

The Grid: Blueprint for a New Computing Infrastructure

Edited by

Ian Foster

Argonne National Laboratory
and the University of Chicago

Carl Kesselman

Information Sciences Institute,
University of Southern California



Morgan Kaufmann Publishers, Inc.
San Francisco, California

Senior Editor Denise E. M. Penrose
Director of Production and Manufacturing Yonie Overton
Assistant Production Manager Julie Pabst
Editorial Coordinator Meghan Keeffe
Copyeditor Ken DellaPenta
Text Design, Illustration, and Composition Windfall Software
Color Insert Preparation Side by Side Studios
Proofreaders Gary Morris and Christine Sabooni
Indexer Steve Rath
Cover Design Ross Carron Design
Cover Image Minoru Toi/Photonica
Printer Courier Corporation

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances where Morgan Kaufmann Publishers, Inc. is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Color Plate credits: Plate 2, courtesy D. Porter, S. Anderson, J. Habermann, and P. Woodward, Laboratory for Computational Science and Engineering, University of Minnesota. Plate 3, courtesy Jason Leigh, Electronic Visualization Laboratory, University of Illinois at Chicago. Plate 4, courtesy of Maria Roussos, Andy Johnson, Jason Leigh, Craig Barnes, Chris Vasilakis, and Tom Moher, Electronic Visualization Laboratory and Interactive Computing Environments Laboratory, University of Illinois at Chicago. Plate 6, courtesy Marcus Thiebaut, ISI. Plate 8, courtesy Randy Hudson, ANL; data courtesy Andrea Malagoli, University of Chicago. Plate 9, courtesy Matthew Szymanski and Michael Papka, ANL; data courtesy of Louisiana State University's Concurrent Computing Laboratory for Material Simulations. Plate 13, courtesy Joe Insley and Michael Papka. Plate 15, reprinted, by permission, from [254] ©1995 IEEE. Plate 16, courtesy Doug Kimmelman, IBM. Plate 17, courtesy Tamara Munzner, Eric Hoffman, K. Claffy, and Bill Fenner; partial support for the work was provided by the NSF.

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA
Telephone 415 / 392-2665
Facsimile 415 / 982-2665
Email mkp@mkp.com
WWW <http://www.mkp.com>
Order toll free 800 / 745-7323

© 1999 Morgan Kaufmann Publishers, Inc.
All rights reserved
Printed in the United States of America

03 02 01 00 99 5 4 3 2

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of the publisher.

Library of Congress Cataloging-in-Publication Data is available for this book.
ISBN 1-55860-475-8

Computing Platforms

Andrew A. Chien

Given that high-performance distributed computing systems are constructed by integrating diverse end systems, it is important to understand key characteristics of these systems with respect to both current and expected future capabilities. This chapter characterizes end systems as basic elements and simple composite elements. In fact, the similarity between a national computational grid and simple composite elements reflects the “fractal nature” of grids, as discussed in Chapter 1. Hence, it is interesting to review and contrast the solutions adopted.

We begin by discussing basic elements, including computing elements, computer systems (e.g., processors and memory), communication elements (e.g., networks), and storage elements (e.g., attached disk and network attached storage). Subsequently, we describe simple composite elements (SCEs) which are collections of basic elements, aggregated with software and sometimes special hardware to provide a qualitatively different interface and capability. Examples of simple composite elements include high-throughput, high-reliability, dedicated high-performance, and shared controllable SCEs. These composite elements can be employed in functions suited to their capabilities—reliable SCEs as name servers, for example—in computational grids. We describe each element type in turn, outlining the current state of the art and outstanding research challenges.

NAS

17.1 BASIC ELEMENTS

The capabilities of basic elements—building blocks for all computing systems—have improved at geometric rates for the past three decades (see Figure 17.1). Such rapid change produces not only tremendous quantitative changes in capability (1,000–10,000 times), but also qualitative changes in capability. Computer systems were once multiton behemoths the size of a small building, and now they are wristwatch-sized gadgets. Multiplying the revolutionary changes enabled by size reductions are equally dramatic increases in storage, compute power, and networking capability. We briefly outline here the continuing revolutionary changes in the technology of basic computing elements.

17.1.1 Computing Elements

Computing elements, the power sources for a computational grid, have evolved at a tremendous rate over the past 20 years and are expected to continue their rapid evolution over the next 10 years. Since the introduction of the microprocessor in 1971 with the Intel 4004, the scale of integration, transistor count, and clock speed have increased rapidly in accord with Gordon Moore's observation in 1965, now commonly known as Moore's law, that the number of transistors would increase exponentially.

$$\text{Transistors} = 20 \times 2^{(yr-1965)/1.5}$$

This law predicts continued doublings every 18 months in our capability to integrate transistors onto a single chip. Compounded over 20 years, this geometric increase in performance has brought us from microprocessors such as the Intel 8080 in 1975 with 4,500 transistors to the Pentium II in 1997 with 7.5 million transistors—a far greater number of transistors than the total number in many entire computer systems in the 1970s. The geometric increase in integration capability has fueled a rapid increase in microprocessor performance as codified, where delivered performance (as measured by MIPS or SpecMarks) increases by 1.5 times each year. Over the past 20 years, microprocessor performance has increased from approximately 100,000 instructions per second in 1975 to nearly 1 billion instructions per second today. Advances in cost-performance have been even more dramatic; technology advancements coupled with high-volume production (tens of millions of units) have driven processor prices down rapidly.

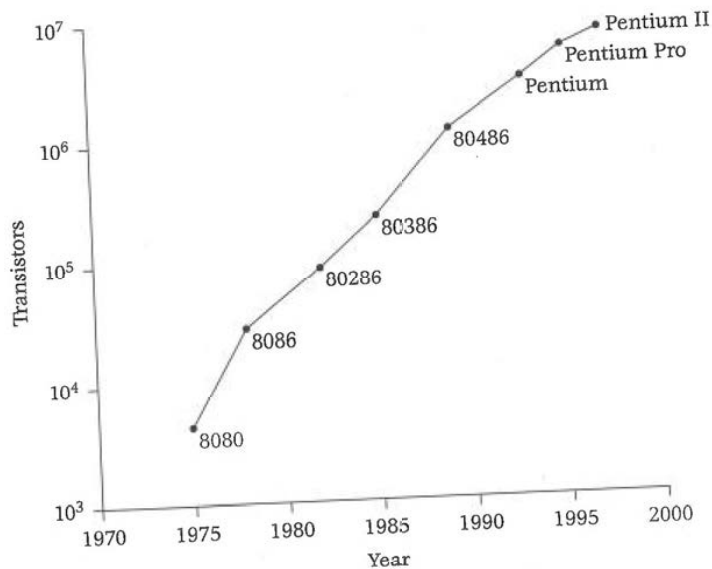
17.1
FIGURE

Transistors

G
p

crea
men
paci
drar
plos
and
pro
adv
ers,
ma
(se

ad
sor
ch

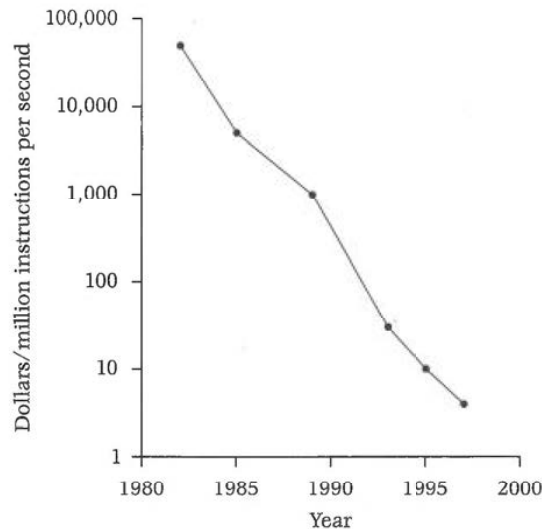


17.1
FIGURE

Geometric increase in the number of transistors integrated on a single microprocessor chip (Intel microprocessors).

