# 背景

MySQL Group Replication（下简称：MGR）是MySQL官方推出的一种基于Paxos协议的状态机复制。在MGR出现之前，用户常见的MySQL高可用方式，无论怎么变化架构，本质就是Master-Slave架构。MySQL 5.7版本开始支持无损半同步复制（lossless semi-sync replication），从而进一步提示数据复制的强一致性。

Master-Slave始终无法解决的一个问题是选主（Leader Election），特别是当由于网络分区发生脑裂时，目前大多的高可用解决方案都会导致双写的问题，这在金融场景下显然是无法接受的。为避免此问题的发生，有时不得不强行关闭一台服务，从而保证同一时间只有一个节点可以写入，然而这时数据库集群可用性又可能会收到极大的影响。

MongoDB、TiDB这些出现的后起之后通过Raft协议来进行选主，从而避免脑裂问题的产生。然而他们依然是单写的场景，即一个高可用集群下，写入依然是单个节点。

# 概述

MGR的解决方案现在来看可说非常完美：

1. 数据一致性保障：确保集群中大部分节点收到日志
2. 多节点写入支持：多写模式下支持集群中的所有节点都可以写入
3. Fault Tolerance: 确保系统发生故障（包括脑裂）依然可用，双写对系统无影响

从MGR公布的特性来基本是数据库所追求的最终完美形式。然而很多同学还会问MGR和无损半同步复制的区别，比如1个集群5个节点，无损半同步复制可以设置至少有2个节点收到ACK请求再提交事务，也能保障数据一致性。

Quorum原则（大部分原则）只是Paxos实现的一小部分，因此无损半同步复制解决的只是日志完整性的问题。若把日志看成是value，则只是解决了日志丢失问题。但是如何在分布式异常场景下确定这个value值，则需要Paxos协议来解决。比如，当发生脑裂情况下，谁是Primary，则MGR通过Paxos协议可以清楚的判断，从而避免双写问题。

MGR默认是Single Primary模式，用户可以通过下面的命令找出集群中的Primary（Leader）节点，这在传统的复制架构下并不能通过数据库本身感知：

mysql> SELECT VARIABLE\_VALUE FROM performance\_schema.global\_status WHERE VARIABLE\_NAME= 'group\_replication\_primary\_member';

MGR另一个模式为Multi-Primary模式。对Paxos协议比较熟悉的同学会说Paxos本身就支持多写的。的确，然而为了解决活锁的性能问题，Paxos的实现大多是Single Leader模式。很明显Leader节点会成为一个瓶颈，所以MGR并没有使用传统的Multi-Paxos算法，而是使用了Mencius算法，解决了单Leader写入性能瓶颈问题。

通过Mencius算法进一步优化Paxos的多写场景是可能的，然而数据库的复杂之处在于写入的数据之间可能是有依赖冲突的。比如节点1、节点2都在对记录1进行更新，在Multi-Primary模式MGR会开启冲突检测机制，并遵循Commit First原则。

最后来看看MGR真正的伟大之处：

1. 采用更先进的Mencius算法，而非传统的Multi-Paxos算法
2. 支持节点多写，并通过冲突检测机制保证同时写入的数据之间不会冲突

# 原理