CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **17**:1239–1270
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.892

Evaluating high-performance computers‡

Jeffrey S. Vetter[1,*,†], Bronis R. de Supinski[2], Lynn Kissel[2], John May[2] and Sheila Vaidya[2]

[1] *Oak Ridge National Laboratory, Oak Ridge, TN 37922, U.S.A.*
[2] *Lawrence Livermore National Laboratory, Livermore, CA 94551, U.S.A.*

SUMMARY

Comparisons of high-performance computers based on their peak floating point performance are common but seldom useful when comparing performance on real workloads. Factors that influence sustained performance extend beyond a system's floating-point units, and real applications exercise machines in complex and diverse ways. Even when it is possible to compare systems based on their performance, other considerations affect which machine is best for a given organization. These include the cost, the facilities requirements (power, floorspace, etc.), the programming model, the existing code base, and so on. This paper describes some of the important measures for evaluating high-performance computers. We present data for many of these metrics based on our experience at Lawrence Livermore National Laboratory (LLNL), and we compare them with published information on the Earth Simulator. We argue that evaluating systems involves far more than comparing benchmarks and acquisition costs. We show that evaluating systems often involves complex choices among a variety of factors that influence the value of a supercomputer to an organization, and that the high-end computing community should view cost/performance comparisons of different architectures with skepticism. Published in 2005 by John Wiley & Sons, Ltd.

KEY WORDS: high-performance computing; performance evaluation; computer architecture; parallel and distributed computing

1. INTRODUCTION

Supercomputing technology has made incredible progress over the last 50 years as shown by the exponential increase in processing power available to supercomputing users at Lawrence Livermore National Laboratory (LLNL) in Figure 1. As documented by various sources including Patterson *et al*. [1], Pfister [2], and Culler *et al.* [3], numerous technological and architectural improvements have contributed to this dramatic advancement. For example, from the late 1970s through the early 1990s, Cray supercomputers, using vector processors, provided much of LLNL's computing capability.

*Correspondence to: Jeffrey S. Vetter, Oak Ridge National Laboratory, Oak Ridge, TN 37922, U.S.A.
†E-mail: vetter@computer.org
‡This article is a U.S. Government work and is in the public domain in the U.S.A.

Published in 2005 by John Wiley & Sons, Ltd.

Received 15 February 2003
Revised 6 November 2003
Accepted 10 November 2003

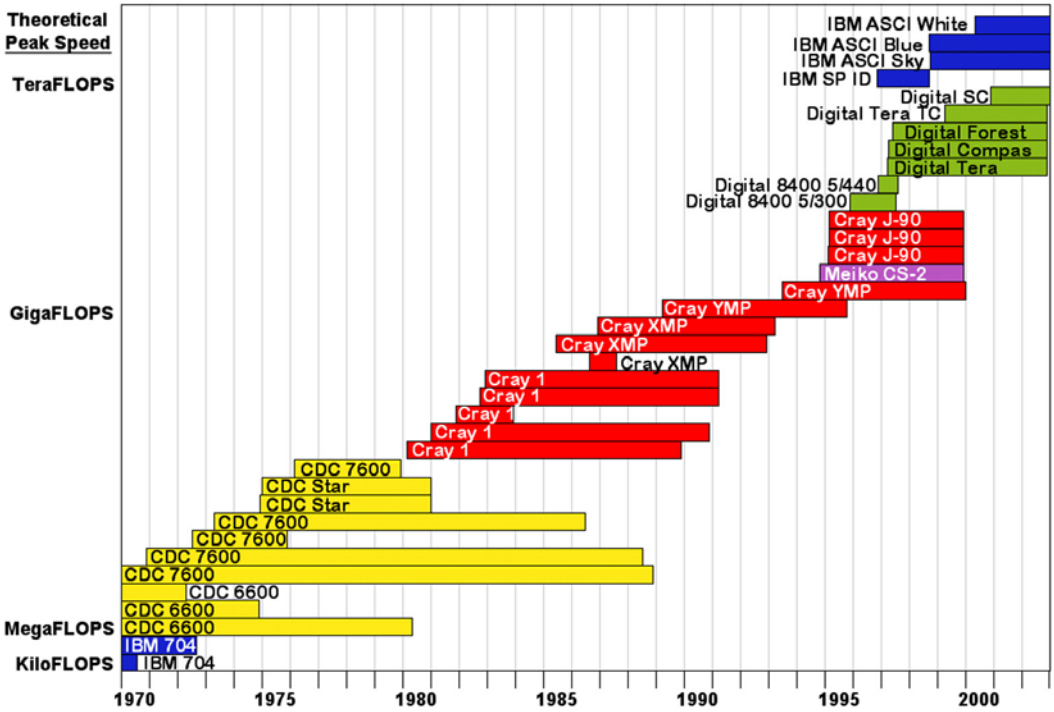

Figure 1. LLNL computer architecture and programming model history.

From the mid-1990s onward, several vendors furnished platforms to LLNL. These platforms were based on mass-produced microprocessors from vendors serving many markets and not limited to high-end computing (HEC). By using systems based on mass-produced microprocessors, LLNL has continued to meet its customer's computational demands, allowing our users to use the latest microprocessor technology while balancing it against cost.

Simultaneously, a majority of organizations in the scientific computing community migrated toward clusters of computing nodes, which were based on mass-produced microprocessors. Most recently, many HEC sites, including LLNL, have built large parallel computers from commodity workstations, connected by inexpensive networks—a strategy popularized by the Beowulf Project [4]. These commodity workstation clusters provide customers with low-cost and often dedicated access to significant computing resources.

Although specific technologies tend to dominate HEC during any particular era, there has always been an assortment of vendors and choices in the HEC marketplace. Many factors influence the value of a technology or computer system to an organization. Developers of scientific applications and the organizations that deploy computing systems for these applications have a vast array of requirements and constraints for successful HEC. Obviously, price and performance are two metrics cited as the

primary factors for these decisions; however, several other factors, such as software compatibility and facility requirements, also impact these decisions. This paper examines some of the many factors that affect a computer system's value. Although many factors can be described numerically, their correct values are often very difficult to measure. This difficulty is due, in part, to the rapidly changing technology in computer architectures, and the lack of practical performance prediction technologies. Furthermore, metrics, such as computer price, programming cost, and total cost of system ownership, are closely held, proprietary information, making longitudinal studies impractical. Thus, deciding which computer or technology is 'best' is often imprecise, and the 'right' answer often varies not only from year to year but also from user to user.

Nevertheless, it is useful to describe the important considerations in evaluating a computer system; this paper looks at several of these in Section 2. We then examine where a number of current and proposed HEC platforms lie in the complex space of capabilities and features in Section 3. Because technology changes so rapidly, the most important features of a computer system—those that drive its cost or define its ability to meet user needs—also change rapidly. Section 4 considers how current technology and future trends will influence the requirements and capabilities for HEC systems, while Section 5 discusses how other considerations, such as software compatibility, counter these trends. Finally, Section 6 explores how to evaluate different priorities for HEC platforms.

1.1. Advanced simulation and computing

Because user needs drive the purchase of a HEC platform, one must understand those needs before evaluating specific metrics. To put our analysis in perspective, we describe the computing requirements that have driven the acquisition of much of the HEC equipment at LLNL since the mid-1990s.

Since 1995, the U.S. Department of Energy (DOE) Advanced Simulation and Computing (ASCI) program has funded the development of new scientific applications and a series of HEC platforms. Several of these systems have been installed and used at LLNL. DOE established ASCI as part of its shift from test-based confidence to science- and simulation-based confidence in the nuclear stockpile. Now part of the DOE's National Nuclear Security Administration (NNSA), ASCI focuses on accelerating the development of simulation capabilities that far exceed what might have been achieved in the absence of the program.

Using advanced weapon codes and HEC, ASCI is creating simulation capabilities that incorporate more complete scientific models based on experimental results, past tests, and theory. The resulting predictive simulations enable the U.S. to assess and to certify the safety, performance, and reliability of nuclear systems. These simulation capabilities also help scientists understand weapons aging, predict when to replace components, and evaluate the implications of changes in materials and fabrication processes to the design life of the aging weapon systems. This science-based understanding is essential to ensuring that changes due to aging or remanufacturing will not adversely affect the stockpile.

To meet the Stockpile Stewardship Program's needs and requirements, ASCI has specific program objectives in performance, safety, reliability, and sustainability.

- Performance: create predictive simulations of nuclear weapon systems to analyze behavior and to assess performance without nuclear testing.
- Safety: predict with high certainty the behavior of full weapon systems in complex accident scenarios.

- Reliability: achieve validated predictive simulations sufficient to extend the lifetime of the stockpile, to predict failure mechanisms, and to reduce routine maintenance.
- Sustainability: use virtual prototyping and modeling to understand how new production processes and materials affect performance, safety, reliability, and aging and, thus, to help define the right configuration of production and testing facilities to manage the stockpile for the next several decades.

ASCI must solve progressively more difficult problems now that the U.S. has moved away from nuclear testing. This requires applications to achieve higher resolution, higher fidelity, three-dimensional physics, and full-system modeling capabilities to reduce reliance on empirical judgments. This level of simulation requires HEC far beyond our current level of performance. This requirement motivates several ASCI collaborations with industry to accelerate development of more powerful computing hardware and the associated software environment to support development of applications that use the new computing systems efficiently and productively. By 2005, the ASCI program is responsible for the following deliverables.

- Development of high-performance, full-system, high-fidelity-physics predictive codes to support weapon assessments, renewal process analyses, accident analyses, and certification.
- Stimulation of the U.S. computer manufacturing industry to create the HEC capability required by ASCI applications.
- Creation of a computational infrastructure and operating environment that makes these capabilities accessible and usable.

The ASCI Program has produced results that may well make it the most successful HEC program in U.S. history. Table I shows ASCI's planned and existing HEC platforms, including three of the top five systems on the TOP500 of the World's fastest computers for November 2002: White at LLNL and two Q systems at Los Alamos National Laboratory (LANL) (number 5 on the list, LLNL's MCR Linux cluster, is not an ASCI machine). These systems in turn have been instrumental in first-time, three-dimensional simulations involving components of a nuclear weapon during an explosion. Such accomplishments are based on the successes of other elements of ASCI research, such as scalable algorithms, programming techniques for thousands of processors, and unequaled visualization capabilities. These accomplishments offer confidence that the challenging goals and objectives facing the program can be achieved.

