ORACLE

CMG'11

Performance Assurance for Packaged Applications

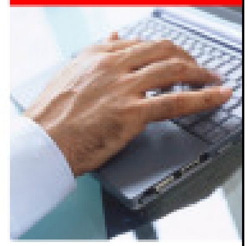
Alexander Podelko
alex.podelko@oracle.com
@apodelko

Paper 1127
Session 622

December 9, 2011

Performance is a critical factor for success of any packaged application implementations. The presentation discusses performance assurance for packaged applications on example of Oracle Enterprise Performance Management. While details are related to this particular set of applications, many approaches discussed would be applicable to most packaged applications. The presentation will discuss a holistic performance assurance approach, top-down approach to performance troubleshooting, potential performance issues and ways to address them.

Agenda



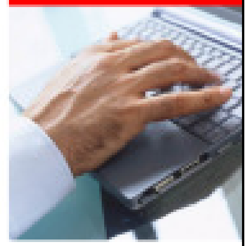
- **Introduction**
- Performance Assurance
- Performance Troubleshooting
- Examples

Oracle Enterprise Performance Management

- An integrated suite of applications
- The integration is good, so from the end user point of view it may be difficult to say what components and datasources are used
  - And especially for those not quite familiar with EPM like administrators or performance testers
- A good example of packaged business applications to discuss performance assurance and troubleshooting

Oracle Enterprise Performance Management (EPM) System includes a suite of performance management applications, a suite of business intelligence (BI) applications, a common foundation of BI tools and services, and a variety of datasources – all integrated using Oracle Fusion Middleware.

Agenda



- Introductions
- **Performance Assurance**
- Performance Troubleshooting
- Examples

Performance Assurance

- EPM products are thoroughly tested, but they may be used very differently
  - Think about Oracle Database: It is still possible to create a slow database in spite of the fact that the Oracle Database software is highly optimized for performance.
- Performance Assurance
  - Ongoing performance risk mitigation during the whole system lifecycle

ORACLE

Performance Assurance for EPM is ongoing performance risk mitigation during the whole system lifecycle. EPM products are thoroughly tested for performance, but performance of specific implementations depends on how they are designed and constructed (metadata, data, forms, grids, rules, etc.- all these artifacts are different for each implementation).

Performance Assurance Steps

- Define performance requirements
  - Number of users, concurrency, what they do
- Design your applications according to best practices
- Verify performance along the way
  - Single user with monitoring provides a lot of information
  - Use realistic volume of data
- Do necessary tuning / configuration

The steps listed are just an outline, some steps will be discussed in more details later in this presentation.

Performance Assurance Steps - Continued

- Do performance testing
  - Closely monitor the system in the process
  - If results are not satisfactory, go back to tuning or design
- Adjust configuration based on performance testing results [and re-test]
- Monitor the system in production
  - Check if the pattern is the same as in testing
  - Check trends
- Do performance testing of major changes and re-designs

ORACLE

The main point that all these activities should continue through the whole system lifecycle and the same performance metrics should be tracked through all steps.

Performance Requirements

- Workload
  - Number of concurrent users
    - From existing systems
    - Percentage of named users
  - What users will do
    - Throughput (how many reports, requests, etc)
    - What components they use
- Performance Metrics
  - Response times
  - Resources utilization

ORACLE

Performance requirements are supposed to be tracked from the system inception through the whole system lifecycle including design, development, testing, operations, and maintenance. However different groups of people are involved on each stage using their own vision, terminology, metrics, and tools that makes the subject confusing when going into details.

Throughput is the rate at which incoming requests are completed. Throughput defines the load on the system and is measured in operations per time period. It may be the number of transactions per second or the number of reports per hour. In most cases we are interested in a steady mode when the number of incoming requests would be equal to the number of processed requests.

The number of users doesn’t, by itself, define throughput. Without defining what each user is doing and how intensely (i.e. throughput for one user), the number of users doesn’t make much sense as a measure of load. What users do also defines what components and how intensely they use.

