Chapter 1
System Models and Enabling Technologies

Summary: Parallel, distributed, and cloud computing systems advance all works of life. This chapter assesses the evolutional changes in computing and IT trends in the past 30 years. These changes are driven by killer applications with variable amounts of workload and datasets at different periods of time. We study *high-performance computing* (HPC) and *high-throughput computing* (HTC) systems in clusters/MPP, *service-oriented architecture* (SOA), grids, P2P networks, and Internet clouds. These systems are distinguished by their architectures, OS platforms, processing algorithms, communication protocols, security demands, and service models. This chapter introduces the essential issues in scalability, performance, availability, security, energy-efficiency, workload outsourcing, datacenter protection, etc. The intent is to pave the way for our readers to study the details in subsequent chapters.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

1.1 Scalable Computing Towards Massive Parallelism

Over the past 60 years, the state of computing has gone through a series of platform and environmental changes. We review below the evolutional changes in machine architecture, operating system platform, network connectivity, and application workloads. Instead of using a centralized computer to solve computational problems, a parallel and distributed computing system uses multiple computers to solve large-scale problems over the Internet. Distributed computing becomes data-intensive and network-centric. We will identify the killer applications of modern systems that practice parallel and distributed computing. These large-scale applications have significantly upgraded the quality of life in all aspects of our civilization.

1.1.1 High-Performance versus High-Throughput Computing

For a long time, *high-performance computing* (HPC) systems emphasizes the raw speed performance. The speed of HPC systems increased from Gflops in the early 1990's to now Pflops in 2010. This improvement was driven mainly by demands from scientific, engineering, and manufacturing communities in the past. The speed performance in term of floating-point computing capability on a single system is facing some challenges by the business computing users. This flops speed measures the time to complete the execution of a single large computing task, like the Linpack benchmark used in Top-500 ranking. In reality, the number of users of the Top-500 HPC computers is rather limited to only 10% of all computer users. Today, majority of computer users are still using desktop computers and servers either locally or in huge datacenters, when they conduct Internet search and market-driven computing tasks.

The development of market-oriented high-end computing systems is facing a strategic change from the HPC paradigm to a *high-throughput computing* (HTC) paradigm. This HTC paradigm pays more attention to high-flux multi-computing. The main application of high-flux computing system lies in Internet searches and web services by millions or more users simultaneously. The performance goal is thus shifted to measure the *high throughput* or the number of tasks completed per unit of time. HTC technology needs to improve not only high speed in batch processing, but also address the acute problem of cost, energy saving, security, and reliability at many datacenters and enterprise computing centers. This book is designed to address both HPC and HTC systems, that meet the demands of all computer users.

In the past, electronic computers have gone through five generations of development. Each generation lasted 10 to 20 years. Adjacent generations overlapped in about 10 years. During 1950-1970, a handful of mainframe, such as IBM 360 and CDC 6400, were built to satisfy the demand from large business or government organizations. During 1960–1980, lower-cost minicomputers, like DEC's PDP 11 and VAX series, became popular in small business and college campuses. During 1970-1990, personal computers built with VLSI microprocessors became widespread in use by mass population. During 1980-2000, massive number of portable computers and pervasive devices appeared in both wired and wireless applications. Since 1990, we are overwhelmed with using both HPC and HTC systems that are hidden in Internet clouds. They offer web-scale services to general masses in a digital society.

Levels of Parallelism: Let us first review types of parallelism before we proceed further with the computing trends. When hardware was bulky and expensive 50 years ago, most computers were designed in a bit-serial fashion. *Bit-level parallelism* (BLP) converts bit-serial processing to word-level processing gradually. We started with 4-bit microprocessors to 8, 16, 32 and 64-bit CPUs over the years. The next wave of improvement is the *instruction-level parallelism* (ILP). When we shifted from using processor to execute single instruction at a time to execute multiple instructions simultaneously, we have practiced ILP through pipelining, superscalar, VLIW (*very-long instruction word*), and multithreading in the past 30 years. ILP demands branch prediction, dynamic scheduling, speculation, and higher degree of compiler

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

support to make it work efficiently.

Data-level parallelism (DLP) was made popular through SIMD (*single-instruction and multiple-data*) and vector machines using vector or array types of instructions. DLP demands both even more hardware support and compiler assistance to work properly. Ever since the introduction of multicore processors and *chip multiprocessors* (CMP), we explore the *task-level parallelism* (TLP). A modern processor explores all of the above parallelism types. The BLP, ILP, and DLP are well supported by advances in hardware and compilers. However, the TLP is far from being very successful due to the difficulty in programming and compilation of codes for efficient execution on multicores and CMPs. As we move from parallel processing to distributed processing, we will see the increase of computing granularity to *job-level parallelism* (JLP). It is fair to say the coarse-grain parallelism is built on top of the fine-grain parallelism.

The Age of Internet Computing : The rapid development of the Internet has resulted in billions of people login online everyday. As a result, supercomputer sites and datacenters have changed from providing high performance floating-point computing capabilities to concurrently servicing huge number of requests from billions of users. The development of computing clouds computing and the widely adoption of provided computing services demand HTC systems which are often built parallel and distributed computing technologies. We cannot meet the future computing demand by pursuing only the Linpack performance on a handful of computers. We must build efficient datacenters using low-cost servers, storage systems, and high-bandwidth networks.

In the future, both HPC and HTC demand multi-core processors that can handle hundreds or thousand of computing threads, tens-of-kilo-thread node prototype, and mobile cloud services platform prototype. Both types of systems emphasize parallelism and distributed computing. Future HPC and HTC systems must satisfy the huge demand of computing power in terms of throughput, efficiency, scalability, reliability etc. The term of high efficiency used here means not only speed performance of computing systems, but also the work efficiency (including the programming efficiency) and the energy efficiency in term of throughput per watt of energy consumed. To achieve these goals, three key scientific issues must be addressed:

(1) *Efficiency* measured in building blocks and execution model to exploit massive parallelism as in HPC. This may include data access and storage model for HTC and energy efficiency.

(2) *Dependability* in terms of reliability and self-management from the chip to system and application levels. The purpose is to provide high-throughput service with QoS assurance even under failure conditions.

(3) *Adaptation* in programming model which can support billions of job requests over massive datasets, virtualized cloud resources, and flexible application service model.

The Platform Evolution: The general computing trend is to leverage more and more on shared web resources over the Internet. As illustrated in Fig.1.1, we see the evolution from two tracks of system development: *distributed computing systems* (DCS) and *high-performance computing* (HPC) systems. On the HPC side, homogeneous supercomputers (*massively parallel processors*, MPP) are gradually replaced by clusters of cooperative computers out of the desire to share computing resources. The cluster is often a collection of computer nodes that are physically connected in close range to each other. Clusters, MPP, and Grid systems are studied in Chapters 3 and 4. On the DCS side, *Peer-to-Peer* (P2P) networks appeared for distributed file sharing and content delivery applications. A P2P system is built over many client machines to be studied in Chapter 5. Peer machines are globally distributed in nature. Both P2P and cloud computing

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

and web service platforms emphasize more on HTC rather than HPC.

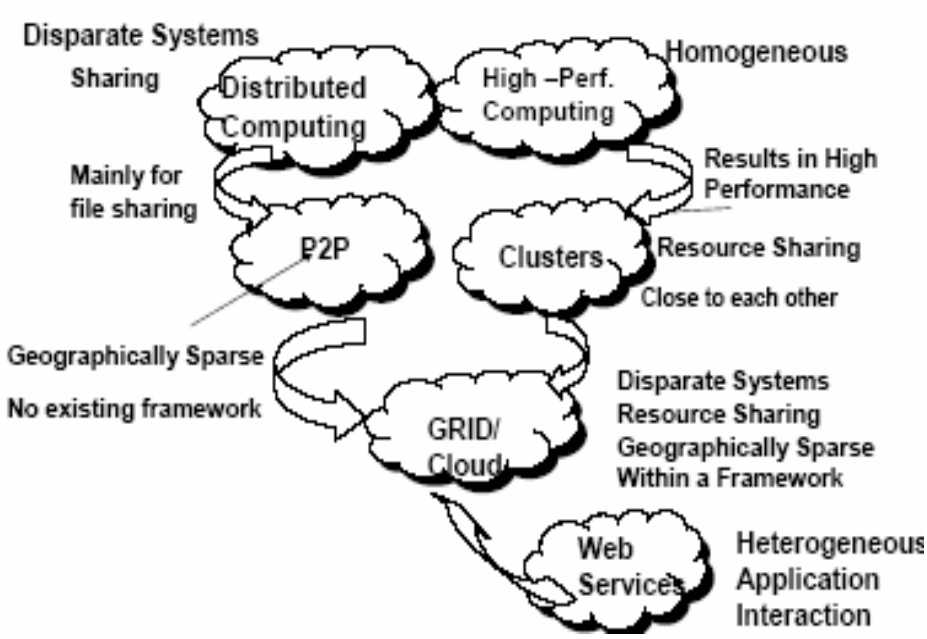

Figure 1.1 Evolutional trend towards web-scale distributed high-throughput computing and integrated web services to satisfy heterogeneous applications.

Distributed Computing Families: Ever since mid 90's, technologies for building *peer-to-peer* (P2P) networks and *network of clusters* were consolidated into many national projects to establish wide-area computing infrastructures, known as *computational grids* or *data grids.* We will study Grid computing technology in Chapter 4. More recently, there is a surge of interest to explore Internet cloud resources for web-scale supercomputing. Internet clouds are resulted from moving desktop computing to a service-oriented computing using server clusters and huge databases at datacenters. This chapter introduces the basics of various parallel and distributed families. Grids and clouds are disparity systems with great emphases on resource sharing in hardware, software, and datasets.

Design theory, enabling technologies, and case studies of these massively distributed systems are treated in this book. Massively distributed systems are intended to exploit a high degree of parallelism or concurrency among many machines. In 2009, the largest cluster ever built has 224,162 processor cores in Cray XT-5 system. The largest computational grid connects any where from ten to hundreds of server clusters. A typical P2P network may involve millions of client machines, simultaneously. Experimental cloud computing clusters have been built with thousands of processing nodes. We devote the material min Chapters 7 and 8 to cover cloud computing Case studies of HPC system as cluster and grids and HTC systems as P2P networks and datacenter-based cloud platforms will be examined in Chapter 9.

1.1.2 Analysis of Top-500 Supercomputers

Figure 1.2 plots the measured performance of the Top-500 fastest computers from 1993 to 2009. The Y-axis is scaled by the sustained speed performance in terms of GFlops, Tfops, and PFlops. The middle curve plots the performance of the No.1 fastest computers recorded over the years. The peak performance increases from 58.7 GFlops to 1.76 PFlops in 16 years. The bottom curve corresponds to the number 500

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

computer speed at each year. It increases from 0.42 GFlops to 20 Tflops in 16 years. The top curve plots the total sum of all 500 fastest computer speed ovet the same period. These plots give a fairly good performance projection for years to come. For example, 1 PFlops was achieved by IBM Roadrunner in June of 2007. It is interesting to observe that the total sum increases almost linearly over the years.

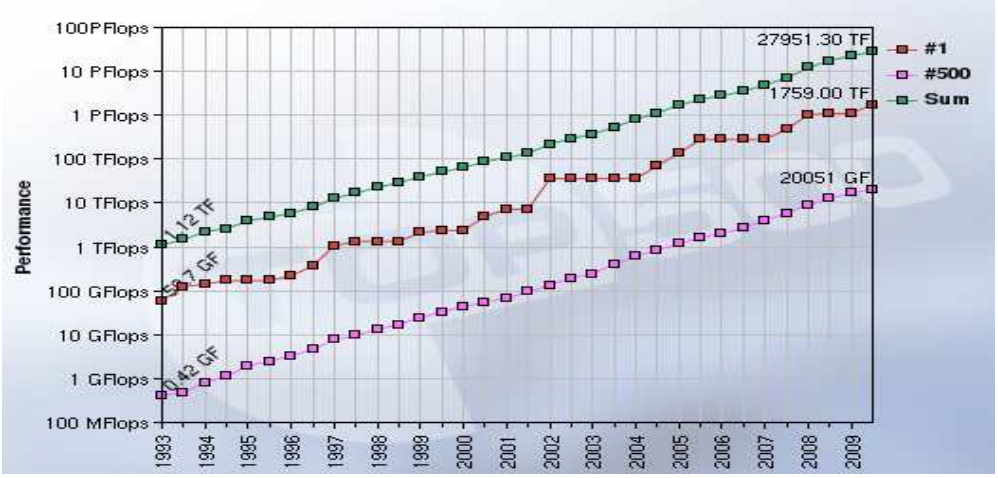

Figure 1.2 The Top-500 supercomputer performance from 1993 to 2009
(Courtesy of Top 500 Organization, 2009)

It is interesting to observe in Fig.1.3 the architectural evolution of the Top-500 supercomputers over the years. In 1993, 250 systems assumed the SMP (*symmetric multiprossor*) architecture shown in yellow area. Most SMPs are built with shared memory and shared I/O devices. The word "symmetric" refers to the fact all processors are equally capable to execute the supervisory and/or the application codes. There were 120 MPP systems (in dark orange area) built then. The SIMD (*single instruction stream over multiple data streams)* machines (some called array processors) and uniprocessor systems disappeared in 1997, while the cluster (light orange) architecture appeared in 1999, The clustered systems grow rapidly from a few to 375 systems out of 500 by 2005. On the other hand, the SMP architecture disappeared gradually to zero by 2002. Today, the dominating architecture classes in the Top-500 list are the clusters, MPP, and constellations (pink). More than 85% of the Top-500 computers used in 2010 adopted the cluster configurations and the remaining 15% chosen the MPP (*massively parallel processor*) architecture.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

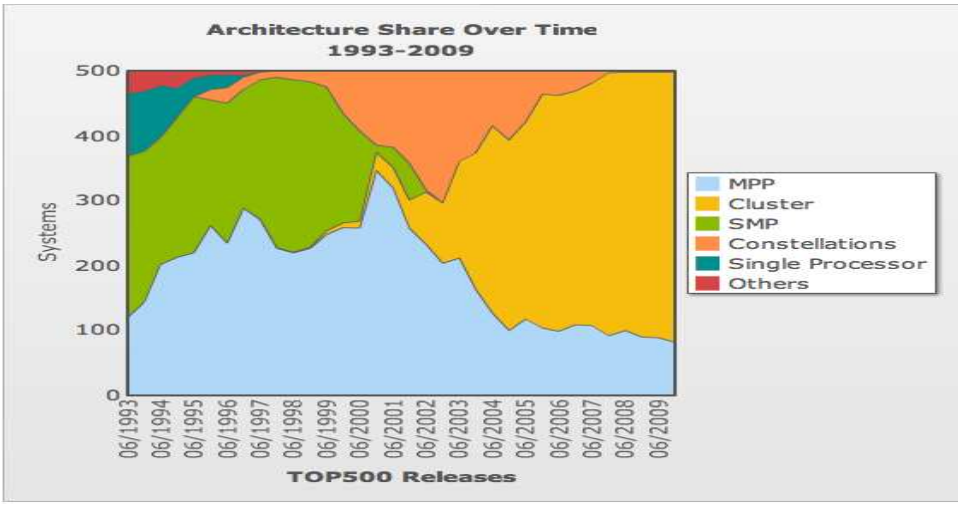

Figure 1.3 Architectural evolution of the Top-500 supercomputers from 1993 to 2009.
(Courtesy of Top 500 Organization, 2009)

In Table 1.1, we summarize the key architecture features, sustained Linpack benchmark performance , and power assumption of five top 5 supercomputers reported in November 2009. We will present the details of the top two systems: Cray Jaguar and IBM Roadrunner as case studies in Chapter 8. These two machines have exceeded the Pflops performance. The power consumptions of these systems are enormous including the cooling electricity. This has triggered the increasing demand of green information technology in recent years. These state of the art systems will be used far beyond 2010 when this book was written.

Table 1.1 Top Five Supercomputers Evaluated in Nov. 2009

System Rank and Name	Architecture Description (Core size, Processor, GHz, OS, and Topology)	Sustained Speed	Power/ system
1. Jaguar at Oak Ridge Nat'l Lab, US	Cray XT-5HE: An MPP built with 224,162 cores in 2.6 GHz Opteron 6-core processors, interconnected by a 3-D torus network	1.759 PFlops	6.95 MW
2. Roadrunner at DOE/NNSA/ LANL, US	IBM BladeCenter QS22/LS21 cluster of 122,400 cores in 12,960 3.2 GHz POWER XCell 8i processors and 6,480 AMD 1.8 GHz Operon dual-core processors, running Linux and interconnected by an InfiniBand network	1.042 PFops	2.35 MW
3. Kraken at NICS, University of Tennessee, US	Crat XT-5-HE : An MPP built with 98,928 cores of 2.6 GHz Opteron 6-core processors interconnected by a 3-D torus network	831 TFops	3.09 MW
4. JUGENE at the FZJ in Germany	IBM BlueGene/P solution built with 294,912 processors: PowerPC core, 4-way SMP nodes, and 144 TB of memory in 72 racks, interconnected by a 3-D torus network	825.5 TFlops	2.27 MW
5. Tianhe-1 at NSC/ NUDT in China	NUST TH-1 cluster of 71,680 cores in Xeon processors and ATI Radeon GPUs, interconnected by an InfiniBand network	563 TFlops	1.48 MW

1.1.3 Killer Applications and Grand Challenges

High-performance computing systems offer transparency in many application aspects. For example,

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

data access, resource allocation, process location, concurrency in execution, job replication, and failure recovery should be made transparent to both users and system management. In Table 1.2, we identify below a few key applications that have driven the development of parallel and distributed systems in recent years. These applications spread across many important domains in our society: science, engineering, business, education, health care, traffic control, Internet and web services, military, and government applications. Almost all applications demand computing economics, web-scale data collection, system reliability, and scalable performance.

