# NUMA-Q: A New Commercial Parallel-Processing Architecture

Russell M. Clapp, Ken F. Dove, Wayne Downer, Thomas D. Lovett,
Paul E. McKenney, and Robert J. Safranek

*Sequent Computer Systems, Inc.*

Traditional shared-memory multiprocessor (SMP) systems are built to a “big bus” architecture that connects processors, memory, and I/O devices. This architecture has enjoyed great success--almost all systems vendors now offer some sort of big-bus SMP system, and these systems enjoy widespread use for many applications, including online transaction processing (OLTP) and decision support.

Unfortunately, as clock rates increase to meet the demands of ever-faster processors, laws of physics dictate that system busses be physically shorter and have fewer connectors, sharply limiting system size. Therefore, Sequent has developed a Cache-Coherent Non-Uniform Memory Access (CC-NUMA) architecture named NUMA-Q[1], which is already delivering impressive increases in performance and scaling, and has great potential for future growth.

## Historical Perspective

As late as the early Eighties, most computer systems vendors designed and implemented their own proprietary processors on printed circuit boards at great expense. As single-chip 32-bit microprocessors became available several companies, including Sequent, decided that computer systems could be built more cost-effectively using these off-the-shelf microprocessors as the fundamental building block. Although these microprocessors were not as fast as board-level processors, the advent of simple and robust cache-coherence[2] protocols enabled multiple microprocessors to be attached to a common backplane memory bus. These symmetric multiprocessing (SMP) systems provided a performance multiplier that, for many important applications, overcame the single processor performance deficit of the microprocessors of the day. Use of single-chip microprocessors also reduced parts count, which increased reliability, thereby enabling these machines to be used for online transaction-processing.[3]

Over time, the combined forces of economics, packaging, performance, and reliability induced the surviving computer systems vendors to build SMP systems based on single-chip microprocessors.

These same forces--particularly performance and reliability--induced Sequent to build CC-NUMA systems based on newly-available “glueless” SMP building blocks tied together with point-to-point interconnects.

## Approaching the Big-Bus Performance Limit

The primary impetus for NUMA-Q was the performance limitations of big-bus architectures, which cannot connect large numbers of processors while still providing both the bandwidth capacity and the memory access latency required by today’s high speed microprocessors. Large bus-based systems may provide high bandwidth to a large number of processors by transferring more data bits per clock cycle, but their ability to support newer high speed processors is limited by three memory-access latency considerations. First, the more processors, the greater the chance that memory transactions will be delayed by other transactions already using the bus. This bus-contention effect can be reduced by increasing pipeline depths and supporting out-of-order transaction completion, but these optimizations are limited by the ability of the microprocessors to hide the resulting increased latencies. Second, large bus-based implementations are often forced to trade off latency for throughput in order to improve overall

[1] The engineering product name of the NUMA-Q product was STiNG, which stands for “Sequent, The Next Generation”. This footnote is sponsored by the Radical Front for the Retention of Engineering Product Names.

[2] Cache-coherence protocols are used by per-processor caches to ensure that all processors see a consistent view of memory.

[3] Applications that require even higher levels of reliability and availability use clustering techniques, where multiple systems are connected with a low-latency network and system software enables them to share disk subsystems and to divide up application workload.

scalability. For example, an out-of-order protocol may allow a processor to write data to memory while another read transaction is outstanding. While this may increase the amount of usable bandwidth on the bus, it also increases the likelihood of delaying the read response because the write may be using the bus when the memory is ready to respond to the read. Finally, large workloads often require large amounts of memory and large numbers of I/O controllers in addition to large numbers of processors. Unfortunately, as the number of loads on the bus increases, the limits imposed by both propagation delay and electrical loading make it more difficult to achieve the high bus-clock frequencies required for low latencies.

For an example of this last consideration, imagine a hypothetical database benchmark that requires one processor for each 10 transactions per second (TPS) and one disk spindle for each TPS.[4] Suppose that a big-bus machine has 16 slots, that one processor board contains a pair of processors, and that one disk controller supports four SCSI busses for a total of 60 disks. Each additional processor board increases throughput by 20 TPS--but only as long as there is room for enough disks. As soon as the machine runs out of room for disks, throughput drops off sharply, as shown in Figure 1. Conversely, each additional disk controller increases throughput by 60 TPS--but only as long as there is room for enough processors.

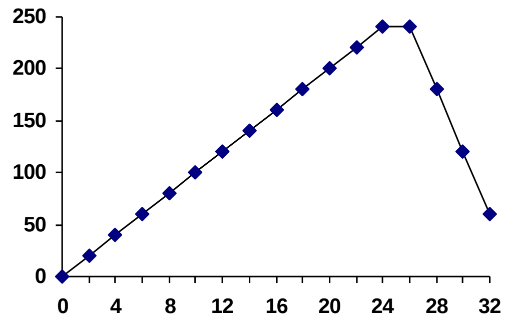

**Figure 1: Big-Bus TPS as a Function of Number of Processors**

The big-bus architecture forces the user to trade off among the processor, I/O, and memory resources required for the workload. This tradeoff only becomes more severe as increasing bus-clock frequencies force the number of slots to decrease.

