Multi-colo Async Replication at LinkedIn

Karthik Appigatla

Staff SRE, LinkedIn

About Me

- Over a decade of experience in databases.
- Working for LinkedIn for last 5 years
- Authored MySQL 8 Cookbook

  https://www.amazon.in/MySQL-8-Cookbook-Karthik-Appigatla-ebook/dp/B0753D434Q
- LinkedIn - https://www.linkedin.com/in/appigatla

LinkedIn's newest data center in Oregon

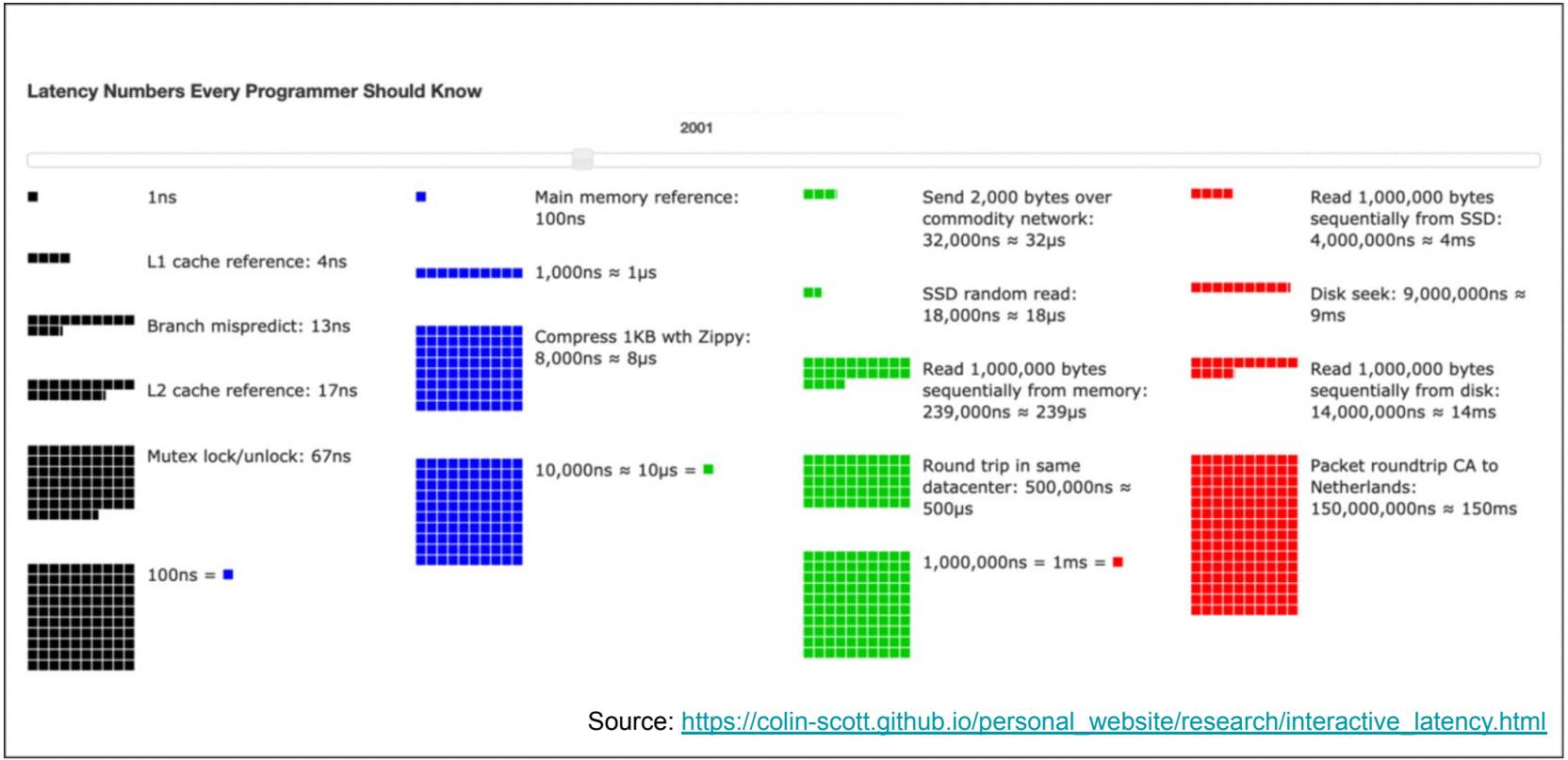
Data Center
a.k.a Colocation Center
a.k.a Colo

Agenda

- Need for Multi-colo
- How we leverage Multi-colo
- Managing Multi-colo Topology through Kafka
- Conflicts in Multi-colo
- Future Database Needs
- Q&A

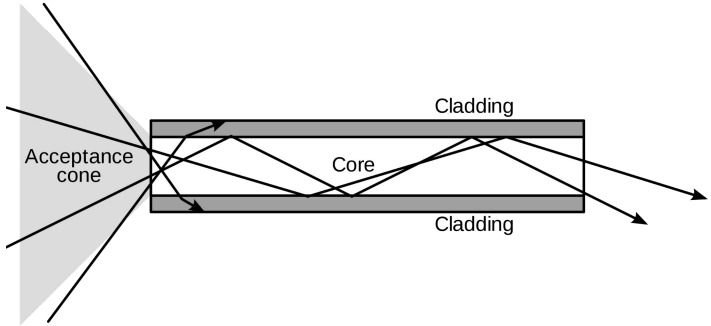
Need for Multi-colo

Latency numbers everyone should know



Source: https://colin-scott.github.io/personal_website/research/interactive_latency.html

Limited by Speed of Light



Light does not travel in absolute straight line inside a fibre optic cable, instead it bounces off the claddings.

