Real Time Data Analytics @ Uber

Ankur Bansal

Apache Big Data Europe

November 14, 2016

UBER

About Me

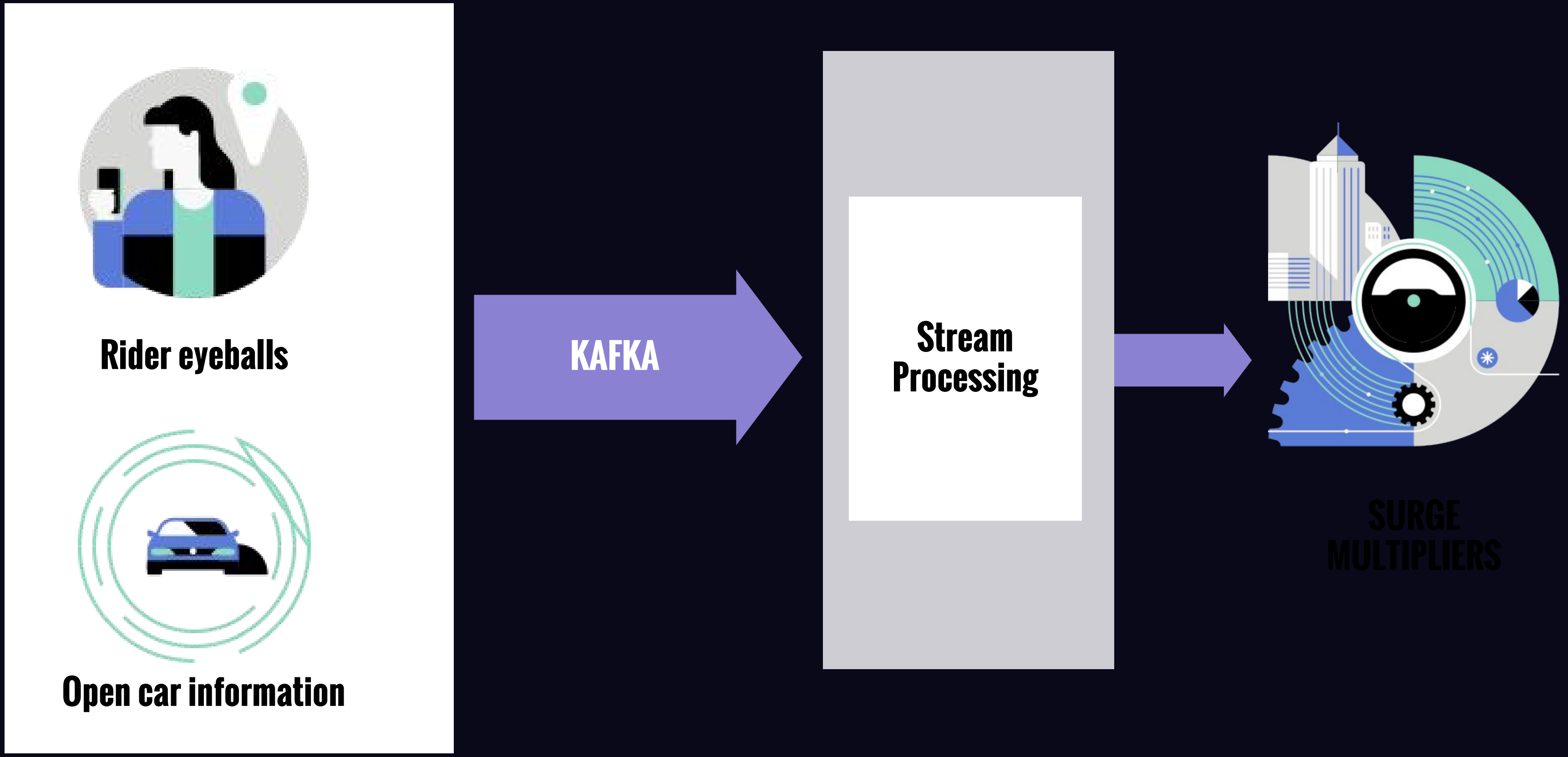
- Sr. Software Engineer, Streaming Team @ Uber
  - Streaming team supports platform for real time data analytics: Kafka, Samza, Flink, Pinot.. and plenty more
  - Focused on scaling Kafka at Uber’s pace
- Staff software Engineer @ Ebay
  - Build & scale Ebay’s cloud using openstack
- Apache Kylin: Committer, Emeritus PMC

Agenda

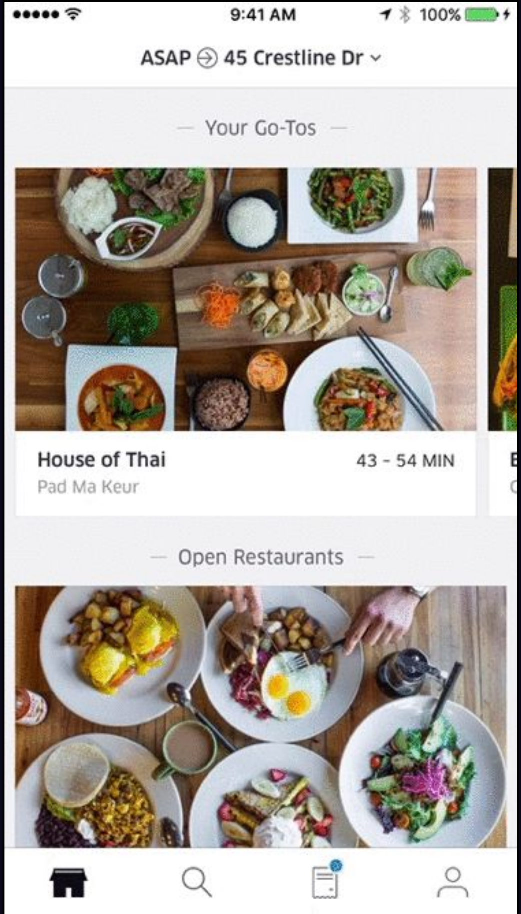
- Real time Use Cases
- Kafka Infrastructure Deep Dive
- Our own Development:
  - Rest Proxy & Clients
  - Local Agent
  - uReplicator (Mirrormaker)
  - Chaperone (Auditing)
- Operations/Tooling

Important Use Cases

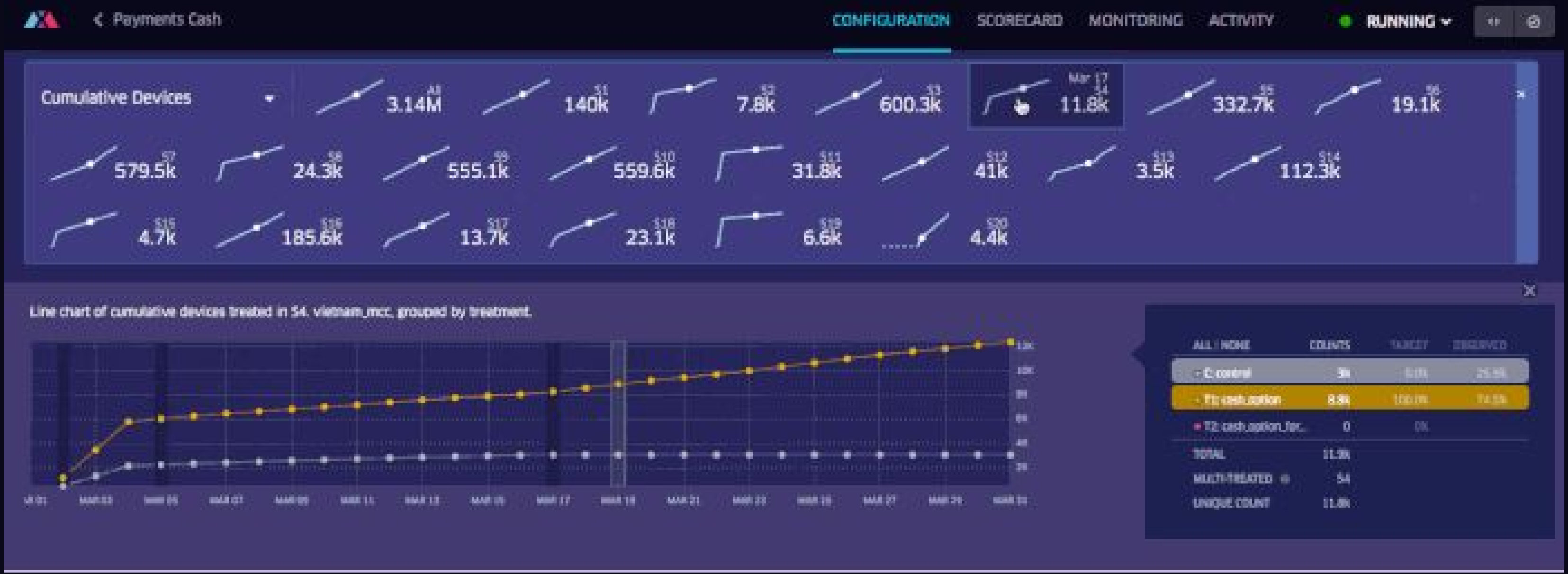
Real-time Price Surging



Real-time Machine Learning - UberEats ETD



Experimentation Platform

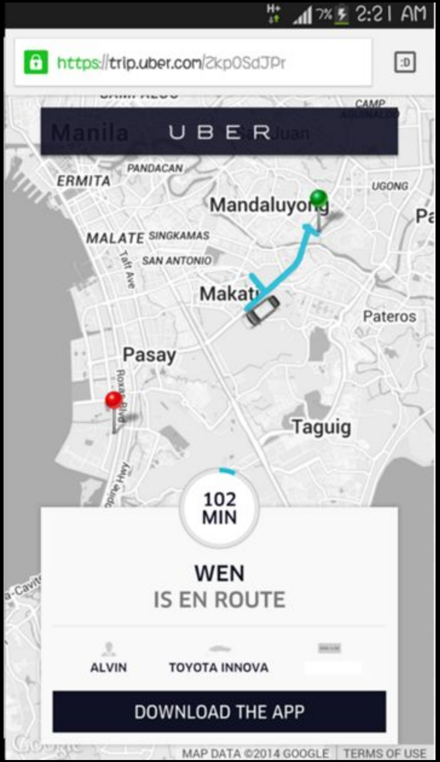


Line chart of cumulative devices treated in S4. vietnam_mcc, grouped by treatment.

ALL / NONE	COUNTS	TARGET	OBSERVED
C: control	3k	0.0%	25.5%
T1: cash_option	8.8k	100.0%	74.5%
T2: cash_option_for...	0	0%	
TOTAL	11.8k		
MULTI-TREATED	54		
UNIQUE COUNT	11.8k		

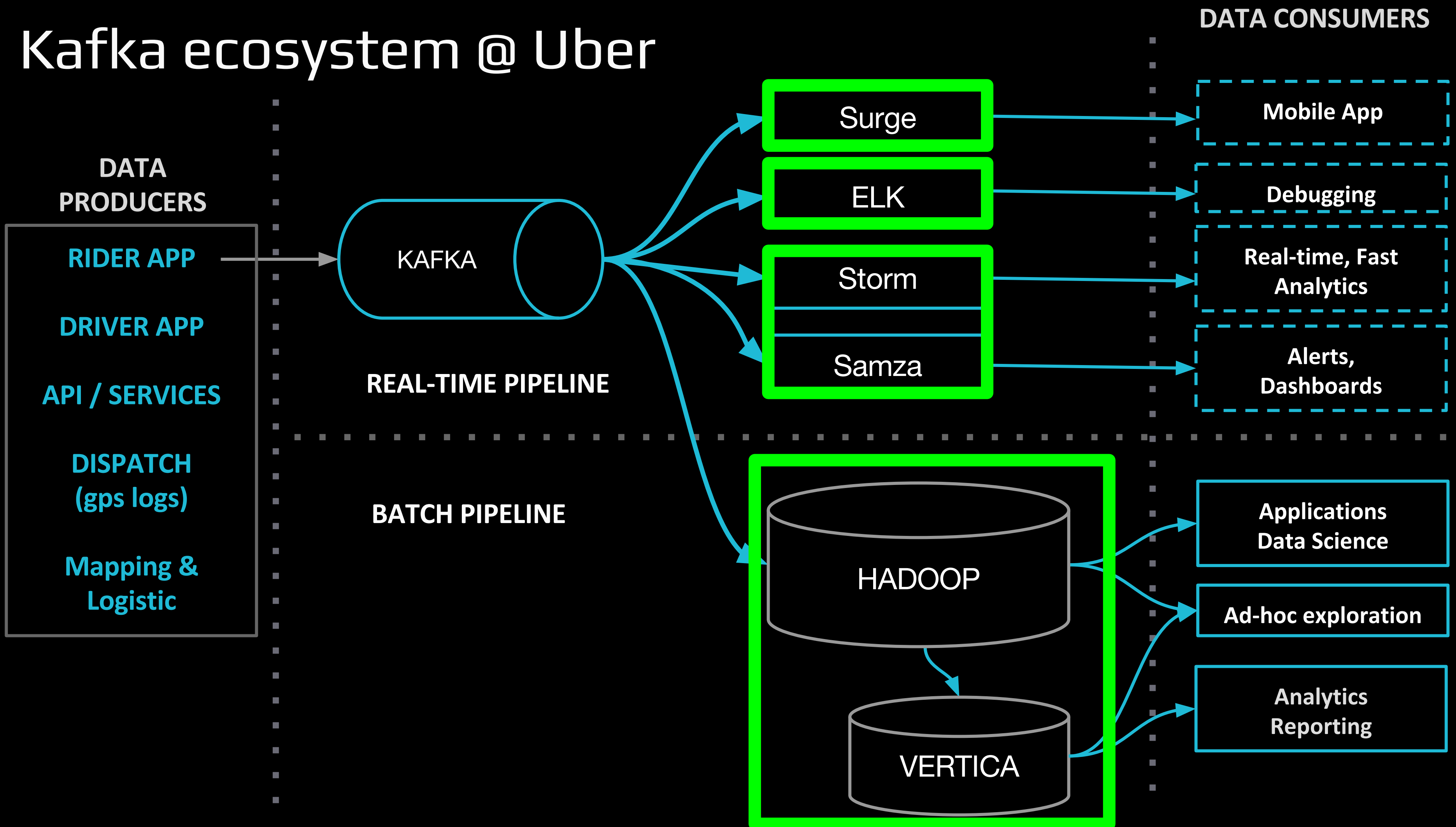
- Fraud detection
- Share my ETA

And many more ...



Apache Kafka is Uber’s Lifeline

Kafka ecosystem @ Uber



Kafka cluster stats

100s of billion

Messages/day

100s TB

bytes/day

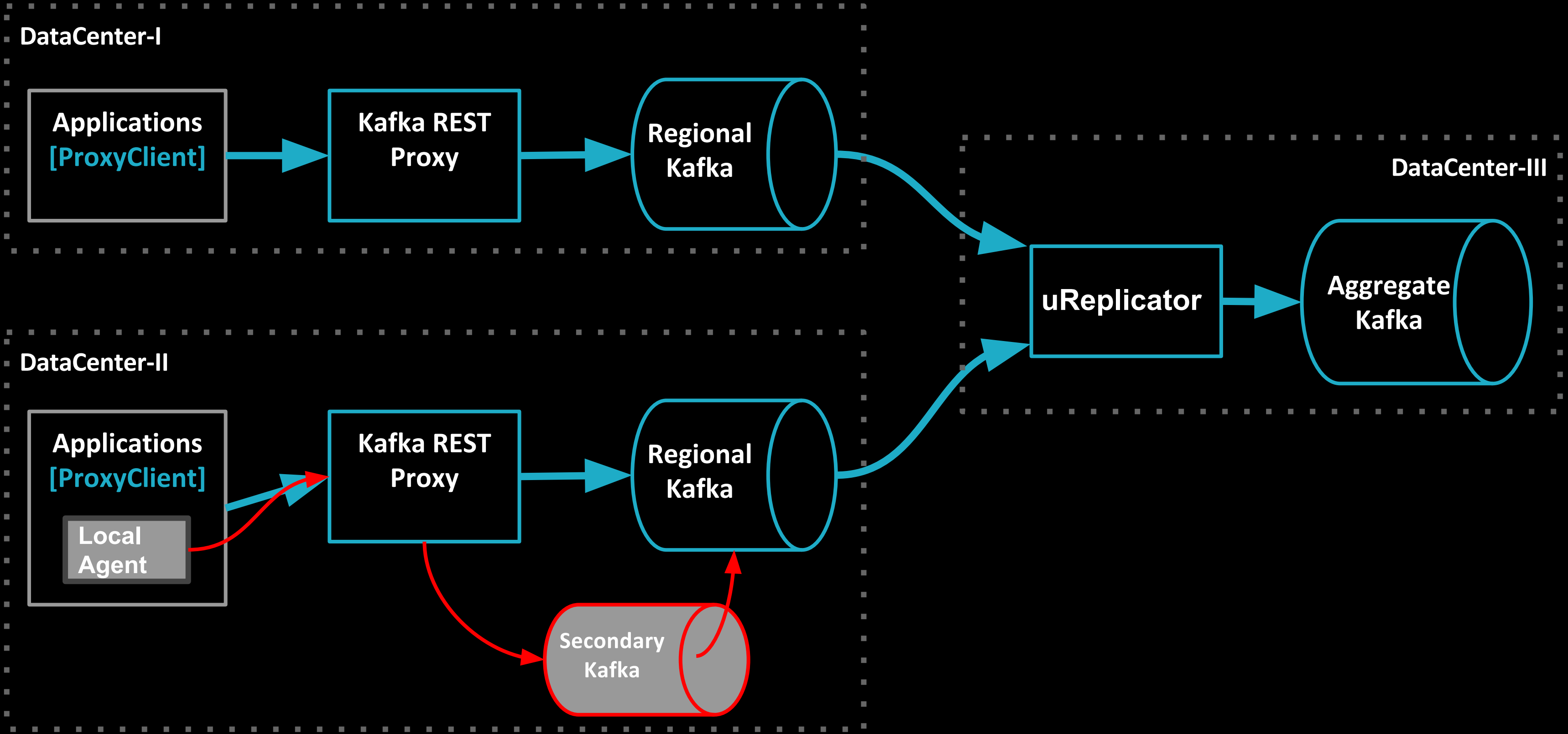
Multiple data centers

Kafka Infrastructure Deep Dive

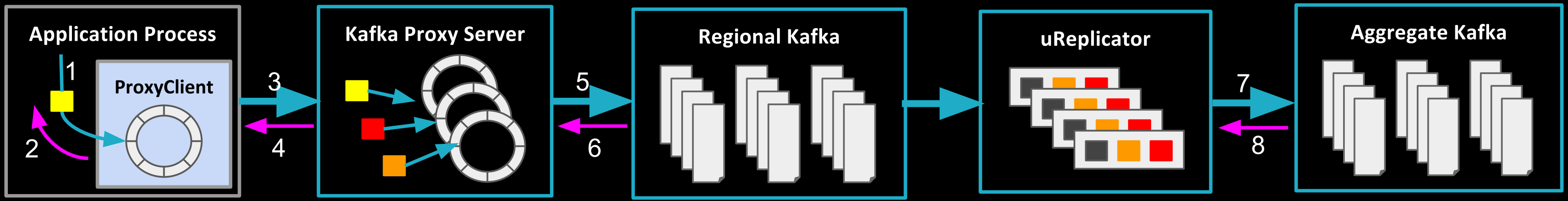
Requirements

- Scale to 100s Billions/day → 1 Trillion/day
- High Throughput (Scale: 100s TB → PB)
- Low Latency for most use cases(<5ms)
- Reliability - 99.99% (#Msgs Available /#Msgs Produced)
- Multi-Language Support
- Tens of thousands of simultaneous clients.
- Reliable data replication across DC

