Where's Waldo?

Using a brute-force approach to find an Execution Plan the CBO hides

Carlos Sierra and Mauro Pagano

Hard parsing review

1. Syntax validation
2. Semantics validation
3. Shared pool search
4. Shared pool memory allocation
5. Get bind variables values
6. Optimize query execution
7. Build parse tree and execution plan
8. Store tree and plan in shared pool

Cost Based Optimizer

"I do the very best I know how—the very best I can; and I mean to keep doing so until the end."

Abraham Lincoln

Motivation

- CBO produces optimal execution plans
  - Almost always
- CBO produces consistent execution plans
  - Sometimes an execution plan changes
- AWR captures good and bad execution plans
  - Only a small set of plans is actually captured
- Sometimes we cannot easily reproduce a good plan!

How many plans?

```
SELECT /* ^^pathfinder_testid */
       e1.email, jh.job_id
  FROM employees e1,
       job_history jh
 WHERE e1.employee_id = jh.employee_id
   AND jh.start_date > '01-JAN-01'
   AND e1.salary > (SELECT AVG(e2.salary)
                      FROM employees e2
                     WHERE e1.department_id = e2.department_id)
   AND e1.department_id IN (SELECT d.department_id
                              FROM departments d,
                                   locations l
                             WHERE d.location_id = l.location_id
                               AND l.country_id = 'US')
/
```

- $a + (b \times T)!$
- Where a and b are constants, T is the number of Tables, and "!" represents "factorial"

Why a plan changes?

- Maybe access paths?
- Maybe CBO statistics?
- Maybe bind variable values?
  - Exacerbated with histograms
- Maybe CBO version and configuration?
- Maybe CBO is pure evil!

CBO versions and configuration

- 7.3, 8.0, 8.1, 9.0, 9.1, 9.2, 10.0, 10.1, 10.2, 11.0, 11.1, 11.2, 12.0, 12.1, 12.2, … plus patch sets
- Parameters
  - optimizer_features_enable
  - optimzer_*
  - _fix_control
  - Others (regular and hidden)

```
ESPQA:demo1> show parameters optimizer

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------
_optimizer_use_feedback              boolean     TRUE
optimizer_capture_sql_plan_baselines boolean     FALSE
optimizer_dynamic_sampling           integer     2
optimizer_features_enable            string      11.2.0.3
optimizer_index_caching              integer     0
optimizer_index_cost_adj             integer     100
optimizer_mode                       string      ALL_ROWS
optimizer_secure_view_merging        boolean     TRUE
optimizer_use_invisible_indexes      boolean     FALSE
optimizer_use_pending_statistics     boolean     FALSE
optimizer_use_sql_plan_baselines     boolean     TRUE
```

Why the CBO “hides” my plan?

