

MySQL Group Replication 深度剖析及实践

冯光普

2018-08-11

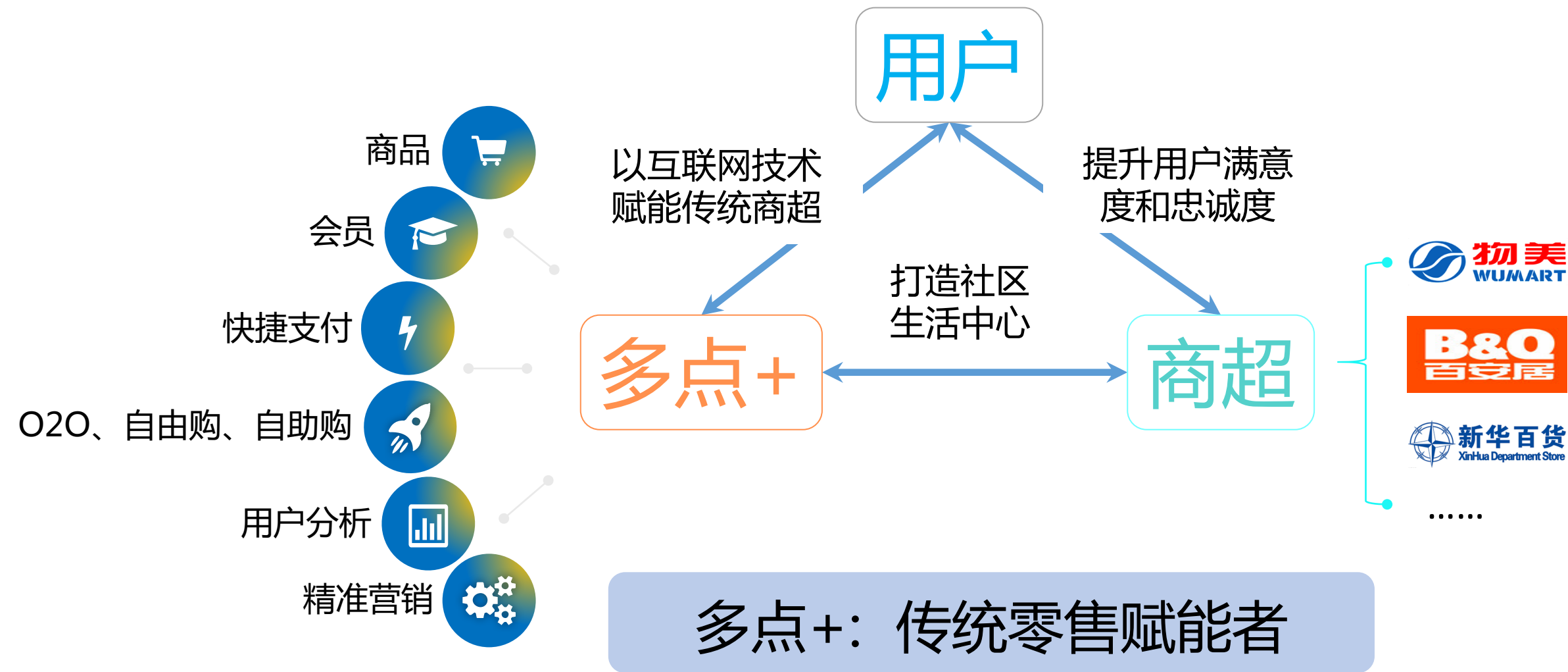


About me

- 『前』 阿里MySQL内核研发
 - Thread pool / Statement timeout
 - SELECT ... FROM UPDATE / Thread running ctl
 - TokuDB / Galera Cluster
- 多点数据库负责人
 - MySQL、PG、Redis、MongoDB
 - 自动化运维平台



About 多点Dmall



主要内容

- MGR特性
- 集群整体架构
- 集群数据同步、冲突检测、流控
- 集群性能分析
- 适用场景及实践
- Q & A

MGR特性

高可用

- 自动failover
- 不丢数据

准同步复制

- 大多数节点ACK
- 最终一致
- 延迟控制

多节点写

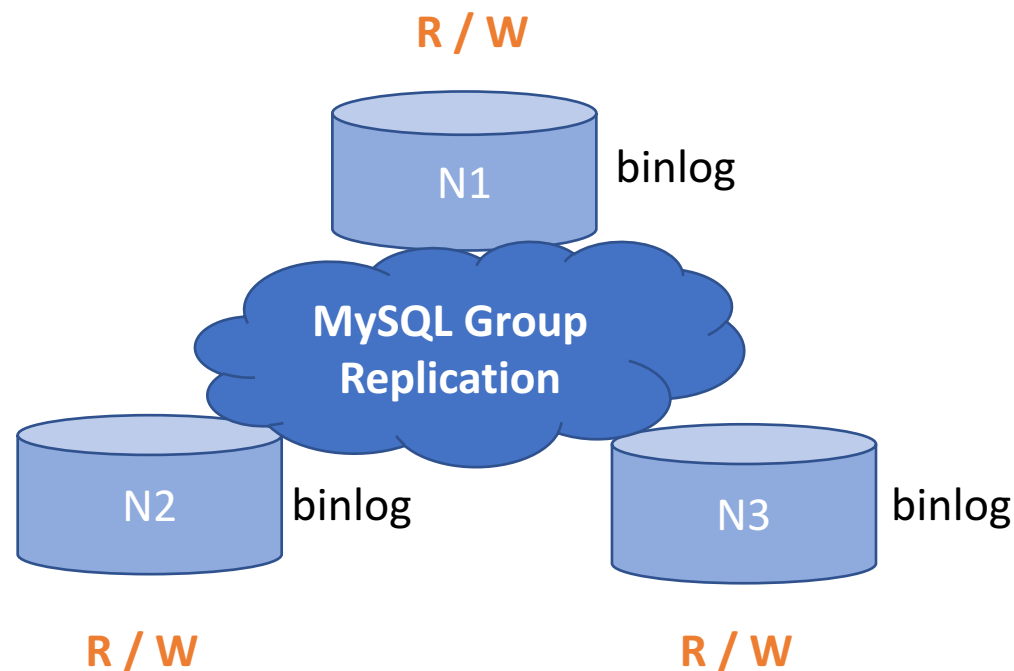
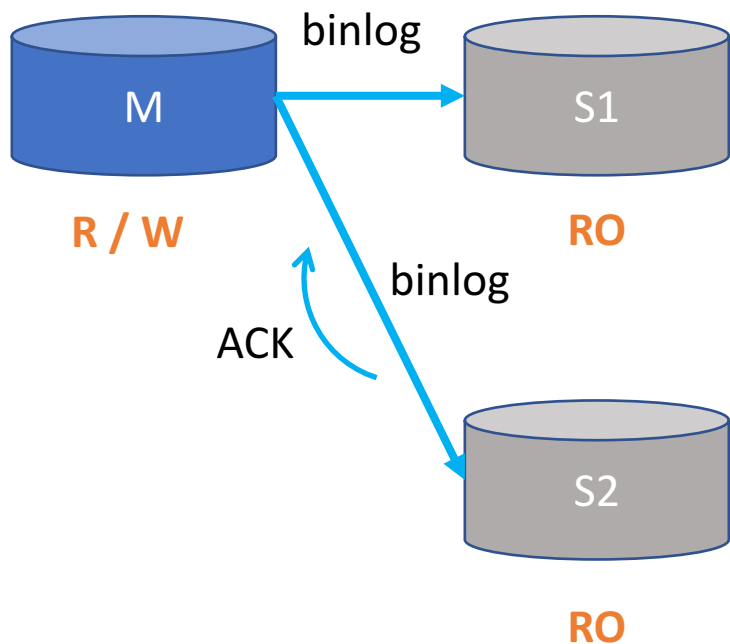
- 自动冲突检测
- Single Primary

