UNIT I: Systems modeling, Clustering and virtualization:
Scalable Computing over the Internet, Technologies for Network based systems, System models for Distributed and Cloud Computing, Software environments for distributed systems and clouds, Performance, Security And Energy Efficiency.

1. SCALABLE COMPUTING OVER THE INTERNET:

→Scalability is the ability of a computer application or product (hardware or software) to continue to function well when it is changed in **size (or) volume** in order to use a user need.
→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.
→Thus, distributed computing becomes data-intensive and network-centric.

1.1 The Age of Internet Computing.
1.2 Scalable Computing Trends and New Paradigms.
1.3 The Internet of Things and Cyber-Physical Systems

1.1. The Age of Internet Computing:

Computer technology has gone through five generations of development

1950 to 1970: Mainframes, including the IBM 360 and CDC 6400, were built to satisfy the demands of large businesses and government organizations.

1960 to 1980: lower-cost mini computers such as the DEC PDP 11 and VAX Series became popular among small businesses and on college campuses

1970 to 1990: widespread use of personal computers built with VLSI microprocessors

1980 to 2000: Massive numbers of portable computers and pervasive devices appeared in both wired and wireless applications

Since1990: The use of both HPC and HTC systems hidden in clusters, grids, or Internet clouds has proliferated.**HTC and HPC are Computational Grids**

The emergence of computing clouds instead demands high-throughput computing (HTC) systems built with parallel and distributed computing technologies.

The Platform Evolution:

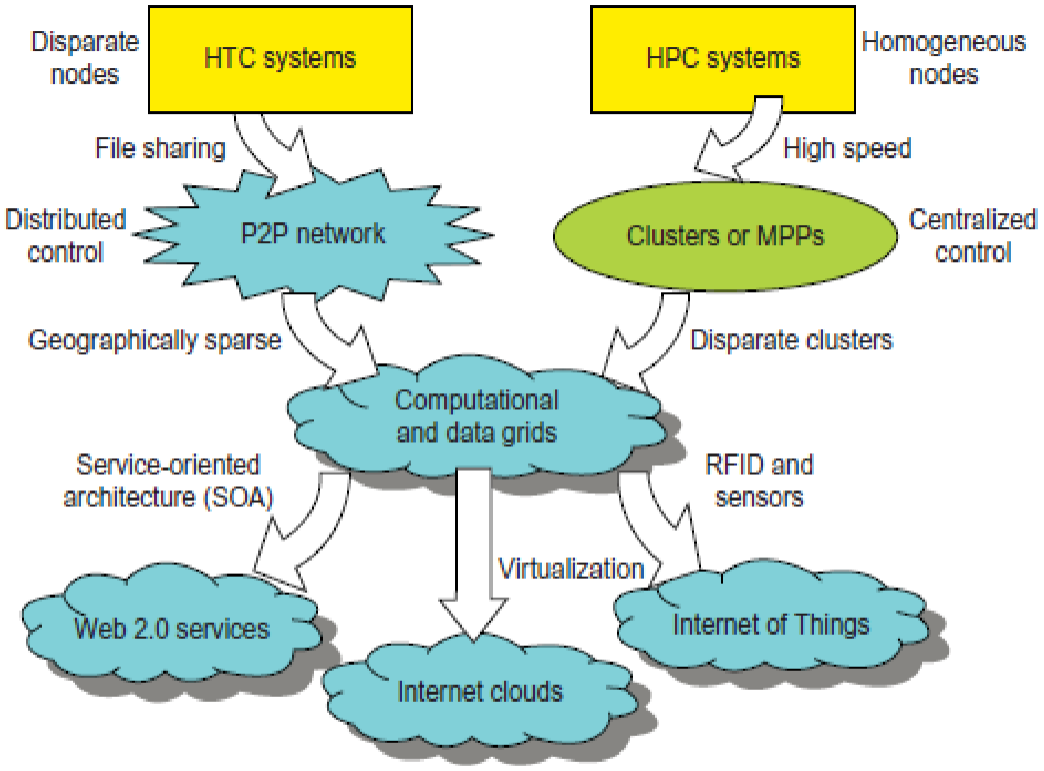

FIGURE 1.1

Evolutionary trend toward parallel, distributed, and cloud computing with clusters, MPPs, P2P networks, grids, clouds, web services, and the Internet of Things.

→The general computing trend is to leverage shared web resources and massive amounts of data over the Internet.

→**Figure 1.1** illustrates the evolution of HPC and HTC systems. On the HPC side, supercomputers (massively parallel processors or MPPs) are gradually replaced by clusters of cooperative computers out of a desire to share computing resources.

High-Performance Computing:

For many years, HPC systems emphasize the raw speed performance. The speed of HPC systems has increased from Gflops in the early 1990s to now Pflops in 2010.

This improvement was driven mainly by the demands from scientific, engineering, and manufacturing communities. For example, the Top 500 most powerful computer systems in the world are measured by floating-point speed in Linpack benchmark results. However, the number of supercomputer users is limited to less than 10% of all computer users. Today, the majority of computer users are using desktop computers or large servers when they conduct Internet searches and market-driven computing tasks.

High-Throughput Computing:

The development of market-oriented high-end computing systems is undergoing a strategic change from an HPC paradigm to an HTC paradigm.

This HTC paradigm pays more attention to high-flux computing. The main application for high-flux computing is in Internet searches and web services by millions or more users simultaneously.

The performance goal thus shifts to measure high throughput or the number of tasks completed per unit of time. HTC technology needs to not only improve in terms of batch processing speed, but also address the acute problems of cost, energy savings, security, and reliability at many data and enterprise computing centers. This book will address both HPC and HTC systems to meet the demands of all computer users.

Three New Computing Paradigms:

With the introduction of SOA, Web 2.0 services become available. Advances in virtualization make it possible to see the growth of Internet clouds as a new computing paradigm. The maturity of radio-frequency identification (RFID), Global Positioning System (GPS), and sensor technologies has triggered the development of the Internet of Things (IoT).

When the Internet was introduced in 1969, Leonard Klienrock of UCLA declared: "As of now, computer networks are still in their infancy, but as they grow up and become sophisticated, we will probably see the spread of computer utilities, which like present electric and telephone utilities, will service individual homes and offices across the country." Many people have redefined the term "computer" since that time. In 1984, John Gage of Sun Microsystems created the slogan, "The network is the computer."

In 2008, David Patterson of UC Berkeley said, "The data center is the computer. There are dramatic differences between developing software for millions to use as a service versus distributing software to run on their PCs." Recently, Rajkumar Buyya of Melbourne University simply said: "The cloud is the computer."

Computing Paradigm Distinctions:

The high-technology community has argued for many years about the precise definitions of centralized computing, parallel computing, distributed computing, and cloud computing. In general, distributed computing is the opposite of centralized computing.

The field of parallel computing overlaps with distributed computing to a great extent, and cloud computing overlaps with distributed, centralized, and parallel computing.

• Centralized computing This is a computing paradigm by which all computer resources are centralized in one physical system. All resources (processors, memory, and storage) are fully shared and tightly coupled within one integrated OS. Many data centers and supercomputers are centralized systems, but they are used in parallel, distributed, and cloud computing applications.

• In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Some authors refer to this discipline as parallel processing. Interprocessor communication is accomplished through shared memory or via message passing.

A computer system capable of parallel computing is commonly known as a parallel computer. Programs running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming.

• Distributed computing This is a field of computer science/engineering that studies distributed systems. A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network.

Information exchange in a distributed system is accomplished through message passing. A computer program that runs in a distributed system is known as a distributed program. The process of writing distributed programs is referred to as distributed programming.

• Cloud computing An Internet cloud of resources can be either a centralized or a distributed computing system. The cloud applies parallel or distributed computing, or both. Clouds can be built with physical or virtualized resources over large data centers that are centralized or distributed. Some authors consider cloud computing to be a form of utility computing or service computing.

Distributed System Families:

Since the mid-1990s, technologies for building P2P networks and networks of clusters have been consolidated into many national projects designed to establish wide area computing infrastructures, known as computational grids or data grids.

Meeting these goals requires yielding the following design objectives:

• Efficiency measures the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput, data access, storage, and power efficiency.
• Dependability measures the reliability and self-management from the chip to the system and application levels. The purpose is to provide high-throughput service with Quality of Service (QoS) assurance, even under failure conditions.
• Adaptation in the programming model measures the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workload and service models.
• Flexibility in application deployment measures the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

Difference Between High Performance Computing and High Throughput Computing:

High-Performance Computing(HPC) is a computing technique to process computational problems, and complex data and to perform scientific simulations. HPC systems consist more a number more processors or computer nodes, high-speed interconnects, and specialized libraries and tools. In order to use HPC systems effectively users must have a proper knowledge of parallel computing and optimization techniques. HPC is used in various fields such as engineering, finance, commercial applications, weather forecasting, and automotive design.

Advantages of HPC:

- **Faster computation:** HPC systems can process large and complex calculations and processes faster end efficiently than the traditional approach.
- **Scalability:** HPC systems are more scalable because they can handle many processors and computing nodes according to their applications.
- **Parallel processing:** HPC systems make use of parallel processing in order to divide the computations into smaller tasks so that they can be processed simultaneously.
- **Improved accuracy:** HPC system performs and gives more accurate simulations and calculations.

High Throughput Computing (HTC) is defined as a type of computing that aims to run a large number of computational tasks using resources in parallel. HTC systems consist of a distributed network of computers known as computing clusters. These systems are used to schedule a large number of jobs effectively. HTC majorly focuses on increasing the overall throughput of the system by running many smaller size tasks parallelly. HTC is commonly used in scientific research and engineering applications in order to process large data sets or perform simulations that require extensive computational power.

Advantages of HTC:

- **Flexibility:** HTC is more flexible and can be used for many computing tasks related to business analytics and scientific research.
- **Cost-Effectiveness:** HTC is more cost-effective as compared to the solutions offered by High-Performance Computing(HTC) as it makes use of hardware and software that is available and less expensive and performs more tasks.
- **Reliability:** HTC systems are mostly designed to provide high reliability and make sure that all tasks run efficiently even if any one of the individual components fails.
- **Resource Optimization:** HTC also does proper resource allocation by ensuring that all the resources that are available are efficiently used and accordingly increases the value of computing resources that are available.

Difference between HPC and HTC:

Parameter	HPC	HTC
Stands for	HPC stands for High-Performance Computing	HTC stands for High Throughput Computing
Definition	HPC is defined as the type of computing that makes use of multiple computer processors in order to perform complex computations parallelly.	HTC is defined as a type of computing that parallelly executes a large number of simple and computationally independent tasks.
Workload	HPC consists of running large-scale, complex, and computationally intensive applications that need significant resources and memory.	HPC consists of running a large number of tasks that are independent and small in size and does not require a large amount of memory and resources.
Processing Power	It is designed to provide maximum performance and speed for large tasks.	HTC is designed to increase the number of tasks that needs to be completed in a given specific amount of time.
Resource Management	For resource management to processes, HPC makes use of job schedulers and resource managers.	For the resource management to processes, HTC makes use of distributed management resources.
Fault Tolerance	To reduce the risk of data loss and data corruption HPC systems have a complex fault tolerance mechanism.	HTC systems do not affect any other running processes due to the failure of an individual task.
Scaling	HPC scales up when few users are running together.	HTC systems scale horizontally for simple tasks and require less computational speed.
Applications	HPC can be used in applications such as engineering design, weather forecasting, drug discovery etc.	HTC can be used in applications such as bioinformatics, research applications, etc.

1.2. Scalable Computing Trends and New Paradigms:

Degrees of Parallelism:

Fifty years ago, when hardware was bulky and expensive, most computers were designed in a bit-serial fashion. In this scenario, bit-level parallelism (BLP) converts bit-serial processing to word-level processing gradually. Over the years, users graduated from 4-bit microprocessors to 8-,16-, 32-, and 64-bit CPUs. This led us to the next wave of improvement, known as instruction-level parallelism (ILP), in which the

processor executes multiple instructions simultaneously rather than only one instruction at a time.

For the past 30 years, we have practiced ILP through pipelining, superscalar computing, VLIW (very long instruction word) architectures, and multithreading. ILP requires branch prediction, dynamic scheduling, speculation, and compiler support to work efficiently. Data-level parallelism (DLP) was made popular through SIMD (single instruction, multiple data) and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly.

Ever since the introduction of multicore processors and chip multiprocessors (CMPs), we have been exploring task-level parallelism (TLP). A modern processor explores all of the aforementioned parallelism types. In fact, BLP, ILP, and DLP are well supported by advances in hardware and compilers. However, TLP is far from being very successful due to difficulty in programming and compilation of code for efficient execution on multicore CMPs.

Innovative Applications:

→Both HPC and HTC systems desire transparency in many application aspects. For example, data access, resource allocation, process location, concurrency in execution, job replication, and failure recovery should be made transparent to both users and system management.

→Table 1.1 highlights a few key applications that have driven the development of parallel and distributed systems over the years. These applications spread across many important domains in science, engineering, business, education, health care, traffic control, Internet and web services, military, and government applications.

Table 1.1 Applications of High-Performance and High-Throughput Systems

Domain	Specific Applications
Science and engineering	Scientific simulations, genomic analysis, etc.
	Earthquake prediction, global warming, weather forecasting, etc.
Business, education, services 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 applications	Internet search, data centers, decision-making systems, etc.
	Traffic monitoring, worm containment, cyber security, etc.
	Digital government, online tax return processing, social networking, etc.
Mission-critical applications	Military command and control, intelligent systems, crisis management, etc.