Advances in microelectronics and packaging not only have fueled increased processor performance, but they have produced geometric improvements in system price-performance. System capacities—main memory capacity, disk capacity, and internal system bandwidths—have all increased dramatically. In addition, system prices have plummeted, producing an explosion in volume for computers with Pentium systems retailing under \$1,000 and PC volumes approaching 100 million units in annual sales. The rapid progress shows no sign of abating in the next decade [504]; technological advances in microelectronic feature size and new markets (home computers, handheld devices, wireless devices, embedded computers, and worldwide markets) continue to drive a virtual cycle of lower cost and higher volume (see Figure 17.2).

In the past 10 years, high-performance systems have also leveraged the advantages of high-performance microprocessors. Massively parallel processors such as the Intel iPSC, Ametek 2010, Transputer Arrays, Thinking Machines CM-5, Intel Paragon, IBM SP-2, Cray T3D/T3E, Intel/Sandia Teraflop



17.2
FIGURE

Technology improvements and high-volume markets drive the geometric decrease in system cost-performance (based on personal computer prices).

machine, and SGI Origin2000 have all exploited ensembles of low-cost, high-performance microprocessors to deliver high performance. In addition, commercial systems have made heavy use of symmetric multiprocessors, SGI's Power Challenge, HP/Convex's Exemplar Series, Digital's Alpha Servers, and Sun's Ultra Enterprise systems, built from high-performance microprocessors to deliver performance on database and other commercial applications. Recent scalable shared-memory systems from these vendors include SGI's Origin2000 and HP/Convex's SPP2000. These systems all benefit from the continued rapid improvement in underlying microprocessor technologies.

Computing performance for microprocessors over the next 10 years will continue to improve at a geometric rate [504]. This improvement will be further accelerated by increasingly effective use of large-scale parallelism in high-end server systems. However, to view this increase merely as a continuation of the past trends understates the rate of change: after 20 years of increase, the base is much larger. Thus, in quantitative terms, the relative size of increase in hardware resources and capability will be much greater. For example, in the next 10 years, we will go from microprocessors with 10 million transistors to those with nearly 1 billion transistors [504]; from volume production of mem-

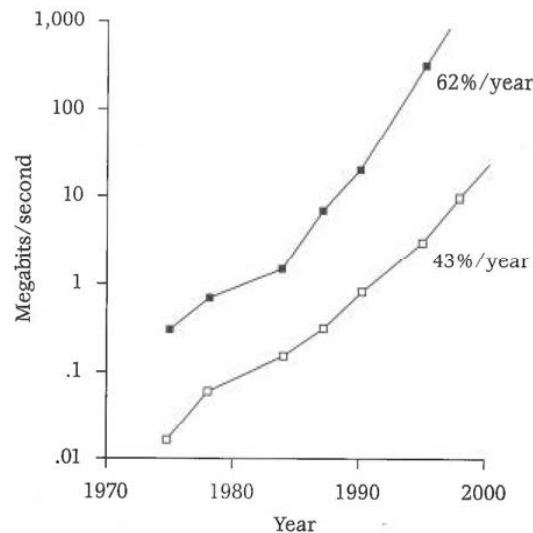
ory chips with 64 MB to 4 GB; and from systems that can manipulate rich text and simple 2D graphics to systems that easily manipulate a range of complex data and real-world inputs, generating rich, high-modality outputs (e.g., 3D immersive virtual reality). In short, the large quantitative change will push computational capability through many thresholds in capability, producing a large qualitative change in system capabilities.

17.1.2 Communication Elements

Communication technologies have also improved geometrically over the past 20 years, producing communication elements with breathtaking capabilities (see Figure 17.3). Thus, the technology base available for networking elements with terabit performance is available today. However, the rate of adoption of such technologies has been slowed, first by its cost and also by the fundamental nature of communication to involve multiple parties. Thus, communication depends on an infrastructure of links, routers, switches, and meta-information structures such as name servers. Infrastructures evolve slowly, as they require large-scale systematic upgrading and investment. Consequently, deployed network performance lags behind our technological capabilities significantly.

Rather than focus on wide area issues that are well covered in Chapter 21, we focus on localized networks that can evolve rapidly in response to local needs. These are most germane to the design of simple composite elements. In particular, we focus on technologies for cluster networks, which are rapidly finding their way into local area, metropolitan area, and corporate intranet infrastructures. These localized networks form subgrids that could be incorporated into a national-scale computational grid. In this area, we expect deployment of revolutionary technology over the next 10 years.

After many years of modest advances, over the past few years we have seen a rapid advance from shared 10 Mb/s networks to 100 Mb/s and switched 10 Mb/s systems. This change is being driven by file service, email, and Web surfing. We expect this change to accelerate with the increasing use of wide area information systems catalyzed by the Web. Future transitions will be driven by rich multimedia content and perhaps novel applications that are not yet extant. High-speed environments based on ATM and system area networks have been the primary vehicles for research on high-speed networks with gigabit technologies over the past five years, and the lessons learned from this research now form the basis of emerging standards such as Gigabit Ethernet and the Virtual Interface Architecture (VIA). To provide the relevant context, we next describe a number of innovative system area networks.



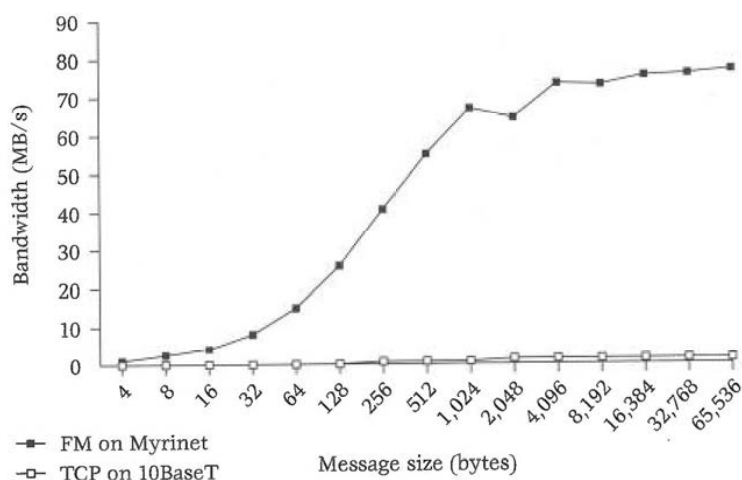
17.3
FIGURE

Geometric increases in communication technology far outstrip their deployment. The upper curve tracks *demonstrated* technology and the lower *deployed* technology for a single fiber.

Cluster Networks

Innovation in cluster networks or system area networks has produced novel link, interface, and software layer technologies. Cluster networks are physically localized, high-speed, and generally low-volume products. Because they are part of a system and therefore are subject to weaker requirements of compatibility, innovation is more rapid. In addition, because their performance is tightly coupled to cluster applications and systems, innovations in network interface design can directly improve system performance. To give an idea of the networking capabilities that underlie current cluster research, we describe a number of networking interfaces.

