



DOAG Regionaltreffen München 2008

Let's do it in parallel

Impressum

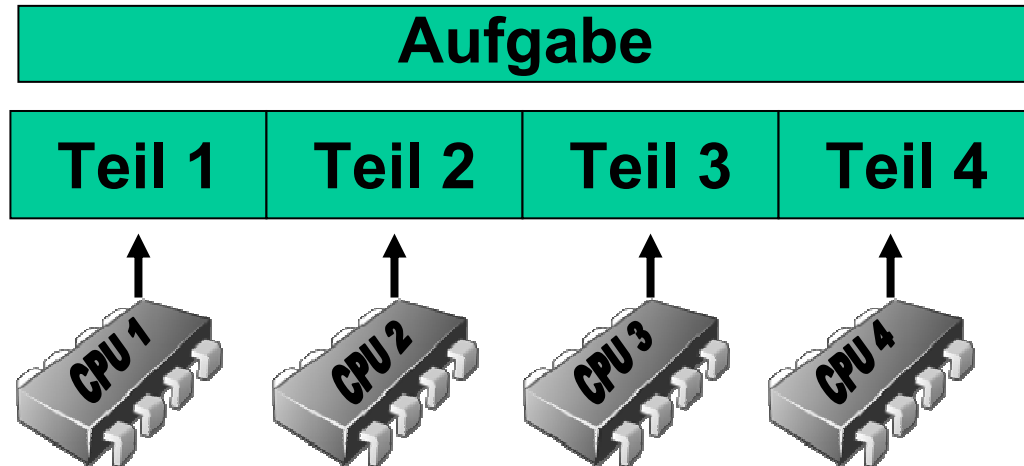
- ◆ Oracle Schulung (SQL, DBA, PL/SQL, Security, 11g, Tuning, Backup & Recovery u.v.m.)
- ◆ Oracle Consulting & Support
- ◆ Oracle Entwicklung & Lizenzvertrieb
- ◆ Marco Patzwahl
MuniQSoft GmbH
Grünwalder Weg 13 a
D-82008 Unterhaching b. München
- ◆ Telefon: +49 (0)89 679090-40
E-Mail: m.patzwahl@muniqsoft.de
Internet: www.muniqsoft.de



www.plsql.de

Einführung

- ◆ Oracle unterstützt seit Version 7 die parallele Verarbeitung von verschiedenen Befehlen.
- ◆ Die genaue Liste der möglichen Operationen ist von der Version abhängig und sollte im Handbuch *Administrators Guide* oder *Data Warehouse Guide* (Kapitel 25) verifiziert werden.



Grundlagen

- ◆ Bei paralleler Ausführung eines Statements verteilt ein Dispatcher-Prozess, der sog. “Parallel Execution Coordinator”, die Aufgabe an verschiedene Prozesse, die “Parallel Execution Servers.
- ◆ Der “Parallel Execution Coordinator” gliedert das Segment (Index, Tabelle, Partitionierte Tabelle oder Index, Cluster) in sog. “Granules” (bestimmte Anzahl physischer Blöcke).
- ◆ Nach Verarbeitung eines Granules durch den “Execution Server”, bekommt er vom “Execution Coordinator” das nächste Granule.
- ◆ Am Ende werden alle Teilresultate durch den “Execution Coordinator” zum Gesamtergebn zusammengesetzt.

Grundlagen

- ◆ Die einer Operation zugeordnete Anzahl der “Execution Servers” nennt man DOP (Degree of parallelism).
- ◆ Besteht ein Statement aus mehreren Operationen (z.B. Lesen der Daten aus der Tabelle und Sortieren der Daten), so spricht man von „inter-operation parallelism“.
- ◆ In diesem Fall kann die Anzahl der „Execution Servers“ doppelt so gross wie der DOP sein.
- ◆ Die PARALLEL-Clause bestimmt den Default-DOP einer Tabelle für anschließend ausgeführte SELECT und DML Statements.
- ◆ Der PARALLEL-Hint setzt sich über eine PARALLEL-Clause hinweg.

