IBM @server xSeries 445 Demonstrates Scalability to 32-Way Running a Business Intelligence Workload

Tricia Thomas
IBM eServer xSeries Performance
Research Triangle Park, NC USA

Abstract

The audience for this paper is IBM clients who are interested in the IBM® eServer® xSeries® high-performance scalable server line, specifically the xSeries 445. This paper discusses optimization techniques used during the running of a Business Intelligence workload on a 32-way x445. After highlighting IBM Enterprise X-Architecture™ technologies that enabled the x445 to scale to 32-way, the paper presents the measurement results and explains tuning techniques that can be applied to high-end xSeries servers to optimize performance for Business Intelligence workloads.

Overview of the x445 Server's Enterprise X-Architecture

In September 1997, IBM announced X-Architecture, a strategy to take the technologies used in IBM's mainframe computers and incorporate these same technologies in its industry-standard, Intel® processor-based servers. Since then, this strategy has evolved into Enterprise X-Architecture (EXA), which has defined the means for delivering unmatched performance, scalability, and availability in an industry-standard Intel processor-based server.

The x445 server showcases Enterprise X-Architecture technology. Initially scalable from 2- to 16-way with the first-generation x440, the second-generation x445 now delivers on IBM's commitment to scale to 32-way.

Several key technologies in IBM's Enterprise X-Architecture roadmap that have enabled the x445 to scale from 2- to 32-way include:

- Modular building-block design
- XpandOnDemand™ scalability
- High-speed coherent interconnection of nodes
- High-speed remote PCI-X I/O support

Other aspects of IBM's Enterprise X-Architecture, including Active Memory™, XceL4™ Server Accelerator Cache, and Copper Diagnostics™, improve the x445's performance, reliability and availability.

Modular building-block design

As a result of continual industry focus on increasing processor speed, the traditional symmetrical multiprocessing (SMP) design was fast approaching its design limits because processor performance gains could not be matched by single SMP memory controller throughput. Highly optimized processors were forced to wait for other less-optimized subsystems (e.g., memory and I/O) to complete requests, limiting overall system performance. Prior to the xSeries 440, eight-way SMP servers were not achieving expected scalability because the supporting subsystems were unable to handle the increased frequency of processor requests.

IBM saw this trend in 1997 and began to move toward today's EXA design of the 4-way SMP scalable node. In this design, each node contains not only the four SMP processors, but also a memory controller, the memory DIMMs, XceL4 System Accelerator Cache, and I/O support for the processors. This design allows the node to work independently from other scalable nodes in the system, thereby reducing memory access queue lengths and increasing performance. In effect, each set of four processors has its own dedicated, high-speed support staff, which reduces the time that the processors are kept waiting and can help processors realize their potential.

One byproduct of the technology that allows two or more scalable nodes to operate as a single SMP server is Non-Uniform Memory Access (NUMA). Unlike traditional SMP systems, the amount of time required to access memory is non-uniform; that is, the local memory can be accessed more quickly than memory that is local to another scalable node in the system.

Even with longer remote latencies, most NUMA systems often outperform modern SMP designs under heavy load because the multiple memory controller design of NUMA better distributes the memory-access load and reduces memory queuing time. However, to function optimally, NUMA systems require modifications to the operating system. In most cases, it is not necessary to implement application software modifications, but to achieve ultimate performance, APIs are available for database applications to fine-tune operation on NUMA-based systems.

XpandOnDemand scalability

XpandOnDemand™ scalability provides customers with an easier, more affordable path to higher performance using industry-standard hardware. Each x445 can scale from 2-way to 8-way in a single chassis. Growing to 16-way can be done simply by connecting two 8-way chassis; connecting four chassis allows scaling up to 32-way.

In addition to the performance benefits gained from the EXA chipset design, the 4-way SMP scalable node allows clients to purchase only the processing power required initially and then add nodes as requirements grow. XpandOnDemand technology allows the client to pay as the system grows, rather than make a large, up-front investment in anticipation of future growth.

High-speed coherent interconnection of nodes

Each x445 can house up to two 4-way SMP scalable nodes. To enable multiple nodes to work together under the control of a single operating system image, IBM has developed a high-speed connection between the nodes known as the SMP expansion port. Data travels across this connection at 3.2 GB/sec, which is the same throughput rate as the processor front-side-bus[1]. These SMP expansion ports are used to connect up to four x445 systems, allowing processor scalability to 32-way. The high-speed interconnect between the 4-way scalable nodes allows

[1] The front-side-bus is the interconnect bus connecting four Intel Xeon MP processors to a shared memory controller. This bus runs at 400MHz and is 8-bytes wide and delivers peak throughput of 3.2GBytes/Sec.

each processor in the system to have high-speed access to the data and resources local to the other processors in the system.

High-speed remote PCI-X I/O support

The x445 incorporates the latest industry standard PCI bus architecture known as PCI-X, which provides for more efficient I/O data transfers, more adapters per I/O bus segment, and faster I/O bus speeds. Together these enhancements, along with PCI-X-enabled adapters, translate into higher system I/O throughput. One limitation of most PCI-based systems, however, is that all PCI adapter slots are located inside the server itself. Because of this, reduction in server size often comes at the expense of PCI-X adapter slots. In an environment where a large number of adapters are a necessity (e.g., one with high-performance database servers), a limit on PCI slot count and the resulting limit on I/O capacity become a problem.

To increase I/O capacity, IBM integrated the Remote I/O (RIO) technology, developed for its pSeries and iSeries line, into the x445 server. An RIO port enables the x445 to access up to 12 additional Active PCI-X adapter slots by attaching the optional RXE-100 Remote Expansion Enclosure. The RXE-100 is connected to the x445 using a 2 GByte/second dedicated I/O port.

The attributes described above are key to enabling the XpandOnDemand scalability demonstrated by the x445. See *Introducing IBM Enterprise X-Architecture Technology* for more information about Enterprise X-Architecture. [1]

Validating 32-Way Scalability

Mid-year 2003, IBM released a statement of direction to make the x445 scalable to 32-way. To validate 32-way scalability, IBM used a Business Intelligence workload, running on Microsoft® Windows® Server 2003 Datacenter Edition and IBM DB2® UDB 8.1.

