
Tools and techniques for advanced debugging
in Unix/solaris environment

OR

Scientific approach to problem identification
and resolution.

By

Riyaj Shamsudeen

Who am I?

- OakTable member
- 15 years using Oracle products
- Over 14 years as Oracle DBA/Oracle APPS DBA
- Certified DBA versions 7.0,7.3,8,8i &9i
- Specializes in performance tuning, Internals and E-business suite
- Currently consulting for Cingular (new AT&T, No I don't have any iPhone [yet])
- Adjunct faculty – Northlake College
- Email: rshams4 at yahoo.com



Disclaimer

These slides and materials represent the work and opinions of the author and do not constitute official positions of my current or past employer or any other organization. This material has been peer reviewed, but author assume no responsibility whatsoever for the test cases.

If you corrupt your databases by running my scripts, you are solely responsible for that.

This material should not should not be reproduced or used without the authors' written permission.

blah..blah..

Program is stuck ?

- Problem statement:

Program is stuck for 2+ hours, program usually completes 30 minutes or so. Of course, database was suspected.

In database server..

- Identified sid for the connection. Query `v$session_wait` to see the state of the session.

```
select sid, event, state from v$session_wait where sid=133;
```

```
  SID STATE      EVENT
```

```
-----
```

```
  133 WAITING      SQL*Net message from client
```

In Application server..

- Let's look at application server side.
- Quick review of top shows that server is not heavily loaded.

```
last pid: 10020; load averages: 4.27, 3.99, 3.93 12:24:32
2282 processes:2269 sleeping, 1 running, 4 zombie, 8 on cpu
CPU states: 68.5% idle, 12.5% user, 7.3% kernel, 11.6% iowait, 0.0% swap
Memory: 48G real, 13G free, 31G swap in use, 13G swap free
```

PID	USERNAME	THR	PRI	NICE	SIZE	RES	STATE	TIME	CPU	COMMAND
9927	oradevt	1	20	0	4900M	4863M	cpu/521	0:47	3.65%	oracle

Load average=runnable processes + running processes in CPU [1]
1, 5 and 15 minutes interval

In Application server..

Scan rate is a good indicator
Indicating paging or swapping,
i.e., memory issues..

- vmstat shows there is no paging or swapping.

```
kthr      memory          page            disk           faults         cpu
r  b  w   swap  free  re  mf  pi  po  fr  de  (sr) m0 m1 m2 m5   in   sy   cs  us  sy  id
0  5  0 14947688 13983648 1774 4034 368 12 12 0 1 1 1 1 1 2184 3670 2527 21 11 68
0  5  0 13302656 13721904 325 1537 0 0 0 0 0 0 0 0 11 5505 30341 15702 6 9 85
0  2  0 13306048 13725280 47 894 0 0 0 0 0 0 0 0 35 6461 38540 19818 7 9 84
0  2  0 13309112 13727320 0 164 0 0 0 0 0 0 0 0 5360 31445 18498 8 6 86
0  3  0 13308840 13727008 32 56 0 0 0 0 0 0 0 0 5901 26625 20070 7 3 90
```

- Let's look at the process using ps command

```
ps -ef |egrep '28393|UID'
UID      PID  PPID  C   STIME TTY      TIME CMD
appluser 28393   1    Jun 27 pts/28 48:10 ar60runb
```

```
ps -ef |egrep '28393|UID'
UID      PID  PPID  C   STIME TTY      TIME CMD
appluser 28393   1    Jun 27 pts/28 48:13 ar60runb
```

