组复制背景：初识MGR---基础篇

1.MGR特性：

在MySQL 8.0中，MGR在可控性、稳定性等方面具备了非常优异的特性，最重要的是其性能也有大幅提升。

本文基于MySQL 8.0.21版本和官方文档进行整理，同时也参考了业界一些技术大牛的个人博客上的一些有独到见解的文章。从特性、集群架构、状态监控，流控，性能分析等方面详细介绍了MGR。

MGR是通过插件实现的，不是MySQL原生内置的，可用于创建弹性、高可用性、高容错的复制拓扑。

在单主模式下，只有一个节点能够接受更新操作，可以实现自动选主。也可以运行在多主模式下，多主模式下所有节点都能接受更新操作，即使多个节点同时发起更新请求， MGR的冲突检测机制也能够保证它们正常执行。

MGR有一个内置的服务，使得在任何给定的时间点集群内的视图能够保持一致，并为集群内的所有成员提供服务。给定Server可以根据需要使其脱离或加入集群内，集群内的视图也会随着组成员的脱离、加入进行相应更新。有时，Server可能会意外脱离集群，在这种情况下，故障检测机制会自动检测到并通知集群该视图已更改。这些变更操作都是自动的，无需人工操作。

2.概述：

传统意义上的创建容错系统的最常见方法是使用组件冗余，换句话说，就是搭建多个备库，因为各种原因，一个备库宕掉后，系统可以继续按预期运行。

这种做法同时也带来了许多的问题，其中最尖锐的就是系统复杂度非常高。具体来说，需要维护和管理多台服务器，而不仅仅是一台服务器。此外，随着业务数据的不断增加，集群内的服务器数量也越来越多，架构也越来越复杂。伴随而来的就是经典的分布式系统问题，例如脑裂这种情况。

因此，最终需要解决的问题就是如何要让多个服务器就系统的状态以及系统所经历的每一次数据变更在集群间达成一致。可以将其概括为使服务器上的每个MySQL实例在状态转换上都达成统一协议，从而使它们看起来像是作为一个整体在运行，最终的结果就是需要集群间的不同成员对外展示的信息在最终达到一个一致的状态。这意味着它们需要以（分布式）状态机来运行。

MySQL组复制为这种分布式状态机服务之间的数据复制提供了强大的协调能力。在同一个集群中的MySQL server，它们会自动进行协调。该集群可以在具有自动选主的单主模式下运行，其中仅一个MySQL server接受写入更新。也可以在多主要模式下运行，在该模式下，所有MySQL server都可以接受写入更新，即使这些写入更新是同一时刻发生的。这种分布式协调能力，也可以使得这些并发请求能够正常执行，而不至于在集群内产生数据冲突。

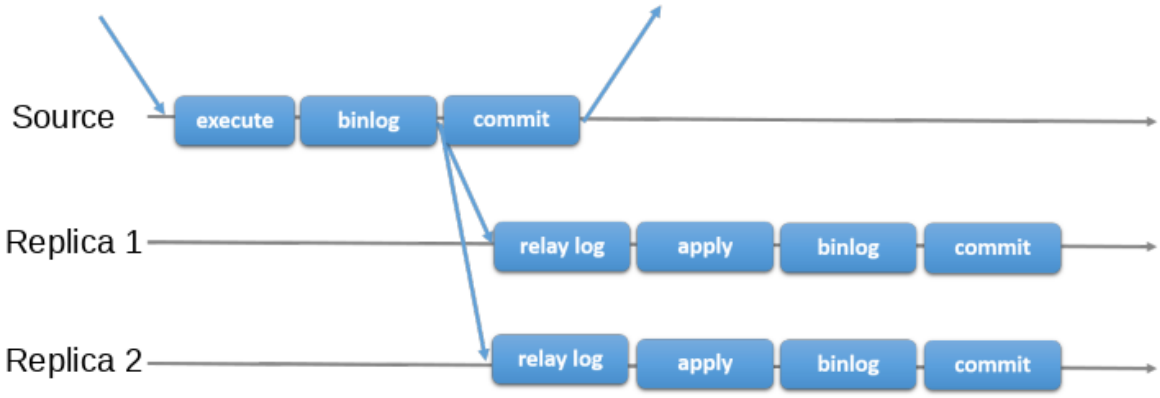
MySQL server可以离开或者加入集群，并且将相应的视图更新。有时MySQL server可能会意外被踢出集群，在这种情况下，故障检测机制会检测到此情况并通知集群视图已更改。这些都是自动的。 对于写入完成要提交的事务，不管是在单主还是多主模式下，集群的大多数成员必须在全局事务顺序中就给定事务的顺序达成一致。然后每个MySQL server自行决定提交或回滚事务，（集群中的每一个节点按照同样的顺序对需要写入的事务做冲突认证检测）但是同一个集群内的所有MySQL server最后做出决定后对外展示的结果都应该是一致的。

