Introduction to Exchange CDB(eCDB)

edgeyang(杨杰)

Agenda

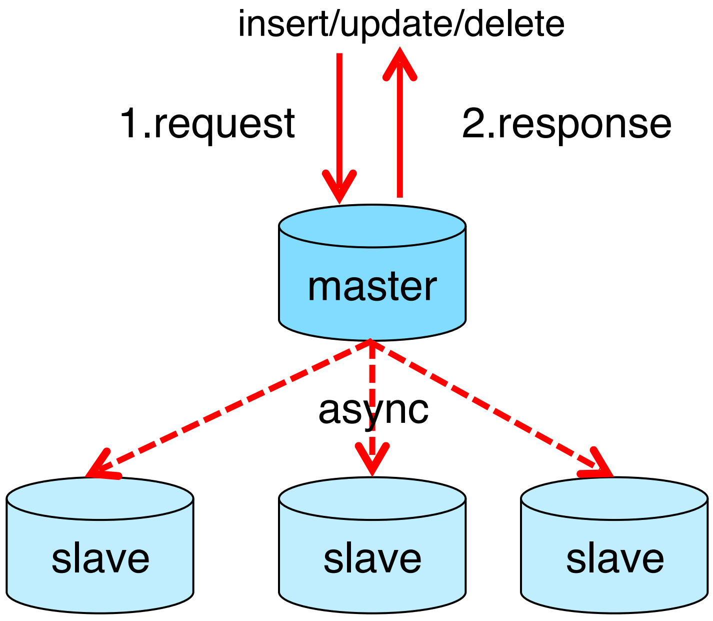
1 Background

2 What is eCDB?

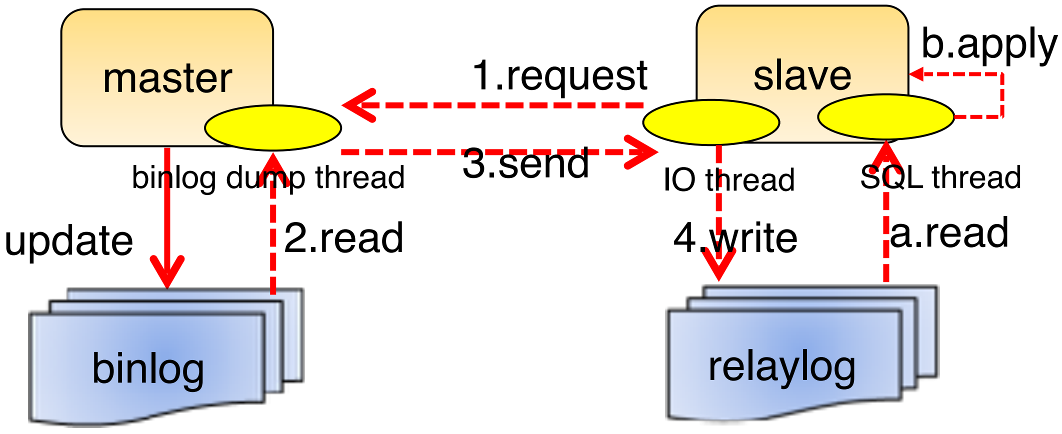
3 Key Technologies

4 Architecture

Background



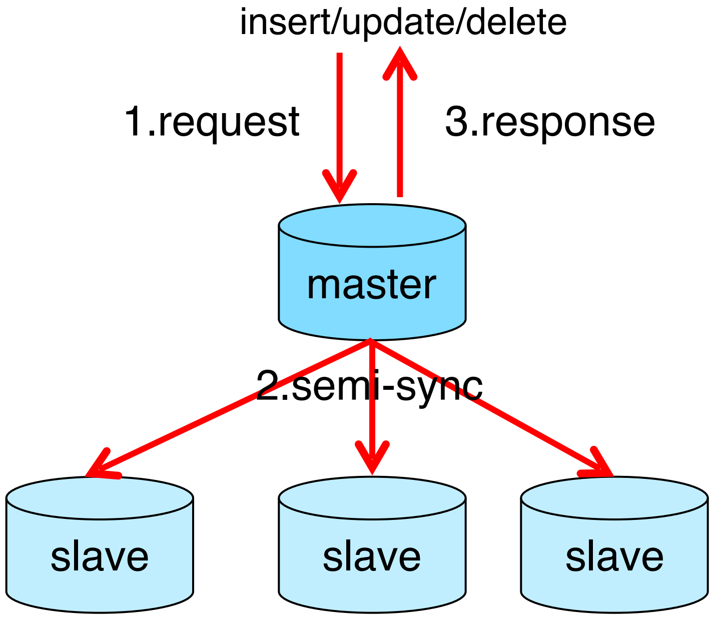
One master vs N slaves



Async replication details

Data loss!

