Oracle® Hyperion Financial Management, Fusion Edition

December 2010

Performance Tuning Guide

[Skip Navigation Links]

How to Use this Guide

This document contains information intended to assist with performance tuning of Financial Management. This tuning guide is intended as an aid for people responsible for the operation and maintenance of Financial Management. Readers should be very familiar with the Financial Management application, database administration, and general operating system concepts to fully leverage the guidelines in this guide.

Caution: Improper settings and configurations may prevent Financial Management from working.

This document presents general guidelines for users. Specific users' actual implementations and environments will vary widely based on business requirements, the user's Financial Management data set, network topology, and hardware usage. Therefore, users must consider how to adapt these guidelines to their own implementations.

All tuning information stated in this guide is only for orientation, every modification has to be tested and its impact should be monitored and analyzed. All test results and performance numbers are only intended as examples to illustrate tuning concepts.

Before implementing any of the tuning settings, it is recommended to carry out end to end performance testing to obtain baseline performance data for the default configurations, make incremental changes to the tuning settings and then collect performance data. Otherwise it may impair system performance.

Note: This guide is specific to Financial Management. It is not release-specific, but generally applies to the 11.1.x releases. Release-specific notes are indicated as appropriate throughout the document.

For the Oracle Enterprise Performance Management System Supported Platform Matrix for each release, click here:
http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

Top of Document

Introduction

This document is written for people who monitor performance and tune the components in an EPM/BI environment. It is assumed that readers know server administration and hardware performance tuning fundamentals, web servers, java application servers and database.

Performance Overview

Introduction to Oracle Hyperion EPM System Performance

To maximize Oracle Hyperion EPM System performance, you need to monitor, analyze, and tune all the components. This guide describes the tools that you can use to monitor performance and the techniques for optimizing the performance of Oracle Hyperion EPM System components, for example Financial Management, Essbase, Reporting and Analysis and Planning.

Performance Terminology

This guide uses the following performance terms:

Scalability

The system's ability to perform within specification under increasing user load, data load and hardware expansion.

Latency

The time between the issuing of a request and the time when the work actually begins on the request.

Think time

The time a real user pauses to think between actions.

Resource utilization

A consumption metric, for example, the percent of CPU usage.

Response time

A time metric, for example round-trip time it takes the server to deliver a Web page.

Throughput

A rate metric (requests per unit of time), for example, requests per second, bits per second. For example, if an application can handle 20 customer requests simultaneously and each request takes one second to process, this site has a potential throughput of 20 requests per second.

Understanding Key Performance Drivers

To optimize your deployment, you must understand the elements that influence performance and scalability. A factor that dictates performance is called a key performance driver. Knowing how the drivers behave in combination further enhances your ability to deploy Oracle Hyperion EPM system optimally, based on the unique requirements of each deployment.

Hardware Capacity - Factors such as number of servers, quantity and speed of processors, available RAM, network speed etc.

Technical Platforms Tuning – Fine tuning other third party software required for installing and running Oracle Hyperion EPM products; for example: relational databases, Java application servers, Web servers, Server / Client Operating System and browsers.

Business Application Design - Application design is an important factor in system performance i.e. structure, size, and use of product features in designing applications' databases, reports, Web data entry forms, calculations and consolidations.

Business Process Usage - Activities carried out by users in the normal flow of your business cycle. Business process usage has three components:

- User activity—Activities available to users for data load or data entry, database processing (consolidations, copy, clear, and so on), and reporting and analysis.
- Rate of user activity – A number of transactions executed by one user per one hour.
- User concurrency—Number of users for each activity being carried out simultaneously.

Financial Management Records and Subcubes

A **record** in Financial Management holds the data for all base periods for a given intersection of dimension members, and a **subcube** is a collection of records that all belong to the same Entity, Scenario, Year, and Value (currency). Within a subcube there are six dimensions: Account, ICP and Custom 1…Custom 4. Because the subcube is a natural unit of data for the purposes of consolidation,

data movement and processing are carried out on a subcube basis in many places in Financial Management.

Tuning Recommendations for Financial Management

Performance tuning Financial Management is a complex and iterative process. To get you started, this document includes recommendations to help you optimize your Financial Management system performance.

Note: tuning has to be done for a particular production workload. Tuning can be conducted when workload is generated by load generation tools like Oracle Application Testing Suite (ATS) or LoadRunner by HP.

This document touches on several areas that provide a quick start for performance tuning Financial Management, including:

- Tune Operating Systems parameters
- Tune HTTP Server parameters
- Tune HTTP Server Compression / Caching
- Tune Oracle Database Parameters
- Hyper-Threading / SMT Considerations

Note: while the list in each of the above stated section is a useful tool in starting your performance tuning, it is not meant to be comprehensive list of areas to tune. You must monitor and track specific performance issues within your implementation to understand where tuning can improve performance.

Top of Document

Diagnosing Performance Problems

When a performance issue arises, it is critical to first determine the cause prior to taking any corrective action. Oracle does not recommend changing performance-related parameter settings or taking other actions until an extensive analysis of the problem has been performed.

Using Monitoring Tools

Oracle strongly recommends using monitoring tools to collect performance data as part of the diagnostic process. Monitoring the Financial Management application server, web servers, database server(s) and network layers provide useful performance data.

The recommended tool to monitor the Financial Management application process performance on the Windows server side is Microsoft Performance monitoring. Steps for configuring Performance Monitor to gather the counters specific for Financial Management application can be found here: https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=953294.1

Performance counters to monitor include:

Subsystem	Counter	Guidance
Memory	Memory: Free System Page Table Entries	"Warning" when Free System Page Table Entries is less than 8,000

		"Critical" when Free System Page Table Entries is less than 5,000. On Financial Management systems if you enable /3GB without proper tuning as per recommendations stated in section "Tune Windows /3GB", you will notice 5000 PTE as initial value after the reboot.
Memory	Memory: Available Mbytes	Should be no lower than 20% to 25% of installed physical memory. In these cases, carefully monitor Paging activity.
Memory	Memory: Page Reads/sec	This counter should be below 1,000 at all times.
Processor	Processor: % Processor Time_Total	Total processor utilization should be lower than about 70% to 80%.
Processor	Processor: % Processor Time_(N)	Each processor instance should be lower than about 70% to 80% utilization.
Disk	Physical Disk: Avg. Disk sec/Transfer	Must be lower than about 25 ms. Rule of thumb: When Avg. Disk Seconds/Transfer (the disk latency counter) is significantly greater than 25 ms, the disk subsystem is unhealthy and is a bottleneck. Please note that this counter does not tell us how to fix the problem; it only indicates there is a problem.
Disk	PhysicalDisk: Average Disk Queue Length	The average should be less than the number of spindles of the disk. If a SAN is being used, ignore this counter and concentrate on the latency counters: PhysicalDisk\Average Disk sec/Read and PhysicalDisk\Average Disk sec/Write.
Disk	PhysicalDisk: Average Disk sec/Read	The average value should be below 20 ms. Spikes (maximum values) should not be higher than 50 ms.
Disk	PhysicalDisk: Average Disk sec/Write	The average value should be below 20 ms. Spikes (maximum values) should not be higher than 50 ms.
Network	Network Interface: Bytes Total/sec	For a 100-Mbps network interface card (NIC), it should be below 6–7 MB/sec.

		For a 1000-Mbps NIC, it should be below 60–70 MB/sec.
Network	Network Interface: Packets Outbound Errors	It should be zero (0) at all times.

Product Process	**Counters / Guidance**
Oracle® Hyperion Financial Management: Instance: 1. IIS Process 2. HsxServer 3. HsvDataSource	Process: % Processor Time – Process processor utilization should be lower than 90%. Process: Private Bytes - reports bytes allocated exclusively for a specific process; its value tends to rise for a leaking process. Process: Working Set - reports the shared and private bytes allocated to a process; its value tends to rise for a leaking process. Process: Page Faults /sec - reports the total number of faults (hard and soft faults) caused by a process; its value tends to rise for a leaking process. Process: Page File Bytes - reports the size of the paging file; its value tends to rise during a memory leak. Process: Handle Count - reports the number of handles that an application opened for objects it creates. Handles are used by programs to identify resources that they must access. The value of this counter tends to rise during a memory leak. Process: Virtual Bytes - The current size, in bytes, of the virtual address space the process is using. The virtual byte of the perfmon process grows at a quick rate and never releases any memory indicating memory leak in application. Process: Virtual Bytes Peak - The maximum

	size, in bytes, of virtual address space the process has used at any one time. The virtual byte of the perfmon process grows at a quick rate and never releases any memory indicating memory leak in application. Process: Pool Nonpaged Bytes - The size, in bytes, of the paged pool, an area of system memory (physical memory used by the operating system) for objects that can be written to disk when they are not being used. Note these additional counters to monitor as well: For "HsvDataSource" Private and Virtual memory rising together. This is a symptom of a memory problem caused by either Financial Management not releasing subcubes, or a memory leak caused by HFM or database drivers. HFM web site: Lock Requests HFM web site: Current Connections Active Server Pages: Requests/sec Active Server Pages: Requests Rejected Active Server Pages: Requests Queued Active Server Pages: Requests Executing

Using Remote Diagnostic Agent (RDA)

Remote Diagnostic Agent (RDA) is a set of command-line diagnostic scripts that are executed by an engine written in the Perl programming language. The data gathered provides a comprehensive picture of the environment that aids in problem diagnosis.

Running RDA can be particularly helpful in determining the size of the subcube(s) in your FM application(s). RDA is available through the My Oracle Support website. To get started, see this knowledge base article:

https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1100612.1&h=Y

Using a Reference Application

A reference application is an application used to diagnose performance issues. A reference application runs a set number of tasks that can be compared with internal timings to help determine if the customer's environment was properly tuned. Running a reference application of some kind can help identify problems. However, there is no one application that can manifest all the performance characteristics of Financial Management. It is very common to see good performance on one

application, and poor performance on another. The parameters involved include data volume, record distribution per subcube, entity structure, number of currencies, etc.

