Altibase FAQ | Monitoring & Diagnostics

Thread process debugging method

Overview

When a Process on UNIX over-occupies CPU or Falls into a situation such as a Process Hang, the Process Debugging utility can be used, provided for each UNIX OS, to find the cause by analyzing the currently Running Call Stack.

UNIX Process Debugging methods are different for each utility and slightly different for each UNIX, so the Command is often confused. It explains how to view each thread stack, focusing on simple Commands.

Using pstack

As a Utility provided by SUN and some other UNIX and LINUX, it is a very useful Command to view the Call Stack of the Running Process by Thread. When it is DB Hang, use it.

How to use

- LINUX

  shell> **pstack** processid

- Sun

  shell> **pstack** -F processid

Example

Example

```
$ uname -a
Linux as48-x64 2.6.9-89.ELsmp #1 SMP Mon Apr 20 10:33:05 EDT 2009 x86_64 x86_64 x86_64 GNU/Linux

$ whereis pstack
pstack: /usr/bin/pstack /usr/share/man/man1/pstack.1.gz

$ ps -ef ¦ grep altibase ¦ grep omegaman
altibase 16745 16454  0 13:55 pts/2    00:00:00 grep altibase
altibase 22978     1  0 Dec04 ?        00:00:07 /home/omegaman/altibase_home/bin/altibase -p boot from admin

$ pstack 22978     ## 22978 is a processid of altibase
Thread 41 (Thread 1084229984 (LWP 22979)):
#0  0x00000039070c2f56 in __select_nocancel () from /lib64/tls/libc.so.6
#1  0x0000000000f40f4b in idvClockThread::run ()
#2  0x0000000000ef797c in idtBaseThread::staticRunner ()
#3  0x0000003907706137 in start_thread () from /lib64/tls/libpthread.so.0
```

© Altibase Corp.

```
#4  0x00000039070c9f03 in clone () from /lib64/tls/libc.so.6
Thread 40 (Thread 1094719840 (LWP 22980)):
```

Using dbx

Provided by AIX, HP, and SUN. You can check the Call Stack information of the Running Process through the dbx command, or the stack for each thread can be checked in the core file.

This section describes the method used on AIX as an example.

How to use

1. Check the process id of the process in question.
   shell> ps -eafl | grep altibase
2. Attach to process.
   shell> dbx -a
3. View all thread information
   (dbx) thread
4. View individual thread information only
   (dbx) thread current
   (dbx) where
5. Detach
   (dbx) detach

   > **caution**
   >
   > The connected process must be released by executing the detach command. Otherwise, if exited, it may end up to the target process, so be careful.

6. Exit
   (dbx) quit

Example

```
$ dbx -a 421920
Waiting to attach to process 421920 ...
Successfully attached to altibase.
warning: Directory containing altibase could not be determined.
Apply 'use' command to initialize source path.

Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g

stopped in _event_sleep at 0x9000000004b6a84 ($t6)
0x9000000004b6a84 (_event_sleep+0xe8) e8410028          ld   r2,0x28(r1)
(dbx) where
_event_sleep(??, ??, ??, ??, ??, ??) at 0x9000000004b6a84
_event_wait(??, ??) at 0x9000000004b6f5c
```

© Altibase Corp.

```
_cond_wait_local(??, ??, ??) at 0x9000000004c305c
_cond_wait(??, ??, ??) at 0x9000000004c3628
pthread_cond_timedwait(??, ??, ??) at 0x9000000004c3e38
run()() at 0x100187228
staticRunner(void*)() at 0x10007ea70
(dbx) list
no source file
(dbx) detach
```

Using gdb

If gdb (GNU debugger) is installed, the stack information for each thread can be checked by using gdb.

How to use

1. attach gdb to the target process
   shell> $gdb $ALTIBASE_HOME/bin/altibase processid
2. Information output for each thread
   (gdb) info threads
3. Output stack trace of all threads
   (gdb) thread apply all bt
4. Switch to a specific thread
   (gdb) t 1 ## switch to thread 1
5. Output the current thread's stack
   (gdb) bt ## stack output
6. Quit
   (gdb) quit

Example (altibase PID: 34569)

```
$ gdb $ALTIBASE_HOME/bin/altibase 34567
(gdb) info threads
42 Thread 31 (LWP 4) 0x329614 in _poll ()
41 Thread 30 0x300cf8 in _lwp_sema_wait ()
40 Thread 29 0x300cf8 in _lwp_sema_wait ()
39 Thread 28 (LWP 29) 0x329614 in _poll ()
38 Thread 27 (LWP 28) 0x329614 in _poll ()
37 Thread 26 (LWP 27) 0x329614 in _poll ()
36 Thread 25 (LWP 26) 0x329614 in _poll ()
5 LWP 28 0x329614 in _poll ()
4 LWP 29 0x329614 in _poll ()
3 LWP 30 0x300cf8 in _lwp_sema_wait ()
* 2 Thread 1 (LWP 1) 0x329614 in _poll ()
1 LWP 1 0x329614 in _poll ()

(gdb) thread apply all bt   ##  Output stack trace of all threads
Thread 2 (Thread 1 (LWP 1)):
#0 0x329614 in _poll ()
#1 0x2f2548 in select_large_fdset ()
#2 0x286908 in __1cKidcManagerGselect6FpnGfd_set_pnOPDL_Time_Value__i_ ()
#3 0x978d8 in __1cMmmtThreadMgrIDispatch6M_nGIDE_RC__ ()
```

© Altibase Corp.

```
#4 0x945f4 in __1cGmmiMgrIMainLoop6F_nGIDE_RC__ ()
#5 0x929e0 in main ()
Thread 1 (LWP 1 ):
#0 0x329614 in _poll ()
#1 0x2f2548 in select_large_fdset ()
#2 0x286908 in __1cKidcManagerGselect6FpnGfd_set_pnOPDL_Time_Value__i_ ()
#3 0x978d8 in __1cMmmtThreadMgrIDispatch6M_nGIDE_RC__ ()
---Type to continue, or q to quit---
#4 0x945f4 in __1cGmmiMgrIMainLoop6F_nGIDE_RC__ ()
#5 0x929e0 in main ()
#0 0x329614 in _poll ()

(gdb) t 1   ##  Switch to thread 1

(gdb) bt   ##  Output stack

(gdb) quit
```