A Holistic Approach To Performance Tuning Oracle Applications Systems Release 11 and 11i

An Oracle White Paper
November 2001
Executive Overview

A number of problem definition methods, decision support systems, continuous improvement techniques, and other methodologies can be usefully employed for troubleshooting Oracle-based systems. However, many of these approaches are necessarily abstract and very generic. They can be so complex that they require specialist problem-solving skills, making their deployment almost as difficult as solving the problem they are endeavoring to identify. This paper describes a proven methodology specifically designed to identify and analyze complex and multifaceted Oracle Applications issues. Although this approach is primarily intended for use with Oracle Applications, it can also be used to identify problems with other Oracle-based systems. This paper should be useful to the whole user community, from end-users and system administrators to the CIO.

The approach to performance tuning Oracle Applications has changed significantly over the past year. Those familiar with the previous paper will recognize that many sections have been reviewed and updated. The biggest updates and changes include the following:

**Desktop PC**  There has been a fundamental shift in PC requirements, from memory in Smart Client to CPU in internet computing. At the same time, CPU speeds have increased significantly. This has fundamentally changed the way you need to tune the client. The benchmarks in this section have also been updated and include Windows XP.

**Database**  Statspack replaces the traditional BSTAT/ESTAT report and is now the preferred analysis and tuning tool. Although it was introduced with Oracle 8.1.6, it is also available for earlier versions though some restrictions apply.

**SQL**  This has been updated with the latest Applications information and contains some very important information that everyone should be aware of.

**Networking**  Oracle Applications network traffic has considerably reduced since the introduction of the internet computing architecture. However, changes in other technologies such as emails with large attachments, audio and video streaming, all mean that many companies are still seeing bandwidth requirements increasing. This section describes Quality of Service (QoS) and Packet Shaping, which are techniques that can help you to restore and maintain service levels for your mission critical applications without needing to upgrade expensive Wide Area Network (WAN) links.

**Concurrent Processing**  This section introduces several enhancements such as automatically moving files to secure areas, printing PostScript under UNIX, archiving, and many other features. Rather than include all the details in this paper, you are referred to white paper (164085.1 written by Andy Tremayne) that provides you with a myriad of functional enhancements.
**Contents**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>7</td>
</tr>
<tr>
<td>1.2</td>
<td>8</td>
</tr>
<tr>
<td>1.3</td>
<td>8</td>
</tr>
<tr>
<td>1.4</td>
<td>8</td>
</tr>
<tr>
<td>1.5</td>
<td>9</td>
</tr>
<tr>
<td>1.5.1</td>
<td>10</td>
</tr>
<tr>
<td>1.5.2</td>
<td>10</td>
</tr>
<tr>
<td>1.5.3</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>19</td>
</tr>
<tr>
<td>4.1.1</td>
<td>19</td>
</tr>
<tr>
<td>4.1.2</td>
<td>20</td>
</tr>
<tr>
<td>4.1.3</td>
<td>21</td>
</tr>
<tr>
<td>4.2</td>
<td>21</td>
</tr>
<tr>
<td>4.2.1</td>
<td>21</td>
</tr>
<tr>
<td>4.2.2</td>
<td>22</td>
</tr>
<tr>
<td>4.2.3</td>
<td>22</td>
</tr>
<tr>
<td>4.2.4</td>
<td>22</td>
</tr>
<tr>
<td>4.2.5</td>
<td>22</td>
</tr>
<tr>
<td>4.2.6</td>
<td>22</td>
</tr>
<tr>
<td>4.2.7</td>
<td>23</td>
</tr>
<tr>
<td>4.3</td>
<td>23</td>
</tr>
<tr>
<td>4.4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>5.1</td>
<td>25</td>
</tr>
<tr>
<td>5.2</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>26</td>
</tr>
<tr>
<td>5.3.1</td>
<td>26</td>
</tr>
</tbody>
</table>
5.3.2 Using The Forms Server Log File ........................................................... 27
5.4 On-Demand Downloading ..................................................................... 28
5.5 Apache .................................................................................................. 28
5.6 Forms Runtime Diagnostics ................................................................. 29
5.7 Performance Collector ......................................................................... 30
5.8 Enabling FRD In Release 11i ................................................................. 32

6 Tuning the Database .................................................................................. 33
6.1 Statspack ............................................................................................. 33
6.1.1 Features and Advantages .................................................................. 34
6.1.2 Using Statspack With Oracle Applications ...................................... 34
6.1.3 Setting Collection Thresholds ........................................................... 35
6.1.4 Running Statspack ........................................................................... 36
6.1.5 Table Growth .................................................................................... 37
6.1.6 Using Statspack with Oracle 9i ......................................................... 37
6.2 Cost And Rule Based Optimization ......................................................... 37
6.3 Gathering Statistics in Releases 11 and 11i ............................................. 38
6.4 Gathering Statistics Prior To Release 11i ............................................... 38
6.5 Monitoring Growth .............................................................................. 39
6.6 Block Size And Related Setup .............................................................. 40
6.7 Package Pinning Strategy ..................................................................... 40
6.8 Shared Pool Reserved Allocation ......................................................... 41
6.9 Latching ............................................................................................... 41
6.10 Hit Ratio .............................................................................................. 43
6.11 Managing The System Load ................................................................. 43

7 Tuning the Server .................................................................................... 45
7.1 Disk I/O: OFA vs Raw Partitions ........................................................... 45
7.2 Memory Bottlenecks .......................................................................... 46
7.3 Paging and Swapping Bottlenecks ......................................................... 46
7.4 CPU Bottlenecks .................................................................................. 46

8 Tuning The SQL Access Paths ............................................................... 47
8.1 Tracing Modules .................................................................................. 47
8.2 Encapsulating SQL ............................................................................. 47
8.3 Using Event 10046 ............................................................................. 48
8.4 Identifying Literal SQL ...................................................................... 49
8.5 Analyzing Literal SQL ........................................................................ 50
8.6 Finding Dynamic Form SQL ............................................................... 51
8.7 High Water Marks .............................................................................. 52
8.8 Expensive SQL .................................................................................... 53
8.9 Customized Views .............................................................................. 53
1 THE PERFORMANCE METHODOLOGY

As with all reactive performance analysis, time is of the essence. You need an investigative method not only to analyze problems quickly, but also to identify the source of the problems when they occur. By using this method, you should be able to quickly focus your investigative efforts and identify the expertise or skill-set needed to resolve the performance problem. This approach continues to prove its worth in Applications 11i.

The increasing breadth and depth of the Applications technology stack means that adopting an 'end-to-end' approach, addressing all system components, is becoming more and more demanding. However, the end-to-end approach is the only way to start investigating any complex issue that may have several contributing factors. Furthermore, in Applications 11i, it is becoming more difficult to locate and resolve issues when working with each technology stack components in isolation.

To expedite your investigation, you need to use an approach that quickly evaluates the performance of each of the interacting components. In other words, an approach that considers the system in its entirety - or 'holistically'. The problem solving techniques in the approach proposed by this paper loosely model other more generic approaches such as Kepner Tregoe and those proposed by the University of Maryland. It integrates some of the best features from each, while maintaining the common approach of five problem definition stages: What, Where, When, Extent, and Priority.

1.1 DECISIONS YOU NEED TO MAKE

Which methodology you decide to use is a matter of personal choice; however, the one documented in this paper has been successfully used during many Applications performance engagements. It documents an approach based on real-world experience that has evolved from numerous performance engagements working across a range of fundamentally disparate problems. It has repeatedly been shown to reduce the effort and expense of tracking down and identifying system problems.

A great deal of time has been expended on maintaining the speed and simplicity of the approach, while ensuring it remains accurate and sufficiently comprehensive to identify the most complex Oracle Applications issues. Not only does it formalize the problem definition process, but it also enables you to correlate the subsidiary effects and possible consequences of poor performance from other technology stack components.

Throwing resources at a performance problem is rarely helpful. This often leads to contention between the parties, with you in the middle trying to mediate. If one of the on-site experts tells you that there are no problems in their areas, applying this methodology can provide a definitive insight.
1.2 DECIDING HOW TO START

Two strategies are commonly deployed when faced with a performance problem. One is to engage a database-tuning expert, who has specialized skills and knowledge of the Oracle tuning tools while the other is to throw hardware at the problem. Neither of these options provides a long-term solution as they only address the symptoms - not the causes.

There is a lot that you can do to quickly locate and identify system problems. The value of the method described in this paper is that while you may need an expert in a particular technology stack to resolve a problem, you do not need to be an expert to identify where the problem is located. Your goal is to narrow the scope of the investigation as quickly and cost-effectively as possible, enabling early identification of the correct resources that you need to help solve the problem. This will not only save you days or possibly weeks of sub-optimal performance; it can also avoid the significant costs associated with extending the working day or having to change the business practices in an attempt to address the shortfall.

1.3 TOOLS USED WITH THIS METHOD

Start by spending a few hours gathering as much information as possible about the entire system and all of its problems. Without this type of comprehensive information, you risk wasting days of effort investigating the wrong area and making little or no progress with what are effectively time-critical issues. The two tools that you will need to assist with information gathering and problem solving are as follows:

- **The Project Documentation Summary (PDS)** This document simply structures your project documentation by collating and summarizing all of the pertinent performance factors across the entire system into a single document. It provides a detailed overview of the Oracle Applications environment. You should take a moment to review this document, which is available on MetaLink. (Document: 165300.1) This will help put the remaining concepts throughout this paper into context. This is the preferred document created by Oracle Development and used by Oracle performance experts worldwide when dealing with complex performance issues.

- **The Problem Definition Flowcharts** These flowcharts help accurately define the exact nature of each performance problem. Not only does this provide an unambiguous problem definition, but it can also highlight commonality and idiosyncrasies that would otherwise go unnoticed. Defining the problem is time-critical, so having an easy-to-follow systematic approach that rapidly narrows the investigative scope is vital.

1.4 DOCUMENTING YOUR SYSTEM

You should not consider the PDS to be just another piece of project documentation. The specification and configuration for each of the system components helps you to understand the main performance indicators across the entire system. In an emergency you could simply complete the mandatory (shaded) sections. By keeping the document up-to-date, you can add it to your toolkit so it is always at hand when addressing time-critical performance issues.
A single person would have to collect the information serially, possibly by interviewing technical staff one at a time. This will introduce a substantial delay and increases the risk of inaccuracies that might lead to you incorrectly focusing your investigation. Instead, you should send each sheet to a technical expert in that area. This will avoid inaccuracies and the entire PDS should only take two or three hours to complete, after which, you can start your initial analysis.

Highlighted fields denote the minimum required information and generally appear at the top of each sheet. Although providing additional information is highly recommended, it may be tempting to only provide minimal information for areas that are considered to be problem free. This is a dangerous tactic and may seriously delay the investigation, or in a worst-case scenario, incorrectly focus the entire investigation. In general, the PDS saves time and summarizes potentially huge amounts of unnecessary detailed system information, which would only cloud the clarity of the document.

Once completed, a fairly simple analysis should clearly identify which areas you need to concentrate your efforts to achieve the maximum performance improvements. In fact, a number of customers who have documented their system have resolved their own performance problems. They used internal technical staff who then had the benefit of being able to understand the interaction with other system components outside of their own domain. This demonstrates that the PDS can be used to cross the technical domains and boundaries that naturally exist within your company when specialists try to protect their own particular part of the system. Other customers, lacking the necessary technical expertise in-house, have used it to quickly narrow and focus their investigation, enabling them to isolate the problem areas and identify the exact resources that they needed to address the problem.

1.5 The Three Types of Application Systems

It is essential to understand that Oracle Applications has three completely different types of processing. The PDS summary page asks “the customer” to classify the main type of performance problems. Each type of system has fundamentally different processing requirements. This single element of information will help you to understand the system profile and immediately start to think about possible causes.
On a poorly tuned system, users will keep multiple forms open dramatically increasing the amount of memory required on the middle tier, which will exacerbate the memory problem, slowing end-user response times further.

1.5.1 **ONLINE TRANSACTION PROCESSING (OLTP)**

Systems that are mainly used for OLTP are characterized as high-throughput, insert/update intensive systems. They have a constantly growing database, and usually a high number of concurrent users. Major concerns for this type of system are up to date statistics, shared pool dynamics, the set of pinned packages, the temporary tablespace type and extent size, a well-designed concurrent processing strategy, literal SQL, indexes, and most importantly, well-tuned SQL statements.

For Applications 11i, the speed and amount of memory on the machine used for the middle tier is very important. The network used to be a major issue with Smart Client, but should be less of a concern unless working over a high latency WAN.

1.5.2 **BATCH PROCESSING**

Most batch data processing modules transfer large amounts of data entered by OLTP users or loaded via interfaces. In Oracle Applications, General Ledger posting processes are a good example of this type of processing. Primary concerns are the amount of memory reserved for sorting records, high water marks on interface tables, the temporary tablespace type, the temporary tablespace extent sizes which are used for sorts that do not fit into memory; the size and number of rollback segments, redo logs, placement of the out and log directories, debug and trace profile options, and the required transaction throughput of either detail or summary data. Batch jobs and reports that do not complete overnight will probably cause performance problems for the on-line users during the day.

1.5.3 **REPORT PROCESSING**

Reports typically sort large amounts of data and exhibit similar requirements to batch processes. Poorly written custom reports can consume huge amounts of memory and CPU. The General Ledger Financial Statement Generator is an example of a very flexible tool that allows code to be written very well, or very badly. Main concerns are temporary tablespace type and extent sizes, the efficiency of the SQL statements, and the transaction throughput.
2 PROBLEM DEFINITION

The approach to defining problems forms a very small but pivotal element of the overall problem resolution process. Accurate problem definition is key to identifying the source of performance issues and will help focus your tuning effort.

Regardless of whether you are new to the problem or working with it for some time, use this process to give yourself a fresh perspective, in essence stepping back and examining the whole picture. This process works equally well in problem solving regardless of whether you are an external consultant or an employee. The process has five stages that are summarized in the following table.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHAT</td>
<td>What is the nature of the problem? Which component is specifically under-performing?</td>
</tr>
<tr>
<td>WHERE</td>
<td>Where does the problem occur?</td>
</tr>
<tr>
<td>WHEN</td>
<td>When does the problem occur?</td>
</tr>
<tr>
<td>EXTENT</td>
<td>How many users or locations are affected? How much functionality is affected? Is the problem isolated?</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>What is the relative importance of this problem?</td>
</tr>
</tbody>
</table>

These five stages ensure that you fully understand the issues, what has been done to resolve the problems and the outcome of that work. This is especially important for Oracle Applications as an implementation usually consists of several modules that can be implemented and used in any number of ways.

If you have done any investigative work, you will probably recognize many of the questions in the problem definition. All the flowcharts do is provide a framework for you to follow. However, as each question is answered, try extending each question by asking the “converse” as this may reveal clues that have never been apparent before. For example, where else might you reasonably expect the problem to occur - but it does not? Using converse questions allows you to narrow the scope of your investigation by excluding those parts of the system that are not affected. Minimizing the scope of your investigation is of paramount importance when dealing with complex problems.
2.1 WHAT

The questions in this step try to specifically characterize the nature of the problem, to ensure you have a full understanding of exactly what the problem is. What is the nature of the problem? What is specifically under-performing? Describe the problem in terms of application modules and/or specific transactions. The answer that “the database is slow” is imprecise and should not be considered acceptable. The answer that “all online applications users in any location using any form experience performance problems during peak load”, is a much clearer detailed problem definition and would lead to a review of all the major components - server, database, middle tier, network, and clients. State the problem clearly using precise terms and metrics; if necessary, include module version numbers and a characterization of the data that can be used to recreate the problem.

For each item in the problem list:

OLTP module, batch process or Application

Q1 Are there any customizations?
   Yes → Possible Cause: Customizations Check for a performance problem in the standard code once the code has been identified as an issue.
   No

Q2 Are all particular transactions or types of functionality slow?
   Yes → Possible Cause: Expand complex issues to identify common elements or contributory factors. The cause depends on the specific type of functionality.
   No

Q3 Is it the latest code version?
   Yes
   No → Possible Cause: Code Version. Investigate performance changes between versions.

Q4 Is this the first run on this instance or at this location?
   Yes
   No → Possible Cause: System setup, system install, Applications install, database setup and configuration.

End

WHAT: Define the Problem
2.2 WHERE

Where does the problem occur? If the problem occurs with specific users, specify the locations and list differences between them. However, this level of detail may not be required if this is a system issue such as on the middle-tier or when running concurrent jobs on the server. Each item in the problem list needs to be assigned to one of the two categories, “batch” or “other”, before investigation commences; the system is excluded from this stage, as you will typically only be investigating a single system. During this stage the holistic approach ensures that the complete system is evaluated.

It is essential to identify where the problem exists and if there are locations where performance is adequate. It follows that it is important to evaluate not only the users' expectations and the relative speeds at which they work, but also the equipment they are using and where they are located. Any variation that establishes satisfactory performance in some areas completely changes the standard tuning paradigm and fundamental approach to tuning. Although this section specifically excludes instances, if the customer has a number of instances on the same machine, you might want to investigate each environment.
2.3 WHEN

When does the problem occur? Note the times of day or dates when the problem occurs. Link the dates to specific events or parts of the Applications or business processing cycle. Identify whether the problem has always existed, whether there was a sudden or gradual deviation, and if the problem persists when the problem module is run on its own. Identify environmental factors and any patterns of system usage when the problem occurs. If the problem has not always existed, attempt to relate these to historic events, module versions, and data volumes.

For each item in the problem list:

- General performance problem list

Q1: Has it always been slow?
- No
  - Q2: Did it change suddenly?
    - No
      - Identify when the problem was first observed and when it was not a problem
    - Yes
      - Identify the date/time
      - Possible Cause: System changes and/or events. Investigate using the customer's system change record

Q3: Is it slow all the time?
- No
  - Identify the times/days it under performs
- Yes
  - Identify system conditions when under performing

Q4: Is it slow when the system is heavily used?
- Yes
  - Possible Cause: System load
- No

Q5: Is it getting slower?
- Yes
  - Possible Cause: Data volume, number of users, network traffic or system load related
- No
  - Possible Cause: Indeterminate

**WHEN**: Specify When the Problem Occurs
2.4 EXTENT

How many users or locations are affected? How much functionality is affected? Is the problem isolated to a single database instance, application or module? When multiple parts of the system are suspect, look for common factors, and try to narrow your investigation to one or two threads. If a problem appears to be isolated, try to establish what makes it different from the remainder of the system.

**EXTENT: Specify the Extent of the Problem**
2.5 PRIORITY

Broadly categorize the issues into three areas: those critical to sustain critical business functions; those that are important, but will not cause a direct business failure; and the remainder, which are desirable, but not required for day to day operation. Consider postponing issues that cannot be recreated or reproduced during the current part of the business lifecycle. Agree a sequence based on the extent, frequency of occurrence, and business requirements.

