第七届

数据技术嘉年华

Data Technology Carnival

云·数据·智能 - 数聚价值智胜未来

关注公众号回复help,
可获取更多经典学习
资料和文档，电子书

系统级追踪方法及案例分享

你是否会时常遇到下面的困惑？

某个Oracle 后台进程慢，但我们却不知道他在做什么！

某个程序无法启动或报错，我们却不知道哪里报错！

客户总是抱怨为什么相同的环境，执行结果却大不同?

我们该怎么办？？？

利用 STRACE 分析 LGWR进程

- LGWR进程写 online redo log会不会用到缓存?
- dd if=/dev/zero of=/tmp/test bs=1048576 count=2048会用到缓存吗?

如何真正写磁盘？

```
[root@rac1 ~]# time dd if=/dev/zero of=/tmp/test bs=1048576
count=2048 ;
2048+0 records in
2048+0 records out
2147483648 bytes (2.1 GB) copied, 4.99759 seconds, 430 MB/s

real    0m4.999s
user    0m0.008s
sys     0m2.591s
```

命令结束的时候数据还没有真正写到磁盘上去，对磁盘的写，我们一般是写到了缓存就返回了。

文件系统缓存的作用

```
[root@rac1 ~]# time dd if=/dev/zero of=/tmp/test bs=1048576
count=2048 oflag=sync;
2048+0 records in
2048+0 records out
2147483648 bytes (2.1 GB) copied, 8.09147 seconds, 265 MB/s

real    0m8.094s
user    0m0.000s
sys     0m2.557s
```

利用 STRACE 分析 LGWR进程

```
 -bash-3.2$ ps -ef |grep  $ORACLE_SID|grep lgwr
oracle    4708     1  0 03:40 ?        00:00:23
ora_lgwr_rac1
[root@rac1 ~]# strace -p 4708 2>&1 |grep -i open
```

添加一组redo log 日志组并进行切换操作

```
ALTER DATABASE ADD LOGFILE
('/oracle/app/oracle/product/11.2.0/db_1/dbs/redo02',
'/oracle/app/oracle/product/11.2.0/db_1/dbs/redo03')
SIZE 50m;
alter system switch logfile;
```

利用 STRACE 分析 LGWR进程

```
open("/oracle/app/oracle/product/11.2.0/db_1/dbs/redo02",
O_RDONLY) = 45
open("/oracle/app/oracle/product/11.2.0/db_1/dbs/redo02",
O_RDWR|O_SYNC) = 45
open("/oracle/app/oracle/product/11.2.0/db_1/dbs/redo03",
O_RDONLY) = 45
open("/oracle/app/oracle/product/11.2.0/db_1/dbs/redo03",
O_RDWR|O_SYNC) = 45
```

可以看到LGWR对新的一组redo进行了打开操作，LGWR对online redo log的open操作采用O_SYNC 标志，该标志表示直接写入存储设备。

TRUSS TUSC STRACE是什么？

- Truss
  Truss is an UNIX utility to trace Unix System Calls in Solaris platform. Truss utility is very useful to understand complex problems at OS level. As Truss utility generates enormous amount of data, Oracle Database Instrumentation is always a first step to troubleshoot a problem. If the problem cannot be distilled by Oracle Database instrumentation, then the use of OS tools such as truss, pstack etc are required.

TRUSS,TUSC,STRACE是什么？

- truss，tusc和strace用来 跟踪一个进程的系统调用或信号产生的情况；

- 它们适用于不同的Unix环境：
  - Truss ： AIX，Solaris
  - Tusc ：HP-Unix（需单独安装）
  - Strace：Linux