Background



semi-sync

1. Weak data consistency
2. Only one slave accepts log
3. Redundant transactions
4. Low performance

Background

p Transactional DB storage features

ü schema is more complex , requires transactional operation

ü Strong consistency

ü Disaster Tolerance across region

p Pains

ü Data inconsistent

ü Longer time in service recovery (RTO)

Fault type	Operation	consistency	RTO
Recoverable	1.Reboot 2.Mysqld restart	√	10min
Unrecoverable	1.Reboot 2.Switch to slave 3. Comparison	×	>10min

What is eCDB?

eCDB = strong consistency

+ across region

+ HA

+ high performance

+ security audit

+ data sharding

+ MySQL

Customized slaves

p Problems:

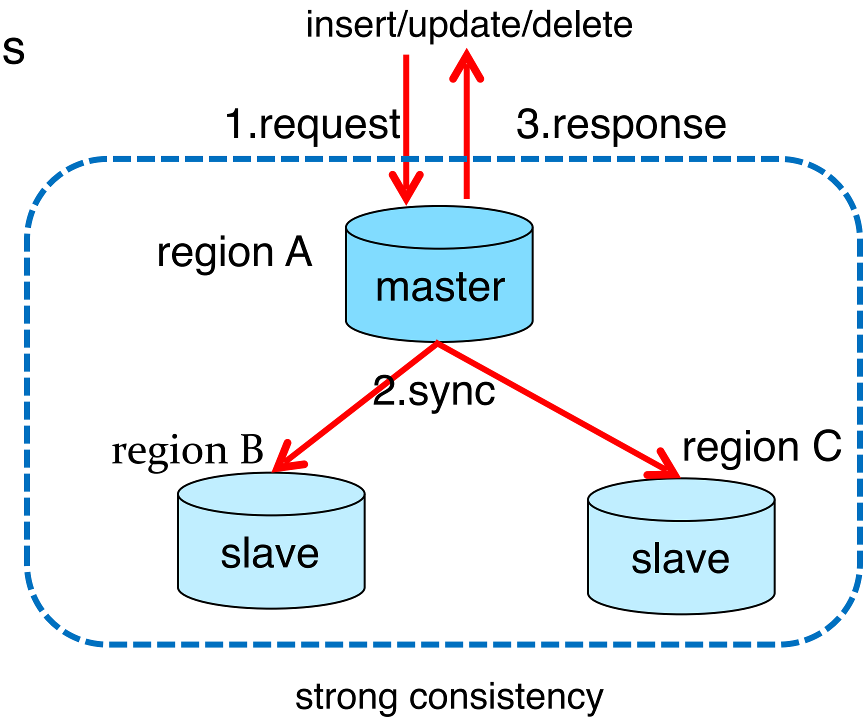
ü Semi-sync downgrade

ü Only one slave accepts log

p Solutions：

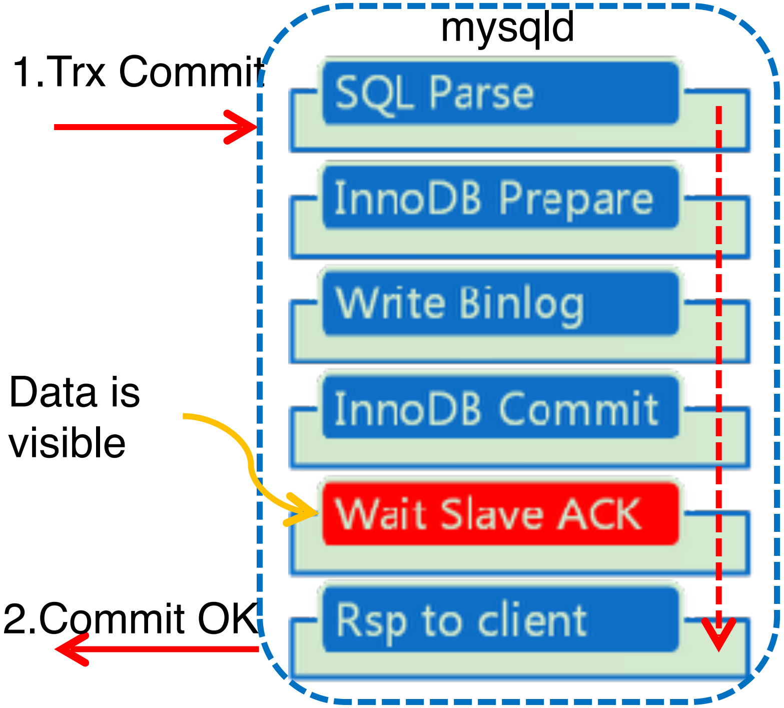
ü No downgrade

ü Customized slaves



Phantom read

- Problem:
  - phantom read



1.Trx Commit

mysqld

SQL Parse

InnoDB Prepare

Write Binlog

InnoDB Commit

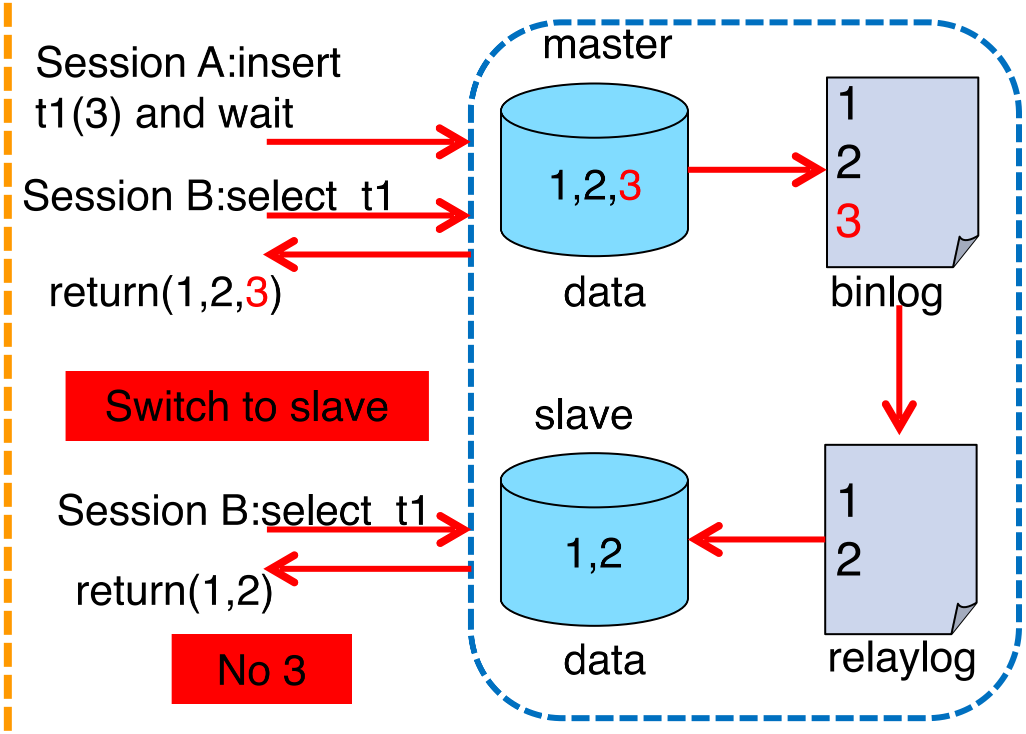
Wait Slave ACK

Rsp to client

Data is visible

2.Commit OK

After commit



Session A:insert t1(3) and wait

Session B:select t1

return(1,2,3)

Switch to slave

Session B:select t1

return(1,2)