2. FACTORS INFLUENCING SUPERCOMPUTING VALUE

Ultimately, LLNL values supercomputers by the science they enable. Arguably, only two metrics matter: Idea-to-Solution (ITS) and Total Cost of Ownership (TCO). ITS is the time from when a user decides to solve a problem on a given platform until that user has an answer to that problem. TCO includes not only the purchase price of the platform but also the related (and sometimes hidden or overlooked) costs of using that platform. ITS and TCO are both difficult to measure; thus, purchases are generally based on some subset of these factors.

Realistically, several factors influence ITS, such as execution time, programmability, reliability, and solution quality. Clearly, a fundamental criterion is performance—the time to run the application

Table I. Deployed and planned ASCI computer systems (Tflops).

System	Host	Installation date	Vendor	Number of processors	Peak floating point rate (Tflops)	Total memory (TB)	Processor type	Footprint (ft^2)	Power (MW)	Construction required
Red (upgrade)	SNL	1997	Intel	9298	3.1		Pentium II Xeon	2500		Use existing
Blue Pacific	LLNL	1998	IBM	5808	3.89	2.6	PowerPC 604e	5100	0.6	Use existing
Blue Mountain	LANL	1998	SGI	6144	3.072		MIPS R10000	12 000	1.6	Use existing
White	LLNL	2000	IBM	8192	12.3	8	POWER3-II	10 000	1.0	Expand existing
Q	LANL	2002	HP Compaq	8192	20.5		Alpha EV 68	14 000	1.9	New
ASCI Linux Cluster	LLNL	2003	IBM	1920	9.2		Pentium 4	1200	0.48	Use existing
Red Storm*	SNL	2004	Cray	10 368	40		AMD Opteron	3000	2.1	New
Purple*	LLNL	2004	IBM	12 220	100	50	POWER5	12 000	4.5	New
BlueGene/L*	LLNL	2004	IBM	131 000	180/360	16/32	PowerPC 440	2500	1.8	Use existing
Earth Simulator	ES	2002	NEC	5120	40	10	NEC SX-5 descendant	34 000	10	New

*Planned systems.

that provides the answer. However, other factors can impact ITS as significantly as performance: programmability—the time to the first correct solution; compatibility—the time to port an existing application to the platform at acceptable performance; and reliability—the time lost due to platform component failures and downtime. Several system aspects influence these ITS components including the software environment, the platform's similarity to those used previously by application programmers and designers and the level of technical support that the platform vendor provides. All factors, including performance, are formidable to measure a priori, when purchase decisions are made. Thus, most procurements must speculate about these factors to some degree.

Like ITS, several factors determine TCO. The platform's initial purchase price is the most visible and most frequently cited. However, other factors such as the building to house it, the power to operate and cool it, the salaries of support staff, and auxiliary systems (e.g. file systems) required to use it effectively can dominate TCO. Like those of ITS, TCO components are proprietary, or hidden in a web of indirect expenses.

The remainder of this section looks at some of the components of ITS and TCO and metrics that estimate those components in more detail. These metrics evaluate how well a system provides a high-performance, stable, familiar computing environment to our users at a reasonable cost to our sponsors. Taken together, they drive our analysis of HEC procurement alternatives.

2.1. Performance

Although numerous performance measures exist, the HEC community has yet to define one that is universally accepted. For over 40 years, the community has proposed, implemented, refined, and eventually ignored a large number of performance metrics and benchmarks [5]. In this section, we define and evaluate several important performance metrics. Clearly, real application performance is a critical metric in any evaluation, but since real applications place a broad spectrum of demands on architectures, it has proven impossible to capture overall performance through any one metric.

Several methods exist for measuring and reporting HEC performance. Generally, users are interested in reducing execution time of a specific task or in increasing the throughput of tasks (i.e. in reducing the execution time of a set of tasks). Execution time can be defined in several ways. We use the wall-clock time or the elapsed time, which is the latency to complete a specific task including all sources of latency (e.g. I/O activities, OS overheads, memory accesses). The set of tasks used to calculate this latency is the system workload. There are generally five levels of tasks for evaluating HEC systems: real applications, scripted applications, kernels, toy benchmarks, and synthetic benchmarks.

The increasingly competitive market for HEC systems has led companies to invest heavily in improving the performance on sets of applications that users measure. These efforts often lead to contention in the selection of benchmarks for evaluating cost–performance tradeoffs. The Standard Performance Evaluation Corporation (SPEC) [6] created a standardized benchmark suite and the rules for reporting its performance in the late 1980s. SPEC has since refined this suite and produced additional versions that target communities for which other system workloads are most relevant. These suites, and others like them [7,8], are useful; nonetheless, most platform RFPs include performance targets for applications specifically of interest to the purchaser.

In the HEC community, four metrics have emerged as widely quoted measures of system performance: peak performance rate, sustained performance rate, time-to-solution, and scalability. Several additional factors influence these performance metrics, as described below.

2.1.1. *Performance metrics*

The execution time required for a computing system to solve a specific task is commonly referred to as the *time to solution* (TTS). In the context of HEC, this metric is the wall-clock time from the start of the application experiment until the solution is produced: $T = T_{\text{stop}} - T_{\text{start}}$. TTS is straightforward to calculate.

With this perspective on application performance, we must realize that all system technologies contribute to TTS, as discussed in Section 4. Any improvement in the performance of one system technology is limited by the amount of time the application can exploit that technology. Specifically, TTS for real applications has several constituent parts: computation, communication, and I/O. In close analogy to Amdahl's Law [9], the overall improvement in TTS due to improvements in one specific technology is limited by the scope of that technology's contribution to TTS. For example, if an application's original TTS is weighted with 90% of its time spent in communication, any improvement in computation technology is limited to an improvement in TTS of 11% (1/0.9). The important consequence of this observation is that a widely quoted performance measure such as peak flop rate may be largely irrelevant to the performance of some real-world applications. Moreover, the system technology that governs overall performance can vary between machines and between applications.

All HEC systems have a maximum number of floating point operations per second that the architecture can compute, often referred to as the *peak floating point rate* (PEAK). Table I shows PEAK is easy to calculate, even for future HEC platforms, given the instruction latency and throughput information for its microprocessor. For example, IBM's POWER3-II microprocessor can calculate two fused multiply add (FMA) instructions on every clock cycle—one FMA in each floating point unit. With a clock rate of 375 MHz, the POWER3-II processor has a PEAK of 1500 Mflops. Further, we can compute PEAK for a cluster of microprocessors by simply multiplying by the number of processors, ignoring communication between processors or processors and their local memory.

Calculating other performance metrics is not as simple as calculating PEAK. *Sustained floating point rate* (SUSTAINED) provides a more realistic measure of the rate at which applications effectively use the HEC platform. SUSTAINED is the number of *actual* floating point operations (flops) performed by a workload divided by the execution time required for those operations. *Percentage of PEAK* (%PEAK)—SUSTAINED divided by PEAK—is often cited as the efficiency of an application or a system workload on a target architecture. SUSTAINED can be measured in several ways. Users can manually count the flops in the code and then time the code region. Alternatively, microprocessor hardware performance counters can be used to calculate flops. The difference between these choices can be substantial, primarily due to the hazards of using performance counters. Counter definitions vary by platform—for example, some platforms count FMAs as one flop, others as two. Further, a complicated operation such as division or square root may be counted as multiple flops even though users consistently view them as a single flop. Different compilers may generate different flops for the same user-level code; if they run in the same time, is the one with more flops better? More distressingly, algorithmic changes can increase flops, and improve SUSTAINED while increasing TTS, the real metric of interest.

With current applications running on scalable clusters of multiprocessors with hundreds, or thousands of processors [10,11], *scalability* of applications and platforms is another important property. By scalability, we mean the ability of a parallel algorithm on a parallel architecture to effectively utilize

an increasing number of processors [3,12,13]. In particular, *speedup* is defined as $T(1)/T(n)$, where $T(1)$ is the single processor timing and $T(n)$ is the n processor timing.

Parallel efficiency is defined as speedup/n (expressed as a percentage). A parallel efficiency of 100% corresponds to the ideal situation that n processors produce the result n times faster—perfect scalability. On occasion, it is desirable to base the speedup and parallel efficiency calculation on other than a single-processor baseline, for instance, when the problem does not fit in the memory of one processor, or performs uncommonly well or poorly in that case.

Other important metrics measure system capabilities beyond its floating-point performance. *Latency* measures the time for the system to complete a minimal-size operation. Memory latency, network latency and IO latency can be important factors in TTS. *Bandwidth* measures the system's ability to perform sustained operations. Memory bandwidth, network bandwidth or IO bandwidth frequently dominates TTS on a given platform for many system workloads.

2.1.2. Observations

All of these performance metrics have strengths and weaknesses. PEAK is a commonly accepted performance metric since it does not require measurement and, thus, it is easy to calculate. However, PEAK is not fully correlated with TTS since %PEAK varies with the platform and the system workload. By contrast, SUSTAINED, %PEAK and scalability require that users select a workload along with their respective input datasets and configuration. For all three, users must estimate the execution time on the platform through empirical measurement, simulation, or analytical modeling—an increasingly difficult task given the complexity of modern architectures and applications. Further, the selected workload, datasets and configuration are often minimized to reduce simulation or modeling effort. In addition, factors such as application development that occur after the platform purchase or the classified or confidential nature of the real datasets and configuration lead to more complexity.

As already discussed, counting flops complicates measuring and estimating SUSTAINED and, thus, %PEAK. Lack of advanced architectural feature standardization further complicates measuring %PEAK. For example, many processors, such as the IBM PowerPC series and the Intel Itanium series, implement FMAs. FMAs are valuable in technical computing because they are common operations in many matrix and vector calculations. Unfortunately, FMAs are not part of the IEEE floating-point standard, so vendors implement them differently, which, in turn, leads to different numerical results across platforms. Although the FMA can be more precise than distinct multiply and add operations, this lack of standardization forces users to disable FMAs when reproducibility across platforms is required. Nevertheless, FMAs are used in calculating PEAK. Ergo, for any application that must disable FMAs, %PEAK is immediately reduced to 50% of the advertised PEAK, even before accounting for the impact of the latency and bandwidth of other system components.

Given the ambiguity of these metrics, vendors often try to reduce procurement activity risk by focusing on unambiguous and reasonably easy-to-calculate metrics. Because procurement activities often contain performance specifications [14], PEAK helps to reduce risk because as a theoretical metric, it is invariant to workloads, compilers, memory, communication, and I/O subsystems.

