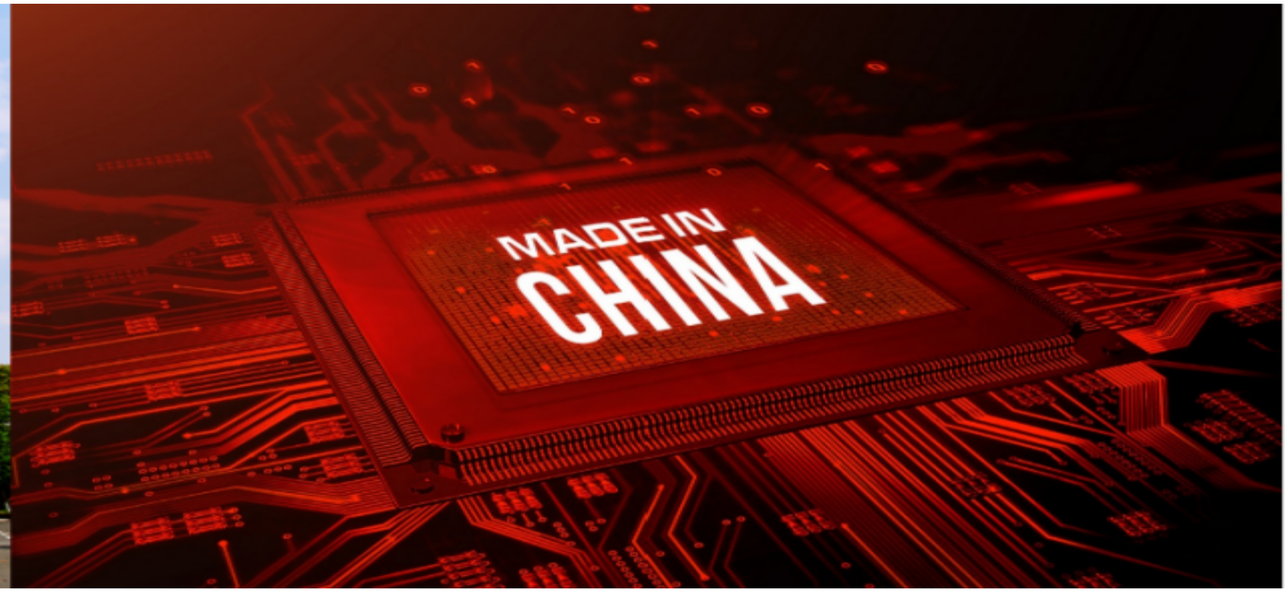
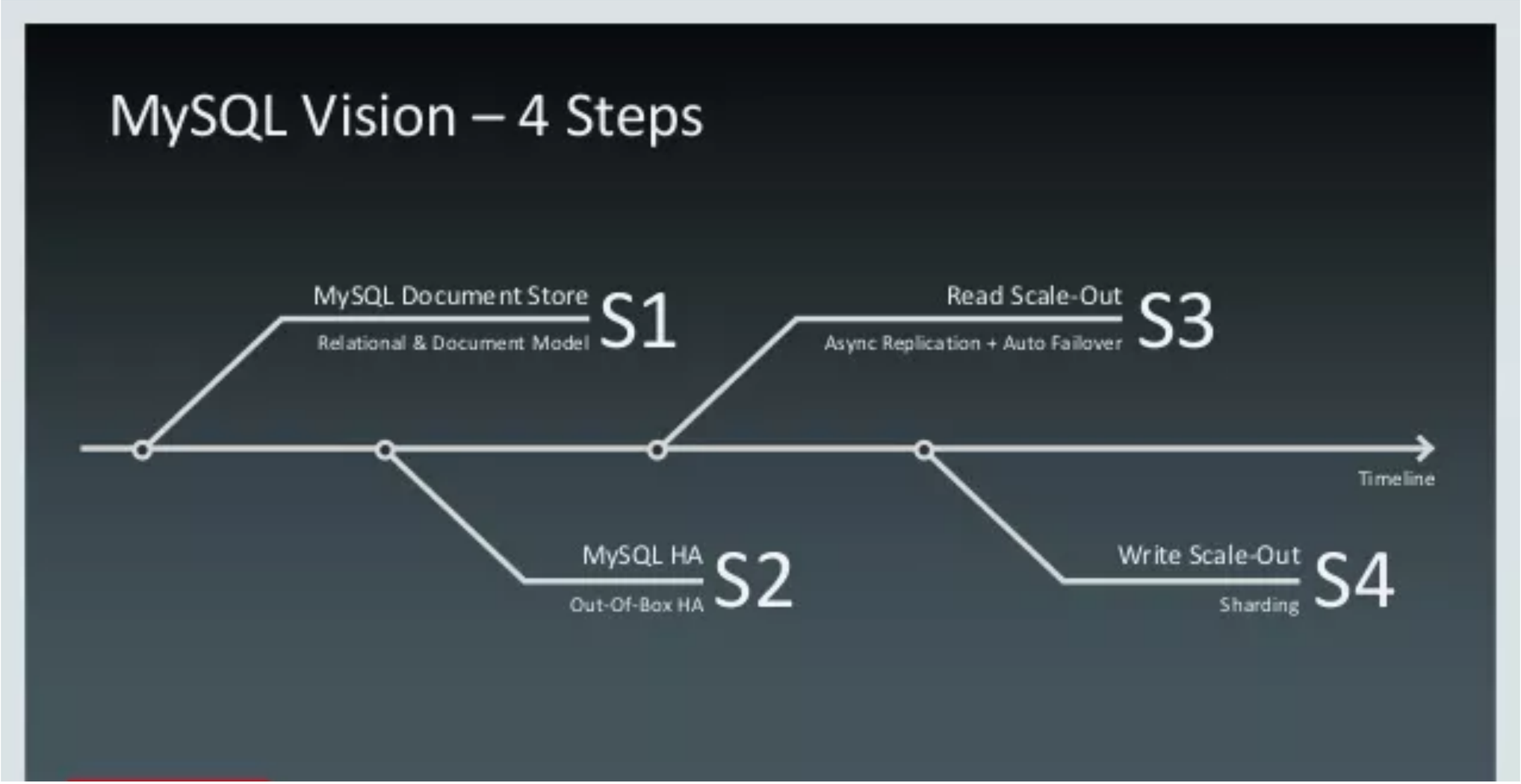


