

## [A stroll through shared pool heaps](#)

Last week, we were discussing about increasing `shared_pool_reserved_size` to combat a performance issue(bug) in a conference call. I thought, it was a common knowledge that `shared_pool` reserved area is part of a `shared_pool` and surprisingly it is not-so-common.

In this blog, we will discuss about `shared_pool` and `shared_pool` reserved area internals. First, we will discuss about details specific to release 9i and then discuss changes in later releases 10g/11g.

### **oradebug command**

We will use `oradebug` command to dump the heap with level 2. Level 2 is to dump `shared_pool` heap in to a trace file.

```
oradebug setmypid
oradebug dump heapdump 2
```

Above command generates a trace file and we will walk through the trace file and review various areas closely.

### **Parameters**

In this test instance, we have a bigger SGA. `Shared_pool` (6GB) and `shared_pool_reserved_size` values are printed below.

```
SQL> show parameter shared_pool
shared_pool_reserved_size          big integer 629145600
shared_pool_size                   big integer 6442450944
```

### **Trace file analysis**

In version 9i, `Shared_pool` is another sub-heap of SGA and it is further split in to multiple sub heaps. For this SGA, `shared pool` heap is split in to 6 sub heaps and this sub heap count is determined by the `shared_pool_size` and can be directly controlled by undocumented parameter `_kghdsidx_count`. Following picture illustrates these 6 sub `shared pool` heaps.

```
-----
| sga heap(1,0) |
-----
| sga heap(2,0) |
-----
| sga heap(3,0) |
-----
| sga heap(4,0) |
-----
| sga heap(5,0) |
```

```
-----  
| sga heap(6,0) |  
-----
```

Trace lines for sga heap (1,0) header information printed below.

```
*****  
HEAP DUMP heap name="sga heap(1,0)" desc=38002df10  
  extent sz=0xfe0 alt=200 het=32767 rec=9 flg=-126 opc=0  
  parent=0 owner=0 nex=0 xsz=0x1000000
```

Each of these sub heaps are split in to extents. Notice the field xsz=0x1000000 above and it is the size of an extent. Converting xsz=0x1000000 to decimal results in an extent size of 16MB. There are 68 such extents in this sub heap.

This sub-heap has 68 extents.  
sga heap (1,0):

```
-----  
| EXTENT 0 addr=5a9000000 |  
-----  
| EXTENT 1 addr=5af000000 |  
-----  
| EXTENT 2 addr=5b5000000 |  
-----  
.....  
-----  
| EXTENT 65 addr=72f000000 |  
-----  
| EXTENT 66 addr=735000000 |  
-----  
| EXTENT 67 addr=73b000000 |  
-----
```

Each extent size is 16MB( I think, this is same as granule size, but I need more test cases to confirm this). So, there are 68 extents of size 16MB in each sub-heap with a total size of 1088MB. Six such sub-heaps adds up to 6528MB, which is closer to our shared\_pool size.

If you haven't noticed already, there is a difference of 96MB between two consecutive extent boundary addresses above. That's because, these extents are striped in memory across the sub-heaps. Meaning, 1st 16MB memory chunk is EXTENT 0 of sub heap 1, next 16MB memory chunk is EXTENT 0 of sub heap 2 etc ( This is not true in 10g+ versions though)

```
sga heap (1,0):          sga heap (2,0):          sga heap (3,0):  
-----  
| EXTENT 0 addr=5a9000000 | | EXTENT 0 addr=5a8000000 | | EXTENT 0 addr=5a7000000 |  
-----  
| EXTENT 1 addr=5af000000 | | EXTENT 1 addr=5ae000000 | | EXTENT 1 addr=5ad000000 |  
-----  
| EXTENT 2 addr=5b5000000 | | EXTENT 2 addr=5b4000000 | | EXTENT 2 addr=5be000000 |  
-----  
.....  
          ...  
          ...
```

```

-----
|EXTENT 65 addr=72f000000 |
-----
|EXTENT 66 addr=735000000 |
-----
|EXTENT 67 addr=73b000000 |
-----

