Introduction to NUMA on xSeries Servers

Positioning Information (withdrawn product)

Main

The Non-Uniform Memory Access (NUMA) architecture is a way of building very large multi-processor systems without jeopardizing hardware scalability. The name NUMA is not completely correct since not only memory can be accessed in a non-uniform manner but also I/O resources.

NUMA effectively means that every processor or every group of processors has a certain amount of memory local to it. Multiple processors or multiple groups of processors are then connected together using special bus systems (for example HyperTransport or the scalability ports of a x445) to provide processor data coherency. The essence of the NUMA architecture is the existence of multiple memory subsystems, as opposed to a single one on a SMP system.

The so called “local” or “near” memory has the very same characteristics as the memory subsystem in a SMP system. But by limiting the number of processors that directly access that memory, performance is improved because of the much shorter queue of requests. Since each group of processors has its local memory, memory on another group of processors would be considered to be remote to the local processor. This remote memory can be accessed but at a longer latency than local memory. All requests between local and remote memory flow over the inter-processor connection (HyperTransport or scalability ports).

Consider an x445 with eight processors and 4 GB of memory. The x445 implementation puts four CPUs on each of the two SMP Expansion Modules, as shown in the figure below. These two expansion modules are connected together by scalability ports.

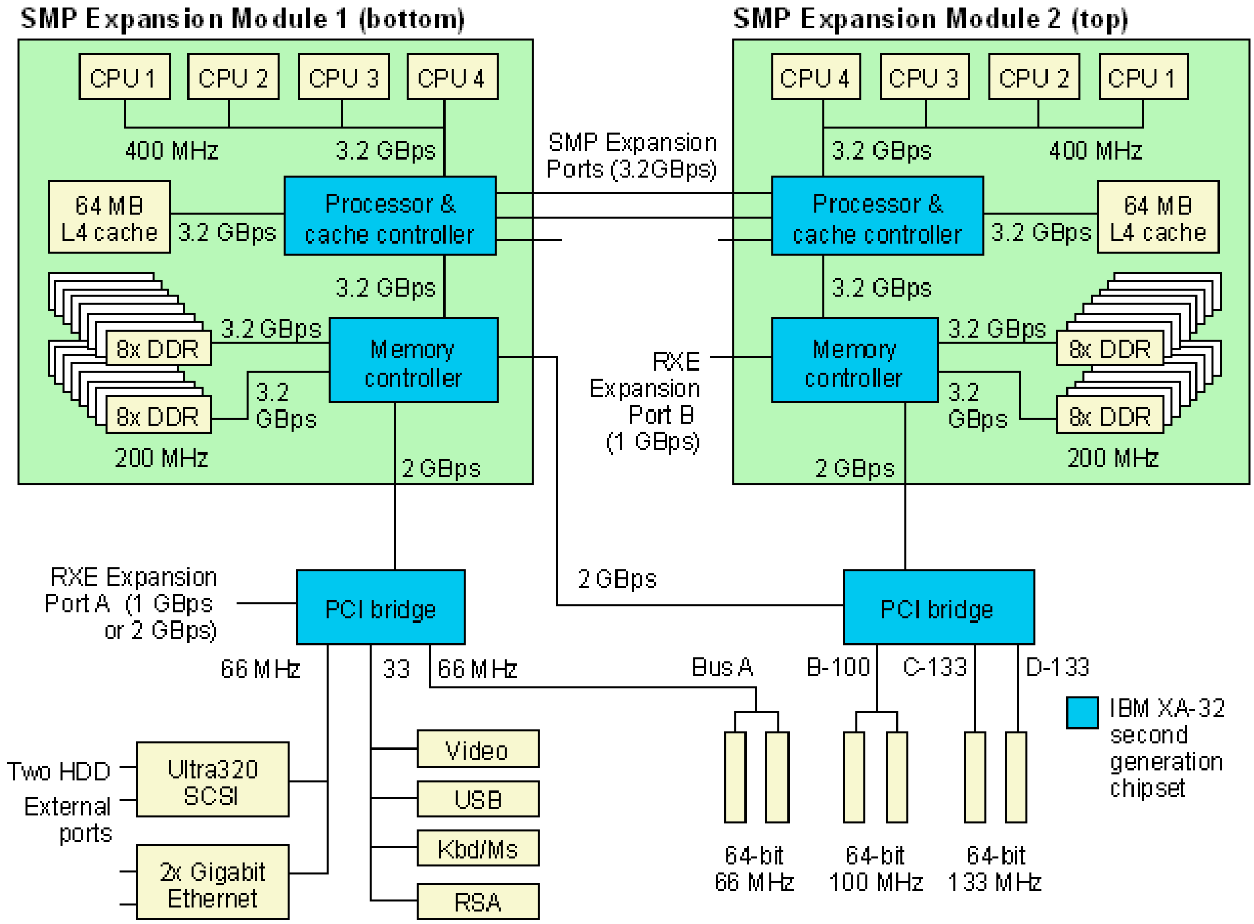

Figure: xSeries 445 system block diagram — two SMP Expansion Modules

An application running on CPUs in SMP Expansion Module 1 may access memory physically located in SMP Expansion Module 2 (a “remote access”). This access incurs longer latency because the travel time to access remote memory on another expansion module is clearly greater.

Many people think this is the problem with NUMA. But this focus on latency misses the actual problem NUMA is attempting to solve. Another way to think about this is to ask yourself this question; you are checking out in your favorite grocery store, with a shopping cart full of groceries. Directly in front of you is a check-out lane with 20 customers standing in line but 50 feet to your left is another check-out lane with only two customers standing in line. Which would you go to? The check-out lane closest to your position has the lowest latency because you don't have far to travel. But the check-out lane 50 feet away has much greater latency because you have to walk 50 feet?

Clearly most people would walk the 50 feet; suffer the latency, to arrive at a check-out lane with only 2 customers instead of 20. We think this way because our experience tells us that the time waiting to check-out with 20 people ahead is far longer than the time needed to walk to the “remote” check-out lane and wait for only two people ahead.

This analogy clearly communicates the performance effects of queuing time vs. latency. In a computer server, with many concurrent outstanding memory requests, we would gladly incur come additional latency (walking) to spread memory transactions (check-out process) across multiple memory controllers (check-out lanes) because this greatly improves performance by reducing the queuing time.