Allgemeines

◆ Vorteile:

- ▶ u.U. höhere Ausführungsgeschwindigkeit

◆ Nachteile:

- ▶ Etwas höherer Verwaltungsaufwand
- ▶ Jede Aufgabe ist nicht beliebig skalierbar
- ▶ Nur beim Cost Based Optimizer verfügbar
- ▶ Ausführungsplan kann sich ändern!

◆ Hinweis:

- ▶ Die ideale Anzahl der parallelen Prozesse beträgt $N-1$ ($N = \text{CPU Anzahl}$)
- ▶ Dazu sollten jedoch auch die Tablespaces aus jeweils $N-1$ Dateien (auf entsprechenden eigenen Platten) bestehen

Ideale Faktoren

- ◆ **Damit die Parallel Query Option benutzt werden kann, sind folgende Punkte notwendig:**
 - ▶ **Oracle Enterprise Edition**
 - ▶ **Sie verfügen über eine Multiprozessormaschine 😊**
 - ▶ **Für paralleles Schreiben:**
 - **Es gibt mehrere Platten auf die die Tablespace-dateien verteilt wurden**

- ◆ **Folgende Parameter wurden gesetzt**
 - ▶ **large_pool_size>0**
 - ▶ **parallel_max_servers>=2**
 - ▶ **Segment-Parameter DEGREE auf Objektebene oder Hints wurden gesetzt**

Ideale Faktoren

◆ Faktoren, die sich günstig auswirken:

- ▶ Die Tabelle hat mehr als ein Extent (Mehr Extents als Dateien des Tablespace)
- ▶ Data Warehouse Applikation (kein OLTP)
- ▶ Tabelle ist groß (z.B. >1GB)
- ▶ Tabelle oder Index ist partitioniert

Wie viele CPU's sind vorhanden?

◆ Windows

- ▶ `echo %NUMBER_OF_PROCESSORS%`

◆ Linux

- ▶ `cat /proc/cpuinfo|grep processor|wc -l`
- ▶ `top`

◆ Solaris

- ▶ `psrinfo -v|grep "Status of processor"|wc -l`

◆ IBM-AIX

- ▶ `lsdev -C|grep Process|wc -l`

◆ HP-UX

- ▶ `ioscan -C processor | grep processor | wc -l`

Was kann parallelisiert werden?

◆ Allgemeine Operationen

- ▶ **Table Scans (z.B. Full Table Scan) (Bis 9.2 nur für partitionierte Tabellen)**
- ▶ **Full Index Scans und Index Partition Scans**
- ▶ **Hash, Nested Loop und Sort Merge Join**
- ▶ **ORDER BY**
- ▶ **SELECT DISTINCT**
- ▶ **UNION und UNION ALL**
- ▶ **Ein PARALLEL-Hint auf eine View wird auf alle in der View referenzierten Tabellen angewendet.**

Was kann parallelisiert werden?

- ▶ **Backup/Restore/Recovery (z.B. mittels RMAN über bis zu 255 Channels)**
- ▶ **Statistikerzeugung mittels dbms_stats (Nicht ANALYZE)**
- ▶ **Crash Recovery**
- ▶ **SQL*Loader**
- ▶ **Ab Version 10g: Datapump Export/Import**

Was kann parallelisiert werden?

◆ DML

- ▶ UPDATE auf part. Tabelle
- ▶ INSERT ... SELECT ...
- ▶ DELETE auf part. Tabelle
- ▶ MERGE

◆ DDL (Normale Tabellen und Indizes)

- ▶ CREATE INDEX
- ▶ CREATE TABLE AS SELECT
- ▶ ALTER INDEX REBUILD [ONLINE]
- ▶ ALTER TABLE MOVE

Was kann parallelisiert werden?

- ◆ **DDL (partitionierte Tabellen und Indizes)**
 - ▶ **CREATE TABLE...AS SELECT**
 - ▶ **ALTER TABLE...MOVE PARTITION**
 - ▶ **ALTER TABLE...SPLIT PARTITION**
 - ▶ **CREATE INDEX**
 - ▶ **ALTER INDEX...REBUILD PARTITION**
 - ▶ **ALTER INDEX...SPLIT PARTITION**
- ◆ **Hinweis: Parallelisierung kommt zum Einsatz, wenn mehrere Partitionen betroffen sind.**

Was kann nicht parallelisiert werden?