This is not to say that big-bus architectures are entirely uncompetitive. On the contrary, we fully expect that manufacturers committed to big bus technology will continue to produce capable and innovative systems for some time to come. In fact, some recent designs have achieved low latencies by using clever arbitration schemes and by eliminating wasted cycles in their bus protocol. However, investments in big-bus technology seem to be approaching a point of diminishing returns. Therefore, alternative architectures are likely to have greater potential.

## Alternative Architectures

We considered four architectures for commercial systems: shared nothing, hierarchical bus, cache-only memory architecture (COMA), and CC-NUMA. Each of these architectures is discussed below.

### Shared Nothing

Shared nothing, or massively parallel processing (MPP), architectures, such as the Intel Paragon® and the Pyramid Reliant® 1000, are attractive from a hardware developer's viewpoint because there is no need for hardware-supported shared memory or cache coherence. These systems provide high, some might even say massive, levels of performance for compute-intensive applications with statically partitionable data and minimal node-to-node communication.

Unfortunately, if a commercial database is statically partitioned, as is required for good MPP performance, then inevitable changes in database structure can result in data skew, which severely degrades

[4] A more realistic example would be complicated by the effects of memory size and of memory and disk contention.

performance.  The repartitioning required to repair data skew is too expensive to be performed online in a continuously available environment.  It is conceivable that database management systems (DBMSs) will eventually solve the problem of data skew, thereby enabling a full migration to the shared-nothing model.  However, for the foreseeable future, MPPs are not suitable for general-purpose commercial applications.

### Hierarchical Bus

Hierarchical bus systems, such as the proposed Gigamax  [12] from Encore, connect SMP building blocks via a higher-level bus. The third-level caches communicate via this higher-level bus and use a standard snooping protocol to maintain coherence among themselves.  Each building block has a third-level cache that holds frequently accessed data fetched previously from other building blocks.  These systems suffer from the same access latency and bus-length problems as the simple SMP systems.  These problems are exacerbated by packaging issues: the SMP building blocks include I/O controllers and memory SIMMs and therefore cannot be easily stacked on a sufficiently short backplane or midplane.  Therefore, hierarchical-bus systems are not promising candidates for large-scale commercial systems.

### COMA

At first glance, a cache only memory architecture (COMA) based system similar to the Data Diffusion Machine [4] or KSR-1 [3], appears inviting.  In a COMA system, all physical memory is managed as a cache, which is referred to as an “attraction memory”.  Managing large physical memories as caches reduces the burden on application software and operating system developers to increase locality, since the COMA hardware automatically moves data near the processor using it.  However, this data movement alone is not sufficient to guarantee good performance and scalability.  Data that is frequently modified by multiple processors must still shuttle back and forth between these processors.

In addition, COMA-based systems do nothing to prevent extremely unfortunate movements of data, such as moving a cache line[5] that will be used only by a single processor to some other node’s attraction memory.  Such unfortunate data movement is liable to occur when the hardware needs to make room for a newly accessed piece of data on a node that has no unused attraction memory.

Finally, in order to map memory blocks to different physical addresses over time, COMA systems require a modification to either the memory controller or the virtual memory subsystem, as in Simple COMA [11].  Thus, any system that is to run existing commodity operating systems must provide hardware support for COMA, which is not available in current glueless SMP building blocks.

### CC-NUMA

CC-NUMA systems, e.g. DASH[7], FLASH[6],Alewife[1], HP-Convex Exemplar, and the SGI Origin, provide a viable method of scaling using small single-processor or SMP building blocks without the drawbacks of the other options.  The familiar shared memory programming paradigm is maintained.  Software that runs well on large-scale shared-memory systems can ignore the NUMA characteristics of the system, provided there is enough cache at the building block level to hold a large fraction of the remote references. Furthermore, restructuring the OS to add NUMA-aware page-allocation, -replication, and -migration policies can increase performance while further reducing the need for NUMA-awareness in application software.  When combined with sufficient cache for remote memory, this OS restructuring eliminates any advantage of COMA over CC-NUMA [2].

Finally, these systems can be built using the standard processor and memory controller chip sets, assuming the SMP bus of the building block allows some form of out-of-order response and allows memory requests to be handled by a bus agent other than a processor or memory controller.

## The Sequent Approach

The advantages of the CC-NUMA approach over the shared-nothing and hierarchical-bus approaches for commercial computing are clear.  The decision to use CC-NUMA instead of COMA was motivated by: (1)

[5] In order to simplify hardware and increase performance, most modern computer systems divide memory up into equal-sized “cache lines” that are moved around the memory subsystem as a unit.  Cache line sizes are typically a power of two ranging from 16 to 128 bytes.

our desire to leverage the commodity quad SMP chipset, including the memory controller, and (2) by the fact that, in the high-end UNIX environment, OS enhancements can provide all the benefits of COMA.

