Evolution of performance management
Oracle 12c adaptive optimization

Nelson Calero

OTN Tour Latinoamérica
Agosto 2016
About me

• Database Consultant at Pythian
• Working with Oracle tools and Linux environments since 1996
• Co-founder and President of the Oracle user Group of Uruguay (2009)
• LAOUC Director of events (2013)

• Computer Engineer
• Oracle ACE (2014)
• Oracle Certified Professional DBA 10g/11g (2008)
• Amazon Solutions Architect – Associate since (2016)
• Oracle University Instructor (2011)
• Blogger and speaker: Oracle Open World, Collaborate, OTN Tour, Regional conferences

http://www.linkedin.com/in/ncalero
@ncalerouy
Pythian overview

- 19 Years of data infrastructure management consulting
- 250+ Top brands
- 11700+ Systems under management
- Over 400 DBAs in 35 countries
- Top 5% of DBA work force, 10 Oracle ACEs, 4 ACED, 3 OakTable members, 2 OCM, 6 Microsoft MVPs, 1 Cloudera Champion of Big Data, AWS Certified Solutions Architect – 2 Professional, 12 Associate
- Oracle, Microsoft, MySQL, Hadoop, Cassandra, MongoDB, and more
- Infrastructure, Cloud, DevOps, and application expertise
Adaptive optimizations

What
• Starting in version 12.1, Oracle optimizer can adjust SQL execution plan during its first execution to create a better performing plan, and use new techniques to do better cardinality estimates on following executions

Why
• These features are enabled by default, controlled by parameters

How
• Adaptive statistics features has no extra cost, available on all editions
• Adaptive plans is available only on Enterprise edition
Adaptive optimizations

There are several cases where the execution plan generated by the optimizer is not the optimal for the underlying tables characteristics. This can be caused by a variety of reasons:

– Structural objects changes (datatype, indexes, partitions)
– System growth (skewed data, concurrency)
– And many more. This is a good collection of reason: [http://jonathanlewis.wordpress.com/2013/12/23/plan-changes/](http://jonathanlewis.wordpress.com/2013/12/23/plan-changes/)

Oracle Database already have several ways to control plan execution that works at different stages in the cost based Optimizer:

– Stored Outlines (deprecated in 11.1)
– SQL Hints
– SQL Patches
– SQL Profiles
– SQL Plan Management
Adaptive optimizations

What

• Several new optimization techniques at query run-time – after execution plan has been chosen
• Adaptive features are evaluated only the first time a query is parsed. Next executions re-use the adaptive plan
• Some features are persisted on SYSAUX tablespace (plan directives)

Why

• Baselines are always the last step – no adaptive optimization over a baseline plan used

How

• Initialization parameters and hints to control them
Adaptive optimizations - How

Old history

Oracle Cost Based Optimizer (CBO)
Before 12g
Adaptive optimizations - How

Execution plan can change during SQL execution

Oracle Cost Based Optimizer (CBO)
Since 12g
Review of Adaptive optimizations features

• Adaptive statistics
  • Dynamic Statistics
  • Automatic Reoptimization
  • SQL Plan Directives
• Adaptive plans
  • Join Methods
  • Parallel Distribution Methods
Adaptive statistics - dynamic statistics

• Was Dynamic sampling feature introduced on 10g
  – stats can be gathered for objects where not enough stats are present during parsing
  – Triggered because of missing or stale stats or complex predicates
  – Level (0-10) control when it fires and blocks size of samples (default=2)

• New in 12c:
  – level 11: Optimizer decides if dynamic stats should be taken and level to use.
    Backported to 11.2.0.4
  – Support for joins and group by predicates

• Dynamic stats are stored in memory (result cache) - reused by other SQL
  – This can be seen on a SQL trace (not 10053)
Adaptive statistics - dynamic statistics

SCOTT@db12102/12.1.0.2> select status, enabled, count(1) from data group by status, enabled order by 1,2;

S E COUNT(1) 
A Y 2499
C N 5000
C Y 2500

Plan hash value: 3102269256

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>00:00:00.01</td>
<td>61</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>00:00:00.01</td>
<td>61</td>
</tr>
<tr>
<td>* 2</td>
<td>TABLE ACCESS FULL</td>
<td>DATA</td>
<td>1</td>
<td>3749</td>
<td>2500</td>
<td>00:00:00.01</td>
<td>61</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

2 - filter("ENABLED"='Y' AND "STATUS"='C')

SQL_ID 3q3tk8z3su2px, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ count(1) FROM data WHERE status = 'C' AND enabled='Y'

© 2014 Pythian Confidential
Adaptive statistics - dynamic statistics

SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv, is_bind_sensitive BS, is_bind_aware BA, IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM v$sql
where sql_id='3q3tk8z3su2px';

