Parallele Programmierung in SQL und PL/SQL

Peter Bekiesch, Dierk Lenz
Herrmann & Lenz Services GmbH
Burscheid

Schlüsselworte
Programmierung, Parallele Ausführung, SQL, PL/SQL

Einleitung

Das Oracle RDBMS bietet innerhalb der Programmiersprachen SQL und PL/SQL diverse Möglichkeiten zur Parallelisierung von größeren Aufgaben. In diesem Vortrag werden die verschiedenen Verfahren erläutert und miteinander verglichen. Behandelt werden u.a.: Paralleles SELECT/DML, Parallelisierung über Scheduler Jobs, das PL/SQL-Package DBMS_PARALLEL_EXECUTE, Synchronisationsobjekte mittels DBMS_LOCK nutzen, sowie externe Verfahren, z.B. Multi-Threading von Datenbankoperationen in Java.

Wozu Parallelisierung?

Parallelisierung ist immer dann eine gute Idee, wenn größere Aufgaben zu erledigen sind. Vorausgesetzt, die entsprechenden Ressourcen stehen zur Verfügung, und die Aufgabe ist in *n* etwa gleich große Unteraufgaben aufteilbar, so kann man die Aufgabe grob in *1/n* der Zeit schaffen. Dies gilt z.B. im Datenbankbereich für Full Table Scans. Allerdings müssen sowohl die CPUs als auch das IO-Subsystem in der Lage sein, entsprechende Leistungen zu liefern.
Nun sollte man nicht auf die Idee kommen und alle Abfragen parallel ausführen, nach dem Motto:

„Warum soll ich noch Indizes pflegen, wenn ein paralleler Full Table Scan ebenfalls in Sekundenbruchteilen zurück ist?“

Der Unterschied: Ein guter indizierter Zugriff behindert keine gleichzeitigen Aktionen anderer Benutzer – ein paralleler Full Table Scan blockiert die Ressourcen der Maschine während der Laufzeit komplett. Daher sollte man Parallelisierung immer gezielt und mit Augenmaß einsetzen.

Welche Arten von Parallelisierung gibt es?

In der Enterprise Edition kann die Oracle Database diverse SQL-Befehle parallelisiert ausführen. Hierzu gehören SELECT, DML (INSERT, UPDATE und DELETE) sowie einige CREATE- und ALTER-Befehle (CREATE TABLE AS SELECT, CREATE INDEX, ALTER INDEX REBUILD, …). Bei der Instanzparametrierung hat sich in der Version 11.2 einiges geändert.

Die parallele Ausführung von SQL verlässt sich auf entsprechende Automatismen des Datenbankkerns. Weitere Flexibilität ergibt sich durch die Verwendung des ebenfalls mit 11.2 eingeführten Packages `dbms_parallel_execute`.

`dbms_parallel_execute` nutzt zur parallelen Abarbeitung der Subtasks den Datenbank-Scheduler. Die direkte Nutzung des Schedulers ist eine weitere Möglichkeit zur Erzeugung eigener paralleler Prozesse.

Weitere Möglichkeiten ergeben sich durch die Verwendung von Advanced Queues. Hierbei werden Nachrichten in eine Queue geschrieben, die von mehreren Prozessen abgeholt werden können. Versteht man die Nachrichten als Arbeitsaufträge, so kann durch die Anzahl der abholenden Prozesse eine parallel arbeitende Umgebung aufgebaut werden.

Im Folgenden gehen wir auf die einzelnen Möglichkeiten ein.

Parallele Ausführung in der Datenbank
Oracle stellt zur parallelen Ausführung eigene Parallel Execution Server (PX-Server) zur Verfügung. Diese Prozesse sind mit der Bezeichnung Pnnn versehen. Durch die Serverparameter `parallel_min_servers` und `parallel_max_servers` wird festgelegt, wie viele Prozesse minimal und maximal gestartet werden.

Wenn nun ein Befehl parallel ausgeführt werden soll, so legt er zunächst seinen Parallelisierungsgrad (DOP, *Degree Of Parallelism*) fest. Der zuständige Serverprozess, der den Befehl ohne Parallelisierung selbst ausführen würde, wird zum Query Coordinator und belegt für die Ausführung maximal 2 * DOP PX-Server. (Die Verwendung von doppelt so vielen PX-Servern wie DOP ist dadurch begründet, dass jede Phase eines SQL-Befehls mit DOP PX-Servern unterstützt wird. Es laufen aber bis zu zwei Phasen gleichzeitig ab, z.B. ein Full Table Scan und eine Sortierung des Ergebnisses.) Nachfolgend ein Beispiel mit 8 PX-Servern und zwei Serverprozessen, die je einen Befehl mit DOP 2 ausführen und somit je vier PX-Server allokieren:

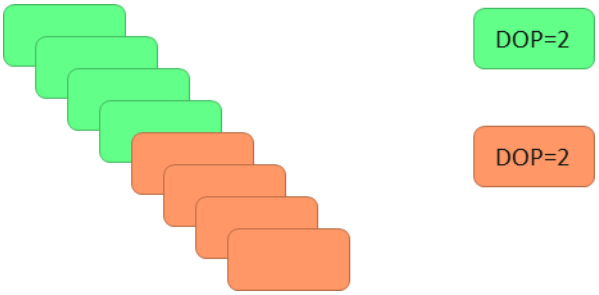

Abb. 1: PX-Server und Query Coordinators

Die Art der Parallelisierung ist seit vielen Oracle Version in dieser Art und Weise gegeben. In 11.2 sind jedoch folgende Neuerungen hinzu gekommen:

- Die automatische Bestimmung des DOP
- In-Memory-Parallelisierung
- Queueing von parallelen Befehlen

Um diese Eigenschaften zu steuern, ist ein neuer Serverparameter hinzugekommen: `parallel_degree_policy`.

