第十四节：组复制性能---进阶篇（一）

1.组复制通信线程：

加载组复制插件并且成功运行该插件时，组通信线程（GCT）循环运行。 GCT接收集群和插件的消息，处理与仲裁和故障检测相关的任务，发送一些保持活跃的消息，并处理MySQL server和MGR之间的传入和传出事务。 GCT等待队列中消息传入，当队列中暂时没有消息传入时，GCT等待且处于静默状态。通过将等待时间配置为更长（进行主动等待），可以减少在操作系统进行上下文切换时从处理器中切换出GCT线程的次数。

要强制GCT进行主动等待，使用group_replication_poll_spin_loops系统变量，该变量使GCT循环在实际轮询队列中查找下一条消息之前，在配置的循环数之内不执行任何操作。

```
mysql> SET GLOBAL group_replication_poll_spin_loops= 10000;
```

2.流量控制：

2.1简介：

组复制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。如果对集群的写入总数不超过集群中任何节点的写容量，则此方法效果很好。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，则这些节点的数据可能开始落后于primary节点。

某些节点落后于集群会带来一些问题，特别是对此类节点的读取可能会在应用程序中读取到非常旧的数据。根据节点落后的原因，集群中的其他节点可能必须保存或多或少的复制上下文，才能满足慢节点的潜在数据传输请求。

然而，在复制协议中有一种机制可以避免快节点和慢节点之间的事务数量相差太多。这称为流量控制机制。

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。

考虑到组复制的设计，在决定是否启用流控机制时，需要考虑两个工作队列：

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。

流量控制取决于两个基本机制：

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。

2.2探测与统计：

监控机制通过在每个节点部署一组探针来收集有关其工作队列和吞吐量的信息而起作用。然后，它将这些信息定期传播到集群内部，与其他节点共享这些监控数据。

这些探针分散在整个插件堆栈中，并允许它们建立相应的度量指标，例如：

1.认证队列大小；

2.复制应用队列大小；

3.认证完成的事务总数；

4.集群内节点中应用的远程事务总数；

5.本地事务的总数。

ps：这些指标值在performance_schema.replication_group_member_stats表中能够查询到。

一旦集群内某个节点收到来自另一个节点的带有统计信息的消息后，它将计算在上一个监控探测期间已认证，应用和本地执行的事务数量等度量指标。

监控数据会定期与集群中的其节点共享。监控周期必须足够长，以便其他节点能够根据这些监控信息来确定当前的写请求量，但也必须足够低，以便对集群带宽的影响降低到最小。监控信息每秒钟都会被共享，一秒的时间间隔通常情况下能够解决这两个问题且能够很好地在这两个问题之间取得一个平衡。

2.3组复制节流：

组复制节流指的是根据从集群中所有节点收集的度量指标，将启动限流机制，并决定是否限制节点能够执行/提交新事务的速率的一种流控机制。

因此，从集群内所有节点获取的度量指标是计算每个节点的写压力的基础：如果一个节点的队列很大（用于认证或应用线程），那么执行新事务的能力应该接近于上一阶段（最后一次探测时间段）认证或复制应用事务的能力。

集群中所有节点的最低写容量决定了该集群的实际写容量，而本地事务的数量决定了向其写入的节点数量，因此也决定了应与多少个节点共享该写入容量。

这意味着每个节点都有一个基于可用容量的既定写入配额，换句话说，它就是在下一个阶段可以安全地（不受流控机制影响）执行的事务。如果认证队列或二进制日志应用队列大小超过用户定义的阈值，则通过限流机制强制执行写限额。

限额将根据上一阶段延迟的事务数量逐步减少10%，，以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出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。

3.消息压缩：

当网络带宽成为性能瓶颈时，消息压缩可以在组通信级别上将吞吐量提高达30-40%。这对于负载较大的大型集群尤其重要。下表列出了在不同的二进制日志格式下的LZ4压缩率。

工作负载模拟程序	行格式压缩比例	语句格式压缩比例
mysqlslapd	4,5	4,1
sysbench	3,4	2,9

集群中N个参与者之间的相互连接的TCP具有对等性质，使得发送方需要发送N次相同数量的数据（例如：集群中有3个节点，那么发送方就要发送3次相同的数据）。此外，二进制日志可能具有较高的压缩比（见上表）。这使得压缩对于包含大事务的工作负载来说是一个引人注目的特性。

对于集群中online节点之间发送的消息，默认情况下，组复制启用消息压缩。是否压缩特定消息取决于使用group_replication_compression_threshold系统变量配置的阈值。有效载荷大于指定字节数的消息将被压缩。

默认压缩阈值为1000000字节。可以使用以下语句将压缩阈值增加到2MB：

```
STOP GROUP_REPLICATION;
SET GLOBAL group_replication_compression_threshold = 2097152;
START GROUP_REPLICATION;
```

