阿里云

MySQL高可用

MGR8.0 | 最佳实践

张彦东

MySQL DBA

MySQL爱好者，热衷于数据库及其周边技术分享。

沃趣科技简介

阿里云

WOQU TECH 沃趣

- **杭州沃趣科技股份有限公司**成立于2012年，专注在数据库生态产品的研发，结合云计算、 x86平台及国产化的发展趋势，为用户提供关系型数据库完整生命周期的云平台解决方案。

- 沃趣科技把**“让客户用上最好的数据库技术”**作为使命，期望成为国内市场份额第一的数据库生态产品公司。

Let Data Drive!

MGR特性

集群架构

数据同步原理

适用场景

高可用方案

01 MGR特性

MGR是什么

- 使用的协议
- 节点状态一致性保证
- 如何保证数据一致性

MGR是什么？

阿里云

通讯协议

基于Paxos算法的GCS原子广播协议，保证了一条事务在集群内要么在全部节点上提交或者回滚。

组成员资格

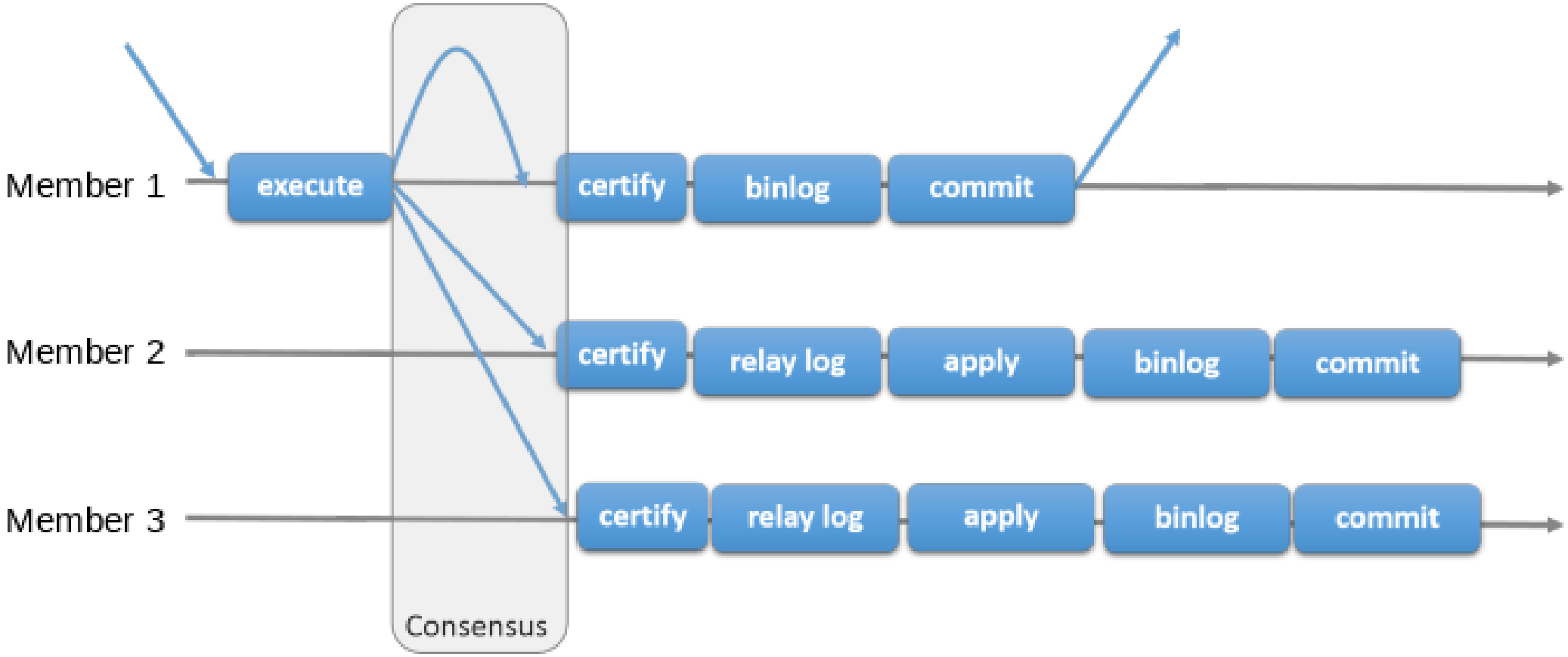
MGR内部提供一个视图服务，集群节点之间相互交换各自的视图信息，从而且实现集群整体的稳态。

