UNIT - I

DISTRIBUTED SYSTEM MODELS AND ENABLING TECHNOLOGIES

This chapter presents the evolutionary changes that have occurred in parallel, distributed, and cloud computing over the past 30 years, driven by applications with variable workloads and large data sets.

- We study both **high-performance and high-throughput computing systems** in parallel computers appearing as computer clusters, service- oriented architecture, computational grids, peer-to-peer networks, Internet clouds, and the Internet of Things.

- These systems are distinguished by their hardware architectures, OS platforms, processing algorithms, communication protocols, and service models applied. We also introduce essential issues on the scalability, performance, availability, security, and energy efficiency in distributed systems.

1.1 SCALABLE COMPUTING OVER THE INTERNET

Scalability is **the ability** of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. For example, it can refer to the capability of a system to increase its total output under an increased load when resources (typically hardware) are added.

1.1.1The Age of Internet Computing

Billions of people use the Internet every day. As a result, supercomputer sites and large data centers must provide high-performance computing services to huge numbers of Internet users concurrently Because of this high demand, high-performance computing (HPC)applications is no longer optimal for measuring system performance. The emergence of computing clouds instead demands high-throughput computing (HTC) systems built with parallel and distributed computing technologies.

- **Internet computing** is the foundation on which e-business runs.
- It is the only architecture that can run all facets of business, from supplier collaboration and merchandise purchasing, to distribution and store operations, to customer sales and service.

The Platform Evolution

- Computer technology has gone through **five generations of development**, with each generation lasting from 10 to 20 years. Successive generations are overlapped in about 10 years.

- For instance, From 1950 to 1970, a handful of mainframes, including the IBM 360 and CDC 6400, were built to satisfy the demands of large businesses and government organizations.

- From 1960 to 1980, lower-cost minicomputers such as the DEC PDP 11 and VAX Series became popular among small businesses and on college campuses.
- From 1970 to 1990, we saw widespread use of personal computers built with VLSI microprocessors.
- From 1980 to 2000, massive numbers of portable computers and pervasive devices appeared in both wired and wireless applications.

Explanation

- **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. The cluster is often a collection of homogeneous compute nodes that are physically connected in close range to one another.
- **On the HTC side**, peer-to-peer (P2P) networks are formed for distributed file sharing and content delivery applications. A P2P system is built over many client.

Peer machines are globally distributed in nature. P2P, cloud computing, and web service platforms are more focused on HTC applications than on HPC applications.

High-Performance Computing

1 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.

2 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.

Three New Computing Paradigms

➢ The maturity of **Radio-frequency Identification (Rfid), Global Positioning System (GPS)**, and sensor technologies has triggered the development of the **Internet of Things (IoT)**.

Computing Paradigm Distinctions

❖ 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.

- **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.
- **Parallel computing** In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Interprocessor communication is accomplished through shared memory or via message passing.
- **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.
- **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.

Scalable Computing Trends and New Paradigms

Includes,

- Degrees of Parallelism
- Innovative Applications
- The Trend toward Utility Computing
- The Hype Cycle of New Technologies
- Fifty years ago, when hardware was bulky and expensive, most computers were designed in a bit-serial fashion.
- 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.

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 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.

➢ For example, distributed transaction processing is often practiced in the banking and finance industry. Transactions represent 90 percent of the existing market for reliable banking systems. Users must deal with multiple database servers in distributed transactions.

The Trend toward Utility Computing

❖ 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.

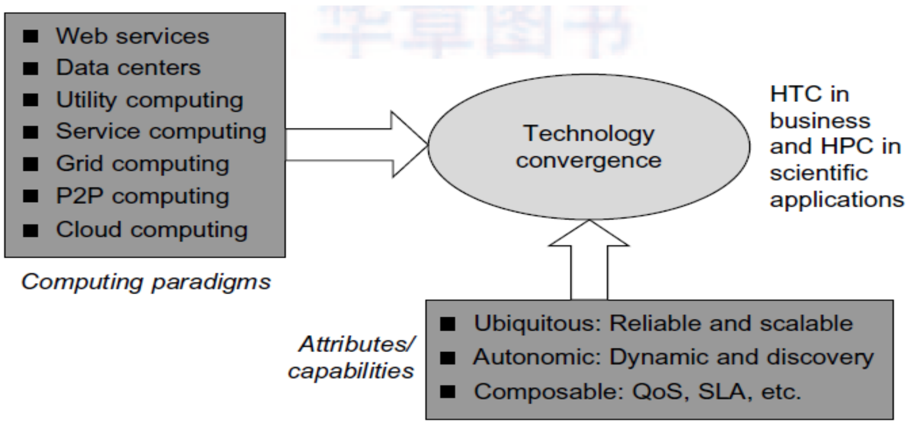


FIGURE 1.2

The vision of computer utilities in modern distributed computing systems.

