pstack, truss, etc

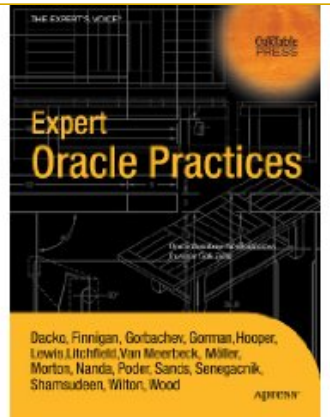
By

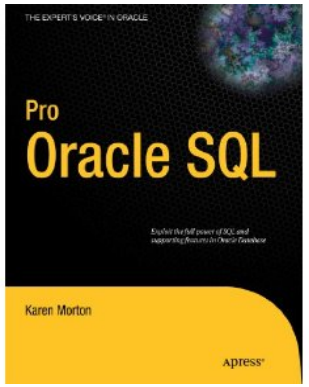
Riyaj Shamsudeen

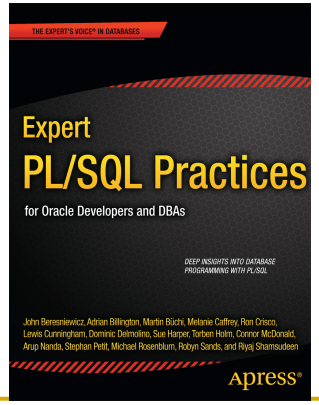
Me

ORACLE ACE Director

OakTable.net







- *19 years using Oracle products/DBA*
- *OakTable member*
- *Oracle ACE Director*
- *Certified DBA versions 7.0,7.3,8,8i,9i &10g*
- *Specializes in RAC, performance tuning, Internals and E-business suite*
- *Chief DBA with OraInternals*
- *Co-author few books*
- *Email: rshamsud@orainternals.com*
- *Blog : orainternals.wordpress.com*
- *Web: www.orainternals.com*