2.2. Acquisition cost

Another important HEC system aspect is cost. When comparing HEC platforms, it is natural to ask which offers the best performance for the money. An organization buying a computer can answer this

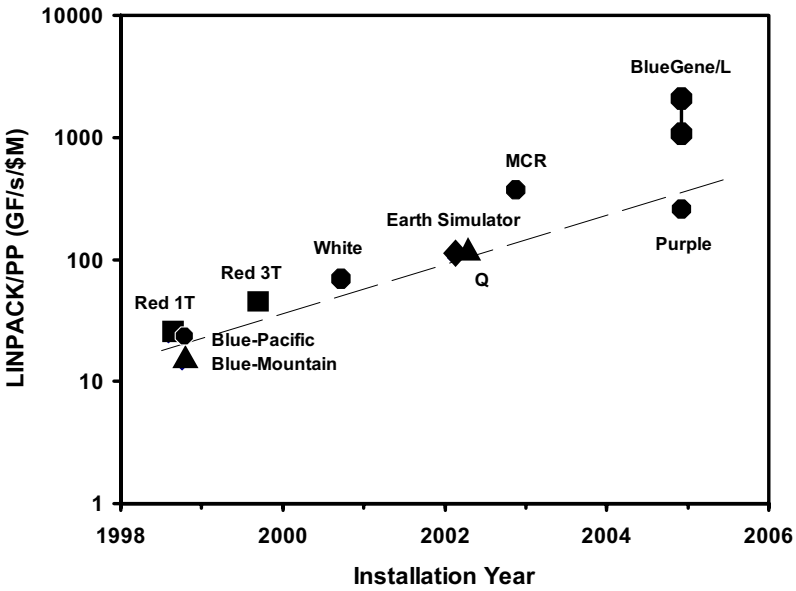

Figure 2. A rough estimate of cost/performance is the ratio of the performance on a standard benchmark, such as LINPACK, to the published acquisition cost (PP).

question relatively easily in terms of the purchase price (PP) if it has received bids from competing vendors. The buyer can then estimate TCO based on the bids, accounting for power consumption and other facilities costs based on details of the proposals. Differences in the details of the proposed systems' hardware and software can make comparisons inexact, and the buyer may need to rely on estimates of the systems' performance on the intended workload. Nevertheless, the comparison is relatively straightforward since the systems are designed for the same purpose, for purchase by the same buyer at the same time.

It is much more complicated for the HEC community as a whole to make cost/performance comparisons because this involves comparisons of classes of technology rather than individual systems. It is easy to estimate the performance per unit cost by dividing a machine's published PEAK or its performance on a well-known benchmark, such as LINPACK, by its announced price (that is, the publicly stated price) (see Figure 2). However, this estimate is misleading, primarily because true costs of the systems are uncertain, technology costs decline precipitously over time, and the difficulty associated with the aforementioned performance metrics issues.

2.2.1. Uncertainties in announced prices

The PP of an HEC platform often bears little connection to the buyer's actual cost to field a stable, fully supported system. At the computer market's highest end, vendors oftentimes offer a discount in

exchange for the prestige of building one of the world's fastest computers. Vendors may also subsidize the cost to gain experience with new technology before introducing it to a broader product line. Thus, PP may be lower than the publicly visible cost of a competing platform. Buyers, for their part, sometimes pay for the development of technology that will eventually appear in future products [15]. This early support, which lowers the PP of later systems, can be through either a research contract with is own announced price and deliverables or a collaborative arrangement in which each party devotes staff time and expertise to solve a problem of mutual interest. PP may also be more complicated than a simple machine purchase. For example, it may pay for a lease rather than a purchase, or the vendor may agree to upgrade certain system components over time.

As an example of HEC purchase complexity, consider the November 2002, agreement between IBM and the DOE to build the ASCI Purple and BlueGene/L (BG/L) computers [16]. The contract's total value is $290 million, of which $267 million will be paid to IBM, and the rest to other suppliers. The computers will have PEAKs of 100 Tflops s^{-1} for ASCI Purple and 180–360 Tflops s^{-1} for BG/L[§], both scheduled for delivery in late 2004. One might assume that the costs of the two machines will be proportional to their PEAK. In fact, BG/L is likely to be considerably cheaper than Purple—in terms of both PP, because BG/L uses simpler processors and less memory and in terms of TCO because BG/L uses less floor space and less power as well as having a lower PP [17]. However, BG/L targets a narrower scope of applications than Purple. This shows that simply dividing the cost by a single performance measure can be misleading: BG/L clearly offers better PEAK per dollar, even though it targets a bounded set of applications when contrasted with ASCI Purple.

This contract also contained two early delivery platforms not described in the contract's press releases. One is a Linux cluster, and the other will be a scaled-down version of Purple, with fewer processors and an earlier generation of the processor and interconnect. Thus, the $290 million announced price includes four large platforms, not two, that will be delivered over about two years (early 2003 to late 2004). IBM and LLNL have not announced separate prices for each platform, however BG/L accounts for less than $100 million of the cost. The announced price does not include the cost of several collaborations that DOE completed with IBM and other vendors to develop new interconnect, storage, and software technology, some of which will be used in Purple and BG/L. In addition to the contract for the purchase of BG/L, for example, $21 million between IBM and LLNL help to develop BG/L technology. Two other contracts, worth about $15 million each, with IBM and Quadrics developed interconnect and switching technology used in ASCI Q, the ASCI Linux Cluster and other machines. Current work is developing new file system technology under a contract worth about $8 million with Cluster File Systems. This technology may be used with BG/L.

The announced cost also excludes the hundreds of staff years that DOE and laboratory staff will spend on planning, collaborative development, testing, system integration, and all of the other effort necessary to field working systems. For example, DOE's personnel costs in support of ASCI Purple over the lifetime of the system, including system support and system software but not applications development, will likely exceed $100 million.

[§]The range for BG/L performance estimates arises from the question of whether one processor per node will be dedicated to communication rather than being available for computation.

2.2.2. *Differing capabilities*

Evaluating performance for the purpose of cost/performance comparisons raises further questions. As described in Section 2.1, PEAK and other flop-based metrics are an incomplete description of a computer's capabilities. Other important measures include memory capacity, secondary and tertiary storage capacity, memory bandwidth, and interconnect latency and bandwidth. In well-designed systems, these parameters should be chosen to produce a computer with a good balance of capabilities for the intended workload.

The range of reasonable choices for the various platform components is quite broad, so one cannot expect the system parameters for different computers to vary in proportion to their PEAK. For example, ASCI White's PEAK is 12.3 Tflop s^{-1} and it has 8 TB of main memory [18] while the Earth Simulator (ES) has a PEAK of 40 Tflop s^{-1} but only 10 TB of main memory [19]. Comparing these systems' cost/performance ratios based only on PEAK would ignore the value of the extra memory per flop in ASCI White. Our experience shows that as the processing capability increases, so does the memory capacity requirements of many traditional scientific applications.

The same consideration applies more subtly to software and support costs. Many HEC purchases include software development tools and run-time software, such as compilers, debuggers, message-passing libraries, mathematical libraries, job scheduling software, and file system software. The purchase may also include vendor technical support and on-site support personnel. These extras add nothing to PEAK. Nevertheless, they are essential because they directly impact the programmability and compatibility components of ITS. Thus, comparisons of the cost per PEAK of different systems should exclude costs that do not contribute to PEAK.

This point is especially important when comparing user-built systems and Linux clusters to traditional HEC platforms. Clusters derive cost savings partly from the use of open-source software and from the lack of vendor technical support. Organizations like universities, which have access to inexpensive labor, can often provide their own support and software development services at lower cost than vendors can, so the choice of a cluster or user-built system is sensible. The same system would cost more to operate at a company or a government laboratory, where labor costs are higher. For these buyers, vendor-supported systems may be more economical in the long run, even though PP is higher.

2.2.3. *Different purchase dates*

A final consideration in comparing prices is the date of purchase and delivery. The rapid decline in computer hardware costs means that systems delivered roughly a year or more apart will have different cost/performance ratios. To properly account for declining costs, any discussion of prices and machines must consider purchase or delivery dates. Since cost declines (or compute power per unit cost increases) roughly by half every 18 months, we apply a discount of 4% per month between the system's delivery date and a reference date to compute indexed costs. For example, ASCI White, delivered in September 2000 at a PP of $85 million, has an indexed PP of $27 million in February 2003.

2.2.4. *Toward a rational cost/performance measure*

Clearly, estimating cost/performance ratios based on PP and PEAK only gives highly approximate results. However, some rational cost measure is necessary for comparisons of computing technology. A rational cost measure should:

- make all costs visible (this may preclude the use of PP, since it does not account for hidden costs and discounts, which vary between buyers);
- include only the components that impact the measured functionality;
- be indexed so as to apply the same delivery date to different systems.

Unfortunately, following these guidelines exactly is often impractical. One cannot always define which components contribute to a performance measure, and separate pricing information for them may not be available. Nevertheless, following this guidance can provide more accurate cost/performance estimates. However, PP will likely continue as the most common estimate of cost, just as PEAK will likely continue as a shorthand description of a system's performance.

For estimates of the cost per PEAK, we propose the list price (LP) of a single node as a cost measure. This includes minimal secondary storage, no interconnect, and no extra-cost software. It includes the main memory bus and main memory. Systems should be configured with comparable amounts of main memory. While LP may not be the actual price that all customers will pay, it is an objective measure that avoids hidden costs and discounts. Consider, for example, LP for two current supercomputer nodes, the IBM RS6000/SP and the Cray/NEC SX-7. Table II lists price, PEAK, and estimated LINPACK data for the two systems. Of course, LINPACK does not represent real workloads, and delivered performance on complex applications is often far less. However, the data show that even when the vector system is nearly three times as efficient, it still costs more than twice as much per sustained Gflops s^{-1} on this measurement.

We should emphasize that we propose this measure only for the purpose of evaluating competing technologies for carrying out floating-point calculations. Obviously, different metrics would apply to other subsystems. Moreover, considerations other than monetary cost might make one system more attractive than another as discussed in the next section.

2.3. Operating costs

The costs of operating modern HEC platforms include prodigious power and cooling requirements, and a substantial support staff. They are typically part of a larger computing center, so that the total facilities challenge is compounded by the needs of large file systems and data archives, high-end visualization and other data services, extensive communications infrastructure and networking equipment, security and other infrastructure services, and the workplace needs of the support organization. Further, these diverse costs are typically shared across multiple platforms so determining their TCO contribution is not simple.

Operating costs (OCs) are an important challenge to the ownership of modern HEC platforms. By OCs, we mean those costs associated with the operation of the platform, other than the PP of the platform or the construction costs (CCs) of preparing the facilities to house it. The TCO for HEC platforms will come as a rueful surprise to institutions that have not carefully analyzed the OCs prior to a major acquisition. At LLNL, staff costs (about 60%), facility expenses (about 20%), and electrical power (about 10%) dominate the total non-capital, non-construction portion of HEC TCO. Combined, these three items account for about 90% of the non-capital, non-construction portion of our HEC TCO.

