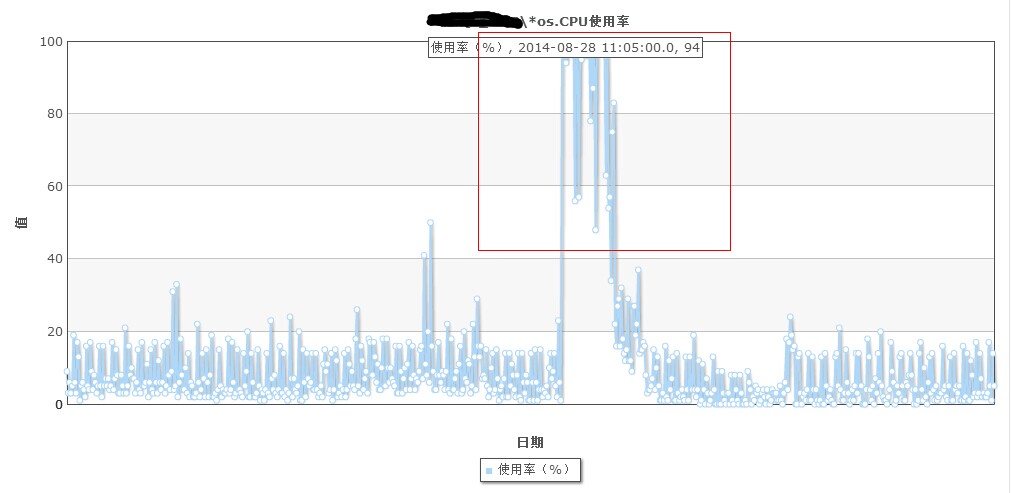
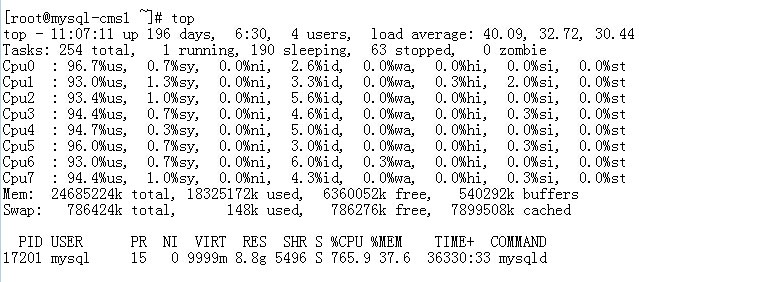
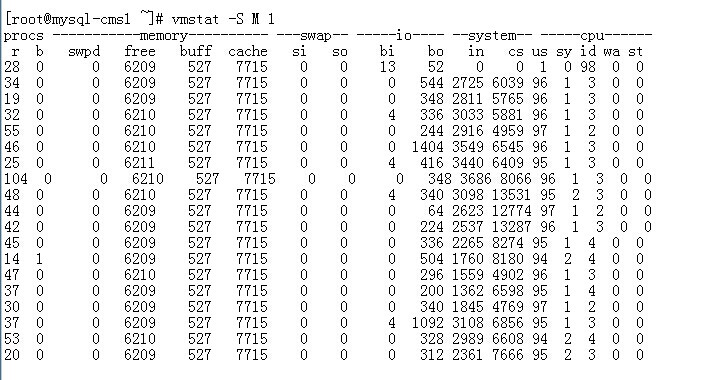
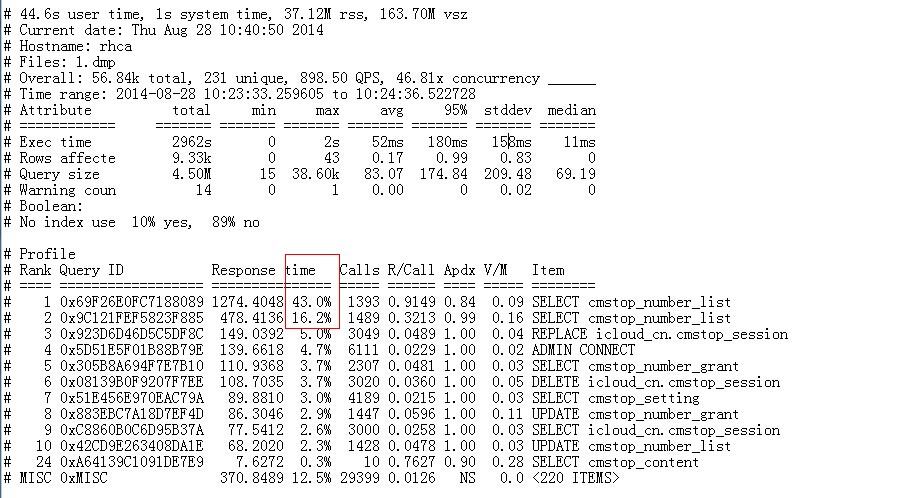
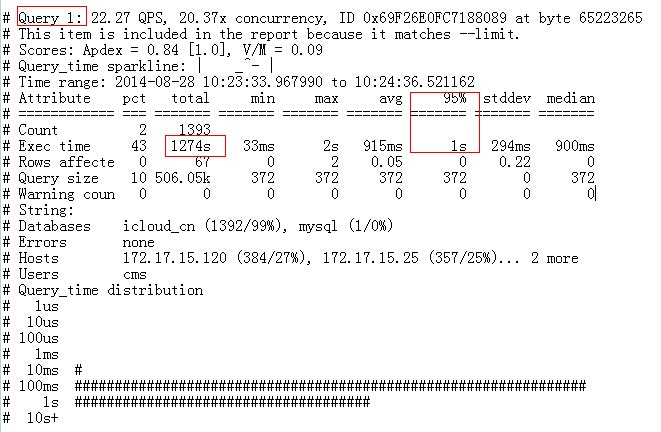
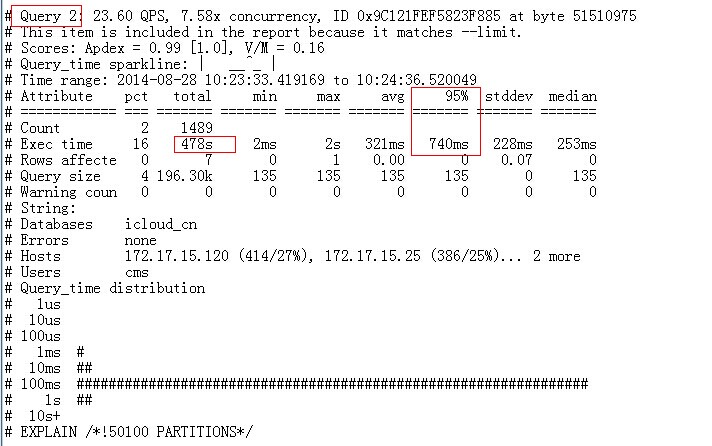
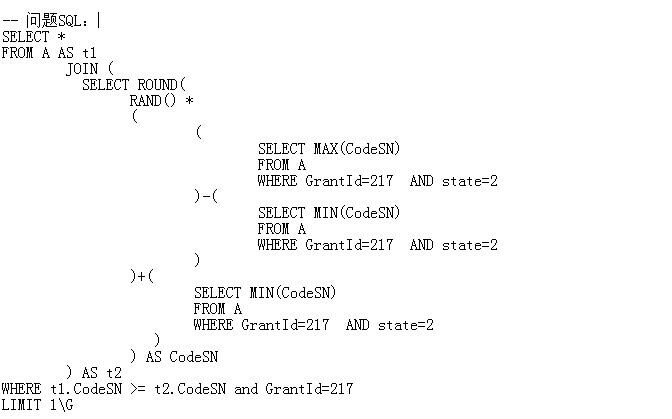
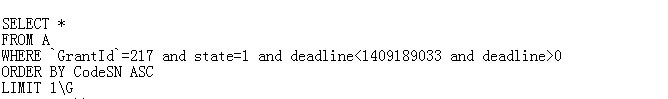
# 2@[MySQL运维] 生产数据库性能优化真实案例