Even before the 32-way development began, the xSeries and DB2 performance teams went to work on an 8-way x440, and later the 8-way x445, to determine the best way to take advantage of its scalable design. The team discovered that maximum database performance could be obtained by leveraging the modular design of these systems and defining a multiple-node logical cluster on top of a single operating system image. This and several other tuning techniques are discussed later in this paper.

Building on the lessons learned in an 8-way environment, the system was scaled up to 32 processors by connecting four 8-way x445 systems together. The 32-way x445 demonstrated a scalability of 2.76 times the performance of an 8-way x445 using a Business Intelligence workload.

Optimizing Performance on the x445 Server

The scalable design of the x445 has allowed it to break through the performance barriers typical of many Intel processor-based SMP systems. Maximizing the performance of this modular design requires:

- Balancing the hardware configuration so that each processor has access to an equal amount of resources
- Affinitizing applications and threads to the processors that initiate them
- Localizing data access

Balancing the hardware

Integral to the scalable design of the x445 is its ability to adapt the configuration of the system based on the number of attached scalable nodes (SMP Expansion Modules). When planning the system configuration, it is important to ensure that each SMP expansion module has access to an equal amount of system memory and I/O. Consideration should also be given to the placement of memory and I/O adapters in the system. Ideally the memory type and placement for each SMP expansion module will be the same. Likewise, each SMP expansion module should have direct connectivity to an equal number of I/O adapters that are installed in PCI slots that have similar performance characteristics. It is often the case that an I/O adapter will operate at a reduced speed when installed in a slower-speed PCI slot; when possible, this should be avoided. Note that for most standard commercial environments, the system workload is only a fraction of the workload run on systems in IBM's performance lab. Small configuration imbalances will not be detected, and the reduced performance that results from a less than optimal PCI adapter configuration is often not noticeable. These tuning recommendations are what we have found to be optimal for very heavy workloads where the highest level of performance is desired.

Affinitizing the workload

Microsoft Windows Server 2003 has been enhanced to provide optimal performance on the x445. To understand the location of processors and memory within the system, the operating system builds a Static Resource Affinity Table. The operating system uses the information in this table to direct a process to run on the processor that initiated it using memory local to the initiating processor. Processor and memory affinitization greatly reduces the amount of remote traffic within a system and optimizes system performance.

Localizing the data

One key to optimizing the x445's performance is localizing the buffer cache, memory and disk resources to the owning processors. A natural follow-on requirement is that the data accessed by the x445's processors should be stored and accessed locally as well.

One way to satisfy this requirement is to partition data across the system in such a way that it remains local to processors that are most likely to access it. In our lab environment, we leveraged DB2's shared-nothing architecture to accomplish this.

Localizing data reduces the frequency of I/O requests serviced by remote processors. The additional latency incurred by a remote request does not affect perceived performance unless the system is under a very heavy load. For environments where it is not possible or desirable to define a partitioned database, system processor utilization should be monitored to confirm that no single processor in the system is running near 100-percent utilization. With sufficient processor headroom, remote I/O requests can be serviced without a noticeable impact on performance.

Optimizing Performance for Business Intelligence

The type of application used will determine the hardware options and software configuration chosen for the x445. The focus of this paper is tuning systems for Business Intelligence workloads. Business Intelligence is the output of Decision Support applications, which provide data-mining capability needed to support a decision-making process. When tuning high-performance servers for Decision Support applications, the first priority is to maximize system throughput. When IBM designed the 8-way x445, the foundation was created for a server capable of 1.1 GB/sec of sequential I/O.[2] The challenge for IT professionals is to configure an I/O subsystem that maximizes system throughput while staying within budget.

Maximizing throughput on the x445 can be approached from two angles: hardware and software. When choosing hardware, it is important to consider memory, I/O adapters, disk enclosures, disk drives, and possibly I/O expansion units. When choosing software, it is important to select operating system and database software that leverages the design of the x445 and integrates well into your environment. For this evaluation, IBM chose Microsoft Windows Server 2003 Datacenter Edition and IBM DB2 UDB 8.1. In the following sections, we discuss the reasoning behind our optimization choices.

Once the hardware and software are installed, but before installing your application, it is important to confirm that your x445 is balanced. Often, small adjustments in the hardware configuration can have a significant impact on performance. This paper recommends tools that we use to verify a balanced configuration before we begin the performance tuning process.

Choosing hardware options to maximize system throughput

To maximize performance of a Business Intelligence workload on the x445, we made the following selections:

- RXE-100 Remote I/O expansion enclosure
- IBM ServeRAID-6M Ultra320 SCSI controller
- 36.4GB 15K RPM Ultra320 SCSI disk drives
- Ultra320 SCSI disk enclosures
- 512MB memory DIMMs

Because we desired optimal performance, we chose the highest-performing products available. Other choices will work well provided they do not become a performance bottleneck for your application. Note that at some point, nearly every system will have a bottleneck. The key is to understand where the system bottleneck is and alleviate it if possible.

I/O expansion enclosure

An I/O expansion enclosure should be added to your configuration if six PCI adapter slots do not provide sufficient connectivity, or if it is necessary to balance hardware resources in your system.

When SMP expansion modules are added to the x445, connectivity of I/O slots in the system changes. The following diagram, reproduced from the *IBM eServer xSeries 445 Planning and Installation Guide* [2], illustrates the change when a 4-way x445 grows to an 8-way.

[2] 1.1GB of sequential throughput measured using DB2 on an 8-way x445 performing 384KB sequential reads.