OCs are generally shared over a wide set of platforms—between ASCI and other programs, LLNL currently operates nine platforms with more than 100 CPUs. These platforms use diverse

Table II. List prices and performance for two supercomputer nodes, as of late 2002. LINPACK data for IBM are from IBM tests; estimated LINPACK data for NEC are based on approximately 95% efficiency of several NEC SX-6 systems reported on the TOP500 Web site.

System	List price (U.S.$)	PEAK (Gflops s^{-1})	LINPACK (Gflops s^{-1})	Cost per Gflops s^{-1} (U.S.$)	Configuration
IBM pSeries 690	860 000	83.2	28	31 000	16 × Power4, 32 GB memory
Cray/NEC SX-7	2 200 000	35.3	34	65 000	4 × SX-7, 32 GB memory

technologies with diverse power, maintenance and system administration requirements. In addition, the computer center also operates a number of smaller systems, visualization servers and data archives. Clearly, proportionally dividing LLNL OCs across its HEC platforms based on PEAK is inaccurate. Alternatively, adding a new system is less expensive than adding a first system due to economies of scale—existing buildings and staff greatly simplify the process, as demonstrated by the recent MCR acquisition.

Often, substantial CCs (for new facilities or extensive modifications to existing facilities) must be borne in order to house new HEC platforms. For example, the Metropolis Center for Modeling and Simulation [20] was built to house ASCI Q and future LANL platforms. This state-of-the-art HEC facility features a 43 500 ft^2 (4040 m^2) machine room floor, 7.1 MW of electrical power (expandable to 30 MW), and 3600 tons (13 MW) of cooling capacity. Other examples include the facility in Yokahama dedicated to ES (34 000 ft^2/3200 m^2 machine floor), and the Terascale Simulation Facility (two 24 000 ft^2/2200 m^2 machine floors) under construction to house ASCI Purple and future LLNL systems. Assigning CCs, like OCs, to specific platforms is difficult since these facilities will serve several platform generations. If an existing building can house a new platform, are its CCs zero or are the CCs of the existing platforms decreased?

The costs for large facilities and high power use impact commercial computing as well as scientific HEC. In commercial server colocation, a third party provides multiple companies with floor space, power, cooling, communications services, and security for their computers. Facilities with tens of thousands of square feet of floor space, and power and cooling capacity in the many megawatt range are common [21]. Although they do not house systems of the same class as LLNL's HEC platforms, the facilities are very similar. Consequently, we expect commercial systems to follow trends that minimize footprint and power consumption, similarly to BG/L [17].

2.3.1. Physical characteristics of platforms

Table I lists the physical characteristics of recent and planned ASCI platforms at LLNL, along with those for ES. The power number listed is the electrical power used directly by the

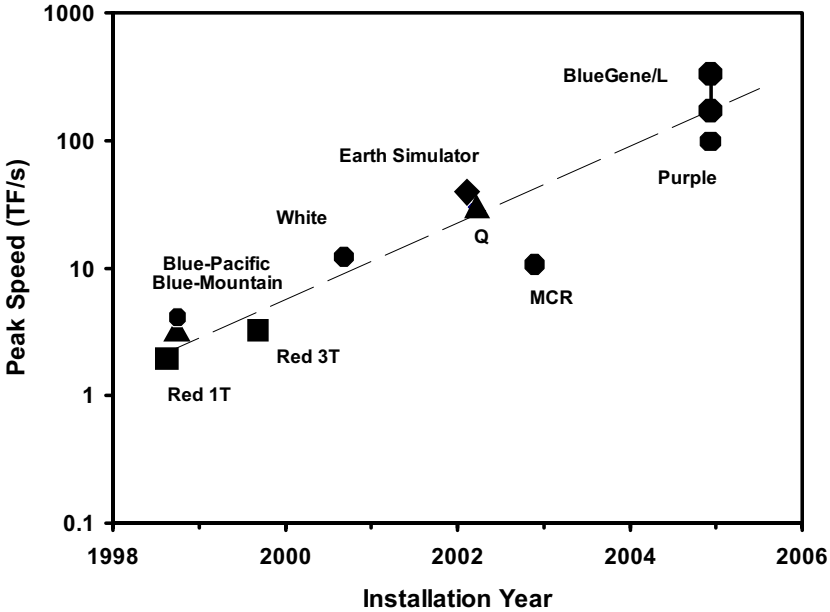

Figure 3. Comparison of the peak speed of ASCI platforms, MCR, and the ES.

computing equipment. The total power required for cooling and other support equipment increases this number by about 70%.

As would be expected, computing capability increases with time for these HEC platforms. Figure 3 shows the PEAK of the major ASCI platforms and ES, plotted against the year of installation. The dashed curve represents a doubling time of 12 months, a remarkable accomplishment, sustained over a 6-year period. Viewed on this scale, the performance of ES is substantially in line with the overall trend of the ASCI platforms. Although it is on the high side of the curve, as Table V in Section 3 illustrates, its performance is more in line with the other platforms when the performance is scaled by PP.

While the platforms' PEAKs follow a similar trend, we see more differences in their space and power requirements. Compared with LLNL's platforms, ES has a significantly larger footprint, and uses considerably more power.

Table I shows that the size of an HEC platform significantly impacts the need for facilities construction. For Blue-Pacific and BG/L, the smallest systems listed, existing computing facilities were used with relatively minor modification. White, a somewhat larger system, required a substantial modification to a building through construction of a 'popout' to expand the building. New facilities, at substantial construction costs, were built to house ES and Purple.

Table III. Metrics for a facilities-based comparison of LLNL ASCI platforms and the ES.

	ASCI Blue-Pacific	ASCI White	ES	ASCI Purple*	BG/L
Installation date	10/1998	9/2000	2/2002	~12/2004	~12/2004
Compute-power efficiency (Gflops s^{-1} kW^{-1})†	6	12	4	22	100–200‡
Compute-space efficiency (Gflops s^{-1} m^{-2})	8	13	13	91	750–1500
Memory-space efficiency (GiB m^{-2})§	6	9	3	45	70–140¶

*Target specifications.
†Gigaflops = 10^9 floating-point operations per second.
‡System operating in either communications co-processor mode, or virtual node mode.
§GibiBytes = 2^{30} bytes = 1 073 741 824 bytes [23].
¶Memory per node of either 256 MiB, or 512 MiB.

The data in Table I imply significant energy costs. Assuming a rate of \$0.6 million MW-year^{-1}, the annual power bill for the facilities housing these systems range from about \$0.6 million year^{-1} for Blue-Pacific to about \$10 million year^{-1} for ES. This is enough electricity to power about 500–8000 average U.S. households¶.

2.3.2. *Facility efficiency metrics*

Considering the importance of space and power in the TCO, non-traditional metrics should become standard for characterization of HEC platforms. Table III reveals that LLNL's ASCI platforms (Blue-Pacific, White and Purple) have demonstrated continuous improvements in these metrics, and are considerably more 'facility friendly' than ES. Moreover, these metrics emphasize the dramatic improvement in effectiveness of the BG/L design. This comparison helps explain the NNSA's keen interest in BG/L as it promises to be a strategy that effectively scales to platform sizes not obtainable with current architectures.

2.4. Portability and compatibility

As mentioned in Section 1, many LLNL applications have a lifetime measured in decades. Until the early 1990s, most platforms and development environments lasted for at least that duration. However, the competitive marketplace for HEC systems has given compatibility and

¶The average residential household in the U.S. consumes about 11 000 kWh year^{-1} [22].

Table IV. Metrics for HEC system evaluation.

Metric	Description	Ease of measure (unequivocal)	Comments
ITS	Idea to solution	No	Sum of multiple development and execution time components
TTS	Time to solution	No	Actual execution time of application
PEAK	Peak performance	Yes	Not fully correlated to TTS
SUSTAINED	Sustained performance	No	Flop count divided by TTS
%PEAK	Percentage of peak	No	SUSTAINED divided by PEAK
TCO	Total cost of ownership	No	Sum of multiple cost components
PP	Purchase price	No	Costs hidden by development agreements, discounts, marketing value
OC	Operating cost (power, cooling, floorspace)	No	Shared across multiple platforms concurrently
CC	Construction cost	No	Shared across multiple generations of platforms
Portability and compatibility	Effort required to move applications to new architecture	No	

portability increased significance. Applications must rely on conservative, open, and widely supported standards. For example, many LLNL applications employ MPI, OpenMP, ANSI C, ANSI C++, and FORTRAN 90. Many vendors provide these in their development environments. Furthermore, since many of our applications are under development, correctness and performance tools must support these language and library standards. As evidenced by the recent ASCI Purple procurement [14], DOE requires vendors to provide these capabilities. We are finding that many tool vendors are supporting only mass-market platforms (e.g. Intel) for mass-market operating systems (e.g. Microsoft Windows™, Linux).

2.5. Summary

Table IV summarizes many important metrics beyond peak performance and initial system price of HEC platforms, which as this section has shown, are often difficult to define unequivocally. In addition, several other important factors beyond cost and performance can play a critical role in HEC evaluation. These factors include reliability, robustness, availability, upgradability, and system balance. Another example is the commitment by the vendor to support HEC after the initial purchase. As the TOP500 demonstrates, the set of vendors in HEC has changed dramatically over time [24]. A proliferation of architectures and companies dotted the HEC landscape in the late 1980s and early 1990s but they then left the HEC market for a variety of reasons [25].

3. EVALUATION

We now present data for several metrics described in Section 2. As we have noted, some of the data are difficult to measure precisely. Nevertheless, our data demonstrate approximate comparisons of several large, well-known systems.

3.1. Programmability

In order for organizations to exploit new technologies, as they become available, they must use widely accepted standards in their applications. However, even if an application runs correctly on a new platform, it may not run efficiently if the platform has significantly different architectural characteristics from current platforms. For example, many codes had to be rewritten when commodity cache-based processors replaced vector-based platforms. The vector-based codes were designed to present long streams of independent computations to the vector processors—the wrong strategy for cache-based systems, which perform best when data are loaded into cache and used repeatedly. Good performance is possible on either, but moving between them can require a large investment.

To estimate the cost of converting a code to a new architecture, consider the average productivity of programmers. A common metric is lines of code (LOC) per day. Estimates vary widely, as does the productivity of individual programmers; we use 20 lines per day as an optimistic estimate [26,27]. Using 250 workdays per year and a typical programmer cost of $250 000 per year (this includes salary, benefits, office space, etc.), we estimate that code modifications cost at least $50 per line. For an existing code base of 10 million lines (which might represent half a dozen or so large applications), changing even 10% of the code to optimize it for a new architecture would cost around $50 million. The lost productivity would perhaps be an even larger burden than the cost: rewriting a million LOC would take 200 programmers a year. Clearly, organizations cannot incur these costs for every new platform.

