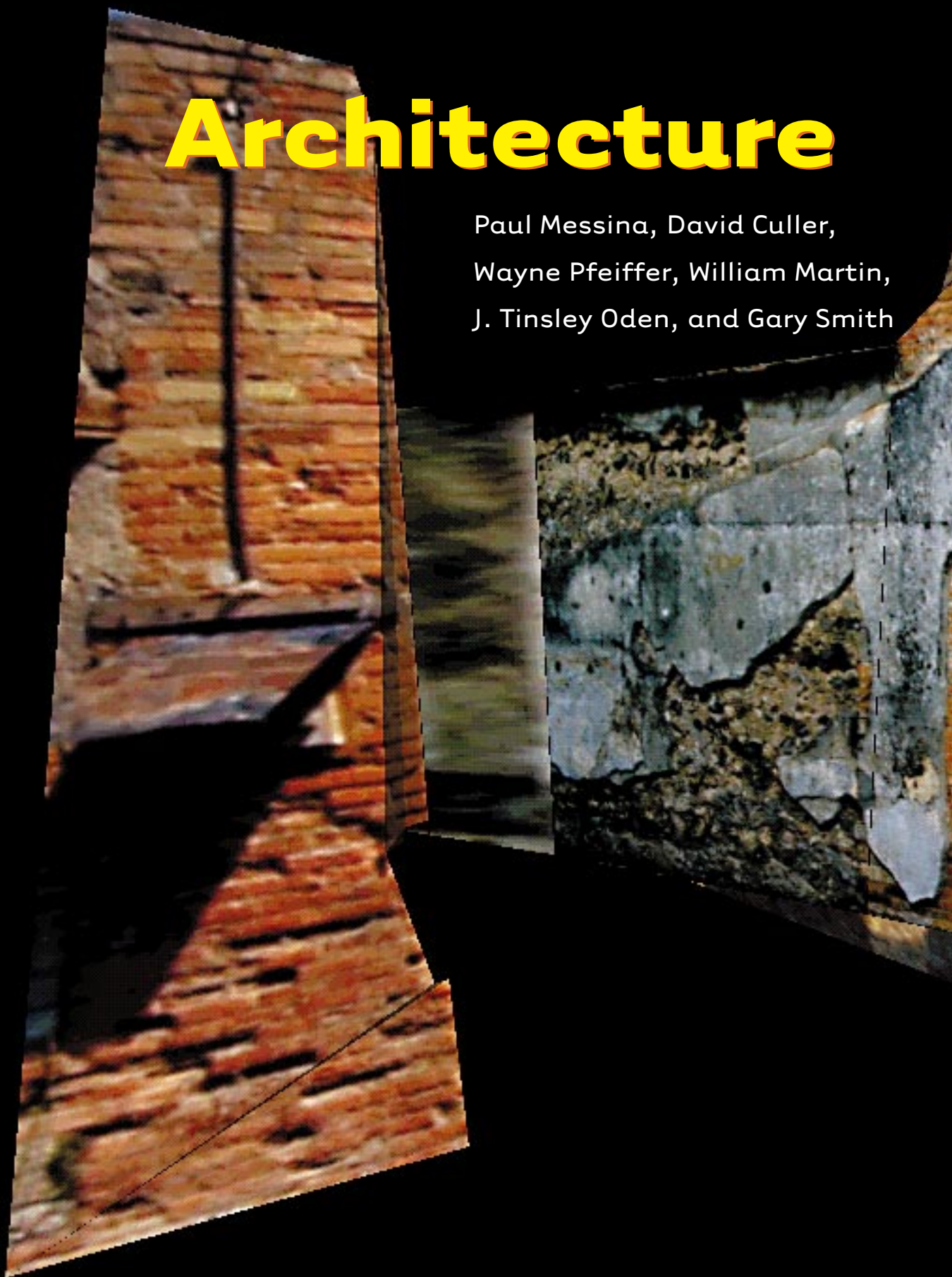
Architecture

Paul Messina, David Culler,
Wayne Pfeiffer, William Martin,
J. Tinsley Oden, and Gary Smith



Uniting commodity processors by the thousands, emerging parallel architectures promise solutions to complex calculations—at least until the laws of physics impose a fundamental barrier to human innovation.