Top of Document

Tune Operating Systems Parameters

Windows Tuning Parameters

This topic describes how to tune the Windows 2003 (SP1 or later) operating system to optimize the performance of your Oracle® Hyperion EPM System. For Windows platforms, the default TCP/IP settings are usually sufficient. However, under heavy loads it may be necessary to adjust the *MaxUserPort* and *TcpTimedWaitDelay*. These parameters determine the availability of user ports requested by an application.

Parameter	Default Value	Suggested Value
TcpTimedWaitDelay This parameter controls the amount of time the OS waits to reclaim a port after an application closes a TCP connection, has a default value of 4 minutes. During a heavy users load, these limits may be exceeded resulting in an address in use: connect exception. Tip: In registry set this parameter using following: HKLM\System\CurrentControlSet\Services\Tcpip\Parameters Value: TcpTimedWaitDelay Value Type: dword Data: 30 (decimal)	240	30
MaxUserPort The number of user-accessible ephemeral ports that can be used to source outbound connections is configurable using this parameter. Tip: In registry set this parameter using following: HKLM\System\CurrentControlSet\Services\Tcpip\Parameters Value: MaxUserPort Value Type: dword Data: 65534 (decimal)	5000	65534

Tune Windows (32 bit, x86) /3GB switch

There are various performance implications when using this switch that must be considered, both application level (EPM) and kernel level (Operating System) factors. This section lists important tuning parameters for /3GB that when tuned can enhance EPM system performance and stability.

Important Note: It is only recommended to enable the /3GB switch when you have large databases and large size applications in Hyperion Financial Management.

Using this switch reduces the memory available in the following system pools:

- System Page Table Entries (PTEs)
- Nonpaged Pool
- Paged Pool

Note: Using only the /3GB switch allocates 1 GB to the kernel and 3 GB to the Usermode space. Therefore it is strongly recommended to implement the following settings in order to make sure program stability and Windows stability:

1. Using a range of memory for the /userva=xxxx switch that is within the range of 2900 to 3030. The following sample Boot.ini file demonstrates how to use the new switch to tune a server to allocate 2,900 MB of User-mode virtual memory and 1,196 MB of Kernel-mode virtual memory. This increases the available kernel space by 172 MB:

```
[boot loader]

timeout=30

default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS

[operating systems]

multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise"
/fastdetect

/NoExecute=OptOut /3GB /Userva=2900
```