- ◆ **Installation der DB (catalog, catproc, ...)**
- ◆ **Alte Exports/Imports (exp/imp)**
- ◆ **Sequentielle Aufgaben**
- ◆ **Große Tablespaces (z.B. Bigfile) anlegen**
- ◆ **Tabellen/Tablespaces/User dropfen**
- ◆ **Tabellenabfragen mit Bitmap Indizes**

Was kann nicht parallelisiert werden?

- ◆ **INSERT Statements mit der VALUES Klausel**
- ◆ **Updates/Deletes auf nicht partitionierten Tabellen**
- ◆ **Tabellen dürfen keine Objekttyp / LOB Spalten besitzen**
- ◆ **Triggers werden für parallele DML Operationen nicht unterstützt**
- ◆ **Replication wird für parallele DML Operationen nicht unterstützt.**
- ◆ **Einschränkungen bzgl. der Ref. Integrität stehen in Metalink Dokument: Note:201978.1**

Parameter parallel_automatic_tuning

- ◆ **parallel_automatic_tuning = true**
- ◆ **Dieser setzt bereits weitere Parameter für die Parallelisierung auf sinnvolle Werte. So wird z.B. auch der Parallelisierungsgrad der Tabelle von Oracle automatisch bestimmt. Jedoch muss die Tabelle trotzdem mit Attribut PARALLEL gekennzeichnet werden.**
- ◆ **Ab Version 10g ist dieser Parameter nicht mehr verfügbar !**
- ◆ **Folgende Parameter werden gesetzt:**
 - ▶ **parallel_execution_message_size =4K (Default 2k)**
 - ▶ **parallel_adaptive_multi_user=true**
 - ▶ **large_pool_size= Wert wird automatisch passend gesetzt**
 - ▶ **processes= Wenn der Wert kleiner als parallel_max_servers ist, wird er erhöht**
 - ▶ **parallel_max_servers= wenn parallel_adaptive_multi_user=true
(cpus * parallel_threads_per_cpu * 4 * 5) sonst
(cpus * parallel_threads_per_cpu * 4 * 8)**

Wichtige Initialisierungsparameter

◆ **cpu_count**

- ▶ **Wird von Oracle - basierend auf der aktuellen CPU-Anzahl - automatisch gesetzt**

◆ **parallel_min_servers**

- ▶ **Minimale Anzahl der Parallelen Slave Prozesse pro Instanz**
- ▶ **Default=0**

◆ **parallel_max_servers**

- ▶ **Maximale Anzahl der Parallelen Slave Prozesse pro Instanz**
- ▶ **Default=(CPU_COUNT x PARALLEL_THREADS_PER_CPU x (2 Falls PGA_AGGREGATE_TARGET > 0; sonst 1) x 5)**
- ▶ **Vorschlag: 10*CPU**

Einstellungen auf Session-Ebene

◆ Syntax:

▶ `ALTER SESSION ENABLE | DISABLE | FORCE
PARALLEL DML | DDL | QUERY [PARALLEL <n>];`

◆ Parallel Query auf Session-Ebene einschalten:

▶ `ALTER SESSION FORCE PARALLEL QUERY;`

◆ Vier parallele Prozesse bei Selects zulassen:

▶ `ALTER SESSION FORCE PARALLEL QUERY PARALLEL 4;`

◆ Acht parallele Prozesse bei DML Befehlen zulassen:

▶ `ALTER SESSION FORCE PARALLEL DML PARALLEL 8;`

Einstellungen auf Session-Ebene

◆ Anmerkungen:

- ▶ **Session-Ebene überschreibt die Einstellungen der INIT.ORA Parameter.**
- ▶ **Hints überschreiben die Einstellung der Session.**
- ▶ **ENABLE benutzt Parallelisierung nur bei Einsatz von Hints bzw. wenn die Tabelle dafür vorbereitet wurde.**
- ▶ **DISABLE schaltet die parallele Verarbeitung in der aktuellen Session aus. Jedoch haben Hints eine höhere Prio!!!**
- ▶ **FORCE verwendet im Zweifelsfall eine Default-Parallelisierung.**

Tabellenebene

◆ Auf Tabellenebene einschalten:

▶ `CREATE TABLE <tab> (...) PARALLEL;`

Oder nachträglich

▶ `ALTER TABLE <tab> PARALLEL;`

◆ oder expliziter Parallelisierungsgrad 4

▶ `CREATE TABLE <tab> (...) PARALLEL 4;`

◆ Parallelisierung ausschalten mittels:

▶ `ALTER TABLE <tab> NOPARALLEL;`

◆ Hinweis: Wenn kein Integer-Wert für PARALLEL angegeben wird, benutzt Oracle einen eigens errechneten Default-Wert.