Process is consuming CPU

Truss - trace system calls and signals

```
truss -p 28393
```

```
llseek(18, 0, SEEK_CUR)          = 0x012EF9A4
fstat(18, 0xFFBFA058)            = 0
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
fstat(18, 0xFFBFA0C0)            = 0
write(18, " q\n 0 . 0 0  1 4 3 . 7"..., 5700) = 5700
fstat(18, 0xFFBFA100)            = 0
write(18, " e n d s t r e a m\n e n"..., 17) = 17
fstat(18, 0xFFBFA100)            = 0
write(18, " 9 8 7 2 0  o b j\n 5"..., 23) = 23
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
lseek(17, 0x0219E000, SEEK_SET)   = 0x0219E000
read(17, "D3\007\0\015CC\0\0\0 qB0"..., 4096) = 4096
lseek(17, 0x0216D000, SEEK_SET)   = 0x0216D000
write(17, "\0  \b\0\0\0 qAA18 L o s"..., 4096) = 4096
lseek(17, 0x0219F000, SEEK_SET)   = 0x0219F000
read(17, "\0\0\0\0\0\0\0\0\00101\001"..., 4096) = 4096
write(18, " 9 8 7 0 0  o b j\n <"..., 189) = 189
fstat(18, 0xFFBFA058)            = 0
llseek(18, 0, SEEK_CUR)          = 0x012F10F4
fstat(18, 0xFFBFA058)            = 0
write(18, " 9 8 7 4 0  o b j\n <"..., 21) = 21
fstat(18, 0xFFBFA058)            = 0
write(18, " 9 8 7 5 0  R > >\n s"..., 18) = 18
fstat(18, 0xFFBFA0C0)            = 0
write(18, " q\n 0 . 0 0  1 4 3 . 7"..., 5736) = 5736
fstat(18, 0xFFBFA100)            = 0
write(18, " e n d s t r e a m\n e n"..., 17) = 17
fstat(18, 0xFFBFA100)            = 0
```

Process seemingly calling
Many seek, write and fstat calls

For, seek, fstat, write, read calls etc,
first argument is the file descriptor.

For read, write call second argument
is the buffer itself.

What files are this process is writing
to ? How big are those files ?

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 (download) Linux - strace
--

Truss

To trace a process and print minimal information

`truss -p <pid>` Example: `truss -p 23898`

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

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

To trace a process, print timestamp 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`

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.

Truss – Few outputs

```
truss -d -o /tmp/truss.out -p 484
```

```
cat /tmp/truss.out:
```

```
Baase time stamp: 1188872874.8745 [ Mon Sep 3 22:27:54 EDT 2007 ]
0.5942 semtimedop(3735584, 0xFFFFFFFF7FFDFDCC, 1, 0xFFFFFFFF7FFDFDB8) Err#11 EAGAIN
0.5949 ioctl(8, (('7'<<8)|72), 0xFFFFFFFF7FFF87F8) = 192
0.5950 ioctl(8, (('7'<<8)|63), 0x1038AA738) = 0
0.5958 semtimedop(3735584, 0xFFFFFFFF7FFFC26C, 1, 0xFFFFFFFF7FFFC258) = 0
0.5998 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD838) = 0
0.6025 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD838) = 0
0.6047 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD838) = 0
0.6054 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFDA48) = 0
0.6059 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD9C8) = 0
0.6064 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD858) = 0
0.6076 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD808) = 0
0.6089 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFD8B8) = 0
1.2775 semtimedop(3735584, 0xFFFFFFFF7FFDFDCC, 1, 0xFFFFFFFF7FFDFDB8) = 0
1.2780 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFF7BF8) = 0
1.2782 ioctl(8, (('7'<<8)|72), 0xFFFFFFFF7FFFA4B8) = 160
1.2783 ioctl(8, (('7'<<8)|63), 0x1038AA738) = 0
1.2785 semtimedop(3735584, 0xFFFFFFFF7FFFD3FC, 1, 0xFFFFFFFF7FFFD3E8) = 0
1.2794 semtimedop(3735584, 0xFFFFFFFF7FFFD3FC, 1, 0xFFFFFFFF7FFFD3E8) = 0
1.2795 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFB08) = 0
1.2797 ioctl(10, (('v'<<24)|('x'<<16)|('o'<<8)|28), 0xFFFFFFFF7FFFB08) = 0
```

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

Recap – Back to our problem

- ❑ DB connection is waiting for the client
 - ❑ Servers are not terribly busy
 - ❑ Front-end process is consuming CPU.
 - ❑ Process is writing and reading from few files
-

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.

