DBMS_PARALLEL_EXECUTE:
Wie man PL/SQL und SQL parallelisieren kann

Jan Ott
Trivadis AG
CH-8152 Glattbrugg (Zürich)

Schlüsselworte
DBMS_PARALLEL_EXECUTE, PL/SQL, SQL, Parallel, Performance

Einleitung
PL/SQL läuft nicht parallel. Daher kann es vorkommen, dass ein PL/SQL Programm läuft. Ein Prozessor voll ausgelastet ist und 23 Prozessoren am Warten sind. Dies kommt daher, dass PL/SQL ein Oracle Prozess ist. Ein Prozess kann nicht auf mehrere Prozessoren verteilt werden. Wir nutzen die Ressourcen nicht.

SQL läuft auch nur unter gewissen Voraussetzungen parallel. Doch selbst wenn wir es geschafft haben, dass das SQL in parallel abgearbeitet wird, kann es immer noch zu Problemen kommen, z.B. „Snapshot too old“.

Die Lösung ist eine solche Aufgabe in kleinere Teile zu zerlegen und die parallel abzuarbeiten.

Oracle stellt uns mit DBMS_PARALLEL_EXECUTE ein Mittel zur Verfügung um aus PL/SQL heraus Aufgaben parallel zu verarbeiten.

Dieser Vortag zeigt auf wie DBMS_PARALLEL_EXECUTE eingesetzt wird. Die Voraussetzungen und Einsatzbereiche werden diskutiert und erläutert. Limitierungen des Einsatzbereiches werden angegeben. Die relevanten Data Dictionary Views werden vorgestellt und erläutert. Das Ganze wird abgerundet mit Praxisbeispielen.

Problemstellung
Ein Kunde hatte folgendes Problem. Er muss einmal pro Woche eine Liste von ID‘s abarbeiten die in Tabelle T_LIST sind. Mit der ID aus der Liste wird ein PL/SQL Programm aufgerufen. Diese Programm, P1, führt verschiedenste Aufgaben aus, die mit einem COMMIT abgeschlossen werden.

Seine Lösung:
```
BEGIN
  FOR i IN (SELECT id FROM t_list)
  LOOP
    p1(i.id);
  END LOOP;
END;
/
```

Ergebnis:
Der PL/SQL Block läuft für Stunden. Das System war bis auf einen Prozessor nicht ausgelastet.

Wichtig:
Die Reihenfolge in der die ID‘s abgearbeitet werden ist frei. Ein Aufruf von P1 wird durch einen andern P1 Aufruf nicht beeinflusst.

Lösung:

Wir starten 20 Prozesse die alle einen Teil der Liste abarbeiten. Dies würde im Idealfall die Dauer auf einen Zwanzigstel reduzieren.

Dies kann man nun mittels Scripts machen oder DBMS_PARALLEL_EXECUTE verwenden.

Schauen wir uns nun Schritt für Schritt an wie das geht.

Voraussetzungen
Zuerst muss geprüft werden ob die nötigen Rechte vorhanden sind. Das Schema braucht

```
GRANT EXECUTE ON dbms_parallel TO scott;
GRANT CREATE JOB TO scott;
```

Haben wir diese Rechte können wir den 1. Schritt machen.

Schritt 1 - CREATE a TASK
Zuerst erstellen wir einen Task und geben ihm einen Namen. Dieser kann dann mittels DBMS_PARALLEL_EXECUTE abgearbeitet werden.

```
dbms_parallel_execute.create_task
    (task_name => 'mein_task'
    ,comment   => 'dies ist mein 1. task'
    );
```

Oracle stellt uns Data Dictionary Views zur Verfügung. Damit können wir sehen ob der Task angelegt wurde und was sein Status ist.

```
SELECT task_name, chunk_type
     , status, task_comment
  FROM user_parallel_execute_tasks;
```

Wir haben gesehen, dass der Task erstellt wurde nun geht es im Nächsten um das Erstellen der Arbeitspakete - CHUNKS.

Schritt 2 - CREATE CHUNKS
Es gibt verschieden Möglichkeiten diese Chunks zu erstellen.

- CREATE_CHUNKS_BY_ROWID:
  Teilt die Aufgabe mittels der Rowid in Blöcke - Chunks.

- CREATE_CHUNKS_BY_SQL:
  Teilt die Aufgabe mittels eines von unser gelieferten SQL in Blöcke auf.

