MySQL 数据库性能对比测试

目 录

1. 制定说明

1.1. 适用范围

本测试评估 MySQL5.7 数据库在麒麟操作系统的性能，测试结果不具备参考性，仅考虑测试脚本。

1.2. 测试服务器

主机	1	2
操作系统版本	Kylin Linux Advanced Server release V10 (Tercel)	Red Hat Enterprise Linux Server release 7.8 (Maipo)
内核版本	4.19.90-23.6.v2101.ky10.x86_64	3.10.0-1160.15.2.el7.x86_64
CPU 核心数	8	8
内存	16GB	16GB
硬盘	同宿主机硬盘	同宿主机硬盘

2. 系统参数

2.1. 参数对比

下面是与 MySQL 数据库性能相关的服务器默认参数：

	麒麟 OS	RH 7
网络		
/proc/sys/net/ipv4/tcp_max_syn_backlog	2048	32768

/proc/sys/net/core/somaxconn	512	32768
/proc/sys/net/core/rmem_max	2097152	4194304
/proc/sys/net/core/wmem_max	2097152	4194304
/proc/sys/net/ipv4/tcp_rmem	4096|131072|6291456	4096|87380|6291456
/proc/sys/net/ipv4/tcp_wmem	4096|16384 |4194304	4096|16384|4194304
/proc/sys/net/ipv4/tcp_max_tw_buckets	65535	65535
/proc/sys/net/ipv4/tcp_keepalive_time	7200	150
/proc/sys/net/ipv4/tcp_keepalive_intvl	75	6
IO		
/sys/block/sdb/queue/scheduler	[mq-deadline] kyber bfq none	noop [deadline] cfq
/sys/block/sda/queue/nr_requests	254	128
/proc/sys/vm/nr_hugepages	0	0
缓存		
/proc/sys/vm/swappiness	10	10
/proc/sys/vm/dirty_ratio	40	30

2.2. 参数描述

tcp_max_syn_backlog	SYN 客户端队列长度
somaxconn	ESTABELLISHED 客户端队列长度
rmem_max	接受 SOCKET 缓冲区大小
wmem_max	发送 SOCKET 缓冲区大小
tcp_rmem	TCP 读缓冲大小：min、default、max
tcp_wmem	TCP 写缓冲大小：min、default、max
tcp_max_tw_buckets	系统同时保持 TIME_WAIT 套接字最大数量
tcp_keepalive_time	当 keepalive 启用时，TCP 发送消息的频度
tcp_keepalive_intvl	当 keepalive 启用时，TCP 未响应重发间隔
scheduler	配置 IO 调度，建议设置 deadline
nr_requests	提升磁盘吞吐量，建议设置 2048
nr_hugepage	传统大页和透明大页，建议关闭
swappiness	该参数决定在内存不够用的情况下，内核对 swap 分区的使用频率，0 表示最大限度使用物理内存，数据库应用建议设置为 0
dirty_ratio	内存中脏数据百分比

3. 系统测试

由于两边系统参数的不同，所以首先将麒麟操作系统的上述配置对齐为 Redhat7。系统测试工具选择 sysbench1.0.17.

3.1. CPU 计算测试

```
SHELL> ./sysbench cpu --threads=xx run
# 10 秒内，通过对素数求和运算估计 CPU 性能
```

线程数为 1

	麒麟 V10	RH7
total time	10.0004s	10.0009s
events	10634	10485
avg（ms）	0.94	0.95
max（ms）	3.56	3.61
95th percentile（ms）	1.01	1.03

线程数为 8

	麒麟 V10	RH7
total time	10.0009s	10.0009s
events	81670	80262
avg（ms）	0.98	1.00
max（ms）	7.08	16.19
95th percentile（ms）	1.01	1.12

线程数为 16

	麒麟 V10	RH7
total time	10.0011s	10.0011s
events	81723	81308
avg（ms）	1.96	1.96
max（ms）	13.00	101.99
95th percentile（ms）	5.00	13.95

根据上述结果：
当进行核心数以下的线程测试中，两者几乎不存在差别；
当线程数超过核心数时，麒麟 V10 在整体计算性能方面更稳定一些。

3.2. 文件 IO 吞吐测试

数据准备：

```
SHELL> ./sysbench fileio --file-num=8 --file-total-size=16G prepare
```

