A technical discussion of DB2 for zSeries on Parallel Sysplex
Spring 2002

DB2 Data Management Software

DB2 for zSeries on Parallel Sysplex: Your Key to Application Availability, Scalability and High Performance

IBM Software
Jeff Josten, Curt Cotner and Casey Young

Contents

1 Availability
2 Scalability
3 Parallel Sysplex Technology
7 Oracle Real Application Clusters
11 ITG Study
12 Conclusion
13 Notices and Trademarks

Introduction

Today's business requires availability – all the time – no excuses. Anyone who has a serious investment in e-business understands the cost of a lost customer that encounters an unavailable Web site. A business that sells across borders is becoming more the norm than the exception. Someone is awake somewhere, and chances are, they are doing business. If your system is unavailable because you are doing maintenance, or your system is cumbersome to use or takes too long to access, your competitor's Web site isn't.

How do we keep our systems available to the 24-hour customer searching for premium service? How do we grow these systems without losing speed? With DB2 UDB for OS/390 and z/OS using Parallel Sysplex technology, you can have what you need -- availability, scalability and high performance.

Availability

To the end user, both planned and unplanned outages look the same. The application isn't responding. If it's a Web application, the customer may move on. To a financial application, the failure of any part of a system may result in financial penalties, the size of which depends on how long the application is unavailable and how much data has been lost. An application design that needs to minimize downtime, must focus on eliminating single points of failure.

With today's complex applications, the opportunity for single points of failure abound. Some, the company doesn't have control over, but a great many, the company does.

In addition to the network systems and routers that carry the traffic, the servers perform functions and processes that are vital to the application. Database applications contain the heart of today's businesses -- information.

Highlights

To protect these applications, redundancy is used in both in the hardware and software arenas. While this works for many of the network and application server set ups, databases present some special difficulties because the performance cost of redundancy can become detrimental as databases grow and transaction requirements increase. This is especially true with online transaction processing (OLTP) applications where database updates need to be coordinated across multiple systems. In addition, the hardware and software costs of this method can mount up. For those applications that replicate in order to have redundant copies, data latency may grow, leaving the replicated data in an earlier state than the primary data – sometimes by hours.

Scalability

In order to thrive, a company has to grow. With content becoming larger (pictures, videos and the like) and more numerous (more customers as companies merge), the need to scale an application without loss of performance or availability is an imperative.

There are many ways to get to a database server: the Internet, client-server, legacy applications, Java. The list goes on and on. But, the heart of the matter is information – information kept in a database. Insuring access to the database maintains performance and availability as the amount of information grows is a matter of designing large databases (multiple tens of terabytes) so that information and processes to access the database can be added as needed.

Insuring access to the database maintains performance and availability

Primarily, there are two methods of clustering databases for increased scalability and availability, each valid under different circumstances. The first method uses a shared-nothing architecture such as DB2 UDB EEE, Informix, Sybase and Teradata. In this case, the data is separated and placed on multiple servers. The essence of shared-nothing architecture is that multiple systems within the cluster do not share memory or disks, but communicate with each other via high-speed inter-system links. These systems are actually computers that can stand on their own. Within a shared-nothing architecture each system is called a node and the database is partitioned such that each partition is managed by one of the nodes. Because they do not have to share any part of the computer architecture, shared-nothing can scale as the business grows, by adding new nodes to the complex and redistributing the data.

Highlights

These database systems are designed to exploit the hardware they were built on. Since there isn't a hardware assisted coupling facility on Linux, UNIX or Windows operating systems, the shared-nothing database will scale more efficiently than a shared disk database on shared-nothing hardware.

The alternative method, and the focus of this paper, is to use a shared-disk database management system (DBMS) cluster architecture, such as is used in DB2 UDB for OS/390 and z/OS taking advantage of Parallel Sysplex technology or Oracle using Oracle Parallel Server (recently rebranded to Real Application Cluster) technology. The essence of a shared-disk architecture is that the database resides on disks that are accessible from all the systems. In a shared disk cluster, each system is considered a member of the clustering configuration and has read/write access to the shared data. (Oracle uses the word "instance" instead of member.) The members communicate with each other regarding their use of this data. Implementation of this concept between Parallel Sysplex and both Oracle Parallel Server or Real Application Cluster technology differ.

