PL/SQL-Tuning

Markus Fiegler
ORDIX AG
Paderborn

Schlüsselworte:

Bulk Binding, Short Circuit Evaluation, NOCOPY, PL/SQL Function Result Cache, Native Kompilierung, PL/SQL Inlining, PLSQL_WARNINGS, Hierarchische Profiler

Einleitung

Die Analyse und Optimierung der Performance unternehmenskritischer Datenbank-Anwendungen hat oft einen sehr großen Einfluss auf deren Erfolg. Dabei führt ein klassisches Datenbank-Tuning oft nur zu einem Teilerfolg. Häufig müssen nicht nur die SQL- sondern auch die PL/SQL-Anweisungen unter die Lupe genommen werden.

In meinem Vortrag werde ich aufzeigen, wie die Laufzeit von PL/SQL-Applikationen deutlich beschleunigt werden kann. Dabei werden unter anderem Features wie PL/SQL Inlining, PL/SQL Function Result Cache oder Native Kompilierung erläutert, die zur Performance-Steigerung von PL/SQL-Applikationen führen. Zudem werden einige Tipps und Tricks aus der Praxis bezogen auf Performance-Verbesserungen von PL/SQL-Programmen dargestellt.

Datentypen

PL/SQL-Tuning beginnt schon bei der Auswahl geeignete Datentypen. Sollen z. B. Ganzzahlen verarbeitet werden, so kann der Einsatz von `PLS_INTEGER` zu deutlichen Performance-Verbesserungen gegenüber z. B. Datentypen wie `INTEGER` oder `NUMBER` führen. Dies ist möglich, da der `PLS_INTEGER` weniger Speicherplatz benötigt als die Datentypen `INTEGER` oder `NUMBER`. Zusätzlich verwendet der `PLS_INTEGER` die Rechnerarithmetik wo gegenüber die Datentypen `INTEGER` und `NUMBER` die Library-Arithmetik nutzen.

Findet eine native Übersetzung der PL/SQL-Programme statt, so führt vor allem der Datentyp `SIMPLE_INTEGER` zu einem zusätzlichen Performance-Gewinn gegenüber dem `PLS_INTEGER`.

Bulk Binding

Da sowohl die PL/SQL- als auch die SQL-Anweisungen in entsprechenden Engines ausgeführt werden, findet bei der Ausführung von PL/SQL-Programmen mit SQL-Anweisungen ein häufiger Kontextwechsel statt, der längere Laufzeiten zur Folge hat.

Um den Overhead beim Kontextwechsel zu reduzieren, kann das so genannte Bulk Binding verwendet werden. Der Vorteil ist, dass mit Hilfe einer Array-Verarbeitung der aufwändige Kontextwechsel zwischen der PL/SQL- und der SQL-Engine reduziert wird.

Short Circuit Evaluation

Oracle stellt mit Short Circuit Evaluation ein Verhalten in PL/SQL zur Verfügung, mit welchem die Auswertung von booleschen Ausdrücken abgebrochen wird, sobald das Ergebnis feststeht. Der Ausdruck wird dabei von links nach rechts ausgewertet. Bei einer UND-Verknüpfung kann z. B. die Verarbeitung schon abgebrochen werden, wenn der linke Teilausdruck bereits unwahr (`FALSE`) ist, da somit gleichzeitig der gesamte Ausdruck unwahr wird.

Der Performance-Vorteil von Short Circuit Evaluation kann noch verstärkt werden, indem aufwändige Teilausdrücke, die z. B. auch aus Funktionen bestehen können, an das Ende eines Ausdrucks verschoben werden. Somit werden aufwändige Teilausdrücke bei der Verarbeitung in einigen Fällen erst gar nicht ausgeführt.

NOCOPY

Im Default-Fall werden `OUT`- und `IN/OUT`-Parameter als Wert (`BY VALUE`) übergeben. Bei der Übergabe als Wert werden die Werte der aktuellen `IN/OUT`-Parameter in den korrespondierenden formalen Parameter kopiert. Beim Verlassen eines Unterprogramms werden die Werte der formalen `OUT`- und `IN/OUT`-Parameter ebenfalls in die korrespondierenden aktuellen Parameter übertragen.

