The Boole Lecture
Trends in High Performance Computing

JACK DONGARRA

Department of Computer Science, University of Tennessee, Knoxville, TN 37996-3450, USA and Oak Ridge National Laboratory
Email: dongarra@cs.utk.edu

The Annual Boole Lecture was established and is sponsored by the Boole Centre for Research in Informatics, the Cork Constraint Computation Centre, the Department of Computer Science, and the School of Mathematics, Applied Mathematics and Statistics at University College Cork. The series is named in honour of George Boole, the first professor of Mathematics at UCC, whose seminal work on logic in the late 1800s is central to modern digital computing. To mark this great contribution, leaders in the fields of computing and mathematics are invited to talk to the general public on directions in science, on past achievements and on visions for the future.

Received 23 January 2004; revised 10 March 2004

1. HISTORICAL PERSPECTIVE

In the last 50 years, the field of scientific computing has undergone rapid change—we have experienced a remarkable turnover of technologies, architectures, vendors and usage of systems. Despite all these changes, the long-term evolution of performance seems to be steady and continuous, following Moore's Law rather closely. In 1965 Gordon Moore, one of the founders of Intel, conjectured that the number of transistors per square inch on integrated circuits would roughly double every year. It turns out that the frequency of doubling is not 12 months, but roughly 18 months [1]. Moore predicted that this trend would continue for the foreseeable future. In Figure 1, we plot the peak performance over the last five decades of computers that have been called 'supercomputers'. A broad definition for a supercomputer is that it is one of the fastest computers currently available. They are systems that provide significantly greater sustained performance than that available from mainstream computer systems. The value of supercomputers derives from the value of the problems they solve, not from the innovative technology they showcase. By performance we mean the rate of execution for floating point operations. Here we chart Kflop/s (kilo-flop/s, thousands of floating point operations per second), Mflop/s (mega-flop/s, millions of floating point operations per second), Gflop/s (giga-flop/s, billions of floating point operations per second), Tflop/s (tera-flop/s, trillions of floating point operations per second) and Pflop/s (peta-flop/s, 1000 trillions of floating point operations per second). This chart shows clearly how well this Moore's Law has held over almost the complete lifespan of modern computing—we see an increase in performance averaging two orders of magnitude every decade.

In the second half of the 1970s, the introduction of vector computer systems marked the beginning of modern supercomputing. A vector computer or vector processor is a machine designed to efficiently handle arithmetic operations on elements of arrays, called vectors. These systems offered a performance advantage of at least one order of magnitude over conventional systems of that time. Raw performance was the main, if not the only, selling point for supercomputers of this variety. However, in the first half of the 1980s the integration of vector systems into conventional

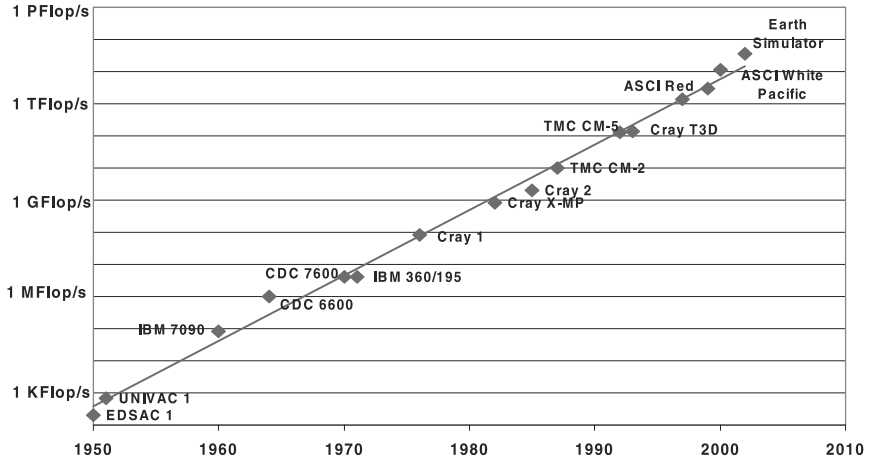

FIGURE 1. Moore's Law and peak performance of various computers over time.

computing environments became more important. Only those manufacturers who provided standard programming environments, operating systems and key applications were successful in getting the industrial customers, which became essential for survival in the marketplace. Performance was increased primarily by improved chip technologies and by producing shared-memory multiprocessor systems, sometimes referred to as symmetric multiprocessors or SMPs. An SMP is a computer system that has two or more processors connected in the same cabinet, managed by one operating system, sharing the same memory and having equal access to input/output devices. Application programs may run on any or all processors in the system; assignment of tasks is decided by the operating system. One advantage of SMP systems is scalability; additional processors can be added as needed up to some limiting factor determined by the rate at which data can be sent to and from memory.