To appreciate the advance in scientific computing power in recent years, consider the peak performance of the fastest computers used in science and engineering increased by a factor of 500 during the 1990s—from a few gigaflops to more than a teraflops (10^{12} floating-point operations per second), while the performance of the fastest available single processor increased by a factor of only about 15. The current world speed record of 1.338Tflops was set earlier this year by an Intel system at Sandia National Laboratories in New Mexico with 9,152 Pentium Pro processors [4]. The demand for computing power is not just for processing speed but for memory size and speed and data input and output rates as well. As the volume of sensor and simulation output data increases, large archival storage systems with rapid data retrieval play an increasingly important role (see R. Moore et al.'s article in this section).

Realistic 3D model reconstruction of Pompeii's Macellum, or market building. (Courtesy, John J. Dobins, Kirk Martini, and John Unsworth, University of Virginia.)

Promise and Limits of Amdahl's Law and Moore's Law

Amdahl's Law says a program can run no faster than the time required to execute its sequential sections. The Law is cast in the form of an equation that accounts for the time spent in the parallel (or vectorizable) and sequential parts of the code. The validity of this concept is easy to grasp. If a program is 80% parallelizable and 20% must be executed sequentially, the maximum reduction in execution time is by a factor of five (if the parallelizable code were run in zero time). Amdahl's Law was thought to provide a practical limit to the number of processors that could be used profitably in a parallel computer system, estimated at around 100 processors. However, this analysis did not account for the fact that for many scientific applications, as the size of the problem being tackled increases, the sequential fraction of the work tends to decrease. Since the speed and memory of the processors in a parallel system enable a user to tackle bigger problems, Amdahl's Law does not apply in these applications.

Moore's Law, coined by Intel founder Gordon Moore, predicts that microprocessor performance will double approximately every 18 months—a calculation that has been true for 30 years but is expected to continue for only a few more years. Moore himself frequently warns that as physical limits are reached in the feature sizes and in the manufacturing techniques used for semiconductor components, it is not clear how to make faster components with today's dominant technology—CMOS. The current estimate is that these limits will be reached within 10 years.

The dramatic increase in the performance delivered to applications follows from a combination of advances, each making the system more scalable [5]:

- The microprocessor building block became much faster (especially for floating-point operations) through greater internal parallelism, use of huge (1–4MB) multilevel caches, and faster clock speeds.
- Schemes were devised for effectively connecting larger numbers of processors and memories.
- Computational researchers learned to use large numbers of processors and deep memory hierarchies more effectively.
- System software and tools improved through experience and the leveraging of mainstream advances.

Systems with hundreds of processors were built in the 1980s using commercial or custom microprocessors, but these processors were much slower than the fastest available at the time (in vector supercomputers). Since then, microprocessor performance has advanced at a tremendous rate, so today the processor module that can be replicated easily in large numbers is also the fastest available technology. It is not the processor alone being replicated but a sophisticated, high-performance memory system as well.

Scalable parallel processors (SPPs) are systems that can be configured with hundreds or even thousands of state-of-the-art processors. The challenge for SPPs is to increase not so much the number of processors but the logical and physical interconnections, allowing a large number to be used effectively as one computing system. In the late 1980s, key concerns included whether Amdahl's Law would limit to 100 or so the number of processors that can be used efficiently (see the sidebar Promise and Limits of Amdahl's Law and Moore's Law). There were also lively debates about single instruction, multiple data (SIMD) vs. multiple instruction, multiple data (MIMD) architectures and about interconnection schemes and topologies (see the sidebar SIMD vs. MIMD vs. VLIW). In 1989, a 64Kb-processor SIMD Thinking Machines CM-2 system (with 2K floating-point units) outstripped the fastest vector systems for certain computations (such as those in hydrodynamics) [12]. However, the range of applications that could run efficiently on the CM-2 was limited by its SIMD architecture. It was not until the Intel Touchstone Delta System was installed at the California Institute of Technology in 1991 that an MIMD SPP substantially exceeded the performance of the largest vector systems in a variety of applications [7]. This advance was a turning point in the computing community's perception of the value of highly parallel computers and in architecture directions that would be taken over the next several years.