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  
   O_RDWR
```

```
1: S_IFREG mode:0644 dev:233,63004 ino:895220 uid:11175 gid:100 size:0  
   O_WRONLY|O_APPEND|O_CREAT
```

```
2: S_IFREG mode:0644 dev:233,63004 ino:895220 uid:11175 gid:100 size:0  
   O_WRONLY|O_APPEND|O_CREAT
```

```
...
```

```
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

ls & df

Device numbers from previous
Slide 233, 63004

```
ls -lt /dev/vx/rdisk/*/*
```

```
crw----- 1 root    root    233,63003 Jul  5 14:19 /dev/vx/rdisk/OAA/data
crw----- 1 root    root    233,49001 Feb 28 2007 /dev/vx/rdisk/APPS/gcontrol
crw----- 1 root    root    233,63004 Dec 15 2006 /dev/vx/rdisk/OAA/app1dev
crw----- 1 root    root    233,49000 Dec 15 2006 /dev/vx/rdisk/APPS/apps
crw----- 1 root    root    233,38001 Dec 15 2006 /dev/vx/rdisk/UAT/data
crw----- 1 root    root    233,63001 Dec 15 2006 /dev/vx/rdisk/OAA/arch
crw----- 1 root    root    233,63002 Dec 15 2006 /dev/vx/rdisk/OAA/backup
crw----- 1 root    root    233,63000 Dec 15 2006 /dev/vx/rdisk/OAA/source
crw----- 1 root    root    233,63005 Dec 15 2006 /dev/vx/rdisk/OAA/tfsource
crw----- 1 root    root    233,38000 Dec 15 2006 /dev/vx/rdisk/UAT/product
```

```
$ df -k
```

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/md/dsk/d0	10327372	4340126	5883973	43%	/
/proc	0	0	0	0%	/proc
...					
/dev/vx/dsk/OAA/app1dev	41943040	32830352	9112688	79%	/apps/oa/app1dev
/dev/vx/dsk/UAT/data	62914560	22409368	40199584	36%	/apps/oa/uat

Find command

Solution: Informed the users
That using a wide search condition
Will take longer to execute.

Both Directory name and inum is known,
this can be mapped to a file name using find command

```
find /apps/oa/appldev -inum 895305
```

```
/apps/oa/appldev/dev/devcomn/admin/out/DEV_XXXXXX/o2452107.out
```

Of course, the process is writing to an output file.

```
ls -lt /apps/oa/appldev/dev/devcomn/admin/out/DEV_XXXXXX/o2452107.out  
-rw-r--r-- 1 appldev dba 6556971555 Jun 27 22:17
```

Of course, this process is writing an huge output file of size 6.5GB. It is going to take couple of hours to complete.

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 ..

Database is causing 'out of memory' exception in java code?

- Problem statement #2:

Vendor modified a java program for performance. Single row inserts were modified to use bulk inserts. Program worked fine in development and testing. Crashes with out of memory exception, for few conditions, in production and database was suspected to be an issue.

No access to the vendor code (Yet..)

Database session

- Let's look at sid:

```
select sid, event from v$session_wait where sid=196
```

```
SID  EVENT
```

```
-----
```

```
196  SQL*Net message from client
```

SID is waiting for the client to send more work, essentially.

pmap

- Error message is memory relevant.
 - Process memory need to be monitored and pmap command can give a breakdown of process memory.
-

pmap <pid>

```
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 ]
77F7A000    24    24    24    -    rw--R  [ anon ]
78000000    72    72    72    -    rwx--  [ anon ]
7814C000   144   144   144    -    rwx--  [ anon ]
783E8000    32    32    32    -    rwx--  [ anon ]
78408000     8     8     8    -    rwx--  [ anon ]
78480000   752   464    -    -    r--s-  dev:85,0 ino:13789
7877E000     8     8     8    -    rw--R  [ anon ]
78800000  36864  8192  8192   -    rwx--  [ anon ]
.....
FF25C000    16     8     8    -    rwx--  libcrun.so.1
FF276000     8     8     -    -    rwxs-  [ anon ]
FF280000   688   688    -    -    r-x--  libc.so.1
FF33C000    32    32    32    -    rwx--  libc.so.1
FF350000    16    16    16    -    rw---  [ anon ]
FF360000     8     8     8    -    rwx--  [ anon ]
FF370000    96    96    -    -    r-x--  libthread.so.1
FF398000     8     8     8    -    rwx--  libthread.so.1
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  -
```

Pmap prints a
Nice memory map
of the Process.
Verious heaps and
Stacks are printed here

Total memory foot print
Also printed.

pmap

```
#!/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
```

Wrote this small shell script, to dump Memory map and stack of this Process, in a loop, every 10 seconds.

