Gordon Bell

ULTRACOMPUTERS

A Teraflop Before Its Time

The quest for the Teraflops Supercomputer to operate at a peak speed of 10^{12} floating-point operations per sec is almost a decade old, and only one three-year computer generation from being fulfilled. The acceleration of its development would require an ultracomputer. First-generation, ultracomputers are networked computers using switches that interconnect thousands of computers to form a multicomputer, and cost $50 to $300 million in 1992. These scalable computers are also classified as massively parallel, since they can be configured to have more than 1,000 processing elements in 1992. Unfortunately, such computers are specialized since only highly parallel, coarse-grained applications, requiring algorithm and program development, can exploit them. Government purchase of such computers would be foolish, since waiting three years will allow computers with a peak speed of a teraflop to be purchased at supercomputer prices ($30 million), due to advancements in semiconductors and the intense competition resulting in "commodity supercomputing." More important, substantially better computers will be available in 1995 in the supercomputer price range if the funding that would be wasted in buying such computers is instead spent on training and software to exploit their power.

In 1989 I described the situation in high-performance computers in science and engineering, including several parallel architectures that could deliver teraflop power by 1995, but with no price constraint [2]. I predicted either of two alternatives: SIMDs with thousands of processing elements or multicomputers with 1,000+ interconnected, independent computers, could achieve this goal. A shared-memory multiprocessor looked infeasible then. Traditional, multiple vector processor supercomputers such as Crays would simply not evolve to a teraflop until 2000. Here is what happened.

1. During 1992, NEC's four-processor SX3 is the fastest computer, delivering 90% of its peak 22Gflops for the Linpeak benchmark, and Cray's 16-processor YMP C90 provides the greatest throughput for supercomputing workloads.

2. The SIMD hardware approach that enabled Thinking Machines to start up in 1983 and obtain DARPA funding was abandoned because it was only suitable for a few, very large-scale problems, barely multiprogrammed, and uneconomical for workloads. It is unclear whether large SIMDs are "generation"-scalable, and they are clearly not "size"-scalable. The main result of the CM2 computer was 10Gflop-level of performance for large-scale problems.

3. Ultracomputer-sized, scalable multicomputers (smC) were introduced by Intel and Thinking Machines, using "Killer" CMOS, 32-bit microprocessors. These product introductions join multicomputers from companies such as Alliant, AT&T, IBM, Intel, Meiko, Mercury, NCUBE, Parsytec, and Transtech. At least Convex, Cray, Fujitsu, IBM, and NEC are working on new-generation smCs that use 64-bit processors. By 1995, this score of efforts, together with the evolution of fast, LAN-connected workstations will create "commodity supercomputing." The author

advocates workstation clusters formed by interconnecting high-speed workstations via new high-speed, low-overhead switches, in lieu of special-purpose multicomputers.

4. Kendall Square Research introduced their KSR 1 scalable, shared-memory multiprocessors (smP) with 1,088 64-bit microprocessors. It provides a sequentially consistent memory and programming model, proving that smPs are feasible. The KSR breakthrough that permits scalability to allow it to become an ultracomputer is based on a distributed, memory scheme, ALLCACHE™ that eliminates physical memory addressing. The ALLCACHE design is a confluence of cache and virtual memory concepts that exploit locality required by scalable, distributed computing. Work is not bound to a particular memory, but moves dynamically to the processors requiring the data. A multiprocessor provides the greatest and most flexible ability for workload since any processor can be deployed on either scalar or parallel (e.g., vector) applications, and is general-purpose, being equally useful for scientific and commercial processing, including transaction processing, databases, real time, and command and control. The KSR machine is most likely the blueprint for future scalable, massively parallel computers.

Figure 1 shows the evolution of supers (four- to five-year gestation) and micro-based scalable computers (three-year gestation). In 1992, petaflop (10^{15} flops) ultracomputers, costing a half-billion dollars do not look feasible by 2001. Denning and Tichy [7] argue that significant scientific problems exist to be solved, but a new approach may be needed to build such a machine. I concur, based on results to date, technology evolution, and lack of user training.

The teraflop quest is fueled by the massive (gigabuck-level) High Performance Computing and Communications Program (HPCC,

1992) budget and DARPA's military-like, tactical focus on teraflops and massive parallelism with greater than 1,000 processing elements. The teraflops boundary is no different than advances that created electronic calculators (kiloflops), Cray computers (megaflops), and last-generation vector supercomputers (Gflops). Vector processing required new algorithms and new programs, and massively parallel systems will also require new algorithms and programs. With slogans such as "industrial competitiveness," the teraflop goal is fundable—even though competitiveness and teraflops are difficult to link. Thus, HPCC is a bureaucrat's dream. Gigabuck programs that accelerate evolution are certain to trade off efficacy, balanced computing, programmability, users, and the long term. Already, government-sponsored architectures and selected purchasing have eliminated benchmarking and utility (e.g., lacking mass storage) concerns as DARPA focus narrowed on the teraflop. Central purchase of an ultracomputer for a vocal minority wastes resources, since no economy of scale exists, and potential users are not likely to find or justify problems that effectively utilize such a machine without a few years of use on smaller machines.

Worlton describes the potential risk of massive parallelism in terms of the "bandwagon effect," where we make the biggest mistakes in managing technology [23]. The article defines "bandwagon" as "a propaganda device by which the purported acceptance of an idea, product or the like by a large number of people is claimed in order to win further public acceptance." He describes a massively parallel bandwagon drawn by vendors, computer science researchers, and bureaucrats who gain power by increased funding. Innovators and early adopters are the riders. The bandwagon's four flat tires are caused by the lack of systems software, skilled programmers, guide-

posts (heuristics about design and use), and parallelizable applications.

The irony of the teraflops quest is that programming may not change very much even though virtually all programs must be rewritten to exploit the very high degree of parallelism required for efficient operation of the coarse-grained, scalable computers. Scientists and engineers will use just another dialect of Fortran that supports data parallelism.

All computers, including true supers, use basically the same, evolving, programming model for exploiting parallelism: SPMD, a single program, multiple data spread across a single address space that supports Fortran [16]. In fact, a strong movement is directed toward the standardization of High Performance Fortran (HPF) using parallel data structures to simplify programming. With SPMD, the same program is made available to each processor in the system. Shared memory multiprocessors simply share a copy in common memory and each computer of a multicomputer is given a copy of the program. Processors are synchronized at the end of parallel work units (e.g., outermost DO loop). Multicomputers, however, have several sources of software overhead due to communication being message-passing instead of direct, memory reference. With SPMD and microprocessors with 64-bit addressing, multicomputers will evolve to be the multiprocessors they simulate by 1995. Thus, the mainline, general-purpose computer is almost certain to be the shared memory, multiprocessor after 1995.

The article will first describe supercomputing evolution and the importance of size-, generation-, and problem-scalability to break the evolutionary performance and price barriers. A discussion about measuring progress will follow. A taxonomy of alternatives will be given to explain the motivation for the multiprocessor continuing to be the mainline, followed by specific industrial options that illustrate real trade-offs. The final sections describe computer design research activities and the roles of computer and computational science, and government.
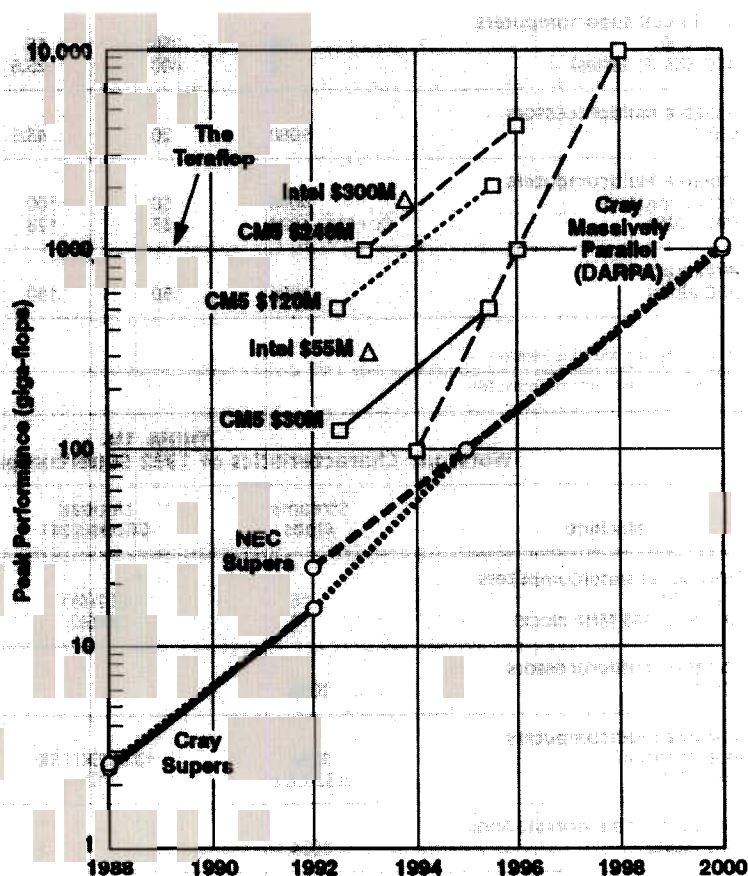
Evolution to the Ultracomputer: A Scalable Supercomputer

Machine scalability allows the $30 million price barrier to be broken for a single computer so that for several hundred million dollars' or a teraflop's worth of networked computers, the ultracomputer, can be assembled. Until 1992, a super-

---

Figure 1. Performance (Gflops) of Cray and NEC supercomputers, and Cray, Intel, and Thinking Machines scalable computers vs. introduction date

computer was defined both as the most powerful central computer for a range of numerically intense computation (i.e., scalar and vector processing), with very large data sets, and costing about $30 million. The notion of machine or "size" scalability, permitting computers of arbitrary and almost unlimited size, together with finding large-scale problems that run effectively have been key to the teraflop race [10]. This is a corollary of the Turing test: People selling computers must be smarter than their computers. No matter how difficult a computer is to use or how poorly it performs on real workloads, given enough time, someone may find a problem for which the computer performs well. The problem owner extols the machine to perpetuate government funding.

In 1988, the Cray YMP/8 delivered a peak of 2.8 Gflops. By 1991,

the Intel Touchstone Delta (672 node multicomputer) and the Thinking Machines CM2 (2K processing element SIMD), both began to supply an order of magnitude more peak power (20 gigaflops) than supercomputers. In supercomputing, peak or advertising power is the maximum performance that the manufacturer guarantees no program will exceed. Benchmark kernels such as matrix operations run at near peak speed on the Cray YMP. Multicomputers require O(25,000) matrices to operate effectively (e.g., 14 Gflops from a peak of 20), and adding processors does not help. For O(1,000) matrices that are typical of super-

computer applications, smCs with several thousand processing elements deliver negligible performance.

