The Quest for Petascale Computing

Although the challenges to achieving petascale computing within the next decade are daunting, several software and hardware technologies are emerging that could help us reach this goal. The authors review these technologies and consider new algorithms capable of exploiting a petascale computer's architecture.

One petaflop per second is a rate of computation corresponding to 10^{15} floating-point operations per second. To be of use in scientific computing, a computer capable of this prodigious speed needs a main memory of tens or hundreds of terabytes and enormous amounts of mass storage. Sophisticated compilers and high memory and I/O bandwidth are also essential to efficiently exploit the architecture. To mask the hardware and software complexities from the scientific end user, it would be advantageous to access and use a petascale computer through an advanced problem-solving environment. Immersive visualization environments could play an important role in analyzing and navigating the output from petascale computations. Thus, petascale computing is capable of driving the next decade of research in high-performance computing and communications and will require advances across all aspects of it.

A report from the President's Information Technology Advisory Committee recommends that federal research programs "… drive high-end computing research by trying to attain a sustained petaops/petaflops on real applications by 2010 through a balance of software and hardware strategies." (For a copy of the committee's report, see www.ccic.gov/ac/report.) The report identifies many applications likely to benefit from petascale computing, including[1]

- nuclear weapons stewardship,
- cryptology,
- climate and environmental modeling,
- 3D protein molecule reconstruction,
- severe storm forecasting,
- design of advanced aircraft,
- molecular nanotechnology,
- intelligent planetary spacecraft, and
- real-time medical imaging.

The application of petascale computing to these areas should enhance US economic competitiveness and our knowledge of the universe around us,[2] and such factors have motivated a small band of experts to address petascale computing issues through a series of workshops and conferences. Such activities engendered small-scale point design studies, which the National Science Foundation, DARPA, and NASA funded to look at different aspects of petascale computer design. Recently, these agencies funded a project to examine in more detail the so-called hybrid technology multithreaded (HTMT) computer as a pathway to petascale computing.

This article discusses the current status of ef-

1521-9615/01/$10.00 © 2001 IEEE

Jack J. Dongarra, *University of Tennessee*
David W. Walker, *Cardiff University*

forts to make petascale computing a reality, but first we review trends in supercomputer performance over the past 50 years.

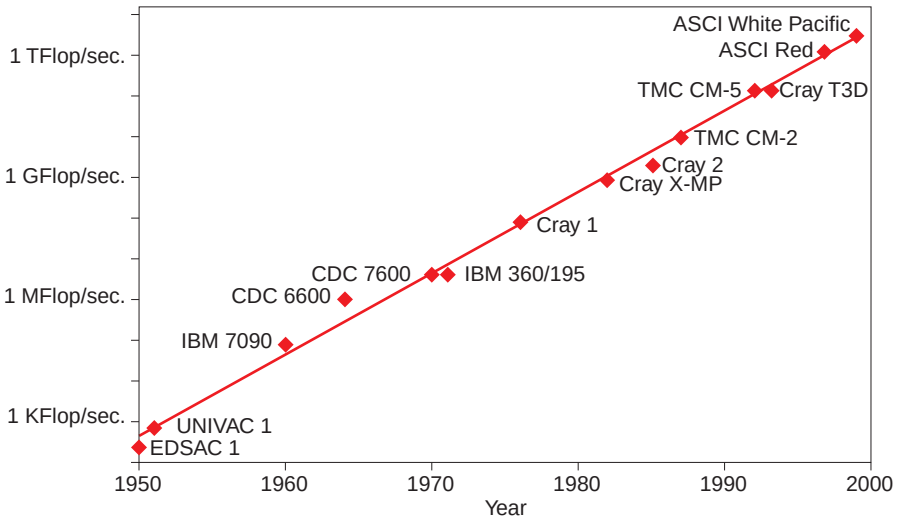

Figure 1. Moore's Law and the peak performance of various computers over time.

