HBase: Recent Improvement And Practice At Alibaba

Allan Yang (Alibaba/HBase Committer)

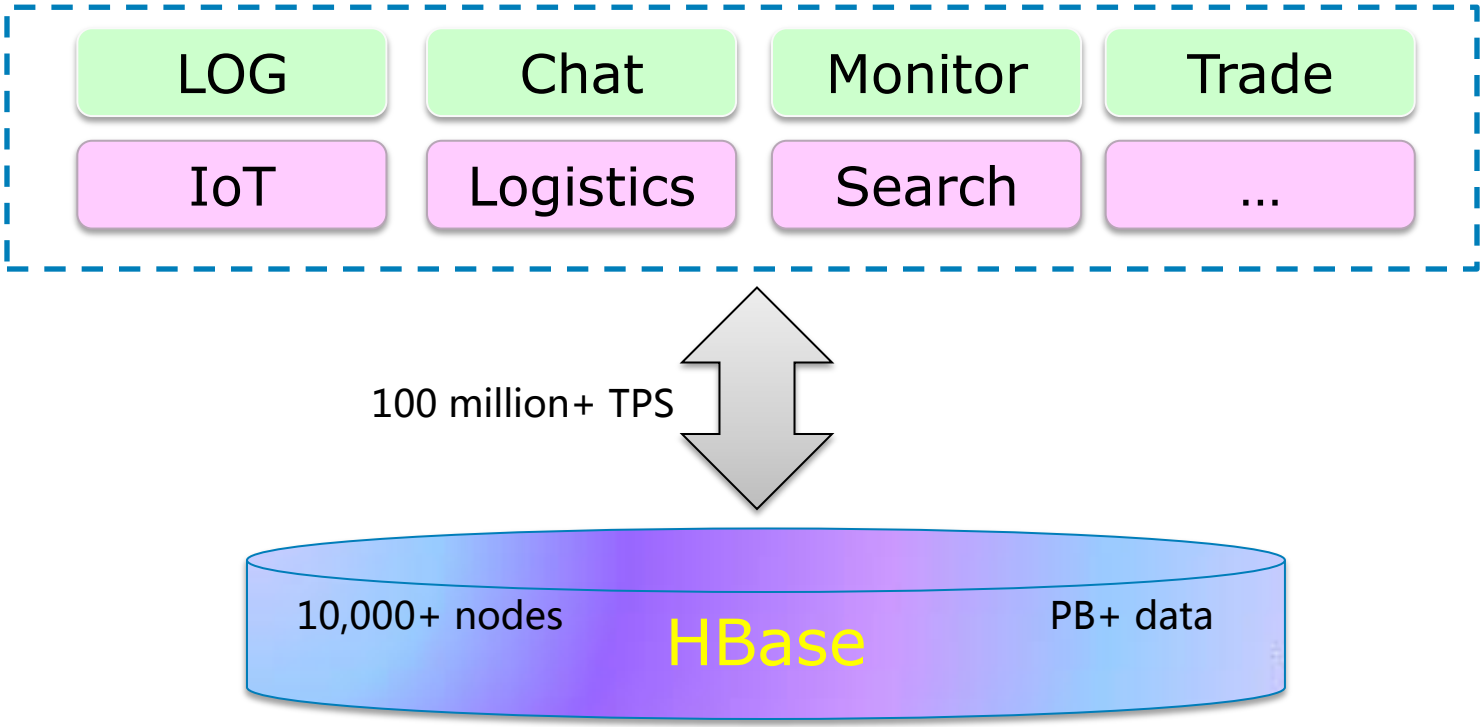
Han Yang (Alibaba)

Alibaba Group

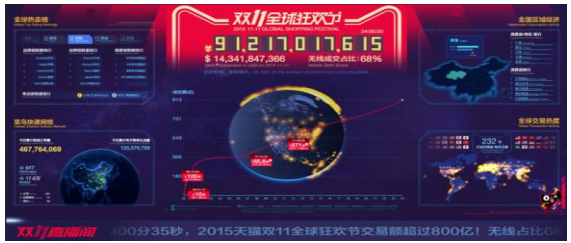
Agenda

- HBase at Alibaba
  - Typical scenarios
- Running Architecture
  - Range data copy
  - Dual Service
- SQL
  - Performance and feature improvements

HBase At Alibaba



Double 11 Festival



Data Service

Low latency query

Data Src

- Trade
- Order
- LOG
- ...

Message Middleware

Real-time Computing

HBase

1GB/s + throughput

1 Million + TPS

HBase

HBase

GMV SCREEN

Risk Management of Ant Financial

Risk Console

Real Time Query

1 Million+ QPS

1 Million+ TPS

Real-Time import

HBase

Incremental export

Daily result import

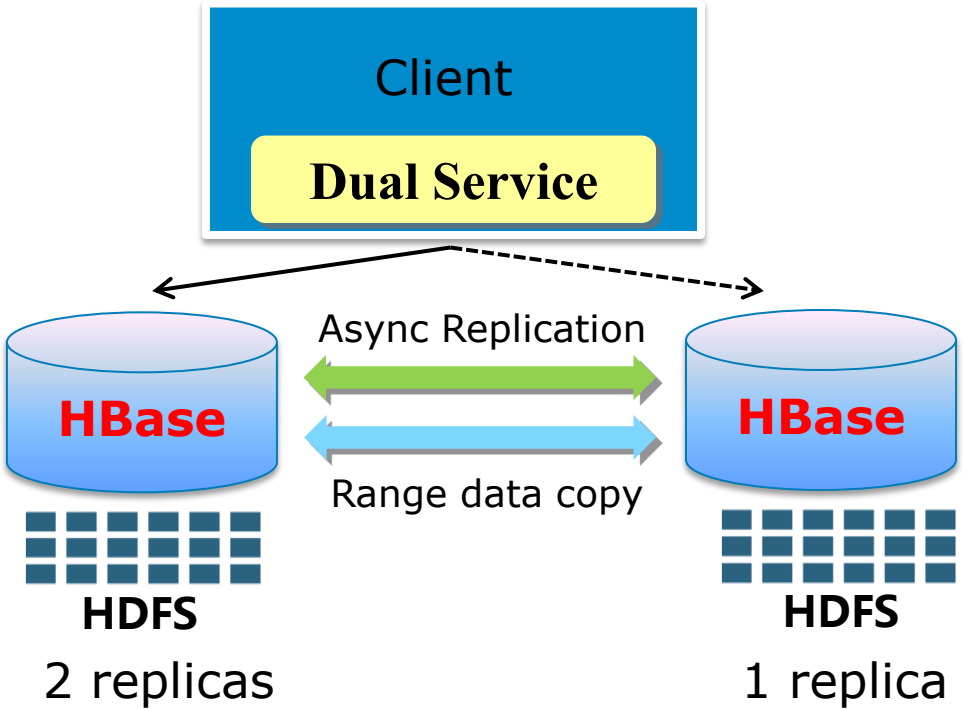
Offline Computing

10TB+ data per day

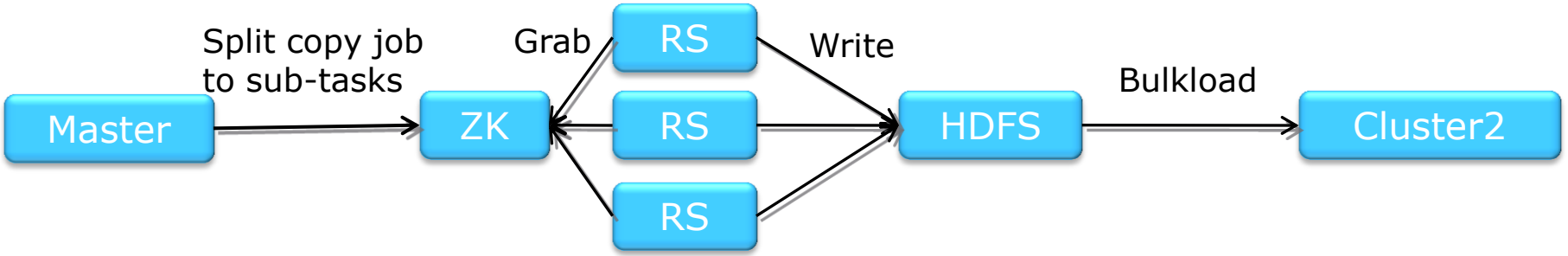
Expire data based on:

- TTL
- Version
- Low value column

Deployment Architecture



Range Data Copy



- A feature provided inside HBase, fully distributed, no MR
- On the fly, no need to stop service
- Recoverable from all kinds of error and disaster

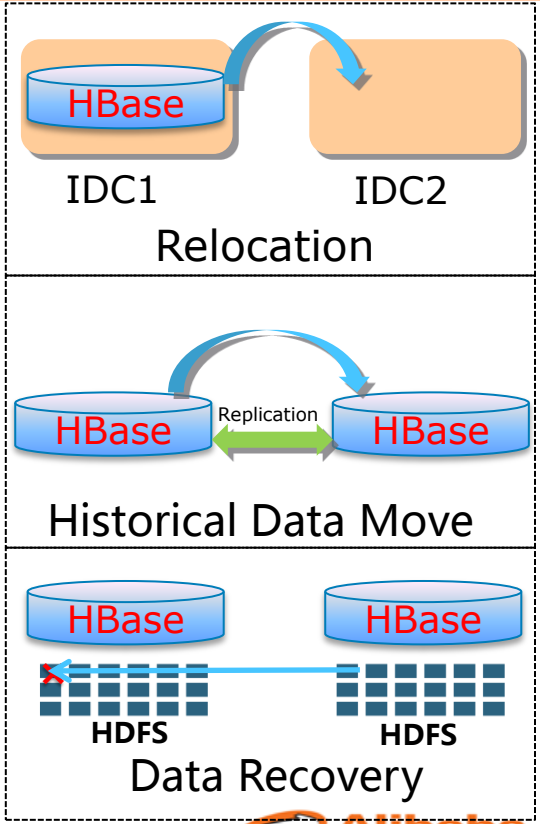
Range Data Copy

Scenarios

- Data Center relocation
- Historical data Movement
- Data Recovery

Other Solution

1. CopyTable
   - Too too slow (scan table)
   - Need MR role
2. Snapshot （Backup in HBase2.0）
   - Need disable table when restore
   - No control of data range
   - Not design for data migration



Dual Service – Why?

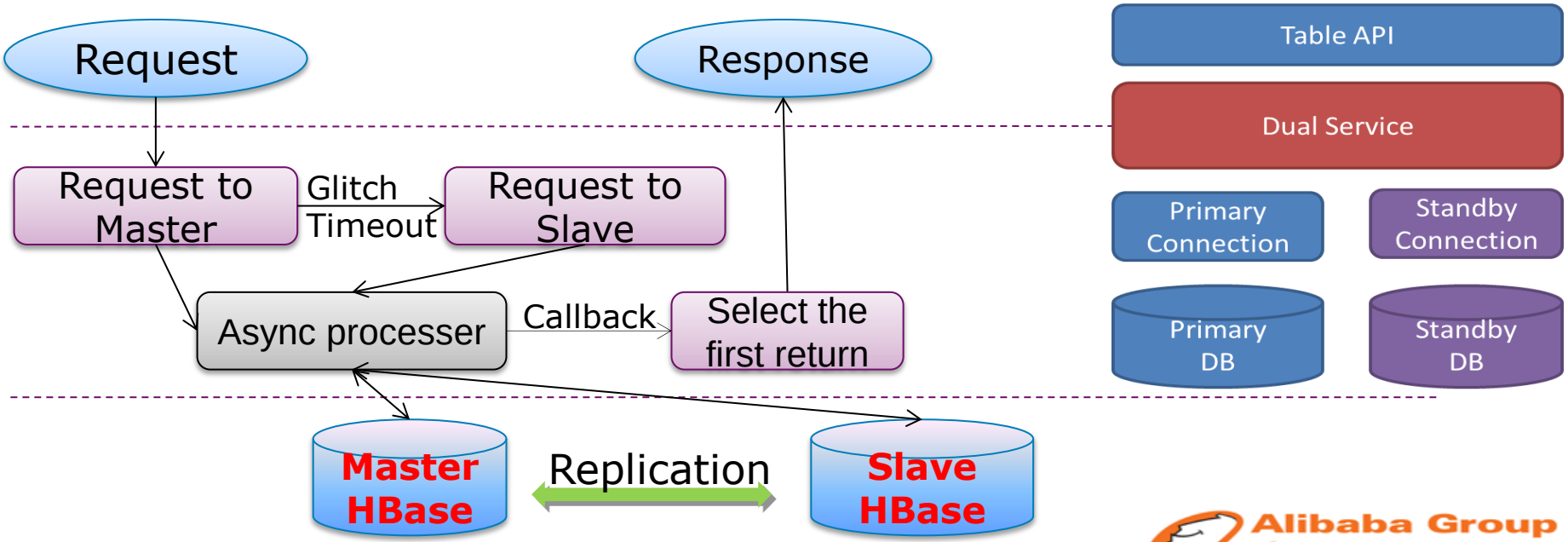
- Region split, balance, RS down …
- GC
- Network
- HDFS

Possible Solution: Region Replicas (HBASE-10070)

- **Need internal replication**
- **Need triple disk space**
- **Replica region is not writable**

Dual Service

- Take advantage of slave cluster
- No extra resources needed



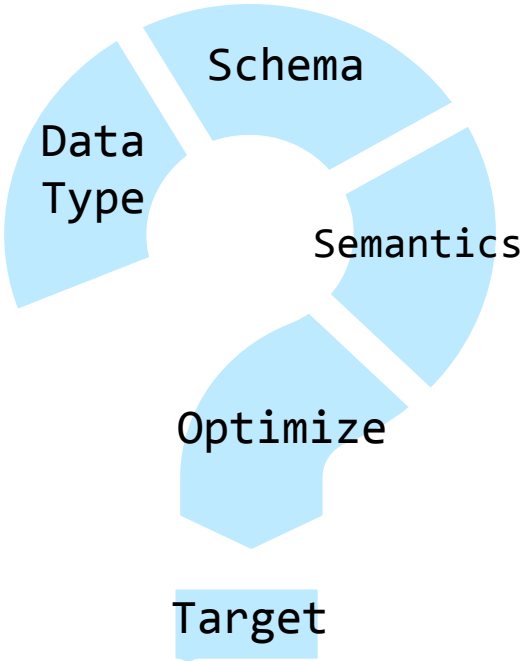
Dual Service - Benchmark

- Let’ s call Request with RT > 50ms a ‘spike’
- Set Glitch Timeout = 40ms(call slave if running after 40ms)
- Spike rate
  - Before Dual Service: Requests with RT>50ms / Total request
  - After Dual Service: (The proportion of request > 40ms in master) * (The proportion of request > 10ms in slave)