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

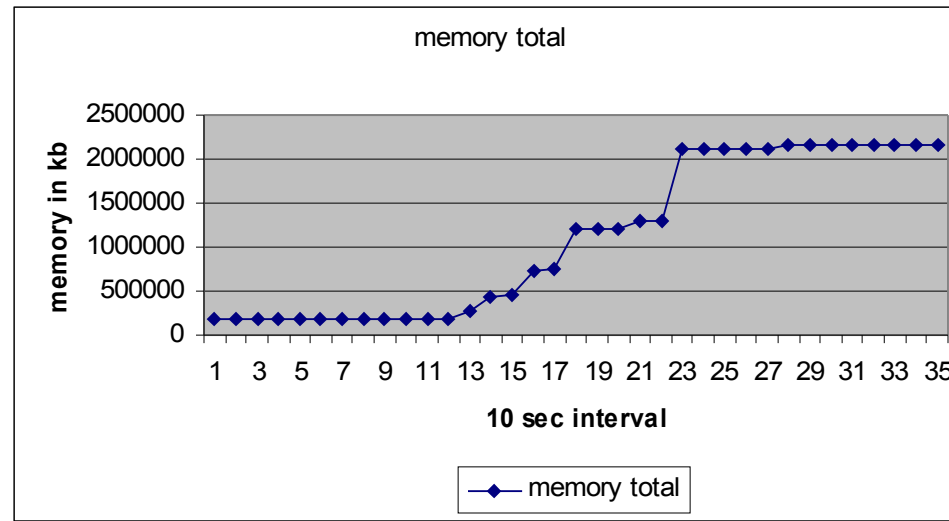
Address	Kbytes	RSS	Anon	Locked	Mode	Mapped File
00010000	72	72	-	-	r-x--	java
00030000	16	16	16	-	rwX--	java
00034000	8808	8720	8720	-	rwX--	[heap]
77980000	1224	1048	-	-	r--s-	dev:273,2000 ino:104403
77CFA000	24	24	24	-	rw--R	[anon]
77F7A000	24	24	24	-	rw--R	[anon]
78000000	72	72	72	-	rwX--	[anon]
78012000	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]
..						
FF370000	96	96	-	-	r-x--	libthread.so.1
FF398000	8	8	8	-	rwX--	libthread.so.1
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	281040	210736	171528	-		

As the process was running, Process memory usage started to grow.

pmap

Graphing the total memory usage column from pmap output, it is visible that this process grabbing more and more memory.

Finally hit 2.1GB and failed shortly after that with 'out of memory' exception.



Recap

- It is established that this is not a database issue.
 - Process memory usage was high.
 - During the whole time, process was not making a single database call.
 - Updated vendor with all our findings and requested code change difference so that we can proceed further.
-

Code fragment

Difference was minimal. It was evident that following code caused it. Code was doing a batch insert, except that batch size was left unlimited !

Of course, if the number of rows to insert was very high, then the size of this array grows huge and resulted in 'out of memory' exception.

[Now, that's why 16GB of memory had to be added to the server, right after the application upgrade !]

```
PreparedStatement stmt = conn.prepareStatement( "insert into test_table values( ?,  
    to_date(?, 'YYYY/MM/DD HH24:MI:SS'),?,?,? )" );  
for( int i=0; i< rec_cnt ; i++ )  
    {  
        stmt.setInt    ( 1, i );  
        stmt.setString( 2, "2004/01/10 23:55:00" );  
        stmt.setDouble( 3, 324.21 );  
        stmt.setLong   ( 4, 324932840 );  
        stmt.setInt    ( 5, 2 );  
        stmt.addBatch();  
    }  
stmt.executeBatch();
```

In conclusion.

- Code change to execute batch every 60,000 rows resolved this issue. This also reduced overall memory usage in the server.
-

A case of high kernel CPU usage

- Problem statement #3:

Server is very busy and run queue size exceeds hundreds. Sudden increase in elapsed time for many key processes and programs.

Server is 32 cpu/128 GB memory server, SGA size not exceeding 24 GB.

vmstat

- vmstat provides a concise view of server health.

```
vmstat 1 5
kthr      memory          page        disk        faults       cpu
r  b  w    swap  free  re  mf  pi  po  fr  de  sr  m0  m1  m4  m5   in   sy   cs  us  sy  id
3  3  0  265724888 296150632 279 2870 279 8 6 0 0 23 4 21 12 9004 66456 38958 9 8 83
2  4  0  260794488 291328496 18 925 0 0 0 0 0 0 0 6 0 9739 74972 49516 11 29 59
0  3  0  260793680 291328248 94 520 0 0 0 0 0 0 0 0 0 9328 83034 47627 10 34 56
323 6 0 260793264 291328176 0 10 0 0 0 0 0 0 0 0 0 10829 61895 50706 11 32 57
251 7 0 260794440 291330168 71 1175 0 0 0 0 0 0 0 0 0 11345 71646 55438 10 23 67
```

Runnable processes
Exceeds 300

Not much swapping
Or paging

30% cpu usage in kernel or
Sys mode. That's 9 CPUs in
Kernel mode usage.

