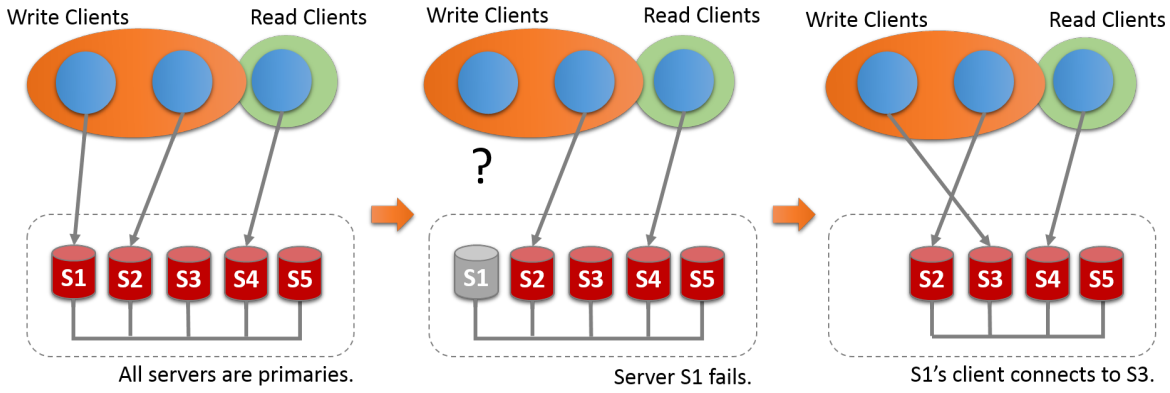
集群架构：多主模式---基础篇

1.多主模式：

在多主模式下（group_replication_single_primary_mode = OFF），所有成员不会区分primary和standby角色。加入该组时，与其他组成员兼容的任何成员都被设置为读写模式，并且可以处理写事务，即使它们是并发执行的。

如果组复制中的某个成员停止接受写事务，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成员。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。下图说明了MGR集群的多主模式下故障转移的实现：



组复制在集群内保证了系统的最终一致性。一旦入栈流量减少，所有组成员将具有相同的数据内容。当流量在集群内部下发时，事务可以在某些成员之间先进行持久化，特别是如果某些成员的写入吞吐量小于其他成员的情况下，则可能导致性能差的节点上读取到旧数据。在多主模式下，写入速度较慢的成员还可能积压过多的事务，从而导致更大的冲突和认证失败风险。为了避免这些问题，可以针对不同的业务场景使用组复制自带的流控机制，以最大程度地减少不同成员之间的事务差异。关于MGR的流控机制，我们在后面进行详细讨论。

从MySQL 8.0.14开始，如果要为集群中的每个事务都拥有一个事务一致性保证，则可以使用group_replication_consistency系统变量来做到这一点。可以选择适合集群工作负载和数据读写优先级的设置，同时考虑到提高一致性整个集群对性能的影响。还可以为单个会话设置该系统变量，用来保护特别是对并发敏感的事务。有关事务一致性的更多信息，我们在后面的章节详细描述。

2.事务检测：

当MGR集群是以多主的模式在线上运行时，集群通过以下两条原则对不同成员之间的事务进行严格的一致性检测，以保证这些事务可以在集群内提交成功：

1.如果在SERIALIZABLE隔离级别下执行事务，则在集群中同步数据时，该事务将提交失败。

2.如果事务是针对具有具有级联约束的外键的表执行的，则在集群中同步数据时，该事务将提交失败。

group_replication_enforce_update_everywhere_checks系统变量控制上述行为。在多主模式下，通常应将系统变量设置为ON，但是可以选择将系统变量设置为OFF来禁用检查。在单主模式下部署时，必须将系统变量设置为OFF。

3.在多主模式的复制拓补中，执行DDL需要格外小心：

MySQL 8.0引入了对原子数据定义语言（DDL）语句的支持，其中完整的DDL语句作为单个原子事务被提交或回滚。但是，DDL语句隐式结束当前会话中处于活动状态的事务，就好像在执行该语句之前执行了COMMIT一样。这意味着DDL语句不能在另一个事务中执行，例如在事务控制语句（START TRANSACTION ... COMMIT）中执行，也不能与同一事务中的其他语句组合。

如果对同一对象进行更改（使用DDL）并更改对象包含的数据（使用DML），则需要通过同一个节点处理更改，而DDL操作尚未完成并在各处复制。否则，当操作中断或仅部分完成时，可能导致数据不一致。如果集群以单主服务器模式部署，则不会发生此问题，因为所有数据更改都是通过同一个节点（主节点）执行的。