Trends in supercomputer performance

In the last 50 years, the scientific computing field has seen rapid changes in vendors, architectures, technologies, and system usage. However, despite all these changes, the long-term evolution of performance seems to remain steady and continuous. Moore's Law is often cited in this context. If we plot the peak performance of the leading supercomputers over the last five decades (see Figure 1), we see that this law holds well for almost the complete lifespan of modern computing—on average, performance increases by two orders of magnitude every decade.[3]

To provide a better basis for statistics on high-performance computers, a group of researchers initiated the Top500 list,[4] which reports the sites with the 500 most powerful computer systems installed. The best Linpack benchmark performance[5] achieved is used as a performance measure in ranking the computers. The Top500 list has been updated twice a year since June 1993.

Although many aspects of the HPC market change over time, the evolution of performance seems to follow empirical laws, such as Moore's Law. The Top500 data provides an ideal basis to verify an observation like this. Looking at the computing power of the individual machines in the Top500—and the evolution of the total installed performance—we can plot the performance of the systems at positions one, 10, 100 and 500 in the list, as well as the total accumulated performance. In Figure 2, the curve of position 500 shows an average increase by a factor of two per year. All other curves show a growth rate of 1.8, plus or minus 0.07 per year.

Based on current Top500 data (which cover the last seven years), and the assumption that the current performance trends will continue for some

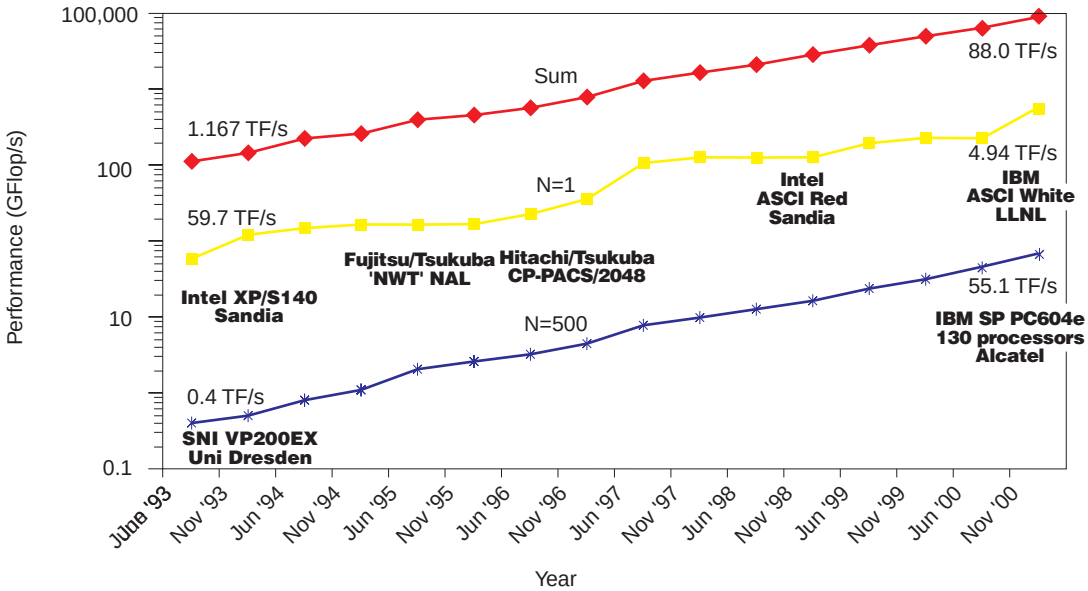

Figure 2. The overall growth of accumulated and individual performance as seen in the Top500 report.