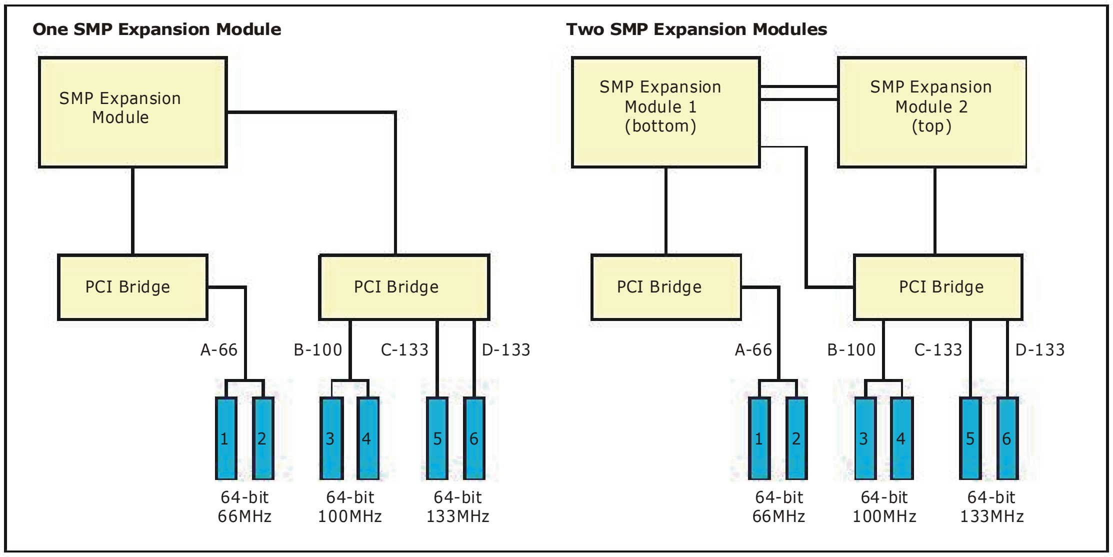
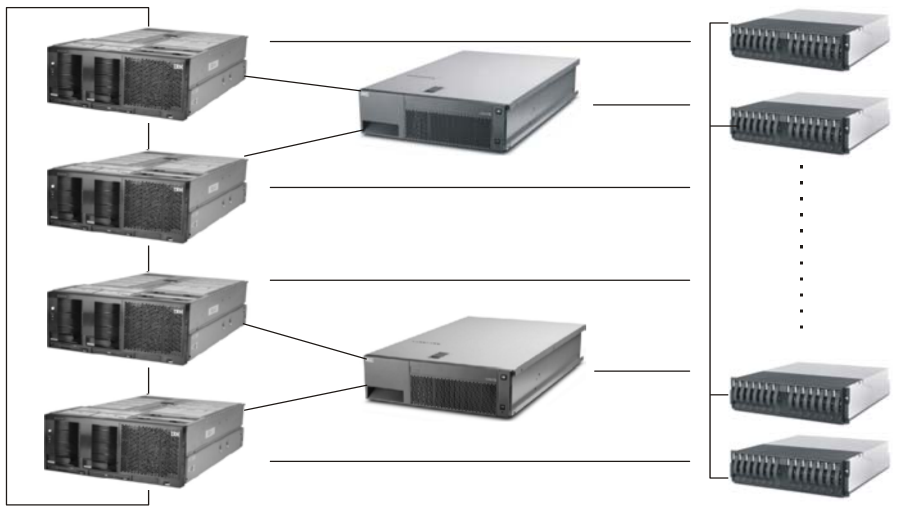


Figure 2. x445's PCI-X Slots with One and Two SMP Expansion Modules

When the x445 is configured with one SMP expansion module, slots 1-6 are directly connected. When the second SMP expansion module is connected, slots 1 and 2 have direct connectivity to the first (bottom) SMP expansion module, and slots 3-6 have direct connectivity to the second (top) SMP expansion module. This changes again when two 8-way x445 systems are connected to form a 16-way SMP server. In this configuration, internal slots (1-6) are local to the first (bottom) SMP expansion module. To provide six local slots for the second (top) expansion module in each 8-way, an IBM RXE-100 Remote I/O Expansion Enclosure must be connected to the system. Connectivity of the slots is the same for 16-way and 32-way x445 systems.

The RXE-100 provides high-speed access to an additional 6 or 12 PCI-X slots. Configured in two "six-packs," each pair of adapter slots supports either two 100MHz PCI-X adapters or one 133MHz PCI-X adapter. Each six-pack has its own high-speed interconnect enabling one RXE-100 to be shared by two x445 servers. To get the maximum speed of 133MHz, install only one adapter per pair of I/O slots (for a total of three adapters). When the fourth, or any subsequent, adapter is required, the additional throughput provided by this adapter will result in a net gain in performance, even though the shared bus will run at the reduced speed of 100MHz.

Figure 3 is a diagram of the 32-way x445 configuration. A horizontal line drawn through the center of the 32-way configuration diagram yields an illustration of two 16-way x445 environments.



Each of Four x445s	Each of Two x RXE-100s	Each of 16 Pairs of Logical Nodes
8 x 2.8GHz/3MB Xeon	4 x ServeRAID-6M Adapter	2 x EXP400
16 x 1GB DIMM		28 x 36.4GB Drive
2 x ServeRAID-6M Adapter		
2 x 36.4GB Drive (in one x445)		
1 x Gigabit Ethernet (in one x445)		

Figure 3. Diagram of 32-Way x445 Configuration.

I/O adapter

When choosing an I/O adapter, it is important to understand the type of I/O required by the application that will run on your system. Will the system be completing large numbers of small (4KB) random I/O requests, or will it be processing large (64KB) sequential I/O requests? This can only be determined with a detailed understanding of your operating environment. If the application is already up and running, it is a simple matter to use Windows System Monitor to determine the disk I/O request size and match that size in the new configuration. If the system is a completely new installation, then some investigation or assumptions must be made.

Fibre Channel technology is the preferred choice for systems requiring connectivity of large numbers of disk drives per I/O adapter. With Business Intelligence workloads, maximizing disk throughput is key. The dual-channel IBM ServeRAID-6M Ultra320 SCSI Adapter provides sufficient capability to maximize system throughput, while allowing for sufficient disk connectivity to maximize its performance. The IBM ServeRAID-6M adapter enabled us to maximize system performance while minimizing total system cost.

When tuning disk I/O at the controller level, stripe size and RAID level need to be considered. Decision-support workloads are characterized primarily by large sequential reads of database data. Disk-write activity makes up a small percentage of the total workload, and usually consists of smaller I/O sizes (16KB). Given this, a 64K stripe size is recommended. The correct stripe size for your application running on an existing system can be determined by monitoring the Avg. Disk Bytes/Read and Avg. Disk Bytes/Write counters for the Physical Disk object using Windows System Monitor. Having an optimized stripe size reduces additional disk I/O operations that occur when the stripe size is too small, and the reading of unnecessary data that occurs when stripe size is too large.