分布式集群

- Share Nothing
- Paxos
- As a plugin

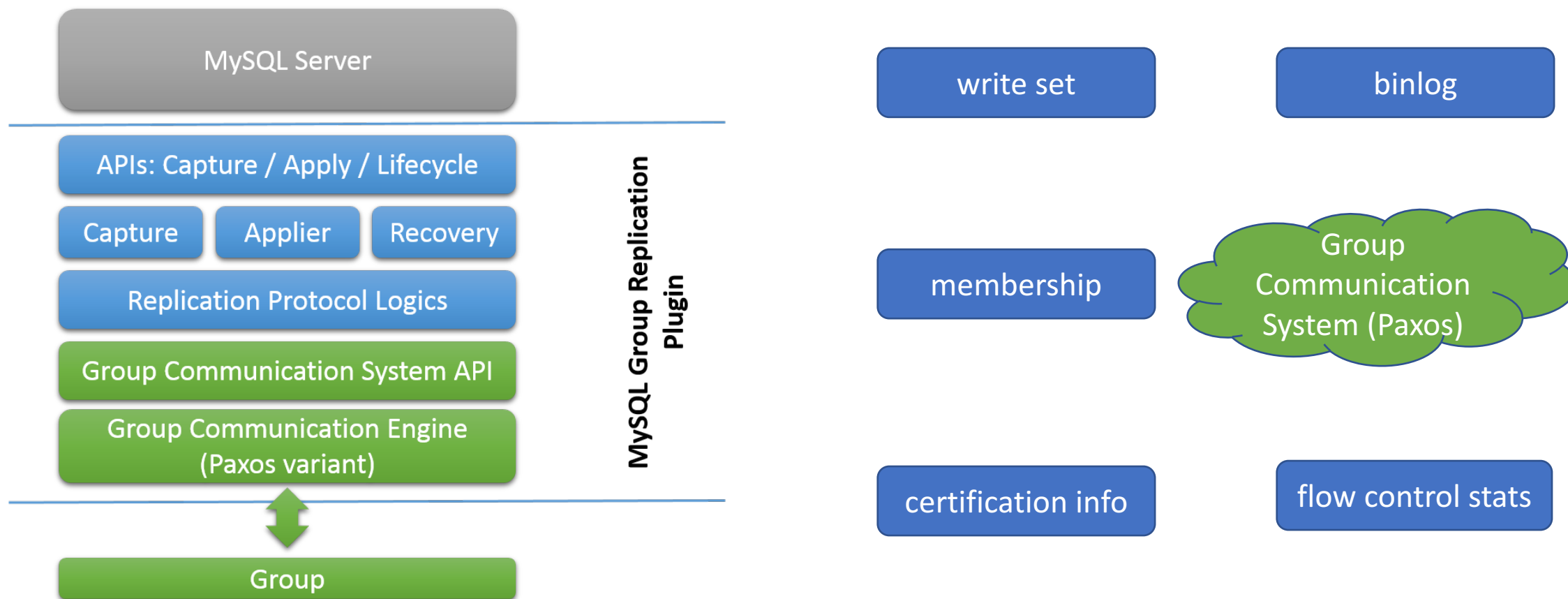
MGR集群架构

- 原生异步复制/半同步复制 VS. MGR

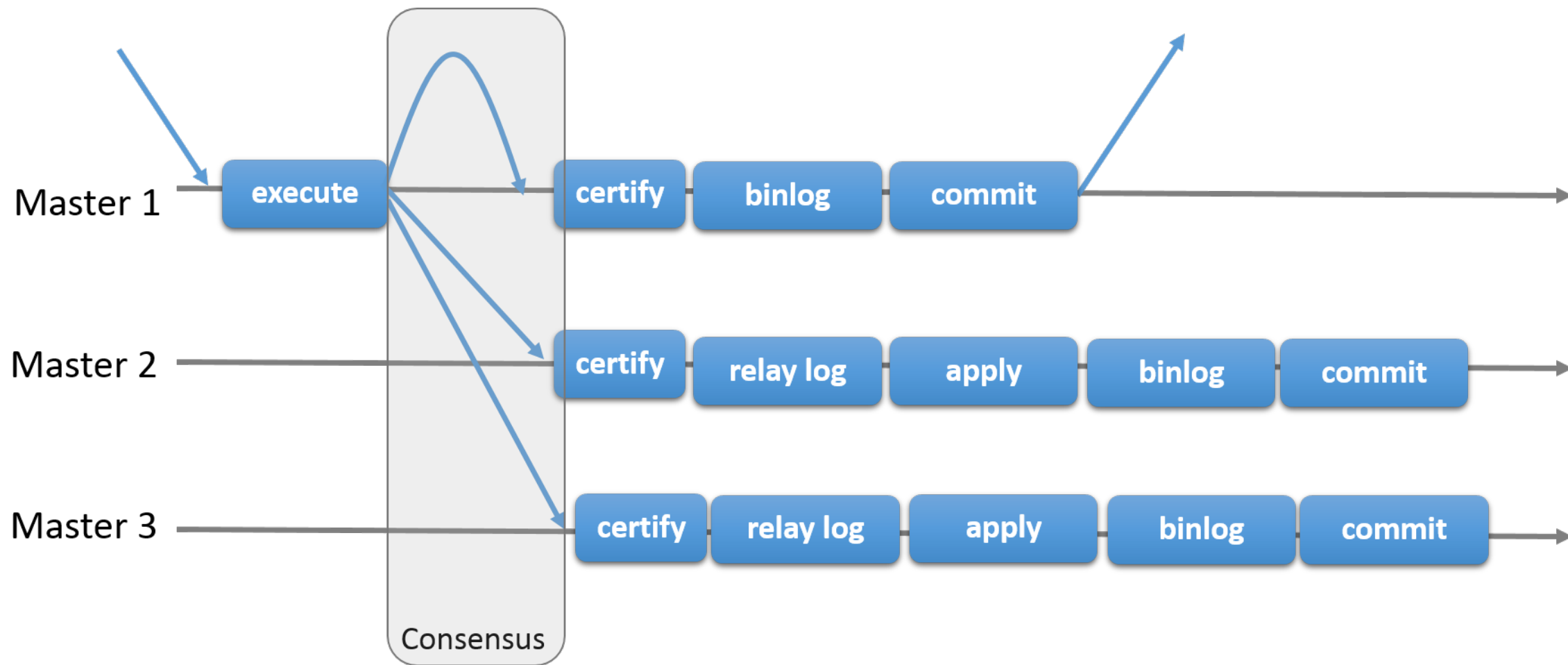


MGR集群架构

- Group Communication System (**GCS**)



MGR集群同步 / replication



MGR集群同步 / replication

数据一致性保证

- State Machine
 - 分布式系统中，各节点执行相同的数据变更序列，状态最终一致
 - Paxos协议
- Certification
 - 各节点独立执行冲突检测，并执行后续动作，不需要其他节点ACK
 - 数据变更中包含事务上下文信息，各节点冲突检测结果一致

MGR集群同步 / replication

- Write set
 - 事务更新涉及的 Primary key / Unique key 集合
- Transaction context log event
 - *std::list<const char*> write_set;*

```
-- sql/handler.cc
```

```
int handler::ha_write_row / ha_update_row / ha_delete_row
```

```
int binlog_log_row
```

```
-- sql/rpl_write_set_handler.cc
```

```
void add_pke
```

```
static void generate_hash_pke
```

```
thd->get_transaction()->get_transaction_write_set_ctx()->add_write_set(hash);
```

MGR集群同步 / replication

```
-- rapid/plugin/group_replication/src/observer_trans.cc
int group_replication_trans_before_commit(Trans_param *param)

Transaction_write_set* write_set= get_transaction_write_set(param->thread_id);
add_write_set(tcle, write_set);
tcle->write(cache);           // 1. Write transaction context to group replication cache.
gle->write(cache);           // 2. Write Gtid log event to group replication cache.
transaction_msg.append_cache(cache);           // 3. Copy group replication cache to buffer.
transaction_msg.append_cache(cache_log);       // 4. Copy binlog cache content to buffer.

certification_latch->registerTicket(param->thread_id);
applier_module->get_flow_control_module()->do_wait();// 5. Flow control step
gcs_module->send_message(transaction_msg);      // 6. Broadcast Transaction Message
certification_latch->waitTicket(param->thread_id); // 7. Waiting for certifier's notify
```

