Oracle Exadata and database memory

Frits Hoogland
`whoami`

- Frits Hoogland
- Working with Oracle products since 1996
- Blog: http://fritshoogland.wordpress.com
- Twitter: @fritshoogland
- Email: frits.hoogland@accenture.com
- Oracle ACE Director
- OakTable Member
Goals & prerequisites

• Goal: Learn about Linux and Oracle database SGA&PGA memory specifics.

• Prerequisites:
  – Understanding of Linux (memory).
  – Understanding of Oracle memory.

• I talk about dedicated (non shared server) conn.
Memory models

- The Oracle database has 3 memory models:
  - Manual
  - ASMM
  - AMM
## Memory models

<table>
<thead>
<tr>
<th>SGA/Shared Memory</th>
<th>Manual</th>
<th>ASMM</th>
<th>AMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_buffer</td>
<td>log_buffer</td>
<td>log_buffer</td>
<td>log_buffer</td>
</tr>
<tr>
<td>db_cache_size</td>
<td>db_cache_size</td>
<td>db_cache_size</td>
<td>db_cache_size</td>
</tr>
<tr>
<td>shared_pool_size</td>
<td>shared_pool_size</td>
<td>shared_pool_size</td>
<td>shared_pool_size</td>
</tr>
<tr>
<td>large_pool_size</td>
<td>large_pool_size</td>
<td>large_pool_size</td>
<td>large_pool_size</td>
</tr>
<tr>
<td>java_pool_size</td>
<td>java_pool_size</td>
<td>java_pool_size</td>
<td>java_pool_size</td>
</tr>
<tr>
<td>streams_pool_size</td>
<td>streams_pool_size</td>
<td>streams_pool_size</td>
<td>streams_pool_size</td>
</tr>
<tr>
<td>PGA/Private Memory</td>
<td>pga_aggregate_target</td>
<td>pga_aggregate_target</td>
<td>pga_aggregate_target</td>
</tr>
</tbody>
</table>

**Manual**
- Configuration files:
  - `log_buffer`
  - `db_cache_size`
  - `shared_pool_size`
  - `large_pool_size`
  - `java_pool_size`
  - `streams_pool_size`

**ASMM**
- Configuration files:
  - `log_buffer`
  - `db_cache_size`
  - `shared_pool_size`
  - `large_pool_size`
  - `java_pool_size`
  - `streams_pool_size`

**AMM**
- Configuration files:
  - `log_buffer`
  - `db_cache_size`
  - `shared_pool_size`
  - `large_pool_size`
  - `java_pool_size`
  - `streams_pool_size`

**Memory Models**

- **Manual**: Individual configuration.
- **ASMM**: Memory target.
- **AMM**: Memory target.
All memory models SGA require:

- System V shared memory
$ sysresv -l fv12102
.. lots of other output ..

Shared Memory:

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>117047297</td>
<td>0x00000000</td>
</tr>
<tr>
<td>117080066</td>
<td>0x00000000</td>
</tr>
<tr>
<td>117014528</td>
<td>0x00000000</td>
</tr>
<tr>
<td>117112835</td>
<td>0xd160bbe8</td>
</tr>
</tbody>
</table>

Semaphores:

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>557058</td>
<td>0x7e75d94c</td>
</tr>
</tbody>
</table>

Oracle Instance alive for sid “fv12102”
$ sysresv -l fv12102
.. lots of other output ..

Oracle Instance alive for sid “fv12102”
SGA - AMM memory

$ sysresv -l fv12102
.. lots of other output ..

Shared Memory:

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>127598593</td>
<td>0x00000000</td>
</tr>
<tr>
<td>127631362</td>
<td>0x00000000</td>
</tr>
<tr>
<td>127565824</td>
<td>0x00000000</td>
</tr>
<tr>
<td>127664131</td>
<td>0xd160bbe8</td>
</tr>
</tbody>
</table>

Semaphores:

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>557058</td>
<td>0x7e75d94c</td>
</tr>
</tbody>
</table>

Oracle Instance alive for sid “fv12102”

$ ls /dev/shm/ora_fv12102_127598593_* | wc -l
  85

$ ls /dev/shm/ora_fv12102_127631362_* | wc -l
  9

$ ls /dev/shm/ora_fv12102_127565824_* | wc -l
  1

$ ls /dev/shm/ora_fv12102_127664131_* | wc -l
ls: cannot access /dev/shm/ora_fv12102_127664131_*: No such file or directory
  0
### SGA - AMM memory

```bash
$ ls -l /dev/shm/ora_fv12102_127598593_*
-rw-r----- 1 oracle oinstall 0 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_0
-rw-r----- 1 oracle oinstall 0 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_1
-rw-r----- 1 oracle oinstall 0 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_10
-rw-r----- 1 oracle oinstall 0 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_11
..  
-rw-r----- 1 oracle oinstall 16777216 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_31
-rw-r----- 1 oracle oinstall 16777216 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_32
-rw-r----- 1 oracle oinstall 16777216 Feb 9 08:39 /dev/shm/ora_fv12102_127598593_33
-rw-r----- 1 oracle oinstall 16777216 Feb 9 08:40 /dev/shm/ora_fv12102_127598593_34
-rw-r----- 1 oracle oinstall 16777216 Feb 9 08:50 /dev/shm/ora_fv12102_127598593_35
```

- **Size either 0 or granule size**

- **Memory segments**
Memory (ASMM & man)

- The SGA can be seen in /proc/PID/maps:

```bash
$ cat /proc/19735/maps
00400000-1093f000 r-xp 00000000 fc:02 272770084 /u01/.../bin/oracle
10b3e000-10dbf000 rw-p 1053e000 fc:02 272770084 /u01/.../bin/oracle
10dbf000-10df0000 rw-p 00000000 00:00 0
12266000-122a8000 rw-p 00000000 00:00 0 [heap]
60000000-60400000 rw-s 00000000 00:0b 140083200 /SYSV00000000 (deleted)
60400000-96400000 rw-s 00000000 00:0b 140115969 /SYSV00000000 (deleted)
96400000-9ea00000 rw-s 00000000 00:0b 140148738 /SYSV00000000 (deleted)
9ec00000-9ec05000 rw-s 00000000 00:04 140181507 /SYSVd160bbe8 (deleted)
3bb3000000-3bb3020000 r-xp 00000000 fc:00 134595 /lib64/ld-2.12.so
..
Memory (AMM)

- Actually, the SGA files can be seen in maps too:

```
$ cat maps
..
ba000000-bb000000 rw-s 00000000 00:10 92010 /dev/shm/ora_fv12102_144474114_4
bb000000-bc000000 rw-s 00000000 00:10 92011 /dev/shm/ora_fv12102_144474114_5
bc000000-bd000000 rw-s 00000000 00:10 92012 /dev/shm/ora_fv12102_144474114_6
bd000000-be000000 rw-s 00000000 00:10 92013 /dev/shm/ora_fv12102_144474114_7
be000000-bf000000 rw-s 00000000 00:10 92014 /dev/shm/ora_fv12102_144474114_8
bf000000-bf002000 rw-s 00000000 00:04 144506883 /SYSVd160bbe8 (deleted)
3bb3000000-3bb3020000 r-xp 00000000 fc:00 134595 /lib64/ld-2.12.so
..
Which memory model?

• Now that we have looked at the memory models.

• The obvious question is: what to choose?
  • Manual.
  • Automatic Shared Memory Management.
  • Automatic Memory Management.
Which memory model?

• *Note the shown behaviour is limited to Linux!*

• *Investigate your own platform for options.*

• I would strongly advise to NOT use AMM.
  • */dev/shm* allocations can not use huge pages.

• For serious deployments huge pages is not a choice. It is the only way to go.

• Only the SGA can be stored in huge pages!
What are huge pages

- Linux administers memory per page.
  - Normal page size 4kB.
- Every process needs maintain a table of it’s virtual addresses to physical addresses.
  - Called ‘pagetable’.
- The process pagetable is not pre-build.
  - Entries are added as pages are touched.
  - This includes (database) buffer cache pages.
What are huge pages

• With hugepages, a page size of 2M is used.
  – Reserved at startup (non-usable for 4kB alloc!);
    • (sysctl.conf) vm.nr_hugepages.
  – Hugepages are non-swappappable.
  – Not automatic: memory allocations need to be done explicitly for huge pages.