```

## Extent chunks

Let's review anatomy of each extent. An extent is made up of many memory chunks. Each chunk has a comment and when a chunk is requested to be allocated from shared\_pool, calling code passes a parameter with a comment and that comment is populated in ksmchcom column.

```

...
EXTENT 0 addr=5a9000000
  Chunk      5a9000040 sz=      40 R-freeable "reserved stoppe"
  Chunk      5a9000068 sz= 1515376 R-free      "              "
  Chunk      5a9171fd8 sz=      40 R-freeable "reserved stoppe"
  Chunk      5a9172000 sz= 15261576   perm      "perm          "
alo=15261576
  Chunk      5a9ffff88 sz=      64   recreate "fixed allocatio"
latch=5b43fa718
  Chunk      5a9ffffc8 sz=      56   recreate "fixed allocatio"
latch=5b43fa628
EXTENT 1 addr=5af000000
  Chunk      5af000040 sz=      40 R-freeable "reserved stoppe"
  Chunk      5af000068 sz= 1515376 R-free      "              "
  Chunk      5af171fd8 sz=      40 R-freeable "reserved stoppe"
  Chunk      5af172000 sz= 15260592   perm      "perm          "
alo=15260592
  Chunk      5affffbb0 sz=      48   free      "              "
...

```

Notice the memory chunk of 1515376 bytes between two 'reserved stopper' and that is allocated for shared pool reserved area. Each extent has one such area allocated for reserved pool. Remaining memory in each extent is associated with shared pool general area.

Of course, reserved area is considered only if there is not enough free memory in general area and the chunk size exceeds \_shared\_pool\_reserved\_min\_alloc parameter.

```

EXTENT 0:
-----
|sz=40 "reserved stopper" |
-----
|sz=1515376 R-free       | <-- reserved pool
-----
|sz=40 "reserved stopper" |
-----
|sz=15261576 perm       | <---General area starts from here -----
-----
|sz=64 "fixed allocation" |

```

```
-----  
|sz=56 "fixed allocation" |  
-----
```

## x\$ksmspr

Of course, much of this details from the trace files are visible in few x\$tables. For example, reserved pool is externalized as x\$ksmspr. Even 'reserved stopper' columns are visible in x\$ksmspr.

```
1* select * from x$ksmspr where rownum < 7
```

ADDR	INDX	INST_ID	KSMCHCOM	KSMCHPTR
KSMCHSIZ KSMCHCLS	KSMCHTYP	KSMCHPAR		
FFFFFFFF7C9322E8	0	1	reserved stoppe	0000000736171FD8
40 R-freea	0 00			
FFFFFFFF7C932290	1	1	free memory	0000000736000068
1515376 R-free	0 00			
FFFFFFFF7C932238	2	1	reserved stoppe	0000000736000040
40 R-freea	0 00			
FFFFFFFF7C9321E0	3	1	reserved stoppe	0000000730171FD8
40 R-freea	0 00			
FFFFFFFF7C932188	4	1	free memory	0000000730000068
1515376 R-free	0 00			
FFFFFFFF7C932130	5	1	reserved stoppe	0000000730000040
40 R-freea	0 00			
...				

There are 1515376 bytes in each extent with 40 bytes stopper in each side of that chunk, 68 extents in each sub-heap and 6 such sub-heaps, with a total of ( 1515456 X 68 X 6) 618,306,048 bytes which matches with x\$ksmspr.

```
select sum(ksmchsiz) from x$ksmspr;
```

```
SUM(KSMCHSIZ)  
-----  
618,306,048
```

## x\$ksmss

Each of these sub-heaps (aka sub shared pool) are also externalized as x\$ksmss. There are 68 X 16MB in each sub-heap, in this SGA, adding up to 1140850688 bytes.

```
1* select ksmdsidx, sum(ksmsslen) from x$ksmss group by ksmdsidx  
SQL> /
```

KSMDSIDX	SUM(KSMSSLEN)
1	1140850688
2	1140850688
3	1140850688
4	1140850688
5	1140850688

6 rows selected.

## Freelists

In release 9i, each sub heap of shared pool has one free list for general area and one free list for reserved area.