Assuming best possible case

Aerial Distance between CA and Netherlands ~ 9000 Kilometers

Speed of light	~ 300000 Kilometers/second
Light takes 9000/300000	~ 30 **milliseconds to travel from CA to Netherlands**
Best possible case	~ **60 milliseconds (Round Trip)**

Still 60ms/0.5ms = 120 times slower than reading from same data center

How bad is it to read from remote colo

Round trip within same datacenter ..... 500,000 ns = 0.5 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns = 150 ms

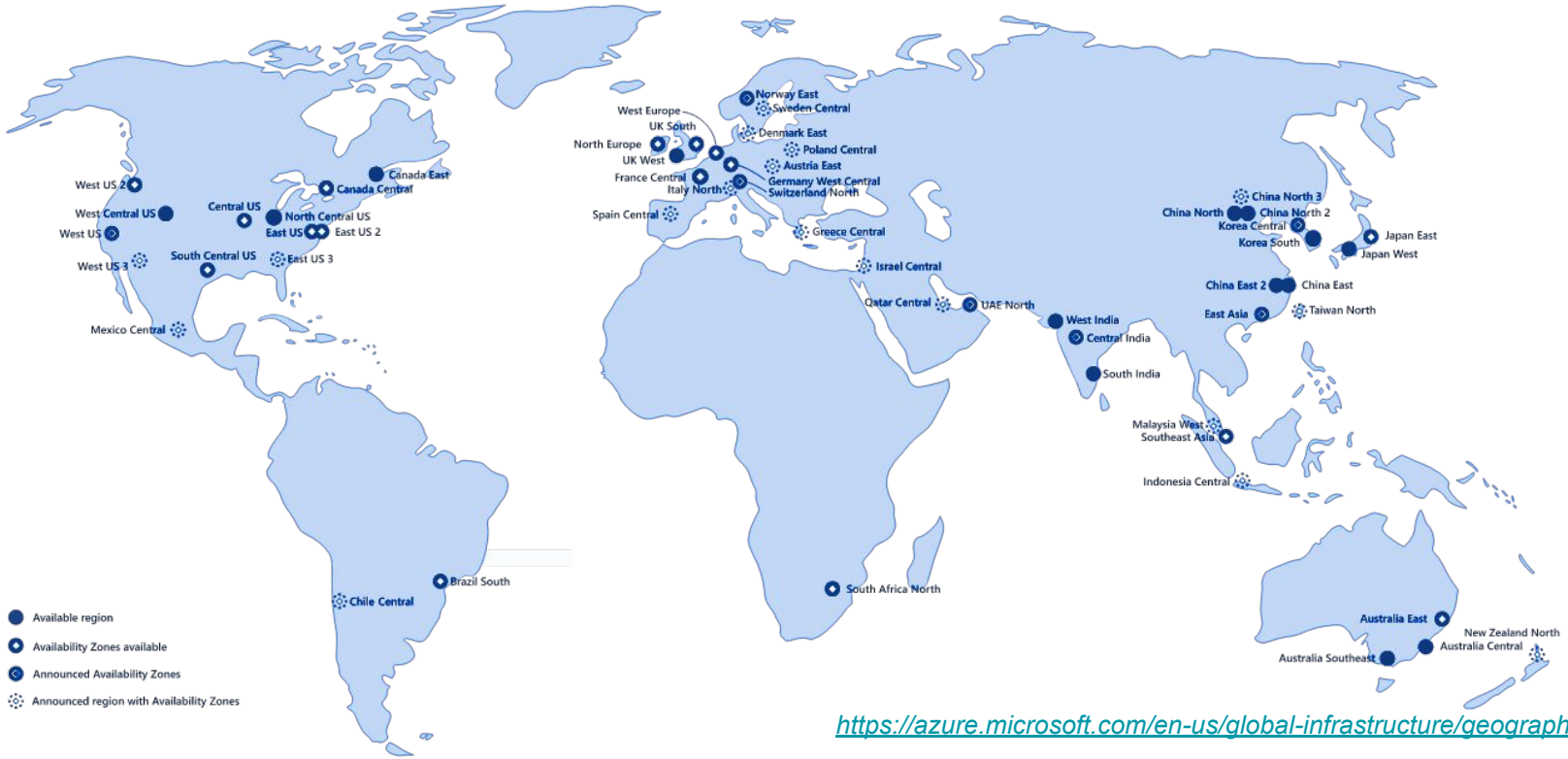
Reading from remote colo is 150ms/0.5ms = 300 times slower than reading from same colo

Read 1 MB sequentially from memory ..... 825,000 ns = 825 µs
Read 1 MB sequentially from disk .... 3,000 ns = 3 µs

Reading from magnetic disk is 825µs/3µs = 275 times slower than reading from main memory

