Performancetuning in Oracle 11g

Inhalt

- Ø SQL Tuning in Oracle 9*i* und 10*g*
- Ø SQL Tuning in Oracle 11*g* - Überblick
- Ø SQL Plan Management
- Ø SQL Performance Analyzer
- Ø Automatic SQL Tuning
- Ø Adaptive Cursor Sharing
- Ø Wechselwirkung verschiedener Komponenten
  - Adaptive Cursor Sharing und CURSOR_SHARING
  - Adaptive Cursor Sharing und SQL Plan Baselines
- Ø Full Table Scans in Oracle 11*g*
- Ø Alternative Tools

Performance Tuning in Oracle 9*i* und 10*g*

Oracle 9*i*

- Statspack
- Extended SQL Tuning (offiziell nicht supportet)

Oracle 10*g*

- Automatic Database Diagnostic Monitor (ADDM)
- Automatic Workload Repository (AWR)
- Active Session History (ASH)
- SQL Tuning Advisor (STA)
- SQL Tuning Sets (STS)
- SQL Access Advisor (SAA)
- Diverse neue Metriken
- Package DBMS_MONITOR

Performance Tuning in Oracle 11*g*

- Tools zu Diagnose und Tuning wie der *SQL Performance Analyzer* stehen neben neu im Kern der Datenbank implementierten Funktionen, die ohne spezielle Konfiguration direkt einen Performancegewinn versprechen, wie z.B. neue Verhaltensweisen des Optimizer oder Adaptive Cursor Sharing.
- Diagnostikwerkzeuge werden ergänzt durch solche, die eine SQL-Optimierung ermöglichen.
- Weiterhin kann man zwischen *proaktiv* und *reaktiv* unterscheiden. Bestimmte Tools erkennen Performanceengpässe selbsttätig, bevor eventuell schwerwiegende Probleme auftreten, die die Nutzerzufriedenheit insgesamt beeinträchtigen. Andere kommen zum Einsatz, wenn erhebliche Degradationen der Antwortzeit aufgetreten sind.
- Schließlich muß zwischen *manuellem* und *automatischem* Vorgehen differenziert werden. Einerseits ist nach wie vor das Handeln des DBA gefordert, andererseits gibt es zunehmend Automatismen, entsprechend der Philosophie von Oracle, den DBA von Routineaufgaben zu entlasten.

SQL Plan Management

Ziel: Verhindern, daß sich Ausführungspläne im Betrieb verschlechtern

Prinzipien:

- Es wird eine Historie der Ausführungspläne geführt.
- SQL muß mehrfach ausgeführt werden (Anweisungs-Log), um Bestandteil einer *SQL Plan Baseline* zu werden.
- Planänderungen werden geprüft und eventuell als Bestandteil der SQL Plan Baseline akzeptiert.
- Die Planprüfung erfolgt unmittelbar und im Rahmen des *Automatischen SQL Tuning*.
- Nur vergleichbare oder bessere Pläne werden verwendet.
- Planhistorie, Anweisungs-Log und SQL Plan Baselines sind Bestandteil der *SQL Management Base* im SYSAUX-Tablespace.
- Ersetzt teilweise Plansteuerungstechniken wie *Stored Outlines* oder *SQL Profile*

Quellen für SQL Plan Baselines:

- Vordefinierte SQL Tuning Sets für kritische SQL
- Automatisches Generieren von SQL Plan Baselines
- Manuelles Integrieren von Plänen aus dem Cursor Cache

SQL Plan Management – Verwaltung

Planstati:

- **Enabled**: Plan kann vom Optimizer verwendet werden.
- **Accepted**: Plan ist Bestandteil der Baseline.
- **Fixed**: Plan kann sich nicht ändern (analog zu Stored Outlines).

Parameter:

- `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES`
- `OPTIMIZER_USE_SQL_PLAN_BASELINES`

DBMS_SPM:

- `ALTER_SQL_PLAN_BASELINE:`
- `CONFIGURE`
- `DROP_SQL_PLAN_BASELINE`
- `EVOLVE_SQL_PLAN_BASELINE`
- `LOAD_PLANS_FROM_CURSOR_CACHE`
- `LOAD_PLANS_FROM_SQLSET`
- `CREATE_STGTAB_BASELINE`
- `PACK_STGTAB_BASELINE`
- `UNPACK_STGTAB_BASELINE`

Planhistorie:

- Aufbewahrung 53 Wochen
- SQL Management Base standardmäßig 10% des SYSAUX Tablespace
- Nicht genutzte Pläne werden gelöscht.
- Konfigurierbar

Planentwicklung

Durchführung: Manuell oder automatisch

Automatische Planentwicklung:

- `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES=TRUE`
- Häufig ausgeführte SQL wird erkannt.
- Der Plan mit den besten Kosten wird der SQL Plan Baseline hinzugefügt
- Wird eine SQL-Anweisung neu geparst, wird in der Baseline nach einem übereinstimmenden Plan gesucht.
- Ein neuer Ausführungsplan wird der Plan-Historie zunächst als nichtakzeptierter Plan hinzugefügt
- Die Performance des Plans wird geprüft und mit der SQL Plan Baseline verglichen.
- Ein besserer oder wenigstens gleich performanter Plan wird der SQL Plan Baseline hinzugefügt.
- **Achtung**: Stored Outlines verhindern dieses Verhalten. Es gibt keine direkte Migration von Stored Outlines zur SQL Plan Baseline.