Application Design

- Application design impacts performance drastically
- Adhere to best practices to ensure application performance
  - Should be applicable to your context
  - You don't need to follow them blindly, but it would be better to have reasons why you need to deviate and do a Proof of Concept to check how it will perform
  - First of all, reasonable number of dimensions, number of members, size of forms, depth of hierarchy, etc.

ORACLE

For example, both very deep member hierarchies and flat member hierarchies may cause issues under load.

See documentation and best practices documents for details for specific applications.

Verify Performance Along the Way

- If you see a performance issue with one user, it probably would be much worse for multiple users
  - Still may be exceptions related to latency or caching
  - Tuning is not usually beneficial for single-user issues
    - Except, for example, very large objects
  - Hardware upgrade is usually not beneficial for single-user issues
    - Except, for example, cpu speed
  - It means that single-user results in the development environment may be representative even if hardware used is significantly less powerful
  - If one user uses a lot of resources, it won't be nice for multiple users

ORACLE

Very large objects (web forms, reports) may require some tuning, like increasing JVM heap size, even for one user.

Hardware upgrade (with exception of cpu speed) is usually not beneficial for single-user issues– assuming that there is no inherent issues with hardware configuration like memory is so small that it starts paging even with one user.

Tuning

- If you have more than a dozen of concurrent users you may need to do some tuning
  - Some defaults are chosen to conserve resources
  - Increase max Java heap size for heavily used components
  - Increase Essbase index and data cache sizes
- It is not recommended to do all existing tuning recommendation without understanding what they mean and testing them under multi-user load
  - Many recommendations are for specific cases only and may even degrade performance otherwise.

ORACLE

Multiple tuning documents are available and should be checked for details. For example:

Essbase Database Administrator Guide, Optimizing Essbase

Hyperion Financial Management (HFM) Performance Tuning Guide, Fusion Edition (Doc ID 1083460.1)

Time Considerations

- If any long-running, resources-consuming tasks are needed, schedule them for the time of minimal load
  - Such as large report books, consolidations, calculations
- EPM activities usually tied to the financial cycle
  - As a part of closing quarter, year, etc.
  - Heavy activity during some period and low during others
- Some EPM activities depends on others
  - Workload mix changes depending on the place in the financial cycle
  - May be several different workload profiles

ORACLE

In cases of any long-running, resources-consuming tasks it may be more efficient just to schedule them for the time of minimal load instead of trying to tune and optimize them to run in parallel with high-concurrency load.

Performance Testing

- EPM products are tested for performance
  - But it doesn't guarantee performance of a specific application
    - Every application is different
- Performance testing of *your* application is a way to alleviate performance risk
  - Highly recommended for large installations
- Performance testing of EPM products is complex
  - If done improperly, may easily led to wrong conclusions
  - Make sure that everything is correlated/parameterized properly if not using Oracle consulting

ORACLE

It is impossible to predict performance of your application without at least some performance testing.

Creating Realistic Load

- Important part of performance testing is creating *realistic* load
- The number of virtual users
- What users do
- Different user names, different POV, etc (script parameterization)

ORACLE

Running multiple users hitting the same set of data (with same Point of View, POV) is an easy way to get misleading results. If it is for reporting, the data could be completely cached and we get much better results than in production. If it is, for example, for web data entry forms, it could cause concurrency issues and we get much worse results than in production. So scripts should be parameterized (fixed or recorded data should be replaced with values from a list of possible choices) so that each user uses a proper set of data. The term “proper” here means different enough to avoid problems with caching and concurrency, which is specific for the system, data, and test requirements.

Scripting Challenges

- Multiple variables to be correlated
  - Including SSO token, repository token, etc.
  - Specific for every component
- Load testing tools may not report errors when correlation or parameterization is incorrect, but the system behavior would be unpredictable
  - Use other ways to check if the script works, such as checking for specific server response, application logs, system state.