Allgemeines zu Hints

◆ Mit Hints können Sie (meistens 😊) den Ausführungsplan eines Statements beeinflussen

◆ Syntax:

▶ `SELECT /*+ HINT */ * FROM ...`

▶ Sollten Sie in Ihren Statements Aliasnamen für Tabellen verwenden, muss dieser Aliasname auch im Hint verwendet werden

▶ `SELECT /*+ PARALLEL(b,4) */ * FROM big b.....;`

Kurzübersicht der Hints für Parallelisierung

◆ Parallelisierung für SQL-Statement ein-, ausschalten:

▶ `/*+ PARALLEL(<tab>,<degree>) */`

▶ `/*+ NOPARALLEL(<tab>) */`

◆ Parallelisierung für Index-Nutzung ein-, ausschalten:

▶ `/*+ PARALLEL_INDEX(<index>,<degree>) */`

▶ `/*+ NOPARALLEL_INDEX(<index>) */`

◆ Parallelisierung bei Joins

▶ `/*+ PQ_DISTRIBUTE(<table> <para1>,<para2>) */`

FAST INSERT

- ◆ Bei einem FAST INSERT wird die Ergebnismenge, die in eine Tabelle eingetragen werden soll, aus einem SELECT ermittelt. Die Daten werden aus Performancegründen hinter die High-Watermark gesetzt.

```
▶ INSERT /*+ PARALLEL(EMP_COPY,4) */ INTO emp_copy
  SELECT /*+ PARALLEL(EMP,4) */ *
  FROM emp;
```

- ◆ Anmerkung: Die serielle Variante lautet:

```
▶ INSERT /*+ APPEND */ INTO emp_copy
  SELECT * FROM emp;
```

- ◆ Mittels Hints werden die Einstellungen der Segmente überschrieben.
- ◆ Nach jedem parallelen DML Befehl muss ein Commit erfolgen.

Parallel Delete für part. Tabellen

◆ Parallele Delete durchführen:

- ▶ `DELETE /*+ parallel(<tab>,<n>) */ FROM <tab>;`
- ▶ `COMMIT;`

◆ Beispiel:

- ▶ `DELETE /*+ parallel(big_emp,8) */
FROM big_tab
WHERE id<1000000;`
- ▶ `COMMIT;`

Parallel Update für part. Tabellen

◆ Paralleler Update:

- ▶ `UPDATE /*+ parallel(<tab>,<n>) */ <tab>
SET <spalte>=<wert>;`

◆ Beispiel:

- ▶ `UPDATE /*+ parallel(big_tab,8) */ big_tab
SET status='OK';`
- ▶ `COMMIT;`

Hints für Parallelisierung

- ◆ Durch die Benutzung von Parallel Hints kann es passieren, dass der Optimizer einen Full Table Scan einem Index Scan vorzieht.

- ▶ `SELECT /*+ FULL(emp) PARALLEL(emp, 4) */ emp
FROM emp;`

- ▶ `SELECT /*+ PARALLEL(emp, 4) */ emp
FROM emp;`

- ◆ Dies kann mit einem Parameter manipuliert werden

- ▶ `optimizer_index_cost_adj = 100 /* Default */`

- ▶ Je kleiner der Wert, desto eher wird ein Index bevorzugt

- ▶ `optimizer_index_cost_adj = 10`

Statement-Ebene

◆ Auf Statement-Ebene einschalten:

- ▶ `SELECT /*+ PARALLEL(<tab>) */ spalte1,spalte2, FROM <tab>;`
- ▶ `SELECT /*+ PARALLEL(<tab>,4) */ spalte1,spalte2, ... FROM <tab>;`
- ▶ `SELECT /*+ ALL_ROWS PARALLEL(<tab>,8) */ spalte1,spalte2 FROM <tab>;`

◆ Auf Statement-Ebene ausschalten:

- ▶ `SELECT /*+ NO_PARALLEL(<tab>) */ spalte1,spalte2, FROM <tab>;`

◆ Anmerkung:

- ▶ **Statement-Ebene überschreibt die Einstellungen der Session-Ebene.**

Parallele Indexerzeugung

◆ Parallelisierung für Indizes einschalten:

- ▶ `CREATE INDEX <index> ON <tab>(<spalte>)
TABLESPACE <tbs>
PARALLEL (DEGREE <n>) [NOLOGGING];`
- ▶ `ALTER INDEX <index> REBUILD [ONLINE]
PARALLEL (DEGREE <n>) [NOLOGGING];`

◆ Ausschalten der Parallelisierung:

- ▶ `CREATE INDEX <index> ON <tab>(<spalte>)
NOPARALLEL; -- Bei Erzeugung ist das Default!`
- ▶ `ALTER INDEX <index> NOPARALLEL;`

Für Rollback-Segmente

- ◆ Regelt die Anzahl der parallelen Rollback-Prozesse bei einem Crash Recovery.
- ◆ Einschalten:
 - ▶ `ALTER SYSTEM SET FAST_START_PARALLEL_ROLLBACK = low; -- 2* CPU Anzahl`
 - ▶ `ALTER SYSTEM SET FAST_START_PARALLEL_ROLLBACK = high; -- 4* CPU Anzahl`
- ◆ Ausschalten:
 - ▶ `ALTER SYSTEM SET FAST_START_PARALLEL_ROLLBACK = false;`

Recovery

◆ Paralleles Recovery einschalten:

- ▶ **RECOVERY_PARALLELISM = 0 | 1 | n**
Anzahl der Prozesse, die bei einem Recovery beteiligt sind
0,1 entspricht einem Prozess
- ▶ Damit können mehrere Datendateien parallel einem Recovery unterzogen werden.
- ▶ Dies gilt aber nur für ein Crash Recovery beim Start der DB-Instanz.
- ▶ Alternative für sonstiges Recovery: Mehrere Kanäle beim RMAN für Recovery verwenden.

Parallelisierung beim RMAN

◆ Automatische Parallelisierung (1-255 Kanäle bei EE)

▶ `CONFIGURE DEVICE TYPE DISK PARALLELISM 8;`

◆ Manuelle Parallelisierung (1-255 Kanäle bei EE)

▶ `RUN`

```
{ ALLOCATE CHANNEL ch1 DEVICE TYPE disk;  
  ALLOCATE CHANNEL ch2 DEVICE TYPE disk;  
  ALLOCATE CHANNEL ch3 DEVICE TYPE disk;  
  ALLOCATE CHANNEL ch4 DEVICE TYPE disk;  
  BACKUP DATAFILE 1,2,3,4;  
}
```

Paralleler Export/Import/Loader

- ◆ **Leider ist eine automatische Parallelisierung bei Export/Import bis 10g nicht möglich. Hier kann nur mit parallelen Export/Import Sessions gearbeitet werden, die eine Teilmenge der Schema/Tabellen bearbeiten.**
- ◆ **Beim SQL*Loader können jedoch mehrere Datendateien angegeben werden, die dann parallel bearbeitet werden.**
 - ▶ `SQLLDR scott/tiger CONTROL=data1ctl
DIRECT=TRUE PARALLEL=TRUE
SQLLDR scott/tiger CONTROL=data2ctl
DIRECT=TRUE PARALLEL=TRUE
SQLLDR scott/tiger CONTROL=data3ctl
DIRECT=TRUE PARALLEL=TRUE`

EXPDP/IMPDP Parameter in 10g

◆ PARALLEL = <n>

- ▶ DEFAULT = 1,
- ▶ Wert sollte der Anzahl der zu erzeugenden Dateien entsprechen
- ▶ Kann nur für die Enterprise Edition (EE) benutzt werden

◆ Beispiel:

- ▶ `expdp userid=system/manager
DUMPFILE=scott%U.dmp DIRECTORY=expimp_dir
TABLES=scott.emp
PARALLEL=4`
- ▶ `impdp scott/tiger@o10g DUMPFILE=scott%U.dmp
DIRECTORY=expimp_dir TABLES=scott.big_tab
PARALLEL=4`

Parallele Statistiken

◆ Ein **ANALYZE TABLE** oder **ANALYZE INDEX** Kommando ist nicht parallelisierbar!