	W/O Dual Service	W/ Dual Service
Spike rate	0.047095%	0.001714%

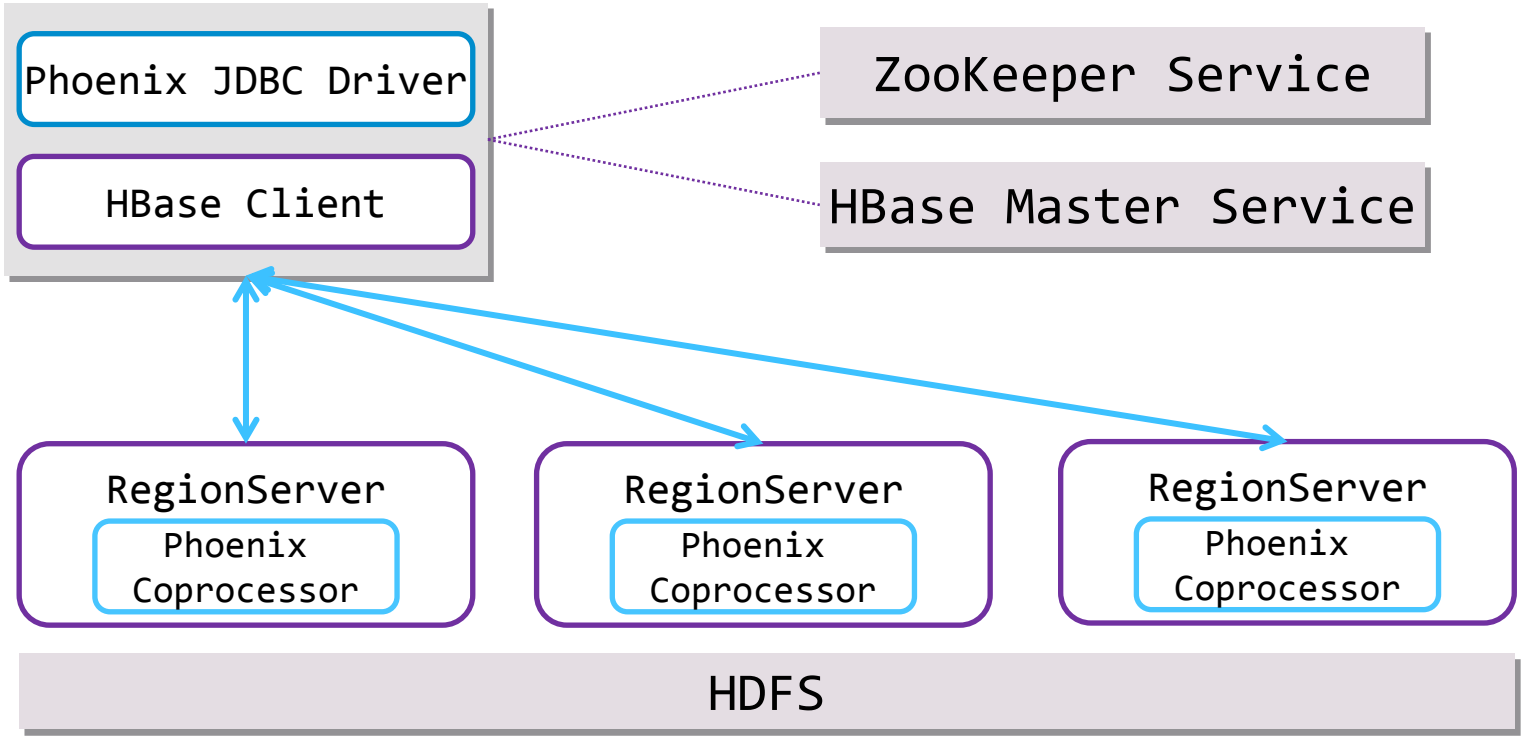
Why SQL?

Easy and Quick to use HBase

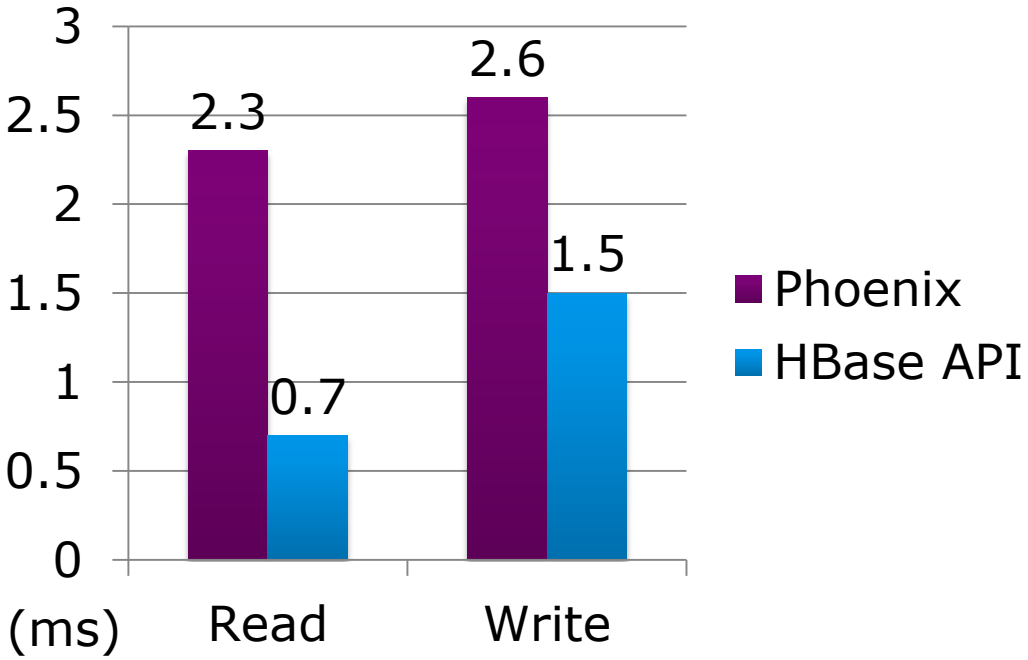


1. Rich data typing
2. Rowkey construction
3. Basic/complex queries
   Query with index
4. Optimize transparently
   Existing tools

