Clustering DB2 for Windows NT

Achieving Scalability and High Availability

Abstract

Ask ten people what clustering means to them and you will get ten different answers. DB2 can take advantage of clusters of Intel based servers running Windows NT in two different ways, first for scalability and second for high availability. This document will explain these capabilities of DB2 in a clustered environment.

Mike Logan
IBM Americas Advanced Technical Support
December 1999

INTRODUCTION

As Microsoft's Windows NT moves from being used as a departmental file application server into supporting mission critical business intelligence and e-business applications, new expectations are being set for the OS. NT-base servers must deliver higher levels of performance, scalability, and availabili ever before. IBM's DB2 for Windows NT helps customers meet those expectations.

IBM developed DB2 to exploit all the key features of Windows NT and to integrate seamlessly into the Windows environment. Capabilities like integ security, performance monitoring, error logging, exploitation of the multithr model, and NTFS and Raw I/O filesystems have allowed DB2 to not only ea the "Designed for BackOffice" logo but to deliver record setting performanc the NT platform as well.

The demands of many businesses today have caused them to outgrow the capabilities of even large 2, 4, or 8 way SMP servers. Many of these custon are turning to clusters of these SMP servers to meet their performance and scalability requirements. With the Enterprise - Extended Edition, DB2 is uniquely positioned to exploit the capabilities of these clusters.

Many customers are also discovering that a traditional NT implementation cannot meet their availability needs, particularly with e-business based applications that have no toleration for down time. To meet these needs, DI integrates with the most prevalent and robust high availability solutions for

This document descibes the capabilities of DB2 for Windows NT versions 5 6 running on the Windows NT Enterprise Edition version 4.0.

CLUSTERING FOR PERFORMANCE AND SCALABILITY

DB2 Enterprise - Extended Edition (EEE)

DB2 Enterprise - Extended Edition (EEE) provides all the capabilities o DB2 Enterprise Edition (EE). This includes all the integration features, "Designed for BackOffice" certification, and all the performance feature What EEE adds is the ability for a single database to span multiple Win NT servers for parallel, clustered processing.

In order to understand how DB2 EEE works, it is first necessary to defir some terminology used in describing a EEE environment.

- A *database* is simply a collection of data.
- A *database manager* is the software that allows users to store an access data in a database.
- DB2 EEE implements a partitioned database system. In a partitioned database system, each database manager manages a portion of the entire database known as a *database partition* .
- A *database partition server (node)* refers to a single database manager, its database partition, and the system resources that it manages.
- A *partitioned database system* or *instance* contains all the database partition servers (nodes), and all share a common configuration.

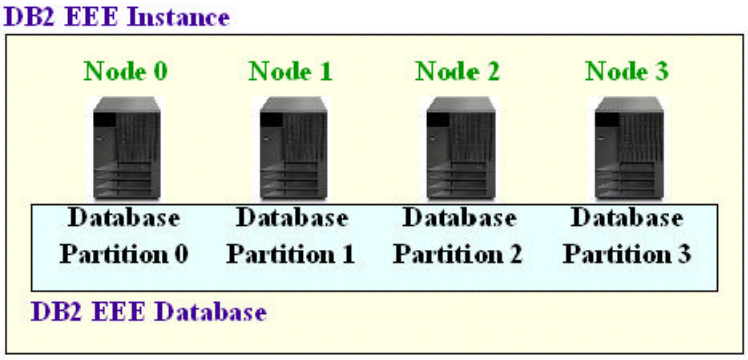

Figure 1: DB2 EEE Concepts

Figure 1 shows a database instance that consists of four database nodes, each of which resides on a separate physical Intel server. The database spans all four nodes so a partition resides on each of the nodes.

