How we used Kafka to scale our Database Infrastructure

Basavaiah Thambara(Basu)

Staff Site Reliability Engineer

(https://www.linkedin.com/in/basavaiaht)

Today's agenda

	Introduction to Espresso
	Espresso - Replication
MySQL	Espresso with MySQL Replication
kafka	Espresso with Kafka Replication
kafka	Advantages of Using Kafka
	How Kafka Based Replication Works
	Conclusion & References

Espresso

Document store

MySQL

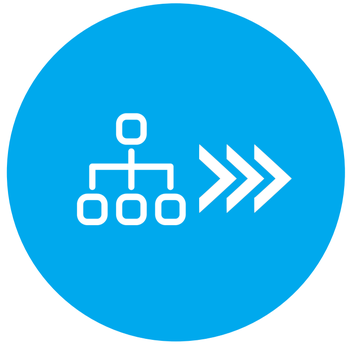
RDBMS & k-v Stores

Consistent & Partition tolerance

Espresso : Features

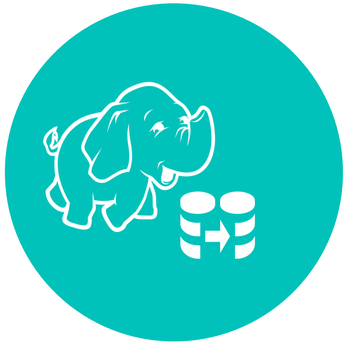
Multi-colo writes

Secondary Indexing

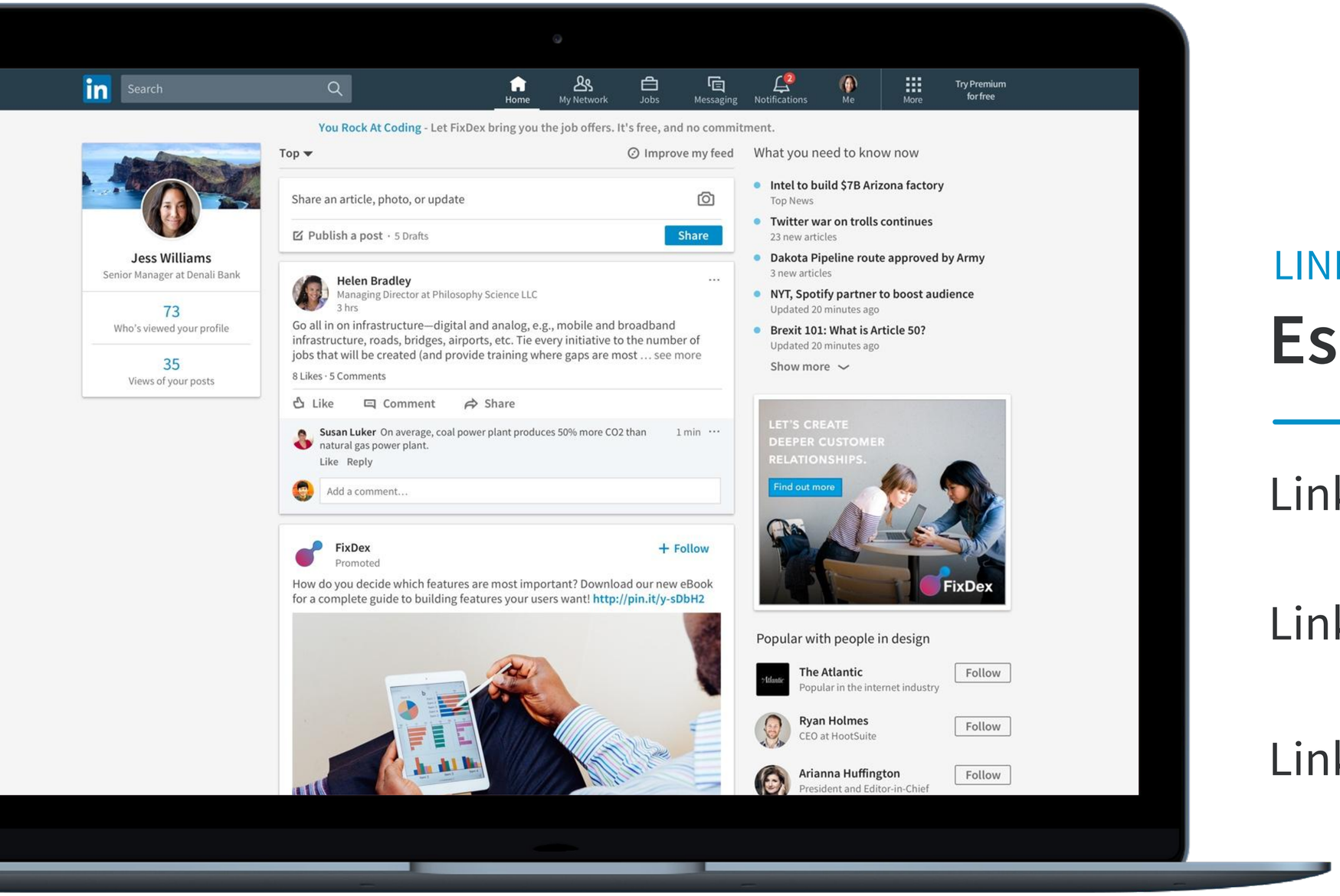


Schema Evolution

Change data capture



Bulk import export



LINKEDIN

Espresso : Use Cases

Linkedin Profiles

Linkedin Invitations

Linkedin InMails, etc.

