

# A Comparative Investigation and Evaluation of Oracle9i and SQL server2000 with respect to Performance and Scalability

By Phathisile Sibanda

Supervisor: Mr John Ebden

Computer Science Department, Rhodes University

26 September 2005

**Abstract:** Performance and scalability are two omnipotent factors determining database availability and reliability. This is especially true for modern computer systems due to the inclusion of the Internet in Online transaction processing (OLTP) and E-commerce applications which use databases. This evolution gave birth to an enduring need for high throughput, good response time, excellent data consistency and concurrency control. This paper evaluates SQL server 2000 and Oracle 9i with respect to performance and scalability. Initial performance tests showed that at low to medium workload both database systems behaved in a similar fashion in terms of throughput and response time. At high load however Oracle 9i emerged victorious in both cases.

## **1. Introduction**

This project explores and evaluates performance and scalability capabilities of SQL server 2000 and Oracle 9i. This would be accomplished through the use of a combination of standard benchmarks (TPC-C) embedded in third party software like Quest Central 2005 for Oracle and SQL server 2000 and performance tuning tools like System Monitor and Analysis Manager included with these products. Furthermore, technical features influencing performance and scalability in these systems will be examined to support the benchmark results at the end of this paper.

## **2. Motivation**

There is a heating debate amongst Database Administrators as to which DBMS performs better under certain conditions. Two factors prompting this debate include the increasing number of open source databases flooding the

market and use of the Internet in databases. The first point has led to competition between commercial and open source database vendors while the second has resulted in a need for high performance database systems. This whole controversy has in the past led database vendors to employ numerous tricks in bid to improve the processing speed of their systems during benchmarking and thus falsely prove that their databases products are superior to other competitors. Some of their tricks include **using high-speed CPUs and clusters, caching up data rows and storing SQL execution plans** into RAM prior to testing thus bypassing disk I/O access and tremendously increasing performance (Burlison D, 2002). As a result the author seeks to provide an independent third voice that will aid DBA in decision making.

## **3. Methodology**

### **3.1 Performance and Scalability Testing Environment**

The performance testing environment comprises of three machines as depicted by Fig1.1 below. The client machine was used as the controller of the experiment for remotely managing, monitoring and controlling both servers. Processes carried out from this machine included remote load and data simulation, benchmark testing using TPC-C<sup>1</sup> benchmark embedded in Quest central software and data capturing for presentation and analysis. All three PCs are connected through a 100 Mb cable to the Rhodes Computer Science LAN. To minimise the effect of the network latency the author carried out

<sup>1</sup> TPC-C stands for Transaction Processing Performance Council Vision C (Transaction Performance Council, 2005)