Sequent's CC-NUMA design, NUMA-Q, uses multiple 4-processor SMP building blocks, or *quads*, each containing processors, caches, memories, I/O busses, and a high bandwidth bus that provides low-latency memory access. The interconnect between multiple SMP quads is based on the Scalable Coherent Interface (SCI) specification [5]. SCI's point-to-point interconnect breaks the length restriction of the single shared bus, since signals no longer need to propagate throughout the entire system in a single clock cycle. Although remote memory accesses crossing the SCI interconnect suffer higher latency, the vast majority (measured at 99.7% in an OLTP workload) of memory references complete within the originating quad, so that average latency remains low.[6] This approach combines high scalability with low latency, in constrast to shared-bus systems' strict trade-off between scalability and latency.

The Sequent approach also allows mixed systems to be constructed. For example, Pentium-Pro®, Xeon, and Tanner quads may be interconnected and booted as a single SMP OS instance. This allows easy upgrade of existing systems without wholesale equipment swaps.

Figure 2 shows the high level block diagram of the STiNG architecture. Every processor in every quad has a common view of the system-wide memory and I/O address space. Within a quad, cache coherence is maintained using a standard Modified-Exclusive-Shared-Invalid (MESI) snooping cache protocol. Each quad contains a bridge board (called IQ-Link) that connects the local SMP bus to the SCI ring. Included on the board is a 32-Mbyte 4-way-set-associative remote cache which maintains copies of blocks fetched from remote memories. This board implements a directory-based cache protocol based on SCI as well as an interface between this SCI-based global cache-coherence protocol and the MESI local-bus protocol, thereby maintaining local and remote cache coherence. In addition, the board provides mechanisms for bridging interrupt messages to remote quads and provides remote diagnostic access.

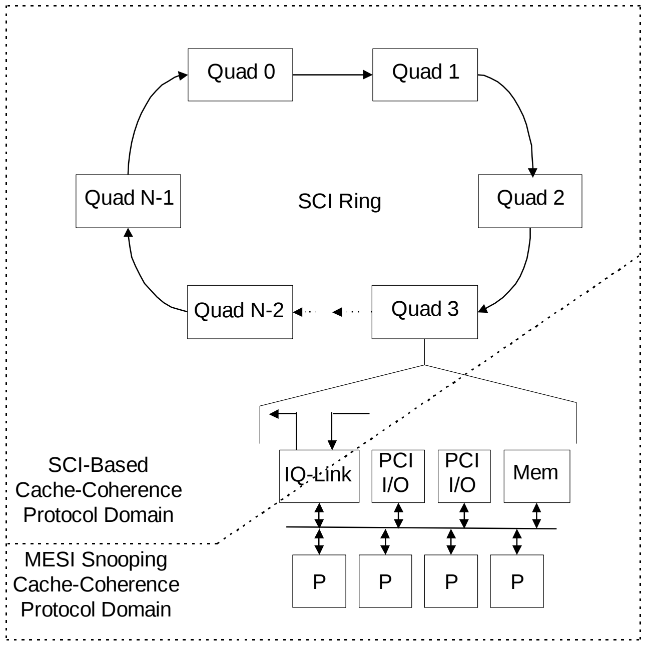

**Figure 2: NUMA-Q block diagram.**

[6] Throughout this paper, averages are weighted by frequency of occurrence in the dynamic reference stream. Instructions that are executed more frequently receive a higher weighting in the average.

NUMA-Q must run conventional business applications and shrink-wrapped operating systems (e.g. Microsoft NT [9]) that conform to future versions of Intel's Multiprocessor Specification (MPS). To achieve this:

1) All I/O devices may be programmed from any quad.
2) All processors must be capable of sending interrupts to any other processor, including those on remote quads, using the standard Intel model.
3) All system memory is accessible to both processors and DMA devices on all quads, except for small memory regions that are used for diagnostic purposes. This unrestricted access is necessary in order to obtain acceptable performance from shrink-wrapped operating systems and commercial DBMSs [3, 11].

Since the hardware met these three requirements, we only needed to add about 1,500 lines of initialization code to our existing SMP UNIX operating system in order to run it on NUMA-Q. This version of our OS provided a stable base with which to do hardware debug. This debug version of the operating system did not run fast, but it did run on the new NUMA-Q hardware. If anything, the lack of optimization helped the hardware debug process by increasing the level of stress on the IQ-Link board.

First-cut NUMA performance optimizations were tested in a NUMA-emulation environment running on older, stable SMP hardware. These optimizations consumed a significant amount of time and effort, produced a significant amount of code, and are the subject of a future paper.

This concurrent debug strategy allowed us to ship NUMA-Q units for revenue within a few days of finalizing the production hardware. Further work on performance optimizations is ongoing and expected to continue for some time.

## Implementation Overview

This section briefly summarizes the initial implementation of the IQ-Link for NUMA-Q. For more detail, refer to [8].