No 3

master

1,2,3

data

1
2
3

binlog

slave

1,2

data

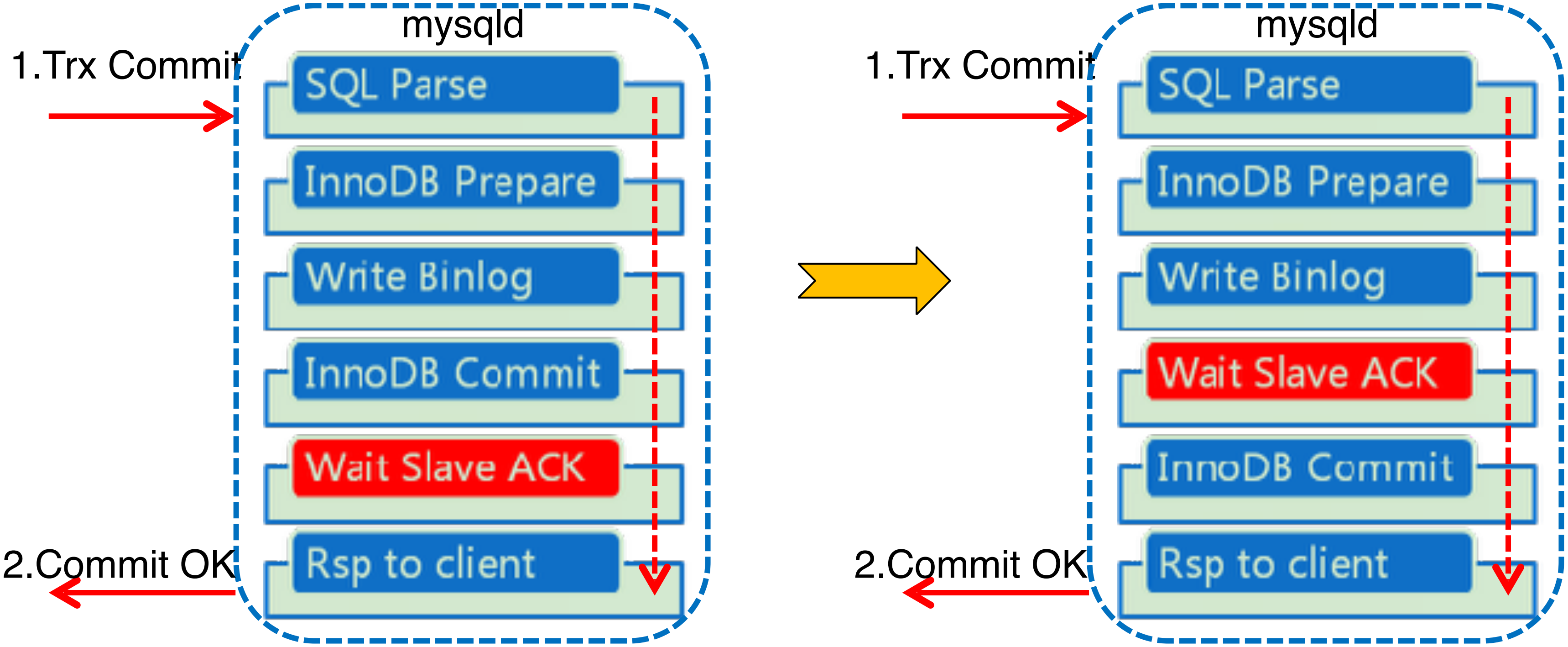
1
2

relaylog

Phantom read

Phantom read

- Solution : Send to slave before commit



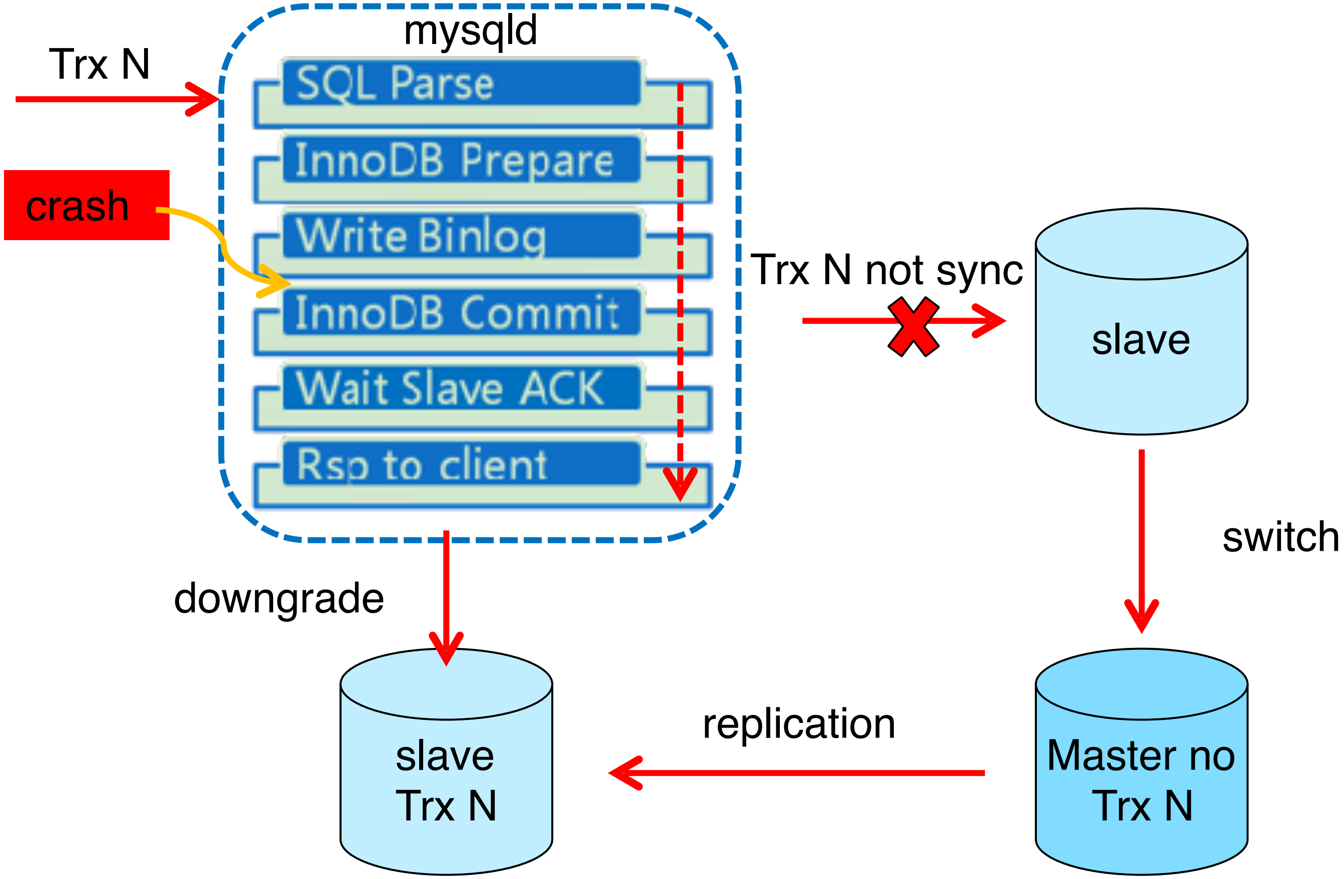
After commit

After sync

Flash back

p Problem:

ü master downgrade to slave , but has redundant transaction(s)