For example, distributed transaction processing is often practiced in banking and finance industry. Distributed banking systems must be designed to scale and tolerate faults with the growing demands. Transactions represent 90% of the existing market for reliable banking systems. We have to deal with multiple database servers in distributed transactions. How to maintain the consistency of replicated transaction records is crucial in real-time banking services. Other complications include short of software support, network saturation, and security threats in these applications. We will study some of the killer applications and the software standards needed in Chapters 8 and 9.

Table 1.2 Killer Applications of HPC and HTC Systems

Domain	Specific Applications
Science and Engineering	Scientific simulations, genomic analysis, etc.
	Earthquake prediction, global warming, weather forecasting, etc.
Business, Education, service industry, and Health Care	Telecommunication, content delivery, e-commerce, etc.
	Banking, stock exchanges, transaction processing, etc.
	Air traffic control , electric power Grids, distance education, etc.
	Health care, hospital automation, telemedicine, etc.
Internet and Web Services and Government	Internet search, datacenters, decision-make systems, etc.
	Traffic monitory , worm containment, cyber security, etc.
	Digital government, on-line tax return, social networking, etc.
Mission-Critical Applications	Military commend, control, intelligent systems, crisis management, etc.

1.2 Enabling Technologies for Distributed Parallelism

This section reviews hardware, software and network technologies for distributed computing system design and applications. Viable approaches to build distributed operating systems are assessed for handling massive parallelism in distributed environment.

1.2.1 System Components and Wide-Area Networking

In this section, we assess the growth of component and network technologies in building HPC or HTC systems in recent years. In Fig.1,4, processor speed is measured by MIPS (*million instructions per second*).

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

The network bandwidth is counted by Mbps or Gbps (*Mega or Giga bits per second*). The unit GE refers to 1 Gbps Ethernet bandwidth.

Advances in Processors: The upper curve in Fig.1.4 plots the processor speed growth in modern micro processors or in *chip multiprocessors* (CMP). We see a growth from 1 MIPS of VAX 780 in 1978 to 1,800 MIPS of Intel Pentium 4 in 2002, and to 22,000 MIPS peak for Sun Niagara 2 in 2008. By Moore's law, the processor speed is doubled in every 18 months. This doubling effect was pretty accurate in the past 30 years. The clock rate for these processors increases from 12 MHz in Intel 286 to 4 GHz in Pentium 4 in 30 years. However, the clock rate has stopped increasingly due to the need to reduce power consumption. The ILP (*instruction-level parallelism*) is highly exploited in modern processors. ILP mechanisms include multiple-issue superscalar architecture, dynamic branch prediction, and speculative execution, etc. These ILP techniques are all hardware and compiler-supported. In addition, DLP (*data-level parallelism*) and TLP (*thread-level parallelism*) are also highly explored in today's processors.

Many processors are now upgraded to have multi-core and multithreaded micro-architectures. The architecture of a typical multicore processor is shown in Fig.1.5. Each core is essentially a processor with its own private cache (L1 cache). Multiple cores are housed in the same chip with a L2 cache that is shared by all cores. In the future, multiple CMPs could be built on the same CPU chip with even the L3 cache on chip. Multicore and multithreaded processors are now built in many high-end processors like Intel Xeon, Montecito, Sun Niagara, IBM Power 6 and X cell processors. Each core could be also multithreaded. For example, the Niagara II is built with 8 cores with 8 threads handled by each core. This implies that the maximum ILP and TLP that can be exploited in Niagaris equal to 64 (= 8 x 8).

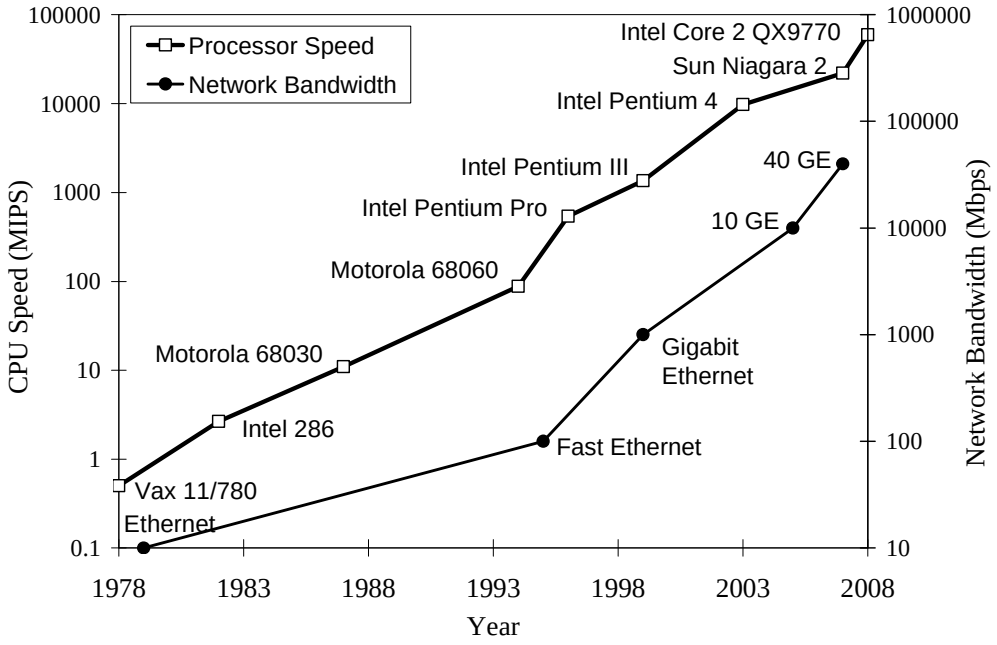

Figure 1.4 Improvement of processor and network technologies over 30 years.

Multicore Architecture: With multiple of the multicores in Fig.1.5 buily on even larger chip, the number of working cores on the same CPU chip could reach hundreds in the next few years. Both IA-32 and IA-64 instruction set architectures are built in commercial processors today. Now, x-86 processors has been extended to serve HPC and HTC in some high-end server processors. Many RISC processors are now replaced by multicore x-86 processors in the Top-500 supercomputer systems. The trend is that x-86

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

upgrades will dominate in datacenters and supercomputers. *Graphic processing units* (GPU) appeared in HPC systems. In the future, Exa-scale (EFlops or 10^{18} Flops) systems could be built with a large number of multi-core CPUs and GPUs. In 2009, the No.1 supercomputer in the Top-500 list (a Cray XT-5 named Jaguar) has already with almost over 30 thousands AMD 6-core Opteron processors resulting a total of 224,162 cores in the entire HPC system.

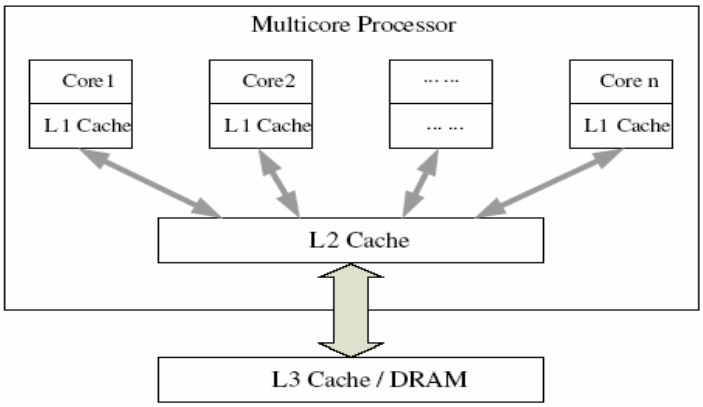

Figure 1.5 The schematic of a modern multicore processor using a hierarchy of caches

Wide-Area Networking : The lower curve in Fig.1.4 plots the rapid growth of Ethernet bandwidth from 10 Mbps in 1979 to 1 Gbps in 1999 and 40 GE in 2007. It was speculated that 1 Tbps network links will be available by 2012. According to Berman, Fox, and Hey [3], we expect a 1,000, 1,000, 100, 10, and 1 Gbps network links, respectively, at international, national, organization, optical desktop, and copper desktop connections in 2006. An increase factor of 2 per year on network performance was reported, which is faster than Moore's law on CPU speed doubling in every 18 months. The implication is that more computers will be used concurrently in the future. High-bandwidth networking increases the capability of building massively distributed systems. The IDC 2010 report has predicted that both InfiniBand and Ethernet will be the two major interconnect choices in the HPC arena.

Memory, SSD, and Disk Arrays: Figure 1.12 plots the growth of DRAM chip capacity from 16 Kb in 1976 to 16 Gb in 2008. This shows that the memory chips get 4 times increase in capacity every 3 years. The memory access time did not improve much in the past. In fact, the memory wall problem is getting worse as the processor gets faster. For hard drives, the capacity increases from 260 MB in 250 GB in 2004. The Seagate Barracuda 7200.11 hard drive reached 1.5 TB in 2008. The increase is about 10 times in capacity every 8 years. The capacity increase of disk arrays is even greater in the years to come. On the other hand, faster processor speed and larger memory capacity result in wider gap between processors and memory. The memory wall becomes even a more serious problem than before. Memory wall still limits the scalability of multi-core processors in terms of performance.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

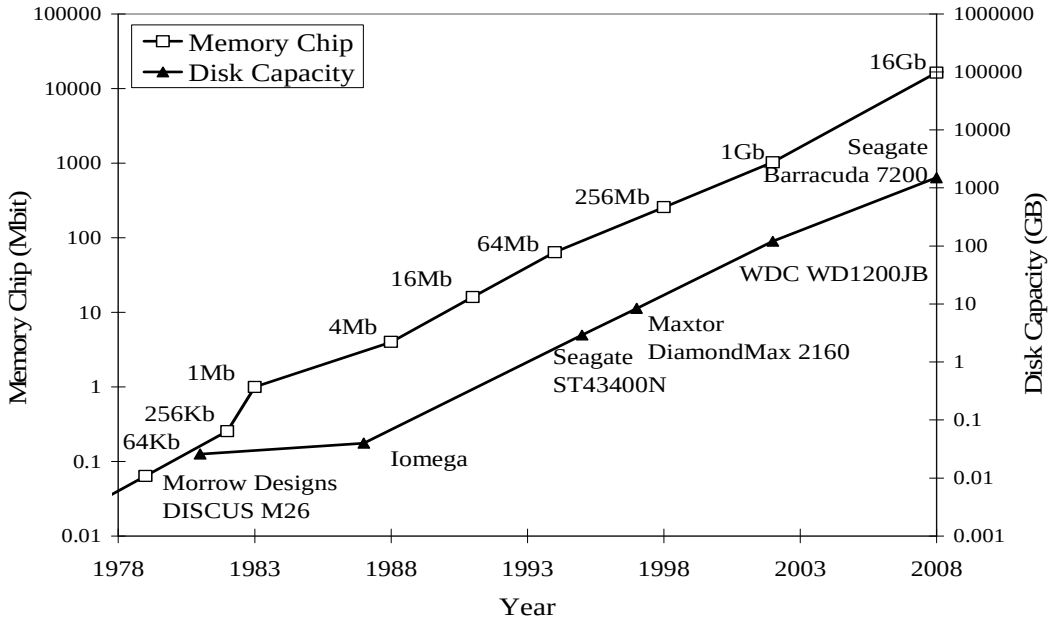

Figure 1.6 Improvement of memory and disk technologies over 30 years

The rapid growth of flash memory and *solid-state drive* (SSD) also impacts the future of HPC and HTC systems. The mortality rate of SSD is not bad at all. A typical SSD can handle 300,000 -1,000,000 write cycles per block. So SSD can last for several years, even they have heavy write usage. Flash and SSD will demonstrate impressive speedups in many applications. For example, the Apple Macbook pro uses 128 GB solid-state hard drive, which is only $150 more than a 500 GB 7200 RPM SATA drive. However to get 256 GB or 512 GB SSD drive, the cost may go up significantly. At present, SSD drives are still too expensive to replace stable disk arrays in the storage market. Eventually, power consumption, cooling and packaging will limit the large system development. The power increases linearly with respect to the clock frequency and quadratically with respect to the voltage applied on chips. We cannot increase the clock rate indefinitely. Lower the voltage supply is very much in demand.

1.2.2 Virtual Machines and Virtualization Middleware

A conventional computer has a single OS image. This offers a rigid architecture that tightly couples application software to a specific hardware platform. Some software running well on one machine may not be executable on anther platform with a different instruction set under a fixed OS management. *Virtual machines* (VM) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.

Virtual Machines: The concept of virtual machines is illustrated in Fig.1.7. The host machine is equipped with the physical hardware shown at the bottom. For example, a desktop with x-86 architecture running its installed Windows OS as shown in Fig.1.7(a). The VM can be provisioned to any hardware system. The VM is built with virtual resources managed by a guest OS to run a specific application. Between the VMs and the host platform, we need to deploy a middleware layer called a *virtual machine monitor* (VMM) . Figure 1.7(b) shows a native VM installed with the use a VMM called a *hypervisor* at the privileged mode. For example, the hardware has a x-86 architecture running the Windows system. The guest OS could be a Linux system and the hypervisor is the XEN system developed at Cambridge University. This hypervisor approach is also called bare-metal VM, because the hypervisor handles the bare hardware (CPU, memory, and I/O) directly.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

Another architecture is the host VM shown in Fig.1.7(c). Here the VMM runs with a non-privileged mode. The host OS need not be modified. The VM can be also implemented with a dual mode as shown in Fig.1.7(d). Part of VMM runs at the user level and another portion runs at the supervisor level. In this case, the host OS may have to be modified to some extent. Multiple VMs can be ported to one given hardware system, to support the virtualization process. The VM approach offers hardware-independence of the OS and applications. The user application and its dedicated OS could be bundled together as a virtual appliance, that can be easily ported on various hardware platforms.

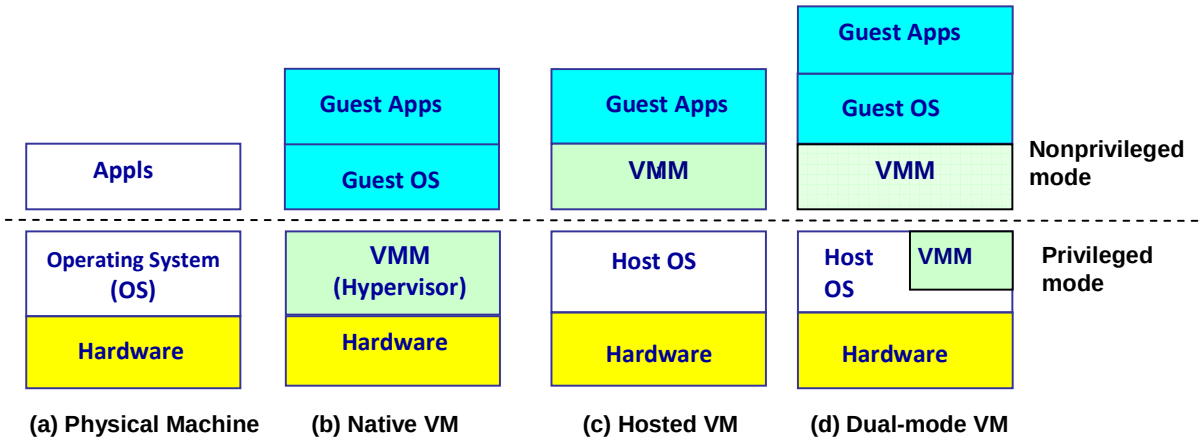

Figure 1.7 Three ways of constructing a virtual machine (VM) embedded in a physical machine.The VM could run on an OS different from that of the host computer.

Virtualization Operations: The VMM provides the VM abstraction to the guest OS. With full virtualization, the VMM exports a VM abstraction identical to the physical machine; so that a standard OS such as Windows 2000 or Linux can run just as they would on the physical hardware. Low-level VMM operations are indicated by Mendel Rosenblum [29] and illustrated in Fig.1..8. First, the VMs can be multiplexed between hardware machines as shown in Fig.1..8(a). Second, a VM can be suspended and stored in a stable storage as shown in Fig.1..8 (b). Third, a suspended VM can be resumed or provisioned to a new hardware platform in Fig.1.8(c). Finally, a VM can be migrated from one hardware platform to another platform as shown in Fig.1.8 (d).

