An Oracle White Paper
November 2015

Best Practices for Minimizing Oracle E-Business Suite Release 12.2.n Upgrade Downtime

Executive Overview 4
Introduction 4
Using this document 4
Preparation and Planning 5
Resolving Performance Issues 5
Overview of the R12.2.n Upgrade Process 6
Online Patching 6
Downtime Mode 7
Online Patching, Custom Code and Performance 8
Prediction of Performance Issues 9
Summary 10
Preparation and Planning - Pre-Upgrade Environment / Activities 10
Preparation and Planning – Release 12.2.n Upgrade Activities ... 10
Resolving Performance Issues - Using Diagnostics 21
Resolving Performance Issues – Common Solutions 25
Key Resources and Documents 29
Preparation and Planning - Pre-Upgrade Environment / Activities ... 33
OATM (Oracle Applications Tablespace Model) 33
Extent Size for Large Transaction Tables 34
Multi-Org Architecture (Upgrade from 11i Only) 34
Purge old data 34
Purge interface data 35
Gather CBO Statistics 35
Fixed Object and Dictionary Statistics 36
Drop MRC Schema (Upgrade from 11i Only) 36
Preparation and Planning – Release 12.2.n Upgrade Activities 36
Planning 37
Upgrade Path and Database Certification 38
Performance and Technology Fixes 38

Removing Unnecessary Workloads / Overheads 38
Upgrade and Initialization Parameters 42
Dynamic / Hyper Threading 50
Resource Manager 51
12.2 Middle Tier Sizing Guidelines 51
Using “downtime mode” for Online Patching 51
Gathering CBO Statistics 52
Resolving Performance Issues 61
Diagnostics to be gathered during the upgrade 61
Useful Scripts 82
Common Solutions 91

Executive Overview

This document offers a strategy for upgrading Oracle E-Business Suite to Release 12.2.n that minimizes downtime, whilst minimizing the implementation and resource required.

In particular it will help to:

- Minimize downtime during the final iteration.
- Minimize number of test iterations used to resolve performance issues.
- Identify issues early. Ideally on the first test iteration.
- Simplify the final iterations, so that they require the minimum of resource

Introduction

Using this document

The summary section contains the high level recommendations and should be reviewed fully.

The detailed ‘‘Preparation and Planning‘‘ and ‘‘Resolving Performance Issues‘‘ sections contain detailed explanations for the recommendations or detailed instructions. They may also contain important notes to be aware of. This section will typically be used for reference.

There are two parts to the suggested approach:

- Preparation and Planning

- Resolving Performance Issues

Both are these are discussed below.

Preparation and Planning

This includes:

- Recommended configuration or setup.
- Tasks that should be run as part of the Release 12.2.n upgrade process.
- Diagnostics that should be captured for all runs.
- Advice on monitoring general performance.
- Advice on determining upgrade parameters (AD Parallel workers, batch size, max_parallel_servers) for each implementation.

This will typically apply to all implementations.

Resolving Performance Issues

This will include advice on how to resolve specific performance issues that are encountered on each implementation.

Although there are some specific areas and jobs which sometimes have performance issues, there are often new undiscovered / unreported performance issues. It is not possible to predict where performance issues will occur for a given customer.

Overview of the R12.2.n Upgrade Process

In an upgrade to R12.2.n the main patches are applied in the following sequence:

- Main Upgrade Driver for R12.2.0 (merged with pre-installed Oracle E-Business Suite Consolidated Upgrade Patch for R12.2.0)
- Online Patching Enablement
- R12.AD.C.Delta.n and R12.TXK.C.Delta.n
- 12.2.n Release Update Pack (RUP)

Autopatch (adpatch) is used to apply all the patches up to and including "Online Patching Enablement".

Online Patching (ADOP) and the underlying Edition-Based Redefinition are only used after "Online Patching Enablement" for the "R12.AD.C.Delta.n", "R12.TXK.C.Delta.n", "12.2.n RUP" and subsequent patches.

Note that although the Online Patching method is used for the latter stages in the upgrade to R12.2.n, these are run during downtime (the application is not running).

Since R12.AD.C.Delta.5 the 12.2.n RUP patches can be applied using "downtime" mode (see below).

So the performance advice and challenges for an upgrade to R12.2.0 are largely the same as for an upgrade to R12.1.n. The same method is used to apply many of the patches (AutoPatch).

However, the Online Patching Enablement patch, Online Patching and the Release Update Packs for 12.2.n are quite different to the main R12.2.0 upgrade, so they may introduce new performance challenges, particularly with internal SQL and ADOP SQL (i.e. SQL run from or on AD_ZD objects).

It is important to stress that although Online Patching (ADOP) provides new logs to track performance, the key diagnostics are still the same ones used for previous Oracle E-Business upgrades (e.g. AWR reports, AD Job Timing Reports, Display Cursor Reports, and SQLT Extracts etc.)

Online Patching

The duration of the cutover phase is kept to a minimum in Online Patching. However, this inevitably results in some additional overhead. In particular:

- Multiple phases (e.g. prepare, apply, finalize, cutover, cleanup).

- Two editions of the file system are updated.
- Checking and validation
- Tables cannot be editioned in EBR (Edition-Based Redefinition). So they are populated/updated across editions. This is usually done using the AD_ZD_TABLE.apply utility to fire the same forward cross edition trigger for all rows, whereas previously (on AutoPatch) SQL set operations were used. This can add an additional performance overhead.
- It can also share resources with the running application during the prepare, apply, finalize and cleanup phases. Although this will not be the case during the R12.2.n upgrade.

So customers should expect online patching to take longer.

Usually the Apply, Cleanup (If FULL, not QUICK) and FS Clone phases take the longest.

The new diagnostics available for ADOP are logs. These give timestamps for particular steps in the upgrade, or list errors that have occurred. However, they do not identify the SQLs or underlying events that caused the performance issue, so it is essential to use AWR reports, AD Job Timing Reports, Display Cursor Reports and SQLT Extracts etc.

Downtime Mode

Note that since R12.AD.C.Delta.5 a new patching capability called downtime mode can be used.

When applying Oracle E-Business Suite patches in this mode, ADOP will first confirm that the application tier services are down, and will then proceed to apply the patch to the run edition of the Oracle E-Business Suite database and file system.

Downtime mode patching does not use an online patching cycle. The process of applying a patch in downtime mode completes more quickly than in online mode, but at the cost of increased system downtime (although this will not be the case during the R12.2.n upgrade).

Note: Release 12.2 patches are not normally tested in downtime mode. Downtime mode is only supported for production use where explicitly documented, or when directed by Oracle Support or Development (e.g. for 12.2.4 RUP or 12.2.5 RUP).

It is recommended that the “Oracle E-Business Suite 12.2.4 Release Update Pack (patch 17919161)” or “Oracle E-Business Suite 12.2.5 Release Update Pack (Patch 19676458)” is applied using "downtime" mode. And this will help reduce the elapsed time.

Online Patching, Custom Code and Performance

Be aware that any custom code may need updating in preparation for an upgrade to R 12.2.n, so that it is compliant with Online Patching standards.

Prior to upgrading to R12.2.n the following must be run on the pre-upgrade environment (e.g. 11i, 12.0 or 12.1) as well as before and after Online Patching Enablement.:

- Online Patching Readiness Report to identify issues in custom database objects that will be fixed automatically Vs needing manual intervention. (ADZDPSUM.sql)
- Manual Fix Readiness Report. (ADZDPMAN.sql)
- Online Patching Database Compliance Checker (ADZDDBCC.sql). Fix any violations of development standards listed in this report.
- Global Standards Compliance Checker script (gscc.pl). Address errors that are reported by this script.

These scripts/reports are designed so that they can also be executed on an environment that has not yet been online patching enabled (e.g. 11i, 12.0, 12.1, 12.2.0).

Note that the some of the SQL statements in the "Online Patching Database Compliance Checker (ADZDDBCC.sql)" have a dependency on EBS 12.2. If the report is run prior to the 12.2 upgrade any SQL failures can be ignored.

Important: All violations that are reported by these utilities must be fixed before enabling online patching.

Note : this will not only ensure that the custom code complies with Oracle E-Business Suite online-patching coding standards but also ensure that Online Patching Enablement and other parts of the upgrade will run more quickly (particularly upgrading Materialized Views).

For more information and the latest patch details (for 11i, 12.0, 12.1, or 12.2) see My Oracle Support document:

- Using the Online Patching Readiness Report in Oracle E-Business Suite Release 12. (Document 1531121.1)

Also see:

> Oracle E-Business Suite Developer's Guide Release 12.2. Sections 3. Preparing for Online Patching, 4 Preparation of Customizations in a Release 12.2 Upgrade and 5 Handling Customizations in an Online Patching-Enabled Environment

And My Oracle Support documents:

- Developing and Deploying Customizations in Oracle E-Business Suite Release 12.2 (Document 1577661.1)
- Oracle E-Business Suite Release 12.2: Technical Planning, Getting Started, and Go-Live Checklist (Document 1585857.1)
-

Also review any customizations to see if any can be removed because they are no longer needed or relevant. Improved functionality in R12.2 may remove the need for a particular customization.

Prediction of Performance Issues

It can be difficult to predict how long a Release 12.2.n upgrade will take (in terms of downtime). In particular, it is difficult to produce reliable or accurate estimates based on database size. The downtime is dependent on the modules and functionality used, and on the hardware specification.

The nature of the cost based optimizer and the mathematics behind contention (e.g. queuing theory) means that jobs that previously had no known performance issues can suddenly have performance issues for particular customers. For example:

- Very small changes in CBO statistics, database initialization parameters and other configuration could lead to a different execution plan being used.
- Queue (wait) times can start to increase rapidly (exponentially) once the load reaches a certain level.

The modules, data distributions, and functionality used, not only across different tables and entities, but chronologically too, can all mean that the time to run a particular upgrade job or SQL varies enormously; or a different execution plan is used. The execution plan that was efficient for one customer may not be for another customer. The parameters or configuration of the environment may cause contention when previously they didn't: in this context, small changes can have a large impact.

Similarly, a performance issue encountered on a job may not be the same as a previous issue on the same job.

On the other hand, some jobs may run long for all customers that use certain modules heavily.

Summary

This section contains a summary of the recommendations to minimize downtime. More details, explanations and instructions can be found in the detailed sections that follow.

Preparation and Planning - Pre-Upgrade Environment / Activities

This includes steps to prevent performance issues and reduce downtime.

These are all activities to be carried out on the pre-upgrade environment prior to Release 12.2.n upgrade.

List of recommended pre-upgrade activities

- If upgrading from 11i then convert to OATM (Oracle Applications Tablespace Model). If upgrading from 12.0 or 12.1 then migrate any existing objects (not in OATM) using the Tablespace Migration Utility.
- Ensure that the large transaction tables populated or modified by the release 12.2 upgrade have a large extent size (1-10MB). Ensure that the tablespaces (for large transaction tables created during release 12.2.0) have large default extent sizes (1-10MB). Small extents will seriously limit the AD Parallel batch size used. They will also increase the occurrence of high water mark waits.
- Convert to the new Multiple Organizations (Multi-Org) architecture (Upgrade from 11i only).
- Purge all old data that is no longer needed prior to the upgrade.
- Flush all the interfaces, such as Auto Invoice, Journal Entry Import, Order Import etc.
- Drop MRC Schema if it still exists (Upgrade from 11i only).
- Gather the schema statistics (with GATHER_AUTO option for all schemas) close to the start of the downtime. Use FND_STATS or Gather Statistics concurrent programs.
- Gather Fixed Object and Dictionary Statistics

Preparation and Planning – Release 12.2.n Upgrade Activities

Only run the Release 12.2.n upgrade with one Oracle RAC node enabled. See the detail section for more information.Planning

Always read the relevant upgrade guide.

- Oracle E-Business Suite Upgrade Guide Release 11i to 12.2

or

- Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2

It is important to identify the latest Consolidated Upgrade Patch (CUP) for R12.2.0, Online Patching Enablement, R12.AD.C.Delta.n, R12.TXK.C.Delta.n and 12.2.n Release Update Packs (RUP) to apply. These will have the latest improvements for timing and performance of the upgrade.

Note that the CUP for 12.2.0 (which is pre-installed and merged with the main 12.2.0 upgrade driver) will contain performance fixes for the main 12.2.0 driver and R12.AD.C.Delta will contain performance fixes for ADOP (Online Patching).

At time of last update (October 2015), these were:

- Oracle E-Business Suite Consolidated Upgrade Patch 6 (CUP6) for R12.2.0 (Patch 19796566:12.2.0)
- Online Patching Enablement (12.2 CA ONLINE ENABLEMENT PATCH 13543062)
- R12.AD.C.Delta.7 (Patch 20745242)
- R12.TXK.C.Delta.7 (Patch 20784380).
- Oracle E-Business Suite 12.2.5 Release Update Pack (Patch 19676458)
- Oracle E-Business Suite Online Help for 12.2.5 Release Update Pack (Patch 19676460)

Note that there are a large number of timing and performance fixes (for the upgrade) in CUP6 for 12.2.0, R12.AD.C.Delta.7 and 12.2.5 RUP.

See My Oracle Support document "Oracle E-Business Suite Release 12.2: Technical Planning, Getting Started, and Go-Live Checklist (1585857.1)" for an overview of the essential information needed to start planning for an upgrade to 12.2.n.

See the following My Oracle Support documents for information on the latest patches, documentation etc.

- Oracle E-Business Suite Release Notes, Release 12.2 (Document 1320300.1)
- Oracle E-Business Suite Release 12.2: Suite-Wide Rollup and AD/TXK Delta Information (Document 1583092.1)

- Applying the Latest AD and TXK Release Update Packs to Oracle E-Business Suite Release 12.2 (Document 1617461.1)

Note that it is very important to:

- Performance test the upgrade on a system that is identical or comparable to the production system (same Architecture, CPU, IO, Memory and data volumes/ages). It should be a recent clone of the production system.
- Start performance testing as early as possible. Do not start it late.
- Allow time for multiple iterations of performance testing (at least 6). So that performance issues can be resolved, utilization maximized and parameters tuned.
- Speed the test cycle up.

Upgrade Path and Database Certification

Check :

- Database Preparation Guidelines for an Oracle E-Business Suite Release 12.2 Upgrade (D Document 1349240.1)
- My Oracle Support Certification Tab

For database versions certified with EBS and the upgrade paths.

Performance and Technology Fixes

Before upgrading to 12.2 look at My Oracle Support document "Oracle E-Business Suite Release 12.2: Consolidated List of Patches and Technology Bug Fixes (Document 1594274.1)" for technology patches to apply before upgrading to 12.2. This also gives information on the latest "EBS Technology Codelevel Checker" Patch (17537119).

Ensure that any recommended performance patches are applied. Check the My Oracle Support document " R12.1 and 12.2 Oracle E-Business Suite Pre-install Patches Report [Video] (Document 1448102.1)".

Removing Unnecessary Workloads / Overheads

- Disable any custom triggers and business events

- Disable all DBMS scheduler (DBMS_SCHEDULER), DBMS Job (DBMS_JOB) and Autotask (DBMS_AUTO_TASK_ADMIN) activities during the upgrade
- Review and disable custom VPD policies as needed
- Disable auditing if enabled.
- Review and disable all debug or logging; do this at all levels (site, responsibility, user level etc.).
- If possible run in noarchivelog mode.
- Disable flashback DB.
- Remove TDE (Transparent Data Encryption) from high volume tables.
- Consider running AutoConfig in parallel on a multi-node system.
- Split any RDBMS Upgrade, Platform Upgrade, Conversion to OATM into a separate downtime period.
- Use TUMS (The Upgrade Manual Script) to avoid running tasks that are not relevant to the installation (Upgrade from 11i Only)
- Minimize Historical Data To Be Upgraded (Upgrade by Request) (Upgrade from 11i Only)
- Parallelize pre and post upgrade technical activities
- Define separate concurrent manager queue for post upgrade jobs. If using RAC then "Parallel Concurrent Processing (PCP)" could be used for post-upgrade concurrent jobs.

Note: currently it is NOT recommended that customers add nodes to their 12.2 Rapid Install upgrade file system until AFTER the upgrade to the latest 12.2.x RUP is complete, so using Distributed AD and Shared APPL_TOP is not relevant.

Note that using a Staged Application System (APPL_TOP) is not an option for upgrades to Release 12.2.n. There are multiple reasons.

Note that "Defer Upload of Patch Information" cannot normally be used, as subsequent patches may rely on information from previous patches, and the Release 12.2.n upgrade is a series of patches.

See ADOP/Autopatch and phtofile option in :

- *Oracle E-Business Suite Maintenance Guide Release 12.2*
- *Oracle E-Business Suite Patching Procedures (12.1, 12.0)*

- *Oracle Applications Maintenance Utilities - Release 11i*

Upgrade and Initialization Parameters

For 32 cores or fewer initially set:

- parallel_max_servers = 2 x number of CPU cores.
- AD Parallel workers – start with 1 x number of CPU cores. Possibly increase to 1.5 x number of CPU cores.
- job_queue_processes = number of CPU cores

For more than 32 cores, start with:

- parallel_max_servers = 1 x number of CPU cores.
- AD Parallel workers = between 0.5 and 1.0 x number of CPU cores.

Based on the performance diagnostics, these values may need to be decreased. It may also be possible to increase them. The level of contention and resource (CPU) usage (in AWR) will be the key to determining this.

Be careful if the hardware has dynamic or hyper-threading. In such cases the number of logical CPUs could be a multiple of the number of cores. If so ensure that the values above are calculated using the number of CPU cores and not the number of logical CPUs.

Only increase the number of AD Parallel workers if the level of contention on all long running AD Parallel jobs is low. Similarly, only increase parallel_max_servers if the level of contention on all parallel query/DML jobs is low. Typically, this means SQL Tuning (poor execution plans) and Contention issues should be resolved before increasing the AD Parallel workers or parallel_max_servers.

Set AD Parallel batch size to 10,000.

Note: Small extents (e.g. 128k = 16 blocks) will seriously limit the batch size. So batches of much less than 1000 are likely, regardless of what batch size is selected.

If possible SGA and PGA should be maximized. Check feedback from AWR Advisory Statistics, but be aware that:

- The advisory statistics are reported for the last snapshot interval only.

- Where there are many workers accessing the same objects at the same time (e.g. AD Parallel jobs), the SGA Target Advisory (and Buffer Pool Advisory) may underestimate the reduction in physical reads obtained from increasing the SGA.

Increasing PGA will improve timings if there I/O waits on temporary space ("direct path read temp" and "direct path write temp"). There may also be physical reads showing on Hash Join, Sort or Group By operations in Display Cursor and SQL Monitor Reports.

Increasing SGA may improve timings if there are medium to high (>10%) I/O waits on database objects (e.g. db file sequential read, db file scattered read, direct path read, direct path write etc.)

Some starting rules of thumb are:

log buffer = 30 to 100 Mb

shared pool = 1 to 4 GB

pga target = 3 to 20 GB

SGA/buffer cache = multi GB: be generous without causing excessive paging

If specified, remove db_file_multiblock_read_count. This is the recommended value for normal running of Oracle E-Business Suite.

Note that using HugePages on Linux can improve SGA allocation and hence performance.

optimizer_dynamic_sampling level should normally be 2 (which is the default if not set) or higher. A value of 4 is recommended during the Release 12.2.n Upgrade, but revert to 2 (or remove) after the upgrade.

Note that the values of the initialization parameters above (except db_file_multiblock_read_count) may be different from the values used for normal running. So be sure to revert after the Release 12.2.n upgrade has completed.

For other initialization parameters, refer to My Oracle Support document, "Database Initialization Parameters for Oracle E-Business Suite Release 12 (Document 396009.1)".

Dynamic/Hyper Threading

Be aware of Dynamic or Hyper Threading. Configuring dynamic/hyper-threading for maximum throughput per CPU cycle is recommended – i.e. 1 thread for each core. On Oracle SPARC T4-4 this can be done by setting dynamic threading to max-ipc.

In addition, disable any power management. Power management will either prevent threading for maximum throughput, or throttle the CPU capacity.

Resource Manager

If using a Resource Plan to specify how resources are distributed among the different resource consumer groups, please review this to ensure that all sessions running during the upgrade downtime window have access to the full CPU resources. *Note that this does not apply to Post Upgrade Concurrent jobs, which will run alongside the normal online workload.*

12.2 Middle Tier Sizing Guidelines

Managed instances JVM sizing should consider both memory and CPU domains.

On 64bit environments, allocating huge heap sizes is not recommended, but rather have more managed instances in the cluster to scale up to the target concurrency levels.

For Admin Server sizing, the default size of 512M is not enough for most installations, setting the XMS to at least 1 GB and the XMX to 2GB is recommended.

An initial guidance on sizing can be found in

- Oracle E-Business Suite Installation Guide: Using Rapid Install, Release 12.2
- My Oracle Support document "Managing Configuration of Oracle HTTP Server and Web Application Services in Oracle E-Business Suite Release 12.2 (Document 1905593.1)"

Using "downtime mode" for Online Patching

It is recommended that the "Oracle E-Business Suite 12.2.4 Release Update Pack (patch 17919161)" or "Oracle E-Business Suite 12.2.5 Release Update Pack (Patch 19676458)" is applied using "downtime" mode. And this will help reduce the elapsed time.