Myricom's Myrinet is a high-speed local-area network with full-duplex, 1.28 Gb/s links. Derived from multicomputer routers, current Myrinet offerings employ wormhole routing and 8×8 crossbar switches [66]. Arbitration in the crossbars is fair and done by a round-robin algorithm, and switch latency is below $1 \mu\text{s}$. The network interface has a programmable CPU (the LANai) with 0.5–1.0 MB of SRAM that attaches to the I/O bus of the host processor.



17.4
FIGURE

User-level access and lightweight protocols allow Fast Messages to deliver usable gigabit performance to APIs. Peak bandwidths of 77 MB/s per second, a half-power message size of less than 256 bytes, and latencies of 11 μ s are all achieved. Reported performance is for 200 MHz Pentium Pro machines connected by 80 MB/s Myrinet network.

The LANai 4.x runs at a range of speeds, matched to an integral multiple (1:1 or 2:1) to the clock speed on the I/O bus. It is single-issue processor and runs at 33 MHz in most PCI bus configurations. It contains three DMA engines, which connect host and LANai memory, LANai memory to the network, and the network to LANai memory. Thus, the LANai can send and receive from the network at the same time.

While Myricom offers device firmware and driver support, it has enabled a wealth of research by opening its interfaces and providing development tools. The research community has produced messaging layers with much higher performance; for example, Fast Messages [431, 432], Active Messages [123], and PM [534] all deliver latencies close to 10 μ s and tens of megabytes per second bandwidth. These layers all map the interface directly into an application's address space, allowing rapid synchronization and data movement between the application and the network. More information about Fast Messages is included in Figure 17.4, and the Illinois HPVMM project is described in Section 17.2.5. For more information on Active Messages, see Section 17.2.4.

Compaq/Tandem's Servernet I [280] is a 50 MB/s full-duplex wormhole-routed network that provides reliable communication. Network operations include put and get operations, essentially network DMA operations to read and write data from remote nodes or I/O devices. Servernet I is based on six-port crossbars and delivers data in packets of 64 bytes. Latency is a few microseconds per switch. Reliability is achieved with symbol-level error detection and link-level CRC. Detected faults are logged and isolated. To further increase availability, each network interface card has two network ports.

Recently, Tandem has announced plans for Servernet II [224], but at this time no concrete schedules for the introduction of Servernet II are available. Servernet II is a 125 MB/s full-duplex network, hosted on 1000BaseX (Gigabit Ethernet) links. It supports a larger packet size, 512 bytes, and 64-bit network addressing. It will employ larger crossbar switches.

A wide range of other networking technologies have been proposed and used for clustering, including Fibrechannel [579], the Scalable Coherent Interface [253], and 100BaseT Ethernet. These are available from multiple vendors and provide a wide range of latencies, bandwidths, and performance guarantees. There are also a number of proprietary interconnects such as IBM's SP-2 [519], as well as several 390 Sysplex interconnects, SGI's Craylink/Spider [220], and Fujitsu/HAL's Mercury [563], which typically employ proprietary interfaces.

Network Interfaces and VIA

With high-speed cluster networks providing the hardware testbeds, there has been a wealth of research in the academic community on low-overhead communication, including Illinois Fast Messages [431], Berkeley's Active Messages [123], U-NET [51], VMMC-2 [172], MINI [255], and Osiris [169]. These efforts have forged a consensus on core requirements for network interfaces to deliver high-communication performance to application programs (see also Chapter 20). This consensus includes four key features: user-level protected network access, hardware support for gather-scatter, early demultiplexing of incoming traffic, and lightweight protocols. These issues are revisited in the broader context of wide area networks in Chapter 20.

High-speed networks require direct application access, as the overhead of operating systems calls is far too expensive. Scatter-gather support can eliminate message copies due to protocol encapsulation. Network interface demultiplexing allows user-level protected access without message copies or operating system involvement. Lightweight protocols eliminate unnecessary

CPU load due to complex legacy protocols for networks with dramatically different properties. The resulting advantages of these systems are illustrated in Figure 17.4, which presents delivered communication performance in the Illinois Fast Messages system [431] and compares it with a good implementation of TCP/IP. The dramatic difference is the accessibility of high bandwidth for small messages, and 20 times reduction in one-way communication latency to 11 μ s.

These consensus features for network interfaces for high-speed networks have formed the basis for the emerging VIA standard, a cluster interface standard proposed by Intel, Compaq, and Microsoft. Specifically, VIA includes user-level protected network access, standard interfaces for specifying gather-scatter, and early demultiplexing of incoming traffic. These features follow closely the consensus for high-performance network interfaces. User-level protected network access is provided through logical communication endpoints. The standard interfaces for gather-scatter enable zero-copy interfaces to protocol stacks, and the protected early demultiplexing enables traffic on different connections to be decoupled without involving the operating system. Thus, VIA presents an important step forward in commercializing and standardizing high-speed network interfaces that can deliver gigabit communication performance to applications. One major shortcoming, however, is that while it specifies the key attributes above, it does not provide a concrete API required for software portability. Thus, commercial implementations may well have distinct (and incompatible) software APIs.

The VIA interface has garnered support from many of the cluster interconnect vendors (Servernet [224, 280], Myrinet [66], etc.) and numerous cluster system vendors and independent software vendors. Thus, it appears likely to become a widely used standard interface for high-performance communication. In addition, vendors are building VIA interfaces for longer-link network standards such as Fibrechannel and 1000BaseX Ethernet, making VIA an interesting future possibility for high-speed LANs and MANs.

Thus, in the communication elements area, the key elements of technological capability are already available, and they can enable dramatic changes in how we share and aggregate digital resources. For example, gigabit network performance makes access to storage (for file systems, databases, and even demand paging) effectively location independent. Further, with network interfaces such as VIA, main memory and even computing cycles can be effectively shared: the cost of moving the associated data is dramatically lower. It appears that future distributed computing can include tightly coupled applications and flexible, even dynamic, associations of computing, storage, main memory, and even network-attached sensors or special devices.

The obstacles to widespread availability of high-speed network performance are largely infrastructure inertia and cost. This area represents a key opportunity for computational grids, which increase the value of network connectivity. One important contribution that grids can make would be to accelerate the deployment of high-bandwidth communication infrastructure because doing so enhances the value of the computational elements. Drawing an analogy to the power grid, it pays to increase the grid capacity to power generators and consumers because it increases the ability of the grid to deliver service and therefore value.

17.1.3 Storage Elements

Storage elements (magnetic disk and tape) [239, 284] continue to improve in both capability and cost-performance at an exponential rate. Because the trends for magnetic disk and tape are similar, we focus only on magnetic disk here. The rate of capacity and performance improvement over the past five years has accelerated. After maintaining a steady rate of improvement in areal density (the number of bits stored per unit area) of approximately 29% a year from 1970 to 1988, recent technology breakthroughs have accelerated the rate of improvement to nearly 60% per year. These improvements, combined with mechanical and packaging improvements (rotational speeds and densities), have accelerated cost-per-byte improvements for hard disks from 40% per year to nearly 100% per year in personal computer systems. Thus, in succession, 500 MB (1995), 1 GB (1996), 2 GB (1997), and 4 GB (1998) have become standard low-cost disk configurations. With continued dramatic growth in the deployment of information technologies, there is every indication that this trend will continue.