The Trend toward Utility Computing:

Figure 1.2 identifies major computing paradigms to facilitate the study of distributed systems and their applications.

Utility computing focuses on a business model in 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, users demand new network efficient processors, scalable memory and storage schemes, distributed OSes, middleware for machine virtualization, new programming

models, effective resource management, and application program development. These hardware and software supports are necessary to build distributed systems that explore massive parallelism at all processing levels.

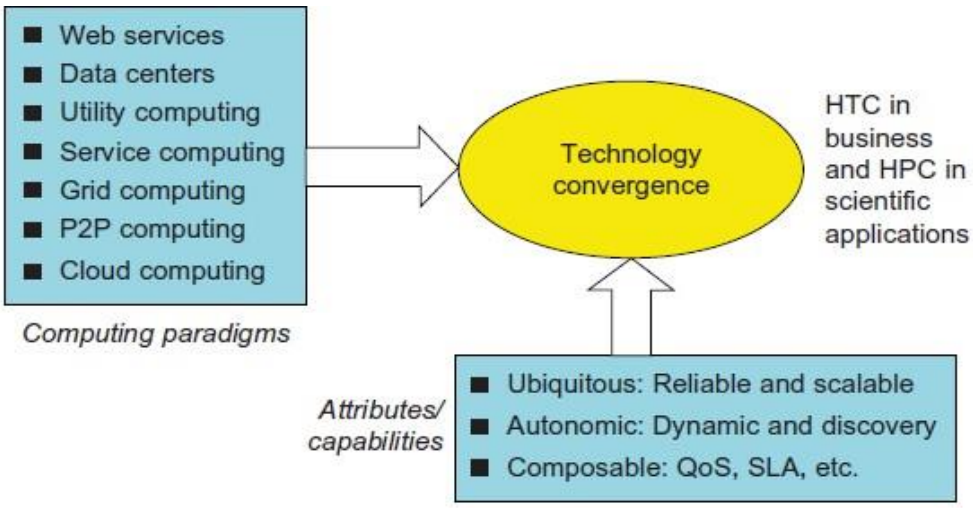

FIGURE 1.2

The vision of computer utilities in modern distributed computing systems.

Hype Cycle of New Technologies:

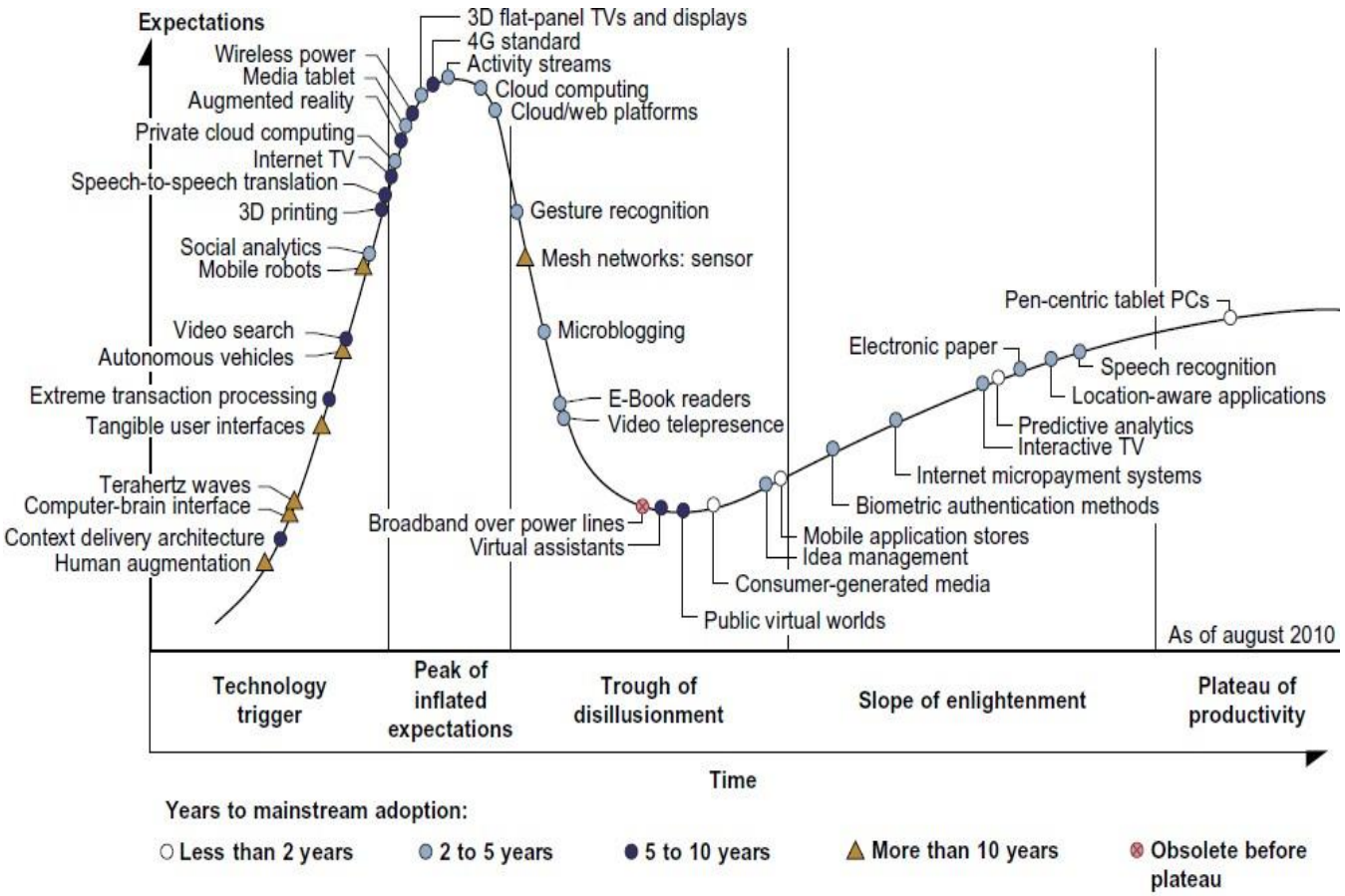

FIGURE 1.3

Hype cycle for Emerging Technologies, 2010.

→Any new and emerging computing and information technology may go through a hype cycle, as illustrated in Figure 1.3.

→The hype cycle in Figure 1.3 shows the technology status as of August 2010. For example, at that time consumer-generated media was at the disillusionment stage, and it was predicted to take less than two years to reach its plateau of adoption. Internet micropayment systems were forecast to take two to five years to move from the enlightenment stage to maturity. It was believed that 3D printing would take five to 10 years to move from the rising expectation stage to mainstream adoption, and mesh network sensors were expected to take more than 10 years to move from the inflated expectation stage to a plateau of mainstream adoption.

→Also as shown in Figure 1.3, the cloud technology had just crossed the peak of the expectation stage in 2010, and it was expected to take two to five more years to reach the productivity stage.

However, broadband over power line technology was expected to become obsolete before leaving the valley of disillusionment stage in 2010. Many additional technologies (denoted by dark circles in Figure 1.3) were at their peak expectation stage in August 2010, and they were expected to take five to 10 years to reach their plateau of success. Once a technology begins to climb the slope of enlightenment, it may reach the productivity plateau within two to five years. Among these promising technologies are the clouds, biometric authentication, interactive TV, speech recognition, predictive analytics, and media tablets.

1.3. The Internet of Things and Cyber-Physical Systems:

The Internet of Things:

→The concept of the IoT was introduced in 1999 at MIT . The IoT refers to the networked interconnection of everyday objects, tools, devices, or computers. One can view the IoT as a wireless network of sensors that interconnect all things in our daily life. These things can be large or small and they vary with respect to time and place. The idea is to tag every object using RFID or a related sensor or electronic technology such as GPS.

→In the IoT era, all objects and devices are instrumented, interconnected, and interacted with each other intelligently. This communication can be made between people and things or among the things themselves. Three communication patterns co-exist: namely H2H (human-to-human), H2T (human-tothing), and T2T (thing-to-thing). Here things include machines such as PCs and mobile phones.

→The dynamic connections will grow exponentially into a new dynamic network of networks, called the Internet of Things (IoT). The IoT is still in its infancy stage of development. Many prototype IoTs with restricted areas of coverage are under experimentation at the time of this writing.

→Cloud computing researchers expect to use the cloud and future Internet technologies to support fast, efficient, and intelligent interactions among humans, machines, and any objects on Earth.

→A smart Earth should have intelligent cities, clean water, efficient power, convenient transportation, good food supplies, responsible banks, fast telecommunications, green IT, better schools, good health care, abundant resources, and so on. This dream living environment may take some time to reach fruition at different parts of the world.

Cyber-Physical Systems:

A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world. A CPS integrates “cyber” (heterogeneous, asynchronous) with “physical” (concurrent and information-dense) objects. A CPS merges the “3C” technologies of computation, communication, and control into an intelligent closed feedback system between the physical world and the information world, a concept which is actively explored in the United States. The IoT emphasizes various networking connections among physical objects, while the CPS emphasizes exploration of virtual reality (VR) applications in the physical world. We may transform how we interact with the physical world just like the Internet transformed how we interact with the virtual world.

1.2 .TECHNOLOGIES FOR NETWORK-BASED SYSTEMS:

1. Multicore CPUs and Multithreading Technologies:

Consider the growth of component and network technologies over the past 30 years. They are crucial to the development of HPC and HTC systems. In **Figure 1.4**, processor speed is measured in millions of instructions per second (MIPS) and network bandwidth is measured in megabits per second (Mbps) or gigabits per second (Gbps). The unit GE refers to 1 Gbps Ethernet bandwidth.

Advances in CPU Processors:

Today, advanced CPUs or microprocessor chips assume a multicore architecture with dual, quad, six, or more processing cores. These processors exploit parallelism at ILP and TLP levels. Processor speed growth is plotted in the upper curve in **Figure 1.4** across generations of microprocessors or CMPs. We see growth from 1 MIPS for the VAX 780 in 1978 to 1,800 MIPS for the Intel Pentium 4 in 2002, up to a 22,000 MIPS peak for the Sun Niagara 2 in 2008. As the figure shows, Moore's law has proven to be pretty accurate in this case. The clock rate for these processors increased from 10 MHz for the Intel 286 to 4 GHz for the Pentium 4 in 30 years.

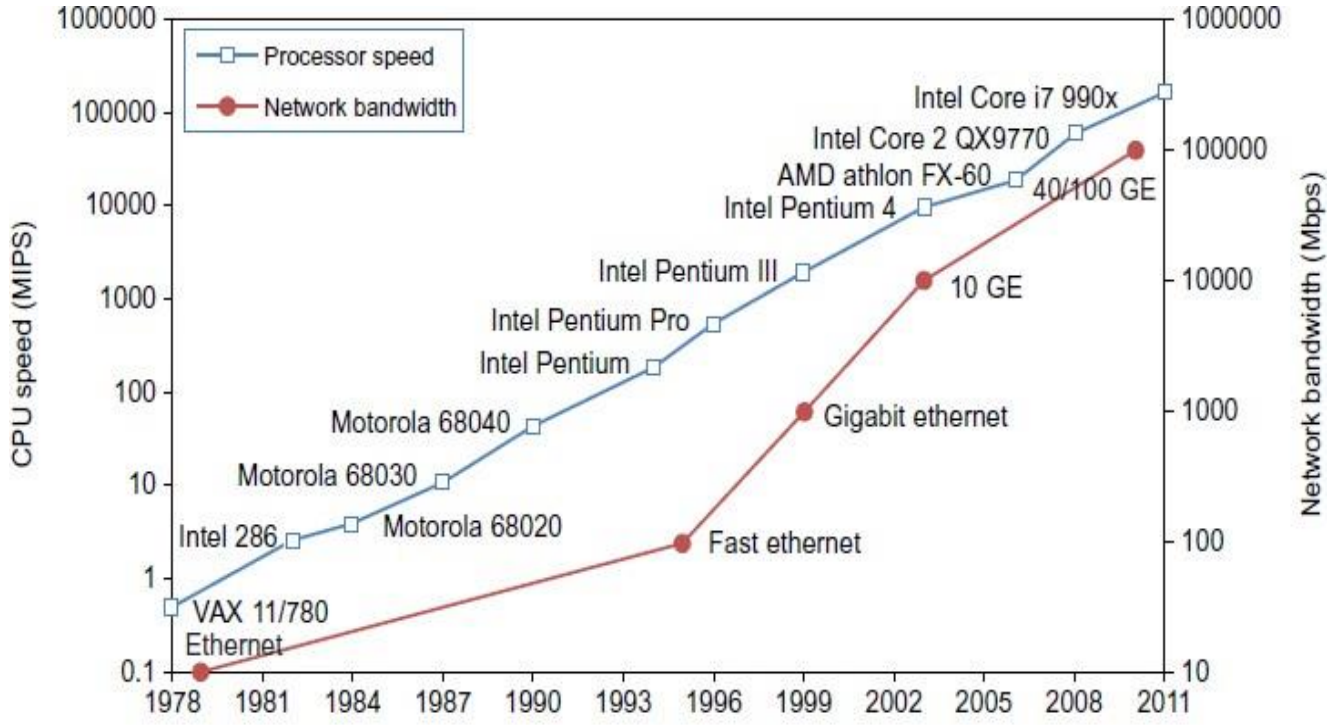

FIGURE 1.4

Improvement in processor and network technologies over 33 years.