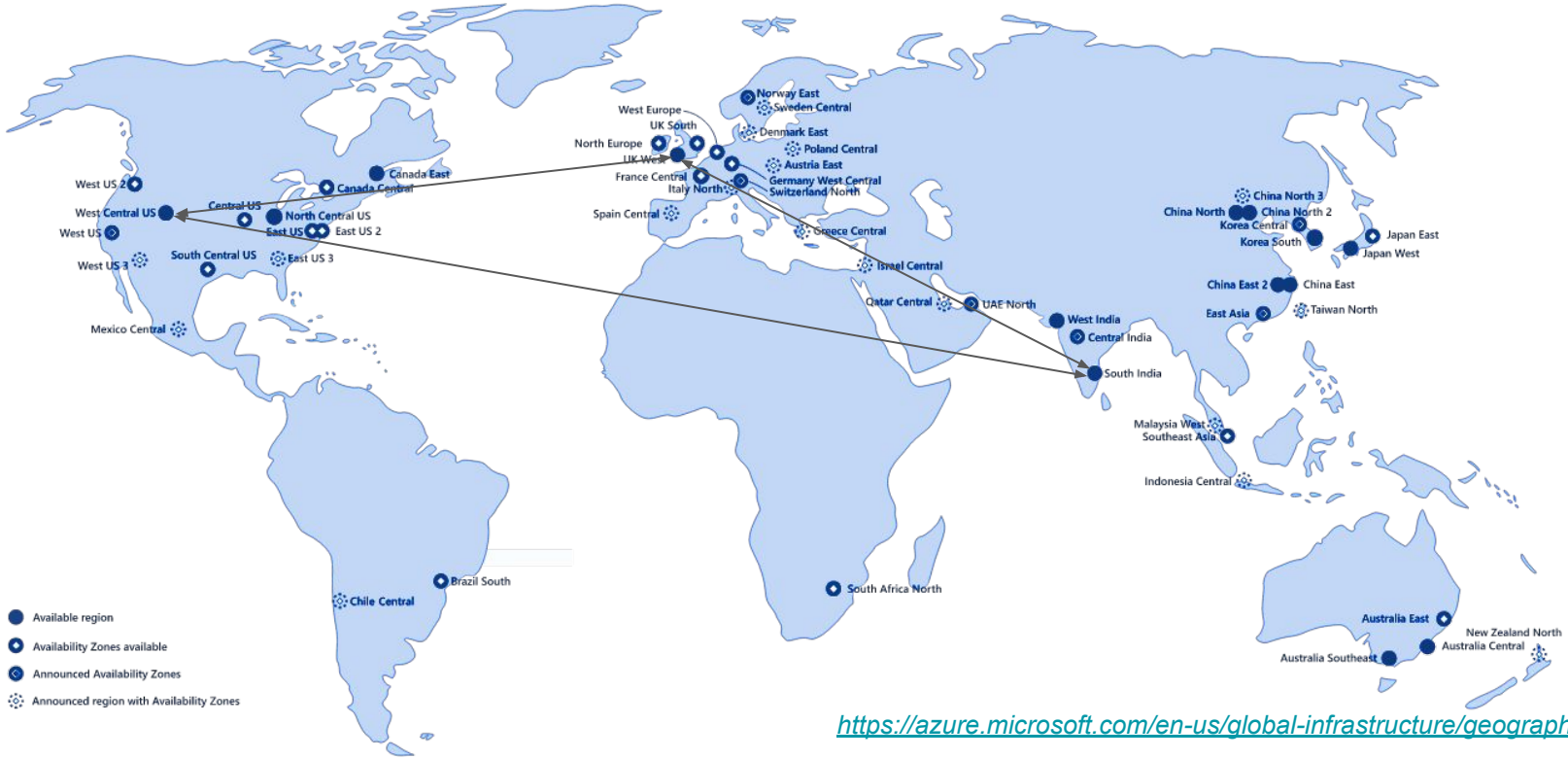
Performance impact of reading from magnetic disk rather than memory	~	Performance impact of reading from cross-colo which is 9000 KMs away

Multi-Colo



https://azure.microsoft.com/en-us/global-infrastructure/geographies/

Multi-Colo



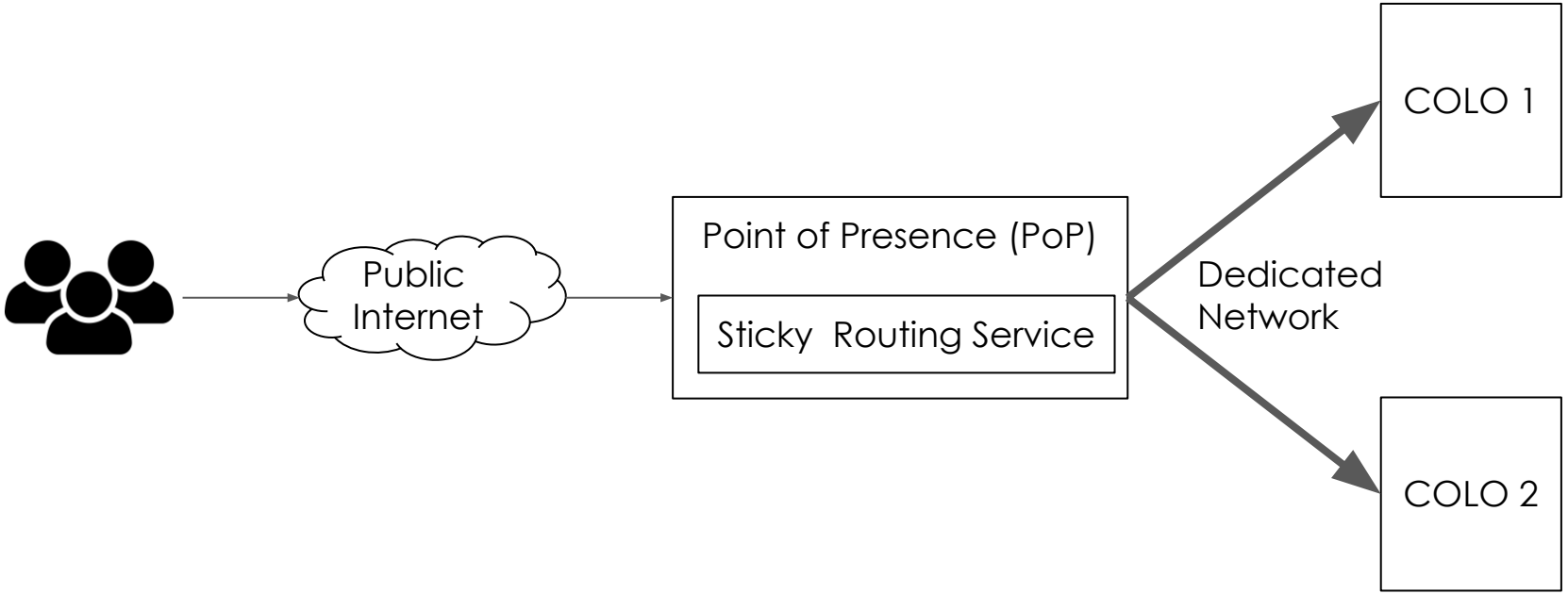
https://azure.microsoft.com/en-us/global-infrastructure/geographies/