Resolve any SQL Tuning Issues on Long Running Jobs Early

SQLs with poor execution plans often exhibit disproportionate use of a particular resource (e.g. I/O, CPU or memory) or a high level of contention (e.g. on blocks in buffer cache or on I/O). This can be misleading if trying to resolve general performance issues or investigating the benefit of changes in architecture or configuration.

Sometimes these architectural or configuration changes can cause regressions on long running jobs with poor execution plans and high resource usage or contention. And these regressions outweigh the benefits elsewhere.

Once any SQL tuning issues on long running jobs are resolved, it is much easier to investigate and assess the benefits of changes in architecture, configuration and numbers of AD Parallel Workers/Parallel Max Servers.

Gathering CBO Statistics

Gathering Schema Statistics

In normal circumstances it should not be necessary to manually gather schema statistics during the Release 12.2.n upgrade. In many circumstances (if necessary) the script adsstats.sql automatically gathers them towards the end of the R12.2.0 upgrade (e.g. phase: last+63).

However, CBO statistics should be gathered for all Oracle E-Business Suite schemas with GATHER_AUTO option using FND_STATS (or gather statistics concurrent program) again, after the entire Release 12.2.n upgrade, but prior to the system being online and available to users.

If the adsstats.sql script is taking a significant amount of time, the upgrade time can be reduced by:

- Exporting schema statistics gathered during test runs (by adsstats.sql - at same point: phase: last+63).
- Importing these statistics instead of running adsstats.sql.

Only do this if:

- The estimated time saving will be significant, and it has been checked that parallel execution is being used effectively (with parallel_max_servers set to a suitable value, such as 2 x number of cores).
- The test environment has the same data volumes as the production environment (i.e. is a clone of production).

Use the recommended scripts in the detailed section.

If the adsstats.sql job is taking a long time during the R12.2.0 upgrade then gathering CBO statistics for specific long running tables (with more than 100 million rows) with a lower sample size (percentage) may help.

See the detailed section for more information.

After go live, watch out for statistics that adsstats.sql has unlocked and which may need to be locked again.

Note that adsstats.sql is often confused with adstats.sql, which disables automatic statistics gathering and gathers fixed object and dictionary statistics.

Fixed Object and Dictionary Statistics

Fixed Object Statistics should be gathered:

- After any associated platform or database upgrade that is part of the overall Oracle E-Business Suite upgrade.
- After any SGA/PGA parameters have changed.
- After Release 12.2.n upgrade, when there is representative activity on the system.

Dictionary Statistics should be gathered:

- After any associated platform or DB upgrade that is part of the overall Oracle E-Business Suite upgrade.
- After the Release 12.2.n upgrade.
- After move to OATM.

There are also changes to fixed and dictionary objects due to Online Patching Enablement (and the underlying Edition-Based Redefinition). As a result internal SQL in Online Patching Enablement, R12.2.n RUPs and other online patches can sometimes be long running. Gathering fixed object or dictionary statistics can help in these circumstances. Particularly on editioning objects.

Fixed object and dictionary statistics are not gathered automatically in the Release 12.2 upgrade by adstats.sql or any other method.

There may be additional specific circumstances during the upgrade where fixed object or dictionary statistics need to be gathered (such as before importing schema statistics or running SQLT or AWR reports when AWR has grown significantly).

If there are only a handful of internal SQLs with inefficient execution plans and only a few objects then statistics could be gathered for specific objects rather than gathering all dictionary or fixed object statistics.

See the detailed section for more information. Do not use adstats.sql to gather fixed object and dictionary statistics (unless as part of a database upgrade). Use the APIs directly, as instructed in detailed section below.

General Diagnostics to be gathered during the upgrade

The following diagnostics are still required for all patches/phases of the upgrade. This includes Online Patching Enablement, the online patching (ADOP) phases (e.g. prepare, apply, finalize, cutover, cleanup) of post 12.2 patches, especially the 12.2.n RUPs.

Automatic Workload Repository (AWR) should be enabled with a snapshot of 30 minutes (the default is 60 minutes). For short upgrades, a shorter snapshot may be more suitable.

The AWR retention period should be long enough to cover the duration of the upgrade run and a significant period afterwards (to gather diagnostics and analyze). The suggestion is N+7 days, where N is the estimated upgrade time, but a longer period will provide more time to gather subsequent diagnostics and statistics.

It is strongly advised that statistics_level is set to ALL (or _rowsource_execution_statistics = TRUE) for the duration of the Release 12.2.n upgrade test runs.

When analyzing Release 12.2.n Upgrade performance issues, the goal is to:

- Prevent wasted test iterations. Aim to provide solutions that solve the issue first time.
- Maximize the number of performance issues investigated on each iteration.
- Maximize the number of performance issues resolved.

To do this it is recommend that the steps outlined in the "Express Diagnosis of Oracle E-Business Suite Release 12 Upgrade Performance Issues (Document 1583752.1)" whitepaper are used. The key steps are:

Before the Oracle E-Business Suite Upgrade:

- Set the statistics_level to ALL or _rowsource_execution_statistics = TRUE (there are no actual statistics without this).

During the Oracle E-Business Suite Upgrade:

- Monitor top SQL in AWR or Cursor Cache (memory) (see useful scripts). This could be internal or application SQL. Enterprise Manager can also be used to identify expensive SQL as it occurs.
- Obtain Display Cursor Reports (ALL +ALLSTATS) for the long running SQLs
- Obtain SQL Monitor Reports for SQL that uses Parallel Query or DML
- Identify when a piece of SQL ran (see useful scripts)
- Match long running SQL with a job (see useful scripts)
- Report on CBO Statistics for all Oracle E-Business Suite tables (see useful scripts)

After the Oracle E-Business Suite Upgrade

- Obtain AD Job Timing Reports
- Identify long running upgrade jobs (See useful scripts)
- Obtain the file versions for long running jobs
- Obtain AWR Reports
- Run afxplain.sql to obtain detailed CBO statistics, database parameters, CBO Parameters and metadata; do this for individual pieces of SQL

For more detailed analysis, the following are required:

- AD utility and worker logs
- SQLT with XTRACT for long-running SQL

In addition, Oracle Support and Development may require:

- AWR Export
- AD Parallel tables export

See the Diagnostics section in "Resolving Performance Issues" below for more details.

Online Patching Enablement - Specific Diagnostics to be gathered during the upgrade

In addition the following diagnostics are available during the Online Patching Enablement patch.

- ADZDSHOWDDLS.sql

Online Patching - Specific Diagnostics to be gathered during the upgrade

In addition the following diagnostics are available for patches applied using online patching.

- ADOP log directories
- The Online Patching Log Analyzer Utility adopscanlog (delivered in R12.AD.C.Delta.n since R12.AD.C.Delta.4). This analyzes the adop log directories
- ADZDSHOWLOG.sql / adzdshowlog.out
- ADOP cycle status
- SQL to identify status of the ADOP phases
- SQL to identify the AD and TXK C Patch levels

For more guidance see:

- Oracle E-Business Suite Maintenance Guide Release 12.2
- Oracle E-Business Suite Upgrade Guide Release 11i to 12.2
- Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2

And My Oracle Support Document:

- 12.2 E-Business Suite - Collecting Online Patching and fs_clone Log Files (Document 1542162.1)

See the Diagnostics section in "Resolving Performance Issues" below for more details.

Resolving Performance Issues - Using Diagnostics

File Versions

To find out versions of files actually used in an upgrade then check the unified driver (or patch driver). Do not rely on the version of the file or object in the file system or database, as it could be a later version applied by a subsequent patch.

Statistics_Level = ALL (or _rowsource_execution_statistics = TRUE)

This is the simplest way to see actual execution statistics (including elapsed time, physical reads, buffer gets etc.) for each execution plan line (on SQLT and Display Cursor report). The alternative of SQL Trace and TKPROF requires editing standard code.

Using this strategy will typically speed up the resolution of issues significantly and may also allow the correct solution to be identified first time.

Alternatively, the same actual execution statistics can be collected by setting the initialization parameter _rowsource_execution_statistics=TRUE (with statistics_level = 'TYPICAL'). This gives a lower overhead than statistics_level=ALL.

Note that setting statistics_level to ALL while Automatic Workload Repository (AWR) is enabled could significantly increase the number of rows inserted into the WRH$_LATCH_CHILDREN table. So monitor the usage of the SYSAUX tablespace to ensure that it does not run out of space.

AWR Export

The AWR Export allows the AWR tables to be loaded into another environment. They can then be analyzed to discover where waits occur (job, SQL, object etc) and the patterns.

AWR Reports

Obtain AWR reports for the period that the upgrade is running. Also obtain AWR reports for the duration of long running jobs and for each individual snapshot.

From these the following can be identified:

- Long running SQL (SQL Statistics/SQL Ordered By)
- Contention and bottlenecks (Top 5 Timed Foreground Events/Foreground Wait Events)
- CPU utilization

If there are high levels of a particular wait(s), first check to see if it only occurs with particular long running SQL and jobs, before assuming it is a system wide configuration or resource issue.

The Active Session History report for particular SQL can also be useful in identifying the waits/events for each row source or object.

See My Oracle Support document "Performance Diagnosis with Automatic Workload Repository (Document 1674086.1)" for more information.

Note that fixed object and dictionary statistics should be gathered before running AWR reports, especially if statistics_level has been set to ALL or a high retention period or a short snapshot interval has been used.

AD Parallel tables

The AD_PARALLEL_UPDATES, AD_PARALLEL_UPDATE_UNITS tables can give information on the actual rows processed, the number of batches, progress over time, and long running batches (that might indicate locks/sleeps or data distribution issues).

AD_TASK_TIMING gives start and end times of jobs and workers, which can help identify all long running jobs, and match long running SQL and performance issues (on AWR) with specific jobs. It also helps identify low worker utilization (and potentially resource utilization) due to phasing waits.

AD Job Timing Reports

The job timing report (adtimrpt.sql) reports the top 100 time consuming jobs. It also reports the timing of each upgrade phase and failed, deferred, re-started and skipped jobs. The detailed report adtimdet.sql (adt.lst) reports on all jobs by phase and by elapsed time.

Note that the "Top 100 Time Consuming Jobs" section of the standard adtimrpt.sql report lists all workers for AD Parallel jobs separately. So the top 100 can be dominated by a handful of jobs.

An alternative is to use the SQL in the "Long Running Upgrade Jobs" section of "Useful Scripts".

AD Utility and Worker Logs

The AD utility and worker logs can also be useful for diagnostics, giving more detail about what happened and when. The AD workers logs (adworknnn.log) will give the activities carried out by each worker and the timings.

SQL Trace – Job Specific

If the diagnostics that are needed to resolve a performance issue could not be obtained from the display cursor, SQL monitor, and AWR, a 10046 SQL Trace (level 16 – ALL_EXECUTIONS (with waits)) should be obtained for the job on the next test run.

Although it is possible to SQL Trace other sessions using DBMS_MONITOR (10g and above) or the Event++ syntax (11g and above), it is advisable to enable SQL Trace for the specific job (by temporarily editing the SQL script). See detailed section for instructions.

SQL Specific

Once the long running SQL (in the long running jobs) has been identified, the following will provide diagnostics for individual pieces of SQL:

- Display Cursor Report. This can be done whilst the jobs are running or very shortly afterwards. However, the cursor must still be in the shared pool (memory), otherwise the actual statistics are lost.
- SQL Monitor Report (for SQLs that use Oracle Parallel Query/DML). This can be done whilst the jobs are running or very shortly afterwards.
- afxplain.sql (provides detailed CBO Statistics, DB/CBO Parameters and Metadata for a SQL)
- SQLT Output using the XTRACT method. This could have a performance impact so should be obtained after patches/phases have completed.

The first three can be provided during the upgrade run, and the last after it has completed.

Online Patching Enablement Specific

The Online Patching Enablement patch is applied using AutoPatch (adpatch). In addition to the general diagnostics above the output from the following script will be useful during Online Patching Enablement:

$ sqlplus apps @$AD_TOP/sql/ADZDSHOWDDLS.sql

Online Patching (ADOP) Logs and Diagnostics

All the ADOP logs are located on the non-editioned file system (fs_ne) in the <INSTALL BASE>/fs_ne/EBSapps/log/adop directory e.g.

/u01/PROD/fs_ne/EBSapps/log/adop

It is easiest and quickest to produce a zip of the entire directory.

The main files of interest are the ADOP logs (e.g. adop_YYYYMMDD_HHMISS.log).

But the adzdshowlog.out, adworker*.log, u*.log, u*.lgi, admrgpch*.log files are all useful and under the same path.

AD Utility and worker logs (Non ADOP) are also stored in the same location. e.g. adrelink.log, adlibin.log, adlibout.log, adworknnn.log.

Online Patching Log Analyzer Utility (adopscanlog)

This is delivered in R12.AD.C.Delta.n (since R12.AD.C.Delta.4).

This utility analyzes adop log directories for errors and warnings, and displays messages to help the user quickly identify any problems that may have occurred during an adop run. It thereby offers an alternative to reviewing log files manually.

ADZDSHOWLOG.sql / adzdshowlog.out

This reports the contents of the AD_ZD_LOGS table. This contains messages on the progress of online patching with timestamps.

ADOP Cycle Status

The current status of the adop cycle can be checked using *adop –status*.

Using SQL to check on status of ADOP and AD and TXK C Patch levels

SQL can also be directly used to check the status of ADOP phases or AD/TXK Patch Levels

Long Running SQL, Contention and Tuning

For long running jobs or SQL it is best to start by investigating if good execution plans are being used. A poor execution plan (or even just one that is moderately sub-optimal) can be the root cause of contention, especially if that contention only occurs during a particular job.

Once any unnecessary contention caused by sub-optimal execution plans has been removed, a small amount of contention (e.g. 5 to -15% on particular waits) between AD Parallel or Parallel Execution sessions can be a useful indicator that the most is being obtained from the available resources.

Be aware that although the entire wait time of some wait events can be classed as contention, this is not the case for all waits, and in particular not for I/O waits such as db file sequential read/db file scattered read and direct path reads/writes.

Resolving Performance Issues – Common Solutions

Known Issues

Once the long running jobs and SQL have been identified, check My Oracle Support for known issues and potential solutions or workarounds.

However, bear in mind that the fix or workaround may not necessarily fix the particular problem that has been observed.

If it cannot be confirmed (from the diagnostics) that the issue is exactly the same then the fix may still be applied, but continue to gather diagnostics and search for solutions until the issue is fully resolved.

Custom Indexes

Create custom indexes for long running jobs where a new index could significantly improve the execution plan and performance.

SQL Profiles for Inefficient Execution Plans

If it has been identified that a long running job has an inefficient execution plan, a SQL Profile could be used to apply hints that will help the CBO choose a better execution plan. SQL tuning expertise will be needed to do this.

Pragmatic Stable Execution Plans

A pragmatic stable execution plan may not be the very best execution plan possible, but it will not be inefficient and the job time will be predictable and stable.

In most cases, there is one join order that will give a good execution plan and minimize throwaway.

For AD Parallel jobs, a pragmatic execution plan will lead with the driving table, accessing using a range of rowids and then use nested loop joins and index access on subsequent tables. The tables which are likely to have the lowest number of rows joined to (either due to selective filters, or total number of rows on the table) should be joined to first.

Long running SQL in Online Patching Enablement / Online Patching

There can sometimes be long running internal SQL with inefficient execution plans in Online Patching Enablement, R12.2.n RUPs and other online patches.

Gathering fixed object or dictionary statistics can help in these circumstances. Particularly on editioning objects.

Long running Cleanup in Online Patching

If cleanup (after applying AD/TXK and 12.2.n RUP patches) is taking too long (particularly "drop covered objects") then consider running quick cleanup rather than standard cleanup.

Note that:

Cleanup and FS_CLONE stages are still required even if patches are being applied using ADOP in downtime mode. So this advice still applies.

If cleanup has not be run from the previous patching cycle the cleanup will run at the start of FS_CLONE. **Long Running Index Creation**

This will typically apply to xdf and odf jobs.

Obtain an AWR report for the snapshots where the index is being created, then investigate further.

Although pre-creation can help in a few circumstances, there is more likely to be another solution. If pre-creation is necessary then take care to do it correctly. See detailed section.

Long Running Index Re-creation

If the definition of a seeded E-Business Suite seeded index has been customized to change the type, columns, order (ASC or DESC), uniqueness, compression or partitioning, then it will be re-created during the upgrade to match the standard definition.

For large tables this is likely to consume significant additional time.

High level of contention on indexes on long running DML jobs

If a long running job runs heavy DML, has a high level of contention and that contention is on particular indexes then consider dropping the indexes before the job and recreating them afterwards (provided the index is not used in the meantime).

Ensure that indexes are recreated in parallel and with exactly the same definition. And remember to ALTER INDEX to revert the degree of parallel (NOPARALLEL).

High Level of "enq: HW – contention" or "enq: HV – contention"

If a long running job inserting into a table and indexes has a high level of waits "enq: HW – contention" or "enq: HV – contention", the following could be done:

- If the wait is occurring largely on particular indexes, drop the indexes before the job and recreate them afterwards (provided the index is not used in the meantime).
- Increase the extent size on the table and indexes.
- Pre-allocate extents for the table and indexes.

- Partition the table and any indexes to spread the load across multiple partitions. Note that the tables and indexes would still be partitioned after go live. So only do this if it will also give definite benefits after going live on the production environment.

High Level of redo log waits "log buffer space", "log file sync", "log_file_switch" etc.

If there are a high level of waits associated with redo log, especially "log buffer space" and "log file sync" then consider:

- changing the configuration of redo logs
- move to faster filer
- increase the size; increase the number or increase the log parallelism (hidden parameter _log_parallelism_max). Consider running with NOLOGGING, but this is not advised.

If getting high "log file switch (checkpoint incomplete)" waits then consider removing (resetting) the initialization parameter log_checkpoint_interval.

Long Running Statistics Gathering (adsstats.sql)

If the upgrade is taking a long time to gather CBO statistics (adsstats.sql), then consider the following strategies:

- Increasing parallel_max_servers and pga_aggregate_target. (adsstats.sql should run with few or no concurrent tasks).
- Using all nodes on an Oracle RAC system.
- Importing statistics gathered during test runs (covered in Preparation and Planning section).

Gathering or Deleting Statistics to resolve specific performance issues

Performance issues may indicate the need to gather or delete statistics for specific objects.

The specifics of doing so will depend on the exact circumstances, and the solutions may not be simple. See the detailed section.

Long-Running Compilation / Re-compilation

If scripts (e.g. adobjcmp.sql/adutlrcmp.sql) which utilizes the job queue (DB Scheduler) to compile/re-compile objects (e.g. UTL_RECOMP.RECOMP_PARALLEL) are taking a long time then check that

the value of initialization parameter job_queue_processes is high enough (a value equal to the number of CPU cores is normally recommended).

Long Running Materialized View xdf/odfs

If there are long running xdf or odf jobs creating materialized views (MV), consider cleaning up or truncating any large MV logs.

Note that this requires complete refresh of all MVs that are dependent on the MV log. Check that the number of rows in the MV log (mlog$) is zero to confirm that all dependent MVs have been refreshed.

Long Running Jobs that might not be needed

Check if any long running jobs are actually needed (e.g. for a module or localization that is not used or not licensed). Oracle Support may advise that a job can be skipped, or offer a fix or workaround.

High I/O waits on temporary space

If there I/O waits on temporary space ("direct path read temp" and "direct path write temp") and there is no scope to increase PGA further (or it has not resolved the issue) then consider moving the temp space to local disk (or faster disk).

Maximum AD Parallel Workers

The Auto Patch utility specifies a maximum number of workers. The number depends on several factors, and under certain circumstances may be less than the number of workers required.

See the detailed section for how this maximum is calculated, and how this issue might be resolved (increase processes parameter, or reduce other sessions/processes).

