0.1 分库分表及mycat的整理实操(未整理马上更新)

```
本文章安排目标:
1. 对于存储层的压力知道如何去提供解决方案和思路
2. 对分库分表的常用手段有全面了解
3. 了解Mysql的主从及binlog
4. 知道Mycat及其他相似的中间件
5. Mycat是什么
6. Mycat中的核心概念及配置文件分析
7. 水平分表实战（单库、跨库）
8. Mycat读写分离实战
9. Mycat 全局序列号
10. Mycat常用分片规则
```

##思路(3why??)

为什么要进行分库分表

```
为什么要分库分表、
超大容量问题
性能问题
如何去做到
垂直切分、 水平切分
1. 垂直分库； 解决的是表过多的问题
2. 垂直分表； 解决单表列过多的问题

水平切分； 大数据表拆成小表
常见的拆分策略
垂直拆分（er分片）

水平拆分
一致性hash
范围切分 可以按照ID
日期拆分
拆分以后带来的问题
跨库join的问题
1. 设计的时候考虑到应用层的join问题。
2. 在服务层去做调用;
A服务里查询到一个list
for(list){
   bservice.select(list);
}
3. 全局表
1. 数据变更比较少的基于全局应用的表
2.
4. 做字段冗余（空间换时间的做法）
订单表。 商家id  商家名称
商家名称变更- 定时任务、任务通知
跨分片数据排序分页
唯一主键问题
用自增id做主键
UUID 性能比较低
snowflake
mongoDB
zookeeper
数据库表
分布式事务问题
多个数据库表之间保证原子性  性能问题； 互联网公司用强一致性分布式事务比较少

分库分表最难的在于业务的复杂度;
前提： 水平分表的前提是已经存在大量的业务数据。而这个业务数据已经渗透到了各个应用节点

如何权衡当前公司的存储需要优化
1. 提前规划（主键问题解决、 join问题）
2. 当前数据单表超过1000W、每天的增长量持续上升

Mysql的主从
```

数据库的版本5.7版本
安装以后文件对应的目录
mysql的数据文件和二进制文件： /var/lib/mysql/
mysql的配置文件： /etc/my.cnf
mysql的日志文件： /var/log/mysql.log

140 为master

1. 创建一个用户’repl’,并且允许其他服务器可以通过该用户远程访问master，通过该用户去读取二进制数据，实现数据同步
create user repl identified by ‘repl； repl用户必须具有replication slave权限，除此之外其他权限都不需要
grant replication slave on *.* to ‘repl’@’%’ identified BY ‘repl’ ;
2. 修改140 my.cnf配置文件，在[mysqld] 下添加如下配置
log-bin=mysql-bin //启用二进制日志文件
server-id=130 服务器唯一ID
3. 重启数据库 systemctl restart mysqld sudo /etc/init.d/mysql start
4. 登录到数据库，通过show master status 查看master的状态信息
142 为slave
1. 修改142 my.cnf配置文件， 在[mysqld]下增加如下配置
server-id=132 服务器id，唯一
relay-log=slave-relay-bin
relay-log-index=slave-relay-bin.index
read_only=1
2. 重启数据库： systemctl restart mysqld
3. 连接到数据库客户端，通过如下命令建立同步连接
change master to master_log_file='mysql-bin 隆.000002',master_log_pos=154;
error:
ERROR 1794 (HY000): Slave is not configured or failed to initialize properly. You must at least set --server-id to enable either a master or a slave. Additional error messages can be found in the MySQL error log.
最后通过如下的操作解决的问题，具体原因还尚未清楚
CHANGE MASTER TO
MASTER_HOST='39.107.245.253',
MASTER_USER='repl',
MASTER_PASSWORD='repl',
MASTER_LOG_FILE='mysql-bin 隆.000001',
MASTER_LOG_POS= 154;
(1)登录数据库后，删除5张表，并重新导入脚本
use mysql
drop table slave_master_info;
drop table slave_relay_log_info;
drop table slave_worker_info;
drop table innodb_index_stats;
drop table innodb_table_stats;

2.重启数据库
change master to master_host='39.107.245.253',
master_port=3306,master_user='repl',master_password='repl',master_log_file='mysql-bin 隆.000001',master_log_pos=154;
红色部分从master的show master status可以找到对应的值，不能随便写。
4. 执行 start slave
5. show slave status\G;查看slave服务器状态，当如下两个线程状态为yes，表示主从复制配置成功
Slave_IO_Running=Yes
Slave_SQL_Running=Yes
主从同步的原理

1. master记录二进制日志。在每个事务更新数据完成之前，master在二日志记录这些改变。MySQL将事务串行的写入二进制日志，即使事务中的语句都是交叉执行的。在事件写入二进制日志完成后，master通知存储引擎提交事务
2. slave将master的binary log拷贝到它自己的中继日志。首先，slave开始一个工作线程—I/O线程。I/O线程在master上打开一个普通的连接，然后开始binlog dump process。Binlog dump process从master的二进制日志中读取事件，如果已经跟上master，它会睡眠并等待master产生新的事件。I/O线程将这些事件写入中继日志
3. SQL线程从中继日志读取事件，并重放其中的事件而更新slave的数据，使其与master中的数据一致

binlog： 用来记录mysql的数据更新或者潜在更新（update xxx where id=x effect row 0）；
文件内容存储：/var/lib/mysql

mysqlbinlog --base64-output=decode-rows -v mysql-bin.000001 查看binlog的内容

binlog的格式
statement ： 基于sql语句的模式。update table set name =””; effect row 1000； uuid、now()
other function
row： 基于行模式； 存在1000条数据变更； 记录修改以后每一条记录变化的值

```
mixed: 混合模式，由mysql自动判断处理
修改binlog_formater,通过在mysql客户端输入如下命令可以修改
set global binlog_format='row/mixed/statement';
或者在vim /etc/my.cnf  的[mysqld]下增加binlog_format=‘mixed’

主从同步的延时问题
主从同步延迟是怎么产生的
1. 当master库tps比较高的时候，产生的DDL数量超过slave一个sql线程所能承受的范围，或者slave的大型query语句产生锁等待
2. 网络传输： bin文件的传输延迟
3. 磁盘的读写耗时：文件通知更新、磁盘读取延迟、磁盘写入延迟
解决方案
1. 在数据库和应用层增加缓存处理，优先从缓存中读取数据
2. 减少slave同步延迟，可以修改slave库sync_binlog属性；
sync_binlog=0  文件系统来调度把binlog_cache刷新到磁盘
sync_binlog=n
3. 增加延时监控
Nagios做网络监控
mk-heartbeat

5. Mysql的主从配置
6. 了解binlog及主从复制原理
```