In der Standardeinstellung `MANUAL` bleibt alles so, wie es vor Version 11.2 war: SQL-Befehle bestimmen ihren DOP ausschließlich selbst. Reichen die maximal verfügbaren PX-Server für eine angeforderte Parallelisierung nicht aus, so wird der DOP automatisch heruntergestuft, im Zweifelsfall bis zur seriellen Ausführung (`NOPARALLEL`).

Die Einstellung `LIMITED` führt zur eingeschränkten Nutzung des automatischen DOP. Hierbei werden Befehle, die entweder auf Objekte zugreifen, deren vorgegebener DOP `DEFAULT` ist, oder eine `PARALLEL`-Klausel ohne DOP verwenden mit einem von Oracle berechneten DOP versehen.

Steht der Wert von `parallel_degree_policy` auf `AUTO`, so sind alle oben genannten Eigenschaften aktiv:

- Oracle entscheidet eigenständig, dass SQL-Befehle parallelisiert werden. Immer wann, wenn die geschätzte Ausführungszeit für den Befehl über dem Wert von parallel_min_time_threshold (Standardwert: AUTO, entspricht 10 sec) liegt, wird parallelisiert und der DOP automatisch bestimmt.
- DB-Objekte werden in parallelen Ausführungsplänen grundsätzlich vollständig physisch gelesen (Full Table Scan, Full Index Scan). Für kleine und mittelgroße Objekte kann Oracle nun eigenständig entscheiden, diese im Buffer Cache abzulegen und somit schneller zur Verfügung zu stellen.
- Falls nicht genügend PX-Server vorhanden sind, wird der SQL-Befehl in einer Queue gehalten, bis er ausgeführt werden kann.

Voraussetzung für diese Eigenschaften ist die IO-Kalibrierung des IO-Subsystems mit `dbms_resource_manager.calibrate_io`.

Das Package dbms_parallel_execute
Das Package `dbms_parallel_execute` bietet folgende Funktionen:

- Eine Task wird mit `create_task` definiert.
- Mit Prozeduren wie `create_chunks_by_rowid` wird eine Tabelle in Portionen gleicher Größe (bzgl. Anzahl Datensätze oder Anzahl Blöcke) aufgeteilt.
- Die vorgenommene Aufteilung wird der Prozedur `run_task` zusammen mit einem SQL-Befehl (typischerweise einem `UPDATE`) übergeben. Hierdurch werden für die einzelnen Portionen Jobs abgeschickt, die naturgemäß parallel ausgeführt werden. Eine maximale Anzahl gleichzeitig laufender Jobs wird an `run_task` übergeben.

Damit steht ein leistungsfähiges API zur Parallelisierung von SQL-Befehlen zur Verfügung – nebenbei bemerkt: auch für die Standard Edition!

Nutzung des Schedulers
Die Idee, den Scheduler zur parallelen Abarbeitung von Teilaufgaben zu nutzen, ist nicht etwa erst durch das Package `dbms_parallel_execute` entstanden. Gerade durch das exakte Timing des (neuen) Datenbank-Schedulers gelingt es, auch im Bereich von Sekundenbruchteilen genau zu arbeiten. (Der „alte“ Scheduler hatte eine Startgenauigkeit von einer Minute und war für solche Aufgaben nicht sehr gut zu gebrauchen.)

Insbesondere auch durch die gute Nachvollziehbarkeit von Job-Ausführungen inklusive eventueller Fehlermeldungen eignet sich der Scheduler sehr gut für zeitkritische Aufgaben.

Advanced Queueing
Schließlich soll noch ein kurzer Blick auf Streams Advanced Queueing (so der vollständige Name) geworfen werden. Die Architektur unterscheidet sich zu der des Schedulers insofern, dass die Arbeitseinheiten nicht zentral verteilt, sondern von den parallel arbeitenden Programmen abgeholt (dequeued) werden. Das ist z.B. dann von Vorteil, wenn diese Programme auf einem anderen Rechner (bzw. diversen anderen Rechnern) laufen, oder wenn nicht im Detail überschaut werden kann, wie viel Aufwand in jeder einzelnen Arbeitseinheit steckt.

Advanced Queues können in der Oracle Datenbank mit einer Nachricht (Payload) versehen werden, deren Typ ein Oracle Objekttyp ist. Dadurch können Advanced Queues sehr flexibel genutzt werden – der abholende Prozess kann beim Abholen der Nachricht bereits alle wichtigen Informationen erhalten. Auch hier ist die gute Nachricht: Advanced Queues können mit der Standard Edition der Oracle Datenbank genutzt werden.

Synchronisationsobjekte mittels DBMS_LOCK nutzen
Synchronisationsobjekte wie Mutex oder Semaphoren sind sicherlich aus diversen anderen Programmiersprachen bekannt, jedoch werden diese selten mit PL/SQL in Verbindung gebracht. Bei intensivem Einsatz von parallelen Programmiermustern kommt man auch in PL/SQL nicht um den Einsatz von Synchronisationsobjekten herum.

Externe Verfahren zur Parallelisierung von Datenbankoperationen
Selbstverständlich können Datenbankoperationen auch von externen Programmen aus parallelisiert werden. Hierbei wird auf wichtige Aspekte und Rahmenbedingungen eingegangen. Abschließend werden exemplarisch einige Fallstricke in Java-Applikationen und dessen Vermeidung erörtert.

Kontaktadressen:

Peter Bekiesch
Herrmann & Lenz Solutions GmbH
Höhestraße 37
51399 Burscheid

Telefon: +49 2174 6712 0
Fax: +49 2174 6712 12
E-Mail peter.bekiesch@hl-solutions.de
Internet: www.hl-solutions.de

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