

Technical Comparison of
Oracle9i Real Application Clusters
vs. IBM DB2 UDB ESE V8.1

*An Oracle White Paper
April 2003*

Technical Comparison of Oracle9i Real Application Clusters vs. IBM DB2 UDB ESE V8.1

EXECUTIVE OVERVIEW	5
INTRODUCTION	6
HARDWARE REQUIREMENTS.....	6
Cluster Interconnect	7
Shared disk vs. shared-nothing storage	8
Traditional Shared-nothing.....	8
Traditional Shared disk.....	9
DB2 ESE Shared-nothing	10
RAC and IBM's Parallel Sysplex Shared-Data.....	10
RAC Shared-Cache.....	11
SYSTEM SOFTWARE REQUIREMENTS	12
DEPLOYMENT	12
Data Partitioning	13
DB2 ESE Partitioning.....	13
A Quick Review Of DB2 Partitioning.....	13
Partition Key Selection Considerations.....	13
SCALABILITY AND PERFORMANCE.....	15
Performance on OLTP applications.....	15
Partition Probes	16
Data Caching.....	17
Two-phase Commit	17
Load Skew	17
Transaction Routing	18
Performance on DSS applications.....	19
Why is RAC Better?.....	20
Packaged Applications with Complex OLTP.....	22
Scalability.....	23
TPC-C Synthetic Benchmarks.....	23
TPC-C Schema Is Inherently Partitionable.....	23
Most TPC-C SQL Accesses are Local	24
Packaged Application Benchmarks.....	25
Market Share under Packaged Applications.....	27
AVAILABILITY	28
Unplanned downtime.....	28

Adding nodes	30
Redistribute existing nodegroup.....	30
Create new nodegroup	30
Piecewise redistribution:.....	30
CONCLUSION.....	31

LIST OF FIGURES

<i>Figure 1: A Cluster Comprises Processor Nodes, Cluster Interconnect, and Disk Subsystem.....</i>	<i>7</i>
<i>Figure 2: A Shared Nothing Cluster Database</i>	<i>8</i>
<i>Figure 3: A Shared Database Cluster</i>	<i>9</i>
<i>Figure 4: Cache Fusion utilizes a scalable shared cache for cache coherency; eliminates disk I/O.....</i>	<i>11</i>
<i>Figure 5 DB2 ESE Partition Probes</i>	<i>16</i>
<i>Figure 6: Amazon.com Peak Load</i>	<i>18</i>
<i>Figure 7: Example Of Adaptive Algorithm For Parallelism.</i>	<i>21</i>
<i>Figure 8: TPC-C Schema</i>	<i>24</i>
<i>Figure 9: SAP Scalability with Oracle9i Real Application Clusters</i>	<i>27</i>

LIST OF TABLES

<i>Table 1: Complexity of Popular OLTP Applications vs. TPC-C</i>	<i>22</i>
<i>Table 2: Local data access in partitioned TPC-C schema</i>	<i>24</i>

Technical Comparison of Oracle9i Real Application Clusters Vs IBM DB2 UDB ESE V8.1

EXECUTIVE OVERVIEW

Whether it is based on the current economic climate or the competitive nature of business, there is a renewed trend towards consolidating global corporate information. This consolidation allows companies to reduce the number of servers and personnel computers needed to run worldwide operations at a tremendous cost savings (see “*How We Saved A Billion Dollars*”¹). Another benefit of this consolidation— putting all your information in a global database on the Internet— is that everyone knows where to look to find the information they needed. This centralized sharing of data makes it easier to build collaborative applications for an ever-growing numbers of both internal and/or external users communities.

But with a global database, reliability, availability and scalability are no longer a competitive advantage, they are a basic requirement. The strain on the IT staff to deliver these requirements is enormous, and the pressure increases as you go up the management chain.

Today’s technology managers are faced with fewer choices and receive greater scrutiny as they plan application and database upgrades and purchases. But it is not just a technology decision, it is a business decisions. ROI calculations are more complex than ever. Price vs. performance ranks high on the decision-makers’ list, but the strategists know that price is relative and they must also consider a host of other factors when evaluating a system upgrade:

- Is the platform strategy flexible so we can re-deploy resources as needs change?
- Will the upgrade affect the day-to-day operating environment cost?
- How difficult will it be to deploy existing application? Or new applications?
- How will the application architecture (OLTP, DSS) affect the database performance?
- How much re-training will administrators need to support operations?

The ultimate proof that RAC works is the success of our RAC customers. Customer applications span OLTP, packaged applications and data warehousing. Customers are running on every hardware platform from Windows to Linux to UNIX to mainframes. See WWW.ORACLE.COM for the latest customer success stories.

¹ Ellison, Larry, Oracle, *How We Saved A Billion Dollars*, 2000

With only two real parallel database choices to pick from, Oracle[®] Real Application Clusters and IBM's DB2 Universal Database ESE V8.1, the choice should be easy.

INTRODUCTION

The focus of this paper is to compare the deployment of a real world application in a cluster environment of Oracle[®] Real Application Clusters (RAC) and IBM's DB2 Universal Database Enterprise Server Edition V8.1 (DB2 ESE) parallel database. The reason for this focus is two fold. First, both databases have many features so we need to limit the scope of this paper to where both products compete. And secondly, if you are addressing the needs of today's IT environments, reliability, availability and scalability through modular growth, then clustering is the only solution that will provide a cost effective solution.

We will first look at the hardware and software platforms to see how they measure up, and what, if any, special requirements are needed, as well as the costs that are associated with them. Next, we will look at the deployment of the application and what it takes to roll out a production system. Once the application is up and running, performance becomes the top priority of the data center. We will break this down into three environments, OLTP, DSS and packaged application because they have different requirements and characteristics that need to be addressed.

We can first note that the hardware and software requirements are similar. The abstraction to the application is also the same with both solutions - it gives an illusion of a single database and there is no need to modify the SQL code. The differences are in ease of deployment, performance and manageability.

First, let's clarify some definitions and terminology that we will use.

In DB2 terminology, the term partition is used to refer to a logical node and the data that is owned by the logical node. This usage is different from the RAC usage where a partition refers to a part of a table or index. In this document, we use the word partition to mean a logical node (i.e., the DB2 usage). Each partition has its own buffer pool, package cache etc. It may be convenient to think of a partition as equivalent to an Oracle instance except that it can access only a subset of the data directly.

HARDWARE REQUIREMENTS

Both databases have similar cluster hardware requirements. A cluster is a group of independent servers that collaborate as a single system. The primary cluster components are processor nodes, a cluster interconnect (private network), and a disk subsystem. The clusters share disk access and resources that manage the data, but each distinct hardware cluster nodes do not share memory. Each node has its own dedicated system memory as well as its own operating system, database instance, and application software as seen in Figure 1.

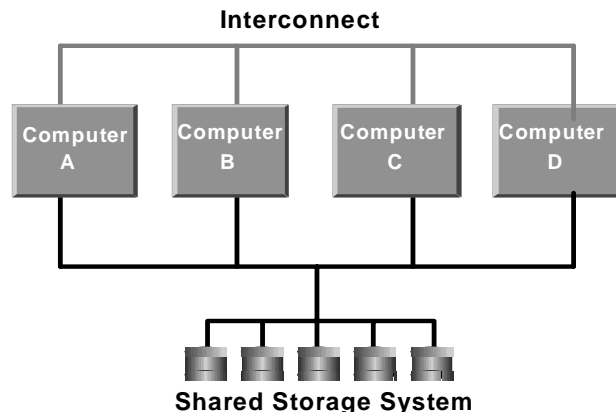


Figure 1: A Cluster Comprises Processor Nodes, Cluster Interconnect, and Disk Subsystem

Clusters can provide improved fault resilience and modular incremental system growth over single symmetric multi-processors systems. In the event of sub-system failures, clustering ensures high availability. Redundant hardware components, such as additional nodes, interconnects, and shared disks, provide higher availability. Such redundant hardware architectures avoid single points-of-failure and provide exceptional fault resilience.

