Ressourcen Management – Alles Automatisch?

Johannes Ahrends
CarajanDB GmbH
Erftstadt

Schlüsselworte

Automatic Memory Management, SGA, PGA, Shared Memory Verwaltung, Resourcen Manager, Instance Caging

Einleitung

Wenn man der Oracle Dokumentation glauben darf (und das sollte man ja), dann braucht man für die Memory Verwaltung der Datenbank bei Oracle 11g nur noch einen einzigen Parameter „MEMORY_TARGET“. Damit ist dann alles in Ordnung und Oracle verwaltet dann die einzelnen Bereiche, SGA mit dem Buffer Cache, dem Shared Pool und diversen kleinerer Pools sowie der PGA automatisch und sieht immer zu, dass der Bereich, der gerade Memory benötigt, diesen auch bekommt.

Aber wie sieht das in der Praxis aus? Gibt es da vielleicht doch die ein oder andere Einschränkung?

Der Vortrag beschäftigt sich mit den Einsatzmöglichkeiten und Grenzen der Memory Verwaltung aber auch der CPU-Zuordnung und Möglichkeiten, die der Resourcen Manager ermöglicht. Leider kann an dieser Stelle nicht auf die Möglichkeiten von Oracle 12c eingegangen werden, da diese noch nicht offiziell verfügbar ist.

Memory Management

Seit Oracle 11g gibt es die Möglichkeit, die Verwaltung des Memorys allein der Oracle Datenbank zu überlassen. Unter Linux muss dafür eine Posix-konforme Konfiguration des Shared-Memories verwendet werden, die die dynamische Vergabe von Shared Memory erst möglich macht. Allerdings muss dabei schon die erste Einschränkung gemacht werden: Der Paramater heißt nicht umsonst „`MEMORY_TARGET`“ und nicht „`MEMORY_SIZE`“. Man kann also auch „über das Ziel hinaus schießen“!

Der so parametrierte Hauptspeicher gliedert sich in zwei unterschiedliche Bereiche: SGA und PGA. Und genau hier liegt auch das Problem: Die PGA, also der Bereich, der für Cursor, Sortierungen, PL/SQL Tabellen und vielen weiteren Strukturen zur Verfügung gestellt wird, kann derzeit nicht nach oben beschränkt werden. Wenn ein Programm z.B. sehr viele Memory-Strukturen allokiert, müssen diese in der PGA zur Verfügung gestellt werden, egal ob die maximal definierte Größe der PGA bzw. des Memorys erreicht wurde oder nicht. Daher kann es dazu kommen, dass das tatsächlich verbrauchte Memory der Instanz größer ist als durch `MEMORY_TARGET` vorgegeben.

Der nächste Part betrifft die SGA. Dieser teilt sich im Wesentlichen in den Bereich Buffer Cache und Shared Pool auf. Für viele Oracle Datenbanken ist es sicherlich okay, dass die Größen hierfür variabel gehalten werden und nur durch die Maximalgröße „`SGA_TARGET`“ bzw. „`SGA_MAX_SIZE`“ begrenzt wird. Allerdings kommt es in der Praxis immer wieder zu Situationen, dass der Buffer Cache im Betrieb kleiner ist als der Shared Pool. Dies ist in der Regel darin begründet, dass die Anwendung(en) keine Bindevariablen benutzen. Das bedeutet, dass die Möglichkeit der Wiederverwendung von SQL-Befehlen im Shared-Pool sehr gering ist, und für das Memory

Management ist das fälschlicherweise das Zeichen, dass der Shared Pool vergrößert werden muss. Und das geschieht in der Regel auf Kosten des Buffer Caches. Die Verwendung des Parameters `CURSOR_SHARING = FORCE` hilft hierbei in der Regel nur bedingt, besser ist es, für die Caches entsprechende Untergrenzen zu definieren.

Ressourcen Sharing

Ein sehr angenehmer Effekt bei der Virtualisierung ist, dass die Ressourcen, also Memory und CPUs, die einzelne Gastsysteme verwenden dürfen, sehr genau zugeordnet werden können. Dabei geht es nicht darum, die Ressourcen exakt zu verteilen, sondern man geht bewusst das Risiko einer Überprovisionierung ein, weil davon ausgegangen wird, dass die Ressourcen nie gleichzeitig benötigt werden. D.h. auf einem Server mit 16 CPUs können 6 Gastsysteme jeweils 4 CPUs erhalten. Die Resourcen-Verwaltung erlaubt es außerdem, dass bestimmte Gastsysteme höher priorisiert werden und daher im Notfall auch mehr CPUs zugeteilt bekommen.

Doch wie sieht das bei Systemen aus, bei denen mehrere Oracle Datenbanken auf einem Rechner (ob physikalisch oder virtuell ist dabei unerheblich) betrieben werden? Mit Version 11 Release 2 gibt es hierfür das Schlagwort „Instance Caging“. Dahinter verbirgt sich zunächst die Möglichkeit einer Instanz eine Anzahl von CPUs zuzuordnen. Während in der Vergangenheit der Serverparameter CPU_COUNT von Oracle aufgrund der Anzahl CPUs bzw. Threads vergeben wurde, ist es jetzt möglich bzw. erlaubt, einer Instanz durch Setzen des Parameters CPU-Ressourcen zuzuteilen. Allerdings funktioniert dies nur, wenn außerdem der Resourcen Manager genutzt wird. D.h der Parameter `RESOURCE_MANAGER_PLAN` muss mindestens auf den Wert „`DEFAULT_PLAN`“ gesetzt sein.

Aber auch, wenn nur eine Datenbank auf dem Server betrieben wird, kann der Resourcen Manager gute Dienste leisten. Gerade im Bereich des Oracle Real Applications Clusters kommt es z.B. immer wieder vor, dass mehrere Anwendungen in einer Datenbank betrieben werden. Über den Resourcen Manager können jetzt bestimmte Services priorisiert werden oder es kann dafür gesorgt werden, dass Programme, die Ad-Hoc Abfragen machen (z.B. SQL Developer) nur wenige Ressourcen verwenden dürfen.

Fazit

Sicherlich ist es zunächst ausreichend für den Betrieb einer Oracle Datenbank die Standardvorgaben zu wählen. Bei näherer Betrachtung stellt sich allerdings heraus, dass ein Feintuning hilfreich ist, und glücklicherweise bietet Oracle hierfür eine ganze Reihe von Möglichkeiten an. In dem Vortrag kann sicherlich nur auf ein paar wenige Beispiele eingegangen werden. In der Praxis hat sich gezeigt, dass zunächst ein gutes Verständnis für die Anwendung, d.h. ein Gespräch mit den Anwendungs-entwicklern unerlässlich ist. Auch Reports, wie AWR oder der Health Check im Toad for Oracle helfen, Entscheidungen zu treffen bzw. getroffene Entscheidungen zu untermauern.

Kontaktadresse:

Johannes Ahrends
CarajanDB GmbH
Siemensstraße 25
D-50374 Erftstadt
Telefon: +49 (22 35) - 1709184
Fax: +49 (22 35) - 1708979
E-Mail johannes.ahrends@carajandb.com
Internet: www.carajandb.com