• Huge pages results:
  – Much smaller page tables.
  – Higher TLB hit%, less CPU.
Huge pages

• Let me provide you a testcase.
  • Machine with 4G memory.
  • Database with buffer cache of ~ 1.2 G.
  • Start 100 connections which do a
    • select count(*) from table;
  • Table size 960M.
Huge pages

• This is very easy to try yourself:
  • Create a table that should fit in the buffercache.
  • alter system set “_serial_direct_read”=never;
  • alter table <tablename> cache;
  • This modifies the BC LRU behaviour

T=0
while [ $T -lt 100 ]; do
  sqlplus ts/ts << EOF &
  select count(*) from <tablenname>;
  exec dbms_lock.sleep(900);
EOF
  let T=$T+1
  sleep 0.3
done
Huge pages

• Results:
  • After startup of database:
    • Non hugepages: 41M - 1.6%
    • Hugepages: 29M - 1.1%
  • After run with 100 sessions using the cache:
    • Non hugepages: 344M - 13.8%
    • Hugepages: 90M - 3.6%

$ grep -i pagetable /proc/meminfo
PageTables: 41278 kB
Hugepages

• Current usage of huge pages:

```
$ grep -i hugepages /proc/meminfo
HugePages_Total:   32768
HugePages_Free:       58
HugePages_Rsvd:       58
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

• Further reading:
Hugepages

• The database side
  – Parameter:
    • use_large_pages (>= 11.2.0.2)
      » TRUE (default)
      » ONLY
      » FALSE
Hugepages

• Oracle 11.2.0.1:
  • No use_large_pages

• Oracle 11.2.0.2 and higher:
  • Set use_large_pages TRUE/ONLY

• Oracle 11.2.0.3 and higher:
  • Mixed normal/huge pages (TRUE)
SGA size

Size matters!
SGA size

BUFFER CACHE   Size matters!   WITH EXADATA
SGA size

BUFFER CACHE Size matters! WITH EXADATA
Bigger is not always better!
SGA size

Block(s) request: In cache?  
Yes  Logical I/O
No

Single block request?  
Yes
Buffered read:
- db file sequential read
- cell single block physical read
No

Grouped single blocks request?  
Yes
Buffered read:
- db file parallel read
- cell list of blocks physical read
No

Multiblock request.

Optimizer decision!
SGA size

All decisions here are done on RUNTIME!
No statistics/optimizer involved.
_direct_read_decision_statistics_driven
>= 11.2.0.2: TRUE -> BLOCKS

Direct path read
↓
# blks
↓
cache >= 99%
↓
dirty > 50%
↓
No
↓
Yes

Buffered read
↓
direct path read
↓
cell smart table/index scan
↓
Yes
↓
No
↓

Multiblock request.
↓
segment < STT
↓
segment > VLOT
↓
Compressed?
(HCC/OLTP)
↓
Yes
↓
No
↓

Buffered read: db file scattered read
↓
cell multiblock physical read
↓

These are the ONLY places where a SMARTSCAN is considered!
SGA size

• How did I get to these values?
  • There is a trace to see the evaluation:

```sql
alter session set events 'trace[nsmtio]';
```

• My setup:
  • Buffercache: 4’352M
  • _small_table_threshold: 106M (2%)
  • _very_large_object_threshold=500 (fixed)
  • actual size= 500%*BC= 21’760M
SGA size

• Small table (6 blocks), STT=13.276

NSMTIO: kcbism: islarge 0 next 0 nblks 6 type 3, bpid 65535, kcbisdbfc 0 kcbnhl 65536 kcbstt 13276 keep_nb 0 kcbnhb 512958 kcbnwp 4
NSMTIO: kcbism: islarge 0 next 0 nblks 6 type 2, bpid 3, kcbisdbfc 0 kcbnhl 65536 kcbstt 13276 keep_nb 0 kcbnhb 512958 kcbnwp 4

NSMTIO: qertbFetch:NoDirectRead:[- STT < OBJECT_SIZE < MTT]:Obect's size: 6 (blocks), Threshold: MTT(66382 blocks), _object_statistics: enabled, Sage: enabled, Direct Read for serial qry: enabled(::::kctfsage::), Ascending SCN table scan: FALSE flashback_table_scan: FALSE, Row Versions Query: FALSE SqlId: fp4x7ms1295y7, plan_hash_value: 4018852438, Object#: 46418, Parition#: 0 DW_scan: disabled
SGA size

Multiblock request.

segment < STT

segment > VLOT

Direct path read: direct path read
    cell smart table|index scan
        Yes

Buffered read: db file scattered read
    cell multiblock physical read

Compressed? (HCC/OLTP)

Yes

# blks cache >= 99%
dirty > 50%

No

Buffered read
    Yes

Direct path read
    No

# blks cache >= 99%
! dirty > 99%

Yes

Buffered read
    Yes

Direct path read
    No
SGA size

- Very large table (5,005,396 blocks), VLOT=3,319,140

NSMTIO: kcbism: islarge 1 next 0 nblks 5005396 type 3, bpdi 65535, kcbisdbfc 0 kcbnhl 65536 kcbstt 13276 keep_nb 0 kcbnbh 512958 kcbnwp 4
NSMTIO: kcbism: islarge 1 next 0 nblks 5005396 type 2, bpdi 3, kcbisdbfc 0 kcbnhl 65536 kcbstt 13276 keep_nb 0 kcbnbh 512958 kcbnwp 4
NSMTIO: kcbimd: nblks 5005396 kcbsttt 13276 kcbpnb 66382 kcbisdbfc 3 is_medium 0
NSMTIO: kcbivlo: nblks 5005396 vlot 500 pnb 663828 kcbisdbfc 0 is_large 1

NSMTIO: qertbFetch:DirectRead:[OBJECT_SIZE>VLOT]

NSMTIO: Additional Info: VLOT=3319140
Object# = 44540, Object_Size = 5005396 blocks
SqlId = 8jbzjj236m5zd, plan_hash_value = 630573765, Partition# = 0
SGA size

Multiblock request.

segment < STT

segment > VLOT

Compressed? (HCC/OLTP)

Direct path read:
direct path read
cell smart table|index scan

Yes

No

Buffered read:
db file scattered read
cell multiblock physical read

Yes

Buffered read

Direct path read

# blks

cache >= 99%
dirty > 50%

Yes

No

# blks

cache >= 99%
! dirty > 99%

Yes

Buffered read

Direct path read

# blks

cache >= 99%
dirty > 99%

Yes

No
**SGA size**

- **Large table (13.328 blocks) - just large enough**

---

NSMTIO: kcbism: isize 1 next 0 nblks 13328 type 2, bpids 3, kcbisdbfc 0 kcbnhl 65536 kcbstt 13276 keep_nb 0 kcbnhb 512958 kcbnwp 4

NSMTIO: kcbimd: nblks 13328 kcbstt 13276 kcbpn 66382 kcbisdbfc 3 is_medium 0

NSMTIO: kcbcmt1: hit age_diff adjts last_ts nbuf nblk has_val kcbisdbfc cache_it 0 280021454 280021454 512958 13328 1 0 1

NSMTIO: kcbvlo: nblks 13328 vlot 500 pn 663828 kcbisdbfc 0 is_large 0

NSMTIO: gertbFetch:[MTT < OBJECT_SIZE < VLOT]: Checking cost to read from caches(local/remote) and checking storage reduction factors (OLTP/EHCC Comp)

NSMTIO: kcbdpc:NoDirectRead:[CACHE_READ]: tsn: 7, objd: 46429, objn: 46429

ckpt: 0, nblks: 13328, ntcache: 13328, ntdist:0

Direct Path for pdb 0 tsn 7 objd 46429 objn 46429

Direct Path 0 ckpt 0, nblks 13328 ntcache 13328 ntdist 0

Direct Path mbdb 0 tdiob 13 txiob 0 tciob 19

Direct path diomrc 128 dios 2 kcbisdbfc 0

NSMTIO: Additional Info: VLOT=3319140

Object# = 46429, Object_Size = 13328 blocks

SqlId = ghfymk6a2nnkm, plan_hash_value = 3246516050, Partition# = 0
SGA size

Multiblock request. -> segment < STT -> segment > VLOT

Buffered read: db file scattered read cell multiblock physical read

Direct path read: direct path read cell smart table | index scan

Compressed? (HCC/OLTP)

Yes No

# blks cache >= 99%
dirty > 50%

Buffered read

# blks cache >= 99%
! dirty > 99%

Buffered read

Direct path read
SGA size

• Large table (226.384 blocks)

NSMTIO: kcbism: islarge 1 next 0 nblks 226384 type 2, bpid 3, kcbisdbfc 0 kcbnhl 65536 kcbstt 13276 keep_nb 0 kcbnbh 512958 kcbnwp 4
NSMTIO: kcbimd: nblks 226384 kcbstt 13276 kcbpnb 66382 kcbisdbfc 3 is_medium 0
NSMTIO: kcbivlo: nblks 226384 vlot 500 pnb 663828 kcbisdbfc 0 is_large 0
NSMTIO: qertbFetch:[MTT < OBJECT_SIZE < VLOT]: Checking cost to read from caches(local/remote) and checking storage reduction factors (OLTP/EHCC Comp)
NSMTIO: kcbdpc:DirectRead: tsn: 7, objd: 21540, objn: 21540
  ckpt: 1, nblks: 226384, ntcache: 1, ntdist:0
  Direct Path for pdb 0 tsn 7 objd 21540 objn 21540
  Direct Path 1 ckpt 1, nblks 226384 ntcache 1 ntdist 0
  Direct Path mndb 0 tdiob 13 txiob 0 tciob 19
  Direct path diomrc 128 dios 2 kcbisdbfc 0
NSMTIO: Additional Info: VLOT=3319140
Object# = 21540, Object_Size = 226384 blocks
SqlId = 71ppm0a6g1ysx, plan_hash_value = 4220890033, Partition# = 0
<table>
<thead>
<tr>
<th></th>
<th>Direct path</th>
<th>Table &gt; STT &amp; &lt; 5 * STT</th>
<th>Table &gt; 5 * STT uncompressed buffered</th>
<th>Table &gt; 5 * STT OLTP comp. buffered</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2.0.1</td>
<td>5 * STT</td>
<td>-</td>
<td>&gt; 99% or &gt; 75% dirty</td>
<td>&gt; 99% and ! 99% dirty</td>
</tr>
<tr>
<td>11.2.0.2</td>
<td>5 * STT</td>
<td>-</td>
<td>&gt; 99% or &gt; 75% dirty</td>
<td>&gt; 99% or &gt; 99% dirty</td>
</tr>
<tr>
<td>11.2.0.3</td>
<td>STT</td>
<td>Always direct path</td>
<td>&gt; 99% or &gt; 50% dirty</td>
<td>&gt; 99% or &gt; 99% dirty</td>
</tr>
<tr>
<td>11.2.0.4</td>
<td>STT</td>
<td>Always direct path</td>
<td>&gt; 99% or &gt; 50% dirty</td>
<td>&gt; 99% or &gt; 99% dirty</td>
</tr>
<tr>
<td>12.1.0.1</td>
<td>STT</td>
<td>Always direct path</td>
<td>&gt; 99% or &gt; 50% dirty</td>
<td>&gt; 99% and ! 99% dirty</td>
</tr>
<tr>
<td>12.1.0.2</td>
<td>STT</td>
<td>Always direct path</td>
<td>&gt; 99% or &gt; 50% dirty</td>
<td>&gt; 99% and ! 99% dirty</td>
</tr>
</tbody>
</table>
SGA conclusion

• You should use manual or ASMM memory.
• Even for the ASM instance(!)
  • MOS note: 1373255.1
• SGA should use huge pages.
• Buffer cache size determines buffering decision.
• Segment must be big enough for direct path.
  • Only direct path is considered for smartscan!
  • This means a buffercache too big can exclude segments from being smartscanned!
SGA conclusion

• With ASMM and manual memory SGA size is static.

• With AMM, shared memory granules are deflated/inflated based on need and PGA usage.
  • On best effort (!).
  • Please don’t use AMM.
PGA

- PGA works the same way with all memory models.
- PGA memory is *private* to the process.
- By default, PGA memory allocations are:
  - done with the `mmap()`.
    - Anonymous memory mapping.
    - Copy on write.
- This can be seen in maps & `smaps`.
$ cat /proc/19735/maps

.. snippet ..

3bb5c00000-3bb5c16000 r-xp 00000000 fc:00 150740 /lib64/libnsl-2.12.so
3bb5c16000-3bb5e15000 ---p 00016000 fc:00 150740 /lib64/libnsl-2.12.so
3bb5e15000-3bb5e16000 r--p 00015000 fc:00 150740 /lib64/libnsl-2.12.so
3bb5e16000-3bb5e17000 rw-p 00016000 fc:00 150740 /lib64/libnsl-2.12.so
3bb5e17000-3bb5e19000 rw-p 00000000 00:00 0

7fd18198f000-7fd1819cf000 rw-p 00000000 00:00 0
7fd1819cf000-7fd181a8f000 rw-p 00000000 00:00 0
7fd181a8f000-7fd181b7f000 rw-p 00000000 00:00 0
7fd181b7f000-7fd181bf000 rw-p 00000000 00:00 0
7fd181bf000-7fd181c0f000 rw-p 00000000 00:00 0

7fd181c0f000-7fd181d95000 r-xp 00000000 fc:02 11036576 /u01/../lib/libshpkavx12.so
7fd181d95000-7fd181f94000 ---p 00186000 fc:02 11036576 /u01/../lib/libshpkavx12.so
7fd181f94000-7fd181fa6000 rw-p 00185000 fc:02 11036576 /u01/../lib/libshpkavx12.so

.. snippet ..
PGA

• Dump PGA allocations:

SYS@fv12102 AS SYSDBA> oradebug setospid 19735
Oracle pid: 22, Unix process pid: 19735, image: oracle@bigmachine.local (TNS V1-V3)
SYS@fv12102 AS SYSDBA> oradebug unlimit
Statement processed.
SYS@fv12102 AS SYSDBA> oradebug dump heapdump 1
Statement processed.
Let’s look inside the dump:

```bash
$ grep -e ^HEAP -e ^Total -e ^EXTENT fv12102_ora_19735.trc
HEAP DUMP heap name="pga heap" desc=0x7fd182686980
EXTENT 0 addr=0x7fd181aff010
..  
EXTENT 35 addr=0x7fd182339b60
Total heap size = 633576
Total free space = 29696
HEAP DUMP heap name="top call heap" desc=0x7fd18268cb80
EXTENT 0 addr=0x7fd181aff008
Total heap size = 393072
Total free space = 390248
HEAP DUMP heap name="top uga heap" desc=0x7fd18268cda0
EXTENT 0 addr=0x7fd181b4f008
Total heap size = 917264
Total free space = 146944
```
• Watch what happens if you compare the heap dump with v$sesstat:

SYS@fv12102 AS SYSDBA> @pga 66
current pga: 979464
current uga: 413720

$ grep -e ^HEAP -e ^Total fv12102_ora_4570.trc
HEAP DUMP heap name="pga heap" desc=0x7f8ebf3eb980
Total heap size = 585096
Total free space = 16392
HEAP DUMP heap name="top call heap" desc=0x7f8ebf3f1b80
Total heap size = 65512
Total free space = 63000
HEAP DUMP heap name="top uga heap" desc=0x7f8ebf3f1da0
Total heap size = 327560
Total free space = 0
PGA

• From a tuning perspective, the PGA consists of two types of memory:

  • Tunable memory.
  • Untunable memory.
PGA

- Tunable memory is:
  - Effectively the workarea's:
    - SORT, HASH JOIN, GROUP BY, BUFFER
    - BITMAP MERGE, BITMAP CONSTRUCTION
  - These are tuned by PGA_AGGREGATE_TARGET
  - Workarea’s are allocated in:

    UGA (session heap in the top uga heap)
PGA

• Documentation describing UGA:

http://docs.oracle.com/database/121/CNCPT/memory.htm#CNCPT1238

• Documentation describing PGA:
PGA

• Untunable memory is:
  • Session information.
  • Private SQL area:
    • Cursor information, cursor state, PLSQL state, variables, PLSQL variables.

• Essentially the PGA memory allocations.
PGA

• For tunable memory area’s
  • _pga_max_size: 200M > 20% P_A_T
  • _smm_(px_)max_size: 10% (50%)

• Tuning works well, buffers can be sized based on PGA memory pressure.
• Easy to flush contents to/from disk.
PGA - assoc. array

• Untunable allocations:

(dioncho_pga_filler.sql)