SPP architectures gained new features, such as very long instruction word (VLIW) and integration of the network onto the chip itself. Several supercomputer vendors used custom processors or coprocessors (including Cydrome, Kendall Square, MasPar, Multiflow, nCube, and Thinking Machines), as well as custom hardware and protocols, for interconnecting processors and memory. However, these dedicated efforts were soon challenged by the pace of advances in the broader technical computing industry. The first teraflops record, in 1996, characterizes many of the decade's dramatic changes. The Intel machine has a highly parallel architecture, and its processors and memory chips are mass-produced and off-the-shelf, not specially designed components manufactured with

The challenge for designers of SPP systems is to keep these complex processors fed with data so they can perform close to their potential.

exotic technologies. On the other hand, the system's processors and memory are connected via custom proprietary interfaces and protocols.

Highly Parallel Designs

The fastest computers achieve their speed through highly parallel designs involving hundreds to thousands of processors, as well as substantial parallelism within the processors. The majority use off-the-shelf microprocessors and memory components to construct the basic node, then replicate these nodes in large numbers. The number of processors has increased, the clock rate has increased, and so has the number of instructions issued and processed in each clock cycle (using multiple pipelined function units). The challenge for designers of SPP systems is to keep these complex processors fed with data so they can perform close to their potential; this challenge pervades interconnection hardware, memory system design, compilation techniques, and algorithm design.

The interconnection hardware for most SPP systems continues to be custom designed and proprietary to its vendors. The nodes are interconnected through a network of switches containing internal crossbars and use the same very large-scale integration technology as processor chips. The specific topology of the interconnect is less visible to the user than it was a few years ago, since all packets are directly routed in hardware and most of the communication time is associated with the node-to-network interface. SPP communication technology was recently transferred into more conventional settings, including commercial machinery, and a number of research projects have built high-performance systems from emerging system-area networks, such as Myrinet in the Berkeley Networks of Workstations (NOW) [3], or even commodity local-area networks, such as fast Ethernet in the Beowulf system (see cesdis.gsfc.nasa.gov/linux/beowulf/).

Memory access has always been a key concern for fast processors, including vector machines, but it has been a less critical issue for microprocessors, until recently. Microprocessors have become much faster, but memory speeds have not kept pace, so multiple levels of caches are used to bridge the gap. And because scientific applications tend to access large data sets, the difficulty of getting the required data to the multiple functional units in a continuous stream has increased. Offsetting these shortcomings, many current high-performance processors incorporate such additional features as instruction reordering and data prefetching. While some of these features put the onus on the compiler for generating code that takes advantage of them, in other cases, the processor hardware does the work. Parallel machines add to the problem of memory access, because the memory is physically distributed as needed. This distribution increases memory access time and makes it variable, with latency depending on the location of the data.

The design trend in SPP architectures is toward larger, more complex nodes. In early MIMD systems, nodes contained single processors. Some systems using relatively slow processors (the Intel Paragon and Meiko CS-2) put two or three processors on each node. More recently, nodes have been designed with eight processors (an option for the IBM SP series) and 16 processors (the Hewlett-Packard Exemplar X-Class) that physically share memory and packaging. Nodes have largely shifted from single to symmetric multiprocessors (SMPs).

This trend in high-performance computer architectures is one aspect of an evolving strategy of leveraging entire systems, not just processors, memory parts, and standard I/O interfaces (the peripheral component interface, or PCI, bus being a key example). For example, Hewlett-Packard, IBM, and Silicon Graphics all use the same processors in their largest-scale systems as they use in their workstations; Hewlett-Packard and Silicon Graphics even use their server-class machines as building blocks, with special interconnects linking the pieces.

Cache coherence has also become an important new architectural feature in SMPs and in SPPs, especially those offering distributed shared memory (DSM). DSM refers to memory that is physically distributed but can be globally addressed (hence shared)

by all processors in the system. Cache coherence is an issue in multiprocessor shared-memory systems, because when a processor changes the value of a variable, the new value is initially written only in that processor's cache, not in processor memory. And during some period of time, that memory location will have an obsolete value of the variable. Therefore, some mechanism has to be employed to ensure that other processors do not access the "stale" value in memory. That is, a way must be found to ensure that when a processor accesses a variable (from either its local cache or main memory), it uses the most recent value in memory. Not surprisingly, there are many ways to achieve cache coherence, and devising efficient cache coherence schemes is a major topic in computer architecture research today. Even where messaging is used for communication, the network interface has to be designed in concert with the node's cache coherence scheme.