From a conceptual view, the cluster requirements for RAC and DB2 are similar, especially when high availability (HA) is a basic requirement. In a cluster, CPU and memory requirements for each node are similar to single systems requirements and leave no differentiation for this discussion. However, there has been a lot of debate on the other cluster components requirements, especially around performance and costs. The two relevant requirements are:

- Cluster Interconnect
- Shared disk vs. shared-nothing storage

Cluster Interconnect

Each node in a cluster needs to keep other nodes in that cluster informed of its health and configuration. This is done periodically by broadcasting a network message, called a heartbeat, across a network. The heartbeat signal is usually sent over a private network, the cluster interconnect, which is used for inter-node communications.

This cluster interconnect is built by installing network cards in each node and connecting them by an appropriate network cable and configuring a software protocol to run across the wire. This interconnect can be a low cost Ethernet card running TCP/IP or UDP or a high-speed proprietary interconnect like Hewlett-Packard's Memory Channel running Reliable DataGram (RDG) or Hyperfabric/2 with Hyper Messaging Protocol (HMP) depending on where you want to be on the price-performance curve. A generic 4-node cluster configuration is shown in Figure 1.

A low latency/high speed interconnect is best for the performance of RAC because of the cache transfer requirements in OLTP applications and for parallel

query communication in DSS applications. DB2 also needs a fast interconnect for the best performance of its coordinator to worker process communication in OLTP and DSS applications.

Shared disk vs. shared-nothing storage

This is probably one of the biggest differences between RAC and DB2 ESE. There have been a lot of papers written about these two database architectures and yet there still seems to be a lot of confusion. It is important that we understand both the concepts and how they have been implemented because this will severely influence the Total Cost of Ownership (TCO) when you deploy your application.

Traditional Shared-nothing

In pure shared nothing architectures, database files are partitioned among the instances running on the nodes of a multi-computer system. Each instance or node has ownership of a distinct subset of the data and all access to this data is performed exclusively by this “owning” instance. In other words, a pure shared nothing system uses a partitioned or restricted access scheme to divide the work among multiple processing nodes. This works fine in environments where the data ownership by nodes changes relatively infrequently. The typical reasons for changes in ownership are either database reorganizations or node failures.

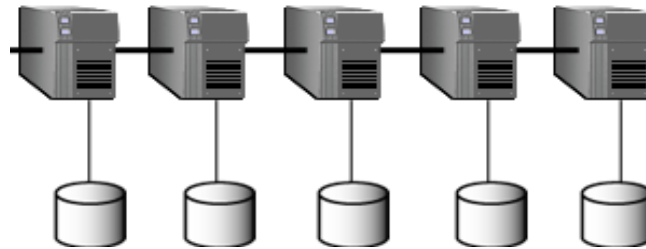


Figure 2: A Shared Nothing Cluster Database

Parallel execution in a shared nothing system is directly based on the data partitioning scheme. When the data is accurately partitioned, the system scales in near linear fashion. This is not as easy as it sounds and is discussed in detail later in this paper because this is where your labor cost can start to grow.

Shared nothing database systems may use a dual ported disk subsystem so that each set of disks has physical connectivity to two nodes in the cluster with a notion of “primary and secondary disk ownership.” Such a configuration protects against system unavailability due to node failures. It requires two node failures to render the system unavailable. While this is unlikely to occur, a single node failure may cause significant performance degradation. This is due to the fact that the one node (in some n node cluster) is now processing roughly twice the work

that any other node is processing and it is on the critical path for transaction completion.

On a superficial level, a pure shared nothing system is similar to a distributed database. A transaction executing on a given node must send messages to other nodes that own the data being accessed. It must also coordinate the work done on the other nodes to perform the required read/write activities (see Two Phase Commit below). Such messaging is commonly known as “function shipping.” However, shared nothing databases are fundamentally different from distributed databases in that they operate one physical database using one data dictionary.

Traditional Shared disk

In a pure shared disk database architecture, database files are logically shared among the nodes of a loosely coupled system with each instance having access to all data. The shared disk access is accomplished either through direct hardware connectivity or by using an operating system abstraction layer that provides a single view of all the devices on all the nodes. “Pure shared disk” is an effective strategy for clustered systems in which equal and direct access to all disks is available from every node. A single node variant of the shared disk scheme maps to a basic SMP system. The ramifications of this will be seen later when we discuss “Adding A Node”.

In the shared disk approach, transactions running on any instance can directly read or modify any part of the database. Such systems require the use of inter-node communication to synchronize update activities performed from multiple nodes. When 2 or more nodes contend for the same data block, traditional shared

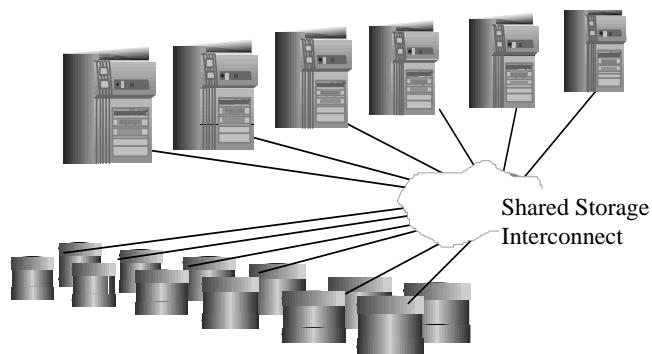


Figure 3: A Shared Database Cluster

disk database systems use disk I/O for synchronizing data access across multiple nodes i.e., the node that has a lock on the data writes the block to disk before the other nodes can access the same data block. The costs of disk I/O for synchronization poses significant challenges for scaling non-partitioned workloads or high contention workloads on traditional shared disk database systems.

The multi-node scalability characteristics of shared disk databases are good for read-mostly applications or applications where you can partition the data for updates which is what we saw in our previous release of Oracle8i Parallel Server. As we will see later, performance will suffer if the partitioning is not done correctly because the cost of maintaining the coherent caches will increase.

A benefit of the shared disk approach is it provides unmatched levels of fault-tolerance with all data remaining accessible even if there is only one surviving node. If a node in the shared disk cluster fails, the system dynamically re-distributes the workload among all the surviving cluster nodes. This ensures uninterrupted service and balanced cluster-wide resource utilization.

DB2 ESE Shared-nothing

Although DB2 ESE is considered shared-nothing, the database must be created on shared-disks in order to provide the highest availability to the data. Shared-nothing only refers to ownership of the data during runtime, not the physical connectivity. But in DB2 ESE it is possible for the disks to be connected only to a subset of those machines that serve as secondary owners of the data in the partition. However, if only a subset is used then some nodes will run hotter, i.e., execute heavier workloads, than others and reduce the overall system throughput and performance. RAC, in contrast, requires full connectivity from the disks to all the machines and hence does not suffer from this. It is interesting to note the IBM's high end Parallel Sysplex solution also avoids this problem by using shared disk.

