Comparing IDS 10.0 and Oracle 10g

by Jacques Roy

How do we select a database that will give us a business advantage over our competitors, a database that provides the best possible return on investment?

This paper focuses on the technical characteristics that make a good OLTP database system and why IBM-Informix Dynamic Server version 10.0 is better than Oracle 10g release 2. One of the major challenges is to use technical information to do the comparison without getting bogged down in details. I try to avoid this pitfall by using technical details to analyze higher level requirements. However, I need to bring in and explain some related concepts relevant to our discussion as you'll see below.

OLTP Requirements

An On-Line Transaction Processing system runs the day-to-day operations of an enterprise. For large enterprises, it must support a large number of connections such things as point of sales and users. This implies a potential large number of transactions.

Many of these systems work in real time: a down or unavailable system can represent lost revenue.
We can define the following requirements for OLTP systems:

Performance: The database system must optimize the use of resources to get the most out of the machine it runs on.

Scalability: the system must be able to handle a large number of requests concurrently and an increasing amount of data. The amount of work that can be done must be proportional to the resources available (CPU, memory, and so on). This can be considered part of the performance requirements.

Reliability: The system must be dependable and stay up at all time.

Availability: The system must be available 24x7. This means that it must minimize any planned or unplanned outages.

Manageability: The system must be easy to manage. The amount of personnel resources you spend maintaining the database system directly impacts the total cost of ownership.

Application Development: How easy is it to develop applications for your database? How standard is it? How can you adapt your database to your design instead of compromising your design to fit the database? These are some of the questions we need to answer.

Performance and Scalability

The performance section of this paper is the longest since it requires important background information to understand what impacts performance at the hardware, operating system, and database server levels. Of course, the design of the application that dictates what must be done has a big impact on performance but software design is outside the scope of this paper.

Performance Background

Before we can discuss database performance, let's look at what impacts performance: The characteristics of the components of a computer system.

We all have a tendency to forget that computer performance is more than CPU mega/giga- hertz. For a better understanding of what influences system performance, we have to look at the major components such as CPU, memory, system bus, disk controller, disk and network.

Let's take a current system such as the IBM p570 that can scale to 16 processors, 512GB of memory, and 79.2TB of disk storage:

CPU: The CPU speed can currently go up to 1.9GHz. Each core includes 2 CPUs with 32KB of level 1 cache each, 1.9MB of level 2 (L2) cache per core, and 36MB of level 3 (L3) cache. The L2 to L3 maximum bandwidth is 48GB/sec.

Memory: The p570 provides a choice of memory modules. Some CPU cycles have to be used to setup hardware memory addressing which in fact dramatically cuts down on the peak performance of 10.6 GB/sec for the DDR1 modules and 24.98 GB/sec for the DDR2 modules.

PCI-X bus: For any extensions such as additional disk controllers, the system must go through this 133MHz 64-bit bus that provides a throughput of 1 GB/sec.

Disks: Disk drive speed is impacted by the time spent positioning the disk heads over the desire cylinder (seek time) and waiting for the data to appear under the heads (rotation latency). The fastest drive available for the p570 rotates at 15000 rotations per minute (RPM). The rotation latency is half a rotation. This translates into a 0.002 second delay. During that time, the CPU mentioned above can execute 3.8 million instructions.

Network: The p570 supports network controllers that can operate at 10/100/1000 Megabits per second. We find 8 bits in one byte. We can then translate the maximum transfer rate of 1000 megabits per second to around 100 MB/sec. Then we have to consider the network packets overhead and potential collisions. This further reduces the effective bandwidth we can expect from this device.

This brief overview of the characteristics of computer components tells us that performance is made up of multiple things. We must optimize the use of the CPU by limiting the delays introduced access to memory. The hardware helps in this with the 3 levels of cache included in the Power5 processor. Even disk drives have caches to minimize the impact of seek time and rotation latency. Having a CPU wait for disk access is catastrophic in terms of performance.

Operating systems have been used to optimize the use of the hardware for a long time. It is interesting to see that a lot of the optimizations that we find in operating systems also apply to database servers.

Operating Systems

Operating systems (OSs) serve several purposes such as providing a high-level interface to the hardware, share resources (CPU, disk, memory, etc) between multiple users, protect users from each others, and optimize the use of the resources.

For a user program to run on a computer, it needs to be loaded into memory so the instructions and data can be accessed by the CPU. A program does not use the physical memory directly. The OS uses the concept of virtual memory that it then maps to physical memory pages for execution. Programs access other resources, such as disks and networks, through system calls.

OSs schedule user programs for execution by allocating time slices to each program in a prioritized round-robin manner. No single program can monopolize the machine since the OS can regain control and schedule another program.

In terms of performance optimization, Lets look at three OS features: file systems, threads, and processor affinity.

A file system optimizes access to disk by doing block I/O and buffering the result in memory. The block I/O access helps in limiting the price paid in each disk access: cylinder to cylinder seek time and rotation latency. For example, the original Unix system used a block size of 1KB. Later, The BSD group came out with the fast file system (FFS). One of the major characteristic of the FFS was the use of a minimum of 4KB block size (configurable). It also use a concept of cylinder group to limit the seek time by keeping administrative information in each cylinder instead of keeping all the administrative information for a file system in one location.

This confirms what I said before. You don't want the CPU to wait for disk access. File systems go one step further by caching the disk information into memory. This accomplished two main things. It allow the operating system to write the information back to disk asynchronously while the CPU is busy with other things and it reduce disk I/Os by making the information available without having to go back to the disks. This optimization technique is also used by database servers.