It is also possible for multiple partitions or *multiple logical nodes (MLN* exist on a single machine. This type of configuration is shown in Figure this configuration, there are still four database nodes, but only two Intel servers. Each server contains two logical nodes and thus two partitions

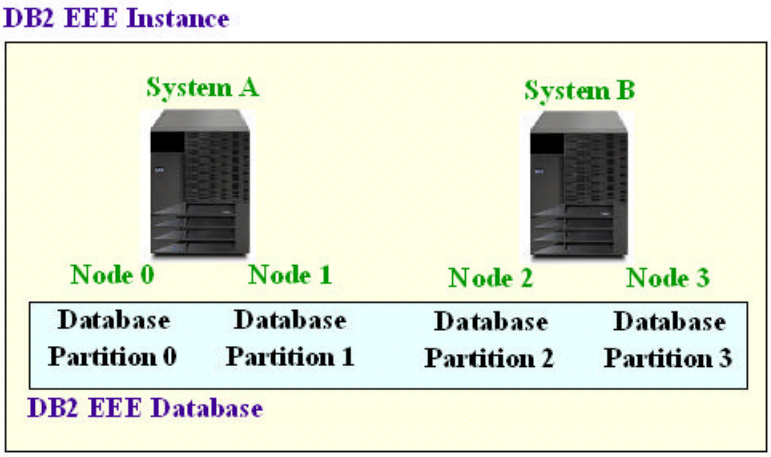

Figure 2: DB2 EEE Multiple Logical Node (MLN) Configuration

the database. Each of the nodes still operates independently, but may sh some system resources. Generally, this type of configuration would onl implemented in very large SMP systems, 8-way or greater; or if other maintenance tasks, such as backups or table reorgs, would benefit from smaller partitions.

DB2 Shared-Nothing Architecture

DB2 EEE implements a *shared-nothing* architecture. This simply mean that DB2's architecture does not require any shared hardware compone This includes CPU, memory and disk. This architecture is important fo several reasons.

First, it eliminates contention for shared resources. Take for example a shared disk architecture. As the number of servers or users grows, there an increased likelihood of an I/O bottleneck as they all compete for the s disk resources. With the shared-nothing architecture, this contention do not happen because each partition has its own disk.

Second, it is completely hardware independent. There are no special requirements for hardware or hardware drivers for this architecture, whi translates into less expensive implementation. Since each server owns a manages its own resources, nothing special is required. The nodes don' even have to be identical, although it is often desirable. In contrast, a shared-disk implementation requires either special hardware or at least special drivers for the hardware to manage the concurrency.

This flexibility of architecture frees DB2 from many of the scalability limitations that plague other architectures. The best demonstration of th was the 1 terabyte TPC-D benchmark published by IBM using DB2 EEI for NT spanning 32, 4-way SMP servers. No other parallel database solution for NT has demonstrated scalability that compares to DB2's capabilities.

How does it work?

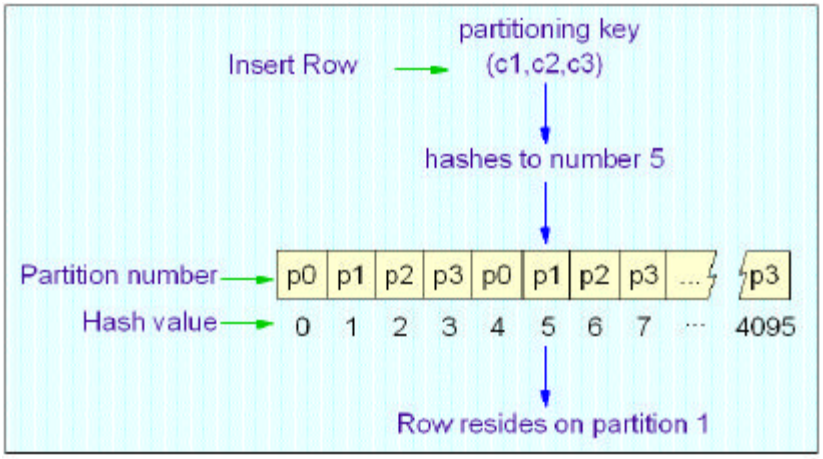

Figure 3: DB2 Partitioning

Remember that DB2 EEE implements this shared-nothing architecture b partitioning the data. Each partition consists of a subset of the entire database. One very important thing to understand is that DB2 automatic maintains the partitioning. DB2 uses a hash-based partitioning algorithr determine the location for any given row.
This concept is illustrated in Figure 3. For each database, DB2 has a partition map that consists of 4096 slots as shown. DB2 assigns partitio to the slots on a round-robin basis. When placing new data in a table, D uses the table's *partitioning key* to hash the row into one of the slots and thus a specific partition.