Both multi-core CPU and many-core GPU processors can handle multiple instruction threads at different magnitudes today. Figure 1.5 shows the architecture of a typical multicore processor.

Each core is essentially a processor with its own private cache (L1 cache). Multiple cores are housed in the same chip with an 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 the chip. Multicore and multithreaded CPUs are equipped with many high-end processors, including the Intel i7, Xeon, AMD Opteron, Sun Niagara, IBM Power 6, and X cell processors. Each core could be also multithreaded.

For example, the Niagara II is built with eight cores with eight threads handled by each core.

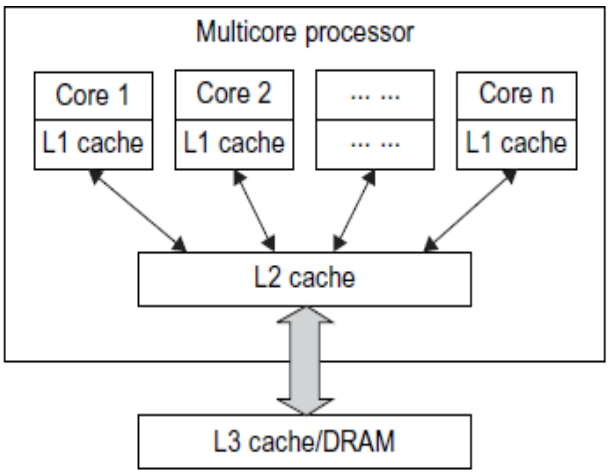

FIGURE 1.5

Schematic of a modern multicore CPU chip using a hierarchy of caches, where L1 cache is private to each core, on-chip L2 cache is shared and L3 cache or DRAM Is off the chip.

Multicore CPU and Many-Core GPU Architectures:

Multicore CPUs may increase from the tens of cores to hundreds or more in the future. But the CPU has reached its limit in terms of exploiting massive DLP due to the aforementioned memory wall problem. This has triggered the development of many-core GPUs with hundreds or more thin cores. Both IA-32 and IA-64 instruction set architectures are built into commercial CPUs. Now, x-86 processors have been extended to serve HPC and HTC systems in some high-end server processors.

Many RISC processors have been replaced with multicore x-86 processors and many-core GPUs in the Top 500 systems. This trend indicates that x-86 upgrades will dominate in data centers and supercomputers. The GPU also has been applied in large clusters to build supercomputers in MPPs.
In the future, the processor industry is also keen to develop asymmetric or heterogeneous chip multiprocessors that can house both fat CPU cores and thin GPU cores on the same chip.

Multithreading Technology:

Consider in Figure 1.6 the dispatch of five independent threads of instructions to four pipelined data paths (functional units) in each of the following five processor categories, from left to right: a four-issue superscalar processor, a fine-grain multithreaded processor, a coarse-grain multithreaded processor, a two-core CMP, and a simultaneous multithreaded (SMT) processor. The superscalar processor is single-threaded with four functional units.

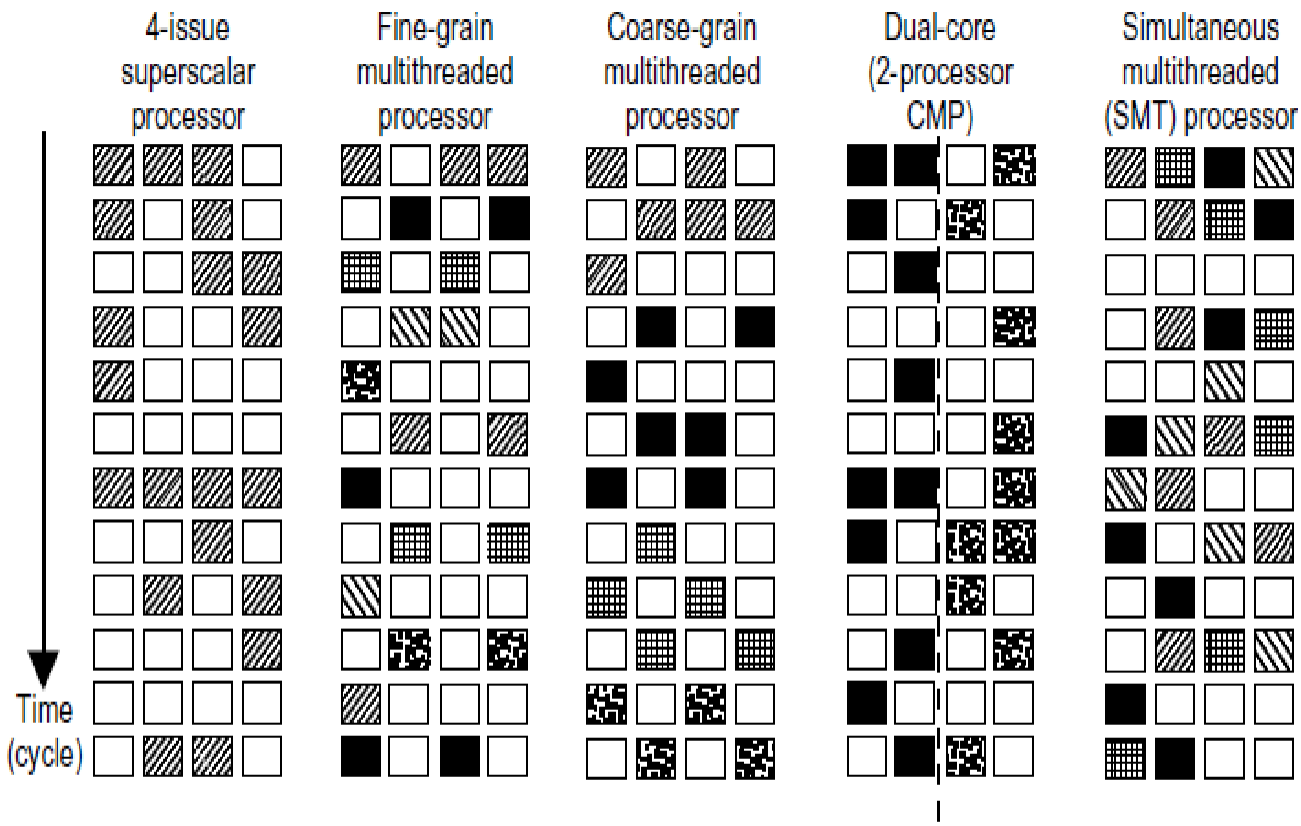

FIGURE 1.6

Five micro-architectures in modern CPU processors, that exploit ILP and TLP supported by multicore and multithreading technologies.

2. GPU Computing to Exascale and Beyond:

A GPU is a graphics coprocessor or accelerator mounted on a computer's graphics card or video card. Traditional CPUs are structured with only a few cores. For example, the Xeon X5670 CPU has six cores. However, a modern GPU chip can be built with hundreds of processing cores.

How GPUs work:

Early GPUs functioned as coprocessors attached to the CPU. Today, the NVIDIA GPU has been upgraded to 128 cores on a single chip.

Modern GPUs are not restricted to accelerated graphics or video coding. They are used in HPC systems to power supercomputers with massive parallelism at multicore and multithreading levels. GPUs are designed to handle large numbers of floating-point operations in parallel.

GPU Programming Model:

Figure 1.7 shows the interaction between a CPU and GPU in performing parallel execution of floating-point operations concurrently. The CPU is the conventional multicore processor with limited parallelism to exploit. The GPU has a many-core architecture that has hundreds of simple processing cores organized as multiprocessors. Each core can have one or more threads.

Essentially, the CPU's floating-point kernel computation role is largely offloaded to the many-core GPU.

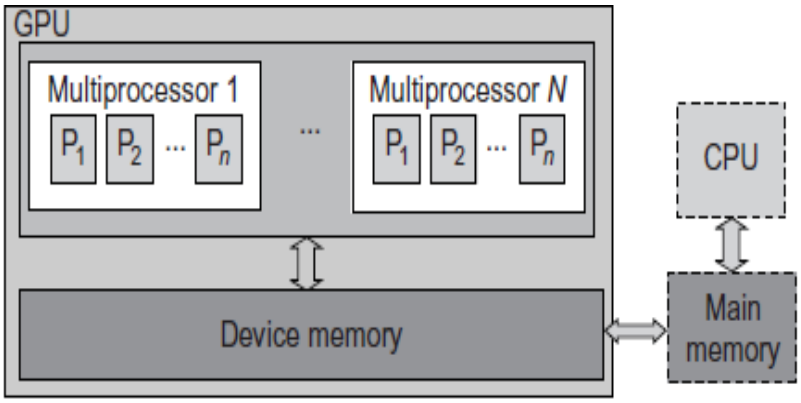

FIGURE 1.7

The use of a GPU along with a CPU for massively parallel execution in hundreds or thousands of processing cores.

Example 1.1 The NVIDIA Fermi GPU Chip with 512 CUDA Cores:

In November 2010, three of the five fastest supercomputers in the world (the Tianhe-1a, Nebulae, and Tsubame) used large numbers of GPU chips to accelerate floating-point computations. Figure 1.8 shows the architecture of the Fermi GPU, a next-generation GPU from NVIDIA. This is a streaming multiprocessor (SM) module. Multiple SMs can be built on a single GPU chip. The Fermi chip has 16 SMs implemented with 3 billion transistors. Each SM comprises up to 512 streaming processors (SPs), known as CUDA cores. The Tesla GPUs used in the Tianhe-1a have a similar architecture, with 448 CUDA cores.

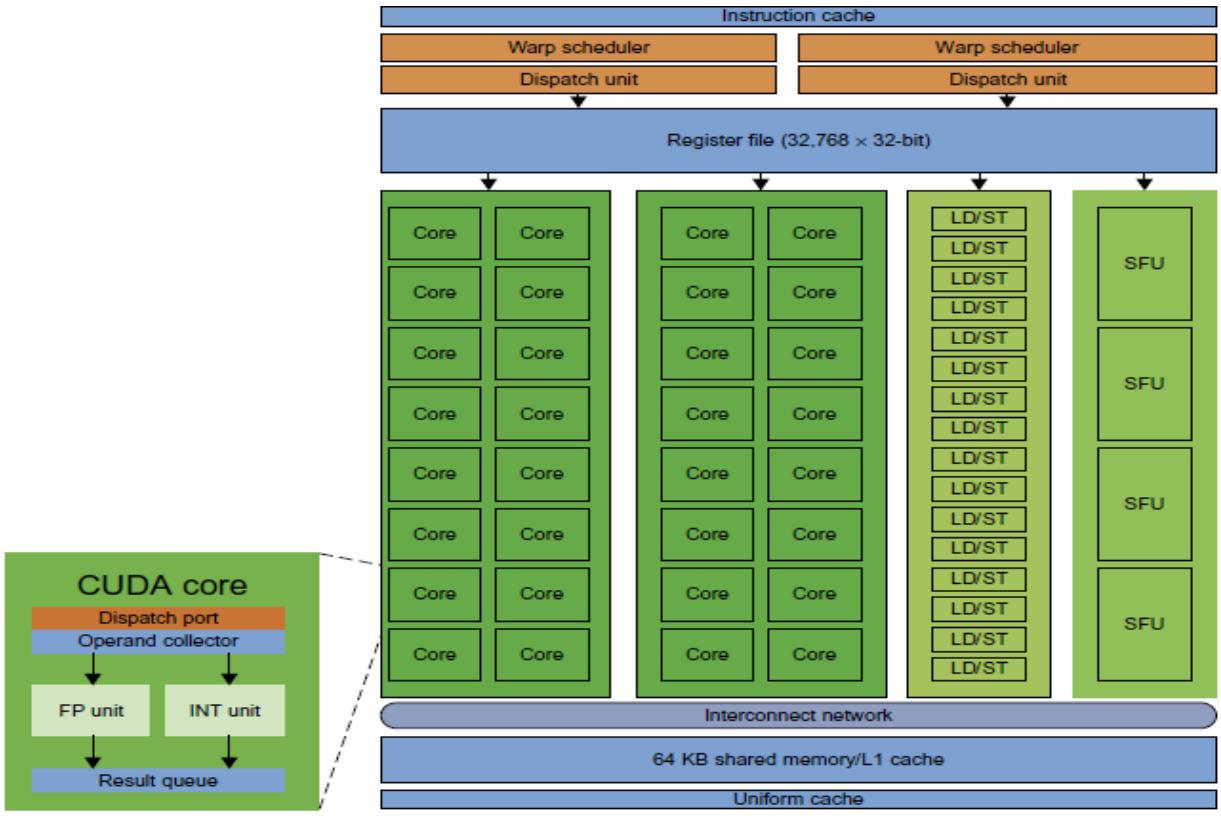

FIGURE 1.8

NVIDIA Fermi GPU built with 16 streaming multiprocessors (SMs) of 32 CUDA cores each; only one SM Is shown. More details can be found also in [49].

Power Efficiency of GPU:

Figure 1.9 compares the CPU and GPU in their performance/power ratio measured in Gflops/watt per core. In 2010, the GPU had a value of 5 Gflops/watt at the core level, compared with less than 1 Gflop/watt per CPU core. This may limit the scaling of future supercomputers. However, the GPUs may close the gap with theCPUs.