Espresso : Current Scale

O(100)
Clusters

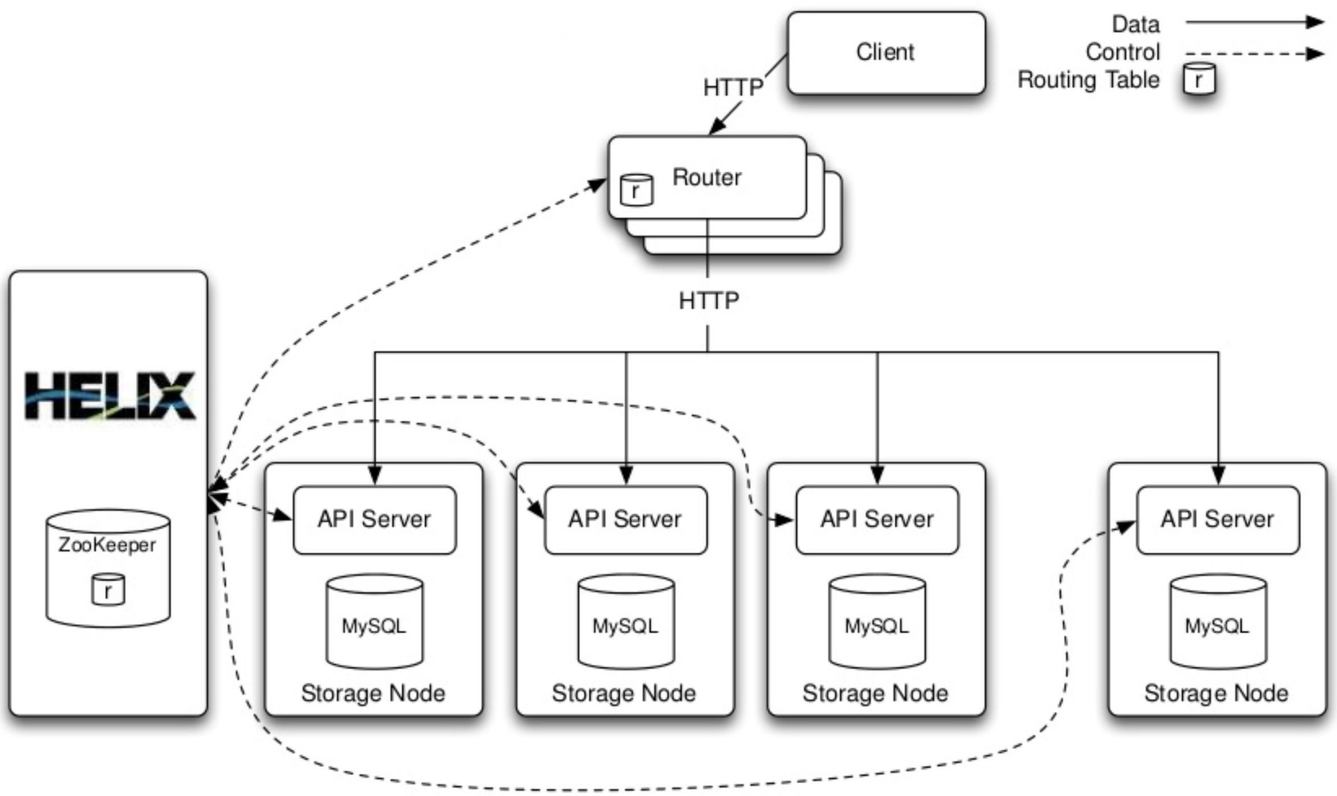
O(10K)
Servers

O(100)
Databases

O(PB)
Data

O(M)
Peak QPS

Espresso : Basic Architecture



- Client/Application
- Router
- Helix
- Zookeeper
- Storage node

Espresso : Replication Requirements

Read Scaling

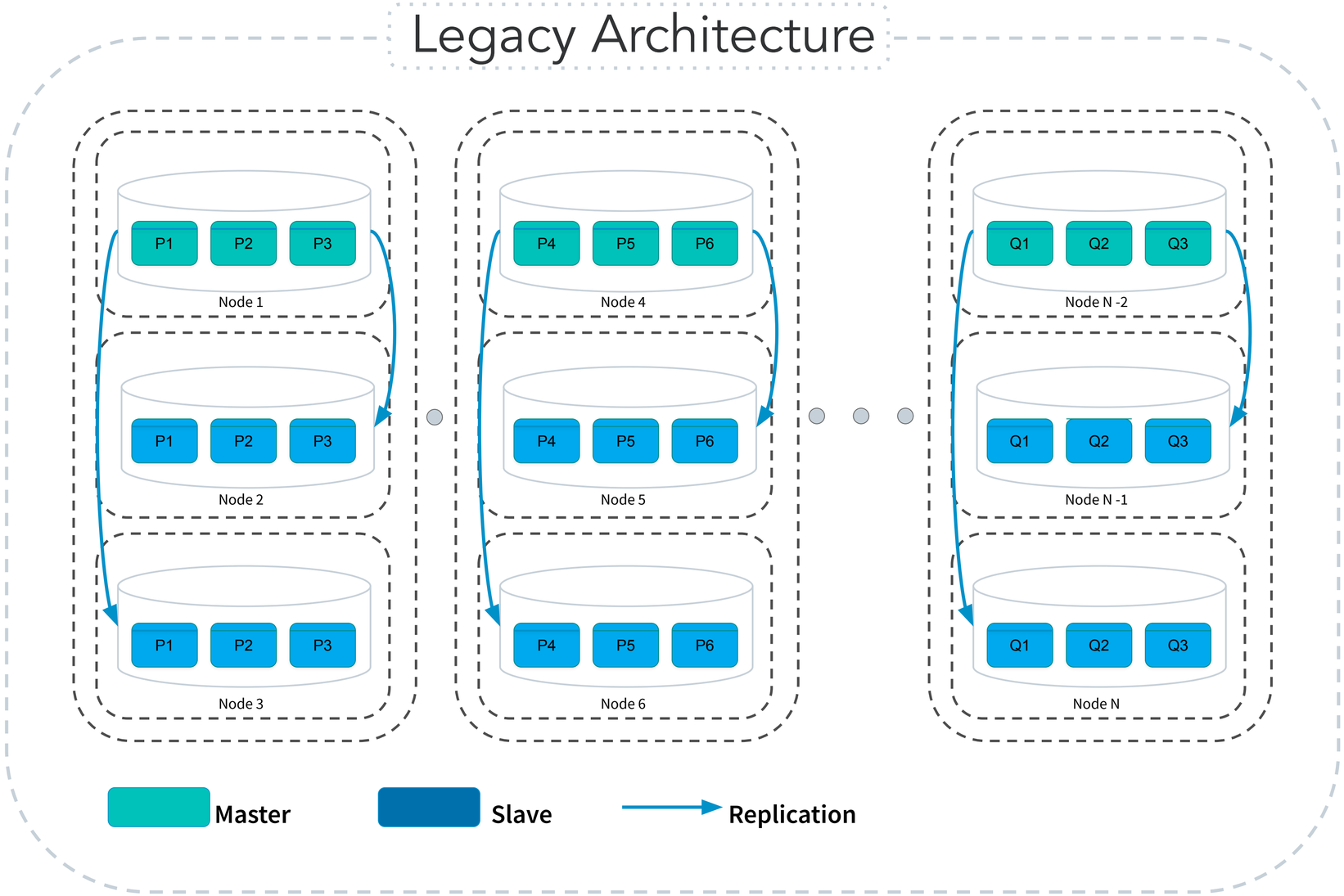
High Availability

Disaster Recovery

Multi-colo writes

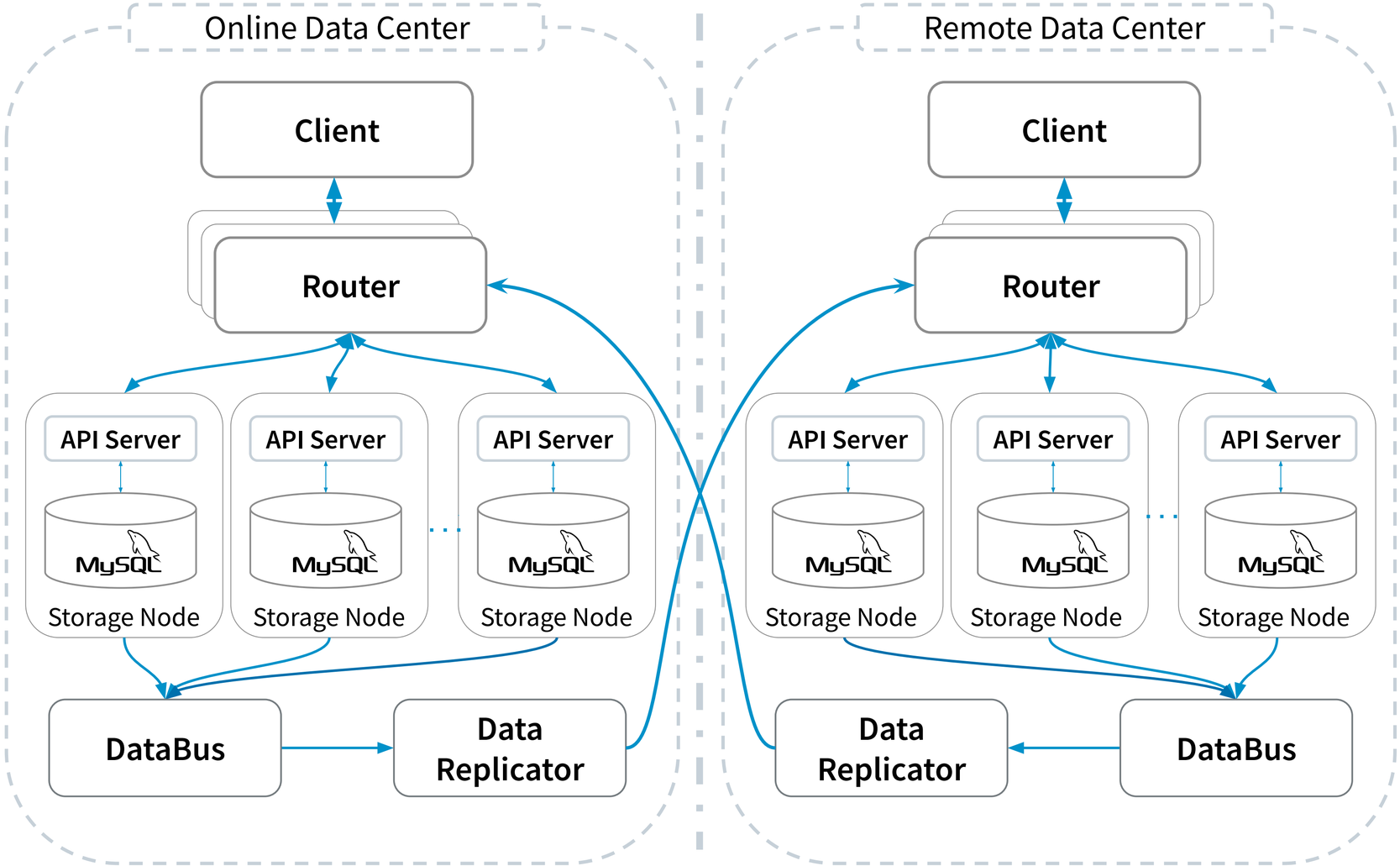
Backups

Espresso : Local Replication



- MySQL Replication
- 3 Copies
- Per Node Replication
- Node Failure

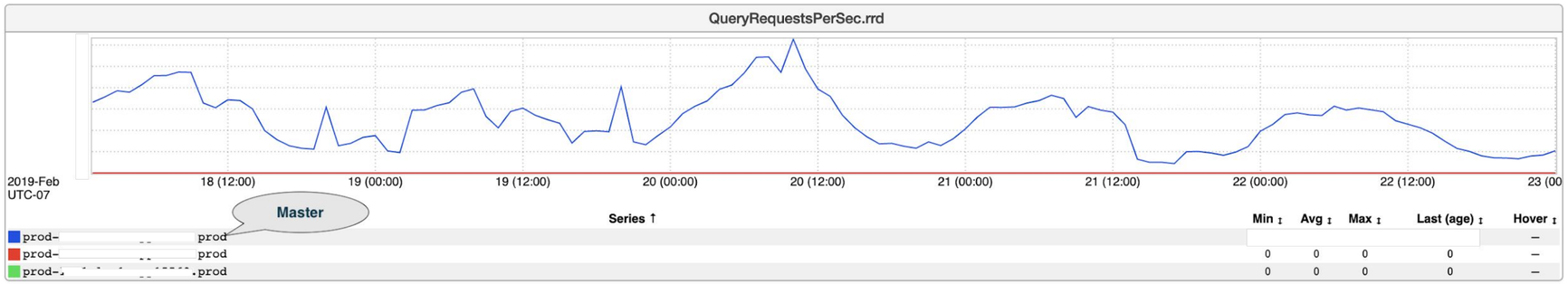
Espresso : Cross Colo Replication (Legacy)



- Databus
- Data Replicator
- Colo failure

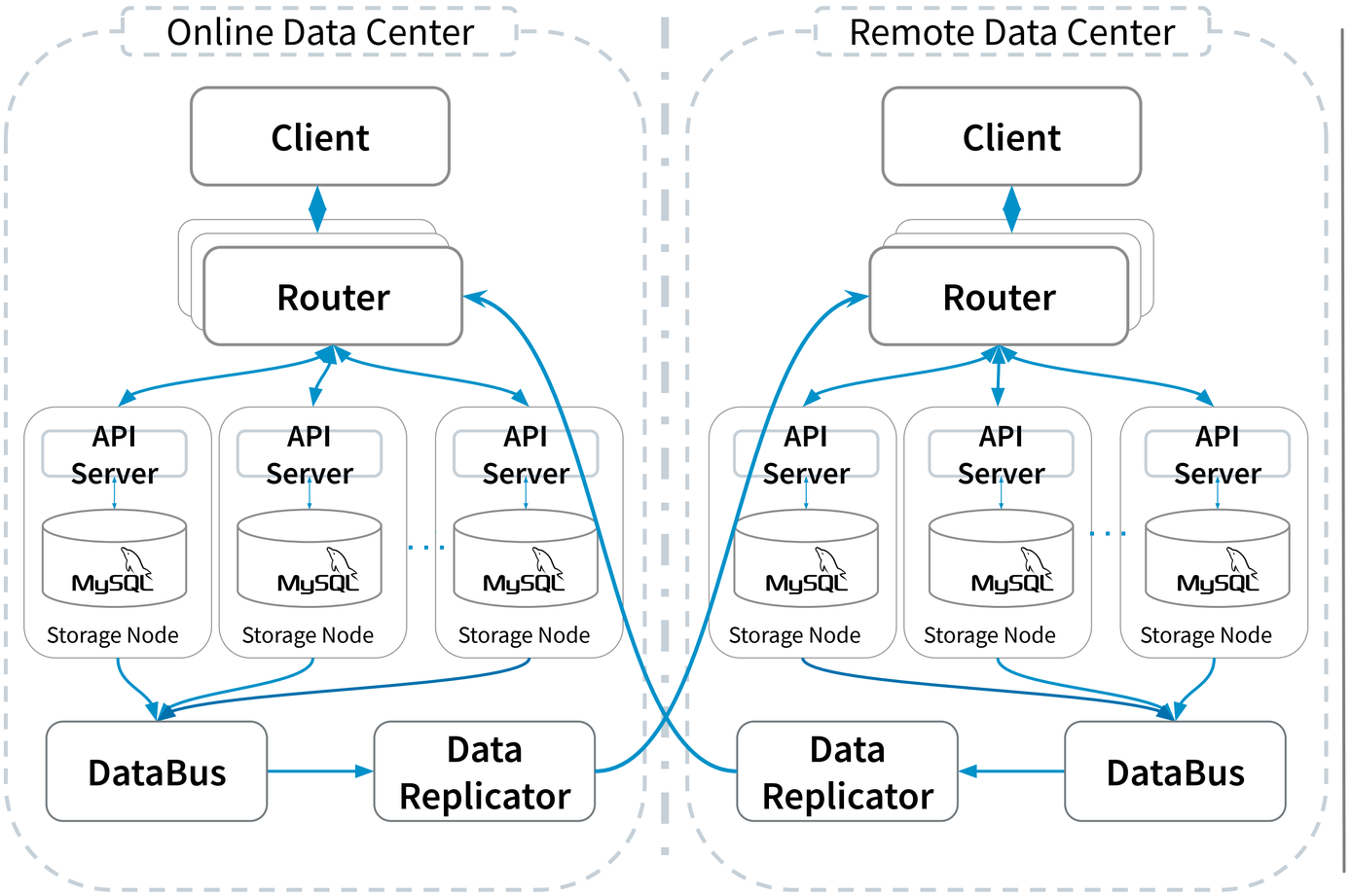
Limitations : Per Instance Replication

Poor Resource Utilization



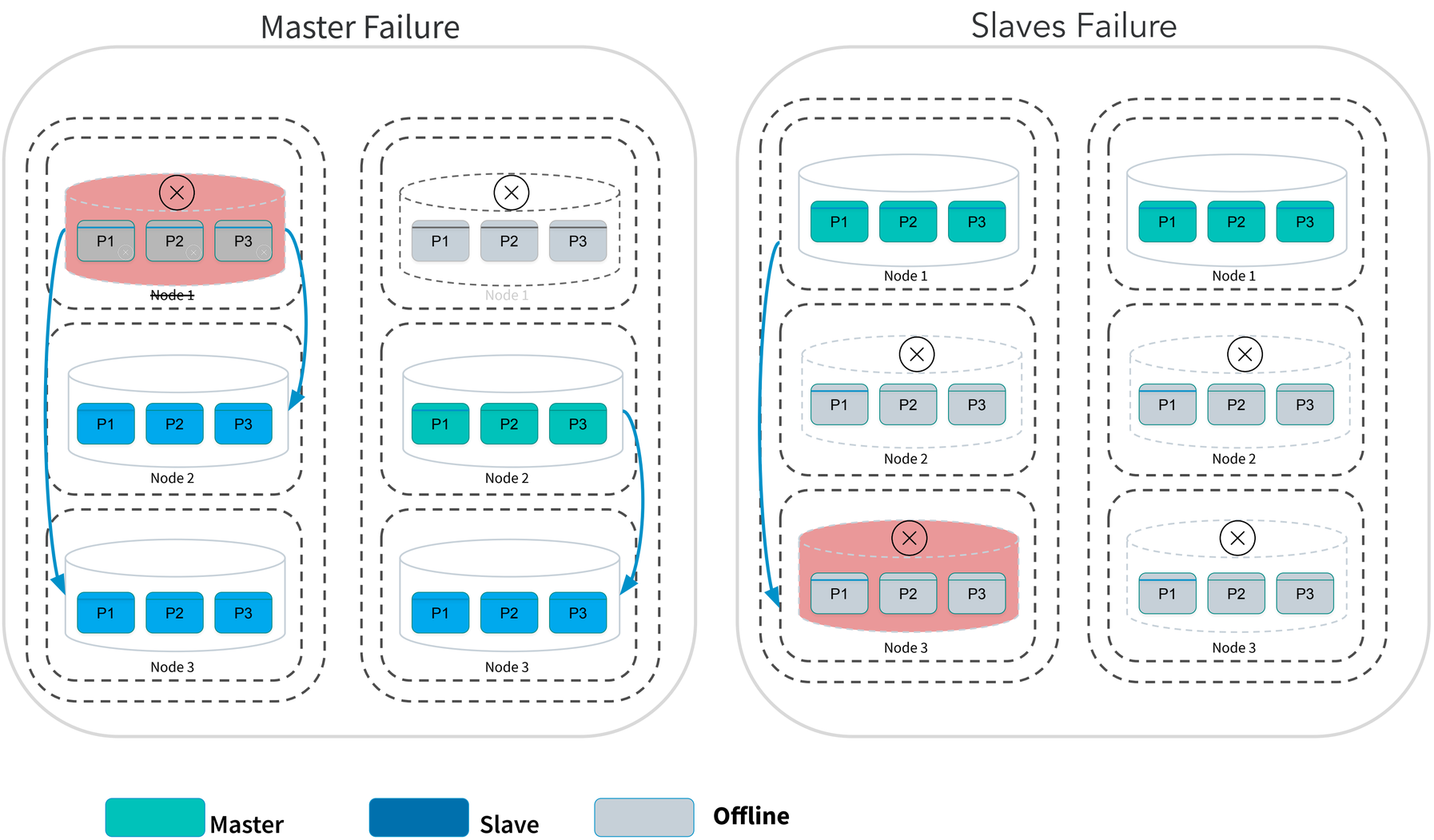
Cross Colo Replication (Legacy)

Limitations : Per Instance Replication



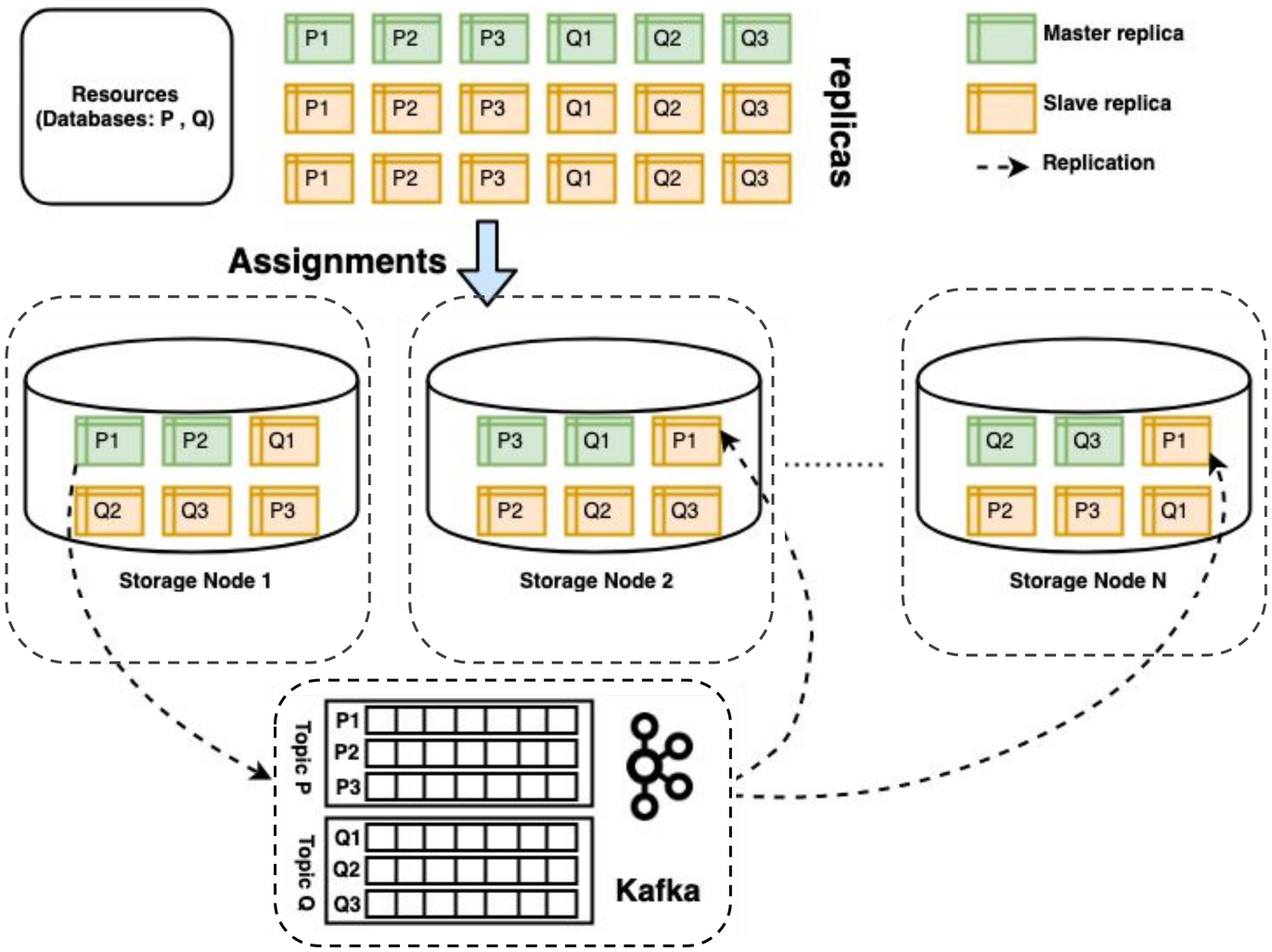
- Databus
  - tightly coupled to storage node
  - operational complexity
  - Uses SSD,higher cost to serve
- Cluster expansion is painful
  - Lot of manual steps
  - Needs databus expansion
  - Requires downtime

Limitations: Per Instance Replication



- Upon master failure, single node gets traffic
- Human intervention to bring up slaves
- Slave-less situation might lead to outage

Espresso : Replication Using Kafka



New architecture

- Per partition replication
- Flexible partition placement
- Every node serves traffic
- Data replicator uses kafka

Advantages: Per Partition Replication

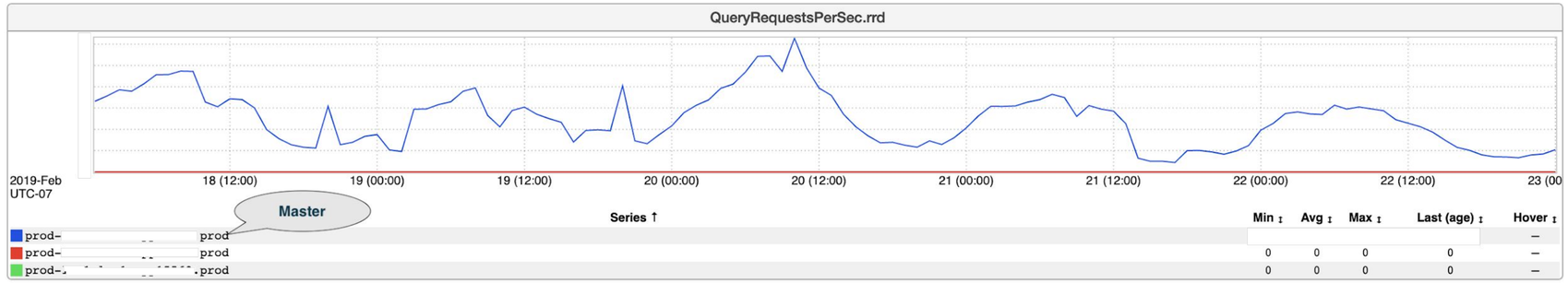
1

Better resource utilization.

