Future General Purpose Supercomputer Architectures

J. E. Smith W.-C. Hsu C. Hsiung

Cray Research, Inc.
Chippewa Falls, WI 54729

Abstract

This paper discusses the challenges facing designers of future general purpose supercomputers, and describes architectural features and characteristics that may be used to meet these challenges. The scope of the discussion ranges from individual processor architecture to large-scale multiprocessor systems. Performance trends and projections for general purpose supercomputer systems are given as part of the discussion.

1. Introduction

"May you live in interesting times."
-- traditional Chinese curse

These are interesting times for many segments of the computer industry. High performance workstations, massively parallel systems, mainframes, mini-supers, and supercomputers are all jostling for position in the scientific and engineering marketplace. Within this wide range of computer systems, the general purpose supercomputer has a clear role. In particular, general purpose supercomputer systems have the following five characteristics.

(1) They are targeted toward large-scale scientific and engineering problems. These are often floating-point intensive, but are not necessarily so. In the future we expect non-floating point (and non-engineering) applications to become more important. Furthermore, the developing network and standard software environment requires high performance on system as well as application code.

(2) They offer the highest available performance, in a variety of respects, in the same system. Processor, memory, and I/O performance, capacities, and bandwidths are all the highest available, and they work cohesively together in the same very tightly coupled system.

(3) They provide time-critical solutions to big problems. Often their value is in reducing the time to a solution. Cost-performance is important, but often performance alone is more important.

(4) They are programmable; they can be used with common programming languages, often with no more effort than with any other type of general purpose computer. The programming environment is directed toward standard interfaces. They are multi-user systems. They offer intuitively simple, well-understood programming models (e.g. flat shared main memory).

(5) They are accessible; they are easily integrated into a network environment, and are able to communicate and function with the rest of the network in the same way as all the other network nodes.

General purpose supercomputers are the only class of computer system that offer all of the above characteristics; other types of systems have some, but not all of them.

In order for supercomputers to maintain their leadership position in general purpose computing, supercomputer architects and designers face several key challenges. A list of these challenges follows; although we list and discuss them separately, they are closely interrelated.

(1) Maintaining good vector/scalar performance balance: this balance is what leads to dependable, high performance for a wide variety of applications, algorithms, and programming styles.

(2) Supporting scalability: supercomputers have evolved from uniprocessors to small-scale multiprocessors. It is evident that future supercomputers will require increasing numbers of processors. This means that key characteristics of future system architectures must be scalable.

(3) Increasing memory system capacity and performance: this must be done while maintaining programmability and usability. This challenge is closely related to the scalability problem.

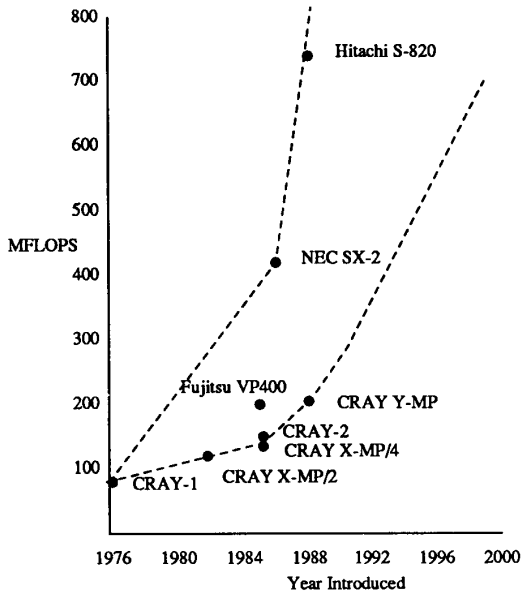
(4) Providing high performance I/O and networking: data movement is always the most difficult problem for the effective use of supercomputers. The performance of future supercomputers will ultimately be measured by how fast they can move data both within the system and across a network.

2. Balanced Vector/Scalar Performance

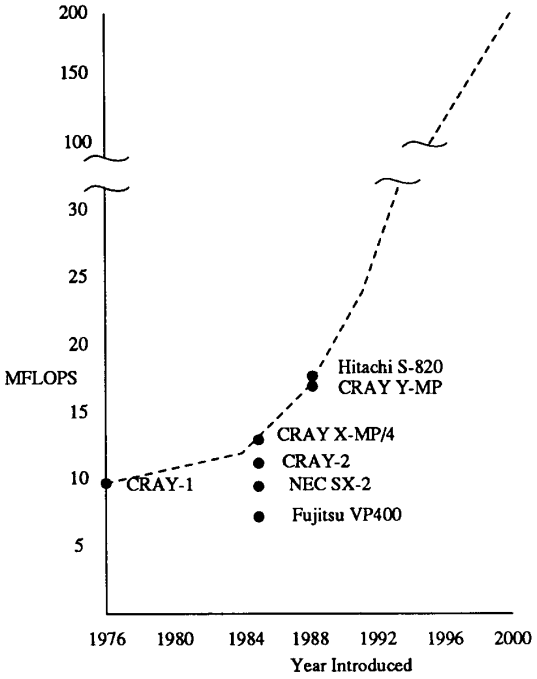
High performance is what makes supercomputers super, and balanced performance is one attribute that makes them general purpose.