Unfortunately, a lack of error messages during a load test does not mean that the system worked correctly. A very important part of load testing is workload verification. We should be sure that the applied workload is doing what it is supposed to do and that all errors are caught and logged. It can be done directly by analyzing server responses or, in cases when this is impossible, indirectly. For example, by analyzing the application log or database for the existence of particular entries.

Sizing and Capacity Planning

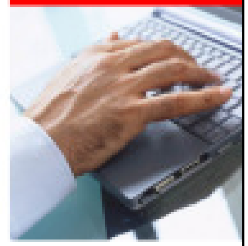
- The *Installation Start Here* document provides typical configurations for some products for 100, 500, and 1000 users with 35% concurrency
- Many difficult to formalize factors
  - Use Oracle services
- Benchmarking documents are usually not good for sizing
  - Benchmarking application may be very different
- More rigorous approach is to use modeling
  - Based on the amount of resources needed per transaction for your application
  - Done as a service by Oracle consultants

ORACLE

The suggested “typical” configuration are for average applications designed according to best practices. As far as performance heavily depends on the way applications are implemented, it is difficult to properly size applications that are unique in one or more ways (and many are) without collecting at least some performance information.

Agenda

- Introduction
- Performance Assurance
- **Performance Troubleshooting**
- Examples



ORACLE

The System is Slow

- Typical knee-jerk reactions are usually not effective
  - Add more hardware
    - May help only if the bottleneck is lack of the specific resource
    - Even if more CPU power is needed, it is difficult to guess where and how much without analysis
  - Submitting a support Service Request
    - Product defects rather rare comparing with configuration and application design issues
    - Nothing can be figured out until the problem is analyzed and narrowed down
      - Submitting a vague SR may slow down investigation

ORACLE

Investigate before act. “Shooting in the dark” rarely helps, but adds frustration.

What Can Be a Problem?

- Lack of hardware resources
  - May be network bandwidth, CPU resources, memory, I/O
- Tuning / configuration
  - On all tiers including network, operating system, storage, application
- Application design
  - May be any part including metadata, forms, rules, etc.
- Product issue
  - Rare comparing with above issues

ORACLE

It may be many reasons for bad performance, including lack of hardware resources, inadequate tuning or configuration, issues with custom application design, or even an issue with the product itself (which is relatively slow). And, of course, it may be a combination of issues.

Performance Troubleshooting

- Use Top Down approach
- Investigate step by step, narrowing the problem
- What exactly and where exactly is slow?
  - Is it slow for one user?
  - Does it change with time?
  - What components are active (see monitoring results)?
  - Do you see slowness in back end?
    - For example, in logs
  - What activity or data it is related too?
    - For example, is it related to a specific web form?

One complication may be that it could be several performance issues disguising each other. It makes investigation more difficult, but still there is no other way as identify and fix every issues one by one. No magic bullets here.

Monitoring

- Ongoing monitoring of all components
  - A way to check system health
  - Input for future changes / capacity planning
  - Input for performance troubleshooting
- May be done using most enterprise monitoring tools
  - May be done with OS-level tools, although it is usually not the best choice for ongoing production monitoring.
- What to monitor?
  - System-level metrics (CPU, memory, I/O, network)
  - Process-level metrics for major components (CPU, memory)
  - Database metrics

ORACLE

Monitoring may be done with OS-level tools (such as Performance Monitor for Windows and vmstat, ps, sar for UNIX), although it is usually nor the best choice for ongoing production monitoring.

Things to monitor: system-level resource utilization metrics, process-level metrics for key processes, database metrics.

Component Diagrams

- Understanding of how requests flow through the system is very important for all performance-related questions:
  - Distributing components over hardware
  - Monitoring
  - Performance testing
  - Performance troubleshooting

Understanding what component is doing what is very important. During performance testing, for example, you need to know what components you need to pay attention to. And, vise versa, seeing activity on a component during monitoring, you may guess what kind of workload may cause this activity.

