链滴

基于 Docker 的 MySQL 主从复制

作者：zouchanglin
原文链接：https://ld246.com/article/1566108236063
来源网站：链滴

<h2 id="再谈谈数据库优化">再谈谈数据库优化</h2>
<img src="https://ld246.com/images/img-loading.svg" alt="1.png" data-src="https://b3ogfile.com/file/2019/08/1-21a9a2b6.png?imageView2/2/interlace/1/format/jpg">
如上图，MySQL 优化呢无非就是从这几个方面入手，第一是数据库设计，第二是 SQL 优化！但随着数据表的增长，数据会越来越多，通过 MySQL 优化是可以解决部分性能问题的，但是一台服务的资源是有限的，无论怎么优化始终无法解决 MySQL 的性能瓶颈问题。举一个经典的例子，一个卡无论再怎么优化发动机，优化传动结构，优化材料设计都没办法拉着金字塔这种沉重的负载，这种情就属于 MySQL 的瓶颈，无论如何优化，只要数据量达到一定的规模，一台 MySQL 肯定是撑不住的
<img src="https://ld246.com/images/img-loading.svg" alt="2.png" data-src="https://b3ogfile.com/file/2019/08/2-3c7f6793.png?imageView2/2/interlace/1/format/jpg">
<h2 id="如何用架构进行优化">如何用架构进行优化</h2>
对于千万级的表如何进行优化呢？见下图，加索引，优化 SQL 都是之前的优化方法，至于不解问题，把问题留给后面的程序员这种做法最好还是放弃吧，啊哈哈，其中加缓存和数据库拆分是很常的解决方案，下面说一下如何通过 MySQL 主从复制来解决 MySQL 的性能瓶颈问题：
<img src="https://ld246.com/images/img-loading.svg" alt="3.png" data-src="https://b3logfil.com/file/2019/08/3-cf0d338b.png?imageView2/2/interlace/1/format/jpg">
<h2 id="toc_h2_2"></h2>
主从复制（又叫做读写分离），主从复制的目的：分散压力
为什么要读写分离呢？如果对数据库的读操作和写操作都在同一个数据库服务器中进行，业务系性能会降低，所以需要进行读写分离，通常情况下遵循二八原则，即 20% 的时候进行写操作，80% 时候进行读操作：
<img src="https://ld246.com/images/img-loading.svg" alt="4.png" data-src="https://b3ogfile.com/file/2019/08/4-1ee0cf37.png?imageView2/2/interlace/1/format/jpg">
生活中有很形象的例子，比如你在自助咖啡厅买咖啡（如果只有一台咖啡机）：
<img src="https://ld246.com/images/img-loading.svg" alt="5.png" data-src="https://b3ogfile.com/file/2019/08/5-8ae878b0.png?imageView2/2/interlace/1/format/jpg">
如果有多台咖啡机，很明显大家买咖啡的效率就上去了：
<img src="https://ld246.com/images/img-loading.svg" alt="6.png" data-src="https://b3ogfile.com/file/2019/08/6-1ef7bfd9.png?imageView2/2/interlace/1/format/jpg">
所以主从复制的简单原理图如下：
<img src="https://ld246.com/images/img-loading.svg" alt="7.png" data-src="https://b3ogfile.com/file/2019/08/7-e8e55387.png?imageView2/2/interlace/1/format/jpg">
一台 MySQL 作为写服务器，另外几台 MySQL 作为读服务器，这样便完成了分散压力，为了几服务器之间的数据一致，所以需要做一个数据库主从复制，即读写分离！
<h2 id="Docker实现MySQL主从复制">Docker 实现 MySQL 主从复制</h2>
那么如何使用 Docker 实现 MySQL 主从复制呢？我先在 Docker 里面跑了两个 MySQL：
Docker 命令是：<code>docker run -p 3306:3306 --restart=always --name mysql_master v /root/mysql/conf:/etc/mysql/conf.d -v /root/mysql/logs:/logs -v /root/mysql/data:/var/lib/ ysql -e MYSQL_ROOT_PASSWORD=123456 -d mysql</code>
<img src="https://ld246.com/images/img-loading.svg" alt="8.png" data-src="https://b3ogfile.com/file/2019/08/8-fcfd3f3e.png?imageView2/2/interlace/1/format/jpg">
接着查看一下两台机器的 IP 地址
<img src="https://ld246.com/images/img-loading.svg" alt="9.png" data-src="https://b3ogfile.com/file/2019/08/9-d40fae41.png?imageView2/2/interlace/1/format/jpg">
首先进入到 master 机器，使用 vi 编辑器编辑 <code>/etc/my.cnf</code>，我直接使用 doc er pull mysql:5.7 ，里面默认是 Ubuntu 系的内核，所以没有 vim 编辑器，可以使用 apt 包管理器 载，下载命令是 apt install vim ，如果无法下载那应该是没有更新，使用 apt update 即可更新！但 我发现其实我的配置文件地址不一样，我的在 <code> /etc/mysql/mysql.conf.d/mysqld.cnf</cod > ，编辑这个配置文件就好了，主机的配置文件：
<img src="https://ld246.com/images/img-loading.svg" alt="11.png" data-src="https://b3ogfile.com/file/2019/08/11-516c7111.png?imageView2/2/interlace/1/format/jpg">
编辑完成后重启 mysql，CentOS 下直接 <code>systemctl restart mysql</code> 即可，但在 Ubuntu 要使用的命令是 <code>service mysql restart</code> （另外，在 Ubuntu 下重启 M