TRUSS常用参数介绍

-a	显示在每一执行系统调用中传递的参数字符串。
-c	计数跟踪系统调用、故障和信号而不是逐行显示跟踪结果。跟踪命令终止或 truss 中断时生成摘要报告。若还使用 -f 标志，计数包含所有跟踪的系统调用、故障和子进程信号。
-d	每行输出包含时间戳记。时间从跟踪开始以每秒显示。跟踪输出的第一行显示测量单个时间戳记的基本时间。缺省不显示时间戳记。
-D	每行输出显示增量时间。增量时间表示从由该线程引起的最后报告事件起计时引起事件的 LWP 的逝去时间。缺省不显示增量时间。
-e	显示在每一执行系统调用中传递的环境字符串。
-f	跟在 fork 系统调用产生的所有子进程之后，并包含跟踪输出中的信号、故障和系统调用。 通常，仅跟踪第一级命令和进程。如果指定 -f 标志，进程标识与每行跟踪输出一起显示哪个进程执行系统调用或接收信号。
-l	显示有关 LWP 进程的标识（线程标识）及 truss 输出。输出中缺省不显示 LWP 标识。
-o Outfile	指定用于跟踪输出的文件。缺省时输出指向标准错误。
-p	将参数作为一列现存进程的进程标识而不是要执行的命令解释到 truss。 倘若进程用户标识或组标识与用户的用户标识或组标识匹配或者用户是特权用户，truss 控制并开始跟踪每个进程。

TUSC 常用参数介绍

```
Usage: tusc [-<options>] <command [args ...]> -OR- <pid [pid ...]> Most common options:
-f: follow forks
-p: print pid values
-e: show environment variables
-a: show arguments to the exec() call
-o file : send output to file
-C: like -c but also print high/low/average stats
-v: verbose (some system calls only)
-T format: print timestamps in the given format see 'man 3c strftime' for possible
formats
```

Strace 常用参数介绍

-c 统计每一系统调用的所执行的时间,次数和出错的次数等.
-d 输出strace关于标准错误的调试信息.
-f 跟踪由fork调用所产生的子进程.
-ff 如果提供-o filename,则所有进程的跟踪结果输出到相应的filename.pid中,pid是各进程的进程号.
-F 尝试跟踪vfork调用.在-f时,vfork不被跟踪.
-h 输出简要的帮助信息.
-i 输出系统调用的入口指针.
-q 禁止输出关于脱离的消息.
-r 打印出相对时间关于,,每一个系统调用.
-t 在输出中的每一行前加上时间信息.
-tt 在输出中的每一行前加上时间信息,微秒级.
-ttt 微秒级输出,以秒了表示时间.
-T 显示每一调用所耗的时间.
-v 输出所有的系统调用.一些调用关于环境变量,状态,输入输出等调用由于使用频繁,默认不输出.
-V 输出strace的版本信息.

STRACE 使用示例

```
[root@rac1 ~]# strace -tt date
10:48:06.281962 execve("/bin/date", ["date"], [/* 23 vars */]) = 0
10:48:06.282369 brk(0)                  = 0x7e90000
10:48:06.282588 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x2b89cf475000
10:48:06.282990 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x2b89cf476000
10:48:06.283224 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
10:48:06.283582 open("/etc/ld.so.cache", O_RDONLY) = 3
10:48:06.283813 fstat(3, {st_mode=S_IFREG|0644, st_size=109457, ...}) = 0
10:48:06.283991 mmap(NULL, 109457, PROT_READ, MAP_PRIVATE, 3, 0) = 0x2b89cf477000
10:48:06.284192 close(3)                = 0
10:48:06.284310 open("/lib64/librt.so.1", O_RDONLY) = 3
10:48:06.284434 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \"\300\"1\0\0\0"..., 832) = 832
10:48:06.284656 fstat(3, {st_mode=S_IFREG|0755, st_size=53448, ...}) = 0
10:48:06.285596 mmap(0x3122c00000, 2132936, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x3122c00000
10:48:06.285842 mprotect(0x3122c07000, 2097152, PROT_NONE) = 0
10:48:06.286258 mmap(0x3122e07000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x3122e07000
10:48:06.286480 close(3)                = 0
10:48:06.286891 open("/lib64/libc.so.6", O_RDONLY) = 3
10:48:06.287125 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\332\201!1\0\0\0"..., 832) = 832
10:48:06.287343 fstat(3, {st_mode=S_IFREG|0755, st_size=1722304, ...}) = 0
10:48:06.287578 mmap(0x3121800000, 3502424, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x3121800000
10:48:06.288001 mprotect(0x312194e000, 2097152, PROT_NONE) = 0
10:48:06.288395 mmap(0x3121b4e000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x14e000) = 0x3121b4e000
10:48:06.288623 mmap(0x3121b53000, 16728, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x3121b53000
10:48:06.289097 close(3)                = 0
10:48:06.289350 open("/lib64/libpthread.so.0", O_RDONLY) = 3
10:48:06.289590 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240W@\"1\0\0\0"..., 832) = 832
10:48:06.290036 fstat(3, {st_mode=S_IFREG|0755, st_size=145824, ...}) = 0
```