Key Resources and Documents

Documentation:

- Oracle E-Business Suite Upgrade Guide Release 11i to 12.2. Especially Planning for an Upgrade (Chapter 1) and appendices Reducing Downtime (F) and Upgrade By request (H)
- Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2. Especially Planning for an Upgrade (Chapter 1) and Preparing for the Upgrade (Chapter 2)
- Oracle E-Business Suite Setup Guide, Release 12.2

- Oracle E-Business Suite Maintenance Guide Release 12.2
- Oracle E-Business Suite Developer's Guide, Release 12.2. Especially Sections 3. Preparing for Online Patching, 4. Preparation of Customizations in a Release 12.2 Upgrade and 5. Handling Customizations in an Online Patching-Enabled Environment
- Oracle E-Business Suite Installation Guide: Using Rapid Install, Release 12.2
- Oracle E-Business Suite Concepts, Release 12.2
- Oracle E-Business Suite Patching Procedures, Release 12.1 and Release 12.0
- Oracle Applications Maintenance Utilities, Release 11i
- Oracle E-Business Suite Multiple Organizations Implementation Guide, Release 12.1
- Oracle E-Business Suite System Administrator's Guide – Configuration, Release 12.1
- Oracle Applications System Administrator's Guide - Configuration, Release 11i
- Oracle Database VLDB and Partitioning Guide - "How Parallel Execution Works" and "Tuning General Parameters for Parallel Execution" sections
- Oracle Database Performance Tuning Guide – "Transporting Automatic Workload Repository Data" section

My Oracle Support documents:

- Oracle E-Business Suite Release Notes, Release 12.2 (Document 1320300.1)
- Oracle E-Business Suite Release 12.2: Suite-Wide Rollup and AD/TXK Delta Information (Document 1583092.1)
- Applying the Latest AD and TXK Release Update Packs to Oracle E-Business Suite Release 12.2 (Document 1617461.1)
- Oracle E-Business Suite Release 12.2: Technical Planning, Getting Started, and Go-Live Checklist (Document 1585857.1)
- Database Preparation Guidelines for an E-Business Suite Release 12.2 Upgrade (Document 1349240.1)
- Oracle E-Business Suite Release 12.2: Consolidated List of Patches and Technology Bug Fixes (Document 1594274.1)
- Best Practices for Gathering Statistics with Oracle E-Business Suite (Document 1586374.1)

- Fixed Objects Statistics (GATHER_FIXED_OBJECTS_STATS) Considerations (Document 798257.1)
- AD Command Line Options for Release R12 (Document 1078973.1)
- Oracle E-Business Suite Release 12.2: Upgrade Sizing and Best Practices (Document 1597531.1)
- R12.1 and 12.2 Oracle E-Business Suite Pre-install Patches Report [Video] (Document 1448102.1)
- R11i / R12: Oracle E-Business Suite Upgrades and Platform Migration (Document 1377213.1)
- Use of Multiple Organizations In Oracle Applications Release 11i (Document 210193.1)
- Reducing Your Oracle E-Business Suite Data Footprint using Archiving, Purging, and Information Lifecycle Management (Document 752322.1)
- How to Send a File to Oracle Support Using FTPS (Document 464666.1)
- Express Diagnosis of Oracle E-Business Suite Release 12 Upgrade Performance Issues (Document 1583752.1)
- Oracle E-Business Suite Performance Guide (Document 1672174.1)
- Oracle E-Business Suite SQL Trace and TKPROF Guide (Document 1674024.1)
- Performance Diagnosis with Automatic Workload Repository (Document 1674086.1)
- All About the SQLT Diagnostic Tool (Document 215187.1)
- How Does Adpatch Determine The Number Of Workers To Recommend? (Document 800024.1)
- Using AutoConfig to Manage System Configurations in Oracle E-Business Suite Release 12 (Document 387859.1) – section "5.1. Running AutoConfig in Parallel Across Multiple Nodes".
- How to verify or create a Database Object using an odf (adodfcmp) or xdf (FndXdfCmp) file? (Document 551325.1)
- Database Initialization Parameters for Oracle E-Business Suite Release 12 (Document 396009.1)
- R12 FAQ for the SLA Upgrade: SLA Pre-Upgrade, Post-Upgrade, and Hot Patch (Document 604893.1)
- Managing Configuration of Oracle HTTP Server and Web Application Services in Oracle E-Business Suite Release 12.2 (Document 1905593.1)

- Developing and Deploying Customizations in Oracle E-Business Suite Release 12.2 (Document 1577661.1)
- Using the Online Patching Readiness Report in Oracle E-Business Suite Release 12. (Document 1531121.1)
- HugePages on Oracle Linux 64-bit (Document 361468.1)
- Shell Script to Calculate Values Recommended Linux HugePages / HugeTLB Configuration (Document 401749.1)
- 12.2 E-Business Suite - Collecting Online Patching and fs_clone Log Files (Document 1542162.1)
- EBS - Technology Area - Webcast Recording 'E-Business Suite - RAC & Parallel Concurrent Processing (PCP)' [video] (Doc ID 1359612.1)

Preparation and Planning - Pre-Upgrade Environment / Activities

This includes steps to prevent performance issues and reduce downtime.

These are all activities to be carried out on the pre-upgrade environment prior to Release 12.2.n upgrade.

OATM (Oracle Applications Tablespace Model)

Migrating existing objects to OATM on the pre-upgrade environment is recommended.

If upgrading from 11i then convert to OATM (Oracle Applications Tablespace Model), whilst still on 11i.

If the environment was previously upgraded from Release 11i to Release 12.0 or 12.1, then the upgrade process created tablespaces for all new products, configured the database for the new tablespace model, and created new objects. However, it did not automatically migrate existing objects. So, migrate any existing objects (not in OATM) using the Tablespace Migration Utility whilst still on 12.0 or 12.1.

If not already done, Oracle strongly recommends that the Tablespace Migration Utility is used to perform this migration now. Note that this utility is not supported for use after Online Patching has been enabled, so the migration cannot be performed after the environment is upgraded to Release 12.2.n.

If the environment is not migrated to OATM now, then tablespaces must continue to be managed separately.

See the following for more information:

- Chapter 1 (Planning for an Upgrade) of the "Oracle E-Business Suite Upgrade Guide Release 11i to 12.2".
- Chapter 2 (Preparing for the Upgrade) of the "Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2"
- Oracle E-Business Suite Setup Guide, Release 12.2
- Oracle E-Business Suite System Administrator's Guide – Configuration, Release 12.1
- Oracle Applications System Administrator's Guide - Configuration, Release 11i

Extent Size for Large Transaction Tables

Ensure that the large transaction tables populated or modified by the release 12.2.0 upgrade have a large extent size (1-10MB).

Ensure that the tablespaces (for large transaction tables created during release 12.2.0) have large default extent sizes (1-10MB).

For production environments and large tablespaces like transaction tables, transaction indexes, interfaces, summaries, archives, and media, a uniform extent size of 1MB or 10MB (with caution) should be considered. For large multi-terabyte system, an extent size of 4-10 MB has been tested successfully.

Small extents will seriously limit the AD Parallel batch size used (e.g. an extent size of 128k = 16 blocks is highly likely to give a batch size of less than 1000) and increase the occurrence of high water mark waits ("enq: HW – contention" or "enq: HV – contention").

See the following for more information:

- Oracle E-Business Suite Upgrade Guide Release 11i to 12.2 / Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2
- Oracle E-Business Suite Setup Guide, Release 12.2

Multi-Org Architecture (Upgrade from 11i Only)

If migrating from 11i then it is possible to convert to the new Multiple Organizations (Multi-Org) architecture while still on Oracle E-Business Suite 11i. See Chapter 1 (Planning for an Upgrade) of the "Oracle E-Business Suite Upgrade Guide Release 11i to 12.2" for more information.

Also see:

- Oracle E-Business Suite Multiple Organizations Implementation Guide, Release 12.1
- My Oracle Support document "Use of Multiple Organizations In Oracle Applications Release 11i (Document 210193.1)"

Purge old data

Purge all old data that is no longer needed prior to the upgrade.

This can be quite a lengthy process, and issues around performance, execution frequency and the periods to be purged in each execution may need to be resolved. So this should be started as soon as possible.

There are over 130 standard purge programs in 11i and 240 in R12.

Use OAM Purge Portal (in 11i and R12) to administrate, configure, initiate and monitor purge programs.

To access the purge portal, use the following navigation:

System Administrator > Oracle Applications Manager >Purging/Critical Activities

A list of purge programs is listed in My Oracle Support document "Reducing Your Oracle E-Business Suite Data Footprint using Archiving, Purging, and Information Lifecycle Management (Document 752322.1)" This contains Archive_Purge_ILM_paper[1].pdf

Purge interface data

Flush all the interfaces, such as Auto invoice, Journal entry import, and order import.

Gather CBO Statistics

Regularly gather schema statistics with GATHER_AUTO option for all schemas while still on Release 11i, 12.0 or 12.1.

Then gather schema statistics (with GATHER_AUTO option for all schemas) close to the start of the Release 12.2.n upgrade.

Always use the FND_STATS package or the Gather Statistics concurrent programs. Do not use the DBMS_STATS package or Analyze.

Zero or Incorrect Statistics on transient/temporary tables

Be aware that the statistics from the pre-upgrade environment may contain some issues, even if they have been gathered with the Auto option.

Check the statistics for transient, temporary or Global Temporary (GT) tables. These are tables which can alternately contain rows for processing (e.g. import/interface/staging), or contain no (or very few) rows.

For GT tables, rows are only visible to the session that inserted them, and those rows only persist for the duration of the session. If statistics on these tables are gathered then only statistics for the rows present (in that table) in that session will be collected: that is likely to result in statistics for 0 rows.

In addition, the statistics for GT tables are not session specific, so there is only one set of statistics and these are visible to all other sessions. Even if the statistics are gathered by a session when the tables are populated, they could still result in suboptimal execution plans if the volume or distribution of data is different for other sessions.

Similarly, for other transient, temporary or staging tables, statistics may have been gathered when the table is empty.

Having zero statistics (or statistics with low row count) could result in poor execution plans for temporary/transient tables that contain a large number of rows.

Ideally, all these temporary, transient or global temporary tables should have no statistics, and dynamic sampling should be used.

However, in specific cases, a strategy of populating with indicative statistics (when the table is populated) or gathering statistics each time a process is run may have been taken. In some cases the seeded Oracle E-Business Suite code will gather statistics.

Fixed Object and Dictionary Statistics

These should have been gathered and be correct/up to date on the pre-upgrade environment.

Drop MRC Schema (Upgrade from 11i Only)

All programs and reports now use the APPS schema. The MRC_APPS schema is no longer needed, so dropping it frees space and reduces processing overhead during the upgrade.

Drop this schema prior to the Release 12.2.n upgrade if it still exists. It should have already been removed on upgrade to 11.5.10.

See "Oracle E-Business Suite Upgrade Guide Release 11i to 12.2".

Preparation and Planning – Release 12.2.n Upgrade Activities

Only run the Release 12.2.n upgrade with one Oracle RAC node enabled. The majority of the elapsed time of the Release 12.2.n upgrade is taken by jobs running DML (INSERT, UPDATE, DELETE).

These jobs use multiple workers/parallel severs, which all access the same objects and blocks at the same time. So the additional communication between nodes on the cluster (and cluster waits) will significantly outweigh the gains from using the additional CPUs to increase throughput. In some cases it can lead to severe contention and deadlocks.

Planning

Always read the relevant upgrade guide.

- Oracle E-Business Suite Upgrade Guide Release 11i to 12.2

or

- Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2

It is important to always apply the latest CUP for 12.2.0, R12.AD.C.Delta, R12.TXK.C.Delta and 12.2.n RUP, as these will have the latest improvements for timing and performance of the upgrade.

Note that the CUP for 12.2.0 (which is pre-installed and merged with the main 12.2.0 upgrade driver) will contain performance fixes for the main 12.2.0 driver and R12.AD.C.Delta will contain performance fixes for ADOP (Online Patching).

Identify the latest Consolidated Upgrade Patch (CUP) for R12.2.0, Online Patching Enablement, R12.AD.C.Delta.n, R12.TXK.C.Delta.n and 12.2.n Release Update Packs (RUP) to apply.

At time of last update (October 2015), these were:

- Oracle E-Business Suite Consolidated Upgrade Patch 6 (CUP6) for R12.2.0 (Patch 19796566:12.2.0)
- Online Patching Enablement (12.2 CA ONLINE ENABLEMENT PATCH 13543062)
- R12.AD.C.Delta.7 (Patch 20745242)
- R12.TXK.C.Delta.7 (Patch 20784380).
- Oracle E-Business Suite 12.2.5 Release Update Pack (Patch 19676458)
- Oracle E-Business Suite Online Help for 12.2.5 Release Update Pack (Patch 19676460)

Note that there are a large number of timing and performance fixes (for the upgrade) in CUP6 for 12.2.0, R12.AD.C.Delta.7 and 12.2.5 RUP.

See My Oracle Support document "Oracle E-Business Suite Release 12.2: Technical Planning, Getting Started, and Go-Live Checklist (1585857.1)" for an overview of the essential information needed to start planning for an upgrade to 12.2.n.

See the following My Oracle Support documents for information on the latest patches, documentation etc.

- Oracle E-Business Suite Release Notes, Release 12.2 (Document 1320300.1)
- Oracle E-Business Suite Release 12.2: Suite-Wide Rollup and AD/TXK Delta Information (Document 1583092.1)
- Applying the Latest AD and TXK Release Update Packs to Oracle E-Business Suite Release 12.2 (Document 1617461.1)

Upgrade Path and Database Certification

Check :

- Database Preparation Guidelines for an Oracle E-Business Suite Release 12.2 Upgrade (Document 1349240.1)
- My Oracle Support Certification Tab

For database versions certified with EBS and the upgrade paths.

Performance and Technology Fixes

Before upgrading to 12.2 look at My Oracle Support document "Oracle E-Business Suite Release 12.2: Consolidated List of Patches and Technology Bug Fixes (Document 1594274.1)" for technology patches to apply before upgrading to 12.2. This also gives information on the latest "EBS Technology Codelevel Checker" Patch (17537119).

Ensure that any recommended performance patches are applied. Check the My Oracle Support document "R12.1 and 12.2 Oracle E-Business Suite Pre-install Patches Report [Video] (Document 1448102.1)".

Removing Unnecessary Workloads / Overheads

To reduce the workload do the following:

- Disable any custom triggers and business events.

- Disable all DBMS scheduler (DBMS_SCHEDULER), DBMS Job (DBMS_JOB) and Autotask (DBMS_AUTO_TASK_ADMIN) activities during the upgrade
- Review and disable custom VPD policies as needed.
- Disable auditing if enabled. The "Oracle E-Business Suite Upgrade Guide, Release 11i to 12.2" and "Oracle E-Business Suite Upgrade Guide, Release 12.0 and 12.1 to 12.2" state that the Oracle AOL Audit Trail should be disabled before upgrade, anyway.
- Review and disable all debug and logging that has been enabled using profiles. Do this at all levels (e.g. site, responsibility, user level.).
- If possible, run in noarchivelog mode.
- Disable flashback DB.
- Remove TDE (Transparent Data Encryption) from high volume tables.
- Consider running AutoConfig in parallel on a multi-node system. See
    - My Oracle Support document "Using AutoConfig to Manage System Configurations in Oracle E-Business Suite Release 12 (Document 387859.1)" – section "5.1. Running AutoConfig in Parallel Across Multiple Nodes".
    - Oracle E-Business Suite Setup Guide Release 12.2 - Chapter 3 Technical Configuration > Advanced AutoConfig Features and Additional Utilities > Running AutoConfig in Parallel Across Multiple Nodes
    - Oracle E-Business Suite Concepts Release 12.2 - Chapter 5 Technical Configuration Tools > Management Tasks > Running AutoConfig in Parallel

Note: currently it is NOT recommended that customers add nodes to their 12.2 Rapid Install upgrade file system until AFTER the upgrade to the latest 12.2.x RUP is complete, so using Distributed AD and Shared APPL_TOP is not relevant.

Note that using a Staged Application System (APPL_TOP) is not an option for upgrades to Release 12.2.n. There are multiple reasons.

Note that "Defer Upload of Patch Information" (See My Oracle Support document "Patching Best Practices And Reducing Downtime (Document 225165.1)") cannot normally be used, as subsequent patches may rely on information from previous patches, and the Release 12.2.n upgrade is a series of patches (e.g. Online Patching Enablement, R12.AD.C.Delta.n, R12.TXK.C.Delta.n and 12.2.n RUP patches all follow R12.2.0).

Split Into Separate Steps

Identify tasks that could be completed in a separate downtime period, prior to the production upgrade.

The following could all be candidates:

- Upgrade Database version (to latest certified for the current Oracle E-Business Suite level)
- Convert to Oracle Applications Tablespace Model (OATM).
- Other planned platform (hardware or OS) upgrades.
- For Upgrade from 11i, the tasks identified in the "Oracle E-Business Suite Upgrade Guide Release 11i to 12.2" Planning for an Upgrade (Chapter 1) and appendices Reducing Downtime (F) and Upgrade By request (H). The latter includes the topic covered in the section below and will include pre-upgrade and post-upgrade tasks.

Use TUMS (The Upgrade Manual Script) to avoid running tasks not relevant to the installation (Upgrade from 11i Only)

The TUMS report lists tasks that can be omitted from the upgrade because they do not apply to the installation (for example, a task required for a product that is not used or a patch that has previously been applied). TUMS is delivered in a patch (18342870), which supplies the scripts needed to examine the system and create the report. It is strongly recommend the TUMS report is created and reviewed before beginning the upgrade.

Download and apply patch 18342870, then run the script adtums.sql to generate the report tumsr12.html.

The tumsr12.htm report lists the steps (identified by the TUMS step key in this book) that do not apply to the installation. Any steps listed in this report may safely be ignored.

See "Oracle E-Business Suite Upgrade Guide, Release 11i to 12.2" Preparing for the Upgrade (Chapter 2) for more information.

Minimize Historical Data To Be Upgraded (Upgrade from 11i Only)

See "Oracle E-Business Suite Upgrade Guide Release 11i to 12.2" appendix Upgrade By request (H).

Also see My Oracle Support documents:

- R12 FAQ for the SLA Upgrade: SLA Pre-Upgrade, Post-Upgrade, and Hot Patch (Document 604893.1)

- Oracle E-Business Suite Release 12.2: Upgrade Sizing and Best Practices (Document 1597531.1)

Note that there are some SLA upgrade jobs in the upgrade from 11i to Release 12.2.0, especially for Oracle Payables, which still process all the historical data, even if only the most current data is being upgraded.

The Release 12.2.0 upgrade migrates all the transactions for the following regardless - Payables (Transaction Entities, Events, Journal Headers, Journal Lines), Receivables (Transaction Entities).

Parallelize pre and post upgrade technical activities

Where practical, perform technical upgrade tasks in parallel. For example, while the upgrade driver is running on the database, Release 12.2 setups or re-register single sign-on could be performed on the application tiers.

Define separate concurrent manager queue for post-upgrade job

There are many concurrent programs automatically submitted in the downtime portion of the upgrade to Release 12.2.n. Many of these programs will run in multiple threads, so the total number of concurrent requests that form part of the post-upgrade step is much higher.

These programs will be picked up and executed by the concurrent manager once the system is up. Consequently, their execution will be mixed with the execution of ongoing concurrent jobs in the system.

It can be a good idea to define a separate concurrent manager queue for these requests, to enable post-upgrade testing of standard run-time requests to proceed without interference from upgrade-related processes.

Such a definition can be achieved by using inclusion and exclusion rules to prevent other manager queues such as standard from picking up these requests, and allow this new manager queue to process these requests only.

Doing this now allows the number of target processes allocated to these post-upgrade concurrent programs to be controlled, including doing this dynamically with the use of work shifts. For additional details on configuring new manager queues, target processes, inclusion/exclusion rules and work shifts, refer to the Applications System Administrator's Guide from Oracle E-Business Documentation Library.

See the following for more information:

- My Oracle Support document "Oracle E-Business Suite Release 12.2: Upgrade Sizing and Best Practices (Document 1597531.1)"
- "Oracle E-Business Suite Upgrade Guide, Release 12.0 and 12.1 to 12.2" - E Managing Concurrent Processes
- "Oracle E-Business Suite Upgrade Guide, Release 11i to 12.2" - K Managing Concurrent Processes

If using RAC then "Parallel Concurrent Processing (PCP)" could be used for post-upgrade concurrent jobs to assign programs to different nodes, but care should be taken over node affinity (there is little point having two different requests accessing the same blocks/rows from different nodes at the same time).

Note that since Oracle E-Business Suite 12.1, Concurrent requests can be directed to a specific database instance or node on a per-program basis.

For more information see MOS document "EBS - Technology Area - Webcast Recording 'E-Business Suite - RAC & Parallel Concurrent Processing (PCP)' [video] (Doc ID 1359612.1)".

Upgrade and Initialization Parameters

Note that the values of the initialization parameters below (except db_file_multiblock_read_count) may be different from the values used for normal running. So be sure to revert after the Release 12.2.n upgrade is complete.

For other initialization parameters, refer to My Oracle Support document "Database Initialization Parameters for Oracle E-Business Suite Release 12 (Document 396009.1)".

AD Parallel Workers and parallel_max_servers

The number of parallel execution threads is determined by the initialization parameter parallel_max_servers.

This will impact the elapsed time of jobs that use parallel SQL and DML. Such jobs include apintbal.sql, apxlaupg.sql, and adsstats.sql.

The number of workers used in the AD Parallel Utility (in Autopatch and ADOP) is also configurable. It is important to get the number of AD Parallel workers and the parallel_max_servers right.

The typical recommendation is to set:

- parallel_max_servers = 2 x number of CPU cores.
- AD Parallel workers – start with 1 x number of CPU cores. Possibly increase to 1.5 x number of CPU cores.

However, note that lower values may turn out to be more optimal, for the following reasons:

- The above values are recommended to maximize CPU utilization. However, the constraints may lie in I/O, memory, or elsewhere. So check memory utilization (no swapping/excessive paging) and I/O response times (histogram shows a fat tail for the longer wait times). Also check for buffer busy waits spread across all objects (high levels on specific objects can often indicate a sub-optimal execution plan).
- The number of CPUs/cores on systems is increasing, and consequently the number of CPU cores presented to the operating system is also increasing. This is particularly true if the hardware has dynamic/hyper-threading capabilities and they are not disabled for the duration of the Release 12.2.n upgrade.