A high level block diagram of the IQ-Link board is shown in Figure 3. The IQ-Link contains three major integrated circuits: the DataPump chip, co-developed by Sequent and Vitesse Semiconductor; the SCLIC ASIC, developed by Sequent; and the OBIC, also developed by Sequent. The IQ-Link also maintains remote-cache-tag and local-memory-directory information for the SCLIC and OBIC. The use of this information is described below, followed by a brief discussion of each of the three ICs.

Two sets of tags are provided close to the local SMP bus to participate in the local snooping protocol. The bus-side local directory SRAM contains two bits of state information for each block in the local memory. These directory-state bits indicate whether each line is: HOME, meaning no other remote cache contains a copy; FRESH, meaning other remote caches contain a read-only copy; or GONE, meaning that other remote caches contain a writable copy and no valid copy is on the local quad. These bits are implemented in SRAM so that snoops can be performed at bus speed. This high-speed bus-side local directory SRAM need not contain any quad ID information, and can therefore be quite small.

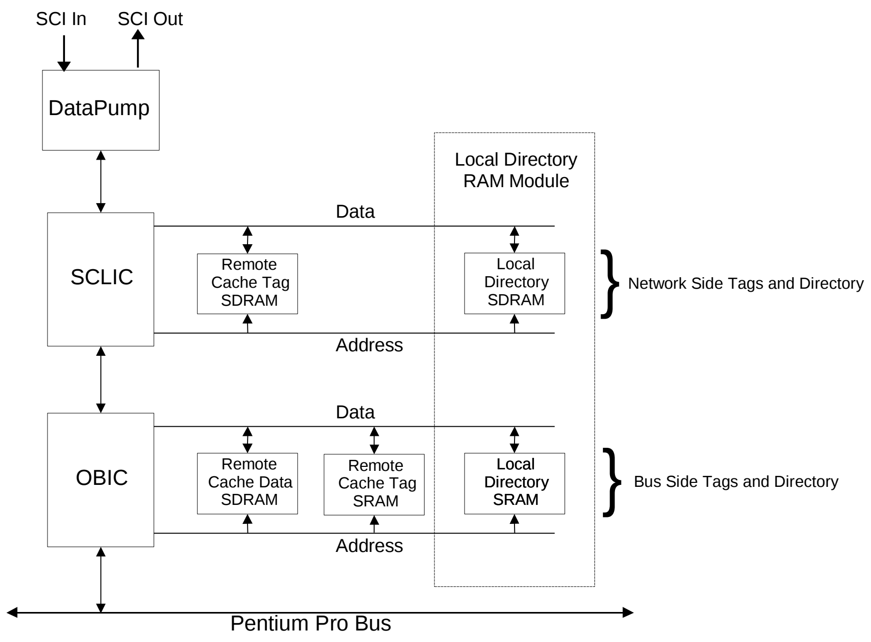

**Figure 3: IQ-Link block diagram.**

The bus-side remote tags provide snooping information for the lines in the 32-MB remote cache, indicating whether a given bus request to remote memory can be satisfied locally from the cache. These are also implemented in SRAM and contain only state information. Bus accesses that hit in the remote cache can be satisfied with a latency similar to that of the local-memory latency. Accesses to remote memories that miss the remote cache are passed to the SCLIC.

The network-side local-memory directory and the network-side remote tags are used in the SCI-based directory-based protocol. These two sets of tags parallel the bus-side local directory and the bus-side remote tags. These duplicate tags reduce communication between the SCLIC and OBIC, thereby reducing latency. Each block in the local memory is represented in the network-side local directory by a 2 bit state field and a 6 bit list head pointer[7]. For each block in the remote cache, the SCLIC maintains a 7 bit state field, a 6 bit forward pointer, a 6 bit backward pointer and a 13 bit address tag. All of the network-side directory and tag information is maintained in SDRAM storage.

The DataPump is a GaAs chip that implements the link layer of the SCI protocol. Its role is to put SCI packets received from the SCLIC on the ring, provide a bypass path for SCI packets targeted to other quads on the ring, and to strip those SCI packets from the ring that are targeted to the local node's SCLIC. SCI packets are transmitted over the ring as multiple 2 byte *symbols*. The DataPump can route the first symbol of a packet to either the SCI output port or the interface to the SCLIC ASIC in 16ns. Each additional symbol is routed in one clock cycle, or 2ns.

The SCLIC is a CMOS ASIC that implements the SCI cache-coherence protocol and provides an APIC bridge for routing interrupts between quads. This part has an embedded programmable instruction sequencer which enables the protocol processing to be implemented in software. The SCLIC has multiple register sets to support 12 read/write/invalidate transactions and one interrupt transaction concurrently. The SCLIC also has a small tag cache which contains state for requests currently in progress.