数据一致性

MGR内部实现了一套不同事务之间修改数据的冲突认证检测机制。

MGR是具备强大的分布式协调能力，可用于创建弹性、高可用性、高容错的复制拓扑的一个MySQL插件。

示例


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

02 集群架构

详细架构

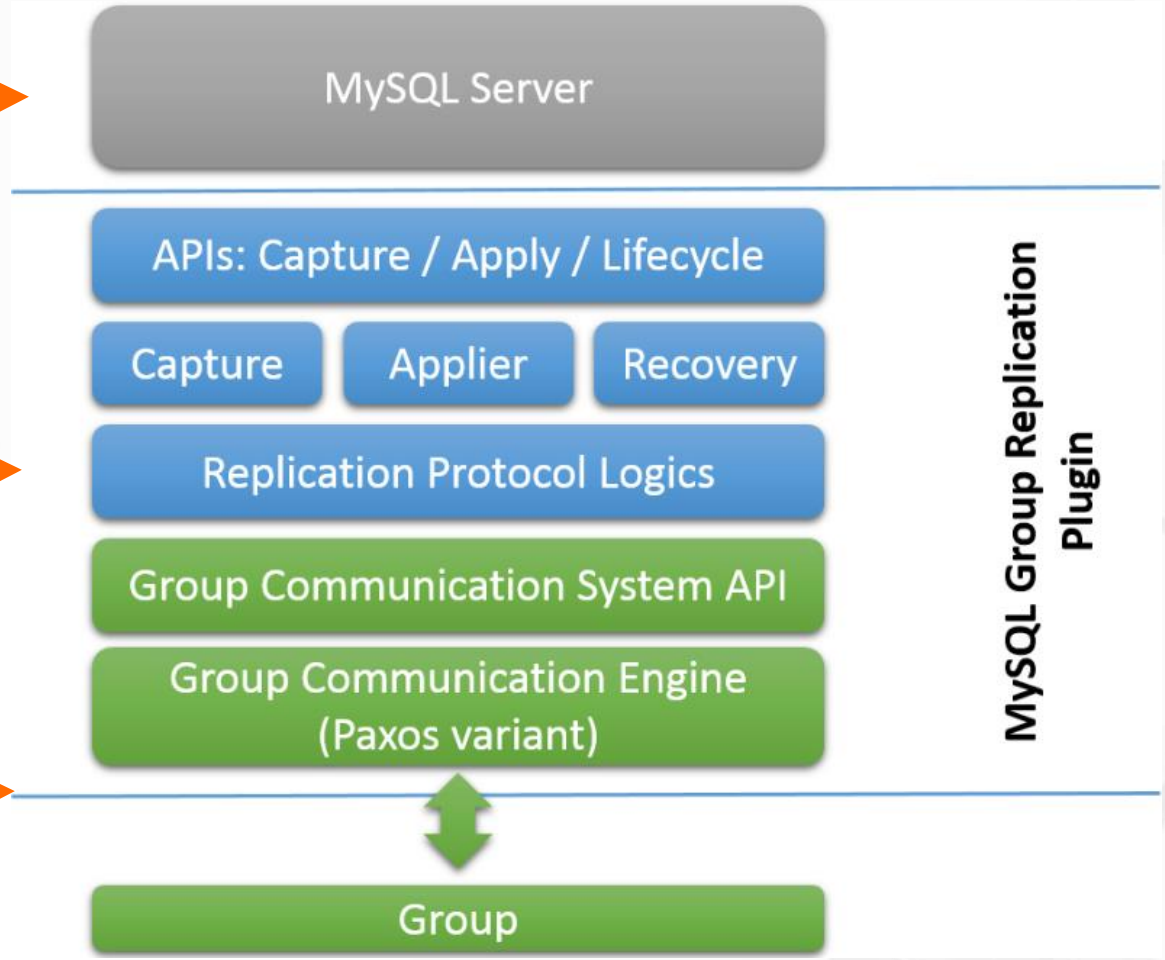
- 单主模式
- 多主模式

MGR插件组成

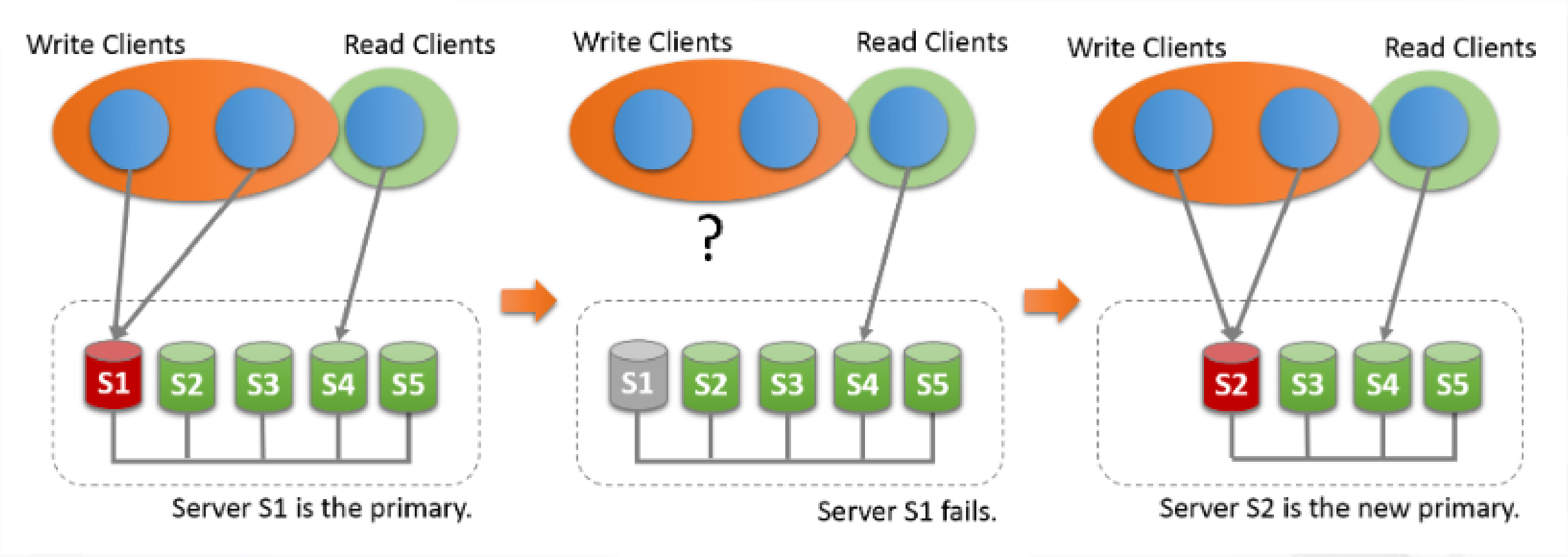
基于MySQL现有的主从复制，利用row格式的binlog和gtid等功能实现的集群架构。

基于 server交互的接口集，在逻辑上将MySQL内核与MGR插件隔绝开来。