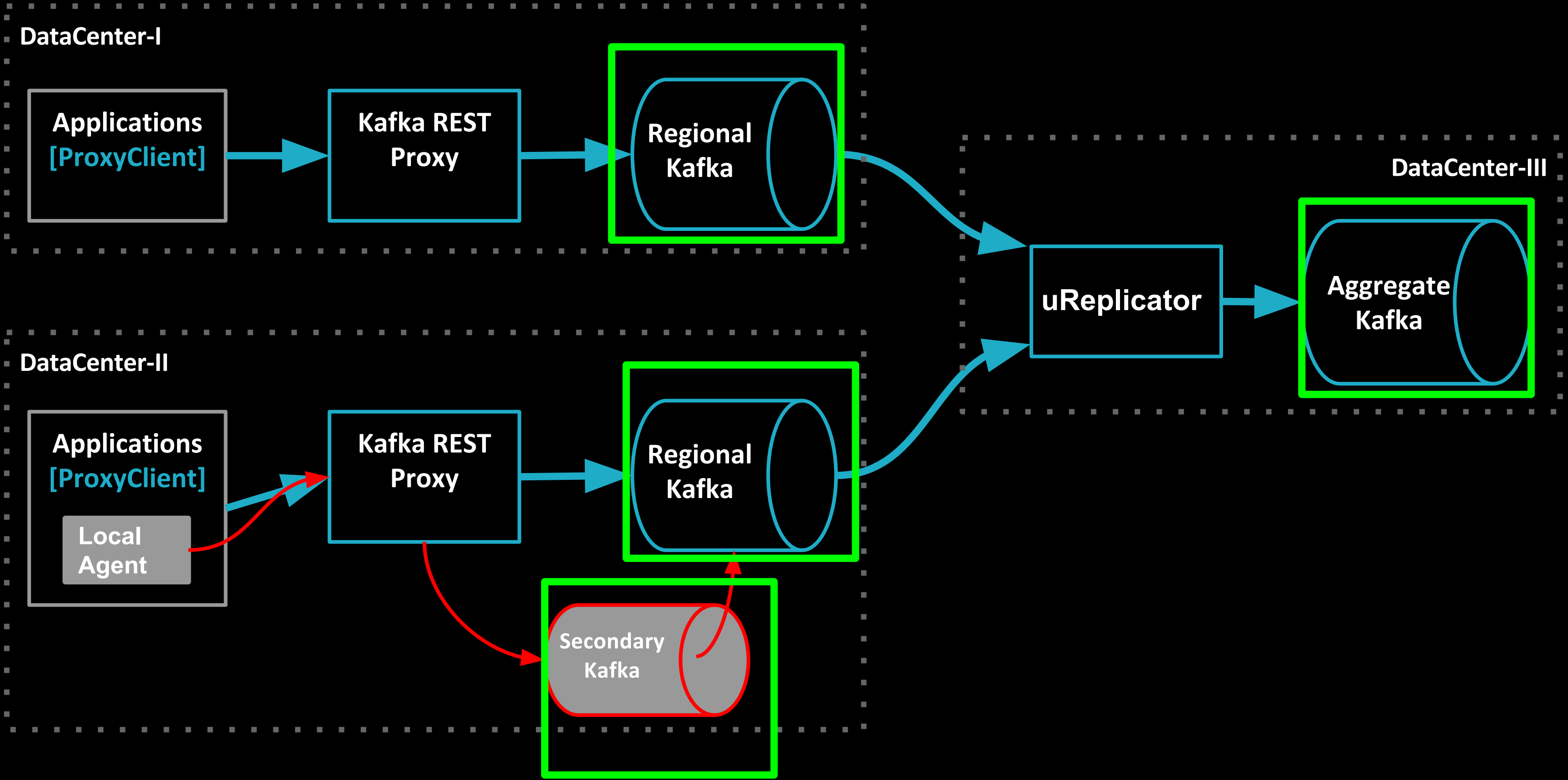
Kafka Pipeline



Kafka Pipeline: Data Flow



Kafka Clusters



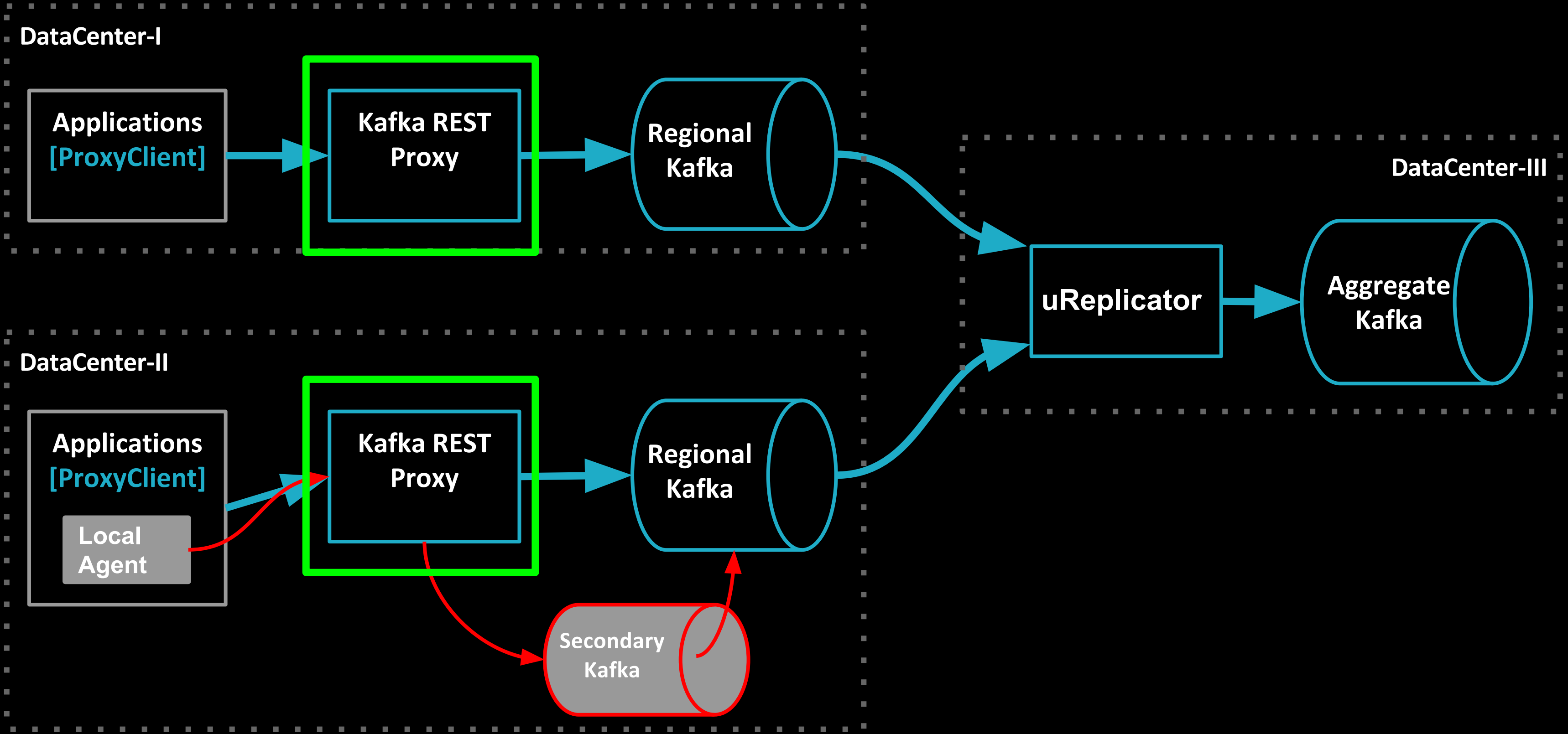
Kafka Clusters

- Use case based clusters
  - Data (async, reliable)
  - Logging (High throughput)
  - Time Sensitive (Low Latency e.g. Surge, Push notifications)
  - High Value Data (At-least once, Sync e.g. Payments)
- Secondary cluster as fallback
- Aggregate clusters for all data topics.

Kafka Clusters

- Scale to 100s Billions/day → 1 Trillion/day
- High Throughput (Scale: 100s TB → PB)
- Low Latency for most use cases(<5ms)
- Reliability - 99.99% (#Msgs Available /#Msgs Produced)
- Multi-Language Support
- Tens of thousands of simultaneous clients.
- Reliable data replication across DC

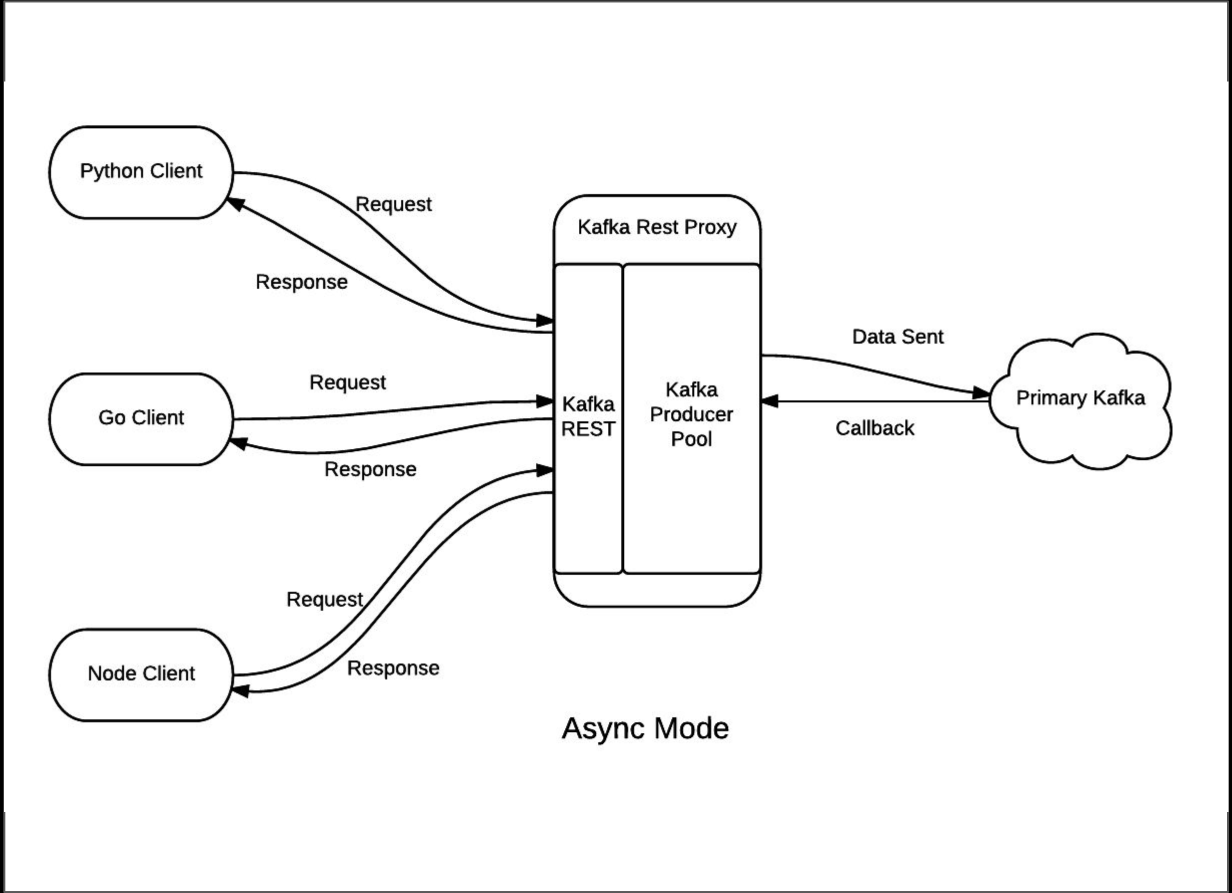
Kafka Rest Proxy



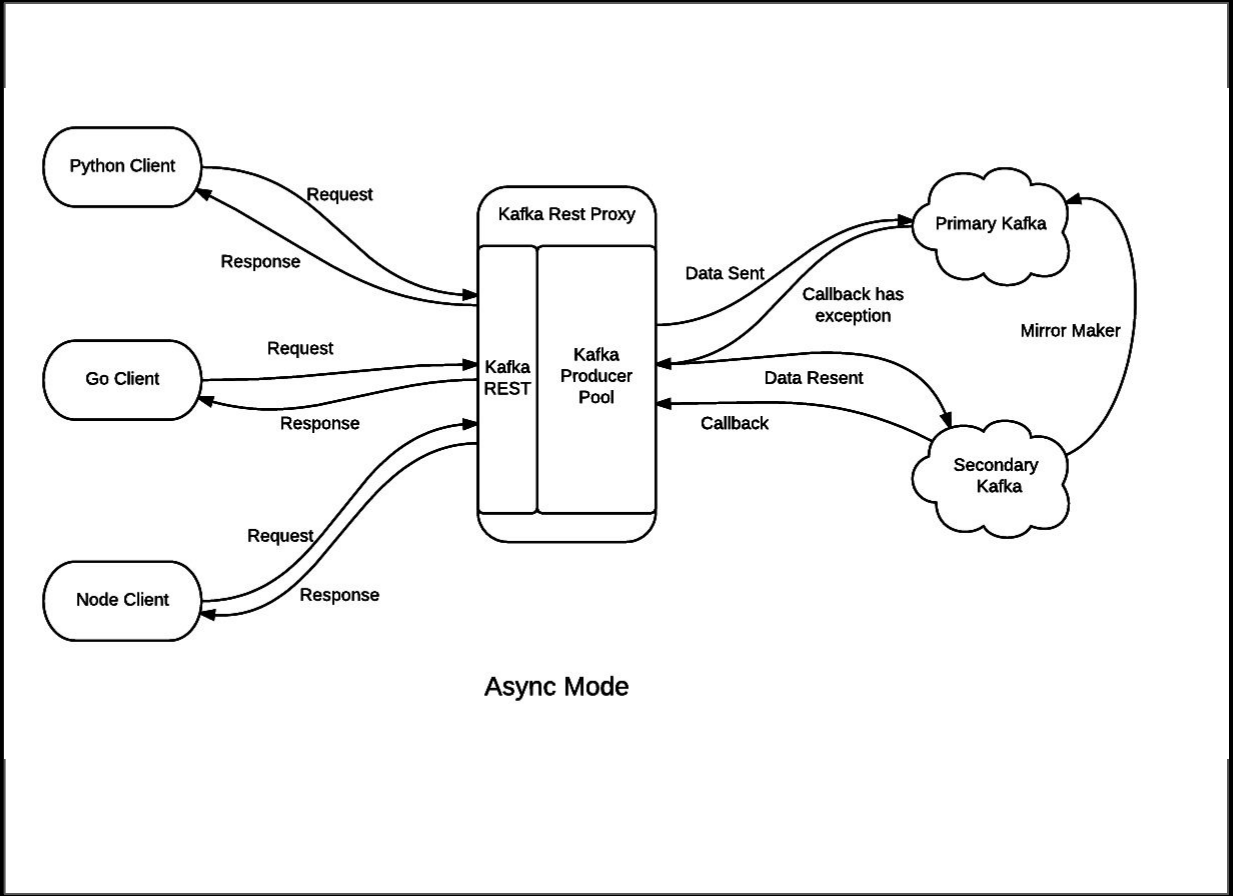
Why Kafka Rest Proxy ?