MGR集群冲突检测 / certification

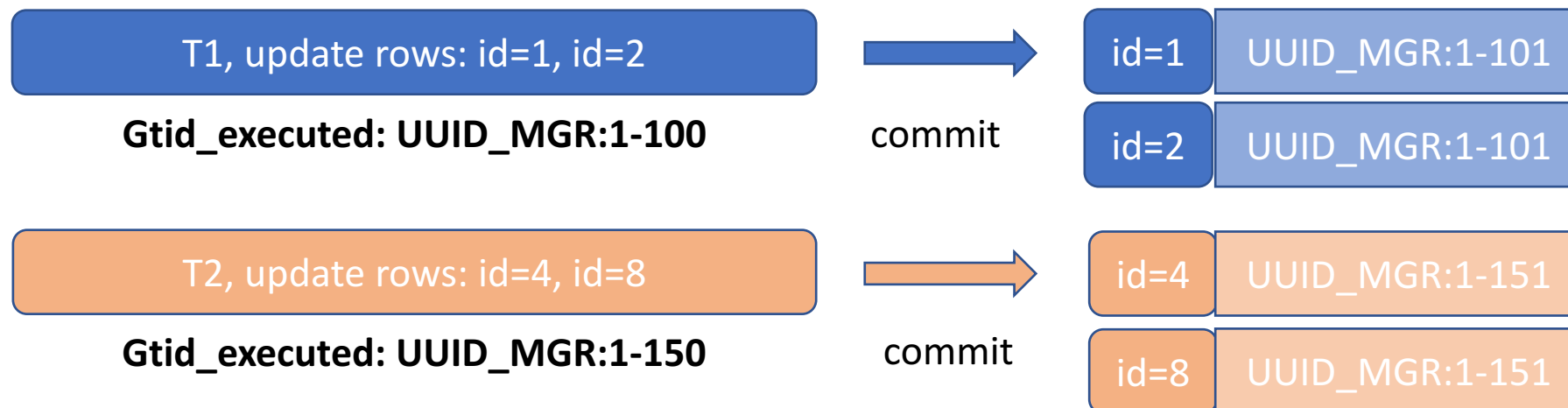
- Certification info

- 已通过冲突检测的write set及其快照版本

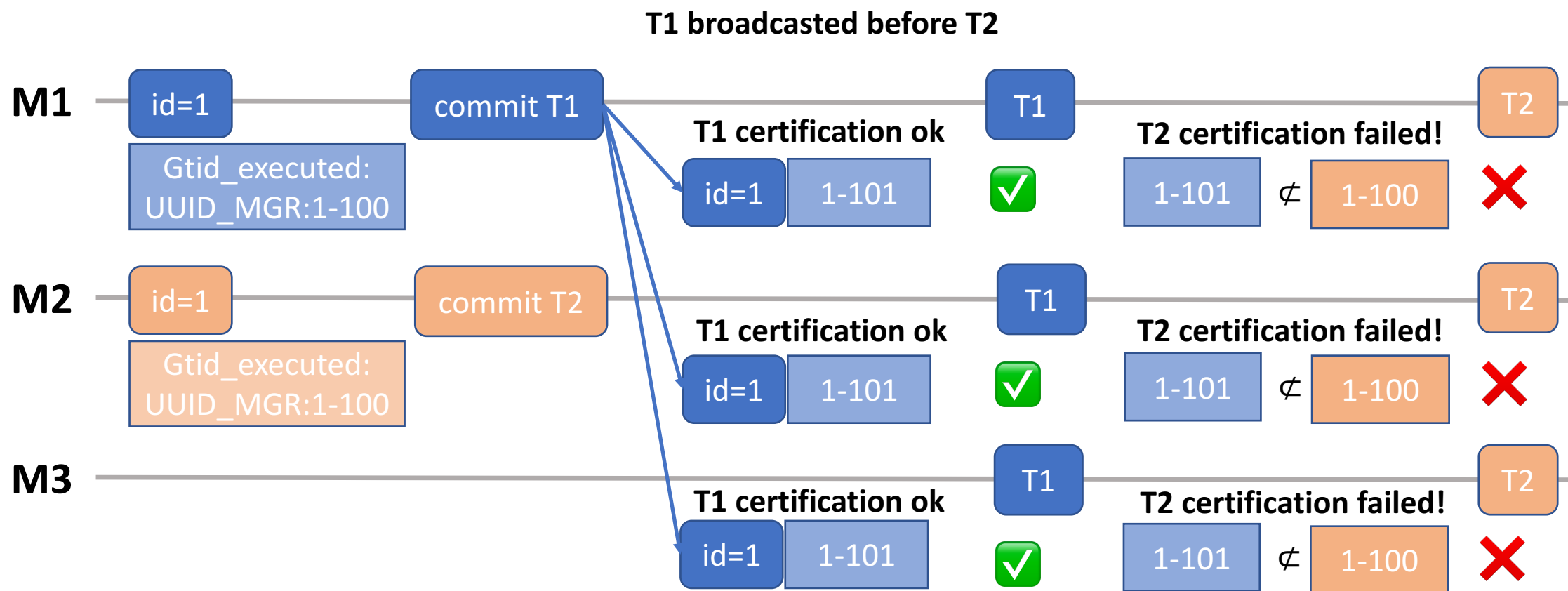
- `typedef std::map<std::string, Gtid_set_ref*> Certification_info;`

- certification通过条件：

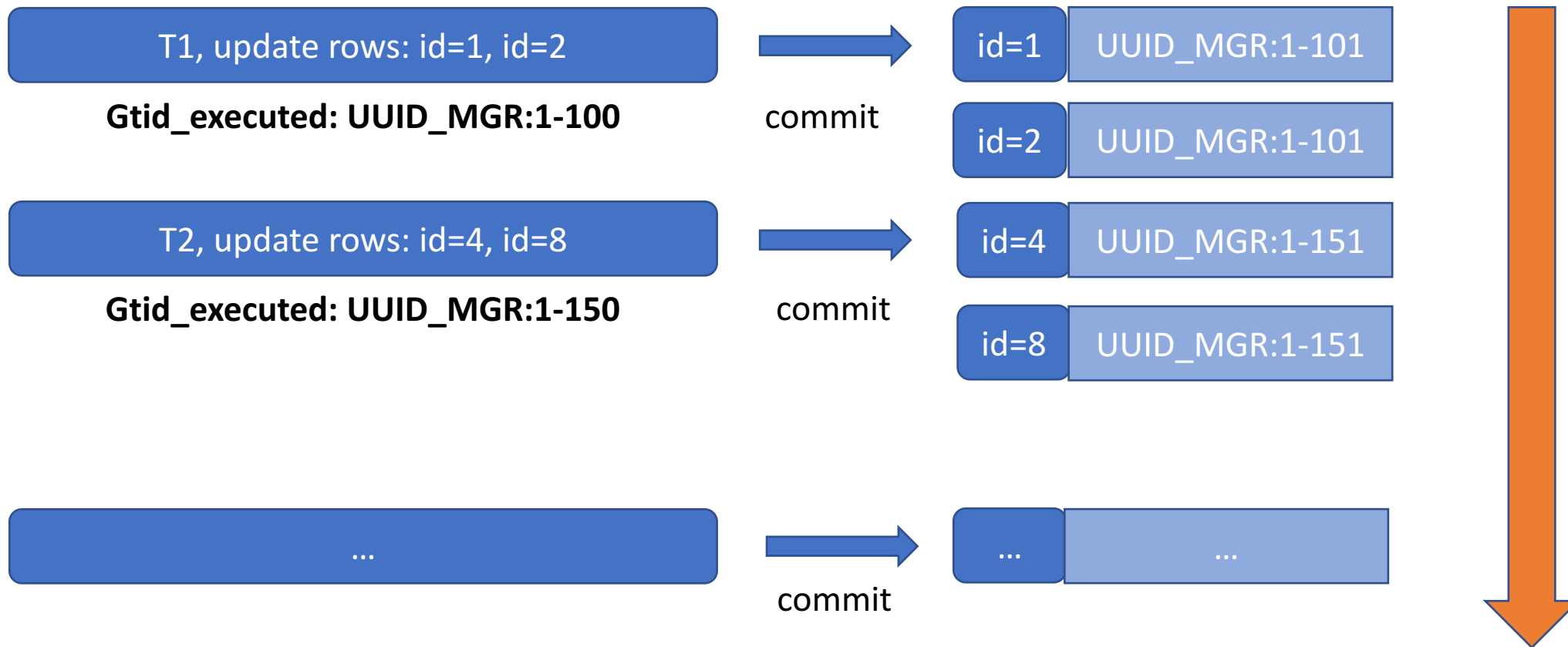
certification info snapshot version $\subset$ **transaction write set snapshot version**



MGR集群冲突检测 / certification



MGR集群冲突检测/ certification



Certification map 不断增大 !