Clearly, we do not want to walk 50 feet to a check-out lane that has 20 customers checking out, when one is directly in front of us with only two customers. So to reduce unnecessary remote access, NUMA systems such as the x445 or the eServer 325 maintain a table of data in the firmware called the *Static Resource Allocation Table* (SRAT). The data in this table is accessible by operating systems such as Windows® Server 2003 (Windows 2000 Server does not support it) and current Linux kernels.

These modern operating systems attempt to allocate resources that are local to the processors being used by each process. So when a process and its threads start on node 0, all execution and memory access will be local to node 0. As more processes are added to the system, the operating system will balance them across the nodes. In this case, most memory accesses will be evenly distributed across the multiple memory controllers, reducing remote access, greatly reducing queuing delays, and improving performance.

The Linux® community (especially the Linux Scalability Effort, http://www.lse.org) has made a tremendous effort to make the Linux kernel NUMA aware. The 2.6 kernel features NUMA awareness in the scheduler (the part of the operating system that assigns system resources to processes), so that the vast majority of processes execute in “local” memory. This information is passed to the operating system via the ACPI interface and the SRAT table.

The AMD Opteron implementation is called *Sufficiently Uniform Memory Organization* (SUMO) and is also a NUMA architecture. In the case of the Opteron, each processor has its own “local” memory with low latency. Every CPU can also access the memory of any other CPU in the system but at longer latency.

You should enable the SRAT information in the system BIOS (if this is configurable) and run a NUMA-aware operating system. Keep in mind that many applications require at least two to four processors to reach maximum performance. In this case, even with NUMA-aware operating systems there can be a high percentage of remote memory accesses in an Opteron system because each processor is the only processor on a node. The frequency of NUMA access will depend upon application type and how users use that application and cannot be estimated without extensive analysis.

For more information on performance tuning, see the IBM Redbook *Tuning IBM eServer xSeries Servers for Performance*, SG24-5287, http://www.redbooks.ibm.com/abstracts/sg245287.html

Related product families

Product families related to this document are the following:

- Processors

Notices

This document, TIPS0476, was created or updated on December 1, 2004.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/TIPS0476
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/TIPS0476.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:
Lenovo®
xSeries®

The following terms are trademarks of other companies:

AMD and AMD Opteron™ are trademarks of Advanced Micro Devices, Inc.

Linux® is the trademark of Linus Torvalds in the U.S. and other countries.

Windows® is a trademark of Microsoft Corporation in the United States, other countries, or both.

IBM® and ibm.com® are trademarks of IBM in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.