

HEAD TO HEAD: A COMPARISON BETWEEN WINDOWS AND LINUX AS AN ORACLE DATABASE PLATFORM

The logo for CSC, consisting of the letters 'CSC' in white on a red background.

Steve Ries

CSC North American Public
Sector

wries@csc.com

CSC Papers

2008

ABSTRACT

Keywords: Oracle, Microsoft Windows, Linux, Quantified Comparison, Database, Performance

In a recent survey of the industry leading relational database management systems, market research firm Evans Data Corp. found that Oracle leads all competitors in every one of the 13 separate categories surveyed, including performance, security and scalability. However, although many database administrators and IT managers consider Oracle the clear choice for their database software, it is not always as evident as to which operating system platform makes the best choice for Oracle. This paper is an attempt to quantify the relative merits of the Windows and Linux operating systems as a underlying platform for Oracle and to experimentally determine, whether either of the two demonstrates superiority for this task. In pursuit of this goal, several experiments are detailed and recorded, in which key criteria are compared between the two operating systems.

INTRODUCTION

Although Oracle Corporation has long been known for industry leading innovations in the area of database technology, in 1999, they made one innovation that left the IT industry scratching their collective heads. In August of that year, Oracle released their top selling relational database management system (RDBMS) for the then-little-known Linux operating system. The move should not have surprised many; since several years earlier, Oracle startled the industry by offering their RDBMS for the Windows operating system, which was scarcely even considered a server-level platform at the time. With these two moves, Oracle had staked out its claim that the days of large monolithic database systems housed on mainframe platforms were gone. History has fallen on the side of Oracle's claim, and as a result, the company now holds a substantial portion of the very lucrative small database server market. While both operating systems have substantially elevated their positions in the enterprise in last few years, it is Linux that has captured the collective imagination of the IT world, with its promise of a low-cost (or even license free) OS that is built on the rock solid, time tested foundation of the UNIX operating system. Yet it is Windows that still maintains a substantial hold on small database market share. The purpose of this paper is to quantitatively examine whether a convincing case can be made establishing either Linux or Windows as a superior platform for the Oracle RDBMS. For the sake of this examination, we're not considering other UNIX variants such as Solaris and AIX due to hardware differences – we will only examine hardware infrastructures based on the x86 chip model. Because of Oracle's large share of the database market, it is hoped that the information herein would equip technology planners to make informed decisions about their future database architectures.

It's important that the question of which operating system to use for Oracle is not viewed as a matter of personal preference or in terms of generalities such as "Linux is too hard to understand" or "Windows is slow", etc. The point of this document is to examine whether there are quantifiable, compelling reasons for using one OS over the other. It's also important to understand that this study focuses on only the *capabilities of the operating systems for running Oracle*. This isn't a promotion or indictment of a particular OS, rather an attempt to show which one is most suited for running high-performance Oracle databases. As such, these results may or may not apply to email servers, web servers, etc.

This paper is divided four sections; first, general reasons for running a database environment on a single platform, second, a list of criteria for determining the suitability of an operating system to run Oracle, and third, the results of a series of side-by-side performance tests comparing Windows and Linux. Finally, the study will end with a set of conclusions that can be drawn from the available data.

WHY A SINGLE PLATFORM?

To properly frame the discussion, consider the question "Why should I consider a single platform for Oracle at all? If there are relative merits for both Linux and Windows, why not run both?" In today's IT shops, the reality of data centers and server farms that are heterogeneous as it pertains to operating system is evident. However, even though companies use different operating systems for different

purposes, there are significant advantages to standardizing a company's Oracle presence on a single operating system.

