CDC on Statement Based Replication (SBR)

Venkat Morampudi, Staff Software Engineer

Agenda

- Introduction to CDC
- MySQL Replication Streams
- Implementing CDC on SBR
- CDC at Box

Introduction to CDC

Change Data Capture (CDC)

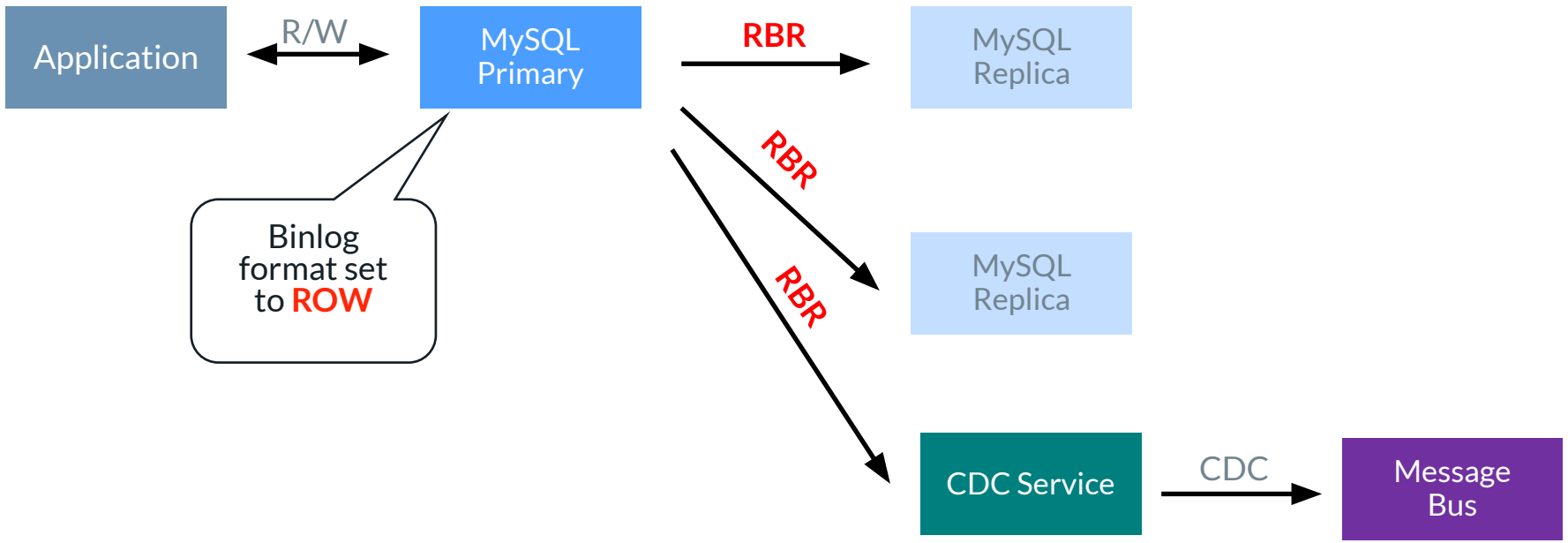
Change Data Capture is a design pattern that enables capturing changes to data and notifying actors so they can react accordingly

Anatomy of a CDC event

- CDC event is composed of
  - Pre-mutation state of the row (before)
  - Post-mutation state of the row (after)
  - Metadata
    - Table
    - Primary key
    - Mutation type


```
{
 "metadata": {
  ”primary_key": 2,
  ”table": "fruit",
  "mutation_type": ”update",
  "timestamp": 1611179777
 },
 ”before": {
  "id": 2,
  "name": ”banana",
  "quantity": 950,
  ”expiry_date": “2020-01-02”
 },
 ”after": {
  "id": 2,
  "name": ”banana",
  "quantity": 720,
  ”expiry_date": “2020-01-02”
 },
}
```

Typical CDC Pipeline Architecture for MySQL



Different Types of MySQL Replication Streams

- Row-based Replication (RBR)
- Statement-based Replication (SBR)