The partitioning key is a column or group of columns in the table. In the example, our partitioning key consists of columns c1, c2, and c3. These columns are combined and hashed to a number between 0 and 4095. Th row is then placed on the partition assigned to that slot. The same funct used when retrieving data so DB2 knows exactly which partition to get t row(s) from to satisfy a query.

As you may have guessed, the key to a successful implementation of DE EEE is chosing a good partitoning key. There are usually two goals in choosing a partitioning key: spreading data evenly and achieving table collocation.

Achieving an even data distribution is usually easier than you might thir Unlike range-based partitioning (partitioning based on A - E on node 0, K, node 1, etc.), the hash based approach usually results in an even distribution. Frequently, simply ensuring that the chosen key has high cardinality (number of distinct entries) is sufficient. In the event that the you wish to use does not produce an even distribution, you can still achi it by using a custom partition map to favor certain nodes.

Choosing a partitioning key that results in table collocation often has the greatest impact on performance. *Table collocation* simply refers to the ability to keep related pieces of data within a single partition. To illustra this concept, consider the following SQL statement:

```
SELECT A.NAME
FROM STAFF A, ORG B
WHERE A.DEPT = B.DEPTNUMB AND B.DEPTNUMB = 10
```

If DEPT is the partitioning key for STAFF and DEPTNUMB is the partitioning key for ORG, all rows in both tables that reference departme 10 will be on the same partition, or collocated. DB2 will recognize that and only process the query on that one node. If they are not partitioned t way, DB2 will use inter-node communication between multiple nodes to resolve the query. Collocation is particularly desirable in an OLTP environment.

In cases where collocation is not possible, users may choose to impleme some tables as *replicated summary tables*. With replicated summary tables, an entire copy of the table may be stored at each node, allowing a access to that table to be contained within a partition.

Inter-node Communication

In order to complete the explanation of how this architecture works, let's define some more terminology. These concepts are illustrated in Figure First of all, in a DB2 EEE environment, not all nodes are created equal. *catalog node* is a single node in the cluster that contains the DB2 system catalog. The *system catalog* contains all the tables that DB2 uses to defi

its resources such as tables, indexes, etc. The system catalog always res

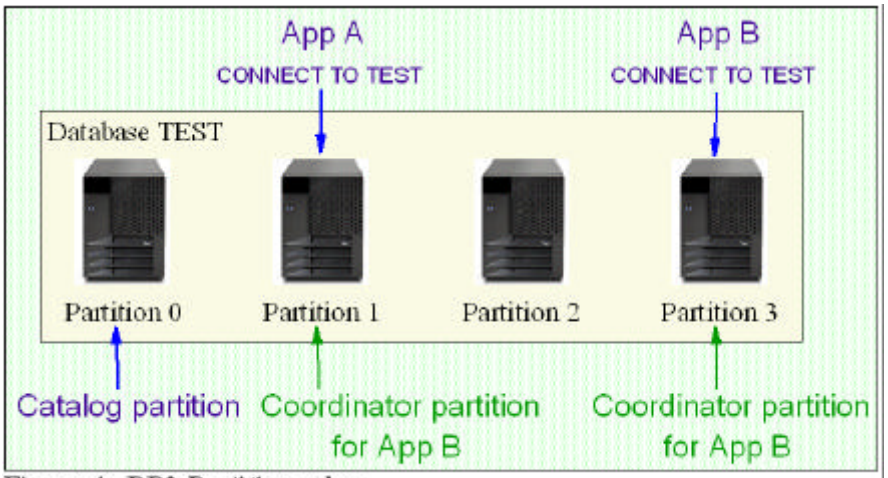

Figure 4: DB2 Partition roles

on one and only one node.