组通信系统，基于paxos协议实现的组通信引擎，主要提供数据一致性核心功能。



单主模式


Write Clients
Read Clients
S1
S2
S3
S4
S5
Server S1 is the primary.
Write Clients
Read Clients
?
S1
S2
S3
S4
S5
Server S1 fails.
Write Clients
Read Clients
S2
S3
S4
S5
Server S2 is the new primary.

集群原理

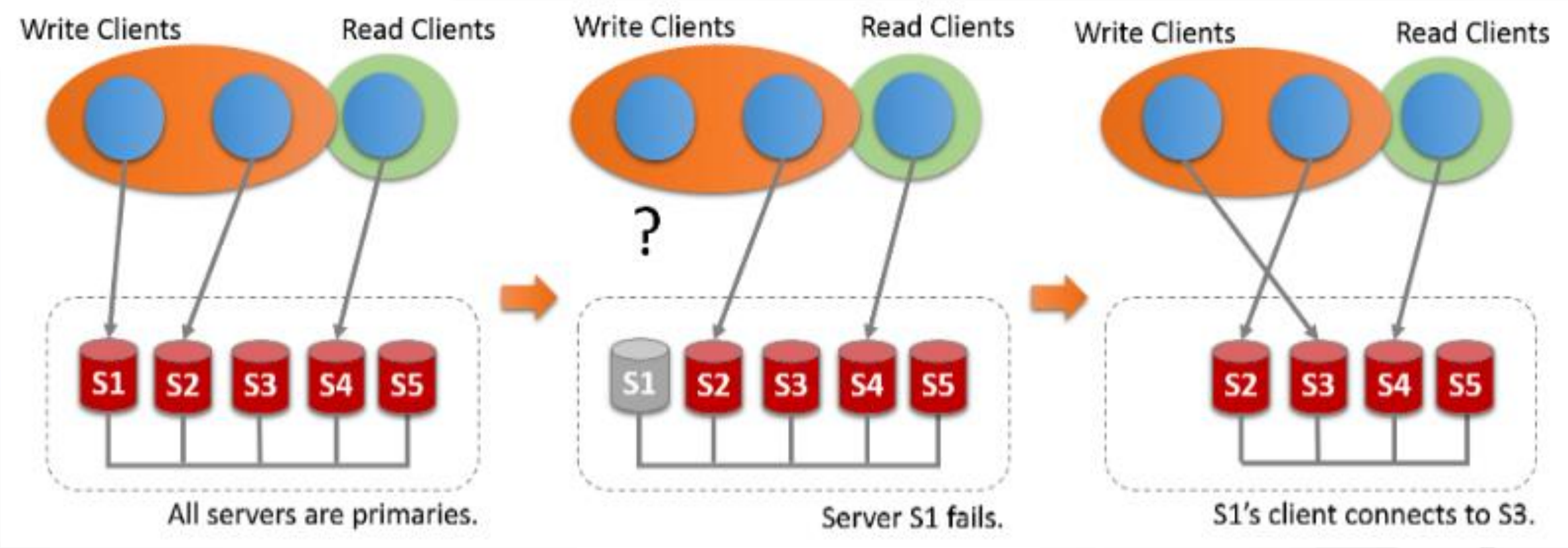
在**单主模式下**（group_replication_single_primary_mode = ON）：