MySQL高可用自动化切换之MGR

关于我

- 叶金荣
- 万里数据库开源生态负责人
- Oracle MySQL ACE Director
- 腾讯云TVP

什么是MGR


MySQL Vision – 4 Steps
MySQL Document Store
Relational & Document Model
S1
Read Scale-Out
Async Replication + Auto Failover
S3
Timeline
MySQL HA
Out-Of-Box HA
S2
Write Scale-Out
Sharding
S4

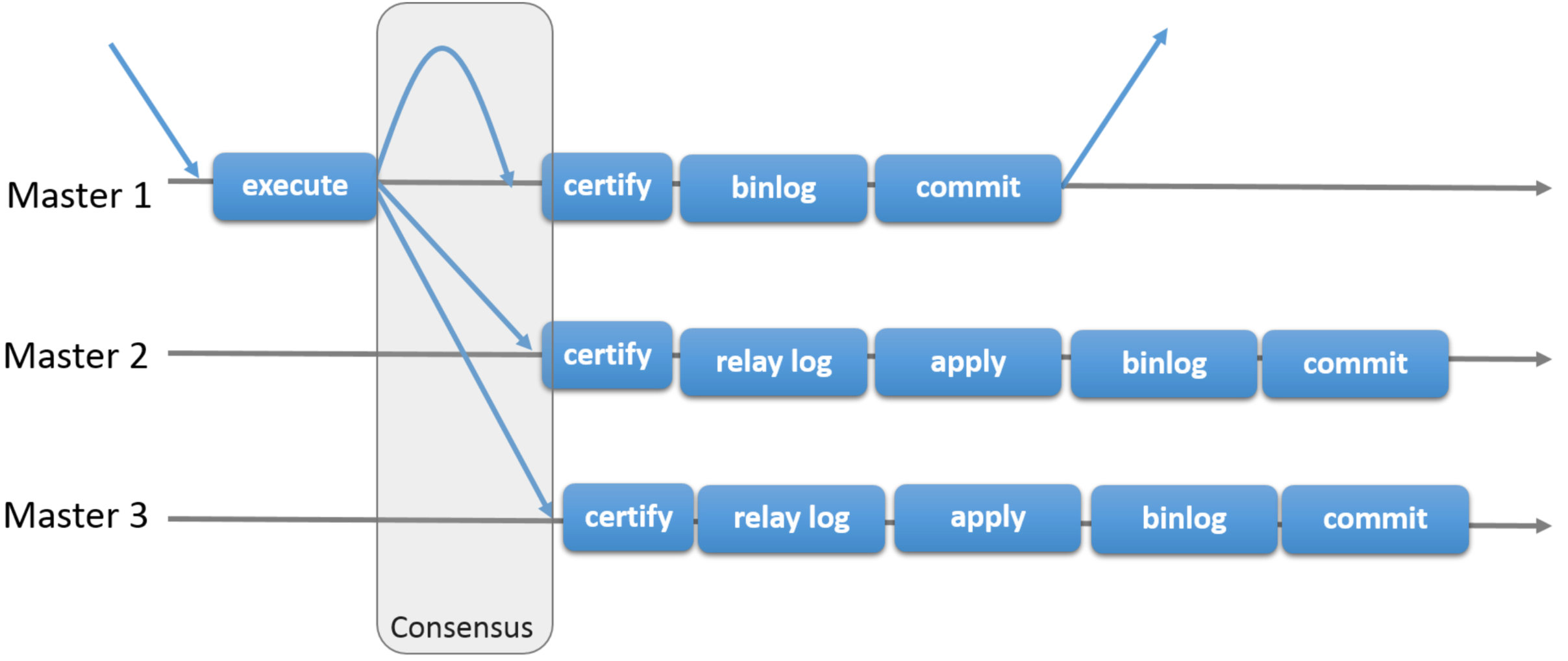
MGR

- MySQL Group Replication，简称MGR/GR，组复制
- shared-nothing
- 数据一致性及高可用集群方案
- 支持故障自动检测及多节点并行写
- 由一组MySQL实例构成，每个实例都有一份完整的数据，实例间通过组通讯消息系统（GCS）交互通信协同。GCS可保证消息的原子性和消息在所有组成员的整体顺序一致