Modern OSs include the concept of threads. A thread is a lightweight object that runs in the context of a process. The use of threads can simplify programming and optimize the use of the system:

> "*In Solaris, creating a process is about 30 times slower than creating a thread, synchronization variables are about 10 times slower, and context switching about 5 times slower.*"
> Threads Primer, page 21

Large applications such as database servers can improve performance and scalability by using threads. It also have an impact in other areas as we will see later. IDS is architected using a threading model.

Over the years, computers have gone from 1 CPU to, now, up to 64 or even 128 CPUs. To optimize the use of CPUs, the concept of processor affinity was developed. Processor affinity is the re-scheduling of a program to the same CPU it ran last. It started as a feature that was optionally called by programs and eventually made it to the scheduling algorithm of the OS. This feature increases the probability that the instructions and data of a program will still be in the CPU cache when it is re-scheduled. This demonstrates the importance of optimizing access to memory to obtain the full performance of the CPUs.

We see that OSs optimize access to disk by buffering data in memory and optimizes access to memory by buffering data in the CPU cache. It also provides additional

facilities to minimize the overhead on the system through the use of threads. Overall, these features contribute in providing high levels of performance and scalability.

Database Server Architecture

A database server runs on a specific hardware architecture. These architectures vary between uni-processors, symmetrical multi-processors (SMP), massively parallel processors (MPP), and so on.

IDS runs on uni-processor and SMP machines. Oracle, through the use of RAC (Real Application Cluster) can also run on MPP type machines. We will see later why this is not a disadvantage to IDS.

The general architecture of all major database servers have a lot of similarities. For example, it always includes the use of:

- Shared memory to buffer data and share information between processes
- Processes to handle physical and logical logging
- Asynchronous I/O and read-ahead I/O

Instead of reviewing the details of the entire architecture, we will focus on some major differences between IDS 10.0 and Oracle 10g. We will look at these architectures as implemented on Unix-type systems as to not confuse some issues.

IDS Architecture

The IDS 10.0 architecture is based on a set of processes called virtual processors (VPs). There are multiple types of virtual processors to handle specific tasks. Each VP is multi-threaded, it can handle multiple tasks and adjust dynamically to the demand of the system by adding or removing threads as it sees fit. The result is that a handful of Unix processes are required to handle any load on the system. This threading model has been designed specifically to optimize database operations. It has been improved over the years to provide maximum performance.

The IDS threading architecture extend to the handling of user connections and the parallelism of SQL queries or other database operations. Each thread gets a time slice of execution on a CPU VP and then gets rescheduled. This way, no single thread will monopolize the use of resources. This insures a smooth operation even with a large number of users and a mix of queries.

Since creation, scheduling and synchronization of thread operations are more efficient than similar operations on processes, IDS provides the optimal foundation for performance and scalability. We'll also see that it also impacts ease of administration.

Oracle Architecture

The Oracle 10g architecture is virtually the same as we saw in previous Oracle releases. In contrast with the IDS architecture, Oracle 10g is process based. A user can connect to Oracle either through the use of a dedicated process or by using shared server processes. Shared server processes provide some level of optimization:

"*In general, it is better to be connected through a dispatcher and a shared server process… A shared server process can be more efficient because it keeps the number of processes required for the running instance low*."
Oracle 10g Administrator's Guide, page 4-1.

Using shared server processes, Oracle adds and removes processes as needed based on some configuration parameters. Query scheduling works as follows:

1. A dispatcher receives a request
2. The dispatcher puts the request on a request queue
3. The next available shared process takes a request form the queue and execute to completion
4. The shared process puts the result in the response queue
5. A dispatcher takes the response from the queue and returns it to the user

This approach of scheduling does not match a time-sharing environment. It is more akin to batch processing. There could be situations where multiple shared processes are tied up with long requests. For this reason, the Oracle 10g documentation states:

"*In the following situations, however, users and administrators should explicitly connect to an instance using a dedicated server process: …*"

The dedicated server process approach creates one process for each connection. A shared server instance can also include net services used specifically to create dedicated server type of connections.

Architecture Comments

The IDS 10.0 architecture provides a superior foundation for performance due to its multi-threaded architecture that provides lower overhead than processes. Oracle agrees:

"*The operating system can spend excessive time scheduling and switching processes.*
…
Due to operating system specific characteristics, your system could be spending a lot of time in context switching. Context switching can be expensive, especially with a large SGA. …"
Oracle 10g Performance Tuning Guide, page 9-9

The earlier thread Primer quote stated that, in Solaris, context switching of processes is about 5 times slower than threads. We can expect similar differences in other operating systems. The impact of context switching becomes more apparent as the system load grows.

The user query scheduling is also important. In the case of Oracle, it has an impact on the number of shared processes that are required to support the system. Oracle configures the minimum and maximum number of shared processes. If more processes are needed, they are created up to the stated maximum, increasing the system overhead. If the maximum is not set properly, it could cause major performance problems by forcing user request to wait no matter how little work they require. For these users, the database server may appear unresponsive if not dead.

As the database system load increases, more processing power must be allocated. Oracle requires more processes where IDS requires more threads. The overhead increase (process/thread creation, synchronization, context switching) of IDS is slower than the one for Oracle once again due to the threaded architecture.

Of course there are other performance and scalability considerations than a better server architecture. This is the subject of the next section.

Other Performance and Scalability Considerations

Performance and scalability start with an architecture that minimizes the system overhead. The next part is to optimize the use of the other resources. It includes parallel execution, indexing, and data partitioning. Other items could be listed but it would take too long to explain the differences. The three listed above constitute major ones that are sufficient to demonstrate differences.

Parallel Execution

Parallel execution is the ability to divide a request in multiple parts that can be executed concurrently. This can be useful when we need to get the result of a complex demanding query as soon as possible or if we have to build indexes. Through this feature, we can execute multiple read operations from multiple disk drives and use multiple CPUs.

