Chapter 15

TRENDS IN HIGH-PERFORMANCE COMPUTING

Jack Dongarra
University of Tennessee and Oak Ridge
National Laboratory

1 HISTORICAL PERSPECTIVE

In last 50 years, the field of scientific computing has undergone rapid change—we have experienced a remarkable turnover of technologies, architectures, vendors, and the 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 [8]. Moore predicted that this trend would continue for the foreseeable future. In Figure 15.1, 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. These 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 (Kiloflop/s, thousands of floating-point operations per second), MFlop/s (Megaflop/s, millions of floating-point operations per second), GFlop/s (Gigaflop/s, billions of floating–point operations per second), TFlop/s (Teraflop/s, trillions of floating-point operations per second), and PFlop/s (Petaflop/s, 1,000 trillions of floating-point operations per second). This chart shows clearly how well Moore's Law has held up 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

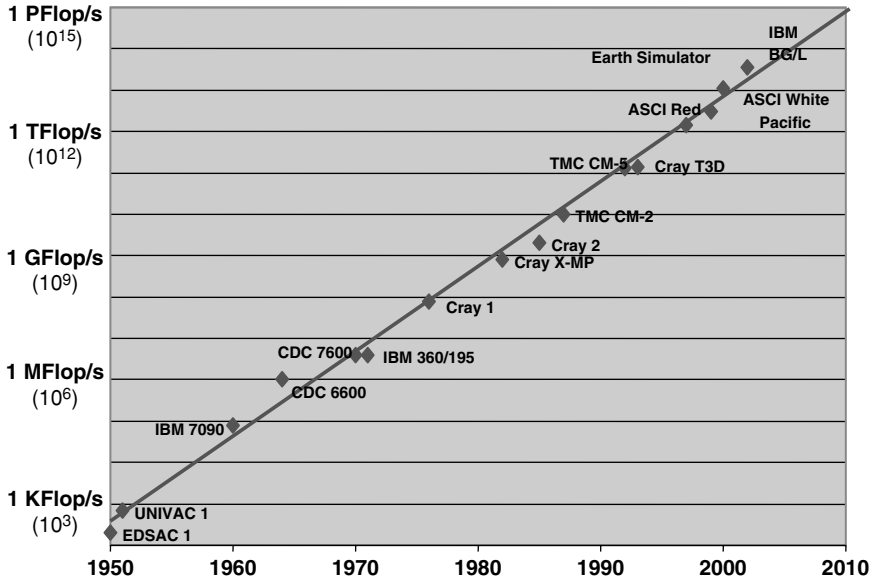

Figure 15.1. Moore's Law and peak performance of various computers over time.

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 computing environments became more important. Only those manufacturers that provided standard programming environments, operating systems, and key applications were successful in getting the industrial customers that 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 of performance of standard microprocessors after the Reduced Instruction Set Computer (RISC) revolution, together with the cost advantage of large-scale parallelism, formed the basis for the "Attack of the Killer Micros." 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 or MPPs was the

consequence. The strict definition of an 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 highest number 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—that is, 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 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 off-the-shelf components are also gaining more and more attention. A cluster is a commonly found computing environment consisting of many PCs or workstations connected together by a local area network. The 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 a cluster of PCs or workstations and can be generalized to a heterogeneous collection of machines with arbitrary architecture.

At the beginning of the 1990s, while the 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 [4] list was begun. This report lists the sites that have the 500 most powerful installed computer systems. The best LINPACK benchmark performance [9] 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 1993, there were already 156 MPP and SIMD systems present (31% of the total 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 of 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 (see Figure 15.2).

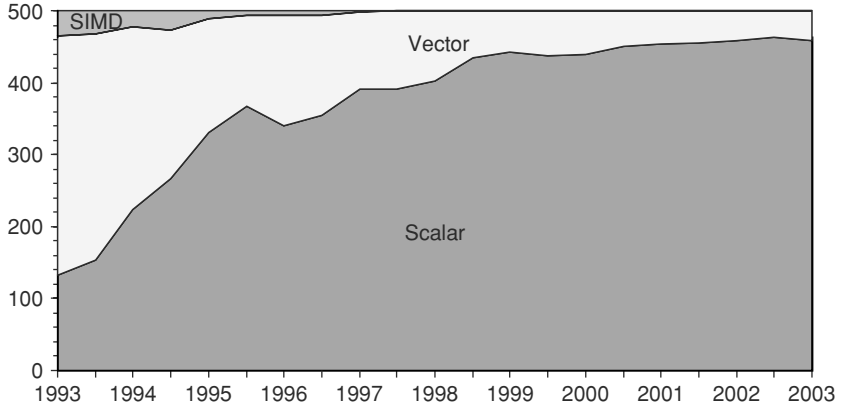

Figure 15.2. Processor design use as seen in the Top500.