RAC and IBM's Parallel Sysplex Shared-Data

Both RAC and IBM's Parallel Sysplex use a "shared data" (as opposed to "shared nothing") approach where every server has access to all the data, and any transaction can run on any server. Workload balancing ensures an even distribution of work. There is a single point of system administration. Users can continue to use an application even if a server is taken off-line, and the combined power of all servers can be used against a single application with near-linear scalability. Even IBM agrees on this:

“Data sharing and workload balancing enable work to be directed to available processors “on the fly” and also permit servers to be dynamically added to the cluster without requiring costly downtime. This can be accomplished without splitting application or databases across multiple server , a time-consuming — and expensive— process, often requiring significant new investment in staff or system management.”³

In addition to requiring a great deal of expensive manual intervention, the partitioned approach creates significant difficulties in capacity planning, performance tuning and availability.²

² IBM, " Parallel Sysplex Cluster Technology: The IBM Advantage", GF22-5015-07 October 2001

³ IBM, "IBM Parallel Sysplex clustering technology", G221-4101-05 October 2000

So all the benefits that IBM promotes for its top end solutions can be found on the same shared disk architecture as Oracle. And with Oracle, you have your choice of multiple operating systems (UNIX, Windows, Linux) with multiple vendors at a price point you can afford.

However, to take advantage of the full power of Parallel Sysplex clustering technology, you will need certain proprietary hardware and software elements⁴

- You need a minimum installation of two coupled LPARs
- Each Parallel Sysplex configuration must have one or more Coupling Facilities and Coupling link
- Multiple processors must be attached to a Sysplex Timer, which sets the time-of-day clocks and maintains time synchronization in a Parallel Sysplex cluster.

So all the benefits that IBM promotes for its top end solutions can be found on the same shared disk architecture as Oracle. And with Oracle, you have your choice of multiple operating systems (UNIX, Windows, Linux) with multiple vendors at a price point you can afford.

RAC Shared-Cache

RAC uses Cache Fusion™, a shared cache coherency technology, that utilizes the high-speed interconnect to maintain cache coherence. RAC removes the need for inter-node synchronization through disk I/O and thus eliminates the bottleneck attributed to traditional shared disk clusters.

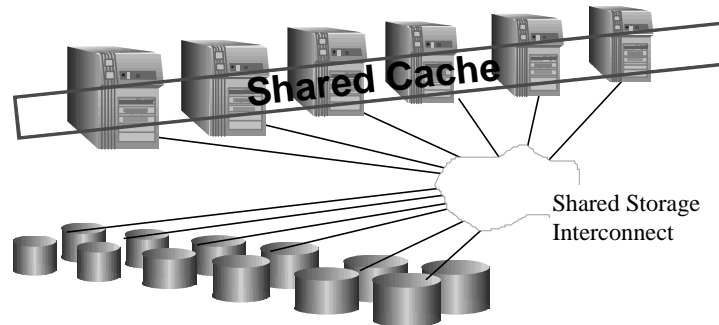


Figure 4: Cache Fusion utilizes a scalable shared cache for cache coherency; eliminates disk I/O

By, utilizing both local/remote server caches, Cache Fusion exploits the caches of all nodes in the cluster for serving database transactions. This is a significant speedup over slower disk I/O and overcomes this fundamental weakness found in traditional shared disk.

The Global Cache Service (GCS) manages the status and transfer of data blocks across the buffer caches of the instances and is tightly integrated with the buffer cache manager to enable fast lookup of resource information in the Global Resource Directory. This directory is distributed across all instances and maintains the status information about resources including any data blocks that

⁴ IBM, "IBM Parallel Sysplex clustering technology", G221-4101-05 October 2000

require global coordination. Update operations do not require disk I/O for synchronization because the local node can obtain the needed block directly from any of the cluster database node caches. This implementation effectively expands the working set of the database cache and reduces disk I/O to dramatically speed up database operation.

SYSTEM SOFTWARE REQUIREMENTS

RAC and DB2 ESE rely on a cluster manager to provide cluster membership services. On some platforms (Windows and Linux), Oracle provides the cluster manager software. Therefore, it is not tied to any limitations in the vendor's cluster software on these platforms. For example, on Windows 2000, DB2 ESE is limited to four or eight nodes (depending on the Windows 2000 edition) in a single cluster and cannot use raw devices with Microsoft Cluster Service (MSCS) for database files. It is possible, however, for a single DB2 ESE database to run across more than eight machines even on Windows 2000. The limitation is that a partition cannot be failed over across machines in different groups.

RAC, on the other hand can run on more than 64 nodes today. This allows you to use more less expensive and smaller nodes to do the same work as fewer but more expensive larger nodes and it extends the ceiling for scaling out to larger, more powerful clusters. And with the support for Cluster File Systems (CFS) and Cluster Alias, you can manage the entire cluster like one large SMP machine.

Summarizing the platform requirements for RAC with DB2 ESE, the hardware and software needs for both databases are about equal. However, when it comes to comparing the cost of deploying the real world application on either platform this will change.

DEPLOYMENT

So far, we have seen that the hardware, system software configuration and installation are similar. Here we will focus on the additional steps required for deploying the database and application on the cluster vis-à-vis deploying it on a single machine.

RAC and DB2 ESE allow users to connect to any machine in the cluster. In some cases, it may be desirable to route client connections to particular machines to improve locality - we address this in the performance section. As such, there are no significant architectural differences with respect to connection configuration.

Deploying a database on RAC from an exclusive instance only requires that the DBA creates as many redo threads and undo tablespaces as the expected number of nodes in the cluster. There is no need to modify the database files.

In contrast, DB2 ESE databases must be migrated to be used with multiple nodes. The data must be physically unloaded and reloaded because DB2 ESE needs to split the data evenly among the disk owned by each node (partitions) in order to

DB2 ESE databases must be migrated to be used with multiple nodes. The data must be physically unloaded and reloaded because DB2 ESE needs to split the data evenly among the disk owned by each node (partitions). Partitioning the data in this way is a very complex task.

make use of the extra hardware resources. Partitioning the data in this way is a very complex task which we will explain here.

Data Partitioning

In a shared nothing environment, data partitioning is a way of dividing your data across multiple servers according to some set of rules. Ideally, you want to get an even distribution to enhance your applications performance which requires a good understanding of the application and its data access patterns.

DB2 ESE Partitioning

In the previous versions of DB2 UDB Enterprise-Extended Edition (EEE) the database partitioning was included. Now, with DB2 UDB ESE Version 8.1, you need to purchase the Database Partitioning Feature (DPF) for the server(s) where you will create the database partitions. So, DB2 ESE with the extra cost DPF option provides the same functionality as the DB2 EEE database.

DB2 ESE with the extra cost DPF option provides the same functionality as the DB2 EEE database.

A Quick Review Of DB2 Partitioning

In DB2, nodes/partitions are assigned to *nodegroups*. Each tablespace is assigned to a *nodegroup*. When a table is created, it is assigned a partition key composed of one or more columns. As rows are inserted into the table, the row's partitioning value is hashed into a hash table where it is assigned to a specific partition based on the current partitioning map. The default partitioning map equally distributes data across partitions in the *nodegroup* but as we will see later it may need adjusting by the DBA in order to prevent data skew or hot spots which affect performance.

A complex application may have several thousand tables. It is possible that not all tables need to be partitioned optimally because only a small percentage of the tables will be heavily accessed. DB2 will automatically partition tables based on primary key or the first non-long column for those tables that do not have a partition key specification. However choosing the partitioning criterion on the few important tables requires a good understanding of the application. This is because there are several competing considerations for optimal partitioning.

Partition Key Selection Considerations

Even data distribution

First, the partitioning key should evenly distribute data to all the logical nodes. This requires understanding of the table definitions and column usage. In some cases, even with this understanding, it is possible that the data cannot be evenly distributed. This can happen when there is low cardinality in a column or unbalanced access to certain keys even in a high cardinality column. For example, if a product table is partitioned by product id, the accesses may be skewed to a particular popular product. DB2 provides a utility called autoloader to change the

partition map when the data is not evenly distributed. But this is not useful in practice for the following reasons.