The choice of RAID level has a direct impact on performance and disk capacity. When ranked according to sequential I/O throughput for read operations, RAID levels 0, 50, 10, 5, 5E, and 1 are the top performers. When RAID protection is required in a Business Intelligence environment, IBM recommends RAID level 50. RAID-50 is known as a composite RAID level.

With RAID-50, the disk array is broken into sets of RAID-5 arrays, which are joined together in a single array. This algorithm provides the highest throughput in a sequential-read environment. See the IBM Redbook, *Tuning IBM eServer xSeries Servers for Performance, for additional information*. [3]

Disk drive technology

When you choose a disk subsystem solution, keep in mind your application's I/O requirements, which should help you determine the type of disk drive and quantity of disks you purchase. Ensure that the performance characteristics of the disk and I/O adapter can support your application's I/O requirements. The most common bottleneck found in client environments results from attaching too few disk drives to the system. In this case, you can expect system behavior characterized by low CPU utilization and high disk latencies, due to queuing a large number of disk I/O requests to too few disk drives. When this happens, the super-powerful processor complex is forced to wait on an overworked disk I/O subsystem.

Ultra320 SCSI technology was used because it significantly outperforms Ultra160 technology for sequential workloads. Ultra320 technology doubles the per-channel throughput that can be achieved on a single channel of a SCSI adapter. Also, the Ultra320 drive provides advanced disk technology, which increases its throughput from 40 MB/sec to 70 MB/sec.[3]

In our measurement environment, twenty-four 36.4GB U320 15K RPM disk drives were connected to each I/O adapter for database data, and four 36.4GB U320 15K RPM disk drives for log data. The database data drives provided sufficient sequential throughput to maximize the performance of the controller.

Several optimizations are shown in the following diagram, which illustrates the configuration of one of several, identically configured, I/O adapters used in our environment.

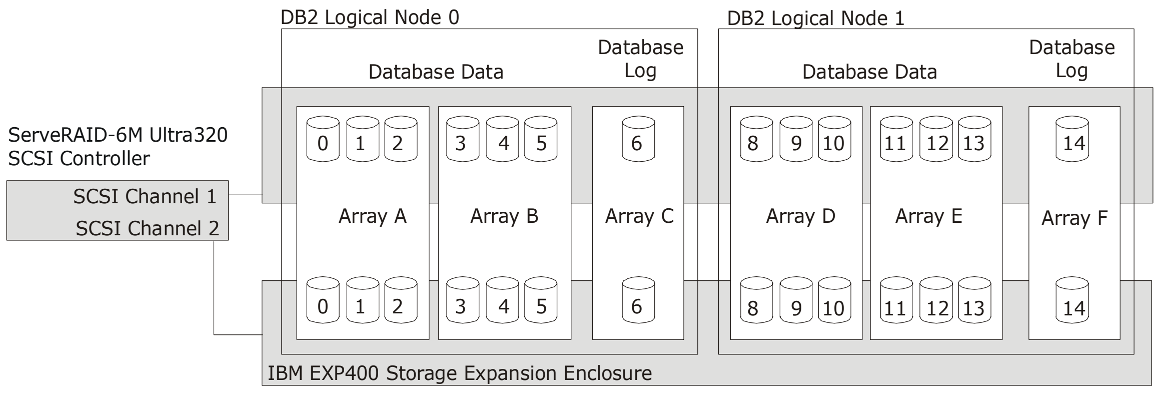


Figure 3. Configuration for Each ServeRAID-6M Controller

The techniques illustrated in Figure 3 can be employed to optimize performance for a standard commercial Business Intelligence environment as well. These techniques include:

- *Isolation of log data on its own redundant, physical disk array*

In any database environment, especially those where data is inserted, updated, and deleted frequently, it is important to isolate log data. This allows the disk arm to remain on the last entry

[3] Sequential 64KB read performance was measured using IOmeter.

in the database log, and eliminates excessive disk-arm movement (latency) that would be required if database data were stored on the same physical disk as the database log.

- *Creation of multiple disk arrays per controller*

Indicators of drive performance include latency (the amount of time required to access and read data from disk) and throughput (the total amount of data that can be read from disk in a defined time interval). Adding disks to a disk array reduces latency by reducing the amount of data stored on each disk and enables the I/O controller to service multiple data requests in parallel. For Business Intelligence workloads, which are characterized by sequential data access, optimizing disk throughput is a primary concern. The number of disks per array must be considered to avoid bottlenecks on the disk adapter and SCSI buses. Furthermore, for parallel databases such as DB/2 system throughput is increased by configuring the database with access to multiple logical disk partitions, which enables greater parallelization of data access.

- *Creation of disk arrays across both channels on the I/O adapter*

When the I/O adapter is configured with an equal number of drives attached to each channel, it is optimal to create the physical disk arrays across both channels on the adapter. This optimization allows for additional parallelization of data access.

- *Partitioning of database data across multiple logical nodes*

To localize data access, the database is partitioned into multiple logical nodes. For a discussion of partitioning, see the section titled, “Configuring DB2 Universal Database 8.1.”

Memory

Each x445 SMP expansion module contains 16 DIMM slots, which are spread across two ports, with four banks each. Memory must be added in pairs, so that a memory bank is either fully populated or empty. Equal amounts of memory should be installed in each port, and for optimal performance both ports should be evenly populated. The figure and the table below were taken from the *IBM eServer xSeries Planning and Installation Guide*. [2] They illustrate the slot placement in the x445 and the recommended DIMM installation order. In the 8-way x445, we measured the same performance when then system was configured with thirty-two 512MB DIMMs as when configured with sixteen 1GB DIMMs.

