# 优化与测试方案

## 1.关于优化

### 1.1什么是基线

1. 当前运行状态记录、快照，其作用是将其与未来的状态做对比，那么未来的时刻产生的关键时刻的新状态则作为下一次的状态做对比
2. 未来时刻产生关键事件后的新状态，作为下一个基线

### 1.2建立基线的作用

计算机科学中，基线是项目储存库中每个工件版本在特定时期的一个“快照”。

比如我们现在有并发事物，那么在某时刻发起一个事物会产生当前数据的快照，那么这个快照就相当理解为一个基线，那么所谓的性能数据的基线就是正常数据收集后的一段时间或者业务数据的负载，将它提供一个正式标准，随后的工作基于此标准，并且只有经过授权后才能变更这个标准。建立一个初始基线后，以后每次对其进行的变更都将记录为一个差值，直到建成下一个基线。

### 1.3基线收集，需关注的重点

1. 系统负载
2. MySQL运行状态

记录以上两个重点以便用于优化，比如做一次优化之前先记录一周的状态，比如最高、最低、平均等。

优化的工作结束之后，我们再去逐渐观察，这样我们就能知道大概情况是如何

但是有些时候有些误区，比如在优化之前tps可能1000 ，优化后可能还是1000，但是但从tps来看的话可能没有什么，但是我们还需要关注它的iops等信息有所下降，这也是一种优化，所以要多纬度的去对比。

建议收集基线数据的状态信息（需要关注的瓶颈）。

### 1.4系统层需关注的信息

* CPU：%user、%idle、%sys、%iowait #最通用的几个指标
* IO：tps、await、svctm、%util
* 内存：free（free、shared、buffers、cached）、used，以及swap
  + 内存的利用率越多越好，所以我们主要关注空闲、使用、还有swap
  + 当前是否使用swap 是否频繁使用swap 也需要关注

### 1.5数据库层需关注的信息

MySQL：tps、rt、lock、hit ratio、waits

rt = response time

lock = row lock、table lock

hit ratio = cache/buffer hit ratio

waits = Innodb\_buffer\_pool\_wait\_free / Innodb\_log\_waits / Table\_locks\_waited / Innodb\_row\_lock\_current\_waits / Innodb\_row\_lock\_waits

在找瓶颈的时候优先这几个地方进行分析，针对分析结果再定义优化方案

### 1.6工作成果量化

所谓的量化，其实无非是对比（相同条件下、相同场景下）

* 相同条件下：比如说同一套硬件，以前之能跑tps到1000，但是优化后tps能达到5000，这是很大的变化
* 相同场景下：之前的资源利用率是10% 但优化后降到了5%
* 相同业务负载下，基线数据产生变化，比如同样的用户量同样的设备，经过优化的之后，tps是降低还是提升，util、cpu等
* 相同基线数据情况下，业务数据发生变化

### 1.7建立目标，考证方案可行及总结

我们需要建立一套标准来验证我们的方法是否可行

* 建立目标
* 针对当前基线中存在的瓶颈进行优化
* 常见瓶颈以IO为主（磁盘IO、网络IO）
* 制定相对应的方案，找到优化的尝试路径
* CPU：%user、%idle、%sys、%iowait
* IO：tps、await、svctm、%util
* 内存：free（free、shared、buffers、cached）、used，以及swap
* MySQL：tps、rt、lock、hit ratio、waits

假设做的优化，针对IO 那么重点记录的是io相关的情况，然后在优化完再判断util等io相关的数值是否产生了变化。

也就是说优化工作是需要针对当前基线数据中的瓶颈进行的，通常的瓶颈，目前来说还是磁盘IO和网络IO。

### 1.8优化IO途径的建议

1. 换SSD，PCIE-SSD（提高IO效率，普通SAS盘5000以内的iops，而新设备可达到数万或者数十万iops）
2. 少做IO的活（合并多次读写为一次，或者前端加内存CACHE；或者优化业务，消除IO）
3. 加大内存（更多hot data和dirty data放在内存中，减少物理IO）
4. 调整文件系统为xfs（相比ext3、ext4提高IOPS能力，高io负载下表现更佳）
5. 调整raid级别为raid 1+0（相比raid1、raid5等提高IOPS能力）
6. 调整写cache策略为wb或force wb（利用阵列卡cache，提高iops）
7. io scheduler（优先使用deadline，如果是SSD，则使用noop

所以我们想要优化IO（磁盘IO或网络IO，）的瓶颈，通常是只要能用钱解决的事情通常就不是事情了，换个设备通常比大量时间花费在优化上要来的强。

网络IO很少，因为通常情况下mysql的tps实际能达到1-2000算是不错的，除非用高配服务器，那么这种情况下它的网络交互也不是问题，所以这种情况下网络IO一般不会成为瓶颈，如果是高配的话，一般网卡的性能要提高上去，比如多网卡绑定的方式，或者换成万兆网卡，正常情况下千兆网卡就够了。

在找瓶颈的时候优先这几个地方进行分析，针对分析结果再定义优化方案。

### 1.9通用的优化方案

* CPU：更换更好（将主频更高一些）、更多核心的CPU（主要看硬件不够用还是核心数不够用，比如频繁切换cpu），比如在在多实例的情况下，最好是有多核心的cpu，另一种，如果运算任务比较重的话这时我们还同时需要提高单CPU的运算能力，也就是提高主频
* IO：更换IOPS性能更高的设备，例如SSD，PCI-E SSD
* 内存：增加内存，合理分配
* MySQL服务：升级版本，使用Percona/MariaDB分支版本以支持更高TPS或者降低锁竞争粒度（主要是降低锁的力度，将大锁拆成小锁，这样可以观测其平衡如何，如果这些小锁拆分比较好，会带来比较大的tps的提升，或者个别参数的调整，因为有可能我们某些参数设置不当而导致tps上不去也是有可能的）

通常情况下优先升级IO设备，比如硬盘，以前的硬盘用的是7200转的盘，现在将其更换为SATA硬盘，或则直接换SSD等；内存也是一样的，直接加内存即可，这样可以有效的利用内存来缓存数据减少IOPS，提高整体TPS。

## 2.关于测试

### 2.1MySQL的压力测试

#### 2.1.1基准压力测试目的

* 采购新设备，评估新设备性能

抛去设备的新旧架构等，我们需要对其进行评估新设备的性能

* 开发新项目，评估数据库性能

新项目上线，我们需要评估新项目存在哪些瓶颈

* 新系统上线前，预估/模拟数据库负载

比如新的操作系统，比如centos7 ，那么我们想试一试换成7的话性能与以前的版本的性能差别在哪里，而且还需要模拟新项目的并发等，我们需要判断一下我们的单台服务器能否扛住多少，多台服务器能扛住多少

* 更换数据库版本，评估性能变化
* 除了性能，还需要关注可靠性

数据库运行一段时间是否稳定，是否可持续，如果发现一些异常，比如数据库不释放内存等也是非常麻烦的事情，最怕的就是内存溢出。

#### 2.1.2测试模型设计

1. 明确测试的核心目标、诉求

测试新版本性能/可靠性

测试新系统性能/可靠性

测试新机器性能/可靠性

测试新业务性能

1. 排除干扰，专注测试目的

不要跑额外的服务，导致性能受到影响

1. 确定测试环境

构建一个合理、合适、科学的测试环境，不会和现实环境差距太大，硬件、系统、配置相当

1. 确定测试过程中的衡量和变量

每一次对比测试循环中，只变更少数因素，不要一次性变更太多因素

1. 保证测试结果的可重复性

保证每个循环都至少有3次测试，每次持续至少30分钟，排除最好和最差的测试结果

#### 2.1.3压测需要注意事项

1. 不能只在本地进行压测

压力测试工具不能跟数据库在一起，因为压力测试工具本身就会产生一些负载，也会产生一些cpu或者消耗一些内存，这样就不科学，所以建议将其分割开来

1. 压测数据量小

内存可能有32G，但是压力测试数据可能有10G，这样它会可能将数据全部放在内存里面，那如果将所有数据放在内存里

1. 压测时间过短

瓶颈刚出现的时候压测已经结束了，肯定是不靠谱的

1. 压测模式太少

肯定要求有比较复杂的读写、随机读写、顺序读写等分别都要压力测试

1. 压力负载过大或过小

通常需要关注的值：

%user, %iowati, %util, svctm, iops, tps

尤其是 ：%user, %iowati, %util, svctm rt（响应时间）不要过大

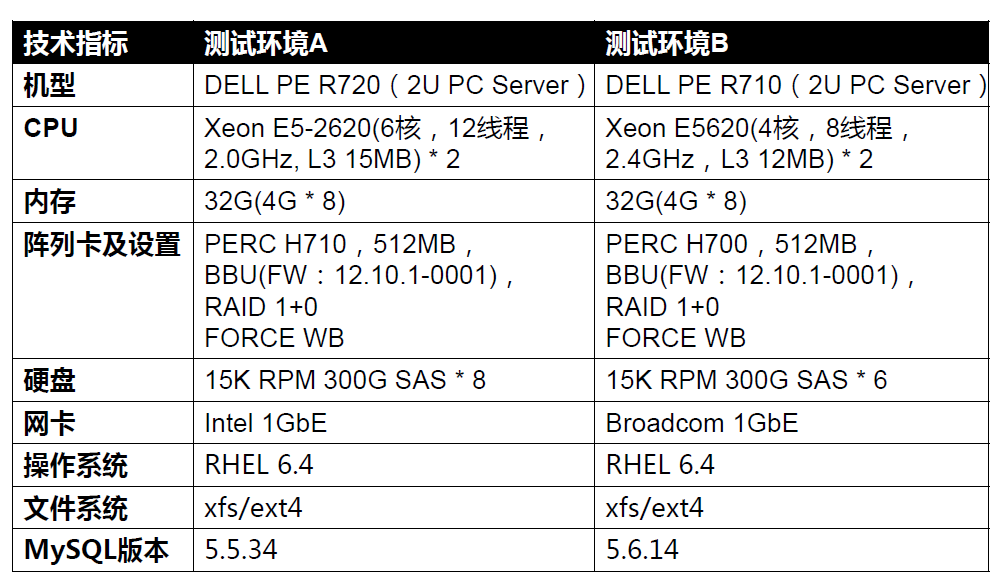
1. 每轮测试完毕要净化环境，如果有条件的话要将数据重新生成一次，如果没有条件的话无论如何要清理cache并重启mysqld：

#清除 OS cache

echo 3 > /proc/sys/vm/drop\_caches

service mysqld restart

#### 2.1.4测试环境



### 2.2MySQL测试工具的使用

#### 2.2.1常用测试工具

1. sysbench

Primarily for MySQL OLTP benchmarking，By MySQL AB

老牌测试工具，如果只做一些cpu测试倒是可以使用这个工具，模拟模式以及表的模拟稍微偏简单一些

主要针对于：cpu、threads、mutex、memory、fileio、oltp

1. tpcc-mysql（重点）

Primarily for MySQL OLTP benchmarking，By Percona

1. iozone

filesystembenchmark tool

1. tpch

Primarily for OLAP benchmarking

1. tcpcopy

模拟生产环境真实请求

* 其他
  + mysqlslap
  + sql-bench

#### 2.2.2以上工具出现的常见场景

* fio做专业的IO压力测试
* 用tpcc-mysql做MySQL的OLTP压力测试
* 用tcpcopy做在线压力模拟测试

比如我们当前业务每秒的交易数达到一千个，现在想模拟每秒交易数为一万个，一种是可以写一个脚本进行模拟，另一种使用tcpcopy将线上用户的请求引入到某个测试环境下，比如当前有10台服务器将其全部引到某个服务器上，比如以前有10台服务器而测试机只有一台服务器，现在将它全部引到某个测试服务器上，相当于压力瞬间大了10倍

或者将线上的前端服务器上的请求复制到其他服务器上然后这些前端服务器将压力施加到测试服务器上，也是可以的

#### 2.2.3sysbench

sysbench是一个模块化的、跨平台、多线程基准测试工具，主要用于评估测试各种不同系统参数下的数据库负载情况。

它主要包括以下几种方式的测试：

1. cpu性能
2. 磁盘io性能
3. 调度程序性能
4. 内存分配及传输速度
5. POSIX线程性能
6. 数据库性能(OLTP基准测试)

目前sysbench主要支持 MySQL,pgsql,oracle 这3种数据库。

##### 2.2.3.1下载源码包

安装epel包后以便安装bzr客户端：

rpm -Uvh http://dl.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm

然后就可以开始安装bzr客户端了：

yum install bzr

修改bzr使用的Python版本：

vim /usr/bin/bzr

将#!/usr/bin/python

改为#!/usr/bin/python2.6

之后，就可以开始用bzr客户端下载sysbench源码了。

cd /tmp

bzr branch lp:sysbench

MySQL中文网便捷下载地址：

<http://imysql.com/wp-content/uploads/2014/09/sysbench-0.4.12-1.1.tgz>

sysbench支持以下几种测试模式：

1. CPU运算性能
2. 磁盘IO性能
3. 调度程序性能
4. 内存分配及传输速度
5. POSIX线程性能
6. 数据库性能(OLTP基准测试)

目前sysbench主要支持 mysql,drizzle,pgsql,oracle 等几种数据库。

##### 2.2.3.2编译安装

编译非常简单，可参考 README 文档，简单步骤如下：

cd /tmp/sysbench-0.4.12-1.1

./autogen.sh

./configure --with-mysql-includes=/usr/local/mysql/include --with-mysql-libs=/usr/local/mysql/lib && make && make install

**注意**：当./autogen.sh时报错时，需安装以下依赖：

yum install m4 autoconf automake libtool

以上./configure路径要根据具体的MySQL安装路径指定，若MySQL为标准安装路径（rpm安装），则可以不指定。

# 如果 make 没有报错，就会在 sysbench 目录下生成二进制命令行工具 sysbench

ls -l sysbench

-rwxr-xr-x 1 root root 3293186 Sep 21 16:24 sysbench

##### 2.2.3.3OLTP测试前准备

初始化测试库环境（总共10个测试表，每个表 100000条记录，填充随机生成的数据）：

cd /tmp/sysbench-0.4.12-1.1/sysbench

mysqladmin create sbtest

./sysbench --mysql-host=1.2.3.4 --mysql-port=3317 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=100000 --rand-init=on prepare

关于这几个参数的解释：

--test=tests/db/oltp.lua 表示调用 tests/db/oltp.lua 脚本进行 oltp 模式测试

--oltp\_tables\_count=10 表示会生成 10 个测试表

--oltp-table-size=100000 表示每个测试表填充数据量为 100000

--rand-init=on 表示每个测试表都是用随机数据来填充的

如果在本机，也可以使用--mysql-socket指定socket文件来连接。加载测试数据时长视数据量而定，若过程比较久需要稍加耐心等待。

真实测试场景中，数据表建议不低于10个，单表数据量不低于500万行，当然了，要视服务器硬件配置而定。如果是配备了SSD或者PCIE SSD这种高IOPS设备的话，则建议单表数据量最少不低于1亿行。

##### 2.2.3.4进行OLTP测试

在上面初始化数据参数的基础上，再增加一些参数，即可开始进行测试了：

./sysbench --mysql-host=1.2.3.4. --mysql-port=3306 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=8 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=3600 \

--max-requests=0 --percentile=99 run >> ./log/sysbench\_oltpX\_8\_20140921.log

几个选项稍微解释下:

--num-threads=8 表示发起 8个并发连接

--oltp-read-only=off 表示不要进行只读测试，也就是会采用读写混合模式测试

--report-interval=10 表示每10秒输出一次测试进度报告

--rand-type=uniform 表示随机类型为固定模式，其他几个可选随机模式：uniform(固定),gaussian(高斯),special(特定的),pareto(帕累托)

--max-time=120 表示最大执行时长为 120秒

--max-requests=0 表示总请求数为 0，因为上面已经定义了总执行时长，所以总请求数可以设定为 0；也可以只设定总请求数，不设定最大执行时长

--percentile=99 表示设定采样比例，默认是 95%，即丢弃1%的长请求，在剩余的99%里取最大值

即：模拟 对10个表并发OLTP测试，每个表1000万行记录，持续压测时间为 1小时。

真实测试场景中，建议持续压测时长不小于30分钟，否则测试数据可能不具参考意义。

**注意：**测试表大小一定要和准备的表大小保持一致。

##### 2.2.3.5测试结果解读：

测试结果解读如下：

sysbench 0.5: multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 8

Report intermediate results every 10 second(s)

Random number generator seed is 0 and will be ignored

Threads started!

-- 每10秒钟报告一次测试结果，tps、每秒读、每秒写、99%以上的响应时长统计

[ 10s] threads: 8, tps: 1111.51, reads/s: 15568.42, writes/s: 4446.13, response time: 9.95ms (99%)

[ 20s] threads: 8, tps: 1121.90, reads/s: 15709.62, writes/s: 4487.80, response time: 9.78ms (99%)

[ 30s] threads: 8, tps: 1120.00, reads/s: 15679.10, writes/s: 4480.20, response time: 9.84ms (99%)

[ 40s] threads: 8, tps: 1114.20, reads/s: 15599.39, writes/s: 4456.30, response time: 9.90ms (99%)

[ 50s] threads: 8, tps: 1114.00, reads/s: 15593.60, writes/s: 4456.70, response time: 9.84ms (99%)

[ 60s] threads: 8, tps: 1119.30, reads/s: 15671.60, writes/s: 4476.50, response time: 9.99ms (99%)

OLTP test statistics:

queries performed:

read: 938224 -- 读总数

write: 268064 -- 写总数

other: 134032 -- 其他操作总数(SELECT、INSERT、UPDATE、DELETE之外的操作，例如COMMIT等)

total: 1340320 -- 全部总数

transactions: 67016 (1116.83 per sec.) -- 总事务数(每秒事务数)

deadlocks: 0 (0.00 per sec.) -- 发生死锁总数

read/write requests: 1206288 (20103.01 per sec.) -- 读写总数(每秒读写次数)

other operations: 134032 (2233.67 per sec.) -- 其他操作总数(每秒其他操作次数)

General statistics: -- 一些统计结果

total time: 60.0053s -- 总耗时

total number of events: 67016 -- 共发生多少事务数

total time taken by event execution: 479.8171s -- 所有事务耗时相加(不考虑并行因素)

response time: -- 响应时长统计

min: 4.27ms -- 最小耗时

avg: 7.16ms -- 平均耗时

max: 13.80ms -- 最长耗时

approx. 99 percentile: 9.88ms -- 超过99%平均耗时

Threads fairness:

events (avg/stddev): 8377.0000/44.33

execution time (avg/stddev): 59.9771/0.00

##### 2.2.3.6清除测试数据

./sysbench --test=tests/db/oltp.lua --oltp-tables-count=10 --num-threads=5 cleanup

##### 2.2.3.7IO测试

./sysbench --num-threads=16 --test=fileio --file-num=100 --file-total-size=3G --file-test-mode=rndrw prepare

./sysbench --num-threads=16 --test=fileio --file-num=100 --file-total-size=3G --file-test-mode=rndrw run

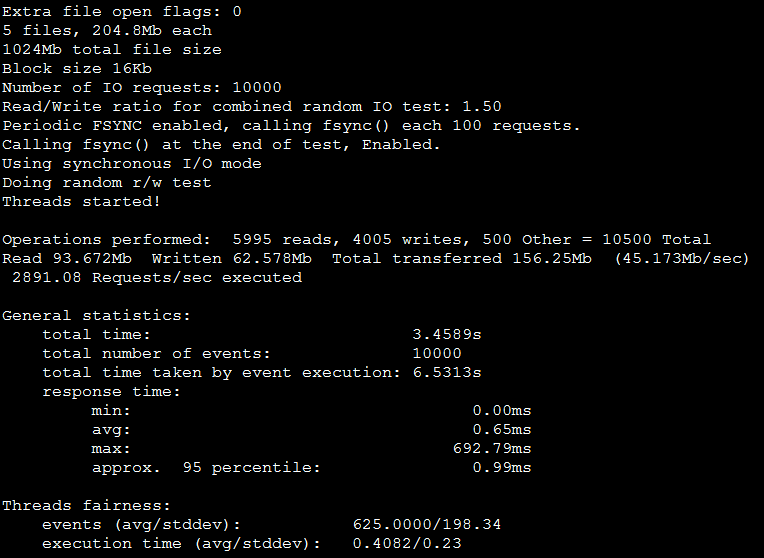
./sysbench --num-threads=16 --test=fileio --file-num=100 --file-total-size=3G --file-test-mode=rndrw cleanup