Phoenix

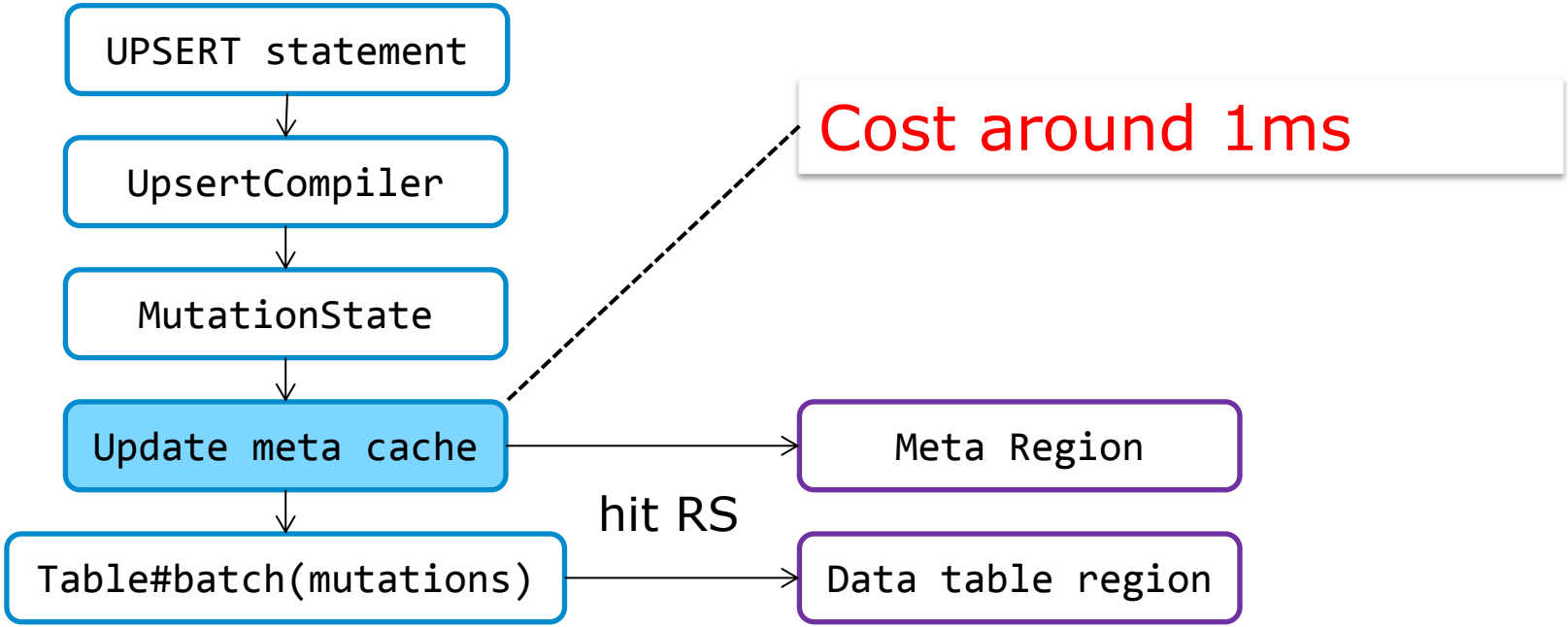

Phoenix JDBC Driver
HBase Client
ZooKeeper Service
HBase Master Service
RegionServer
Phoenix Coprocessor
RegionServer
Phoenix Coprocessor
RegionServer
Phoenix Coprocessor
HDFS

Performance against HBase API



- Single row select/Scan
- Single row upsert/Put

Why is UPSERT much slower?



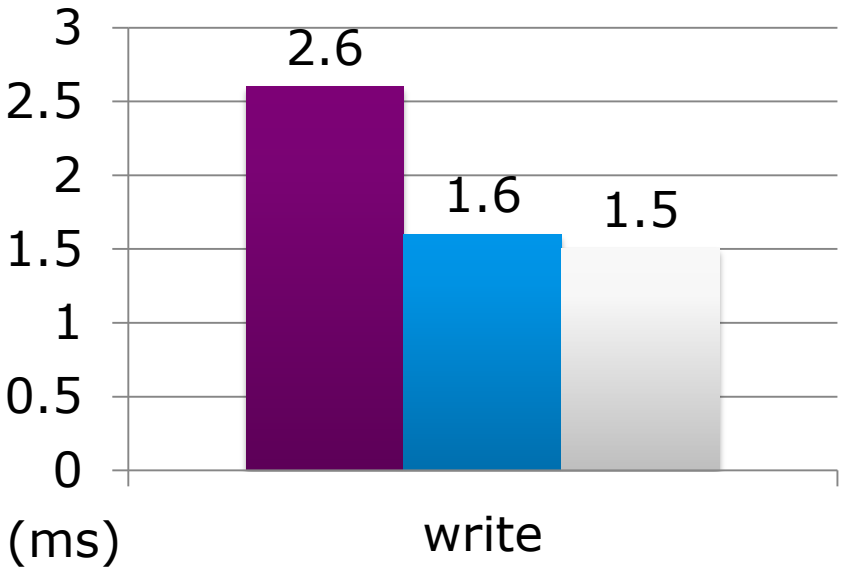
What is Meta Cache?

- Meta data of a Phoenix table in each Client
- Schema
  - Columns, types, properties
- Indexes
  - Add new index
  - Drop an index

Meta Cache Update Policy

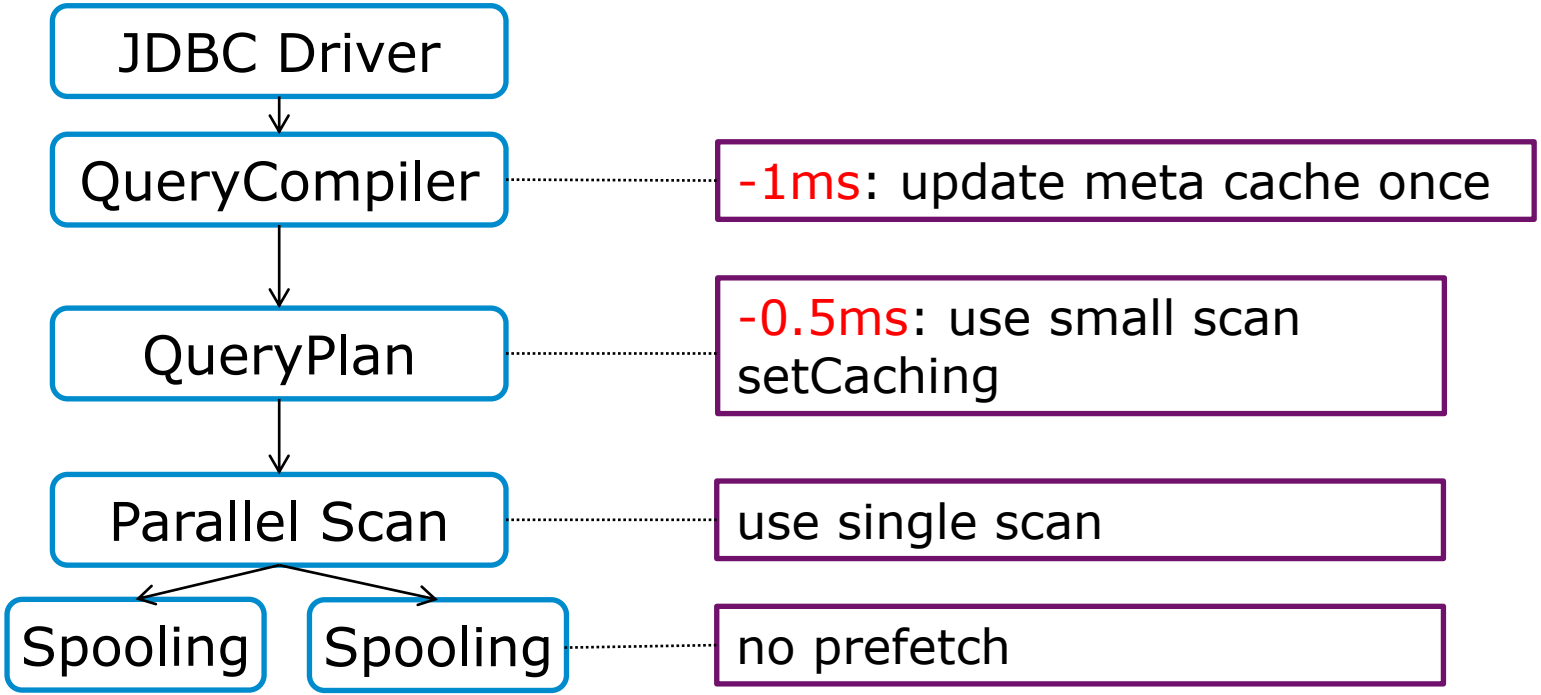
- Init at the 1st time
- Update meta periodically(PHOENIX-2520)
- Update meta at mistakes
- Version for each meta update
  - Request with meta version
  - Server always has the latest version

Lift UPSERT performance



Reduce 38% latency

Lift SELECT performance



Parallel or Single?