This is not to say that programs can never be modified or that new architectures can never be accommodated, but it is clear that an organization interested in reducing TCO and ITS cannot ignore the cost of major architectural changes.

3.2. Cost/Performance

Efficiency comparisons may be the most controversial question in the discussion of vector versus commodity processors. The LINPACK numbers shows a distinct advantage for the vector machine, but LINPACK is not a real world code. It is sometimes possible to run the same code on the two different architectures, but since most codes are initially developed for one or the other, a bias may exist toward better performance on the code's 'native' system. Another possibility is to consider the winners of the Gordon Bell Prize. This has the advantage of comparing codes that are optimized to run on the machines where they are measured. The disadvantage is that the codes are likely to do different types of calculations, so results between any two codes are not comparable. Nevertheless, if we average the performance of several codes, we may get an idea of the general trend. In 2002, three of the Gordon Bell finalists ran on ES [19,25,29]. They achieved 26.6, 16.4, and 14.9 Tflops s^{-1}, corresponding to 65, 40, and 36% of peak for that machine. Two other finalists ran on traditional MPPs. One ran on a Compaq Alpha cluster [30], and the other ran on ASCI Red (an Intel system) and ASCI White [31].

These codes achieved 12, 30, and 23% of peak, respectively. All achieved noteworthy performance yet there is obviously a wide range of performance levels, even among the three ES codes. A crude measure of the performance advantage of floating-point efficiency that the vector machine offers can be calculated as the ratio of the average percentages of peak for the two categories of codes. This works out to about 2.2.

We use a ratio of 2.5 to be generous to ES and adjust the advertised peak of non-vector platforms downward to account for the apparent performance advantage of the vector machines. Table V shows how this adjustment affects the relative costs of the systems. It also accounts for the power usage, and it shows PP indexed by delivery dates. ASCI Purple and BG/L are not considered here because of the greater uncertainty in PP for them and because we have no experience yet on which to base performance estimates for BG/L.

We estimate each platform has a five-year lifetime and compute the cost of the total delivered flops (the aggregate number, not the rate). We assume that each platform delivers its full-sustained performance 24 h a day over its entire lifetime, so our estimate does not account for differences in reliability and availability. Interestingly, ASCI White and ES are essentially identical by this rough measure. The most expensive platform in this list is Blue-Pacific, even accounting for the declining cost of computer power. One reason may be that the Blue-Pacific contract included several smaller systems. The cheapest platform by far on our list is LLNL's MCR Cluster. Its low cost arises in part from the use of commodity hardware and open source software. However, its PP did not include much of the support software that came with the IBM platforms.

3.3. ASCI application performance study

In this section, we present performance data for three important benchmark applications included in the set used for the procurement of ASCI Purple. We measured the performance of these applications across several platforms with different hardware and software configurations. Generally, our results show steady improvement of each application's performance as new platforms are designed.

3.3.1. sPPM

sPPM [11] solves a three-dimensional gas dynamics problem on a uniform Cartesian mesh, using a simplified version of the Piecewise Parabolic Method. The algorithm uses a split scheme of X, Y, and Z Lagrangian and remap steps, which are computed as three separate sweeps through the mesh per timestep. Message passing updates ghost cells from neighboring domains three times per timestep.

Three versions of this application exist. DEFAULT is the original application as distributed on the ASCI Purple Web site. The COMP version shows improvements in sPPM performance that resulted from manual optimizations being integrated into the compiler, leaving the source code unchanged. The HAND version of the code allows any changes to the source code, including a total rewrite of the application for a target system. These three cases demonstrate the potential improvements in application performance that can result from considerable optimization effort. That said, these types of extensive optimization efforts must be balanced against other organization goals and the number of target platforms.

Figure 4 shows the scaled speedup for sPPM across seven platforms. Although the speedup in sPPM is less than that of our other two applications, the trends in improved TTS are obvious.

Table V. Adjusted performance and indexed costs for several LLNL machines and the ES. Indexed costs reflect estimated cost of the system in February 2003, based on a 4% monthly decrease in cost. US$M denotes U.S.$million.

System	PEAK (Tflops s^{-1})	Adjusted performance (Tflops s^{-1})	PP (US$M)	Total power cost (US$M)	Lifetime (years)	Total cost (US$M)	U.S.$/$10^9$ sustained flop	Installation date	Indexed PP (US$M)	Indexed U.S.$/$10^9$ sustained flop
ASCI Blue-Pacific	3.9	1.6	93	3	5	96	380	October 1998	12	62
ASCI White	12.3	4.9	85	5	5	90	110	September 2000	27	42
ES	41	41	350	50	5	400	55	February 2002	220	43
MCR Cluster (LLNL)	11.2	4.5	14	2	5	16	20	October 2002	12	20

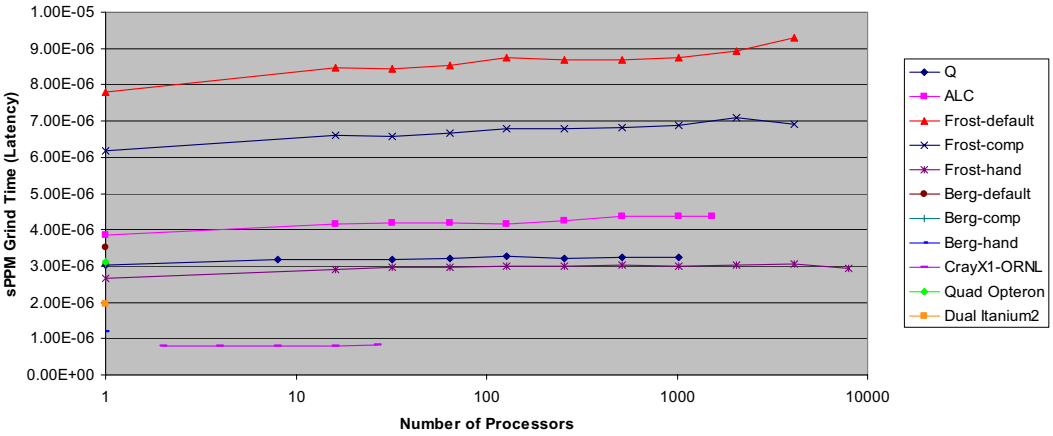

Figure 4. sPPM TTS for various architectures.

3.3.2. UMT

UMT is a three-dimensional, deterministic, multigroup, unstructured mesh photon transport code. The algorithm solves the first-order form of the steady-state Boltzmann transport equation. The equation's energy dependence is modeled using multiple photon energy groups, and the angular dependence is modeled using a collocation of discrete directions. The spatial variable is modeled with an upstream corner balance finite volume differencing technique. The solution proceeds by tracking through the mesh in the direction of each ordinate. For each ordinate direction all energy groups are transported, accumulating the desired solution on each zone in the mesh. The code generates the mesh at run-time using a 2D unstructured mesh that it extrudes it in the third dimension using a user-specified amount.

Figure 5 shows the scaled speedup for UMT across several platforms. As the experiments demonstrate, the speedup in UMT improves across platforms. The best performance for UMT is on ASCI Q.

3.3.3. IRS

IRS [32] is an implicit radiation code that solves the radiation transport equation by the flux-limited diffusion approximation using an implicit matrix. IRS uses the preconditioned conjugate gradient method (PCCG) for inverting a matrix equation. A planar radiation wave diffuses through a rectangular mesh from one end to another in the algorithm. The problems execute for longer than it takes to traverse the spatial problem. This increases the radiation iteration count dramatically, which stresses certain aspects of parallel communications. As Figure 6 depicts, the scaled speedup of IRS has continued to improve on LLNL platforms over the last five years.

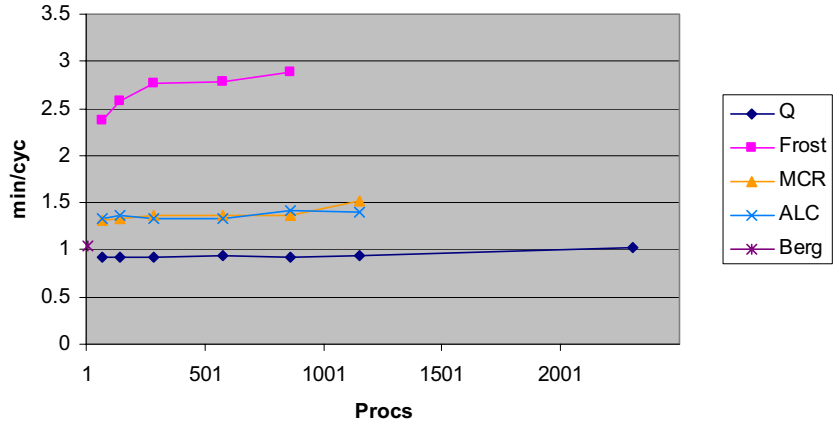

Figure 5. UMT TTS across several platforms for a scaled workload.

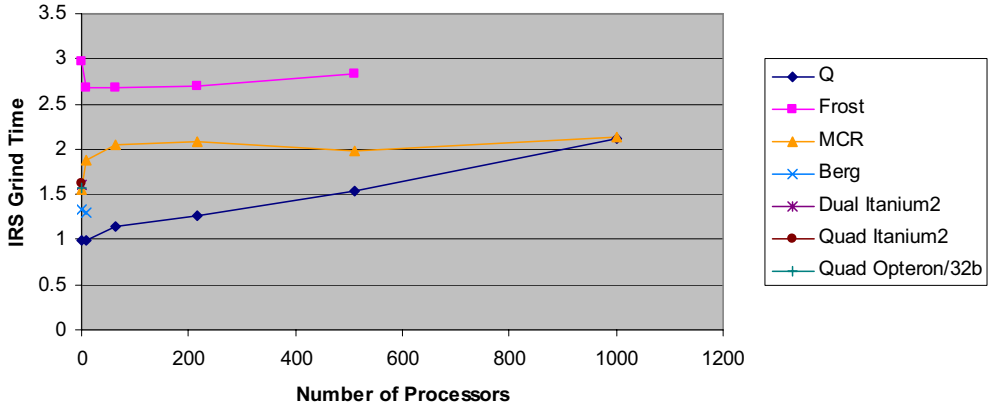

Figure 6. IRS TTS across several platforms for a scaled workload.

4. SYSTEM TECHNOLOGY

In this section, we examine the technologies underlying the components of HEC platforms. It is important to understand HEC system component technology trends since the critical element in making a cost-effective computer changes over time. For example, the looming Memory Wall, that is the difference in the rates of improvement of processor and main memory speeds [33]; indicates that memory subsystems will grow in importance in the near-term future.