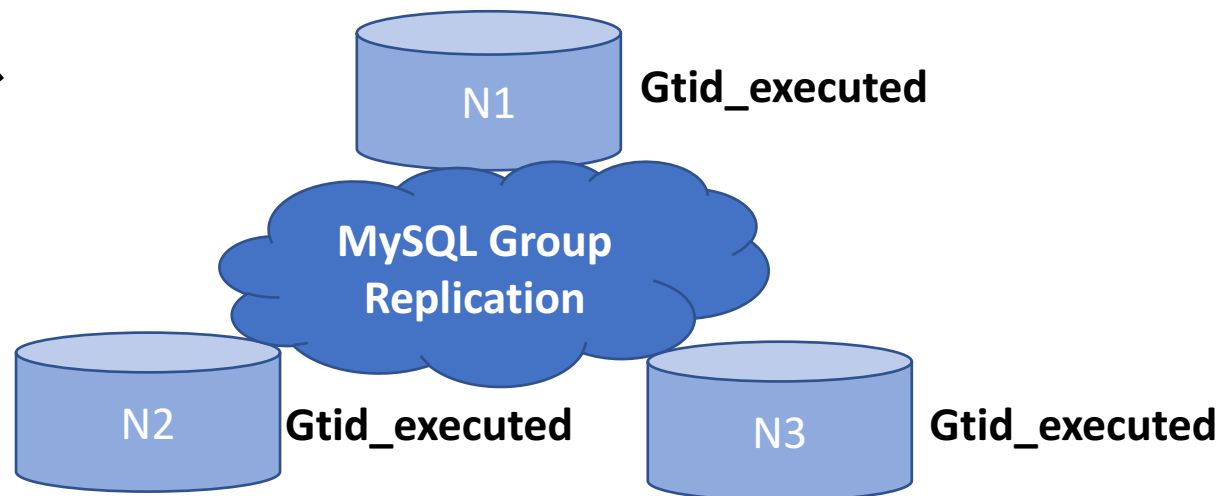
MGR集群冲突检测/ certification

- 节点之间定期交换事务执行信息

- Gtid_executed
- 每60s

- Certification map GC

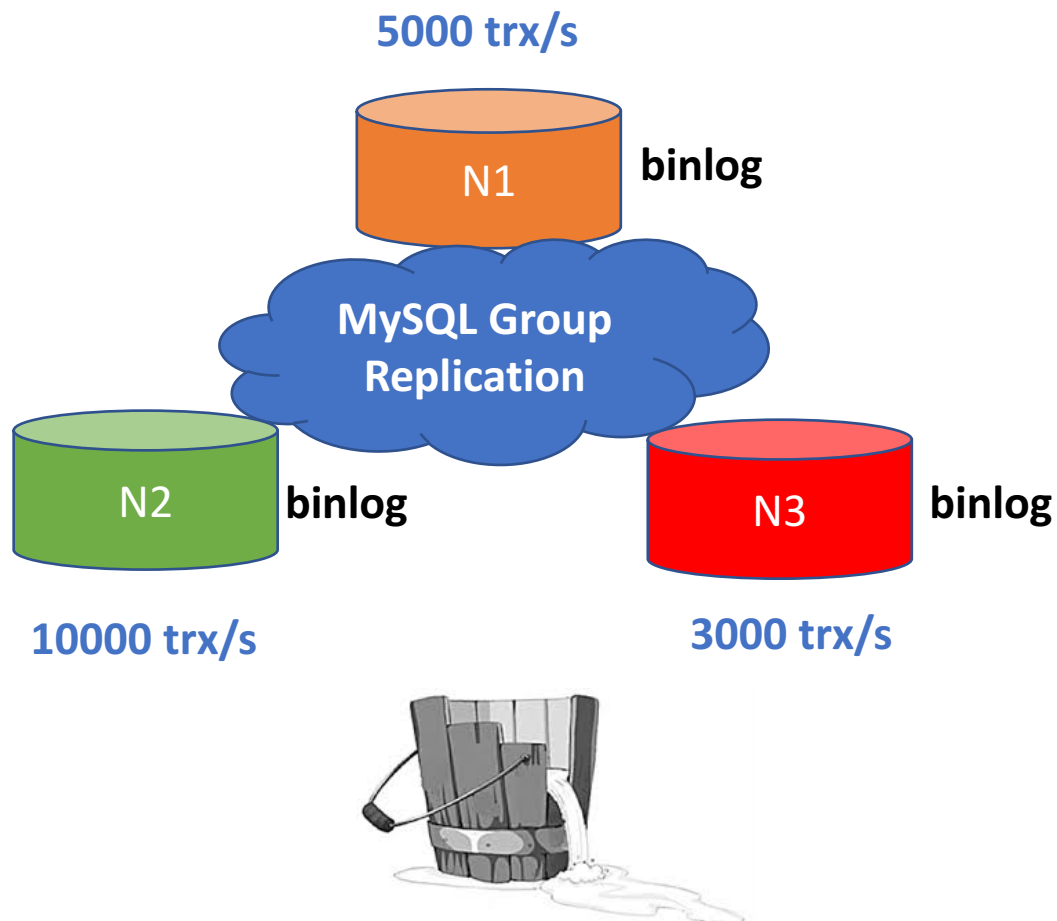
- Minimal Gtid_executed of all nodes
- Purge:



certification info snapshot version $\subset$ Minimal gtid_executed

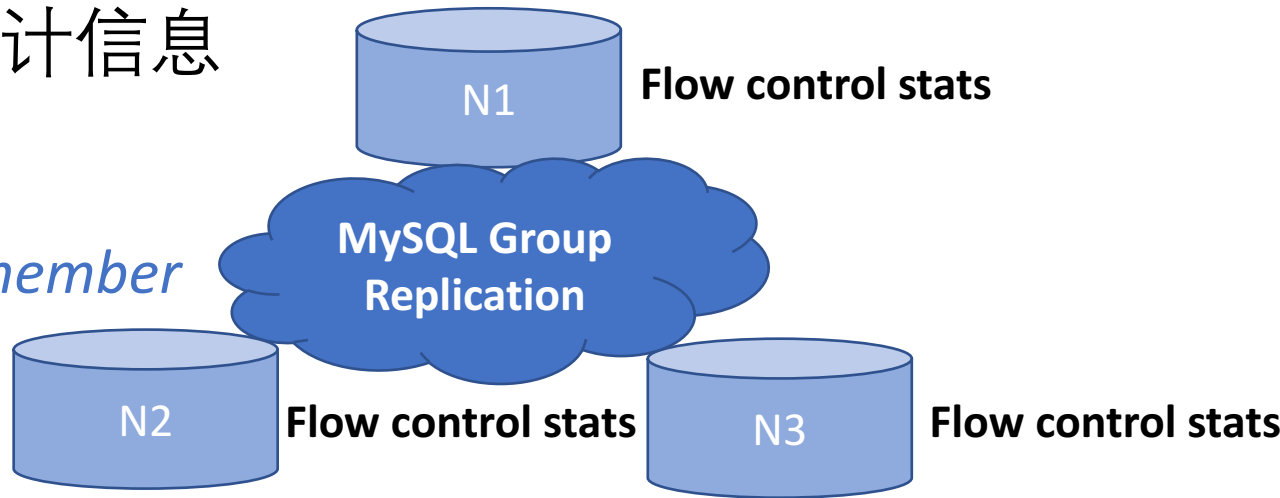
MGR集群流控 / flow control

- 各节点性能不完全一致
- 各节点异步 certification / apply
- Flow control
 - 保证集群延迟可控
 - 读写分离场景
 - 集群性能：木桶原理



