Endlich mehr Memory Management mit Oracle 12.2

Sebastian Winkler
AbbVie Deutschland GmbH & Co. KG
Ludwigshafen am Rhein

Schlüsselworte
Memory Management, Oracle 12.2, Multitenant

Einleitung

Der Anfang vom Ende von Automatic Memory Management - kurz AMM? Das könnte man jedenfalls meinen, wenn man aktuell eine 12.2er Datenbank mit dem DBCA installiert. Denn neuerdings ist es nicht mehr möglich, AMM zu verwenden, wenn die Maschine mehr als 4 GB physikalischen Arbeitsspeicher mitbringt. 4 GB RAM ist heutzutage nun wirklich nicht mehr viel, bleibt also das bewährte ASMM. Und trotzdem - nach wie vor empfiehlt Oracle in seiner aktuellen Dokumentation die Nutzung von AMM. Wie passt das zusammen? In diesem Vortrag möchte ich mögliche Gründe, sowie die Vor- und Nachteile der verschiedenen Speichertechniken vorstellen. Besonderer Fokus liegt hier natürlich auch auf Oracle 12.2 und der Multitenant-Option. Bis jetzt war es nicht möglich, speicherfressende PDBs in ihre Schranken zu weisen. Mit 12.2 wird das endlich anders und man kann auch auf PDB-Ebene nicht nur minimale, sondern vor allem auch maximale PGA und SGA festlegen. Was gilt es also zu beachten bei den neuen PDB-Memory-Parametern und helfen diese wirklich? Noch bevor wir aber soweit sind, muss auch unser darunterliegendes Betriebssystem richtig konfiguriert sein. Wo Windows uns wenig Spielraum lässt, wird er unter Linux häufig nicht richtig genutzt, oder gar falsch konfiguriert. Das geht auf die Performance. Um das Ganze also abzurunden, werden wir uns auch noch um die richtige Konfiguration von Hugepages und um das Sizing der Datenbank gegenüber dem Betriebssystem kümmern.

Memory Management

Was bedeutet eigentlich Memory Management? Dieser Begriff beinhaltet zunächst einmal die optimale Aufteilung der Speicherbereiche der Oracle Instanz den Anforderungen der Datenbank entsprechend aufrecht zu erhalten. Diese Aufgabe gestaltet sich häufig schwierig, da auch heute noch Speicher für die meisten Datenbanken ein begrenztes Gut darstellt, sich Anforderungen an Datenbanken im Laufe der Zeit ändern und zuweilen auch unbegrenzter Speicher nicht vor Fehlkonfigurationen bewahrt. Gerade Oracle 12c hat mit der Multitenant Option vieles gerade im Bereich des Memory Management nicht einfacher gemacht. Mit dem ersten Release war es zwar möglich viele Pluggable Databases parallel unter einer Verwaltung isoliert zu betreiben, aber genau diese Isolation hat im Bereich Speicherverwaltung gefehlt.

Mit 12.2 hat sich dem einen oder anderen DBA aber zunächst ein gravierenderes Problem gestellt, sofern er bisher Automatic Memory Management kurz AMM genutzt hat. Hier stößt man nämlich unter Umständen auf folgende Fehlermeldung:

```
[INS-35178]The Automatic Memory Management option is not allowed when the
total physical memory is greater than 4GB.
```

Oracle unterstützt also AMM nicht mehr, sofern der Server mehr als 4GB Arbeitsspeicher verwaltet. Heißt auch, sollte ich es nachträglich trotzdem konfigurieren wollen, ist das wohl keine gute Idee und was ist dann mit meinen 11g Datenbanken die ich schon lange mit AMM und mehr als 4GB RAM betreibe? Es nützt alles nichts, die Konsequenz lautet, ich muss mich mit ASMM, also dem Quasi „Vorgänger“ beschäftigen. Dabei hat Oracle, angefangen bei Oracle 9i automatisiertes PGA Management mit dem Parameter PGA_AGGREGATE_TARGET eingeführt. Bei Oracle 10g mit