While cost per byte and available capacity have improved at nearly 100% per year, performance factors for disk drives have failed to improve at that rate. Peak transfer rates continue to improve rapidly, tied to increases in areal density and small increases in rotational speeds, with current disks such as Seagate's 10,000 RPM Cheetah delivering 10–15 MB/s. However, seek times are improving only slowly, coupled to improvements in mechanical technologies and gradual shrinking of the platters, and therefore the seek distances involved. Average seek times of 7–10 ms remain typical. Thus, in the face of geometric performance improvements in other elements of computing systems, disk access times are essentially constant and therefore a bottleneck of increasing importance.

The rapid decrease in the cost of storage has significant implications for computational grids. Not only do technology improvements make it possible

Machine	Computing	Memory	Disk	Network
<i>5-year machines (2003)</i>				
Personal computer	8 GIPS	1 GB	128 GB	1 Gb/s
Supercomputer	80 TIPS	10 TB	1,280 TB	10 Tb/s
<i>10-year machines (2008)</i>				
Personal computer	64 BIPS	16 GB	2 TB	10 Gb/s
Supercomputer	640 TIPS	160 TB	20 PB	100 Tb/s

17.1

Future capabilities of personal computers and supercomputers based on recent growth rates.

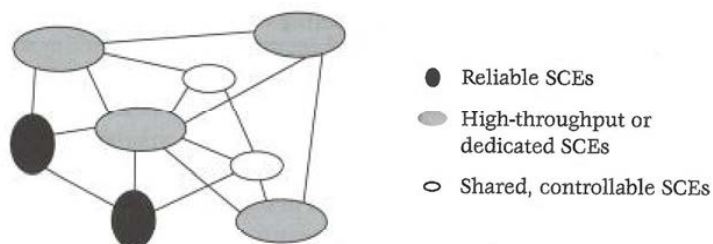
TABLE

and cost-effective to store large quantities of data, but the exploding storage capability in lowest-cost systems dictates that huge amounts of storage will be available on networks of any size, and the improving capabilities of high-speed networks will enable grids to use that storage flexibly. For example, in our local PC-based instructional laboratory at the University of Illinois, the file server is configured with 16 GB of disk. However, the 100 desktop machines associated with that server contain a far larger quantity—over 200 GB. This reflects the new reality that much of the real cost of storage is administration and backup. Thus, uses of storage will change. In our PC laboratory, only the file server's disks are archived each night; the majority of online storage is considered scratch space. Clearly, current management and administration techniques will change to exploit this excess capability. Further discussion of this topic and its relation to grids can be found in Chapter 5.

17.1.4 Basic Elements of the Future

To get a perspective on the dramatic exponential improvement of capabilities in basic elements, it is useful to extrapolate the growth rates to 5-year and 10-year horizons. Table 17.1 shows the basic capabilities of our personal computers and supercomputers (10,000-processor parallel machines) in the years 2003 and 2008 if recent historical growth rates in performance and capability continue.

The extrapolations for networks are probably the least accurate, since technology is readily available, but technology deployment rates are unpredictable. The networking rates are based on estimated network interface speeds in standard high-volume configurations. Thus, they form an upper



17.5
FIGURE SCEs nested in a larger grid. Each type of SCE can play a unique role in providing a cost-effective grid.

bound and reflect the likely available performance in local area networks, with accessible performance in metropolitan area and wide area networks limited by advances in national and global initiatives in infrastructure efforts such as Internet II [518] and its successors (see also Chapter 21). For networks within supercomputer configurations, the available network bandwidths may be 5–10 times greater because of the use of specialized hardware.

17.2 SIMPLE COMPOSITE ELEMENTS: LOCAL GRIDS

Simple composite elements are richly connected, relatively homogeneous collections of basic system elements (compute, memory, communication, and storage). They are often single administrative domains and in many cases are already thought of as a single system. As illustrated in Figure 17.5, SCEs (or clusters, in the terminology of Section 2.5) are the building blocks for wide area, national, and international grids. SCEs are worthy of particular study for these reasons.

1. Local grid technologies can reduce the number of problems higher-level computational grids must solve.
2. Local grids use resources and software to implement the external properties of the composite element affecting its utility or integration into larger grids.
3. Local grids form the basis for the larger computational grid. Thus, their evolution is an integral part of the challenges in building larger grids.

Together, these integral relationships make understanding technologies for SCEs and their capabilities a crucial element of understanding issues in building grids. For example, in a national computational grid, reliable composite elements can be used to provide management (access control and scheduling) and basic services (naming and routing). Other composite elements can provide resource pools with distinct capabilities such as network data caching, persistent storage, and predictable computation power. However, composing two SCEs together may be challenging if they correspond to different administrative domains or employ distinct data representations or different network protocols.

In the following sections, we first describe the two key distinguishing features of SCEs: their external interfaces and guarantees and their hardware requirements. External interfaces and guarantees affect the use and utility of SCEs in the larger grid context. Hardware requirements determine the SCEs capabilities that can be exploited for building computational grids. Then, we describe a series of state-of-the-art SCEs and technologies: high-throughput clusters, reliable clusters, dedicated high-performance clusters, and shared controllable-performance clusters. Finally, we close with a discussion of the current capabilities, challenges, and future directions.

17.2.1 Capabilities and Requirements

SCEs are defined by their external interfaces and guarantees and by their internal hardware requirements and their ability to deliver efficient, flexible use of that hardware to applications. Since other chapters discuss managing interfaces and guarantees in computational grids (see Chapter 11), we focus on providing a classification here. In addition, because an SCE technology's internal hardware requirements and capabilities are integrally related to its applicability, we also provide a classification of hardware requirements. Together, the two classifications delineate both current-day SCE systems and many currently under research.

External interfaces and guarantees define how SCEs are used by applications and how they can be integrated into larger grids. Five attributes capture the important distinctions among a wide range of SCEs: capacity, aggregate performance, reliability, predictability, and sharability. *Capacity* corresponds to the total throughput of the SCE in dimensions of compute, memory, communication, and storage. *Aggregate performance* corresponds to the SCE's capability to deliver aggregated compute, memory, communication, and storage performance. *Reliability* is defined as the likelihood of unavailability of resource or unavailability or loss of data. *Predictability* captures an application's

capability to predict the delivered capacity or performance. *Sharability* captures whether the SCE can be shared, integrating a number of computations on one resource for tighter coupling or simply multitasking.

Hardware requirements and capabilities distinguish the range and capability of the technologies used to build an SCE. These constrain the range of an SCE and distinguish it from higher-level grids. Five attributes capture many important distinctions in applicability: heterogeneity, networking requirements, distributed resources, changes in constituent systems, and scalability. *Heterogeneity* in compute, networking, and storage elements influences the inclusiveness of an SCE environment and its ability to include both legacy and new systems. *Networking requirements*—special hardware, link length limited, high bandwidth, and so on—all limit the locales and cost constraints under which an SCE technology can be deployed. Whether an SCE technology can exploit *distributed resources* (links tens of meters or thousands of kilometers) limits the geographical extent of the SCE. Whether an SCE technology requires *changes in its constituent systems* has a significant impact on the deployment requirements, technology insertion cost, and therefore deployability of a technology. Finally, *scalability* of a system influences the number of nodes that can be deployed and the SCE's ability to manage and deliver their performance.

In the following, we use a two-part framework—external interfaces and hardware requirements—to understand and distinguish the wide range of cluster systems and systems of systems that have been built by both researchers and commercial vendors. Each type has distinct capabilities and provides different challenges and advantages for integration into a grid environment. They include high-throughput, high-reliability, dedicated high-performance, and shared controllable-performance SCEs.