The first of these is efficiency in administration. This can take many forms, but for a factor of chief concern, consider security. In an age where security consistently ranks number one in the concerns of IT shops, company database strategies should take into account the need for efficiency in all aspects of security. Although database software patches and updates were once considered a luxury, they have recently come to the forefront of a database administrator's concerns. Two years ago, Oracle Corporation began the CPU (Critical Patch Update) program - it's policy of issuing essential quarterly security patches. When a shop neglects currency on Oracle's CPUs, it does so at its own peril. But consider the task set before a DBA to keep the company's database environment secure. Let's use an example from a small to medium sized database shop with 25 servers that host 35 databases. Let's say that 5 of the servers are large legacy UNIX machines, 10 are Linux servers and 10 are Windows. Since each critical patch is different for each operating system, the DBA must contend with installing, testing and apply three different patches every three months. Now let's go further and say that each of those three separate operating systems hosts databases from two different major versions of Oracle (for example, at the time of this writing, Oracle 9i and Oracle 10g). Since every security patch is also delineated by Oracle software version, we're now up to six separate installation and testing iterations. If we were to differentiate based on minor releases (another requirement for security patches), we would fragment the DBA's efforts exponentially – and it has to be done every three months. Clearly, standardization on one operating system would at least help to alleviate a situation like this.

Another benefit in standardizing operating systems is to allow for the portability of database scripting. While SQL scripting itself is completely portable across Oracle systems on differing OS's, the kinds of scripts that actually drive database work, such as job scripting, are not. There are monumental differences between Windows batch scripting and Linux shell scripting, both in syntax and capabilities. The use of scripting for automated deployments or administration is made much more difficult by these differences. Consider the example of scripts to do backup and recovery. Uniform backup scripting can be done with Oracle's Recovery Manager (RMAN), however the control scripts that actually call RMAN and control what happens as a result of the backup are more operating system dependent. This means that the goal of rolling out truly standardized backups, while not impossible, is made more difficult in an environment with multiple operating systems.

Finally, another benefit of standardized operating systems is the relevance of server information from environment to environment. For example, suppose we have an application running against an Oracle development database on Windows. We tune the system, the database parameters and even the SQL code to provide the best performance possible. Then, we move it to a TEST database running on Linux. How much of the work done to tune performance on the Windows environment will transfer over to the Linux environment? Almost none. Deep performance tuning differs too much from OS to OS to be of any real use. Using a

DEVELOPMENT → TEST → PRODUCTION progression model on a single operating system lends itself most readily to accurate performance tuning.

WHAT CRITERIA?

If we establish that it is preferable to operate on a database environment with a single, standardized operating system, what would be the criteria by which we could determine which OS should be used? In order for one to distinguish itself from another when running Oracle, that OS must excel at the following

- *Performance* – Oracle on the OS in question should service database activity in a more timely fashion than other operating systems.
- *Efficiency* – The above performance should be accomplished by utilizing resources more efficiently. In essence, it should run faster while using fewer resources.
- *Security* – The OS should provide a secure platform on which Oracle runs. If the OS is insecure enough to allow penetration onto the server, there's little that Oracle can do to stop penetration of the database.
- *Stability* – The OS should provide a stable platform on which Oracle runs. If the server continually crashes, the database won't be available either.
- *Administration* – The OS should provide an adequate toolset of commands in order to aid in the administration of the database.

Put bluntly, Linux excels in these areas, while Windows flounders on many. Consider the following.

- *Efficiency* – Because of its roots in standard UNIX, Linux was built as a *multi-process* operating system. This means that all system operations, including Oracle, get separate processes in order to complete a unit of work. Windows, however, was created as a *multi-threaded* operating system. This means that a single process is often split into multiple threads, or virtual processes. This is fine, even preferable, for Windows' origins on the desktop. However, since Microsoft chose not to significantly alter this threaded behavior from their standard desktop OS, Windows is not practical as a large, server operating system. As a result, many operations, especially Oracle, suffer from the sharing of a single process to do work. The result is a highly inefficient way to handle Oracle requests. It is a staggering fact to most DBAs, but Oracle on Windows is forced to run *all* of its background processes through a *single oracle.exe process*. These background processes are at the core of how Oracle works. If they must share time between each other, the result will often be slower, especially as load increases.
- *Security* – As noted above, if the OS itself is not secure, then there is nothing really securing the database. An OS that is penetrable presents a database that is penetrable. Windows security problems are well known in the IT industry, forcing constant application of patches, virus scanners, etc in order to even function. In the case of Linux, however, viruses are nearly unknown. Generally, the only security problems that are present with Linux are those that pertain to

various tools that Linux includes, such as FTP, which are oftentimes not used. The cause for this difference is debatable; however, whether you believe that Windows is truly written with more vulnerabilities or whether you think that these holes exist only because Microsoft is a bigger target, the fact still remains – these security holes are common, well publicized and show no sign of improving anytime soon.