线程数16，文件数100，文件总大小3G，测试模式随机读写

测试模式如下：

|  |  |
| --- | --- |
| **seqwr** | 顺序写 |
| **seqrewr** | 顺序重写 |
| **seqrd** | 顺序读 |
| **rndrd** | 随机读 |
| **rndwr** | 随机写 |
| **rndrw** | 随机读写 |

测试结果样例：



记录45.173Mb/sec即可。

测试基准：

–file-num=100

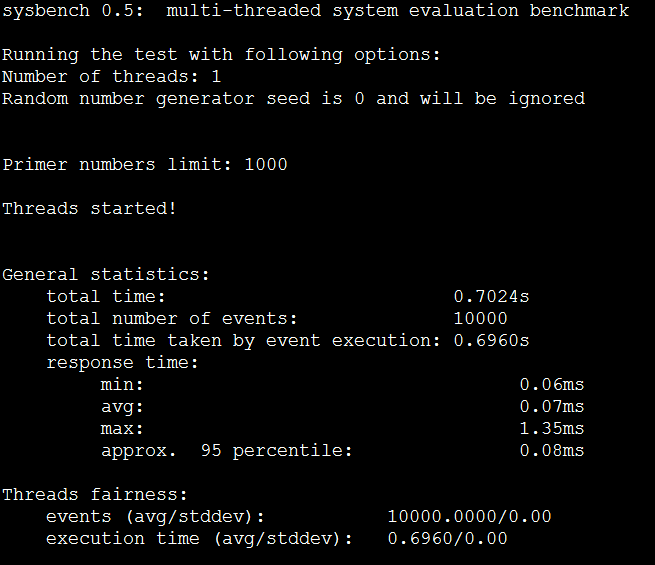
–file-total-size=物理内存~物理内存的8倍

##### 2.2.3.8cpu测试

./sysbench --test=cpu --cpu-max-prime=1000 run

计算最大素数500000

测试结果如下：



记录total time即可。

测试基准：

–cpu-max-prime=500,000

#### 2.2.4tpcc-mysql

TPC-C是专门针对联机交易处理系统（OLTP系统）的规范，一般情况下我们也把这类系统称为业务处理系统。

tpcc-mysql是percona基于TPC-C(下面简写成TPCC)衍生出来的产品，专用于MySQL基准测试。其源码放在launchpad上，用bazaar管理，项目地址：<https://code.launchpad.net/~percona-dev/perconatools/tpcc-mysql>。

##### 2.2.4.1下载源码包

安装epel包后以便安装bzr客户端：

rpm -Uvh http://dl.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm

然后就可以开始安装bzr客户端了：

yum install bzr

修改bzr使用的Python版本：

vim /usr/bin/bzr

将#!/usr/bin/python

改为#!/usr/bin/python2.6

之后，就可以开始用bzr客户端下载tpcc-mysql源码了。

cd /tmp

bzr branch lp:~percona-dev/perconatools/tpcc-mysql

MySQL中文网便捷下载地址：

<http://imysql.com/wp-content/uploads/2014/09/tpcc-mysql-src.tgz>

下载到本地后，先执行 gunzip 解压缩文件，再执行 tar xf 解包，直接 tar zxf 可能会报告异常。

tpcc-mysql的业务逻辑及其相关的几个表作用如下：

New-Order：新订单，一次完整的订单事务，几乎涉及到全部表

Payment：支付，主要对应 orders、history 表

Order-Status：订单状态，主要对应 orders、order\_line 表

Delivery：发货，主要对应 order\_line 表

Stock-Level：库存，主要对应 stock 表

其他相关表：

客户：主要对应 customer 表

地区：主要对应 district 表

商品：主要对应 item 表

仓库：主要对应 warehouse 表

##### 2.2.4.2编译安装

编译非常简单，只需要一个 make 即可。

cd /tmp/tpcc-mysql/src

make

如果 make 没有报错，就会在 /tmp/tpcc-mysql 下生成 tpcc 二进制命令行工具 tpcc\_load 、 tpcc\_start

**注意**：若MySQL不是通过rpm安装，需要将mysql\_config文件在/usr/bin/下建立软连接。

##### 2.2.4.3TPCC测试前准备

初始化测试库环境

cd /tmp/tpcc-mysql

mysqladmin create tpcc1000

mysql -f tpcc1000 < create\_table.sql

初始化完毕后，就可以开始加载测试数据了

tpcc\_load用法如下：

tpcc\_load [server] [DB] [user] [pass] [warehouse]

或者

tpcc\_load [server] [DB] [user] [pass] [warehouse] [part] [min\_wh] [max\_wh]

选项 warehouse 意为指定测试库下的仓库数量。

真实测试场景中，仓库数一般不建议少于100个，视服务器硬件配置而定，如果是配备了SSD或者PCIE SSD这种高IOPS设备的话，建议最少不低于1000个。

执行下面的命令，开始灌入测试数据：

cd /tmp/tpcc-mysql

./tpcc\_load localhost tpcc1000 tpcc\_user "tpcc\_password" 1000

在这里，需要注意的是 tpcc 默认会读取 /var/lib/mysql/mysql.sock 这个socket 文件。

因此，如果你的 socket 文件不在相应路径的话，可以做个软连接，或者通过TCP/IP的方式连接测试服务器，例如：

cd /tmp/tpcc-mysql

./tpcc\_load 1.2.3.4:3306 tpcc1000 tpcc\_user "tpcc\_password" 1000

加载测试数据时长视仓库数量而定，若过程比较久需要稍加耐心等待。

tpcc\_load其实是可以并行加载的，一方面是可以区分 ITEMS、WAREHOUSE、CUSTOMER、ORDERS 四个维度的数据并行加载。

另外，比如最终想加载1000个 warehouse的话，也可以分开成1000个并发并行加载的。看下 tpcc\_load 工具的参数就知道了：

usage: tpcc\_load [server] [DB] [user] [pass] [warehouse]

OR

tpcc\_load [server] [DB] [user] [pass] [warehouse] [part] [min\_wh] [max\_wh]

\* [part]: 1=ITEMS 2=WAREHOUSE 3=CUSTOMER 4=ORDERS

本来想自己写个并行加载脚本的，后来发现万能的github上已经有人做好了，我就直接拿来用了，tpcc\_load\_parallel.sh，加载效率至少提升10倍以上。

##### 2.2.4.4进行TPCC测试

tpcc\_start 工具用于tpcc压测，其用法如下：

tpcc\_start -h server\_host -P port -d database\_name -u mysql\_user \

-p mysql\_password -w warehouses -c connections -r warmup\_time \

-l running\_time -i report\_interval -f report\_file

几个选项稍微解释下

-w 指定仓库数量

-c 指定并发连接数

-r 指定开始测试前进行warmup的时间，进行预热后，测试效果更好

-l 指定测试持续时间

-i 指定生成报告间隔时长

-f 指定生成的报告文件名

现在我们来开启一个测试案例：

tpcc\_start -hlocalhost -d tpcc1000 -u tpcc\_user -p "tpcc\_password" \

-w 1000 -c 32 -r 120 -l 3600 \

-f tpcc\_mysql\_20140921.log >> tpcc\_caseX\_20140921.log 2>&1

即：模拟 1000个仓库规模，并发 32个线程进行测试，热身时间为 120秒, 压测时间为 1小时。

真实测试场景中，建议预热时间不小于5分钟，持续压测时长不小于30分钟，否则测试数据可能不具参考意义。

##### 2.2.4.5TPCC测试结果解读：

发起测试：

./tpcc\_start -h 1.2.3.4 -P 3306 -d tpcc10 -u tpcc -p tpcc \

-w 10 -c 64 -r 30 -l 120 \

-f tpcclog\_201409211538\_64\_THREADS.log >> tpcc\_noaid\_2\_20140921\_64.log 2>&1

测试结果输出如下：

-- 本轮tpcc压测的一些基本信息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* ###easy### TPC-C Load Generator \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

option h with value '1.2.3.4' -- 主机

option P with value '3306' -- 端口

option d with value 'tpcc10' -- 数据库

option u with value 'tpcc' -- 账号

option p with value 'tpcc' -- 密码

option w with value '10' -- 仓库数

option c with value '64' -- 并发线程数

option r with value '30' -- 数据预热时长

option l with value '120' -- 压测时长

option f with value 'tpcclog\_20140921\_64\_THREADS.res' -- 输出报告日志文件

[server]: 1.2.3.4

[port]: 3306

[DBname]: tpcc10

[user]: tpcc

[pass]: tpcc

[warehouse]: 10

[connection]: 64

[rampup]: 30 (sec.)

[measure]: 120 (sec.)

RAMP-UP TIME.(30 sec.)

-- 预热结束，开始进行压测

MEASURING START.

-- 每10秒钟输出一次压测数据

10, 8376(0):2.744|3.211, 8374(0):0.523|1.626, 838(0):0.250|0.305, 837(0):3.241|3.518, 839(0):9.086|10.676

20, 8294(0):2.175|2.327, 8292(0):0.420|0.495, 829(0):0.206|0.243, 827(0):2.489|2.593, 827(0):7.214|7.646

…

110, 8800(0):2.149|2.458, 8792(0):0.424|0.710, 879(0):0.207|0.244, 878(0):2.461|2.556, 878(0):7.042|7.341

120, 8819(0):2.147|2.327, 8820(0):0.424|0.568, 882(0):0.208|0.237, 881(0):2.483|2.561, 883(0):7.025|7.405

-- 以逗号分隔，共6列

-- 第一列，第N次10秒

-- 第二列，新订单成功执行压测的次数(推迟执行压测的次数):90%事务的响应时间|本轮测试最大响应时间，新订单事务数也被认为是总有效事务数的指标

-- 第三列，支付业务成功执行次数(推迟执行次数):90%事务的响应时间|本轮测试最大响应时间

-- 第四列，订单状态业务的结果，后面几个的意义同上

-- 第五列，物流发货业务的结果，后面几个的意义同上

-- 第六列，库存仓储业务的结果，后面几个的意义同上

-- 压测结束

STOPPING THREADS................................................................

-- 第一次结果统计

[0] sc:100589 lt:0 rt:0 fl:0 -- New-Order，新订单业务成功(success,简写sc)次数，延迟(late,简写lt)次数，重试(retry,简写rt)次数，失败(failure,简写fl)次数

[1] sc:100552 lt:0 rt:0 fl:0 -- Payment，支付业务统计，其他同上

[2] sc:10059 lt:0 rt:0 fl:0 -- Order-Status，订单状态业务统计，其他同上

[3] sc:10057 lt:0 rt:0 fl:0 -- Delivery，发货业务统计，其他同上

[4] sc:10058 lt:0 rt:0 fl:0 -- Stock-Level，库存业务统计，其他同上

in 120 sec.

-- 第二次统计结果，其他同上

[0] sc:100590 lt:0 rt:0 fl:0

[1] sc:100582 lt:0 rt:0 fl:0

[2] sc:10059 lt:0 rt:0 fl:0

[3] sc:10057 lt:0 rt:0 fl:0

[4] sc:10059 lt:0 rt:0 fl:0

(all must be [OK]) -- 下面所有业务逻辑结果都必须为 OK 才行

[transaction percentage]

Payment: 43.47% (>=43.0%) [OK] -- 支付成功次数(上述统计结果中 sc + lt)必须大于43.0%，否则结果为NG，而不是OK

Order-Status: 4.35% (>= 4.0%) [OK] -- 订单状态，其他同上

Delivery: 4.35% (>= 4.0%) [OK] -- 发货，其他同上

Stock-Level: 4.35% (>= 4.0%) [OK] -- 库存，其他同上

[response time (at least 90% passed)] -- 响应耗时指标必须超过90%通过才行

New-Order: 100.00% [OK] -- 下面几个响应耗时指标全部 100% 通过

Payment: 100.00% [OK]

Order-Status: 100.00% [OK]

Delivery: 100.00% [OK]

Stock-Level: 100.00% [OK]

50294.500 TpmC -- TpmC结果值（每分钟事务数，该值是第一次统计结果中的新订单事务数除以总耗时分钟数，例如本例中是：100589/2 = 50294.500）

script目录下的一些脚本主要是一些性能数据采集以及分析的，可以自行摸索下怎么用。

### 2.3MySQL压力测试基准值

通常，我们会出于以下几个目的对MySQL进行压力测试：

1. 确认新的MySQL版本性能相比之前差异多大，比如从5.6变成5.7，或者从官方版本改成Percona分支版本；
2. 确认新的服务器性能是否更高，能高多少，比如CPU升级了、阵列卡cache加大了、从机械盘换成SSD盘了；
3. 确认一些新的参数调整后，对性能影响多少，比如 innodb\_flush\_log\_at\_trx\_commit、sync\_binlog 等参数；
4. 确认即将上线的新业务对MySQL负载影响多少，是否能承载得住，是否需要对服务器进行扩容或升级配置；

针对上面这几种压测的目的，相应的测试方法也有所不同。

先说第四种，需要和线上业务结合起来，这时候就需要自行开发测试工具，或者利用 tcpcopy 将线上实际用户请求导向测试环境，进行仿真模拟测试。

对于前三种，我们通常采用基准测试就可以。比较常用的MySQL基准压力测试工具有 tpcc-mysql、sysbench、mysqlslap 等几个。

基于促进同行间的交流，统一MySQL压测标准，并且可以相互分享、对比、借鉴测试结果的目的。因此老叶特别发起MySQL压力测试基准值倡议。建议大家采用以下几种压力测试基准值。



关于压力测试的其他几个方面：

1、如何避免压测时受到缓存的影响

a、填充测试数据比物理内存还要大，至少超过 innodb\_buffer\_pool\_size值，不能将数据全部装载到内存中，除非你的本意就想测试全内存状态下的MySQL性能。

b、每轮测试完成后，都重启mysqld实例，并且用下面的方法删除系统cache，释放swap（如果用到了swap的话），甚至可以重启整个OS。

[root@imysql.com]# sync -- 将脏数据刷新到磁盘

[root@imysql.com]# echo 3 > /proc/sys/vm/drop\_caches -- 清除OS Cache

[root@imysql.com]# swapoff -a && swapon –a

## 关于监控

在测试时可以使用sar命令对系统进行监控。

sar（System Activity Reporter系统活动情况报告）是目前 Linux 上最为全面的系统性能分析工具之一，可以从多方面对系统的活动进行报告，包括：文件的读写情况、系统调用的使用情况、磁盘I/O、CPU效率、内存使用状况、进程活动及IPC有关的活动等。

安装：

yum install sysstat

sar命令常用格式

sar [options] [-A] [-o file] t [n]

其中：

t为采样间隔，默认值是1，n为采样次数

-o file表示将命令结果以二进制格式存放在文件中，file是文件名。

options 为命令行选项，sar命令常用选项如下：

-A：所有报告的总和

-u：输出CPU使用情况的统计信息

-v：输出inode、文件和其他内核表的统计信息

-d：输出每一个块设备的活动信息

-r：输出内存和交换空间的统计信息

-b：显示I/O和传送速率的统计信息

-a：文件读写情况

-c：输出进程统计信息，每秒创建的进程数

-R：输出内存页面的统计信息

-y：终端设备活动情况

-w：输出系统交换活动信息

-n：输出网络信息

### 3.1CPU资源监控

例如，每10秒采样一次，连续采样3次，观察CPU 的使用情况，并将采样结果以二进制形式存入当前目录下的文件test中，需键入如下命令：

sar -u -o test 10 3

屏幕显示如下：

17:06:16 CPU %user %nice %system %iowait %steal %idle

17:06:26 all 0.00 0.00 0.20 0.00 0.00 99.80

17:06:36 all 0.00 0.00 0.20 0.00 0.00 99.80

17:06:46 all 0.00 0.00 0.10 0.00 0.00 99.90

Average: all 0.00 0.00 0.17 0.00 0.00 99.80

输出项说明：

CPU：all 表示统计信息为所有 CPU 的平均值。

%user：显示在用户级别(application)运行使用 CPU 总时间的百分比。

%nice：显示在用户级别，用于nice操作，所占用 CPU 总时间的百分比。

%system：在核心级别(kernel)运行所使用 CPU 总时间的百分比。

%iowait：显示用于等待I/O操作占用 CPU 总时间的百分比。

%steal：管理程序(hypervisor)为另一个虚拟进程提供服务而等待虚拟 CPU 的百分比。

%idle：显示 CPU 空闲时间占用 CPU 总时间的百分比。

1. 若 %iowait 的值过高，表示硬盘存在I/O瓶颈

2. 若 %idle 的值高但系统响应慢时，有可能是 CPU 等待分配内存，此时应加大内存容量

3. 若 %idle 的值持续低于10，则系统的 CPU 处理能力相对较低，表明系统中最需要解决的资源是 CPU 。

如果要查看二进制文件test中的内容，需键入如下sar命令：

sar -u -f test

### 3.2inode、文件和其他内核表监控

例如，每10秒采样一次，连续采样3次，观察核心表的状态，需键入如下命令：

sar -v 10 3

屏幕显示如下：

17:10:49 dentunusd file-nr inode-nr pty-nr

17:10:59 6301 5664 12037 4

17:11:09 6301 5664 12037 4

17:11:19 6301 5664 12037 4

Average: 6301 5664 12037 4

输出项说明：

dentunusd：目录高速缓存中未被使用的条目数量

file-nr：文件句柄（file handle）的使用数量

inode-nr：索引节点句柄（inode handle）的使用数量

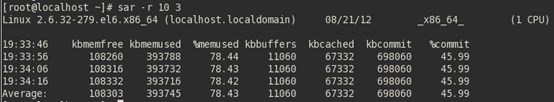
pty-nr：使用的pty数量

### 3.3内存和交换空间监控

例如，每10秒采样一次，连续采样3次，监控内存分页：

sar -r 10 3

屏幕显示如下：



输出项说明：

kbmemfree：这个值和free命令中的free值基本一致,所以它不包括buffer和cache的空间.

kbmemused：这个值和free命令中的used值基本一致,所以它包括buffer和cache的空间.

%memused：这个值是kbmemused和内存总量(不包括swap)的一个百分比.

kbbuffers和kbcached：这两个值就是free命令中的buffer和cache.

kbcommit：保证当前系统所需要的内存,即为了确保不溢出而需要的内存(RAM+swap).

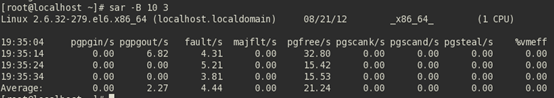
%commit：这个值是kbcommit与内存总量(包括swap)的一个百分比.

### 3.4内存分页监控

例如，每10秒采样一次，连续采样3次，监控内存分页：

sar -B 10 3

屏幕显示如下：



输出项说明：

pgpgin/s：表示每秒从磁盘或SWAP置换到内存的字节数(KB)

pgpgout/s：表示每秒从内存置换到磁盘或SWAP的字节数(KB)

fault/s：每秒钟系统产生的缺页数,即主缺页与次缺页之和(major + minor)

majflt/s：每秒钟产生的主缺页数.

pgfree/s：每秒被放入空闲队列中的页个数

pgscank/s：每秒被kswapd扫描的页个数

pgscand/s：每秒直接被扫描的页个数

pgsteal/s：每秒钟从cache中被清除来满足内存需要的页个数

%vmeff：每秒清除的页(pgsteal)占总扫描页(pgscank+pgscand)的百分比

### 3.5I/O和传送速率监控

例如，每10秒采样一次，连续采样3次，报告缓冲区的使用情况，需键入如下命令：

sar -b 10 3

屏幕显示如下：

18:51:05 tps rtps wtps bread/s bwrtn/s

18:51:15 0.00 0.00 0.00 0.00 0.00

18:51:25 1.92 0.00 1.92 0.00 22.65

18:51:35 0.00 0.00 0.00 0.00 0.00

Average: 0.64 0.00 0.64 0.00 7.59

输出项说明：

tps：每秒钟物理设备的 I/O 传输总量

rtps：每秒钟从物理设备读入的数据总量

wtps：每秒钟向物理设备写入的数据总量

bread/s：每秒钟从物理设备读入的数据量，单位为 块/s

bwrtn/s：每秒钟向物理设备写入的数据量，单位为 块/s

### 3.6进程队列长度和平均负载状态监控

例如，每10秒采样一次，连续采样3次，监控进程队列长度和平均负载状态：

sar -q 10 3

屏幕显示如下：

19:25:50 runq-sz plist-sz ldavg-1 ldavg-5 ldavg-15

19:26:00 0 259 0.00 0.00 0.00

19:26:10 0 259 0.00 0.00 0.00

19:26:20 0 259 0.00 0.00 0.00

Average: 0 259 0.00 0.00 0.00

输出项说明：

runq-sz：运行队列的长度（等待运行的进程数）

plist-sz：进程列表中进程（processes）和线程（threads）的数量

ldavg-1：最后1分钟的系统平均负载（System load average）

ldavg-5：过去5分钟的系统平均负载

ldavg-15：过去15分钟的系统平均负载

### 3.7系统交换活动信息监控

例如，每10秒采样一次，连续采样3次，监控系统交换活动信息：

sar - W 10 3

屏幕显示如下：

19:39:50 pswpin/s pswpout/s

19:40:00 0.00 0.00

19:40:10 0.00 0.00

19:40:20 0.00 0.00

Average: 0.00 0.00

输出项说明：

pswpin/s：每秒系统换入的交换页面（swap page）数量

pswpout/s：每秒系统换出的交换页面（swap page）数量

### 3.8设备使用情况监控

例如，每10秒采样一次，连续采样3次，报告设备使用情况，需键入如下命令：

# sar -d 10 3 –p

屏幕显示如下：

17:45:54 DEV tps rd\_sec/s wr\_sec/s avgrq-sz avgqu-sz await svctm %util

17:46:04 scd0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

17:46:04 sda 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

17:46:04 vg\_livedvd-lv\_root 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

17:46:04 vg\_livedvd-lv\_swap 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

其中：

参数-p可以打印出sda,hdc等磁盘设备名称,如果不用参数-p,设备节点则有可能是dev8-0,dev22-0

tps:每秒从物理磁盘I/O的次数.多个逻辑请求会被合并为一个I/O磁盘请求,一次传输的大小是不确定的.

rd\_sec/s:每秒读扇区的次数.

wr\_sec/s:每秒写扇区的次数.

avgrq-sz:平均每次设备I/O操作的数据大小(扇区).

avgqu-sz:磁盘请求队列的平均长度.

await:从请求磁盘操作到系统完成处理,每次请求的平均消耗时间,包括请求队列等待时间,单位是毫秒(1秒=1000毫秒).

svctm:系统处理每次请求的平均时间,不包括在请求队列中消耗的时间.