- Figure 1.2 identifies major computing paradigms to facilitate the study of distributed systems and their applications. These paradigms share some common characteristics.
- **First,** they are all ubiquitous in daily life. Reliability and scalability are two major design objectives in these computing models.
- **Second,** they are aimed at autonomic operations that can be self- organized to support dynamic discovery.

The Hype Cycle of New Technologies

- Any new and emerging computing and information technology may go through a hype cycle, Generally illustrated in Figure 1.3. This cycle shows the expectations for the technology at five different stages.

- 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.

1.1.3 The Internet of Things and Cyber-Physical Systems

- Two Internet development trends:
- The Internet of Things
- Cyber-Physical Systems.

These evolutionary trends emphasize the extension of the Internet to everyday objects.

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.
- The IoT needs to be designed to track 100 trillion static or moving objects simultaneously. The IoT demands universal addressability of all of the objects or things. To reduce the complexity of identification, search, and storage, one can set the threshold to filter out fine-grain objects.

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.

- The IoT emphasizes various networking connections among physical objects, while the CPS emphasizes exploration of virtual reality (VR) applications in the physical world

TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

- Multicore CPUs and Multithreading Technologies
- Advances in CPU Processors
- Multicore CPU and Many-Core GPU Architectures
- Multithreading Technology
- GPU Computing to Exascale and Beyond
- How GPUs Work
- GPU Programming Model
- Power Efficiency of the GPU
- Memory, Storage, and Wide-Area Networking
- Memory Technology
- Disks and Storage Technology

- System-Area Interconnects
- Wide-Area Networking
- Virtual Machines and Virtualization Middleware
- Virtual Machines
- VM Primitive Operations
- Virtual Infrastructures
- Data Center Virtualization for Cloud Computing
- Data Center Growth and Cost Breakdown
- Low-Cost Design Philosophy
- Convergence of Technologies

Multi core CPUs and Multithreading Technologies

Advances in CPU Processors:

- Today, advanced CPUs or microprocessor chips assume a multi core architecture with dual, quad, six, or more processing cores. These processors exploit parallelism at ILP and TLP levels.

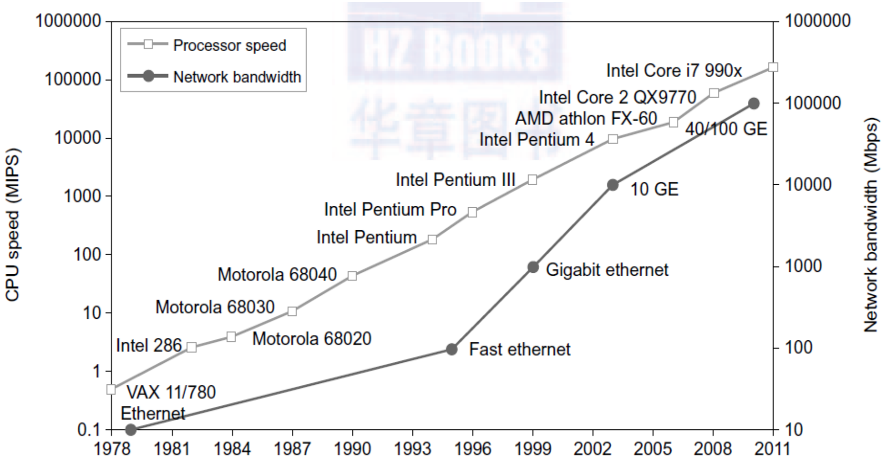


FIGURE 1.4

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.

2. 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.

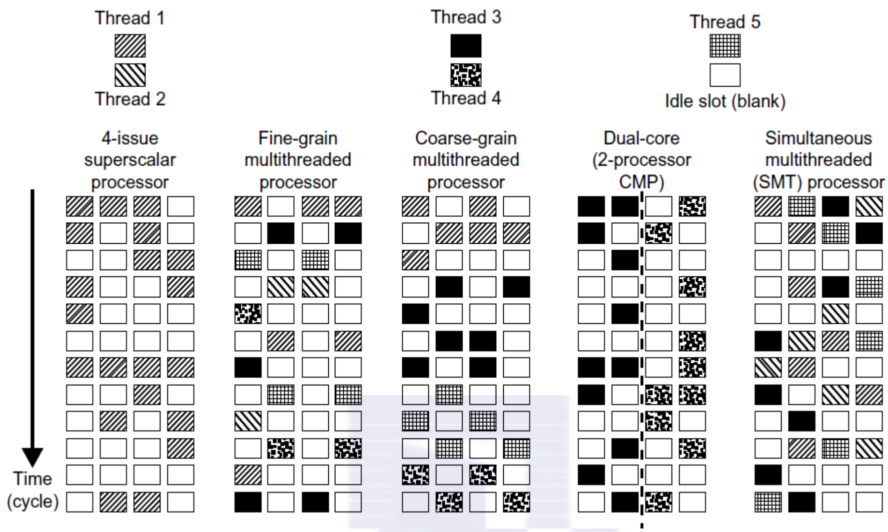


FIGURE 1.6

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

➢ 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. Each of the three multithreaded processors is four-way multithreaded over four functional data paths.