IO 测试

1. 随机读

```
SHELL> ./sysbench fileio --file-total-size=16G --file-test-mode=rndrd \
```

```
--time=180 --events=100000000 --threads=8 --file-num=8          \
--file-extra-flags=direct --file-fsync-freq=1 --file-block-size=4096 run
```

rndrd

	麒麟 V10	RH7
total time	180.0010s	180.0006s
events	2036206	2220056
throughput	44.19 read, MiB/s	48.18 read, MiB/s
avg（ms）	0.71	0.65
max（ms）	329.20	52.47
95th percentile（ms）	1.39	1.37

2.顺序读

```
SHELL> ./sysbench fileio --file-total-size=16G --file-test-mode=seqrd \
--time=180 --events=100000000 --threads=8 --file-num=8          \
--file-extra-flags=direct --file-fsync-freq=1 --file-block-size=4096 run
```

seqrd

	麒麟 V10	RH7
total time	180.0144s	180.0004s
events	2988996	3165773
throughput	64.86 read, MiB/s	68.70 read, MiB/s
avg（ms）	0.48	0.45
max（ms）	64.43	67.38
95th percentile（ms）	0.80	0.54

3.随机写

```
SHELL> ./sysbench fileio --file-total-size=16G --file-test-mode=rndwr \
--time=180 --events=100000000 --threads=8 --file-num=8          \
--file-extra-flags=direct --file-fsync-freq=1 --file-block-size=4096 run
```

rndrw

	麒麟 V10	RH7
total time	180.0005s	180.0018s
events	6832492	3590929
throughput	16.47 written, MiB/s	8.66 written, MiB/s
avg（ms）	0.21	0.40
max（ms）	64.14	36.11
95th percentile（ms）	0.92	1.61

4.顺序写

```
SHELL> ./sysbench fileio --file-total-size=16G --file-test-mode=seqwr \
--time=180 --events=100000000 --threads=8 --file-num=8          \
```

--file-extra-flags=direct --file-fsync-freq=1 --file-block-size=4096 run

seqwr

	麒麟 V10	RH7
total time	180.0120s	180.0111s
events	1104895	1093097
throughput	2.66 written, MiB/s	2.64 written, MiB/s
avg（ms）	1.30	1.32
max（ms）	165.74	745.74
95th percentile（ms）	10.46	2.26

根据上述结果：
在文件 IO 随机写测试中，麒麟 V10 在文件 IO 吞吐表现明显优于 Redhat7 操作系统；
其他情况并无明显区别。

3.3. 线程测试

SHELL> ./sysbench threads --threads=xx --thread-yields=100 --thread-locks=2 run # 10 秒内，测试多线程的性能

线程 64

	麒麟 V10	RH7
total time	10.0043s	10.0024s
events	145471	153318
avg（ms）	4.40	4.17
max（ms）	67.36	56.33
95th percentile（ms）	16.41	16.12

线程 128

	麒麟 V10	RH7
total time	10.0080s	10.0062s
events	150629	146980
avg（ms）	7.93	8.71
max（ms）	99.94	128.23
95th percentile（ms）	31.94	34.95

线程 256

	麒麟 V10	RH7
total time	10.0080s	10.0062s
events	145859	138710
avg（ms）	17.56	18.47

max（ms）	210.36	245.25
95th percentile（ms）	71.83	73.13

根据上述结果，在多线程测试中，两者性能接近。

4. 数据库 OLTP 测试

MySQL 数据库适合承载 OLTP 型事务，通过 sysbench 工具对比不同操作系统下的性能差异。该项测试基于 MySQL5.7.32 主从半同步复制集群，内存池为 4G。

创建测试库：

```
mysql> create database sysbenchtest;
```

数据准备：

```
SHELL> /mysqldata/sysbench/sysbench-1.0.17/bin/sysbench
/mysqldata/sysbench/sysbench-1.0.17/share/sysbench/oltp_read_write.lua --mysql-host=127.0.0.1
--mysql-port=3306 \
--mysql-user=admin \
--mysql-password='P@ssw0rd' \
--mysql-db=sysbenchtest \
--table-size=1000000 \
--tables=4 \
--threads=8 prepare
```