- *Stability* – Although Microsoft has made significant strides in the area of stability in recent years, Linux is still generally considered to be significantly more stable strictly from the perspective of server crashes, commonly recording 300+ days of uptime without a crash. This kind of uptime puts Linux machines in the same availability class as most of the major UNIX variants at a significantly lower cost. While it is arguable that the term “stability” includes more than base uptime statistics, if your server is unavailable, so is your data. It is also noteworthy on the subject of stability that Linux is what is known as the “primary 32-bit platform” for Oracle. That means that the Oracle software is developed on Linux first, tested on Linux first and released on Linux first. That kind of commitment early in the development process adds much to the stability of Oracle on Linux as a platform.
- *Administration* – Although Microsoft markets the Windows line of products giving the impression that they are easier to use, nothing could be further from the truth when it comes to Oracle database administration. Consider these examples.
 - One of the most frustrating things for an Oracle DBA is the complete absence of the basic OS tools needed to operate Oracle on Windows. A good example is the lack of a simple utility like the Linux “tail” command. This command allows the DBA to interactively watch log files, such as the alert log, while an operation is occurring, such as the startup of a database. There is no such command in Windows. Instead, you double-click the file, then close the file. Double-click it again, then close it again. Double-click, close. Repeat until you see the results.
 - Another example of this shortcoming are scripting languages like Perl. Many organizations have made a strong commitment to using language like Perl in their job level database scripting. However, Microsoft does not share this commitment. As such, a shop using Windows to support Oracle must contend with doing a secondary installation of a Windows-specific scripting language implementation, further hampering the efficiency of administration.
 - As another example, consider that recently fewer and fewer FTP (file transfer protocol) servers are being allowed in IT shops because of their inherent insecurity, while ssh (Secure Shell) and sftp (Secure FTP) are on the rise. Yet Windows has no ssh client of any kind included with its operating system; instead, one must be purchased and installed to do the most basic tasks. Linux includes secure functionalities like this as a part of the base operating system.
 - Because Windows was not originally designed as a server operating system, it has a closed facility to access shared memory, requiring the use of “services”. Since Oracle relies heavily on shared memory, it must use these services to

funnel all database processing. As a result, when you start a database on Windows, you don't just start the database; you must start the services as well. This complicates database diagnosis procedures. If the database is down, is because of the database or the services? Which is really down?

- While some might pose the argument that Windows is easier to administer because more people are familiar with the Windows interface, that fact is not necessarily a benefit. The question isn't how many people are comfortable with the Windows interface, but rather how many DBAs choose to use the Windows interface. Certainly there are some, but by and large, DBAs overwhelmingly choose a standard UNIX command line interface for dealing with Oracle, since Oracle is so much more common on UNIX than Windows.

The subject of performance will be covered extensively in the next section - the results of a series of comparisons between the two operating systems. These tests were conducted to be as fair and even-handed as possible. The server hardware used was identical in every way. The database versions were the same. The amount of memory allocated to Oracle was the same. In each case, a basic database operation was run against both and the results compared. An explanation of these tests, the conditions used and the results is as follows.

THE CONDITIONS

In order for this experiment to be a true test of the capabilities of Oracle on each operating system, the conditions under which the test is conducted need to be as controlled as possible. Thus, two identical servers were used as hosts, identical versions of Oracle were used, etc. The specific conditions are outlined as follows.