- Simplified Client API
- Multi-lang support (Java, NodeJs, Python, Golang)
- Decouple client from Kafka broker
  - Thin clients = operational ease
  - Less connections to Kafka brokers
  - Future kafka upgrade
- Enhanced Reliability
  - Primary & Secondary Kafka Clusters

Kafka Rest Proxy: Internals



Kafka Rest Proxy: Internals

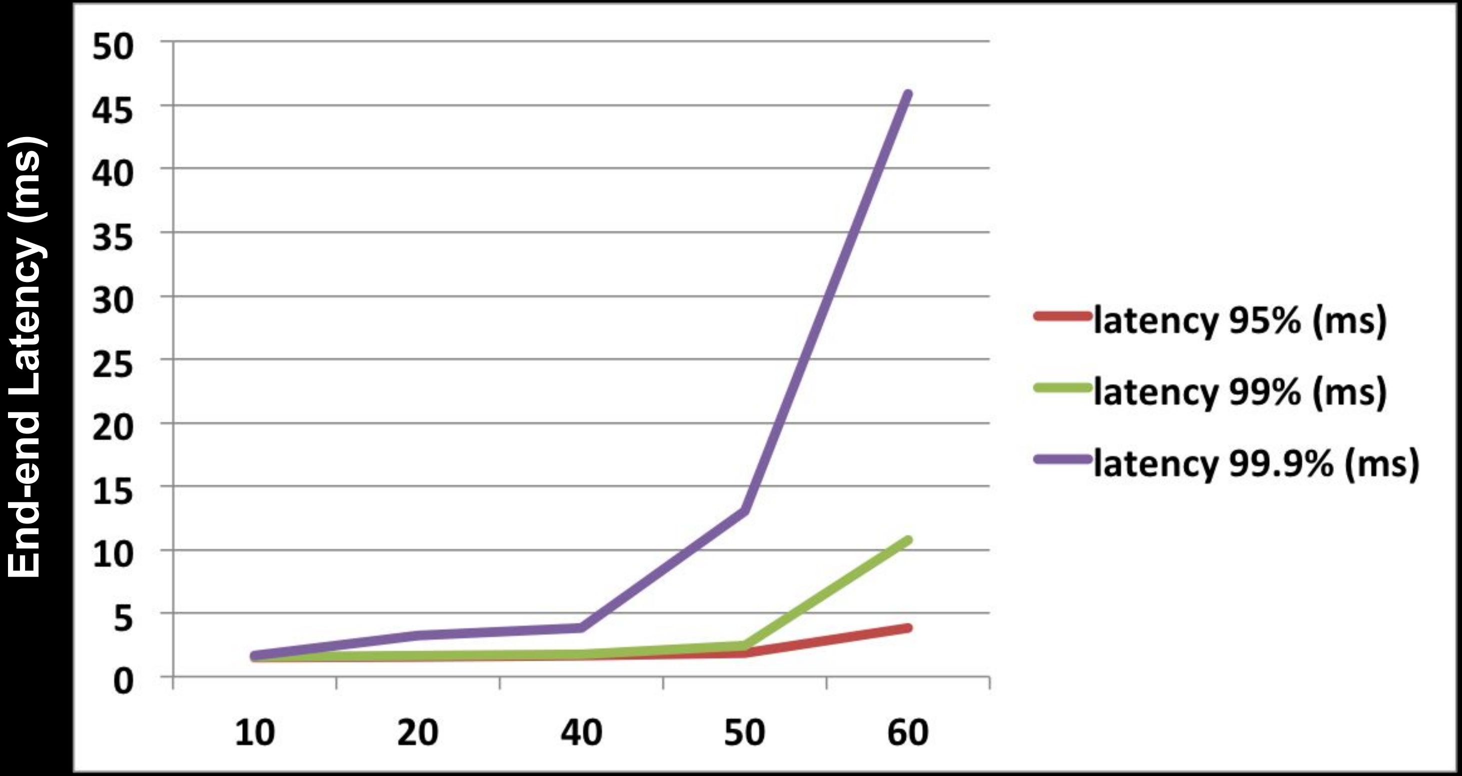


Kafka Rest Proxy: Internals

- Based on Confluent’s open sourced Rest Proxy
- Performance enhancements
  - Simple http servlets on jetty instead of Jersey
  - Optimized for binary payloads.
  - Performance increase from 7K* to 45-50K QPS/box
- Caching of topic metadata.
- Reliability improvements*
  - Support for Fallback cluster
  - Support for multiple Producers (SLA based segregation)
- Plan to contribute back to community

*Based on benchmarking & analysis done in Jun ’2015

Rest Proxy: performance (1 box)

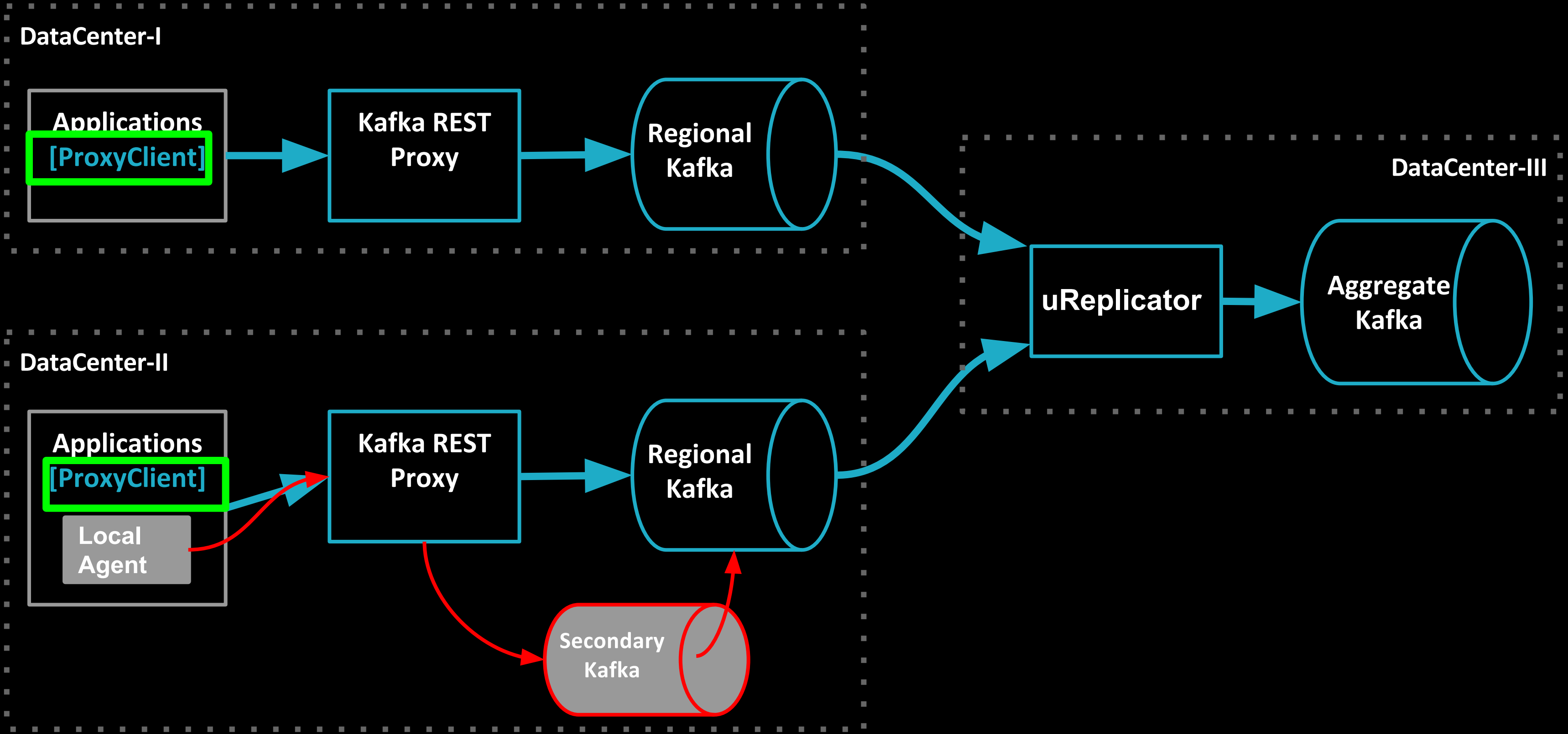


Message rate (K/second) at single node

Kafka Clusters + Rest Proxy

- Scale to 100s Billions/day → 1 Trillion/day
- High Throughput (Scale: 100s TB → PB)
- Low Latency for most use cases(<5ms)
- Reliability - 99.99% (#Msgs Available /#Msgs Produced)
- Multi-Language Support
- Tens of thousands of simultaneous clients.
- Reliable data replication across DC

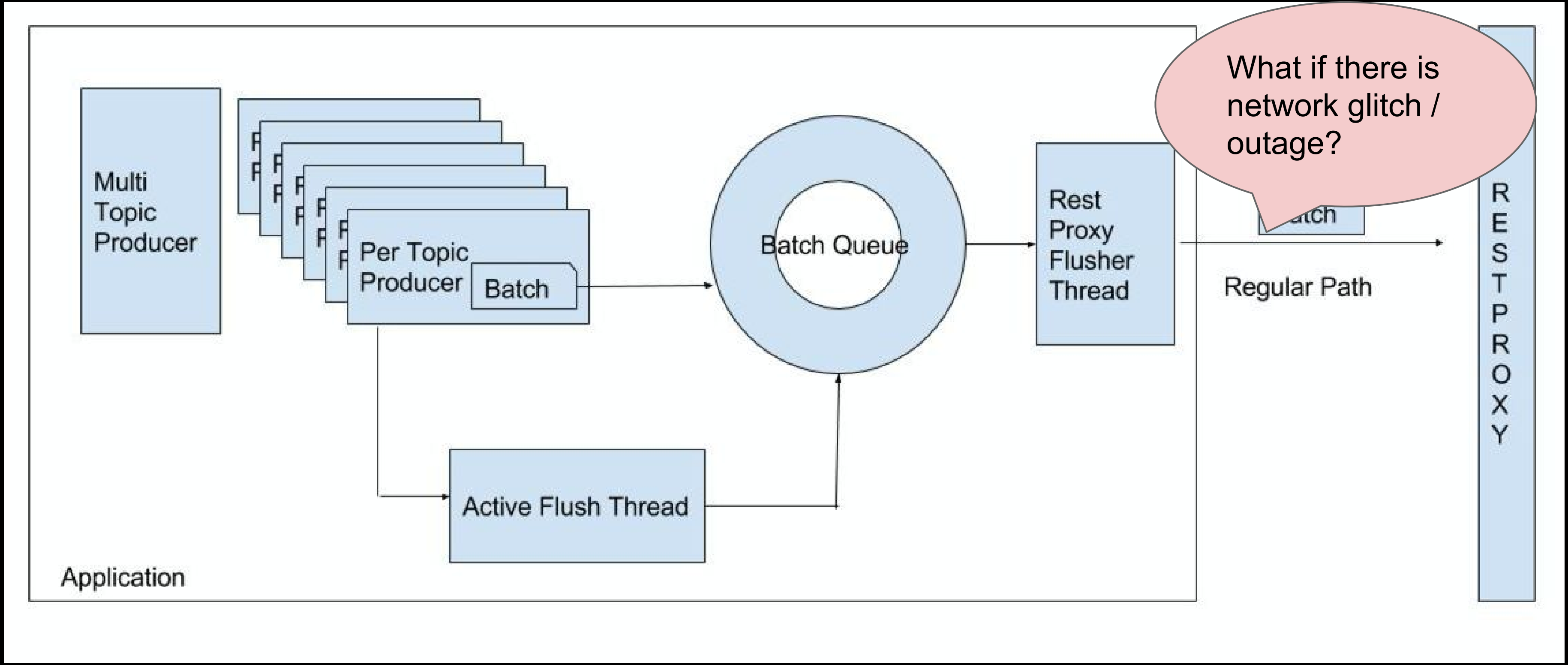
Kafka Clients



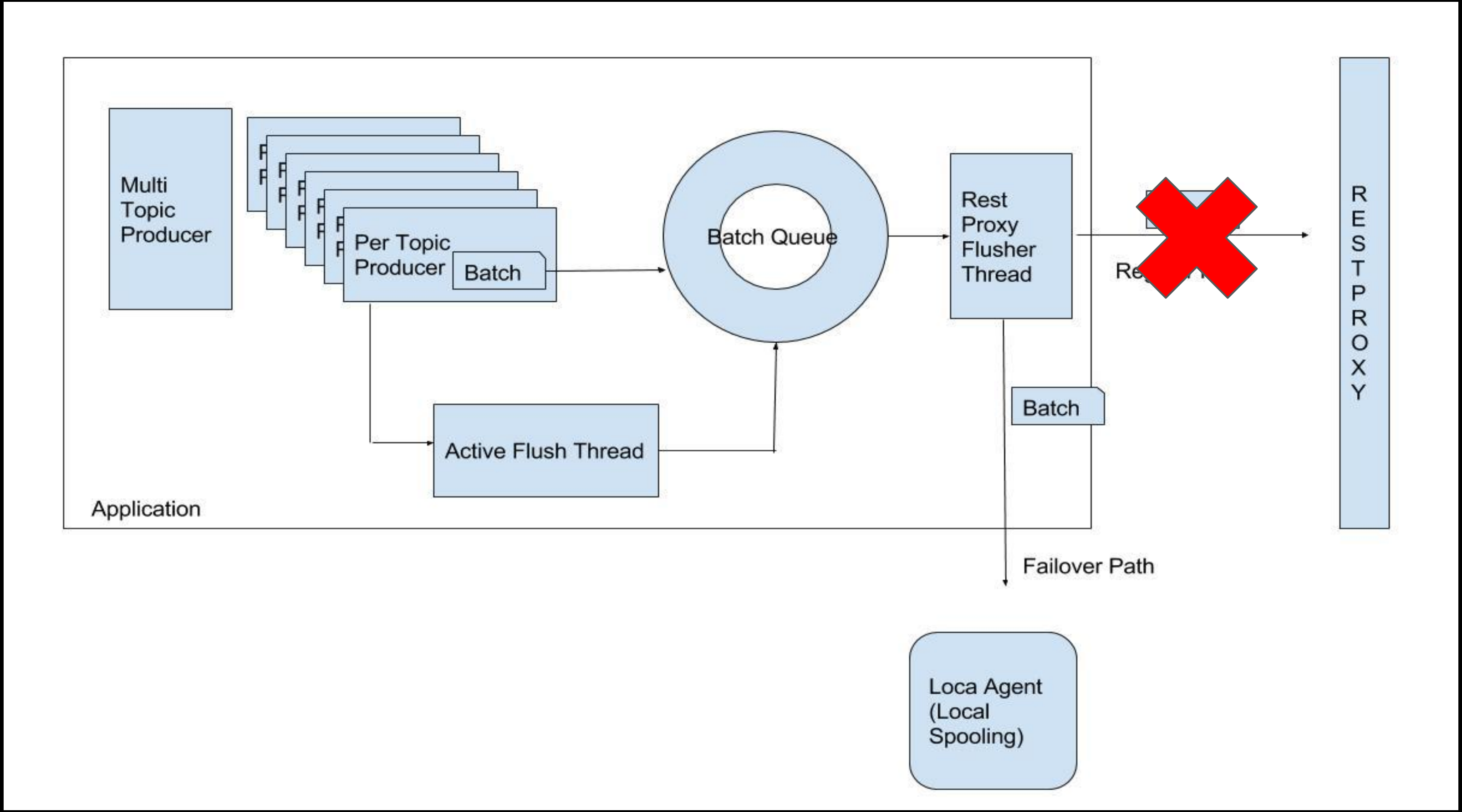
Client Libraries

- Support for multiple clusters.
- High Throughput
  - Non-blocking, async, batching
  - <1ms produce latency for clients
  - Handles Throttling/BackOff signals from Rest Proxy
- Topic Discovery
  - Discovers the kafka cluster a topic belongs
  - Able to multiplex to different kafka clusters
- Integration with Local Agent for critical data

Client Libraries



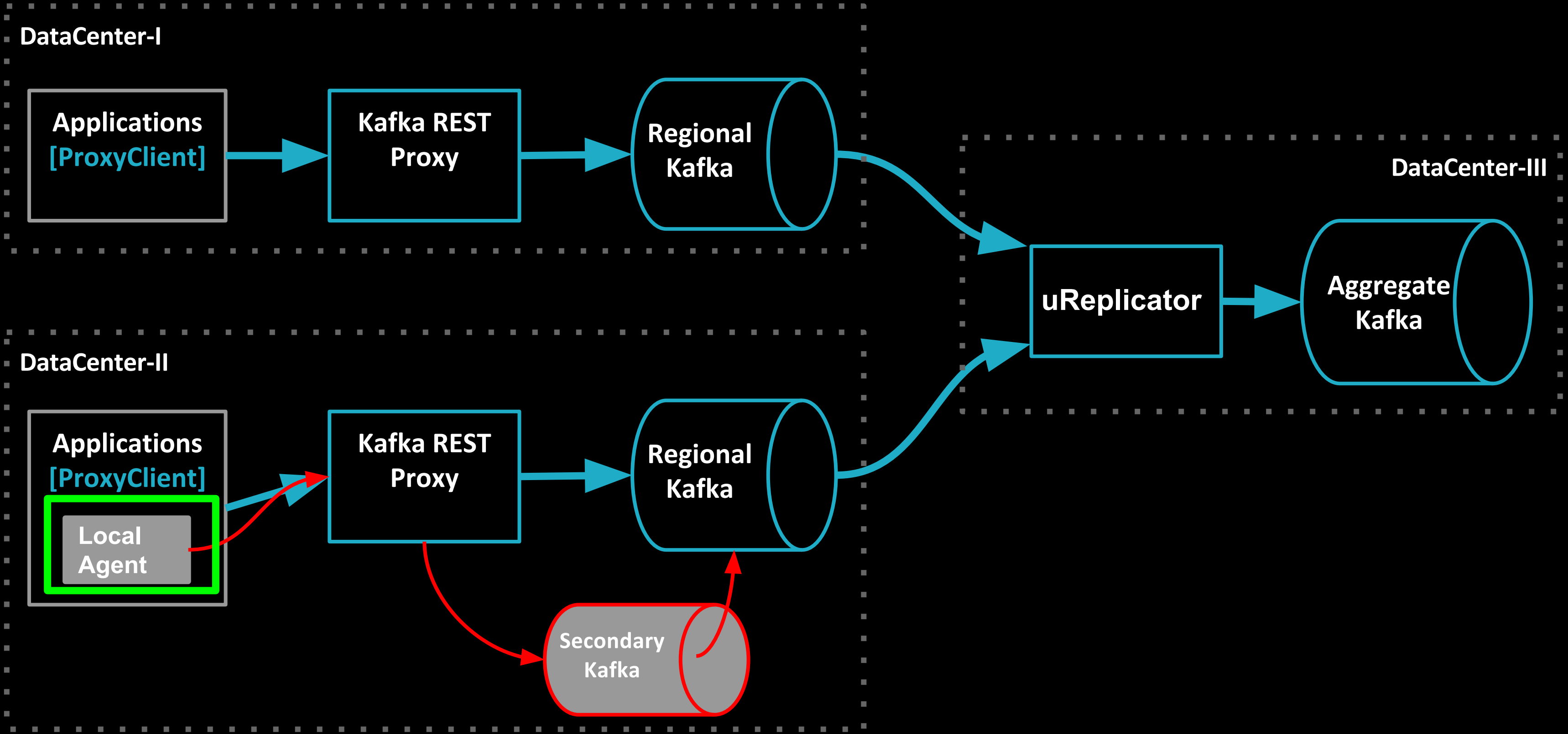
Client Libraries



Kafka Clusters + Rest Proxy + Clients

- Scale to 100s Billions/day → 1 Trillion/day
- High Throughput (Scale: 100s TB → PB)
- Low Latency for most use cases(<5ms)
- Reliability - 99.99% (#Msgs Available /#Msgs Produced)
- Multi-Language Support
- Tens of thousands of simultaneous clients.
- Reliable data replication across DC