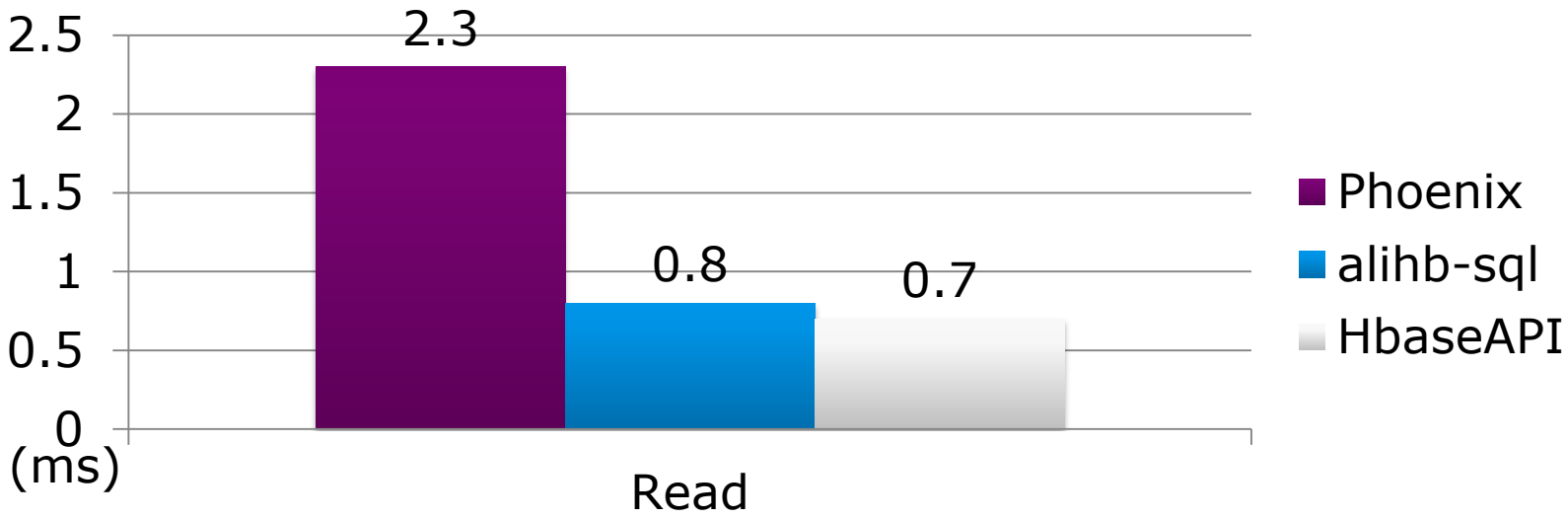
- Use small scan properly
- Use parallel scan unless it's necessary

Scenarios	Phoeinx's plan	alihb-sql's plan
full table scan	parallel big scan	single big scan
single row select	parallel big scan	single small scan
single region range scan	single big scan	single small scan
cross region range scan	parallel big scan	single big scan
aggregation	parallel big scan	parallel big scan

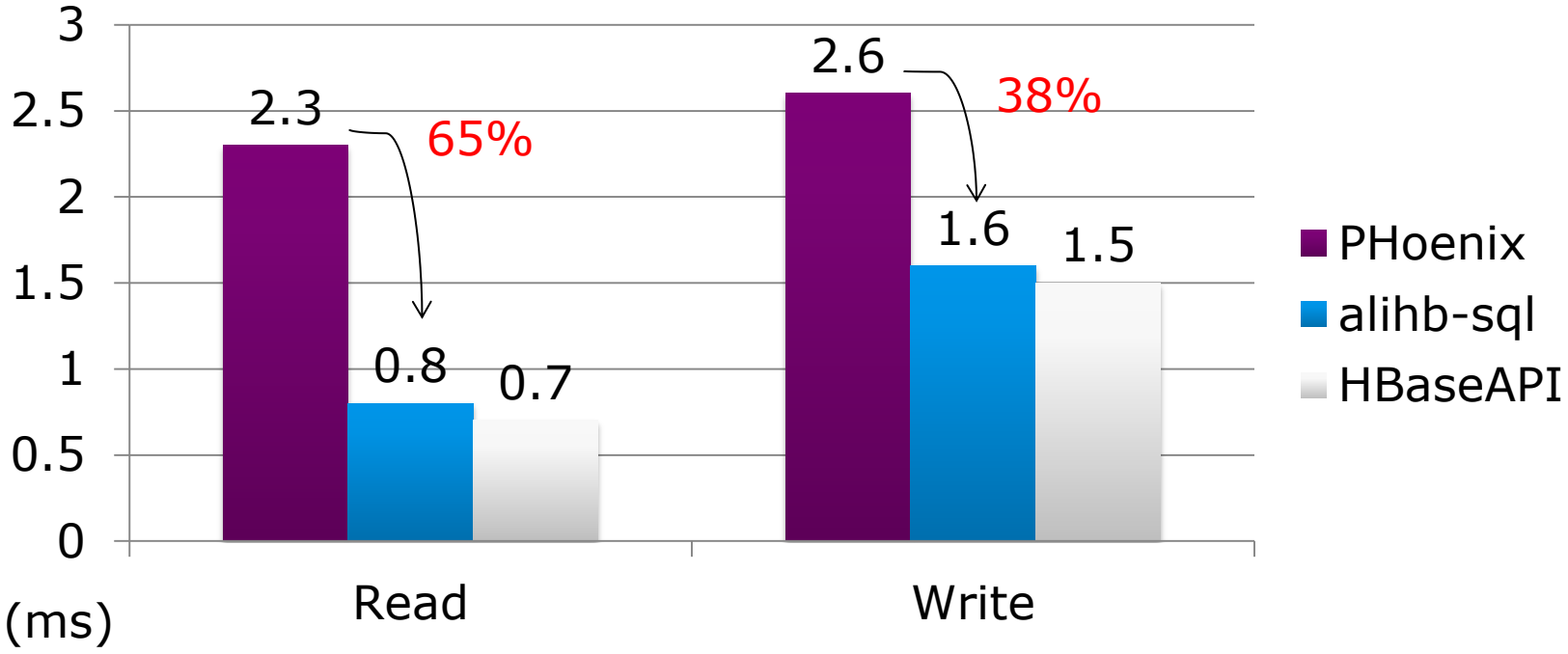
Lift SELECT performance

select * from tt where a = 10;