Manuelle Planentwicklung:

- PL/SQL-Prozedur `DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE`
- Analyse problematischer SQL mit dem ***SQL Tuning Advisor*** und Akzeptieren des empfohlenen Ausführungsplans.
- Funktionen `LOAD_PLANS_FROM_SQLSET`, `LOAD_PLANS_FROM_CURSOR_CACHE`

SPM – Einsatzszenarien

Datenbank-Upgrade

- Sammlung gut optimierter Pläne in einem SQL Tuning Set auf der Datenbank 10g
- Übertragen des STS auf die Datenbank 11g
- Laden des STS in die SQL Plan Baseline

Software Deployment

- Implementierung der neuen Software auf einem Testsystem
- Test einer eventuell mitgelieferten SQL Plan Baseline
- Entwicklung der Pläne
- Packen der optimierten Pläne in eine Staging Table
- Export/Import der Staging Table auf das Produktivsystem
- Entpacken der SQL Plan Baseline.

SPM und SPA

- Anwendung des ***SQL Performance Analyzer*** in einem Upgrade-Szenarium nach 11g (Parameter `OPTIMIZER_FEATURES_ENABLE`)
- Sammlung der Pläne für SQL mit Planregression in einem SQL Tuning Set auf dem Ursprungssystem (10g)
- Übertragen des STS auf das Zielsystem (11g) und laden in die SQL Plan Baseline

Automatisches SQL-Tuning

Ablauf

1. Top-SQL wird auf der Basis der Informationen des AWR erkannt.
2. Für diese SQL wird in einem automatischen Wartungsjob *seriell* der ***SQL Tuning Advisor*** ausgeführt.
3. Die generierten SQL-Profile werden getestet.
4. Die Profile werden implementiert, wenn ein *dreifacher* Performancegewinn zu verzeichnen ist. (Summe von CPU *und* I/O, keines der beiden darf sich verschlechtern)

Einschränkungen **(keine automatische Optimierung)**

- Ø Seltene bzw. adhoc-Abfragen
- Ø Parallele Abfragen
- Ø Unlängst optimierte Anweisungen
- Ø Rekursive SQL
- Ø DML und DDL
- Ø Anweisungen, die durch Transaktionssperren inperformant sind
- Ø Automatisch implementiert werden nur Profile, keine anderen Empfehlungen des STA (Statistiken, Indizes, Restrukturierung der SQL).

AST – Verwaltung

Autotask

- Ein- und Ausschalten
- Ressourcennutzung
- Wartungsfenster

Task-Parameter (`DBMS_SQLTUNE`)

- Ein- und Ausschalten der automatischen Implementierung (Standard ist Ausgeschaltet!)
- Zeitlimit für den Job
- Zeitlimit für die einzelne zu optimierende SQL-Anweisung
- Deaktivieren der Testausführung zur Zeitersparnis (Profile werden generiert und implementiert, aber nicht getestet)
- Maximale Anzahl der zu implementierenden SQL-Profile
- Gültigkeit der Task-Ausführung (Wann kann eine Anweisung wieder Gegenstand eines Task werden?)

AST im OEM

Zusammenfassung der Ergebnisse des automatischen SQL Tunings

Das automatische SQL Tuning wird während Fenstern zur Systemwartung als automatisierte Wartungs-Task ausgeführt, die nach Möglichkeiten sucht, die Ausführungspläne von stark belasteten SQL-Anweisungen zu verbessern.

Task-Status

Automatisches SQL Tuning (SYS_AUTO_SQL_TUNING_TASK) ist aktuell **Aktiviert** Konfigurieren

Automatische Implementierung von SQL-Profilen ist aktuell **Deaktiviert**
Hochempfohlene SQL-Profile **0**

Zusammenfassung der Task-Aktivität

In dem Diagramm mit der Aktivitätszusammenfassung werden die Vorteile der Task-Aktivitäten bei stark belasteteten SQL-Anweisungen des Systems dargestellt. Nur Profile, die die SQL-Performance signifikant verbessern, wurden implementiert.

Zeitraum Alle Los Bericht anzeigen

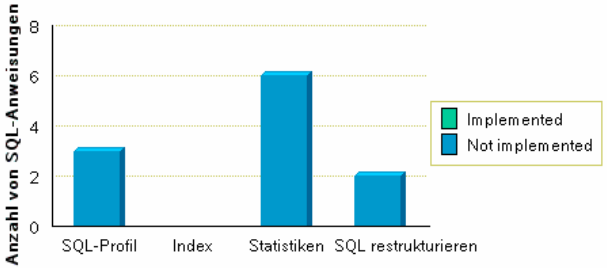
Anfangsdatum **02.11.2008 11:03:59 (UTC+01:00)** Enddatum **30.11.2008 12:01:24 (UTC+01:00)**

Gesamt-Task-Statistiken

Ausführungen **7** In Frage kommende SQL **15** Eindeutige SQL überprüft **9**

Status von SQL überprüft

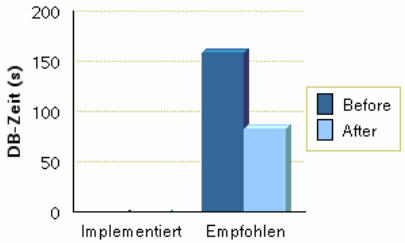
Aufgliederung nach Ergebnistyp



Statistiken der Profilauswirkungen

DB-Zeitvorteil bei optimierter SQL (Sekunden pro Woche)

Implementiert (Sek) **0** Potenziell (Sek) **76**



Real Application Testing

Ziel: *Proaktives* Change Management

Zwei Bestandteile:

- **Database Replay**
  Workload des Produktivsystems wird in seiner Intensität und zeitlichen Abfolge auf einem Testsystem „abgespielt“.
  Dreistufiger Prozeß: **Capture** → **Process** → **Replay** Workload
  Sinnvoll auf drei verschiedenen Systemen auszuführen.
- **SQL Performance Analyzer (SPA)**
  Die aus dem Produktivsystem extrahierten SQL-Anweisungen werden **seriell** auf einem Testsystem ausgeführt und bezüglich der Änderungen der Antwortzeit bewertet. Es gibt zwei voreingestellte Szenarien:
  - Upgrade Ê `OPTIMIZER_FEATURES_ENABLE`
  - Änderung beliebiger Initialisierungsparameter

Einschränkung: Leider ist Real Application Testing eine Zusatzoption zur Enterprise Edition und verlangt außerdem die Lizenzierung von Diagnostic und Tuning Pack

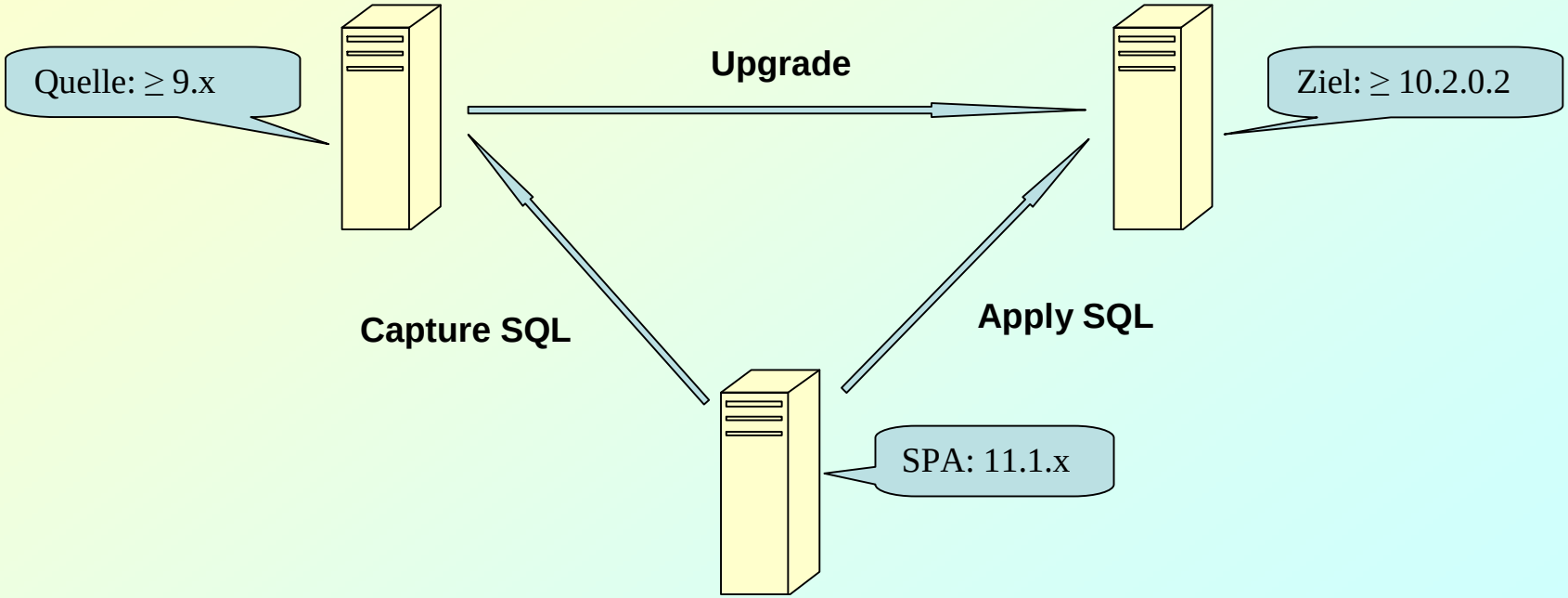
SQL Performance Analyzer – Ablauf

1. **Sammlung der zu analysierenden Anweisungen in einem SQL Tuning Set (STS) auf dem Produktivsystem**
2. **Übertragung des STS auf das Testsystem**
3. **Anwendung der SQL des STS durch den SPA**
4. **Änderungen vornehmen**
5. **Erneute Anwendung der SQL des STS durch den SPA**
6. **Vergleich der Anwortzeiten durch den SPA (Welche Anweisungen laufen gleich schnell oder gar besser und welche schlechter?)**
7. **Besser oder gleich: Anlegen einer SQL Plan Baseline**
8. **Schlechter: Anwendung des SQL Tuning Advisor**
9. **Integration der SQL-Profile (Resultat des STA) in die Baseline**
10. **Übertragung der Baseline auf das Produktivsystem**
11. **Implementierung der Änderungen auf dem Produktivsystem**

Neu im SPA

Real Application Testing Now Available for Earlier Releases: Metalink 560977.1 und 562899.1