Fostered by several government programs, scalable parallel computing using distributed memory became the focus of interest at the end of the 1980s. A distributed-memory computer system is one in which several interconnected computers share the computing tasks assigned to the system. Overcoming the hardware scalability limitations of shared memory was the main goal of these new systems. The increase in performance of standard microprocessors after the Reduced Instruction Set Computer (RISC) revolution, together with the cost advantage of large-scale parallelism, formed the basis of the 'Attack of the Killer Micros' [2]. The transition from emitted coupled logic (ECL) to complementary metal-oxide semiconductor (CMOS) chip technology and the usage of 'off the shelf' commodity microprocessors instead of custom processors for massively parallel processors (MPPs) was the consequence. The strict definition of MPP is a machine with many interconnected processors, where 'many' is dependent on the state of the art. Currently, the majority of high-end machines have fewer than 256 processors, with the most on the order of 10,000 processors. A more practical definition of an MPP is a machine whose architecture is capable of having many processors—i.e. it is scalable. In particular, machines with a distributed memory design (in comparison with shared memory designs) are usually synonymous with MPPs since they are not limited to a certain number of processors. In this sense, 'many' is a number larger than the current largest number of processors in a shared-memory machine.

2. STATE OF SYSTEMS TODAY

The acceptance of MPP systems not only for engineering applications but also for new commercial applications, especially for database applications, emphasized different criteria for market success, such as the stability of the system, continuity of the manufacturer and price/performance. Success in commercial environments is now a new important requirement for a successful supercomputer business. Due to these factors and the consolidation in the number of vendors in the market, hierarchical systems built with components designed for the broader commercial market are currently replacing homogeneous systems at the very high end of performance. Clusters built with components off the shelf are also gaining more and more attention. A cluster is a commonly found computing environment and consists of many PCs or workstations connected together by a local area network. PCs and workstations, which have become increasingly powerful over the years, can together be viewed as a significant computing resource. This resource is commonly known as clusters of PCs or workstations, and can be generalized to a heterogeneous collection of machines with arbitrary architecture.

At the beginning of the 1990s, while multiprocessor vector systems reached their widest distribution, a new generation of MPP systems came on the market, claiming to equal or even surpass the performance of vector multiprocessors. To provide a more reliable basis for statistics on high performance computers, the Top500 [3] list was begun. This report lists the sites that have the 500 most powerful installed computer systems. The best LINPACK benchmark performance [4] achieved is used as a performance measure to rank the computers. The Top500 list has been updated twice a year since June 1993. In the first Top500 list in June

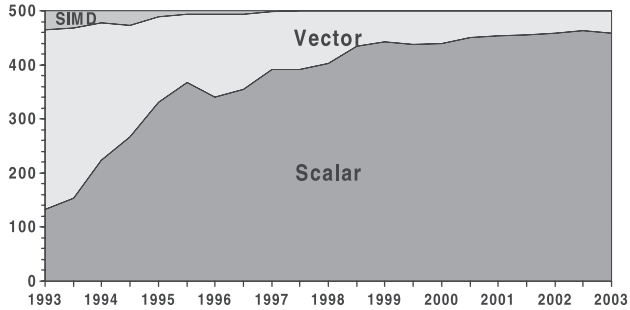

FIGURE 2. Processor design used as seen in the Top500.

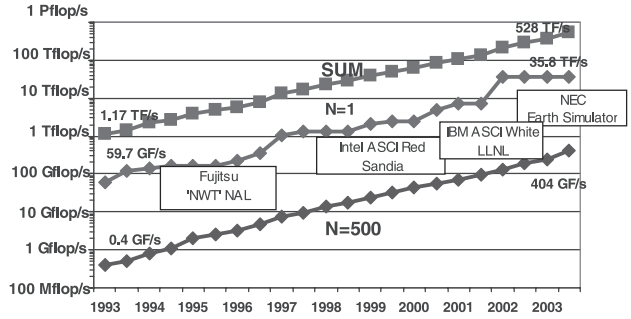

FIGURE 3. Performance growth at fixed Top500 rankings.