Mpstat 1 5

Many CPUs are in Kernel mode.

This is not an I/O Issue, as wt for I/O is not high.

- mpstat provides per processor level statistics

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
33	9	0	11	80	2	933	38	135	501	0	1614	12	35	17	37
34	12	0	6	66	2	1246	24	123	549	0	1212	7	33	16	44
35	153	0	16	77	2	927	31	117	421	0	1529	11	29	12	48
36	62	0	7	70	2	544	26	110	280	0	1143	13	27	9	51
37	419	0	127	83	2	664	35	103	339	0	1348	24	29	12	36
38	13	0	645	2170	2107	588	39	93	2456	0	440	6	47	4	43
39	267	0	389	58	2	735	21	98	388	0	1038	17	33	18	32
64	133	0	6	66	2	421	32	74	207	0	10888	45	31	0	24
65	185	0	342	65	2	726	29	103	346	0	9341	11	31	6	53
66	5	0	6	46	2	673	12	88	313	0	494	5	31	5	60
67	23	0	344	1087	1016	437	39	90	425	0	1131	9	38	2	51
68	2	0	9190	53	2	421	15	116	282	0	517	7	32	2	60
69	431	0	6	55	2	569	20	107	275	0	1227	29	31	1	39
..															
134	0	0	5	81	1	534	38	96	312	0	3165	14	25	0	61
135	0	0	280	1087	1030	428	25	85	424	0	918	14	39	2	44
160	385	0	291	97	1	1244	34	193	571	0	9384	12	30	0	58
161	0	0	862	55	1	1131	7	138	590	0	683	4	31	1	64
162	6	0	8	61	1	1125	12	117	557	0	1977	6	28	0	66
163	2	0	4	66	1	719	20	112	495	0	1709	6	32	1	62
164	119	0	12	78	1	647	32	106	348	0	732	9	27	2	62
165	5	0	969	74	1	869	29	136	400	0	684	4	32	1	63

sar queue size

Sar confirms observations from Vmstat output.

```
sar -q 1 5
```

15:23:54	runq-sz	%runocc	swpq-sz	%swpocc
15:23:55	267.0	81	0.0	0
15:23:56	154.0	93	0.0	0
15:23:57	0.0	0	0.0	0
15:23:58	1.0	78	0.0	0
15:23:59	0.0	0	0.0	0
Average	140.7	52	0.0	0

Observations

Is that bad? Not necessarily, but needs further review.

- CPU run queue jumps to 100s intermittently.
 - CPUs are used in kernel mode- over 30% kernel mode cpu usage.
 - No observable abnormal swapping or paging activity
 - No observable I/O issues.
-

