vmware®

VMware ESX Server 2

NUMA Support

ESX Server 2 provides memory access optimization for both Intel processors and AMD Opteron processors in server architectures that support NUMA (nonuniform memory access). This white paper provides background on NUMA technologies and a detailed description of the sophisticated NUMA optimizations available in ESX Server 2.

The document contains the following sections:

This white paper is intended for partners, resellers, and advanced system administrators who wish to understand the unique advantages of implementing a VMware ESX Server solution on various NUMA-supported platforms such as those available from Hewlett-Packard, IBM, and Sun Microsystems.

Introduction

As the x86 processor architectures have moved further into mission critical roles in the datacenter, hardware and software vendors have worked to meld the most powerful technologies of the mainframe and RISC UNIX worlds with the compelling economics of commodity platforms and popular operating systems. This white paper describes how two of these enterprise class technologies, the mainframe-style partitioning of VMware ESX Server 2 and the highly scalable NUMA system designs from vendors such as Hewlett-Packard, IBM, and Sun Microsystems complement and support each other.

VMware ESX Server 2 is a thin virtualization layer, which enables mainframe-class virtual machine capabilities on commodity hardware. Because ESX Server provides a secure, high-performance operating environment with advanced resource management capabilities, it is an ideal platform for server consolidation.

NUMA systems are advanced server platforms with more than one system bus. They can harness large numbers of processors in a single system image with superior price/performance ratios.

vmware®

For example, IBM's Enterprise X-Architecture, available in the IBM eServer x440, eServer x445, and eServer x460, as well as the Fujitsu-Siemens Primergy T850 and the HP ProLiant DL 385 and ProLiant DL 585, and Sun Microsystems Sun Fire V202 and V402, provide a NUMA platform to support industry-standard operating systems, including Windows and Linux.

What is NUMA?

For the past decade, processor clock speed has skyrocketed at rates exceeding even the predictions of Moore's Law. A multi-gigahertz CPU, however, needs to be supplied with an enormous amount of memory bandwidth in order to use its processing power effectively. Even a single CPU running a memory-intensive workload, such as a scientific computing application, can find itself constrained by memory bandwidth.

These problems are amplified many times over on symmetric multiprocessing (SMP) systems, where many processors must compete for bandwidth on the same system bus. High-end RISC UNIX systems, with prices ranging into the millions of dollars, often try to solve this problem by building a system bus capable of transferring dozens of gigabytes of data per second. Such a solution, however, is enormously expensive and still limited in scalability.

NUMA is an alternative approach that links several small, cost-effective nodes via a high-performance interconnect. Each node contains both processors and memory, much like a small SMP system. However, an advanced memory controller allows a node to use memory on all other nodes, creating a single system image. When a processor accesses memory that does not lie within its own node (remote memory), the data must be transferred over the NUMA interconnect, which is slower than accessing local memory. Thus, memory access times are "non-uniform," depending on the location of the memory, as the technology's name implies.

NUMA Challenges for Operating Systems

Because a NUMA architecture provides a single system image, it can often run an operating system with no special optimizations. For example, Windows 2000 is fully supported on the IBM x440, although it is not designed for use with NUMA.

However, there are many disadvantages to using such an operating system on a NUMA platform. The high latency of remote memory accesses can leave the processors under-utilized, constantly waiting for data to be transferred to the local node, and the NUMA interconnect can also become a bottleneck for applications with high-memory bandwidth demands. Furthermore, performance on such a system may be highly variable, for example, if an application has memory located locally on one benchmarking run, but a subsequent run happens to place all of that memory on a remote node. This phenomenon can make capacity planning much more difficult. Finally, processor clocks may not be synchronized between multiple nodes, so applications that read this clock directly may behave incorrectly.

Some high-end UNIX systems provide support for NUMA optimizations in their compilers and programming libraries. This, however, requires software developers to tune and recompile their programs for optimal performance, and there is no guarantee that optimizations for one system model will work well on the next generation of the platform. Other systems have allowed an administrator to manually decide the node on which an application should run. While this may be desirable for certain applications that demand 100% of their memory to be local, it creates an enormous administrative burden and can lead to imbalance between various nodes when workloads change.

