Multiprocessor Design Options and the Silicon Graphics S^2MP Architecture

Daniel Lenoski

Silicon Graphics, Inc.

lenoski@sgi.com

Most multiprocessors can be classified as one of four types: message-passing or MPP systems, clusters, shared-memory multiprocessors, and parallel vector processors. This paper discusses a new model, the scalable shared memory multiprocessor (SSMP), which removes the drawbacks of the traditional SMP systems. The first part of this paper reviews current multiprocessor alternatives, and introduces the scalable SMP architecture. The second part of the paper focuses on a particular instance of a scalable SMP, the Silicon Graphics Origin multiprocessor and its S^2MP memory architecture. We give an overview of the Origin system and then discuss some of the important design trade-offs in the architecture and implementation of the system. In the final section we examine technology trends and predict that clusters of scalable shared-memory multiprocessors will become the dominate multiprocessor architecture.

1 Traditional Parallel Architectures

While a wide-variety of parallel machines have been developed over the years, the most widely-used multiprocessor machines fall into four categories. The first is the message passing, or massively parallel – MPP architectures. The second is clusters. Clusters are similar to the MPP systems, but their computing nodes are based on high volume workstations or small scale SMP nodes. The third class of system is shared memory multiprocessors, and the last class of system is parallel vector processors.
MPP machines are also referred to as a distributed memory systems. They consists of a collection of CPU-memory nodes that are connected by a high speed interconnection network. The structure of an individual node is similar to a stand alone computer, except the nodes are usually smaller, and most of them are not connected directly to I/O devices. Generally, the packaging of these systems is geared for a large processor count. Examples of this class of

machine include Intel's Paragon, Thinking Machines CM5, and the Cray T3D and T3E.
The strength of MPP systems lies in their scalability. The fact that the nodes are small and are connected by a high-speed interconnection network, allows these systems to grow to hundreds or even thousands of processors. The general drawback is that programming them requires restructuring the application into a message-passing style. This requires that programmers rewrite their programs to explicitly manage all data communication. In addition, performance often suffers since the overhead of passing a message is measured in tens to hundreds of microseconds; which is tens of thousands of instructions on a modern microprocessor. These performance and programming overheads have limited the use of these machines to a small user base who could justify the effort to recode their application in return for the high aggregate computing power of a large MPP.
Clusters address the volume issues of MPPs by replacing the integrated MPP nodes with standard workstation or SMP nodes. Systems in this class include IBM's SP series, the DEC TruCluster systems, and SGI's PowerChallenge Arrays. These systems are popular because they can leverage the volume of their individual nodes to hit better price- performance points. The structure of these machines differs from that of MPPs in the sense that the interconnection network connects to the I/O subsystem instead of being integrated into the memory bus. Physically, these machines are generally not as tightly packaged as the MPP machines since the nodes each have I/O controllers, disk, etc.
Unfortunately, the fact that clusters are less integrated, in terms of their global interconnect, implies that the overhead of communicating between nodes of the cluster is higher than in MPP systems. Clusters also suffer from the same problem of message-passing programming and performance overheads as MPP systems do.
The next class of system, the shared- memory multiprocessor or symmetric multiprocessor, is quite different from the first two. Generally, SMPs combine a number of processors on a high performance bus that provides both high bandwidth and low latency to central memory and I/O devices. These systems employ snoopy cache coherence to keep processor utilization high and reduce bus loading. There are numerous examples of this class of system, ranging from the high-end SGI PowerChallenge, Sun Ultra Server, and DEC Alpha Server to the low end, two and four-processor PC systems. The SMP system's primary advantage is the shared memory programming model, which is a more natural extension to the uniprocessor model than the message-passing model. Shared memory also permits low latency interprocessor communication. Finally, the large central memory and I/O resources in an SMP are directly accessible to all processes running on the system, unlike the distributed resources of an MPP or cluster.
The last class of MP system is the parallel vector processor or PVP. PVPs differ from the other classes in the sense that they are based on specialized vector processors instead of high-volume microprocessors. They are also differ-

ent in that the vector processors operate directly out of a high-speed memory without intermediate caches. They can achieve high throughput by hiding the latency to memory by operating on vectors instead of individual data words. The primary example of this class of system is Cray's line of vector machines including the J90, C90, and the latest T90. The high-end vector machines are based on bipolar technology and utilize very high bandwidth interconnect to an SRAM main memory. This makes PVPs unique in the sense that their performance can remain high on codes that cannot use caches effectively. Unfortunately, they also suffer from high costs and low volume due to their special purpose nature. However, PVPs do serve some very important scientific applications that need very high performance, but can not utilize caches well.

The ideal multiprocessor would combine the best of all of these architectures. It would provide the scalability of MPP systems; provide the cost economics of a cluster-based system; provide the programming model and tight coupling of an SMP and provide the floating point performance and high memory bandwidth of the parallel vector processors.

2 Scalable Shared-Memory Multiprocessing

A scalable SMP (SSMP) restructures the SMP class to incorporate the advantages of the other architectures while retaining the programming model and low latency communication of the SMP.

To understand the structure of an SSMP, let's begin by looking at the SMP in more detail. The bus structure of the SMP is key to its tight integration and cache coherence, but also to the scalability limit of the system. The cost of the bus itself limits how small a system can effectively configured. The fixed bandwidth of the bus limits how far the SMP can scale to support a large number of processors.