MGR

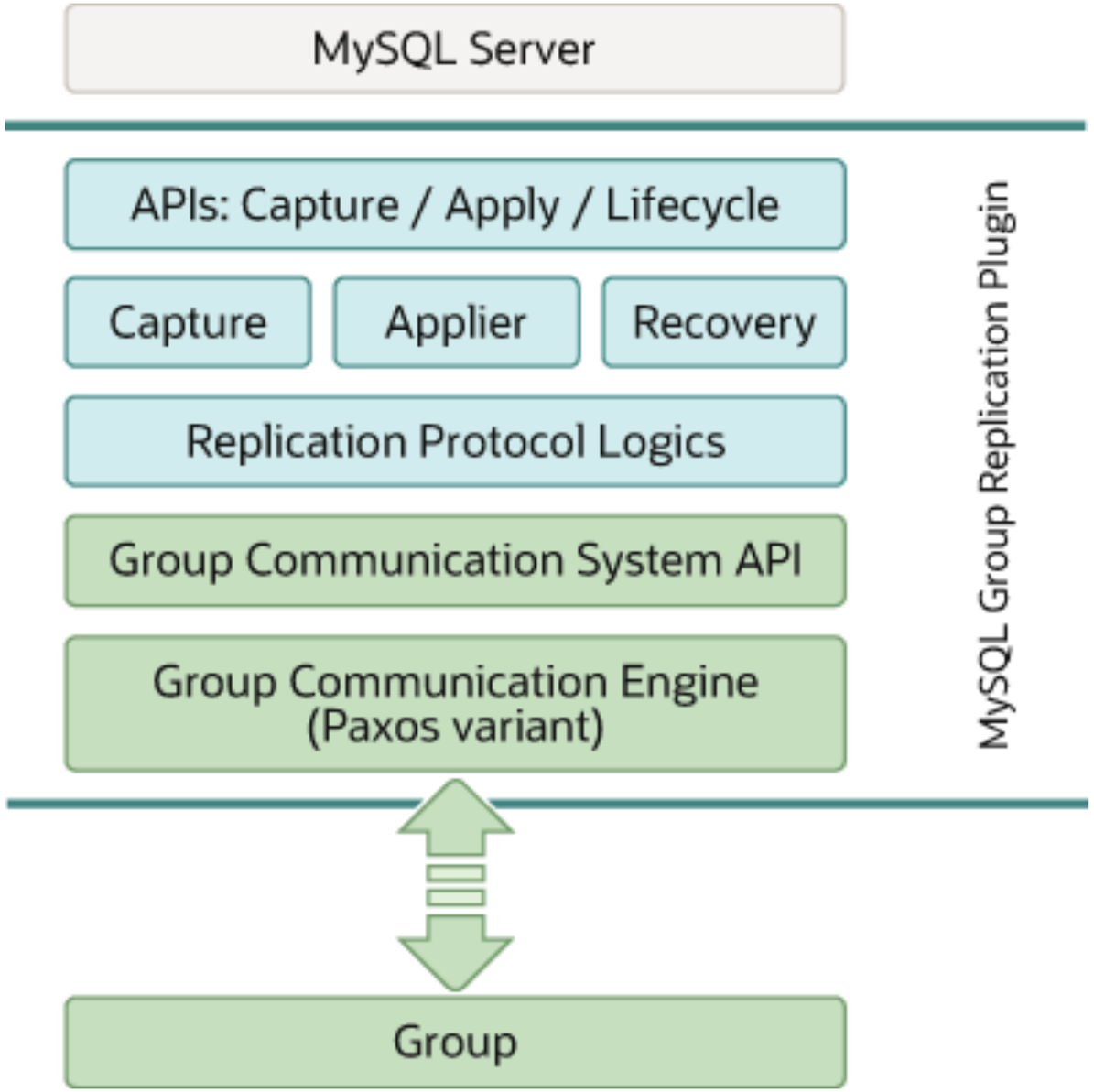
- 基于Paxos协议和原生复制，多数节点同意即可通过事务提议
- 具备高可用、自动故障检测功能，可自动切换
- 可弹性扩展，快速新增和移除节点
- 有单主和多主模式
- 支持多节点写入，具备冲突检测机制，可以适应多种应用场景需求

MGR


Master 1
execute
certify
binlog
commit
Master 2
certify
relay log
apply
binlog
commit
Master 3
certify
relay log
apply
binlog
commit
Consensus