2. Increase the SystemPages in registry: The setting being updated controls the allocable memory for the operational system caches, file caches among others.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management
```

Open the Data: SystemPages

Enter its new value (hex): FFFFFFFF

Results: Above tuning settings were tested on Windows 2003 Enterprise Edition (32 bit, SP2) with 8GB RAM. Testing has shown that applying the above tuning settings offers performance and stability for Hyperion Financial Management servers.

3. After applying above settings, reboot the server.

Note: You should add this setting to the Boot.ini file only per a manufacturer's recommendation.

Tip: /3GB switch applies to following EPM processes i.e. compiled with the /LARGEADDRESSAWARE switch:

Interaction of the /3GB and /PAE switches

There is often confusion between when to use the /3GB switch and when to use the /PAE switch in the BOOT.INI file. In some cases it is desirable to use both.

Caution: You need to carefully monitor "Free System Page Table Entries" and fine tune SystemsPages" as per section "Tune Windows (32 bit, x86) /3GB switch".

On a server with between 4 GB and 16 GB of RAM hosting applications that have been compiled with the "/LARGEADDRESSAWARE" switch to use more than 2 GB of RAM per process or hosting many applications (processes), each contending for limited physical memory, it would be desirable to use both the /3GB and /PAE switches. This will deliver the best performance possible for such a system (provided enough SystemPages are available for server after applying these switches).

Important: /PAE switch permits the operating system to see and make use of physical memory above and beyond 4 GB. Recent Service Packs for Windows Server 2003 Enterprise Edition now has PAE enabled by default; older versions of these operating systems may not.

Side Effects of NOT Tuning when /3GB and /PAE switches are enabled

Here are the system performance and stability issues you will encounter if you do not monitor and tune "SystemPages" when /3GB and /PAE switches are added in "booti.ini" file:

- Windows 2003 Enterprise Edition server hangs under high users loads.
- Financial Management consolidations become unresponsive under high users loads.
- Network driver related errors.

Top of Document

Tune the Web Server

Key tuning steps for the Financial Management web server and other EPM components are included in the EPM System Configurator, which is installed with the first EPM System product installed on a computer and is used to configure all products installed on the computer. For more information, see the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide*.

Tune HTTP Server Compression / Caching

Why use Web Server Compression / Caching for EPM products?

For LAN based usage of Oracle's ® Hyperion EPM applications, it is not recommended to enable compression / caching for HTTP servers. It is only recommended to enable compression / caching when EPM applications are used over the WAN and remote users are accessing from high latency locations.

- Bandwidth Savings: Enabling HTTP compression can have a dramatic improvement on the latency of responses, while improving the throughput capacity of the system. By compressing static files and dynamic application responses, it will reduce the remote (high latency) user response time.
- Improved request/response latency. Caching components (i.e. images, js and css) provide an accurate control of every cache in the proxy chain including the browser's one. Such an accurate use of the cache makes it possible to suppress the payload of the HTTP reply using the 304 status code. Minimizing round trips over the Web to revalidate cached items can make a huge difference in browser page load times.

Important Note: Some versions of hot fixes for Internet Explorer 6.0 browser have issues in handling compression (deflate / gzip) logic. For more information, read the following Microsoft KnowledgeBase articles:

http://support.microsoft.com/kb/871205

http://support.microsoft.com/kb/947195

Though Microsoft has released hot fixes to fix these issues, in some cases it does not resolve decompression issues with Internet Explorer 6.0.

Pre-requisites

Before you enable HTTP compression on the Web Server, it is strongly recommended that you review and implement the following requirements:

• In Internet Explorer 6 set your Privacy preferences to Medium (Default).

• Review all the existing IE and Firefox browser settings as outlined in the "Preparing Web Browsers" section of the Oracle Hyperion Enterprise Performance Management System Installation Start Here.

Important Note: After you have enabled compression in Internet Explorer 6.0, if you encounter JavaScript error (80020101) when accessing Workspace etc., it may be related to browser security settings i.e. if Trusted sites security level is set to Medium or EPM site is not included in Trusted sites.

• Use only following supported browsers:

Internet Explorer 6.0, Internet Explorer 7.x and Firefox 2.0.x

Tip: Once compression is enabled in the web server, it is strongly recommended to conduct a series of tests with multiple concurrent users in the test environment. During the test it is important to evaluate how much processor time is typically being used for the web server.

Note: These settings apply only to releases 9.3.1 and 11.1.1.3.

IIS 6.0 HTTP Server

This topic describes how to enable compression and caching of your Oracle® Hyperion EPM System.

How to Enable Compression

1. Specify the following additional file types to compress. Edit the HcFileExtensions metabase property (for static files) and the HcScriptFileExtensions metabase property (for dynamic files) to apply compression to file types for both deflate and gzip compressions:

• HcScriptFileExtensions set to asp,dll,exe

• HcFileExtensions set to htm,html,txt,js,css

2. As stated below, change the level of compression (in gzip, deflate) for dynamic files to 7.

Important Note: raise the dynamic compression level above seven only if you need more network bandwidth and have sufficient CPU capacity to handle the extra load.

Deflate Compression Level

File: C:\Windows\System32\inetsrv\Metabase.xml

Location: /LM/W3SVC/Filters/Compression/deflate

Value: HcDynamicCompressionLevel

Value Type: Number

Range: 0-10

Set value to 7

GZip Compression Level

File: C:\Windows\System32\inetsrv\Metabase.xml

Location: /LM/W3SVC/Filters/Compression/deflate

Value: HcDynamicCompressionLevel

Value Type: Number

Set value to 7

3. HTTP Compression should be enabled on the site level as performance may be hindered if enabled globally. Use two new metabase properties that are DoStaticCompression and DoDynamicCompression, to enable compression at the individual site (i.e. for Financial Management and Workspace).

Tip: Run the following command line options to enable compression on the EPM Workspace site level:

Path: C:\Inetpub\AdminScripts

Command: "cscript adsutil.vbs set w3svc/1/root/workspace/DoStaticCompression true"

Path: C:\Inetpub\AdminScripts

Command: "cscript adsutil.vbs set w3svc/1/root/workspace/DoDynamicCompression true"

Tip: Run the following command line options to enable compression on the

Hyperion Financial Management site level:

Path: C:\Inetpub\AdminScripts

Command: "cscript adsutil.vbs set w3svc/1/root/HFM/DoStaticCompression true"

Path: C:\Inetpub\AdminScripts

Command: "cscript adsutil.vbs set w3svc/1/root/HFM/DoDynamicCompression true"

4. Restart the IIS HTTP Server.

Note: These settings apply only to releases 9.3.1 and 11.1.1.3.

Important Note: IIS compression directory should be excluded from the antivirus software's scan list. If not excluded, antivirus scanning of IIS compression directory may result in 0-Byte file. Read MS KB article at

http://support.microsoft.com/kb/817442

How to Enable Caching

1. Enable content expiration and set the expiration to at least "30 Days".

Note: For step by step information on how to enable caching refer to IIS 6.0 Help.

2. The directories that need to have this setting enabled on are the following:

HFM

- HFM\Common\theme_tadpole\images_global
- HFM\Common\theme_tadpole\images_hfm
- HFM\Common\theme_tadpole_hc\images_global
- HFM\Common\theme_tadpole_hc\images_hfm
- HFM\Images

Workspace

- Workspace\Images
- Workspace\Img
- Workspace\Themes\Theme_Tadpole\Images_global
- Workspace\Themes\Theme_Tadpole\Images_homepage
- Workspace\Themes\Theme_Tadpole\Images_product
- Workspace\Themes\Theme_Tadpole_hc\Images_global
- Workspace\Themes\Theme_Tadpole_hc\Images_homepage
- Workspace\Themes\Theme_Tadpole_hc\Images_product
- Workspace\Wsmedia\art
- Workspace\Wsmedia\images
- Workspace\Wsmedia\publish
- Wsmedia\art
- Wsmedia\images
- Wsmedia\publish

Note: These settings apply only to releases 9.3.1 and 11.1.1.3.

Important Note: The Http.sys file does not include the Cache Control: maxage header when the Http.sys file sends an HTTP 304 response in IIS 6.0. For a fix, please read MS KB article http://support.microsoft.com/kb/931490

Performance Gain Test

A test with and without compression enabled was conducted to see the difference in CPU utilization at the same request rate. The test data below provides an indication of the impact of switching on static / dynamic content compression.

	Non-Compressed	Compressed
Requests / Sec	181	189
CPU %	34%	49%
Network Card Utilization %	20%	7%

At approximately the same request rate, dynamic / static content compression consumes more CPU. At the same time less data are sent between the client and server and the network card is utilized more efficiently.

The following table summarizes the improvement for Workspace and Financial Management transactions over a 512 Kbps line for one remote user (ran multiple iterations over 30 minutes).

Transaction Name	Before (Non-Compressed)	After (Compressed)	% Improvement
Workspace_Logon	41	21	49%
HFM_Open_Application	33	10	70%
HFM_Open_Data_Form	15	6	60%
HFM_Change_POV_Select_Account	10	8	20%
HFM_Open_Grid	16	9	44%
Workspace_Log_off	4	2	50%
Total Throughput (bytes) (Total generated network traffic for one user with multiple iterations)	3,577,482	850,979	76%
Average Network Delay (ms)	343	349	-
Packet Loss %	0	0	-

Note: Testing was done on single Financial Management web server with an Intel dual-core 3.4-GHz web server and 4 GB RAM.

Top of Document

Tuning Financial Management Applications

This topic describes how to tune the Hyperion Financial Management to optimize the performance of your Financial Management applications.

In general, running more than one Financial Management application at a time on a single application server will impact the performance of all applications. Do not attempt to run more than 3 or 4 applications at a time, even if other applications are idle, because idle Financial Management applications require database connections and CPU time to run.

Financial Management Application Server Registry Tuning

Default Memory Settings

MaxNumDataRecordsInRAM

In general MaxNumDataRecordsInRAM is the most significant setting as it decides how many records to maintain in RAM.

HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server

Value: MaxNumDataRecordsInRAM

Type: DWORD

Data: 1000000 (Decimal) (Default setting)

Important Note: When total number of records in RAM goes above this value, the FreeLRU is called to release records from memory to free up some memory for the Financial Management server. The informational message "FreeLRUCachesIfMoreRAMIsNeeded released data cubes" is recorded in the HsvEventlog.log (located on the Financial Management application server). These messages can also be viewed in the System Messages web page.

MinDataCacheSizeinMB

By setting this value at a higher number, you can reduce the number of DataCache growth attempts and hence reduce memory fragmentation.

Typically DataCache is grown on a per need basis, and it will grow 25 MB maximum at a time.

HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server

Value: MinDataCacheSizeInMB

Type: DWORD

Data: 130 (Decimal) (Default setting)

MaxDataCacheSizeinMB

The maximum amount of memory that the Financial Management application server will allocate to store the cell values and cell status. If more memory is required by system then the cell value and cell status will be paged out to disk based on the LRU logic. Workaround is to increase the cache size.

HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server

Value: MaxDataCacheSizeInMB

Type: DWORD

Data: 260 (Decimal) (Default setting)

Important Note: If performance of a consolidation operation decreases because of paging, you can increase MaxDataCacheSizeInMB to minimize paging. This value should be more than the total memory usage allowed by "MaxNumDataRecordsInRAM", so that system does not page out the cells unnecessarily to disk. Too low MaxDataCacheSizeInMB setting means that you will run out of memory to store data records and begin paging, which reduces system performance.

MaxNumCubesInTemporaryCache (0-10000, default 100)

The number of subcubes kept in memory during data retrieval (grids, WDEFs, reports, etc.). In the case of dense applications, lowering this number can alleviate high memory usage during data retrieval (WDEFs, Grids, Reports). Consolidations are not affected by this parameter.

Note: This setting applies only to releases 11.1.1.3, 9.3.1.4, and 9.2.1.2.

MaxNumCubesInRAM (100-500000, default 30000)

The number of cubes that are held in memory at any given time. Lowering this setting can alleviate high memory usage for sparse applications (i.e. applications with many entities, but relatively few records per entity). This setting affects all data operations including consolidations and data retrievals.

Note: This setting applies only to releases 11.1.2 and 9.3.3.

Financial Management 32bit Memory Settings for Larger Applications

Table below shows common recommended settings for large monthly applications:

MaxNumDataRecordsInRAM	MaxDataCacheSizeinMB
1,000,000 (default)	260 (default)
2,000,000	350
3,000,000	450

Rule of thumb: For every 1 million increase in number of records for a monthly application, we need an extra 200 MB of physical space.

Important Note: In Financial Management 32 bit, 3,000,000 is probably the maximum value for MaxNumDataRecordsInRAM that should ever be used provided /3GB switch is also enabled on the Financial Management application server (refer to section "Tune Windows /3GB switch" on how to properly tune /3GB switch performance). That would also result in 450MB of RAM for data caching. In order to set larger MaxNumDataRecordsInRAM i.e. more than 3 million and MaxDataCacheSizeinMB to more than 500 consider the Financial Management 64bit version on Windows 2003 (x64) 64bit OS.

Recommended Tuning Parameters for Dense Apps for Financial Management 32bit

Dense applications are those with few subcubes, but large number of records per subcube. It is recommended to follow below steps to arrive at memory settings for dense applications.

i. Turn on /3GB switch (refer to section "Tune Windows /3GB switch" on how to properly tune /3GB switch performance).

ii. Monitor "HsvDataSource.exe" Private / Virtual Bytes usage using "perfmon" or HsvEventlog to make sure parameters can be increased safely.

iii. Increase MaxNumDataRecordsInRAM value to 2 or 3 million.

iv. Set MaxDataCacheSizeinMB to 450 to 600 MB.

v. Set NumMinutesBeforeCheckingLRU to 5.

vi. Set NumCubesLoadedBeforeCheckingLRU to 100 or lower, depending on density. Important Note: On Financial Management 32bit, to handle large subcubes, you need to set NumMinutesBeforeCheckingLRU to 5 and NumCubesLoadedBeforeCheckingLRU to 50.

Caution: By lowering the setting for NumMinutesBeforeCheckingLRU to 5 the Financial Management app server will check the cache more often, thereby allowing less number of subcubes to be cached and also adding extra overhead of freeing the cache and increasing round trips to database for fetching the records may degrade overall response times for reports. Also note by freeing the cache more often, Financial Management will generate more database activity.

Important Note: In Financial Management 32 bit version for handling large subcubes we have to lower the value for NumMinutesBeforeCheckingLRU setting because using default values will result in less frequent cache flushes and you may run into the risk of memory allocation problems under heavy concurrent load.

Financial Management 64bit Memory Settings for Larger Applications

The following table contains suggested values for parameters depending on available physical memory. This is done with the assumption that Financial Management is the only memory-intensive process running on the machine and running only a single Financial Management application.

Important Note: If multiple Financial Management applications will be active, then divide the total physical memory installed on the server by the number of Financial Management applications to arrive at the "Available Physical Memory" for each application.

Available Physical Memory	NumDataRecordsinRAM	MaxDataCacheSizeinMB
4 GB	4,000,000	500
8 GB	10,000,000	1500
16 GB	30,000,000	4500
32 GB	60,000,000	9000

Registry Settings for Multiple Server Tasks

Several registry settings that control the amount of consolidations that can run at any one time in a multi-server environment. If you need to run several concurrent consolidations for example, you can change the values in these registry settings.

NumConsolidationThreads

Controls the multi-threading of consolidations per application server.

Minimum value is 1; maximum value is 8.

Important Note: Lowering the value limits the system's utilization of system resources, resulting in slower consolidation performance. Raising the value results in higher CPU utilization and may affect other components' performance.

Tip: Tests conducted to evaluate the impact of increasing NumConsolidationThreads from 4 to 8. Results below show that one month consolidation times became faster.

Transactions	**92 users NumConsolidationThreads = 4 Average Response Time (seconds)**	**92 users NumConsolidationThreads = 8 Average Response Time (seconds)**
01_Run_Consolidation_ A_1105	10.11	3.22
02_Run_Consolidation_A_0005	16.15	9.47
03_Run_Consolidation_ A_2205	7.75	3.19
04_Run_Consolidation_A_3305	18.67	9.17
05_Run_Consolidation_B_1105	8.21	3.14
06_Run_Consolidation_B_0005	13.26	9.27
07_Run_Consolidation_B_2205	7.69	6.20
08_Run_Consolidation_B_3305	18.29	9.41
09_Run_Consolidation_C_0005	30.59	22.08

Note: CPU utilization on Financial Management application servers during test execution was acceptable with capacity available to accommodate more intense workload.

Important Note: Before increasing this value, ensure all registry settings must be same to all application servers; you should test to see how many current consolidations will run on a given server before total consolidation time is actually worse when running concurrent consolidations vs. consolidations waiting in queue.

MaxNumConcurrentConsolidations

Controls the number of concurrent consolidations allowed per application server. Any consolidations executed above the value are queued as Scheduled Consolidations. Minimum value is 1; maximum value is 8. Default value is 4.

Scenario: If you have three Financial Management application servers, each server could run 8 maximum numbers of concurrent consolidations but the default value of NumConsolidationsAllowed will limit you to running only 8 concurrent consolidations total on the three servers against one application. Example, users submit six consolidations on ServerA then users submit two more consolidations on ServerB – these 8 consolidations will all run. At the same time if users submit the next consolidation on ServerC, it will not run until one of the previous 8 finish (in "Running tasks" web page it will have a status of Scheduled Start).

Important Notes: Before increasing this value, ensure all registry settings must be same to all application servers; you should test to see how many current consolidations will run on a given server before total consolidation time is actually worse when running concurrent consolidations vs. consolidations waiting in queue.

NumConsolidationsAllowed

Controls the number of consolidations allowed per application across all the application servers. Default value is 8 and the range is 1–20.

Note: If you increase NumConsolidationsAllowed to 20 then you will have up to 20 consolidations spread among the Financial Management application servers before getting a Scheduled Start consolidation.

If MaxNumConcurrentConsolidations is set to 8, when users submit more than 8 consolidations on a single Financial Management application server, the 9th and beyond consolidation will also be in a scheduled start status.

Important: Before increasing this value, ensure all registry settings must be same to all application servers; you should test to see how many current consolidations will run on a given server before total consolidation time is actually worse when running concurrent consolidations vs. consolidations waiting in queue.

In releases 9.2.1.1, 9.3.1.2, 11.1.1.0 and later, there are no cluster-wide limitations on number of running consolidations, but the limitation of MaxNumConcurrentConsolidations (by default 8 consols per server) is still in effect.

Changing a Server Registry Setting

To change a server registry setting:

1. Select Start, and then Run.
2. In the Open box, type regedt32, and click OK. Select
3. HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server
4. Double-click the registry setting that you want to change, change the value, and click OK.

Database Tuning

The following housekeeping practices are recommended when using Oracle / SQL Server databases with Financial Management:

For Financial Management tables <appname>_DATA_AUDIT, <appname>_TASK_AUDIT and HFM_ERRORLOG, it is recommended to implement the following housekeeping best practices:

Quarterly - Business to review the Audit logs, archive and delete.

Half-Yearly - Archive System Messages and truncate table.

Tip: Put alerts in place so that action can be taken if these tables grow beyond the recommended number of records (> 500,000). Note large audit tables will have severe impact on Financial Management performance.

Basic Design Considerations

1. If Data Audit feature is not part of business requirements then it is recommended to turn off auditing of data. It has been observed degradation in performance for Financial Management application with Data Audit table more than 10GB.

Tip: To turn off auditing of members, change the EnableDataAudit attribute to N for all members in your metadata file.

2. Rules must always be tested prior to loading in a production environment to avoid any pitfalls of poorly designed rules (may cause data explosion from rules). So efficient rules are critical for acceptable system performance.

3. Hyperion Financial Management does all processing through subcubes while they are stored in RAM, so the larger the subcube, the bigger the hit on performance. Try to minimize subcube size as much as possible i.e. no subcube should exceed the 200,000 base record limits in order to ensure optimal system performance.

4. Loading to or calculating zeros in a Financial Management application is not recommended. Zeros are stored as data, which increases the database size and can affect performance. It is recommended that only numeric information, such as 1000, be stored in Financial Management.

5. If Financial Management is not properly shut down, temporary files may remain upon reboot. To ensure optimal performance, it is recommended that you delete all *.db.* file names from the Financial Management Server Working folder before launching Financial Management.

6. For attaching multiple documents to Data Grid / Process Unit, Oracle recommends that you attach no more than three documents to a cell. Each document should be smaller than 100K to limit the performance effect on the database.

Tip: You can set a size limit for document attachments and a maximum number of document attachments by user when you create an application. You can set the limits in the AppSettings attribute for the application metadata.

7. Make use of Consolidate All option only under appropriate circumstances. If this option is used the system does not skip entities with NODATA, which can have a significant impact on consolidation performance. Tip: The Consolidate (Impacted Consolidation) is the most efficient option, because only entities that require logic or consolidation are updated by the system. The Consolidate All with Data option is useful for updating system status from OK SC to OK after metadata changes.

8. For Lifecycle Management (LCM), if you are running multiple Financial Management LCM migrations on large applications and experience an out of memory exception in IIS web server then open IIS and change the following settings for HFMLCMservice:

1. Right-click on HFMLCMService Application pool and open the Properties page.

2. Select the Recycling tab and under Memory recycling, set these values:

Maximum virtual memory (in megabytes): 1000

Maximum used memory (in megabytes): 800

3. Click the Health tab and change the Shutdown time limit to 10800 seconds (3 hours).

4. Click on Apply, then OK to close the Properties page.

5. Perform IIS reset.

9. Oracle provides an index utility that is designed to inspect the indexes on the database tables for most versions of the Oracle/Hyperion Financial Management software. The utility is designed to examine the indexes on the database tables and compare them against the required indexes to see if any changes should be made.

Important Note: The utility can be used after a database migration, database restore or some other tasks where the validity of the indexes may be in question. The utility can also be used to generate indexes creation scripts for a DBA to use to migrate the indexes to separate tablespaces to improve Financial Management's database performance.

The index update utility operates in three modes. The first is to examine the existing database table indexes on an Financial Management application and generate a report outline any potential problems that may exist. The second option will generate a report and a file of SQL commands to drop and recreate any possibly missing or incorrect indexes. The third form generates a report and instantly executes the commands to drop/recreate indexes.

Top of Document

Application-specific Settings

Some settings in the registry that previously were only environment-level settings have been expanded to an application level. To use the Financial Management application-specific (Per App) settings, use this procedure:

For each Financial Management application server on which you want to use the application-specific settings:

1. Open the MS Windows Registry Editor tool `(regedit.exe)`.
2. Navigate to the following key:

```
HKEY_LOCAL_MACHINE\Software\Hyperion Solutions\Hyperion Financial Management\Server
```

3. Create a new sub-key named with the application prefix (for example, Comma).
4. In the new sub-key, create each of the values that you want to override.

Note: The order of precedence is as follows:

1. If an application-specific setting does not exist and an installation registry setting does, the installation registry setting is used.
2. If an application-specific setting does not exist, the setting defined in the Server key is used.
3. If no application-specific or server setting is defined, the default value is used. See the *Financial Management Administrator's Guide* for settings and default values.

These settings can be overridden with application-specific settings:

Memory Settings (6)	Connection Settings (2)	Thread Settings (2)
NumMinutesBeforeCheckingLRU NumCubesLoadedBeforeCheckingLRU MaxNumCubesInRAM MaxNumDataRecordsInRAM MinDataCacheSizeInMB MaxDataCacheSizeInMB	NumMaxDBConnections SQLCommandTimeout	NumConsolidationThreads NumVBScriptEngines

Application Server Clusters

Financial Management allows for the grouping of one or more application servers behind a "friendly" cluster name. Advantages of using an HFM cluster over an explicit HFM application server by name is that the HFM cluster allows you to add and remove servers without end users having to make any changes. When more than one HFM application server is added to an HFM cluster, new user sessions randomly pick one of the HFM application servers to connect.

For more information, see this Knowledge Base article on My Oracle Support:

https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1254300.1

Top of Document

Tuning Oracle 11g Databases

Introduction

Financial Management requires a relational database to store application data. Each Financial Management application contains a set of tables, indexes, stored procedures, and other objects. Because the number and sizes of these objects vary according to the user's data set, application design, and reporting requirements, it is difficult to specify a concrete set of rules for setting up the database. This document addresses the two most common issues that arise during deployment to Oracle databases:

- Oracle DB running out of memory to support the required number of database connections
- Poor performance during reporting and consolidation

Both of these issues can be traced to improper Oracle parameter settings and configurations. Creating a System Global Area (SGA) that is too large limits the amount of free physical memory to support user connections and activities. Conversely, creating an SGA that is too small, causes additional disk access, and slows down performance.

This document guides you through the process of monitoring a typical database and determining the proper initialization settings to maximize performance. You should repeat this process periodically to keep up with changes to your data set, workload, and application design.

Oracle 11g has made the process of monitoring and tuning the database much easier than previous versions. We strongly recommend using Oracle Enterprise Manager (both Data Control and Grid Control) to monitor, diagnose, and tune database performance. To obtain accurate instance statistics, we recommend users enable Oracle database Automatic Maintenance Tasks.

It is very important that tuning not be done immediately after database startup. At that point the buffer caches are empty and no statistics have been collected. Always test and tune your database after a period of activity on Financial Management applications.

Tuning Guidelines for Oracle 11g Databases

Oracle Initialization Parameters

Many initialization parameters can be fine-tuned to improve database performance. We will concentrate on those parameters that are known to greatly influence Financial Management performance with Oracle database.

CURSOR_SHARING

CURSOR_SHARING determines what kind of SQL statements can share the same cursors. If this parameter is set to EXACT, only statements with identical text can share the same cursor. If this parameter is set to FORCE, statements that differ in some literals but are otherwise identical can share and reuse SQL cursors, unless the literals affect the meaning of the statement. Tests show that setting this parameter to a value of FORCE may improve Financial Management consolidation and reporting performance significantly. This is because with this parameter set to FORCE, Oracle database spends less time parsing SQL statements, and requires less memory.

Suggested setting: FORCE

MEMORY_TARGET

MEMORY_TARGET and MEMORY_MAX_TARGET below are two new parameters in Oracle database release 11g. These two parameters determine the use of Automatic Memory Management for Oracle database. Oracle strongly recommends the use of Automatic Memory Management to manage the memory on your system. By setting these two parameters to non-zero values, Oracle enables Automatic Memory Management, and tunes to the target memory size, redistributing memory as needed between system global area (SGA) and the instance program global area (PGA). As a result, the following parameters will be automatically sized:

SGA_TARGET

SGA_MAX_SIZE

DB_CACHE_SIZE

SHARED_POOL_SIZE

LARGE_POOL_SIZE

JAVA_POOL_SIZE

STREAMS_POOL_SIZE

PGA_AGGREGATE_TARGET

However, LOG_BUFFER is not affected by Automatic Memory Management, and still needs to be manually sized. We will show how to tune LOG_BUFFER later.

Since MEMORY_TARGET specifies the total memory size of SGA and PGA, it should be set to a relatively high value to achieve better performance. Financial Management consolidation and reporting are memory intensive tasks, and hence require abundant memory. Considering 32-bit operating systems have limits on available address space (typically 2 GB or 3 GB), we recommend this parameter to be set to at least 1.2 GB. Generally, higher values are associated with better Oracle database performance, so we would like to set this parameter as high as possible without running out of virtual address space.

Suggested setting: minimum 1.2GB, generally higher than 1.2GB (depending on the environment)

Please review the section "Calculate Number of Processes for Oracle Database Release 11g" when determining the amount of memory to allocate to the Oracle 11g instance.

MEMORY_MAX_TARGET

MEMORY_MAX_TARGET specifies the maximum value to which a DBA can set the MEMORY_TARGET parameter. It serves as an upper limit so MEMORY_TARGET cannot be set too high accidentally. It also reserves memory for the Oracle database instance, in case you want to increase

MEMORY_TARGET at runtime without a restart. Therefore, MEMORY_MAX_TARGET should be no lower than MEMORY_TARGET.

Suggested setting: no lower than MEMORY_TAGET

SGA_TARGET

SGA_TARGET specifies the total size of all SGA components. If Automatic Memory Management is enabled, and SGA_TARGET is set to a non-zero value, then this value serves as the minimum size of SGA.

Suggested setting: 0 if Automatic Memory Management is enabled; otherwise refer to "Determining Memory Settings for Oracle Database Release 11g"

SGA_MAX_SIZE

SGA_MAX_SIZE specifies the maximum size of the SGA for the lifetime of the instance. This parameter sets the upper limit for SGA_TARGET. If Automatic Memory Management is enabled, Oracle database cannot increase the total size of SGA components beyond SGA_MAX_SIZE.

Suggested setting: Leave it at default if Automatic Memory Management is enabled; otherwise refer to "Determine Memory Settings for Oracle Database Release 11g"

PGA_AGGREGATE_TARGET

PGA_AGGREGATE_TARGET specifies the total PGA memory available to all server processes attached to the instance. If Automatic Memory Management is enabled, and PGA_AGGREGATE_TARGET is set to a non-zero value, then this value serves as the minimum size of PGA.

Suggested setting: 0 if Automatic Memory Management is enabled; otherwise refer to "Determine Memory Settings for Oracle Database Release 11g".

LOG_BUFFER

LOG_BUFFER specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. Financial Management is a high-update transactional system, and the database constantly uses log buffer. Properly sizing log buffer can improve the database performance. In general, larger values for LOG_BUFFER reduce redo log file I/O, particularly if transactions are long or numerous. If the buffer is too small, the system waits until the buffer is clear before adding new updates, so it is important to set this buffer correctly to improve database performance.

Suggested setting: Start with 8MB. Refer to "Determine Memory Settings for Oracle Database Release 11g."

OPTIMIZER_MODE

OPTIMIZER_MODE establishes the default behavior for choosing an optimization approach for the instance. You can set the optimizer mode to FIRST_ROWS for optimal online application response, or to ALL_ROWS to minimize total execution time for batch operations. Because Financial Management deals only with the total returned data set, minimizing the total execution time is more appropriate.

Suggested setting: ALL_ROWS

OPTIMIZER_INDEX_COST_ADJ

OPTIMIZER_INDEX_COST_ADJ lets you tune optimizer behavior for access path selection to be more or less index friendly; i.e. to make the optimizer more or less prone to selecting an index access path over a full table scan. The default for this parameter is 100 percent, at which the optimizer evaluates index access paths at the regular cost. But Financial Management transactions generally favor index access paths more than full table scan paths. So we recommend a lower value for this parameter.

Suggested setting: 50

PROCESSES

PROCESSES specifies the maximum number of operating system user processes that can simultaneously connect to Oracle database. Because Financial Management only works with dedicated servers, each connection requires a process. Each Financial Management application requires a minimum of 20 database connections in addition to the number specified by the Financial Management database connection pool setting. For example, if the Financial Management database connection pool setting is 40, then each Financial Management application on each application server requires at least 60 Oracle database connections.

Suggested setting: Refer to "Calculate Number of Processes for Oracle Database Release 11g".

SESSIONS

This parameter specifies the maximum number of sessions that can be created on the database system. Because every login requires a session, this parameter effectively determines the maximum number of concurrent users on the Oracle database. The default value is 1.1 * PROCESSES + 5. We do not recommend setting this parameter below its default value.

TRANSACTIONS

TRANSACTIONS specifies the maximum number of concurrent transactions. Because some transactions may be recursive, this parameter should be greater than SESSIONS, and in turn, PROCESSES, to allow for recursive transactions. The default value is 1.1 * SESSIONS. We do not recommend setting this parameter below its default value.

OPEN_CURSORS

OPEN_CURSORS specifies the maximum number of open cursors (handles to private SQL areas) a session can have at once. It is important to set the value of OPEN_CURSORS high enough to prevent your application from running out of open cursors. Assuming that a session does not open the number of cursors specified by OPEN_CURSORS, there is no added overhead to setting this value higher than actually needed.

Suggested setting: 5000

SESSION_CACHED_CURSORS

SESSION_CACHED_CURSORS specifies the number of session cursors to cache. Repeated parse calls of the same SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Subsequent parse calls will find the cursor in the cache, and do not need to reopen the cursor. Performance of Financial Management applications will benefit from this cache because Financial Management connections are also cached.

Suggested setting: 50

TRACE_ENABLED

TRACE_ENABLED controls tracing of the execution history, or code path, of Oracle database. Enabling this option by setting the parameter to TRUE puts additional overhead on the database and is not recommended for a normal Financial Management application environment.

Suggested setting: FALSE

STATISTICS_LEVEL

STATISTICS_LEVEL specifies the level of statistics collection for database and operating system. The Oracle Database collects these statistics for a variety of purposes, including making self-management decisions. The default setting of TYPICAL ensures collection of all major statistics required for database self-management functionality, and provides the best overall performance.

Suggested setting: TYPICAL

TIMED_STATISTICS

TIMED_STATISTICS specifies whether statistics related to time are collected. Starting with Oracle database release 11.1.0.7.0, the value of the TIMED_STATISTICS parameter cannot be set to FALSE if the value of STATISTICS_LEVEL is set to TYPICAL or ALL .

Suggested setting: TRUE

TIMED_OS_STATISTICS

TIMED_OS_STATISTICS specifies (in seconds) the interval at which Oracle collects operating system statistics when a request is made from the client to the server or when a request completes. Enabling this option by setting the parameter to a number greater than 0 severely degrades the performance of Financial Management applications.

Suggested setting: 0

How to Determine Memory Settings for Oracle Database Release 11g

This section outlines how to monitor and view Oracle system-related statistics, and tune Oracle database memory parameters. There are many ways to determine optimal memory settings, but the preferred way is to use memory advisors, including Memory Advisor, SGA Advisor, Shared Pool Advisor, Buffer Cache Advisor, and PGA Advisor. You must have an Oracle login with DBA privileges to use these advisors and to perform the following tasks. Notice that most of the queries below have equivalent graphical interfaces through Oracle Enterprise Manager.

Total Memory Sizing (MEMORY_TARGET)

MEMORY_TARGET specifies the Oracle system-wide usable memory, including both SGA and PGA. Prior to Oracle database release 11g, SGA and PGA had to be tuned separately.

If a database is upgraded from Oracle 10g to 11g, MEMORY_TARGET can be determined by simply adding SGA_TARGET and PGA_AGGREGATE_TARGET from the Oracle 10g database.

If a database is upgraded from Oracle 9i to 11g, MEMORY_TARGET can be determined by adding PGA_AGGREGATE_TARGET and all SGA components, including DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, JAVA_POOL_SIZE, etc.

If a database has not been upgraded from an earlier version, and has no history references, we recommend this parameter to be initially set to 1 to 3 GB, depending on system resources and system limits. After the database has been in use for some time, this parameter can be tuned as follows (This tuning also applies to the above two upgrade scenarios):