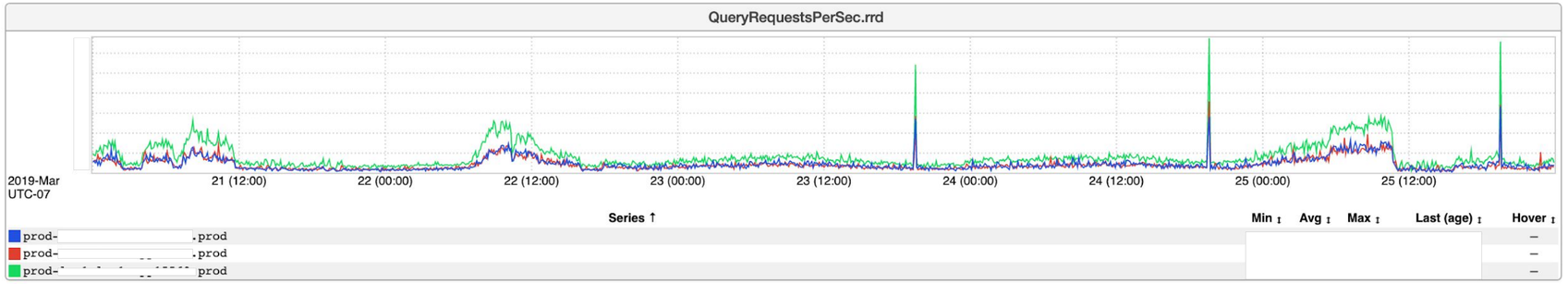
Advantages: Per Partition Replication

1

Better resource utilization.



Resource Utilization (Legacy)



Resource Utilization (kafka)

Advantages: Per Partition Replication

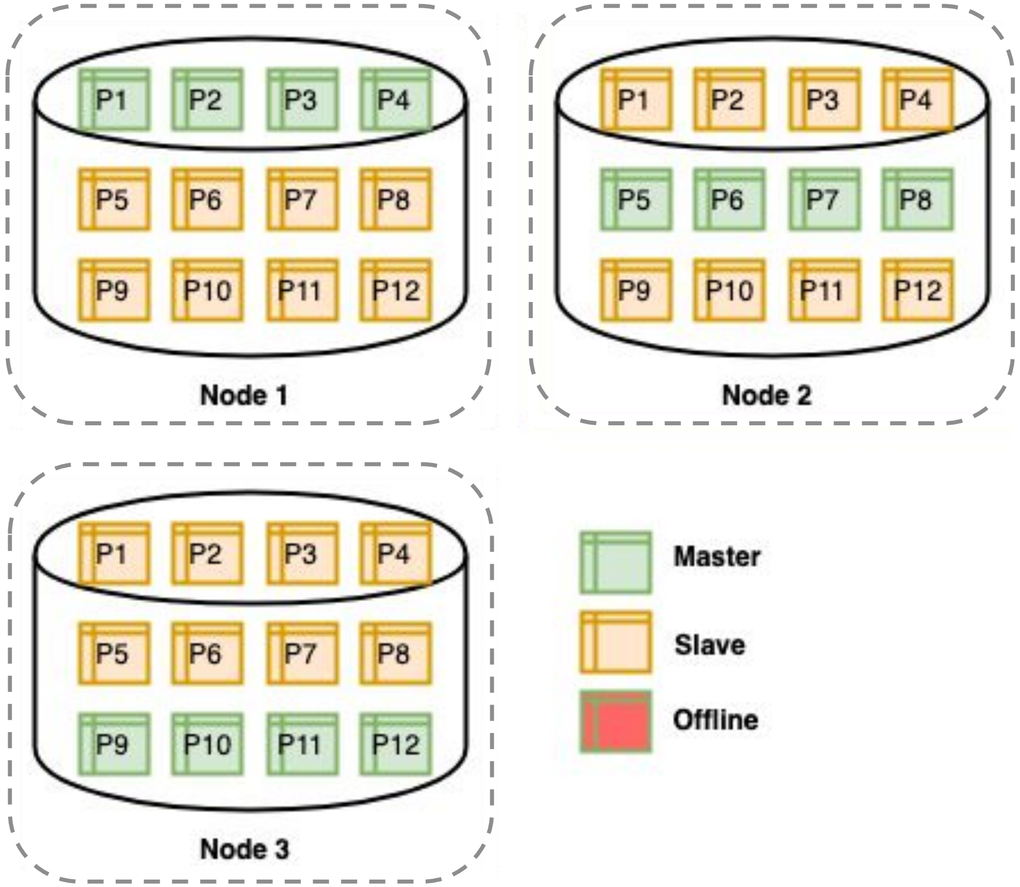
1

Better resource utilization.

2

Easy cluster expansion.

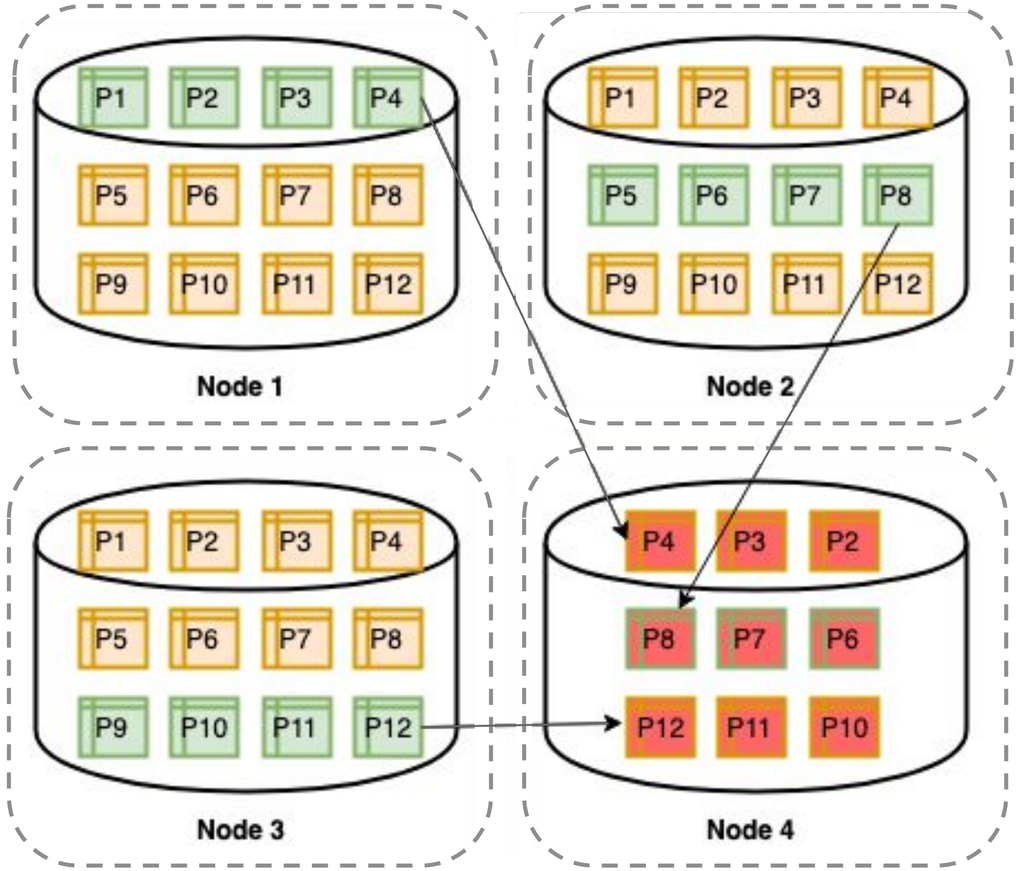
Cluster Expansion



Initial cluster state with 12 partitions,

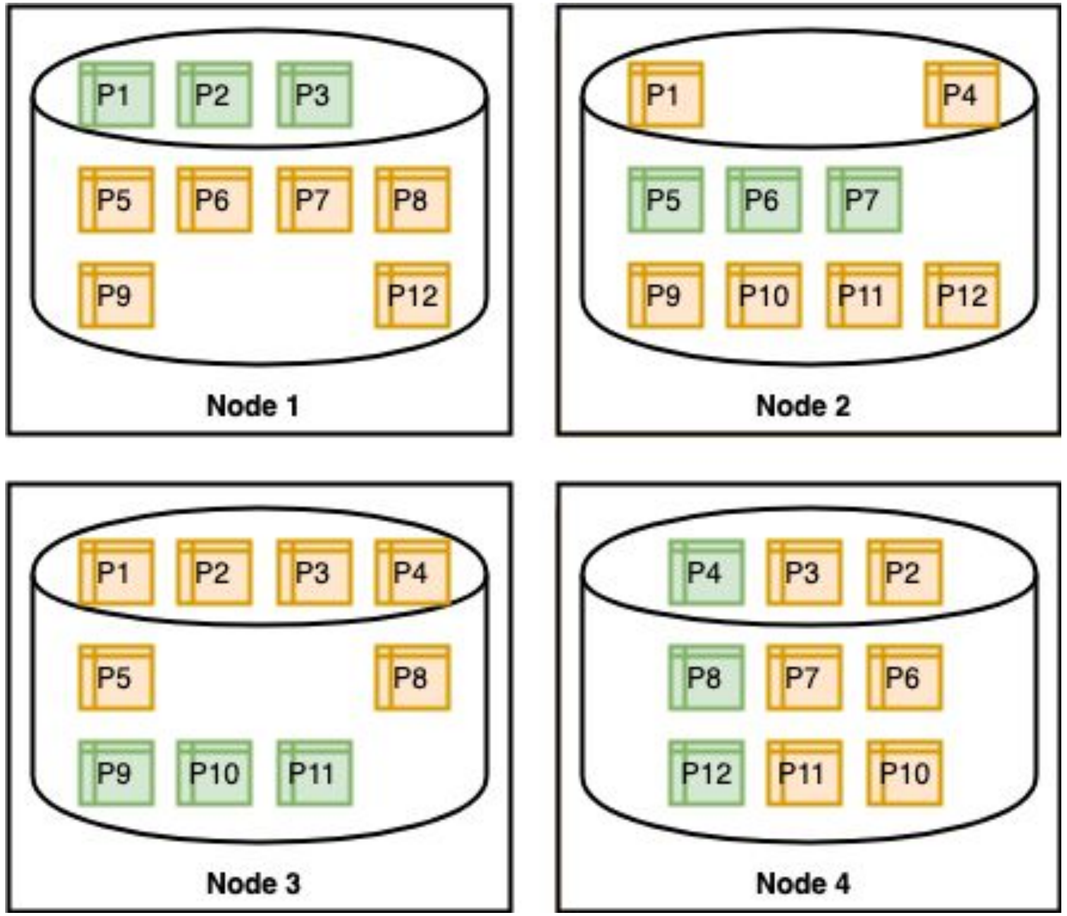
3 storage nodes, replication factor=3

Cluster Expansion



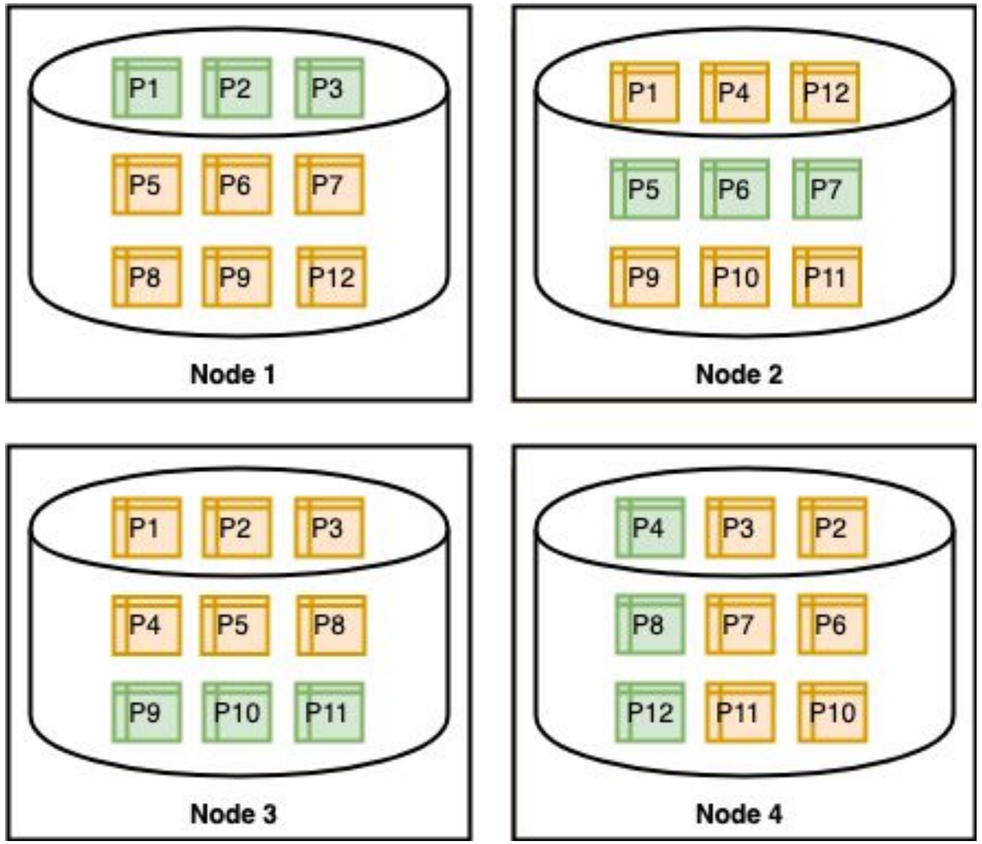
Adding a node: Helix will send offline to Slave for new node