IDS has been engineered from the ground up to support parallel execution. It divides the query into multiple threads of executions that are connected through an internal mechanism called exchanges. This provides independence between the different specialized threads (scan, sort, group, etc.). The exchanges balance the load between multiple worker threads to give them the same amount of work. This results in an optimum utilization of the machine resources.

IDS supports a number of parallel operations. It includes index building, sorting, recovery, scanning, joining, aggregation, grouping, and the execution of user-defined routines (UDR).

IDS queries parallelism is determined by multiple factors. It includes the amount of memory allocated to parallel queries, the number of parallel queries that are allowed to run concurrently, and the number of dbspaces accessed in a particular query.

Oracle also provides parallel queries in its enterprise edition. When an Oracle instance starts up, it creates a pool of processes that are used for parallel operations. The degree of parallelism is determined by hints, session setting, table definition setting, and index definition setting. In brief, Oracle parallelism is determined by how many parallel processes you allocate for an operation.

> *"…parallel server processes remain associated with a statement throughout its execution phase. When the statement is completely processed, these processes become available to process other statements.*"
> Oracle 10g Administrator's Guide, page 4-15.

Once again, we see Oracle using processes instead of threads. This uses more system resources than threads as we saw earlier. Oracle parallelism is determined by the DBA or the user who provides the number of processes to use to parallelize the query instead of

letting the database server determine the requirements based on the resources available for parallel operations (memory, CPUs, number of disks). This makes Oracle's parallelism features harder to use and error prone for optimization.

Data Partitioning

Data partitioning provides the ability of distributing the data over multiple disk drives. Each disk can be searched in parallel to provide a higher throughput.

IDS has been supporting partitioning for several years. It allows the distribution of the data either round-robin or by range (expression).

The round-robin scheme distributes the data evenly over a group of disks. It allows you to use all the disk resources without having to know the distribution of the data. It is also useful in environment where you need to process all, or a large portion, of the data like in data marts/warehouses, or in reporting. The range partitioning scheme uses expressions to distribute the data over logical storage spaces called dbspaces. Since the server knows the expression used to distribute the data, it can use that knowledge to eliminate storage fragments during a query and access only the disks that could contain the requested rows. This can greatly reduce the number of rows that must be evaluated, requiring less resources and improving performance.

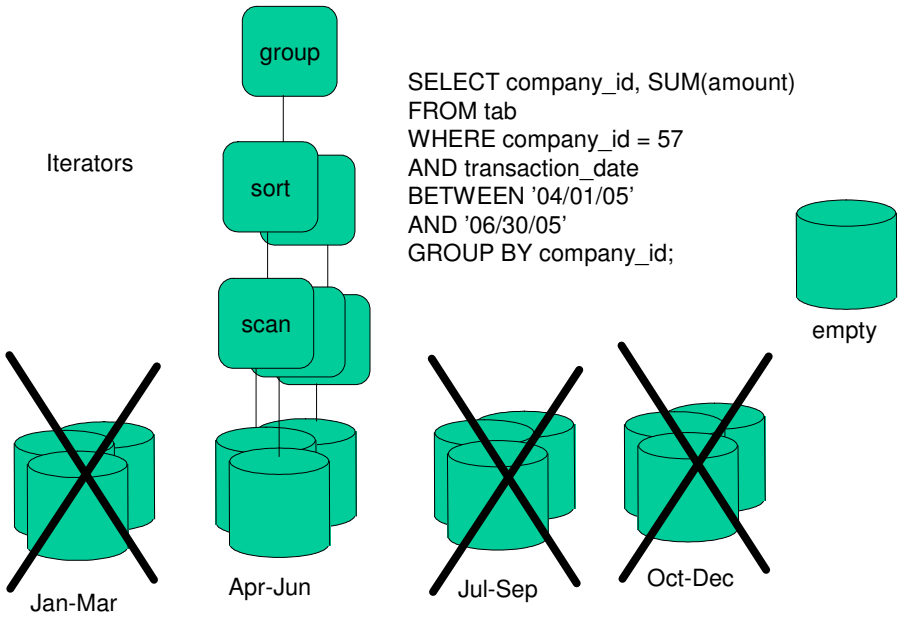
With IDS 10.0, we now have the additional capability to store multiple partitions in one dbspace. This will be particularly advantageous when using finer-grained partitioning where the number of partitions is much larger than the number of physical disk drives. For example, we may want to partition data by day over a 15-month rolling window. You may still want to use 15 different disks and further divide the data by day on each disk. Any query identifying a day or a range of day (like a week) would greatly reduce the amount of data that would be read from disk.

This feature is important because by reduce the amount of data that must be read from disk, we reduce potential disk waits. It also makes better use of the buffer cache by loading into it only the relevant information instead of taking space that could be better used. Indexes can also follow the range partitioning. This means that each partition would have a small index that is faster to search. This gives us another potential performance benefit.

Starting with IDS 9.21 (2001), this feature has been enhanced further. It now allows you to alter a table and attach or detach table fragments. With the fragments following the partitioning expression, old fragments can be removed and new fragments added without having to rebuild the indexes.

If we use a rolling window of data as mentioned above, we can benefit in multiple ways. We could load a new month of data in a new table, create the appropriate index on it, and then quickly attach it to the main table. We could also quickly remove a range of data from the main table, backup the data and free the storage for further use. The result is higher availability and more efficient operations when it comes to removing a range of values.

The following figure illustrates the use of range partitioning in an SQL query.



Oracle decided to follow the IDS lead and provide support for partitioning. It supports three partitioning methods: range, hash, and composite. Hash partitioning uses a hash key to decide in which partition the row will be stored. This provides fragment elimination benefits only in the case of equality conditions. Composite partitioning partitions data using the range method and within each partition, sub-divides it using the hash method.