Performance

p **Key points**

ü time-consuming of Single transaction

ü Throughput(QPS)

p **Per transaction**

ü $T_{total}=T_{sql}+T_{engine}+T_{replica}$

ü $T_{replica}=T_{network+}T_{fsync}$

ü benchmark：

- ü time-consuming of Full cache single transaction : 3.82ms
- ü time-consuming of Semi-sync single transaction : 8.33ms including RTT 2.6ms
- ü SAS disk order 512B write + fsync delay 0.13ms
- ü T_{fsync} = 8.33-3.82-2.6=1.9ms

p **Throughput**

ü Throughput = concurrent number / time-consuming of single transaction

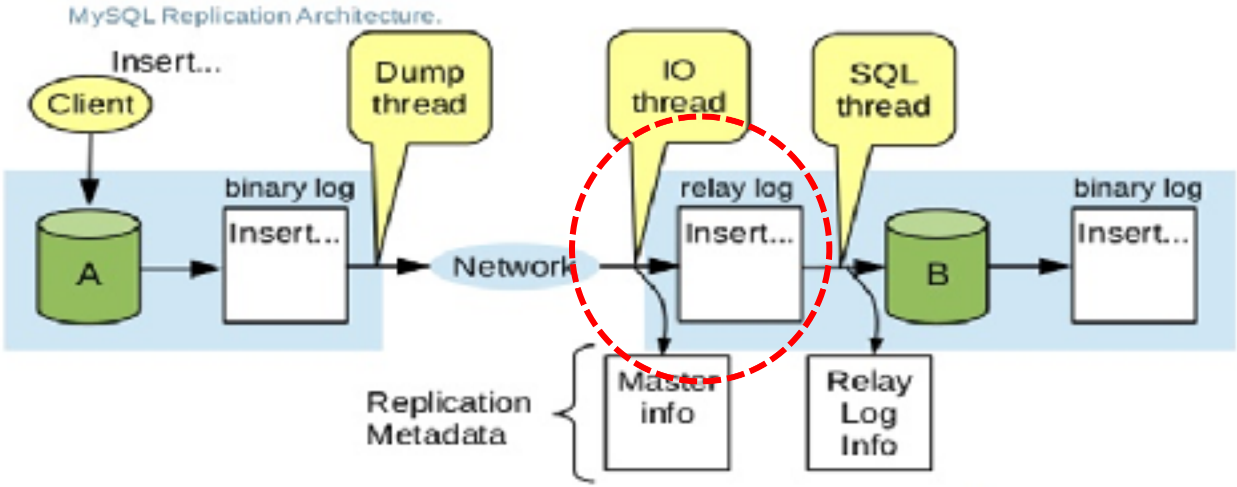
Optimized slave

p Problem : IO thread delay

ü Lock conflicts

ü Random small IO

ü Serialization



Version	Type	SGL TX(ms)	RTT	Benchmark
MySQL5.7	async	3.82	2.60	100%
MySQL5.7	semi-sync	8.33	2.60	46.30%

$T_{sql} + T_{engine} = 3.82ms$, $T_{sql} + T_{engine} + T_{replica} = 8.33ms$, $T_{replica} = T_{network} + T_{fsync}$

