MySQL复制架构

复制的体系结构有以下一些基本原则：

(1) 每个slave只能有一个master；

(2) 每个slave只能有一个唯一的服务器ID；

(3) 每个master可以有很多slave；

(4) 如果你设置log_slave_updates，slave可以是其它slave的master，从而扩散master的更新。

MySQL不支持多主服务器复制(Multimaster Replication)——即一个slave可以有多个master。但是，通过一些简单的组合，我们却可以建立灵活而强大的复制体系结构。

1. 单主多从

适用于写操作少，读操作多的数据库

结构虽然简单，但是，它却非常灵活，足够满足大多数应用需求。一些建议：

(1) 不同的slave扮演不同的作用(例如使用不同的索引，或者不同的存储引擎)；

(2) 用一个slave作为备用master，只进行复制；

(3) 用一个远程的slave，用于灾难恢复；

但当slave增加到一定数量时，slave对master的负载以及网络带宽都会成为一个严重的问题。

2. 双主动模式的主主

两个数据库都能读写，这就有更新冲突的可能。MySQL并不支持其它一些DBMS支持的多主服务器复制(Multimaster Replication)，这是MySQL的复制功能很大的一个限制(多主服务器的难点在于解决更新冲突)，但是，如果你实在有这种需求，你可以采用MySQL Cluster，以及将Cluster和Replication结合起来，可以建立强大的高性能的数据库平台。通过其它一些方式来模拟这种多主服务器的复制。

3. 主动被动模式的主主

这是master-master结构变化而来的，它避免了M-M的缺点，实际上，这是一种具有容错和高可用性的系统。它的不同点在于其中一个服务只能进行只读操作。

4. 级联复制架构 Master - Slaves - Slaves

在有些应用场景中，可能读写压力差别比较大，读压力特别的大，一个Master可能需要上10台甚至更多的Slave才能够支撑注读的压力。这时候，Master就会比较吃力了，因为仅仅连上来的SlaveIO线程就比较多了，这样写的压力稍微大一点的时候，Master端因为复制就会消耗较多的资源，很容易造成复制的延时。

这时候我们就可以利用MySQL可以在Slave端记录复制所产生变更的BinaryLog信息的功能，也就是打开—log-slave-update选项。然后，通过二级（或者是更多级别）复制来减少Master端因为复制所带来的压力。也就是说，我们首先通过少数几台MySQL从Master来进行复制，这几台机器我们姑且称之为第一级Slave集群，然后其他的Slave再从第一级Slave集群来进行复制。从第一级Slave进行复制的Slave，我称之为第二级Slave集群。如果有需要，我们可以继续往下增加更多层次的复制。这样，我们很容易就控制了每一台MySQL上面所附属Slave的数量。这种架构我称之为Master-Slaves-Slaves架构。

这种多层级联复制的架构，很容易就解决了Master端因为附属Slave太多而成为瓶颈的风险。

上述瓶颈问题 ： 毕竟Slave并没有减少写的量，所有Slave实际上仍然还是应用了所有的数据变更操作，没有减少任何写IO。相反，Slave越多，整个集群的写IO总量也就会越多，我们没有非常明显的感觉，仅仅只是因为分散到了多台机器上面，所以不是很容易表现出来。

此外，增加复制的级联层次，同一个变更传到最底层的Slave所需要经过的MySQL也会更多，同样可能造成延时较长的风险。

而如果我们通过分拆集群的方式来解决的话，可能就会要好很多了，当然，分拆集群也需要更复杂的技术和更复杂的应用系统架构。

5. 带从服务器的主主

这种结构的优点就是提供了冗余。在地理上分布的复制结构，它不存在单一节点故障问题，而且还可以将读密集型的请求放到slave上。

这种DualMaster与级联复制结合的架构，最大的好处就是既可以避免主Master的写入操作不会受到Slave集群的复制所带来的影响，同时主Master需要切换的时候也基本上不会出现重搭Replication的情况。

但是，这个架构也有一个弊端，那就是备用的Master有可能成为瓶颈，因为如果后面的Slave集群比较大的话，备用Master可能会因为过多的SlaveIO线程请求而成为瓶颈。

当然，该备用Master不提供任何的读服务的时候，瓶颈出现的可能性并不是特别高，如果出现瓶颈，也可以在备用Master后面再次进行级联复制，架设多层Slave集群。当然，级联复制的级别越多，Slave集群可能出现的数据延时也会更为明显，所以考虑使用多层级联复制之前，也需要评估数据延时对应用系统的影响。

参考 ： http://blog.csdn.net/hguisu/article/details/7325124/