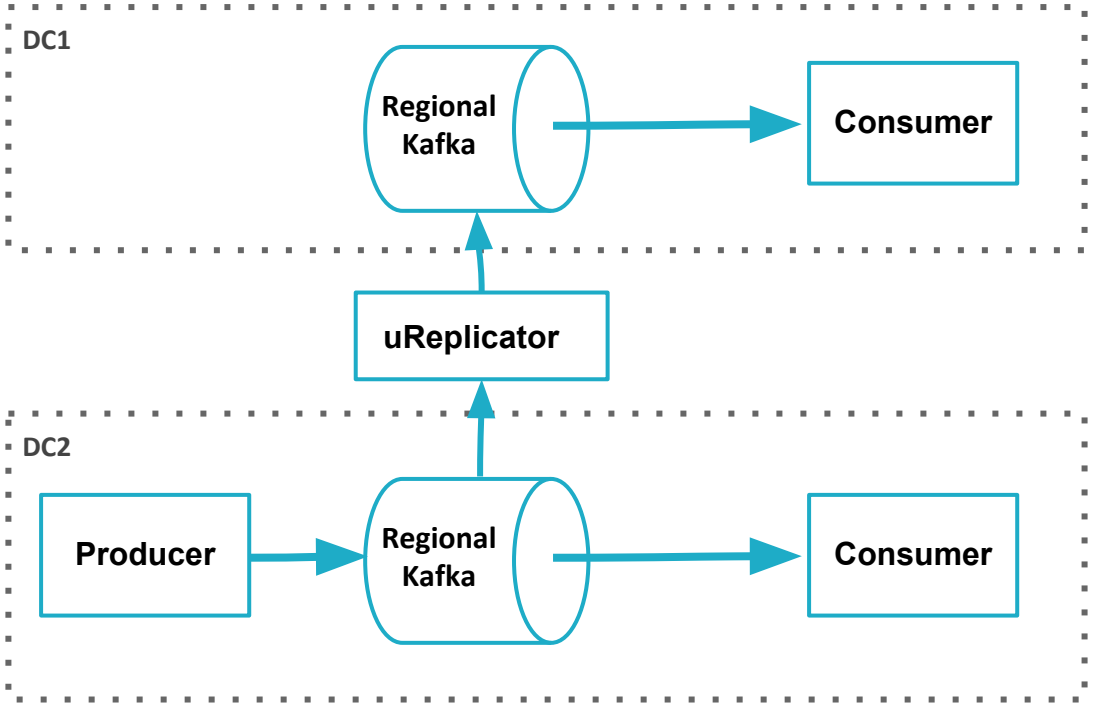
- Setup uReplicator from new cluster to old cluster
- Move producer
- Move consumer

Topic Migration



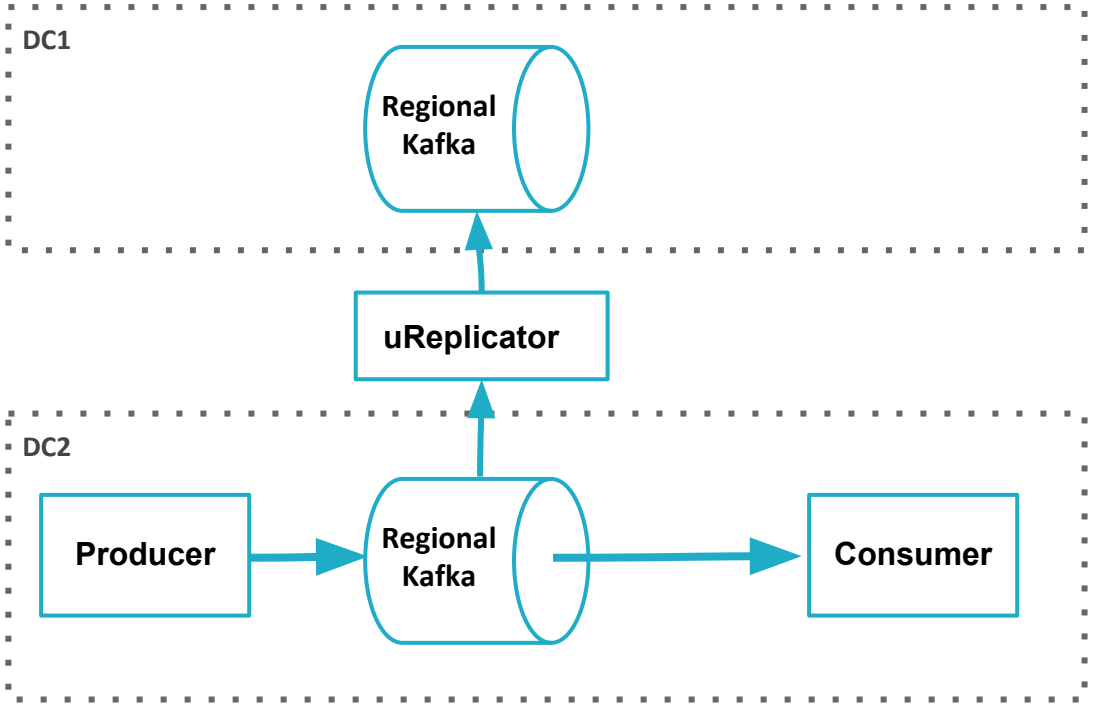
- Setup uReplicator from new cluster to old cluster
- Move producer
- Move consumer
- Remove uReplicator

