Same Plan
Different Performance

Mauro Pagano

Mauro Pagano

- Consultant/Developer/Analyst
- Oracle → Enkitec → Accenture
- DBPerf and SQL Tuning
- Training
- Tools (SQLT, SQLd360, PLNFND)

SQL is slower….

- Same SQL experiences different performance in systems that are identical (or supposed to)
- First check the execution plan
- Most of the time plan is different, address it
- But what if the plan is the same?

CBO is innocent (this time, maybe)

- Exec plan is where CBO's job end (kind of)
- Same plan means CBO *"worked"* the same
- Doesn't mean everything else ***IS*** the same
- Shift focus on next step, SQL execution

Apples vs oranges?

- Make sure the comparison is fair (data)
- All external factors should be similar
  - CPU should be similar
  - IO should be similar
  - Memory should be similar

“Everything is the same!”

- Plan, data and hardware match, now what?
- Dig into how the SQL is executed
- Wait events and session statistics
- Factors
  - – configuration, storage layout, load

Old friends get-together

- Wait events
  - Do they match?
  - Are they close in cardinality?
  - Do we spend the same time on them?
- Session statistics
  - Do they match?
  - Are they close in values?

Back to the plan for a second

- Exec plan is made of lots of small steps
- Each one produces/handles/consumes rows
- Same behaviors in short and long plans
- Keep it simple, focus on the step
- Remove the noise if possible (reduce TC)

Each scenario is a quiz

- SQL is provided
- Changes to the initial setup are disclosed
- Each run in one environment

- Identify what’s different and why

Setup

- Linux x86-64, 11.2.0.3
- 1 table, 1M rows, 3 columns, no index
  - N1 unique
  - N2 100 NDV
  - C1 100chars long padded string
- Identical hardware, same DDL to create table
- Controlled environments to isolate behavior
- Simplest SQL to reproduce desired behavior

Scenario #1

- SQL

  -
    ```
    select /*+ INDEX(TEST1M) */ count(*)
      from test1m
    where n1 between 1 and 1000000
    ```

- Environment
  - Added index on N1

Scenario #1 – Run (A)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.31       0.45       2228       2228          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.32       0.45       2228       2228          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=2228 pr=2228 pw=0 time=451619 us)
   1000000    1000000    1000000   INDEX RANGE SCAN TEST1M_IDX (cr=2228 pr=2228

 Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
db file sequential read                        2228        0.00          0.15
```

Scenario #1 – Run (B)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.07       0.08          0       2228          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.08       0.08          0       2228          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=2228 pr=0 pw=0 time=80038 us)
   1000000    1000000    1000000   INDEX RANGE SCAN TEST1M_IDX (cr=2228 pr=0 pw=0
```

Scenario #1 Solution

- Buffer Cache cold/warm
- (Part of) the data already in memory
- Reduced number of physical reads (pr)
- Faster performance because less reads
- Number of (same) wait events is lower
- Isolated environment likely to read more

Scenario #2

- SQL
  - select /*+ FULL(TEST1M) */ count(*) from test1m
- Environment
  - No changes from original setup

Scenario #2 – Run (A)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.57       1.51      28574      28584          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.57       1.52      28574      28584          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=28584 pr=28574 pw=0 time=1513999 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=28584 pr=28574

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                       1        0.00          0.00
  db file scattered read                      240        0.02          1.07
```

Scenario #2 – Run (B)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      1.04       2.42      14286      28583          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      1.04       2.42      14286      28583          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=28583 pr=14286 pw=0 time=2424726 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=28583 pr=14286

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                      5732        0.01          0.89
  db file scattered read                       4277        0.00          0.75
```

Scenario #2 – Run (A) - Waits

```
WAIT #140245916217600: nam='db file scattered read' ela= 4834 file#=26 block#=16002 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 4020 file#=26 block#=16130 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 2452 file#=26 block#=16258 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 8712 file#=26 block#=16386 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 6417 file#=26 block#=16514 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 2267 file#=26 block#=16642 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 2637 file#=26 block#=16770 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 2304 file#=26 block#=16898 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 1809 file#=26 block#=17026 blocks=126
WAIT #140245916217600: nam='db file scattered read' ela= 2661 file#=26 block#=17154 blocks=126
```

Scenario #2 – Run (B) - Waits