Using data for the Livermore FORTRAN Kernels [McMa86], Fig. 1 contains single-processor vector (Fig. 1a) and scalar (Fig. 1b) performance data for Cray

CH2916-5/90/0000/0796/$01.00 © IEEE



a) Single Processor Vector Performance



b) Scalar Performance

Fig. 1. Supercomputer Performance

Research supercomputers and three other current-generation supercomputers. While other benchmarks may better represent true supercomputer problems, the 24 Livermore kernels have a long history with benchmark results available for virtually every supercomputer built. The performance data for each machine are the most up-to-date that we have; admittedly, the data for Cray Research machines are more recent than the others. Although the method is imperfect, we chose the 8 fastest kernels for each machine to characterize its vector performance and the 8 slowest to characterize its scalar performance. Harmonic means are used to summarize the sets of 8 values.

Table 1 illustrates the balance between vector and scalar processing that is achieved in the systems. In the table, the **vector balance point** is defined to be the fraction of vector operations that yields equal time being spent in vector and scalar modes. For example, if a system is capable of 9 Mflops in vector mode, and 1 Mflops in scalar code, equal time will be spent in each mode if a code is .9 vector and .1 scalar; in this case, the vector balance point is defined to be .9. By using this measure, we do not mean to imply that it is optimal for a system to spend equal time in vector and scalar modes. However, if the vector balance point is significantly below the typical level of vectorization, then vector hardware resources are idle a high percentage of the time. In this case, it is possible that these hardware resources could have been used more effectively in other ways.

Table 1 shows that vector balance points are as low as .9 in the CRAY-1 ,with other Cray Research processors slightly higher. Other supercomputers in Table 1 have higher vector balance points ranging up to .98 (requiring 98% vector code before there is equal utilization of vector and scalar hardware).

Table 1: Supercomputer Performance Balance

	Cray 1S	Cray 2S	Cray X-MP	Cray Y-MP	Hit. S820	NEC SX2	Fuji. VP400
vect. perf. (mflops)	85.0	151.5	143.3	201.6	737.3	424.2	207.1
scal. perf. (mflops)	9.8	11.2	13.1	17.0	17.8	9.5	6.6
vect. bal. point	.90	.93	.92	.92	.98	.98	.97

To provide good overall performance in the future, it is necessary for both components of the vector/scalar performance ratio to be improved proportionately. The following subsections discuss ways in which this can be done.

2.1. Vector Performance

Vector processing is the type of processing most closely associated with supercomputers. From a conceptual viewpoint, increasing vector performance is one of the simpler problems facing a supercomputer designer. Replicating functional unit pipelines in a single processor is straightforward; this approach is used in the NEC SX-2 [Wata87], the Hitachi S-820 [EoML88], and the Fujitsu VP series [MiUc83]. Another approach is to clock vector pipelines at "double-rate" with respect to the scalar operations. This has been done in both the Hitachi and Fujitsu systems, and will be done in the NEC SX-3 [Iway89]. The CRAY-2 is similar in that its scalar issue rate is half its vector operation rate. The processors providing the highest vector performance in Fig. 1a all use either multiple vector pipelines or high-rate vector clocks, or both. It should be kept in mind, however, that both these techniques improve the steady-state vector rate, but they do not improve vector startup times. This means that longer vectors are typically required in order to really achieve the higher levels of vector performance.

The dotted lines in Fig. 1 illustrate a "lower bound" vector performance level based on single-width vector pipelines, and an upper performance level where there are increasing numbers of vector pipelines per processor.

As for future supercomputers, both techniques for improving vector performance will no doubt be used. For example, the CRAY Y-MP-16 (C-90) will use double vector pipelines per processor. It is already clear that supercomputers will provide real sustainable vector performance in the range of .5 to 2 GFLOPS per processor by 1992 This performance should be in the 1 to 4 GFLOPS range by the mid '90's, and at least double again toward the end of the century to 2 to 8 GFLOPS per processor by the year 2000. Peak vector performance per processor will be at least twice as great as these sustained performance estimates.