%util:I/O请求占CPU的百分比,比率越大,说明越饱和.

1. avgqu-sz 的值较低时，设备的利用率较高。

2. 当%util的值接近 100% 时，表示设备带宽已经占满。

要判断系统瓶颈问题，有时需几个 sar 命令选项结合起来

怀疑CPU存在瓶颈，可用 sar -u 和 sar -q 等来查看

怀疑内存存在瓶颈，可用 sar -B、sar -r 和 sar -W 等来查看

怀疑I/O存在瓶颈，可用 sar -b、sar -u 和 sar -d 等来查看

### 3.9网络资源监控

# sar -n DEV | more

Linux 2.6.9-42.0.3.ELlargesmp (prolin3) 12/27/2008

12:00:01 AM IFACE rxpck/s txpck/s rxbyt/s txbyt/s rxcmp/s txcmp/s rxmcst/s

12:10:01 AM lo 4.54 4.54 782.08 782.08 0.00 0.00 0.00

12:10:01 AM eth0 2.70 0.00 243.24 0.00 0.00 0.00 0.99

12:10:01 AM eth1 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:10:01 AM eth2 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:10:01 AM eth3 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:10:01 AM eth4 143.79 141.14 73032.72 38273.59 0.00 0.00 0.99

12:10:01 AM eth5 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:10:01 AM eth6 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:10:01 AM eth7 0.00 0.00 0.00 0.00 0.00 0.00 0.00

12:10:01 AM bond0 146.49 141.14 73275.96 38273.59 0.00 0.00 1.98

… and so on …

Average: bond0 128.73 121.81 85529.98 27838.44 0.00 0.00 1.98

Average: eth8 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Average: eth9 3.52 6.74 251.63 10179.83 0.00 0.00 0.00

Average: sit0 0.00 0.00 0.00 0.00 0.00 0.00 0.00

## 4.关于本次测试方案

### 4.1测试目的

评估现有数据库的单点、集群性能，找出性能瓶颈，建立基线，为后续优化做准备，测试MHA功能，测试复制的一致性、延迟性，测试Atlas的功能及性能。

### 4.2测试模型设计

1. 确定测试环境
2. 分别使用sysbench和tpcc-mysql对MySQL进行测试
3. 使用sysbench对cpu、IO进行测试
4. 测试时不跑额外的服务
5. 测试程序与数据库在不同的服务器上
6. 每一次对比测试循环中，只变更少数因素，不要一次性变更太多因素
7. 保证每个循环都至少有3次测试，每次持续至少30分钟，排除最好和最差的测试结果
8. 记录测试数据，以excel方式整理测试报告

### 4.3注意事项

1. 不能只在本地进行压测

压力测试工具不能跟数据库在一起，因为压力测试工具本身就会产生一些负载，也会产生一些cpu或者消耗一些内存，这样就不科学，所以建议将其分割开来

1. 压测数据量小

内存可能有32G，但是压力测试数据可能有10G，这样它会可能将数据全部放在内存里面，那如果将所有数据放在内存里

1. 压测时间过短

瓶颈刚出现的时候压测已经结束了，肯定是不靠谱的

1. 压测模式太少

肯定要求有比较复杂的读写、随机读写、顺序读写等分别都要压力测试

1. 压力负载过大或过小

通常需要关注的值：

%user, %iowati, %util, svctm, iops, tps

尤其是 ：%user, %iowati, %util, svctm rt（响应时间）不要过大

1. 每轮测试完毕要净化环境，如果有条件的话要将数据重新生成一次，如果没有条件的话无论如何要清理cache并重启mysqld：

#清除 OS cache

echo 3 > /proc/sys/vm/drop\_caches

service mysqld restart

### 4.4MySQL单点性能基线测试

#### 4.4.1测试场景

针对单点MySQL服务器，使用sysbench测试MySQL性能基线。测试时对CPU、IO进行监控，找出性能瓶颈。对文件系统ext3、ext4、XFS分别经行测试，比较性能。

#### 4.4.2测试环境

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| MySQL IP | 10.51.23.196:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | ext3/ext4/XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |

#### 4.4.3配置参数

#innodb

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

#buffer

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 1M

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 500

memlock = off

#### 4.4.4测试脚本

在数据库添加测试用的账户tpcc：

grant all on sbtest.\* to tpcc@'%' identified by 'tpcc';

测试前准备测试数据：

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

开始测试：

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

测试基准：

测试工具： sysbench

测试表数据行： 10,000,000（24G）

测试模式： complex

测试表引擎： innodb

测试表数量： 10个

测试时间： 30分钟

并发线程数：16、32、64、128、256

在测试中期对CPU、IO进行监控

CPU监控：

sar -u -o test 120 5

IO监控：

sar -d -o test 120 5

#### 4.4.5测试结果

Ext3：

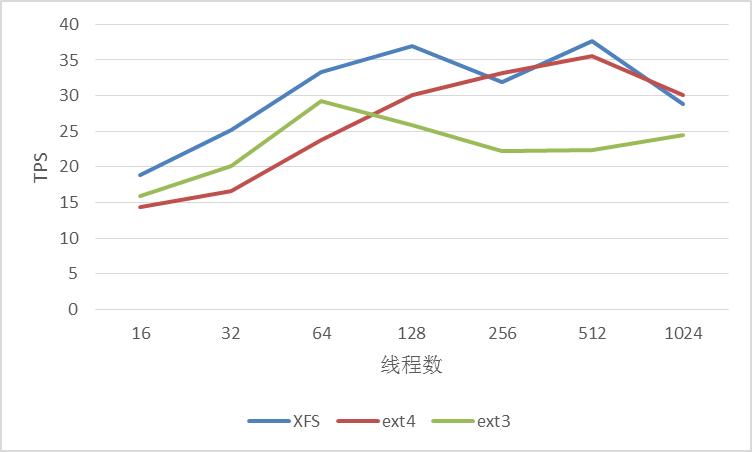
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 16 | 15.94 | 286.86 | 1784.61 | 1.56 | 13.00 | 84.81 | 102.74 |
| 32 | 20.08 | 361.38 | 3173.47 | 1.78 | 18.70 | 78.82 | 102.66 |
| 64 | 29.29 | 527.26 | 4183.37 | 2.31 | 31.46 | 65.29 | 102.76 |
| 128 | 25.88 | 465.80 | 29102.79 | 2.54 | 41.97 | 54.51 | 102.81 |
| 256 | 22.28 | 401.04 | 86576.37 | 2.33 | 41.78 | 54.69 | 102.78 |
| 512 | 22.38 | 402.83 | 146403.57 | 2.52 | 43.42 | 52.52 | 102.60 |
| 1024 | 24.48 | 440.72 | 256014.22 | 2.90 | 51.59 | 43.67 | 102.99 |

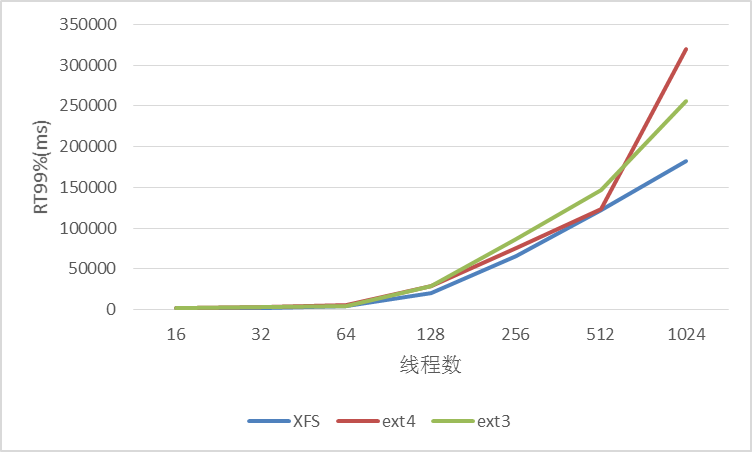
Ext4：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 16 | 14.35 | 258.23 | 1962.83 | 0.93 | 12.77 | 85.95 | 101.69 |
| 32 | 16.65 | 353.69 | 3033.23 | 1.11 | 16.18 | 82.28 | 101.87 |
| 64 | 23.71 | 426.72 | 6206.80 | 1.45 | 25.79 | 72.72 | 101.75 |
| 128 | 30.11 | 542.07 | 29488.64 | 1.61 | 34.31 | 63.48 | 101.75 |
| 256 | 33.20 | 597.53 | 75349.22 | 1.76 | 36.59 | 60.89 | 101.80 |
| 512 | 35.59 | 640.57 | 123438.58 | 2.44 | 60.28 | 35.85 | 102.10 |
| 1024 | 30.02 | 540.41 | 319400.10 | 2.11 | 51.06 | 45.56 | 101.93 |

XFS：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 16 | 18.88 | 339.83 | 1840.49 | 1.11 | 14.08 | 84.39 | 101.67 |
| 32 | 25.18 | 453.30 | 2495.42 | 1.41 | 17.94 | 80.08 | 101.78 |
| 64 | 33.37 | 600.61 | 3872.45 | 1.76 | 30.08 | 67.47 | 102.28 |
| 128 | 36.90 | 664.22 | 20657.76 | 2.13 | 43.84 | 53.26 | 102.30 |
| 256 | 31.90 | 575.15 | 65578.00 | 2.19 | 54.01 | 42.87 | 102.09 |
| 512 | 37.59 | 676.59 | 122812.03 | 2.02 | 48.78 | 48.17 | 101.96 |
| 1024 | 28.86 | 519.42 | 181941.85 | 2.33 | 48.20 | 47.92 | 101.83 |





#### 4.4.6结果分析

数据库性能很低，瓶颈在于磁盘IO上，文件系统优劣：XFS>ext4>ext3，但性能提升并不十分明显，数据库支持最大并发量在512左右。

### 4.5MySQL配置参数优化测试

#### 4.5.1测试场景

以4.4单点性能基线测试为基础，在文件系统XFS下，通过tuning-primer脚本检查需要优化的参数，调整MySQL配置参数，对比MySQL数据库性能。

#### 4.5.2测试环境

MySQL主从配置相同

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| MySQL IP | 10.252.124.43:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |

#### 4.5.3配置参数

原始参数：

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 1M

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 500

memlock = off

调整参数：

query\_cache\_size = 0

#### 4.5.4测试脚本

在数据库添加测试用的账户tpcc：

grant all on sbtest.\* to tpcc@'%' identified by 'tpcc';

测试前准备测试数据：

./sysbench --mysql-host=10.252.124.43 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

开始测试：

./sysbench --mysql-host=10.252.124.43 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

测试基准：

测试工具： sysbench

测试表数据行： 10,000,000

测试模式： complex

测试表引擎： innodb

测试表数量： 10个

测试时间： 30分钟

并发线程数： 128

#### 4.5.5测试结果

原始结果：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 128 | 36.90 | 664.22 | 20657.76 | 2.13 | 43.84 | 53.26 | 102.30 |

优化后结果：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 128 | 32.65 | 587.69 | 21567.68 | 2.20 | 43.23 | 53.71 | 102.28 |

#### 4.5.6结果分析

数据库瓶颈主要在于IO，参数优化并无明显提升性能，对于参数的优化还要根据生产上实际应用运行状态进一步优化。

### 4.6MySQL复制一致性、延迟性测试

#### 4.6.1测试场景

使用sysbench对主节点master1进行压力测试的同时，使用pt-table-checksum对复制一致性进行检查，使用pt-heartbeat对复制延迟进行检查。

#### 4.5.2测试环境

MySQL主从配置相同

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| Master1 IP | 10.51.23.196:3316 |
| Master2 IP | 10.252.124.43:3316 |
| Slave IP | 10.168.6.6:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |

#### 4.5.3配置参数

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 0

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 500

memlock = on

#### 4.5.4测试脚本

在数据库添加测试用的账户tpcc：

grant all on sbtest.\* to tpcc@'%' identified by 'tpcc';

测试前准备测试数据：

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

开始测试：

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

测试基准：

测试工具： sysbench

测试表数据行： 10,000,000

测试模式： complex

测试表引擎： innodb

测试表数量： 10个

测试时间： 30分钟

并发线程数： 128

在master1上进行复制延迟测试：

pt-heartbeat -D test --update --create-table -h localhost --daemonize

pt-heartbeat -D test --check -h 10.252.124.43 -P 3316 -u tpcc -ptpcc --master-server-id 1

pt-heartbeat -D test --check -h 10.168.6.6 -P 3316 -u tpcc -ptpcc --master-server-id 1

在master1上进行复制一致性检查，需要等待压测结束，在percona数据库中创建dsns表，并插入dsn值：

CREATE TABLE `dsns` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`parent\_id` int(11) DEFAULT NULL,

`dsn` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

mysql> insert into dsns(dsn) values('h=10.252.124.43,u=root,p=Root!1');

mysql> insert into dsns(dsn) values('h=10.168.6.6,u=root,p=Root!1');

pt-table-checksum -d sbtest --recursion-method dsn=h=localhost,D=percona,t=dsns --no-check-binlog-format --nocheck-replication-filters

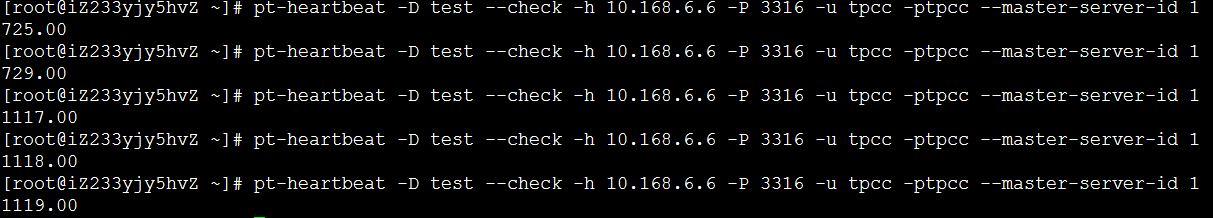
pt-table-checksum -d rkylin --recursion-method dsn=h=localhost,D=percona,t=dsns --no-check-binlog-format --nocheck-replication-filters --replicate-check-only

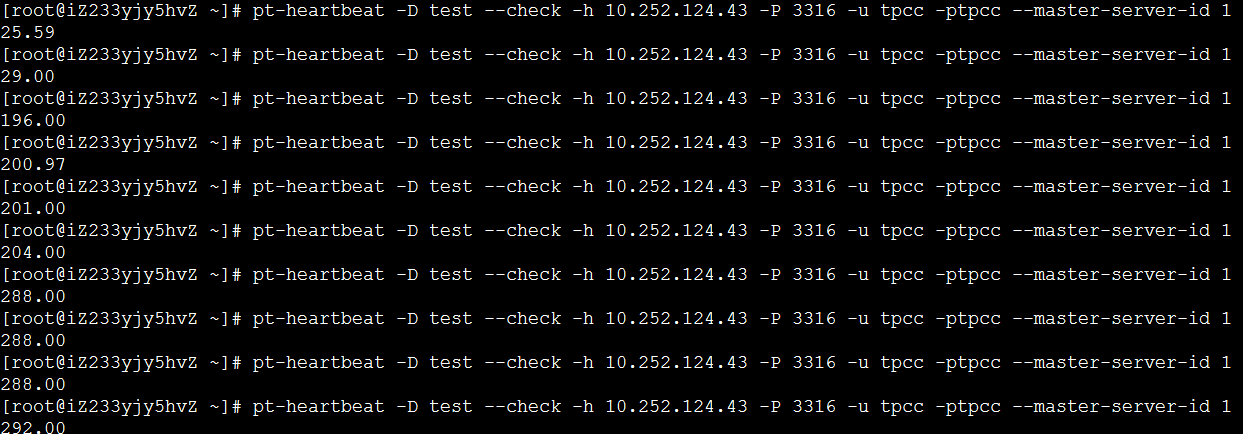
恢复脚本：

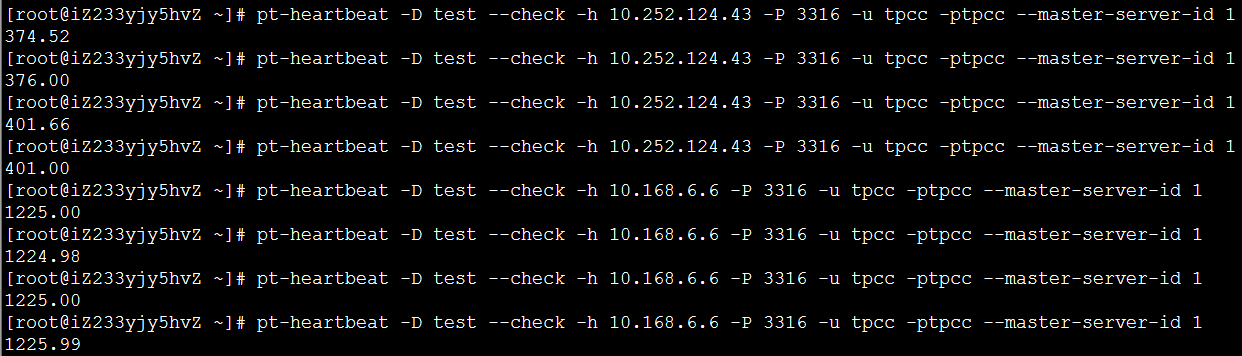
pt-table-sync --execute --verbose --print --replicate percona.checksums h=localhost