17.2.2 High-Throughput SCEs

In high-throughput computing, pooled resources are utilized to achieve high throughput on a set of compute jobs [348, 587]. Systems such as Condor, Utopia, and Symbio manage clusters of workstations as pooled resource servers with the primary application being compute-bound sequential jobs. While all three of these systems provide cluster access and resource management facilities, Condor also attempts to increase the pool of available resources (see Chapter 13 for more details).

External Interfaces and Guarantees

The external interface of a high-throughput cluster is high capacity for computation and a sharable resource. In fact, most such systems share resources with interactive users, and some include elaborate mechanisms for ensuring good interactive response. While these systems provide no special support for aggregate performance, interfaces for loosely coupled parallel computing such as PVM [227] are now becoming available. No special support for reliability or predictability is provided.

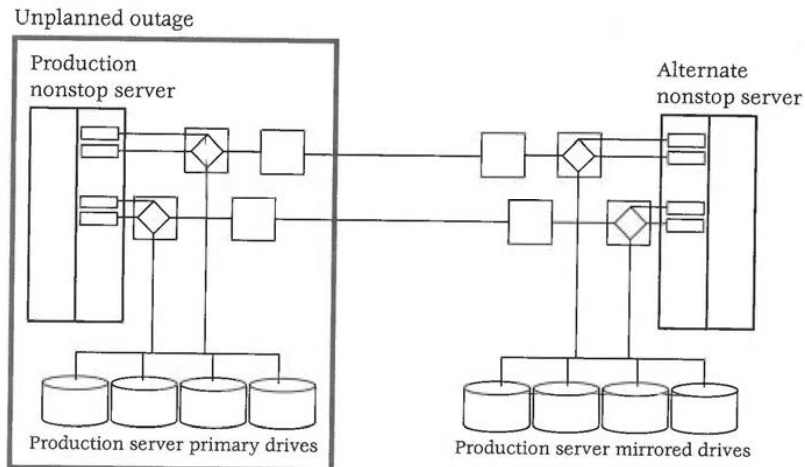
Hardware Requirements

High-throughput SCEs have minimal hardware requirements, running on a wide range of processor and network environments, and tolerating both processor and network heterogeneity in type and speed. High-throughput systems are also used on widely distributed resources, such as for pooling workstation resources across the worldwide sites for a corporation. In addition, high-throughput SCEs do not require significant change to the underlying systems (depending only on some common job controls and special system libraries) and can scale to larger numbers of processors (hundreds to thousands) with little difficulty.

High-Throughput SCEs in Grids

High-throughput SCEs are flexible, powerful systems for achieving high throughput on large numbers of sequential jobs. Thus, they are well-matched grid elements for such tasks. Systems manage a wide range of heterogeneity automatically (instruction set, memory configuration, network, etc.) and schedule compute resources efficiently to reduce turnaround time for jobs. In addition, effective sharing of resources with interactive users increases the pool of resources available to the SCE dramatically. (A more detailed discussion of high-throughput computing can be found in Chapter 13.)

However, because high-throughput SCEs primarily focus on processing large numbers of small but compute-bound jobs, such SCEs do little to efficiently aggregate resources for larger computations, enhance reliability, and improve performance predictability. Each of these issues is addressed by at least one of the other types of SCEs described next.



17.6

A Tandem failover cluster with mirrored disks.

FIGURE

17.2.3 High-Reliability SCEs

High-reliability SCEs provide computational resources with extremely low probability of service interruption and data loss [245]. In high-reliability SCEs, commonly called *reliable clusters*, additional computing resources are deployed to replicate the state of an application, and responsibility for the computation is “failed over” atomically in the case of software, hardware, or any other failure. An example of a reliable cluster is shown in Figure 17.6. Failover transfers responsibility for the computation to the additional hardware, which takes up the task seamlessly, so clients see no interruption of service. This approach, typified by Tandem’s Guardian system [245], has been adopted by a wide variety of vendors for highly available systems.

External Interfaces and Guarantees

Reliable clusters use replication for reliability, but can also add resources for scalability for many kinds of applications. However, with the exception of a few large data manipulation applications, the scalability is generally used to increase system capacity, not to aggregate on a large job. Of course, reliable clusters provide a reliability guarantee to applications and are generally a sharable resource. However, because of failover delays and dynamic load

sharing, most reliable systems do not provide strong guarantees of predictable response.

Hardware Requirements

Reliable clusters generally prefer compatible hardware to enable failover, data sharing, and convenient restoration from checkpoints. However, for cold standbys, less powerful configurations can be deployed to reduce cost, provided lower performance is tolerable in a failover situation. Custom networking is employed among cluster nodes and between primaries and standbys to ensure fault detection and isolation at the earliest possible time. Reliable clusters can be physically localized or distributed over a wide area network. Traditionally, reliable systems used special operating systems (e.g., Tandem's NonStop kernel), but many recent systems have been implemented as a middleware layer, so system change is no longer required. Finally, reliability clusters can also include multiple nodes for scalability in capacity.

High-Reliability SCEs in Grids

High-reliability elements are a natural choice for simple composite elements. The internal substructure of such elements is encapsulated, allowing them to be viewed as reliable, high-capacity systems. Such systems can provide a wealth of important services reliably, facilitating rigorous reasoning about operation, bootstrap procedures, reconfiguration, failure modes, and so on for grid systems.

17.2.4 Dedicated High-Performance SCEs

Dedicated high-performance SCEs merge basic elements into a single resource pool for computing, memory, and storage, allowing these resources to be applied to a single computation. Since microprocessors have become the fastest processors available, collections of microprocessors (massively parallel processors) or entire systems (scalable clusters or networks of workstations) have become an attractive and cost-effective way to achieving high performance [246, 519, 26, 285, 121, 54]. By employing standard workstation or PC building blocks and scalable networks, these dedicated high-performance clusters can be scaled to arbitrarily large configurations. These systems were initially applied to supercomputing tasks and were operated as dedicated systems, with space sharing used to run two or more applications simultaneously. To connect hundreds or thousands of nodes together with high efficiency, dedicated high-performance systems employ high-speed custom networks with

sharing, most reliable systems do not provide strong guarantees of predictable response.

Hardware Requirements

Reliable clusters generally prefer compatible hardware to enable failover, data sharing, and convenient restoration from checkpoints. However, for cold standbys, less powerful configurations can be deployed to reduce cost, provided lower performance is tolerable in a failover situation. Custom networking is employed among cluster nodes and between primaries and standbys to ensure fault detection and isolation at the earliest possible time. Reliable clusters can be physically localized or distributed over a wide area network. Traditionally, reliable systems used special operating systems (e.g., Tandem's NonStop kernel), but many recent systems have been implemented as a middleware layer, so system change is no longer required. Finally, reliability clusters can also include multiple nodes for scalability in capacity.

High-Reliability SCEs in Grids

High-reliability elements are a natural choice for simple composite elements. The internal substructure of such elements is encapsulated, allowing them to be viewed as reliable, high-capacity systems. Such systems can provide a wealth of important services reliably, facilitating rigorous reasoning about operation, bootstrap procedures, reconfiguration, failure modes, and so on for grid systems.

17.2.4 Dedicated High-Performance SCEs

Dedicated high-performance SCEs merge basic elements into a single resource pool for computing, memory, and storage, allowing these resources to be applied to a single computation. Since microprocessors have become the fastest processors available, collections of microprocessors (massively parallel processors) or entire systems (scalable clusters or networks of workstations) have become an attractive and cost-effective way to achieving high performance [246, 519, 26, 285, 121, 54]. By employing standard workstation or PC building blocks and scalable networks, these dedicated high-performance clusters can be scaled to arbitrarily large configurations. These systems were initially applied to supercomputing tasks and were operated as dedicated systems, with space sharing used to run two or more applications simultaneously. To connect hundreds or thousands of nodes together with high efficiency, dedicated high-performance systems employ high-speed custom networks with

limited physical extent (tens of meters). These networks employ parallel data links and custom signaling to deliver high performance, as with the custom cluster networks described in Section 17.1.2.