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>PLAN_HASH_VALUE</th>
<th>FULL_PHV</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3q3tk8z3su2px</td>
<td>0</td>
<td>3102269256</td>
<td>3774542448</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>2</td>
</tr>
</tbody>
</table>
Adaptive statistics - dynamic statistics

```sql
SCOTT@db12102/12.1.0.2> SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
  is_bind_sensitive BS, is_bind_aware BA,
  IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM v$sql
where sql_id='3q3tk8z3su2px';
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>PLAN_HASH_VALUE</th>
<th>FULL_PHV</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3q3tk8z3su2px</td>
<td>0</td>
<td>3102269256</td>
<td>3774542448</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>2</td>
</tr>
</tbody>
</table>

SCOTT@db12102/12.1.0.2> alter session set optimizer_dynamic_sampling=11;

Session altered.

SCOTT@db12102/12.1.0.2> SELECT /*+ GATHER_PLAN_STATISTICS */ count(1) FROM data
WHERE status = 'C' and enabled='Y';

Lets run the query again with new level 11:
Adaptive statistics - dynamic statistics

SQL_ID 3q3tk8z3su2px, child number 1

-------------------------------------

SELECT /*+ GATHER_PLAN_STATISTICS */ count(1) FROM data
WHERE status = 'C' and enabled='Y'

Plan hash value: 3102269256

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>00:00:00.01</td>
<td>61</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>00:00:00.01</td>
<td>61</td>
</tr>
<tr>
<td>* 2</td>
<td>TABLE ACCESS FULL</td>
<td>DATA</td>
<td>1</td>
<td>2500</td>
<td>2500</td>
<td>00:00:00.01</td>
<td>61</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

2 - filter(("ENABLED"='Y' AND "STATUS"='C'))

Note

-----

- dynamic statistics used: dynamic sampling (level=AUTO)
Adaptive statistics - dynamic statistics

SCOTT@db12102/12.1.0.2>
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv, is_bind_sensitive BS, is_bind_aware BA, IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe FROM v$sql
where sql_id='3q3tk8z3su2px';

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>PLAN_HASH_VALUE</th>
<th>FULL_PHV</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3q3tk8z3su2px</td>
<td>0</td>
<td>3102269256</td>
<td>3774542448</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>2</td>
</tr>
<tr>
<td>3q3tk8z3su2px</td>
<td>1</td>
<td>3102269256</td>
<td>3774542448</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>2</td>
</tr>
</tbody>
</table>

New cursor created because of dynamic sampling
Access path analysis for TAB1

******************************

SINGLE TABLE ACCESS PATH

Single Table Cardinality Estimation for TAB1

 SPD: Directive valid: dirid = 3423856709119923569, state = 1, flags = 1, loc = 1 {EC(92667)[3, 4, 5]}
 SPD: Return code in qosdDSDirSetup: EXISTS, estType = TABLE

... 

Table: TAB1  Alias: TAB1
 Card: Original: 10000.000000  >> Single Tab Card adjusted from 1869.000000 to 5000.000000 due to adaptive dynamic sampling
 Rounded: 5000  Computed: 5000.000000  Non Adjusted: 1869.000000
Review of Adaptive optimizations features

- Adaptive statistics
  - Dynamic Statistics
- Automatic Reoptimization
- SQL Plan Directives
- Adaptive plans
  - Join Methods
  - Parallel Distribution Methods
Adaptive statistics - Automatic reoptimization

Stats from previous query execution is used to create a better plan

1) statistics feedback
   • was cardinality feedback introduced on 11.2
     – when SQL has no stats, multiple predicates or where no accurate selectivity can be computed
     – after execution, cardinality estimates are compared to actual values on each operation, and differences stored for use on next execution
     – stored in memory by statement
   • SQL is marked as reoptimizable – v$sql.is_reoptimizable
   • Join stats are compared
   • Plan directive can be created to persist this information
Adaptive statistics – statistics feedback

Similar query, now selecting all data instead of count(*)

```
SCOTT@db12102/12.1.0.2> select status, enabled, count(1) from data
  group by status, enabled
order by 1,2;
S E COUNT(1)
B Y 2499
C N 5000
C Y 2500
```

---

```
SQL_ID 0h00c8djb1f0h, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ *
FROM data
WHERE status = 'B' and enabled='Y'
```

Plan hash value: 3160396028

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>2499</td>
<td>00:00:00.01</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS FULL</td>
<td>DATA</td>
<td>1</td>
<td>1249</td>
<td>2499</td>
<td>00:00:00.01</td>
<td>228</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - filter(("STATUS"='B' AND "ENABLED"='Y'))

Similar query, now selecting all data instead of count(*)

```
SCOTT@db12102/12.1.0.2> select status, enabled, count(1) from data
  group by status, enabled
order by 1,2;
S E COUNT(1)
B Y 2499
C N 5000
C Y 2500
```

---

```
SQL_ID 0h00c8djb1f0h, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ */ * FROM data
WHERE status = 'B' and enabled='Y'
```

Plan hash value: 3160396028

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>2499</td>
<td>00:00:00.01</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS FULL</td>
<td>DATA</td>
<td>1</td>
<td>1249</td>
<td>2499</td>
<td>00:00:00.01</td>
<td>228</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - filter(("STATUS"='B' AND "ENABLED"='Y'))

Similar query, now selecting all data instead of count(*)

