Allstate®

You're in good hands.

SQL Tuning Toolkit for Advanced DBAs

A customer success story

Mark Fleming

Database Administrator

Oct 2, 2011

Mark Fleming

Oracle DBA 3+ years; DB2 DBA 5+ years;
performance, tuning, design, support
Programmer 20+ years

"So get Allstate. You could save money and be better protected from mayhem, like me."
~ Dean Winters, Mayhem

Allstate Insurance Company

- Leader in Insurance: home, auto, life and other major brands
- Company Rating by Best Insurance Reports A+
- Solid History, Founded in 1931
- 2nd Largest Personal Lines Insurer in the U.S.
- Revenues: $31.4 billion, Net income: $928 million
- 36,000 employees, 12,000 agencies owners and exclusive financial representatives

"That's Allstate's Stand. Are you in Good Hands?"

~ Dennis Haysbert

IT Info

- DBA area: ~ 100+ DBAs supporting ***Oracle***, SQL Server, DB2, and IMS
- 286 Cluster databases, 595+ instances, 160 hosts, 250 Oracle applications

"Shop Less. Get More. Call Allstate."

Case 1: Ambiguous Column

…or Finding the Needle in the Haystack

Situation:

NextGen - Claims transaction processing

One of the largest mission-critical systems at Allstate:
~ 11K-14K unique SQL executed per day
~ 850,000,000 to 1,000,000,000 transactions/day
~ 12.8 TB of data – 24x7 availability

While preparing for 11g upgrade – discover SQL getting ORA-00918: column ambiguously defined in an 11g test environment

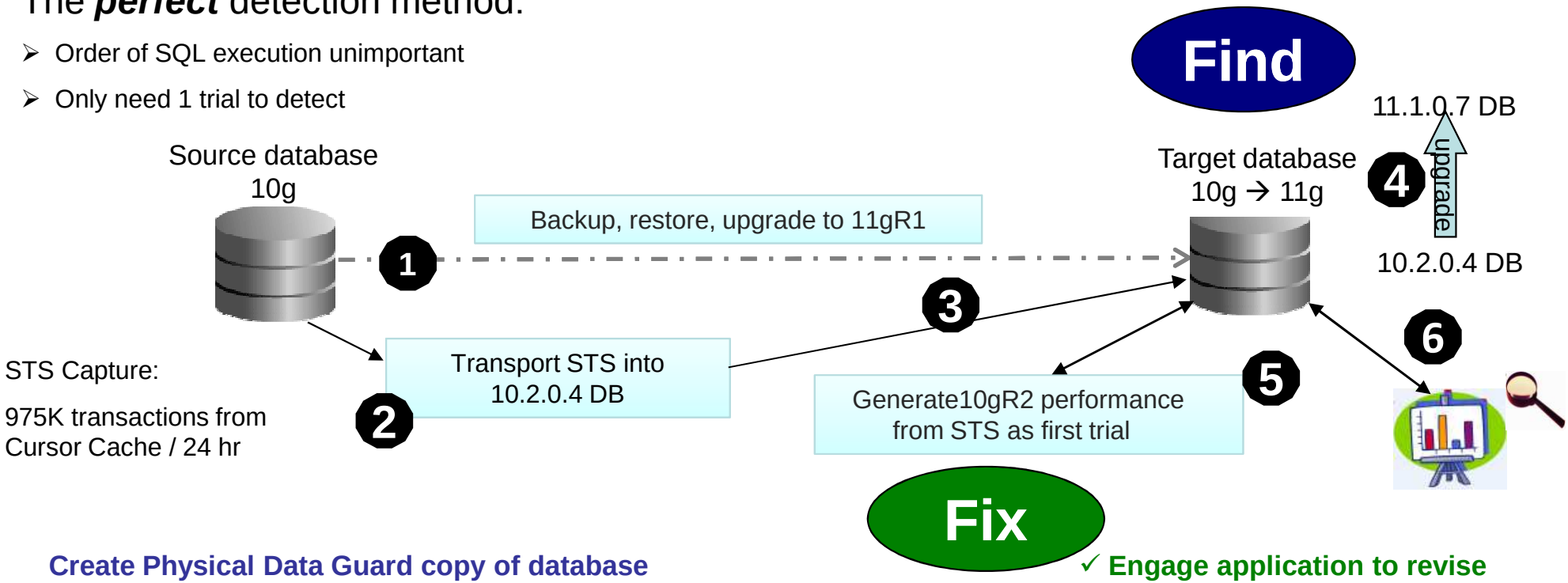
Challenges: Find it

- How to identify? It's not easy to detect. As easy as finding a n… well, you know…
- Can't be fixed by optimizer hint, alter system/session setting, hidden parameter, statistics, or SQL Profile.