This step may occur during natural conversation for each of the issues, but you will need to revisit the step at the end to review the relative importance of each issue, and decide which must be dealt with first. For this reason, there is an optional link during the What, Where, When, and Extent discussion, and a mandatory link once you have completed the step.

For each item in the problem list:

- General performance problem list

  - Does it only have a small extent/impact?
    - Yes → Consider reducing priority
    - No →
      - Is there a viable workaround?
        - Yes → Consider reducing priority
        - No →
          - Is the problem in production?
            - Yes →
              - How often is it run?
                - Rarely → Consider reducing priority
                - Frequently → Identify when it needs to be fixed by
            - No → Consider reducing priority

  - Sequence issues in order of priority

Priority: Specify the Relative Priority of the Problem

In order to make problem definition even faster, the questions have been collated into the following checklist. Questions from the flowcharts are shown in bold italics.
<table>
<thead>
<tr>
<th>Stage</th>
<th>Questions and Actions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHAT</td>
<td><strong>What is the problem?</strong> List the precise problem details. Specify the problem characteristics. Categorize the problem as time, volume, or transaction based. Identify any related documentation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are there any customizations?</strong> Identify the nature of any customizations. Check if the problem reproduces when using the standard Applications module.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are all particular types of transactions or types of functionality slow?</strong> Identify separate stages. Identify which stages are slow. Determine whether all types or parts of a transaction are slow. Identify common factors.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is it the latest code version?</strong> Obtain and test the latest version.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is this the first run on this instance or at this location?</strong> Establish if this is the first time a module has been used. Identify system or environment changes. Use converse questions.</td>
<td></td>
</tr>
<tr>
<td>WHERE</td>
<td><strong>Where does the problem occur?</strong> Identify the locations. Use converse questions. Identify common factors.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is it slow when run alone?</strong> (Batch only) Categorize the type of workload. Debate rescheduling the program.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is it slow everywhere?</strong> (OLTP only) Identify differences between locations, clients, users, or network.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are any third-party products likely to affect network traffic?</strong> (OLTP only) Establish if the problem is localized. Identify the network traffic profile: saturation, bandwidth, and latency.</td>
<td></td>
</tr>
<tr>
<td>WHEN</td>
<td><strong>When does the problem occur?</strong> Identify an underlying trend. Correlate with system events and changes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Has it always been slow?</strong> Was there a time when performance was acceptable?</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Did it change suddenly?</strong> Identify if there was a gradual or sudden change. Identify when the problem was first observed and when it was not a problem. Note the date and time.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is it slow at all the time?</strong> Identify the times/days it under-performs. Categorize the problem as continuous, periodic or sporadic.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is it slow when the system is heavily used?</strong> Categorize the system resource profile: CPU, memory, and disk I/O. Debate rescheduling other programs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Is it getting slower?</strong> Identify changes in the data volume, number of users, network traffic and system load.</td>
<td></td>
</tr>
<tr>
<td>EXTENT</td>
<td><strong>What is the extent of the problem?</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are All Instances Slow?</strong> Review the data file layout and database initialization parameters. Categorize and compare the Applications implementations. Consider reducing the size or number of instances.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are all applications slow?</strong> Determine if the problem occurs across all Applications or just a subset of modules. Review the data file layout.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are all batch programs slow?</strong> Review the data file layout and database initialization parameters. Review the Applications implementation and Profile options. Review the database activity.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Are all Forms slow?</strong> Identify common factors. Determine if the problem only occurs when using particular forms or functions. Review the database activity.</td>
<td></td>
</tr>
<tr>
<td>PRIORITY</td>
<td><strong>What is the priority of each problem?</strong> Categorize the issues into three areas: those necessary to sustain critical business functions, those that are important but not essential, and those that are desirable.</td>
<td></td>
</tr>
</tbody>
</table>
3 THE SIX TUNING AREAS

Oracle Applications tuning starts with a pre-built application where Oracle Development has already completed the design and SQL optimization. There are several application tuning methodologies in the public domain, but these are not generally appropriate for dealing with pre-built applications. The reactive approach below is targeted at the areas that yield the greatest return with the least implementation cost. The sequence of steps is flexible depending on the type of problem - the exact sequence you need to follow should be defined during the audit phase.

Although there are six stages, tuning is not always a sequential process and occasionally areas have to be revisited, or cannot be fully investigated, as there are other bottlenecks in the system that need to be dealt with first. You should always start by tuning the client; the results you get from plugging it into the switch interconnecting the various servers will help determine which of the technology stacks you should investigate next. You will see that there is dashed connection between “Tune the Client” and “Tune the Network”. The reason for this is that internet computing overcomes Smart Client’s sensitivity to latency. This does not mean that you should never investigate the network next, just that it is highly unlikely to be a major concern. Do not worry if you do not have a clear direction at the moment as you will have more information in the “What Next” step (at the end of the next section).

Ideally, one or two people will perform the entire investigation together. This approach has the benefit of knowledge transfer, quick impact analysis, and problem consolidation across their particular areas of expertise. However, there is an inherent problem; if too many technical people are involved, each is usually convinced that the problem is not in their area, so the investigation only achieves limited improvement or (more often) fails.

The Six Tuning Areas

- Tune The Client
- Tune The Database Structure
- Tune The Middle Tier
- Tune The Server(s)
- Tune Application SQL Access Paths
- Tune The Network

The Six Tuning Areas
4 TUNING THE CLIENT

The underlying assumption in performance tuning is that the equipment is adequate to achieve an acceptable performance (possibly with some patches or tweaks). While it is not possible to make a highly tuned 133MHz perform as well as a 600MHz Pentium, one way around the problem (which is typically overlooked), is the zero-budget solution of redeploying faster PCs to the high-latency or power users. This will enable them to achieve the necessary throughput for your critical business transactions.

There are several different versions of Windows, and they may need to be configured differently on different client PCs. Tuning the client PC is a definitive, easy, and relatively quick step that should not take more than an hour on an average Windows installation. Patches are available for most versions of Windows, and tuning the client can return up to 66% performance increase for Smart Client and around 40% for internet computing. Not only does tuning the client remove a major bottleneck, but it also ensures you have a test machine that can be used with confidence for benchmarks across the system. Plugging a PC directly into the same switch as the server (and middle-tier) eliminates any network bottlenecks, immediately isolating the tuning exercise to the database, application, and server.

4.1 BENCHMARK INFORMATION

It is impossible to establish if your overall system performance is as good as it can be without some point of reference, as you will never be able to determine if the performance levels are reasonable or below par. The comparative timings in this section provide a starting point.

4.1.1 RELEASE 11i WINDOWS/BROWSER BENCHMARK INFORMATION

The purpose of this benchmark is to compare browser performance against each version of Windows. This should identify which browser is best and on which version of Windows. It also provides a set of figures that you can compare your performance against. The browsers were Netscape 4.7 and Internet Explorer versions 5.5 and 6. The test machines were a very small UNIX server (running the database, concurrent processing and middle tier) and a Dell Optiplex GX100 500MHz Celeron 128MB RAM. JInitiator 1.1.8.16 was used for all tests. The timings shown in the graph represent normal user operation, which is to say that all necessary files are cached. Note that Netscape has been shown to be significantly slower in rendering many of the Applications screens and is only included here for comparison purposes. Internet Explorer is now the Oracle recommended browser for Oracle Applications. Always check MetaLink for the latest certified versions - Windows XP was not certified at the time of writing, but provides a useful comparison.
During the test, the machine load and network response time were constantly monitored for consistency. Dual timings and other appropriate controls were performed throughout. Note that:

- Service Pack 6a had to be installed for IE 6 on Windows NT Workstation.
- Windows XP is shipped with IE6. It is neither practical nor reasonable to regress IE 6
- Windows 95 and 98 SE (second edition) timings are identical - several performance patches had to be applied and both releases needed a significant amount of tuning.

Windows 95/ 98 produced the slowest times even after extensive tuning. The remaining times are remarkably close. Ignoring Applications startup, that should only happen once or twice a day, IE 6 appears consistently better than IE 5.5 (except for one or two test results). Netscape figures are all very similar but are slowest on Windows 2000 - even slower than on Windows 95/ 98. The best combination for Applications is much more difficult to determine as there is no clear overall winner. Statistically, the best results are IE 6 on Windows NT, then XP, then Windows 2000 in third place, but these are all within 3%.

4.1.2 RELEASE 11i CPU SPEED/WINDOWS BENCHMARK INFORMATION

This next set of results compares the time taken to load the Release 11i logon screen using Internet Explorer with cached JAR files for a range of Pentium-class processors and differing amounts of memory. The server hosting the middle tier was considerably faster than used for the previous set of tests and so you cannot directly compare the results. However, it does show that the speed of the middle tier is very important. It is worth repeating that these do not directly correspond to transaction times, as most users will only log on once or twice per day.

<table>
<thead>
<tr>
<th>PC Specification</th>
<th>Windows 95</th>
<th>Windows NT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pentium 200MHz 64MB</td>
<td>34s</td>
<td>36s</td>
</tr>
<tr>
<td>Pentium 266MHz 64MB</td>
<td>25s</td>
<td>25s</td>
</tr>
<tr>
<td>Pentium 300MHz 64MB</td>
<td>23s</td>
<td>20s</td>
</tr>
<tr>
<td>Pentium 400MHz 64MB</td>
<td>19s</td>
<td>15s</td>
</tr>
</tbody>
</table>

Release 11i startup times

Use Windows 95/98 with 266MHz or below; on 300MHz or higher use Windows NT/2000
4.1.3 PC SPEED AND LATENCY BENCHMARK INFORMATION

These tests compare Application startup tests using different specification PCs across varying latencies. Although they are a little dated, as they use AppletViewer rather than the JInitiator, they still contain some very important information. The timings are actually very comparable with the Windows/Browser timings. The latency was measured on a LAN (6ms), WAN (300ms), and between the UK and Australia (1400ms). As with the other results in this paper, these test results shows average values that you should expect to easily achieve or surpass.

<table>
<thead>
<tr>
<th>Latency</th>
<th>6ms</th>
<th>300ms</th>
<th>1400ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>133MHz Win 95 48MB</td>
<td>66</td>
<td>67.7</td>
<td>80</td>
</tr>
<tr>
<td>233MHz Win 95 128MB</td>
<td>30</td>
<td>36.5</td>
<td>53</td>
</tr>
<tr>
<td>300MHz NT 4.0 128MB</td>
<td>25.5</td>
<td>29.4</td>
<td>35</td>
</tr>
<tr>
<td>400MHz NT 4.0 128MB</td>
<td>21.4</td>
<td>26.5</td>
<td>35</td>
</tr>
</tbody>
</table>

Applications Startup Times (seconds)

Although a large amount of memory was used on the clients, this was not absolutely necessary. The amount of memory you will need depends on the particular combination of applications that you run simultaneously, and other factors such as the speed of the hard disk or the location of the paging file(s).

Notice that the time did not change on the 1400ms network link between the 300MHz and 400MHz PCs, as it did in other tests. This is because there were a significant number of retries on the 1400ms network; from this, you can speculate that the network had become the limiting factor. Furthermore, there is only a small amount of performance degradation over a WAN when using 300MHz clients or better.

4.2 TUNING WINDOWS

A complete performance checklist is included in Appendix A. Unfortunately, there is too much information to validate or explain all of the actions in this document; however, most people with a basic knowledge of Windows should be able to follow it. The discussion is provisionally about Windows 95/98, but most, if not all of the actions also apply to Windows NT and Windows 2000. Some of the main points are described in detail below.

4.2.1 DISK FRAGMENTATION

Hard disk fragmentation occurs over a length of time in all Windows environments and in fact upgrading from Windows 95 to NT or (Windows 2000 on NTFS) results in a fragmented file system due to the structure and placement of the Master File Table. Fragmentation means the files are stored in non-contiguous sectors on a disk, and so it takes longer to scan the hard disk to locate and assemble all of the file fragments. Severe fragmentation has quite an impact on all applications on the client; experience shows 4% is measurably detrimental.
There are two main considerations. Firstly, defragment before installing Oracle Applications Smart Client code or JInitiator. Secondly, disable the swap file before defragmenting, as the swap file fragments are immovable and will result in fragmentation after optimization, when it shrinks on reboot. Many defragmenters do not consider the optimal positioning of files and simply defragment based on the directory structure rather than frequency of file access. This in itself can introduce a significant delay in a file system suffering from such file sequence fragmentation. Windows NT does not come with a defragmenter but there are several excellent utilities available on the market.

4.2.2 VIDEO

Video cards are restricted by the available video bandwidth. This relates not only to the type of bus, but also the video drivers that govern the data transfer rates, throughput, and transfer modes. Check for new video drivers, which can provide around 20% video improvement, even on video cards that are only a few months old. Using 256 colors, instead of more, can save around .5 to .75 sec when opening a form, for both Applications 11i and Smart Client.

4.2.3 SCREEN SAVERS & WALLPAPERS

Screen Savers, including some supplied with Windows, use up to 8MB of system memory that is then not available to other applications, or causes an additional 8MB of paging to the swap file. This is also true of wallpapers and high-quality graphics/pictures. Even if you have a lot of memory, use the blank screen saver or none at all. Further, use a plain wallpaper or 16-color picture, both of which use the same amount of video bandwidth and memory.

4.2.4 CPU INTENSIVE TOOLBARS

Several office-type application suites have floating menu bars which are typically known as shortcut toolbars. Not only are they always in the way, slowing productivity, but some frequently use as much as 10% of the CPU, as they are frequently polled. This is easily demonstrated by timing Applications, with and without the toolbars enabled. It is faster and more efficient to use keyboard shortcuts to activate applications. Other associated items, such as fast find and fast start indexing programs should also be disabled in an Applications environment.

4.2.5 SPEEDING UP DATA ENTRY

A 10-15% increase in data-entry throughput can be achieved by speeding up the keyboard response speed. This can be done in two ways. Firstly, in the BIOS, ensure the Keyboard Rate is set to 30 characters per second and Keyboard Delay is set to a minimum. If these settings are not available, investigate obtaining a full BIOS setup program. Secondly, in the Windows Control Panel/Keyboard, set the Repeat Delay and Repeat Rate to optimal. This works in all versions of Windows.

4.2.6 TWEAKUI

Microsoft PowerToys is a free suite of utilities that extends many of the standard Windows features. It enables you to make some dramatic performance improvements to the Windows interface, to increase data-entry efficiency without needing to manipulate the registry directly. A particularly worthwhile option is to increase the speed as which menus appear. This and the keyboard speed improvement mentioned above gives the impression of a much faster client. This is available for all versions including Windows XP.
4.2.7 SERVICE PACKS - WINDOWS NT AND WINDOWS 2000

All versions of 5.5 are certified (SP 1 & SP 2). As Windows service packs are frequently updated, you should check MetaLink for certification information.

4.3 JAR FILE CACHING

You can monitor caching activity by looking at the contents of the Jcache directory under the JInitiator directory (version-dependent, but typically C:\ Program Files\ Oracle\ JInitiator\ Jcache). To confirm caching activity, close the browser and Oracle Applications, and delete the contents of the cache directory. When you restart Applications, you can watch the files being downloaded using Windows Explorer.

With JInitiator 1.1.7.15 and later, you can log the caching using any one of the following five options:

- **Off**  The default
- **Cache.verbose=TRUE**  Displays verbose output for all cache operations in the Java Console window. This is set TRUE by default in 1.1.8.x and later.
- **Cache.verbose.hit=TRUE**  Records when a file is retrieved from cache (cache hit)
- **Cache.verbose.miss=TRUE**  Records when a file is not able to be retrieved from cache (cache miss)
- **Cachelogfile=filename**  Stores verbose output in a file

These are specified in the JInitiator Control Panel as run-time parameters. An example of using the Cache.verbose=true option as shown below.
4.4 WHAT NEXT?

You should now have a tuned PC that can be used with confidence. Plugging the PC directly into the same switch as the server negates the effect of the network. At this point you should benchmark again. If you have achieved the necessary speed, then you need to focus on the PC or network; the amount of tuning you had to do to the PC will indicate which.

If you have not reached the target figures, then you need to review the database and server performance. If you can repeatedly isolate a problem, then you should trace the problem area. If you are unable to reproduce the problem, then you should concentrate your attention on the database and server. If the timings are acceptable, then moving the client away from the server one hop (network device) at a time will establish the location of a network problem. This process is shown diagrammatically below.
5 TUNING THE MIDDLE-TIER

This section briefly describes some of the topical middle tier issues such as whether to have a single server or multiple machines. It provides some guidelines for setting up Java Virtual Machines (JVMs) before going on to describe some of the more useful debugging techniques.

5.1 ONE OR MORE MACHINES?

One of the most important factors that you will need to consider in your transition to Release 11i is whether you should have one very large machine or several smaller machines. In order to answer the question, you will need to understand the different machine load profiles.

A middle tier hosts the forms process and for this reason, memory is most important with CPU speed next and finally disk I/O. The database server has a completely different load profile. Disk I/O and CPU are much more of a concern than memory. Having separate machines means that you can add middle tier machines as necessary, which provides a very scalable solution; it also simplifies the administration and management of each machine.

5.2 THE JAVA VIRTUAL MACHINE

One of the most common questions being asked is “How many JVMs do we need and how many users should we allocate to each?” The later part of this question is the easiest to answer first. Some benchmark tests suggest 20 concurrent users per JVM, while Oracle support suggest around 50 concurrent users per JVM. These figures refer to active users that are actually working at a specific moment in time; in reality, there might be many hundred users per JVM. It is clear that the answer lies between these two extremes. The difference depends on the type of Applications mix and how busy the users are.

You need to start by working out how many users that each JVM will support by performing a saturation test given your particular Applications mix and business. Once you have established a safe setting, you need to set the maximum number of concurrent users in security.maxConnections. Remember that some Self Service operations cover a very short period of time and as a result, you will occasionally see very short-term peaks. You need to check that you leave sufficient overhead to cover the peak periods.

In order to find out the number of Self Service Web Application users that have logged in you need to query ICX_SESSIONS. For example, to find out how many people logged in during the last hour you would run the following query:

```sql
SELECT COUNT(DISTINCT USER_ID)
FROM ICX_SESSIONS
WHERE LAST_CONNECT > SYSDATE - 1/24
AND USER_ID != '-1';
```

This is very useful as you can modify it to show how many people are connected per JVM. A count of V$SESSION for the different program types using STATUS=’ACTIVE’ will show you the number of active users.

The next question is “How much memory should we allocate to each JVM?” Each JVM has a lower “soft” and upper “hard” memory limit. For example, the settings: -ms128m and -mx256m allocate a heap size with a lower limit of 128MB and an upper limit of 256MB.
The garbage collector works very hard to keep memory within the soft limit. After a period of time, it will give up and then extends the memory allocation until it reaches the hard limit. Having a significant gap between the soft and hard limits is beneficial during testing, as you can measure the amount of memory (using the AM Pool monitor) used by each of the applications.