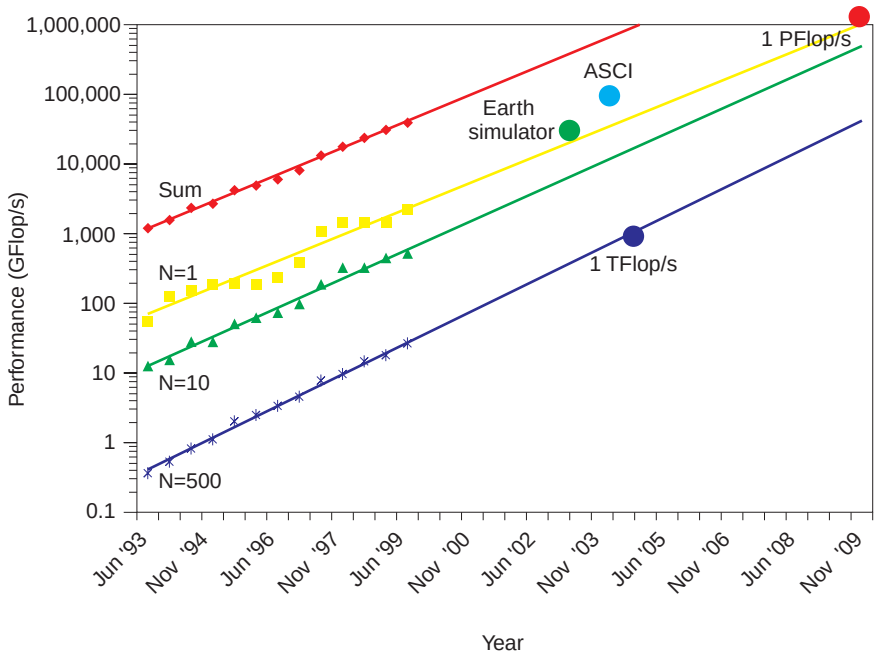

Figure 3. Extrapolation of recent growth rates of performance seen in the Top500.

time to come, we can extrapolate the observed performance. In Figure 3, we do this by using linear regression on a logarithmic scale. This means that exponential growth is fitted for all levels of performance in the Top500. These simple fits to the data show surprisingly consistent results. Based on the extrapolation from these fits, we can expect the first 100 TFlop/s system to be available by 2005, which is one or two years later than the Accelerated Strategic Computing Initiative (ASCI) anticipates. By 2005, no system smaller then 1 TFlop/s will make the Top500 list.

Looking even further into the future, we can speculate, based on the current doubling of performance every year, that the first petaflop system should be available around 2009. However, due to the rapid changes in the technologies used in HPC systems, predicting such a system's architecture is difficult—although the HTMT architecture and IBM's Blue Gene design offer promising avenues to petaflop/s performance. Even though the HPC market has changed substantially in the 25 years since the Cray 1's debut, there is still no end in sight to the rapid cycles of technological innovation that characterize supercomputing's history.

Hardware technologies for petascale computing

We can group hardware technologies for achieving petaflop/s computing performance into five main categories: conventional technologies, processing-in-memory (PIM) designs, designs based on superconducting processor technologies, special-purpose hardware designs, and schemes that use the aggregate computing power of Web-distributed processors. We only consider here technologies currently available or that are expected to be available in the near future. Thus, we don't discuss designs based on spectulative technologies, such as quantum[6] or macromolecular[7] computing, although they might be important in the long run.

Conventional technology

Systems based on conventional technology are those produced by extrapolating current mainstream compute chip design and fabrication technologies into the future. This extrapolation is largely based on the projects published by the Semiconductor Industry Association, the leading trade association representing US computer chip manufacturers.[8] SIA predicts 3.5-GHz clock speeds by 2006, which should increase to 6 GHz by 2009. Assuming 70-nanometer lithography, the SIA also expects chips to possess 8 Gbytes of DRAM and 1 Gbyte of SRAM by 2009.

Rick Stevens at Argonne National Laboratory posits that a 2.5-Pflop/s system will be available by 2010.[9] This system is based on 4.8-GHz processors and is composed of 8,000 symmetric multiprocessor nodes. Each node contains 16 CPUs and has an expected performance of 300 Gflop/s. The system's total memory would be 32.8 Tbytes, with 1.3 Pbytes of disk storage. Stevens predicts the system will cost approximately $16 million. However, because it would need 8 megawatts of power, energy consumption would cost about $8 million per year.