These VM operations enable a virtual machine to be provisioned to any available hardware platform. They make it flexible to port distributed application executions. Furthermore the VM approach will significantly enhance the utilization of server resources. Multiple server functions can be consolidated on the same hardware platform to achieve higher system efficiency. This will eliminate server sprawl via deployment of systems as VMs. These VMs move transparency to the shared hardware. According to a claim by VMWare, the server utilization could be increased from current 5-15% to 60-80%.

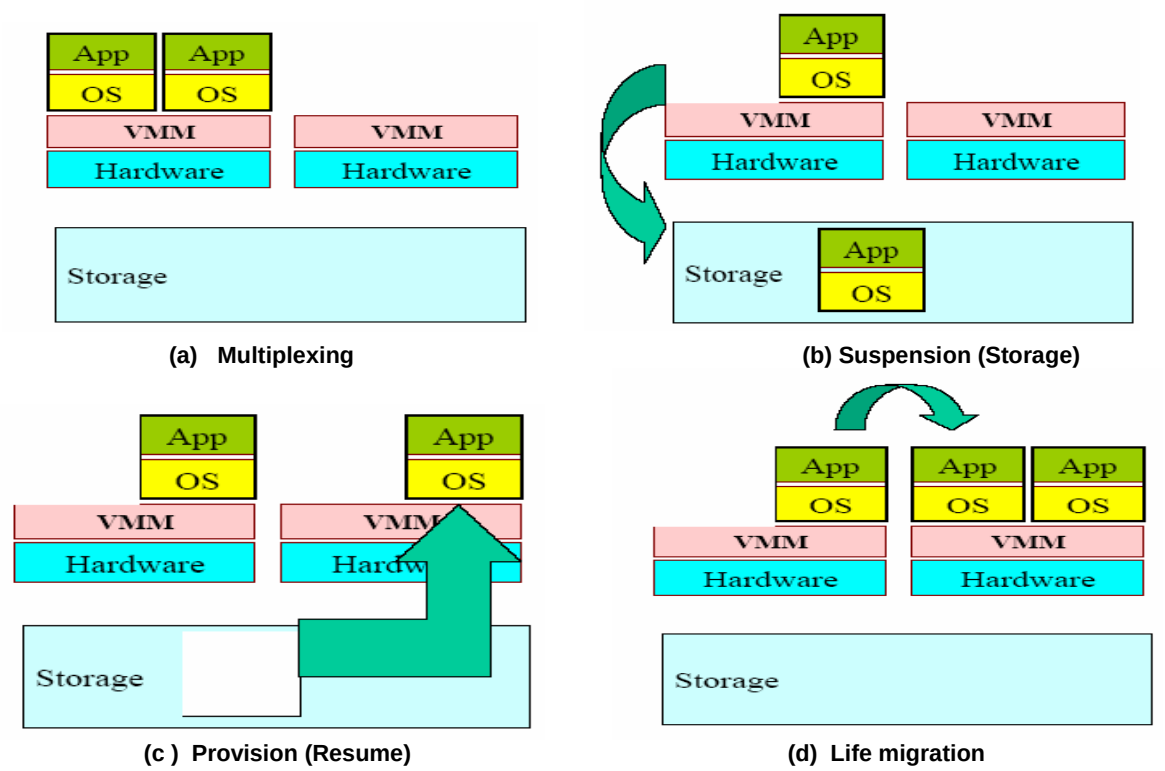

Figure 1.8 Virtual machine multiplexing, suspension, provision, and migration in a distributed computing environment, (Courtesy of M. Rosenblum, Keynote address, *ACM ASPLOS* 2006 [29])

Virtual Infrastructures: This is very much needed in distributed computing. Physical resources for compute, storage, and networking at the bottom are mapped to the needy applications embedded in various VMs at the top. Hardware and software are then separated. Virtual Infrastructure is what connects resources to distributed applications. It is a dynamic mapping of the system resources to specific applications. The result is decreased costs and increased efficiencies and responsiveness. Virtualization for server consolidation and containment is a good example. We will study virtual machines and virtualization support in Chapter 2. Virtualization support for clusters, grids and clouds are studied in Chapters 3, 4, and 6, respectively.

1.2.3 Trends in Distributed Operating Systems

The computers in most distributed systems are loosely coupled. Thus the distributed system has inherently multiple system images. This is mainly due to the fact that all node machines run with an independent operating system. To promote resource sharing and fast communications among node machines, we desire to have a *distributed* OS that manages all resources coherently and efficiently. Such a system is most likely to be a closed system. They rely on message passing and *remote procedure call* (RPC) for internode communications. It should be pointed out that a distributed OS is crucial to upgrade the performance, efficiency, and application flexibility of distributed applications. A distributed system could not face the shortcomings in restricted applications and lack of software and security support, until a well-built distributed OSs are in widespread use.

Distributed Operating Systems : Tanenbaum [26] classifies three approaches to distributing the resource management functions in a distributed computer system. The first approach is to build a *network OS* over a

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

large number of heterogeneous OS platforms. Such a network OS offers the lowest transparency to users. Network OS is essentially a distributed file system. Independent computers rely on file sharing as a means of communication. The second approach is to develop middleware to offer limited degree of resource sharing like what was build for clustered systems (Section 1.2.1). The third approach is to develop a *distributed OS* to achieve higher use or system transparency.

Amoeba vs. DCE: Table 1.3 summarizes the functionalities of a distributed OS Amoeba and a middleware-based DCE developed in the last two decades. To balance the resource management workload, the functionalities of such a DOS should be distributed to any available server. In this sense, the conventional OS runs only on a centralized platform. With the distribution of OS services, the DOS design should take either a light-weight microkernel approach like the Amoeba [27] or extend an existing OS like the DCE [5] by extending UNIX. The trend is to free up users from most resource management duties. We need new web-based operating systems to support virtualization of resources in distributed environments. We shall study distributed OS installed in distributed systems in subsequent chapters.

Table 1.3 Feature Comparison of Two Distributed Operating Systems

Operating System Functionality	AMOEBA developed at Vrije University, Amsterdam [32]	DCE as OSF/1 by Open Software Foundation [5]
History and Current System Status	Developed at VU and tested in European Community, version 5.2 released in 1995, written in C.	Release as OSF/1 product, DEC was built as user extension on top of an existing OS like UNIX, VMS, Windows, OS/2, etc.
Distributed OS Architecture	Microkernel-based, location transparent, using many servers to handle files, directory, replication, run, boot, and TCP/IP services	This is a middleware-OS providing a platform for running distributed applications The system supports RPC, security, and other DCE Threads.
Amoeba Microkernel or DEC Packages	A special microkernel handles low-level process , memory, I/O, and communication functions	DCE packages handle file, time, directory, and security services, RPC, authentication at user space.
Communication Mechanisms	Use a network-layer FLIP protocol and RPC to implement point-to-point and group communications	DCE RPC supports authenticated communication and other security services in user programs

1.2.4 Parallel and Distributed Programming Environments

Four programming models are specifically introduced below for distributed computing with expected scalable performance and application flexibility. We summarize four distributed programming models in Table 1.4. Some software toolsets developed in recent years are also identified here. MPI is the most popular programming model for message-passing systems. Google's MapReduce and BigTable are for effective use of resources from Internet clouds and data centers. The service clouds demand extending Hadoop, EC2, and S3 to facilitate distributed computing applications over distributed storage systems.

Message-Passing Interface (MPI) is the primary programming standard used to develop parallel programs to run on a distributed system. MPI is essentially a library of subprograms that can be called from C or Fortran to write parallel programs running on a distributed system. We need to embody clusters, Grid and P2P systems with upgraded web services and utility computing applications. Besides MPI, distributed programming can be also supported with low-level primitives like PVM (*parallel virtual machine*). Both MPI and PVM are described in Hwang and Xu [20].

MapReduce: This is a web-programming model for scalable data processing on large clusters over large

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

datasets [11]. The model is applied mainly in web-scale search and cloud computing applications. The user specifies a *map function* to generate a set of intermediate key/value pairs. Then the user applies a *reduce function* to merge all intermediate values with the same intermediate key. MapReduce is highly scalable to explore high degree of parallelism at job levels. A typical MapReduce computation process many handle Terabybe of data on tens of thousand or more client machines. Hundreds of MapReduce programs are likely to be executed, simultaneously. Thousands of MapReduce jobs are executed on Google's clusters everyday.

Table 1.4 Parallel and Distributed Programming Models and Toolsets

Model	Objectives and Web Link	Attractive Features Implemented
MPI	Message-Passing Interface is a library of subprograms that can be called from C or Fortran to write parallel programs running on distributed computer systems [2, 21]	Specify synchronous or asynchronous point-to-point and collective communication commands and I/O operations in user programs for message-passing execution
MapReduce	A web programming model for scalable data processing on large cluster over large datasets, applied in web search operations [12]	A map function to generate a set of intermediate key/value pairs. A Reduce function to merge all intermediate values with the same key
Hadoop	A software platform to write and run large user applications on vas datasets in business and advertising applications. http://hadoop.apache.org/core/	Hadoop is scalable, economical, efficient and reliable in providing users with easy access of commercial clusters

Hadoop Library: Hadoop offers a software platform that was originally developed by a Yahoo group. The package enable users write and run applications over vast distributed data. Attractive features include: (1) Scalability: Hadoop can easily scale to store and process petabytes of data in the Web space. (2) Economy: An open-source MapReduce minimizes the overheads in task spawning and massive data communication, (3) Efficiency: Processing data with high-degree of parallelism across a large number of commodity nodes and (4) Reliability: This refers to automatically keeping multiple data copies to facilitate redeployment of computing tasks upon unexpected system failures.

Open Grid Service Architecture (OGSA): The development of grid infrastructure is driven by pushing need in large-scale distributed computing applications, These applications must count on a high degree of resource and data sharing. Table 1..5 introduces the OGSA (*Open Grid Service Architecture*) as a common standard for general public use of grid services. Genesis II is a its realization. The key features covers distributed execution environment, PKI (*Public Key Infrastructure*) services using local *certificate authority* (CA), trust management and security policies in grid computing.

Globus Toolkits and Extensions: *Globus* is middleware library jointly developed by the US Argonne National Laboratory and USC Information Science Institute during the past decade. This library implemented some of the OGSA standards for resource discovery, allocation, and security enforcement in a Grid environment. The Globus packages support multi-site mutual authentication with PKI certificates. Globus has gone through several versions released subsequently. The current version GT 4 is in use in 2008. Sun SGE and IBM Grid Toolbox: Both Sun Microsystems and IBM have extended Globus for business applications. We will cover grid computing principles and technology in Chapter 5 and grid applications in Chapter 9.

Table 1.5 Grid Standards and Toolkits for scientific and Engineering Applications

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

Grid Standards	Major Grid Service Functionalities	Key Features and Security Infrastructure
OGSA Standard	Open Grid Service Architecture offers common grid service standards for general public use	Support heterogeneous distributed environment, bridging CA, multiple trusted intermediaries, dynamic policies, multiple security mechanisms, etc.
Globus Toolkits	Resource allocation, Globus security infrastructure (GSI), and generic security service API	Sign-in multi-site authentication with PKI, Kerberos, SSL, Proxy, delegation, and GSS API for message integrity and confidentiality
Sun Grid Engine (SGE)	Supporting local grids and clusters in enterprise or campus Intranet grid applications	Using reserved ports, Kerberos, DCE, SSL, and authentication in classified hosts at various trust levels and resource access restrictions
IBM Grid Toolbox	AIX and Linux grids built on top of Globus Toolkit, autonomic computing, Replica services	Using simple CA, granting access, grid service (ReGS), supporting Grid application framework for Java (GAF4J), GridMap in IBM IntraGrid for security update, etc.

1.3 Distributed Computing System Models

A *massively parallel and distributed computing system* or in short a *massive system* is built over a large number of autonomous computer nodes. These node machines are interconnected by *system-area networks* (SAN), *local-are networks* (LAN), or *wide-area networks* (WAN) in a hierarchical manner. By today's networking technology, a few LAN switches can easily connect hundreds of machines as a working cluster. A WAN can connect many local clusters to form a very-large cluster of clusters. In this sense, one can build a massive system to have millions of computers connected to edge networks in various Internet domains.

System Classification: Massive systems are considered highly scalable to reach a web-scale connectivity, either physically or logically. In Table 1.6, we classify massive systems into four classes: namely the *clusters, P2P networks, computing grids,* and *Internet clouds* over huge datacenters. In terms of node number, these four system classes may involve hundreds, thousands, or even millions of computers as participating nodes. These machines work collectively, cooperatively, or collaboratively at various levels. The table entries characterize these four system classes in various technical and application aspects.

From the application prospective, clusters are most popular in supercomputing applications. In 2009, 417 out of the top-500 supercomputers were built with a cluster architecture. It is fair to say that clusters have laid the necessary foundation to build large-scale grids and clouds. P2P networks appeal most to business applications. However, the content industry was reluctant to accept P2P technology for lack of copyright protection in ad hoc networks. Many national grids built in the past decade were underutilized for lack of reliable middleware or well-coded applications. Potential advantages of cloud computing lie in its low cost and simplicity to both providers and users.

New Challenges: Utility computing focuses on a business model, by which customers receive computing resources from a paid service provider. All grid/cloud platforms are regarded as utility service providers. However, cloud computing offers a broader concept than utility computing. Distributed cloud applications run on any available servers in some edge networks. Major technological challenges include all aspects of computer science and engineering. For example, we need new network-efficient processors, scalable memory and storage schemes, distributed OS, middleware for machine virtualization, new programming model, effective resource management, and application program development in distributed systems that explore massive parallelism at all processing levels.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

Table 1.6 Classification of Distributed Parallel Computing Systems

Functionality, Applications	Multicomputer Clusters [11, 21]	Peer-to-Peer Networks [13, 33]	Data/Computational Grids [4, 14, 33]	Cloud Platforms [7, 8, 22, 31]
Architecture, Network Connectivity and Size	Network of compute nodes interconnected by SAN, LAN, or WAN, hierarchically	Flexible network of client machines logically connected by an overlay network	Heterogeneous cluster of clusters connected by high-speed network links over selected resource sites.	Virtualized cluster of servers over many datacenters via service-level agreement
Control and Resources Management	Homogeneous nodes with distributed control, running Unix or Linux	Autonomous client nodes, free in and out, with distributed self-organization	Centralized control, server oriented with authenticated security, and static resources management	Dynamic resource provisioning of servers, storage, and networks over massive datasets
Applications and network-centric services	High-performance computing, search engines, and web services, etc.	Most appealing to business file sharing, content delivery, and social networking	Distributed super-computing, global problem solving, and datacenter services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM BlueGene, Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, Papestry, KaZaA, Skype, JXTA, and .NET	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, IBM IntraGrid, etc.	Google App Engine, IBM Bluecloud, Amazon Web Service(AWS), and Microsoft Azure,

1.3.1 Clusters of Cooperative Computers

A computing cluster is built by a collection of interconnected stand-alone computers, which work cooperatively together as a single integrated computing resource. To handle heavy workload with large datasets, clustered computer systems have demonstrated impressive results in the past.

Cluster Architecture: Figure 1.9 shows the architecture of a typical sever cluster built around a low-latency and high-bandwidth interconnection network. This network can be as simple as a SAN (e.g. Myrinet) or a LAN (e.g. Ethernet). To build a larger cluster with more nodes, the IN can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches. Through hierarchical construction using SAN, LAN, or WAN, one can build scalable clusters with increasing number of nodes. The whole cluster is connected to the Internet via a VPN gateway. The gateway IP address could be used to locate the cluster over the cyberspace.

Single-System Image: The system image of a computer is decided by the way the OS manages the shared cluster resources. Most clusters have loosely-coupled node computers. All resources of a server node is managed by its own OS. Thus, most clusters have multiple system images coexisting simultaneously. Greg Pfister [27] has indicated that an ideal cluster should merge multiple system images into a *single-system image* (SSI) at various operational levels. We need an idealized cluster operating system or some middlware to support SSI at various levels, including the sharing of all CPUs, memories, and I/O across all computer nodes attached to the cluster.