You should find that the amount of memory (hard limit) varies between 256MB and 400MB. Again, this depends on your processor speed, which will determine the actual number of users, and the Applications mix. Once you have established the amount of memory that you need for each JVM, set ms very close to mx, which will reduce the amount of work done by the garbage collector. Although there are no proven guidelines at this time, 80-90% should provide a good starting point.

Also ensure that you have set "ulimit -n 1024" when starting Apache and JServ processes. You can check that the file descriptors have been unlimited using the following command: pfiles <pid> where <pid> is the process id of the java process.

5.3 MAPPING USERS

As part of a troubleshooting or performance monitoring exercise, it is often useful to be able to map a session on the Forms Server back to a user. There are two ways of mapping users in Applications. The first relies uses Oracle Applications sign-on auditing, whereas the second approach uses the Forms server log file.

5.3.1 SIGN-ON AUDITING

The Sign-on: Audit Level profile option is used to set Applications auditing, which stores information about who signs on to Oracle Applications and when in the FND_LOGINS table. Each row includes the user, the ORACLE process (PID), the operating system process (SPID), the operating system terminal, and the operating system login name. If Sign-On Audit has been enabled at the User level or higher, then you get one row for each time a person signs on to an application (using the sign-on screen) during the time Sign-On Audit is active.

The Applications help screen contains the following values:

- Session SID maps to V$SESSION.SID
- SERIAL# maps to V$SESSION.SERIAL_NO
- AUDSID maps to V$SESSION.AUDSID

The following SQL can be used to map a user session to a database session:

```
select U.USER_NAME
from FND_USER U, FND_LOGINS L, V$SESSION V
where U.USER_ID = L.USER_ID
and L.SPID = V.PROCESS
and V.AUDSID = &1
```

```
USER_NAME
-----------------
OPERATIONS
```
5.3.2 USING THE FORMS SERVER LOG FILE

The connection details in the Forms Server log file can be used to map a process back to an IP address. The following extract from a Forms Server log file shows the process ID and IP addresses for each client connection:

Sun May 28 16:24:02 MDT 2000: starting Forms45 Server
FORMS CONNECTION ACTIVITY LOG FILE
Developer/2000:Forms/LogRecord

[Sun May 28 16:24:02 2000 MDT]: Server Start-up Data:
  Server Log Filename: /users/applmgr/log/F45_May28.log
  Server Hostname: calsunapps3
  Server Port: 9000
  Server Pool: 1
  Server Process Id: 5166

[Sun May 28 16:24:02 2000 MDT]: READY FOR CONNECTION [Port 9000]
[Sun May 28 16:34:35 2000 MDT]: Connection Request [ConnId=0, Addr=140.87.100.62:1167]
[Sun May 28 16:34:35 2000 MDT]: Client Connected [ConnId=0, PID=1253]

The following diagram shows a client connection to a Forms runtime process with a process ID (PID) of 1253. The connection request, IP address, and PID are captured in the Forms Server log file. The Forms runtime process then connects to the Oracle database server through an SQL*Net / Net8 process. Finally, the connection has an Oracle Session ID assigned within the database instance. The session and process ID information is captured in the V$SESSION and V$PROCESS views. If signon auditing is enabled, it can be combined with information from the relevant Application Object Library tables to record form and responsibility information.

For Oracle Developer Version 1.6.1, using the command `ps -ef | grep f45` shows the f45runw Forms processes and associated IP addresses. However, the IP addresses shown by this command are not the correct ones for the processes. The Forms Server log file correctly matches IP addresses and Forms runtime processes.
5.4 On-Demand Downloading

On-demand downloading of JAR files is available in Release 11i. This is enabled by the fnldlist.jar file, which contains lists of the class files contained in each JAR file. The startup HTML page is generated by a CGI script, which is configured through two files: appsweb.cfg and appsbase.htm. By passing the parameter config=alljar to the CGI program, the JavaScript in appsbase.htm creates a startup HTML file that excludes fnldlist.jar, thereby disabling on-demand loading, and causing all JAR files to be downloaded.

Downloading all the JAR files at once is useful during testing and configuration. It can be configured using the ICX/forms_launcher profile option, which can be set on an individual or site-wide basis. The CGI script URL in the profile option is appended with ?config=alljar; for example, http://cal.oracle.com:8002/dev60/cgi/f60cgi?config=alljar.

Other parameters that can be passed include config=debug for debugging options, and config=shortjars to enable download of the most commonly used JAR files. This list of parameters is being extended all the time.

5.5 Apache

The Oracle-supplied Apache products bundled with certain Applications products such as Internet Procurement are called the iAS Web Server because they are a component of Oracle's internet Application Server (iAS) product. The bulk of the configuration for Apache (Version 1.3.9 in Release 11i) is done in the httpd.conf file. On UNIX, it is located under the common utilities directory; for example, /util/apache/1.3.9/Apache/Apache/conf. In the Vision demonstration database, it is in the viscomn directory.

As part of Applications Rapid Install, Apache is pre-configured with port options, server name, and directory information. The default settings take into account most of the previously discussed performance considerations; however, the following parameter changes can be made to improve performance, depending on your installation:

- **HostnameLookups** Specifies that DNS lookups should be used for requests. Set this to OFF.
- **ErrorLog** Specifies the location of the error log file. Specify a different location to minimize contention.
- **LogLevel** Specifies the level of detail captured to the error log. Several levels can be used (default warn). Set this to the lowest level of emerg (emergency) once the server is running in a steady state.
- **AccessFileName** Specifies the name of the access control file (default .htaccess), which is not used by Oracle Applications. Comment this out to reduce file system reads.
- **MaxRequestsPerChild** Specifies how many requests each HTTPD process (HTTP Daemon) handles before being terminated; it is designed to minimize memory leaks that may occur with continuous, long-term use of Apache. The default is 0, which is unlimited. Consider setting this to 10000.

Apache has a large number of configuration parameters; as with most software, start with the default settings. When tuning, change one parameter at a time and measure the effects before making any further changes.
5.6 **FORMS RUNTIME DIAGNOSTICS**

The Forms Runtime Diagnostics (FRD) utility collects diagnostic information from a running form, including all the user actions. It produces extensive logging that captures both user and Form events, including the following:

- Trigger firing
- Built-in PL/SQL execution
- Messages
- Unhandled exceptions
- All external user events

The large volume of information and level of detail captured by FRD mean that it should not be your first recourse when diagnosing form problems. For example, if you log on to Oracle Applications Release 11, query Journal Batches in the General Ledger responsibility, and then exit, FRD will generate a log file of 1.8MB. However, it can be valuable when developing custom forms, or if you need to track and analyze a complex and elusive problem.

FRD requires Developer Server 1.6.1 Patch 6 (Forms 4.5.10.12). The easiest way to start FRD for Oracle Applications Forms is to create a modified HTML startup file that specifies `record=collect` and the log file location in the `serverArgs` parameter, as follows:

- **In Netscape**
  ```
  serverArgs="module=/u02/appltop/11.0/fndl/11.0.28/forms/US/FNDSCSGN userid=applsyspub/pub@VIS1103 fndnam=apps record=collect log=/users/applmgr/frd.log"
  ```

- **In Internet Explorer**
  ```
  <PARAM NAME="serverArgs" VALUE="/module=/u02/appltop/11.0/fndl/11.0.28/forms/US/FNDSCSGN userid=applsyspub/pub@VIS1103 fndnam=apps record=collect log=/users/applmgr/frd.log"/>
  ```

Note that FRD will quickly fill up disk space with log files and will also reduce I/O performance on the system. It is a diagnostic tool and should only be enabled on an as-needed basis, as this will create a trace files for each connection. The following example is an extract from a generated log file:

```
Forms Runtime Diagnostics Collection Log
Created: 24-MAY-2000 00:21:57
File Name: /users/applmgr/frd.log
Process ID: 2721
Forms 4.5 (Runform) Version 4.5.10.17.0 (Production)
PL/SQL Version 1.2.5.0.7 (Production)
Oracle Virtual Graphics System Version 2.1.12.13.0 (Production)
Oracle Multimedia Version 2.0.8.0.0 (Production)
Oracle Tools Integration Version 1.2.1.9.0 (Production)
Oracle Tools Common Area Version 2.0.5.0.4
Oracle CORE Version 4.0.5.0.0 - Production
Opened file: /u02/appltop/11.0/fndl/11.0.28/forms/US/FNDSCSGN.fmx
```
ON-LOGON Trigger Fired:
Form: FNDSCSGN
State Delta:
FORM FNDSCSGN
STATUS NEW
BLOCK EXPORT
STATUS NEW
RECSTATUS ""
FIELD WHERE
CANVAS EXPORT
GEOMETRY 0,0:667,192
ENABLED TRUE
NAVIGABLE TRUE
INSERTABLE TRUE

5.7 PERFORMANCE COLLECTOR

Performance Collector, available in Oracle Forms 6i, is an extension of FRD that enables you to collect timing information for key forms activities. To enable Performance Collector, specify record=performance and a log file location in the serverArgs parameter of the HTML startup file. The log file can then be analyzed using a Perl script called f60parse.pl. This is found in the $ORACLE_HOME/bin directory in a Release 11i installation, and generates a series of four HTML reports, two of which are very useful in a tuning exercise.

An example of a Performance Collector report summary is shown in the summary screen, including the Applications sign-on, selecting the General Ledger responsibility, navigating to the Enter Journals screen, and logging off. While a full description of the screen shown on the next page is outside the scope of this paper, some points are worth mentioning.

Timings are recorded for activities that take place on the client, network, Forms Server, and database server. When investigating the performance of a form, your first recourse should be to trace the transaction. SQL Trace will show you the execution times for SQL that were run against the database, but will not show the amount of processing by each trigger. By using both SQL Trace and Performance Collector, you can gain an understanding of all the interactions that take place. Furthermore, when investigating custom forms, the correlation between the statement and event times will help identify the location of the problem SQL in the form.

Clicking on any of the actions shown in the summary screen will automatically navigate to a more detailed screen, as shown in the next diagram. The top section, Details Level 1, expresses the total time taken for a given activity, categorized in terms of each of the technology stack components: client, network, database server, and Forms Server. The lower section, Details Level 2, provides a finer level of detail.
## Summary

<table>
<thead>
<tr>
<th>#</th>
<th>Action</th>
<th>Time of Occurrence</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Absolute Time (msec)</td>
<td>Relative Time (msec)</td>
<td>User + Client + Net Time (msec)</td>
</tr>
<tr>
<td>0</td>
<td>WINDOW START</td>
<td>197601399.082764</td>
<td>0.000</td>
<td>1801110</td>
</tr>
<tr>
<td>1</td>
<td>WINDOW SIGNON WINDOW.ACTIVATE</td>
<td>197605047.994791</td>
<td>3648.912</td>
<td>1801110</td>
</tr>
<tr>
<td>2</td>
<td>VALUE SIGNON.USERNAME</td>
<td>197610111.323230</td>
<td>8712.240</td>
<td>0.000</td>
</tr>
<tr>
<td>3</td>
<td>KEY _</td>
<td>197610111.323230</td>
<td>8712.240</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>VALUE SIGNON.PASSWORD</td>
<td>197612958.702828</td>
<td>11559.626</td>
<td>0.000</td>
</tr>
<tr>
<td>5</td>
<td>CLICK SIGNON.CONNECT BUTTON</td>
<td>197612958.702828</td>
<td>11559.626</td>
<td>0.000</td>
</tr>
<tr>
<td>6</td>
<td>USER EXIT.Responsibilities.&quot;General&quot;</td>
<td>197652841.953240</td>
<td>51442.876</td>
<td>56713.749</td>
</tr>
<tr>
<td>7</td>
<td>WINDOW SIGNON WINDOW.DEACTIVATE</td>
<td>197657960.440889</td>
<td>56561.363</td>
<td>0.000</td>
</tr>
<tr>
<td>8</td>
<td>WINDOW NAVIGATOR.ACTIVATE</td>
<td>197657960.440889</td>
<td>56561.363</td>
<td>0.000</td>
</tr>
<tr>
<td>9</td>
<td>ACTIVATED NAVIGATOR.ACTIVATE</td>
<td>197682096.640983</td>
<td>80697.566</td>
<td>154.992</td>
</tr>
<tr>
<td>10</td>
<td>ACTIVATED NAVIGATOR.LIST</td>
<td>197683412.132775</td>
<td>82013.050</td>
<td>887.656</td>
</tr>
<tr>
<td>11</td>
<td>WINDOW NAVIGATOR.DEACTIVATE</td>
<td>197684775.360013</td>
<td>83376.277</td>
<td>0.000</td>
</tr>
<tr>
<td>12</td>
<td>WINDOW FOLDER.ACTIVATE</td>
<td>197684775.360013</td>
<td>83376.277</td>
<td>0.000</td>
</tr>
<tr>
<td>13</td>
<td>WINDOW FOLDER.DEACTIVATE</td>
<td>197684775.360013</td>
<td>83376.277</td>
<td>0.000</td>
</tr>
<tr>
<td>14</td>
<td>WINDOW FOLDER.QS.ACTIVATE</td>
<td>197684775.360013</td>
<td>83376.277</td>
<td>0.000</td>
</tr>
<tr>
<td>15</td>
<td>WINDOW FOLDER.QS.CLOSE</td>
<td>197755364.597759</td>
<td>153965.515</td>
<td>0.000</td>
</tr>
<tr>
<td>16</td>
<td>WINDOW FOLDER.QS.DEACTIVATE</td>
<td>197755364.597759</td>
<td>153965.515</td>
<td>0.000</td>
</tr>
<tr>
<td>17</td>
<td>WINDOW FOLDER.ACTIVATE</td>
<td>197755364.597759</td>
<td>153965.515</td>
<td>0.000</td>
</tr>
<tr>
<td>18</td>
<td>WINDOW FOLDER.CLOSE</td>
<td>197757322.076422</td>
<td>156392.938</td>
<td>435.036</td>
</tr>
<tr>
<td>19</td>
<td>WINDOW NAVIGATOR.ACTIVATE</td>
<td>197758490.763560</td>
<td>157091.681</td>
<td>6688.040</td>
</tr>
<tr>
<td>20</td>
<td>MENU FILE.EXIT.Oracle Applications</td>
<td>197762382.206675</td>
<td>560983.124</td>
<td>38690.846</td>
</tr>
</tbody>
</table>

## Performance Collector Summary

### Action 6: USER EXIT Responsibilies."General"

#### Details Level 1

<table>
<thead>
<tr>
<th>GroupEvent</th>
<th>Total Time (msec)</th>
<th>Total Time (%)</th>
<th>Min Time (msec)</th>
<th>Max Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client+Net</td>
<td>26752.170</td>
<td>87.856</td>
<td>38.421</td>
<td>36713.749</td>
</tr>
<tr>
<td>DBPROC</td>
<td>11.996</td>
<td>0.029</td>
<td>11.996</td>
<td>11.996</td>
</tr>
<tr>
<td>FServer</td>
<td>4910.784</td>
<td>11.739</td>
<td>0.000</td>
<td>4901.951</td>
</tr>
<tr>
<td>GrandTotal</td>
<td>41574.950</td>
<td>99.624</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Details Level 2

<table>
<thead>
<tr>
<th>FSEvent</th>
<th>count</th>
<th>Total Time (msec)</th>
<th>Total Time (%)</th>
<th>Min Time (msec)</th>
<th>Max Time (msec)</th>
<th>Avg Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client+Net</td>
<td>1</td>
<td>38.421</td>
<td>0.029</td>
<td>38.421</td>
<td>38.421</td>
<td>38.421</td>
</tr>
<tr>
<td>FServer</td>
<td>1</td>
<td>1.164</td>
<td>0.008</td>
<td>1.164</td>
<td>1.164</td>
<td>1.164</td>
</tr>
<tr>
<td>FSP+DE</td>
<td>1</td>
<td>4901.951</td>
<td>11.718</td>
<td>4901.951</td>
<td>4901.951</td>
<td>4901.951</td>
</tr>
<tr>
<td>DBPROC</td>
<td>47</td>
<td>11.996</td>
<td>0.029</td>
<td>0.053</td>
<td>3.760</td>
<td>0.255</td>
</tr>
<tr>
<td>FSM+DE</td>
<td>46</td>
<td>7.669</td>
<td>0.018</td>
<td>0.039</td>
<td>5.688</td>
<td>0.167</td>
</tr>
<tr>
<td>User+Client+Net</td>
<td>1</td>
<td>36713.749</td>
<td>87.764</td>
<td>36713.749</td>
<td>36713.749</td>
<td>36713.749</td>
</tr>
<tr>
<td>GrandTotal</td>
<td>41674.950</td>
<td>99.624</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Performance Collector Details Levels
Details Level 1 shows that the amount of database processing time (DBProc) is minimal compared to the time spent in the client and network portion of the technology stack. This illustrates why timing the startup of a form is a very good benchmark test. By plugging the PC directly into the same switch as the database server and middle tier, the network time becomes insignificant and isolates the tuning exercise to the client. If the PC is subsequently tuned, the client is also effectively eliminated as a variable, and the investigation can then focus on the database and middle tier.

While Performance Collector is intended for use in custom development environments, you may find the event logging it provides is useful when diagnosing forms problems in general. Further information can be obtained from the Performance Collector for Oracle Forms 6i white paper, available from Oracle Technet (http://technet.oracle.com).

5.8 Enabling FRD in Release 11i

When using the Applications Personal Home Page to launch 11i, the "ICX: Forms Launcher" in the System Profile Options screen can be used to pass parameters to enable the Forms Runtime Diagnostics (FRD) utility. This should only ever be used to pass this type of parameter at the USER level, and not at the SITE level, where it would affect all users. Although it may be set at site level, the value should only contain the URL (without any parameters!). Depending on which Applications release you are using, the ICX: Forms Launcher profile option may not be updatable at the USER level, in which case you will need to access the Application Developer responsibility, and select the 'Updatable' checkbox for this profile option.

You need to set the value of the ICX: Forms Launcher profile option to include the URL and parameters at the USER level. Each parameter needs to be preceded with an '&'. In its simplest form you would use:

http://<machine_name>.<domain_name>/cgi-bin/f60cgi?config=<sid>&record=collect&log=<writeable directory for Apps>/<logfile>. So, to enable FRD, the profile value may look like:

http://aksun.us.oracle.com/cgi-bin/f60cgi?config=ak100&record=collect&log=/u01/app/logs/test.tfr.

This means that the process to create an FRD log file becomes very simple:

1. Obtain the URL value of SITE level profile option 'ICX:Forms Launcher'.
2. Append '&record=collect&log=/mydir/FRD.log ' to the URL and set the ICX:Forms Launcher profile option at USER level.
3. Open a new browser session and go to the new URL.
4. Do the same steps to reproduce the problem or until the error is generated.
5. Immediately log out.
6. Reset the USER level profile option

Alternatively, if you need to quickly check a session yourself, parameters may be passed directly in the URL:


When using this technique, the operating system user that owns the forms process must have write access to the specified directory.
6 TUNING THE DATABASE

If you are not performing your tests against the production system, then the test system should be representative of the production system or target implementation. Tuning application SQL is usually dependent on realistic data volumes and data-distribution; this is particularly true when using the cost-based optimizer in Applications Release 11i. Although the best solution is for test to be a copy of production, in Release 11i, you can export the statistics from production and import them into your test environment, so that the generated SQL execution plans will accurately reflect production regardless of data volumes.

