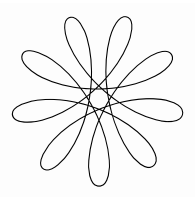
Automic™
Let's Automate Business.

WHITE PAPER

Oracle Database 10g/11g Tuning for Applications Manager v8.0

For more information about Automic products please visit www.automic.com.



Introduction

Attention: This white paper is intended for Oracle Administrators ONLY.

While reading this paper, keep in mind that it relates to your Oracle configuration, NOT your Applications Manager's configuration. This topic is not authoritative. If you need more information, please refer to an Oracle resource.

Overview

Regular maintenance of the Applications Manager's Oracle database is vital to its performance. Applications Manager's aw_inc_update database procedure may be one place that can cause long delays and affect the overall performance.

The best indicator that the aw_inc_update procedure may need some additional tuning is to submit the TEST_JOB with a 60 second prompt value. On systems where aw_inc_update is slow, it can take up to 30 seconds to see the TEST_JOB appear in the backlog.

Examining the Oracle Execution Plans

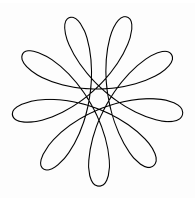
In tuning the procedure aw_inc_update, you need to examine the plans for the following queries:

```
SELECT    1
FROM AW_JOB_ACTIVITY
WHERE SO_AW_SEQ > MAX_SEQ AND SO_AW_SEQ < 999999999999;

SELECT *
FROM AW_JOB_ACTIVITY
WHERE SO_AW_SEQ > MAX_SEQ AND SO_AW_SEQ < 999999999999 AND SO_MODULE
!= 'ChAiN_sUbMiT'
ORDER BY TABLE_FLAG DESC, SO_AW_SEQ ASC;
```

Applications Manager v8.0 uses a view aw_job_activity, which queries views which in turn queries more views. The problem with these queries is that they were written for the rule based optimizer not cost based. Therefore, the aw_inc_update procedure is a good place for some additional tuning.

The explain plans on these queries change dramatically as additional job history information is collected. In a development environment, the cost could be 30. In a production environment under a full daily cycle, the cost can change to 6,000.



Below is a best case plan for the query above.

explain_plan_be...

Notice there are no hash joins, just nested loops. Every table is accessed via an index. This is the ideal route. This was generated from an Applications Manager v8.0 environment with very little history data.

As we add history data to the same environment, the query starts to get far more complicated. The data is skewed and Oracle starts to try to make allowances for the data. This is where it gets very complicated because we are querying views of views.

Adding 44,000 jobs to the history within the same environment and then re-running the schema the plan for the above query now looks like this.

explain_plan_wo...

The cost is now 6,900. Notice all of the full table scans and hash joins. This requires a lot of effort from oracle execute plan and we will start to see slowdowns in the backlog.

To combat this, we can tell oracle to prefer index scans over full tables by artificially reducing the perceived cost of an index scan.

This statement tells Oracle to reduce the perceived cost of an index by 80 percent.

```
alter session set optimizer_index_cost_adj = 20;
```

This statement tells Oracle that 70 percent of the indexes are cached in memory

```
alter session set optimizer_index_caching = 70;
```

This statement removes db_file_multiblock_read_count parameter. **In 10gR2 and above, this value is automatically calculated and should not be specified.**

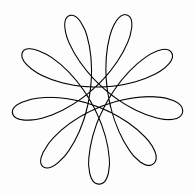
```
alter session set db_file_multiblock_read_count = 0;
```

It is recommended that you add these parameters via aw_master_stmts.

```
insert into aw_master_stmts (aw_sql, aw_order)
values ('alter session set optimizer_index_cost_adj = 20', 101);

insert into aw_master_stmts (aw_sql, aw_order)
values ('alter session set optimizer_index_caching = 70', 101);

insert into aw_master_stmts (aw_sql, aw_order)
values ('alter session set db_file_multiblock_read_count = 0', 101);
commit;
```



Summary

None of these optimizer parameters will fix the issue. It will, however reduce the cost slightly and speed up execution time.

Other customers have added SQL profiles or stored outlines to help with the issue. Unfortunately, Oracle's SQL Tuning advisor has to recommend a new profile/outline and in some cases it does not or the one it recommends is not helpful.

As stated above the problem with these queries is that they were written for the rule based optimizer and not the cost based optimizer. If you add the deprecated /*+ rule */ hint to the statement, the plan is perfect.