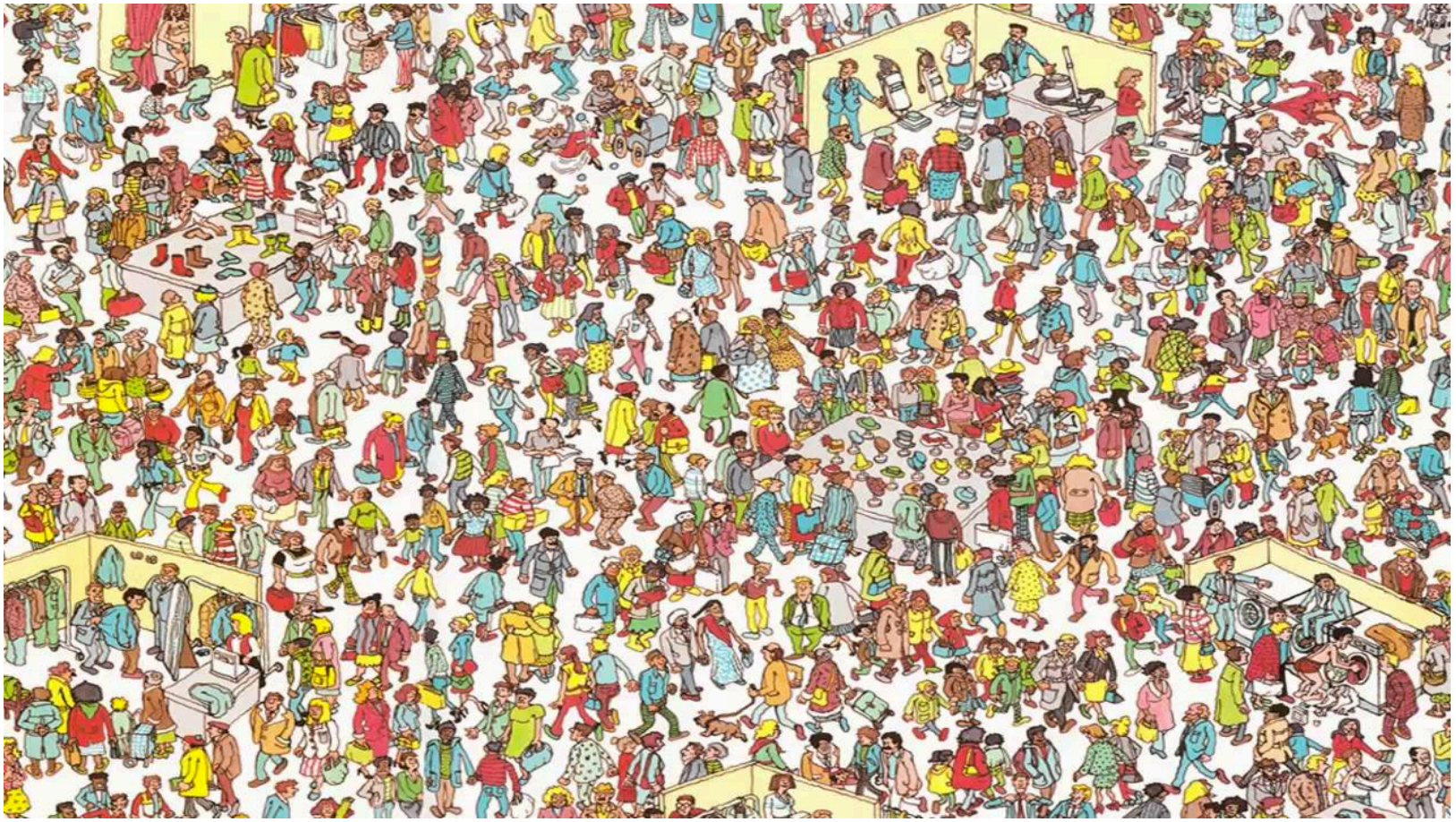
- CBO only evaluates a subset of possible plans
  - Reduce parse time by reducing plans to evaluate
- “Optimal” plan means
  - The one with lowest cost
- Cost depends on
  - Representative CBO statistics (and bind values)
  - Simplified modeling and arbitrary heuristics

What if?

- We loop over all CBO parameters
- For each iteration we produce an execution plan
- Execute each plan and measure its performance
- Report all executions and their performance

- *Then maybe we get to produce a good plan the CBO failed to relinquished as an optimal plan*

Like finding Waldo!



Meet Pathfinder

- Free software *(by Mauro Pagano)*
- Installs nothing
- Uses brute-force analysis to find “Waldo”
  - Execution plans for most CBO related parameters
    - Near 1,000 on 11.2.0.3 and over 1,500 on 12.1.0.2
    - Most produce same execution plan (baseline)
      - Some might produce more than one plan due to cardinality feedback
- Reports all executions (plan and performance)

Three reasons to use Pathfinder

- To discover several plans not selected by the CBO
  - Incidentally some may perform better than baseline
- To identify what causes a wrong result
  - Which query transformation or patch is causing it
- To narrow reason of a long parse
  - When executed on SQL that returns no rows