Row-based Replication (RBR)

- The binary log stores the record-level changes that occur to database tables
- State of the row before and after can be extracted from the Binlog event
- Binlog event doesn't contain the table metadata, i.e. it does not record the field names, only field number.

query

```
UPDATE `super_market`.`fruit`
SET quantity= quantity + 300
WHERE name = 'pear'
```

binlog

```
### UPDATE `super_market`.`fruit`
### WHERE
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2= 'pear' /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
###   @3=273 /* INT meta=0 nullable=0 is_null=0 */
### SET
###   @1=1 /* INT meta=0 nullable=0 is_null=0 */
###   @2='pear' /* VARSTRING(20) meta=20 nullable=0 is_null=0 */
###   @3=573 /* INT meta=0 nullable=0 is_null=0 */
# at 569
```

Statement-based Replication (SBR)

- The binary log stores the SQL statements used to change databases
- Binlog event doesn’t contain pre-mutation state nor post-mutation state

```
UPDATE `super_market`.`fruit`
SET quantity= quantity + 300
WHERE name = 'pear'
```
query

```
UPDATE `super_market`.`fruit`
SET quantity= quantity + 300
WHERE name = 'pear'
```
binlog

RBR vs SBR

Summary

Requirements for CDC event	RBR	SBR
table	✓	✓
primary key	✓	✗
mutation type	✓	✓
before	✓	✗
after	✓	✗

Implementing CDC on SBR

How do we get the pre-mutation state

Our Solution

SQL Comments

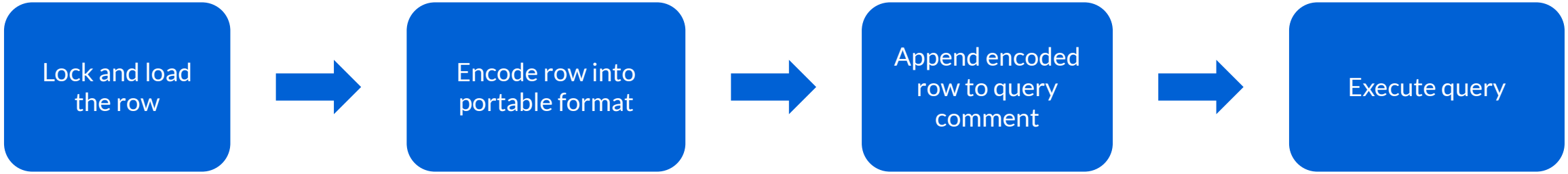
MySQL Query Comments

- MySQL supports placing comments within SQL statements
- MySQL ignores comments when parsing SQL statements
- Comments are preserved in the binlog with statement-based replication

Sample query with comments

```
select name, quantity from fruit where id in (123, 456) /*
trace_id=8826724f58f5586a.8826724f58f5586a<:8826724f58f5586a
&application=webapp
&user_id=12576
*/
```