```
SCOTT@db12102/12.1.0.2> select status, enabled, count(1) from data
  group by status, enabled
order by 1,2;
S E COUNT(1)
B Y 2499
C N 5000
C Y 2500
```
Adaptive statistics – statistics feedback

SQL_ID 0h00c8djb1f0h, child number 1

SELECT /*+ GATHER_PLAN_STATISTICS */ * FROM data
WHERE status = 'B' and enabled='Y'

Plan hash value: 3160396028

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>2499</td>
<td>2499</td>
<td>00:00:00.01</td>
<td>228</td>
</tr>
<tr>
<td>* 1</td>
<td>TABLE ACCESS FULL DATA</td>
<td></td>
<td>1</td>
<td>2499</td>
<td>2499</td>
<td>00:00:00.01</td>
<td>228</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - filter("STATUS"='B' AND "ENABLED"='Y')

Note

---

- statistics feedback used for this statement

Executed the query again
Adaptive statistics – statistics feedback

SCOTT@db12102/12.1.0.2>
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
  is_bind_sensitive BS, is_bind_aware BA,
  IS_REOPTIMIZEABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM  v$sql
where sql_id='0h00c8djb1f0h';

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>PLAN_HASH_VALUE</th>
<th>FULL_PHV</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h00c8djb1f0h</td>
<td>0</td>
<td>3160396028</td>
<td>2510600540</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>0h00c8djb1f0h</td>
<td>1</td>
<td>3160396028</td>
<td>2510600540</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>1</td>
</tr>
</tbody>
</table>

New cursor created because of statistics feedback
Adaptive statistics - Automatic reoptimization

2) Performance feedback

- new ADAPTIVE value for PARALLEL_DEGREE_POLICY
- the degree of parallelism (DOP) is automatically chosen as in AUTO mode, but at the end actual performance is compared with estimated, which can lead to a reoptimization on next execution (cursor marked as reoptimizable and statistics feedback stored for reuse)
Review of Adaptive optimizations features

- Adaptive statistics
  - Dynamic Statistics
  - Automatic Reoptimization
- SQL Plan Directives
- Adaptive plans
  - Join Methods
  - Parallel Distribution Methods
Adaptive statistics – SQL plan directives

– Stored feedback from SQL execution to adjust statistics on query expressions for next execution, created by automatic reoptimization
  • not in all cases (previous example about statistics)

– Stored in memory (result cache), flushed to SYSAUX each 15 min.
  • Purge policy in place
  • DBMS_SPD package to maintain them
  • Cannot be manually created

– DBA_SQL_PLAN_* views

⇒ All you want to know about it: UKOUG15 Frank Pachot session “SQL Plan Directives - The Memory of the 12c Optimizer”
Adaptive statistics – SQL plan directives

SYS@db12102/12.1.0.2> SELECT d.type, d.state, d.ENABLED, count(distinct d.directive_id) cant_dir, reason
FROM dba_sql_plan_directives d, dba_sql_plan_dir_objects o
WHERE d.directive_id=o.directive_id
GROUP BY d.type, d.state, d.ENABLED, reason
ORDER BY 1,2,3;

<table>
<thead>
<tr>
<th>TYPE</th>
<th>STATE</th>
<th>ENA</th>
<th>CANT_DIR</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>SUPERSEDED</td>
<td>YES</td>
<td>2</td>
<td>GROUP BY CARDINALITY MISESTIMATE</td>
</tr>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>SUPERSEDED</td>
<td>YES</td>
<td>15</td>
<td>JOIN CARDINALITY MISESTIMATE</td>
</tr>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>SUPERSEDED</td>
<td>YES</td>
<td>14</td>
<td>SINGLE TABLE CARDINALITY MISESTIMATE</td>
</tr>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>USABLE</td>
<td>YES</td>
<td>22</td>
<td>JOIN CARDINALITY MISESTIMATE</td>
</tr>
<tr>
<td>DYNAMIC_SAMPLING</td>
<td>USABLE</td>
<td>YES</td>
<td>24</td>
<td>SINGLE TABLE CARDINALITY MISESTIMATE</td>
</tr>
</tbody>
</table>
Adaptive statistics – SQL plan directives

SYS@db12102/12.1.0.2> SELECT o.owner, d.type, d.state, d.ENABLED,
   1  count(*) cant, count(distinct d.directive_id) cant_dir
   1  count(distinct object_name) cant_obj
FROM  dba_sql_plan_directives d, dba_sql_plan_dir_objects o
WHERE ddirective_id=o.directive_idgroup by o.owner, d.type, d.state, d.ENABLED
ORDER BY 1,2,3;

<table>
<thead>
<tr>
<th>OWNER</th>
<th>TYPE</th>
<th>STATE</th>
<th>ENA</th>
<th>CANT</th>
<th>CANT_DIR</th>
<th>CANT_OBJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTXSYS</td>
<td>DYNAMIC_SAMPLING</td>
<td>USABLE</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DBSNMP</td>
<td>DYNAMIC_SAMPLING</td>
<td>SUPERSEDED</td>
<td>YES</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SYS</td>
<td>DYNAMIC_SAMPLING</td>
<td>SUPERSEDED</td>
<td>YES</td>
<td>93</td>
<td>31</td>
<td>43</td>
</tr>
<tr>
<td>SYS</td>
<td>DYNAMIC_SAMPLING</td>
<td>USABLE</td>
<td>YES</td>
<td>116</td>
<td>46</td>
<td>36</td>
</tr>
</tbody>
</table>
Review of Adaptive optimizations features

• Adaptive statistics
  ✓ Dynamic Statistics
  ✓ Automatic Reoptimization
  ✓ SQL Plan Directives

• Adaptive plans
  • Join Methods
  • Parallel Distribution Methods
Adaptive plans

• Subplans are created at parse time and decision to choose one is deferred until actual execution. A default plan is created.

• A new step in the execution plan monitors rows processed (collector) to validate if predicted cardinality is accurate.

• A subplan is chosen, default plan can change:
  – join method (between Nested Loop (NL) and Hash Join (HJ))
  – parallel distribution method

• New adaptive plan is reused on future executions, no more optimization are repeated.
Adaptive plans – example

```
> select table_name, num_rows
  from user_tables where table_name in ('T1','T2');

TABLE_NAME    NUM_ROWS
----------      -------
T1            91138
T2            99

> insert into t2
  select rownum+100, mod(rownum,10) type, object_name data
  from dba_objects where rownum < 1000;

999 rows created.