- 该变量在所有组成员中必须设置为相同的值。

- 该集群具有一个设置为读写模式的主节点。组中的所有其他成员都设置为只读模式（super-read-only = ON）。

- 读写节点通常是引导该组的第一个节点。加入该集群的所有其他只读节点均需要从读写节点同步数据，并自动设置为只读模式。

多主模式


Write Clients
Read Clients
S1
S2
S3
S4
S5
All servers are primaries.
Write Clients
Read Clients
?
S1
S2
S3
S4
S5
Server S1 fails.
Write Clients
Read Clients
S2
S3
S4
S5
S1's client connects to S3.

集群原理

在**多主模式下**（group_replication_single_primary_mode = OFF）：

- 所有节点不会区分primary和standby角色。

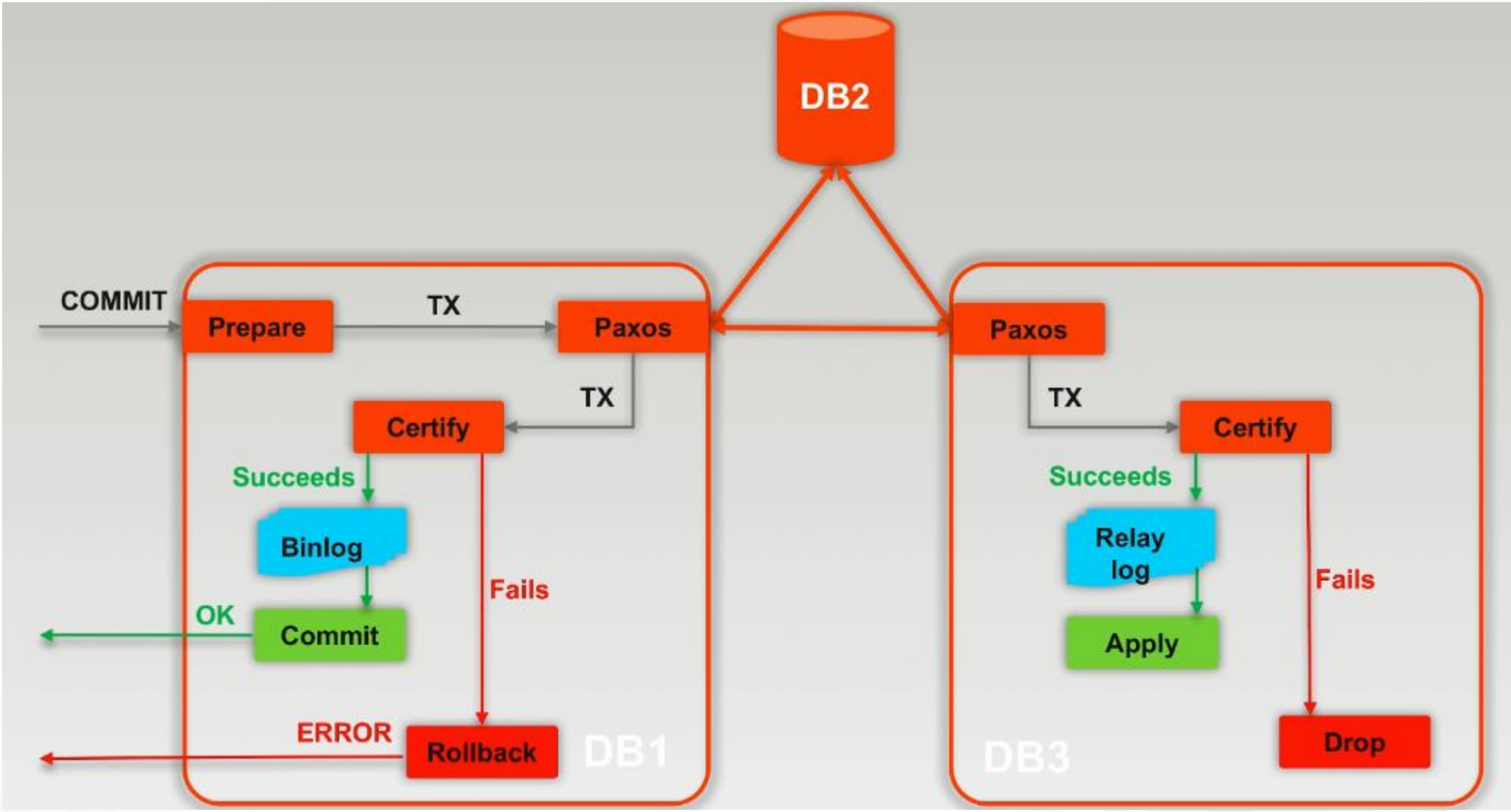
- 加入该集群时，与其他组成员兼容的任何节点都被设置为读写模式，并且可以处理写请求，即使它们在集群内是并发执行的。