#### 4.5.5测试结果

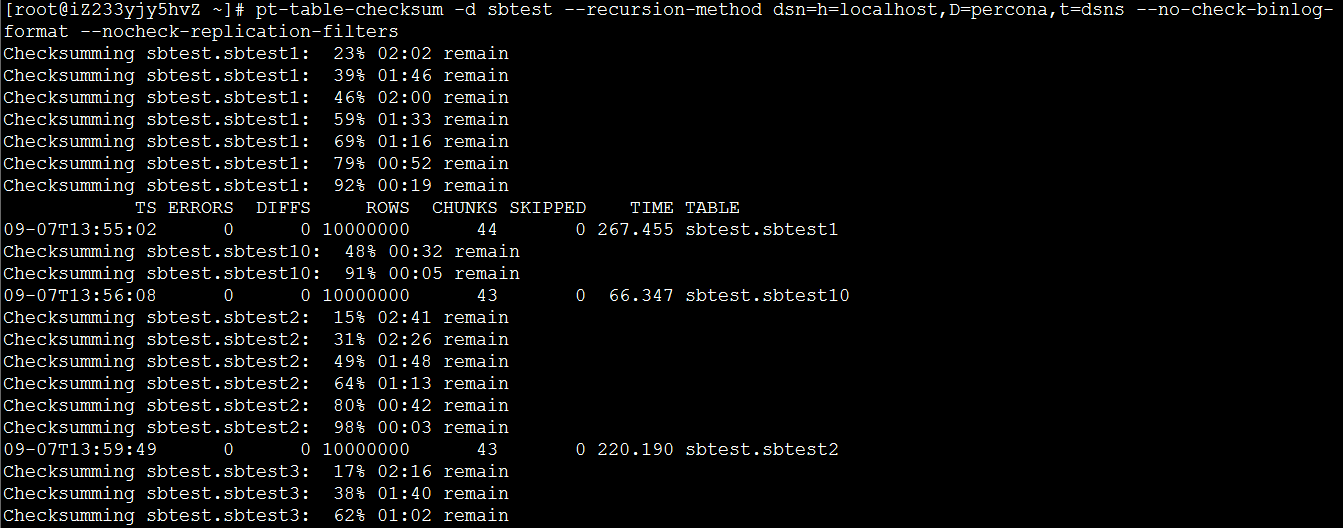
复制延迟结果：

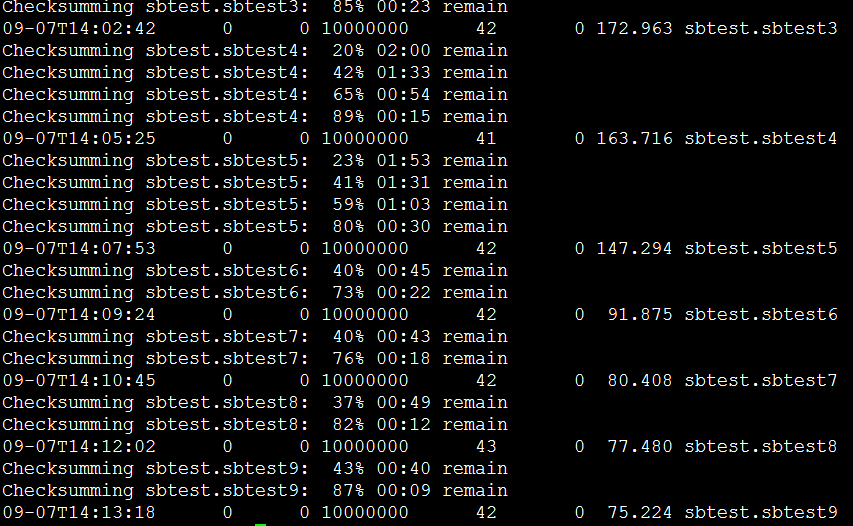






复制一致性结果：





#### 4.5.6结果分析

压测中，复制延迟会不断增大，半同步延迟为异步复制延迟的1/3左右。分析原因在于从服务器的IO能力比备的弱，主从复制保持了一致。

### 4.7MySQL复制性能基线测试

#### 4.7.1测试场景

针对MySQL复制架构，使用sysbench测试MySQL性能基线。测试时对CPU、IO进行监控，找出性能瓶颈。与单点性能基线进行比较。

#### 4.7.2测试环境

MySQL主从配置相同

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| Master1 IP | 10.51.23.196:3316 |
| Master2 IP | 10.252.124.43:3316 |
| Slave IP | 10.168.6.6:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |

#### 4.7.3配置参数

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 0

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 1100

memlock = on

#### 4.7.4测试脚本

在数据库添加测试用的账户tpcc：

grant all on sbtest.\* to tpcc@'%' identified by 'tpcc';

测试前准备测试数据：

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

开始测试：

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

测试基准：

测试工具： sysbench

测试表数据行： 10,000,000

测试模式： complex

测试表引擎： innodb

测试表数量： 10个

测试时间： 30分钟

并发线程数：16、32、64、128、256、512、1024

在测试中期对CPU、IO进行监控

CPU监控：

sar -u -o test 120 5

IO监控：

sar -d -o test 120 5

#### 4.7.5测试结果

单点性能结果：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 16 | 18.88 | 339.83 | 1840.49 | 1.11 | 14.08 | 84.39 | 101.67 |
| 32 | 25.18 | 453.30 | 2495.42 | 1.41 | 17.94 | 80.08 | 101.78 |
| 64 | 33.37 | 600.61 | 3872.45 | 1.76 | 30.08 | 67.47 | 102.28 |
| 128 | 36.90 | 664.22 | 20657.76 | 2.13 | 43.84 | 53.26 | 102.30 |
| 256 | 31.90 | 575.15 | 65578.00 | 2.19 | 54.01 | 42.87 | 102.09 |
| 512 | 37.59 | 676.59 | 122812.03 | 2.02 | 48.78 | 48.17 | 101.96 |
| 1024 | 28.86 | 519.42 | 181941.85 | 2.33 | 48.20 | 47.92 | 101.83 |

复制性能结果：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| 16 | 14.79 | 266.13 | 2222.46 | 0.95 | 14.13 | 84.57 | 101.61 |
| 32 | 18.00 | 324.04 | 3428.27 | 1.08 | 17.59 | 80.91 | 101.64 |
| 64 | 25.69 | 462.33 | 4813.90 | 1.59 | 27.26 | 70.53 | 101.99 |
| 128 | 29.31 | 527.57 | 21076.24 | 1.92 | 43.66 | 53.71 | 101.78 |
| 256 | 23.97 | 431.45 | 66686.55 | 2.64 | 61.37 | 35.03 | 102.30 |
| 512 | 29.80 | 536.46 | 127113.31 | 2.04 | 49.84 | 46.95 | 101.90 |
| 1024 | 26.47 | 476.38 | 247499.28 | 2.28 | 47.24 | 49.32 | 101.59 |

#### 4.7.6结果分析

复制的性能要略低于单点的性能，和推测的结果一样。

### 4.8MHA(无VIP)自动切换功能及数据一致性测试

#### 4.7.1测试场景

针对MySQL高可用MHA架构，在无keepalived情况下，模拟主库宕机，实现MHA自动切换，对切换后的数据库进行一致性验证。

#### 4.7.2测试环境

MySQL主从配置相同

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| Master1 IP | 10.51.23.196:3316 |
| Master2 IP | 10.252.124.43:3316 |
| Slave IP | 10.168.6.6:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |
| MHA manager IP | 120.26.127.218 |

#### 4.7.3配置参数

MHA配置参数：

[server default]

manager\_log=/mydata/masterha/manager.log

manager\_workdir=/mydata/masterha/app1

master\_binlog\_dir=/mydata/mysql/binlog

#master\_ip\_failover\_script=/usr/local/bin/master\_ip\_failover

master\_ip\_online\_change\_script=/usr/local/bin/master\_ip\_online\_change

password="Rkylin\_web!1"

ping\_interval=1

remote\_workdir=/mydata/masterha/binlog

repl\_password="Slaveadmin!1"

repl\_user="slaveadmin"

report\_script=/usr/local/bin/send\_report

ssh\_user=root

user="rkylinadmin"

[server1]

candidate\_master=1

check\_repl\_delay=0

hostname=10.51.23.196

master\_binlog\_dir=/mydata/mysql/binlog

port=3316

[server2]

candidate\_master=1

check\_repl\_delay=0

hostname=10.252.124.43

master\_binlog\_dir=/mydata/mysql/binlog

port=3316

[server3]

hostname=10.168.6.6

master\_binlog\_dir=/mydata/mysql/binlog

port=3316

数据库配置：

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 0

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 1100

memlock = on

#### 4.7.4测试步骤

必须先启动MHA Manager，否则无法自动切换，当然手动切换不需要开启MHA Manager监控。

nohup masterha\_manager --conf=/etc/app1.cnf --remove\_dead\_master\_conf --ignore\_last\_failover < /dev/null > /mydata/masterha/app1/manager.log 2>&1 &

步骤如下：

（1）使用sysbench生成测试数据

在主库（10.51.23.196）上进行sysbench数据生成，在sbtest库下生成sbtest表，共1000W记录。

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

（2）停掉slave sql线程，模拟主从延时（120.26.71.71）

mysql> stop slave io\_thread;

Query OK, 0 rows affected (0.08 sec)

（3）模拟sysbench压力测试

在主库上（10.51.23.196）进行压力测试，持续时间为30分钟，产生大量的binlog。

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

（4）关掉主库mysql进程，模拟主库发生故障，进行自动failover操作。

[root@192.168.3.110 ~]# service mysql stop

#### 4.7.5查看MHA切换日志

在120.26.127.218上查看日志

Fri Sep 11 14:31:10 2015 - [warning] Got error on MySQL select ping: 2006 (MySQL server has gone away)

Fri Sep 11 14:31:10 2015 - [info] Executing SSH check script: save\_binary\_logs --command=test --start\_pos=4 --binlog\_dir=/mydata/mysql/binlog --output\_file=/mydata/masterha/binlog/save\_binary\_logs\_test --manager\_version=0.56 --binlog\_prefix=mysql-bin

Fri Sep 11 14:31:10 2015 - [info] HealthCheck: SSH to 10.51.23.196 is reachable.

Fri Sep 11 14:31:11 2015 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Fri Sep 11 14:31:11 2015 - [warning] Connection failed 2 time(s)..

Fri Sep 11 14:31:12 2015 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Fri Sep 11 14:31:12 2015 - [warning] Connection failed 3 time(s)..

Fri Sep 11 14:31:13 2015 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Fri Sep 11 14:31:13 2015 - [warning] Connection failed 4 time(s)..

Fri Sep 11 14:31:13 2015 - [warning] Master is not reachable from health checker!

Fri Sep 11 14:31:13 2015 - [warning] Master 10.51.23.196(10.51.23.196:3316) is not reachable!

Fri Sep 11 14:31:13 2015 - [warning] SSH is reachable.

Fri Sep 11 14:31:13 2015 - [info] Connecting to a master server failed. Reading configuration file /etc/masterha\_default.cnf and /etc/app1.cnf again, and trying to connect to all servers to check server status..

Fri Sep 11 14:31:13 2015 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Fri Sep 11 14:31:13 2015 - [info] Reading application default configuration from /etc/app1.cnf..

Fri Sep 11 14:31:13 2015 - [info] Reading server configuration from /etc/app1.cnf..

Fri Sep 11 14:31:13 2015 - [info] GTID failover mode = 0

Fri Sep 11 14:31:13 2015 - [info] Dead Servers:

Fri Sep 11 14:31:13 2015 - [info] 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Alive Servers:

Fri Sep 11 14:31:13 2015 - [info] 10.252.124.43(10.252.124.43:3316)

Fri Sep 11 14:31:13 2015 - [info] 10.168.6.6(10.168.6.6:3316)

Fri Sep 11 14:31:13 2015 - [info] Alive Slaves:

Fri Sep 11 14:31:13 2015 - [info] 10.252.124.43(10.252.124.43:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:13 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Fri Sep 11 14:31:13 2015 - [info] 10.168.6.6(10.168.6.6:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:13 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Checking slave configurations..

Fri Sep 11 14:31:13 2015 - [info] Checking replication filtering settings..

Fri Sep 11 14:31:13 2015 - [info] Replication filtering check ok.

Fri Sep 11 14:31:13 2015 - [info] Master is down!

Fri Sep 11 14:31:13 2015 - [info] Terminating monitoring script.

Fri Sep 11 14:31:13 2015 - [info] Got exit code 20 (Master dead).

Fri Sep 11 14:31:13 2015 - [info] MHA::MasterFailover version 0.56.

Fri Sep 11 14:31:13 2015 - [info] Starting master failover.

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] \* Phase 1: Configuration Check Phase..

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] GTID failover mode = 0

Fri Sep 11 14:31:13 2015 - [info] Dead Servers:

Fri Sep 11 14:31:13 2015 - [info] 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Checking master reachability via MySQL(double check)...

Fri Sep 11 14:31:13 2015 - [info] ok.

Fri Sep 11 14:31:13 2015 - [info] Alive Servers:

Fri Sep 11 14:31:13 2015 - [info] 10.252.124.43(10.252.124.43:3316)

Fri Sep 11 14:31:13 2015 - [info] 10.168.6.6(10.168.6.6:3316)

Fri Sep 11 14:31:13 2015 - [info] Alive Slaves:

Fri Sep 11 14:31:13 2015 - [info] 10.252.124.43(10.252.124.43:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:13 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Fri Sep 11 14:31:13 2015 - [info] 10.168.6.6(10.168.6.6:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:13 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Starting Non-GTID based failover.

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] \*\* Phase 1: Configuration Check Phase completed.

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] \* Phase 2: Dead Master Shutdown Phase..

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] Forcing shutdown so that applications never connect to the current master..

Fri Sep 11 14:31:13 2015 - [warning] master\_ip\_failover\_script is not set. Skipping invalidating dead master IP address.

Fri Sep 11 14:31:13 2015 - [warning] shutdown\_script is not set. Skipping explicit shutting down of the dead master.

Fri Sep 11 14:31:13 2015 - [info] \* Phase 2: Dead Master Shutdown Phase completed.

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] \* Phase 3: Master Recovery Phase..

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] \* Phase 3.1: Getting Latest Slaves Phase..

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] The latest binary log file/position on all slaves is mysql-bin.000002:214847228

Fri Sep 11 14:31:13 2015 - [info] Latest slaves (Slaves that received relay log files to the latest):

Fri Sep 11 14:31:13 2015 - [info] 10.168.6.6(10.168.6.6:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:13 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] The oldest binary log file/position on all slaves is mysql-bin.000002:211876884

Fri Sep 11 14:31:13 2015 - [info] Oldest slaves:

Fri Sep 11 14:31:13 2015 - [info] 10.252.124.43(10.252.124.43:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:13 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:13 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] \* Phase 3.2: Saving Dead Master's Binlog Phase..

Fri Sep 11 14:31:13 2015 - [info]

Fri Sep 11 14:31:13 2015 - [info] Fetching dead master's binary logs..

Fri Sep 11 14:31:13 2015 - [info] Executing command on the dead master 10.51.23.196(10.51.23.196:3316): save\_binary\_logs --command=save --start\_file=mysql-bin.000002 --start\_pos=214847228 --binlog\_dir=/mydata/mysql/binlog --output\_file=/mydata/masterha/binlog/saved\_master\_binlog\_from\_10.51.23.196\_3316\_20150911143113.binlog --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.56

Creating /mydata/masterha/binlog if not exists.. ok.

Concat binary/relay logs from mysql-bin.000002 pos 214847228 to mysql-bin.000002 EOF into /mydata/masterha/binlog/saved\_master\_binlog\_from\_10.51.23.196\_3316\_20150911143113.binlog ..

Binlog Checksum enabled

Dumping binlog format description event, from position 0 to 120.. ok.

No need to dump effective binlog data from /mydata/mysql/binlog/mysql-bin.000002 (pos starts 214847228, filesize 214847228). Skipping.

Binlog Checksum enabled

/mydata/masterha/binlog/saved\_master\_binlog\_from\_10.51.23.196\_3316\_20150911143113.binlog has no effective data events.

Event not exists.

Fri Sep 11 14:31:14 2015 - [info] Additional events were not found from the orig master. No need to save.

Fri Sep 11 14:31:14 2015 - [info]

Fri Sep 11 14:31:14 2015 - [info] \* Phase 3.3: Determining New Master Phase..

Fri Sep 11 14:31:14 2015 - [info]

Fri Sep 11 14:31:14 2015 - [info] Finding the latest slave that has all relay logs for recovering other slaves..

Fri Sep 11 14:31:14 2015 - [info] HealthCheck: SSH to 10.168.6.6 is reachable.

Fri Sep 11 14:31:14 2015 - [info] Checking whether 10.168.6.6 has relay logs from the oldest position..

Fri Sep 11 14:31:14 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=find --latest\_mlf=mysql-bin.000002 --latest\_rmlp=214847228 --target\_mlf=mysql-bin.000002 --target\_rmlp=211876884 --server\_id=3 --workdir=/mydata/masterha/binlog --timestamp=20150911143113 --manager\_version=0.56 --relay\_log\_info=/mydata/mysql/data/relay-log.info --relay\_dir=/mydata/mysql/data/ :

Opening /mydata/mysql/data/relay-log.info ... ok.

Relay log found at /mydata/mysql/binlog, up to mysql-relay-bin.000005

Fast relay log position search succeeded.

Target relay log file/position found. start\_file:mysql-relay-bin.000005, start\_pos:211877047.

Target relay log FOUND!

Fri Sep 11 14:31:14 2015 - [info] OK. 10.168.6.6 has all relay logs.

Fri Sep 11 14:31:14 2015 - [info] HealthCheck: SSH to 10.252.124.43 is reachable.

Fri Sep 11 14:31:14 2015 - [info] Searching new master from slaves..

Fri Sep 11 14:31:14 2015 - [info] Candidate masters from the configuration file:

Fri Sep 11 14:31:14 2015 - [info] 10.252.124.43(10.252.124.43:3316) Version=5.6.23-log (oldest major version between slaves) log-bin:enabled

Fri Sep 11 14:31:14 2015 - [info] Replicating from 10.51.23.196(10.51.23.196:3316)

Fri Sep 11 14:31:14 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Fri Sep 11 14:31:14 2015 - [info] Non-candidate masters:

Fri Sep 11 14:31:14 2015 - [info] Searching from candidate\_master slaves which have received the latest relay log events..

Fri Sep 11 14:31:14 2015 - [info] Not found.

Fri Sep 11 14:31:14 2015 - [info] Searching from all candidate\_master slaves..

Fri Sep 11 14:31:14 2015 - [info] New master is 10.252.124.43(10.252.124.43:3316)

Fri Sep 11 14:31:14 2015 - [info] Starting master failover..

Fri Sep 11 14:31:14 2015 - [info]

From:

10.51.23.196(10.51.23.196:3316) (current master)

+--10.252.124.43(10.252.124.43:3316)

+--10.168.6.6(10.168.6.6:3316)

To:

10.252.124.43(10.252.124.43:3316) (new master)

+--10.168.6.6(10.168.6.6:3316)

Fri Sep 11 14:31:14 2015 - [info]

Fri Sep 11 14:31:14 2015 - [info] \* Phase 3.3: New Master Diff Log Generation Phase..

Fri Sep 11 14:31:14 2015 - [info]

Fri Sep 11 14:31:14 2015 - [info] Server 10.252.124.43 received relay logs up to: mysql-bin.000002:211876884

Fri Sep 11 14:31:14 2015 - [info] Need to get diffs from the latest slave(10.168.6.6) up to: mysql-bin.000002:214847228 (using the latest slave's relay logs)

Fri Sep 11 14:31:14 2015 - [info] Connecting to the latest slave host 10.168.6.6, generating diff relay log files..

Fri Sep 11 14:31:14 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=generate\_and\_send --scp\_user=root --scp\_host=10.252.124.43 --latest\_mlf=mysql-bin.000002 --latest\_rmlp=214847228 --target\_mlf=mysql-bin.000002 --target\_rmlp=211876884 --server\_id=3 --diff\_file\_readtolatest=/mydata/masterha/binlog/relay\_from\_read\_to\_latest\_10.252.124.43\_3316\_20150911143113.binlog --workdir=/mydata/masterha/binlog --timestamp=20150911143113 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.56 --relay\_log\_info=/mydata/mysql/data/relay-log.info --relay\_dir=/mydata/mysql/data/

Fri Sep 11 14:31:15 2015 - [info]

Opening /mydata/mysql/data/relay-log.info ... ok.

Relay log found at /mydata/mysql/binlog, up to mysql-relay-bin.000005

Fast relay log position search succeeded.

Target relay log file/position found. start\_file:mysql-relay-bin.000005, start\_pos:211877047.

Concat binary/relay logs from mysql-relay-bin.000005 pos 211877047 to mysql-relay-bin.000005 EOF into /mydata/masterha/binlog/relay\_from\_read\_to\_latest\_10.252.124.43\_3316\_20150911143113.binlog ..

Binlog Checksum enabled

Binlog Checksum enabled

Dumping binlog format description event, from position 0 to 283.. ok.

Dumping effective binlog data from /mydata/mysql/binlog/mysql-relay-bin.000005 position 211877047 to tail(214847391).. ok.

Binlog Checksum enabled

Binlog Checksum enabled

Concat succeeded.

Generating diff relay log succeeded. Saved at /mydata/masterha/binlog/relay\_from\_read\_to\_latest\_10.252.124.43\_3316\_20150911143113.binlog .

scp iZ23k0d7o6qZ:/mydata/masterha/binlog/relay\_from\_read\_to\_latest\_10.252.124.43\_3316\_20150911143113.binlog to root@10.252.124.43(22) succeeded.

Fri Sep 11 14:31:15 2015 - [info] Generating diff files succeeded.

Fri Sep 11 14:31:15 2015 - [info]

Fri Sep 11 14:31:15 2015 - [info] \* Phase 3.4: Master Log Apply Phase..