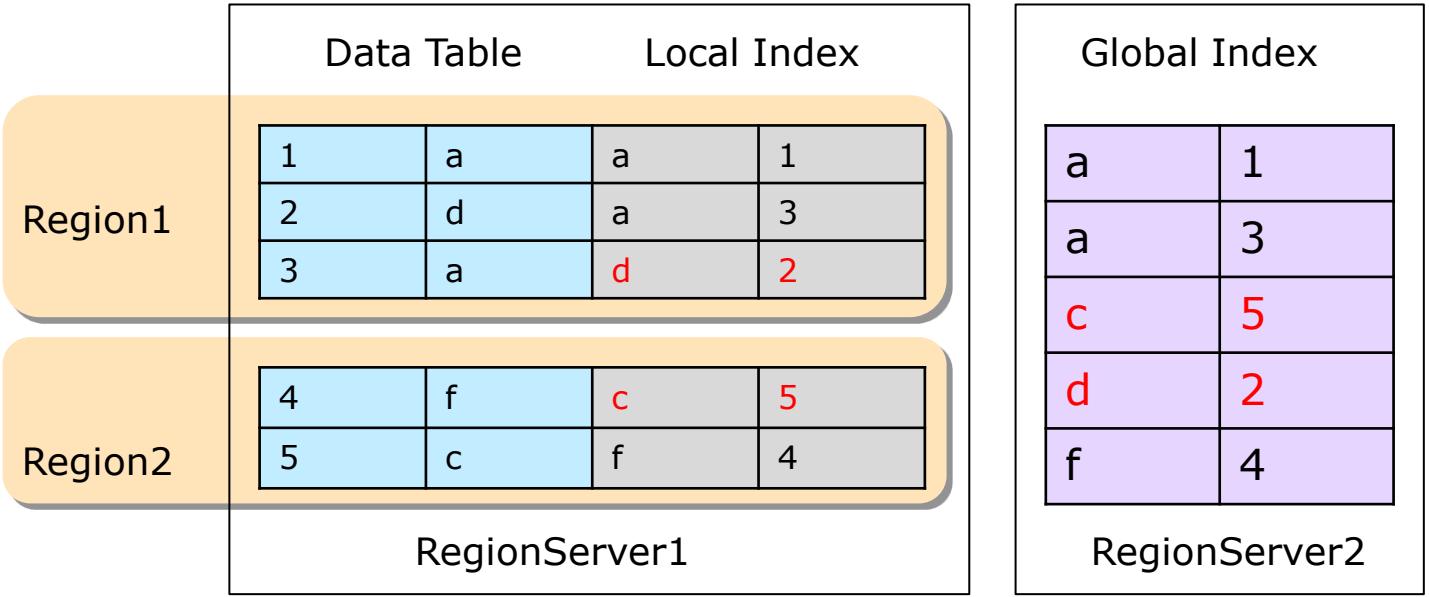
Reduce 65% latency



Improved performance

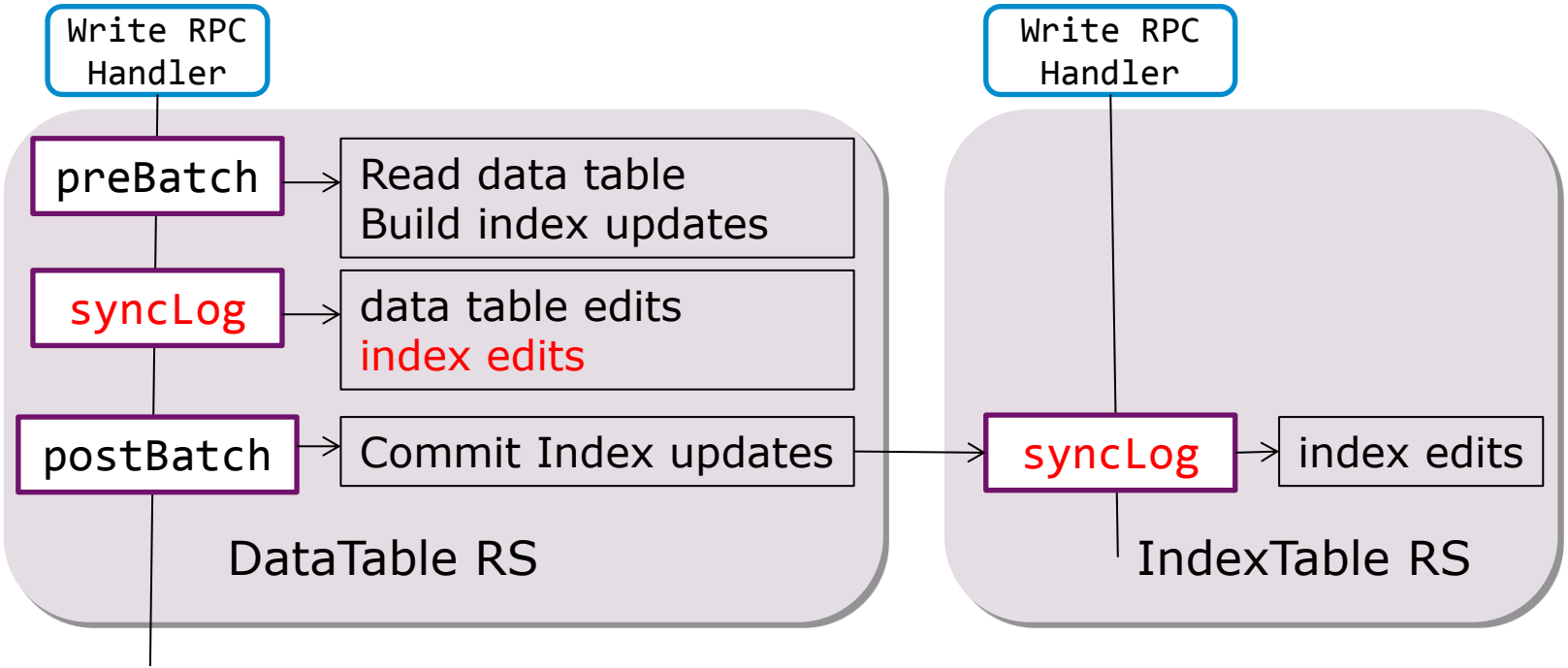


Secondary Indexing

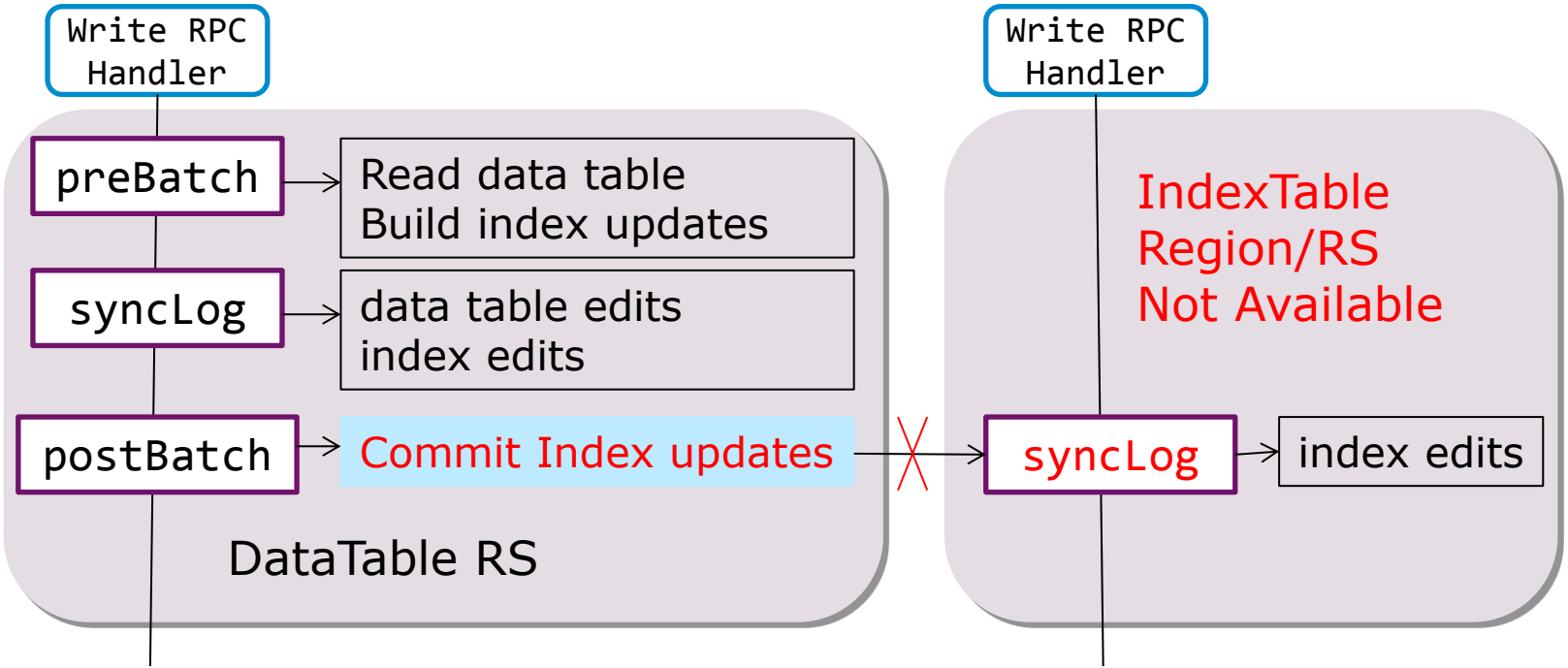