> alter system flush shared_pool;
```

Let's see how a query is resolved without any adaptive optimization

```
> alter session set optimizer_adaptive_features=false;

Session altered.
```
Adaptive plans – example

SQL_ID 5bvpk0nnd4323, child number 1

---

SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --, t2.data FROM t2, t1 WHERE t1.id = t2.id2

Plan hash value: 1142304008

---

Predicate Information (identified by operation id):

4 - access("T1"."ID"="T2"."ID2")

Big difference in estimated rows

Nested Loop is used

23 rows selected.
Adaptive plans – example

Same query now with adaptive optimization enabled:

```sql
SCOTT@db12102/12.1.0.2> alter session set optimizer_adaptive_features=true;
Session altered.
Elapsed: 00:00:00.00

SCOTT@db12102/12.1.0.2> alter system flush shared_pool;
System altered.
Elapsed: 00:00:00.04
```
Adaptive plans – example

SQL_ID 5bvpk0nnd4323, child number 0

```
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --, t2.data FROM t2, t1 WHERE t1.id = t2.id2
```

Plan hash value: 324465990

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
<th>OMem</th>
<th>1Mem</th>
<th>Used-Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td>1</td>
<td>1098</td>
<td>00:00:00:00.01</td>
<td>560</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:00.01</td>
<td>560</td>
<td>1263K</td>
<td>1263K</td>
<td>1302K (0)</td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:00.01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
<td>1</td>
<td>91138</td>
<td>00:00:00:14</td>
<td>553</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Predicate Information (identified by operation id):
```
1 - access("T1"."ID"="T2"."ID2")
```

Note
-----
- this is an adaptive plan

estimation didn't improved but plan has changed to HJ
Adaptive plans – example - adaptive steps

SQL_ID 5bvpk0nnd4323, child number 0

SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data -- t2.data FROM t2, t1 WHERE t1.id = t2.id2

Plan hash value: 324465990

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
<th>OMem</th>
<th>LMem</th>
<th>Used-Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>1098</td>
<td>00:00:00:01</td>
<td>560</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:01</td>
<td>560</td>
<td>1263K</td>
<td>1263K</td>
<td>1302K (0)</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>INDEX FULL SCAN</td>
<td>T2_PK</td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00:01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>INDEX UNIQUE SCAN</td>
<td>T1_PK</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00:00:00:01</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00:00:00:01</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
<td>1</td>
<td>1</td>
<td>91138</td>
<td>00:00:00:14</td>
<td>553</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

- access("T1"."ID"="T2"."ID2")
- access("T1"."ID"="T2"."ID2")

Note

- this is an adaptive plan (rows marked '-' are inactive)
Adaptive plans – statistics collector step

Nested loop

Sub plans are computed in cursor

Hash Join

Statistics collector

Rows are buffered by collector
After a threshold a decision is made

T2 access
index T2_PK

T1 access
Index T1_PK

T1 access
FULL
Adaptive plans – statistics collector step

Once plan is decided, collector is no longer used

- Nested loop
- Hash Join
- T2 access index T2_PK
- T1 access Index T1_PK
- T1 access FULL
Adaptive plans – extra work?

• Sub plans are parsed
• During first execution, plan change can be decided. It does not restart the query, as rows already processed are used
• Following executions use the generated plan, no adaptive optimization is performed

```
SELECT sql_id, child_number, is_bind_sensitive BS, is_bind_aware BA, IS_REOPTIMIZABLE RE,
       IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM v$sql
WHERE sql_id='5bvpk0nnd4323';
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5bvpk0nnd4323</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>5</td>
</tr>
</tbody>
</table>
Adaptive plans – nested loop instead of HJ

SQL_ID 5bvpk0nd4323, child number 0

-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --, t2.data FROM t1, t2 WHERE t1.id = t2.id2

Plan hash value: 1142304008

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>- 1</td>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td>1</td>
<td>99</td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td>1</td>
<td>99</td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td>1</td>
<td>99</td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>- 4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>5</td>
<td>INDEX FULL SCAN</td>
<td>T2_PK</td>
<td></td>
<td>1</td>
<td>99</td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>* 6</td>
<td>INDEX UNIQUE SCAN</td>
<td>T1_PK</td>
<td></td>
<td>99</td>
<td>1</td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T1</td>
<td></td>
<td>99</td>
<td>1</td>
<td>99</td>
<td>00:00:00.01</td>
</tr>
<tr>
<td>- 8</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00:00:00.01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):
---------------------------------------------------

1 - access("T1"."ID"="T2"."ID2")
6 - access("T1"."ID"="T2"."ID2")

Note
-----
- this is an adaptive plan (rows marked '-' are inactive)

earlier HJ was used when A-rows=1098
Adaptive plans – what triggers HJ/NL?