Supers 1992 to 1995

By mid-1992 a completely new generation of computers have been introduced. Understanding a new generation and redesigning it to be less flawed takes at least three years. Understanding this generation should make it possible to build the next-generation supercomputer class machine, that would reach a teraflop of peak power for a few, large-scale applications by the end of 1995.

Table 1 shows six alternatives for high-performance computing, ranging from two traditional supers, one smP, and three "commodity supers" or smCs, including 1,000 workstations. Three metrics characterize a computer's performance and workload abilities. Linpeak is the operation rate for solving a system of linear equations and is the best case for a highly parallel application. Solving systems of linear equations is at the root of many scientific and engineering applications. Large, well-programmed applications typically run at one-fourth to one-half this rate. Linpack 1K × 1K is typical of problems solved on supercomputers in

Table 1a.
Physical Characteristics of 1992 Supercomputing Alternatives

Machine	Proc's #	Clk MHz	Peak Gflops	Price $M	Mp.size GB	I/O.Bw GB
Traditional supercomputers						
Cray C90	16	250	16	30	2(16)**	13.6
NEC SX3 (R series)	4	400	25.6	25	8(64)	5.4
Scalable Multiprocessors						
KSR 1	1088	20	43.5	30	34.8	15.3
Scalable Multicomputers						
Intel Paragon§	4096	50	300	55	128	?
TMC CM5†	Cc+Cio+1024	33	128	30	32+	?
Workstations						
DEC alpha	1024	150	150	20	32	100

*Author's estimate
**Fast access blocked, secondary memory
§Available: Q1 1993. Four processors form a multiprocessor node; a fifth processor handles communication.
†Cc := control computer; Cio := i/o computers;

Table 1b.
Workload Characteristics of 1992 Supercomputing Alternatives

Machine	Streams #jobs	Linpeak Gflops(size)	Lin1K Gflops	LFKWorkload Mflops
Traditional supercomputers				
Cray C90	16	13.7(4K)	9.7	16 × 44
NEC SX3 (345 MHz clock)	4	20(6K)	13.4	4 × 39
Scalable multiprocessors				
KSR 1	1088		513(32Proc.)	1088 × 6.6
Scalable multicomputers				
Intel Paragon	1024	13.9(25K)/.5K	?	1K × 6*
TMC CM5	≤32 Cc's	70*	32 × ?	≤32 × 6*
LAN-connected workstations				
DEC Alpha	1024		64	1K × 15

*Author's estimate
Unavailable from manufacturer

1992. The Livermore Fortran Kernels (LFK) harmonic mean for 24 loops and 3 sizes, is used to characterize a numerical computer's ability, and is the worst-case rating for a computer as it represents an untuned workload.

New-generation, traditional or "true" multiple vector processor supercomputers have been delivered by Cray and NEC that provide one-fourth to one-eighth the peak power of the smCs to be delivered in 1992. "True" supercomputers use the Cray design formula: ECL circuits and dense packaging technology to reduce size, allow the fastest clock; one or more pipelined vector units with each processor provide peak processing for a Fortran program; and multiple vector processors communicate via a switch to a common, shared memory to handle large workloads and parallel processing. Because of the dense physical packaging of high-power chips and relatively low density of the 100,000 gate ECL chips, the inherent cost per operation for a supercomputer is roughly 500 to 1,000 peak flops/$ or 4 to 10 times greater than simply packaged, 2 million transistor "killer" CMOS microprocessors that go into leading edge workstations (5,000 peak flops/$). True supercomputers are not in the teraflops race, even though they are certain to provide most of the supercomputing capacity until 1995.

Intel has continued the pure multicomputer path by introducing its third generation, Paragon with up to 4K Intel i860 microprocessor based nodes, each with a peak of 4 × 75 Mflops for delivery in early 1993. Intel is offering a 6K node, $300 million, special ultracomputer for delivery in late 1993 that would provide 1.8 peak teraflops or 6K peak flop/$.

In October 1991, Thinking Machines Corp. (TMC) announced its first-generation multicomputer consisting of Sun servers controlling up to 16K Sparc microprocessor-based computational nodes, each with four connected vector processors to be delivered in 1992. The CM5 workload ability of a few Sun servers is small compared to a true supercomputer and to begin to balance the computer for general utility would require several disks at each node. The expected performance for both supercomputer-sized problems and a workload means that the machine is fundamentally special-purpose for highly parallel jobs. The CM5 provides 4,300 peak flops/$.

In both the Paragon and CM5 it is likely that the most cost-effective use will be with small clusters of a few (e.g., 32) processors.

1995 Supers: Vectors, Scalable Multicomputers or Multiprocessors

Traditional or "true" supercomputers have a significant advantage in being able to deliver the computational power during this decade because they have evolved for four, four-year generations for almost 20 years,[1] and have an installed software-based, programming paradigm, trained programmers, and wider applicability inherent in finer granularity. The KSR-1 scalable multiprocessor runs traditional, fine-grained supercomputer Fortran programs, and has extraordinary single-processor scalar and commercial (e.g., transaction processing) throughput.

The smCs are unlikely to be alternatives for general-purpose computing or supercomputing because they do not deliver significant power for scalar- and finer-grained applications that characterize a supercomputer workload. For example, the entire set of accounts using the Intel smC at Cal Tech is less than 200, or roughly the number of users that simultaneously use a large super. Burton Smith [20] defines a general-purpose computer as: 1. Reasonably fast execution of any algorithm that performs well on another machine. Any kind

---

[1]Cray 1 (1975), Cray IS (1978), Cray XMP-2, 4 (1982, 1984), Cray YMP-8 (1988), and Cray C-90 (1992)

---

of parallelism should be exploitable. 2. Providing a machine-independent programming environment. Software should be no harder to transport than to any other computer. 3. A storage hierarchy performance consistent with computational capability. The computer should not be I/O bound to any greater extent than another computer.

Whether traditional supercomputers or massively parallel computers provide more computing, measured in flops/month by 1995 is the object of a bet between the author and Danny Hillis of Thinking Machines [11]. Scalable multicomputers (smCs) are applicable to coarse-grained, highly parallel codes and someone must invent new algorithms and write new programs. Universities are rewarded with grants, papers, and the production of knowledge. Hence, they are a key to utilizing coarse-grained, parallel computers. With pressure to aid industry, the Department of Energy laboratories see massive parallelism as a way to maintain staffs. On the other hand, organizations concerned with cost-effectiveness, simply cannot afford the effort unless they obtain uniquely competitive capabilities.

Already, the shared, virtual memory has been invented to aid in the programming of multicomputers. These machines are certain to evolve to multiprocessors with the next generation. Therefore, the mainline of computing will continue to be an evolution of the shared memory multiprocessor just as it has been since the mid-1960s [4]. In 1995, users should be able to buy a scalable parallel multiprocessor for 25K peak flops/$, and a teraflop computer would sell for about $40 million.

Measuring Progress

Supercomputer users and buyers need to be especially cautious when evaluating performance claims for supercomputers. Bailey's [1] twelve ways to obfuscate are:

1. Quote 32-bit performance results as 64-bit performance
2. Present inner kernel performance as application performance, neglect I/O
3. Employ assembly, micro-code and low-level code
4. Scale up problem size to avoid performance drop-off when using large numbers of processors with overhead or inter-communication delays or limits
5. Quote performance results extrapolated to a full system based on one processor
6. Compare results against unoptimized, scalar Cray code
7. Compare direct run-time with old code on an obsolete system (e.g., Cray 1)
8. Quote additional operations that are required when using a parallel, often obsolete, algorithm
9. Quote processor utilization, speedup, or Mflops/$ and ignore performance
10. Mutilate the algorithm to match the architecture, and give meaningless results
11. Compare results using a loaded vs. dedicated system
12. If all else fails, show pictures and videos

Each year a prize administered by the ACM and IEEE Supercomputing Committees awards a prize to reward practical progress in parallelism, encourage improvements, and demonstrate the utility of parallel processors [9]. Various prize categories recognize program speedup through parallelism, absolute performance, performance/price, and parallel compiler advances. The first four years of prizes are given in Table 2.

The 1987 prize for parallelism was won by a team at Sandia National Laboratory using a 1K node NCUBE and solving three problems. The team extrapolated that with more memory, the problem could be scaled up to reduce overhead, and a factor of 1,000 (vs. 600) speedup could be achieved. An NCAR Atmospheric Model running on the Cray XMP had the

highest performance. In 1989 and 1990, a CM2 (SIMD with 2K processing elements) operated at the highest speed and the computation was done with 32-bit floating point numbers. The problems solved were 4 to 16 times larger than would ordinarily have been solved with modified problems requiring additional operations [1].

Benchmarking

The benchmarking process has

been a key to understanding computer performance until the teraflop started and peak performance replaced reality as a selection criterion. Computer performance for an installation can be estimated by looking at various benchmarks of similar programs, and collections of benchmarks that represent a workload [5]. Benchmarks can be synthetic (e.g., Dhrystones and Whetstones), kernels that represent real code (e.g., Livermore Loops, National Aerody-

Scalability

The perception that a computer can grow forever has always been a design goal (e.g., IBM System/360 [c1964] provided a 100:1 range, and VAX existed at a range of 1,000:1 over its lifetime). Ideally, one would start with a single computer and buy more components as needed to provide size scalability. Similarly, when new processor technology increased performance, one would add new-generation computers in a generations-scalable fashion. Ordinary workstations provide some size and generation scalability, but are LAN-limited. By providing suitable high-speed switching, workstation clusters can supply parallel computing power and are an alternative to scalable multicomputers. Problem scalability is the ability of a problem, algorithm, or program to exist at a range of sizes so it can be used efficiently and correctly on a given, scalable computer.

Worlton [23] discusses Amdahl's law and the need for a very large fraction, F, of a given program to be parallel, when using a large number of processors, N to obtain high efficiency, $E(F,N)$.

$$E(F,N) = 1 / (F + N \times (1 - F))$$

Thus, scaling up slow processors is a losing proposition for a given fraction of parallelism.

For an efficiency of 50%, requires $1 - F = 1 / (N - 1)$; for 1,000 processors F must be 0.999 parallel.

Size Scalability: Locality is the Key
Size scalability has become an academic topic [12, 20]. Size scalability simply means that a very large computer such as the ultracomputer can be built. Typical definitions fails to recognize cost, efficiency, and whether such a large-scale computer is practical (affordable) in a reasonable time scale. For example, a cross-point switch is supposedly not scalable because the switching area grows $O(n2)$ even though switch cost may be insignificant. Much supercomputer cost is the processor-memory switch, and scaling is accomplished by having different switch/cabinets for different size computers. For example, Cray Research, Intel, and Thinking Machines all build active switches into their cabinets into which computing elements are plugged. The CM5 and KSR computers require switching cabinets when going beyond the first levels of their hierarchies. KSR interconnects processors using a near zero cost ring, since each node just connects to its next neighbor. No computers are truly scalable in a linear fashion.