- CREATE_CHUNKS_BY_COL:
  Teilt die Aufgabe mittels einer numerischen Spalte in Blöcke auf.

Sehen wir uns mal die Aufteilung nach Zeilen an.

```
dbms_parallel_execute.create_chunks_by_number_col
    (task_name    => 'mein_task'
    ,table_owner  => 'SCOTT'
    ,table_name   => 'T_LIST'
    ,table_column => 'ID'
    ,chunk_size   => 10
    );
```

Wir erstellen die Chunks „mein_task“ für das Schema „Scott“. Die Blöcke sollen auf der Tabelle T_LIST basieren. Zum Aufteilen nehmen wir die ID‘s und die Grösse der Blöcke soll 10 sein.

Prüfen wir nun ob die Chunks erstellt wurden.

```
SELECT task_name, chunk_type, status
     , table_owner, table_name, number_column
  FROM user_parallel_execute_tasks;

SELECT chunk_id, task_name, status
     , start_id, end_id
  FROM user_parallel_execute_chunks;
```

Schritt 3 - Run
Definieren des Statements das ausgeführt werden soll. Dies kann ein PL/SQL Block sein oder ein SQL Statement.

```
dbms_parallel_execute.run_task
  (task_name      => 'mein_task'
  ,sql_stmt       => 'BEGIN
                        FOR i IN :start_id..:end_id
                        LOOP
                          p1(i.id);
                        END LOOP;
                      END;
                     '
  ,language_flag  => dbms_sql.native
  ,parallel_level => 10
  );
```

Hier ist wichtig, dass in dem definierten Statement die zwei Variablen „start_id“ und „end_id“ enthalten sind. Hier wird auch angegeben wie viele parallele Prozesse laufen sollen. In unserem Fall haben wir 10 definiert.

Schritt 4 - Beobachten und Eingreifen
Oracle stellt Views zur Verfügung um den Fortschritt zu beobachten.

```
SELECT task_name, chunk_type, status, job_prefix, sql_stmt, parallel_level
  FROM user_parallel_execute_tasks;

SELECT chunk_id, status, start_id, end_id, job_name, start_ts, end_ts
  FROM user_parallel_execute_chunks
 WHERE task_name = 'task_test_case';
```

Wir könnten dies auch in PL/SQL kontrollieren mit:

- TASK_STATUS
  Damit kann man den Status abfragen

- STOP_TASK
  Stoppt den Task

- RESUME_TASK
  Startet den Task wieder, wenn er gestoppt wurde

- DROP_TASK
  Löscht den Task wieder aus der Liste, wenn er abgearbeitet ist.

Grösse der CHUNKS und die Anzahl der parallelen Prozesse
Das Starten der Jobs kostet Ressourcen. Daher sollte man die Chunks nicht zu klein wählen. Wie viele parallel Prozesse laufen, sollte auch vorsichtig gewählt werden.

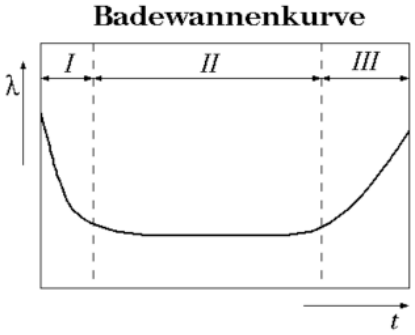

Abb. 1: Badewannenkurve

Diese Kurve bedeutet, dass wir durch Parallelität immer schneller werden mit höherer Parallelität, das ist der Bereich I. Dann kommen wir in einen Bereich der Sättigung II. Hier bringen mehr parallele Prozesse nichts mehr. Dann kommen wir noch in den III Bereich. Hier verkehrt sich das Ganze ins Gegenteil.

Praxisbeispiel
Leider mussten wir feststellen, dass bei einem Import BLOBs nicht parallel geladen werden können. Eine Möglichkeit ist den Export in Teile aufzuteilen damit kann auch der Import parallel gemacht werden.

Wir haben das auch mittels DBMS_PARALLEL_EXECUTE getestet. Dies war bedeutend schneller zudem mit weniger Aufwand verbunden.

Für mehr Praxisbeispiele besuchen Sie den Vortrag.

Kontaktadresse:
Jan Ott
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg (Zürich)

Telefon: +41 (0) 79 205 89 66
Fax: +41 (0) 44 808 70 21
E-Mail jan.ott@trivadis.com
Internet: www.trivadis.com