2.2 GPU Computing to Exascale and Beyond

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. Furthermore, each core on a GPU can handle eight threads of instructions. This translates to having up to 1,024 threads executed concurrently on a single GPU. This is true massive parallelism, compared to only a few threads that can be handled by a conventional CPU.

2. GPU Programming Model:

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. The CPU instructs the GPU to perform massive data processing. The bandwidth must be matched between the on-board main memory and the on-chip GPU memory.

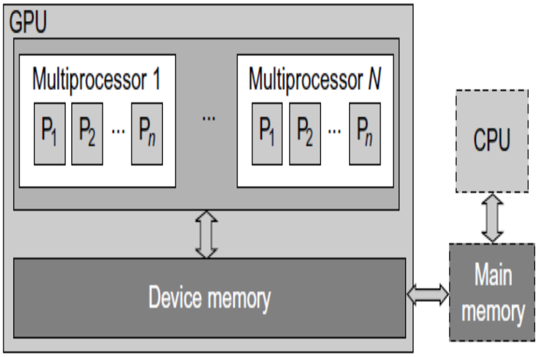


FIGURE 1.7

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

3. Power Efficiency of the GPU:

- Bill Dally of Stanford University considers power and massive parallelism as the major benefits of GPUs over CPUs for the future.

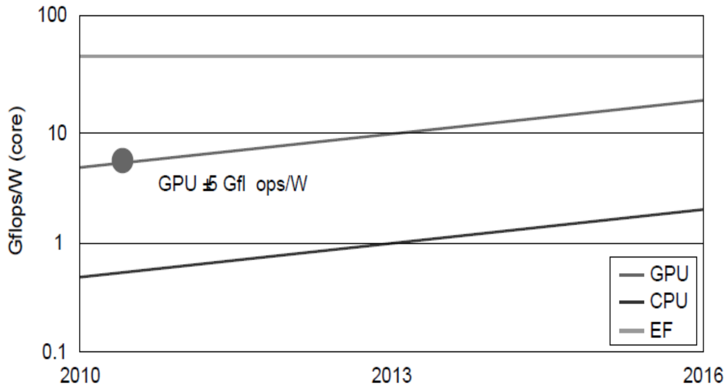


FIGURE 1.9

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.

Memory, Storage, and Wide-Area Networking

1. Memory Technology:

➢ The upper curve in Figure 1.10 plots 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.

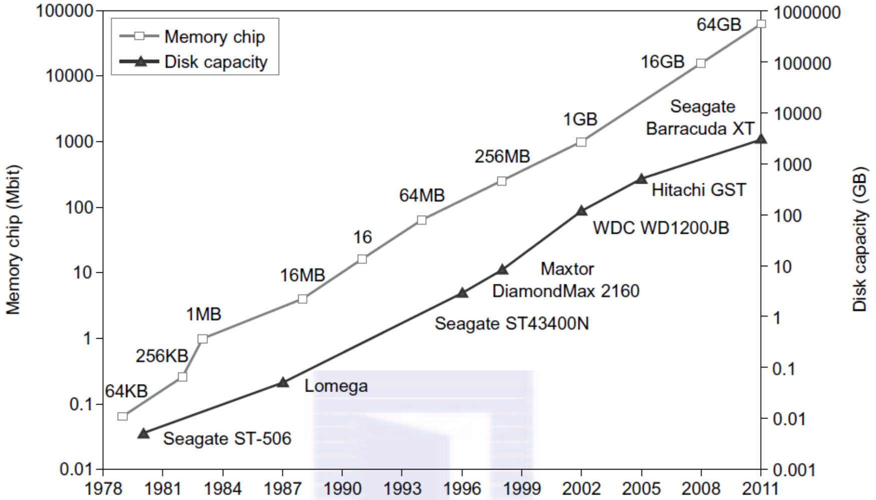


FIGURE 1.10

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

Disks and Storage Technology:

- Beyond 2011, disks or disk arrays have exceeded 3 TB in capacity. The lower curve in Figure 1.10 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.