Topic Migration



- Setup uReplicator from new cluster to old cluster
- Move producer
- Move consumer
- Remove uReplicator

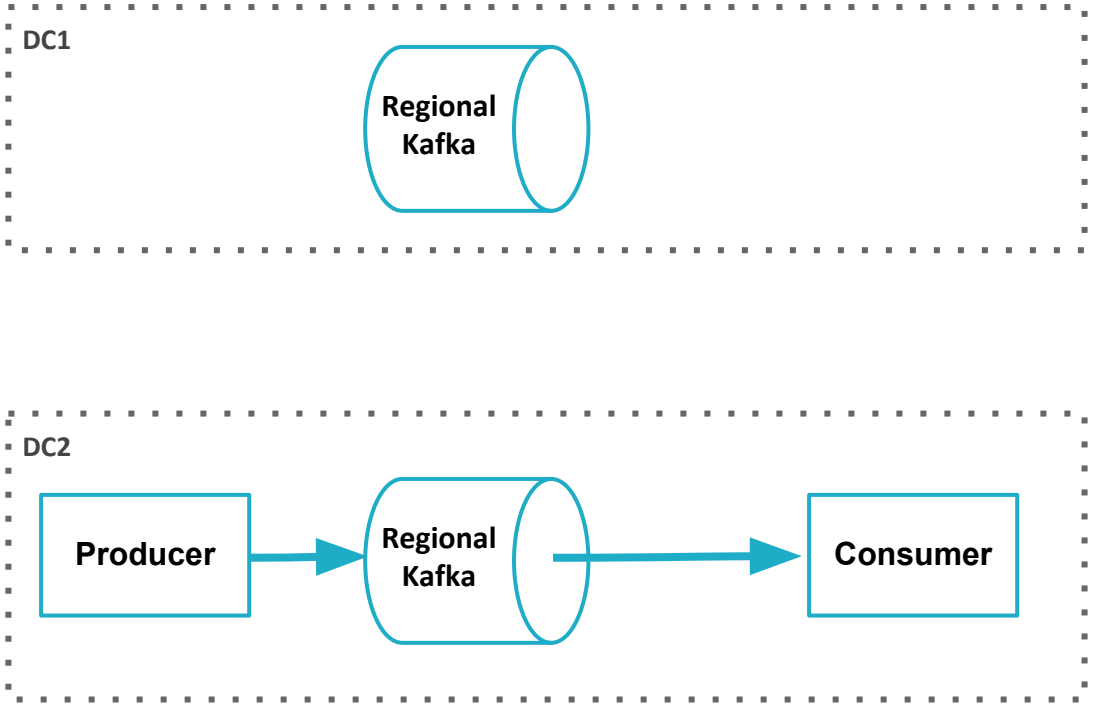
Topic Migration



- Setup uReplicator from new cluster to old cluster
- Move producer
- Move consumer
- Remove uReplicator

Topic Migration

Uber



- Setup uReplicator from new cluster to old cluster
- Move producer
- Move consumer
- Remove uReplicator

Replication - Use Cases

Uber

- Aggregation
  - Global view
  - All-active
  - Ingestion
- Migration
  - Move topics between clusters/DCs