如果将group_replication_compression_threshold设置为零，则禁用消息压缩。

组复制使用LZ4压缩算法来压缩在集群中发送的消息。LZ4压缩算法支持的最大输入大小为2113929216字节。此限制低于group_replication_compression_threshold系统变量的最大可能值，该值与XCom接受的最大消息大小匹配。因此，LZ4最大输入大小是消息压缩的实际限制，并且在启用消息压缩时无法提交超过此大小的事务。使用LZ4压缩算法时，不要为group_replication_compression_threshold设置大于2113929216字节的值。

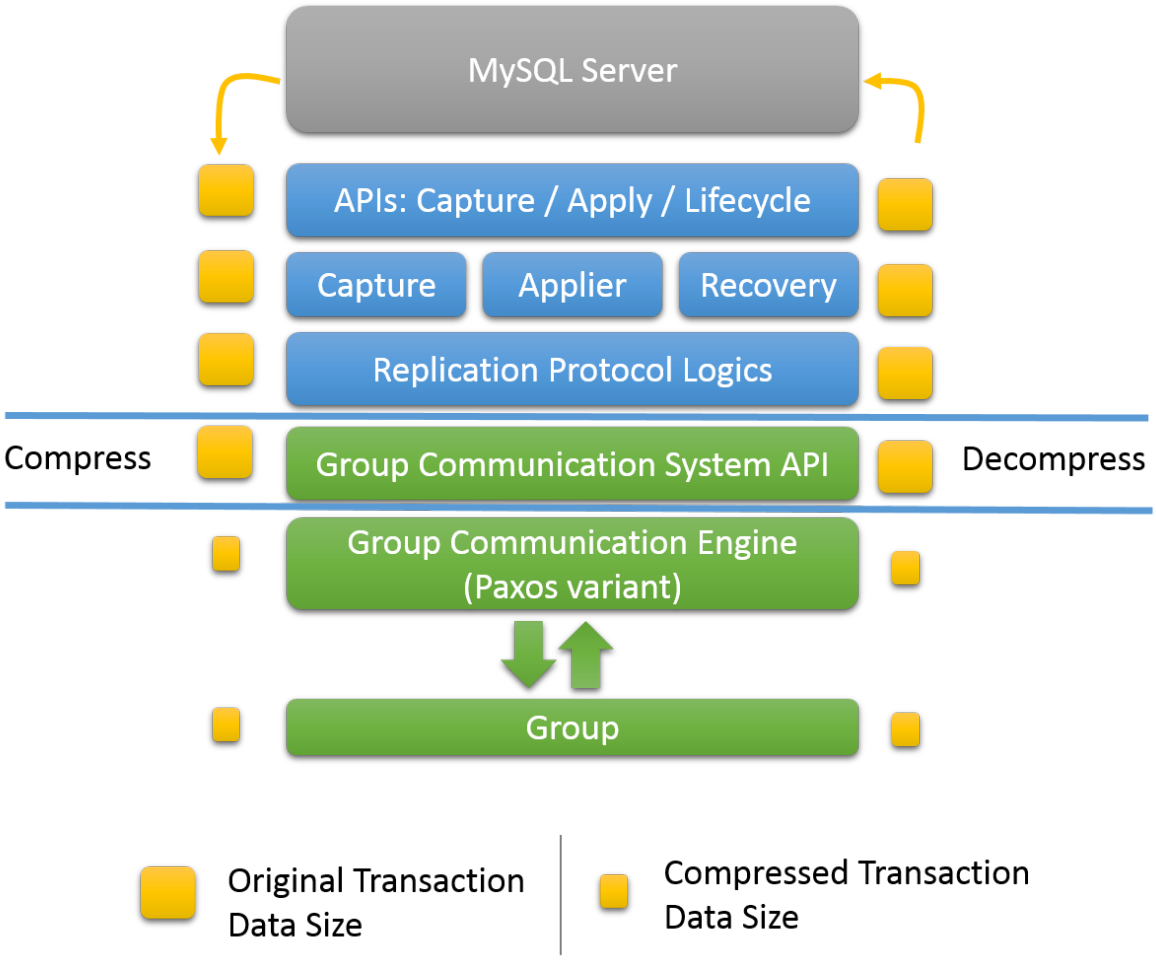
组复制并不要求group_replication_compression_threshold的值在所有节点上都相同。但建议在所有节点上设置相同的值，以避免不必要的事务回滚，消息传递失败或消息恢复失败。

从MySQL 8.0.18起，还可以通过引导节点二进制日志中的状态转移方法为发送给分布式恢复的消息配置压缩。从集群中已有的引导节点发送到加入节点的这些消息的压缩，使用group_replication_recovery_compression_algorithm和group_replication_recovery_zstd_compression_level系统变量来分别控制。