1993, there were already 156 MPP and SIMD systems present (31% of the total of 500 systems).

The year 1995 saw remarkable changes in the distribution of the systems in the Top500 according to customer types (academic sites, research labs, industrial/commercial users, vendor installations and confidential sites). Until June 1995, the trend in the Top500 data was a steady decrease in industrial customers, matched by an increase in the number of government-funded research sites. This trend reflects the influence of governmental high performance computing (HPC) programs that made it possible for research sites to buy parallel systems, especially systems with distributed memory. Industry was understandably reluctant to follow this path since systems with distributed memory have often been far from mature or stable. Hence, industrial customers stayed with their older vector systems, which gradually dropped off the Top500 list because of low performance (Figure 2).

Beginning in 1994, however, companies such as SGI, Digital and Sun began selling SMP models in their workstation families. From the very beginning, these systems were popular with industrial customers because of the maturity of the architecture and their superior price/performance ratio. At the same time, IBM SP systems began to appear at a reasonable number of industrial sites. While the IBM SP was initially intended for numerically intensive applications, in the second half of 1995 the system began selling successfully to a larger commercial market, with dedicated database systems representing a particularly important component of sales.

It is instructive to compare the growth rates of the performance of machines at fixed positions in the Top500 list with those predicted by Moore's Law. To make this comparison, we separate the influence of the increasing processor performance and of the increasing number of processors per system on the total accumulated performance. (To get meaningful numbers we exclude SIMD systems from this analysis as they tend to have extremely high processor numbers and extremely low processor performance.) In Figure 3 we plot the relative growth of the total number of processors and of the average processor performance, defined as the ratio of total accumulated performance to the number of processors. We find that these two factors contribute almost equally to the annual total performance growth—a factor of 1.82. On average, the number of processors grows by a factor of 1.30 each year and the processor performance by a factor of 1.40 per year, compared with the factor of 1.58 predicted by Moore's Law.

3. PROGRAMMING MODELS

Standard parallel architectures support a variety of decomposition strategies, such as decomposition by task (task parallelism) and decomposition by data (data parallelism). Data parallelism is the most common strategy for scientific programs on parallel machines. In data parallelism, the application is decomposed by subdividing the data space over which it operates and assigning different processors to the work associated with different data sub-spaces. Typically this strategy involves some data sharing at the boundaries, and the programmer is responsible for ensuring that this data sharing is handled correctly—i.e. data computed by one processor and used by another are correctly synchronized.

Once a specific decomposition strategy is chosen, it must be implemented. Here, the programmer must choose the programming model to use. The two most common models are:

- the shared-memory model, in which it is assumed that all data structures are allocated in a common space that is accessible from every processor and
- the message-passing model, in which each processor (or process) is assumed to have its own private data space and data must be explicitly moved between spaces as needed.

In the message-passing model, data are distributed across the processor memories; if a processor needs to use data that are not stored locally, the processor which owns that data must explicitly 'send' the data to the processor that needs it. The latter must execute an explicit 'receive' operation, which is synchronized with the send, before it can use the communicated data.

To achieve high performance on parallel machines, the programmer must be concerned with scalability and load balance. Generally, an application is thought to be scalable if larger parallel configurations can solve proportionally larger problems in the same running time as smaller problems on smaller configurations. Load balance typically means that the processors have roughly the same amount of work, so that no one processor holds up the entire solution.

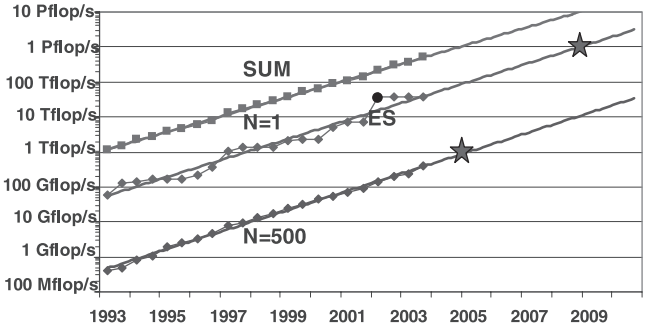

FIGURE 4. Extrapolation of Top500 results.

To balance the computational load on a machine with processors of equal power, the programmer must divide the work and communications evenly. This can be challenging in applications applied to problems that are unknown in size until run time.