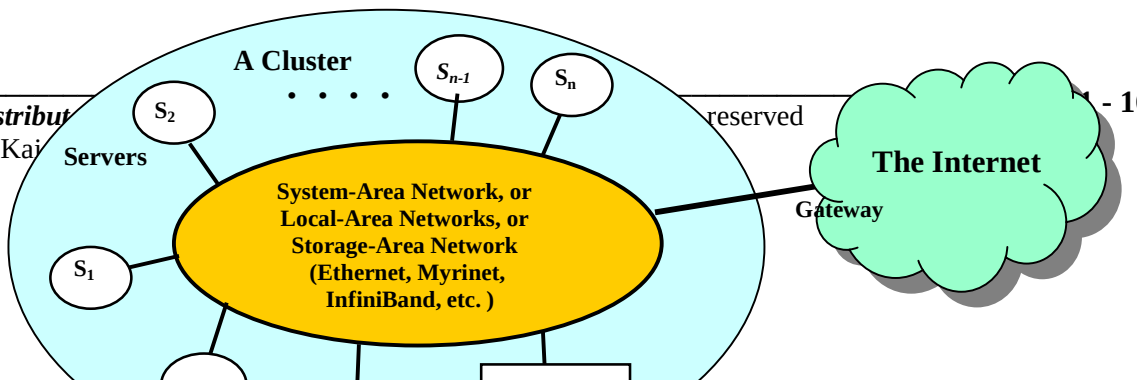

Figure 1.9 A cluster of servers (S_1, S_2,…,S_n) interconnected by a high-bandwidth system-area or local-area network with shared I/O devices and disk arrays. The cluster acts as a single computing node attached to the Internet throught a gateway.

A single system image is the illusion, created by software or hardware that presents a collection of resources as an integrated powerful resource. SSI makes the cluster appear like a single machine to the user, applications, and network. A cluster with multiple system images is nothing but a collection of independent computers. Figure 1.10 shows the hardware and software architecture of a typical cluster system. Each node computer has its own operating system. On top of all operating systems, we deploy some two layers of middleware at the user space to support the high availability and some SSI features for shared resources or fast MPI communications.

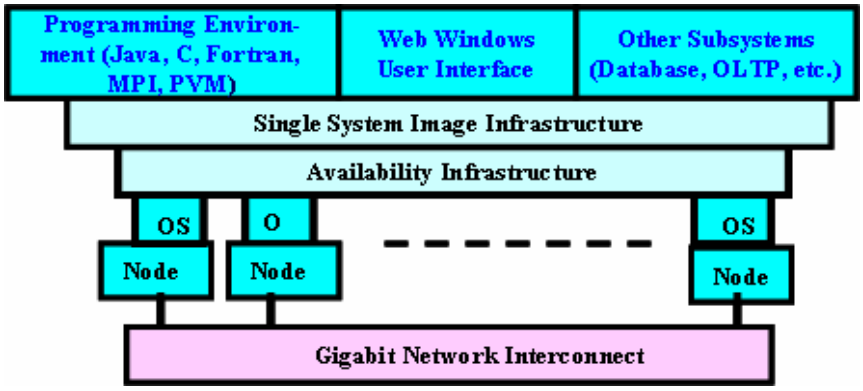

Figure 1.10 The architecture of a working cluster with full hardware, software, anAd middleware support for availability and single system image.

For example, since memory modules are distributed at different server nodes, they are managed independently over disjoint address spaces. This implies that the cluster has multiple images at the memory-reference level. On the other hand, we may want all distributed memories to be shared by all servers by forming a *distributed shared memory* (DSM) with a single address space. A DSM cluster thus has a *single-system image* (SSI) at the memory-sharing level. Cluster explores data parallelism at the job level with high system availability.

Cluster Design Issues: Unfortunately, a cluster-wide OS for complete resource sharing is not available yet. Middleware or OS extensions were developed at the user space to achieve SSI at selected functional

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

levels.Without the middleware, the cluster nodes cannot work together effectively to achieve cooperative computing. The software environments and applications must rely on the middleware to achieve high performance. The cluster benefits come from scalable performance, efficient message-passing, high system availability, seamless fault tolerance, and cluster-wide job management as summarized in Table 1.7. Clusters and MPP designs are treated in Chapter 3.

Table 1.7 Critical Cluster Design Issues and Feasible Implementations

Features	Functional Characterization	Feasible Implementations
Availability Support	Hardware and software support for sustained high availability in cluster	Failover, failback, checkpointing, roll back recovery, non-stop OS, etc
Hardware Fault-Tolerance	Automated failure management to eliminate all single points of failure	Component redundancy, hot swapping, RAID, and multiple power supplies, etc.
Single-System Image (SSI)	Achieving SSI at functional level with hardware and software support, middleware, or OS extensions.	Hardware mechanisms or middleware support to achieve distributed shared memory (DSM) at coherent cache level.
Efficient Communications	To reduce message-passing system overhead and hide latencies	Fast message passing , active messages, enhanced MPI library, etc.
Cluster-wide Job Management	Use a global job management system with better scheduling and monitory	Apply single-job management systems such as LSF, Codine, etc
Dynamic Load Balancing	Balance the workload of all processing nodes along with failure recovery	Workload monitory, process migration, job replication and gang scheduling, etc.
Scalability and Programmability	Adding more servers to a cluster or adding more clusters to a Grid as the workload or data set increases	Use scalable interconnect, performance monitory, distributed execution environment, and better software tools

1.3.2 Grid Computing Infrastructures

In 30 years, we have experienced a natural growth path from Internet to web and grid computing services. Internet service such as the *Telnet* command enables connection from one computer to a remote computer. The Web service like *http* protocol enables remote access of remote web pages. Grid computing is envisioned to allow close interactions among applications running on distant computers, simultaneously. *Forbes Magazine* has projected the global grow of IT-based economy from $1 Trillion in 2001 to 20 Trillion by 2015. The evolution from Internet to web and grid services is certainly playing a major role to this end.

Computing Grids: Like an electric-utility power grid, a *computing grid* offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. Grid is often constructed across LAN,WAN, or Internet backbone networks at regional, national, or global scales. Enterprises or organizations present grids as integrated computing resources. They can be viewed also as *virtual platforms* to support *virtual organizations*. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops and PDAs can be used as access devices to a grid system. The grid software and middleware are needed as applications and utility libraries and databases, Special instruments are used to search for life in the galaxy, for example.

Figure 1.11 shows the concept of a computational grid built over three resource sites at the University of Wisconsin at Madison, University of Illinois at Champaign-Urbana, and California Institute of Technology. The three sites offer complementary computing resources, including workstations, large

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

servers, mesh of processors, and Linux clusters to satisfy a chain of computational needs. Three steps are shown to the chain of weather data collection, distributed computations, and result analysis in atmospheric simulations. Many other even larger computational grids like NSF TeraGrid and EGEE, and ChinaGrid have built similar national infrastructures to perform distributed scientific grid applications.

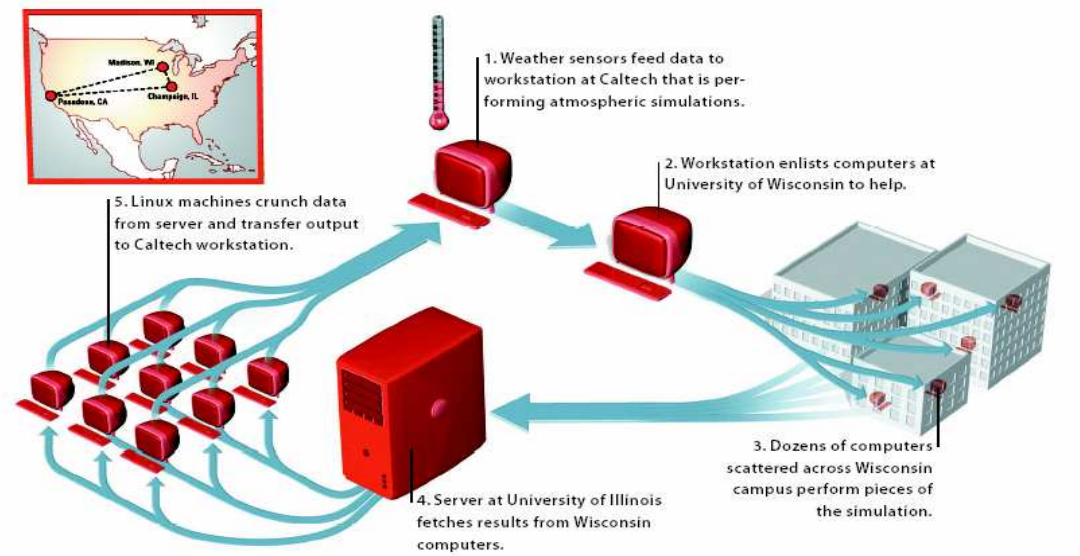

Figure 1.11 An example computational Grid built over specialized computers at three resource sites at Wisconsin, Caltech, and Illinois. (Courtesy of Michel Waldrop, "Grid Computing", IEEE *Computer Magazine*, 2000. [34])

Grid Families: Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platform development by IBM, Microsoft, Sun, HP, Dell, Cisco, EMC, Platform Computing, etc New *grid service providers* (GSP) and new grid applications are opened rapidly, similar to the growth of Internet and Web services in the past two decades. In Table 1.8, we classify grid systems developed in the past decade into two families: namely *computational or data grids* and *P2P grids.* These computing grids are mostly built at the national level. We identify their major applications, representative systems, and lesson learned so far. Grid Computing will be studied in Chapters 4 and 8.

Table 1.8 Two Grid Computing Infrastructures and Representative Systems

Design Issues	Computational and Data Grids	P2P Grids
Grid Applications reported	Distributed Supercomputing, National Grid Initiatives, etc	Open grid with P2P flexibility, all resources from client machines
Representative Systems	TeraGrid in US, ChinaGrid, UK e-Science, etc.	JXTA, FightAid@home, SETI@home
Development Lessons learned	Restricted user groups, middleware bugs, rigid protocols to acquire resources	Unreliable user-contributed resources, limited to a few apps.

1.3.3 Service-Oriented Architectures (SOA)

Technology has advanced at breakneck speed up over the last decade with many changes that

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

are still occurring. However in this chaos, the value of building systems in terms of services has grown in acceptance and it has become a core idea of most distributed systems. Always one builds systems in layered fashion as sketched below in Fig.1.12. Here we use the rather clumsy term "entity" to denote the abstraction used as the basic building block. In Grids/Web Services, Java and CORBA, an entity is respectively a service, Java object or a CORBA distributed object in a variety of languages.

The architectures build on the traditional 7 OSI layers providing the base networking abstractions. On top of this we have a base software environment which would be .NET or Apache Axis for Web Services, the Java Virtual Machine for Java or a Broker network for CORBA. Then on top of this base environment, one builds a higher-level environment reflecting the special features of the distributing computing environment and represented by the green box in Fig.1.12. This starts with Entity Interfaces and Inter-entity communication which can be thought of as rebuilding the top 4 OSI layers but at the entity and not the bit level.

The entity interfaces correspond to the WSDL, Java method and CORBA IDL specifications in these example distributed systems. These interfaces are linked with customized high level communication systems – SOAP, RMI and IIOP in the three examples. These communication systems support features including particular message patterns (such as RPC or remote procedure call), fault recovery and specialized routing. Often these communications systems are built on message oriented middleware (enterprise bus) infrastructure such as WebSphereMQ or JMS (Java Message Service) which provide rich functionality and support virtualization of routing, sender and recipients.

In the case of fault tolerance, we find features in the Web Service Reliable Messaging framework that mimic the OSI layer capability (as in TCP fault tolerance) modified to match the different abstractions (such as messages versus packets, virtualized addressing) at the entity levels. Security is a critical capability that either uses or re-implements the capabilities seen in concepts like IPSec and secure sockets in the OSI layers. Entity communication is supported by higher level services for registries, metadata and management of the entities discussed in Section 4.4.

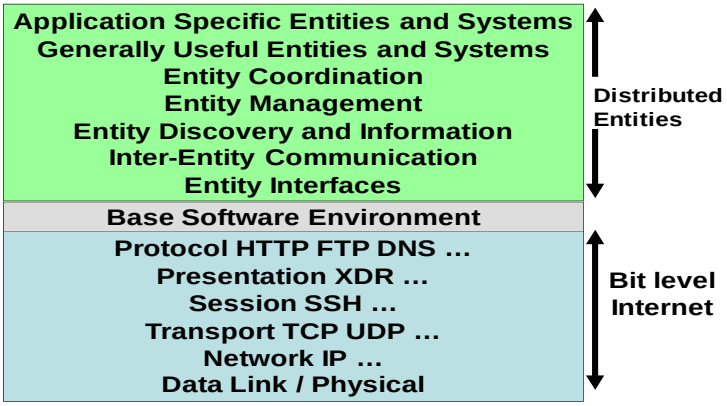

Fig. 1.12. General layered architecture for distributed entities

Here one might get several models with for example Jini and JNDI illustrating different approaches within the Java distributed object model. The CORBA Trader Service, UDDI, LDAP and ebXML are other examples of discovery and information services described in Section 4.4. Management services include service state and lifetime support; examples include the CORBA Life Cycle and Persistent State, the different Enterprise Javabean models, Jini's lifetime model and a suite of Web service specifications that

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

we will study further in Chapter 4.

We often term this collection of entity level capabilities that extend the OSI stack the “Internet on the Internet”: or the “Entity Internet built on the Bit Internet”. The above describes a classic distributed computing model and as well as intense debate on the best ways of implementing distributed systems there is competition with "centralized but still modular" approaches where systems are built in terms of components in an Enterprise Javabean or equivalent approach.

The latter can have performance advantages and offer a "shared memory" model allowing more convenient exchange of information. However the distributed model has two critical advantages -- namely higher performance (from multiple CPU's when communication is unimportant) and a cleaner separation of software functions with clear software re-use and maintenance advantages. We expect the distributed model to gain in popularity as the default approach to software systems. Here the early CORBA and Java approaches to distributed systems are being replaced by the service model shown in Fig.1.13.

Loose coupling and support of heterogeneous implementations makes services more attractive than distributed objects. The architecture of this figure underlies modern systems with typically two choice of service architecture -- Web Services or REST systems. These are further discussed in chapter 4 and have very distinct approaches to building reliable interoperable systems. in Web services, one aims to fully specify all aspects of the service and its environment. This specification is carried with communicated messages using the SOAP protocol. The hosting environment then becomes a universal distributed operating system with fully distributed capability carried by SOAP messages.

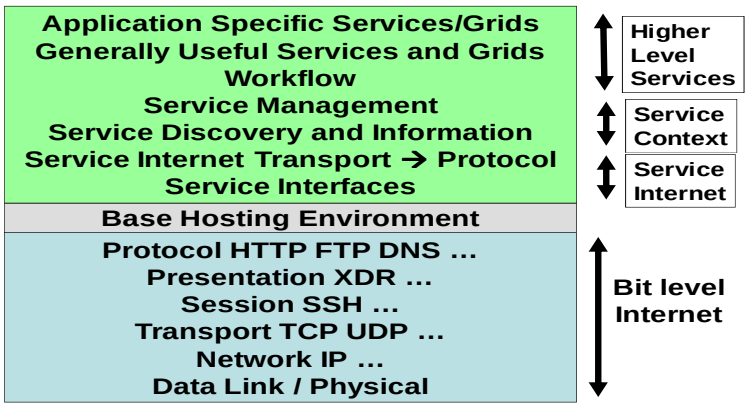

Figure 1.13 Layered architecture for web srvices and grids

Experience has seen mixed success for this approach as it has been hard to agree on key parts of the protocol and even harder to robustly and efficiently implement the universal processing of the protocol (by software like Apache Axis). In the REST approach, one adopts simplicity as the universal principle and delegated most of the hard problems to application (implementation specific) software. In a Web Service language REST has minimal information in the header and the message body (that is opaque to generic message processing) carries all needed information. REST architectures are clearly more appropriate to rapidly technology environments that we see today.

However, the ideas in Web Services are important and probably will be needed in mature systems at a different level in stack (as part of application). Note that REST can use XML schemas but not used that are part of SOAP; "XML over HTTP" is a popular design choice. Above the communication and management layers, we have the capability to compose new entities or distributed programs by integrating several

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

entities together as sketched in Fig.1.14. In CORBA and Java, the distributed entities are linked with remote procedure calls and the simplest way to build composite applications is to view the entities as objects and use the traditional ways of linking them together. For Java, this could be as simple as writing a Java program with method calls replaced by RMI (Remote Method Invocation) while CORBA supports a similar model with a syntax reflecting the C++ style of its entity (object) interfaces.