automatisierter SGA Verwaltung mit dem Parameter SGA_TARGET weitergemacht und mit Oracle 11g dann schließlich Automatic Memory Management eingeführt, bei dem man nur noch einen gesamten Speicherbereich definieren musste, welchen Oracle dann benutzt um PGA und SGA zusammen dynamisch zu verwalten. Mit zwei Parametern ist das Ganze relativ einfach zu steuern MEMORY_TARGET und MEMORY_MAX_TARGET. Wobei MEMORY_MAX_TARGET das obere Limit darstellt und man MEMORY_TARGET dynamisch ohne Neustart der Instanz erhöht und verringert werden kann. SGA_TARGET und PGA_AGGREGATE_TARGET stellen bei dieser Konfiguration ein Minimum dar. Will man Oracle die Speicherverwaltung komplett überlassen, sollte man beide Werte auf 0 setzen. Unter Linux/Unix Systemen sollte man auch das Shared Memory File System überprüfen.

```
df -k /dev/shm

Filesystem           1K-blocks      Used Available Use% Mounted on
tmpfs                  1253644    325466    665848  35% /dev/shm
```

Es sollte groß genug sein, um MEMORY_MAX_TARGET unterzubringen. Andernfalls erhält man folgende Fehlermeldung:

```
ORA-00845: MEMORY_TARGET not supported on this system
```

Um das Automatic Memory Management zu verwalten, stellt Oracle ein paar V$ Views bereit, bspw.:

```
V$MEMORY_CURRENT_RESIZE_OPS
V$MEMORY_DYNAMIC_COMPONENTS
V$MEMORY_RESIZE_OPS
V$MEMORY_TARGET_ADVICE
```

Wobei die letzte View V$MEMORY_TARGET_ADVICE Hinweise zum Anpassen des MEMORY_TARGET gibt. Es zeigt eine Reihe von möglichen Werten an und als Faktor des aktuellen Parameters errechnet es dann, basierend auf dem aktuellen Workload eine potenzielle Datenbankzeit mit den verschiedenen anderen Memory Größen:

```
SELECT * FROM v$memory_target_advice ORDER BY memory_size;

MEMORY_SIZE MEMORY_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR    VERSION
----------- ------------------ ------------ ------------------- ----------
        304                .75         5048              1.0038          2
        405                  1         5036                   1          2
        506               1.25         5036                   1          2
        607                1.5         5036                   1          2
        708               1.75         5036                   1          2
        809                  2         5036                   1          2

6 rows selected.

SQL>
```

Das Ganze AMM ist in der Theorie gut gedacht, führt in der Praxis aber leider immer wieder zu irreführenden Memory Werten, Overhead durch hin- und her Skalieren und letztlich auch zu Bugs bzw. Fehlern. Wenn man jetzt zurück blickt auf Automatic Shared Memory Management kurz ASMM, dann haben wir hier die verlässlichere Technik. Wir müssen also zunächst SGA_TARGET setzen statt MEMORY_TARGET. Grundsätzlich definiert man die Größe, welche man beim Instanzstart zur

Verfügung haben möchte. Das meiste davon wird als Buffer Cache und Shared Pool verwendet. Wie bei MEMORY_TARGET auch, gibt es hier einen entsprechenden SGA_MAX_SIZE Parameter. Es muss schließlich noch PGA_AGGEGATE_TARGET gesetzt werden. Der Shared Pool kann und wird von allen Prozessen verwendet werden, zusätzlich kann jeder einzelne Prozess privaten Speicher allokieren, das findet im PGA statt und wird über PGA_AGGREGATE_TARGET gesteuert. Beide Parameter dürfen zusammen genommen natürlich nicht den verfügbaren Speicher überschreiten und das Betriebssystem sollte auch noch genügend Luft zum Atmen übrighaben. Mehr dazu im Vortrag.

