慢查询日志分析工具pt-query-digest

1. 工具简介

pt-query-digest是用于分析mysql慢查询的一个工具，它可以分析binlog、General log、slowlog，也可以通过SHOWPROCESSLIST或者通过tcpdump**抓取的**MySQL**协议数据来进行分析**。可以把分析结果输出到文件中，分析过程是先对查询语句的条件进行参数化，然后对参数化以后的查询进行分组统计，统计出各查询的执行时间、次数、占比等，可以借助分析结果找出问题进行优化。
pt-query-digest是一个perl脚本，只需下载并赋权即可执行。

```
[root@test1 ]# wget percona.com/get/pt-query-digest
[root@test1 ]# chmod u+x pt-query-digest
```

2.语法及重要选项

```
pt-query-digest [OPTIONS] [FILES] [DSN]
```

--create-review-table 当使用--review参数把分析结果输出到表中时，如果没有表就自动创建。
--create-history-table 当使用--history参数把分析结果输出到表中时，如果没有表就自动创建。
--filter 对输入的慢查询按指定的字符串进行匹配过滤后再进行分析
--limit限制输出结果百分比或数量，默认值是20,即将最慢的20条语句输出，如果是50%则按总响应时间占比从大到小排序，输出到总和达到50%位置截止。
--host mysql服务器地址
--user mysql用户名
--password mysql用户密码
--history 将分析结果保存到表中，分析结果比较详细，下次再使用--history时，如果存在相同的语句，且查询所在的时间区间和历史表中的不同，则会记录到数据表中，可以通过查询同一CHECKSUM来比较某类型查询的历史变化。
--review 将分析结果保存到表中，这个分析只是对查询条件进行参数化，一个类型的查询一条记录，比较简单。当下次使用--review时，如果存在相同的语句分析，就不会记录到数据表中。
--output 分析结果输出类型，值可以是report(标准分析报告)、slowlog(Mysql slow log)、json、json-anon，一般使用report，以便于阅读。
--since 从什么时间开始分析，值为字符串，可以是指定的某个"yyyy-mm-dd [hh:mm:ss]"格式的时间点，也可以是简单的一个时间值：s(秒)、h(小时)、m(分钟)、d(天)，如12h就表示从12小时前开始统计。
--until 截止时间，配合—since可以分析一段时间内的慢查询。

3. 标准分析报告解释

第一部分：总体统计结果,如下图

```
# 320ms user time, 60ms system time, 20.80M rss, 120.95M vsz
# Current date: Wed Apr 16 23:22:56 2014
# Hostname: DBtest1
# Files: slow.log_201404120402
# Overall: 266 total, 55 unique, 0.00 QPS, 0.00x concurrency _____________
# Time range: 2013-12-13 14:09:43 to 2014-04-11 14:48:27
# Attribute          total     min     max     avg     95%  stddev  median
# ============     ======= ======= ======= ======= ======= ======= =======
# Exec time          1661s      2s    140s      6s     15s     13s      3s
# Lock time            60ms       0     5ms   225us   490us   301us   152us
# Rows sent          4.81M       0 834.85k  18.51k  56.74k  82.54k    7.70
# Rows examine     225.02M       0   4.80M 866.26k   2.88M   1.09M 222.42k
# Query size        42.43k      18     297  163.34  284.79   87.49  118.34
```

Overall：总共有多少条查询，上例为总共266个查询。
Time range：查询执行的时间范围。
unique：唯一查询数量，即对查询条件进行参数化以后，总共有多少个不同的查询，该例为55。
total：总计 min:最小 max：最大 avg:平均
95%：把所有值从小到大排列，位置位于95%的那个数，这个数一般最具有参考价值。
median：中位数，把所有值从小到大排列，位置位于中间那个数。

第二部分：查询分组统计结果,如下图

```
# Profile
# Rank Query ID           Response time  Calls R/Call   V/M   Item
# ==== ================== ============== ===== ======== ===== ============
#    1 0x6C99D5A3D50C280C 221.7532 13.3%     3  73.9177  0.15 SELECT ZoneDB.T_USER_INFO_TAB
#    2 0x07716DBE7CA0B5F2 201.0117 12.1%    36   5.5837  0.01 SELECT V_T_ZONECOUNT
#    3 0xCE331DECE88AB069 153.2066  9.2%    59   2.5967  0.11 SELECT T_ZONE_MEMBER_TAB T_GROUP_IN
FO_TAB T_USER_INFO_TAB
#    4 0xBFA7B3CF8B6B8CCB 149.5162  9.0%    56   2.6699  0.12 SELECT T_ZONE_MEMBER_TAB T_GROUP_IN
FO_TAB T_USER_INFO_TAB
#    5 0xB13C834B8870FDC0 139.9880  8.4%     1 139.9880  0.00 SELECT information_schema.INNODB_BU
FFER_PAGE
#    6 0x01BE1650C1FC5105 103.8187  6.2%    42   2.4719  0.00 DELETE ConfigDB.CounterData
#    7 0x523891E6A708BDF5  93.8042  5.6%     2  46.9021 77.12 SELECT V_T_ZONECOUNT
#    8 0xFCD1D329940955EA  62.2734  3.7%     2  31.1367 33.73 SELECT t_zone_id_offer_list
#    9 0x7B1950F49C8994CE  47.7538  2.9%     4  11.9385  0.00 INSERT ConfigDB.CounterData
#   10 0x151980763C16B8BF  42.1930  2.5%     1  42.1930  0.00 SELECT T_ZONE_MEMBER_TAB
#   11 0x704B8415D2088C4B  38.7853  2.3%     5   7.7571  0.00 SELECT CounterData
#   12 0xD03E899AAFE271A3  28.3059  1.7%     3   9.4353  0.03 SELECT T_ZONECOUNT
#   13 0x7F2F1AD498279ABD  27.6718  1.7%     1  27.6718  0.00 SELECT ZoneDB.T_ZONE_MEMBER_TAB
```