How to leverage Multi-colo

Multi-Colo Advantages

- Low latency reads and writes across all geographic locations
- High Availability across regions
- Easy failover across regions

LinkedIn's EDGE



Multi-Colo Challenges

- Consistency
- Topology
- Conflicts

Multi-Colo Challenges

- Consistency
  - Strongly consistent within a colo
  - Eventually consistent across colos

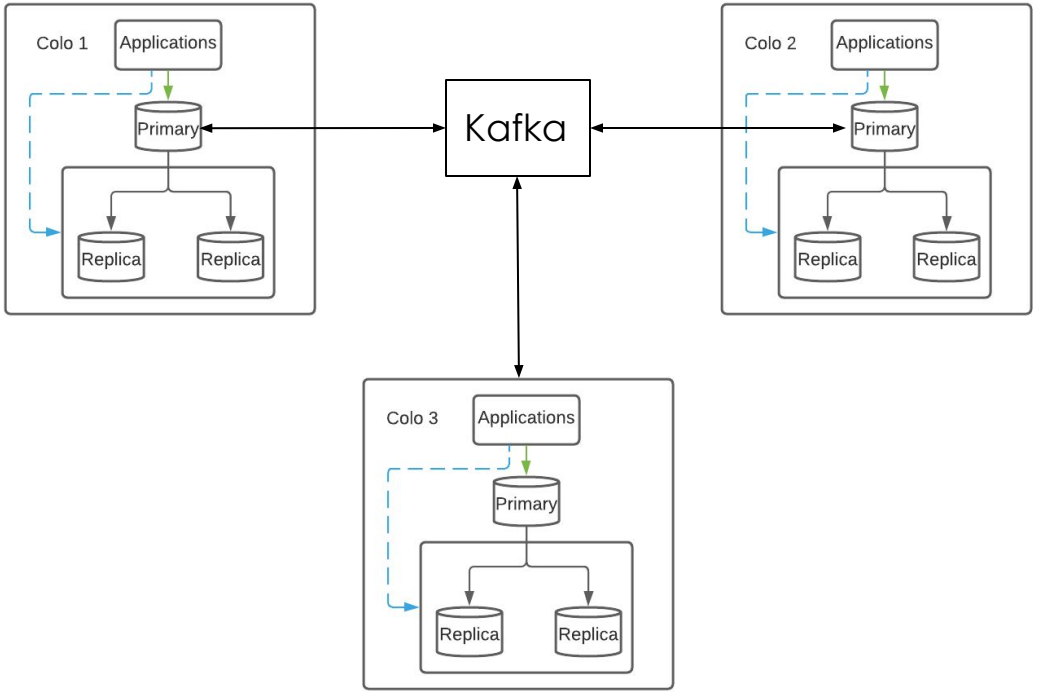
Multi-Colo Challenges

- Consistency
- Topology
- Conflicts

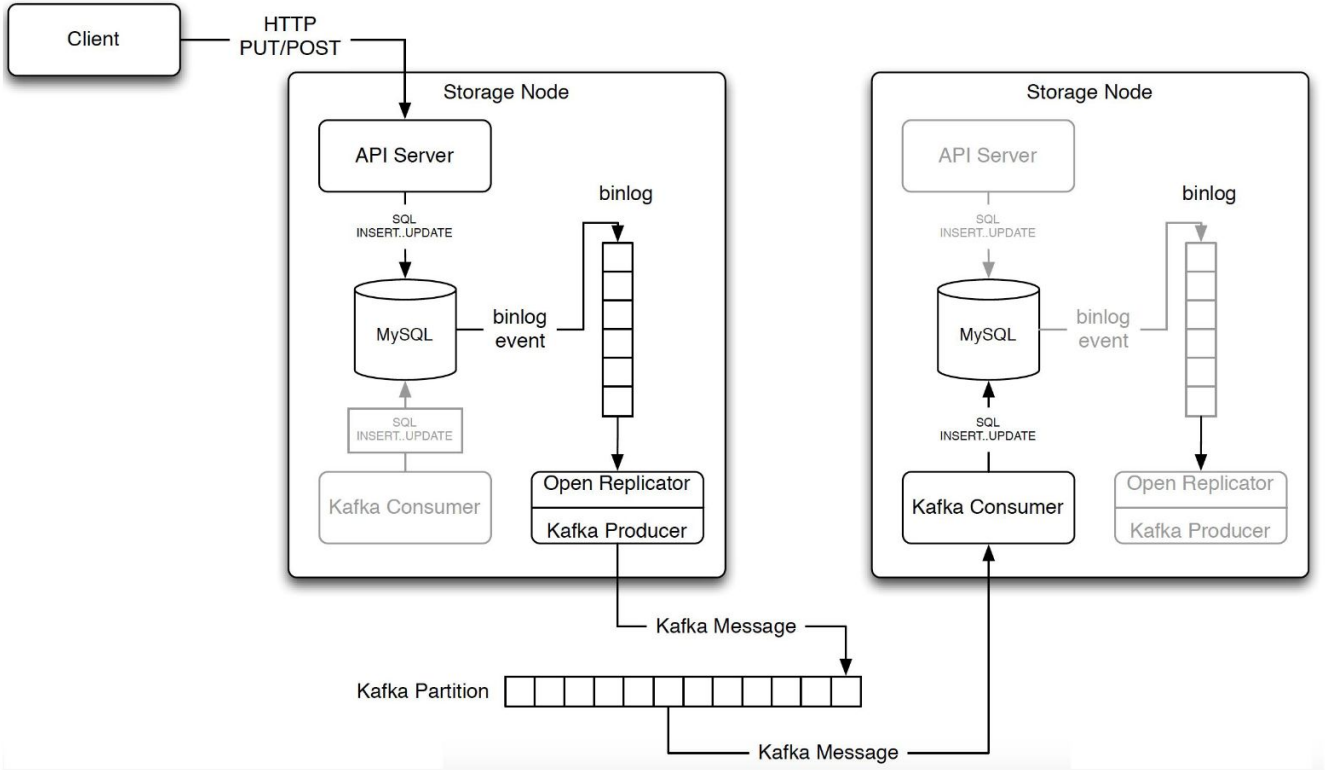
Multi-Colo Challenges

- Topology
  - Gets heavier as the number of colos increase
  - Solution: Kafka

Multi-Colo using Kafka



Multi-Colo using Kafka



Kafka Requirements

- Highly Available
- No Data loss
- No reordering
- Apply changes exactly once

Kafka Config

Kafka Config - Broker

- replication factor = 3
- min.isr = 2 (in sync replicas = 2)
- No unclean leader election

Kafka Config - Producer

- No data loss
  - block.on.buffer.full = TRUE
  - retries = Long.MAX_VALUE
  - acks = all
- No reordering
  - max.in.flight.requests.per.connection = 1
  - linger.ms = 0 (Deliver messages as soon as they are received, do not wait for batch)
- On on send failure
  - close producer in callback with close(0)
  - create new producer
  - resume from last checkpoint

Kafka Config - Consumer

- Exactly once
  - auto.offset.commit = 0
  - commit offsets only after the messages are processed
  - Do not process older GTIDs