The recommendations above are usually valid if there are less than 32 CPU cores. However, beyond 32, the levels of contention can increase significantly, outweighing any gains in CPU and resource utilization obtained from increasing the workers/threads.

So, if the number of CPU cores is 32 or more, then start with parallel_max_servers and AD Parallel workers below the above levels. For example:

- parallel_max_servers = 1 x number of CPU cores.
- AD Parallel workers to between 0.5 and 1.0 x number of CPU cores.

Be careful if the hardware has dynamic or hyper-threading. In such cases, the number of logical CPUs could be a multiple of the number of cores. If so, calculate the values above using the number of CPU cores and not the number of logical CPUs.

If hyper-threading or dynamic threading is enabled and CPU utilization is low, the hyper-threading configuration may need to be changed. See the Dynamic / Hyper threading section below.

Only increase the number of AD Parallel workers if the level of contention on all long running AD Parallel jobs is low. Similarly, only increase parallel_max_servers if the level of contention on all long running parallel query/DML jobs is low. Typically, this means SQL Tuning (poor execution plans) and contention issues should be resolved *before* increasing the AD Parallel workers or parallel_max_servers.

Note that parallel_servers_target should have no value (default). In the upgrade process there is normally only one parallel DML/SQL job running at any one time, so it should never have a value less than parallel_max_servers.

Job_queue_processes

Similarly, job_queue_processes = number of CPU cores is normally recommended.

If there are a large number of CPU cores (or dynamic/hyper-threading) then a reduced value of job_queue_processes may be better.

However, job_queue_processes is only used extensively on scripts adobjcmp.sql/adutlrcmp.sql, which utilizes the job queue (DB Scheduler) to compile objects (e.g. UTL_RECOMP.RECOMP_PARALLEL).

SGA, Shared Pool, PGA etc.

If possible, SGA and PGA should be maximized.

Check the feedback from AWR Advisory Statistics and the SQLT / Display Cursor, which (if statistics_level is set to ALL or _rowsource_execution_statistics=TRUE), should show the temporary space used by sorts, hash joins, etc.

Note that the AWR Advisory Statistics may underestimate the SGA required.

- The advisory statistics are all reported for the last snapshot interval only. If the AWR report covers more than one snapshot interval and the last snapshot interval has a lower workload (e.g. the process being observed completes a long time before the end snapshot) then the advisory statistics could underestimate the advised values.
- Where there are many workers accessing the same objects at the same time (e.g. AD Parallel jobs), the SGA Target Advisory (and Buffer Pool Advisory) may underestimate the reduction in physical reads obtained from increasing the SGA.

  For the Release 12 Upgrades, an improvement in performance (due to reduction in physical I/O such as db file sequential read) has been noticed when SGA size is increased, even though the SGA Target Advisory on AWR indicates that the SGA is adequately sized. Customers have used values that are 2x the recommended value, and still seen significant benefit.

  The reason why the SGA Target Advisory does not recommend a larger value is not known, but it is suspected that it is due to the nature of the Release 12 Upgrade process (almost all rows on each table are repeatedly read; however, each AD Parallel or Parallel Execution Thread only accesses a small percentage of rows).

Increasing PGA will improve timings if there I/O waits on temporary space ("direct path read temp" and "direct path write temp"). There may also be physical reads showing on Hash Join, Sort or Group By operations in Display Cursor and SQL Monitor Reports.

Increasing SGA may improve timings if there are medium to high (>10%) I/O waits on database objects (e.g. db file sequential read, db file scattered read, direct path read, direct path write etc.)

Note that a very large SGA/buffer cache can cause excessive paging or swapping. It can also cause excessive latch waits as blocks in the buffer share the same latches.

Some starting rules of thumb are:

log buffer = 30 to 100 Mb

shared pool = 1 to 4 GB

pga target = 3 to 20 GB

SGA / buffer cache = multi GB. Be generous, but avoid causing excessive paging.

SGA and HugePages (Linux)

Using HugePages on Linux can improve SGA allocation and hence performance.

See My Oracle Support document "HugePages on Oracle Linux 64-bit (Document 361468.1)" for information on how to configure HugePages.

Use the script in My Oracle Support document "Shell Script to Calculate Values Recommended Linux HugePages / HugeTLB Configuration (Document 401749.1)" to calculate the recommend hugepages configuration.

Note that PGA allocation does not benefit from HugePages.

db_file_multiblock_read_count

If specified, remove db_file_multiblock_read_count. This is the recommended value for normal running of Oracle E-Business Suite.

Dynamic Sampling

A number of key high volume tables (particularly in the Sub Ledger Accounting (XLA) schema) are created and populated during the Release 12 upgrade. These often have no CBO statistics gathered (prior to adsstats.sql running), and dynamic sampling could be of benefit.

The Release 12.2.n upgrade is a batch process, and most of the SQL is executed less frequently and for a larger number of rows. So increasing the dynamic sampling level should not be much of an overhead, provided the sample size (blocks) is not increased too much.

For Oracle E-Business Suite, it is normally recommended not to set the database parameter optimizer_dynamic_sampling, which means that the default value (2) will be used. However, a value of 4 is recommended for the upgrade, provided it is reverted to default (2) afterwards.

Note that, at level 4, the sample size is 64 blocks. Level 5, 6, 7, 8, 9 and 10 increase the sample size to 128, 256, 512, 1024, 4086 and All blocks.

AD Parallel Batch Size

10K is suitable for most installs: test other values if time allows. Changing the batch size is not normally recommended. As this is limited by the extent size on the driving table, the ability to change the actual batch size used will be limited anyway.

Only change the batch size if there are serious performance issues that can be solved by increasing or decreasing the batch size (see below). And only once the SQL concerned has been confirmed as having optimal execution plans.

Having larger extent sizes for the larger transaction tables populated or modified by the Release 12.2.0 upgrade is suggested. In any case, it is best practice to have larger extent sizes for tables with high growth.

Note there are some limitations on batch size and how it is calculated. These are described below.

Small extents (e.g. 128k = 16 blocks) will significantly limit the batch size. The batches are likely to much less than 1000, regardless of what batch size is selected.

Even if there are large extents, the actual batch sizes may vary considerably from the ones chosen because of storage overheads (percent free, block/row headers) and compression.

There are advantages and disadvantages with large and small batch sizes. Larger batch sizes have less overhead, and can reduce the overhead of sub-optimal execution plans. However, there is more risk of the workers trying to perform the same operations at the same time, and thus causing contention (especially if there are multiple pieces of SQL in the job). There is also more risk of remainder batches delaying the end time of a job.

Note that rows_processed column on AD_PARALLEL_UPDATE_UNITS can often contain values that do not correspond to the actual batch size used (or entered). The value of rows processed is usually SQL%ROWCOUNT

following a DML statement, so it is typically the number of rows inserted or updated as a result of the SQL, not the number of rows used to drive the SQL. Sometimes there is more than one SQL/DML in the job and it is the count from the latest SQL.

Changing the AD Parallel Batch Size

Only change the batch size if there are experience serious performance issues that can be resolved by increasing or decreasing the batch size. And before doing so, check that the execution plans are optimal.

- High overhead of repeated SQL repeatedly performing the same operation => Increase batch size.
- High overhead of repeated full scans/range scans that cannot be solved by tuning => Increase batch size
- High contention between workers, occurring in highly defined peaks, at start of job (workers are doing same activity at same time/synchronized) => Decrease batch size so that workers become unsynchronized sooner.
- Low worker utilization at end of a job due to large remainder batch => Decrease batch size.

Be aware that changing the overall batch size is a very imprecise action. Although it may improve the performance of one job, the performance of another job may regress.

The batch size for individual AD Parallel jobs can be changed by hard coding the batch size in the SQL script. However, (because of the possibility of extreme values being chosen or editing errors) this is not strictly supported.

e.g.

```
 --l_batch_size               VARCHAR2(30) := &&2;
  l_batch_size               VARCHAR2(30) := 100000;
```

Or in the unified driver itself.

e.g.

```
#sql bom patch/115/sql cstmtaupg.sql none none none sqlplus_repeat &phase=upg+72
checkfile:bom:patch/115/sql:cstmtaupg.sql &un_inv &batchsize
sql bom patch/115/sql cstmtaupg.sql none none none sqlplus_repeat &phase=upg+72
checkfile:bom:patch/115/sql:cstmtaupg.sql &un_inv 100000
```

Batch Size Calculation and its limitations

Note that (with the current version of AD_PARALLEL_UPDATES_PKG - adprupdb.pls 120.1 2006/05/08) there are some limitations on batch size and how it is calculated.

This means that the actual batch sizes can often be quite different from those entered.

The extent size can also cause some limitations with batch size.

The code in AD_PARALLEL_UPDATE_PKG calculates the number of blocks to be used for each batch as:

1. B = (batch_size * Average_row_length)/8192
2. It rounds this to the nearest 10.
3. If the average row length is 0 or the result is 0 then a default of 200 blocks is used.

Each extent on the driving table is then split into "Update Units" containing B blocks each, with the remainder being placed into its own update unit. Each update unit is then used as a batch.

There are several limitations:

- If B is greater than the extent size, the whole extent is used as an update unit. Consequently, the actual batch size cannot be greater than the rows that are stored in an extent.
- When calculating the number of blocks (B) required for a batch, AD_PARALLEL_UPDATES_PKG does not take into account the storage overhead for the block header, percent free or row header, and so can underestimate the number of blocks required. This effect is magnified if the average row length is small.
- If advanced compression or compression is used, this is not reflected in the average row length. The average row length will consequently be overstated, and a larger number of blocks used for each batch.
- Rounding the number of blocks required to the nearest 10 adds an additional inaccuracy.
- Splitting each extent into update units of B blocks also adds inaccuracy. If the number of remainder blocks is much less than B, this remainder batch will be much smaller. If there is a low number of update units per extent, this will have more influence on the batch size.
- If the CBO Statistics are not populated, the smaller of either 200 blocks or the extent size will be used as a batch size.

This all means that the actual batch is usually smaller than that entered, and can vary in size significantly. It can be therefore be difficult to manage.

Looking at the following three examples will help explain all this.

Example 1

Average Row Length 200, batch size 10000, extent size 256 blocks, percent free and storage overheads result in only 70% of block being occupied by row data.

B = 244.1 blocks = 240 blocks (rounded to the nearest 10).

So that will result in two update units for each extent: one of 240 blocks, and one of 16 blocks.

This will in turn result in two batch sizes:

1 x 240 blocks * (8192 bytes / average row length 200) * 70% = Average of 6881 rows

1 x 16 blocks * (8192 bytes / average row length 200) * 70% = Average of 459 rows.

Example 2

Average Row Length 200, batch size 10000, extent size 16 blocks, percent free and storage overheads result in only 70% of block being occupied by row data.

B = 244.1 blocks = 240 blocks (rounded to the nearest 10).

However, the extent size is 16. So the batch is limited to 16 blocks.

This will result in a batch size of 16 blocks * (8192 / actual average row length 200) * 70% = Average of 459

Example 3

Average Row Length 200, batch size 10000, extent size 256 blocks, percent free and storage overheads result in only 70% of block being occupied by row data, compression ratio is 2.

B = 244.1 blocks = 240 blocks (rounded to the nearest 10).

So that will result in two update units for each extent. One of 240 blocks and one of 16 blocks.

This will in turn result in two batches:

1 x 240 blocks * (8192 bytes / average row length 200) * 70% * compression ratio 2 = Average of 13,762 rows

1 x 16 blocks * (8192 bytes / average row length 200) * 70% * compression ratio 2 = Average of 918 rows.

Remainder Batch Issue if Batch Size is too large

There is a risk that a very large (actual) batch size could result in a job being unduly delayed by a small number of batches that process after all other batches.

Let's refer to these as remainder batches.

If T = number of threads (AD workers), R = total rows to be processed, B = Batch size

And B is slightly less than R/T (e.g. R/(T+1))

Then this will result in T+1 batches assigned across T threads.

So one thread will have twice as much work as the others.

If 192,000 rows (of the driving table) were processed on 32 workers then R/T would be around 6,000.

So (assuming a large extent size) choosing a batch size of 10,000 might result in remainder batches. Therefore, in this case a much smaller batch size (such as 1,000) might be preferable.

However, do not reduce the batch size excessively, as small batch sizes can result in larger overheads on each batch.

Dynamic / Hyper Threading

Many hardware vendors are now offering a version of Dynamic Threading or Hyper Threading. In these cases the hardware can be configured to present virtual CPUs (also known as Logical CPUs) to the operating system (and ultimately database). This number of virtual CPUs will be much larger than the number of cores.

For example, the Oracle SPARC T5-8 has up to 8 sockets (physical CPUS) with 16 cores each: 128 cores in total. However, with Dynamic Threading it can run a maximum of up to 1024 virtual CPUs (or threads).

This is ideal for applications with a large number of online users, where the number of active sessions is many times the number of CPUs. Traditionally these active sessions would have shared the CPUs and there would have been a large amount of context switching. Dynamic threading can reduce context switching and thereby improve performance.

However, this is not needed for cases such as the Release 12.2.n upgrade, where there are a smaller number of active sessions, each with a high workload. In this case, the CPU capability available to each AD Parallel (or Parallel Execution) worker/thread, could be limited.

Clearly a large number of workers/threads (e.g. 256) all performing the same operation will result in massive contention that will far outweigh the benefits of greater CPU utilization.

So the recommendation is to configure the dynamic threading/hyper threading for maximum throughput per CPU cycle – i.e. 1 thread for each core. On Oracle SPARC T4-4 and T5-8 this was historically done by setting dynamic threading to max-ipc.

In addition, disable any power management as this will either prevent configuration of threading for maximum throughput, or throttle the CPU capacity.

Resource Manager

If using a Resource Plan to specify how resources are distributed among the different resource consumer groups, please review this to ensure that all sessions running during the upgrade downtime window have access to the full CPU resources.

Note that this does not apply to Post Upgrade Concurrent jobs, which will run alongside the normal online workload.

12.2 Middle Tier Sizing Guidelines

Managed instances JVM sizing should consider both memory and CPU domains.

On 64bit environments, allocating huge heap sizes is not recommended, but rather have more managed instances in the cluster to scale up to the target concurrency levels.

For Admin Server sizing, the default size of 512M is not enough for most installations, setting the XMS to at least 1 GB and the XMX to 2GB is recommended.

An initial guidance on sizing can be found in

- Oracle E-Business Suite Installation Guide: Using Rapid Install, Release 12.2
- My Oracle Support document "Managing Configuration of Oracle HTTP Server and Web Application Services in Oracle E-Business Suite Release 12.2 (Document 1905593.1)"

Using "downtime mode" for Online Patching

It is recommended that the “Oracle E-Business Suite 12.2.4 Release Update Pack (Patch 17919161)” or “Oracle E-Business Suite 12.2.5 Release Update Pack (Patch 19676458)” is applied using "downtime" mode. And this will help reduce the elapsed time.

Gathering CBO Statistics

For more information see My Oracle Support document "Best Practices for Gathering Statistics with Oracle E-Business Suite (Document 1586374.1)".

Gathering Oracle E-Business Suite Schema Statistics

In the pre-upgrade environment, schema statistics for all Oracle E-Business Suite schemas should have been gathered using FND_STATS (or the Gather Statistics concurrent program) with the GATHER_AUTO option.

The Release 12.2.n upgrade will gather all Oracle E-Business Suite schema statistics again (using adsstats.sql) towards the end of the R12.2.0 upgrade (last+63). Note that adsstats.sql is not run in the R12.2.n RUPs.

It is important to be aware of the difference between adsstats.sql and adstats.sql. They are two distinct scripts:

- adsstats.sql gathers Oracle E-Business Suite schema statistics.
- adstats.sql disables the 10g/11g automatic statistics gather jobs, and gathers the Dictionary and Fixed Object Statistics.

After the Release 12.2.n upgrade, but prior to the system being available to users, gather CBO statistics again for all Oracle E-Business Suite schemas using FND_STATS (or the Gather Statistics concurrent program) with the GATHER_AUTO option. Do not use DBMS_STATS or Analyze.

In principle, CBO statistics for Oracle E-Business Suite schemas should not need to be gathered at any other time during the Release 12.2.n upgrade. However, in practice, as a result of performance issues, the statistics for specific objects may need to be gathered or deleted (see Resolving Performance Issues section), especially on tables created by the Release 12.2.n upgrade or on temporary/transient tables.

Importing Oracle E-Business Suite Schema Statistics

If the adsstats.sql script is taking a significant amount of time to run, the Release 12.2.n upgrade time can be reduced by:

- Exporting statistics gathered during test runs (by adsstats.sql - at same point: phase: last+63).
- Importing these statistics instead of running adsstats.sql.

Only do this if the test environment has the same data volumes as the production environment (i.e. is a clone of production).

However, this does complicate the upgrade. So only do this if the estimated time savings will be significant, and there is confidence in adopting this approach.

Note that if parallel_max_servers is set to a value much less than 2 x number of cores, consider increasing parallel_max_servers before resorting to export/import.

If importing statistics is also long running, it may have a performance issue that could be resolved by gathering fixed object and dictionary statistics prior to importing. Fixed object and dictionary statistics would normally only be gathered after the Release 12.2.n upgrade is complete (i.e. after R12.2.n RUP). So these are additional activities and their elapsed time should be deducted from any gains made by importing statistics.

The statistics can be exported using:

```
FND_STATS.backup_schema_stats(schema_name => 'ALL', statid => '<your statid>')
```

And imported using:

```
FND_STATS.restore_schema_stats(schema_name => 'ALL', statid => '<your statid>')
```

There are two reasons that FND_STATS.backup_schema_stats and restore_schema_stats should be used:

- Only FND_STATS is supported with Oracle E-Business Suite.
- FND_STATS.backup_schema_stats and restore_schema_stats will only backup and restore Oracle E-Business Suite schemas (around 85% of objects on a Release 12.2 database). In contrast, DBMS_STATS.export_database_stats and import_database_stats will export/import all schema statistics and also gather fixed object, dictionary and system statistics. So the workload will be considerably less if FND_STATS is used.

The following SQL scripts are suggested for exporting and importing statistics.

Exporting Statistics

```
set verify off
whenever sqlerror exit failure rollback;
whenever oserror  exit failure rollback;

declare
begin
FND_STATS.BACKUP_SCHEMA_STATS(schemaname => 'ALL', statid => '<your identifier>');
exception
when others then
raise_application_error(-20000, sqlerrm ||' Error while executing
                                FND_STATS.BACKUP_SCHEMA_STATS package.');
end;
/
exit;
```

Importing Statistics

```
set verify off
whenever sqlerror exit failure rollback;
whenever oserror  exit failure rollback;

declare
begin
FND_STATS.RESTORE_SCHEMA_STATS(schemaname => 'ALL', statid => '<your identifier>');
exception
when others then
raise_application_error(-20000, sqlerrm ||' Error while executing
                                FND_STATS.RESTORE_SCHEMA_STATS package.');
end;
/
exit;
```

However, at the time of writing there is an issue with DBMS_STATS.IMPORT_SCHEMA_STATS: if a schema has no tables, an exception will be raised that causes FND_STATS.RESTORE_SCHEMA_STATS to fail before completion. There are several Oracle E-Business Suite schemas with no tables.

A workaround is to use the following script:

```
set verify off
whenever sqlerror exit failure rollback;
whenever oserror exit failure rollback;

DECLARE
CURSOR schema_cur IS
WITH
schema_tabcount AS
(SELECT c5, COUNT(*) num_obj
FROM fnd_stattab
WHERE statid = <your identifier>'
GROUP BY c5
```


```
)
SELECT upper(oracle_username) sname
FROM   fnd_oracle_userid
WHERE oracle_id BETWEEN 900 AND 999
AND read_only_flag = 'U'
AND EXISTS
(SELECT 'exists' from schema_tabcount WHERE c5 = upper(oracle_username) AND num_obj >0)
UNION ALL
SELECT DISTINCT upper(oracle_username) sname
FROM            fnd_oracle_userid a,
                fnd_product_installations b
WHERE           a.oracle_id = b.oracle_id
AND EXISTS
(SELECT 'exists' from schema_tabcount WHERE c5 = upper(oracle_username) AND num_obj >0)
ORDER BY        sname;

BEGIN
   FOR c_schema IN schema_cur
   LOOP
      BEGIN
         DBMS_STATS.IMPORT_SCHEMA_STATS(c_schema.sname, 'FND_STATTAB', <your
identifier>', 'APPLSYS');
      EXCEPTION
         WHEN OTHERS THEN
            dbms_output.put_line('Error on '||c_schema.sname);
            dbms_output.put_line(SUBSTR(SQLERRM,1,80));
      END;
   END LOOP;
EXCEPTION
WHEN OTHERS THEN
raise_application_error(-20000, sqlerrm ||' Error while executing adsstats_restore');
end;
/
exit;
```

Further strategies for reducing adsstats.sql elapsed time

If the adsstats.sql job is taking a long time during the R12.2.0 upgrade there are two further approaches that may help for large environments, where there are tables with more than 100 million rows.

Sample specific long running tables at a lower percentage

Check the table FND_STATS_HIST to see which tables are taking the longest.