- The partition map is changed based on current data. Future access patterns may create different skews.
- The partition map will be changed for all tablespaces (and all tables within these tablespaces) that are within the same node group.
- The data may be evenly distributed but the access may be skewed (consider a popular product within a large number of products). It is impossible to eliminate the access skew through data redistribution.
- Redistributing data blocks application updates. It is a high-maintenance off-line operation.

Table collocation

RAC transparently optimizes the message traffic for resolving inter-node cache conflicts.

Another consideration is when you are joining columns you must be careful about choosing your partitioning keys to allow for collocated joins (i.e. a join that can be performed on the same node without inter-partition communication). Collocation is key for performance on DB2. According to IBM, to implement collocation, the designer must examine “if the tables are in the same node group and they have the same number of partition key columns and the data types of partition key columns are pairwise compatible then the same partition key values of rows in different tables result in assignments to the same data partition”⁵.

In some schemas, it is not possible to collocate all the joins. For example, in the TPC-D schema the ORDER table and the ITEMS table can be collocated using the ORDER key, but the join of the ITEMS table with the PARTS table cannot be collocated because the PARTS table is partitioned by PART NUMBER (there is no ORDER key in the PARTS table). While high-speed interconnects make this less crucial, reducing message traffic is still important for cluster database performance. RAC transparently optimizes the message traffic for resolving inter-node cache conflicts.

Data Access Patterns

A third consideration for determining the partitioning keys is the table sizes and frequency of all the queries that will be used. If a table does not have the appropriate collocation columns, it may be possible to create replicated copies on each node. However, this could have some performance impacts which we will discuss in the following section.

⁵ Gene Kligerman, IBM, "DB2 UDB EEE as an OLTP Database", U29 DB2 and Business Intelligence Conference, Las Vegas October 2000

"Our applications are changing on a weekly basis." Matt Swann, Director Database Systems & Engineering, Amazon.com.

The challenges of data partitioning for real world applications can be costly and complex. While some applications change infrequently and it is possible to perform this analysis just once, there are many other applications such as web-based retailers that change their application profiles fairly often. Amazon.com, a successful online retailer, list millions of unique new and used items in categories ranging from books to travel services. According to Matt Swann, Director Database Systems & Engineering at Amazon.com, "the SQL code used by our application changes every week. The pace of this change is dictated by the features that our business demands and cannot be impeded by time-consuming analysis or data redistribution."

Primary or unique keys

Finally, any unique index or primary key must be a superset of the partitioning key. This restriction is present because DB2 ESE only supports indices that are equi-partitioned with the table. Hence it will be impossible to deploy applications that need to enforce uniqueness constraints on columns other than the partitioning columns without application changes.

In summary, shared nothing databases usually require that you manually partition the data for optimal performance. *Shared cache database clusters do not require data partitioning scheme to scale.* With shared nothing clusters, choosing a good partitioning scheme is difficult and must be chosen with care to keep the workload balanced across all nodes and avoid costly re-partitioning. Such re-partitioning efforts are time consuming, labor intensive and could lead to human error (which is the number one reason for downtime). Also this is necessary whenever the clustered system is scaled up by adding nodes (see Adding Nodes). In the real world, applying this technique to a database that contains read/write data is not trivial.

Shared cache database clusters do not require data partitioning scheme to scale.

We can conclude that is not possible to deploy applications "out-of-the-box" on a DB2 ESE system while it is possible to do so on RAC.

SCALABILITY AND PERFORMANCE

There are many aspects to scalability and performance and it is very dependent on both the type of application (OLTP, DSS, Packaged Application) and the workload that it runs. Here we will look at some of the characteristics of each of these application types and how the two database architectures influence the workload's performance.

Performance on OLTP applications

In OLTP environments, a typical transaction is fairly short when compared with a decision support system (DSS). The performance is measured either by throughput (i.e., number of transaction processed in a unit of time) or response time (i.e., amount of time a given transaction will take). We have already

discussed how data partitioning adds complexity to an application's deployment, now let's look at how it affects an application's performance.

Partition Probes

DB2 ESE restricts indexes to be equi-partitioned with the table. Since you must choose only one partition key, all queries not on that partition key are broadcast to all the partitions in the cluster. It is common for important tables in OLTP applications to have up to tens of indexes. A large number of indexes are needed to support fast lookups of the data based on several different SQL where clauses. Since DB2 ESE forces the index to be equi-partitioned with the table, queries on

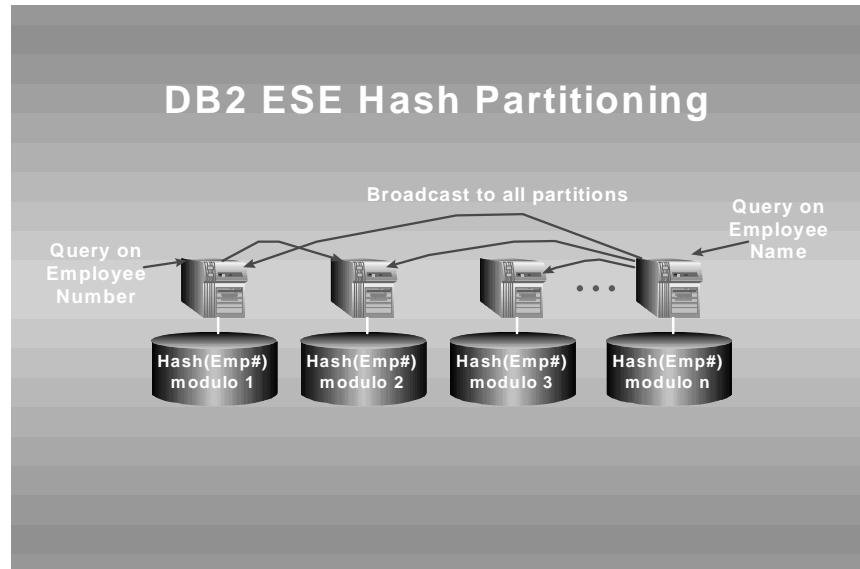


Figure 5 DB2 ESE Partition Probes

indexes other than the partitioning key (or a prefix of the partitioning key) will result in broadcast to all the partitions. This causes multiple partition probes for queries that do not result in partition pruning.

In Figure 5, for example, if the employee table is partitioned by employee number and there is an index on employee name, a lookup of an employee by name will require DB2 ESE to probe the employee name index on all partitions. The total work performed to lookup the employee by name grows with the number of partitions. And if the employee table has more than two indexes, for example 10 indexes then 90% of the queries will have to be broadcast to all partitions (assuming that the queries to all keys are evenly distributed). To calculate the wasted work, let us say an index lookup consumes X cpu cycles and Y logical IOs. If you have N nodes in the cluster, then the same index lookup will consume $N * X$ cpu cycles and $N * Y$ logical IOs. Hence adding more nodes, does not improve your performance .

By contrast, RAC can execute the same query by accessing only the appropriate index pages in the single B-Tree employee name index.

Data Caching

RAC uses the global cache service (GCS) to ensure cache coherency. The global cache service allows RAC to cache infrequently modified data in as many nodes that need the data and have space in their caches. Further access to this data can be performed at main-memory speeds. DB2 ESE systems, on the other hand, must use inter-node communication to access data from another partition even if it has not been modified since the last access.

IBM suggests that you create replicated copies of the table for each partition to reduce inter-node communications. According to IBM⁶ if some tables cannot be partitioned on the same key *but* they are modest in size *and* are typically read-only *then* you can use Replicated Tables⁶. With Replicated Tables, the contents of an entire table (or a portion of it) can be copied to every database partition in the *nodegroup*. However, replicated tables waste storage space. And since the selection of these tables is not automatic you must be careful in choosing which table(s) are being replicated because updating a replicated table impacts the performance.