A size-scalable computer is designed from a small number

namic Simulation), numerical libraries (e.g., Linpack for matrix solvers, FFT), or full applications (e.g., SPEC for workstations, Illinois's Perfect Club, Los Alamos Benchmarks). No matter what measure is used to understand a computer, the only way to understand how a computer will perform is to benchmark the computer with the applications to be used. This also measures the mean time before answers (mtba), a most important measure of computers productivity.

Several Livermore Loop metrics, using a range of three vector lengths for the 24 loops, are useful: the arithmetic mean typifying the best applications (.97 vector ops), the arithmetic mean for optimized applications (.89 vector ops), geometric mean for tuned workload (.74 vector ops), harmonic mean for untuned workload (.45 vector ops), and harmonic mean compiled as a scalar for an all-scalar operation (no vector ops). Three Linpack mea-

surements are important: Linpack 100×100, Linpack $1,000 \times 1,000$ for typical supercomputing applications, and Linpeak (for an unconstrained sized matrix). Linpeak is the only benchmark that is run effectively on a large multicomputer. Massive multicomputers can rarely run an existing supercomputer program (i.e., the dusty deck programs) without a new algorithm or new program. In fact, the best benchmark for any computer is whether a manufacturer can and is

of basic components, with no single bottleneck component, so the computer can be incrementally expanded over its designed scaling range, delivering linear incremental performance for a well-defined set of applications. The components include computers, processors or processing elements, memories, switches, and cabinets. For example, since the highly parallel computers are interconnected by switches, the bandwidth of the switch should increase linearly with processing power. It is clear that a balanced, general-purpose teraflop computer is not feasible based on I/O considerations. For example, if I/O requirements increase with performance as in a general-purpose computer, then roughly 0.1 terabytes/sec or 20,000 5MB/sec disks operating in parallel would be needed to balance the computer (about one bit of data transferred for every flop). Emitting video from a computer for direct visualization is one way to effectively utilize the I/O bandwidth and reduce mass storage.

The key to size scalability is a belief in spatial and temporal locality, since very large systems have inherently longer latencies than small, central systems. All supercomputers are predicated to some degree on locality (i.e., once a datum is accessed a near physical neighbor will be accessed [spatial]

and the same datum will be repeatedly accessed [temporal]). Locality of program execution is the phenomenon that allowed the first one-level store computer, Atlas to be built. This led to the understanding of paging, virtual memory, and working sets that are predicated on locality [6]. Caches exploit spatial and temporal locality automatically. Large register arrays, including vector registers are mechanisms for a compiler to exploit and control locality.

In 1989, building computers that scaled economically over a very wide range of implementations looked impractical because of the enormous interconnection bandwidth requirements. The Cray YMP 8 and CM2 scaled over a range of eight (eight processors in the Cray, and 8K to 64K processing elements in the CM2). The Cray C-90 scaling range is 16, and other implementations of the Cray architecture increase the performance range a factor of 5. Also, four C90s can be interconnected providing 64 peak Gflops. The CM5 scaling range is 32 for 1,024 computers and KSR 1 has a range of 128 (8 to 1,088) processor-memory pairs. The CM5 scaling range extends to 512 with 16K computers as a $480 million ultracomputer. In practical terms, a scalable computer is one that can exist at the largest size an organization (including a country) will ever buy.

Generation (time) Scalability

Generation (time) scalability is as important as size scalability, since the basic microprocessor nodes become obsolete every three years. Furthermore, the time to find an algorithm and write a program is long, requiring significant investment that needs to be preserved. A generation-scalable computer can be implemented in a new technology, and thus take advantage of increased circuit and packaging technologies. Since CMOS evolves rapidly, the interconnection bandwidth must grow at the same rate as processing speed and memory. For example, it is irrelevant to have a design that can exploit "next-generation" microprocessor nodes without increasing the switch bandwidth and decreasing the overhead and latency proportionally. All characteristics of a computer must scale proportionally: processing speed, memory speed and sizes, interconnect bandwidth and latency, I/O, and software overhead in order to be useful for a given application.

Problem Scalability: Key to Performance

Problem scalability defines whether an application is feasible on a computer with given granularity characteristics. In practical terms, problem scalability means that a program can be made large enough to operate efficiently

willing to benchmark a user's programs.

Two factors make benchmarking parallel systems difficult: problem scalability (or size) and the number of processors or job streams. The maximum output is the perfectly parallel workload case in which every processor is allowed to run a given-size program independently and the uniprocessor work-rate is multiplied by the number of processors. Similarly, the minimum wall clock time should be when all processors are used in parallel. Thus, performance is a surface of varying problem size (scale) and the number of processors in a cluster.

The Commercial Alternatives
The quest generated by the HPCC Program and the challenge of parallelism has attracted almost every computer company lashing together microprocessors (e.g., Inmos Transputers, Intel i860s, and Sparcs) to provide commodity, multicomputer supercomputing in every price range from PCs to ultracomputers. In addition, traditional supercomputer evolution will continue well into the twenty-first century. The main fuel for growth is the continued evolution of "killer" CMOS microprocessors and the resulting workstations.

on a computer with a given granularity. Problems such as Monte Carlo simulation and "ray tracing" are "perfectly parallel" since their threads of computation almost never come together. Obtaining parallelism (i.e., performance) has turned out to be possible with new algorithms and new codes. Problem granularity (operations on a grid point/data required from adjacent grid points) must be greater than a machine's granularity (node operation rate/node-to-node communication data-rate) in order for a computer to be effective. Several kinds of messages must pass among distributed computer nodes: *a priori* messages that a compiler can generate to ensure that data is available to a node before it is needed; computer address data, requiring messages for both address and data; and various broadcast and synchronization information. For example, message-passing and random access references are sufficiently large to render today's multicomputers, ineffective for classical benchmarks such as the Livermore kernels. Denning and Tichy [7] discuss the effects of problem scalability and granularity on performance.

In the case of models of physical structures, as the problem size is scaled up by increasing the grid points, the work or potential parallelism and memory increases at least

$O(n^3)$, where n is the problem dimension. The communication with other cells only grows as $O(n^2)$. Thus, communication overhead can often be reduced or ignored compared to the computation if a problem can be made large enough (i.e., get enough grid points) to still fit in primary memory of a distributed node. For example, in solving LaPlace's equation computation is 7 n^3 (the time to average the neighboring points) and on a distributed memory computer, the communication is 6 n^2, where n is the problem dimension. Thus, a 100Mflop computer intercommunicating at 1 Megaword per sec is balanced when the computation time of .07 n^3 microsec equals the communication time of 6 n^2 microsec. That is, n must be larger than 86, to hold the 3d array of 640K points or just 5MB. About 1 microsec, however, is required to send or receive a word on multicomputers, representing an opportunity cost of 2 × 100 operations. For a problem that would fill a 32MB memory, n is about 160. This size problem requires 0.3 sec of computation and 2 × 0.15 sec of send and receive time in which the processor is idle. The iteration time is 0.6 seconds, resulting in a computation rate of 50Mflops with only 0.15 sec of communication link time, and smaller problems would run more slowly. Since the memory is full, the problem cannot be

further scaled to increase efficiency.

For some problems, scaling a problem may produce no better results than a coarser grain, and less costly solution. Another risk of problem-scaling is to exacerbate the limited I/O capability. For example, the 25K × 25K matrix that Intel's Delta used for a 15Gflop matrix multiply takes 5GB of memory, and 500 sec to load using 10MB disks. The matrix multiply time is ≈2000 sec. In contrast, a C90 achieves peak power on a 4K × 4K matrix that takes only 10 sec to compute and 128MB to store. Contrast this structure with computers connected via Ethernet, which has a total bandwidth of 1 million words per sec for all nodes, and message-passing overheads of at least 1,000 microsec (100,000 operations on a 100Mflop computer). However, given a large enough problem and enough memory per node, even such a collection of workstations can be scaled to have a long enough grain to be effective for solving the preceding problem.

Figure 2 shows the structure of a basic unit of multithreaded computation independent of whether it is run on a SIMD, multiprocessor or multicomputer, or has distributed or shared components. Hockney and Jesshope [14] formulated models that predict performance as a function of a computer's characteristics and

In 1989 the author estimated that several traditional supers would be announced this year by U.S. companies. Cray Research and NEC have announced products, and Fujitsu has announced its intent to enter the U.S. supercomputer market. Seymour Cray formed Cray Computer. Supercomputer Systems Inc. lacks a product, and

DARPA's Tera Computer Company is in the design phase. Germany's Suprenum project was stopped. The French Advanced Computer Research Institute (ACRI) was started. Numerous Japanese multicomputer projects are underway. Supercomputers are on a purely evolutionary path driven by increasing clock speed, pipelines, and processors. Clock increases in speed do not exceed a

factor of two every five years (about 14%). In 1984, a committee projected the Cray 3 would operate in 1986. In 1989, the 128-processor Cray 4 was projected to operate at 1GHz. in 1992. In 1992, the 16-processor Cray 3, projected to operate at 500MHz, was stopped. NEC has pioneered exceptional vector speeds, retaining the title of the world's fastest vector processor. The NEC vector processor uses 16 parallel pipelines and performs

problem parallelism. The computation starts with a sequential thread (1), followed by supervisory scheduling (2), where the processors begin threads of computation (3), followed by inter-computer messages that update variables among the nodes when the computer has a distributed memory (4), and finally synchronize prior to beginning the next unit of parallel work (5). The communication overhead period (3) inherent in distributed memory structures is usually distributed throughout the computation and possibly completely overlapped. Message-passing overhead (send and receive calls) in multicomputers can be reduced by specialized hardware operating parallel with computation. Communication bandwidth limits granularity, since a certain amount of data has to be transferred with other nodes in order to complete a computational grain. Message-passing calls and synchronization (5) are nonproductive.

The Paragon should be able to operate on relatively smaller-grained problems than a CM5, since hardware granularity (node operation rate per internode communication rate) appears to be lower. The CM5 requires a problem-grain length of at least 102 operations per word transferred (128Mflops/10/8Mwords per sec); this means that for every word transmitted to another node, at least 102 operations

Figure 2. Structure of multithreaded, parallel computation

have to be carried out in order to avoid waiting for data. Paragon is projected to be 3 operations per word. To reduce this effective grain size, the problem just has to be scaled up to "bundle" a large number of computational grid points within a physical node to reduce internode communication. Very large-grained problems typically have several thousand operations per grain in order to reduce all overheads.