```sql
declare
    type vc2_ar is table of varchar2(32767) index by pls_integer;
    vc vc2_ar;
    v varchar2(32767);
begin
    for idx in 1 .. 30000 loop
        v := rpad('x',32767,'x');
        vc(idx) := v;
    end loop;
end;
/
```
PGA - assoc. array

• Untunable allocations:

SYS@fv12102 AS SYSDBA> @dioncho_pga_filler
begin pga size : 3273224
begin uga size : 1037704
end  pga size : 920252936
end  uga size : 1109312
parameter pat : 524288000

PL/SQL procedure successfully completed.
PGA - open cursors

- Untunable allocations:

```
(oc_simplified.sql)

create or replace procedure recursive_open_cursor( p_number in number )
as
  l_cursor sys_refcursor;
begin
  if ( p_number = 0 ) then
    dbms_lock.sleep(120);
    return;
  end if;
  open l_cursor for select count(*) from dual;
  recursive_open_cursor(p_number-1);
  close l_cursor;
end;
/
```
PGA - open cursors

• Memory allocated after logging on:

SYS@fv12102 AS SYSDBA> @pga 66
current pga: 1110536
current uga: 348232

• Run oc_simplified.sql in session with sid 66:

TS@fv12102 > exec recursive_open_cursor( 900 );
.. session waiting in dbms_lock.sleep ..

• Memory allocated with 900 cursors:

SYS@fv12102 AS SYSDBA> @pga 66
current pga: 27128328
current uga: 22155736
PGA - open cursors

- Of course this can be tested with multiple connections:

```bash
T=0
while [ $T -lt 100 ]; do
    sqlplus ts/ts << EOF &
    exec recursive_open_cursor( 900 );
EOF
    let T=$T+1
    sleep 0.3
done
echo "done."
```
PGA - open cursors

• This gotten me some interesting results.
• First of all, it lead to failure messages:

(screen output while running the load script)

TS@fv12102 > TS@fv12102 > TS@fv12102 > TS@fv12102 > TS@fv12102 > TS@fv12102 > BEGIN recursive_open_cursor( 900 ); END;

* 
ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 16049
Session ID: 32 Serial number: 32745
The root cause is Oracle 12’s new PGA limit parm:

- **PGA_AGREGATE_LIMIT**

(alert_fv12102.log)

Tue Mar 03 14:39:58 2015
PGA memory used by the instance exceeds PGA_AGREGATE_LIMIT of 2048 MB
Immediate Kill Session#: 330, Serial#: 5840
Immediate Kill Session: sess: 0x9522fac8 OS pid: 15917

Tue Mar 03 14:41:36 2015
PGA memory used by the instance exceeds PGA_AGREGATE_LIMIT of 2048 MB
Immediate Kill Session#: 13, Serial#: 49196
Immediate Kill Session: sess: 0x95727268 OS pid: 15923
PGA - open cursors

• PGA_AGGREGATE_LIMIT default value:
  • The greater of:
    • 200% of PGA_AGGREGATE_TARGET.
    • 3MB * PROCESSES.
    • 2 GB.
  • Can be below 200% PAT if:
    • PAT > 90% TotMem-SGA, but minimal 100% PAT.

• In order to ignore this, I set P_A_L to 20G.
• I ran the 100 sessions again.
• The server started swapping.
• Oracle will tell you if the system is swapping:

(alert_fv12102.log)
Tue Mar 03 14:46:49 2015
WARNING: Heavy swapping observed on system in last 5 mins.
pct of memory swapped in [38.06%] pct of memory swapped out [9.74%].
Please make sure there is no memory pressure and the SGA and PGA
are configured correctly. Look at DBRM trace file for more details.
Errors in file /u01/app/oracle/diag/rdbms/fv12102/fv12102/trace/fv12102_dbrm_14757.trc
(incident=58464):
ORA-00700: soft internal error, arguments: [kskvmstatact: excessive swapping observed],
[ ], [ ], [ ], [ ], [ ], [ ], [ ], [ ], [ ], [ ]
Incident details in: /u01/app/oracle/diag/rdbms/fv12102/fv12102/incident/incdir_58464/
fv12102_dbrm_14757_i58464.trc
PGA - open cursors

• Once the sessions have logged on.

• ...and fetched their cursors....

• This is my PGA_AGGREGATE_TARGET:

```sql
SYS@fv12102 AS SYSDBA> select value/power(1024,2) from v$pgastat
where name = 'aggregate PGA target parameter';
```
```
VALUE/POWER(1024,2)
-------------------
500
```

• This is the true allocated PGA memory:

```sql
SYS@fv12102 AS SYSDBA> select value/power(1024,2) from v$pgastat
where name = 'total PGA allocated';
```
```
VALUE/POWER(1024,2)
-------------------
2208.83203
```
PGA_AGGREGATE_LIMIT

- SYS and background processes* are **not** subject to the limit.
- The limiting is not “absolute”.

Let’s test this:

- I set PGA_AGGREGATE_LIMIT to 600M.
- And run dioncho_pga_filler.sql again.
PGA_AGGREGATE_LIMIT

• Result (pga_aggregate_limit set to 600m):

TS@fv12102 > @dioncho_pga filler
error message : ORA-04036: PGA memory used by the instance exceeds PGA_AGGREGATE_LIMIT
begin pga size : 2748936
begin uga size : 1298312
end   pga size : 967242248
end   uga size : 1633176
parameter pat  : 524288000

PL/SQL procedure successfully completed.
The alert.log includes the ORA-4036:

Tue Mar 03 20:06:31 2015
Errors in file /u01/app/oracle/diag/rdbms/fv12102/fv12102/trace/fv12102 ora_27060.trc (incident=63384):
ORA-04036: PGA memory used by the instance exceeds PGA_AGGREGATE_LIMIT
Incident details in: /u01/app/oracle/diag/rdbms/fv12102/fv12102/incident/incdir_63384/fv12102 ora_27060 i63384.trc

The process trace file shows a little more:

$ grep -e ^Process -e Just -e ^sending /u01/app/oracle/diag/rdbms/fv12102/fv12102/trace/fv12102 ora_27060.trc
Process may have gone over pga_aggregate_limit
Just allocated 65536 bytes
Process may have gone over pga_aggregate_limit
Just allocated 65536 bytes
Process may have gone over pga_aggregate_limit
Just allocated 65536 bytes
Process may have gone over pga_aggregate_limit
Just allocated 65536 bytes
sending 4036 interrupt
Let’s run recursive_open_cursor(900) again.

Run with 33 sessions:

```sql
SYS@fv12102 AS SYSDBA> select name, value from v$pgastat
where name in ('aggregate PGA target parameter','total PGA allocated');
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregate PGA target parameter</td>
<td>524288000</td>
</tr>
<tr>
<td>total PGA allocated</td>
<td>987976704</td>
</tr>
</tbody>
</table>

PGA allocated memory grew to 987M.

PGA_AGGREGATE_LIMIT=600 ??
There’s a MOS note: 1520324.1:

- Limiting process size with database parameter PGA_AGGREGATE_LIMIT.
- “The PAL initialization parameter enables you to specify a hard limit on PGA memory usage”.
- The background process CKPT checks every three seconds whether the amount of memory exceeds...

Ah... so starting sessions 3 seconds after each other should be limited correctly!
• Let’s re-run recursive_open_cursor(900).

• Include a sleep 3.