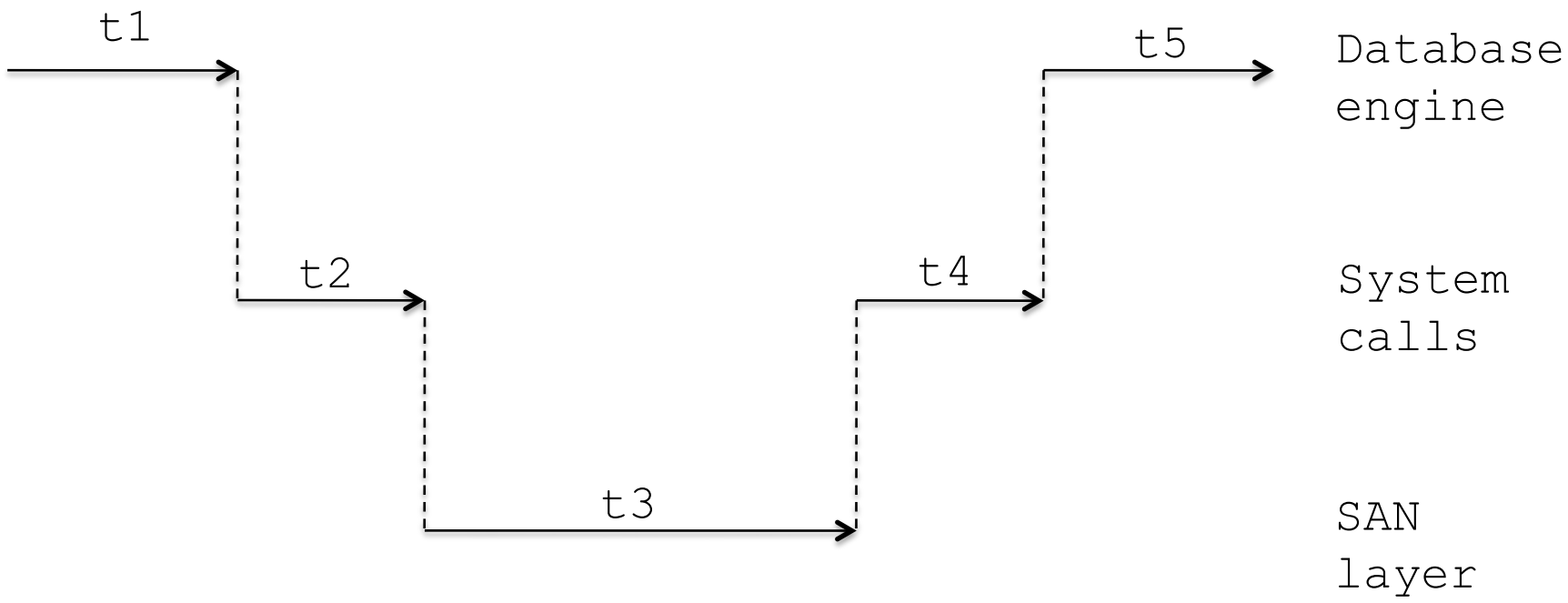
Agenda

- Truss with few examples.
- Demos of truss (hopefully)
- Proc tools intro: pmap, pfiles etc.
- pstack

Truss

- Truss is to trace UNIX system calls and signals.
- Use only if Oracle instrumentation is not sufficient to debug.
- Nearly, all platforms provides tools similar to Truss utility in Solaris.

Read



Db file sequential read = t1 + t2 + t3 + t4 + t5

pread response time = t2 + t3 + t4

I/O response time = t3

Truss

Description:

The truss utility traces the system calls and the signal process receives.

Options:

```
truss [-fcaeildD] [ - [tTvx] [!] syscall ,...] [ - [sS] [!]
signal ,...] [ - [mM] [!] fault ,...] [ - [rw] [!] fd ,...] [ -
[uU] [!] lib ,... : [:] [!] func ,...] [-o outfile] com- mand |
-p pid...
```

```
Solaris – truss
Hpux- tusc
Linux – strace
AIX - truss
```

Truss - trace system calls and signals

```
truss -p 28393
```

…

```
fstat(18, 0xFFBFA100)                                   = 0
lseek(17, 0x0216B000, SEEK_SET)                         = 0x0216B000
write(17, "C8021686 )\0\0 )D0\0\0\0"..., 4096)          = 4096
lseek(17, 0x0219D000, SEEK_SET)                         = 0x0219D000
read(17, "\0\0\0\00101\001FF  \0\0"..., 4096)           = 4096
```

Truss – Few outputs

```
$ truss  -d -E -p  1873

Base time stamp:  1310009834.7781  [ Wed Jul  6 22:37:14 CDT 2011 ]
 0.0124     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.1128     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.1130     0.0000    mmap(0xFFFFFD7FFC1DE000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED, 7, 0) = 0xFFFFFD7FFC1DE000
 0.2132     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.3138     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.4142     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.5146     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.6150     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.7163     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
 0.8181     0.0000    semtimedop(7, 0xFFFFFD7FFFDF9328, 1, 0xFFFFFD7FFFDF9340) Err#11 EAGAIN
```

Time stamp displacement From base timestamp. Seconds.fraction of sec

Elapsed time in the system call.

Semtimedop is the call used by the database processes to sleep Waiting for an event or timer expiry.

Foreground reads

```
...
13.2066  0.0001 pread(256, "06A2\0\098 AC001A9 . 5\0".., 8192, 0x14030000) = 8192
13.3202  0.0001 pread(261, "06A2\0\0C6 ?C001 P . 5\0".., 8192, 0x1418C000) = 8192
13.3215  0.0000 pread(262, "06A2\0\0 3 EC00101 . 5\0".., 8192, 0x14666000) = 8192
13.3250  0.0000 pread(257, "06A2\0\099 >C001C6 * 5\0".., 8192, 0x14532000) = 8192
13.3618  0.0001 pread(257, "06A2\0\0CE FC00101 . 5\0".., 8192, 0x1479C000) = 8192
13.4259  0.0001 pread(263, "06A2\0\0 B BC001CD . 5\0".., 8192, 0x16384000) = 8192

...
```

Foreground process reads 8K blocks from the disk in to buffer cache.