The first step in the evolution of the SMP is to remove the bus, and replace it with a switch. The switch removes the bus bottleneck by giving the system scalable bandwidth that can grow as the system grows. Further, the switch can be small to reduce costs when the system is small, and grow as the system grows. One additional change is that we must also change the means of cache coherence since the snoopy schemes used on bus-based SMPs rely on broadcasting every memory reference to every cache. This is done by adding directories to memory so that the memory knows which processors hold a copy of a particular memory block and can send point-to-point messages to keep caches coherent.

The overall effect of replacing the bus with a switch is that the bus bottleneck is removed and the system is much more scalable. We have also increased modularity, in the sense that the switch structure can be grown as the numbers

of processors are grown in order to provide more interconnect performance. But we have still not achieved the ideal structure because the switch still adds latency and uses shared bandwidth for memory locations that are accessed only by a single processor.
The solution to this is to push portions of the memory through the switch and distribute the shared memory and I/O with sets of processors. With a distributed-shared- memory or DSM structure, memory and I/O that has affinity to a given set of processors can be accessed with lower latency and does not use the shared bandwidth of the global interconnect. Memory bandwidth increases naturally as processors are added. Further, modularity is greatly increased because each "node" is a complete functioning unit and an entry system need not have a global switch at all. DSM machines are also referred to as NUMA or non-uniform memory access machines. This is in contrast to traditional bus-based SMP or switch-based PVP systems where all memory is equally distant and there is uniform memory access or UMA. NUMA architectures that support caching of local and remote memory are referred to as CC- NUMA for cache-coherent NUMA.
The DSM or CC-NUMA system has the same basic structure, and thus scalability, as a MPP or workstation cluster. The primary difference though, is that the memory is accessible and cache-able by all processors. Furthermore, all I/O can be accessed by each processor and I/O devices can DMA directly into any portion of memory as in an SMP. All that has changed from the switch-based SMP is that memory and I/O resources have been distributed along with the processors.

3 Directory-Based Cache Coherence

Directory-based cache coherence is key to SSMP because the only processors involved with the coherence operations for a given memory block are those that have previously read or written that block. Thus, the overhead of cache coherence is never more than a fraction of the traffic required to access the memory block if it were never cached at all. Furthermore, memories and directories only interact with processors, never with one another. Thus, bandwidth to global cache coherent memory can be scaled with directories by simply adding additional memory banks, or in the case of the DSM or CC-NUMA systems, by adding additional nodes to the system.
Directory-based cache coherence was originally proposed by Censier and Feautrier in 1978[3]. However, during the 1980's, commercial SMP systems were based on snoopy cache-coherence because snoopy schemes were simpler and placed the burden of coherence on the caches themselves. Renewed interest in directory-based cache- coherence began in the academic community in the late 1980's when the inherent bottlenecks of bus-based SMP systems be-

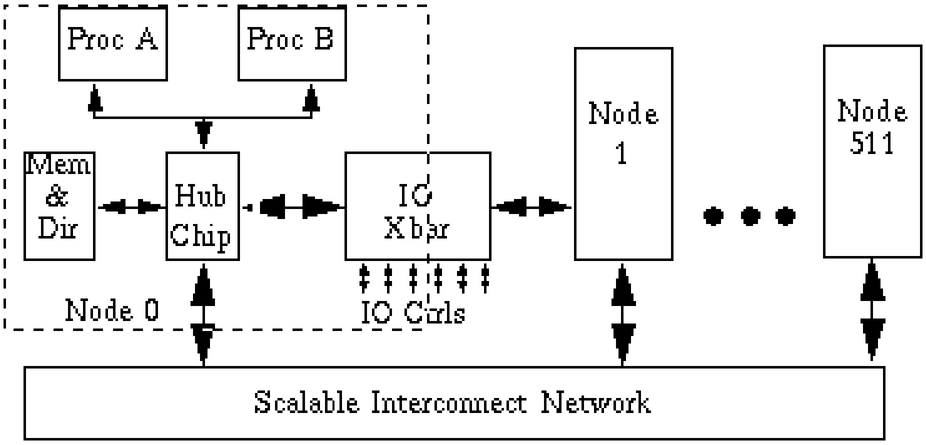

Fig. 1. Origin block diagram

gan to be felt. A few research groups built prototype machines to explore the performance and design issues with these scalable systems. Two of the major efforts were the DASH project at Stanford[6] and ALEWIFE at MIT[1]. Another early effort at a directory protocol implementation was taken on by the IEEE Scalable Coherence Interface working group[5]. This group defined an interface standard for modules that includes a directory- based cache coherence scheme that could be used to build up SSMPs out of nodes conforming to the SCI standard.

The earliest commercial based DSM system was the Kendall Square Research KSR-1 that was introduced in 1991 and the Convex Exemplar, introduced in 1993. The Exemplar was based on the IEEE Scalable Coherent Interface structure. These early machines were not very successful with KSR eventually folding; and Convex struggling financially and eventually being acquired by Hewlett-Packard. The limited acceptance of these early DSM machines was due to improved bus technology which yielded bus- based SMPs with more than a gigabyte per second of memory bandwidth, and to the fact high- performance switches could only be built with expensive bipolar or gallium arsenide technology at this time.

Today, the need for higher performance and greater scalability has driven much interest in DSM systems. Technology improvements in commodity CMOS have also made these systems much more cost- effective. Some of the announced DSM systems include SGI Origin servers, Sequent's NUMA-Q systems and Data General's NUMALiine. Convex in conjunction with HP has announced their 2nd major generation of Exemplar DSM systems, the X-class. Many other products are rumored to be in the works from other major computer vendors.