如果存在网络分区，导致集群内成员之间无法达成一个一致性的结果，那么在解决此问题之前，系统会一直处于一个阻塞的状态。因此，组复制还有一个内置的自动裂脑保护机制。通过这种内置的机制来防止发生网络分区时，可能导致数据不一致的问题。

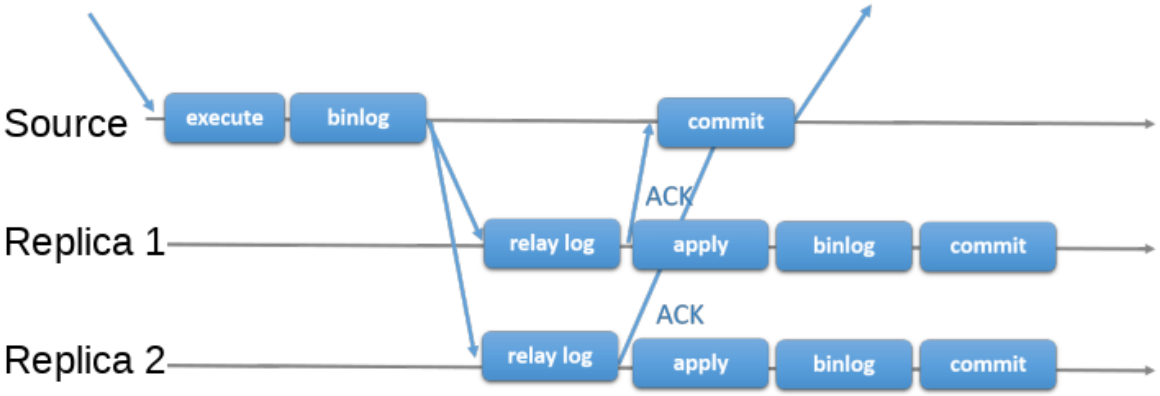
所有上面提到的这些特性都由（GCS）协议提供支持。GCS协议提供故障检测机制，组成员之间数据的一致性以及安全且完全有序的消息传递。所有这些属性对于在一个可以实现分布式数据复制的集群来说是至关重要的，Paxos算法的实现可确保在不同的组成员之间来实现分布式数据复制的一致性。

3.主从复制：

传统的MySQL复制使用一种简单的主从复制方法。master执行事务，提交事务，然后将它们稍后（异步）发送到slave以重新执行（在基于语句的复制中）或应用（在基于行的复制中）。默认情况下，所有server均具有数据的完整副本。



还有半同步复制，它向协议添加了一个数据确认步骤。这意味着主服务器在提交时等待备服务器确认已接收到事务。只有这样，主服务器才会继续提交操作。



在正式开始MGR单主模式之前，通过在上面的两张图片，带着大家简单回顾一下MySQL的异步复制和半同步复制。

4.组复制：

接下来，我们正式进入主题。组复制是一种可用于实施容错系统的技术。在组复制中，每个服务器都有自己的完整数据副本，通过集群内不同成员之间的消息传递相互交互。通信层利用原子广播来保证集群内状态信息在不同的成员之间达成一致。

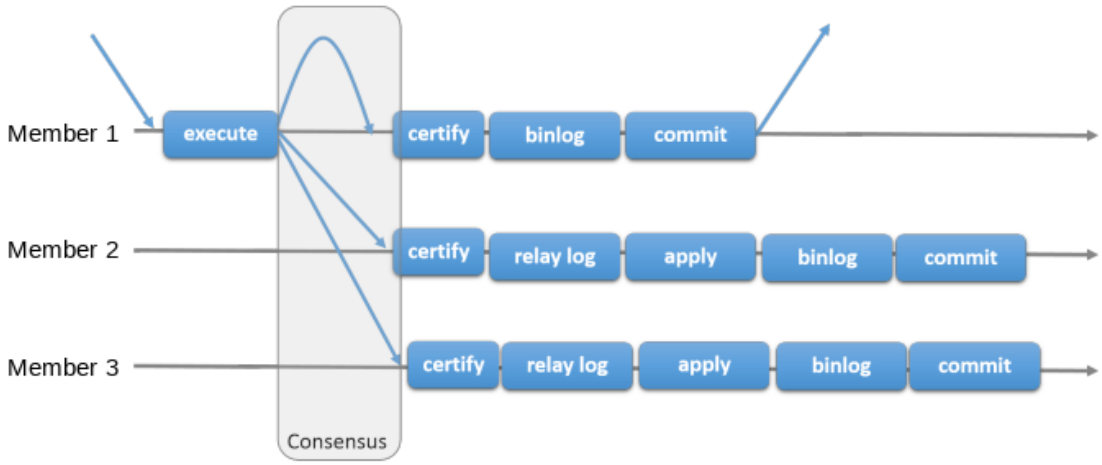
一个MGR集群由多个MySQL server组成，该集群中的每个MySQL server可以随时独立执行事务。但是需要注意的是，所有读写事务只有在经过集群内的冲突认证检测之后才能提交。换句话说，对于任何读写事务，能否成功提交是取决于集群内成员之间的相互协调。（具体的原理我们后面讨论，这里大家有个概念即可）因此提交操作是否成功不是发起写请求的那个MySQL server单方面可以决定的。只读事务无需集群内的任何协调即可立即提交。