Werden dabei große Datenstrukturen wie z. B. `LOB`-Datentypen, Records oder Collections als Wert an Unterprogramme übergeben, können schnell Performance-Engpässe entstehen.

Eine Abhilfe verschafft hier der Compiler Hint `NOCOPY`, der in der Parameterliste einer Prozedur- oder Funktionssignatur angegeben werden kann. Mit Hilfe des `NOCOPY` Hints wird eine Übergabe einer Referenz (`BY REFERENCE`) ermöglicht.

Da es sich bei `NOCOPY` lediglich um einen Hint handelt, kann die Übergabe als Referenz nicht erzwungen werden. Dies hat zur Folge, dass beim Verlassen des Unterprogrammes mit einer Unhandled Exception nicht sicher davon ausgegangen werden kann, dass der Wert des aktuellen Parameters bereits aktualisiert wurde. Im Default-Fall wird bei einer Unhandled Exception der Wert eines formalen `OUT`- bzw. `IN/OUT`-Parameters nicht in den korrespondierenden Parameter kopiert. Bei der `NOCOPY`-Option wird hingegen bei Benutzung der Referenz der Wert des aktuellen Parameters verändert.

Erfolgt der Aufruf des Unterprogrammes per RPC (Remote Procedure Call), so wird der `NOCOPY` Hint ignoriert und die Parameter werden als Wert übergeben. Der `NOCOPY` Hint wird ebenfalls ignoriert, wenn z. B. bei der Parameterübergabe eine implizite Konvertierung stattfindet oder wenn bei der Parameterübergabe mit Records die Deklarationen zwischen korrespondierenden Feldern differieren.

PL/SQL Function Result Cache

Um die Anzahl der Funktionsaufrufe in PL/SQL zu reduzieren, können die so genannten deterministischen Funktionen verwendet werden. Der Nachteil der deterministischen Funktionen ist allerdings, dass diese nur innerhalb von SQL-Anweisungen und nicht innerhalb von PL/SQL gelten.

Darüber hinaus gelten die deterministischen Funktionen immer nur innerhalb einer Datenbank-Session. Zusätzlich wird pro SQL-Aufruf eine deterministische Funktion je unterschiedlichem Übergabeparameter mindestens einmal ausgeführt.

Ab Oracle 11g kann alternativ zu den deterministischen Funktionen der so genannte PL/SQL Function Result Cache verwendet werden. Bei dem Result Cache handelt es sich um einen session-übergreifenden Cache für Ergebnisse aus SQL-Abfragen und PL/SQL-Funktionen.

Der große Vorteil des Result Cache gegenüber den deterministischen Funktionen ist die Tatsache, dass er sowohl innerhalb von SQL- als auch in PL/SQL-Aufrufen gilt. Zusätzlich können zwischengespeicherte Funktionsergebnisse in einem Result Cache auch durch andere Anwender verwendet werden.

Der PL/SQL Function Result Cache sollte allerdings immer nur dann verwendet werden, wenn sich die zugrundeliegenden Daten einer PL/SQL-Funktion selten ändern. Dies beruht auf der Tatsache, dass bei jeder Änderung der zugrundeliegenden Daten der Result Cache invalidiert wird.

Zusätzlich sollte der PL/SQL Function Result Cache auch nicht verwendet werden, wenn einer PL/SQL-Funktion die aktuelle Tageszeit inklusive sekundengenauer Uhrzeit als Parameter übergeben wird. In diesem Fall würde jeder Aufruf der PL/SQL-Funktion zu einem neuen PL/SQL Function Result Cache-Eintrag führen und die zwischengespeicherten Ergebnisse würden kaum wieder verwendet werden.

Native Kompilierung

Mit Hilfe der so genannten Nativen Kompilierung kann die Ausführung von PL/SQL-Programmen deutlich beschleunigt werden. Bis Oracle 10g musste bei der Nutzung der Nativen Kompilierung ein externer C-Compiler genutzt werden und der durch die native Übersetzung erzeugte Maschinen-Code (Shared Library Unit oder Dynamic Link Library) wurde auf der Betriebssystem-Ebene abgelegt.