Mechanics of Appending Pre-mutation State to Comments



Requirements for Serializer

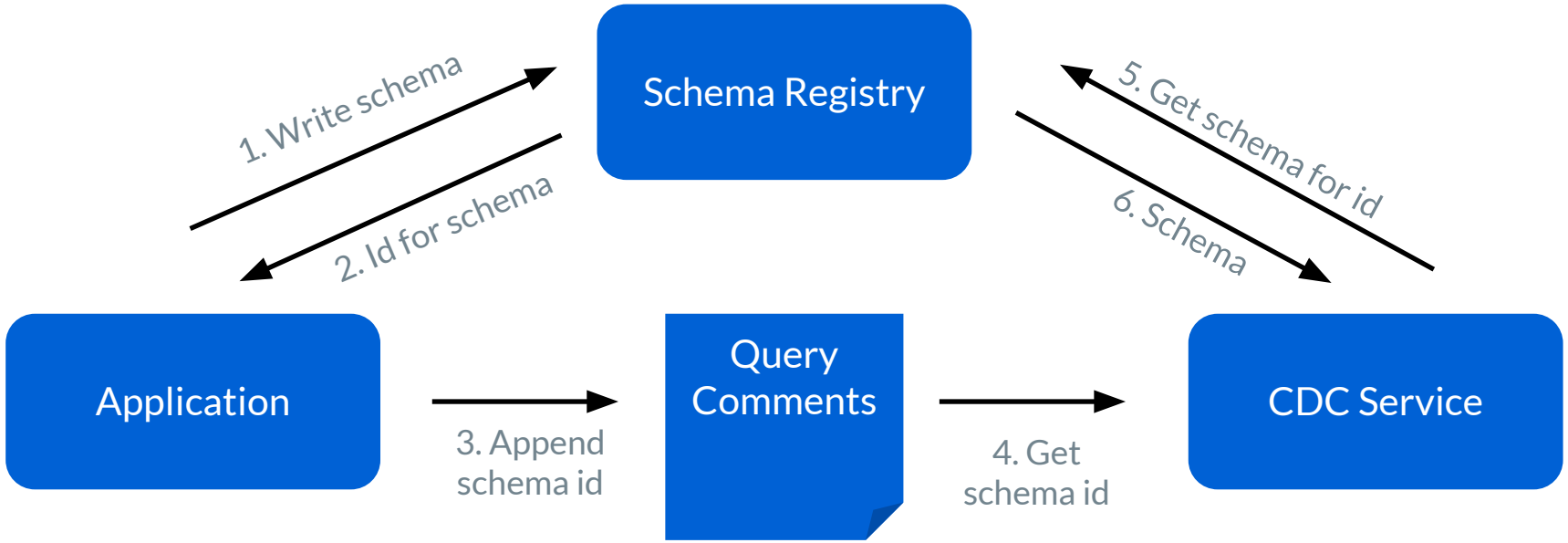
- Fully typed
  - Schema support
  - Schema evolution support
- Compact output
  - Encoded binary data should be very compact, so it takes up less storage space
  - Separate the schema from the encoded data
- Very fast

Avro for Serialization

Requirements for Serializer	
Fully Typed with Schema	✓
Schema Evolution	✓
Compact	✓
Separate Schema from Data	✓
Pretty Fast	✓

Schema Store

- Avro schema of encoded pre is stored in Confluence Schema Registry
- Every schema stored in Schema Registry has a numeric id associated with it
- Schema Id is appended to Query comments



Sample Query Comment

```
/*
trace_id=8826724f58f5586a.8826724f58f5586a<:8826724f58f5586a
&application=webapp
&pre_mutation_snapshot_schema_id=1091
&pre_mutation_snapshot_binary=FQQoYXQlZF9sxS5rXXISMTXSMQXzMDQSMDECssKQ1EoCFQESMDISMzX0MDXxX
hQmdS5QLXRlc3QCZQQoYXQlZF9sxS5rX2Zvcl9CX1ZvdmFfR2FsY2hlbmtvQ3QfZmlsZV8xMDXyMDMSMzXSMQQgc
GFzc3dvcmRfZm9yX0IgVm92YSBHYSxQxGVux28QcyBmxSxlIDESMDISMzXzMDXxXgXSc2hhcmVkX2xpbmsXXhYxMD
XyMDMSQDXSMgK0So3USgISMTXSMQXzMDQSMDICEmZ1bmMtdGVzdXQmc2hhcmVkX2xpbmtfZm9yX0QfVm92YV
9HYSxQxGVux28Qc19mxSxlXzESMDISMzXzMDXyXmBSYXQzd29yZF9mb3QfQiBSb3ZhIEdhbGQoZS5rbydzIGZpbGUg
MTXSMQXzMDMSMDICXX==
*/
```

Caveats with comments

Inclusion of pre-mutation state in comments results in query size explosion

- May need to increase max_allowed_packet
- Our existing max_allowed_packet value is big enough to support our needs
- p99 of query size increase is less than 100KB for us