当读写事务准备在集群内提交时，发起读写请求的当前MySQL server会自动广播已更改的行和相应的write set（已更新的行的唯一标识符）。由于事务是通过原子广播发送的，因此该组中的所有成员都将接收该事务，或者将接受到的事务全部丢弃。如果他们收到了，那么相对于之前发送的其他事务，他们都将以相同的顺序收到。因此，集群内的所有成员都以相同的顺序接收相同的数据集，并且为需要提交的数据集建立了全局队列。

然而，多主模式下，在同一集群内不同的MySQL server上同时执行的事务之间可能存在冲突。通过在冲突检测的过程中检查并比较两个不同并发事务的write set，可以检测到此类冲突，这个过程我们称之为事务冲突认证。在检测过程中，冲突检测是在行级别执行的：如果在不同服务器上执行的两个并发事务更新同一行，则存在冲突。冲突检测算法中决定了较早发起的事务会成功提交，后面发起的事务会终止且回滚，因此在该MySQL server上回滚并由组中的其他成员丢弃该事务。（丢弃接收到的write set）例如，如果两个事务t1和t2在不同的节点上同时执行，都更改了同一行，并且t2在t1之前排序，则t2成功提交，并且t1被回滚。请注意，如果两个事务之间的冲突经常发生，那么在同一个服务器上执行它们是一个好习惯，在那里，它们有机会在本地锁管理器上进行同步，而不必由由于冲突检测而回滚。

对于已经通过冲突检测的数据集，如果不破坏集群数据的一致性和有效性，MGR复制允许成员之间不按照应用程序的写入顺序提交数据。组复制是最终的一致性系统，这意味着只要是通过了冲突认证检测的事务，不要求在所有组成员的事务提交顺序都完全一致，只需要保证组复制内的所有成员最终具有相同的数据内容即可。例如，在多主模式下，尽管尚未应用全局顺序中较早的事务，但是本地事务可能会在通过冲突检测后立即被持久化。当确定不同的write set之间没有冲突时，这是允许的。在单主模式下，在可读写节点上，并发，无冲突的本地事务与组复制所定义的全局事务顺序以不同的顺序进行提交和持久化的是允许的。在不接受来自客户端的写操作的只读服务器上，事务始终按照集群内预先协商好的全局顺序进行持久化。

下图描述了MySQL组复制协议，通过将其与MySQL复制（甚至MySQL半同步复制）进行比较，可以看到一些区别。为了清楚起见，此图中缺少一些Paxos相关的内容。



5.MGR适用场景：

通过MGR，可以创建出一个具有冗余的容错系统。当集群内有成员出现故障时，只要不是全部或大多数组成员（组内超过半数的成员）出现故障，则系统仍然可用。根据故障节点数量的不同，集群可能会降低性能或可伸缩性，但它仍然可用。节点故障是独立的。它们由一个单独的集群服务进行跟踪监控，该服务依赖于一个分布式故障检测器，该检测器能够在任意节点脱离集群时发出信号（无论是出于自愿还

是由于意外停机，都会发出这样的信号）。当故障节点恢复正常或有新的MySQL Server加入集群时，由一个恢复程序来控制整个过程，以确保当Server加入集群时，它们会自动更新组成员视图等相关信息。而且多主更新的特性可确保即使在单个节点发生故障时，也不会阻塞更新。总之，组复制能够保证数据库服务是连续可用的。

要注意：在集群有成员发生崩溃的时候（少数成员发生崩溃），尽管数据库服务本身是可用的，但必须自行将连接到故障节点中的那些应用客户端连接重定向到另一个健康的节点，或者通过应用程序相关的自动故障转移程序将故障自动转移到到另一个健康的节点中。这种在某个单一节点发生故障时的客户端访问转移不是组复制需要解决的问题。通过负载均衡或某种形式的中间件更适合处理这个问题。总之，组复制 能够提供一个高可用性、高弹性、高可靠的MySQL服务。

下面是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。