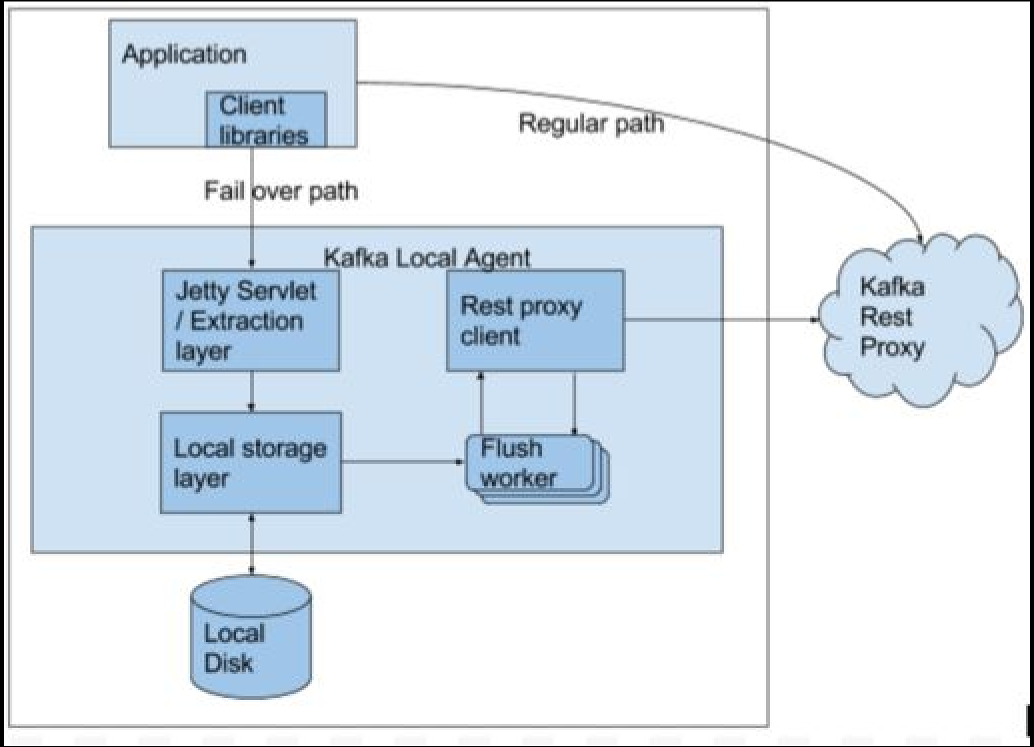
Local Agent


DataCenter-I
Applications
[ProxyClient]
Kafka REST Proxy
Regional Kafka
DataCenter-II
Applications
[ProxyClient]
Local Agent
Kafka REST Proxy
Regional Kafka
Secondary Kafka
DataCenter-III
uReplicator
Aggregate Kafka

Local Agent

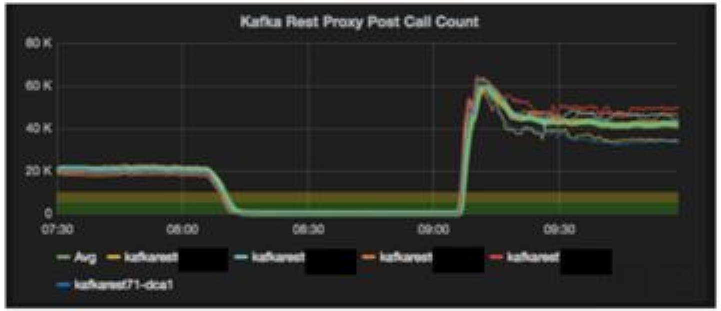
- Local spooling in case of downstream outage/backpressure
- Backfills at the controlled rate to avoid hammering infrastructure recovering from outage
- Implementation:
  - Reuses code from rest-proxy and kafka’s log module.
  - Appends all topics to same file for high throughput.

Local Agent Architecture

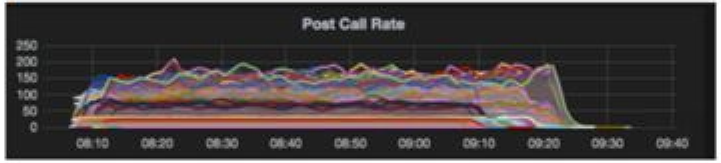


Local Agent in Action

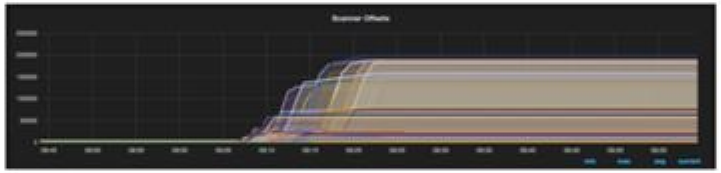
#1: Rest proxy data incoming traffic coming down to 0 around 8:10. Recovery around 9:10 with a spike from client libraries buffers



#2: Traffic starts moving to local agent in outage window



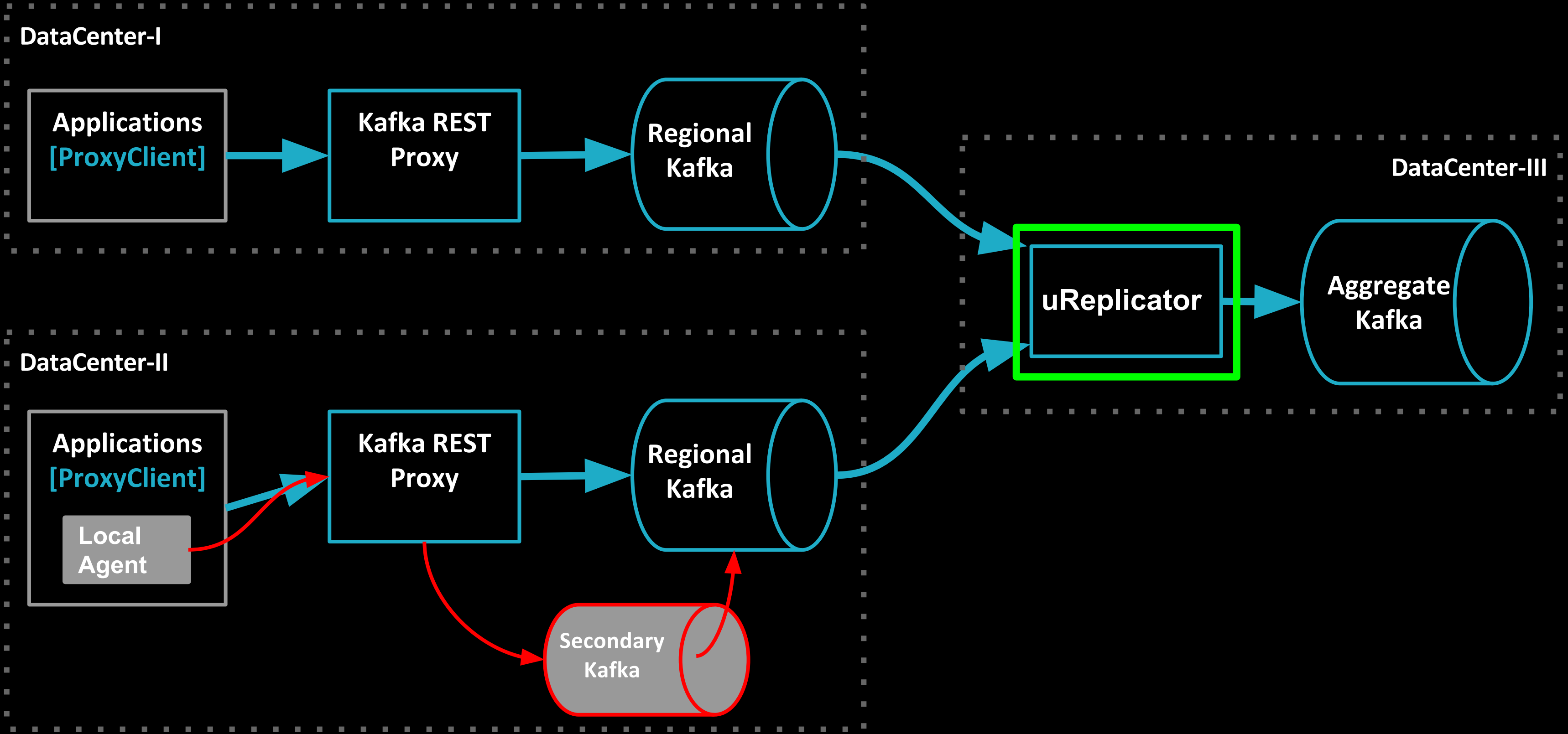
#3: Local agent producing to Rest proxy after recovery



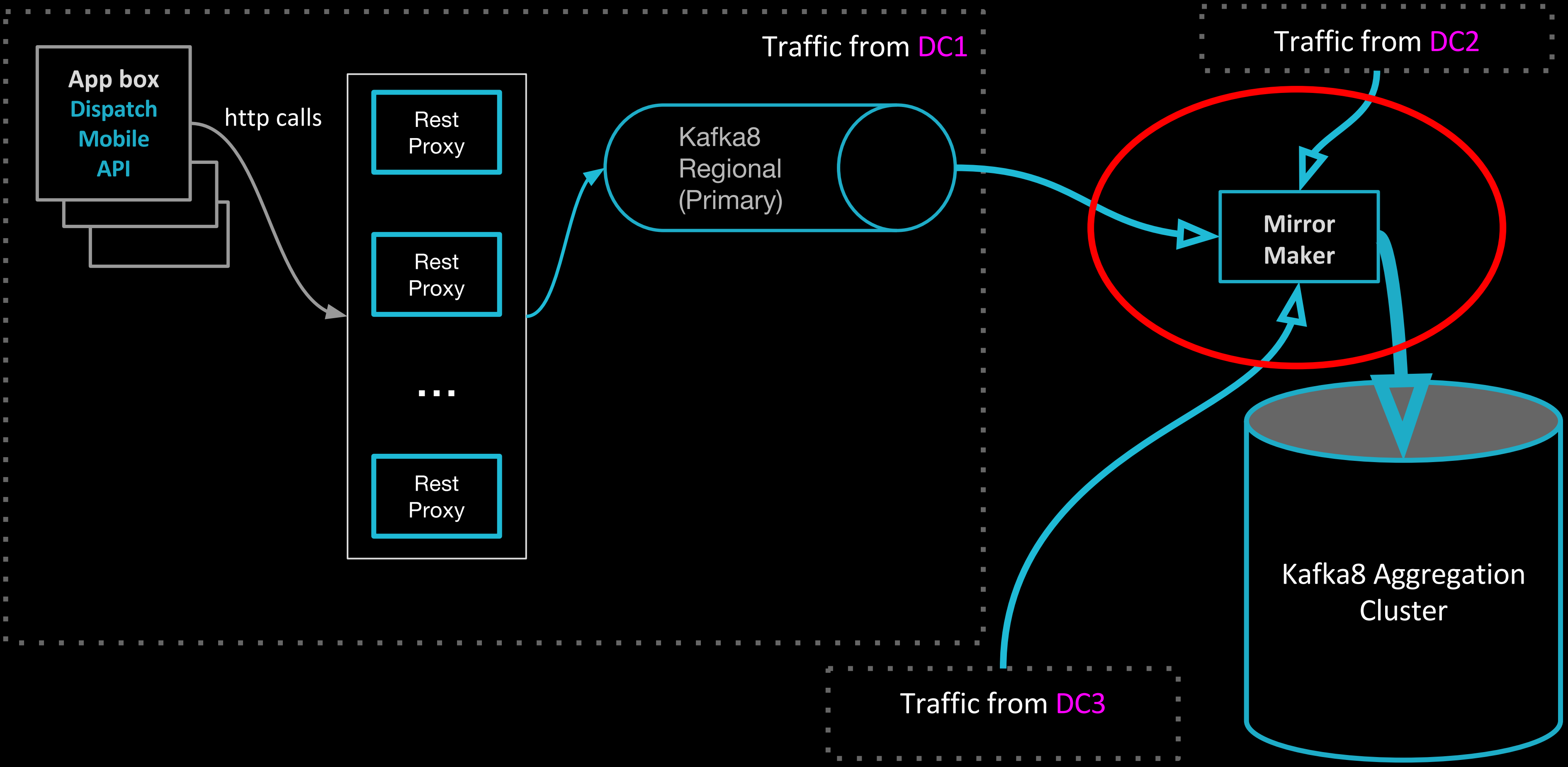
Kafka Clusters + Rest Proxy + Clients + Local Agent

- Scale to 100s Billions/day → 1 Trillion/day
- High Throughput (Scale: 100s TB → PB)
- Low Latency for most use cases(<5ms)
- Reliability - 99.99% (#Msgs Available /#Msgs Produced)
- Multi-Language Support
- Tens of thousands of simultaneous clients.
- Reliable data replication across DC