prstat -mL

prstat provides process level statistics.
Flag m is for microstate accounting

PID	USERNAME	USR	SYS	TRP	TFL	DFL	LCK	SLP	LAT	VCX	ICX	SCL	SIG	PROCESS/LWPID
13893	oracle	84	0.5	0.1	0.0	0.0	0.0	2.1	14	39	180	3K	0	oracle/1
15309	oracle	54	18	0.0	0.0	0.0	0.0	29	0.0	416	271	23K	0	oracle/1
19096	oracle	69	3.5	0.1	0.0	0.0	0.0	14	13	742	178	2K	0	oracle/1
28759	oracle	69	0.4	0.0	0.0	0.0	0.0	18	12	104	119	831	0	oracle/1
15328	oracle	39	27	0.0	0.0	0.0	0.0	34	0.0	615	233	20K	0	oracle/1
15296	oracle	47	19	0.0	0.0	0.0	0.0	34	0.0	617	129	38K	0	oracle/1
15331	oracle	31	34	0.0	0.0	0.0	0.0	38	0.0	344	172	21K	0	oracle/1
27950	oracle	59	3.4	0.0	0.0	0.0	0.0	25	13	636	148	2K	0	oracle/1
15332	oracle	17	42	0.0	0.0	0.0	0.0	42	0.0	13K	376	75K	0	ps/1
20285	oracle	53	4.1	0.1	0.0	0.0	0.0	30	13	957	178	8K	0	oracle/1
11155	oracle	42	11	0.0	0.0	0.0	0.0	34	13	1K	106	16K	0	oracle/1
15299	oracle	28	15	0.0	0.0	0.0	0.0	58	0.0	259	42	13K	0	oracle/1
28250	oracle	21	21	0.0	0.0	0.0	0.0	43	15	5K	178	43K	0	oracle/1
24136	oracle	33	5.1	0.2	0.0	0.0	0.0	48	13	1K	133	4K	0	oracle/1
15292	oracle	26	11	0.0	0.0	0.0	0.0	64	0.0	402	67	14K	0	oracle/1
13306	oracle	30	6.5	0.0	0.0	0.0	0.0	53	11	1K	126	16K	0	oracle/1
27150	oracle	22	7.9	0.0	0.0	0.0	0.0	57	13	2K	172	11K	0	oracle/1
...														
27614	oracle	6.1	4.5	0.0	0.0	0.0	0.0	79	11	891	60	2K	0	oracle/1
28432	oracle	4.2	5.6	0.0	0.0	0.0	0.0	87	3.3	26	12	108	0	oracle/1
28310	oracle	4.5	4.9	0.0	0.0	0.0	0.0	80	11	1K	57	5K	0	oracle/1
28328	oracle	4.3	4.9	0.0	0.0	0.0	0.0	82	9.0	867	74	4K	0	oracle/1
28264	oracle	4.1	4.2	0.0	0.0	0.0	0.0	83	8.9	851	93	4K	0	oracle/1
28248	oracle	4.1	4.1	0.1	0.0	0.0	0.0	86	5.4	63	45	6K	0	oracle/1
28343	oracle	4.0	4.0	0.0	0.0	0.0	0.0	87	4.6	27	46	120	0	oracle/1
28277	oracle	3.9	4.1	0.0	0.0	0.0	0.0	86	6.4	971	40	4K	0	oracle/1
28324	oracle	3.9	3.9	0.0	0.0	0.0	0.0	82	10	860	24	4K	0	oracle/1
28436	oracle	3.4	4.4	0.0	0.0	0.0	0.0	91	1.0	26	25	92	0	oracle/1

LAT column suggests that pid 13893 was waiting for CPU 14% of its time, during that observation Interval.

More prstat -mL

PID	USERNAME	USR	SYS	TRP	TFL	DFL	LCK	SLP	LAT	VCX	ICX	SCL	SIG	PROCESS/LWPID
23131	oraprod	56	4.8	0.7	0.0	0.0	0.0	0.3	39	5	7K	.1M	0	oracle/1
25566	oraprod	57	0.0	0.8	0.0	0.0	0.0	4.3	38	16	7K	13	0	oracle/1
24588	root	19	11	-	-	-	-	0.0	-	1K	5K	9K	0	sendmail/1
18093	oraprod	17	9.8	0.3	0.0	0.0	0.0	43	30	2K	3K	7K	0	oracle/1
23495	oraprod	19	7.8	0.4	0.0	0.0	0.0	40	33	2K	4K	6K	0	oracle/1
23499	oraprod	18	7.8	0.3	0.0	0.0	0.0	45	29	2K	2K	7K	0	oracle/1
23483	oraprod	18	7.0	0.3	0.0	0.0	0.0	39	36	2K	2K	7K	0	oracle/1
23559	oraprod	18	7.5	0.3	0.0	0.0	0.0	39	35	2K	3K	7K	0	oracle/1
23557	oraprod	18	7.2	0.3	0.0	0.0	0.0	35	40	1K	3K	5K	0	oracle/1
23485	oraprod	18	6.9	0.4	0.0	0.0	0.0	40	35	1K	4K	6K	0	oracle/1
23561	oraprod	16	7.3	0.2	0.0	0.0	0.0	50	26	2K	2K	7K	0	oracle/1
23493	oraprod	16	6.9	0.3	0.0	0.0	0.0	38	39	2K	2K	5K	0	oracle/1
21665	oraprod	15	8.3	0.3	0.0	0.0	0.0	39	38	2K	3K	6K	0	oracle/1
796	oraprod	14	8.5	0.2	0.0	0.0	0.0	49	28	2K	2K	7K	0	oracle/1
1345	oraprod	14	8.3	0.2	0.0	0.0	0.0	46	31	2K	2K	7K	0	oracle/1
21667	oraprod	14	8.3	0.2	0.0	0.0	0.0	44	33	2K	2K	7K	0	oracle/1
23491	oraprod	16	6.9	0.4	0.0	0.0	0.0	39	38	1K	3K	5K	0	oracle/1
23481	oraprod	16	6.0	0.4	0.0	0.0	0.0	36	41	1K	4K	5K	0	oracle/1
23553	oraprod	15	6.7	0.3	0.0	0.0	0.0	41	37	2K	2K	6K	0	oracle/1
23563	oraprod	15	5.9	0.3	0.0	0.0	0.0	34	44	1K	2K	5K	0	oracle/1
23489	oraprod	15	5.7	0.4	0.0	0.0	0.0	33	46	1K	3K	5K	0	oracle/1
24824	oraprod	15	5.4	0.2	0.0	0.0	0.0	71	8.0	950	3K	5K	0	oracle/1
807	oraprod	13	7.3	0.3	0.0	0.0	0.0	33	46	1K	3K	5K	0	oracle/1
18097	oraprod	13	7.6	0.3	0.0	0.0	0.0	38	42	2K	3K	5K	0	oracle/1