```
SQL> select * from v$memory_target_advice order by memory_size;

MEMORY_SIZE MEMORY_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR    VERSION
----------- ------------------ ------------ -------------------- ----------
        180                 .5          458                1.344          0
        270                .75          367               1.0761          0
        360                  1          341                    1          0
        450               1.25          335                .9817          0
        540                1.5          335                .9817          0
        630               1.75          335                .9817          0
        720                  2          335                .9817          0
```

The row with the MEMORY_SIZE_FACTOR of 1 shows the current size of memory, as set by the MEMORY_TARGET initialization parameter, and the amount of DB time required to complete the current workload. In previous and subsequent rows, the results show a number of alternative MEMORY_TARGET sizes. For each alternative size, the database shows the size factor (the multiple of the current size), and the estimated DB time to complete the current workload if the MEMORY_TARGET parameter were changed to the alternative size. Notice that for a total memory size smaller than the current MEMORY_TARGET size (360 in this example), estimated DB time (ESTD_DB_TIME) increases. Notice also that in this example, there is nothing to be gained by increasing total memory size beyond 450MB, because the ESTD_DB_TIME value is not decreasing. So, in this example, the suggested MEMORY_TARGET size is 450MB.

SGA Sizing (SGA_TARGET)

Normally, SGA is tuned automatically by Oracle database if Automatic Memory Management is enabled. But a DBA can still monitor the size of SGA and find out if it is at the optimal size.