Demo: tp.ksh, dbf_seq_gen.sql

DTrace validation

```
0  => pread                        timestamp : 603738039993
0   | default_physio:phread   oracle sysinfo: timestamp : 603738051273
0   | swtch:pswitch            oracle sysinfo: timestamp : 603738095445
0   | pread:sysread            oracle sysinfo: timestamp : 603738885753
0   | pread:readch             oracle sysinfo: timestamp : 603738888173
0  <= pread                    timestamp : 603738890591 elapsed : 850598
```

DTrace analysis of pread call shows 0.8ms

Linux FG

```
$strace -tttT -o /tmp/s1.lst -p 2395
```

```
1331096471.539412 pread(25, "\6\242\0\0000HA\0w~\v
   \0\0\0\1\6\250e\0\0\1\0\3\0\2\0\0\0v~\v\0"..., 8192,
   113639424) = 8192 <0.000360>
1331096471.539996 pread(25, "\6\242\0\0001HA\0\320~\v
   \0\0\0\1\6TJ\0\0\1\0\2\0\2\0\0\0\317~\v\0"..., 8192,
   113647616) = 8192 <0.000626>
1331096471.540881 pread(25, "\6\242\0\0003HA\0\34\177\v
   \0\0\0\1\6jW\0\0\1\0\0\0\2\0\0\0\33\177\v\0"..., 8192,
   113664000) = 8192 <0.000474>
1331096471.541567 pread(25, "\6\242\0\0004HA\0[\177\v
   \0\0\0\1\6\33/\0\0\1\0\0\0\2\0\0\0Z\177\v\0"..., 8192,
   113672192) = 8192 <0.000436>
```

Foreground process reads 8K blocks from the disk in to buffer cache.

Enqueue waits

```
4.3380  0.5417  0.0000 semtimedop(7, 0xFFFFFD7FFFDF6AD8, 1, 0xFFFFFD7FFFDF6AF0) Err#11 EAGAIN
4.8800  0.5420  0.0000 semtimedop(7, 0xFFFFFD7FFFDF6AD8, 1, 0xFFFFFD7FFFDF6AF0) Err#11 EAGAIN
5.4218  0.5418  0.0000 semtimedop(7, 0xFFFFFD7FFFDF6AD8, 1, 0xFFFFFD7FFFDF6AF0) Err#11 EAGAIN
5.9642  0.5424  0.0000 semtimedop(7, 0xFFFFFD7FFFDF6AD8, 1, 0xFFFFFD7FFFDF6AF0) Err#11 EAGAIN
5.9644  0.0002  0.0000 times(0xFFFFFD7FFFDF7040)                                 = 109615
5.9644  0.0000  0.0000 times(0xFFFFFD7FFFDF7040)                                 = 109615
6.5062  0.5418  0.0000 semtimedop(7, 0xFFFFFD7FFFDF6AD8, 1, 0xFFFFFD7FFFDF6AF0) Err#11 EAGAIN
6.5979  0.0917  0.0000 semtimedop(7, 0xFFFFFD7FFFDF6AD8, 1, 0xFFFFFD7FFFDF6AF0) = 0
```

Lock waiters also sleep on semtimedop call with 0.5ms sleep.

Demo: @enq_wait from two session, ./tp.ksh

Truss

To trace a process, print timestamp offset from the start of TRUSS command and print minimal information

truss -d -p <pid> ***Example: truss -d -p 23898***

To trace a process, send output to a file and print minimal information.

truss -o /tmp/truss.out -p <pid>

Example: truss -o /tmp/truss.out -d -p 23898

To trace a process, follow its children and print minimal information

truss -f -p <pid> ***Example: truss -f -p 23898***

Demo: truss listener (tplistener.ksh), ./sqpo for connection

Connection creation

1432 is the listener process. Double fork creating a connection.

```
1432/1:          4.9712  0.0006 fork1() = 5002
5002:    4.9711  0.0000 fork1() (returning as child ...) = 1432
5002:    4.9763  0.0000 getpid() = 5002 [1432]
…
5002:    4.9791  0.0004 fork1() = 5004
5004:    4.9791  0.0000 fork1() (returning as child ...) = 5002
5004:    4.9820  0.0000 getpid() = 5004 [5002]
…
5004:    4.9860  0.0000 setsid() Err#1 EPERM
5004:    4.9880  0.0019 execve("/u02/app/oracle/product/11.2.0/
   dbhome_2/bin/oracle", 0x004DA7A0, 0x006FDD80)  argc= 2
```

Oracle binary executed to create oracle process.