STRACE 使用示例

```
[root@rac1 ~]# strace -c date
Thu Nov  9 10:52:12 CST 2017
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 33.33    0.000013           1        16           mmap
 17.95    0.000007           1         7           mprotect
 10.26    0.000004           1         6           open
 10.26    0.000004           1         7           close
 10.26    0.000004           1         8           fstat
  7.69    0.000003           1         5           read
  2.56    0.000001           0         3           brk
  2.56    0.000001           1         1         1 access
  2.56    0.000001           1         1           execve
  2.56    0.000001           1         1           arch_prctl
  0.00    0.000000           0         1           write
  0.00    0.000000           0         2           stat
  0.00    0.000000           0         1           lseek
  0.00    0.000000           0         3           munmap
  0.00    0.000000           0         2           rt_sigaction
  0.00    0.000000           0         1           rt_sigprocmask
  0.00    0.000000           0         1           getrlimit
  0.00    0.000000           0         1           futex
  0.00    0.000000           0         1           set_tid_address
  0.00    0.000000           0         1           clock_gettime
  0.00    0.000000           0         1           set_robust_list
------ ----------- ----------- --------- --------- ----------------
100.00    0.000039                    70         1 total
```

案例分析1 –数据库SQLPLUS 连接慢

- 背景：
  - ✓操作系统： AIX 7.1
  - ✓数据库： 11.2.0.3 RAC 2 nodes
  - ✓故障现象：Sqlplus 连接慢

- 难点所在：
  - ✓如何检查慢在哪里?
  - ✓重配置环境变量不能解决问题.
  - ✓检查硬件的安装环境没有问题.

案例分析1 –数据库SQLPLUS 连接慢

问题 判断：

- 1. 首先需要区分是本地连接慢? 澄清： Sqlplus / as sysdba 慢，慢约5秒钟左右.
- 2. 远程连接慢不慢?
  - 2.1) 本地使用远程连接慢不慢 sqplus username/passwdd@tns 澄清： 测试也慢，，慢约5秒钟左右.
  - 2.2) 其它主机使用远程慢不慢? 澄清： 同样也慢， 慢约5秒钟左右.

分析:
以上判断表明 这个链接慢 并非一定要和Listener 有关，所以先需要 解决 SQLPLUS / as sysdba 的问题.

案例分析1 –数据库SQLPLUS 连接慢

- 慢在哪里？如何分析？

→ 需要使用Truss 工具.

solaris: truss -aefldDE -o /tmp/truss-lsnr.log -p <PID_of_listener>

AIX: truss -Dlafeo /tmp/truss-lsnr.log -p <PID_of_listener>

HP: tusc aef -o /tmp/tusc-lsnr.log -T "%H:%M:%S" -p <PID_of_listener>.