```
SQL> select * from v$sga_target_advice order by sga_size;

SGA_SIZE   SGA_SIZE_FACTOR ESTD_DB_TIME ESTD_DB_TIME_FACTOR ESTD_PHYSICAL_READS
---------- --------------- ------------ ------------------- ------------------
       290              .5       448176              1.6578            1636103
       435             .75       339336              1.2552            1636103
       580               1       201866                   1             513881
       725            1.25       201866                   1             513881
       870             1.5       201866                   1             513881
      1015            1.75       201866                   1             513881
      1160               2       201866                   1             513881
```

Based on the similar analysis to above, we can see the current setting of SGA_TARGET is already optimal.

PGA Sizing (PGA_AGGREGATE_TARGET)

Like SGA, PGA is also automatically tuned by Oracle database if Automatic Memory Management is enabled. The following query can be used to monitor if PGA size is properly sized. The result will be shown similar to the query results of v$memory_target_size and v$sga_target_size.

```
SQL>  select * from v$pga_target_advice order by pga_target_for_estimate;
```

LOG_BUFFER Sizing

In the system view v$sysstat, the value for redo buffer allocation retries reflects the number of times a user process waits for space in the redo log buffer. This value should be near zero for a properly sized database. For example:

```
select name, value
from v$sysstat
where name = 'redo buffer allocation retries'
```

```
NAME                                                  VALUE
----------------------------------------------------  -------------

redo buffer allocation retries          1021967
```

If the log buffer has no room for updates, the database must wait and retry. In this example, the database has retried a total of 1,021,967 times. To improve performance, increase the LOG_BUFFER parameter value. This value is expressed in bytes and must be a multiple of the "log block size" value, which is the operating system block size. For a Financial Management application, set LOG_BUFFER to a minimum of 8MB, and then use the above query to monitor the performance and increase as necessary. If LOG_BUFFER needs to be increased, we recommend growing it by 50% at a time.

How to Calculate the Number of Processes for Oracle Database Release 11g

The number of user processes that can simultaneously connect to Oracle database is limited by the PROCESSES initialization parameter of the Oracle database instance. For an Financial Management application, the value of this parameter depends on the connection pool size parameter of that application. Each Financial Management application requires a minimum of 20 database connections, plus the Financial Management database connection pool setting value. For example, if the Financial Management database connection pool setting is 40, then each Financial Management application requires at least 60 Oracle database connections. As a result, the value of PROCESSES should be set higher than 60.

Note: The total number of servers in an Financial Management cluster, and the total number of applications, affect the number of required database connections.

Let us use an example to illustrate how to calculate the number of processes that will connect to the Oracle database. Suppose a cluster of two Financial Management application servers has two applications on each server. The Financial Management database connection pool setting is 40. The minimum number of Oracle database connections is calculated as the sum of the connection pool setting plus 20, times the number of applications, times the number of application servers: (40 + 20) * 2 * 2 = 240. For additional safety, multiply this number by a factor of 1.1 to allow for ancillary connections and general usage of the database. Considering that Oracle database also has some background processes, we add 20 to this number to arrive at the value of PROCESSES. So, in this case, PROCESSES should be set to 260.

In general,

PROCESSES = (Financial Management connection pool setting + 20) * (Number of Financial Management applications) * (Number of Financial Management application servers) * 1.1 + 20.

Other Considerations

Shared Server vs. Dedicated Server

Financial Management requires all connections to Oracle database to be served by dedicated server processes. In other words, Financial Management does not work with shared server processes. Dedicated server processes consume more CPU and memory resources but achieve better performance. To use dedicated server, the net service name value should include the SERVER=DEDICATED clause in the connect descriptor. Here is an example of net service configured for dedicated server processes:

```
HFMDB =

  (DESCRIPTION =

    (ADDRESS = (PROTOCOL = TCP)(HOST = hfm.oracle.com)(PORT = 1521))

    (CONNECT_DATA =
```

```
      (SERVER = DEDICATED)
      (SERVICE_NAME = HFMDB1)
    )
  )
```

Online Redo Log Files Size

The size of the redo log files can influence performance, because the behaviors of the database writer and archiver processes depend on the redo log sizes. Generally, larger redo log files provide better performance. Undersized log files increase checkpoint activity and reduce performance. However, small log files and frequent checkpoints reduce recovery time. So, if daily operational efficiency is more important than minimizing recovery time, then set online redo log files to a relatively large value. Hundreds of MB is a normal size for Financial Management databases. But the preferred way to determine the size of redo log files is to enable FAST_START_MTTR_TARGET, and run a typical database workload for a while. Then run the following query to get the optimal size of redo logs.