Truss – Word of caution

At every system call, truss inspects the process. This *potentially* could slow down the process.

So, Truss critical processes, only when it is necessary to do so.

PMON behavior

```
…
0.7998  0.0002 open("/proc/1456/psinfo", O_RDONLY)              = 40
0.7998  0.0000 read(40, "\0\0\00201\0\0\0B005\0\0"., 416)       = 416
0.7999  0.0001 close(40)                                        = 0

0.7999  0.0000 open("/proc/1458/psinfo", O_RDONLY)              = 40
0.7999  0.0000 read(40, "\0\0\00201\0\0\0B205\0\0"., 416)       = 416
0.8000  0.0001 close(40)                                        = 0

0.8000  0.0000 open("/proc/1462/psinfo", O_RDONLY)              = 40
0.8001  0.0001 read(40, "\0\0\00202\0\0\0B605\0\0"., 416)       = 416
0.8001  0.0000 close(40)                                        = 0
…
```

PMON is checking the status of database connection processes. This is how PMON identifies dead process to cleanup.

PMON – detect killed process

```
...
20.8818  0.0000 open("/proc/2372/psinfo", O_RDONLY)     Err#2 ENOENT
20.8826  0.0008 pollsys(0x0EA992B0, 3, 0xFFFFFD7FFFDFAFD0, 0x00000000) = 0
20.8828  0.0002 write(42, "04 Z\0\006\0\0\0\0\0\0\0".., 1114)    = 1114
20.8838  0.0010 write(37, "04 Z\0\006\0\0\0\0\0\0\0".., 1114)    = 1114
20.8843  0.0005 pollsys(0x0EA992B0, 3, 0xFFFFFD7FFFDFB0E0, 0x00000000) = 2
```

Process 2372 was killed and ENOENT thrown. PMON detected the process death and cleaned up the resources.

DBWR truss

```
…
$ truss -d -D -p  1473 |more

Base time stamp:  1327357172.8340  [ Mon Jan 23 16:19:32 CST 2012 ]
…
/1:semtimedop(7, 0xFFFFFD7FFFDFD188, 1, 0xFFFFFD7FFFDFD1A0) (sleeping...)
/1:3.2222  3.0003 semtimedop(7, 0xFFFFFD7FFFDFD188, 1, 0xFFFFFD7FFFDFD1A0) Err#11 EAGAIN
…
/1:semtimedop(7, 0xFFFFFD7FFFDFD188, 1, 0xFFFFFD7FFFDFD1A0) (sleeping...)
/1:6.2228  3.0004 semtimedop(7, 0xFFFFFD7FFFDFD188, 1, 0xFFFFFD7FFFDFD1A0) Err#11 EAGAIN
…
/1:semtimedop(7, 0xFFFFFD7FFFDFD188, 1, 0xFFFFFD7FFFDFD1A0) (sleeping...)
/1:9.2372  3.0143 semtimedop(7, 0xFFFFFD7FFFDFD188, 1, 0xFFFFFD7FFFDFD1A0) Err#11 EAGAIN
…
```

Semtimedop call is used by DBWR to sleep for 3 seconds waiting on semaphore. Wakesup at semaphore activity or timeout.