Oracle's partitioning schemes only support methods where you must analyze your data to make sure you can distribute it evenly over your partitions. The composite partitioning allows you to partition within a partition. This will likely cause more disk head movement, reducing the throughput of the disk device. This makes this option of limited usefulness.

Oracle does provide the ability to attach and detach new partitions. The "ALTER TABLE . . . DROP PARTITION" marks all partitions of the global index unusable. This means that any reorganization to the partitioning of a table will cause major down time since you will have to rebuild your indexes.

Overall, Oracle provides less useful options for partitioning and does not provide the "attach/detach" feature that is essential in many business environments.

Indexing

All database products provide indexing capabilities. IDS supports B-tree indexing, clustered index, and R-tree index. IDS also supports an API that supports the implementation of new indexing methods such as text searches (such as in the Excalibur Text datablade). The B-tree index can also be used to create an index on the result of a user-defined routine (UDR). This is called a functional index.

Oracle supports the following types of indexes: B-tree, cluster, reverse key, hash, bitmap, and bitmap join, functional, R-tree, and other specialized indexes. The first two types

could be considered the workhorses of database systems and have been around for a long time.

The reverse key index differs from the standard B-tree by reversing the byte of each column indexed. Oracle states that it can help avoid performance degradation in Oracle RAC but at a price of less functionality.

> "*…can help avoid performance degradation with Real Application Clusters … Using the reverse key arrangement eliminates the ability to run an index range scanning query on the index*"
> Oracle 10g Database concepts, page 5-29, 5-30

A hash index is a different type of index that is limited in use. It provides good performance for equality operations but is unusable for range type operations.

A bitmap index is used in a data warehouse environment and is useful when very few different values are used in the columns indexed.

> "*Bitmap indexes are not suitable for OLTP applications with large number of concurrent transactions modifying the data*"
> Oracle 10g Database concepts, page 5-31

Oracle had to follow Informix's lead and implement the R-tree index. This type of indexes provides efficient searches for multi-dimensional problems. Its primary application is in spatial and geodetic applications. Oracle has deprecated its quadtree index method for spatial applications.

> "*… the use of quadtree indexes is discouraged, and you are strongly encouraged to use R-tree indexing.*"
> Oracle 10g, Spatial user's guide and reference, page 1-9

Interestingly, the Oracle R-tree indexing method does not appear to be integrated in Oracle as an index:

> "*An R-tree index is stored in the spatial index table… The R-tree index also maintains a sequence object to insure that simultaneous updates by concurrent users can be made to the index.*"
> Oracle 10g, Spatial user's guide and reference, page 1-10

Spatial data is becoming increasingly important to many companies. It can be considered a key indexing method for many OLTP applications. The Oracle quote above shows that it decided to cut corners to "catch up" to Informix instead of providing a optimized solution.

What we saw in this indexing section is that Oracle provides more indexing choices than IDS partly to address some of Oracle's shortcomings. It also shows that at least the R-tree index is not properly integrated in Oracle.

IDS provides clear choices in indexing. There is no confusion on why you want to use one method over another. All the indexing methods are integrated with the engine and provide the best possible performance.

Hardware Scalability

People sometime express some concerns about scalability of a single machine. They wonder if they should look into a product that can make the database server span multiple physical machines. This is part of the sales pitch of Oracle RAC. Since Oracle RAC is an integral part of an Oracle solution, we discuss it in details later in this document.

TPC-C is a standard benchmark that provides some performance information on systems used in an OLTP environment. It is interesting to note that the top two TPC-C benchmark results (as of this writing) are done on single machines. The highest TPC-C result has a result of 3.2 million tpmc (transactions per minutes) on a 64-processor IBM p595. This system had 2TB of memory and 243TB of disk storage.

Only a handful of very large companies worldwide may require this type of performance. Considering Moore's law (processor speed doubles every 18-24 months) we can expect single machine performance to continue to increase at the same rate. So, assuming that the current hardware performance is more than sufficient for a company's current environment, it should continue to be sufficient in the future when we consider Moore's law. Furthermore, managing a single machine environment is much simpler than managing a database that spans multiple physical machines.

Reliability

It is very difficult to compare products for reliability since most of the information provided could be considered subjective. We could point to the Oracle "unbreakable" campaign that generated many disruptions to Oracle sites to prove the point that it was not unbreakable.

IDS 10.0 continues to improve on the reliability of this product to increased levels of testing in the lab. In addition, many customers are pleased with the reliability of IDS. El Salvador Tax Authority, an IDS customer, says:

> "*Storing 10 years of tax records on a safe and reliable platform is vital to the operations of El Salvador.*"

There are many anecdotal evidences that refers to almost forgotten systems because they just run. Some systems come down only because they decided to cycle their computer systems on a quarterly basis. Finally, a large customer reported a large number of instances up and running at 99.998% of the time. This includes maintenance time. This represents a little more than 5 minutes of down time per year of availability. This bodes well for reliability.

IDS is a leading provider in areas such as retail/distribution, telecommunications, electronics, banking and healthcare. All these customers require reliability to run their mission-critical business applications and IDS provides what they need.

Availability

Availability includes the ability to manage your system online (minimize planned outages), to reduce unavailability of data for any reason, and to quickly recover from failure.

IDS 10.0 provides many online operations such as the unique IDS attach/detach feature we saw earlier and index creation and removal. IDS also provides many system management and tuning operations that can be executed while the database system is in running. Customers mention features such as online backups, fast recovery after power issues, speed to trouble shoot issues with the engine or applications along with the extreme ease at which I can get the data needed to see potential performance issues or catastrophic issue before they occur help to keep IDS running at top performance as making the database more available.