Multi-Colo Challenges

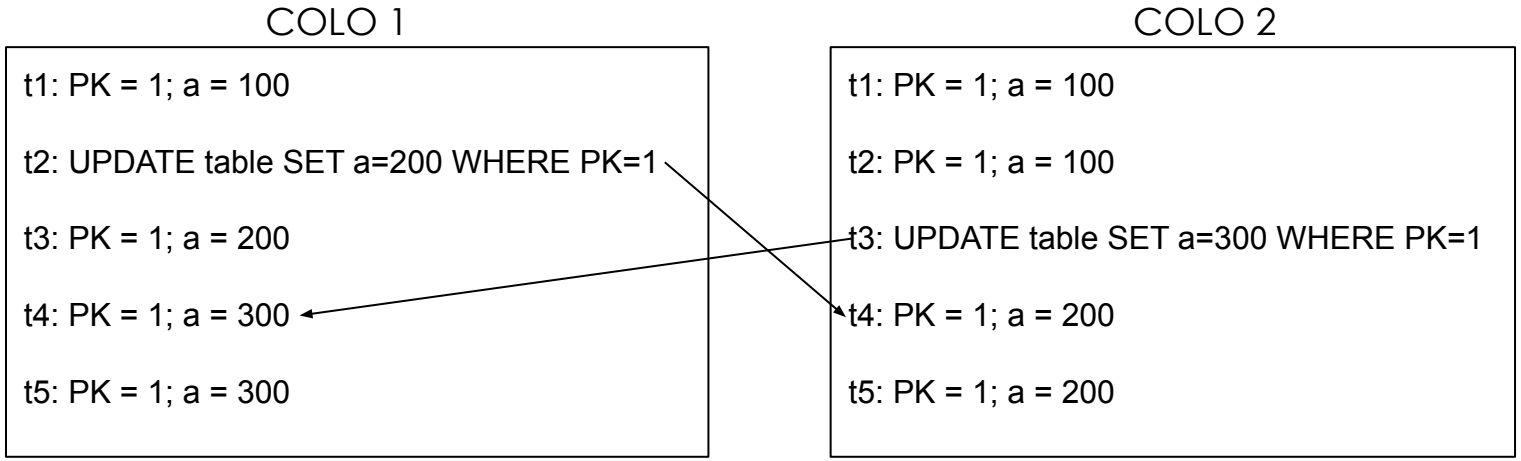
- Consistency
- Topology
- Conflicts

Multi-Colo Challenges - Conflicts

- Primary Key Collisions

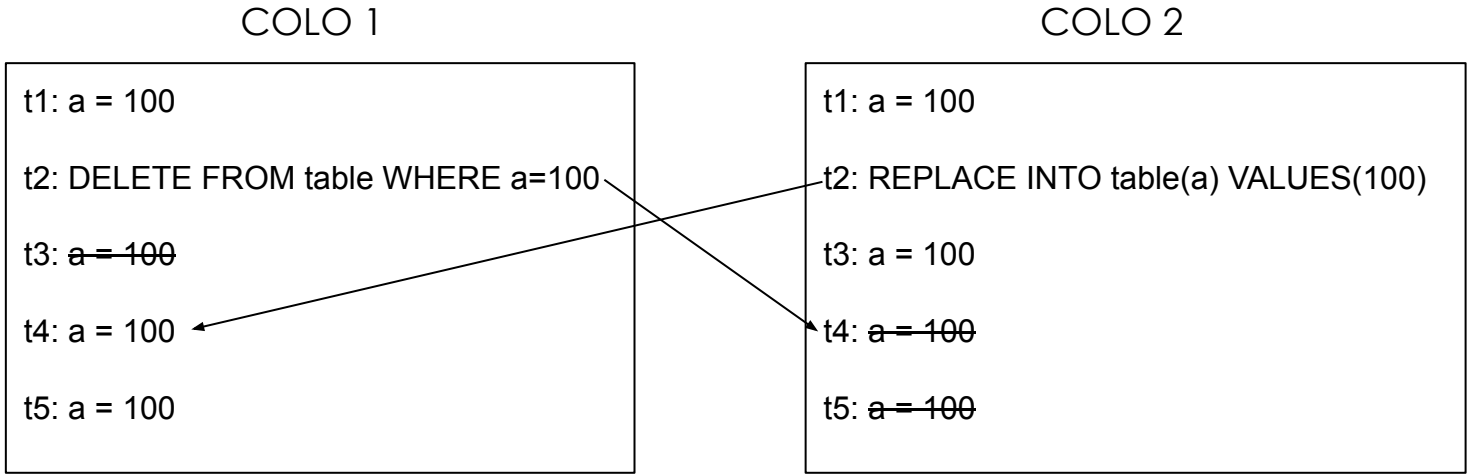
Multi-Colo Challenges - Conflicts

- Simultaneous Inserts/Updates in multiple colos



Multi-Colo Challenges - Conflicts

- Row Rebirth - Delete a record in one colo and replace in another colo