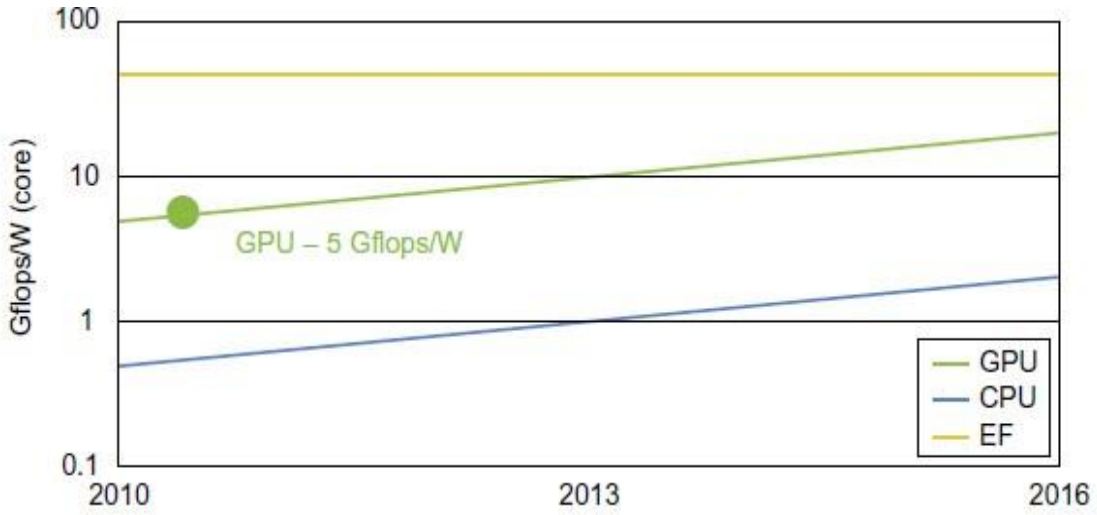

FIGURE 1.9

The GPU performance (middle line, measured 5 Gflops/W/core in 2011), compared with the lower CPU performance (lower line measured 0.8 Gflops/W/core in 2011) and the estimated 60 Gflops/W/core performance in 2011 for the Exascale (EF in upper curve) in the future.

3. Memory, Storage and Wide-Area Networking:

4.Memory Technology:

The growth of DRAM chip capacity from 16 KB in 1976 to 64 GB in 2011. This shows that memory chips have experienced a 4x increase in capacity every three years. 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, capacity increased from 260 MB in 1981 to 250 GB in 2004. The Seagate Barracuda XT hard drive reached 3 TB in 2011.

Disks and Storage Technology:

Beyond 2011, disks or disk arrays have exceeded 3 TB in capacity. The lower curve in Figure shows the disk storage growth in 7 orders of magnitude in 33 years. The rapid growth of flash memory and solid-state drives (SSDs) 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 to 1 million write cycles per block. So the SSD can last for several years, even under conditions of heavy write usage. Flash and SSD will demonstrate impressive speedups in many applications.

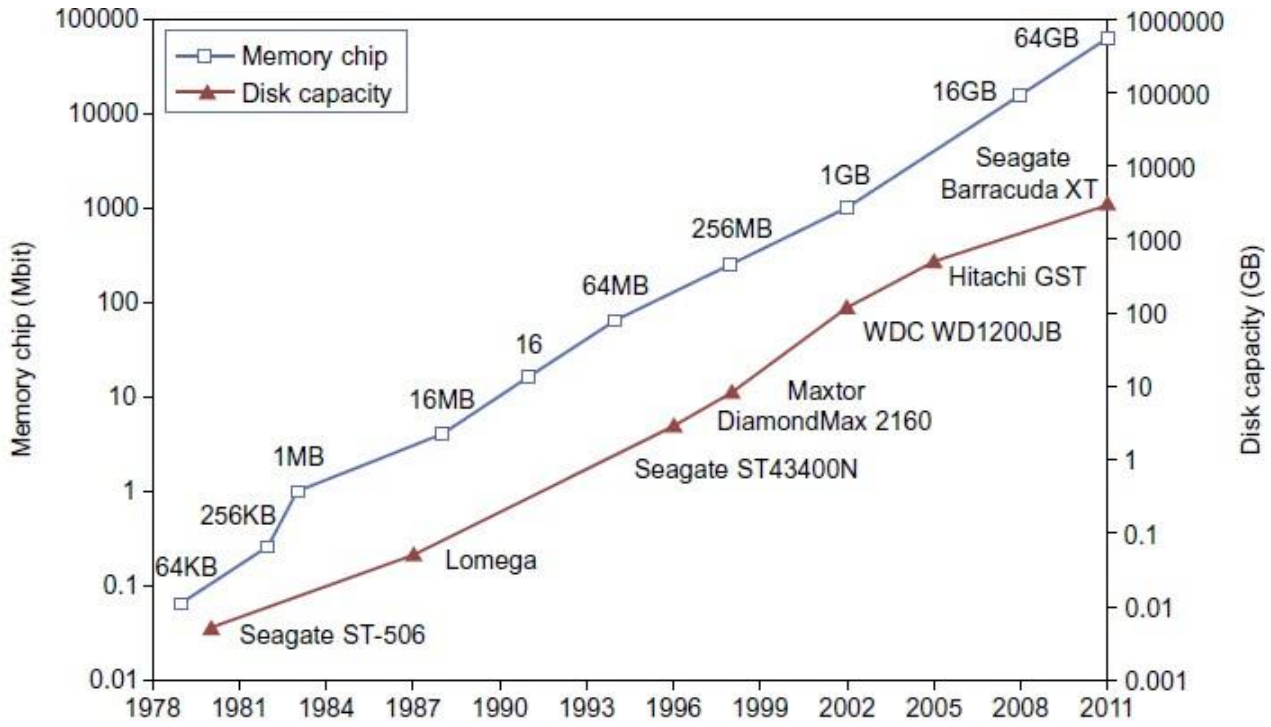

FIGURE 1.10

Improvement in memory and disk technologies over 33 years. The Seagate Barracuda XT disk has a capacity of 3 TB in 2011.

System-Area Interconnects:

The nodes in small clusters are mostly interconnected by an Ethernet switch or a local area network (LAN). As Figure 1.11 shows, a LAN typically is used to connect client hosts to big servers. A storage area network (SAN) connects servers to network storage such as disk arrays. Network attached storage (NAS) connects client hosts directly to the disk arrays. All three types of networks often appear in a large cluster built with commercial network components.

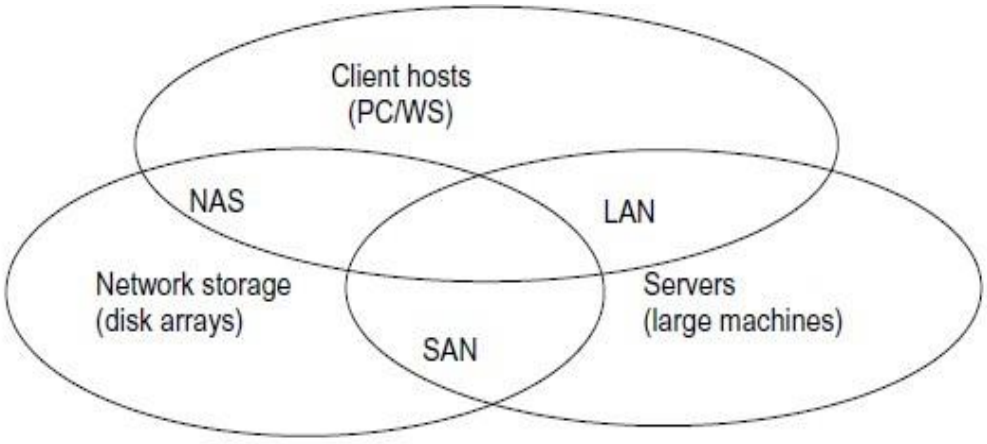

FIGURE 1.11

Three interconnection networks for connecting servers, client hosts, and storage devices; the LAN connects client hosts and servers, the SAN connects servers with disk arrays, and the NAS connects clients with large storage systems in the network environment.

Wide-Area Networking:

The lower curve in Figure 1.10 plots the rapid growth of Ethernet bandwidth from 10 Mbps in 1979 to 1 Gbps in 1999, and 40 ~ 100 GE in 2011. It has been speculated that 1 Tbps network links will become available by 2013. According to Berman, Fox, and Hey , network links with 1,000, 1,000, 100, 10, and 1 Gbps bandwidths were reported, respectively, for international, national, organization, optical desktop, and copper desktop connections in 2006.

5. 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 another platform with a different instruction set under a fixed OS. Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.

Figure 1.12 illustrates the architectures of three VM configurations.

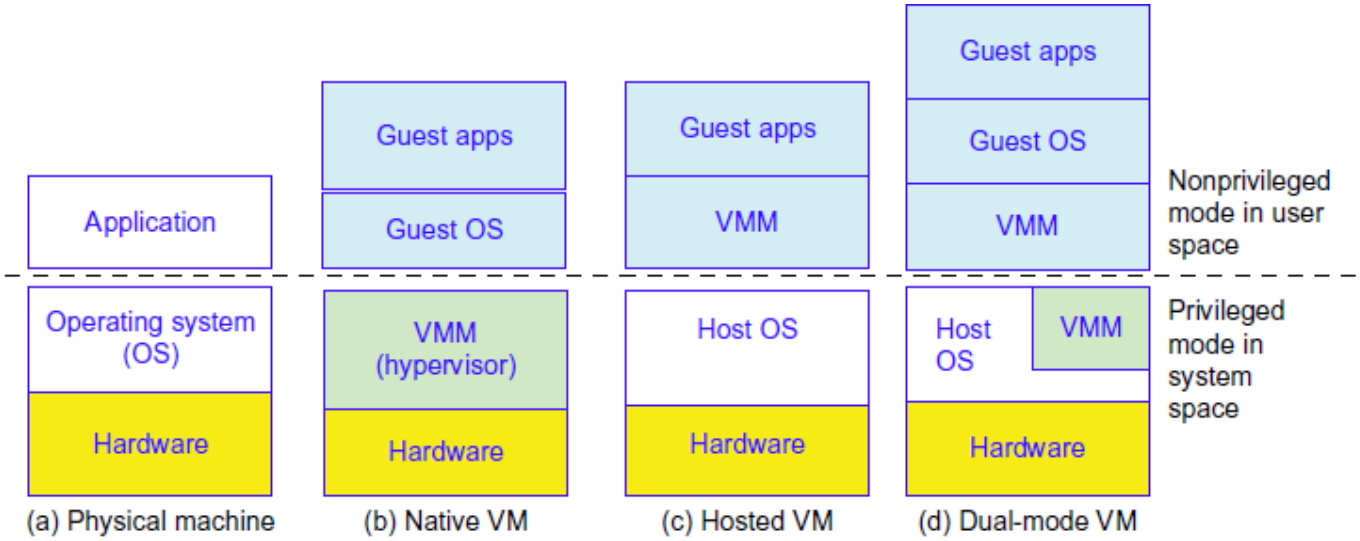

FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

Virtual Machines:

In Figure 1.12, The host machine is equipped with the physical hardware, as shown at the bottom of the figure. An example is an x-86 architecture desktop running its installed Windows OS, as shown in part (a) of the figure.

The VM can be provisioned for 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, one needs to deploy a middleware layer called a virtual machine monitor (VMM). Figure 1.12(b) shows a native VM installed with the use of a VMM called a hypervisor in privileged mode. For example, the hardware has 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. Another architecture is the host VM shown in Figure 1.12(c). Here the VMM runs in nonprivileged mode. The host OS need not be modified. The VM can also be implemented with a dual mode, as shown in **Figure 1.12(d).**

Part of the VMM runs at the user level and another part 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 a given hardware system to support the virtualization process. The VM approach offers hardware independence of the OS and applications. The user application running on its dedicated OS could be bundled together as a virtual appliance that can be ported to any hardware platform. The VM could run on an OS different from that of the host computer.

VM Primitive 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 it would on the physical hardware. Low-level VMM operations are indicated by Mendel Rosenblum and illustrated in **Figure 1.13.**

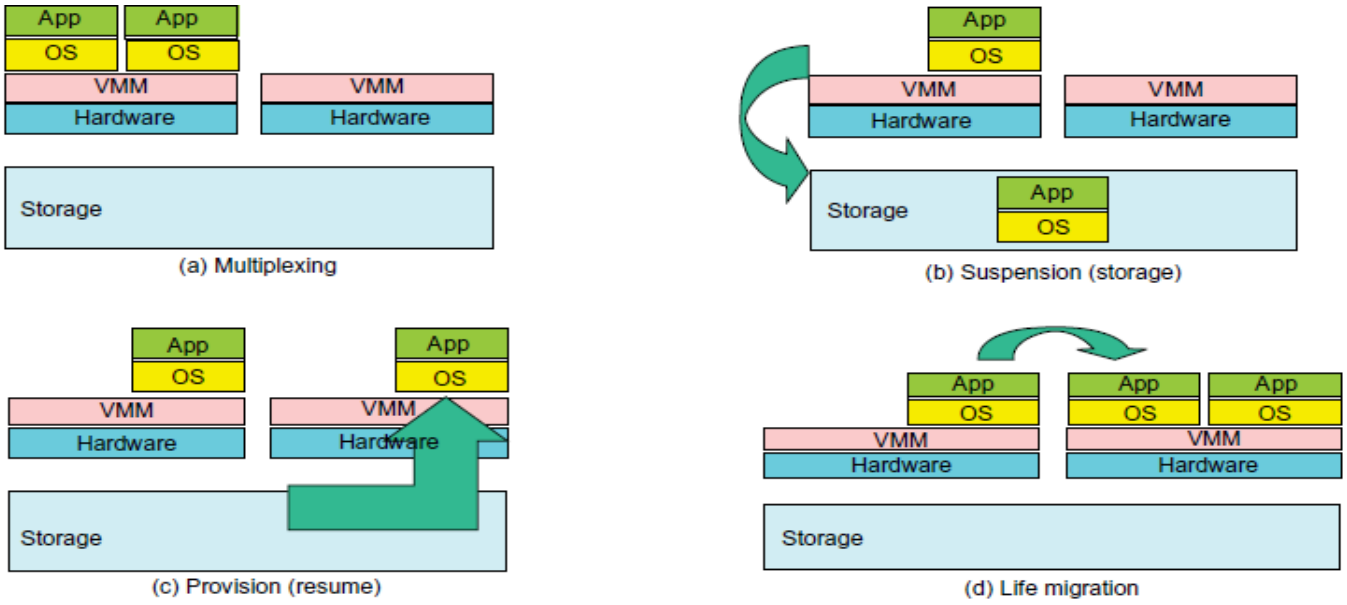

FIGURE 1.13

VM multiplexing, suspension, provision, and migration in a distributed computing environment.