Parallel Sysplex Technology

Parallel Sysplex Technology was developed to with online transaction databases (OLTP) in mind. These types of systems require millions of concurrent updates daily. With the change of business intelligence and data warehouse databases away from a read-only and towards a near real-time view of the business, this technology can assist the maintenance of this crucial data.

To use an overworked, but still useful example, what if someone wants to move money from checking to savings at the same time his or her partner is moving money from savings to checking? If financial order is to be maintained, it is important that one person 'wins' in the event of a collision. That is, someone will need to complete their transaction before the other one is allowed to begin. Within one database image, such as a DB2 subsystem, this ordering is maintained by locking the data until one unit of work (debit from checking; credit to savings) is complete. DB2, for example, knows that the unit of work is complete when the application commits the transaction.

DB2, for example, knows that the unit of work is complete when the application commits the transaction.

Highlights

The method used for this coordination has to give similar response time to the response time achieved with locking within one subsystem.

But what if these two transactions are being run from two different members within a cluster? Requests for data update will need to be coordinated. The method used for this coordination has to give similar response time to the response time achieved with locking within one subsystem. How that coordination is performed differs from relational database management system (RDBMS) to RDBMS.

IBM's Parallel Sysplex technology provides special purpose clustering hardware in the form of Coupling Facilities (CFs) and Sysplex Timers. This zSeries (formerly S/390) hardware was built in conjunction with Parallel Sysplex software development to insure that the hardware was designed for multi-system shared-disk clusters. Within the z/OS operating system a special component, XCF/XES, provides a robust set of clustering services including "group management," intersystem messaging, interaction with coupling facilities and recovery management.

. . . communication with the CF is extremely fast, thanks to the special purpose hardware. In fact, it is so fast, that access times are measured in micro-seconds.

How does this work? First, let's take a conceptual look at what is involved in data accessed from multiple systems. DB2 member M1 retrieves Page A from Table 1 in the database for reading. M1's read interest in a Table 1 is noted in the Coupling Facility. M2 retrieves the same page and its read interest is also recorded in the Coupling Facility. Since there is no conflict with either member's read interest, nothing further needs to be done. It is important to note that communication with the CF is extremely fast, thanks to the special purpose hardware. In fact, it is so fast, that access times are measured in micro-seconds. Processes don't need to relinquish control of the CP as is needed with disk I/O or inter-system message passing.

Coordination gets complex when one of the members needs to insert, update or delete a row from a specific page in the table – the type of processing normally found in OLTP workloads. Based on the type of statement issued, information about the intention to update is sent to the Coupling Facility. So, if M1 registers its intent to update Page A in the Coupling Facility, M2 will not be allowed to update the same page until M1 is finished. However, if M2 wants to do the update, it isn't automatically refused. There is a negotiation process to determine if M1 is really going to do the update, or has done the update and hasn't committed. Depending on the results of the negotiation, M2 may be allowed to do its update prior to M1.

Highlights

incoming buffer invalid signals do not cause CP interruptions on the receiving systems, thus avoiding a significant source of overhead that is incurred by shared-disk systems without CF hardware assists.

The Coupling Facility also contains the group bufferpool. The group bufferpool represents a true shared cache. (Shared Cache is a term that Oracle uses to describe their RAC implementation when in fact none of the buffer caches in Oracle are truly shared because they are running on shared-nothing hardware.) Once a member has updated a page and committed the change, the page is written to the group bufferpool. If necessary, the CF hardware sends buffer invalidate signals to the other members that have registered interest in the page. The page writes and invalidations can be done extremely quickly with minimal overhead due to the special purpose CF hardware and the fact that signals are sent to only those members that have an interest in the page. Another critical point is that incoming buffer invalid signals do not cause CP interruptions on the receiving systems, thus avoiding a significant source of overhead that is incurred by shared-disk systems without CF hardware assists. If another member is waiting for this page, the invalidated buffer can be detected via a special purpose hardware instruction, and the page can be retrieved from the group bufferpool rather than re-retrieving it from disk, which is orders of magnitude slower.

As you can imagine, all this coordination takes resources. If resource allocation is not done correctly, it can impact performance. To understand the impact, it's necessary to delve a little deeper into the mechanics of computer architecture.