Fri Sep 11 14:31:15 2015 - [info]

Fri Sep 11 14:31:15 2015 - [info] \*NOTICE: If any error happens from this phase, manual recovery is needed.

Fri Sep 11 14:31:15 2015 - [info] Starting recovery on 10.252.124.43(10.252.124.43:3316)..

Fri Sep 11 14:31:15 2015 - [info] Generating diffs succeeded.

Fri Sep 11 14:31:15 2015 - [info] Waiting until all relay logs are applied.

Fri Sep 11 14:31:15 2015 - [info] done.

Fri Sep 11 14:31:15 2015 - [info] Getting slave status..

Fri Sep 11 14:31:15 2015 - [info] This slave(10.252.124.43)'s Exec\_Master\_Log\_Pos equals to Read\_Master\_Log\_Pos(mysql-bin.000002:211876884). No need to recover from Exec\_Master\_Log\_Pos.

Fri Sep 11 14:31:15 2015 - [info] Connecting to the target slave host 10.252.124.43, running recover script..

Fri Sep 11 14:31:15 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user='rkylinadmin' --slave\_host=10.252.124.43 --slave\_ip=10.252.124.43 --slave\_port=3316 --apply\_files=/mydata/masterha/binlog/relay\_from\_read\_to\_latest\_10.252.124.43\_3316\_20150911143113.binlog --workdir=/mydata/masterha/binlog --target\_version=5.6.23-log --timestamp=20150911143113 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.56 --slave\_pass=xxx

Fri Sep 11 14:33:41 2015 - [info]

MySQL client version is 5.6.23. Using --binary-mode.

Applying differential binary/relay log files /mydata/masterha/binlog/relay\_from\_read\_to\_latest\_10.252.124.43\_3316\_20150911143113.binlog on 10.252.124.43:3316. This may take long time...

Applying log files succeeded.

Fri Sep 11 14:33:41 2015 - [info] All relay logs were successfully applied.

Fri Sep 11 14:33:41 2015 - [info] Getting new master's binlog name and position..

Fri Sep 11 14:33:41 2015 - [info] mysql-bin.000073:1017466248

Fri Sep 11 14:33:41 2015 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='10.252.124.43', MASTER\_PORT=3316, MASTER\_LOG\_FILE='mysql-bin.000073', MASTER\_LOG\_POS=1017466248, MASTER\_USER='slaveadmin', MASTER\_PASSWORD='xxx';

Fri Sep 11 14:33:41 2015 - [warning] master\_ip\_failover\_script is not set. Skipping taking over new master IP address.

Fri Sep 11 14:33:41 2015 - [info] Setting read\_only=0 on 10.252.124.43(10.252.124.43:3316)..

Fri Sep 11 14:33:41 2015 - [info] ok.

Fri Sep 11 14:33:41 2015 - [info] \*\* Finished master recovery successfully.

Fri Sep 11 14:33:41 2015 - [info] \* Phase 3: Master Recovery Phase completed.

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] \* Phase 4: Slaves Recovery Phase..

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] \* Phase 4.1: Starting Parallel Slave Diff Log Generation Phase..

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] -- Slave diff file generation on host 10.168.6.6(10.168.6.6:3316) started, pid: 25238. Check tmp log /mydata/masterha/app1/10.168.6.6\_3316\_20150911143113.log if it takes time..

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] Log messages from 10.168.6.6 ...

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] This server has all relay logs. No need to generate diff files from the latest slave.

Fri Sep 11 14:33:41 2015 - [info] End of log messages from 10.168.6.6.

Fri Sep 11 14:33:41 2015 - [info] -- 10.168.6.6(10.168.6.6:3316) has the latest relay log events.

Fri Sep 11 14:33:41 2015 - [info] Generating relay diff files from the latest slave succeeded.

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] \* Phase 4.2: Starting Parallel Slave Log Apply Phase..

Fri Sep 11 14:33:41 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] -- Slave recovery on host 10.168.6.6(10.168.6.6:3316) started, pid: 25240. Check tmp log /mydata/masterha/app1/10.168.6.6\_3316\_20150911143113.log if it takes time..

Fri Sep 11 14:34:57 2015 - [info]

Fri Sep 11 14:34:57 2015 - [info] Log messages from 10.168.6.6 ...

Fri Sep 11 14:34:57 2015 - [info]

Fri Sep 11 14:33:41 2015 - [info] Starting recovery on 10.168.6.6(10.168.6.6:3316)..

Fri Sep 11 14:33:41 2015 - [info] This server has all relay logs. Waiting all logs to be applied..

Fri Sep 11 14:34:57 2015 - [info] done.

Fri Sep 11 14:34:57 2015 - [info] All relay logs were successfully applied.

Fri Sep 11 14:34:57 2015 - [info] Resetting slave 10.168.6.6(10.168.6.6:3316) and starting replication from the new master 10.252.124.43(10.252.124.43:3316)..

Fri Sep 11 14:34:57 2015 - [info] Executed CHANGE MASTER.

Fri Sep 11 14:34:57 2015 - [info] Slave started.

Fri Sep 11 14:34:57 2015 - [info] End of log messages from 10.168.6.6.

Fri Sep 11 14:34:57 2015 - [info] -- Slave recovery on host 10.168.6.6(10.168.6.6:3316) succeeded.

Fri Sep 11 14:34:57 2015 - [info] All new slave servers recovered successfully.

Fri Sep 11 14:34:57 2015 - [info]

Fri Sep 11 14:34:57 2015 - [info] \* Phase 5: New master cleanup phase..

Fri Sep 11 14:34:57 2015 - [info]

Fri Sep 11 14:34:57 2015 - [info] Resetting slave info on the new master..

Fri Sep 11 14:34:57 2015 - [info] 10.252.124.43: Resetting slave info succeeded.

Fri Sep 11 14:34:57 2015 - [info] Master failover to 10.252.124.43(10.252.124.43:3316) completed successfully.

Fri Sep 11 14:34:57 2015 - [info]

----- Failover Report -----

app1: MySQL Master failover 10.51.23.196(10.51.23.196:3316) to 10.252.124.43(10.252.124.43:3316) succeeded

Master 10.51.23.196(10.51.23.196:3316) is down!

Check MHA Manager logs at iZ23zc5fzqaZ:/mydata/masterha/manager.log for details.

Started automated(non-interactive) failover.

The latest slave 10.168.6.6(10.168.6.6:3316) has all relay logs for recovery.

Selected 10.252.124.43(10.252.124.43:3316) as a new master.

10.252.124.43(10.252.124.43:3316): OK: Applying all logs succeeded.

10.168.6.6(10.168.6.6:3316): This host has the latest relay log events.

Generating relay diff files from the latest slave succeeded.

10.168.6.6(10.168.6.6:3316): OK: Applying all logs succeeded. Slave started, replicating from 10.252.124.43(10.252.124.43:3316)

10.252.124.43(10.252.124.43:3316): Resetting slave info succeeded.

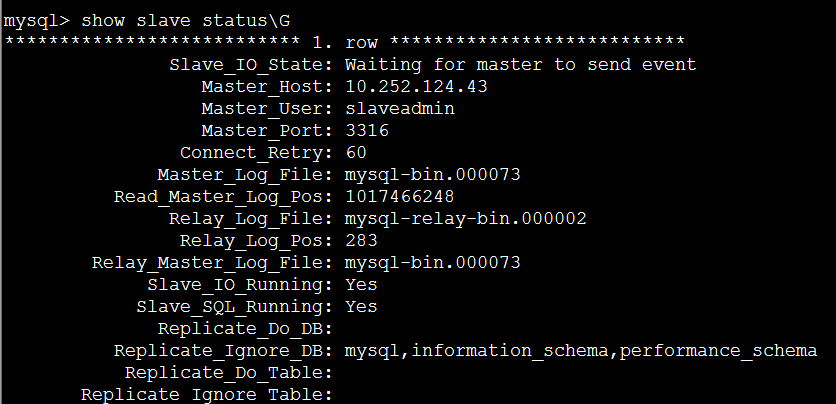
Master failover to 10.252.124.43(10.252.124.43:3316) completed successfully.

Fri Sep 11 14:34:57 2015 - [info] Sending mail..

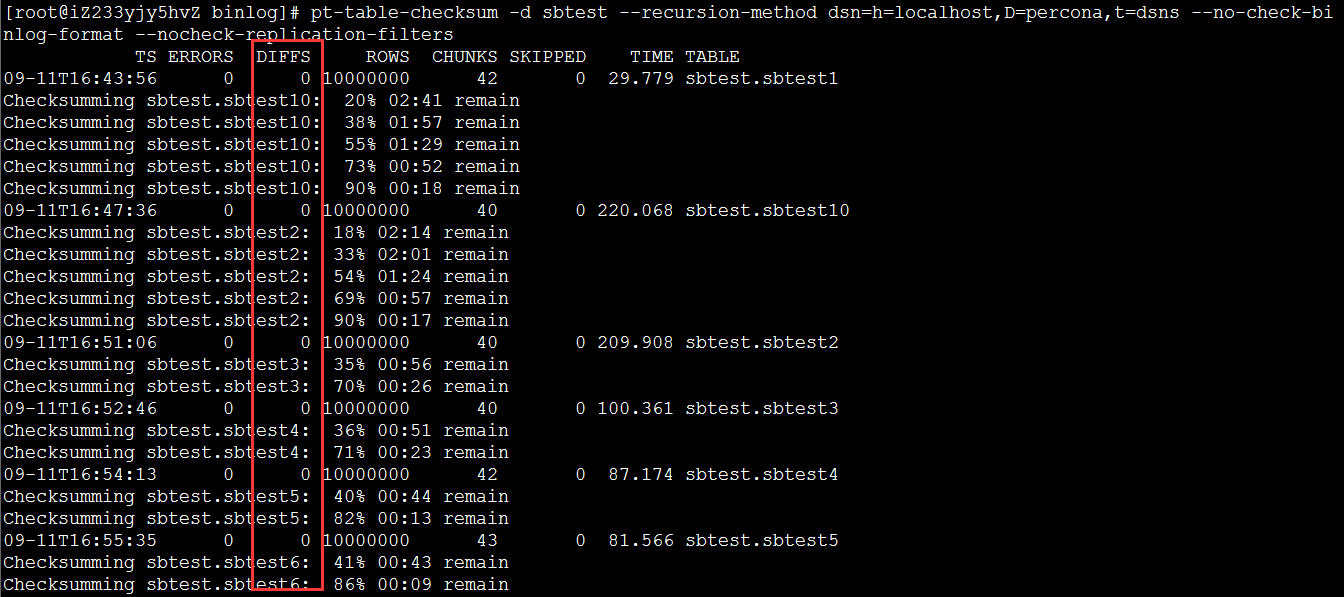
Unknown option: conf

#### 4.7.6测试结果

切换完毕，查看从服务器是否切到新的master 10.252.124.43上。



一致性结果：



#### 4.7.7结果分析

MHA可以完美实现无VIP的自动复制切换，数据能够保持一致。

### 4.9复制可接受的最大并发测试

#### 4.9.1测试场景

对复制主节点经行不同并发线程的压力测试，同时监控复制延迟，评估在多大并发量是数据库基本无延迟。

#### 4.9.2测试环境

MySQL主从配置相同

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| Master1 IP | 10.51.23.196:3316 |
| Master2 IP | 10.252.124.43:3316 |
| Slave IP | 10.168.6.6:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |

#### 4.9.3配置参数

数据库配置：

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 0

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 1100

memlock = on

#### 4.9.4测试步骤

（1）使用sysbench生成测试数据

在主库（10.51.23.196）上进行sysbench数据生成，在sbtest库下生成sbtest表，共1000W记录。

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

（2）模拟sysbench压力测试

在主库上（10.51.23.196）进行压力测试，持续时间为30分钟，产生大量的binlog。

./sysbench --mysql-host=10.51.23.196 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=off \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

测试基准：

测试工具： sysbench

测试表数据行： 10,000,000（24G）

测试模式： complex

测试表引擎： innodb

测试表数量： 10个

测试时间： 30分钟

并发线程数：16、32、64、128

（3）登录从数据库，查看测试延迟

Mysql> show slave status\G

#### 4.9.5测试结果

在并发线程数为32时，从服务器没有延迟，到64线程时就会出现延迟了，并随着时间不断增大，主备在64线程时延迟保持在4秒以内。

#### 4.9.6结果分析

目前复制架构，在并发连接数为60左右时，主备基本无延迟，从库延迟会不断增大；在连接数30左右时，从库没有延迟。因为主备使用的半同步复制，所以延迟会比从库小。

### 4.10Atlas功能测试

#### 4.9.1测试场景

在MySQL复制架构之上建立atlas代理中间层，实现读写分离，负载均衡，IP过滤（指定客户端连接），主库宕机，从可以继续进行读访问；从库宕机，可以自动摘除；事物读从主读，非事物读从从库读。

#### 4.9.2测试环境

MySQL主从配置相同

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| Master1 IP | 10.51.23.196:3316 |
| Master2 IP | 10.252.124.43:3316 |
| Slave IP | 10.168.6.6:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |
| Atalas IP | 10.252.82.111 |

#### 4.9.3配置参数

Atlas配置参数：

[mysql-proxy]

#管理接口的用户名

admin-username=admin

#管理接口的密码

admin-password=admin

#Atlas后端连接的MySQL主库的IP和端口，可设置多项，用逗号分隔

proxy-backend-addresses=10.51.23.196:3316

#Atlas后端连接的MySQL从库的IP和端口，@后面的数字代表权重，用来作负载均衡，若省略则默认为1，可设置多项，用逗号分隔

proxy-read-only-backend-addresses=10.168.6.6:3316@1,10.252.124.43:3316@3

#用户名与其对应的加密过的MySQL密码，密码使用PREFIX/bin目录下的加密程序encrypt加密，下行的user1和user2为示例，将其替换为你的MySQL的用户名和加密密码！

pwds=atlas:GX7TVypEBgw=,rkylinadmin:qCyFeuAFF15So2ktcXUZdg==,tpcc:D/P0riIcOmU=

#设置Atlas的运行方式，设为true时为守护进程方式，设为false时为前台方式，一般开发调试时设为false，线上运行时设为true,true后面不能有空格。

daemon=true

#设置Atlas的运行方式，设为true时Atlas会启动两个进程，一个为monitor，一个为worker，monitor在worker意外退出后会自动将其重启，设为false时只有worker，没有monitor，一般开发调试时设为false，线上运行时设为true,true后面不能有空格。

keepalive=false

#工作线程数，对Atlas的性能有很大影响，可根据情况适当设置

event-threads=8

#日志级别，分为message、warning、critical、error、debug五个级别

log-level=message

#日志存放的路径

log-path=/usr/local/mysql-proxy/log

#SQL日志的开关，可设置为OFF、ON、REALTIME，OFF代表不记录SQL日志，ON代表记录SQL日志，REALTIME代表记录SQL日志且实时写入磁盘，默认为OFF

sql-log=ON

#慢日志输出设置。当设置了该参数时，则日志只输出执行时间超过sql-log-slow（单位：ms)的日志记录。不设置该参数则输出全部日志。

#sql-log-slow = 10

#实例名称，用于同一台机器上多个Atlas实例间的区分

#instance = test

#Atlas监听的工作接口IP和端口

proxy-address=0.0.0.0:3316

#Atlas监听的管理接口IP和端口

admin-address=0.0.0.0:3315

#连接池的最小空闲连接数，可根据业务请求量大小适当调大或调小

#min-idle-connections=128

#分表设置，此例中person为库名，mt为表名，id为分表字段，3为子表数量，可设置多项，以逗号分隔，若不分表则不需要设置该项

#tables = person.mt.id.3

#默认字符集，设置该项后客户端不再需要执行SET NAMES语句

charset=utf8

#允许连接Atlas的客户端的IP，可以是精确IP，也可以是IP段，以逗号分隔，若不设置该项则允许所有IP连接，否则只允许列表中的IP连接

client-ips=10.252.124.43,10.168.6.6,10.51.23.196,10.251.233.151,10.252.82.111

#Atlas前面挂接的LVS的物理网卡的IP(注意不是虚IP)，若有LVS且设置了client-ips则此项必须设置，否则可以不设置

#lvs-ips = 192.168.1.1

#### 4.9.4测试步骤

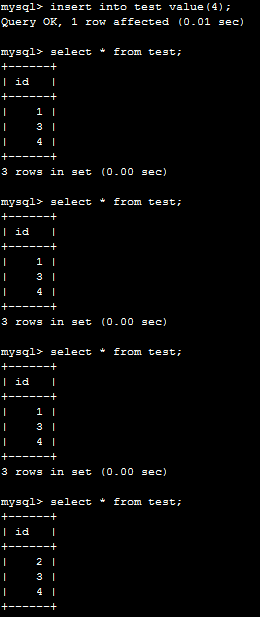
（1）在主上创建测试数据库rkylin，在其中创建表test（id int），插入数据（1,2,3）。

（2）为了辨别读写分离，调整从上的数据，使与主不一致，主备上为（1,3），从上为（2,3）。

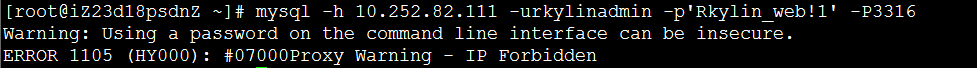
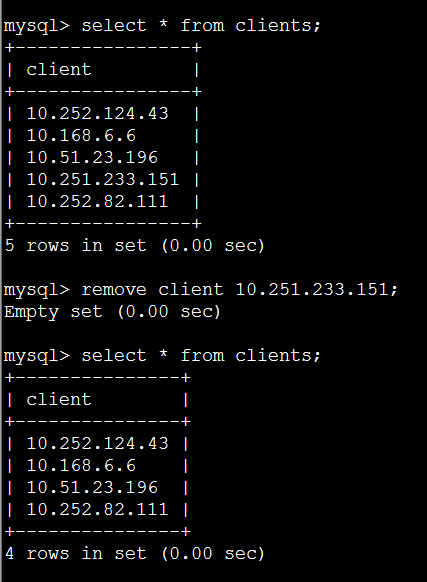
（3）从10.251.233.151登录到atlas代理，向test中插入一条新记录（4），查看数据，以确定读写分离和负载均衡，当前负载为1:3。

mysql -h 10.252.82.111 -urkylinadmin -p'Rkylin\_web!1' -P3316

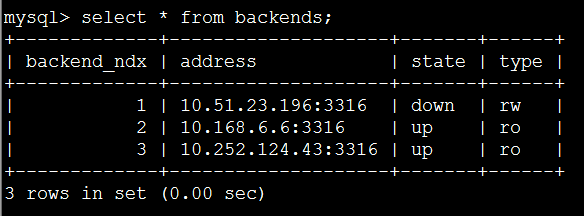
mysql> use rkylin;

