日志同步优化功能设计文档

修订历史

版本	修订日期	修订描述	作者	备注
Cedar 0.2	2016-07-05	日志同步优化功能设计文档	储佳佳 郭进伟	

1 需求分析

Cedar 0.1版本中已经实现了三集群架构下的主备节点日志强同步&强一致的功能。但是在应用过程中发现，该功能存在性能优化空间，而且在一些极端场景中仍然会出现日志丢失的情况。为了规避异常、提升整体性能，在Cedar 0.2中我们对日志同步流程进行了优化。优化目标主要有下述几点：

- 将最大可提交日志号写入日志记录中，避免额外线程频繁将提交点信息写入文件的操作，减少主节点非必需的资源开销。
- 调整备节点接收日志后的处理顺序为先将日志写入磁盘，再回复主UPS，最后回放日志，减小主节点答复客户端的延迟，提升更新操作的性能。
- 优化主备切换时对于未决日志的处理方式，规避日志丢失情况的出现。

2 功能简述

Cedar支持主备集群的配置，每个集群内配置一台RootServer（集群管理节点，以下简称“RS”）和一台UpdateServer（事务管理节点，以下简称“UPS”），为了提供高可用性，主集群的UPS处理事务更新请求后，会将相关日志同步到备集群，备集群进行日志回放、写盘，从而达到和主集群一致的状态。主备集群UPS之间的日志传递、日志写盘、日志回放等一系列操作即是日志同步模块的主要功能。

在Cedar中，日志同步采取的是强同步&强一致的机制，每次主UpdateServer接收到事务的更新请求时，会生成操作日志，接着将更新操作的日志发送到所有的备UpdateServer。备UpdateServer接收到主UpdateServer的日志后，确认将日志写入磁盘成功后才能响应主UpdateServer，主UpdateServer接收到多数派（超过半数）UpdateServer的响应之后才能提交对应的事务。

主备UpdateServer日志同步的流程如下：

1. 主UPS将CommitLog异步发送给所有的备UPS，然后将该批日志同步刷入到本地磁盘；
2. 备UPS将从主UPS接收到的CommitLog存入本地磁盘；
3. 备UPS将日志写盘成功后应答主UPS；并在应答消息中携带下一次刷磁盘的日志序列号和本机的状态；
4. 备UPS进行日志回放；
5. 主UPS收到备UPS的响应后，更新本地保存的备UPS信息；
6. 主UPS根据多数派规则，从所有的备UPS信息中获取到可以提交的日志号，从而相应的事务进入到提交阶段。

3 设计思路

3.1 子功能模块划分

Cedar 0.2在0.1的基础上进行日志同步功能的优化，主要的修改点可分为“日志传递提交点信息”、“调换备节点日志写盘和日志回放的顺序”“主备切换时未决日志的处理”三个子模块。

3.2 总体设计思路

相较于Cedar 0.1版本中用单独的线程频繁地将日志提交点信息写入本地文件，0.2中修改了原有的日志记录的结构体，将日志提交点信息加入其中，每次主备节点同步日志时，日志提交点信息随着日志记录在主备节点之间进行同步。备机在接收到主机发送来的日志时，会解析日志获取到提交点信息，并将本地小于该提交点的日志进行提交。

Cedar 0.1中备机接收到日志后的处理流程为，先进行日志回放，再写入本地磁盘，最后回复主机。因为采取强一致的模型，主机需要收集到至少多数派备机的回复后才能进行事务的提交操作，备机的保守式操作（回复主机发生在确保回放、落盘操作都完成以后）会影响到主机的事务响应效率。所以在Cedar 0.2中，修改备机接收日志后的处理流程为，先将日志写入本地磁盘，再回复主机，最后备机进行日志回放。这样就能在保证强一致的前提下，尽可能减少主机响应客户端事务请求的延迟时间。

当主节点因为租约过期、网络抖动等原因切换为备节点时，本地可能会有一些未决日志（由主机生成但还未同步到备机或是还未进行提交的日志），为了避免丢失日志的异常情况，当切换为备节点时，该节点需要对从主节点接收到的新日志和本地未决日志进行逐一进行比对的操作，若接收到的日志和本地未决日志相同，则保留本地未决日志，若不同，则删除不同点之后的所有未决日志，从主机重新拉取相应的日志。等处理完本地未决日志后，前任主UPS可进入正常的日志同步流程。

4. 参考文献

[1] Guo J, Zhang C, Cai P, et al. Low Overhead Log Replication for Main Memory Database System[M]// Web-Age Information Management. 2016.

[2] Ongaro D, Ousterhout J. In search of an understandable consensus algorithm[J]. Draft of October, 2013.

[3] Rao J, Shekita E J, Tata S. Using Paxos to build a scalable, consistent, and highly available datastore[J]. Computer Science, 2011, 4(4):243-254.