Ideally, the system software should provide transparent NUMA support, so that applications can benefit immediately without modifications. The system should maximize the use of local memory and schedule programs intelligently without the need for constant administrator intervention. Finally, it must respond well to changing conditions, without compromising fairness or performance, if the system is intended to support uptimes of months or years.

ESX Server NUMA Scheduling

ESX Server uses a sophisticated NUMA scheduler to dynamically balance processor load and memory locality. Each virtual machine managed by the NUMA scheduler is assigned a home node—one of the system's NUMA nodes containing both processors and local memory, as indicated by the System Resource Allocation Table (SRAT). When memory is allocated to a virtual machine, it preferentially comes from the local memory associated with its home node. The NUMA scheduler may dynamically change a virtual machine's home node in order to respond to changes in system load, reducing processor load imbalances and improving memory locality.

Some virtual machines are not managed by the ESX Server NUMA scheduler. For example, if you manually set the processor affinity for a virtual machine, the NUMA scheduler may not be able to automatically manage this virtual machine. Also, virtual machines that have more virtual processors than the number of physical processors available on a single hardware node cannot be managed automatically. Virtual machines that are not managed automatically by the NUMA scheduler still run fine; they simply don't benefit from ESX Server's NUMA optimizations.

The intelligent, adaptive NUMA scheduling and memory placement policies in VMware ESX Server 2 can manage all virtual machines transparently, so that administrators do not need to deal with the complexity of balancing virtual machines between nodes by hand. However, manual override controls are also available, so that advanced administrators can still optimize their systems as they see fit.

It is important to note that these optimizations work seamlessly regardless of the type of guest operating system. ESX Server provides powerful, transparent NUMA support even to guests that do not support NUMA hardware, such as Windows NT 4.0. This unique feature of VMware ESX Server allows users to take advantage of cutting-edge new hardware, even when tied to legacy operating systems.

VMware NUMA Optimization Algorithms

This section describes the algorithms used internally by VMware ESX Server to maximize application performance while still maintaining resource management guarantees. VMware ESX Server system administrators seeking a more detailed discussion of specific NUMA-related configuration options should refer to the documentation or to the "numa (8)" man page in the Service Console.

Home nodes and initial placement

ESX Server 2 assigns each virtual machine a home node when the virtual machine begins running. A virtual machine runs only on processors within its home node, and its newly-allocated memory comes from the home node as well. Thus, if a virtual machines home node does not change, it uses only local memory, avoiding the performance penalties associated with remote memory accesses to other NUMA nodes. New virtual machines are initially assigned to home nodes in a "round robin" fashion, with the first virtual machine going to the first node, the second virtual machine to the second node, and so forth. This policy ensures that memory is evenly used throughout all nodes of the system.

Several commodity operating systems, such as Windows 2003 Server, provide this level of NUMA support, which is known as initial placement. It may be sufficient for systems that run only a single workload, such as a benchmarking configuration, which does not change over the course of the system's uptime. However, initial placement is not sophisticated enough to guarantee good performance and fairness for a datacenter-class system that is expected to support changing workloads with an uptime measured in months or years.

To understand the weaknesses of an initial-placement-only system, consider the following example: an administrator starts four virtual machines and the system places two of them on the first node; the second two virtual machines are placed on the second node. Now consider what happens if both virtual machines on the second node are stopped, or if they simply become idle. The system becomes completely imbalanced, with the entire load placed on the first node. Even if the system allows one of the remaining virtual machines to run remotely on the second node, it suffers a serious performance penalty, because all of its memory remains on its original node.

Dynamic load balancing and page migration

To overcome these weaknesses, ESX Server 2 combines the traditional initial placement approach with a dynamic rebalancing algorithm. Periodically (every two seconds by default), the system examines the loads of the various nodes and determines whether it should rebalance the load by moving a virtual machine from one node to another. This calculation takes into account the relative priority of each virtual machine to guarantee that performance is not compromised for the sake of fairness.

The rebalancer selects an appropriate virtual machine and changes its home node to the least-loaded node. When possible, the rebalancer moves a virtual machine that already has some memory located on the destination node. From that point on (unless it is moved again), the virtual machine allocates memory on its new home node and it runs only on processors within the new home node.

