Performance tuning: Scientific approach to bottleneck identification

By

Riyaj Shamsudeen

The PYTHIAN GROUP your database maestros

OakTable.net

Who am I?

- 16 years using Oracle products
- Over 15 years as Oracle DBA
- Certified DBA versions 7.0,7.3,8,8i &9i
- Specializes in performance tuning, Internals and E-business suite
- Currently working for The Pythian Group www.pythian.com
- OakTable member
- Email: rshamsud at gmail.com
- Blog : orainternals.wordpress.com

OraInternals
Performance.Recovery.EBS11i

OakTable.net

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.

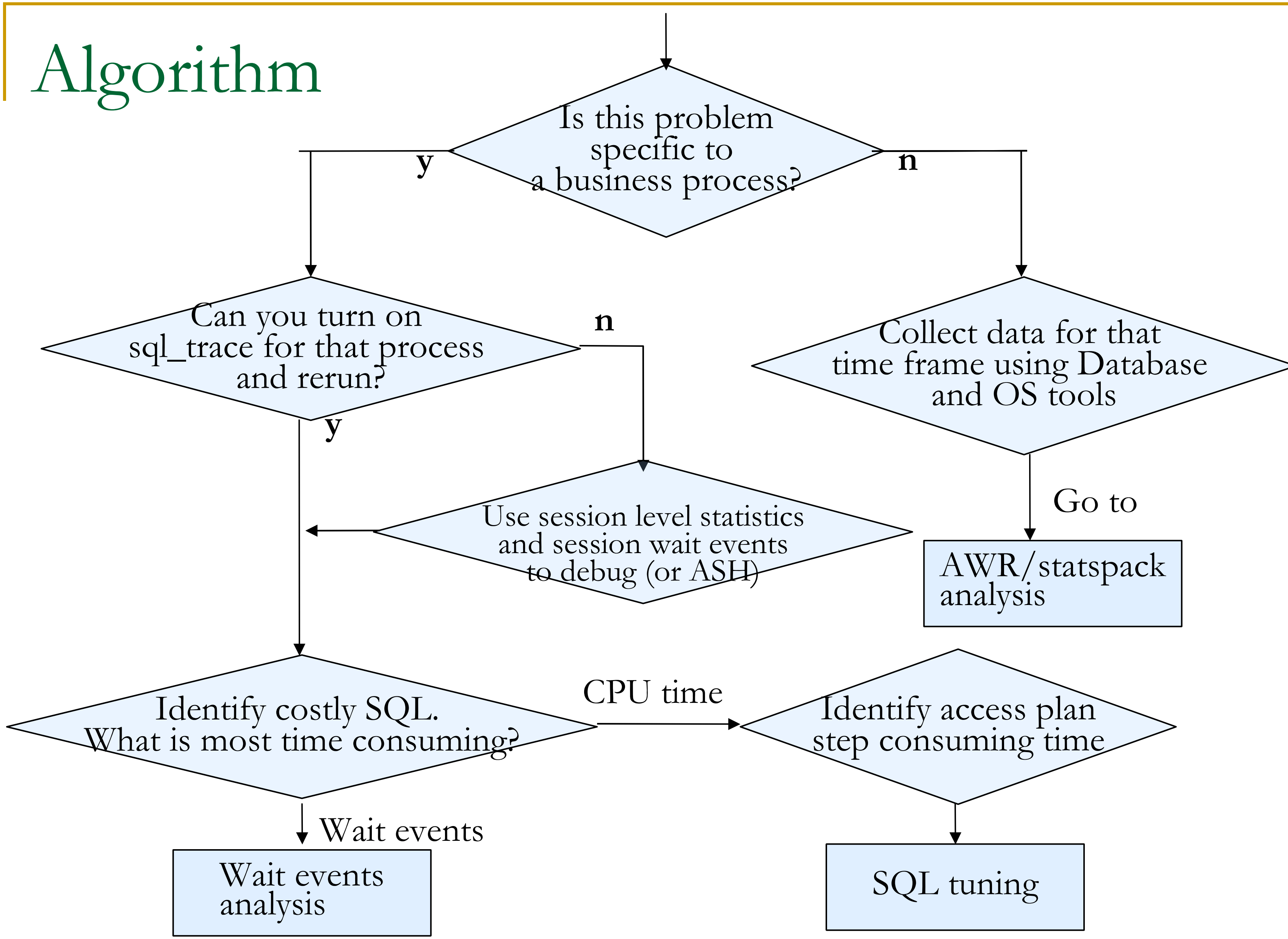
Agenda

- Algorithm – basic approach to tuning
- Issue: High CPU usage
- Issue: I/O problem
- Issue: ASMM
- Issue: Stuck program
- Issue: Out of memory exception

Algorithm

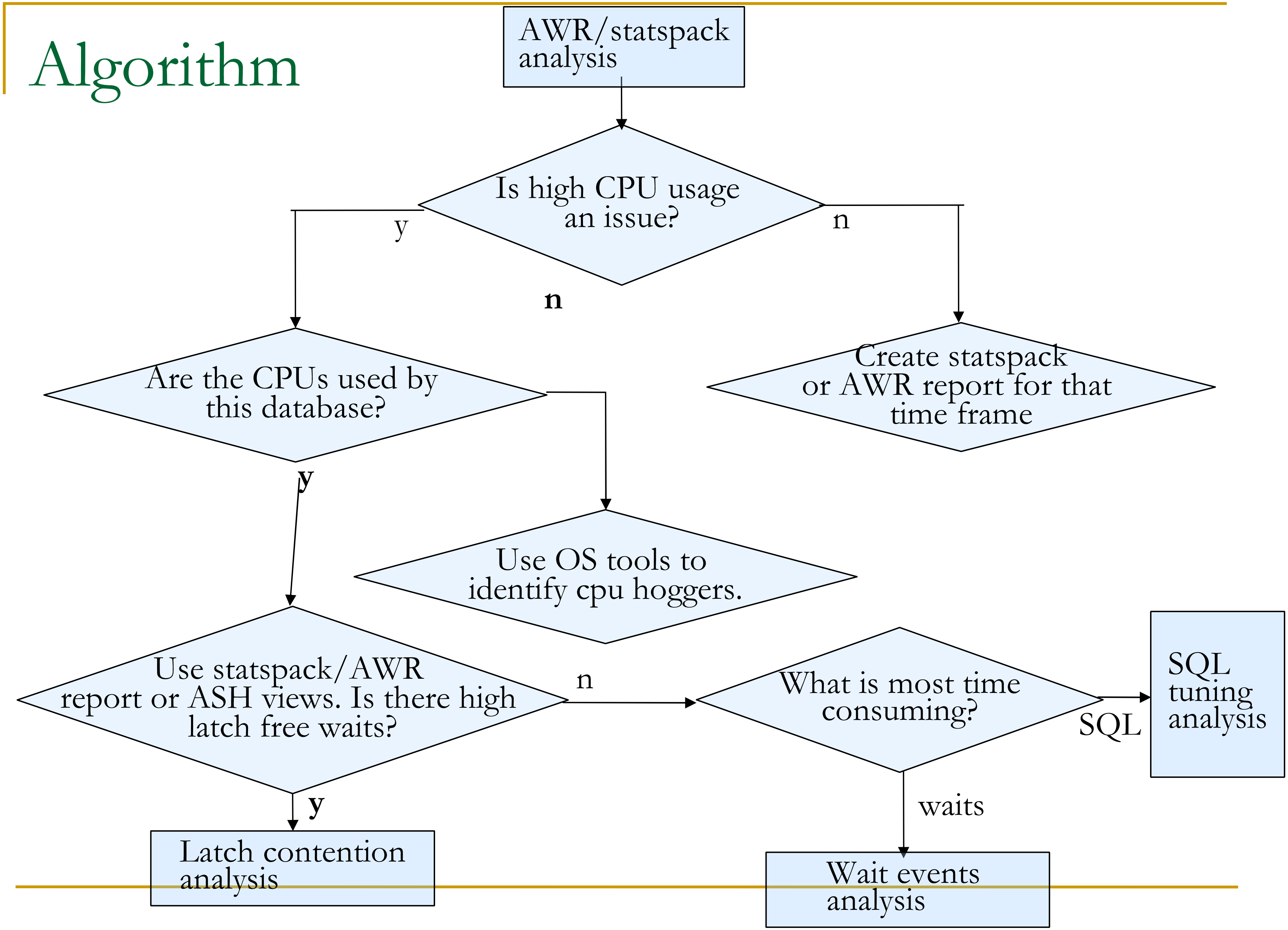
- Any scientific method has set of basic principles and algorithm to debug.
- This is my first attempt to create an algorithm that should encompass basic principles and provide a framework.
- This algorithm is not perfect and never will be. It is intended as a guideline.