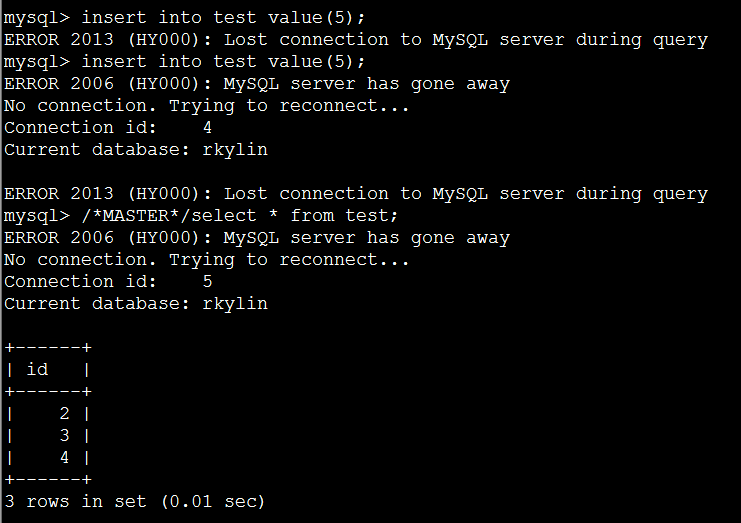


（4）去掉配置中的10.251.233.151客户端，再尝试登录。

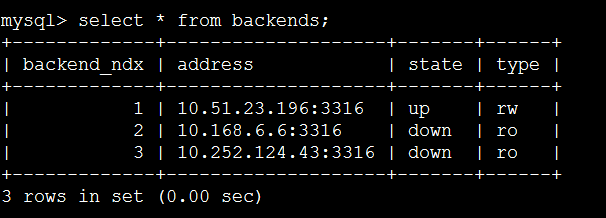


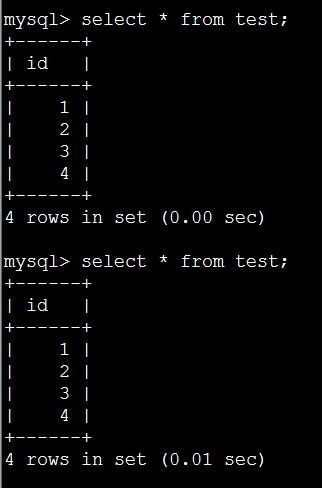
（5）停掉主库，继续查询数据。



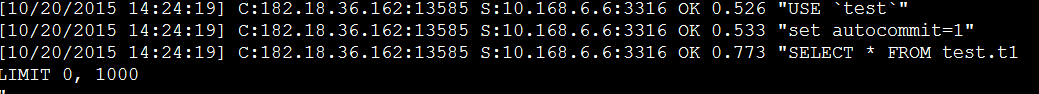


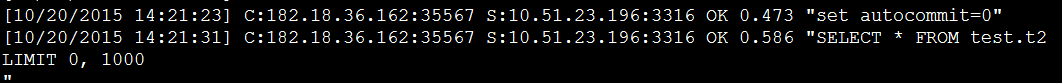
（6）停掉从库，继续查询数据

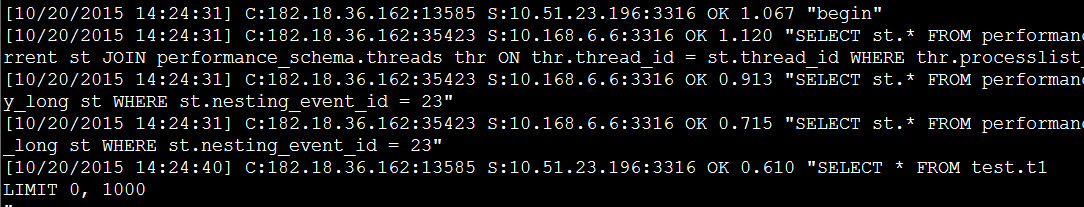




（7）当auto\_commit=0时，进行select操作，更改auto\_commit=1，再进行select操作，查看从哪个库读数据，显示使用begin、commit语句，查看从哪个库读。







#### 4.9.5结果分析

1. Atlas可以实现读写分离和负载均衡
2. 可以过滤IP客户端
3. 主库宕机，从库可以继续读
4. 从库宕机，可以在主库读写
5. 事物语句会从主库读，非事物语句从从库读。

### 4.11Atalas性能测试

#### 4.11.1测试场景

在MySQL复制架构之上建立atlas代理中间层，一主两从即一写两读，slave1和slave2的权重是3：1,使用sysbench测试MySQL性能基线。测试时对CPU、IO进行监控，找出性能瓶颈。与单点性能基线进行比较。

#### 4.11.2测试环境

Atlas配置

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| Atlas IP | 10.252.82.111:3316 |
| MySQL master IP | 10.51.23.196:3316 |
| MySQL slave1 IP | 10.252.124.43:3316 |
| MySQL slave2 IP | 10.168.6.6:3316 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |
| MySQL版本 | 5.6.23 |
| Sysbench IP | 10.251.233.151 |

#### 4.11.3配置参数

原始参数：

innodb\_buffer\_pool\_size = 10G

innodb\_data\_file\_path = ibdata1:12M:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 64M

innodb\_log\_file\_size = 3G

innodb\_log\_files\_in\_group = 3

innodb\_flush\_neighbors = 1

tx\_isolation = REPEATABLE-READ

innodb\_flush\_method = O\_DIRECT

sort\_buffer\_size = 2M

join\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 512K

thread\_stack = 512K

key\_buffer\_size = 8M

binlog\_cache\_size = 4M

query\_cache\_size = 1M

query\_cache\_type = 0

tmp\_table\_size = 16M

max\_heap\_table\_size = 16M

max\_connections = 500

memlock = off

#### 4.11.4测试脚本

在数据库添加测试用的账户tpcc：

grant all on sbtest.\* to tpcc@'%' identified by 'tpcc';

测试前准备测试数据：

./sysbench --mysql-host=10.252.82.111 --mysql-port=3316 --mysql-user=tpcc --mysql-password=tpcc \

--test=tests/db/oltp.lua --oltp\_tables\_count=10 --oltp-table-size=10000000 --rand-init=on prepare

开始测试：

./sysbench --mysql-host=10.252.82.111 --mysql-port=3316 --mysql-user=tpcc \

--mysql-password=tpcc --test=tests/db/oltp.lua --oltp\_tables\_count=10 \

--oltp-table-size=10000000 --num-threads=128 --oltp-read-only=on \

--report-interval=10 --rand-type=uniform --max-time=1800 \

--max-requests=0 --percentile=99 run

测试基准：

测试工具： sysbench

测试表数据行： 10,000,000

测试模式： complex

测试表引擎： innodb

测试表数量： 10个

测试时间： 30分钟

并发线程数： 128

#### 4.11.5测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 服务器 | 线程数 | TPS | QPS | RT99%(ms) | %user | %iowait | %idle | %util |
| Atlas | 256 | 0 | 1742.49 | 3817.21ms | 3.88 | 80.67 | 13.18 | 100.47 |
| 2.83 | 20.50 | 74.82 | 68.83 |
| Slave(直连) | 256 | 0 | 976.15 | 6138.44ms | 3.18 | 82.59 | 13.04 | 102.19 |
| Atlas | 128 | 0 | 1662.31 | 2625.69ms | 3.33 | 38.96 | 55.71 | 99.67 |
| 3.00 | 25.22 | 69.97 | 90.72 |
| Slave(直连) | 128 | 0 | 1228.01 | 2856.11ms | 3.59 | 37.87 | 57.33 | 102.01 |
| Atlas | 64 | 0 | 1744.53 | 1438.17ms | 0.78 | 4.53 | 94.17 | 32.77 |
| 0.76 | 4.80 | 93.93 | 20.20 |
| Slave(直连) | 64 | 0 | 1184.41 | 1563.91ms | 3.47 | 23.82 | 71.49 | 102.15 |
| Atlas | 32 | 0 | 1456.63 | 824.90ms | 3.17 | 14.22 | 80.68 | 99.44 |
| 2.75 | 9.32 | 86.17 | 92.07 |
| Slave(直连) | 32 | 0 | 1156.39 | 1014.45ms | 3.38 | 18.26 | 77.28 | 102.86 |
| Atlas | 16 | 0 | 950.99 | 625.76ms | 2.02 | 10.43 | 86.42 | 99.21 |
| 1.88 | 9.52 | 87.37 | 94.91 |
| Slave(直连) | 16 | 0 | 909.14 | 696.13ms | 2.62 | 11.54 | 85.09 | 102.70 |

#### 4.11.6结果分析

当并发线程数小的时候，atlas性能提高的不明显，随着并发量增大，atlas的优势就越来越明显了。

### 4.12IO性能测试

#### 4.12.1测试场景

使用sysbench对服务器IO进行测试。

#### 4.12.2测试环境

Atlas配置

|  |  |
| --- | --- |
| 机型 | 阿里云服务器 |
| MySQL slave1 IP | 10.252.124.43 |
| MySQL slave2 IP | 10.168.6.6 |
| MySQL master IP | 10.51.23.196 |
| CPU | 8核，8线程，2.594GHz |
| 内存 | 16G |
| 阵列卡和设置 | 无 |
| 硬盘 | 256G |
| 网卡 | 千兆 |
| 操作系统 | CentOS 6.7 |
| 文件系统 | XFS |

#### 4.12.4测试脚本

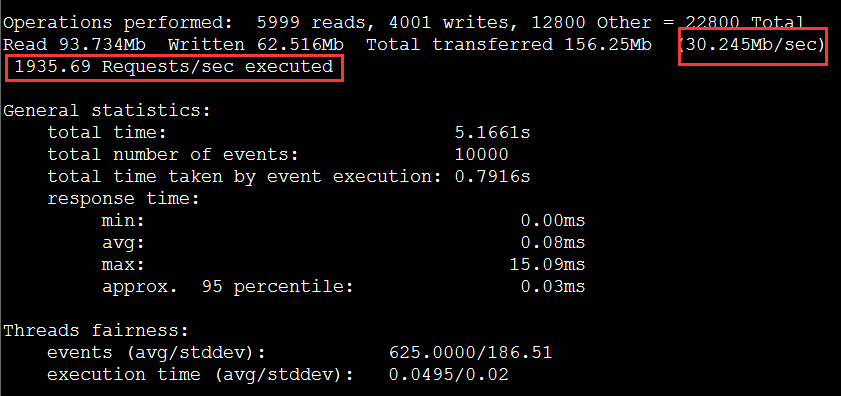
./sysbench --test=fileio --num-threads=16 --file-total-size=5000M --file-test-mode=rndrw prepare

./sysbench --test=fileio --num-threads=16 --file-total-size=5000M --file-test-mode=rndrw run

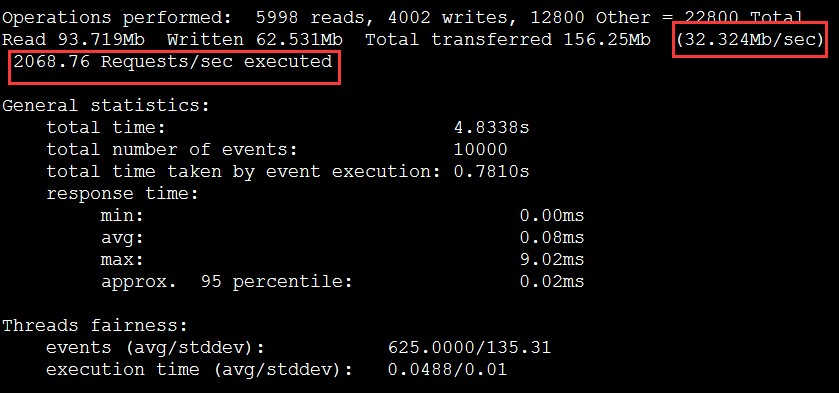
./sysbench --test=fileio --num-threads=16 --file-total-size=5000M --file-test-mode=rndrw cleanup

#### 4.12.5测试结果

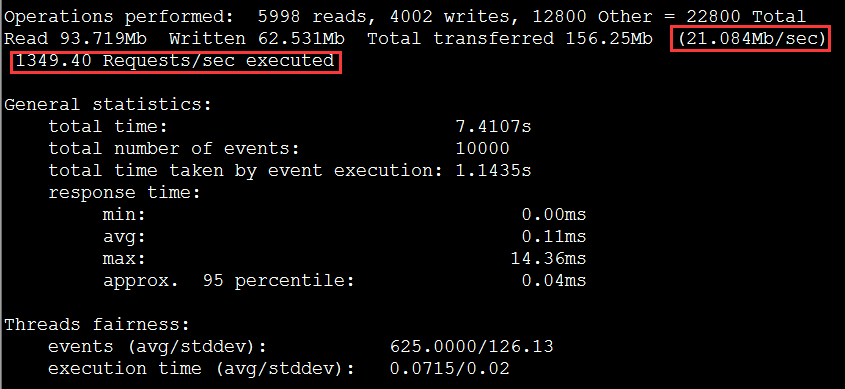
Master：



Slave1：



Slave2：



#### 4.12.6结果分析

Master与slave1的IO能力都为每秒2000左右的请求，slave2为每秒1300左右请求，相对差一些。

### 4.13MHA测试

到此为止，基本MHA集群已经配置完毕。接下来就是实际的测试环节了。通过一些测试来看一下MHA到底是如何进行工作的。下面将从MHA自动failover、手动failover两种方式来介绍MHA的工作情况。

#### 自动Failover

必须先启动MHA Manager，否则无法自动切换，当然手动切换不需要开启MHA Manager监控。

自动failover模拟测试的操作步骤如下：

（1）使用sysbench生成测试数据（使用yum快速安装），sysbench的使用说明可以查看文档sysbench使用详解

#yum install sysbench -y

在主库（192.168.3.110）上进行sysbench数据生成，在sbtest库下生成sbtest表，共100W记录。

# sysbench --test=oltp --oltp-table-size=1000000 --oltp-read-only=off --mysql-db=sps --oltp-table-name=sbtest1 --init-rng=on --num-threads=16 --max-requests=0 --oltp-dist-type=uniform --max-time=1800 --mysql-user=root --mysql-socket=/var/lib/mysql/mysql.sock --mysql-password=123456 --db-driver=mysql --mysql-table-engine=innodb --oltp-test-mode=complex perpare

（2）停掉slave sql线程，模拟主从延时（192.168.3.118）

mysql> stop slave io\_thread;

Query OK, 0 rows affected (0.08 sec)

mysql>

另外两台slave我们没有停止io线程，所以还在继续接收日志。

（3）模拟sysbench压力测试

在主库上（192.168.3.110）进行压力测试，持续时间为3分钟，产生大量的binlog。

# sysbench --test=oltp --oltp-table-size=1000000 --oltp-read-only=off --mysql-db=sps --oltp-table-name=sbtest1 --init-rng=on --num-threads=16 --max-requests=0 --oltp-dist-type=uniform --max-time=1800 --mysql-user=root --mysql-socket=/var/lib/mysql/mysql.sock --mysql-password=123456 --db-driver=mysql --mysql-table-engine=innodb --oltp-test-mode=complex run

sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 16

Initializing random number generator from timer.

Doing OLTP test.

Running mixed OLTP test

Using Uniform distribution

Using "BEGIN" for starting transactions

Using auto\_inc on the id column

Threads started!

Time limit exceeded, exiting...

(last message repeated 15 times)

Done.

OLTP test statistics:

queries performed:

read: 15092

write: 5390

other: 2156

total: 22638

transactions: 1078 (5.92 per sec.)

deadlocks: 0 (0.00 per sec.)

read/write requests: 20482 (112.56 per sec.)

other operations: 2156 (11.85 per sec.)

Test execution summary:

total time: 181.9728s

total number of events: 1078

total time taken by event execution: 2910.4518

per-request statistics:

min: 934.29ms

avg: 2699.86ms

max: 7679.95ms

approx. 95 percentile: 4441.47ms

Threads fairness:

events (avg/stddev): 67.3750/1.49

execution time (avg/stddev): 181.9032/0.11

（4）开启slave（192.168.3.118）上的IO线程，追赶落后于master的binlog。

mysql> start slave io\_thread;

Query OK, 0 rows affected (0.00 sec)

mysql>

（5）杀掉主库mysql进程，模拟主库发生故障，进行自动failover操作。

[root@192.168.3.110 ~]# pkill -9 mysqld

（6）查看MHA切换日志，了解整个切换过程，在192.168.3.123上查看日志（注意：由于测试多次切换过程，下面摘录的是192.168.3.118为主机，192.168.3.110为候选主机的切换log）：

[root@192.168.3.123 ~]# tail -f /var/log/masterha/app1/manager.log

IN SCRIPT TEST====/etc/init.d/keepalived stop==/etc/init.d/keepalived start===

Checking the Status of the script.. OK

Thu Jun 11 13:52:09 2015 - [info] OK.

Thu Jun 11 13:52:09 2015 - [warning] shutdown\_script is not defined.

Thu Jun 11 13:52:09 2015 - [info] Set master ping interval 1 seconds.

Thu Jun 11 13:52:09 2015 - [info] Set secondary check script: /usr/local/bin/masterha\_secondary\_check -s B-dev10 -s B-dev18 --user=root --master\_host=B-dev18 --master\_ip=192.168.3.118 --master\_port=3306

Thu Jun 11 13:52:09 2015 - [info] Starting ping health check on 192.168.3.118(192.168.3.118:3306)..

Thu Jun 11 13:52:09 2015 - [info] Ping(SELECT) succeeded, waiting until MySQL doesn't respond..

Thu Jun 11 13:54:01 2015 - [warning] Got error on MySQL select ping: 2013 (Lost connection to MySQL server during query)

Thu Jun 11 13:54:02 2015 - [info] Executing SSH check script: save\_binary\_logs --command=test --start\_pos=4 --binlog\_dir=/var/lib/mysql --output\_file=/tmp/save\_binary\_logs\_test --manager\_version=0.53 --binlog\_prefix=mysql-bin

Thu Jun 11 13:54:02 2015 - [info] Executing seconary network check script: /usr/local/bin/masterha\_secondary\_check -s B-dev10 -s B-dev18 --user=root --master\_host=B-dev18 --master\_ip=192.168.3.118 --master\_port=3306 --user=root --master\_host=192.168.3.118 --master\_ip=192.168.3.118 --master\_port=3306

Thu Jun 11 13:54:02 2015 - [info] HealthCheck: SSH to 192.168.3.118 is reachable.

Thu Jun 11 13:54:02 2015 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Thu Jun 11 13:54:02 2015 - [warning] Connection failed 1 time(s)..

Monitoring server B-dev10 is reachable, Master is not reachable from B-dev10. OK.

Thu Jun 11 13:54:03 2015 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Thu Jun 11 13:54:03 2015 - [warning] Connection failed 2 time(s)..

Monitoring server B-dev18 is reachable, Master is not reachable from B-dev18. OK.

Thu Jun 11 13:54:04 2015 - [info] Master is not reachable from all other monitoring servers. Failover should start.

Thu Jun 11 13:54:04 2015 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Thu Jun 11 13:54:04 2015 - [warning] Connection failed 3 time(s)..

Thu Jun 11 13:54:04 2015 - [warning] Master is not reachable from health checker!

Thu Jun 11 13:54:04 2015 - [warning] Master 192.168.3.118(192.168.3.118:3306) is not reachable!

Thu Jun 11 13:54:04 2015 - [warning] SSH is reachable.

Thu Jun 11 13:54:04 2015 - [info] Connecting to a master server failed. Reading configuration file /etc/masterha\_default.cnf and /etc/masterha/app1.cnf again, and trying to connect to all servers to check server status..

Thu Jun 11 13:54:04 2015 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Thu Jun 11 13:54:04 2015 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Thu Jun 11 13:54:04 2015 - [info] Reading server configurations from /etc/masterha/app1.cnf..

Thu Jun 11 13:54:13 2015 - [info] Dead Servers:

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:13 2015 - [info] Alive Servers:

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.110(192.168.3.110:3306)

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.115(192.168.3.115:3306)

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.117(192.168.3.117:3306)

Thu Jun 11 13:54:13 2015 - [info] Alive Slaves:

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.110(192.168.3.110:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:13 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:13 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.115(192.168.3.115:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:13 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:13 2015 - [info] 192.168.3.117(192.168.3.117:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:13 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:13 2015 - [info] Checking slave configurations..

Thu Jun 11 13:54:13 2015 - [info] read\_only=1 is not set on slave 192.168.3.110(192.168.3.110:3306).

Thu Jun 11 13:54:13 2015 - [warning] relay\_log\_purge=0 is not set on slave 192.168.3.110(192.168.3.110:3306).

Thu Jun 11 13:54:13 2015 - [info] read\_only=1 is not set on slave 192.168.3.115(192.168.3.115:3306).

Thu Jun 11 13:54:13 2015 - [info] read\_only=1 is not set on slave 192.168.3.117(192.168.3.117:3306).

Thu Jun 11 13:54:13 2015 - [info] Checking replication filtering settings..

Thu Jun 11 13:54:13 2015 - [info] Replication filtering check ok.

Thu Jun 11 13:54:13 2015 - [info] Master is down!

Thu Jun 11 13:54:13 2015 - [info] Terminating monitoring script.

Thu Jun 11 13:54:13 2015 - [info] Got exit code 20 (Master dead).

Thu Jun 11 13:54:13 2015 - [info] MHA::MasterFailover version 0.53.

Thu Jun 11 13:54:13 2015 - [info] Starting master failover.

Thu Jun 11 13:54:13 2015 - [info]

Thu Jun 11 13:54:13 2015 - [info] \* Phase 1: Configuration Check Phase..

Thu Jun 11 13:54:13 2015 - [info]

Thu Jun 11 13:54:19 2015 - [info] Dead Servers:

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:19 2015 - [info] Checking master reachability via mysql(double check)..

Thu Jun 11 13:54:19 2015 - [info] ok.

Thu Jun 11 13:54:19 2015 - [info] Alive Servers:

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.110(192.168.3.110:3306)

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.115(192.168.3.115:3306)

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.117(192.168.3.117:3306)

Thu Jun 11 13:54:19 2015 - [info] Alive Slaves:

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.110(192.168.3.110:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:19 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:19 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.115(192.168.3.115:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:19 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:19 2015 - [info] 192.168.3.117(192.168.3.117:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:19 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:22 2015 - [info] \*\* Phase 1: Configuration Check Phase completed.