读写测试：

```
SHELL> /mysqldata/sysbench/sysbench-1.0.17/bin/sysbench
/mysqldata/sysbench/sysbench-1.0.17/share/sysbench/oltp_read_write.lua --mysql-host=127.0.0.1
--mysql-port=3306 \
--mysql-user=admin \
--mysql-password='P@ssw0rd' \
--mysql-db=sysbenchtest \
--table-size=1000000 \
--tables=4 \
--threads=xx \
--time=180 \
--report-interval=10 run
```

线程 16

	麒麟 V10	RH7
total time	180.0109s	180.0303s
tps	1354.88	1270.65
qps	27097.59	25413.05
avg（ms）	11.81	12.59
max（ms）	143.28	88.59

95th percentile（ms）	17.63	18.61

线程 32

	麒麟 V10	RH7
total time	180.0109s	180.0172s
tps	1702.26	1607.23
qps	34045.27	32144.57
avg（ms）	18.80	19.91
max（ms）	101.28	112.67
95th percentile（ms）	27.66	30.26

线程 64

	麒麟 V10	RH7
total time	180.0297s	180.0180s
tps	1747.93	1725.87
qps	34958.52	34517.39
avg（ms）	36.61	37.08
max（ms）	138.18	199.25
95th percentile（ms）	54.83	61.08

线程 128

	麒麟 V10	RH7
total time	180.0702s	180.0345s
tps	1818.59	1773.29
qps	36371.86	35465.80
avg（ms）	70.36	72.17
max（ms）	219.68	388.49
95th percentile（ms）	106.75	123.28

根据上述结果，在 OLTP 测试中，两者性能接近。

5. 数据库 OLAP 测试

TPC-H 是 TPC 组织提供的 OLAP 测试工具包，用于评估数据库的分析型查询能力。TPC-H 查询包含 8 张数据表，22 条复杂 SQL 查询，并且大多数查询包含若干表 JOIN、子查询和 GROUP BY 聚合等。

安装部署如下：

1.上传 TPC-H 工具包
2.解压并打开 dbgen 目录
SHELL> cd /mysqldata/tpc-h/TPC-H_Tools_v3.0.0/dbgen
3.复制 makefile 文件
SHELL> cp makefile.suite makefile
4.修改 makefile 文件中的 CC\DATABASE\MACHINE\WORKLOAD 参数

```
SHELL> vim makefile
#################
## CHANGE NAME OF ANSI COMPILER HERE
#################
CC          =          gcc
# Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata)
#                                  SQLSERVER, SYBASE, ORACLE, VECTORWISE
# Current values for MACHINE are:    ATT, DOS, HP, IBM, ICL, MVS,
#                                  SGI, SUN, U2200, VMS, LINUX, WIN32
# Current values for WORKLOAD are:    TPCH
DATABASE= MYSQL
MACHINE = LINUX
WORKLOAD = TPCH
#
5.新增 tpcd.h 文件中的宏定义
SHELL> vim tpcd.h
#ifdef    MYSQL
#define GEN_QUERY_PLAN    ""
#define START_TRAN          "START TRANSACTION"
#define END_TRAN            "COMMIT"
#define SET_OUTPUT          ""
#define SET_ROWCOUNT       "limit %d;\n"
#define SET_DBASE           "use %s;\n"
#endif
6.完成编译
SHELL> make
```

工具使用：

```
dbgen：数据生成工具
qgen：SQL 生成工具
```

数据准备：

```
SHELL> ./dbgen -s 10
#  表示 10G 数据
SHELL> ls -lh *.tbl
-rw-r--r-- 1 root root 234M Aug 17 10:54 customer.tbl
-rw-r--r-- 1 root root 7.3G Aug 17 10:54 lineitem.tbl
-rw-r--r-- 1 root root 2.2K Aug 17 10:54 nation.tbl
-rw-r--r-- 1 root root 1.7G Aug 17 10:54 orders.tbl
-rw-r--r-- 1 root root 1.2G Aug 17 10:54 partsupp.tbl
-rw-r--r-- 1 root root 233M Aug 17 10:54 part.tbl
-rw-r--r-- 1 root root    389 Aug 17 10:54 region.tbl
-rw-r--r-- 1 root root    14M Aug 17 10:54 supplier.tbl
SHELL> ls -lh *.ddl
```

```
-rw-r--r-- 1 root root 3.8K Dec  5  2018 dss.ddl
```