4 The Origin S^2MP architecture

A block diagram of the Origin architecture is shown in Figure 1. The basic building block of the Origin system is a dual-processor node. In addition to the processors, a node contains up to 4 GB of main memory and its corresponding directory memory, and has a connection to the global interconnect and a portion of the IO subsystem.
The architecture supports up to 512 nodes, for a maximum configuration of 1024 processors and 1 TB of main memory. The nodes can be connected together via any scalable interconnection network. The cache coherence protocol employed by the Origin system does not require in-order delivery of point-to-point messages to allow maximum flexibility in implementing the interconnect. The S^2MP architecture provides global addressability of all memory, and in addition, the IO subsystem is also globally addressable. Physical IO operations (PIOs) can be directed from any processor to any IO device. IO devices can DMA to all memory in the system, not just their local memory. The following sections summarize each of the major subsytems within an Origin system.

4.1 CPU / Memory Subsystem

An Origin node contains two MIPS R10000 processors with their secondary caches and a portion of the globally addressable memory. The Hub chip connects the two processors to the local memory as well as providing a port into the inter-node network (referred to as Craylink). Each node also contains a high-speed link to some of the IO controllers on the system.
Figure 2 shows a block diagram of the Hub chip. The peak raw data bandwidth is listed for the four hub external ports. The hub chip is divided into five major sections: the crossbar (XB), the IO interface (II), the network interface (NI), the processor interface (PI), and the memory and directory interface (MD). All interfaces communicate with each other via FIFOs that connect to a central crossbar.
The processor interface section of the hub contains logic for implementing the request tracking for both processors. Active read and write requests are tracked via a coherent request buffer (CRB), with twelve CRB entries per processor. The PI also includes the protocol table for its portion of the cache coherence protocol. The PI is responsible for controlling the flow of requests to and from the R10000 processors.
The IO interface contains the translation logic for interfacing to the XIO subsystem. The IO section also contains the logic for two block transfer engines (BTEs) which are able to perform memory to memory copies at near the peak of a node's memory bandwidth. It also implements the IO request tracking portion of the cache coherence protocol via the IO request buffers (IRB) and

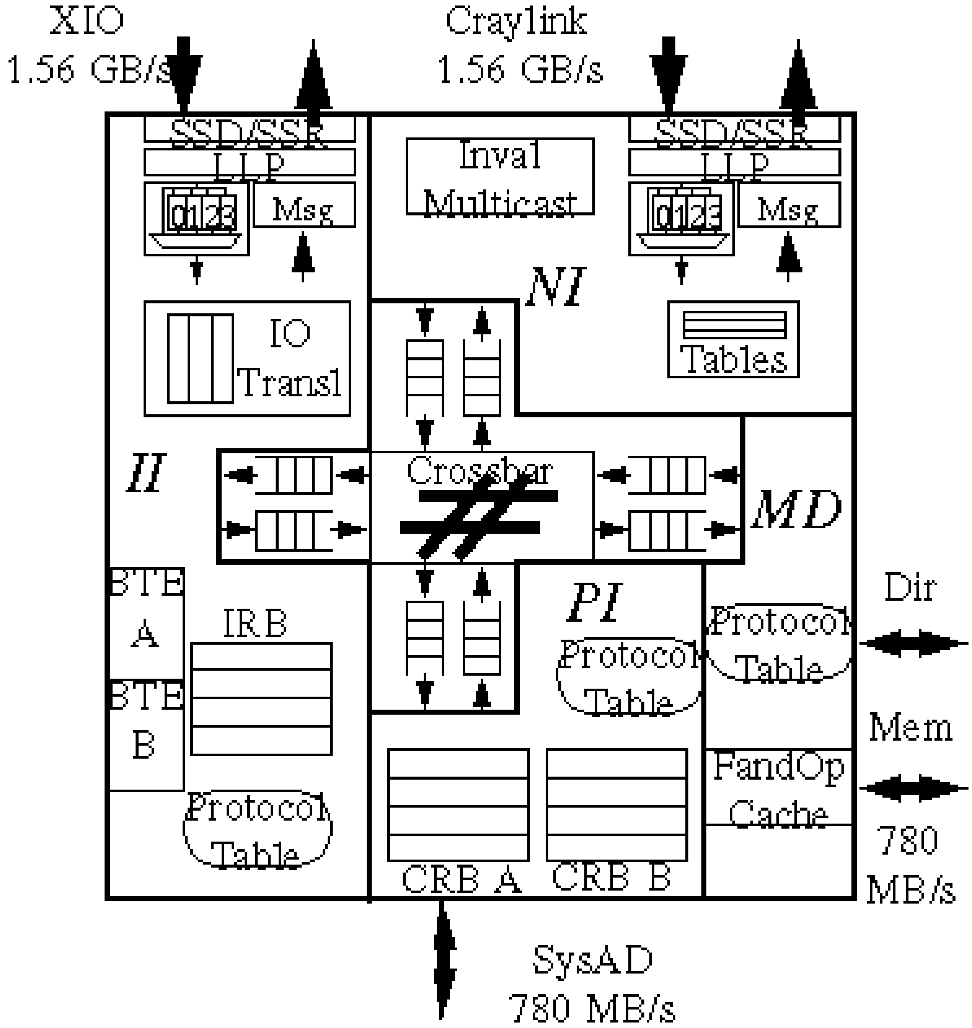

Fig. 2. Hub ASIC block diagram

the IO protocol table. The IRB tracks both full and partial cache line DMA requests by IO devices as well as full cache line requests by the BTEs.
The network interface takes messages from the II, PI, and MD and sends them out on the Craylink network. It also receives incoming messages for the MD, PI, II, and local Hub registers from the Craylink network. Software programmable routing tables are provided in the NI to support the pipelined routing used by the SPIDER chip[4].
Finally, the memory/directory section contains logic to interface to external data and directory memory implemented with synchronous DRAMs (SDRAMs). The MD performs the directory portion of the cache coherence protocol via its protocol table and generates the appropriate requests and/or replies for all incoming messages.