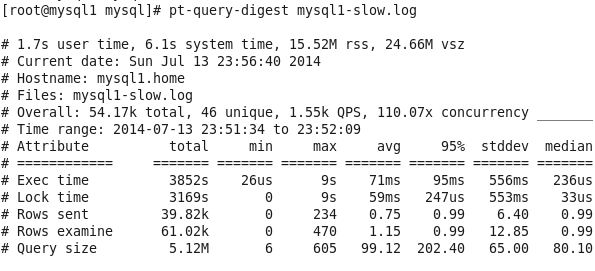
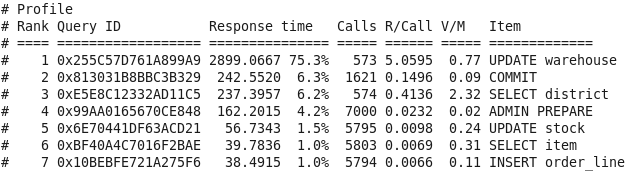
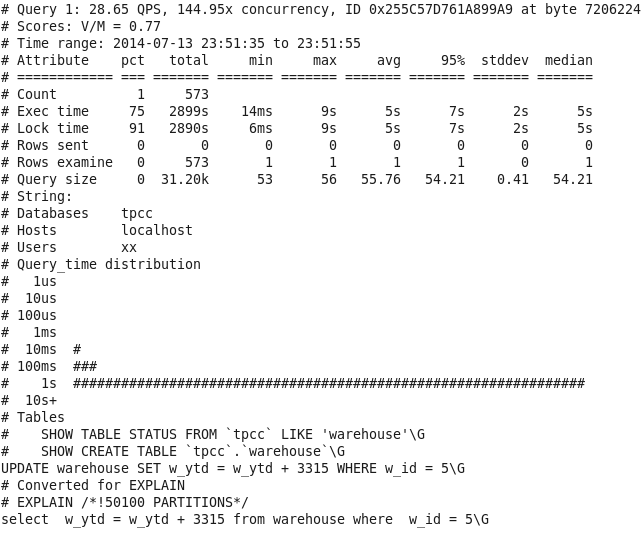
- 炼数成金 - Powered by Discuz!