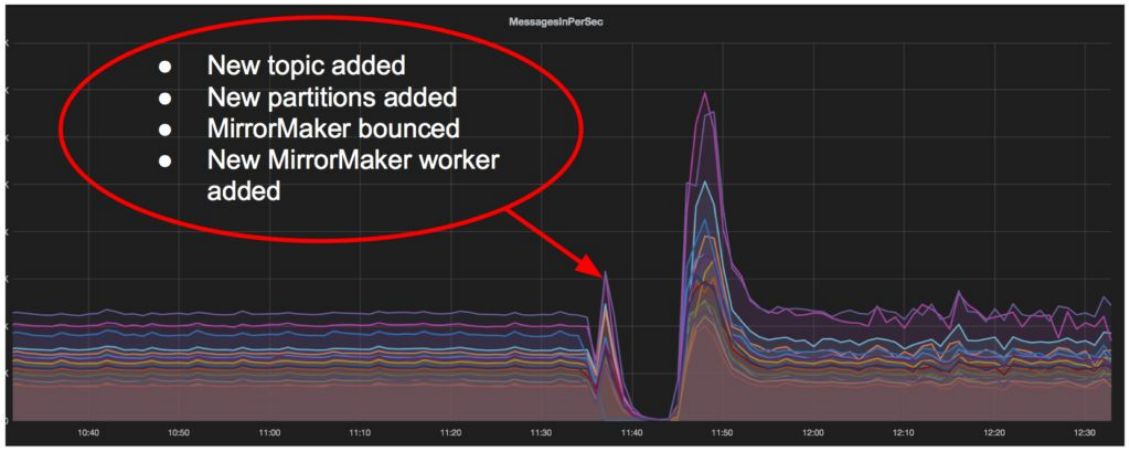
Agenda

01 Apache Kafka at Uber
02 Apache Kafka pipeline & replication
03 uReplicator
04 Data loss detection
05 Q&A

Motivation - MirrorMaker

Uber

- Pain point
  - Expensive rebalancing
  - Difficulty adding topics
  - Possible data loss
  - Metadata sync issues



Requirements

Uber

- Stable replication
- Simple operations
- High throughput
- No data loss
- Auditing

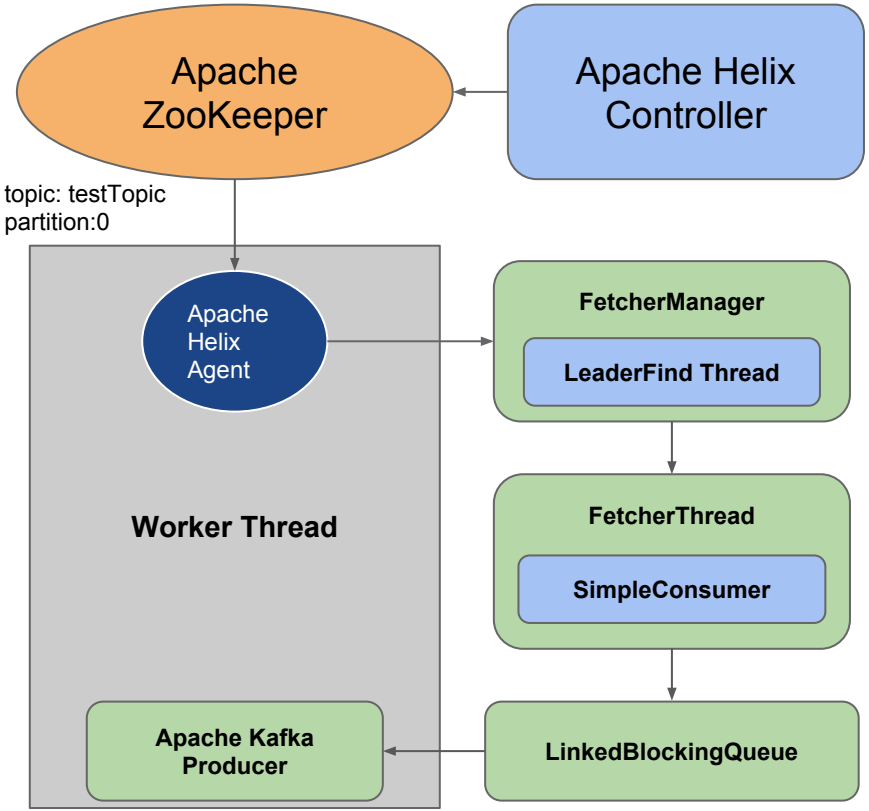
Design - uReplicator

Uber

- Apache Helix
- uReplicator controller
  - Stable replication
    - Assign topic partitions to each worker process
    - Handle topic/worker changes
  - Simple operations
    - Handle adding/deleting topics