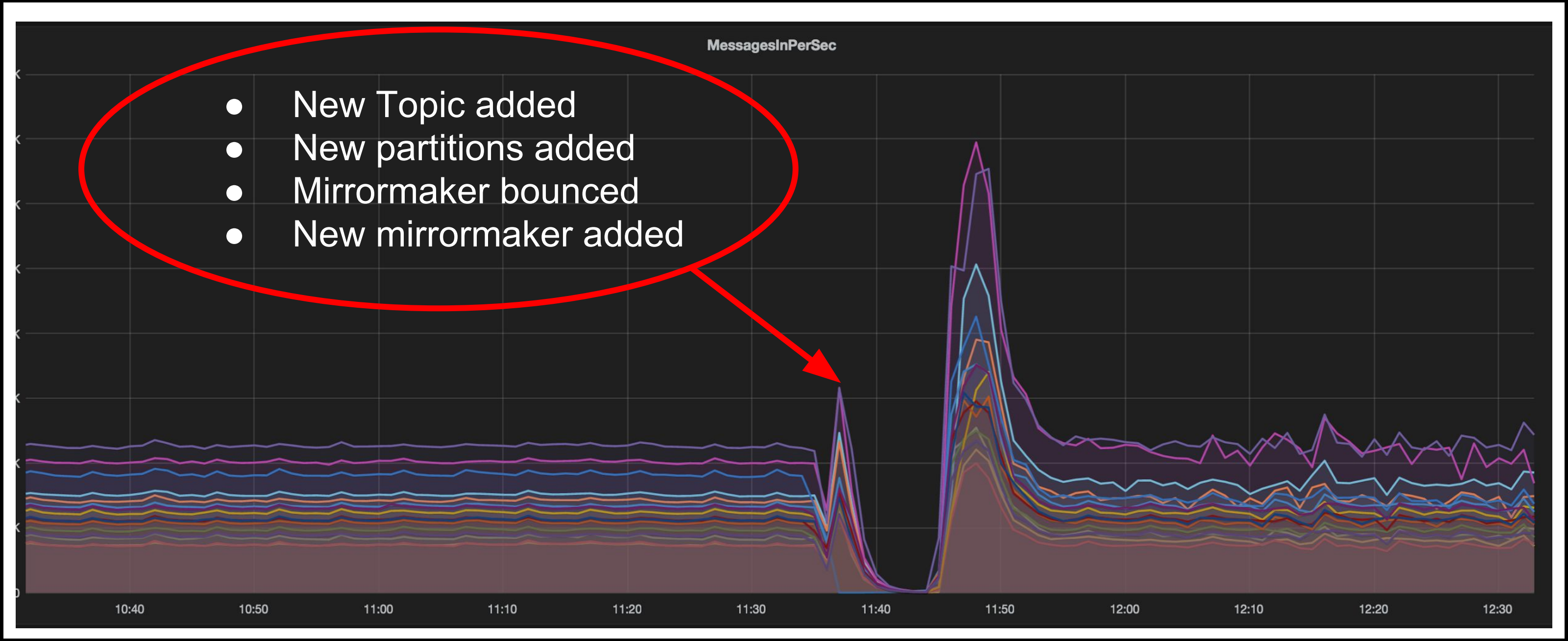
uReplicator



Multi-DC data flow

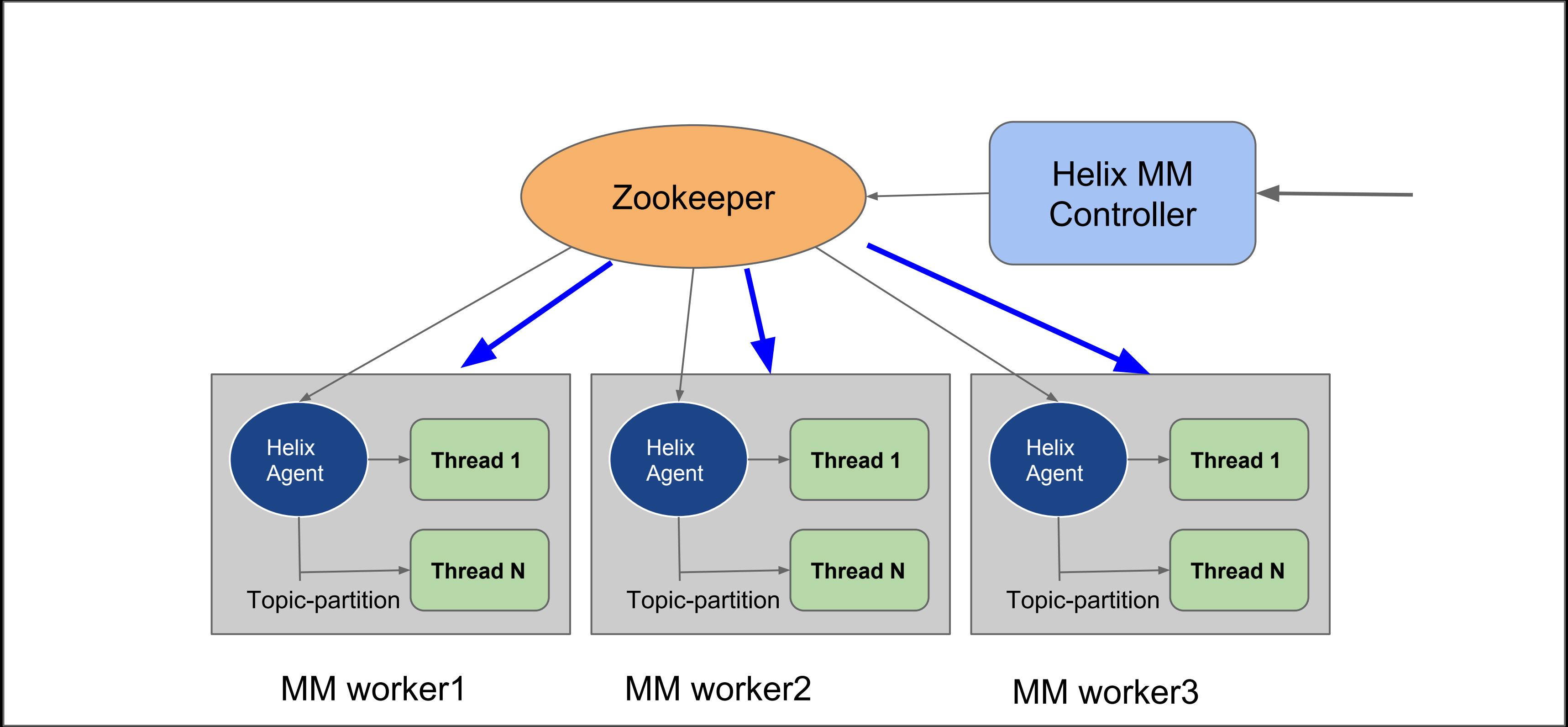


Mirrormaker : existing problems

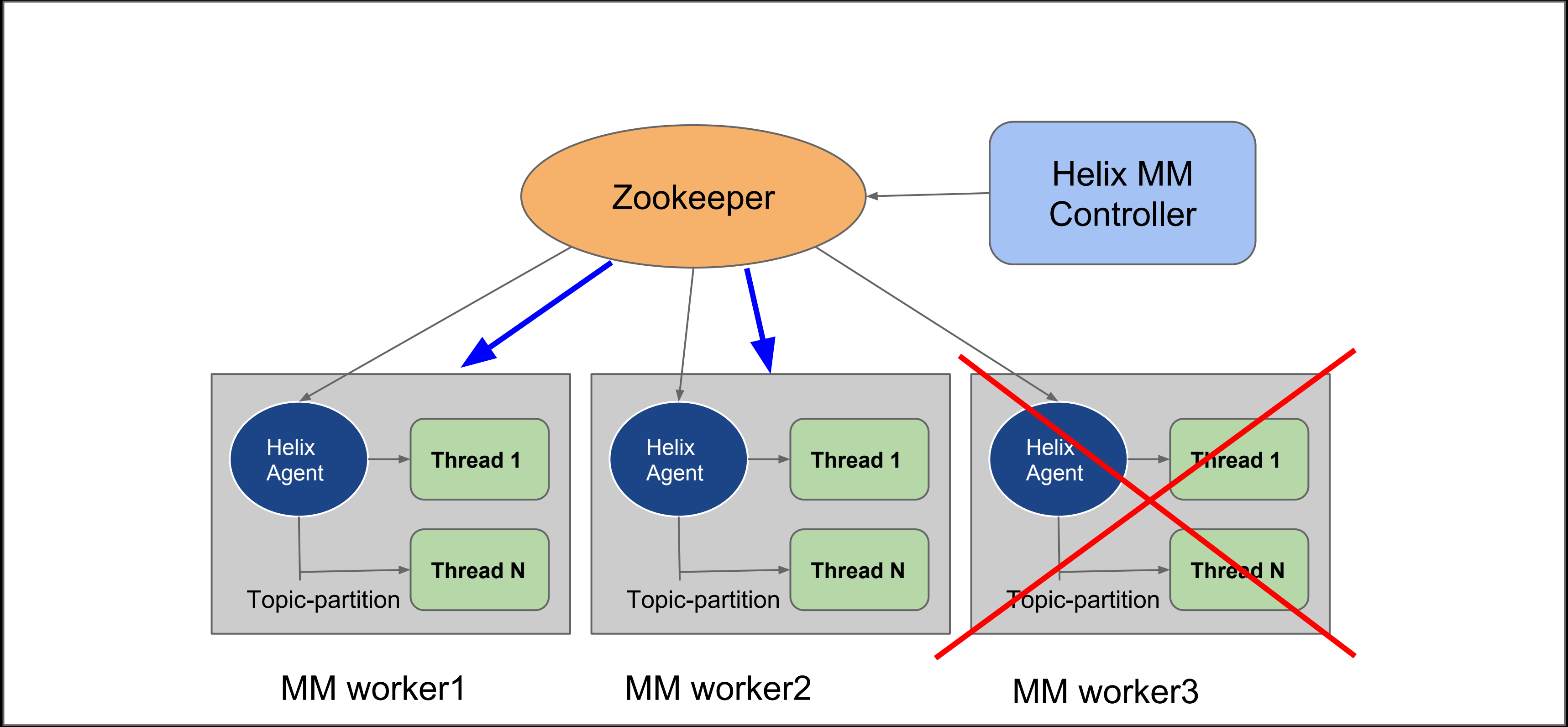


- New Topic added
- New partitions added
- Mirrormaker bounced
- New mirrormaker added

uReplicator: In-house solution



uReplicator



Kafka Clusters + Rest Proxy + Clients + Local Agent

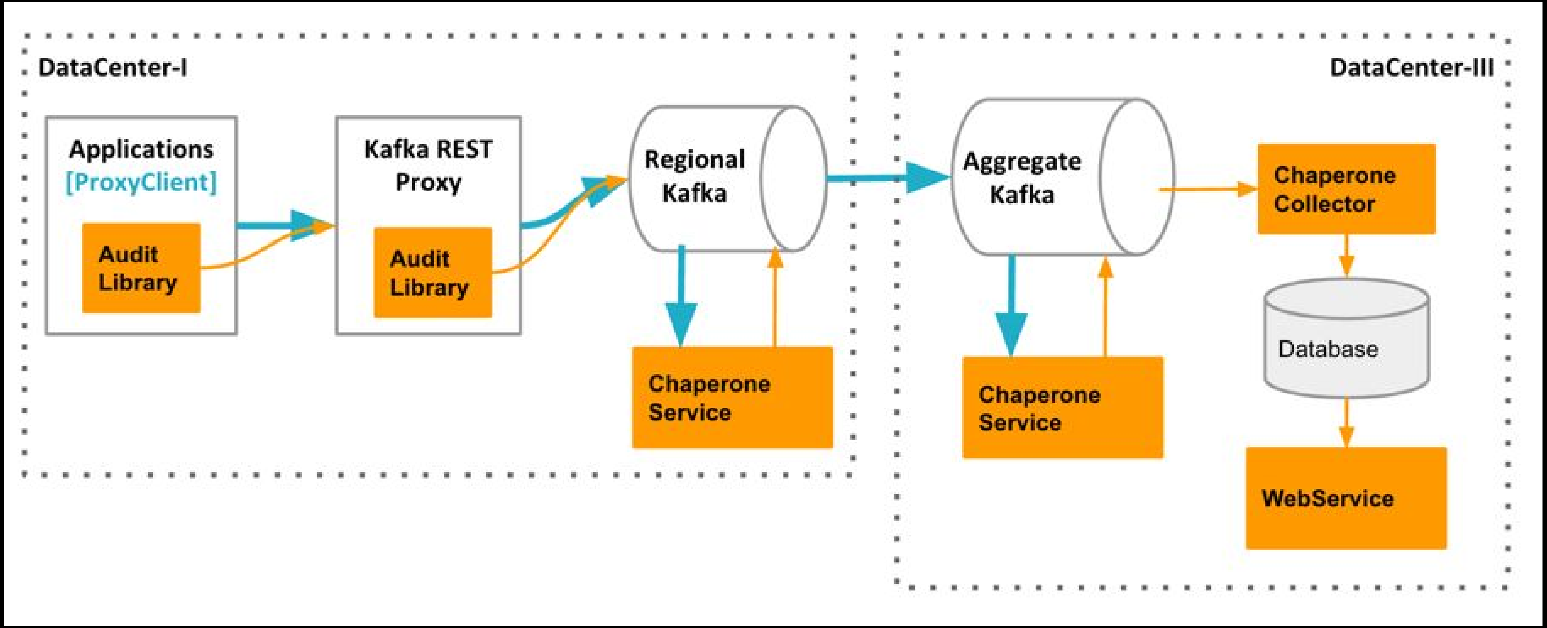
- Scale to 100s Billions/day → 1 Trillion/day
- High Throughput (Scale: 100s TB → PB)
- Low Latency for most use cases(<5ms)
- Reliability - 99.99% (#Msgs Available /#Msgs Produced)
- Multi-Language Support
- Tens of thousands of simultaneous clients.
- Reliable data replication across DC

uReplicator

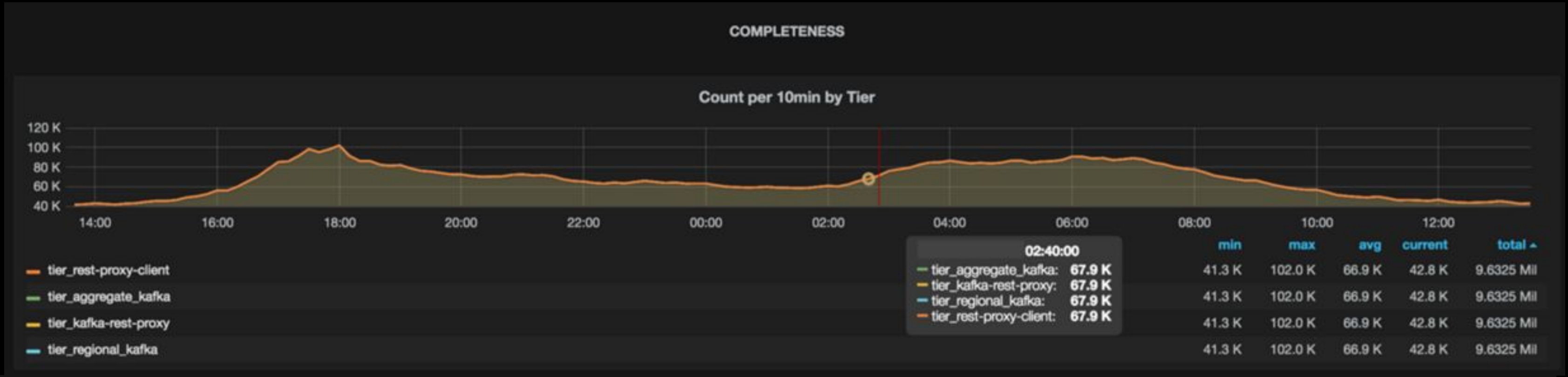
- Running in production for 1+ year
- Open sourced: https://github.com/uber/uReplicator
- Blog: https://eng.uber.com/ureplicator/

Chaperone - E2E Auditing

Chaperone Architecture



Chaperone : Track counts

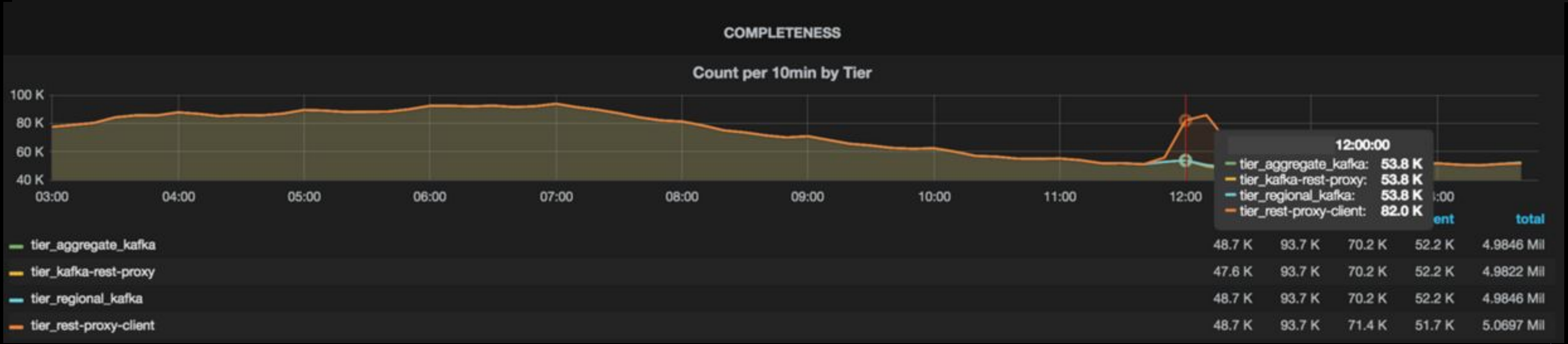


COMPLETENESS

Count per 10min by Tier

```
02:40:00
— tier_aggregate_kafka:   67.9 K
— tier_kafka-rest-proxy:  67.9 K
— tier_regional_kafka:    67.9 K
— tier_rest-proxy-client: 67.9 K
```

	min	max	avg	current	total ▴
tier_rest-proxy-client	41.3 K	102.0 K	66.9 K	42.8 K	9.6325 Mil
tier_aggregate_kafka	41.3 K	102.0 K	66.9 K	42.8 K	9.6325 Mil
tier_kafka-rest-proxy	41.3 K	102.0 K	66.9 K	42.8 K	9.6325 Mil
tier_regional_kafka	41.3 K	102.0 K	66.9 K	42.8 K	9.6325 Mil



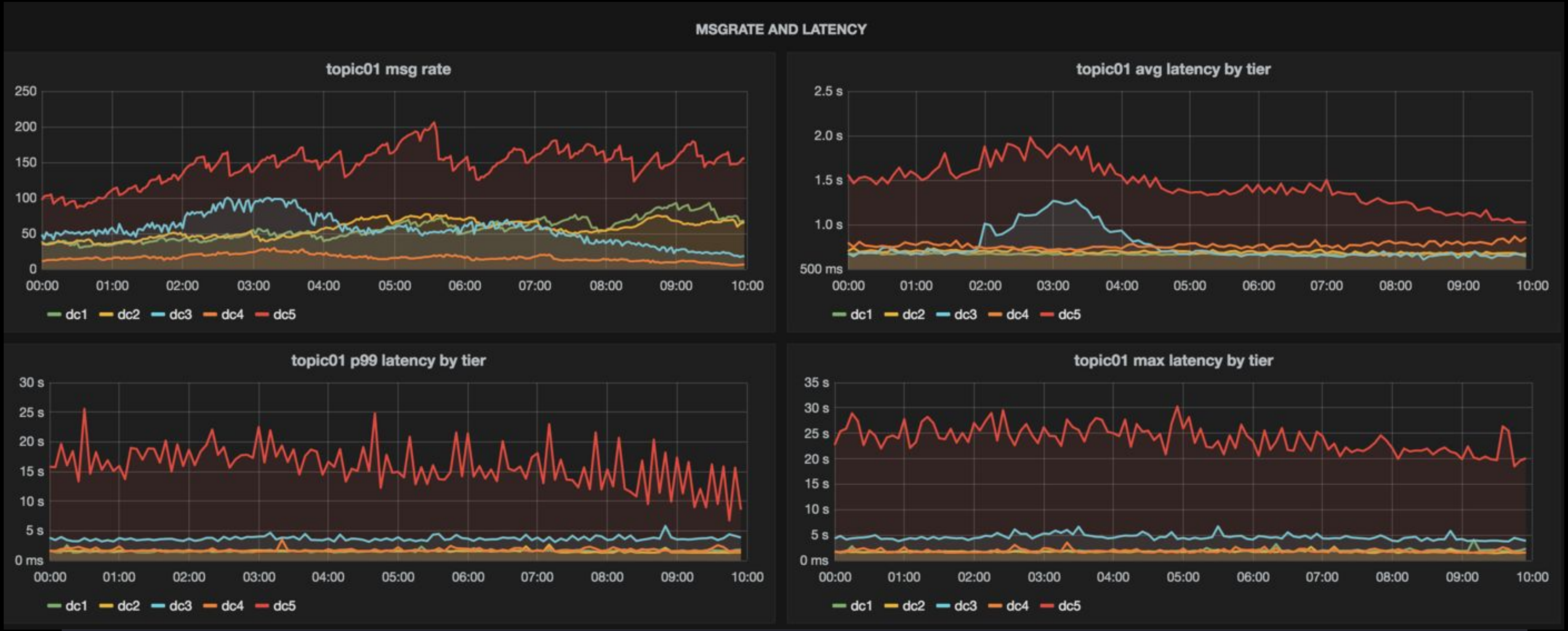
COMPLETENESS

Count per 10min by Tier

```
12:00:00
— tier_aggregate_kafka:   53.8 K
— tier_kafka-rest-proxy:  53.8 K
— tier_regional_kafka:    53.8 K
— tier_rest-proxy-client: 82.0 K
```

				ent	total
tier_aggregate_kafka	48.7 K	93.7 K	70.2 K	52.2 K	4.9846 Mil
tier_kafka-rest-proxy	47.6 K	93.7 K	70.2 K	52.2 K	4.9822 Mil
tier_regional_kafka	48.7 K	93.7 K	70.2 K	52.2 K	4.9846 Mil
tier_rest-proxy-client	48.7 K	93.7 K	71.4 K	51.7 K	5.0697 Mil

