HIGH-PERFORMANCE COMPUTING: CLUSTERS, CONSTELLATIONS, MPPS, AND FUTURE DIRECTIONS

By Jack Dongarra, Thomas Sterling, Horst Simon and Erich Strohmaier

IN A RECENT PAPER,[1] GORDON BELL AND JIM GRAY PUT FORTH A VIEW OF THE PAST, PRESENT, AND FUTURE OF HIGH-PERFORMANCE COMPUTING (HPC) THAT IS BOTH INSIGHTFUL AND THOUGHT PROVOKING. IDENTIFYING KEY TRENDS WITH A GRACE and candor rarely encountered in a single work, the authors describe an evolutionary past drawn from their vast experience and project an enticing and compelling vision of HPC's future.

Yet, the underlying assumptions implicit in their treatment, particularly those related to terminology and dominant trends, conflict with our own experience, common practices, and shared view of HPC's future directions. Taken from our vantage points of the Top500 list,[2] the Lawrence Berkeley National Laboratory NERSC computer center, Beowulf-class computing,[3] and research in petaflops-scale computing architectures,[4] we offer an alternate perspective on several key issues in the form of a constructive counterpoint.

A New Path

Terminology and taxonomies are subjective, with common usage dictating practical utility. Yet, in spite of its imperfections, technical nomenclature can be a powerful tool for describing, distinguishing, and delineating among related concepts, entities, and processes. Bell and Gray incorporate a fundamental assumption throughout their reasoning, which, although defensible and advocated by notable researchers,[5] corrupts the terminology's power as a tool to represent and differentiate. Specifically, their paper implies that essentially every parallel system employing replicated resources is a cluster. In this well-intentioned effort to provide a unifying principle, though, the authors have eliminated a powerful concept even as they intended to reinforce it. The concept of the commodity cluster has driven an important trend in parallel processing over the past decade, delivering unprecedented performance-to-cost and providing exceptional flexibility and technology tracking. By expanding the scope of the term, they've deprived this important term of its seminal meaning and implication.

One objective of this article is to restore the strength and value of the term "cluster" by degeneralizing its applicability to a restricted subset of parallel computers. We'll further consider this class in conjunction with its complementing terms *constellation*, *Beowulf class*, and *massively parallel processing systems* (MPP) based on the classification used by the Top500 list, which has tracked the HPC field for more than a decade.

As Bell and Gray convincingly articulate, the impact of Moore's law and the economy of scale of mass-market computing components in easily integrated ensembles will have a significant, even dominant, impact on the evolution of high-performance systems in the near future. The Top500 list already clearly reflects this trend with the vast majority of all systems represented on the list being products of some form of clustering. Moreover, as Bell and Gray point out, Beowulf-class clusters are having a significant impact on medium-to-high-scale systems throughout the science and technical computing arena as well as in the commercial sector. Also referred to as Linux clusters or PC clusters, Beowulfs are perhaps more widely used than any other type of parallel computer because of their low cost, flexibility, and accessibility. Indeed, among the top 10 systems on the most recent list (November 2004 at www.top500.org), five are commodity clusters, three of which are Linux clusters, not unlike the original Beowulf-class systems, and two are constellations.

One consequence of the progress anticipated beyond what Bell and Gray envisioned is the form and content of future computer centers, which will evolve as available technologies and system architecture classes advance. Instead of becoming obsolete, the computer center's role will likely grow in importance, evolving to meet the challenges of new architectures, programming models, mass storage, and accessibility via the Grid. The emergence of Beowulf and other commodity clusters will definitely alter the mix of resources that will comprise a medium-to-large-sized computer center, but the respon-

 Copublished by the IEEE CS and the AIP 1521-9615/05/$20.00 © 2005 IEEE

sibilities and services that necessitate building such facilities will continue to be critical, especially to the high-end computing and large data archive communities. Already we see in the US Department of Energy and the US National Science Foundation sector the development of new and larger computing centers to house the next generation of high-end systems, including very large Beowulf clusters. The computer centers of the future will be charged with the administration, management, and training associated with bringing these major resources to bear on mission-critical applications.

