Project Report

On

Resource Utilization Analyzer Using Server Farms

Prepared by: Suhas K S & Shwetha G

S.J.C.E. Mysore, Eighth Semester : Computer Science

ACKNOWLEDGEMENT

"Experience is the best teacher", is an old adage. The satisfaction and pleasure that accompany the gain of experience would be incomplete without mention of the people who made it possible.

First and foremost we ought to pay due regards to this renowned institution, which provided us a platform and an opportunity for carrying out this project work.

We are especially grateful to our Mentor Mr. Santhosh Hegde & our Assistant Professor Mr. Anil Kumar K .M for their whole hearted encouragement and guidance in carrying out this project.

Students.

Table of Contents

CHAPTER I:

SOFTWARE REQUIREMENT SPECIFICATION

1. Introduction

1.1) Purpose

Objective of project is to build a resource utilization analyzer for server farms. This tool can be used to analyze the usage patterns of servers over time in server farms and help manage the available resources better.

The time stamped utilization data from each server (like memory usage, cpu usage , network load etc) can be stored in a informix database using Informix time series data blade and analyzed quickly because of inherent performance provided by the time series data blade.

1.2) Definitions and Abbreviations:

DEFINITIONS:

Resource: This can be defined as any physical or virtual entity of limited availability that needs to be consumed to obtain benefit from it.

In other words A system resource is any part of the computer such as disk drives, memory capacity, processor time that might be used by a computer program. This is allocated by OS so that programs can run efficiently.

Server Farms: A server farm or server cluster is a collection of computer servers usually maintained by an enterprise to accomplish server needs far beyond the capability of one machine. Server farms often have backup servers, which can take over the function of primary servers in the event of a primary server failure. Server farms are typically collocated with the network switches and/or routers which enable communication between the different parts of the cluster and the users of the cluster.

Triggers: The SQL CREATE TRIGGER statement provides a way for the database management system to actively control, monitor, and manage a group of tables whenever an insert, update, or delete operation is performed. The statements specified in the SQL

trigger are executed each time an SQL insert, update, or delete operation is performed. An SQL trigger may call stored procedures or user-defined functions to perform additional processing when the trigger is executed.

Unlike stored procedures, an SQL trigger cannot be directly called from an application. Instead, an SQL trigger is invoked by the database management system on the execution of a triggering insert, update, or delete operation. The definition of the SQL trigger is stored in the database management system and is invoked by the database management system, when the SQL table, that the trigger is defined on, is modified.

In a database, a trigger is a set of Structured Query Language (SQL) statements that automatically "fires off" an action when a specific operation, such as changing data in a table, occurs. A trigger consists of an event (an INSERT, DELETE, or UPDATE statement issued against an associated table) and an action (the related procedure). Triggers are used to preserve data integrity by checking on or changing data in a consistent manner.

1.3) References

The references are as follows:

- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.
- Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. Operating system principles, 7th edition, Wiley India Edition.
- http://web.inter.nl.net/hcc/J.Steunebrink/chkcpu.html
- http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=%2Fsqlp%2Frbafysqltrig.html

1.4) Overview

The remainder of this document, the first providing a full description of the project requirements and analysis. It lists all the functions performed by the system. The second chapter concentrates on the conceptual detailed description and theoretical implementation of the system. The third chapter concerns details of each of the system functions and actions in full for the software developer's assistance.

2. Overall Description

2.1) Product Perspective

This software proposes a viable analyzing of the resource utilization of systems which mainly concentrates on the analysis to resource consumption rather than considering atomicity. Our project mainly concentrates on generation of warning message to the admin informing about the status of server resource utilization status periodically.

2.2) Product Features

- Database maintaining information of resource utilization of all servers in server farms.
- While updating databases automatic status calculation is used to achieve ease of use.
- Flexible algorithm for quick adaption to specific needs.
- Fully open source environment is used to build the project.
- Unlimited number of tuples can be maintained in the database.
- Includes MIS reports.
- Scheduled the entire program synchronously.