Chaperone : Track Latency

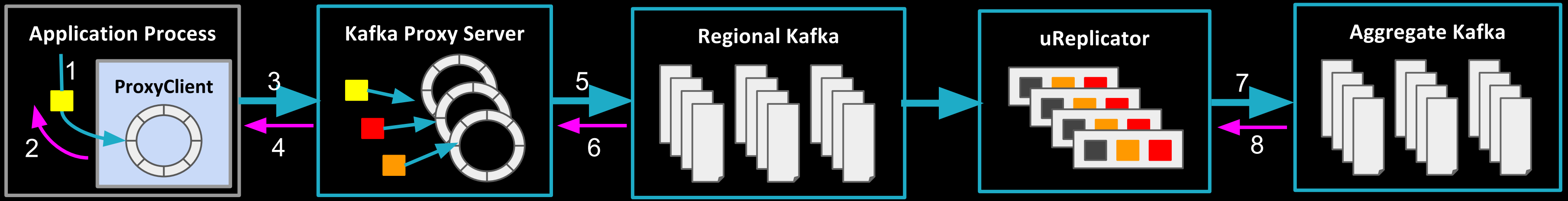


Chaperone

- Running in production for 1+ year
- Planning to open source in ~2 Weeks

At-least Once Kafka

Why do we need it?



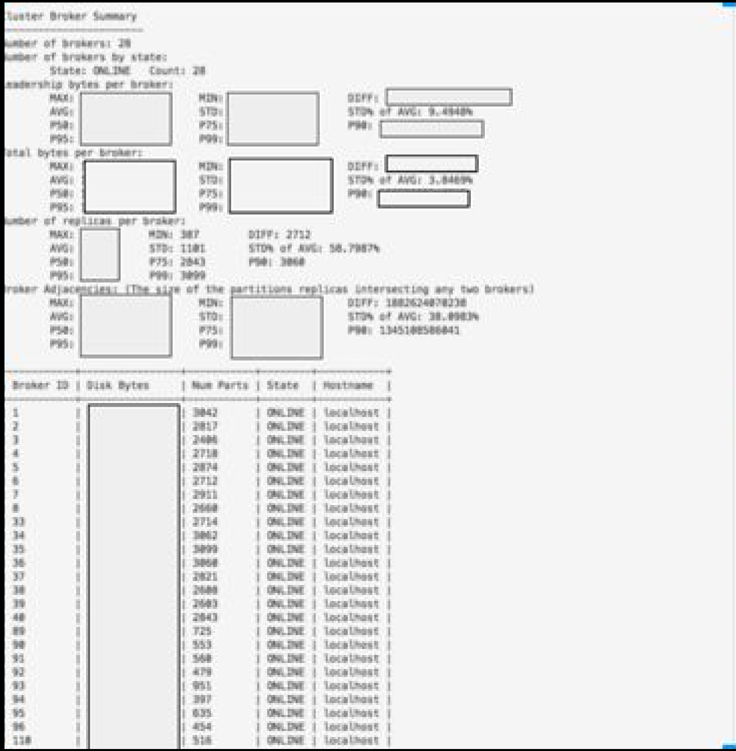
- Most of infrastructure tuned for high throughput
  - Batching at each stage
  - Ack before produce (ack'ed != committed)
- Single node failure in any stage leads to data loss
- Need a reliable pipeline for High Value Data e.g. Payments

How did we achieve it?

- Brokers:
  - min.insync.replicas=2, can only torrent one node failure
  - unclean.leader.election= false, need to wait until the old leader comes back
- Rest Proxy:
  - Partition Failover
- Improved Operations:
  - Replication throttling, to reduce impact of node bootstrap
  - Prevent catching up nodes to become ISR

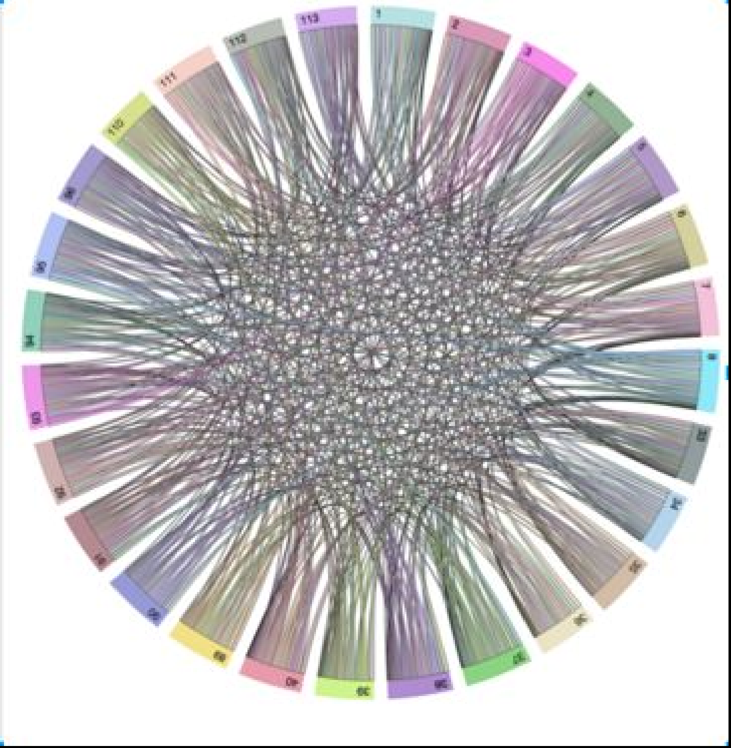
Operations/Tooling

Partition Rebalancing



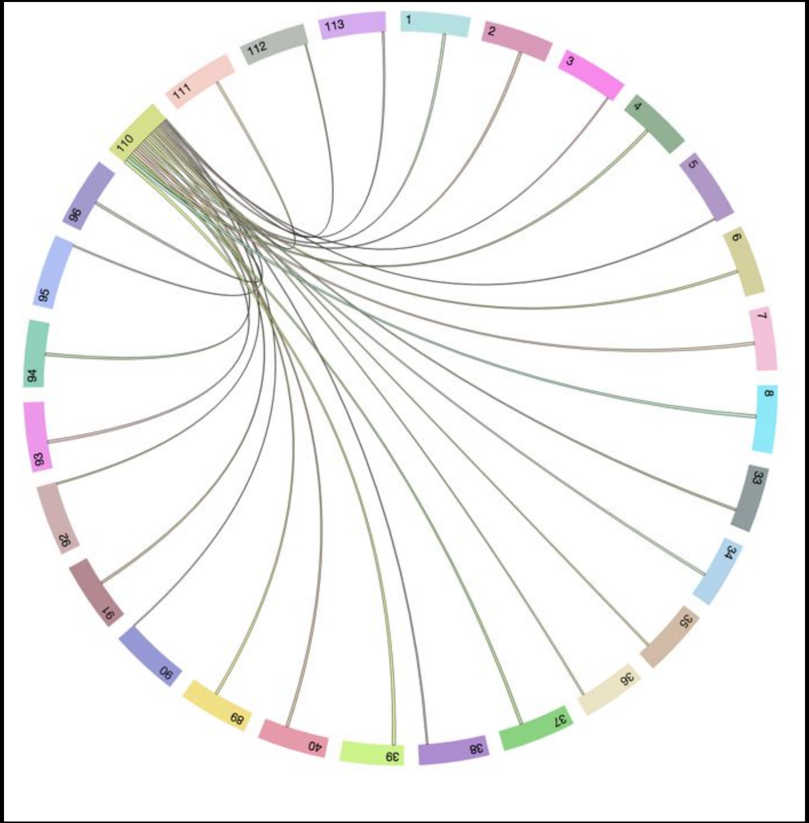
```
luster Broker Summary
---------------------
umber of brokers: 28
umber of brokers by state:
        State: ONLINE    Count: 28
eadership bytes per broker:
        MAX: [redacted]      MIN: [redacted]      DIFF: [redacted]
        AVG:                 STD:                 STD% of AVG: 9.4940%
        P50:                 P75:                 P90: [redacted]
        P95:                 P99:
otal bytes per broker:
        MAX: [redacted]      MIN: [redacted]      DIFF: [redacted]
        AVG:                 STD:                 STD% of AVG: 3.8469%
        P50:                 P75:                 P90: [redacted]
        P95:                 P99:
umber of replicas per broker:
        MAX: [redacted]  MIN: 387        DIFF: 2712
        AVG:             STD: 1101       STD% of AVG: 58.7987%
        P50:             P75: 2843       P90: 3060
        P95:             P99: 3099
roker Adjacencies: (The size of the partitions replicas intersecting any two brokers)
        MAX: [redacted]      MIN: [redacted]      DIFF: 1882624070230
        AVG:                 STD:                 STD% of AVG: 38.0983%
        P50:                 P75:                 P90: 1345100580041
        P95:                 P99:

+----------+----------------+-----------+--------+-----------+
| Broker ID | Disk Bytes     | Num Parts | State  | Hostname  |
+----------+----------------+-----------+--------+-----------+
 1          | [redacted]     | 3042      | ONLINE | localhost |
 2          |                | 2817      | ONLINE | localhost |
 3          |                | 2406      | ONLINE | localhost |
 4          |                | 2718      | ONLINE | localhost |
 5          |                | 2874      | ONLINE | localhost |
 6          |                | 2712      | ONLINE | localhost |
 7          |                | 2911      | ONLINE | localhost |
 8          |                | 2660      | ONLINE | localhost |
 33         |                | 2714      | ONLINE | localhost |
 34         |                | 3062      | ONLINE | localhost |
 35         |                | 3099      | ONLINE | localhost |
 36         |                | 3060      | ONLINE | localhost |
 37         |                | 2821      | ONLINE | localhost |
 38         |                | 2600      | ONLINE | localhost |
 39         |                | 2603      | ONLINE | localhost |
 40         |                | 2843      | ONLINE | localhost |
 89         |                | 725       | ONLINE | localhost |
 90         |                | 553       | ONLINE | localhost |
 91         |                | 560       | ONLINE | localhost |
 92         |                | 479       | ONLINE | localhost |
 93         |                | 951       | ONLINE | localhost |
 94         |                | 397       | ONLINE | localhost |
 95         |                | 635       | ONLINE | localhost |
 96         |                | 454       | ONLINE | localhost |
 110        |                | 516       | ONLINE | localhost |
```

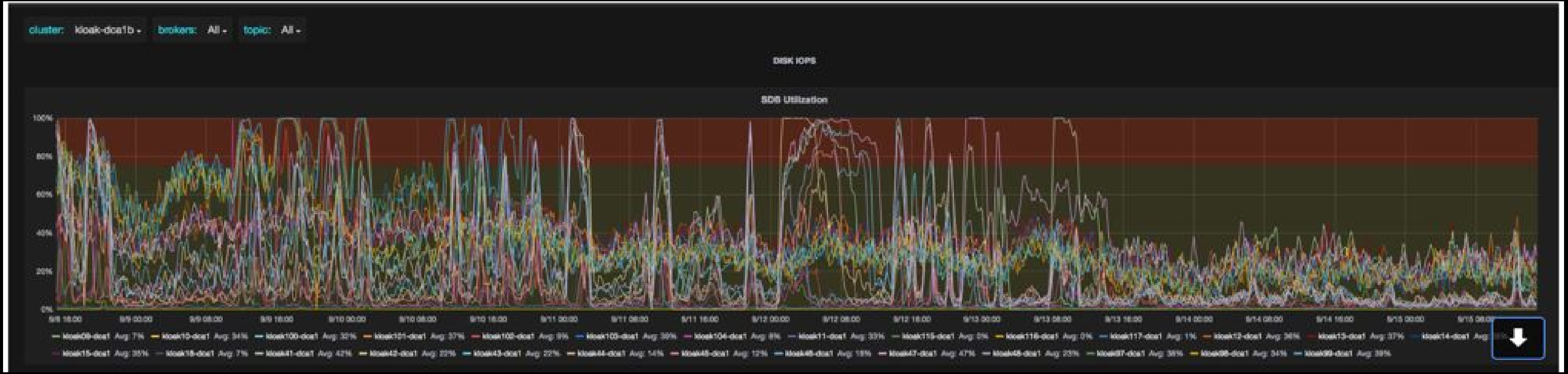


Partition Rebalancing

- Calculates partition imbalance and inter-broker dependency.
- Generates & Executes Rebalance Plan.
- Rebalance plans are incremental, can be stopped and resumed.
- Currently on-demand, Automated in the future.



XFS vs EXT4


cluster: kloak-dca1b
brokers: All
topic: All
DISK IOPS
SDB Utilization
100%
80%
60%
40%
20%
0%
9/8 16:00
9/9 00:00
9/9 08:00
9/9 16:00
9/10 00:00
9/10 08:00
9/10 16:00
9/11 00:00
9/11 08:00
9/11 16:00
9/12 00:00
9/12 08:00
9/12 16:00
9/13 00:00
9/13 08:00
9/13 16:00
9/14 00:00
9/14 08:00
9/14 16:00
9/15 00:00
9/15 08:00
kloak09-dca1 Avg: 7%
kloak10-dca1 Avg: 34%
kloak100-dca1 Avg: 32%
kloak101-dca1 Avg: 37%
kloak102-dca1 Avg: 9%
kloak103-dca1 Avg: 39%
kloak104-dca1 Avg: 8%
kloak11-dca1 Avg: 33%
kloak115-dca1 Avg: 0%
kloak116-dca1 Avg: 0%
kloak117-dca1 Avg: 1%
kloak12-dca1 Avg: 36%
kloak13-dca1 Avg: 37%
kloak14-dca1 Avg:
kloak15-dca1 Avg: 35%
kloak16-dca1 Avg: 7%
kloak41-dca1 Avg: 42%
kloak42-dca1 Avg: 22%
kloak43-dca1 Avg: 22%
kloak44-dca1 Avg: 14%
kloak45-dca1 Avg: 12%
kloak46-dca1 Avg: 18%
kloak47-dca1 Avg: 47%
kloak48-dca1 Avg: 23%
kloak97-dca1 Avg: 38%
kloak98-dca1 Avg: 34%
kloak99-dca1 Avg: 39%

Summary: Scale

- Kafka Brokers:
  - Multiple Clusters per DC
  - Use case based tuning
- Rest Proxy to reduce connections and better batching
- Rest Proxy & Clients
  - Batch everywhere, Async produce
  - Replace Jersey with Jetty
- XFS

Summary: Reliability

- Local Agent
- Secondary Clusters
- Multi Producer support in Rest Proxy
- uReplicator
- Auditing via Chaperone

Future Work

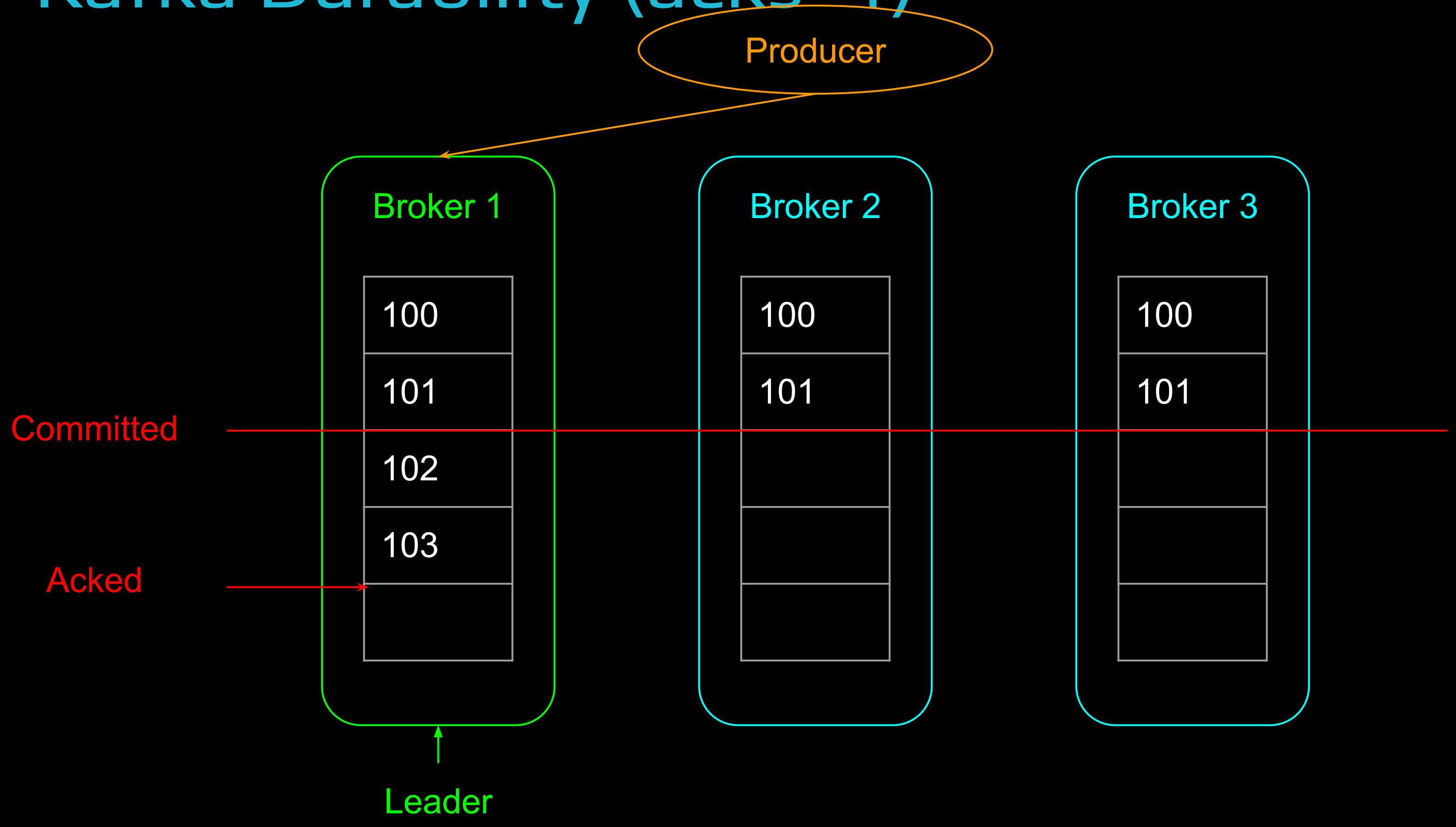
- Open source contribution
  - Chaperone
  - Toolkit
- Data Lineage
- Active Active Kafka
- Chargeback
- Exactly once mirroring via uReplicator

Questions ?