Thu Jun 11 13:54:22 2015 - [info]

Thu Jun 11 13:54:22 2015 - [info] \* Phase 2: Dead Master Shutdown Phase..

Thu Jun 11 13:54:22 2015 - [info]

Thu Jun 11 13:54:22 2015 - [info] Forcing shutdown so that applications never connect to the current master..

Thu Jun 11 13:54:22 2015 - [info] Executing master IP deactivatation script:

Thu Jun 11 13:54:22 2015 - [info] /usr/local/bin/master\_ip\_failover --orig\_master\_host=192.168.3.118 --orig\_master\_ip=192.168.3.118 --orig\_master\_port=3306 --command=stopssh --ssh\_user=root

IN SCRIPT TEST====/etc/init.d/keepalived stop==/etc/init.d/keepalived start===

Disabling the VIP on old master: 192.168.3.118

Thu Jun 11 13:54:23 2015 - [info] done.

Thu Jun 11 13:54:23 2015 - [warning] shutdown\_script is not set. Skipping explicit shutting down of the dead master.

Thu Jun 11 13:54:23 2015 - [info] \* Phase 2: Dead Master Shutdown Phase completed.

Thu Jun 11 13:54:23 2015 - [info]

Thu Jun 11 13:54:23 2015 - [info] \* Phase 3: Master Recovery Phase..

Thu Jun 11 13:54:23 2015 - [info]

Thu Jun 11 13:54:23 2015 - [info] \* Phase 3.1: Getting Latest Slaves Phase..

Thu Jun 11 13:54:23 2015 - [info]

Thu Jun 11 13:54:23 2015 - [info] The latest binary log file/position on all slaves is mysql-bin.000001:145958297

Thu Jun 11 13:54:23 2015 - [info] Latest slaves (Slaves that received relay log files to the latest):

Thu Jun 11 13:54:23 2015 - [info] 192.168.3.110(192.168.3.110:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:23 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:23 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Thu Jun 11 13:54:23 2015 - [info] The oldest binary log file/position on all slaves is mysql-bin.000001:142466719

Thu Jun 11 13:54:23 2015 - [info] Oldest slaves:

Thu Jun 11 13:54:23 2015 - [info] 192.168.3.115(192.168.3.115:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:23 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:23 2015 - [info]

Thu Jun 11 13:54:23 2015 - [info] \* Phase 3.2: Saving Dead Master's Binlog Phase..

Thu Jun 11 13:54:23 2015 - [info]

Thu Jun 11 13:54:24 2015 - [info] Fetching dead master's binary logs..

Thu Jun 11 13:54:24 2015 - [info] Executing command on the dead master 192.168.3.118(192.168.3.118:3306): save\_binary\_logs --command=save --start\_file=mysql-bin.000001 --start\_pos=145958297 --binlog\_dir=/var/lib/mysql --output\_file=/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53

Creating /tmp if not exists.. ok.

Concat binary/relay logs from mysql-bin.000001 pos 145958297 to mysql-bin.000001 EOF into /tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog ..

Dumping binlog format description event, from position 0 to 120.. ok.

Dumping effective binlog data from /var/lib/mysql/mysql-bin.000001 position 145958297 to tail(145969603).. ok.

Concat succeeded.

Thu Jun 11 13:54:26 2015 - [info] scp from root@192.168.3.118:/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog to local:/var/log/masterha/app1/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog succeeded.

Thu Jun 11 13:54:26 2015 - [info] HealthCheck: SSH to 192.168.3.110 is reachable.

Thu Jun 11 13:54:27 2015 - [info] HealthCheck: SSH to 192.168.3.115 is reachable.

Thu Jun 11 13:54:28 2015 - [info] HealthCheck: SSH to 192.168.3.117 is reachable.

Thu Jun 11 13:54:29 2015 - [info]

Thu Jun 11 13:54:29 2015 - [info] \* Phase 3.3: Determining New Master Phase..

Thu Jun 11 13:54:29 2015 - [info]

Thu Jun 11 13:54:29 2015 - [info] Finding the latest slave that has all relay logs for recovering other slaves..

Thu Jun 11 13:54:29 2015 - [info] Checking whether 192.168.3.110 has relay logs from the oldest position..

Thu Jun 11 13:54:29 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=find --latest\_mlf=mysql-bin.000001 --latest\_rmlp=145958297 --target\_mlf=mysql-bin.000001 --target\_rmlp=142466719 --server\_id=110 --workdir=/tmp --timestamp=20150611135413 --manager\_version=0.53 --relay\_log\_info=/var/lib/mysql/relay-log.info --relay\_dir=/var/lib/mysql/ :

Opening /var/lib/mysql/relay-log.info ... ok.

Relay log found at /var/lib/mysql, up to mysqld-relay-bin.000003

Fast relay log position search succeeded.

Target relay log file/position found. start\_file:mysqld-relay-bin.000003, start\_pos:79548.

Target relay log FOUND!

Thu Jun 11 13:54:30 2015 - [info] OK. 192.168.3.110 has all relay logs.

Thu Jun 11 13:54:30 2015 - [info] Searching new master from slaves..

Thu Jun 11 13:54:30 2015 - [info] Candidate masters from the configuration file:

Thu Jun 11 13:54:30 2015 - [info] 192.168.3.110(192.168.3.110:3306) Version=5.6.25-log (oldest major version between slaves) log-bin:enabled

Thu Jun 11 13:54:30 2015 - [info] Replicating from 192.168.3.118(192.168.3.118:3306)

Thu Jun 11 13:54:30 2015 - [info] Primary candidate for the new Master (candidate\_master is set)

Thu Jun 11 13:54:30 2015 - [info] Non-candidate masters:

Thu Jun 11 13:54:30 2015 - [info] Searching from candidate\_master slaves which have received the latest relay log events..

Thu Jun 11 13:54:30 2015 - [info] New master is 192.168.3.110(192.168.3.110:3306)

Thu Jun 11 13:54:30 2015 - [info] Starting master failover..

Thu Jun 11 13:54:30 2015 - [info]

From:

192.168.3.118 (current master)

+--192.168.3.110

+--192.168.3.115

+--192.168.3.117

To:

192.168.3.110 (new master)

+--192.168.3.115

+--192.168.3.117

Thu Jun 11 13:54:30 2015 - [info]

Thu Jun 11 13:54:30 2015 - [info] \* Phase 3.3: New Master Diff Log Generation Phase..

Thu Jun 11 13:54:30 2015 - [info]

Thu Jun 11 13:54:30 2015 - [info] This server has all relay logs. No need to generate diff files from the latest slave.

Thu Jun 11 13:54:30 2015 - [info] Sending binlog..

Thu Jun 11 13:54:31 2015 - [info] scp from local:/var/log/masterha/app1/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog to root@192.168.3.110:/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog succeeded.

Thu Jun 11 13:54:31 2015 - [info]

Thu Jun 11 13:54:31 2015 - [info] \* Phase 3.4: Master Log Apply Phase..

Thu Jun 11 13:54:31 2015 - [info]

Thu Jun 11 13:54:31 2015 - [info] \*NOTICE: If any error happens from this phase, manual recovery is needed.

Thu Jun 11 13:54:31 2015 - [info] Starting recovery on 192.168.3.110(192.168.3.110:3306)..

Thu Jun 11 13:54:31 2015 - [info] Generating diffs succeeded.

Thu Jun 11 13:54:31 2015 - [info] Waiting until all relay logs are applied.

Thu Jun 11 13:59:46 2015 - [info] done.

Thu Jun 11 13:59:46 2015 - [info] Getting slave status..

Thu Jun 11 13:59:46 2015 - [info] This slave(192.168.3.110)'s Exec\_Master\_Log\_Pos equals to Read\_Master\_Log\_Pos(mysql-bin.000001:145958297). No need to recover from Exec\_Master\_Log\_Pos.

Thu Jun 11 13:59:46 2015 - [info] Connecting to the target slave host 192.168.3.110, running recover script..

Thu Jun 11 13:59:46 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.3.110 --slave\_ip=192.168.3.110 --slave\_port=3306 --apply\_files=/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog --workdir=/tmp --target\_version=5.6.25-log --timestamp=20150611135413 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Thu Jun 11 13:59:49 2015 - [info]

MySQL client version is 5.6.25. Using --binary-mode.

Applying differential binary/relay log files /tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog on 192.168.3.110:3306. This may take long time...

Applying log files succeeded.

Thu Jun 11 13:59:49 2015 - [info] All relay logs were successfully applied.

Thu Jun 11 13:59:49 2015 - [info] Getting new master's binlog name and position..

Thu Jun 11 13:59:49 2015 - [info] mysql-bin.000001:11760

Thu Jun 11 13:59:49 2015 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.3.110', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000001', MASTER\_LOG\_POS=11760, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

Thu Jun 11 13:59:49 2015 - [info] Executing master IP activate script:

Thu Jun 11 13:59:49 2015 - [info] /usr/local/bin/master\_ip\_failover --command=start --ssh\_user=root --orig\_master\_host=192.168.3.118 --orig\_master\_ip=192.168.3.118 --orig\_master\_port=3306 --new\_master\_host=192.168.3.110 --new\_master\_ip=192.168.3.110 --new\_master\_port=3306

IN SCRIPT TEST====/etc/init.d/keepalived stop==/etc/init.d/keepalived start===

Enabling the VIP - 192.168.3.250 on the new master - 192.168.3.110

Thu Jun 11 13:59:49 2015 - [info] OK.

Thu Jun 11 13:59:49 2015 - [info] \*\* Finished master recovery successfully.

Thu Jun 11 13:59:49 2015 - [info] \* Phase 3: Master Recovery Phase completed.

Thu Jun 11 13:59:49 2015 - [info]

Thu Jun 11 13:59:49 2015 - [info] \* Phase 4: Slaves Recovery Phase..

Thu Jun 11 13:59:49 2015 - [info]

Thu Jun 11 13:59:49 2015 - [info] \* Phase 4.1: Starting Parallel Slave Diff Log Generation Phase..

Thu Jun 11 13:59:49 2015 - [info]

Thu Jun 11 13:59:49 2015 - [info] -- Slave diff file generation on host 192.168.3.115(192.168.3.115:3306) started, pid: 17896. Check tmp log /var/log/masterha/app1/192.168.3.115\_3306\_20150611135413.log if it takes time..

Thu Jun 11 13:59:49 2015 - [info] -- Slave diff file generation on host 192.168.3.117(192.168.3.117:3306) started, pid: 17897. Check tmp log /var/log/masterha/app1/192.168.3.117\_3306\_20150611135413.log if it takes time..

Thu Jun 11 13:59:52 2015 - [info]

Thu Jun 11 13:59:52 2015 - [info] Log messages from 192.168.3.117 ...

Thu Jun 11 13:59:52 2015 - [info]

Thu Jun 11 13:59:49 2015 - [info] Server 192.168.3.117 received relay logs up to: mysql-bin.000001:142482387

Thu Jun 11 13:59:49 2015 - [info] Need to get diffs from the latest slave(192.168.3.110) up to: mysql-bin.000001:145958297 (using the latest slave's relay logs)

Thu Jun 11 13:59:50 2015 - [info] Connecting to the latest slave host 192.168.3.110, generating diff relay log files..

Thu Jun 11 13:59:50 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=generate\_and\_send --scp\_user=root --scp\_host=192.168.3.117 --latest\_mlf=mysql-bin.000001 --latest\_rmlp=145958297 --target\_mlf=mysql-bin.000001 --target\_rmlp=142482387 --server\_id=110 --diff\_file\_readtolatest=/tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog --workdir=/tmp --timestamp=20150611135413 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --relay\_log\_info=/var/lib/mysql/relay-log.info --relay\_dir=/var/lib/mysql/

Thu Jun 11 13:59:52 2015 - [info]

Opening /var/lib/mysql/relay-log.info ... ok.

Relay log found at /var/lib/mysql, up to mysqld-relay-bin.000003

Fast relay log position search succeeded.

Target relay log file/position found. start\_file:mysqld-relay-bin.000003, start\_pos:95216.

Concat binary/relay logs from mysqld-relay-bin.000003 pos 95216 to mysqld-relay-bin.000003 EOF into /tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog ..

Dumping binlog format description event, from position 0 to 283.. ok.

Dumping effective binlog data from /var/lib/mysql/mysqld-relay-bin.000003 position 95216 to tail(3571126).. ok.

Concat succeeded.

Generating diff relay log succeeded. Saved at /tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog .

scp B-dev10:/tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog to root@192.168.3.117(22) succeeded.

Thu Jun 11 13:59:52 2015 - [info] Generating diff files succeeded.

Thu Jun 11 13:59:52 2015 - [info] End of log messages from 192.168.3.117.

Thu Jun 11 13:59:52 2015 - [info] -- Slave diff log generation on host 192.168.3.117(192.168.3.117:3306) succeeded.

Thu Jun 11 13:59:53 2015 - [info]

Thu Jun 11 13:59:53 2015 - [info] Log messages from 192.168.3.115 ...

Thu Jun 11 13:59:53 2015 - [info]

Thu Jun 11 13:59:49 2015 - [info] Server 192.168.3.115 received relay logs up to: mysql-bin.000001:142466719

Thu Jun 11 13:59:49 2015 - [info] Need to get diffs from the latest slave(192.168.3.110) up to: mysql-bin.000001:145958297 (using the latest slave's relay logs)

Thu Jun 11 13:59:50 2015 - [info] Connecting to the latest slave host 192.168.3.110, generating diff relay log files..

Thu Jun 11 13:59:50 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=generate\_and\_send --scp\_user=root --scp\_host=192.168.3.115 --latest\_mlf=mysql-bin.000001 --latest\_rmlp=145958297 --target\_mlf=mysql-bin.000001 --target\_rmlp=142466719 --server\_id=110 --diff\_file\_readtolatest=/tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog --workdir=/tmp --timestamp=20150611135413 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --relay\_log\_info=/var/lib/mysql/relay-log.info --relay\_dir=/var/lib/mysql/

Thu Jun 11 13:59:53 2015 - [info]

Opening /var/lib/mysql/relay-log.info ... ok.

Relay log found at /var/lib/mysql, up to mysqld-relay-bin.000003

Fast relay log position search succeeded.

Target relay log file/position found. start\_file:mysqld-relay-bin.000003, start\_pos:79548.

Concat binary/relay logs from mysqld-relay-bin.000003 pos 79548 to mysqld-relay-bin.000003 EOF into /tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog ..

Dumping binlog format description event, from position 0 to 283.. ok.

Dumping effective binlog data from /var/lib/mysql/mysqld-relay-bin.000003 position 79548 to tail(3571126).. ok.

Concat succeeded.

Generating diff relay log succeeded. Saved at /tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog .

scp B-dev10:/tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog to root@192.168.3.115(22) succeeded.

Thu Jun 11 13:59:53 2015 - [info] Generating diff files succeeded.

Thu Jun 11 13:59:53 2015 - [info] End of log messages from 192.168.3.115.

Thu Jun 11 13:59:53 2015 - [info] -- Slave diff log generation on host 192.168.3.115(192.168.3.115:3306) succeeded.

Thu Jun 11 13:59:53 2015 - [info] Generating relay diff files from the latest slave succeeded.

Thu Jun 11 13:59:53 2015 - [info]

Thu Jun 11 13:59:53 2015 - [info] \* Phase 4.2: Starting Parallel Slave Log Apply Phase..

Thu Jun 11 13:59:53 2015 - [info]

Thu Jun 11 13:59:53 2015 - [info] -- Slave recovery on host 192.168.3.115(192.168.3.115:3306) started, pid: 17910. Check tmp log /var/log/masterha/app1/192.168.3.115\_3306\_20150611135413.log if it takes time..

Thu Jun 11 13:59:53 2015 - [info] -- Slave recovery on host 192.168.3.117(192.168.3.117:3306) started, pid: 17911. Check tmp log /var/log/masterha/app1/192.168.3.117\_3306\_20150611135413.log if it takes time..

Thu Jun 11 14:14:14 2015 - [info]

Thu Jun 11 14:14:14 2015 - [info] Log messages from 192.168.3.117 ...

Thu Jun 11 14:14:14 2015 - [info]

Thu Jun 11 13:59:53 2015 - [info] Sending binlog..

Thu Jun 11 13:59:53 2015 - [info] scp from local:/var/log/masterha/app1/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog to root@192.168.3.117:/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog succeeded.

Thu Jun 11 13:59:53 2015 - [info] Starting recovery on 192.168.3.117(192.168.3.117:3306)..

Thu Jun 11 13:59:53 2015 - [info] Generating diffs succeeded.

Thu Jun 11 13:59:53 2015 - [info] Waiting until all relay logs are applied.

Thu Jun 11 13:59:53 2015 - [info] done.

Thu Jun 11 13:59:54 2015 - [info] Getting slave status..

Thu Jun 11 13:59:54 2015 - [info] This slave(192.168.3.117)'s Exec\_Master\_Log\_Pos(mysql-bin.000001:142481989) does not equal to Read\_Master\_Log\_Pos(mysql-bin.000001:142482387). It is likely that relay log was cut during transaction. Need to recover from Exec\_Master\_Log\_Pos.

Thu Jun 11 13:59:54 2015 - [info] Saving local relay logs from exec pos to read pos on 192.168.3.117: from mysqld-relay-bin.000002:142482152 to the end of the relay log..

Thu Jun 11 13:59:54 2015 - [info] Executing command : save\_binary\_logs --command=save --start\_file=mysqld-relay-bin.000002 --start\_pos=142482152 --output\_file=/tmp/relay\_from\_exec\_to\_read\_192.168.3.117\_3306\_20150611135413.binlog --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --relay\_log\_info=/var/lib/mysql/relay-log.info --binlog\_dir=/var/lib/mysql/

Thu Jun 11 13:59:54 2015 - [info]

Creating /tmp if not exists.. ok.

Concat binary/relay logs from mysqld-relay-bin.000002 pos 142482152 to mysqld-relay-bin.000002 EOF into /tmp/relay\_from\_exec\_to\_read\_192.168.3.117\_3306\_20150611135413.binlog ..

Dumping binlog format description event, from position 0 to 283.. ok.

Dumping effective binlog data from /var/lib/mysql/mysqld-relay-bin.000002 position 142482152 to tail(142482550).. ok.

Concat succeeded.

Thu Jun 11 13:59:54 2015 - [info] Connecting to the target slave host 192.168.3.117, running recover script..

Thu Jun 11 13:59:54 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.3.117 --slave\_ip=192.168.3.117 --slave\_port=3306 --apply\_files=/tmp/relay\_from\_exec\_to\_read\_192.168.3.117\_3306\_20150611135413.binlog,/tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog,/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog --workdir=/tmp --target\_version=5.6.25-log --timestamp=20150611135413 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Thu Jun 11 14:14:12 2015 - [info]

Concat all apply files to /tmp/total\_binlog\_for\_192.168.3.117\_3306.20150611135413.binlog ..

Copying the first binlog file /tmp/relay\_from\_exec\_to\_read\_192.168.3.117\_3306\_20150611135413.binlog to /tmp/total\_binlog\_for\_192.168.3.117\_3306.20150611135413.binlog.. ok.

Dumping binlog head events (rotate events), skipping format description events from /tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog.. dumped up to pos 283. ok.

/tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog has effective binlog events from pos 283.

Dumping effective binlog data from /tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog position 283 to tail(3476193).. ok.

Dumping binlog head events (rotate events), skipping format description events from /tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog.. dumped up to pos 120. ok.

/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog has effective binlog events from pos 120.

Dumping effective binlog data from /tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog position 120 to tail(11426).. ok.

Concat succeeded.

All apply target binary logs are concatinated at /tmp/total\_binlog\_for\_192.168.3.117\_3306.20150611135413.binlog .

MySQL client version is 5.6.25. Using --binary-mode.

Applying differential binary/relay log files /tmp/relay\_from\_exec\_to\_read\_192.168.3.117\_3306\_20150611135413.binlog,/tmp/relay\_from\_read\_to\_latest\_192.168.3.117\_3306\_20150611135413.binlog,/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog on 192.168.3.117:3306. This may take long time...

Applying log files succeeded.

Thu Jun 11 14:14:12 2015 - [info] All relay logs were successfully applied.

Thu Jun 11 14:14:12 2015 - [info] Resetting slave 192.168.3.117(192.168.3.117:3306) and starting replication from the new master 192.168.3.110(192.168.3.110:3306)..

Thu Jun 11 14:14:14 2015 - [info] Executed CHANGE MASTER.

Thu Jun 11 14:14:14 2015 - [info] Slave started.