Cluster Expansion



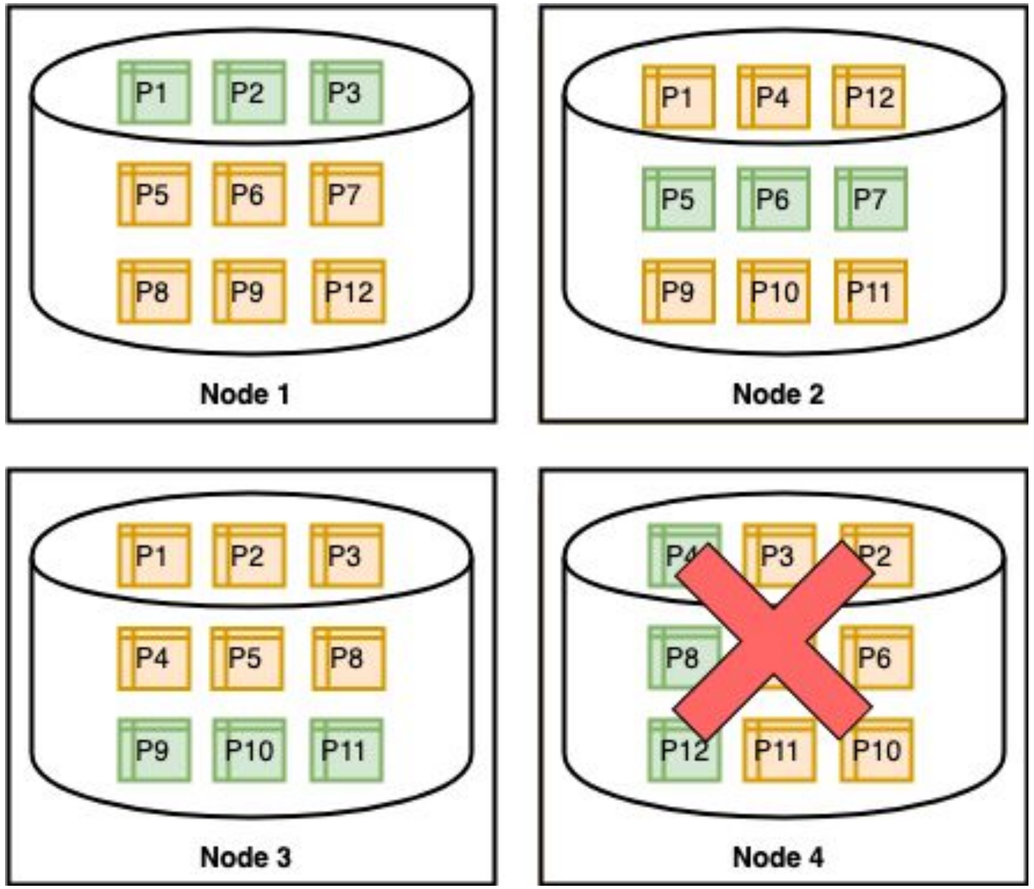
Once partitions on new node are ready, transfer ownership and drop old

Cluster Expansion



Cluster state after expansion with 12 partitions, 4 storage nodes, r=3

Advantages : Per Partition Replication



Node failure

- parallel mastership handoff
- parallel restore of slaves

Advantages: Per Partition Replication

1

Better resource utilization.

2

Easy cluster expansion.

3

No human intervention.

Advantages: Per Partition Replication

4

Databus complexity eliminated.

5

Cost savings.

6

Single platform.

Internal replication

Cross colo replication

Change capture for nearline

Implementing Kafka based replication

1

Requirements

2

Solution

- Broker and producer config
- Implement

Implementing Kafka based replication

1

Requirements

- Guaranteed Delivery
- In-Order
- Exactly Once(sort of)

Implementing Kafka based replication

Broker config

- Kafka broker config
  - replication factor =3
  - min.isr = 2
  - Disabled unclean leader elections

2

Solution

- Broker and producer config
- Implement

Implementing Kafka based replication

Producer Config

- **acks = “all”**
- **Infinite retries**
- **block.on.buffer.full = true**

- **max.in.flight.requests.per.connection = 1**
- **linger.ms = 0**
- **on non-retryable exception**
  - **destroy producer**
  - **create new producer**
  - **resume from last checkpoint**

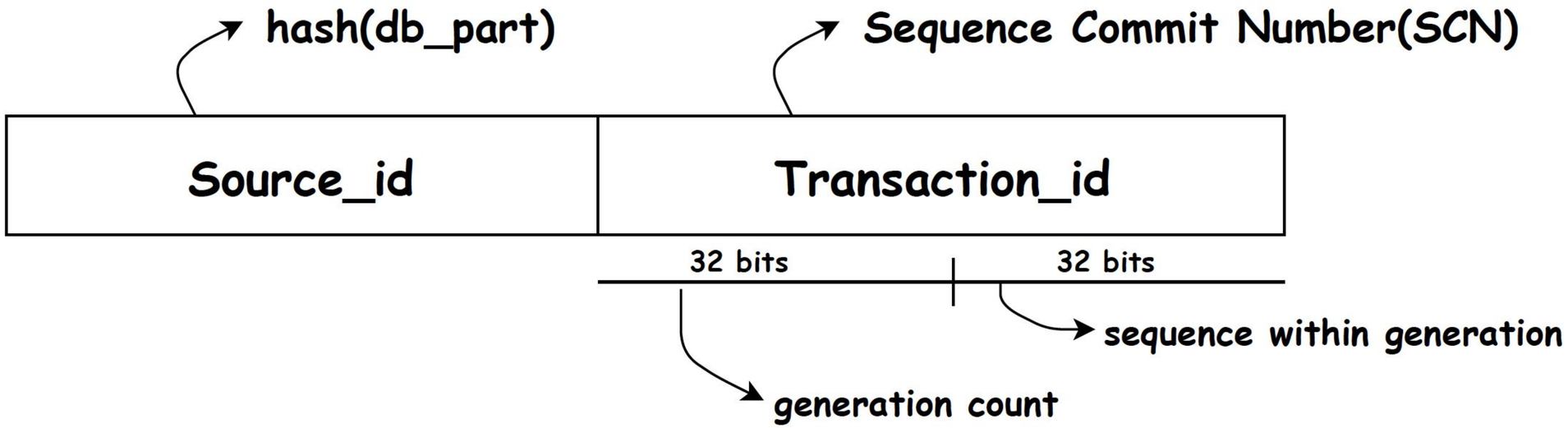
2

Solution

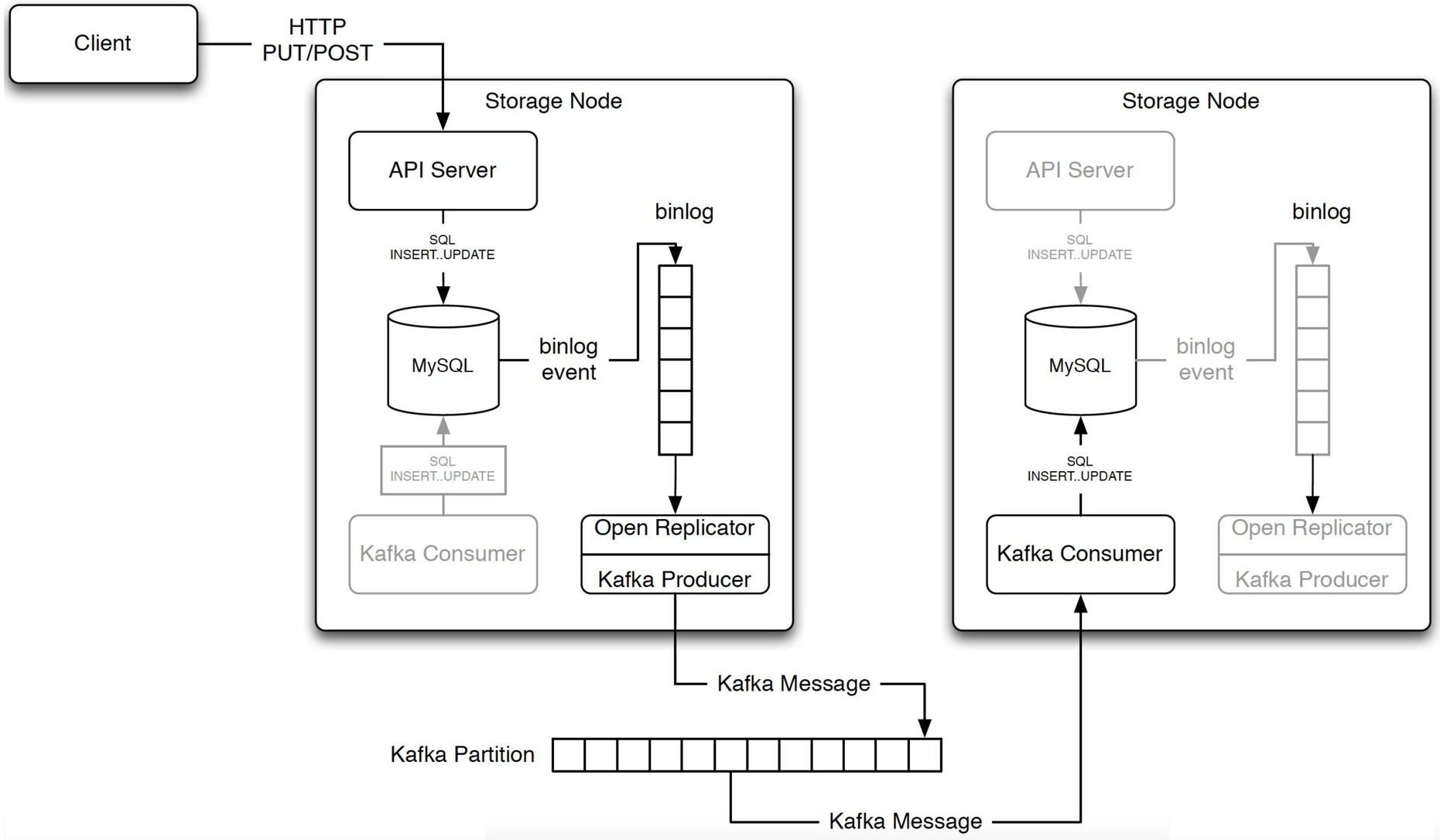
- Broker and producer config
- Implement

Global Transaction Identifier

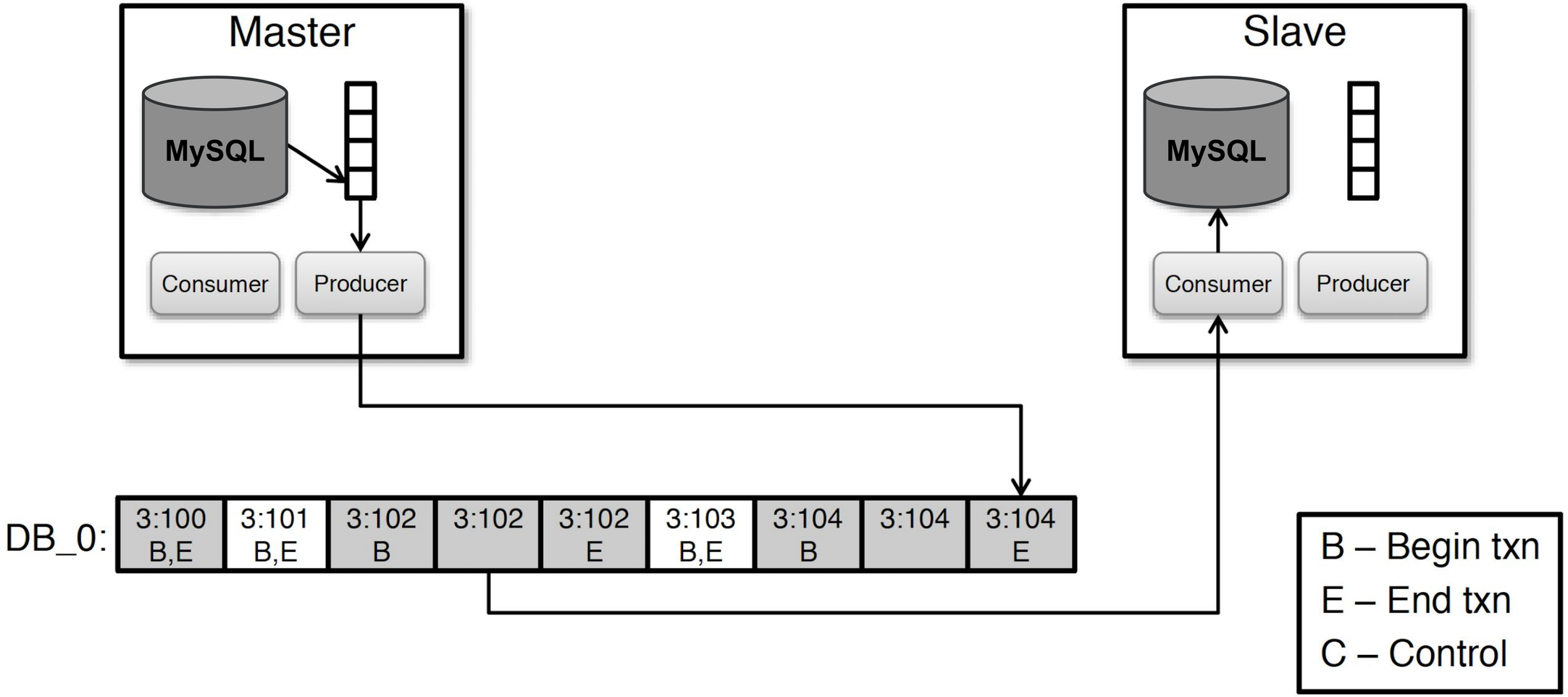
- Global transaction identifier(GTID)
- Unique