Parallel Sysplex uses CPU-synchronous processing to communicate read/write interest and bufferpool contents to the Coupling Facility, thus making it a fast type of processing

The heart of any computer is its central processing unit (CPU). Speed of communication needs to be very fast and no part of the CPU can afford to stand idle. Therefore, the more work you can do on behalf of the application synchronously, the faster the application will run. Any time the CPU needs to suspend work on the application to issue a command to another portion of the system such as the I/O subsystem or inter-system messaging, it slows the application down. Additionally, the CPU has a portion of memory, called cache, that it uses as a placeholder for information that it needs. Cache is a relatively small workspace, so if it contains something that is not valuable to the CPU at a particular point in time, out it goes.

Parallel Sysplex uses CPU-synchronous processing to communicate read/write interest and bufferpool contents to the Coupling Facility, thus making it a fast type of processing. As mentioned earlier, the XCF/XES provides the services needed for this communication at a high speed. Due to this speed, there is

Highlights

the additional benefit of keeping the application information available in the CPU cache, avoiding the necessity of having to re-retrieve it. Because hardware and operating system components are the foundation of the Parallel Sysplex architecture, the performance and small amount of overhead of this type of clustering architecture is impressive.

benchmarks published by Oracle show a 20% degradation every time you double the number of nodes.

Most workloads will encounter less than 15 percent overhead when going from 1-way to 2-way data sharing. Subsequent members can be added with less than one percent overhead. The method used by Oracle to do this type of communication, described later, can't come close to the performance or the low overhead. In fact, benchmarks published by Oracle show a 20% degradation every time you double the number of nodes. With this type of scaling, overall performance will degrade once you begin adding nodes above an eight to twelve node base.

Complexity in architecture leads to some complexity in recovery if any member of a system should fail. Parallel Sysplex architectures mitigate this by providing multiple paths to the data and redundancy in the Coupling Facility. Since the Coupling Facility is the manager of the Parallel Sysplex environment, the capability of having a second Coupling Facility is built into Parallel Sysplex technology, without sacrificing performance.

The very nature of Parallel Sysplex architecture lends itself to failover processing.

The very nature of Parallel Sysplex architecture lends itself to failover processing. If several CPUs can access the same information, transactions against the data can be rerouted to another CPU. Front-end load-balancing and failover is provided by DB2 Connect's connection concentration. Further protection can be obtained by utilizing tools such as WebSphere Edge Server.

However, if a failure should occur, a failed member needs to be recovered. The length of time that a system takes to recover is dependent on the method it uses to externalize database changes and the process that needs to occur when recovery is started. This information is essential when planning recovery strategies.

Externalizing changes generally means writing the information to some form of disk storage. The frequency of the externalization needs to be balanced against the performance of a system. Too frequent and performance suffers. Not frequent enough and recovery suffers. DB2 doesn't externalize data to

Highlights

disk during a transaction, but rather "bundles" a number of transactions and externalizes changes in bulk. When Parallel Sysplex is involved, the Coupling Facility is responsible for insuring that the individual members don't have too great a backlog of changes to externalize. Using the Coupling Facility, DB2 can clean buffers much sooner with minimal performance impact and thus achieve better recovery times as compared with OPS/RAC which must use expensive disk I/O to clean buffers. Additionally, the Coupling Facility insures that changes to the database are coordinated, with no need to transfer updates between members before externalizing changes. Hence, there is no need to coordinate between members during recovery, thus shortening the time it takes to be up and running again.

The beauty of this technology is the ability you have to add maintenance to one member at a time to keep your systems up and running while keeping up to date on service.

One more note about availability and Parallel Sysplex: any software you have requires maintenance. The beauty of this technology is the ability you have to add maintenance to one member at a time to keep your systems up and running while keeping up to date on service. The technology of Parallel Sysplex allows you to roll in a new release, one member at a time. You can keep your data sharing group running with an outage of only a few minutes -- even across a release migration.

Oracle Real Application Clusters

Oracle has embraced the shared-disk clustering model based primarily on their initial attempt on DEC VAX, but with some significant differences from the Parallel Sysplex model -- differences that we believe affect the performance of applications distributed on the Oracle model. But, before we look at the implementation of the clustering model, let's spend some time reviewing how Oracle performs locking in general.

Oracle Locking

In contrast to DB2, Oracle doesn't acquire read locks when they process a SELECT statement. Instead, they guarantee that the application will always see the data as it existed when the OPEN CURSOR was performed. If they don't acquire locks and yet still allow changes to the data, how can they make that guarantee?