Algorithm



Is this problem specific to a business process?
y
n
Can you turn on sql_trace for that process and rerun?
n
y
Collect data for that time frame using Database and OS tools
Go to
AWR/statspack analysis
Use session level statistics and session wait events to debug (or ASH)
Identify costly SQL. What is most time consuming?
CPU time
Identify access plan step consuming time
Wait events
Wait events analysis
SQL tuning

Algorithm



AWR/statspack analysis
Is high CPU usage an issue?
y
n
n
Are the CPUs used by this database?
Create statspack or AWR report for that time frame
y
Use OS tools to identify cpu hoggers.
Use statspack/AWR report or ASH views. Is there high latch free waits?
n
What is most time consuming?
SQL
SQL tuning analysis
y
waits
Latch contention analysis
Wait events analysis

Agenda

- Algorithm – basic approach to tuning
- Issue: Stuck program
- Issue: High CPU usage
- Issue: I/O problem
- Issue: ASMM
- Issue: Out of memory exception

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 sows 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  0 5360 31445 18498 8 6 86
 0 3 0 13308840 13727008 32 56 0 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\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, 0xFFFFFFFF7FFFDFCC, 1, 0xFFFFFFFF7FFFDFB8) 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, 0xFFFFFFFF7FFFDFCC, 1, 0xFFFFFFFF7FFFDFB8) = 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), 0xFFFFFFFF7FFFBE08) = 0
 1.2797 ioctl(10, (('V'<<24)|('X'<<16)|('O'<<8)|28), 0xFFFFFFFF7FFFBE08) = 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/rdsk/*/*

crw-------   1 root     root     233,63003 Jul  5 14:19 /dev/vx/rdsk/OAA/data
crw-------   1 root     root     233,49001 Feb 28  2007 /dev/vx/rdsk/APPS/gcontrol
crw-------   1 root     root     233,63004 Dec 15  2006 /dev/vx/rdsk/OAA/appldev
crw-------   1 root     root     233,49000 Dec 15  2006 /dev/vx/rdsk/APPS/apps
crw-------   1 root     root     233,38001 Dec 15  2006 /dev/vx/rdsk/UAT/data
crw-------   1 root     root     233,63001 Dec 15  2006 /dev/vx/rdsk/OAA/arch
crw-------   1 root     root     233,63002 Dec 15  2006 /dev/vx/rdsk/OAA/backup
crw-------   1 root     root     233,63000 Dec 15  2006 /dev/vx/rdsk/OAA/source
crw-------   1 root     root     233,63005 Dec 15  2006 /dev/vx/rdsk/OAA/tfsource
crw-------   1 root     root     233,38000 Dec 15  2006 /dev/vx/rdsk/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/appldev
                    41943040 32830352 9112688   79%    /apps/oa/appldev
/dev/vx/dsk/UAT/data
                    62914560 22409368 40199584  36%    /apps/oa/uat
```

Find command

<u>Solution</u>: 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.

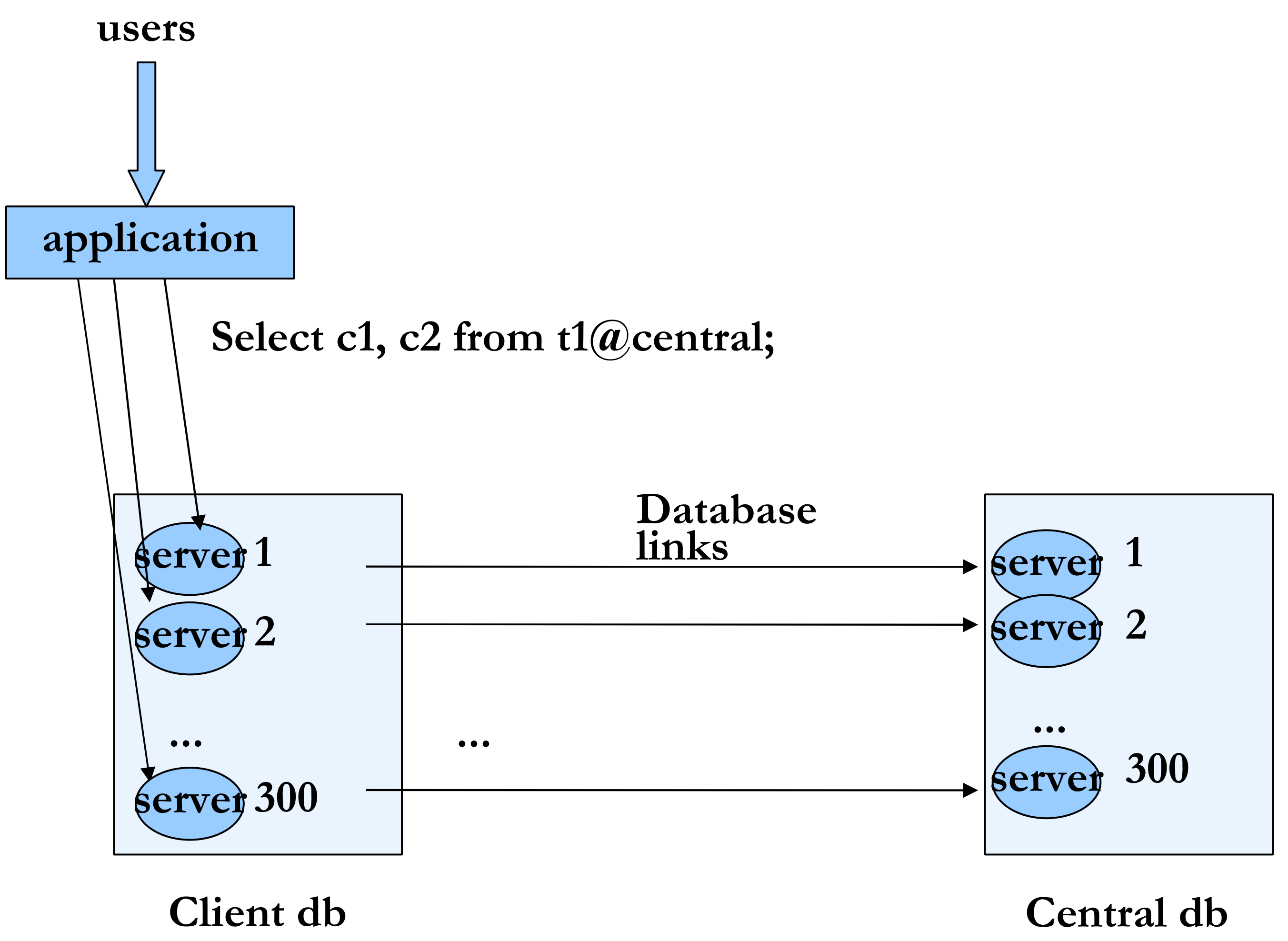
ls –lt /apps/oa/appldev/dev/devcomn/admin/out/DEV_xxxxxx/o2452107.out
-rw-r--r-- 1 appldev dba 6556971555 Jun 27 22:17

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

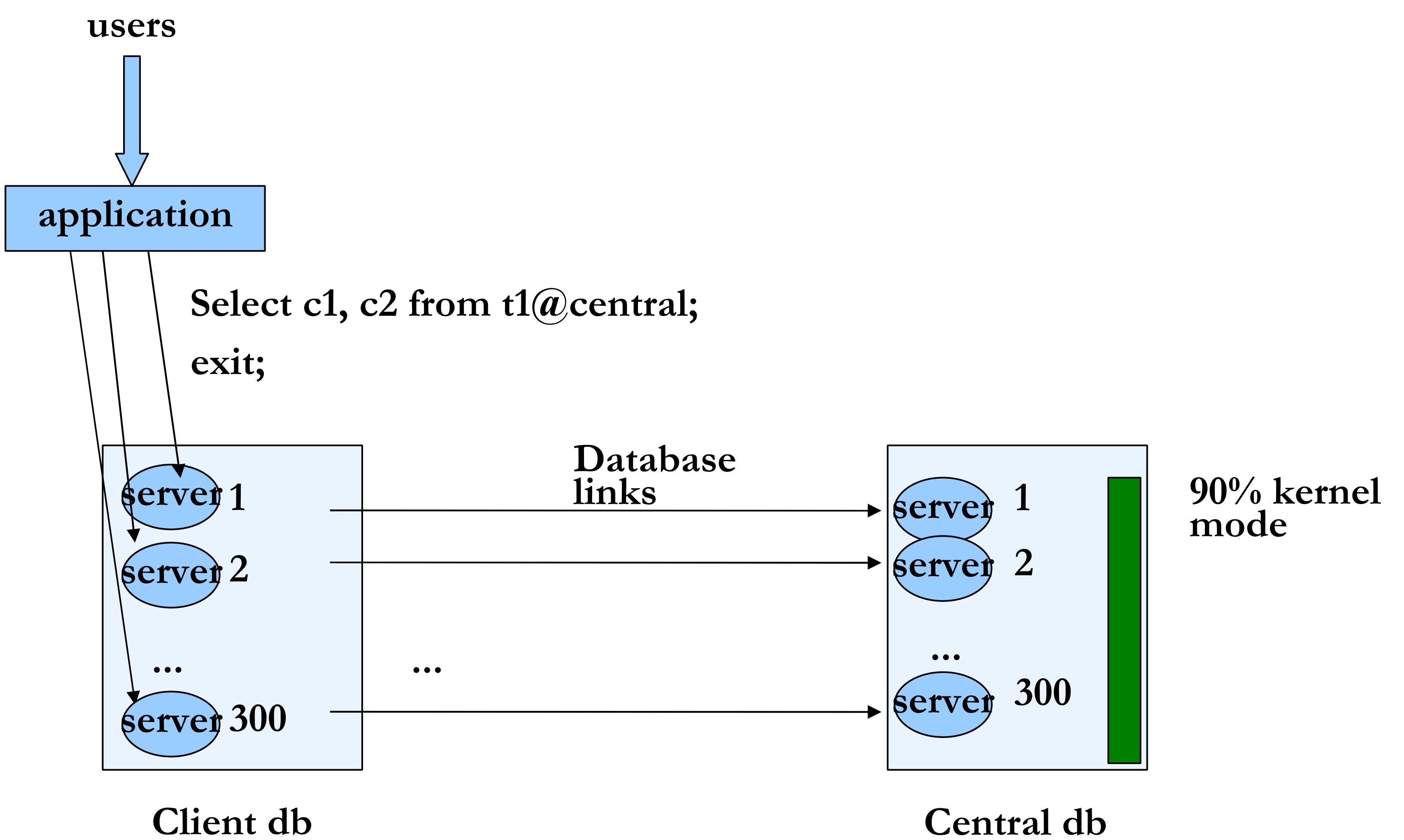
Agenda

- Algorithm – basic approach to tuning
- Issue: Stuck program
- Issue: High CPU usage
- Issue: I/O problem
- Issue: ASMM
- Issue: Out of memory exception

Application-Normal working



Issue: High Kernel mode CPU usage



Sys mode CPU usage

- Restart of an application core releases 300 database link based connections in another central database.
- This results in high kernel mode CPU usage in central database.