Single-processor vector performance should not be given too much emphasis, however. The most effective general purpose supercomputers may not have the highest per-processor vector performance. The techniques of adding functional unit pipelines and cutting vector clock frequencies increases *computational* bandwidth, but there needs to be a corresponding increase in *memory* bandwidth to support vector computation. The problem of supplying sufficient memory bandwidth will be discussed later. It is sufficient to say here, however, that supplying memory bandwidth in a supercomputer system is more difficult and expensive than supplying computational bandwidth within the processor. Hence, while it is tempting to add large numbers of vector pipelines to get extremely high per-processor vector performance, this might not be the most cost-effective way to get overall system performance. The solution chosen at Cray Research is to divide up a large number of vector pipelines (and memory ports) among multiple processors, with auto-tasking being employed to get them to work together on the same problem, when required. However, the system also has additional flexibility that allows the processor resources to be applied to different jobs, or to work in parallel on non-vector jobs. We predict that as other manufacturers get multiprocessing software in place, the trend toward ever-increasing vector pipeline widths will level off.

2.2. Scalar Processing

Historically, scalar processing has been the key to successful general purpose supercomputers, and it will continue to be so. It has also become difficult to improve scalar performance in a way that maintains a good balance with vector processing.

Relative vector and scalar performance for Cray Research machines is shown in Table 2. In the table, performance is measured in floating point operations per clock period. (found by multiplying MFLOPS from Fig. 1 by the clock period). The table illustrates that over three generations of Cray Research Systems (CRAY-1, X-MP, Y-MP), the vector performance has scaled closely with clock period, with some improvements due to increased memory ports. On the other hand, scalar performance per clock period has declined slightly from one Cray Research generation to the next, primarily due to longer memory latencies (in clock periods). With the 16 processor, double-pipe Y-MP-16 (C-90), the scalar trend will continue, and the vector trend will be accelerated, so the vector/scalar performance ratio will be approximately 50 percent greater than in the Y-MP/8.

Table 2: Normalized Performance

	Cray 1S	Cray X-MP	Cray Y-MP
FLOPs per clock period (vector)	1.063	1.218	1.209
FLOPs per clock period (scalar)	0.122	0.111	0.102

This trend toward higher vector/scalar ratios is mitigated somewhat by better quality vectorizing compilers, algorithms better adapted to vector processing, increased problem sizes, and more experienced programmers, all of which lead to less processing in scalar mode. Nevertheless, in the future, we see the need for a "quantum step" in scalar performance.

There are two complementary approaches to improving scalar performance. The first is to reduce latency (in absolute terms) for each of the major functions: memory, functional units, branches. Of these, memory is becoming the most difficult, because larger memory systems, while providing more memory bandwidth, also tend to have relatively longer latencies. The second approach is higher parallelism (overlap) in scalar processing.

Trends in memory latency for two supercomputer families are shown in Fig. 2. Note that in the figure two different y-axis scales are used. Latency in clock periods (right-hand scale) is plotted with crosses and dashed lines; latency in nanoseconds (left-hand scale) is plotted with squares and solid lines. Among the CRAY-1, X-MP/2, X-MP/4, and Y-MP/8 series, latency in absolute terms has been improving over the years; although not keeping up with latency improvements for other important operations. Furthermore, in relative terms, as measured by clock periods, memory latency has gotten worse. Similar trends are beginning to appear in the NEC SX-2 (1 processor) and SX-3 (4 processors).

For attacking the memory latency problem in general purpose computers, a data cache is commonly used. However, in the vector-intensive supercomputer world,

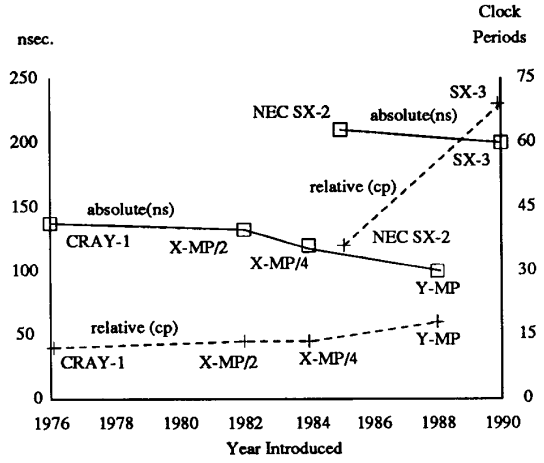


Fig. 2: Memory latency: absolute (ns) and relative (cp).

this approach is often unsuccessful, because 1)large-scale scientific and engineering applications usually manipulate large data structures, which makes use of temporal locality difficult, and 2) arrays are often accessed along rows, columns, and diagonal, or with irregular patterns (requiring a vector gather) which may not take advantage of spatial locality.