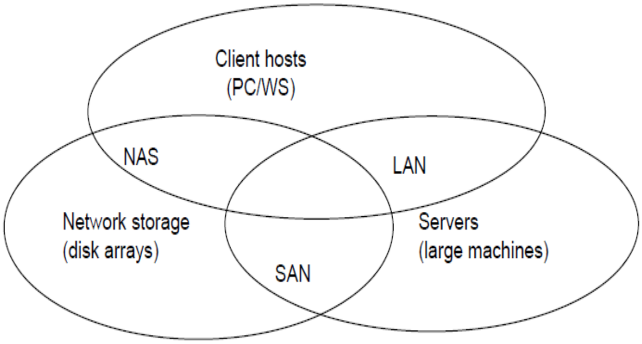


FIGURE 1.11

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.

3 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.

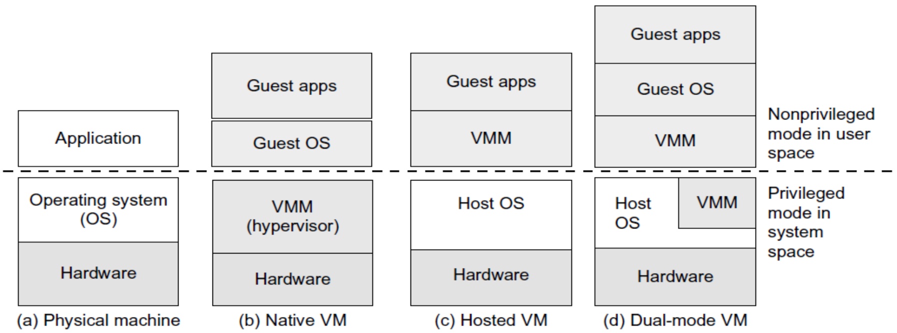


FIGURE 1.12

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

1. Virtual Machines:

➢ 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).

➢ 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.

2. 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.

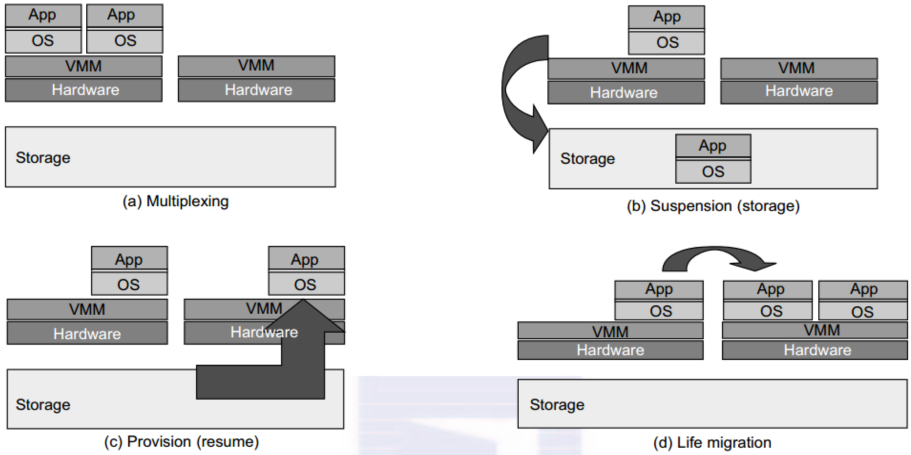


FIGURE 1.13

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

- First, the VMs can be multiplexed between hardware machines, as shown in Figure 1.13(a).
- Second, a VM can be suspended and stored in stable storage, as shown in Figure 1.13(b).
- Third, a suspended VM can be resumed or provisioned to a new hardware platform, as shown inFigure 1.13(c).

3. 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.

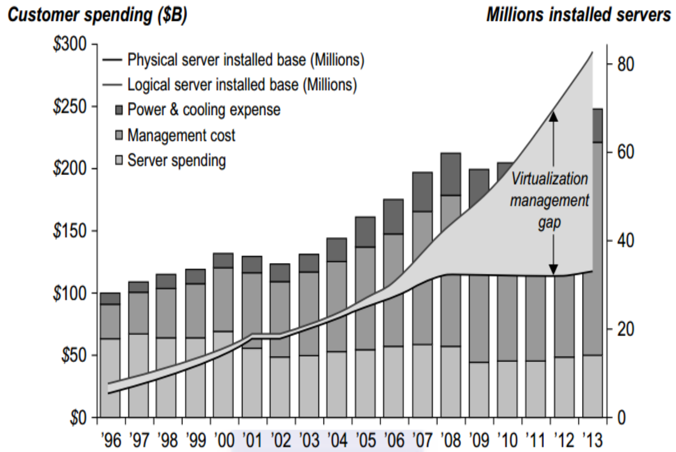


FIGURE 1.14

Growth and cost breakdown of data centers over the years.