The OBIC is a CMOS ASIC that implements the Pentium Pro® bus interface and controls the remote cache which caches data from memory located on remote quads. The OBIC *snoops* every transaction on the Pentium Pro bus, and *defers* those that can not be completed in order. These out of order transactions are passed to the SCLIC for processing, and usually generate one or more SCI commands to be issued to

[7] The 6-bit pointer limits NUMA-Q to 64 quads, rather than the 16,384 quads that are specified by the SCI standard. We do not expect this to be a practical limitation in the near future: NUMA-Q systems may be clustered as needed to achieve increased performance as well as higher availability. Our implementation deviates from SCI in a few other areas, one of which is described in a later section.

the SCI ring.  When a response is returned to the OBIC, a *deferred reply* transaction completes the original request.  The OBIC determines which requests must be deferred by consulting the remote cache tags or local directory, which tracks the state of remote cache lines and local memory lines respectively.  Also, the OBIC is responsible for servicing incoming requests from remote quads.  It does this by placing transactions on the Pentium Pro bus, which may target the remote cache, local memory, a processor's cache, or one of the PCI busses.

## Special Problems

We encountered any number of quirks, catch-22s, and interesting problems while implementing NUMA-Q.  The following sections describe three of these problems and how we solved them.  These problems are I/O locality, difficulties with the SCI cache-coherence protocol, and forcing several machines, each of which was intended to act as a standalone PC, to instead work together to emulate a large SMP machine.

## I/O Locality

If each disk of a CC-NUMA system were connected to a single node, an application would ideally be optimized such that each node accessed only its local disks, as shown in Figure 4.  Such optimizations are similar to the partitioning required by MPP systems.   An important difference is that MPP requires the entire application to be partitioned, but a CC-NUMA system requires only the most heavily exercised portions be partitioned.  This section describes how NUMA-Q can efficiently run many applications *without* the need to partition their I/O.

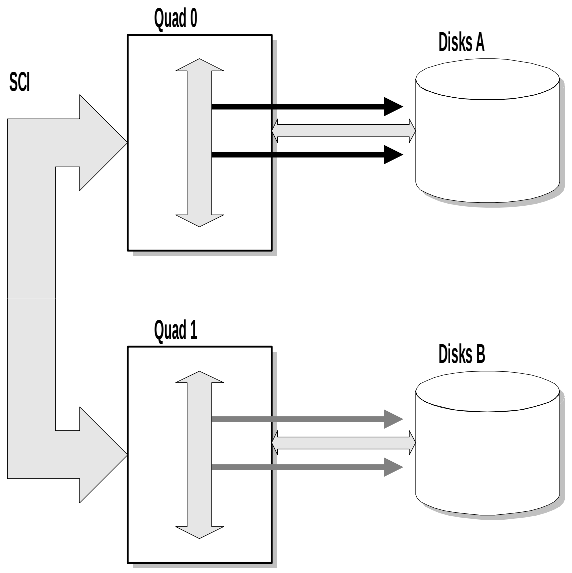

**Figure 4:  I/O From NUMA-Aware Application**

For example, the high-bandwidth, low-latency SCI ring enables many unpartitioned applications to attain adequate performance, as shown in Figure 5.  Other applications may be limited by the additional latency imposed by the SCI ring and its interaction with the PCI I/O bridge chipset.

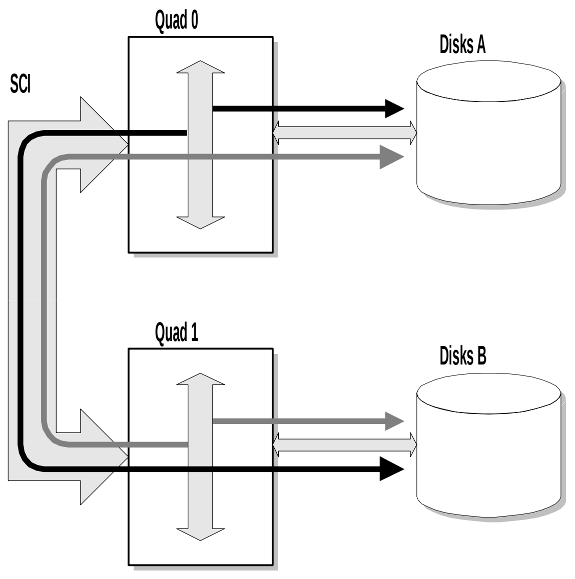

**Figure 5: I/O From Unoptimized Application**

To address the needs of these other applications, NUMA-Q allows each quad to locally access all disks. This is accomplished by using Fibre Channel switches, as shown in Figure 6, and by providing support in the operating system for steering I/Os to the "local path". Fibre Channel is specifically designed to transfer large I/Os with low latency, and allows for up to 10KM links between a NUMA-Q host and its disks, or between multiple NUMA-Q hosts cooperating as a loosely-coupled cluster. In addition, the operating system provides failure tolerance by using a "remote path" if a local path becomes unavailable. This approach enables NUMA-Q to better optimize the use of its cache-coherent interconnect resources, reduce the average latency of both I/O and memory accesses, be significantly less sensitive to application I/O access patterns, and provide enhanced system availability as well.