This article, while congratulating Bell and Gray in opening up this line of discourse, offers a constructive expansion on their original themes and seeks to correct specific areas of their premise with which we take exception. The long-term future of HPC architectures will involve innovative structures that support new paradigms of execution models, which in turn will greatly enhance efficiency in terms of performance, cost, space, and power while enabling scalability to tens or hundreds of petaflops. The conceptual framework offered here implies the directions of such developments and resonates with recent advances being pursued by the computer architecture research community.

Commodity Clusters

Bell and Gray, in conjunction with their distinguished colleagues, see an important unifying principle emerging in HPC's evolution: the integration of highly replicated components (many of which were designed and fabricated for more general markets) as the driving force for a convergent architecture. They call this architecture a cluster and distinguish it only from the minority set of vector supercomputers (such as NEC SX-6 and Cray X1) that exploit vectors in custom processor architecture designs. This convergent architecture model of the evolution of supercomputer design is compelling, readily apparent, and wrong. We respectfully assert an alternate perspective that is rich in detail and has value in its ability as an enabling framework for reasoning about computing structures and methods. In particular, we assert that the term "cluster" is best employed not as a synonym for essentially the universal set of parallel computer system organizations, but rather as a specific class of such systems. Therefore we state that NOT everything is a cluster.

The term "cluster" is best employed not as a synonym for essentially the universal set of parallel computer system organizations, but rather as a specific class.

We limit the scope of the definition of a cluster to a parallel computer system comprising an integrated collection of independent *nodes*, each of which is a system in its own right capable of independent operation and derived from products developed and marketed for other stand-alone purposes. A commodity cluster is a cluster in which both the network and the compute nodes are commercial products available for procurement and independent application by organizations (end users or separate vendors) other than the original equipment manufacturer. Beowulf-class clusters and workstation clusters were once two distinct system types, but with the blurring or outright elimination of any meaningful differences in capability between PCs and workstations, the differentiation between these two types of clusters has also largely lost any meaning. This is particularly true with the wide usage of Linux as the base node operating system, a strategy originally pioneered by Beowulf-class clusters.

The Top500 list represents two broad classes of clusters: cluster-NOW and constellation systems. Both are commodity cluster systems distinguished by the dominant level of parallelism. Although more complex system structures are possible (such as super clusters), commodity clusters usually comprise two levels of parallelism. The first is the number of nodes connected by the global communications network, in which a node contains all the cluster's processor and memory resources. The second is the number of processors in each node, usually configured as a symmetric multiprocessor (SMP). If a commodity cluster has more nodes than microprocessors in any one of its nodes, the dominant mode of parallelism is at the first level (the cluster-NOW category). If a node has more microprocessors than there are nodes in the commodity cluster, the dominant mode of parallelism is at the second level (a constellation). The distinction is not arbitrary: it can have a serious impact on cluster programming. A cluster-NOW system, for example, is programmed almost exclusively with the message-passing interface (MPI), whereas a constellation is likely to be pro-

grammed at least in part with OpenMP, by using a threaded model. Very often, a constellation is space-shared and not time-shared, with each user getting his or her own node; space-sharing a cluster-NOW system means allocating some number of nodes to a particular user.

The critical distinction between our usage of the term cluster and that proposed by Bell and Gray is that in our narrower definition, all constituent top-level components of the system are commodity with no significant cost to the full cluster system's fabrication other than that of installation and network integration. Developing a cluster as part of a market line requires no hardware development investment other than in packaging, which might be little more than cosmetic.

This very low cost of development and exploitation of economy of scale distinguishes commodity clusters.

It is this very low cost of development and exploitation of economy of scale that distinguish commodity clusters from all other forms of scalable parallel systems; it's also this important distinction that we want to retain in our revised definition of a cluster. However, accepting Bell and Gray's broader interpretation means sacrificing commodity clusters' crucial benefit: exceptional performance-to-cost, invulnerability to specific vendor decisions, flexibility in configuration and expansion, rapid tracking of technology advances, direct use of a wide range of available (often open-source) software, portability between clusters, and a wide array of component choices. Moreover, commodity clusters provide scaling from the very small (a few nodes) to the very large (approaching 10,000 processors).

Most of these benefits come from the specific attribute that the constituent components are off the shelf with no specialty parts. This single property has made commodity clusters the dominant training environment for parallel programmers, yet the definition Bell and Gray propose would obscure, even eliminate, this seminal quality. We propose to retain it, and thus offer the following definition for a commodity cluster: a parallel computer exclusively comprising commodity computing subsystems and commercial networks such that the computing nodes are developed and employed in stand-alone configurations for broad (even mass) commercial markets, and the networks are dedicated to the private use of the cluster (non-worldly).