Beginning in 1994, however, companies such as SGI, Digital, and Sun began selling symmetric multiprocessor (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 increasing processor performance and that of the increasing number of processors per system on the total accumulated performance. (To get meaningful numbers, we exclude the SIMD systems for this analysis, since these tend to have extremely high processor numbers and extremely low processor performance.) In Figure 15.3 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 has grown by a factor of 1.30 each year and the processor performance by a factor 1.40 per year, compared to the factor of 1.58 predicted by Moore's Law.

3 PROGRAMMING MODELS

The 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 subspaces. Typically, this strategy involves some data sharing at the boundaries, and the programmer is

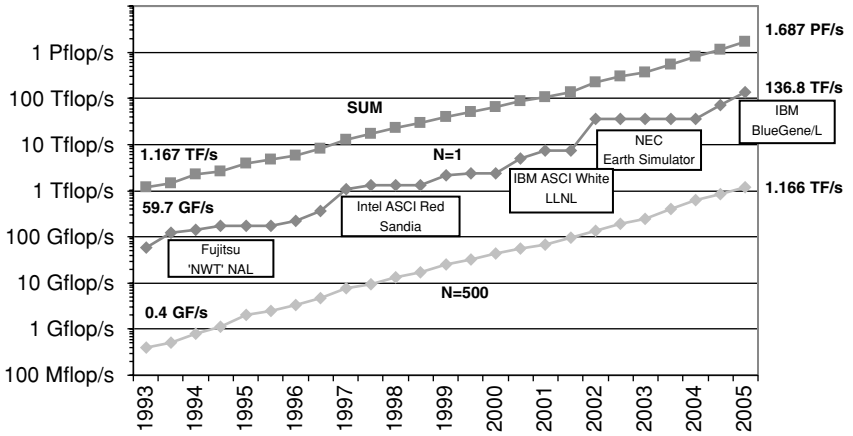

Figure 15.3. Performance growth at fixed Top500 rankings.

responsible for ensuring that this data sharing is handled correctly—that is, that data computed by one processor and used by another is 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 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 that owns those data must explicitly "send" the data to the processor that needs them. 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. To balance the computational load on a machine with processors of equal power, the programmer must divide the work and communications evenly. This division 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 Accelerated Strategic Computing Initiative (ASCI), High Performance Computing and Communications, and the PetaOps initiative. In Figure 15.4, 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 than 1 TFlop/s should be able to make the Top500 ranking.

Looking even farther into the future, we speculate that based on the current doubling of performance every twelve to fourteen months, the first PetaFlop/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 PetaFlops 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.

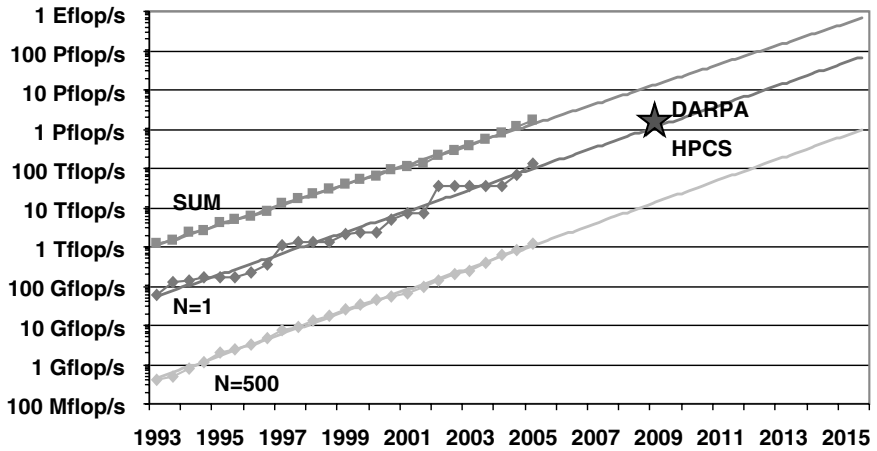

Figure 15.4. Extrapolation of Top500 results.

4.1 Grid Computing

Grid computing provides for a virtualization of distributed computing and data resources such as processing, network bandwidth, and storage capacity to create a single system image, providing users and applications seamless access to the collective resources. Just as an Internet user views a unified instance of content via the Web, a grid user essentially sees a single, large virtual computer.

Grid technologies promise to change the way organizations tackle complex computational problems. However, the vision of large-scale resource sharing is not yet a reality in many areas—Grid computing is an evolving area of computing, where standards and technology are still being developed to enable this new paradigm.