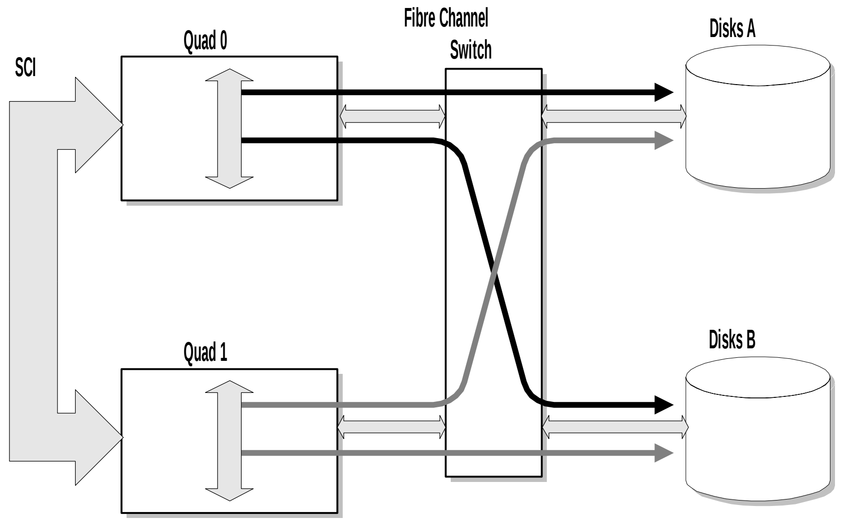

**Figure 6: I/O From Unoptimized Application With Fibre Channel Switch**

This switch-based I/O subsystem allows many applications to run efficiently without partitioning their I/O load. Applications that do large I/Os (such as decision-support applications) gain the most benefit. More importantly, it allows specialization. The design of the SCI subsystem is optimized for memory traffic, while that of the Fibre Channel switch is optimized for I/O traffic.

Equally important is the ability to configure the system with redundant paths to storage. A system configured as shown in Figure 7 can tolerate a failure of any component in the I/O fabric without loss of data or decrease in performance; the OS will automatically reroute requests around the failure. I/O-fabric components may be repaired in a hot-swap fashion, with no down-time required. This I/O-fabric redundancy and resiliency greatly increases system availability.

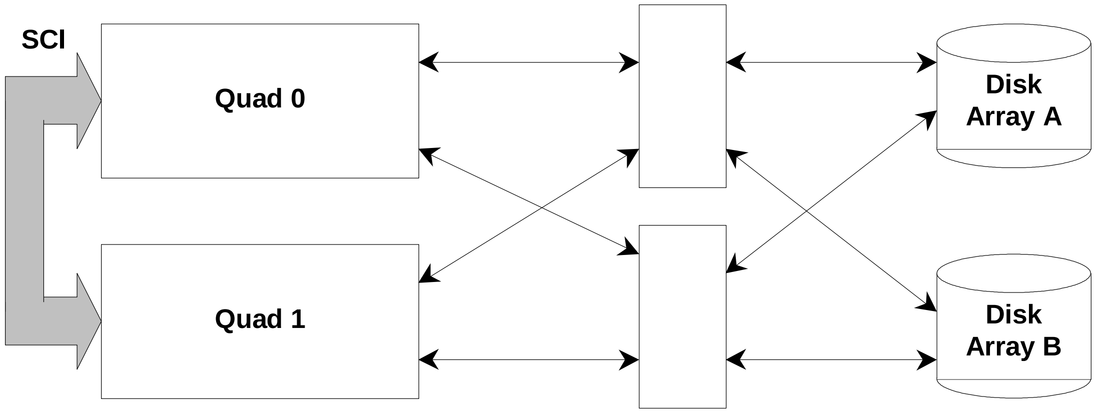

**Figure 7: Redundant Paths to I/O Devices**

## Difficulties With the Standard SCI Cache-Coherence Protocol

The SCI cache-coherence protocol was designed for systems with separate processing and memory elements. In particular, if a given cache line is “HOME”, then this line will be located only in its memory module and thus will not be found in any CPU’s cache. In contrast, NUMA-Q’s memory and processors are co-resident, as required for the low-latency memory access demanded by today’s high-speed microprocessors. Therefore, a cache line’s “HOME” quad has four processors, any or all of which might have cached that particular line.

The way to resolve this difference is to carefully merge the states of the P6 bus protocol with those of the SCI cache-coherence protocol. If a standard SCI memory module were to receive an SCI command indicating that a processor wished to modify a cache line not present in any processor’s cache, the memory module would mark its own copy of the cache line “GONE” and send a reply message containing the desired cache line. However, if a NUMA-Q quad receives this same message, it would also need to issue P6 bus transactions in order to ensure that none of its local processors have a copy of the cache line. In addition, the quad must deal with races that can occur if one of the local processors decides to modify the same cache line at about the same time as the SCI message is received.

More details on issues surrounding NUMA-Q’s adaptation of the SCI protocol may be found in [10].