Demo: tp.ksh on DBWR, alter system checkpoint

DB startup

A Shared memory segment is created for SGA.

```
ps -ef|grep oracle
truss -d -E -f  -o /tmp/dbstart.lst -p 2522
```

```
4.8050  0.0000 shmget(IPC_PRIVATE, 4194304, 0660)              = 58
4.8051  0.0000 shmat(58, 0, IPC_ALLOC)                 = 0xFFFFFD7FFC600000
4.8051  0.0000 shmctl(58, IPC_RMID, 0)                         = 0
4.8054  0.0000 shmget(3433015348, 536879104, 0660|IPC_CREAT|IPC_EXCL) = 59
4.8055  0.0000 shmget(3433015349, 0, 0)                        Err#2 ENOENT
4.8055  0.0000 shmget(3433015350, 0, 0)                        Err#2 ENOENT
4.8055  0.0000 shmget(3433015351, 0, 0)                        Err#2 ENOENT
4.8059  0.0004 shmat(59, 0x60000000, IPC_ALLOC)                = 0x60000000
```

A sqlplus connection process was TRUSSed to get all system calls with –f flag.

Demo: startup db

Shared memory segment

```
0xcc9fa834 = 3433015348
```

```
4.8054  0.0000 shmget(3433015348, 536879104, 0660|IPC_CREAT|IPC_EXCL) = 59
```

```
$ipcs -ma

Shared Memory:

T ID KEY        ... CGROUP    NATTCH SEGSZ

m 59 0xcc9fa834... Oinstall      54 536879104
M  1 0xea42bf0c... oinstall      37 285220864
```

Shared memory segment was created with that shmget call, 59 shm id.

Demo: startup db

PMON startup

2522 is sqlplus connection process, a double fork to pmon pid 2540

```
2522:     6.4786  0.0024 fork1()                              = 2538
2538:     6.4786  0.0000 fork1() (returning as child ...)= 2522
2538:     6.4812  0.0000 getpid()                       = 2538 [2522]
2538:     6.4819  0.0000 lwp_self()                     = 1
..
2538:     6.4823  0.0000 close(21)                      = 0
2538:     6.4824  0.0000 schedctl()                  = 0xFFFFFD7FFDDEE000
2538:     6.4831  0.0006 fork1()                         = 2540
2540:     6.4831  0.0000 fork1() (returning as child ...) = 2538
```

```
$strings /proc/2540/psinfo
Oracle
ora_pmon_solrac2
```

PMON attach

```
2540: 6.5195  0.0000 shmget(3433015348, 0, 0) = 59
2540: 6.5195  0.0000 shmctl(59, IPC_STAT, 0xFFFFFD7FFFDFEA40) = 0
2540: 6.5196  0.0000 shmat(59, 0, 0) = 0xFFFFFD7FDCA00000
2540: 6.5197  0.0000 shmdt(0xFFFFFD7FDCA00000) = 0
2540: 6.5197  0.0000 shmget(3433015349, 0, 0) Err#2 ENOENT
2540: 6.5198  0.0000 shmget(3433015350, 0, 0) Err#2 ENOENT
2540: 6.5198  0.0000 shmget(3433015351, 0, 0) Err#2 ENOENT
2540: 6.5200  0.0002 shmat(59, 0x60000000, IPC_ALLOC) = 0x60000000
```

PMON process attached to the shared memory segment through a shmat call.

Truss

To trace a process and include/exclude specific system calls.

$ truss -d -E -t read -p 1468

```
Base time stamp:  1330727893.5418  [ Fri Mar  2 16:38:13 CST 2012 ]
 1.7195  0.0000 read(38, "\0\0\00201\0\0\0BE05\0\0".., 416)      = 416
 1.7200  0.0000 read(38, "\0\0\00201\0\0\0C005\0\0".., 416)      = 416
 1.7201  0.0000 read(38, "\0\0\00201\0\0\0C405\0\0".., 416)      = 416
```

$ truss -d -E -t !read -p 1468

```
Base time stamp:  1330728046.1275  [ Fri Mar  2 16:40:46 CST 2012 ]
pollsys(0x0EA992B0, 3, 0xFFFFFD7FFFDFB0F0, 0x00000000) (sleeping...)
 2.1865  0.0001 pollsys(0x0EA992B0, 3, 0xFFFFFD7FFFDFB0F0, 0x00000000) = 0
 2.1867  0.0000 times(0xFFFFFD7FFFDFC970)                          = 128012
 2.1868  0.0000 times(0xFFFFFD7FFFDFC970)                          = 128012
```

Demo: @dbf_seq_gen tp_pread.ksh, tp_all.ksh, tp_exc.ksh

Linux

Strace is the equivalent in Linux.
-tt is to print the timestamp with microseconds.
-T is to print the time spent in the system call.