$T_{fsync} = T_{replica} - T_{network} = 8.33-3.82-2.60 = 1.91ms$

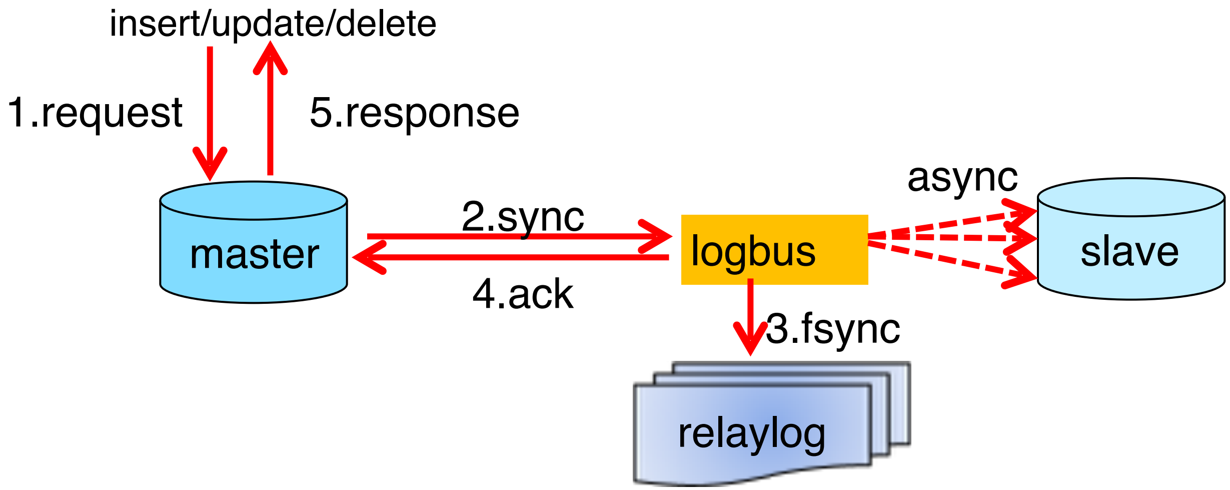
Optimized slave

p Solutions : fast log transfer

ü Less locks

ü Merge IO

ü multi-threaded async replay on salve

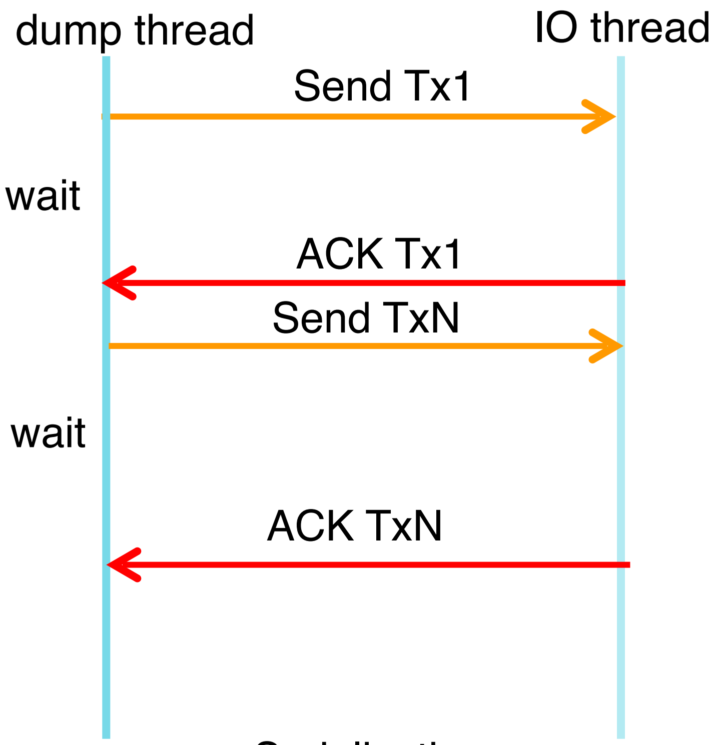


Version	Type	SGL TX(ms)	RTT	Benchmark
MySQL5.7	async	3.82	2.60	100%
MySQL5.7	semi-sync	8.32	2.60	46.30%
MySQL5.7	logbus	5.92	2.60	66.79%

optimized master

p Problem : Serialization

ü QPS $_{(max)}$ = 1000 / RTT = 1000 / 3 = 330.