Two-phase Commit

Any transaction that modifies data in more than one partition on a DB2 ESE system must use the two-phase commit protocol to insure the integrity of transactions across multiple machines. That means you will have to wait for at least two log forces and at least one round-trip message as part of the two-phase commit process before committing a transaction. This increases the response time of the OLTP application.

In RAC, a commit only needs to wait for a log force on the node that is running the transaction. If that transaction had accessed data modified by other nodes in the cluster, these blocks are transferred using the high-speed interconnect without incurring disk I/Os. RAC does require a log force of modifications present in the block before transferring it. However, even on an insert intensive benchmark such as SAP Sales and Distribution Benchmark only a very small percentage of these transfers are blocked by a log force (less than 5%). This is because the log of modifications to a block is continuously forced to disk in the background by the log writer well before it is needed by another node.

DB2 ESE transactions, on the other hand, have to write the prepare records in the first phase of the commit and wait for it to be forced before proceeding to the second phase of the two-phase commit.

Load Skew

DB2 ESE systems may suffer from load skew for two reasons. First, the underlying data may not be evenly distributed in all partitions. This is especially

⁶ Gene Kligerman, IBM, "DB2 EEE in an OLTP Environment", C14 International DB2 User's Group conference, Orlando May 2001

Scalability – Peak Day

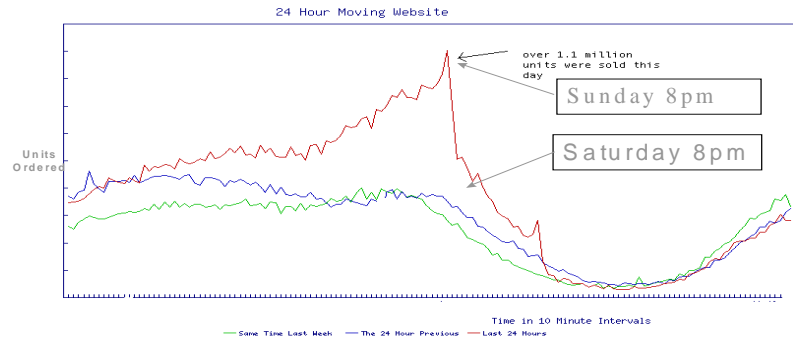


Figure 6: Amazon.com Peak Load

true with low cardinality data. The term cardinality refers to the number of cardinal (basic) members in a SET. Cardinality can be finite (a non-negative integer) or infinite. For example, the cardinality of the set of states in the United States is 50. If you are partitioning this in a 4 nodes cluster, 2 nodes would have 12 states and 2 nodes would have 13 states, which isn't too bad. But if the states were divided by geographic regions (NE, SE, NW, SW) and then you ran some applications based on population, the load would be heaviest on the node for the NE.

Second, the data accesses may be skewed to a small set of column values due to seasonal trends even when the underlying data is evenly distributed. RAC does not suffer from load skew because there is no single node that owns the data, all the nodes can access all the data. If you take a look at Figure 6, the chart shows units ordered per 10 minute intervals. The green line is the previous weeks orders, the blue line is the previous 24 hours orders (Saturday 8 p.m.) and the red line is for the last 24 hours (Sunday 8 p.m.). The spike in the red line is due to a new product release becoming available which pushed the load up to 1.1 million units in a day. Hence, even when the cardinality of a column such as a product ID is high, the load can still be skewed due to high demand for a particular product .

Transaction Routing

When appropriate, it is possible to further improve performance on RAC by routing transactions to a subset of the nodes in a cluster. This improves data affinity and reduces inter-node communication. The routing can be performed easily through the use of service names in the Oracle Net configuration.

Routing transactions by function is more cumbersome with DB2 ESE because it requires knowledge of the location of the data accessed by the transactions. It is also less flexible to changes in load because executing the transactions on more

(or less) number of logical nodes without data redistribution will result in sub-optimal performance.

In some situations, a RAC system can also be configured using appropriate middleware to route requests based on the application's bind values. In these situations, RAC does not require the underlying data to be partitioned. You have a choice in determining how the transactions will be routed. In DB2 ESE, on the other hand, the transactions have to be routed based on the data partitioning. And since only hash partitioning is supported (i.e. range and list partitioning are not supported), you have less flexibility in routing requests.

Performance on DSS applications

Usually a decision support system's workload consists of complex, long running queries, so the focus of the database design is to ensure that each query is executed in parallel across all the database nodes in the cluster to reduce the overall elapsed query time.

We will not compare the quality of the parallel execution plans generated by the optimizers as those comparisons are not specific to clusters. For the same reason, we simply note that Oracle[®] supports more partitioning options for tables as well as indexes that give you even greater scalability and performance from your database. And we have already talked about the consequences of the different database architectures for deploying applications and the effects they have on system throughput if the data is not partitioned correctly.

In the following two sections, we will focus first on the similarities between IBM DB2 ESE and RAC for Parallel Execution. And then we will show how Oracle[®] Real Application Clusters can use the resources available in a cluster more effectively than a DB2 ESE system.

Similarities between IBM DB2 ESE and Oracle[®] RAC for DSS Parallel Execution

The key requirement for scalable data warehousing is the ability to parallelize individual database operations across multiple CPUs and possibly multiple nodes, so that as data volumes increase the database can continue to provide the same levels of performance by increasing the size of the hardware configuration.

There are several techniques for implementing parallel execution which are common to all major database vendors. The first, and most basic, implementation of parallelism is a parallelism schema based upon partitioning. The basic idea is to break every table into N separate partitions, and then when executing a query against these partitions, execute the query with N separate processes (one for each partition). The preferred partitioning technique for this parallelism schema is hash partitioning, which ensures that each partition has the same amount of data.

The roots of this parallel execution architecture is in shared-nothing MPP environments. Each node of an MPP was assigned a single hash-partition. This

technique ensured that each node of the MPP could work independently on separate partitions of data.

Oracle provides this functionality. Oracle provides hash partitioning, and Oracle's parallel model can take advantage of all of the necessary optimizations for execution parallel operations in a clustered environment, including function-shipping, and partition-wise joins (and group-by and sorts). Oracle uses direct reads and direct writes to disk to avoid cache coherency overhead in DSS applications.

The second phase of parallel execution is to support intra-partition parallelism. With intra-partition parallelism, the degree of parallelism for each query is no longer limited by the number of partitions. Instead, the database can devote multiple parallel processes to each partition.

While a basic one-process-per-partition model was sufficient for MPP configurations which only had one cpu per node, intra-partition parallelism is a necessary piece of functionality in order to execute efficiently on today's parallel hardware environments, in which each node typically contains at least 4 CPU's (and often many more CPU's).

Oracle has provided intra-partition parallelism since the inception of its parallel execution functionality (this is due to the fact that Oracle actually introduced parallelism before introducing partitioning). Thus, Oracle has always been able to parallelize any query, regardless of its partitioning scheme, using any degree of parallelism.

Why is RAC Better?

RAC's parallel query architecture is unique in its ability to dynamically determine the degree of parallelism for every query. Unlike DB2 ESE, in which the degree of parallelism is primarily determined by the partitioning scheme of the underlying tables, query parallelism within Oracle is intelligently determined, based on the number of CPUs, the number of files to be accessed, and other variables.

RAC implements an adaptive algorithm for determining the necessary degree of parallelism. This advanced algorithm considers the current load on the data-warehouse when choosing the degree of parallelism. This feature improves the throughput of large data warehouses executing concurrent parallel queries.

For example, suppose that one user issues a query on a data warehouse when no other users are connected. Oracle may choose to execute that query using a degree of parallelism of 40. If nine additional users log into the data warehouse and submit queries, then Oracle may choose a degree of parallelism of 4. As more users issue new queries (or as old users complete their queries), the degree of parallelism for newly submitted queries may be increased or decreased based on system load. At any given time, the data warehouse is using all available

Unlike DB2 ESE, in which the degree of parallelism is primarily determined by the partitioning scheme of the underlying tables, query parallelism within Oracle is intelligently determined, based on the number of CPUs, the number of files to be accessed, and other variables.

resources, but at no time are there so many parallel processes as to overwhelm the data warehouse.