How to handle Collisions

- Primary Key Collisions
  - Global Key generator Service
  - Auto increment with colo name as prefix

    a1, a2, a3, a4... (for colo a)

    b1, b2, b3, b4...(for colo b)
  - Auto increment with offsets

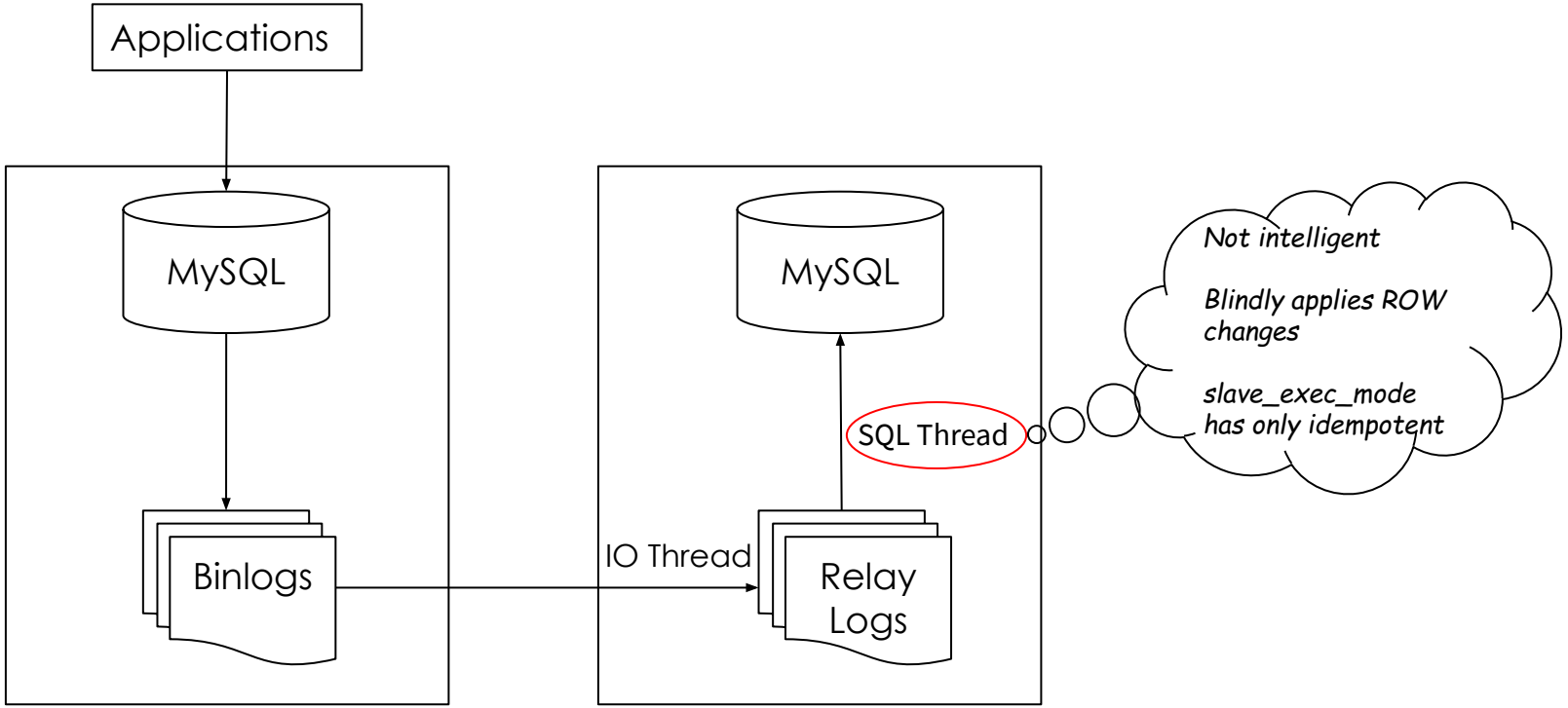
    5n, 5n+1, 5n+2, 5n+3, 5n+4 (if 5 colos are there)
  - UUID