- Subsets cardinality

- Previous examples:
  - T2 with 99 rows, updated stats -> adaptive NL
  - T2 with 1098 rows, stats saying 99 rows -> adaptive HJ

- Threshold used by collector = inflection point
  - It can be seen on 10053 trace
Adaptive plans – optimizer trace (10053)

SQL> alter session set tracefile_identifier='adjhj2';
SQL> exec dbms_sqldiag.dump_trace(p_sql_id=>'9vbxajh8hsng8',p_child_number=>0,
p_component=>'Compiler',p_file_id=>'');

SQL> select tracefile from v$process
   where addr = (select paddr from v$session where sid = userenv('sid'));

TRACEFILE
----------------------------------------------
/u01/app/oracle/diag/rdbms/db12102/db12102/trace/db12102_ora_2558_adjhj2.trc

[oracle@bigdatalite ~]$ grep -c inflection
/u01/app/oracle/diag/rdbms/db12102/db12102/trace/db12102_ora_2558_adjhj2.trc 67

[oracle@bigdatalite ~]$ grep inflection
/u01/app/oracle/diag/rdbms/db12102/db12102/trace/db12102_ora_2558_adjhj2.trc | tail
AP: Costing Nested Loops Join for inflection point at card 134.68
AP: Costing Hash Join for inflection point at card 134.68
AP: Searching for inflection point at value 134.68
AP: Costing Nested Loops Join for inflection point at card 135.30
AP: Costing Hash Join for inflection point at card 135.30
AP: Costing Hash Join for inflection point at card 135.30
DP: Found point of inflection for NLJ vs. HJ: card = 135.30
Adaptive plans – keeping track of plan changes

PLAN_HASH_VALUE is still useful?

We need to look after FULL_PLAN_HASH_VALUE

```
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv, 
    is_bind_sensitive BS, is_bind_aware BA, 
    IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe 
FROM v$sql 
WHERE sql_id = '5bvpk0nnd4323';
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>PLAN_HASH_VALUE</th>
<th>FULL_PHV</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5bvpk0nnd4323</td>
<td>0</td>
<td>1142304008</td>
<td>1330942548</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>2</td>
</tr>
<tr>
<td>5bvpk0nnd4323</td>
<td>0</td>
<td>324465990</td>
<td>1330942548</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>1</td>
</tr>
<tr>
<td>5bvpk0nnd4323</td>
<td>1</td>
<td>1142304008</td>
<td>830081733</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>2</td>
</tr>
</tbody>
</table>

NOTE: output from different runs
Adaptive plans – displaying plan

- `dbms_xplan.display_cursor` - available formats:
  - `format => 'adaptive'`
    - always accurate
    - without it, final plan is shown not including discarded steps.
    - add ‘allstats last’ to see estimated and actual rows
  - `format => 'report'`
    - only works when OPTIMIZER_ADAPTIVE_REPORTING_ONLY=TRUE
    - display what could have been the plan if adaptive features were enabled

```
select * from table(dbms_xplan.display_cursor(format => 'adaptive allstats last'));
```

- autotrace: does not show the final plan

- `dbms_xplan.display_sql_plan_baseline`
  - not always showing the adaptive steps, or the correct note section
SPM (baselines) with adaptive plan

From docs:
- final plan used is captured as baseline if using automatic capture
- when there is a baseline and a new adaptive plan appears, the default plan is captured and marked as adaptive
- accepted plans are never adaptive
- evolution of an adaptive plan accepts the real used plan after evaluating all possible new plans, deleting the old adaptive one

⇒ Lets test this using same queries from previous examples
SPM (baselines) with adaptive plan

@test-join-nl  -- only 99 rows on T2

alter session set
  optimizer_capture_sql_plan_baselines = TRUE;

-- using same SQL:
SELECT /*+ GATHER_PLAN_STATISTICS */
  t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;
/
  -- executed two times

alter session set
  optimizer_capture_sql_plan_baselines = FALSE;

col signature for 99999999999999999999
col sql_handle for a20
col plan_name for a30

select signature, sql_handle, plan_name, enabled,
  accepted, fixed, adaptive, sql_text
from dba_sql_plan_baselines;

<table>
<thead>
<tr>
<th>SIGNATURE</th>
<th>SQL_HANDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN_NAME</td>
<td>ENA ACC FIX ADA</td>
</tr>
<tr>
<td>SQL_TEXT</td>
<td>---------------</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>12148921326023345300</td>
<td>SQL_a899a39409ca0494</td>
</tr>
<tr>
<td>SQL_PLAN_aj6d3kh4wn14n317a0ac5</td>
<td>YES YES NO NO</td>
</tr>
</tbody>
</table>

select signature, sql_handle, plan_name, enabled,
  accepted, fixed, adaptive, sql_text
from dba_sql_plan_baselines;

<table>
<thead>
<tr>
<th>SIGNATURE</th>
<th>SQL_HANDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN_NAME</td>
<td>ENA ACC FIX ADA</td>
</tr>
<tr>
<td>SQL_TEXT</td>
<td>---------------</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>12148921326023345300</td>
<td>SQL_a899a39409ca0494</td>
</tr>
<tr>
<td>SQL_PLAN_aj6d3kh4wn14n317a0ac5</td>
<td>YES YES NO NO</td>
</tr>
</tbody>
</table>

SELECT /*+ GATHER_PLAN_STATISTICS */
  t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2
SPM (baselines) with adaptive plan

SQL_ID  5bvpk0nnd4323, child number 3

SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --, t2.data FROM t2, t1 WHERE  t1.id = t2.id2

Plan hash value: 1142304008

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td>99</td>
<td>00:00:00.01</td>
<td>122</td>
</tr>
<tr>
<td>1</td>
<td>NESTED LOOPS</td>
<td></td>
<td>99</td>
<td>99</td>
<td>00:00:00.01</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>99</td>
<td>99</td>
<td>00:00:00.01</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INDEX FULL SCAN</td>
<td>T2_PK</td>
<td>1</td>
<td>99</td>
<td>00:00:00.01</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>* 4</td>
<td>INDEX UNIQUE SCAN</td>
<td>T1_PK</td>
<td>99</td>
<td>1</td>
<td>00:00:00.01</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T1</td>
<td>99</td>
<td>1</td>
<td>00:00:00.01</td>
<td>99</td>
<td></td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

4 - access("T1"."ID"="T2"."ID2")

After query is executed

same plan and baseline is used
SPM (baselines) with adaptive plan