IDS also supports a high-performance peer-to-peer replication feature that allows the sharing of any subset of data between hundreds of machines possibly distributed around the world. Several installations, in fact, have such environments. IDS is the only database system able to provide replication at this scale.

For high-availability disaster recovery, IDS provides an easy to setup and manage feature, called HDR, that allows you to replicate a database instance to another location. The second machine can be used as a read-only server for queries and reporting. Since the load on that second machine is less than the primary one, the available CPU cycles could be used for non mission critical operations that can be interrupted in the case of a disaster involving the primary machine. This way, if a disaster occurs, your enterprise still has 100% of resources dedicated to its production applications. The HDR feature in include in IDS enterprise edition and is available as an option for IDS workgroup edition.

Oracle provides some online operations. Let's assume that they are comparable to IDS (with at least the exception of the partitioning feature mentioned earlier) and instead focus on Oracle solution for disaster recovery. Oracle lists a number of solutions. For high-availability, it comes down to Oracle RAC and Data Guard.

RAC is the primary mean promoted by Oracle for high availability. The next section discusses Oracle RAC in details for use in performance/scalability and for high-availability. It explains how RAC works and its limitations. It also explains why Data Guard is a better approach to disaster recovery.

The other solution provide by Oracle, Data Guard, provides a primary-standby setup. The standby machine can be used as a read-only machine by interrupting the replication for the time of the queries:

> "*Data Guard maintains a physical standby database by performing Redo Apply. When it is not performing recovery, a physical standby database can be open in read-only mode, …*"
> Oracle 10g, Data Guard Concepts and Administration, page 2-1

This makes it difficult to use the standby machine. Data Guard can also be used in a logical standby mode to replicate part of the primary database. It can replicate to up to nine standby machines. This is a far cry to the possibilities provided by IDS enterprise replication that provides peer-to-peer replication in topologies supporting hundreds of systems.

Overall, IDS seems to have at least a slight advantage over Oracle in availability. Since w have discussed performance/scalability and availability, we are now ready to look at Oracle RAC since it is an essential part of the Oracle database solution.

Oracle Real Application Cluster (RAC)

RAC is Oracles's premier solution for scalability and high-availability. Oracle pushes the idea of using multiple commodity-based computers to build an enterprise level system. Let's look at the architecture of this solution to see how it stacks up in these areas.

RAC Architecture

Oracle RAC supports a single database view over a number of computer systems (up to 100 theoretically). It is a "share everything" environment. This means that every server participating in the cluster has access to all the data on all the disks. To support this environment, Oracle must coordinate the database operations between servers. This is done by adding such things as:

- Global cache service process
- Global enqueue service process
- Global resource directory distributed over all of the active instances
- Cache fusion that facilitates the transfer of data blocks between instances

These components are in addition to the ones required in an instance that is not in a RAC environment. You must also install Oracle Clusterware before installing RAC.

RAC Scalability

Oracle promotes the implementation of RAC on several machines based on commodity processors for scalability. This section compares the scalability of this approach compared to single machine scalability. If we have a cluster of four single-CPU machines with 1GB of memory each and some disk drives, we would compare to a four-CPU machine with 4GB of memory and the appropriate number of disk controllers and disk drives. How do the scalability of these two approaches compare? My contention here is that RAC is the wrong approach for scalability as explained below.

System Overhead

Taking the configuration example given above, we see that the RAC approach uses four copies of the operating system and related processes instead of one. It also uses four database instances with all its processes and data structures instead of one.

To coordinate between the instances, it also requires the additional processes and structures as mentioned above. Oracle provides some warning with such statements as:

> "*The SGA size requirements for RAC are greater than the SGA requirements for single-instance Oracle databases due to cache fusion*"
> Oracle 10g Clusterware and RAC Administration and Deployment Guide, page 1-6

SGA stands for System Global Area. This is a structure that is maintained in shared memory.

An operating system requires memory to run. For example, the minimum configuration for Linux is around 32MB with 64MB recommended. You have to add to this the requirements of a database server instance. Oracle's quick installation guide for Linux

lists hardware requirements of 1024MB for memory with 400MB of disk space in /tmp and between 1.5GB and 3.5GB of disk space for software installation.

A RAC implementation requires multiple copies of the operating system, multiple copies of the Oracle database software (and Oracle clusterware), and additional memory overhead on each machine for cache fusion. It is obvious that the 4GB of memory provided in the cluster is not equivalent to the 4GB provided on a single machine. The difference in effective memory is measured in tens of megabytes.

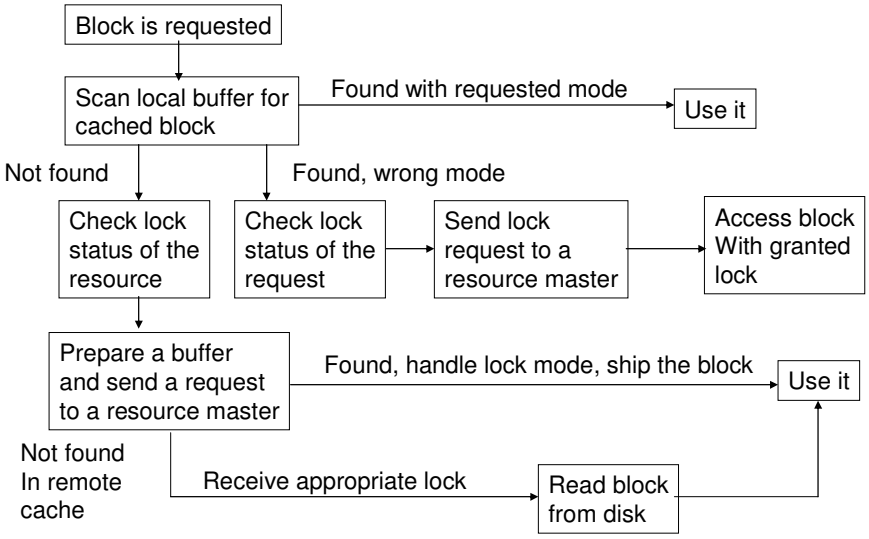
We saw in the performance section of this document that computer components have different performance characteristics. You can improve your system performance by using more memory and reduce your disk access (I/O). Oracle RAC effectively reduces your capacity to take advantage of this critical performance improvement.