导入数据库：

```
mysql> create database tpchtest;
mysql> use tpchtest;
mysql> source ./dss.ddl;
mysql> load data local infile 'customer.tbl'  into table customer fields terminated by '|';
mysql> load data local infile 'lineitem.tbl'  into table lineitem fields terminated by '|';
mysql> load data local infile 'nation.tbl'  into table nation fields terminated by '|';
mysql> load data local infile 'orders.tbl'  into table orders fields terminated by '|';
mysql> load data local infile 'partsupp.tbl'  into table partsupp fields terminated by '|';
mysql> load data local infile 'part.tbl'  into table part fields terminated by '|';
mysql> load data local infile 'region.tbl'  into table region fields terminated by '|';
mysql> load data local infile 'supplier.tbl'  into table supplier fields terminated by '|';
```

创建约束和索引：

```
# 创建外键
use tpchtest;

-- For table REGION
ALTER TABLE REGION
ADD PRIMARY KEY (R_REGIONKEY);
-- For table NATION
ALTER TABLE NATION
ADD PRIMARY KEY (N_NATIONKEY);
ALTER TABLE NATION
ADD FOREIGN KEY NATION_FK1 (N_REGIONKEY) references REGION(R_REGIONKEY);
COMMIT WORK;
-- For table PART
ALTER TABLE PART
ADD PRIMARY KEY (P_PARTKEY);
COMMIT WORK;
-- For table SUPPLIER
ALTER TABLE SUPPLIER
ADD PRIMARY KEY (S_SUPPKEY);
ALTER TABLE SUPPLIER
ADD FOREIGN KEY SUPPLIER_FK1 (S_NATIONKEY) references
NATION(N_NATIONKEY);
COMMIT WORK;
-- For table PARTSUPP
ALTER TABLE PARTSUPP
ADD PRIMARY KEY (PS_PARTKEY,PS_SUPPKEY);
COMMIT WORK;
-- For table CUSTOMER
```

```
ALTER TABLE CUSTOMER
ADD PRIMARY KEY (C_CUSTKEY);
ALTER TABLE CUSTOMER
ADD FOREIGN KEY CUSTOMER_FK1 (C_NATIONKEY) references
NATION(N_NATIONKEY);
COMMIT WORK;
-- For table LINEITEM
ALTER TABLE LINEITEM
ADD PRIMARY KEY (L_ORDERKEY,L_LINENUMBER);
COMMIT WORK;
-- For table ORDERS
ALTER TABLE ORDERS
ADD PRIMARY KEY (O_ORDERKEY);
COMMIT WORK;
-- For table PARTSUPP
ALTER TABLE PARTSUPP
ADD FOREIGN KEY PARTSUPP_FK1 (PS_SUPPKEY) references SUPPLIER(S_SUPPKEY);
COMMIT WORK;
ALTER TABLE PARTSUPP
ADD FOREIGN KEY PARTSUPP_FK2 (PS_PARTKEY) references PART(P_PARTKEY);
COMMIT WORK;
-- For table ORDERS
ALTER TABLE ORDERS
ADD FOREIGN KEY ORDERS_FK1 (O_CUSTKEY) references CUSTOMER(C_CUSTKEY);
COMMIT WORK;
-- For table LINEITEM
ALTER TABLE LINEITEM
ADD FOREIGN KEY LINEITEM_FK1 (L_ORDERKEY)    references
ORDERS(O_ORDERKEY);
COMMIT WORK;
ALTER TABLE LINEITEM
ADD FOREIGN KEY LINEITEM_FK2 (L_PARTKEY,L_SUPPKEY) references
            PARTSUPP(PS_PARTKEY,PS_SUPPKEY);
COMMIT WORK;
# 创建索引
create index i_s_nationkey on supplier (s_nationkey);
create index i_ps_partkey on partsupp (ps_partkey);
create index i_ps_suppkey on partsupp (ps_suppkey);
create index i_c_nationkey on customer (c_nationkey);
create index i_o_custkey on orders (o_custkey);
create index i_o_orderdate on orders (o_orderdate);
create index i_l_orderkey on lineitem (l_orderkey);
create index i_l_partkey on lineitem (l_partkey);
create index i_l_suppkey on lineitem (l_suppkey);
```

```
create index i_l_partkey_suppkey on lineitem (l_partkey, l_suppkey);
create index i_l_shipdate on lineitem (l_shipdate);
create index i_l_commitdate on lineitem (l_commitdate);
create index i_l_receiptdate on lineitem (l_receiptdate);
create index i_n_regionkey on nation (n_regionkey);
analyze table supplier;
analyze table part;
analyze table partsupp;
analyze table customer;
analyze table orders;
analyze table lineitem;
analyze table nation;
analyze table region;
```