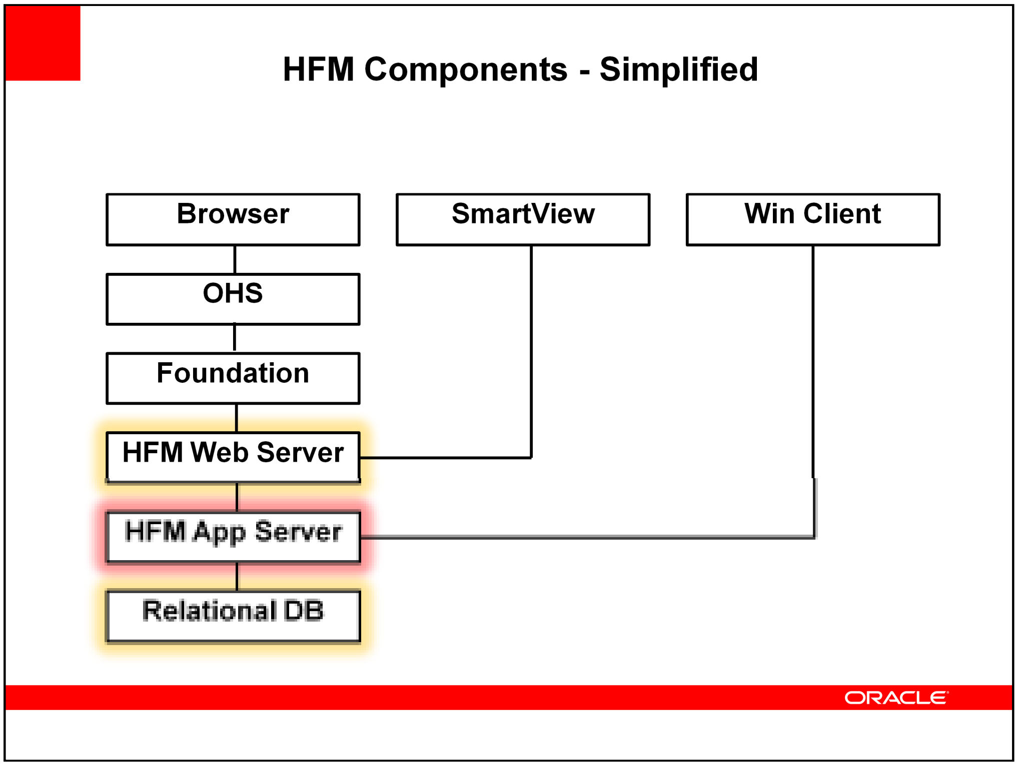
EPM Requests Flow

- The collaboration between components is really sophisticated, but not all of them are equally critical from the performance point of view
  - Focus on high-concurrency user requests
  - Simplified component diagrams are presented here to highlight high-concurrency components
  - See the "Installation Starts Here" manual for more detailed diagrams

ORACLE

It doesn't mean that other components never had performance issues – it just mean that they are used mostly by few users or for one-time kinds of activities, usually with low concurrency. Due to the time limitation, only the most high-concurrency products and paths are discussed. The presentation mainly discusses the products typically having the highest concurrency in most EPM implementations: Hyperion Planning, Hyperion Financial Management, Hyperion Essbase, and reporting solutions (Hyperion Financial Reporting and Hyperion SmartView). Actually a detailed discussion even about a single product hardly may fit a single presentation timeframe, so here these products are mentioned rather as examples to illustrate the advocated approaches. Further details could be found in manuals and product-specific documents.

More information in the Component Architecture documents at http://www.oracle.com/technetwork/middleware/bi-foundation/resource-library-090986.html