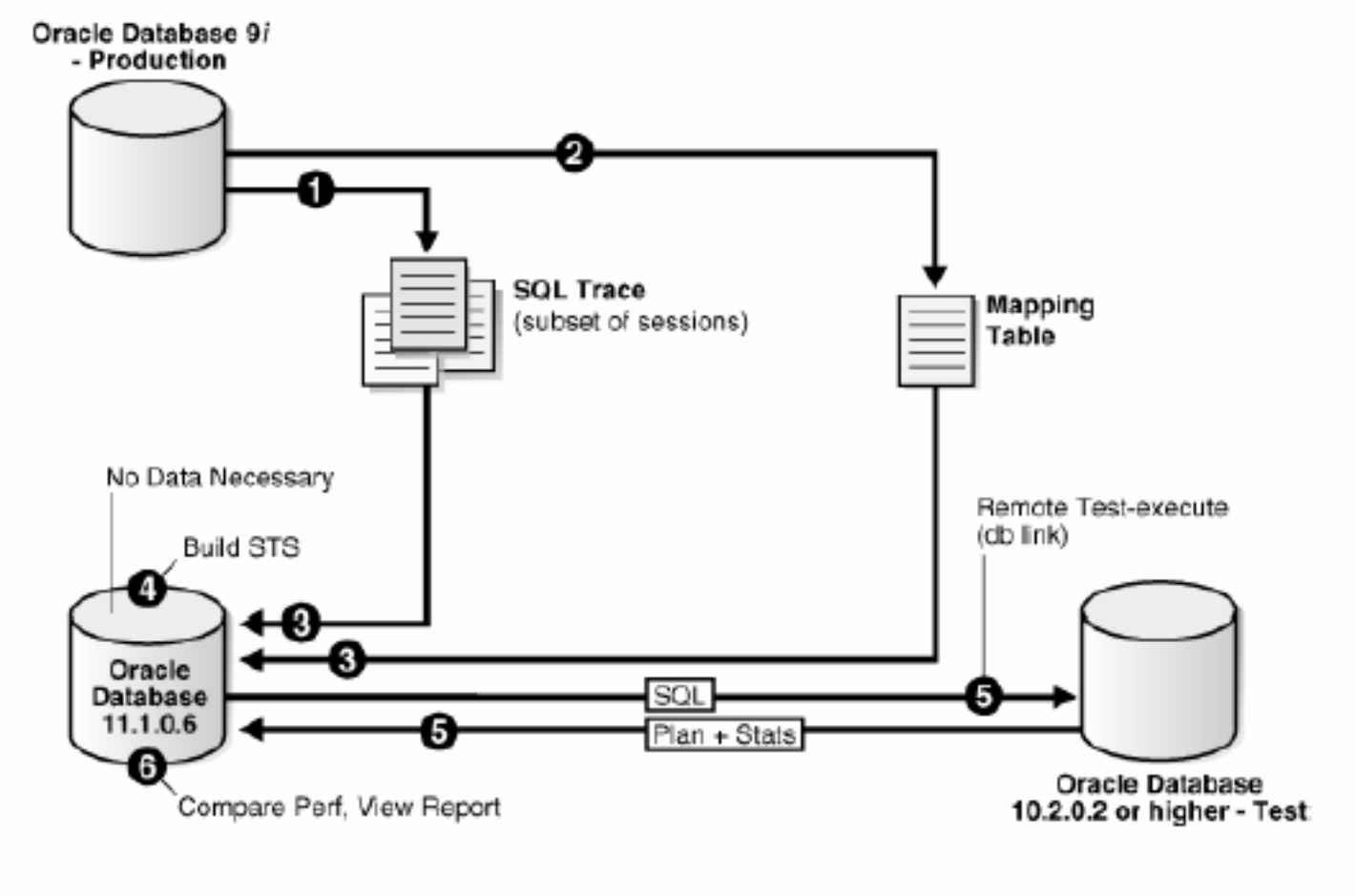
Notwendig: 11g-Datenbank, auf der SPA läuft und One Off Patch 6903335 angewendet ist, eventuell Patches auf Quelle und Ziel



Anmerkung: Geht auch mit Database Replay, dann muß das Ziel aber **mindestens** 11.1.0.6 sein.

SPA und Upgrade von 9*i*

Problem: Oracle 9*i* kennt keine SQL Tuning Sets!



Bind Peeking in 10*g* I

```
Abfrage: SELECT COUNT(*) FROM test_bp WHERE owner = :a;
Tabelle TEST_BP
OWNER                            COUNT(*)
------------------------------ ----------
SCOTT                                  11
OE                                     10
SH                                     17
SYSMAN                                681
SYS                                 870870
rows per distinct key
---------------------
              174317,8
```

Ohne Bind Peeking

```
ALTER SESSION SET "_OPTIM_PEEK_USER_BINDS"=FALSE;
:a := 'SYS'
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT  |             |       |       |   362 (100)
|   1 |  SORT AGGREGATE   |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_BP_IDX |   174K|   852K|   362   (1)
:a := 'SCOTT'
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT  |             |       |       |   362 (100)
|   1 |  SORT AGGREGATE   |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_BP_IDX |   174K|   852K|   362   (1)
```

Bind Peeking in 10*g* II

Mit Bind Peeking

```
ALTER SESSION SET "_OPTIM_PEEK_USER_BINDS"=TRUE;
:a := 'SCOTT'
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT   |             |       |       |     3 (100)
|   1 |  SORT AGGREGATE    |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_BP_IDX |    15 |    75 |     3   (0)
:a := 'SYS'
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT   |             |       |       |     3 (100)
|   1 |  SORT AGGREGATE    |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_BP_IDX |    15 |    75 |     3   (0)
```

Umgekehrte Reihenfolge der Statements

```
:a := 'SYS'
| Id  | Operation              | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT       |             |       |       |   495 (100)
|   1 |  SORT AGGREGATE        |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |   872K|  4258K|   495   (3)
:a := 'SCOTT'
| Id  | Operation              | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT       |             |       |       |   495 (100)
|   1 |  SORT AGGREGATE        |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |   872K|  4258K|   495   (3)
```

Adaptive Cursor Sharing in 11*g* – Prinzip

- Adaptive Cursor Sharing ermöglicht die gemeinsame Verwendung der Cursor für Anweisungen mit Binde-Variablen.
- Kompromiß zwischen der Reduktion der Parse-Zeit durch Cursor Sharing und der Suche nach dem wirklich optimalen Ausführungsplan
- Vorteile:
  - Es wird erkannt, wenn unterschiedlich selektive Werte für Bindevariablen von verschiedenen Ausführungsplänen profitieren.
  - Die Anzahl der (verwendeten) untergeordneten (Child-) Cursor wird reduziert.