In supercomputers, the use of vector registers with vector load instructions supporting a variety of access patterns has proven to be effective for vectorizable codes. For scalar processing, Cray Research supercomputers have relied on data prefetching and local memories to reduce the effects of memory latency. Nevertheless, the increase in relative latency in future supercomputers may expose the limitations of this approach. Fig. 3 illustrates the results of a study where several scalar programs were simulated with varying memory latencies. The Y-MP architecture was used, and the compiler generated code optimized for the given latency. The graph illustrates data points taken at memory latencies of 17 and 40 clock periods; data taken at points in between fit the same curves.

The upper line in Fig. 3 illustrates scalar performance without a data cache; here the scalar performance degrades about 40% when memory latency goes from 17 clock periods to 40. The second line in the graph illustrates performance with a 4K word data cache. Here the overall scalar performance degrades much less. This provides some evidence that in future supercomputers scalar data caches may be warranted. In fact, the trend is already evident in the Japanese supercomputers (having longer memory latencies) which have already adopted data caches.

Because a scalar data cache does not capture vector data, the size at which it is effective may be smaller than the data caches of non-vector processors. It may have other features that distinguish it from more general-purpose caches, as well. These features may tend to make a data cache look more like a large prefetch buffer than a repository for frequently-used data.

The use of a scalar data cache is not without its problems. These problems include: 1) the memory coherence problem: both the traditional inter-processor problem and an intra-processor problem involving vector and scalar references within the same processor, and 2) data access patterns that result in frequent misses or misses coming in bursts. Solutions to both these problems will probably require software assistance, and finding effective solutions will determine the eventual suitability of data caches in future supercomputers.

A more general method of improving scalar performance is to increase parallelism within scalar code. This will no doubt include approaches that use superscalar [Groh90] and dynamic instruction scheduling techniques [Smit89]. **Superscalar** processing enables multiple

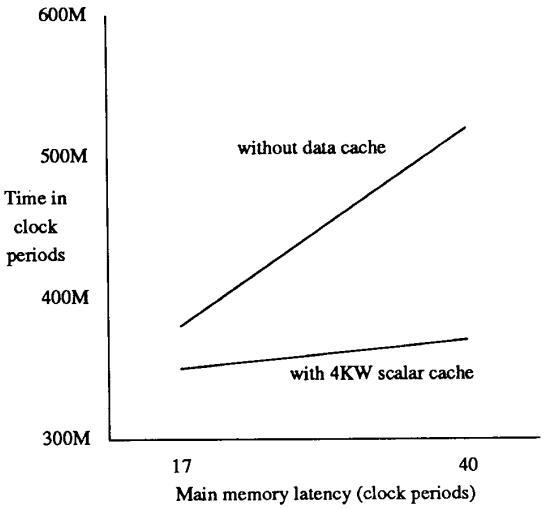


Fig. 3: Simulated performance on 7 scalar benchmarks.

instruction issues per clock, and **dynamic scheduling** involves run-time re-ordering of instructions. These are areas where non-supercomputer architectures have been "out front" in recent years, partly because these smaller systems do not provide vectors, and advanced scalar techniques can help compensate for the lack of vectors.

As with data caches, there are special considerations when scalar units are designed for vector processors. In particular, they only operate on non-vector types of code, so they may not need to support high execution bandwidths. This is particularly true with current compiler technology, where the degree of multiple instruction issue may be less important than dynamic scheduling. Future development of scalar compilation techniques that find parallelism beyond traditional block boundaries might lead to good use of the higher instruction issue bandwidth, however. In any case, the basic character of scalar codes processed by a vector supercomputer may be quite different than in a computer without vectors, and this implies superscalar processing units that may also have a different character.

We also observe that vector supercomputers offer another partial solution to the scalar performance problem. Because vector processing is the fastest single-processor mode of operation, there is always pressure to increase the types of codes that will vectorize. Solutions involve improvements to compiler software, and there is excellent progress being made in this area. In the hardware area, there is the issue of whether vector architecture enhancements can increase vectorization. Of particular importance are codes that involve recurrences, ambiguous subscripts, and conditional statements.

As for forecasting future scalar performance; we see from Fig. 1, that from the CRAY-1 to X-MP, progress was relatively slow, but has speeded up considerably in recent years. These improvements are largely due to improved technology; supercomputers have been benefiting from the same gate-speed improvements as others, but have also been benefiting from significantly-reduced processor chip counts. Beyond future technology-related performance improvements, we see a boost of an additional factor of 1.5 to 2 in the 1995 time-frame (shown as a discontinuity in Fig. 1b) when supercomputer manufacturers incorporate a new generation of scalar processing techniques. Overall, technology and architecture advances translate into about a factor of four scalar performance improvement by the middle of the decade, and another factor of two by the year 2000.