select * from tt where (pk between 1 and 3) and col = 3;
select * from tt where col = 3;

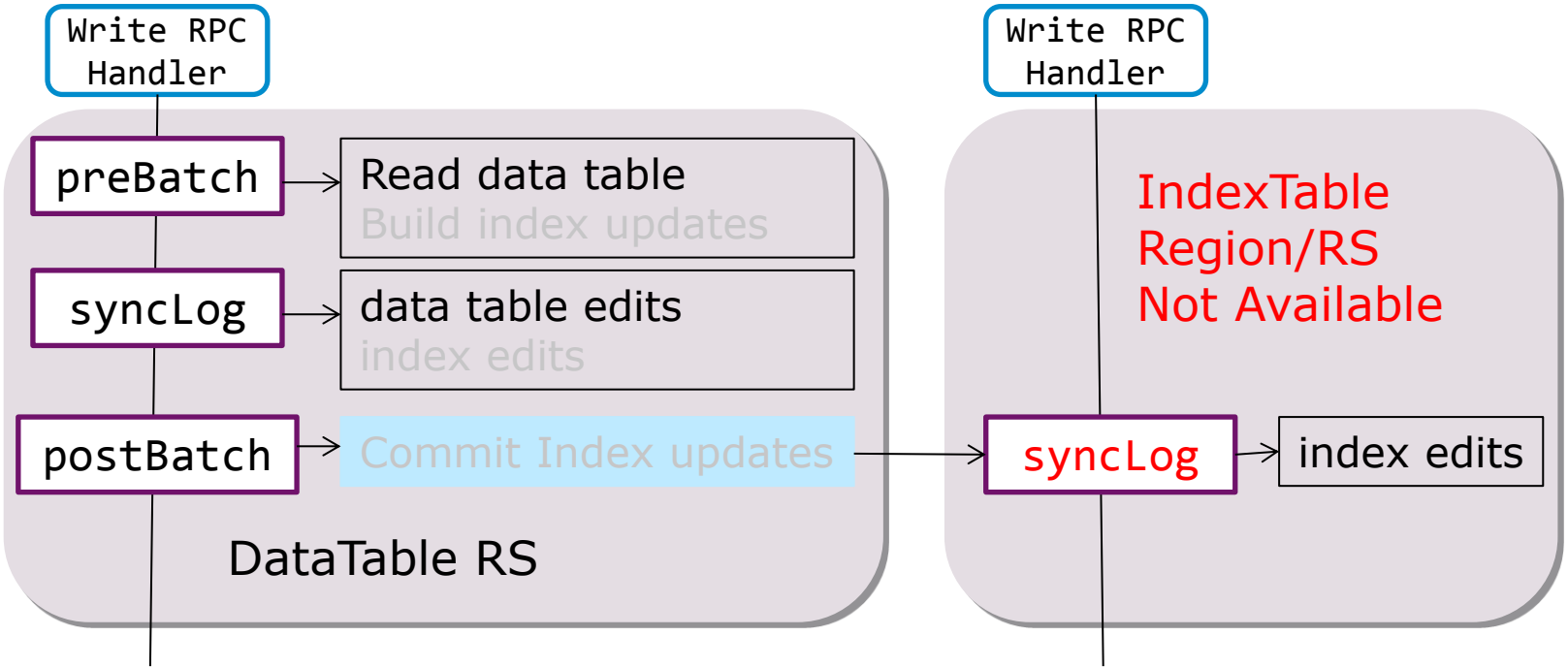
How Global Index Works?