◆ Verwenden Sie stattdessen **dbms_stats**

```
▶ BEGIN dbms_stats.gather_schema_stats(  
  ownname=>'SCOTT',  
  ...  
  degree=>4); END;
```

```
▶ BEGIN dbms_stats.gather_table_stats(  
  ownname=>'SCOTT',  
  tabname='BIG_EMP'  
  ...  
  degree=>4); END;
```

Prüfung der Parallelisierung

◆ Prüfen Sie den Ausführungsplan

- ▶ Dort müssten Einträge stehen wie `/*+ ROWID(A1) */` oder `PX Coordinator`. Dieser Text steht nicht im Original-SQL-Befehl
- ▶ In der Tabelle `plan_table` existiert eine Spalte `other_tag`. Wenn diese leer ist, wurde seriell verarbeitet.
- ▶ Mit dem Skript `?/rdbms/admin/utlxplp.sql` kann der Ausführungsplan für die Parallelausführung besser angezeigt werden.

Prüfung der Parallelisierung

- ◆ **Setzen Sie in der Session, die gerade den parallelen Befehl ausgeführt hat, den folgenden SQL-Befehl ab:**

- ▶

```
SELECT * FROM v$sqlsesstat
WHERE statistic in ('Queries Parallelized',
                   'Allocation Height');
```
- ▶

```
STATISTIC                LAST_QUERY_SESSION_TOTAL
-----
Queries Parallelized      1                1
```

- ◆ **Welche Session hat Parallelisierung aktiviert?**

- ▶

```
SELECT username,sid,serial#,
pdml_status,pddl_status,pq_status
FROM v$session;
```

Prüfung der Parallelisierung

◆ Während die Abfrage läuft:

- ▶ `SELECT slave_name,status, cpu_secs_total
FROM v$pdg_slave;`
- ▶ P000 BUSY 14
- ▶ ...
- ▶ P007 BUSY 4

◆ Prüfung, welche Default Parallelisierung die Tabelle/ der Index benutzt:

- ▶ `SELECT owner,table_name,degree
FROM dba_tables;`
- ▶ `SELECT owner,index_name,degree
FROM dba_indexes;`

Ausführungspläne

Serieller Ausführungsplan

Id	Operation	Name	Rows	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		1	24318 (2)	00:04:52		
1	SORT AGGREGATE		1				
2	PARTITION RANGE ALL		8010K	24318 (2)	00:04:52	1	4
3	TABLE ACCESS FULL	BIG_PART	8010K	24318 (2)	00:04:52	1	4

Paralleler Ausführungsplan

Id	Operation	Name	Rows	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		1	6724 (1)	00:01:21					
1	SORT AGGREGATE		1							
2	PX COORDINATOR									
3	PX SEND QC (RANDOM)	:TQ10000	1					Q1,00	P->S	QC (RAND)
4	SORT AGGREGATE		1					Q1,00	PCWP	
5	PX BLOCK ITERATOR		8010K	6724 (1)	00:01:21	1	4	Q1,00	PCWC	
6	TABLE ACCESS FULL	BIG_PART	8010K	6724 (1)	00:01:21	1	4	Q1,00	PCWP	

Performance Messungen

- ◆ **Hardware:HP ProLiant ML370 G5, RAM: 20 GB**
- ◆ **Processors : 2 x Quadcore 2,333 MHz Intel(R) Xeon(R) CPU**
- ◆ **Platten: 6 x 146 GB SAS Platten im Raid 5**
- ◆ **SLES 10 in einer VMWare mit 2GB Ram und 4 CPU**
- ◆ **Oracle 10.2**
 - ▶ **sga_target= 305 MB**
 - ▶ **pga_aggregate_target= 100 MB**
- ◆ **Beispiel-Tabelle mit 8 Mio Zeilen, Zeilenlänge 93B => Tabellengröße: ca. 710MB**
- ◆ **Testszenario: Jede Messung läuft 4 Mal, vor jeder Messung wird ein STARTUP FORCE durchgeführt.**
- ◆ **Mittelwert von 4 Messungen**

