Sequent's NUMA-Q™ SMP Architecture

How it works and where it fits in high-performance computer architectures

SEQUENT

Contents

Introduction

The first white paper on Sequent's new high-performance computer architecture, *Sequent's NUMA-Q™ Architecture*, explained the basic components of this new approach to symmetric multiprocessing (SMP) computing. This document offers more detail on how NUMA-Q works and why SMP applications can run on it unchanged, with higher performance than on any other SMP platform on the market today. It also differentiates the NUMA-Q architecture from related technologies such as Replicated Memory Clusters and other forms of Non-Uniform Memory Access, such as NUMA and CC-NUMA. It explains why single-backplane "big-bus" SMP architectures are reaching the limits of their usefulness and why the NUMA-Q SMP architecture is the best solution to high-end commercial computing for the rest of this decade and beyond.

Background terms and technologies

Because there has been so much talk in the press about NUMA (Non-Uniform Memory Access) architectures, it is necessary to distinguish between the original NUMA and recent offshoots: cache-coherent NUMA (CC-NUMA), Replicated Memory Clusters (RMC), Cache-Only Memory Architecture (COMA) and, most recently, Sequent's NUMA-Q (NUMA using quads). It is also useful to go over the basics of Symmetric Multiprocessing (SMP) computing, clustered computing, Massively Parallel Processing (MPP), and other high-speed computer architectures, to appreciate what Sequent's NUMA-Q architecture brings to the table.

A node is a computer consisting of one or more processors with related memory and I/O. Each node requires a single copy, or instance, of the operating system (OS).

A Symmetrical Multiprocessing (SMP) node contains two or more identical processors, with no master/slave division of processing. Each processor has equal access to the computing resources of the node. Thus, the processors and the processing are said to be symmetrical (see Figure 1). Each SMP node runs just one copy of the OS. This means the interconnection between processors and memory inside the node must utilize an interconnection scheme that maintains coherency. Coherency means that, at any time, there is only one possible value held uniquely or shared among the processor for every datum in memory.

Coherency is not a problem for computers with only one processor, because each datum can only be modified by the one processor at any moment. In multiprocessor systems, however, many different processors could be attempting

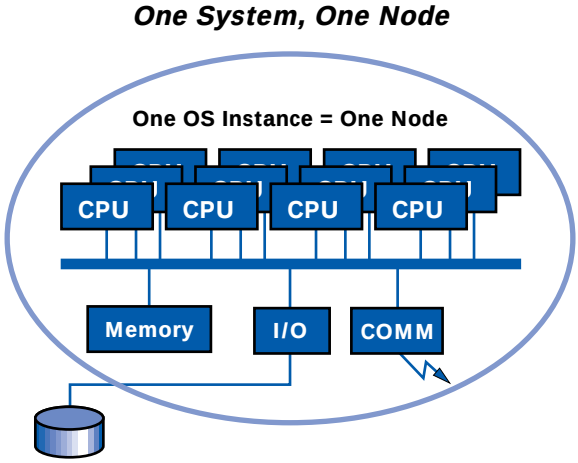

Figure 1: A single SMP node with 12 processors

to modify that single datum or holding their own copies of it in their caches at the same instant. Coherency, whether implemented in hardware or in software, is required to prevent multiple processors from trying to modify the same datum at the same instant. Inside an SMP node, the hardware ensures that write operations on the same datum are serialized. Additionally, the hardware must ensure that stale copies of data are removed. Hardware coherency is a requirement for an SMP machine running a single instance of the OS over multiple CPUs.

To get the best performance out of SMP nodes, the coherency must have low latency and high efficiency. SMP systems have up until now, had a maximum of 12 to 32 processors per node, because the requirement for a low latency, coherent interconnect mandated a single backplane. This puts a physical limit on how many processors and memory boards can be attached. Therefore, from a purely hardware perspective, growing a system larger than 32 processors required connecting nodes using slower software instead of hardware coherency techniques and suffering a dramatic change in programming and performance characteristics.

SMP nodes are very attractive to developers applications, because the OS does a great deal of the work in scaling performance and utilizing resources as they are added. The application does not have to change as more processors are added. In addition, the application doesn't need to care which CPU(s) it runs on. Latency to all parts of memory and I/O are the same from any CPU. The application developer sees a uniform address space. It is this latter point, and the fact that SMP architectures from different vendors look fundamentally the same, that simplifies software porting in SMP systems. Software portability continues to be one of the major advantages of SMP platforms. As an interesting aside, the latency-to-memory is mitigated with large caches on each CPU. Many software applications actually tune their performance by improving their cache hit rate. Nonetheless, once a cache miss takes place, the average latency-to-memory is the same, regardless of what CPU incurred the miss.

It is also relatively easy to extract maximum performance from SMP systems, while t is still difficult to program clusters of small nodes to work together on the same problem—another key reason why a single very large node is more attractive than a collection of clustered small nodes. Within a single node, communication between the CPUs and memory is usually in the one to four-microsecond range. Data is shared by different CPUs through the global shared memory.

The typical SMP architecture employs a "snoopy bus" as the hardware mechanism to maintain coherency. A snoopy bus can be compared to a party-line telephone system. Each processor has its own local cache, where it keeps a copy of a small portion of the main memory that the processor is most likely to need to access (this greatly enhances performance of the processor). To keep the contents of memory coherent between all the processors' caches, each processor "snoops" on the bus, looking for reads and writes between other processors and main memory that affect the contents of its own cache. If processor B requests a portion of memory that processor A is holding, processor A preempts the request and puts its value for the memory region on the bus, where B reads it. If processor A writes a changed value back from its cache to memory, then all other processors see the write go past on the bus and purge the now obsolete value from their own caches, and so forth. This maintains coherency between all the processors' caches.

There are some variants on the single-bus SMP node shown in Figure 1. A large coherent SMP node can be built

with more than one system bus, but this entails inevitable cost and manageability trade-offs. NCR has implemented a dual-bus structure with a single memory shared between buses. Coherency is managed by keeping a record of the state and location of each block of data in the central memory. This type of caching is called directory-based caching and has the advantage in this case of having double the amount of bus bandwidth. The disadvantages include more complex memory hardware and the additional latency for memory transfers spanning both buses.

The Cray® SuperServer 6400 SMP architecture uses four buses. All the CPUs are simultaneously attached to all four buses and implement "snoopy cache" protocols to maintain coherency. Snoopy cache is a hardware caching mechanism whereby all CPUs "snoop" all activity on the buses and check to see if it affects their cache contents. Each CPU is responsible for tracking the contents of only its own cache. This differs from directory-based caching protocols in that the latter uses a directory to look up the state and location of cache lines. This directory can be centrally located or dispersed between the CPUs, with each having a unique portion of the directory. In the Cray 6400 with four buses and snoopy cache, there is the potential for four times the bus bandwidth, but an obvious drawback is that you must use four times the hardware on every CPU to maintain coherency.