- 如果组复制中的某个节点停止接受写事务，例如，在某个节点意外宕机的情况下，可以将与其连接的客户端重定向或故障转移到处于读写模式的任何其他健康的节点。

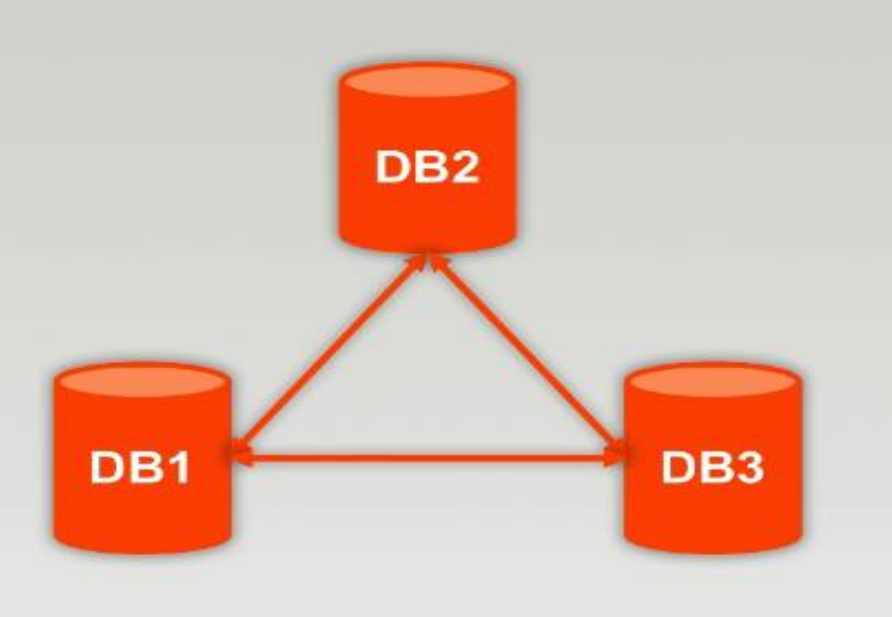
- 组复制本身不处理客户端故障转移，因此需要使用中间件框架（例如MySQL Router 8.0，代理，连接器或应用程序本身）来实现。

03 数据同步原理

原理示意图


DB2
COMMIT
Prepare
TX
Paxos
TX
Certify
Succeeds
Binlog
Fails
OK
Commit
ERROR
Rollback
DB1
Paxos
TX
Certify
Succeeds
Relay log
Apply
Fails
Drop
DB3