For example, the IBM SP-2 and Intel/Sandia Teraflop systems both make use of high-volume microprocessors as their basic computation engines. The IBM SP-2 system goes further, employing entire workstations as the basic building blocks, while the Intel/Sandia system employs special system boards and packaging. Both systems use custom high-performance interconnects, delivering $\approx 5\text{--}100\text{ MB/s}$ of network bandwidth to each node in the system and latencies of $20\text{--}100\text{ }\mu\text{s}$. The IBM SP-2 system uses a standard AIX (IBM's UNIX) workstation operating system and a collection of middleware to provide job scheduling, program loading, file input/output, and so on. Allowing a single job on each node enables high performance on dedicated jobs—direct access to networks, management of local memory, and so on. The Intel/Sandia system employs a custom operating system, PUMA [246], which provides a bare run time on each node, freeing the maximum resources for the user computation. Standard services such as multitasking and virtual memory are not provided on the compute nodes. Instead, all resources on a compute node are dedicated, enabling explicit resource management by the application. Required operating system services such as file access and external network input/output are hosted on system service nodes.

The predominant programming model on these systems is explicit message passing, typically via a standard interface such as MPI. This model enables the achievement of high performance at the price of explicit programmer management of naming and data movement. Higher-level interfaces, such as HPF and distributed shared memory, are used to a lesser degree. For a wide range of applications, high-performance MPI implementations have been constructed, so for these applications, dedicated high-performance SCEs can effectively aggregate their compute performance. The systems all provide single-system image (uniform monitoring, resource usage, file system access, etc.). Distributed shared-memory systems have demonstrated techniques for efficient memory pooling [341, 176], although significant questions remain about how to manage and share such pools as well as how to best implement virtual memory in such an environment. Efficient scheduling remains a difficult challenge, as schedulers typically focus on processors, utilizing memory, network, disk, and other resources poorly.

Berkeley Network of Workstations

The University of California at Berkeley's network of workstations (NOW) is a well-documented research system designed as a dedicated high-performance

SCE. It consists of Sun UltraSPARC systems linked by Myricom's Myrinet [26]. At the project's initiation, the objective was to deliver supercomputer performance based on high-volume workstation building blocks. To this end, the NOW project built three key pieces of software: GLUnix [25], xFS [27], and Active Messages [123]. Subsequently, the research focus of the NOW has broadened significantly; extensions are discussed in Section 17.2.6.

GLUnix is a simple middleware layer that provides queuing, gang scheduling, and resource monitoring essential for users to submit, run, and monitor their jobs. This functionality is equivalent to commercial packages such as Platform Computing's Load Sharing Facility (LSF), with the exception of GLUnix's gang scheduling, which enables good performance on tightly coupled parallel jobs. The goals for GLUnix remain much more ambitious, including automated resource parameterization, standard mechanisms for layering operating systems extension, fault tolerance, and system management. However, many of these goals remain unrealized to date.

xFS is a scalable parallel file system built running across the Berkeley NOW [27]. Key ideas include parallel disk activity, achieving a consistent recoverable view of the file system, and managing available file system space. xFS combines ideas from log-structured file systems with disk striping to demonstrate promising scalability and performance.

Active Messages (AM) is a high-performance low-level messaging layer that provides low-latency communication [553]. AM was first implemented on the Thinking Machines CM-5 and demonstrated that dramatically lower communication costs could be achieved by removing buffering, interrupts, thread synchronization, and rich user APIs. The AM implementation of the Berkeley NOW delivers latencies near 10 μ s, and bandwidths approaching 40 MB/s (limited by underlying I/O bus hardware performance). This supercomputer-class communication performance is delivered to applications via MPI, a modified sockets interface, and the Split-C programming language.

External Interfaces and Guarantees

Dedicated high-performance SCEs aggregate resources to speed individual computations and are scalable to hundreds or thousands of nodes. Thus, they provide both pooled capacity for sequential jobs and high performance for parallel computations. In fact, systems such as IBM's LoadLeveler will schedule in combination both uniprocessor and dedicated parallel jobs. However, because they focus on highest single-job performance (supercomputing), dedicated SCEs have not delivered reliability or predictability and are not generally sharable (other than via space partitioning). Note that many commercial reliable cluster products have an element of scalability, but generally do not

deliver the levels of performance described for dedicated high-performance SCEs.

Hardware Requirements

In dedicated high-performance SCEs, aggregate performance is the primary objective, so scalability to hundreds or thousands of nodes is supported. Hardware attributes, such as heterogeneity, that degrade performance are not generally included. Software features that reduce performance (e.g., process pairing for reliability) are also not included. In addition, because the SCE is viewed as a single system, changes to the underlying systems (operating system and motherboard) are sometimes required. Networks and interfaces are virtually always customized. Recently, under pressure from low-cost, high-volume products, many vendors have chosen to use unmodified workstations and operating systems, differentiating only with middleware software with functionality similar to GLUnix.

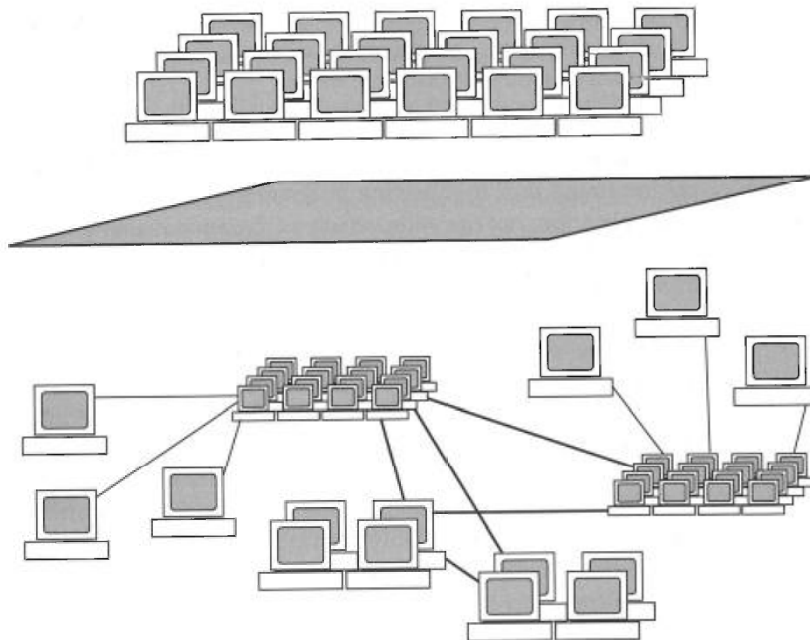
Dedicated High-Performance SCEs in Grids

Dedicated high-performance SCEs can be assets in a grid environment. Indeed, several such systems were employed in the I-WAY demonstration at Supercomputing '95 [155]. However, their dedicated-use model significantly reduces their effectiveness. This observation provides a major impetus for the shared controllable-performance systems described in the next section. Broadening the model of use is essential both for improving resource utilization and for supporting a broader class of resource-intensive online and interaction applications for the future.