Even though the database physically spans multiple machines, an applic may connect to any of the machines in the cluster and have access to the entire database. The partitioning is completely transparent to the applic Any application code or SQL statement that runs against the non-partitic version of DB2 will run unmodified against DB2 EEE (no recompile, re or optimizer hints necessary). When an application connects to the database, the node it connects to becomes its *coordinating node*. A DB2 agent (NT thread) on that machine will coordinate the efforts of any nod that must participate to resolve a particular SQL request. The request er DB2 at this node, an access plan is generated here by the optimizer, and coordinator sends the request to the correct node(s), and performs any fi processing on the result set.

To minimize network traffic and data movement, DB2 uses a function-shipping model as opposed to a data-shipping model. When th DB2 optimizer processes a statement, it breaks the statement into its low level operations and then attempts to order those operations such that m of the predicates, joins, etc. can be done within a single partition. Throu this process, DB2 can perform much of the work local to the data and av shipping data between nodes, especially in cases where there is a high degree of collocation.

At this point a skeptic might say that this design is actually a "shared-interconnect" instead of a shared-nothing architecture. While th viewpoint could be argued, it is not necessarily relevant. Because DB2' optimizer strives to minimize data shipping, there may not be enough tra

to cause a problem with the interconnect. Many smaller implementation - 4 nodes, can easily support the traffic with standard 100 Mb ethernet. larger or more heavily used systems there are a number of alternatives including gigabit ethernet or Virtual Interface (VI) Architecture based switches. VI is a standards based architecture designed to meet the need for a high-volume interconnect for Intel based servers. Many of these switches actually allow for a dedicated path between each node in the cluster, eliminating any shared interconnect concerns.

Maintenance

A common concern with new or prospective DB2 EEE customers is maintenance of a EEE environment. The good news is that it is not muc different than any DB2 NT implementation. You have a few new tasks, choosing partitioning keys, but for the most part everything remains the same. You have the same graphical tool set to assist with administrative tasks, the same scheduling capabilites, and the same integration with NT things like performance monitoring and error reporting.

Creating the database is only a little different. With the exception of add the partitioning keys, the same DDL can be used to define EEE database any other DB2 database. Loading data into the tables can be slightly different. If using INSERTS or IMPORT to load a table, nothing is different. If using the LOAD utility, then DB2 requires that the data be partitioned prior to loading. Fortunately, DB2 provides a utility that automates the loading process for EEE called the DB2 Autoloader. This simple utility performs the partitioning of the input data (split), any mov of data, and the loading of data very efficiently.

CLUSTERING FOR HIGH AVAILABILITY

Although there have been a number of improvements in the reliability ai availability of Intel servers, like RAID storage, redundant components, a hot-swap capability, there are still cases where a single NT/Intel based solution will or must come down. If you've ever experienced the infam blue-screen-of-death, then you undertstand . In an NT world, the solutic to this possibility tends to be a failover scenario. In this scenario, when machine fails another takes over its function with minimal downtime or interruption to the user. On NT, as on other platforms, DB2 exploits the best of the high availability offerings for the platform. On NT, this is Microsoft Cluster Services(MSCS) or IBM's Netfinity Availability Extensions for MSCS(NAE).

MICROSOFT CLUSTER SERVICES (MSCS)

Microsoft Cluster Services (MSCS)

MSCS is the Microsoft clustering-solution software used with Windows Server, Enterprise Edition. MSCS version 1.0 supports clusters of two specially linked servers running Windows NT Server, Enterprise Editior The primary function of MSCS occurs when one server in a cluster fails taken offline. When this occurs, the other server in the cluster takes over failed server's operations. MSCS does not provide load balancing or parallel processing.

All server versions of DB2 (Workgroup Edition, Enterprise Edition, and Enterprise - Extended Edition) are MSCS aware. In a MSCS environme DB2 supports both active-passive (hot standby) and active-active (mutu takeover) configurations.