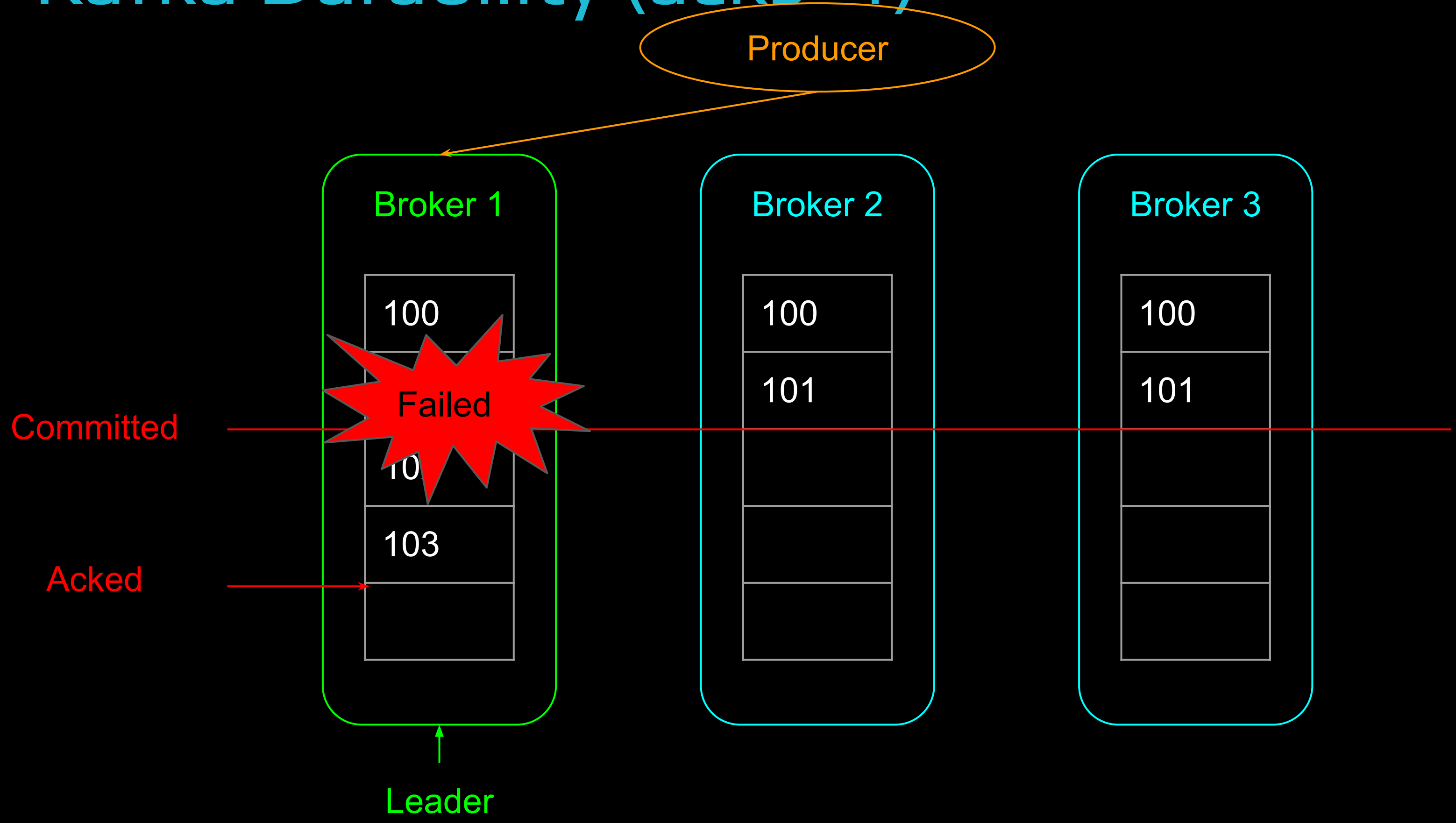
ankur@uber.com

Extra Slides

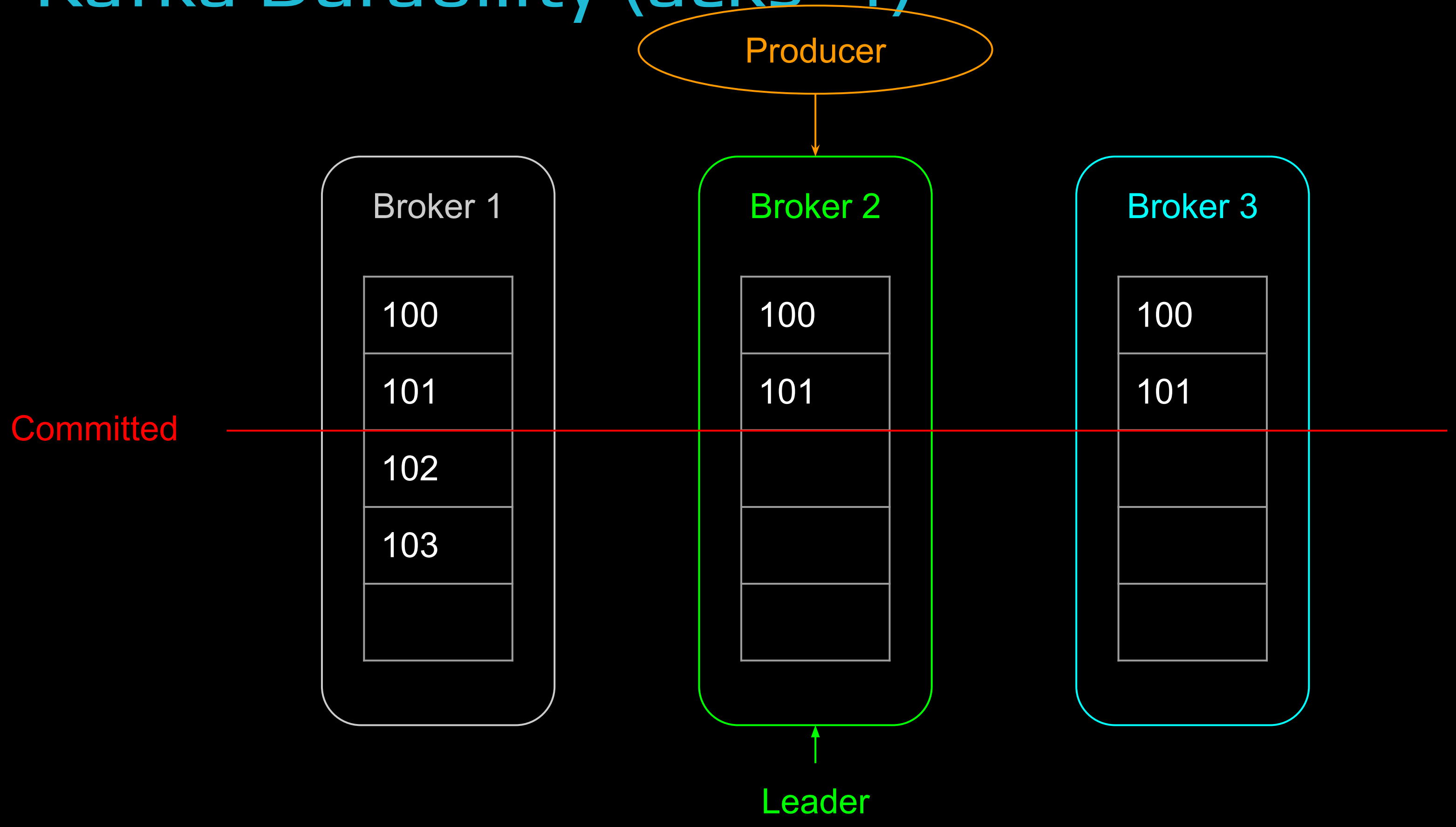
Kafka Durability (acks=1)



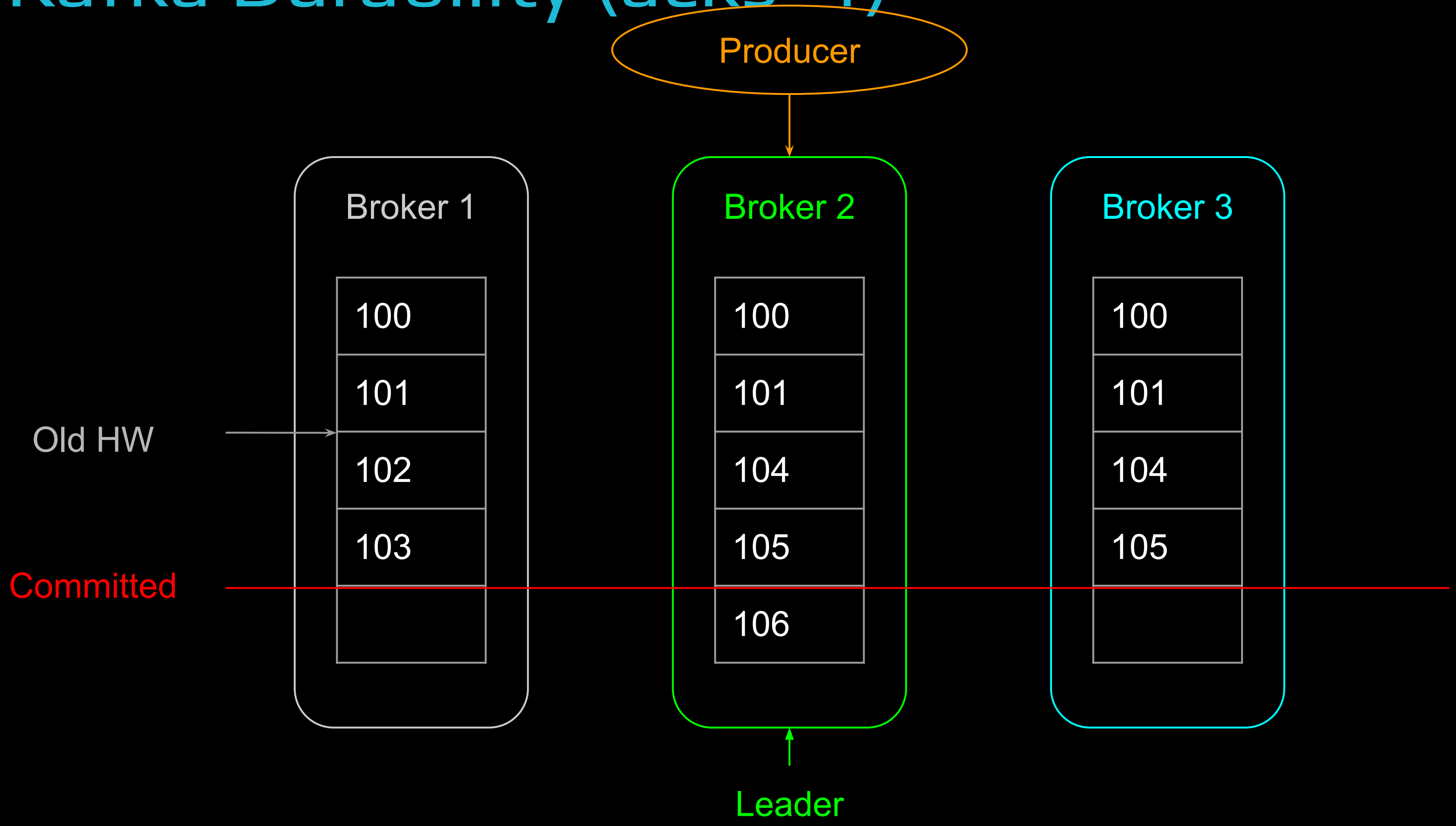
Kafka Durability (acks=1)



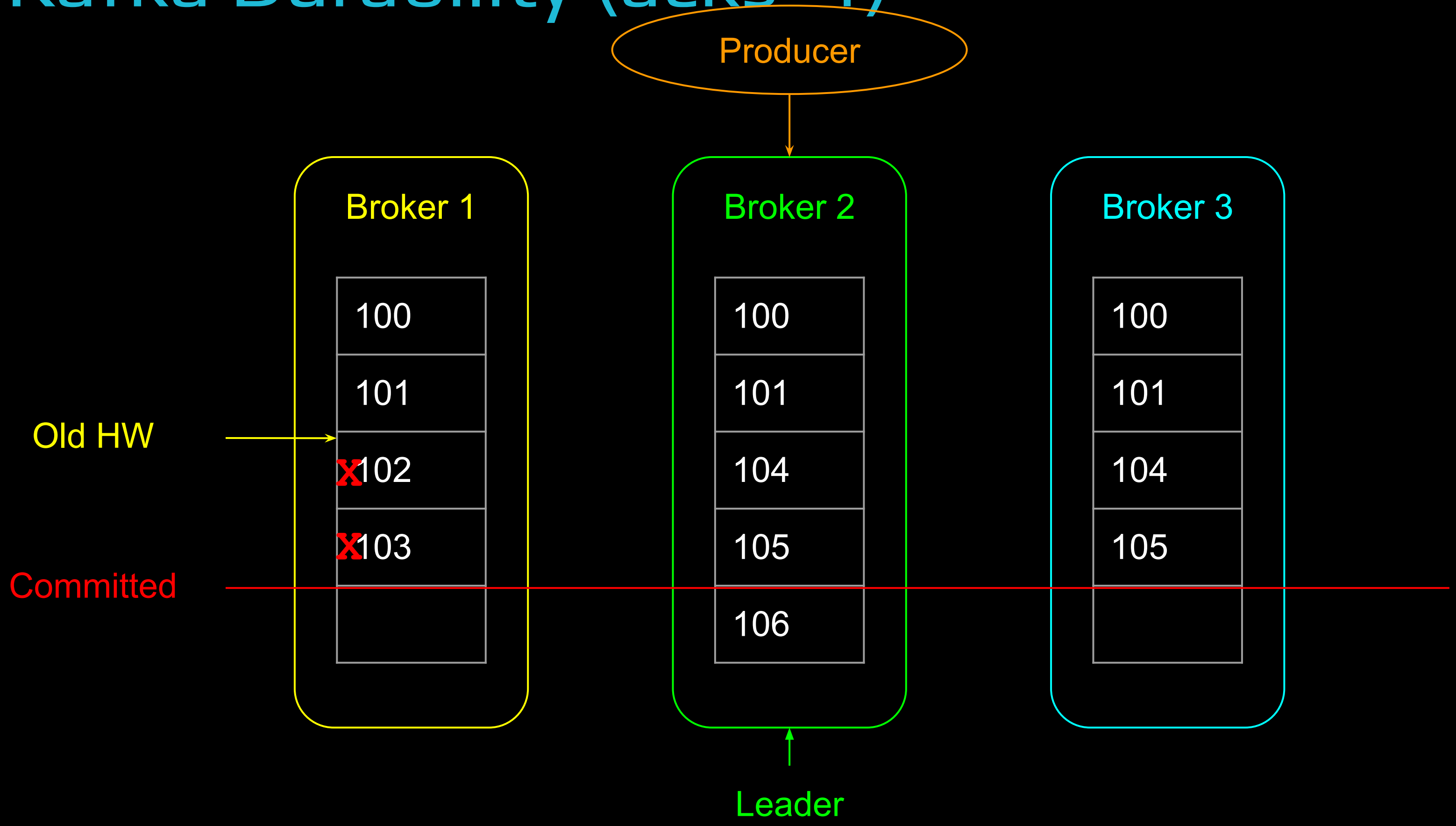
Kafka Durability (acks=1)