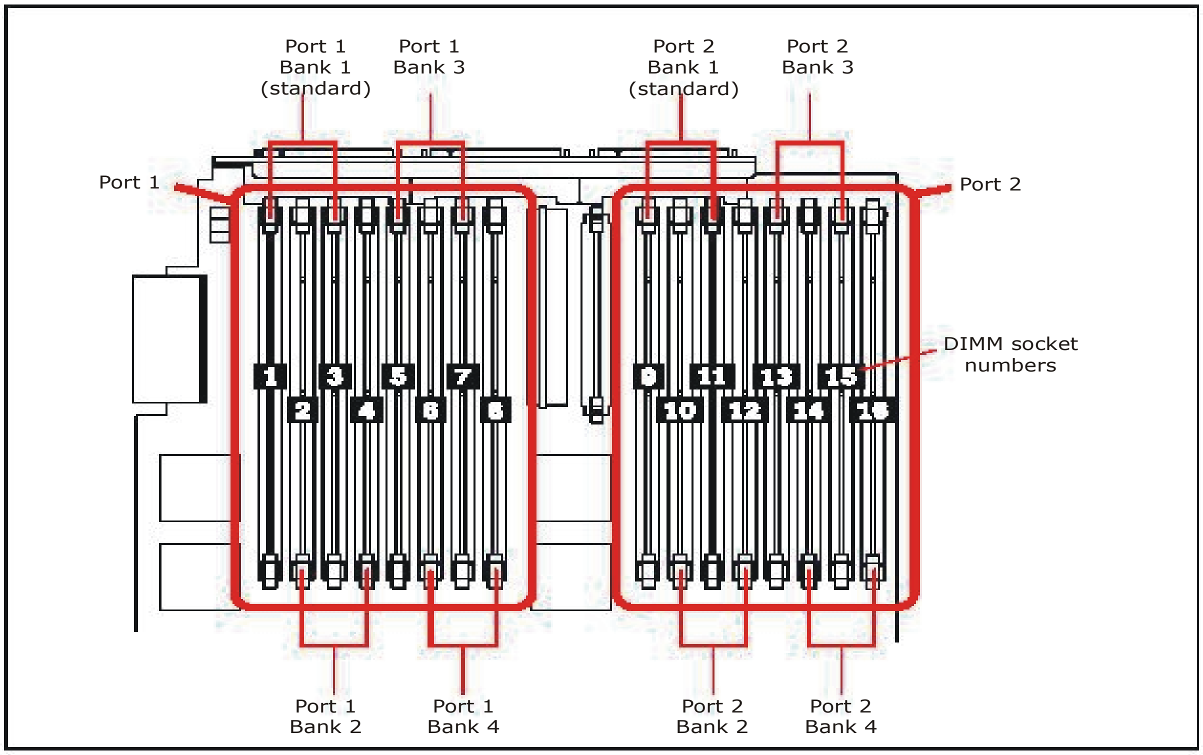


Figure 4. DIMM Sockets on the x445 SMP Expansion Module

Sequence	Port	Bank	Slot Numbers
1	1	1	1, 3
2	2	1	9, 11
3	1	2	2, 4
4	2	2	10, 12
5	1	3	5, 7
6	2	3	13, 15
7	1	4	6, 8
8	2	4	14, 16

Table 1. Recommended DIMM Installation Order

For additional information on tuning memory for the x445, see the *IBM eServer xSeries Planning and Installation Guide*. [2]

Configuring Windows Server 2003

Microsoft Windows Server 2003 Enterprise Edition was used in our 8-way measurement environment. Microsoft Windows Server 2003 Datacenter Edition is more appropriate for 16- and 32-way x445 systems because of its increased memory and processor support, as well as its higher system availability requirements. Both versions handle NUMA processing identically. The NUMA enhancements contained in Windows Server 2003 are enabled automatically. Although Windows 2000 is supported on the x445, it is not recommended for 8-way configurations and above when optimal performance is desired. The reason is that although Windows 2000 is NUMA-compatible, it does not contain NUMA enhancements.

On Intel's 32-bit architecture, the maximum addressable memory space is 4GB, which is typically divided into 2GB for the operating system and 2GB for the application. The Windows operating system has delivered various methods for extending the application address space: first to 3GB for enabled applications, and then beyond 4GB with the introduction of the /PAE switch.

The following illustrates the setting of /PAE and /3GB in boot.ini

```
[boot loader]
timeout=15
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise /PAE /3GB" /fastdetect /PAE /3GB
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise" /fastdetect
```

Enabling Hyper-Threading on the x445 8-way configuration caused the operating system and applications running on it to see each physical processor as a pair of logical processors. These additional logical processors increased the overall system processing capability allowing two threads to run concurrently on a single physical processor. In an 8-way Business Intelligence environment, we have seen throughput gains as high as 13% when Hyper-Threading technology is enabled compared to when it is disabled. Hyper-Threading was not enabled on the 32-way x445 because the maximum number of processors supported by Microsoft Windows Server 2003 Datacenter Edition is 32.

See the Red Paper, "Windows Server 2003 Datacenter Edition on the IBM eServer xSeries 445," [5] for more information on configuring Microsoft Windows Server 2003 on the x445 server.

Configuring DB2 Universal Database 8.1

IBM DB2 UDB is a mature, highly optimized relational database management system. It is ideally suited for running Business Intelligence workloads on the x445 because it is designed for leadership performance in partitioned (clustered) environments and best leverages the x445's Enterprise X-Architecture technology. DB2's partitioned database environment also enables breaking through the 4GB memory barrier that exists for other database management systems on the IA-32 platform.

The following sections discuss important tuning considerations when optimizing Business Intelligence workloads on the x445.

Shared-Nothing architecture

DB2's Shared-Nothing architecture makes it possible for data that is partitioned across multiple nodes in a cluster to be processed collectively to produce a single query result. In a shared-nothing cluster, the data on one node in a cluster environment cannot be seen or processed by processors located in another node. To satisfy a query against a shared-nothing database, DB2 divides that query into parts that can be executed on each node in parallel. When the results from the query are returned to the managing node in the cluster, they are aggregated and returned as a single result to the user.

This type of partitioned database is ideally suited for the x445's scalable design. DB2 is designed to manage data locally and minimize traffic between the nodes in a cluster. Defining a "logical" cluster environment on a single SMP x445 maximizes the server's performance and scalability by minimizing the traffic between scalable nodes in the server. This approach leverages the optimized design of the x445 4-way SMP scalable node, while minimizing latencies incurred by accessing data local to another scalable node in the system.

When a DB2 database is partitioned, each database partition is referred to as a logical node (LN). When defining clustered databases on the x445, we chose to define one LN per processor in the system. On the 8-way x445, we defined 8 LNs; on the 32-way configuration, we defined 32 LNs.