3. Supporting Scalability

The use of parallel processing is the third element of the scalar/vector/parallel processing triad that is offered by general purpose supercomputers. Multiprocessor supercomputer systems are now at the point where they have to face some important "scalability" problems in order to continue with their evolution. The dominant scalability problem involves support of shared main memory for ever-increasing numbers of processors and memory ports. A second scalability problem involves high speed coordination of increasing numbers of parallel tasks.

The major problem is that as the number of processors and vector memory ports is increased, the memory bandwidth must increase correspondingly, but memory latency should remain low. These are contradictory goals. Another important consideration is programmability; the model of memory presented to the parallel application programmer must be simple. To date, in supercomputer systems, this has been the case. The memory model is a flat, shared memory, with all data equally accessible to all the processors. We think the successful multiprocessor supercomputers will be the ones that keep the memory model closest to the currently-supported flat shared model. This is one of the key aspects in making them general purpose.

It is instructive to look at the processor-memory interconnection network in the CRAY Y-MP. The network is illustrated in Fig. 4. This network has evolved as Cray Research multiprocessor systems have evolved. This network is actually a form of multistage interconnection network [GoLi73]. The network is controlled by a form of circuit switching, where all conflicts are worked out early in the memory access process, and all the requests from a given port return to the port in order.

We see two dominant system structures being considered to tackle the memory scalability problem. The

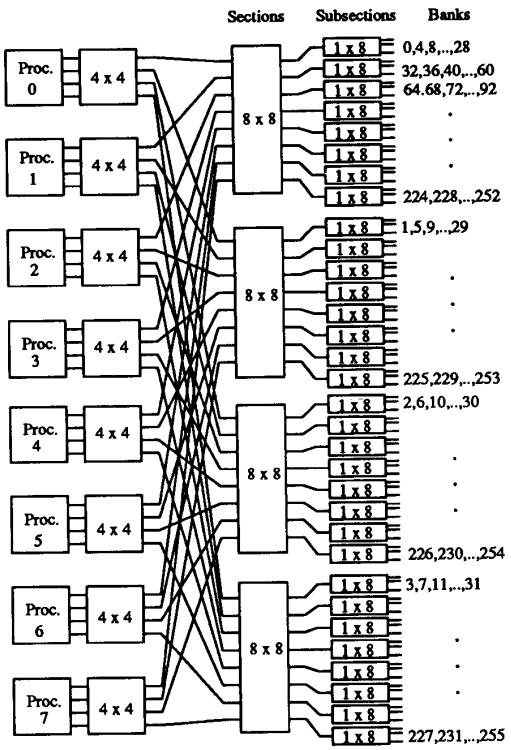
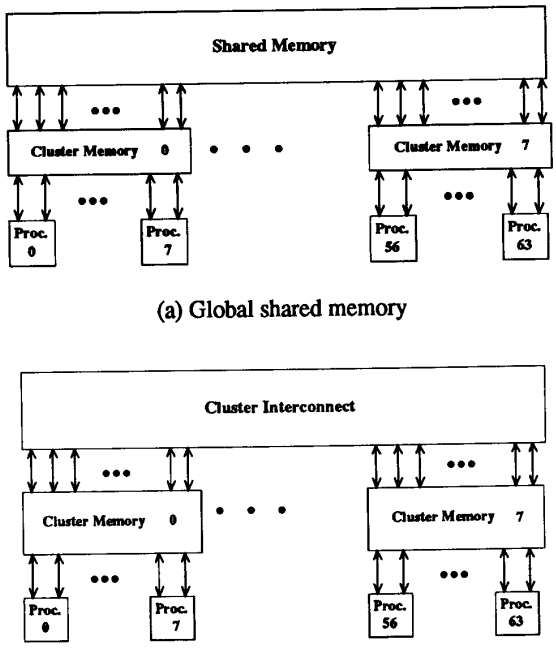


Fig. 4. CRAY Y-MP Interconnection Network.



(a) Global shared memory

(b) Global interconnect

Fig. 5. Two types of clustered system architectures.

first is the extension of the multistage interconnection network, while maintaining a flat memory system architecture. Techniques such as data cacheing, or heavier reliance on processor-based local memories (as in the CRAY-2), will be used for offsetting any increased latencies. The second system structure uses clustered processor/memory subsystems. A clustered system is represented by the experimental Cedar system [KDLS86], developed at Illinois. Here, a number of processors similar to today's systems are clustered together to share a portion of main memory. The clusters are interconnected via a global shared memory (Fig. 5a), or they are interconnected directly (Fig. 5b). The systems in Fig. 5 have eight clusters of eight processors each. In a clustered system, latencies within a cluster are fixed, but global latencies are longer and less predictable. Furthermore the intra-cluster bandwidths may be higher than the inter-cluster bandwidths. The programming model for the clustered system is also more complicated.