Steve Wallach of CenterPoint Venture Partners has put forward a design for a commercial off-the-shelf petaflop system that is similar to that of Stevens.[10] Wallach proposes an 8,192-node system with four CPUs per node and a performance of 120 Gflop/s per node. This yields a system with a total performance of approximately 1 Pflop/s. In estimating his system's power requirements, Wallach includes the en-

ergy the main memory and the CPUs will consume, giving a total of about 4.6 megawatts. Wallach's design proposes an optical interconnection network between the system's nodes based on OC768 channels. These would provide a bisection bandwidth of 50 Tbyte/s.

PIM designs

A central issue in the design and use of HPC systems is the fact that the peak processing speeds of CPUs has increased at a faster rate than memory access speeds. With conventional technologies, the ratio of memory access time to CPU cycle time will continue to increase and is expected to exceed 400 by 2010.[11] This has led to HPC systems with complex memory hierarchies. A number of latency-hiding techniques reduce this disparity's impact to improve the efficiency with which applications use the hardware, including hardware and compiler support for multithreaded programming and structuring algorithms to maximize data reuse in the upper levels of memory.

PIM designs seek to reduce the memory access bottleneck by integrating processing logic and memory on the same chip, thereby dramatically improving the memory access time to CPU cycle time ratio. Peter Kogge and his colleagues at IBM developed Execube, the first such chip.[12] Analog Devices introduced the Sharc "system-on-a-chip" in 1994, incorporating a digital signal processor and 0.5 Mbytes of SRAM. In 1996, Mitsubishi produced the M32R/D, the first commercial DRAM PIM chip, which integrates a 20-MHz processor and 2 Mbytes of DRAM. Developers have since produced a number of PIM products and projects (see www.ai.mit.edu/projects/aries/course/notes/pim.html).

Digital superconductor technologies

Clock speeds for semiconductor logic circuits are expected to reach a peak of a few GHz within the next 10 years, mainly due to prohibitively large power requirements. However, digital superconductor technology, based on Rapid Single-Flux-Quantum (RSFQ) logic, has achieved speeds of 760 GHz and has extremely low power consumption. These features, together with a simple fabrication technology, make digital superconductor devices a promising basis for petascale computers. In such devices, a single quantum of magnetic flux encodes digital bits, while data are transferred as picosecond "SFQ" pulses. However, logic devices based on superconducting niobium must be cooled to between 4 to 5 degrees Kelvin, although the cost of refrigeration would make up only a fraction of a petascale system's total cost. Current plans[13] are to use 0.8 μm RSFQ technology as a key component in an HTMT petascale computer.

Special-purpose hardware

Researchers on the Grape project (short for *gra*vity *p*ipe, see http://grape.c.u-tokyo.ac.jp/grape) at the University of Tokyo have designed and built a family of special-purpose attached processors for performing the gravitational force computations that form the inner loop of *N*-body simulation problems. The computational astrophysics community has extensively used Grape processors for *N*-body gravitational simulations. A Grape-4 system consisting of 1,692 processors, each with a peak speed of 640 Mflop/s, was completed in June 1995 and has a peak speed of 1.08 Tflop/s.[14] In 1999, a Grape-5 system won a Gordon Bell Award in the performance per dollar category, achieving $7.3 per Mflop/s on a tree-code astrophysical simulation. A Grape-6 system is planned for completion in 2001 and is expected to have a performance of about 200 Tflop/s. A 1-Pflop/s Grape system is planned for completion by 2003.

Clock speeds for semiconductor logic circuits are expected to reach a peak of a few GHz within the next 10 years.

Web-based petascale computing

The Web's aggregate computing power is enormous, but because of severe latency and bandwidth constraints, we can harness it for only very loosely coupled applications. We can divide examples of Web-based computations into those that are community-based (and use underutilized PCs across the Web) and those that involve a higher degree of coordination and more dedicated resources. Examples of community-based Web computing projects include

- Cracking the 56-bit DES encryption standard, which the Electronic Frontier Foundation's "Deep Crack" computer achieved in January 1999 in a record time of 22 hours and 15 minutes with a worldwide network of nearly 100,000 PCs.
- The Great Internet Mersenne Prime Search, which recently discovered a two-million-digit prime number by using a network of tens of thousands of PCs. This work was done under the auspices of Entropia (www.entropia.com)

and claims a performance of 1 Tflop/s.
- The SETI@home project, which uses underutilized PCs across the Web to search for extraterrestrial intelligence.[15]

We need one million 1-Gflop/s PCs to reach a performance of 1 Pflop/s for a community-based Web computing project; it is not inconceivable to achieve this within the next few years. Here, the definition of an application becomes rather strained because once the computational parts that come together to solve a problem become sufficiently uncoordinated, it is not clear that they still comprise an application in the traditional sense.