Hot Standby

The hot standby configuration is the simplest. As shown in Figure 5, this configuration consist of DB2 actively running on one server in the cluster pair. The second server is a dedicated hot standb that does nothing unt a failure occurs. In the event of a failure, all

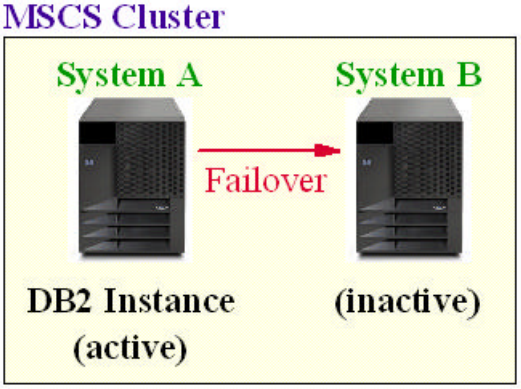

Figure 5: DB2 Hot Standby Scenario

The above examples have shown high availability of non-partitioned databases. Since EEE databases often span more than the two machines supported by MSCS, can it be used to provide high availability in this environment? Absolutely. Because of the shared nothing architecture o DB2 EEE where each node has its own resources, any two nodes in a EI cluster can be paired together in an MSCS cluster. For example, Figure shows a four node EEE implementation that would span two MSCS clusters in an mutual takeover scenario. In a EEE/MSCS environment y can have either a mutual takeover configuration , a hot standby configuration, or a mixture of both. You might want a mixture in a situa where you had an odd number of EEE nodes. Since MSCS only suppor

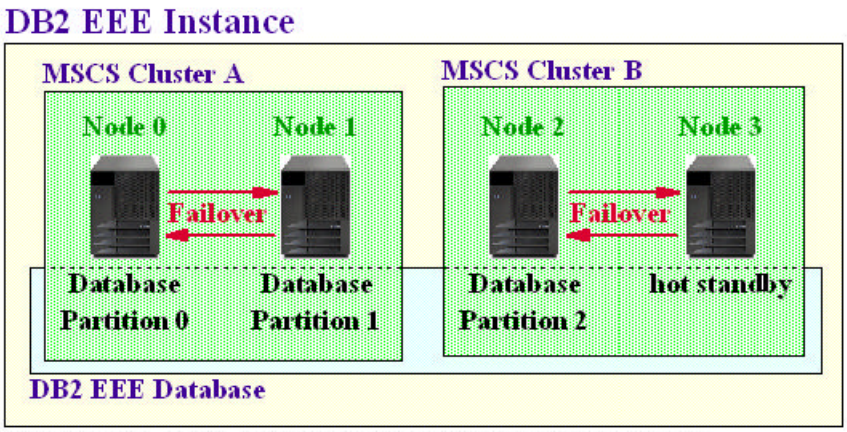

Figure 8: DB2 EEE and MSCS (odd # nodes)

pairs, you would still need one extra to provide failover for the odd node This environment is illustrated in Figure 8.

How Does It Work?

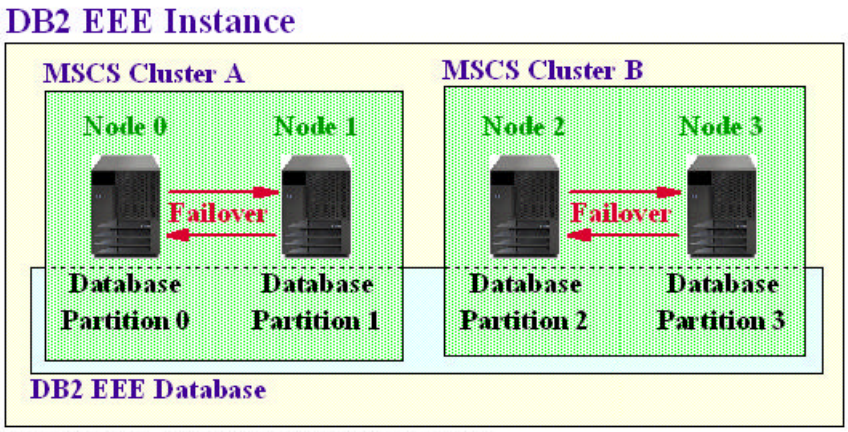

Figure 7: DB2 EEE and MSCS

IBM Netfinity Availability Extensions for MSCS (NAE)