Thu Jun 11 14:14:14 2015 - [info] End of log messages from 192.168.3.117.

Thu Jun 11 14:14:14 2015 - [info] -- Slave recovery on host 192.168.3.117(192.168.3.117:3306) succeeded.

Thu Jun 11 14:14:37 2015 - [info]

Thu Jun 11 14:14:37 2015 - [info] Log messages from 192.168.3.115 ...

Thu Jun 11 14:14:37 2015 - [info]

Thu Jun 11 13:59:53 2015 - [info] Sending binlog..

Thu Jun 11 13:59:53 2015 - [info] scp from local:/var/log/masterha/app1/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog to root@192.168.3.115:/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog succeeded.

Thu Jun 11 13:59:53 2015 - [info] Starting recovery on 192.168.3.115(192.168.3.115:3306)..

Thu Jun 11 13:59:53 2015 - [info] Generating diffs succeeded.

Thu Jun 11 13:59:53 2015 - [info] Waiting until all relay logs are applied.

Thu Jun 11 13:59:53 2015 - [info] done.

Thu Jun 11 13:59:53 2015 - [info] Getting slave status..

Thu Jun 11 13:59:53 2015 - [info] This slave(192.168.3.115)'s Exec\_Master\_Log\_Pos(mysql-bin.000001:142466642) does not equal to Read\_Master\_Log\_Pos(mysql-bin.000001:142466719). It is likely that relay log was cut during transaction. Need to recover from Exec\_Master\_Log\_Pos.

Thu Jun 11 13:59:53 2015 - [info] Saving local relay logs from exec pos to read pos on 192.168.3.115: from mysqld-relay-bin.000002:142466805 to the end of the relay log..

Thu Jun 11 13:59:53 2015 - [info] Executing command : save\_binary\_logs --command=save --start\_file=mysqld-relay-bin.000002 --start\_pos=142466805 --output\_file=/tmp/relay\_from\_exec\_to\_read\_192.168.3.115\_3306\_20150611135413.binlog --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --relay\_log\_info=/var/lib/mysql/relay-log.info --binlog\_dir=/var/lib/mysql/

Thu Jun 11 13:59:54 2015 - [info]

Creating /tmp if not exists.. ok.

Concat binary/relay logs from mysqld-relay-bin.000002 pos 142466805 to mysqld-relay-bin.000002 EOF into /tmp/relay\_from\_exec\_to\_read\_192.168.3.115\_3306\_20150611135413.binlog ..

Dumping binlog format description event, from position 0 to 283.. ok.

Dumping effective binlog data from /var/lib/mysql/mysqld-relay-bin.000002 position 142466805 to tail(142466882).. ok.

Concat succeeded.

Thu Jun 11 13:59:54 2015 - [info] Connecting to the target slave host 192.168.3.115, running recover script..

Thu Jun 11 13:59:54 2015 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.3.115 --slave\_ip=192.168.3.115 --slave\_port=3306 --apply\_files=/tmp/relay\_from\_exec\_to\_read\_192.168.3.115\_3306\_20150611135413.binlog,/tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog,/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog --workdir=/tmp --target\_version=5.6.25-log --timestamp=20150611135413 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Thu Jun 11 14:14:35 2015 - [info]

Concat all apply files to /tmp/total\_binlog\_for\_192.168.3.115\_3306.20150611135413.binlog ..

Copying the first binlog file /tmp/relay\_from\_exec\_to\_read\_192.168.3.115\_3306\_20150611135413.binlog to /tmp/total\_binlog\_for\_192.168.3.115\_3306.20150611135413.binlog.. ok.

Dumping binlog head events (rotate events), skipping format description events from /tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog.. dumped up to pos 283. ok.

/tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog has effective binlog events from pos 283.

Dumping effective binlog data from /tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog position 283 to tail(3491861).. ok.

Dumping binlog head events (rotate events), skipping format description events from /tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog.. dumped up to pos 120. ok.

/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog has effective binlog events from pos 120.

Dumping effective binlog data from /tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog position 120 to tail(11426).. ok.

Concat succeeded.

All apply target binary logs are concatinated at /tmp/total\_binlog\_for\_192.168.3.115\_3306.20150611135413.binlog .

MySQL client version is 5.6.25. Using --binary-mode.

Applying differential binary/relay log files /tmp/relay\_from\_exec\_to\_read\_192.168.3.115\_3306\_20150611135413.binlog,/tmp/relay\_from\_read\_to\_latest\_192.168.3.115\_3306\_20150611135413.binlog,/tmp/saved\_master\_binlog\_from\_192.168.3.118\_3306\_20150611135413.binlog on 192.168.3.115:3306. This may take long time...

Applying log files succeeded.

Thu Jun 11 14:14:35 2015 - [info] All relay logs were successfully applied.

Thu Jun 11 14:14:35 2015 - [info] Resetting slave 192.168.3.115(192.168.3.115:3306) and starting replication from the new master 192.168.3.110(192.168.3.110:3306)..

Thu Jun 11 14:14:37 2015 - [info] Executed CHANGE MASTER.

Thu Jun 11 14:14:37 2015 - [info] Slave started.

Thu Jun 11 14:14:37 2015 - [info] End of log messages from 192.168.3.115.

Thu Jun 11 14:14:37 2015 - [info] -- Slave recovery on host 192.168.3.115(192.168.3.115:3306) succeeded.

Thu Jun 11 14:14:37 2015 - [info] All new slave servers recovered successfully.

Thu Jun 11 14:14:37 2015 - [info]

Thu Jun 11 14:14:37 2015 - [info] \* Phase 5: New master cleanup phease..

Thu Jun 11 14:14:37 2015 - [info]

Thu Jun 11 14:14:37 2015 - [info] Resetting slave info on the new master..

Thu Jun 11 14:14:37 2015 - [info] 192.168.3.110: Resetting slave info succeeded.

Thu Jun 11 14:14:37 2015 - [info] Master failover to 192.168.3.110(192.168.3.110:3306) completed successfully.

Thu Jun 11 14:14:37 2015 - [info] Deleted server1 entry from /etc/masterha/app1.cnf .

Thu Jun 11 14:14:37 2015 - [info]

----- Failover Report -----

app1: MySQL Master failover 192.168.3.118 to 192.168.3.110 succeeded

Master 192.168.3.118 is down!

Check MHA Manager logs at B-dev23:/var/log/masterha/app1/manager.log for details.

Started automated(non-interactive) failover.

Invalidated master IP address on 192.168.3.118.

The latest slave 192.168.3.110(192.168.3.110:3306) has all relay logs for recovery.

Selected 192.168.3.110 as a new master.

192.168.3.110: OK: Applying all logs succeeded.

192.168.3.110: OK: Activated master IP address.

192.168.3.117: Generating differential relay logs up to 192.168.3.110 succeeded.

192.168.3.115: Generating differential relay logs up to 192.168.3.110 succeeded.

Generating relay diff files from the latest slave succeeded.

192.168.3.117: OK: Applying all logs succeeded. Slave started, replicating from 192.168.3.110.

192.168.3.115: OK: Applying all logs succeeded. Slave started, replicating from 192.168.3.110.

192.168.3.110: Resetting slave info succeeded.

Master failover to 192.168.3.110(192.168.3.110:3306) completed successfully.

看到最后的Master failover to 192.168.3.110(192.168.3.110:3306) completed successfully.说明备选master现在已经上位了。

从上面的输出可以看出整个MHA的切换过程，共包括以下的步骤：

• 连接主机三次失败确认主机宕机，摘除宕机的vip

• 检查从库中使用的最新binlog file/pos

• 检查从库中使用的最老binlog file/pos

• 获取宕掉主机的binlog：根据步骤1获取的binlog位置开始保存未同步到从机的binlog(导出binlog描述事件，导出影响的binlog数据)；将导出的binlog发送到mha-manage服务器指定目录下

• healthcheck:监控ssh到其他的机器是否正常

• 检查最新binlog file/pos的从机是否有需要恢复其他从机所需的所有relay log

• 如果其他从机跟最新从机pos不一致，需要最新从机的relay log进行一致性恢复

• 检查是否设置候选主机

• 开始候选主机切换：先检查候选主机是否有所有的relay log，没有首先从最新从机恢复，然后将mha收到宕机主机缺省的binlog发送给候选主机，识别差异的relay log并将差异的事件应用所有的relay log，执行vip漂移到候选主机，候选主机恢复结束

• 其他从机relay log恢复：先检查从机是否有所有的relay log，没有首先从最新从机恢复，然后将mha收到宕机主机缺省的binlog发送给从机，识别差异的relay log并将差异的事件应用所有的relay log

• 将所有从机指向为候选主机

启动MHA Manger监控，查看集群里面现在谁是master。

（7）检查切换后数据是主从数据是否一致

在主机和候选主机上分别安装percona-toolkit，安装流程及使用文档请查看percona-toolkit 简明教程。

这里简单说下pt-table-checksum 的工作原理： pt-table-checksum 在主上执行检查语句在线检查mysql复制的一致性， 生成replace 语句，然后通过复制传递到从，再通过update更新master\_src 的值。通过检测从上this\_src 和master\_src 的值从而判断复制是否一致。注意： 使用的时候选择业务地峰的时候运行，因为运行的时候会造成表的部分记录锁定。 使用–max-load 来指定最大的负载情况，如果达到那个负载这个暂停运行。 如果发现有不一致的数据，可以使用pt-table-sync 工具来修复。

在切换后的新主机上使用命令：

[root@192.168.3.110 ~]# pt-table-checksum --nocheck-replication-filters --chunk-size=1000000 --no-check-binlog-format --replicate=sps.checksums --databases=sps h=127.0.0.1,u=root,p=123456,P=3306

TS ERRORS DIFFS ROWS CHUNKS SKIPPED TIME TABLE

06-23T18:06:38 0 0 0 1 0 3.885 sps.sps\_account

06-23T18:06:42 0 0 3 1 0 3.118 sps.sps\_activiti\_goods

06-23T18:06:46 0 0 0 1 0 4.613 sps.sps\_activiti\_prop

06-23T18:06:49 0 0 2 1 0 3.013 sps.sps\_activiti\_re

06-23T18:06:52 0 0 3 1 0 2.858 sps.sps\_activity

06-23T18:06:55 0 0 1 1 0 3.359 sps.sps\_app\_version

06-23T18:06:58 0 0 4 1 0 2.894 sps.sps\_brand

06-23T18:07:02 0 0 3272 1 0 3.367 sps.sps\_district

06-23T18:07:05 0 0 16 1 0 2.935 sps.sps\_goods

06-23T18:07:08 0 0 6 1 0 3.761 sps.sps\_goods\_brand\_cat

06-23T18:07:12 0 0 332 1 0 3.418 sps.sps\_goods\_category

06-23T18:07:16 0 0 0 1 0 4.175 sps.sps\_goods\_category\_template

06-23T18:07:20 0 0 5 1 0 4.061 sps.sps\_goods\_comment

06-23T18:07:24 0 0 9 1 0 3.784 sps.sps\_goods\_desc

06-23T18:07:27 0 0 4 1 0 3.487 sps.sps\_goods\_favorite

06-23T18:07:33 0 0 14 1 0 5.847 sps.sps\_goods\_media

06-23T18:07:37 0 0 2 1 0 3.811 sps.sps\_goods\_recommend

06-23T18:07:41 0 0 2 1 0 3.623 sps.sps\_goods\_recommend\_group

06-23T18:07:44 0 0 4 1 0 3.402 sps.sps\_goods\_recommend\_group\_cate

06-23T18:07:48 0 0 12 1 0 3.662 sps.sps\_goods\_sku\_attribute\_value

06-23T18:07:52 0 0 19 1 0 3.951 sps.sps\_goods\_stock

06-23T18:07:55 0 0 6 1 0 3.557 sps.sps\_goods\_template

06-23T18:07:58 0 0 2 1 0 3.226 sps.sps\_goods\_tmpl\_sku\_attribute\_line

06-23T18:08:03 0 0 5 1 0 4.183 sps.sps\_goods\_tmpl\_sku\_attribute\_line\_value

06-23T18:08:08 0 0 208733 1 0 5.422 sps.sps\_member

06-23T18:08:11 0 0 2 1 0 3.304 sps.sps\_member\_account\_seq

06-23T18:08:15 0 0 27721 1 0 3.575 sps.sps\_member\_address

06-23T18:08:18 0 0 2 1 0 3.116 sps.sps\_member\_baby\_birthday

06-23T18:08:21 0 0 1 1 0 2.876 sps.sps\_member\_grade\_change\_info

06-23T18:08:24 0 0 0 1 0 3.563 sps.sps\_member\_grade\_dict

06-23T18:08:28 0 0 16 1 0 3.848 sps.sps\_member\_points\_seq

06-23T18:08:31 0 0 0 1 0 2.958 sps.sps\_menu

06-23T18:08:35 0 0 0 1 0 3.687 sps.sps\_message

06-23T18:08:39 0 0 5 1 0 3.589 sps.sps\_message\_text

06-23T18:08:41 0 0 746 1 0 2.848 sps.sps\_order

06-23T18:08:46 0 0 94 1 0 4.529 sps.sps\_order\_goods\_detail

06-23T18:08:49 0 0 23 1 0 3.030 sps.sps\_order\_status

06-23T18:08:52 0 0 0 1 0 3.390 sps.sps\_payment

06-23T18:08:57 0 0 13 1 0 4.559 sps.sps\_promotion

06-23T18:09:00 0 0 15 1 0 2.986 sps.sps\_promotion\_goods

06-23T18:09:04 0 0 0 1 0 4.158 sps.sps\_role

06-23T18:09:08 0 0 0 1 0 3.880 sps.sps\_role\_menu

06-23T18:09:11 0 0 16 1 0 2.997 sps.sps\_seller

06-23T18:09:15 0 0 4 1 0 3.808 sps.sps\_shop\_trolley

06-23T18:09:18 0 0 6 1 0 3.409 sps.sps\_sign\_in

06-23T18:09:21 0 0 2 1 0 3.067 sps.sps\_sku\_attribute

06-23T18:09:25 0 0 5 1 0 3.871 sps.sps\_sku\_attribute\_value

06-23T18:09:30 0 0 1 1 0 4.466 sps.sps\_store

06-23T18:09:33 0 0 0 1 0 3.378 sps.sps\_store\_account

06-23T18:09:36 0 0 6 1 0 3.331 sps.sps\_sys\_config

06-23T18:09:40 0 0 8 1 0 3.391 sps.sps\_sys\_message

06-23T18:09:43 0 0 14 1 0 3.182 sps.sps\_ticket

06-23T18:09:46 0 0 12 1 0 3.614 sps.sps\_ticket\_goods

06-23T18:09:50 0 0 140 1 0 3.419 sps.sps\_ticket\_his

06-23T18:09:53 0 0 12 1 0 3.041 sps.sps\_ticket\_prop

06-23T18:09:56 0 0 0 1 0 3.091 sps.sps\_user\_role

06-23T18:10:00 0 0 5 1 0 3.592 sps.sps\_warehouse

从结果中，我们可以看到DIFFS都为0，说明主从数据一致，故障切换流程没有数据丢失。

#### 手动Failover

手动failover，这种场景意味着在业务上没有启用MHA自动切换功能，当主服务器故障时，人工手动调用MHA来进行故障切换操作，具体命令如下：

注意：如果MHA Manager检测到没有dead的server，将报错，并结束failover：

Mon Jun 21 21:23:33 2015 - [info] Dead Servers:

Mon Jun 21 21:23:33 2015 - [error][/usr/local/share/perl5/MHA/MasterFailover.pm, ln181] None of server is dead. Stop failover.

Mon Jun 21 21:23:33 2015 - [error][/usr/local/share/perl5/MHA/ManagerUtil.pm, ln178] Got ERROR: at /usr/local/bin/masterha\_master\_switch line 53

进行手动切换命令如下：

[root@192.168.3.123~]# masterha\_master\_switch --master\_state=dead --conf=/etc/masterha/app1.cnf --dead\_master\_host=192.168.3.110 --dead\_master\_port=3306 --new\_master\_host=192.168.3.118 --new\_master\_port=3306 --ignore\_last\_failover

#### 修复宕机的Master

通常情况下自动切换以后，原master可能已经废弃掉，待原master主机修复后，如果数据完整的情况下，可能想把原来master重新作为新主库的slave，这时我们可以借助当时自动切换时刻的MHA日志来完成对原master的修复。下面是提取相关日志的命令：

# grep -i "All other slaves should start" manager.log

Mon Jun 21 22:28:33 2015 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.3.118', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000022', MASTER\_LOG\_POS=506716, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

获取上述信息以后，就可以直接在修复后的master上执行change master to相关操作，重新作为从库了。

### 4.14PT-TABLE-CHECKSUM测试

在master上进行复制一致性检查，在percona数据库中创建dsns表，并插入dsn值：

CREATE TABLE `dsns` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`parent\_id` int(11) DEFAULT NULL,

`dsn` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

mysql> insert into dsns(dsn) values('h=10.252.124.43,u=rkylinadmin,p=rkylin\_capital!1');

mysql> insert into dsns(dsn) values('h= 10.168.106.17, u=rkylinadmin,p=rkylin\_capital!1');

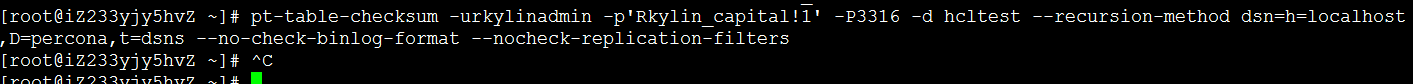
1. 在master上创建测试库hcltest，在slave上hcltest库下创建表a：

use hcltest;

create table a(id int);

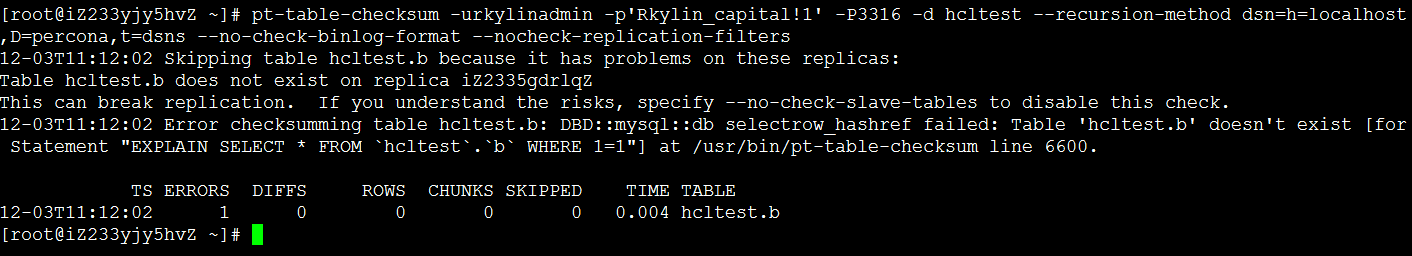
在master上检查一致性：

pt-table-checksum -d hcltest --recursion-method dsn=h=localhost,D=percona,t=dsns --no-check-binlog-format --nocheck-replication-filters



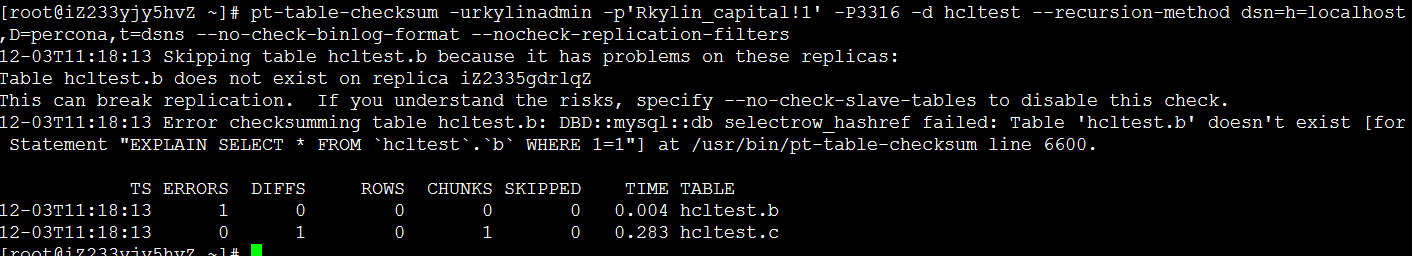
发现并不能监测出不一致，就是说主库如果没有表a，从库有表a，是监测不出来的。

1. 在master上创建表b(id int)，在slave上drop table b，检查：



会发现监测出表b在slave上不存在。

1. 在master上创建表c(id int)，在slave上insert into c value(1);，检查：



监测到c表有不一致。