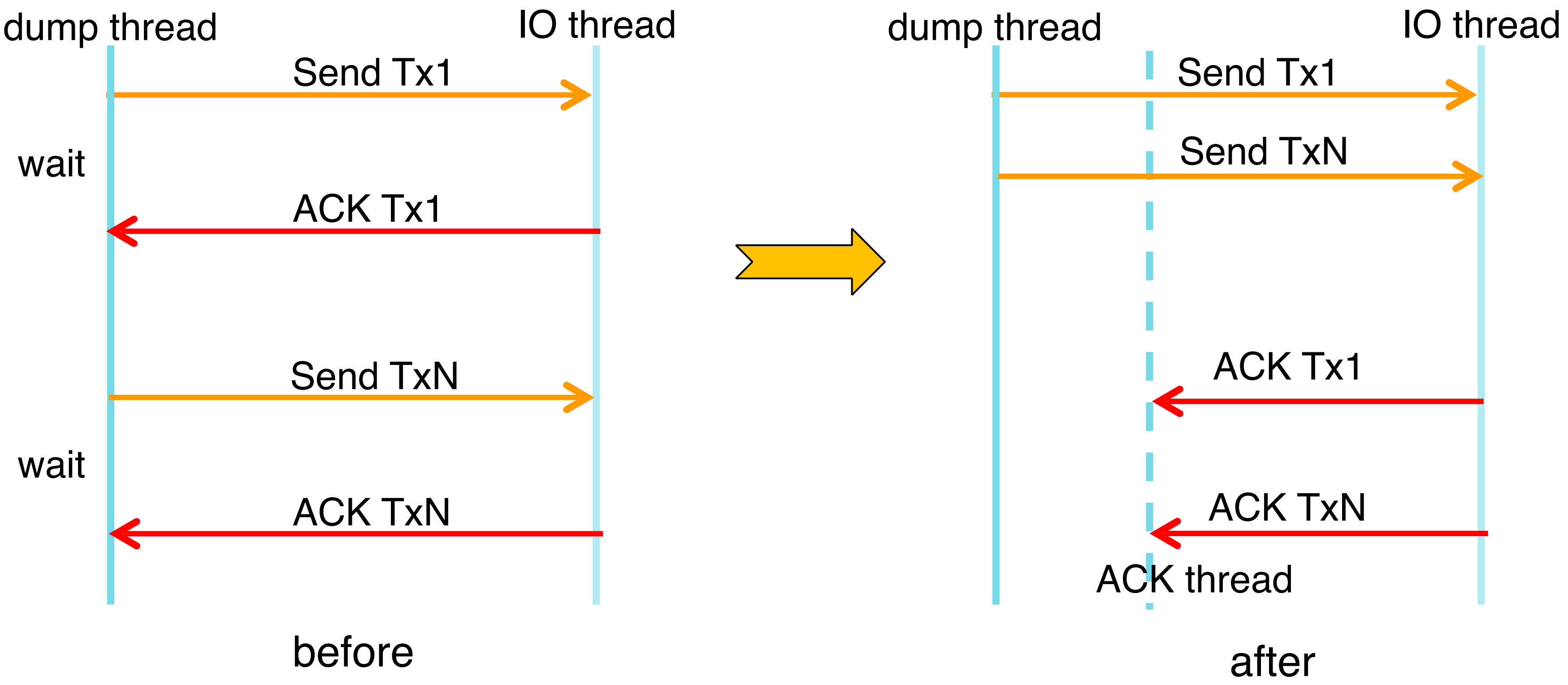
Serialization

optimized master

p Solution: break into two threads

ü Dump thread

ü Ack thread



optimized master

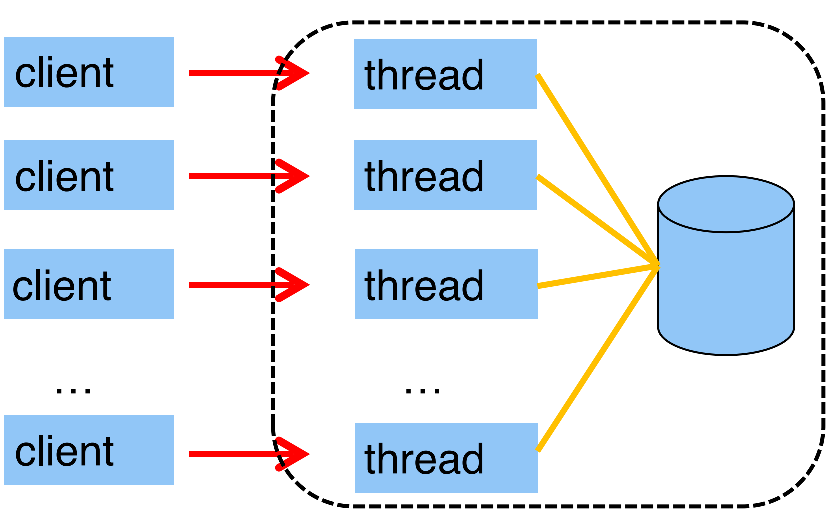
p Problem

ü Cost of threads