Additionally, the message-passing time is lost time that also limits system performance. Assuming the operating system is not involved in passing a message, David Culler, at Berkeley has measured 3.3 microsec for a CM5 to send and receive a message plus 1.0 microsec per word sent or received. During this lost time of 2 microsec, 256 operations could have been carried out. Paragon attempts to reduce the message-passing overhead and increase the bandwidth by having a sepa-

rate processor and block transfer hardware manage message transfers.

In a distributed memory multiprocessor no messages are explicitly passed; however, hardware processes messages and caches data. Multiprocessors avoid several sources of software overhead inherent in 1992 multicomputers: converting the addresses of each 32-bit computer into a single, >32-bit global address; deciding in which computer to locate data, including the possibility of relocating and renaming data dynamically— i.e., controlling locality by simulating caches; passing messages containing variables that other nodes need just in time for another computer to use; and passing computed addresses and data when random access of memory is required. ■
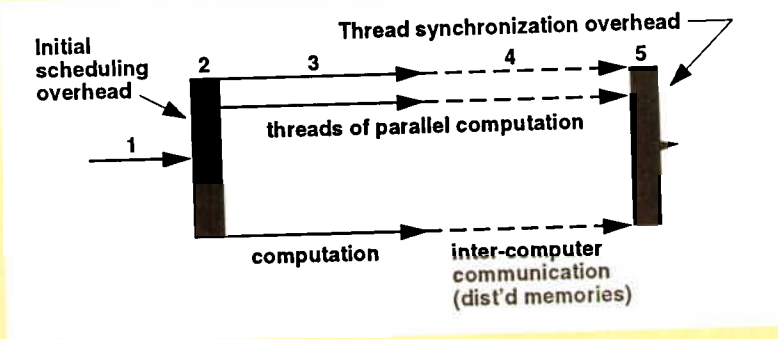
operations at a 6.4Gflop rate. Fujitsu's supercomputer provides two, 2.5Gflop vector units shared by four scalar processors. Every four-or five years the number of processors is doubled, providing a gain of 18% per year. The C90 increased the clock frequency by 50% to 250MHz, doubled the number of pipelines and the number of processors over the YMP. Figure 1 projects a 1995 Cray supercomputer that will operate at 100Gflops using a double-speed clock, twice the number of processors, and twice the number of pipelines.

In most production environments, throughput is the measure and the C-90 clearly wins, since it has a factor of four times the number of processors of either Fujitsu or NEC. In an environment where a small number of production programs are run, additional processors with scalar capability may be of little use. Environments that run only a few coarse-grained codes can potentially use smCs for large-grained problems.

The traditional supercomputer market does not look toward high growth because it provides neither the most cost-effective solution for simpler scalar programs, nor the peak power for massively parallel applications. Scalar codes run most cost-effectively on workstations, while very parallel code may be run on massively parallel computers, provided the granularity is high and the cost of writing the new code is low. Despite these factors, I believe traditional supercomputers will be introduced in 2000.

"Killer" CMOS Micros for Building Scalable Computers
Progress toward the affordable teraflop using "killer" CMOS micros is determined by advances in microprocessor speeds. The projection [2] that microprocessors would improve at a 60%-per-year rate,

Table 2.
Gordon Bell Prize Winners

Year	Peformance Gflops	Price/Performance Gflops/$1 million	Speed-up
1987	0.45	0.03	600
1988	1.0	0.05	800
1989	6.0	0.5	1100
1990	14.0	2.0	1800

Table 3.
Contemporary Microprocessor Performance

Micro	Year	Clock (MHz)	I-Spec	f-Spec (Specmarks*)	Spec	Linpack (Mflops)	Lapeak (Mflops)	LFK(hm)	Pk
DEC VAX780	78.2	5*	1	1	1	0.15		0.16	1
DEC Alpha	92.2	200			150†		85†	20†	200†
Fujitsu-VP	92.4	50				>50	95	12.5	108
HP PA	92.2	100	—	—	138	56	67	—	200
HP PA	91.1	66	52	78	102	24		13.3	66
IBM RS6000	91.2	42	33	120	72	27	70	14.5	83
Intel 486 PC	91.3	50	28	15	19	2.0	—	1.8	
MIPS R3000	88.3	25			18	4.2	7	3.6	8
MIPS R4000	92.2	100**	60	77	70	17.5	36	11.5	50
SUN Sparc 2	91.3	40	22	27	25	4.1	—	—	—
1995 Micro	95	200–400			300				400–800

*CISC architecture. A comparable RISC architecture would operate at approx. 2MHz.
**External clock rate is 50MHz.
†Estimate

Tera Taxonomy

The taxonomy of the tera candidates, shown in Figure 3, includes only shared-memory multiprocessors and various multicomputers (MIMDs) . A superscalar or extra long word RISC, a vector processor, and a processor with thousands of processing elements are just SIMD processors, since they all have a single instruction stream.

Distributed (Boudoir) vs. Centralized (Dance Hall) Computing: Locality Beliefs
Two attributes structure the taxonomy: multiprocessors vs. multicomputers; and scalability using a physically distributed vs. a central memory. Scalability measures whether it is practical to construct ultracomputers. A memory is either centralized in a pool, "dance hall" (processors—switch—memories); or distributed with each processor enabling scalability, "boudoir" ([processor-memory]—switch), and is the key to scalability. Switches bottleneck overall performance and limit system size in every computer; thus the switch is the determinant of a computer's scalability. Switches such as the CM5 has, allow a very large network of computers to be put together just as an arbitrary number of workstations or telephones can be interconnected. Although large switches permit arbitrary peak power, the cost is prob-lem granularity, mean time before applications, and limited applicability.

Multiprocessors vs. Multicomputers
The hardware distinction between multiprocessors and multicomputers is whether the system has and maintains a single address space and a single coherent memory and whether explicit messages are required to access memory on other computing nodes as shown in the programming view in Figure 4. The question is similar to RISC vs. CISC, since multicom-

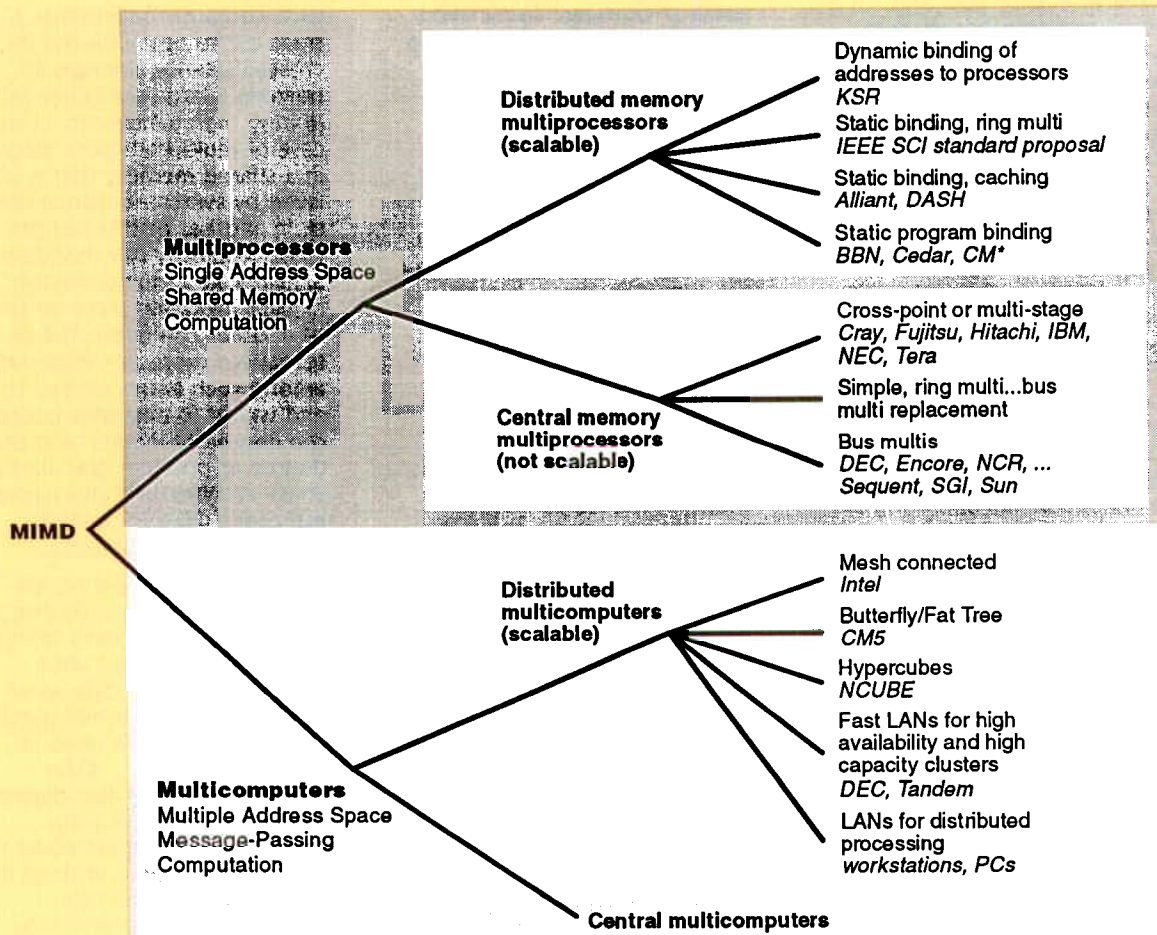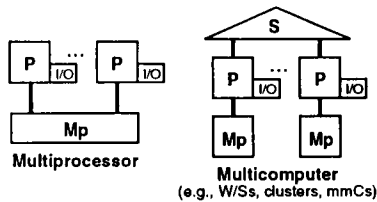**Figure 3.** Taxonomy of multiprocessors and multicomputers

Distributed memory multiprocessors (scalable)
- Dynamic binding of addresses to processors *KSR*
- Static binding, ring multi *IEEE SCI standard proposal*
- Static binding, caching *Alliant, DASH*
- Static program binding *BBN, Cedar, CM**

Central memory multiprocessors (not scalable)
- Cross-point or multi-stage *Cray, Fujitsu, Hitachi, IBM, NEC, Tera*
- Simple, ring multi...bus multi replacement
- Bus multis *DEC, Encore, NCR, ... Sequent, SGI, Sun*

Multiprocessors Single Address Space Shared Memory Computation

Distributed multicomputers (scalable)
- Mesh connected *Intel*
- Butterfly/Fat Tree *CM5*
- Hypercubes *NCUBE*
- Fast LANs for high availability and high capacity clusters *DEC, Tandem*
- LANs for distributed processing *workstations, PCs*

Multicomputers Multiple Address Space Message-Passing Computation

MIMD

Central multicomputers

Multiprocessor

Multicomputer
(e.g., W/Ss, clusters, mmCs)

puter operating systems are evolving to carry out the functions (e.g., address construction, caching, message-passing for data access) that multiprocessor hardware provides. The differences are:

1. Multiprocessors have a single address space supported by hardware. Each computer of a multicomputer has its own address space. Software forms a common, global address space as a concatenation of a computer's node number and the computer node's address to support the SPMD program model.
2. Multiprocessors have a single, uniformly accessible memory and are managed by and provide a single, timeshared operating system programming environment (i.e., Unix). Multicomputers are a collection of independent, interconnected computers under control of a LAN connected, distributed workstation-like operating system. Each computer has a copy or kernel of the operating system.