Consistency: Index Updates Failure



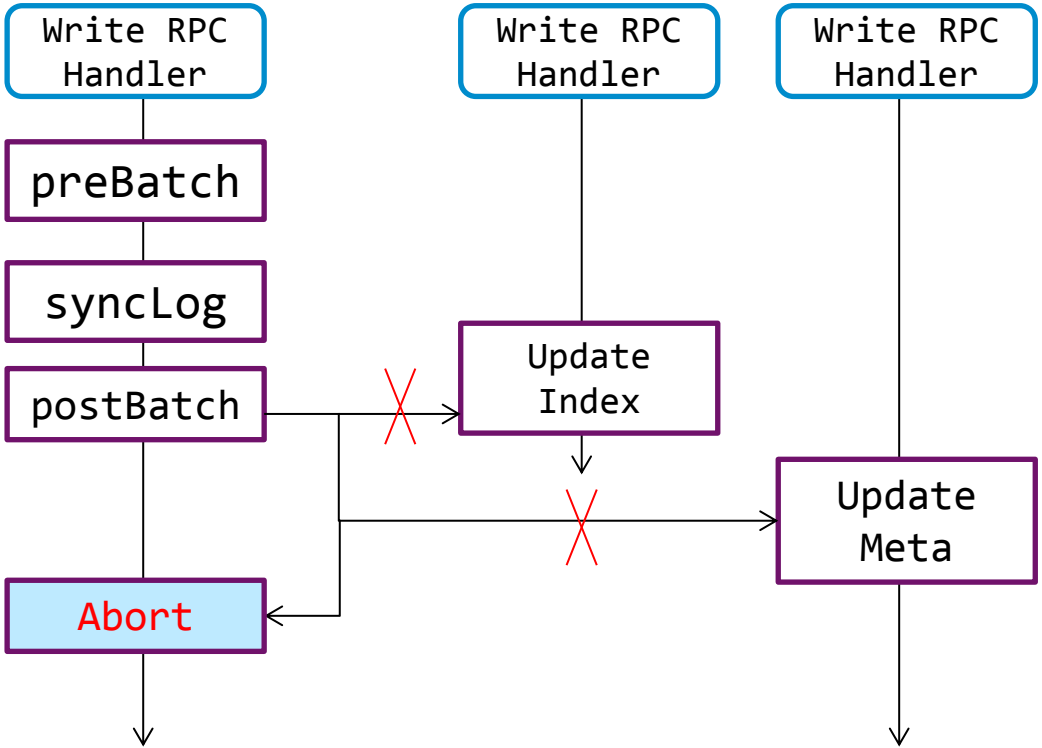
Solution I: Disalbe Index writing



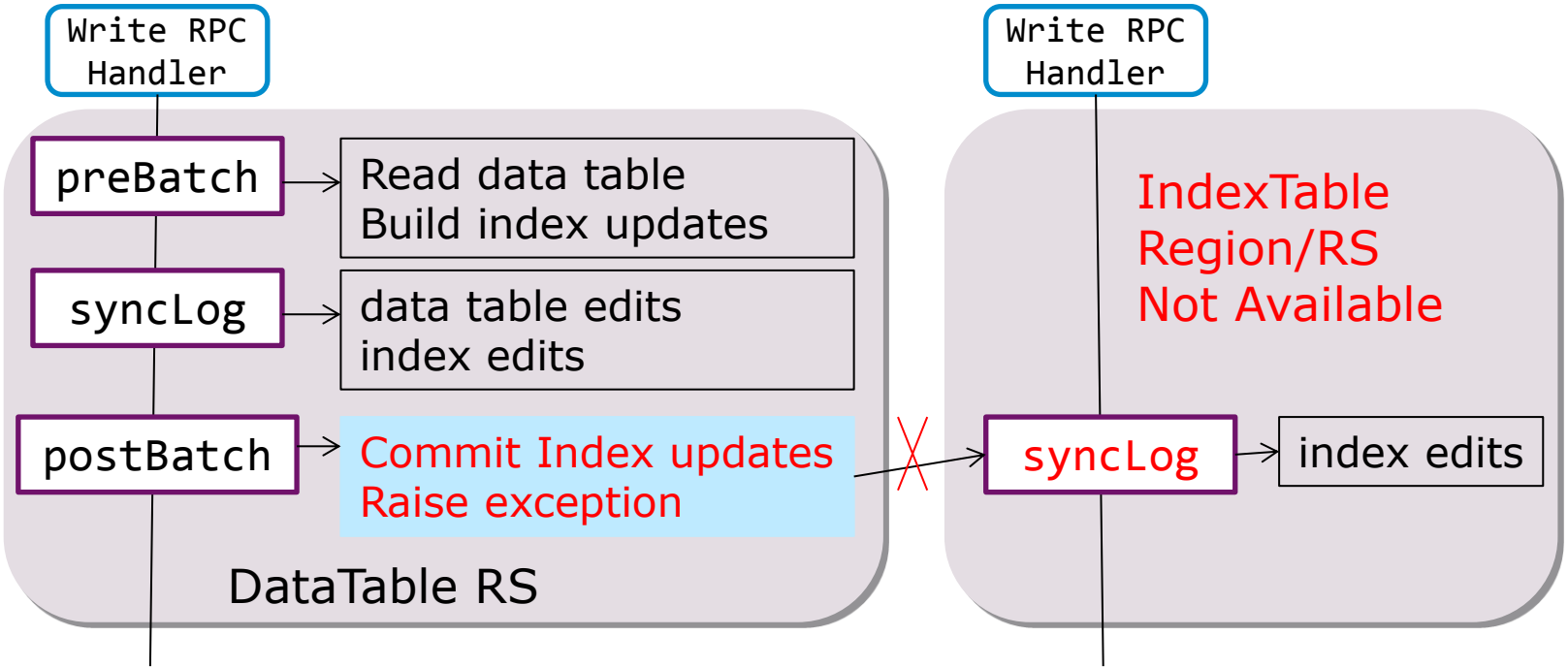
Solution I: Disable Index writing

- Update meta (may cause chain collapse)
  - Set index state to DISABLE
  - Set index disable timestamps
- Query degenerated to full table scan over data table

Solution I: What If Update Meta Failed?



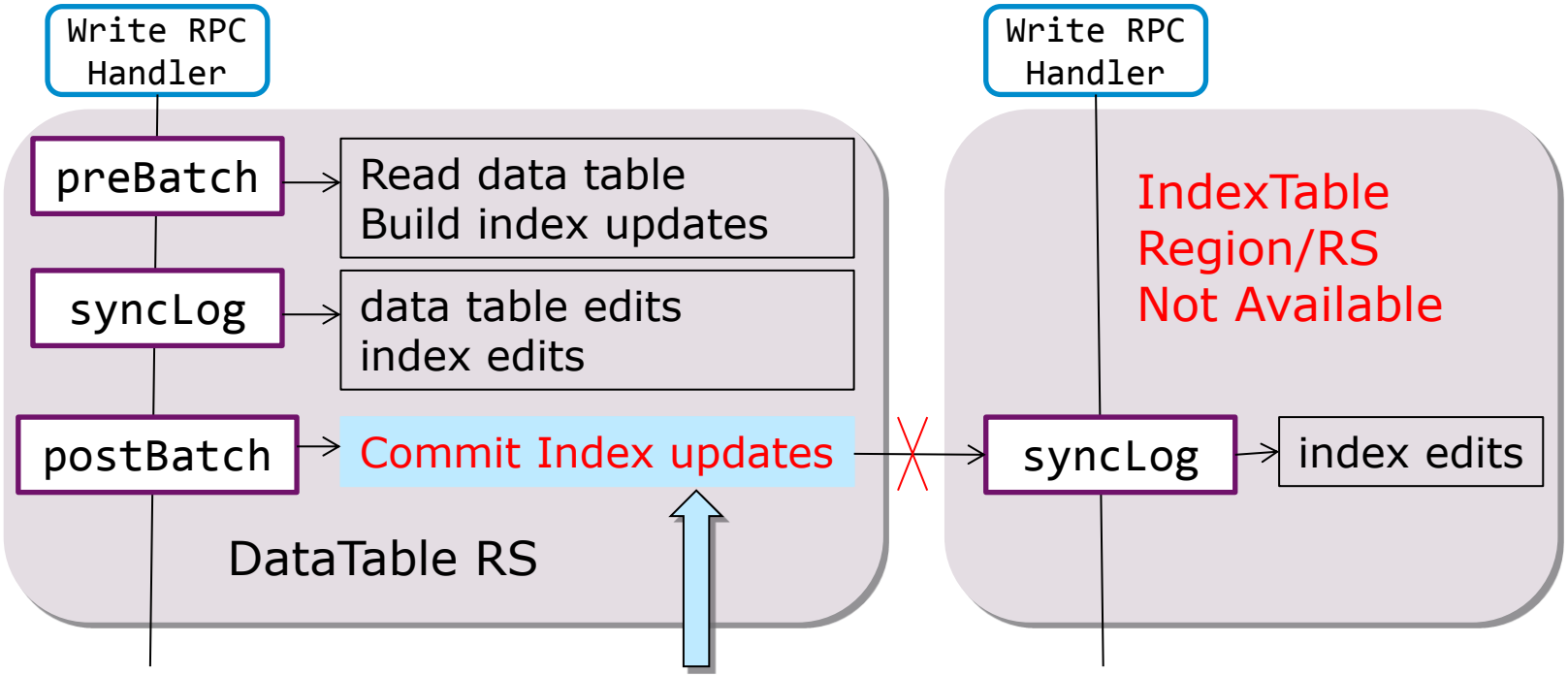
Solution II: Disalbe Data Table Writing



Solution II: Disable Data Table Writing

- Still need to update meta data
- Index still active
- Partially rebuild index background
  - Range scan over full data table

Solution III: Infinite Retry



Infinite retry until succeed, do so at failover replay

Solution III: Infinite Retry

- Block data table writing
- Index active
- No need to rebuild index
  - Retry all the updates on wire

Other improvements

- Index merge query
- Increment
- CheckAndMutate
- MultiVersion and timestamps
- Data import/export
- Slow SQL metrics
- ...

Thanks!

FAQ