UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

CUSTOMIZED DATABASE AND SYSTEMS TUNING METHODOLOGY: A CASE OF PAYROLL PROCESSING OPTIMIZATION

BY

MICHAEL NJOROGE MUKIRI
P58/61516/2010

SUPERVISOR: MR. ANDREW KAHONGE

MAY 2013

Submitted in partial fulfillment of the requirements of Masters of Science in Computer Science

DECLARATION

This research project, as presented on this report is my original work and to the best of my knowledge has not been presented for any other university award.

Mukiri Michael Njoroge

P58/61516/2010

Signed: ………………………………….

Date: ……………………………………..

This project has been submitted as part of fulfillment of the requirements for the award of Masters of Science in Computer Science of the School of Computing and Informatics of the University of Nairobi, with my approval as the University Supervisor.

Mr. Andrew Mwaura

Signed: ………………………………….

Date: ……………………………………..

ACKNOWLEDGEMENTS

My sincere appreciation to the following persons who made this research project a success.

Mr. Andrew Mwaura, my supervisor for his invaluable counsel and guidance.

Dr. Eike Schallen, my supervisor in Germany for his invaluable counsel and guidance.

Prof Gunter Saake, for his guidance and feedback.

Dr. Dan Orwa, for his guidance and feedback.

Dr. Agnes Wausi, for her guidance and feedback.

Lecturers, School of Computing and Informatics for sharing knowledge and contribution in developing Kenyans through higher education.

DAAD, for facilitating my studies.

My family, for believing in me.

May God bless you.

DEDICATION

For my wife Loise Muthoni.

For my daughter Yvonne Wanjiru.

ABSTRACT

Performance of database applications is becoming increasingly important as more and more processes are automated in organizations. Most businesses rely heavily on database systems and non-optimal performance of computer system has an immediate negative impact on business. With internet becoming cheaper and millions of mobile phones being used to access systems, business applications are under pressure to perform optimally and support business needs.

An evaluation and analysis of existing database and application tuning methodologies was conducted and gaps identified. An integrated holistic database and application fine tuning methodology was developed that tried to address the identified gaps. The aim of the methodology is to guide the tuning expert in identifying bottlenecks in various tiers (Database, Network, and Application) that can have impact on performance. In each of the tiers the main bottlenecks that should be looked out for are highlighted and resolutions of how to resolve them are also suggested. The developed methodology makes no assumption as to which tier contains the bottleneck but gives guidelines on how various bottlenecks can be identified in each tier. In addition an overview of existing tools for gathering performance statistics and utilizations statics for various tiers is given. To address the challenge of choosing which bottleneck to resolve first, ranking formulae is suggested that ranks bottlenecks based on their overall expected improvement in performance after resolution of the bottleneck, cost of resolving the bottleneck and the time taken to resolve the bottleneck. The bottleneck with highest rank is resolved first.

The methodology was applied on a poorly performing payroll application and it was proven that response time of the payroll application could be improved significantly by eliminating identified bottlenecks. By use of the methodology it was illustrated how tuning experts can approach the tuning exercise in a more structured and holistic approach with the sole purpose of identifying and resolving bottlenecks in any tier as fast and cost effective as possible. The tuning developed database application tuning methodology can be in-cooperated in database and systems administrators’ standard operating manuals as a tuning guide.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADDM	: Automated Database Diagnostic Monitor.
APP	: Application
ATA	: Advanced Technology Attachment
CBO	: Cost Based Optimizer
CPU	: Central processing Unit
KPH	: Kilometers per hour
SQL	: Structured Query Language
RAID	: Redu`ndant Array of Independent Disks
RDBMS	: Relational Database Management System
SCSI	: Small Computer System Interface
SAS	: Serial attached SCSI
SATA	: Serial ATA
SQA	: Semantic Query-based Annotation
SSD	: Solid State Drives

CHAPTER 1: INTRODUCTION

1.1. BACKGROUND

System performance has become increasingly important as computer systems get larger and more complex with Internet playing a bigger role in business applications. Systems access is increasing almost exponentially with the proliferation of mobile devices. It is expected in the near future that mobile devices will surpass personal computers in accessing systems. Consequently, with increased usage and as the world become exceedingly computerized more will be demanded from existing applications. Systems that do not scale up to higher levels of workload appropriately or are currently experiencing performance challenges will almost grind to a halt, users and business needs will continue not to be met satisfactorily. The necessity for business applications to meet business needs and to scale up adequately is a key motivator of performance tuning.

For database application systems to function optimally, performance has to be designed and built into a system. It does not just happen. An application depends on multiple components for its optimal performance, it performs sub optimally not because its components are saturated but because one component acts as a bottleneck to overall performance of system (Sasha, 2003).Performance problems are usually the result of contention for, or exhaustion of, some system resource. When a system resource is exhausted, the system cannot scale to higher levels of performance. By eliminating resource conflicts, systems can be made scalable to the levels required by the business. (Immanuel, 2011)

It is widely believed by tuning experts that Performance tuning and optimizations of applications and databases is more of an art than a science. It is a task which is not easy to quantify or automate with decision rules. Burleson (2009) compares database application tuning to fixing a vehicle as it cruises down the road at 80 KPH. It’s a dynamic environment where variables and parameters change constantly and the act of measuring performance can have an impact upon performance itself. It is for this reason that a systematic and structured approach of tuning and optimizing database applications is paramount to ensuring the task of tuning is not a nightmare for the expert. Performance strategies are required that give clear and simple steps that can lead to identification and resolution of bottlenecks which will result to dramatic improvement of database application system performance

System performance is further complicated by databases and applications being dynamic environments where data volumes and user populations grow, new versions of applications and databases being implemented, and server configurations changing constantly. These database and application changes make performance unpredictable and uncertain. (Quest, 2010)

1.2. PROBLEM STATEMENT

Many methodologies exist of tuning and optimizing application and databases separately, little attention has been given to formulating a holistic approach of tuning applications and databases in a good documented and systematic approach.

Existing application tuning approaches are either application specific or are proposed by vendors whose tuning software's use the recommended approaches. Existing database tuning methodologies focus so much on the database that they downplay the role of other components such as the operating system and middle layer tier in overall performance of the application.

1.3. PURPOSE STATEMENT

The purpose of the project has been to develop a customized and integrated application and database tuning methodology from selected existing methodologies. The customized methodology will be holistic in nature and will take into consideration all components that have an impact on system performance.

The methodology will also be applied to solve a real world performance application issue.

1.4. SIGNIFICANCE OF THE STUDY

The study developed a customized integrated application and database tuning approach that can assist both novices and experts to tune their applications

Organizations can include the developed methodology as standard operating procedures of Database Administrators for directing tuning activities

The study has also demonstrated how the methodology can be used in a real world performance problem scenario.

1.5. OBJECTIVES

The overall object of the study was to propose a methodology to find a solution as opposed to recommending specific remedies for specific problems. The reason for this is that solutions belong to a problem and there is no single problem which is always the culprit for poor performance in database applications. The proposed methodology was also applied to fine tune a poorly performing payroll application for a large organization

Specific objectives of the project were:-

1. To study systems performance tuning especially in database oriented applications.
2. To review the existing application and database tuning methodologies with an intention of identifying gaps.
3. To develop a customized integrated holistic tuning approach that addresses identified gaps.
4. To apply the customized methodology in a real world performance problem.

1.6. RESEARCH QUESTIONS

1. What are the various factors/bottleneck that contribute to poorly performing database applications?
2. Do existing fine tuning methodologies have gaps that can be addressed?
3. How can an integrated customized database and application tuning methodology be developed to address identified gaps?
4. Can the methodology developed be applied in a real world performance problem scenario?

1.7. SCOPE AND LIMITATION

The study focused on reactive database application tuning or bottleneck elimination as opposed to proactive tuning. In reactive tuning a performance optimization exercise is carried out where system performance has already degraded to below users' expectation or set baseline. Proactive tuning on the other hand is continuously monitoring system behavior and resource usage and occasionally making configuration changes. (Immanuel, 2011)

The study also focuses on transactional oriented applications where response time and throughput is a critical factor to overall system performance.

1.8. EXPECTED PROJECT OUTCOMES AND THEIR POTENTIAL IMPACTS

The result will be a customized tuning methodology that can be used in identifying and subsequent resolution of bottlenecks. Some of the distinct benefits and advantages of utilizing the methodology include:

- Clearly documented systematic steps to guide an application tuning exercise.
- Critical examination of the main tiers that have an impact on system performance without downplaying or overemphasizing role played by any component.
- Guidance on potential bottlenecks to look out for in the various tiers.
- For identified bottlenecks best approach in resolving them are recommended.
- Development of an integrated application and database tuning methodology.
- End result of application of the methodology is a system with the following characteristics
    - Better response times: The tasks are completed in a smaller amount of time.
    - Higher Throughput: Faster execution of tasks which means increased throughput. A large number of tasks can be performed in a given unit of time.

CHAPTER 2: LITERATURE REVIEW

2.1. INTRODUCTION

Performance tuning or optimization of system performance is mainly concerned with improving the system to perform better. Performance tuning as explained in the scope and limitation section can either be proactive or reactive. In reactive tuning motivation for such an exercise can be referred to as a performance problem. This performance problem can be a crippling bottleneck that makes the system unable to meet business needs or completely unusable. Immanuel (2011) postulates as follows

> *"Usually, the purpose for tuning is to reduce resource consumption or to reduce the elapsed time for an operation to complete. Either way, the goal is to improve the effective use of a particular resource. In general, performance problems are caused by the overuse of a particular resource. The overused resource is the bottleneck in the system. "*

Performance problems also arise when a system is unable to scale gracefully with an increased workload. Designers, developers and administrators usually aim for linear scalability this is whereby system throughput is directly proportional to the computer resources (number of CPUS', memory).

Several components or tiers contribute to the overall system performance, these are:-

- Application tier
- Middle layer/Application server
- Network tier
- Database tier
- Hardware and operating system tier

Identification of which tier is the bottleneck and subsequent tuning of the tier and any other related tier is key to resolving performance problems in applications.

2.2. CURRENT TUNING FOCUS

Due to the potential possibility of degraded system performance affecting business negatively various researchers and vendors have focused their energies on system performance tuning. Majority of these are:-

- Mainstream database vendors for example, oracle,db2,SQL server
- Performance tuning experts for example Burleson Consulting, Quest software
- Mainstream ERP vendors for example PeopleSoft ,SAP
- Performance tuning Books for example Database Tuning by Dennis Sasha and The Data Access Handbook by John Goodson

Depending on the vendor or researcher directing the tuning exercise focus seems to shift from the application server, middle tier to database tier.

Quest (2010) asserts that SQL optimization, indexing, and database parameters changes offer the best opportunity to improve database and application performance. Careful tuning of the two results to considerable savings by delaying expensive hardware upgrade, avoiding time-consuming database redesigns which in turn lead to systems been able to meet the business needs.

Quest (2010) further states that poor performing SQL statements and indexes are responsible for 60 to 90 percent of application performance problems. He claims that according to industry experts, SQL activities typically consume as much as 70% of the system resources in a database server. In many cases, resource consumption by SQL activity can be as reach as high as 80-90%.

.John (2009) on the other hand focuses on the middle layer, he takes the stance that within the last 10 years, the vast majority of slowdown in applications that are database oriented has shifted from slowdown on the database server itself to slowdown on (or caused by) the application server. Problems at the application level include database driver issues, connection pooling and erroneous configurations. He further emphasizes that breakthrough database and application performance can be achieved by optimizing middleware and connectivity He claims that traditional database tuning isn't nearly enough to solve the performance problems that applications experience and that 75-95% of the time it takes to process a data request is typically spent in the database middleware.

Consider, another view expressed by Isam (2011) is that when troubleshooting a poorly performing application, investigations should be done to identify where majority of the application time is being spent. He concurs with Quest (2010) that 80 % of performance issues are database related.

Toadworld (2010) stresses the importance of not underestimating application tuning. The author claims that as much as 80 percent of performance gains will be accomplished by application tuning through effective writing of SQL statements.

Immanuel (2011) proposes a more balanced focus when he states as follows:-

> "*The ultimate measure of success is the user's perception of system performance. The performance engineer's role is to eliminate any bottlenecks that degrade performance. These bottlenecks could be caused by inefficient use of limited shared resources or by abuse of shared resources, causing serialization. Because all shared resources are limited, the goal of a performance engineer is to maximize the number of business operations with efficient use of shared resources."*

2.3. REVIEW OF CURRENT TUNING METHODOLOGIES

ISAM (2011) advances the following application tuning methodology to solve Performance Issues related to Ebusiness suite. The methodology has some tuning steps that are applicable to most database oriented applications.

1. The problem should be defined clearly and a clear understanding of the performance problem and where the application is spending most time obtained.

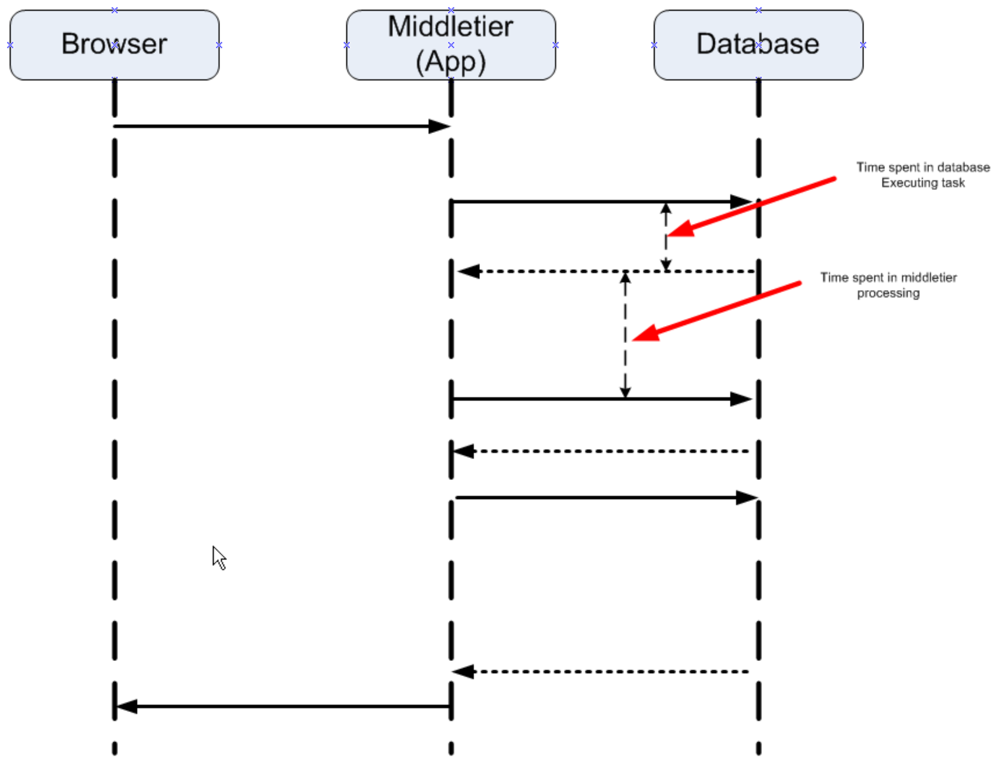
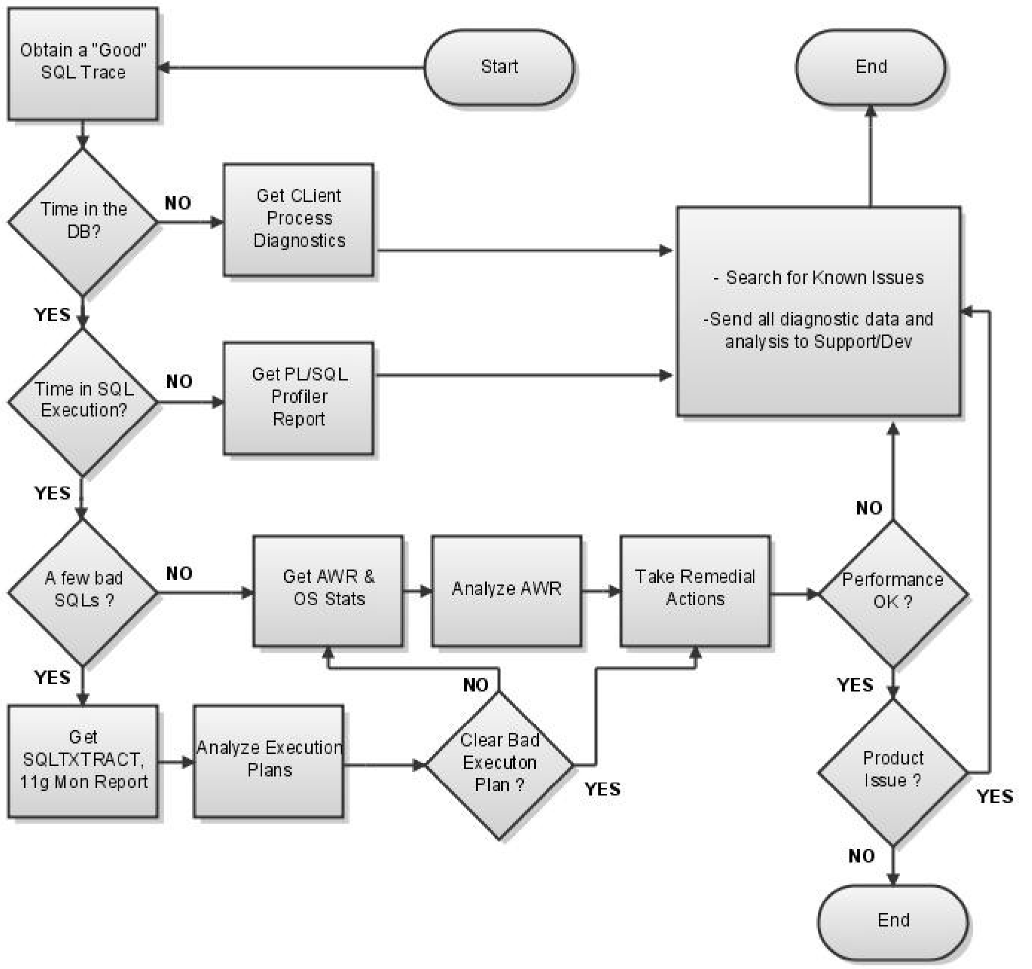

Browser
Middletier (App)
Database
Time spent in database Executing task
Time spent in middletier processing

Figure 1: Problem Identification

2. The right data to analyze the performance issue should be gathered; the use of specific oracle database tools is demonstrated.

3. The root cause of the problem should be isolated and possibly additional data gathered. Investigations of where the time is going? Who is consuming the most time? Why is that happening? Should be carried out, he states that 80% of performance issues are database related.



Obtain a "Good" SQL Trace
Start
End
Time in the DB?
NO
Get CLient Process Diagnostics
YES
- Search for Known Issues
-Send all diagnostic data and analysis to Support/Dev
Time in SQL Execution?
NO
Get PL/SQL Profiler Report
YES
NO
A few bad SQLs ?
NO
Get AWR & OS Stats
Analyze AWR
Take Remedial Actions
Performance OK ?
YES
NO
YES
Get SQLTXTRACT, 11g Mon Report
Analyze Execution Plans
Clear Bad Executon Plan ?
YES
Product Issue ?
YES
NO
End

Figure 2: Root Cause Analysis

4. A search for a known solution or workaround that addresses the root cause of the problem should be conducted.
5. If it is a product issue, it should be passed on to the right information to support or the development team through the regular channels
6. A temporary workaround should be identified to alleviate the issue while a product fix is obtained.

An alternative tuning methodology is proposed by Toadworld (2010), though biased to the database side it nevertheless contains general guidelines that can be used to tune applications.

The methodology starts by emphasizing the need to establish a set of quantifiable goals that directly relate to a reason for tuning. The goals ought to be kept in mind as modifications are evaluated and considered for the system. The tuning goals should be specific and measurable rather than generic.

The methodology stresses that the operating system should be performing at its peak before any attempt to tune the database are conducted.

The following structured methodology is proposed:-

1. **Optimization of the Application Workload**

   Applications should contain effective SQL statements; this is achieved by utilizing hints, indexes, and bind variables whenever necessary to obtain optimal performance.
   The application developer should have a solid understanding of SQL processing, including:

   - DML (Data Manipulation Language)
   - DDL (Data Definition Language)
   - Transaction control
   - Shared SQL and PL/SQL areas
   - Optimizer modes
   - Parallel query

2. **Tuning Contention**

   The author argues that contention occurs when a process competes with another process for the same resource simultaneously. This causes processes to wait for a resource on the database system and can have an effect on performance. Investigations should be carried out to identify and resolve contentions.

3. **Minimize Physical IO**

Careful sizing of memory structures to allow sufficient information to be stored in memory is advocated for. This is because memory access is significantly faster than disk access, it is always better to satisfy requests for information in memory than from disk. Tuning memory allocation involves proper memory distribution to each database area while ensuring that paging and swapping is not occurring at the operating system level.

4. **Optimize Physical IO**

   Disk contention usually occurs when multiple processes try to access the same disk simultaneously. When the maximum number of accesses to a disk has been reached, other processes will need to wait for access to the disk. The author emphasizes the importance of ensuring database files are evenly distributed throughout the operating systems to ensure disk contention does not occur.

5. **Best Practices**

The following of best practices as guidelines to ensure that tasks are done in a way that is recognized as generally acceptable is advocated for. Caution is however given that special considerations in specific environments can preclude the use of one or more of recommended best practices.

Immanuel (2011) on behalf of oracle proposes a more balanced methodology that is highly iterative. He notes that a common pitfall in performance tuning is to mistake the symptoms of a problem for the actual problem itself. The author suggests that performance statistics indicate the symptoms, and that identifying the symptom is not sufficient data to implement a remedy. An example of slow physical I/O which is usually caused by poorly-configured disks is given; it could be that contrary to the norm a significant amount of unnecessary physical I/O on those disks is being caused by poorly-tuned SQL.

According to Immanuel (2011) different forms of contention are symptoms that can be fixed by making changes to three key areas:-

- Changes in the application, or the way the application is used
- Changes in the database
- Changes in the host hardware configuration Often

The author holds the opinion that the most effective way of resolving a bottleneck is to change the application
A top down performance tuning methodology is proposed, while focus is on the database, the methodology nevertheless has very good general application tuning guidelines.

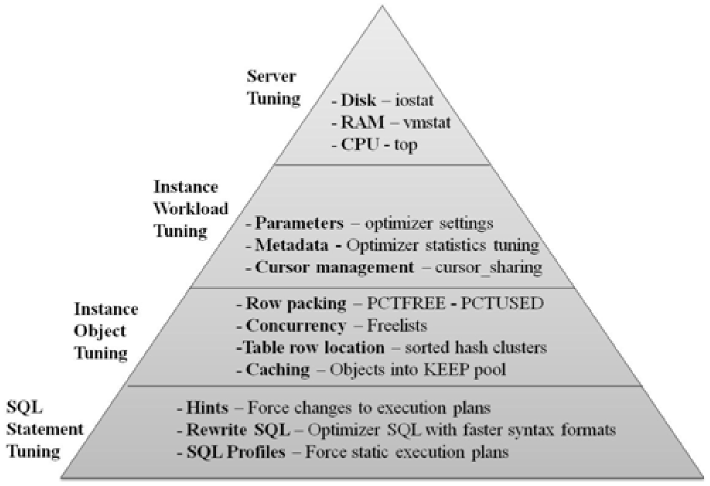

Server Tuning
- Disk – iostat
- RAM – vmstat
- CPU - top
Instance Workload Tuning
- Parameters – optimizer settings
- Metadata - Optimizer statistics tuning
- Cursor management – cursor_sharing
Instance Object Tuning
- Row packing – PCTFREE - PCTUSED
- Concurrency – Freelists
-Table row location – sorted hash clusters
- Caching – Objects into KEEP pool
SQL Statement Tuning
- Hints – Force changes to execution plans
- Rewrite SQL – Optimizer SQL with faster syntax formats
- SQL Profiles – Force static execution plans

Figure 3: Top down Tuning Methodology

Steps in the oracle performance improvement method

The following initial standard checks are performed:

1. Candid feedback is obtained from the users. Performance project's scope and subsequent performance goals are determined; performance goals for the future are also identified.

   i. A full set of operating system, database, and application statistics from the system are obtained when the performance is both good and bad.