As suggested by its name, the Netfinity Availability Extensions for MSC extend the capabilites provided by MSCS. NAE provides the same failc type availability, but does so for clusters of up to eight nodes. With NA multiple server failures can be tolerated without loss of service. This ab for larger clusters makes NAE ideal for providing high availability for D EEE databases or for a farm of non-partitioned DB2 Servers. Figure 9 show an example of an eight-node EEE system with cascading failover support. This example is similar to the mutual takeover scenario of MS Each node is running one DB2 partition and acting as the backup node f another.

Figure 10 shows and example of a many-to-one failover scenario. This scenario is similar to the hot standby configuration of MSCS. However, with

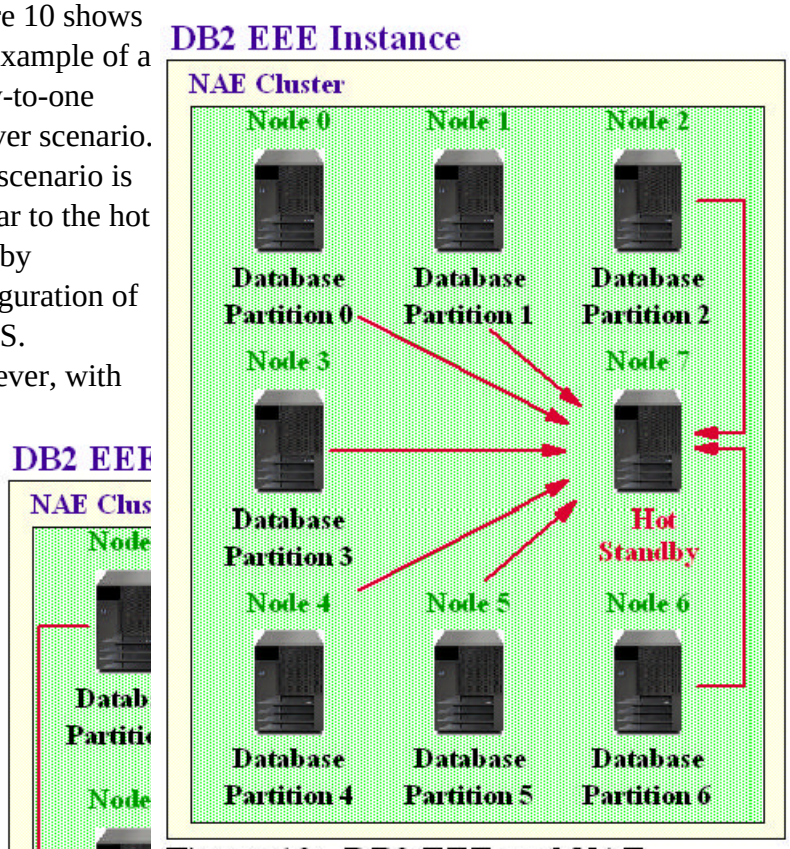

Figure 10: DB2 EEE and NAE Many-to-one Failover

Figure 9: DB2 EEE and NAE Cascading Failover

NAE, only one node is inactive to support 7 active nodes. This can be a much more cost-effective way to provide high availability with no degradation in performance. Performance would only begin to degrade the failure of two or more cluster nodes. Another good possiblity is tha