For RAC, the adaptive algorithm intelligently addresses clusters and massively parallel (MPP) environments. Consider the above scenario on a three-node cluster. Initially, with two queries executing in the data warehouse, Oracle may choose to use a degree of parallelism of 12, with the 12 parallel processes evenly distributed across all nodes of the cluster. As more queries are submitted, the degree of parallelism will be scaled back. In the case of 6 concurrent queries, not only will there be a degree of parallelism of four for each query, but each query will also be localized to a single node of the cluster. In this case, the degree of parallelism is small enough for each node to handle all of the parallel query processes associated with a single query.

This ability to localize a query will minimize inter-node communication during query execution, and will improve query performance (see figure 7). This extremely powerful functionality is available only with Oracle, because Oracle is the only database to provide a dynamic parallel query architecture with robust multi-node capabilities.

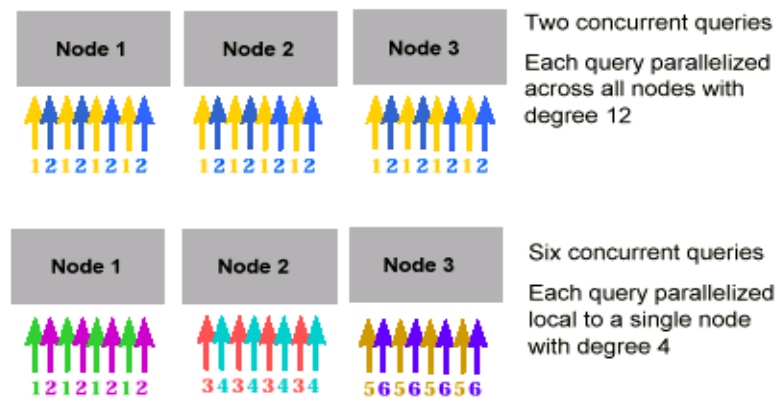


Figure 7: Example Of Adaptive Algorithm For Parallelism.

In summary, the key benefits of Oracle's real-world parallel query architecture are:

- Eliminates the need to trade-off manageability against performance. Oracle9i administrators have the freedom to choose the data partitioning scheme that best maps to their business requirements, in terms of availability and manageability. Oracle's parallel architecture transparently delivers scalable exploitation of query performance, but is not limited by data partitioning and multi-node hardware architectures.
- Maximum utilization of all available processing resources. Oracle's dynamic intra-partition parallel execution enables the server to spread the

workload across multiple processors or nodes, even in cases where a single partition is being accessed.

- A common technology foundation that delivers performance, scalability and manageability on all parallel hardware architectures: SMP, NUMA, Clusters, and MPP systems.

Packaged Applications with Complex OLTP

Popular OLTP Applications such as those from Oracle, SAP, PeopleSoft or Siebel have thousands of tables and unique global indexes. Table 1 gives an indication of the complexity of real world applications as compared to a TPC-C benchmark.

These applications require global unique indexes on non-primary key columns both for speedy data access as well as for ensuring data integrity – e.g., the unique index on *Customer_Number* in the *RA_Customers* table in the Oracle eBusiness Suite ensures that there is only one customer with a particular value of the unique business key (but not primary key) Customer Number. Without these indexes, mission-critical application data can be duplicated.

	Tables	Primary Key Indexes	Alternate Key Indexes	Number of Unique SQL Statements
TPC-C	9	9		20
PeopleSoft	7000+	6400+	900	100,000+
Oracle eBusiness Suite (ERP only)	8000+	1600+	5100	290,000+
SAP	20000	16000+	2800+	273,000+

Table 1: Complexity of Popular OLTP Applications vs. TPC-C

Applications also do not partition their data accesses cleanly – it is not feasible to find partitioning keys for application tables that yield a high proportion of “local” data accesses. Local accesses are those in which the requirements of a query can be satisfied exclusively by the contents of a single partition of data. Most significant queries in Oracle eBusiness Suite, SAP, or PeopleSoft join multiple tables, and different queries use different alternate keys in the join predicates. And non-local data accesses incur the unacceptable performance overhead of frequent distributed transactions.

Thus, to deploy a PeopleSoft or SAP application on a shared nothing database like DB2 ESE would require a Herculean undertaking. Applications developed for Oracle on an SMP require no additional effort to run on RAC.

Real-world OLTP applications cannot reasonably be deployed to run on shared nothing databases.

Scalability

When talking about scalability and performance, we should address all the misinformation about benchmarks. Benchmarks can sometimes be a good way to compare two products running a specific set of transactions against a specific workload provided you understand the benchmark metrics and how they have been implemented.

TPC-C Synthetic Benchmarks

The TPC-C benchmark (currently, version 5.0) is widely used as an indicator of the scalability of OLTP databases and associated hardware. The benchmark models part of an order-entry application for a wholesale supplier of discrete products with a number of geographically distributed sales offices and warehouses. It is a well understood scenario which contains 5 types of OLTP transactions:

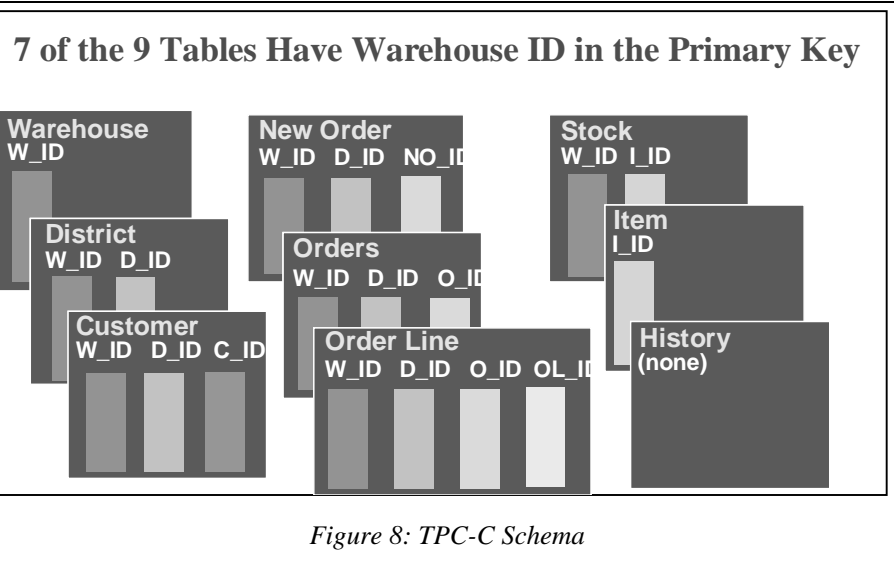
- New Orders - insert new customer orders, 45% of workload
- Payment - update accounts, 43%
- Delivery - batch transaction, 4%
- Order Status - retrieve order status, 4%
- Stock Level - monitor warehouse inventory, 4%

The benchmark measures New Order transactions/minute (tpmC) while enforcing a mix of specific percentages of Payment, Order Status, Delivery and Stock Level transactions. Results are reported on the Transaction Processing Council (TPC) web site (<http://www.tpc.org>) under two categories: Cluster and Non-Cluster results. So let's see how this artificially constructed benchmark (rather than a workload from a real company) with its certain peculiarities has been exploited to obtain great results on every imaginable hardware and software configuration.

TPC-C Schema Is Inherently Partitionable

The TPC-C schema consists of 9 tables, and 7 out of these 9 tables have Warehouse_ID as part of their primary key. Transactions operate against these 9 tables by accessing data through both primary and secondary keys.