Design - uReplicator

- uReplicator worker
  - Apache Helix agent
  - Dynamic Simple Consumer
  - Apache Kafka producer
  - Commit after flush



Requirements

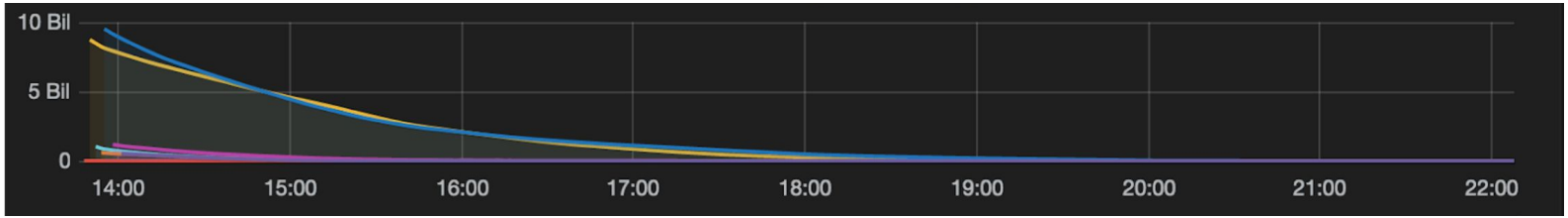
- ~~Stable replication~~
- ~~Simple operations~~
- High throughput
- ~~No data loss~~
- Auditing

Uber

Performance Issues

Uber

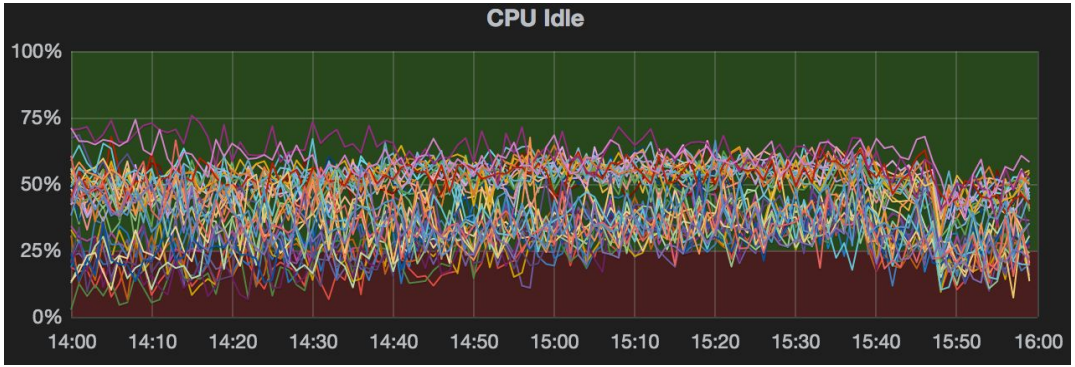
- Catch-up time is too long (4 hours failover)
  - Full-speed phase: 2~3 hours
  - Long tail phase: 5~6 hours