   "Missing statistics are analogous to missing evidence at a crime scene: They make detectives work harder and it is more time-consuming."

   ii. A sanity check of operating systems of all computers involved with user performance is carried out. While conducting the check hardware or operating system resources that are fully utilized are identified. A list of any over-used resources is made as symptoms for analysis later. In addition, all hardware is checked to confirm there're no errors or diagnostics.

2. A check for the top ten most common mistakes with Oracle Database is carried out. It is determined if any of these are likely to be the problem.

3. A conceptual model of what is happening on the system using the symptoms as clues to understand what caused the performance problem is built.

4. A series of remedy actions and the anticipated behavior to the system is proposed, the actions are in turn applied in the order that can benefit the application the most. It is recommended that a golden rule in performance work is to change one thing at a time and then measure the differences, however this is sometimes impracticable in real time systems since the accompanying downtime is unacceptable. Multiple changes can be applied at the same time as long they are isolated so that the effects of each change can be independently validated. .

5. Changes made are validated to confirm if they have achieved desired effect, and if the user's perception of performance has improved. If this is not the case more bottlenecks are identified and refinement of the conceptual model continues until understanding of the application becomes more accurate.

6. The last three are repeated until performance goals are met or become impossible due to other constraints.

Immanuel (2011) concludes that this method identifies the biggest bottleneck and it uses an objective approach to performance improvement. The focus of the methodology is on making large performance improvements by increasing application efficiency and eliminating resource shortages and bottlenecks. The author claims that minimal (less than 10%) performance gains are made from instance tuning, and large gains (100% +) are made from isolating application inefficiencies.

2.4. SUMMARY AND CRITIQUE OF REVIEWED TUNING METHODOLOGIES

Tuning Methodology	**Critique**
ISAM (2011) application tuning methodology	• More focused on tuning specific application (Ebusiness suite) though some tuning steps are applicable • Does not address all system components adequately for example hardware and operating system
Toadworld (2010) database tuning methodology	• Highly biased on tuning the database
Immanuel (2011) application and database	• Gives a more balance approach to tuning all tiers that affect system performance, however, biased to the oracle database.

Table 1: Summary and Critique of Reviewed Tuning Methodologies

From the reviewed methodologies and other literature and sources a hybrid one will be developed that picks the best tuning steps and tips to come up with a holistic tuning approach that can be applied to tuning any database oriented application. The methodology will in turn be applied in a real world problem scenario.

2.5. CONCEPTUAL MODEL

A conceptual model is a mental model that captures ideas in a problem domain it represents 'concepts' (entities) and relationships between them. The aim of a conceptual model is to express the meaning of terms and concepts used by domain experts to discuss the problem, and to find the correct relationships between different concepts (Fowler (1997).

From the literature reviewed the model below (figure 4) will guide development of the integrated tuning methodology and subsequent tuning exercise.

As depicted in the conceptual model the methodology will focus on identification and resolution of bottlenecks in all tiers that have impact on performance of database application systems. The Tuning exercise is an iterative one and continues until all performance goals are achieved or a compromise is made.

Tuning at the application layer will borrow heavily from ISAM (2011) application tuning methodology while Immanuel (2011) application and database tuning methodology will contribute immensely to database tuning.

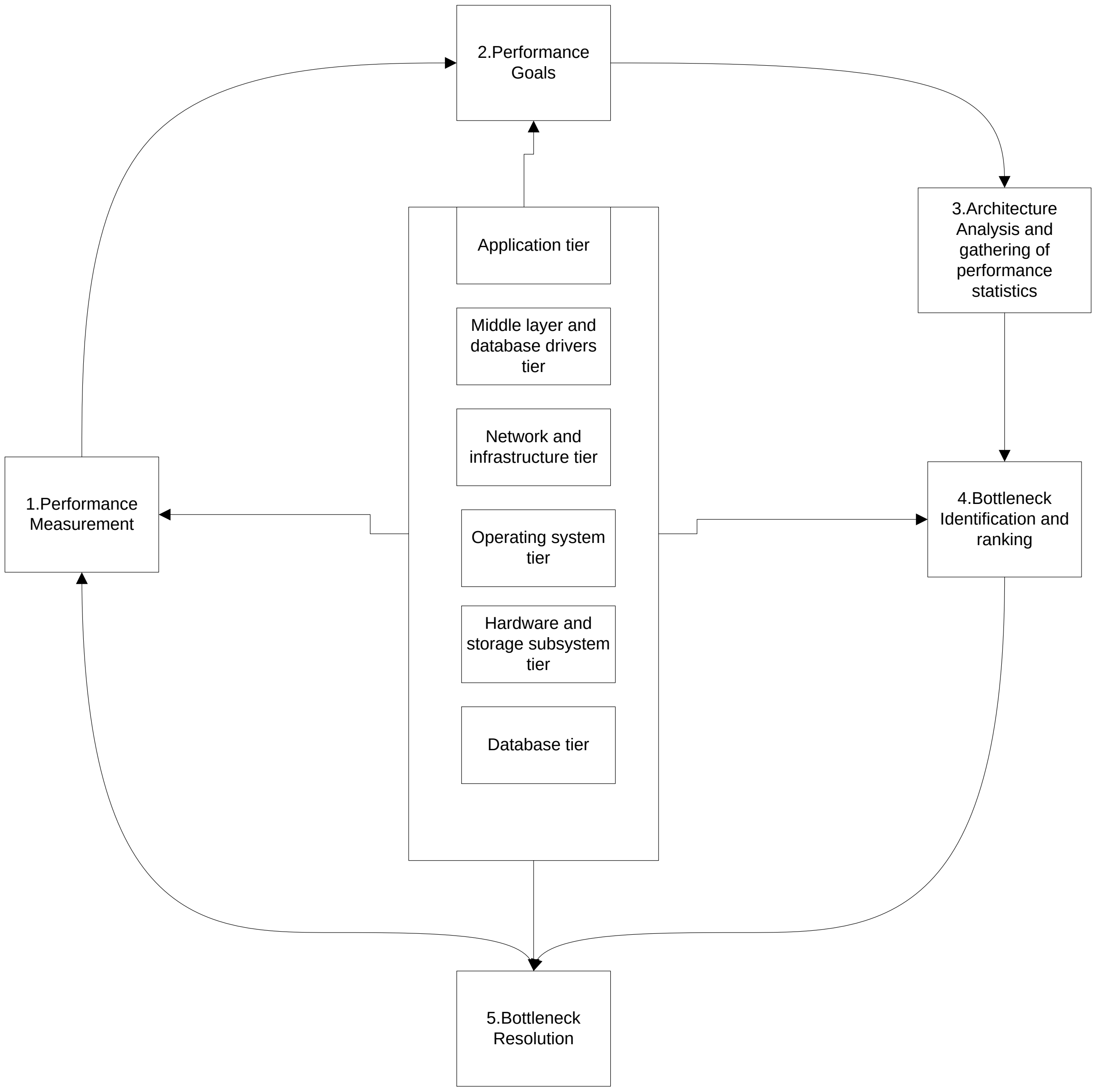

2.Performance Goals
3.Architecture Analysis and gathering of performance statistics
Application tier
Middle layer and database drivers tier
Network and infrastructure tier
Operating system tier
Hardware and storage subsystem tier
Database tier
1.Performance Measurement
4.Bottleneck Identification and ranking
5.Bottleneck Resolution

Figure 4: Conceptual Model

CHAPTER 3: METHODOLOGY

3.1. RESEARCH DESIGN

Research design refers to the overall strategy that will be used to integrate the different components of the study in a coherent and logical way, thereby, ensuring research questions are addressed effectively; it constitutes the blueprint for the collection, measurement, and analysis of data.

In the study two key research design types were utilized. These are

1. Content analysis
2. Experimental design

(University of Texas,2010) in their article titled "conduct research" defines content analysis as follows:

"Content analysis is the systematic examinations of written or recorded communication in order to break down, identifies, and analyze the presence or relations of words, word sense, characters, sentences, concepts, or common themes. The focus of the analysis should be a critical examination, rather than a mere description, of the content. Examples of content include student journals, essays, online discussions, or any form of written, visual, or oral communication.

Content analysis works best when the purpose is to gain insight into a precise and focused research problem or topic. It can help you to recognize patterns that you might miss using other methods"

Content analysis was utilized in the evaluation and analysis of existing fine methodologies. Gaps identified were addressed when developing the intergrated holistic methodology.

Experimental design was used when applying the customized tuning methodology to the poorly performing payroll application. Independent and dependent variables were identified. Performance metrics were measured before applying the methodology and after to determine its effectiveness. Some of the identified variables are:-

Independent Variables

This variables will be obtained from tiers which have bottlenecks, they include

Tier	Variable
Operating system tier	Memory available Number of CPUs
Network Tier	Network Speed
Database Tier	Memory allocation Number of indexes

Table 2: Independent Variables

Dependent Variables

Since the goal of the tuning exercise is to achieve system performance in the eyes of the user, most of these variables will be derived from the application tier, they include:-

1. Throughput (Number of payroll transactions processed per second)
2. Response time (How long it takes to process a given number of payroll records)

The table below shows relationship between the independent and dependent variables, that is, which independent variables impact on which dependent variable.

Independent Variables	Dependent Variables
Memory available	Throughput Response time
Number of CPUs	
Network Speed	
Memory allocation	
Number of indexes	

Table 3: Independent and Dependent Variables

During experimentation as depicted in the figure below values of Independent variables which will have been identified as bottlenecks by utilization of methodology were altered and effect of the alterations noted on the dependent variable.

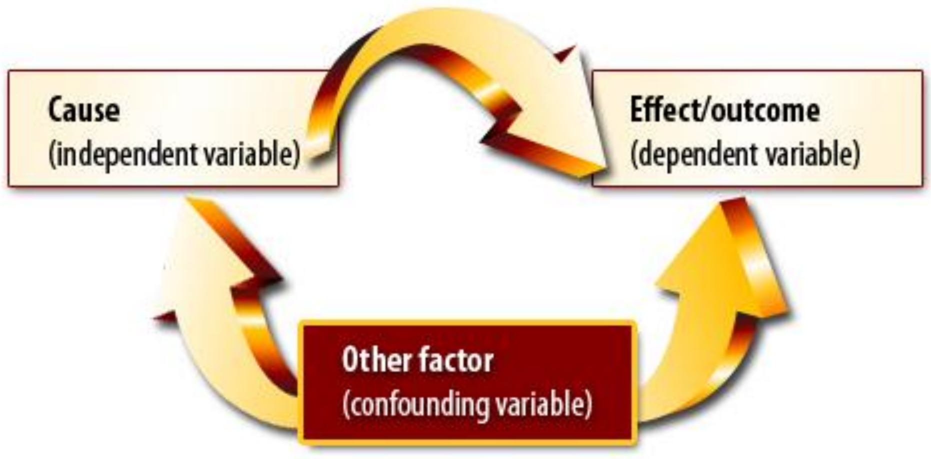


Figure 5: Variables

In developing the customized tuning approach and testing the same with a payroll application that is currently experiencing performance problems, several distinct phases were conducted, the phases ensured that various systems components that have an impact on system performance were examined and the subsequent tuning approach was tested. The phases were:-

1. Study of systems performance tuning in general.
2. Review of current performance tuning methodologies
3. Development of a customized database application methodology which focuses on
    a. Measurement of system performance before tuning exercise.
    b. Identification of tiers with bottlenecks
    c. Iterative resolution of identified bottlenecks
    d. Measurement of system performance after tuning exercise

    The methodology addresses all tiers that have impact on system performance, these are:-
    a. Application tier
    b. Middle layer and database drivers tier
    c. Network and infrastructure tier
    d. Operating system tier.
    e. Hardware and storage subsystem tier
    f. Database tier
4. Application of methodology on a poorly performing payroll processing system.
5. Analysis and evaluation of tuning approach

3.2. DATA SOURCES

The main data source was a poorly performing payroll application for an organization with six thousand employees. Other data sources include operating system, network, middle layer and connectivity, storage systems and the database.

Data from payroll application

1. Response time - Response time is a measure of how quickly the system responds to a request. It is how long it takes to finish a given task, for example how long does the payroll application take to process 1000 records. In the poorly performing payroll application 200 payroll records are processed in 80 seconds in the current environment.
2. Throughput - Throughput is a measure of how much work the system can do in a given period of time, for example how many payroll records can the application process in one minute
3. Number of payroll records
4. Number of payroll runs

Data from operating system tier

1. Memory settings
2. Number of CPUs
3. Operating system architecture 32 bit vs. 64 bit
4. Input Output (IO) settings (Max file open handles, virtual disks)
5. Swapping statistics

Data from middle layer and connectivity tier

1. Type of database driver in use
2. Connection pooling used

Data from hardware and storage subsystem tier

1. Redundant Array of Independent Disks (RAID) settings (Raid 0,Raid 0+1,Raid 5)
2. Processor type, cache sizes

Data from Database tier

1. Memory allocations
    a. System global area (SGA) sizing.
    b. Memory management, self-tuning vs. manual
2. Wait analysis
    a. Events that contributing to most wait time
3. Poorly performing SQL statements
4. Database statistics, for example from ADDM output

3.3. DATA COLLECTION

In measuring performance of the payroll application before, during and after performance tuning exercise the following data was collected.

Response time – Time to taken to process a payroll with six thousand records.

Throughput – Number of payroll records processed in one secs.

For each of the above values the payroll was run four times (four monthly processes) and the mean values taken, payroll was run for 200 employees and different payroll historical data were considered, such Different sets of Response time and throughput were obtained by varying the independent variables which were identified as contributing to bottlenecks. The set and corresponding Independent variables fed into the analysis section.

3.4. DATA COLLECTION TOOLS

The following tools were used to collect performance data

1. Network monitoring and analysis tools - netsat, GNOME monitor these tools were used to obtain the network speeds and latency and determine if the network is a bottleneck.
2. Operating system monitoring and analysis tools – top. ps ,free and sar, these tools will measure operating system metrics such cpu utilization, memory utilization, swapping levels.
3. Database monitoring and diagnostics tools –ADDM and Enterprise manage, the tools will provide database diagnostics information such as memory sizing, indexes, execution plans.
4. Hardware monitoring and diagnostics tools –vmstat, iostat, these tools will provide insight on how the underlying hardware layer is performing, key values will be number of page reads and page writes.

3.5. DATA ANALYSIS

In analysis of the tuning exercise key questions were:-

1. Was the methodology concise and easy to apply?
2. Were the tuning goals achieved?
3. Was the application tuned within acceptable time limit?

By analyzing data gathered using the above mentioned tools bottlenecks were identified and options for resolving them identified.

By comparing performance baselines before and after resolution of bottlenecks, percentage improvement in response time was calculated for different of options of evaluating the identified bottlenecks. Some performance tuning tools were detailed enough to give expected improvement in performance.

A formulae for ranking the identified bottlenecks was formulated and used to rank the bottlenecks.

3.6. LIMITATIONS OF METHODOLOGY

A lot of literature has been written on system performance tuning. One of the key phases of the methodology is review of literature. There is a danger of too much time being spent on this phase which can impact negatively on the overall time line. Caution was exercised to ensure literature from published journals, tuning experts and major technology players was not given preference and too much time was not spent reviewing works whose authenticity and validity cannot be ascertained.

CHAPTER 4: RESULTS

4.1. INTEGRATED DATABASE APPLICATION TUNING HOLISTIC METHODOLOGY

In development of the integrated holistic methodology a lot of material is borrowed heavily from the oracle improvement performance method (ISAM, 2011) and (Oracle- B10500_0, 2002) which is stated in the critic of existing methodologies was found to offer a more balanced approached. Tools for gathering performance statistics in the various tiers and for identifying and assisting in elimination of bottlenecks are also suggested. Most common bottlenecks in each tiers are also given and guidelines on how they can be resolved also suggested. The developed methodology has eight steps which are defined below.

Step 1: The first step is to get feedback from the user on how the system is behaving, user perception of the system is very important since it can be used to set critical success factors for the tuning exercise Isam (2011). Typical users feedback include ;-

i. “The system takes a long time to generate a receipt”
ii. “The search page in our ecommerce website takes too long to retrieve products leading to the customer giving up and consequently loss of sales”

Step 2: Collect baseline Performance measurement of the application when it is performing below users’ expectation and when the system is performing well. Since we are focusing on the users experience, performance of the database application is best measured using the response time, (how long it takes to perform a certain task) examples of response time include :-

i. One payroll record is calculated in 1 sec, the response time in this case is 1 second.
ii. Each page in website loads in an average of 15 secs, response time is 15 secs.
iii. Throughput of the application is also measured, throughput is amount of work done in a unit amount of time, and examples include:-
iv. In one minute 60 payroll records are calculated, throughput is 60 records.

Step 3: Based on the users feedback and the baseline measurement collected realistic performance goals are set, success in a performance tuning exercise is best defined in terms of real business goals rather than system statistics Isam (2011). The performance goals should be quantitative rather than quantitative to ease task of determining improvement in performance. Isam (2011) goes ahead to give the following samples of very good realistic business goals :-

i. "The billing run must process 1,000,000 accounts in a three-hour window."
ii. "At a peak period on a Web site, the response time must not exceed five seconds for a page refresh."
iii. "The system must be able to process 25,000 trades in an eight-hour window."

Step 4: Analyzing system infrastructure, in this step the following data should be gathered:-

Architecture of the system should be well understood; the different components to make the system work and their interrelationships should be well understood. Detailed configuration information of all servers in use by the application should be gathered, this include

i. Different kind of servers in use for example web application servers, database servers, mail servers.

ii. For each of the server identified the following information should be collected

  a. Hardware architecture
    i. 64 bit vs 32 bit architectures
    ii. Memory available
    iii. Disk type (SCSI,SAS,SATA, SSD)
    iv. Disk speeds, 10k,15k
    v. Raid configurations, RAID 0,10,1,5
    vi. CPU architecture, cores, sizes and level of cache, clock speed
  b. Operating systems configurations
    i. Swap allocation
    ii. Network card settings
    iii. Kernel settings
    iv. Other optimization settings
  c. Network settings
    i. Buffer settings

Step 5: Resource utilization and performance statics of all the tiers involved should be gathered, this information will be useful in identifying bottlenecks, information to collect include :-

i. Operating system and hardware statistics

Operating system statistics provide information on the usage and performance of the main hardware components of the system, as well as the performance of the operating system itself. This information is crucial for detecting potential resource exhaustion, such as CPU cycles and physical memory, and for detecting bad performance of peripherals, such as disk drives.

Operating system statistics are only an indication of how the hardware and operating system are working. (Oracle- B10500_0, 2002)

The following statistics should be collected for each server utilized by the system, (Oracle- B10500_0, 2002) gives a good description of why the statistics are needed.

a. CPU utilization

CPU utilization is the most important operating system statistic in the tuning process. CPU utilization should be obtained for the entire system and for each individual CPU on multi-processor environments. Utilization for each CPU can detect single-threading and scalability issues.

Most operating systems report CPU usage as time spent in user space or mode and time spent in kernel space or mode. These additional statistics allow better analysis of what is actually being executed on the CPU.

b. Memory utilization and Virtual memory Statistics

Memory is faster that disk access, most applications improve immensely by having more memory available to them since they don't have to result to virtual memory. High memory utilization might be an indication of inadequate memory available. Improper utilization of available memory or even memory leaks.

Virtual memory statistics should mainly be used as a check to validate that there is very little paging or swapping activity on the system. System performance degrades rapidly and unpredictably when paging or swapping occurs.

c. Disk utilization and I/O statistics

Since applications and databases resides on a set of disks, the performance of the I/O subsystem is very important to the performance of the applications. Most operating systems provide extensive statistics on disk performance. The most important disk statistics are the current response time and the length of the disk queues. These statistics show if the disk is performing optimally or if the disk is being overworked. If a disk shows response times over 20 milliseconds, then it is performing badly or is overworked. This can be immediately singled out as one of the potential bottlenecks. If disk queues start to exceed two, then the disk is also a potential bottleneck of the system.

d. Network Utilization

Network statistics can be used in much the same way as disk statistics to determine if a network or network interface is overloaded or not performing optimally. In today's networked applications, network latency can be a large portion of the actual user response time. For this reason, these statistics are a crucial debugging tool.

To gather the above operating system statistics, various tools are available that can be used by a performance analyst, below is an overview of some of the available tools for Linux based and windows operating systems. For a listing of some of the available monitoring tools, refer to APPENDIX A.

ii. Database performance statistics

Database statistics provide information on the type of load on the database, as well as the internal and external resources used by the database. When database resources become exhausted, it is possible to identify bottlenecks in the application.

To collect database performance statistics in addition to vendor specific tools, each database vendor provides a performance statistics gathering and tuning tools.

For a listing of some of the available database statistics gathering and monitoring tools, refer to APPENDIX A.

iii. Application statistics

Application statistics are reputed to be the most difficult statistics to gather, nevertheless, they are the most important statistics in measuring any performance improvements made to the system. At a minimum, application statistics should provide a daily summary of user transactions processed for each working period. More complete statistics provide precise details of what transactions were processed and the response times for each transaction type. Detailed statistics also provide statistics on the decomposition of each transaction time spent in the application server, the network, the database, and any other involved tier (Oracle- B10500_0, 2002).

The best statistics require considerable instrumentation of the application. This is best built into the application from the start, because it is difficult to retrofit into existing applications. Various tools exist for gathering application statistics from various vendors, however, most of these tools are nor generic, that is, they cannot be used to gather statistics for applications developed in all platforms, they are specific to development platforms such as dot net applications, J2EE applications and web based applications. Some can however give statistics for all windows applications.

Tools that gather application performance statistics , diagnose and test applications and also help in identifying bottlenecks are often referred to as application profiling tools, some of these tools help performance engineers determine how much memory is being used and also identify memory leaks (Misty,2012) some of the available tools are :-

For a listing of some of the available application gathering and monitoring tools, refer to APPENDIX A

Step 6: Identification of bottlenecks

A bottleneck is any resource hardware, network, or software that limits the performance of an application. Bottlenecks directly affect performance and scalability by limiting the amount of data throughput or restricting the number of application connections. (Oracle-RBI, 2010)

Impact of performance bottlenecks

i. Deterioration of response time to almost unacceptable levels
ii. Inefficient resource utilization
iii. Application does not scale as required
iv. Decrease in throughput that is, less number of transactions is processed per unit time.
v. Loss in business revenue due to customer dissatisfaction

A bottleneck can reside in any of the tiers web server, application server, database server and network resources. Most applications have the following architecture and some can miss some layers such as the web and load balancing tier for non-web based systems.