案例分析1 –数据库SQLPLUS 连接慢

```
23332716: 4168287 23332716: 41682875: 0.0003:   kfcntl(1, F_GETFL, 0x0000000000000008) = 67110914
23332716: 4168287 23332716: 41682875: 0.0003:   access("./login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/usr/bin/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/etc/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/usr/sbin/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/usr/ucb/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/ods/bin/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/usr/bin/X11/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/sbin/login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("./login.sql", 0) Err#2  ENOENT
23332716: 4168287 23332716: 41682875: 0.0002:   access("/ods/bin/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/ods/oraods/product/11.2.0.3/dbhome_1/bin/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/usr/bin/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/bin/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/usr/bin/X11/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/usr/local/bin/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/ods/oraods/common/oracle/bin/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/ods/oraods/common/oracle/sql/login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("./login.sql", 0) Err#2  ENOENT
                  23332716: 41682875: 0.0002:   access("/ods/oraods/product/11.2.0.3/dbhome_1/rdbms/admin/login.sq
```

分析:

可以看出用户环境变量.Profile 存在问题，需要首先修复，然而，待修复此问题发现还是慢.....

案例分析1 –数据库SQLPLUS 连接慢

- 通一步检查Truss 函数栈，分析发现：

```
24249628:  20186551: 0.0003:    sigprocmask(2, 0x0FFFFFFFFFFFC450, 0x0FFFFFFFFFFFC470) = 0
24249628:  20186551: 0.0002:    sigprocmask(2, 0x09001000A033EB70, 0x0FFFFFFFFFFFC2F0) = 0
24249628:  20186551: 0.0002:    _sigaction(10, 0x0000000000000000, 0x0000000110689AB0) = 0
24249628:  20186551: 0.0002:    thread_setmystate(0x0FFFFFFFFFFFBEF0, 0x0000000000000000) = 0
24249628:  20186551: 0.0002:    sigprocmask(2, 0x09001000A033EB70, 0x0FFFFFFFFFFFC2F0) = 0
24249628:  20186551: 0.0002:    _sigaction(10, 0x0FFFFFFFFFFFC490, 0x0000000000000000) = 0
24249628:  20186551: 0.0002:    thread_setmystate(0x0FFFFFFFFFFFBEF0, 0x0000000000000000) = 0
24249628:  20186551: 0.0002:    sigprocmask(2, 0x09001000A033EB70, 0x0FFFFFFFFFFFC2F0) = 0
24249628:  20186551: 0.0002:    _sigaction(11, 0x0000000000000000, 0x0000000110689AE0) = 0
24249628:  20186551: 0.0002:    thread_setmystate(0x0FFFFFFFFFFFBEF0, 0x0000000000000000) = 0
24249628:  20186551: 0.0002:    sigprocmask(2, 0x09001000A033EB70, 0x0FFFFFFFFFFFC2F0) = 0
24249628:  20186551: 0.0002:    _sigaction(11, 0x0FFFFFFFFFFFC490, 0x0000000000000000) = 0
24249628:  20186551: 0.0002:    thread_setmystate(0x0FFFFFFFFFFFBEF0, 0x0000000000000000) = 0
```

这些函数代表什么？？？

分析:

之后对这些函数进行分析，其主要功能如下：

We see in that truss that main reason we are waiting is that the forked thread is looping on some signal handling modification :

They impact the way memory is configured (read/write protected, guard pages...)

案例分析1 –数据库SQLPLUS 连接慢

- 上述发现连接的时候慢 是在于内存页分配有关...

- 进一步检查内存参数发现：
  - PRE_PAGE_SGA 设置为True。
  - 接下来通过测试可以重现此问题，实际验证的确如此：设置为True ，问题重现，设置为False，问题 消失.
  - 并且发现在这个参数为True 的情况下，内存越大可能连接变慢会表现更加地明显.

- 是正常行为吗？澄清：根据Notes 289585.1，该问题表现为正常行为.

“PRE_PAGE_SGA can increase the process startup duration, because every process that starts must access every page in the SGA.”