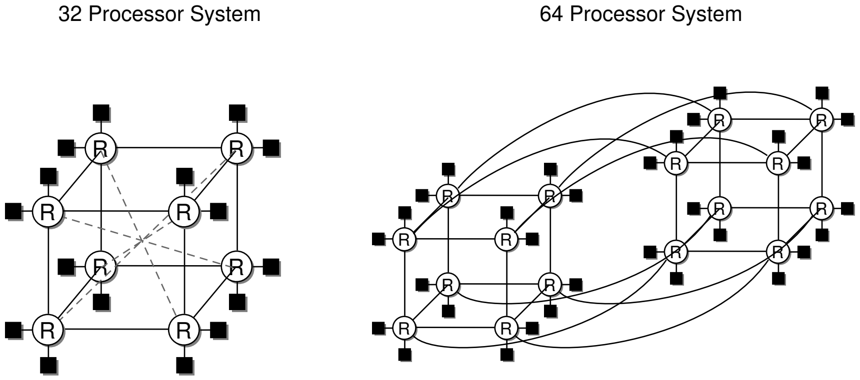

Fig. 3. 32P and 64P Bristled Hypercubes

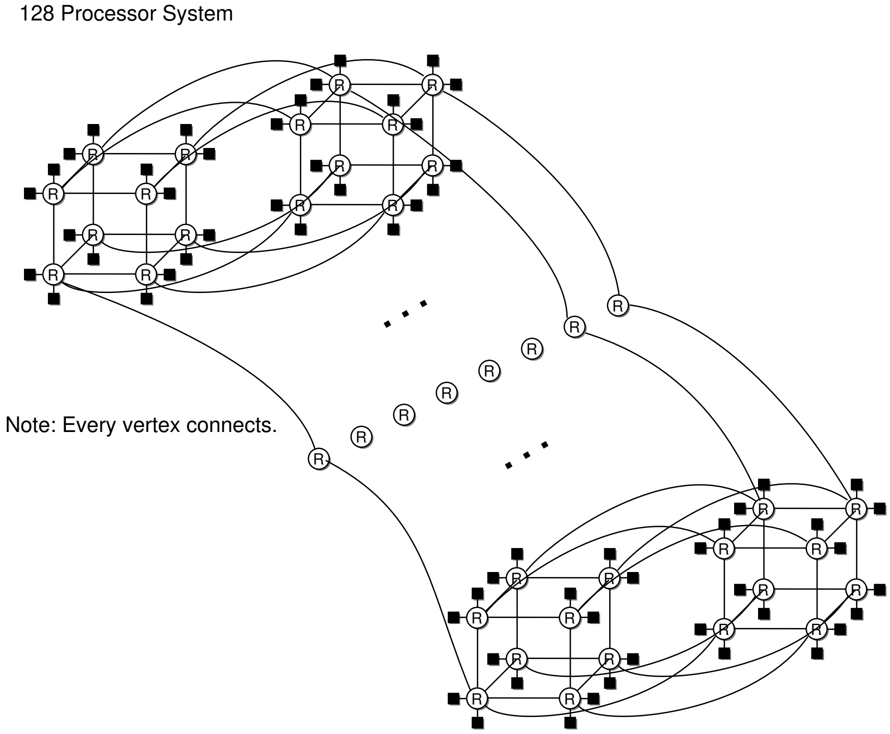

Fig. 4. 128P Hierarchical Fat Hypercube

4.2 Network topology

The interconnect employed in the Origin system is based on the SGI SPIDER router chip[4]. The main features of the SPIDER chip are:

- six pairs of unidirectional links per router
- software programmable routing tables
- low latency (41 ns pin-to-pin) wormhole routing
- DAMQ buffer structures[4] with global arbitration to maximize utilization under load.