• Run with 33 sessions again:

SYS@fv12102 AS SYSDBA> select name, value from v$pgastat
where name in ('aggregate PGA target parameter','total PGA allocated');

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregate PGA target parameter</td>
<td>524288000</td>
</tr>
<tr>
<td>total PGA allocated</td>
<td>985552896</td>
</tr>
</tbody>
</table>

• PGA allocated memory **still** grew to 987M.

• PGA_AGGREGATE_LIMIT=600!
PGA_AGGREGATE_LIMIT

• Further testing with recursive_open_cursor(900):
  • ORA-4036 kicked in at approx. 1010M.
  • $1010/600 \times 100 = 168\%$ !!
PGA_AGGREGATE_LIMIT

• Extra information
  • Bug 20992304 (non-public)
  • pga_aggregate_limit internally adjusted because user-specified value is below the minimum. However, show parameter does not reflect this. And no error returned to user.
  • Bug 19820445 (public)
  • SHOW PARAMETER PGA_AGGREGATE_LIMIT CAN NOT SHOW THE REAL VALUE
PGA limit before 12c

• Oracle introduced an event in their white paper on ‘Exadata consolidation best practices’.
  • event 10261.
  • Limit the size of the PGA heap.
  • To be able to limit PGA usage before Oracle 12c.
  • **undocumented event.**

  • Level is amount of kilobytes for the size limit.

```sql
SYS@fv12102 AS SYSDBA>
alter system set event = '10261 trace name context forever, level nnnn' scope=spfile;
```
PGA limit before 12c

• Testing event 10261:
  • Works on (at least) 11.2.0.4 - 11.2.0.1.
  • Is more hard limiting: Acted on PGA size 677M.
  • ALL processes subject to limit (including SYS).
  • *Is a process level limit, not an instance level limit!*

• Hitting limit:
  • version 11.2.0.4: ORA-10260 limit size n of the ..
  • < version 11.2.0.4: ORA-00600 [723], [alloc size], [top uga heap]
PGA Conclusion

• PGA memory is needed for every process.
  • A lot of the per process memory needed can **not** be tuned.
• Actual PGA memory used is untunable memory required times number of processes.
• Untunable memory usage includes:
  • Number of open cursors, variables, arrays, collections, PL/SQL elements.
PGA Conclusion

• PGA_AGGREGATE_TARGET tunes workarea sizing.

• Actual usage of PGA can be different from P_A_T, which means way less or way more.

• Do not use PGA_AGGREGATE_TARGET for the intention of sizing.
  • Exception might be low connection count, high sort/hash/bitmap using connections (data warehouse).
PGA Conclusion

• With 12c, there’s PGA_AGGREGATE_LIMIT.
  • Limits, but not as you would expect.
  • Kicks in rather late in my tests (168%).
  • Once limit is hit, sessions which get ORA-4036 seem quite random to me.
• The limiting acts on PGA allocation.
PGA Conclusion

• Pre 12c, event 10261 could be used
  • This event is a limit on the *per process* PGA size.
  • All sessions are subject to this limit, incl. SYS.
  • Does not prevent excess PGA allocation due to huge number of connections.
• If you need this, ask blessing from support.
• Limited usability, more last resort type solution.
Q&A

Thank you for attending!
• Thanks:
  • Jason Arniel, Tanel Poder, Martin Bach, KJ Jongsmma, Enkitec.
• Non-used slides after this point.
Hugepages

• 11.2.0.1
  • Set `vm.nr_hugepages`
  • Startup database
  • Either all is in hugepages, or not.
  • Watch `/proc/meminfo`
Hugepages

• 11.2.0.2
  • Set `vm.nr_hugepages`
  • Startup database
  • Either all is in hugepages, or not.
  • Look in `alert.log`:

Starting ORACLE instance (normal)
******************** Huge Pages Information ********************
Huge Pages memory pool detected (total: 1030 free: 1030)
Memlock limit too small: 65536 to accommodate segment size: 1050673152
Huge Pages allocation failed (free: 1030 required: 501)
Allocation will continue with default/smaller page size
**********---------------------------------*************
Hugepages

• 11.2.0.2
  • Set vm.nr_hugepages
  • Startup database
  • Either all is in hugepages, or not.
  • Look in alert.log:

Starting ORACLE instance (normal)

****************** Huge Pages Information ******************
Huge Pages memory pool detected (total: 1030 free: 1030)
DFLT Huge Pages allocation successful (allocated: 501)

***************************************************
Hugepages

- 11.2.0.3 / 11.2.0.4 / 12.1.0.1
- Set `vm.nr_hugepages`
- Startup database
- Look in alert.log:

Starting ORACLE instance (normal)
************************** Large Pages Information **************************
Total Shared Global Region in Large Pages = 1002 MB (100%)

Large Pages used by this instance: 501 (1002 MB)
Large Pages unused system wide = 529 (1058 MB) (alloc incr 4096 KB)
Large Pages configured system wide = 1030 (2060 MB)
Large Page size = 2048 KB
***********************************************************
Hugepages

• 11.2.0.3 / 11.2.0.4 / 12.1.0.1

• Mixed normal and huge pages usage:

Starting ORACLE instance (normal)

*************** Large Pages Information ***************
Total Shared Global Region in Large Pages = 600 MB (59%)
Large Pages used by this instance: 300 (600 MB)
Large Pages unused system wide = 0 (0 KB) (alloc incr 4096 KB)
Large Pages configured system wide = 300 (600 MB)
Large Page size = 2048 KB

RECOMMENDATION:
Total Shared Global Region size is 1002 MB. For optimal performance, prior to the next instance restart increase the number of unused Large Pages by atleast 201 2048 KB Large Pages (402 MB) system wide to get 100% of the Shared Global Region allocated with Large pages

******************************************************************************
Hugepages

12.1.0.2

Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
Mon Jan 12 01:39:17 2015
Per process system memlock (soft) limit = 227G
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2050M
Available system pagesizes:
4K, 2048K
Supported system pagesize(s):
PAGESIZE AVAILABLE_PAGES EXPECTED_PAGES ALLOCATED_PAGES ERROR(s)
Mon Jan 12 01:39:17 2015
4K Configured 3 3 NONE
Mon Jan 12 01:39:17 2015
2048K 1030 1025 1025 NONE
Mon Jan 12 01:39:17 2015
*******************************************************************************
Hugepages

- **12.1.0.2**

Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
Per process system memlock (soft) limit = 227G
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2050M
Available system pagesizes:
  - 4K, 2048K
Supported system pagesize(s):

<table>
<thead>
<tr>
<th>PAGESIZE</th>
<th>AVAILABLE_PAGES</th>
<th>EXPECTED_PAGES</th>
<th>ALLOCATED_PAGES</th>
<th>ERROR(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K</td>
<td>Configured</td>
<td>3</td>
<td>220472</td>
<td>NONE</td>
</tr>
<tr>
<td>2048K</td>
<td>600</td>
<td>1025</td>
<td>594</td>
<td>NONE</td>
</tr>
</tbody>
</table>

RECOMMENDATION:
1. For optimal performance, configure system with expected number
   of pages for every supported system pagesize prior to the next
   instance restart operation.

Mon Jan 12 02:08:54 2015
**********************************************************************

enkitec
v$process_memory_detail

• Populate v$process_memory_detail:
• Get PID of the session to get memory details on.

```
select pid from v$process p, v$session s where s.paddr=p.addr and s.sid = nn;
```

• Issue alter session (as sys):

```
alter session set events 'immediate trace name pga_detail_get level PID';
```

• Clean up v$process_memory_detail:

```
alter session set events 'immediate trace name pga_detail_cancel level PID';
```

• Might require execution in the session to get v
  $process_memory_detail loaded.
Memory models

• With all 3 memory models:
  • PGA is allocated in the process’ address space.
  • Private.
Virtual memory

• In order to understand memory management, a little OS theory is needed.