1.2.5 Data Center Virtualization for Cloud Computing

➢ Almost all cloud platforms choose the popular x86 processors. Low-cost terabyte disks and Gigabit Ethernet are used to build data centers.

1. 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.

2. 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.
- Recent advances in SOA, Web 2.0, and mashups of platforms are pushing the cloud another step forward.
- Finally, achievements in autonomic computing and automated data center operations contribute to the rise of cloud computing.

Software Environments For Distributed Systems And Clouds

Service-Oriented Architecture (SOA):

1.1 Layered Architecture for Web Services and Grids:

These interfaces are linked with customized, high-level communication systems: SOAP, RMI, and IIOP in the three examples.

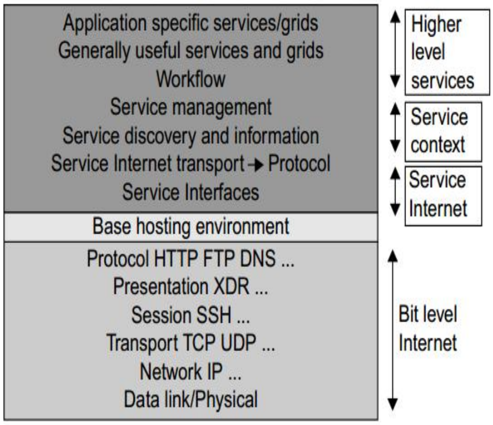


FIGURE 1.20

Layered achitecture for web services and the grids.

➢ These communication systems support features including particular message patterns (such as Remote Procedure Call or RPC), fault recovery, and specialized routing.

2. Web Services and Tools:

➢ Loose coupling and support of heterogeneous implementations make services more attractive than distributed objects.

1.4.2 Trends toward Distributed Operating Systems:

1.4.2.1 Distributed Operating Systems:

➢ Tanenbaum identifies three approaches for distributing resource management functions in a distributed computer system.

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

1.4.2.2 Amoeba versus DCE:

DCE is a middleware-based system for distributed computing environments. The Amoeba was academically developed at Free University in the Netherlands.

1.4.2.3 MOSIX2 for Linux Clusters:

➢ MOSIX2 is a distributed OS [3], which runs with a virtualization layer in the Linux environment.

1.4.2.3 Transparency in Programming Environments:

The user data, applications, OS, and hardware are separated into four levels. Data is owned by users, independent of the applications.

➢ The OS provides clear interfaces, standard programming interfaces, or system calls to application programmers.

1.4.3 Parallel and Distributed Programming Models

➢ we will explore four programming models for distributed computing with expected scalable performance and application flexibility.

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

1.4.3.1 Message-Passing Interface (MPI):

- This is the primary programming standard used to develop parallel and concurrent programs to run on a distributed system.
- Besides MPI, distributed programming can be also supported with low-level primitives such as the Parallel Virtual Machine (PVM).

1.4.3.2 MapReduce:

- This is a web programming model for scalable data processing on large clusters over large data sets.

- A typical MapReduce computation process can handle terabytes of data on tens of thousands or more client machines.

1.4.3.3 Hadoop Library:

- Hadoop offers a software platform that was originally developed by a Yahoo! group. The package enables users to write and run applications over vast amounts of distributed data.

1.4.3.4 Open Grid Services Architecture (OGSA) :

- The development of grid infrastructure is driven by large-scale distributed computing applications.

Table 1.8 Grid Standards and Toolkits for Scientific and Engineering Applications [6]

Standards	Service Functionalities	Key Features and Security Infrastructure
OGSA Standard	Open Grid Services Architecture; offers common grid service standards for general public use	Supports a heterogeneous distributed environment, bridging CAs, multiple trusted intermediaries, dynamic policies, multiple security mechanisms, etc.
Globus Toolkits	Resource allocation, Globus security infrastructure (GSI), and generic security service API	Sign-in multisite authentication with PKI, Kerberos, SSL, Proxy, delegation, and GSS API for message integrity and confidentiality
IBM Grid Toolbox	AIX and Linux grids built on top of Globus Toolkit, autonomic computing, replica services	Uses simple CA, grants access, grid service (ReGS), supports grid application for Java (GAF4J), GridMap in IntraGrid for security update

1.4.3.5 Globus Toolkits and Extensions:

➢ Globus is a middleware library jointly developed by the U.S. Argonne National Laboratory and USC Information Science Institute over the past decade.

1.5 PERFORMANCE, SECURITY, AND ENERGY EFFICIENCY

1.5.1 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.