http://f.dataguru.cn/forum.php?mod=viewthread&action=printable&tid=361155

**作者:**dowa    **时间:**2014-8-29 11:21  
**标题:**[MySQL运维] 生产数据库性能优化真实案例  
*本帖最后由 dowa 于 2015-1-12 01:01 编辑*  
  
案例背景：  
    线上某台mysql生产库服务器，CPU负载告警，使用率持续接近100%，丝毫没有回落的意思！  
  
1. 性能表象的数据采集       
  
       
     登陆DB服务器 ： 、mysql进程的CPU占用率765.9%(8核)，CPU基本上消耗殆尽!  
         
  
  
2. 问题全面诊断：  
         
     结论：  
          内存，磁盘都不是问题，中断请求数还算可以接受，CPU出现性能瓶颈，有大量任务等待CPU时间片；  
          基本上可以断定是单纯的CPU问题，且是由于mysql进程引发；  
          根据CPU负载模型，可以断定出问题的地方：（用户态的CPU使用率是瓶颈）  
                     1. mysql进行大量的内存操作，逻辑读；例如：缺失关键索引    
                     2. 大量的CPU运算任务；例如：2表的笛卡尔积                 
  
  3. MySQL层性能采集  
     此处可以考虑使用慢日志，由于磁盘不是瓶颈，不会带来太大的额外负载；  
     这里我采用tcpdump和pt-query-digest来抓问题SQL；  
  
监控：  
        tcpdump -i eth0 -s 65535 -x -nn -q -tttt port 3306 -c 200000 > 1.dmp &  
  
分析：  
        pt-query-digest --type tcpdump 1.dmp --limit 10 > report.log   
  
            
        基本上搞定TOP2SQL，问题就能得到缓解；  
        这2语句的执行时间基本是在1S的样子，且消耗了大量CPU时间：  
            
            
  
SQL优化：  
       TOP 1 SQL  
            
       同一个表访问了N次，且是select \*，想必大家都知道怎么优化了吧！（吐槽一下！）  
          
       TOP 2 SQL  
            
        该语句基本上是全表扫描，优化的余地不多；select \* 是一个，建一个复合索引，使用索引扫描替换全表扫描  
  
4. 解决办法  
        临时停止相应的业务，将优化方案提给开发部门！

# MySQL性能剖析工具(pt-query-digest)

- Oracle数据库与大数据解决方案-炼数成金-Dataguru专业数据分析社区 http://f.dataguru.cn/thread-748278-1-1.html

这个工具同样来自percona-toolkit  
该工具集合的其他工具  
My[SQL](http://www.dataguru.cn/article-8711-1.html?union_site=innerlink) Slave异常关机的处理 (pt-slave-restart)  
<http://blog.itpub.net/29254281/viewspace-1183545/>  
  
验证MySQL主从一致性(pt-table-checksum&pt-table-sync)  
<http://blog.itpub.net/29254281/viewspace-1208415/>  
  
它可以剖析各种日志,从而分析性能问题  
包括  
1.慢日志(默认)  
2.tcpdump  
3.general log  
4.binlog  
5.show processlist  
  