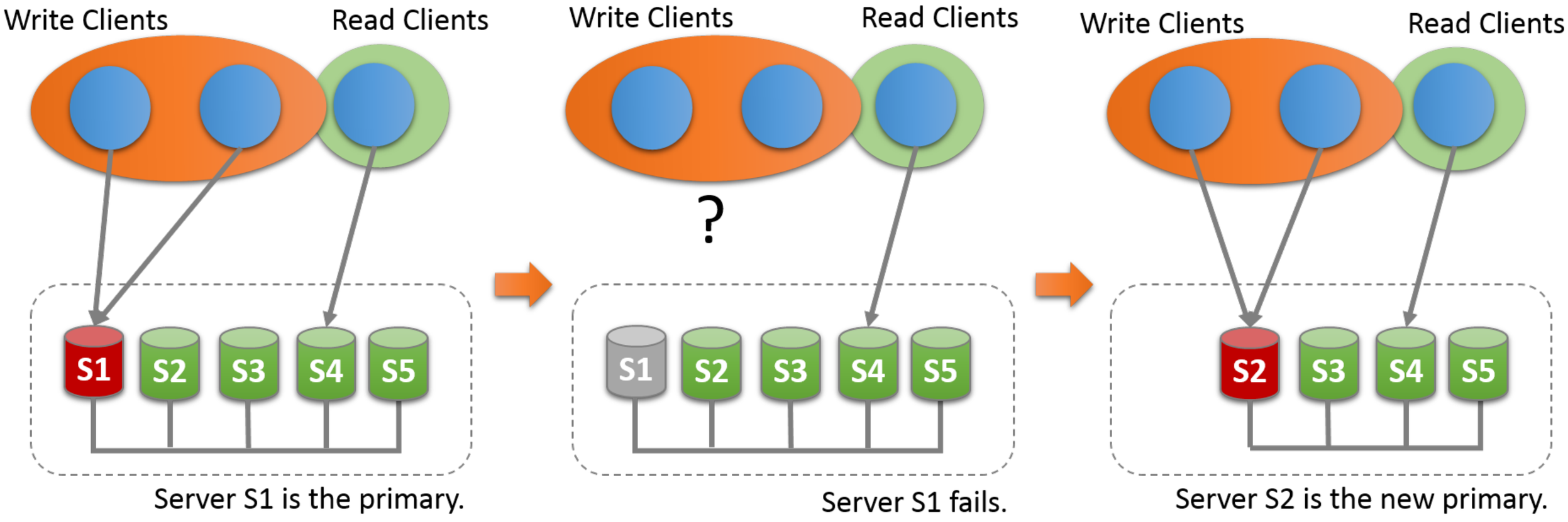
MGR

- Capture，跟踪本节点的事务
- Applier，执行远程事务
- Recovery，负责故障恢复时，选择donar节点，catch up binlog等
- Replication Protocol Logics，消息封装、接收XCOM返回的消息、发送本节点消息给XCOM、冲突检测等
- GCE: GCS的具体实现Xcom (eXtended COMmunications)