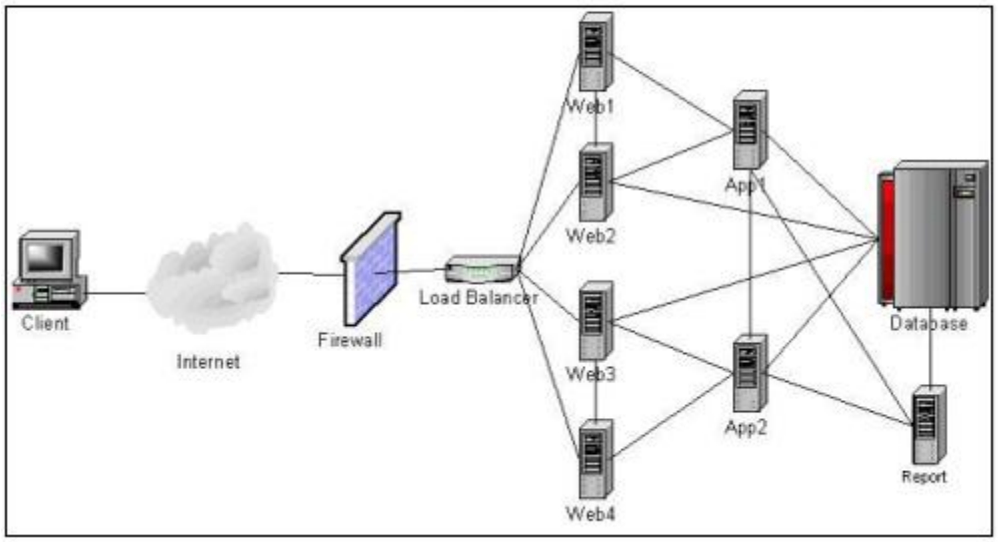


Figure 6:N-tier physical architecture

The statics collected in the prior stage provide vital information in identifying the tiers with bottlenecks.

It is tough to test each and every component's performance thoroughly, and a guided search is the best way out. Server hardware and network resources are usually assumed to be the main culprits for lower performance. Server upgrades are usually considered as the best source of performance optimization and it is not uncommon for engineers to solve performance problems by throwing more hardware (Thomas, 2012), sometimes it is a cheaper option in the short term and it some instances it is not work especially if the application code is very inefficient.

In each of the tiers the following guidelines can be followed to identify potential bottlenecks:-

i. Operating system tier and hardware tier

It is always best to consider operating system statistics as a diagnostic tool just like doctors use body temperature, pulse rate, and patient pain when making a diagnosis and not rush into installing more hardware resources immediately. This is a reactionary response to a series of symptoms shown in the operating system statistics. For example high memory utilization might be a result of a memory leak in an application and adding more memory to the server will not solve the problem.

Linux operating system will be used as guide to illustrate how bottlenecks in the operating system can be identified; most examples are derived from (Eduardo, 2007) in the white paper "Linux Performance and Tuning Guidelines"

a. CPU bottlenecks

For servers whose primary role is that of an application or database server, the CPU is a critical resource and can often be a source of performance bottlenecks. High CPU utilization does not always mean that a CPU is busy doing work; it might be waiting on another subsystem. The system should be looked as a whole and at all subsystems because there could be a cascade effect within the subsystems.

Using the available operating systems tools a CPU bottleneck can be identified; one tool that can be used is uptime the uptime command can be used to see how long the server has been running and how many users are logged on, as well as for a quick overview of the average load of the server.
The system load average is displayed for the past 1minute, 5 minute, and 15 minute intervals. The optimal value of the load is 1, which means that each process has immediate access to the CPU and there are no CPU cycles lost. The typical load can vary from system to system. For a uniprocessor workstation, 1 or 2 might be acceptable, whereas values of 8 to 10 on multiprocessor servers can be observed. For example the output of uptime for a cpu strapped server, for the last 15 minutes is as follows.

```
18:03:16 up 1 day, 2:46, 6 users, load average: 182.53, 92.02, 37.95
```

The load averages on the server for the last 5,10 and 15 minutes are 182.53, 92.02, 37.95 which are very high.

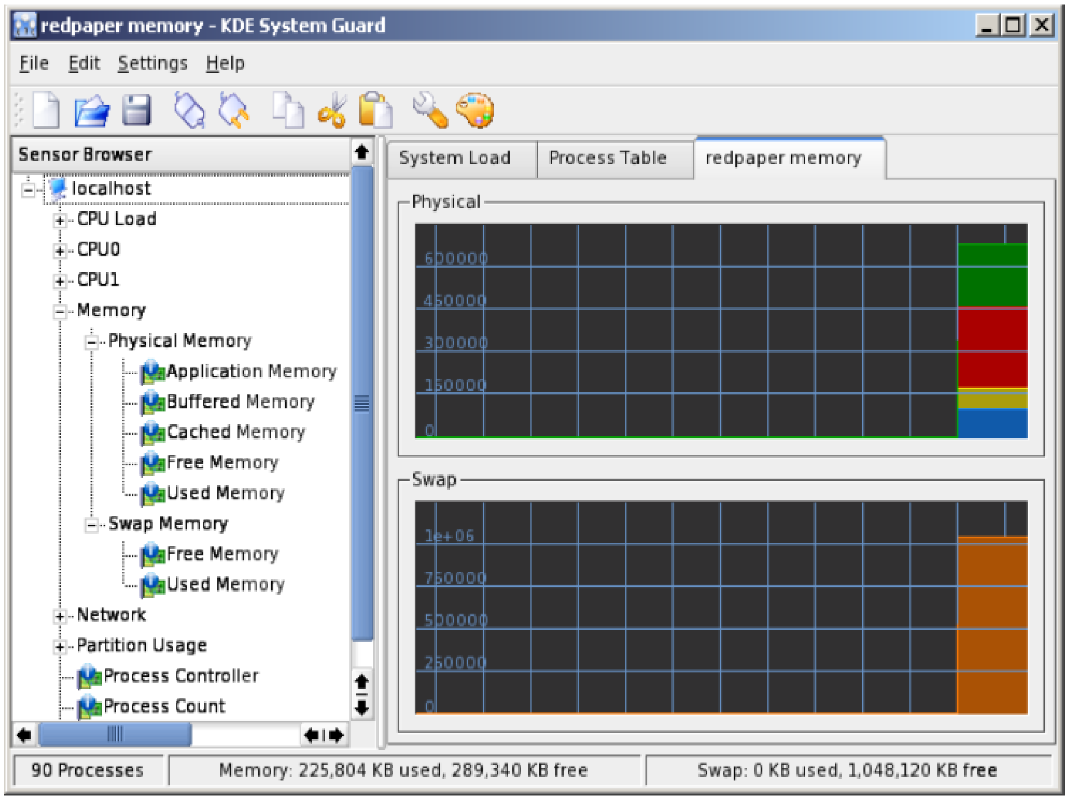
There is a common misconception that the CPU is the most important resource of the server. This is not always the case, and most servers are often over configured with CPU and under configured with disks, memory, and network subsystems. Only specific applications that are truly CPU intensive can take advantage of today’s high-end processors.

b. Memory bottlenecks

In Operating systems, many programs run at the same time. These programs support multiple
users, and some processes are more used than others and thus use more memories than others.

One of the key indicators of a memory bottleneck is excessive swapping and paging. Paging moves individual pages to swap space on the disk; swapping is a bigger operation that moves the entire address space of a process to swap space in one operation.

A key memory indicator used for analysis is amount of memory available; this indicates how much physical memory is available for use. If, after an application is started, this value decreases significantly, there might be a memory leak with the application. The free tool in Linux and resource monitor in windows can be used to analyze memory usage.



90 Processes | Memory: 225,804 KB used, 289,340 KB free | Swap: 0 KB used, 1,048,120 KB free

Figure 7: KDE System Guard memory monitoring

Number of Page faults can also be used to identify problems with memory; there are two types of page faults: soft page faults, when the page is found in memory, and hard page faults, when the page is not found in memory and must be fetched from disk. Accessing the disk will slow your application considerably. The sar –B command in Linux can provide useful information for analyzing page faults, specifically columns pgpgin/s and pgpgout/s.

Paging can be a serious performance problem when the amount of free memory pages falls below the minimum amount specified, because the paging mechanism is not able to handle the requests for physical memory pages and the swap mechanism is called to free more pages. This significantly increases I/O to disk and will quickly degrade a server's performance.

If a server is always paging to disk (a high page-out rate), more memory can be added. However, for systems with a low page-out rate, it might not affect performance.

If a memory bottleneck is identified, the following actions can remedy the situation

i. Tune the swap space using bigpages, hugetlb, shared memory.
ii. Increase or decrease the size of pages.
iii. Improve the handling of active and inactive memory.

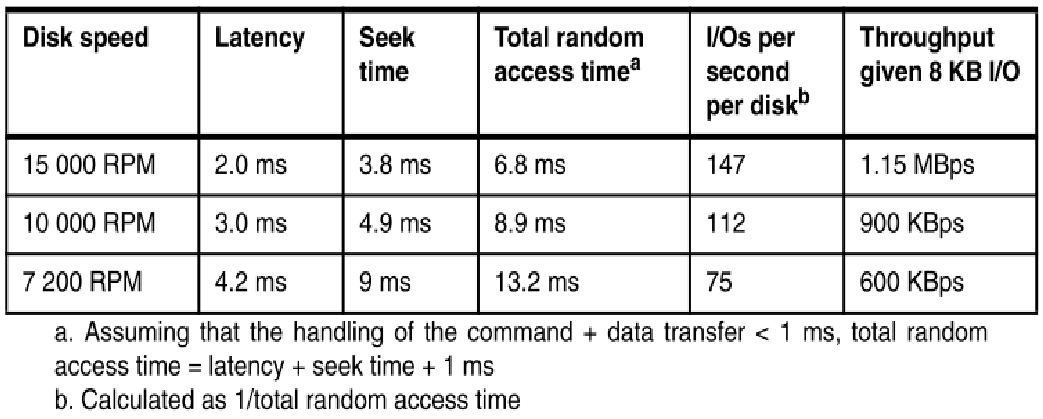
iv. Adjust the page-out rate.
v. Limit the resources used for each user on the server.
vi. Stop the services that are not needed
vii. Add memory.

c. Disk bottlenecks

The disk subsystem is often the most important aspect of server performance and is usually the most common bottleneck. However, problems can be hidden by other factors, such as lack of memory. Applications are considered to be I/O-bound when CPU cycles are wasted simply waiting for I/O tasks to finish.

The most common disk bottleneck is having too few disks. Most disk configurations are based on capacity. The disk subsystem is perhaps the most challenging subsystem to properly configure.
Besides looking at raw disk interface speed and disk capacity, it is also important to understand the workload. Is disk access random or sequential? Is there large I/O or small I/O? Answering these questions provides the necessary information to make sure the disk subsystem is adequately tuned.

Disk manufacturers tend to showcase the upper limits of their drive technology's throughput.
However, taking the time to understand the throughput of the workload will help to have true expectations of underlying disk subsystem.



Disk speed	Latency	Seek time	Total random access time[a]	I/Os per second per disk[b]	Throughput given 8 KB I/O
15 000 RPM	2.0 ms	3.8 ms	6.8 ms	147	1.15 MBps
10 000 RPM	3.0 ms	4.9 ms	8.9 ms	112	900 KBps
7 200 RPM	4.2 ms	9 ms	13.2 ms	75	600 KBps

a. Assuming that the handling of the command + data transfer < 1 ms, total random access time = latency + seek time + 1 ms
b. Calculated as 1/total random access time

Figure 8:True throughput for 8 KB I/Os for different drive speeds

The following general guidelines can be followed when designing a disk subsystem:-

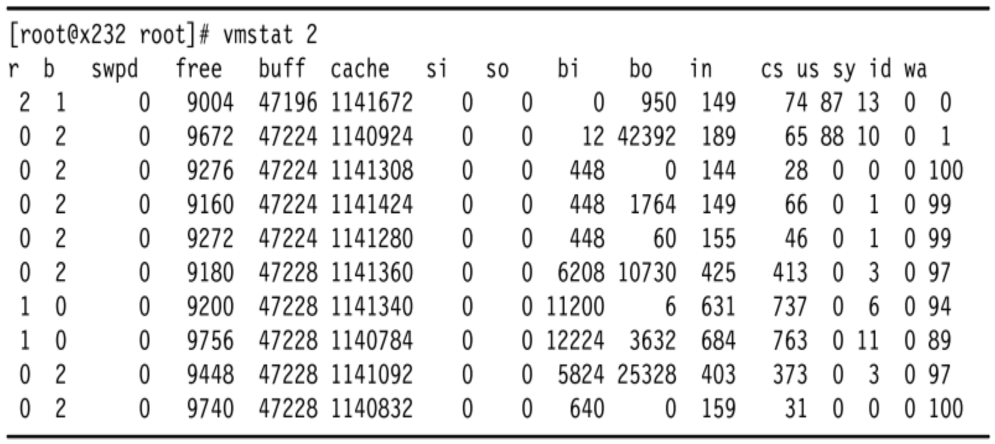
i. Random read/write workloads usually require several disks to scale.
ii. The bus bandwidths of SCSI or Fibre Channel are of lesser concern.
iii. Larger databases with random access workload will benefit from having more disks.
iv. Larger SMP servers will scale better with more disks. Given the I/O profile of 70% reads and 30% writes of the average commercial workload,

v. A RAID-10 implementation will perform 50% to 60% better than a RAID-5.
vi. Sequential workloads tend to stress the bus bandwidth of disk subsystems.
vii. Pay special attention to the number of SCSI buses and Fibre Channel controllers when maximum throughput is desired.
viii. Given the same number of drives in an array, RAID-10, RAID-0, and RAID-5 all have similar streaming read and write throughput.

Two of the ways that can be used to approach disk bottleneck analysis are real-time monitoring and tracing. **Real-time monitoring** must be done while the problem is occurring. This might not be practical in cases where system workload is dynamic and the problem is not repeatable.
However, if the problem is repeatable, this method is flexible because of the ability to add objects and counters as the problem becomes clear.

Tracing is the collecting of performance data over time to diagnose a problem. This is a good way to perform remote performance analysis. Some of the drawbacks include the potential for having to analyze large files when performance problems are not repeatable, and the potential for not having all key objects and parameters in the trace and having to wait for the next time the problem occurs for the additional data.

One way to track disk usage on a Linux system is by using the vmstat tool. The important columns in vmstat with respect to I/O are the bi (blocks sent to a block device (blocks/s)) and bo (blocks received from a block device (blocks/s)) fields. These fields monitor the movement of blocks in and out of the disk subsystem. High values signify high I/O activity. Having a baseline is key to being able to identify any changes over time.

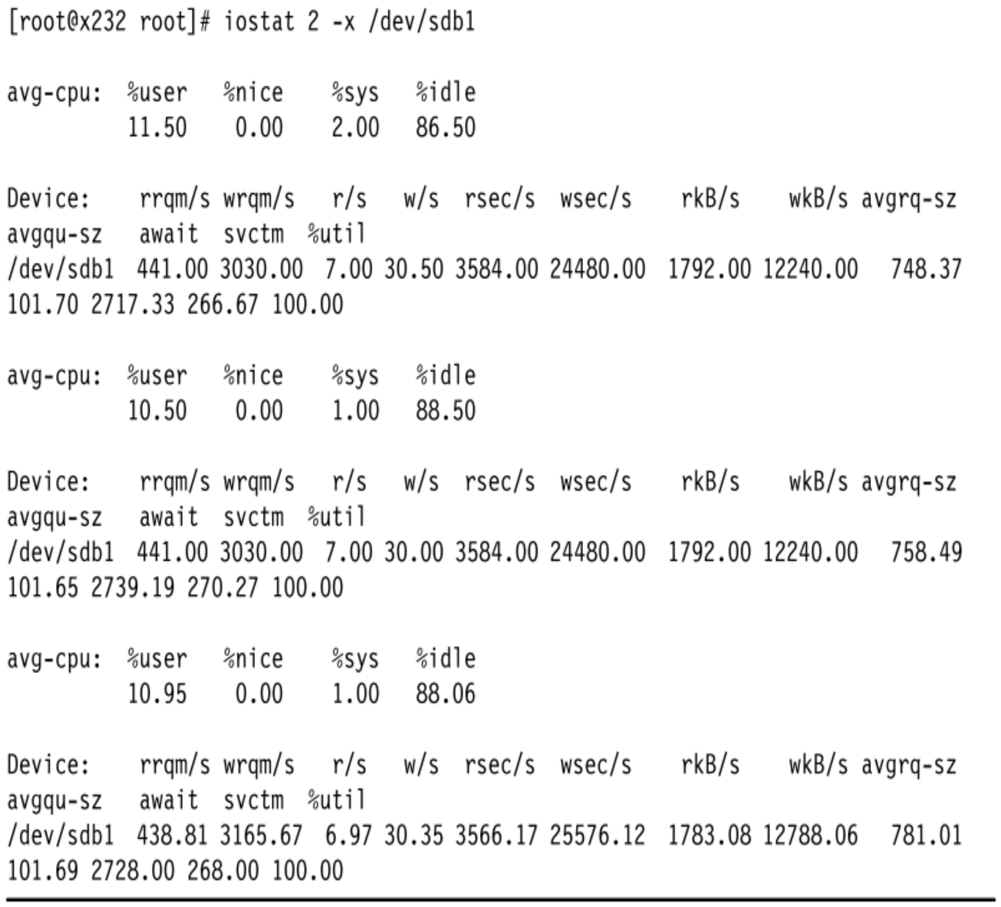


```
[root@x232 root]# vmstat 2
r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 2  1      0   9004  47196 1141672    0    0     0   950  149    74 87 13  0  0
 0  2      0   9672  47224 1140924    0    0    12 42392  189    65 88 10  0  1
 0  2      0   9276  47224 1141308    0    0   448     0  144    28  0  0  0 100
 0  2      0   9160  47224 1141424    0    0   448  1764  149    66  0  1  0 99
 0  2      0   9272  47224 1141280    0    0   448    60  155    46  0  1  0 99
 0  2      0   9180  47228 1141360    0    0  6208 10730  425   413  0  3  0 97
 1  0      0   9200  47228 1141340    0    0 11200     6  631   737  0  6  0 94
 1  0      0   9756  47228 1140784    0    0 12224  3632  684   763  0 11  0 89
 0  2      0   9448  47228 1141092    0    0  5824 25328  403   373  0  3  0 97
 0  2      0   9740  47228 1140832    0    0   640     0  159    31  0  0  0 100
```

Figure 9: vmstat output

Performance problems can be encountered when too many files are opened, read and written to, then closed repeatedly. This could become apparent as seek times (the time it takes to move to the exact track where the data is stored) start to increase. Using the iostat tool, the I/O device loading can be monitored in real time. Different options enable deeper drill down to gather the necessary data.

Below illustration shows a potential I/O bottleneck on the device /dev/sdb1. This output shows average wait times (await) of about 2.7 seconds and service times (svctm) of 270 ms.



```
[root@x232 root]# iostat 2 -x /dev/sdb1

avg-cpu:  %user   %nice    %sys   %idle
          11.50    0.00    2.00   86.50

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/sdb1  441.00 3030.00  7.00 30.50 3584.00 24480.00  1792.00 12240.00   748.37
101.70 2717.33 266.67 100.00

avg-cpu:  %user   %nice    %sys   %idle
          10.50    0.00    1.00   88.50

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/sdb1  441.00 3030.00  7.00 30.00 3584.00 24480.00  1792.00 12240.00   758.49
101.65 2739.19 270.27 100.00

avg-cpu:  %user   %nice    %sys   %idle
          10.95    0.00    1.00   88.06

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/sdb1  438.81 3165.67  6.97 30.35 3566.17 25576.12  1783.08 12788.06   781.01
101.69 2728.00 268.00 100.00
```

Figure 10: Potential I/O bottleneck

If the disk subsystem is identified to be system bottleneck some of the remedies that can be undertaken include:-

i. If the workload is of a sequential nature and it is stressing the controller bandwidth, the solution is to add a faster disk controller. However, if the workload is more random in nature, then the bottleneck is likely to involve the disk drives, and adding more drives will improve performance.
ii. Add more disk drives in a RAID environment. This spreads the data across multiple physical disks and improves performance for both reads and writes. This will increase the number of I/Os per second. Also, use hardware RAID instead of the software implementation provided by operating systems. If hardware RAID is being used, the RAID level is hidden from the OS.

iii. Consider using Linux logical volumes with striping instead of large single disks or logical volumes without striping.

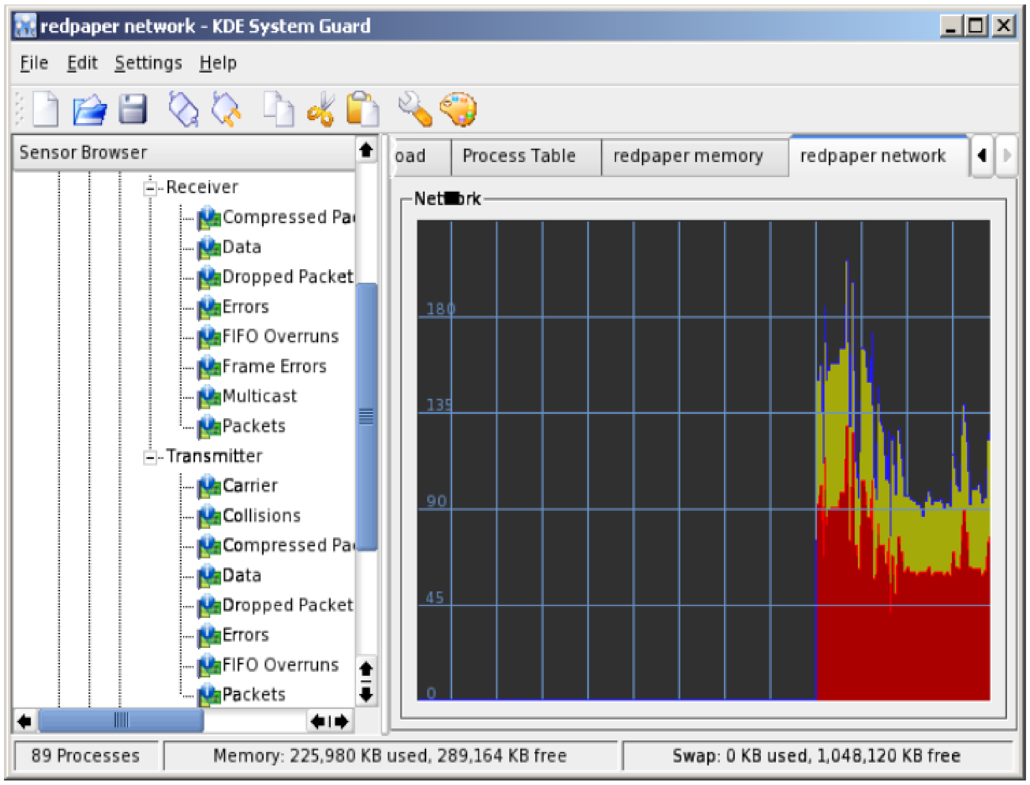
iv. Offload processing to another system in the network (users, applications, or services).

d. Network bottlenecks

One of the most common performance issues of the network is packet size. Network packets carry an application's messages via the database middleware to the database and vice versa. The size of the packets makes a difference in the performance of database application. Fewer packets sent between the application and the database equates to better performance -- fewer packets mean fewer trips to and from the database.

To analyze network performance anomalies in order to detect network bottlenecks, most operating systems include traffic analyzers.