4.版本兼容性：

为了获得最佳的兼容性和性能，集群中的所有成员应运行相同版本的MySQL Server，因此应运行相同版本的组复制。在多主模式下，这更为重要，因为所有成员通常都将以读写模式加入该集群。如果集群中包含运行多个MySQL Server版本的成员，则某些成员可能与其他成员不兼容，因为它们支持其他成员不具备的功能或缺少其他成员具有的功能。为了防止这种情况，当新成员加入（包括以前已升级并重新启动的成员）时，该成员将对组的其余成员执行兼容性检查。

这些兼容性检查的结果在多主模式下尤其重要。如果新加入成员所运行的MySQL Server版本高于现有组成员所运行的最低版本，则它将加入该组，但保持只读模式。 （在以单主模式运行的集群中，无论如何，新添加的成员在任何情况下均默认为只读。）运行MySQL 8.0.17或更高版本的成员在检查其兼容性时会考虑该发行版的补丁程序版本。运行MySQL 8.0.16或更低版本或MySQL 5.7的成员仅考虑主要版本。

在具有使用不同MySQL Server版本的成员的多主模式下运行的集群中，组复制会自动管理运行在MySQL 8.0.17或更高版本的成员的读写状态和只读状态。如果成员离开集群，则运行当前最低版本的成员将自动设置为读写模式。当使用group_replication_switch_to_multi_primary_mode自定义函数时，将以单主模式运行的组更改为以多主模式运行时，组复制会自动将成员设置为正确的模式。如果新增成员运行的MySQL版本高于组中存在的最低版本的成员，则该成员将自动置于只读模式，而运行最低版本的成员将处于读写模式。

5.MGR服务：

5.1成员资格：

复制组由一组MySQL Server构成。集群具有一个唯一的名称，名称形式为UUID字符串。集群内的成员是动态的，成员可以随时脱离集群，也可以随时加入集群。每当有Server加入或脱离集群时，集群的相关信息都会进行自动调整。

如果一个Server加入了集群，则它会从集群的现有成员中自动获取自身缺失的数据状态以便和集群保持数据同步。如果一个成员脱离了集群，其余的成员会发现该成员脱离了集群，并自动重新配置集群。

组复制定义了集群内哪些成员在线且是活跃成员。在线成员列表被称为组视图。集群中的每个成员都具有一个一致的视图，即表示在给定的时刻集群中哪些成员是活跃成员。

组成员不仅在事务提交时必须达成一致，对于组视图的变更也必须达成一致。如果现有成员同意新的Server加入集群，则集群将被重新配置，以便将新节点集成到集群中，这将触发组视图的变更。如果组成员脱离集群，该集群将动态地重新配置，并触发组视图的变更。

在成员自愿脱离集群的情况下，它首先启动动态组重新配置，在此期间，所有组成员必须就新的组视图达成一致。但是，如果某个组成员是非自愿地脱离了集群，例如：因为意外宕机了或网络连接中断了，那么脱离集群的成员就不能启动动态重新配置。在这种情况下，组复制的故障检测机制会在短时间内识别出该成员已经脱离了集群，并建议将已脱离组成员排除在外并进行动态重新配置组。重新配置组需要得到组中大多数组成员的同意。但是，如果此时集群内不能达成一致，例如：由于集群内活跃节点数量少于节点总数量的半数时，系统就不能动态重新配置集群，这种情况下集群会阻塞写访问以防止出现脑裂的情况。直到人工介入处理。

在故障检测机制检测到其故障之前，或在重新配置集群以删除该故障成员之前，允许组成员短暂离线，然后尝试重新加入集群。在这种情况下，重新加入的成员可能会丢失它以前的事务数据，如果此时其他成员向它发送了包含该成员崩溃前的状态的消息，则这就可能会导致一些问题（例如：可能导致数据不一致。）

为了解决这个问题，从MySQL 5.7.22及其之后的版本中，当Server加入一个集群时，会被赋予一个唯一的标识符。这使组复制能够察觉到同一Server的新化身（虽然具有相同的地址，但不同的化身具有不同的标识符）试图加入组时，而其旧化身仍然会作为成员列出。在通过重新配置组并删除旧的化身之前，将阻止新的化身加入组。如果通过系统变量group_replication_member_expel_timeout的设置增加了疑似故障的成员在被驱逐出集群之前允许返回集群的等待时间，则只要疑似故障的成员不是真的故障了，那么被怀疑的那个成员在这个等待时间内就可以尝试重新加入集群。如果在被怀疑出故障的成员上执行了重启组复制，则该成员将成为新的化身，在怀疑超时之前（怀疑期时间内）不能重新加入组。