Although DSMs have globally addressable memory, the time required to access a memory location varies depending on the location of the physical

Reflecting the Trends

From the late 1980s to the early 1990s, the San Diego Supercomputer Center (SDSC) relied primarily on multiprocessor vector systems, while Caltech emphasized distributed-memory MIMD systems with hundreds of processors. In the mid-1990s, Caltech and SDSC each acquired large Intel Paragon machines, which were the direct descendants of the Intel Touchstone Delta at Caltech and the predecessors of the Sandia Intel machine holding the current world speed record.

As NPACI's leading-edge site, SDSC now has a 272-processor Cray T3E system (MIMD, distributed memory) [9], a 128-processor IBM SP (MIMD, distributed memory) [11], and a 14-processor Cray T90 (vector MIMD, shared memory). Smaller Cray T3Es, IBM SPs, and Cray J90s are operated at NPACI sites at the University of Texas and at the University of Michigan. In the more experimental categories, Caltech has a 256-processor Hewlett-Packard Exemplar (MIMD, distributed shared memory) [1], the University of California, Berkeley, has a 100-processor Sun Microsystems NOW (MIMD, distributed memory) [3], and SDSC has a Tera Corp. multithreaded architecture, or MTA, machine (MIMD, shared memory, multithreaded) [2].

Cray T3E. The Silicon Graphics/Cray T3E is a distributed memory, MIMD system that scales to 1,024 nodes (several systems of maximum configuration are in use). The nodes consist of two Digital Equipment Alpha processors sharing memory connected through a 3D toroidal network. While memory is not shared globally, very low latency mechanisms are used for accessing the memory of remote nodes, and the bandwidth of the links in the interconnection network is quite high (600MB/sec). Many of the top 100 systems in the University of Mannheim Top 500 ranking of the fastest computers worldwide are large configurations of T3Es (for more on this ranking, go to www.top500.org).

Vector machines. The fastest processors today are still found in vector systems, with peak speeds of 8Gflops (as in the Hitachi S-3800, with a 2.0nsec cycle), 2Gflops (the NEC SX/4, with an 8.0nsec cycle), and 1.8Gflops (the Cray T90, with a 2.2nsec cycle). These speeds are impressive, but note that the fastest cycle speed (2.0nsec) is the same as that of the 500MHz version of Digital's Alpha 21164 processor, which has a peak speed of 1Gflops. The vector processors' higher peak speed is due more to internal parallelism than to cycle speed. Multiple-processor configurations are common in high-end vector systems, with as many as 32 in the NEC SX/4 and the Cray T90 and even more in the Fujitsu VPP family.

IBM SP family. The IBM SP family uses complete workstations or servers as nodes for parallel computers, rather than specially packaging the component chips, to track its mainline technology more rapidly. The nodes are connected through a network interface card to a multistage interconnection of crossbar switches, providing uniform access time to other nodes of a machine of a given size (as the number of nodes increases, more stages have to be added to the switches). Communication performance between nodes is limited primarily by the network interface.

Until recently, each node featured an RS/6000 processor and memory, plus peripherals, such as disks and network connections. Over time, the speed of the processors and the crossbar switch has increased, but each node was basically identical to an RS/6000 workstation. In 1996, IBM introduced a new type of node, one in which up to eight PowerPC processors share memory, so these nodes constitute a small shared memory multiprocessor. These processors are much slower for floating-point arithmetic than the RS/6000 RISC, but they will soon (within the next six months) be replaced by faster ones. One can mix nodes with single or multiple and new or older processors in the same chassis and connect them to the same crossbar switch. Nodes will also soon be available with larger numbers of much faster processors sharing memory within each

memory unit with respect to the processor accessing it. That is, DSMs have nonuniform memory access (NUMA). The magnitude of the effect of location on access time is governed by the properties and size of the network used to connect the memories and processors, as well as the details of the cache-coherence strategy for systems that have it. Therefore, algorithmic and compilation techniques for using caches effectively with a single processor are central to the effective use of SPPs.