由上图可见，这部分对查询进行参数化并分组，然后对各类查询的执行情况进行分析，结果按总执行时长，从大到小排序。
Response：总的响应时间。
time：该查询在本次分析中总的时间占比。

calls：执行次数，即本次分析总共有多少条这种类型的查询语句。
R/Call：平均每次执行的响应时间。
Item ：查询对象

第三部分：每一种查询的详细统计结果，如下图：

```
# Query 12: 0.06 QPS, 0.52x concurrency, ID 0xD03E899AAFE271A3 at byte 4069
# This item is included in the report because it matches --limit.
# Scores: V/M = 0.03
# Time range: 2013-12-30 16:21:10 to 16:22:04
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          1       3
# Exec time      1     28s      9s     10s      9s     10s   554ms      9s
# Lock time      0   279us    91us    94us    93us    93us     1us    93us
# Rows sent      8 440.94k 146.98k 146.98k 146.98k 146.98k       0 146.98k
# Rows examine   0   1.15M 391.57k 391.74k 391.62k 380.41k       0 380.41k
# Query size     0     192      64      64      64      64       0      64
# String:
# Databases    ZoneDB
# Hosts
# Users        uws_AudioDB
# Query_time distribution
#   1us
#  10us
# 100us
#   1ms
#  10ms
# 100ms
#    1s  ################################################################
#  10s+  ##############################
# Tables
#    SHOW TABLE STATUS FROM `ZoneDB` LIKE 'T ZONECOUNT'\G
```

http://blog.csdn.net/seteor

由上图可见，12号查询的详细统计结果，最上面的表格列出了执行次数、最大、最小、平均、95%等各项目的统计。
Databases：库名
Users：各个用户执行的次数（占比）
Query_time distribution ：查询时间分布，长短体现区间占比，本例中1s-10s之间查询数量是10s以上的两倍。
Tables：查询中涉及到的表
Explain：示例

4.用法示例
(1)直接分析慢查询文件：

```
pt-query-digest  slow.log > slow_report.log
```

(2)分析最近12小时内的查询：

```
pt-query-digest  --since=12h  slow.log > slow_report2.log
```

(3)分析指定时间范围内的查询：

```
pt-query-digest slow.log --since '2014-04-17 09:30:00' --until '2014-04-17 10:00:00'> > slow_report3.log
```

(4)分析指含有select语句的慢查询

```
pt-query-digest--filter '$event->{fingerprint} =~ m/^select/i' slow.log> slow_report4.log
```

(5) 针对某个用户的慢查询

```
pt-query-digest--filter '($event->{user} || "") =~ m/^root/i' slow.log> slow_report5.log
```

(6) 查询所有所有的全表扫描或full join的慢查询

```
pt-query-digest--filter '(($event->{Full_scan} || "") eq "yes") ||(($event->{Full_join} || "") eq "yes")' slow.log> slow_report6.log
```

(7)把查询保存到query_review表

```
pt-query-digest  --user=root –password=abc123 --review  h=localhost,D=test,t=query_review--create-review-table  slow.log
```

(8)把查询保存到query_history表

```
pt-query-digest  --user=root –password=abc123 --review  h=localhost,D=test,t=query_ history--create-review-table
slow.log_20140401
pt-query-digest  --user=root –password=abc123--review  h=localhost,D=test,t=query_history--create-review-table  slow.log_20140402
```

(9)通过tcpdump抓取mysql的tcp协议数据，然后再分析

```
tcpdump -s 65535 -x -nn -q -tttt -i any -c 1000 port 3306 > mysql.tcp.txt
pt-query-digest --type tcpdump mysql.tcp.txt> slow_report9.log
```

(10)分析binlog

```
mysqlbinlog mysql-bin.000093 > mysql-bin000093.sql
pt-query-digest  --type=binlog  mysql-bin000093.sql > slow_report10.log
```

(11)分析general log

```
pt-query-digest  --type=genlog  localhost.log > slow_report11.log
```