SQL 会导致用户退出 Docker 容器，Docker 关闭，所以需要再次开启 MySQL 的容器，使用 <code docker exec -it mysql_master /bin/bash</code> 进入容器，然而在 CentOS 下却不需要）
接下来给主机添加一个用户，并设置密码，然后重启容器：
<img src="https://ld246.com/images/img-loading.svg" alt="12.png" data-src="https://b3ogfile.com/file/2019/08/12-f659c45b.png?imageView2/2/interlace/1/format/jpg">
接下来进入主机，使用如下命令即可看到 binaryLog，以及 binaryLog 文件的偏移量：
<img src="https://ld246.com/images/img-loading.svg" alt="13.png" data-src="https://b3ogfile.com/file/2019/08/13-2df73c1b.png?imageView2/2/interlace/1/format/jpg">
接着需要配置从机，和编辑主机的配置文件是一样的，只不过从机的配置只需要指定 server_id 可，我们指定从机的 server_id=2，然后停止同步线程， 并做主从配置，完成后开启同步线程：
<img src="https://ld246.com/images/img-loading.svg" alt="14.png" data-src="https://b3ogfile.com/file/2019/08/14-e820e5da.png?imageView2/2/interlace/1/format/jpg">
检查是否开启成功：
<img src="https://ld246.com/images/img-loading.svg" alt="15.png" data-src="https://b3ogfile.com/file/2019/08/15-ba3c5351.png?imageView2/2/interlace/1/format/jpg">
查看主从复制的状态
<img src="https://ld246.com/images/img-loading.svg" alt="16.png" data-src="https://b3ogfile.com/file/2019/08/16-0f06750f.png?imageView2/2/interlace/1/format/jpg">
接下来测试一下，主机新建数据库，从机自动同步，主机新建表，从机自动同步，主机插入数据从机自动同步：
<img src="https://ld246.com/images/img-loading.svg" alt="17.png" data-src="https://b3ogfile.com/file/2019/08/17-0c26b280.png?imageView2/2/interlace/1/format/jpg">
<h2 id="主从复制的原理">主从复制的原理</h2>
上面已经完成了 MySQL 的主从复制，接下来看看主从复制究竟是怎么实现的？
<img src="https://ld246.com/images/img-loading.svg" alt="10.png" data-src="https://b3ogfile.com/file/2019/08/10-6f003408.png?imageView2/2/interlace/1/format/jpg">
上面的图其实很能说明问题，主机的数据发生更改的时候会产生一个 BinaryLog 文件，然后从的 IO 线程会去取这个二进制文件，取回来之后会将主机的 BinaryLog 拷贝到中继日志中，SQL 线程接通过中继日志来改变自身的数据。那么说到这里可能很多人会有疑问，不是说主机用来写，从机用读吗？为啥还是需要从机的写操作，事实上这并不是写操作，而且直接修改数据的操作，举一个很简的例子，如果是普通的修改数据，会首先找到要修改的数据的二进制位置，这样才能发生更改，如果接告诉你在哪个位置要修改成什么数据，会不会更快呢？很明显，主从复制是非常能提交数据库系统效率的。
一个小坑：如果主机上面的并发量特别高的话，从机同步数据的能力会下降，甚至一个数据插入主机几小时后才能同步到从机，这种情况的根本原因就是从机的 IO 线程是单线程的，如果配置为多程就能解决这种问题！