Binlog files may get too big

- Might affect binlog file retention period on disk
- Not any worse than Row-Based Binlogs
- Additional 6-8MB/sec data added to our binlogs across all the shards

May need to impose restrictions on the mutation cardinality

- Protects from appending very large pre-mutation binaries to comments

Recap

Requirements for CDC event	SBR
table	✓
primary key	✗
mutation type	✓
before	✓✗
after	✗

SQL Comments

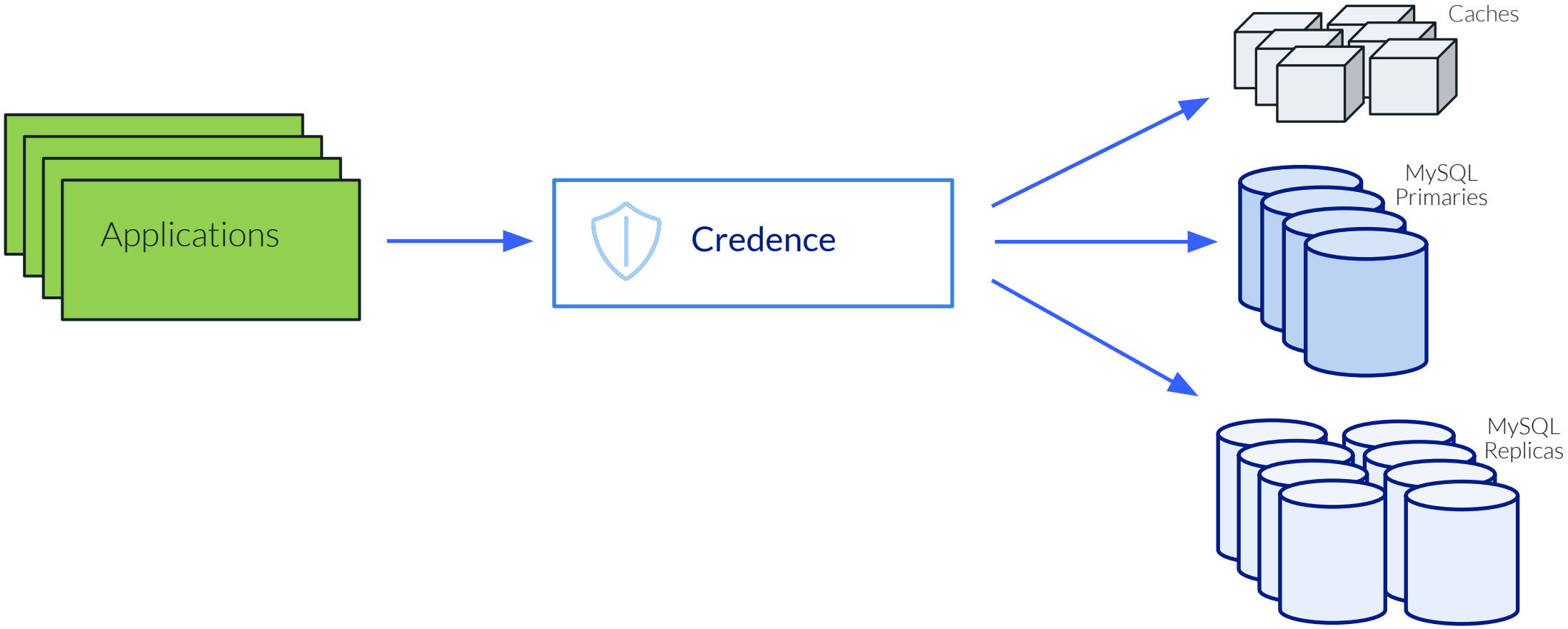
Computing post-mutation state & primary key

Credence

Distributed Data Access Service at Box

- Provides a uniform way to interact with relational data at Box
- Primarily responsible for protecting MySQL
- Provides strongly **opinionated** APIs supporting **limited** set of data access patterns