Cache Fusion Overhead

All database systems use memory to keep data close to the processor instead of having to always go get it from disk. Since memory performance is much higher than disk performance, it greatly improves the overall performance and scalability of database systems.

Each Oracle instance in a RAC environment runs on its own machine and has its own cache. Since all the instances are part of one logical view of the database, Oracle needs a mechanism to keep track of the data in each cache and share the data between machines. In older Oracle releases, the key component of this feature was the distributed lock manager (DLM). Oracle has made some improvements and now uses a mechanism called cache fusion.

Cache fusion allows the database instance to retrieve a data block from a buffer cache on another instance of the cluster through a network connection. This is an unnecessary operation on a single instance system. On a single instance system, the data block is either in memory or on disk. With RAC, it is much more complex. The following diagram explains the process:



What we learn from this diagram is that if the block is in the proper mode in the local cache, it can be used as is. This is similar to a single server instance. In any other cases, we have to talk to a resource manager to handle the locking. Once the lock is handled, we can use the block if it was in the local cache. If not, we have to find out if it is in a remote cache, check the locking mode, and transfer the block to the local machine. If it is not in a remote cache, we obtain an appropriate lock on it and retrieve it from disk.

The transfer from cache to cache is better than having to go to disk (avoiding rotation latency and seek time) but it is still additional processing that is not needed on a single system. This transfer is done over a network connection. Remember that network speed is at least an order of magnitude slower than memory and the CPU is even faster. Oracle gives a warning about this transfer:

> "*The interconnect and internode communication protocols can affect Cache Fusion performance. In addition, the interconnect bandwidth, its latency, and the efficiency of the IPC protocol determine the speed with which Cache Fusion processes block transfers*"
> Oracle 10g Clusterware and RAC Administration and Deployment Guide, page 12-1

The scalability of Oracle RAC is directly related to how often you can avoid going to a remote cache. The best case scenario is to always use the local cache. Then it is not using Oracle RAC but behaves as a single machine instance but still have additional overhead as discussed earlier.

The worst case scenario is that you always have to retrieve a block from a remote cache. Then you have to go through acquiring the global lock and transfer the block. We saw earlier that network speed is much less than memory and processor speed. The block transfer time is very significant when we work at computer speed. The end result would be to reduce your computer system speed from CPU speed to network speed.

A fairer scenario would approximate the chances of having a block local to 100 divided by the number of nodes. In a 4-node RAC environment, you would find a block locally 25% of the time. This means that you would suffer the block transfer on average 75% of the time. This effectively reduces your system performance from CPU/memory speed to network speed.

Many things can impact this ratio. We saw earlier that Oracle provides a reverse key index to help avoid performance degradation in RAC. If multiple machines require the same index or data blocks, the RAC nodes may spend a lot of time waiting for the blocks to become available and transferred to their node.

After a lot of monitoring and tuning, you may be able to figure out how to divide your data so each part is mostly independent from each other part and only one node from the cluster accesses a specific part. This is in fact partitioning the data so each node is independent from another, defeating the purpose of RAC scalability since each machine is in fact an independent database instance. Oracle hints that way:

> "*If you hash partitioned tables and indexes for OLTP environments, then you can greatly improve performance in your RAC database. Note that you cannot use index range scan on an index with hash partitioning*"
> Oracle 10g Clusterware and RAC Administration and Deployment Guide, page 1-12

If you were to hash a table by department, then you could allocate departments to nodes and be sure that no other node would access that data as long as they do not do any operations that require index range scan. This also means that each department is limited to the processing power and memory of a specific node. As soon as you have to allocate multiple nodes to a department, you are back to the problems described above.

Query Processing Limitations

Database systems are able to divide a query in multiple parts and execute the parts in parallel. This provides a better throughput for these queries. In an OLTP system, you may have to run some reports that require the processing of a large amount of data. This can include sorting the data, aggregating results, and so on.

With what we now know about cache fusion, it is easy to see that a running a reporting query could impact the performance of the other nodes by forcing many block copy from machine to machine. Then it would be better to run reporting in off-hours when the OLTP users are offline. Of course the reporting window could be hard to find in a 24X7 operation that operates over multiple time zones.

Large queries perform better when they can use a lot of resources. With Oracle RAC, a report is limited to the processing power of one node. Coming back to our example of commodity processors used in a 4-Node RAC environment, the report query would be limited to one processor and 1GB of memory instead of being able to use 4 processors and 4GB of memory. This limitation can be made even worse if a large amount of data needs to be sorted. As soon as it runs out of memory, it has to use the disks to hold partial results. This greatly reduces the processing speed.

You may have more than one report to run. You could then run each report query on a different node. That would take advantage of more processors and more memory. If you

do that, you are back to the problem of machine-to-machine block copy, greatly increasing the time required to perform these reports.

RAC Scalability Conclusion

We see above that Oracle RAC has serious scalability problems due to significant system and data access overhead. Adding nodes only compounds the problems. Any effort aimed at reducing this overhead would require significant monitoring and tuning. Any change to the processing environment would require additional monitoring and tuning. The end result would still be less scalability that a single system.

For best performance and scalability, better resource utilization, Oracle RAC should be avoided.