$strace -ttT -p 5164

```
Process 5164 attached - interrupt to quit
..
09:34:38.712285 open("/proc/5166/stat", O_RDONLY) = 28 <0.000154>
09:34:38.712684 read(28, "5166 (oracle) S 1 5166 5166 0 -1"..., 999) = 226
   <0.000182>
09:34:38.713081 close(28)                = 0 <0.000168>
09:34:38.713455 open("/proc/5170/stat", O_RDONLY) = 28 <0.000180>
```

Read of 226 bytes took 1.82 ms at 09:34:38.712684

AIX & HP

- Truss is available in AIX & HP too.
- In HP-UX, truss is called tusc; Usually, there are soft links to tusc named truss.
- man pages on truss or tusc should give all options available.

```
In AIX, -E flag, is not available.
```

truss & pfiles

Truss:

```
write(18, " 9 8 7 1   0   o b j\n <"..., 21)      = 21
fstat(18, 0xFFBFA058)                              = 0
write(18, " 9 8 7 2   0   R > >\n s"..., 18)      = 18
```

Pfiles:

- pfiles can be used to associate this file ids with file names.
- Pfiles lists the files currently opened by a process. In few unix platform, this can be achieved by lsof command also.

Demo: find trace file name

pfiles

Using these device numbers and
Inode numbers, file names can be mapped.

```
pfiles 28393
28393:  ar60runb P_CONC_REQUEST_ID=2452107 STARTDATE='012006'
                 ENDDATE='122006'
  Current rlimit: 4096 file descriptors
   0: S_IFIFO mode:0000 dev:272,0 ino:7325504 uid:11175 gid:100 size:0
...
  17: S_IFREG mode:0644 dev:233,63004 ino:895242 uid:11175 gid:100 size:
   102522880
      O_RDWR|O_CREAT|O_TRUNC
  18: S_IFREG mode:0644 dev:233,63004 ino:895305 uid:11175 gid:100 size:
   25491841
      O_RDWR|O_CREAT|O_TRUNC
```

This is the file_id
In the truss output

This is the device id
Of the form minor,major

Inode number

Pfiles & proc tools

Many tools available, aka proc tools

pflags, pcred, pldd, psig, ***pstack,*** *pfiles, pwdx, pstop, prun, pwait,* ***ptree,*** *ptime*

WARNINGS

The following proc tools stop their target processes while inspecting them and reporting the results: pfiles, pldd, pmap, and pstack.

A process can do nothing while it is stopped. Stopping a heavily used process in a production environment, even for a short amount of time, can cause severe bottlenecks ..

pmap

- Process memory need to be monitored and pmap command can give a breakdown of process memory.
- Useful in Oracle environments to differentiate SGA and PGA of a connection process.
- Some information is cryptic though.

In AIX, this tool is named as procmap
In Linux, & HP-UX, pmap is available

pmap <pid>

Pmap prints a nice memory map of the Process. Various heaps and Stacks are printed here.

```
$ pmap -x 2540 |more
2540:   ora_pmon_solrac2
         Address      Kbytes         RSS       Anon     Locked Mode   Mapped File
0000000000400000     232736       37864          -          - r-x--  oracle
000000000E757000       1424         476        144          - rw---  oracle
000000000E8BB000        156          32         32          - rw---  oracle
000000000E8E2000       1972        1132       1124          - rw---    [ heap ]
0000000060000000     526336      321508          -          - rwxs-    [ dism
    shmid=0x3b ]
FFFFFD7FFCAA0000         64           -          -          - rwx--    [ anon ]
FFFFFD7FFCABE000         72           8          8          - rw---    [ anon ]
FFFFFD7FFCAD0000         64          12         12          - rw---    [ anon ]
FFFFFD7FFCAE0000         64          20         20          - rw---    [ anon ]
..
---------------- ---------- ---------- ---------- ----------
        total Kb     780044      368316       2072          -
```

Total memory mapping size

Demo: testcase1.sql, pmap -x

pmap of pmon

-s flag prints Hardware Address Translation.