Several database factors are critical for Oracle Applications. For example, periodically gathering statistics, database block size, pinning strategy, the amount of literal SQL and adherence to Oracle Flexible Architecture (OFA) can all have a major effect on memory, I/O bottlenecks, and resource contention. This section covers salient Application performance issues that are not generally discussed elsewhere.

Oracle Applications 11i requires certain init.ora parameters to be set for CBO use. You should validate your current settings by running the script $FND_TOP/sql/AFCHKCBO.sql (available in patch 1245516).

6.1 STATSpack

Broadly, there are two audiences for the reports. Firstly, Application Developers reviewing high load SQL/PLSQL, and secondly DBAs checking the overall health of a database, and looking for common issues not necessarily related to SQL tuning.

The UTLSTAT scripts, commonly referred to UtlBstat/ UtlEstat been shipped with the server since Oracle 6. The 8.1.6 release of 8i includes a new utility known as Statspack, which is intended as a replacement for the UTLSTAT scripts. Statspack not only extends the original scripts by capturing performance data for new database features, but also includes a summary page, which is better organized and enables quicker drill down. As Statspack has been written to include server features available in 8.1.6 and later, you cannot run it against an earlier release. However, modules that will run against earlier versions are available but are unsupported.

This section does not attempt to detail the installation and usage of Statspack. The installation readme is very comprehensive and contains a lot of additional information that you should find useful. There are three excellent white papers called “Diagnosing Performance With Statspack” available on the Oracle Technology Network at [http://technet.oracle.com/deploy/performance/content.html](http://technet.oracle.com/deploy/performance/content.html).
6.1.1 FEATURES AND ADVANTAGES

Whenever you start tuning, you need to create a baseline. Statspack reports created when there is no performance problem can provide a valuable comparison with reports created when there is a problem. Statspack has several advantages over its predecessor.

- It can be scheduled to run on a regular basis (typically every hour) and will capture the load over the snapshot period. In Statspack terms, a snapshot is a single performance data collection point and should not be confused with the database replication feature. Each snapshot is identified by a 'snapshot id', which is a unique number generated at the time the snapshot is taken.

- It stores performance statistics in Oracle tables, and has totally separate data collection and reporting facilities. When you run the performance report, it will prompt you for the two snapshot id's the report will process. (The first can be considered as the BSTAT and the second, as an ESTAT). The report calculates the instance activity between the two specified snapshots.

- This means that you can perform ad hoc analysis in addition to reviewing standard Statspack reports. Being able to analyze information in this way simplifies historical and trend analysis.

- It captures and reports high-load SQL statements, which are candidates for tuning. One of the most interesting statistics is the percentage coverage. The higher the coverage, the more likely you will have just a handful of SQL statements that are generating the majority of the load.

It is highly recommended that from Oracle 8i onwards, all Oracle Applications customers use Statspack for collecting database performance data. If you don’t have any baseline information available, then you should install and run Statspack as soon as possible. Although the need should never arise, you should not run BSTAT/ESTAT as the same user as Statspack as they both reference the STATS$WAITSTAT table.

6.1.2 USING STATSPACK WITH ORACLE APPLICATIONS

So long as your server is running 8i, you can use Statspack with all versions of Oracle Applications. The required files can be found in $ORACLE_HOME/rdbms/admin and have a prefix of 'sp'. You should always use the newer 8.1.7 version of Statspack, regardless of which version of 8i you are using as the space management is far more efficient and the report has a better layout on the summary page.

One of the most obvious limitations with the standard reports is the absence of the module name in the high-load SQL sections. When using Statspack with Oracle Applications, having a module name enables you to directly associate the SQL with a form or program; furthermore the absence of a module name implies that the source is custom code. This makes your analysis far simpler and quicker and the module name should therefore be considered as essential.
### 6.1.3 Setting Collection Thresholds

You can change the amount of information that Statspack gathers by specifying a snapshot level. The level dictates how much data Statspack collects. Level 5 is the default and should be used for Oracle Applications in Oracle 8i. The levels are as follows:

- **Level 0**: Statspack collects general performance statistics
- **Level 5**: Statspack additionally collects performance data about high-resource-usage SQL statements.
- **Level 10**: This is the highest available setting, which collects all the information at Level 5 in addition to child-latch information. Gathering this amount of information is rarely necessary and can take some time. You should use it only on the advice of Oracle personnel.

When you run a snapshot at level 5 (or above), you can change the thresholds to make them more appropriate for your instance. When working with Oracle Applications, you will find that the defaults are quite low and as a result you will see many more SQL statements than are “significant”. Setting a higher level will also reduce the amount of processing required to collect the statistics. The following table shows the default and recommended parameters for Oracle Applications; the non-default values are highlighted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Recommended</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_snap_level</td>
<td>5</td>
<td>5</td>
<td>Snapshot level</td>
</tr>
<tr>
<td>i_executions_th</td>
<td>100</td>
<td>1,000</td>
<td>SQL Threshold: number of times the statement was executed</td>
</tr>
<tr>
<td>i_parse_calls_th</td>
<td>1,000</td>
<td>1,000</td>
<td>SQL threshold: number of parse calls</td>
</tr>
<tr>
<td>i_disk_reads_th</td>
<td>1,000</td>
<td>10,000</td>
<td>SQL threshold: number of disk reads</td>
</tr>
<tr>
<td>i_buffer_gets_th</td>
<td>10,000</td>
<td>100,000</td>
<td>SQL threshold: number of buffer gets</td>
</tr>
<tr>
<td>i_sharable_mem_th</td>
<td>1,048,576</td>
<td>1,048,576</td>
<td>SQL Threshold: amount of sharable memory</td>
</tr>
<tr>
<td>i_version_count_th</td>
<td>20</td>
<td>20</td>
<td>SQL Threshold: number of versions of a SQL statement</td>
</tr>
<tr>
<td>i_all_init</td>
<td>'FALSE'</td>
<td>'TRUE'</td>
<td>Initialization Parameter Level <strong>see note</strong></td>
</tr>
</tbody>
</table>

** The standard Statspack report should list all the non-default database initialization parameters, however, it does not include the parameters that start with an underscore. If you are reviewing a Statspack report in isolation, then this might be misleading and you might incorrectly assume that none of the underscore parameters have been set. The main disadvantage with setting the I_ALL_INIT parameter to 'TRUE' is that you will waste space in the parameter table. The problem is caused by STATSPACK using V$SYSTEM_PARAMETER rather than V$PARAMETER. The Statspack developers are treating this as a bug in V$SYSTEM_PARAMETER and the problem is not fixed until Oracle 9i.
In order to implement the recommended threshold changes for Oracle Applications, you could execute a snapshot using the following command (this will temporarily override the necessary defaults):

```
SQL> execute Statspack.snap(
    i_executions_th => 1000,
    i_disk_reads_th => 10000,
    i_buffer_gets_th => 100000,
    i_all_init => 'TRUE');
```

You can use `statspack.snap` to permanently save the new default parameter values in the `STATS$STATSPACK_PARAMETER` table, using either `statspack.snap` at the time that you are taking a snapshot, or by using `statspack.modify` at any other time. Use the following command for Applications:

```
SQL> execute statspack.modify_statspack_parameter(
    i_executions_th => 1000,
    i_disk_reads_th => 10000,
    i_buffer_gets_th => 100000,
    i_all_init => 'TRUE');
```

### 6.1.4 Running Statspack

Before running Statspack, you should set `timed_statistics` to `TRUE` for your instance as this provides timing data on the report, which makes it is possible to undertake a much more accurate analysis. If it is not set then you can issue the SQL command `ALTER SYSTEM SET timed_statistics = TRUE` when connected as a DBA. The alter system command is an immediate operation, insofar that it is propagated to all connected sessions and becomes the default for new sessions. However, it does not come into operation for connected sessions until any currently executing command has finished. For example, if you have a long-running SQL statement, which typically takes an hour to complete, enabling `timed_statistics` while the statement is being executed will not result in the overall execution times being reported. In fact, you will get a mixture of timed statistics: I/O will be correctly reported, but not CPU time.

In other words, enabling timed statistics partway through a sampling period may produce results that are likely to be meaningless. If you enable timed statistics at the session level, wait a suitable amount of time before taking an initial snapshot. To take a snapshot, simply login as the PERFSTAT user, modify the default parameters and execute the SQL command: `execute statspack.snap`.

As with BSTAT/ESTAT the recommended sampling period for statspack is one hour. The sample period should not normally be less than an hour. If it is too short, long-running jobs may not be correctly recorded in the report. Also, it would be very inefficient to analyze the large number of reports that might be needed. Conversely, if a sample period is too long, for example, 24 hours or more, it is likely to cover the different profiles of both online and batch processing, the latter being much more I/O intensive. Any slack periods will also be included. The effect will be to average out peaks and troughs, making it very difficult to identify specific periods of interest.

This means that in order to monitor longer periods, you should collect several snapshots at regular intervals of approximately an hour. You can do this using one of the following methods:

- The `dbms_job` procedure
- An operating system utility such as `cron` on UNIX or `at` on Windows NT
6.1.5 TABLE GROWTH

The two tables that have the highest growth are STATS$SQL_SUMMARY and STATS$SQLTEXT. STATS$SQLTEXT is the main problem, especially with older versions of Oracle Applications, due to the large volume of literal SQL statements. You will need to run the Statspack purge program (sppurge.sql) to get around this problem. To avoid rollback segment related errors you need to explicitly specify a large rollback segment using the 'set transaction use rollback segment' command before running this script. Alternatively, you can specify a small range of snapshot id's to purge. For long-term information, consider creating a separate repository. Once you have exported the information from the main database, you can truncate the main tables.

6.1.6 USING STATSPACK WITH ORACLE 9I

When you upgrade to Oracle 9i, you will also need to upgrade Statspack, as the 8i version is incompatible with Oracle 9i. The only major change for data collection in 9i is the addition of a new snap level for collecting execution plans, Level 6. The default is still set to 5, so for Oracle Applications the Statspack.snap() level should be increased to Level 6.

6.2 COST AND RULE BASED OPTIMIZATION

The two available database optimization modes are rule-based (RBO) and cost-based (CBO). The RBO uses a fixed set of ranked rules to choose an execution plan based on the syntax and structure of the SQL statement, whereas the CBO uses statistics that reflect the data in the tables. In general, using the CBO safeguards against reindexing when data volumes change.

Release 11i is based on the CBO; however, hints have been prevalent in Receivables, Project Accounting, Manufacturing, and General Ledger since Release 10.7. Simple example hints are shown below. The first may be beneficial in a report; the second when returning information to a single-record form.

- SELECT /*+ ALL_ROWS */ empno, ename, sal, job FROM emp
- SELECT /*+ FIRST_ROWS */ empno, ename, sal, job FROM emp

Prior to Oracle Applications Release 11i, the optimizer mode initialization parameter must be set to RULE, as the majority of the SQL statements have been tuned for this optimizer. However, the RBO is automatically overridden by the CBO whenever SQL statements included cost-based hint – so do not change the optimizer mode! In other words, if you are running any release since 10.7, you need to gather statistics.

For cost-based optimization to be effective, the necessary statistics for the tables and indexes must reflect the exact nature of the data distribution. The statistics may be based on computation of the whole table, or on an estimate based on a sample of the rows. Estimation has the benefit that it is faster than computation and uses less space to sort the information. The statistics need to be periodically compiled using the ANALYZE command and are stored primarily in USER_INDEXES and USER_TABLES in the data dictionary.

- ANALYZE TABLE gl.gl_je_lines COMPUTE STATISTICS
- ANALYZE TABLE gl.gl_je_lines ESTIMATE STATISTICS SAMPLE 20 PERCENT
After gathering the schema level statistics Gather Schema Statistics creates the histogram for the columns specified in the FND_HISTOGRAM_COLS table. Furthermore, it populates default statistics for all the INTERFACE tables as specified in the FND_EXCLUDE_TABLE_STATS table. Before gathering the statistics, FND_STATS creates a backup of the existing statistics so that if the database should happen to slow down after gathering new statistics, you can use FND_STATS.RESTORE_SCHEMA_STATS to restore the system to its previous status.

In 11i, you can check the date and time that statistics were last gathered using FND_STATS.VERIFY_STATS. Alternatively, you can check the status of individual tables from SQL*Plus. For example, to check the GL schema you would use the following:

```sql
SELECT table_name, num_rows,
       to_char(last_analyzed, 'MM/DD/YYYY HH:MI') Last_Analyzed
FROM dba_tables
WHERE owner = 'GL';
```

Prior to Release 11i, you should always use the COMPUTE method, unless there are space or time constraints when working with large tables. If an inadequate sample size is used with ESTIMATE, then at some sites performance has reduced by up to 40% and so it is imperative to ensure that the sample size is sufficient to reflect the total data distribution. The main problem using a sample size of 20% is that a skewed data distribution will be reflected in the statistics; this is common in Applications. In Release 11i, the efficiency of the ESTIMATE approach has been significantly improved, and testing has shown that small samples can produce very accurate statistics.

The 10.7 Applications script $AD_TOP/sql/ADXANLYZ.sql by default uses an estimated sample of 20%; in R11 it uses 50,000 rows. To determine how much of a large table needs to be analyzed, consider an incremental approach starting at 20% and increasing by 10% each time, taking a snapshot of the statistics at each level. Comparing the variance in the snapshots will indicate when an appropriate sample size has been analyzed. Note that a full COMPUTE is automatically invoked when using a sample size of 50% or higher. If the data is analyzed at the correct level, where hints have been used, results from using the cost-based approach should be as good as, or better than, those from using rule-based optimization.

The question of how often to analyze is more difficult, as it requires an understanding of how quickly the data distribution changes. One strategy is to split the tables into sets. For example, Applications has numerous tables, such as FND_LOOKUPS, that contain static information. In addition, once your system has been running for a few months, reference information such as inventory items will not change very much. You could analyze this subset once every six months. Next, group the applications tables that do not change very much and analyze these every three months. For tables that do change frequently, such as PO_HEADERS, you could analyze these monthly. This means that tables which change frequently or are very large, such as GL_JE_LINES, can be analyzed individually, reducing the amount of time each ANALYZE run takes. You can snapshot the statistics before and after analyzing to check your strategy.
Comparing snapshots for volatile tables will help determine not only the frequency with which to analyze, but also the best composition for sets of tables to be analyzed during each scheduled run.

In order to measure a performance change, or to address a performance problem in a particular module, SQL Trace can be used to identify candidate tables. (Incidentally, SQL Trace will also show that some recursive SQL contains hints; however, **you should never analyze the SYS schema**.) If you use this approach, all tables in a SQL*Plus statement will need to be analyzed. If problems ensue, then the statistics can be deleted immediately by using the `ANALYZE <table> DELETE STATISTICS` command. Both 10.7 and R11 contain a smattering of hints, so analyzing all modules may be beneficial. From the SQL command prompt in 7.3 or higher, you should alter the session to set the `sort_area_size` to 5MB or more and set the `db_file_multiblock_read_count` to 32 (Oracle will reduce this value if it is too high) when, for example, rebuilding indexes, analyzing schemas or large tables. In Release 11i, you must set the mandatory CBO `init.ora` parameters and analyze all the schemas. Further, `ANALYZE` is no longer supported, and `ADXANLYZ.sql` will use the FND_STATS package instead.

### 6.5 Monitoring Growth

Consider using the "MONITORING" option to help understand the growth pattern of individual tables. Monitoring has a very small overhead and can be easily set on a table using the SQL*Plus command `ALTER TABLE <tablename> MONITORING`. Once enabled, the server maintains information about how many rows are affected by a modification (insert, update, delete, or direct load). Periodically (every 3 hours or a clean shutdown), SMON incorporates the information into the data dictionary. These changes are visible through the `DBA_TAB_MODIFICATIONS` view, which then provides a very simple method of collating information on high-volume or transaction Applications tables.

In order to query the information back, use the following query:

```sql
SELECT table_owner, table_name, inserts, updates, deletes, timestamp, truncated
FROM sys.dba_tab_modifications;
```

The information in this table is cumulative from the time that you enable monitoring. To make this information more useful in an Applications environment, copy the information into an archive table on, different parts of the business lifecycle or perhaps on a monthly basis. This will enable you to track the data volume changes over those periods. Note that using the 'NOMONITORING' option will reset all the information from the `DBA_TAB_MODIFICATIONS` view.

You can check which tables monitoring has been set on using the SQL*Plus query:

```sql
SELECT table_name, monitoring
FROM dba_tables;
```
### 6.6 Block Size And Related Setup

For best performance, all the data to satisfy a query such as a full table scan would exist in a single block, or consecutive blocks that have been read into the read-ahead I/O buffer during a sequential read operation. This implies that ideally the database block size should be equal to, or a multiple of, the operating system block size.

There is a tremendous amount of speculation regarding block size and there are many factors involved with data storage. Many documents discuss row chaining, which occurs when a record is too big to fit into a single block. Excessive row chaining indicates that the block size is too small, but this is not the main reason why such a dramatic beneficial effect occurs when increasing the block size for Oracle Applications.

Changing the block size from 2KB to 8KB can increase performance by 40% for batch programs and large data-sort queries, and by 4% to 5% for light data-sort queries. Why do we get these improvements? Although there is a reduction in chaining of large data rows, the overriding factor is table reorganization, and the reduction in height of B-Tree indexes and associated searches, which are used with almost every database access. There are numerous examples where a customer has rebuilt the database and reaped the benefit. In Release 11i, there are only two choices, 8KB and 4KB, and for nearly all customers it will be 8KB.

An inextricably linked topic is the maximum number of database blocks read during each I/O read-operation, as defined by the `db_file_multiblock_read_count` initialization parameter. Often this is set incorrectly and there appears to be confusion regarding the small, medium, and large settings in the `init.ora` file. If this parameter is inadvertently set too high, Oracle will reset the value and use the value from `V$PARAMETER`. The maximum number of blocks that can be read in one I/O request is defined by `max_io_size/db_block_size`. Setting the value correctly based on the `max_io_size` returns a performance improvement for all aspects of Applications sequential read operations.

Another point regarding `max_io_size` is the ideal system stripe size. Consider a system with a system read capability of 64KB, an 8KB block size and `db_file_multiblock_read_count` of 8. This means that Oracle would request 64KB in a single I/O read-operation. This being the case, using a stripe size of 16KB would perform 4 system I/Os for each Oracle sequential read request. It follows that not only should `db_file_multiblock_read_count × db_block_size = max_io_size` but also the stripe size should be equal to, or multiples of `max_io_size`.