实验通过tpcc模拟一定的压力,并开启binlog,general log,slow log和tcpdump收集日志。  
关于tpcc参考:<http://blog.itpub.net/29254281/viewspace-1195589/>  
  
1.slow log方式  
将slow\_query\_log设置为on  
long\_query\_time设置为0  
重新连接服务器(一般意味着中间件服务器重启)  
这样慢日志将记录所有的操作。  
  
通过pt-query-digest分析慢日志文件,生成剖析报告  
首先是概要信息,包括一共有54.17k个操作,46个语句,平均每秒1.55k查询,剖析报告覆盖的时间等。  
  
概要还包括类似Top SQL的分析  
可以从响应时间查看对服务器性能影响最大的SQL,其中R/Call指的是  平均每次执行的响应时间 ResponseTime/Calls  
  
然后就是每条SQL的详细剖析报告,  
pct指的是该项占概要报告中的百分比  
  
2.tcpdump方式  
因为慢日志方式需要重新连接,而生产环境重启中间件是非常昂贵的操作。  
所以在数据库服务器上抓包分析也是一种不错的选择。  
pt-query-digest对于抓包有一定的格式。(-x -nn -q -tttt)  
-s:源端口  
-c:抓包的数量  
tcpdump -s 65535 -x -nn -q -tttt -i any -c 100000 port 3306 > test.dmp  
在完成抓包采集之后进行分析  
pt-query-digest --limit 10 --type tcpdump test.dmp  
limit 10 限制显示排名前10的操作  
  
其他方式生成的信息会有缺失,可能达不到预期的效果。  
而processlist方式实验失败,尚不清楚如何使用这种方式剖析性能。

# pt-query-digest查询日志分析工具

- 天宇骑士 - 博客园 http://www.cnblogs.com/wjoyxt/p/5960384.html

1、工具介绍

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。

# wget -O /usr/bin/pt-query-digest  percona.com/get/pt-query-digest

# chmod 755 /usr/bin/pt-query-digest

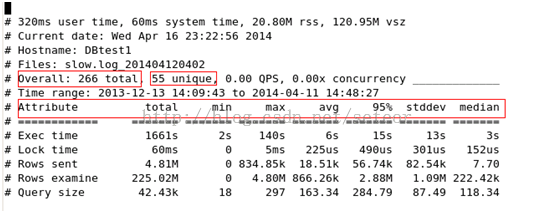
**2、语法及重要选项**

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**

******

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。

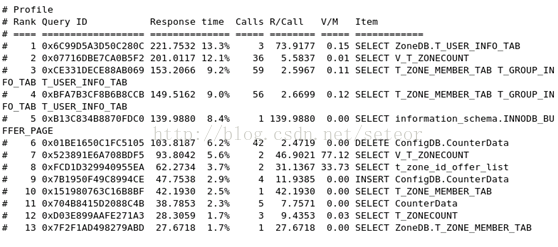
unique: 唯一查询数量，即对查询条件进行参数化以后，总共有多少个不同的查询，该例为55。

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**



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，可以从该响应时间查看对服务器性能影响最大的SQL。

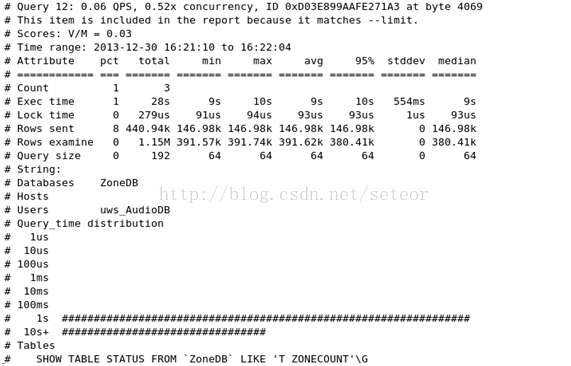
time: 该查询在本次分析中总的时间占比。

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：**



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  
**pct**:指的是该项占概要报告中的百分比

**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\_20140401pt-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**

因为慢日志方式需要重新连接,而生产环境重启中间件是非常昂贵的操作，所以在数据库服务器上抓包分析也是一种不错的选择。

pt-query-digest对于抓包有一定的格式----(-x -nn -q -tttt)