A good traffic analyzer is KDE System Guard because of its graphical interface and ease of use.



89 Processes | Memory: 225,980 KB used, 289,164 KB free | Swap: 0 KB used, 1,048,120 KB free

Figure 11:KDE System Guard network monitoring

It is important to remember that there are many possible reasons for network performance problems and that sometimes problems occur simultaneously, making it even more difficult to pinpoint the origin.

The table below can help determine problems with the network.

Network Indicator	**Analysis**
Packets received Packets sent	Shows the number of packets that are coming in and going out of the specified network interface. Check both internal and external interfaces.
Collision packets	Collisions occur when there are many systems on the same domain. The use of a hub may be the cause of many collisions.
Dropped packets	Packets may be dropped for a variety of reasons, but the result can affect performance. For example, if the server network interface is configure to run at 100 Mbps full duplex, but the network switch is configured to run at 10 Mbps, the router may have an ACL filter that drops these packets. For example: *iptables -t filter –A FORWARD –p all –I eth2 -0 eth1 –s 172.18.0.0/24 –j DROP*
Errors	Errors occur if the communication lines are of poor quality. In these situations, corrupted packets must be resent, thereby decreasing network throughput.
Faulty adapters	Network slowdowns often result from faulty network adapters. When this kind of hardware fails, it might begin to broadcast junk packets on the network,

Table 4: Indicators for network analysis

If a network bottleneck is suspected the following steps can be used to solve some of the problems:-

i. Ensure that the network card configuration matches router and switch configurations (for example, frame size).
ii. Modify how your subnets are organized.
iii. Use faster network cards.
iv. Tune the appropriate IPV4 TCP kernel parameters.
v. If possible, change network cards and recheck performance.
vi. Add network cards and bind them together to form an adapter team, if possible.

ii. Application bottlenecks

As illustrated in the introductory section of identifying bottlenecks, a high percentage of bottlenecks are found in the application. Business logic of an application resides on the application server and web servers. Application server hardware, software and application design can affect the performance to great extent. Poor application server performance can be a critical source of performance bottlenecks.

A lot of writing has been done in this area. Some of the papers that can be used to help a performance engineer identify bottlenecks in the application and web tiers are.

i. Web applications Performance Symptoms and Bottlenecks Identification, by Thomas of agile toad
ii. Rapid Bottleneck Identification A Better Way to do Load Testing, an Oracle White Paper
iii. Application performance testing series by Scott Barber of AuthenTec

Most of the guidelines given below are derived from the above papers.

Some of the most common application bottlenecks include:-

i. Memory leaks
ii. Useless or inefficient garbage collection
iii. DB connections poor configuration
iv. Useless or inefficient code transactions
v. Sub-optimal session model
vi. Application server poor configuration
vii. Useless or inefficient object access model
viii. Useless or inefficient security model
ix. Less utilization of OS resources

Some of the bottlenecks to look out for in web servers include

i. Broken links
ii. Inadequate transaction design
iii. Very tight security
iv. Inadequate hardware capacity
v. High SSL transactions
vi. Server poorly configured
vii. Servers with ineffective load balancing
viii. Less utilization of OS resources
ix. Insufficient throughput

To identify bottlenecks in the application and web tier, the following guidelines can be followed:-

i. Ensure that the operating system, hardware and the network are configured optimally for the application they are to run and they contain no major bottleneck.
ii. Profile the application using appropriate tools, some of the profiling tools are discussed in the gathering statistics section. For each application a profiling tool can be found. Profiling an application is very useful in identifying bottlenecks in application, especially bottlenecks that are inherent in the application, such as memory leaks, improper function calls, and inefficient garbage collection among others.
iii. Perform thorough tests on the application. Some of the tests include :-
iv. Throughput testing
v. Concurrency testing
vi. Stress testing
vii. The (Oracle-RBI, 2010) methodology recommends starting with the simplest test cases first and then moving on to those with increased complexity. If the simplest test case works and the next level of

complexity fail, the bottleneck lies in the newly added complexity. By uncovering bottlenecks using a tiered approach, one can quickly identify issues as well as isolate issues in components of which you have limited knowledge.

viii. Analyze the application SQL statements using database SQL analyzers for poorly performing SQL queries. All modern database management systems have tools that can be used to identify such SQLs. Some of the tools are discussed in the database gathering statistics section.

In general, a database application should be written to:

a. Reduce network traffic
b. Limit disk input/output
c. Optimize application-to-driver interaction
d. Simplify queries

iii. Database Bottlenecks

Database performance is most critical for application performance as this is the main culprit in performance bottlenecks. Database software, hardware and design can really impact the whole system performance. Some of the database bottlenecks according to (Thomas, 2012) are:-

i. Inefficient or ineffective SQL statement
ii. Small or insufficient query plan cache
iii. Inefficient/ineffective SQA query model
iv. Inefficient/ineffective DB configurations
v. Small/insufficient data cache
vi. Excess DB connections
vii. Excess rows at a time processing
viii. Missing/ineffective indexing
ix. Inefficient/ineffective concurrency model
x. Outdated statistics
xi. Deadlocks

By use of operating system, DBMS performance tuning tools and or SQL statements performance problems in the database can be identified. Most of these database tools are discussed in the database statistics collection section. Thresholds that have been exceeded should be carefully investigated to determine where the performance problem lies. For example examining the top 5 wait events in an oracle database reveals the following

Top 5 Timed Events	**% Total**		
Event	**Waits**	**Time(s)**	**Elapsed Time**
CPU time	4,851	4,042	55.76
db file sequential read	1,968	1,997	27.55
log file sync	299,097	369	5.08
db file scattered read	53,031	330	4.55
log file parallel write	302,680	190	2.62

Table 5: Top five databases wait

The database is CPU bound and it is spending significant time waiting for CPU, Assuming that the database cache and the SQLs are already optimized; more CPU's or faster CPUs will improve the performance of this database. (Burleson, 2010)

In identifying and resolving database bottlenecks the performance engineer should be on the lookout for the following common mistakes usually found in databases. (ISAM, 2011)

i. Bad Connection Management

   The application connects and disconnects for each database interaction. This problem is common with stateless middleware in application servers. It has over two orders of magnitude impact on performance, and it is totally unscalable.

ii. Bad Use of Cursors and the Shared Pool

   Not using cursors results in repeated parses. If bind variables are not used, then there is hard parsing of all SQL statements. This has an order of magnitude impact in performance, and it is totally unscalable Use cursors with bind variables that open the cursor and execute it many times. Be suspicious of applications generating dynamic SQL.

iii. Getting Database I/O Wrong

   Many sites lay out their databases poorly over the available disks. Other sites specify the number of disks incorrectly, because they configure disks by disk space and not I/O bandwidth.

iv. Long Full Table Scans

   Long full table scans for high-volume or interactive online operations could indicate poor transaction design, missing indexes, or poor SQL optimization. Long table scans, by nature, are I/O intensive and unscalable.

v. In Disk Sorting

   In disk sorts for online operations could indicate poor transaction design, missing indexes, or poor SQL optimization. Disk sorts, by nature, are I/O-intensive and unscalable.

vi. Schema Errors and Optimizer Problems

   In many cases, an application uses too many resources because the schema owning the tables has not been successfully migrated from the development environment or from an older implementation.

Examples of this are missing indexes or incorrect statistics. These errors can lead to sub-optimal execution plans and poor interactive user performance. When migrating applications of known performance, export the schema statistics to maintain plan .

Likewise, optimizer parameters set in the initialization parameter file can override proven optimal execution plans. For these reasons, schemas, schema statistics, and optimizer settings should be managed together as a group to ensure consistency of performance.

(Burleson, 2010) gives the following silver bullet tips when identifying and tuning database bottlenecks

i. Fix the symptom first – the root cause can always be addressed later
ii. Time is critical – quick fix, instance wide adjustments are often the best option
iii. Be creative – traditional time consuming methods do not apply here.
iv. Once done root cause needs to be found and sorted out for the long term correction of the root cause of the problem.

Step 7: Ranking in terms of cost and impact

Once bottlenecks have been identified in any of the tiers, a choice has to be made on which bottleneck is to be resolved first. Bottlenecks that have huge improvement in performance and have minimal costs both monetary and time wise should be considered.

As hardware becomes cheaper and faster, more hardware can be used to sort out sub-optimal databases; it is often a safe, cost effective and timely solution to an acute database performance issue. The examples below by (Burleson, 2010) illustrate such scenarios

i. Moving to faster 64 bit processor server as opposed to expensive SQL statement tuning for a CPU bound database.
ii. Move to faster solid state disks for a heavily I/O bound application due to poorly application code which costs less and is less risky as opposed to incurring huge costs in rewriting the application.

(Burleson, 2010) goes ahead to give the following real world scenario:

> "Just last week I had a client who was having a huge CPU bottleneck, and the root cause was excessive parsing and really sub-optimal SQL execution plans. They chose to spend $50k for faster processors (15 minutes to fix) rather than spend $100k to tune 2,000 SQL statements (6 weeks to fix)."

To rank bottlenecks in terms of their cost and impact the following formulae is suggested:-

$$\boldsymbol{Rank} = \frac{\boldsymbol{Impact}}{\boldsymbol{Y} * \boldsymbol{Cost} * \boldsymbol{Time}}$$

Where

Rank: Is the Numerical value assigned to each bottleneck.

Impact: Value that represents the overall improvement of application performance on resolution of the impact. A higher value signifies a huge improvement in performance. Rank is directly proportionate to impact.

Cost: Value that represents the monetary implication of resolving the bottleneck. The higher the value the more expensive it is to fix the bottleneck. Rank is inversely proportionate to cost.

Time: Value that represents how long it takes to resolve the bottleneck. The higher the value the longer it takes to fix the bottleneck. Rank is inversely proportionate to Time.

Y: Coefficient that depicts the importance of cost and time. The higher the value the more important the costs and time aspect. Rank is inversely proportionate to Y.

Derivation of ranking formulae

In deriving formulae it is imperative to show how the different formulae sub components relate to the whole. A formulae can be proven/validated either mathematically or using experimentation. Proving the formulae mathematically is suggested as future works , an experimental approach is used to validate the formulae by using it to rank identified bottlenecks in a poorly performing payroll application. Future works can also be carried out to validate the formulae using other poorly performing systems. An attempt to explain how the formula was arrived at is made below.

A major challenge faced by tuning experts is the myriad of options that are available to them with each having different impact on overall system performance (Burleson, 2010). The formulae aims to provide the tuning expert with a simple tool which he can use to simplify the decision making process of deciding the order in which to resolve identified bottlenecks.

Rank is the output of the formulae which is a numeric valued arrived at after applying the formulae to a bottleneck.

Impact is a value that represents the overall improvement of application performance on resolution of the impact. For example in a poorly performing web application, one of the bottlenecks identified can be lack of sufficient memory in the webserver causing a high rate of swapping. Resolving the bottleneck by adding more memory to the webserver results to 40% overall improvement in response time for the web application. Our Impact is thus 40% for this bottleneck. A higher Impact value signifies a huge improvement in performance. A high impact means the bottleneck should be among the first to be resolved, it logically follows that the higher the impact of resolving a bottleneck the higher the rank of the bottleneck, thus rank is directly proportionate to impact $\boldsymbol{Rank\ \alpha\, Impact}$

Cost is a value that represents the monetary implication of resolving the bottleneck. Proceeding with our hypothetical example, resolving the memory bottleneck is done by buying additional memory. Let us assume the memory will cost USD 500. The cost of the bottleneck is thus USD 500. A scale can be adopted depending on the figures involved, for example the following scale can be used

USD 100 – 300	= 1
USD 300 – 500	= 2
USD 500 – 700	= 3
USD 700 – 900	= 4
USD 900 – 1100	= 5

Thus the cost of resolving the bottleneck using a scale would be 2. The higher the cost the more expensive it is to resolve the bottleneck and thus the order of resolving the bottleneck should be lowered consequently lowering the rank. Therefore, Rank is inversely proportional to cost. $\boldsymbol{Rank}\ \alpha\ \frac{\mathbf{1}}{\boldsymbol{Cost}}$

Time: Value that represents how long it takes to resolve the bottleneck. Proceeding with our example, buying and fixing memory in the server can approximately take 3 hours. The below numeric scale dependent on the values involved can be adopted.

1 - 3 hrs.	= 1
3 - 5 hrs.	= 2
5 - 7 hrs.	= 3
7 - 9 hrs.	= 4
9 - 11 hrs.	=5

The time value for resolving the bottleneck will thus be 1. The higher the time value the longer it takes to fix the bottleneck and the bottleneck should thus be ordered lower in the priority of resolving the bottlenecks.. Rank is thus inversely proportionate to Time. $\boldsymbol{Rank}\ \alpha\ \frac{\mathbf{1}}{\boldsymbol{Time}}$

Y is a Coefficient that depicts the importance that an organization or the tuning expert gives to cost and time. In some organization cost might not be a big factor meaning the organization is willing to spend large amounts of money to resolve the bottleneck, in others budget and cost constraints can be a huge factor in decision making. The higher the coefficient value the more important the costs and time aspect are to the Organization. Rank is thus inversely proportionate to Y,$\boldsymbol{Rank}\ \alpha\ \frac{\mathbf{1}}{\boldsymbol{Y}}$

We thus have four subcomponents of the equation:

$$\boldsymbol{Rank\ \ \alpha\ Impact}$$

$$\boldsymbol{Rank}\ \alpha\ \frac{\mathbf{1}}{\boldsymbol{Cost}}$$

$$\boldsymbol{Rank}\ \alpha\ \frac{\mathbf{1}}{\boldsymbol{Time}}$$

$$\boldsymbol{Rank} \ \alpha \ \frac{\mathbf{1}}{\boldsymbol{Y}}$$

Combining the four subcomponents gives us the final equation

$$\boldsymbol{Rank} = \frac{\boldsymbol{Impact}}{\boldsymbol{Y} * \boldsymbol{Cost} * \boldsymbol{Time}}$$

Step 8: Iterative resolution of identified bottlenecks in the tiers based on the ranking

Once the bottlenecks have been ranked they are resolved iteratively until the performance goals have been realized.

4.2. APPLICATION OF METHODOLOGY

Application the integrated holistic Methodology on a poorly performing payroll application

Step 1: Getting feedback from the user.

The user was asked to describe performance problems that he is experiencing with the payroll and he gave the following response.

“The payroll application takes too long to process the 6000 university employee payroll, on average it takes 4 hours for a single run and since many runs are made before the month is closed, the process is a tedious and time consuming process and at a minimum takes 3 days to finalize the monthly payroll processing”

“When the payroll application was developed 10 years ago, it is used to run very fast, as the year progresses the payroll has become slower”

From the users response the following can be deduced:-

i. The user is unhappy with the response time
ii. The user is unhappy with the throughput of the application
iii. The payroll has become slower as more data is added.
iv. The payroll database has 10 years historical data.

Step 2: Collecting baseline Performance measurement of the application

For the purpose of tuning the exercise, it was decided to use a different database server in which only the payroll application would be running. The decision was arrived at because of the following reasons:-

i. The live database server was in use by other application and the environment was changing rapidly thus it was difficult to accurately measure improvement in performance after changes in configuration.
ii. Being a live server there were limitations on configuration changes that would be effected without adversely affecting other users of the system.
iii. It was important to carry out scalability tests and vary the amount of historical data in the payroll database while observing change in response times, this could not be achieved in the live server. To reduce the amount of time taken to running the payroll during the testing exercise, for the purpose of collecting the baseline data, a payroll data with three years historical data was to used. For the purpose of determining the response time a sample size of 200 payroll records runs was to be considered. By running SQL statements on the payroll application the following data was collected:-
iv. The response time is time 101 secs to process 200 records translating to 0.5 secs per payroll record.
v. The throughput of the application is 200 payroll records in 101 secs, translates to 2 payroll records per second.

Step 3: Setting performance goals

After discussion with the user, the following performance goal is set

"The payroll application must process 6000 payroll records in under two hours."

Step 4: Analyzing system infrastructure

The payroll application is a client server application and its architecture is as follows

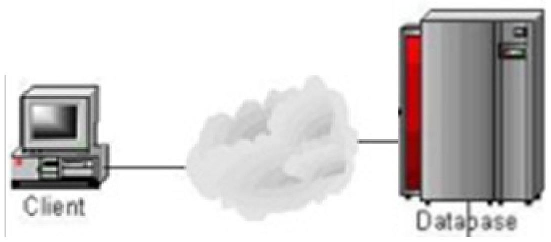


Figure 12: Payroll architecture

The payroll application runs in the users clients machine, all payroll processing calculations are performed in the client machine and results of the calculations saved in the database.

The payroll application is designed using Oracle forms which in modern times can be termed as a legacy application.

Clients Computer specification

Operating system: Windows 7 32 bit

Memory available: 2 GB

Processor: Intel duo core processor

Test Database server

Operating system: Oracle Enterprise Linux 5 32 bit

Memory available: 2 GB

Processor: 2 Intel i7 core processor (4 cores in total)

Database version: Oracle 11g with 40% memory assigned

Step 5: Collection of resource utilization and performance statics of all the tiers

i. Operating system and hardware statistics

With the payroll running the following statistics are collected

1. CPU utilization

Client Application Computer

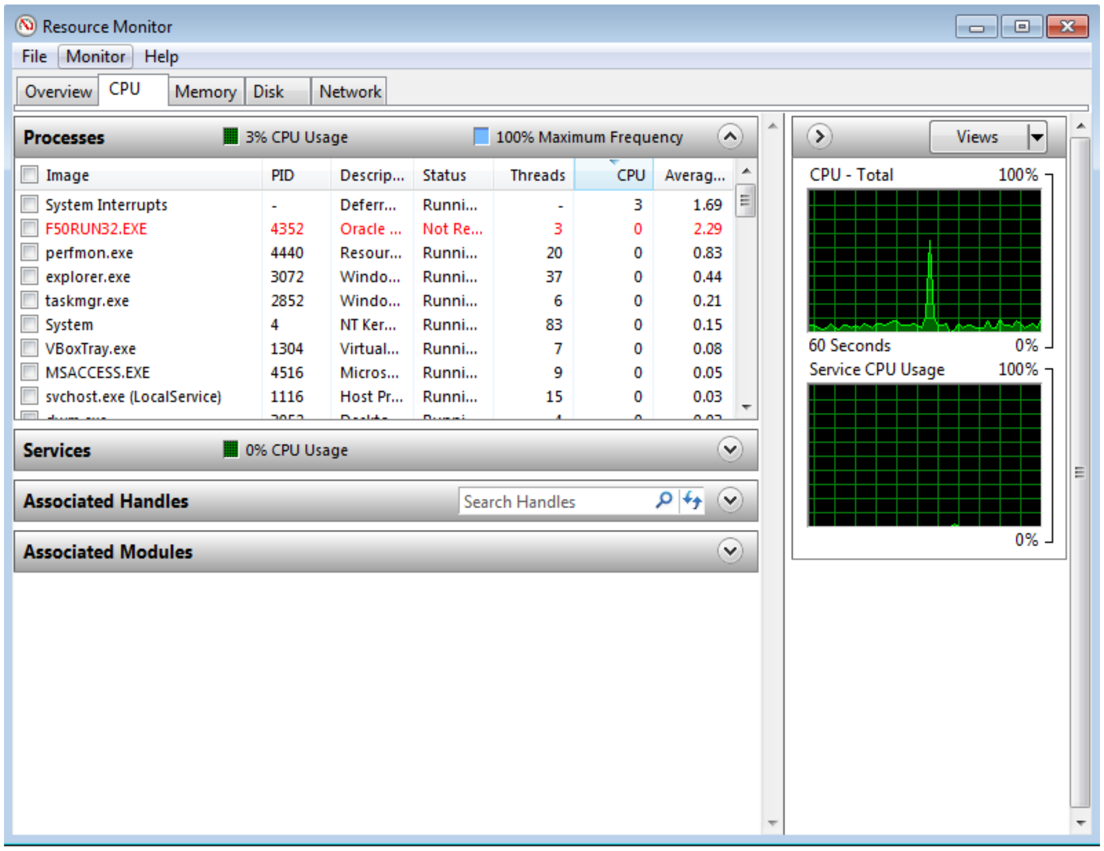
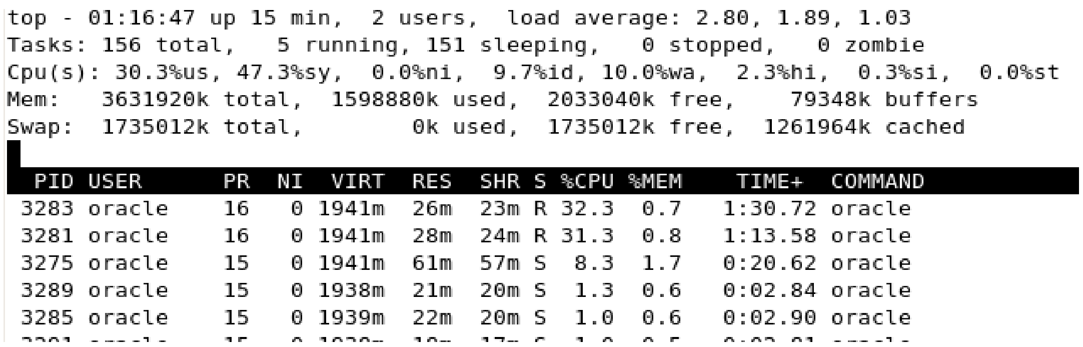


Image	PID	Descrip...	Status	Threads	CPU	Averag...
System Interrupts	-	Deferr...	Runni...	-	3	1.69
F50RUN32.EXE	4352	Oracle ...	Not Re...	3	0	2.29
perfmon.exe	4440	Resour...	Runni...	20	0	0.83
explorer.exe	3072	Windo...	Runni...	37	0	0.44
taskmgr.exe	2852	Windo...	Runni...	6	0	0.21
System	4	NT Ker...	Runni...	83	0	0.15
VBoxTray.exe	1304	Virtual...	Runni...	7	0	0.08
MSACCESS.EXE	4516	Micros...	Runni...	9	0	0.05
svchost.exe (LocalService)	1116	Host Pr...	Runni...	15	0	0.03