Use the following SQL to show the latest FND_STATS runs (request_id) with the number of tables analyzed. The <request_id> for the adsstats.sql run in the R12.2.0 upgrade can then be identified.

```
SELECT
request_id,
count(*) tables_analyzed,
TO_CHAR(MAX(last_gather_end_time),'DD-MON-YYYY HH24:MI:SS') max_end_time,
MAX(last_gather_end_time) max_end_time_int
```


```
FROM fnd_stats_hist
WHERE object_type = 'CASCADE'
AND last_gather_end_time IS NOT NULL
GROUP BY request_id
ORDER BY MAX(last_gather_end_time) DESC
```

Once the request_id has been identified, the following SQL will give the tables where gathering statistics took the longest along with information on the number of rows, blocks and sample size.

```
SELECT fsh.schema_name, fsh.object_name, fsh.object_type,
ROUND((fsh.last_gather_end_time - fsh.last_gather_start_time)*24*3600,0) time_secs,
to_char(fsh.last_gather_start_time,'DD-MON-YYYY HH24:MI:SS') start_time,
to_char(fsh.last_gather_end_time,'DD-MON-YYYY HH24:MI:SS') end_time,
dt.num_rows,
dt.blocks,
dt.avg_row_len,
ROUND(DECODE(NVL(num_rows,0),0,NULL,(sample_size*100)/num_rows),0) est_pct,
sample_size
FROM fnd_stats_hist fsh
     ,dba_tables dt
WHERE dt.owner (+) = fsh.schema_name
AND dt.table_name (+) = fsh.object_name
-- AND fsh.request_id = <request_id>
-- AND fsh.object_type = 'CASCADE'
AND fsh.last_gather_end_time IS NOT NULL
ORDER BY (fsh.last_gather_end_time - fsh.last_gather_start_time)*24*3600 desc
```

The longest running ones are candidates for running at a lower estimate percentage (sample size) on the next run. By sampling at a lower percentage the accuracy of the CBO statistics will be reduced, so take care to monitor the performance of post R12.2.0 activities (e.g. R12.2.n RUP and post upgrade patches/jobs).

These tables can then be gathered at a lower percentage directly before running adsstats.sql. Or adsstats.sql can be edited directly to gather these tables at a lower percentage prior to gathering schema statistics (FND_STATS.GATHER_SCHEMA_STATISTICS).

The command is:

```
fnd_stats.gather_table_stats('<owner>', <table_name>', <estimate_percent>,
<parallel_degree>);
```

Where <parallel-degree> is the setting of parallel_max_servers (e.g. v_parallel in adsstats.sql itself).

It is advised that the following are used:

- 3 percent for tables between 100 million and 1 billion rows

- 1 percent for tables with more than 1 billion rows.

Incremental Statistics for Partitioned Tables

Incremental statistics gathering is a new Oracle 11gR2 DBMS_STATS feature and is fully supported by FND_STATS. Oracle derives global statistics by scanning only new or modified partitions rather than the whole table, which is highly beneficial with high volume Oracle E-Business Suite tables.

This will have no impact for partitioned tables during the upgrade itself, but it will speed up statistics gathering post go live.

It should be set for the following new partitioned tables: XLA_AE_HEADERS, XLA_AE_LINES, XLA_DISTRIBUTION_LINKS, XLA_EVENTS, XLA_TRANSACTION_ENTITIES, XLA_TRIAL_BALANCES, AP_DBI_LOG, AP_LIABILITY_BALANCE.

Run the following command to invoke it:

dbms_stats.set_table_prefs('<schema_name>', '<table_name>', 'INCREMENTAL', 'TRUE');

Locked Statistics

There may be a reason to lock the statistics on some tables. For example, the table could be a temporary or transient table (or Global Temporary table), where a representative set of statistics has been gathered, which should always be used and never overwritten.

In such cases, be aware that the adsstats.sql script will unlock the statistics and re-gather. And it is likely that these tables will be empty (or not have representative data volumes) during the Release 12.2.n upgrade.

It would be useful to obtain a list of these prior to the upgrade, so that they can be locked and excluded.

For Oracle E-Business Suite *lock statistics using the DBMS_STATS.LOCK_TABLE_STATS procedure, and then exclude them from subsequent FND_STATS gathering jobs by using FND_STATS.LOAD_XCLUD_TAB procedure.*

Once the statistics are locked using the DBMS_STATS.LOCK_TABLE_STATS procedure, FND_STATS will skip the tables on which statistics are locked in the subsequent statistics gathering jobs. Gather Schema / Table Statistics Concurrent Program will display a message in the request log file saying that statistics are locked on the table.

FND_STATS.LOAD_XCLUD_TAB will tell FND_STATS to skip the table from the Gather Schema Statistics Concurrent Program. It will seed an entry in the FND_EXCLUDE_TABLE_STATS table.

Fixed Object and Dictionary Statistics

These should have been gathered on the pre-upgrade environment.

Oracle Database 10g and above use the CBO for most of the dictionary SQL, so statistics on the dictionary objects and fixed objects must be gathered.

A fixed object (X$ tables) resides in memory only, and typically records the information about the instance or memory structures. The v$ dynamic performance views are defined on top of X$ tables e.g. V$SQL and V$SQL_PLAN.

Data dictionary tables (e.g. SYS.USER$, SYS.TS$, SYS.SEG$, SYS.OBJ$, SYS.TAB$, SYS.FILE) are stored on data files like normal application tables.

If there is internal SQL (on V$ views or on SYS/SYSTEM objects) appearing high in AWR and TKPROF reports, it is likely that dictionary and fixed object statistics need to be gathered.

Note that the FND_STATS API does not gather statistics for dictionary or fixed objects. The DBMS_STATS APIs need to be used.

Fixed Object Statistics

Typically, fixed object statistics will need to be gathered when there have been significant changes to fixed objects. For example, after:

- A major database upgrade.
- Platform upgrades, especially Exadata.
- Application upgrades, or addition of a new application module.
- Changes to the SGA/PGA configuration of the database.
- Significant changes in the workload or number of sessions.

For optimal performance, statistics on fixed objects should be gathered when there is representative activity (typical load) on the system.

The command to run is:

```
execute DBMS_STATS.GATHER_FIXED_OBJECTS_STATS(no_invalidate=>FALSE);
```

Usually the "no_invalidate=>FALSE" argument will not be needed. However, the procedures DBMS_STATS.set_database_prefs, set_global_pefs, set_schema_prefs or set_table_prefs could have been used to set the default value for NO_INVALIDATE to TRUE

For the Release 12.2.n upgrade, the fixed object statistics should be gathered:

- After any associated platform or database upgrade that is part of the overall Oracle E-Business Suite upgrade.
- After any SGA/PGA parameters have changed.
- After the Release 12.2.n upgrade, when there is representative activity on the system.

There are also changes to fixed objects due to due to Online Patching Enablement (and the underlying Edition-Based Redefinition). As a result internal SQL in Online Patching Enablement, R12.2.n RUPs and other online patches can sometimes be long running. Gathering fixed object statistics can help in these circumstances.

See the following My Oracle Support documents for more information:

- Best Practices for Gathering Statistics with Oracle E-Business Suite (Document 1586374.1)
- Fixed Objects Statistics (GATHER_FIXED_OBJECTS_STATS) Considerations (Document 798257.1)

Dictionary Statistics

Normally, the Dictionary Statistics need to be analyzed after a sufficient number of DDL operations have occurred in the database (for example, if there is a significant change in the database when a lot of new objects are created or rebuilt).

Use the following command (in Oracle Database 10g and above) to gather statistics for all system schemas, including SYS and SYSTEM.

```
execute DBMS_STATS.GATHER_DICTIONARY_STATS( estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, options => 'GATHER AUTO',
no_invalidate=>FALSE);
```

Usually the "no_invalidate=>FALSE" argument will not be needed. However, the procedures DBMS_STATS.set_database_prefs, set_global_pefs, set_schema_prefs or set_table_prefs could have been used to set the default value for NO_INVALIDATE to TRUE.

It is not normally feasible to parallelize the gathering of dictionary statistics.

Dictionary statistics are gathered in the degree of the table (which will normally be 1). So they will normally be gathered serially. There is the option of setting degree=AUTO_DEGREE. But this parameter is ignored on options

GATHER_AUTO, so GATHER would have to be used instead. So, parallel processing can be used at the expense of gathering for all tables (whether they need to be gathered or not). This is a choice for each customer.

For the Release 12.2.n upgrade, Dictionary Statistics should be gathered:

- After any associated platform or DB upgrade that is part of the overall Oracle E-Business Suite upgrade.
- After the Release 12.2.n upgrade.
- After move to OATM.

There are also changes to dictionary objects due to due to Online Patching Enablement (and the underlying Edition-Based Redefinition). As a result internal SQL in Online Patching Enablement, R12.2.n RUPs and other online patches can sometimes be long running. Gathering dictionary statistics can help in these circumstances. Particularly on editioning objects.

For more information see My Oracle Support document "Best Practices for Gathering Statistics with Oracle E-Business Suite (Document 1586374.1)".

Gathering Statistics for Specific Dictionary or Fixed Objects

If there are only a handful of internal SQLs with inefficient execution plans and only a few objects then specific objects could be targeted rather than gathering all dictionary or fixed object statistics.

e.g.

exec dbms_stats.gather_table_stats(ownname=>'SYS', tabname=>'OBJ$', no_invalidate=>false);

exec dbms_stats.gather_table_stats(ownname=>'SYS', tabname=>'X$KQFP', no_invalidate=>false);

adstats.sql

This script:

- Disables the Oracle 10g/11g Automatic Statistics gather jobs.
- Gathers Dictionary and Fixed Object Statistics.

However, it can only be run with the database in restricted mode.

Do not use this script to gather fixed object or dictionary statistics as part of the Oracle E-Business Suite Release 12.2.n upgrade itself (although it may be run as part of an associated database upgrade). Use the APIs directly, as instructed above.

It is recommended that this script is run as part of the upgrades of Oracle E-Business Suite instances to Oracle Database 11g. Details are included in the upgrade instructions.

Notes:

- The adstats.sql script is not run automatically as part of the Release 12.2.n upgrade, and will not be in the R12.2.0 or other drivers.
- For optimal performance, statistics on fixed objects should be gathered when there is a typical load in the system. This script is designed to be run only when the database is in restricted mode.
- The script does not gather system statistics.
- As mentioned earlier, adstats.sql is sometimes confused with adsstats.sql, which gathers Oracle E-Business Suite schema statistics.

Resolving Performance Issues

This section includes guidance on:

- Diagnostics to gather during the upgrade, particularly diagnostics to gather when long running jobs and SQL are encountered.
- Common (generic) issues and their solutions. Specific issues will be covered in bugs on My Oracle Support.

Diagnostics to be gathered during the upgrade

Note that the new diagnostics available for online patching (ADOP) are logs. These give timestamps for particular steps in the upgrade, or list errors that have occurred. However, they do not identify the SQLs or underlying events that caused the performance issue, so it is essential to use the general diagnostics listed below (in section "General Diagnosis of Oracle E-Business Suite Release 12.2.n Upgrade Performance Issues").

For most upgrades, it is recommended that Automatic Workload Repository is enabled, with a snapshot of 30 minutes (default is 60 minutes). For shorter upgrades, a smaller snapshot may be more suitable.

The AWR retention period should be long enough to cover the duration of the upgrade run and a significant period afterwards (to gather diagnostics and analyze). The suggestion is N+7 days, where N is the estimated upgrade time, but a longer period will provide more time to gather subsequent diagnostics and statistics.

It is strongly recommend that statistics_level is set to ALL (or _rowsource_execution_statistics=TRUE) for the duration of the Release 12.2.n upgrade dry runs.

Uploading Files to Service Requests

In some cases it is not feasible/possible to upload files to an SR using My Oracle Support (e.g. the file is too large). In this case files could be uploaded using SFTP. See My Oracle Support document "How to Send a File to Oracle Support Using FTPS (Document 464666.1)".

General Diagnosis of Oracle E-Business Suite Release 12.2.n Upgrade Performance Issues

The following diagnostics are still required for all patches/phases of the upgrade. This includes Online Patching Enablement, the online patching (ADOP) phases (e.g. prepare, apply, finalize, cutover, cleanup) of post 12.2 patches, especially the 12.2.n RUPs.

When analyzing Release 12.2.n upgrade performance issues, the goal is to:

- Prevent wasted test iterations. Aim to provide solutions that solve the issue first time.
- Maximize the number of performance issues resolved.

Upgrade jobs cannot be tested in isolation. They can only be tested on the next iteration. If a fix does not work, it is a potential wasted test iteration.

To do this the following are needed:

- Actual statistics: So it is possible to see exactly which execution plan steps are inefficient, rather than those that might be inefficient. The likely impact of a performance fix can also be estimated. There will then be a higher probability of providing a fix that can solve the performance issue first time. Also, it will be possible to identify marginal fixes (i.e. fixes that reduce elapsed times by 10-50%, for example by having more specific index access). These fixes often reduce contention between workers.
- Diagnostics that are quick and easy to run.

- Diagnostics that have very little impact on the performance of the Release 12.2.n upgrade. If they can safely be run during the upgrade then the results are be obtained sooner and the issue resolved more quickly.

The key preparatory steps are:

- Before the Release 12.2.n upgrade, set the statistics_level to ALL (or _rowsource_execution_statistics = TRUE).
- Automatic Workload Repository (AWR) should be enabled with a snapshot of 30 minutes (the default is 60 minutes). For short upgrades, a shorter snapshot may be more suitable.
- The AWR retention period should be long enough to cover the duration of the upgrade run and a significant period afterwards (to gather diagnostics and analyze). The suggestion is N+7 days, where N is the estimated upgrade time, but a longer period will provide more time to gather subsequent diagnostics and statistics.

During the upgrade:

- Monitor top SQL in AWR or Cursor Cache (memory) (see useful scripts). This could be internal or application SQL. Enterprise Manager can also be used to identify expensive SQL as it occurs.
- Obtain Display Cursor Reports (ALL +ALLSTATS) for long-running SQL
- Obtain SQL Monitor Reports for SQL that uses Parallel Query or DML
- Identify when a piece of SQL ran (see useful scripts)
- Match long-running SQL with a job (see useful scripts)
- Report on CBO Statistics for all Oracle E-Business Suite tables (see useful scripts)

After the Release 12.2.n upgrade:

- Obtain AD Job Timing Reports
- Identify long-running upgrade Jobs (see useful scripts)
- Obtain the file versions for long running jobs
- Obtain AWR reports
- Run the afxplain.sql script to obtain detailed CBO statistics, database parameters, CBO parameters, and metadata: do this for individual pieces of SQL.

Other Diagnostics

For more detailed analysis the following will be required:

- AD utility and worker logs.
- SQLT With XTRACT for long-running SQL

In addition, Oracle Support and Development may require:

- AWR Export
- AD Parallel tables export

Online Patching Enablement - Specific Diagnostics to be gathered during the upgrade

In addition the following diagnostics are available during the Online Patching Enablement patch.

- ADZDSHOWDDLS.sql

Online Patching - Specific Diagnostics to be gathered during the upgrade

In addition the following diagnostics are available for patches applied using online patching.

- ADOP log directories
- The Online Patching Log Analyzer Utility adopscanlog (delivered in R12.AD.C.Delta.n since R12.AD.C.Delta.4). This analyzes the adop log directories.
- ADZDSHOWLOG.sql / adzdshowlog.out
- ADOP cycle status
- SQL to identify status of the ADOP phases
- SQL to identify the AD and TXK C Patch levels

For more guidance see:

- Oracle E-Business Suite Maintenance Guide Release 12.2
- Oracle E-Business Suite Upgrade Guide Release 11i to 12.2
- Oracle E-Business Suite Upgrade Guide Release 12.0 and 12.1 to 12.2

And My Oracle Support Document:

- 12.2 E-Business Suite - Collecting Online Patching and fs_clone Log Files (Document 1542162.1)

Statistics_Level = ALL (or _rowsource_execution_statistics = TRUE)

This can be set using the command:

SQL>alter system set statistics_level='ALL'

This is the simplest way to see actual row source statistics (including elapsed time, physical reads, buffer gets etc) for each execution plan line (on SQLT and Display Cursor report). The alternative of SQL Trace and TKPROF requires editing standard code.

Note that the internal views v$sql_plan_statistics and v$sql_plan_statistics_all will not contain any actual row source statistics for execution plan lines if statistics_level = TYPICAL, even if timed_statistics = TRUE.

When the STATISTICS_LEVEL parameter is set to ALL, additional statistics are added to the set of statistics collected with the TYPICAL setting. The additional statistics are timed OS statistics and plan execution statistics, which include row source statistics.

Using this strategy will typically speed up the resolution of issues significantly and may also allow the correct solution to be identified first time.

Alternatively, the same actual execution statistics can be collected by setting the initialization parameter _rowsource_execution_statistics=TRUE (with statistics_level = 'TYPICAL'). This gives a lower overhead than statistics_level=ALL.

Many technical architects and DBAs at customers (or implementing partners) can be resistant to setting statistics_level = ALL (or _rowsource_execution_statistics = TRUE), believing that this can slow down performance significantly.

Two points are relevant here:

- Although setting statistics_level = ALL / _rowsource_execution_statistics = TRUE will have some performance impact, it is likely to be small and not significant. The Release 12 upgrade is made up of batch processes, and so the statistics workload is a much lower proportion of the total.
- Even if the performance impact is significant, the goal is to reduce the elapsed times for the latter dry runs and go live (when it will be feasible to revert statistics_level / _rowsource_execution_statistics to its previous value). So suffering an increase in elapsed time during an early stage of testing is not an issue.

So there may be a small impact on elapsed time and the work that needs to be done initially, but it will help to subsequently reduce the elapsed time and amount of re-work that needs to be done.

Note that setting statistics_level to ALL while AWR is enabled could significantly increase the number of rows inserted to the WRH$_LATCH_CHILDREN table. So monitor the SYSAUX tablespace to ensure that it does not run out of space.

SQL Trace – SQL Script Specific

If it is not possible to resolve a performance issue from the display cursor, SQL monitor and AWR diagnostics, then obtain a 10046 SQL Trace (level 16 – ALL_EXECUTIONS (with waits)) for the script/job on the next test run.

Level 16 (ALL_EXECUTIONS) is required because sometimes (especially for AD Parallel jobs) performance issues do not occur until later executions and the first execution (default on level 8) does not give enough information.

SQL Trace can be enabled at system level. However, this will produce a large number of trace files, and it may be difficult to locate the correct traces. (Tip: on UNIX, use *grep* to search files for key SQL strings or script/module names).

Although it is possible to SQL Trace other sessions using DBMS_MONITOR (10g and above) or the Event++ syntax (11g and above).

See My Oracle Support document "Oracle E-Business Suite SQL Trace and TKPROF Guide (Document 1674024.1)" section "Obtaining Traces (TKPROF) in E-Business Suite" > "For another database session using DBMS_MONITOR (10g and above)" / "For another database session using the Event++ syntax (11g and above)".

There are some limitations in this case.

- *To obtain the diagnostics needed (parse, execute and fetch statistics; row source statistics) then the trace should be enabled before the process or SQL has started.*
- *There must be a way of identifying the sessions to be traced. E.g. client identifier (which is not set during the R12.2.n Upgrade scripts), service, module (which is sometimes set), action, session ID and Serial Number, OS Process ID, Oracle Process ID or Process Name. These may not be known before the job/script is run.*
- *Also, there could be multiple parallel slaves or AD workers.*

For this reason it is advisable to enable SQL Trace for specific scripts by temporarily editing the SQL script as follows:

Add the following before the main body of the script.

- alter session set tracefile_identifier='<identifier string>';
- alter session set events '10046 trace name context forever, level 16';
- alter session set statistics_level='ALL' ;
- alter session set max_dump_file_size='UNLIMITED';

Add the following after the main body of the script.

- alter session set events '10046 trace name context off';

Another advantage of enabling the trace at script level is that each script can be given different trace identifiers, so finding the traces becomes easy.

For more information see My Oracle Support document "Oracle E-Business Suite SQL Trace and TKPROF Guide (Document 1674024.1)", especially section "Obtaining Traces (TKPROF) in E-Business Suite" > "From SQL*Plus".

SQL Specific

Once AWR or TKPROF have been used to identify the long-running SQL (in the long-running jobs), the output below should be provided:

Note that for these to be useful, statistics_level should be set to ALL (or _rowsource_execution_statistics = TRUE). The actual statistics are needed to be able to identify the expensive execution plan steps or execution plans that are moderately sub-optimal. (The execution plan itself may look OK without this information).

Display Cursor Report

This displays the actual execution plan of any cursor loaded in the cursor cache. At the basic level it shows the runtime execution plan. However, the format ALL also includes extra information such as pruning, parallel execution, predicate, projection, alias and remote SQL information.

The +ALLSTATS option (which includes IOSTATS and MEMSTATS) will include actual statistics for each execution plan step. These include:

- Elapsed time
- Physical reads
- Buffer gets

- Memory used (in PGA) for memory intensive operations (such as hash-joins, sorts, bitmap operators etc).

However, this additional information is only provided if statistics_level=ALL / _rowsource_execution_statistics = TRUE

Note that SQLT with XTRACT will also report actual row source statistics in the same circumstances. However, display_cursor provides a simpler view of the information. It can also be run during the upgrade, while the long running SQL is in progress, without much of an overhead.