RAC High-Availability

We just saw above that RAC is the wrong solution for scalability. This leaves the promotion of RAC for high-availability. The best approach for our discussion would probably be to limit ourselves to two nodes. Since this would make things worse in the case of a failure, we'll keep our discussion to our previous example of four commodity-based machines.

The first thing to note about RAC is that it addresses node failure but does not say anything about disk failure. If you want to have a high-availability environment, you must plan for disk failure.

Another problem is the possibility of site disaster. If a site goes down completely due to fire, earthquake or any other reason, you lose your entire RAC environment. For some companies, losing a data center is a consideration they must account for. In that case, RAC is not sufficient. You would have to add Oracle Data Guard to the environment. As we saw earlier, Data Guard is inferior to the IDS HDR solution.

For the rest of this section, we limit ourselves to discuss the RAC capabilities without considerations for the high-availability limitations mentioned in the previous paragraphs.

Oracle claims that when a node fails, the database server continues to run and you continue to run your applications. I want to address two major issues after a failure: recovery and performance after recovery.

Recovery

Oracle RAC must do several things when a node fails:

- Detect the node failure
- Remaster the datablocks that were controlled by the failed node
- Log takeover and page locking
- Perform redo and undo recovery

The recovery of the filed node is done by one of the surviving node. During at least the time of the log takeover and page locking, the database is frozen because until all the pages are locked, there is no way for any surviving instance to know what pages need recovery.

This shows that when there is a failure, any database product needs to take some time to recover from the failure. This implies some time when the database is not available. Oracle RAC does not provide faster recovery than the IDS high-availability disaster recovery (HDR).

Performance after Recovery

We already know the performance problems of this architecture: The four machines do not provide the performance levels of the equivalent single machine. What happens after recovery from a failure? We run in degraded mode. We have to plan for that otherwise we can end up not having appropriate performance and response time to serve the mission critical applications. This gives us two choices: either we over-configure our systems or we take some applications offline.

If you did performance tuning on your cluster, a recovery from failure could be a disaster. Imagine that you tuned your system where each node handles a specific load and does a minimum machine-to-machine block copy. What happens when a node fails? One node cannot handle the load that was on the failed node (except if each node runs at 50% of capacity or less). The applications that ran on the failed node must be distributed over multiple nodes. This re-distribution causes the remaining nodes to have to copy data between them. The additional buffer activities on the active nodes and the additional machine-to-machine block copy between nodes increases the overhead on the overall cluster. This reduces the overall performance ever further. The result could be an Oracle RAC that is up and running but that cannot handle the load of the production applications. The response time could be such as being barely better to not having the system available.

Using RAC for high-availability forces you to over-configure your environment to accommodate the lost of a node and the additional overhead if you don't want to run a crippled environment after recovery.

What about Active-Active?

The most frequently heard argument for Oracle RAC high-availability is the fact that it runs in an active-active mode: all the nodes are working concurrently on the same database. It seems to be more of an emotional argument than anything else. I can't do anything about the emotions but I can point out multiple issues that show that this is not an advantage.

We saw throughout our discussion on RAC that you must allocate extra resources to match the scalability of a single machine: 4 x 1 CPU does not equal 4 CPUs and 4 x 1GB of memory does not equal 4GB of memory. You must also allocate additional resources to handle failure. This all sounds to me like a hidden active-passive environment. It gets worse.

We saw that a query (really a user session) runs on a specific node. That query has access only to the processing resources on one node. In a sense, we can consider that this is an active-passive environment for each query: one node active, three nodes passive. If the query could have used more CPUs and more memory: too bad.

Finally, we have to understand that the standby machine in an IDS HDR environment does not mean that the hardware cannot be used for other purposes when everything goes

well. For one, you can offload reporting to the standby instance, freeing cycles for the interactive users, effectively increasing the number of users that you can support, especially in a 24x7 environment.

The standby machine will have processing power to spare since it does not have as much to do as the primary server. You can use that spare processing power for less critical tasks that can be suspended when the primary machine fails.

So, instead of hiding your "passive" processing by fragmenting it over multiple machines, why not put it all in the same place and use that machine properly? This way you can optimize the processing on your production systems and make intelligent use of the extra processing to give yourself a business advantage.

RAC Conclusion

This section gave you the technical arguments why Oracle RAC is an inferior solution for both scalability and high-availability. The overhead of the coordination between nodes greatly reduces the scalability. It also fragments resources such as CPU and memory which greatly limits their optimum use.

RAC is also inferior for high-availability. It does not provide any recovery advantages over an IDS solution. It also wastes resources that could be better used to help your enterprise.

To make things worse, the RAC environment requires continuing monitoring and tuning. Making it harder to manage.

Manageability

One major part of manageability is to have a database system that uses a strong foundation, a strong architecture, to adjust to the demand of production applications. This includes the ability to easily adjust to the number of users, the number of queries, and the demands of these queries. IDS multi-threaded architecture provides this strong foundation for dynamic resources allocation.

IDS can be easily configured by taking into account the number of CPUs available on the system, the amount of memory dedicated to the database system, and a rough estimate of the number of users. The monitoring can be done with server studio, web interface (ISA), command lines, and SQL queries to system catalogs.

There are several reasons why IDS is easy to manage. The first reason is that the architecture makes it simpler to manage connections and query executions with dynamic allocation of resources as needed. Another important reason is that it is easy to automate most of the task so database administrators can be pro-active instead of reactive. This frees up their time to do development of stored procedures, functions or triggers, performance analysis, application design, review the new features and determine how we can best leverage them, implementing these features, training others and of course miscellaneous support.