MGR集群流控 / flow control

- 节点之间定期交换flow control统计信息
 - *certifier queue size*
 - *replication applier queue size*
 - *total # of remote trx applied in the member*
 - *total # of trx certified*
 - *total # of local transactions*

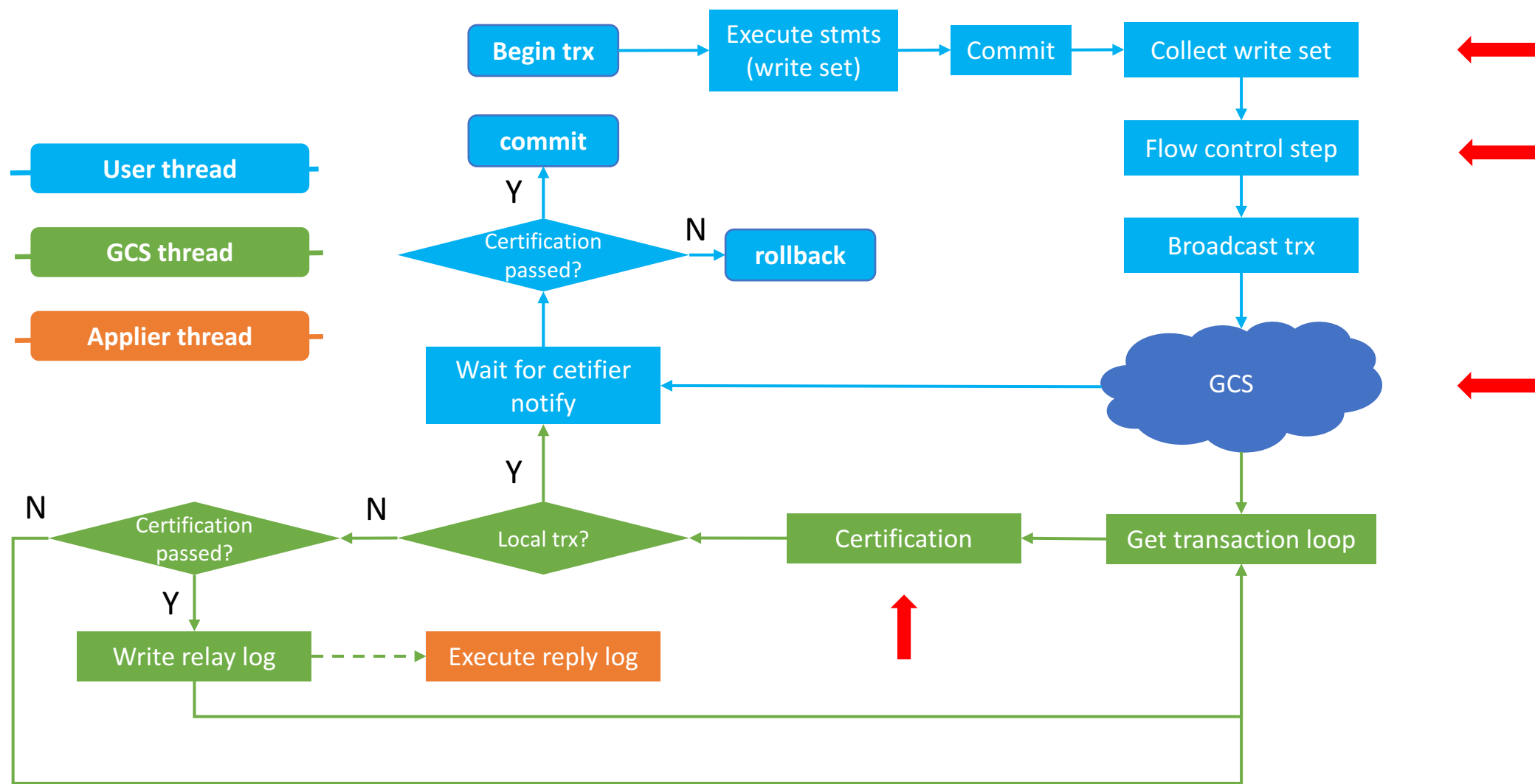


- 计算出最慢的节点处理能力 (N trx/s)
- 节点事务配额 : $\text{Quota} = 0.9 * (N / \# \text{ of write nodes})$

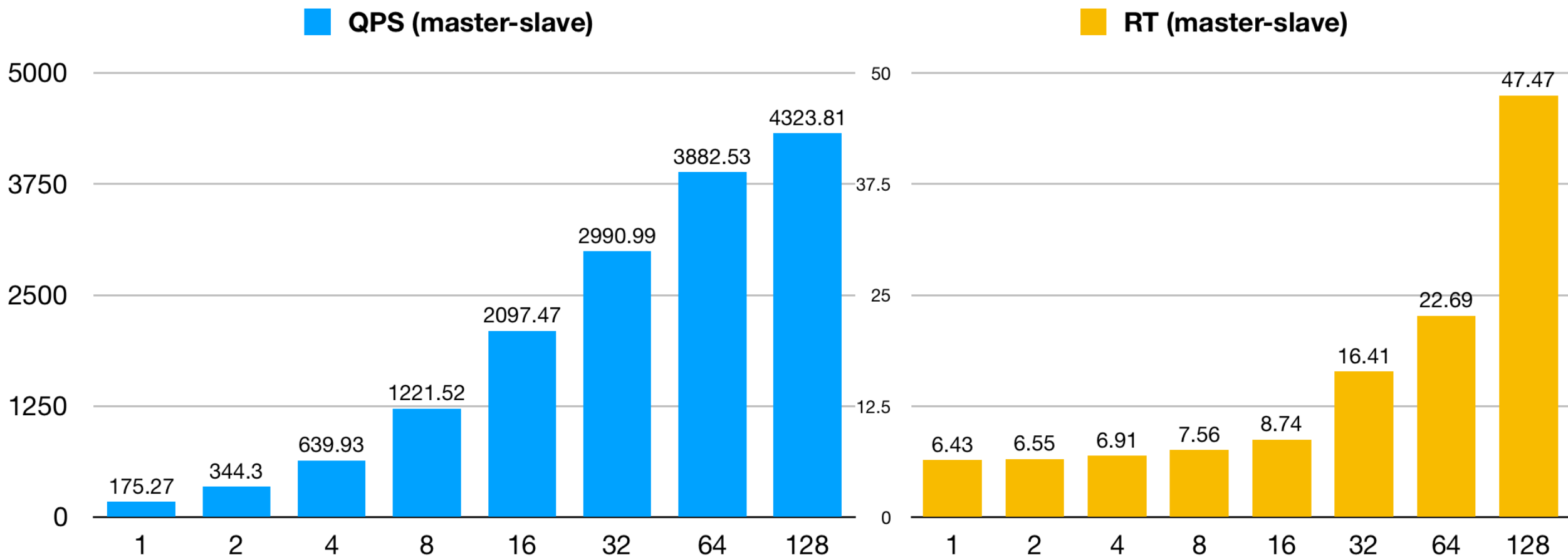
MGR集群流控 / flow control

- group replication flow control mode
 - QUOTA / DISABLED
- group replication flow control certifier threshold
 - default: 25000
- group replication flow control applier threshold
 - default: 25000
- group replication flow control hold percent
 - default: 10