During FETCH, Oracle must examine each row that was previously part of the OPEN to determine if it was changed between the time it was opened and the time it was fetched. If the row was modified by another user, Oracle has

Highlights

. . . as more transactions access the same page, this layer grows in size. This means there are fewer rows per page and more disk storage is needed for the same information.

to search through rollback segments to find the row image that existed during the OPEN. In a high-volume, high-concurrency update workload, this algorithm can introduce extremely high CPU and I/O costs.

How does Oracle determine that a row is in use by another system if it doesn't use locks? Well, in truth, Oracle does have a locking mechanism; it's built into the data page (called a block in Oracle), rather than being handled separately as is done in DB2. The lock structure on the Oracle data page is kept in a section of the header called the transaction layer. This section is variable in size and will grow as more transactions simultaneously access records on the data page. Every time a transaction accesses a row on a data page, an entry is put in the Interested Transaction List (ITL) on that page, using up 24 bytes per entry. Thus, as more transactions access the same page, this layer grows in size. This means there are fewer rows per page and more disk storage is needed for the same information.

This approach has several drawbacks.

The first drawback has already been mentioned – each transaction that accesses data requires 24 bytes on every page that it touches, using space that could be reserved for data. Oracle provides a MAXTRANS parameter on each table to limit the growth of this transaction layer, but that means that some transactions will need to wait until an ITL slot opens up. To do that, it goes to sleep, periodically wakes up and checks and goes back to sleep if an ITL slot is not available. If a slot opens up while it's asleep, another transaction may grab the slot. This means there is no guarantee that a transaction can acquire the lock it wants and locks are not allocated to the transactions in the order they are requested.

It is difficult for the DBA to accurately predict how much space a table will need. If a table is "hot," the transaction layer can grow exponentially and the size does not automatically shrink when a transaction is through with the page. This means the DBA must monitor the use of ITL slots to insure there is room for data.

In contrast, DB2 employs a patented locking mechanism that stores locks in memory.

Highlights

DB2, unlike Oracle, guarantees that a transaction can acquire the lock it wants and locks are allocated to the transactions in the order they are requested.

The way DB2 works is that every lock requested has a name that is stored in a memory area known as the locklist. If multiple applications try to lock the same row for update, then DB2 creates a linked list of these lock requests in memory, not on the data page. In many cases, multiple transactions may share a lock at the same time. When an exclusive update lock is requested on a row that already has a lock, the transaction waits until the current locker releases the lock and sends a notification message to the next transaction on the queue.

This means that DB2, unlike Oracle, guarantees that a transaction can acquire the lock it wants and locks are allocated to the transactions in the order they are requested.

Oracle Real Application Clusters (RAC)

In 1996 Oracle announced Oracle7 Parallel Server which included enhancements to their use of a clustering architecture. They stated that "Oracle7 Parallel Server advances Oracle7 server scalability and reliability for OLTP and decision support applications by exploiting the processing power of clustered...computers." They went on to announce that the product efficiently coordinated "OLTP processing activity across nodes to provide a seamless computing resource" and that their "cache management technology" provided "unsurpassed performance optimizations for your clustered...systems." Two key features of this technology were that it minimized messaging due to remote calls to Lock Manager and it reduced I/O.

Parallel Server didn't particularly catch on with Oracle customers and in 2000, they announced Real Application Clusters (RAC). In their white paper they stated that their Cache Fusion architecture "provides flexible and effortless scalability for e-business applications." They noted that "traditional shared disk databases, synchronizing database resources across the cluster database poses significant challenges with regard to achieving cluster scalability." Two key breakthroughs of this technology were that it minimized messaging due to remote calls to Lock Manager and it reduced I/O. (Sound familiar?)

Let's take a look at how Distributed Lock Manager (DLM) works contrasted to the Coupling Facility of IBM's Parallel Sysplex technology. Much of the

Highlights

excessive performance overhead of Oracle's DLM stems from the fact that, as the name implies, the DLM is distributed in nature, meaning that the locks to be managed are distributed amongst the Oracle instances, as in a shared-nothing architecture. This must be done because the platforms that Oracle runs on are shared-nothing hardware architectures, and there is no Coupling Facility equivalent which would allow Oracle to efficiently share its locks. It is this unnatural mapping of a shared-disk DBMS architecture on top of shared-nothing hardware architectures that causes Oracle's shared-disk approach much of its difficulties.

