PARALLEL COMPUTING IN THE COMMERCIAL MARKETPLACE
Research and innovation at work

Erik Hagersten and Greg Papadopoulos
Sun Microsystems, Inc.

Abstract

This is a good time for parallel-computer development and research in both academia and industry. The performance improvements predicted by Moore's Law has proven itself quite accurate over many years. However, the doubling of processor performance every 18 months cannot keep up with the growing demand of many applications. The performance of database applications has been doubling every 9-10 months. At last, parallel-computer technology has come to play an important role in the commercial marketplace.

Multiprocessing has been an active research area for almost 40 years and commercial parallel computers have been available for more than 35 years. After getting off to a slow start, this area has now taken off . Shared-memory multiprocessors have dominated this development. This is an area that has sprung out of tireless research and numerous published breakthrough results.

This article analyzes some of the reasons for the sudden acceptance of the relatively old parallel computing field, outlines the key properties of a successful parallel computer of the '90s and identifies some important research areas and key technologies for the future.

1. Introduction

Parallel-computer architecture is a classic research topic and has attracted good researchers and produced ground-breaking results the last 40 years. This technology has been applied to the commercial world over time with mixed results. Only in the last five years has this turned from being "interesting and promising" to being one of the key technologies of the world's largest computer companies. Commercial parallel computers have evolved and matured in different distinctive waves:

1. **The pioneer parallel computers (1960-2000):** The first parallel computers were built from a handful of mainframe-style CPUs, connected through a switch to one common shared memory. This fairly expensive and non-scalable style of building top-of-the-line computers culminated with the CRAY-XMP architecture. This is still a viable technology provided by some mainframe vendors, but, its importance is diminishing.
2. **The massively parallel processors (MPPs) (1980-1995)**: The cold war and its quest for computepower fueled a focused effort in academia and primarily start-up companies to build the ultimate scalable machine, dominated by massively parallel processors, MPPs. Many different experimental approaches were tested during this time, yet, the dominating architecture was that of the message-passing multicomputers (MMCs). The MPPs were much harder to use than anticipated. You had to *really* want to solve a problem with them! A lack of market volume and the end of the cold war terminated this wave abruptly. During the same time period, the

introduction of the workstation's distributed computing power also decreased the need for centralized servers.

3. **The general-purpose parallel computers (1993 - Present):** Like the disappearance of the specialized processors (such as the LISP processors) in favor of the more general purpose RISC/CISC microprocessors of the 80s, this third wave is dominated by general-purpose parallel computers rather than the more specialized MPPs. The parallel computers of this wave are often compatible with already existing operating systems and are cost-effective both at the high-end of the market as well as the mid-range, overcoming the MPPs' deficiency of scaling up, but not scaling down, which allows this third wave to appeal to a much wider and growing market.

2. General-purpose Parallel Computers

So why did parallel computers wait until the mid 90s before taking off? We can identify a number of important factors which all coincided and contributed to the parallel computer revolution:

- A return to centralized computing driven by its ease of management
- Cheaper disks allowing for more on-line data
- Growing database sizes and a need to analyze the data
- Parallelizable databases from independent software vendors (ISVs)
- The explosion of Internet, Intranet, Extranet and the World Wide Web
- Increasing speed of cheap microprocessors
- The arrival of stable and scalable versions of the UNIX operating system

However, the primary reason for the boom is probably the introduction of various forms of shared-memory multiprocessors, here collectively referred to as SMMs. At this time, several major computer vendors started offering shared-memory architectures running fairly stable versions of UNIX. Suddenly, this became a technology that could be relied on beyond high-performance HPC applications [Brewer97, Charlesworth98, Laudon97, Lovett96, Singhal96, Clark96, Protic98].

These machines are applicable to a wider range of applications than the MMCs of the previous wave and compatible with many widely used UNIX desktop applications, some of them already parallelized. Overnight, these parallel computers seamlessly became centralized computing resources serving many workstation users. These architectures were also much more focused on medium-sized scalability and offered more appropriate solutions for the sweet spot of the market than the world-record-lusting MMCs of the previous wave. This provided for a much larger customer base and made the parallel computers less dependent on one single application area or a single source of financing. In 1997, the UNIX server market was estimated to be almost $24 billion and growing at a yearly rate of 22% [IDC98]. This can be compared to the technical supercomputer market the MMCs tried to feed on in 1995 (machines costing more than $1 million) estimated at $750 million and an annual growth of 3.4% [Dataquest96,Stenstrom97].