To implement the flat system, we need to first recognize that scalable networks are being ued already (Fig 4). The only hurdle to scaling this network is to control it in a more distributed, packet-switched manner. This does not present any fundamental problems, but most research in interconnection networks does not directly apply, and many problems remain to be studied in a more realistic context.

To implement the clustered system, we define a system of approximately the size achievable today (8 to 32 processors) to be a single cluster, and then design a higher-level network to interconnect the clusters.

Both methods for scaling multiprocessors will lead to additional determinism problems in the processors. These translate to sequential consistency problems [DuSB88], and problems implementing vector memory chaining when vector memory accesses return out-of-order.

We conclude that both schemes are natural evolutions from where we are today, and by the mid 90's all the leading-edge supercomputer manufacturers will likely have their solutions to the scaling problem in place. In the near future, supercomputers will have on the order of 64 processors, with 256 to 1024 processors being achievable this decade.

Another important scaling problem has to do with multi-task coordination. In today's systems this is done

via shared registers. Although done with low latency, access to the shared registers is done serially. This leads to the so-called "convoy" problem where the total latency experienced by a series of tasks grows linearly with the number of tasks. This happens at synchronization barriers, for example and is a case of the "hotspot" problem identified some number of years ago [PfNo85]. Solutions may not have to be as general as full-fledged combining networks as proposed in [PfNo85], though. A solution may be a combination of software approaches, and special hardware for coordination of specific tasks [BePo90].

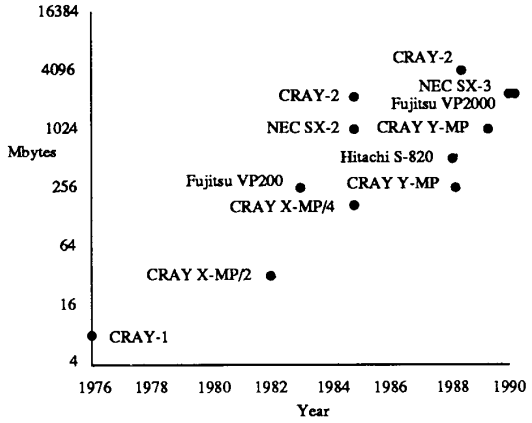
A related issue is the development of software runtime systems and special hardware to assist with scheduling. The CRAY X-MP pioneered this area, but the architecture was specified with no prior experience with autotasking software. Other, more recent approaches [CGMW88] have been able to advance the state-of-the-art, and further advances will no doubt continue in future generations of supercomputers.

4. Large-Scale Memory Systems

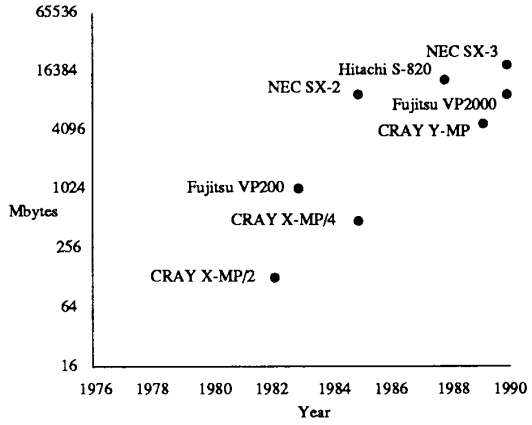
A high-performance memory system must provide low latency for scalar processing, high bandwidth for vector and parallel processing, and large size for "grand challenge" problems and system throughput. To achieve the above requirements, an effective memory hierarchy is necessary. Before we discuss the memory hierarchy further, we review the trends in memory sizes.

As any supercomputer user will testify, memory size translates into overall system performance. General purpose supercomputers often provide very large main memory (usually in SRAMs) in order to (1) support large memory jobs; (2) improve system throughput in a multiprogramming environment; (3) obtain higher memory bandwidth through a greater degree of interleaving. Most general purpose supercomputers also provide extended memory (usually in DRAMs like the SSD in Cray Research machines) to support high speed scratch file accesses and out-of-core applications. Fig. 6a shows the increasing trend in main memory size, and Fig. 6b shows the trend of extended memory. (The survey papers [Neve89] and [ISSS89] were used as sources for much of the data presented in this paper.) The trends in Fig. 6 are driven by the fact that both DRAM and SRAM chips follow the exponential growth pattern of a factor of 4 every three years at roughly the same cost. The size of extended memory is about 4 times that of main memory. This is driven by the fact that, with comparable technologies, the capacity of DRAMs is roughly 4 times that of SRAMs.

Besides technology improvements, increasing problem sizes and the trend towards parallel processing also stimulate a rapid growth in the memory size. We estimate the main memory size will be increased to 4 to 8 GBytes by mid-90, and 16 to 32 GBytes by the year



(a) Main Memory Size



(b) Extended Memory Size

Fig. 6. Supercomputer Memory Sizes.