90% of the transactions must have a response time of less than 5 seconds, except the stock-level, which must be less than 20 seconds. The fact that most tables reference the warehouse identifier makes this benchmark well suited for a partitioned database because it allows those tables to be collocated. Only the ITEM table uses a different key (HISTORY doesn't have a key) but its size is small and can easily be replicated across all the partitions.



Most TPC-C SQL Accesses are Local

The TPC-C schema can be easily partitioned using Warehouse_ID as the partitioning key. With this partitioning, the vast majority (over 99%) of data accesses are local (node running a transaction only accesses data local to that node):

TPC-C benchmark results obtained on shared nothing databases lack relevance to performance on real-world OLTP applications.

Transaction Type	% of Mix	% of “Local” SQL
New Order	45	99.5
Payment	43	95
Order Status	4	100
Delivery	4	100
Stock Level	4	100

Table 2: Local data access in partitioned TPC-C schema

Consequently, a TPC-C benchmark implementation is really nothing more than a number of mostly independent databases running mostly local transactions. By amassing huge numbers of CPUs in loosely coupled configurations, various vendors have obtained very large TPC-C numbers. It is difficult to find this kind of workload in real-world applications – neither the data nor the transactions can be shoehorned to fit neatly into well-defined partitions.

Running the TPC-C benchmark is no longer a test of hardware or software performance but a test of procuring enough hardware to break the current record.

The consumer of benchmark data – you, the database customer! – should be aware of these idiosyncrasies of the TPC-C benchmark before accepting “clustered” benchmark results. Running this benchmark is no longer a test of hardware or software performance but a test of procuring enough hardware to break the current record.

Packaged Application Benchmarks

As more companies move to packaged applications like SAP, PeopleSoft, Siebel and Oracle eBusiness Suite, application benchmarks are being run to compare performance across platforms. The larger ISVs, like Oracle and SAP, even have multiple benchmarks to test different workloads in clustered environments. This variety of benchmarks give a more realistic view of the customer’s environment but as with all benchmarks you have to be careful when you are interpreting the results.

One particular benchmark is SAP’s Sales & Distribution Parallel Standard Application Benchmark, (SD-Parallel). The benchmark transactions of each component usually reflect the data throughput of an installation (for example orders per day, number of goods movements, etc.). However, benchmark transactions do not reflect reporting, because the resource consumption of customer-defined reports depends on the volume of data. The SAP Standard Application Benchmark consists of a number of script files that simulate the full business workflow of how an order line item is processed:

- Creating The Order
- Creating A Delivery Note For This Order
- Displaying The Order
- Changing The Delivery
- Posting A Goods Issue
- Listing Orders
- Creating An Invoice

Although SAP has some 20,000+ tables installed (plus another 20,000+ indexes), the SD Parallel actively uses 140 hot tables in the benchmark with a high rate of concurrent inserts. It uses LONG RAW columns in many of the inserted and queried tables, i.e. the rows inserted are fairly long (4K and more) which causes fast growing tables. There are approximately 75 user SQL statements used by the UPDATE processes and 150 user SQL statements used by the DIALOG processes.

Because all changes (or requests for changes) are posted in a set of tables, these tables experience heavy update activity. The DIALOG processes post the changes in these tables and the UPDATE processes process these tables and apply the actual change to the database.

The benchmark is response time critical and it is important to get uniform response times from the database. The DIALOG users expect their data to have been applied by the UPDATE processes within 20 seconds after the request for an update has been made (or two dialog steps later). The benchmark results in a failure if this data is not present. Likewise, a benchmark is only certifiable if the average dialog response time is less than two seconds. For this reason, careful balance between the speed of processing both the update processes and the dialog processes is important.

Since the data distribution can significantly influence the benchmark result in a parallel environment, the SAP Benchmark Council established a rule for data distribution in order to establish a means to reproduce and compare results within parallel benchmarks. The additional rule is as follows:

Equally distribute the benchmark users across all database nodes for the used benchmark clients (*Round-Robin-Method*).

This means that users for a certain client are equally distributed across all nodes. There is no data-to-node affinity (partitioning) with this benchmark. It causes a large amount of inter-node traffic and cache-to-cache transfers.

The SAP Standard Application Benchmarks measure all performance relevant parameters such as database request times, wait times, CPU utilization, average dialog response by a given number of benchmark users (with a fixed think time of 10 seconds between each dialog step) and the achieved throughput.

In November 2001, Oracle and Hewlett-Packard (then Compaq) announced an initial performance tests with SAP R/3 4.6C and the Oracle® Database with Real Application Clusters that showed excellent scalability and throughput results. The testing environment included SAP® R/3® 4.6C and Oracle® Real Application Clusters running the parallel SD workload on a four-node HP Tru64 UNIX ES45 AlphaServer cluster with 16 processors.

The robustness of RAC technology provided the foundation for near linear scaling which enables on-demand capacity planning and flexibility in deployment. The performance increased by a factor of 1.81 when going from a one-node configuration to a two-node configuration, and by a factor of

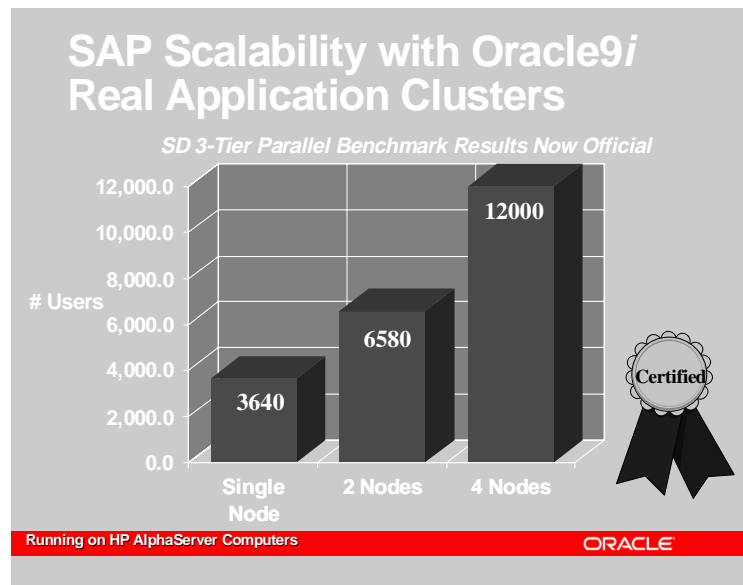


Figure 9: SAP Scalability with Oracle9i Real Application Clusters

3.30 going from a one- to four-node configuration.

The SD benchmark puts considerable load on the system. This is why this particular benchmark has become a de-facto standard in the ERP (Enterprise Resource Planning) industry. Results of benchmark tests are published by hardware partners to inform customers and interested parties in the ERP environment about the currently available performance and scalability of the mySAP.com business solution and the tested platform.

Market Share under Packaged Applications

While benchmarks provide a guideline for comparing the two databases, the real truth is in the market numbers. Which database are your partners using? Which database is your competition using? Although IBM's claim that their strong partnerships with leading ISVs has increased DB2 sales, the fact remains, according to both analysts and the ISVs themselves that Oracle's lead as the database under enterprise applications such as SAP, PeopleSoft and Siebel continues to grow while IBM DB2's share continues to shrink.

According to the more than 600 application developers polled by an Evans Data Survey last spring, the Oracle Database ranks the number one choice for application development.

Surveys indicate that most major packaged application vendors support Oracle databases first. This is true across all categories, including ERP, CRM, Procurement and Supply Chain. Oracle slightly expanded its market share, from 49% in 1999 to 50% in 2000 with a 5% increase of revenue while IBM's revenue dropped 14% with a corresponding decrease in market share⁷.