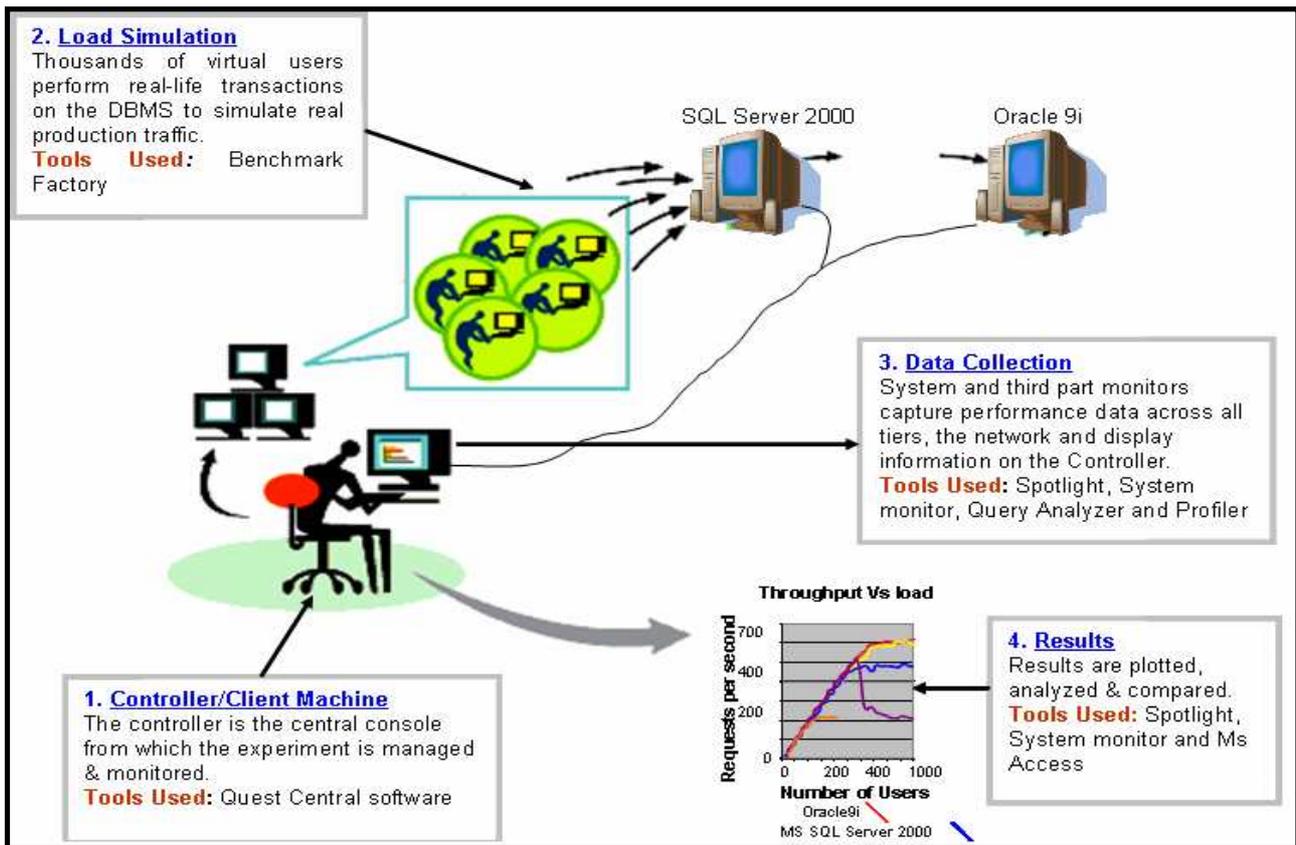


Fig 1.1 Performance Testing Environment

Experiments during periods of low network activity

NOTE also that remote monitoring and performance testing was not used for all scenarios in which case network sensitivity was not an issue.

The following provides definitions of two (functional and performance testing processes) processes used by the author to perform performance and scalability testing.

### 3.2 Functional/Baseline Test

#### a) Baseline Definition

A baseline is simply "a set of measurements that tell you how a server behaves while at rest", is basically a complete record of the system performance taken immediately after the server becomes operational, but before the server is put under any measurable load. The baseline is taken in a controlled environment, so that

external activities do not impact the baseline measurements. All significant features of the system performance are measured against predefined metrics<sup>2</sup> and later recorded and analyzed.

Functional testing goals include: Telling you all about the performance of a server under normal conditions for later comparison with performance test results and helping you build filters to catch "do not respond" situations before performance testing the servers. This allows you to respond to problems that need to be rectified before stressing your application begins.

### 3.3 Performance Tests

Performance testing comprised four processes:

- a) Load testing.

<sup>2</sup> These predefined matrixes are simply performance objectives released with the each product as certification importance. For example CPU utilisation in Microsoft server should not exceed 85%.

- b) Stress testing.
- c) Load and data simulation.
- d) Data capture and presentation.

### a) **Load Testing**

RPM Solutions Pty Ltd 2003 defines load testing as tests that determine the system behaviour at various workloads (low, intermediate and critical loads). Its main objective is to determine how system components react as the workload is gradually increased. Performance objectives/counters used for this project include response time, throughput rates, resource utilization (CPU, network latency, disk I/O subsystem and memory usage) and workload. These were chosen because response time and throughput objectives affect database performance from the user perspective while resource utilization and workload affected database scalability.

#### **The load Testing Process**

The number of users being simulated is gradually increased until one of the stated objectives is violated. For example, you may have stated as one of your objectives that the CPU usage of the server may not exceed 75%. When load testing this server, the number of users being simulated will be increased until the server's CPU usage is greater than 75%, assuming that no other objective has been violated before this occurs.

The load testing process consists of six steps (Microsoft Corporation Staff 2004):

- i. Identification of key scenarios: key scenarios are those scenarios that are critical for performance.
- ii. Identification of workload: the total application load is distributed among the scenarios identified in the first step.
- iii. Identification of metrics: This is where the various metrics that will be collected are identified.
- iv. Create test cases: The various test cases are identified and created.

