[精選] MySQL提高性能，緩解數據庫壓力，你會做分層分離嗎？

學習與分享 PHP自學中心 昨天

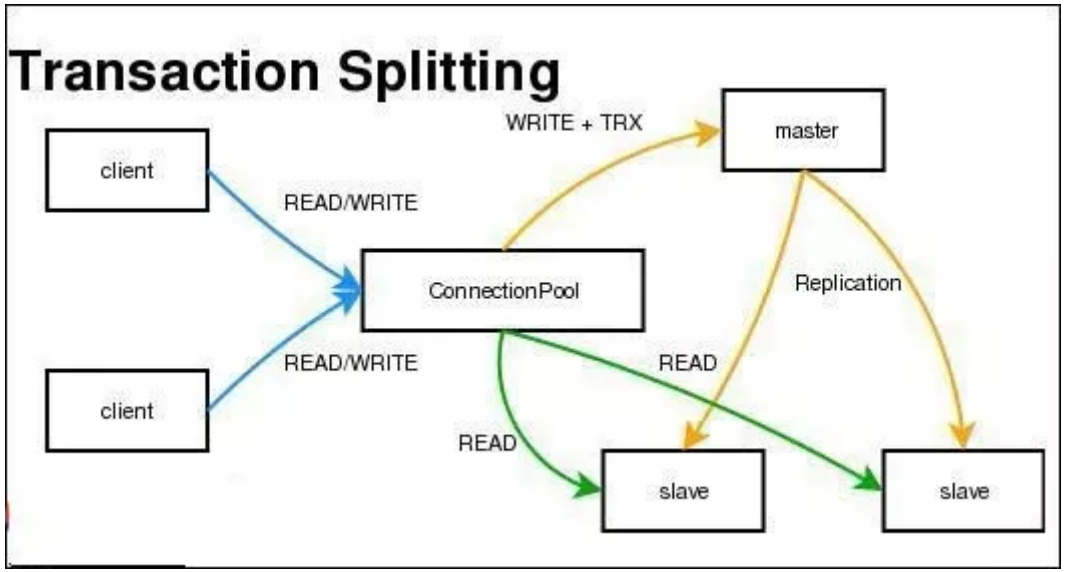
學習與交流：PHP技術交流微信群
商務合作加微信：2230304070

雖然知道處理大數據量時，數據庫為什麼要做識別分離，原因很簡單：識別分離是MySQL優化的替代品，它可以提高性能，緩解數據庫壓力，減輕服務器壓力。

一什麼是讀寫分離

MySQL Proxy最強大的一項功能是實現“讀寫分離（Read / Write Splitting）”。基本的原理是讓主數據庫處理事務性查詢，而從數據庫處理SELECT查詢。

當然，主服務器也可以提供查詢服務。使用讀寫分離最大的作用無非是環境服務器壓力。可以看下這張圖：



二讀寫分離的好處

1.增加冗餘

2.增加了機器的處理能力

3.對於讀操作主要的應用，使用讀寫分離是最好的場景，因為可以確保寫入的服務器壓力更小，而讀又可以接受點時間上的延遲。

三讀寫分離提高性能之原因

1.物理服务器增加，负荷增加

2.主从只负责各自的写和读，极大程度的缓解X锁和S锁争用

3.从库可配置myisam引擎，提升查询性能以及节约系统开销

4.从库同步主库的数据和主库直接写还是有区别的，通过主库发送来的binlog恢复数据，但是，最重要区别在于主库向从库发送binlog是异步的，从库恢复数据也是异步的

5.读写分离适用与读远大于写的场景，如果只有一台服务器，当select很多时，update和delete会被这些select访问中的数据堵塞，等待select结束，并发性能不高。对于写和读比例相近的应用，应该部署双主相互复制