5.2故障检测：

组复制支持一种故障检测机制，该机制能够发现和列出哪些组成员是静默的，并假定静默的组成员已经挂机。总体上讲，故障检测器是一种分布式服务，它提供关于哪些组成员可能死机的信息。当组成员静默时就会引发怀疑。当组成员A在给定的时间段内没有接收来自组成员B的消息时，将发生消息超时并引发怀疑。稍后，如果组内其他所有的成员通过协商之后，都同意对该成员的怀疑可能是真实的（多数节点判定的结果），则集群就会判定被怀疑的成员确实发生了故障。这意味着组中其他成员能够通过采取协调一致的决策来将故障成员驱逐出组（被判定为发生故障的成员）。

如果某个组成员与组中的其他成员发生了网络隔离，那么它会怀疑集群中其他所有成员都发生故障了。由于无法与组内的其他成员进行协商（因为仲裁成员数不足），它的怀疑无效。此时，它也无法执行任何本地事务（只读事务可以执行）。

5.3容错性：

组复制建立在Paxos分布式算法的实现之上，以提供组成员之间的分布式协调。因此，它需要大多数组成员处于活跃状态才能达到仲裁成员数，才能够做出决策。该要求会直接影响到系统在不影响自身和整体可用性的情况下能够容忍发生故障的成员数量。可以容忍发生故障的成员数量（假设为f个）和要求组内总成员数量（假设为n个）之间的关系为：n = 2 x f + 1。

如果要容忍至少一个组成员发生故障，那么，组内的总成员数量至少需要3个。因此，当一个组成员发生故障时，仍然有2个组成员是活跃成员，即活跃成员占多数（三分之二），此时，集群内可通过仲裁机制自动驱逐故障成员并允许系统继续对外提供服务。但是，如果第二个组成员再发生故障（非自愿脱离组），则该集群（剩下一个成员）会发生阻塞，因为缺少多数成员来做出合理的决策，所以无法自动恢复到能对外提供服务的状态，此时需要人工介入处理。