### 6.7 Package Pinning Strategy

Each Application has several packages that can be loaded into the shared SQL and PL/SQL memory areas. It can take a considerable time to load and parse a large package; having packages resident in memory yields the best performance for OLTP users. The time to load and parse packages is proportional to their size.

Theoretically, pinning all packages improves performance; once pinned, they cannot be swapped out of memory unless they are explicitly unpinned, at which time they will be normally aged out of the buffer. For this reason you should also have an unpinning strategy. Pinning all packages is not always possible due to the amount of available memory.

---

In Release 11i, `db_file_multiblock_read_count` must be set to 8, but can be increased for index rebuilds, or when analyzing with `FND_STATS`.

Always pin the core SYS and APPS packages, as an absolute minimum.
It is not necessary to pin small infrequently used packages or those used by long-running batch processes or reports, as these will be implicitly pinned for the duration of the task. However, if packages are not explicitly pinned, very large packages may not be able to load due to insufficient or fragmented memory within the shared buffer pool. For Release 11i, the worst case scenario is 150MB.

Importantly, when the amount of shared pool memory falls below a critical level, the server has to dedicate resources to managing the limited space available. When a very large package is referenced, a huge amount of flushing and buffer coalescing can take place to create a contiguous area of free memory known as a ‘chunk’. Prior to Oracle 7.3, if there were no chunks large enough, the package would fail to load; an ORA-4031 error would occur, and the failure would be recorded in the X$KSMRLRU table, as well as the database alert log.

Pinning rarely used packages is very wasteful of the shared memory area. Suitable candidates are large packages that are used infrequently, meaning they have to be constantly reloaded, or perhaps those with the highest number of executions in the table V$DB_OBJECT_CACHE. Fragmentation is reduced when packages are pinned in descending size order (determined from DBA_OBJECTS) - this is vital for Applications on Oracle7. Note that in Oracle8i, V$DB_OBJECT_CACHE also reports PL/SQL, but the execution count is no longer incremented. This has been reintroduced in 8.1.7; you may want to check if a backport is available for your platform.

### 6.8 Shared Pool Reserved Allocation

The shared_pool_reserved_size specifies the amount of the shared pool to be reserved for SQL statements and packages requiring large memory allocations. It should be set to approximately 10% of the shared_pool_size, subject to a maximum of 50MB. The exact size depends on the product mix; you should monitor utilization using V$SHARED_POOL_RESERVED. In order that not every request is automatically obtained from this sub-pool, setting the shared_pool_reserved_min_alloc parameter prevents small statements below a certain size being allocated space.

When using Applications with Oracle7 and 8, the default value is 5000 bytes (with a minimum of 5000 bytes) and should not be changed for most installations. In 11i/8i the default of 4400 bytes (with a minimum of 4000 bytes) is set by the undocumented _shared_pool_reserved_min_alloc initialization parameter. There are many PL/SQL allocations of around 4300 bytes, which is greater than the standard chunk size. When loading PL/SQL of below 4KB, the SGA may displace several objects, whereas loading PL/SQL of 4300 bytes will displace as many as 50 objects and the extra work this places on the database is significant. Setting the _shared_pool_reserved_size to less than 4300 will obviate the associated performance overhead.

### 6.9 Latching

If you are encountering a performance problem which only shows after a period of database activity, and is partially cleared by flushing of the shared pool and completely cleared by restarting the database, then it is likely that you are experiencing contention on the library cache latch or shared pool latch. This can be verified by obtaining a Statspack report and checking the library cache and shared pool latch waits.
In Oracle Applications, this high-impact problem has been identified as:

1. **Related to views**
   Selecting from views uses too much shared cursor memory. On most platforms this bug (724620) is fixed in Oracle Release 7.3.4.4, 8.0.4.3 and 8.0.5.2.

2. **Non-sharable cursors and parsing of bind variables**
   This affects many aspects of Oracle Applications, fixes include PL/SQL packages, procedures, Oracle Forms and Reports. To alleviate these issues, a number of Applications patches must be applied.

If you are not able to upgrade immediately to the appropriate RDBMS version or apply the Applications patches, then changing the database initialization parameters might alleviate the problem. As there are several permutations of database setup and factors that contribute to the latching problem, it is not possible to provide a standard set of rules to correct the problem. The following two factors play a very important part:

- **CURSOR_SPACE_FOR_TIME** When set to TRUE, this pins SQL areas in the shared context area and prevents the deallocation of private SQL areas until the Application cursors are closed. When set to FALSE, both the shared SQL area and the library cache need to be checked when attempting to find a SQL statement. This parameter has not been established as having a major effect on Oracle Applications performance and should generally be set to FALSE.

  If you decide to set CURSOR_SPACE_FOR_TIME to TRUE you should increase the size of the shared pool by 50%, otherwise this will compound the latching problem. In other words, you will need to allocate a shared pool large enough to prevent parse errors, while ensuring that there is sufficient free memory to prevent extensive swapping. In order to minimize the shared pool contention, you should also ensure that ROW_CACHE_CURSORS is set to 200 for everyday usage and session_cached_cursors at 100. The session_cached_cursors parameter increases the number of SQL statements that can be moved, when parsed repeatedly, into the session cursor cache. Subsequent parse calls will find the cursor in the cache and need not reopen the cursor; this is highly beneficial for users who frequently switch between the same group of forms.

- **SHARED_POOL_RESERVED_SIZE** When the shared pool fragments or if the 'small chunk freelist' is growing excessively, flushing the shared pool will cause a short-term peak in CPU and I/O resources, but should provide relief from contention. The frequency of flushing depends on the system usage, but if it is necessary to flush during the working day, it will help to pin commonly used packages, functions, and triggers that execute more than 100 times. This may, however, result in insufficient deallocation, which will then require manual unpinning or in the worst case, a database restart. The shared_pool_reserved_size and shared_pool_reserved_min_alloc parameters were discussed earlier in the Package Pinning Strategy section.
6.10 Hit Ratio

Using the buffer cache hit-ratio is now a dated method of tuning, though the buffer cache activity remains key to general performance meaning that you cannot ignore it. Having said that, users still appear to be most satisfied with Applications performance when the hit ratio is between 94% and 97%. Some theorists might discount this upper limit and while this has not been proven using real-world experience, some systems provide acceptable performance with a hit ratio outside this range. When this happens, there is usually some compensatory factor such as a large amount of spare CPU, or exceptionally well-balanced system resource utilization.

The absolute minimum size of the buffer cache is 5000 blocks. For Oracle Applications the recommended minimum is 10,000-20,000 for a typical system (100-300 concurrent users), 30,000 for a large system (500 concurrent users), and up to 50,000 (500+ concurrent users) for the very largest systems. Greater than 50,000 will simply burn CPU and waste space, especially if bad SQL is still allowed to flood the buffer cache.

The buffer cache hit-ratio is tuned by changing the db_block_buffers setting. An estimate of the performance increase for a specific number of buffers can be gained by setting the db_block_lru_extended_statistics initialization parameter to the number of db_block_buffers you are thinking of adding. For example set db_block_lru_extended_statistics = 500 and restart the database. The statistics are computed and stored in X$KCBRBH. A similar approach can be used to determine a reduction of the hit-ratio. When a non-zero value is used, there will be a noticeable drop in performance; hence this should not ordinarily be set on your production system.

6.11 Managing The System Load

There are a huge scope for monitoring, and things that should be monitored that will affect the database. Many of these topics are included in this paper. Statspack is invaluable as a tuning tool and is one of the main tools that will show high-load and literal SQL. In addition, there are several other areas that need attention, which include pinning packages, and formulating and implementing a concurrent processing strategy.

All of these are relatively long term, for example, the best unit of work for Statspack is one hour. Occasionally, you may need to be able to quickly respond and identify processes that are using excessive machine resources. For example, a handful of users running long open queries is generally enough to reduce the performance for every Applications user. You should add the following query to your standard toolkit and have it at hand for this type of emergency situation.
The following query is mainly intended to identify online users as batch jobs can be monitored via the Concurrent Manager:

```
select C.SID, C.SERIAL# SER, C.MODULE, 
    SUBSTR(C.PROGRAM,1,10) PROGRAM, 
    C.LAST_CALL_ET LAST_CALL, C.SQL_HASH_VALUE, 
    D.SPID, C.PROCESS 
from V$SESSION C, V$PROCESS D 
where C.STATUS='ACTIVE' -- the session is running in the database 
    and C.LAST_CALL_ET>300  -- has been running for over 300 seconds 
    and C.PADDR = D.ADDR 
    and C.LOCKWAIT is null -- not waiting on a (row) lock 
    and (C.PROGRAM like 'f60%' 
        or C.PROGRAM like 'sqlplus%')  -- only interesting programs 
/
```

<table>
<thead>
<tr>
<th>SID</th>
<th>SER</th>
<th>MODULE</th>
<th>PROGRAM</th>
<th>LAST_CALL</th>
<th>SQL_HASH_VALUE</th>
<th>SPID</th>
<th>PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>3</td>
<td>PODAMGR@br</td>
<td>56</td>
<td>1425819161</td>
<td>5984</td>
<td>5826</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>2</td>
<td>INVTMRPM@b</td>
<td>31</td>
<td>1425819161</td>
<td>6048</td>
<td>5839</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>121</td>
<td>ARXCWMAI f60runm@br</td>
<td>17775</td>
<td>2531375889</td>
<td>16465</td>
<td>16413</td>
<td></td>
</tr>
</tbody>
</table>

If a long running operations calls recursive SQL (e.g. PL/SQL function in SQL statement), the SQL_HASH_VALUE will be constantly changing. This doesn't mean that there isn't a top-level user query, which is performing badly, just that it will be harder to track down.
7 TUNING THE SERVER

Server upgrades are generally very expensive and tuning Oracle can resolve many problems. For example, changing the database architecture to comply with the Oracle Flexible Architecture methodology can resolve problems by moving the database files to even out the disk activity. It is essential to know the memory, disk types, configuration, file layout, and system I/O amongst other things. The database and applications need to be tuned before it is possible to determine whether the servers have sufficient capacity for the implementation. Advice from the Oracle platform specific manuals is invaluable. Rather than deal with standard tuning issues, some hot topics are discussed.

7.1 DISK I/O: OFA VS RAW PARTITIONS

Oracle Flexible Architecture (OFA) defines a default directory structure and file system layout for database installations on UNIX and Windows NT. It is generally accepted that indexes and tables need to be separated into different tablespaces. If they are all located on one disk, the head would be constantly moving between the two, and so locating the tablespaces on separate disks minimizes disk head movement. A useful extension of this approach, which is commonly overlooked, is that it is better to ensure the disks are on separate controllers, enabling the operating system to concurrently perform input/output operations through a number of channels. Further, if you create large reports, separating the .log and .out directories will also obviate drive contention.

The increasing use of RAID arrays and logical volume managers does not really make this advice obsolete. While these provide good automatic load balancing across disk devices, some control is lost, and if a table and its associated index tablespace are located in the same RAID disk, an indexed access to a table may produce activity on the same physical disk. The most important point for Oracle Applications is to use several mount points with RAID arrays to reduce I/O queuing. Always check the read and write times in the Statspack report, which should never be more than a few milliseconds - typical PC hard disk access time are around 10-20ms.

Raw devices are generally only required for OPS and RAC, though some manufacturers are demonstrating these types of systems using standard UNIX file systems. Some Applications customers have been pushed down the route of converting to raw partitions to overcome disk I/O bottlenecks. Theoretically, using raw devices instead of UNIX file buffering improves disk I/O and some customers that have performed the conversion have noticed a gain of around 10%-15%. However, a database export and import may have achieved the same return (as row chaining is eliminated and the indexes are rebuilt and balanced). Raw partitions also have the disadvantage that they are significantly more difficult to administer, though having as many raw sections as there are data files/tablespaces can make the changes easier.

Although there are some situations in which raw devices are the correct choice, due to the administration overhead, the only reason for exclusively using raw devices should be that disk I/O is the only remaining system performance bottleneck that cannot be resolved by tuning the database, server, or application. In our experience, they have never been required for satisfactory Applications performance.
7.2 MEMORY BOTTLENECKS

Since memory access is much faster than disk access, it is desirable for data requests to be satisfied by access to memory rather than access to disk. Tuning memory allocation and management involve distributing available memory to Oracle memory structures. This is done by correctly sizing the SGA and optimizing the number of database buffers.

A word of warning. If increasing the amount of Oracle memory leaves insufficient for the operating system, the SGA may be paged to disk. If this occurs, trace files will still report logical reads although the operating system has had to fetch the page from disk. On a system with limited memory, balance increasing the size of the buffers with the increased likelihood of swapping. On a database server not running MTS, the SGA should never exceed half the physical memory.

7.3 PAGING AND SWAPPING BOTTLENECKS

Insufficient paging space can dramatically affect a system’s performance. The system can demonstrate such problems as slow response times, failure to spawn sub-processes, or complete system hangs. Most UNIX systems should have total swap space between two and four times their physical memory size. The minimum is twice physical memory; Oracle Applications requires paging space to be at least 3-4 times that of the amount of physical memory.

7.4 CPU BOTTLENECKS

If the system is found to have a CPU bottleneck, the workload should be examined to determine if any of the work could be moved to off-peak periods. For example, batch processes or non-critical reports could be run overnight. By moving less critical jobs to periods of the day when the system is lightly loaded, you can improve throughput for users who run during those peak-demand periods. If there is no scope for rearranging the workload, investigate and tune the applications code before considering a hardware upgrade.
8 TUNING THE SQL ACCESS PATHS

When dealing with system performance, application SQL tuning should be considered a special case when either processes do not complete within a required period, or place a significant load on the system to the detriment of other activity. Tuning activity in this area generally deals with isolated module problems. The two approaches to identifying expensive SQL are:

- **Top-down** By reviewing the high-load SQL area in the Statspack report
- **Bottom-up** By tracing a specific module

8.1 TRACING MODULES

In Oracle Applications, it is imperative to compare the sum of the “elapsed” totals to the real-world run time for the transaction or process. For example, if a process takes one day but the trace file only shows one hour, optimizing every statement in the trace file will still mean that the process will take 23 hours! Historically, problems of this nature are long running and have sometimes taken several months to resolve. If the times do not match, TKProf will not have converted all of the information from the raw trace file.

There are two reasons why the TKProf output may not be representative of the process. First, the problem may not be in the database; time may be spent elsewhere, such as on the network. Second, TKProf may not convert all the information from the raw trace file. PL/SQL procedures are a prime example of cases in which only the parse time is shown and not the individual times for each of the statements in the package. Conversely, if the totals are representative of the “real-world run time,” then you have at least greatly narrowed the investigative focus, as the problem is definitely in the database.

8.2 ENCAPSULATING SQL

In order to simulate conditions in the form, you need to use bind variables from within SQL*Plus. Consider using the following type of SQL*Plus construct, which overcomes this problem and uses bind variables:

```sql
VARIABLE sql_output REFCURSOR
VARIABLE b1 NUMBER
ALTER SESSION SET SQL_TRACE TRUE;
Begin
  :b1 :=51829;
  OPEN :sql_output FOR
  SELECT NAME, H.STATUS, H.CONTROL_TOTAL, L.DESCRIPTION
  FROM GL_JE_HEADERS H, GL_JE_LINES L
  WHERE H.JE_HEADER_ID=L.JE_HEADER_ID
  AND H.JE_HEADER_ID = :B1;
End;
ALTER SESSION SET SQL_TRACE FALSE;
print sql_output
```

If the trace and real-world times do not match, discard the TKProf file and use the raw trace file instead.
Remember that the performance and execution plan for a particular SQL statement on a test system may not be representative of the performance of the same execution plan on a production system. This is because of differing data volumes.

Assuming the SQL code is efficiently written, the next step is to look at anomalous data volumetrics and data distributions identified during the investigation or audit stage. For example, 1,000,000 customers would be considered a large number in any implementation. Additional indexes may help, but adding indexes into Oracle Applications is especially difficult, as you need to identify other modules that are affected in order to test them. There are several papers and books that discuss standard indexing practice, but it is worth remembering that not all full table scans are noteworthy. For example, scans of very small tables (smaller than the _small_table_threshold initialization parameter), or those that are read in a single system I/O read-operation should not be considered as candidates for optimization. The UTLOIDXS.sql script may be used to analyze possible new candidate columns for indexing. In 11i, FND_STATS can be used to look for candidate histogram columns.

### 8.3 Using Event 10046

Trace may also be enabled for a forms session or report, within a stored procedure, in a concurrent program, or for the entire instance. For tracing to be effective, setting the timed_statistics parameter to TRUE is highly recommended, as Oracle Development usually requests this information.

For more advanced troubleshooting, it is possible to increase the level of information recorded in the trace file by setting the event 10046. This can be set to four different levels, which are shown in the following table. For SQL tuning in Applications, it is commonly used to trace bind values, which can then be used to determine whether data skew is causing performance problems.

<table>
<thead>
<tr>
<th>Included Information</th>
<th>Event 10046 Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>null or 1: Same as standard SQL_TRACE functionality</td>
<td>As Level 1 + Bind variable information</td>
</tr>
<tr>
<td>4: As Level 1 + Bind variable information</td>
<td>As Level 1 + Session Wait Information (can be used to identify full table scans)</td>
</tr>
<tr>
<td>8: As Level 1 + Session Wait Information (can be used to identify full table scans)</td>
<td>As Level 1 + Bind variable + Session Wait Information</td>
</tr>
</tbody>
</table>

Some examples of setting the event for a SQL session are as follows:

- alter session set events '10046 trace name context forever, level 8';
- alter session set events '10046 trace name context off';
In Applications, you would typically enable event 10046 using the DBMS_SUPPORT package to target specific sessions. Alternatively, in Applications, any valid SQL statement can be used in System: Initialization SQL Statement - custom profile option. For example, to write the bind variable values to the trace file using, you would set the profile option to the following value:

```
'ALTER SESSION SET EVENTS ='''''' || ' 10046 TRACE NAME CONTEXT FOREVER, LEVEL 4' || ''''''
```

You can use this type of approach anywhere, for example, you could set the event in a report by using the BEFORE-REPORT trigger as follows:

```
SRW.DO_SQL('alter session set event = "10046 trace name context forever level 4"').
```

8.4 IDENTIFYING LITERAL SQL

This is a very important area for Applications as literal (unshared) SQL can severely limit scalability and throughput. The cost of parsing a new SQL statement is expensive, in terms of both CPU requirements and the number of times the library cache and shared pool latches may need to be acquired and released. It has been proved that converting literal SQL to use bind variables in commonly occurring statements can help eliminate problems with shared pool performance, and greatly improve scalability.

SQL load is divided into Execution load and Shared Pool load.

1. Execution Load

- SQL statements with high buffer_gets/disk_reads per execution. The traditional, poorly tuned SQL statement.

- SQL statements with a high number of executions, and a high total for buffer_gets/disk reads. The individual statement load is not high (buffer_gets/execution) but the cumulative effect is substantial - the standard `SELECT` statement for profile options is a good example of this.