Note that the display cursor report should be run as soon as possible. If it is delayed, the cursor may have been flushed from memory or invalidated. In such a case, no data will be available.

The report can be produced by running the following SQL script:

```
SET pages 0
SET lines 300
SET LONG 10000
SET LONGCHUNKSIZE 10000
SPOOL<report_name>.txt
SELECT * FROM TABLE(dbms_xplan.display_cursor('<sql_id>', NULL, 'ALL +ALLSTATS'));
SPOOL OFF;
```

For more information see the "Display Cursor" section in My Oracle Support document "Oracle E-Business Suite Performance Guide (Document 1672174.1)"

If the SQL is no longer in memory, but is in the AWR, use the Display AWR report (DBMS_XPLAN.DISPLAY_AWR) instead. However, this does not report on actuals: it does not have a +ALLSTATS option, and there are no actual statistics for execution plan steps stored in AWR. Note that SQLT with XTRACT method will not report on actual statistics in this case either.

Note that the display cursor and AWR reports only show the sql_text (first 1000 characters) and not the full_text.

So, if necessary, run the following SQL script to obtain the full SQL text.

```
SET pages 0
SET lines 300
SET LONG 10000
SET LONGCHUNKSIZE 10000
SPOOL<report_name>.txt
SELECT sql_id, sql_text, sql_fulltext FROM v$SQL
WHERE sql_id = '<sql_id>';
SPOOL OFF;
```

SQL Monitor Report

The main advantage of this is that it gives a good view of how parallel SQL/DML performs across stages of the plan and parallel slaves.

It can also give a good idea of the actual executions and row counts for each execution plan line even if "statistics_level" initialization parameter is not set to ALL (or "_rowsource_execution_statistics" is not set to TRUE) at the time the SQL is executed.

It can be run during the upgrade, while the long running SQL is in progress, without much of an overhead.

It can be produced by running the following SQL script:

```
set trimspool on
set trim on
set pages 0
set long 10000000
set longchunksize 10000000
set linesize 200
set termout off
spool sql_monitor_for_<sql_id>.htm
variable my_rept CLOB;
BEGIN
:my_rept := dbms_sqltune.report_sql_monitor(sql_id => '<sql_id>', report_level =>
'ALL', type => 'HTML');
END;
/
print :my_rept

spool off;

set termout on
```

For more information see the "SQL Monitor Report" section in My Oracle Support document "Oracle E-Business Suite Performance Guide (Document 1672174.1)"

afxplain.sql

Sometimes it is not possible to use SQLT on an environment. For example a customer may object to installing SQLT on their production environment or they may be concerned about the performance impact on a production system.

$FND_TOP/sql/afxplain.sql can be used to produce some of the same output as SQLT.

It does not require any installation on the environment (other than the script file itself) and has a negligible performance impact. It can be run whilst the upgrade is in progress.

It provides the following information:-

- Database/Apps version, O/S version and instance/database name
- Explain Plan with cost and predicate information
- Amount of shared memory
- Object information including Package, View and Triggers
- Statistics for all objects referenced by SQL (including columns and indexes)
- Non-default Database parameters (init.ora)

Note that the output will *not* contain the following:

- Actual counts of rows on each table.
- Actual execution plan, but this should be available from the display_cursor (or display_awr).

Create a file containing the SQL the diagnostics are required for. In this example it is called query.sql.

Run:

afxplain.sql *query.sql* Y N

The Y N parameters respectively specify that a CBO Trace (10053) is required, but that statistics will not be exported.

The output will be in file *query.sql.out*

The trace for the 10053 event will be in the user dump destination.

For more information see the "SQLT" > "Alternative to SQLT (afxplain.sql)" section in My Oracle Support document "Oracle E-Business Suite Performance Guide (Document 1672174.1)"

SQLT Output using the XTRACT method

This uses a significant amount of system resources, so should not be run during the upgrade. Instead, Display Cursor Report and afxplain.sql can be used to obtain much of the information that is included in the SQLT output.

See the following My Oracle Support documents for more information:

- Oracle E-Business Suite Performance Guide (Document 1672174.1), section "SQLT"
- All About the SQLT Diagnostic Tool (Document 215187.1)

The SQLT should be provided on the same environment where the performance issue was observed, and should be run as soon after the relevant program/process as possible.

Be aware of any actions that may alter the data that SQLT is capturing (that is, actions that take place after the observed performance issue, but before SQLT is run). For example, statistics being gathered, removed, locked, imported, or data being deleted from temporary or interface tables.

To run SQL with XTRACT the sql_id or hash_value will be needed.

In Oracle Database 11g, the TKPROF will give the sql id and hash value.

For Oracle 11g. the raw trace gives the sql_id (sqlid=). In 10g and before, it gives the hash value (hv=).

Other methods of obtaining the sql_id include using AWR, Oracle Enterprise Manager, and directly querying V$SQL.

The SQLT provides execution plans, CBO statistics, database parameters, CBO parameters, performance statistics, and metadata (schema object definitions and so on) associated with the SQL.

Depending on the SQLT parameters it can also contain supporting information such as AWR reports, ASH Reports, ADDM Reports, CBO (10053) trace, SQL Monitor report, EBS Initialization Parameters healthcheck report (bde_chk_cbo), Test Case Builder and SQL Tuning Advisor.

It is particularly useful if access to the instance is not possible. Even if access is possible, it gives all the information in one place (one zip file).

The XTRACT method is needed because:

- It obtains the runtime execution plan from memory or AWR
- It is RAC aware.
- If the SQL was executed when "statistics_level" was set to ALL (or "_rowsource_execution_statistics" is set to TRUE) then it will contain actual row source statistics (I/O, buffer gets, elapsed time) on the execution plan (provided the cursor is still in memory (cursor cache).

Note that SQLT runs AWR and ASH reports. Some dictionary objects (particularly WRH$_LATCH_CHILDREN, especially if statistics_level is set to ALL) will have grown significantly during the upgrade. So, it may be necessary to gather fixed object and dictionary statistics before running SQLT.

SQLT can take quite a while to run.

To reduce the workload, it is recommended that the following are run (from SQL*Plus) before running sqltxtract.sql:

To disable Test Case Builder TCB and/or SQL Tuning Advisor

```
EXEC sqltxplain.sqlt$a.set_param('test_case_builder', 'N');
EXEC sqltxplain.sqlt$a.set_param('sta_time_limit_secs', '30');
```

To disable the automatic export of a test case repository

```
EXEC sqltxplain.sqlt$a.set_param('export_repository', 'N');
```

If SQLT still takes a long time, and the schema objects used by the SQL contain a large number of sub-partitions, the granularity of the data collected can be reduced as follows:

```
EXEC sqltxplain.sqlt$a.set_param('c_gran_segm', 'PARTITION');
EXEC sqltxplain.sqlt$a.set_param('c_gran_cols', 'PARTITION');
EXEC sqltxplain.sqlt$a.set_param('c_gran_hgrm', 'PARTITION');
```

Note that these commands can all be run as APPS. They do not need to be run as user SQLTXPLAIN

These values are stored in a table called SQLTXPLAIN.SQLI$_PARAMETER. Once they are set, they do not need to be re-set for each execution of SQLT. The current values can be checked by querying this table.

To reduce the time further the counting of rows on tables can be disabled, by running the following. However, information on the actual number of rows in each table will be lost.

```
EXEC sqltxplain.sqlt$a.set_param('count_star_threshold', '0');
```

All of this assumes that a SQLT version greater than 1.4.4.4 (April 2, 2012) is being used.

AWR Export

The AWR can be exported to a dump file (data pump export file) using $ORACLE_HOME/rdbms/admin/awrextr.sql.

The $ORACLE_HOME/rdbms/admin/awrload.sql script can be used to load the export dump into a local database, where SQL can be run on various AWR tables to get detailed analysis of where waits occur (job, SQL, object etc), and any patterns.

The key tables are DBA_HIST_ACTIVE_SESS_HISTORY, DBA_HIST_SEG_STAT_OBJ, DBA_HIST_SYSTEM_EVENT , DBA_HIST_SEG_STAT, DBA_HIST_WAITSTAT, DBA_HIST_ACTIVE_SESS_HISTORY, DBA_HIST_SQLSTAT, DBA_HIST_SQLTEXT.

Alternatively, SQL can be run directly on the source environment (i.e. without the need to run export/import).

One advantage of the export /import strategy is that diagnostics can be kept after the original environment has been refreshed.

See Oracle Database Performance Tuning Guide, Transporting Automatic Workload Repository Data.

AWR Reports

Obtain AWR reports for:

- The whole period that the upgrade is running.
- For the duration of long-running jobs (i.e. between the snapshots taken just before the job starts and just after it finishes).
- Each individual snapshot.

Typically, the snapshot interval should be 30 minutes (the default is 1 hour). This can be altered to suit the overall elapsed time of the upgrade/window. If the upgrade/window is shorter then smaller delays are significant, and a shorter snapshot may be needed. For upgrades taking 8-15 hours, a snapshot of 15 minutes would be more suitable. When running AWR reports for specific jobs the report should include the period that the job was running, without containing too much before or afterwards.

The AWR retention period should be long enough to cover the duration of the upgrade run and a significant period afterwards (to gather diagnostics and analyze). The suggestion is N+7 days, where N is the estimated upgrade time, but a longer period will provide more time to gather subsequent diagnostics and statistics.

awrrpt.sql will typically be used to generate the AWR reports. Always choose HTML report type. On an Oracle RAC instance, awrrpti.sql will usually suffice, as the upgrade will be run on one Oracle RAC node only.

AWR reports can be automated. This is useful if producing a large number of AWR reports, particularly for successive snapshots. See the "Automating AWR Reports" section in My Oracle Support document "Performance Diagnosis with Automatic Workload Repository (Document 1674086.1)".

The three main starting points to look for in AWR reports are long-running SQL (SQL Statistics /SQL Ordered By); contention/bottlenecks (Top 5 Timed Foreground Events/Foreground Wait Events); and CPU utilization. However, many other useful statistics can be obtained from AWR reports, depending on the situation. Some are covered elsewhere in this document.

If there are high levels of a particular wait(s), first check to see if it only occurs with particular long-running SQL and jobs, before assuming that it is a system configuration or resource issue.

The following can also be run:

- awrsqrpt.sql (or awrsqrpi.sql) to report on the execution plans for particular long-running SQL.
- ashrpt.sql (or ashrpti.sql) to report on the Active Session History for particular SQL. This gives useful information on the waits and events for each row source or object.

Note that some fixed objects and dictionary objects (particularly WRH$_LATCH_CHILDREN) will have grown significantly during the upgrade. This is especially the case if statistics_level = ALL, or there is a high retention period or a short snapshot interval.

So fixed object and dictionary statistics may need to be gathered before running AWRs.

See My Oracle Support document "Performance Diagnosis with Automatic Workload Repository (Document 1674086.1)" for more information.

AD Parallel tables export

The key tables to export (either using Data Pump Export or Original Export) are:

- AD_PARALLEL_UPDATES
- AD_PARALLEL_UPDATE_UNITS
- AD_TASK_TIMING

These can then be imported onto a local database using either Data Pump Import or Original Import, and then be analyzed.

The key items to look for here are the actual rows processed on each batch; the number of batches; the progress over time (to spot jobs that slow down); and any significant irregularity in rows processed/time taken for each batch that might suggest locks/sleeps or data distribution issues. Also the start times and end times of jobs and workers (AD_TASK_TIMING), which can help match long running SQL and performance issues (on AWR etc) with specific jobs.

The AD Job Timing Report (below) only reports the Top 100 Time Consuming Jobs. However, for AD Parallel jobs each worker is reported as a separate job, so if there are a high number of AD workers (for example, 16, 24 or 32), it may only report a handful of jobs. In this case, the AD_TASK_TIMING table can be queried directly to identify all the long running jobs.

AD_TASK_TIMING can also be queried to identify low worker utilization (and potential resource utilization) due to phasing waits.

See the "Useful Scripts" part of this document for report that will show all long running jobs and phasing waits.

AD Job Timing Report

This reports:

- Number of successful, failed, deferred, re-started or skipped jobs.
- The top 100 time consuming jobs.
- The failed, deferred, re-started and skipped jobs.
- The timing of each upgrade phase, with the total number of jobs, and the number deferred, re-started and skipped.

However, it only reports the Top 100 Time Consuming Jobs, and for AD Parallel jobs it considers each worker to be a different job. This means it may only report a handful of jobs, and the AD_TASK_TIMING table will need to be queried directly (see section above and "Useful Scripts" section).

When ADOP, AutoPatch or AD Administration is run, it automatically generates an AD Job Timing report (adt<session_id>.lst). The contents of this report can be accessed from Oracle Application Manager, or reports can be obtained for completed upgrade sessions from the APPL_TOP/admin/<SID>/out directory. The report is called adt<session_id>.lst.

The AD Job Timing Report can also be run for AD Administration jobs from the command line.

```
$ cd $APPL_TOP/admin/<SID>/out

$ sqlplus <APPS username>/<APPS password> @$AD_TOP/admin/sql/adtimrpt.sql \
<session id> <output file>
```

Where <session_id> is the session of the timing statistics required, and <output file> is the name of the file where the statistics will be written.

$AD_TOP/admin/sql/adtimdet.sql can also be run in a similar way. This gives details on all jobs ordered by phase or elapsed time. This is useful for finding out how long any job took to run, and also where the "Top 100 Time Consuming Jobs" is dominated by multiple workers of a few jobs.

Note that the SQL scripts may be in $AD_TOP/sql (not admin/sql).

See "Oracle E-Business Suite Maintenance Guide Release 12.2" for more information.

AD Utility and Worker Logs

All AD utilities record their processing actions and any errors that they encounter in log files. Many utilities prompt for the name of the log file, but default to <utility_name>.log. For example, for AD Administration the default log file is adadmin.log. For AutoPatch, it is adpatch.log.

AD utilities that process jobs in parallel also write details to worker log files. The adwork<number>.log files (adwork001.log, adwork002.log...) reside in the $APPL_TOP/admin/<SID>/log directory, where <SID> is the value of the ORACLE_SID or TWO_TASK variable (UNIX).

For Online Patching (ADOP) the AD Utility and Worker logs are located in the non-editioned file system (fs_ne) in the <INSTALL BASE>/fs_ne/EBSapps/log/adop directory e.g.

/u01/PROD/fs_ne/EBSapps/log/adop

See ADOP Logs and Diagnostics below.

Online Patching Enablement - Specific Diagnostics

The Online Patching Enablement patch is applied using AutoPatch (adpatch). In addition to the general diagnostics above the output from the following script will be useful during Online Patching Enablement:

$ sqlplus apps @$AD_TOP/sql/ADZDSHOWDDLS.sql

ADOP Logs and Diagnostics

All the ADOP logs are located on the non-editioned file system (fs_ne) in the <INSTALL BASE>/fs_ne/EBSapps/log/adop directory e.g.

/u01/PROD/fs_ne/EBSapps/log/adop

Each cycle of ADOP creates a subdirectory corresponding to the patch session ID, e.g.

/u01/PROD/fs_ne/EBSapps/log/adop/n

Where n is the session ID.

It is easiest and quickest to produce a zip of the entire directory.

The main files of interest are the ADOP logs (e.g. adop_YYYYMMDD_HHMISS.log).

But the adzdshowlog.out, adworker*.log, u*.log, u*.lgi, admrgpch*.log files are all useful and under the same path.

When running ADOP the on screen terminal output will mention which ADOP session ID is in use.

e.g. /u01/PROD/fs_ne/EBSapps/log/adop/9/apply_20121011_024437

The session ID directory will contain a trace file for each phase (e.g. adop_20130316_091340.log) and a corresponding log directory for each phase containing other logs (e.g. apply_20130316_091340).

The timestamp of the trace file and the corresponding log directory will match.

Non ADOP Logs

The same log directory for each phase (e.g. apply_20130316_091340) also contains some AD Utility and worker logs.

These include adrelink.log, adlibin.log, adlibout.log, adworknnn.log. The most useful are the adworknnn.log files that show the jobs run on each AD Parallel worker along with timestamps.

Online Patching Log Analyzer Utility

This is delivered in R12.AD.C.Delta.n (since R12.AD.C.Delta.4).

This utility analyzes adop log directories for errors and warnings, and displays messages to help the user quickly identify any problems that may have occurred during an adop run. It thereby offers an alternative to reviewing log files manually.

The Log Analyzer utility can be run without options:

To scan all log directories of the latest adop session for errors:

$ adopscanlog

The utility can also be run with various options. Examples include:

To scan log directories relating to the latest run of adop in the latest session:

$ adopscanlog -latest=yes

To scan log directories relating to the latest run of the specified phase, in the latest session:

$ adopscanlog -latest=yes -phase=<phase_name>

To scan all log directories of a given session (represented by a session_id) for errors:

$ adopscanlog -session_id=<number>

To see a complete list of supported parameters:

$ adopscanlog -help

adzdshowlog.out

This reports the contents of the AD_ZD_LOGS table. This contains messages on the progress of online patching with timestamps. The contents of this table will be truncated every time cleanup/prepare phase is run.

This can also be obtained for previous phases by running the following script:

$ sqlplus apps @$AD_TOP/sql/ADZDSHOWLOG.sql

Or running the SQL

```
SELECT * FROM ad_zd_logs ORDER BY log_sequence desc;
```

SQL to determine status of ADOP phases

The following SQL statement will show the status for each adop phase along with its corresponding session id.

```
SELECT adop_session_id, prepare_status, apply_status, finalize_status,
       cutover_status, cleanup_status, abort_status, status, abandon_flag, node_name
FROM ad_adop_sessions
ORDER BY adop_session_id;
```

This is effectively a history of online patching in an environment.

The following statuses apply to all phases:

Y : the phase is done

N : the phase has not been completed

X : the phase is not applicable

R : the phase is running (in progress)

F : the phase has failed

P : (is applicable only to APPLY phase) at least one patch is already applied for the session id

C : the status of this ADOP session has completed

Note: Numerical statuses are only relevant for the cutover phase... These status values are updated when a step has completed, and are as follows:

N : the phase has not been completed

0 : cutover/force_shutdown has started

1 : "force_shutdown" step has successfully executed

3 : "db_cutover" step has successfully executed

4 : "fs_cutover" step has successfully executed

6 : "force_startup" step has successfully executed

Y : the phase is done

Cutover statuses

cutover_status='Y' 'COMPLETED'

cutover_status not in ('N','Y','X') and status='F' 'FAILED'

cutover_status='0' 'CUTOVER STARTED'

cutover_status='1' 'SERVICES SHUTDOWN COMPLETED'

cutover_status='3' 'DB CUTOVER COMPLETED'

cutover_status='D' 'FLIP SNAPSHOTS COMPLETED'

cutover_status='4' 'FS CUTOVER COMPLETED'

cutover_status='5' 'ADMIN STARTUP COMPLETED'

cutover_status='6' 'SERVICES STARTUP COMPLETED'

cutover_status='N' 'NOT STARTED'

cutover_status='X' 'NOT APPLICABLE'

Check the current status of the adop cycle

Source the run filesystem environment file and run command

adop -status

Usage:

adop -status generates a summary report

adop -status <sessionID> generates a summary report for that session ID

adop -status -detail generates a detailed report

Check AD and TXK C Patch levels

Run :

```
SELECT codelevel FROM AD_TRACKABLE_ENTITIES WHERE abbreviation in ('txk','ad');
```

Long Running SQL, Contention and Tuning

For long running jobs or SQL, it is best to start by first investigating if good execution plans are being used. A poor execution plan (or even just one that is moderately sub-optimal) can be the root cause of contention, especially if that contention only occurs during a particular job.

Often a sub-optimal execution plan can cause contention and/or additional waits because it is accessing unnecessary database blocks. This might be because of a full table scan, or an index that is not particularly selective. This increases the probability of different AD workers accessing the same blocks at the same time. Once the execution plan no longer accesses those unnecessary blocks, the contention/waits may reduce or disappear.

However, to identify sub-optimal execution plans, check the actual row counts, buffer gets, and elapsed time of each execution plan step. Quite often execution plans that look good at first glance (e.g. use selective indexes) may not be close to the best. Also note that an execution plan that is quite good may still have one step where some unnecessary blocks are accessed, increasing the risk of

contention as workers or threads are added. This is why setting statistics_level = ALL / _rowsource_execution_statistics = TRUE is advised.

File Versions

When an SR or Bug is raised, Oracle Support and Development will ask for the version of the job (file) that has the issue, or the version of code (file) used by the job.

It may be necessary to check what is done in a particular job, or the exact metadata/object definition in a file. When doing this, be sure to identify the version of the files used in the upgrade.

For example, for an R12.2.n upgrade, the version present in the file system or database after all the upgrade steps – including R12.2.n RUP and post upgrade steps – may be different from the one used during the R12.2.0 upgrade.

To find out the version of a file used during the R12.2.0 upgrade, check the unified driver used for that part of the upgrade: the same principle applies to the 12.2.n RUP. (For example:

$ cat u_merged.drv|grep –A5 cstpostimportaad.sql).

A Little Contention is good

Once any unnecessary contention caused by sub-optimal execution plans has been removed, a small amount of contention (e.g. 5 to 15% on particular waits) between AD Parallel or Parallel Execution sessions can be a useful indicator that the most is being obtained from the available resources.