• Linux (and the major Unixes) use Virtual Memory.
Virtual memory

• Virtual memory means each process sees memory as a contiguous address space.
• By default, all memory is private.
• The CPU’s memory management unit (MMU) translates virtual to real memory addresses.
• Virtual memory also means the storage of memory pages could be disk instead of RAM memory.
Virtual memory

• An overview of a process’ address space is in:
  • /proc/<PID>/maps

• Let’s look at an example: myself looking in maps with the ‘cat’.
Virtual memory

```
$ cat /proc/self/maps

00400000-0040b000  r-xp  00000000 fc:00 328     /bin/cat
0060a000-0060b000  rw-p  0000a000 fc:00 328     /bin/cat
0060b000-0060c000  rw-p  00000000 00:00 0
0080a000-0080b000  rw-p  0000a000 fc:00 328     /bin/cat

01e38000-01e59000  rw-p  00000000 00:00 0     [heap]

3bb3000000-3bb3020000  r-xp  00000000 fc:00 134595 /lib64/ld-2.12.so
3bb321f000-3bb3220000  r--p  0001f000 fc:00 134595 /lib64/ld-2.12.so
3bb3220000-3bb3221000  rw-p  00020000 fc:00 134595 /lib64/ld-2.12.so
3bb3221000-3bb3222000  rw-p  00000000 00:00 0
3bb3b8e000-3bb3b8f000  rw-p  0018e000 fc:00 134689 /lib64/libc-2.12.so
3bb398a000-3bb3b8a000  --p  0018a000 fc:00 134689 /lib64/libc-2.12.so
3bb3b8f000-3bb3b94000  rw-p  00000000 00:00 0
7f71680d7000-7f71680da000  rw-p  00000000 00:00 0
7f71680e4000-7f71680e5000  rw-p  00000000 00:00 0
7fff43133000-7fff43154000  rw-p  00000000 00:00 0     [stack]
7fff431fb000-7fff431fd000  r-xp  00000000 00:00 0     [vdso]
ffffffffffffff6000000-ffffffffffffff601000  r-xp  00000000 00:00 0     [vsyscall]
```
Virtual memory

• Modern OS’es make very smart use of executables and libraries.

• Only one copy is used whenever possible.
• Usage is Copy On Write (COW).
Virtual memory

• How much memory does my process use?

• cat /proc/self/maps

• The correct answer is:
Virtual memory

Not enough information is provided at the process level to make an exact calculation.
Virtual memory

• As you might have guessed, all the smartness of sharing means true usage gets a bit unclear.
  • ...And we didn’t even look at Oracle yet (!!)

• To understand what I mean, look at:
  • /proc/<PID>/status
Virtual memory

$ cat /proc/self/status

...  

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VmPeak:</td>
<td>100988 kB</td>
</tr>
<tr>
<td>VmSize:</td>
<td>100988 kB</td>
</tr>
<tr>
<td>VmLck:</td>
<td>0 kB</td>
</tr>
<tr>
<td>VmPin:</td>
<td>0 kB</td>
</tr>
<tr>
<td>VmHWM:</td>
<td>484 kB</td>
</tr>
<tr>
<td>VmRSS:</td>
<td>484 kB</td>
</tr>
<tr>
<td>VmData:</td>
<td>184 kB</td>
</tr>
<tr>
<td>VmStk:</td>
<td>136 kB</td>
</tr>
<tr>
<td>VmExe:</td>
<td>44 kB</td>
</tr>
<tr>
<td>VmLib:</td>
<td>1704 kB</td>
</tr>
<tr>
<td>VmPTE:</td>
<td>52 kB</td>
</tr>
<tr>
<td>VmSwap:</td>
<td>0 kB</td>
</tr>
<tr>
<td>Threads:</td>
<td>1</td>
</tr>
</tbody>
</table>

VmSize: Total allocated and mmapped memory. “total visible in use memory”. Both shared and non-shared.

VmRSS: Resident Set Size: All memory truly used (“touched”). Both shared and non-shared.
Automatic memory management

- AMM can’t use hugepages.

- This disqualifies the usage of AMM for anything serious.

- Don’t use AMM.
  - Unless you got a valid reason.
Hugepages

• O/S facility
  – Memory managed per 2M instead of 4kB (page).
  – Hugepages are non-swappable.
  – Significant reduction in CPU usage.
  – Potentially severe reduction in pagetable memory allocations.

• If you are not using hugepages and are doing anything serious, you are doing it wrong!
Hugepages

• Only SGA can be stored in hugepages.

• Hugepages are reserved at startup.
  – /proc/sys/vm/nr_hugepages - vm.nr_hugepages
  – Can be modified after startup.
  – Only non-fragmented memory can be reassigned to be used as huge pages.
  – You can do severe damage by allocating too much.
Hugepages

• The database side
  – Parameter:
    • use_large_pages (>= 11.2.0.2)
      » TRUE (default)
      » ONLY
      » FALSE
Hugepages

• 11.2.0.1
  • Set vm.nr_hugepages
  • Startup database
  • Either all is in hugepages, or not.
  • Watch /proc/meminfo
Hugepages

• 11.2.0.2
  • Set vm.nr_hugepages
  • Startup database
  • Either all is in hugepages, or not.
  • Look in alert.log:

Starting ORACLE instance (normal)
********************************* Huge Pages Information *************************
Huge Pages memory pool detected (total: 1030 free: 1030)
Memlock limit too small: 65536 to accommodate segment size: 1050673152
Huge Pages allocation failed (free: 1030 required: 501)
Allocation will continue with default/smaller page size
******************************************************************************
Hugepages

• 11.2.0.2
  • Set vm.nr_hugepages
  • Startup database
  • Either all is in hugepages, or not.
  • Look in alert.log:

    Starting ORACLE instance (normal)
    ************************** Huge Pages Information **************************
    Huge Pages memory pool detected (total: 1030 free: 1030)
    DFLT Huge Pages allocation successful (allocated: 501)
    ********************************************
Hugepages

• 11.2.0.3 / 11.2.0.4 / 12.1.0.1
• Set vm.nr_hugepages
• Startup database
• Look in alert.log:

Starting ORACLE instance (normal)
****************** Large Pages Information ******************
Total Shared Global Region in Large Pages = 1002 MB (100%)

Large Pages used by this instance: 501 (1002 MB)
Large Pages unused system wide = 529 (1058 MB) (alloc incr 4096 KB)
Large Pages configured system wide = 1030 (2060 MB)
Large Page size = 2048 KB
************************************************************************
Hugepages

• 11.2.0.3 / 11.2.0.4 / 12.1.0.1

• Mixed normal and huge pages usage:

Starting ORACLE instance (normal)

****************** Large Pages Information ******************
Total Shared Global Region in Large Pages = 600 MB (59%)
Large Pages used by this instance: 300 (600 MB)
Large Pages unused system wide = 0 (0 KB) (alloc incr 4096 KB)
Large Pages configured system wide = 300 (600 MB)
Large Page size = 2048 KB

RECOMMENDATION:
Total Shared Global Region size is 1002 MB. For optimal performance,
prior to the next instance restart increase the number
of unused Large Pages by atleast 201 2048 KB Large Pages (402 MB)
system wide to get 100% of the Shared
Global Region allocated with Large pages

*******************************************************************************
Hugepages

• 12.1.0.2

Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
Mon Jan 12 01:39:17 2015
Per process system memlock (soft) limit = 227G
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2050M
Available system pagesizes:
  4K, 2048K
Supported system pagesize(s):

<table>
<thead>
<tr>
<th>PAGESIZE</th>
<th>AVAILABLE_PAGES</th>
<th>EXPECTED_PAGES</th>
<th>ALLOCATED_PAGES</th>
<th>ERROR(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K</td>
<td>Configured</td>
<td>3</td>
<td>3</td>
<td>NONE</td>
</tr>
<tr>
<td>2048K</td>
<td>1030</td>
<td>1025</td>
<td>1025</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**********************************************************************
**Hugepages**

- **12.1.0.2**

Dump of system resources acquired for SHARED GLOBAL AREA (SGA)

Per process system memlock (soft) limit = 227G

Expected per process system memlock (soft) limit to lock

  SHARED GLOBAL AREA (SGA) into memory: 2050M

Available system pagesizes:

  4K, 2048K

Supported system pagesize(s):

<table>
<thead>
<tr>
<th>PAGESIZE</th>
<th>AVAILABLE_PAGES</th>
<th>EXPECTED_PAGES</th>
<th>ALLOCATED_PAGES</th>
<th>ERROR(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K</td>
<td>Configured</td>
<td>3</td>
<td>220472</td>
<td>NONE</td>
</tr>
<tr>
<td>2048K</td>
<td>600</td>
<td>1025</td>
<td>594</td>
<td>NONE</td>
</tr>
</tbody>
</table>

RECOMMENDATION:

1. For optimal performance, configure system with expected number
   of pages for every supported system pagesize prior to the next
   instance restart operation.