Truss again

For this process, during this observation period, 11.182 second spent in user mode, 0.344 seconds in kernel mode. What happened to other 4.146 seconds ?

- Truss of that process, with `-c` flag
`truss -c -p 13893`

syscall	seconds	calls	errors
write	.001	20	
times	.326	9360	
semctl	.000	2	
ioctl	.015	48	
	-----	-----	-----
sys totals:	.344	9430	0
usr time:	11.182		
elapsed:	15.670		

Trussing more..

```
truss -c -p 11689
```

syscall	seconds	calls	errors
_exit	.000	1	
read	.015	252	
write	.022	243	
open	.001	4	
close	.000	8	
brk	.000	4	
access	.001	4	
times	.059	1755	
shmdt	.005	1	
semctl	.000	7	
semtimedop	.003	37	
ioctl	.015	193	
poll	.000	2	
sigprocmask	.000	2	
sigaction	.001	22	
getcontext	.000	1	
mmap	.000	1	
munmap	.010	4	
setsockopt	.000	1	
	-----	-----	----
sys totals:	.139	2542	0
usr time:	.254		
elapsed:	1.420		

Trussing another process throws more light in to the problem.

_exit call indicates process destruction

shmdt, semctl calls indicates possible disconnect from the database

Is there a problem with frequent database Connections ?

Trussing listener

Listener spawns new connections using fork system call and 9100 is the pid of the server process

Trussing database listener..

```
sigaction(SIGPIPE, 0xffffffff7fff8b60, 0xffffffff7fff8c68) = 0
poll(0x10040e958, 6, -1) = 1
read(16, "\0E2\0\001\0\0\001 901 ,", 8208) = 226
ioctl(16, FIONBIO, 0xffffffff7fff9e3c) = 0
fcntl(16, F_SETFD, 0x00000000) = 0
pipe() = 17 [18]
pipe() = 19 [20]
fork1() = 9100
waitid(P_PID, 9100, 0xffffffff7fff6d80, WEXITED|WTRAPPED) = 0
close(17) = 0
close(20) = 0
read(19, " N T P 0 9 1 0 1\n", 64) = 10
getpid() = 22186 [1]
```

ps -ef |grep 9100|grep -v egrep returns no lines.

Indeed, pid 9100 shortlived.

Long story short..

- From the listener log file, using a tool by Arup Nanda [3], there was a clear evidence that number of connections to the database has increased.
 - These sessions were disconnecting from the database quickly too.
 - Querying applications table revealed that one of the internal program was submitting hundreds of requests and those requests were completing within seconds.
-

In conclusion.

- That program was disabled for couple of days earlier due to another issue and was enabled on that day.
 - Catch-up work for those couple of days caused increase in increase in request frequency, leading to high kernel mode CPU issue.
 - What did we do ? Nothing. This is not an issue as this program will process pending requests and stabilize itself.
-

Stuck JAVA code?

- Problem statement #3:

Performance of a new java code was not great. It was supposed to process at least 100 items per minute, but processes less than 10 items per minute.

