Sunday, January 8. 2012

Extermination

Buffer Extermination? WTF? Normally i'm seeing wait events like "buffer busy", "log sync" or "db file sequential read" when doing my research in Oracle installation in Top5 events when i'm called because of a situation where the performance is not quite at the level the customer wants. I was sitting in front of the console of an system still using 10g as it's database.

I want to add, that the performance problem had its root somewhere else and was quickly found, however this log wait sparked my interest. A much simpler reason. It was the curiosity afterwards, why there were peaks in the wait event statistics in regular intervals with this wait event i never saw before.

"Buffer exterminate"? WTF ... again. Sounds dangerous. Never saw that before in that list, and than my brain rotated … what the heck is "Buffer Exterminate", i have an idea, something is ringing in my head, but somehow my long-term memory management unit of my brain was unable to stage this information in to current working set. Okay … ask Dr. Google.

Metalink [ID 259137.1] is of great help here. The "buffer exterminate" wait occurs (and can only occur), when the buffer cache is shrunk dynamically and a session wants to access data that is in the granule of the buffer that is chosen by Oracle for removal from the buffer cache. The session wanting the block has to wait until the buffer to be removed has been freed to read it from disk then. You can't simply read the block from disk without waiting, as the block in the granule may represent a new state of the block than the one on the disk an simply reading the one from the disk would yield just outdated data. So you can just wait until the granule has been released.

Before you ask, what a granule is: Oracle doesn't allocate memory in the SGA bytewise, but in so called granules. A granule is 4 MB of memory, when your SGA is up to 1 GB when the instance starts. It's 16 MB when your SGA is larger than 1GB at startup.

In Oracle DB 10g, there is a feature called "Automatic Shared Memory Management". The idea is, that Oracle itself monitors the load and configures the layout of the SGA. I think of automatic means as a very good feature. It's like with manual and automatic gearboxes. Surely, a good driver can accelerate faster with a manual gearbox than with an automatic gearbox, however an automatic gearbox is faster and better than 99% of all drivers. That said, given the existence of behaviour patterns explained by the Dunning-Kruger-effect (h/t to Chris Colomb for hinting me to this interesting phenomenon), 99% of drivers think are part of the 1%. This is especially epidemic in Germany. But back to the issue. It's the same with tuning of systems

You activate the ASMM by setting the parameter SGA_TARGET to a value unequal to zero. Now the system sizes the buffer cache (DB_CACHE_SIZE), shared pool (SHARED_POOL_SIZE), large pool (LARGE_POOL_SIZE) and Java pool (JAVA_POOL_SIZE) automatical within the limit set by SGA_TARGET. If one of the other parameters controling one of the mentioned memory areas is set to a value other than 0, the value is assume as the minimum amount of memory.

Of course: When you have fixed SGA_TARGET and you want to grow one part, another has to shrink. It's obvious that you can't do shrinking simply by throwing the block out of the memory. There may be dirty blocks in that granule(changed blocks that weren't written to disk so far by the database writer to the database file, just to the redo logs).

This works really good and this relieves the admin from investing time to find good values for some of the most important SGA parameters.

However if your database tries to move memory back and forth from one kind of shared memory to another tens of times per hour this is surely not without impact on your database performance. I had such a situation in this case. The system started to move around memory in minute intervals just to move it back a minute or two later. As most automatic systems they will work perfectly within their specification, but you may hit a situation where tries to get most out of a situation with restricted resources, where the SGA is confronted with the situation that all components want more memory and as soon you remove memory from one parts, the other part cries and wants its memory back. That's similar to the argument with your significant other about what's the half of the blanket. Better have two blankets Or to get back to the topic: Have enough SGA ...

How do you find out, how many resizing operations took place? You can look that up by a select statement as described in this blog:
select START_TIME, component, oper_type, oper_mode,status, initial_size/1024/1024 "INITIAL", target_size/1024/1024 "TARGET", FINAL_SIZE/1024/1024 "FINAL", END_TIME from v$sga_resize_ops order by start_time, component;
With this statement you will see the recent history of resizings.

In this case a slight increase (4 gigs) of the target size of the SGA moved the system away from growing and shrinking the buffers back and forth. And not a single "buffer extermination" was seen afterwards and no peaks in the wait time statistics and the number of resizing ops was down to one per hour. And that was more than okay.

Other solutions would be the deactivation of ASMM (by setting SGA_TARGET to zero) and configuring everything manually(doing it the old way) or setting some reasonable minima for the values controlled by ASMM. Important to know: In the amount specified SGA_TARGET is not only the amount of memory for the four parts mentioned before, it's for the complete SGA. So the amount of memory used for other parts of the SGA than those managed by ASMM has to deducted from the SGA_TARGET size. And this reduced amount of SGA is available for the SGA areas managed by ASMM.

`Posted by Joerg Moellenkamp in General at 14:12`

I never heard of this wait event till this blog post. I learn something everyday.
Anonymous on Jan 9 2012, 01:04