Multitenant
Im vorherigen Release, also Oracle 12.1 konnte man die Menge, die eine einzelne PDB benutzt nicht steuern. Als Konsequenz daraus, konnte eine einzelne aus dem Ruder laufende PDB jede Menge Speicher auffressen und damit die Performance anderer PDBs derselben Instanz in die Knie zwingen. Oracle 12.2 gibt hier die Möglichkeit die Speicherbereiche einer PDB zu kontrollieren und macht eine Konsolidierung in die Multitenant Umgebung verlässlicher und sicherer.

Folgende Parameter können jetzt auf PDB Ebene gesetzt werden:

Minimaler Buffer Cache:	DB_CACHE_SIZE
Minimaler Shared Pool:	SHARED_POOL_SIZE
Maximale PGA:	PGA_AGGREGATE_LIMIT
Maximum PGA:	PGA_AGGREGATE_TARGET
Minimum SGA:	SGA_MIN_SIZE
Maximum SGA:	SGA_TARGET

Es gibt einige Einschränkungen für die CDB bei der Nutzung dieser Parameter, so wird bspw. NONCDB_COMAPTIBLE auf FALSE, welches noch einige ALTER DATABASE oder ALTER SYSTEM Befehle außerhalb des ROOT Containers zugelassen haben. MEMORY_TARGET ist auch hier nicht mehr nutzbar. Letztlich haben die einzelnen Parameter Größenbeschränkungen um eine zu hohe bzw. „Über-“ Allokierung der Speichers zu verhindern.

Setzen der Parameter wie bisher:

```
CONN / AS SYSDBA
ALTER SESSION SET CONTAINER=pdb1;

SQL> ALTER SYSTEM SET sga_target=2G SCOPE=BOTH;

System altered.

SQL> SHOW PARAMETER sga_target;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
----
sga_target                           big integer 2G
SQL>
```

Fehler produzierender Versuch einer „Über-„Allokation:

```
SQL> ALTER SYSTEM SET sga_target=6G SCOPE=BOTH;
ALTER SYSTEM SET sga_target=6G SCOPE=BOTH
*
```

```
ERROR at line 1:
ORA-02097: parameter cannot be modified because specified value is invalid
ORA-56747: invalid value 6442450944 for parameter sga_target; must be smaller
than parameter sga_target of the root container

SQL>
```

Für das Monitoring des Ressourcenverbrauchs der einzelnen PDBs hat Oracle auch hier ein paar Views geliefert. V$RSRCPDBMETRIC und V$RSRCPDBMETRIC_HISTORY geben Auskunft über CPU, I/O, Memory und weitere Werte. Diese beiden unterscheiden sich im Zeitraum der Betrachtung, wobei die erste View für jede PDB eine Zeile der letzten Minute zurück gibt und die zweite für jede PDB 61 Zeilen, also die letzte Stunde zurück gibt. DBA_HIST_RSRC_PDB_METRIC hält dann noch AWR Snapshots entsprechend der AWR Retention Period bereit.

Falls man hier noch nicht ausreichend Stellschrauben zur Verfügung hat bietet sich die Auseinandersetzung mit dem Resource Manager unter 12c an. Auf CDB Ebene kann man die Ressourcenzuweisung an einzelne PDBs steuern um eine oder einige PDBs zu priorisieren. Auf PDB Ebene kann man die Ressourcenzuweisung zu einzelnen Sessions steuern, wie man es aus vorherigen Releases bereits kennt. Mehr dazu im Vortrag.