```
WAIT #140245916165224: nam='db file sequential read' ela= 124 file#=26 block#=16002 blocks=1
WAIT #140245916165224: nam='db file scattered read' ela= 139 file#=26 block#=16004 blocks=2
WAIT #140245916165224: nam='db file sequential read' ela= 117 file#=26 block#=16007 blocks=1
….<<another 38 waits here>>
WAIT #140245916165224: nam='db file sequential read' ela= 132 file#=26 block#=16113 blocks=1
WAIT #140245916165224: nam='db file sequential read' ela= 123 file#=26 block#=16116 blocks=1
WAIT #140245916165224: nam='db file scattered read' ela= 142 file#=26 block#=16118 blocks=2
WAIT #140245916165224: nam='db file scattered read' ela= 141 file#=26 block#=16121 blocks=2
WAIT #140245916165224: nam='db file scattered read' ela= 135 file#=26 block#=16124 blocks=2
WAIT #140245916165224: nam='db file sequential read' ela= 119 file#=26 block#=16127 blocks=1
```

Scenario #2 Solution

- Buffer cache status (cold/warm)
- (Part of) the data already in memory
- Reduced number of physical reads (pr)
- Number of (same) wait events is higher
- Wait events details help track it down
  - Non-contiguous blocks read
- Slower performance because smaller reads

Scenario #3

- SQL
  - select /*+ FULL(TEST1M) */ count(*)
    from test1m
- Environment
  - No changes
  - BC warm

Scenario #3 – Run (A)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.92       2.96      14286      28583          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.92       2.96      14286      28583          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=28583 pr=14286 pw=0 time=2967930 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=28583 pr=14286

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                      5732        0.10          1.17
  db file scattered read                       4277        0.28          1.13
```

Scenario #3 – Run (B)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.11       1.01      28573      28575          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.11       1.02      28573      28575          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=28575 pr=28573 pw=0 time=1019952 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=28575 pr=28573

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  enq: KO - fast object checkpoint                2        0.00          0.00
  direct path read                              179        0.03          0.90
```

Scenario #3 – Solution

- Buffered vs Direct Path reads (different waits too)
- (Part of) the data already in memory
- Direct Path
  - skips Buffer Cache and reads whole table every time
  - consistent performance
  - number of wait events is consistent
- Buffered vs Direct Path decision is made AFTER plan selection (several criteria)

Scenario #4

- SQL
  - select /*+ FULL(TEST1M) */ count(*) from test1m
- Environment
  - No changes
  - BC cold

Scenario #4 – Run (A)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.57       3.08      15872      15884          1           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.57       3.10      15872      15884          1           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=15884 pr=15872 pw=0 time=3086869 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=15884 pr=15872

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file scattered read                       2005        0.05          2.53
```

Scenario #4 – Run (B)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.32       1.66      15872      15881          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.32       1.66      15872      15881          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=15881 pr=15872 pw=0 time=1660864 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=15881 pr=15872

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file scattered read                        141        0.05          1.41
```

Scenario #4 – Run (A) - Waits

```
WAIT #139702845969088: nam='db file scattered read' ela= 265 file#=25 block#=306 blocks=8
WAIT #139702845969088: nam='db file scattered read' ela= 257 file#=25 block#=314 blocks=8
WAIT #139702845969088: nam='db file scattered read' ela= 259 file#=25 block#=322 blocks=8
WAIT #139702845969088: nam='db file scattered read' ela= 254 file#=25 block#=330 blocks=8
.....
WAIT #139702845969088: nam='db file scattered read' ela= 217 file#=25 block#=378 blocks=6
WAIT #139702845969088: nam='db file scattered read' ela= 270 file#=25 block#=386 blocks=8
WAIT #139702845969088: nam='db file scattered read' ela= 283 file#=25 block#=394 blocks=8
WAIT #139702845969088: nam='db file scattered read' ela= 263 file#=25 block#=402 blocks=8
```

Scenario #4 – Run (B) - Waits