First, the VMs can be multiplexed between hardware machines, as shown in Figure (a).
Second, a VM can be suspended and stored in stable storage, as shown in Figure (b).
Third, a suspended VM can be resumed or provisioned to a new hardware platform, as shown in Figure (c).

Finally, a VM can be migrated from one hardware platform to another, as shown in Figure (d).

Virtual Infrastructures:

Physical resources for compute, storage, and networking at the bottom of Figure 1.14 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 system resources to specific applications. The result is decreased costs and increased efficiency and responsiveness. Virtualization for server consolidation and containment is a good example of this.

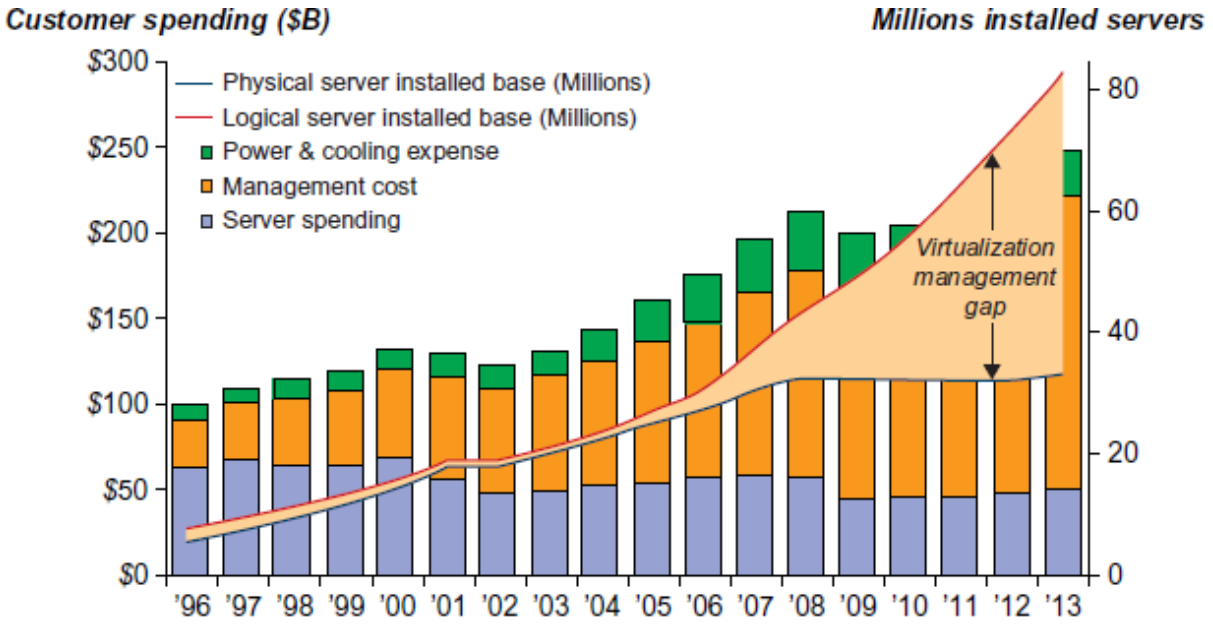

FIGURE 1.14

Growth and cost breakdown of data centers over the years.

6. Data Center Virtualization for Cloud Computing:

Data Center Growth and Cost Breakdown:

A large data center may be built with thousands of servers. Smaller data centers are typically built with hundreds of servers. The cost to build and maintain data center servers has increased over the years. According to a 2009 IDC report, typically only 30 percent of data center costs goes toward purchasing IT equipment (such as servers and disks), 33 percent is attributed to the chiller, 18 percent to the uninterruptible power supply (UPS), 9 percent to computer room air conditioning (CRAC), and the remaining 7 percent to power distribution, lighting, and transformer costs. Thus, about 60 percent of the cost to run a data center is allocated to management and maintenance. The server purchase cost did not increase much with time. The cost of electricity and cooling did increase from 5 percent to 14 percent in 15 years.

Low-Cost Design Philosophy:

High-end switches or routers may be too cost-prohibitive for building data centers. Thus, using high-bandwidth networks may not fit the economics of cloud computing. Given a fixed budget, commodity switches and networks are more desirable in data centers. Similarly, using commodity x86 servers is more desired over expensive mainframes. The software layer handles network traffic balancing, fault tolerance, and expandability. Currently, nearly all cloud computing data centers use Ethernet as their fundamental network technology.

Convergence of Technologies:

By linking computer science and technologies with scientists, a spectrum of e-science or e-research applications in biology, chemistry, physics, the social sciences, and the humanities has generated new insights from interdisciplinary activities. Cloud computing is a transformative approach as it promises much more than a data center model. It fundamentally changes how we interact with information. The cloud provides services on demand at the infrastructure, platform, or software level.

Iterative MapReduce extends MapReduce to support a broader range of data mining algorithms commonly used in scientific applications. The cloud runs on an extremely large cluster of commodity computers. Internal to each cluster node, multithreading is practiced with a large number of cores in many-core GPU clusters. Data-intensive science, cloud computing, and multicore computing are converging and revolutionizing the next generation of computing in architectural design and programming challenges. They enable the pipeline: Data becomes information and knowledge, and in turn becomes machine wisdom as desired in SOA.

3. SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING:

Distributed and cloud computing systems are built over a large number of autonomous computer nodes. These node machines are interconnected by SANs, LANs, or WANs in a hierarchical manner. With 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 with millions of computers connected to edge networks.

Massive systems are considered highly scalable, and can reach web-scale connectivity, either physically or logically. In Table 1.2, massive systems are classified into four groups: clusters, P2P networks, computing grids, and Internet clouds over huge data centers. In terms of node number, these four system classes may involve hundreds, thousands, or even millions of computers as participating nodes.

Table 1.2 Classification of Parallel and Distributed Computing Systems

Functionality, Applications	Computer Clusters [10,28,38]	Peer-to-Peer Networks [34,46]	Data/ Computational Grids [6,18,51]	Cloud Platforms [1,9,11,12,30]
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 clusters interconnected by high-speed network links over selected resource sites	Virtualized cluster of servers over data centers via SLA
Control and Resources Management	Homogeneous nodes with distributed control, running UNIX or Linux	Autonomous client nodes, free in and out, with self-organization	Centralized control, server-oriented with authenticated security	Dynamic resource provisioning of servers, storage, and networks
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 supercomputing, global problem solving, and data center services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc.	Google App Engine, IBM Bluecloud, AWS, and Microsoft Azure

1. Clusters of Cooperative Computers:

A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource. In the past, clustered computer systems have demonstrated impressive results in handling heavy workloads with large data sets.

Cluster Architecture:

Figure 1.15 shows the architecture of a typical server cluster built around a low-latency, 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 interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches. Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes. The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. 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 are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.

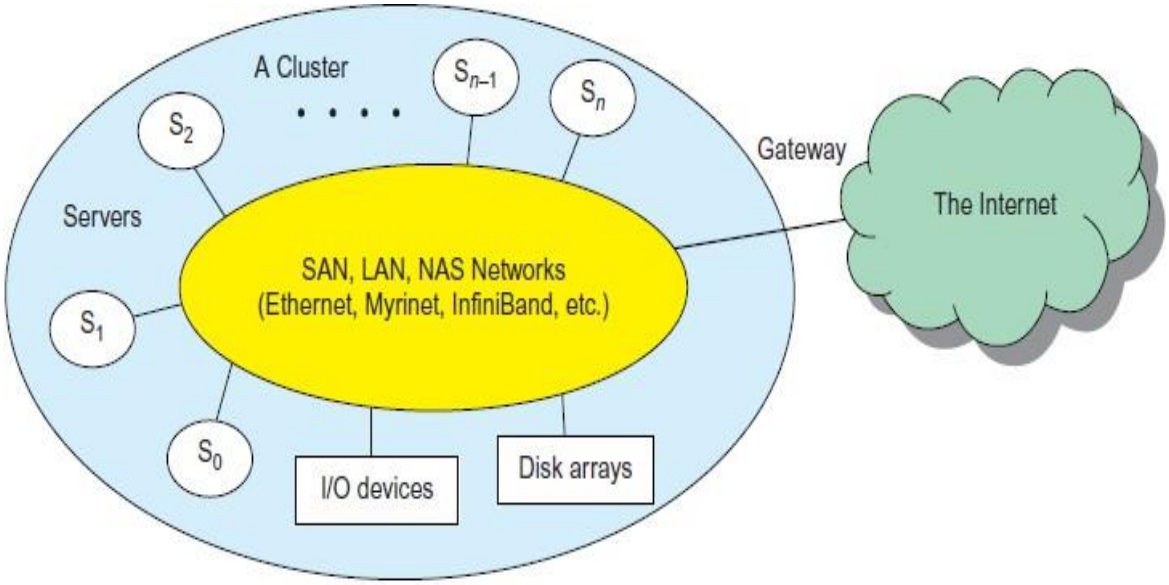

FIGURE 1.15

A cluster of servers interconnected by a high-bandwidth SAN or LAN with shared I/O devices and disk arrays; the cluster acts as a single computer attached to the Internet.

Single-System Image:

Greg Pfister [38] has indicated that an ideal cluster should merge multiple system images into a single-system image (SSI). Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes.

Hardware, Software, and Middleware Support:

Clusters exploring massive parallelism are commonly known as MPPs. Almost all HPC clusters in the Top 500 list are also MPPs. The building blocks are computer nodes (PCs, workstations, servers, or SMP), special communication software such as PVM or MPI, and a network interface card in each computer node. Most clusters run under the Linux OS. The computer nodes are interconnected by a high-bandwidth network (such as Gigabit Ethernet, Myrinet, InfiniBand, etc.). Special cluster middleware supports are needed to create SSI or high availability (HA). Both sequential and parallel applications can run on the

cluster, and special parallel environments are needed to facilitate use of the cluster resources.

Major Cluster Design Issues:

Table 1.3 Critical Cluster Design Issues and Feasible Implementations

Features	Functional Characterization	Feasible Implementations
Availability and Support	Hardware and software support for sustained HA in cluster	Failover, failback, check pointing, rollback recovery, nonstop OS, etc.
Hardware Fault Tolerance	Automated failure management to eliminate all single points of failure	Component redundancy, hot swapping, RAID, 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 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	Using a global job management system with better scheduling and monitoring	Application of single-job management systems such as LSF, Codine, etc.
Dynamic Load Balancing	Balancing the workload of all processing nodes along with failure recovery	Workload monitoring, 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 of scalable interconnect, performance monitoring, distributed execution environment, and better software tools

2. Grid Computing Infrastructures: (**********)

Internet services such as the Telnet command enables a local computer to connect to a remote computer. A web service such as HTTP enables remote access of remote web pages. Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.

Computational 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. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale. Enterprises or organizations present grids as integrated computing resources. They can also be viewed 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.

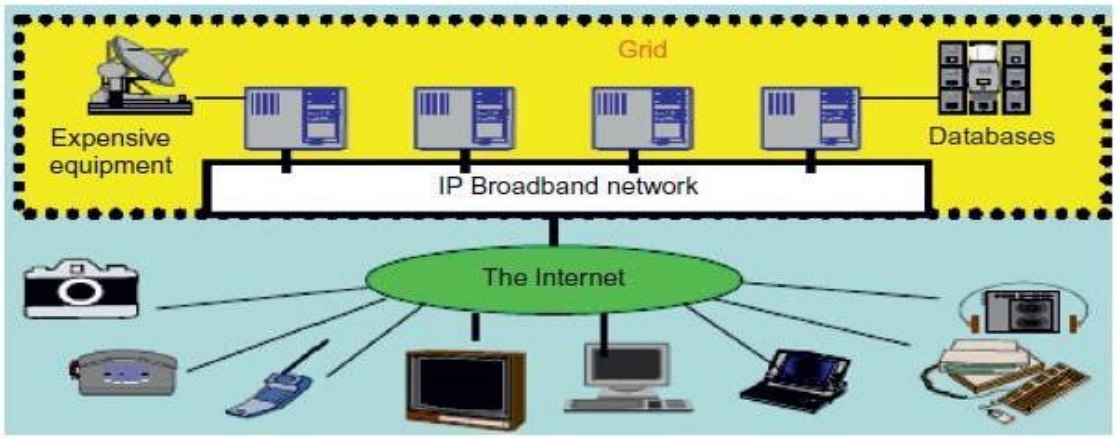

FIGURE 1.16

Computational grid or data grid providing computing utility, data, and information services through resource sharing and cooperation among participating organizations.