## Making PCs Work Together

Since NUMA-Q is constructed from commodity 4x PentiumPro multiprocessor PCs, one of the early problems that needed to be solved was the construction of a shared memory map. Construction of a multi-quad shared memory image occurs after the standard BIOS[8] executes, but before control is passed to the operating system loader with a description of the unified memory image. A key design decision was that all resources be globally sharable, giving even the lowest level system software the appearance of a large SMP system. This decision allowed us to improve memory locality in an incremental fashion rather than forcing wholesale software rewrites simply to boot the operating system. We were successful in running an OS constructed for a 1 quad system on an 8 quad machine with only minor changes to the lowest level initialization routines.

[8] “BIOS” stands for “Basic Input/Output System, and is the mechanism by which PC-based I/O boards (and other hardware) are initialized and operated. Performance considerations force us to bypass BIOS for normal I/O operations, however, we do rely on BIOS to initialize the I/O boards. Allowing BIOS to do this initialization insulates the operating system from many low-level details of the I/O-board chipsets.

Leveraging a standard BIOS in each quad allowed us to take advantage of standard PC style memory layout for diagnostics and support for commodity operating systems. Since each quad thinks it is an independent machine, a “meta-BIOS” layer needs to build the shared memory image. The meta-BIOS is responsible for performing interconnection tests, initializing the IQ-Link cache and tags, laying out the physical memory map and constructing the system configuration tables for use by the OS loader. A copy of the meta-BIOS along with machine microcode is contained in flash disk on each quad along with a description of the memory map. In the event of hardware failure or reconfiguration, a management diagnostic processor (MDC) is present in each quad to act as a back door to program the flash disk, report and diagnose errors, and monitor hardware state.

## Summary

The traditional big-bus SMP architecture is approaching a point of diminishing returns. We therefore evaluated four alternative architectures: shared nothing, hierarchical bus, COMA, and CC-NUMA. We chose CC-NUMA because of its ability to provide both high throughput and low latencies when used with commercially available hardware, operating systems, and applications. This CC-NUMA machine, named NUMA-Q, was initially based on quad-Pentium-Pro building blocks connected to an SCI ring, forming a large-scale shared-memory multiprocessor. The second-generation product is based on quad-Xeon building blocks, and is backward compatible with the first generation.

Experience with NUMA-Q since its introduction in 1996 has been very encouraging. We expect it to continue to do quite well, particularly on OLTP, decision-support, and related applications, through at least the next decade.

## Performance (Sidebar)

The initial implementation of NUMA-Q showed up to a 6-fold increase in performance over its predecessor.[9] In late 1998, the second-generation Xeon®-based NUMA-Q system set a record of 93,900 TPM/C, beating the previous best mark by more than 40%. Although this mark has since been surpassed, it is still in the top 5 as of this writing.

Performance simulation results presented in [8] show how system throughput is limited by the latency of second-level cache misses. Latency measurements on real hardware differ somewhat from the simulated results. First, the latency to complete an in order request that hits the remote cache or is satisfied by local memory is somewhat higher than reported in [8], and this is due in part to the use of different processor and bus frequencies than those assumed in the simulations. With a 180MHz Pentium Pro processor and 60MHz bus, this latency is between 400 and 500ns at loads between 110 and 290MB/s (this latency decreases to 130ns best-case on an idle machine). On the other hand, the latency for a remote reference is considerably less than reported in [8], because the NUMA-aware OS modifications improved the application’s memory locality more than expected, and because the local memory controllers introduce more latency than expected. This better locality and greater latency prevents saturation of the SCLIC protocol processor, thereby reducing remote latency. In general, this remote-reference latency is highly workload dependent, because varying the load placed on the IQ-Link causes the latency to vary widely. While running a stress test that caused a maximum number of remote transfers to occur, the latency for requests that missed the remote cache was measured to be about 8.4us. For a TPC/B[10] workload, the average remote-access latency is approximately 5.8us for a 4-quad system (this latency decreases to 2.5us on an idle machine). These data points suggest that typical remote-to-local-latency ratios will be approximately 12 to 1.

It is important recall that system throughput for a processor-bound workload is limited by the latency to satisfy <u>all</u> second-level cache misses, not just those that require remote memory access. With the better than expected local memory locality observed for TPC/B runs in the lab, the average time to satisfy a processor L2 cache miss is about 720ns for a 4-quad system and 590ns for a 2-quad system.[11] This latency

[9] The predecessor to NUMA-Q is the Symmetry-5000 line of symmetric multiprocessors. These machines combine Pentium processors in a Big Bus architecture.

[10] TPC/B benchmarks were run to validate simulations and to make meaningful comparisons to the older Symmetry-5000 machines.

is similar to that observed under heavy load in many large single bus systems today which can not support as many processors, I/O controllers, or physical memory locations as NUMA-Q can.

The second-generation Xeon-based systems offer greatly increased performance. Remote latency has improved by a factor of two and local latency has improved by 50%. Further substantial improvements are being implemented in upcoming generations of the NUMA-Q architecture.

More detailed IQ-Link performance data will be published in the future, including detailed workload profiles and bandwidth consumption at the different datapaths in the system.