Figure 13: Client computer CPU utilization

CPU usage = 3 %

Database Server



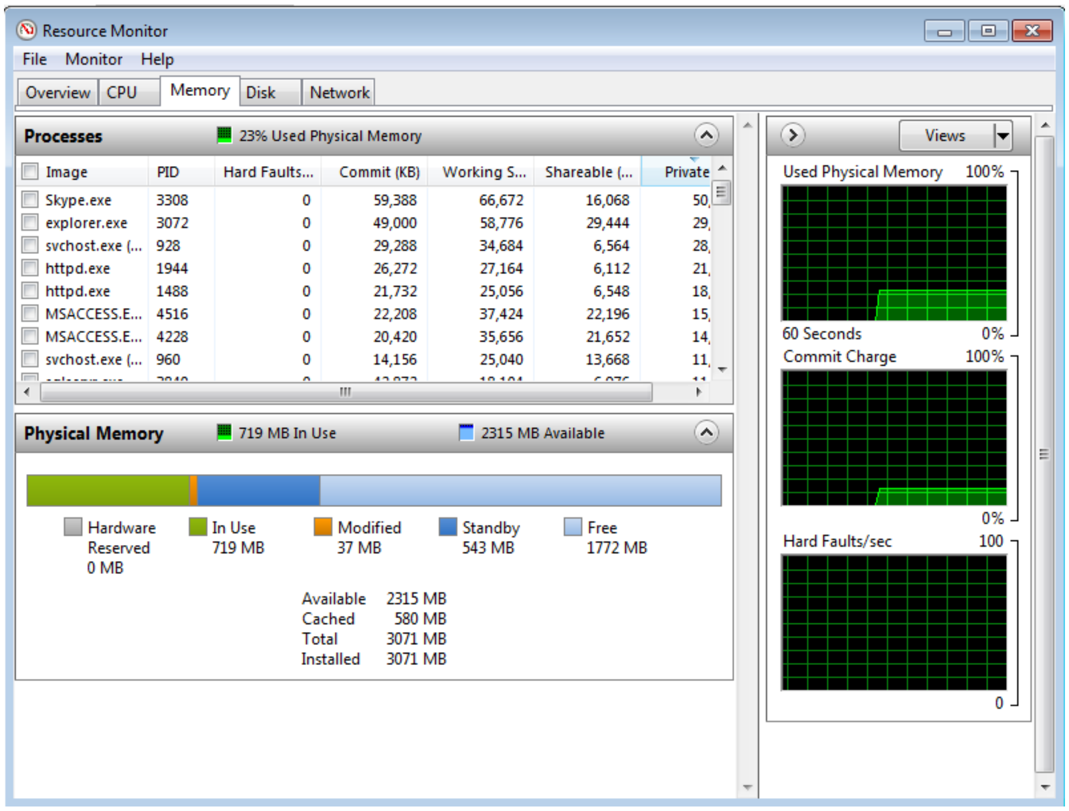
```
top - 01:16:47 up 15 min,  2 users,  load average: 2.80, 1.89, 1.03
Tasks: 156 total,   5 running, 151 sleeping,   0 stopped,   0 zombie
Cpu(s): 30.3%us, 47.3%sy,  0.0%ni,  9.7%id, 10.0%wa,  2.3%hi,  0.3%si,  0.0%st
Mem:   3631920k total,  1598880k used,  2033040k free,    79348k buffers
Swap:  1735012k total,        0k used,  1735012k free,  1261964k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3283 oracle    16   0 1941m  26m  23m R 32.3  0.7   1:30.72 oracle
 3281 oracle    16   0 1941m  28m  24m R 31.3  0.8   1:13.58 oracle
 3275 oracle    15   0 1941m  61m  57m S  8.3  1.7   0:20.62 oracle
 3289 oracle    15   0 1938m  21m  20m S  1.3  0.6   0:02.84 oracle
 3285 oracle    15   0 1939m  22m  20m S  1.0  0.6   0:02.90 oracle
```

Figure 14: Database server top CPU utilization

Average CPU usage 30%

2. Memory utilization and Virtual memory Statistics

Client Application Computer



Processes 23% Used Physical Memory

Image	PID	Hard Faults...	Commit (KB)	Working S...	Shareable (...	Private
Skype.exe	3308	0	59,388	66,672	16,068	50,
explorer.exe	3072	0	49,000	58,776	29,444	29,
svchost.exe (...	928	0	29,288	34,684	6,564	28,
httpd.exe	1944	0	26,272	27,164	6,112	21,
httpd.exe	1488	0	21,732	25,056	6,548	18,
MSACCESS.E...	4516	0	22,208	37,424	22,196	15,
MSACCESS.E...	4228	0	20,420	35,656	21,652	14,
svchost.exe (...	960	0	14,156	25,040	13,668	11,

Physical Memory 719 MB In Use 2315 MB Available

Hardware Reserved	In Use	Modified	Standby	Free
0 MB	719 MB	37 MB	543 MB	1772 MB

Available	2315 MB
Cached	580 MB
Total	3071 MB
Installed	3071 MB

Figure 15: Client computer memory utilization

Memory utilization = Used Memory / Total memory * 100

1299/ 3071 * 100 = 42 %

Database Server

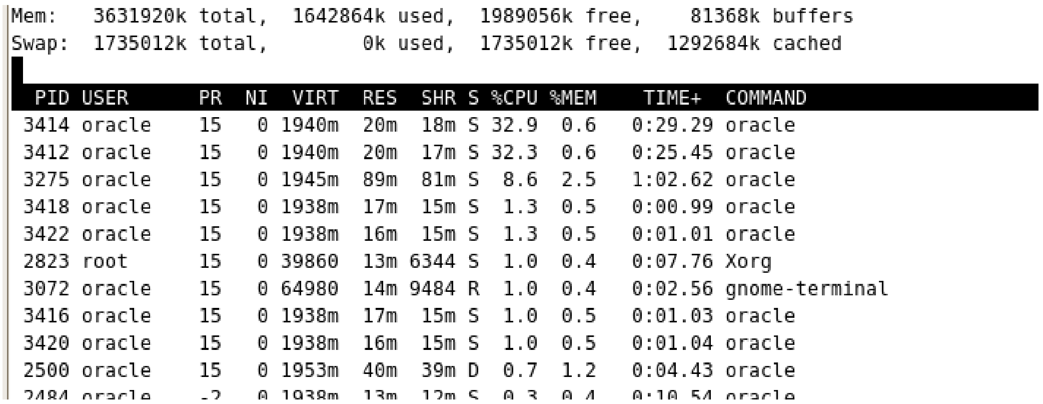

```
Mem:   3631920k total,  1642864k used,  1989056k free,    81368k buffers
Swap:  1735012k total,        0k used,  1735012k free,  1292684k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3414 oracle    15   0 1940m  20m  18m S 32.9  0.6   0:29.29 oracle
 3412 oracle    15   0 1940m  20m  17m S 32.3  0.6   0:25.45 oracle
 3275 oracle    15   0 1945m  89m  81m S  8.6  2.5   1:02.62 oracle
 3418 oracle    15   0 1938m  17m  15m S  1.3  0.5   0:00.99 oracle
 3422 oracle    15   0 1938m  16m  15m S  1.3  0.5   0:01.01 oracle
 2823 root      15   0 39860  13m 6344 S  1.0  0.4   0:07.76 Xorg
 3072 oracle    15   0 64980  14m 9484 R  1.0  0.4   0:02.56 gnome-terminal
 3416 oracle    15   0 1938m  17m  15m S  1.0  0.5   0:01.03 oracle
 3420 oracle    15   0 1938m  16m  15m S  1.0  0.5   0:01.04 oracle
 2500 oracle    15   0 1953m  40m  39m D  0.7  1.2   0:04.43 oracle
 2484 oracle    -2   0 1938m  13m  12m S  0.3  0.4   0:10.54 oracle
```

Figure 16: Database Server top memory utilization

Memory utilization = Used Memory / Total memory * 100

1642864 / 3631920 * 100 = 45 %

Swap utilization = 0 %

3. Disk utilization and I/O statistics

Client Application Computer Disk utilization

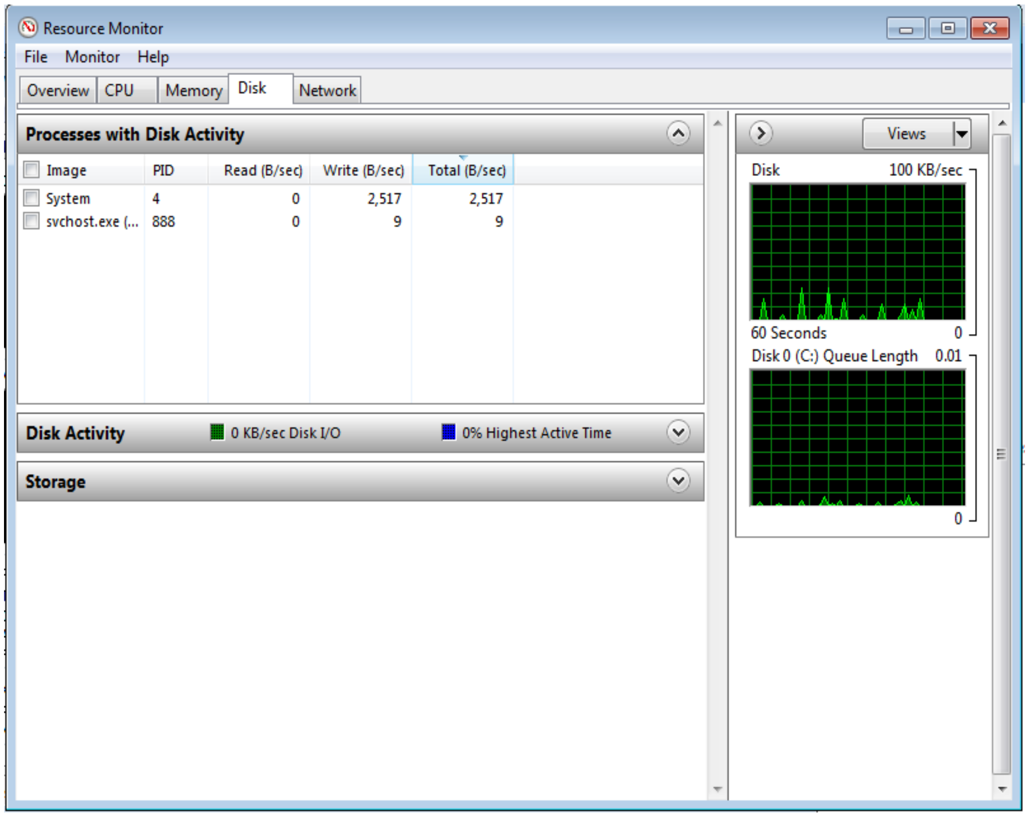

Resource Monitor
File Monitor Help
Overview CPU Memory Disk Network
Processes with Disk Activity
Views
Disk 100 KB/sec
60 Seconds 0
Disk 0 (C:) Queue Length 0.01
0
Storage

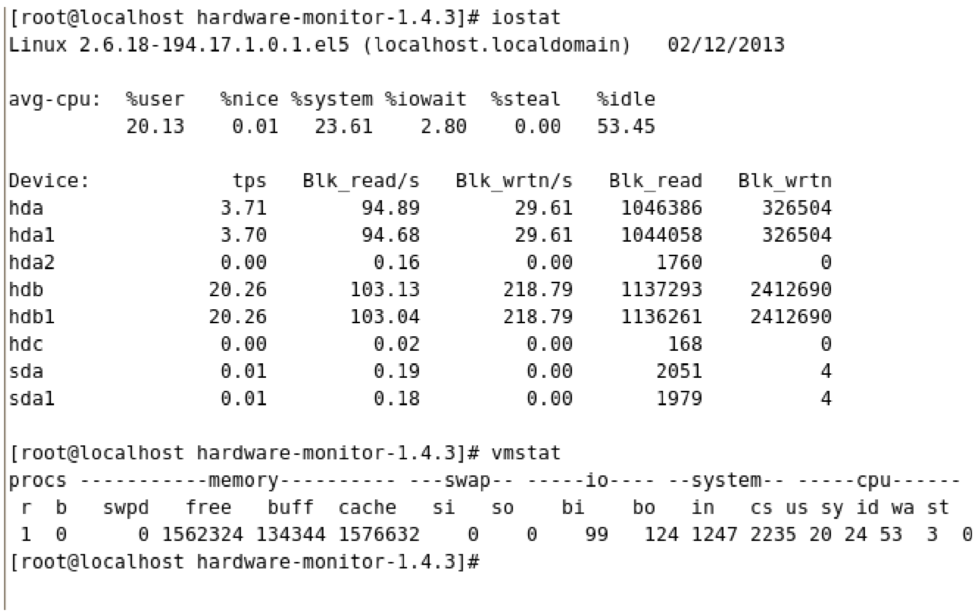
Image	PID	Read (B/sec)	Write (B/sec)	Total (B/sec)
System	4	0	2,517	2,517
svchost.exe (...	888	0	9	9

Disk Activity 🟩 0 KB/sec Disk I/O 🟦 0% Highest Active Time

Figure 17: Client computer Disk utilization

Client Application Computer - Disk Activity 0%

Database Server Disk (I/O) utilization



```
[root@localhost hardware-monitor-1.4.3]# iostat
Linux 2.6.18-194.17.1.0.1.el5 (localhost.localdomain)   02/12/2013

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          20.13    0.01   23.61    2.80    0.00   53.45

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hda               3.71        94.89        29.61    1046386     326504
hda1              3.70        94.68        29.61    1044058     326504
hda2              0.00         0.16         0.00       1760          0
hdb              20.26       103.13       218.79    1137293    2412690
hdb1             20.26       103.04       218.79    1136261    2412690
hdc               0.00         0.02         0.00        168          0
sda               0.01         0.19         0.00       2051          4
sda1              0.01         0.18         0.00       1979          4

[root@localhost hardware-monitor-1.4.3]# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 1562324 134344 1576632    0    0    99   124 1247 2235 20 24 53  3  0
[root@localhost hardware-monitor-1.4.3]#
```

Figure 18: Database Server I/O utlization

Database server I/O activity

Blocks read per sec – 94.89

Blocks written per sec – 29.61

4. Network Utilization

Client Application Computer

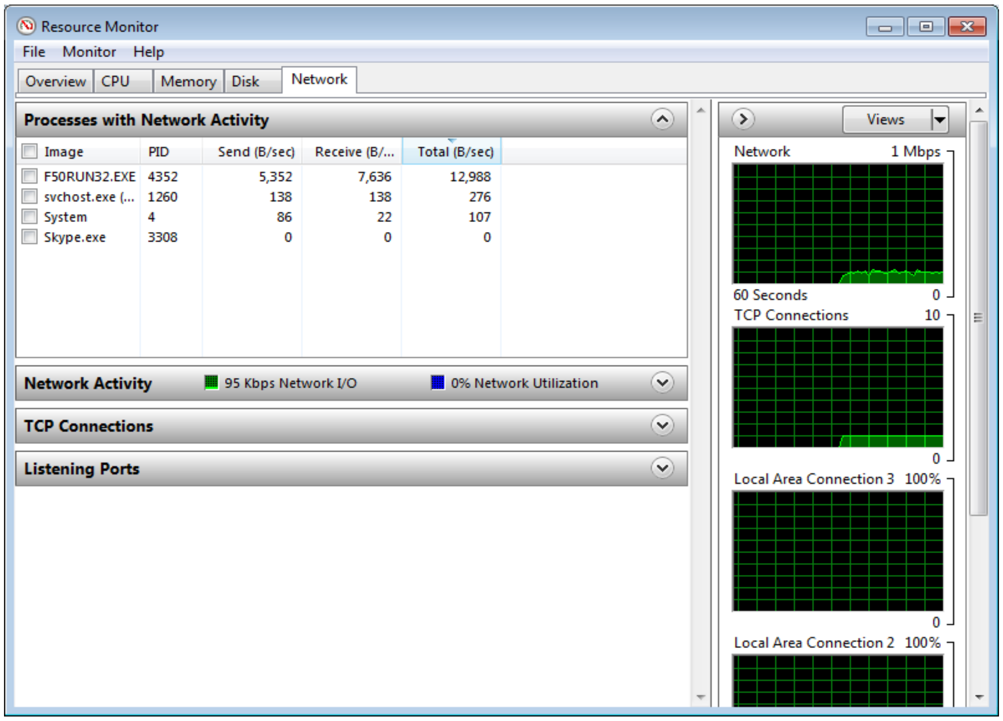
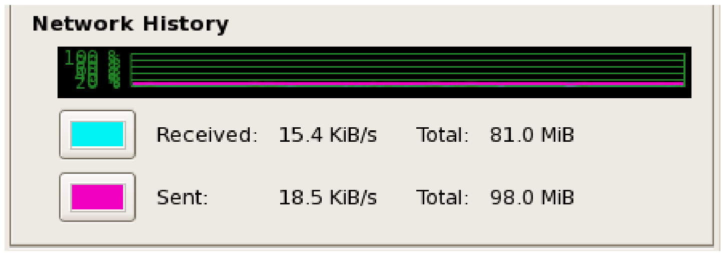

Resource Monitor
File Monitor Help
Overview CPU Memory Disk Network
Processes with Network Activity
Network Activity 🟩 95 Kbps Network I/O 🟦 0% Network Utilization
TCP Connections
Listening Ports
Views
Network 1 Mbps
60 Seconds 0
TCP Connections 10
0
Local Area Connection 3 100%
0
Local Area Connection 2 100%

Image	PID	Send (B/sec)	Receive (B/...	Total (B/sec)
F50RUN32.EXE	4352	5,352	7,636	12,988
svchost.exe (...	1260	138	138	276
System	4	86	22	107
Skype.exe	3308	0	0	0

Figure 19: Client computer network utilization

Client Application Computer Network Utilization - 0%

Database Server Network Utilization



🟦	Received:	15.4 KiB/s	Total:	81.0 MiB
🟪	Sent:	18.5 KiB/s	Total:	98.0 MiB

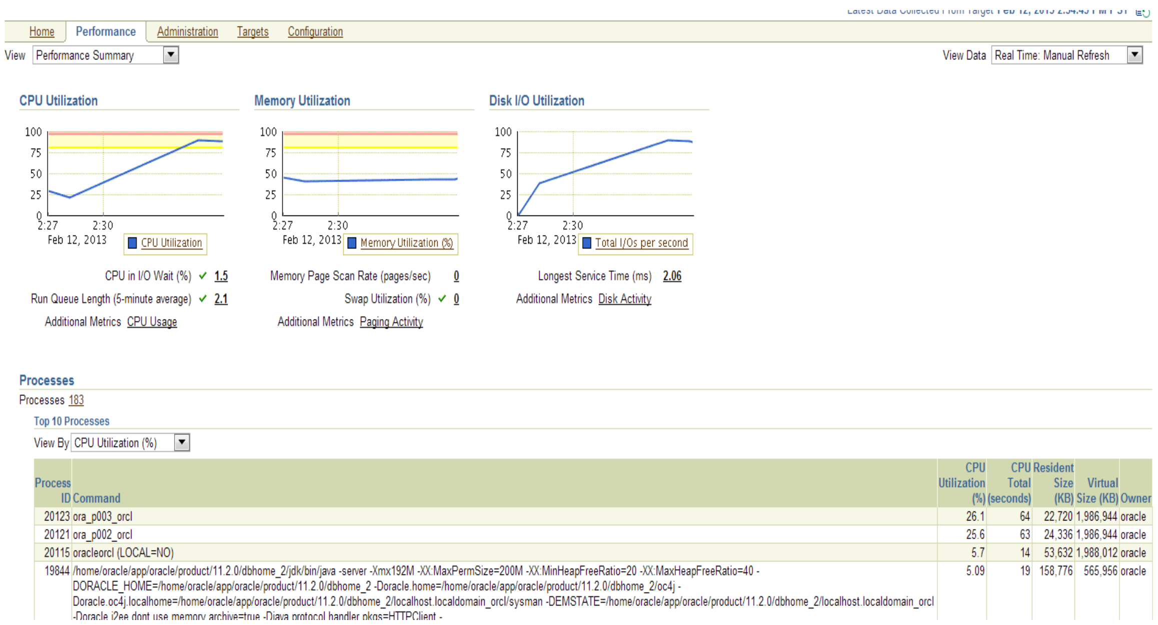
Figure 20: Database server network utilization

Database Server Network Utilization - < 20%

ii. Database and application performance statistics

The payroll application runs on an oracle database, the Oracle Enterprise Manager Tools were used to collect the performance statistics.

Overview of Performance



View Performance Summary

View Data Real Time: Manual Refresh

CPU in I/O Wait (%) ✓ 1.5
Run Queue Length (5-minute average) ✓ 2.1
Additional Metrics CPU Usage

Memory Page Scan Rate (pages/sec) 0
Swap Utilization (%) ✓ 0
Additional Metrics Paging Activity

Longest Service Time (ms) 2.06
Additional Metrics Disk Activity

Processes

Processes 183

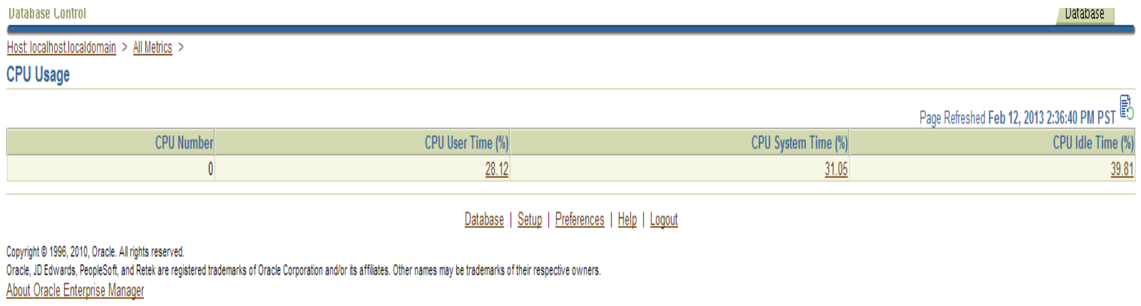
Top 10 Processes

View By CPU Utilization (%)

Process ID	Command	CPU Utilization (%)	CPU Total (seconds)	Resident Size (KB)	Virtual Size (KB)	Owner
20123	ora_p003_orcl	26.1	64	22,720	1,986,944	oracle
20121	ora_p002_orcl	25.6	63	24,336	1,986,944	oracle
20115	oracleorcl (LOCAL=NO)	5.7	14	53,632	1,988,012	oracle
19844	/home/oracle/app/oracle/product/11.2.0/dbhome_2/jdk/bin/java -server -Xmx192M -XX:MaxPermSize=200M -XX:MinHeapFreeRatio=20 -XX:MaxHeapFreeRatio=40 -DORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_2 -Doracle.home=/home/oracle/app/oracle/product/11.2.0/dbhome_2/oc4j -Doracle.oc4j.localhome=/home/oracle/app/oracle/product/11.2.0/dbhome_2/localhost.localdomain_orcl/sysman -DEMSTATE=/home/oracle/app/oracle/product/11.2.0/dbhome_2/localhost.localdomain_orcl -Doracle.j2ee.dont.use.memory.archive=true -Djava.protocol.handler.pkgs=HTTPClient -	5.09	19	158,776	565,956	oracle

Figure 21: Database performance overview

CPU Usage



Host: localhost.localdomain > All Metrics >

CPU Usage

Page Refreshed Feb 12, 2013 2:36:40 PM PST

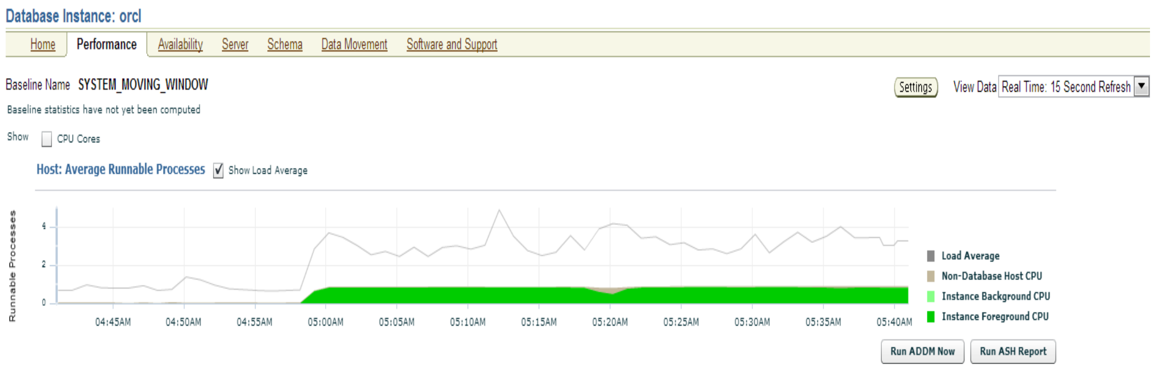
CPU Number	CPU User Time (%)	CPU System Time (%)	CPU Idle Time (%)
0	28.12	31.05	39.81