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, and others. New grid service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades.

In Table 1.4, grid systems are classified in essentially two categories: computational or data grids and P2P grids. Computing or data grids are built primarily at the national level.

Table 1.4 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 built in US, ChinaGrid in China, and the e-Science grid built in UK	JXTA, FightAid@home, SETI@home
Development Lessons Learned	Restricted user groups, middleware bugs, protocols to acquire resources	Unreliable user-contributed resources, limited to a few apps

3. Peer-to-Peer Network Families:

An example of a well-established distributed system is the client-server architecture. In this scenario, client machines (PCs and workstations) are connected to a central server for compute, e-mail, file access, and database applications. The P2P architecture offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented. In this section, P2P systems are introduced at the physical level and overlay networks at the logical level.

P2P Systems:

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 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.

Figure 1.17 shows the architecture of a P2P network 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 the TCP/IP and NAI protocols. Thus, the physical network varies in size and topology dynamically due to the free membership in the P2P network.

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 as shown in Figure 1.17.

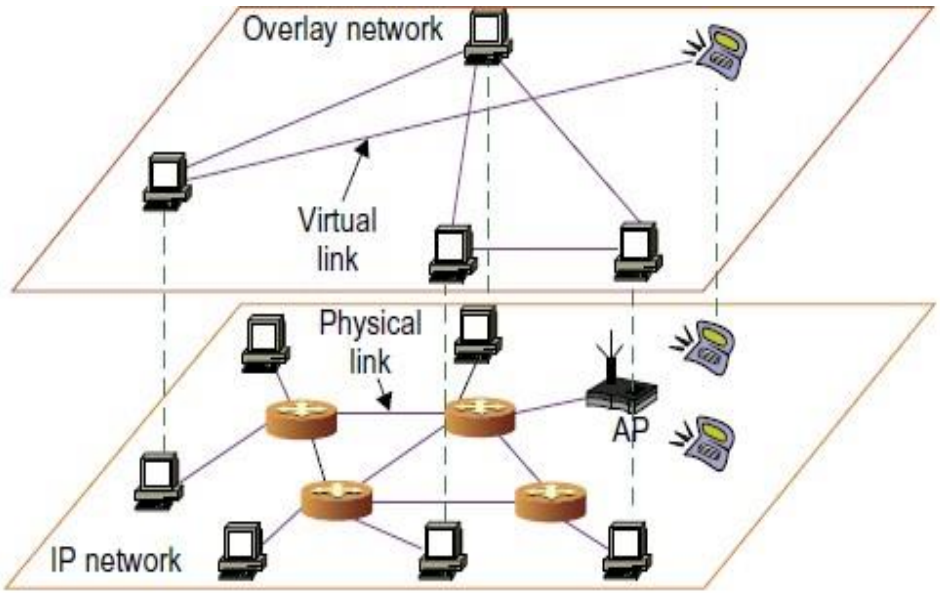

FIGURE 1.17

The structure of a P2P system by mapping a physical IP network to an overlay network built with virtual links.

P2P Application Families:

Based on application, P2P networks are classified into four groups, as shown in Table 1.5. The first amily is for distributed file sharing of digital contents (music, videos, etc.) on the P2P network. This includes many popular P2P networks such as Gnutella, Napster, and BitTorrent, among others. Collaboration P2P networks include MSN or Skype chatting, instant messaging, and collaborative design, among others. The third family is for distributed P2P computing in specific applications. For example, SETI@home provides 25 Tflops of distributed computing power, collectively, over 3 million Internet host machines. Other P2P platforms, such as JXTA, .NET, and FightingAID@home, support naming, discovery, communication, security, and resource aggregation in some P2P applications.

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 the OS; and different network connections and protocols make it too complex to apply in real applications.

Table 1.5 Major Categories of P2P Network Families [46]

System Features	Distributed File Sharing	Collaborative Platform	Distributed P2P Computing	P2P 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 serious online copyright violations	Lack of trust, disturbed by spam, privacy, and peer collusion	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.

4. Cloud Computing over the Internet:

"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 and user-facing applications." Based on this definition, a cloud allows workloads to be deployed and scaled out quickly through rapid provisioning of virtual or physical machines. The cloud supports redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures. Finally, the cloud system should be able to 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 data sets dynamically. The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers. Cloud computing leverages its low cost and simplicity to benefit both users and providers.

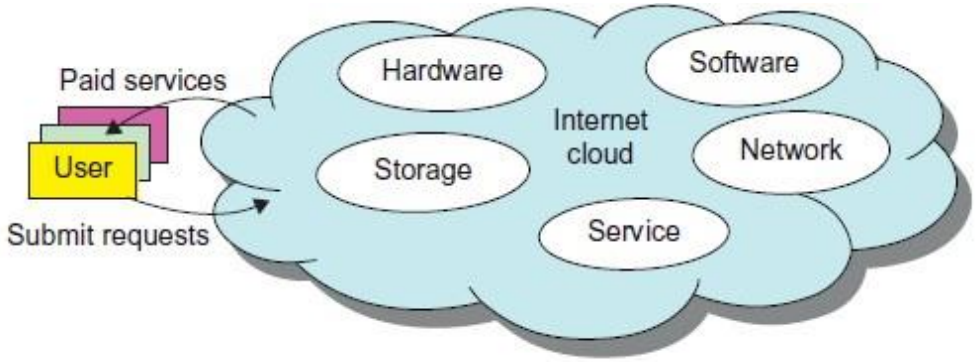

FIGURE 1.18

Virtualized resources from data centers to form an Internet cloud, provisioned with hardware, software, storage, network, and services for paid users to run their applications.

The Cloud Landscape:

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 us from these problems.

Figure 1.19 depicts the cloud landscape and major cloud players, based on three cloud service models.

• **Infrastructure as a Service (IaaS):** This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric. 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 specify when to request and release the needed resources.

• **Platform as a Service (PaaS):** This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java. The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the 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, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), 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.

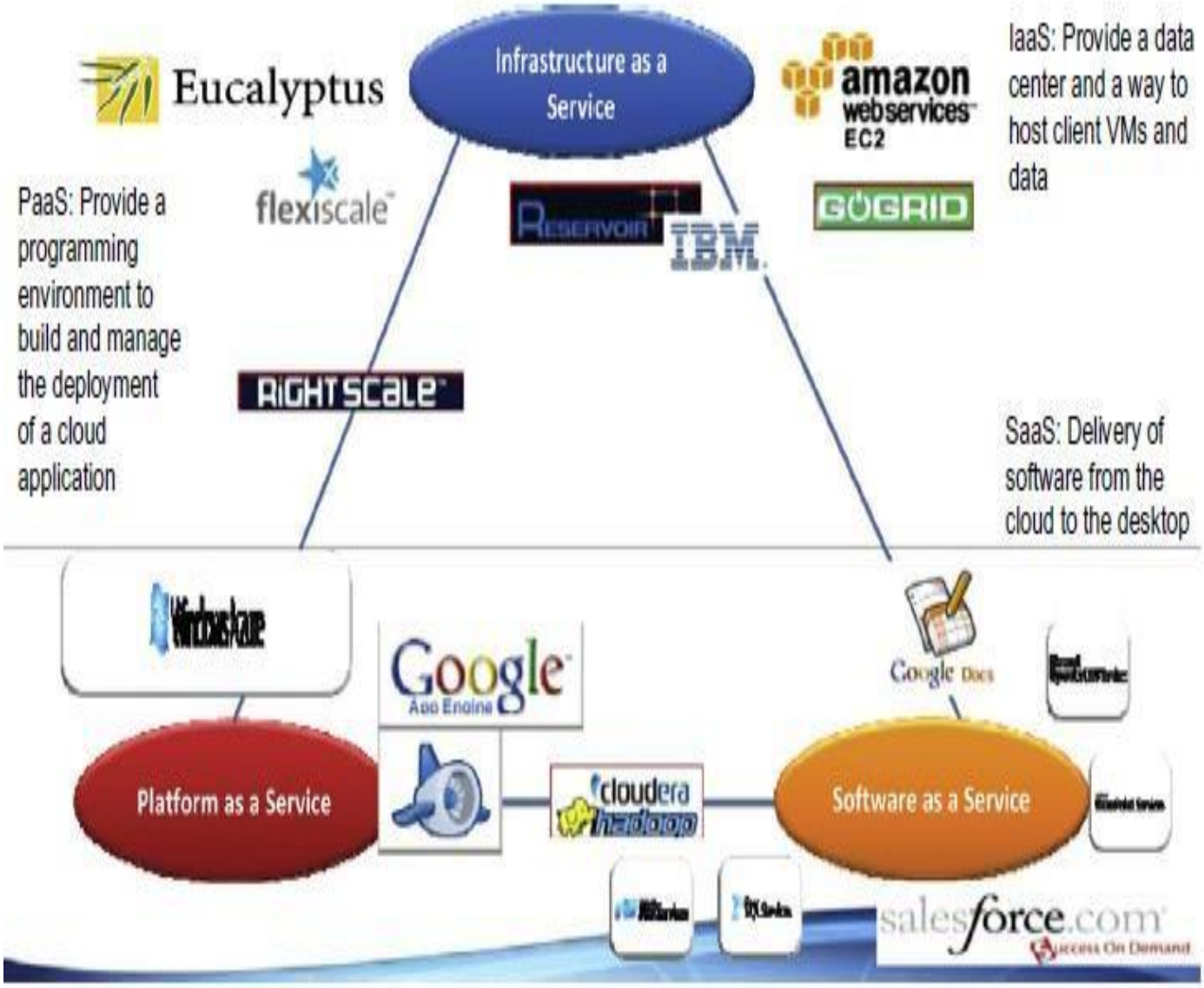

FIGURE 1.19

Three cloud service models in a cloud landscape of major providers.

The following list highlights eight reasons to adapt the cloud for upgraded Internet applications and web services:

1. Desired location in areas with protected space and higher energy efficiency
2. Sharing of peak-load capacity among a large pool of users, improving 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

4. SOFTWARE ENVIRONMENTS FOR DISTRIBUTED SYSTEMS AND CLOUDS: (********)

1. Service-Oriented Architecture (SOA):

In grids/web services, Java, and CORBA, an entity is, respectively, a service, a Java object, and a CORBA distributed object in a variety of languages. These architectures build on the traditional seven Open Systems Interconnection (OSI) layers that provide 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, and a broker network for CORBA. On top of this base environment one would build a higher level environment reflecting the special features of the distributed computing environment.

Figure 1.20 shows the layered architecture for distributed entities used in web services and grid systems.

Layered Architecture for Web Services and Grids (**********):

The entity interfaces correspond to the Web Services Description Language (WSDL), Java method, and CORBA interface definition language (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 Remote Procedure Call or RPC), fault recovery, and specialized routing. Often, these communication systems are built on message-oriented middleware (enterprise bus) infrastructure such as Web- Sphere MQ or Java Message Service (JMS) which provide rich functionality and support virtualization of routing, senders, and recipients.

In the case of fault tolerance, the features in the Web Services Reliable Messaging (WSRM) framework 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 reemployments the capabilities seen in concepts such as Internet Protocol Security (IPsec) and secures sockets in the OSI layers. Entity communication is supported by higher level services for registries, metadata, and management.

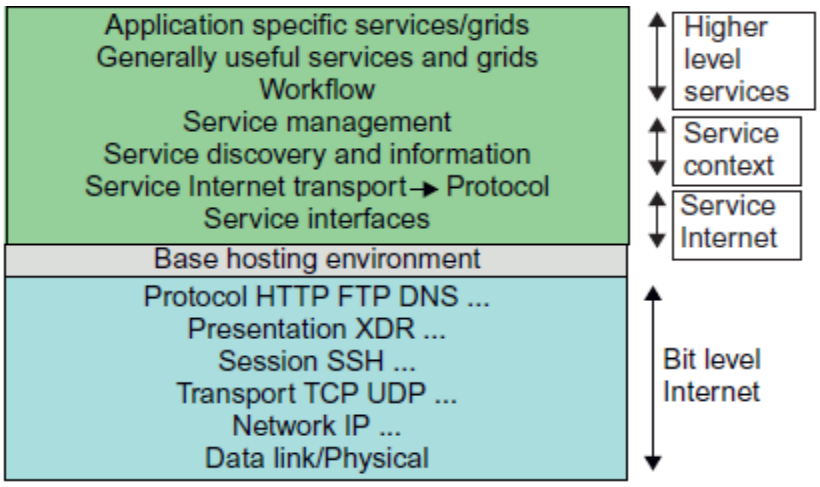

FIGURE 1.20

Layered achitecture for web services and the grids.

Web Services and Tools:

Loose coupling and support of heterogeneous implementations make services more attractive than distributed objects. The above picture corresponds to two choices of service architecture: web services or REST systems. Both web services and REST systems 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 Simple Object Access Protocol (SOAP). The hosting environment then becomes a universal distributed operating system with fully distributed capability carried by SOAP messages. This approach has mixed success as it has been hard to agree on key parts of the protocol and even harder to efficiently implement the protocol by software such as Apache Axis.

The Evolution of SOA:

As shown in **Figure 1.21**, service-oriented architecture (SOA) has evolved over the years. SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds (also known as interclouds), and systems of systems in general. A large number of sensors provide data-collection services, denoted in the **figure** as SS (sensor service). A sensor can be a ZigBee device, a Bluetooth device, a WiFi access point, a personal computer, a GPA, or a wireless phone, among other things. Raw data is collected by sensor services. All the SS devices interact with large or small computers, many forms of grids, databases, the compute cloud, the storage cloud, the filter cloud, the discovery cloud, and so on. Filter services (fs in the figure) are used to eliminate unwanted raw data, in order to respond to specific requests from the web, the grid, or web services.

