

性能运维 -- 借助pstack + strace排查SQL性能问题

原创 金仓数据库 2024-03-05

3406

一、pstack 和 strace

1. pstack

pstack用来跟踪进程栈，这个命令在排查进程问题时非常有用，比如我们发现一个服务一直处于 working 状态（如假死状态，好似死循环），使用这个命令就能轻松定位问题所在。可以在连续一小段时间内（比如：每秒执行一次，连续10次），多执行几次pstack，若发现代码栈总是停在同一个位置，那个位置就需要重点关注，很可能就是出问题的地方。

示例

```
[kingbase@localhost ~]$ pstack 2050
#0  0x00007f95424d1c53 in __select_nocancel () from /lib64/libc.so.6
#1  0x000000000008cc180 in KesMasterMain ()
#2  0x000000000004a726c in main ()
```

解读顺序从下往上，先执行main 函数，然后调用KesMasterMain 然后是 __select _nocancel 函数。如果实际生产中出现了应用程序长时间等待的情况，可以通过pstack 判断应用程序卡在了哪一步。

但是pstack 只能看到程序执行的函数以及对应的内存地址，并不能显示每步的执行时间。

2. strace

strace常用来跟踪进程执行时的系统调用和所接收的信号。

在Linux世界，进程不能直接访问硬件设备，当进程需要访问硬件设备(比如读取磁盘文件，接收网络数据等等)时，必须由用户态模式切换至内核态模式，通过系统调用访问硬件设备。strace可以跟踪到一个进程产生的系统调用，包括参数，返回值，执行消耗的时间。

示例：

```
strace -o output.txt -T -tt -e trace=all -p 2050
参数含义：
-o 将结果输出到文件
-e trace=all 跟踪进程的所有系统调用
-p 进程号
-tt 在每行输出的前面显示时间(精确到毫秒)
-T 显示每次系统调用所花费的时间
```



沐雨听风

关注

67

文章

45

粉丝

263K+

浏览量



获得了 21 次点赞



内容获得 9 次评论



获得了 42 次收藏

TA的专栏



金仓数据库技术专栏

收录 66 篇内容



SQL与DB性能

收录 45 篇内容



数据库运维

收录 11 篇内容

< >

热门文章

知识点滴 -- KingbaseES 函数编译执行

2023-08-08

24719浏览

知识点滴 -- 函数三种稳定态及其对函数调用次数的影响

2023-04-18

22042浏览

SQL优化 -- 利用Rownum条件Count Stop特性优化SQL的一个案例

2023-08-21

9239浏览

SQL优化 -- 针对窗口函数的一个SQL优化案例

2023-04-19

8713浏览

SQL优化 -- 一例 Union All 引发的性能问题

2023-08-09

7446浏览

最新文章

知识点滴 -- CTE Recursive 如何实现 Or


```
[kingbase@localhost ~]$ tail -f output.txt
19:33:39.107104 select(6, [3 4 5], NULL, NULL, {40, 451375}) = 0 (Timeo
19:34:19.590769 rt_sigprocmask(SIG_SETMASK, ~[ILL TRAP ABRT BUS FPE SEG
19:34:19.590982 open("kingbase.pid", O_RDWR) = 10 <0.000043>
19:34:19.591128 read(10, "2050\n/o", 7) = 7 <0.000028>
19:34:19.591225 close(10) = 0 <0.000032>
19:34:19.591318 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0 <0.000050>
19:34:19.591690 select(6, [3 4 5], NULL, NULL, {60, 0} <detached ...>
```

这里的read(10, "2050\n/o", 7)、close(10) 都是linux内核层执行的指令，<0.000032> 这些内容代表的是执行时间。

二、排查耗时的步骤

1、确认进程号

可以通过系统视图确认sql对应的pid

```
SELECT * FROM sys_stat_activity
```

2、打印进程信息

```
pstack 进程号
```

3、查看strace信息

```
strace -o output.txt -T -tt -e trace=all -p 171264
```

4、查看output.txt 并分析执行时间

由于这里会出现很细linux系统的函数所以需要借助百度等搜索工具确认函数对应的操作含义。

三、慢 SQL 问题排查步骤

1. 测试sql

```
EXPLAIN ANALYZE SELECT * FROM "app_family" af2 WHERE NOT EXISTS (SE
```

```
Hash Anti Join (cost=78.38..395702.04 rows=4999850 width=33) (actual t
Hash Cond: ((af2.family_id)::text = (af.family_id)::text)
-> Seq Scan on app_family af2 (cost=0.00..332500.00 rows=5000000 wi
-> Hash (cost=76.50..76.50 rows=150 width=4) (actual time=2.136..2.
Buckets: 16384 (originally 1024) Batches: 1 (originally 1) Me
-> Seq Scan on app_family2 af (cost=0.00..76.50 rows=150 width=4)
Planning Time: 0.466 ms
Execution Time: 14814.652 ms
```