Meanwhile, traditional high-speed vector processors also continue to evolve and are still used heavily but are no longer the systems on which the largest computations are carried out. Cycle speeds have not increased substantially in the past decade, but performance has been boosted by incorporating more pipelines and functional units into each vector processor. In addition, the number of processors in typical configurations has been increased from four to 16 or more. The fastest installed commercial vector system today is a Fujitsu-built machine with 167 vector processors at Japan's National Aerospace Lab-

node. These systems are generally configured with dedicated I/O nodes and compute partitions, so the network serves as a scalable backplane.

Sun NOW. The Berkeley NOW is an experimental system extending the complete-system-per-node approach to literally stack racks of commodity workstations and SMPs (or even PCs), interconnecting them through a generic high-speed network. It uses Sun UltraSparc workstations and SMPs. To eliminate the overhead associated with traditional networking stacks, NOW allows user processes direct access to virtual networks. An intelligent network interface enforces protection and shares the physical network, so multiple parallel applications and parallel operating system functions can be mixed freely. The global system layer offers a uniform view of the collective resources and allows interactive use. Because of its incremental scalability and tolerance for node failures, this approach is used for very large Internet servers, such as the Inktomi search engine developed at the University of California, Berkeley. Owing to the low cost, many labs are constructing their own custom cluster systems.

Hewlett-Packard Exemplar X-Class. The Exemplar X-Class, also known as the SPP 2000, pushes the other end of the design spectrum. This ccNUMA DSM system has nodes (called hypernodes) consisting of SMPs with 16 (180MHz PA-8000) processors and up to 4GB of memory with 16Gb chips. Within a hypernode, processors and memory are connected through an 8 3 8 "bars" nonblocking crossbar switch, in which each link can send and receive 960MB/sec in each direction. The hypernodes are connected through the Coherent Toroidal Interconnect (CTI), a proprietary technology using protocols based on the Scalable Coherent Interconnect standard. Multiple CTI rings are used to form a toroidal topology to link the hypernodes. The aggregate interconnection bandwidth between a pair of nodes is 3.84GB/sec. Maximum system size is 32 nodes (512 processors), with a peak speed of 369Gflops. Message-passing and shared-memory programming models are supported.

Tera MTA-1. The Tera system uses fully custom-designed processors with a variety of sophisticated hardware designs providing uniform access to a large shared memory as well as latency tolerance. The goal is a very high fraction of peak speed even on programs accessing memory in irregular patterns.

While high memory bandwidth is part of the approach (the performance goal is one load or store per clock cycle per processor, regardless of memory location), several other mechanisms are employed. One approach to hiding latency is to have a number of tasks (or threads) eligible to run on each processor. Then, when the program calls for a memory operation that cannot be satisfied immediately, that task can be held in abeyance, and another task that has all its data and is ready to execute some instructions can be given access to the processor's functional units. For this approach to be effective, the work to be done between task switches must take (much) longer than the time required to switch tasks. The MTA-1's processors switch tasks after every instruction and thus can support many active threads per processor with little or no time penalty.

Another memory access issue in parallel computers is how to synchronize updates to a given memory location when several processors are updating the same word. Every word in the MTA-1 memory has a "full-empty" bit indicating whether a word has been updated and can synchronize memory accesses in a single cycle. These mechanisms and others are aimed at keeping the processors busy instead of spending a large fraction of their time waiting for data, assuming there is enough work ready to be done at all times, that is, there is enough parallelism. Given that very fine-grain parallelism can be exploited, processors can be kept busy most of the time.

oratory. Some vector systems are moving toward the use of complementary metal oxide (CMOS) processors, such as in the Cray J90 family and in some of the Fujitsu line. Although the clock speed is noticeably slower, CMOS vector processors deliver much better price/performance.

More commercial software is available for vector machines than for systems based on microprocessors, and their system software is still more sophisticated and stable. Consequently, they excel in providing throughput in a typical production environment. However, for the largest Grand Challenge-type applications, almost all current systems are the new breed of highly scalable, parallel systems built largely out of commodity parts (see the sidebar Reflecting the Trends).