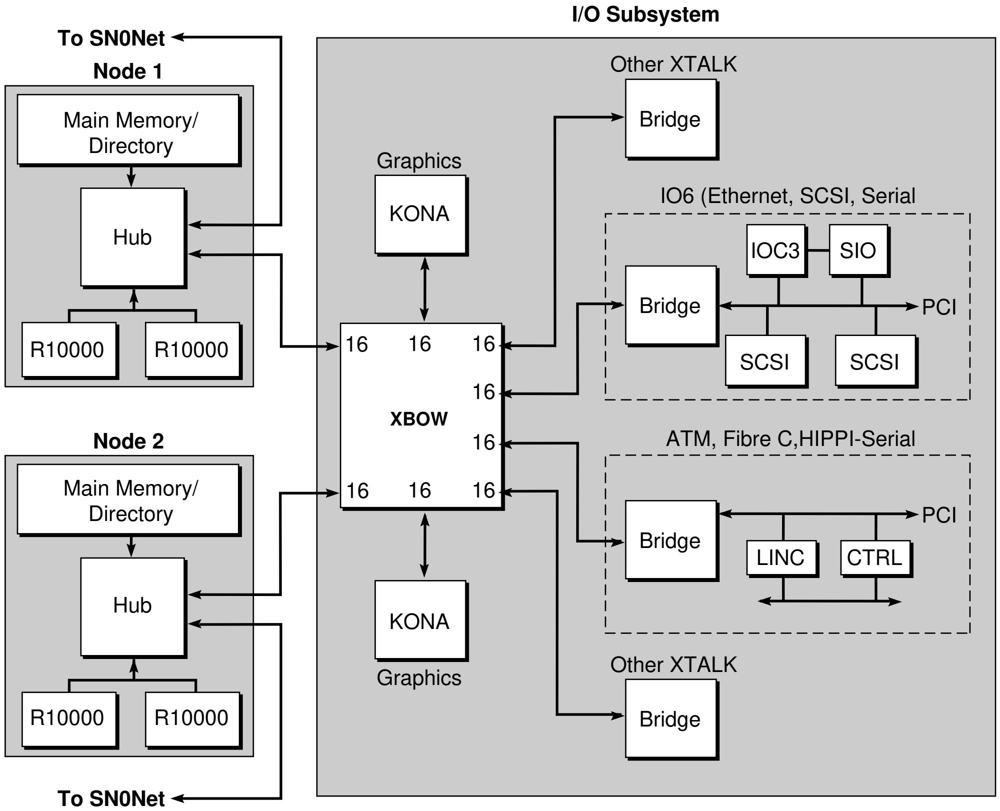

Fig. 5. Example IO subsystem block diagram

- four virtual channels per physical channel
- CRC checking on each packet with retransmission on error via a go-back-n sliding window protocol

The Origin 2000 employs SPIDER routers to create a bristled fat hypercube interconnect topology. The network topology is bristled in that two nodes are connected to a single router instead of one. The fat hypercube comes into play for systems beyond 32 nodes (64 processors). For up to 32 nodes, the routers connect nodes in a bristled hypercube as shown in Figure 3. The SPIDER routers are labeled using R, the nodes are the block boxes connecting to the routers. In the 32 processor configuration, the otherwise unused SPIDER ports are shown as dotted lines being used for Express Links which connect the corners of the cube to reduce latency and increase bandwidth.

Beyond 64 processors, a hierarchical fat hypercube is employed. Figure 4 shows the topology of a 128 processors Origin system. The vertices of four 32 processor hypercubes are connected to eight meta-routers. To scale up to 1024 processors, each of the single meta-routers in the 128 processor system is replaced with thirty-two routers that form a 5-D hypercubes and connect thirty-two of the local hypercubes.

4.3 IO subsystem

The Origin I/O system is also crossbar based. Figure 5 shows one possible configuration of IO cards connected to two nodes. Using the same link technology as in the Craylink interconnect, each Hub link provides a peak of to 1.56 GB/sec of bandwidth to the six XIO cards connected to it As shown in Figure 5, at the heart of the IO subsystem is the Crossbow (Xbow) ASIC. The crossbow has many similarities with the SPIDER router. The primary difference in the Xbow being a simplification of buffering and arbitration protocols given the chip's more limited configurations. These simplifications reduce the cost and permit eight ports to be integrated on a single chip.
The highest performance XIO cards (graphics and high-speed networking) connect directly to the XIO, but most of the cards bridge XIO to an embedded PCI bus with multiple external interfaces (e.g., 4-port Ultra SCSI, 2-port Fibre-Channel, 4-port 100BaseTX and 4-port OC3 cards). The IO bandwidth together with integration provide IO performance which is added as a PCI-bus at a time versus individual PCI cards seen on other systems.

4.4 Product design

The Origin 2000 is a highly modular design. The basic building block for deskside and rack use is shown in Figure 6. This module has slots for 4 node, 2 router, 12 XIO boards, and up to 6 Ultra SCSI devices. This module can be used as a stand-alone deskside system, or two modules can be mounted in a rack to form a 16 processor system. For systems beyond 16 processors, multiple racks are connected together via Craylink cables.
The Origin 200 is a cost-reduced version of the Origin 2000. The basic building block for the Origin 200 is a two processor tower. This tower contains one or two processors, up to 2 GB of memory, 3 standard PCI slots, and up to 6 Ultra SCSI devices. In addition, a Craylink port is provided which allows two towers to be connected together to form a single shared-memory system containing up to four processors, as shown in Figure 7.

5 Origin Design Trade-offs

One of the major goals for the Origin system was to keep both absolute memory latency and the ratio of remote to local latency as low as possible to provide an easy migration path for existing SMP software. To keep the local memory latency low, the intra-node interconnect consists of single Hub chip that implements a full four-way crossbar between processors, local memory