der Siblings by 功能

2025-07-16 17浏览

SQL优化 -- 视图内部函数调用引发的性能问题

2025-04-30 44浏览

SQL优化 -- 如何在一条语句同时返回 Rows and Count

2025-02-19 97浏览

知识点滴 -- old_snapshot_threshold 参数开启导致索引无法使用案例

2025-01-17 158浏览

知识点滴 -- Where子句函数条件执行顺序

2024-12-19 61浏览

目录

• 一、pstack 和 strace

- 1. pstack

- 2. strace

• 二、排查耗时的步骤

• 三、慢 SQL 问题排查步骤

- 1. 测试sql
- 2. 查询对应pid
- 3. pstack 分析 32255 进程
- 4. strace 分析

• 四、总结

2. 查询对应pid

SELECT pid,query FROM sys_stat_activity

27097	
27096	
32251	SHOW search_path
32252	SELECT c.oid,c.*,d.description,pg_catalog.pg_get_viewdef(c.oid FROM pg_catalog.pg_class c LEFT OUTER JOIN pg_catalog.pg_description d ON d.objoid=c.oid AND d.obj WHERE c.relnamespace=\$1 AND c.relkind not in ('i','I','c') AND relname
32255	"EXPLAIN ANALYZE SELECT * FROM "app_family" af2 WHERE NOT "
536	SELECT pid,query FROM sys_stat_activity
27092	

3. pstack 分析 32255 进程

从pstack的分析结果可以看到这个sql的执行过程，但是并不能反馈出慢的步骤。但是如果在sql 执行过程中 通过pstack 多次查看进程，都显示卡在了同一个函数，那就很大可能该函数属于慢的问题点。

[kingbase@localhost ~]\$ pstack 32255
#0 0x0000000000985fbb in hash_search_with_hash_value ()
#1 0x000000000092209a in BufferTableLookup ()
#2 0x00000000009248fc in ReadBufferCommon ()
#3 0x0000000000925383 in ReadBufExtended ()
#4 0x0000000000516382 in HeapGetPage ()
#5 0x0000000000516a0b in HeapGettupPageMode ()
#6 0x0000000000517b3e in HeapGetNextSlot ()
#7 0x00000000006e7761 in SequenceNext ()
#8 0x00000000006e837e in ExecRowScan ()
#9 0x00000000006aaff3 in ExecutorProcNodeInstr ()
#10 0x00000000006d363a in ExecHJoin ()
#11 0x00000000006aaff3 in ExecutorProcNodeInstr ()
#12 0x00000000006a7f38 in StandardExecRun ()
#13 0x00007f95380ea6e5 in KDBExplainExecutorRun () from /opt/Kingbase/E
#14 0x00007f9535bd3d75 in kbss_ExecutorRun () from /opt/Kingbase/ES/V9/
#15 0x000000000068dddde in ExplainOnePlan ()
#16 0x000000000068e09f in ExplainOneQueryPlan ()
#17 0x000000000068e6bd in ExplainQuery ()
#18 0x000000000094e4b7 in standard_ProcessUtility ()
#19 0x00007f9537b67813 in SynonymProcUtility () from /opt/Kingbase/ES/V
#20 0x00007f95378ece31 in PlsqlUtilityCommand () from /opt/Kingbase/ES/
#21 0x00007f95376c33fa in ForceViewProcUtil () from /opt/Kingbase/ES/V9.
#22 0x00007f95374b636c in flashback_ProcessUtility () from /opt/Kingbas
#23 0x00007f9535bd706b in kbss_ProcessUtility () from /opt/Kingbase/ES/
#24 0x000000000094b14c in PortalRunUtility ()
#25 0x000000000094c202 in FillPortalStore ()
#26 0x000000000094ccdd in PortalRun ()
#27 0x00000000009473d9 in ExecSimpleQuery ()
#28 0x0000000000949c7a in BackendMain ()


```
#29 0x000000000004a6683 in ForegroundStartup ()
#30 0x000000000008cc21c in KesMasterMain ()
#31 0x000000000004a726c in main ()
```

4. strace 分析

从strace 分析可以看到从21:36:07 开始到 21:36:22 进程32255一直进行pread64操作，先后涉及文件描述符48、49、50。这时候我们借助百度确认一下pread64的函数的作用