```
Tue Sep  9 17:46:34 2008CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
Tue Sep  9 17:46:34 2008  0  561   0    9   554  237  651  219   87  491    0  4349    9  91   0   0
Tue Sep  9 17:46:34 2008  1 1197   1   34   911    0 2412  591  353  630    0 15210   30  63   0   7
Tue Sep  9 17:46:34 2008  2   58   0    9   313    0  613  106  190  105    0  3562    8  90   0   2
Tue Sep  9 17:46:34 2008  3  161   0   26   255    0  492   92  161  530    0  2914    6  92   0   2
Tue Sep  9 17:46:34 2008  4    0   0    0    86    1    2    3    1   63    0     8    0 100   0   0
Tue Sep  9 17:46:34 2008  5  283   0   34   662    0 1269  153  326  211    0  6753   13  77   0  10
Tue Sep  9 17:46:34 2008  6  434   0   43   349    0  589   54  170 1534    0  3002    7  88   0   5
Tue Sep  9 17:46:34 2008  7  142   0   17   317   81  411  105   51   61    0  2484    4  93   0   3
Tue Sep  9 17:46:34 2008  8  213   0    0  6423 6321  697   36  148  546    0  3663    6  86   0   9
Tue Sep  9 17:46:34 2008  9  291   0   17   363    0  639   41  154 1893    0  3214    9  85   0   6
Tue Sep  9 17:46:34 2008 10  149   0    8   456    0  964  194  193   77    0  5797   10  81   0   8
Tue Sep  9 17:46:34 2008 11   17   0    0   104    0   42    3    9  183    0   207    0  95   0   5
Tue Sep  9 17:46:34 2008 12   30   0    0   195    0  279  110   31   80    0  1590    3  97   0   0
Tue Sep  9 17:46:34 2008 13  288   0    9   449    0  844  117  158  127    0  4486    7  85   0   8
Tue Sep  9 17:46:34 2008 14  155   0    0   430    0  744  102  160   83    0  3875    7  80   0  13
Tue Sep  9 17:46:34 2008 15   16   0    0   237    0  359  115   31  124    0  2074    3  91   0   6
```

Latch free

- There is also another, perceived, secondary symptom. Since all CPUs are in use, user processes will not get enough CPU cycles.

- This usually results in another symptom. In this case, latch contention on enqueues latches are another issue.

- We can't be sure that this is a secondary symptom, but kernel mode CPU usage issue must be resolved before attempting to resolve latch contention issue.

Issue(2): Latch free

Of 655seconds elapsed time, there were 455 seconds of waits for latch free

```
                  Snap Id      Snap Time        Sessions Curs/Sess
                --------- ------------------- -------- ---------
Begin Snap:          3131 09-Sep-08 17:46:17     5,030       1.9
  End Snap:          3132 09-Sep-08 17:47:16     4,995       2.0
   Elapsed:                 0.98 (mins)
   DB Time:                10.55 (mins)
```

Almost all latch free waits were for enqueues latch

```
Event                                 Waits    Time (s)   (ms)   Time Wait Class
------------------------------ ------------ ----------- ------ ------ ----------
latch free                              868         455    525   71.9      Other
latch: row cache objects                103         189   1833   29.8 Concurrenc
log file sync                           885          92    103   14.5     Commit
CPU time                                             85          13.4
db file parallel write                3,868          10      3    1.6 System I/O
```

latch contention is for enqueues latches:

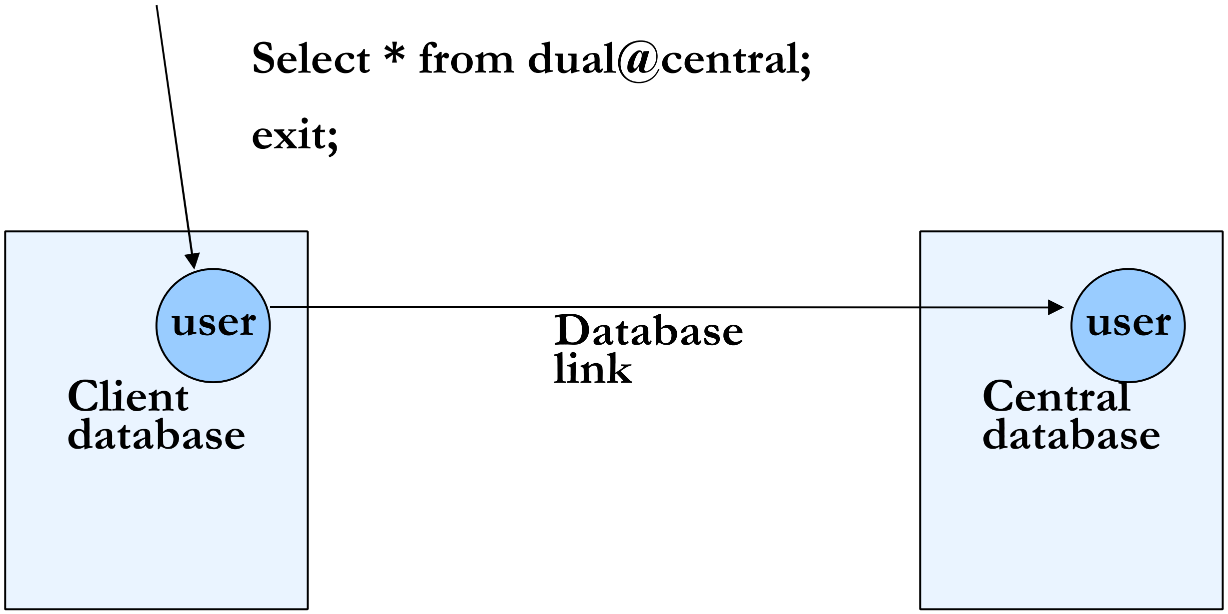
```
                                          Pct    Avg   Wait                 Pct
                                   Get    Get   Slps   Time       NoWait NoWait
Latch Name                    Requests   Miss  /Miss    (s)     Requests   Miss
------------------------ -------------- ------ ------ ------ ------------ ------
enqueues                      1,355,722    3.3    0.0    452            0    N/A
```

Testing logoffs

- We will ignore latch contention for now.
- To test kernel mode CPU issue, we will use database link in a schema, connecting to test schema in central database.
- From client database, selecting over database link creates a new connection in central database:

select * from dual@central;

Testing logoffs



Truss

- Identified the connection from test schema @ client database to test schema @ central database.