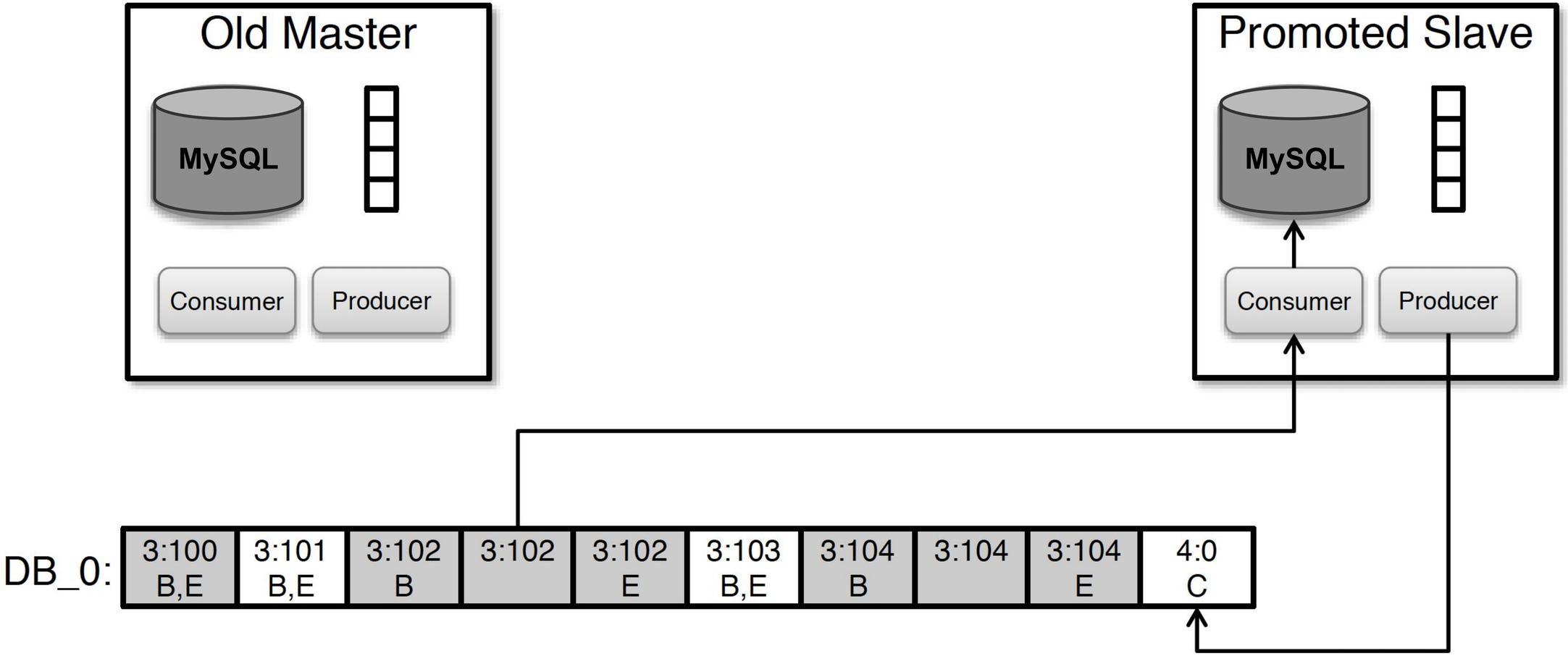
Replication flow



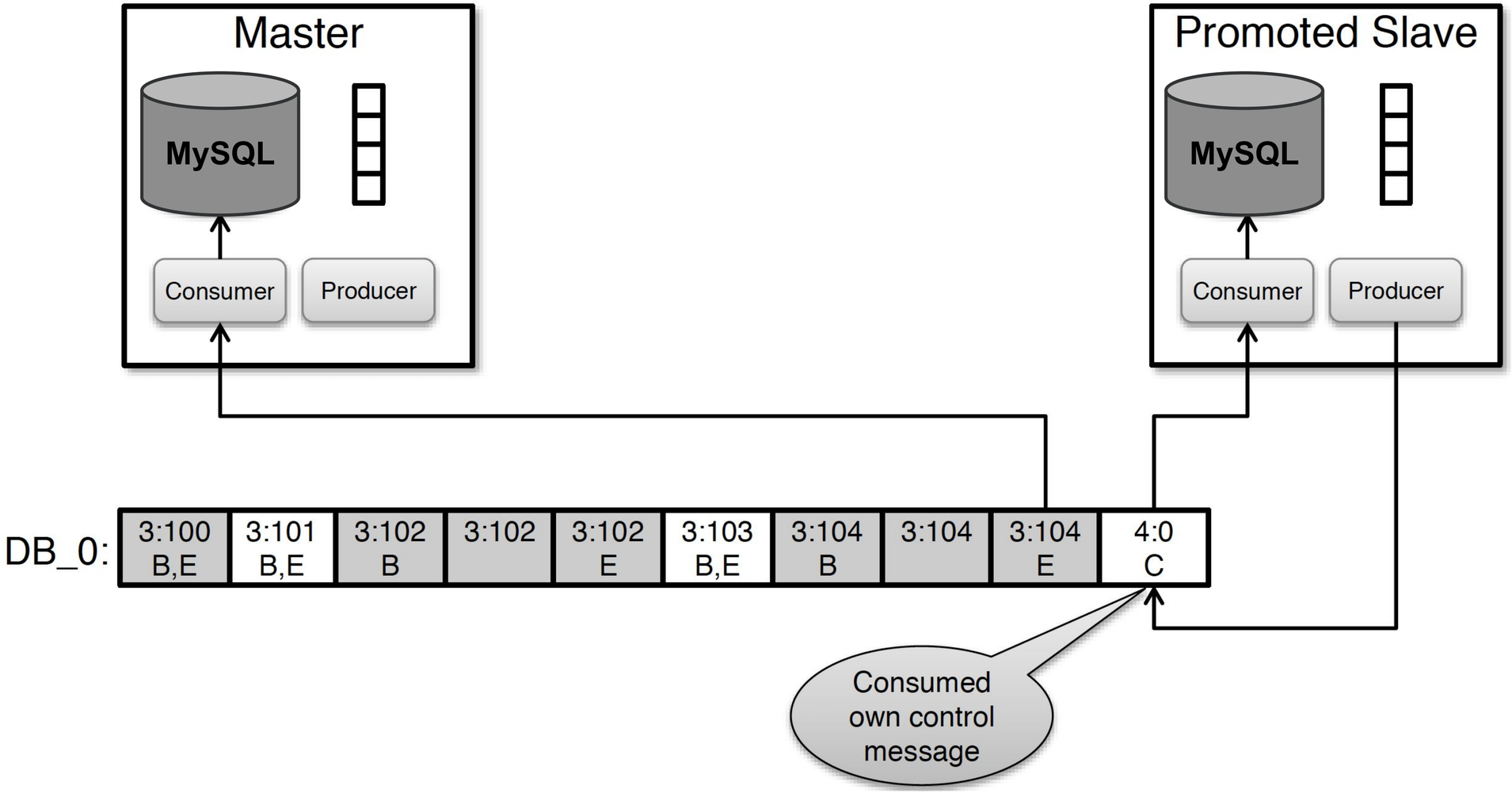
Message protocol



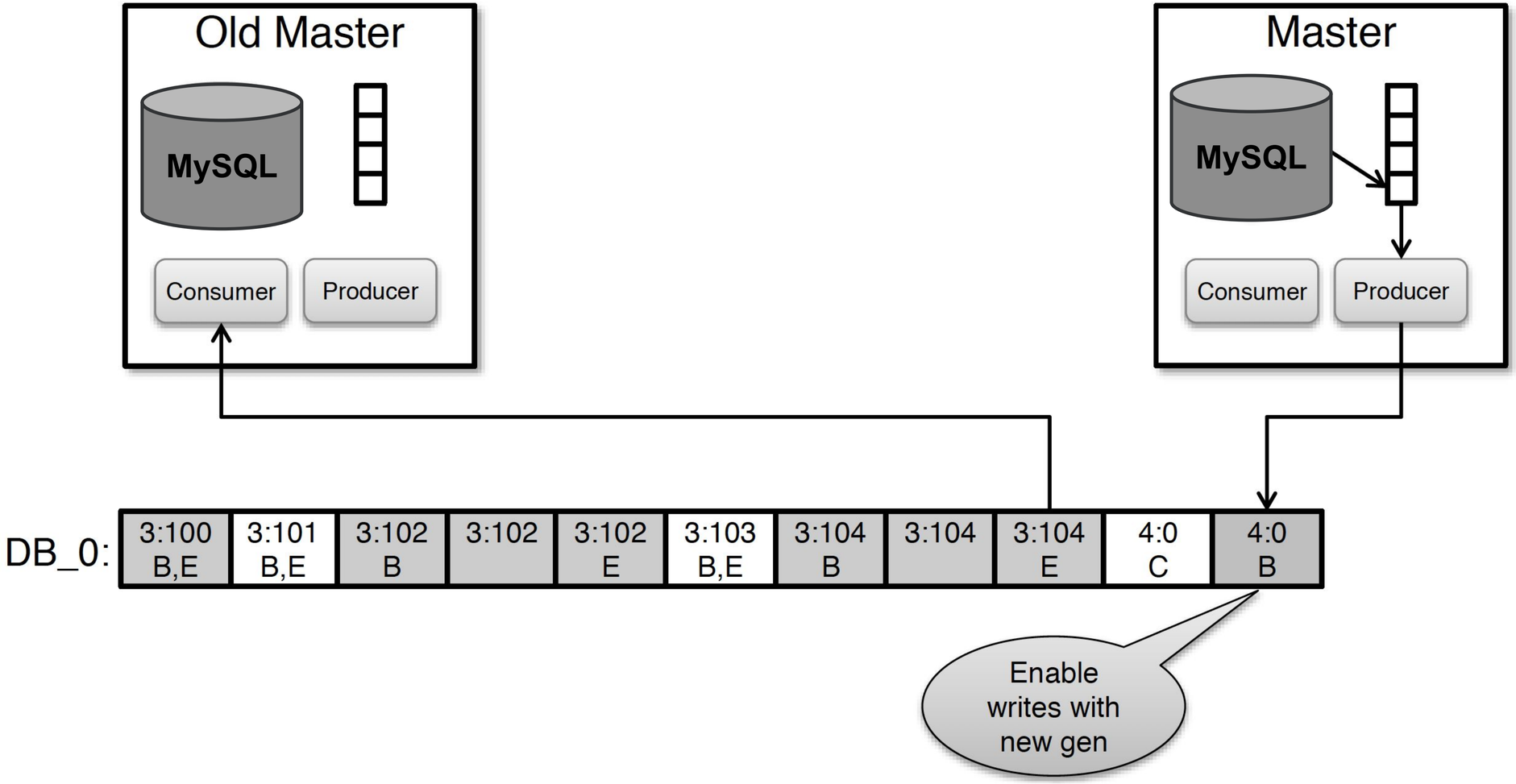
Message protocol - Mastership Handoff



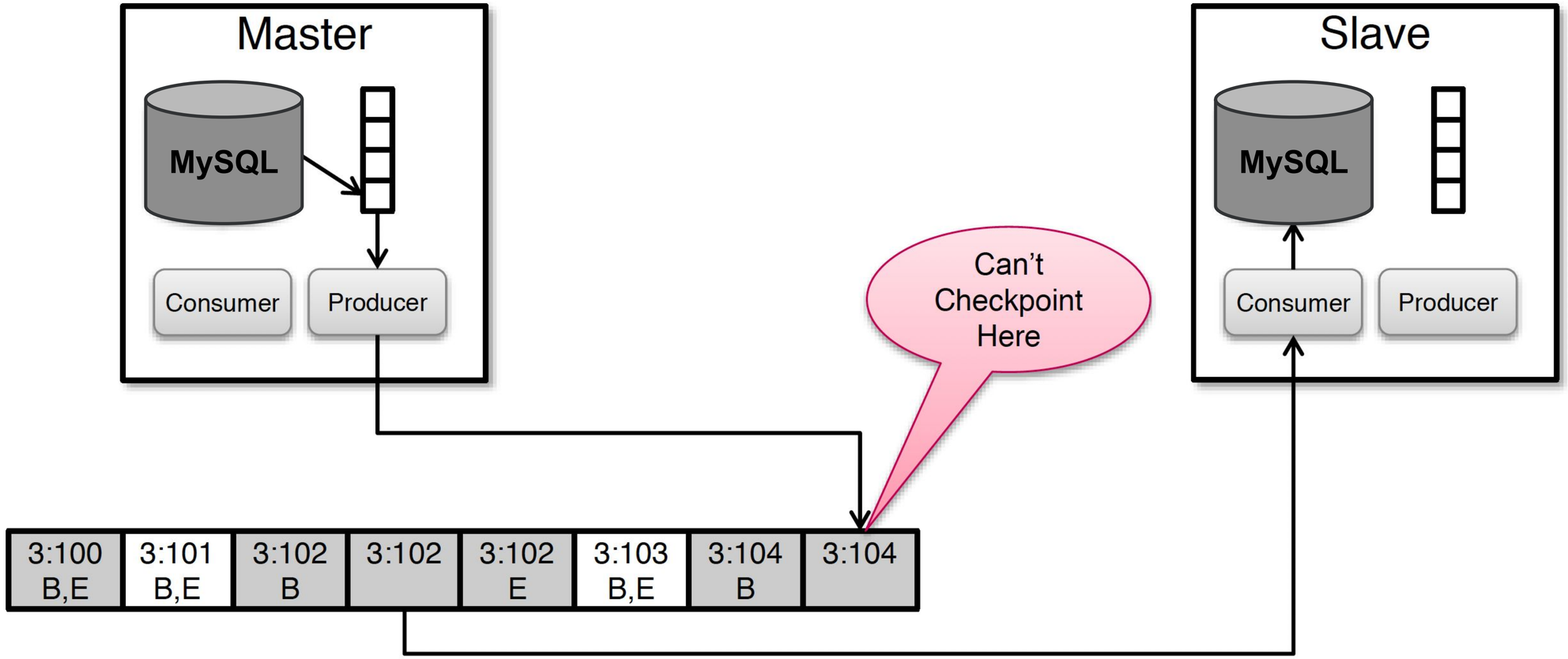
Message protocol - Mastership Handoff



Message protocol - Mastership Handoff

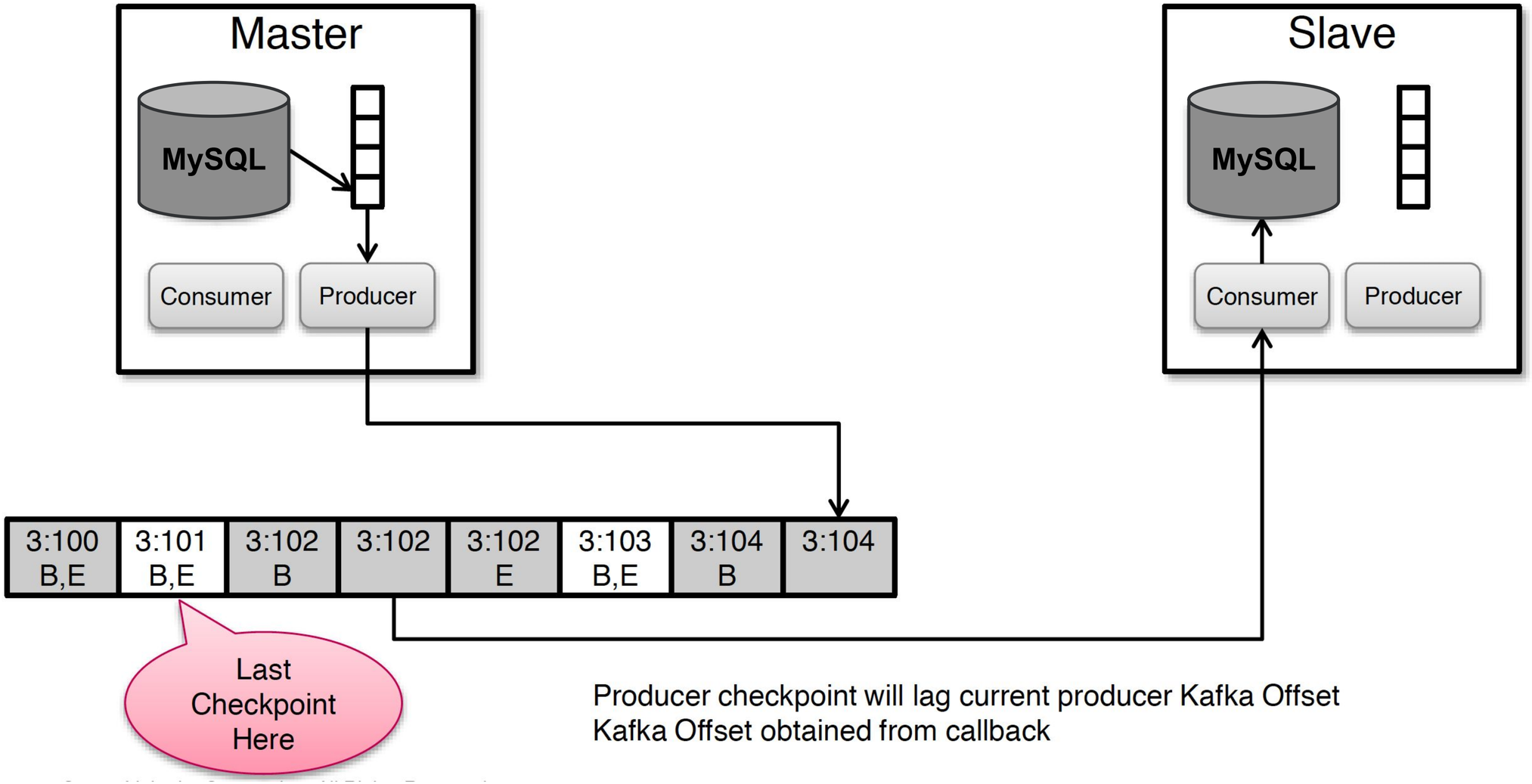


Checkpointing - Producer



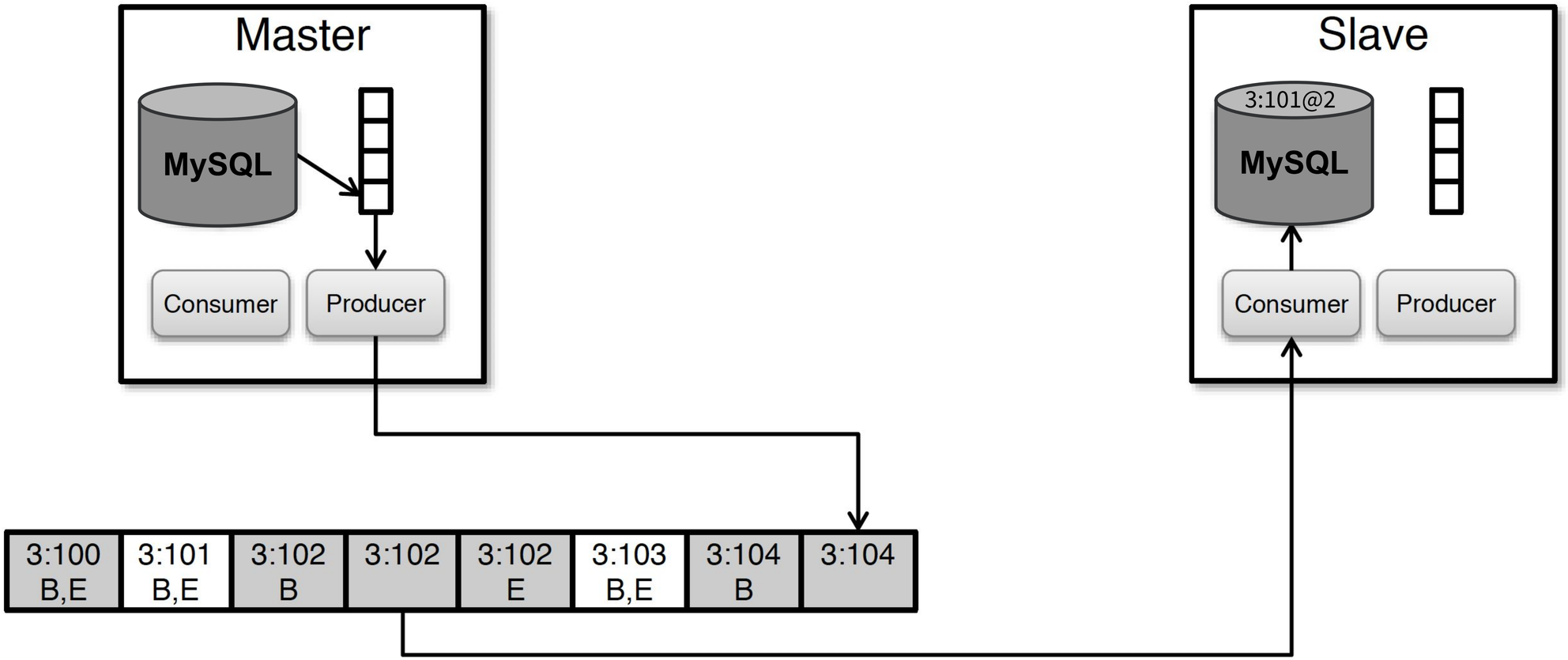
Periodically writes (SCN, Kafka Offset) to MySQL table