How to handle Collisions

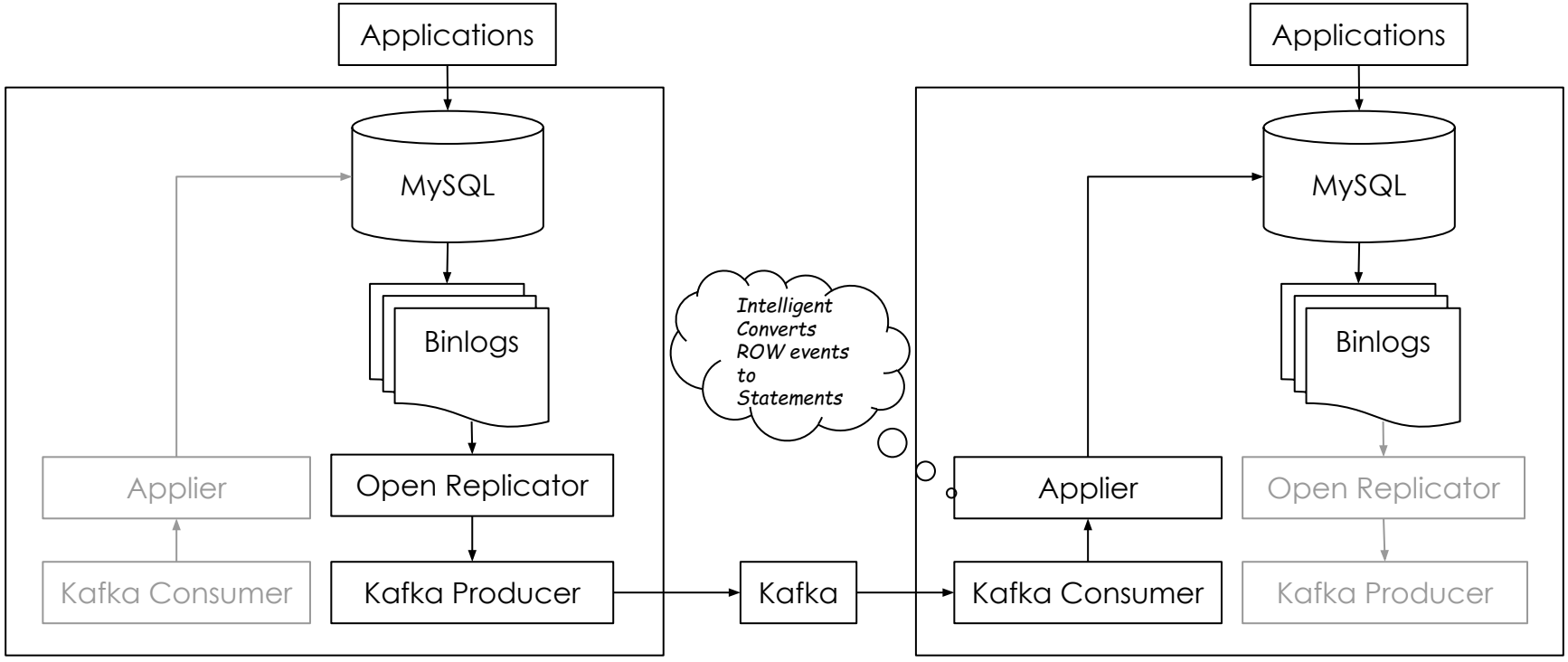
- Conflicts Resolution Mechanisms
  - Last writer win (Max value of a timestamp column)
  - First writer win (Min value of a timestamp column)
  - Delta (Append to existing value)

Multi-Colo Implementation

Traditional Replication



Multi-Colo Implementation



Applier Mechanism

- Converts ROW event to STATEMENT
- Behaves as an application and writes to the database
- Handles failures based on the desired conflict resolution
- Update - “Row already exists”
  - Last written wins
- Insert Failure - “Row already exists”
  - Converts Insert to Update and applies based on last writer win
- Update Failure - “Row does not exist”
  - Converts Update into Insert

Multi-Colo Schema Requirements

- Additional Columns on each table
  - Last modified column
  - Status
- Triggers on each table
  - Before Insert
  - Before Update
- ROW Image FULL

Multi-Colo Schema Requirements

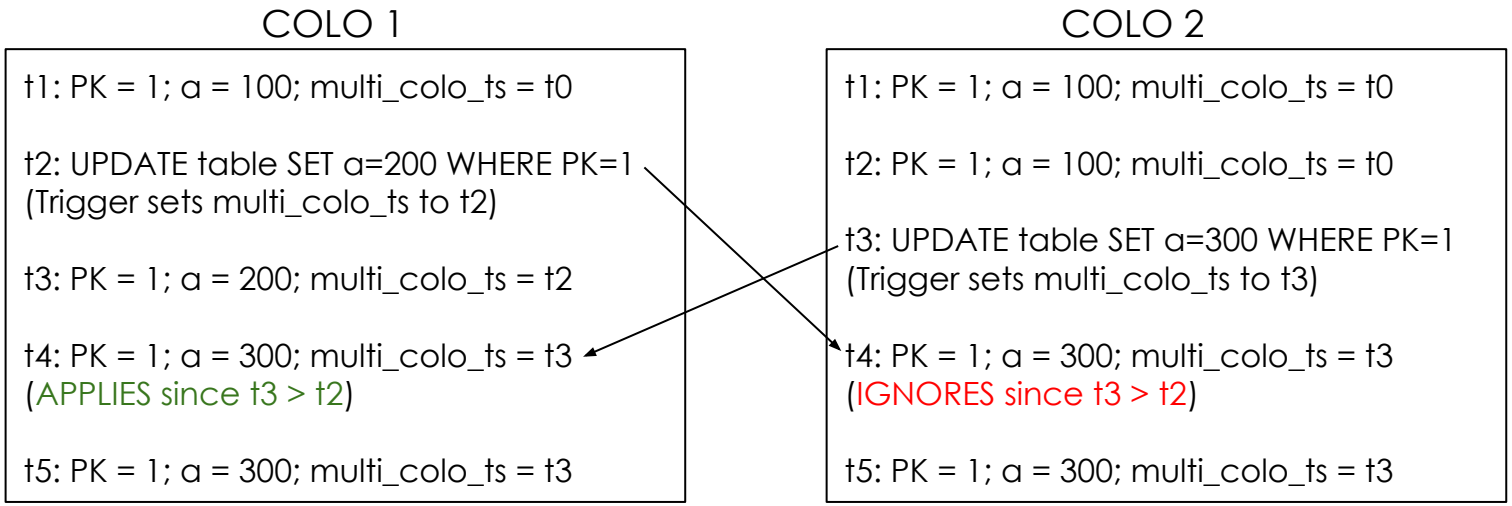
```
/* Multi-colo columns */
   mulit_colo_ts          DATETIME        DEFAULT CURRENT_TIMESTAMP,
   multi_colo_status    VARCHAR(1)      DEFAULT 'A',
```