Millions of Concurrent Operations

The need for increased computer power—speed and memory size—is greater than ever. If processors and memories could be made faster and faster, the needed capability could be provided by only a handful of processors coupled to a large, single-level memory. Such a system would be comparatively easy to use. Unfortunately, the building blocks that will be available to build the fastest computer systems will force us to deal with millions of concurrent operations, more complicated memory hierarchies, and much greater memory access latencies.

In the past 10 years, peak computer speeds increased by nearly three orders of magnitude. What if we want another three-order-of-magnitude increase in the next 10 years? Will there be a smooth continuation of the current trend for the foreseeable future? Probably not. Whereas 10 years ago it was clear how to design and build a gigaflops computer and five years ago there was a clear path to teraflops machines, there is no obvious and realistic approach for obtaining the next three-order-of-magnitude increase in performance—to a system capable of 10^{15} floating-point operations per second (1Pflops). In fact, an even more ambitious goal would be to build a petaOps system, given that historically it has been more difficult to increase the rate of logical and integer operations than that of floating-point operations.

The dominant component technology—CMOS—will reach its limits in a decade or so. The clock speed for high-performance microprocessors between now and 2010 is forecast by the Semiconductor Industry Association (SIA) to increase from 500MHz to 1,100MHz, or less than a factor of three. While the state of the art in CMOS fabrication is ahead of recent SIA projections, the consensus at the NASA Ames Workshop on Device Modeling in 1997 was that Moore's Law will hold true for at least two more generations, until feature sizes of 0.08–0.13μm are reached. At that point, in about 2010, the limits of CMOS on silicon will have been reached.

The implications of hitting this wall in the next 10–15 years are sobering. One is that when CMOS reaches its limits, new devices and fabrication technologies (not manufacturable with current technology) will have to be used to continue the historic trends. This prospect is worrisome, since it usually takes 10–15 years to put new technologies into commercial production, and we may be less than 15 years away from reaching the limits of CMOS. To put this prediction into perspective, recall that in 1984 Seitz and Matisoo [10] discussed engineering limits of semiconductor design. Minimum feature sizes were 2μm and pushing toward 1μm, which at the time was thought to be close to the limit for photolithography using violet light. Feature sizes of 0.1μm had been demonstrated in the laboratory by 1984, yet even today, that is a smaller feature size than is in use commercially.

This realization leads one to consider several approaches:

- Component technologies other than CMOS to achieve much higher speeds;
- Further enhancements to processor architectures to produce more results per processor cycle and better utilization of each transistor (reducing power requirements); and
- New system architectures (including topologies and interconnection schemes) to deal with the very large number of processors and memory units needed for petaflops systems.

A second consequence would be that to continue building faster computers, more parallelism will have to be exploited. If cycle speeds increase by only a factor of three, architecture will have to contribute a factor of nearly 90 to get the expected 256-fold increase in speed in 12 years. Microprocessors will have to support more concurrent operations, and more processors will have to be used in each computer system. It is already difficult to use the parallelism in today's computer systems and processors. Even another two orders of magnitude in parallelism pose quite a challenge. Even if the Moore's Law factor of 256 is achieved, another factor of four will be needed to build a petaflops computer with CMOS microprocessors.

Memory Access

Yet another trend portends more difficulty in achieving much higher application performance in the coming years—the disparity between speed increases

in processors (60% per year) and in DRAM memories (7% per year). This trend, coupled with physically distributed memory architectures, is leading to very nonuniform memory access times, with latencies ranging from a couple of processor cycles for data in cache to hundreds of thousands of cycles. To mitigate these effects, it will be necessary to introduce additional levels of memory hierarchy and architectural features to handle many concurrent threads.

A thread is a computer science abstraction denoting a group of instructions scheduled as one unit. Multithreaded programs provide a way to hide memory access latency by decomposing the work to be done into individual tasks (threads); determining any data or synchronization constraints that have to be met for execution; holding threads in suspense while data is fetched from the memory hierarchy; and scheduling threads on processors or functional units when the requisite data is available.

The multithreading approach hides memory latency if the ratio of threads to processors is high enough, meaning there is always work to be done by a sufficient number of "ready" threads. While ready threads are executing, data needed by threads not yet schedulable can be migrating up the memory hierarchy, behind the scenes. The Tera MTA-1 architecture described in the sidebar is the leading example of multithreaded architecture today.