`Pmap -xs <pid>`

```
...
          Address     Kbytes        RSS  ...  Pgsz Mode   Mapped File
000000000EACF000         16          -         -    - rw---     [ heap ]
0000000060000000       2048       2048         4K   rwxs-     [ dism shmid=0x7 ]
0000000060200000     378880     174080            - rwxs-     [ dism shmid=0x7 ]
0000000077400000       4096       4096         4K   rwxs-     [ dism shmid=0x7 ]
0000000077800000      12288      12288            - rwxs-     [ dism shmid=0x7 ]
0000000078400000       4096       4096         4K   rwxs-     [ dism shmid=0x7 ]
0000000078800000      12288      12288            - rwxs-     [ dism shmid=0x7 ]
0000000079400000       2048       2048         4K   rwxs-     [ dism shmid=0x7 ]
0000000079600000       2048       2048              rwxs-     [ dism shmid=0x7 ]
0000000079800000      10240      10240         4K   rwxs-     [ dism shmid=0x7 ]
000000007A200000       4096       4096            - rwxs-     [ dism shmid=0x7 ]
000000007A600000       6144       6144         4K   rwxs-     [ dism shmid=0x7 ]
000000007AC00000       2048       2048            - rwxs-     [ dism shmid=0x7 ]
000000007AE00000       2048       2048         4K   rwxs-     [ dism shmid=0x7 ]
…
---------------- ---------- ---------- ---------- ----------
        total Kb     781064     377672
```

Shared memory segment that can grow or shrink.

pmap

A small shell script, to dump Memory map and stack of a process, in a loop, every 10 seconds.

```
#! /bin/ksh
 pid=$1
 (( cnt=1000 ))
 while  [[ $cnt -gt 0 ]];
  do
        date
        pmap -x $pid
        pstack $pid
        echo $cnt
        (( cnt=cnt-1 ))
        sleep 10
 done
```

pmap

```
Address  Kbytes     RSS    Anon  Locked Mode   Mapped File
00010000      72      72       -       - r-x--  java
00030000      16      16      16       - rwx--  java
00034000    8744    8680    8680       - rwx--    [ heap ]
77980000    1224    1048       -       - r--s-  dev:273,2000 ino:104403
77CFA000      24      24      24       - rw--R    [ anon ]
...
FF39A000       8       8       8       - rwx--  libthread.so.1
FF3A0000       8       8       -       - r-x--  libc_psr.so.1
FF3B0000     184     184       -       - r-x--  ld.so.1
FF3EE000       8       8       8       - rwx--  ld.so.1
FF3F0000       8       8       8       - rwx--  ld.so.1
FF3FA000       8       8       8       - rwx--  libdl.so.1
FFB80000      24       -       -       - -----    [ anon ]
FFBF0000      64      64      64       - rw---    [ stack ]
-------- ------- ------- ------- -------
total Kb  182352   65568   26360       -
```

Process initially started with a memory usage of 182MB

pmap

```
Address  Kbytes     RSS    Anon  Locked Mode   Mapped File
00010000      72      72       -       - r-x--  java
00030000      16      16      16       - rwx--  java
00034000    8808    8720    8720       - rwx--    [ heap ]
77980000    1224    1048       -       - r--s-  dev:273,2000 ino:104403
77CFA000      24      24      24       - rw--R    [ anon ]
77F7A000      24      24      24       - rw--R    [ anon ]
78000000      72      72      72       - rwx--    [ anon ]
78012000      64      64      64       - rwx--    [ anon ]
7814C000     144     144     144       - rwx--    [ anon ]
78170000       8       8       8       - rwx--    [ anon ]
78172000       8       8       8       - rwx--    [ anon ]
78174000       8       8       8       - rwx--    [ anon ]
78176000     104     104     104       - rwx--    [ anon ]
..
FF3F0000       8       8       8       - rwx--  ld.so.1
FF3FA000       8       8       8       - rwx--  libdl.so.1
FFB80000      24       -       -       - -----    [ anon ]
FFBF0000      64      64      64       - rw---    [ stack ]
-------- ------- ------- ------- -------
total Kb  281040  210736  171528       -
```

YATC (Yet Another Test Case)

Program is running for many hours. Recently there was a *minor* code change to the program.