- v. Simulate load: The various loads are simulated using a load testing tool, performance metrics are also collected.
- vi. Analyse results: Analyse the data collected during load testing.

### b) **Stress Testing**

Stress testing is basically a kind of performance testing where the server is subjected to very high loads well over its threshold, while denying it the resources required processing that load. A good example used for this research work the author removed the random think time<sup>3</sup> introduced in simulating simultaneous users from the load testing above. This meant that the servers had to deal with concurrent users who at the same time fired requests to the server. This exceedingly overloaded the server since no new resources were introduced to help the system. All users had to compete for system resources, for instance CPU cycles and memory, which heavily impacted system scalability and performance. Pushing the system way over its capacity in stress testing unearthed numerous bugs which the author monitor to gauge server breaking point which was used for scalability comparison in terms of how many users were able to do meaningful work before the system failed. Some of the system bugs monitored for this exercise included:

- ✓ *Synchronisation issues*
- ✓ *Race conditions*
- ✓ *Memory leaks*
- ✓ *Loss of data.*

#### **The stress Testing Process**

For stress testing to be feasible instead of having a list of possible scenarios you need to select one particular case to stress test. However it is possible to have a combination of scenarios for which you have performance objectives. The stress testing process requires you to obtain peak load capacity statistics from the load testing

---

<sup>3</sup> Random think time is the time spent by the user between two consecutive requests. This is the time when a user thinks of what to search for in the database (Remember we are simulating real production traffic).

process which serves as input. The actual process however is a six-step process as load testing with some minor differences.

The load testing process consists of six steps as used for this project (Microsoft Corporation Staff 2004):

- i. Identification of key scenarios: Identify scenarios that need to be stress tested.
- ii. Identification of workload: the total application load is distributed among the scenarios identified in the first step.
- iii. Identification of metrics: This is where the various metrics that will be collected are identified.
- iv. Create test cases: The various test cases are identified and created.

### c) **Load and Data Simulation**

Using load testing tools, Benchmark Factory and Quest Central software, test cases were created for the identified scenarios and run against the servers using the specified workload, collecting the required metrics for each test run. The user load was incrementally increased for each test until one of the stated objectives was violated, or until the systems reached its breaking point which such as a “server busy” message or abnormal CPU usage.

### d) **Data capture and Presentation**

A number of performance tools were used to measure and record data from performance tests. Microsoft Excel was used to present the collected information for analysis and interpretation. Graphs were drawn for:

- ✓ Throughput versus user load.
  - ✓ Response Time versus user load.
- (See Figure 1.2 and 1.3)

Where any bottlenecks were identified the database was tuned in order to remove them and the load tests were re-run and analysed again. Where performance did not achieve the stated goals, optimisations was implemented in order

to increase the database performance.

The following section example demonstrates the preceding processes. Load testing was used to compare the both servers with respect to throughput and response time.

## 4. **EXAMPLE: Performance Tests (Response time and Throughput) Load Testing**

### 1. Identified key scenario

For this project the author chose those user paths (scenarios) which either had a significant performance impact, or are resource intensive.

The scenario used for this example was:

- ✓ Querying the database for certain information.

### 2. Identified workload model

Performance characteristics or workload for the above listed scenario was defined. To do this, the following aspects were considered:

- ✓ The number of users for each scenario.
- ✓ The rate of requests received from simultaneous and concurrent users for a certain period of time.
- ✓ The pattern of requests: the rate of requests that individual functions of the servers experienced.

Using Quest Central the number of users was incrementally increased from 50<sup>4</sup> up to 1000 for this example.

### 3. Test Case

Number of Users: varied from 1 to 1000.

Test Duration: 3 hour.

Random Think Time<sup>5</sup>: Random think time varied between 1 and 5 was used.

### User Profile for Users Searching the DB

% of Concurrent User: 25

% of Simultaneous Users: 75

% Total Users: 100

---

<sup>4</sup> NOTE: The author with 50 users because number less than that produced insignificant measurements. However 50 users had reasonable response time and throughput.