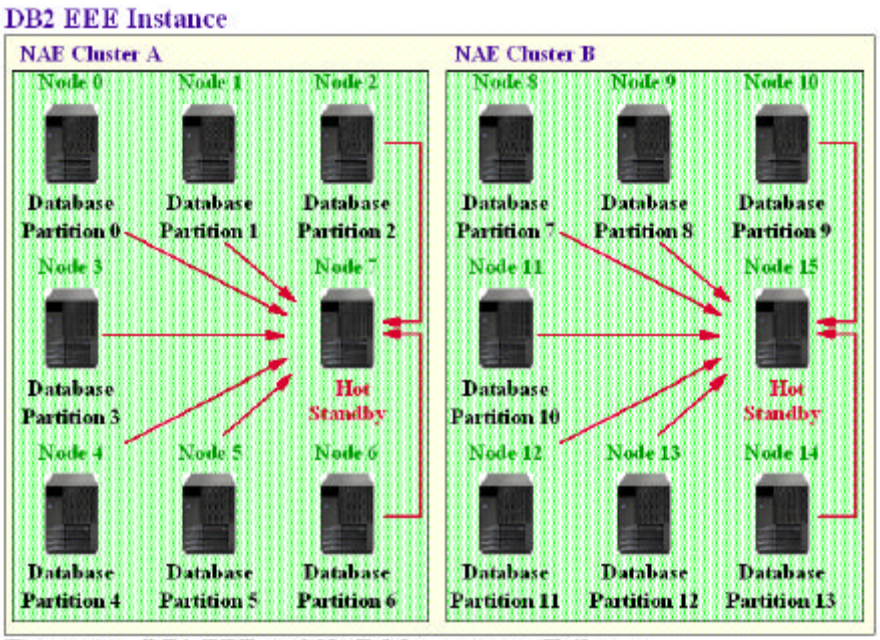

Figure 11: DB2 EEE and NAE Many-to-one Failover
13 Database Partitions, 16 Nodes

some of these nodes could be database, some file and print servers, etc allowing a single spare to server a variety of functions. Perhaps the best of NAE is the way it is sold and packaged. IBM only sells the product a part of a complete availability solution. The availability cluster is delive configured and functional.

The Ultimate in NT Availability and Scalability

Just like with MSCS, DB2 EEE can span multiple NAE clusters for high availability scenarios that go beyond the current eight node limitation. T allows incredible possiblities in terms of scaling to support any database or any number of users on the NT platform while maintaining system availability and reliability. Figure 11 shows one example of how a EEE /NAE configuration of greater than 8 nodes might look. This configurat has 13 of the 16 nodes active as database partitions. Two of the 16 node are acting as hot spares. This system could operate at consistent performance levels with a loss of a single node in each cluster and conti processing with failure of several nodes.

How Does it Work?

In most aspects, NAE behaves exactly like MSCS. In fact, behind the scenes each node is installed as a single node MSCS cluster. NAE mana all the interaction between these single node clusters to provide one larg cluster. NAE uses the same API as MSCS, with a few extensions, and a very similar user interface. Like MSCS it requires a shared interconnect a shared storage bus (currently only supports fibre). Each node will still have one or more disk resources assigned exclusively to that node. The failure detection and failover works just like MSCS. Cluster resources belong to virtual servers and the physical location of the resources is transparent to clients just as in MSCS. DB2 clients would be affected by failure in an NAE resource exactly as they would with MSCS.

SUMMARY

IBM has committed to making DB2 the database of choice on Windows This is being done by delivering the best in relational technology and by exploiting the capabilities of Windows. DB2's clustering capabilities fo both scalability and high availability are key examples of this effort.

FOR MORE INFORMATION

IBM DB2 for Windows NT Home Page:
http://www-4.ibm.com/software/data/db2/udb/udb-nt

IBM Data Management Home Page:
http://www.software.ibm.com/data

IBM DB2 Product and Service Technical Library
http://www.software.ibm.com/data/db2/library

IBM Netfinity Avalability Extensions for MSCS
http://www.pc.ibm.com/us/netfinity/mscs.html

Microsoft Cluster Services
http://www.microsoft.com/ntserver/ntserverenterprise/exec/overview/Cling/Default.asp