案例分析1 –数据库SQLPLUS 连接慢

- 问题解决：
  - 通过将pre_page_sga 设置成false（也就是缺省）问题解决
  - 本地连接以及远程连接 的时间都在秒级别以内完成.
- 建议：
  - 在目前版本情况下9i,10g,11g的版本 该参数保持缺省.
  - 此建议同样为Oracle Best Practice 设置.
    - Set **PRE_PAGE_SGA**=false. If set to true, it can significantly increase the time required to establish database connections. In cases where clients might complain that connections to the database are very slow then consider setting this parameter to false, doing so avoids mapping the whole SGA and process startup and thus saves connection time
- 参考文档：
  - RAC and Oracle Clusterware Best Practices and Starter Kit (Platform Independent) (Doc ID 810394.1)
  - Health Check Alert: Consider setting PRE_PAGE_SGA to FALSE (Doc ID 957525.1)

案例分析2 –数据库监听连接慢

- 背景：
  - ✓操作系统： HP-UX 11.31
  - ✓数据库： 9.2.0.8
  - ✓故障现象：业务通过监听连接慢

- 难点所在：
  - ✓如何检查慢在哪里?
  - ✓系统资源不存在瓶颈
  - ✓检查配置文件没有问题

案例分析2 –数据库监听连接慢

问题 判断：

- 1. 通过监听连接数据库的时间在1-5秒之间，但是连接后执行SQL语句不慢。
- 2. ping数据库主机返回非常快，ftp传输文件也相当 快。
- 3.在数据库主机上使用TNSPING以及通过监听连接数据库也非常慢（1-5秒之间），不通过监听的sqlplus “/ as sysdba”速度非常快。
- 4.重启监听后，在较短的时间内，数据库连接较快，但很快又恢复到非常慢的情况。
- 5.检查监听日志listener.log文件，可以发现连接相对比较频繁

分析:

上面的各项诊断，所花的时间并不长，但是得到的结果却能够帮助我们缩小搜索故障来源的范围。上面的诊断结果表明，数据库本身没有问题，网络没有问题。那么问题会在哪里?

案例分析2 –数据库监听连接慢

响应时间=服务时间+等待时间

1秒 = ？ + ？

分析:

响应时间为从客户端发起数据库连接到最终连接上数据库所经过的时间，服务时间为从连接过程中监听和Server Process真正处理连接请求所消耗的时间，而等待时间就是在连接过程中排队等待处理的时间。

案例分析2 –数据库监听连接慢

- 从监听日志来看，平均每秒的连接数为10个左右。
- 通过Tusc工具确认监听连接的处理时间

案例分析2 –数据库监听连接慢

- tusc -o /tmp/495.out -T “%H:%M:%S” -fp 495
- 在客户端连接数据库
- 连接成功后，中断Tusc运行

案例分析2 –数据库监听连接慢

- 在得到的输出文件4595.out中，内容格式如下：
- (Attached to process4595 ("/oracle/app/oracle/product/9.2.0/bin/tnslsnr LISTENER-inherit") [64-bit])
  1244792894.476963 [4595] read(17, 0x9fffffffffff6950, 64) ....[sleeping]
  1244792894.487425 [4595] read(17, "N T P 0 1 9 1 8 5 \n", 64) ..... = 11
  1244792894.488420 [4595] getpid()..... =4595 (1)
  1244792894.488498 [4595] fcntl(17, F_SETFD, 1)..... = 0
  1244792894.488735 [4595] write(16, "\0\0\0= ", 4)..... = 4
  1244792894.489328 [4595] write(16, "(A D D R E S S = (P R O T O C"., 61) ... = 61
  1244792894.489445 [4595] write(16, "\0\00401", 4)..... = 4
  1244792894.476963，系统调用完成时的时间戳，以秒为单位，也就是从1970年1月1日0时0分0秒至当前所经过的秒数。
  [4595]，发起系统调用的进程号。
  Read，系统调用函数名，(17, 0x9fffffffffff6950, 64)，系统调用的参数。
  [sleeping]，系统调用的结果。