Demo: testcase2.sql, pstack_loop.ksh <pid> 20

pstack

```
$pstack 2544
2544:    oraclesolrac1 (DESCRIPTION=(LOCAL=YES)
    (ADDRESS=(PROTOCOL=beq)))
 000000000ab1418f pevm_SUBSTR () + 12f
 000000000aad49bf pfrinstr_SUBSTR () + 5f
 000000000aac5880 pfrrun_no_tool () + 40
 000000000aac6a6f pfrrun () + 4df
 000000000ab2e3fa plsql_run () + 2ea
 000000000aaa4a83 peicnt () + 143
 000000000a0fba56 kkxexe () + 216
 0000000004447b5c7 opiexe () + 2757
 0000000004d54695 kpoal8 () + ce5
 0000000004472693 opiodr () + 433
 0000000008e67f69 ttcpip () + 599
 0000000004444cfc0 opitsk () + 600
 0000000004445bb75 opiino () + 675
 0000000004472693 opiodr () + 433
 0000000004441f4e opidrv () + 32e
 0000000005672197 sou2o () + 57
 0000000000159eac9 opimai_real () + 219
 0000000000568f2de ssthrdmain () + 14e
 0000000000159e89b main () + cb
 0000000000159e67c ???????? ()
```

pstack

In AIX, this tool is named as procstack

```
2544:   oraclesolrac2 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
000000000ab135cc pevm_MOVC_i () + 5c
000000000aac9c9e pfrinstr_MOVC () + 2e
000000000aac5880 pfrrun_no_tool () + 40
000000000aac6a6f pfrrun () + 4df
000000000ab2e3fa plsql_run () + 2ea
000000000aaa4a83 peicnt () + 143
…
2544:   oraclesolrac2 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
00000000091a2a3b kghuwrlength () + 4b
0000000009e348f3 plsm0vc_rsz () + 43
000000000ab141ca pevm_SUBSTR () + 16a
000000000aad49bf pfrinstr_SUBSTR () + 5f
000000000aac5880 pfrrun_no_tool () + 40
000000000aac6a6f pfrrun () + 4df
000000000ab2e3fa plsql_run () + 2ea
…
2544:   oraclesolrac2 (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))
000000000ab26125 pevm_icd_call_common () + c5
000000000aacb5e3 pfrinstr_BCAL () + 53
 000000000aac5880 pfrrun_no_tool () + 40
000000000aac6a6f pfrrun () + 4df
000000000ab2e3fa plsql_run () + 2ea
000000000aaa4a83 peicnt () + 143...
```

Oradebug short_stack

- Oradebug short_stack also can be used to get process stack.

- Example:

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug short_stack
    ksedsts()+1123<-ksdxfstk()+33<-ksdxen_int()+5127<-ksdxen()+14<-
    opiodr()+1075<-ttcpip()+1433<-opitsk()+1536<-opiino()+1653<-opiodr()
    +1075<-opidrv()+814<-sou2o()+87<-opimai_real()+537<-ssthrdmain()
    +334<-main()+203<-_start()+108
SQL> oradebug short_stack
  ksedsts()+1123<-ksdxfstk()+33<-ksdxen_int()+5127<-ksdxen()+14<-
  opiodr()+1075<-ttcpip()+1433<-opitsk()+1536<-opiino()+1653<-opiodr()
  +1075<-opidrv()+814<-sou2o()+87<-opimai_real()+537<-ssthrdmain()
  +334<-main()+203<-_start()+108
SQL>
```

Thank you for attending!

If you like this presentation, you will love my 2-part intensive, online RAC webinar.

http://www.orainternals.com/training/advanced-rac-training

Email: rshamsud@gmail.com

Week	Dates	Time
Week1	TBA	8AM – 2PM PDT
Week 2	TBA	8AM - 2PM PDT