执行测试：

```
# 22 个复杂查询如下压缩包
```

22个查询.zip

```
# 执行下列脚本，以统计每个查询的耗时。
SHELL> vim exeSQL.sh
#!bin/bash
USER='admin'
PORT=3306
HOST=127.0.0.1
DB='tpchtest'
export MYSQL_PWD='P@ssw0rd'
mysqlconnectstring="/mysqldata/mysql/base/5.7.32/bin/mysql -u${USER} -h${HOST}
-P${PORT} -D${DB}"
############################################################
for i in $(seq 1 22)
do
        st_time=`date +%s`
           ${mysqlconnectstring} < Q${i}.sql > tmp.log
        ed_time=`date +%s`
        echo "Q${i} executed time(s):$(( $ed_time - $st_time ))"

        systemctl restart mysqld_3306.service
        # 每次重启数据库以清空内存池
done
```

查询结果耗时如下，单位为秒：

	麒麟 V10	RH7

Q1	224	167
Q2	36	27
Q3	4733	2909
Q4	1067	443
Q5	1578	761
Q6	66	51
Q7	1423	672
Q8	1793	1241
Q9	8255	7326
Q10	937	673
Q11	72	49
Q12	133	78
Q13	147	128
Q14	221	213
Q15	1088	714
Q16	18	20
Q17	232	48
Q18	154	62
Q19	509	88
Q20	1336	466
Q21	264	176
Q22	10	10

根据上述结果，在 OLAP 测试中，RH7 性能明显优于麒麟 V10。

6. 数据库主从延迟测试

利用 sysbench 模拟业务负载，对比在相同参数下不同操作系统的 MySQL 主从延迟是否存在差异。

创建测试库：

```
mysql> create database sysbenchtest;
```

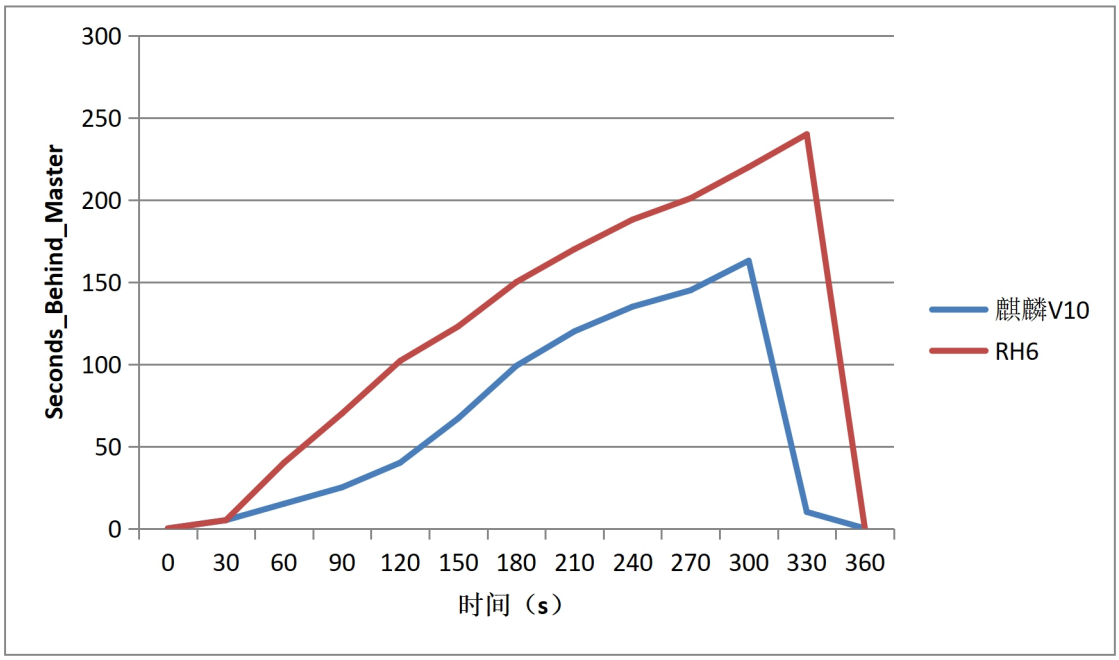
数据准备：

```
SHELL> /mysqldata/sysbench/sysbench-1.0.17/bin/sysbench
/mysqldata/sysbench/sysbench-1.0.17/share/sysbench/oltp_read_write.lua --mysql-host=127.0.0.1
--mysql-port=3306 \
--mysql-user=admin \
--mysql-password='P@ssw0rd' \
--mysql-db=sysbenchtest \
--table-size=1000000 \
--tables=4 \
--threads=8 prepare
```