Breaking through the 4GB memory barrier on 32-bit systems

Each LN of a DB2 partitioned database is run as a separate process with its own address space. For the 8-way x445 analysis, we configured the system with 16GB of memory and defined 8 LNs, each with the traditional 2GB application address space. Similarly, we scaled the 32-way to 64GB of memory and 32 LNs. Configuring the system in this way allows DB2 to directly access memory with the shortest path length. In both configurations, the /PAE switch was used to access memory above 4GB. The /3GB setting was not evaluated because each LN had access to 2GB of physical memory. The contents of boot.ini as used in our environment are
shown below:

```
[boot loader]
timeout=15
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise /PAE" /fastdetect /PAE
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Windows Server 2003, Enterprise" /fastdetect
```

See the Redbook, *Tuning IBM eServer xSeries Servers for Performance* [3], for additional information about extended memory addressability.

Optimizing disk I/O

Two types of table spaces can be used to store database data on disk: system-managed space (SMS) and database-managed space (DMS). With SMS, database data is stored in operating system files. With DMS, data storage is managed by the database software (in this case, DB2). DMS is the preferred method for Business Intelligence workloads because the database software stores data contiguously on the disk. This is especially important for sequential-read performance and for optimal prefetching of data. Because the Business Intelligence workload is primarily sequential, prefetching of data is most often effective for the simple reason that the data a processor needs next has already been copied from disk into memory.

To accommodate all types of workloads, DB2 provides the user with great flexibility for defining how data is stored on disk. The following description introduces several concepts that should be studied before designing a database disk layout.

The first step in setting up a partitioned database is to define the logical nodes (LNs) of the database. Each LN will be affinitized to a processor, which has direct connectivity to a group of hard disks. On these hard disks, a set of DB2 containers (unformatted Windows disk partitions) should be defined. Next a table space is created across the containers, tables are created in the table space, and finally, database data is loaded into the table. Indices, which are also created on a table in a table space may be created before or after database data is loaded into a table.

Defining a table space requires setting three key parameters that directly affect performance of the I/O subsystem: page size, extent size and prefetch size. DB2 stores data on a page. For Business Intelligence workloads, a large page size of 16KB is recommended. In DB2, the smallest allocation unit of space is known as an *extent*. The extent size is defined as the number of contiguous pages to be allocated at one time. DB2 allocates an extent in each database container in a round-robin fashion enabling data to be balanced across the available disk subsystem. DB2 allows the user to define extent size. The extent size chosen for our Business Intelligence environment is based on the following formula:

Extent size = stripe size * number of disks in physical array/ buffer pool page size

When a RAID subsystem is configured, a stripe size is defined. For the 32-way configuration, we defined a 64KB stripe size, an array size of six disk drives, and a 16KB buffer pool page size. Plugging these values into the formula (64*6/16) yields an extent size of 24 (16KB pages). With an extent size of 24, DB2 allocates twenty-four 16KB pages to each database container in a table space before moving to the next one.

The last parameter is *prefetch size*. This should be set to a multiple of the extent size times the number of containers in the table space. In our environment we chose a prefetch size of 48 (16KB pages), which was derived by multiplying the extent size (24) by the number of containers in the table space (per logical node)(2). This means that each time DB2 executes a prefetch operation, it will read 48 16KB pages from disk and store them in memory. Because data stored in this logical node of the database is striped across two containers, DB2 will be able to prefetch 48 16KB pages as two parallel 24*16KB or 384KB I/O operations. Note that 384KB divided by 64KB equals 6, which is the number of drives defined in each of our disk arrays. So during a prefetch operation, DB2 will issue one I/O, which will result in reading 64KB of data from each of 12 disk drives simultaneously. Not only is the necessary data moved into memory before it is needed, it is moved in efficiently with a high degree of parallelism. This reduces unnecessary latency, which lengthens response time. As DB2 processes database data, it is likely to do fewer physical I/Os because data will have already been prefetched into memory.

The following table summarizes the disk I/O optimization concepts.

Concept	**Description**	**Sample Value/Size**
Logical Node (LN)	A partition of database.	1 per processor
Container	An unformatted Windows disk partition where database data will be stored.	n/a
Table Space	Area on disk where database pages are stored. A table space may span the logical nodes of a database.	n/a
System Managed Space (SMS)	Type of table space where database data is stored in files that are managed by the operating system.	n/a
Database Managed Space	Type of table space where database data is stored in unformatted Windows disk partitions and is managed by DB2.	Recommended for performance
Data/Index Page	Area on disk where database data or database index data is stored. Should equal buffer pool page size.	16KB
Buffer Pool Page	An area in memory where database pages are loaded for processing. Should equal Data/Index Page size.	16KB
Number of Disks per Phyiscal Array	Defined at the I/O controller level, represents the number of physical disk drives that are included in a physical disk array.	6
Stripe Size	Defined at the I/O controller level, represents the amount of data that will be written to a single physical disk before moving to the next disk in the disk array.	64KB
Extent	The smallest allocation unit of disk space. Consists of a number of contiguous pages. The extent size chosen for our Business Intelligence environment is based on the following formula: Extent size = stripe size * the number of physical disks in array/buffer pool page size	24 16KB pages
Prefetch Size	The amount of data that will be read into memory when sequential I/O operations are detected. Should be set to a multiple of the extent size times the number of containers in the table space.	48 16KB pages

Table 2. Summary of Disk I/O Optimization Concepts

The code fragments below illustrate the commands referenced earlier. *These code fragments highlight aspects of the creation of a partitioned database environment. They are not intended to be comprehensive and should be used in conjunction with the Microsoft Windows Server 2003 and DB2 UDB 8.1 documentation.*

Prepare disks for database data by creating raw partitions using Windows Server DISKPART utility
When a large number of partitions must be defined at the operating system level, a technique available under the Windows operating system, known as "mount points" can be used. An example of the creation of mount points is shown below.