ü Thread blocking idle

p Solution

ü Thread pool(MariaDB)



MySQL work model

```
while(1)
{
    receive sql;
    execute sql(wait response);
    rsp client;
}
```

Thread workflow

HA

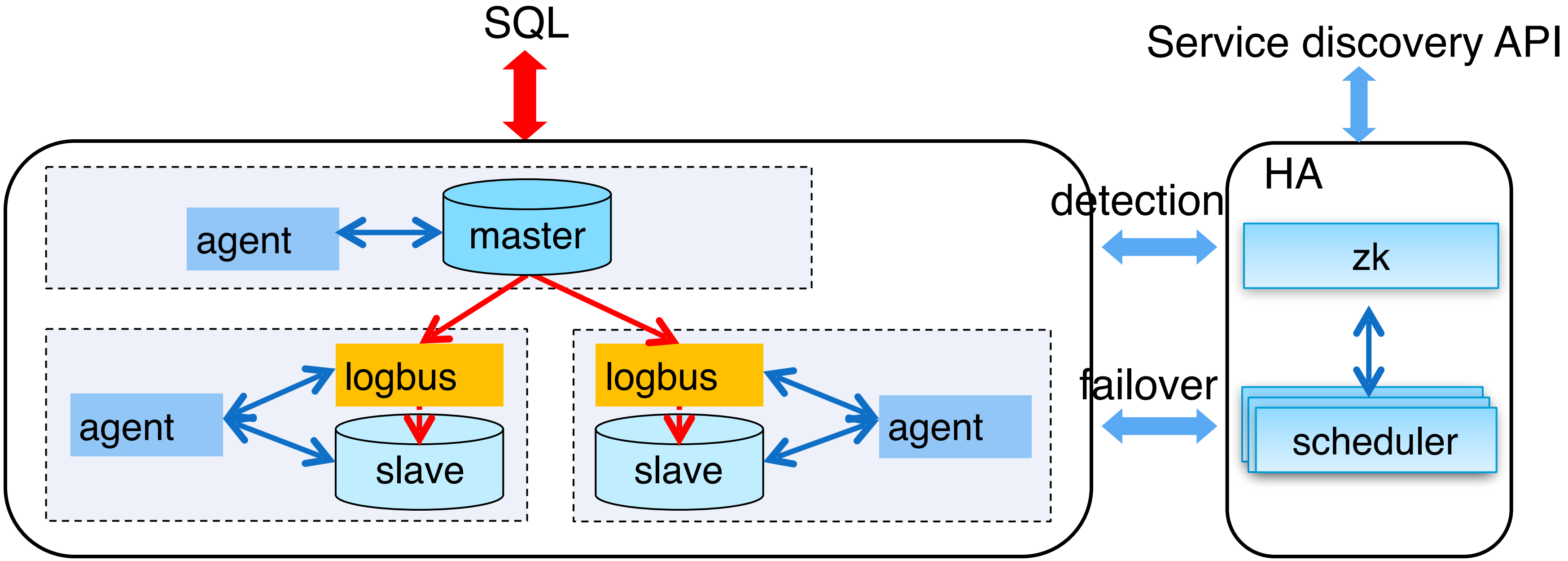
p How-to less RTO?