 -s:源端口

 -c:抓包的数量

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

可以使用 --limit 10 限制显示排名前10的操作

**(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

**官方文档：https://www.percona.com/doc/percona-toolkit/2.2/pt-query-digest.html**

**参考资料：http://blog.csdn.net/seteor/article/details/24017913**

# 使用pt-query-digest分析mysql slow query log

- - 博客频道 - CSDN.NET

http://blog.csdn.net/gzh0222/article/details/9273155

high performance [**MySQL**](http://lib.csdn.net/base/mysql)笔记：

孔子说“工欲善其事，必先利其器”， 今天就介绍一下percona toolkit中的pt-query-digest。

下载地址：

http://www.percona.com/software/percona-toolkit/

官方文档：

http://www.percona.com/doc/percona-toolkit/pt-query-digest.html

请先确定在my.ini中打开了mysql的slow\_query\_log，并且保证long\_query\_time参数设置得很合理。

|  |
| --- |
| **[sql]** [view plaincopy](http://blog.csdn.net/lanbingkafei/article/details/8525187)    2. # Log slow queries. Slow queries are queries which take more than the 3. # amount **OF** **TIME** defined IN "long\_query\_time" OR which do NOT USE 4. # indexes well, IF log\_short\_format **IS** NOT enabled. It **IS** normally good idea 5. # **TO** have this turned **ON** IF you frequently **ADD** NEW queries **TO** the 6. # system. 7. slow\_query\_log 9. # ALL queries taking more than this amount **OF** **TIME** (IN seconds) will be 10. # trated **AS** slow. Do NOT USE "1" **AS** a VALUE here, **AS** this will RESULT IN 11. # even very fast queries being logged **FROM** **TIME** **TO** **TIME** (**AS** MySQL 12. # currently measures **TIME** **WITH** **SECOND** accuracy **ONLY**). 13. long\_query\_time = 2 |

pt-query-digest是一个perl脚本，只需下载即可。

[root@sso bianxuehui]# wget percona.com/get/pt-query-digest

[root@sso bianxuehui]# file pt-query-digest

pt-query-digest: a perl script text executable

[root@sso bianxuehui]# ll pt-query-digest

-rw-r--r-- 1 root root 499727 09-02 00:01 pt-query-digest

[root@test\_dx modify]# chmod u+x pt-query-digest

使用如下，如果slow log够大的话，会消耗相当多的cpu和内存，所以最好把slow log和pt-query-digest放到其它的server上面运行。

[root@test\_dx bianxuehui]# ./pt-query-digest slow.log >digest.log

看一下digest.log的内容。  
第一部分是摘要：

|  |
| --- |
| **[sql]** [view plaincopy](http://blog.csdn.net/lanbingkafei/article/details/8525187)     3. # 390ms USER **TIME**, 10ms system **TIME**, 15.67M rss, 105.84M vsz 4. # **CURRENT** **DATE**: Thu **DEC** 29 13:22:42 2011 5. # Hostname: test\_dx 6. # Files: dxdb1-slow.log 7. # Overall: 776 total, 11 **UNIQUE**, 0.00 QPS, 0.00x concurrency \_\_\_\_\_\_\_\_\_\_\_\_\_ 8. # **TIME** range: 2011-09-10 04:03:19 **TO** 2011-12-29 05:02:51 9. # Attribute total **MIN** **MAX** avg 95% stddev median 10. # ============ ======= ======= ======= ======= ======= ======= ======= 11. # **EXEC** **TIME** 5657s 2s 33s 7s 23s 6s 5s 12. # LOCK **TIME** 33s 0 19s 43ms 98us 715ms 38us 13. # **ROWS** sent 323.38k 0 107.36k 426.73 0.99 6.35k 0 14. # **ROWS** examine 323.39k 0 107.36k 426.74 0 6.35k 0 15. # Query **SIZE** 217.95k 38 562 287.61 420.77 81.78 284.79 |

从这份摘要中可以看出，dxdb1-slow.log中共有776条SQL语句， 去掉重复(至于怎样才算是重复，下文会有介绍)，有11条SQL。  
然后又有很多的Attribute和很多的值，比如total(总和)，min最小值, max最大值, avg平均值,  
95%(\*)，stddev（不知道是啥，标准差？）和median中位数。  
中位数的概念就是把这些值从小到大排序，位置位于最中间的那个数。

在这些值中，最有意义的恐怕就是95%了，与中位数类似，它也是把所有值从小到大排列，位置位于95%的那个数。  
它过滤了一些非常大的值，因此更有统计学上的意义，其实统计学上更喜欢用四分位数(1/4, 1/2, 3/4), 中位数也是四分位数之一。