只写测试：

```
SHELL> /mysqldata/sysbench/sysbench-1.0.17/bin/sysbench
/mysqldata/sysbench/sysbench-1.0.17/share/sysbench/oltp_write_only.lua --mysql-host=127.0.0.1
--mysql-port=3306 \
--mysql-user=admin \
--mysql-password='P@ssw0rd' \
--mysql-db=sysbenchtest \
--table-size=1000000 \
--tables=4 \
--threads=xx \
--time=180 \
--report-interval=10 run
```

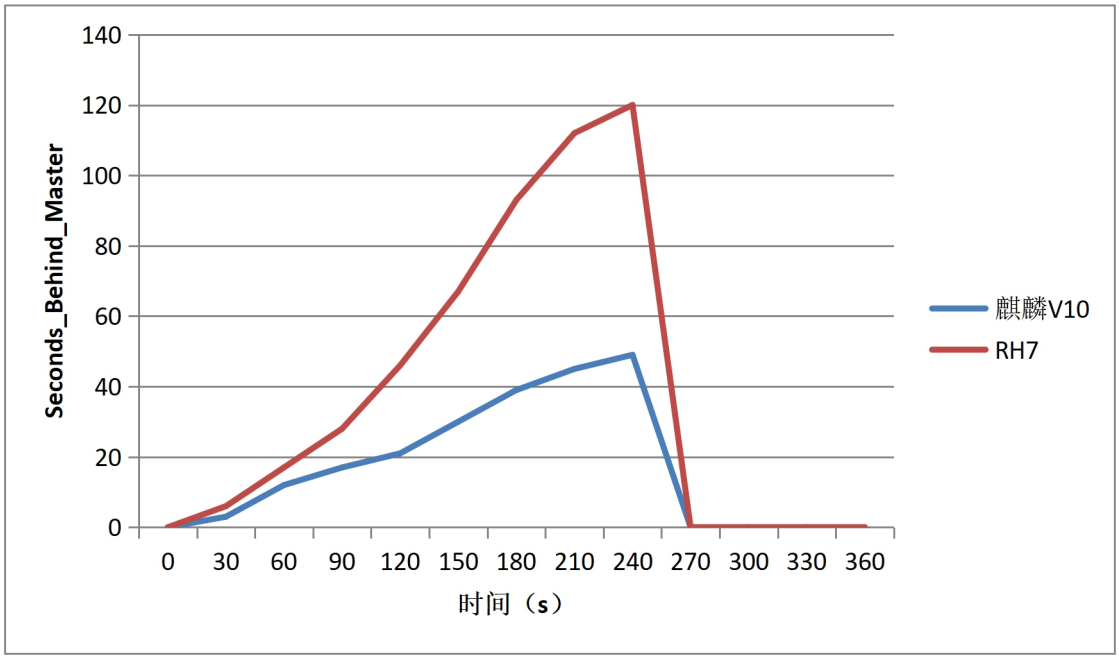
延迟曲线图(64 线程)



麒麟最高延迟：2.65min

RH7 最高延迟：4.00min

延迟曲线图(32 线程)



麒麟最高延迟：49s

RH7 最高延迟：2min

根据上述结果，在主从集群写负载测试中，麒麟 V10 操作系统具备更低的落盘延迟。