Informationen über Adaptive Cursor Sharing

Views zum Adaptive Cursor Sharing:

V$SQL	Zwei neue Spalten: is_bind_sensitive is_bind_aware
V$SQL_CS_HISTOGRAM	Verteilung der Ausführungen auf die Buckets des Ausführungshistogramms
V$SQL_CS_SELECTIVITY	Zeigt die Selektivitäts-Bereiche, die für das jeweilige Prädikat und den dazugehörigen Child Cursor verwendet werden, um zu prüfen, ob der Cursor gemeinsam genutzt werden kann.
V$SQL_CS_STATISTICS	Ausführungsstatistiken eines Cursors für das Cursor Sharing

Adaptive Cursor Sharing – Ablauf I

Cursor mit Bindevariable (bind sensitive):

```
SELECT COUNT(*) FROM test_bp WHERE owner=:a;
```

Cursor wird überwacht!

`:a := 'SCOTT'` **Hard Parse mit Bind Peeking: Selektivität hoch** **→ Index Range Scan**

`:a := 'SYS'` **Soft Parse** **→ Index Range Scan**

```
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT  |             |       |       |     3 (100)
|   1 |  SORT AGGREGATE   |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_BP_IDX |     5 |    25 |     3   (0)
```

Vergleich der Ausführungsstatistiken: Große Differenz!

`:a := 'SYS'` **Hard Parse mit Bind Peeking: Selektivität niedrig** **→ Index Fast Full Scan**

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
aa6ygbdb48m3x          1            1 Y Y
aa6ygbdb48m3x          1            2 Y Y
```

Ê Zwei Child Cursor mit unterschiedlichen Ausführungsplänen

```
PREDICATE    CHILD_NUMBER LOW         HIGH
------------ ------------ ---------- ----------
=A                      2 0.899987   1.099984
=A                      1 0.000010   0.000013
```

Adaptive Cursor Sharing – Ablauf II

```
SELECT sql_id, executions, child_number,
is_bind_sensitive s, is_bind_aware a
FROM v$sql
WHERE UPPER(sql_text) LIKE '%COUNT(*) FROM
TEST%'
```

```
:a := 'SY...'    ...Parse mit Bind Peeking: Selektivität hoch    → Index Range Scan
:a := 'SYS'      Soft Parse                                      → Index Range Scan
```

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
```

Vergleich der Ausführungsstatistiken: Große Differenzen

```
:a := 'SYS'    Hard Parse mit Bind Peeking: Selektivität niedrig    → Index Fast Full Scan
```

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
aa6ygbdb48m3x          1            1 Y Y
aa6ygbdb48m3x          1            2 Y Y
```

Ê Zwei Child Cursor mit unterschiedlichen Ausführungsplänen

```
PREDICATE    CHILD_NUMBER LOW         HIGH
------------ ------------ ---------- ----------
=A                      2 0.899987   1.099984
=A                      1 0.000010   0.000013
```

Adaptive Cursor Sharing – Ablauf III

Cursor mit Bindevariable (bind sensitive):

```
SELECT COUNT(*) FROM test_bp WHERE owner=:a;
```

Cursor wird überwacht!

`:a := 'SCOTT'` **Hard Parse mit Bind Peeking: Selektivität hoch** **→ Index Range Scan**
`:a := 'SYS'` **Soft Parse** **→ Index Range Scan**

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
```

Vergleich der Ausführungsstatistiken: Große Differenzen

`:a := 'SYS'` **Hard Parse mit Bind Peeking: Selektivität niedrig** **→ Index Fast Full Scan**

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y
```

```
| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT      |             |       |       |   518 (100)
|   1 |  SORT AGGREGATE       |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |   872K|  4260K|   518   (3)
```

```
PREDICATE    CHILD_NUMBER LOW         HIGH
------------ ------------ ----------- -----------
=A                      2 0.899987    1.099984
=A                      1 0.000010    0.000013
```

Adaptive Cursor Sharing – Ablauf IV

Cursor mit Bindevariable (bind sensitive):

```
SELECT COUNT(*) FROM test_bp WHERE owner=:a;
```

Cursor wird überwacht!

`:a := 'SCOTT'` **Hard Parse mit Bind Peeking: Selektivität hoch** **→ Index Range Scan**
`:a := 'SYS'` **Soft Parse** **→ Index Range Scan**

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
```

Vergleich der Ausführungsstatistiken: Große Differenzen

`:a := 'SYS'` **Hard Parse mit Bind Peeking: Selektivität niedrig** **→ Index Fast Full Scan**

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
```

```
SELECT predicate, child_number, low, high
FROM v$sql_cs_selectivity
```

```
aa6ygbdb4                1            2 Y Y
```

Ê Zw Child Cursor mit unterschiedlichen Ausführungsplänen