2000. The extended memory will likely be in the range of 16 to 32 GBytes and 64 to 128 GBytes respectively.

Typical computer systems contain a memory hierarchy as follows.

Virtual Memory	Disk
Real Memory	DRAM
Cache Memory	SRAM

For general purpose supercomputers, the memory hierarchy looks more like the following.

Data Files	Disk
Extended Memory	DRAM
Fast Shared Memory	SRAM
Cache/Local Memory	RAM-on-array

One difficult architectural question to be resolved in the future is whether the extended memory should be included in the address space and completely managed by the operating system, or whether it should be explicitly controlled by programmers as today's SSD. A demand-paged virtual memory system essentially hides the memory hierarchy from the user in order to ease the programming efforts. However, when large memory jobs running with arbitrary access patterns, frequent page faults result in unacceptable performance. Whatever the solution, the memory hierarchy for future supercomputers must (1) be relatively easy to program; (2) be effectively managed by the operating system; and (3) be visible and accessible to the users for optimizations.

The large size of future memory systems also raises the issue of reliability. One memory bank failure may cause the whole memory system to become unavailable. The interconnection between the processor and the memory is also worth particular attention. The increasing number of components requires increasing connections and the increasing connections increases the likelihood of system failures. Providing cost effective solutions for reliable interconnection networks will be a challenge to designers of future supercomputers.

5. High Performance I/O and Networking

As processor speeds and memory sizes of supercomputers increase, users do not just run the same problems faster, they run larger problems. With the aggregate speed of supercomputers increasing at least three to five times each generation, job size has been increasing accordingly, as have I/O bandwidth requirements.

Fig. 7 illustrates aggregate I/O bandwidths that are supported by current and past supercomputer systems. In the figure (and in this section) we define I/O as the transfer of data between the mainframe and peripherals or a network. Even though extended storage, such as Cray Research's SSD, is a key component in handling data movement of large permanent storage files and as a memory paging device, its traffic is not considered here as part of the I/O activity. Fig. 7 (when compared with Fig.1) shows that over the past generations of supercomputers, I/O bandwidths have not always been well-correlated with computational performance. However, the situation seems to be improving, as evidenced by the soon-to-be-available 1990 systems.

Designers have at least two basic choices for I/O processor architectures, both can be found in Cray Research systems. The first is exemplified by the CRAY Y-MP IOS, which uses IO processors that are very flexible and can do relatively complex processing. The second is used in the CRAY-2, where a simple front-end processor controls high-speed channels, with most of the IO management being done by the mainframe's operating system.

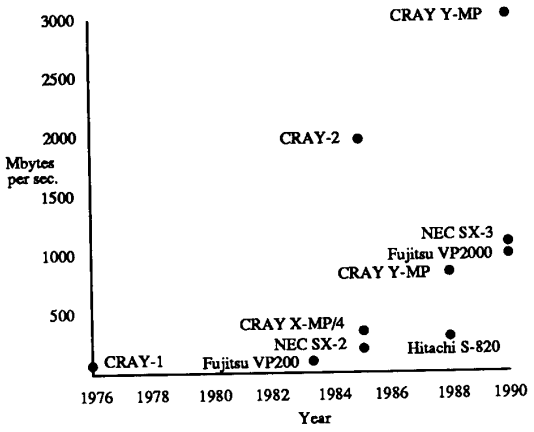


Fig. 7. Supercomputer I/O Performance.

On the peripheral side, the use of disk arrays to enhance the data transfer bandwidth will become common place in the very near future. High speed fiber optic connections between the mainframe and the peripherals will become a key element for supporting I/O activities.

We anticipate that in the next 5 to 10 years, as a result of increasing use of both high performance disk arrays and high-speed networks, data movement will become a more pressing issue for supercomputer vendors. Supercomputers will be able to deliver more than 50 GBytes/sec. I/O transfer rate within the next five years, and this capacity will at least double by the end of the decade.

In order to maintain their general purpose character, future supercomputers will play an important part not merely as standalone systems, but also as an integral part of a networked computing environment. This environment will likely consist of heterogeneous multi-vendor computational elements, file servers, and display devices networked together with high bandwidth connections.

To support "high performance" networking, two issues stand out as most important. The first one is the ability to handle network massages efficiently and the second one is to be able to handle network bulk data transfers. A natural next step is distributed computing on the network, which will totally revolutionalize the way computations are done. Before this can be accomplished, there are serious portability and OS architecture issues that need to be addressed. In this paper, however, we will focus only on the required hardware capabilities.