```
SQL> select optimal_logfile_size from v$instance_recovery;
```

For more details about how to tune MTTR target and the size of online redo log files, please refer to the *Oracle Database Performance Tuning Guide*.

Tablespaces and Segments Fragmentations

Over time, updates and deletes on objects within a tablespace can create pockets of empty space that individually are not large enough to be reused for new data. This type of empty space is referred to as fragmented free space. Objects with fragmented free space can result in much wasted space, and can impact database performance. Financial Management consolidation performs extensive updates, inserts, and deletes, so it is very important to monitor the fragmentation of tablespaces and defragment them regularly. The preferred way to defragment and reclaim this space is to perform an online segment shrink. For more information about how to use online segment, please refer to *Oracle Database Administrator's Guide* or consult Oracle database support services.

Index Fragmentations

Financial Management applications usually create hundreds or even thousands of indexes. As application data changes over time, indexes could become fragmented. Regular monitoring and defragmentation of these indexes can improve Financial Management performance. However, index rebuild is a time-consuming and resource-intensive operation. We do not recommend any index rebuild while Financial Management applications are in operation. Oracle Enterprise Manager provides user-friendly interfaces to monitor the statistics of indexes. For more details about how to monitor and defragment indexes through Oracle Enterprise Manager, please refer to Oracle database documentation.

Regular Maintenance and Tuning Plans

The preceding sections outline the typical process to correctly size the Oracle memory parameters. Performance tuning, by its nature, is iterative. Removing one performance bottleneck might not lead to immediate performance improvement because another bottleneck might be revealed. Therefore, this process should be repeated until the performance is acceptable. Because Financial Management application data changes constantly from period to period, regular database maintenance and tuning plans will help users proactively monitor and tune Oracle database performance, and hence prevent potential performance issues in the future. For more information and additional tuning options, consult the Oracle database support.

Top of Document

64-Bit Environments

What is 64-bit Financial Management?

This is the 64-bit port of Financial Management. It is functionally identical to 32-bit Financial Management and uses the same code base. The first version of 64-bit Financial Management is 11.x.

To maximize efficiency and flexibility, the Financial Management Subcube Engine employs memory-intensive, "calc on the fly" algorithms. To achieve their best performance, these algorithms require that the data for subcubes reside in memory as much as possible. In this paper, we will discuss the memory requirements of Financial Management's Subcube Engine and the performance boost we can expect from migrating to a 64-bit environment.

The Financial Management Application Server executes the consolidation and aggregation logic, and the heart of the server, the "Subcube Engine," is responsible for storing and indexing the individual records.

32-bit vs. 64-bit Environments

In a 32-bit environment, each process has access to a flat 4GB virtual address space, out of which a certain portion is reserved by the operating system. For MS Windows, the reserved portion is 2GB by default, though this can be reduced to 1GB, giving the user process (in this case, Financial Management) access to a maximum of 3GBs of address space. Note that a virtual address space is not the same as physical memory; instead, the virtual address space is the means through which a process accesses physical memory. The operating system is responsible for mapping these virtual addresses to physical memory.

Given the fact that at most 3GB of virtual address space is available to a process, we can see that no more than 3GB of physical memory can be directly accessed by a single process, even if the computer is equipped with more than 3GBs of memory. In reality, not all of this 3GB can be used. This is because the algorithms used by the OS to allocate memory for a process are not fully efficient, and leave gaps between blocks of memory.

The promise of 64-bit processors and operating systems is the removal of such limitations, so that processes can take better advantage of available memory. By increasing the address space from 4GB (2^{32}) to a theoretical 16 Exabytes (2^{64}, a 4 billion-fold increase), the address space bottleneck is removed.

Financial Management's Memory Requirements

Financial Management's Subcube Engine achieves its best performance when all the subcubes required during a given query reside in memory. The amount of memory required to process a given query is the "memory footprint" of that query. We can generalize this concept to a group of queries (consolidations, reports, grid views, etc.) and speak of the memory footprint of a given usage scenario. As long as the memory footprint of a query or group of queries fits within the 32-bit address space, Financial Management's server can hold all necessary records in memory, and maximum efficiency is guaranteed.

In a 32-bit environment, when the memory footprint exceeds available address space, Financial Management will offload the least recently used records. Depending on Financial Management's memory settings, these records are either paged out to disk, or completely discarded. In the former case, the records will be swapped back into memory once they are needed. In the latter, the records will be fetched from the database if the need arises. Since both disk and network/database transfer speeds are slower than memory speeds, a performance penalty is incurred in either case. It is in these cases that the 64-bit address space can help performance.

Who Benefits?

If the memory footprint consistently exceeds the 2GB or 3GB limit, we should expect to see a benefit from the 64-bit version of Financial Management. In this section we will analyze the situation further.

First, we should note that the memory footprint is a dynamic concept that depends not only on how much data is contained in the application, but also on how much of that data is in use at any given time. In other words, memory footprint depends on the level of concurrency, and therefore on the number of users accessing the application.

As an example, consider an application with data in 10 scenarios and 5 years, and let us assume that each scenario/year combination contains 100 MB of data. Now, if we assume that no more than 10 scenario/year combinations are accessed at any one time, the memory footprint is 1 GB and the application can live comfortably within the 32-bit address space. But, if 40 scenario/year combinations are accessed concurrently, the memory footprint increases to 4GB, well beyond the limits of the 32-bit address space.

Although this example may not be completely realistic, it serves to illustrate an important point: In considering the memory requirements of a given application, we should take into account not just the total data in the application, but what portion of that data is in use. So an application that holds millions of records of historical data that are not used routinely may not benefit from the 64-bit version of Financial Management. The same consideration applies to other distributions of data, for example, across years or entities.

So the question is, what types of Financial Management applications typically have memory footprints of more than 2GB or 3GB? One category consists of applications with data in many scenario/year combinations. We can further subdivide this category according to the distribution of records among subcubes. Since the paging and flushing of records in Financial Management is done on a subcube basis, applications with very large subcubes have more granularity during page-out, and as a consequence can show slower performance once the memory footprint increases.

Another category of applications that benefit from the larger 64-bit address space are weekly applications. As noted before, a record in Financial Management holds the data for all the base periods. As a result, the memory footprint of weekly applications is more than four times larger than the memory footprint of monthly applications.

Since applications with large memory footprints shift their memory requirements to disk, and especially database, these applications can put an undue burden on the network and the relational database server. An important benefit of the 64-bit version of Financial Management is the reduction of such loads, allowing Financial Management to coexist better with other applications in the customer's environment.

What Aspects of Performance Are not Affected?

It should be evident by now that the main impact of the 64-bit architecture is on availability of memory. Any aspect of performance that is purely CPU-bound, or exclusively dependent on the performance and responsiveness of the network or relational database, will not be affected.

Let us consider an example to clarify this. Assume that a user wishes to look at a grid in Financial Management where the base data for the cells consists of 200,000 records in 10 subcubes. As a result of this request, the following steps are carried out by Financial Management:

1. Data for the 10 subcubes is requested from the database (assuming the subcubes are not resident in memory).

2. Data is retrieved by the database and transmitted over the network to Financial Management.

3. The records are indexed on-the-fly by Financial Management server, and stored in the subcube engine.

4. Aggregation is performed in Financial Management server as needed and the resulting cells are transmitted to Financial Management Web layer.

5. Financial Management client formats the cells as a grid and displays them to the user.

Of these, Step 2 is dependent on database and network; Steps 3 and 4 are CPU-bound in the application server, and Step 5 depends on the performance of the Web Server and browser. So, how can the 64-bit port help with performance? The answer is in the reuse of the data once it has been stored in memory (Step 3). The price for loading the records for the first time needs to be paid in any case. If enough memory is available to hold all 10 subcubes throughout the processing stages, there will be no difference between the 32-bit and 64-bit versions. However, if one or more of the subcubes need to be offloaded, the performance delay will put the 32-bit version at a disadvantage. This is especially true if these same subcubes need to be accessed later, either by the same or other users. The extra memory available to the 64-bit version will allow more users to access more data concurrently. In this way, Financial Management server's core operations, for example, aggregation, consolidation, etc., will be run as fast as the processor permits. The time taken by these operations, disregarding memory access, can be reduced by using faster processors.

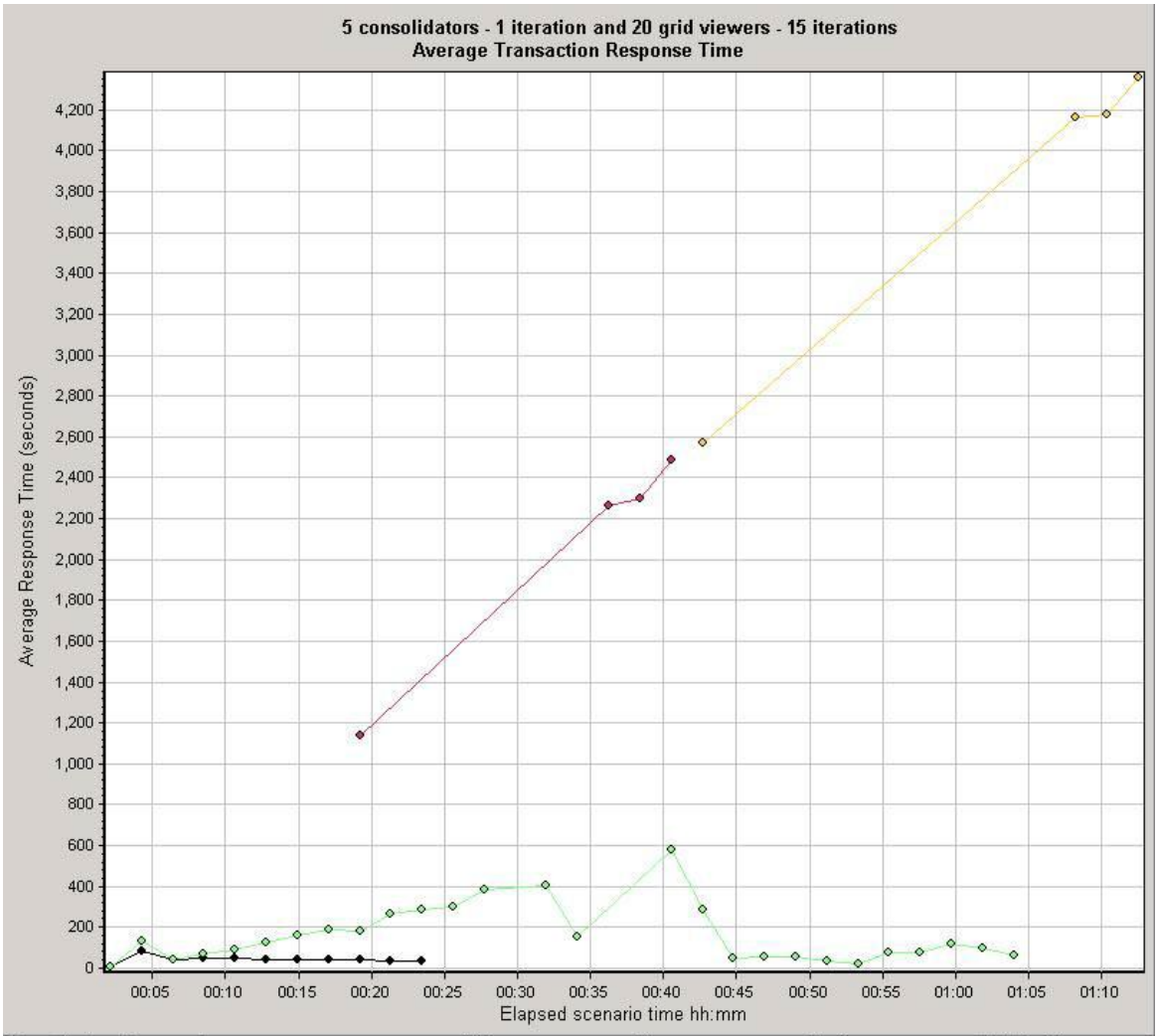
Case Studies

To demonstrate the concepts discussed in this paper, we have carried out performance comparisons between 32-bit and 64-bit versions of Financial Management. The tests consist of a mix of consolidations and grid views (data retrieval). The consolidations and grid views were chosen to be non-overlapping. The 32-bit version was given a cache size of 2M or 4M records and the /3GB switch of Windows was used. These parameters represent the outer limits of feasibility for 32-bit Financial Management. The 64-bit Financial Management was run with a cache size of 10M or 15M records, sizes that are impossible to achieve within the 32-bit address space.

Actual customer applications were used in the test, each representing a class of applications that benefit most from the 64-bit version.

Case Study 1

Application 1 is a large application with ~2500 entities with many large subcubes. The number of base-level records per year in all subcubes for the Actual scenario and years 2003-2007 was between 2 to 6 million. The total number of records accessed in 5 simultaneous consolidations was 13.6 million records. The 64-bit version was run with a cache size of 15M and the 32-bit version used a 4M-record cache. This latter parameter is twice the usual size used at customer sites, and was chosen to ensure that the 32-bit version has all the advantages it can get.



Color	Scale	Measurement	Minimum	Average	Maximum	Std. Deviation
	1	Consolidation: 64bit	1135.682	2097.811	2489.827	487.623