May only checkpoint offset at end of valid transaction!

Checkpointing - Producer ...



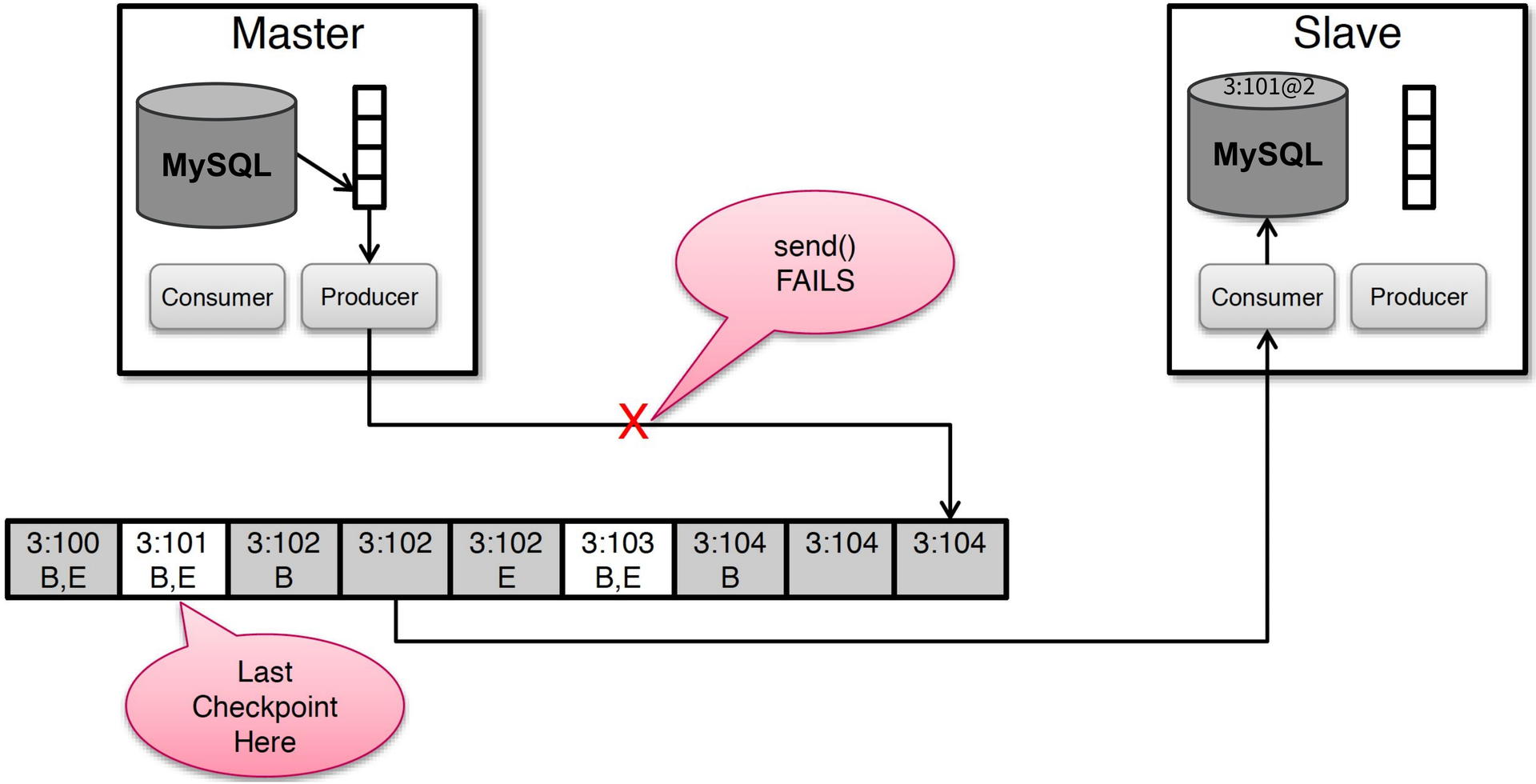
Producer checkpoint will lag current producer Kafka Offset
Kafka Offset obtained from callback

Checkpointing - Consumer

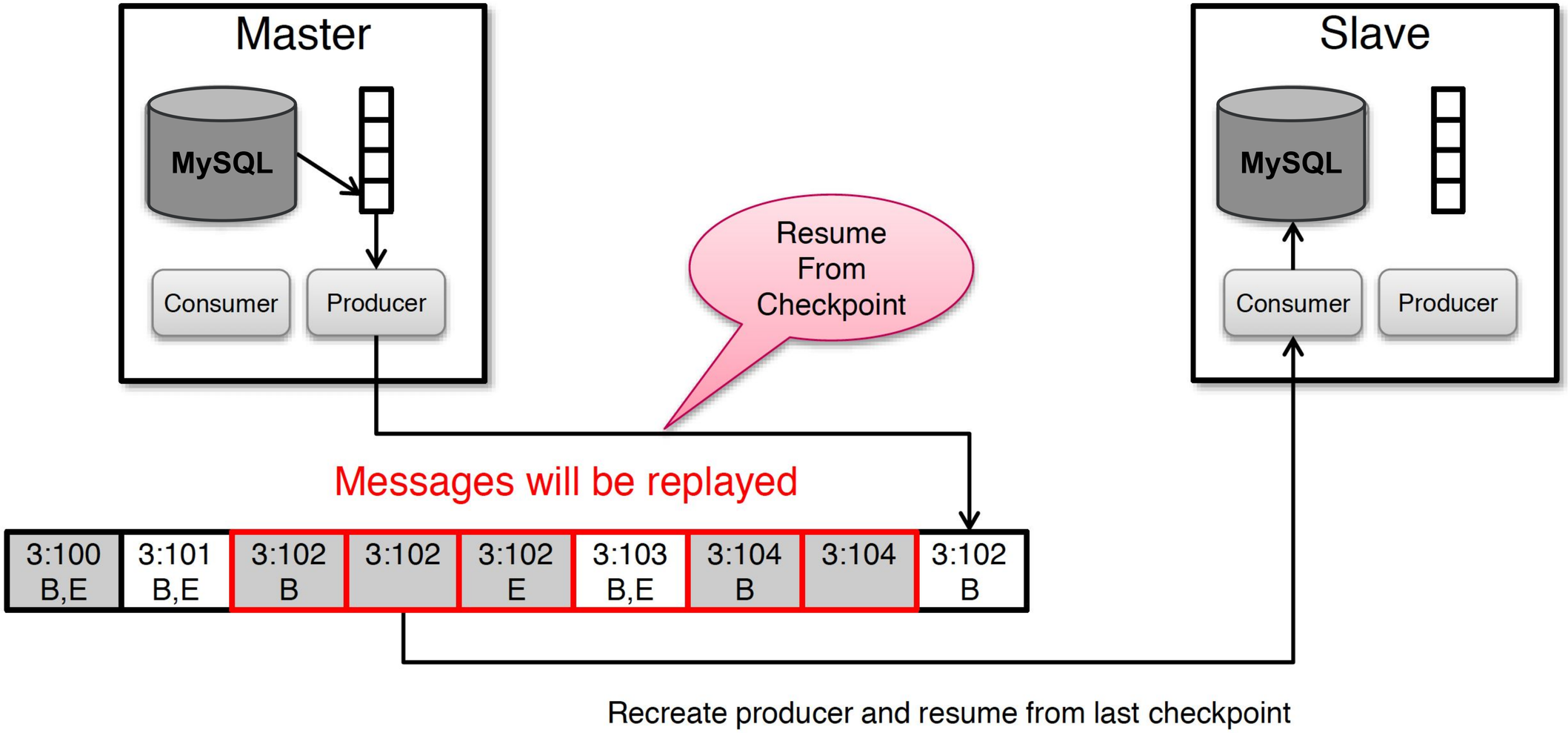


Slave updates (SCN, Kafka Offset) row for every committed txn

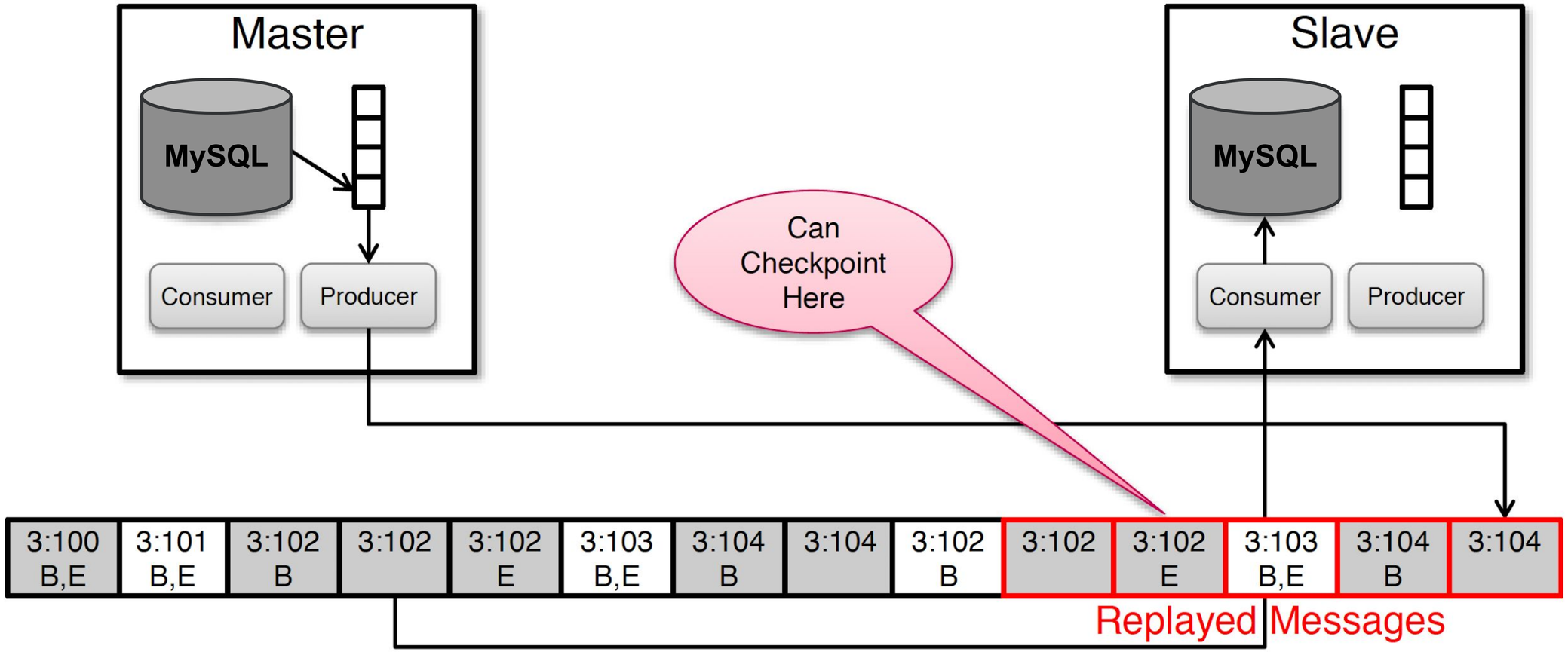
Producer Failure



Producer Failure...