```
select sid, serial#, LOGON_TIME,LAST_CALL_ET from v$session where
  logon_time > sysdate-(1/24)*(1/60)
       SID    SERIAL# LOGON_TIME           LAST_CALL_ET
---------- ---------- -------------------- ------------
      1371      35028 12-SEP-2008 20:47:30            0
      4306      51273 12-SEP-2008 20:47:29            1 <---
```

- Truss is an utility to trace unix system calls.
- Starting truss on that pid in central db: d flag will print offset timestamp for that system call.

truss -p <pid> -d -o /tmp/truss.log

- Logged off from content database and this should trigger a log-off from remote core database.

Truss

- Reading truss output in central database connection, we can see ten shmdt calls are consuming time.

```
18.4630 close(10)                                   = 0
18.4807 shmdt(0x380000000)                          = 0
18.5053 shmdt(0x440000000)                          = 0
18.5295 shmdt(0x640000000)                          = 0
18.5541 shmdt(0x840000000)                          = 0
18.5784 shmdt(0xA40000000)                          = 0
18.6026 shmdt(0xC40000000)                          = 0
18.6273 shmdt(0xE40000000)                          = 0
18.6512 shmdt(0x1040000000)                         = 0
18.6752 shmdt(0x1240000000)                         = 0
18.6753 shmdt(0x1440000000)                         = 0
```

Shmdt calls are used to detach from shared memory segments during session logoff.

18.5295-18.5053=0.0242

- Each call consumed approximately 0.024 seconds or 24ms.

Shmdt calls

- There are 10 shared memory segments for this SGA. So, there are 10 shmdt calls.

```
ipcs -ma|grep 14382
m 1241514024   0x97e45100 --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   73728 14382  1090 21:32:30 21:32:30  1:32:50
m 1241514023   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:32:42
m 1241514022   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8187281408 14382  1090 21:32:30 21:32:30  1:32:34
m 1241514021   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:32:26
m 1241514020   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:32:18
m 1241514019   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:32:11
m 1241514018   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:32:03
m 1241514017   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:31:56
m 1241514016   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   8153726976 14382  1090 21:32:30 21:32:30  1:31:49
m  889192479   0          --rw-r-----   oracle orainvtr   oracle orainvtr   5436
   2986344448 14382  1090 21:32:30 21:32:30  1:31:47
```

8GB

Fun with numbers

- Each session consumes approximately 0.24 seconds. Shmdt is a system call and so cpu usage will be in kernel mode.
- 300 connections would consume 72 seconds.
- But, as concurrency goes up, these calls will slow down and use more mutex spins, which in turn will increase kernel mode cpu usage.
- At best case, with 12 concurrent processes, this would last for 6 seconds or so.
- This is matching with our observation of 6 seconds high kernel mode cpu usage.

Reducing shmdt calls

- Number of shmdt calls can be reduced by having one shared memory segment.
- Database engine tries to create biggest segment possible at initial startup and slowly reduces segment size, until segments can be created successfully.
- In this case, SHMMAX kernel parameter value is limiting and that's why RDBMS is creating ten segments, instead of one.
- So, to reduce number of shared memory segments, we need to increase or set SHMMAX to maximum value. There is no downside to that.
- This will likely require a server reboot.
- This change will also necessitate database restart too.

dtrace

- While we are almost certain that shmdt calls are causing this issue, we need to perform a verification analysis too.
- Dtrace can be used to see what calls are executed by peeking at CPUs.
- It is a possibility that dtrace might give a different view of the problem, but still it is prudent to perform this analysis too.
- Following dtrace command to be executed right before app core is shutdown and should be ctrl+C after CPU usage subsides.