ü Fault detection : zookeeper + lease

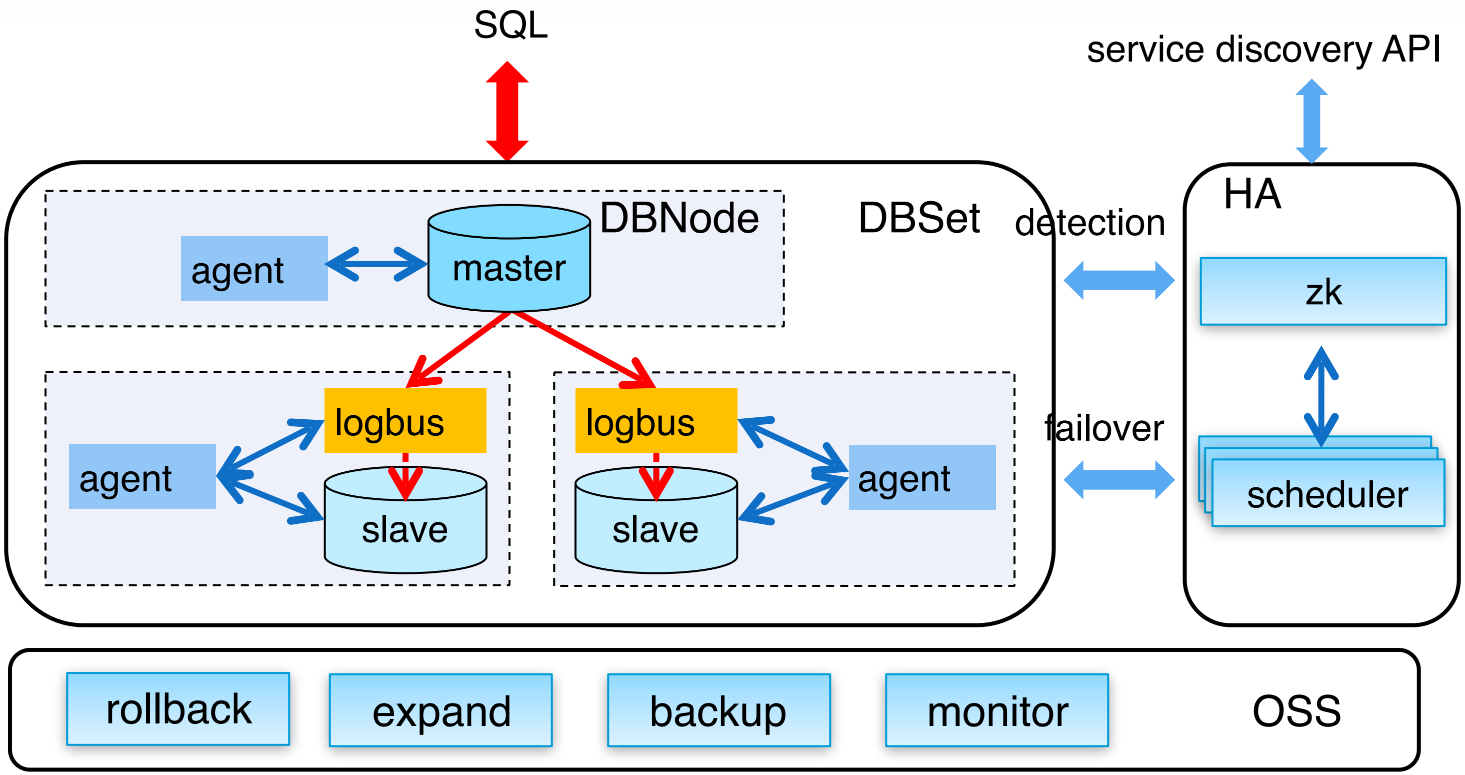
ü Failover : multi-thread replay

ü Ability to discover service

p RTO(aws 60-120s,ecdb <30s)



architecture



Logbus:fast log transfer

Agent : monitor mysqld/logbus

DBNode : logbus + agent + mysqld

HA： Fast fault detection & failover

Thank You

CONTACT US

- WeChat Group