The ease of management is illustrated by examples such as a large international retail customer supporting thousands of IDS instances with a staff of less than 20 DBAs. Another customer, Transportation Clearing House (TCH) states:

"*IDS has greatly reduced our need for customer service personnel and associated expenditures. We have thus achieved over 500 percent growth while maintaining our level of staffing and overhead costs.*"

With Oracle, you have to decide how many processes an instance can have and how many sessions (client connections) can be handled by these processes. It also includes parameters for how many shared server processes, the maximum number of shared server processes, number of dispatchers, and the maximum number of dispatchers. All related to the number of processes to run.

"*In Oracle Database 10g the parameters have been categorized into basic and advanced categories. Administrators will be able to restrict their day-to-day interactions with just with 28 basic parameters. The advanced parameters have been preserved to allow expert DBAs to adapt the behavior of the Oracle Database to meet unique requirements…*"
Oracle Database 10g: The Self-Managing Database

For DBAs to have day-to-day interactions with 28 basic parameters does not sound like an easy to manage database, let alone considering that self-managed, but this is a major improvement for Oracle 10g.

We would quickly get lost in details if we were to try to compare all the features of manageability between IDS 10.0 and Oracle 10g but just consider backup and recovery. Oracle has the following manuals on the subject:

- Backup and Recovery Basics (226 pages)
- Backup and Recovery Advanced User's Guide (458 pages)
- Recovery Manager Quick Start Guide (26 pages)
- Recovery Manager Reference (332 pages)

This is in contrast with the IDS 10.0 Backup and Restore Guide (391 pages) that includes the description of two backup utilities and the point-in-time table level restore capability. This gives us an indication on the effort required to understand this part of database management.

Application Development

IDS provides support for the standard interfaces including ESQL/C, ODBC, and JDBC. IDS comes from an heritage of software development. It started with the Informix 4GL language. This language is still widely used around the world. IBM now provides a modernization path through its EGL language. Customers also have the option of using other products 100% compatible with Informix 4GL such as products coming from the company Four J's.

As far as development tools for IDS are concerned, IBM's approach is to support widely used tools. On the Java side, this means using tools based on the open-source IDE Eclipse. IBM also supports integration with tools such as Microsoft Visual Studio and the .Net environment.

Oracle claims to follow standards but their approach is to develop their own tools and even define their own types in the database instead of using the standard ones. For example, Oracle discourage the use of the `VARCHAR` standard type:

> "*Do not use the `VARCHAR` datatype. Use the `VARCHAR2` datatype instead. Although the `VARCHAR` datatype is currently synonymous with `VARCHAR2`, …*"
> Oracle 10g SQL Reference, page 2-10

Another important aspect of development is to provide ease of use when it comes to using the database. IDS is the leader in database extensibility. This allows the system architects to adapt the database to their business environment instead of compromising their design to fit the database. The difference between IDS and Oracle is more evident in advanced features. For example, an IDS spatial query looks like:

```
SELECT A.Feature_ID
FROM A
WHERE
ST_Overlaps(A.shape,
  ST_GeomFromText(
    'POLYGON(x1 y1, x2 y2, x3 y3, x4 y4 …)',
    5     -- OpenGIS requirement
  )
);
```

This query used standard types and operations from the GIS standard and lets the database engine decide on the best way to solve the query.

Oracle claims to have the same capabilities, including R-tree indexes as we saw earlier. However, they need to provide a more complex spatial query than IDS, the use of proprietary types and even information on how to solve the query:

```
SELECT A.Feature_ID
FROM TARGET A
WHERE
sdo_relate(A.shape,
  mdsys.sdo_geometry(
    2003,
    NULL,
    NULL,
    mdsys.sdo_elem_info_array(1,1003,1),
    mdsys.sdo_ordinate_array(
      x1,y1, x2,y2, x3,y3, x4,y4, …
    )
  ),
  'mask=anyinteract querytype=window'
) = 'TRUE'
```

We see that the Oracle query requires more information that is required to define a polygon. It also requires additional information on how to solve the query. This increases the complexity faced by the developers. The result is longer development, more difficult maintenance, and potentially more performance tuning problems.

The subject of application development could cover hundreds of pages. The example above shows the difference in approach where IDS want to keep it simple and well integrated and Oracle want to provide the functionality without focusing on ease of use.

Conclusion

At a high level, all database systems look alike. They can all get the job done. The real issue is at what cost? Corporations get a business advantage by optimizing their operations at all levels of the company. This includes the amount of hardware required to run their mission critical applications, the number of people required to manage these systems, and the availability of these systems.

We saw in this document that Informix provides a stronger foundation to optimize the performance of computer systems. This results in higher performance and scalability.

In the harder to define areas of reliability and availability, we saw that IDS is a proven product that is second to none. In fact, the IDS high-availability disaster recovery (HDR) is superior to any Oracle solution including RAC and Data Guard. In fact, we saw that Oracle RAC is inadequate for both scalability and high-availability.

When it comes to system management, IDS was designed for hands-off management that is ideal for environments that have limited DBA expertise.

By using IDS 10 instead of Oracle 10g, you can better optimize you computer systems and become more efficient in your business. This translates into real business advantage that can help you become a leader or maintain your leadership in your industry.

References

- *IBM @server p5 570 Technical Overview and Introduction*
  IBM redbook
- *The Design of the Unix Operating System*
  Maurice J. Bach
  ISBN 0-13-201799-7
- *The Design and Implementation of the 4.3BSD UNIX Operating System*
  Samuel J. Leffler and al.
  ISBN 0-201-06196-1
- *A Guide to Multithreaded Programming, Thread Primer*
  Bill Lewis, Daniel J. Berg
  ISBN 0-13-443698-9
- *IDS 10.0 Documentation*
- *Oracle10gR2 Documentation*
- *Oracle Database 10g: The Self-Managing Database*
  (Oracle white paper)