```
dtrace -n 'syscall:::entry { @Calls[probefunc] = count(); }'
```

- For comparison, we should also execute above command for same number of seconds when it is considered normal.

Dtrace – top sys call

- We also need to use top_sys_calls.sh to understand top sys calls per second basis using top_sys_calls.sh script.
- Also dtrace is considered as a safe option in Solaris platform.

After SHMMAX change

- After SHMMAX change, we expected SGA to be fit in one segment.
- Surprise! Surprise! Surprise!
- Started to truss on database startup to see why the instance is creating multiple shared memory segments.

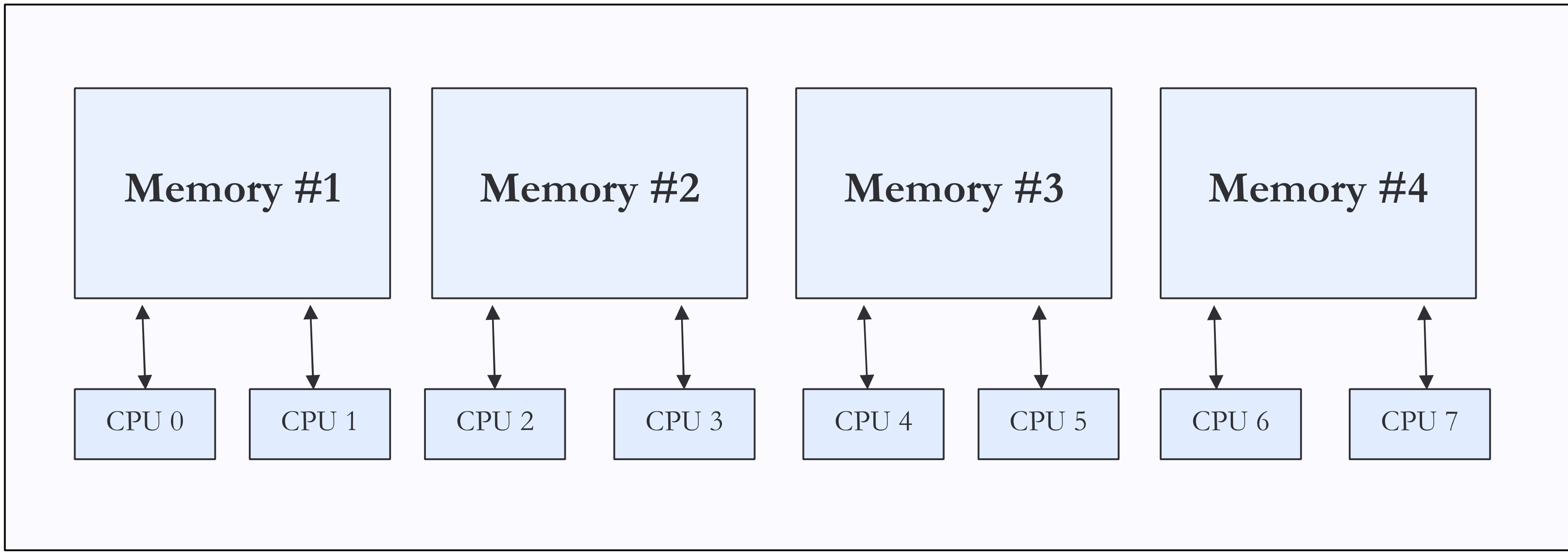
```
17252:   4.5957 munmap(0xFFFFFD7FFDAE0000, 32768)              = 0
17252:   4.5958 lgrp_version(1, )                              = 1
17252:   4.5958 _lgrpsys(1, 0, )                               = 42
17252:   4.5958 _lgrpsys(3, 0x00000000, 0x00000000)            = 19108
17252:   4.5959 _lgrpsys(3, 0x00004AA4, 0x06399D60)            = 19108
17252:   4.5959 _lgrpsys(1, 0, )                               = 42
17252:   4.5960 pset_bind(PS_QUERY, P_LWPID, 4294967295, 0xFFFFFD7FFFDFB11C) = 0
17252:   4.5960 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFFDFB0D4, 0x00000000) = 0
17252:   4.5961 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFFDFB0D4, 0x061AA2B0) = 0
```

pset_bind and _lgrpsys

- pset_bind and _lgrpsys gave a clue that this might be related to NUMA (Non-uniform memory access) architecture issues.

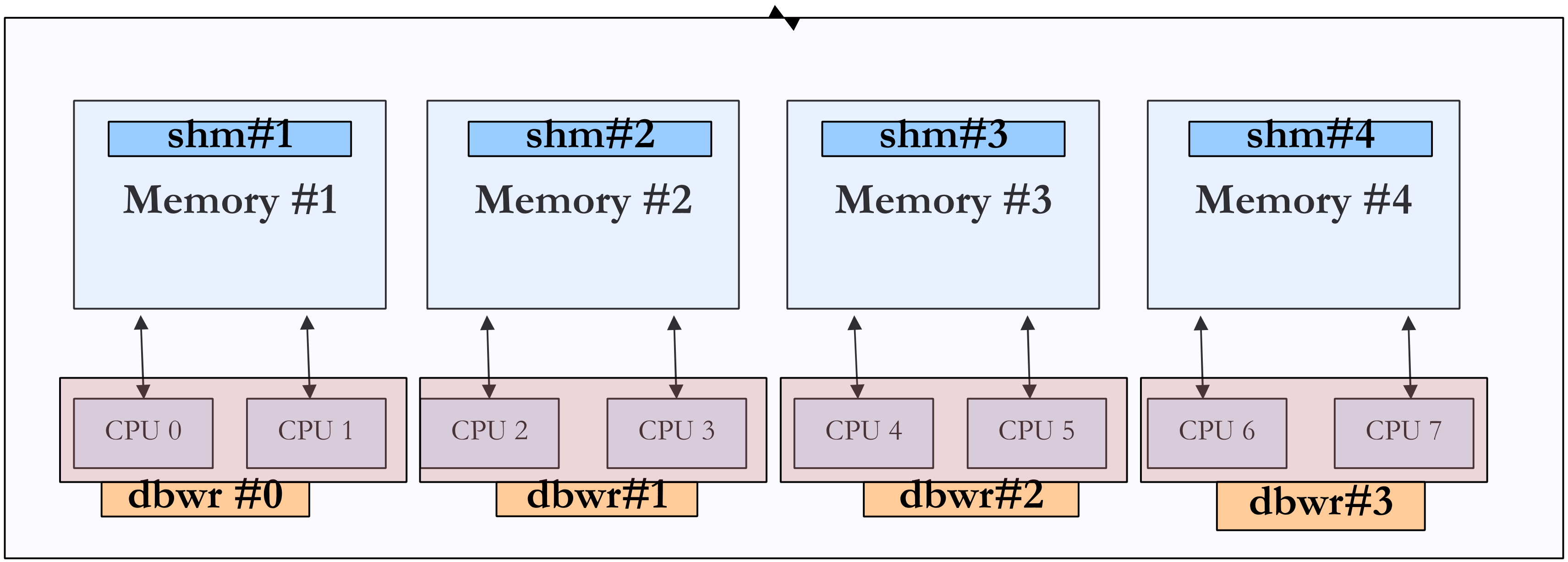
```
17252:   4.5957 munmap(0xFFFFFD7FFDAE0000, 32768)                  = 0
17252:   4.5958 lgrp_version(1, )                                  = 1
17252:   4.5958 _lgrpsys(1, 0, )                                   = 42
17252:   4.5958 _lgrpsys(3, 0x00000000, 0x00000000)                = 19108
17252:   4.5959 _lgrpsys(3, 0x00004AA4, 0x06399D60)                = 19108
17252:   4.5959 _lgrpsys(1, 0, )                                   = 42
17252:   4.5960 pset_bind(PS_QUERY, P_LWPID, 4294967295, 0xFFFFFD7FFFDFB11C) = 0
17252:   4.5960 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFFDFB0D4, 0x00000000) = 0
17252:   4.5961 pset_info(PS_MYID, 0x00000000, 0xFFFFFD7FFFDFB0D4, 0x061AA2B0) = 0
```

NUMA architecture (High level overview only)



- For cpu0 & cpu1, memory #1 is local. Other memory areas are remote for cpu0 & cpu1.
- Access to local memory is very fast compared to remote memory access.

NUMA architecture (High level overview only)



- To make use of NUMA technology, Oracle spreads SGA across all memory areas.
- Then Binds DBWR to a CPU set. That DBWR handles all writes from that shared memory segment. LGWR is also bound to a processor set.

Back to our issue

- In Solaris, NUMA technology is implemented as locality groups.
- _lgrpsys and pset_bind calls are to get current locality group information and bind processes to a processor set.
- Now, we can understand why SGA was split into multiple segments.
- But, Do we really have that many locality groups in this server?

Locality groups

- With the help of an eminent SA, we were able to download locality group aware tools from openSolaris.org and execute that in that platform.

```
:~#/usr/local/bin/lgrpinfo
lgroup 0 (root):
        Children: 10 12 14 15 17 19 21 23
        CPUs: 0-15
        Memory: installed 65024 Mb, allocated 2548 Mb, free 62476 Mb
        Lgroup resources: 1-8 (CPU); 1-8 (memory)
        Latency: 146
lgroup 1 (leaf):
        Children: none, Parent: 9
        CPUs: 0 1
        Memory: installed 7680 Mb, allocated 1964 Mb, free 5716 Mb
        Lgroup resources: 1 (CPU); 1 (memory)
        Load: 0.105
        Latency: 51
...
```

There were many locality groups defined and seven of them were leaf node locality groups.

Locality groups are defined as an hierarchical tree.

Latency

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	146	146	113	113	113	113	113	113	146	146	146	146	146	146	146	146	113	146	113	146	146	146	146	146	146
1	146	51	81	81	113	113	113	113	146	81	113	113	146	113	146	146	113	146	113	146	146	146	146	146	146
2	113	81	51	113	81	113	113	81	113	113	113	81	113	113	113	113	113	113	113	113	113	113	113	113	113
3	113	81	113	51	113	81	81	113	113	113	113	113	113	81	113	113	113	113	113	113	113	113	113	113	113
4	113	113	81	113	51	81	81	113	113	113	113	113	113	113	113	113	81	113	113	113	113	113	113	113	113
5	113	113	113	81	81	51	113	81	113	113	113	113	113	113	113	113	113	113	81	113	113	113	113	113	113
6	113	113	113	81	81	113	51	113	81	113	113	113	113	113	113	113	113	113	113	113	81	113	113	113	113
7	113	113	81	113	113	81	113	51	81	113	113	113	113	113	113	113	113	113	113	113	113	113	81	113	113
8	146	146	113	113	113	113	81	81	51	146	146	146	146	146	146	146	113	146	113	146	113	146	113	113	81
9	146	81	113	113	113	113	113	113	146	81	113	113	146	113	146	146	113	146	113	146	146	146	146	146	146
10	146	113	113	113	113	113	113	113	146	113	113	113	146	113	146	146	113	146	113	146	146	146	146	146	146
11	146	113	81	113	113	113	113	113	146	113	113	81	146	113	146	146	113	146	113	146	146	146	146	146	146
12	146	146	113	113	113	113	113	113	146	146	146	146	113	146	146	146	113	146	113	146	146	146	146	146	146
13	146	113	113	81	113	113	113	113	146	113	113	113	146	81	146	146	113	146	113	146	146	146	146	146	146
14	146	146	113	113	113	113	113	113	146	146	146	146	146	146	113	146	113	146	113	146	146	146	146	146	146
15	146	146	113	113	113	113	113	113	146	146	146	146	146	146	146	113	113	146	113	146	146	146	146	146	146

...

Summary

- Indeed 10 shared memory segments were created, one for a locality groups.
- We can disable NUMA or reduce number of NUMA nodes.

  *._enable_NUMA_optimization=FALSE

  *._db_block_numa = <smaller number>
- Of course, these are underscore parameters and so need Oracle support.
- Note 399261.1 describes that.
- It looks like, there is one shared memory segment per locality group, one encompassing all locality groups and one small bootstrap segment.

Agenda

- Algorithm – basic approach to tuning
- Issue: High CPU usage
- Issue: I/O problem
- Issue: ASMM
- Issue: Stuck program
- Issue: Out of memory exception

I/O problem?

- Performance of an application is poor in some areas. Client could not pinpoint any specific process.
- DBA blamed Storage and vice versa.
- 10 minutes statspack report showed:

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                   wait   Call
Event                                            Waits    Time (s)   (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
db file sequential read                        581,785       1,718      3   73.6
db file scattered read                          29,932         308     10   13.2
CPU time                                                       235          10.1
SGA: allocation forcing component growth         2,259          22     10    1.0
log file parallel write                          4,770          11      2     .5
```

I/O problem ?