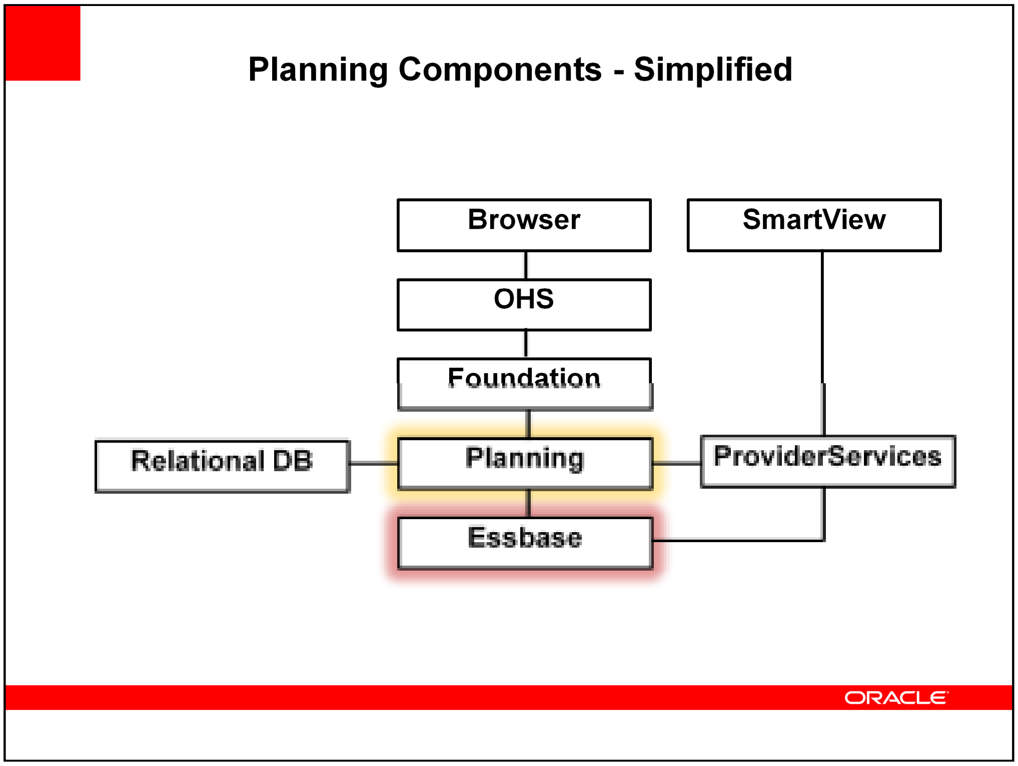


This is a simplified HFM component diagram for the components and flows usually involved in high-concurrency transactions. The components needed most attention from the performance point of view highlighted with yellow and red glow.

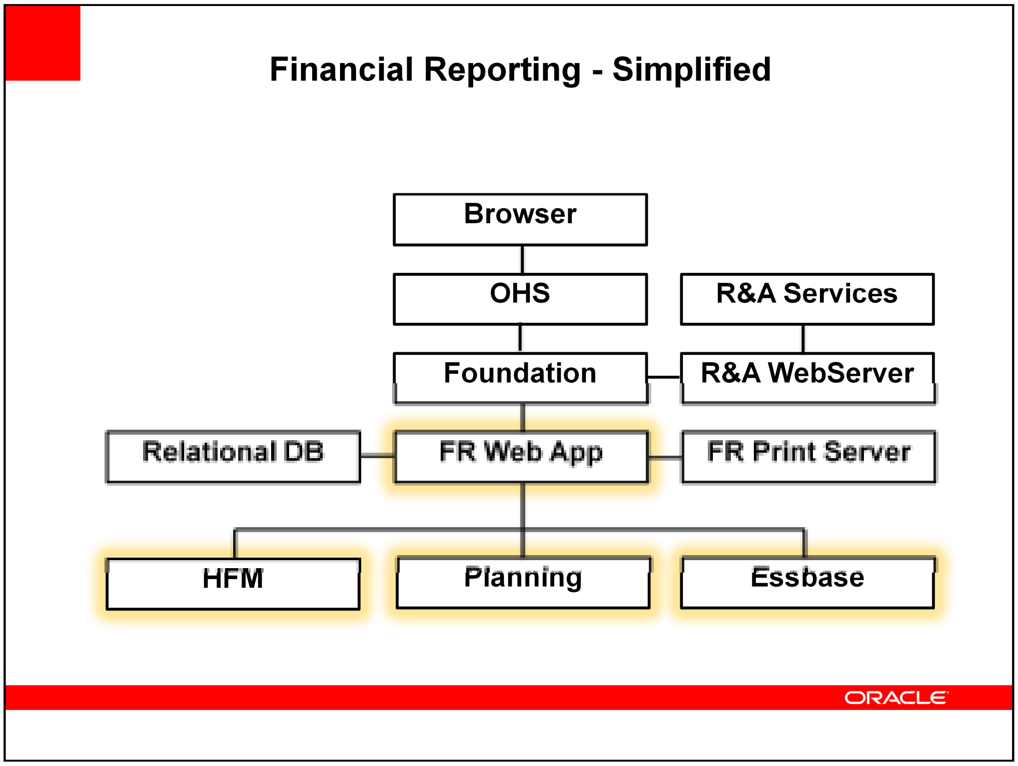
The choice of components / highlighting is based on the author personal experience only and was simplified to fit presentation slides. Other components may be important from performance point of view too.