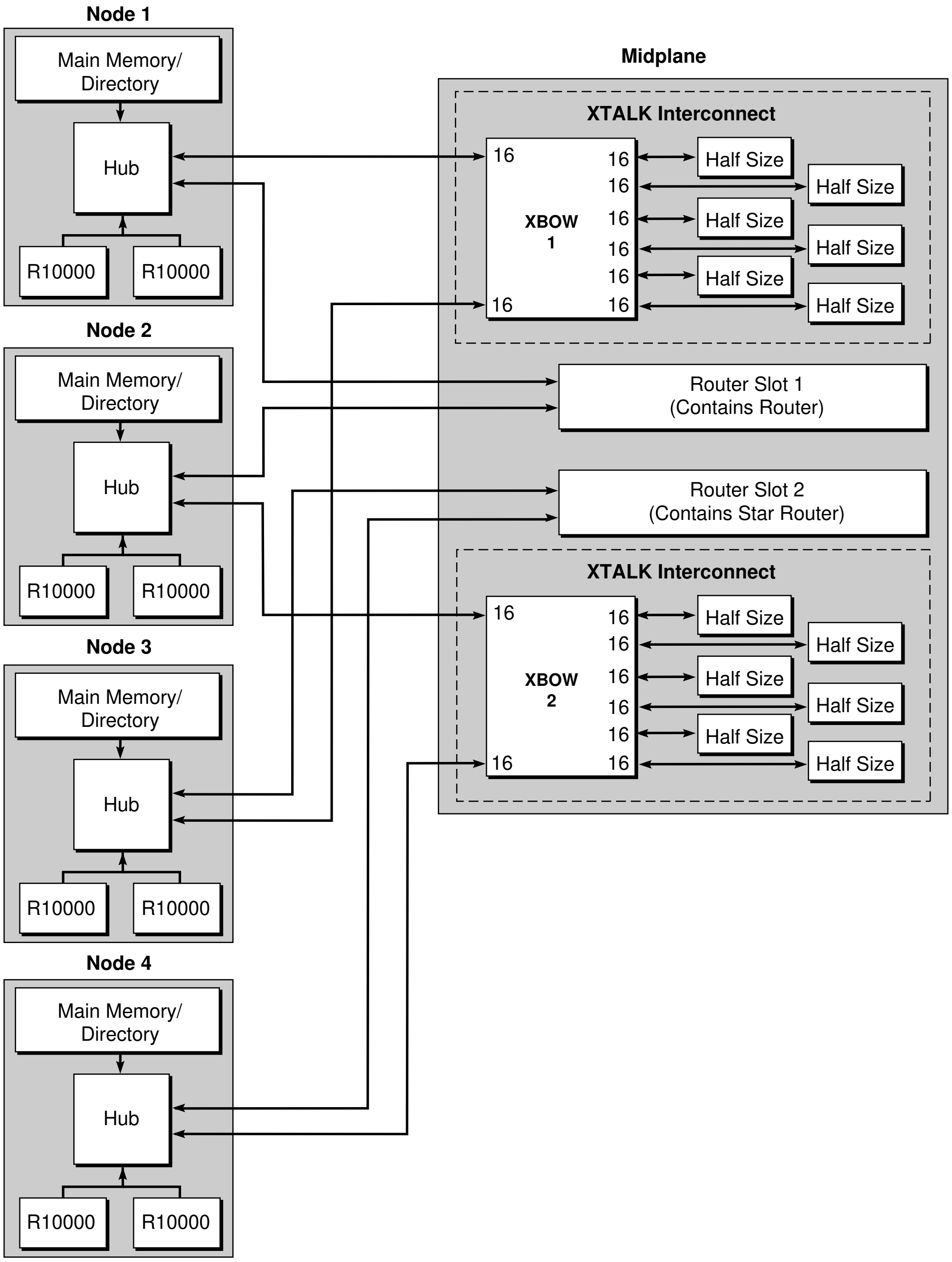

Fig. 6. Deskside module block diagram

and the I/O and network interfaces. To keep the remote memory latency low, the global interconnect is based on a six-ported router chip configured in a multi-level fat-hypercube topology. Together, these chips minimize both intra- and inter-node delays to both local and remote memory.

The single chip node design permits a cost-effective cross-bar interconnect within the hub chip. The crossbar is important since it allows local processors

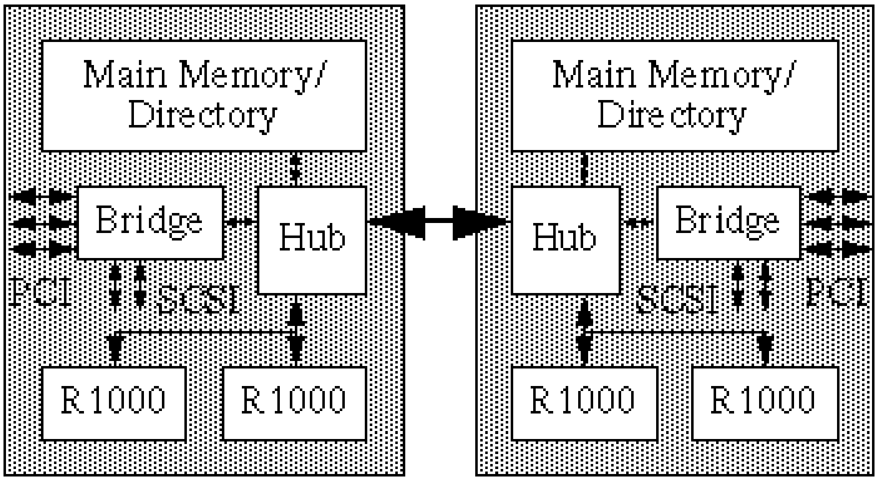

Fig. 7. Origin 200 block diagram

to access remote memory without interfering with remote processors accessing the local memory. An alternative scheme that has been implemented on a number of systems including DASH and the Sequent NUMA-Q, is to add to a DSM interface card to an existing small scale bus-based system. The problem with this type of design is that in order to access remote memory, the bus must be traversed three times. This adds considerable latency to these remote accesses. Furthermore, assuming all processors are accessing remote memory, then there are conflicts on the bus passing local data to remote processors, while also passing remote data to local processors. Since remote accesses require multiple bus transactions, remote bandwidth will be reduced by at least a factor of two, and typically closer to three. This reduction in bandwidth and increased latency can lead to a large disparity between local and remote cache miss penalties.