3. A multiprocessor has a common work queue that any processor may access and be applied to. In a multicomputer, work (programs and their data) is distributed among the computers, usually on a static basis. As the load on the computers or clusters change, work may have to be moved.

Any node in a multiprocessor may run any size job from its shared, virtual memory. In mul-

ticomputers, a job's size is limited by a node's memory size, and a computer is incapable of or ineffective at running a collection of large, scalar programs that typify perfectly parallel applications such as digital simulation.

4. Multiprocessors communicate data implicitly by directly accessing a common memory. Multicomputers explicitly pass messages that may or may not be hidden from the user by hardware and the compiler.

When a multiprocessor is used for parallel processing, data and programs are equally accessible to all processors. Programs and their data must be allocated among computers in order to minimize message passing overhead. To minimize message-passing data may also have to be moved and renamed as in a virtual memory system.

5. Multiprocessors provide a single, sequentially consistent memory and program model. Since message-passing is used to move data in multicomputers, different copies of variables may reside in various computer nodes at one time.
6. Distributed memory multiprocessors have an automatic mechanism, caching, to implicitly control locality. As a datum is accessed it is automatically moved to another processor's memory. With multicomputers every nonlocal access requires software for address translation, message-passing access, and memory management to deal with copies of data.
7. Multiprocessors provide the most efficient support for message-passing applications because messages are passed by passing pointers as in uniprocessors. Multicomputers require moving data.
8. A multiprocessor is inherently general-purpose, since any collection of small to large

and sequential-to-parallel programs can be operated on at any time. A multicomputer operates best on a very large, parallel program which is run to completion. If any of the nodes lack a facility that must be obtained in other nodes, bottlenecks can occur when accessing other nodes.

Because of the general-purposeness, scalable multiprocessors can be applied to real time, command and control, commercial transaction processing and database management. Multicomputers are not general-purpose in terms of either applications or job size mix.

Latency Inherent with Performance: There's No Free Lunch
Each computer represents a trade-off to deal with the increased latency inherent in building a large computer requiring high bandwidth. In the case of multiprocessors, data is in a shared memory that is delayed by switching (dance hall) or in another distributed processor-memory pair (boudoir). Similarly, in a multicomputer, explicit messages must be sent to another computer. The alternatives represent trade-offs among such issues as how to and where to deal with latency, the degree of locality, and the degree of problem granularity. These architectural alternatives represent different beliefs about application locality:

1. SIMDs: Put processing elements with memory, do operations fast, allocate data to minimize communication with other nodes, send data when required and wait when parts of the computation need to share data. (CM1 ... CM2) Thinking Machines abandoned SIMD since only one very-large-scale parallel job could be run at a given time, making it cost-ineffective and non-general-purpose. Also, SIMDs have negligible scalar perfor-

mance, making them useless for anything but massively parallel, coarse-granularity applications.

2. Multivector processor supercomputers: Use vector processors to move more data (a vector) in one instruction, overlap instructions, and join operations of several instructions together in a single pipeline (chaining). The vector registers hide the latency that comes with bandwidth. Employ programmed controlled buffer memories to further cache data (Crays, Fujitsu, Hitachi, NEC)

3. Distributed multiprocessors Caching, pre-fetching, and post-storing are used to hide latency. Programs and data migrate to a processor-memory node on demand. Hardware automatically replicates data in local nodes using caches and maintains memory coherence. (KSR, DASH, T*, Alewife)

4. Multicomputers: Couple processors and memories to form cost-effective computer nodes. Place the same program in all nodes and allocate data across all computers to minimize moving data. When data movement is required in nonperfectly parallel programs, the compiler generates messages to transfer data to other nodes. Build mechanisms to broadcast data and recombine results (TMC and Intel)

5. Multistream (or multithreaded), multiprocessors: Provide a constant, but long latency path between physical processors and memory. Build multi-instruction stream processors whereby one physical processor acts as many separate processors. Pre-fetch and post-store data to cover the long, constant latency. This processor can be used in all the preceding computers (Tera, T*, Alewife)

The Species
The specific distributed multicomputers of Figure 3 are seg-

mented by interconnection bandwidth. LAN-connected workstations have the lowest bandwidth, but in the future could provide a significant amount of computing power by utilizing various parallel computing environments such as Linda, Parasoft's Express, or the Parallel Virtual Machine (PVM). Since 1975 Tandem has been using clusters of computers for redundancy and increased capacity; DEC introduced VAX clusters in 1982. Seitz (Cal Tech) pioneered the multicomputer for supercomputing, and Intel has built three generations based on much of this work. Many companies offer multicomputers using Transputers or Intel i860 processors for practical and pedagogical use.
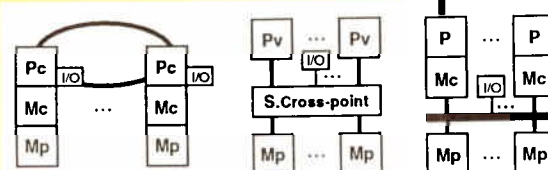
Two alternative interconnect switches are used for nonscalable multiprocessors. A single bus, Figure 5 is the simplest way to build a multiple microprocessor or "multi" [2]. With the evolution of microprocessors to support "multis" any computer from the simplest PC can easily become a multiprocessor. Multis are limited by the capability of the bus that is formed on printed wiring, and hence is not capable of significant size or generation scalability. In the future, the bus will be replaced by a ring, providing the essential features of a bus, but scales with size and generation (i.e., clock speed since a chip only drives a neighbor), as shown in Figure 6. The bandwidth for a "ring multi" increases as the number of nodes increase (using multiple tokens) at the expense of increased latency (to be hidden by a cache).

Mainframes and supers use cross-points and multistage networks to interconnect processors and memories (Figure 7). Since up to three memory accesses may be required to execute a statement such as A = B + C in order to compute

one flop, it is easy to see why the switch connecting processors and memory limits a computer. As a switch is increased in size and bandwidth, latency and grain increase. Worse yet, scalar performance decreases.

Three scalable multiprocessors were built as research efforts beginning with CMU's Cm*. The single address space was used to eliminate message-passing and to simplify the naming and allocation of memory. All required programs to be located in particular nodes and suffered from the same flaw on which multicomputers are predicated. Stanford's DASH binds programs statically to nodes, but uses caching to reduce latency when a remote node requires data. This reduces or eliminates the need for perfect data-to-node assignment. Nevertheless, over long-term use, it is imperative to move data permanently to the computing node requiring the data.

The KSR smP to be described solves the data-to-node assignment problem inherent in a distributed memory mP by providing hardware that controls the automatic migration of memory to any that may need it. KSR's breakthrough occurred by conceptually eliminating physical addresses and making the memory into a cache so that information could be automatically moved to a processor when needed. ∎

providing a quadrupling of performance each three years still appears to be possible for the next few years (Table 3). The quadrupling has its basis in Moore's Law stating that semiconductor density would quadruple every three years. This explains memory-chip-size-evolution. Memory size can grow proportionally with processor performance, even though the memory bandwidth is not keeping up. Since clock speed only improves at 25% per year (a doubling in three years), the additional speed must come from architectural features (e.g., superscalar or wider words, larger cache memories, and vector processing).

The leading edge microprocessors described at the 1992 International Solid State Circuits Conference included: a microprocessor based on Digital's Alpha architecture with a 150- or 200MHz clock rate; and the Fujitsu 108 (64)|216 (32-bit) Mflop Vector Processor chip that works with Sparc chips. Using the Fujitsu chip with a microprocessor would provide the best performance for traditional supercomputer-oriented problems. Perhaps the most important improvement to enhance massive parallelism is the 64-bit address enabling a computer to have a large global address space. With 64-bit addresses and substantially faster networks, some of the limitations of message-passing multicomputers can be overcome.

In 1995, $20,000 distributed computing node microprocessors with peak speeds of 400 to 800 Mflops can provide 20,000 to 40,000 flops/$. For example, such chips are a factor of 12 to 25 times faster than the vector processor chips used in the CM5 and would be 4.5 to 9 times most cost-effective. Both ECL and GaAs are unlikely runners in the teraflop race since CMOS improves so constantly in speed and density. Given the need for large on-chip cache memories and the additional time and cost penalties for external caches, it is likely that CMOS will be the semi-conductor technology for scalable computers.

Not Just Another Workstation: A Proposal for Having Both High-Performance Workstations and Massive Parallelism

Workstations are the purest and simplest computer structure able to exploit microprocessors since they contain little more than a processor, memory, CRT, network connection, and i/o logic. Furthermore, their inherent CRTs solve a significant part of the i/o problem. A given workstation or server node (usually just a workstation without a CRT, but with large memory and a large collection of disks) can also become a multicomputer.

Nielsen of Lawrence Livermore National Laboratory (LLNL) has outlined a strategy for transitioning to massively parallel computing [18]. LLNL has made the observation that is spends about three times as much on workstations that are only 15% utilized, as it does on supercomputers. By 1995, microprocessor-based workstations could reach a peak of 500Mflops, providing 25,000 flops per dollar or 10 times the projected cost-effectiveness of a super. This would mean that inherent in its spending, LLNL would have about 25 times more unused peak power in its workstations than it has in its central supercomputer or specialized massively parallel computer.

The difficult part of using workstations as a scalable multicomputer (smC) is the low-bandwidth communication links that limit their applicability to long-grained problems. Given that every workstation environment is likely to have far greater power than a central super, however, the result should clearly justify the effort. An IEEE standard, the Scalable Coherent Interface or SCI, is being implemented to interconnect computers as a single, shared-memory multiprocessor. SCI uses a ring, such as KSR, to interconnect the computers. A distributed directory tracks data as copies migrate to the appropriate computer node. Companies such as Convex are exploring the SCI for interconnecting HP's micros as an alternative and preferred mini-super that can also address the supercomputing market.