HEC systems have dozens of technology decision points that reflect the requirements and the budgets of the system purchasers. These decision points range from commodity components that provide reasonable performance at a low cost to specialized components that satisfy parochial requirements, often for a high cost. For our purposes, commodity components are not only mass-produced components but also are essentially identical components available from multiple vendors in large volumes.

In today's HEC environment, many low-cost computer systems, such as the Beowulf systems [14], are true commodity platforms that use only commodity components. At the opposite extreme, specialized (proprietary) computer systems employ numerous proprietary components yet a large portion of these systems remain commodity platforms. For example, even specialized systems like GRAPE [34] and QCDSP use traditional commodity RAM main memory and commodity storage subsystem components.

We examine trends for four essential components of HEC platforms: microprocessors; memory subsystems; interconnects and storage subsystems. Technology and market trends drive all of these individual components. Thus, we examine semiconductor trends before discussing the individual components.

4.1. Semiconductor technology trends

Primary technology advances that impact HEC have come in the area of semiconductor devices, principally CMOS logic and DRAM, and related areas such as packaging, pins, and cooling [23]. Advances in optical communication are also beginning to contribute to high-end systems, primarily as inter-rack interconnect, and will play a more substantial role in the future. Although it is unclear how and when emerging technologies, such as MRAM and nanotechnology-based circuits and devices, will also affect future directions.

Semiconductor devices will continue to increase significantly in logic density, storage capacity, clock rates, and I/O. CMOS circuits will achieve 1 billion logic transistors per die and clock rates that approach 10 GHz, although chip-wide clock signal distribution may be somewhat slower. Memory capacity per chip will reach 1 billion bytes of DRAM storage. While the pins available for signaling will grow slowly, they are likely to approach 1000 pins for some dies, with signaling rates between 2 and 8 GHz. Achieving these pin counts and signaling rates may require differential pairs for improved signal to noise, which will reduce the total number of available I/O ports.

These continued improvements will aggravate several aspects of component operation. The delays measured in clock cycles due to communication latencies, contention for shared resources, and overhead functions for managing concurrent activities and parallel resources will increase by as much as an order of magnitude due to the product of increased clock rates and greater system diameter of the largest systems. Bandwidth to processor chips must increase as both the number of arithmetic units per processor and the per-processor switching speeds are increased.

4.2. Microprocessors

The history of microprocessor technology, as voluminously documented in textbooks, technical papers, and the press, demonstrates that until recently there was an ample difference between the

performance of the mass-market microprocessor and those used in costly high-end workstations and supercomputers. However, a dramatic convergence in the capabilities of these microprocessors has created a situation where some of the most powerful microprocessors are available in personal computers. Furthermore, the technology trends in semiconductor capability continue to track Moore's law and will continue to do so for at least the next 5–10 years.

Currently, microprocessors have several architectural trends [36–38] including advanced superscalar processors, processors in memory (PIM), simultaneous multithreading (SMT), multiprocessors on a chip, and raw processors. Microprocessor chips for high-end computers will continue to evolve significantly from today's conventional designs. One approach, referred to as 'system on a chip' (SOC) [39], will put an entire computer node on a single die [40]. The microprocessor, its cache, and DRAM memory along with external I/O interface ports will be a single integrated element. A second approach will put a number of microprocessors on a single chip each with their own caches and sharing an L3 cache. This 'SMP on a chip' or chip multiprocessor (CMP) [41,42] strategy will support full cache coherence among the multiple processors on the chips and have them all work out of a common external main memory. BG/L will employ dual processors per chip with shared caches (but independent L1 caches for which hardware coherence is not provided). In both cases, the processors are similar to those architectures of today but benefit from the denser integration that new semiconductor technologies will enable.

More radical designs for both processor and memory will be possible. Processors may have far greater numbers of arithmetic units organized in large arrays programmable on a per instruction basis (e.g. Streams, Trips). Substantial logic may be integrated directly within the memory chips. These 'processor in memory' or PIM architectures can provide limited memory-oriented operation support such as buffering multiple simultaneous memory requests, handling sophisticated compound atomic operations (e.g. fetch and add), or doing content addressable searches. Alternatively, PIMs [43,44] may be designed to perform sophisticated programmable functions directly in the memory where low temporal locality precludes effective use of processor caches.

Advances in microprocessor technology will be influenced heavily by the cost of process technology and fabrication plants. Since these costs increase exponentially with decreases in feature size, massive capital investments are required to implement new architectures and technologies (Moore's second law [45,46]). For example, since 2001, Intel has spent nearly $12 billion on new equipment [45] for chip manufacturing. Another example includes IBM's recent $2.5 billion semiconductor manufacturing facility investment [47]. This massive initial investment in high-end technology combines with the small margins of commodity markets to limit the number of alternatives available to the market.

4.3. Memory subsystem

Memory system design and performance are critical factors for the overall performance for scientific computing workloads. Crucial differences in memory system technologies exist not only between proprietary and commodity systems, but also within commodity systems. In this section, we discuss the wide variety of components and designs used to implement memory systems.

Any random access memory (RAM) consists of arrays of memory cells—electronic devices that retain a charge to represent a one. There are two types of memory cells—static RAM (SRAM) cells use a small set (four to six) of transistors to retain the charge, while dynamic RAM (DRAM) cells consist of a single transistor and a capacitor to store the charge. The basic difference in cell design implies that

DRAM devices are denser than SRAM devices, generally providing four to six times as many cells per unit area when implemented with similar process technologies. Thus, DRAM devices are significantly less expensive in general than similar capacity SRAM devices. Alternatively, the SRAM memory cell design provides lower latency and zero wait state accessibility.

Traditionally, SRAM has been preferred for on-chip and cache memories since these levels of the memory hierarchy are on or close to the critical path. Also, DRAM processes were incompatible with CPU processes so DRAM devices could not be included in logic devices [48]. However, embedded DRAM (eDRAM) is now common [49] and will be used more frequently as modern applications require larger caches. Already, POWER4 systems use two 16 MB eDRAM chips to implement the off-chip L3 cache [50], while the BG/L compute ASIC will include a 4 MB L3 implemented in eDRAM [17]. Since eDRAM latencies will improve relative to those of SRAM, eDRAM will be used more and more frequently [51]. However, eDRAM latencies will remain higher than those of SRAM, so portions of the memory hierarchy, such as L1 caches, will continue to use SRAM.

Main memories have almost used DRAM because of the significantly higher cell densities. Fast-Page Mode (FPM) DRAM, Extended Data Out (EDO) DRAM, Synchronous DRAM (SDRAM), Double Data Rate (DDR) SDRAM and Enhanced SDRAM (ESDRAM) [52] represent a straightforward evolutionary technology path of advances to how the DRAM is accessed and of caching within the DRAM device from the basic design of an array of memory cells [53,54]. SyncLink DRAM (SLDRAM) [55] and Direct Rambus (RDRAM) [56], two of the most recent main memory technologies to emerge, use address/control packet protocols to support higher frequencies and provide performance gains through critical-word-first-ordering over smaller width buses [57]. It is unclear which of these technologies will become the most prevalent.

Figure 7 presents the lowest price found for a variety of DRAM technologies during an Internet survey of DIMM or RIMM prices conducted in January 2003. This survey demonstrates that DRAM costs have dropped significantly over time—all of the prices were below $1 per MB. Not surprisingly, the most recent and highest performance technologies tend to cost more. However, this trend actually is reversed over time—although not shown, we found that FPM and EDO DRAMs were the most expensive per MB. Figure 7 does not include SLDRAM, which is not yet widely available in commodity DRAMs.

The memory subsystems of today's vector-based and scalar-based systems have many similarities. Highly banked main memories originated in early vector systems. Experiences with vector memory systems helped motivate current DRAM designs such as RDRAM and SLDRAM. We expect other aspects of vector memory systems, such as the Impulse memory controller support for scatter/gather [58], to become common in scalar-based systems. Alternatively, early vector machines, such as the Cray I, had SRAM main memories. However, current main memory sizes make SRAM prohibitively expensive [59] so current vector systems and SMPs use similar DRAMs: the Cray X1 accesses its standard RDRAM through a crossbar interconnect [60], while the IBM Nighthawk II used in ASCI White accesses its standard SDRAM through a crossbar interconnect.

Although the memory systems of scalar and vector systems still diverge significantly above the main memory level, they are becoming more similar. Original vector systems used little or no cache, instead using the equivalent chip space for large vector register banks. However, vector systems now include caches—the scalar units of the ES processors have 64 KB data caches, while the Cray X1 processor has instruction and data caches associated with each of its four scalar units and 2 MB that the units and associated vector pipes share.

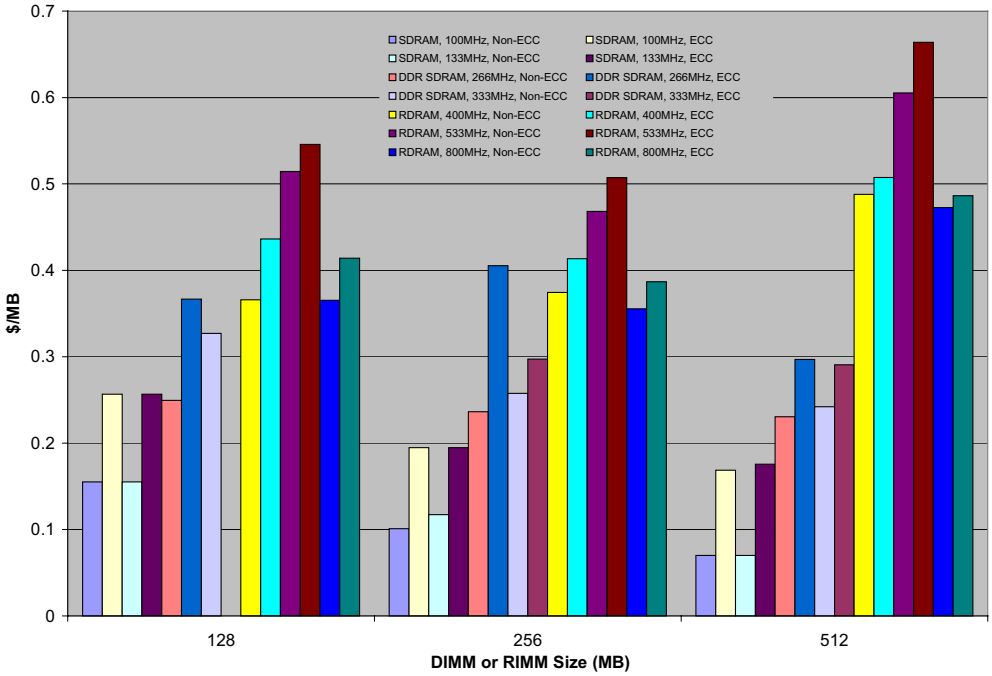

Figure 7. Memory capacity and cost ratios.