- Clear separation is maintained between server and client side.
- Official data completely secured.
- Simulation of whole project (different modules) is seen through a same window.

2.3) General Constraints

- The Server and client should be connected properly.
- Client should know the server's ip address.
- Client should be working in Linux platform and server should be in the windows platform.

2.4) Assumptions and Dependencies

The project built is just a simulation. Here we assume that a single computer is used to perform operations of server simultaneously other computer acts as n- clients. I.e. the problem is simulated for 1 server and 1 client then it is assumed that it for any number of clients.

3) Specific requirements

Supported Operating Systems

CLIENT: **Linux hosts** (32-bit and 64-bit). Among others, this includes:

- Ubuntu 9.04 ("Jaunty Jackalope") and onwards.
- Ubuntu ultimate 2.3 and onwards.

SERVER: **Windows (**32 bit or 64 bit)

- Xp sp2/ XP sp3/ windows 7

3.1) External interface requirements

3.1.1) User interface

- Mainly the software runs in the background and hence except admin to see the result in warning status the working of the software in hidden from the other users.

3.1.2) Hardware Interface

Minimum Requirements:

- 500 MHz processor
- 256 MB Main Memory
- INTEL Architecture machines

3.1.3) Software Interface

Analyzer is totally a set of object oriented program which stores data in Informix database. Hence embedded SQL/java(JDBC) is used as a development tool.

4. Other Nonfunctional Requirements

4.1) Performance Requirements

- Should run on 500 MHz, 64 MB machine.
- 90% of the responses should be within 2 sec, except if other processes are in the running state.

4.2) Safety Requirements

It should provide for Fault Tolerance. Data stored in a data repository, regarding the resource data information should not become corrupted in case of a system crash or power failure.

4.3) Security Requirements

- The files in which the resource information is present should be secured against malicious deformations or intrusions.
- The information of the administrator should be isolated from other actors.

5. Other Requirements

5.1) Database Requirement

- The software should provide a data repository which stores all the resource namely cpu, memory, disk, network information of all clients. It should also update the status table to insert warning message without error.

CHAPTER II:

LITERATURE

A typical transaction-processing application undergoes different demands throughout its various operating cycles. Peak loads during the day, week, month, and year, as well as the loads imposed by decision-support (DSS) queries or backup operations, can have significant impact on any system that is running near capacity. You can use direct historical data derived from your particular system to pinpoint this impact.

You must take regular measurements of the workload and performance of your system to predict peak loads and compare performance measurements at different points in your usage cycle. Regular measurements help you to develop an overall performance profile for your database server applications. This profile is critical in determining how to improve performance reliably.

For the measurement tools that the database server provides, see Database Server Tools. For the tools that your operating system provides for measuring performance impacts on system and hardware resources, see Operating-System Tools.

Utilization is the percentage of time that a component is actually occupied, as compared with the total time that the component is available for use. For instance, if a CPU processes transactions for a total of 40 seconds during a single minute, its utilization during that interval is 67 percent.

Measure and record utilization of the following system resources regularly:

- CPU
- Memory
- Disk

A resource is said to be *critical* to performance when it becomes overused or when its utilization is disproportionate to that of other components. For instance, you might consider a disk to be critical or overused when it has a utilization of 70 percent and all other disks on the system have 30 percent. Although 70 percent does not indicate that the disk is severely overused, you could improve performance by rearranging data to balance I/O requests across the entire set of disks.

How you measure resource utilization depends on the tools that your operating system provides for reporting system activity and resource utilization. Once you identify a resource that seems

overused, you can use database server performance-monitoring utilities to gather data and make inferences about the database activities that might account for the load on that component. You can adjust your database server configuration or your operating system to reduce those database activities or spread them among other components. In some cases, you might need to provide additional hardware resources to resolve a performance bottleneck.