Mon Jan 12 02:08:54 2015

******************************************************************************
Memory model

• That leaves manual and ASMM as choices.

  – Some DBAs still like to set all memory areas.
  – ASMM favoured growing the shared pool.
    • Mainly seen in version 10.2.
    • Lack of bind variables.
    • Evaporating the buffer cache.
    • Recent 11.2 versions seem to be more sensible.
Memory model

• Even without ASMM Oracle can and will resize.
  – Since 11.2.0.2.
  – See bug: 13340694.
  – Expected behaviour.

• Can be disabled by an undocumented parameter.
  – _memory_imm_mode_without_autosga
About resizes.

• If you use an SPFILE
  – Oracle records the new value of a resize:
    • `<instancename>.__db_cache_size=<nr>`
    • Double underscore.

• So once an instance resizes, it potentially doesn’t need to go through the resize to again to reach the current (assumed ideal) size.
SGA allocation

• Aside from AMM:
  – SGA memory allocation is done at startup.
  – SGA memory allocation is static.
  – *Always* allocated as System V shared memory

• With AMM:
  – A few tiny System V shared memory segments are allocated
  – Real allocations are done in granules in /dev/shm
sysresv

• Utility provided with Oracle (8.1.5+)
  – Function:
    • Show resources used.
    • Frees system V shared resources when the instance is dead.
sysresv

- shows (running sysresv without any arguments):
  - Kernel settings for shmmmax, shmall, shmmni
  - Output of ‘ipcs -a’
  - Output of ‘ulimit -a’
  - /dev/shm (tmpfs - AMM!) usage

- Shared memory segments of $ORACLE_SID
- Semaphore sets of $ORACLE_SID
[oracle@bigmachine v12102 bin]$ sysresv

IPC Resources for ORACLE_SID "v12102" :
Maximum shared memory segment size (shmmax): 4398046511104 bytes
Total system shared memory (shmall): 17592186044416 bytes
Total system shared memory count (shmmni): 4096
*************************** Dumping ipcs output ***************************

------- Shared Memory Segments -------

<table>
<thead>
<tr>
<th>key</th>
<th>shmid</th>
<th>owner</th>
<th>perms</th>
<th>bytes</th>
<th>nattch</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000000000</td>
<td>104464384</td>
<td>oracle</td>
<td>640</td>
<td>4096</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0000000000</td>
<td>104497153</td>
<td>oracle</td>
<td>640</td>
<td>4096</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0xb92c57c8</td>
<td>104529922</td>
<td>oracle</td>
<td>640</td>
<td>24576</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>0x0000000000</td>
<td>292159491</td>
<td>oracle</td>
<td>640</td>
<td>4194304</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>0x0000000000</td>
<td>292192260</td>
<td>oracle</td>
<td>640</td>
<td>1241513984</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>0x0000000000</td>
<td>292225029</td>
<td>oracle</td>
<td>640</td>
<td>754974720</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>0x0000000000</td>
<td>292257798</td>
<td>oracle</td>
<td>640</td>
<td>148066304</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>0x240b65ec</td>
<td>292290567</td>
<td>oracle</td>
<td>640</td>
<td>12288</td>
<td>69</td>
<td></td>
</tr>
</tbody>
</table>
sysresv

------- Semaphore Arrays -------

key        semid      owner      perms      nsems
0x77aeec14 229378     oracle     640        142
0x77aeec15 262147     oracle     640        142
0x77aeec16 294916     oracle     640        142
0x017b884c 1605637    oracle     640        104

------- Message Queues -------

key        msqid      owner      perms      used-bytes      messages

************** End of ipcs command dump **************
<table>
<thead>
<tr>
<th>Resource</th>
<th>Limit 1</th>
<th>Limit 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>core file size</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
<tr>
<td>data seg size</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
<tr>
<td>scheduling priority</td>
<td>0 KB/0 KB</td>
<td></td>
</tr>
<tr>
<td>file size</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
<tr>
<td>pending signals</td>
<td>31 KB/31 KB</td>
<td></td>
</tr>
<tr>
<td>max locked memory</td>
<td>227 GB/227 GB</td>
<td></td>
</tr>
<tr>
<td>max memory size</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
<tr>
<td>open files</td>
<td>64 KB/64 KB</td>
<td></td>
</tr>
<tr>
<td>POSIX message queues</td>
<td>800 KB/800 KB</td>
<td></td>
</tr>
<tr>
<td>real-time priority</td>
<td>0 KB/0 KB</td>
<td></td>
</tr>
<tr>
<td>stack size</td>
<td>32 MB/32 MB</td>
<td></td>
</tr>
<tr>
<td>cpu time</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
<tr>
<td>max user processes</td>
<td>16 KB/16 KB</td>
<td></td>
</tr>
<tr>
<td>virtual memory</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
<tr>
<td>file locks</td>
<td>UNLIMITED/UNLIMITED</td>
<td></td>
</tr>
</tbody>
</table>

******************* End of Resource Limits Dump *******************

Total /dev/shm size: 914595840 bytes, used: 658268160 bytes
### sysresv

**Shared Memory:**

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>292192260</td>
<td>0x00000000</td>
</tr>
<tr>
<td>292225029</td>
<td>0x00000000</td>
</tr>
<tr>
<td>292257798</td>
<td>0x00000000</td>
</tr>
<tr>
<td>292159491</td>
<td>0x00000000</td>
</tr>
<tr>
<td>292290567</td>
<td>0x240b65ec</td>
</tr>
</tbody>
</table>

**Semaphores:**

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1605637</td>
<td>0x017b884c</td>
</tr>
</tbody>
</table>

Oracle Instance alive for sid "v12102"
### Shared Memory:

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>292192260</td>
<td>0x000000000</td>
</tr>
<tr>
<td>292225029</td>
<td>0x000000000</td>
</tr>
<tr>
<td>292257798</td>
<td>0x000000000</td>
</tr>
<tr>
<td>292159491</td>
<td>0x000000000</td>
</tr>
<tr>
<td>292290567</td>
<td>0x240b65ec</td>
</tr>
</tbody>
</table>

### Semaphores:

<table>
<thead>
<tr>
<th>ID</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1605637</td>
<td>0x017b884c</td>
</tr>
</tbody>
</table>

Oracle Instance alive for sid "v12102"

### Shared Memory Segments

<table>
<thead>
<tr>
<th>key</th>
<th>shmid</th>
<th>owner</th>
<th>perms</th>
<th>bytes</th>
<th>nattch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>104464384</td>
<td>oracle</td>
<td>640</td>
<td>4096</td>
<td>0</td>
</tr>
<tr>
<td>0x00000000</td>
<td>104497153</td>
<td>oracle</td>
<td>640</td>
<td>4096</td>
<td>0</td>
</tr>
<tr>
<td>0xb92c57c8</td>
<td>104529922</td>
<td>oracle</td>
<td>640</td>
<td>24576</td>
<td>0</td>
</tr>
<tr>
<td>0x00000000</td>
<td>292159491</td>
<td>oracle</td>
<td>640</td>
<td>4194304</td>
<td>22</td>
</tr>
<tr>
<td>0x00000000</td>
<td>292192260</td>
<td>oracle</td>
<td>640</td>
<td>1241513984</td>
<td>68</td>
</tr>
<tr>
<td>0x00000000</td>
<td>292225029</td>
<td>oracle</td>
<td>640</td>
<td>754974720</td>
<td>68</td>
</tr>
<tr>
<td>0x00000000</td>
<td>292257798</td>
<td>oracle</td>
<td>640</td>
<td>148066304</td>
<td>68</td>
</tr>
<tr>
<td>0x240b65ec</td>
<td>292290567</td>
<td>oracle</td>
<td>640</td>
<td>12288</td>
<td>68</td>
</tr>
</tbody>
</table>

### Semaphore Arrays

<table>
<thead>
<tr>
<th>key</th>
<th>semid</th>
<th>owner</th>
<th>perms</th>
<th>nsems</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x77aeec14</td>
<td>229378</td>
<td>oracle</td>
<td>640</td>
<td>142</td>
</tr>
<tr>
<td>0x77aeec15</td>
<td>262147</td>
<td>oracle</td>
<td>640</td>
<td>142</td>
</tr>
<tr>
<td>0x77aeec16</td>
<td>294916</td>
<td>oracle</td>
<td>640</td>
<td>142</td>
</tr>
<tr>
<td>0x017b884c</td>
<td>1605637</td>
<td>oracle</td>
<td>640</td>
<td>104</td>
</tr>
</tbody>
</table>
SGA summary

• Database SGA memory:
  – Should be configured as ASMM or manual mem.
  – Is static in allocated o/s space.
  – Should be “backed” by huge pages.