4.4. Interconnection network technologies

A high-performance interconnection network converts a collection of nodes into a single HEC system and its performance has been shown to dramatically impact the performance of scientific applications. Designers of today's HEC platforms can select from an array of interconnect choices: Gigabit Ethernet, Quadrics QsNet, Scalable Coherent Interface, Myrinet, Infiniband, and a collection of proprietary interconnects, such as the IBM SPSwitch2. ASCI White and Blue-Pacific use an SPSwitch2. ASCI Q and MCR use Quadrics QsNet. These choices offer a range of topologies, switching strategies, scalability, price, and performance parameters. The basic performance model for these networks can be divided into network latency and bandwidth, which result from system design parameters like the network topology and switching strategy.

Most systems have more than one network. Many systems use Ethernet for a reliable, inexpensive RAS network for managing nodes and a high-performance interconnect like Quadrics QsNet for application communication and, possibly, IO. Similarly, BG/L will have five networks including a three-dimensional torus, a global tree, and a Gigabit Ethernet, each targeting specific communication tasks.

Several less public concerns also impact design choices, such as the installation and weight of network cabling. For example, ASCI White uses almost 83 miles of cabling with its SPSwitch2 while the 83 000 copper cables of ES's single stage crossbar switch span approximately 1800 miles.

4.5. Storage subsystem

Scientific applications perform three main types of input and output [61]:

- required I/O, including reading input data and storing final results;
- checkpoint I/O, which saves a computation's intermediate state so that it can be restarted later in case the system fails or the job is interrupted;
- data staging, which is moving data between main memory and disk when the full data set is too large to fit in main memory.

These I/O operations demand several features from I/O subsystems, including the following:

- low access time (measured from when an I/O request is issued until the first byte arrives at its destination);
- high transfer rate (number of bytes moved per unit time);
- globally accessible files (all files accessible from all nodes);
- efficient concurrent access from multiple nodes to a single file.

Not every form of I/O needs all these features at once, but a general-purpose HEC platform must offer reasonable solutions for all forms of I/O. As a result, some HEC platforms use multiple I/O systems with different combinations of features. Three types of I/O systems are used in parallel computers: node-attached disks; parallel file systems using the message-passing network; and parallel file systems using a storage-area network (SAN).

Node-attached disks, disks that are directly connected to compute nodes (by the term 'disk', we mean an individual disk drive or a multiple-disk storage device) meet the requirement for fast access and rapid transfers (from individual nodes) well. They avoid contention for network bandwidth and the latency that arises when data traverse multiple links. For these reasons, node-attached disks are often used for checkpointing and data staging. This solution implies costs in ease of use: node-attached disks require explicit data movement from larger, centralized storage systems for data-staging and required I/O; for checkpoint, explicit data movement is also required or the job must be restarted on the same set of nodes. Further, some large systems do not offer node-attached disks because they add to the cost, power consumption, and size of a system.

Node-attached disks do not offer globally accessible files or concurrent access is poorly suited for the required I/O. To make files available to all the nodes, most HEC platforms use a global file system. These systems connect storage devices to compute nodes through a network, either the interconnection network that the system uses for message passing or a separate SAN. Global file system software makes all files accessible to all nodes on the network. A global file system is called a parallel file system if it uses file striping to distribute data over multiple storage devices so that nodes can access an individual file in parallel, which improves the aggregate transfer rate.

Files are usually striped over a subset of nodes that serve as I/O nodes, each connected to a large storage device, when the global file system uses the message-passing network. When multiple nodes

read or write concurrently, the aggregate transfer rate can be quite high. On the other hand, the access time is greater than it would be for node-attached disks because of the time to traverse the network, the I/O node, and finally the connection to the storage device.

A newer approach to parallel file systems uses a SAN and network-attached storage devices instead of the message passing network and I/O nodes. SAN-based systems have several advantages over file systems that use the message-passing network. First, they use dedicated resources for moving and processing data instead of compute nodes and message-passing network capacity. Second, intelligent storage devices can be directly connected to the SAN. Third, and most importantly for sites like LLNL that have several HEC platforms, SAN-based systems can serve multiple systems since they are not tied to a specific system and can work with different computer architectures. This last advantage allows users to generate data on a system designed for computing power and then visualize it on one designed for rendering without using explicit data movement between them. Further, it will save money on storage hardware in the long run since it eliminates the need to duplicate files across multiple file systems. Sharing access to files over a network has been in use for many years through, for example, the Network File System (NFS) [62]. However, these early systems were not designed to handle striped files or to support efficient concurrent writing from multiple nodes. Although SAN-based parallel I/O systems are not yet widely used, several implementations are under way, including Lustre [63].

5. PROGRAMMING MODELS

ASCI programming environments stress most software development tools due to the scale of the hardware architecture, software complexity, and the requirement of compatibility/portability across ASCI platforms. Development, testing, and validation of multi-physics applications require 4–6 years. These codes have at least a 10-year lifespan and may be used for several decades. Thus, they must run on not only today's HEC platforms, but also future ones.

This long lifespan implies strategic importance for the programming model choice: code portability and longevity are major objectives. ASCI codes use standards-conforming languages and do not exploit idiosyncratic processor-specific optimizations or non-standard compiler features since they must execute at all three ASCI sites (LLNL, LANL and Sandia National Laboratories). While small code modifications to improve performance on one specific platform do occur, major changes that tie one application to one platform are avoided since the performance improvement can be limited and predicted improvement must outweigh other priorities, such as adding new features to the application.

Figure 1 illustrates that our users could rely on a relatively stable development environment for almost 25 years. For example, users programming on the CDC 6600 in 1972 could rely on the same vendor and general operating environment for nearly 15 years. During the mid-1990s, a wide range of platforms became available and challenged this stability. The difficulties presented by this environment led to the ASCI program paying significant attention to the portability and compatibility of programming models, languages, and software environments.

ASCI's platform diversity has led nearly all of today's LLNL applications to use a combination of three programming models: the serial model, the symmetric multiprocessor model using OpenMP, and the message-passing model using MPI. Most applications use MPI for coarse-grain concurrency, and OpenMP, or possibly POSIX threads, for fine-grain concurrency. All three programming models combine with standard languages, such as ANSI C and FORTRAN 90, to provide a portable and

compatible software development environment across all current major HEC platforms. ASCI has also invested in software tools to provide a consistent development environment across important platforms including the Etnus' Totalview™ debugger and the Pallas' Vampir™ performance analysis tool.

6. CONCLUSIONS AND RECOMMENDATIONS

We conclude our analysis with four primary conclusions and four recommendations. Overall, we find that there are no universal metrics, models or benchmarks for determining what HEC platform will best serve a given user community and that rational purchasing decisions must account for a number of extrinsic factors, like portability/compatibility and risk.

6.1. Universal metrics

The HEC community has long understood that PEAK correlates only roughly to TTS for an application of interest. As noted in Section 2.1, other features of an architecture, such as latency and bandwidth, also influence TTS. Measures, such as %PEAK and cache utilization, can show how a particular application interacts with a particular architectural component, and this information can be useful in the process of tuning a code. However, because algorithms exercise computers in such complex ways, no single metric fully and accurately describes the performance of real applications on a machine. We state this obvious and uncontroversial conclusion because PEAK and %PEAK will likely continue to be used as shorthand measures of architectural and application performance. The HEC community should beware of assigning too much importance to them.

6.2. Diverse workloads limit feasibility of one benchmark

Many benchmark codes capture a HEC platform's performance more accurately than a single metric can. For workloads with only one major application, or a set of applications with similar performance characteristics, a few benchmark results may describe performance completely. However, workloads with a wider variety of applications [64,65] are much harder to benchmark. For example, a platform may deliver %PEAK values for different algorithms that vary by more than a factor of 10. To predict overall performance accurately, one would have to weight benchmark results by the expected usage of different algorithms in real applications. However, this information may not be known in advance, since different inputs to the same application may change the proportion of time it spends in different algorithms. In general, the broader the mixture of algorithms, the harder it is to predict a machine's performance based on benchmarks.

6.3. Extrinsic factors matter

Even the most accurate benchmarks describe only a part of a platform's value. Other intrinsic features, such as power consumption and facilities requirements, impact a system's cost. So do many extrinsic features, such as acquisition cost and the cost of writing or modifying software to run efficiently on the platform. Code developers may specifically tune algorithms to run well on one platform, or they may write code that is easy to maintain and port but not well optimized for any one system. Given the long lifespan of most large scientific codes, sacrificing some performance in favor of portability often pays off in the long run.

Yet another extrinsic factor is the risk involved in buying a cutting-edge HEC platform. Organizations must consider whether a new system is likely to meet technical goals on schedule and whether its vendor will remain in business to support it throughout its lifetime. The consequence of failure can be a loss of both money and time. These risks are difficult to quantify precisely because the community of HEC vendors changes continually, so previous experience is not always an accurate guide.

6.4. Procurement evaluations must reflect organizational requirements

The reemergence of vector architectures and the growing use of commodity clusters has increased the need for a rational approach to evaluating HEC platforms. We have presented several measures for their utility and cost. It is tempting to combine these measures into a model that projects the total cost per unit of delivered computational work. However, we believe that no model can provide reliable broad conclusions about the relative value of different types of platforms since there are too many uncertainties. First, TTS, which is the performance measure that really matters, depends intimately on the details of the algorithms in use and the problems being solved. Purchase price, TCO, and risk introduce further uncertainties. Finally, and often unacknowledged, the cost of porting a code base to a new machine can vary drastically among organizations based on the state of the existing code and the organization's willingness to spend time and money on system-specific optimizations.

6.5. Recommendations

We make four recommendations based on our analysis.

First, the industry's reliance on PEAK demonstrates the dearth of accurate performance prediction capabilities. Further, the lack of powerful application analysis and architecture design tools limits the consideration of possible alternative architectures. Thus, we recommend the development of capabilities in this area, such as efficient simulators and hardware-assisted monitoring, so that users and designers will be better able to explore the design space of HEC platforms.

Second, our historical data demonstrate that factors other than performance, such as facilities and power requirements, will become increasingly important as constraining metrics. Thus, we recommend an increased industry focus on power efficient hardware designs to ameliorate HEC facility costs.