Multi-Colo Schema Requirements

```
/* Multi-colo Triggers Before Insert */
DROP TRIGGER IF EXISTS <database_name>.<table_name>_MULTI_COLO_BEFORE_INSERT;
DELIMITER //
CREATE TRIGGER <database_name>.<table_name>_MULTI_COLO_BEFORE_INSERT
   BEFORE INSERT
   ON <database_name>.<table_name>
   FOR EACH ROW
BEGIN
   IF user() LIKE 'multi_colo_user@localhost' THEN
      SET new.multi_colo_status := 'M';
   ELSE
      -- :new comes from an application write
      SET new.multi_colo_status := 'A';
      SET new.multi_colo_ts = NOW(6); -- micro seconds
   END IF;
END //
```

Multi-Colo Challenges - Conflicts

- Simultaneous Inserts/Updates in multiple colos



- At t3, colo1 and colo2 are inconsistent
- At t5, they are eventually consistent

Multi-Colo Challenges - Conflicts

- Application Guidelines to avoid conflicts
  - All tables should have PK
  - No update on PK
  - No Unique Key
  - No hard deletes (Row-rebirth)
  - No Foreign Key

Future Database Needs

Problems with current architecture

- Can’t leverage Multi-Threaded-Replication
- Overhead to resolve conflicts

Why Clustering does not solve the problem?

- Synchronous
- Adds latency because of certification process

Future Database Needs

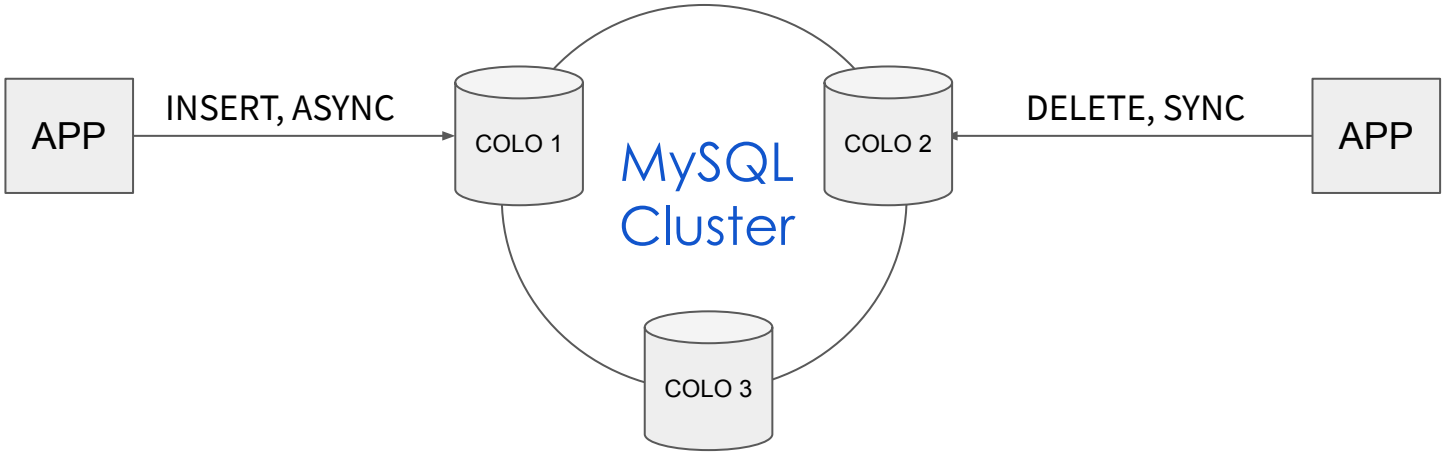
- Provide mechanisms to resolve conflicts automatically
  - /etc/my.cnf

```
default_conflict_resolution          = 'last_writer_win'
conflict_resolution_table            = "company.emp=first_writer_win"
conflict_resolution_table            = "company.emp=first_writer_win"
conflict_resolution_ts_column        = 'multi_colo_ts'
conflict_resolution_status_column = 'multi_colo_status'
```

```
CREATE TABLE `emp` (
  `id` int unsigned NOT NULL,
  `name` varchar(15) NOT NULL,
  `sal` int unsigned NOT NULL,
  `multi_colo_status` enum('A', 'M'),
  `multi_colo_ts` TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;
```

Future Database Needs

- Clustering solutions could have options to
  - Disable certification
  - Choose consistency



Thank You!

Q & A