• This makes it easy for o/s memory planning.
PGA - O/S memory

• Modern platforms use virtual memory.
  – Virtual means every process has it’s own *private* address space.
  – Process memory is not visible to other processes.
PGA - O/S memory

• Every process needs memory.
  – It needs to store the executable executing.
  – It needs to store/access (shared) libraries.
  – It needs a heap (allocated memory, variables).
  – It needs a stack (save c function state).
## PGA - O/S memory

```bash
$ cat /proc/3257/maps
```

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Permissions</th>
<th>Offset</th>
<th>Size</th>
<th>File Offset</th>
<th>File Size</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>00400000-004d4000</td>
<td>r-xp</td>
<td>00000000</td>
<td>156</td>
<td></td>
<td></td>
<td>/bin/bash</td>
</tr>
<tr>
<td>006d4000-006dd000</td>
<td>rw-p</td>
<td>000d4000</td>
<td>156</td>
<td></td>
<td></td>
<td>/bin/bash</td>
</tr>
<tr>
<td>006dd000-006e3000</td>
<td>rw-p</td>
<td>00000000</td>
<td>0:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008dc000-008e5000</td>
<td>rw-p</td>
<td>000dc000</td>
<td>156</td>
<td></td>
<td></td>
<td>/bin/bash</td>
</tr>
<tr>
<td>00bd3000-00c36000</td>
<td>rw-p</td>
<td>00000000</td>
<td>0:00</td>
<td></td>
<td></td>
<td>[heap]</td>
</tr>
<tr>
<td>3bb321000-3bb322000</td>
<td>r-xp</td>
<td>00000000</td>
<td>134595</td>
<td></td>
<td></td>
<td>/lib64/ld-2.12.so</td>
</tr>
<tr>
<td>3bb322000-3bb322100</td>
<td>rw-p</td>
<td>00020000</td>
<td>134595</td>
<td></td>
<td></td>
<td>/lib64/ld-2.12.so</td>
</tr>
<tr>
<td>3bb3221000-3bb322200</td>
<td>rw-p</td>
<td>00000000</td>
<td>0:00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3bb3400000-3bb340200</td>
<td>r-xp</td>
<td>00000000</td>
<td>136676</td>
<td></td>
<td></td>
<td>/lib64/libdl-2.12.so</td>
</tr>
<tr>
<td>3bb3402000-3bb360200</td>
<td>---p</td>
<td>00020000</td>
<td>136676</td>
<td></td>
<td></td>
<td>/lib64/libdl-2.12.so</td>
</tr>
<tr>
<td>3bb3602000-3bb360300</td>
<td>---p</td>
<td>00020000</td>
<td>136676</td>
<td></td>
<td></td>
<td>/lib64/libdl-2.12.so</td>
</tr>
<tr>
<td>3bb3603000-3bb360400</td>
<td>rw-p</td>
<td>00030000</td>
<td>136676</td>
<td></td>
<td></td>
<td>/lib64/libdl-2.12.so</td>
</tr>
<tr>
<td>3bb3800000-3bb398a00</td>
<td>r-xp</td>
<td>00000000</td>
<td>134689</td>
<td></td>
<td></td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>3bb398a000-3bb3b8a00</td>
<td>---p</td>
<td>0018a000</td>
<td>134689</td>
<td></td>
<td></td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>3bb3b8a000-3bb3b8e00</td>
<td>---p</td>
<td>0018a000</td>
<td>134689</td>
<td></td>
<td></td>
<td>/lib64/libc-2.12.so</td>
</tr>
<tr>
<td>3bb3b8e000-3bb3b8f00</td>
<td>rw-p</td>
<td>0018e000</td>
<td>134689</td>
<td></td>
<td></td>
<td>/lib64/libc-2.12.so</td>
</tr>
</tbody>
</table>

**Executable**

**Anonymous memory**

**Heap**

**Shared Library**
PGA - O/S memory

...
PGA - O/S memory

- Accounting per process memory in Linux is hard.
  - A lot of memory is mapped (mmap()) in to process memory.
  - Some of it (executable/shared libraries) is COW.
    - Memory only allocated on change or first to use.

- Accounting per process memory in Linux is hard.
PGA - O/S memory

• You might ask yourself:

  – Why do I care?
## PGA - Oracle foreground

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Mode</th>
<th>Access</th>
<th>Offset</th>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>00400000-1093f000</td>
<td>r-xp</td>
<td>00000000</td>
<td>fc:02 272770084</td>
<td>/u01/.../bin/oracle</td>
</tr>
<tr>
<td>10b3e000-10dbf000</td>
<td>rw-p</td>
<td>1053e000</td>
<td>fc:02 272770084</td>
<td>/u01/.../bin/oracle</td>
</tr>
<tr>
<td>10dbf000-10df0000</td>
<td>rw-p</td>
<td>00000000</td>
<td>00:00 0</td>
<td></td>
</tr>
<tr>
<td>117b3000-1181a000</td>
<td>rw-p</td>
<td>00000000</td>
<td>00:00 0</td>
<td></td>
</tr>
</tbody>
</table>

### Executable
- **/u01/.../bin/oracle**
- **/lib64/libtinfo.so.5.7**
- **/lib64/libnss_files-2.12.so**

### SGA
- **/SYSV00000000 (deleted)**
- **/SYSV00000000 (deleted)**
- **/SYSV00000000 (deleted)**
- **/SYSV240b65ec (deleted)**

- **3bb3b8f000-3bb3b94000**
  - **rw-p**
  - **00000000**
  - **00:00 0**
  - **/lib64/libtinfo.so.5.7**
- **3bb580000-3bb581d000**
  - **r-xp**
  - **00000000**
  - **fc:00 136285**
  - **/lib64/libtinfo.so.5.7**
- **3bb581d000-3bb5a1d000**
  - **---p**
  - **0001d000**
  - **fc:00 136285**
  - **/lib64/libnss_files-2.12.so**
- **3bb5a1d000-3bb5a21000**
  - **rw-p**
  - **0001d000**
  - **fc:00 136285**
  - **/lib64/libtinfo.so.5.7**
- **7fefb3d76000-7fefb9c07000**
  - **r--p**
  - **00000000**
  - **fc:00 131106**
  - **/usr/lib/locale/locale-archive**
- **7fefb9c07000-7fefb9c13000**
  - **r-xp**
  - **00000000**
  - **fc:00 131129**
  - **/lib64/libnss_files-2.12.so**
- **7fefb9c13000-7fefb9e13000**
  - **---p**
  - **0000c000**
  - **fc:00 131129**
  - **/lib64/libnss_files-2.12.so**
- **7fefb9e13000-7fefb9e14000**
  - **r--p**
  - **0000c000**
  - **fc:00 131129**
  - **/lib64/libnss_files-2.12.so**
- **7fefb9e14000-7fefb9e15000**
  - **rw-p**
  - **0000d000**
  - **fc:00 131129**
  - **/lib64/libnss_files-2.12.so**
- **7fefb9e15000-7fefb9e18000**
  - **rw-p**
  - **00000000**
  - **00:00 0**