OS

Zum Abschluß noch ein paar Worte zum Betriebssystem, speziell Linux/Unix. Wenn Sie eine Oracle Datenbank auf einem Linux Server mit mehr als 16 GB physical Memory betreiben und die System Global Area (SGA) größer als 8 GB ist, sollten Sie HugePages konfigurieren. Oracle verspricht durch diesen Schritt eine höhere Performance. Eine HugePages Konfiguration bedeutet, dass der Linux Kernel mit „large pages“, wie Oracle sie allgemein nennt umgehen kann. Also statt der Standard 4 KB auf x86 und x86_64 oder 16 KB auf IA64 Systemen, dann mit 4 MB auf x86, 2 MB auf x86_64 und 256 MB auf IA64 System. Größere Pages bedeuten, dass das System weniger Page Tables benutzen, weniger Mappings verwalten muss, und so den Aufwand für deren Verwaltung und die Zugriffe reduziert werden. Allerdings gibt es eine Einschränkung seitens Oracle, denn Automatic Memory Management unterstützt leider keine HugePages. Sollten Sie Automatic Memory Management (AMM) benutzen, also haben MEMORY_TARGET gesetzt, müssen Sie es an dieser Stelle abschalten und wieder auf Automatic Shared Memory Management (ASMM) umstellen, also wieder SGA_TARGET und PGA_AGGREGATE_TARGET setzen. Des Weiteren gibt es noch eine Neuerung namens Transparent Hugepages (THP), welche ausgeschalten werden sollte. Sie werden in einem Red Hat Linux ab Version 6 oder einem entsprechenden Derivat mitgeliefert. Sowohl Oracle als auch Red Hat empfehlen die Abschaltung von THP.

Zunächst sollten wir unseren aktuellen „physikalisch“ verfügbaren Arbeitsspeicher überprüfen. Im Beispiel haben wir also knapp 128 GB RAM. SGA_TARGET und PGA_AGGREGATE_TARGET zusammen, sollten nicht größer als der vorhandene Arbeitsspeicher sein. Zudem sollte ausreichend Speicher für OS Prozesse selbst zur Verfügung stehen.

```
grep MemTotal /proc/meminfo

MemTotal: 132151496 kB
```

Im zweiten Schritt überprüfen wir unsere Datenbankparameter. Zunächst: AMM ausgeschalten? MEMORY_TARGET und MEMORY_MAX_TARGET sollten auf 0 stehen.

```
SQL> select value from v$parameter where name = 'memory_target';
```

```
VALUE
-----
0
```

Wie groß ist unsere SGA? In diesem Beispiel knapp 40 GB. Wichtig: In der nachfolgenden Abfrage wird direkt in kB umgerechnet (value/1024). Sodass man mit diesem Wert anschließend weiter rechnen kann.

```
SQL> select value/1024 from v$parameter where name = 'sga_target';

VALUE
--------
41943040
```

Schließlich sollte per Default der Parameter use_large_pages in der Datenbank eingeschalten sein.

```
SQL> select value from v$parameter where name = 'use_large_pages';

VALUE
-----
TRUE
```

In diesem Beispiel verwenden wir einen x86_64 Red Hat Enterprise Linux Server. Also sollte per default die Hugepagesize auf 2 MB gesetzt sein.

```
grep Hugepagesize /proc/meminfo

Hugepagesize: 2048 kB
```

Für die Kalkulation der Anzahl der Hugepages gibt es eine einfache Faustformel:

```
SGA / Hugepagesize = Anzahl Hugepages
```

Also diesem Beispiel folgend.

```
41943040 / 2048 = 20480
```

Betreibt man auf einem Server mehrere Datenbanken, so sollte man in die Berechnung die SGA aller Instanzen einbeziehen.

```
(SGA 1. Instanz + SGA 2. Instanz + … etc.) / Hugepagesize = Anzahl Hugepages
```

Alternativ bietet Oracle unter der Doc ID 401749.1 auch ein Skript namens hugepages_settings.sh an, welches die entsprechende Berechnung vornimmt. Dieses bezieht auch eine Prüfung der Kernel Version und die aktuell verwendeten Shared Memory Bereiche durch das SGA ein. Beachten sollte man, dass sich die Berechnung auf die aktuelle SGA Größe und Verwendung bezieht. Das heißt also erst SGA anpassen, Datenbank durchstarten und dann das Skript ausführen. Ergebnis sollte dann folgender Satz sein. Überprüfen Sie doch einfach einmal Ihre Berechnung mit dem Ergebnis des Skripts:

```
Recommended setting: vm.nr_hugepages = 20480
```