- Two identical Dell servers were used as hosts.
 - Dell 6800 servers
 - 16 GB of RAM
 - Four 3166 MHz processors with hyper threading
- Windows host
 - operating system – Windows 2003 Enterprise Edition
 - patch level – SP1 build 3790
- Linux host
 - operating system - Red Hat Enterprise Linux AS release 3 (Update 4)
 - kernel - 2.4.21-27.Elsmg
- All tests run locally with no network interaction, so as to eliminate random network latency from the equation.
- No other users were connected during testing in order to eliminate the potential for interference due to lock contention.
- Both operating systems were 32-bit.

- Two identical installations of Oracle were used - version 10.2.0.1.0.
- The same number, size and structure of redo logfiles were used.
- The same sized System Global Area (Oracle's shared memory) – 1200M – was used

```
SQL> show sga
```

```
Total System Global Area 1258291200 bytes
```

- Two identical copies of the same tables were used.
 - Table#1 segment size = 3072M (approx. 3 GB)
 - 67,009 rows of data, including a BLOB field
 - Average row size = 48,072 bytes, including BLOBs
 - Used as a large table for benchmarking I/O speed
 - Table#2 segment size = 256 M.
 - 521,774 rows of data
 - Average row size = 426 bytes
 - Used as a small table for benchmarking computational speed
- Two identical users were created on each database with identical permissions. No resource limits were used.
- The initialization parameter files of both databases were as identical as possible. The only differences were related to names and paths.
- No partitioning of tables was used.
- No performance enhancements or special tuning was done on either machine prior to testing.
- The Linux host ran with asynchronous I/O turned on, since that is a feature of the Linux kernel and should be considered when comparing the speed of both operating systems. The Windows operating system is not capable of asynchronous I/O.

THE TESTS

In order to properly test the functionality of Oracle, we have to create a balance between doing “realistic” work, while generating enough concentrated database usage to adequately gather statistics. To that end, several tests were constructed, each focusing on different areas of database operations. The metrics recorded for most of these tests center around time – i.e. how long a particular operation took. Then, a measurement of disk space is recorded, whether it's the size of the export file, or size of the table segment. From these two metrics, a basis of comparison, the “throughput” is constructed. The throughput is defined to be the amount of

space operated on divided by the time that the operation took. It is this “throughput” metric that is recorded on the graphs in the “Results” section.

Export/import test – In the first test, a generalized test for export and import speeds was run. First, an export of table #1 is run in conventional mode, followed by a truncate and an import done likewise in conventional mode. Next, the same table is exported and re-imported using direct (no SQL buffer) mode.

RMAN backup test – Next, a multichannel tablespace level backup of the tablespaces was run that contains both test tables. A full backup was not done due to the fact that the databases contained differing amounts of unused data in other tablespaces. To keep the conditions equivalent, only the tablespace that was common to both databases was backed up.

Computational Benchmark test – For this test, a stored procedure was built that reads data from Table #2, does different forms of data transformation on it using standard Oracle functions, and then writes the results back into another table. This test operates on the smaller table, and thus does less I/O, but must do a large amount of computation, since nine different data transformations are done for each row in the table. The transformations are as follows

1. Number to character
2. Substring of a character string to number
3. Date to character
4. Length of a character string
5. Addition of two numbers
6. Upper/lower case transformation of a character string
7. Search within a character string for a different string
8. Application of the Oracle hashing algorithm to a character string
9. Application of the Oracle hashing algorithm to a number

In addition to the throughput speeds of this test, the CPU usage of both servers was tracked and averaged during the execution of the test. This gives an indication of “how hard” each CPU is working in order to service the request.

“Create table as select” test – In this test, Table #1 (the BLOB table) was copied using the Oracle command “create table as select” or CTAS. The CTAS command reads data from the source table and inserts it into a destination table. In that respect, it is essentially an insert using the direct path method. It simulates both high read and high write activity. In addition to running a default level CTAS, subsequent tests were run using Oracle’s parallelism feature. Parallelism allows multiple processes or threads to divide and concurrently process separate units of work, then re-combine the results. Generally, parallelism is expected to improve the performance of a SQL operation, although most statements have a “critical mass” where adding more parallel processes does not help, and can possibly hurt,

performance. Thus, six CTAS tests were run, ranging from parallelism of 1 to parallelism of 32.