Database | Setup | Preferences | Help | Logout

Copyright © 1996, 2010, Oracle. All rights reserved.
Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
About Oracle Enterprise Manager

Figure 22: Database CPU usage

Runnable processes



Baseline Name SYSTEM_MOVING_WINDOW

Baseline statistics have not yet been computed

View Data Real Time: 15 Second Refresh

Figure 23: Database runnable processes

Average Active sessions

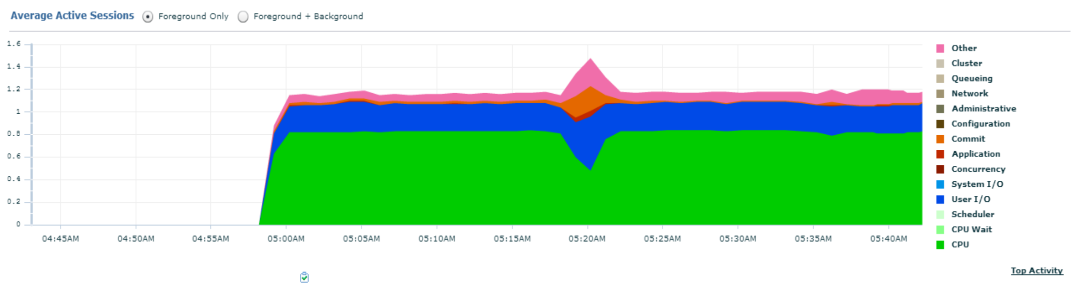


Figure 24: Database Average Active sessions

Throughput

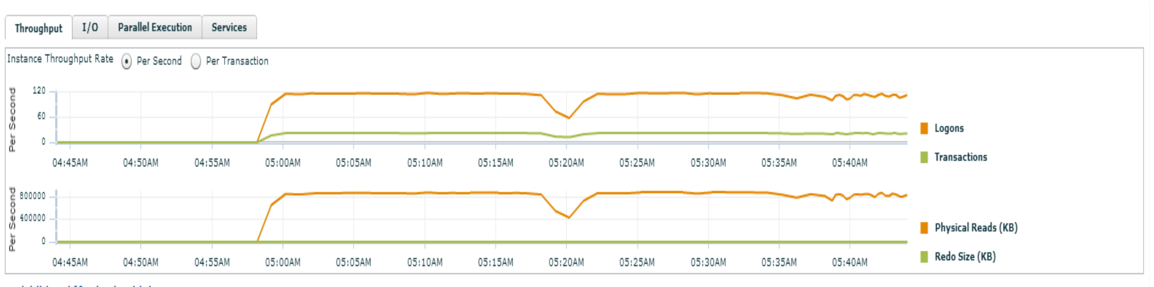


Figure 25: Database throughput

IO

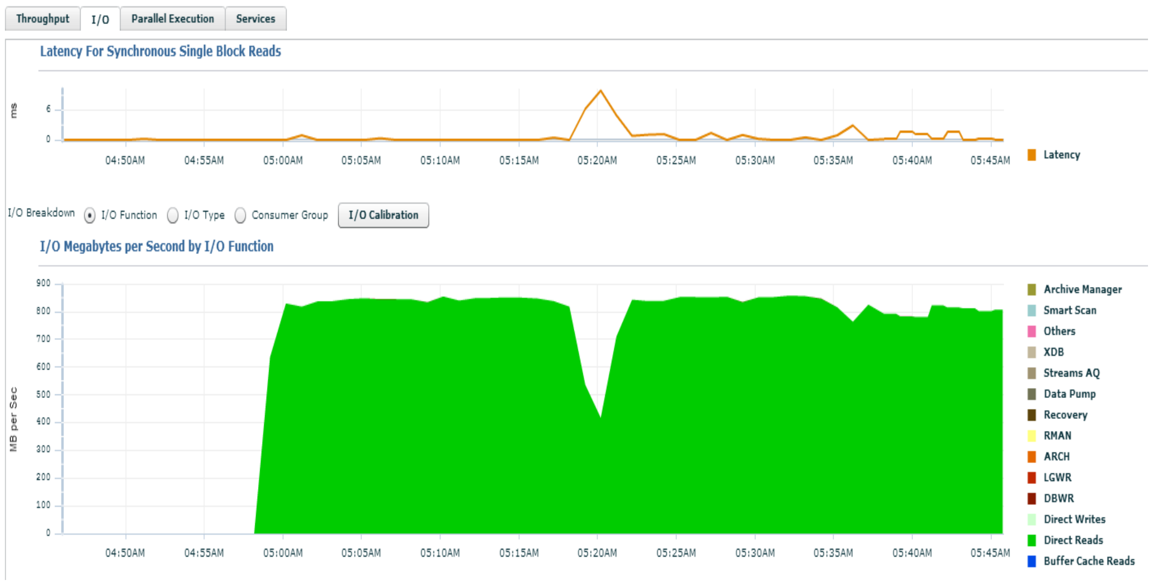


Figure 26: Database I/O throughput

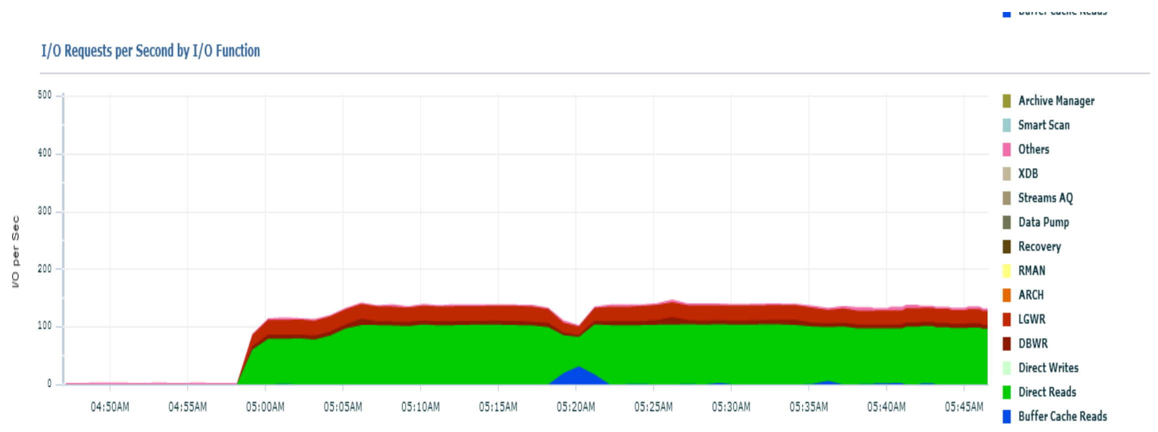


Figure 27: Database I/O throughput

Parallel Executions

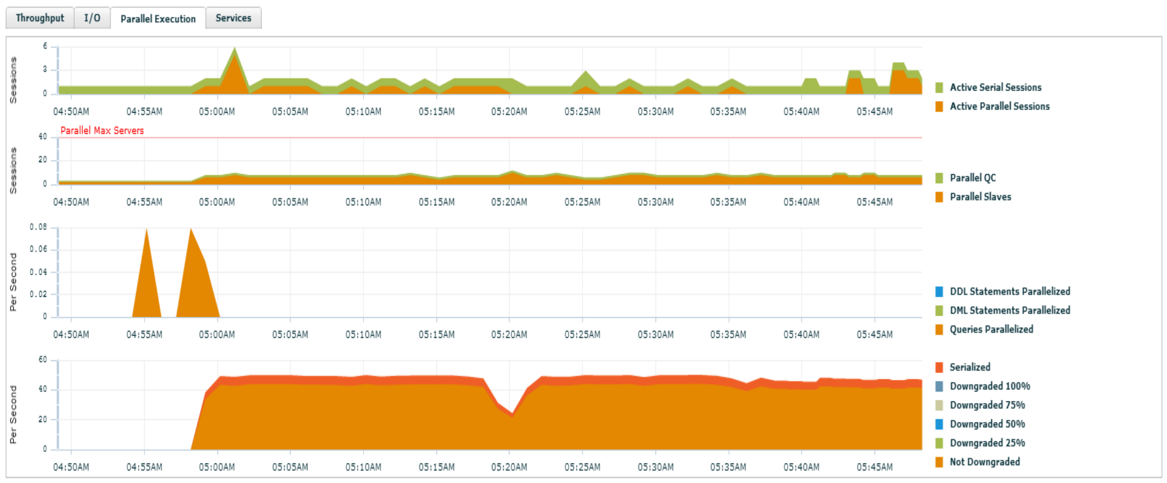
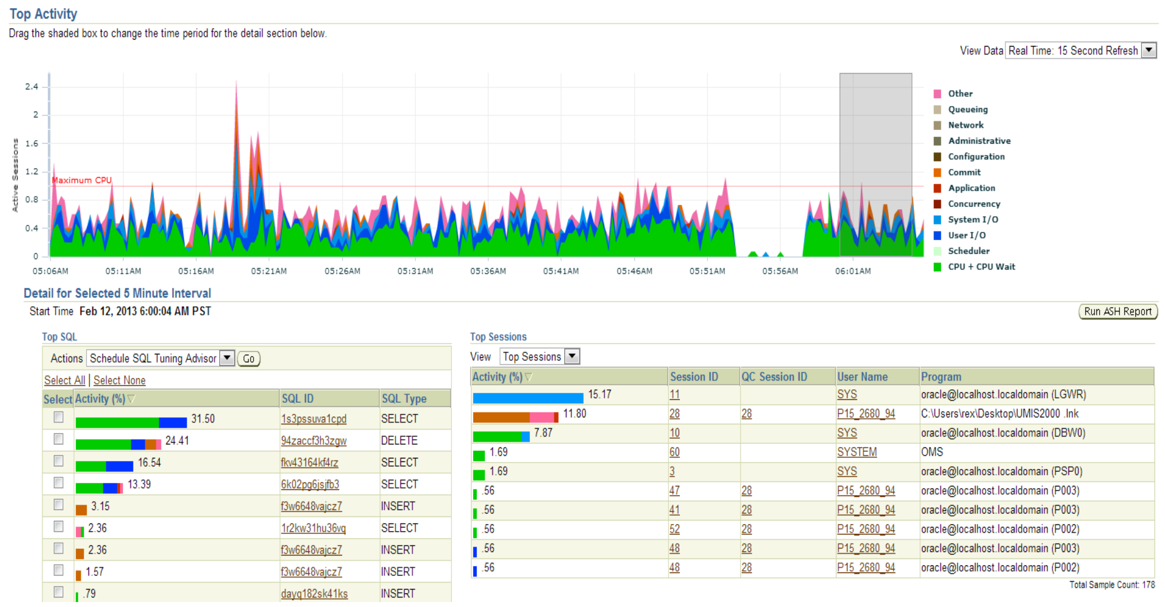


Figure 28: Database Parallel Executions

Top Activity



Top Activity

Drag the shaded box to change the time period for the detail section below.

View Data Real Time: 15 Second Refresh

Detail for Selected 5 Minute Interval

Start Time **Feb 12, 2013 6:00:04 AM PST**

Top SQL

Actions Schedule SQL Tuning Advisor

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
☐	31.50	1s3pssuva1cpd	SELECT
☐	24.41	94zaccf3h3zgw	DELETE
☐	16.54	fkv43164kf4rz	SELECT
☐	13.39	6k02pg6jsjfb3	SELECT
☐	3.15	f3w6648vajcz7	INSERT
☐	2.36	1r2kw31hu36vq	SELECT
☐	2.36	f3w6648vajcz7	INSERT
☐	1.57	f3w6648vajcz7	INSERT
☐	.79	dayq182sk41ks	INSERT

Top Sessions

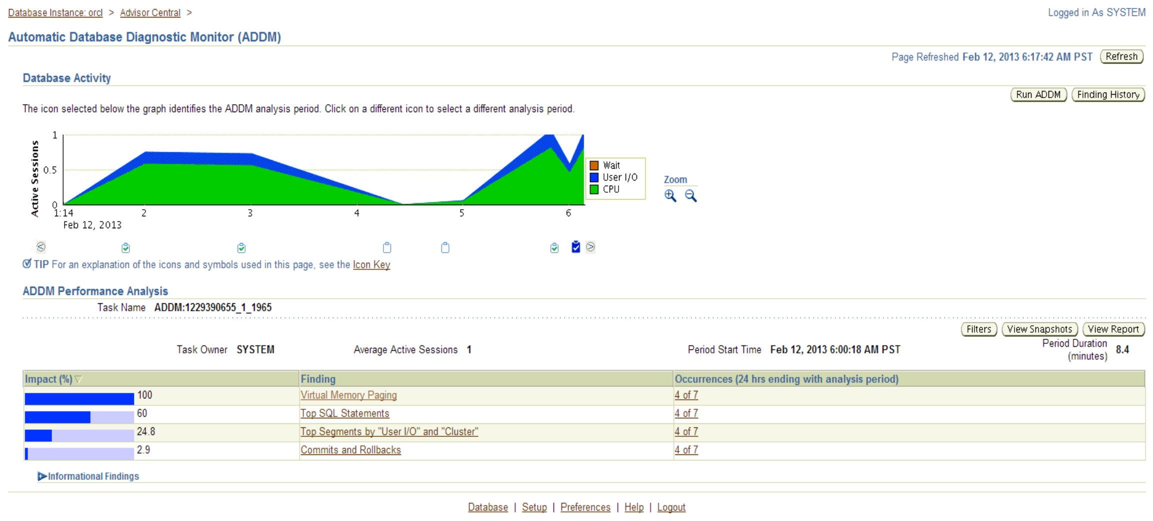
View Top Sessions

Activity (%)	Session ID	QC Session ID	User Name	Program
15.17	11		SYS	oracle@localhost.localdomain (LGWR)
11.80	28	28	P15_2680_94	C:\Users\rex\Desktop\UMIS2000 .lnk
7.87	10		SYS	oracle@localhost.localdomain (DBW0)
1.69	60		SYSTEM	OMS
1.69	3		SYS	oracle@localhost.localdomain (PSP0)
.56	47	28	P15_2680_94	oracle@localhost.localdomain (P003)
.56	41	28	P15_2680_94	oracle@localhost.localdomain (P003)
.56	52	28	P15_2680_94	oracle@localhost.localdomain (P002)
.56	48	28	P15_2680_94	oracle@localhost.localdomain (P003)
.56	48	28	P15_2680_94	oracle@localhost.localdomain (P002)

Total Sample Count: 178

Figure 29: Database Top Activity

Automated Database Diagnostic Monitoring tool (ADDM) Run



Database Instance: orcl > Advisor Central >

Logged in As SYSTEM

Automatic Database Diagnostic Monitor (ADDM)

Page Refreshed **Feb 12, 2013 6:17:42 AM PST**

Database Activity

The icon selected below the graph identifies the ADDM analysis period. Click on a different icon to select a different analysis period.

TIP For an explanation of the icons and symbols used in this page, see the Icon Key

ADDM Performance Analysis

Task Name **ADDM:1229390655_1_1965**

Task Owner **SYSTEM** Average Active Sessions **1** Period Start Time **Feb 12, 2013 6:00:18 AM PST** Period Duration (minutes) **8.4**

Impact (%)	Finding	Occurrences (24 hrs ending with analysis period)
100	Virtual Memory Paging	4 of 7
60	Top SQL Statements	4 of 7
24.8	Top Segments by "User I/O" and "Cluster"	4 of 7
2.9	Commits and Rollbacks	4 of 7

▶Informational Findings

Database | Setup | Preferences | Help | Logout

Figure 30: Database Automated Database Diagnostic Monitoring tool (ADDM) Run

For a listing of the complete ADDM output, refer to APPENDIX B

Step 6: Identification of bottlenecks

Once performance statistics from the various tiers was collected using the various tools a critical analysis of the data was conducted with the aim of identifying bottlenecks.

i. Operating system tier and hardware tier

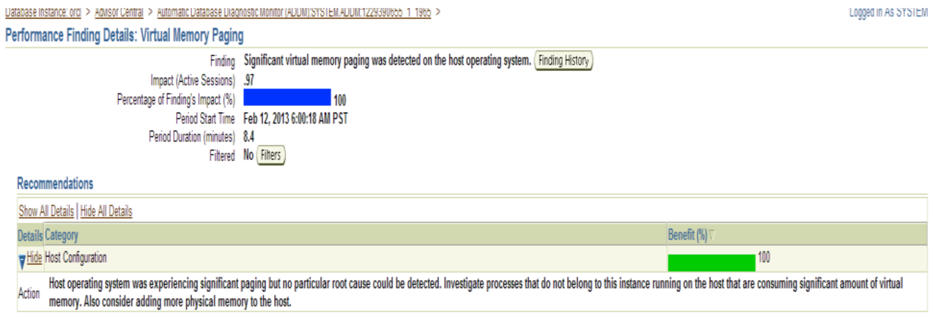
1. CPU bottlenecks

In both the client computer and the database server CPU was not a bottleneck. In the client computer CPU utilization was an average of 3% and on the database server 30%. The oracle database monitoring tool reported a slightly higher CPU usage of 60% but went ahead to explicitly state that CPU was not a bottleneck in its ADDM output which states as follows.
“CPU was not a bottleneck for the instance.”

2. Memory bottlenecks

   In both the client computer and the database server memory utilization never reached 100% and there was existence of free memory. Due to presence of free memory virtual memory in the clients’ windows application machine was not utilized and swapping in the Linux database server did not occur too.

   Oracle ADDM tool reported that excess virtual paging was occurring in the database host machine as illustrated below.



Database Instance: orcl > Advisor Central > Automatic Database Diagnostic Monitor (ADDM):SYSTEM.ADDM:1229390655_1_1965 >   Logged in As SYSTEM

Performance Finding Details: Virtual Memory Paging

Finding	Significant virtual memory paging was detected on the host operating system. Finding History
Impact (Active Sessions)	.97
Percentage of Finding's Impact (%)	100
Period Start Time	Feb 12, 2013 6:00:18 AM PST
Period Duration (minutes)	8.4
Filtered	No Filters

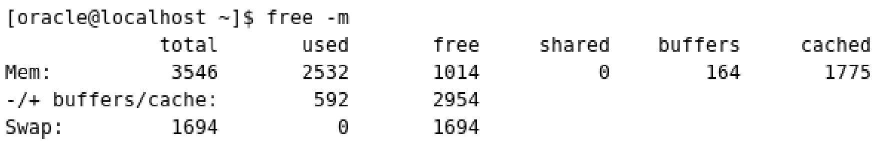
Recommendations

Show All Details | Hide All Details

Details	Category	Benefit (%)
Hide	Host Configuration	100
Action	Host operating system was experiencing significant paging but no particular root cause could be detected. Investigate processes that do not belong to this instance running on the host that are consuming significant amount of virtual memory. Also consider adding more physical memory to the host.	

Figure 31: Oracle virtual paging

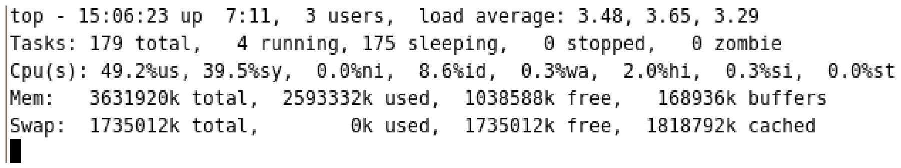
Further analysis indicated that this finding was not collaborated by other operating systems such as free and top which indicated existence of free memory and no swapping taking place.



```
[oracle@localhost ~]$ free -m
             total       used       free     shared    buffers     cached
Mem:          3546       2532       1014          0        164       1775
-/+ buffers/cache:        592       2954
Swap:         1694          0       1694
```

Figure 32: Free memory utilization

Output from Linux free tool



```
top - 15:06:23 up  7:11,  3 users,  load average: 3.48, 3.65, 3.29
Tasks: 179 total,   4 running, 175 sleeping,   0 stopped,   0 zombie
Cpu(s): 49.2%us, 39.5%sy,  0.0%ni,  8.6%id,  0.3%wa,  2.0%hi,  0.3%si,  0.0%st
Mem:   3631920k total,  2593332k used,  1038588k free,   168936k buffers
Swap:  1735012k total,        0k used,  1735012k free,  1818792k cached
```

Figure 33: Top memory utilization

Output from top Linux tool

Review of literature on the inconsistency revealed that this is due to a bug in the ADDM tool (Kirill, 2012)

Above inconsistency illustrates challenges experienced during performance tools and shy deeper analysis of performance issues should be conducted before conclusions are arrived at. Use of more than one tool is also recommended in order to validate the results obtained.

3. Disk bottlenecks

The client windows machine indicated barely noticeable disk activity of 0%.

The database server had noticeable disk activity of 94.89 Blocks read per sec and 29.61 Blocks written per sec.

Oracle enterprise manager indicated significant disk I/O.

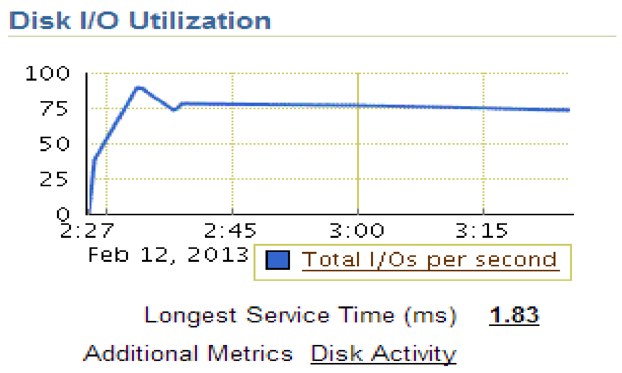


Figure 34: Oracle I/O utilization

Further analysis showed that specific SQL statements were causing the high disk activity.

Output from ADDM.

“ Finding 3: Top Segments by "User I/O" and "Cluster"

Impact is .24 active sessions, 24.77% of total activity.

Individual database segments responsible for significant "User I/O" and

"Cluster" waits were found.

Recommendation 1: Segment Tuning

Estimated benefit is .24 active sessions, 24.77% of total activity.

Action

Run "Segment Advisor" on TABLE "P15_2680_94.PROCESSED_DATA" with object

ID 101437.

Related Object

```
      Database object with ID 101437.
  Action
    Investigate application logic involving I/O on TABLE
    "P15_2680_94.PROCESSED_DATA" with object ID 101437.
    Related Object
      Database object with ID 101437.
  Action
    Look at the "Top SQL Statements" finding for SQL statements consuming
    significant I/O on this segment. For example, the DELETE statement with
    SQL_ID "94zaccf3h3zgw" is responsible for 100% of "User I/O" and
    "Cluster" waits for this segment.
  Rationale
    The I/O usage statistics for the object are: 97240 full object scans,
    52098200 physical reads, 3233 physical writes and 52098200 direct reads.

  Symptoms That Led to the Finding:
  ---------------------------------
    Wait class "User I/O" was consuming significant database time.
    Impact is .24 active sessions, 24.77% of total activity.”.
```

The SQL identified causing the high I/O needed to be looked or movement to faster solid states disks considered.

4. Network bottlenecks

For the client machine network utilization was barely noticeable being below 0 % and for the database server it was below 20 %.

Oracle ADDM tool reported explicitly that network was not a bottleneck as quoted below.

“Wait class "Network" was not consuming significant database time”.