MGR集群性能分析



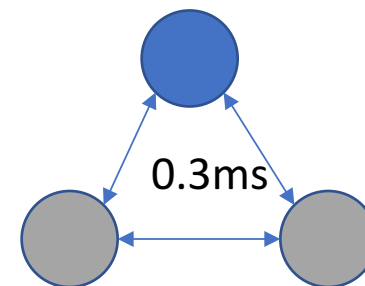
MGR集群性能分析



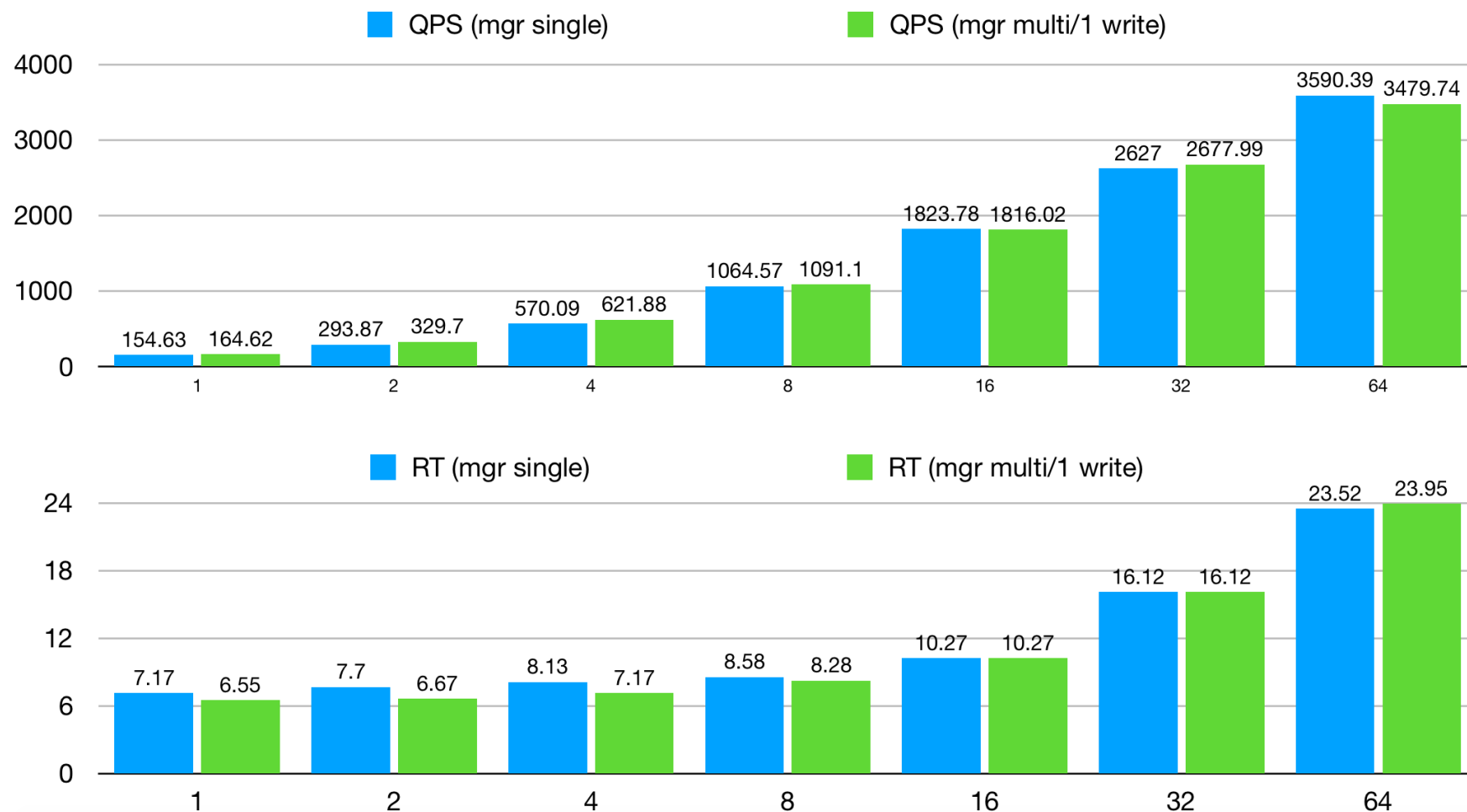
MGR集群性能分析



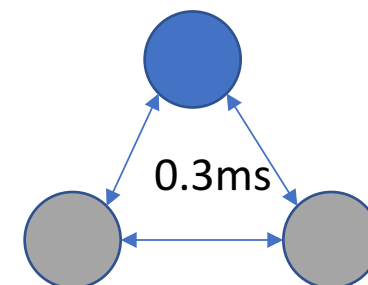
92%



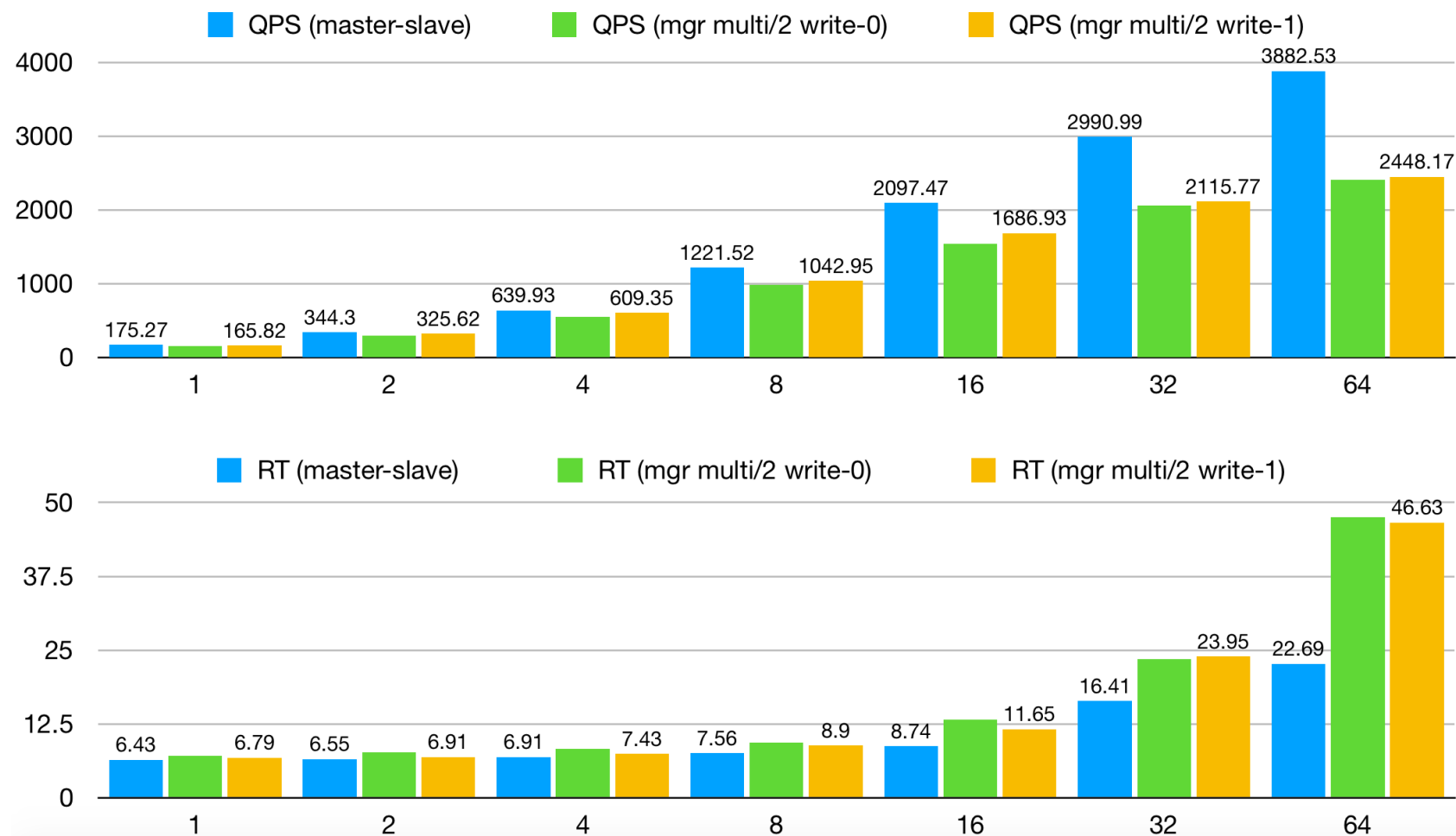
MGR集群性能分析



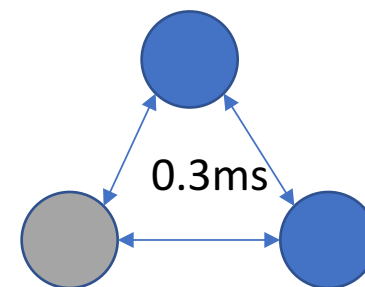
97%



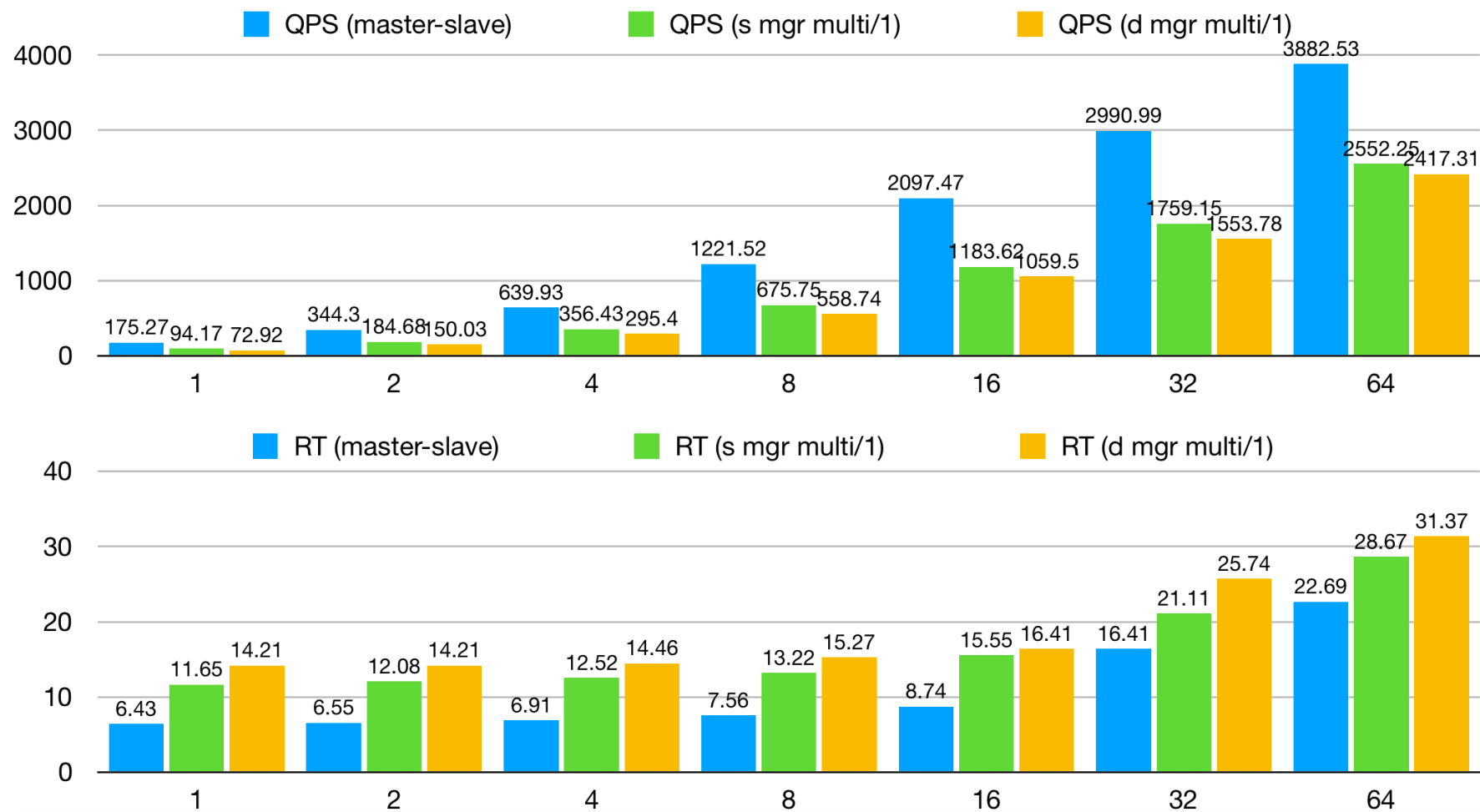
MGR集群性能分析



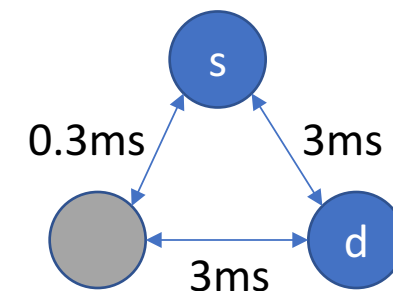
100%



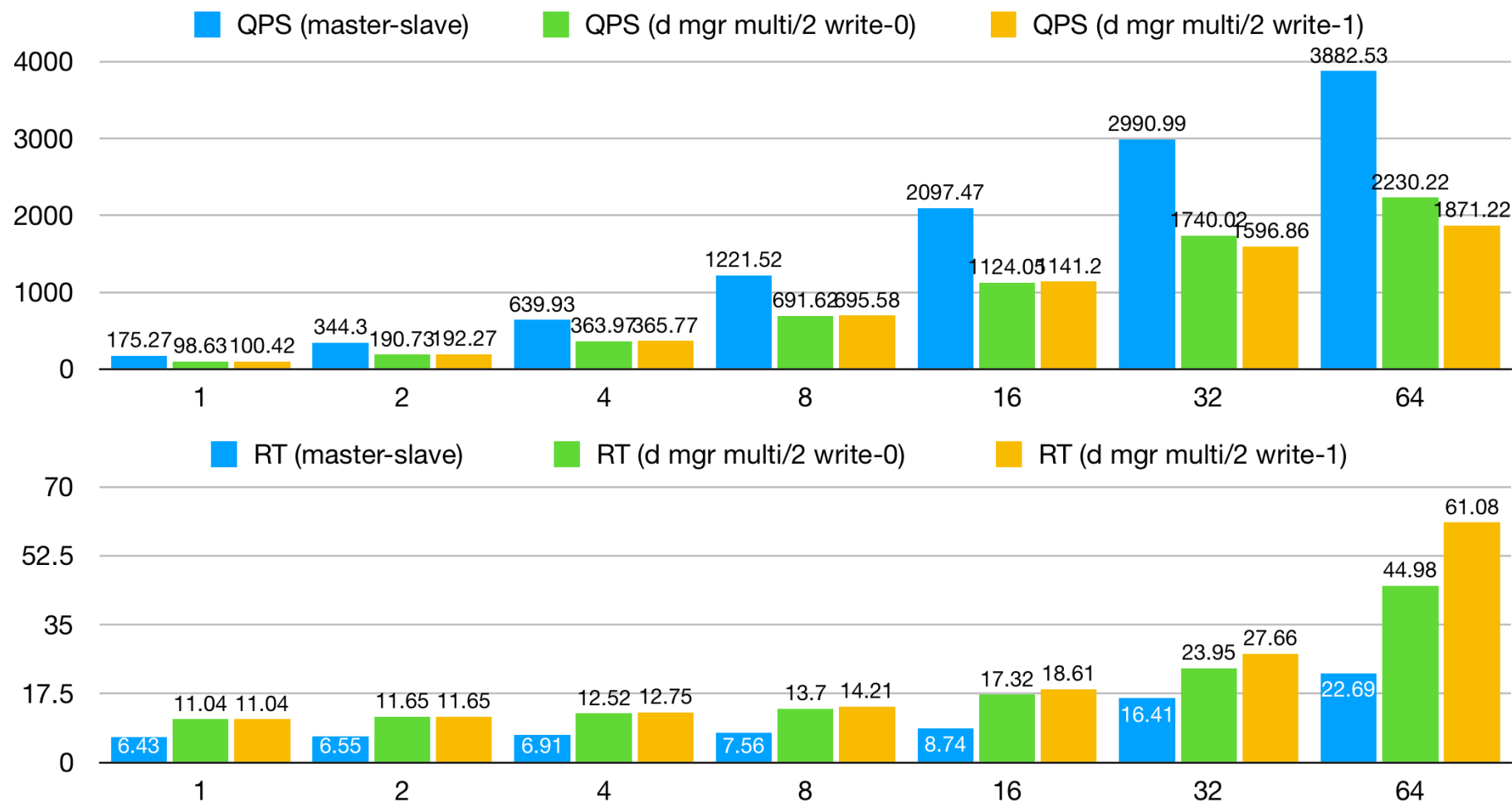
MGR集群性能分析



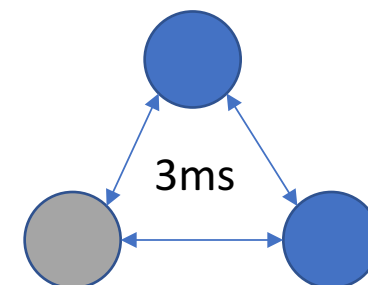
66%



MGR集群性能分析



100%



MGR集群性能分析

- Sysbench OLTP标准压测（CPU、内存都未到达瓶颈）
 - data: 15G, BP: 32G, 24CPU/48G
- 局域网内（RTT < 0.5ms）
 - MGR性能单节点写入能达到原生异步复制90%
 - Certification对性能影响非常小，测试中为5%内
 - 多节点写，集群QPS吞吐可以达到原生异步复制，RT增大
- 同城异地网络（RTT < 5ms）
 - 受RTT影响，MGR性能单节点写入能达到原生异步复制67%+
 - 多节点写，集群QPS吞吐可以达到原生异步复制，RT增大

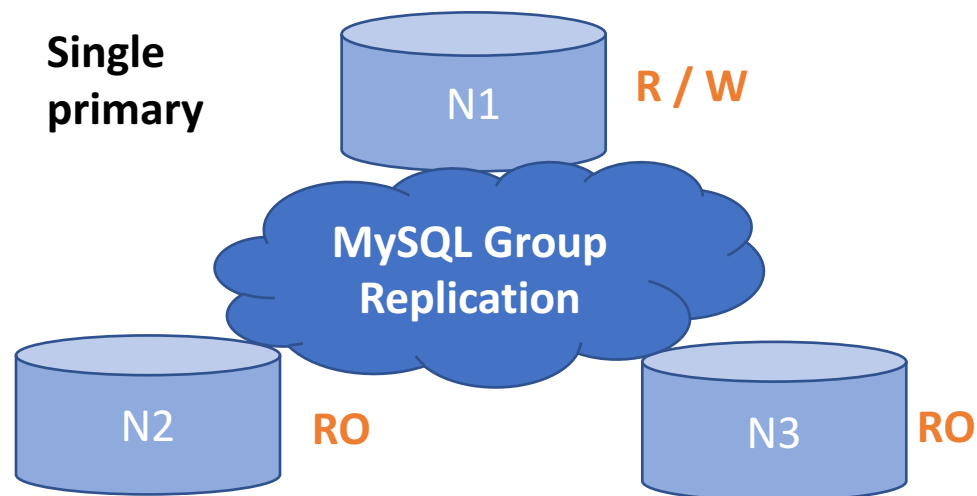
MGR适用场景 / 实践

- 集群部署网络环境
 - 局域网 / 同城网络
 - 异地网络 (RTT > 30ms) ?
- 限制：
