# 测试MySQL时区设置对查询效率的影响

## 测试环境

**操作系统：**CentOS 6.4

**MySQL版本：**5.6.21

**表结构：**

|  |
| --- |
| CREATE TABLE `t1` (  `id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT COMMENT '自增列',  `order\_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '业务序号',  `create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',  `type` varchar(64) NOT NULL DEFAULT '',  PRIMARY KEY (`id`),  UNIQUE KEY `uniq\_order\_id` (`order\_id`),  KEY `idx\_create\_time\_type` (`create\_time`,`type`)  ) ENGINE=InnoDB AUTO\_INCREMENT=27001 DEFAULT CHARSET=utf8mb4; |

**数据构造：**

|  |
| --- |
| #!/bin/bash  loop=1  while [ "$loop" -le 3000 ]  do  mysql --login-path=local -D test -e "insert into t1(order\_id, type, create\_time) values (RAND() \* 100000000000, RAND() \* 1000000000, date\_add(now(), interval rand() \* 200 second));"  loop=`expr $loop + 1`  done |

**查询语句：**

|  |
| --- |
| select count(\*) from test.t1 where create\_time>= '2017-07-01 00:00:00' and create\_time < '2017-08-0100:00:00' AND type='A'; |

**lua脚本：**

|  |
| --- |
| pathtest = string.match(test, "(.\*/)") or ""  dofile(pathtest .. "common.lua")  function thread\_init(thread\_id)  set\_vars()  end  function event(thread\_id)  rs = db\_query("select count(\*) from test.t1 where create\_time>= '2017-07-01 00:00:00' and create\_time < '2017-08-0100:00:00' AND type='A';")  end |

测试命令

|  |
| --- |
| sysbench --test=/usr/share/doc/sysbench/tests/db/select.lua --max-time=900 --num-threads=1 --mysql-socket=/var/lib/mysql/mysql.sock --mysql-db=test --mysql-user=us\_jyx --mysql-password='123456' run >> sysbench.log |

## 测试结果

A、在时区为SYSTEM且多线程并发时，存在明显的系统CPU时间，而且CPU的使用达不到4核百分百的使用

B、在单线程时，时区为SYSTEM的QPS明显低于设置时区的查询

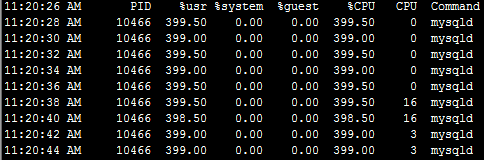
1、time\_zone=’system’,4并发线程

|  |
| --- |
| sysbench 0.5: multi-threaded system evaluation benchmark  Running the test with following options:  Number of threads: 4  Random number generator seed is 0 and will be ignored  Threads started!  OLTP test statistics:  queries performed:  read: 7721  write: 0  other: 0  total: 7721  transactions: 0 (0.00 per sec.)  deadlocks: 0 (0.00 per sec.)  read/write requests: 7721 (8.58 per sec.)  other operations: 0 (0.00 per sec.)  General statistics:  total time: 900.2375s  total number of events: 7721  total time taken by event execution: 3600.6482s  response time:  min: 252.09ms  avg: 466.34ms  max: 606.89ms  approx. 95 percentile: 496.94ms  Threads fairness:  events (avg/stddev): 1930.2500/5.26  execution time (avg/stddev): 900.1621/0.08 |



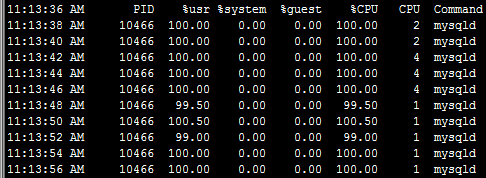
2、time\_zone=’+08:00’,4并发线程

|  |
| --- |
| sysbench 0.5: multi-threaded system evaluation benchmark  Running the test with following options:  Number of threads: 4  Random number generator seed is 0 and will be ignored  Threads started!  OLTP test statistics:  queries performed:  read: 10000  write: 0  other: 0  total: 10000  transactions: 0 (0.00 per sec.)  deadlocks: 0 (0.00 per sec.)  read/write requests: 10000 (277.02 per sec.)  other operations: 0 (0.00 per sec.)  General statistics:  total time: 36.0980s  total number of events: 10000  total time taken by event execution: 144.3321s  response time:  min: 13.95ms  avg: 14.43ms  max: 34.55ms  approx. 95 percentile: 15.04ms  Threads fairness:  events (avg/stddev): 2500.0000/8.57  execution time (avg/stddev): 36.0830/0.00 |



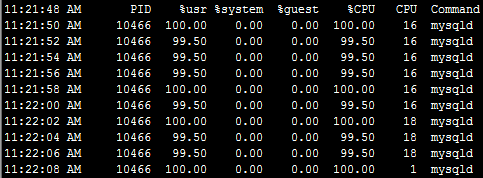
3、time\_zone=’system’,1并发线程

|  |
| --- |
| sysbench 0.5: multi-threaded system evaluation benchmark  Running the test with following options:  Number of threads: 1  Random number generator seed is 0 and will be ignored  Threads started!  OLTP test statistics:  queries performed:  read: 10000  write: 0  other: 0  total: 10000  transactions: 0 (0.00 per sec.)  deadlocks: 0 (0.00 per sec.)  read/write requests: 10000 (22.16 per sec.)  other operations: 0 (0.00 per sec.)  General statistics:  total time: 451.3609s  total number of events: 10000  total time taken by event execution: 451.3416s  response time:  min: 44.71ms  avg: 45.13ms  max: 50.34ms  approx. 95 percentile: 45.67ms  Threads fairness:  events (avg/stddev): 10000.0000/0.00  execution time (avg/stddev): 451.3416/0.00 |



4、time\_zone=’+08:00’,1并发线程

|  |
| --- |
| sysbench 0.5: multi-threaded system evaluation benchmark  Running the test with following options:  Number of threads: 1  Random number generator seed is 0 and will be ignored  Threads started!  OLTP test statistics:  queries performed:  read: 10000  write: 0  other: 0  total: 10000  transactions: 0 (0.00 per sec.)  deadlocks: 0 (0.00 per sec.)  read/write requests: 10000 (74.22 per sec.)  other operations: 0 (0.00 per sec.)  General statistics:  total time: 134.7322s  total number of events: 10000  total time taken by event execution: 134.7129s  response time:  min: 13.30ms  avg: 13.47ms  max: 24.80ms  approx. 95 percentile: 13.84ms  Threads fairness:  events (avg/stddev): 10000.0000/0.00  execution time (avg/stddev): 134.7129/0.00 |



## 对比

QPS对比情况。分别为“SYSTEM”和“+8”时区，以及

## 总结

从测试的结果可以看出来将时区设置为“SYSTEM”会导致MySQL带有timestamp类型比较的查询效率急剧下降。特别是在多线程并发时，很多资源被用于系统开销，MySQL无法充分的使用CPU资源。这将导致在实际生产环境中的触发高CPU使用的风险，降低了MySQL提供服务的可用性。