HPC System Taxonomy

Current HPC system architectures aren't well characterized by the notion that only two architecture classes—Cray-style vector supercomputers and parallel clusters—exist. The first class perhaps refers only to the SX NEC product line, the remaining Cray T90s, and the new Cray X1 line. What Bell and Gray consider to fall under the second category is more accurately represented by several distinct classes of parallel computing systems presenting less of a monoculture than might first appear, including

- multicomputers or multiprocessors,
- Symmetric multiprocessors (SMPs),
- distributed shared memory (DSM),
- distributed memory,
- tightly integrated MPP,
- commodity clusters, including (but not restricted to) Beowulf-class systems, and
- constellations (also a subclass of clusters).

These terms in most cases have common meaning within the literature. However, Bell and Gray observe that while constituting the common lexicon, this set of terms is not very useful in providing a general or coherent terminology to consider alternatives or to represent new architectures when they come into being.

Distinguishing the Properties of Parallel Systems

Supercomputers differ from more broadly commercialized systems in several ways that could help determine the seminal parameters or dimensions that distinguish among different classes of systems.

Performance. Although the ultimate measure of the effectiveness of a given system's execution is its response time or time to completion for a given application, other imperfect measures attempt to correlate with this fundamental metric. Mips and Gflops are among these metrics, but they represent the dependent variable or the resulting value derived from other system structures and characteristics. Except as the primary driver for the highest levels of performance, performance need not be part of the classification scheme.

Parallelism. Hardware and software parallelism determines the amount of concurrent work and there-

fore achievable peak performance. The semantics of parallelism, including its granularity and the hardware mechanisms that support parallel execution, determine the effectively exploitable amount of it. Classes of architecture can be distinguished by the kinds of parallelism they use.

Control. The hardware support mechanisms incorporated in an architecture to efficiently control the system can significantly change the system's operation and performance and distinguish among classes of systems. An SIMD computer and a cluster differ dramatically in their hardware support for system-wide parallel execution. The amount of overhead in the critical time path for controlling parallel actions thus largely depends on the hardware control mechanisms (or lack thereof).

Latency management. The wait for remote access and service strongly factors into a supercomputer's efficiency and scaling. It includes the long distances that messages have to travel, the delays when contending for shared resources such as network bandwidth, and the service times for actions such as assembling and interpreting messages. Latency management includes pipelining vectors, multithreading, avoidance via explicit locality management, caching, and message-driven computing. How a system manages latency is an important distinguishing characteristic.

Namespace distribution. From an abstract viewpoint, a system comprises a collection of namespaces and actions that can be performed on named entities. Shared memory systems versus distributed memory systems are one such division. Names of I/O ports or channels (including whether they're local to part of the system or globally accessible) is another. Process ID (again local or global) is also a discriminator, as is whether processor nodes constitute an explicit namespace or are simply a pool of anonymous resources. Logical namespaces can thus characterize a system, at least in part.

Reliance on commodity. In recent years, the economics of system implementation has dominated development. Some people consider the degree to which the system architecture exploits commodity components, subsystems, or systems as building blocks for very large structures to be an essential attribute in determining a system concept's likely success or failure. Beowulf clusters exclusively comprise commodity components, systems, and networks; the SX-6 employs custom vector processor architectures, motherboards, and networks, but uses commodity memory chips. Commodity components might be cheaper because of their economy of scale, but they could lack many functional attributes essential for efficient scalable supercomputing.

Commodity components might be cheaper, but they could lack many functional attributes.

A Framework for Characterizing Parallel Architectures

We suggest a naming schema that delineates parallel computing systems according to key dimensions of system attributes rather than the random terms with which we're all familiar. Our rationale is to demonstrate that alternative naming methods are possible, rather than to impose a specific new method on the community. For example, as Bell and Gray make clear, the term MPP, although used pervasively throughout the history of the Top500 list, is confusing, misleading, and provides little specification of system type. The notion of it has been abused, confused, and ironically derived from a different type of system than that to which it is ordinarily applied (The original MPP was a SIMD-class computer, which is a separate classification of the Top500 list).