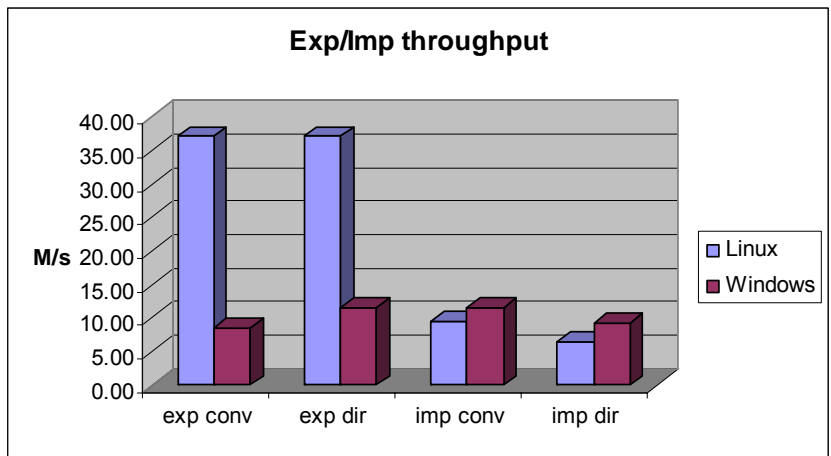
“Insert/select” test – In the “insert/select” test, data from table #2 is selected and read into a buffer, then written out to a pre-created copy of the table. It is slightly different from the CTAS test in that the table is pre-created and the buffering scheme used by Oracle is slightly different. It is another good test of I/O performance.

“Join by Primary Key” test – With the “Join by PK” test, the focus shifts somewhat away from I/O performance and moves toward testing SQL performance and computational speed. The Join by PK test simulates running a join of two tables using the primary key and then writing the resulting join to a table. This test is also run with varying degrees of parallelism, ranging from 1 to 32.

THE RESULTS

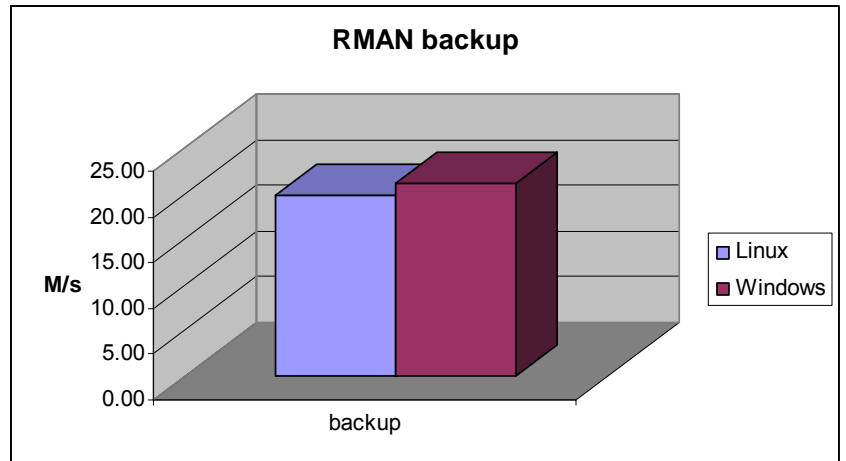
Graphs of the results of each test are shown below, along with a short discussion of the results.