4. FUTURE TRENDS

Based on the current Top500 data (which cover the last 13 years) and the assumption that the current rate of performance improvement will continue for some time to come, we can extrapolate the observed performance and compare these values with the goals of government programs such as the Department of Energy's (DOE) Accelerated Strategic Computing Initiative (ASCI), High Performance Computing and Communications and the PetaOps initiative. In Figure 4, we extrapolate the observed performance using linear regression on a logarithmic scale. This means that we fit exponential growth to all levels of performance in the Top500. This simple curve fit of the data shows surprisingly consistent results. Based on the extrapolation from these fits, we can expect to see the first 100 Tflop/s system by 2005. By 2005, no system smaller then 1 Tflop/s should be able to make the Top500 ranking.

Looking even further in the future, we speculate that, based on the current doubling of performance every year to 14 months, the first Pflop/s system should be available around 2009. Due to the rapid changes in the technologies used in HPC systems, there is currently no reasonable projection possible for the architecture of the Pflop systems at the end of the decade. Even as the HPC market has changed substantially since the introduction of the Cray 1 three decades ago, there is no end in sight for these rapid cycles of architectural redefinition.

There are two general conclusions we can draw from these figures. First, parallel computing is here to stay. It is the primary mechanism by which computer performance can keep up with the predictions of Moore's Law in the face of the increasing influence of performance bottlenecks in conventional processors. Second, the architecture of high-performance computing will continue to evolve at a rapid rate. Thus, it will be increasingly important to find ways to support scalable parallel programming without sacrificing portability. This challenge must be met by the development of software systems and algorithms that promote portability while easing the burden of program design and implementation.

5. TRANSFORMING EFFECT ON SCIENCE AND ENGINEERING

Supercomputers have transformed a number of science and engineering disciplines, including cosmology, environmental modeling, condensed matter physics, protein folding, quantum chromodynamics, device and semiconductor simulation, seismology and turbulence. As an example, consider cosmology—the study of the universe, its evolution and structure—where one of the most striking paradigm shifts has occurred. A number of new, tremendously detailed observations deep into the universe are available from such instruments as the Hubble Space Telescope and the Digital Sky Survey [5]. However, until recently, it has been difficult, except in relatively simple circumstances, to tease from mathematical theories of the early universe enough information to allow comparison with observations.

However, supercomputers have changed all of that. Now, cosmologists can simulate the principal physical processes at work in the early universe over space-time volumes sufficiently large to determine the large-scale structures predicted by the models. With such tools, some theories can be discarded as being incompatible with the observations. Supercomputing has allowed comparison of theory with observation and thus has transformed the practice of cosmology.

Another example is the DOE's ASCI, which applies advanced capabilities in scientific and engineering computing to one of the most complex challenges in the nuclear era—maintaining the performance, safety and reliability of the nation's nuclear weapons without physical testing. A critical component of the agency's Stockpile Stewardship Program (SSP), ASCI research develops computational and simulation technologies to help scientists understand ageing weapons, predict when components will have to be replaced and evaluate the implications of changes in materials and fabrication processes for the design life of ageing weapons systems. The ASCI program was established in 1996 in response to the administration's commitment to pursuing a comprehensive ban on nuclear weapons testing. ASCI researchers are developing high-end computing capabilities far above the current level of performance and advanced simulation applications that can reduce the current reliance on empirical judgements by achieving higher resolution, higher fidelity, three-dimensional physics and full-system modelling capabilities for assessing the state of nuclear weapons.

Parallelism is a primary method for accelerating the total power of a supercomputer. That is, in addition to continuing to develop the performance of a technology, multiple copies are deployed that provide some of the advantages of an improvement in raw performance but not all.

Employing parallelism to solve large-scale problems is not without its price. The complexity of building parallel supercomputers with thousands of processors to

solve real-world problems requires a hierarchical approach—associating memory closely with central processing units (CPUs). Consequently, the central problem faced by parallel applications is managing a complex memory hierarchy, ranging from local registers to far-distant processor memories. It is the communication of data and the coordination of processes within this hierarchy that represent the principal hurdles to effective, correct and widespread acceptance of parallel computing. Thus today's parallel computing environment has architectural complexity layered upon a multiplicity of processors. Scalability, the ability of hardware and software to maintain reasonable efficiency as the number of processors is increased, is the key metric.