Many of the lessons learned in the HTMT project are likely to be incorporated in first-generation petaflop computers.

Computing resources distributed over the Web have been used for a number of scientific applications—generally, they possess a greater degree of coordination than the community-based Web computing projects discussed earlier. However, the higher latency and low bandwidth that currently characterize communication on the Web mean that we can exploit concurrency in such distributed scientific applications only at coarse granularity. Communication between the coarse grain components of these applications is typically performed with a network-oriented transport layer such as Corba, Globus,[16] or Legion.[17] An example of this class of Web-based application is the numerical relativity computations that have been performed using computers distributed across the US and Europe.[18] Because of high communication costs, typical scientific simulation applications are unlikely to achieve petaflop/s performance across the Web in the foreseeable future.

The HTMT project

The HTMT project is a partnership between industry, academia, and the US Government to develop a petascale computer by the year 2010. The intention is to exploit a number of leading-edge hardware technologies to build a scalable architecture that delivers high performance and has acceptable cost and power consumption. The HTMT machine has a deep memory hierarchy, and latency reduction and hiding are key in its design and use. Here are some of the technologies central to the HTMT concept:

- superconductor RSFQ logic, which is the basis for superconductor processing elements (Spells) with a clock speed of around 150 GHz. HTMT contains 2,048 processors, each capable of 600 Gflop/s, giving a total peak speed of 1.2 Pflop/s;
- SRAM and DRAM PIM chips;
- a data vortex optical interconnection network with communication speed of 500 Gbps per channel; and
- hardware support, for latency management.

The HTMT architecture has four levels of distributed memory. Each Spell has 1 Mbyte of cryogenically cooled RAM maintained at liquid helium temperature. The next two levels in the memory hierarchy are SRAM and DRAM PIM-based memory, and the fourth is holographic 3/2 memory. The SRAM PIMs are cooled to liquid nitrogen temperature and are connected to DRAM main memory by an optical packet switch network.

Latency management across these multiple levels of memory is crucial to the HTMT execution model. Multithreading is a commonly used way of hiding latency by context switching between concurrent threads and is the basis of the HTMT execution model. Multithreading and context prefetching are important techniques for latency tolerance in the hierarchical memory. In the HTMT execution model, a multilevel multithreaded context-management strategy, known as *thread percolation*, automatically migrates contexts through the memory hierarchy to keep the Spells supplied with work.[19] The memory hierarchy's PIM-based levels control thread percolation—they decide which computations will be performed next at a coarse granularity, and the Spells schedule and perform the computations at a fine granularity.

Programming methodologies for petascale computing

Many of the lessons learned in the HTMT project are likely to be incorporated in first-generation petaflop computers—a multithreaded execution model and sophisticated context management are approaches that might be necessary to provide hardware and system-level software support. These approaches could help control the impact of latency across multiple levels of hierarchical memory. Thus, although a petaflop computer might have a global name space, the application programmer

will still need to be aware of the memory hierarchy and program in a style that makes optimal use of it and exploits data locality wherever possible. Algorithms and applications will have to be carefully crafted to achieve an acceptable fraction of peak performance. Furthermore, applications will have to possess a high degree of concurrency to provide enough threads to keep the processors busy.

The main programming methodologies likely to be used on petascale systems will be

- A data parallel programming language, such as HPF, for very regular computations.
- Explicit message passing (using, for example, MPI) between sequential processes, primarily to ease porting of legacy parallel applications to the petascale system. In this case, an MPI process would be entirely virtual and correspond to a bundle of threads.
- Explicit multithreading to get acceptable performance on less regular computations.