Export/Import test



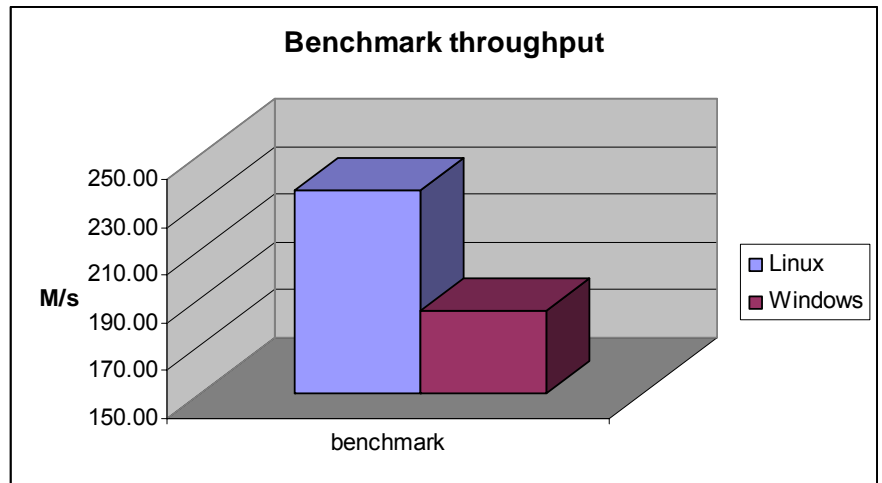
In these four iterations of the export/import test, results under Linux are shown in blue, while the results under Windows are shown in maroon. Again, the graph measures throughput, which is the amount of space operated on divided by the time consumed by the operation. It is expected that imports are much slower than exports, usually taking about three times longer. While the import results, both conventional and direct, show a slightly higher throughput for Windows (about 20%), the exports show a decidedly higher throughput for Linux, sometimes yielding more than 300% faster throughput than Windows.

RMAN Backup test



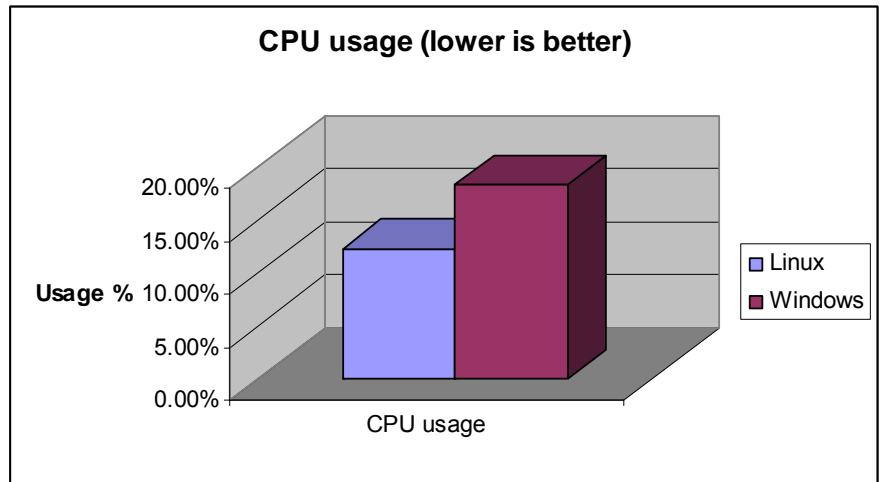
The backup test is nearly equal, with Windows showing a roughly 7% higher throughput for RMAN backup operations. This particular test was run several times, each time yielding the same results.