- Only 7.7% of single block reads exceeded 8ms response time.
- Performance of single block read is within acceptable range. We need to reduce demand.

```
                               Total ----------------- % of Waits ------------------
Event                          Waits  <1ms  <2ms  <4ms  <8ms <16ms <32ms  <=1s   >1s
-------------------------- ----- ----- ----- ----- ----- ----- ----- ----- -----
...
db file scattered read           29K  45.7   8.5   9.0  12.1  11.5   8.0   5.2    .0
db file sequential read         581K  76.8   2.9   4.2   8.5   5.1   1.5   1.1    .0
...
```

I/O problem?

- We need to find SQLs/processes consuming I/O cycles.
- Reviewing SQLs consuming I/O cycles, one SQL consumes 233K out of 581K physical reads.

```
 Elapsed                   Elap per              CPU                         Old
  Time (s)   Executions    Exec (s)  %Total    Time (s)  Physical Reads Hash Value
---------- ------------ ---------- ------ ---------- --------------- ----------
    848.72        8,565       0.10   31.6     396.71         233,839 4109372156
Module: oradb10@lawapp (TNS V1-V3)
select /*+ INDEX_ASC("PYD" "PYDSET1")*/  TO_CHAR("PYD"."COMPANY"
,'FM0999'), TO_CHAR("PYD"."EMPLOYEE",'FM099999999'),
 TO_CHAR("PY"."CHECK_ID",'FM099999999999'), "PYD"."DED_CODE", "PYD"."PROCES
S_LEVEL", "PYD"."WORK_STATE", TO_CHAR("PYD"."PAYROLL_YEAR",'FM0999'),
 TO_CHAR("PYD"."QUARTER",'FM0'), TO_CHAR("PYD"."EDM_SEQ_NBR
```

I/O problem ?

- Problem was evident after looking at access plan.
- DBA dropped and recreated index PYDSET1. While recreating he made a typo in index name. CBO chose an inefficient index causing increased single block reads.
- Solution was easy: Rename that index and let Oracle re-parse those SQLs.

Agenda

- Algorithm – basic approach to tuning
- Issue: Stuck program
- Issue: High CPU usage
- Issue: I/O problem
- Issue: ASMM
- Issue: Out of memory exception

ASMM ?

- Problem: Application was very slow. Database was recently upgraded to 10g. Management was quite unhappy and blamed it on 10g.
- Even rollback to 9i was on the table as an option. Following report is for 3 minutes duration.

```
Top 5 Timed Events                                                  Avg %Total
~~~~~~~~~~~~~~~~~~                                                 wait   Call
Event                                         Waits    Time (s)    (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
db file sequential read                        96,535         704      7   30.1
SGA: allocation forcing component growth       50,809         498     10   21.3
library cache pin                                 180         219   1218    9.4
latch: shared pool                              2,767         217     78    9.3
log file switch completion                        225         216    958    9.2
```

ASMM ?

- From top 5 timed events, high amount of SGA activity, waits for library cache pin and latch contention etc.
- Application opens new connections if it “detects” that SQL is hung.

```
Top 5 Timed Events                                                   Avg %Total
~~~~~~~~~~~~~~~~~~                                                  wait   Call
Event                                            Waits    Time (s)   (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
db file sequential read                         96,535         704      7   30.1
SGA: allocation forcing component growth        50,809         498     10   21.3
library cache pin                                  180         219   1218    9.4
latch: shared pool                               2,767         217     78    9.3
log file switch completion                         225         216    958    9.2
```

ASMM ?

- 'SGA: allocation forcing component growth' gives a clue that there is SGA re-alignment occurring.
- Looking at SGA area section of statspack:

  Buffer cache increased by 32 MB &

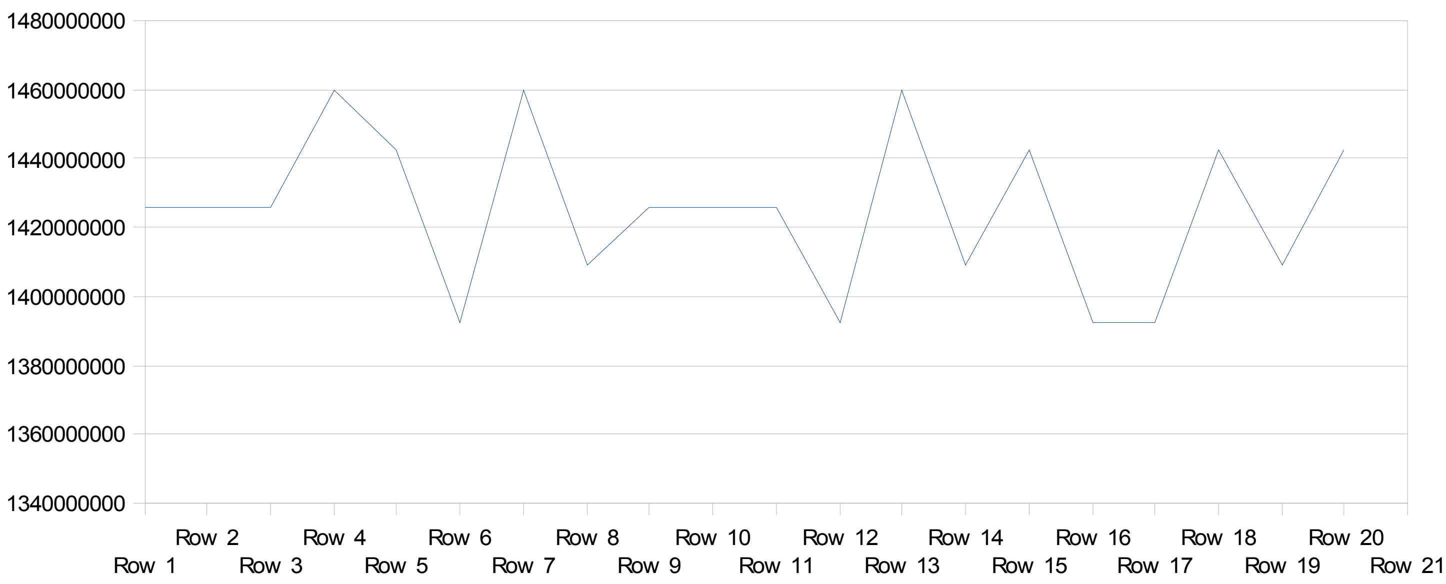
  Shared pool decreased by 32 MB

Snap Id	Cache	Prior Size (MB)	New Size (MB)	Difference (MB)
181	Buffer Cache	1,376	1,344	-32
	Shared Pool	288	320	32

ASMM ?

- Plotting buffer_cache size from perfstat table shows that buffer cache is in constant reorganization.