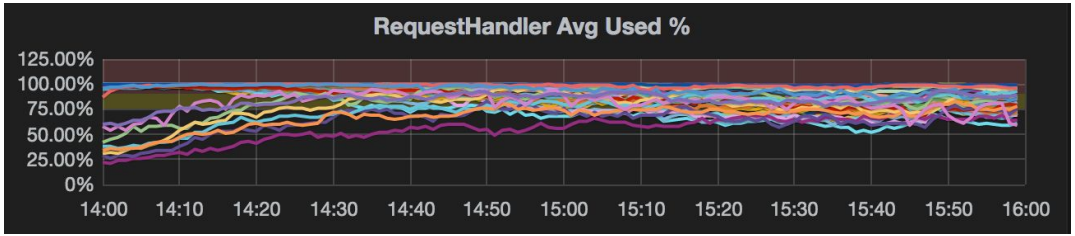


Problem: Full Speed Phase

Uber

- Destination brokers are bound by CPUs





Solution 1: Increase Batch Size

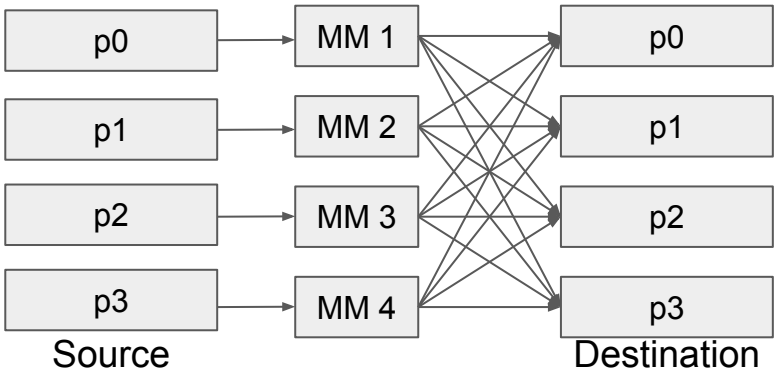
Uber

- Destination brokers are bound by CPUs
  - Increase throughput
  - producer.batch.size: 64KB => 128KB
  - producer.linger.ms: 100 => 1000
- Effect
  - Batch size increases: 10~22KB => 50~90KB
  - Compress rate: 40~62% => 27~35% (compressed size)

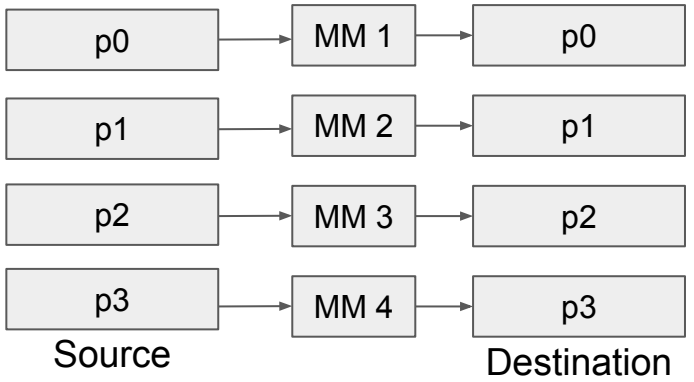
Solution 2: 1-1 Partition Mapping

Uber

- Round robin
- Topic with N partitions
- N^2 connection
- DoS-like traffic



- Deterministic partition
- Topic with N partitions
- N connection
- Reduce contention