案例分析2 –数据库监听连接慢

- 下面我们将根据这些调用信息来详细分析监听的处理过程。
- 监听接受客户端的TCP连接，并获取客户端发过来的TNS数据包。
  - 1244792894.645890 [4595] getsockname(9, 0x9fffffffffff7dc0,0x9fffffffffff7db0) ..... = 0
  - 1244792894.646092 [4595] getpeername(9, 0x9fffffffffff7dc0,0x9fffffffffff7db0) ..... ERR#235 ENOTCONN
  - 1244792894.646355 [4595] accept(9, 0x9fffffffffff7dd0,0x9fffffffffff7db0) ..... = 14
  - 1244792894.646500 [4595] getsockname(14, 0x9fffffffffff7dc0,0x9fffffffffff7db0) ..... = 0
  - 1244792894.646611 [4595] ioctl(14, FIONBIO, 0x9fffffffffff7dfc) .....= 0
  - 1244792894.646770 [4595] setsockopt(14, 0x6, TCP_NODELAY,0x9fffffffffff7e54, 4) ..... = 0
  - 1244792894.646859 [4595] fcntl(14, F_SETFD, 1)..... = 0

监听进程接受连接将使用操作系统调用accept。

案例分析2 –数据库监听连接慢

- 1244792894.769552 [4595] read(17, "N T P 0 1 9 1 9 0 \n", 64)..... = 11
- 1244792894.769849 [4595] getpid()..... = 4595 (1)
- 1244792894.769968 [4595] fcntl(17, F_SETFD, 1)..... = 0
- 1244792894.770531 [4595] write(16, "\0\0\0= ", 4)..... = 4
- 1244792894.770869 [4595] write(16, "(A D D R E S S = (P R O T O C".., 61) ..... = 61
- 1244792894.771035 [4595] write(16, "\0\00401", 4)..... = 4
- 1244792894.785428 [4595] read(17, "\0\0\0\0", 4)..... = 4
- 1244792894.786896 [4595] read(17, "\0\0\0\0", 4) .....= 4
- 1244792894.787165 [4595] close(16)..... = 0
- 1244792894.787289 [4595] close(17)..... = 0
- 1244792894.787590 [4595] close(14) .....= 0
- 1244792894.787981 [4595] lseek(3, 0, SEEK_CUR)..... = 497458800
- 1244792894.788147 [4595] write(3, "1 2 - J U N - 2 0 0 9 1 5 : 4 "., 172) ..... = 172
- 1244792894.788385 [4595] getsockname(9, 0x9fffffffffff7dc0,0x9fffffffffff7db0) ..... = 0
- 1244792894.788548 [4595] getpeername(9, 0x9fffffffffff7dc0,0x9fffffffffff7db0) ..... ERR#235 ENOTCONN
- 1244792894.788756[4595] accept(9, 0x9fffffffffff7dd0, 0x9fffffffffff7db0) ..... = 14

案例分析2 –数据库监听连接慢

- 监听处理是从1244792894.645890开始，到1244792894.788147结束，总共消耗的时间大约是140ms
- 考虑到Tusc工具可能对连接带来部分影响，那么监听的处理时间大约在100ms 左右，这样计算的话，监听一秒能处理的连接数为10个左右。

- 问题总结：
  - 由此可以看到，客户端连接时，的确是由于连接队列太长，导致要等很长时间才能得到监听进程的处理。也就是数据库连接过于频繁，引起请求队列过长，导致了连接等待时间变得很长。

案例分析2 –数据库监听连接慢

- 针对这个监听的问题，解决办法主要有以下两种：
  - 1. 将使用短连接方式的应用，修改为使用连接池方式，这是最根本最彻底的解决方案。
  - 2. 增加更多的监听，以增加监听处理能力。

一个分享交流的地方

微信号：eyygle

Long Press QR Code To
Identify The Concern

长按二维码识别关注

扫一扫，加入我们，分享更多知识

THANKS