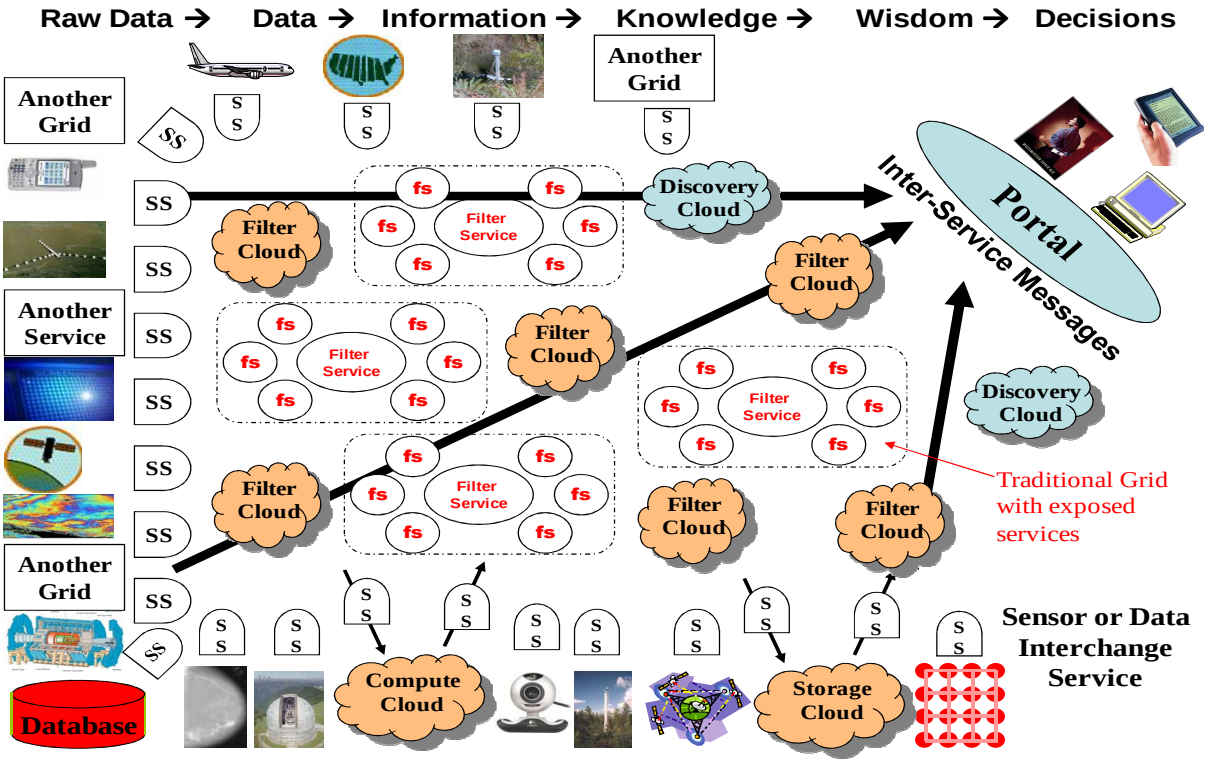

Figure 1.14. Grids of Clouds and Grids where SS refers to Sensor Service and fs to a filter or transforming service.

There are also very many distributed programming models built on top of these of these basic constructs. For Web Services, workflow technologies are used to coordinate or orchestrate services with special specifications used to define critical business process models such as two phase transactions. In section 4.2, we describe the general approach used in workflow, the BPEL Web Service standard and several important workflow approaches Pegasus, Taverna, Kepler, Trident and Swift. In all approaches one is building collections of services which together tackle all or part of a problem. As always one ends with systems of systems as the basic architecture.

Allowing the term Grid to refer to a single service or represent a collection of services, we find the architecture of Fig.1.14. Here sensors represent entities (such as instruments) that output data (as messages) and Grids and Clouds represent collections of services that have multiple message-based inputs and outputs. The figure emphasizes the system of systems or "Grids and Clouds of Grids and Clouds" architecture. Most distributed systems requires a web interface or portal shown in Fig.1.14 and two examples (OGFCE and HUBzero) are described in Section 4.3 using both Web Service (portlet) and Web 2.0 (gadget) technologies.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

1.3.4 Peer-to-Peer Network Families

A well-established distributed system is the *client-server architecture*. Client machines (PC and workstations) are connected to a central server for compute, or Email, file access, database applications. The *peer-to-peer* (P2P) *architecture* offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented. In this section, we introduce P2P systems at the physical level and overlay networks at the logical level.

P2P Networks: In a P2P system, every node acts as both a client and a server, providing part of the system resources. Peer machines are simply client computers connected to the Internet. All client machines act autonomously to join or leave the system freely. This implies that no master-slave relationship exists among the peers. No central coordination or no central database is needed. In other words, no peer machine has a global view of the entire P2P system. The system is self-organizing with distributed control.

The architecture of a P2P network is shown in Fig.1.15 at two abstraction levels. Initially, the peers are totally unrelated. Each peer machine joins or leaves the P2P network, voluntarily. Only the participating peers form the *physical network* at any time. Unlike the cluster or grid, a P2P network does not use a dedicated interconnection network. The physical network is simply an ad hoc network formed at various Internet domains randomly using TCP/IP and NAI protocols. Thus, the physical network varies in size and topology dynamically due to the free membership in the P2P network.

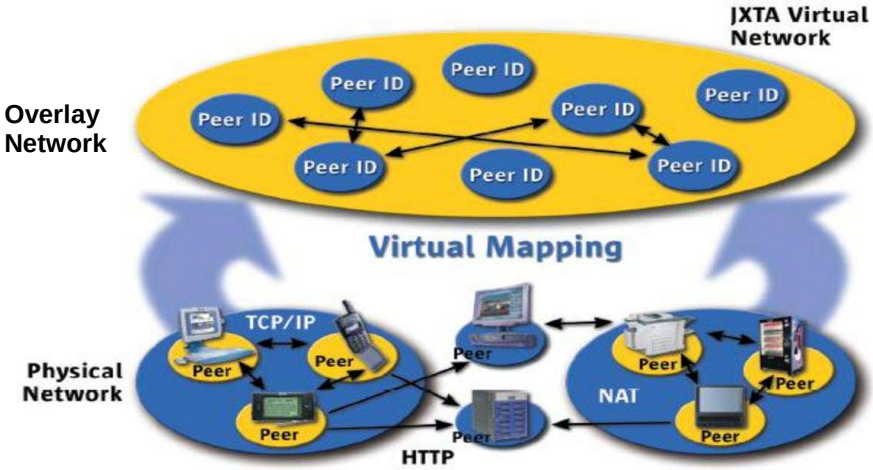

Figure 1.15 The structure of a peer-to-peer system by mapping a physical network to a virtual overlay network (Courtesy of JXTA, http://www.jxta.com)

Overlay Networks: Data items or files are distributed in the participating peers. Based on communication or file-sharing needs, the peer IDs form an *overlay network* at the logical level. This overlay is a virtual network formed by mapping each physical machine with its ID, logically through a virtual mapping shown in Fig.1.7. When a new peer joins the system, its peer ID is added as a node in the overlay network. When an existing peer leaves the system, its peer ID is removed from the overlay network, automatically. Therefore, it is the P2P overlay network that characterizes the logical connectivity among the peers.

There are two types of overlay networks: *unstructured* versus *structured.* An *unstructured overlay network* is characterized by a random graph. There is no fixed route to send messages or file among the nodes. Often, flooding is applied to send a query to all nodes in an unstructured overlay, thus ending up with heavy network traffic and nondeterministic search results. *Structured overlay networks* follow certain connectivity topology and rules to insert or remove nodes (Peer IDs) from the overlay graph. Routing mechanisms are developed to take advantage of the structured overlays.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

P2P Application Families: Based on applications, we classify P2P networks into four classes in Table 1.9. The first family is for distributed file sharing of digital contents (music, video, etc.) on the P2P network. This includes many popular P2P networks like Gnutella, Napster, BitTorrent, etc. Collaboration P2P networks include MSN or Skype chatting, instant messaging, collaborative design, etc. The third family is for distributed P2P computing in specific applications. For example, SETI@home provides 25 Tflops distributed computing power, collectively, over 3 million Internet host machines. Other P2P platforms like JXTA, .NET, and FightingAID@home, support naming, discovery, communication, security, and resource aggregation in some P2P applications. We will study these topics in Chapters 5 and 8.

Table 1.9 Major Categories of Peer-to-Peer Network Families

System Features	Distributed File Sharing	Collaborative Platform	Distributed P2P Computing	Peer-to-Peer Platform
Attractive Applications	Content distribution of MP3 music, video, open software, etc.	Instant Messaging, Collaborative design and gaming	Scientific exploration and social networking	Open networks for public resources
Operational Problems	Loose security and on-line copyright violations	Lack of trust, disturbed by spam, privacy, and peer collusions	Security holes, selfish partners, and peer collusion	Lack of standards or protection protocols
Example Systems	Gnutella, Napster, eMule, BitTorrent, Aimster, KaZaA, etc.	ICQ, AIM, Groove, Magi, Multiplayer Games, Skype, etc.	SETI@home, Geonome@ home, etc.	JXTA, .NET, FightingAid@ home, etc.

P2P Computing Challenges: P2P computing faces three types of heterogeneity problems in hardware, software and network requirements. There are too many hardware models and architectures to select from. Incompatibility exists between software and OS. Different network connections and protocols make it too complex to apply in real applications. We need system scalability as the workload increases. System scaling is directly related to performance and bandwidth.

Data location is also important to affect collective performance. Data locality, network proximity, and interoperability are three design objectives in distributed applications. The P2P performance is affected by routing efficiency and self-organization by the participating peers. Fault Tolerance, failure management, and load balancing are other important issues in using overlay networks. Lack of trust among the peers posts another problem. Peers are strangers to each other. Security, privacy, and copyright violations are major worries by industry to apply P2P technology in business applications.

1.3.5 Virtualized Cloud Computing Infrastructure

Gordon Bell, Jim Gray, and Alex Szalay [3] have advocated: "Computational science is changing to be data-intensive. Supercomputers must be balanced systems, not just CPU farms but also petascale I/O and networking arrays." In the future, working with large data sets will typically mean sending the computations (programs) to the data, rather than copying the data to the workstations. This reflects the trend in IT that moves computing and data from desktops to large datacenters, where on-demand provision of software, hardware, and data as a service. Data explosion leads to the idea of cloud computing.

Cloud computing has been defined differently by many users and designers. Just to cite a few, IBM being a major developer of cloud computing has defined cloud computing as: "A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive, user-facing applications, allow workloads to be deployed and scaled-out

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

quickly through the rapid provisioning of virtual machines or physical machines, support redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures; and monitor resource use in real time to enable rebalancing of allocations when needed."

Internet Clouds: Cloud computing applies a virtualized platform with elastic resources on-demand by provisioning hardware, software, and datasets, dynamically. The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at datacenters. Cloud computing leverages its low cost and simplicity that benefit both users and the providers. Machine virtualization has enabled such cost-effectiveness. Cloud computing intends to satisfy many heterogeneous user applications simultaneously. The cloud ecosystem must be designed to be secure, trustworthy, and dependable.

Ian Foster defined cloud computing as follows: "A large-scale distributed computing paradigm that is driven by economics of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet". Despite some minor differences in the above definitions, we identify six common characteristics of Internet clouds as depicted in Fig.1.16.

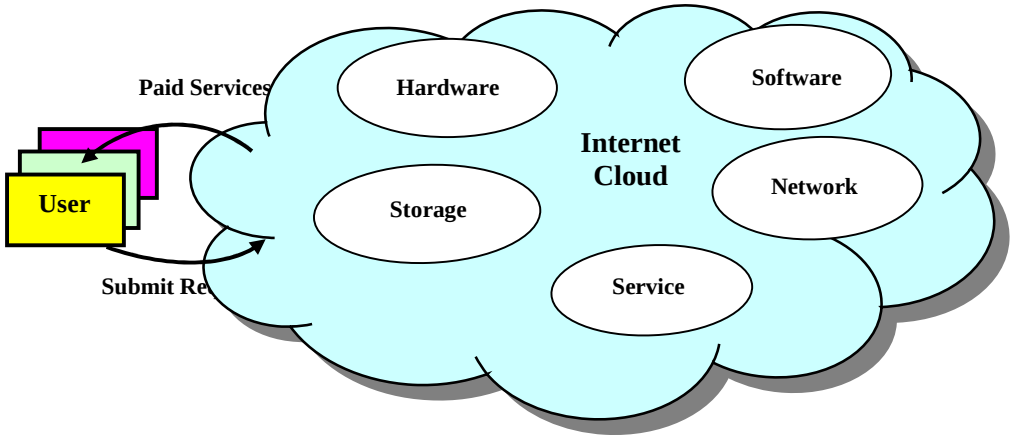

Figure 1.16 Concept of virtualized resources provisioning through the Internet cloud, where the hardware, software, storage, network and services are put together to form a cloud platform.

(1) Cloud platform offers a scalable computing paradigm built around the datacenters.
(2) Cloud resources are dynamically provisioned by datacenters upon user demand.
(3) Cloud system provides computing power, storage space, and flexible platforms for upgraded web-scale application services.
(4) Cloud computing relies heavily on the virtualization of all sorts of resources.
(5) Cloud computing defines a new paradigm for collective computing, data consumption and delivery of information services over the Internet.
(6) Clouds stress the cost of ownership reduction in mega datacenters.

Basic Cloud Models: Traditionally, a distributed computing system tends to be owned and operated by an autonomous administrative domain (e.g., a research laboratory or company) for on-premises computing needs. However, these traditional systems have encountered several performance bottlenecks: constant system maintenance, poor utilization and increasing costs associated with hardware/software upgrades. Cloud computing as an on-demand computing paradigm resolves or relieves from these problems. In

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

Figure 1.17, we introduce the basic concepts of three cloud computing service models. More cloud details are given in Chapters 7, 8 and 9.

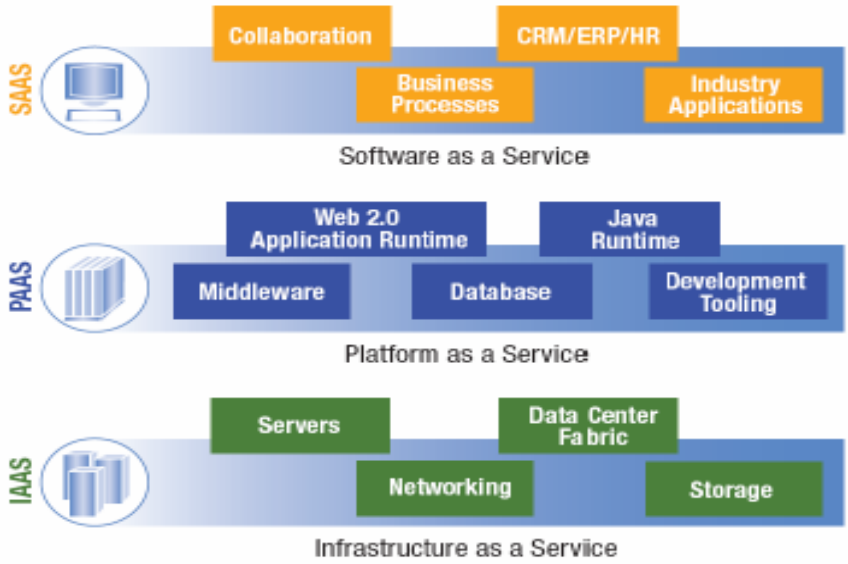

Figure 1.17 Basic concept of cloud computing models and services provided
(Courtesy of IBM Corp. 2009)

Infrastructure as a Service (IaaS): This model allows users to server, storage, networks, and datacenter fabric resources. The user can deploy and run on multiple VMs running guest OSes on specific applications. The user does not manage or control the underlying cloud infrastructure, but can specifv when to request and release the needed resources.

Platform as a Service (PaaS)*:* This model provides the user to deploy user-built applications onto a virtualized cloud platform The platform include both hardware and software integrated with specific programming interfaces. The provide supplies the API and software tools (e.g., Java, python, Web 2.0, .Net). The user is freed from managing the underlying cloud infrastructure.

Software as a Service (SaaS): This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, CRM (*consumer relationship mamagment)*, ERP (*enterprise resources planning*), HR (*human resources*) and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

Internet clouds offer four deployment modes: *private*, *public*, *managed*, and *hybrid* [22]. These modes demand different levels of security implications. The different service level agreements and service deployment modalities imply the security to be a shared responsibility of all the cloud providers, the cloud resource consumers and the third party cloud enabled software providers. Advantages of cloud computing have been advocated by many IT experts, industry leaders, and computer science researchers.

Benefits of Outsourcing to The Cloud: Outsourcing local workload and/or resources to the cloud has

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

become an appealing alternative in terms of operational efficiency and cost effectiveness. This outsourcing practice particularly gains its momentum with the flexibility of cloud services from no lock-in contracts with the provider and the use of a pay-as-you-go pricing model. Clouds are primarily driven by economics—the pay-per-use pricing model similar to basic utilities of electricity, water and gas. From the consumer's perspective, this pricing model for computing has relieved many issues in IT practices, such as the burden of new equipment purchases and the ever-increasing costs in operation of computing facilities (e.g., salary for technical supporting personnel and electricity bills).

Specifically, a sudden surge of workload can be effectively dealt with; and this also has an economic benefit in that it helps avoid over provisioning of resources for such a surge. From the provider's perspective, charges imposed for processing consumers' service requests—often exploiting underutilized resources—are an additional source of revenue. Since the cloud service provider has to deal with a diverse set of consumers, including both regular and new/one-off consumers, and their requests most likely differ from one another, the judicious scheduling of these requests plays a key role in the efficient use of resources for the provider to maximize its profit and for the consumer to received satisfactory service quality (e.g., response time). Recently, Amazon introduced EC2 Spot instances for which the pricing dynamically changes based on the demand-supply relationship (http://aws.amazon.com/ec2/spot-instances/). Accountability and security are two other major concerns associated with the adoption of clouds. These will be treated in Chapters 7.