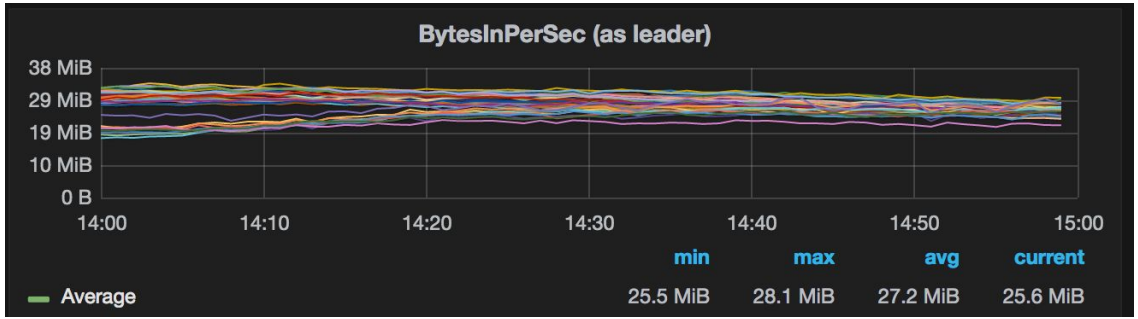


Full Speed Phase Throughput

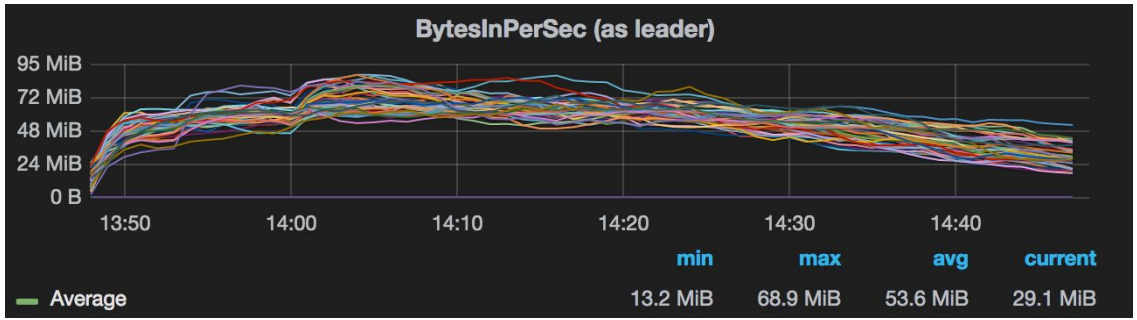
Uber

- First hour: 27.2MB/s => 53.6MB/s per aggregate broker

Before



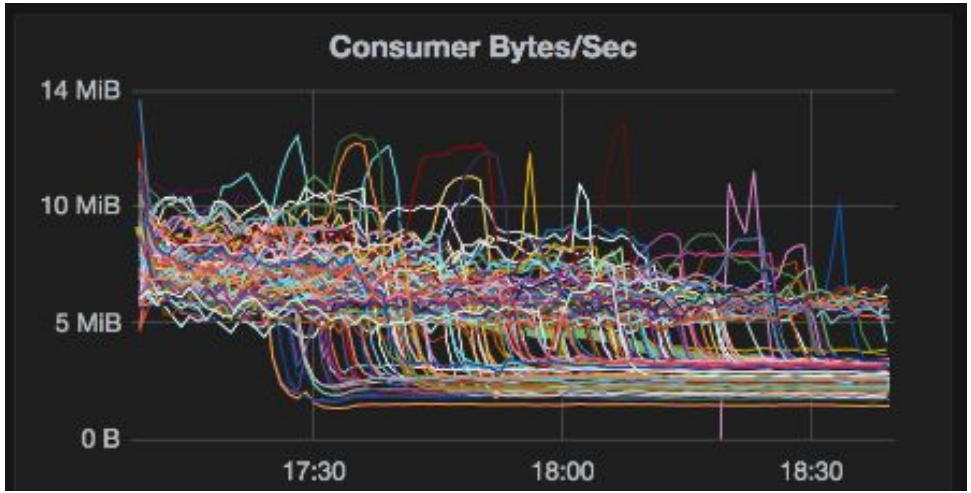
After



Problem 2: Long Tail Phase

Uber

- During full speed phase, all workers are busy
- Some workers catch up and become idle, but the others are still busy



Solution 1: Dynamic Workload Balance

Uber

- During full speed phase, all workers are busy
- Original partition assignment
  - Number of partitions
  - Heavy partition on the same worker
- Workload-based assignment
  - Total workload when added
    - Source cluster bytes-in-rate
    - Retrieved from Chaperone3
  - Dynamic rebalance periodically
    - Exceeds 1.5 times the average workload

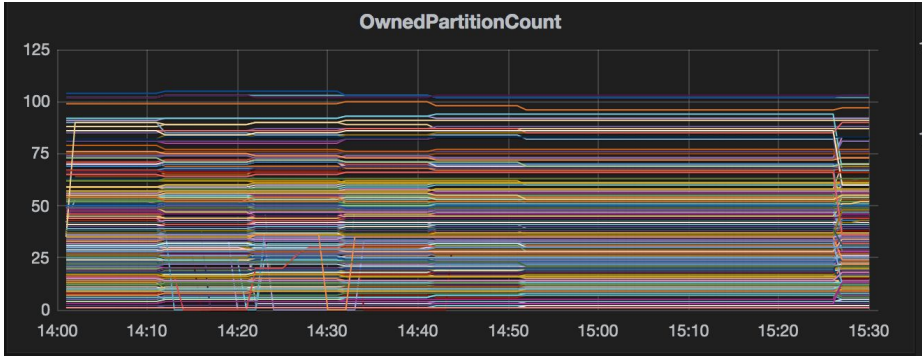
Solution 2: Lag-Feedback Rebalance

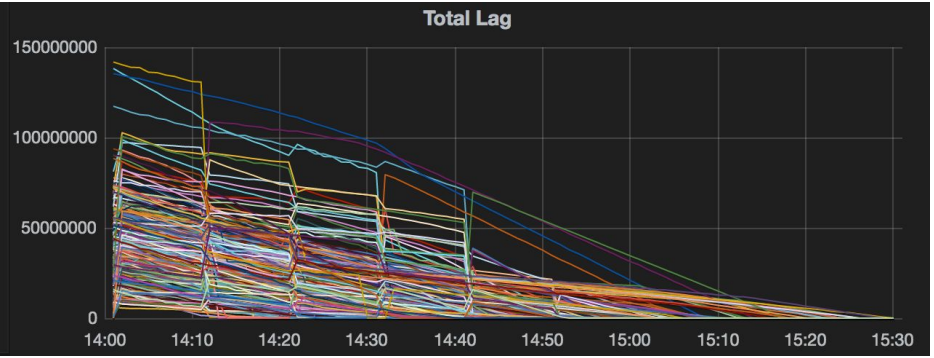
Uber

- During full speed phase, all workers are busy
- Monitors topic lags
  - Balance the workers based on lags
    - Larger lag = heavier workload
  - Dedicated workers for lagging topics