3. Important Properties of Shared-Memory Architectures

The commercial database market dominates the server market today. This is why database applications are the focus of many parallel-computer architecture design decisions. These applications are a good fit for shared-memory multiprocessors (SMMs). Contrary to many shared-memory proponents, however, we do not believe their loudly claimed "simpler programming model" compared with message-passing multicomputers (MCCs) is the primary reason. An even more important property is the simple resource-sharing model provided by SMMs. The SMM, everything-is-shared, view of the world provides a simple performance and resource-sharing model for processors, memory and I/O. While such properties may matter less for some regular and easily partitionable technical HPC applications, flexible resource sharing is crucial to many of the less predictable commercial applications with widely shifting demands for resources. This is what lends the SMMs to a wider market and, thus, a general purpose solution.

An SMM does not require data and code to be placed in any special node, nor does it need any special configure-to-fit hardware organization for an application to run well. Popular code and data structures are easily shared by all the CPUs. A suspended process may be rescheduled on any processor.[1] Managing the memory is also easier: any free physical memory can be utilized when any processor needs more memory allocated. Imagine a hypothetical case of two parallel tasks, one needing 1 GB of memory and one CPU, and another task requiring 100 MB of memory and four CPUs. In order to run the two tasks in parallel, an MMC built from five identical one-CPU nodes will need 1 GB memory each node; i.e., a total of 5 GB, of which 3.9 GB will not be utilized, in order to run well. On the other hand, an SMM with only 1.1 GB of total memory and at least five CPUs will run these two tasks well in parallel. As long as the total number of resources in a shared-memory machine matches the total requirements of the two tasks, that architecture will fit. The resource scheduling in an SMM can be resolved dynamically at runtime while the MMC requires static scheduling of resources. Some of the static sceduling has to be done before the MCC hardaware is even assembled while other scheluding can do be done before, or at, compiletime.

Another important property of the SMM is the creation of a seamless family of scalable architectures. The workstation with two CPUs, the work-group server with four CPUs and the supercomputer with 64 CPUs can all have the same programming model and be binary-compatible with each other.

However, the introduction of nonuniformity in some shared-memory implementations makes all these SMM features less attractive and makes the SMM behave more and more like an MMC. More about that later.

[1] Running it on the CPU where it last ran has a small advantage since its hot cache can be leveraged, i.e., affinity scheduling.

Yet another important, but rarely discussed, issue is the complexity involved in the management of the large-scale systems. Adding and removing resources from an SMM does not change its being a single system running one operating system--the SMM comes up and down and gets upgraded like a single system. So, adding resources to an SMM does not add complexity at the same rate. The same cannot be said of the MMCs or the network of workstations, where an increased number of nodes increases the number of systems to manage and adds substantially to management complexity.

One last important property of the SMMs is their excellent cost/performance characteristics. There is nothing intrinsically more costly about a well-designed SMM compared to a network of workstations (NOW) or an MMC as long as the volume of the product families is the same. Note that the bulk of the cost of a parallel computer rests in its disks, memory, processors and cache modules. Everything else being equal, i.e., with the same programming and managing effort, the communication requirement of both SMM and MMC is roughly the same. A well-designed NOW/MMC would, therefore, roughly have the same cost as an SMM. Many papers on this subject compare the prices of different systems with the same number of processors, but, with interconnects of widely different latencies and bandwidths. Note that price involves many more system properties than just number of processors. What about value-added features, such as a better use of resources, achieved performance, cost of ownership, and simply customer demand? Once more, the task is not to find any single suitable application in order to prove a point, but, rather to build an architecture appealing to a wide market [tpc98].