另外，根据SQL语句的不同，可能会有更多的Attribute显示出来。 比如Tmp tables, Tmp disk tbl，Tmp tbl size之类的。

继续看第二部分：

|  |
| --- |
| **[sql]** [view plaincopy](http://blog.csdn.net/lanbingkafei/article/details/8525187)    2. # Profile 3. # Rank Query ID Response **TIME** Calls R/CALL Apdx V/M Item 4. # ==== ================== =============== ===== ====== ==== ===== ======== 5. # 1 0x4BA72E7D091B848A 4002.9799 70.8% 547 7.3181 0.19 5.57 **INSERT** test\_show 6. # 2 0x4AE6E5D741E52A10 463.9715 8.2% 57 8.1399 0.12 3.98 **INSERT** test\_?click 7. # 3 0x120DE9D5CDE2DF8A 455.9982 8.1% 62 7.3548 0.17 5.72 **INSERT** cpm\_pop 8. # 4 0x88B2139A82CBB181 275.9844 4.9% 34 8.1172 0.24 7.41 **INSERT** **UPDATE** test\_stat\_1 9. # 5 0x0F6895DB36DE570A 191.5783 3.4% 20 9.5789 0.23 7.96 **INSERT** **UPDATE** test\_stat\_2 10. # 6 0x30FE5C2032672827 92.1919 1.6% 25 3.6877 0.36 0.59 SHOW **TABLE** STATUS 11. # 7 0x40B69AB7631E4445 84.5306 1.5% 13 6.5024 0.15 3.48 **SELECT** test\_show 12. # MISC 0xMISC 89.7177 1.6% 18 4.9843 NS 0.0 <4 ITEMS> |

这一部分显示了最慢的七种类型的SQL语句。  
我这里最慢的是INSERT INTO test\_show …… 共有547条语句，虽然每次插入的数据都是不同的，但也被归于同一类型的语句了。

第三部分最重要了。  
以排名第七的SQL为例。

|  |
| --- |
| **[sql]** [view plaincopy](http://blog.csdn.net/lanbingkafei/article/details/8525187)   1. # Query 7: 0.00 QPS, 0.00x concurrency, ID 0x40B69AB7631E4445 **at** byte 203205 2. # Scores: Apdex = 0.15 [1.0]\*, V/M = 3.48 3. # Query\_time sparkline: | ^\_| 4. # **TIME** range: 2011-11-24 05:02:15 **TO** 2011-12-10 05:02:43 5. # Attribute pct total **MIN** **MAX** avg 95% stddev median 6. # ============ === ======= ======= ======= ======= ======= ======= ======= 7. # COUNT 1 13 8. # **EXEC** **TIME** 1 85s 2s 20s 7s 10s 5s 5s 9. # LOCK **TIME** 0 1ms 0 118us 80us 108us 29us 86us 10. # **ROWS** sent 0 12 0 1 0.92 0.99 0.26 0.99 11. # **ROWS** examine 0 16 0 3 1.23 2.90 1.09 0.99 12. # Query **SIZE** 0 1.16k 91 91 91 91 0 91 13. # String: 14. # DATABASES test 15. # Hosts dx\_web\_1 16. # Users test\_user 17. # Query\_time distribution 18. # 1us 19. # 10us 20. # 100us 21. # 1ms 22. # 10ms 23. # 100ms 24. # 1s ################################################################ 25. # 10s+ ################### 26. # TABLES 27. # SHOW **TABLE** STATUS **FROM** `test` LIKE 'test\_show'\G 28. # SHOW **CREATE** **TABLE** `test`.`test\_show`\G 29. # EXPLAIN /\*!50100 PARTITIONS\*/ 30. **SELECT** COUNT(\*) **AS** modify\_\_\_c **FROM** test\_show **WHERE** ip=1303055780 AND web\_id=5 AND ad\_id = 4\G |

从上面可以看出，共有13条语句，[95%]Exec time是10s，时间长得比较离谱了。[**数据库**](http://lib.csdn.net/base/mysql)为test，主机为dx\_web\_1，用户名为test\_user,然后是query time的分布图，这个图太恶心了，不过也可以看得出来大部分是处于1-10s之间的， 还有一些超过10秒了。

最后是几条SQL语句， 是pt-query-digest生成的，这些语句有助于分析问题。

事实上，pt-query-digest的功能远远不止于此，除了直接分析slow query log并生成报告，它还可以filter, replay, transform queries for mysql,postgresql, memcached!

参考文档：

http://dev.mysql.com/doc/refman/5.1/en/slow-query-log.html

http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html#sysvar\_long\_query\_time

http://www.percona.com/doc/percona-server/5.1/diagnostics/slow\_extended.html