Given the requirements of high concurrency and latency tolerance, highly tuned software libraries and advanced problem-solving environments are important in achieving high performance and scalability on petascale computers. Clearly, the observation that applications with a high degree of parallelism and data locality perform best on high-performance computers will be even truer for a petascale system characterized by a deep memory hierarchy. It is also often the case that to achieve good performance on an HPC system, a programming style must be adopted that closely matches the system's execution model. Compilers should be able to extract concurrent threads for regular computations, but a class of less regular computations must be programmed with explicit threads to get good performance. It might also be necessary for application programmers to bundle groups of threads that access the same memory resources into strands in a hierarchical way that reflects the hierarchical memory's physical structure. Optimized library routines will probably need to be programmed in this way.

IBM's proposed Blue Gene computer (www.research.ibm.com/bluegene) uses massive parallelism to achieve petaop/s performance. Therefore, it can be based on a less radical design than the HTMT machine, although it still uses on-chip memory and multithreading to support a high-bandwidth, low-latency communication path from processor to memory. Blue Gene is intended for use in computational biology applications, such as protein folding, and will contain on the order of one million processor–memory pairs. Its basic building block will be a chip that contains an array of processor–memory pairs and interchip communication logic. These chips will be connected into a 3D mesh.

New algorithms for petascale computing

Computations that possess insufficient inherent parallelism (or in which we cannot exploit parallelism efficiently because of data dependencies) are not well suited for petascale systems. Therefore, a premium is placed on algorithms that are regular in structure but that may have a higher operation count over more irregular algorithms with a lower operation count. Dense matrix computations, such as those the Lapack software library provides,[20] optimized to exploit data locality are an example of regular computations that should perform well on a petascale system.

The accuracy and stability of numerical methods for petascale computations are also important issues because some algorithms can suffer significant losses of numerical precision. Thus, slower but more accurate and stable algorithms are favored over faster but less accurate and stable ones. To this end, hardware support for 128-bit arithmetic is desirable, and interval-based algorithms may play a more important role on future petascale systems than they do on current high-performance machines. Interval arithmetic provides a means of tracking errors in a computation, such as initial errors and uncertainties in the input data, errors in analytic approximations, and rounding error. Instead of being represented by a single floating-point number, each quantity is represented by a lower and upper bound within which it is guaranteed to lie. The interval representation, therefore, provides rigorous accuracy information about a solution that is absent in the point representation.

Slower but more accurate and stable algorithms are favored over faster but less accurate and stable ones.

The advent of petascale computing systems might promote the adoption of completely new methods to solve certain problems. For example, *cellular automata* are highly parallel, are very

regular in structure, can handle complex geometries, and are numerically stable, and so are well suited to petascale computing. CA provide an interesting and powerful alternative to classical techniques for solving partial differential equations. Their power derives from the fact that their algorithm dynamics mimic (at an abstract level) the fine-grain dynamics of the actual physical system, permitting the emergence of macroscopic phenomenology from microscopic mechanisms. Thus, complex collective global behavior can arise from simple components obeying simple local interaction rules. CA algorithms are well suited for applications involving nonequilibrium and dynamical processes. The use of CA on future types of "programmable matter" computers is discussed elsewhere[21]—such computers can deliver enormous computational power and are ideal platforms for CA-based algorithms. Thus, on the five-to-10-year timescale, CA will play an increasing role in the simulation of physical (and social) phenomena.

Cellular automata will play an increasing role in the simulation of physical (and social) phenomena.

Another aspect of software development for petascale systems is the ability to automatically tune a library of numerical routines for optimal performance on a particular architecture. The Automatic Tuned Linear Algebra Software project exemplifies this concept.[22] Atlas is an approach for the automatic generation and optimization of numerical software for computers with deep memory hierarchies and pipelined functional units. With Atlas, numerical routines are developed with a large design space spanned by many tunable parameters, such as blocking size, loop nesting permutations, loop unrolling depths, pipelining strategies, register allocations, and instruction schedules. When Atlas is first installed on a new platform, a set of runs automatically determines the optimal parameter values for that platform, which are then used in subsequent uses of the code. We could apply this idea (with further research) to other application areas, in addition to numerical linear algebra, and extend it to develop numerical software that can dynamically explore its computational environment and intelligently adapt to it as resource availability changes.