```
PREDICATE    CHILD_NUMBER LOW         HIGH
------------ ------------ ---------- ----------
=A                      2 0.899987   1.099984
=A                      1 0.000010   0.000013
```

Getrennte Selektivitätsbereiche (Buckets)

Adaptive Cursor Sharing – Ablauf V

Ausführung des Cursor mit anderer Belegung der Bindevariable

`:a := 'SH ', :a := 'SYSMAN', :a := 'OE'` Soft Parse

Ê Drei neue Child Cursor mit gleichen Ausführungsplänen
Ê Vergleich der Ausführungsstatistiken: Keine Differenzen
Ê Zusammenführung der Selektivitätsbereiche

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
aa6ygbdb48m3x          1            1 Y Y
aa6ygbdb48m3x          1            2 Y Y
aa6ygbdb48m3x          1            3 Y Y
aa6ygbdb48m3x         36            4 Y Y
aa6ygbdb48m3x          9            5 Y Y

PREDICATE    CHILD_NUMBER LOW         HIGH
------------ ------------ ---------- ----------
=A                      5 0.899228   1.099057
=A                      4 0.000005   0.000006
=A                      4 0.000728   0.000889
=A                      4 0.000016   0.000025
=A                      3 0.000005   0.000006
=A                      3 0.000728   0.000889
=A                      3 0.000016   0.000019
=A                      2 0.000728   0.000889
=A                      2 0.000016   0.000019
=A                      1 0.000016   0.000019
```

Zusammenführung von Selektivitätsbereichen

Adaptive Cursor Sharing und Bereichabfragen I

```
SELECT count(*) FROM test_rs WHERE obj_id > :a;
obj_id ist UNIQUE, Werte von 1 … 10.000
```

:a := 99999
:a := 1

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
4tf4p8g02dhht          2            0 Y N
```

Wiederholte Ausführungen mit verschiedener Variablenbelegung:

:a := 99999
:a := 1

```
4tf4p8g02dhht          2            0 Y N
4tf4p8g02dhht          1            1 Y Y
4tf4p8g02dhht          1            2 Y Y
```

Adaptive Cursor Sharing und Bereichabfragen II

```
SELECT count(*) FROM test_rs WHERE obj_id > :a;
obj_id ist UNIQUE, Werte von 1 … 10.000
```

:a := 99999
:a := 1

```
SQL_I          EXECUTIONS CHILD_NUMBER S A
-----          ---------- ------------ - -
4tf4p8                  2            0 Y N

| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT  |             |       |       |     2 (100)|
|   1 |  SORT AGGREGATE   |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_RS_IDX |     1 |     5 |     2   (0)|
```

:a := 99999
:a := 1

```
4tf4p8g02dhht          2            0 Y N
4tf4p8g02dhht          1            1 Y Y
4tf4p8g02dhht          1            2 Y Y
```

Adaptive Cursor Sharing und Bereichabfragen III

```
SELECT count(*) FROM test_rs WHERE obj_id > :a;
obj_id ist UNIQUE, Werte von 1 … 10.000
```

:a := 99999
:a := 1

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
4tf4p8g02dhht          2            0 Y N
```

```
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT  |             |       |       |     2 (100)|
|   1 |  SORT AGGREGATE   |             |     1 |     5 |
|*  2 |   INDEX RANGE SCAN| TEST_RS_IDX |     1 |     5 |     2   (0)|
```

:a := 99999
:a := 1

```
4tf4p                  2            0 Y N
```

```
| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT      |             |       |       |    59 (100)
|   1 |  SORT AGGREGATE       |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_RS_IDX | 99170 |   484K|    59   (2)
```

Adaptive Cursor Sharing und CURSOR_SHARING

```
SELECT COUNT(*) FROM test_bp WHERE owner = 'SCOTT';
usw. für 'SYS', 'SH', 'OE' und 'SYSMAN' ...
```

CURSOR_SHARING = SIMILAR

SQL_ID	EXECUTIONS	CHILD_NUMBER	S	A
4hkg2w543mxuw	5	0	Y	N
4hkg2w543mxuw	5	1	Y	N
4hkg2w543mxuw	5	2	Y	N
4hkg2w543mxuw	5	3	Y	N
4hkg2w543mxuw	5	4	Y	N

CURSOR_SHARING = FORCE

SQL_ID	EXECUTIONS	CHILD_NUMBER	S	A
4hkg2w543mxuw	2	0	Y	N
4hkg2w543mxuw	1	1	Y	Y
4hkg2w543mxuw	1	2	Y	Y
4hkg2w543mxuw	1	3	Y	Y
4hkg2w543mxuw	**16**	4	Y	Y
4hkg2w543mxuw	**4**	5	Y	Y

Adaptive Cursor Sharing und SPM – I

```
ALTER SESSION SET optimizer_capture_sql_plan_baselines=true;
```

```
a := 'SYS'
a := 'SCOTT'
…
```

```
| Id  | Operation            | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT     |             |       |       |   518 (100)
|   1 |  SORT AGGREGATE      |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |  867K|  4235K|   495   (3)
```

```
SQL_ID         EXECUTIONS CHILD_NUMBER S A
-------------- ---------- ------------ - -
aa6ygbdb48m3x           2            0 Y N
aa6ygbdb48m3x           2            2 N N
```

```
PLAN_NAME                      ENABLED ACCEPTED
------------------------------ ------- --------
SYS_SQL_PLAN_0d77894f83f328b7     YES       NO
SYS_SQL_PLAN_0d77894fd769c429     YES      YES
```

Adaptive Cursor Sharing und SPM – II

```
ALTER SESSION SET optimizer_capture_sql_plan_baselines=true;
```

```
a := 'SYS'
a := 'SCOTT'
…
```

```
| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT      |             |       |       |   518 (100)
|   1 |  SORT AGGREGATE       |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |     5 |    25 |   495   (3)
```

```
SQL_ID        ---------- ---------- - -
aa6ygbdb48m3x          2          0 Y N
aa6ygbdb48m3x          2          2 N N
```

```
PLAN_NAME                      ENABLED ACCEPTED
------------------------------ ------- --------
SYS_SQL_PLAN_0d77894f83f328b7     YES       NO
SYS_SQL_PLAN_0d77894fd769c429     YES      YES
```