Credence Architecture



Mutation Queries Supported By Credence

- Multi-row inserts with explicit column values
- Conditional updates only by primary key
  - New column values are explicit
- Delete only by primary key

Insert	INSERT INTO `super_market`.`fruit` (id ,name ,quantity ,expiry_date) VALUES (1 ,'apple' ,100 ,'2021 - 01 - 01 ') , (2 ,'banana' ,950 ,'2021 - 01 - 02 ')
Update	UPDATE `super_market`.`fruit` SET quantity = CASE WHEN id = 1 THEN 88 WHEN id = 2 THEN 950 ELSE quantity END WHERE id IN (1 ,2)
Delete	DELETE FROM `super_market`.`fruit` WHERE id IN (1 ,2)

Queries Not Supported by Credence

Mutation queries with MySQL computed column values

- ex: update fruit set quantity = quantity + 100 where id = 2
- Clients who need to execute this kind of query would read the locked row to compute final values before executing mutation queries in a transaction

Delete/Update queries with unbounded where clause

- ex: delete fruit where expiry_date <1609509729
- Clients who need to execute this kind of query would load the primary keys first using the read apis and issue delete queries using primary keys in a transaction

How to Compute Post-mutation State & PK

- Parse mutation queries to extract columns modified and their new values (diff) as well primary keys
- Post can be computed by merging pre-mutation state with diff within CDC pipeline
  - post = pre + diff
- No additional changes to query size

Compute Post-mutation State & PK for Insert

```
INSERT INTO `super_market`.`fruit` ( id, name , quantity, expiry_date )
VALUES
        ( 1 ,'apple' ,100 ,'2021 - 01 - 01 ' )
/*
trace_id=8826724f58f5586a.8826724f58f5586a<:8826724f58f55
86a
&application=webapp
*/
```

Compute Post-mutation State & PK for Insert

Raw diff

```
( id, name , quantity, expiry_date )
( 1 ,'apple' ,100 ,'2021 - 01 - 01 ' )
```

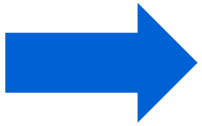
diff

Columns changed	New values
id	1
name	apple
quantity	100
expiry_date	2021-01-01

Compute Post-mutation State & PK For Insert

diff

Columns changed	New values
id	1
name	apple
quantity	100
expiry_date	2021-01-01



post

Columns changed	New values
id	1
name	apple
quantity	100
expiry_date	2021-01-01

Compute Post-mutation State & PK for Update

```
UPDATE `super_market`.`fruit`
SET quantity = CASE WHEN id = 1 THEN 88 ELSE quantity END
WHERE id IN ( 1 )
/*
trace_id=8826724f58f5586a.8826724f58f5586a<:8826724f58f558
6a
&application=webapp
&pre_mutation_snapshot_schema_id=1091
&pre_mutation_snapshot_binary=FQQoYXQlZF9sxS5rXXISMTXSM
QXzMDQSMDECssKQ1EoCFQESMDISMz==
*/
```

Compute Post-mutation State & PK For Update

Raw diff

```
quantity = CASE WHEN id = 1 THEN 88
ELSE quantity END
```

diff

Columns changed	New values
quantity	88

Raw pre

```
pre_mutation_snapshot_schema_id=1091&
pre_mutation_snapshot_binary=FQQoYXQl
ZF9sxS5rXXISMTXSMQXzMDQSMDECss
KQ1EoCFQESMDISMzX0MDXxXhQmdS5
QLXRlc3QCZQQoYXQlZF9sxS5rX2Zvcl9
CX1ZvdmF==
```

pre

Columns changed	New values
id	1
name	apple
quantity	100
expiry_date	2021-01-01

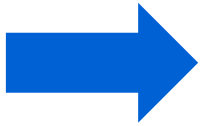
Compute Post-mutation State & PK for Update

pre

Columns changed	New values
id	1
name	apple
quantity	100
expiry_date	2021-01-01

diff

Columns changed	New values
quantity	88



post

Columns changed	New values
id	1
name	apple
quantity	88
expiry_date	2021-01-01

Downside of Parsing Queries

- Query parsing can get complicated and often error prone
- Binlog tailer implementation is tightly coupled with data access service that creates and executes queries