2. Shared Pool Load

- Literal SQL statements. We are concerned with statements that use literals with a large data set - e.g. order no. Narrow data sets such as 'Yes'/‘No’ are of much less concern.

A SQL statement cannot be shared if it uses literal predicates rather than bind variables, or where the value of the literal differs between subsequent executions of the statement. For example, WHERE `name = '104 M1 Inventory 340883'` should be re-written as `WHERE name =:b1`. However, where there is a narrow data set, such as ORG_ID, the constants will always be the same so the statement is sharable.

A qualification to this is that the cost-based optimizer performs well when statements use literals in their predicates. Consider the following statement:

```
SELECT name 
FROM   gl_je_headers 
WHERE  budget_version_id < 3000;
```

If there were an index on the BUDGET_VERSION_ID column, the CBO would use histogram statistics to decide if it would be fastest to do a full table scan or use an index scan.
If a bind variable is used (:b1), the CBO has no way of knowing the percentage of rows above or below it. As a rule of thumb, bind variables typically provide best overall system performance for online Applications users where you want statements that are scalable and sharable, where the plans are predictable and do not frequently change. Literal values may be better for batch processing where the statements are run infrequently, and you want to give the cost-based optimizer the best possible information so that it can accurately cost the use of an index, rather than having to use default selectivity.

As the amount of CPU spent on parsing is typically only a small percentage of that used to execute each statement, it is more important to give the optimizer as much information as possible, rather than minimize the parse time. This is why the Applications FND_STATS package maintains the histograms for certain predefined columns.

It is unlikely that you will be able to eliminate all literal SQL, but you can reduce the need for it when writing custom code. This can be achieved by restricting its use to narrow data sets and columns for which the FND_STATS package created histograms. You can use the following query to identify candidate literal statements that are not being shared:

```sql
SELECT SUBSTR(sql_text,1,100) "SQL", count(*), SUM(executions) "TotExecs"
FROM v$sqlarea
WHERE executions = 1
GROUP BY SUBSTR(sql_text,1,100)
HAVING count(*) > 30
ORDER BY 2;
```

This identifies statements with the same leading 100 characters for which there are at least 30 different occurrences, but which have only been executed once. In other words, this query highlights a possible literal SQL statement.

### 8.5 Analyzing Literal SQL

When analyzing the statements in the shared pool, you should start by ensuring that they are literal statements i.e. that the statements are identical except for the literal values. Consider the following example:

```sql
SELECT f.name, f.folder_id,
       u.user_name,
       DECODE(f.autoquery_flag, 'Y','*') autoquery,
       DECODE(f.public_flag, 'Y', '*') avail_public
FROM fnd_folders f, fnd_user u
WHERE f.object = 'OE_ORDERS'
AND f.language = 'US'
AND ((f.created_by = 9382)
     OR (f.public_flag = 'Y'))
AND f.created_by = u.user_id(+)
ORDER BY f.name
```
If this is the case look at the type of literals - in this example 'OE_ORDERS' and 'US'. You always need to establish exactly what varies between the statements, such as the language or an object? You will always have some literal SQL statements, and so you need to work out the subset of these that are significant.

The literals used in this example are not an issue as they reference very narrow data sets. Instead, the main problem to look for is conditions such as 'ORDER_NO = 123456' as there can be many millions of orders, and the number of SQL statements is effectively unbounded. Note that there will be many hundreds of similar statements for this sort of case.

You also need to consider shared memory usage, such as when a SQL statement uses an excessive amount of memory (typically > 1MB). This is normally captured during development testing but some cases may slip through with customizations. As you can imagine, the worst case is literal SQL where each version uses a large amount of shared memory. For example, 100 statements each using 1MB each will quickly eat up the shared pool.

In general by eliminating the unbounded literal SQL statements, the memory problems will take care of themselves. The future direction of the Oracle Server is to automatically convert statements to be sharable by replacing literal values, such as a batch name of "104 M1 Inventory 34083," with bind variables. Although this has been introduced in a limited form in Oracle8i, Applications Release 11i does not use this feature. Finally, when investigating your instance remember that as a general rule, bind variables are best for OLTP, whereas literal values may be better for batch.

8.6 FINDING DYNAMIC FORM SQL

Occasionally, when dealing with development issues or support questions, it is useful to find the dynamic SQL being executed when a form performs a query without having to use trace. Since Release 11, you can find the current query that is being executed by a form using either Examine | SYSTEM: LAST_QUERY.

A slightly different way of doing this, which helps tremendously when customizing Applications as it maps fields and columns, is to crash the form. This works in all Applications versions, and not only lists the tables, columns, constants, and ordering, but also includes restrictions (including folder conditions) on records that are queried.

![Finding the Dynamic SQL statement](image)
The procedure is very simple:
1. Put the form into query mode (Query Enter)
2. Enter #XXX in any ‘database’ field
3. Execute the query - This will create an error
4. Acknowledge the error, cancel the query, and select Help/View Database Error
5. Review the database error, which will show the query being executed at the time the form “crashed”

The select statement lists all form fields being retrieved from the database. The where-clause conditions apply regardless of whether an “open query” was executed or any additional query criteria were used. For example, on the Purchase Order lines Zone, you would find that cancelled lines are not displayed, which makes answering support questions very simple.

8.7 High Water Marks

Applications interface tables are prone to index stagnation and High Water Mark (HWM) problems. When data is written to a table, the highest block used, known as the HWM, is recorded in the header block. When data is loaded, the HWM is used to determine the start point for a data load and records are loaded above this mark, even if the previous set of records has been deleted. The high-water mark is not used during index scans but when a full table scan occurs, the whole table is read up to the HWM regardless of the number of records in the table. Running a “SELECT COUNT(*)” query will typically take a long time to show few or zero records.

If you have analyzed the table, the number of empty data-blocks is recorded in DBA_TABLES. Alternatively, the DBMS_SPACE package available in Oracle 7.3 and later can be used. Clearing the problem is straightforward - use the TRUNCATE command when the table is empty.
This is a very important problem area for Applications and should be the first thing to be checked when an interface becomes slower over time. The performance return can be huge. One notable scan of GL_INTERFACE took over 20 minutes to return a count of 0 records. This was found when average size postings started taking nearly 40 hours to complete. The posting time reduced to about 2 hours after the problem was rectified.

**8.8  EXPENSIVE SQL**

The first stage of any Application performance analysis has to be to check that the code is efficient. The most expensive system SQL activity can be determined using BUFFER GETS/EXECUTIONS (for CPU) or DISK READS/EXECUTIONS (for disk I/O) in V$SQLAREA. This only contains the first 1000 characters of the SQL statement and needs to be linked to V$SQLTEXT using the HASH_VALUE and ADDRESS in order to retrieve the complete statement. Note that from 11.0.3, the form or report name is stored in the MODULE column of V$SQL and a backport is available for 10.7, as patch 1163783.

**8.9  CUSTOMIZED VIEWS**

Quite often new views are created for customized work or to simplify complicated expressions used in Oracle Data Query. It is very important when creating views that will be heavily used to ensure that they are based on base-tables and not on other views. This type of tuning has returned 30% performance increase across the parts of the system where it was referenced. Further, many developers forget to consider the cost-based approach when writing SQL to access the views.
9 TUNING THE NETWORK

One of the biggest problems facing network managers today is being able to size network links to minimize congestion. Taking the simplest case of only Oracle Applications users, you will find that they have fairly set work patterns and tend to have well defined peak periods. However, it is very difficult to predict the exact specification for other types of Applications traffic such as downloading or printing reports, email, etc. as this type of traffic varies not only throughout the day, but also through the monthly business cycle. As you will see, increasing bandwidth does not always provide a satisfactory resolution to support mission-critical systems.

9.1 ISN'T BANDWIDTH ENOUGH?

Oracle Applications network traffic has considerably reduced since the introduction of the internet computing architecture. However, changes in other technologies mean that many companies are still seeing bandwidth requirements increasing. Changes in technology such as emails with large attachments, audio and video streaming, and video conferencing all need a high data throughput and place ever increasing demands on networks that still need to support mission-critical systems. Some of the types of low priority traffic that will affect the performance of your mission critical applications include:

- Large file downloads from corporate and external web sites
- Email synchronization
- Downloading large email attachments
- MP3 uploads and downloads
- RealPlayer and other types of demanding voice and streaming traffic

Regardless of the type of traffic, it is clear that large data transfers can be very detrimental and mission-critical applications will suffer while less important traffic is allowed to dominate the network. Any type of network congestion will undermine Oracle Applications performance. Not only does this reduce productivity but it also increases the number of complaints, which in turn places additional strain on the support infrastructure.

When investigating performance, many incorrectly focus all tuning efforts on the database and rarely consider the network, even though there is a proliferation of network management software and tools available. When a network problem is identified, the trend appears to assume that the majority is legitimate traffic supporting mission critical applications. As a result, network managers are spending an increasing percentage of their budget on bandwidth upgrades in an attempt to resolve the performance problems.
With the advent of network switches and 100 MB network cards, bandwidth problems in LAN environments are becoming less common. WAN links, however, continue to be problematic because of their cost and relatively low bandwidth. The Global Single Instance deployment model used by Oracle Applications places additional demands on what are traditionally low bandwidth WAN links. The following picture shows an example of a network utilization graph.

The three screenshots in this section are selected Packeteer shaper management screens. Their inclusion is not intended as a product endorsement. There are several shapers available and you should investigate the merits of each.

This shows a network that is running at peak capacity when the users return from their lunch break. You might think that the solution to this problem is to increase the bandwidth. However, increasing the size of the network link does not guarantee the availability of adequate bandwidth for mission-critical applications. For example, some customers have doubled the capacity of a link, usually at considerable cost, only to find that the Application performance problems persist. Not only does this demonstrate that a different approach is required, but also that rising network costs now mean that it is becoming more important to maximize the performance of existing WAN links. Depending on the network infrastructure, there are two main approaches that are available:

- Packet Shapers
- Quality of Service

One of the challenges you will face when investigating these technologies is that vendors sometimes use the same term to describe different things and so it is essential that you fully understand each.
9.2 Packet Shapers

A packet shaper controls the packet stream for a number of different traffic classes, which in turn can control the priority and amount of bandwidth allocated to each. Their main purpose is to guarantee mission critical traffic the bandwidth it requires by using techniques such as TCP rate control or varying the sliding window protocol to slow down and limit less urgent traffic. The main advantage of packet shapers is that they can be used to prevent specific types of traffic from monopolizing a network link, meaning that you can deliver predictable performance for Oracle Applications running over a WAN.

The following picture shows an example of traffic on a local segment where it is easy to identify non-critical and unsanctioned traffic. If you purchase bandwidth from a service provider it is easy to calculate how much each traffic class is costing your company. You should of course start by focusing your tuning effort on non-essential traffic.

Regardless of whether a packet shaper is implemented as an in-line network device or as a software program, most will allow you to control traffic flow based on the type of class, protocols, subnets and so on. The ability to identifying and categorizing the types of competing network traffic is the first step toward solving performance problems. This next picture shows how traffic flows can be shaped and controlled.
Packet Shaping

You would normally have to identify the specific type of traffic relating to the application that you want to control and create corresponding bandwidth-allocation rules. However, some shapers are more intelligent and can classify the traffic by application. This means that controlling the bandwidth changes from finding out the type of traffic, which incidentally may be used across a number of applications, to being able to specify the priority for each particular application. In other words you do not need to be a network expert to effectively set up this type of shaper. Some packet shapers have expert features that simply allows you to nominate Oracle Applications as a high priority mission-critical application and will manage the bandwidth automatically from there.

Traffic Analysis
9.2.1 Control Policies

There are different ways of limiting, or conversely, guaranteeing bandwidth for types of network traffic or specific applications. The names, types and features of each policy vary in the way that they are defined and implemented across vendors. For this reason, you should always check your understanding of specific features that you intend to use with each vendor, rather than making assumptions about how they may work. Some of the common types of packet shaper policies are as follows:

- **A Rate Policy** This limits each individual traffic class. You need to establish how this works on each particular shaper as some specify a rate limit on a per-session basis whereas some specify a rate limit for the total traffic flow. For example, an FTP rate limit of 10Kb on a per-session basis will mean that 12 users could still flood a 128Kb link, whereas a total link limit would mean each user would get around 830 bytes each and you would still have 118Kb available for other applications.

- **A Partition (Or dedicated stream policy)** Both of these approaches effectively create a virtual pipe of a specific size for one or more specified traffic classes. On some shapers, you can set a policy that will allow it to burst within a specified limit (which may be the entire bandwidth of the link), or alternatively, you can allow other types of traffic to use any unused bandwidth.

- **A Priorities Policy** This simply prioritizes particular traffic classes or applications access.

9.2.2 What to Look for in Packet Shapers

In order to be able to implement an effective policy, you need to understand exactly what is happening on your network. Some shapers will do this automatically whenever a new traffic stream is detected. By keeping network statistics, you should be able to spot trends and changes on the network before they become a serious problem and impact the business. This will also provide you with a mechanism, which will allow you to compare actual and contracted performance. This will help make service-level agreements (SLAs) more meaningful. It will also mean that you can quickly react to new types of traffic.

9.3 Quality of Service

Quality of Service (QoS) defines a set of features (and tools), often implemented within routers, that allows you to classify and differentiate the service levels provided to various types of network traffic. This enables you to provide a better and more predictable data delivery service for selected types of network traffic. It is intended to overcome the routing delays and lost packets that are atypical of erratic periods of uncontrolled network congestion that are so detrimental to mission critical applications.
Once you have classified the type of packets, you can use QoS to assign an appropriate traffic management policy such as congestion management, bandwidth allocation, and delay bounds for each traffic class. QoS cannot create additionally bandwidth, but it does enable you to limit bandwidth for non-critical traffic (such as FTP file transfers) to provide a consistent response for critical traffic during periods of network congestion. Typical features include shaping network traffic and setting traffic priorities across the network to manage congestion, which in turn will improve the loss characteristics and improve efficiency. This means that in an ideal world, QoS will expedite the handling of mission-critical applications, such as Oracle Applications, while sharing network resources with non-critical applications. However, it is important to remember that QoS is an end-to-end solution and can only be as good as the weakest link in the chain.

It is clear that QoS designated high-priority applications traffic must never be allowed to completely flood the network and exclude all lower priority traffic. Over the past couple of years, the number of methods and protocols for setting quality of service (QoS) in network equipment has increased dramatically. Advanced queuing algorithms, traffic shaping, and other features such as access-list filtering, can make the process of choosing a QoS strategy for your network a much more daunting task.

QoS is normally implemented by specifying policies for each network device. It is essential that you can differentiate between the different types of traffic in order to be able to apply a policy to shape the traffic flow and perform intelligent queuing. Understanding your set of applications and their business criticality is essential to creating effective policies that will improve performance across the entire network. For example, a policy that guarantees bandwidth to a particular application is fairly easy to understand. However, this would not necessarily resolve problems for time-sensitive applications. Instead prioritizing this type of traffic would be more effective. You might need to use a combination of policies and so for this reason, you will need to understand all the techniques available.

To complicate matters further, the function of a router (and hence QoS policy) will change depending on its placement and intended operation. For example, Cisco makes the distinction between edge routers that perform packet classification and admission control, and backbone routers that perform congestion management and avoidance.
Most QoS managers will enable you to define the following policies:

- **Packet Classification**  This assigns an IP precedence to a traffic flow, which is known as coloring. This is applied to all traffic on its first encounter with a network device. Coloring is a very common approach to use, though you would normally only color time-sensitive applications. Committed Access Rate (CAR) is one of its main features, which uses specific bits in the IP header to support packet classification. For example, you would set particular bits to specify high priority for all packets with a destination of the Oracle Applications server.

- **Rate Limiting**  This controls the rate of traffic allowed to enter or exit a network device. CAR includes a rate-limiting feature that specifies what should happen to the traffic when it equals, or exceeds, a specified rate limit. This type of policy usually drops all packets that exceed the rate limit without trying to smooth the flow. You could, for example, set an Internet browsing traffic limit of 10% for HTTP traffic.

- **Packet Shaping**  This is a more sophisticated approach to limit the bandwidth of the entire traffic flow. It also differs as shaping policies only work on outbound traffic, whereas rate-limiting policies can work on both inbound and outbound traffic.

- **Queuing**  In addition to managing the outbound queue, a queuing policy specifies which classes of network packets should be dropped for congestion management and avoidance. When a packet is about to be placed on the outbound queue, it is checked to see if a coloring policy was applied, which will then affect how the packet is queued. A shaping or rate limiting policy will affect the amount of available bandwidth to the type of packet stream.

### 9.3.1 Queuing Algorithms and Congestion Management

One way to control congestion is to use a queuing algorithm to sort the traffic and then use one of a number of approaches to prioritize it when forwarding it across the network. Example algorithms include First In First Out (FIFO), Priority Queuing (PQ), and Weighted Fair Queuing (WFQ), each of which addresses a different type of network issue. The first two of these are very simple and relatively intuitive. However, Weighted Fair Queuing normally provides better performance and is commonly used; its operation is best explained by referencing the diagram. As packets are received (for example, by a router), they are classified by traffic type and sorted into an appropriate queue. The WFQ Engine and buffer on the output stream processes queues as they fill (or a timeout occurs) and transmits the packets in the new sequence. As you can see, in this case it redresses the balance across the types to traffic by Byte volume.
An Example WFQ Engine

An example network device is shown below that has rate limiting on both the input and output queues. When it receives a data stream, if the traffic does not exceed the link bandwidth then the packets will be transferred through the bypass queue. When the level of traffic increases and exceeds the available link bandwidth, then congestion management will start and the packets will be placed in queues. Once this starts to happen the shaping and in this case, rate limiting engine will control the traffic in the transmission queue.

Congestion Management

There are a number of realities that network managers are faced with:
- Both old and new applications invariably grow to fill available bandwidth
- Providing additional bandwidth simply postpones the problem and allows more room for bandwidth hogs
- WAN bandwidth is expensive
- Existing network infrastructures are typically not application aware
Packet Shapers and QoS solutions each have their relative merits and applicability to a given environment. They both address the four issues outlined above and have been successfully implemented by Oracle Applications customers. However, it is important to remember that these technologies will not save a hopelessly under provisioned network. The following drawing is based on a Cisco Networker’s presentation on how QoS applies equally well to packet shapers, and summarizes the applicability of these technologies:

![Network Utilization Diagram]

Irrespective of the solution that you ultimately decide to adopt, the first hurdle to overcome when faced with bandwidth related network issues is developing an understanding of your traffic patterns. A number of organizations have successfully used tools such as IPAUDIT [http://ipaudit.sourceforge.net/], packet shapers [http://www.packeteer.com/] and network sniffers to analyze their traffic patterns. The information provided by these tools will allow you to start formulating short and long term strategies to address a network congestion problem.