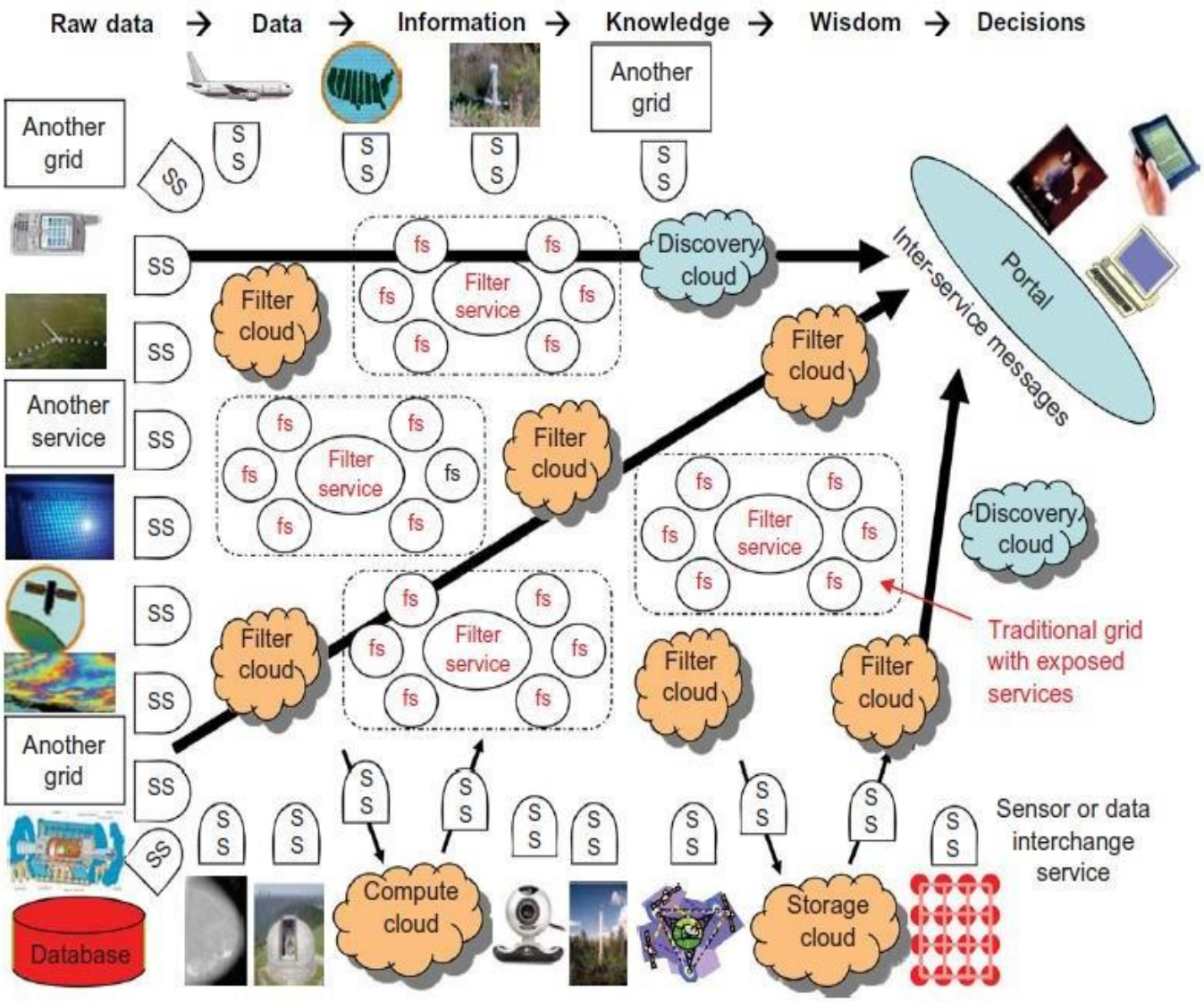

FIGURE 1.21

The evolution of SOA: grids of clouds and grids, where "SS" refers to a sensor service and "fs" to a filter or transforming service.

Grids versus Clouds:

The boundary between grids and clouds are getting blurred in recent years. For web services, workflow technologies are used to coordinate or orchestrate services with certain specifications used to define critical business process models such as two-phase transactions.

In general, a grid system applies static resources, while a cloud emphasizes elastic resources. For some researchers, the differences between grids and clouds are limited only in dynamic resource allocation based on virtualization and autonomic computing. One can build a grid out of multiple clouds. This type of grid can do a better job than a pure cloud, because it can explicitly support negotiated resource allocation. Thus one may end up building with a system of systems: such as a cloud of clouds, a grid of clouds, or a cloud of grids, or inter-clouds as a basic SOA architecture.

2. Trends toward Distributed Operating Systems: (****)

A distributed system inherently has 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 communication among node machines, it is best to have a distributed OS that manages all resources coherently and efficiently. Such a system is most likely to be a closed system, and it will likely rely on message passing and RPCs for internode communications. It should be pointed out that a distributed OS is crucial for upgrading the performance, efficiency, and flexibility of distributed applications.

Distributed Operating Systems:

There are three approaches for distributing resource management functions in a distributed computer system. The first approach is to build a network OS over a large number of heterogeneous OS platforms. Such an OS offers the lowest transparency to users, and is essentially a distributed file system, with independent computers relying on file sharing as a means of communication. The second approach is to develop middleware to offer a limited degree of resource sharing, similar to the MOSIX/OS developed for clustered systems. The third approach is to develop a truly distributed OS to achieve higher use or system transparency.

Table 1.6 compares the functionalities of these three distributed operating systems

Table 1.6 Feature Comparison of Three Distributed Operating Systems

Distributed OS Functionality	AMOEBA Developed at Vrije University [46]	DCE as OSF/1 by Open Software Foundation [7]	MOSIX for Linux Clusters at Hebrew University [3]
History and Current System Status	Written in C and tested in the European community; version 5.2 released in 1995	Built as a user extension on top of UNIX, VMS, Windows, OS/2, etc.	Developed since 1977, now called MOSIX2 used in HPC Linux and GPU clusters
Distributed OS Architecture	Microkernel-based and location-transparent, uses many servers to handle files, directory, replication, run, boot, and TCP/IP services	Middleware OS providing a platform for running distributed applications; The system supports RPC, security, and threads	A distributed OS with resource discovery, process migration, runtime support, load balancing, flood control, configuration, etc.
OS Kernel, Middleware, and Virtualization Support	A special microkernel that handles low-level process, memory, I/O, and communication functions	DCE packages handle file,time, directory, security services, RPC, and authentication at middleware or user space	MOSIX2 runs with Linux 2.6; extensions for use in multiple clusters and clouds with provisioned VMs
Communication Mechanisms	Uses a network-layer FLIP protocol and RPC to implement point-to-point and group communication	RPC supports authenticated communication and other security services in user programs	Using PVM, MPI in collective communications, priority process control, and queuing services

3. Parallel and Distributed Programming Models(*******)

Four programming models for distributed computing with expected scalable performance and application flexibility.

Table 1.7 summarizes three of these models, along with some software tool sets developed in recent years. As we will discuss, 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. Service clouds demand extending Hadoop, EC2, and S3 to facilitate distributed computing over distributed storage systems. Many other models have also been proposed or developed in the past.

FIGURE 1.22

A transparent computing environment that separates the user data, application, OS, and hardware in time and space – an ideal model for cloud computing.

Table 1.7 Parallel and Distributed Programming Models and Tool Sets

Model	Description	Features
MPI	A library of subprograms that can be called from C or FORTRAN to write parallel programs running on distributed computer systems [6,28,42]	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 clusters over large data sets, or in web search operations [16]	*Map* function generates a set of intermediate key/value pairs; *Reduce* function merges all intermediate values with the same key
Hadoop	A software library to write and run large user applications on vast data sets in business applications (http://hadoop.apache.org/core)	A scalable, economical, efficient, and reliable tool for providing users with easy access of commercial clusters

5. PERFORMANCE, SECURITY, AND ENERGY EFFICIENCY:

1. Performance Metrics and Scalability Analysis: (**********)

Performance metrics are needed to measure various distributed systems. In this section, we will discuss various dimensions of scalability and performance laws. Then we will examine system scalability against OS images and the limiting factors encountered.

Performance Metrics:

In a distributed system, performance is attributed to a large number of factors. System throughput is often measured in MIPS, Tflops (tera floating-point operations per second), or TPS (transactions per second). Other measures include job response time and network latency. An interconnection network that has low latency and high bandwidth is preferred. System overhead is often attributed to OS boot time, compile time, I/O data rate, and the runtime support system used. Other performance-related metrics include the QoS for Internet and web services; system availability and dependability; and security resilience for system defense against network attacks.

Dimensions of Scalability:

Users want to have a distributed system that can achieve scalable performance. Any resource upgrade in a system should be backward compatible with existing hardware and software resources.

• Size scalability This refers to achieving higher performance or more functionality by increasing the machine size. The word "size" refers to adding processors, cache, memory, storage, or I/O channels. The most obvious way to determine size scalability is to simply count the number of processors installed. Not all parallel computer or distributed architectures are equally sizescalable. For example, the IBM S2 was scaled up to 512 processors in 1997. But in 2008, the IBM BlueGene/L system scaled up to 65,000 processors.

• Software scalability This refers to upgrades in the OS or compilers, adding mathematical and engineering libraries, porting new application software, and installing more user-friendly programming environments. Some software upgrades may not work with large system configurations. Testing and fine-tuning of new software on larger systems is a nontrivial job.

• Application scalability This refers to matching problem size scalability with machine size scalability. Problem size affects the size of the data set or the workload increase. Instead of increasing machine size, users can enlarge the problem size to enhance system efficiency or cost-effectiveness.

• Technology scalability This refers to a system that can adapt to changes in building technologies, such as the component and networking technologies discussed in Section 3.1. When scaling a system design with new technology one must consider three aspects: time, space, and heterogeneity. (1) Time refers to generation scalability. When changing to new-generation processors, one must consider the impact to the motherboard, power supply, packaging and cooling, and so forth. Based on past experience, most systems upgrade their commodity processors every three to five years. (2) Space is related to packaging and energy concerns. Technology scalability demands harmony and portability among suppliers. (3) Heterogeneity refers to the use of hardware components or software packages from different vendors. Heterogeneity may limit the scalability.

Scalability versus OS Image Count:

In Figure 1.23, scalable performance is estimated against the multiplicity of OS images in distributed systems deployed up to 2010. Scalable performance implies that the system can achieve higher speed by adding more processors or servers, enlarging the physical node's memory size, extending the disk capacity, or adding more I/O channels.

The OS image is counted by the number of independent OS images observed in a cluster, grid, P2P network, or the cloud. SMP and NUMA are included in the comparison. An SMP (symmetric multiprocessor) server has a single system image, which could be a single node in a large cluster. By 2010 standards, the largest shared-memory SMP node was limited to a few hundred processors. The scalability of SMP systems is constrained primarily by packaging and the system interconnect used. NUMA (nonuniform memory access) machines are often made out of SMP nodes with distributed, shared memory. A NUMA machine can run with multiple operating systems, and can scale to a few thousand processors communicating with the MPI library. For example, a NUMA machine may have 2,048 processors running 32 SMP operating systems, resulting in 32 OS images in the 2,048-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 based on the cluster nodes concurrently in use. The cloud could be a virtualized cluster. As of 2010, the largest cloud was able to scale up to a few thousand VMs.

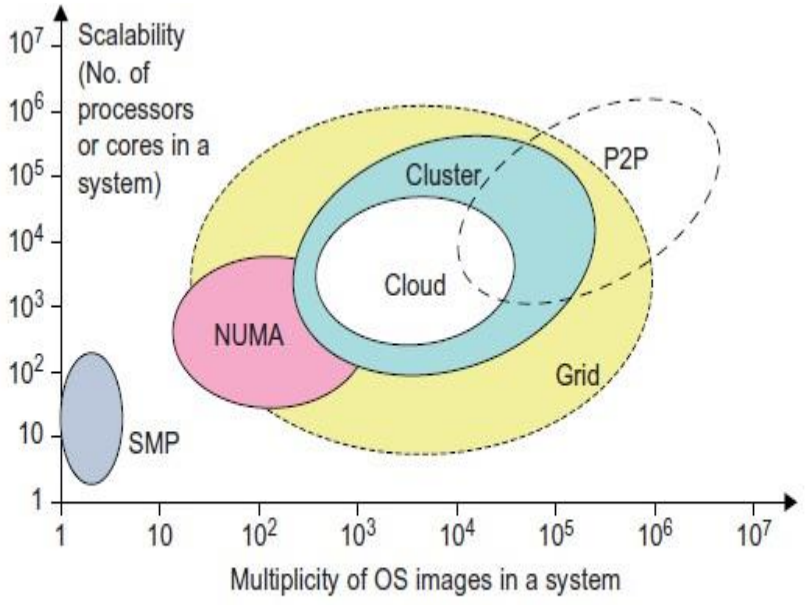

FIGURE 1.23

System scalability versus multiplicity of OS images based on 2010 technology.