Every strategy, including ours, reflects the critical sensitivities of its time. Here, we emphasize four dominant dimensions for characterizing parallel computing systems:

- clustering,
- namespace,
- parallelism, and
- latency and locality management.

Any system can be clustered with like systems to yield a larger ensemble system, but doing so doesn't mean that all the attributes of the constituent uniform systems are conveyed unmodified to the aggregate system. The important factor here is a synthesis of existing standalone subsystems developed for a different, presumably larger, market and user workload. The alternative, a monolithic system, is not a product of clustering but a structure of highly replicated basic components. Other appropriate designators, perhaps those that reflect a specific hierarchy, could exist—how would

Is There Light at the End of the Tunnel?

By Dan Reed, Renaissance Computing Institute (RENCI)

From the beginning of the digital age, supercomputers have been time machines that let researchers peer into the future, both intellectually and temporally. Intellectually, supercomputers help researchers bring to life models of complex phenomena when economics or other constraints preclude experimentation. Computational cosmology, which tests competing theories of the universe's origins by computationally evolving cosmological models, is one such example. Given our inability to conduct cosmological experiments (we can't create variants of the current universe and observe its evolution), computational simulation is the only feasible way to conduct experiments.

Temporally, supercomputers reduce the time to solution by enabling scientists to evaluate larger or more complex models than would be possible on conventional systems. Although this might seem prosaic, the practical difference between obtaining results in hours, rather than weeks or years, is substantial—it qualitatively changes the range of experiments we can conduct. Climate-change studies that simulate thousands of Earth years, for example, are only feasible if the time to simulate a climactic year is small. Moreover, conducting parameter studies (for example, to assess sensitivity to different conditions such as the rate of fluorocarbon or CO_2 emissions) is only possible if the time required for each simulation is small.

Hence, the supercomputing challenge has always been to deliver the highest possible performance to applications, subject to economic, political, and engineering constraints. Fueled by weapons research and national security concerns, in the US, government supercomputing needs could substantively influence the commercial market until the 1980s. With the explosive growth of personal computing and the end of the cold war, supercomputing is now a much smaller fraction of the overall computing market, with concomitantly less economic influence. This economic milieu has had profound effects on all aspects of supercomputing—research and development, marketing, procurement, and operation. Most notably, the explosive growth of clustered systems, based on commodity building blocks, has reshaped the computing market.

Although this democratization of supercomputing has had many salutatory effects, including the growth of commodity clusters across laboratories and universities, it isn't without its negatives. Not all algorithms map efficiently to the predominant cluster programming model of loosely coupled, message-based communication. Hence, some researchers and their applications have suffered due to lack of access to more tightly coupled supercomputing systems. Second, an excessive focus on peak performance at low cost, which favors commodity clusters, has limited research

we represent a supercluster (a cluster of clusters), for example?

Namespace indicates how far a single namespace is shared across a system. Although many possible namespaces exist (such as variables, process IDs, I/O ports, and so on), user variables illustrate the concept here. A distributed namespace is one in which one node's variables aren't directly visible to another, whereas a shared namespace is one in which all variables in all nodes are visible to all other nodes. Cache coherence adds to the shared namespace attribute by providing hardware support for managing copies. This illustrates that we can combine multiple attributes (lexically concatenated) to provide a complex descriptive.

Parallelism reflects the forms of action concurrency that the hardware architecture can exploit and support. Conventional distributed memory MPPs (old usage) are limited to communicating sequential processes (such as message passing), whereas vector computers exploit fine-grained vectors and pipelining. This property field exposes the means by which the overhead of managing parallel resources and concurrent tasks is supported and therefore made efficient.

Latency and locality management defines the mechanisms and methods incorporated to tolerate latency effects. Caches, pipelining, prefetching, multithreading, and message-driven computing mechanisms are among the possible mechanisms that avoid or hide access latencies. The more flexible and less sensitive to application attributes such as temporal and spatial locality, the greater the overall efficiency of execution that is likely to be achieved.