Dynamic Workload Rebalance

- Periodically adjust workload every 10 minutes
- Multiple lagging topics on the same worker spreads to multiple workers





Requirements

- ~~Stable replication~~
- ~~Simple operations~~
- ~~High throughput~~
- ~~No data loss~~
- Auditing

Uber

Requirements

Uber

- Stable replication
- Simple operations
- ~~High throughput~~
- ~~No data loss~~
- Auditing

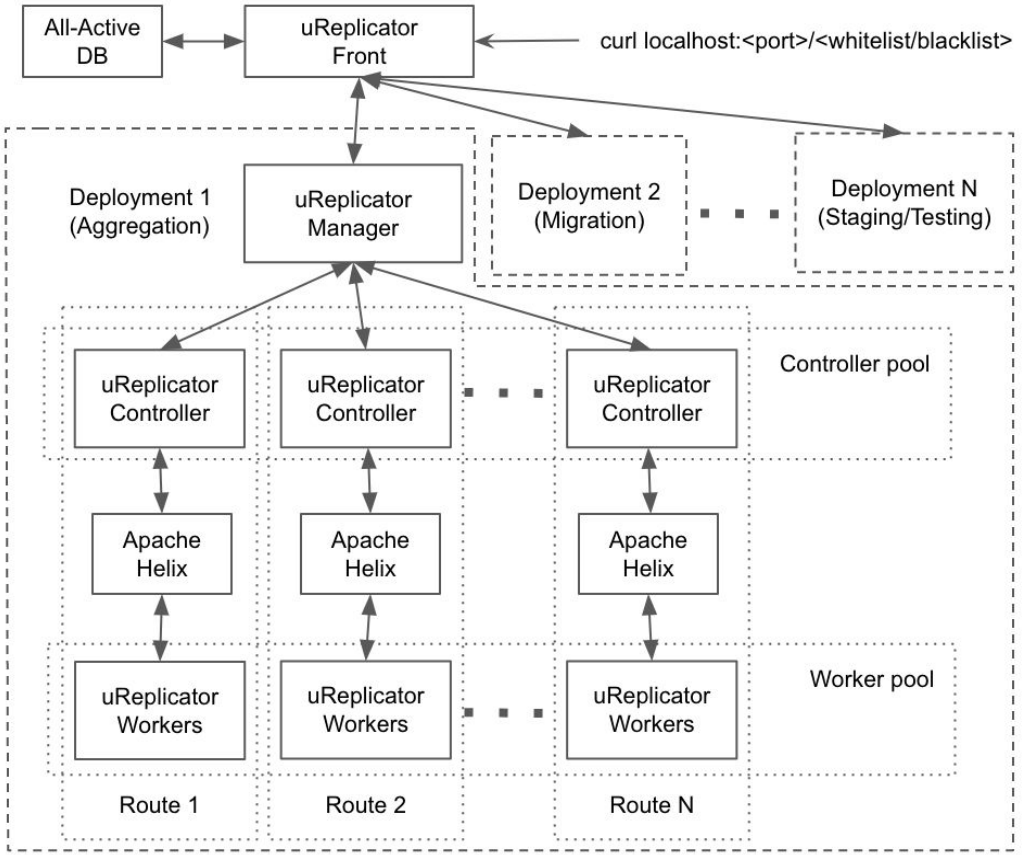
More Issues

Uber

- Scalability
  - Number of partitions
  - Up to 1 hour for a full rebalance during rolling restart
- Operation
  - Number of deployments
  - New cluster
- Sanity
  - Loop
  - Double route

Design - Federated uReplicator

- Scalability
  - Multiple route
- Operation
  - Deployment group
  - One manager