The new Sun® Ultra Enterprise 6000 uses a switch to connect all CPUs, memory, and I/O. This switch replaces the traditional backplane but essentially serves the same function. It also has the same drawbacks in that all memory, CPU, and I/O traffic must pass through the switch. The system supports just 16 slots for the CPU/memory and I/O boards. While this new switch has increased the bus bandwidth somewhat, the big-bus problems remain: the requirement for low latency constrains these architectures to a small number of connected CPUs, and increases in the bus or switch speed cannot match the rate at which the CPUs increase performance.

Massively Parallel Processing (MPP or "shared nothing") nodes traditionally consist of a single CPU, a small amount of memory, some I/O, a custom interconnect between nodes, and one instance of the OS for each node. The interconnect between nodes (and therefore between instances of the OS residing on each of the nodes) does not require hardware coherency, because each node has its own OS and, therefore, its own unique physical memory address space. Coherency is, therefore, implemented in software via message-passing.

Messaged-based software coherency latencies are typically hundreds to thousands of times slower than hardware coherency latencies. On the other hand, they are also much less expensive to implement. In a sense, MPPs sacrifice latency to achieve connectivity of greater numbers of processors. This gives MPP computers the opportunity to connect hundreds or even thousands of nodes.

MPP performance is notoriously sensitive to the latency incurred by the software message-passing protocol and the underlying hardware medium for the messages (be it a switch or a mesh or a network). In general, performance-tuning of MPPs involves partitioning the data to minimize the amount of data that must be passed between the nodes. Applications that have a natural partitioning in the data run well on large MPPs—a video-on-demand application, for example.

Figure 2 shows a four-node MPP system. In MPP systems, a node is also called a cell or a plex. The current trend for MPPs is to make the power of the node greater by adding multiple processors, essentially turning it into it an SMP node. This is what NCR offers with its Bynet interconnect and Tandem with its ServerNet. Pyramid has also

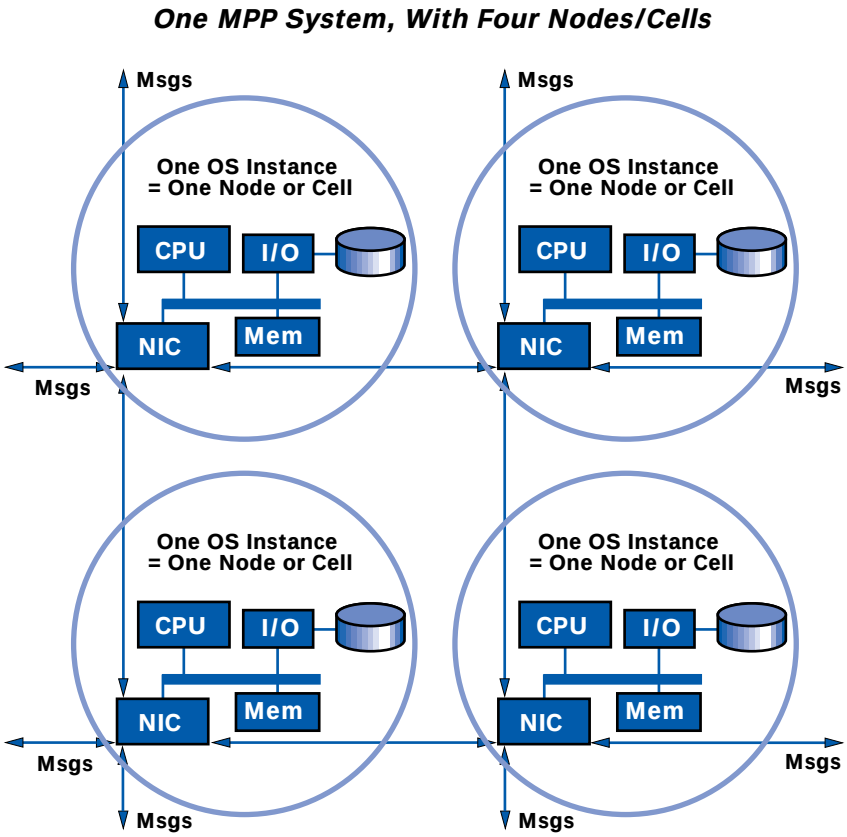

Figure 2: A four-node MPP system

suggested that its Meshine interconnect can be used to connect its SMP platforms as well as its single-CPU cells. The node operating system in MPPs is likely to be a scaled-down version called a microkernel.

MPP architectures are attractive to hardware developers, because they present simpler problems to solve and cost less to develop. Because there is no hardware support for shared memory or cache coherency, connecting large numbers of processors is straightforward. These systems have been shown to provide high levels of performance for compute-intensive applications with statically partitionable data and minimal requirements for node-to-node communication. They are not attractive to application developers, however, the complexity of managing the message between cells and dividing up the application to maximize performance is frequently overwhelming. Most commercial applications are not well-suited to MPP, as database structure changes over time and the overhead of re-partitioning data to avoid hot spots is too great in a continuously available environment.

The key difference between a single SMP node and an MPP system is that, within the SMP node, the data coherence is managed exclusively in hardware. This is indeed fast but also expensive. In an MPP system with a similar number of processors, the coherence between the nodes is managed by the software. It is, therefore, slower but much less expensive. You must manage the traffic between nodes so that it doesn't become a performance inhibitor.

The most effective way to manage this flow of data between nodes is to partition the data (i.e., try to put the data close to the node that is most likely to use it) most of the time. Data sharing

is fundamental to Online Transaction Processing (OLTP) applications. All users want to share data as quickly as possible, preferably in memory. To force OLTP users to partition data is to severely limit performance. This is why MPP systems are rarely proposed for OLTP applications. MPPs have the potential to work better for Decision Support Systems (DSS), although DSS queries have a tendency to serialize, because each step of the query may access a different data partition. This requires a large amount of interconnect bandwidth between nodes.

The bulk of the inter-node traffic in an MPP system is not coherency messages but rather shared data. The sharing of data between processors inside an SMP node requires no data movement, since all processors access the same shared memory. When data must be shared between multiple memories found in multiple nodes, the data must be replicated or transferred from one memory to another. Repeated processor accesses to data in the memory of a different node simply takes too long. This makes data sharing between nodes slow and difficult, at best, and explains why MPP systems perform best for statically partitioned processing (cases where the contents of the database don't change frequently). Unfortunately for MPPs, in commercial markets, problems with statically partitioned data are rare.

A cluster (or clustered system) consists of two or more nodes with the following requirements: (a) each is running its own copy of the OS, (b) each is running its own copy of the application, and (c) the nodes share a common pool of other resources, such as disk drives and possibly tape drives (see Figure 3). Contrast to this, MPP systems, in which nodes do not share storage resources. This is the primary difference between clustered SMP systems and traditional MPP systems.

It is important to note that in a cluster the separate instances of the application must be aware of each other—and must execute locks to maintain coherency within the database—before attempting to update any part of the common pool of storage (the database). It is this requirement that makes clusters more difficult to manage and scale than a single SMP node. However, clusters offer several advantages in return: continuous application availability and superior performance.