A cluster of workstations interconnected at speeds comparable to Thinking Machines's CM5, would be advantageous in terms of power, cost-effectiveness, and administration compared with LAN-connected workstations and supercomputers. Such a computer would have to be centralized in order to have low latency. Unlike traditional timeshared facilities, however, processors could be dedicated to individuals to provide guaranteed service. With the advent of HDTV, low-cost video can be distributed directly to the desktop, and as a byproduct users would have video conferencing.

smPs: Scalable Multiprocessors
The Kendall Square Research KSR 1. The Kendall Square Research KSR 1 is a size-and-generation-scalable, shared-memory multiprocessor computer. It is formed as a hierarchy of interconnected "ring multis." Scalability is achieved by connecting 32 processors to form a "ring multi" operating at oneGB/sec (128 million accesses per sec). Interconnection bandwidth within a ring scales linearly, since every ring slot may contain a transaction. Thus, a ring has roughly the capacity of a typical cross-point switch found in a supercomputer room that interconnects 8 to 16, 100MB/sec HIPPI channels. The KSR 1 uses a two-level hierarchy to interconnect 34 rings (1,088 processors), and is therefore massive. The ring design supports an arbitrary number of levels, permitting ultras to be built.

Each node is comprised of a primary cache, acting as a 32MB primary memory, and a 64-bit superscalar processor with roughly the same performance as an IBM RS6000 operating at the same clock-rate. The superscalar proces-

sors containing 64 floating-point and 32 fixed-point registers of 64 bits is designed for both scalar and vector operations. For example, 16 elements can be pre-fetched at one time. A processor also has a 0.5MB sub-cache supplying 20 million accesses per sec to the processor (computational efficiency of 0.5). A processor operates at 20MHz. and is fabricated in 1.2 micron CMOS. The processor, sans caches, contains 3.9 million transistors in 6 types of 12 custom chips. Three-quarters of each processor consists of the Search Engine responsible for migrating data to and from other nodes, for maintaining memory coherence throughout the system, using distributed directories, and ring control.

The KSR 1 is significant because it provides size- (including I/O) and generation-scalable smP in which every node is identical; an efficient environment for both arbitrary workloads (from transaction processing to timesharing and batch) and sequential to parallel processing through a large, hardware-supported address space with an unlimited number of processors; a strictly sequential consistent programming model; and dynamic management of memory through hardware migration and replication of data throughout the distributed, processor memory nodes, using its Allcache mechanism.

With sequential consistency, every processor returns the latest value of a written value, and results of an execution on multiple processors appear as some interleaving of operations of individual nodes when executed on a multithreaded machine. With Allcache, an address becomes a name and this name automatically migrates throughout the system and is associated with a processor in a cache-like fashion as needed. Copies of a given cell are made by the hardware and sent to other nodes to reduce access time. A processor can pre-fetch data into a local cache and post-store data for other cells. The hardware is designed to exploit spatial and tempo-

ral locality. For example, in the SPMD programming model, copies of the program move dynamically and are cached in each of the operating nodes' primary and processor caches. Data such as elements of a matrix move to the nodes as required simply by accessing the data, and the processor has instructions that pre-fetch data to the processor's registers. When a processor writes to an address, all cells are updated and memory coherence is maintained. Data movement occurs in sub-pages of 128 bytes (16 words) of its 16K pages.

Every known form of parallelism is supported via KSR's Mach-based operating system. Multiple users may run multiple sessions, comprising multiple applications, comprising multiple processes (each with independent address spaces), each of which may comprise multiple threads of control running simultaneously sharing a common address space. Message-passing is supported by pointer-passing in the shared memory to avoid data copying and enhance performance.

KSR also provides a commercial programming environment for transaction processing that accesses relational databases in parallel with unlimited scalability, as an alternative to multicomputers formed from multiprocessor mainframes. A 1K-node system provides almost two orders of magnitude more processing power, primary memory, I/O bandwidth, and mass storage capacity than a multiprocessor mainframe. For example, unlike the typical tera-candidates, a 1,088-node system can be configured with 15.3 terabytes of disk memory, providing 500 times the capacity of its main memory. The 32- and 320-node systems are projected to deliver over 1,000 and 10,000 transactions per sec, respectively, giving it over a hundred times the throughput of a multiprocessor mainframe.
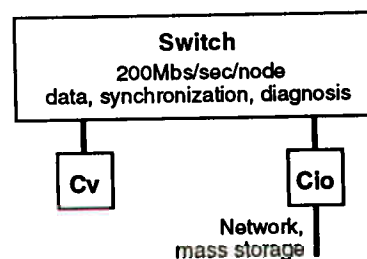
smCs: Scalable Multicomputers for "Commodity Supercomputing"

Multicomputer performance and

applicability are determined by the number of nodes and concurrent job streams, the node and system performance, I/O bandwidth, and the communication network bandwidth, delay, and overhead time. Table 1 gives the computational and workload parameters, but for a multicomputer operated as a SPMD, the communications network is quite likely the determinant for application performance.

Intel Paragon: A Homogeneous Multicomputer. This is shown in Figure 8. A given node consists of five i860 microprocessors: four carry out computation as a shared-memory multiprocessor operating at a peak of 300Mflops rate, and the fifth handles communication with the message-passing network. Each processor has a small cache, and the data-rate to primary memory is 50 million accesses per sec, supporting a computational intensity of 0.67 for highly select problems. The message-passing processor and the fast 2D mesh topology provide the very high, full-duplex data-rate among the nodes of 200MB/sec. The mesh provides primitives to support synchronization and broadcasting.

Paragon is formed as a collection of nodes controlled by the OSF1 (Mach) operating system with micro kernels that support message-passing among the nodes. Each node can be dynamically configured to be a service processor for general-purpose timesharing, or part of a parallel-processing partition, or an

Figure 8. Intel Paragon multicomputer

Switch
200Mbs/sec/node
data, synchronization, diagnosis

Cv   Cio

Network,
mass storage

I/O computer because it attaches to a particular device. A variety of programming models are provided, corresponding to the evolution toward a multiprocessor. Two basic forms of parallelism are supported: SPMD using a shared virtual memory and MIMD. With SPMD, a single, partition-wide, shared virtual memory space is created across a number of computers, using a layer of software. Memory consistency is maintained on a page basis. With MIMD a program is optimized to provide the highest performance within a node using vector processing, for example. Messages are explicitly passed among the nodes. Each node can have its own virtual memory.

CM5: A Multicomputer Designed to Operate as a Collection of SIMDs.
The CM5 is shown in Figure 9 consisting of 1 to 32 Sun server control computers, Cc, (for a 1K-node system), on which user programs run; the computational computers, Cv, with vector units; Sun-based I/O server nodes, Cio; and a switch to interconnect the elements. The system is divided into a number of independent partitions with at least 32 Cv's that are managed by one Cc. A given partitioning is likely to be static for a relatively long time (e.g., work shift to days). The Sun servers and I/O computers run variants of Sun O/S, providing a familiar user-operating environment together with all the networking, file systems, and

Figure 9. CM5 multicomputer

graphical user interfaces. Both the SPMD and message-passing programming models are supported. Each of the computational nodes, Cv, can send messages directly to Cio's, but other system calls must be processed in the Cc.

The computational nodes are Sparc micros that control four independent vector processors that operate independently on 8MB memories. A node is a message-passing multicomputer in which the Sparc moves data within the four 8MB memories. Memory data is accessed by the four vector units at 16Maccess per sec (Maps) each, providing memory bandwidth for a computational intensity of 0.5. Conceptually, the machine is treated as an evolution of the SIMD CM2 that had 2K floating-point processing elements connected by a message-passing hypercube. Thus, a 1K-node CM5 has 4K processing element, and message-passing among the 4 vector units and other nodes is controlled by the Sparc processors. The common program resides in each of the nodes. Note that using Fujitsu Vector Processing chips instead of the four CM5 vector chips would increase peak performance by a factor of 3.3, making the 1995 teraflop peak achievable at the expense of a well-balanced machine.

Computational intensity is the number of memory accesses per flop required for an operation(s) of a program. Thus depending on the computational intensity of the operations, speed will vary greatly. For example, the computational intensity of the expression $A = B + C$ is 3, since 3 accesses are required for every flop, giving a peak rate of 21Mflops from a peak of 128. A C90 provides 1.5Maps per 1Mflops, or a 16-processor system is capable of operating at 8Gflops.

The switch has three parts: diagnosis and reconfiguration; data message-passing; and control. The data network operates at 5MB/sec full duplex. A number of control messages are possible, and all proc-

essors use the network. Control network messages include broadcasting (e.g., sending a scalar or vector) to all nodes unless it abstains, results recombining (network carries out arithmetic and logical operations on data supplied by each node), and global signaling, synchronization for controlling parallel programs. The switch is wired into each cabinet that holds the 256 vector computers.

A Score of Multicomputers. Cray Research has a DARPA contract to supply a machine capable of peak teraflop operation by 1995, and a sustained teraflop by 1997 using DEC Alpha microprocessors. Convex has announced it is working on a massively parallel computer, using HP's microprocessor as its base. IBM has several multicomputer systems that it may productize based on RS6000 workstations. Japanese manufacturers are building multicomputers, for example, using a comparatively small number (100s) of fast computers (i.e., 1Gflop) interconnected via very high-speed networks in a small space.

A number of multicomputers have been built using the Inmos Transputer and Intel i860 (e.g., Transtech Parallel Systems). Mercury couples 32, 40MHz. i860s and rates the configuration at 2.5Gflops for signal processing, simulation, imaging, and seismic analysis. Meiko's 62-node multicomputer has a peak of 2.5Gflops and delivered 1.3Gflops for Linpeak, or approximately half its peak on a O(8500) matrix [8]. Parsytec GC consists of 64 16K nodes, delivering a peak of 400Gflops. The nCUBE 2 system has up to 8K nodes with up to 64MB per node.

Multicomputers are also built for specific tasks. IBM's Power Visualizer uses several i860s to do visualization transformations and rendering. AT&T DSP3 Parallel Processor provides up to 2.56Gflops, formed with 128 signal-processing nodes. The DSP3 is used for such tasks as signal- and image-processing, and
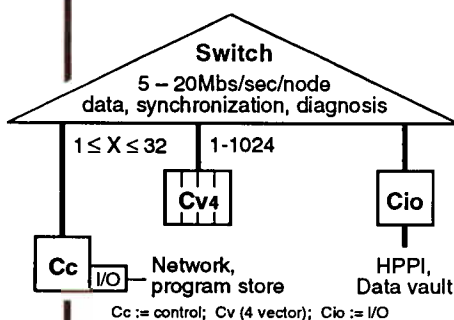
speech recognition. The DARPA-Intel iWarp™ developed with CMU is being used for a variety of signal- and image-processing applications that are typically connected to workstations. An iWarp node provides only 10Mflops (or 20Mflops for 32-bit precision) and 0.5 to 16MB of memory per node. Each node can communicate at up to 320MB per sec on 8 links.