Resource Utilization

Whenever a system resource, such as a CPU or a particular disk, is occupied by a transaction or query, it is unavailable for processing other requests. Pending requests must wait for the resources to become available before they can complete. When a component is too busy to keep up with all its requests, the overused component becomes a bottleneck in the flow of activity. The higher the percentage of time that the resource is occupied, the longer each operation must wait for its turn.

You can use the following formula to estimate the service time for a request based on the overall utilization of the component that services the request. The expected service time includes the time that is spent both waiting for and using the resource in question. Think of service time as that portion of the response time accounted for by a single component within your computer, as the following formula shows:

$$S = P/(1-U)$$

S

is the expected service time.

P

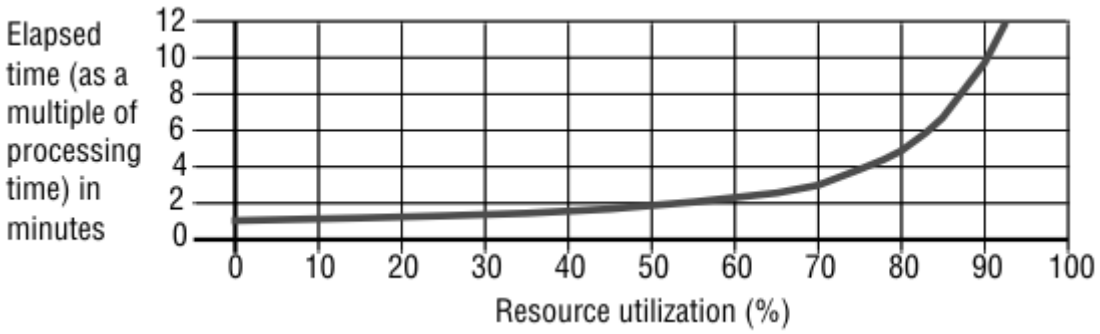
is the processing time that the operation requires once it obtains the resource.

U

is the utilization for the resource (expressed as a decimal).

As Figure 2 shows, the service time for a single component increases dramatically as the utilization increases beyond 70 percent. For instance, if a transaction requires 1 second of processing by a given component, you can expect it to take 2 seconds on a component at 50 percent utilization and 5 seconds on a component at 80 percent utilization. When utilization for the resource reaches 90 percent, you can expect the transaction to take 10 seconds to make its way through that component.

Figure 2. Service Time for a Single Component as a Function of Resource Utilization



If the average response time for a typical transaction soars from 2 or 3 seconds to 10 seconds or more, users are certain to notice and complain.

Important:

Monitor any system resource that shows a utilization of over 70 percent or any resource that exhibits symptoms of overuse as described in the following sections.

When you consider resource utilization, also consider whether increasing the page size of a standard or temporary dbspace is beneficial in your environment. If you want a longer key length than is available for the default page size of a standard or temporary dbspace, you can increase the page size.

CPU Utilization

You can use the resource-utilization formula from the previous section to estimate the response time for a heavily loaded CPU. However, high utilization for theCPU does not always indicate a performance problem. The CPU performs all calculations that are needed to process transactions. The more transaction-related calculations that it performs within a given period, the higher the throughput will be for that period. As long as transaction throughput is high and seems to remain proportional to CPU utilization, a high CPU utilization indicates that the computer is being used to the fullest advantage.

On the other hand, when CPU utilization is high but transaction throughput does not keep pace, the CPU is either processing transactions inefficiently or it is engaged in activity not directly related to transaction processing. CPU cycles are being diverted to internal housekeeping tasks such as memory management. You can easily eliminate the following activities:

- Large queries that might be better scheduled at an off-peak time
- Unrelated application programs that might be better performed on another computer

If the response time for transactions increases to such an extent that delays become unacceptable, the processor might be swamped; the transaction load might be too high for the computer to manage. Slow response time can also indicate that the CPU is processing transactions inefficiently or that CPU cycles are being diverted.

When CPU utilization is high, a detailed analysis of the activities that the database server performs can reveal any sources of inefficiency that might be present due to improper configuration. For information about analyzing database server activity, see Database Server Tools.