The whole concept of distributed computing is to treat the network as a backplane for workstations. To support this concept more effectively, the traditional bus based local area network (LAN) is inadequate for

scalability, bandwidth, and latency reasons. A more flexible way is needed to handle heterogeneity and task-level parallelism that can provide a general and systematic way of delivering high performance connectivity in a multicomputer environment. This is why we perceive the emergence of the crossbar switch as a fundamental element for network architectures and an increase in use of industry standard channels.

Point-to-point crossbar nodes will be able to serve as the backplane and the traffic cop of the network. The crossbar switch can come in various speeds depending on the requirements. On the high end, we anticipate a full duplex 200 MByte/sec. HiPPI-based switch will play an important role in the near future, followed by 1 GByte/sec. capabilities by the late '90s. More than a backplane, this kind of switch will allow better network traffic and multiplicity. Shared file servers, disk arrays, online tape systems, visualization devices, etc., can all be connected via this setting.

On the supercomputer side, hardware capabilities will be required for better data sharing across processes on the network and to allow better management of memory and other shared resources. Furthermore, for (hard) realtime environments, predictable, low latency, response time on the network and/or I/O are required. We anticipate predictable response in the tens of microseconds to be imposed on supercomputers in the not too distant future.

6. Summary: the General Purpose Supercomputer of the Future

Based on the above discussion, we foresee future supercomputer architectures that consist of 64 processors by mid-decade and 256 by the end of the decade. These systems will be scalable, either by using multistage interconnection networks in flat systems, or by using clustering. Each processor will provide sustained vector performance approaching 8 Gigaflops by the year 2000. As is the case today, high vector performance will be achieved by multiple pipelines and high-rate vector clocks. Scalar performance per processor will increase at eight-fold, with improvements in architecture making a significant contribution to this improvement. The aggregate system performance on large scale parallel problems will be at least 1 Teraflops by the end of the decade.

Memory systems and I/O will keep pace. Main memories will approach 32 Gbytes, with extended memories four times as large. We expect the architecture of the memory hierarchy to undergo significant improvements during that time. I/O performance will grow to about 50 GBytes/sec., and support for high-speed networking will become a major component of the I/O architecture.

7. References

[BePo90] Beckmann, C. J., and Polychronopolous, C. D., "Fast Barrier Synchronization Hardware," *Supercomputing '90*, Nov. 1990.

[CGMW88] Chastain, M. et. al., "The Convex 240 Architecture," *Supercomputing '88*, Nov. 1988, pp. 321-329.

[DuSB88] Dubois, M., Scheurich, C., and Briggs, F. A., "Synchronization, Coherence, and Event Ordering in Multiprocessors," *Computer*, Feb. 1988, pp. 9-21.

[EoML88] Eoyang, C., Mendez, R. H., and Lubeck, O. M. "The birth of the second generation: The Hitachi S-820/80", *Proc. Supercomputing '88*, Nov. 1988, pp. 296-303.

[KDLS86] Kuck, D. J., et. al., "Parallel Supercomputing and the Cedar Approach," *Science* Feb. 1986, pp. 967-974.

[GoLi73] Goke, L. R., and Lipovski, G. J., "Banyan Networks for Partitioning Multiprocessor Systems," *Proc. 1st Annual Symposium on Computer Architecture*, Dec. 1973, pp. 21-28.

[Groh90] Grohoski, G. F., "Machine Organization of the IBM RISC System/6000 Processor," *IBM Journal of Research and Development*, Jan. 1990, pp. 37-58.

[ISSS89] IEEE Scientific Supercomputer Subcommittee, "Supercomputer Hardware," *Computer*, Nov. 1989, pp. 63-68.

[Iway89] Iwaya, A., "Advanced Architecture and Technology of Supercomputer SX-3 Series," 3rd ISR Supercomputing Workshop, Aug. 1989.

[McMa86] McMahon, F. M., "The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range," UCRL-53745, Lawrence Livermore National Laboratory, Livermore, Calif., Dec. 1986

[MiUc83] Miura, K., and Uchida, K. "FACOM vector processor system: VP-100/VP-200," *Proc. NATO Advanced Res. Workshop on High-Speed Computing*, June 1983.

[Neve89] "Supercomputers: the Next Generation", ONRFE Scientific Information Bulletin 14, 1989, pp. 77-95.

[PfNo85] "Hot Spot Contention and Combining in Multistage Interconnection Networks," *IEEE Transactions on Computers*, Oct. 1985, pp. 943-948.

[Smit89] Smith, J. E., "Dynamic Instruction Scheduling and the Astronautics ZS-1," *Computer*, July 1989, pp. 21-35.

[Wata87] Watanabe, T. "Architecture and performance of NEC supercomputer SX system," *Parallel Computing*, 1987, pp.247-255.