The future will be more complex yet. Distinct computer systems will be networked together into the most powerful systems on the planet. The pieces of this composite whole will be distinct in hardware (e.g. CPUs), software (e.g. operating system) and operational policy (e.g. security). This future is most apparent when we consider geographically distributed computing on the Computational Grid [6]. There is great emerging interest in using the global information infrastructure as a computing platform. By drawing on the power of high-performance computing resources, geographically distributed, it will be possible to solve problems that cannot currently be attacked by any single computing system, parallel or otherwise.

Computational physics applications have been the primary drivers in the development of parallel computing over the last 20 years. This set of problems has a number of features in common, despite the substantial specific differences in problem domain.

(i) Applications were often defined by a set of partial differential equations (PDEs) on some domain in space and time.
(ii) Multiphysics often took the form of distinct physical domains, with different processes dominant in each.
(iii) The life cycle of many applications was essentially contained within the computer room, building or campus.

These characteristics focused attention on discretizations of PDEs, the corresponding notion of resolution = accuracy and solution of the linear and non-linear equations generated by these discretizations. Data parallelism and domain decomposition provided an effective programming model and a ready source of parallelism. Multiphysics, for the most part, was also amenable to domain decomposition and could be accomplished by understanding and trading information about the fluxes between the physical domains. Finally, attention was focused on the parallel computer, its speed and its accuracy, and relatively little attention was paid to I/O beyond the confines of the computer room.

The Holy Grail for software is portable performance. That is, software should be re-usable across different platforms and provide significant performance, say, relative to peak speed, for the end user. Often, these two goals seem to be in opposition to each other. Languages (e.g. Fortran, C) and libraries (e.g. Message Passing Interface (MPI) [7], Linear Algebra Libraries, i.e. LAPACK [8]) allow the programmer to access or expose parallelism in a variety of standard ways. By employing standards-based, optimized libraries, the programmer can sometimes achieve both portability and high performance. Tools (e.g. svPablo [9], Performance Application Programmers Interface (PAPI) [10,11]) allow the programmer to determine the correctness and performance of their code and, if falling short in some ways, suggest various remedies.

ACKNOWLEDGEMENTS

This research was supported in part by the Applied Mathematical Sciences Research Program of the Office of Mathematical, Information and Computational Sciences, US Department of Energy under contract DE-AC05-00OR22725 with UT-Battelle, LLC.

REFERENCES

[1] Moore, G. E. (1965) Cramming more components onto integrated circuits. *Electronics Mag.*, **38**(8), 114–117.
[2] Brooks, E. (1989) The attack of the killer micros. In *Teraflop Computing Panel, Supercomputing '89*, Reno, Nevada, 12–17 November.
[3] Top500 Report. (2004) http://www.top500.org
[4] Dongarra, J. J. (2003) *Performance of Various Computers Using Standard Linear Equations Software (Linpack Benchmark Report)*. Computer Science Technical Report CS-89-85, University of Tennessee. http://www.netlib.org/benchmark/performance.pdf.
[5] York, D. G. *et al*. (2000) The Sloan Digital Sky Survey: technical summary. *Astronom. J.*, **120**, 1579–1587.
[6] Foster, I. and Kesselman, C. (eds) (1998) *Computational Grids: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers.
[7] Snir, M., Otto, S., Huss-Lederman, S., Walker, D. and Dongarra, J. (1996) *MPI: The Complete Reference*. MIT Press, Boston.
[8] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammaring, S., McKenney, A. and Sorensen, D. (1999) *LAPACK Users' Guide* (3rd edn). SIAM Publication, Philadelphia.
[9] DeRose, L. and Reed, D. A. (1999) SvPablo: a multi-language architecture-independent performance analysis system. In *Proc. Int. Conf. on Parallel Processing (ICPP'99)*, Fukushima, Japan, 21–24 September, pp. 311–318.
[10] Browne, S., Dongarra, J., Garner, N., Ho, G. and Mucci, P. (2000) A portable programming interface for performance evaluation on modern processors. *Int. J. High Perform. Comput. Appl.*, **14**(3), 89–204.
[11] Dongarra, J., London, K., Moore, S., Mucci, P. and Terpstra, D. (2001) Using PAPI for hardware performance monitoring on Linux systems. In *Proc. Conf. on Linux Clusters: The HPC Revolution*, June 25–27, 23 pp. National Center for Supercomputing Applications (NCSA), University of Illinois, Urbana, EL.