Another, still emerging, approach for improving access to memory is called processor in memory (PIM) [6]. Latency-hiding to overcome processor-memory speed differences is the focus of a great deal of architecture research, and the resulting mechanisms, such as the use of large caches and memory prefetching, consume a large number of transistors in today's systems. PIM promises to overcome this problem and is likely to be widely used in commodity processor architectures in addition to more specialized high-performance systems. Intelligent RAM is a similar recent effort [8].

PIM places processing logic on the memory chip. DRAM, despite being the most commonly used memory type due to its low cost, suffers from very low bandwidth. This shortcoming is not inherent in DRAM but a result of how a DRAM chip is interfaced to the rather narrow memory buses off-chip. DRAM is organized into rows and columns. Memory access operations place an entire selected row into an on-chip row buffer at high bandwidth (such as 256 to 4,096 bits at a time); column access selects the

SIMD vs. MIMD vs. VLIW

Computer architectures can incorporate parallelism in many different forms, including SIMD, MIMD, and VLIW.

SIMD. SIMD architectures consist of systems with multiple processors to which the same instruction is broadcast on each cycle. Each processor executes the same operation but on its own (different) operands (or it executes no operation if it is inappropriate). Hence the name: single instruction, multiple data. SIMD architectures typically employ simple processors in large quantities, often organized in 2D arrays, and are effective for applications that lend themselves to lock-step modes of operation on many distinct data. Data can be loaded in parallel into each of thousands of processors, and thousands of operations can be carried out in a single cycle. For example, configurations of the CM-2 had as many as 65,536 processors. The drawback of SIMD machines is that many problems do not have a regular pattern, and most processors execute null instructions on each cycle.

MIMD. MIMD systems have multiple processors into which different programs and data can be loaded. Consequently, each processor can execute different instructions at any given point in time. Even when the same program is loaded into each processor's memory (a common approach), the program can take different branches depending on the data in its memory and therefore execute different instructions from its companions.

VLIW. VLIW processors have more than one functional unit for such tasks as load/store, floating-point add/multiply, and integer arithmetic/logic operations. (Processors with recent incarnations of this approach are also called *superscalar*.) During each cycle, at the stage when conventional processors would fetch a single instruction to be executed, a VLIW processor fetches several instructions simultaneously. To make this possible, the instructions are grouped in a single "instruction word" the processor reads from memory. The operations specified by the instructions in that word are then carried out simultaneously by the processor's functional units. For example, in a single cycle, the processor might execute one phase of two distinct floating-point operations, issue a load for operands needed for instructions that will be executed a few cycles in the future, and perform an integer operation to decode an address. In VLIW machines, the operations must be scheduled carefully to manage the movement of operands so they are in the right place at the right time and the multiple functional units can be kept busy most of the time. Compiler technology has been developed to make this orchestration possible.

desired word from the row buffer and sends it onto the off-chip data bus a few (1–9) bits at a time. Placing processors onto the same chip and arranging the bit buffers makes it possible to achieve much higher bandwidths. A 16Mb IBM DRAM recently achieved 2GB/sec internal bandwidth, and a 1Gb synchronous Mitsubishi RAM achieved internal bandwidths of more than 60GB/sec. Besides giving much greater bandwidth to memory, this design frees up data pins for off-chip transfers to I/O devices or connections to other processor-memory chips for parallel processing.

To have a balanced system as system speeds and memory sizes increase, I/O performance has to increase as well. Since individual devices, such as disk controllers and network interfaces, are relatively slow and not improving quickly, high levels of parallelism have to be used to achieve required I/O performance. Software for managing this parallelism is quite limited today, often requiring undue effort from the user. As petaflops systems emerge, the I/O and storage challenges become quite demanding, involving the following:

- Online storage. A modest disk system for a petaflops computer with 34TB of memory needs a capacity of 1PB. For disk technology in 2007, this need would require 22,000 drives of 45GB each. Their aggregate bandwidth would be 352GB/sec.
- I/O. Checkpointing 34TB of memory to disk in five minutes requires 113GB/sec sustained. How can 22,000 disks be driven in parallel at 33% of their aggregate peak speed?
- Archival storage. A petaflops computer with 1PB of online disk space requires an exabyte of archival storage. With today's tape and robot technology, this calls for 500,000 cubic feet of space.