17.2.5 Shared Controllable-Performance SCEs

The objective of shared controllable-performance SCEs is to deliver predictable high performance in a shared-resource, heterogeneous, distributed environment. This objective generalizes the usage model of dedicated high-performance SCEs in several dimensions—resource sharing, heterogeneity, distribution, and predictability. Alternatively, shared controllable-performance SCEs can be thought of as extending the high-throughput SCE model with scalable high performance and predictable performance. Thus, shared controllable-performance SCEs combine the capabilities of all the SCEs described except reliability. Thus, shared controllable-performance SCEs include APIs that provide functional portability, as well as resource and performance models to enable performance portability of programs. We term the combined

17
FIG



17.7

FIGURE

HPVMs provide simple, stable, and uniform virtual machine abstraction on top of varied distributed and heterogeneous computing resources. The programmer can think of a regular array of resources, and the HPVM maps that abstraction effectively onto a complex resource environment.

API, resource, and performance model a *high-performance virtual machine* (HPVM). An HPVM has the following attributes:

- ♦ *APIs* that provide functional portability for communication, scheduling, and system resource access.
- ♦ A *performance model* that enables applications to be architected with respect to a balance of operation costs. It includes support for tightly coupled parallelism (e.g., a high-performance, parallel virtual machine). The performance model is faithfully implemented by the HPVM.
- ♦ *Flexible implementations* that can exploit heterogeneous and distributed resources. Variety in configuration, speed, and capability of networks, computing, and storage elements is managed to deliver a high-fidelity performance model.

- ♦ *Adaptive resource management* that enables the HPVM to execute in a shared-resource environment, adapting to the resources available while preserving the performance model.

As shown in Figure 17.7, the idea behind HPVMs is to provide programmers with a simple API and resource execution model. This simplifies the programming task by allowing programmers to focus on the complexity of the application, not the wide variety of dynamic resource situations for every program deployment environment. It also reduces complexity significantly from a typical QoS or realtime system management approach in which tasks and resources are characterized and managed individually. HPVMs adaptively manage the resources available to preserve the agreed-on performance model, insulating the application from underlying resource changes. If cataclysmic change occurs and the resource contract can no longer be maintained, HPVM notifies the applications, allowing appropriate action to be taken.

Construction of effective high-performance virtual machines requires meeting a number of research challenges in high-performance predictable communication, heterogeneity management, performance models, and adaptive resource management. We describe each of these challenges in some detail below:

- ♦ *High-performance predictable communication*: Extending the high-performance communication in dedicated high-performance SCEs to predictability in a shared heterogeneous environment requires inventing new techniques that can reconcile divergent requirements of quality of service and high performance.
- ♦ *Management of heterogeneity*: HPVMs must manage heterogeneity in configuration and type of both computing systems and networks. This capability is essential to maximize both throughput and the resources available to applications and to preserve the performance model for applications.
- ♦ *Performance models*: Defining appropriate performance models that capture relative operation costs is critical for application architectures for HPVMs. Performance models from existing and widely used parallel machines are likely to be appropriate starting points.
- ♦ *Adaptive resource management*: In a dynamic resource environment, resource availability is subject to system load, failures, and changes in priority. New resource management techniques that provide a stable resource model for applications and decide intelligently when to adjust their resources or migrate them must be developed. These techniques must preserve system stability in the face of changing application demands, priori-

ties, and resource availability. Such schedulers complement both application (Chapter 12) and system schedulers (Chapter 13).

These research challenges make the construction of shared controllable-performance SCEs a challenging endeavor. These challenges are significantly beyond the state of the art and other SCEs described in this chapter. Success in meeting these challenges will form the first steps toward the vision of flexibly harnessing network resources for a wide range of large-scale computing tasks (resource-intensive, interactive, and realtime) in a dynamic resource environment with changing availability and priorities.

External Interfaces and Guarantees

Successful shared controllable-performance SCEs can pool resources for capacity and aggregate performance. In addition, they will provide predictable parallel performance for applications and allow flexible sharing of resources. Although the focus is not currently on reliability, doubtless the practical and commercial value of such capabilities would make them a natural next objective.

Hardware Requirements

Shared controllable-performance SCEs will execute on heterogeneous computing and networking hardware. To achieve efficient tight coupling, the network hardware will need to support both low latency and high bandwidth. These HPVMs will execute on distributed resources and will support scalability to thousands of nodes. Again, efficient tight coupling may limit geographic distribution because of fundamental limits imposed on signal propagation velocities (speed of light) in optical fibers. On the issue of system change, it is at present unclear whether such is required. The Illinois HPVM project (see Section 17.2.6) is pursuing an approach that depends only on stock hardware and operating systems. Other recent efforts presume modified operating systems [175, 279].

Shared Controllable-Performance SCEs in Grids

Shared controllable-performance SCEs can be broadly useful resources in grid environments. Well-defined APIs and performance models allow the internal structure of elements to be completely hidden from users. In addition, the sharability combined with predictable performance allows multiple applications to be scheduled and composed on such HPVMs, producing an aggregate that in turn has predictable performance. Such composition is essential for

online, interactive, and realtime high-performance distributed computing applications.

17.2.6 Illinois HPVM Project and Similar Efforts

The goal of the Illinois HPVM project [121] is to develop shared controllable-high-performance SCEs. This goal subsumes all of the goals described above: high-performance predictable communication, management of heterogeneity, stable performance models, and adaptive resource management. The objective is to support not only traditional high-performance computing applications, but also high-performance distributed systems, such as remote visualization, media-rich collaboration, and interactive virtual reality data exploration. By combining high-speed networking with the capability to manipulate large amounts of data as well as perform large amounts of computation, a wide range of existing applications will become distributable, and a new class of distributed applications will be catalyzed.

The HPVM project is building on commodity hardware and software to enable the widest deployability for the high-performance virtual machine environment. Basic parameters for the key hardware and software component technologies are the following:

- ♦ *Computing nodes*: x86 (IA32) and PCI computing systems
- ♦ *Operating systems*: Windows NT and Linux
- ♦ *Networks*: Myrinet, Servernet, and future VIA networks such as Gigabit Ethernet, with lesser performance available on interconnects such as 100BaseT, ATM, and 10BaseT

These component requirements characterize a large fraction of commercially available computing resources, ensuring the potential for widespread deployment. High-speed networks and interfaces are required to support efficient coupling of resources across the cluster. A small number of operating systems bounds the effort to build middleware that coordinates resource scheduling across the SCE.

Based on these underlying technologies, the HPVM project has built the following key software elements to meld resources into a shared controllable-performance SCE.

- ♦ *Fast Messages (FM)*: A high-performance flexible messaging layer (with several implementations) that both delivers underlying hardware performance and supports efficient composition with higher-level communica-

tion layers. On Myrinet, latencies of 11 μ s and bandwidths of 77 MB/s have been demonstrated. FM 2.1 supports multiprocess usage and dynamic network name spaces efficiently, enabling its use in parallel computing and distributed computing applications (see Figure 17.4 and [431] for more information).

- ♦ *APIs (MPI, Shmem Put/Get, and Global Arrays)*: A wide range of standard communication APIs support easy application usage. Research on implementing distributed computing APIs such as sockets and distributed objects (e.g., DCOM and CORBA) with similar performance is in progress.
- ♦ *Coordinated resource management (dynamic coscheduling)*: A range of flexible, adaptive resource coordination techniques are being explored. Dynamic coscheduling (DCS) [85, 520] achieves coordinated thread scheduling for parallel jobs without requiring operating systems modifications. Memory and input/output coordination technologies are also being explored.
- ♦ *Predictable communication*: Based on a novel technology of self-synchronizing schedules, FM-QoS uses external network scheduling to achieve predictable performance: latency, bandwidth, and jitter at a 25 μ s scale [131]. Such an external approach is required to deal with network heterogeneity. Extension of this approach to multiple networks is being explored.
- ♦ *Front end and administration tools*: Basic tools are being explored for distributed cluster access, scheduling, job submission, and monitoring. HPVM's Java-based front end allows uniform remote access from any computing platform, and exploiting Platform Computing's LSF provides robust management and monitoring software.