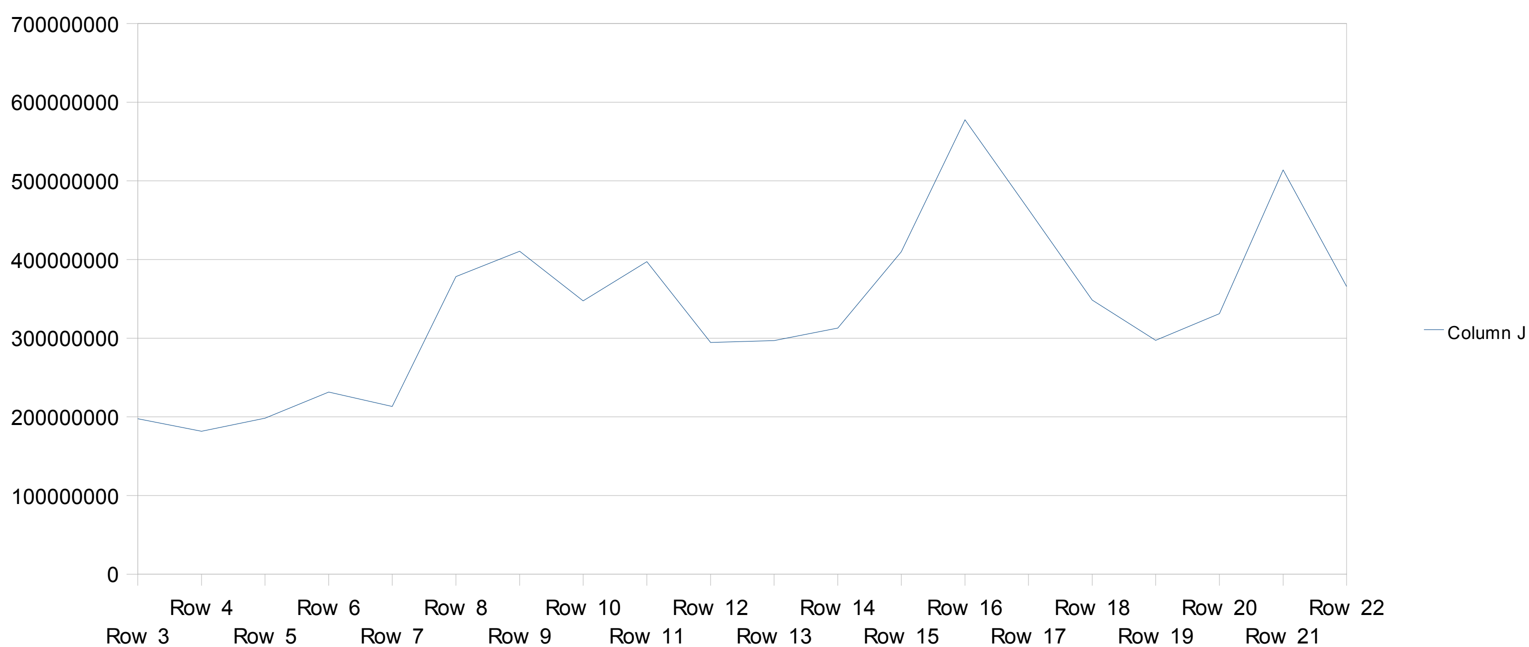
```
select * from perfstat.stats$sgastat where
name='buffer_cache'  order by snap_id;
```



ASMM?

- Plotting heaps with KGH:NO ACCESS tag shows turbulant reorganization.

select * from perfstat.stats$sgastat where name='KGH: NO ACCESS' order by snap_id



ASMM ?

- In summary, constant reorganization of SGA areas caused this issue.
- DBA has setup sga_target and commented out all other memory parameters.
- Memory was allocated and deallocated constantly from shared pool. This increased parsing, library cache pin waits and shared pool latch contention etc.
- buffer_cache was undergoing this reorganization too, resulting in higher disk reads.

Agenda

- Algorithm – basic approach to tuning
- Issue: I/O problem
- Issue: ASMM
- Issue: Stuck program
- Issue: Out of memory exception