9.4 ANALYZING THE NETWORK

Protocols, topologies, switches, hubs, routers, and bridges make networking a complex area. The use of a compression technology can also introduce problems. To identify a performance problem, you only require a rudimentary knowledge rather than a detailed understanding of every minute detail of the OSI model. For example, moving a tuned PC will easily identify a problem that can then be passed on to a network expert.
An accurate, detailed network diagram showing size and latency of all links, location and number of all Oracle users and every network device crossed between the desktop and target database is invaluable. It should show bridges, routers, switches, hubs, topologies, bandwidth, and protocol distribution, as well as any firewalls or known bandwidth limitation problem areas. It should indicate whether any WAN usage is domestic or international. If the Maximum Transmission Unit (MTU) on any network device has been changed, this should be noted, together with the default and new value. Any Network Management Systems, traffic management, or priority queuing set on any device should also be clearly identified.

Generally, network characteristics are analogous to I/O constraints. The amount of data per second that the network can transfer is known as the bandwidth; latency is the time delay to transfer the information. Bandwidth can be checked by comparing the available network bandwidth with the network segment loading, which is determined by multiplying the transactions by the average number of bytes for each transaction and by the number of users on the network segment. Latency may be influenced both by device latency, and waiting for the network to be available because the bandwidth has been exceeded or collisions have occurred. The following chart answers one of the most common questions regarding network bandwidth utilization.

When evaluating performance, obtain a network latency map for a full size packet. The maximum transmission unit size can be determined using the `-f` ping option to prevent packet fragmentation (see below), increasing the packet size while there are no errors and noting any timeouts. This should include the time from the client to the various servers and from these to the database server. The internet computing architecture has overcome the Smart Client sensitivity to latency. Acceptable internet computing performance has been achieved on networks of up to 1000ms latency, though high speed PCs were used in the tests.

The client, connected to the same device as the server, should be used to produce best-case benchmarks. Once the figures have been derived, it is an easy task to move the client around the network to identify problem areas. If this approach is not used, and access to the network is restricted, a test area should be agreed and used where performance targets have to be achieved. The test location should not have an unacceptable amount of latency or suffer bandwidth problems.
9.5 Diagnostic Tools

Details of packet retries, collisions and bandwidth utilization may be obtained by placing routers, bridges, switches, and hubs, in diagnostic mode where possible. Network monitors and probes are useful in distributed systems, primarily to check that no network resource is being exceeded. In the absence of a sniffer, operating system commands such as ping are very useful for establishing that packets are not being fragmented and for perhaps tracing the actual route a packet takes.

You should always use a full size packet when measuring latency. For example, to send an unfragmented full-size Ethernet packet, use the command `ping -f -l 1472 -n 50 <host>` . The priority of the ping command may have been reduced on some network devices; this should be checked when your measurements do not make sense.

Trace Route is a route tracing utility that uses the IP Time to Live (TTL) field and ICMP error messages to determine the route from one host to another through a network. The problem with this utility is that it tends to show the optimal route, rather than the actual route that was taken.
Historically, if FND_CONCURRENT_REQUESTS contained more than 5000 rows, then system performance would be adversely affected. Nowadays, customers usually need to maintain many thousands of requests online during each business cycle and it is not uncommon to see 100,000 requests maintained on-line. If you think that the number of requests is a major concern, review the top 10 SQL statements from V$SQLAREA. If you have problems you will typically find between two and five statements referencing the concurrent requests in the top 10 statements. The normal approach to improve performance is to run the purge concurrent requests program to reduce the amount of information held in the concurrent requests table.

You can control concurrent processing performance by setting the concurrent manager cache size and sleep time parameters, to balance the work done by concurrent managers searching for suitable requests to process, against the work done by the managers in processing the requests. However, you should note that when there are no suitable requests waiting to be processed, the concurrent managers may place additional load on the server just by continuously scanning the queue in an attempt to find work.

If you have configured a queue for medium or long-running jobs of, say, 10 minutes or longer, then it makes no sense to scan the queue every minute. Increasing the sleep time may release the system resources from unnecessary queue scans. Consider reducing the sleep time from the default of 60 seconds for queues that usually run short requests submitted as Pending/Normal, but avoid setting manager sleep time less than that of the Conflict Resolution Manager if you have a mixed Applications workload comprising both Pending/Standby and Pending/Normal requests.

If the sleep time and cache size of a concurrent manager are not set at optimal values, you may find that the concurrent manager sleeps after processing requests in its cache before rescanning the concurrent requests table. This can result in a buildup of requests in the queue and a loss of valuable processing time. Increasing the cache size (the number of requests cached each time the queue is scanned) to at least twice the number of target processes may solve the problem. The concurrent manager will process all jobs in the cache before rescanning the queue, and will be processing continuously, provided there are sufficient suitable requests waiting in the concurrent requests table. However, beware of setting the cache size too high if you frequently reprioritize jobs, as a high cache value will reduce the number of times that the request table is scanned, with the result that reprioritizations may not be noticed until all jobs in the cache have been completed.

It is important to customize concurrent managers to suit your site’s individual concurrent processing workload. If you have a high number of small concurrent requests and experience the problem that short requests are being queued for long periods, you should consider creating a separate manager to process these small requests and exclude them from other queues. Test the effects of any changes that you intend to make. Increasing the number of workers or reducing the sleep time may increase the throughput of the concurrent managers at the cost of increasing the workload on the concurrent processing server. Both these changes increase the possibility of the concurrent managers locking each other out. If your system is resource-constrained, then not only will you slow down the processing for every online Applications user, but you will also find that throughput does not increase, as jobs take longer to run.
Analyzing the FND_CONCURRENT_PROGRAMS, FND_CONCURRENT_REQUESTS, FND_CONCURRENT_PROCESSES, FND_ORACLE_USERID, and FND_CONCURRENT_QUEUES tables using a full COMPUTE will enable use of the FND_CONCURRENT_REQUESTS_N2 index, substantially improving performance. For Release 11, you should check an index exists on the FND_CONC_RELEASE_CLASSES table for columns RELEASE_CLASS_ID and OWNER_REQ_ID.

10.1 **Monitoring Concurrent Programs**

This section shows how you can find expensive or slow-running programs, and provides an overview of some monitoring tools available. When you have identified a specific problem with one of the concurrent programs, you will need to trace its execution to determine the nature of the problem. You can do that using one of the approaches described in the following sections.

10.2 **Enabling SQL Trace**

Since Release 10.7, trace can be enabled by the system administrator at the user level by using the profile option Utilities: SQL Trace. In earlier releases, concurrent programs could only be traced by running them from the command line. You can use either of the following methods to determine the command used to run a particular program:

Using the Trace/Debug parameter
Using the FDSQLCHK environment variable

10.3 **Application or Program Trace/Debug Parameter**

Many of the Applications modules use profile options for tracing and debugging. Examples include: AR: Enable SQL Trace, INV: Debug Trace, MRP: Trace Mode, and PO: Enable SQL Trace for Receiving Processor. If you have a test system in which the problem can be reproduced in isolation, consider using this approach.

Many concurrent programs contain a parameter for tracing and debugging. Examples of trace parameters include SQL Trace, Trace Mode, Trace Option, and Trace Switch. Examples of debug parameters include Debug, Debug Mode, and Debug Switch. You may need to make these parameters visible in the Control Concurrent Parameters screen in order to be able to see and set them when submitting a request. If there is a trace parameter, this should be considered the preferred method for tracing the module. Enabling debug options provides diagnostic information, but it does not usually provide a trace file.

10.4 **Using the FDSQLCHK Environment Variable**

The environment variable FDSQLCHK can be set to the value FULL to trace any concurrent manager requests. You only need to set the variable at the operating system level and restart the concurrent manager to pick up the new setting. The disadvantage of this approach is that you have to restrict all other concurrent jobs, otherwise you may waste time matching the trace files to the appropriate programs. An alternative, which works well in a production environment, is to use the following approach:

Always remember to disable any debug or trace profile options at the end of the test.
First, in the development or test environment:

1. Rename the executable program
2. Create a script using the original name and a single line: `echo $*`. This will echo all the arguments and save them to the regular output file.
3. Rename the script and restore the original executable
4. Next, in the production environment:
5. Set the environment variable `FDSQLCHK=FULL`
6. Run the concurrent request from the operating system using the `CONCSUB` command with the arguments from the output file. This will generate a trace file for this program only.
7. Unset the `FDSQLCHK` environment variable

Be aware that arguments $0 through to $4 are reserved for Applications:

$0 - Shell script name
$1 - Oracle username/password
$2 - User_id
$3 - Oracle Applications username
$4 - Concurrent request id

If the program passes more than nine parameters, using $* will not display them in the output file. To work around this problem, you will need to modify the program along the following lines:

```
echo $*
shift
param9=$9
echo $param9
shift
param9=$9
echo $param9
```

### 10.5 Enabling Trace for a Report

In Applications, a trace switch parameter was added to a number of reports to enable trace files to be produced. If the parameter exists in the program definition parameters but is not visible, change the Display option to “Yes” in the Define Concurrent Programs form. You can quickly check any report using the UNIX command `grep -i p_trace_switch <file>`. If you want to trace a custom report, add a before report trigger step:

```
IF (:p_trace_switch = 'Y') THEN
  SRW.DO_SQL('alter session set sql_trace TRUE');
END IF;
```

Next, define a new report parameter `P_TRACE_SWITCH`: TYPE=Character, LENGTH=1, INITIAL VALUE=N The AP_SRS_YES_NO_OPT value set may be used with: Security = No, Display size = 4, Token = P_TRACE_SWITCH
10.6 **Using Oracle Trace**

If all else fails, submit the program and use the ORADEBUG approach. The easiest way to do this quickly is to have the user log off the system and take a snapshot or copy of V$SESSION. Have them log on and find the SPID entry that relates to them using the following SQL statement:

```sql
select S.OSUSER, 
    S.USERNAME, 
    S.PROGRAM, 
    P.SPID PID 
from SYS.V_$PROCESS P, SYS.V_$SESSION S 
where P.ADDR = S.PADDR;
```

<table>
<thead>
<tr>
<th>OSUSER</th>
<th>USERNAME</th>
<th>PROGRAM</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>applmgr</td>
<td>oracle@uksn203</td>
<td>(PMON)</td>
<td>12537</td>
</tr>
<tr>
<td>applmgr</td>
<td>oracle@uksn203</td>
<td>(DBW0)</td>
<td>12539</td>
</tr>
<tr>
<td>applmgr</td>
<td>oracle@uksn203</td>
<td>(LGWR)</td>
<td>12541</td>
</tr>
<tr>
<td>applmgr</td>
<td>oracle@uksn203</td>
<td>(CKPT)</td>
<td>12543</td>
</tr>
<tr>
<td>applmgr</td>
<td>oracle@uksn203</td>
<td>(SMON)</td>
<td>12545</td>
</tr>
<tr>
<td>applmgr</td>
<td>oracle@uksn203</td>
<td>(RECO)</td>
<td>12547</td>
</tr>
<tr>
<td>applmgr</td>
<td>APPS</td>
<td>sqlplus@uksn203 (TNS V1-V3)</td>
<td>1779</td>
</tr>
</tbody>
</table>

In this example, the Applications user has a PID of 1779. Next, use Server Manager to set the ORADEBUG session and the event. For example:

```
SVRMGR > ORADEBUG SETOSPID 1779
SVRMGR > ORADEBUG EVENT 10046 trace name context forever, level 12
```

When the process has completed, use the following command:

```
SVRMGR > ORADEBUG EVENT 10046 trace name context off
```
10.7 Monitoring Concurrent Jobs Using Standard Scripts

Oracle Applications provides a number of very useful scripts in $FND_TOP/sql. Not only can you use them to monitor the concurrent managers, but you can also use them to assist with diagnosis of performance problems.

$FND_TOP/sql Scripts for monitoring concurrent managers

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afcmcreq</td>
<td>Prints the log file name of managers that can run a given request</td>
</tr>
<tr>
<td>Afcmstat</td>
<td>Displays all defined managers and shows current status of manager processes</td>
</tr>
<tr>
<td>Afimchk</td>
<td>Checks to see if the concurrent manager monitor is actually running</td>
</tr>
<tr>
<td>Afimlock</td>
<td>Monitors locks that the ICM and CRM are waiting for. Run this if you have long delays when submitting jobs.</td>
</tr>
<tr>
<td>afimpmon</td>
<td>Sets the PMON method based on the argument (for CONC_PMON_METHOD)</td>
</tr>
<tr>
<td>Afimveri</td>
<td>Resets the concurrent manager monitor</td>
</tr>
<tr>
<td>Afqpmrid</td>
<td>Returns the operating system process id of the FNDLIBR process of a concurrent manager request. Use this information to run ORADEBUG for specific requests.</td>
</tr>
</tbody>
</table>

$FND_TOP/sql Scripts for reviewing concurrent requests

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrqpend</td>
<td>Lists all the Pending/Normal requests. These include jobs on hold, future-dated requests, and those waiting for a manager.</td>
</tr>
<tr>
<td>Afrqrun</td>
<td>Lists all Running, Terminating, and Paused requests</td>
</tr>
<tr>
<td>Afrqscm</td>
<td>Prints the log file names of managers that can run a given request</td>
</tr>
<tr>
<td>Afrqstat</td>
<td>Summarizes concurrent request execution since a particular date</td>
</tr>
<tr>
<td>Afrqwait</td>
<td>Selects all the Pending requests with status Q</td>
</tr>
</tbody>
</table>

10.8 Analyzing Historical Information

It is highly recommended that you maintain historical information from the FND_CONCURRENT_REQUESTS table. Simply copy the information before running the purge requests program, or create a trigger on the table. This will enable effective performance analysis of batch jobs. This feature is scheduled to be included in a post-11i release.

Purging the concurrent requests queues will normally improve the performance of the concurrent managers and the performance of some Applications screens. However, depending on the type of business, you may need to maintain many thousands of requests online during each business cycle.
Use the following to calculate the number of concurrent requests per week:

```
SELECT COUNT(*) Total,
       sum(decode(greatest(0,ceil
               (sysdate - actual_completion_date)),
               least(7,ceil(sysdate-actual_completion_date)
               ),1,0)) Week4,
       sum(decode(greatest(8,ceil
               (sysdate - actual_completion_date)),
               least(14,ceil(sysdate-actual_completion_date)
               ),1,0)) Week3,
       sum(decode(greatest(15,ceil
               (sysdate - actual_completion_date)),
               least(21,ceil(sysdate-actual_completion_date)
               ),1,0)) Week2,
       sum(decode(greatest(22,ceil
               (sysdate - actual_completion_date)),
               least(28,ceil(sysdate-actual_completion_date)
               ),1,0)) Week1
FROM FND_CONCURRENT_REQUESTS
WHERE ACTUAL_COMPLETION_DATE is not null;
```

<table>
<thead>
<tr>
<th>TOTAL</th>
<th>WEEK4</th>
<th>WEEK3</th>
<th>WEEK2</th>
<th>WEEK1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5432</td>
<td>122</td>
<td>131</td>
<td>77</td>
<td>102</td>
</tr>
</tbody>
</table>

### 10.9 Current Issues

There are two main performance issues with the concurrent manager in Release 11i that you will be able to identify very quickly:

**Patch 1815061** Child cache buffers chains’ latch contention on FND_CONCURRENT_QUEUES. When running a high workload of Concurrent Manager jobs, the system experiences a serious slowdown that looks like a hang to the user. The system is not hanging, but running very slowly.

**Patch 1815716** This is the tracking bug for 1668016. In this bug, the number of running FNDLIBR processes increases higher than the target number of declared processors. This mainly causes problems shutting down the concurrent managers and the only solution is to kill the FNDLIBR processes because they do not self terminate.
10.10 Enhancing Concurrent Processing

This white paper (Note No. 164085.1) introduces several techniques that can be used to enhance and automate the Oracle Applications concurrent processing suite. It starts by presenting some simple changes that control the concurrent program that they are associated with. The paper continues by introducing more powerful techniques, which can be adapted to work on a range of concurrent requests. These examples are more complex but a lot of effort has gone into ensuring that they are simple enough to ensure that anybody with a rudimentary knowledge of shell scripting and SQL*Plus should be able to understand them.

Several concurrent processing functions can be automated using standard Oracle Applications functionality. For example, when used with Oracle Alert, you can easily set up an alert that will page a system administrator if there is a problem with a concurrent manager or one of the queues. This type of standard functionality can greatly enhance your management capabilities and is already well documented.

The types of examples described in this paper differ in that they control the output whereas Oracle Alert tends to be more event driven.

- Moving sensitive report files to secure directories
- Redirecting concurrent output to specific directories (other than $APPLCSF/$APPLOUT)
- Compressing reports and distributing them during off-peak periods over Wide Area Network links (which may also be low quality links)
- Faxing reports, orders or other documents
- Converting documents to Adobe’s Portable Document Format (PDF) format ready for distribution
- Converting to PDF without Adobe
- Printing PDFs from Unix
- Emailing reports and documents
- Publishing information or complete reports to the web
- Archiving selected request output before purging the concurrent requests

As you can see, most of the functions in this list would normally require some form of manual intervention; however, they can all be automated. In most of the cases, the necessary changes usually only take a few minutes. For example, you can modify the printer command line to move sensitive files the instant that they complete.

The paper describes conversion routines that you can use to convert a report into a variety of output formats without having to rerun a concurrent request multiple times. This type of approach can be easily be extended to automatically zip specific types of documents or convert them to PDF format, which could then be kept online for differing lengths of time depending on their type. While zipping is simply a matter of running a utility, conversion to PDF is more involved. There is a proliferation of document conversion utilities in the public domain; when making your selection, it is best to select those that have a command line interface, as these will be easier to automate.

The final point in the list can give you complete control over archiving and purging. Although the paper does not propose a full solution, it does provide you will all the necessary tools to create your own.
11 TUNING THE USERS

Inefficient user queries are an aspect of performance that is commonly overlooked. Teaching users to query efficiently and effectively reduces the load on almost every component in the system and returns information more quickly. Such techniques are also useful in extending the functionality of screens and screen-reports when using the ‘export’ function. In general, this type of training should be delivered as a special short course, in isolation from other topics.

In one particular Smart Client engagement, the users of a data-processing department were forced to use the keyboard when their mice were taken away. After an initial drop, there was roughly a 15% rise in productivity. This shows that looking at areas other than the Application system can be worthwhile. In this case, the high performance gain was mainly due to the limited number of screens operations that were used. In Release 11i, it is possible to cancel blind queries.

11.1 QUERING FORMS

All user queries in forms may be subject to restrictions built into the form, folder, or pre-queries in the form code that may be conditionally applied. Simple queries can be performed by entering any query criteria while in query mode. The rows returned to the screen are the intersection of the database records, any in-built form restriction and any query criteria entered by the user.

![Form Query Restrictions Diagram]

Form Query Restrictions

Most users know to enter query reduction criteria in fields. Quite often they will execute multiple queries, or perhaps swap between forms to retrieve the information they require. Better ways of querying not only speed up data entry for users, but also reduce the load of multiple queries on the system by eliminating the requirement for them. These methods are discussed in the next few examples.