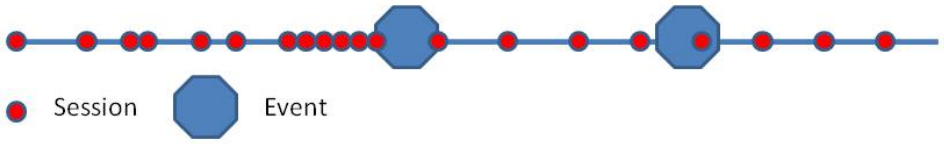
However, measuring the level of contention for some events is difficult. For typical contention waits due to sessions accessing the same block (in buffer), latch or row, the whole of the wait can be considered to be contention. But for waits on resources such as disk (e.g. db file sequential read), there is a minimum average wait time even when the resource is not busy. In such a case, look at how long the wait has increased, and the length/fatness of the tail on the histogram.

Shifting Bottlenecks

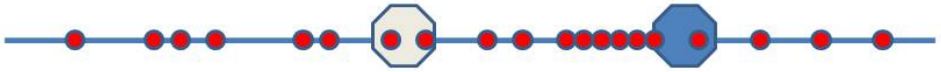
Be aware of shifting bottlenecks. Sometimes an action is taken that reduces a particular wait significantly, but another wait significantly increases. The obvious assumption is that the action has caused the new wait. However, this may not be the case: it may just have revealed the wait.

This does not mean that action should not be taken to remove the new wait. It just means that reversing the original action may not be the solution.

Here is an illustration of this effect.



Sessions arrive at first event in random pattern and they have to wait. So the first event is a bottleneck. Because the first event/bottleneck is in effect regulating the flow, there are few waits at the second event.



When the first bottleneck is removed, the sessions arrive at the second event in a random pattern and have to wait. The second event then becomes a bottleneck.

Useful Scripts

Top SQL between two snapshots in Cursor Cache or AWR

If SQL is still in memory (cursor cache) the following can be used to identify long running SQLs that may not have been written to the AWR yet (at last snapshot).

```
SELECT * FROM
(SELECT
   ss.sql_id,
   ROUND(SUM(ss.elapsed_time/1000000),0)  elapsed_time_secs,
   ROUND(SUM(ss.cpu_time/1000000),0)  cpu_time_secs,
   SUM(ss.disk_reads)  disk_reads,
   SUM(ss.direct_writes)  direct_writes,
```

```
   SUM(ss.buffer_gets)  buffer_gets,
   SUM(ss.px_servers_executions)  px_server_execs,
   SUM(ss.rows_processed)  rows_processed,
   SUM(ss.executions)  executions,
   SUM(ss.application_wait_time)  apwait_secs,
   SUM(ss.sharable_mem)  sharable_mem,
   SUM(ss.total_sharable_mem)  total_sharable_mem
 FROM v$sqlstats  ss
 GROUP BY  ss.sql_id
 ORDER BY 2  DESC)
 WHERE ROWNUM  <= 100;
```

The following SQL script will report the longest running SQLs between two AWR snapshots.

```
SELECT * FROM
(SELECT
dhs.sql_id,
ROUND(SUM(dhs.elapsed_time_delta/1000000),0) elapsed_time_secs,
ROUND(SUM(dhs.cpu_time_delta/1000000),0) cpu_time_secs,
SUM(dhs.disk_reads_delta) disk_reads,
SUM(dhs.buffer_gets_delta) buffer_gets,
SUM(dhs.px_servers_execs_delta) px_server_execs,
SUM(dhs.rows_processed_delta) rows_processed,
SUM(dhs.executions_delta) executions,
ROUND(SUM(dhs.iowait_delta/1000000),0) iowait_secs,
ROUND(SUM(dhs.clwait_delta/1000000),0) clwait_secs,
ROUND(SUM(dhs.ccwait_delta/1000000),0) ccwait_secs,
ROUND(SUM(dhs.apwait_delta/1000000),0) apwait_secs
FROM dba_hist_sqlstat dhs
,v$database d
WHERE dhs.dbid = d.dbid
AND snap_id > <begin snap> and snap_id <= <end snap>
GROUP BY dhs.sql_id
ORDER BY 2 DESC)
WHERE ROWNUM <= 100;
```

Where <begin snap> and <end snap> are the start and end snapshot IDs.

Identify when a piece of SQL ran

The following SQL will show when a particular piece of SQL ran (i.e. between which snapshots). This is useful in matching SQLs to jobs.

```
SELECT
dhs.sql_id,
dsn.snap_id,
dsn.begin_interval_time,
dsn.end_interval_time,
ROUND(SUM(dhs.elapsed_time_delta/1000000),0) elapsed_time_secs
FROM dba_hist_sqlstat dhs
,v$database d
```

```
,dba_hist_snapshot dsn
WHERE dhs.dbid = d.dbid
AND dsn.snap_id = dhs.snap_id
AND dsn.dbid = dhs.dbid
AND dsn.instance_number = dhs.instance_number
AND dhs.sql_id = '<sql_id>'
AND dsn.snap_id > <begin_snap> and dsn.snap_id <= <end_snap>
GROUP BY dhs.sql_id, dsn.snap_id, dsn.begin_interval_time, dsn.end_interval_time
ORDER BY dsn.snap_id;
```

Where <begin snap> and <end snap> are the start and end snapshot IDs.

Long Running Upgrade Jobs

Note that the "Top 100 Time Consuming Jobs" section of the standard adtimrpt.sql report lists all workers for AD Parallel jobs separately. So the top 100 can be dominated by a handful of jobs.

The following SQL can be used to list all jobs in order of maximum elapsed time (descending), but reporting all workers of an AD Parallel job in one line. It only reports completed jobs.

Note that <session_id> is the ID for the upgrade "session" and not a user session.

Be aware of jobs that are called multiple times in the same phase (e.g. akload.class, LoadMap.class, XDOLoader.class).

```
SELECT
phase,
phase_name,
product,
job_name,
max_elapsed_time,
min_start_time,
max_end_time,
workers
FROM
(SELECT
   phase,
   phase_name,
   product,
   job_name,
   MAX(elapsed_time) elapsed_time_unconv,
   LPAD(FLOOR(MAX(elapsed_time)*24), 4)||':'||
   LPAD(FLOOR((MAX(elapsed_time)*24-floor(MAX(elapsed_time)*24))*60), 2, '0')||':'||
   LPAD(MOD(ROUND(MAX(elapsed_time)*86400), 60), 2, '0')
   max_elapsed_time,
   INITCAP(TO_CHAR(MIN(start_time),' MON DD HH24:MI',
               'NLS_DATE_LANGUAGE = american')) min_start_time,
   INITCAP(TO_CHAR(MAX(end_time),' MON DD HH24:MI',
               'NLS_DATE_LANGUAGE = american')) max_end_time,
   count(worker_id) workers
```


```
FROM ad_task_timing
WHERE session_id = <session_id>
GROUP BY phase, phase_name, product, job_name)
ORDER BY elapsed_time_unconv DESC;
```

Matching long-running SQL with a job

The following variant will give jobs running at any point between two time intervals, with the longest running jobs first. This is useful in matching SQLs to jobs:

<start_time> = start of period to report in format YYYYMMDDHH24MISS

<end_time> = end of period to report in format YYYYMMDDHH24MISS

Note that the job must have completed for it to be reported by this script.

```
SELECT
phase,
phase_name,
product,
job_name,
max_elapsed_time,
min_start_time,
max_end_time,
workers
FROM
(SELECT
   phase,
   phase_name,
   product,
   job_name,
   MAX(elapsed_time) elapsed_time_unconv,
   LPAD(FLOOR(MAX(elapsed_time)*24), 4)||':'||
   LPAD(FLOOR((MAX(elapsed_time)*24-floor(MAX(elapsed_time)*24))*60), 2, '0')||':'||
   LPAD(MOD(ROUND(MAX(elapsed_time)*86400), 60), 2, '0')
   max_elapsed_time,
   INITCAP(TO_CHAR(MIN(start_time),' MON DD HH24:MI',
               'NLS_DATE_LANGUAGE = american')) min_start_time,
   INITCAP(TO_CHAR(MAX(end_time),' MON DD HH24:MI',
               'NLS_DATE_LANGUAGE = american')) max_end_time,
   count(worker_id) workers
FROM ad_task_timing
WHERE session_id = <session_id>
AND
(
start_time BETWEEN TO_DATE('<start_time>','YYYYMMDDHH24MISS')
          AND TO_DATE('<end_time>','YYYYMMDDHH24MISS')
OR
NVL(end_time, start_time+elapsed_time)
           BETWEEN TO_DATE('<start_time>','YYYYMMDDHH24MISS')
           AND TO_DATE('<end_time>','YYYYMMDDHH24MISS')
)
GROUP BY phase, phase_name, product, job_name)
ORDER BY elapsed_time_unconv DESC;
```

To find upgrade AD jobs that are in progress, use adctrl option 1 (Show worker status).

When they started can be determined by looking at the patch log file.

e.g.

```
$ cat u_merged.log|grep -A2 cstpostimportaad.sql
```

```
Assigned: file cstpostimportaad.sql on worker 48 for product bom username BOM.
Time is: Fri Mar 22 2013 22:48:54
```

Report of Jobs Running at Different Times During The Upgrade

It can be useful to know:

- Which jobs were running at a particular time: this helps identify the job that a long running SQL (in AWR) belongs to.
- The particular jobs causing events (such as contention/waits) on AWR.
- Which jobs were running at the same time as another job.
- Which jobs were running during a long period of low worker utilization. However, be aware that this may be legitimate: for example, a job that runs parallel SQL/DML, or creates objects in parallel.

The following script reports on the jobs that were running at specified intervals during the upgrade.

< session_id> = Id for the upgrade session

<interval> = The interval in minutes

<start_time> = start of period to report in format YYYYMMDDHH24MISS

<end_time> = end of period to report in format YYYYMMDDHH24MISS

This date/time format is used because it is same in all regions and has no delimiters.

-1 for <start time> and <end time> reports the whole period of upgrade session

-1 for <end time> just reports the jobs running at the start time

Note that this script does not show all jobs run in any interval - just those that were running at the sample times.

FND_TABLES is used as a dummy table to generate interval rows - this will give a maximum of around 20,000 intervals, so choose an interval size that does not result in more intervals than this.

```
BREAK ON time_slot

WITH
ad_start_end AS
(SELECT MIN(start_time) start_time, MAX(end_time) end_time
FROM ad_task_timing
WHERE session_id = <session_id>
)
,intervals AS
(SELECT (NVL(TO_DATE(DECODE(<start_time>,'-
1',NULL,<start_time>),'YYYYMMDDHH24MISS'),st.start_time) + (((rownum-1)*
TO_NUMBER(<interval>))/(24*60))) interval_time
FROM
fnd_tables ft,
ad_start_end st
WHERE rownum <= DECODE(<start_time>
                      ,'-1'
                      ,CEIL(((st.end_time-
st.start_time)*24*60)/TO_NUMBER(<interval>))+1
                      ,DECODE(<end_time>
                             ,'-1'
                             ,1
                             ,CEIL(((TO_DATE(DECODE(<end_time>,'-
1',NULL,<end_time>),'YYYYMMDDHH24MISS')
                                    -TO_DATE(DECODE(<start_time>,'-
1',NULL,<start_time>),'YYYYMMDDHH24MISS'))
                                    *24*60)
                                    /TO_NUMBER(<interval>))+1
                             )
                       )
)
SELECT
TO_CHAR(di.interval_time,'DD-MON-YYYY HH24:MI:SS') time_slot,
adt.job_name job_running,
COUNT(adt.worker_id) workers
FROM
intervals di,
ad_task_timing adt
WHERE di.interval_time BETWEEN adt.start_time
AND NVL(adt.end_time,
        DECODE(adt.elapsed_time
              ,NULL
              ,di.interval_time
              ,adt.start_time+adt.elapsed_time)
        )
AND adt.session_id = <session_id>
GROUP BY di.interval_time, job_name
ORDER BY di.interval_time, job_name;
```

Report of worker utilization

The following script can be used to report worker utilization at specific intervals during the upgrade:

<session_id> = ID for the upgrade session

<interval> = The interval in minutes

As with the previous script, FND_TABLES is used as a dummy table to generate interval rows - this will give a maximum of around 20,000 intervals, so choose an interval size that does not result in more intervals than this.

Look for long periods of time where worker utilization is much lower than the number of AD workers. However, be aware that it may be legitimate: for example, a job that runs parallel SQL/DML, or creates objects in parallel.

```
WITH
ad_start_end AS
(SELECT MIN(start_time) start_time, MAX(end_time) end_time
FROM ad_task_timing
WHERE session_id = <session_id>
)
,intervals AS
(SELECT (st.start_time + (((rownum-1)* TO_NUMBER(<interval>))/(24*60))) interval_time
FROM
fnd_tables ft,
ad_start_end st
WHERE rownum <= CEIL(((st.end_time-st.start_time)*24*60)/TO_NUMBER(<interval>))+1
)
SELECT
TO_CHAR(di.interval_time,'DD-MON-YYYY HH24:MI:SS') time_slot,
COUNT(adt.worker_id) workers,
DECODE(MIN(adt.job_name),MAX(adt.job_name),MIN(adt.job_name),'Multiple') job_running
FROM
intervals di,
ad_task_timing adt
WHERE di.interval_time BETWEEN adt.start_time
AND NVL(adt.end_time,
        DECODE(adt.elapsed_time
              ,NULL
              ,di.interval_time
              ,adt.start_time+adt.elapsed_time)
        )
AND adt.session_id = <session_id>
GROUP BY di.interval_time
ORDER BY di.interval_time;
```

Report of batch/worker utilization for a specific job

The following script can be used to report the number of batches in progress for intervals during a specific job. It also shows if the size of batches or speed of processing is increasing or decreasing.

<script_name> = script/job to be analyzed

<interval> = the interval in minutes

Again, FND_TABLES is used as a dummy table to generate interval rows - this will give a maximum of around 20,000 intervals, so choose an interval size that does not result in more intervals than this.

Note that there could be previous upgrades with the same script name. So this script may need to be edited to ensure that only data for the most recent run of the script is picked up.

Also be aware that the script_name may not be the same as the job_name from the task timing reports. For example, it may contain an appended version number. The script name is usually assigned to the variable l_script_name in the actual script.

```
WITH
ad_start_end AS
(SELECT MIN(start_date) start_time, MAX(end_date) end_time
FROM ad_parallel_update_units aduu,
     ad_parallel_updates adu
WHERE aduu.update_id = adu.update_id
AND adu.script_name = '<script_name>'
)
,intervals AS
(SELECT (st.start_time + (((rownum-1)* TO_NUMBER(<interval>))/(24*60))) interval_time
FROM
fnd_tables ft,
ad_start_end st
WHERE rownum <= CEIL(((st.end_time-st.start_time)*24*60)/TO_NUMBER(<interval>))+1
)
SELECT
adu.script_name,
TO_CHAR(di.interval_time,'DD-MON-YYYY HH24:MI:SS') time_slot,
COUNT(*) batches,
ROUND(AVG(aduu.rows_processed),0) avg_row_process,
ROUND(AVG((aduu.end_date - aduu.start_date)*(24*60*60)),0) avg_bat_tim_secs,
DECODE(SUM(aduu.rows_processed),0,0,ROUND(((SUM(aduu.end_date-
aduu.start_date))*(24*60*60)*10000)/(SUM(aduu.rows_processed)),0)) sec_per_10k_rows,
ROUND(AVG(aduu.end_block + 1 - aduu.start_block),1) avg_blocks_per_bat
FROM
intervals di,
ad_parallel_update_units aduu,
ad_parallel_updates adu
WHERE aduu.update_id = adu.update_id
AND adu.script_name = '<script_name>'
AND di.interval_time BETWEEN aduu.start_date AND NVL(aduu.end_date,di.interval_time)
GROUP BY adu.script_name, di.interval_time
ORDER BY adu.script_name, di.interval_time;
```

Report on CBO Statistics for all Oracle E-Business Suite tables

It may be necessary to report on the CBO statistics for Oracle E-Business Suite tables during the upgrade, particularly before adsstats.sql is run.

The script adsstats.sql will populate the statistics correctly before the end of the upgrade. The fact that tables may have incorrect statistics during the upgrade will not be visible. So it may not be possible to see that a table had null, zero or inaccurate CBO statistics, and that this is the reason for an expensive execution plan.

```
SELECT owner, table_name, num_rows, TO_CHAR(last_analyzed,'DD-MON-YYYY HH24:MI:SS')
last_analyzed
FROM all_tables
WHERE owner IN
(SELECT upper(oracle_username) sname
FROM   fnd_oracle_userid
WHERE  oracle_id BETWEEN 900 AND 999
AND read_only_flag = 'U'
UNION ALL
SELECT DISTINCT upper(oracle_username) sname
FROM           fnd_oracle_userid a,
fnd_product_installations b
WHERE          a.oracle_id = b.oracle_id
)
ORDER BY owner, table_name;
```

Analyze contention (waits) and where they occur

The following shows the sql_ids on which a particular wait occurs between two snapshots in AWR.

<db_id> is the database ID, <inst_num> is the instance number, <wait name> is the name of the wait, and <begin snap> and <end snap> are the start and end snapshot IDs.

```
select ss.sql_id, ss.time_waited, ss.counts_waited, tt.total_time,
ROUND((ss.time_waited*100/tt.total_time),1) percent
from
(select s.sql_id
, COUNT(*) counts_waited, SUM(time_waited) time_waited
from DBA_HIST_ACTIVE_SESS_HISTORY s, DBA_HIST_SEG_STAT_OBJ o
where s.dbid = <db_id> and s.instance_number = <inst_num>
and o.dbid (+) = s.dbid
and o.obj# (+) = s.current_obj#
and s.event = '<wait name>'
and snap_id > <begin_snap> and snap_id <= <end_snap>
group by s.sql_id) ss
,(select SUM(time_waited) total_time
from DBA_HIST_ACTIVE_SESS_HISTORY t
where t.dbid = <db_id> and t.instance_number = <inst_num>
and t.event = '<wait name>'
and t.snap_id between <begin_snap> and <end_snap>) tt
order by ss.counts_waited desc;
```

The following SQL shows the objects on which a particular wait occurs for a given SQL ID (between two snapshots in AWR).

Where <db_id> is the database ID, <inst_num> is the instance number, <wait name> is the name of the wait, <sql_id> is the SQL ID and <begin snap>, and <end snap> are the start and end snapshot IDs.

```
select ss.sql_id, ss.event, ss.current_obj#, ss.owner, ss.object_name, ss.object_type,
ss.time_waited, ss.counts_waited, tt.total_time,
ROUND((ss.time_waited*100/tt.total_time),1) percent
from
(select s.sql_id, s.event, s.current_obj#, o.owner, o.object_name, o.object_type
, COUNT(*) counts_waited, SUM(time_waited) time_waited
from DBA_HIST_ACTIVE_SESS_HISTORY s, DBA_HIST_SEG_STAT_OBJ o
where s.dbid = <db_id> and s.instance_number = <inst_num>
and s.sql_id = '<sql_id>'
and s.event = '<wait name>'
and o.dbid (+) = s.dbid
and o.obj# (+) = s.current_obj#
and snap_id between <begin_snap> and <end_snap>
group by s.sql_id, s.event, s.current_obj#, o.owner, o.object_name, o.object_type) ss
,(select SUM(time_waited) total_time
from DBA_HIST_ACTIVE_SESS_HISTORY t
where t.dbid = <db_id> and t.instance_number = <inst_num>
and t.sql_id = '<sql_id>'
and t.event = '<wait name>'
and t.snap_id > <begin_snap> and t.snap_id <= <end_snap>) tt
order by ss.counts_waited desc
```

Common Solutions

Known Issues

Once the long running jobs and SQL have been identified, check My Oracle Support for known issues and potential solutions or workarounds.

However, bear in mind that a fix or workaround may not necessarily fix the particular problem that is observed. Some bugs and SRs do not show the symptoms of the performance problem, but only the solution.

If possible, evidence (diagnostics) should be obtained to justify the suggested fix.

If it cannot be confirmed (from the diagnostics) that the issue is exactly the same then the fix may still be applied, but continue to gather diagnostics and search for solutions until the issue is fully resolved.

Many situations have been encountered where a customer has focused on a particular known issue and fix, only to find out that their performance issue was different from the known one and therefore the required fix was different.

This is because each customer uses the Oracle E-Business Suite modules and functionality in a different way, and has different volumes and distributions of data. They may also have a different database version or configuration, hardware configuration, and CBO and database initialization parameters.

Custom Indexes

There will be cases where a long-running job has an inefficient execution plan, and so a new index will have an enormous impact.

Typically, the execution plan will use a full table scan, an unselective index range scan, or an index skip scan, all of which can result in a large number of rows being filtered out when the table is accessed.

In such cases, a custom index could be created to filter out the rows, reducing the number of rows accessed on the table or the need for an index skip scan.

Ensure that an SR is still raised for the performance issue (supplying diagnostic evidence). The performance issue may still need to be resolved in the standard code.