Memory Utilization

Although the principle for estimating the service time for memory is the same as that described in Resource Utilization and Performance,, you use a different formula to estimate the

performance impact of memory utilization than you do for other system components. Memory is not managed as a single component such as a CPU or disk, but as a collection of small components called *pages*. The size of a typical page in memory can range from 1 to 8 kilobytes, depending on your operating system. A computer with 64 megabytes of memory and a page size of 2 kilobytes contains approximately 32,000 pages.

When the operating system needs to allocate memory for use by a process, it scavenges any unused pages within memory that it can find. If no free pages exist, the memory-management system has to choose pages that other processes are still using and that seem least likely to be needed in the short run. CPUcycles are required to select those pages. The process of locating such pages is called a *page scan*. CPU utilization increases when a page scan is required.

Memory-management systems typically use a *least recently used* algorithm to select pages that can be copied out to disk and then freed for use by other processes. When the CPU has identified pages that it can appropriate, it *pages out* the old page images by copying the old data from those pages to a dedicated disk. The disk or disk partition that stores the page images is called the *swap disk, swap space,* or *swap area*. This paging activity requires CPUcycles as well as I/O operations.

Eventually, page images that have been copied to the swap disk must be brought back in for use by the processes that require them. If there are still too few free pages, more must be paged out to make room. As memory comes under increasing demand and paging activity increases, this activity can reach a point at which the CPU is almost fully occupied with paging activity. A system in this condition is said to be *thrashing*. When a computer is thrashing, all useful work comes to a halt.

To prevent thrashing, some operating systems use a coarser memory-management algorithm after paging activity crosses a certain threshold. This algorithm is called *swapping*. When the memory-management system resorts to swapping, it appropriates all pages that constitute an entire process image at once, rather than a page at a time.

Swapping frees up more memory with each operation. However, as swapping continues, every process that is swapped out must be read in again, dramatically increasing disk I/O to the swap device and the time required to switch between processes. Performance is then limited to the speed at which data can be transferred from the swap disk back into memory. Swapping is a symptom of a system that is severely overloaded, and throughput is impaired.

Many systems provide information about paging activity that includes the number of page scans performed, the number of pages sent out of memory (*paged out*), and the number of pages brought in from memory (*paged in)*:

- Paging out is the critical factor because the operating system pages out only when it cannot find pages that are free already.
- A high rate of page scans provides an early indicator that memory utilization is becoming a bottleneck.
- Pages for terminated processes are freed in place and simply reused, so paging-in activity does not provide an accurate reflection of the load on memory. A high rate of paging in can result from a high rate of process turnover with no significant performance impact.

You can use the following formula to calculate the expected paging delay for a given CPU utilization level and paging rate:

$$PD = (C/(1-U)) * R * T$$

PD

is the paging delay.

C

is the CPU service time for a transaction.

U

is the CPU utilization (expressed as a decimal).

R

is the paging-out rate.

T

is the service time for the swap device.

As paging increases, CPU utilization also increases, and these increases are compounded. If a paging rate of 10 per second accounts for 5 percent of CPUutilization, increasing the paging rate to 20 per second might increase CPU utilization by an additional 5 percent. Further increases in paging lead to even sharper increases in CPU utilization, until the expected service time for CPU requests becomes unacceptable.

Disk Utilization

Because each disk acts as a single resource, you can use the following basic formula to estimate the service time, which is described in detail in Resource Utilization:

$$S = P/(1-U)$$

However, because transfer rates vary among disks, most operating systems do not report disk utilization directly. Instead, they report the number of data transfers per second (in operating-system memory-page-size units.) To compare the load on disks with similar access times, simply compare the average number of transfers per second.

If you know the access time for a given disk, you can use the number of transfers per second that the operating system reports to calculate utilization for the disk. To do so, multiply the average number of transfers per second by the access time for the disk as listed by the disk manufacturer. Depending on how your data is laid out on the disk, your access times can vary from the rating of the manufacturer. To account for this variability, it is recommended that you add 20 percent to the access-time specification of the manufacturer.