Make directories
```
D:\mkdir d:\mp
D:\mkdir d:\mp\EMPLOYEE_DATA_14
D:\mkdir d:\mp\EMPLOYEE_INDEX_14
D:\mkdir d:\mp\EMPLOYEE_DATA_15
D:\mkdir d:\mp\EMPLOYEE_INDEX_15
```

Create diskpart input script similar to the following (createPartitions.scr)
```
select disk 14
create volume simple size=1000
assign mount=d:\MP\EMPLOYEE_DATA_14
create volume simple size=400
assign mount=d:\MP\EMPLOYEE_INDEX_14
create volume simple size=800
select disk 15
create volume simple size=1000
assign mount=d:\MP\EMPLOYEE_DATA_15
create volume simple size=400
assign mount=d:\MP\EMPLOYEE_INDEX_15
create volume simple size=800
```

Run diskpart to create partitions
```
Diskpart /s createPartitions.scr
```

Create a Partitioned Database Environment
```
set db2instance=HRINST
db2icrt HRINST  /s ese /u:userid,password /r 80000,80040 /p D:\hrinst
db2ncrt /n:1  /u:userid,password /i:%1 /p:1  /o:ibmserv
db2ncrt /n:2  /u:userid,password /i:%1 /p:2  /o:ibmserv
db2nlist
db2ilist
```

Create Bufferpool for data storage in memory
```
create bufferpool BP16K
  size -1
  pagesize 16k;
```

Create a DMS tablespace
```
create regular tablespace EMPLOYEE_TABLE
  pagesize 16K
  managed by database
  using ( device 'd:\MP\EMPLOYEE_data_D14'          5850M,
          device 'd:\MP\EMPLOYEE_data_D15'          5850M) on node(0)
  using ( device 'd:\MP\EMPLOYEE_data_D17'          5850M,
          device 'd:\MP\EMPLOYEE_data_D18'          5850M) on node(1)
  bufferpool   BP16K
  extentsize     24
  prefetchsize   48
  overhead      3.0
  transferrate  0.40;
```

Create a table in the tablespace
```
CREATE TABLE HRDB.EMPLOYEE ( EMPSERIAL    INTEGER NOT NULL,
                             EMPNAME     CHAR(30) NOT NULL,
                             EMPFNAME  CHAR(30)  NOT NULL,
                             EMPMI CHAR(1)  NOT NULL,
                            …
                                )
    IN EMPLOYEE_TABLE
    INDEX IN EMPLOYEE_INDEXES
    PARTITIONING KEY(EMPSERIAL) USING HASHING
    ORGANIZE BY (  (EMPNAME) );
```

Load data into table

```
connect to hrdb;

load from employee.tbl.1,
        employee.tbl.2,
        employee.tbl.3,
        employee.tbl.4,
        employee.tbl.5,
        employee.tbl.6,
        employee.tbl.7,
        employee.tbl.8 of del
   MODIFIED BY COLDEL| FASTPARSE MESSAGES d:\tmp0\employee.msg
   REPLACE INTO HRDB.employee NONRECOVERABLE CPU_PARALLELISM 16
   partitioned db config mode load_only
   output_dbpartnums (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31)
   part_file_location x:\rawdatadirectory;
commit work;

connect reset;
```

Place database logs on dedicated redundant storage

```
set db2node=0
db2start
db2 update db cfg for HRDB using newlogpath d:\mp\log_0
set db2node=1
db2 update db cfg for HRDB using newlogpath d:\mp\log_1
```

Optimizing memory utilization

DB2 provides the user with great flexibility for deciding how to divide available system memory among different areas within a DB2 database environment. We have already seen how defining a multiple-node logical database allows us to maximize use of memory above 4GB. There are several other memory-tuning parameters. The most important of these is the DB2 buffer pool, which is a memory-resident cache for holding database data. Because accessing memory is much faster than accessing disk, it is important to maximize the size of the buffer pool available to your database application. The larger your buffer pool, the more likely data will remain in the buffer pool for more than one access.

For Business Intelligence environments, start with a buffer pool of roughly one-half the memory available to your application. For the 8-way x445 measurement environment, we defined a buffer pool for each LN to be 60,000 16KB pages or 937.5MB. This is just less than one-half of the 2GB application address space available to each LN.

DB2 also gives you the flexibility to define buffer pool page size. The page size chosen for the buffer pool must equal the page size of the table space that contains database data and index pages that will cached there. Buffer pool page size is defined at the time the buffer pool is created. The following code fragment illustrates creation of a buffer pool.

```
create bufferpool BP16K
  size -1
  pagesize 16k;
```

Another important parameter to consider is sort heap, which is the number of 4KB pages in memory to use for sorting. Memory used for sort heap is separate from memory used for the buffer pool. We selected a sort heap value of 6400 4KB pages for the 8-way x445 Business Intelligence environment.

DB2 configuration parameters can be created at an instance or database level. Database manager configuration parameters are created at the instance level and are the same for all databases within the instance. Database configuration parameters are unique to a database. Buffer pool size and sort heap size are examples of database configuration parameters. The following command sequence illustrates the setting of each of these parameters. See DB2 UDB 8.1 documentation for a complete listing of DB2 configuration parameters.

```
Db2start
Db2 connect to HR
Db2 update db config for HRDB using BUFFPAGE 60000
Db2 update db config for HRDB using SORTHEAP 6400
Db2 connect reset
Db2stop
```

Because the amount of memory for each LN was the same in the 8-way and 32-way measurement environments, their memory-tuning parameters were set using similar configuration values.

Setting processor affinity

In this paper, we have emphasized the importance of localizing resources to the x445 scalable node. Once the database is set up correctly, it is important to ensure that DB2 assigns a logical node (LN) to run on the processor that has a direct physical connection to data belonging to that logical node of the database. Without correct affinity settings, processors might routinely access data owned by processors in other scalable nodes, creating increased, unnecessary traffic and contention across the system. When Hyperthreading is enabled on the 8-way x445, processor numbers 0 through 7 correspond to physical processors, processor numbers 8 through 15 correspond to logical processors. In a DB2 multiple logical node environment, ensure that a LN is affinitized to a single physical processor and its corresponding logical processor. A simple table scan against data owned by a logical node in the system, in conjunction with the performance tab in Task Manager can be used to verify that processor affinity is set correctly. The following script illustrates the setting of processor affinity.See the DB2 UDB 8.1 documentation for additional information on setting processor affinity.

```
rem Set affinities for LN0 -> LN7  for 8 processor system with hyperthreading enabled.
db2set db2processors=4,12  -i hrinst 0
db2set db2processors=5,13  -i hrinst 1
db2set db2processors=6,14  -i hrinst 2
db2set db2processors=7,15  -i hrinst 3
db2set db2processors=0,8   -i hrinst 4
db2set db2processors=1,9   -i hrinst 5
db2set db2processors=2,10  -i hrinst 6
db2set db2processors=3,11  -i hrinst 7
```

Verifying the Configuration

As you configure the x445, it is important to take time to validate that system behavior is what you expected. Using Windows utilities such as Windows System Monitor and IOmeter, you can verify that the hardware configuration is balanced, that processors are controlling the disk drives you expect them to, and that the I/O subsystem is delivering the level of performance throughput you expect for your Business Intelligence application.