Adaptive Cursor Sharing und SPM – III

```
ALTER SESSION SET optimizer_capture_sql_plan_baselines=true;
```

```
a := 'SYS'
a := 'SCOTT'
…
```

```
SQL_ID
---------
aa6ygbdb4
aa6ygbdb4
```

```
| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)
|   0 | SELECT STATEMENT      |             |       |       |   518 (100)
|   1 |  SORT AGGREGATE       |             |     1 |     5 |
|*  2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |     5 |    25 |   495   (3)

SQL plan baseline SYS_SQL_PLAN_0d77894fd769c429 used for this statement
```

```
PLAN_NAME                      ENABLED ACCEPTED
------------------------------ ------- --------
SYS_SQL_PLAN_0d77894f83f328b7     YES       NO
SYS_SQL_PLAN_0d77894fd769c429     YES      YES
```

Adaptive Cursor Sharing und SPM – IV

```
ALTER SESSION SET optimizer_capture_sql_plan_baselines=true;
```

```
a := 'SYS'
a := 'SCOTT'
…
```

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
-------------
aa6ygbdb48m3x
aa6ygbdb48m3x
```

```
select plan_name, enabled, accepted
from dba_sql_plan_baselines
where upper(sql_text) like '%COUNT(*) FROM TEST%';
```

```
PLAN_NAME                      ENABLED ACCEPTED
------------------------------ ------- --------
SYS_SQL_PLAN_0d77894f83f328b7     YES       NO
SYS_SQL_PLAN_0d77894fd769c429     YES      YES
```

Adaptive Cursor Sharing und SPM – V

```
select * from table(dbms_xplan.display_sql_plan_baseline(sql_handle
=> 'SYS_SQL_af5365350d77894f', format  => 'basic'));
```

```
SQL handle: SYS_SQL_af5365350d77894f
SQL text: SELECT COUNT(*) FROM test_bp WHERE owner = :a
--------------------------------------------------------------
Plan name: SYS_SQL_PLAN_0d77894fd769c429
Enabled: YES Fixed: NO Accepted: YES Origin: AUTO-CAPTURE
--------------------------------------------------------------
Plan hash value: 1833838675
-------------------------------------------
| Id  | Operation             | Name        |
-------------------------------------------
|   0 | SELECT STATEMENT      |             |
|   1 |  SORT AGGREGATE       |             |
|   2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |
-------------------------------------------
Plan name: SYS_SQL_PLAN_0d77894f83f328b7
Enabled: YES Fixed: NO Accepted: NO Origin: AUTO-CAPTURE
--------------------------------------------------------------
Plan hash value: 1215588110
-----------------------------------------
| Id  | Operation         | Name        |
-----------------------------------------
|   0 | SELECT STATEMENT  |             |
|   1 |  SORT AGGREGATE   |             |
|   2 |   INDEX RANGE SCAN| TEST_BP_IDX |
```

Adaptive Cursor Sharing und SPM – VI

```
select DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(&sqlid) bericht from dual;
```

```
Report
-------------------------------------------------------------------------------
Inputs:
-------
SQL_HANDLE = SYS_SQL_af5365350d77894f
Plan: SYS_SQL_PLAN_0d77894f83f328b7
-----------------------------------
improvement ratio >= 613,93.
  Plan was changed to an accepted plan.
                   Baseline Plan      Test Plan     Improv. Ratio
                   -------------      ---------     -------------
Execution Status:       COMPLETE       COMPLETE
Rows Processed:                1              1
Elapsed Time(ms):             50              0
  CPU Time(ms):               48              0
  Buffer Gets:              1837              3           612,33
  Disk Reads:                  0              0
  Direct Writes:               0              0
  Fetches:                     0              0
Executions:                    1              1
Report
Summary
-------------------------------------------------------------------------------
Number of SQL plan baselines verified: 1.
Number of SQL plan baselines evolved: 1.
```

Adaptive Cursor Sharing und SPM – VII

```
select * from table(dbms_xplan.display_sql_plan_baseline(sql_handle
=> &sqlid, format  => 'basic'));
```

```
SQL handle: SYS_SQL_af5365350d77894f
SQL text: SELECT COUNT(*) FROM test_bp WHERE owner = :a
--------------------------------------------------------------
Plan name: SYS_SQL_PLAN_0d77894fd769c429
Enabled: YES Fixed: NO Accepted: YES Origin: AUTO-CAPTURE
--------------------------------------------------------------
Plan hash value: 1833838675
-----------------------------------------------
| Id  | Operation             | Name        |
-----------------------------------------------
|   0 | SELECT STATEMENT      |             |
|   1 |  SORT AGGREGATE       |             |
|   2 |   INDEX FAST FULL SCAN| TEST_BP_IDX |
-----------------------------------------------
Plan name: SYS_SQL_PLAN_0d77894f83f328b7
Enabled: YES Fixed: NO Accepted: YES Origin: AUTO-CAPTURE
--------------------------------------------------------------
Plan hash value: 1215588110
----------------------------------------
| Id  | Operation         | Name        |
----------------------------------------
|   0 | SELECT STATEMENT  |             |
|   1 |  SORT AGGREGATE   |             |
|   2 |   INDEX RANGE SCAN| TEST_BP_IDX |
```

Adaptive Cursor Sharing und SPM – VIII

```
SELECT COUNT(*) FROM test_bp WHERE owner = :a;
SELECT * FROM TABLE(DBMS_XPlan.Display_Cursor);
```