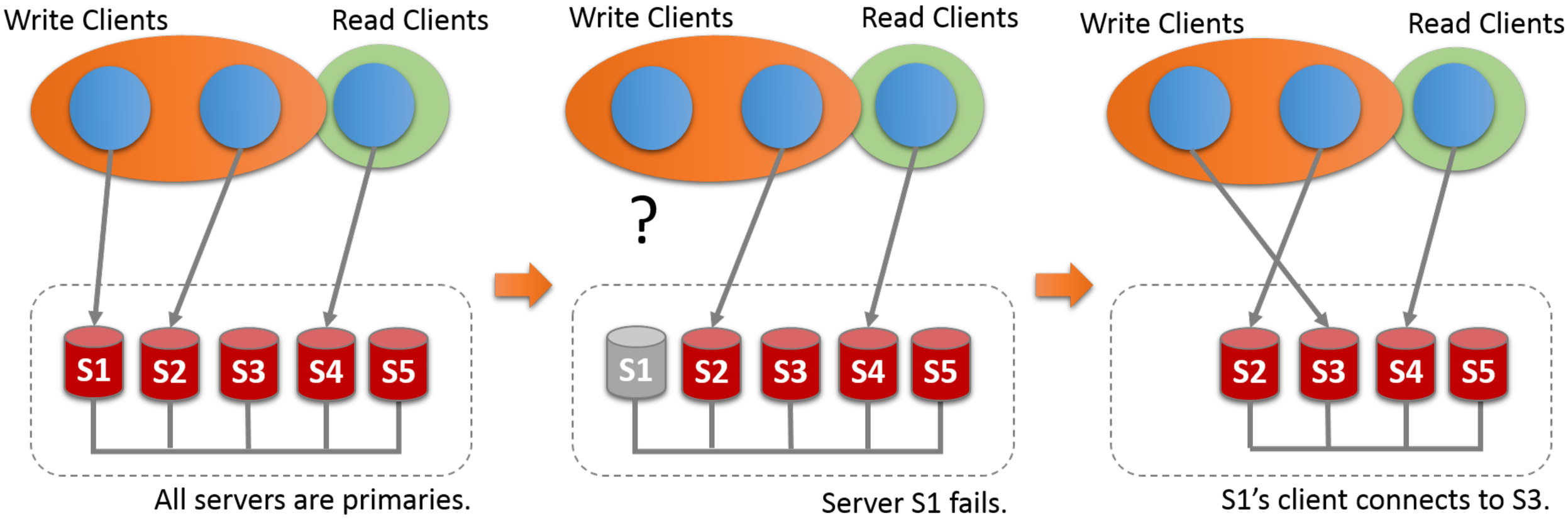
MGR

- single primary



MGR

- multi primary



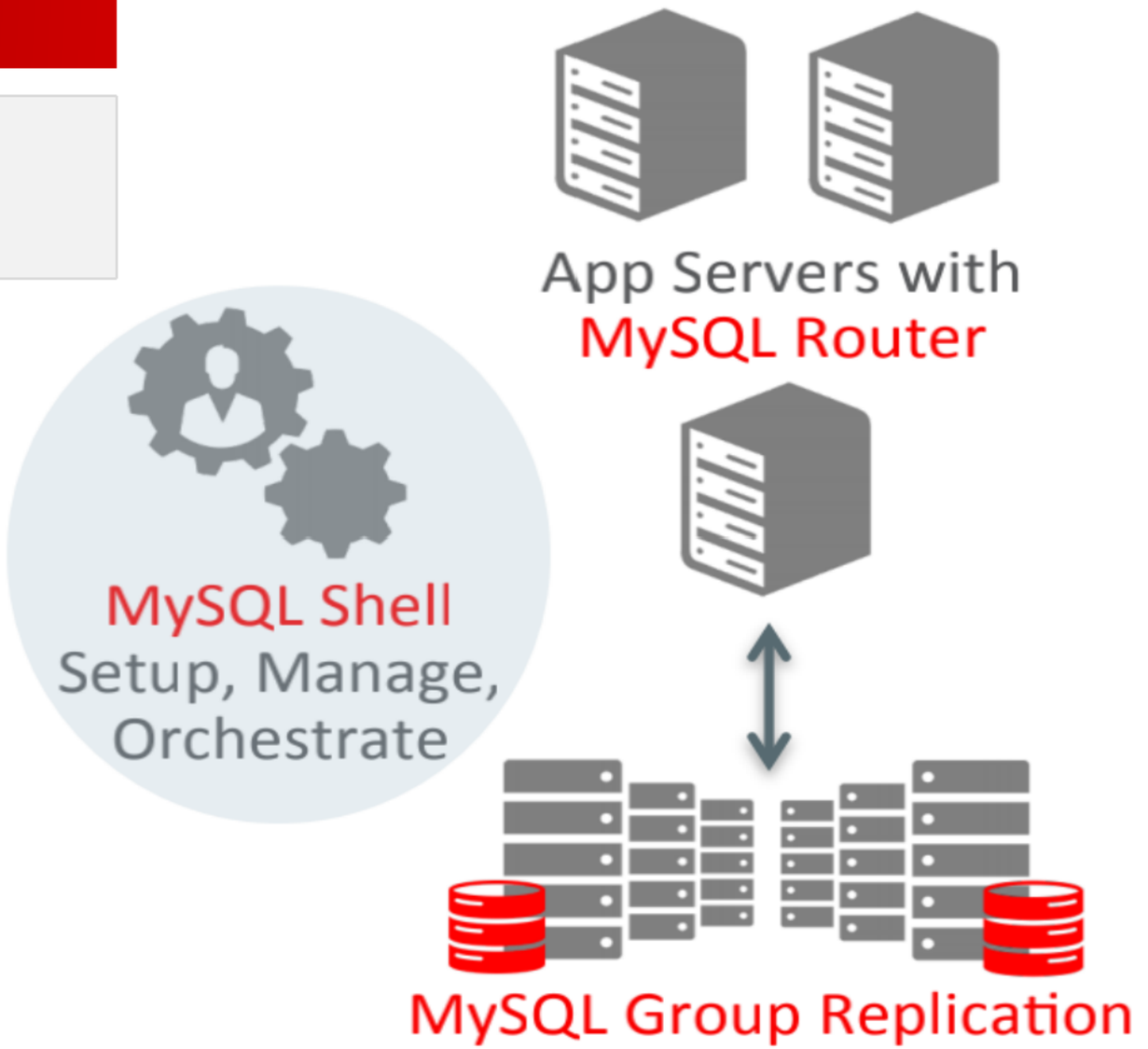
MGR

- 类比产品
  - MariaDB Galera Cluster
  - Percona XtraDB Cluster (PXC)
- 关键差异点
  - Group Communication System
  - Binlogs & Gcache
  - Node Provisioning
  - Partition
  - Flow Control
  - Cross plateform
  - DDL