## The Future of Parallel Processing (Sidebar)

The requirements for high-end computing will continue to exceed that which can be obtained from commodity technology. First, many important applications solve exponentially-complex problems such as the traveling-salesman problem or the bin-packing problem, problems for which exponential increases in computing power provide only incremental increases in problem size. Second, applications as diverse as gaming, data warehousing, and climate modeling could benefit from far greater computing power and I/O capacity than can be brought to bear today. Third, Internet is the precursor of a global computing infrastructure that will require heavy computing equipment able to meet the demands of an entire world full of ever-more-powerful personal computers. Finally, users will continue to demand ever-increasing reliability and availability. These ever-increasing requirements, combined with the compelling price-performance of commodity equipment, will continue to favor CC-NUMA architectures.[12]

But will commodity processor performance continue its exponential increase? If so, a CC-NUMA architecture that combines ever-more-capable commodity components is favored. If not, life will become quite interesting. Performance will stagnate first for “ill-behaved” real-world applications that exhibit poor memory locality or that cannot exploit clever advances in microarchitecture, and later for even the best-behaved benchmarks.[13] This stagnation will further increase the importance of parallel processing, perhaps inspiring single-chip multiprocessors that might form the basis for a future CC-NUMA architecture.

However, stagnating performance would also renew interest in novel architectures. If viable, such architectures would appear first as coprocessors, where they could be specialized to a particular task, thereby preserving existing software investments and reducing implementation effort and risk. Such coprocessors could slow or perhaps even reverse the current movement of functionality into the processor chipset.

Either way, we will enter the new millennium with no shortage of useful tasks to accomplish or of interesting problems to solve.

## References

[1] A. Agarwal, R. Bianchini, D. Chaiken, K. L. Johnson, D. Kranz, J. Kubiatowicz, B.-H. Lim, K. Mackenzie, and D. Yeung. The MIT Alewife machine: Architecture and performance. In

[11] This average latency is the time-series average of the latencies of reference that actually took place while running an OLTP workload.

[12] The reduced parts count enabled by the commodity SMP architecture of the quads provides increased reliability and availability. Applications requiring even higher levels of reliability and availability use clustered configurations.

[13] In fact, those who believe that microprocessor performance is stagnating might argue that there is an ongoing search for better-behaved benchmarks so that CPU designers can at least meet the letter, if not the spirit, of Moore’s Law. The recent tendency to measure CPU performance on extremely well-behaved multimedia benchmarks might be held up as but one example of this search. Others might note that commercial benchmark numbers are continuing to increase rapidly. Still others might counter that much of the improvements in commercial benchmarks are due to software optimizations, rather than just faster hardware.

*Proceedings of the 22nd International Symposium on Computer Architecture*, pages 2-13, June 1995.

[2] J. Chapin, S. A. Herrod, M. Rosenblum, and A. Gupta. Memory system performance of UNIX on CC-NUMA multiprocessors. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems,* pages 1-13, May 1995.

[3] S. Frank, H. Burkhardt III, and J. Rothnie. The KSR1: Bridging the gap between shared memory and MPPs. In *Proceedings of the 38th IEEE Computer Society International Conference (Spring Compcon),* pages 285 - 294, February 1993.

[4] E. Hagersten, A. Landin, and S. Haridi. DDM - A cache-only memory architecture. *IEEE Computer,* pages 44-54, vol. 25, no. 9, September 1992.

[5] IEEE Computer Society. *IEEE Standard for Scalable Coherent Interface (SCI),* IEEE Std 1596-1992, New York, New York, August, 1993.

[6] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy. The Stanford FLASH multiprocessor. In *Proceedings of the 21st International Symposium on Computer Architecture,* pages 302-313, April 1994.

[7] D. Lenoski, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta, and J. Hennessy. The DASH prototype: Logic overhead and performance. *IEEE Transactions of Parallel and Distributed Systems*, pages 41-61, vol. 4, no. 1, January 1993.

[8] T. Lovett and R. Clapp. STiNG: A CC-NUMA computer system for the commercial marketplace. In *Proceedings of the 23rd International Symposium on Computer Architecture*, pages 308-317, May 1996.

[9] S. Perl and R. Sites. Studies of Windows NT performance using dynamic execution traces. In *USENIX Association Second Symposium on Operating Systems Design and Implementation (OSDI'96),* pages 169-183, November 1996.

[10] R. Safranek, Considerations in implementing a system based on SCI. In *Proceedings for The Fourth International Workshop on SCI-based High-Performance Low-Cost Computing*; pages 12-22, July 1996.

[11] A. Saulsbury and A. Nowatzyk. Implementing simple COMA on S3-MP. Presentation at *The Fifth Workshop on Scalable Shared Memory Multiprocessors*, Santa Margherita Ligure, Italy, June 1995. http://playground.sun.com:80/pub/S3.mp/simple-coma/isca-95/present.html

[12] A. W. Wilson, Jr. Hierarchical cache/bus architecture for shared memory multiprocessors. In *Proceedings of the 14th International Symposium on Computer Architecture*, pages 244-253, June 1987.