Oracle 11g Release 2: Änderungen „unter der Haube“

Dierk Lenz
Herrmann & Lenz Services GmbH
Burscheid

Schlüsselworte
Oracle 11g Release 2, New Features

Einleitung

Die *New Features* der neuen Datenbankversion sind mittlerweile in vielen Vorträgen diskutiert worden. Allerdings sind gerade in 11.2 viele Dinge intern geändert worden. Dies führt teilweise zu nicht erwartetem Verhalten des Oracle RDBMS. Der Vortrag beschreibt einige dieser Neuerungen und kann somit bei der Analyse von Fehlersituationen oder *eigenartigem* Verhalten helfen.

Wieder einmal: Cursor-Sharing

Die Version, mit der der Serverparameter `cusror_sharing` eingeführt wurde, war Oracle8i. Er hat die Diskussionen rund um das Thema Bindevariablen und Wiederverwendung von SQL-Code im Shared Pool um viele Aspekte erweitert – und sorgt nach wie vor für Verwirrung. Zur Wiederholung nochmals die Fakten:

Man unterscheidet für die Weitergabe von Werten an SQL-Befehle zwei Methoden: Literale und Bindevariablen. Verwendet man Literale, so wird der aktuelle Wert als Konstante in den SQL-Text eingefügt (…`WHERE KDNR = 4711`). Das ist unpraktisch, da der gleiche Befehl mit einem anderen Wert einen anderen Text hat und daher erneut im Shared Pool gespeichert werden muss, obwohl es doch ablauftechnisch genau das Gleiche tut. Daher kann man statt des Literals eine Bindevariable verwenden (…`WHERE KDNR = :VAR1`). Vor der Ausführung bindet man den gewünschten Wert an die Variable. Das hat mehrere Vorteile:

1. Der SQL-Befehl muss nur einmal wirklich übersetzt werden.
2. Mehrere Benutzer können eine einige Kopie des Befehls im Shared Pool nutzen.

Nun gibt es immer wieder Anwendungen, die dieses einfache Prinzip nicht beachten. Das führt oft zu Problemen mit dem Shared Pool; lange Laufzeiten beim Übersetzen der SQL-Befehle sind noch eine harmlose Folge. Hier kommt nun der Serverparameter `cursor_sharing` ins Spiel: Stellt man ihn auf den Wert `FORCE`, so übernimmt der Oracle Server einen Teil der Programmierarbeit, tauscht vor dem Übersetzen Literale in Bindevariablen, übersetzt den SQL-Befehl und bindet sofort danach den Literalwert an die Variable. Dies verhindert zumindest das *Fluten* des Shared Pools.

Das mechanische Ersetzen von Literalen hat aber einen entscheidenden Nachteil: Gezielt verwendete Literale, die den Optimizer mit wichtigen Informationen versorgen sollen – bei nicht gleichverteilten Spaltenwerten – werden ebenfalls ersetzt. Entsprechende Histogramme werden nicht mehr zur Generierung unterschiedlicher Ausführungspläne genutzt, was oft zu längeren Laufzeiten führt. Hierfür hat Oracle den Wert `SIMILAR` für `cursor_sharing` vorgesehen: Der Optimizer prüft nun vor der Ersetzung der Literale, ob eventuell Informationen zur Nutzung von Histogrammen verloren gehen. Daher werden nicht mehr alle Literale ersetzt. Historisch war dieser Wert immer dann eine gute Option, wenn es Tabellen mit entsprechenden Werteverteilungen gab, unabhängig von der Nutzungsart der Datenbank.

Kommen wir zur nun zum neuen Verhalten von Oracle 11g Release 2: Wenn `cursor_sharing` auf `SIMILAR` steht und Literale für Spalten, die Histogramme besitzen, ersetzt werden, dann wird für jede Ausführung des entsprechenden SQL-Befehls ein eigener sogenannter Child Cursor angelegt. Das bedeutet, dass man sehr viele Kopien des SQL-Befehls in `v$sql` findet. (Wir haben mehrere Tausend beobachtet.)

Damit tritt u.U. das Fluten des Shared Pools wieder auf. Somit muss man von der Verwendung von `SIMILAR` bei Systemen mit OLTP-ähnlicher Last dringend abraten.

Parallele Ausführung von SQL-Befehlen

Parallel Execution (PX), früher unter dem Begriff Parallel Query bekannt, existiert bereits seit Oracle7 (Version 7.1). Mit Oracle 11g Release 2 wurden nun einige neue Parameter eingeführt, die die Kontrolle über automatische Parallelisierung vereinfachen sollen.

An dieser Stelle ist zunächst die Diskussion erwähnenswert, inwiefern eine automatische Parallelisierung überhaupt erwünscht ist. Gegen die Automatisierung spricht, dass Parallelisierung von DB-Aktionen die Hardware-Infrastruktur in allen Belangen maximal ausnutzt – IO-Subsystem, CPU und Hauptspeicher sind hier allesamt einbezogen. Das spricht zunächst dafür, dass man Parallelisierung gezielt einsetzt. Das bedeutet, dass man einzelne SQL-Befehle über Hints parallelisiert. Die Hinterlegung eines DOP (*Degree of Parallelism*, Parallelisierungsgrad) im Data Dictionary für Tabellen und Indizes führt hingegen dazu, dass jeder SQL-Befehl, der auf das entsprechende Objekt mit einer Full-Scan-Methode zugreift, parallelisiert wird. Das bedeutet oft, dass zu viele Aktionen parallelisiert werden, was einerseits zur Überlastung der Infrastruktur, andererseits zum Downgrade vieler Aktionen bis hin zur Serialisierung (nicht-parallele Ausführung) führen kann.