Design - Federated uReplicator

Uber

- uReplicator Manager
  - Decide when to create new route
  - Decide how many workers in each route
  - Auto-scaling
    - Total workload of route
    - Total lag in the route
    - Total workload / expected workload on worker
    - Automatically add workers to route

Design - Federated uReplicator

Uber

- uReplicator Front
  - whitelist/blacklist topics (topic, src, dst)
  - Persist in DB
  - Sanity checks
  - Assigns to uReplicator Manager

Requirements

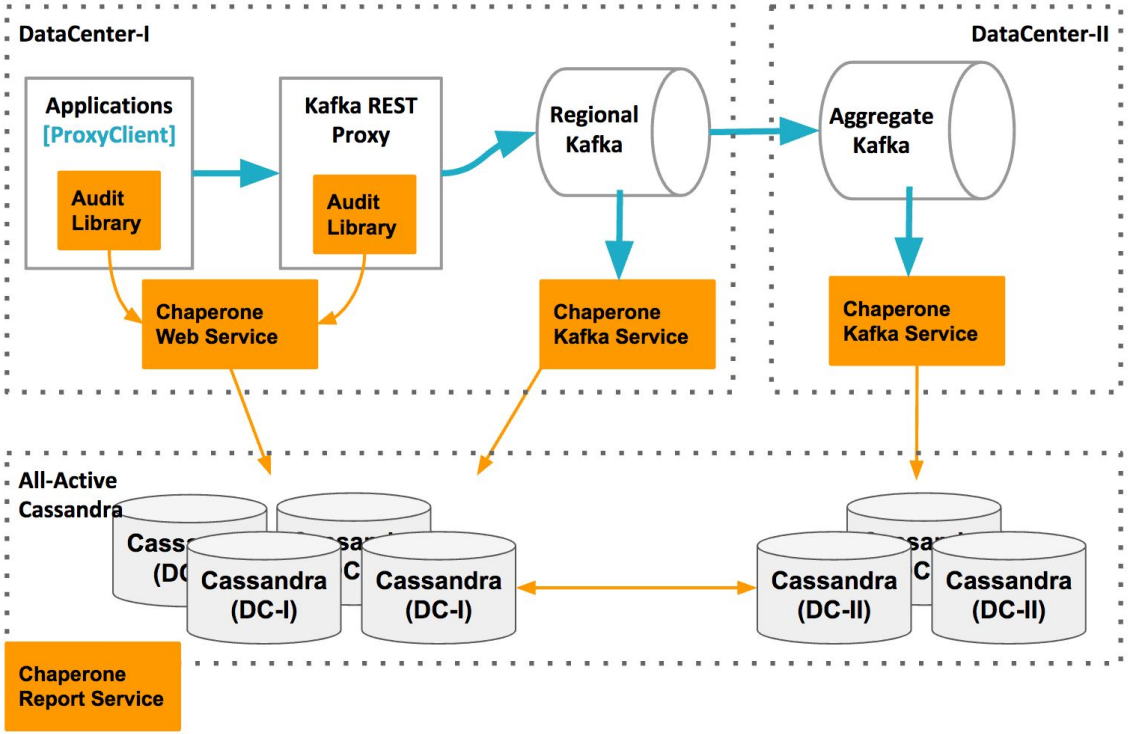
- ~~Stable replication~~
- ~~Simple operations~~
- ~~High throughput~~
- ~~No data loss~~
- Auditing

Uber

Agenda

01 Apache Kafka at Uber
02 Apache Kafka pipeline & replication
03 uReplicator
04 Data loss detection
05 Q&A

Detect data loss



- It checks data as it flows through each tier of the pipeline
- It keeps ts-offset index to support ts-based query
- It also checks latency from the PoV of consumer
- It also provides workload for each topic

Requirements

- ~~Stable replication~~
- ~~Simple operations~~
- ~~High throughput~~
- ~~No data loss~~
- ~~Auditing~~

Uber

Blogs & Open Source

Uber

- uReplicator
  - Running in production for 2+ years
  - Open sourced: https://github.com/uber/uReplicator
  - Blog: https://eng.uber.com/ureplicator/
- Chaperone
  - Running in production for 2+ years, audit almost all topics in our clusters
  - Open sourced: https://github.com/uber/chaperone
  - Blog: https://eng.uber.com/chaperone/

Agenda

01 Apache Kafka at Uber
02 Apache Kafka pipeline & replication
03 uReplicator
04 Data loss detection
05 Q&A