Im nächsten Schritt müssen wir die Anzahl der Hugepages im entsprechenden Server configFile eintragen. Dafür benötigen Sie root Berechtigungen. Unter Red Hat Linux 6 /etc/sysctl.conf

```
vi /etc/sysctl.conf

vm.nr_hugepages=20480
```

Richtig eingetragen, sollte folgendes Ergebnis zurückgeliefert werden:

```
grep vm.nr_hugepages /etc/sysctl.conf

vm.nr_hugepages=20480
```

Als weiteres müssen wir in der /etc/security/limits.conf die Parameter hard und soft memlock für den oracle User eintragen. Diese Werte sollten kleiner als der verfügbare Speicher in kB sein. Mindestens sollten aber unsere SGA bzw. unsere Hugepages hineinpassen. Daraus ergibt sich folgende Berechnung:

```
Anzahl Hugepages * Hugepagesize = minimum Memlock
```

Dem Beispiel folgend:

```
20480 * 2048 = 41943040


vi /etc/security/limits.conf

oracle soft memlock 41943040

oracle hard memlock 41943040
```

Wie bereits angesprochen müssen wir ab Red Hat Linux 6 Transparent Hugepages ausschalten:

```
cat /sys/kernel/mm/redhat_transparent_hugepage/enabled

[always] madvise never


echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled

echo never > /sys/kernel/mm/redhat_transparent_hugepage/defrag


cat /sys/kernel/mm/redhat_transparent_hugepage/enabled

always madvise [never]
```

Haben Sie alle Parameter gesetzt starten Sie den Server komplett durch. Alternativ kann man mit sysctl -p die Parameter auch für den laufenden Server aktivieren.

Sitzt memlock korrekt?

```
ulimit -l

41943040
```

Sind HugePages richtig konfiguriert und auch in Nutzung?

```
grep Huge /proc/meminfo

AnonHugePages: 538624 kB
HugePages_Total: 20480
HugePages_Free: 12292
HugePages_Rsvd: 8188
HugePages_Surp: 0
Hugepagesize: 2048 kB
```

Sind Transparent Hugepages ausgeschalten?

```
cat /sys/kernel/mm/redhat_transparent_hugepage/enabled

always madvise [never]
```

Nutzt die Datenbank HugePages? Dazu schauen wir uns das Alert Log an. Nach dem „Starting ORACLE instance (normal)“ sollte als folgender Eintrag „Large Pages Information“ zu finden sein:

```
********************** Large Pages Information ******************
Per process system memlock (soft) limit = 100 GB
Total Shared Global Region in Large Pages = 40 GB (100%)
Large Pages used by this instance: 20481 (40 GB)
Large Pages unused system wide = 0 (0 KB)
Large Pages configured system wide = 20481 (40 GB)
Large Page size = 2048 KB
*****************************************************************
```

Bei einer Fehlkonfiguration gibt Oracle an dieser Stelle auch Empfehlungen für die richtige Konfiguration. Im nachfolgenden Beispiel fehlt eine einzige Page, also 2048 kB memlock um 100% des SGA als Hugepages zu allokieren:

```
********************** Large Pages Information ******************
...
...
RECOMMENDATION:
Total System Global Area size is 40 GB. For optimal performance,
prior to the next instance restart:
1. Increase the number of unused large pages by
at least 1 (page size 2048 KB, total size 2048 KB) system wide to
get 100% of the System Global Area allocated with large pages
2. Large pages are automatically locked into physical memory.
Increase the per process memlock (soft) limit to at least 40 GB to lock
100% System Global Area's large pages into physical memory
*****************************************************************
```

Kontaktadresse:
Sebastian Winkler
AbbVie Deutschland GmbH & Co. KG
Knollstraße
D-67061 Ludwigshafen

Telefon: +49 (0) 621-589 2007
Mobil: +49 (0) 172-408 1139
E-Mail sebastian.winkler@abbvie.com
Internet: www.abbvie.de