There are also more packaged applications deployed on Oracle than on DB2 UDB. In fact, according to a Gartner Group report of May 2001, over 71% of all SAP implementations run on an Oracle database, whereas only 8% are deployed on DB2 UDB⁸. Due to the larger Oracle installed base, there are significantly more implementers with experience deploying packaged applications on Oracle than DB2 UDB. Every 6 months, SAP trains 128 times more Oracle DBAs than DB2 UDB DBAs⁹,

Oracle remains the preferred database for Siebel's implementations with 64% of all new implementations on Unix and Windows¹⁰.

Finally, if you want another independent validation, then check the Web sites of Siebel, SAP and PeopleSoft. 60-70% of the customer case studies run Oracle, less than 5% run DB2.

AVAILABILITY

RAC and DB2 ESE support all the conventional methods for maintaining high availability - backup (both on-line and off-line), restore and recovery. However, when you look at the details of each of these features in the Oracle Database - standby database, fast restart recovery, block level recovery and flashback query - you will understand the advantages Oracle provides for minimizing the downtime and hence reducing the TCO of the system. Because we are comparing the two databases in just a cluster environment, we will discuss most of these features in another paper on high availability. For now, we will see how the different cluster database architectures can affect the data availability and the cost of running the system.

Unplanned downtime

In the event of an unexpected node failure, both systems can transparently recover the database. Here, we do not delve on the recovery times of those algorithms that are also applicable to a single node database.

We can expect the recovery times to be faster with RAC for the following reasons.

⁷ AMR Research, "Field Survey of Database Best Practices", September 2001

⁸ Gartner Group, "SAP Platform Choices: What Are the Current Trends?", May 2001

⁹ SAP Training Schedule, May - November, 2001

¹⁰ Tom Siebel, Q2 Earnings Call, August 2001

1. DB2 ESE relies on the cluster manager to restart the partition on a surviving node. This requires the DB2 processes to be started, shared memory to be initialized and database files to be opened.
2. In some cases, RAC will be able to apply redo generated by the failed node on blocks that are already present in the surviving nodes' caches. In these cases, the blocks do not need to be read from disk.
3. And most importantly, after the database has been recovered, applications can be expected to obtain their original response times faster in RAC because the data and the packages needed by the application may have already been cached in the surviving nodes. And with the trend for larger and larger caches, DB2 ESE will take longer and longer to warm the cache in the failed-over partition.

RAC allows the failed connections to be evenly distributed across the surviving nodes. With DB2 ESE, the underlying cluster manager determines which of the surviving nodes takeover the disks belonging to the failed partition(s). To avoid load skew, DB2 ESE must be configured such that each surviving node takes over ownership of the same amount of data. This is achieved by creating multiple partitions on each node. For example, if there are four nodes, three partitions are created on each node for a total of twelve partitions¹¹. If there are n nodes, for even redistribution of partitions to the surviving nodes under all failure scenarios, the number of partitions equals the least common multiple of n, n-1, ...,1. For each partition a preferred owner or takeover list is created using the cluster software (such as HACMP, MSCS) so that the partitions are evenly redistributed across the nodes.

Clearly, in an high-availability configuration, the number of partitions grows quickly with the number of nodes. This creates several problems.

1. It takes more work to administer the cluster. Each partition has its own configuration parameters, database files and redo logs that need to be administered.
2. Each physical node's resources may be underutilized. Although multiple partitions are owned by the same physical node, the partitions cannot share memory for the buffer pool, package cache etc. This causes under-utilization because it is possible to make better use of a single piece of memory rather than fragmented pieces of memory.
3. The probability of load and/or data skew increases with the number of partitions.

¹¹ Nomani, Aslam and Pham, Lan IBM, DB2 Universal Database for Windows High Availability Supporting Using Microsoft Cluster Server - Overview, TR-74.176 May 2001

4. The interprocess communication for a given workload increases with the number of partitions. For example, an application that scales to four logical nodes may not scale to twelve logical nodes. However, for high-availability, DB2 ESE will force the application to run on twelve partitions.

Adding nodes

Adding a node to a shared-disk cluster is seamless when using Oracle 9i Real Application Clusters.

Data partitioning in a shared-nothing environment makes adding new servers to a cluster time consuming and costly, because redistribution of partitioned data according to the new partitioning map is required. Here's what a DBA or Sysadmin has to do add a node to a DB2 ESE database:

- Add hardware
- Configure new partition (set partition-specific parameters, etc.)
- Redistribute the data to spread it across a larger number of partitions

Redistributing the data in a DB2 ESE system involves DBA work and downtime for the database. There are three ways to redistribute this data but all of them interrupt database operations:

- Redistribute existing *nodegroup*
- Create new *nodegroup*
- Piecewise redistribution

Redistribute existing nodegroup

The data in the *nodegroup* is inaccessible until the command completes. The time taken for the command to complete grows with the amount of data to be redistributed. Since this is an in-place redistribution the operation is logged and prone to running out of log space.

Create new nodegroup

Replicas of the old table are created in the new *nodegroup*. This requires sufficient space to store the data stored in the old *nodegroup*. Even with this space, the data in the old *nodegroup* will not be available for modification while it is being copied to the new *nodegroup*. Further, all dependencies such as indexes, triggers, constraints and privileges will need to be recreated.

Piecewise redistribution:

This is similar to the first option except that data in the hash buckets can be redistributed one at a time. This spreads the redistribution over a longer period of time controlled by the user, thereby limiting the window of unavailability at any given time. This is an immense management burden, and will require that the database be off-line for a non-trivial amount of time.

Consider, on the other hand, the management tasks needed when you add a node to RAC:

- Add hardware
- Configure new instance (set instance-specific parameters, etc.)

And that's it! No data re-partitioning, no off-line maintenance – just a seamless scale-up. RAC allows nodes to be added without interrupting database access.

CONCLUSION

This document has shown that there are numerous drawbacks in deployment and performance when using DB2 ESE as compared to RAC. It is clear that RAC is usable in a broader range of applications and scenarios. Although the hardware and system software requirements are similar, DB2 ESE shifts a lot of the burden over to the users while implementing their shared-nothing solution.

Today, there are over a 100 production sites that have proven RAC scales all types of applications. RAC's shared-cache architecture provides many key benefits to enterprise e-business application deployments:

- **Flexible and effortless scalability for all types of applications;** application users can log onto a single virtual high performance cluster server. Adding nodes to the database is easy and manual intervention is not required to partition data when processor nodes are added or when business requirements change. Cluster scalability for all applications out-of-the box — without modification.
- **A higher availability solution than traditional cluster database architectures;** this architecture provides customers near continuous access to data with minimal interruption by hardware and software component failures. The system is resilient to multiple node failures and component failures are masked from end-users.
- **A single management entity;** a single system image is preserved across the cluster for all management operations. DBAs perform installation, configuration, backup, upgrade, and monitoring functions once. Oracle then automatically distributes the management functions to the appropriate nodes. This means the DBA manages *one* virtual server.
- **A lower Total Cost of Ownership;** unlike its shared nothing competitor that requires a great deal of expensive manual intervention and creates significant difficulties in capacity planning, performance tuning and availability, RAC's shared-cache architecture enables a smooth, unlimited growth with near-linear scalability, to handle the most demanding real world workloads.

The more complex the application or the more dynamic the workload, the more you need Oracle® Real Application Clusters.



Technical Comparison of Oracle Real Application Clusters vs. IBM DB2 UDB ESE

April 2003

Author: Sashikanth Chandrasekaran & Bill Kehoe

Contributing Authors: Vineet Buch, Mark Bauer, Rick Greenwald, George Lumpkin, Mahdu Reddy

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the internet.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2002 Oracle Corporation
All rights reserved.