```
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv, SQL_PLAN_BASELINE,  
is_bind_sensitive BS, is_bind_aware BA,  
IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe  
FROM v$sql  
WHERE sql_id='5bvpk0nnd4323';
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>PLAN_HASH_VALUE</th>
<th>FULL_PHV</th>
<th>SQL_PLAN_BASELINE</th>
<th>BS</th>
<th>BA</th>
<th>RE</th>
<th>AP</th>
<th>EXE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5bvpk0nnd4323</td>
<td>0</td>
<td>1142304008</td>
<td>1330942548</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>1</td>
</tr>
<tr>
<td>5bvpk0nnd4323</td>
<td>1</td>
<td>1142304008</td>
<td>1330942548</td>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>2</td>
</tr>
<tr>
<td>5bvpk0nnd4323</td>
<td>3</td>
<td>1142304008</td>
<td>830081733</td>
<td>SQL_PLAN_aj6d3kh4wn14n317a0ac5</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

⇒ Baseline plan is the same as adaptive but FULL_PLAN_HASH_VALUE is different
  – FPHV **830081733** is the non-adaptive seen earlier
SPM (baselines) with adaptive plan

What if we add more data to T2 as we did before, plan changes to HJ?
⇒ No, baseline is used and no new plan is captured

If the query uses bind variables, plan changes to HJ?
⇒ now the adaptive plan appears as a non accepted baseline

```sql
var t number;
exec :t := 2;
SELECT ...
WHERE t1.id = t2.id2 and t1.type=:
t;
```
SPM (baselines) with adaptive plan - binds

This is the query with binds when it runs without any baseline, T2 with 1098 rows

SQL_ID a54t8xmmcpqa, child number 0

-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data, t2.data FROM t2, t1 WHERE t1.id = t2.id2 and t1.type=:t

Plan hash value: 2959412835

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
<th>OMem</th>
<th>lMem</th>
<th>Used-Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td></td>
<td>11</td>
<td>00:00:00.01</td>
<td>488</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*  1</td>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>00:00:00.01</td>
<td>488</td>
<td>1263K</td>
<td>1263K</td>
<td>1286K (0)</td>
</tr>
<tr>
<td>-   2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00.01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-   3</td>
<td>NESTED LOOPS</td>
<td></td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00.01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-   4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td>1</td>
<td></td>
<td>1098</td>
<td>00:00:00.01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
<td>1</td>
<td>99</td>
<td>1098</td>
<td>00:00:00.01</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-   6</td>
<td>INDEX UNIQUE SCAN</td>
<td>T1_PK</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00:00:00.01</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-   7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00:00:00.01</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*   8</td>
<td>TABLE ACCESS FULL</td>
<td>T1</td>
<td>1</td>
<td>1</td>
<td>912</td>
<td>00:00:00:00.01</td>
<td>481</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note
-----
- this is an adaptive plan (rows marked '-' are inactive)
SPM (baselines) with adaptive plan - binds

SCOTT@db12102/12.1.0.2> SELECT sql_id, child_number, exact_matching_signature, plan_hash_value,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP
FROM v$sql
where sql_id='a54t8xnmpcqza';

SQL_ID        CHILD_NUMBER EXACT_MATCHING_SIGNATURE PLAN_HASH_VALUE BS BA RE AP
------------- -------------- ------------------------ --------------- -- -- -- --
a54t8xnmpcqza 0    3456246857634883437      2959412835  Y  N  N  Y  ← normal exec, no baseline
a54t8xnmpcqza 1    3456246857634883437      2959412835  Y  N  N  Y  ← baseline capture is enabled
a54t8xnmpcqza 2    3456246857634883437      2959412835  N  N  N  N  ← no capture, baseline enabled

select signature, sql_handle, plan_name, enabled, accepted, fixed, adaptive
from dba_sql_plan_baselines;