```
WAIT #139702846026760: nam='db file scattered read' ela= 13508 file#=25 block#=258 blocks=126
WAIT #139702846026760: nam='db file scattered read' ela= 9016 file#=25 block#=386 blocks=126
```

Scenario #4 – Solution 1

- Different *db_file_multiblock_read_count* value
- Same number of blocks read from disk
- Number of (same) wait events is higher
- Wait events details help track it down
  - Contiguous blocks read
- Slower performance because smaller reads

Scenario #4 – Solution 2

- Different extent size (64k vs 1M)
- Same number of blocks read from disk
- Number of (same) wait events is higher
- Wait events details help track it down
  - Contiguous blocks read
- Same params/stats but different storage org
- Slower performance because smaller reads

Scenario #5

- SQL
  - select /*+ FULL(TEST1M) */ count(*)
    from test1m
- Env changes
  - No changes
  - BC cold, MBRC and extent are identical

Scenario #5 – Run (A)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0          0
Execute      1      0.00       0.00          0          0          0          0
Fetch        2      0.26       0.72      14285      14297          1          1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.27       0.72      14285      14297          1          1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=14297 pr=14285 pw=0 time=723883 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=14297 pr=14285

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file scattered read                        128        0.04          0.51
```

Scenario #5 – Run (B)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.44       1.29      28574      28586          1           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.44       1.29      28574      28586          1           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=28586 pr=28574 pw=0 time=1291333 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=28586 pr=28574

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file scattered read                        240        0.04          0.95
```

Scenario #5 – Solution 1

- Different PCTFREE (0 vs 50)
- Higher number of blocks read for same data
- Reads are of the same size hence more reads
- Data is more spread out, room for changes
- Slower performance because more reads

Scenario #5 – Solution 2

- Empty blocks below HWM
- Higher number of blocks read for same data
- Reads are of the same size hence more reads
- Data has been deleted, FTS reads everything
- Slower performance because more reads

Scenario #6

- SQL
  - select /*+ FULL(TEST1M) */ count(*)
    from test1m

- Env changes
  - No changes
  - BC cold, MBRC, PCTFREE and extent are identical

Scenario #6 – Run (A)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.44       1.29      28574      28586          1           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.44       1.29      28574      28586          1           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=28586 pr=28574 pw=0 time=1291333 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=28586 pr=28574

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file scattered read                        240        0.04          0.95
```

Scenario #6 – Run (B)

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.73       2.49      28803      58584          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.74       2.49      28803      58584          0           1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=58584 pr=28803 pw=0 time=2492596 us)
   1000000    1000000    1000000   TABLE ACCESS FULL TEST1M (cr=58584 pr=28803

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file scattered read                        240        0.23          1.73
  cell single block physical read               230        0.01          0.06
```

Scenario #6 – Waits and SesStats

- Wait events show
  - single block reads from UNDO tbs for obj#=0

```
WAIT #140029131327704: nam='db file scattered read' ela= 15412 file#=26 block#=15618 blocks=126 obj#=74828
WAIT #140029131327704: nam='cell single block physical read' ela= 220 … bytes=8192 obj#=0
WAIT #140029131327704: nam='db file scattered read' ela= 11786 file#=26 block#=15746 blocks=126 obj#=74828
WAIT #140029131327704: nam='cell single block physical read' ela= 233 … bytes=8192 obj#=0
WAIT #140029131327704: nam='db file scattered read' ela= 5938 file#=26 block#=15874 blocks=126 obj#=74828
WAIT #140029131327704: nam='cell single block physical read' ela= 224 … bytes=8192 obj#=0
WAIT #140029131327704: nam='db file scattered read' ela= 12162 file#=26 block#=16002 blocks=126 obj#=74828
```

- v$sesstat shows high
  - data blocks consistent reads - undo records applied

Scenario #6 - Solution

- Different concurrency/workload
- Higher number of blocks read for same data
- Waits -> reads from UNDO tbs
- SesStats -> UNDO records applied
- Slower performance because more reads + more work to recreate the correct image

Scenario #7

- SQL
  -
    ```
    select /* 1st run */ n1,c1
       from test1m
     where n1 in (1,1000,5000)
    ```

- Env changes
  - Index on TEST1M(N1)
  - BC cold, MBRC, PCTFREE and extent are identical
  - No concurrency at the time SQL is executed

Scenario #7 – Why so many cr/pr?