In general, a multithreaded execution model implies that the application developer does not have control over the detailed order of arithmetic operations (such control would incur a large performance cost). Thus, numerical reproducibility, which is already difficult to achieve with current architectures, will be lost. A corollary of this is that for certain problems, it will be impossible to predict a priori which solution method will perform best or converge at all. An example is the iterative solution of linear systems for which the matrix is nonsymmetric, indefinite, or both. In such cases, an "algorithmic bombardment" approach can help. The idea here is to concurrently apply several methods for solving a problem in the hope that at least one will converge to the solution.[23] A related approach is to apply a fast, but unreliable, method first and then to check a posteriori if any problem occurred in the solution. If so, we use a slower method to fix the problem. These poly-algorithmic approaches can be available for application developers either as black boxes or with varying degrees of control over the methods used.

Two promising avenues for achieving petaflop performance by the year 2010 are the IBM Blue Gene initiative and the HTMT design, both of which use a number of emerging hardware technologies. A team involving Compaq, Sandia National Laboratories, and Celera Genomics announced in January 2001 their intention to develop a 100 Teraop/s computer by 2004, with the intention of building a 1 Petaop/s machine in a second development phase.

A deep memory hierarchy with disparate latencies and bandwidths characterizes the HTMT design, but any computer designed for petascale performance needs to address the memory access bottleneck problem. Good performance on these types of machines need to exploit on-chip integration of processors and memory, and a multithreaded execution model, with support at the hardware, system software, and compiler levels. In addition, latency-tolerant algorithms with low bandwidth requirements will be of crucial importance in achieving an acceptable fraction of peak performance. These algorithms generally have higher operation counts than the corresponding optimal sequential code, but they access the memory hierarchy more efficiently. Thus, CA algorithms might come into favor over traditional methods for solving partial differential equations on petascale architectures.

Intelligent, resource-aware numerical algo-

rithms that can automatically tune themselves are also a desirable feature. Extensive use of highly tuned software libraries and problem-solving environments to assist in application composition and job submission will be extremely important. Novel approaches, such as algorithmic bombardment, will address issues of numerical reproducibility. We will need interval arithmetic methods to get robust and accurate solutions to some problems.

Acknowledgments

This article is partly based on presentations made at the Second Conference on Enabling Technologies for Petaflop/s Computing, held 15–19 February 1999 in Santa Barbara, California.