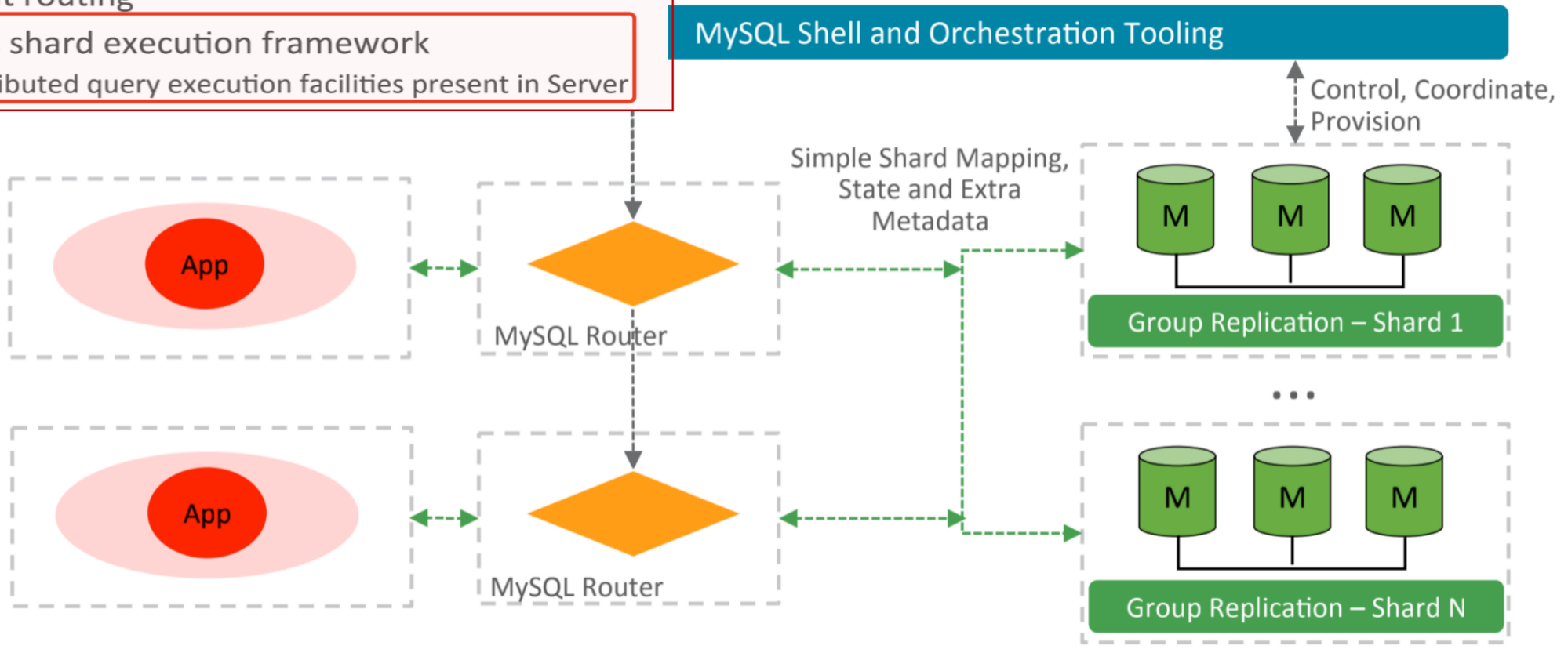
MIC及架构方案

InnoDB Cluster解决方案

- MySQL Router：路由客户端连接
- MySQL Shell: 集群搭建、管理工具

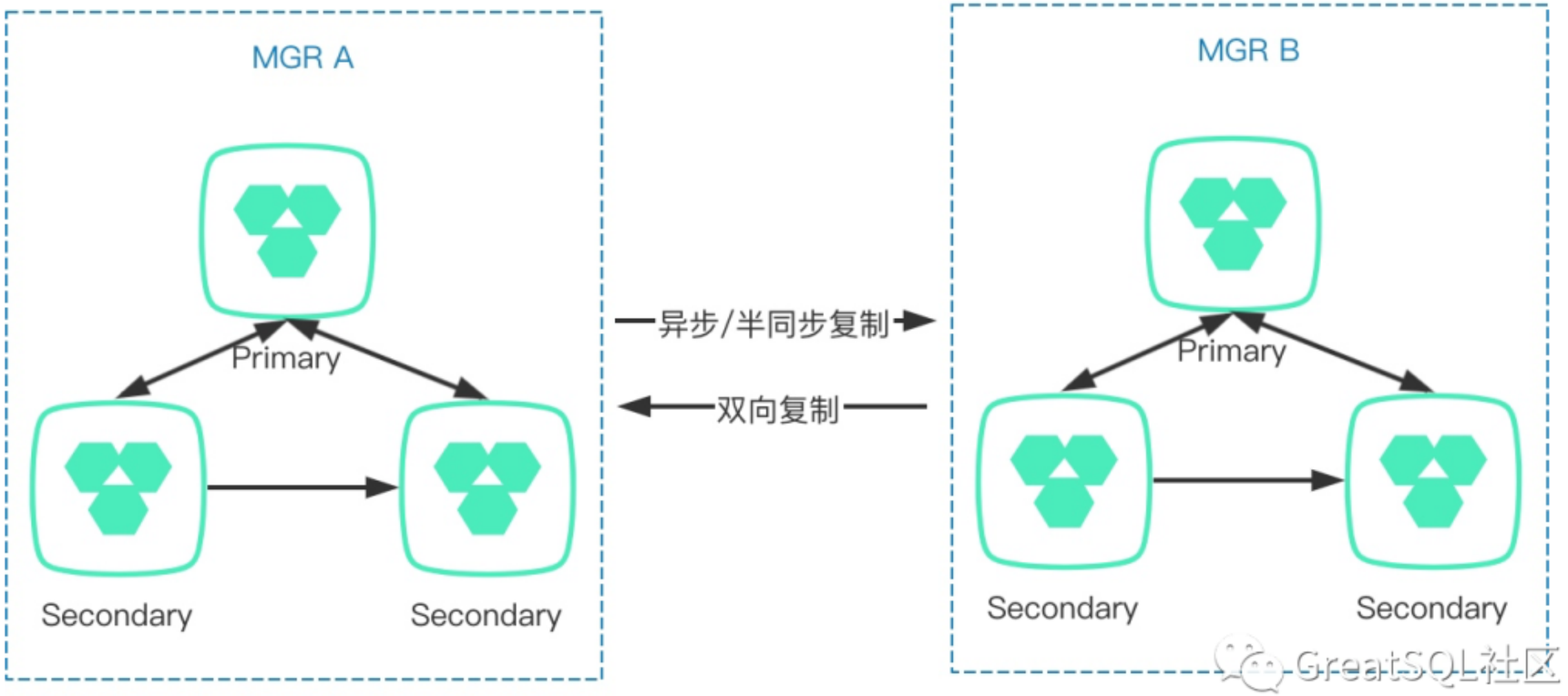


- MySQL Router
  - Manages shard mappings and related metadata
  - Manages client routing
  - Provides cross shard execution framework
    - On top of distributed query execution facilities present in Server