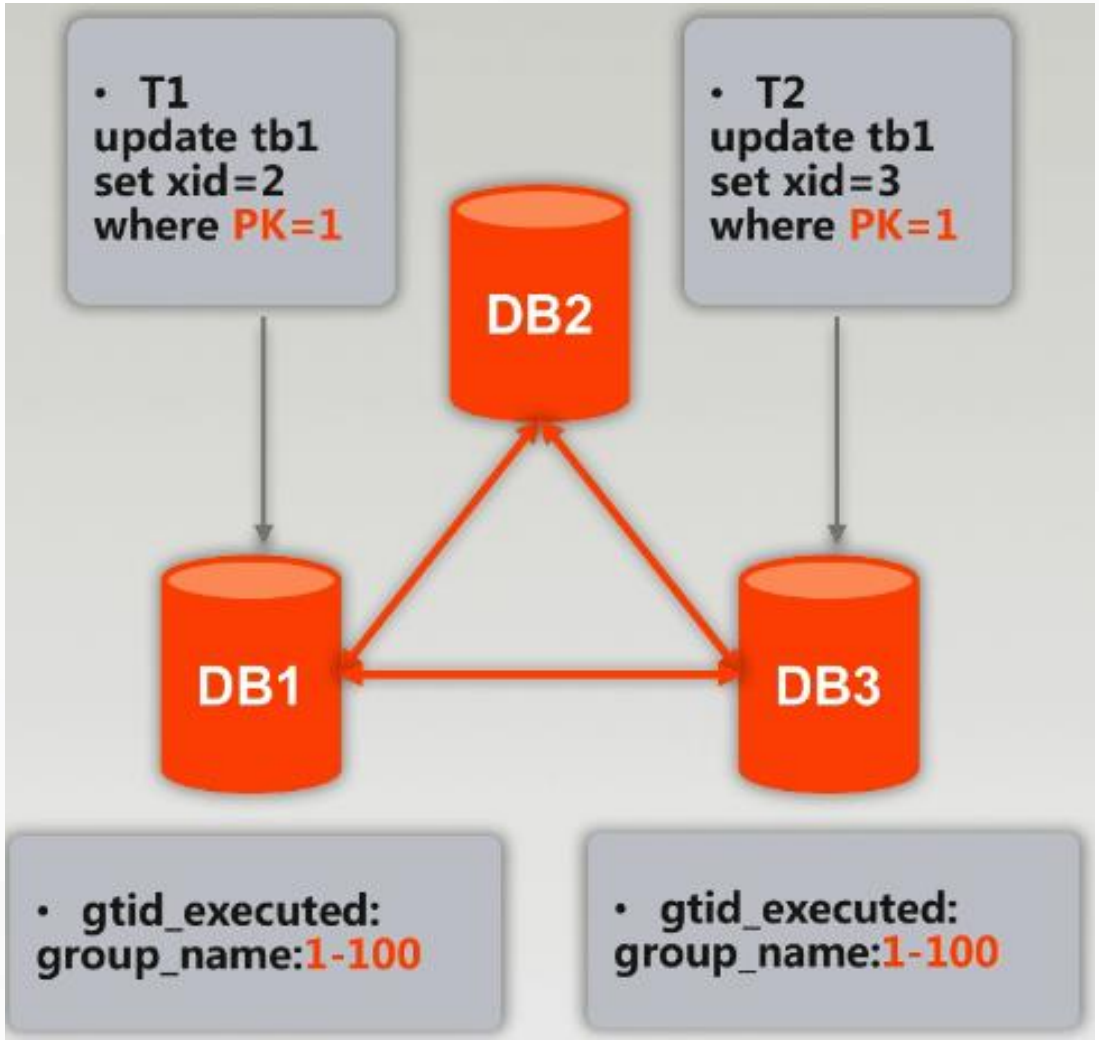
冲突检测原理



PK HASH	GTID SET
db1.tb1.pk=1	group_name:1-10
db1.tb1.pk=2	group_name:1-15
.....	.....

- 每个事务的gtid set和对应的主键hash值组成事务认证列表。
- **gtid set:**标记数据库的快照版本，事务提交前从gtid_execute变量中获取该值。
- 事务提交前，数据库中执行了的gtid集合,随着binlog中的event通过原子广播的方式分发到集群的所有节点上进行事务冲突检测。

冲突检测示例



T1:update tb1 set xid=1 where pk=1(snapshot:group_name:1-100)

T2: update tb1 set xid=2 where pk=1(snapshot:group_name:1-100)

certify

冲突检测说明

- 若T2修改的数据在冲突检测数据库中无匹配记录，则判定为通过冲突检测认证。
- 若T2中的gtid set包含了冲突检测数据库中相同主键值的gtid set，则冲突认证检测通过。
- 冲突认证检测通过后，每个节点的冲突检测数据库按照如下规则变更：

1. 若在冲突检测数据库中无匹配记录，则向其中插入一条新的记录。
2. 如果有记录，则更新冲突检测数据库中的gtid set值。

success

Let Data Drive!