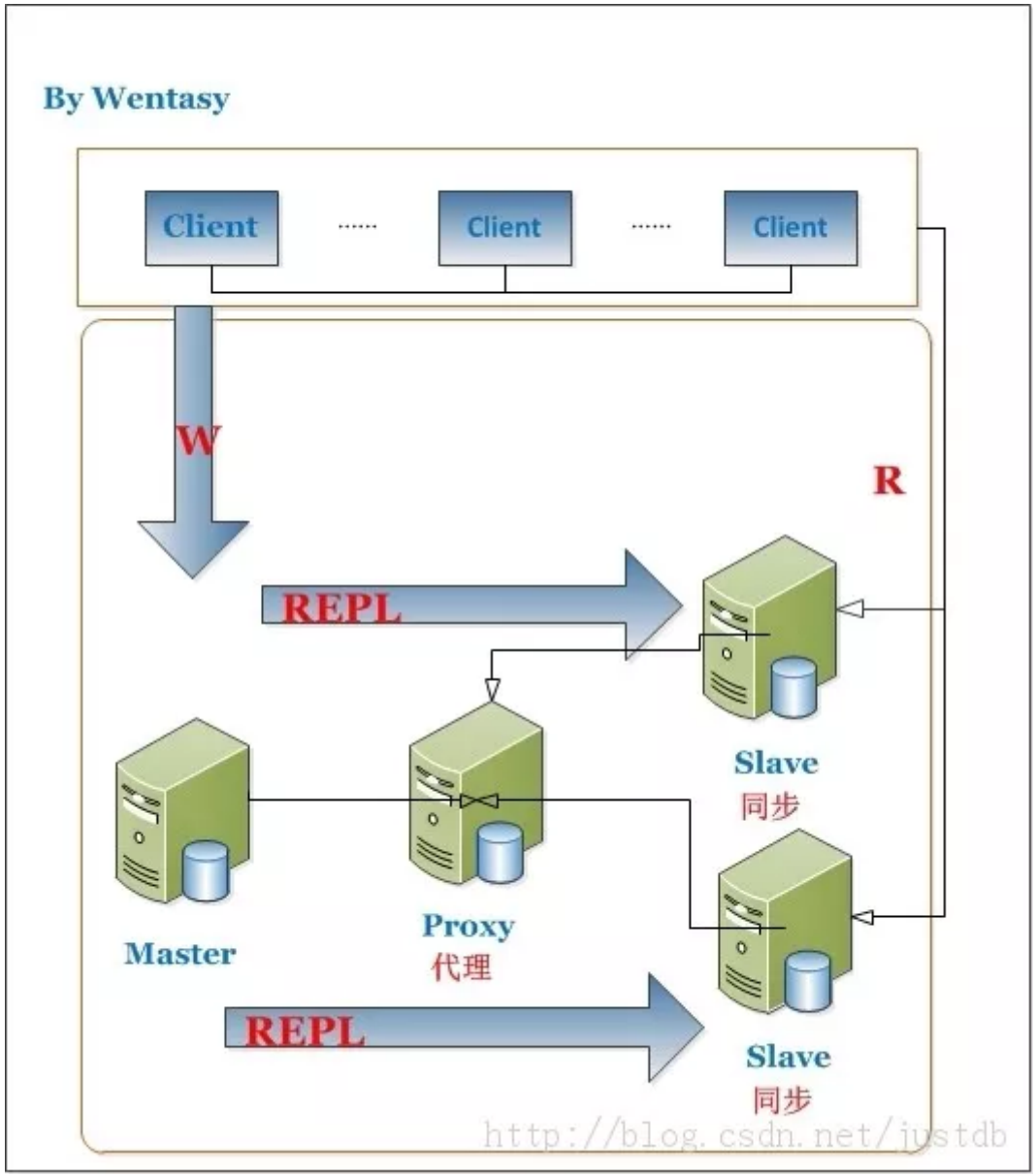
6.可以在从库启动是增加一些参数来提高其读的性能，例如--skip-innodb、--skip-bdb、--low-priority-updates以及--delay-key-write=ALL。当然这些设置也是需要根据具体业务需求来定得，不一定能用上

7.分摊读取。假如我们有1主3从，不考虑上述1中提到的从库单方面设置，假设现在1 分钟内有10条写入，150条读取。那么，1主3从相当于共计40条写入，而读取总数没变，因此平均下来每台服务器承担了10条写入和50条读取（主库不 承担读取操作）。