11.1.1 CLEVER QUERIES

This method explains how to query back rows using parameters that are more complex than normal, including those where fields are specifically null. For example, finding journal headers starting with ‘ABC’, together with those starting with ‘DEF’, all in one query rather than two.
To create a clever query:

- Enter Query Mode
- Enter ‘a’ in the field to be used in the query
- Execute Query
- A query window will appear in which the search criteria are entered.
- Click ‘OK’ to execute the query.

The following operators may be used:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>= xyz</td>
<td>Equals xyz</td>
<td>xyz can be a number, a word, or date enclosed in single quotes</td>
</tr>
<tr>
<td>!= xyz</td>
<td>Not Equals to xyz</td>
<td>As above</td>
</tr>
<tr>
<td>&lt; xyz</td>
<td>Less than xyz</td>
<td>As above</td>
</tr>
<tr>
<td>&gt; xyz</td>
<td>Greater Than xyz</td>
<td>As above</td>
</tr>
<tr>
<td>like xyz</td>
<td>Similar to xyz</td>
<td>like ‘xyz’ may contain the _ wildcard %</td>
</tr>
<tr>
<td>Between x and y</td>
<td>Between x and y</td>
<td>a and b may be numbers, words or dates</td>
</tr>
<tr>
<td>in (x,y,z,...)</td>
<td>exists in list</td>
<td>As above</td>
</tr>
<tr>
<td>is null</td>
<td>is empty</td>
<td>For example printed_date is null</td>
</tr>
</tbody>
</table>

This method may be used to return two specific invoices on the same query on Invoices Entry form using, for example, INVOICE_NUM IN (1234, 5678), or to return a list of Approved Purchase Orders that have not been printed, using (PRINTED_DATE IS NULL) and (APPROVAL_FLAG = ‘Y’). The more complex functions of SQL*Plus can also be used. For example, the AND and OR operators can be used to collect a number of criteria together into a single query.

11.1.2 Extending the Queries

This method can be extended enabling a ‘clever query’ in multiple fields using :a, :b, :c and so on. For example, use the Employees form to query back all employees with the name ‘SMITH’ or ‘KING’ and salaries between 1000 and 3000 by entering ‘a’ in the name field and ‘b’ in the salary field. Execute the query, then in the query window use :a in (‘SMITH’, ‘KING’) and :b between 1000 and 3000.

11.2 Record Retrieval

When users query and review multiple records in a multi-row block, they tend to scroll down a record at a time. The cumulative time for scrolling down, say, six records individually is far greater than pulling the scroll bar down to retrieve the next set of records. A form is usually set to buffer a certain number of rows. Unless a record is still cached, each time a record is scrolled, extra round trips are made across the network; the database has to retrieve and sort rows; the system has to access the associated data block. It is far more efficient to pull the scrollbar down and retrieve the next set of records.
12 OPTIMIZING FINANCIAL STATEMENT GENERATOR

This is a collection of hints and tips from a collection of technical bulletins to improve the performance of FSGs.

12.1 IMPROVING GENERAL PERFORMANCE

You may find these general rules useful:

- Use summary accounts whenever possible to improve FSG and other tasks such as recurring journals, mass allocations, and so on.
- Define rollup groups and summary templates that exactly cover the Accounting Flexfield ranges that will be used in the report.
- Regularly run the GL Optimizer using the ‘Maintain Indexes and Gather Statistics’ option after major postings.
- If there has been a lot of activity on your system, also consider running the GL Optimizer before submitting a group of several long-running FSG reports.

12.2 IMPROVING PERFORMANCE WHEN DESIGNING A REPORT

These are recommendations that can improve the performance of a particular report:

- Minimize the number of Accounting Flexfield ranges.
- Minimize the number of possible child ranges for parent segment values.
- Minimize the number of flexfields to process (possibly by defining summary accounts)
- Minimize the number of periods to process
- Minimize the number of currencies, budgets, and encumbrances in the report

If an FSG is doing full table scans or more than 20% of the rows are being processed, either change the report to use fewer accounts or consider creating summary accounts.

12.2.1 ADDITIONAL INDEXES

If a report does not run within the required batch window, creating one or more single segment indexes on GL_CODE_COMBINATIONS may improve the performance of summary account creation:

- Create indexes on GL_CODE_COMBINATIONS on each of the accounting flexfield segments in the report. If you have non-selective indexes, then try creating indexes on just the selective indexes first.
- Create a concatenated index on all of the segments used in the Accounting flexfield.
- Run the GL Optimizer, but do not select the option to create indexes, as this will delete the indexes that you have just created.
Note that GL_CODE_COMBINATIONS is the only table in GL that you are allowed to add indexes to. You will need to test this carefully, as any performance improvement will depend on the index selectivity. Further, you may experience a reduction in posting performance caused by the additional index updates.

12.2.2 THE SORT_AREA_SIZE PARAMETER

Run a Statspack report for the period the FSG is running and review the figures for memory and disk sorts. If you find that the ratio is less than, say, 10,000 memory sorts to 1 disk sort, consider increasing the size of the sort area by increasing the value of the SORT_AREA_SIZE parameter in your init.ora file. Again, you need to approach this with care, as, for example, in Release 11i a large sort area will favor full-table scans over index scans. Your DBA will need to monitor the effect of the change on the whole system over a period of time.

12.2.3 CREATING A REPORT SET

In order to run FSGs from the Standard Reports Submission screen:

Using the System Administrator responsibility:

1. Navigate to the Define Concurrent Program screen (Concurrent, Program, Define)
   Query the program: Financial Statement Generator.
   Set or check the ‘Use in SRS’ checkbox.

2. Navigate to the Define Responsibility screen (Security, Responsibility, Define)
   Query the GL Responsibility from which you want to be able to submit the FSGs as standard reports.
   Select or make a note of the Request Group Name.

3. Navigate to the Define Request Security Groups screen (Security, Responsibility, Request)
   Query the required Request Group.
   Enter a 'Requests' record as follows:
   Type: Program
   Name: Financial Statement Generator
   Application: Oracle General Ledger

You will now be able to run FSGs from the standard reports submission screen for that responsibility.
13 TUNING AND PURGING WORKFLOW

Workflow is used extensively in Applications and if unmanaged can be the source of poor performance. This section discusses the setup and purge routines that are available to you. It also describes the purge options and how to deal with very large data volumes.

13.1 ORACLE APPLICATIONS PROFILE OPTIONS

Some products of Oracle Applications have utilized the internal features of Oracle Workflow and brought them to the surface as configurable application profile options. Two profile options have historically affected performance:

- **Account Generator: Run in Debug Mode**
- **Purchasing: Workflow Processing Mode**

13.1.1 ACCOUNT GENERATOR: RUN IN DEBUG MODE

This controls the Forced Synchronous mode of the Workflow Engine. Always ensure that this is switched off unless you are trying to debug the output. The two settings are as follows:

- **Yes** The Account Generator workflows runs in audit mode and status information is saved to the Workflow history tables.
- **No** The Account Generator workflows run in non-audit mode and no status information is saved, therefore increasing performance and reducing the wait time when generating an account.

13.1.2 PURCHASING: WORKFLOW PROCESSING MODE

This profile option can be set to Online or Background:

- **Online** The Workflow Engine will run the Purchase Order and Requisition workflows online when transactions are created.
- **Background** This causes the purchase order and requisition workflows to be deferred to the Background Engine. This can increase throughput; but before setting this value, ensure that you have configured at least one Background Engine and that it is running.

13.2 PURGING RUN-TIME INFORMATION

When the engine executes any type of workflow other than Forced Synchronous, status information is stored into the Workflow history tables. Over time, this can amount to many thousands of rows; the growth of these tables depend on the complexity of the workflows and how often they are used.

The problems associated with large volumes of data can be avoided by regularly purging run-time information. There are two types of information:

- **Design time information** This is the workflow definition information that was created within the Oracle Workflow Builder.
- **Run-time information** This includes the status history, Item Attribute values, and notifications that are created during the execution of a workflow process.

Managing the volume of information is performed using Workflow Purge APIs. They only operate on obsolete run-time information, leaving all design-time information intact.
information intact. Administration scripts are provided to operate on both run-
time and design-time information. Both APIs and scripts are discussed in the
following sections.

13.2.1 Workflow Purge APIs

The Purge APIs are a set of procedures contained in the WF_PURGE package.
The following table lists some of the procedures available and the parameters that
they accept, which control the amount of data to purge.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WF_PURGE.ITEMS</td>
<td>Itemtype Itemkey Enddate</td>
<td>Removes all run-time data associated with completed items.</td>
</tr>
<tr>
<td>WF_PURGE.ACTIVITIES</td>
<td>Itemtype Enddate</td>
<td>Removes obsolete activities versions. These are versions of activities that are no longer used by any item.</td>
</tr>
<tr>
<td>WF_PURGE.NOTIFICATIONS</td>
<td>Itemtype Enddate</td>
<td>Removes old notifications. These are notifications that are no longer used by any item.</td>
</tr>
<tr>
<td>WF_PURGE.TOTAL</td>
<td>Itemtype Itemkey Enddate</td>
<td>Purges both item and activity data.</td>
</tr>
<tr>
<td>WF_PURGE.TOTALPERM</td>
<td>Itemtype Itemkey Enddate</td>
<td>Deletes all eligible obsolete run-time data that has a persistence type of Permanent.</td>
</tr>
<tr>
<td>WF_PURGE.ADHOCDIRECTORY</td>
<td>Enddate</td>
<td>Purges all ad hoc users and roles that are not associated with a notification.</td>
</tr>
</tbody>
</table>

In their simplest form, the purge procedures can be run with no parameters at all. For example, to purge both item and activity data for completed workflow processes use the following:

```
execute wf_purge.total;
```

The concept of workflow persistence was introduced in Oracle Workflow 2.5. The Persistence Type controls how long a status audit trail is maintained for each instance of the Item Type. The three types of persistence are Permanent, Temporary, and Synchronous. Synchronous, when run with no auditing, does not have any history information. The history information for workflows marked as Permanent can only be removed by WF_PURGE.TOTALPERM; the history information for workflows marked as Temporary is removed by WF_PURGE.TOTAL.

For best performance, run WF_PURGE.TOTAL frequently, and WF_PURGE.TOTALPERM periodically, but be aware of your site’s policy on status history retention. Further details can be found in the Oracle Workflow User Guide.
13.3 Using the Generic Purge Procedures

You can run execute WF_PURGE.TOTAL from the SQL command prompt. This will not delete anything in process or active. However you should also run execute WF_PURGE.TOTALPERM, which will delete any items with a persistence type of permanent and execute WF_PURGE.NOTIFICATIONS.

Note: If you run into problems such as running out of rollback space, you will need to purge by item type or something else to limit the amount of data (committing between each iteration).

13.3.1 Standard Purge Scripts

These scripts remove not only run-time data, but also Item Type and workflow definition information. The following table lists and describes some of the available scripts. Carefully consider the impact of running these scripts, as purging run-time information for incomplete workflow processes may result in the underlying application being placed into an unstable state. Any transactions that are incomplete when the process has been removed may not be able to be completed.

Caution: Extreme care should be taken when using these scripts!

<table>
<thead>
<tr>
<th>Script</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FND_TOP/sql/wfrmitms.sql</td>
<td>Itemtype</td>
<td>Removes the status information for a specified Item Type and Item Key</td>
</tr>
<tr>
<td>$ORACLE_HOME/admin/sql/wfrmitms.sql</td>
<td>Itemkey</td>
<td></td>
</tr>
<tr>
<td>$FND_TOP/sql/wfrmtype.sql</td>
<td>None</td>
<td>Purges all run-time information for a given item type</td>
</tr>
<tr>
<td>$ORACLE_HOME/admin/sql/wfrmtype.sql</td>
<td>None - You will be prompted</td>
<td></td>
</tr>
</tbody>
</table>

13.4 Useful Queries

The main tables of concern are WF_ITEM_ACTIVITY_STATUS and WF_ITEM_ATTRIBUTE_VALUES. The use of the queries is self evident. The first provides a summary of the workflows and their activity status, the second provides a break down by item type, which you may find useful when dealing with large data volumes.

```
SELECT item_type,activity_status,count(*)
FROM  wf_item_activity_statuses
GROUP BY item_type,activity_status;
```

```
ITEM_TYP ACTIVITY  COUNT(*)
-------- --------  ---------
INVTROAP COMPLETE  8
WFERROR COMPLETE   413298
INVTROAP DEFERRED  1
CREATEPO ERROR     5
```
SELECT item_type, count(*)
FROM   wf_item_attribute_values
GROUP BY item_type;

<table>
<thead>
<tr>
<th>ITEM_TYP</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>APEXP</td>
<td>4490</td>
</tr>
<tr>
<td>AZNF003</td>
<td>5</td>
</tr>
<tr>
<td>AZNF004</td>
<td>5</td>
</tr>
<tr>
<td>AZNF005</td>
<td>5</td>
</tr>
<tr>
<td>CREATEPO</td>
<td>360</td>
</tr>
<tr>
<td>ECO_APP</td>
<td>35</td>
</tr>
</tbody>
</table>

### 13.4.1 Addressing Workflow Errors
In order to address the errors, go in through the Workflow Web agent as SYSADMIN, view the waiting notifications on these errors and review their content. They will contain why the WFERRO process was started. You can take the appropriate action from the notifications: abort, retry, ignore.

### 13.4.2 Working with Large Numbers of Errors
If you have a huge number of errors then you could use the SQL queries above to speed up your analysis, rather than reviewing each record. Consider using the wfretry.sql script to restart the processes from SQL*Plus. Note that it is not a good idea to go through and abort all of these error’d processes as they could be active legitimate workflows that are pending within the application. Only if you are happy that all of the error’d processes can be deleted then you could consider using the wf_engine.abortProcessAPI. Again, use extreme caution when manipulating data this way.
14 CONCLUSION

An accurate problem definition is key to identifying the source of performance issues and will help focus the tuning effort to achieve the performance targets. The best benchmark figures for the system are obtained by initially tuning the client, the database and the server in that order. Batch jobs and reports should be rescheduled to minimize the effect of on-line users before tuning application SQL access paths or using the tuned client to benchmark the network.

Throughout the tuning exercise, make a change and measure the affect. Once the entire system has been checked and tuned, re-benchmark. If targets are still not met, stop and possibly reconsider the targets before recommending a hardware upgrade. Always investigate every area. Look for novel ways to tune such as the way that forms are used and queried, or perhaps group jobs that access the same tables to run at the same time.

Although tuning is a science, there is a lot of common sense involved.
APPENDIX A - WINDOWS PERFORMANCE CHECKLIST

Memory is most important for Smart Client whereas CPU is more important for internet computing. Due to popular demand, the checklists have been changed to the Windows NT/2000 versions.

<table>
<thead>
<tr>
<th>Windows NT/2000 Basic System Performance Checklist</th>
<th>CPU</th>
<th>Disk</th>
<th>Mem</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check the Windows version</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify file system integrity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Back up the Registry</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create an Emergency Repair Disk (ERD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disable automatically started programs in the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Startup folder and the Registry</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tune the BIOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modes of Operation: Set to Optimal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard Repeat Rate: Maximize throughput</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caching: Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM Shadowing: Disable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floppy Drive Seek: Disable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unused IDE Channels: Disable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tune the Boot Process</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce selection time for operating system to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce selection time for hardware profiles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Windows NT/2000 File System Performance Checklist</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty the Recycle Bin</td>
<td></td>
</tr>
<tr>
<td>Uninstall unused Windows NT components</td>
<td></td>
</tr>
<tr>
<td>Delete temporary files</td>
<td></td>
</tr>
<tr>
<td>Delete cached Internet files</td>
<td></td>
</tr>
<tr>
<td>Delete generic files</td>
<td></td>
</tr>
<tr>
<td>Delete document indexes</td>
<td></td>
</tr>
<tr>
<td>Archive large numbers of small documents</td>
<td></td>
</tr>
</tbody>
</table>
### Windows NT/2000 General System Performance Checklist

<table>
<thead>
<tr>
<th>Operation</th>
<th>Control Panel and Settings</th>
<th>CPU</th>
<th>Disk</th>
<th>Mem</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable screen savers or use blank screen</td>
<td>Control Panel</td>
<td>Display</td>
<td>Screen Saver. Create a shortcut to C:\WINNT\Black16.scr and assign a shortcut key. This is very important on server and middle-tier machines!</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Remove wallpapers</td>
<td>Control Panel</td>
<td>Display</td>
<td>Screen Saver</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Upgrade video drivers</td>
<td>Control Panel</td>
<td>Display</td>
<td>Settings</td>
<td>Display Type, latest version</td>
<td>✔️</td>
</tr>
<tr>
<td>Decrease screen colors</td>
<td>Control Panel</td>
<td>Display</td>
<td>Settings</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Remove animated cursors</td>
<td>Depends on type used. Do not use Desktop Themes.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Remove shortcut toolbars</td>
<td>Start Menu</td>
<td>Programs</td>
<td>Startup</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Tune data entry speed</td>
<td>Control Panel</td>
<td>Keyboard; set Repeat Delay short and Repeat Rate fast.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Check disk monitoring</td>
<td>Run the command diskperf and ensure that disk monitoring is disabled.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

### Tune the Windows Environment

<table>
<thead>
<tr>
<th>Task</th>
<th>Control Panel and Settings</th>
<th>CPU</th>
<th>Disk</th>
<th>Mem</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximize application performance</td>
<td>Control Panel</td>
<td>System</td>
<td>Performance. For Performance Boost for Foreground Application, move slider to Maximum.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Disable unnecessary services</td>
<td>Control Panel</td>
<td>Services. As a minimum, consider disabling the Alerter, Clipbook Server, Computer Browser, Messenger, Schedule, and Spooler.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Disable protocols and re-sequence bindings</td>
<td>Control Panel</td>
<td>Network</td>
<td>Protocols</td>
<td>Network Access Order</td>
<td>✔️</td>
</tr>
</tbody>
</table>

### Tune Memory

<table>
<thead>
<tr>
<th>Task</th>
<th>CPU</th>
<th>Disk</th>
<th>Mem</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce installed fonts</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Remove font aliasing (10SC only)</td>
<td>Delete \Tools\Common\uifont.ali</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

### Tune Disk I/O

<table>
<thead>
<tr>
<th>Task</th>
<th>CPU</th>
<th>Disk</th>
<th>Mem</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defragment the disk</td>
<td>Use a third-party utility to defragment disks (built in to Windows 2000).</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Optimize the paging file</td>
<td>Ensure the adequate size of paging file(s). Add more and locate on a separate, dedicated drive or partition using FAT with a 32KB cluster size.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Do not use compressed files</td>
<td>Clean up the file system and decompress any compressed files.</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

### Tweak UI enhancements

- **Mouse**: Maximize menu speed and review mouse sensitivity settings
- **General**: Disable Windows animation and smooth scrolling

### Apply service packs and hot fixes

Check for latest versions, allowing a suitable period for problems to be discovered before installation