```
PLAN_TABLE_OUTPUT
------------------------------------------
--
SQL_ID  aa6ygbdb48m3x, child number 2
An uncaught error happened in prepare_sql_statement : ORA-01403: Keine Daten
gefunden
NOTE: cannot fetch plan for SQL_ID …
```

EVOLVE_SQL_PLAN_BASELINE bedeutet Invalidierung des Cursor!

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
aa6ygbdb48m3x          1            1 N N
```

Weitere Ausführungen mit der wechselnder Variablenbelegung:

```
SQL_ID        EXECUTIONS CHILD_NUMBER S A
------------- ---------- ------------ - -
aa6ygbdb48m3x          2            0 Y N
...
aa6ygbdb48m3x          3            6 Y Y
aa6ygbdb48m3x          2            7 Y Y
aa6ygbdb48m3x          6            8 Y Y
```

Mit jedem Wechsel der Variablenbelegung neuer Child Cursor!

Fazit: Das Wechselverhältnis von Adaptive Cursor Sharing und SQL Plan Management ist nicht unproblematisch!

Adaptive Cursor Sharing verwalten

- Parameter `CURSOR_SHARING`:
  - `CURSOR_SHARING != EXACT`: Für Anweisungen, die Literale im Prädikat enthalten, werden unter Umständen systemgenerierte Bindevariablen verwendet.
  - Adaptive Cursor Sharing kann bei diesen Anweisungen verwendet werden. (`FORCE`)
- SQL Plan Management (SPM):

  `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES = TRUE`

  Unter Umständen wird nur der erste generierte Plan Bestandteil der Baseline:

  Lösung:
  - Parameter auf FALSE einstellen und die Anwendung ausführen, bis alle Pläne in den Cursor Cache geladen sind.
  - Danach muß man den Cursor Cache manuell als SQL Plan Baseline dem SPM zur Verfügung stellen.
- Kann mit `_optimizer_adaptive_cursor_sharing` oder `OPTIMIZER_FEATURE_ENABLE` $\leq$ `10.2.0.4` ausgeschaltet werden .

Full Table Scans in Oracle 11*g*

```
SELECT COUNT(*) FROM test_dr;
Kein Index, ca. 8 Mio. Zeilen, 390 MB, entspricht 40% Buffer Cache
```

Ausführung der Query	Antwortzeit 10*g* in Sek.	Antwortzeit 11*g* in Sek.
1	42	**10**
2	0,4	**10**

- Grund:
  10*g* liest beim zweiten Mal aus dem Buffer Cache
  11*g* liest diese Tabellengröße **immer** durch **Direct Path Read**
  Schwellenwert ist Tabelle/Buffer Cache > 0,1 (10%)
  - Unter der Schwelle ist das Verhalten wie in 10*g*
  - Es gibt keinen Schalter, um das Verhalten umzustellen. (`_serial_direct_read=false` hilft nicht!)
- Warteereignisse:
  - `KJC: Wait for msg sends to complete`
  - `enq: KO - fast object checkpoint`
  - `direct path read` (128 Blöcke in einem Vorgang)
- Lösung: Verwendung des **SQL Result Cache**
  `SELECT /*+ result_cache */ COUNT(*) FROM test_dr;`
  Antwortzeit bei wiederholter Ausführung **< 1 cs**

Alternative Tools

- Extended SQL Tracing (DBMS_MONITOR), Auswertung mit:
  - TKPROF (evt. Vorbereitung mit TRCSESS)
  - TRCA (Trace Analyzer von Carlos Sierra – Metalink 224270.1)
  - TVDXTAT (Trivadis Extended Tracefile Analysis Tool von Chr. Antognini
    http://antognini.ch/2008/10/introduce-tvdxtat/)
  - ORASRP (Oracle Session Resource Profiler – Egor Starostin
    http://oracledba.ru/orasrp/)
- Lite Onboard Monitor (LTOM) von Carl Davis (Metalink 352363.1)

  Automatische Echtzeitanalyse der Datenbank auf der Basis von Regeln

  Folgende Modi sind einstellbar:
  - Automatic Hang Detection: Warten auf Non Idle Wait Events triggert spezifische Reports
  - System Profiler: Polling wichtiger v$-views und OS-Statistiken, Analyse der Daten mit dem graphischen Tool LTOMg
  - Automatic Session Tracing: Automatisches 10046-Tracing für Sessions, die bestimmte Regeln verletzen, entweder in Dateien oder in den Speicher
- Instantaneous Problem Detection OS Tool (IPD/OS)
  (http://www.oracle.com/technology/products/database/clustering/index.html)

  Echtzeitüberwachung von Betriebssystem und Clusterware!

  Repository ist eine Berkley DB
- Weitere Tools: Metalink 169935.1

Vorteil: Keine zusätzlichen Lizenzkosten.

Literatur

- A. Held: Oracle 11*g*. Neue Features. Hanser, München 2008
- S. R. Alapati, C. Kim: Oracle Database 11*g*. New Features for DBAs and Developers. Apress, Berkeley (CA) 2007.
- Chr. Antognini: Troubleshooting Oracle Performance. Apress, Berkeley (CA) 2008.
- Metalink http://metalink.oracle.com
- OTN http://otn.oracle.com
- Oracle11*g* Database Online Documentation, Release 1, vor allem folgende Teile:
  - Oracle Database Administrators Guide
  - Oracle Database Performance Tuning Guide

Dr. Frank Haney
info@it-haney.de
Tel.: 03641-210224

ORACLE | CERTIFIED PROFESSIONAL