OHS stands for Oracle HTTP Server.

*Foundation consisted of two components, Shared Services and Workspace, before version 11.1.2.



The main components for Planning from the performance point of view are Planning Web application (a J2EE application) and Essbase as its main datastore. Relational Database is used mostly as the repository, so usually is not a bottleneck.



The main components here from the performance point of view are Financial Reporting Web application and data sources.

To illustrate the importance of request flow understanding: Financial Reporting Print server is used only for pdf printing. So it is one of the most important components to monitor if pdf printing is involved and completely irrelevant if there is no pdf printing.

*There were three components (Financial Reporting Web applications server, Reports Server and Scheduler Server – last two standalone Java applications) instead of single Financial Reporting Web applications server before version 11.1.2.

Mapping to System Processes

- Each component may be mapped to one or several system processes
- Most Web applications are represented by HyS9<name> processes on Windows and Java processes on *unix.
  - Use ps –ef| grep <name> on *unix to find PID
- Key processes for HFM are HsvDataSource (one per application*) for App server, w3wp for Web server
- Key processes for Essbase are ESSSVR (one per application*)

* application, according to the traditional EPM terminology, refers to a specific implementation for the given product

ORACLE

Each component may be mapped to one or several system processes. Most Web applications are represented by HyS9<name> processes on Windows and Java proceses on *unix. Use ps –ef| grep <name> on *unix to find PID for specific component.

Key processes for HFM applications server are HsvDataSource and for Essbase - ESSSVR. One such process is spawn per application, so it may be multiple such processes (while orchestrating HsxServer and ESSBASE processes respectively usually don't use much resources). Key process for HFM Web server is w3wp.

A combination of all artifacts, including metadata, data, forms, rules, etc. is traditionally referred in EPM as an application. It creates some terminological confusion: the product itself may be referred to as an application and one specific implementation inside such product is referred as an application. Talking about performance assurance in this presentation we usually mean an implementation for the given product.

Essbase Application Logs

[Fri May 13 10:56:09 2011]Local/gsi1/Plan1/admin/6844/Info(1003037)

Data Load Updated [30507] cells

[Fri May 13 10:56:09 2011]Local/gsi1/Plan1/admin/6844/Info(1003051)

Data Load Elapsed Time for [SQL] with [AIFData.rul] : [6.516] seconds

[Fri May 13 10:09:05 2011]Local/gsi1/Plan1/admin/6500/Info(1020055)

Spreadsheet Extractor Elapsed Time : [0.157] seconds

[Fri May 13 10:09:05 2011]Local/gsi1/Plan1/admin/6500/Info(1020082)

Spreadsheet Extractor Big Block Allocs -- Dyn.Calc.Cache : [0] non-Dyn.Calc.Cache : [0]