One Clustered System, With Two Nodes

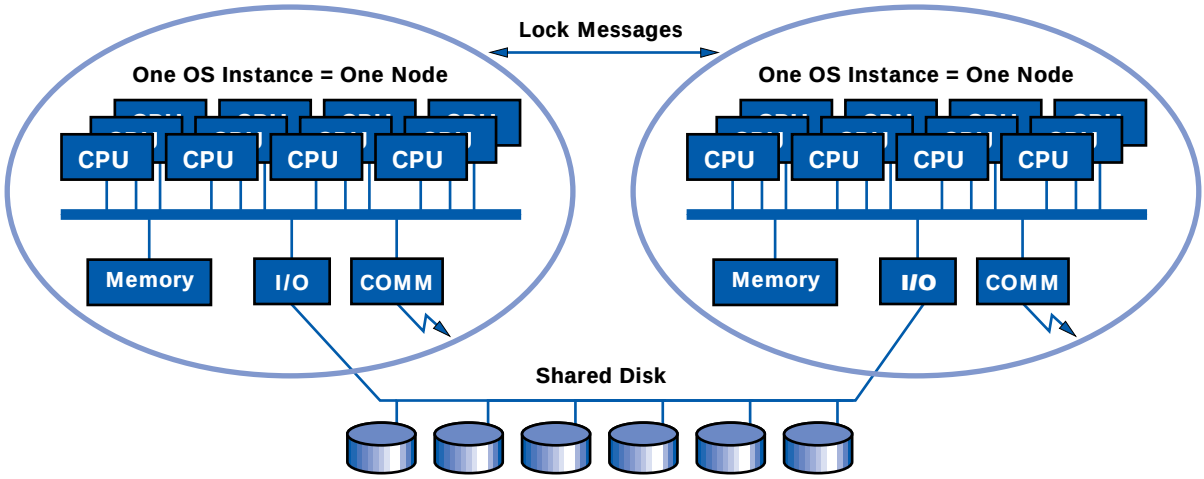

Figure 3: A two-node SMP clustered system

Getting additional performance from clusters is more difficult than scaling within a node. A key barrier is the cost of communicating outside the single-node environment. Communication that passes outside a node must endure the long latencies of software coherency. Applications with lots of interprocess communication work better inside SMP nodes, because communication is very fast. Applications scale more efficiently in both clusters and MPP systems when you reduce the need for communication between processes that span nodes. This generally involves data partitioning. OPS (Oracle® Parallel Server), the most widely known cluster-ready application, uses this approach.

A failover configuration generally consists of a cluster running specialized software. The application is not being used simultaneously on both nodes but rather one node is set up to fail over to another in the event of a crash. This is not a true cluster but is best described as a failover configuration of multiple systems. Applications do not need to be programmed for clusters, since only one instance of the application is ever active. In a failover configuration, you only get the performance of a single node at a time. Also, the application may be unavailable for a short time while the failover is in progress.

A system. To avoid confusion, one should not speak in terms of systems but of nodes. A system can consist of one or more nodes. When a system has two or more nodes that simultaneously share a common pool of storage, it is a clustered system. Note that a clustered system consisting of multiple nodes (each running a unique copy of the OS and the application) is considered a single system. Too often the industry refers to a system as a single-node system, while a clustered system and an MPP system are used to denote specific multi-node systems.

A Replicated memory cluster (RMC) is a clustered system with a memory replication or memory transfer mechanism between nodes and a lock traffic interconnect (see Figure 4). Memory transfers are performed using software coherency techniques. RMC systems provide faster message-passing to applications and relieve the individual nodes from having to go out to disk to get the same pages of memory. On an RMC system, getting data from memory in other nodes is hundreds of times faster than returning

One Replicated Memory Clustered System (RMC), With Two Nodes: NU

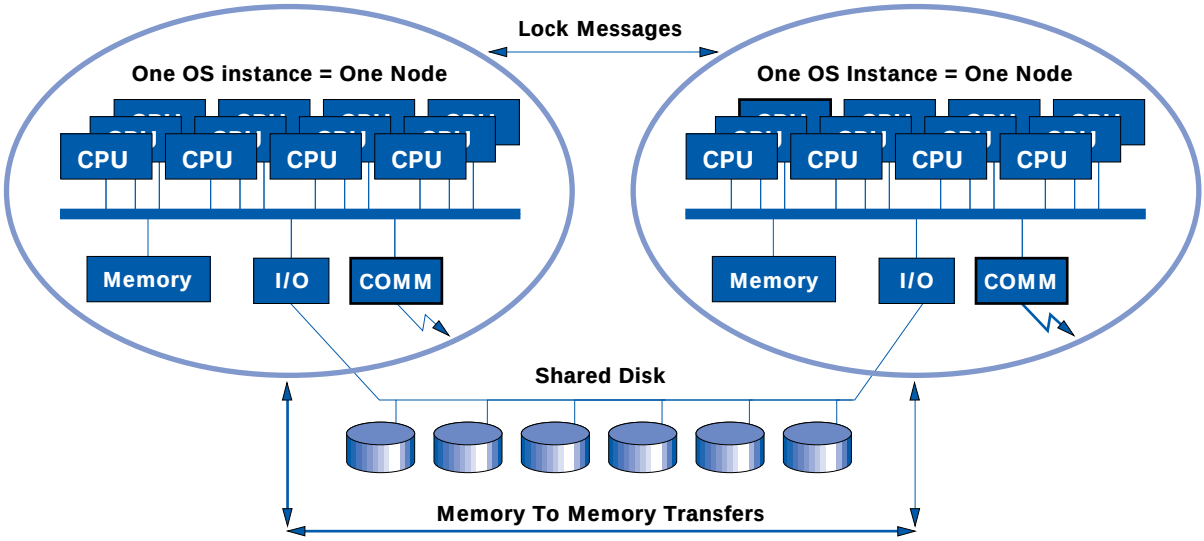

Figure 4: A Replicated Memory Cluster (RMC) system with two SMP nodes.

to disk for it. Clearly, the performance boost will only be realized if there is a need for the nodes to share data, and the application is able to take advantage of it. RMCs are faster than traditional networked-based message-passing because, once a connection is established, a message can be passed from the application code without the intervention of the operating system. Examples of this type of technology are Sequent's Scalable Data Interconnect (SDI), DEC's Memory Channel on the TruCluster, and Tandem's ServerNet. This memory-to-memory transfer is similar to the interconnects between MPP nodes and serves a similar function.

Some vendors use the term "NUMA" to describe their RMC systems, while some journalists have used the term "shared memory clusters" (SMC) to describe both NUMA and RMC. SMG is a poor descriptor, as it is easily confused with the shared memory inside SMP nodes. "Global shared memory clusters" is also a poor choice, because there is no concept of a single-memory image in RMC. These are multiple memories, with multiple memory maps and operating systems, reflecting portions of memory to one another's memories.

NUMA stands for non-uniform memory access. The NUMA category contains several different architectures, all of which can loosely be regarded as having a non-uniform memory access latency: RMC, MPP, CC-NUMA, and COMA. Despite the fact that all of these can be described as NUMA architectures, they are quite different. RMC and MPP have multiple nodes and the "NUMA" part is the inter-nodal software coherency. In CC-NUMA and COMA, which will be discussed later, the hardware coherency is intra-nodal; the "NUMA" piece is inside one node.