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

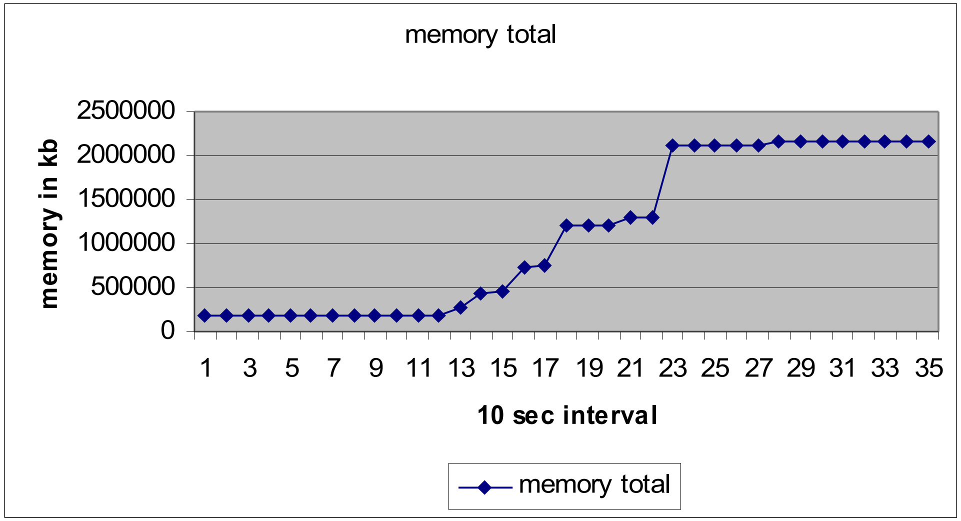
```
Address   Kbytes      RSS    Anon  Locked Mode   Mapped File
00010000      72       72       -       - r-x--  java
00030000      16       16      16       - rwx--  java
00034000    8808     8720    8720       - rwx--    [ heap ]
77980000    1224     1048       -       - r--s-  dev:273,2000 ino:104403
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.

Agenda

- Algorithm – basic approach to tuning
- Issue: Stuck program
- Issue: High CPU usage
- Issue: I/O problem
- Issue: ASMM
- Issue: Out of memory exception
- Issue: Database freeze due to high CPU usage

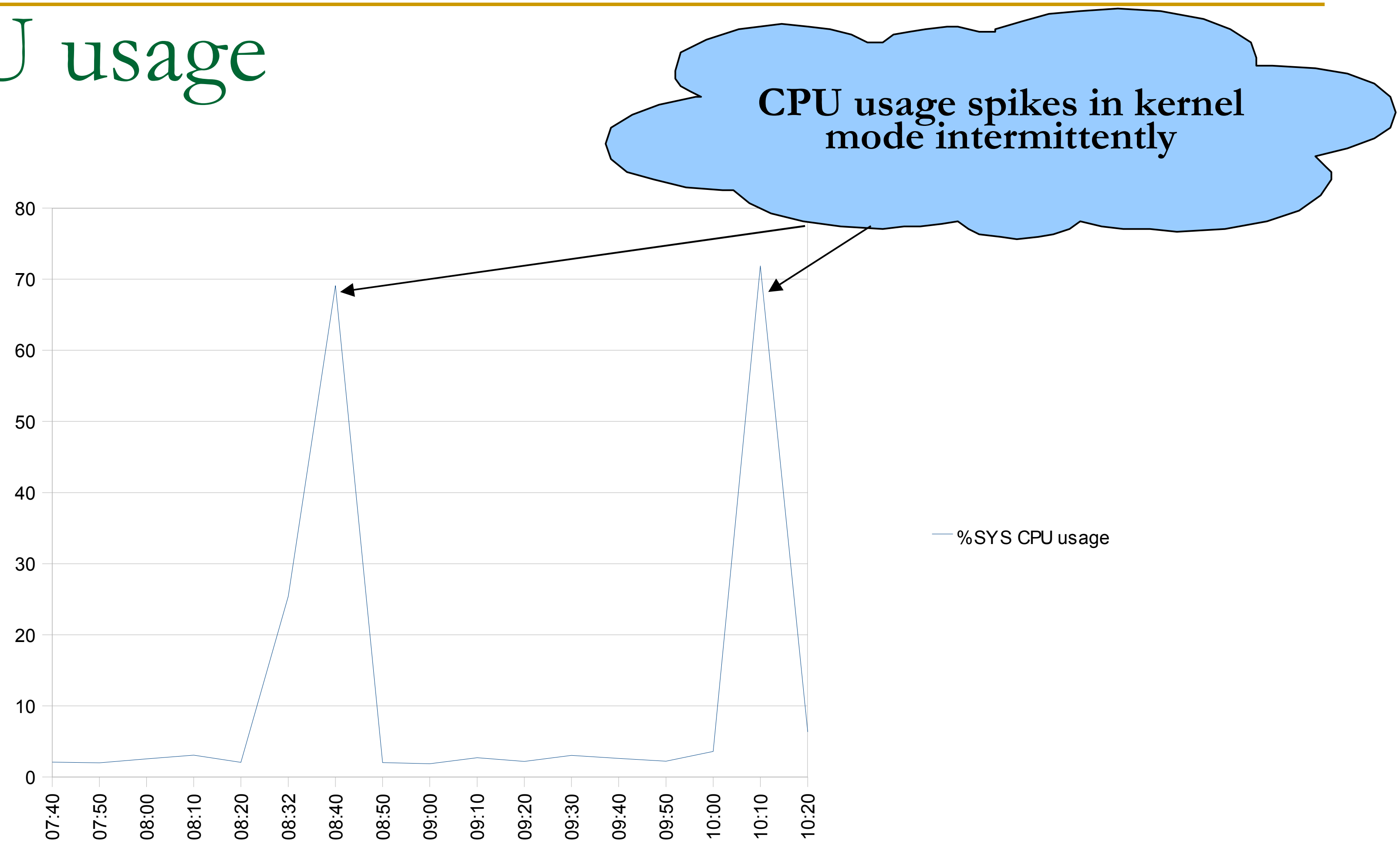
Problem description

- Database was intermittently freezing and business was severely affected.
- This is a central database and client had already worked with vendor support and problem was still unresolved.
- Symptom: High Kernel mode CPU usage intermittently in a 4 CPU dual core, hyper threading enabled Linux server.

CPU usage

07:20:01 AM	CPU	%user	%nice	%system	%iowait	%idle
07:30:01 AM	all	4.85	0.00	77.40	4.18	13.58
07:40:01 AM	all	16.44	0.00	2.11	22.21	59.24
07:50:01 AM	all	23.15	0.00	2.00	21.53	53.32
08:00:01 AM	all	30.16	0.00	2.55	15.87	51.41
08:10:01 AM	all	32.86	0.00	3.08	13.77	50.29
08:20:01 AM	all	27.94	0.00	2.07	12.00	58.00
08:32:50 AM	all	25.97	0.00	25.42	10.73	37.88
08:40:02 AM	all	16.40	0.00	69.21	4.11	10.29
08:50:01 AM	all	35.82	0.00	2.10	12.76	49.32
09:00:01 AM	all	35.46	0.00	1.86	9.46	53.22
09:10:01 AM	all	31.86	0.00	2.71	14.12	51.31
09:20:01 AM	all	26.97	0.00	2.19	8.14	62.70
09:30:02 AM	all	29.56	0.00	3.02	16.00	51.41
09:40:01 AM	all	29.32	0.00	2.62	13.43	54.62
09:50:01 AM	all	21.57	0.00	2.23	10.32	65.88
10:00:01 AM	all	16.93	0.00	3.59	14.55	64.92
10:10:01 AM	all	11.07	0.00	71.88	8.21	8.84
10:20:01 AM	all	46.08	0.00	6.35	21.03	26.54

CPU usage



CPU usage spikes in kernel mode intermittently
80
70
60
50
40
30
20
10
0
07:40
07:50
08:00
08:10
08:20
08:32
08:40
08:50
09:00
09:10
09:20
09:30
09:40
09:50
10:00
10:10
10:20
%SYS CPU usage

Tools?

- Client was using few tools, but none of the tools were helpful in this case.
- Well problem is that those tools are averaging out over a longer period of time and doesn't show any issues.
- We decided to look at all statistics at 8:40AM and 10:10AM

Sar -r

Watch free memory closely around the time of incident

- Fortunately, sar data was handy. Looking at free memory, something is odd.

07:40:01 AM	kbmemfree	kbmemused	%memused	kbbuffers	kbcached	kbswpfree	kbswpused
07:50:01 AM	225968	20323044	98.90	173900	7151144	19186948	1785900
08:00:01 AM	206688	20342324	98.99	127600	7084496	19134200	1838648
08:10:01 AM	214152	20334860	98.96	109728	7055032	19138552	1834296
08:20:01 AM	209920	20339092	98.98	21268	7056184	19139548	1833300
08:32:50 AM	86176	20462836	99.58	8240	7040608	19018136	1954712
08:40:02 AM	1157520	19391492	94.37	79096	7012752	18753272	2219576
08:50:01 AM	1523808	19025204	92.58	158044	7095076	18887204	2085644
09:00:01 AM	775916	19773096	96.22	187108	7116308	18895220	2077628
09:10:01 AM	430100	20118912	97.91	218716	7129248	18908060	2064788
09:20:01 AM	159700	20389312	99.22	239460	7124080	18817920	2154928
09:30:02 AM	265184	20283828	98.71	126508	7090432	18814584	2158264
10:41:54 AM	78588	20470424	99.62	4092	6962732	18239648	2733200
10:50:01 AM	4787684	15761328	76.70	77400	6878012	18418460	2554388
11:00:01 AM	2636892	17912120	87.17	143780	6990176	18523344	2449504
11:10:01 AM	1471236	19077776	92.84	186540	7041712	18575912	2396936

Data analysis

- It is evident that free memory dropped to a smaller value.
- Then, free memory went up by couple of Mbs.
- We can derive two things:
  - Enormous memory pressure at 8:32
  - Memory was released back at 8:40
- Of course, there is paging and swapping going on.
- But, can that be justify high CPU usage in kernel mode?
- What about database freeze?

Memory breakup!

- Server has 20GB of memory.
- SGA uses approximately 10GB. Database uses direct I/O and so UNIX buffer usage must be minimal.
- PGA target is 2GB and maximum ever allocated is 800MB.
- No other application running in that database.
- Connection count is 500.
- So, where is remaining 9GB is used?
- Client argument is that there shouldn't be any paging or swapping.

Memory breakup!

5GB allocated for PageTables alone!

```
cat /proc/meminfo
MemTotal:      20549012 kB
MemFree:         236668 kB
Buffers:          77800 kB
Cached:         7189572 kB
SwapCached:      108332 kB
Active:         9738788 kB
Inactive:         55376 kB
HighTotal:            0 kB
HighFree:             0 kB
LowTotal:      20549012 kB
LowFree:         236668 kB
SwapTotal:     20972848 kB
SwapFree:      18600172 kB
```

```
Dirty:                556 kB
Writeback:              0 kB
Mapped:           9672544 kB
Slab:              162968 kB
CommitLimit:     31247352 kB
Committed_AS:    42142384 kB
PageTables:       5007924 kB
VmallocTotal:   536870911 kB
VmallocUsed:       275316 kB
VmallocChunk:   536595383 kB
HugePages_Total:        0
HugePages_Free:         0
Hugepagesize:        2048 kB
```

Memory breakup!

HugePages are not being used.

```
cat /proc/meminfo
MemTotal:       20549012 kB
MemFree:          236668 kB
Buffers:           77800 kB
Cached:          7189572 kB
SwapCached:       108332 kB
Active:          9738788 kB
Inactive:          55376 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:       20549012 kB
LowFree:          236668 kB
SwapTotal:      20972848 kB
SwapFree:       18600172 kB
```

```
Dirty:                556 kB
Writeback:              0 kB
Mapped:           9672544 kB
Slab:              162968 kB
CommitLimit:     31247352 kB
Committed_AS:    42142384 kB
PageTables:       5007924 kB
VmallocTotal:   536870911 kB
VmallocUsed:       275316 kB
VmallocChunk:   536595383 kB
HugePages_Total:        0
HugePages_Free:         0
Hugepagesize:        2048 kB
```

Page size

- OS page size is 4KB.
- 20GB /4KB is 5.2 million OS pages.
- SGA is part of these 5.2 million OS pages.
- Just like any other memory page, SGA buffers also can be paged out.
- If there is a free memory need, then kscand/kswapd scans 5.2 million OS pages looking to free memory.
- That's why we had all CPUs used by kernel trying to free memory during memory starvation.

Solution

- Fix is easy enough.
- We need to reduce pagetables size and reduce pages needed to be scanned by kscand/kswapd.
- Enter hugepages.
- We setup SGA to use hugepages. SGA using hugepages is locked in memory and not page dout.
- Pagetables size went down to 200MB or so.
- Database performance was within acceptable level.

References

- Oracle support site. Metalink.oracle.com. Various documents
- Internal's guru Steve Adam's website

www.ixora.com.au

- Jonathan Lewis' website

www.jlcomp.daemon.co.uk

- Julian Dyke's website

www.julian-dyke.com

- 'Oracle8i Internal Services for Waits, Latches, Locks, and Memory'

by Steve Adams

- Tom Kyte's website

Asktom.oracle.com

- Blog: http://orainternals.wordpress.com