4. Increasing the Availability

The model of resource sharing in SMM today has been taken to yet another level by the introduction of dynamically configurable domains [Charlesworth98, Teodosiu97]. An SMM can be divided into several domains, each configured to include a set of resources, such as CPUs, I/O and memory. Each domain runs its own operating system and has its own isolated error-handling system. A domain can be dynamically reconfigured to include more or less resources by moving the domain boundaries while the applications are running. This effectively turns a *physical* SMM into a heap of resources consisting of CPUs, memory and I/O, which can be configured into a number of error-isolated *logical* SMMs, each dynamically customized to fit the customers' current resource needs. Resources can be moved from one domain to another while the operating systems and the applications in both domains keep running.

Dynamic domains provide a degree of coarse-grained resource management while overcoming the largest Achilles' heel of an SMM: the vulnerability of the single operating-system kernel. A single hardware or software error may bring down the entire machine. The domains provide error isolation, and an error will only bring down the domain in which the error occurred.

The term "cluster" is often used to describe an MMC built from many independent nodes, each running its own operating system, in contrast to the MPPs, where the same operating system controlled all the nodes. Further availability can be achieved by running a cluster-based high-availability database software, connecting several large SMM machines[2]. The clustering software implements a special high-availability message-passing protocol between the SMM nodes; it allows the database to stay up even if one of the SMM nodes crashes. This combines the best of two worlds: the scalability and resource sharing provided by the large SMMs with the high availability of clustering software.

So, what happens when the capacity of an SMM is exceeded, or high availability is needed and a cluster is required? Doesn't this negate the advantages of an SMM? Wouldn't it be better to jump to a pure MMC with a more regular architecture or to a huge number of thin PC nodes? The answer to this question differs depending on whom you ask. For companies lacking the technology needed to build large SMMs and capable only of providing thin nodes, the answer is "Yes", of course. However, if you ask a company capable of providing both small and large SMMs with clustering capabilities, the answer will be an emphatic "No". The partitioning problem grows more difficult with the number of partitions to manage; so, in every case, we would claim that for a given amount of resources, *fewer bigger* is better than *many small*.

5. Fewer Bigger or Many Small

At the most basic level, an SMM is about pooling resources, while an MMC and cluster is about statically dividing them. The driving principle of an SMM design is to provide an easy and efficient way of sharing processing, memory and input/output resources. That is, all of the resources in an SMM are almost equivalent or symmetric. This symmetry is a great simplifying assumption for both the application developer and the system/application administrators. It decouples the exploitation of parallelism from the placement of data.

Conversely, an MMC introduces significant nonuniformity. Good performance is achieved only when most of the data a processor needs is found in its local memory or on its local disk. Otherwise, the data has to be communicated from one processor to another via the MMC's communication network, which is at least an order of magnitude less efficient than communications between a processor and its local storage, even for the very best interconnection network. The bottom line here is that the applications developer and administrator must be acutely aware of the need to align threads of computation with the placement of data involved in the computation. This is hard because it adds a big constraint. Not only must one decompose the control into separate computation streams or threads (true for SMM and MMC), but must also decompose the data to line up with the control (which is automatic in SMMs).

Looking specifically at how decision-support database software views the difference between MMC and SMM, MMC systems are much more sensitive to how a given

[2] Or connecting several logical SSMs in the same physical SSM machine

query maps onto the data partitioning selected at the time the database was built. Thus, performance becomes much more variable and far less predictable. Before you divide the database data among the nodes, you must know what query will be asked and how it will get parallelized. Even if you can master this complex task, it will not help you much when you put to it a completely different query, unless you redistribute your data in the meantime [Carlile96]. The nonuniformity is a bug and not a feature.

It is far better to cluster fewer, bigger nodes than more, smaller ones! This is true not just for performance, but for system administration as well, as the next examples illustrates.

6. System Examples

Imagine that the requirements for a large data warehouse dictate a system with 1000 disks, 100 processors and 20 gigabytes of memory. You are given three specifications to choose from:

- *MMC*: 100 single-processor nodes, each with 10 disks and 200 MB of memory.
- *Cluster of SMMs:* Four 25-processor SMM nodes, each with 250 disks and 5 GB of memory.
- *SMM:* A single 100-processor SMM machine with 1000 disks and 20 GB of memory.

Which do you prefer? By the "fewer, bigger" rule, you would prefer the SMM. However, higher availability and/or scalability requirements may also prompt you to chose the cluster of SMMs.