The Teradata/NCR/AT&T systems are used for database retrieval and transaction-processing in a commercial environment. The system is connected by a tree-structured switch, and the hundreds of Intel 486 leaf nodes processors handle applications, communications, or disk access. A system can process over 1,000 transactions per sec working with a single database, or roughly four times the performance of a multiprocessor main-frame.

Programming Environments to Support Parallelism

Although the spectacular increases in performance derived from microprocessors are noteworthy, perhaps the greatest breakthrough has come in software environments such as Linda, the Parallel Virtual Machine (PVM), and Parasoft's Express that permit users to structure and control a collection of processes to operate in parallel on independent computers using message passing. Of course, user interface software, debuggers, performance-monitoring, and many other tools are part of these basic parallel environments.

Several programming models and environments are used to control parallelism. For multiprocessors, small degrees of parallelism are supported through such mechanisms as multitasking and Unix™ pipes in an explicit or direct user control fashion. Linda extends this model to manage the creation and distribution of independent processes for parallel execution in a shared address space. Medium (10 to 100) and high degrees of parallelism (1,000) for a single job can be

carried out in either an explicit message-passing or implicit fashion. The most straightforward implicit method is the SPMD model for hosting Fortran across a number of computers. A Fortran 90 translator should enable multiple workstations to be used in parallel on a single program in a language evolutionary fashion. Furthermore, a program written in this fashion can be used equally effectively across a number of different environments from supercomputers to workstation networks. Alternatively, a new language, having more inherent parallelism, such as dataflow may evolve. Fortran will adopt it.

Research Computers

Much of university computer architecture research is aimed at scalable, shared-memory multiprocessors (e.g., [20]) and supported by DARPA. In 1991, MITI sponsored the first conference on shared-memory multiprocessors in Tokyo, to increase international understanding. It brought together research results from 10 universities (eight U.S, two Japanese), and four industrial labs (three U.S., one Japanese). This work includes, directory schemes to efficiently "track" cached data as it is moved among the distributed processor-memory pairs, performance analysis, interconnection schemes, multithreaded processors, and compilers.

Researchers at the University of California, Berkeley are using a 64-node CM5 to explore various programming models and languages including dataflow. Early work includes a library to allow the computer to simulate a shared memory multiprocessor. An equally important part of Berkeley's research is the Sequoia 2000 project being done in collaboration with NASA and DEC that focuses on real-time data acquisition of 2 terabytes of data per day, secondary and tertiary memories, and very large databases requiring multiple accesses.

Seitz at Cal Tech, developed the

first multicomputer (interconnected via a hypercube network) and went on to develop high-bandwidth grid networks that use wormhole routing. The basic switch technology is being used in a variety of multiprocessor and multicomputers including Intel and Cray.

The CEDAR project at the University of Illinois is in the completion phase, and scores of papers describe the design, measurements, and the problem of compiling for scalable multiprocessors. Unfortunately, CEDAR was built on the now defunct Alliant Multiprocessor.

MIT has continued researching multiple machine designs. The Monsoon dataflow computer became operational with 16, 5Mflop nodes and demonstrates scalability and implicit parallelism using a dataflow language. The next dataflow processor design, T* is multithreaded to absorb network latency. The J-machine is a simple multicomputer designed for message-passing and low-overhead system calls. The J-machine, like the original RISC designs, places hardware functions in software to simplify the processor design. A J-machine, with sufficient software, carries out message-passing functions to enable shared-memory multiprocessing. Alewife, like Stanford's DASH is a distributed multiple multithreaded processor that is interconnected via a grid switch. Additional efforts are aimed at switches and packaging, including a 3D interconnection scheme.

Rice University continues to lead compiler research in universities and was responsible for the HPF Forum. HPF is a successor to Fortran D (data parallelism) that was initially posited for all high-performance machines (SIMDs, vector multiprocessors, and multicomputers). The challenge in multicomputers is initial data allocation, control of memory copies, and avoiding latency and overhead when passing messages.

Stanford's DASH is a scalable multiprocessor with up to 64 pro-

Role of Computer and Computational Science

A recent panel of computer and computational scientists described their reservations about the progress in parallel computing, expressing concerns about training and people interested in computational science, machine availability, and lack of standards caused by too many programming models [19]. They compared progress to the difficulty in learning to use vector processors. A recent study by the IEEE Technical Committee on Supercomputing (IEEE 1992) showed that out of approximately 8,000 users of the NSF Supercomputer centers, less than 100 computer science and about 200 electrical engineering users used a negligible amount of the resources. With today's parallel computers that are an artifact of massive federal funding, however, computer science has been slowly attracted to helping understand fundamentals.

In 1987, as assistant director of computing at NSF, I urged the computer science community to become involved in parallelism by using and understanding the plethora of computers that can be applied to computational science [2]. This would entail understanding applications, including solving problems using new parallel structures, writing texts, training students, and carrying out research. Today, much computer science research is devoted to some form of parallelism. Making significant contributions in parallelism requires understanding and solving problems that are usually numerically intensive. Numerical analysis is not part of the computer science core curriculum. In a similar fashion, the results of supercomputing often require visualization (also not part of the curriculum). Visualization also changes the nature of I/O and mass storage. Now, I suggest the following:

1. Collaborating with scientists and engineers on real problems using real computers. This enhances the training of computational scientists who understand and enrich computer science.
2. Training and understanding using traditional, uniprocessor supercomputers and shared-memory multiprocessors that provide fine granularity. If code runs poorly on a super or a shared-memory multiprocessor, it is certain to run poorly on a distributed multicomputer.
3. Installing, teaching, and using environments composed of existing workstations that have very long granularity. These can and must be dealt with using programming environments such as Linda or PVM.

Attaching SIMDs and multicomputers to workstations for specialized problems.

4. Progressing to problems and algorithms that can tolerate the latencies inherent in multicomputers.
5. Designing benchmarks and workloads typifying new programs and computers to enhance understanding. Collaborating on computers that are being designed, and making them run well is much more important than producing any more computers.

We must thoroughly understand the machines we have by using and measuring them on a range of real problems. The goal should be to look at a problem / program / algorithm and know *a priori* how an application will run, based on the computer / compiler as measured by its various parameters. Making a parallel application run effectively is an *ad hoc* art. ∎

cessors arranged in a grid of 4 × 4 nodes. Each node consists of a four-processor Silicon Graphics multiprocessor that is roughly equivalent to a uniprocessor with an attached vector unit. DASH demonstrated linear speedups for a wide range of algorithms, and is used for compiler research. Some applications have reached over 100Mflops for 16 processors, which is about the speed a four-vector processor system of comparable speed achieves. Since the system is relatively slow, it is unclear which principles have applicability for competitive computers.

DARPA has funded Tera Computer to start-up. Tera, a second-generation HEP, is to have 256 128-instruction stream processors or 32K processors and operate in 1995. With a multiple instruction stream or multithreaded processor, any time a processor has to wait for a memory access, the next instruction stream is started. Each processor is built using very fast gate arrays (e.g., GaAs) to operate at 400MHz. The expected latency to memory is between 40 and 100 ticks, but since each processor can issue multiple requests, a single physical processor appears to support 16 threads (or virtual processors). Thus, a processor appears to have access to a constant, zero latency memory. Since a processor is time-shared, it is comparatively slow and likely to be unusable for scalar tasks, and is hardly a general-purpose computer according to Smith's definition [21]. The physical processors connect to 512 memory units, 256 I/O cache units (i.e., slower memories used for buffering I/O), and I/O processors through a 4K-node interconnection network. In order to avoid the "dance hall" label, the network has four times more nodes than are required by the components. Tera's approach has been to design a computer that supports a parallelizing compiler.

Summary and Conclusions

In 1989 I projected that supercom-

puters would reach a teraflop by 1995 using either a SIMD or multicomputer approach, but neglected to mention the price. By mid-1992, scalable multicomputers have improved by a factor of 5 to 10 to reach 100 ± 20 Gflops at a supercomputer price level, and the SIMD approach was dropped. Scalable multicomputers also break through the $30 million supercomputer barrier to create a teraflop ultracomputer ($50 to 400 million), and are not recommended buys. In 1995 semiconductor gains should increase performance by a factor of four and competition should reduce the price a factor of two to supercomputer levels as projected in Figure 1. Given the number of applications and state of training, waiting for the teraflop at the supercomputer price level is recommended.

Multiprocessors have been the mainline of computing because they are general-purpose and can handle a variety of applications from transaction-processing to time-sharing, that are highly sequential to highly parallel. KSR demonstrated that massive, scalable

Government Policy: Why We Don't Need "State" Computer Architectures

In January 1992, the President signed a law authorizing the spending of $1 billion for various agencies to comply with the HPCC Act. The law provides for building a National Research and Education Network (NREN), as well as work on parallel computing, algorithms, and computer science education. The HPCC Report (OSTP, 1992) outlines the role of the various agencies (DARPA, DOC, DOE, EPA, NIH/NLM, and NSF) in computing systems, software technology and algorithms, the network, and basic research and human resources. The report outlines a variety of grand challenges in science and engineering, ranging from weather and climate prediction and global change, to astronomy, semiconductors, superconductors, speech, and vision. According to the report's budget, DARPA has two-thirds of the budget, or over $100 million in 1992 for high-performance computing systems. Other agencies have the grand challenge problems. Undoubtedly, the most important aspect of the program will be training and the network. So far, the architectures and companies resulting from this massive funding have been less than spectacular, which confirms my opinion that DARPA should not directly fund the development of computers at companies.

DARPA has a long and successful record of sponsoring university research that creates companies and products such as MIPS, Sun, and Sparc in cases where no products or technology existed. It fostered AI, graphics, operating systems, packet switching, speech understanding, time-sharing, VLSI design, and workstations at universities. Supercomputing, including using a massive (>1000) number of processors is a commercial area that has been developed by industry and does not require the selection or support of particular architectures or companies. DARPA's role in the development of massive parallelism can be terminated because it has been picked up by industry. Almost a dozen companies are building multicomputers that compete with DARPA's incestuous product divisions (Cray, Intel, Tera, and Thinking Machines). The situation of funding the design and purchase of a computer is not a healthy one.