Queuing models use a method of a few descriptors separated by slashes to describe a broad range of queue system types. We consider here a similar syntax, using four fields to represent parallel architecture classes, but we extend such nomenclature to let multiple designators in any given field permit a richer description space. We suggest the fields and examples for each as follows:

- Clustering: *c* for commodity cluster or *m* for monolithic system.
- Naming: *d* for distributed, *s* for shared, or *c* for cache-coherent.
- Parallelism: *t* for multithreading, *v* for vector, *c* for communicating sequential processes or message passing, *s* for systolic, *w* for very long instruction word (VLIW), *h* for producer or consumer, *p* for parallel processes, and so on.
- Latency: *c* for caches, *v* for vectors, *t* for multithreaded, *m* for processor in memory, *p* for parcel or message-driven split-transaction, *f* for prefetching, and *a* for explicit allocation.

into new architectures, programming models, and system software. The result has been the emergence of a supercomputing monoculture composed predominantly of commodity clusters and small symmetric multiprocessors (SMPs).

In an earlier article, Gordon Bell and Jim Gray[1] describe the reasons for this monoculture's emergence, and they comment on some of its possible implications for supercomputing centers, community access to supercomputing, and distributed, peer-to-peer computing via computational grids. In the main text of the article presented here, Jack Dongarra and his colleagues disagree with many, though not all, of Bell and Gray's premises and extrapolations. In particular, they argue that Bell and Gray have defined clusters more broadly than the commonly accepted definition (that is, a computing system built largely or entirely from separately purchasable commodity components). They also suggest that economies of scale will continue to make supercomputing centers an attractive mechanism for serving the research computing needs of the very highest end national users.

Both articles argue strongly that we have substantially underinvested the research needed to develop a new generation of architectures, programming systems, and algorithms. The result is a paucity of new approaches to managing the increasing disparity between processor speeds and memory access times (the so-called von Neumann bottleneck).

These and other developments have stimulated a reexamination of current policies and approaches to supercomputing. In the US, the White House Office of Science and Technology Policy (OSTP), in coordination with the National Science and Technology Council, commissioned the creation of the interagency High-End Computing Revitalization Task Force (HECRTF). The interagency HECRTF was charged with developing a five-year plan to guide future US investments in high-end computing; the Computing Research Association (CRA) recently published a workshop report from US research community discussions,[2] and the interagency HECRTF report itself recently appeared.[5]

Researchers in every discipline at the HECRTF workshop cited the difficulty in achieving high, sustained performance (relative to peak) on complex applications to reach new, important scientific thresholds. They also made compelling cases for sustained computing performance of 50 to 100 times beyond that currently available. A complementary set of workshops,[3,4] commissioned by individual research agencies, reached similar conclusions. The result is a renewed debate about research strategies and investment in high-end computing.

In the 1990s, the US High-Performance Computing and Communications (HPCC) program supported the development of several new computer systems. In retrospect, we didn't learn the critical lesson of 1970s vector computing—the need for long-term, balanced investment in both hard-

continued on p. 26

Admittedly, we could probably add other designations to this list: for example, the Earth Simulator would be *m/s/v/v*, the Tera MTA (multithreaded architecture) would be *m/s/t/t*, the SGI Origin would be *m/c/p/c*, and Red Storm would be *m/d/c/a*. The community would have to work out the precise codification in greater detail and accuracy, so clearer guidelines are needed for such a representation schema to be established or to foster community acceptance. Here, we simply suggest a strategy for addressing this challenge. The basic concept permits the system to be described by its attributes rather than as a collection of terms not necessarily making up a coordinated lexicon. MPP, for example, could mean a distributed memory system, a large shared memory system with or without cache coherence, a large vector system, and so on: it covers too many categories and hides too many salient differences to be a useful tool for description. On this point, we have come to agree with Bell and Gray.

Traversing the Trans-Petaflops Performance Regime

Clusters, as Bell and Gray suggest, could have a long life as an architecture principle because many workloads can tolerate their limitations and benefit from their economy of scale. It's also quite probable that sometime between 2009 and 2012, one or more commodity clusters will exhibit a peak performance of 1 Pflops or greater. That said, the evolution of high-end computing isn't likely to lead ultimately to the convergent cluster architecture Bell and Gray predict, but rather to a new class of parallel architectures that respond to the opportunities and challenges presented by the technology trends that drive it.

The memory wall, or von Neumann bottleneck (among other terms), represents the disparity between processor clock rates and memory cycle times as well as system-wide remote access latencies. By the end of the decade with no changes in structure, due to the increase in memory densities and processor clock speeds it will take at least 10 times as long (measured in processor cycles) to touch every word once in a given memory chip. Without methods for tolerating latency, computations will suffer a 1,000-cycle critical time delay for access to remote memory, perhaps even longer.