<sup>5</sup> Check foot note 3 for a definition of “Random Think Time”

#### 4. Used Counters

Object	Performance Counter	Description
Processor	% Processor Time/_Total	% Processor Time is the percentage of elapsed time that the processor spends to execute a non-Idle thread.
Memory	Available Bytes	Available Bytes is the amount of physical memory, in bytes, immediately available for allocation to a process or for system use. It is equal to the sum of memory assigned to the standby (cached), free and zero page lists.
	Page Reads/sec	Page Reads/sec is the rate at which the disk was read to resolve hard page faults. It shows the number of reads operations, without regard to the number of pages retrieved in each operation.
Network Interface	Bytes Total/sec	Bytes Total/sec is the rate at which bytes are sent and received over each network adapter, including framing characters. Network Interface\Bytes Received/sec is a sum of Network Interface\Bytes.
Disk I/O system	% Disk Time	% Disk Time is the percentage of elapsed time that the selected disk drive was busy servicing read or writes requests.
	Avg. Disk Bytes/Transfer	Avg. Disk Bytes/Transfer is the average number of bytes transferred to or from the disk during write or read operations.

Table 1.2 Matrices<sup>6</sup> used

#### 5. Results and Analysis

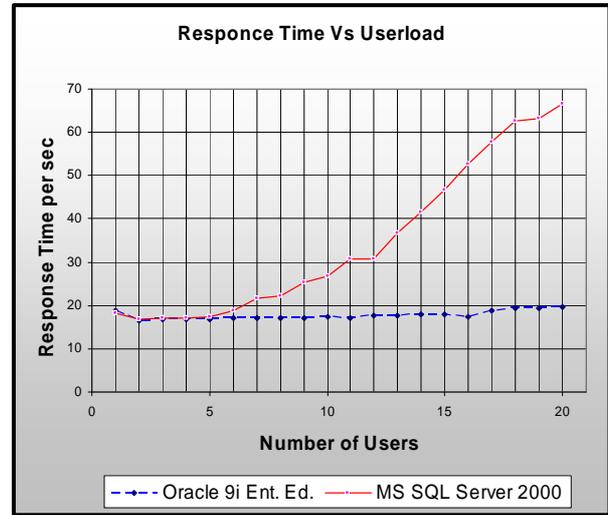


Fig 1.2 Chart shows results for Response time Vs User load.

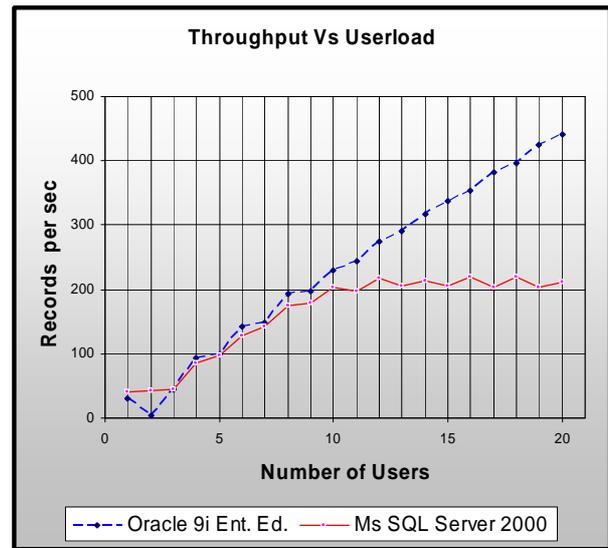


Fig 1.3 Chart shows results for Throughput Vs User load.

Figure 1.2 shows the response time measured for the first load testing process. The scenario used for this test involved searching the database for a certain number of records according to the criteria specified in the test script for each group of users. Two groups of users, simultaneous and concurrent, were used as described in 3 Test

<sup>6</sup> Matrices used are basically measuring units used for resources used.

Case above. Figure 1.3 shows throughput plotted against load. An explanation and analysis of these results is given in the succeeding section.

## **5. A Technical Analysis of the Load Testing Results**

In this section of the paper I explain in brief the reasons behind the results in the preceding section. At high workload Oracle 9i had lower response time and consequently high throughput returns. However with fewer users executing queries both systems produced similar results in terms of response time and throughput. According to SQL Server 2000 Resource kit, p. 57 the architectural implementation of the concurrency model is the root cause for these differences.