Pathfinder v1501 (2015-10-02): Plan Finder

dbname:demo connect string:sys/enk1tec@demo as sysdba startime:2015/11/13 08:06:07

Test#	CFB Reparse#	Setting	Plan Hash Value	Elapsed Time	CPU Time	Buffer Gets	Rows Processed	Execution Plan	V$SQL Details
1		BASELINE	566236885	.081	.075	760	2	plan	details
2		ALTER SESSION SET "_add_stale_mv_to_dependency_list" = FALSE;	566236885	.039	.038	116	2	plan	details
3		ALTER SESSION SET "_allow_level_without_connect_by" = TRUE;	566236885	.041	.04	116	2	plan	details
4		ALTER SESSION SET "_always_anti_join" = 'CHOOSE';	566236885	.038	.036	116	2	plan	details
5		ALTER SESSION SET "_always_anti_join" = 'HASH';	566236885	.039	.039	116	2	plan	details
6		ALTER SESSION SET "_always_anti_join" = 'MERGE';	566236885	.042	.042	116	2	plan	details
7		ALTER SESSION SET "_always_anti_join" = 'NESTED_LOOPS';	566236885	.04	.039	116	2	plan	details
8		ALTER SESSION SET "_always_anti_join" = 'OFF';	566236885	.039	.038	116	2	plan	details
9		ALTER SESSION SET "_always_semi_join" = 'CHOOSE';	566236885	.037	.036	116	2	plan	details
10		ALTER SESSION SET "_always_semi_join" = 'HASH';	566236885	.038	.038	116	2	plan	details
11		ALTER SESSION SET "_always_semi_join" = 'MERGE';	566236885	.038	.037	116	2	plan	details
12		ALTER SESSION SET "_always_semi_join" = 'NESTED_LOOPS';	566236885	.038	.037	116	2	plan	details
13		ALTER SESSION SET "_always_semi_join" = 'OFF';	566236885	.035	.034	116	2	plan	details
14		ALTER SESSION SET "_always_star_transformation" = TRUE;	566236885	.042	.042	116	2	plan	details
15		ALTER SESSION SET "_and_pruning_enabled" = FALSE;	566236885	.039	.039	116	2	plan	details
16		ALTER SESSION SET "_b_tree_bitmap_plans" = FALSE;	566236885	.037	.036	116	2	plan	details
17		ALTER SESSION SET "_bloom_filter_enabled" = FALSE;	566236885	.041	.041	116	2	plan	details
18		ALTER SESSION SET "_bloom_folding_enabled" = FALSE;	566236885	.039	.038	116	2	plan	details
19		ALTER SESSION SET "_bloom_minmax_enabled" = FALSE;	566236885	.039	.038	116	2	plan	details
20		ALTER SESSION SET "_bloom_predicate_enabled" = FALSE;	566236885	.039	.038	116	2	plan	details
21		ALTER SESSION SET "_bloom_predicate_pushdown_to_storage" = FALSE;	566236885	.039	.037	116	2	plan	details
22		ALTER SESSION SET "_bloom_pruning_enabled" = FALSE;	566236885	.039	.038	116	2	plan	details
23		ALTER SESSION SET "_bt_mmv_query_rewrite_enabled" = FALSE;	566236885	.042	.041	116	2	plan	details
24		ALTER SESSION SET "_complex_view_merging" = FALSE;	566236885	.037	.036	114	2	plan	details
25		ALTER SESSION SET "_connect_by_use_union_all" = 'FALSE';	566236885	.039	.038	116	2	plan	details
26		ALTER SESSION SET "_connect_by_use_union_all" = 'OLD_PLAN_MODE';	566236885	.044	.042	116	2	plan	details
27		ALTER SESSION SET "_connect_by_use_union_all" = 'TRUE';	566236885	.038	.037	116	2	plan	details
28		ALTER SESSION SET "_convert_set_to_join" = TRUE;	566236885	.04	.041	116	2	plan	details
29		ALTER SESSION SET "_cost_equality_semi_join" = FALSE;	566236885	.037	.035	116	2	plan	details
30		ALTER SESSION SET "_db_file_optimizer_read_count" = 4;	566236885	.039	.039	116	2	plan	details