If no changes to HPC system architecture occur, we'll have to configure the most expensive parts of the system—the I/O bandwidth and memory bandwidth—via ever larger and expensive caches to optimize for the least expensive component, the arithmetic logic unit. Fortunately, new architectures will be able to exploit between

continued from p. 25

ware and software. Achieving high-performance for complex applications requires a judicious match of computer architecture, system software, and software development tools. Most researchers in high-end computing believe the key reasons for our current difficulties in achieving high performance on complex scientific applications can be traced to inadequate research investment in software and the use of processor and memory architectures that aren't well matched to scientific applications.

Today, scientific applications are developed with crude software tools (compared to those used in the commercial sector). Low-level programming, based on message-passing libraries, means that application developers must provide deep knowledge of application software behavior and its interaction with the underlying computing hardware. This is a tremendous intellectual burden that, unless rectified, will continue to limit the usability of high-end computing systems, restricting effective access to a small cadre of researchers. We need only look at the development history of Microsoft Windows to recognize the importance of an iterated cycle of development, deployment, and feedback to develop an effective, widely used product. High-quality research software isn't cheap: it is labor intensive, and its successful creation requires the opportunity to incorporate the lessons learned from previous versions.

Hence, we must begin a coordinated research and development effort to create high-end systems that are better matched to the characteristics of scientific applications. This will require a broad program of basic research into computer architectures, system software, programming models, software tools, and algorithms. In addition, we must fund the design and construction of large-scale prototypes of next-generation high-end systems that includes balanced exploration of new hardware and software models, driven by scientific application requirements. After experimental assessment and community feedback, the most promising efforts should then transition to even larger scaling testing and vendor product creation, and new prototyping efforts should be launched. This critical cycle of prototyping, assessment, and commercialization must be a long-term, sustaining investment, not a one-time crash program.

References

1. G. Bell and J. Gray, "What's Next in High-Performance Computing," *Comm. ACM*, vol. 45, no. 2, 2002, pp. 91–95.
2. D.A. Reed, ed., *Workshop on the Roadmap for the Revitalization of High End Computing*, Computing Research Assoc., Jan. 2004; www.cra.org/reports/supercomputing.pdf.
3. *A Science-Based Case for Large Scale Simulation*, US Dept. of Energy, June 2003, www.pnl.gov/scales/.
4. *Revolutionizing Science and Engineering through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*, US Nat'l Science Foundation, Jan. 2003, www.cise.nsf.gov/evnt/reports/toc.htm.
5. *Federal Plan for High-End Computing: Report of the High-End Computing Revitalization Task Force (HECRTF),* May 2004, www.nitrd.gov/pubs/2004_hecrtf/20040702_hecrtf.pdf

Dan Reed is director of the interdisciplinary Renaissance Computing Institute, which spans the University of North Carolina at Chapel Hill, North Carolina State University, and Duke University. He also holds the Chancellor's Eminent Professorship at the University of North Carolina at Chapel Hill and is a member of the President's Information Technology Advisory Committee (PITAC). Reed conducts research in high-performance computing, fault-tolerance, performance measurement, and interdisciplinary applications. He is a fellow of the IEEE and ACM. Until December 2003, he was director of the NSF-funded National Center for Supercomputing Applications (NCSA) and chief architect of the NSF TeraGrid, a nationally distributed high-performance computing Grid built from commodity clusters. Contact him at reed@renci.org.

one and two orders of magnitude more Arithmetic Logic Units (ALUs) for a given scale system than is possible today. Such architecture classes include multicore system on a chip (SOC), processor in memory (PIM), streaming, and vector.

Although Moore's law will apply unabated and commodity components will dominate system design in the short run, Moore's law will eventually flat-line due to atomic and quantum effects, and conventional components will provide low efficiency such that little gain will be achieved for larger systems, even for larger problems. (This is already happening, but certain embarrassingly parallel problems will always be the exception.) As problems get larger, they often increase their sequential time scale as well (taking more simulation time steps for the same simulated period). At some point, architecture will begin to dominate, and the field will ultimately emancipate itself from the narrow commercial off-the-shelf (COTS) mentality. Although it's impossible to predict the future, some possibilities are evident even now.