由binlog_transaction_compression系统变量激活的二进制日志事务压缩（自MySQL 8.0.20起可用）还可用于节省带宽。在集群节点之间传输事务负载时，它们保持压缩状态。如果将二进制日志事务压缩与组复制的消息压缩结合使用，则消息压缩将很少有机会对数据进行操作，但仍可以压缩标头以及未压缩的那些事件和事务有效负载。

在集群中发送的消息的压缩发生在组通信引擎级别，即数据移交给组通信线程之前，因此它在mysql用户会话线程的上下文中进行。如果消息有效负载大小超过了由group_replication_compression_threshold设置的阈值，则在将事务负载发送给集群之前对其进行压缩，并在接收到该消息时对其进行解压缩。节点收到消息后，将检查消息信封以验证消息是否被压缩。如果需要，节点在将事务传递到上层之前先解压缩事务。

在MGR插件中，支持压缩的是组通讯API组件，如下图 所示:



MGR 插件架构如上图所示，压缩功能在插件的五层（位于MySQL Server和复制组之间，刨去MySQL Server层，它是MGR插件的第四层，也就是组通讯引擎层)。压缩和解压缩的工作任务由组通信系统API处理。

4.消息分段：

在组复制节点之间发送异常大的消息时，可能导致某些节点被判定为发送失败并从该集群中驱逐。这是因为组复制的组通信引擎（XCom，一个Paxos变体）使用的单线程占用的消息处理时间太长，因此某些节点可能会报告接收者失败。从MySQL 8.0.16开始，默认情况下，大型消息会自动拆分为片段，分别发送并由接收节点重新组合。

系统变量group_replication_communication_max_message_size指定用于组复制通信的最大消息大小，在该最大消息大小之上，消息将被分段。默认的最大消息大小为10485760字节（10 MiB）。允许的最大值与slave_max_allowed_packet系统变量的最大值相同，为1073741824字节（1 GB）。group_replication_communication_max_message_size的设置必须小于slave_max_allowed_packet设置，因为应用程序线程无法处理大于slave_max_allowed_packet的消息片段。要关闭分段，请为group_replication_communication_max_message_size指定零值。

与大多数其他Group Replication系统变量一样，必须重新启动Group Replication插件才能使变更生效。例如：

```
STOP GROUP_REPLICATION;
SET GLOBAL group_replication_communication_max_message_size= 5242880;
START GROUP_REPLICATION;
```

当集群内所有节点均已接收并重新组合了消息的所有片段时，就认为分段消息的消息传递已完成。分段消息的头部中包含信息，这些信息使节点能够在消息传输期间加入集群，以恢复在加入集群之前发送的较早的消息片段。如果加入节点未能恢复消息片段，则将其从集群中驱逐出去。

为了使复制组使用分段，所有组成员必须都在MySQL 8.0.16或更高版本上，并且该集群使用的“组复制”通信协议版本必须支持消息分段。可以使用group_replication_get_communication_protocol（）UDF来检查将集群正在使用的通信协议，该UDF返回该集群支持的最旧的MySQL Server版本。 MySQL 5.7.14的版本允许压缩消息，而MySQL 8.0.16的版本也允许消息分段。如果所有组成员都运行在MySQL 8.0.16或更高版本，并且不需要允许低版本的节点加入，则可以使用group_replication_set_communication_protocol（）UDF将通信协议版本设置为MySQL 8.0.16或更高版本，这样就能够确保消息分段功能在集群中所有节点上正常运行。

如果复制组由于某些成员不支持而无法使用消息分段，则可以使用系统变量group_replication_transaction_size_limit来限制该组接受的最大事务大小。在MySQL 8.0中，默认设置约为143 MB。超过此大小的事务将回滚。还可以使用系统变量group_replication_member_expel_timeout允许额外的时间（最多一个小时），然后才将疑似提交失败的节点从该集群中驱逐出去。

5.XCom缓存管理：

5.1XCom缓存简介：

用于组复制的组通信引擎（XCom, Paxos变体）包含了一个消息（及其元数据）缓存，该消息是作为集群内不同节点之间交换协商一致性协议的一部分。在其他用途中，消息缓存可用于在一段时间内无法与其他集群节点通信的节点在重新返回到集群时进行恢复。

从MySQL 8.0.16开始，可以使用group_replication_message_cache_size系统变量为XCom的消息缓存设置缓存大小限制。如果达到缓存大小限制，则XCom会删除已决定并已交付的最旧的条目。

在MySQL 8.0.16之前，缓存大小为1 GB，MySQL 8.0.16中的缓存大小和默认设置相同。考虑到MySQL Server其他对象和缓存池的大小，请确保系统上有足够的内存用于所选的缓存大小限制。请注意，使用group_replication_message_cache_size设置的限制仅适用于缓存中存储的数据，并且缓存结构需要额外的50 MB内存。