6.1. Adding more memory or disk per node

Suppose you needed to expand memory, add more disk storage or upgrade the operating system or some software. You may go through the installation procedure 100 times for the MMC or only 4 times for the cluster of SMMs; however, SMM operation is done just once.[3]

6.2. Effects of data skew for a decision-support database

A small amount of data skew, say 1%, can easily be caused by an uneven layout of a database with respect to a specific database query. That is, imagine that for a given query, 99% of the required data are evenly distributed across the nodes (partitions), but that the remaining 1% are skewed onto a single node (partition). In practice, this degree of skew is encountered frequently in decision-support databases. On the MMC, the 1% skew causes a stunning 50% loss in performance: all nodes make their way through their 1/100 of the 99% of the data in parallel (i.e., each node performs 0.99% of the total work in parallel.) Now, the remaining 1% of the task rests on a single node and is plowed through serially while the other 99 nodes idly wait for it. This makes the utilization of the nodes about 50% (i.e., a slowdown close to 50% is

[3] Modern SMMs even allow you to hot-plug the new resources while the system runs.

encountered compared to a fully-utilized system.) This is example is a direct consequence of Almdahl's well-known law of serialization when applied to skewed data.

On the cluster of SMMs, the slowdown is close to 4%. Three of the four nodes do little less than 25% of the work, while the one carrying the skew is asked to handle about 26%. Here, all four nodes are busy most of the time working on their 1/4 of the 99% of the data. The remaining 1% of the data can be operated on by the 25 processors in the node carrying the skew -- which is why the waiting time for the 75 processors in the three idle nodes is substantially less than for the MMC. This is a great example of the resource pooling that big SMM nodes provides. All of the assets of the node can be applied to handling the skewed data. The SMM experiences no slowdown. Since it is a single system, no data skew is architecturally visible [Carlile96].

6.3. Growing the Database

Finally, we'll look at a rather common procedure: growing the database. For all systems, we have the option of adding more disks per node or adding more nodes. Adding more nodes is always an issue because it usually requires that the database be repartitioned to span the new nodes, a rather involved procedure that may make the system unavailable while the action is performed. The benefit is that the processing performance is scaled as the disks are added, so the response time of the system should remain more or less constant.

Adding more disks per node typically doesn't require repartitioning and in most systems can be performed while the database is operational. Unfortunately, we have just added more data to the system without scaling up the processing. This means that response times for queries that touch all the data will get longer in proportion to the amount of data added. This is always true for the MMC because the processing per node is fixed. On the cluster of SMM, however, we have the option of adding more processors and memory to each node, up to the SMM's capacity. Thus, administrators do not have to compromise performance for ease of growing the database. Of course, eventually, the capacity of each SMM will be exhausted and the number of nodes will need to increase. At least this event is less frequent and can be part of a long-term planning cycle.