In one sense we agree with Bell and Gray. Clusters (even in our narrower sense of the term) will indefinitely remain as an important part of supercomputing because no matter how large or innovative a system is, someone can always assemble a larger one through clustering—it's like turbo charging. As more than a decade of research has shown us, though, clustering could be achieved while retaining a user global namespace and with some cross-system latency hiding. This isn't unreasonable as long as

- the logical interface to the external

world includes access to the address management and translation mechanisms, and
- the different cluster nodes can subdivide the namespace in a mutually exclusive, collectively exhaustive, and logically consistent manner.

But to cluster in this way means that the processing components have to "know" about each other and their local worlds, which is outside the scope of current COTS processor architectures and within the realm of custom system designs.

In terms of the conventional balance of bytes per flops, we'll lose the ratio of 1:1 because of the roughly 1,000-to-1 difference between the required memory and the ALUs needed to match it. Instead, an entirely different set of metrics and balance requirements will drive future architectures based on bandwidth, overhead time, and latency tolerance. It's possible that the concept of the processor as we know it will disappear as aggregates of finer-grained cell-like constructs that integrate logic, state, and data transfer merge into one highly replicated element. Power and reliability through active reconfiguration (graceful degradation) will become as important as optimized throughputs. Computation will have to be abstracted (virtualized) with respect to the underlying physical execution medium to let it adapt to the constantly shifting organization. SOC, SMP on a chip, and PIM will become important elements that bring memory closer to logic.

Simultaneously, Logic Intensive Processor Architectures (LIPAs) such as Stanford's streaming architecture, the University of Texas at Austin's Trips (for tera-op reliable intelligently adaptive processing system) architecture, and Cray's Cascade architecture will exploit large internal arrays of ALUs. New technologies could permit systems such as the Hybrid Technology Multithreaded (HTMT) architecture to achieve far higher densities of computation, as would 3D packaging (capable of putting 500,000 chips in a cubic meter). For important applications, special-purpose devices could still play a role, and field-programmable gate arrays might prove of value to make the applications more accessible and general. High-bandwidth optical communication, perhaps even directly connected to the chips, will permit bisection bandwidths well beyond many petabits per second across a system. The choices are so rich—and the driving motivation so compelling—that COTS-based systems will go the way of the mainframe uniprocessor at some point. Supercomputers and desktops just aren't the same thing, in spite of the fact that they both perform calculations.

Centers in the 21st Century

Computing centers evoke images of white lab coats and large front panels behind layers of glass—mere mortals had limited or indirect access. These centers housed the largest processing and storage facilities, cost millions of dollars, and some even contained the ultimate high-IQ system: the supercomputer. For batch processing, access was through submitted decks, either physical (punched cards) or virtual (jobs submitted from terminals). Some of us remember keypunch and terminal rooms managed by the central institutional computer centers of the day. As minicomputers, workstations, and ultimately PCs incrementally permeated the computing community, the computer center's role evolved and narrowed, but it retained its critical contribution to large-scale computation. However, with the cluster's emergence—particularly Beowulf-class systems—some contend that the computer center as an institution is at its end.

Without question, commodity clusters and Beowulfs have resulted in local sites obtaining, applying, and maintaining systems with capabilities ordinarily reserved for the pristine conditions of the classic machine room; this is often accomplished through minimal upfront costs, leveraging extant talent for system support services. For certain contexts, such as academic environments and research laboratories, this trade-off works well for systems of a few dozen to a couple of hundred nodes, but beyond that, resources are usually stressed, sometimes severely, especially when such clusters are shared among several users and applications. As few as 50 nodes can demand a full support person. Although this seems a little high, at some level, managing a commodity cluster can become a full-time job.

Mass storage can be an important part of a system's capability, even one assumed to be dedicated to compute-intensive applications. Large archival tertiary storage facilities are becoming increasingly valuable to a full-service scientific computing environment and can require expert administration. Similarly, networking of cluster systems to the external user base, either within an administrative domain or over the Internet, adds to the responsibilities of such systems. Software upgrades, hardware diagnosis and replacement, and account, job, and user interface management all entail significant usage of time and talent in maintaining a large cluster.