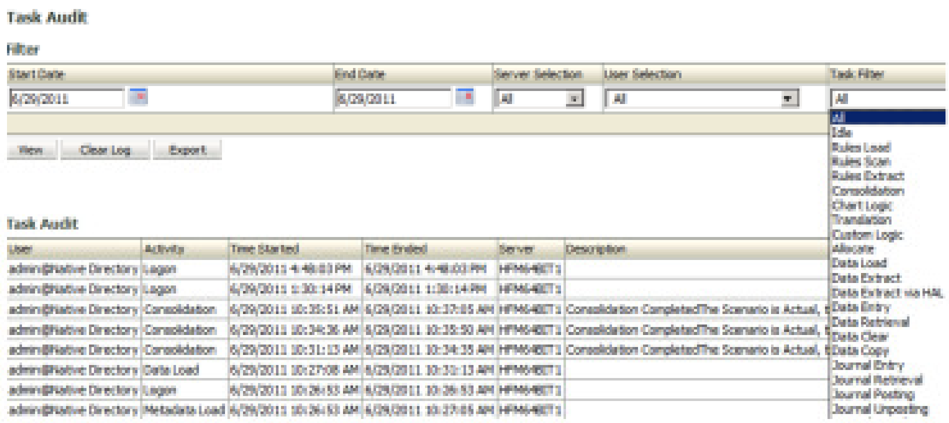
[Fri May 13 10:12:41 2011]Local/gsi1/Plan1/admin5308/Info(1020055)

Spreadsheet Extractor Elapsed Time : [0.031] seconds

ORACLE

Essbase application logs provides timing for all transactions. Look for 'Elapsed Time' records.

HFM Task Audit



ORACLE

Started and ended times for many HFM tasks may be found in Task Audit (data retrieval only for Financial Reporting) in most convenient form. In the logs it would be separate records for starting and for ending tasks.

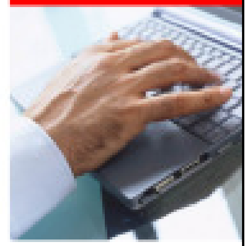
Service Request

- If you still believe that it is a product issue, submit SR with full results of your analysis, monitoring details, and logs
  - Statement "system is slow" is not enough
- There may be additional tools for investigation, such as debug and profiling flags, but they need to driven by Oracle support

ORACLE

The more issue would be investigated and narrowed down, the more chances that support would be able to help.

Agenda



- Introduction
- Performance Assurance
- Performance Troubleshooting
- **Examples**

ORACLE

Exemplary Performance Issues

- Let's discuss several typical performance problems
  - Each has a recognizable pattern
  - Happens often enough to be recognized

ORACLE

Many issues have a very recognizable pattern and happen often enough to be aware about them.

Examples: CPU Issues

- Using [almost] all available CPU
- May indicate lack of hardware resources
  - Add more servers for the component
  - Verify that adding hardware will solve the problem

ORACLE

Verify that adding hardware will solve the problem. For example, if the server is maxed out with 150 users and you need to support 200 users, there is a good chance that adding a second server will solve the problem (to be sure it need to be tested). However, if the server is maxed out with 10 users and you need to support 200 users, it is better to re-visit design and tuning; adding hardware doesn't look like a good option.

Example: Dynamic Members in Essbase

- High Essbase CPU during concurrent reading
- Dynamic members are very useful in some cases, but they are recalculated each time they retrieved
- That actually means that they shouldn't be used if retrieved concurrently
  - Only in cases when they are retrieved occasionally
- Solution: make dynamic members retrieved concurrently stored or remove them from concurrent activities such as reports / web forms

Dynamic members is an example of issues that can't be found without multi-user workload. It may be fine with one user and expose itself only under concurrent load.

Examples: Memory Issues

- Servers should have enough memory for all components
  - Using all machine memory (paging, swapping) kills performance
- 32-bit application process memory is limited
  - Windows 2GB (3GB in some cases)
  - Memory-hungry applications may benefit from 64-bit
- Java application memory consumption is defined by JVM settings
  - Max heap size –Xmx
  - Monitor actual heap size

ORACLE

To investigate JVM memory issues in most cases you need to monitor actual heap size (that usually require additional tools, some comes with Application Servers). In some cases Java process memory may be monitored if initial –Xms and maximum –Xmx heap size set to different values, but results may be obscured by the way OS manage memory.

