ORA-4031 and Shared Pool Duration

Filed under: Oracle database internals, Performance tuning, RAC — orainternals @ 4:18 pm Edit This

After reading my earlier post on shared pool A stroll through shared pool heap , one of my client contacted me with an interesting ORA-4031 issue. Client was getting ORA-4031 errors and shared pool size was over 4GB (in a RAC environment). Client DBA queried v$sgastat to show that there is plenty of free memory in the shared pool. We researched the issue and it is worth blogging. Client DBA was confused as to how there can be ORA-4031 errors when the shared pool free memory is few GBs.

Heapdump Analysis

At this point, it is imperative to take heapdump in level 2 and Level 2 is for the shared pool heap dump. [Please be warned that it is not advisable to take shared pool heap dumps excessively, as that itself can cause performance issue. During an offline conversation, Tanel Poder said that heapdump can freeze instance as his clients have experienced.]. This will create a trace file in user_dump_dest destination and that trace file is quite useful in analyzing the contents of shared pool heap. Tanel Poder has an excellent script heapdump_analyzer . I modified that script adding code for aggregation at heap, extent and type levels to debug this issue further and it is available as heapdump_dissect.ksh . (with a special permission from Tanel to publish this script.)

Shared pool review

You can read much more about shared pool in my earlier blog entry posted above. Just as a cursory review, shared pool is split in to multiple sub heaps. In 10g, each of those sub heaps are divided in to even smaller sub heaps, let's call it mini-heaps. For example, in this specific database, there are three sub heaps. Each of those sub heaps are further split in to four mini-heaps (1,0), (1,1), (1,2) and (1,3) each.

```
sga heap(1,0)
sga heap(1,1)
sga heap(1,2)
sga heap(1,3)

sga heap(2,0)
sga heap(2,1)
sga heap(2,2)
sga heap(2,3)

sga heap(3,0)
sga heap(3,1)
sga heap(3,2)
sga heap(3,3)
```

One or more extents are allocated to these mini-heaps dynamically as these areas grow over the course of instance life until there is no more free memory to allocate. In this blog

entry, let's focus on the contents of these sub-heaps.
Following shows output of heapdump_dissect script at extent level aggregation. As you can see sga heap (1,0) is allocated with 34 extents of 16M each. [16M comes from _ksmg_granule_size.]

```
Extent level summary
------------------
Sub heap          Extent             Type              Allocation cmt   Size
---------------- -----------------   ---------------   --------------  --------
   sga heap(1,0)          EXTENT 0              free                    1792
   sga heap(1,0)          EXTENT 0              perm   perm             15096064
   sga heap(1,0)          EXTENT 0           R-free                     616
   sga heap(1,0)          EXTENT 0           R-perm   perm             1678560
   sga heap(1,0)          EXTENT 0        R-freeable   reserved stoppe 96
   sga heap(1,0)          EXTENT 1              free                    808
   sga heap(1,0)          EXTENT 1              perm   perm             15097048
   sga heap(1,0)          EXTENT 1           R-free                     1248
   sga heap(1,0)          EXTENT 1           R-perm   perm             1677928
   sga heap(1,0)          EXTENT 1        R-freeable   reserved stoppe 96
   sga heap(1,0)          EXTENT 2              free                    456
   sga heap(1,0)          EXTENT 2              perm   perm             15097400
   sga heap(1,0)          EXTENT 2           R-free                     1679176
   sga heap(1,0)          EXTENT 2        R-freeable   reserved stoppe 96
 ...
   sga heap(1,0)         EXTENT 34              free                    160
   sga heap(1,0)         EXTENT 34              perm   perm             15097696
   sga heap(1,0)         EXTENT 34           R-free                     1679176
   sga heap(1,0)         EXTENT 34        R-freeable   reserved stoppe 96
```

Duration: A KGH policy

It gets interesting here. Notice that all "sga heap(1,0)" chunks has allocation comment as "perm". Allocation comment "perm" is passed for permanent chunks. In a nutshell, all permanent chunks are allocated from first mini-heap in each of these sub heaps i.e. sga heap(1,0), sga heap(2,0) and sga heap(3,0) and so on. They are only allocated in the first mini-heap and not in any other mini-heap in these sub-heaps. Another example: PL/SQL DIANA type chunks are allocated only in fourth mini-sub-heap (1,3) (2,3) and (3,3) [in this instance].

```
   sga heap(1,3)          freeable   PL/SQL DIANA    1318912
...
   sga heap(2,3)          freeable   PL/SQL DIANA    1675296
...
   sga heap(3,3)          freeable   PL/SQL DIANA    1458176
```

Point is that there is a new KGH policy based upon duration of a chunk. All chunks are classified based upon their type (or more formally based upon the duration of that chunk) and the chunks with the same duration is allocated from the same shared pool mini-heaps. I think, this is a great idea, especially since only few heaps of shared pool tend to be over worked and that should not unnecessarily flush other parts of the sub-heap. Another important thing to consider is that shared pool extents can be de-allocated and allocated to other parts of SGA. For example, shared pool extents can be deallocated from shared pool and allocated to buffer cache to increase buffer cache size [These reallocated

chunks will be marked with a comment KGH:NO ACCESS]. By keeping the perm chunks in the first mini-heap, extents can be deallocated quite easily, without the need to move the perm chunks to different areas.

So, What's the problem?

Drawback is that these chunks are contained in that mini-heap. For example, let's say that chunks with permanent duration must be allocated and there is no free space in the first mini-heap (say sga heap(1,0)) and if a new extent can't be added to that mini-heap then ORA-4031 error is thrown even if there is plenty of free space in some other mini-heap (say sga heap(1,1), sga heap(1,2) etc). So permanent chunks will be allocated only from sga heap(N, 0) [where N is sub-heap id 1, 2,3...] and if that mini-heap runs out of space, ORA-4031 will be thrown.

This is why client encountered ORA-4031 errors. Even though there was plenty of free memory in other pools, simply permanent chunks can not be allocated in the first mini-heap leading to ORA-4031 errors. Of course, other chunks in that mini-heap can not be deallocated either since those chunks are also permanent chunks. Chances of these errors occurring in other sub-heaps such as sga heap(N,1), sga heap(N,2) etc are less since recreatable/freeable chunks can be flushed to accommodate incoming requests. Quick resolution was to increase shared pool_size temporarily (until we can reduce perm chunk usage due to another issue, which will be resolved soon).

Parameter:_enable_shared_pool_durations

Above undocumented parameter controls this specific KGH policy and default is true. [Note that setting an undocumented parameter needs Oracle support blessing in a production database]. I tested this in my test database setting parameter _enable_shared_pool_durations=false and mini-heaps disappeared. If this parameter is false, then duration of a chunk is not considered. Following example shows that EXTENT 0 in sga heap(1,0) contains non-perm allocations if this parameter is set to FALSE.

```
EXTENT 0 addr=3CC00000
...
  Chunk 3cfdf23c sz=     4096    freeable  "sql area       "  ds=3CFE4E0C
  Chunk 3cfe023c sz=      540    recreate  "KQR PO         "  latch=41AE8FD4
...
```

Summary

In essence, duration enabled chunk allocation policy has been introduced in 10g [I don't know exact version]. This is usually beneficial and ORA-4031 errors are possible even if there is plenty of free space in other mini-heaps.