Third, custom ASICs based on commodity designs such as that used in BG/L [17] provide affordable alternatives to commodity microprocessors. Since scientific applications present specialized needs compared with other application workloads, we recommend continued exploration of hybrid custom/commodity designs to provide more efficient use of power, transistors, and capabilities.

Finally, organizations should evaluate HEC platforms based on a broad set of measures that describe their needs and costs, and for the HEC community as a whole to look skeptically on oversimplified claims of cost and performance.

ACKNOWLEDGEMENTS

We gratefully thank Michel McCoy, Mark Seager, Pete Eltgroth, Kim Yates, Scott Futral, John Gyllenhaal, Bor Chan, and Thomas Sterling for their input.

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. DOE by the University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. This document was released as UCRL-JC-151967.

REFERENCES

1. Patterson DA, Hennessy JL, Goldberg D. *Computer Architecture: A Quantitative Approach* (3rd edn). Morgan Kaufmann: San Francisco, CA, 2003.
2. Pfister GF. *In Search of Clusters: The Coming Battle in Lowly Parallel Computing*. Prentice-Hall: Upper Saddle River, NJ, 1995.
3. Culler DE, Singh JP, Gupta A. *Parallel Computer Architecture: A Hardware Software Approach*. Morgan Kaufmann: San Francisco, CA, 1999.
4. Sterling TL (ed.). *Beowulf Cluster Computing with Linux*. MIT Press: Cambridge, MA, 2002; xxxiii,496.
5. Bailey DH. Twelve ways to fool the masses when giving performance results on parallel computers. *Supercomputing Review* 1991; (August):54–55.
6. Standard Performance Evaluation Corporation. SPEC benchmarks, 2003. http://www.spec.org/.
7. Bailey DH *et al.* The NAS parallel benchmarks. *The International Journal of Supercomputing Applications* 1991; **5**(3):63–73.
8. Cybenko G, Kipp L, Pointer L, Kuck D. Supercomputer performance evaluation and the perfect benchmarks. *ACM SIGARCH Computer Architecture News (Proceedings 1990 International Conference on Supercomputing)* 1990; **18**(3):254–266.
9. Amdahl GM. Validity of the single-processor approach to achieving large scale computing capabilities. *Proceedings of AFIPS Conference*, 1967.
10. Calder AC *et al.* High-performance reactive fluid flow simulations using adaptive mesh refinement on thousands of processors. *Proceedings of SC2000: High Performance Networking and Computing Conference*, 2000 (electronic publication).
11. Mirin AA *et al.* Very high resolution simulation of compressible turbulence on the IBM-SP system. *Proceedings of the SC99: High Performance Networking and Computing Conference*, 1999 (electronic publication).
12. Foster I. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley: Reading, MA, 1995.
13. Kumar V, Grama A, Gupta A, Karypis G. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin Cummings: Redwood City, CA, 1994.
14. LLNL. ASCI purple computing system. *Request for Proposal No. B519700*, Lawrence Livermore National Laboratory, Livermore, CA, 2002.
15. HPCWire. *Cray Previews New Technologies for Cray SV2 Series*. HPCWire: San Diego, CA, 2001.
16. U.S. DOE. IBM to build World's fastest supercomputers for U.S. Department of Energy, 2002. http://www-1.ibm.com/servers/eserver/pseries/news/pressreleases/2002/nov/asci_purp.
17. Adiga N *et al*. An overview of the BlueGene/L Supercomputer. *Proceedings of SC2002*, 2002.
18. Lawrence Livermore National Laboratory. Hardware environment for ASCI White, 2002. http://www.llnl.gov/asci/platforms/white/hardware.
19. Yokokawa M, Itakura KI, Uno A, Ishihara T, Kaneda Y. 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator. *Proceedings of SC2002*, 2002.
20. Dedication of the Nicholas C. Metropolis Center for Modeling and Simulation. Los Alamos National Laboratory, Los Alamos, NM, 17 May 2002. http://www.lanl.gov/worldview/news/pdf/MetropolisCenter.pdf.
21. #1 Colocation Services. http://www.onecolocationservices.com/.

22. Annual Energy Review 2001. Energy Information Administration, U.S. Department of Energy, Washington, DC, November 2002. http://www.eia.doe.gov/emeu/aer/pdf/03842001.pdf.
23. Letter symbols to be used in electrical technology—Part 2: Telecommunications and electronics. *EC60027-3* (2nd edn), IEC, November 2000.
24. University of Mannheim, University of Tennessee, and NERSC/LBNL. TOP500 Supercomputer Sites, 2003. http://www.top500.org/.
25. Kuck DJ. *High Performance Computing: Challenges for Future Systems*. Oxford University Press: New York, 1996.
26. Card DN. Published sources of benchmarking data, 2003. http://www.software.org/Library/pubBenchmarking.asp.
27. Software engineering program: Software measurement guidebook. *DC Report NASA-GB-001-94*, NASA, Washington, DC, 1995.
28. Sakagami H, Murai H, Seo Y, Yokokawa M. 14.9 TFLOPS three-dimensional fluid simulation for fusion science with HPF on the Earth Simulator. *Proceedings of SC2002*, 2002.
29. Shingu Y *et al.* A 26.58 Tflops global atmospheric simulation with the spectral transform method on the Earth Simulator. *Proceedings of SC2002*, 2002.
30. Phillips JC, Zheng G, Kumar S, Kale LV. NAMD: Biomolecular simulation on thousands of processors. *Proceedings of SC2002*, 2002.
31. Bhardwaj M, Pierson K, Reese G, Walsh T, Day D, Alvin K, Peery J, Farhat C, Lesoinne M. Salinas: A scalable software for high-performance structural and solid mechanics simulation. *Proceedings of SC2002*, 2002.
32. Anderson WK, Gropp WD, Kaushik DK, Keyes DE, Smith BF. Achieving high sustained performance in an unstructured mesh CFD application. *Proceedings of Supercomputing*, 1999 (electronic publication).
33. Wulf WA, McKee SA. Hitting the memory wall: Implications of the obvious. *Computer Architecture News* 1995; **26**(1):20–24.
34. Hut P, Makino J. Astrophysics on the GRAPE family of special-purpose computers. *Science* 1999; **283**(5401):501–5.
35. ITRS Committee. The International Technology Roadmap for Semiconductors, 2003. http://public.itrs.net.
36. Burger D, Goodman JR. Billion-transistor architectures. *Computer* 1997; **20**(9):46–48.
37. Kozyrakis CE *et al.* Scalable processors in the billion-transistor era: IRAM. *Computer* 1997; **30**(9):75–78.
38. Patt YN, Patel SJ, Evers M, Friendly DH, Stark J. One billion transistors, one uniprocessor, one chip. *Computer* 1997; **30**(9):51–57.
39. Nunomura Y, Shimizu T, Tomisawa O. M32R/D: Integrating DRAM and microprocessor. *IEEE Micro* 1997; **17**(6):40–48.
40. Nair R. Effect of increasing chip density on the evolution of computer architectures. *IBM Journal of Research and Development* 2002; **46**(2/3):223–234.
41. Diefendorff K. Power4 focuses on memory bandwidth. *Microprocessor Report* 1999; **13**(October).
42. Hammond L, Hubbert BA, Siu M, Prabhu MK, Chen M, Olukotun K. The Stanford Hydra CMP. *IEEE Micro* 2000; **18**(2):71–84.
43. Brockman JB, Kogge PM, Sterling T, Freeh VW, Kuntz SK. Microservers: A new memory semantics for massively parallel computing. *Proceedings of the ACM International Conference on Supercomputing (ICS)*, 1999; 454–463.
44. Draper J *et al.* The architecture of the DIVA processing-in-memory chip. *Proceedings of the ACM International Conference on Supercomputing (ICS)*, 2002.
45. HPCWire. *Intel to Produce Two-processor Itanium Chip*. HPCWire: San Diego, CA, 2003.
46. Carlson J, Huck J. *Itanium Rising: Breaking Through Moore's Second Law of Computing Power*. Prentice-Hall: Upper Saddle River, NJ, 2002.
47. Lohr S. IBM opening a $2.5B specialized chip plant. *New York Times* 2002; (2 August).
48. Saulsbury A, Pong F, Nowatzyk A. Missing the memory wall: The case for processor/memory integration. *Proceedings of 23rd International Symposium on Computer Architecture (ISCA)*, 1996; 90–101.
49. In-Stat/MDR Market Alerts. Embedded DRAM market to grow 21.3%—High-end communications to dominate consumption. *Press Release*, 2002.
50. Tendler JM, Dodson S, Fields S, Le H, Sinharoy B. IBM eServer: POWER4 system microarchitecture. *Technical White Paper*, IBM, 2001.
51. IBM Microelectronics Division, White Plains, NY. IBM Embedded DRAM, 2002.
52. Enhanced Memory Systems. *16 Mbit Enhanced Synchronous DRAM: Preliminary Data Sheet*, Enhanced Memory Systems 1999.
53. Gries M. A survey of synchronous RAM architectures. *Technical Report TIK-Report 71*, Swiss Federal Institute of Technology Europe, 1999.
54. Mitra T. Dynamic random access memory: A survey. *Technical Report*, State University of New York at Stony Brook, 1999.
55. Gillingham P. SLDRAM: High-performance open standard memory. *IEEE Micro* 1997; **17**(6):29–39.
56. Crisp R. Direct Rambus technology: The new main memory standard. *IEEE Micro* 1997; **17**(6):18–28.

57. Cuppu V, Jacob B, Davis B, Mudge T. A performance comparison of contemporary DRAM architectures. *Proceedings of the 26th International Symposium on Computer Architecture (ISCA)*, 1999; 222–233.
58. Carter JB, Hsieh WC, Stoller LB, Swanson M, Zhang L, McKee SA. Impulse: Memory system support for scientific applications. *Journal of Scientific Programming* 1999; **7**(3–4):195–209.
59. In-Stat/MDR Market Alerts. Good times fleeting for SRAM market. *Press Release*, 2001.
60. Partridge R. Cray launches X1 for extreme supercomputing. *Technology Trends*. D. H. Brown Associates: Port Chester, NY, 2002.
61. Miller EL, Katz RH. Input/output behavior of supercomputing applications. *Proceedings of Supercomputing*, 1991; 567–576.
62. Gould E, Xinu M. The network file system implemented on 4.3 BSD. *Proceedings of the USENIX Association Summer Conference Proceedings*, 1986; 294–298.
63. Cluster File Systems Inc. *Lustre: A Scalable, High Performance File System*. Cluster File Systems Inc.: Medford, MA, 2002.
64. Vetter JS, Mueller F. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.
65. Vetter JS, Yoo A. An empirical performance evaluation of scalable scientific applications. *Proceedings of SC2002*, 2002.