Die Neuerungen umfassen die Bereiche

- Automatische Bestimmung des DOP (Auto-DOP),
- Queueing von parallelen Befehlen
- In-Memory-Ausführung.

Die automatische Bestimmung des DOP bedeutet, dass nach der Bestimmung eines Ausführungsplanes versucht wird, die Ausführung mit einem berechneten DOP zu parallelisieren, sollte die berechnete Ausführungszeit für die serielle Ausführung über einem Schwellwert liegen.

Sollten die verfügbaren PX-Server einer Instanz nicht für den gewünschten DOP reichen, so wurde der DOP des Befehl in früheren Versionen heruntergesetzt. Nun ist es möglich, den Befehl zu queuen.

Bei der In-Memory-Ausführung entscheidet Oracle automatisch, ob ein DB-Objekt im Buffer Cache gehalten wird, um wiederholtes physisches Lesen durch PX-Server zu vermeiden.

Die Verwendung dieser Funktionalitäten wird über den Wert des Serverparameters `parallel_degree_policy` gesteuert:

- `MANUAL` schaltet alle genannten Funktionen aus.
- `LIMITED` nutzt den Auto-DOP für Befehle, die (z.B. per Hint) als parallel markiert sind, jedoch ohne Angabe eines DOP. Queueing von parallelen Befehlen und In-Memory-Ausführung finden nicht statt.
- `AUTO` nutzt alle genannten Funktionen.

Der Schwellwert für den Auto-DOP wird über den Parameter `parallel_min_time_threshold` gesteuert und in Sekunden angegeben. Bei der Einstellung `AUTO` werden 10 Sekunden angenommen.

Automatisches Speichermanagement
Automatisches Speichermanagement wird durch die Serverparameter `sga_target` bzw. `memory_target` gesteuert. Hierdurch werden die Größen von Buffer Cache, Shared Pool und – bei der Benutzung von `memory_target` – der PGAs automatisch den Laufzeitbedürfnissen angepasst.

Setzt man beide Parameter auf den Wert 0, so ist das automatische Speichermanagement ausgeschaltet – jedenfalls steht das so im Handbuch. Da das automatische Speichermanagement nicht in allen Fällen wirklich gut funktioniert (z.B. bei vielen SQL-Befehlen mit Literalen), ist es im Normalfall auszuschalten.

Die Aktionen des automatischen Speichermanagements können über die Views `v$sga_resize_ops` bzw. `v$memory_resize_ops` nachvollzogen werden. Überraschenderweise sind dort ab 11.2 auch Einträge zu finden, wenn `sga_target` und `memory_target` auf 0 stehen. Mit 11.2 gibt es sogenannte *Immediate Memory Allocation Requests*, um für den Shared Pool eine Vergrößerung zu erzeugen und ORA-04031-Fehler zu vermeiden.

Da die Vergrößerung des Shared Pools auf Kosten des Buffer Caches stattfindet und somit Performance-relevant ist, ist es u.U. erstrebenswert, dies zu verhindern (und den ORA-04031 zu akzeptieren). Hierzu gibt es einen Parameter `_memory_imm_mode_without_autosga`, der zu diesem Zweck auf `FALSE` gesetzt werden muss.

Data Guard: ORA-16191 Beim Log-Shipping
Das Log-Shipping für Data Guard erfordert eine Anmeldung der primären DB bei der Standby-DB. Dies erfordert traditionell, dass Passwortdateien für beide DBs existieren müssen – mit identischem `SYS`-Passwort.

Oracle 11g bringt einige Änderungen im Bereich Sicherheit mit, z.B. die Unterscheidung von Groß- und Kleinschreibung für Passworte. Des weiteren werden Passwort-Hashes nun mit dem SHA-1-Algorithmus erzeugt, der wesentlich besser als der bisher verwendete Algorithmus ist. SHA-1 verwendet unter anderem ein Salt, d.h. eine zusätzliche zufällige Zeichenfolge, die an das Passwort angehängt wird bevor der Hash berechnet wird. Das hat wiederum zur Folge, dass der Passwort-Hash für das gleiche Passwort nicht immer gleich ist.

Zurück zum Log-Shipping: Da das Passwort beiden DBs nicht bekannt ist, erfolgt die Authentisierung durch den Vergleich der Passwort-Hashes. Sind diese jedoch unabhängig voneinander erzeugt worden, so sind sie – wegen des Salt – trotzdem unterschiedlich und das Log-Shipping läuft in den o.a. Fehler.

Das Problem kann umgangen werden, indem man entweder auf den alten Algorithmus umstellt oder indem man die Passwortdatei nur einmal erstellt und dann für die andere DB kopiert, anstelle sie neu zu erzeugen.

Aus Sicherheitsgründen ist die zweite Methode auf jeden Fall vorzuziehen. Es ist aber zu bedenken, dass bei einer Änderung des `SYS`-Passworts die Passwortdatei auf jeden Fall neu kopiert werden muss.

Verhalten beim Entziehen von `UNLIMITED TABLESPACE`
Entzieht man einem Benutzer das `UNLIMITED TABLESPACE` Systemprivileg, so sind (ab 11.2.0.2) alle eventuell vorher bestehenden Tablespace-Quoten ebenfalls entzogen. In früheren Versionen wurden diese wieder aktiv.

Nun müssen in einem solchen Fall die privaten Tablespace-Quoten erneut vergeben werden.

Kontaktadresse:
Dierk Lenz
Herrmann & Lenz Services GmbH
Höhestraße 37
51399 Burscheid

Telefon:	+49 2174 6712 0
Fax:	+49 2174 6712 22
E-Mail	dierk.lenz@hl-services.de
Internet:	www.hl-services.de