References

1. D. Bailey, "Onwards to Petaflops Computing," *Comm. ACM*, vol. 40, no. 6, June 1997, pp. 90–92.
2. K. Kennedy, "Do We Need A Petaflops Initiative?" *Parallel Computing Research*, vol. 5, no. 1, Winter 1997; www.crpc.rice.edu/CRPC/newsletters/win97/index.html.
3. E. Strohmaier et al., "The Marketplace of High-Performance Computing," *Parallel Computing*, vol. 25, nos. 13–14, Dec. 1999, pp. 1517–1545.
4. J.J. Dongarra, H.W. Meuer, and E. Strohmaier, *Top500 Supercomputer Sites*, tech. report UT-CS 99-434, Dept. of Computer Science, Univ. of Tennessee, Knoxville, 1999.
5. J.J. Dongarra, *Performance of Various Computers Using Standard Linear Equations Software*, tech. report CS-89-85, Dept. of Computer Science, Univ. of Tennessee, Knoxville, 2000.
6. D. Aharonov, "Quantum Computation," *Ann. Reviews of Computational Physics*, D. Stauffer, ed., World Scientific, Singapore, 1998.
7. L. Adelman, "Molecular Computation of Solution to Combinatorial Problems," *Science*, vol. 266, 1994, p. 1021.
8. *International Technology Roadmap for Semiconductors*, Semiconductor Industry Association, Austin, Texas, 1999; http://public.itrs.net/files/1999_SIA.Roadmap/Home.htm.
9. R. Stevens et al., "The High Performance Computing Extreme Linux Open Source Systems Software Initiative," Second Extreme Linux Workshop, 1999 at the Usenix Ann. Technical Conf., www.extremelinux.org/activities/usenix99/docs/betagrid/StevensEL99/index.htm.
10. S. Wallach, "Petaflop Architectures," *Proc. Second Conf. Enabling Technologies for Petaflop/s Computing*, to be published; www.cacr.caltech.edu/pflops2/presentations/WallachPeta2.pdf.
11. P.M. Kogge, "In Pursuit of the Petaflop: Overcoming the Latency/Bandwidth Wall with PIM Technology," *Proc. Second Conf. Enabling Technologies for Petaflop/s Computing*, to be published; www.cacr.caltech.edu/pflops2/presentations/KoggePeta2.pdf.
12. P.M. Kogge, "The EXECUBE Approach to Massively Parallel Processing," *Proc. Int'l Conf. Parallel Processing*, CRC Press, 1994.
13. P. Bunyk et al., "RSFQ Subsystem for Petaflops-Scale Computing: 'COOL-0,'" *Proc. Third Petaflop Workshop*, 1999, pp. 3–9.
14. J. Makino et al., "Grape-4: A Massively Parallel Special-Purpose Computer for Collisional *N*-Body Simulations," *Astrophysical J.*, vol. 480, no. 1, May 1997, pp. 432–446.
15. E. Korpela et al., "SETI@home: Massively Distributed Computing for SETI," *Computing in Science & Eng.*, vol. 3, no. 1, Jan./Feb. 2001, pp. 78–83
16. I. Foster and C. Kesselman, "The Globus Project: A Status Report," *Future Generation Computer Systems*, vol. 15, nos. 5–6, Oct. 1999, pp. 607–621.
17. A.S. Grimshaw, W.A. Wulf, and the Legion Team, "The Legion Vision of a Worldwide Virtual Computer," *Comm. ACM*, vol. 40, no. 1, Jan. 1997.
18. W. Benger et al., "Numerical Relativity in a Distributed Environment," *Proc. Ninth SIAM Conf. Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1999.
19. G.R. Gao et al., *The HTMT Program Execution Model*, CAPSL Technical Memo 09, Dept. of Electrical and Computer Eng., Univ. of Delaware, 1997.
20. E. Anderson et al., *LAPACK User's Guide*, SIAM Press, Philadelphia, 1995.
21. T. Tofolli, "Programmable Matter Methods," *Future Generation Computer Systems*, vol. 16, nos. 2–3, Dec. 1999, pp. 187–201.
22. R.C. Whaley, A. Petitet, and J.J. Dongarra, "Automated Empirical Optimizations of Software and the ATLAS Project," to be published in *Parallel Computing*.
23. R. Barrett et al., "Algorithmic Bombardment for the Iterative Solution of Linear Systems: A Poly-Iterative Approach," *J. Computational and Applied Mathematics*, vol. 74, nos. 1–2, 1996, pp. 91–10.

Jack J. Dongarra is a University Distinguished Professor of Computer Science at the University of Tennessee, Knoxville. His technical interests include the design and implementation of open source software packages and systems. He earned a BS in mathematics from Chicago State University, an MS in computer science from the Illinois Institute of Technology, and a PhD in applied mathematics from the University of New Mexico. He is a fellow of the AAAS, ACM, and the IEEE and a member of the National Academy of Engineering. Contact him at the Dept. of Computer Science, Univ. of Tennessee, 1122 Volunteer Blvd., Knoxville, TN 37996-3450; dongarra@cs.utk.edu.

David W. Walker is Professor of High-Performance Computing in the Department of Computer Science at the Cardiff University in the UK. His research interests focus on software, algorithms, and environments for computational science on HPC systems and the design and implementation of problem-solving environments for scientific computing. He received a BA in mathematics from the University of Cambridge and his MSc and PhD in physics from the University of London. Contact him at the Dept. of Computer Science, Cardiff Univ., PO Box 916, Cardiff CF24 3XF, UK; david@cs.cf.ac.uk.