Amdahl's Law:

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

All system or communication overhead is ignored here. The I/O time or exception handling time is also not included in the following speedup analysis. Amdahl's Law states that the *speedup factor* of using the n-processor system over the use of a single processor is expressed by:

$$\text{Speedup} = S = T/[\alpha T + (1-\alpha)T/n] = 1/[\alpha + (1-\alpha)/n]$$

2. Fault Tolerance and System Availability:

System Availability:

HA (high availability) is desired in all clusters, grids, P2P networks, and cloud systems. A system is highly available if it has a long mean time to failure (MTTF) and a short mean time to repair (MTTR). System availability is formally defined as follows:

System Availability =MTTF/(MTTF +MTTR)
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 designing for testability will help to enhance system availability and dependability.
In Figure 1.24, the effects on system availability are estimated by scaling the system size in terms of the number of processor cores in the system.

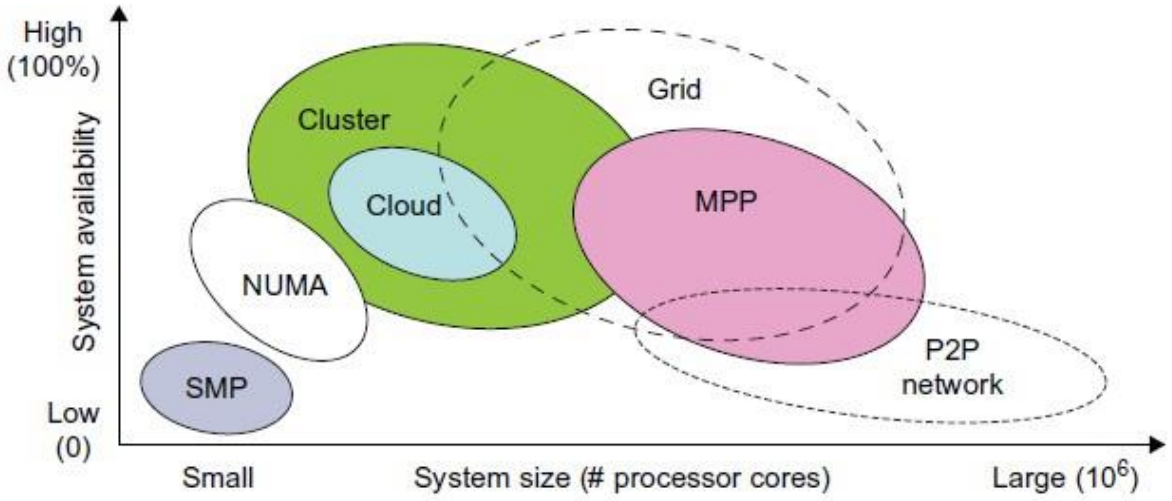

FIGURE 1.24
Estimated system availability by system size of common configurations in 2010.

3. Network Threats and Data Integrity:

Threats to Systems and Networks:

Network viruses have threatened many users in widespread attacks. These incidents have created a worm epidemic by pulling down many routers and servers, and are responsible for the loss of billions of dollars in business, government, and services. **Figure 1.25** summarizes various attack types and their potential damage to users. As the figure shows, information leaks lead to a loss of confidentiality.

Loss of data integrity may be caused by user alteration, Trojan horses, and service spoofing attacks. A denial of service (DoS) results in a loss of system operation and Internet connections.

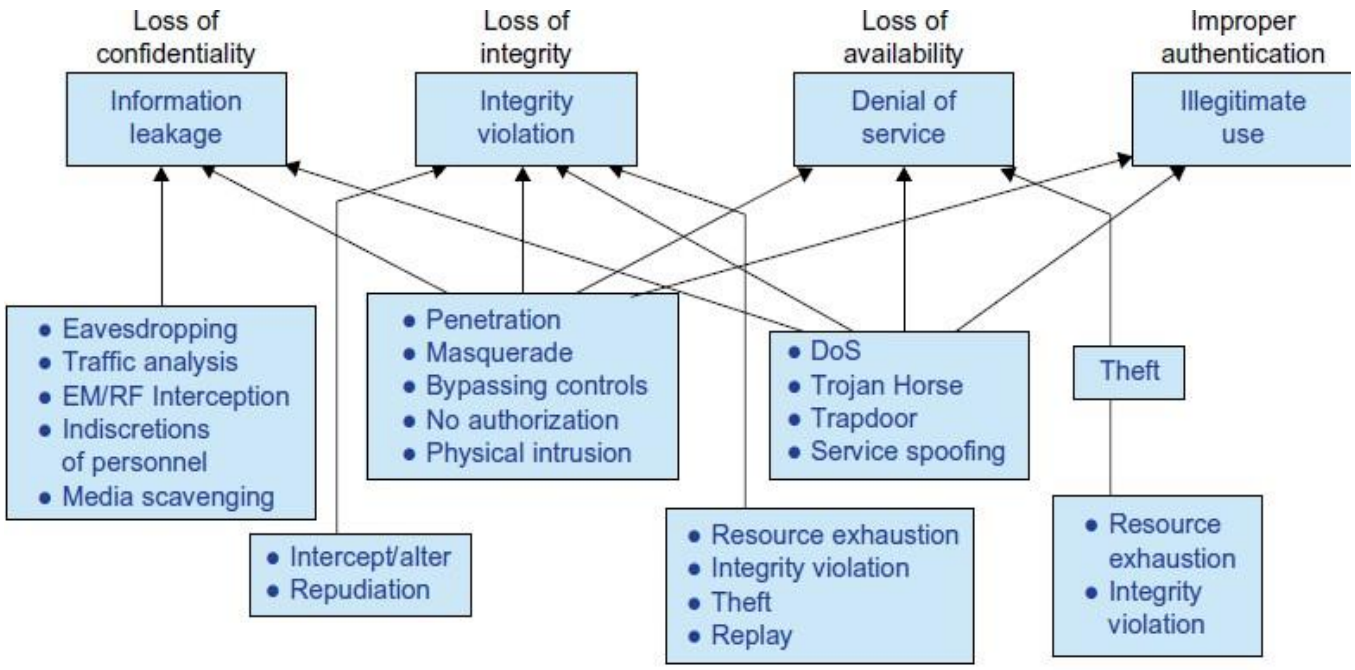

FIGURE 1.25

Various system attacks and network threats to the cyberspace, resulting 4 types of losses.

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). Online piracy has hindered the use of open P2P networks for commercial content delivery.

System Defence 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 themselves as access control policies or tokens, cryptographic systems, and so forth. However, an intruder could always penetrate a secure system because there is always a weak link in the security provisioning process. The second generation detected intrusions in a timely manner to exercise remedial actions.

These techniques included firewalls, intrusion detection systems (IDSes), PKI services, reputation systems, and so on. The third generation provides more intelligent responses to intrusions.

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 encountered new challenging issues including energy efficiency, and workload and resource outsourcing. These emerging issues are crucial not only on their own, but also for the sustainability of large-scale computing systems in general. This section reviews energy consumption issues in servers and HPC systems, an area known as distributed power management (DPM).

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, operational support, and energy every year. Therefore, companies should thoroughly identify whether their installed server farm (more specifically, the volume of provisioned resources) is at an appropriate level, particularly in terms of utilization. It was estimated in the past that, on average, one-sixth (15 percent) of the full-time servers in a company are left powered on without being actively used (i.e., they are 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.

Reducing Energy in Active Servers:

In addition to identifying unused/underutilized servers for energy savings, it is also necessary to apply appropriate techniques to decrease energy consumption in active distributed systems with negligible influence on their performance.

Power management issues in distributed computing platforms can be categorized into four layers (see Figure 1.26): the application layer, middleware layer, resource layer, and network layer.

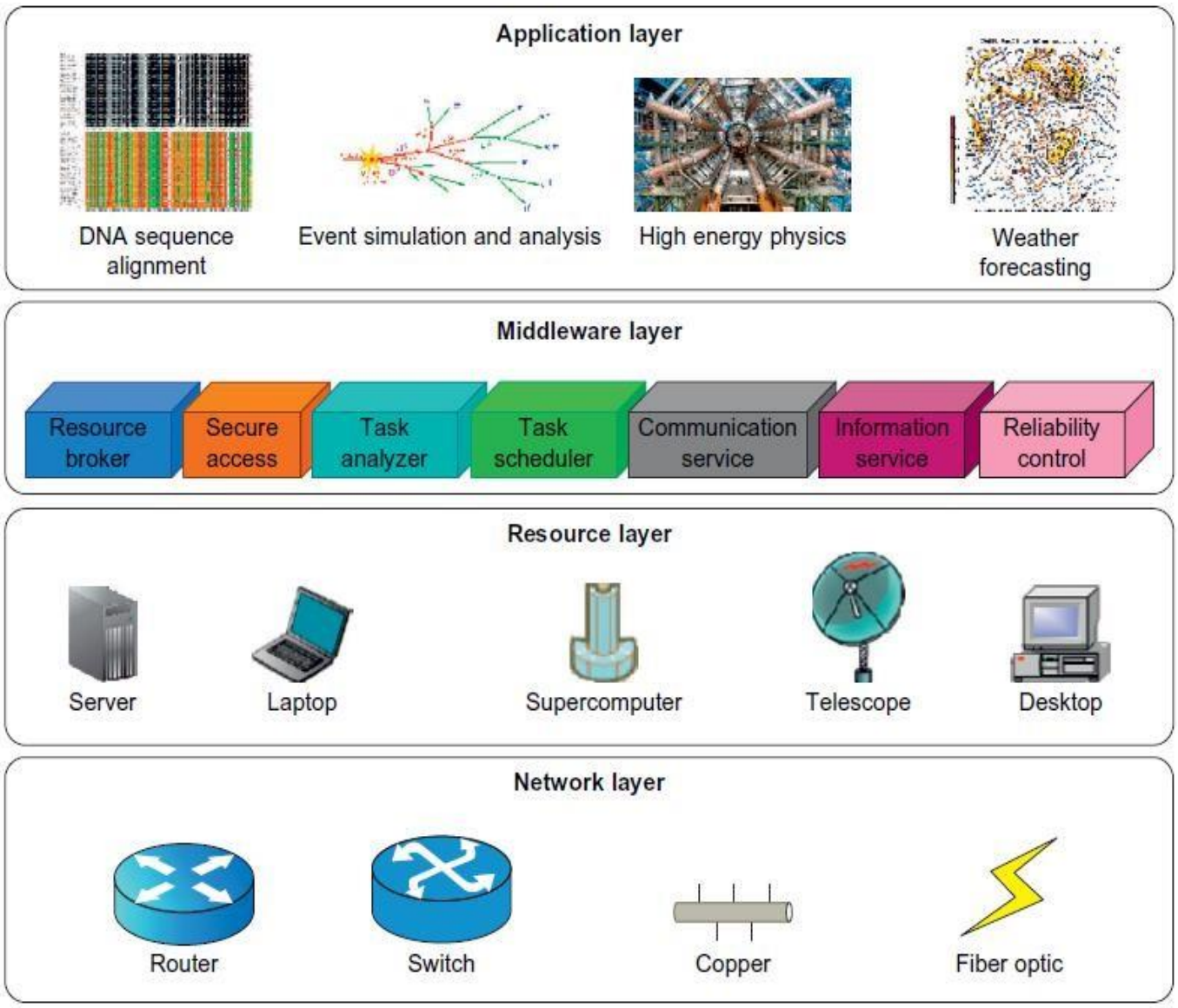

FIGURE 1.26

Four operational layers of distributed computing systems.

Application Layer:

Until now, most user applications in science, business, engineering, and financial areas tend to ncrease a system's speed or quality. By introducing energy-aware applications, the challenge is to design sophisticated multilevel and multi-domain energy management applications without hurting performance. The first step toward this end is to explore a relationship between performance and energy consumption. Indeed, an application's energy consumption depends strongly on the number of instructions needed to execute the application and the number of transactions with the storage unit (or memory). These two factors (compute and storage) are correlated and they affect 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 capabilities. It is also responsible for applying energy-efficient techniques, particularly in task scheduling. Until recently, scheduling was aimed at minimizing makespan, that is, the execution time of a set of tasks. Distributed computing systems necessitate 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 the operating system; 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 management (DPM) and dynamic voltage-frequency scaling (DVFS) are two popular methods incorporated into recent computer hardware systems . In DPM, hardware devices, such as the CPU, have the capability to switch from idle mode to one or more lower power modes. In DVFS, energy savings are achieved based on the fact that the power consumption in CMOS circuits has a direct relationship with frequency and the square of the voltage supply. Execution time and power consumption are controllable by switching among different frequencies
and voltages

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, determining how to measure, predict, and create a balance between energy consumption and performance.

Two major challenges to designing energy-efficient networks are:

• The models should represent the networks comprehensively as they should give a full understanding of interactions among time, space, and energy.

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

IMPORTANT QUESTIONS

1. Explain Scalable Computing over the Internet
2. Explain the age of internet computing?
3. Give the Applications of High-Performance and High-Throughput Systems.
4. Compare and contrast between HPC and HTC.
5. Illustrate different computing paradigms
6. Interpret about Degrees of Parallelism.
7. State the Layered architecture for web services and the grids.
8. What are the advantages of Cloud Computing over the Internet? Explain?
9. Give the architecture of P2P systems. What are the major categories of P2P Network families?
10. Describe various Technologies for Network based systems,
11. Discuss in detail about different system models for distributed and cloud computing?
12. Explain Software environments for distributed systems and clouds
13. Explain Performance and Security?
14. Discuss performance Metrics and Scalability Analysis for virtual Machines.
15. Explain about the Energy efficiency in Distributed computing.