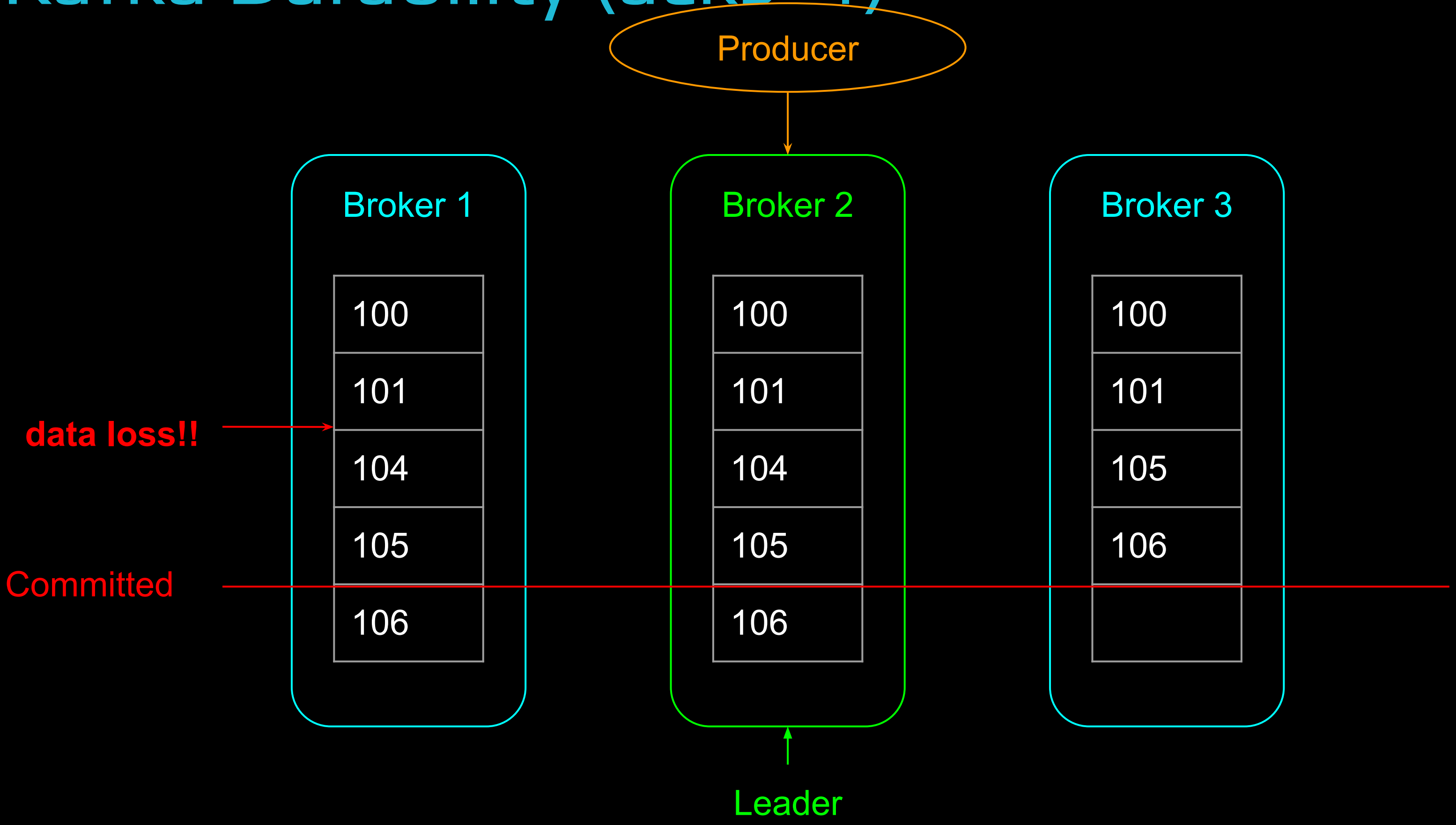
Kafka Durability (acks=1)



Kafka Durability (acks=1)

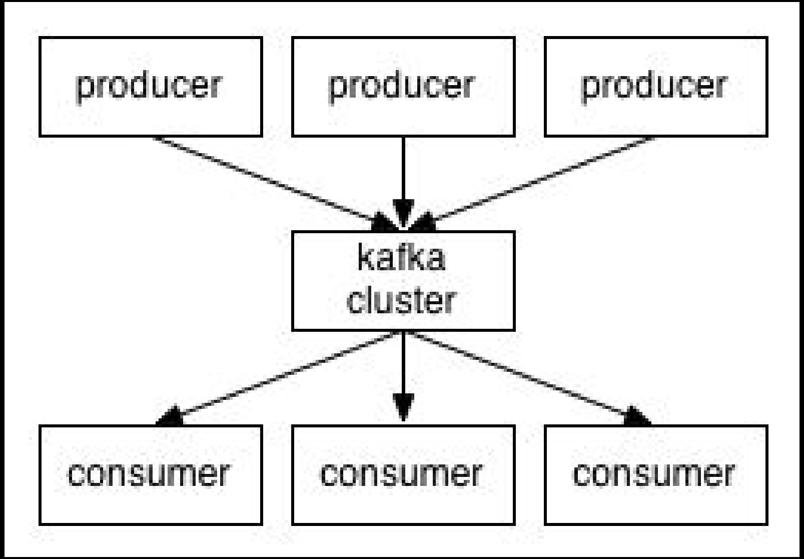


Kafka Durability (acks=1)



kafka

Distributed Messaging system

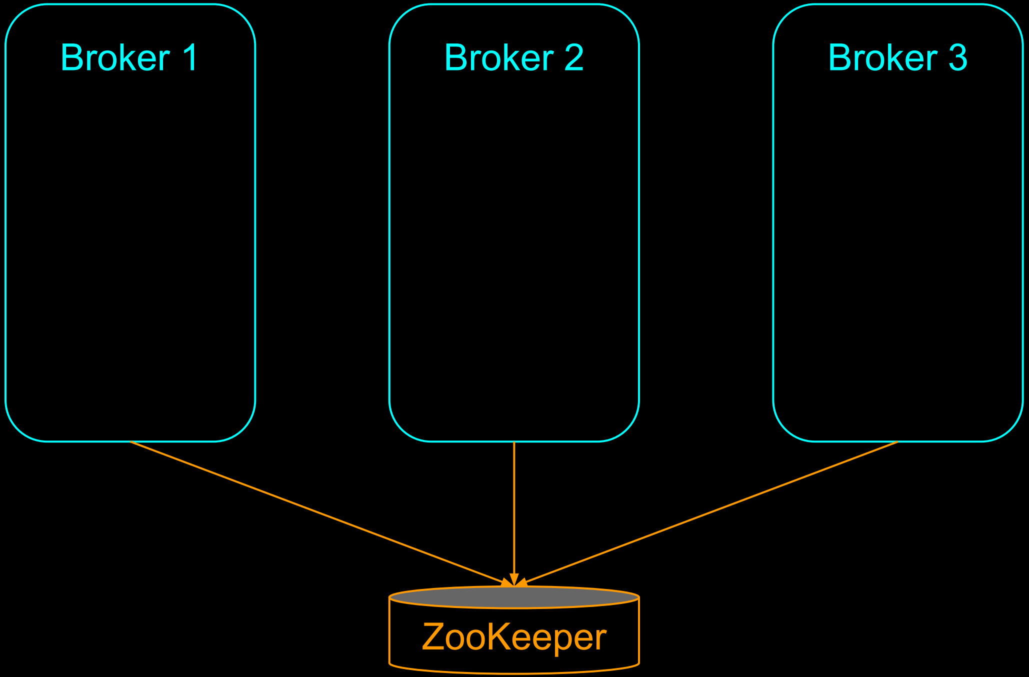


- High throughput
- Low latency
- Scalable
- Centralized
- Real-time

* Supported in Kafka 0.8+

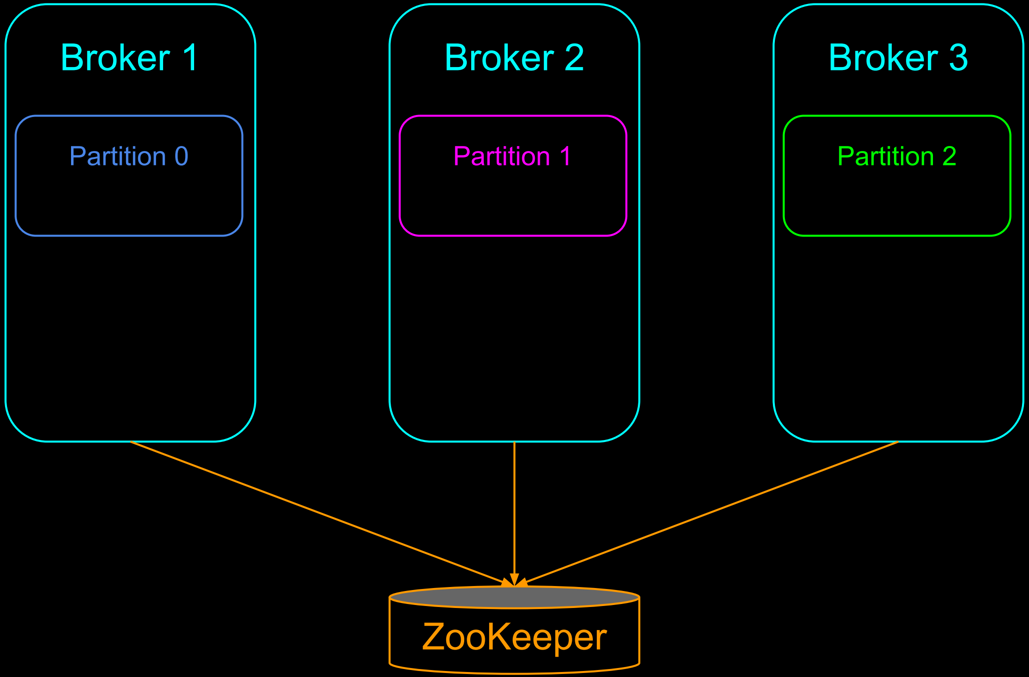
What is Kafka?

- Distributed
- Partitioned
- Replicated
- Commit Log



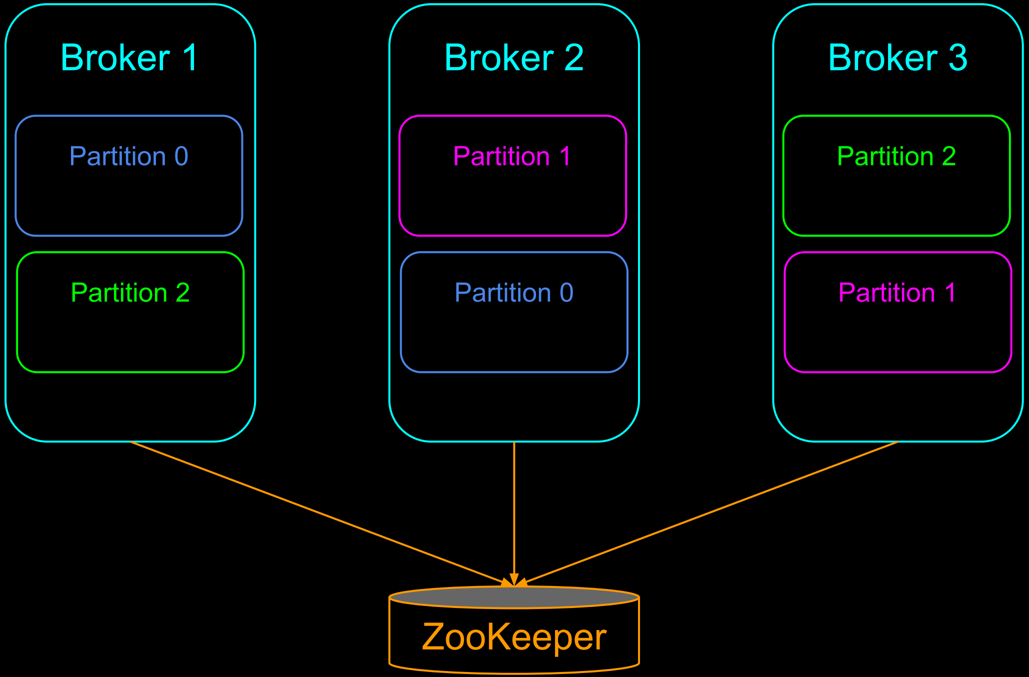
What is Kafka?

- Distributed
- Partitioned
- Replicated
- Commit Log



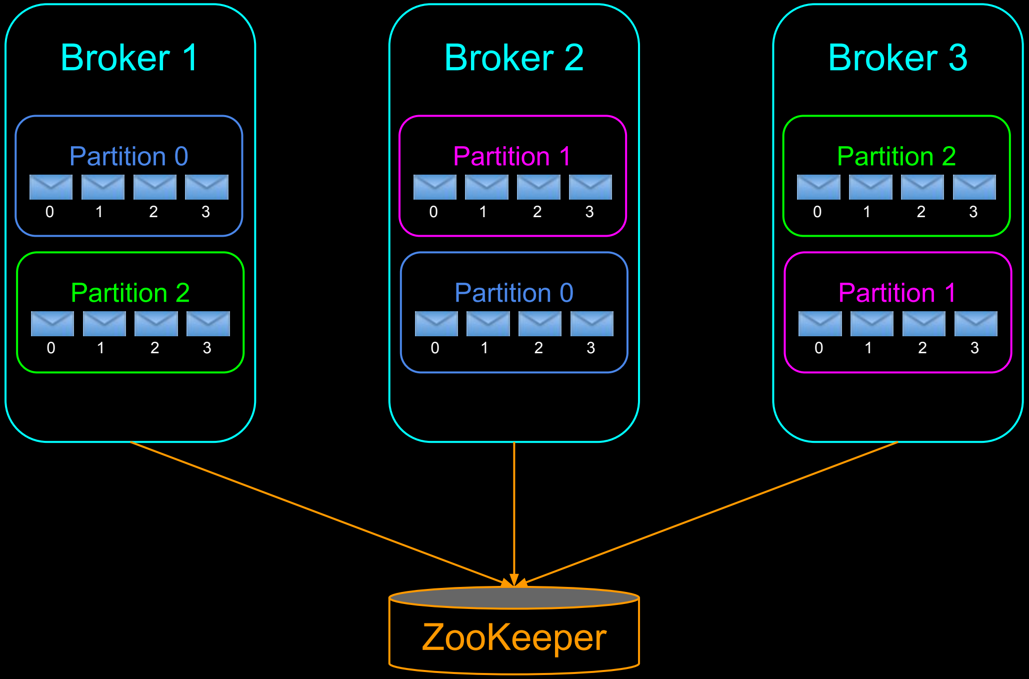
What is Kafka?

- Distributed
- Partitioned
- Replicated
- Commit Log

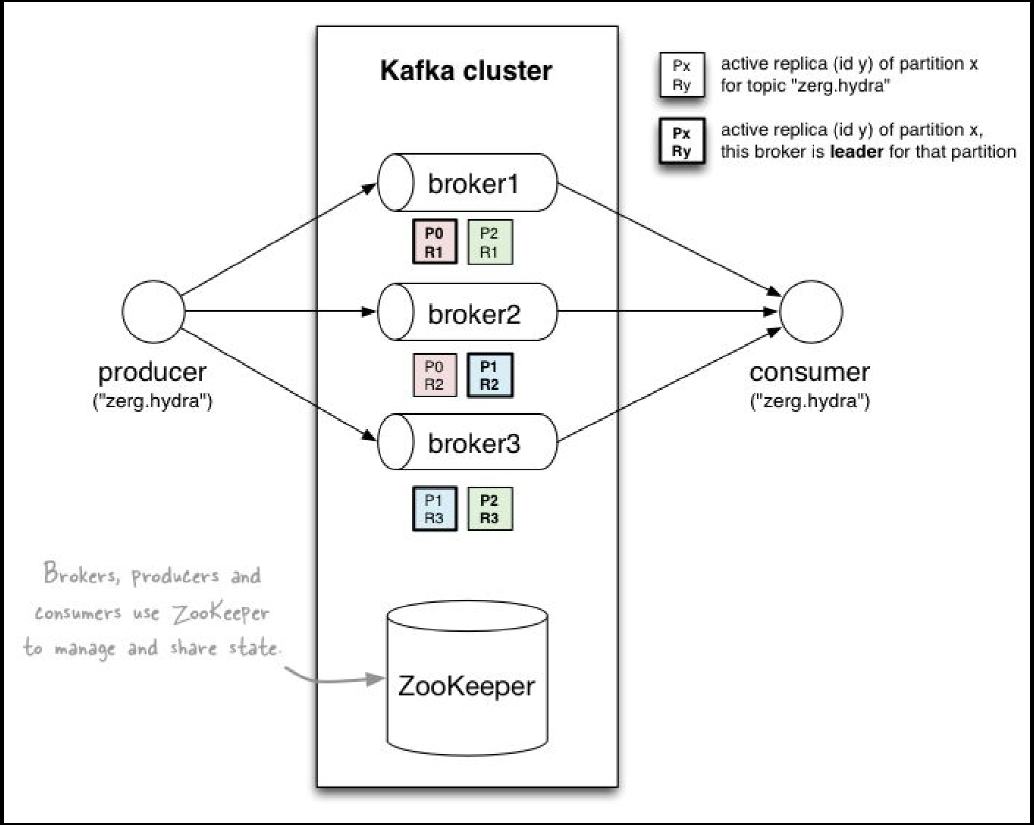


What is Kafka?

- Distributed
- Partitioned
- Replicated
- Commit Log



Kafka Concepts


Kafka cluster
Px Ry
active replica (id y) of partition x for topic "zerg.hydra"
Px Ry
active replica (id y) of partition x, this broker is leader for that partition
broker1
P0 R1
P2 R1
broker2
P0 R2
P1 R2
broker3
P1 R3
P2 R3
producer ("zerg.hydra")
consumer ("zerg.hydra")
Brokers, producers and consumers use ZooKeeper to manage and share state.
ZooKeeper