The HPVM 1.0 release (August 1997) provides a substrate on which to build many of the technologies of a shared controllable-performance SCE. The HPVM 1.0 release provides high-performance communication, sharable resources, and efficient parallel execution on distributed resources. Thus, current HPVM clusters operate much like sharable, distributable massively parallel processors. These already enable high-performance distributed applications of a unique nature (for the latest information, see www-csag.cs.uiuc.edu/projects/hpvm.html). However, a number of important challenges remain:

- ♦ *Heterogeneity*: HPVM clusters currently manage heterogeneity in machine configuration, but not in network hardware. Network heterogeneity has

yet to be reconciled with high performance. A wealth of interesting questions remain about how to integrate, manage, and allocate heterogeneous network resources.

- ♦ *Performance models*: What are appropriate performance models for virtual machines to support large classes of applications? Beyond static computation, memory, and computation balance at one extreme and best effort in all three at another, what are the viable, usable resource abstractions that applications need?
- ♦ *Adaptive resource management*: Given appropriate models, technologies are needed to implement them robustly in a dynamic resource environment. Further, appropriate APIs for notification and adaptation must be developed that capture and present the information an application needs to respond intelligently. And such technologies must ensure dynamically stable behavior in the simple composite element.

While the Illinois HPVM project specifically targets shared controllable-performance SCEs, several other cluster projects also explore some of the same basic challenges, though typically with a different perspective. The two most relevant are addressed below.

RWCP

The Real World Computing Project in Japan has been building high-performance SCEs that are sharable to a limited degree. The PM layer [534] provides direct user access to the network, multiprocess usage, and high-performance communication. In addition, a modified Berkeley Software Design Incorporated (BSDI) kernel supports coarse-grained coordinated parallel scheduling on signals, called Time-Space-Shared Scheduling [279].

NOW II

Recent efforts at the Berkeley NOW project have also extended in the direction of sharable high-performance SCEs. The Active Messages II interface [123] provides multiuser access and connection virtualization, enabling multiprogrammed use of the system with good network performance. In addition, the Active Messages II implementation supports a dynamic connection model, allowing it to be used in similar fashion to Illinois Fast Messages 2.1 to build distributed applications. In addition, NOW researchers have been exploring job coscheduling in a shared-resource environment. Their approach, called implicit coscheduling, has demonstrated that local kernel schedulers can achieve

Element type	Capacity	Aggregatable	Reliable	Predictable	Sharable
Basic compute	Fixed	—	No	No	Yes
Basic storage	Fixed	—	No	No	Yes
Basic network	Fixed	—	No	No	Yes
High-throughput SCE	Scalable	No	Partial	No	Yes
Reliable SCE	Limited scaling	No	Yes	No	Yes
Dedicated high-performance SCE	Scalable	Yes	No	No	No
Shared controllable-performance SCE	Scalable	Yes	No	Yes	Yes

17.2 Each SCE type provides unique capabilities.

TABLE

coscheduling and share parallel resources efficiently by employing the traditional I/O protocol at communication operations—block and boost priority at resume and decay priority if success [174]. Further enhancements modify node schedulers to demonstrate fair and proportional share scheduling [175]. To date, the performance results are demonstrated for Split-C, a programming model for SPMD programs that contains frequent barrier synchronizations. Additional experiments are needed to validate the broader applicability of the results.

17.3 SUMMARY

In this chapter, we have focused on the capabilities of basic and SCEs. The capabilities of basic computation, communication, and storage elements continue to improve geometrically, producing a grid rich in resources and capable of rich sharing because of the availability of high-bandwidth links. Simple composite elements provide both aggregate capabilities and qualitatively different capabilities, such as reliability (summarized in Table 17.2). These capabilities will determine the role and contribution of SCEs to larger grids.

There are several important challenges in building bridges from useful SCEs to large-scale grids. Three major elements of these challenges are composition (interfacing), performance guarantees, and security and data integrity.

448

Composing both basic and SCEs into larger grids is a complex challenge, requiring directory services, protocols, compatible services and data representations, and conversion where necessary. While a general solution must involve all element types, special problems are raised here by simple composite elements. They include interfaces, semantics, scheduling, and management for aggregated memory resources, aggregated persistent storage resources, aggregated communication resources, and migration/interoperation. The distinct challenges for SCEs here include developing interfaces to aggregate resources that provide simple semantics and high performance, yet reflect the fact that even within an SCE, the hardware elements come and go dynamically. A number of lower-level issues are also critical: how to map aggregated resources to the disjoint basic elements within an SCE (e.g., where do the I/O requests and computations go?), whether such mappings are static or dynamic, and how users can manage their usage of the resources.

Another important challenge is enabling grid computations to achieve reasonable global performance. In most current approaches, this is predicated on some ability to reserve (or guarantee) performance from grid elements. However, such techniques must be reconciled with local resource management policies designed to achieve local resource and computational efficiency. Thus, challenges for SCEs in grids include the following:

- ♦ Mechanisms to ensure predictable performance for memory, computation, and storage both for basic and aggregated resources
- ♦ Techniques for coordinated scheduling across basic (within an SCE) and global memory, computation, and storage resources
- ♦ Policies for predictable performance, which manage the needs of global and local computations against available resources

Finally, if users are to achieve high productivity in a computing environment spanning numerous physical resources and administrative domains, they must be insulated from a wealth of security and data integrity concerns. For example, users may wish to use remote storage facilities for performance or even cost advantages. However, if they must worry about unauthorized data disclosure or unexpected loss of data, users are likely to be scarce. Thus, important challenges for SCEs in grids include providing secure access to resources and data security for participating computations, and providing data integrity guarantees for data, independent of availability or failure of any individual data repository.

ACKNOWLEDGMENTS

The HPVM and Fast Messages work described in this chapter reflects the efforts of a number of talented graduate students, including Scott Pakin, Mario Lauria, Vijay Karamcheti, Matt Buchanan, Kay Hane Connelly, Louis Giannini, and Geetanjali Sampemane, as well as research programmers Greg Koenig and Qian Liu.

The research described here was supported in part by DARPA order #E313 through the U.S. Air Force Rome Laboratory contract F30602-96-1-0286 and contract F30602-97-2-0121, NSF grant MIP-92-23732, and NASA grant NAG 1-613. Support from Intel Corporation, Tandem Computer, Hewlett-Packard, and Microsoft is also gratefully acknowledged. Andrew Chien is supported in part by NSF Young Investigator Award CCR-94-57809.

FURTHER READING

For more information on the topics covered in this chapter, see www.mkp.com/grids and also the following references:

- ♦ Pakin et al. [431] and Chun et al. [123] give descriptions of fast communication methods.
- ♦ Gray and Reuter [245] provide information relevant to reliable SCEs.
- ♦ Pfister [450] gives a discussion of workstation and PC clusters.
- ♦ Snir et al. [519] describe a dedicated high-performance SCE, the IBM SP.
- ♦ Chien et al. [121] provide material on a sharable controllable-performance SCE project, HPVM.