Free lists in general area is organized by its size and a range of size is associated with a bucket. For example, bucket 252 below covers chunk sizes ranging from 16408 to 32791 bytes. Of course, this improves performance as just one bucket can be accessed to find a free chunk closer to requested size. Chunks can be broken, recreatable chunks can be flushed etc to allocate a chunk.

```
(General area)
FREE LISTS:
Bucket 0 size=32
Bucket 1 size=40
  Chunk      6518cb670 sz=      40    free    "          "
  Chunk      6518ee428 sz=      40    free    "          "
  Chunk      6518fe9d0 sz=      40    free    "          "
...
Bucket 252 size=16408
  Chunk      65187a340 sz=    27992    free    "          "
Bucket 253 size=32792
Bucket 254 size=65560
  Chunk      651172000 sz=   7345640    free    "          "
  Chunk      645316db0 sz=    89360    free    "          "
```

But freelists for reserved area (in 9i and below), chunks are not organized by size.

```
RESERVED FREE LIST:
  Chunk      5a9000068 sz=   1515376  R-free  "          "
  Chunk      5af000068 sz=   1515376  R-free  "          "
  Chunk      5b5000068 sz=   1515376  R-free  "          "
...
```

In 9i, general area is searched for a chunk big enough to satisfy requests. If it isn't possible to get a chunk either by breaking bigger chunk, then reserved pool is checked if the request is bigger than `_shared_pool_min_alloc` parameter. Then only LRU lists checked to flush chunks. So, this can result in holding `shared_pool` latches longer.

## 10g and above

We have shown that how reserved area is a part of shared pool size. In 10g and above, few things in this blog entry have changed (improved). We will try to explore these few of those changes:

1. In 10g+, these shared pool sub-heaps are further divided in to sub-heaps. Also, not all memory is allocated at startup to these sub heap areas. Sub heap 0 is holding all unallocated memory and released to other sub heaps as memory pressure in the shared\_pool increases. This is a very good idea since ORA-4031 is thrown even if free memory is depleted from one sub-heap and other sub-heaps still have ample amount of free memory (in 9i). This issue can be reduced, by holding big chunk of free memory in heap 0 and re-distributing to other sub-heaps as memory pressure increases. Notice that shared pool sub heap 1 is divided in to further sub heaps (1,0), (1,1) and (1,2) below. Sub heap 0 is not visible in the trace file, but x\$ksmss gives it away.

```
HEAP DUMP heap name="sga heap" desc=380000030
HEAP DUMP heap name="sga heap(1,0)" desc=380045968
HEAP DUMP heap name="sga heap(1,1)" desc=3800471c0
HEAP DUMP heap name="sga heap(1,2)" desc=380048a18
HEAP DUMP heap name="sga heap(1,3)" desc=38004a270
HEAP DUMP heap name="sga heap(2,0)" desc=38004f190
HEAP DUMP heap name="sga heap(2,1)" desc=3800509e8
...
```

```
SQL> select ksmdsidx, sum(ksmsslen) from x$ksmss group by ksmdsidx;
```

KSMDSIDX	SUM(KSMSSLEN)
0	4831838208
1	486554880
2	620757584
3	486539576
4	402653840
5	402653536
6	419430608
7	436207888

```
SQL> select ksmssnam, sum(ksmsslen) from x$ksmss where ksmdsidx=0
group by ksmssnam order by 2
```

ksmssnam	SUM(KSMSSLEN)
free memory	4831838208
...	

2. In 10g+, reserved free list is also organized by buckets and each bucket covering a range of chunk size. In 9i, this concept was only applicable to general area, not reserved pool area.

```
RESERVED FREE LISTS:
```

```
Reserved bucket 0 size=32
Reserved bucket 1 size=4100
Reserved bucket 2 size=4104
...
Reserved bucket 15 size=65560
Chunk          7c4000088 sz= 4529992 R-free      "          "
```

3. In 9i, we saw that all sub-shared\_pool heaps were of same size. But, in 10g, due to dynamic redistribution of memory from sub-heap 0 to other heaps, heap sizes can vary wildly.

4. In 9i, we saw that extents are striped across multiple sub-heaps. But, in 10g, that isn't true anymore, due to dynamic redistribution of memory.

5. If sga\_target and memory\_target parameters are in use, this gets more complicated. Part of shared\_pool itself can be reallocated and tagged as 'KGG:NO ACCESS' comment, which means that part of that shared pool is allocated to buffer cache (ASMM or AMM). In this case, shared pool objects can be flushed, extents deallocated from sub heaps and tagged as KGG: NO ACCESS. Of course, excessive redistribution of memory between buffer\_cache and shared\_pool can have dramatic effect on performance. This can be mitigated by having minimum values for various areas.

In summary, reserved area is just part of shared\_pool and there has been many improvements in this area. Knowledge about shared pool internals will be very useful, especially to understand and resolve performance issues.