SIGNATURE SQL_HANDLE PLAN_NAME              ENA ACC FIX ADA  ← new plan captured
--------------------- ------------------------------- -- -- -- --
3456246857634883437 SQL_2ff70e347d63276d SQL_PLAN_2zxsf6jyq69vdaf65d2833 YES NO NO NO
3456246857634883437 SQL_2ff70e347d63276d SQL_PLAN_2zxsf6jyq69vdce68bf8c YES YES NO NO
SPM (baselines) with adaptive plan - binds

select * from table(dbms_xplan.display_sql_plan_baseline (plan_name => SQL_PLAN_2zxsf6jyq69vdaf65d283'));

--------
Plan name: SQL_PLAN_2zxsf6jyq69vdaf65d283     Plan id: 2947928707
Enabled: YES     Fixed: NO     Accepted: NO     Origin: AUTO-CAPTURE
Plan rows: From dictionary
--------

Plan hash value: 2384946331

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>12</td>
<td>552</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>1</td>
<td>NESTED LOOPS</td>
<td></td>
<td>12</td>
<td>552</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>12</td>
<td>552</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS BY INDEX ROWID BATCHED</td>
<td>T1</td>
<td>12</td>
<td>396</td>
<td>2 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>INDEX RANGE SCAN</td>
<td>T1_IDX</td>
<td>12</td>
<td></td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>INDEX UNIQUE SCAN</td>
<td>T2_PK</td>
<td>1</td>
<td></td>
<td>0 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T2</td>
<td>1</td>
<td>13</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

4  - access("T1"."TYPE"=:T)
5  - access("T1"."ID"="T2"."ID2")
SPM (baselines) with adaptive plan - binds

```sql
select * from table(dbms_xplan.display_sql_plan_baseline (plan_name => 'SQL_PLAN_2zxsf6jyq69vdcea8bf8c'));
```

Plan name: SQL_PLAN_2zxsf6jyq69vdcea8bf8c  Plan id: 3467165580
Enabled: YES  Fixed: NO  Accepted: YES  Origin: AUTO-CAPTURE
Plan rows: From dictionary

Plan hash value: 2959412835

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
<td>99</td>
<td>4554</td>
<td>102 (100)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - access("T1"."ID"="T2"."ID2")

Note
---
- this is an adaptive plan

Original plan

Plan steps are not complete?
=> It is when using format ‘adaptive’

Plan steps are not complete?
=> It is when using format ‘adaptive’

Original plan

Plan steps are not complete?
=> It is when using format ‘adaptive’

Plan steps are not complete?
=> It is when using format ‘adaptive’

Plan steps are not complete?
=> It is when using format ‘adaptive’

Plan steps are not complete?
=> It is when using format ‘adaptive’
SPM (baselines) with adaptive plan - evolution

exec : e := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(SQL_HANDLE=>'SQL_2ff70e347d63276d', COMMIT => 'NO');

Task Information:
---------------------------------------------
Task Name : TASK_263
Task Owner : SCOTT
Execution Name : EXEC_263
Execution Type : SPM EVOLVE
Scope : COMPREHENSIVE
Status : COMPLETED
Started : 12/03/2015 06:15:01
Finished : 12/03/2015 06:15:01
Last Updated : 12/03/2015 06:15:01
Global Time Limit : 2147483646
Per Plan Time Limit : UNUSED
Number of Errors : 0

SUMMARY SECTION
------------------------------------------------------------
Number of plans processed : 1
Number of findings : 1
Number of recommendations : 1
Number of errors : 0

DETAILS SECTION
---------------------------------------------------------------------
Object ID : 2
Test Plan Name : SQL_PLAN_2zxsf6jyq69vdafb5d283
Base Plan Name : SQL_PLAN_2zxsf6jyq69vdcea8bf8c
SQL Handle : SQL_2ff70e347d63276d
Parsing Schema : SCOTT
Test Plan Creator : SCOTT
SQL Text : SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data, t2.data FROM t2, t1 WHERE t1.id = t2.id2 and t1.type=:t

Execution Statistics:
-------------------------------
<table>
<thead>
<tr>
<th>Base Plan</th>
<th>Test Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time (s)</td>
<td>.000289</td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>.000278</td>
</tr>
<tr>
<td>Buffer Gets</td>
<td>48</td>
</tr>
<tr>
<td>Optimizer Cost</td>
<td>139</td>
</tr>
<tr>
<td>Disk Reads</td>
<td>0</td>
</tr>
<tr>
<td>Direct Writes</td>
<td>0</td>
</tr>
<tr>
<td>Rows Processed</td>
<td>0</td>
</tr>
<tr>
<td>Executions</td>
<td>10</td>
</tr>
</tbody>
</table>
SPM (baselines) with adaptive plan - evolution

FINDINGS SECTION

Findings (1):

1. The plan was verified in 0.08000 seconds. It passed the benefit criterion because its verified performance was 25.63122 times better than that of the baseline plan.

Recommendation:

Consider accepting the plan.

Same two plans shown earlier follows in that report, none of them adaptive
Adaptive plans – challenges

- Several ways to see the plan used by a query
  - Autotrace does not show the final plan always
  - dbms_xplan.display_sql_plan_baseline has problems too

- Several execution plans for the same query
  - FULL_PLAN_HASH_VALUE needs to be used
  - But it is different for the same “sql/final plan” if adaptive features were not used

- When table cardinality changes, plans can change, but only after hard parsing
Review of Adaptive optimizations features

- Adaptive statistics
  - Dynamic Statistics
  - Automatic Reoptimization
  - SQL Plan Directives
- Adaptive plans
  - Join Methods
- Parallel Distribution Methods
Adaptive plans - Parallel Distribution Methods

- New hybrid hash distribution method
  - defers decision to use hash or broadcast to execution time
- Collector step in front of parallel coordinator
- Threshold is 2*DOP
- Enabled by default
Adaptive plans - Adaptive bitmap pruning

- This is not documented under query optimizer adaptive features

- New parameter `optimizer_strans_adaptive_pruning` - allow adaptive pruning of star transformation bitmap trees

- Article by Frank Pachot:
  [http://www.slideshare.net/pachot/nl-2014-4adaptivebitmappruning](http://www.slideshare.net/pachot/nl-2014-4adaptivebitmappruning)
Questions?