冲突检测说明

T1:update tb1 set xid=1 where pk=1(snapshot:group_name:1-100)

certify

若T1修改的数据在冲突检测数据库中有匹配到记录，且T1的gtid set不包括冲突检测数据库中的gtid set，则判定为冲突检测不通过。

注意：冲突检测不通过时，不会更新认证数据库里的gtid set。

04 性能分析

存在的问题

- MGR可确保仅在集群中的大多数节点都已收到事务并就并发发送的所有事务之间的相对顺序达成一致后，才提交事务。
- 在流量小的时候不存在任何的性能问题。当流量突增时，如果集群中某些节点的写入吞吐量比其他节点少，尤其是小于primary节点，则这些节点的数据和primary节点的数据存在偏差。
- 在单主模式的集群中，如果发生故障转移，在新的primary节点可以立刻接受写入请求的情况下，则存在集群内事务一致性的问题。

存在的问题

当集群扩容时，例如由3节点集群扩容到5节点集群：

- 无论采用clone的方式还是用xtrabackup做全量数据恢复后，都避免不了在集群扩容期间产生的增量数据以二进制日志的方式来追平。

- 若新扩容的节点配置较低，写入吞吐差，则新加入的节点很有可能一直处于recover的状态，该节点很难达到online的状态。

事务一致性保证

阿里云

group_replication_consistency：

- **EVENTUAL：**

开启该级别的事务（T2），事务执行前不会等待先序事务（T1）的回放完成，也不会影响后序事务等待该事务回放完成。

- **before：**

开启了该级别的事务（T2），在开始前首先要等待先序事务（T1）的回放完成，确保此事务将在最新的数据上执行。

- **AFTER：**

开启该级别的事务（T1），只有等该事务回放完成。其他后序事务（T2）才开始执行，这样所有后序事务都会读取包含其更改的数据库状态，而不管它们在哪个节点上执行。

事务一致性保证

- **BEFORE_AND_AFTER:**

- 开启该级别等事务（T2），需要等待前序事务的回放完成（T1）；
- 同时后序事务（T3）等待该事务的回放完成

- **BEFORE_ON_PRIMARY_FAILOVER:**

- 在发生切换时，连到新主的事务会被阻塞，等待先序提交的事务回放完成；
- 这样确保在故障切换时客户端都能读取到主服务器上的最新数据，保证了一致性。

流控机制

阿里云

流控机制试图解决以下问题：

1. 集群内节点之间不会相差太多的事务；
2. 快速适应集群内不断变化的负载，例如集群内的写压力暴增或集群中增加更多的节点；
3. 均分可用写容量到集群内的所有节点上；
4. 避免减少吞吐量而造成资源浪费。

两个基本机制

1. 监控集群内所有节点以收集有关吞吐量和队列大小的一些统计信息，以便对每个节点能承受的最大写入压力进行有根据的评估；

2. 对集群中的所有节点的并发写能力时刻保持监控，一旦某节点的并发压力超过了集群中所有节点的平均写能力，就会对其执行流量控制。

两个基本队列

认证队列和二进制日志应用队列，当其中一个队列的大小超过用户定义的阈值时，就会触发流量控制机制。

对于流量控制配置：

首先，需要选择对谁配置流量控制，是对认证队列、还是针对应用队列、还是两者都需要配置流量控制。然后，对需要配置流量控制的对象（认证队列和应用队列）设置流量控制阈值。

流控过程

- 将根据上一阶段延迟的事务数量逐步减少10%，让触发限流机制的队列减小到限流机制被触发的阈值之内，待到恢复后，为避免在队列大小超过阈值时出现吞吐量的陡增，在此之后，每个时间段的吞吐量只允许增长相同的10%。