Despite CPU and memory not being a bottleneck. Experiments were carried out to determine if the payroll response time would improve if more hardware resources were utilized. (Burleson, 2010) in his Oracle performance, hardware & RAM tuning optimization article indicates that use of faster processors almost always leads to improvement in performance.

Since resource utilization on the clients machine are negligible, it only the database location that was varied in the experiment.

The following three computers were used with specifications indicated.

Specifications	**HP Laptop 620**	**Dell Power edge 880 server**	**HP Elite Book**
Operating System	Oracle Enterprise Linux	Oracle Enterprise Linux	Oracle Enterprise Linux
Processor Type	Intel® Core™2 Duo Processor T6670	Intel® Xeon® Processor E7-4807	Intel® Core™ i7-3520M Processor
Processor clock speed	2.2 GHz	1.86 GHz	2.9 GHz
Max Turbo Frequency			3.6 GHz
Processor cache	2 MB	18 MB Intel® Smart Cache	4 MB Intel® Smart Cache
Processor cores	2	6	2
Processor Threads	0	12	4
Processor release date	Q3'09	Q2'11	Q2'12
Memory Available	8 GB	64GB	8GB
Oracle Version	Oracle 11g R2	Oracle 11g R2	Oracle 11g R2

Table 6: Server Specifications

A payroll database with two months historical data was chosen and 200 payroll records processed in each computer. The average responses time in each computer are illustrated below.

Computer	**Response Time (secs)**
HP Laptop 620	88
Dell Power edge 880 server	80
HP Elite Book	30

Table 7: Response time per server

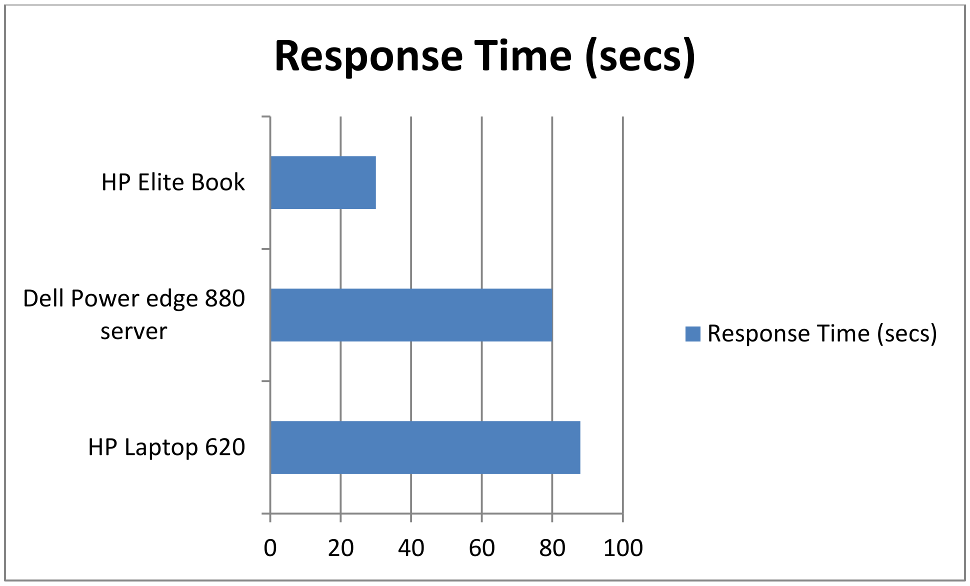


Figure 35: Payroll response time per server

As depicted in the chart the faster more recent processor of the Elite book computer performed considerably better than the other processors. There is a 65 % (88-30)/88 * 100 improvement in performance by using a faster processor.

Despite the Dell Power Edge server having 64 GB of RAM and 24 cores available, it did not lead to significant improvement in performance.

For detailed values of the experiment refer to Appendix C.

i. Application and Database Bottlenecks

An analysis of the application activity indicates existence of the following bottlenecks.

1. Problem with SQL statements

Using the ADDM tool it was found that a performance gain of 60 % could be obtained by tuning selected SQL statements.

Below are excerpts from ADDM analysis

" Finding 2: Top SQL Statements

Impact is .58 active sessions, 60% of total activity.

SQL statements consuming significant database time were found. These statements offer a good opportunity for performance improvement.

Recommendation 1: SQL Tuning

Estimated benefit is .35 active sessions, 36% of total activity."

2. High commit rate

Analysis of the payroll application transactions indicated that the application was committing records to the database too frequently. The application was performing 1242 transactions per minute. A performance improvement of 2.9% can be gained by reducing the rate of commit and increasing the size of transactions.

Below An excerpt from ADDM

```
“ Finding 4: Commits and Rollbacks
Impact is .03 active sessions, 2.91% of total activity.
-----------------------------------------------------
Waits on event "log file sync" while performing COMMIT and ROLLBACK
operations
were consuming significant database time.

  Recommendation 1: Application Analysis
  Estimated benefit is .03 active sessions, 2.91% of total activity.
  ----------------------------------------------------------------
  Action
    Investigate application logic for possible reduction in the number of
    COMMIT operations by increasing the size of transactions.
  Rationale
    The application was performing 1242 transactions per minute with an
    average redo size of 1001 bytes per transaction.”
```

3. Scalability bottleneck.

While collecting user feedback, it was noted that the user indicated the payroll application seemed to slow down further with each year.

An experiment was set up to investigate the scalability of the payroll application.

Payroll databases with different historical data were considered and the average response times collected.

Payroll historical data	**Response time**
2 months	33
1 year	50
2 years	75
3 years	101

Table 8: Payroll historical data vs response time

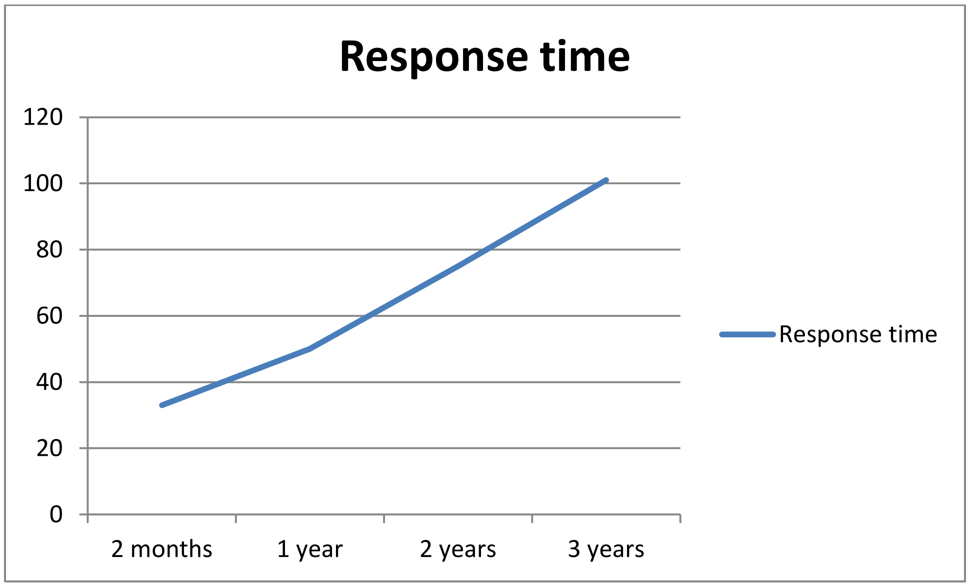


Figure 36: Payroll response time vs. historical data

As depicted in the chart the application scales very poorly as more data is added. Performance degrades by 67% (101-33)/101 * 100 if the database contains 3 years historical data as compared to having historical data of only two months. Each year performance degrades roughly by 20% as more data is added.

In summary the following bottlenecks have been identified in the payroll application.

1. Disk I/O bottleneck: This is attributed to application logic and poorly performing SQL, the bottleneck can be resolved by rewriting application logic and identified SQL statements or adding faster disks such as Solid State Disks. Estimated performance improvement 24%.
2. Poor performing SQL statements: Tuning Identified SQL statements offer opportunity for improving performance of the database application by 60 %. Moving to faster processors also realized 65% improvement in performance.
3. High Commit rate: Caused by small transaction size, can be resolved by increasing transaction size by rewriting application code. Estimated improve in performance 3%.
4. Poor scalability: The payroll application scales very poorly with an estimated 20% degrade in performance each year. Since current payroll contains 10 years historical data, archiving 9 years historical data can lead to 180% improvement in performance.

Step 7: Ranking in terms of cost and impact

Using the formulae

$$\boldsymbol{Rank} = \frac{\boldsymbol{Impact}}{\boldsymbol{Y*Cost*Time}}$$

Which is discussed in the methodology section for each bottleneck a rank was calculated.

For the values of Impact a percentage range was used and for the values of Cost, Time and Coefficient Y values in the ranges of [1 -10] were assigned based on the analysis and findings obtained in identification of the bottleneck section.

	Bottleneck	**Option to address bottleneck**	**Impact %**	**Cost**	**Time**	**Y**	**Rank**
1	Disk I/O bottleneck	Rewriting application logic and SQL	20	7	6	7	0.07
2	Disk I/O bottleneck	Move to faster Solid State Disks	20	4	1	7	0.71
3	Poor performing SQL statements	Tune SQL Statements	60	8	10	7	0.11
4	Poor performing SQL statements	Move to faster processors	60	3	1	7	2.86
5	High Commit rate	Rewriting application logic	2	6	7	7	0.07
6	Poor scalability	Archive old data	180	1	1	7	25.71

Table 9: Unranked bottlenecks

Step 8: Iterative resolution of identified bottlenecks in the tiers based on the ranking

Based on the rank calculated for each bottleneck, the table below orders the bottleneck by rank.

	Bottleneck	**Option to address bottleneck**	**Impact %**	**Cost**	**Time**	**Y**	**Rank**
1	Poor scalability	Archive old data	180	1	1	7	25.71
2	Poor performing SQL statements	Move to faster processors	60	3	1	7	2.86
3	Disk I/O bottleneck	Move to faster Solid State Disks	20	4	1	7	0.71
4	Poor performing SQL statements	Tune SQL Statements	60	8	10	7	0.11
5	Disk I/O bottleneck	Rewriting application	20	7	6	7	0.07

		logic and SQL					
6	High Commit rate	Rewriting application logic	2	6	7	7	0.07

Table 10 : Ranked bottlenecks

The performance goal was to have the payroll application process 6000 payroll records in less than two hours. This is a 50% improvement in performance.

By resolving the scalability bottleneck through archiving of old data performance of the application will improve by 180% and the target will we be met.

Further resolution of identified bottlenecks will lead to better improvement in payroll processing.

CHAPTER 5: DISCUSSIONS CONCLUSION AND RECOMMENDATIONS

5.1. ACHIEVEMENTS

The following were achieved as per the objectives set out in chapter one.

Objective 1: To study systems performance tuning especially in database oriented applications.

An in depth study of systems performing tuning was carried out. Various literatures were reviewed and factors that affect system performance were identified. It was realized that all tiers that an application interacts with can be a source of bottlenecks.

Objective 2: To review the existing application and database tuning methodologies with an intention of identifying gaps.

Various fine tuning methodologies were reviewed and detailed in the literature survey section. It was found that most existing methodology had some bias to a tier depending on the origin of the methodology. A summary and critique of existing methodology was given.

Objective 3: To develop a customized integrated holistic tuning approach that addresses identified gaps.

Using the reviewed methodology and addressing gaps that were identified, a holistic methodology was developed. The methodology focused on the following:

1. Tools for gathering and analyzing performance statistics in ach tier.
2. How to identify bottlenecks in each tier and suggestions on resolution of some of the identified bottlenecks.
3. Ranking of bottlenecks to assist in choosing which bottlenecks to resolve at minimal cost and time to the organization while achieving the greatest impact. A formula to aid in ranking was suggested.

Objective 3: To apply the customized methodology in a real world performance problem.

The developed methodology was applied to a poorly performing payroll application and the following bottlenecks identified.

1. Disk I/O bottleneck:
2. Poor performing SQL statements
3. High Commit rate
4. Poor scalability

Various options for resolving the bottlenecks were discussed and analyzed and their overall impact arrived at.

Best improvement in performance was realized by using faster processors and maintaining historical data to a minimum as illustrated in figure 39 and 40.

The ranking formulae were applied to each bottleneck and the bottlenecks ranked based on the overall impact, cost and time to resolve.

The bottlenecks that offered greatest improvement in performance at minimum cost and time to the Organization were identified.

5.2. VALIDATION OF THE CONCEPTUAL MODEL

In the beginning of the tuning exercise the following conceptual model was developed to guide the tuning exercise.

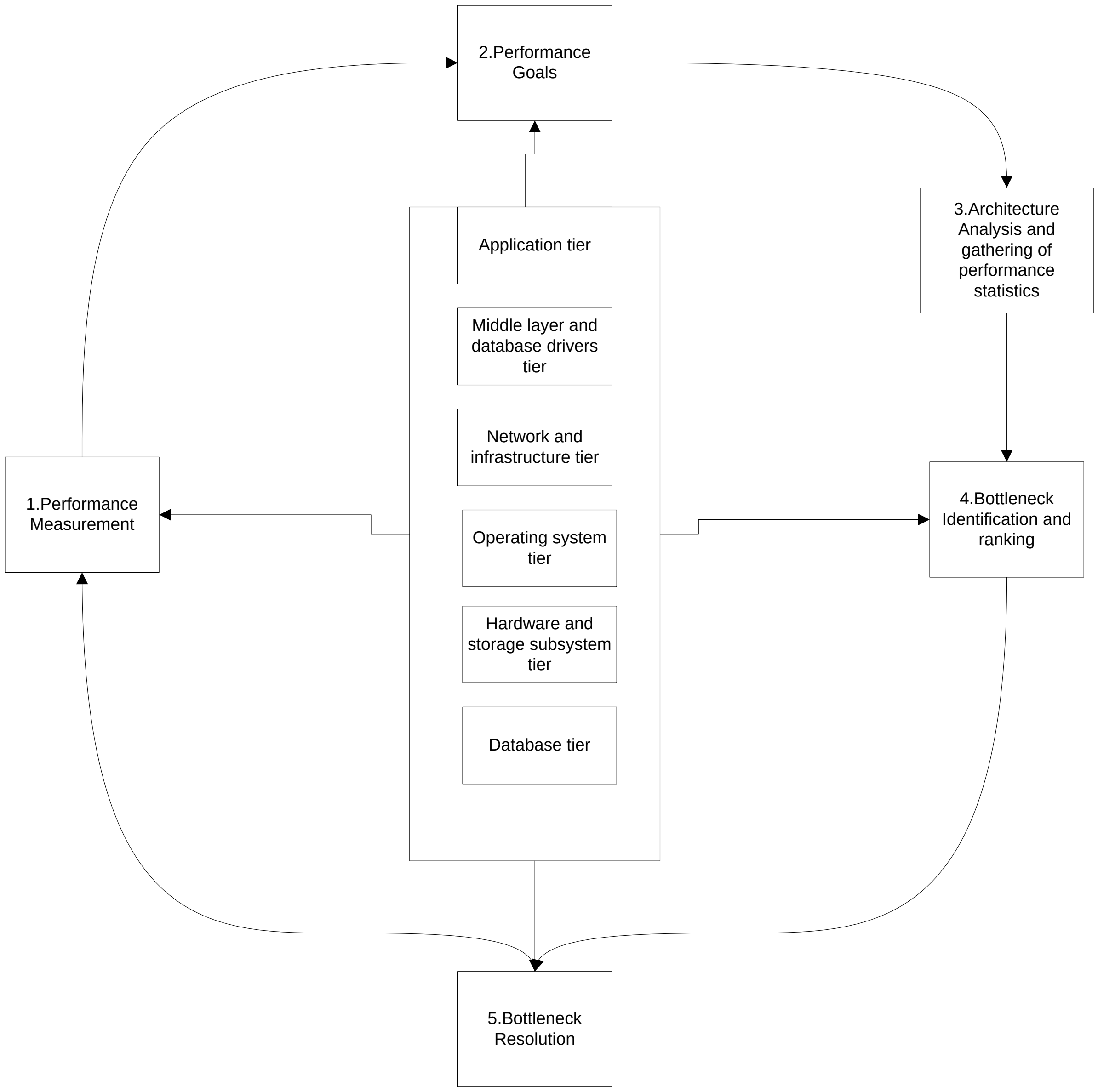

2.Performance Goals
3.Architecture Analysis and gathering of performance statistics
Application tier
Middle layer and database drivers tier
Network and infrastructure tier
Operating system tier
Hardware and storage subsystem tier
Database tier
1.Performance Measurement
4.Bottleneck Identification and ranking
5.Bottleneck Resolution

Figure 37: Conceptual model

Step I in the model is performance evaluation, this was carried and the following baseline performance measurement values collected

The response time was 101 secs to process 200 records translating to 0.5 secs per payroll record.

The throughput of the application was 200 payroll records in 101 secs, translates to 2 payroll records per second.

Step 2 in the model is setting of performance goals, in consultation with the user the following goal was set

"The payroll application must process 6000 payroll records in under two hours."

Step 3 involved architecture analyses and gathering of performance statistics, this was extensively carried out for all the tiers as depicted in Operating system and hardware statistics, CPU utilization, Memory
the tiers as detailed in section 4 and 5 in the Application of the Methodology chapter

Stage 4 in the model is Identification and ranking of bottlenecks, this was done and the following bottlenecks identified.

ii. Disk I/O bottleneck: This is attributed to application logic and poorly performing SQL, the bottleneck can be resolved by rewriting application logic and identified SQL statements or adding faster disks such as Solid State Disks. Estimated performance improvement 24%.

iii. Poor performing SQL statements: Tuning Identified SQL statements offer opportunity for improving performance of the database application by 60 %. Moving to faster processors also realized 65% improvement in performance.

iv. High Commit rate: Caused by small transaction size, can be resolved by increasing transaction size by rewriting application code. Estimated improve in performance 3%.

v. Poor scalability: The payroll application scales very poorly with an estimated 20% degrade in performance each year. Since current payroll contains 10 years historical data, archiving 9 years historical data can lead to 180% improvement in performance.

Ranking of the bottlenecks based on different methods of resolving them resulted to the following ranking table

	Bottleneck	**Option to address bottleneck**	**Impact %**	**Cost**	**Time**	**Y**	**Rank**
1	Poor scalability	Archive old data	180	1	1	7	25.71
2	Poor performing SQL statements	Move to faster processors	60	3	1	7	2.86
3	Disk I/O bottleneck	Move to faster Solid State Disks	20	4	1	7	0.71
4	Poor performing SQL statements	Tune SQL Statements	60	8	10	7	0.11
5	Disk I/O bottleneck	Rewriting application logic and SQL	20	7	6	7	0.07

6	High Commit rate	Rewriting application logic	2	6	7	7	0.07

Table 11: Ranked bottlenecks

Step 5 in the model is resolving identified bottlenecks until performance goal is met, The performance goal was to have the payroll application process 6000 payroll records in less than two hours. This is a 50% improvement in performance.

By resolving the scalability bottleneck through archiving of old data performance of the application will improve by 180% and the target was met. Further resolution of the bottleneck would lead to better improvement in performance.

Thus all the stages of the conceptual model were validated and found to hold.

5.3. VALIDATION OF THE RANKING FORMULAE

An attempt to validate the formulae proposed using data collected from the poorly performing application was made. This method of validating a formula is referred to as experimental as opposed to mathematically validating the formulae.

$$\mathbf{Rank} = \frac{\mathbf{Impact}}{\mathbf{Y * Cost * Time}}$$

The above formula used to rank the identified bottlenecks in the poorly performing application, Values of Impact were derived from various performance tools used to analyze the poorly performing application and also from experimentation.

Values of cost, time and Coefficient were arrived at with consultation with the user of the payroll application.

After application of the formulae using the values obtained it was found out that bottlenecks that had huge impact in overall improvement of performance and cost less and while taking minimum to resolve had the highest rank. A good example is the poor scalability bottleneck, it had an Impact of 180 % and Cost and Time values of 1 and coefficient of 7. The rank of the bottleneck was 25.71which was the highest.

5.4. LIMITATIONS AND CHALLENGES

In the developed integrated methodology illustrations were given using Linux and Windows operating systems which are perceived to be to be the most common operating systems.

In the developed integrated methodology option for switching to a different database system was not considered.

Obtaining different type of computers with different hardware configurations was a challenge.

5.5. DISCUSSIONS

Due to the importance of systems performance in the current business context the study set out to review existing fine tuning methodologies and develop a customized tuning methodology based on identified gaps and test the developed methodology on an existing application.

Many methodologies exist of tuning and optimizing application and databases separately, after reviewing available literature, it was discovered not much attention has been given to formulating a holistic approach of tuning applications and databases in a good documented and systematic approach.

Existing application tuning approaches are either application specific or are proposed by vendors whose tuning software's use the recommended approaches. Existing database tuning methodologies focus so much on the database that they downplay the role of other components such as the operating system and middle layer tier in overall performance of the application.

The study achieved its objectives of developing a customized holistic database and systems fine tuning methodology and demonstrating how the methodology can be used to tune a poorly performing payroll application.

Main stages of the developed methodology are:

1. Obtaining feedback from the user on how the system is behaving
2. Collecting baseline Performance measurement of the application when it is performing below users' expectation and when the system is performing well.
3. Setting realistic performance goals which are realistic.
4. Analyzing system infrastructure
5. Collecting resource utilization and performance statics of all the tiers involved.
6. Identification of bottlenecks by analyzing statistics collected, using various tools and experimentation.
7. Ranking of identified bottlenecks by using impact, Cost and Time. A formula for calculating the rank for each bottleneck is suggested.
8. Iterative resolution of identified bottlenecks in the tiers based on the ranking

Highlights of the methodology include:

1. Overview of various tools available for gathering resource utilization statistics in various operating systems.
2. Overview of tools for analyzing performance of various applications and databases.
3. Guidelines on how to identify bottlenecks in various tiers that have impact on system performance.
4. A simple formula that can be used to rank bottlenecks based on Impact, time and cost.

5. Tips on how to resolve bottlenecks in various tiers.

The developed methodology is can be used by system administrators, database administrators and performance engineer to trouble shoot and fine tune their applications and can be in cooperated in their standard operating procedures.

5.6. SUGGESTED FURTHER RESEARCH

Enhance the methodology and include illustrations from other operating systems.

Enhance the methodology give a comparative analysis of performance of different DBMS and what factors to consider in deciding to change the DBMS.

Mathematically prove and derive the suggested ranking formula and or use further experiments to prove it.

REFERENCES

Azeem, M., 2002, A Successful Performance Tuning Methodology Quest Software, Inc. (NASDAQ: QSFT).

Dennis, S. and Philippe, B., 1992. Database Tuning – Principles, Experiments and Troubleshooting Techniques, Prentice-Hall.

Donald, K., 2009, Inside Oracle fully automated SQL tuning, Burleson Consulting

Burleson, 2010, Oracle performance, hardware & RAM tuning optimization, Burleson Consulting

Chris Farrell , P., 2010, The Complete Guide to .NET Performance Testing and Optimization, [Online] Available at: <http://en.wikipedia.org/wiki/List_of_performance_analysis_tools.html> [Accessed , 24 November 2012]

Eduardo Ciliendo, T., 2007, Linux Performance and Tuning Guidelines, IBM

Fowler, Martin., 1997, Analysis Patterns, Reusable object models, Addison-Wesley Longman

Jitesh K., 2005, PeopleSoft Global Payroll Performance Analysis & Tuning approach, Tata Consultancy Services

John, G and Robert, A., 2009, The Data Access Handbook: Achieving Optimal Database Application Performance and Scalability. Prentice-Hall, Upper Saddle River, New Jersey

Mark, D., 2010, Guide for Developing High-Performance Database Applications, Oracle Corporation

Misty Faucheux, 2012, Windows Profiling Tools, eHow Contributor, [Online] Available at: <http://www.ehow.com/list_7166039_windows-profiling-tools.html> [Accessed 10 December 2012].