查询后发现pread64 函数是从指定偏移开始读文件。也就是说该sql 从07到22 历时15s左右都在进行文件的读取操作，涉及48、49、50 三个文件。


```
收集指令  
strace -o output.txt -T -tt -e trace=all -p 32255
```

[kingbase@localhost ~]\$ more output.txt

```
21:36:01.883491 epoll_wait(3, [{EPOLLIN, {u32=23034888, u64=23034888}}]  
21:36:07.195191 recvfrom(10, "Q\0\0\0\222EXPLAIN ANALYZE SELECT * "...,  
21:36:07.195553 lseek(50, 0, SEEK_END) = 166756352 <0.000013>  
21:36:07.195627 lseek(52, 0, SEEK_END) = 500768768 <0.000010>  
21:36:07.195662 lseek(54, 0, SEEK_END) = 113106944 <0.000010>  
21:36:07.195693 lseek(45, 0, SEEK_END) = 614400 <0.000010>  
21:36:07.195724 lseek(46, 0, SEEK_END) = 393216 <0.000010>  
21:36:07.195755 lseek(47, 0, SEEK_END) = 385024 <0.000010>  
21:36:07.195971 lseek(50, 0, SEEK_END) = 166756352 <0.000017>  
21:36:07.196088 kill(27091, SIGUSR1) = 0 <0.000024>  
21:36:07.196148 pread64(48, "\v\0\0\0\230\375\200\341\0\0\0\0\0\0\0\0\0\0\  
21:36:07.202769 pread64(48, "\v\0\0\0\270\233\201\341\0\0\0\0\0\0\0\0\0\0\  
21:36:07.202916 pread64(48, "\v\0\0\0\H9\202\341\0\0\0\0\0\0\0\0\0\0\0\  
21:36:07.202961 pread64(48, "\v\0\0\0\210\347\202\341\0\0\0\0\0\0\0\0\0\0\  
21:36:07.203004 pread64(48, "\v\0\0\0\250\205\203\341\0\0\0\0\0\0\0\0\0\0\  
21:36:07.203668 pread64(48, "\v\0\0\0\0008#\204\341\0\0\0\0\0\0\0\0\0\0\0\  
21:36:07.203764 pread64(48, "\v\0\0\0\x\321\204\341\0\0\0\0\0\0\0\0\0\0\0\  
21:36:07.203915 pread64(48, "\v\0\0\0\Hg\205\341\0\0\0\0\0\0\0\0\0\0\0\0\  
21:36:07.203959 pread64(48, "\v\0\0\0(\r\206\341\0\0\0\0\0\0\0\0\0\0\0\0\  
  
. . . . .  
  
21:36:20.227660 pread64(49, "\r\0\0\0\260^T\241\0\0\0\0\0\0\0\0\0\0\0\0\  
21:36:20.227841 pread64(49, "\r\0\0\0\220&U\241\0\0\0\0\0\0\0\0\0\0\0\0\  
  
. . . . .  
  
21:36:21.999330 pread64(50, "\r\0\0\0\240\214\203\331\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999398 pread64(50, "\r\0\0\0\xI\204\331\0\0\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999465 pread64(50, "\r\0\0\0\0\370\204\331\0\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999532 pread64(50, "\r\0\0\0`\245\205\331\0\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999599 pread64(50, "\r\0\0\0\250r\206\331\0\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999680 pread64(50, "\r\0\0\0\0H!\207\331\0\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999780 pread64(50, "\r\0\0\0\220\316\207\331\0\0\0\0\0\0\0\0\0\0\  
21:36:21.999857 pread64(50, "\r\0\0\0\300\233\210\331\0\0\0\0\0\0\0\0\0\0"
```


【版权声明】本文为墨天轮用户原创内容，转载时必须标注文章的来源（墨天轮），文章链接，文章作者等基本信息，否则作者和墨天轮有权追究责任。如果您发现墨天轮中有涉嫌抄袭或者侵权的内容，欢迎发送邮件至：contact@modb.pro进行举报，并提供相关证据，一经查实，墨天轮将立刻删除相关内容。


文章被以下合辑收录



金仓数据库技术专栏（共66篇）

金仓数据库技术专栏，介绍金仓数据库性能优化技术、集群技术，以及SQL优化技术等，致力于打造DBA技术交流、学习空间。

收藏合辑



数据库运维（共11篇）

介绍数据库运维相关技术问题

收藏合辑

评论

分享你的看法，一起交流吧～

相关阅读

知识点滴 -- CTE Recursive 如何实现 Order Siblings by 功能

沐雨听风 17次阅读 2025-07-16 17:08:02