### **Concurrency model** **Multi-version read consistency**

In multi-user environments concurrency control ensures that data updates made by one user do not adversely affect those made by other users. Examples of these include:

- ✓ Dirty, or uncommitted, reads occur when a transaction can read changes made to the database that have not yet been committed.
- ✓ Non-repeatable reads occur when a transaction re-reads data it has previously read and finds that another committed transaction has modified or deleted the data.
- ✓ Phantom reads happen when a transaction executes a query twice returning a set of rows that satisfy a search condition, and finds that the second query can retrieve additional rows which were not returned by the first query, because other applications were able to insert rows that satisfy the condition.

Oracle's implementation of multi-version read consistency always provides consistent and accurate results. When an update occurs in a

transaction, the original data values are recorded in the database undo records. Oracle uses the current information in the undo records to construct a read-consistent view of a table's data, and to ensure that a version of the information, consistent at the beginning of the uncommitted transaction, can always be returned to any user. SQL Server 2000 on the contrary, have to choose a workaround to avoid concurrency problems, such as locking data to prevent it from changing while being read or preventing or queries from reading changed but uncommitted information.

SQL Server 2000 does not provide multi-version read consistency. Instead it requires applications to either use shared locks for read operations, with various levels of isolation, or to accept dirty reads. Shared locks prevent data that is read from being changed by concurrent transactions. Clearly, this implementation restricts the ability of the system to properly service concurrent requests in environments involving a mix of reads and writes, as explained in Microsoft documentation (SQL Server 2000 Resource kit, p. 57).

Concurrency control has a great bearing on the number of users who can access the database and do meaningful work. Oracle 9i excellent implementation of the multi-version read consistency model ensured that it scaled well as the load increase thus producing better throughput. SQL server uses shared lock which prevented multiple reads thus reducing performance in terms of response time as load increased. Other architectural aspects worthy noting includes indexing, partitioning and clustering capabilities that these systems provide.

## **6. Conclusion**

In view of the load testing results mentioned above it could be said, although not conclusive, that Oracle 9i is more scalable than SQL server 2000. The scenario used in this paper of searching the database provided a good environment which exhibited intensive resource usage in terms of I/O disk reads memory usage, network latency and CPU utilisation. Basically at low workload both database achieved similar graphs in terms of response time and throughput rates. Any evaluation of the concurrency model as implemented by the database engines was used to explain the case where Oracle 9i out played SQL Server 2000. A conclusive analysis however could be achieved by looking at the whole range of factors affecting performance in database systems such as the indexing capabilities. These are structures that provide faster access to data in database, partitioning and clustering capabilities which basically involve separating large database table and other structures to make them smaller for faster searching.

## **7. Future work**

The most possible extensions to this project as include:

- Applying the knowledge gained on from this investigation to Open Source databases.
- Using a different Operating System for the same experiment. For instance using Linux.

## **8. Reference**

Burleson D, August 8 2002."Oracle Guru"  
Database Benchmark Wars: What you need to know [Online]. Available from:  
[http://www.dba-oracle.com/art\\_db\\_benchmark.htm](http://www.dba-oracle.com/art_db_benchmark.htm) . Last Updated 05/08/2002, Access date 15/05/05.

J.D. Meier,Srinath Vasireddy, Ashish Babbar, and Alex Mackman, 2004. Improving .Net Application performance and scalability [Online]. Microsoft Corporation. Available from:  
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapter16.asp>  
Last Updated May 2004, Access Date 06/05/05.

Microsoft SQL Server documentation, 2005. Understanding Locking in SQL Server, [Online] Available from:  
[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac\\_8\\_con\\_7a\\_7xde.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/acdata/ac_8_con_7a_7xde.asp) . Access date 15/05/05

RPM Solutions Pty Ltd, 2004. Performance Tests [Online] Available from:  
[http://www.loadtest.com.au/types\\_of\\_tests/performance\\_tests.htm](http://www.loadtest.com.au/types_of_tests/performance_tests.htm) .Last Updated August 04, 2004 , Access date 06/05/05.

Transaction processing Performance Council, 2005. Top Ten TPC-C by Performance [Online]. Available from:  
[http://www.tpc.org/tpcc/results/tpcc\\_perf\\_result.s.asp](http://www.tpc.org/tpcc/results/tpcc_perf_result.s.asp) .Last Updated 21-Sept-2005 7:52 PM Access date 14/05/05.