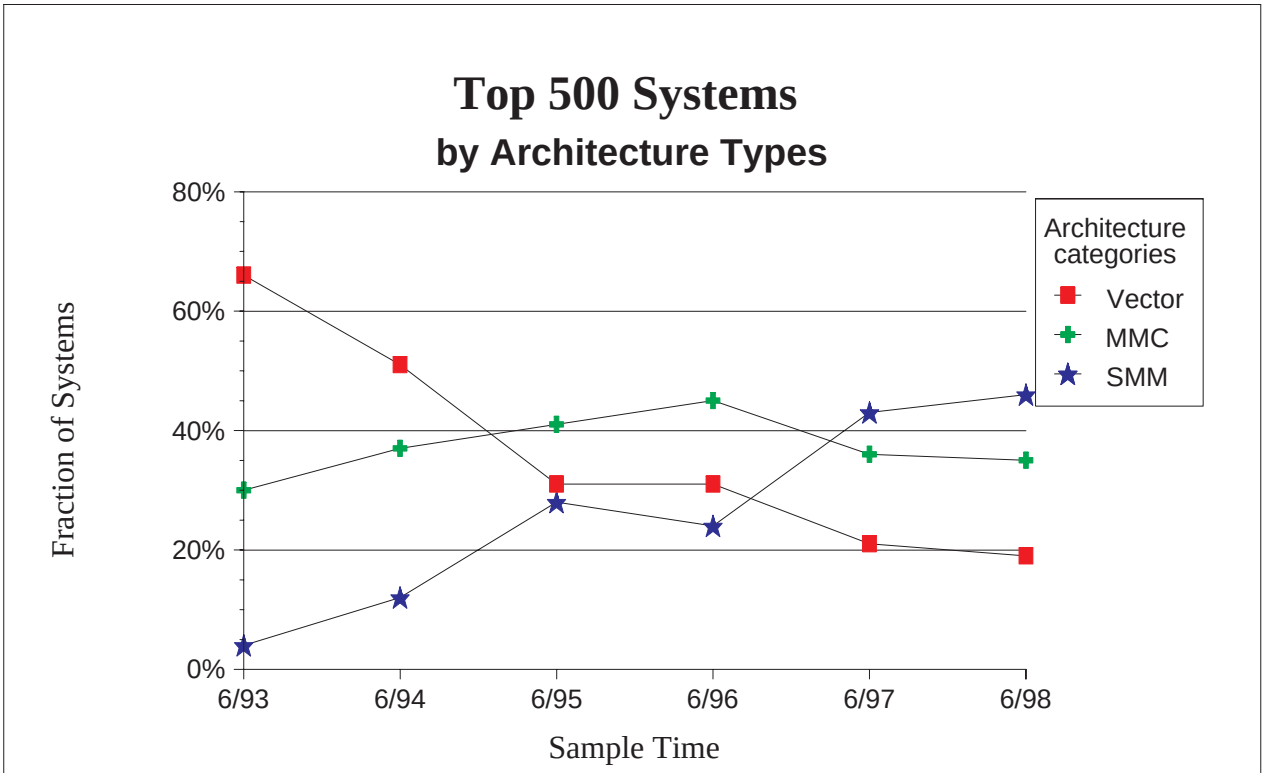
7. Technical HPC Computing

So far, we have discussed how commercial computing has impacted the current wave of parallel computers. Contrary to commercial applications, technical HPC applications are typically programmed using some "standard" message-passing library such as MPI or PVM. Does not this make the MMCs the obvious choice in this market? Not necessarily. We have to clearly differentiate between a programming model and the architectural implementation. There is nothing to preclude one from implementing a message-passing library on top of shared memory. The efficiency of such an implementation, of course, depends on the efficiency of the underlying SMM. It is our experience that a tightly-coupled low-latency SMM with support for cache bypass lends itself to a lower latency MPI implementation than a traditional MMC.

High-end computing is currently shifting away from tailor-made, one-of-a-kind expensive solutions. In 1995, the HPC technical market for machines costing more than $1 million represented only 3% of the total server market [Dataquest96, Stenstrom97]. This is clearly not a market large enough to drive "architectural changes," and this is why so many high-end companies went out of business. Also the HPC market is moving away from tailor-made exclusive computers. The boutique-shopping era is over.

Today, the HPC market is often addressed by systems built from building blocks that also suits the much larger commercial market. This can be exemplified by studying the processors used in the world's 500 fastest computers[top500]. As late as 1993, 88% of the computers listed on the top 500 list were based on proprietary CPU architectures, and only 12% were based on commercially available RISC/CISC microprocessors. Today, about 80% of the top 500 entries are based on commercial microprocessors. The move toward commercial building blocks can also been seen by the architecture types represented in the top 500 list, as shown in Figure 1. In 1993, only 4% of the systems were SMMs. In 1998, that number is 46% and rising.

Fig 1: *The architectural mix among the world's 500 fastest computers keeps changing [top500]. The vector systems dominated the early '90s, MMC (including clusters) were most popular in the mid-'90s and the SMM is currently the most common architecture.*



However, this SMM dominance can not yet be seen among the top 100 systems, where only 10% of the systems are SMMs [top500]. We predict SMMs will become important as buildingblocks also for the top 100 systems. We have already seen how