Execution Plan

```
Inst: 2   Child: 0    Plan hash value: 566236885

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                | Starts | E-Rows |E-Bytes| Cost (%CPU)|
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                     |      1 |        |       |    11 (100)|
|   1 |  NESTED LOOPS SEMI                 |                     |      1 |      1 |    76 |    11  (19)|
|   2 |   NESTED LOOPS                     |                     |      1 |      1 |    74 |     9  (23)|
|*  3 |    HASH JOIN                       |                     |      1 |     17 |   765 |     8  (25)|
|   4 |     VIEW                           | VW_SQ_1             |      1 |     11 |   286 |     4  (25)|
|   5 |      HASH GROUP BY                 |                     |      1 |     11 |    77 |     4  (25)|
|   6 |       TABLE ACCESS STORAGE FULL    | EMPLOYEES           |      1 |    107 |   749 |     3   (0)|
|   7 |     TABLE ACCESS STORAGE FULL      | EMPLOYEES           |      1 |    107 |  2033 |     3   (0)|
|*  8 |    TABLE ACCESS BY INDEX ROWID     | JOB_HISTORY         |     38 |      1 |    29 |     1   (0)|
|*  9 |     INDEX RANGE SCAN               | JHIST_EMPLOYEE_IX   |     38 |      1 |       |     0   (0)|
|  10 |   VIEW PUSHED PREDICATE            | VW_NSO_2            |      3 |      1 |     2 |     2   (0)|
|  11 |    NESTED LOOPS                    |                     |      3 |      1 |    13 |     2   (0)|
|  12 |     TABLE ACCESS BY INDEX ROWID    | DEPARTMENTS         |      3 |      1 |     7 |     1   (0)|
|* 13 |      INDEX UNIQUE SCAN             | DEPT_ID_PK          |      3 |      1 |       |     0   (0)|
|* 14 |     TABLE ACCESS BY INDEX ROWID    | LOCATIONS           |      3 |      4 |    24 |     1   (0)|
|* 15 |      INDEX UNIQUE SCAN             | LOC_ID_PK           |      3 |      1 |       |     0   (0)|
-----------------------------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$2E20A9F9
   4 - SEL$683B0107 / VW_SQ_1@SEL$7511BFD2
   5 - SEL$683B0107
   6 - SEL$683B0107 / E2@SEL$2
```


Demo

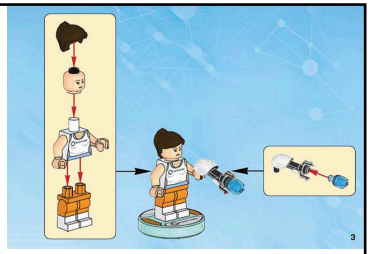
What if I find a better plan?

- Better performing plans are possible even common
- Can I "pin" one of those better plans?
  - You can create a SQL Plan Baseline or a SQL Profile
  - Note:
    - Consider this route only as a temporary workaround
    - Free SPM scripts under cscripts or SQLT

Disclaimers

- Pathfinder may take hours to execute
  - Use only on SQL that takes a few seconds per execution
- Not a replacement for proper SQL Tuning
  - More of a band-aid or discovery exercise
- Use preferably on a Test or Development tier
  - Although it could be used on Production

Pathfinder instructions



- Download from http://mauro-pagano.com/
- Unzip and cd to pathfinder-master
- Put your SQL into script.sql provided
  - SQL must contain /* ^^pathfinder_testid */
- Connect as SYS and execute pathfinder.sql
- Pass user/password of user who can execute script
  - Optionally include TNS alias: user/password@alias

References and Contact

- http://mauro-pagano.com/
  - Download pathfinder
- http://carlos-sierra.net/
  - Download cscripts which include SPM scripts