Computational Benchmark test



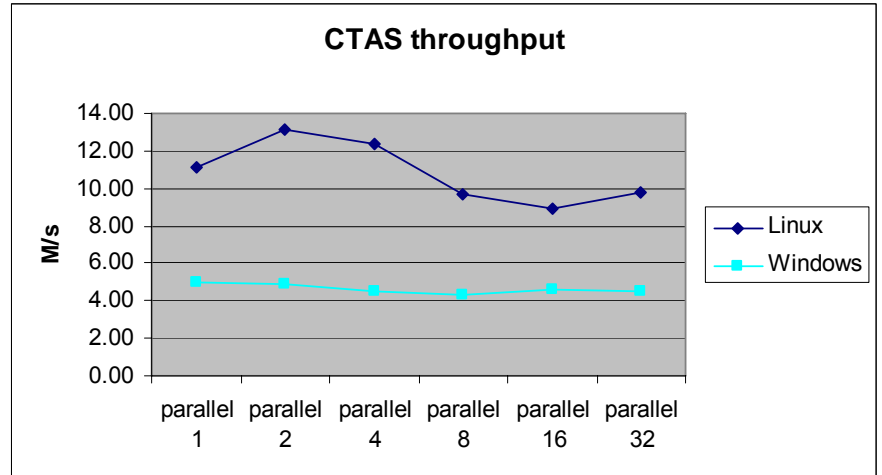
The Benchmark test is interesting because of its reliance on CPU in order to accomplish the task. As noted in the “Tests” section, nine different computational transformations are done for every one of the 521,774 table rows in the Benchmark test. It’s also the only test that uses PL/SQL and Oracle SQL functions. This is the first test where the results show a conclusive throughput difference between the two operating systems. The Linux results are nearly 30% faster than those on Windows (note that the graph’s base “zero” value begins at 150 M/s). As a further test, the average CPU usage statistics during the Benchmark test were also taken.

Computational Benchmark test (secondary test)



The significance of this secondary test is that it shows that not only is the Oracle on Linux database faster than the one on Windows in terms of raw computational speed, it manages to be faster while utilizing the CPU in a more efficient manner. In this case, the data collected shows that Linux processed the statements 30% faster than Windows while using 6% fewer CPU cycles.

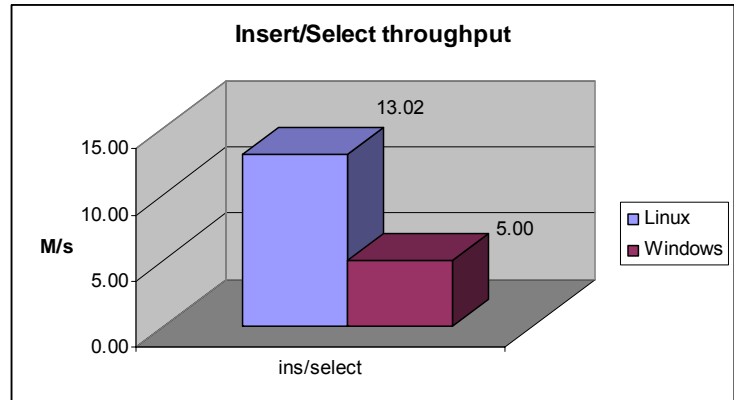
“Create table as select” test



The CTAS test is the first of the tests where Linux begins to run away with the results. In this test, we’re running various iterations of the same operation, each time with a different level of parallelism. In each case, the operation running on Linux is between 100 and 170% faster than its Windows counterpart – two to almost three times faster. However, the most striking thing about these results is the fact that the Windows results aren’t even consistent with expected behavior. With the addition of parallelism, we should never see a performance graph that is flat. Some degrees of parallelism should help the statement’s performance, and others should hurt it. With the Linux results, we see that, for this particular operation, a parallel degree of 2 or 4 will yield faster results than a degree of 16.

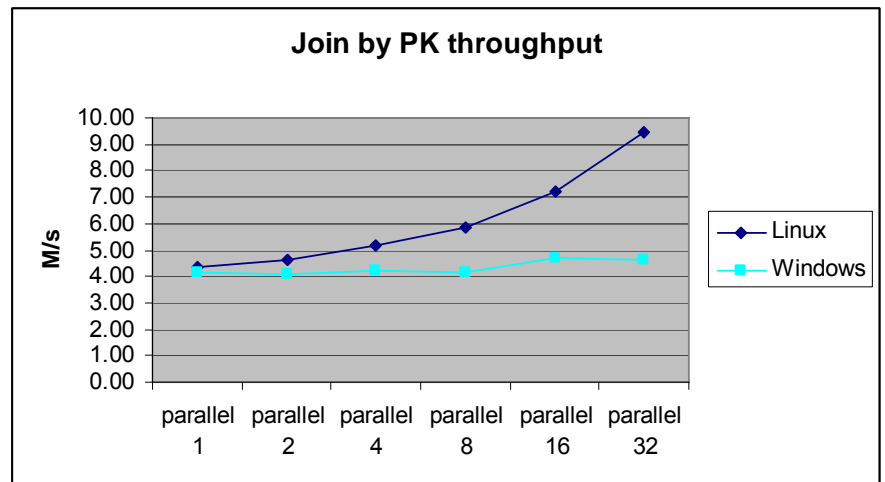
This is expected behavior according to the “critical mass” rule mentioned earlier. However, with the Windows results, not only is it slower, but parallelism has no little measurable effect at all on the outcome.