Why can RMC be regarded as a NUMA implementation? The rationale is that if a node requests some data from its own memory, the latency is fairly small. If the same node requests data that must be transferred or replicated from the memory of another node via the message-passing hardware and software, latency is substantially longer. Therefore, loosely speaking, you have non-uniform memory access. However, the memory transfers in RMC systems use software coherency, because each of the nodes has its own OS instance and its own memory map. This makes the RMC "NUMA" system quite different from a CC-NUMA node

Clusters	NUMA				UMA (Big Bus)
	MPP	RMC	CC-NUMA	COMA	
S5000 clusters	SP-2 Switch	Sequent® S5000 with SDI	NUMA-Q™	KSR	Pyramid® Nile series
HP T500 clusters	NCR® Bynet	DEC® TruCluster with memory Channel	DG CC:NUMA	SUN S3.mp	Sequent S5000
SPARCcenter Clusters	Tandem® ServerNet	HP SPP1600	Stanford Flash		HP® T500
Pyramid Clusters			MIT Alewife		DEC TurboLaser
			Sun S3.mp		SPARCCenter 2000 SGI Challenge
Multiple Nodes			Single Nodes		

Figure 5: Taxonomy of computer architectures

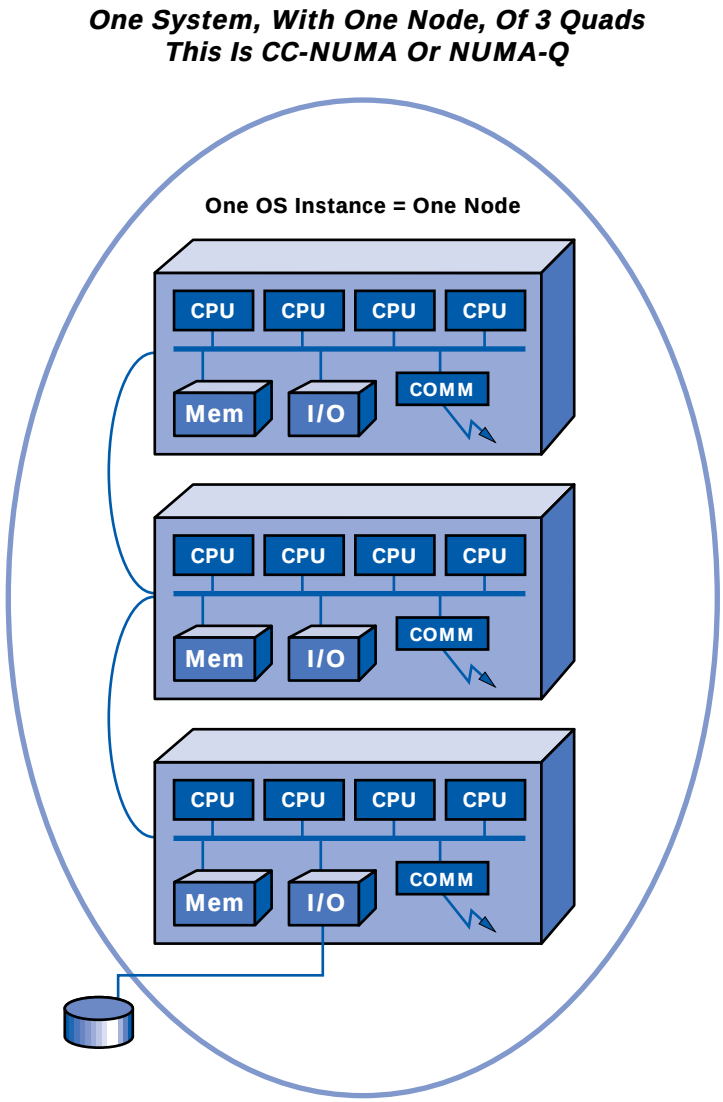

Figure 6: A 12-processor CC-NUMA node

where a single memory is shared between processors within a node using hardware cache coherency. Figure 5 shows a taxonomy of the various architectures and some examples of each.

CC-NUMA means cache coherent non-uniform memory access. CC-NUMA is a type of NUMA but one that is quite different from RMC. In a CC-NUMA system, distributed memory is tied together to form a single memory. There is no copying of pages or data between memory locations. There is no software message-passing. There is simply a single memory map, with pieces physically tied together with a copper cable and some very smart hardware (rather than a backplane). Hardware cache coherent means that there is no software requirement for keeping multiple copies of data up to date or for transferring data between multiple instances of the OS or application. It is all managed at the hardware level, just as in any SMP node, with a single instance of the OS and multiple processors. However, instead of using a snoopy bus to maintain coherency, a directory-based coherency scheme is employed. Figure 6 shows a 12-processor, single-node CC-NUMA system.

COMA, or Cache-Only Memory Architecture, is a rival architecture to CC-NUMA with similar goals but a different implementation. Instead of

distributing pieces of memory and keeping the whole thing coherent with a sophisticated interconnect as we did in NUMA-Q, a COMA node has no memory, only large caches (called attraction memories) in each quad. The interconnect still has to maintain coherency, and one copy of the OS still runs over all of the quads, but there is no "home" memory location for a particular piece of data. The COMA hardware can compensate for poor OS algorithms relating to memory allocation and process scheduling. However, it requires changes to the virtual memory subsystem of the OS and requires custom memory boards in addition to the cache coherency interconnect board.

NUMA-Q is Sequent's implementation of a CC-NUMA architecture. It is a CC-NUMA architecture with a quad processor complex associated with each piece of distributed memory. NUMA-Q means CC-NUMA with quads. We chose to separate four processors beside each distributed piece of the address map, and also add a PCI bus with seven slots. The collection of four processors, some amount of memory, and seven PCI slots is referred to as a quad. Multiple quads can be joined with a hardware-based cache-coherent connection to form a larger single SMP node, in the same way that processor boards are added to a backplane of a conventional big-bus SMP node. NUMA-Q is the architectural description of all of Sequent's future large SMP nodes.

NUMA-Q is the logical growth path for SMP. Because the cache-coherency protocol based on the snoopy bus becomes backplane-limited somewhere between 12 and 32 processors, it's necessary to migrate to a more scalable architecture for hardware-based cache-coherency. This is especially true as the processors become more powerful and the limits for snoopy bus cache coherency move down from 32 processors. The best-known alternative hardware cache-coherency protocols are based on a directory-based cache protocol. Sequent has determined the best variant to be the cache-coherent, non-uniform memory architecture employing SCI.

A NUMA-Q quad consists of four processors, memory, and seven PCI slots on two PCI channels. It could be the only processing element in a system. If this were the case, it would be a single-node system, and the quad would correctly be described as a node. When a system consists of multiple quads, it is still a single-node system, since only one instance of the OS is running over all quads. In this case it is incorrect to refer to any one of the quads as a node. It should be noted that the academic community uses the term node to refer to the smallest aggregation of processors and memory in a CC-NUMA system. This usage is not acceptable in the commercial community because of the confusion with the definition of a cluster node.

The IQ-Link™ interconnect is Sequent's coherent interconnect between quad buses. This interconnect is implemented strictly in hardware and requires no software to maintain coherency. For the most part, it is invisible to the software just as caches are—being just a more advanced form of a cache-coherent backplane. Instead of placing 12 processor boards on a single backplane, we plug a copper link between groups of processors and memory. The IQ-Link maintains cache coherency just as SMP backplanes do. This permits a single instance of the OS to be spread over multiple quads.