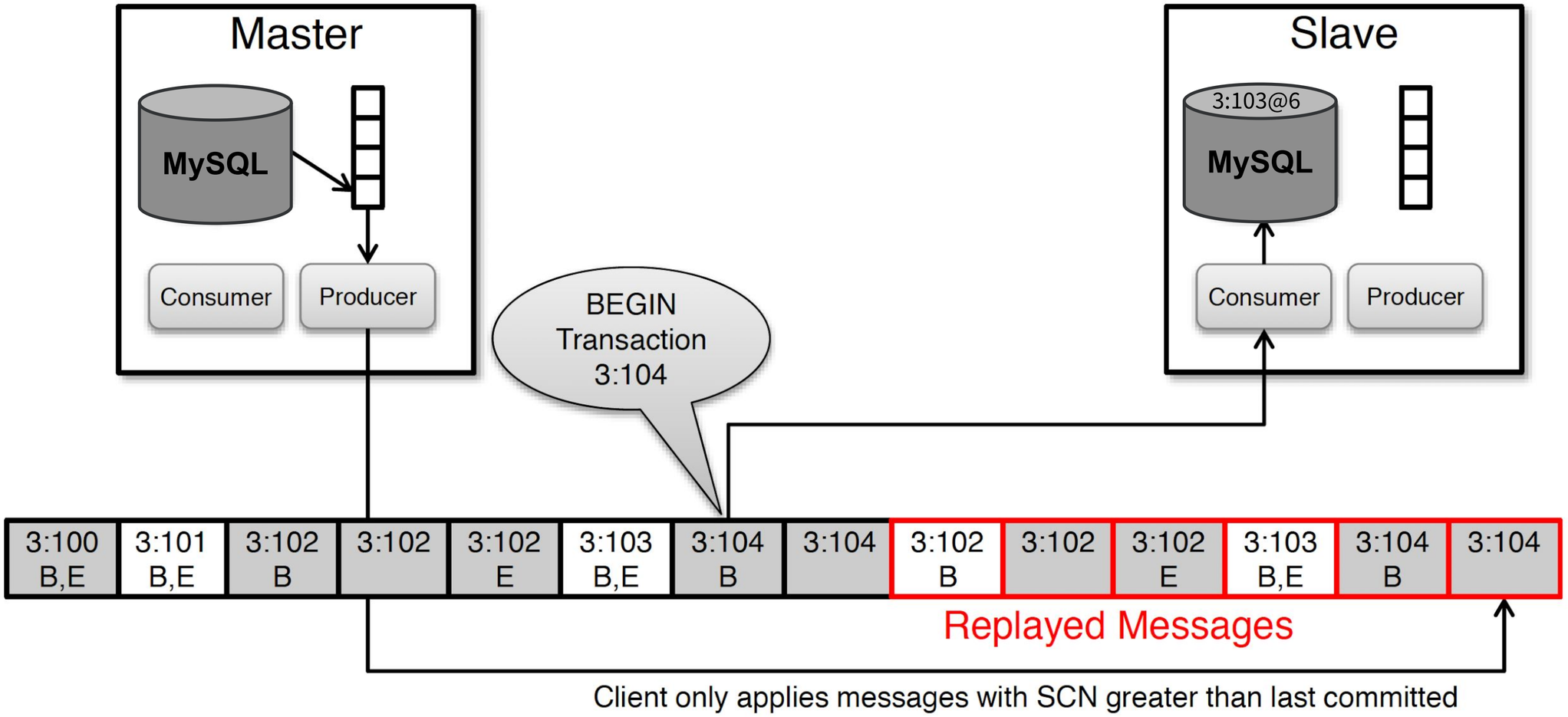


Producer Failure...

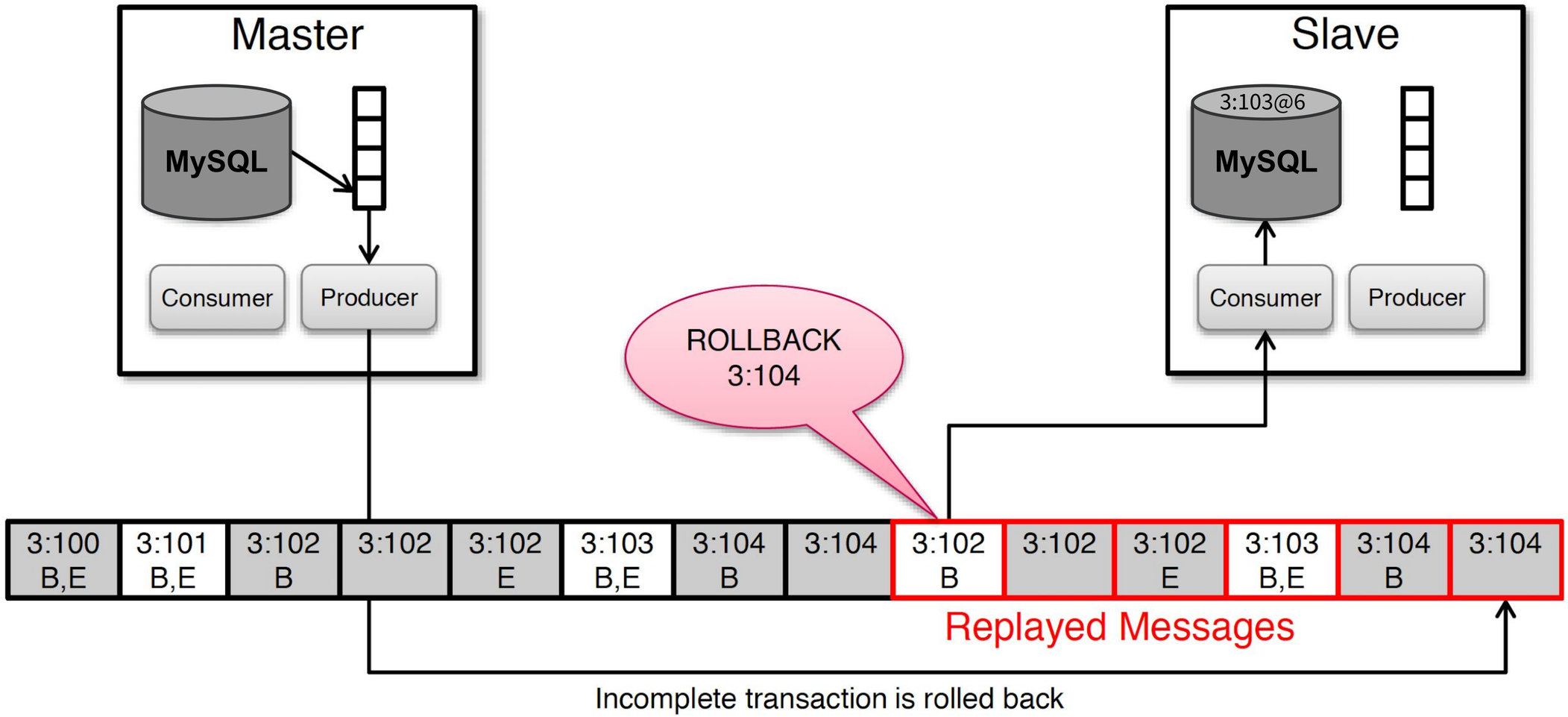


Kafka stream now contains replayed transactions
(possibly including partial transactions)

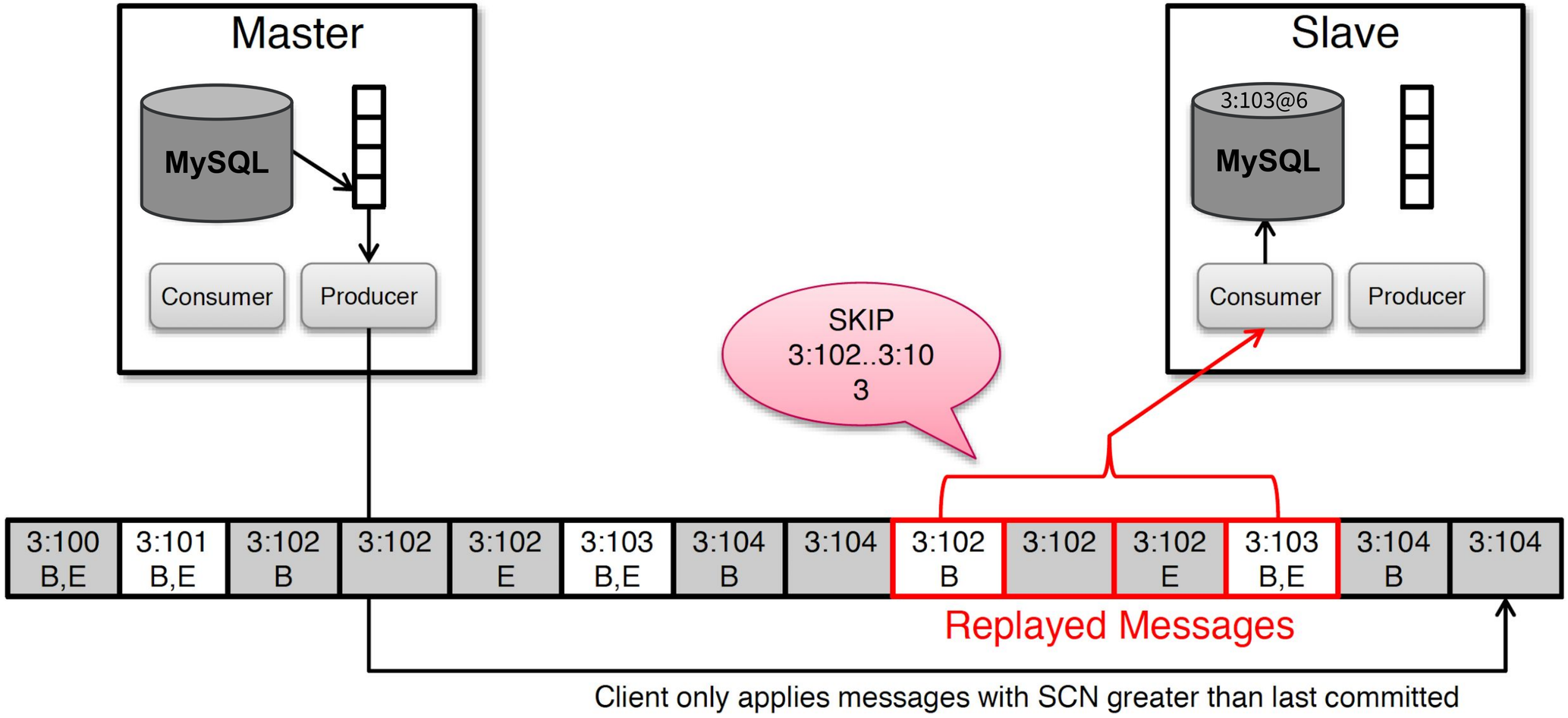
Producer Failure...



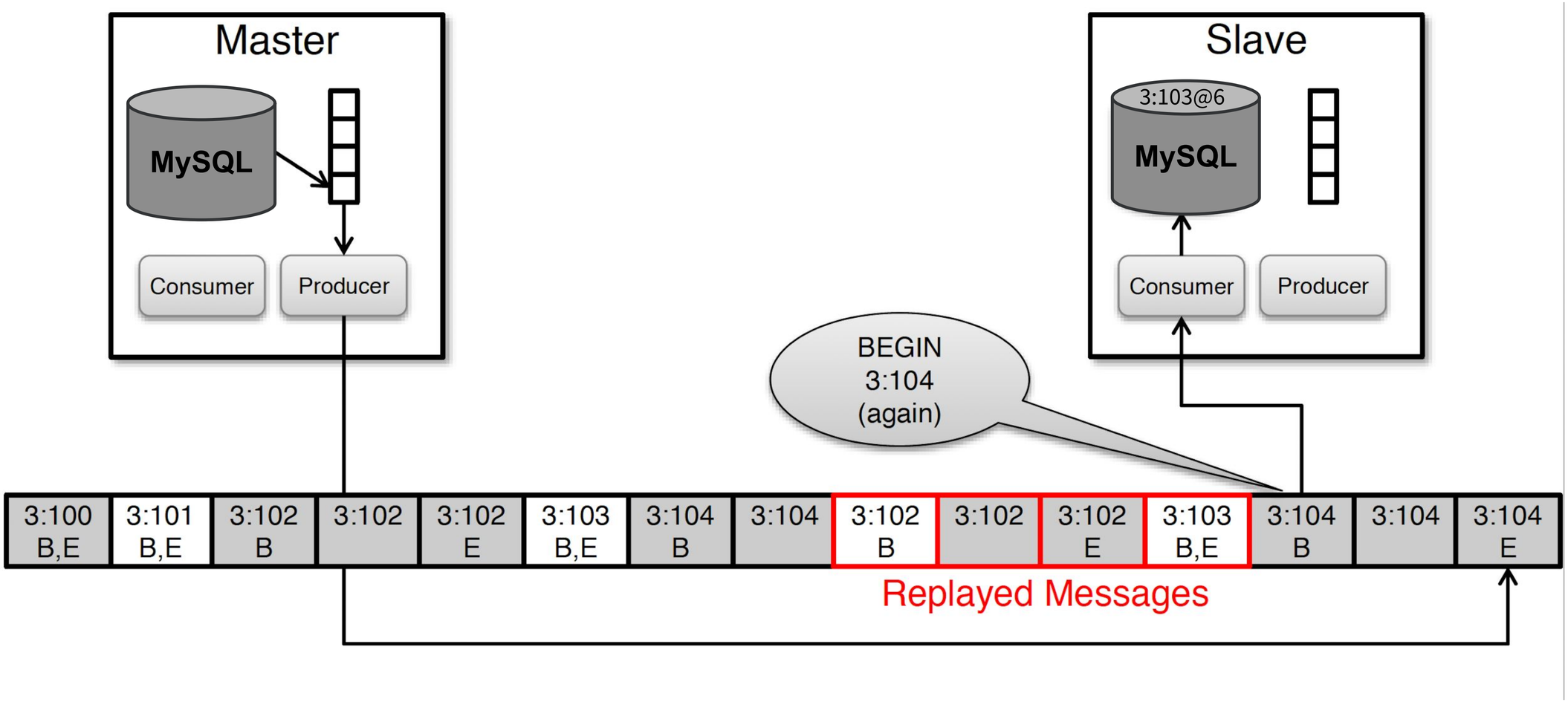
Producer Failure...