Techniques to Avoid Query Parsing

/ Append diff to query comments

/ Append post-mutation state to query comments

Both options results in further explosion of query and binlog file size

Implementing CDC on SBR - Recap

Requirements for CDC event	SBR
table	✓
primary key	✗
mutation type	✓
before	✗
after	✗

Query Parsing

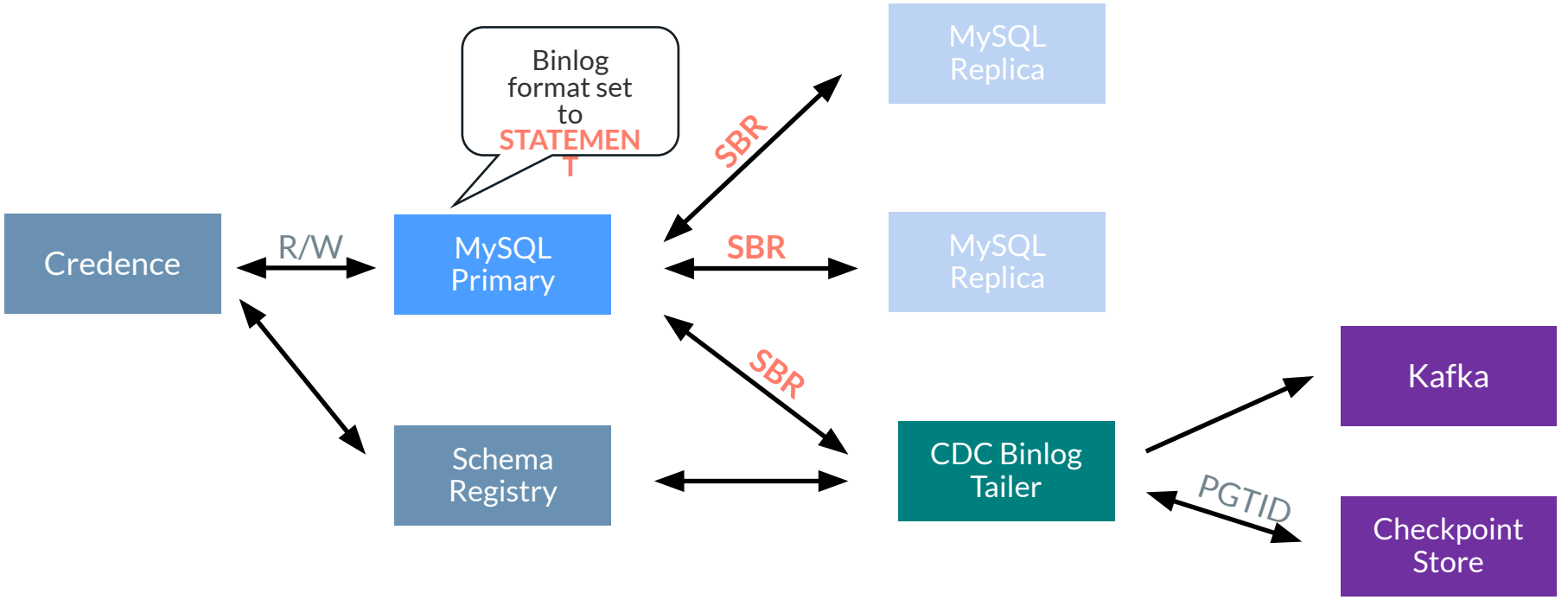
SQL Comments

Query Parsing

CDC at Box

CDC at Box

Architecture



Scale of CDC at Box

100s	10,000s	10s
MySQL Shards	CDC events per sec	CDC Consumers

Venkat Morampudi
vmorampudi@box.com