MMCs of large SMMs can add extra availability and scalability beyond the SMMs in the commercial market. So, the large commercial market has also turned such combined solutions into commodity technology. High-end, cost-effective technical HPC computers assembled from large shared-memory nodes connected together using cluster technology have already entered the HPC arena. The latest supercomputers in the Department of Energy's ASCI program have moved away from the traditional MMC built from skinny nodes to clusters of large SMMs. The ASCI Blue Mountain from SGI, consisting of 48 clustered SMMs with 128 CPUs in each node, has dethroned the ASCI Red build from Intel's quad P6 nodes and is now now the world's fastest computer with a record 1.6 GFLOPS Linpack number. The "bigger is better" rule, as discussed above, still applies. The often regular HPC applications can also explore efficient "nearest neighbor" message-passing communication implemented on top of the shared memory among all the processors residing in the same SMM node. Using commercial building blocks not only drives the cost/performance down, but also has a much higher level of reliability, availability and serviceability (RAS) than the one-of-a-kind systems. We believe that sophisticated RAS features will increase the availability of large HPC systems.

8. SMM Implementation Issues

Today, most SMMs sold are so-called Symmetric Multiprocessors (SMP). This term is not very well defined. Most often, SMP is used as a synonym for Uniform Memory Architectures, or UMA, where the access time from any processor to any part of the shared memory is the same. Our definition of SMP is an architecture which is so uniform that the placement of code and data does not really matter. This allows the shared memory to be viewed as the shared resource, as exemplified above. For this to be true, we believe the access time to different parts of the entire memory, as well as to dirty data in other caches, should differ less than a factor of 2. This rather tough requirement constrains SMP implementations to physically reside in one enclosure. The physical limitations of elevator shafts and airplane cargo holds put an upper limit on the size of an enclosure and also an upper limit on SMP implementations. The number of CPUs per SMP (i.e., per enclosure) has been increasing steadily, depending on the implementation technology, and is now at 64 CPUs [Charlesworth98].

Distributed shared memory, implemented across several enclosures, can allow SMM implementations beyond the scalability limit of an enclosure, pushing the "bigger is better" rule even further with the introduction of the largest SMM node. Currently, SGI supports up to 128 CPU SMMs [Laudon97], and Sequent can build SMMs up to 252 CPUs [Lovett96]. Both these architectures are SMM implementations with nonuniform memory accesses (NUMA) very similar to earlier research machines, such as the Stanford Dash [Lenoski90]. The ratio between local memory and remote memory for these architectures is a factor between 3 and 20, depending on the size of the system. Here, the placement of data and processes matters more than in the general SMM case, yet less than the MMC issues outlined above.

However, architectural scalability is only one part of the problem (probably the easy part). Such an architecture must be accompanied by a scalable operating system and a scalable SMM application. Scaling an operating system while maintaining its stability is a very complicated task. It has taken eight years of focused development to move the scalability of Sun's operating system from a 4-way system to one that reliably can handle 64-way commercial servers. Today, the most scalable SMM TPC benchmarks published use 64 CPUs from Sun and 32 CPUs from DG, SGI and Sequent [tpc98]. Our experience shows the SMMs to double their number of CPUs every 2-3 years. Sun's next generation SMM will more than double the number of CPUs. Figure1 shows how SMM systems have also gained more acceptance in the high-performance world. The market share for SMM has increased over the last year while the market share for MMCs and vector processors has decreased. We believe this is a trend which will continue. We predict that each vendor will cluster the smallest possible number of the largest possible SMMs to meet the demands of this market.

9. Future Research Topics for Shared Memory Architectures

As SMM grows in importance in the commercial world, the interest in SMM research will remain high. Keep the following guidelines in mind:

- Target research toward general-purpose multiprocessors. A single embarrassingly parallel application does not prove a point.
- Scalability is just one of many important properties of an SMM system. Remember that most parallel processors are in the 4-32 processor range.
- NUMAs with too much nonuniformity degrade to MMC-like performance with properties similar to the MMCs discussed above. Scalable solutions that minimize the application's exposure to nonuniformity are therefore important. Ideally, a nonmodified SMP application should perform well on any SMM, regardless of size.
- Scalable operating system and algorithms are important. This is the current bottleneck in efficient SMM implementations. Leaving the effect of the operating system out of the research picture will limit its value.
- Efficient cluster interconnects supporting large SMMs are not a very well explored topic. SMMs, no matter how large, can be clustered together for scalability and/or high availability reasons.