Overall, a system performs only as fast as its slowest component. Here are a few examples of how a simple configuration error can result in lost performance:

1. If the 8-way x445 is configured with one I/O adapter with six drives, the I/O subsystem will perform at the maximum combined throughput of those six disks, regardless of the fact that the system and I/O adapter are capable of much higher throughput.
2. If the 8-way x445 is configured so that all physical disk I/O is directly connected to four of the eight processors, then half the time, processors will have to wait additional time for I/O to be returned from other processors in the system.

3. If the 8-way x445 is configured correctly with a partitioned DB2 database, but you forget to set processor affinity, then your DB2 applications (one per LN) may run on any processor in the system. This could result in significant unnecessary traffic across the SMP expansion ports and unnecessary contention in the system.

A few simple tests made early in the configuration process will help ensure that your x445 is configured optimally. These tests will also help you get to know what “normal” behavior on your system looks like. This will also help you to detect problems in the future.

Conclusion

With the completion of the evaluation of a Business Intelligence workload on the 32-way x445, IBM confirmed the performance benefits of its Enterprise X-Architecture strategy and supporting technology. The measurement results demonstrated the x445’s superior scalability.

To achieve this proof point, IBM eServer xSeries teamed up with Microsoft and IBM DB2 to showcase the superior operating system and database environments available for this platform.

Microsoft Windows Server 2003 has been optimized for performance on the x445. Microsoft’s Datacenter Edition provides the scalability, reliability and availability demanded from an enterprise-class, high-performance, mission-critical 32-way server.

DB2 has mastered partitioning of data across multiple servers in a clustered environment. The investment DB2 has made in minimizing the amount of data communicated between nodes has yielded scalability in a clustered environment that is second to none. IBM leveraged this investment in technology to realize the strengths of the x445 design.

While a 32-way configuration showcases the scalability of the x445, a 32-processor system is not required to realize the benefits of the x445 design. The tuning techniques discussed in this paper can be used on 8- and 16-way x445 systems to achieve leadership performance.

In summary, to achieve optimal performance on the x445:

- Configure the system in a balanced way so that each processor has equal access to system resources.
- Affinitize applications and threads to the processors that initiate them.
- Localize data to the processors that access that data.

When optimizing performance for Business Intelligence:

- Choose and configure hardware options to maximize system throughput.
- Use Microsoft Windows Server 2003 Enterprise Edition or Datacenter Edition, which each contain NUMA enhancements that enable localization of data access on the x445.
- Define a DB2 UDB cluster environment on the x445 to enable localization of data access and to maximize utilization of memory above the 4GB boundary.
- Tune DB2 memory and I/O optimization parameters for workloads consisting primarily of sequential read access.

Finally at various points in the process of defining your configuration, take time to measure your system’s performance to ensure that it is exhibiting behavior you expect. Take care to configure the x445 in a way that exploits its modular design. Tools such as Microsoft Windows System Monitor and the publicly available IOmeter (www.iometer.org) can be used to validate system balance, application affinity, workload localization, and optimal system throughput.

Third-generation Enterprise X-Architecture

With each successive generation of Enterprise X-Architecture, IBM more fully delivers on the vision of offering the highest performance, scalability, and availability attainable on an industry-standard server. Inspired by mainframe technologies of the eServer zSeries, IBM xSeries continues to improve in developing new capabilities in partitioning, virtualization, reliability, and systems management. With the planned completion and delivery of the third-generation of EXA servers in 2005, IBM is once again upping the ante on competition and declaring to the marketplace that it is not enough to be a leader in innovation. With the third-generation of Enterprise X-Architecture, IBM is well-positioned to maintain its leadership position in the market for high-performance industry-standard servers. With planned support for up to 32-way including EM64T aka 64-bit extensions, and many other advanced EXA features, third-generation EXA looks to deliver on its strategy: taking industry-standard servers deeper into the enterprise than ever before. These truly have become the ”bet-your-business” servers of the 21st Century, and IBM is the company to deliver them.

Acknowledgements

The author would like to acknowledge the many individuals who contributed to the 32-way x445 performance evaluation, especially the dedicated efforts made by Des Daly (DB2 Performance), Sue Goodwin (IBM Windows System Development), Trish Hogan (xSeries Performance), Joe Jakubowski (xSeries Performance), Chris Karamattas (IBM Windows System Development), and Lui Tang (DB2 Performance).

References and Additional Information

[1] *Introducing IBM Enterprise X-Architecture Technology*, by Mark T. Chapman, August 2001.

[2] *IBM eServer xSeries Planning and Installation Guide* (SG248870), www.ibm.com/redbooks.

[3] *Tuning IBM xSeries Servers for Performance* (SG245287), www.ibm.com/redbooks.

[4] *Enterprise X-Architecture Remote I/O and IBM eServer xSeries Servers*, by Mark T. Chapman, November 2001.

[5] “Windows Server 2003 Datacenter Edition on the IBM eServer xSeries 445,” August 2003 (REDP-3700-00), www.ibm.com/redbooks.

[6] Tuning IBM DB2 Databases for Intel Itanium 2-Based Systems, Guojain Cai and Terry Synch, March 18, 2004, www-106.ibm.com/developerworks/db2/library/techarticle/dm-0403cai/

[7] Disk Subsystem Performance Analysis for Windows, March 2004, www.microsoft.com/whdc/hwdev/tech/storage/subsys_perf.mspx.