While minimizing latency is important, achieving higher performance on a DSM system versus an SMP system relies on having a good fraction of references satisfied by local memory. Locality can be greatly aided by the operating system when it can allocate memory for processes on the same node as the processors they are running on. For single-threaded jobs this is fairly easy; for parallel jobs, it is not clear which processor will reference a given memory location the most.

Many DSM systems (including the Convex Exemplar systems and Sequent NUMA-Q) implement a third-level node or cluster cache to help improve locality automatically in hardware. Such a cache can reduce the number of capacity misses that must be satisfied by remote memory, however, they do not help communication misses, and are subject to conflict misses themselves. In these situations, the cluster cache has a negative effect on both remote bandwidth and latency.

The problem with the cluster cache is that the cluster cache must be large to be effective, and thus is made of DRAM. Using a cluster cache implies that misses in this cache result in three DRAM accesses: one to determine that the block is not in the local cluster cache; a second at the home to fetch the data from real memory; and a third to reallocate the data into the local cluster cache. Likewise, latency is obviously going to be impacted by these DRAM accesses.
In contrast, S^2MP does not use a cluster cache and instead relies on page migration to improve locality. Page migration is assisted by hardware which keeps 64 reference counts on each 4KByte page of memory. On every access to memory, the count of the accessing node is incremented and compared with the home node. If the count is higher than a given programmable threshold, the hardware interrupts one of the local processors. This counting function does not affect the bandwidth of the data memory, it's implemented in the directory memory. Within the hub, there the block transfer engine can copy the page at near the memory bandwidth limit. The BTE allows migration without polluting the cache of either the local or remote processor.
Similar to a cluster cache, the migration scheme used in Origin can help optimize locality without increasing latency or decreasing bandwidth to remote memory. The only downside of this approach is that it does not react as quickly as the cluster cache to changes in access patterns, but this problem is mitigated by the filtering of references by the processor caches.
Another important aspect of the design of a large, complex system, support of high reliability. Given the distributed nature of a system like Origin, it is natural to exploit the redundancy and modularity of the system to provide high availability. Origin employes all of the normal reliability techniques such as single-bit error correcting and double-bit error detecting on all SRAM and SDRAM memory arrays. Likewise, features such as using very high levels of integration and redundant cooling and power give an inherent boost to reliability. More unique features of Origin include support for very robust interconnection links. In particular, both the Craylink and XIO channels use a full 16 bit CRC over each 128 bit packet that is transferred. Hardware across each link supports a sliding window protocol, with receiver acknowledgments, and sender retransmit capabilities. This provides fault tolerance for transient errors on the high-speed links. Furthermore, all I/O is dual hosted and routing on the Craylink interconnect is programmable and can use alternate routes in the fat hypercube topology. This allows for reconsideration of the network to avoid hard link errors, or downed modules or nodes within the system.
Another issue inherent in reliability of shared memory systems is the potential for faulty hardware, or errant software, to corrupt the system in such a way that the redundant portions of the system cannot continue because they cannot be sure that the faulty pieces have or have not corrupted shared data. In order to provide protection from this, S^2MP provides access control bits on all pages of memory and for all hardware accessible registers and I/O devices. This access control is in the form of a 64 bit access mask. For memory, these

values are implemented in the directory with the page counters for memory. They provide both read and read-write access to a particular page. The 64 bit access control allows fine control for systems with less than 128 processors and coarse control, groups of 8 nodes, for systems up to 1024 processors.
These firewalls provide the basis for partitioned systems using multiple kernels with limited memory sharing for applications. Or, they provide a more scalable, single system image in which the kernel uses shared- memory, but is structured in such a way to permit recovery of the shared data structures that may become corrupted upon a failure[8][9].

6 Future Trends

When comparing the different multiprocessor architectures, it is instructive to examine the technology trends, and how these trends will make the options either more or less attractive. Three important technology trends are as follows:

1. CPU processing rates will continue to rise rapidly. This in turn will require memory subsystems to offer lower latency and higher bandwidths.
2. Integration levels will allow more and more function to be implemented on-chip. Together with the first trend, this will drive microprocessor designs to incorporate direct main memory control and to eventually integrate memory on the CPU chip [7].
3. Cost-economics are driven by volume. Small systems are high-volume and offer cost leadership. As computer usage becomes more and more popular, the cost-performance delta between high-volume systems and low-volume, specialized systems will grow.

Table 1: shows how the different multiprocessor architectures stand relative to these trends. Here we see that traditional SMP systems suffer since they always put memory on the far side of the bus (away from the processor) and their centralized memory can not be integrated with the individual processors. Likewise, PVPs use a centralized memory, but are not as sensitive to this on code that vectorizes. However, the custom vector processors are not sold in high-volume and do not realize the same cost-performance as standard microprocessors. While MPP systems can take advantage of highly integrated memory systems, they suffer from low-volume since they are specialized for very large parallel application use. As stated before, cluster systems solve this problem by building their nodes from standard, high-volume workstation or PC nodes. SSMP systems also mesh well with the technology trends since they're distributed memory allows for tight integration with the CPU. Likewise, SSMP systems scale down to small systems sold in high-volume.

Table 1: MP architectures against trends.