因此，虽然写入没变，但是读取大大分摊了，提高了系统性能。另外，当读取被分摊后，又间接提高了写入的性能。所以，总体性能提高了，说白了就是拿机器和带宽换性能。MySQL官方文档中有相关演算公式：官方文档 见6.9FAQ之“MySQL复制能够何时和多大程度提高系统性能”

8.MySQL复制另外一大功能是增加冗余，提高可用性，当一台数据库服务器宕机后能通过调整另外一台从库来以最快的速度恢复服务，因此不能光看性能，也就是说1主1从也是可以的。

四 读写分离示意图



五 读写分离模拟

实验环境简介

serv01：代理服务器 192.168.1.11 serv01.host.com

serv08：主服务器（主要写数据，可读可写） 192.168.1.18 serv08.host.com

serv09：从服务器（主要读数据） 192.168.1.19 serv09.host.com

操作系统版本

RHEL Server6.1 64位系统

使用到的软件包版本

mysql-5.5.29-linux2.6-x86_64.tar.gz

mysql-proxy-0.8.2-linux-glibc2.3-x86-64bit.tar.gz

第一步，搭建MySQL服务器，清空日志。注意：代理服务器中不需要装MySQL

第二步，拷贝mysql-proxy-0.8.2-linux-glibc2.3-x86-64bit.tar.gz文件，解压文件

```
[root@larrywen 1005]# scp /opt/soft/ule-mysql/mysql-proxy-0.8.2-linux-glibc2.3-x86-64bit.tar.gz 192.168.1.11:/opt [root@serv0
```

第三步，serv08主服务器创建用户，serv09从服务器创建用户，注意用户名和密码一致

```
serv08 mysql> grant all on *.* to 'larry'@'192.168.1.%' identified by 'larry'; Query OK, 0 rows affected (0.00 sec) serv09 mys
```

第四步，serv09从服务器更改设置，开启slave，查看slave状态。创建测试数据库，插入测试数据

```
serv09 mysql> change master to master_host='192.168.1.18', master_user='larry', master_password='larry', master_port=3306, mas
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> start slave;
Query OK, 0 rows affected (0.00 sec)
mysql> show slave status \G; *************************** 1. row *************************** Slave_IO_State: Waiting for master
mysql> create database larrydb;
Query OK, 1 row affected (0.00 sec)
mysql> use larrydb; Database changed
mysql> create table user(id int, name varchar(30));
Query OK, 0 rows affected (0.01 sec)
mysql> insert into user values(1,'larrywen');
Query OK, 1 row affected (0.01 sec)
mysql> insert into user values(2,'wentasy');
Query OK, 1 row affected (0.00 sec)
mysql> select * from user; +------+----------+ | id | name | +------+----------+ | 1 | larrywen | | 2 | wentasy | +------+----
```

第五步，为了查看现象，serv09从服务器关闭slave

```
mysql> stop slave;
Query OK, 0 rows affected (0.01 sec)
```

第六步，serv 01查看是否有MySQL用户，修改rw-splitting.lua文件，修改如下几个参数

```
[root@serv01 mysql-proxy]# id mysql uid=500(mysql) gid=500(mysql) groups=500(mysql) [root@serv01 mysql-proxy]# vim rw-splittin
[root@serv01 mysql-proxy]# cat rw-splitting.lua | grep -e min_idle_connections -e max_idle_connections -e is_debug min_idle_co
is_debug = true--是否打开Debug调试，为了查看调试信息，这里设置为true
```

第七步，启动mysql-proxy

```
[root@serv01 mysql-proxy]# /etc/init.d/mysql-proxy start Starting mysql-proxy: --先确定是否可以连接
[root@serv01 ~]# mysql -ularry -plarry -h 192.168.1.18 Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL con
mysql> exit Bye
[root@serv01 ~]# mysql -ularry -plarry -h 192.168.1.19 Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL con
mysql> exit Bye
```