SQL Profiles for Inefficient Execution Plans

If, when using the diagnostics above (especially display cursor report or SQL Trace), it can be identified that a long running job has an inefficient execution plan, a SQL Profile could be used to apply hints that will help the CBO choose a better execution plan. SQL tuning expertise will be needed to do this.

Ensure that an SR is still raised for the performance issue (with diagnostic evidence). The performance issue may still need to be resolved in the standard code.

The syntax for hints in SQL Profiles is stricter than for hints applied directly in the SQL. If the syntax is wrong, they are just ignored. Lower and upper case can still be used, quotes (“) omitted, and the leading query block names left out.

However, the best approach is to:

1. Run EXPLAIN PLAN on the SQL (with hints added).

2. Get the fully formed hints from the outline by running:

   ```
   SELECT * FROM TABLE(dbms_xplan.display(FORMAT => 'ALL +OUTLINE')).
   ```

   This helps get the correct syntax.

3. Create the SQL profile script and execute it.

4. Run EXPLAIN PLAN on the SQL (without hints added). This shows if the SQL Profile has been applied (the Notes section at the end of the output will contain 'SQL profile "<profile_name>" used for this statement') and the explain plan should show that the hints have been applied.

Here is an example of a script to create a SQL Profile, applying hints.

```
DECLARE

l_sql_fulltext clob := NULL;
lv_hint SYS.SQLPROF_ATTR := SYS.SQLPROF_ATTR ();

BEGIN

l_sql_fulltext := 'SELECT AAGC.APPLICATION_ID, AAGC.APPROVAL_GROUP_ID FROM
AME_APPROVAL_GROUP_CONFIG AAGC WHERE SYSDATE BETWEEN AAGC.START_DATE AND
NVL(AAGC.END_DATE - (1/86400), SYSDATE) AND NOT EXISTS (SELECT NULL FROM
AME_ACTION_USAGES AAU, AME_RULE_USAGES ARU, AME_ACTIONS AA, AME_ACTION_TYPES AAT WHERE
AA.ACTION_ID = AAU.ACTION_ID AND AAU.RULE_ID = ARU.RULE_ID AND ARU.ITEM_ID =
AAGC.APPLICATION_ID AND SYSDATE BETWEEN AA.START_DATE AND NVL(AA.END_DATE - (1/86400),
SYSDATE) AND SYSDATE BETWEEN AAT.START_DATE AND NVL(AAT.END_DATE - (1/86400), SYSDATE)
AND AA.PARAMETER = TO_CHAR(AAGC.APPROVAL_GROUP_ID) AND AAT.ACTION_TYPE_ID =
AA.ACTION_TYPE_ID AND AAT.NAME IN (''pre-chain-of-authority approvals'' , ''post-chain-
of-authority approvals'' , ''approval-group chain of authority'') AND ROWNUM < 2) ORDER
BY AAGC.APPLICATION_ID';

lv_hint.EXTEND;
lv_hint(1) := 'BEGIN_OUTLINE_DATA';
lv_hint.EXTEND;
lv_hint(2) := 'USE_NL(@"SEL$2" "AAT"@"SEL$2")';
lv_hint.EXTEND;
lv_hint(3) := 'USE_NL(@"SEL$2" "ARU"@"SEL$2")';
lv_hint.EXTEND;
lv_hint(4) := 'USE_NL(@"SEL$2" "AAU"@"SEL$2")';
lv_hint.EXTEND;
lv_hint(5) := 'LEADING(@"SEL$2" "AA"@"SEL$2" "AAU"@"SEL$2" "ARU"@"SEL$2"
"AAT"@"SEL$2")';
lv_hint.EXTEND;
lv_hint(6) := 'INDEX(@"SEL$2" "AAT"@"SEL$2" ("AME_ACTION_TYPES"."ACTION_TYPE_ID"
"AME_ACTION_TYPES"."START_DATE" "AME_ACTION_TYPES"."END_DATE"))';
lv_hint.EXTEND;
lv_hint(7) := 'INDEX(@"SEL$2" "ARU"@"SEL$2" ("AME_RULE_USAGES"."RULE_ID"
"AME_RULE_USAGES"."START_DATE" "AME_RULE_USAGES"."END_DATE"))';
lv_hint.EXTEND;
lv_hint(8) := 'INDEX(@"SEL$2" "AAU"@"SEL$2" ("AME_ACTION_USAGES"."ACTION_ID"))';
```


```
lv_hint.EXTEND;
lv_hint(9) := 'INDEX(@"SEL$2" "AA"@"SEL$2" ("AME_ACTIONS"."PARAMETER"))';
lv_hint.EXTEND;
lv_hint(10) := 'FULL(@"SEL$1" "AAGC"@"SEL$1")';
lv_hint.EXTEND;
lv_hint(11) := 'LEADING(@"SEL$1" "AAGC"@"SEL$1")';
lv_hint.EXTEND;
lv_hint(12) := 'IGNORE_OPTIM_EMBEDDED_HINTS';
lv_hint.EXTEND;
lv_hint(13) := 'END_OUTLINE_DATA';

dbms_sqltune.drop_sql_profile
(name => 'R1220_AMEMIGCFG_1'
,ignore => TRUE
);

dbms_sqltune.import_sql_profile(
sql_text => l_sql_fulltext
,category => 'DEFAULT'
,name => 'R1220_AMEMIGCFG_1'
,profile => lv_hint
,description => 'R1220 amemigcfg.sql  23y6y8d0r9v61'
,force_match => TRUE
);

END;
/
```

SQL Baselines can also be used to import an execution plan from cursor cache, AWR or SQL Tuning set. However, this does not have the ability to import hints.

In conclusion, SQL Profiles:

- Are easy to apply.
- Can be used to directly specify the join order, access and join methods.
- Are very stable.

SQL Tuning Advisor limitations.

Using SQL Tuning Advisor (STA) to resolve performance issues during the R12.2.n upgrade has the following drawbacks.

- It could delay provision of diagnostics and a subsequent solution.
- The fact that STA has to be used to get an acceptable execution plan still indicates that there is an underlying issue that needs to be resolved.

- It can only identify more efficient execution plans if the tables are populated (with representative data) at the time STA is run, or in the session in which STA is run. So it cannot be used for Global Temporary tables or Temporary/ Interim/Transitional tables.
- It can only identify more efficient execution plans if the CBO statistics are correct.
- The SQL Profiles that STA produces do not use hints to specify the actual join order, access and join methods. Instead they use the OPT_ESTIMATE hint (and usually SCALE_ROWS) to correct the cardinality over/underestimates that resulted in the "poor" execution plans. These are not necessarily stable, particularly if CBO statistics change later. SQL profiles that use hints are preferred.

Pragmatic Stable Execution Plans

In most cases, there is one join order that will give a good execution plan and minimize throwaway (that is, unnecessary access to rows that are filtered out at a later stage). Often, that join order is the natural one, following the flow of the application.

AD Parallel Jobs

AD Parallel jobs will access batches of rowids from a driving table (specified when calling AD_PARALLEL_UPDATES_PKG.INITIALIZE_ROWID_RANGE to initialize AD Parallel).

In almost all cases these batches will contain a very small percentage of the total rows. So the execution plan should lead with the driving table (accessing it by rowid), and then use nested loop join method and index access. This will give a pragmatic plan for AD Parallel jobs. It may not be the very best execution plan, but it will not be particularly inefficient.

For small tables (particularly lookup or reference), a full table scan and hash join may be better, but (compared to the overall workload) the difference could be negligible.

rowid, leading, use_nl and index hints will typically be used. The undocumented cardinality hint could also be used to specify a low cardinality for the driving table (e.g. cardinality(ai 1).

Parallel SQL

For jobs that use Parallel SQL, full table scans and hash joins will usually, but not always, be better. They will also benefit from using a join order that minimizes throwaway.

Long running SQL in Online Patching Enablement / Online Patching

There can sometimes be long running internal SQL with inefficient execution plans in Online Patching Enablement, R12.2.n RUPs and other online patches. This will show up as internal SQL (on V$ views or on SYS/SYSTEM objects) appearing high in AWR and TKPROF reports.

Gathering fixed object or dictionary statistics can help in these circumstances. Particularly on editioning objects.

If there are only a handful of internal SQLs with inefficient execution plans and only a few objects then specific objects could be targeted rather than gathering all dictionary or fixed object statistics.

Long running Cleanup in Online Patching

If cleanup (after applying AD/TXK and 12.2.n RUP patches) is taking too long (particularly "drop covered objects") then consider running quick cleanup rather than standard cleanup.

```
adop phase=cleanup cleanup_mode=quick
```

Later, when there is more time, cleanup can be run again in standard mode (which is the default):

```
adop phase=cleanup
```

Note that:

Cleanup and FS_CLONE stages are still required even if patches are being applied using ADOP in downtime mode. So this advice still applies.

If cleanup has not be run from the previous patching cycle the cleanup will run at the start of FS_CLONE. So if FS_CLONE (after applying AD/TXK and 12.2.n RUP patches) is taking too long to cleanup (particularly "drop covered objects") then quick cleanup should be run prior to FS_CLONE.

Drop covered objects is unfortunately a serial operation. The number of objects to drop depends on how many objects are affected by the patch (which will be a large number in the case of the E-Business Suite 12.2.n RUP).

Long Running Index Creation

This will typically apply to xdf and odfs.

Obtain an AWR report for the snapshots where the index is being created.

Not running with parallel DML

First of all check that the index is being created in parallel (e.g. CREATE INDEX …….. PARALLEL) when the odf file is processed by AutoPatch. The DDL command will show up in the AWR (SQL Ordered By).

If the index is being created in serial then the AutoPatch parameter parallel_index_threshold may need to be decreased.
This parameter specifies the number of blocks in a table. If a table contains fewer blocks than the threshold setting, indexes are created with parallel AD workers and serial DML (each worker creating a different index). If the table contains more blocks than the threshold setting, indexes are created with one worker and parallel DML. The valid values are 0 to 2147483647. If set to 0, indexes are created with parallel workers and serial DML. Default value: 20000; meaning a threshold of 20,000 blocks.

Otherwise the statistics for the table may need to be gathered. If the statistics are missing or incorrect (e.g. too low) then AutoPatch (adodfcmp) may get the wrong number of blocks and choose to create with serial DML.

If the index cannot be created in parallel, consider pre-creating the index – see below.

parallel_max_servers / parallel degree

Check if parallel_max_servers needs increasing.

If the SQL is not using as many parallel slaves as expected, then it is possible that the number of parallel slaves is being limited by other simultaneous tasks.

The AWR can be checked for any simultaneous tasks taking place at the same time as the create index.

Another limitation could be the CPU_COUNT * PARALLEL_THREADS_PER_CPU parameters.

CPU_COUNT, if set, should either be the number of CPU cores or zero.
Note that if CPU_COUNT is not set, or set to zero, it will default to the number of CPUs reported by the operating system. This is therefore acceptable.

PARALLEL_THREADS_PER_CPU if present should be 2. If it is not set, the default is platform-dependent (normally 2), and adequate in most cases.

Note that Automatic Parallel Degree Policy (PARALLEL_DEGREE_POLICY) should not be enabled - the default is MANUAL.

See "Oracle Database VLDB and Partitioning Guide" "How Parallel Execution Works" and "Tuning General Parameters for Parallel Execution" sections.

pga_aggregate_target

Create index will use space in the PGA for sorting. If (and only if) there is significant usage of temporary space should increasing pga_aggregate_target be considered.

Contention between parallel slaves

If the index is being created in parallel (and with enough slaves), the performance issue may be caused either by contention between the parallel slaves creating the index or contention with another simultaneous process.

Note that if the parallel degree is too high this will also cause contention.

Obtain an AWR report for the period when the index is being created and check for the contention (waits, CPU usage).

If there is no contention on AWR, it is unlikely that pre-creating the index will resolve the issue. The same command would be used to pre-create, with the same parallel degree.

Pre-Creating

If the index is being created (by odf) in serial, and either it cannot be created in parallel or there is significant contention when the index is created (by odf) in parallel, then pre-creating the index is likely to help. However, there are some points to bear in mind:

- Do not do it too soon. If the table subsequently has heavy DML (insert, delete or update of a column used in the index) prior to where the normal creation would have occurred (with odf), then that DML is likely to take much longer. So if possible, pre-create the index after any job that executes heavy DML on the table.
- Also, be careful to create the index with exactly the same definition as in the odf file. adodfcmp will compare the index that has already been pre-created with the definition in the odf file. If they do not match, adodfcmp may need to alter or re-create the index anyway.
- Remember to ALTER INDEX to revert the degree of parallel (e.g. NOPARALLEL) afterwards.

So if pre-creating indexes manually, do it as late as possible.

Long Running Index Re-creation

If the definition of a seeded E-Business Suite seeded index has been customized to change the type, columns, order (ASC or DESC), uniqueness, compression or partitioning, then it will be re-created during the upgrade to match the standard definition.

For large tables this is likely to consume significant additional time.

High level of contention on indexes on long running DML jobs

If a long running job running heavy DML (typically INSERT, but it could be UPDATE or DELETE), has a high level of contention, and that contention is on particular indexes, consider dropping the indexes before the job and re-creating them afterwards (provided that the index is not used by any execution plan in the meantime).

Alternatively, consider dropping the indexes prior to the upgrade and re-creating them afterwards. However, be sure that the indexes are not being used during the upgrade.

Check if indexes have been used during any particular period by using the command 'ALTER INDEX <index> MONITORING USAGE', and then querying the view V$OBJECT_USAGE.

In addition, the AWR table DBA_HIST_SQL_PLAN will indicate which indexes are in use, and DBA_HIST_ACTIVE_SESS_HISTORY will indicate how much time is used in maintaining the indexes for each piece of DML.

Ensure that indexes are re-created in parallel and with exactly the same definition. And remember to ALTER INDEX to revert the degree of parallel (e.g. NOPARALLEL) afterwards.

When deciding this, take into account the percentage of rows being inserted/deleted/updated during the DML, as re-creating the index will be for all rows. This will be particularly useful on custom indexes, where dropping the indexes prior to the upgrade and re-creating them afterwards is advised (provided that the custom index does not make any of the upgrade jobs significantly quicker).

High Level of "enq: HW – contention" or "enq: HV – contention"

This is related to the case above.

If a long-running job inserting into a table and indexes has a high level of waits ("enq: HW – contention" or "enq: HV – contention") then the following could be done:

- If the wait is occurring largely on particular indexes, drop the indexes before the job and re-create them afterwards (as above), provided the index is not used in the meantime.
- Increase the extent size on the table and indexes.

- Pre-allocate extents for the table and indexes.
- Partition the table and any indexes to spread the load across multiple partitions. Note that the tables and indexes will still be partitioned after go live. So only do this if the partitioning method will also give benefits after going -live on the production environment).

These "enq: HW – contention" or "enq: HV – contention" are often accompanied by higher levels of other waits such as "enq: TX – contention" or "buffer busy waits".

The HW enqueue is used to manage the allocation of space beyond the high water mark of a segment. So the wait "enq: HW – contention" typically means that sessions are waiting for the allocation of extents.

The wait "enq: HV – contention" is the same, except it occurs when parallel slaves are waiting for an extent to be allocated.

The AWR tables can be queried directly to get more detail on where waits are occurring (e.g. on which object). See the "Useful Scripts" section.

High Level of redo log waits ''log buffer space'', ''log file sync'', "log_file_switch" etc.

If there are a high level of waits associated with redo log, especially "log buffer space" and "log file sync", consider changing the configuration of redo logs; moving to a faster filer; increasing the size or number of logs; or increasing the log parallelism (hidden initialization parameter _log_parallelism_max).

Consider running with NOLOGGING, but this is not advised. It would require turning logging off for all tables and indexes and then switching it back on post-upgrade for all the tables and indexes where it had been switched off.

If getting high "log file switch (checkpoint incomplete)" waits then consider removing (resetting) the initialization parameter log_checkpoint_interval. The default is 0, which has the same effect as setting the parameter to infinity and causes the parameter to be ignored. This reduces the number of checkpoints.

Long Running Statistics Gathering (adsstats.sql)

If the upgrade is taking a long time to gather CBO statistics (adsstats.sql), consider the following strategies:

- Increasing parallel_max_servers and pga_aggregate_target (adsstats.sql should run with few or no concurrent tasks).
- Using all nodes on Oracle RAC system.
- Importing statistics gathered during test runs.

Note that the last suggestion is a strategic action that may be taken for all Release 12.2.n upgrades. So it is considered and described in more detail in the Preparation and Planning section.

Gathering or Deleting Statistics to resolve specific performance issues

Performance issues may indicate the need for statistics for specific objects to be gathered or deleted (see Resolving Performance Issues section).

New Tables

The Release 12 upgrade creates and populates many tables. For most tables the statistics are not gathered until late in the upgrade (i.e. by adsstats.sql in the last+63 phase of the R12.2.0 upgrade).

So the statistics may be empty (for new Release 12 tables), and dynamic sampling may not have resulted in a good execution plan. Alternatively, the statistics may be incorrect.

In such a case, it may be a good idea to gather statistics (using FND_STATS.GATHER_TABLE_STATS) for a specific table at a particular stage of the upgrade.

However, only gather statistics for specific tables when the following is known:

- Incorrect or missing statistics on that table is definitely causing the performance issue.
- There are no significant inserts to the table afterwards (or significant changes in the number of distinct values (NDV) on key join/filter columns or the distribution of values on columns with a histogram). In such cases, the CBO statistics will then be incorrect and could result in subsequent poor execution plans.

Remember that having no statistics can be better than incorrect statistics (as dynamic sampling will be used) and they are certainly better than zero statistics (gathered when table was not populated).

Guidance about gathering statistics can also be sought from Oracle Support.

Temporary Tables

If performance issues are due to transient or temporary tables (including Global Temporary tables), and the statistics for these tables contain unrepresentative statistics, it may be a good idea to do one of two things:

- Before the upgrade, delete the statistics using DBMS_STATS.DELETE_TABLE_STATS, lock using DBMS_STATS.LOCK_TABLE_STATS, and then exclude using FND_STATS.LOAD_XCLUD_TAB. Dynamic sampling will then be used.
- Gather representative statistics using FND_STATS.GATHER_TABLE_STATS at an appropriate point or import representative statistics using FND_STATS.RESTORE_TABLE_STATS (with CASCADE = TRUE/default). Then lock using DBMS_STATS.LOCK_TABLE_STATS and exclude using FND_STATS.LOAD_XCLUD_TAB. The representative statistics will have been previously exported using FND_STATS.BACKUP_TABLE_STATS (with CASCADE = TRUE/default).

See the "Preparation and Planning - Pre-Upgrade Environment/Activities", "Gather CBO Statistics", "Zero or Incorrect Statistics on transient /temporary tables" section.

Long-Running Compilation / Re-compilation

If scripts (e.g. adobjcmp.sql/adutlrcmp.sql) which utilizes the job queue (DB Scheduler) to compile/re-compile objects (e.g. UTL_RECOMP.RECOMP_PARALLEL) are taking a long time then do the following :

- Check that the value of initialization parameter job_queue_processes is high enough (a value equal to the number of CPU cores is normally recommended).

Long-Running Materialized View xdf/odfs

If there are long-running xdf or odf jobs creating materialized views (MV), consider cleaning up or truncating of any large MV logs.

Note that this requires complete refresh of all MVs that are dependent on the MV log. Check that the number of rows in the MV log (mlog$) is zero to confirm that all dependent MVs have been refreshed.

Long-Running Jobs that might not be needed

Check if any long-running jobs are actually needed, especially if they do not alter any data (insert, update or delete any rows).It could be that some jobs are for a module or localization that is not used or not licensed.

Oracle Support may advise that a job can be skipped, or offer a fix or workaround.

Skipping Jobs

If Oracle Support has identified that a particular job is not required for the instance, there are two options for skipping it:

- Comment the job out of the unified driver (u driver) for the install.
- Run the adctrl utility and choose hidden option 8.

High I/O waits on temporary space

If there I/O waits on temporary space ("direct path read temp" and "direct path write temp") and there is no scope to increase PGA further (or it has not resolved the issue) then consider moving the temp space to local disk (or faster disk).

Maximum AD Parallel Workers

Be aware that AutoPatch specifies a maximum number of workers. The number depending on several factors, and under certain circumstances may be less than the number of workers required. So the issue will need to be resolved.

There will be an error message similar to the following:

AD utilities can support a maximum of 999 workers. Your current database

configuration supports a maximum of Z workers. Oracle recommends that you use

between X and Y workers.

In the above message, X, Y and Z are calculated as follows:

X = 2 * cpu_count

Y = 4 * cpu_count

Z = (processes - ((total rows of v$process) + parallel_max_servers)) / 2

Where:

processes = processes db initialization parameter.

parallel_max_servers = parallel_max_servers db initialization parameter.

total rows of v$process = select count(1) from v$process

cpu_count = cpu_count db initialization parameter (or default value).

So, the maximum is basically 50% of the available "processes" (from the database initialization file), once the current number of processes (rows in v$processes) and the maximum that could be used by Parallel Query/DML have been deducted.

See My Oracle Support document "How Does Adpatch Determine The Number Of Workers To Recommend? (Document 800024.1)"

Best Practices for Minimizing Oracle E-Business Suite Release 12.2.n Upgrade Downtime
Nov 2015
Author: Jim Machin

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together