	Improved BW and Latency	CPU Integration	Small, High-Volume Systems
Traditional SMP	×	×	√
Vector (PVP)	n/a	×	×
MPP	√	√	×
Cluster	√	√	√
SSMP	√	√	√

The conclusion being that only the structure of clusters and SSMPs are favored by the technology trends. The strengths of these two systems are compared below:

Cluster Systems:

+ Good fault isolation due to ”shared-nothing” structure.
+ Highest-volume and potentially lowest cost since clusters can be based on commodity non-scalable node components.
+ Support for heterogeneous nodes.
+ More controlled communication allows for less capable and thus less expensive interconnect network.

Scalable Shared-Memory Multiprocessor:

+ Tighter coupling (lower latency and higher bandwidth) permit higher performance parallel processing.
+ Simplified programming paradigm.
+ Exploit high-performance, centralized resources (such as large main memory and high-performance storage and networking).
+ Simplified administration from single-system image.

Since SSMP systems are distributed they can also be built out of modular components that provide similar fault isolation as clusters. Overall, we anticipate that future systems will consist of clusters of SSMP systems. SSMP system size will grow to be at least as large as the largest parallel job. They may also be somewhat larger than this in order to exploit resource sharing and to reduce the total number of systems in a cluster. Likewise, future systems will be clustered down to the boundaries where communication is limited, physical distance prohibits low latency, or there are heterogeneous systems.

The boundary between SSMP and clusters will vary greatly based on application mix. For systems dominated by a throughput mix with uniform resource requirements, the SSMP nodes might contain only 2-4 processors. For systems used for the largest physical modeling problems with highly parallel workloads, SSMP systems would scale into the thousands of processor range.

7 Conclusion

There are four common multiprocessor architectures today: message-passing machines, clusters, parallel vector processors, and shared-memory multiprocessors.

Academic and now commercial work on distributed shared memory machines has shown that the SMP programming model can be made to scale to large processor counts with high performance. The two enabling technologies for these SSMP systems are directory-based cache coherence and scalable interconnection networks. These technologies allow the SMP model to stretch to the design space previously only covered by massively parallel, message-passing, systems. The SMP programming model and modular structure of DSM architectures also allow them to scale down to small configurations that are sold in high-volume. This allows DSM systems to have the cost structure of clustered workstations. Finally, the scalable interconnect used in DSM systems enables their memory performance to approach that of parallel vector processors.

The goal of SGI's Origin systems is to deliver a single system that can cover all of the SMP, MPP, and cluster design space, as well as the low end of the PVP space. Achieving this goal has required a set of careful trade-offs that maintain both high performance and the look and feel of an SMP system to the user.

The most important factors in achieving a truly scalable SMP look and feel are low latency and high bandwidth to local and remote memory. Some of the techniques used by Origin include: a 64-bit superscalar microprocessor with a highly parallel memory interface; a single-chip node controller with an internal crossbar and tight coupling to local memory and the global interconnect; a high dimensional fat-hypercube interconnect topology; and an optimized coherence protocol. The result is that remote memory latency is within a factor of two to three of local, and that remote memory bandwidth is comparable to local bandwidth.

At SGI, we believe that only two of the parallel architectures will be suitable in the future. For high-performance parallel computing, SSMP systems will dominate due to its simple programming paradigm and tight multiprocessor coupling. For more distributed workloads, clusters of uniprocessor systems and SSMPs are most appropriate.

At SGI, we are pursuing both clustered and SSMP architectures. Today, we offer clustered systems connected by multiple 100 MByte/sec HIPPI interconnects. In the future, HIPPI-6400[2] will be used to boost cluster performance to over 800MB/sec per link. SGI's first SSMP system is the Origin line of servers. In the future, Origin's S^2MP architecture will be used to build a line of SGI/Cray SSMP machines that span from the desktop to teraflop supercomputers.

8 References

[1] Anant Agarwal, Ricardo Bianchini, David Chaiken, Kirk L. Johnson, David Kranz, John Kubiatowicz, Beng-Hong Lim, Kenneth Mackenzie, and Donald Yeung. The MIT Alewife machine: Architecture and Performance. In Proceedings of the 22nd Annual International Symposium on Computer Architecture, pages 2-13, June 1995.

[2] Greg Chesson. HIPPI-6400 Overview. In Hot Interconnects '96.

[3] L. Censier and P. Feautrier. A New Solution to Coherence Problems in Multicache Systems. IEEE Transactions on Computers C-27(12):1112-1118, Dec. 1978.

[4] Mike Galles. Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER chip. In Hot Interconnects '96.

[5] IEEE. ANSI/IEEE Standard 1596-1992, Scalable Coherent Interface (SCI), 1992.

[6] Daniel Lenoski, James Laudon, Kourosh Gharachorloo, Anoop Gupta, and John Hennessy. The directory-based cache coherence protocol for the DASH multiprocessor. In Proceedings of the 17th Annual International Symposium on Computer Architecture, pages 148-159, May 1990.

[7] D. Patterson et. al. A Case for Intelligent RAM. IEEE Micro, 17(2):34-44, Mar 1997.

[8] Mendel Rosenblum, John Chapin, Dan Teodosiu, Scott Devine, Tirthankar Lahiri, and Anoop Gupta, Implementing efficient fault containment for multiprocessors. Communications of the ACM, 39(3):52-61, September, 1996.

[9] Steve Whitney, Ken Jacobsen, et. al., The SGI Origin Software Environment and Application Performance. In Proceedings of IEEE COMPCON '97.