“Insert/select” test



The insert/select test is another I/O intensive test designed to emphasize the ability of each platform to move large amounts of data. The results show another dramatic difference in throughput results between Linux and Windows. The Linux/Oracle database for this test was 160% faster than the one on Windows.

“Join by Primary Key” test



The “Join by Primary Key” test is probably the most important of all the tests run in this experiment for two reasons. First, it probably simulates a “real world” environment better than any of the tests. It’s a combination of I/O throughput and computational speed that is typical of how most databases are used. It requires disk, memory and CPU resources for reading, writing and sorting. Secondly, it shows that, while both Linux and Windows configurations started at the same performance level, *Windows performance again does not improve with the addition of parallel processes, while Linux performance achieves a scalability that approaches linearity.* Linux again arrives at a throughput level over 100% greater

than Windows, twice as fast, and may have continued even higher if given more parallel processes. The results of this test dramatically demonstrate the difference between a multi-process operating system such as Linux or UNIX and a multi-threaded operating system like Windows. Multi-threaded operating systems are fine for desktop systems where single threaded operations are common. However, for a server operating system, these tests show that simply putting multiple threads through one process doesn't get the job done.

CONCLUSIONS

People often say that the only differences between Windows and Linux come down to preference. Some even refer to the comparison as a "religious" argument. The only true difference in the performance of production systems, they say, comes only from the hardware. The results of our Oracle specific testing show otherwise. Using completely identical hardware, one operating system stood out from the other as a foundation for Oracle. While Linux did not completely dominate every test, it should be obvious that the vast majority of the results strongly favors Oracle on Linux.

Consider the data from the tests:

- SQL transformation statements that ran 30% faster with 6% less CPU
- High I/O insert/selects that ran 160% faster
- Statements that utilize parallelism with almost linear scalability resulting in twice the performance
- Exports that ran 300% faster

The basic idea of the results is that CPU intensive operations were decidedly faster on Linux and high I/O operations were *tremendously* faster. The results should not be surprising, considering our previous arguments regarding the multi-threaded nature of the Windows operating system. Linux was built from the design and philosophy of UNIX. UNIX is not a desktop operating system – it was designed at its core to run as a server. Some of the earliest ports of Oracle modeled many of its features after UNIX. Further, Linux has enterprise-level features such as multiprocessing, process-accessible shared memory and asynchronous I/O. Windows, however, is an immensely successful desktop operating system that has attempted to evolve into a server operating system. Whether or not it will someday evolve to be a serious platform for database servers has yet to be determined, but, from our results, it would appear that day is not yet here.

Disclaimer

The information, views and opinions expressed in this paper constitute solely the author's views and opinions and do not represent in any way CSC's official corporate views and opinions. The author has made every attempt to ensure that the information contained in this paper has been obtained from reliable sources. CSC is not responsible for any errors or omissions or for the results obtained from the use of this information. All information in this paper is provided "as is," with no guarantee by CSC of completeness, accuracy, timeliness or the results obtained from the use of this information, and without warranty of any kind, express or implied, including but not limited to warranties of performance, merchantability and fitness for a particular purpose.

In no event will CSC, its related partnerships or corporations, or the partners, agents or employees thereof be liable to you or anyone else for any decision made or action taken in reliance on the information in this paper or for any consequential, special or similar damages, even if advised of the possibility of such damages.

© Copyright 2008 Computer Sciences Corporation.