1.5.1.1 Performance Metrics :

- We discussed CPU speed in MIPS and network bandwidth in Mbps to estimate processor and network performance.

1.5.1.2 Dimensions of Scalability

The following dimensions of scalability are characterized in parallel and distributed systems:

Size scalability

This refers to achieving higher performance or more functionality by increasing the machine size.

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.

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.

Technology scalability

This refers to a system that can adapt to changes in building technologies, such as the component and networking technologies.

1.5.1.3 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.

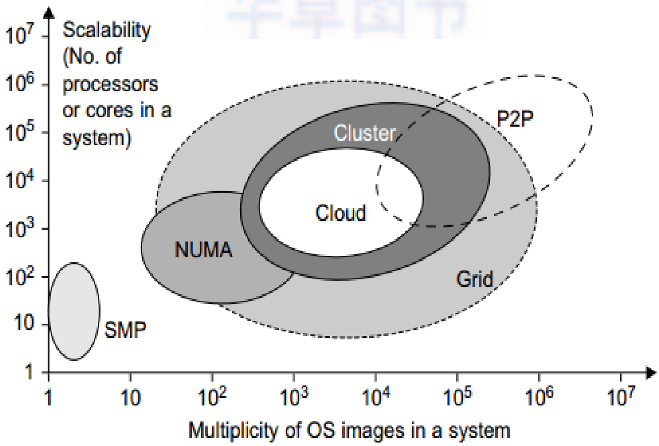


FIGURE 1.23

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

1.5.1.4 Amdahl's Law:

- Amdahl's Law states that the speedup factor of using the n-processor system over the use of a single processor is expressed by:

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

- The maximum speed up of n is achieved only if the sequential bottleneck α is reduced to zero or the code is fully parallelizable with α = 0.
- As the cluster becomes sufficiently large, that is, n → ∞, S approaches 1/α, an upper bound on the speedup S.
- Surprisingly, this upper bound is independent of the cluster size n. The sequential bottleneck is the portion of the code that cannot be parallelized.
- For example, the maximum speedup achieved is 4, if α = 0.25 or 1 − α = 0.75, even if one uses hundreds of processors.
- Amdahl's law teaches us that we should make the sequential bottleneck as small as possible.
- Increasing the cluster size alone may not result in a good speedup in this case.

1.5.1.5 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 data set. This was called fixed-workload speedup .

$$E = S/n = 1/[\alpha n + 1 - \alpha]$$

- 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 × 256 + 0.75] = 1.5% is observed. This is because only a few processors (say, 4) are kept busy, while the majority of the nodes are left idling.

1.5.1.6 Gustafson's Law :

- To achieve higher efficiency when using a large cluster, we must consider scaling the problem size to match the cluster capability. This leads to the following speedup law proposed by John Gustafson (1988), referred as scaled-workload speedup in.
- Let W be the workload in a givenprogram.When using an n-processor system, the user scales 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 a scaled workload W′ on n processors is defined by a scaled-workload speedup 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, the following efficiency expression is obtained:

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

➢ For the preceding program with a scaled workload, we can improve the efficiency of using a 256-node cluster to

$$E' = 0.25/256 + 0.75 = 0.751.$$

➢ One should apply Amdahl's law and Gustafson's law under different workload conditions. For a fixed workload, users should apply Amdahl's law.

1.5.2 Fault Tolerance and System Availability:

➢ In addition to performance, system availability and application flexibility are two other important design goals in a distributed computing system.

1.5.2.1 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:

$$System\ Availability = MTTF/(MTTF + MTTR)$$

- 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.

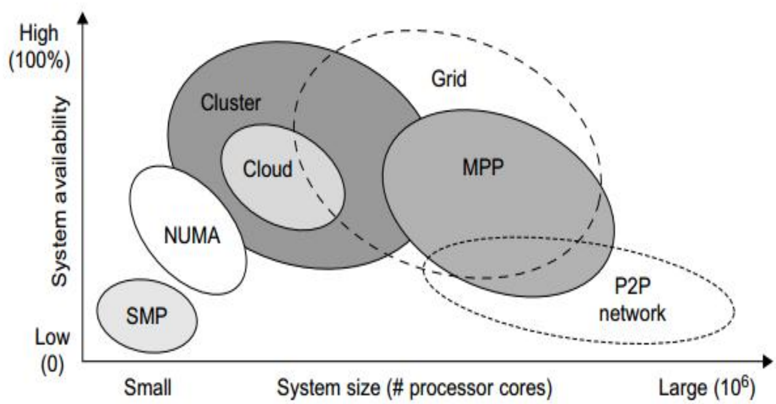


FIGURE 1.24

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

1.5.3 Network Threats and Data Integrity:

- Clusters, grids, P2P networks, and clouds demand security and copyright protection if they are to be accepted in today's digital society.
- This section introduces system vulnerability, network threats, defense countermeasures, and copyright protection in distributed or cloud computing systems.