 - InnoDB / ROW / GTID
 - 事务隔离级别：READ COMMITTED
 - [group replication transaction size limit](#)
 - Multi master模式：同一个表DDL / DML落到相同节点
 - <https://bugs.mysql.com/bug.php?id=89058>
- 性能可能下降30%以上 ==> 数据安全+高可用

MGR适用场景 / 实践

ProxySQL

Single
primary

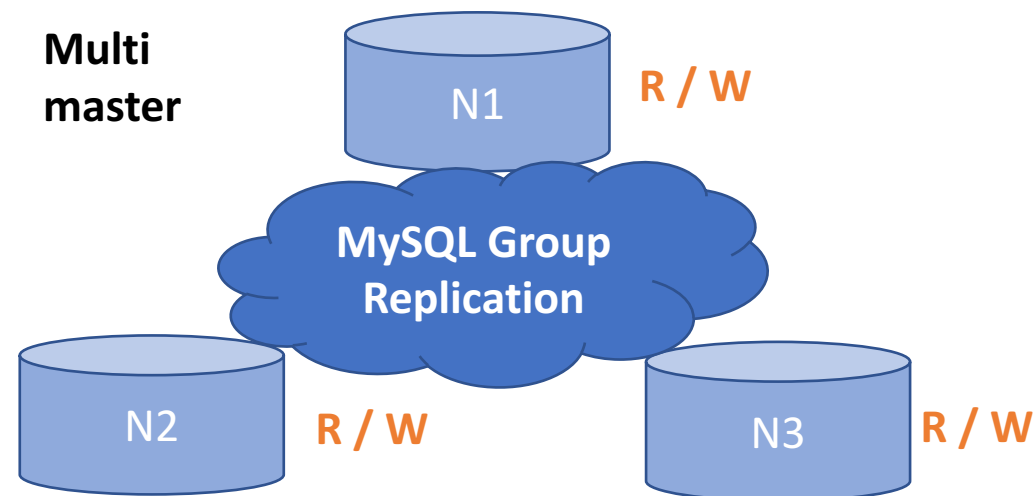


Custom failover
[group_replication_primary_member](#)

VIP/DNS

Haproxy / ProxySQL

Multi
master



Custom DNS HA

DNS

```
performance_schema.replication_group_members.Member_state = ONLINE
```

MGR适用场景 / 实践

```
mysql> SHOW GLOBAL STATUS LIKE 'group_replication_primary_member';
```

Variable_name	Value
group_replication_primary_member	c032e9c7-99e4-11e8-8ac3-525400034dc1

```
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM performance_schema.replication_group_members;
```

CHANNEL_NAME	MEMBER_ID	MEMBER_HOST	MEMBER_PORT	MEMBER_STATE
group_replication_applier	ae23bd5e-99e4-11e8-8867-5254007ac14e	172.16.3.11	3306	ONLINE
group_replication_applier	afb1b626-99e4-11e8-88e4-525400661c00	172.16.3.14	3306	ONLINE
group_replication_applier	c032e9c7-99e4-11e8-8ac3-525400034dc1	172.16.3.2	3306	ONLINE

```
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM performance_schema.replication_group_member_stats\G
```

```
***** 1. row *****
```

```
CHANNEL_NAME: group_replication_applier
```

```
VIEW_ID: 15336072637117830:3
```

```
MEMBER_ID: c032e9c7-99e4-11e8-8ac3-525400034dc1
```

```
COUNT_TRANSACTIONS_IN_QUEUE: 0
```

```
COUNT_TRANSACTIONS_CHECKED: 4
```

```
COUNT_CONFLICTS_DETECTED: 0
```

```
COUNT_TRANSACTIONS_ROWS_VALIDATING: 0
```

```
TRANSACTIONS_COMMITTED_ALL_MEMBERS: 30951147-d56e-46a5-bb8a-ac2739f4d742:1-30:1000018-1000023
```

```
LAST_CONFLICT_FREE_TRANSACTION: 30951147-d56e-46a5-bb8a-ac2739f4d742:1000024
```

```
1 row in set (0.00 sec)
```

Q & A



Thanks!