10. Conclusion

Parallel Computers are here in large numbers, increasing every day. Parallel Computers are an undisputed success. Shared-Memory Multiprocessors (SMMs) are a key component of most parallel computers, either as the parallel computer's sole architecture or as the architecture for its cluster nodes. We expect large SMMs to meet most requirements, and they are by far the best choice when they do. Combining Moore's Law of double processor performance every 18 months with the SMM's doubling number of CPUs every two to three years puts the SMMs on a performance doubling rate of 10-12 months.

Clustering technology will become the dominant high-availability technology rather than a scalability technology. Clustering a handful of large SMM nodes provides sufficient availability in combination with the best scalability, lowest administration complexity and the best resource management.

20 years ago it was believed that efficient SMMs could not be built. Today, most manufacturers of microprocessors, operating systems and computer systems expend substantial effort to push the scalability of their SMM systems. Companies previously offering 4-way SMMs are pushing for 8-way systems today, companies offering 16-way systems are now pushing for 32-way systems, and so on. Sun will double, at least, the number of CPUs in its next generation system from today's 64 to more than 128. We are speaking in unison: fewer bigger, instead of many small. Thanks to all the good SMM research.

References

[ASCI] The Accelerated Strategic Computing Initiative at Los Almos, Lawrence Livermore and Sandia laboratories, sponsored by the Department of Enery http://w4.lanl.gov/Internal/projects/asci/platforms.html

[Brewer97] T. Brewer and G. Astfalk. *The evolution of the HP/Convex Exemplar.* In Proceedings of *COMPCON* Spring 1997.

[Carlile96] Brad Carlile. *Seeking the Balance: Large SMP Warehouses. Database Programming & Design*, August 1996.

[Clark96] R. Clark and K. Alnes. *SCI Interconnect Chipset and Adapter: Building Large Scale Enterprise Servers with Pentium Pro SHV Nodes.* In *Proc. IEEE Hot Interconnects*, Stanford CA, P 41-52, Aug 1996

[Charlesworth98] Alan Charlesworth. *STARFIRE: Extending the SMP Envelope. IEEE Micro* Jan/Feb 1998. (`http://www.sun.com/servers/enterprise/10000/wp/`)

[Dataquest96] *Dataquest Market Data 1996.*

[Teodosiu97] Dan Teodosiu, Joel Baxter, Kinshuk Govil, John Chapin, Mendel Rosenblum, and Mark Horowitz. *Hardware Fault Containment in Scalable Shared-Memory Multiprocessors.* In Proc *24th Annual International Symposium on Computer Architecture*, 1997

[IDC98] *IDC Market Data 1998.*

[Laudon97] James Laudon and Daniel Lenoski. *The SGI Origin: A cc-NUMA Highly Scalable Server*. ACM/IEEE *International Symposium on Computer Architecture (ISCA)*, June 1997

[Lenoski90] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy. *The directory-based protocol for the DASH multiprocessor. International Symposium on Computer Architecture (ISCA)*, 1990

[Lovett96] Tom Lovett and Russel Clapp. *STiNG: A cc-NUMA computer system for the commercial marketplace.* ACM/IEEE *International Symposium on Computer Architecture (ISCA)*, June 1996

[Protic98] J. Protic, M. Tomasevic, V. Milutinovic. *Distributed Shared Memory: Concepts and Systems*, IEEE Computer Science Press, Los Alamitos, California, 1998.

[Singhal96] Ashok Singhal, David Broniarczyk, Fred Cerauskis, Jeff Price, Leo Yaun, Chris Cheng, Drew Doblar, Steve Fosth, Nalini Agarwal, Kenneth Harvery, Erik Hagersten, Bjørn Liencres. *A High Performance Bus of Large SMPs.* In *Proc. IEEE Hot Interconnects*, Stanford CA, P 41-52, Aug 1996.

[Stenstrom97] Per Stenström, Erik Hagersten, David Lilja, Margaret Martonosi and Madan Venugopal. *Shared-Memory Multiprocessing: Significant Issues and Research Needs*. IEEE Computer 1997.

[top500] *Top 500 list http://www.netlib.org/benchmark/top500.html*

[tpc98] *TPC-C and TPC-D Results.* http://www.tpc.org