The difference between the backplane-based big-bus SMP nodes and a multi-quad NUMA-Q SMP node is that the NUMA-Q interconnect overcomes the limitations of the single backplane and permits the building of very large SMP nodes. Big-bus systems are designed for throughput, not latency. By using short buses inside each quad, we achieve very low latency for most accesses. IQ-Link allows us to build very big systems with

multiple short, low-latency buses. The Sequent-designed IQ-Link interconnect offers lower latency and higher effective throughput than any other alternative available today. The result is better system scalability and overall performance.

A clustered system of NUMA-Q SMP nodes runs the same clusters software as single-backplane SMP clusters. When the memory-to-memory transfer mechanism is added, it becomes a *replicated memory cluster with NUMA-Q nodes.* The software coherency and memory transfer medium between nodes is likely to be implemented with Fibre Channel. In Sequent's case, all of the software used today in the software-coherent memory transfers between S5000 nodes (ptx/SDI and ptx/CLUSTERS) and will be used in the NUMA-Q-based RMC systems.

One might ask why there is a need to build RMC systems with NUMA-Q nodes at all, when very large NUMA-Q nodes can be built instead. There are two reasons, and they are the same reasons for using clusters today: availability and scalability beyond one node. Application availability is better for clusters than for single nodes, and for many business problems, application availability is of paramount importance. Scalability is important because, no matter how big the nodes become, there will always be applications that need larger nodes. Figure 7 shows a two-node NUMA-Q cluster system.

Latency, the key differentiator

One of the key differences between all the architectures described here is the programming model. Programming differences stem directly from latency difference. The whole point of using

One Replicated Memory Cluster System (RMC), With Two Nodes, Each Having Three Quads

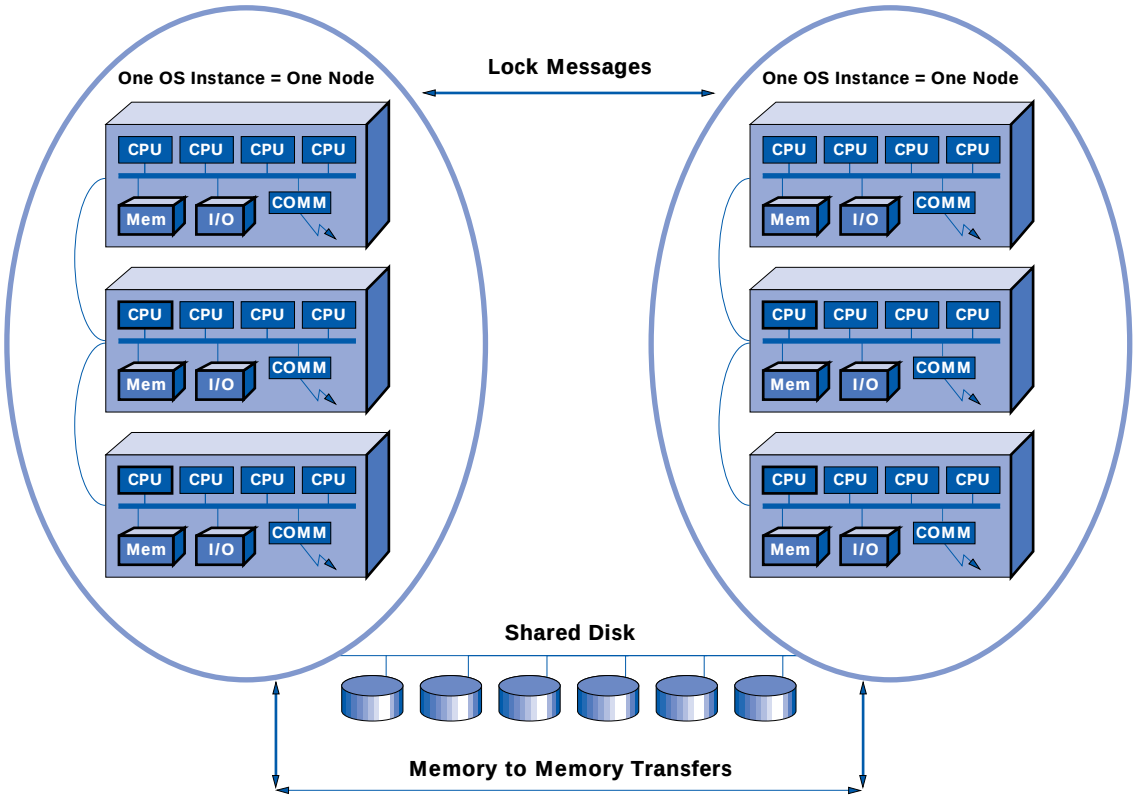

Figure 7: A Replicated Memory Cluster built with two NUMA-Q nodes

memory is to locate frequently-used data in microseconds rather than the milliseconds it takes to retrieve it from disk. In the case of MPPs with distributed disks, accesses to remote disks (where the disks are attached to a different node than the requesting node) can be in the tens of milliseconds.

Because disk accesses are so long, computer designers often add a software-coherent interconnect between nodes of a cluster to yield what some call "global shared memory." The resulting Replicated Memory Clusters are a type of NUMA, and the accesses to remote memory is in the hundreds of microseconds. This is certainly hundreds of times faster than going out to disks for the same data, but hundreds of microseconds is still hundreds of times slower than local memory speeds. Thus, the programming model must minimize these types of transfers by partitioning the data as well in advance as possible. You will note that RMC remote accesses, disk accesses, and remote disk accesses all involve both software and hardware, while local memory accesses are implemented completely in hardware.

Local memory accesses in big-bus architectures are implemented entirely in hardware, they are coherent, and they are quite fast-usually less than two microseconds. This is why SMP programming is so advantageous to application programmers. The programmer does not have to worry about partitioning data in memory, because all parts of memory are accessed from any CPU, with equally fast times.

NUMA-Q has taken this process one step further. Not only is the worst-case memory access time roughly the same as the big-bus architecture accesses, but a majority of the accesses will be ten times faster! This means that the average memory access in a NUMA-Q system is actually faster than any big-bus architecture today. In addition, a 32-processor NUMA-Q system has more memory bandwidth than any other SMP architecture today. (It should be clear by now that NUMA-Q memory accesses are quite different than RMC transfers, even when RMC architectures are referred to as NUMA.) The "Giga-bus" vendors are touting 256-bit-wide buses and one to three GB/s bandwidths, but these are the end of the line for the single big-bus or switch-based systems. NUMA-Q is the first commercial venture into CC-NUMA and promises to lead the industry in achieving high bandwidth and low latency for large SMP nodes.

Figure 8 shows the actual latency of the containers of data on a log scale. (Because the differences are so vast, you can't show them on a linear scale.) It is interesting to translate these times from microseconds into something we are more familiar with: seconds. If the NUMA-Q memory latency of one microsecond were the same as waiting on your PC to respond for one second, then the RMC (NUMA clusters) would keep you waiting two to three minutes, a disk access would delay you three hours, and a remote disk access six to twelve hours.