Chapter 6 offers details of datacenter design, cloud platform architecture and resource deployment, Chapter 7 provides major cloud platforms built and various cloud services being offered. Listed below are 8 motivations of adapting the cloud for upgrading Internet applications and web services in general.

(1). Desired location in areas with protected space and better energy efficiency.
(2). Sharing of peak-load capacity among a large pool of users, improving the overall utilization
(3). Separation of infrastructure maintenance duties from domain-specific application development.
(4). Significant reduction in cloud computing cost, compared with traditional computing paradigms.
(5). Cloud computing programming and application development
(6). Service and data discovery and content/service distribution
(7). Privacy, security, copyright, and reliability issues
(8). Service agreements, business models, and pricing policies**.**

Representative Cloud Providers : In Table 1.10, we summarize the features of three cloud platforms built up to 2008. The Google platform is a closed system, dynamically built over a cluster of servers,. These servers selected from over 460,000 Google servers worldwide. This platform is proprietary in nature, only programmable by Google staff. Users must order the standard services through Google. The IBM BlueCloud offers a total system solution by selling the entire server cluster plus software packages for resources management and monitoring, WebSphere 2.0 applications, DB2 databases, and virtualization middleware. The third cloud platform is offered by Amazon as a custom-service utility cluster. Users lease special subcluster configuration and torage space to run custom-coded applications.

The IBM BlueCloud allows cloud users to fill out a form defining their hardware platform, CPU, memory, storage, operating system, middleware, and team members and their associated roles. A SaaS bureau may order travel or secretarial services from a common cloud platform. The MSP coordinates service delivery and pricing by user specifications. Many IT companies are now offering cloud computing services. We desire a software environment that provides many useful tools to build cloud applications over large datasets. In addition to MapReduce, BigTable, EC2, and 3S and the established environment packages like Hadoop, AWS, AppEngine, and WebSphere2. Details of these cloud systems are given in Chapter 7 and 8.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

Table 1.10 Three Cloud Computing Platforms and Underlying Technologies [21]

Features	Google Cloud [18]	IBM BlueCloud [7]	Amazon Elastic Cloud
Architecture and Service Models applied	Highly scalable server clusters, GFS, and data-centers operating with PaaS or SaaS models	A sever cluster with limited scalability for distributed problem solving and web-scale under a PaaS model	A 2000-node utility cluster (iDataPlex) for distributed computing/storage services under the IaaS model
Technology, Virtualization, and Reliability	Commodity hardware. Application-level API, simple service, and high reliability	Custom hardware, Open software, Hadoop library, virtualization with XEN and PowerVM, high reliability	e-commerce platform, virtualization based on XEN, and simple reliability
System Vulnerability, and Security Resilience	Datacenter security is loose, no copyright protection, Google rewrites desktop applications for web	WebSphere-2 security, PowerVM could be tuned for security protection, and access control and VPN support	Rely on PKI and VPN for authentication and access control, lack of security defense mechanisms

1.3 Performance, Security, and Energy-Efficiency

In this section, we introduce the fundamental design principles and rules of thumb for building massively distributed computing systems. We study scalability, availability, programming models, and security issues that are encountered in clusters, grids, P2P networks, and Internet clouds.

1.4.1 System Performance and Scalability Analysis

Performance metrics are needed to measure various distributed systems. We present various dimensions of scalability and performance laws. Then we examine system scalability against OS image and the limiting factors encountered.

Performance Metrics: We have used *CPU speed* in MIPS and *network bandwidth* in Mbps in Section 1.3.1 to estimate processor and network performance. In a distributed system, the performance is attributed to a large number of factors. The s*ystem throughput* is often measured by the MIPS rate, Tflops (*Tera floating-point operations per second*), TPS (*transactions per second*), etc. Other measures include the job *response time* and *network latency.*

We desire to use an interconnection network that has low latency and high bandwidth. System overhead is often attributed to OS boot time, compile time, I/O data rate, and run-time support system used. Other pereformanc-related metrics include the *quality of service* (QoS) for Internet and Web services; *system availability* and *dependability;* and *security resilience* for system defense against network attacks. We will study some of these in remaining subsections.

Dimensions of Scalability: We want to design a distributed system to achieve scalable performance. Any resource upgrade in a system should be backward compatible with the existing hardware and software resources. Overdesign may not be cost-effective. System scaling can increase or decrease resources depending on many practical factors. We characterize the following dimensions of scalability in parallel

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

and distributed systems.

a) ***Size Scalability:*** This refers to achieve higher performance or performing more functionality by increasing the *machine size*. The word "size" refers to adding the number of processors; more cache, memory, storage or I/O channels. The most obvious way to simple counting the number of processors installed. Not all parallel computer or distributed architectures are equally size-scalable. For example, IBM S2 was scaled up to 512 processors in 1997. But in 2008, the IBM BlueGene/L system can scale up to 65,000 processors.

b) ***Software Scalability:*** This refers to upgrades in OS or compilers, adding mathematical and engineering libraries, porting new application software, and install more user-friendly programming environment. Some software upgrade may not work with large system configurations. Testing and fine-tuning of new software on larger system is a non-trivial job.

c) ***Application scalability:*** This refers to the match of *problem size* scalability with the *machine size* scalability. Problem size affects the size of the data set or the workload increase. Instead of increasing machine size, we enlarge the problem size to enhance the system efficiency or cost-effectiveness.

d) ***Technology Scalability:*** This refers to a system that can adapt to changes in building technologies, such as those component and networking technologies discussed in Section 3.1. Scaling a system design with new technology must consider three aspects: *time*, *space*, and *heterogeneity*. Time refers to generation scalability. Changing to new-generation processors, one must consider the impact to motherboard, power supply, packaging and cooling, etc. Based on the past experience, most system upgrade their commodity processors every 3 to 5 years. Space is more related to packaging and energy concerns. Heterogeneity scalability demands harmony and portability among different component suppliers.

Scalability vs. OS Image Count: In Fig.1.18, we estimate the *scalable performance* against the *multiplicity of OS images* in distributed systems deployed up to 2010. Scalable performance implies that the system can achieve higher speed performance by adding more processors or servers, enlarging the physical node memory size, extending the disk capacity, or adding more I/O channels, etc. The OS image is counted by the number of independent OS images observed in a cluster, grid, P2P network, or the cloud. We include the SMP and NUMA in the comparison. An SMP server has a single system image. Which could be a single node in a large cluster. By 2010 standard, the largest shared–memory SMP node has at most hundreds of processors. This low scalability of SMP system is constrained by the packaging and system-interconnect used.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

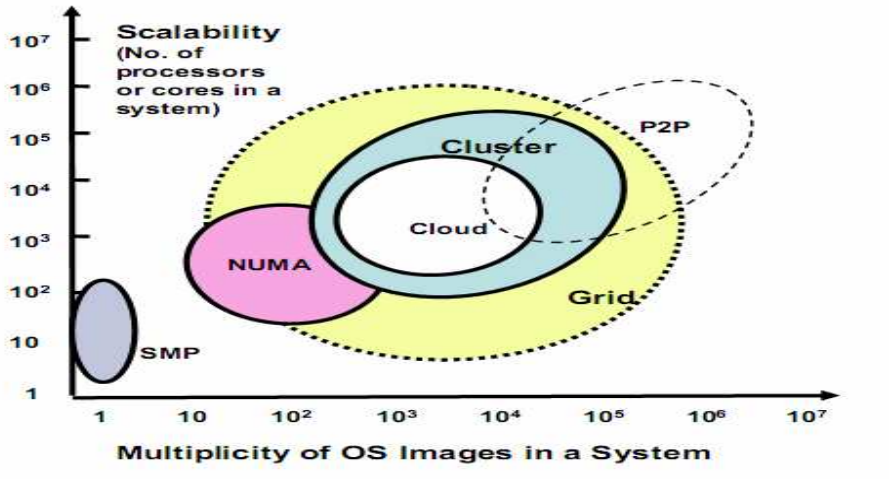

Figure 1.18 System scalability versus multiplicity of OS images in HPC clusters, MPP, and grids and HTC systems like P2P networks and the clouds. (The magnitude of scalability and OS image count are estimated based on system configurations deployed up to 2010. The SMP and NUMA are included for comparison purpose)

The NUMA machines are often made out of SMP nodes with distributed shared memories. A NUMA machine can run with multiple operating systems. It can scale to a few thousands of processors communicating with MPI library. For example, an NUMA machine may have 2048 processors running by 32 SMP operating systems. Thus, there are 32 OS images in the 2048-processor NUMA system. The cluster nodes can be either SMP servers or high-end machines that are loosely coupled together. Therefore, clusters have much higher scalability than NUMA machines. The number of OS images in a cluster is counted by the cluster nodes concurrently in use. The cloud could be a virtualized cluster. By 2010, the largest cloud in use commercially has size that can scale up to a few thousand VMs at most.

Reviewing the fact many cluster nodes are SMP (multiprocessor) or multicore servers, the total number of processors or cores in a cluster system is one or two orders of magnitude greater than the number of OS images running in the cluster. The node in a computational grid could be either a server cluster, or a mainframe, or a supercomputer, or a *massively parallel processor* (MPP). Therefore, OS image count in a large grid structure could be hundreds or thousands times fewer than the total number of processors in the grid. A P2P network can easily scale to millions of independent peer nodes, essentially desktop machines. The performance of a P2P file-sharing network depends on the *quality of service* (QoS) received in a public networks. We plot the low-speed P2P networks in Fig.1.15. Internet clouds are evaluated similarly to the way we assess cluster performance.

Amdahl's Law: Consider the execution of a given program on a uniprocessor workstation with a total execution time of $\boldsymbol{T}$ minutes. Now, the program has been parallelized or partitioned for parallel execution on a cluster of many processing nodes. Assume that a fraction $\boldsymbol{\alpha}$ of the code must be executed sequentially, called the *sequential bottleneck*. Therefore, (1- $\boldsymbol{\alpha}$) of the code can be compiled for parallel execution by $\boldsymbol{n}$ processors. The total execution time of the program is calculated by $\boldsymbol{\alpha T + (1-\alpha)T/n}$, where the first term is the sequential execution time on a single processor. The second term is the parallel execution time on $\boldsymbol{n}$ processing nodes.

We will ignore all system or communication overheads, I/O time, or exception handling time in the then following speedup analysis. Amdahl's Law states that: The *speedup factor* of using the $\boldsymbol{n}$-processor system over the use of a single processor is expressed by:

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

$$\textbf{Speedup} = S = T / [\, \alpha T + (1-\alpha)T/n \,] = 1 / [\, \alpha + (1-\alpha)/n \,] \qquad (1.1)$$

The maximum speedup of n is achieved, only if the *sequential bottleneck* α is reduced to zero or the code is fully parallelizable with $\alpha = 0$. As the cluster becomes sufficiently large, i.e. $n \to \infty$, we have $S = 1/\alpha$, an upper bound on the speedup S. Surprisingly, this upper bound is independent of the cluster size n. Sequential bottleneck is the portion of the code that cannot be parallelized. For example, the maximum speedup achieved speedup is 4, if $\alpha = 0.25$ or $1-\alpha = 0.75$, even we use hundreds of processors. Amdahl's law teaches us that we should make the sequential bottleneck as small as possible. By increasing the cluster size alone may not give us a good speedup we expected.

Problem with Fixed Workload: In Amdahl's law, we have assumed the same amount of workload for both sequential and parallel execution of the program with a fixed problem size or dataset. This was called *fixed-workload speedup* by Hwang and Xu [14]. To execute a fixed workload on n processors, parallel processing may lead to a s*ystem efficiency* defined as follows:

$$E = S / n = 1 / [\, \alpha n + 1-\alpha \,] \qquad (1.2)$$

Very often the system efficiency is rather low, especially when the cluster size is very large. To execute the aforementioned program on a cluster with $n = 256$ nodes, extremely low efficiency $E = 1/[0.25 \times 256 + 0.75] = 1.5\%$ is observed. This is due to the fact that only a few processors (say 4) are kept busy, while the majority of the nodes are left idling.

Scaled-Workload Speedup: To achieve higher efficiency in using a large cluster, we must consider scaling the problem size to match with the cluster capability. This leads to the following speedup law proposed by John Gustafson (1988). Let W be the workload in a given program. When we use an n-processor system, we scale the workload to $W' = \alpha W + (1-\alpha) nW$. Note that only the parallelizable portion of the workload is scaled n times in the second term. This scaled workload W' is essentially the sequential execution time on a single processor. The parallel execution time of W' workload on n processors is kept at the level of the original workload W. Thus, a *scaled-workload speedup* is defined as follows:

$$S' = W'/W = [\, \alpha W + (1-\alpha) nW \,] / W = \alpha + (1-\alpha) n \qquad (1.3)$$

This speedup is known as Gustafson's Law. By fixing the parallel execution time at level W, we achieve the following efficiency expression:

$$E' = S' / n = \alpha/n + (1-\alpha) \qquad (1.4)$$

For the above program with a scaled workload, we can improve the efficiency of using a 256-node cluster to $E' = 0.25/256 + 0.75 = 0.751$. We shall apply either the Amdahl's Law or Gustafson's Law under different workload conditions. For fixed workload, we apply Amdahl's law. To solve scaled problems, we apply Gustafson's Law.

1.4.2 System Availability and Application Flexibility

In addition to performance, system availability and application flexibility are two other most important design goals in a distributed computing system. We check these related two concerns, separately.

System Availability: *High availability* (HA) is desired in all clusters, grids, P2P, and cloud systems. A system is highly available if it has *long mean time to failure* (MTTF) and short *mean time to repair* (MTTR).

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

The *system availability* is formally defined as flows:

$$\textit{System Availability} = \textit{MTTF} / (\textit{MTTF} + \textit{MTTR}) \quad (1.5)$$

The system availability is attributed to many factors. All hardware, software, and network components may fail. Any failure that will pull down the operation of the entire system is called a *single point of failure.* The rule of thumb is to design a dependable computing system with no single point of failure. Adding hardware redundancy, increasing component reliability, and design for testability will all help enhance the system availability and dependability.

In Fig.1.19, we estimate the effects on system availability by scaling the system size in term of the number of processor cores in a system. In general, as a distributed system increases in size, the availability decrease due to higher chance of failure and difficulty to isolate the failures. Both SMP and MPP are most vulnerable under the mangement of a single OS. Increasing system size will result in higher chance to break down. The NUMA machine has limited improvement in availability from an SMP due to use of multiple system managements.

Most clusters are designed to have high-availability (HA) with failover capability, even as the cluster gets much bigger. Vrtualized clouds form a subclass of the hosting server clusters at various datacenters. Hence a cloud has an estimated availability similar to that of the hosting cluster. A grid is visualized as a hierarchical cluster of clusters. They have even higher availability due to the isolation of faults. Therefore, clusters, clouds, and grids have a decreasing availability as system gets larger. A P2P file-sharing network hass the highest aggregation of client machines. However, they operate essentially independently with low availability even many peer nodes depart or fail simultaneously.

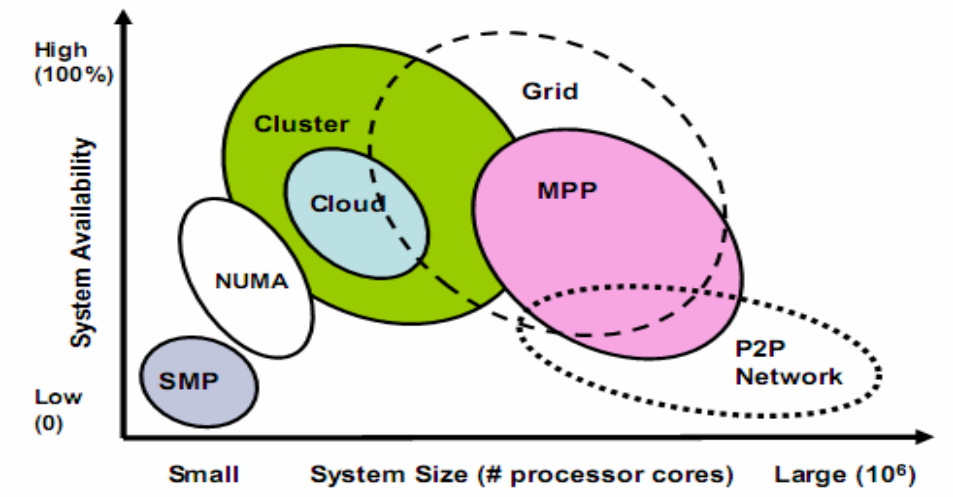

Figure 1.19 Estimated effects on the system availability by the size of clusters, MPP, Grids, P2P file-sharing networks, and computing clouds. (The estimate is based on reported experiences in hardware, OS, storage, network, and packaging technologies in available system configurations in 2010.)