📧 calero@pythian.com

🐦 @ncalerouy

🌐 http://www.linkedin.com/in/ncalero
References - documentation

Database SQL Tuning Guide – Query optimizer concepts
https://docs.oracle.com/database/121/TGSQL/tgsql_optncpt.htm#TGSQL192

Optimizer with Oracle Database 12c

Understanding Optimizer Statistics with Oracle Database 12c

Several people blogging about this topics
- Frank Pachot - http://www.slideshare.net/pachot
- Tim Hall - https://oracle-base.com/articles/12c/adaptive-query-optimization-12cr1

Database Licensing Information - Oracle Database Editions
https://docs.oracle.com/database/121/DBLIC/editions.htm#DBLIC116
References – views to find information

V$SQL -  IS_RESOLVED_ADAPTIVE_PLAN
           IS_REOPTIMIZABLE
           FULL_PLAN_HASH_VALUE

DBA_SQL_PLAN_DIRECTIVES / DBA_SQL_PLAN_DIR_OBJECTS

DBA_SQL_PLAN_BASELINES - ADAPTIVE

dbms_xplan.display – ADAPTIVE format

V$SQL_PLAN/ DBA_HIST_SQL_PLAN - OTHER_XML
  notes of adaptive features used

V$SQL_REOPTIMIZATION_HINTS
References - controlling adaptive features

Initialization parameters for all features:

• OPTIMIZER_ADAPTIVEFEATURES (Default: TRUE)
  Disables all adaptive features
• OPTIMIZER_ADAPTIVE_REPORTING_ONLY (Default: FALSE)
  Only reports what plans should have been if the feature is enable
• OPTIMIZER_FEATURES_ENABLE – 12.1.0.1 or higher

Parameters per feature

• OPTIMIZER_DYNAMIC_SAMPLING = number => 11 is the new adaptive feature
• _OPTIMIZER_ADAPTIVE_PLANS
• _OPTIMIZER_USE_FEEDBACK

Hints (statement level):

/*+ NO_ADAPTIVE_PLAN */ - new in 12.1.0.2, but statistics feedback works
/*+ ADAPTIVE_PLAN */
/*+ DYNAMIC_SAMPLING */
Global statement trace for already existing and new connections:
    dbms_sqldiag.dump_trace(p_sql_id=>'0a14b3yhux040', p_child_number=>0,
        p_component=>'Compiler', p_file_id=>'trace_0a14b3yhux040');
=> This can be used after statement execution

Classic optimizer trace:
    ALTER SESSION SET EVENTS 'trace[sql_optimizer.*]';
-- same but for Oracle versions older than 11g
    ALTER SESSION SET EVENTS='10053 trace name context forever, level 1';

SPM Trace at session level:
    ALTER SESSION SET EVENTS 'trace[RDBMS.SQL_Plan_Management.*]';
drop table t1 purge;
create table t1 as
    select rownum id, mod(rownum,100) type,
        object_name data
    from dba_objects;
alter table t1 add constraint t1_pk
    primary key (id);

drop table t2 purge;
create table t2 as
    select rownum id2, mod(rownum,10) type,
        object_name data
    from dba_objects
    where rownum < 100;
alter table t2 add constraint t2_pk
    primary key (id2);

exec dbms_stats.gather_table_stats(USER, 't1');
exec dbms_stats.gather_table_stats(USER,'t2');

set autotrace off
col data for a30
SELECT /*+ GATHER_PLAN_STATISTICS */
    t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

select * from table
    (dbms_xplan.display_cursor
    (format=>'adaptive allstats last'));

References - script used - Adaptive plans
References - script used - Adaptive plans

```
insert into t2
select rownum+100,
    mod(rownum,10) type,
    object_name data
from dba_objects
where rownum < 1000;

-- no need to update stats for plan to change

set autotrace off
col data for a30
SELECT /*+ GATHER_PLAN_STATISTICS */
    t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

select * from table(dbms_xplan.display_cursor
    (format=>'adaptive allstats last'));

-- Plan didn’t changed here, still NL used
-- after parsing sql again, hash_join kicks in

alter system flush shared_pool;

col data for a30
SELECT /*+ GATHER_PLAN_STATISTICS */
    t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

select * from table(dbms_xplan.display_cursor
    (format=>'adaptive allstats last'));

SELECT sql_id, child_number, plan_hash_value,
    full_plan_hash_value full_phv,
    is_bind_sensitive BS, is_bind_aware BA,
    IS_REOPTIMIZABLE RE, executions exe,
    IS_RESOLVED_ADAPTIVE_PLAN AP
FROM   v$sql
WHERE   sql_id='5bvpk0nnd4323';
```

test-join-hj.sql