Rebalancing is an effective solution to maintain fairness and ensure that all nodes are fully utilized. However, the rebalancer may need to move a virtual machine to a node on which it has allocated little or no memory. In this case, the virtual machine incurs a performance penalty associated with a large number of remote memory accesses. ESX Server 2 can eliminate this penalty by transparently migrating memory from the virtual machine's original node to its new home node. The system selects a page (4 kilobytes of contiguous memory) on the original node and copies its data to a page in the destination node. Then, the system uses the virtual machine monitor layer and the processor's memory management hardware to seamlessly remap the virtual machine's view of memory, so that it uses the page on the destination node for all further references, eliminating the penalty of remote memory access.

When a virtual machine moves to a new node, ESX Server 2 immediately begins to migrate its memory in this fashion, usually at a rate of approximately 100 kilobytes (25 pages) per second. It adaptively manages this rate to avoid overtaxing the system, particularly when the virtual machine has little remote memory remaining or when the destination node has little free memory available. The memory migration algorithm also ensures that ESX Server does not move memory needlessly if a virtual machine is moved to a new node for only a short period.

When all these techniques of initial placement, dynamic rebalancing, and intelligent memory migration work in conjunction, they ensure good memory performance on NUMA systems, even in the presence of changing workloads. When a major workload change occurs, for instance when new virtual machines are started, the system takes time to readjust, migrating

virtual machines and memory to new, optimal locations. After a short period, typically seconds or minutes at the longest, the system completes its readjustments and reaches a steady state.

Transparent Page Sharing Optimized for NUMA

Many ESX Server workloads present opportunities for sharing memory across virtual machines. For example, several virtual machines may be running instances of the same guest operating system, have the same applications or components loaded, or contain common data. In such cases, ESX Server uses a proprietary transparent page sharing technique to securely eliminate redundant copies of memory pages. With memory sharing, a workload running in virtual machines often consumes less memory than it would when running on physical machines. As a result, higher levels of overcommitment can be supported efficiently.

ESX Server's transparent page sharing has also been optimized for use on NUMA systems. On NUMA systems, pages are shared per-node, so each NUMA node has its own local copy of heavily-shared pages, which means when VMs use shared pages, they don't need to access remote memory.

The ESX Server approach does not require any cooperation from the guest operating system. Also, the ESX Server page sharing options are the same regardless of whether the server is a NUMA system. You may use the MemShareScanVM and MemShareScanTotal configuration options to control the rate at which the system scans memory to identify opportunities for sharing memory. For more information on these options, see Service Console Commands in the ESX Server Administration Guide.

Manual NUMA Controls

Some advanced administrators may prefer to control the memory placement and processor utilization by hand. This may be useful, for example, if a virtual machine runs a memory-intensive workload, such as an in-memory database or a scientific computing application with a large dataset. Such an application may see performance improvements if 100% of its memory is allocated locally, while virtual machines managed by the automatic NUMA optimizations often have a small percentage (5-15%) of their memory located remotely. An administrator may also wish to optimize NUMA placements manually if the system workload is known to be simple and unchanging; for example, an eight-processor system running eight virtual machines with similar workloads would be easy to optimize by hand.

ESX Server 2 provides two sets of controls for NUMA placement, so that administrators can control both memory and processor placement of a virtual machine. The ESX Server Web-based Management User Interface allows the user to indicate that a virtual machine should use only the processors on a given node (through the **Only Use Processors** option) and that it should only allocate memory on the desired node (through the **Memory Affinity** option). If both of these are set before a virtual machine starts, the virtual machine runs only on the desired node, and all of its memory is allocated locally. An administrator can also manually move a virtual machine to another node after the virtual machine has started running. In this case, the page migration rate of the virtual machine should also be set manually, so that memory from the virtual machine's previous node can be moved to its new node. The ESX Server 2 documentation contains a full description of how to set these options.

Note that manual NUMA placement may interfere with the ESX Server resource management algorithms, which attempt to give each virtual machine a fair share of the system's processor resources. For example, if ten virtual machines with processor-intensive workloads are manually placed on one node, and only two virtual machines are manually placed on another node, it is

impossible for the system to give all twelve virtual machines equal shares of the system's resources. Users should take these issues into account when placing NUMA manually.