Examples: I/O Issues

- Planning writes back – the same requirements as for OLTP systems
- Relational databases could have high I/O
  - HFM, FDM, ERPI
- Striping
  - The best RAID performance is with striping of data across multiple drives (RAID-0), which may be combined with mirrored disks (RAID-0+1 or RAID-10)
  - No RAID-5
- Split index from data files from control files along different I/O channels if possible

ORACLE

RAID-5 is optimized for reading, not writing. It introduces significant overheads for extensive writing.

Examples: Network Issues

- EPM provides rich web interface improving user experience
  - It may be not performing too well over networks with high latency or low bandwidth (remote offices)
  - It should checked in every situation when users are not in the same LAN as servers
    - For example, real network bandwidth measured and compared with network throughput generated by a user
  - If it is the case, software/hardware HTTP compression may alleviate the problem
    - Another solution maybe using Citrix/Remote Desktop

ORACLE

HTTP compression adds overheads, so it may be not a good solution for LAN users.

Scripting Example: HFM Consolidation

- Need a loop to be created in the script

```
web_custom_request("XMLDataGrid.asp_4",
  "URL=http://{WebSrv}/hfm/Data/XMLDataGrid.asp?Action=PROCMGT
  EXECUTE&TaskID={ConsolMode}&Rows=0&ColStart=0&ColEnd=0&S
  elType=1&Format=JavaScript",...
do {
  sleep(3000);
  web_reg_find("Text=1", "SaveCount=abc_count", LAST);
  web_custom_request("XMLDataGrid.asp_5",
  "URL=http://{WebSrv}/hfm/Data/XMLDataGrid.asp?Action=GETCONS
  OLSTATUS",…}
  while(strcmp(lr_eval_string("{abc_count}"), "1") == 0);
```

ORACLE

What each request is doing is defined by the *?Action=* part. In some context/versions, during the recording, you get multiple *GETCONSOLSTATUS* requests, the number of *GETCONSOLSTATUS* requests recorded depends on the processing time. If playback such script, it will work in the following way: the script submits the consolidation in the *EXECUTE* request and then calls *GETCONSOLSTATUS* three times. If we have a timer around these requests, the response time will be almost instantaneous. While in reality the consolidation may take many minutes or even hours (yes, this is a good example when sometimes people may be happy having one hour response time in a Web application). If we have several iterations in the script, we will submit several consolidations, which continue to work in background competing for the same data, while we report sub-second response times.

Consolidation scripts require creating an explicit loop around GETCONSOLSTATUS to catch the end of consolidation.

Scripting Example: HFM Web Data Entry Forms

- To parameterize, we need not only department names, but also department IDs from the repository

```
web_submit_data("WebFormGenerated.asp","Action=http://hfmtest.us
  .schp.com/HFM/data/WebFormGenerated.asp?FormName=Tax+Q
  FP",
  ITEMDATA,
  "Name=SubmitType", "Value=1", ENDITEM,
  "Name=FormPOV", "Value=TaxQFP", ENDITEM,
  "Name=FormPOV", "Value=2007", ENDITEM,
  "Name=FormPOV", "Value=Periodic", ENDITEM,
  "Name=FormPOV", "Value={department_name}", ENDITEM,
  "Name=MODVAL_19.2007.50331648.1.{department_id}.14.409.21
  30706432.4.1.90.0.345", "Value=<1.7e+2>;;", ENDITEM, LAST);
```

ORACLE

Another example is HFM Web Data Entry Forms. To parameterize such script, we need not only department names, but also department ids (which are internal representation not visible to users – should be extracted from the metadata repository). If department ids are not parameterized, the script won't work – although no errors will be reported.

Summary

- Performance Assurance is ongoing performance risk mitigation during the whole system lifecycle
  - Including design, development, testing, and production
- Performance testing of *your* application is a way to alleviate performance risk
- Performance testing of EPM products is not straightforward
- Use top down approach for performance troubleshooting

ORACLE