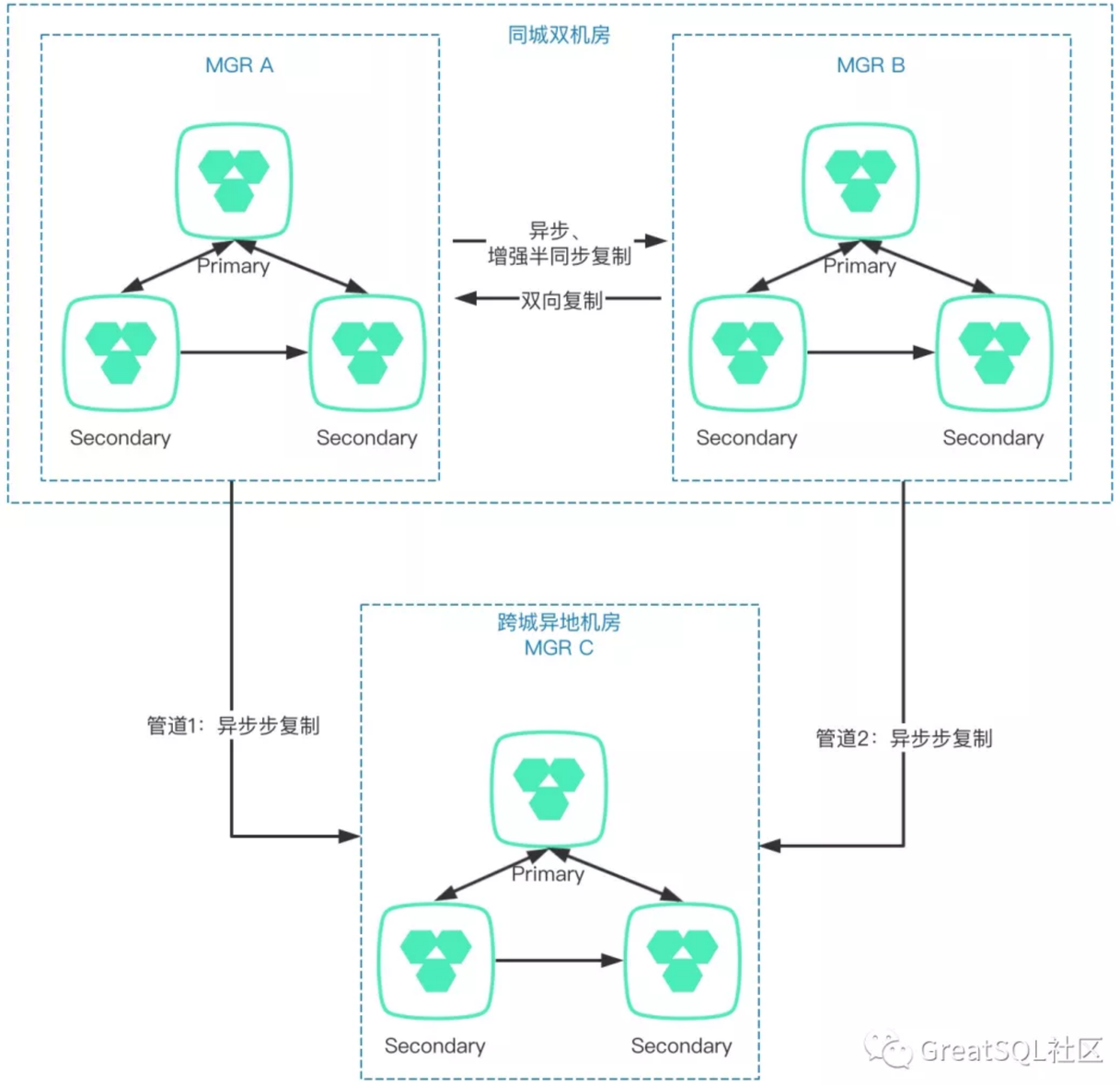
MIC

- 同城跨IDC架构



MIC

- 跨城多IDC架构



MGR最佳实践

MGR最佳实践

- 奇数节点（最多9个）
- 低延迟网络，避免WAN部署
- 单主模式
- 表要有主键
- InnoDB引擎
- 不要用到外键

MGR最佳实践

- log_error_verbosity=3
  - 默认为2，只输出Error，Warning
  - 3包含Information，记录更多日志，便于问题排查
- group_replication_bootstrap_group=OFF
  - 只在新集群初始化过程中打开
  - 集群引导完成后立刻关闭
  - 避免节点重启后，分裂出新集群

MGR最佳实践

- group_replication_transaction_size_limit=150000000
  - 单个事务大小上限
  - 尽量避免大事务出现
- group_replication_communication_max_message_size=10M
  - 将大事务切分成小包，进行paxos传递

MGR最佳实践

- group_replication_flow_control_mode=DISABLE | QUOTA
  - 流控开关
- group_replication_flow_control_certifier_threshold=25000
  - 触发流控的待认证的队列长度
- group_replication_flow_control_applier_threshold=25000
  - 触发流控的待执行的队列长度

MGR最佳实践

- binlog_transaction_dependency_tracking=WRITESET
  - 提升slave执行的并发度
- slave_checkpoint_period=2
  - BEFORE级别下，提升从库读性能
- slave_parallel_workers=CPU数*2
  - 合理配置，提升从库执行效率
- slave_parallel_type=LOGICAL_CLOCK
- slave_preserve_commit_order=ON

MGR最佳实践

- group_replication_consistency
- 如果只在PRIMARY节点读写
  - 默认使用EVENTUAL
  - 如果要求写事务在其它节点同时应用，则使用AFTER
- 如果多节点读负载均衡，且不希望读到旧数据
  - 大量写入场景，使用BEFORE
  - 少量写入场景，使用AFTER
  - 指定特定需要的事务使用BEFORE

MGR最佳实践

Great DB 万里数据库

- group_replication_unreachable_majority_timeout=10
  - 网络分区时，少数派等待此时长后，状态变为Error，回滚pending事务
- group_replication_autorejoin_tries=3
  - 自动尝试连入集群的次数，尝试间隔5min
- group_replication_exit_state_action=READ_ONLY
  - 退出集群后，server的状态设置
  - 配合自己的路由软件进行合理设置
- group_replication_member_expel_timeout=5
  - 将suspicious节点踢出集群的等待时长
  - 如果网络环境一般，可以适当调大30-60，不要太大

MGR最佳实践

- 不同版本不要混用
- SELECT … FOR UPDATE会导致死锁
- 不支持串行隔离级别
- 对同一个表DDL和DML不要在不同节点进行
- 不要跑大事务

MGR最佳实践

```
[root@GreatSQL][(none)]> select MEMBER_ID as id, COUNT_TRANSACTIONS_IN_QUEUE as trx_que,