The early efforts in Grid computing started as projects to link US supercomputing sites, but now that initiative has grown far beyond its original intent. In fact, there are many applications that can benefit from the Grid infrastructure, including collaborative engineering, data exploration, high-throughput computing, and of course distributed supercomputing.

Ian Foster [12] defines a Grid as a system that

- coordinates resources that are not subject to centralized control . . . (A Grid integrates and coordinates resources and users that live within different control domains—for example, the user's desktop vs. central computing, different administrative units of the same company, or different companies—and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system.)
- . . . using standard, open, general-purpose protocols and interfaces . . . (A Grid is built from multipurpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. As discussed further below, it is important that these protocols and interfaces be standard and open. Otherwise, we are dealing with an application-specific system.)
- . . . to deliver nontrivial qualities of service. (A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands so that the utility of the combined system is significantly greater than that of the sum of its parts.)

At its core, grid computing is based on an open set of standards and protocols — e.g., Open Grid Services Architecture (OGSA) — that enable communication across heterogeneous, geographically dispersed environments. With grid computing, organizations can optimize computing and data resources, pool them for large capacity workloads, share them across networks, and enable collaboration.

A number of challenges remain to be understood and overcome in order for Grid computing to achieve widespread adoption. The major obstacle is the need for seamless integration over heterogeneous resources to accommodate the wide variety of different applications requiring such resources.

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 [2]. 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 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 Accelerated Strategic Computing Initiative (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. As a critical component of the agency's Stockpile Stewardship Program (SSP), ASCI research develops computational and simulation technologies to help scientists understand aging 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 aging weapons systems. The ASCI program was established in 1996 in response to the Administration's commitment to pursue a comprehensive ban on nuclear weapons testing. ASCI researchers are developing high-end computing capabilities far above the current level of performance, as well as advanced simulation applications that can reduce the current reliance on empirical judgments by achieving higher resolution, higher fidelity, 3-D physics, and full-system modeling 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 for 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 [10]. 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 that are 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 twenty years. This set of problems has a number of features in common, despite the substantial specific differences in problem domain:

1. Applications were often defined by a set of partial differential equations (PDEs) on some domain in space and time.
2. Multiphysics often took the form of distinct physical domains with different processes dominant in each.
3. 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 being equivalent to accuracy, and solution of the linear and nonlinear 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 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 reusable across different platforms and should 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] and Linear Algebra Libraries, i.e., LAPACK [3]) 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 [11] and Performance Application Programmers Interface (PAPI) [6]) allow programmers to determine the correctness and performance of their codes and, if falling short in some ways, to suggest various remedies.

ACKNOWLEDGMENTS

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

REFERENCES

[1] E. Brooks (1989): The Attack of the Killer Micros. Teraflop Computing Panel, Supercomputing '89, Reno, Nevada.

[2] Donald G. York et al. September (2000): The American Astronomical Society. The Sloan Digital Sky Survey: Technical Summary, *The Astronomical Journal*, *120*:1579–1587.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammaring, A. McKenney, and D. Sorensen (1999): *LAPACK Users' Guide – Third Edition*. SIAM Publication, Philadelphia.

[4] Top500 Report. http://www.top500.org/

[5] J. Dongarra, K. London, S. Moore, P. Mucci, and D. Terpstra (2001): Using PAPI for Hardware Performance Monitoring on Linux Systems. Terpstra. In *Proceedings of the Conference on Linux Clusters: The HPC Revolution*.

[6] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci (2000): A Portable Programming Interface for Performance Evaluation on Modern Processors. *International Journal of High Performance Computing Applications*, *14*(3), 189–204.

[7] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra (1996): MPI: The Complete Reference. MIT Press, Boston.

[8] G.E. Moore (1965): Cramming More Components onto Integrated Circuits. *Electronics 38*(8), 114–117.

[9] J.J. Dongarra (2003): Performance of Various Computers Using Standard Linear Equations Software (Linpack Benchmark Report). University of Tennessee Computer Science Technical Report, CS-89-85. http://www.netlib.org/benchmark/performance.pdf

[10] I. Foster and C. Kesselman (eds) (1998): *Computational Grids: Blueprint for a New Computing Infrastructure*. Morgan Kaufman.

[11] L. DeRose and D. A. Reed (1999): SvPablo: A Multi-Language Architecture-Independent Performance Analysis System. *Proceedings of the International Conference on Parallel Processing (ICPP'99)*, Fukushima, Japan.

[12] I. Foster, What is the Grid? A Three Point Checklist. *GRIDToday,* July 20, 2002.