Many labs, groups, and organizations can now acquire—within their budgets—a cluster system capable of substantial performance, but they're often not prepared to engage the resources or fulfill the responsibilities of maintaining and managing a complex comput-

ing facility. To address this gap, the computer center can be redefined to manage a large cluster for an owner organization, providing the expertise, infrastructure, and environment necessary for maintaining continued cluster operation. These facilities can be amortized across a diversity of systems, thus limiting the burden on any one system, permitting rapid deployment and high availability, avoiding the learning curve of untrained administrators, and simplifying decommissioning at the end of the system's lifetime.

Moreover, placing a moderately sized commodity cluster in the same administrative facility with other comparable systems enables their synthesis to form superclusters, the peak capability of which can be brought to bear on significant user applications by common agreement and shared protocols. It also makes available large data storage reservoirs within the computer center that might not otherwise be accessible. Revamped computer centers will complement the commodity cluster's strengths, extending its price-performance advantage by leveraging investment in the needed administrative facilities while providing very large online data archives that anyone can access. In all likelihood, computer centers will remain—if not grow in importance—in response to the new trends in cluster computing.

Bell and Gray touch on some potential future directions that might drive high-end computing through the end of this decade and beyond, thus justifying investment (of time and funding) by industry, government sponsors, and the research community. We share much of their view, but we want to emphasize certain distinctions. Although we agree that commodity clusters will play an important role for the foreseeable future, research in this area is still required. Packaging, interconnection networks, and system software, as well as latency-tolerant algorithms and fault tolerance, are areas demanding further pursuit. System software research in particular must establish fully supportive system-wide environments for resource management, administration, and programming, including tools for correctness and performance debugging.

The restriction of devising systems constrained to comprise mostly COTS components precludes the innovation critical to achieving high efficiency as well as programmability and reliability. Custom architecture is an important opportunity to pursue, in spite of the conventional wisdom that dismisses specialty designs as infeasible in today's market climate. A new class of processor architecture intended for a role in highly parallel systems must be simple in design, permitting a short design cycle as well as easy modeling, simulation, debugging, and compilation.

With a more strict definition of a cluster, we envision a continued evolution of the computer center's role—one in which it will adapt to the new requirements and opportunities of parallel system classes while providing massive data archival stores.

References

1. G. Bell and J. Gray, "What's Next in High-Performance Computing," *Comm. ACM*, vol. 45, no. 2, 2002, pp. 91–95.
2. E. Strohmaier et al., "The Marketplace of High-Performance Computing," *Parallel Computing*, vol. 25, Dec. 1999, pp. 1517–1544.
3. W. Gropp, E. Lusk, and T. Sterling, *Beowulf Cluster Computing with Linux*, MIT Press, 2003.
4. T. Sterling, P. Messina, and P.H. Smith, *Enabling Technologies for Petaflops Computing*, MIT Press, 1995.
5. G. Pfister, *In Search of Clusters*, Pearson Education, 1997.

Jack Dongarra is a university distinguished professor at the University of Tennessee and Oak Ridge National Laboratory. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. Dongarra received a PhD in applied mathematics from the University of New Mexico. He is a fellow of the AAAS, ACM, and the IEEE and a member of the National Academy of Engineering. Contact him at dongarra@cs.utk.edu .

Thomas Sterling is a faculty associate at the Center for Advanced Computing Research at the California Institute of Technology and a principal scientist at the NASA Jet Propulsion Laboratory. His research interests include high-performance computing systems, processors in memory architecture, Beowulf computing, and parallel system software. Sterling received a PhD in electrical engineering from the Massachusetts Institute of Technology. He is a member of the IEEE and the ACM. Contact him at tron@cacr.caltech.edu.

Horst Simon is associate laboratory director for computing sciences and director of NERSC at Lawrence Berkeley National Laboratory. His research interests include the development of sparse matrix algorithms, algorithms for large-scale eigenvalue problems, and domain decomposition algorithms for unstructured domains for parallel processing. Simon has a PhD in mathematics from the University of California, Berkeley. He is a member of SIAM and the IEEE Computer Society Contact him at HDSimon@lbl.gov.

Erich Strohmaier is a computer scientist in the Future Technologies Group at Lawrence Berkeley National Laboratory. His research interests include performance characterization of scientific application, performance evaluations of HPC systems, and computer system benchmarking. Strohmaier has a Dr.rer.Nat. in theoretical physics from the University of Heidelberg. He is a member of the AAAS, the ACM, and the IEEE. Contact him at Estrohmaier@lbl.gov.