The Limits

System performance cannot be expected to continue growing at historical rates for much longer. To the extent it grows, it will be primarily at the cost of complexity.

Meanwhile, the structure of the highest-performance systems of the future will pose formidable software challenges. At the system architecture level, the operating system will have to manage millions of active threads and data movement across several more levels of memory hierarchy. Processor architectures, with their ever-increasing internal parallelism and multilevel memories, will require compiler and run-time systems technology well beyond the state of the art. Application software will have to carry an even greater share of the burden for efficient use of the systems.

It is clear that much long-term research is needed to address the hardware, architecture, and software challenges of petaflops systems. With only a decade before current technologies run out of steam, now is the time to start investigating how to get the next factor of 1,000 in system performance. C

References

1. Abandah, G., and Davidson, E. Effects of architectural and technological advances on the HP/Convex Exemplar's memory and communication performance. In *Proceedings of the 25th Annual International Symposium on Computer Architecture* (Barcelona, Spain, June 27–July 1 1998), pp. 318–329.
2. Alverson, G., Briggs, P., Coatney, S., Kahan, S., and Korry, R. Tera hardware-software cooperation. In *Proceedings of the ACM\IEEE SC97* (San Jose, Calif., Nov. 15–21). IEEE Computer Society Press, Piscataway, N.J., 1997.
3. Anderson, T., Culler, D., and Patterson, D. A case for NOW (Networks of Workstations). *IEEE Micro 15,* 1 (Feb. 1995), 54–56; see also now.cs.berkeley.edu.
4. ASCI97 (Accelerated Strategic Computing Initiative). www.sandia.gov/ASCI/Red/
5. Culler, D., Singh, J., and Gupta, A. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers, San Francisco, Calif., 1998.
6. Kogge, P. Execube: A new architecture for scalable MPPs. In *Proceedings of the International Conference on Parallel Processing* (St. Charles, Ill., Aug. 1994), pp. 77–84.
7. Messina, P. The concurrent supercomputing consortium: Year 1. *Paral. Distrib. Tech. 1,* 1 (Feb. 1993), 9–16.
8. Patterson, D., Anderson, T., Cardwell, N., Fromm, R., Keeton, K., Kozyrakis, C., Thomas, R., and Yelick, K. A case for Intelligent RAM. *IEEE Micro 17,* 2 (Apr. 1997).
9. Scott, S. Synchronization and communication in the T3E multiprocessor. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems* (Cambridge, Mass., Oct. 1996), pp. 26–36.
10. Seitz, C., and Matisoo, J. Engineering limits on computer performance. *Phys. Today 37,* 5 (May 1984), 38–45.
11. Stunkel, C., Shea, D., Abali, B., Atkins, M., Bender, C., Grice, D., Hochschild, P., Joseph, D., Nathanson, B., Swetz, R., Stucke, R., Tsao, M., and Varker, P. The SP2 high-performance switch. *IBM Syst. J. 34,* 2 (1995), 185–204.
12. Tucker, L., and Robertson, G. Architecture and applications of the Connection Machine. *Comput. 21,* 8 (Aug. 1988), 26–38.

Paul Messina (messina@cacr.caltech.edu) is assistant vice president for scientific computing and director of the Center for Advanced Computing Research at the California Institute of Technology, Pasadena, Calif.

David Culler (culler@cs.berkeley.edu) is a professor in the Computer Science Division of the University of California, Berkeley.

Wayne Pfeiffer (wpfeiffer@ucsd.edu) is deputy director of the San Diego Supercomputer Center and leader of the NPACI Resources Working Group.

Wiliam Martin (wrm@umich.edu) is associate dean for academic affairs, professor of nuclear engineering and radiological sciences, and director of the Laboratory for Scientific Computation at the University of Michigan, Ann Arbor, Mich.

J. Tinsley Oden (oden@ticam.utexas.edu) holds the Cockrell Family Regents Chair in Engineering and is director of the Texas Institute for Computational and Applied Mathematics at the University of Texas, Austin.

Gary Smith (gary@hpcf.cc.utexas.edu) is associate director of academic computing and instructional technology services at the University of Texas, Austin.

© 1998 ACM 0002-0782/98/1100 $5.00