1.4.3 Security Threats and Defense Technologies

Clusters, Grids, P2P, and Clouds all demand security and copyright protection. These are crucial to their acceptance by a digital society. In this section, we introduce the system vulnerability, network threats, defense countermeasures, and copyright protection in distributed or cloud computing systems.

Threats To Systems and Networks : Network viruses have threatened many users in widespread attacks

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

constantly. These incidents created worm epidemic by pulling down many routers and servers. These attacks had caused billions of dollars loss in business, government, and services. Various attack types and the potential damages to users are summarized in Fig.1.20. Information leakage leads to the loss of confidentiality. Loss of data integrity may be caused by user alteration, Trojan horse, and service spoofing attacks. The *denial of service* (DoS) result in loss of system operation and Internet connections.

Lack of authentication or authorization lead to illegitimate use of computing resources by attackers. Open resources like datacenters, P2P networks, grid and cloud infrastructures could well become the next targets. We need to protect clusters, grids, clouds, and P2P systems. Otherwise, no users dare to use or trust them for outsourced work. Malicious intrusions to these systems may destroy valuable hosts, network, and storage resources. Internet anomalies found in routers, gateways, and distributed hosts may hinder the acceptance of these public-resource computing services.

Security Responsibilities: We identify three security requirements: *confidentiality, integrity, and availability* for most internet service providers and cloud users. As shown in Fig.1.21, in the order of SaaS, PaaS, and IaaS, the providers gradually release the responsibilities of security control to the cloud users. In summary, the SaaS model relies on the cloud provider to perform all security functions. On the other extreme, the IaaS model wants the users to assume almost all security functions except leaving the availability to the hands of the providers. The PaaS model relies on the provider to maintain data integrity and availability, but burdens the user with confidentiality and privacy control.

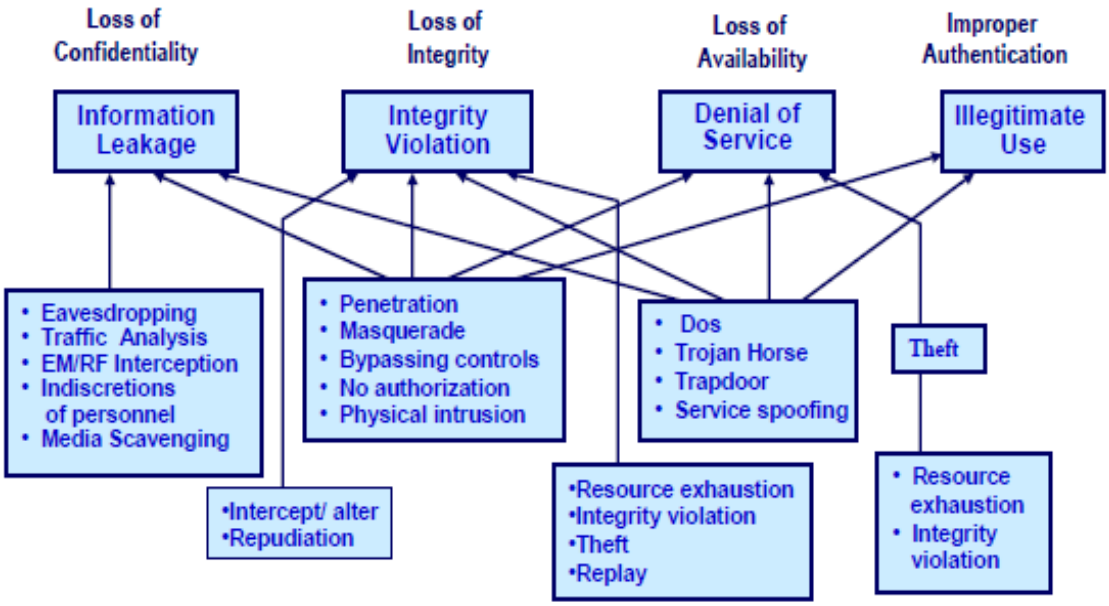

Figure 1.20 Various system attacks and network threats to cyberspace.

System Defense Technologies: Three generations of network defense technologies have appeared in the past. In the first generation, tools were designed to prevent or avoid intrusions. These tools usually manifested as access control policies or tokens, cryptographic systems, etc. However, the intruder can always penetrate a secure system because there is always a weakest link in the security provisioning process. The second generation detects intrusions timely to exercise remedial actions. These techniques include firewalls, *Intrusion Detection Systems* (IDS), PKI service, reputation systems, etc. The third generation provides more intelligent responses to intrusions.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

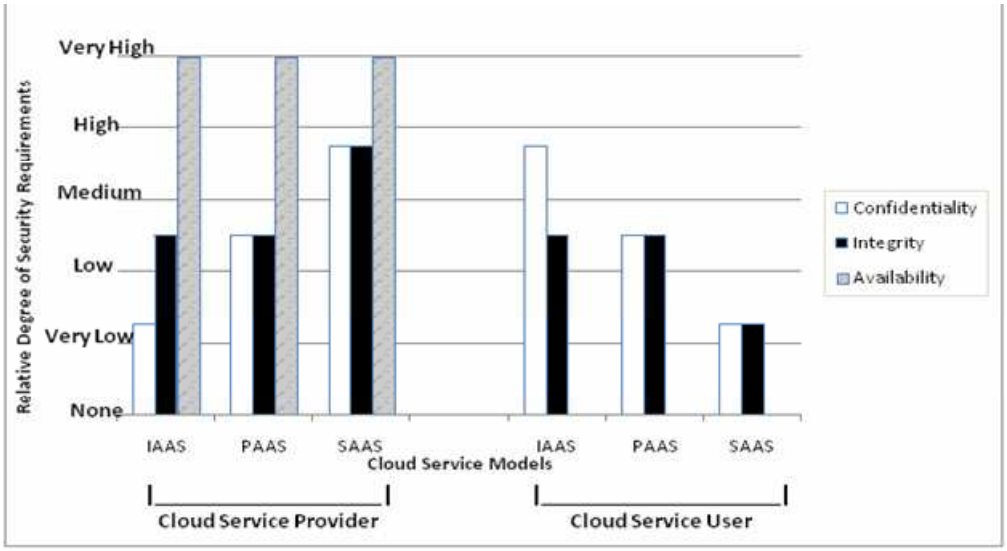

Figure 1.21: Internet security responsibilities by cloud service providers and by the user mass.

Copyright Protection: Collusive piracy is the main source of intellectual property violations within the boundary of a P2P network. Paid clients (colluders) may illegally share copyrighted content files with unpaid clients (pirates). On-line piracy has hindered the use of open P2P networks for commercial content delivery. One can develop a proactive content poisoning scheme to stop colluders and pirates from alleged copyright infringements in P2P file sharing. Pirates are detected timely with identity-based signatures and time-stamped tokens. The scheme stops collusive piracy without hurting legitimate P2P clients. We will cover grid security, P2P reputation systems, and copyright-protection issues in Chapters 5 and 7.

Data Protection Infrastructure: Security infrastructure is needed to support safeguard web and cloud services. At the user level, we need to perform trust negotiation and reputation aggregation over all users. At the application end, we need to establish security precautions in worm containment and intrusion detection against virus, worm, and DDoS attacks. We need also deploy mechanism to prevent on-line piracy and copyright violations of digital contents. In Chapter 6, we will study reputation system for protecting distributed systems and datacenters.

1.4.4 Energy-Efficiency in Distributed Computing

Primary performance goals in conventional parallel and distributed computing systems are high performance and high throughput, considering some form of performance reliability, e.g., fault tolerance and security. However, these systems recently encounter new challenging issues including energy efficiency, and workload and resource outsourcing. These emerging issues are crucial not only in their own, but also for the sustainability of large-scale computing systems in general. In this section, we review energy consumption issues in servers and HPC systems. The issue of workload and resource outsourcing for cloud computing is discussed. Then we introduce the protection issues of datacenters and explore solutions.

The energy consumption in parallel and distributed computing systems raises various monetary, environmental and system performance issues. For example, Earth Simulator and Petaflop are two example systems with 12 and 100 megawatts of peak power, respectively. With an approximate price of 100 dollars per megawatt, their energy costs during peak operation times are 1,200 and 10,000 dollars per hour; this is

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

beyond the acceptable budget of many (potential) system operators. In addition to power cost, cooling is another issue that must be addressed due to negative effects of high temperature on electronic components. The rising temperature of a circuit not only derails the circuit from its normal range but also results in decreasing the lifetime of its components.

Energy consumption of unused servers: To run a server farm (data center) a company has to spend a huge amount of money for hardware, software (software licences), operational support and energy every year. Therefore, the company should thoroughly identify weather the installed server farm (more specifically, the volume of provisioned resources) is at an appropriate level in terms particularly of utilization. Some analysts estimate that on average around one-sixth (15%) of the full-time servers in a company is left powered on without being actively used (i.e., idling) on a daily basis. This indicates that with 44 million servers in the world, around 4.7 million servers are not doing any useful work.

The potential savings by turning off these servers are large, globally $3.8 billion in energy costs alone and $24.7 billion in the total cost of running non-productive servers according to a study by 1E Company in partnership with the *Alliance to Save Energy* (ASE). With the respect to environment, this amount of energy wasting is equal to entering 11.8 million tons of carbon dioxide per year which is equivalent to the CO^2 pollution of 2.1 million cars. This ratio in the U.S comes to 3.17 million tons of carbon dioxide, or 580,678 cars. Therefore, the first step in IT departments is to analyze their servers to find out unused and/or underutilized servers.

Reducing energy in active servers: In addition to the identification of unused/under-utilized servers for energy savings, the application of appropriate techniques to decrease energy consumption in active distributed systems with negligible influence on their performance is necessary. Power management issue in distributed computing platforms can be categorized into four layers (Fig.1.22): application layer, middleware layer, resource layer and network layer.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

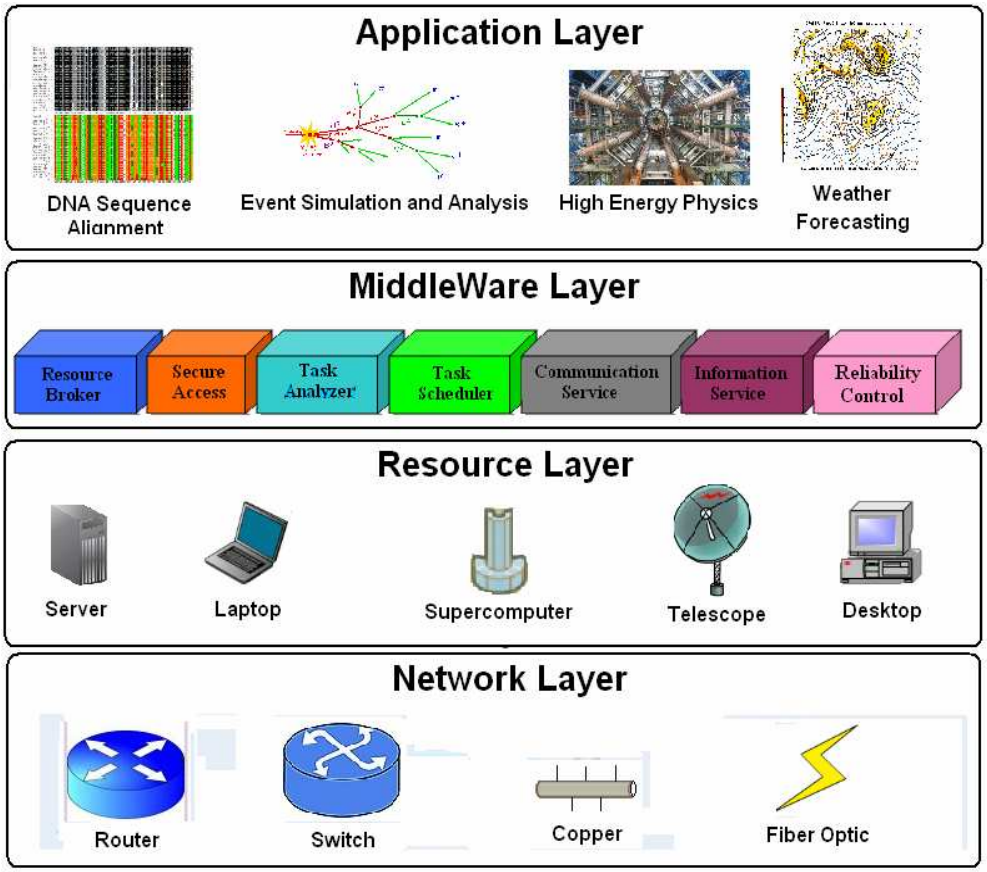

Figure 1.22 Four operational layers of distributed computing systems

Application layer: Until now, most user applications in science, business, engineering, and financial areas, tend to increase the speed or quality performance. By introducing energy-aware applications, the challenge is how to design sophisticated multilevel and multi-domain energy management applications without hurting performance. The first step is to explore a relationship between performance and energy consumption. Indeed, the energy consumption of an application has a strong dependency with the number of instructions needed to execute the application and the number of transactions with storage unit (or memory). As well these two factors (computation and storage) are correlated and they affect application completion time.

Middleware layer: The middleware layer acts as a bridge between the application layer and the resource layer. This layer provides resource broker, communication service, task analyzer, task scheduler, security access, reliability control and information service. This layer is susceptible for applying energy-efficient techniques particularly in task scheduling. Until recently, scheduling is aimed to minimize a cost function generally the makespan, i.e., the whole execution time of a set of tasks. Distributed computing systems necessitates a new cost function covering both makespan and energy consumption.

Resource layer: The resource layer consists of a wide range of resources including computing nodes and storage units. This layer generally interacts with hardware devices and also operating system; and therefore it is responsible for controlling all distributed resources in distributed computing systems. In the recent past, several mechanisms have been developed for more efficient power management of hardware and operating systems. The majority of them are hardware approaches particularly for processors. *Dynamic power*

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

management (DPM) and *dynamic voltage-frequency scaling* (DVFS) are two popular methods incorporated in recent computer hardware systems In DPM, hardware devices, such as CPU have the capability to switch from idle mode to one or more lower-power modes. In DVFS, energy savings are achieved on the fact that the power consumption in CMOS circuits has the direct relationship with frequency and the square of voltage supply. In this case, the execution time and power consumption are controllable by switching between different frequencies and voltages. Figure 1.23 shows the principle of the DVFS method. This method enables the exploitation of the slack time (idle time) typically incurred by inter-task relationships (e.g., precedence constraints) [24]. Specifically, the slack time associated with a task is utilized to execute the task in a lower voltage-frequency. The relationship between energy and voltage-frequency in CMOS circuits is related by the following expression:

$$\begin{cases} E = C_{eff} f v^2 t \\ f = K \dfrac{(v - v_t)^2}{v} \end{cases} \quad \textbf{(1.6)}$$

where v, $C_{eff,}$ K, and v_t are the voltage, circuit switching capacity, a technology dependent factor, and threshold voltage, respectively. The parameter t is the execution time of the task under clock frequency f.

By reducing voltage and frequency the energy consumption of device can be reduced. However, both DPM and DVFS techniques may cause some negative effects on power consumption of a device in both active and idle, and create a transition overload for switching between states or voltage/frequencies. Transition overload is especially important in DPM technique: if the transition latencies between lower-power modes are assumed to be negligible, then energy can be saved by simply switching between these modes. However, this assumption is rarely valuable and therefore switching between low-power modes affects performance.

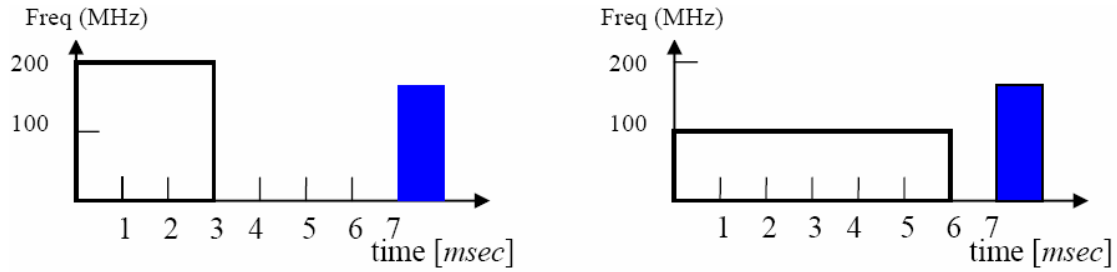

Figure 1.23 DVFS technique (right) original task (left) voltage-frequency scaled task (Courtesy of R.Ge, et al, "Performance Constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters", *Proc. of ACM Supercomputing Conf.*, Wash. DC, 2005 [16].)

Another important issue in the resource layer is in the storage area. Storage units interact with the computing nodes greatly. This huge amount of interactions keeps the storage units always active. This results in large energy consumption. Storage devices spend about 27% of the total energy consumption in a data center. What is even worse is this figure increases rapidly due to 60% increase in storage need annually.