第八步，查看现象

```
[root@serv01 ~]# /etc/init.d/mysql-proxy start Starting mysql-proxy:
[root@serv01 ~]# mysql -ularry -plarry -h 192.168.1.11 [connect_server] 192.168.1.11:51054 [1].connected_clients = 0 [1].pool
mysql>
mysql> use larrydb; [read_query] 192.168.1.11:51054 current backend = 0 client default db = client username = larry query = SE
mysql> select * from user;
[read_query] 192.168.1.11:51054 current backend = 0 client default db = larrydb client username = larry query = select * from
 +------+----------+
| id | name | +------+----------+
| 1 | larrywen |
| 2 | wentasy |
+------+----------+
2 rows in set (0.00 sec)
mysql> insert into user values(3,'jsutdb');
[read_query] 192.168.1.11:51644 current backend = 0 client default db = larrydb client username = larry query = insert into us
Query OK, 1 row affected (0.00 sec) serv08
mysql> select * from user;
 +------+----------+ | id | name |
+------+----------+ | 1 | larrywen |
| 2 | wentasy | +------+----------+
2 rows in set (0.00 sec) serv09
mysql> select * from larrydb.user;
+------+----------+ | id | name |
+------+----------+ | 1 | larrywen |
| 2 | wentasy |
| 3 | jsutdb |
 +------+----------+
3 rows in set (0.00 sec)
```

第九步，以上的测试虽有效果，但不是预期。排查原因，重新配置。发现proxy-read-only-backend-addresses和proxy-backend-addresses参数配置出错，proxy-read-only-backend-addresses应该配置成从服务器的IP地址，proxy-backend-addresses应该配置成主服务器的IP地址。

```
[root@serv01 ~]# vim /etc/init.d/mysql-proxy
[root@serv01 ~]# cat /etc/init.d/mysql-proxy #!/bin/sh # # mysql-proxy This script starts and stops the mysql-proxy daemon # #
```

第十步，测试。插入数据，可以发现连接的是主服务器，查询的时候也是主服务器。说明主服务器和从服务器均有读的的功能。

```
[root@serv01 ~]# mysql -ularry -plarry -h 192.168.1.11 [connect_server] 192.168.1.11:57891 [1].connected_clients = 0 [1].pool
mysql> insert into user values(5,'test');
Query OK,
1 row affected (0.01 sec)
[read_query] 192.168.1.11:57893 current backend = 0 client default db = larrydb client username = larry query = insert into us
mysql> select * from user;
+------+----------+ | id | name |
+------+----------+ | 1 | larrywen |
| 2 | wentasy |
| 5 | test | +------+----------+
3 rows in set (0.00 sec)
[read_query] 192.168.1.11:57893 current backend = 0 client default db = larrydb client username = larry query = select * from
+------+----------+ | id | name |
+------+----------+ | 1 | larrywen |
 | 2 | wentasy |
 | 5 | test | +------+----------+
3 rows in set (0.00 sec)
serv09从服务器查询数据，发现不可查询到，说明从服务器只读
 mysql>
mysql> select * from larrydb.user;
+------+----------+ | id | name |
 +------+----------+ | 1 | larrywen |
| 2 | wentasy |
| 3 | jsutdb |
 | 4 | db | +------+----------+
4 rows in set (0.00 sec)
```

第十一步，开启slave。发现数据同步成功。

```
mysql> start slave;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from larrydb.user;
+------+----------+
| id   | name     |
+------+----------+
|    1 | larrywen |
|    2 | wentasy  |
|    3 | jsutdb   |
|    4 | db       |
|    5 | test     |
+------+----------+
5 rows in set (0.00 sec)
```

文章參考：https://blog.csdn.net/samjustin1/article/details/52640125

以上是本文的全部內容，希望對你的學習有幫助，也感謝你對PHP自學中心的支持

閱讀報紙

喜歡此內容的人還喜歡

SQL發現是否存在，別再計數了，很耗費時間的！

小鹿學Java

為什麼MySQL不推薦使用join?

Java之間

2021HW參考| Linux安全應急-排查指南及命令

LemonSec