Immanuel, C and Lance, A., 2011, Oracle Database Performance Tuning Guide 11g Release 2 (11.2), Oracle Corporation

Sitansu, S.M., 2002. Database Performance Tuning and Optimization, Springer Verlag

Isam, A and Lester, G., 2011, Tuning All Layers of E-Business Suite, Oracle Corporation

Kirill Loifman, 2012, Oracle ADDM shows Virtual Memory Paging on 10gR2 11gR2, [Online] Available at: <https://forums.oracle.com/forums/thread.jspa?threadID=2132991> [Accessed 09 February 2013].

Oracle - B10500_01, 2002, Monitoring and Improving Application Performance, Oracle Corporation

Oracle, 2010, Rapid Bottleneck Identification A Better Way to do Load Testing, Oracle Corporation

Quest, S., 2010, Performance Tuning for Mission-Critical Database Applications an Example in PeopleSoft, Quest Software (Nasdaq: QSFT)

Thomas, 2012, Web applications Performance Symptoms and Bottlenecks Identification, Agiletoad

Toadworld., 2010, 5 Step Tuning Methodology – Overview, [Online] Available at: <http://www.toadworld.com/KNOWLEDGE/KnowledgeXpertforOracle/tabid/648/TopicID/OPS3/Default.aspx> [Accessed 1 Feb 2012].

University of Texas,2010, Conduct research , [Online] Available at: <http://www.utexas.edu/academic/ctl/assessment/iar/research/plan/method/content.php> [Accessed 12 January 2013].

APPENDIX A: RESOURCE UTILIZATION COLLECTION TOOLS

To collect resource utilization statistics (IBM, 2011) in Linux Performance and Tuning Guidelines gives a good overview of the available Linux tools and their function.

Tool	**Most useful tool function**
Top	Process activity
Vmstat	System activity, Hardware and system information
uptime, w	Average system load
ps, pstree	Displays the processes
Free	Memory usage
Iostat	Average CPU load, disk activity
Sar	Collect and report system activity
Mpstat	Multiprocessor usage
Numastat	NUMA-related statistics
Pmap	Process memory usage
Netstat	Network statistics
Iptraf	Real-time network statistics
tcpdump, ethereal	Detailed network traffic analysis
Nmon	Collect and report system activity
Strace	System calls
Proc file system	Various kernel statistics
KDE system guard	Real-time systems reporting and graphing
Gnome System Monitor	Real-time systems reporting and graphing
Lmbench	Microbenchmark for operating system functions
Iozone	File system benchmark
Netperf	Network performance benchmark

Table 12: Linux resource utilization tools

For windows in (arik books) in his Operating System and Process Monitoring Tools survey provides a good description the available windows monitoring tools.

Tool	**Most useful tool function**
Task Manager (taskmgr)	fast look into the current system state
Performance Monitor (perfmon)	acts as both a real time and log-based performance monitoring tool
Process Monitor (pmon)	real-time monitoring of process performance
Process Explode (pview)	provides a vast amount of performance data of processes, threads, memory, and the system in general.

Process Viewer (pviewer)	shows a subset of the process performance measurements shown by Process Explode

Table 13: Windows resource utilization tools

The table below lists some of the tools that can be used for database monitoring and resource utilization for main of the databases.

Vendor	**Database Target**	**Tool Name**	**Most useful tool function**
Oracle	Oracle	Oracle Enterprise manager	• Provide real time statistics of the following resources utilized by the database • Cpu • Processors • Disk Usage • Memory usage • Sessions consuming most resources • Users consuming most resource • Wait analysis
Oracle	Oracle	ADDM	• Analyzes statistics to provide automatic • Diagnosis of major performance issues.
Jayam Systems	Oracle	MyOra	• Real Time Performance Monitoring using graphs, bar charts & pie charts. • Multiple Performance Monitoring Windows for same or different databases. • Ability to show System Waits using pie charts. • View Top 5 SQLs and its details using interactive 3-D bar charts. • View Top 5 Resource Using Sessions and its details using interactive 3-D bar charts.
IBM	DB2	Monitor Switches	Activates collections of DB2 useful information such as time spent in sorts and other important tuning information
IBM	DB2	Snapshot Monitor Administrative Views	Views that allow querying of performance information by use of SQL
IBM	DB2	Activity Monitor	GUI tool that monitors the following • application performance and concurrency

			• resource consumption • SQL statement usage of a database or database partition.
Microsoft	SQL Server	SQL Server Management Studio	The Activity Monitor in SQL Server Management Studio graphically displays information about: • Processes running on an instance of SQL Server. • Blocked processes. • Locks. • User activity.
Microsoft	SQL Server	sp_trace_setfilter (Transact-SQL)	• SQL Server Profiler tracks engine process events • Monitor server and database activity such as • deadlocks • fatal errors • login activity
Red Gate Software Ltd	SQL Server	SQL Monitor	• Real time performance data • Long running queries • Blocked processes • Centralized alerts
Sun Oracle	Mysql	Mysqladmin	• Provides tool for querying MySQL performance counters increment • Contains information about server run status
Percona	Mysql	Percona Toolkit	• MySQL Performance Analyses and maintenance tools • Summarize MySQL servers • Analyze queries from logs and tcpdump • Collect vital system information when problems occur
Webyog	Mysql	Monyog	• Deadlock Monitoring • Query Analyzer with Query Sniping • Disk Monitoring & Lock Monitoring • Real-Time Monitoring • Error log Monitoring

Table 14: Database monitoring and resource utilization tool

:

Application gathering statistics tools

Vendor	**Development Platform Target**	**Tool Name**	**Most useful tool function**
CodersNotes	Windows based applications	Very Sleepy	Save application performance information in documents Generates call graphs - graphical representation of procedures and how they relate to each other C and C++ CPU (central processing unit) profiler.
Microsoft	Windows-based applications and Windows-hosted applications	Microsoft Source Profiles PROFILE.EXE. PROFILEW.EXE	Collect statistics and information about Windows applications
Microsoft	Windows based applications	VSPerfCmd	Allows profiling applications in a certain mode Restrict access to the profiler. Profile applications over all Windows sessions. Profile using commands. Set how often application samples should be taken.
Microsoft	Windows based applications	Xperf	Visualizer for analyzing Event Trace Log (ETL) files.
Microsoft	.NET Applications	CLR Profiler	memory profiling tool
Red Gate	.NET Applications	ANTS Memory and Performance Profiler 5.0	Memory and Performance Profililing
Linux	Linux based applications	Linux perf tools	Profile application and hardware use

Table 15: Application gathering statistics tools

In addition to the above tools (en.wikipedia.org) in an article titled List of performance analysis tools provides an overview of the following tools for analyzing performance of applications

Name/Vendor	**Operating System**	**Development platform**	**Most useful tool function**	**License**
AQtime by Smart Bear Software	Windows	.NET 1.0 to 4.0 applications (including ASP.NET applications), Silverlight 4.0 applications, Windows 32- and 64-bit applications including C, C++, Delphi for Win32 and VBScript and JScript functions	Performance profiler and memory/resource debugging toolset	Proprietary
CodeAnalyst by AMD	AMD, Intel hardware x86 based machines	Linux, Windows	GUI based code profiler; does only basic timer-based profiling on Intel processors. Based on OProfile.	Free/open source (GPL) or proprietary
Caliper by HP	HP-UX with Intel Itanium Integrity platform (IA-64).		Profiling tool	
DevPartner by Mi cro Focus		.NET, Java	Test suite that automatically detects and diagnoses software defects and performance problems.	Proprietary
DTrace by Sun Microsystems	Solaris, Linux, BSD, Mac OS X		Comprehensive dynamic tracing framework for troubleshooting kernel and application problems on production systems in real time.	Free/open source (CDDL)

Name/Vendor	**Operating System**	**Development platform**	**Most useful tool function**	**License**
dynaTrace byCompuware		.NET, Java, AJAX (for web sites)	Application Performance Management	Proprietary
DynInst	Linux, Windows,BlueGene/Q		API to allow dynamic injection of code into a running program	Free/open source
GlowCode	Windows	64-bit and 32-bit applications, C, C++, .NET, and dlls generated by any language compiler.	Performance and memory profiler which identifies time-intensive functions and detects memory leaks and errors	Proprietary
gprof	Linux/Unix	Any language supported by gcc	Several tools with combined sampling and call-graph profiling. A set of visualization tools, VCG tools, uses the Call Graph Drawing Interface (CGDI) to interface with gprof. Another visualization tool which interfaces with gprof is KProf.	Free/open source - BSD version is part of 4.2BSD and GNU version is part of GNU Binutils (by GNU Project)
Linux Trace Toolkit	Linux	Requires patched kernel	Collects data on processes blocking, context switches, and execution time. This helps identify	

Name/Vendor	**Operating System**	**Development platform**	**Most useful tool function**	**License**
			performance problems over multiple processes or threads. Superceded by LTTng.	
LTProf	Windows	Visual C++, Borland CBuilder, Delphi and VB applications	CPU profiling tool	
LTTng (Linux Trace Toolkit Next Generation)	Linux		System software package for correlated tracing of kernel, applications and libraries	
OProfile[2]	Linux	Profiles everything running on the Linux system, including hard-to-profile programs such as interrupt handlers and the kernel itself.	Sampling profiler for Linux that counts cache misses, stalls, memory fetches, etc.	
Oracle Solaris StudioPerformance Analyzer[3]	Linux, Solaris	C, C++, Fortran, Java; MPI	Performance and memory profiler	
Paraver	Linux, Mac OS X, Windows [4]	For parallel computing clusters	Performance analysis tool based on trace files; allows viewing the progress of the application in a temporal axis and also perform accumulation of performance metrics in a table like regular profilers.	Free/open source (LGPL)

Name/Vendor	**Operating System**	**Development platform**	**Most useful tool function**	**License**
PGPROF by The Portland Group	Linux, MacOS X, Windows	C, C++, and Fortran applications using OpenMP and MPI parallelism	Sampling and compiler-based instrumentation for application profiling	Proprietary
PmcTools	FreeBSD		Provides non-intrusive, low-overhead and innovative ways of measuring and analysing system performance. It exploits the same underlying counters as Linux'OProfile.	
perf tools	Linux kernel 2.6.31+		Sampling profiler	
Performance Application Programming Interface(PAPI)	Various		Library for hardware performance counters on modern microprocessors	
Pin by Intel	Linux, Windows		Dynamic binary instrumentation system that allows users to create custom program analysis tools	Proprietary but free for non-commercial use
pprof, part of gperftools by Google			Sampling profiler with context-sensitive call graph capability.	
Rational	AIX, Linux, Solaris,		Performance profiling tool, memory	Proprietary

Name/Vendor	**Operating System**	**Development platform**	**Most useful tool function**	**License**
PurifyPlus	Windows		debugger and code coverage tool	
Shark by Apple	Mac OS X (discontinued with 10.7)		Performance analyzer	Free
SlowSpotter and ThreadSpotter by Acumem	Linux, Solaris	Most compiled languages including Ada	Diagnose performance problems related to data locality, cache utilization and thread interactions.	
Sysprof	Linux		Sampling CPU profiler that uses a kernel module to profile the entire system, as opposed to a single application. It displays the time spent in each branch of the applications' calltrees.[7]	
Systemtap	Linux		Programmable system tracing/probing tool; may be scripted to generate time- or performance-counter- or function-based profiles of the kernel and/or its userspace.	
Valgrind	Linux	Any, including assembler	System for debugging and profiling; supports	Free/open source

Name/Vendor	Operating System	Development platform	Most useful tool function	License
			tools to either detect memory management and threading bugs, or profile performance (cachegrind and callgrind).KCacheGrind , valkyrie and alleyoop are front-ends for valgrind.	(GPL)
VTune Amplifier XE byIntel Corporation	Linux, Windows	C, C++, Fortran, .NET, Java	Tool for serial and threaded performance analysis. Hotspot, call tree and threading analysis works on both Intel and AMD x86 processors. Hardware event sampling that uses the on chip performance monitoring unit requires an Intel processor.	Proprietary
RotateRight Zoom	Linux	Supports most compiled languages on ARM and x86 processors.	Graphical and command-line statistical (event-based) profiler	

Table 16: Application performance analysis tools

APPENDIX B: ADDM OUTPUT REPORT

```
ADDM Report for Task 'ADDM:1229390655_1_1965'
        -------------------------------------------

Analysis Period
---------------
AWR snapshot range from 1964 to 1965.
Time period starts at 12-FEB-13 06.00.19 AM
Time period ends at 12-FEB-13 06.08.41 AM

Analysis Target
---------------
Database 'ORCL' with DB ID 1229390655.
Database version 11.2.0.2.0.
ADDM performed an analysis of instance orcl, numbered 1 and hosted at
localhost.localdomain.

Activity During the Analysis Period
-----------------------------------
Total database time was 489 seconds.
The average number of active sessions was .97.

Summary of Findings
-------------------
   Description                           Active Sessions      Recommendations
                                      Percent of Activity
   ---------------------------------------  -------------------  ---------------
1  Virtual Memory Paging                  .97 | 100            1
2  Top SQL Statements                     .58 | 60             2
3  Top Segments by "User I/O" and "Cluster"  .24 | 24.77       1
4  Commits and Rollbacks                  .03 | 2.91           2


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


          Findings and Recommendations
          ----------------------------

Finding 1: Virtual Memory Paging
Impact is .97 active sessions, 100% of total activity.
-----------------------------------------------------
Significant virtual memory paging was detected on the host operating system.

   Recommendation 1: Host Configuration
   Estimated benefit is .97 active sessions, 100% of total activity.
   ----------------------------------------------------------------
   Action
      Host operating system was experiencing significant paging but no
      particular root cause could be detected. Investigate processes that do
      not belong to this instance running on the host that are consuming
      significant amount of virtual memory. Also consider adding more physical
      memory to the host.
```

```
Finding 2: Top SQL Statements
Impact is .58 active sessions, 60% of total activity.
----------------------------------------------------
SQL statements consuming significant database time were found. These
statements offer a good opportunity for performance improvement.

   Recommendation 1: SQL Tuning
   Estimated benefit is .35 active sessions, 36% of total activity.
   --------------------------------------------------------------
   Action
      Run SQL Tuning Advisor on the DELETE statement with SQL_ID
      "94zaccf3h3zgw".
      Related Object
         SQL statement with SQL_ID 94zaccf3h3zgw.
         DELETE FROM PROCESSED_DATA WHERE PROCESSED_DATA.PAYROLL_NO = :b1  AND
         PROCESSED_DATA.PRD_CODE = :b2
   Rationale
      The SQL spent 100% of its database time on CPU, I/O and Cluster waits.
      This part of database time may be improved by the SQL Tuning Advisor.
   Rationale
      Database time for this SQL was divided as follows: 100% for SQL
      execution, 0% for parsing, 0% for PL/SQL execution and 0% for Java
      execution.
   Rationale
      SQL statement with SQL_ID "94zaccf3h3zgw" was executed 935 times and had
      an average elapsed time of 0.11 seconds.
   Rationale
      At least one execution of the statement ran in parallel.
   Rationale
      Full scan of TABLE "P15_2680_94.PROCESSED_DATA" with object ID 101437
      consumed 100% of the database time spent on this SQL statement.

   Recommendation 2: SQL Tuning
   Estimated benefit is .19 active sessions, 20% of total activity.
   --------------------------------------------------------------
   Action
      Run SQL Tuning Advisor on the SELECT statement with SQL_ID
      "fkv43164kf4rz".
      Related Object
         SQL statement with SQL_ID fkv43164kf4rz.
         SELECT COUNT(*)   FROM PROCESSED_DATA  WHERE
         PROCESSED_DATA.PAYROLL_NO = :b1  AND PROCESSED_DATA.PRD_CODE = :b2
   Rationale
      The SQL spent 83% of its database time on CPU, I/O and Cluster waits.
      This part of database time may be improved by the SQL Tuning Advisor.
   Rationale
      Database time for this SQL was divided as follows: 100% for SQL
      execution, 0% for parsing, 0% for PL/SQL execution and 0% for Java
      execution.
   Rationale
      SQL statement with SQL_ID "fkv43164kf4rz" was executed 935 times and had
      an average elapsed time of 0.099 seconds.
   Rationale
      At least one execution of the statement ran in parallel.
   Rationale
      Full scan of TABLE "P15_2680_94.PROCESSED_DATA" with object ID 101437
      consumed 83% of the database time spent on this SQL statement.


Finding 3: Top Segments by "User I/O" and "Cluster"
Impact is .24 active sessions, 24.77% of total activity.
-------------------------------------------------------
Individual database segments responsible for significant "User I/O" and
"Cluster" waits were found.

   Recommendation 1: Segment Tuning
   Estimated benefit is .24 active sessions, 24.77% of total activity.
   ------------------------------------------------------------------
   Action
      Run "Segment Advisor" on TABLE "P15_2680_94.PROCESSED_DATA" with object
      ID 101437.
      Related Object
         Database object with ID 101437.
   Action
      Investigate application logic involving I/O on TABLE
      "P15_2680_94.PROCESSED_DATA" with object ID 101437.
      Related Object
         Database object with ID 101437.
   Action
      Look at the "Top SQL Statements" finding for SQL statements consuming
      significant I/O on this segment. For example, the DELETE statement with
      SQL_ID "94zaccf3h3zgw" is responsible for 100% of "User I/O" and
      "Cluster" waits for this segment.
   Rationale
      The I/O usage statistics for the object are: 97240 full object scans,
      52098200 physical reads, 3233 physical writes and 52098200 direct reads.

   Symptoms That Led to the Finding:
   ---------------------------------
      Wait class "User I/O" was consuming significant database time.
      Impact is .24 active sessions, 24.77% of total activity.


Finding 4: Commits and Rollbacks
Impact is .03 active sessions, 2.91% of total activity.
------------------------------------------------------
Waits on event "log file sync" while performing COMMIT and ROLLBACK operations
were consuming significant database time.

   Recommendation 1: Application Analysis
   Estimated benefit is .03 active sessions, 2.91% of total activity.
   ------------------------------------------------------------------
   Action
      Investigate application logic for possible reduction in the number of
      COMMIT operations by increasing the size of transactions.
   Rationale
      The application was performing 1242 transactions per minute with an
      average redo size of 1001 bytes per transaction.

   Recommendation 2: Host Configuration
   Estimated benefit is .03 active sessions, 2.91% of total activity.
   ---------------------------------------------------------------
   Action
      Investigate the possibility of improving the performance of I/O to the
      online redo log files.
   Rationale
      The average size of writes to the online redo log files was 0 K and the
      average time per write was 3 milliseconds.
   Rationale
      The total I/O throughput on redo log files was 0 K per second for reads
      and 30 K per second for writes.
   Rationale
      The redo log I/O throughput was divided as follows: 0% by RMAN and
      recovery, 100% by Log Writer, 0% by Archiver, 0% by Streams AQ and 0% by
      all other activity.

   Symptoms That Led to the Finding:
   ---------------------------------
      Wait class "Commit" was consuming significant database time.
      Impact is .03 active sessions, 2.91% of total activity.



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

          Additional Information
          ----------------------

Miscellaneous Information
-------------------------
Wait class "Application" was not consuming significant database time.
Wait class "Concurrency" was not consuming significant database time.
Wait class "Configuration" was not consuming significant database time.
CPU was not a bottleneck for the instance.
Wait class "Network" was not consuming significant database time.
Session connect and disconnect calls were not consuming significant database
time.
Hard parsing of SQL statements was not consuming significant database time.
```

APPENDIX C: EXPERMENTATION DATA SHEETS

Intel 2 duo core,8GB RAM, HOST os Solaris 32 bit,														
Application server setings cpu 1, memory 2GB Held constant														
Database memory 1 GB held constant														
Number of cpus available to database servers	start	stop	secs	start	stop	secs	start	stop	Sec	start	stop	Secs	Response Time (Seconds)	Average
1	11;32:55	11;34:11	76	11:40;41	11;42:06	85	11:46:29	11:47:57	88	11:52:58	11:51:30	88	84	337
2	10:33:02	10.35.12	130	10:59:50	11:02:28	158	11:07:53	11'10:29	156	11:16:51	11:19:30	159	215	0.93
3														
4														
HP I7 4 cores,8GB RAM, HOST Windows 64 bit,														
Application server setings cpu 1, memory 2GB Held constant														
Database memory 1 GB held constant														
Payroll data for only two months existing														
Number of cpus available	start	stop	secs	start	stop	secs	start	stop	sec	start	stop	Secs	Response Time (Seconds)	total

to database servers														
1	4:42:10	4:42:40	30	4:31:31	4:32:02	31	4:35:44	4:36:14	30	4:39:14	4:39;43	29	30	120
2	4:50:27	4:51:01	34	4:54:35	4:55:09	34	4:59:24	4:59:58	34	5:02:39	5:03;13	34		
3	5:13:21	5:13:59	38	5:17:57	5:18:36	39	7:31:59	7:32:39	40	7:34:48	7:35:29	41		
4	7:42:34	7:43:20	46	7:47:32	7:48:19	47	7:51:11	7:52:02	51	7:53:39	7:54:28	0:00:00		
Memory tests for database server														
cpu 1, application server settings held constant,memory allocated to database changed to 40% of total memory														
Memory available to database servers	start	stop	secs	start	stop	secs	start	stop	sec	start	stop	secs		
1GB	8:55:23	8:55:54	31	8:57:32	8:58:03	31	9:00:21	9:00:53	32	9:03:35	9:04:06	31		
2GB	10:19:38	10:20:10	32	10:22:57	10:23:29	32	10:34:37	10:35:11	34	10:37:51	10:38:23	32		
3GB	10:54:55	10:55:29	34	11:01:19	11:01:52	33	11:08:52	11:09:24	32	11:11:52	11:12:24	32		
4GB	12:58:36	12:59:10	34	1:01:43	1:02:19	36	1:05:41	1:06:16	35	1:09:20	1:09:54	34		
HP I7 4 cores,8GB RAM, HOST Windows 64 bit,														
Testing using payroll data for previous years, application server, 1 cpu,3GB, database server 1cpu, 4GB														

Historicsl data available	start	stop	secs	start	stop	secs	Average response time							
2 months	5:58:28	5:59:00	32	6:06:21	6:06:54	33	33							
1 year	5:29:34	5:30:24	50	5:39:21	5:38:31	50	50							
2 years	4:50:18	4:51:32	74	4:55:13	4:56:28	75	75							
3 years	7:11:13	7:12:53	100	7:17:11	7:18:52	101	101							
Dell poweredge server 64 GB RAM, Intel Xeon 4807, 1.87 GHZ, details provided														
Application server setings cpu 1, memory 2GB Held constant														
Database memory 1 GB held constant														
Payroll data for only two months existing														
Number of cpus available to database servers	start	stop	secs	start	stop	secs	start	stop	sec	start	stop	secs	Avg response time	
1	4:55:46	4:57:08	82	4:59:55	5:01:10	75	5:04:34	5:05:54	80	5:21:24	5:22:48	82	80	319

Table 17: Experimental data sheets