	1	Consolidation: 32bit	2573.187	3888.571	4358.93	661.853
	1	Viewing Grid: 64bit	5.809	47.209	153.886	25.163
	1	Viewing Grid: 32bit	9.088	111.633	746.474	131.848

The graph demonstrates the benefits of the larger address space available to 64-bit Financial Management. Both consolidation and grid views show great improvements. The percentage improvement is larger for grids compared to consolidations (57% vs. 46% for average response times; 79% vs. 43% for maximum response times). This is because consolidation times are heavily dependent on the performance of components external to Financial Management, namely the database and rules engine. The core cause of slowness of the 32-bit Financial Management is contention for the same limited memory space among all the tasks running concurrently on the same machine. To improve performance, we would traditionally depend on the tasks being run on several machines in a cluster. With the 64-bit version, we can now run more tasks on the same machine (given sufficient CPU power) and thereby reduce the number of machines necessary to handle the same workload.

Case Study 2

We used a large weekly application for this test. The test scenario was the same as the previous case: 5 consolidation users, 1 iteration, and 20 grid viewers - 20 iterations. The number of base-level records per year in this application is between 1 and 2 million. The number of entities is around 1100. The total number of records accessed during the test was around 4.5 million records. Because each

record in a weekly application is more than 4 times wider than that of a monthly application, the figure of 4.5 million records is the equivalent of a monthly application with 19.5 million records.

This time, we compared the performance of the two versions with cache sizes of 4M and 10M records. Please note that the 4M cache size is not a realistic figure for the 32-bit version, and it is used here to demonstrate the limits of 32-bit Financial Management.

First, a comparison of 32-bit and 64-bit versions with the same 4M record cache size demonstrates the two versions perform similarly, with the 64-bit version having a slight edge.

	Average Response Times (seconds) **4m MaxRecords**		
	32 Bit	**64 Bit**	**% Diff**
Grid View	160	143	-10.6%
Consols	5550	5359	-3.4%

Once we go beyond the limits of the 32-bit architecture, we can see how utilizing the larger address space can push Financial Management's 64Bit performance well beyond what can be achieved by the 32-bit version.

	Average Response Times (seconds) **32-bit 4m vs. 64-bit 10m MaxRecords**		
	4m	**15m**	**% Diff**
Grid View	160	38	-76.3%
Consols	5550	1016	-81.7%

Case Study 3

This test was run against a monthly application with ~500 entities. The total number of records accessed during the test was ~8.5 million. Thus, this application is smaller than the ones used in the two previous tests, however, the smaller number of entitles yielded a much denser application. We used a more realistic cache size of 2M records for 32-bit Financial Management.

We can see that despite the smaller size, this application still benefits from the 64-bit Financial Management. The improvement is more visible in grid views, because there is more cache reuse in data retrieval patterns associated with the read-only nature of grid access.

	Average Response Times (seconds) 32-bit 2m vs. 64-bit 10m MaxRecords		
	2m	**10m**	**% Diff**
Grid View	12.9	9	-30.2%
Consols	5252	4752	-9.5%

Case Study 4

This test was run with the same application as Case Study 1. This time we increased the number of concurrent consolidations to 10. The total number of records accessed in these consolidations was 25 million. The 64-bit version was run with a cache size of 30M and the 32-bit version used a 2M-record cache. This latter parameter is what is generally used at customer sites. The total memory footprint of the test was 7 GB.

Because of the large memory footprint required for this particular test, the 64-bit version is outperforming 32-bit Financial Management by a factor of 9.5 to 14 times. In a real-life situation, this same load would be handled acceptably by multiple application servers in a cluster. What this implies is that 64-bit Financial Management can either handle the workload with fewer application servers (and therefore lower TCO), or scale beyond what was possible with 32-bit Financial Management given the same number of servers.

Individual Consolidations – Broken down by EACH Consolidation:

			32 Bit	64 Bit	
Transaction	Scenario	Year	Duration	Duration	Variance
Consolidation	budget	2003	00:04:00.657	00:03:30.566	-12.50%
Consolidation	actual	2003	00:12:01.360	00:11:01.239	-8.33%
Consolidation	budget	2004	00:04:00.646	00:03:30.511	-12.52%
Consolidation	actual	2004	00:15:31.654	00:12:31.492	-19.34%
Consolidation	budget	2005	00:10:31.127	00:10:01.273	-4.73%
Consolidation	actual	2005	00:18:01.871	00:14:01.615	-22.21%
Consolidation	budget	2006	00:16:01.718	00:14:31.707	-9.36%
Consolidation	actual	2006	00:18:01.807	00:14:01.663	-22.20%
Consolidation	budget	2007	00:19:01.916	00:16:31.848	-13.14%
Consolidation	actual	2007	00:17:31.825	00:14:01.160	-20.03%

Grids and Consolidations Concurrently – 20 Users:

	32 Bit	64 Bit	
Transaction Name	Average	Average	Variance
Initial Web Grid Opening after Opening of Application	212.36	22.43	-89.44%
Changing Year in Web Grid *	248.27	26.26	-89.42%
Consolidation from the Top of the Entity Hierarchy	12,961.79	953.16	-92.65%

*change_year includes the re-drawing of the grid

Case Study 5

To demonstrate a case where the 64-bit version does not show dramatic speed improvements, we also ran a test with 10 concurrent users without running concurrent consolidations. The test was designed to keep the memory footprint under 2 GB.

Grids – 10 Concurrent Users:

	32 Bit	64 bit	
Transaction Name	Average	Average	Variance
e_open_LAF_grid_initial_open_grid	16.715	18.004	7.71%
f_open_LAF_grid_change_year	17.054	18.565	8.86%

* change_year includes the re-drawing of the grid

The 32-bit version can hold all the necessary data for this workload in memory. The slight speed degradation for the 64-bit version is mostly within the margin of error for the test. In general, for workloads that can be handled adequately within the 32-bit address space, the performance of the two versions should be within a few percentage points. The 32-bit version will have a slight edge due to the higher overhead of managing larger memory structures in the 64-bit version.

Summary

The Financial Management Subcube Engine employs memory-intensive, "calc on the fly" algorithms to maximize efficiency and flexibility, but Financial Management's performance was always somewhat constrained by the amount of available memory allocated by 32-bit Windows operating system. The promise of 64-bit processors and operating systems is the removal of such limitations, so that processes can take better advantage of available memory. These tests prove that once the 32-bit memory constraints are eliminated by moving to 64-bit operating system, significant performance improvements can be expected. Furthermore, 64-bit Financial Management can either handle the workload with fewer application servers (and therefore lower TCO), or scale beyond what was possible with 32-bit Financial Management given the same number of servers.

Frequently Asked Questions

Which Operating Systems are supported?

The x86-64 versions of Microsoft Windows are supported. On the application server, this includes Windows 2003 and later. On the client side, the x86-64 versions of Windows XP and Vista are supported.

Which CPUs are supported?

The x86-64 architecture is supported. This includes the AMD64 and Intel 64 (formerly EM64T) processors. The Itanium processor (IA-64 architecture) is currently NOT supported.

How does one migrate from 32-bit to 64-bit Financial Management? Will 64-bit Financial Management work with an application created under 32-bit Financial Management?

The upgrade path from a previous release running 32-bit to 64-bit Financial Management is similar to an ordinary version upgrade. When upgrading from a previous release of Financial Management, use the Schema Upgrade Utility to upgrade the database schema to support the new version of Financial Management (11.x or newer), schema upgrade is not required when moving from 11.x or newer 32-bit to 64-bit of the same release.

Which components of the system need to be 64-bit? In particular, does the relational database need to be 64-bit?

The database can be either 32-bit or 64-bit as long as it is a supported DBMS type and version.

Financial Management consists of tiers: Application Server, Database Server, Web Server, remote client. The tiers can be mixed and matched as required. 32-bit clients and web servers can connect to a 64-bit application server (and vice-versa) just as the 64-bit application server can connect to either a 32-bit or 64-bit DBMS.

Considerations need to be made for third-party and extending software. By default, the Financial Management installation software will only install 64-bit software on a 64-bit operating system (this will be addressed in future releases). This means that only the 64-bit client components will be installed on a Financial Management application server. When 32-bit connectivity is required (as is required by Financial Data Management and other systems), they may not function unless the 32-bit client software is installed on the 64-bit application server. Please refer to the Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide for more information. If this is an issue, contact the Support group for steps to resolve this.

What are the benefits of 64-bit Financial Management?

The main benefit of 64-bit Financial Management is the ability to hold substantially more data in memory at one time. Depending on the size of the application, and its usage profile, the extra memory can lead to significant speed improvements, while simultaneously reducing the load on the relational database.

What are the memory limitations of 64-bit Financial Management?

The limit of virtual address space is dictated by MS Windows and is 8 TB (8192 GB), compared to a maximum of 3 GB in 32-bit Windows. Since the physical memory in almost any current computer is far smaller than 8 TB, the implication is that, in practical terms, 64-bit Financial Management is limited by physical memory, rather than virtual memory. In other words, 64-bit Financial Management can take advantage of all available physical memory once the proper memory parameter adjustments are made (see below).

Are there any memory settings that need to be tuned for 64-bit Financial Management?

Yes. Financial Management's default memory settings are appropriate for a small to medium size application in a 32-bit environment. To take advantage of the extra memory in a 64-bit environment we recommend the following settings for a monthly application. The relevant registry settings areMaxNumDataRecordsInRAM and MaxDataCacheSizeinMB which need to be created or changed in [HKEY_LOCAL_MACHINE\SOFTWARE\Hyperion Solutions\Hyperion Financial Management\Server] on each application server's Windows registry. The following table contains suggested values for these parameters depending on available memory. This is done with the assumption that Financial

Management is the only memory-intensive process running on the machine and running only a single Financial Management application. If multiple Financial Management applications will be active, then divide the Total physical Memory installed on the server by the number of Financial Management applications to arrive at the "Available Physical Memory" for each application.

Available Physical Memory	NumDataRecordsinRAM	MaxDataCacheSizeinMB
4 GB	4,000,000	500
8 GB	10,000,000	1500
16 GB	30,000,000	4500
32 GB	60,000,000	9000

Example: On a server with 24 GB of RAM with 2 active monthly Financial Management applications, the MaxNumDataRecordsInRAM value should be 22,500,000 and the MaxDataCacheSizeinMB value should be 3375.

For a weekly application, divide the MaxNumDataRecordsInRAM by 4, without changing the value in the last column.

What kind of applications will see the most benefit?

Applications with large memory requirements will see the most benefit. This includes applications with one or more of the following characteristics:

- Large scenarios (millions of records per year),
- Dense applications (many large subcubes)
- Large memory footprint as a result of many scenarios being accessed concurrently
- Weekly applications

What kind of applications will see the least benefit?

Small applications, where the total memory footprint of the application even under load can fit comfortably in the 32-bit memory space.

Top of Document

ORACLE
ENTERPRISE PERFORMANCE MANAGEMENT SYSTEM