Ab Oracle 11g ist kein externer C-Compiler mehr nötig und der Maschinen-Code muss nicht mehr auf der Betriebssystem-Ebene abgelegt werden, sondern wird direkt für den Anwender transparent in der Datenbank gespeichert (`Tablespace SYSTEM`).

Die Performance-Verbesserungen bei der Ausführung von nativ übersetzten PL/SQL-Objekten kann dabei zwischen 30-99% betragen.

PL/SQL Inlining

Werden aufgrund der Modularisierung und Wiederverwendung viele Prozeduren und Funktionen verwendet, so entsteht bei deren Aufruf in Folge der Code-Sprünge ein Overhead im Programmfluss. Um diesen Overhead zu minimieren wurde ab Oracle 11g das so genannte PL/SQL Inlining eingeführt.

Mit PL/SQL Inlining werden Prozedur- oder Funktionsaufrufe während der Kompilierungszeit direkt durch den Inhalt der Prozedur bzw. Funktion ersetzt. Der Code wird optimiert, indem der Compiler Aufrufe von Unterprogrammen durch den Code des Unterprogramms ersetzt. Damit wird der Overhead bei einem Aufruf von Unterprogrammen verhindert.

PL/SQL Inlining kann zum einen zur Kompilierungszeit systemweit oder auf der Session-Ebene über den Initialisierungsparameter `PLSQL_OPTIMIZE_LEVEL` aktiviert werden. Alternativ dazu kann PL/SQL Inlining auch direkt im Code über die Compiler-Anweisung `PRAGMA INLINE` eingeschaltet werden.

PLSQL_WARNINGS

Um Hinweise auf Performance-Engpässe von PL/SQL-Programmen bereits zur Kompilierungszeit zu erhalten, können die so genannten PL/SQL-Warnungen verwendet werden. Durch diese können neben eventuellen Laufzeitfehlern auch potenzielle Performance-Probleme schon zum Zeitpunkt der Kompilierungidentifiziert werden.

SQL in PL/SQL

Neben dem PL/SQL-Tuning sollte ein klassisches SQL-Tuning natürlich immer mit einer hohen Priorität versehen werden. In den meisten Fällen verfügen vor allem SQL-Anweisungen über ein sehr großes Optimierungspotential.

Hierarchische Profiler

Um Performance-Engpässe in PL/SQL-Programmen innerhalb der Datenbank zu lokalisieren, kann ab Oracle 11g der sogenannte hierarchische Profiler (`DBMS_HPROF PL/SQL-Package`) verwendet werden. Der hierarchische Profiler geht dabei so vor, dass zunächst Daten bei der Ausführung einer PL/SQL-Einheit gesammelt werden, die anschließend ausgewertet werden können.

Im Gegensatz zum Vorgänger Profiler (`DBMS_PROFILER PL/SQL-Package`) ist bei der Verwendung des hierarchischen Profilers ein Umweg über das File-System notwendig. Der Grund dafür ist, dass die Datensammlung des hierarchischen Profilers über ein Directory-Objekt in eine Trace-Datei vorgenommen wird.

Fazit

Abschließend kann gesagt werden, dass die Optimierungsmöglichkeiten von PL/SQL-Programmen in einer Oracle Datenbank sehr vielfältig sind. Durch die Analyse der Performance-Schwachstellen und den Einsatz geeigneter Maßnahmen kann die Laufzeit von PL/SQL-Programmen signifikant verbessert werden. Damit kann sowohl die Akzeptanz der Anwender als auch der Erfolg einer Anwendung deutlich gesteigert werden.

Kontaktadresse:

Markus Fiegler
ORDIX AG
Westernmauer 12-16
D-33098 Paderborn

Telefon: +49 (0) 5251-10630
Fax: +49 (0) 180-1673490
E-Mail info@ordix.de
Internet: www.ordix.de