- 当前的限流机制不会影响到触发限流机制阈值内的事务，但是会延迟应用超过阈值的事务，直到本监控周期结束。

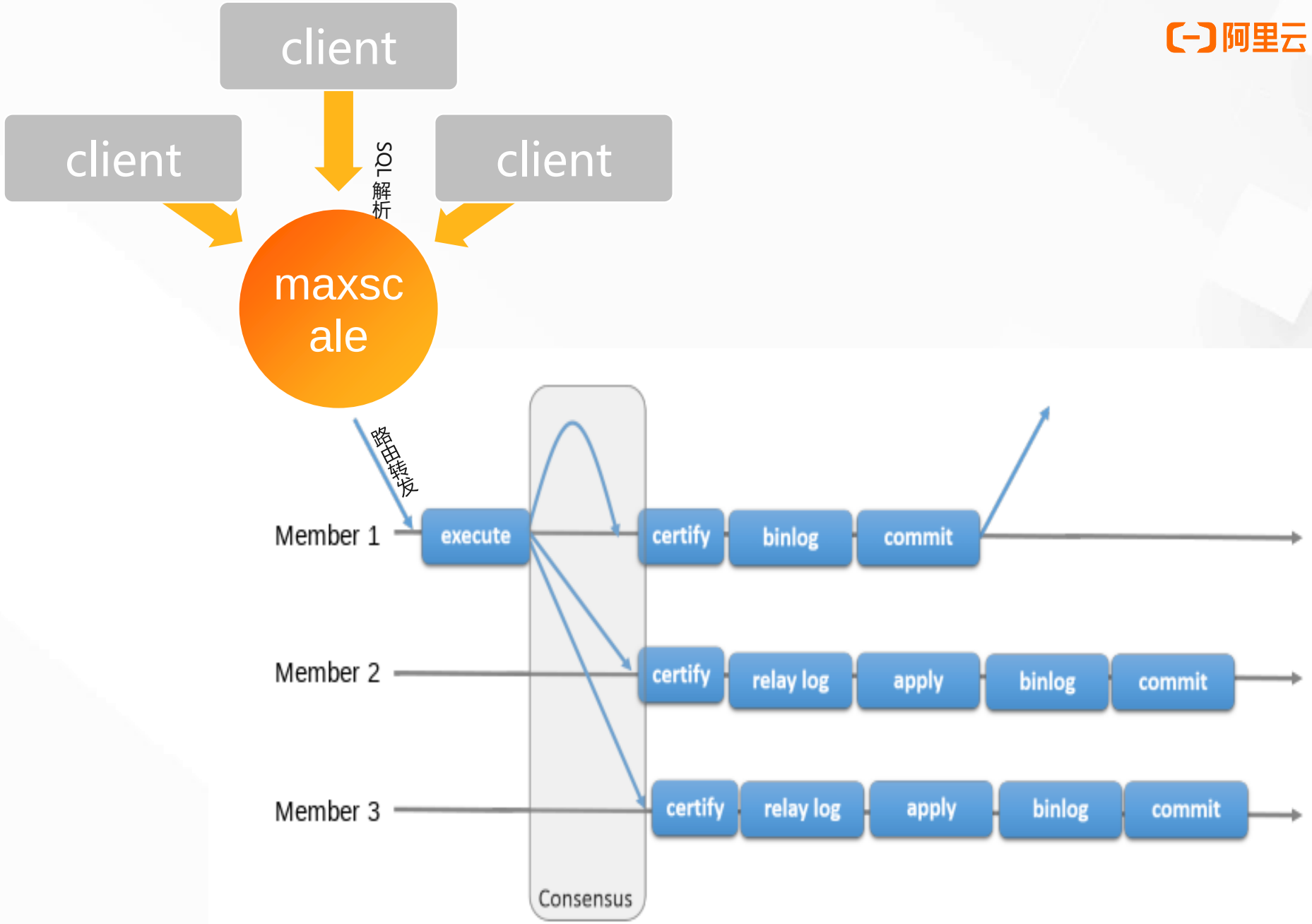
- 如果触发节流机制的阈值设置的非常小，部分事务的延迟可能会接近一个完整的监控周期。

05 适用场景

MGR的一些典型使用场景：

- **弹性复制：**
- 需要非常灵活的复制基础设施的环境，其中MySQL Server的数量必须动态增加或减少，并且在增加或减少Server的过程中，对业务的副作用尽可能少。例如，云数据库服务。
- **高可用分片：**
- 分片是实现写扩展的一种流行方法。基于MGR实现的高可用分片，其中每个分片都会映射到一个复制组上（逻辑上需要一一对应，但在物理上一个复制组可以承载多个分片）。
- **替代主从复制：**
- 在某些情况下，使用一个主库会造成单点争用。在某些情况下，向整个组内的多个成员同时写入数据，对应用来说可能伸缩性更强。

06 高可用方案


client
client
client
SQL
解析
maxsc
ale
路由转发
Member 1
execute
certify
binlog
commit
Member 2
certify
relay log
apply
binlog
commit
Member 3
certify
relay log
apply
binlog
commit
Consensus

阿里云

感谢聆听