I know of no successful products developed by funding company product development, including the vast array of military computers. DARPA funded Burroughs to build the unsuccessful Illiac IV in the late 1960s, a 64-processor SIMD. ARPA funded BBN to provide the first switching computers for ARPAnet. BBN was successful for almost a decade during which it had a technology monopoly as the government paid for product development and bought and tested its products. In 1992, BBN is a minor supplier in a flourishing communications market. Similarly, BBN's government-funded computer development that was initiated by DARPA folded in 1991. The several hundred million dollars of funds that went into a couple of massively parallel computers this last decade could have been used to provide substantially more computing power to real users. By way of contrast, NSF spends about $60 million annually to support four supercomputing centers. Funding to create a monopoly company only inhibits the development of technology and products at other companies. Product development contracts and purchase orders have not created lasting companies, and are not likely to in the future. Companies funded in an incestuous fashion simply cannot stand up to a "real" market, and will not be able to compete internationally. A large fraction of the market is reduced or eliminated when government funds and buys its own designs, closing the early adopter and innovator government markets to privately financed computer companies. Furthermore, once started, DARPA-funded companies require continued funding to remain healthy . . . just as the defense contractors that are being downsized.

distributed, shared-memory multiprocessors (smPs) are feasible. Multicomputers from the score of companies combining computers will evolve to multiprocessors just to reduce overhead in simulating a single-memory address space, memory access, and supporting efficient multiprogramming. Next-generation multicomputers are likely to resemble BBN's distrib-uted memory computers as they evolve to become multiprocessors.

Important gains in parallelism have come from software environments that allow networks of computers to be applied to a single job. Thus every laboratory containing fast workstations has or will have its own supercomputer for highly parallel applications. The rapid increase in microprocessor power ensures that the workstation will perform at near super speed for sequential applications. LAN environments can provide significant supercomputing for highly parallel applications by 1995. It is critical for companies to provide fast, low-overhead switches that will allow users to configure multicomputers composed of less than 100 high-

By taking over computer design through funding and purchasing, users do not benchmark or understand the machines they are given or forced to use. At a time when gigabucks for teraflops may induce brain damage, it is critical to consider all the factors of an architecture and especially the mean time before answers. In the future, government laboratories are likely to be measured on their ability to replicate and transfer results, and having programs that run well across a variety of machines is more important than exploiting the latest fad. When government support for the HPCC program ends, it is the market rather than a bureaucrat's dream of an architecture or industry, that determines economics.

At the Sid Fernbach Memorial Symposium, February 1992, speakers reiterated the policy that Fernbach used, as head of computation at Lawrence Livermore National Laboratory, to help supercomputing come into existence: be a knowledgeable, demanding, tolerant, and helpful customer. Department of Energy laboratories purchased, not funded the development of early computers by providing needs specifications.

Following is a suggested policy to support development of high-performance computing:

1. The concept of the ultracomputer is so artificial and deleterious to the computer industry that buying a single ultra should be discouraged. An affordable teraflop will come by 1995. Let evolution work to produce better computers that are balanced and usable, not aimed at a single, peak number.
2. Support users to purchase machines that can be justified for specific programs at various agencies and organizations that have "grand challenge" problems. Contracts would be open bid, and benchmarks that characterize the user workload would be required. The reestablishment of benchmarking would cause reality to replace hope as a buying criteria.

Allow universities to choose the computers they buy. Don't control the purchasing or the design of computers from Washington (funding agency, Congress). Given the specialized nature and high cost of a supercomputer or ultracomputer, users (e.g., weapons designers) who can justify them from tool and experiment budgets should simply buy them.

3. Encourage collaboration. Any company should be free to work with a laboratory project to produce technology, prototype, or product. Fund university projects (not their codevelopers) where a codeveloping company is capable of or likely to be able to take the product to market.

Encourage laboratories to obtain clustered computers based on workstation nodes providing more than an order of magnitude more peak power than supers. Such machines would provide the same power as scalable multicomputers, but in addition, dedicated power for visualization and video conferencing.
4. Do not fund computer development *per se*. Industry has always been able to fund good ideas. There is really no effective way to select the "right" winner from Washington. Once started, funding becomes an ongoing government responsibility and right for startups: get the money and do it. In the case of large companies: if the technology is worth funding, fund it. A company only takes government money to build a computer if the project is not worthwhile, and they are supporting its staff.
5. Encourage the use of computers in universities vs. designing more computers by people who have never used or built a computer. The world is drowning in computers that absorb programmers trying to realize peak performance.
6. Eliminate funding by congressional-directed centers even though this might work. Chose centers based on competence.
7. In the very unlikely event that no one is building the appropriate computer and a special one must be funded for clear military need, build a few prototypes (≤2), open the process to all bidders without the usual military procurement hassles. ∎

performance workstations, because these are likely to provide the most cost-effective and useful computing environments.

A petaflop (10^{15} floating-point operations per sec) at less than \$500 million level is likely to be beyond 2001. By 2001, semiconductor gains provide an increase factor of 16 over 1995 computers. Better packaging and lower price margins through competition could provide another increase factor of two or three. The extra increase factor of 20 for a petaflop is unclear. Based on today's results and rationales, a petaflop before its time is as bad an investment as the teraflop before its time. Evolution and market forces are just fine . . . if we will just let them work.

Acknowledgments

The author would like to thank Peter Denning and other reviewers for their helpful suggestions in the editing process. These individuals provided performance data and helpful insight: David Culler (University of California, Berkeley), Charles Grassl (Cray Research), Justin Rattner (Intel), Ron Jonas and Ruby Lee (HP), Chani Pangali (Kendall Square Research), Frank MacMahon (Lawrence Livermore National Laboratories), John Mashey (MIPS), Tadashi Watanabe (NEC), Jack Dongarra (Oak Ridge National Laboratory and University of Tenn.), Mike Humphrey (Silicon Graphics Inc.), Burton Smith (Tera Computer), David Douglas and John Mucci (Thinking Machines Corp.), and Jack Worlton (Worlton & Assoc.) **C**

References

1. Bailey, D.H. Twelve ways to fool the masses when giving performance result on parallel computers. *Supercomput. Rev.* (Aug. 1991), 54–55.
2. Bell, G. The future of high performance computers in science and engineering. *Commun. ACM 32,* 9 (Sept. 1989), 1091–1101.
3. Bell, G. 11 rules of supercomputer design. University Video Communications Videotape, vol. III, Stanford Calif., 1989.
4. Bell, G. Three decades of multi-processors. CMU Computer Science: 25th Anniversary Commemorative R. Rashid, Ed. ACM Press, Addison-Wesley, Reading, Mass., 1991, pp. 3–27.
5. Berry, M., Cybenko, G., and Larson, J. Scientific benchmark characterizations. *Parallel Comput. 17,* (1991), 1173–1194.
6. Denning, P.J. Working sets past and present. *IEEE Trans. Softw. Eng. SE-6* (Jan. 1980), 64–84.
7. Denning, P.J. and Tichy, W. Highly parallel computation. *Science 250,* xx (Nov. 30, 1990), 1217–1222.
8. Dongarra, J.J. Performance of various computers using standard linear equation software. University of Tennesse and Oak Ridge National Laboratory, CS-89-85, Jan. 13, 1992.
9. Dongarra, J.J., Karp, Miura, K. and Simon, H.D. Gordon Bell Prize Lectures. In *Proceedings Supercomputing 91* (1991), pp 328–337.
10. Gustafson, J.L., Montry, G.R. and Benner, R.E. Development of parallel methods for a 1024 processor hypercube. *SIAM J. Sci. Stat. Comput. 9,* 4 (July 1988), 609–638.
11. Hennessy, J.L. and Patterson, D.A. *Computer Architecture: A Quantitative Approach.* Morgan Kaufman, San Mateo, Calif., 1990.
12. Hill, M.D. What is scalability? *Computer Architecture News 18,* 4 (Dec. 1990), 18–21.
13. Hillis, D. and Steele, G. The CM5, University Video Communications, Two videotapes, vol. IV, Stanford Calif., 1992.
14. Hockney R.W. and Jesshope, C.R. *Parallel Computers 2,* Adam Hilger, Bristol, 1988.
15. IEEE Computer Society Technical Committee on Supercomputing Applications. NSF Supercomputer Centers Study, Feb. 1992.
16. Li, K. and Schafer, R. A hypercube shared virtual memory system. *1989 International Conference on Parallel Systems,*
17. Office of Science and Technology Policy (OSTP). Grand challenges 1993: High performance computing and communications, A report by the Committee on Physical, Mathematical, and Engineering Sciences of the Federal Coordinating Council for Science, Engineering, and Technology in the Office of Science and Technology Policy, National Science Foundation, 1992.
18. Nielsen, D.E. A stategy for smoothly transitioning to massively parallel computing. *Energy and Tech. Rev.* (Nov. 1991), 20–31.
19. Pancake, C.M. Software support for parallel computing: Where are we headed? *Commun. ACM 34.,* 11 (Nov. 1991), 52–66.
20. Scott, S.L. A cache coherence mechanism for scalable, shared-memory multiprocessors. In *Proceedings of the International Symposium of Shared Memory Multiprocessing Information Processing Society of Japan,* (Tokyo, Apr., 1991) 49–59.
21. Smith, B. The end of architecture. *Comput. Architecture News 18.,* 4 (Dec. 1990), 10–17.
22. Watanabe, The NEC SX3 Supercomputer, University Video Communications, Videotape, Stanford Calif., 1992.
23. Worlton, J. A critique of "massively" parallel computing. Worlton & Associates Tech. Rep. 41, Salt Lake City Utah, May 1992.
24. Worlton, J. The MPP Bandwagon. *Supercomput. Rev.,* To be published.

CR Categories and Subject Descriptors: B.0 [**Hardware**]: General; C.0 [**Computer Systems Organization**]: General—*instruction set design (e.g., RISC, CISC);* J.1 [**Computer Applications**]: Administrative Data Processing—*government, system architectures*

General Terms: Design, Management

Additional Key Words and Phrases: Government policy, parallel processing, scientific programming

About the Author:
GORDON BELL is a computer industry consultant. He has been vice president of R&D at Digital Equipment Corp., a founder of various companies, and was the first assistant director of the NSF Computing Directorate. From 1966 to 1972 he was a professor of computer science at Carnegie Mellon University. In 1991 Bell won the National Medal of Technology and in 1992 was awarded the IEEE's Von Neumann Medal. **Author's Present Address:** 450 Old Oak Court, Los Altos, CA 94022

TRADING MIPS AND MEMORY FOR

As part of its mission to profile the people and economy of the U.S., the Census Bureau collects industry and occupation data for individuals in the labor force. For the 1990 Decennial Census, each of an estimated 22 million natural language responses to questions on the census long form had to be classified into one of 232 industry categories and 504 occupation categories. If done fully by hand the cost of this task would be on the order of $15 million.

Knowledge Engineering

Robert H. Creecy
Brij M. Masand
Stephen J. Smith
David L. Waltz