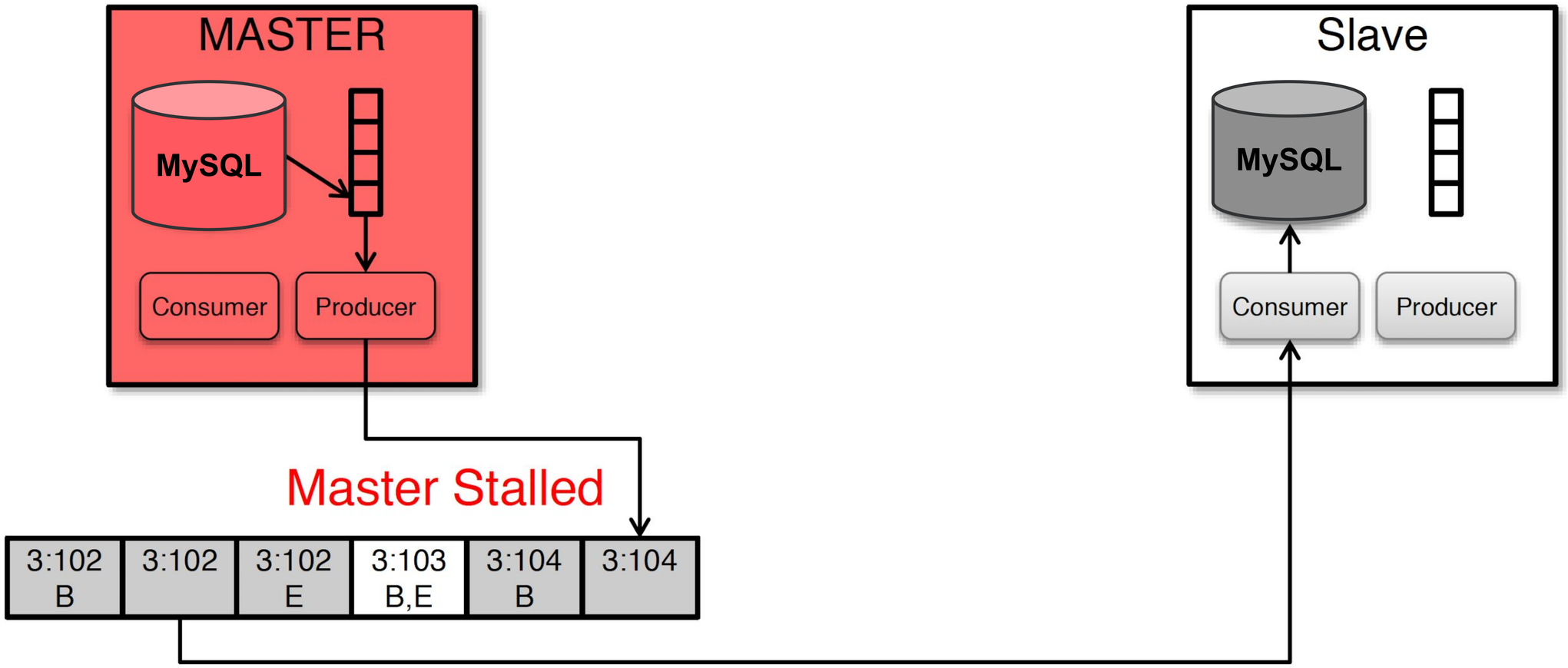
Producer Failure...



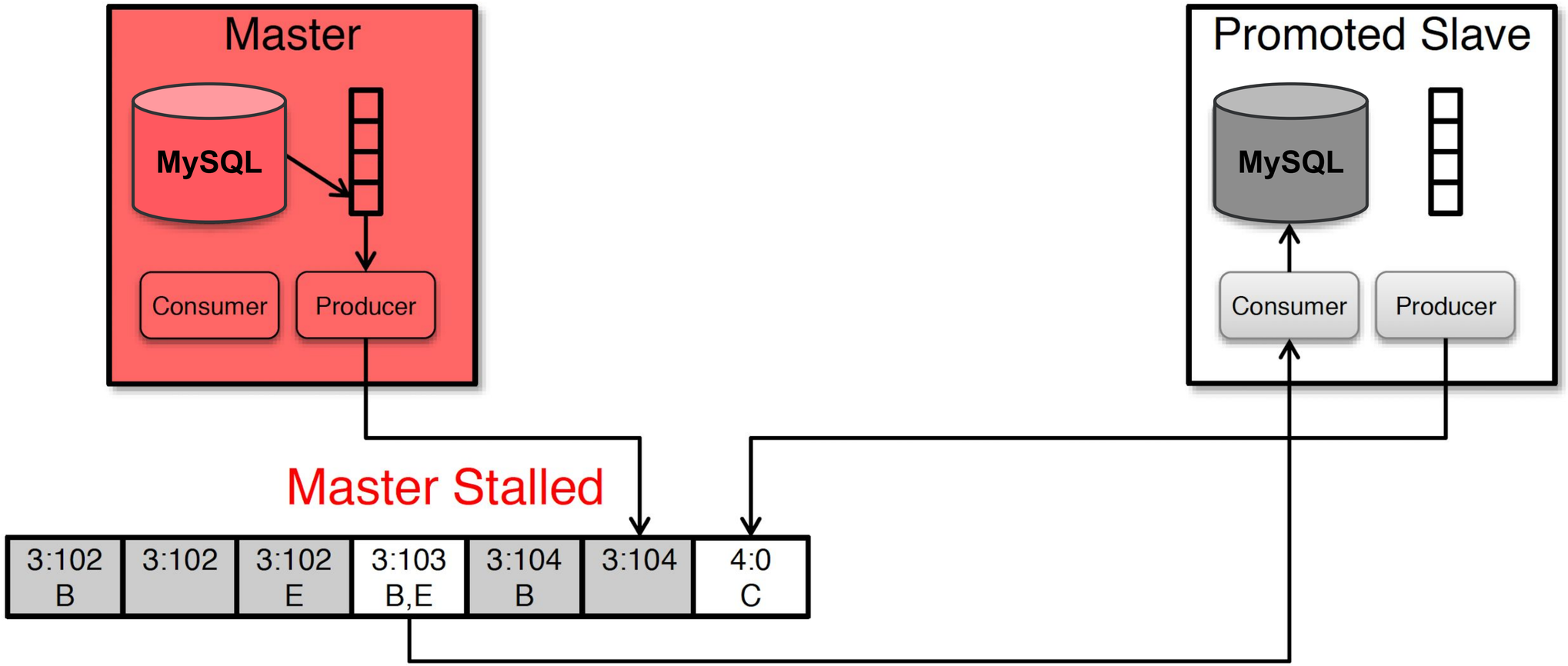
Producer Failure...



Zombie Writes

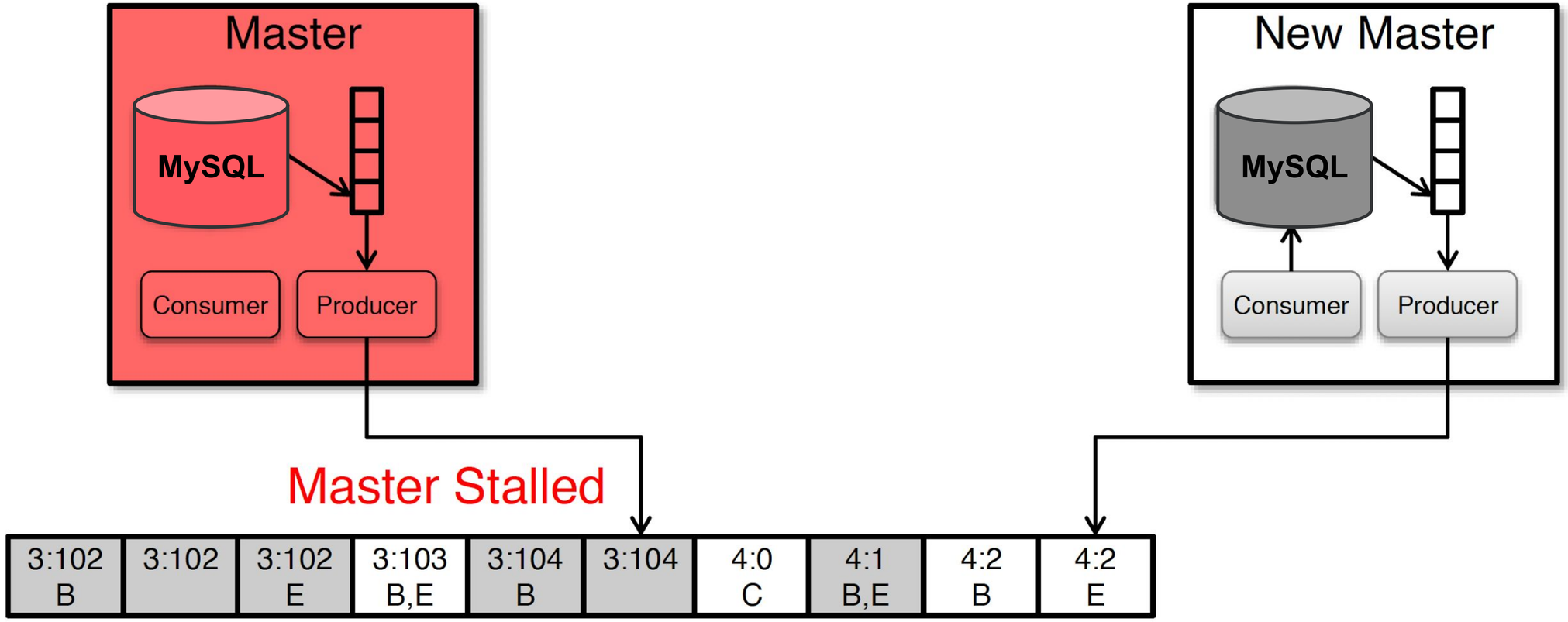


Zombie Writes...



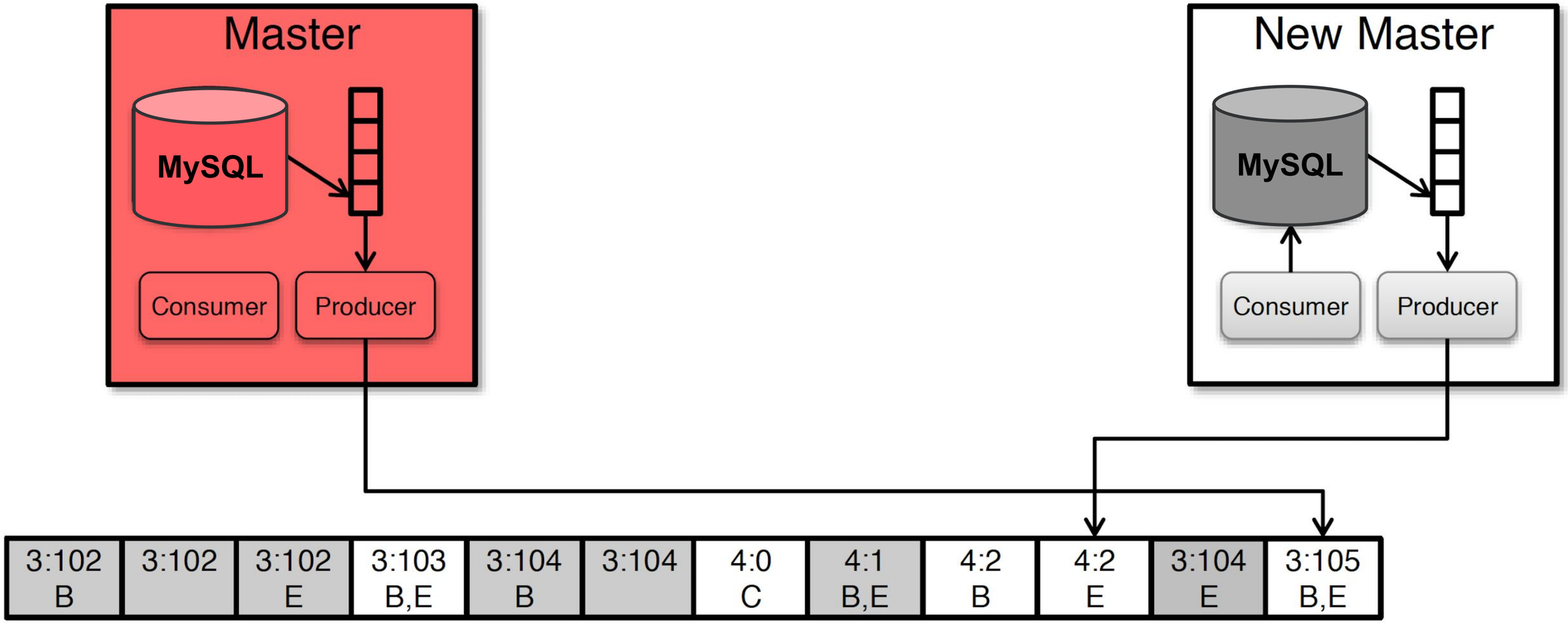
Helix sends SlaveToMaster transition to one of the slaves

Zombie Writes...



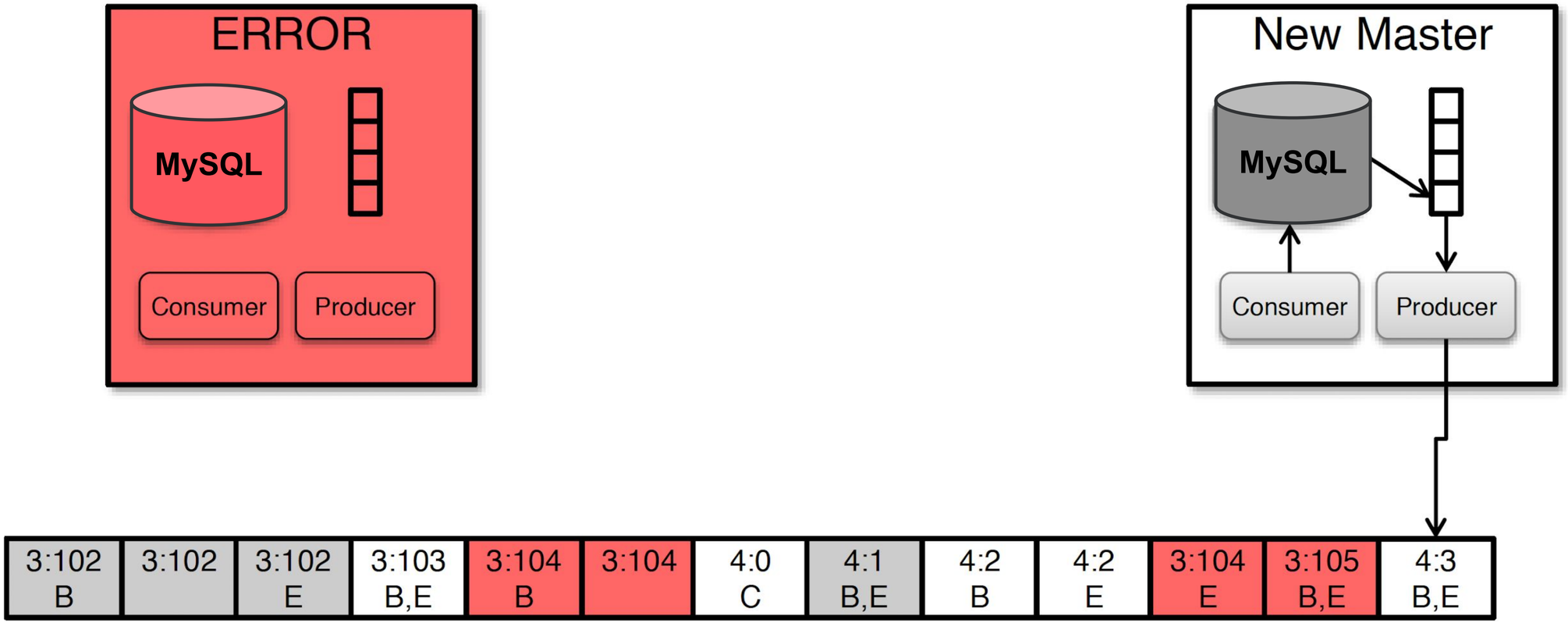
Slave becomes master and starts taking writes

Zombie Writes...



Stalled Master resumes and sends binlog entries to Kafka

Zombie Writes...



Former master goes into ERROR state
Zombie writes filtered by all consumers based on increasing SCN rule

Conclusion

- LinkedIn leveraged Kafka to scale Espresso
- Kafka helped to Unify data pipelines
- Reduced operational complexity
- Saved $$$

References

1. https://engineering.linkedin.com/espresso/introducing-espresso-linkedins-hot-new-distributed-document-store
2. https://engineering.linkedin.com/blog/2016/04/kafka-ecosystem-at-linkedin
3. https://www.slideshare.net/ConfluentInc/espresso-database-replication-with-kafka-tom-quiggle
4. https://www.slideshare.net/JiangjieQin/no-data-loss-pipeline-with-apache-kafka-49753844

Q&A?