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.75         11         18          0           3
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.75         11         18          0           3


Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         3          3          3  INLIST ITERATOR  (cr=18 pr=11 pw=0 time=235681 us)
         3          3          3   TABLE ACCESS BY INDEX ROWID TEST1M (cr=18 pr=11
         3          3          3    INDEX RANGE SCAN TEST1M_IDX (cr=9 pr=5 pw=0

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
db file sequential read                         11        0.18          0.52
```

Scenario #7 – Waits and SesStats

- Wait events show
  - single block reads from data tbs, same obj#

```
WAIT #140...: nam='db file sequential read' ela= 7414 file#=26 block#=2356 blocks=1 obj#=75022
WAIT #140...: nam='db file sequential read' ela= 41395 file#=26 block#=131 blocks=1 obj#=74828
WAIT #140...: nam='db file sequential read' ela= 181594 file#=26 block#=78403 blocks=1 obj#=74828
```

- v$sesstat shows high
  - table fetch continued row

Scenario #7 - Solution

- Row migration, index points to original rowid
- Higher number of blocks read for same data
- Waits -> reads are from data tbs
- SesStats -> table fetch continued row
- Slower performance because more reads + more work to find all the row pieces
- Similar behavior happens with chained rows

Scenario #8

- SQL
  - select /* 2nd run */ n1,c1, ***ora_rowscn***
    from test1m
    where rownum <= 5000

- Env changes
  - Index on TEST1M(N1)
  - BC cold, MBRC, PCTFREE and extent are identical
  - No concurrency at the time SQL is executed

Scenario #8 - Why so many seq read?

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        6      0.03       0.22        393       5378          0        5000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        8      0.03       0.22        393       5378          0        5000

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
      5000       5000       5000  COUNT STOPKEY (cr=5378 pr=393 pw=0 time=91193
      5000       5000       5000   TABLE ACCESS FULL TEST1M (cr=5378 pr=393

Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  db file sequential read                       381        0.00          0.19
  db file scattered read                          2        0.00          0.00
```

Scenario #8 – Waits and SesStats

- Wait events show
  - single block reads from data tbs, same obj#

```
WAIT #1405…: nam='db file scattered read' ela= 6434 file#=26 block#=132 blocks=4 obj#=74828
WAIT #1405…: nam='db file sequential read' ela= 193 file#=26 block#=78670 blocks=1 obj#=74828
WAIT #1405…: nam='db file sequential read' ela= 182 file#=26 block#=78686 blocks=1 obj#=74828
WAIT #1405…: nam='db file sequential read' ela= 3445 file#=26 block#=7890 blocks=1 obj#=74828
```

- v$sesstat shows high
  - table fetch continued row

Scenario #8 - Solution

- Row migration, pseudo col needs row header
- Higher number of blocks read for same data
- Waits -> reads are from data tbs
- SesStats -> table fetch continued row
- Slower performance because more reads + more work to find all the row pieces
- Similar behavior happens with chained rows

Other things to consider

- Same PHV with small differences
  - Predicate ordering
  - Column projection
- Exadata Optimizations
  - Exadata Smart Flash Cache
  - Storage indexes
- External to the database
  - File system / SAN / Disk caching
  - Read-ahead optimizations

Conclusions

- Same plan can still run differently
- Storage organization and concurrency impact
- Fix one scenario can introduce another, ie.
  - low PCTFREE higher chance of row migration
  - high caching slows down buffered mreads
- Find a balance to achieve optimal performance

Q&A

References

- 'DB_FILE_MULTIBLOCK_READ_COUNT' AND EXTENTS MANAGEMENT (Doc ID 181272.1)
- Higher 'direct path read' Waits in 11g when Compared to 10g (Doc ID 793845.1)
- Why Is My Query Sometimes Slower Than Other Times with Higher Consistent Gets Although No Change in Execution Plan? (Doc ID 1558349.1)
- Row Chaining and Row Migration (Doc ID 122020.1)

Contact Information

ORApeeps

- mauro.pagano@gmail.com
- http://mauro-pagano.com
- @mautro