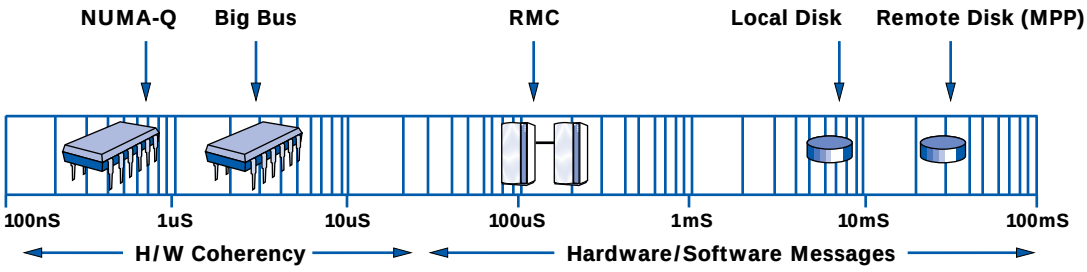

Figure 8: A log scale showing the relative latencies of memory, Replicated Memory Clusters, and disks.

How NUMA-Q works

The first NUMA system was the Butterfly machine, developed by BBN in 1981. The first operational CC-NUMA system was the Stanford DASH. The team working on this system took the opportunity to study the Irix operating system (SGI's Unix operating system) on a 32-processor CC-NUMA node in 1992. The Stanford team is currently building the follow-on to the DASH, called FLASH. One goal of FLASH is to integrate the SMP cache-coherent shared-memory model and the MPP software-based cache-coherent message-passing model into one architecture.

Sequent chose to use quads, with four processors per portion of memory, as the basic building block for NUMA-Q SMP nodes. Adding I/O to each quad further improves performance. Therefore, the NUMA-Q architecture not only distributes physical memory but puts four processors and seven PCI slots next to each part. In a node with three quads, one third of the physical memory will be close (from a memory access latency perspective) to four processors, and two thirds will be "not quite so close." This might lead the reader to believe that two thirds of the memory accesses will be slow, and only one third fast. Fortunately, without any modification of the SMP-based applications, this is not the case. In fact, a majority of a processor's memory accesses will be very fast indeed, and only a small percentage will be "not quite as fast." This is because of the large local quad memory and the large remote cache on the IQ-Link board.

NUMA-Q memory configuration

The memory in each quad is not local memory in the traditional sense. Rather, it is one third of the physical memory address space and has a specific address range. The address map is divided evenly over memory, with each quad containing a contiguous portion of address space. In our example, shown in Figure 9, quad 0 has physical addresses 0-1GB, quad 1 has physical addresses 1-2GB, and quad 2 has physical addresses 2-3GB. Only one copy of the OS is running and, as in any SMP system, it resides in memory and runs processes without distinction and simultaneously on one or more processors.

To simplify descriptions hereafter, the memory segment on the quad under discussion will be referred to as "local quad memory" and the memory on the other quads as "remote quad memory." Therefore, memory accesses to local quad memory are very fast; accesses to remote quad memory are not as fast. Access latencies to the one physical memory space are non-uniform, which is why NUMA-Q is a true NUMA architecture.

The Intel® Pentium® Pro processors have both an L1 cache and an L2 cache inside the chip. For reasons explained later, a majority of any processor's L2 cache misses will be found to be addresses that fall into the range of the local quad memory, and so are serviced very quickly. The ability of the code and data accesses to stay within a given region of address space is called "spatial locality." If the address is outside the range of the local quad memory, the IQ-Link 32MB cache is searched. This 32MB cache is accessed at the same speed as the local quad memory. If the data is not found in the IQ-Link cache (also called the remote cache), the IQ-Link sends a request out on the IQ-Link bus to get the data. After the data is fetched from another quad, it is stored in the remote cache of the IQ-Link board of the requesting quad, but it cannot, and is not, stored in the memory of this quad unless the software explicitly copies one part of memory to another.

As an example, suppose that quad 0 in Figure 9 needs to test and set a semaphore. The address is within the 1-2GB address range so does not fall within the range of the local quad memory on

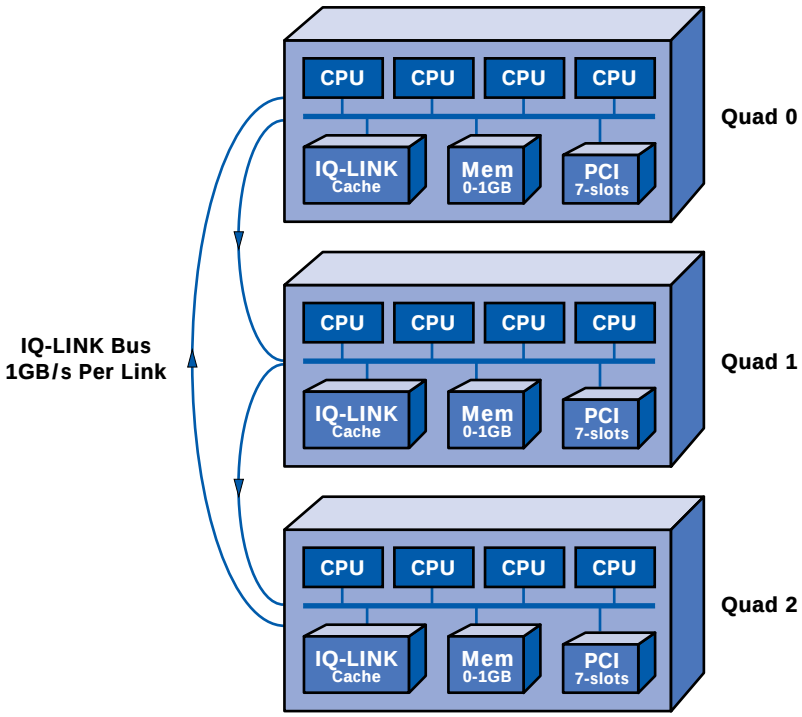

Figure 9: A three-quad NUMA-Q node, showing the IQ-Link bus

quad 0. The IQ-Link in quad 0 recognizes that the address on the Pentium Pro bus is outside of the range set for the local quad memory. It forwards the request, via the IQ-LINK bus, to the IQ-LINK on quad 1. Quad 1's IQ-Link receives the request, requests access to the Pentium Pro bus, fetches a copy of the data from quad 1 memory, and sends it back to quad 0, where it is stored and subsequently modified in the remote cache on the IQ-Link of quad 0.

This transfer took two hops in that the request "hopped" from quad 0 to quad 1, and the returning data "hopped" from quad 1, through quad 2, back to quad 0. Since the packet was not addressed to quad 2, it is bypassed directly from quad 2's input to output in about 16 nsec. The ring is uni-directional, so the returning data had to pass through the IQ-Link on quad 2 to get back to quad 0. From the time the Pentium Pro on quad 0 first had the cache miss in its internal L2 cache until the data was returned to that Pentium Pro and it is able to use that data, is about 3.3 microseconds. This latency is similar to that observed for a cache-miss in a conventional single-backplane, big-bus architecture but it occurs much less frequently.