The following example shows how to calculate the utilization for a disk with a 30-millisecond access time and an average of 10 transfer requests per second:

$$U = (A * 1.2) * X$$
$$= (.03 * 1.2) * 10$$
$$= .36$$

U

is the resource utilization (this time of a disk).

A

is the access time (in seconds) that the manufacturer lists.

X

is the number of transfers per second that your operating system reports.

You can use the utilization to estimate the processing time at the disk for a transaction that requires a given number of disk transfers. To calculate the processing time at the disk, multiply the number of disk transfers by the average access time. Include an extra 20 percent to account for access-time variability:

$$P = D (A * 1.2)$$

P

is the processing time at the disk.

D

is the number of disk transfers.

A

is the access time (in seconds) that the manufacturer lists.

For example, you can calculate the processing time for a transaction that requires 20 disk transfers from a 30-millisecond disk as follows:

$$P = 20 (.03 * 1.2)$$
$$= 20 * .036$$
$$= .72$$

Use the processing time and utilization values that you calculated to estimate the expected service time for I/O at the particular disk, as the following example shows:

$S = P/(1-U)$
$= .72 / (1 - .36)$
$= .72 / .64$
$= 1.13$

CHAPTER III:

IMPLEMENTATION DETAILS

CLIENT SIDE:

To extract the system information from the client side commands used are

- For CPU - mpstat
- For MEMORY- vmstat –s
- For DISK- vmstat –D
- For NETWORK- netstat –s

The data extracted using above commands is processed and stored in server database (Informix). To execute the above process periodically in the background "Crontab" is used.

SERVER SIDE:

- **TABLE DETAILS**

1. CPU (to store cpu resource info)

2. MEMORY

3. DISK

4. NETWORK

5. STATUS (to store the status info of all resource and warning msg)

CPU

dttime - datetime year to second
clientID - nchar(20)
%usr - float
%sys - float
%iowait - float
%idle - float

MEMORY

dttime - datetime year to second
clientID - nchar(20)
totalMem - int
usedMem - int
activeMem- int
inactiveMem- int
freeMem - int
bufferMem- int
swapMem - int

DISK

dttime - datetime year to second
clientID - nchar(20)
disks - int
reads - int
partition- int
writes - int

NETWORK

dttime - datetime year to second
clientID - nchar(20)
totalReceived- int
forwarded- int
inDiscarded- int
inDelivered- int
outRequests- int

- **FORMULAE TO ANALYZE RESOURCE UTILIZATION**

a. CPU

The parameters stored in database are %usr, %system, %iowait, %idle

CPU utilization = 1 - %idle (<20%= low, 20 to 80 % = normal , >80% high)

If %iowait > 60% imbalance n process mixture.

When the CPU is in high mode %usr and %sys gives the information about the cause of status.

b. Memory

If memory utilization = (used memory / total memory) * 100

(<20%= low, 20 to 80 % = normal , >80% high)

- ✓ If (active memory / used memory) > 0.8 then, Active mem consumption is high
- ✓ If (used memory /total memory) > 0.8 then, Used mem consumption is high
- ✓ If (buffer memory / used memory) > 0.8 then, Buffer mem consumption is high

c. Network

- ✓ if (Total Received / 2) < input discarded then Network Failure

  Similarly, if (forwarded / 2) > input delivered then Network Failure then status is "high".
- ✓ If Out request = 0 then, status = "low" Otherwise Status = "Normal".

d. Disks

- ✓ **Read Factor= (reads/Disks)*100**
- ✓ **Write Factor= (write/disk)*100**

- ✓ Read Factor > 80 or write Factor > 80 then status = "high"
- ✓ Read Factor and write Factor ranges between 20 to 80 status = "normal".
- ✓ Read Factor and write Factor ranges between 0 to 20 status = "low"

Program is written such that while executing if any resource's status is high then the warning message will alarmed to admin. The mentioned above program is scheduled with the help of windows task scheduler to run in background periodically.