Network layer: Routing and transferring packets and enabling network services to the resource layer are the main responsibility of the network layer in distributed computing systems. The major challenge to build energy-efficient networks is again how to measure, predict and make balance between energy consumption and performance. Two major challenges to design energy-efficient networks are identified below:

- The models should represent the networks comprehensively as they should give a full understanding

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

of interactions between time, space and energy.

- New energy-efficient routing algorithms need to be developed. New energy-efficient protocols should be developed against network attacks.

As information resources drive economic and social development, datacenters become increasingly important as where the information items are stored, processed, and services provided. Datacenters becomes another core infrastructure just like power grid and transportation systems. Traditional datacenter suffers from high construction and operational cost, complex resource management and poor usability, low security and reliability, and huge energy consumption etc. It is necessary to adopt new technologies in next generation datacenter designs as studies in Chapter 7.

1.5 References and Homework Problems

In the past 4 decades, parallel processing and distributed computing have been hot topics for research and development. Earlier work in this area were treated in several classic books [1, 11, 20, 21]. More recent coverage can be found in newer books [6, 13, 14, 16, 18, 26] published beyond 2000. Cluster computing was covered in [21, 27] and grid computing in [3, 4, 14, 34]. P2P networks are introduced in [13, 33]. Cloud computing is studied in [7-10, 15, 19, 22, 23, 31]. Virtualization techniques are treated in [28-30]. Distributed algorithms and parallel programming are studied in [2, 12, 18, 21, 25]. Distributed operating systems and software tools are covered in [5, 32]. Energy efficiency and power management are studied in [17, 24, 35]. Clusters serve as the foundation of distributed and cloud computing. All of these topics will be studied in more details in subsequent chapters.

References

[1] G. Almasi and A. Gottlieb, *Highly Parallel Computing*, Banjamin-Cummins Publisher, 1989.

[2] G. Andrewa, *Foundations of multithreaded, Parallel and Distributed Programming*, Addison-Wesley, 2000.

[3] G. Bell, J. Gray. And A. Szalay, "Petascale Computational Systems : Balanced Cyberstructure in a Data-Centric World", *IEEE Computer Magazine*, 2006

[4] F. Berman, G. Fox, and T. Hey (Editors), *Grid Computing*, Wiley and Sons, 2003, ISBN: 0-470-85319-0

[5] M. Bever, et al, "Distributed Systems, OSF DCE, and Beyond", in *DCE-The OSF Distributed Computing Environment*, A. Schill (edtor), Belin, Springer-Verlag, pp. 1-20, 1993

[6] K. Birman, *Reliable Distributed Systems: Technologes, Web Services, and Applications*, Springer Verlag 2005.

[7] G. Boss, et al, "Cloud Computing-The BlueCloud Project ", www.ibm.com/developerworks/ websphere/zones/hipods/ Oct. 2007

[8] R. Buyya, C. Yeo; and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," *10th IEEE Int'l Conf. on High Perf. Computing and Comm.*, Sept. 2008

[9] F. Chang, et al., "Bigtable: A Distributed Storage System for Structured Data", OSDI 2006.

[10] T. Chou, *Introduction to Cloud Computing : Business and Technology*, Lecture Notes at Stanford University and at Tsinghua University, Active Book Press, 2010.

[11] D. Culler, J. Singh, and A. Gupta, *Parallel Computer Architecture*, Kaufmann Publishers,

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

1999.

[12] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *Proc. of OSDI 2004*.

[13] J. Dillimore, T. Kindberg, and G. Coulouris, *Distributed Systems: Concepts and Design,* (4th Edition), Addison Wesley, May 2005, ISBN -10-03-2126-3545.

[14] J. Dongarra, et al, (editors), *Source Book of Parallel Computing*, Morgan Kaufmann, 2003.

[15] I. Foster, Y. Zhao, J.Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," *Grid Computing Environments Workshop*, 12-16 Nov. 2008.

[16] V. K. Garg, *Elements of Distributed Computing*, Wiley-IEEE Press, 2002.

[17] R.Ge, X. .Feng, and K.W.Cameron, "Performance constrained distributed DVS scheduling for scientific applications on power-aware clusters", *Proc. Supercomputing Conf.*, Wash. DC, 2005.

[18] S. Ghosh, *Distributed Systems- An Algorithmic Approach*, Chapman & Hiall/CRC, 2007.

[19] Google, Inc. " Google and the Wisdom of Clouds", http://www.businessweek.com/ magazine/content/ 0752/ b4064048925836.htm

[20] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programming*, McGraw-Hill, 1993.,

[21] K. Hwang and Z. Xu: *Scalable Parallel Computing*, McGraw-Hill, 1998.

[22] K. Hwang, S. Kulkarni, and Y. Hu, "Cloud Security with Virtualized Defense and Reputation-based Trust Management", *IEEE Conf. Dependable, Autonomous, and Secure Computing* (DAC-2009), Chengdu, China, Dec.14, 2009

[23] K. Hwang and D. Li, " Security and Data Protection for Trusted Cloud Computing", *IEEE Internet Computing*, September. 2010.

[24] Kelton Research, "1E / Alliance to Save Energy Server Energy & Efficiency Report", http://www.1e.com/ EnergyCampaign/downloads/Server_Energy_and_Efficiency_Report_2009.pdf , Sept. 2009.

[25] Y. C. Lee and A. Y. Zomaya, "A Novel State Transition Method for Metaheuristic-Based Scheduling in Heterogeneous Computing Systems," *IEEE Trans. Parallel and Distributed Systems*, Sept. 2008.

[26] D. Peleg, *Distributed Computing : A Locality-Sensitive Approach*, SIAM Publisher, 2000.

[27] G.F. Pfister, *In Serach of Clusters*, (second Edition), Prentice-Hall, 2001

[28] M. Rosenblum and T. Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends", *IEEE Computer*, May 2005, pp.39-47.

[29] M. Rosenblum, "Recent Advances in Virtual Machines and Operating Systems", Keynote Address, *ACM ASPLOS* 2006

[30] J. Smith and R. Nair, *Virtual Machines*, Morgan Kaufmann , 2005

[31] B. Sotomayor, R. Montero, and I. Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds", *IEEE Internet Computing*, Sept. 2009

[32] A. Tannenbaum, *Distributed Operating Systems*, Prentice-Hall, 1995.

[33] I. Taylor, *From P2P to Web Services and Grids* , Springer-Verlag, London, 2005.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

[34] M. Waldrop, "Grid Computing", IEEE *Computer Magazine*, 2000

[35] Z. Zong, "Energy-Efficient Resource Management for High-Performance Computing Platforms", *PhD Dissertation*, Auburn University, August 9, 2008

Homework Problems

Problem 1.1: Map ten abbreviated terms and system models on the left with the best-match descriptions on the right. Just enter the description label (**a, b, c, …,j**) in the underlined blanks in front of the terms.

_____ **Globus**	**(a)**	A scalable software platform promoted by Apache for web users to write and run applications over vast amounts of distributed data.
_____ **BitTorrent**	**(b)**	A P2P network for MP3 music delivery using a centralized directory server
_____ **Gnutella**	**(c)**	The programming model and associated implementation by Google for distributed mapping and reduction of very large data sets
_____ **EC2**	**(d)**	A middleware library jointly developed by USC/ISI and Argonne National Lab. for Grid resource management and job scheduling
_____ **TeraGrid**	**(e)**	A distributed storage program by Google for managing structured data that can scale to very large size.
_____ **EGEE**	**(f)**	A P2P file-sharing network using multiple file index trackers
_____**Hadoop**	**(g)**	A critical design goal of cluster of computers to tolerate nodal faults or recovery from host failures.
_____ **SETI@home**	**(h)**	The service architecture specification as an open Grid standard
_____ **Napster**	**(i)**	An elastic and flexible computing environment that allows web application developers to acquire cloud resources effectively
_____ **Bigtable**	**(j)**	A P2P Grid over 3 millions of desktops for distributed signal processing in search of extra-terrestrial intelligence

Problem 1.2: Circle only one correct answer in each of the following questions.

(1) In today's Top 500 list of the fastest computing systems, which architecture class dominates the population ?

- **a.** Symmetric shared-memory multiprocessor systems
- **b.** Centralized massively parallel processor (MPP) systems.
- **c.** Clusters of cooperative computers.

(2) Which of the following software packages is particularly designed as a distributed storage management system of scalable datasets over Internet clouds?

- **a.** MapReduce
- **b.** Hadoop
- **c.** Bigtable

(3) Which global network system was best designed to eliminate isolated resource islands ?

- **a.** The Internet for computer-to-computer interaction using Telnet command
- **b.** The Web service for page-to-page visits using http:// command
- **c.** The Grid service using midleware to establish interactions between applications running on a federation of cooperative machines.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

(4) Which of the following software tools is specifically designed for scalable storage services in distributed cloud computing applications ?

- **a.** Amazon EC2
- **b.** Amazon S3
- **c.** Apache Hadoop library

(5) In a cloud formed by a cluster of servers, all servers must be select as follows:

- **a.** All cloud machines must be bulit on physical servers
- **b.** All cloud machines must be built with virtual servers
- **c.** The cloud machines can be either physical or virtual servers.

Problem 1.3: Content delivery networks have gone through three generations of development: namely the client-server architecture, massive network of content servers, and P2P networks. Discuss the advantages and shortcomings of using these content delivery networks.

Problem 1.4: Conduct a deeper study of the three cloud platform models presented in Table 1.6. Compare their advantages and shortcomings in development of distributed applications on each cloud platform. The material in Table 1.7 and Table 1.8 are useful in your assessment.

Problem 1.5: Consider parallel execution of an MPI-coded C program in SPMD (single program and multiple data streams) mode on a server cluster consisting of n identical Linux servers. SPMD mode means that the same MPI program is running simultaneously on all servers but over different data sets of identical workload. Assume that 25% of the program execution is attributed to the execution of MPI commands. For simplicity, assume that all MPI commands take the same amount of execution time. Answer the following questions using Amdahl's law:

(a) Given that the total execution time of the MPI program on a 4-server cluster is T minutes. What is the speedup factor of executing the same MPI program on a 256-server cluster, compared with using the 4-server cluster. Assume that the program execution is deadlock-free and ignore all other run-time execution overheads in the calculation.

(b). Suppose that all MPI commands are now enhanced by a factor of 2 by using active messages executed by message handlers at the user space. The enhancement can reduce the execution time of all PMI commands by half. What is the speedup of the 256-server cluster installed with this MPI enhancement, computed with the old 256-server cluster without MPI enhancement?

Problem 1.6: Consider a program to multiply two large-scale $N \times N$ matrices, where N is the matrix size. The sequential multiply time on a single sever is $T_1 = c\,N^3$ minutes, where c is a constant decided by the server used. A MPI-code parallel program requires $T_n = c\,N^3/n + d\,N^2 / n^{0.5}$ minutes to complete execution on an n-server cluster system, where d is a constant determined by the MPI version used. You can assume the program has a zero sequential bottleneck ($\alpha = 0$). The second term in T_n accounts for the total message passing overhead experienced by n servers.

Answer the following questions for a given cluster configuration with $n = 64$ servers and $c = 0.8$ and $d = 0.1$. Parts (a, b) have a fixed workload corresponding to the matrix size $N = 15{,}000$. Parts (c, d) have a scaled workload associated with an enlarged matrix size $N' = n^{1/3} N = 64^{1/3} \times 15{,}000 = 4 \times 15{,}000 = 60{,}000$. Assume the same cluster configuration to process both workloads. Thus the system parameters n, c, and d stay unchanged. Running the scaled workload, the overhead also increases with the enlarged matrix size N'.

(a) Using Amdahl's law, calculate the speedup of the n-server cluster over a single server.

(b) What is the efficiency of the cluster system used in Part (a) ?

(c) Calculate the speedup in executing the scaled workload for an enlarged $N' \times N'$ matrix

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

on the same cluster configuration using Gustafson Law.

(d) Calculate the efficiency of running the scaled workload in Part (c**)** on the 64-processor cluster.

(e) Compare the above speedup and efficiency results and comment on their implications.

Problem 1.7: Cloud computing is an emerging distributed computing paradigm. An increasing number of organizations in industry and business sectors adopt cloud systems as their system of choice. Answer the following questions on cloud computing.

(a) List and describe main characteristics of cloud computing systems.

(b) Discuss key enabling technologies in cloud computing systems.

(c) Discuss different ways for cloud service providers to maximize their revenue.

Problem 1.8: Compare the similarities and differences between traditional computing clusters/grids and the computing clouds launched in recent years. You should consider all technical and economic aspects as listed below. Answer the following questions against real example systems or platforms built in recent years. Also discuss the possible convergence of the two computing paradigms in the future..

(a) Hardware, software, and networking support

(b) Resource allocation and provisioning methods

(c) Infrastructure management and protection.

(d) Supporting of utility computing services

(e) Operational and cost models applied.

Problem 1.9: Answer the following questions on *personal computing* (PC) and *high-performance computing* (HPC) systems:

(a) Explain why the changes in *personal computing* (PC) and *high-performance computing* (HPC) were evolutionary rather revolutionary in the past 30 years.

(b) Discuss the drawbacks in disruptive changes in processor architecture. Why memory wall is a major problem in achieving scalable performance?

(c) Explain why x-86 processors are still dominating the PC and HPC markets ?

Problem 1.10: Multi-core and many-core processors have appeared in widespread use in both desktop computers and HPC systems. Answer the following questions in using advanced processors, memory devices, and system interconnects.

(a) What are differences between multi-core CPU and GPU in architecture and usages ?

(b) Explain why parallel programming cannot match with the progress of processor technology.

(c) Suggest ideas and defend your argument by some plausible solutions to this mismatch problem between core scaling and effective programming and use of multicores.

(d) Explain why flash memory SSD can deliver better speedups in some HPC or HTC applications.

(e) Justify the prediction that Infiniband and Ehternet will continue dominating the HPC market.

Problem 1.11 Compare the HPC and HTC computing paradigms and systems. Discuss their commonality and differences in hardware and software support and application domains.

Problem 1.12 Answer the roles of multicore processors, memory chips, solid-state drives, and disk arrays. in building current and future distributed and cloud computing systems.

Problem 1.13 What are lopment trends of operating systems and programming paradigms in modern

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

distributed systems and cloud computing platforms ?

Problem 1.14 Distinguish P2P networks from Grids and P2P Grids by filling the missing table entries. Some entries are already given. You need to study the entries in Table 1.3 , Table 1.5, and Table 1.9 before you try to distinguish these systems precisely.

Discuss the major advantages and disadvantages in the following challenge areas:

(a) Why virtual machines and virtual clusters are suggested in cloud computing systems ?

(b) What are the breakthrough areas needed to build virtualized cloud systems cost effectively ?

(c) What is your observations of the impact of cloud platforms on the future of HPC industry ?

Problem 1.16: Briefly explain each of the following cloud computing services. Identify two clouder providers in each service category.

(a) Application cloud services

(b) Platform cloud services

(c) Compute and storage services

(d). Co-location cloud services

(e). Network cloud services.

Table 1.11 Comparison among P2P Networks, Grids, and P2P Grids

Features	P2P Networks	Grid Systems	P2P Grids
Applications and Peer or Node Roles	Distributed file sharing, content distribution, peer machines acting as both clients and servers		
System Control and Service Model			Policy-based control in a grid infrastructure, all services from clent machines
System Connectivity		Static conections with high-speek links over grid resource sites	
Resource Discovery and Job Management	Autonomous peers without discovery, no use of a central job scheduler		
Repersentative Systems		NSF TeraGrid, UK EGGE Grid, China Grid	

Problem 1.15: plain the impacts of machine virtualization to business computing and HPC systems.

Problem 1.17: **Br**iefly explain the following terms associated with network threats or security defense in a distributed computing system:

(a) Denial of service (DoS)

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.

(b) Trojan horse
(c) Network worms
(d) Masquerade
(e) Evasdropping
(f) Service sproofing
(g) Authorization
(h) Authentication
(i) Data integrity
(j) Confidentaility

Problem 1.18: Briefly answer following questions on green information technology and energy efficiency in distributed systems. You can find answers in later chapters or search over the Web.

(a) Why power consumption is critical to datacenter operations ?
(b) Justify Equation (1.6) by reading a cited information source.
(c) What is dynamic voltage frequency scaling (DVFS) technique ?

Problem Problem 1.19: Distinguish the following terminologies associate with multithreaded processor architecture:

(a) What is fine-grain multithreading architecture ? Identify two example processors.
(b) What is course-grain multithreading architecture ? Identify two example processors.
(c) What is simultaneously multithreading (SMT) architecture ? Identify two example proccesors.

Problem 1.20: Characterize the following three cloud computing models:

(a) What is an IaaS (Infrastructure as a Service) cloud ? Give one example system.
(b) What is a PaaS (Platform as a Service) cloud ? Give one example system.
(c) What is a SaaS (Sofftware as a Service) cloud ? Give one example system.

Distributed Computing : Clusters, Grids and Clouds, All rights reserved
by Kai Hwang, Geoffrey Fox, and Jack Dongarra, May 2, 2010.