The data request in this example is for a semaphore, and the intent is to do a test and set. The initial request is received by quad 1, whose memory contains the semaphore. The IQ-link on quad 1 retrieves the data from its memory and observes that the request is with "intent to modify." The IQ-Link therefore makes a record of this intent, and also a record of the quad address to which the data is being dispatched, and returns the data to quad 0. Subsequently, if a processor on quad 1 attempts to read this data from its own local quad memory, the IQ-Link intercepts the request, since it knows that the copy in that memory location is "stale." A request is issued back to the last known location of the modified data (quad 0 in this case), the data is retrieved, and updated in memory in quad 1. Remember that quad 0 does not store this datum in its memory, only in its IQ-Link cache. It cannot put it in memory since the address for this data does not fall into the address range of quad 0.

Obviously, this is a simple example, and there are complex operations of coherency requiring four or more hops to retrieve data. However, it should serve to prove that the hardware is responsible for maintaining coherency, and the software view is one large contiguous memory, with a single copy of the OS. It is true that this architecture is more complicated than the traditional cache technologies of single-backplane, snoopy bus systems. However, the complexity is necessary to overcome the bandwidth limitations of the big-bus architecture.

Cache coherence in NUMA-Q

When one examines the transfers over a single bus in single-backplane big-bus SMP architectures such as that shown in Figure 1, one realizes that some of the transfers are I/O (between the I/O ports and memory) and others are simply CPU accesses to memory called "capacity misses" or "conflict misses" caused by the size and configuration of the L2 cache. The remainder are cache-to-cache transfers between CPUs called "coherency misses."

The drawback with the big-bus architecture is that all caches have to have visibility to all transfers to maintain coherency, so they implement a snoopy cache protocol. As the bus is redesigned to make it faster, the length of the bus must shrink. An alternative is to remove some of the traffic from this bus and free up bandwidth in an attempt to meet the bandwidth requirements of faster processors. Some vendors have added a separate I/O bus and dual-ported memory for this purpose. This adds substantial cost and complexity to the system architecture and only limited bandwidth improvement. Other vendors have built dual system buses with coherency managed by a dual-ported memory and a directory-based cache-coherence protocol. This approach fails to deliver twice the bus bandwidth, however, because so much coherency activity has to cross the memory between the buses, consuming bandwidth on both buses while increasing average latency.

In NUMA-Q, Sequent has attempted to separate the coherency misses from the I/O and the CPU-to-memory traffic (capacity and conflict misses). In this way, the CPU-to-memory traffic and the I/O traffic can occur simultaneously on multiple 500MB/s buses. The IQ-Link coherency bus interconnects and monitors all of the CPU-memory buses and assures data coherency between them. The coherency bus or "IQ-Link bus" bandwidth has been increased to 1GB/s per link. The CPU-to-memory and I/O bandwidth is 500MB/S for each quad. As you add more quads, you add more CPU-to-memory and I/O bandwidth.

For a system with eight quads (32 processors), the CPU-to-memory bandwidth is eight times 500MB/s or 4GB/S. With two PCI buses in each quad, 2GB/S of I/O bandwidth can theoretically be supported. The IQ-Link bandwidth remains at 1GB/S per link, as it is connected to all eight quads. However, most of its traffic will be coherency misses, and to a much lesser extent some "not so close" memory fetches or capacity or conflict misses, and even some I/O on occasion. The ability for the I/O subsystems to be centralized and yet still accessible from any quad without requiring I/O traffic over the IQ-Link is very important.

When comparing these bus bandwidths to conventional big-bus architectures, one must aggregate the 500MB/s quad buses, since these are the buses carrying the bulk of the traffic common to all SMPs: the capacity misses and the I/O. It would be incorrect to compare just the IQ-Link bus to a conventional big bus since the majority of the IQ-Link traffic is just coherency misses. Since the NUMA-Q architecture is designed to support up to 63 quads under one instance of the OS, this means that the theoretical maximum bus bandwidth for I/O and capacity misses is 32GB/S (63 * 500MB/s). This compares to single backplane speeds today in the range of 240MB/s to 1.6GB/S. The IQ-Link is a point-to-point connection from IQ-Link to IQ-Link, ultimately forming a ring.

All indications lead us to believe that the interconnection of multiple 1GB/s links in a ring will result in an 1.6 GB/s aggregate throughput for the IQ-Link ring.

Optimizing software for NUMA-Q

Clearly, the success of Sequent's NUMA-Q architecture depends on the following:

1. The SMP applications software does not have to be changed to get the most performance out of this architecture.
2. The frequency of "not so close" accesses is substantially less than that of "close" accesses.
3. The latency of "not-so-close" accesses is very short.
4. The bandwidth of the IQ-Link is much greater than that required for large OLTP, DSS, and business communication solutions.

How might SMP software developers optimize their software for NUMA-Q? One could make software changes to accommodate NUMA-Q if points two, three, and four above weren't true. If points two, three, and four, are true, however, then the fact that a small percentage of accesses take longer than the rest can largely be ignored by programmers, in the same way that the working of caches is ignored. Clearly, tuning by programmers for cache size and set associativity can and do yield some incremental performance, in the same way tuning to a NUMA-Q architecture can yield some small incremental performance. However, the important point is that most of the potential system performance can be achieved without changing SMP applications software for NUMA-Q.

In designing the IQ-Link interconnect, Sequent ran many tests using Oracle and Informix applications. These tests showed us that these software programs demonstrate good code and data spatial locality (i.e., they didn't randomly hop all over the address map for code and data). In an eight-quad node, a great deal more than an eighth of the accesses are found in local quad memory. Specifically, over 51 percent of the L2 cache misses are found in the local quad memory on an eight-quad node, and an additional 30 percent are found in the 32MB remote cache on the IQ-Link. The remaining 19 percent of the L2 cache misses may require remote memory access. This proves that point two above is true.

Memory latencies in NUMA-Q

What of the memory latencies in the NUMA-Q architecture? In traditional single-backplane big-bus SMP architectures, the average memory latency is about two micro-seconds. This is mitigated to some extent by large (2MB) L2 caches, but nonetheless plays a significant role in the overall performance of the node. The reason that the latency is much larger than the access time of DRAMS (about 60ns)is that the request has to go through the process of missing in the L2 cache on the processor board, requesting access to the backplane bus, getting onto the backplane bus and into the queue for a memory access, over to the memory board, and ultimately returning with the data.

In a NUMA-Q node, an access to local quad memory is roughly ten times faster than a memory access in a single backplane SMP node. This is because: 1) the memory is close to the processors, 2) there are just four processors, 3) there is no backplane between the processors and the local quad memory, and 4) the bus between the processors and the memory can operate at much higher speeds. The latency of the local quad memory can be as low as 180ns from L2 cache miss until the data is returned to the processor.

After the latency of local quad memory, our next concern is with the latency of the IQ-Link 32MB remote cache, which is the first place searched for data whose address does not fall into the range offered by the local quad memory. This remote cache is accessed simultaneously with, and at the same latency as, the local quad memory, thus providing an

additional 32MBs of memory that is also ten times faster than memory in the single-backplane big-bus architectures.