Performance Messungen

	NO PARALLEL	PARALLEL 2	PARALLEL 4
SELECT sum(object_id) FROM big;	Z: 21s K: 24319	Z: 7,7s K: 13449	Z: 7s K: 6725
exec dbms_stats.gather_table_stats...	Z: 29s	Z: 23s	Z: 20,5s
CREATE INDEX big_ix ON big(object_name)	Z: 59,5s K: 32852	Z: 48s K: 17713	Z: 35s K: 8856
UPDATE big SET status=status;	Z: 4m 6s K: 24456	Z: 4m 20s K: 13525	Z: 5m 30s K: 6762
DELETE FROM big WHERE object_type='PACKAGE'; --134000	Z: 40s K: 24422	Z: 28s K: 13506	Z: 26s K: 6753

Z: Zeit K: Kosten im Ausführungsplan

Performance Messungen

◆ Tuning:

- ▶ Log Buffer von 7 MB auf 3 MB
- ▶ 3 Redologs von 50 MB auf 250 MB

	NO PARALLEL	PARALLEL 2	PARALLEL 4
UPDATE big SET status=status;	Z: 2m 40s	Z: 2m 50s	Z: 2m 50s
DELETE FROM big WHERE object_type='PACKAGE'; --134000	Z: 17s	Z: 13s	Z: 19s
CREATE TABLE big2 AS SELECT * FROM big;	Z: 23s	Z: 18s	Z: 16s
CREATE INDEX big_ix ON big(object_name)	Z: 56s	Z: 43s	Z: 31s

Performance Messungen (part. Tabelle)

- ◆ Bei folgenden Tests wurde eine partitionierte Tabelle mit 4 Partitionen verwendet:

	NO PARALLEL	PARALLEL 2	PARALLEL 4
EXEC dbms_stats.gather_table_stats	Z: 51s	Z: 31s	Z: 25s
SELECT count(*) FROM big_part	Z: 11s	Z: 9s	Z: 8s
DELETE FROM big_part WHERE mod(object_id,1000)=0;	Z: 14s	Z: 15s	Z: 11s
UPDATE big_part SET c=c;	Z: 2m27s	Z: 2m12s	Z: 2m55s

Performance Messung (part. Tabelle)

- ◆ Der Speicher für die SGA wurde von 300 MB auf 1 GB und `pga_aggregate_target` von 100 MB auf 300 MB erhöht

	NO PARALLEL	PARALLEL 2	PARALLEL 4
EXEC <code>dbms_stats.gather_table_stats</code>	Z: 31s	Z: 33s	Z: 27s
<code>SELECT count(*) FROM big_part</code>	Z: 12s	Z: 11s	Z: 10s
<code>DELETE FROM big_part WHERE mod(object_id,1000)=0;</code>	Z: 15s	Z: 15s	Z: 11s
<code>UPDATE big_part SET c=c;</code>	Z: 2m 45s	Z: 2m 55s	Z: 2m 55s
<code>*UPDATE big_part SET c='VALID';</code>	Z: 2m 48s	Z: 2m 45s	Z: 2m 47s

* Messung wurde nach dem Vortrag wiederholt, da sie von anwesender Oracle Mitarbeiterin angezweifelt wurde ☺

Impressum

- ◆ **Oracle Schulung (SQL, DBA, PL/SQL, Security,...)**
- ◆ **Oracle Consulting & Support**
- ◆ **Oracle Entwicklung & Lizenzvertrieb**
- ◆ **Marco Patzwahl**
MuniQSoft GmbH
Grünwalder Weg 13 a
D-82008 Unterhaching b. München
- ◆ **Telefon: +49 (0)89 679090-40**
Fax: +49 (0)89 679090-50
E-Mail: m.patzwahl@muniqsoft.de
Internet: www.muniqsoft.de