1.5.3.1 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.

1.5.3.2 Security Responsibilities:

- Three security requirements are often considered: confidentiality, integrity, and availability for most Internet service providers and cloud users.
- The PaaS model relies on the provider to maintain data integrity and availability, but burdens the user with confidentiality and privacy control.

1.5.3.3 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).

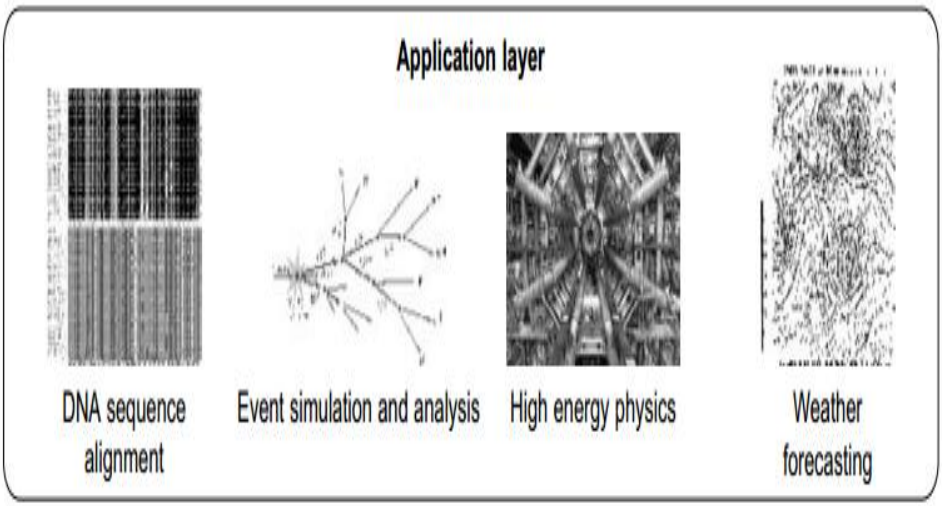
1.5.3.4 System Defense Technologies:

- 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 themselves as access control policies or tokens, cryptographic systems, and so forth.
- The second generation detected intrusions in a timely manner to exercise remedial actions.
- The third generation provides more intelligent responses to intrusions.

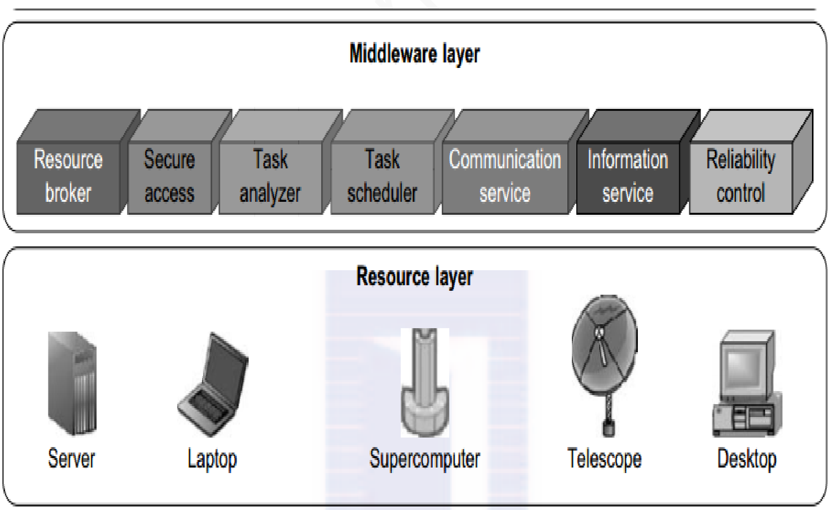
1.5.4 Energy Efficiency in Distributed Computing:

- 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).
- Protection of data centers demands integrated solutions. Energy consumption in parallel and distributed computing systems raises various monetary, environmental, and system performance issues.

Application Layer



Middleware layer, Resource layer



➢ The middleware layer acts as a bridge between the application layer and the resource layer.