If you'll recall, Oracle must examine each row during FETCH to determine changes and if changes are made, retrieve the original row from the rollback segments. In a clustered environment, this search has to include rollback segments from all instances (the Oracle name for members).

RAC's Parallel Cache Management (PCM) locks are allocated on a per file basis. The user must determine the optimal number of PCM locks for each file based on its intersystem access characteristics. For example, if a file has 1000 pages and 100 PCM locks are defined, then each PCM lock covers ten pages. For our example, let's assume that there are two instances, Instance1 and Instance2 accessing the same file. PCM Lock1 covers pages one to ten. If Instance1 updates the first page, it acquires PCM Lock1 that covers pages one through ten. If Instance2 wants to update the second page, a "false ping" occurs and intersystem messaging is needed to transfer the updated pages and the PCM Lock1 from Instance1 to Instance2. In this case, each page is transferred directly from one instance to another through intersystem messaging and perform log-based disk I/O to record the page ownership.

. . . with Oracle, the fewer locks you have, the more "false pinging" will occur and the more data will need to move between systems. This is detrimental to performance.

PCM locks consume memory in the Oracle System Global Area (SGA) which is their global shared memory. The more locks you allocate, the more memory is required. However, with Oracle, the fewer locks you have, the more "false pinging" will occur and the more data will need to move between systems. This is detrimental to performance.

The only way to avoid this type of performance degradation is to partition the database and assign the portions to different instances, similar to a shared-nothing architecture. The application must then be rewritten to be "cluster aware." This imposes a heavy cost to the customer as well as removing manyof the benefits of clustering, including availability.

Highlights

If you extrapolate these numbers, you can see that adding servers to the Oracle solution will actually decrease your performance power from the six that you have.

ITG Study

In the summer and fall of 2000, the International Technology Group (ITG) performed a study of scalability for e-business performance. These findings were reported by Brian Jeffery during the International DB2 Users Group (IDUG) in 2001.

They focused on four types of applications -- front-end (online purchasing and Web self service) and back-end (billing and accounts receivable, and general ledger posting) systems. They studied 15 exceptionally high-volume corporate users, using both DB2 for z/OS on Parallel Sysplex and Oracle on the Sun Solaris operating system. The Parallel Sysplex environment software included DB2, CICS/ESA, COBOL on the OS/390 operating system on IBM G6 hardware. The Oracle application consisted of Sun E10000 servers with 400 MHz CPUs, Oracle 8i, BEA Tuxedo and Veritas Cluster software on a Sun Solaris operating system.

The workload consisted of over 58,000 concurrent users for online purchasing, close to 68,000 concurrent users for online self service, over six million online billing and accounts receivable lines an hour and over seven and a half million general ledger lines per hour.

In order to run the workload, the Oracle solution required 45 percent overhead with six nodes (e.g.. a cluster with six servers). When adding up the overhead, it means you would need to purchase six servers to get the performance power of a little over three (0.55 x 6 = 3.3). The IBM solution required less than 14 percent overhead using three members, with an incremental increase of less than one percent for each additional server.

If you extrapolate these numbers, you can see that adding servers to the Oracle solution will actually decrease your performance power from the six that you have. Based on this example, the IBM solution will obtain greater performance levels with more scalability.

Highlights

IBM's DB2 UDB for OS/390 and z/OS using Parallel Sysplex technology is superior in terms of day-to-day performance, availability even during normal maintenance, scalability and performance during recovery.

Conclusion

Cluster technology was developed to increase the availability and scalability of applications that access the multiple terabytes of data that companies now store. Yet, not all clustering technologies are the same – particularly in the area of performance, and particularly when data is frequently updated as in most mission critical OLTP applications. This document has covered how IBM's DB2 UDB for OS/390 and z/OS using Parallel Sysplex technology is superior in terms of day-to-day performance, availability even during normal maintenance, scalability and performance during recovery. DB2 data sharing and zSeries Parallel Sysplex have been running mission critical applications for nearly a decade, and the solution has earned the trust of customers throughout the world, across nearly all industry segments. This trust has been earned through solving real world customer problems, not through hyperbole or glitzy marketing campaigns. It's a mature solution that continues to evolve to meet the ever increasing demands of customers.

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Printed in United States of America
8-02