COUNT_TRANSACTIONS_REMOTE_IN_APPLIER_QUEUE as app_que, COUNT_TRANSACTIONS_CHECKED as chkd,
COUNT_TRANSACTIONS_REMOTE_APPLIED as apped, COUNT_TRANSACTIONS_LOCAL_PROPOSED as proposed from
performance_schema.replication_group_member_stats;
+--------------------------------------+---------+---------+-----------+-----------+-----------+
| id                                   | trx_que | app_que | chkd      | apped     | proposed  |
+--------------------------------------+---------+---------+-----------+-----------+-----------+
| 4d10a9db-e53e-11eb-aeaf-525400fb993a |       0 |     326 | 459419476 | 459419154 |         0 |
| 5f031f98-e53e-11eb-bd31-525400e802e2 |       0 |       7 | 459290196 | 459290198 |         0 |
| ab884cc1-e095-11eb-876c-525400e2078a |       1 |       0 | 459673911 |         2 | 459673915 |
+--------------------------------------+---------+---------+-----------+-----------+-----------+
```

等待冲突检测队列

等待apply队列

MGR最佳实践

```
[root@GreatSQL][(none)]> select RECEIVED_TRANSACTION_SET as gtid_lag from
performance_schema.replication_connection_status
  where channel_name = 'group_replication_applier'
  union all
  select variable_value from performance_schema.global_variables
  where variable_name = ‘gtid_executed’;

+-----------------------------------------------------------+
| gtid_lag                                                  |
+-----------------------------------------------------------+
| aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaa1:1-612197303   | 已获取的事务GTID
| aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaa1:1-612196721   | 本地已apply的事务GTID
+-----------------------------------------------------------+
```

Great DB
万里数据库

不忘DB初心，牢记万里使命

谢谢