Database

- Problem statement #3:

Performance of a new java code was not great. It was supposed to process at least 100 items per minute, but processes less than 10 items per minute.

In database server..

- Identified sid for the connection. Query `v$sqlsession_wait` to see the state of the session.

```
select sid, event, state from v$sqlsession_wait where sid=133;
```

```
  SID STATE      EVENT
```

```
-----
```

```
  156 WAITING    SQL*Net message from client
```



truss

■ Truss of the process doesn't show much:

```
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/2:    lwp_cond_wait(0x00036538, 0x00036520, 0xFE37FD38) Err#62 ETIME
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/8:    poll(0xFBD7FD80, 0, 50)                = 0
/7:    lwp_cond_wait(0x000369D8, 0x000369C0, 0x00000000) (sleeping...)
↑
      Thread #
```

Memory map looks fine too.

Address	Kbytes	RSS	Anon	Locked	Mode	Mapped File
00010000	72	72	-	-	r-x--	java
00030000	16	16	16	-	rwX--	java
00034000	3784	3704	3704	-	rwX--	[heap]
F1800000	4096	4096	4096	-	rwX--	[anon]
F1F10000	960	440	440	-	rwX--	[anon]
F2000000	4096	-	-	-	rwX--	[anon]
F5800000	4096	4096	4096	-	rwX--	[anon]
F9980000	1208	1032	-	-	r--s-	dev:273,2000 ino:104405
...						
FFB80000	24	-	-	-	-----	[anon]
FFBF2000	56	56	56	-	rw---	[stack]

total kb	64144	55168	17264	-		

pstack

Thread #1 is sleeping.

■ pstack shows current stack of the process. Let's look at pstack for this java process:

```
pstack 16858 |more
```

```
16858: java TestCase
```

```
----- lwp# 1 / thread# 1 -----
```

```
ff31dfdc poll (ffbfefb38, 0, fa0)
```

```
ff37e6a8 select (0, 0, 0, 0, ffbfec08, ffbfec08) + 6c
```

```
fecf0a20 __1cIos_sleep6Fxi_i_ (0, fa0, 1, 0, 4, 0) + 1f8
```

```
fecf07ec __1cCosFsleep6FpnGThread_xi_i_ (36b70, 0, fa0, 1, 10, 0) + 21c
```

```
fed6da4c JVM_sleep (36c04, ffbfed60, 0, fa0, 10, 0) + 27c
```

```
f9c0be48 ???????? (0, b8, 5, 8, 0, ffbfed78)
```

```
f9c05c64 ???????? (ffbfef78, 0, 0, f9c155e0, 3349f0, ffbfee18)
```

```
f9c00118 ???????? (ffbfef00, ffbff0e0, a, f58fa768, f9c0aae0, ffbfefec)
```

```
fecc84a8
```

```
__1cJJavaCallsLcall_helper6FpnJJavaValue_pnMmethodHandle_pnRJavaCallArguments_pnGThread__v_ (ffbff0d8, ffbfefb4, ffbfefe4, 36b70, 36b70, 0) + 274
```

```
fecdc674
```

```
_1cRjni_invoke_static6FpnHJNIEnv__pnJJavaValue_pnI_jobject_nLJNIcallType_pnK_jmethodID_pnSJNI_ArgumentPusher_pnGThread__v_ (36c04, f
```

```
fbff0d8, 0, 0, fbd90, ffbff0bc) + 218
```

```
fed660ec jni_CallStaticVoidMethod (36c04, 36dc8, fbd90, 36dd8, 36c04, ff0000) + 13c
```

```
00012ea4 main (36dac, 260, 0, fbd90, 488, 268) + 158c
```

```
000118f0 _start (0, 0, 0, 0, 0, 0) + 108
```

In short..

- Apparently, code was designed to run in parallel.
 - Concurrency implemented using java concurrency mechanisms
 - Code sleeps if it can't take a lock on that resource.
 - Code was modified.
-

References

- A. Cockcroft. *Sun Performance and Tuning*. SunSoft Press, Mountain View, California, 1st edition, 1995.
 - Man pages for proc hosted: <http://www.cs.biu.ac.il/cgi-bin/man?proc+1>
 - DBAzone article on “Mining information from Listener log” By Arup Nanda
<http://www.dbazine.com/oracle/or-articles/nanda14>
-