Finally, we must know the latency of a remote quad memory access. We must understand both the minimum latency and also those things that contribute to the maximum latency. We must understand the frequencies of these different latencies. Here we will only mention the minimum latency for an eight-quad system, which is 3.3 microseconds from the processor L2 cache miss until the data is returned. This is almost as fast as a memory access in a single-backplane big-bus SMP system, but occurs much less frequently because of the 512MB-4GB of local quad memory, and the 32MB remote cache in each IQ-Link. Compare this to typical single-backplane big-bus processor caches of just 2MB-4MB.

When one sums it all up and takes cache hit rates into account, the average latency in a single-backplane big-bus SMP node today is one to four microseconds, whereas the same software on the NUMA-Q node, with no changes, would have an average latency of just 1 to 2.5 microseconds. This is one of the reasons that applications vendors do not have to change their SMP software to get maximum performance from NUMA-Q systems; the cumulative effect of latency to remote quad memory is very small indeed.

IQ-Link bus bandwidth

Finally, there is the bandwidth of the IQ-Link bus. It is our expectation that for a 32-processor, single-node running at roughly 20,000 to 25,000 transactions per minute, the IQ-Link is not even 30 percent utilized. This is attributable to the efficiency of our coherency protocol, the inherent good spatial locality of the applications available today and, of course, the very high realizable bandwidth of the IQ-Link. Sequent expects a 32-processor NUMA-Q node to triple our current high-end performance.

If the applications companies subsequently choose to make changes that benefit NUMA-Q (and, coincidentally, other SMP platforms and MPP), they might get up to 20 percent additional performance.

The IQ-Link has a great deal of intelligence built into it. It must be able to keep track of the location of all data it has “checked-out” from its own quad, as well as all the data it is caching in its 32 MB remote cache. It monitors the Pentium Pro bus and simultaneously uses a directory-based coherence protocol to keep track of all requests it makes and requests it responds to on the IQ-Link bus. It is the go-between from the four-state snoopy cache coherence of the Pentium Pro bus, to the 39 state directory-based coherence protocol of the IQ-Link bus.

The bottom line is that the applications software won’t have to change, because the hardware and the operating system work together to ensure excellent locality and low latencies. The result is that most of the potential performance of a NUMA-Q node can be realized with existing binaries. Applications companies will modify or port their applications only if they feel strongly about getting the remaining small percentage improvements by changing their software to improve the locality even more. The NUMA-Q architecture will run their software unchanged faster than any other SMP node available to date. Any further tuning will just widen this performance gap. It is interesting to note that this work will also yield improvements in conventional single-backplane big-bus systems because of the large cache sizes, and also in MPPs, where locality of reference is the most critical factor to scalability.

Summary

Sequent started commercial Unix®-based SMP in 1983. Today, many vendors offer Unix-based SMP platforms. Sequent is now pioneering the next step beyond Big Bus SMP—the NUMA-Q SMP architecture. We believe that some day, all high-end servers will be built using this architecture.

Data access latency is clearly the determining factor in system performance today. We put memory in our systems to mitigate against disk latency. SMP platforms in the past have provided an easy programming model, since the latency to all parts of memory has been equally small. Attempts to share data between nodes by transferring data between the memories are successful if the alternative is to return to disk for the data. However, they still require substantial programming changes to optimize performance because inter-node latencies are much larger than intra-node latencies.

The most efficient way to get high performance and keep the programming model simple (no data partitioning) is to build the single nodes as large as possible before going to a second node. With the limitations placed on the size of backplanes and system buses, this cannot be done with snoopy cache and still get to 32 or more processors. The best way to accomplish this is to use the directory-based cache protocols and CC-NUMA. This has the added advantages of permitting the connection of up to 252 processors, getting enormous memory and I/O bus bandwidth, and still achieve average memory latencies below any system available today. Best of all, you don't have to change your SMP applications software to reap most of the performance and out-perform all existing big-bus SMP platforms.

Additional information

Additional white papers regarding NUMA-Q based systems will be available on Sequent's web site: *http://www.sequent.com/solutions/whitepapers/index.html#numa-q*.
Implementation and Performance of a CC-NUMA System, by Russell Clapp and Tom Lovett. An in-depth technical look at the cache coherence of NUMA-Q and the expected performance. This paper is to appear in the proceedings of the 23rd Annual International Symposium on Computer Architecture, May 1996, under the title, *A CC-NUMA Computer System f*or the Commercial Marketplace. The version of the paper on the web under *http://www.sequent.com/products/highend_srv/imp_wp1.html*, replaces some of the internal engineering terms and code names with more widely recognized industry terms.

Considerations in Implementing a System Based on SCI, by Robert J. Safranek. Provides even more detail on the NUMA-Q cache coherence. SCI (Scalable Coherent Interface) is a IEEE specification (1596-1992) that defines a directory based cache coherence protocol, the physical interface, and the packet formats. Sequent's IQ-Link used this as a starting point.

Subsequent NUMA-Q White Papers will discuss:

- The business problems NUMA-Q uniquely solves.
- NUMA-Q availability features, single points of failure and extended outage, and how to minimize them.
- Hardware redundancy, hardware support for online replacement and insertion of key components.
- Fibre Channel-based I/O and multipathing.
- The AV-Link component and NUMA-Q availability and manageability.

- The management and diagnostic controller, and virtual console software.
- Measured performance numbers for SMP software applications.
- Changes to the Unix operating system to get incremental improvement from NUMA-Q architectures.
- How the OS and application spread out; how the database SGA will be spread out.
- Group affinity and soft affinity and what effect they have.
- Readiness of the commodity quad to enter the data center.

For answers to your questions, email Sequent at: *numa-q@sequent.com*

References

Sequent's NUMA-Q Architecture Whitepaper
http://www.sequent.com/products/highend_srv/arch_wp1.html

Scalable Data Interconnect: An Architecture For building Large Data Warehouses That Have No Limits
http://www.sequent.com/products/midrange_srv/symmetry/sdi_wp1.html

The Requirements and Performance of Enterprise Computer Solutions: SMP, Clustered SMP, and MPP Whitepaper
http://www.sequent.com/products/midrange_srv/symmetry/smp_wp1.html

Corporate headquarters:
Sequent Computer Systems, Inc.
15450 SW Koll Parkway
Beaverton, Oregon 97006-6063
(503) 626-5700 or (800) 257-9044
URL: http://www.sequent.com

European headquarters:
Sequent Computer Systems, Ltd.
Sequent House
Unit 3, Weybridge Business Park
Addlestone Road
Weybridge
Surrey KT15 2UF
England
+44 1932 851111

With offices in:
Australia, Austria, Brazil, Canada, Czech Republic, France, Germany, Hong Kong, India, Indonesia, Japan, Korea, Malaysia, The Netherlands, New Zealand, Philippines, Singapore, Thailand, United Kingdom, and United States.

With distributors in:
Brazil, Brunei, China, Croatia, Czech Republic, Greece, Hong Kong, Hungary, India, Japan, Korea, Kuwait, Malaysia, Mexico, Oman, Philippines, Poland, Russia, Saudi Arabia, Slovak Republic, Slovenia, South Africa, Sri Lanka, Thailand, Ukraine, and United Arab Emirates.

PD-1134 6/97

SEQUENT