选择group_replication_message_cache_size设置时，请参考驱逐节点之前的时间段内的预期消息量。此时间段的长度由group_replication_member_expel_timeout系统变量控制，该系统变量确定除了最初的5秒检测时间（节点返回到集群而不是被驱逐）之外，还允许的等待时间（最多一小时）。请注意，在MySQL 8.0.21之前，此时间段默认为从节点变得不可用起的5秒，这恰好是创建可疑之前的检测时间段，因为group_replication_member_expel_timeout系统变量设置的附加expel超时默认为零。从8.0.21开始，退出超时默认为5秒，因此默认情况下，节点缺席至少10秒钟后才会被驱逐。

应该对集群内所有节点设置相同的缓存大小限制，因为尝试重新连接的不可达节点会随机选择任何其他节点以恢复丢失的消息。因此，每个节点的缓存中都应该有相同的消息。

如果一个不可达节点尝试重新恢复连接时，需要一条恢复消息，但该消息已从消息缓存中删除，则该节点无法重新连接。如果使用了系统变量group_replication_member_expel_timeout（该系统变量在MySQL 8.0.13中引入）指定一个额外的延迟时间，则更有可能出现这种情况。当不可达的节点恢复时可能需要使用到的消息在消息缓存中已经被删除时，组复制的组通信系统(GCS)通过一条警告消息来发出警告。此警告消息记录在所有活跃的组成员上（对于每个不可到达的成员仅记录一次）。尽管组成员不能确定不可到达的节点最后看到的消息是什么消息，但是警告消息表明缓存大小可能不足以支撑通过系统变量group_replication_member_expel_timeout设置的在驱逐成员之前的等待时间内总的消息大小。在这种情况下，可以增加缓存大小限制，以便消息缓存能够存放组成员重新加入集群所需的所有遗漏消息。

5.2增加缓存大小：

如果某个节点缺席的时间不足以将其从集群中驱逐出去，则可以通过从另一个成员的XCom消息缓存中检索丢失的事务来重新连接并重新开始加入该集群。但是，如果由于达到节点的最大大小限制而从其他节点的XCom消息缓存中删除了在该节点不在期间发生的事务，则该节点无法以这种方式重新加入集群。

当从消息缓存中删除当前无法访问的节点可能需要恢复的消息时，组复制的组通信系统（GCS）会通过警告消息发出警报。该警告消息记录在集群内所有online节点上（每个不可达节点仅记录一次）。尽管组成员不能肯定地知道什么消息是无法访问的节点看到的最后一条消息，但是警告消息表示缓存大小可能不足以支持在驱逐节点之前选择的等待时间。

在这种情况下，考虑参考group_replication_member_expel_timeout系统变量指定的时间段内的预期消息量，再增加group_replication_message_cache_size限制，再加上5秒检测周期，以便缓存包含节点成功返回所需的所有丢失消息。 如果希望节点在不寻常的时间内无法访问，也可以考虑暂时增加缓存大小限制。

5.3减少缓存大小：

MySQL 8.0.20为止，XCom消息缓存大小的最小设置为1 GB。从MySQL 8.0.21开始，最小设置为134217728字节（128 MB），这使得可以在可用内存量有限的主机上进行部署。如果主机位于不稳定的网络上，则不建议将group_replication_message_cache_size设置为非常低，因为较小的消息缓存会使组成员在短暂失去连接后很难重新连接。

如果重新连接的节点无法从XCom消息缓存中检索其所需的所有消息，则该节点必须离开该集群并重新加入该集群，以便使用分布式恢复从另一个节点的二进制日志中检索丢失的事务。从MySQL 8.0.21开始，默认情况下，离开集群的节点会进行三次自动重新加入尝试，因此重新加入集群的过程仍然可以在没有干预的情况下进行。但是，与从XCom消息缓存中检索消息相比，使用分布式恢复进行重新加入的过程要长得多，也更加复杂，因此该节点花了更长的时间才可以使用，并且该集群的性能可能会受到影响。在一个稳定的网络上，该网络可以最大程度地减少节点暂时失去连接的频率和持续时间，这种情况的发生频率也应该最小化，这样该集群就可以承受较小的XCom消息缓存大小，而不会对其性能产生重大影响。

如果考虑减少缓存大小限制，则可以使用以下语句查询performance_schema.memory_summary_global_by_event_name：

```
SELECT * FROM performance_schema.memory_summary_global_by_event_name
  WHERE EVENT_NAME LIKE 'memory/group_rpl/GCS_XCom::xcom_cache';
```

将返回消息高速缓存的内存使用情况统计信息，包括当前高速缓存的条目数和高速缓存的当前大小。如果减小缓存大小限制，则XCom会删除已确定并交付的最旧的条目，直到当前大小低于该限制为止。XCom可能会暂时超过缓存大小限制，而此删除过程正在进行中。