Solution: *SQL Performance Analyzer*

The ***perfect*** detection method:

- Order of SQL execution unimportant
- Only need 1 trial to detect



1. **Create Physical Data Guard copy of database**
2. **Capture SQL from Production database using SQL Tuning Set**
3. **Transport STS into 10g target database (export/import)**
4. **Upgrade target database to 11g**
5. **Create first trial from STS to get SQL performance/errors**
6. **Review error report.**

✓ **Engage application to revise SQL to remediate.**

✓ **SQL can be verified in 11g test environment.**

Case 2: Full Outer Join Regression

…or Did you really need a FOJ?

Situation:

NextGen – 11g preparation using DB Replay

RAT replay encounters regressed FOJ
~ 4 hour workload captured
~ DB replay "stuck" on several FOJ
~ Projected run time: 49 days

```
********************************
* Awesome replay status report *
********************************
Replay has been running for:      +000000000 03:55:08.589684000
Current clock is: 6607927038954
Replay is waiting on clock: 6607927038961
1342 threads are currently being replayed.
Next ticker is process 29271 (16,19163)  in instance 1 and is waiting on
WCR: replay clock
Estimated progression in replay:  00% done.
Estimated time before completion: +000000049 02:36:15.886623000
Estimated total time for replay:  +49 06:31:24.943089
Estimated final time for replay:  15-MAY-11 17:23:12
```

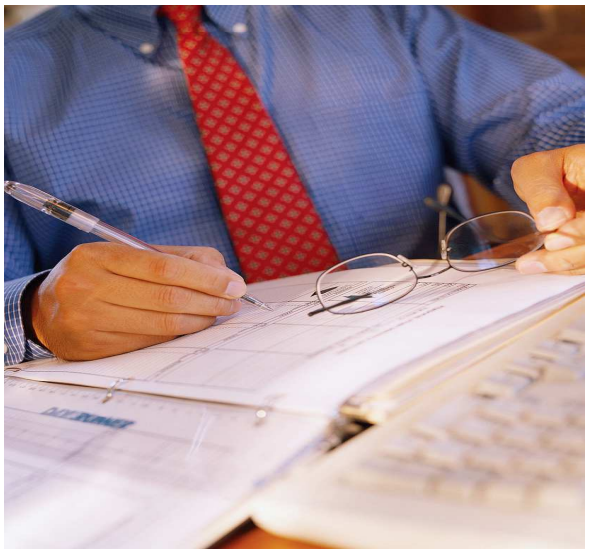
Solution: *SQL Profile / Optimizer Hint*

Detection using DB Replay

- Database setup using Physical DataGuard copy (similar to SPA)
- Workload Capture taken in production for 4-hr time period
- DB Replay run to "test out" 11g behavior

Find

AWR data used to contrast previous performance with current DB Replay performance for this SQL



snap_id	sql_id	plan_hash_value	execs	elapsed_time	ET/exec
80524	bbpknx8y8scu2	1312820390	187	54.920844	0.294
80725	bbpknx8y8scu2	694296214	1	164.238964	164.239

Fix

Several options to choose from:

- SQL Profile to force access to 10g behavior (plan outline)
- Hint: OPTIMIZER_FEATURES_ENABLE('10.2.0.4')
- Use hidden parameter _optimizer_native_full_outer_join=off
- Rewrite query using LEFT OUTER JOIN

Solution: *SQL Profile / Optimizer Hint*

Re-Run DB Replay

- Flashback to GRP
- Implement SQL Profile—quick fix. Long-term solution is to rewrite query using Left Outer Joins.
- Rerun DB Replay to verify "fixed" behavior

Validate

AWR data used to contrast previous performance with current DB Replay performance for this SQL

snap_id	sql_id	plan_hash_value	execs	elapsed_time	ET/exec
80725	bbpknx8y8scu2	694296214	1	164.238964	164.239
81115	bbpknx8y8scu2	2072138755	201	193.652570	0.963

Success

Implemented solution ...

- Allowed rest of DB Replay to complete
- Shows that access is once again sub-second
- Above all demonstrates the flexibility of DB Replay, allowing you to tweak your solution and try different alternatives

Questions

Any Questions?