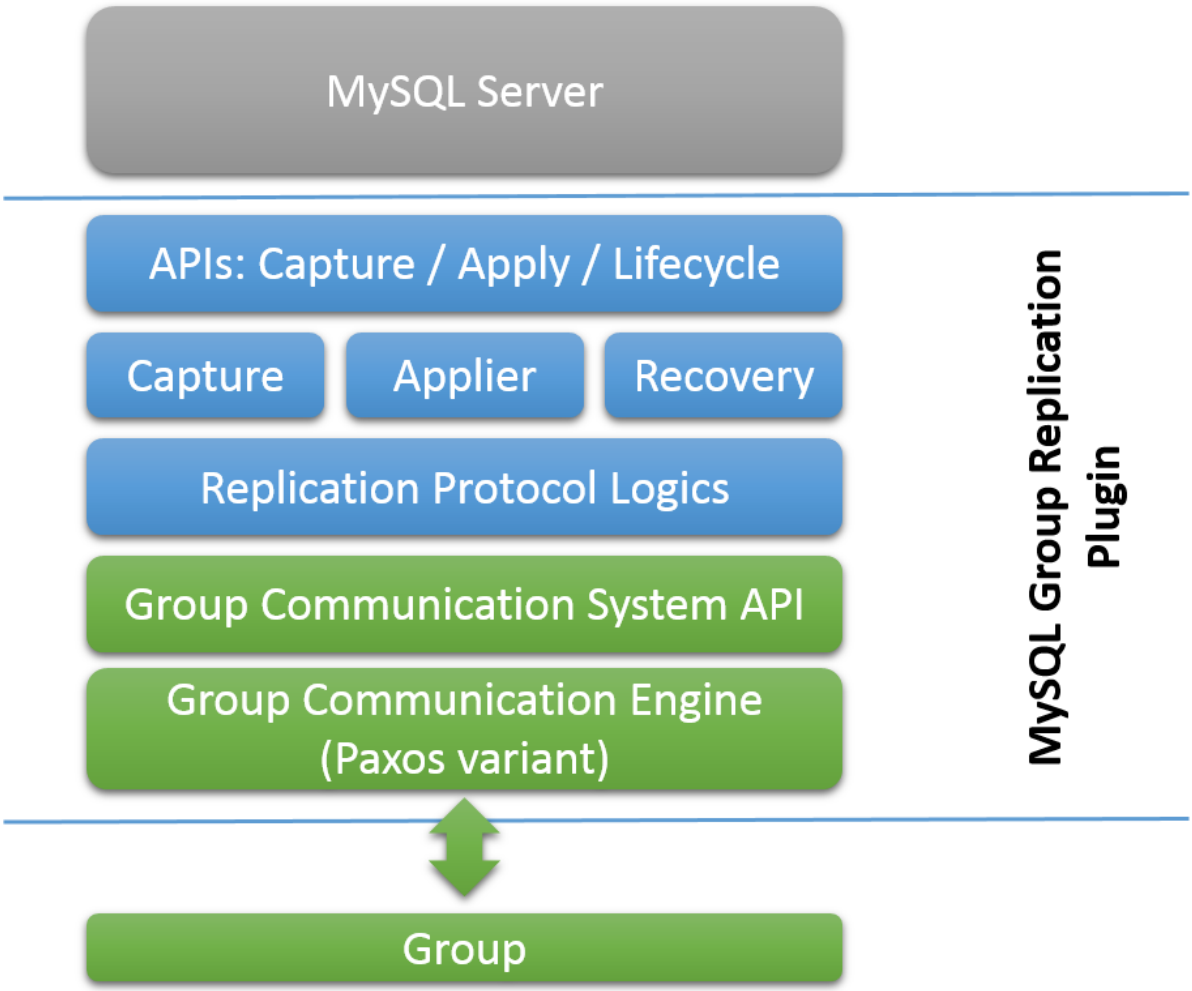
通常，基于性能和维护成本的考虑，组内的成员总数量不建议超过7个，最大只能9个。

5.4可观察性：

虽然MGR插件内置了很多自动化功能。但有时可能需要了解幕后发生了什么。如果有需要，可以通过performance_schema下的表查询集群的整个状态(包括视图、冲突统计信息和服务状态等)。复制协议的分布式特性、以及组成员在事务和元数据上的一致性，要求组内的一些元数据和状态信息在组内所有成员间相互同步，这就使得检查组的状态等信息变得更加简单。你只需要连接到集群内的任意一个成员中，并通过performance_schema下的相关复制状态信息表执行select语句进行查询，就可以获取到组相关的本地和全局信息。有关更多信息，我们放在MGR状态监控中会详细讨论。

5.MGR插件架构：

MGR的总体架构图如下：



MGR 插件包含一组用于捕获，应用和生命循环的API，这些API控制着MGR插件如何与MySQL Server交互。这些接口是放置在事务执行管道中的一些钩子（它们将MySQL Server的核心与MGR插件隔离开来），逻辑上将MySQL Server内核与MGR插件隔离开来。其中有一些接口提供把通讯信息从Server发送给MGR插件（例如：Server启动、恢复、接受连接以及Server即将提交事务的事件通知），有一些接口提供把通讯信息从MGR插件发送给MySQL Server（例如：MGR插件命令MySQL Server提交、终止一个正在执行的事务，或者让该事务写入relay log中排队等候处理）。

在这些API的下一层，是一组组件(capture、applier、Recovery)，组复制中的三个核心模块，当上层API发生调用，将根据调用类型路由到下面这三个模块执行相应的逻辑：

capture 组件负责跟踪与正在执行的事务相关的上下文信息。

applier 组件负责在数据库上应用远程事务，其实就是读取relay log中数据进行回放。

Recovery 组件管理数据库节点的分布式恢复相关的工作，以及负责在一个新的Server加入集群时选择一个引导节点，协调新加入节点的数据追赶的更新步骤（包括相关的数据回追，失败处理等），以及对选择引导节点失败之后做出一些响应。简而言之就是管理集群成员的recovery。

继续沿着堆栈向下，复制协议模块包含复制协议中的一些特定逻辑。例如：处理冲突检测，接收和传播事务到集群中。

MGR 插件体系结构的最后两层是组通信系统(GCS) API和基于paxos的组通信引擎(XCom)的实现。GCS API是一个高级API，它抽象了构建复制状态机所需的属性，具体属性我们在前文已经描述过了。因此，它将消息层的实现与插件的其余上层组件解耦。组通信引擎处理与复制组成员之间的通信，主要提供基于Paxos协议的变体实现数据一致性的核心功能。