IBM Enterprise X-Architecture Overview

The IBM Enterprise X-Architecture, which first appeared on the IBM eServer x440 and Fujitsu Siemens Primergy T850 in 2002, supports servers with up to four nodes (also called CECs or SMP Expansion Complexes in IBM's terminology) and each node may contain up to four Intel Xeon MP processors for a total of 16 CPUs. The next generation IBM eServer x445 uses an enhanced version of the Enterprise X-Architecture and scales to eight nodes with up to four Xeon MP processors for a total of 32 CPUs. The third-generation IBM eServer x460 provides similar scalability but also supports 64-bit Xeon MP processors. The high scalability of all these systems stems from the Enterprise X-Architecture's NUMA design that is shared with IBM's high end POWER4-based pSeries servers. A more detailed description of the Enterprise X-Architecture can be found in IBM's Redbook *IBM eServer xSeries 440 Planning and Installation Guide* by doing a search from

www.redbooks.ibm.com

AMD Opteron-Based Systems

AMD Opteron-based systems, such as the HP ProLiant DL585 Server, also provide NUMA support. The BIOS setting for node interleaving determines whether the system behaves more like a NUMA system, or more like a Uniform Memory Architecture (UMA) system. For more information, see the "HP ProLiant DL585 Server Technology" technology brief at

h200005.www2.hp.com/bc/docs/support/SupportManual/c00180597/c00180597.pdf

See also the *HP ROM-Based Setup Utility User Guide* at

docs.hp.com/en/347569-003/347569-003.pdf

By default, node interleaving is disabled, so each processor has its own memory. In this case, the BIOS builds an System Resource Allocation Table (SRAT), so ESX Server detects the system as NUMA and applies NUMA optimizations. If you enable node interleaving (also known as interleaved memory), no System Resource Allocation Table (SRAT) is built by the BIOS, so ESX Server does not detect the system as NUMA.

Currently shipping Opteron processors have either one or two cores per socket. When node memory is enabled, the memory on the Opteron processors is divided such that each socket has some local memory, but memory for other sockets is remote. Thus, the single-core Opterons have a single processor per NUMA node and the dual-core Opterons have two processors per NUMA node.

SMP virtual machines (having two virtual processors) cannot reside within a NUMA node that has a single core, such as the single-core Opteron processors. This also means they cannot be managed automatically by the ESX Server NUMA scheduler. Virtual machines that are not managed automatically by the NUMA scheduler still run fine; they simply don't benefit from ESX Server's NUMA optimizations. Uniprocessor virtual machines (with a single virtual processor) can reside within a single NUMA node and are managed automatically by the ESX Server NUMA scheduler.

Conclusions

The manual and automatic NUMA optimizations in ESX Server 2 allow users to fully exploit the advanced scalability features of platform servers using Intel processors and AMD Opteron processors that provide NUMA support. By providing a dynamic, self-optimizing NUMA load balancer in conjunction with patented memory migration techniques, ESX Server can maintain excellent memory performance even in the face of changing workloads. Users seeking more information on the technical details of NUMA system configuration should consult the ESX Server documentation or the "numa(8)" man page on an ESX Server system.

Glossary

ccNUMA (cache coherent nonuniform memory access) – describes a NUMA architecture in which the system ensures that data stored in processor caches is coherent across all nodes. Virtually all modern NUMA systems are cache coherent, so the term "ccNUMA" is considered redundant. See NUMA.

migration – Migration refers to the movement of a software entity between two nodes in a NUMA system. For instance, page migration is the process by which pages of physical memory can be moved from one node to another.

node – The basic building block of a NUMA system, a node is a group of processors and memory connected to the same system bus, possibly connected to other nodes via a NUMA interconnect. Nodes are also referred to as CECs, SMP Expansion Modules, and quads in IBM terminology.

NUMA (nonuniform memory access) – describes a system architecture in which the machine is organized into a series of nodes containing processors and memory.

remote memory – memory that does not lie in the node from which it is being accessed.

SRAT (system resource allocation table) – table that keeps track of memory allocated to a virtual machine.

VMware, Inc. 3145 Porter Drive Palo Alto, CA 94304 www.vmware.com
 Revision: 20050610 Item: ESX-ENG-Q205-108