Resource Layer

➢ 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

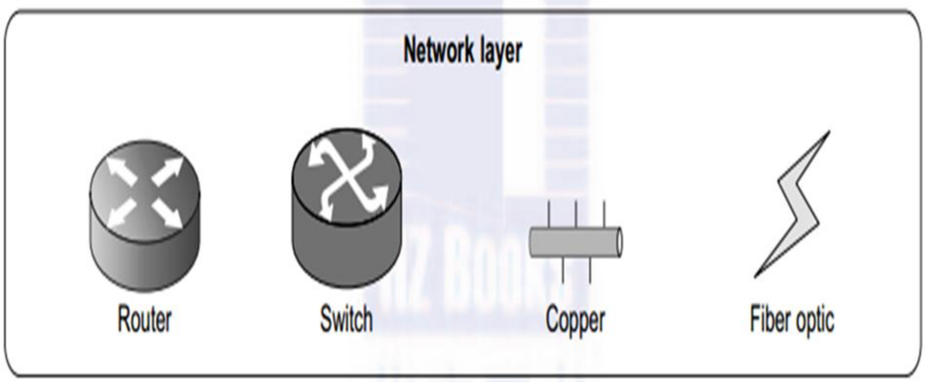


FIGURE 1.26

Four operational layers of distributed computing systems.

➢ 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.

- As information resources drive economic and social development, data centers become increasingly important in terms of where the information items are stored and processed, and where services are provided.
- The relationship between energy and voltage frequency in CMOS circuits is related by:

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

where v, C_{eff}, K, and v_t are the voltage, circuit switching capacity, a technology dependent factor, and threshold voltage, respectively, and the parameter t is the execution time of the task under clock frequency f. By reducing voltage and frequency, the device's energy consumption can also be reduced.

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.

- A WAN can connect many local clusters to form a very large cluster of clusters.
- Many national grids built in the past decade were underutilized for lack of reliable middleware or well-coded applications.
- Potential advantages of cloud computing include its low cost and simplicity for both providers and users.
-

Clusters of Cooperative Computers

- A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource.

Cluster Architecture

- Figure 1.15 shows the architecture of a typical server cluster built around a low-latency, high bandwidth interconnection network.

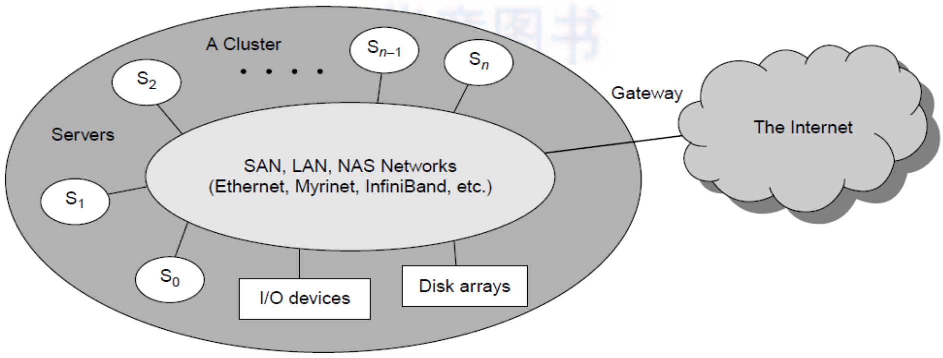


FIGURE 1.15

A cluster of servers interconnected by a high-bandwidth SAN or LAN with shared I/O devices and disk arrays;

Single-System Image

- 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 Top500 list are also MPPs.
- Special cluster middleware supports are needed to create SSI or high availability (HA).

Major 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 levels.

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 webpages.

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

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

Table 1.4 Two Grid Computing Infrastructures and Representative Systems

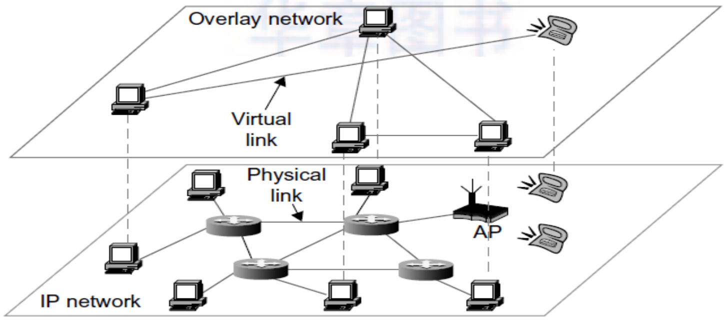
Design Issues	Computational and Data Grids	P2P Grids
Grid Applications Reported	Distributed supercomputing, National Grid initiatives, etc.	Open grid with P2P flexibility, all resources from client machines
Representative Systems	TeraGrid built in US, ChinaGrid in China, and the e-Science grid built 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

Peer-to-Peer Network Families

- The P2P architecture offers a distributed model of networked systems.

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.



RE 1.17
structure of a P2P system by mapping a physical IP network to an overlay network built with virtual

P2P Computing Families

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.

1.3.4 Cloud Computing over the Internet

• Cloud computing has been defined differently by many users and designers. For example, IBM, a major player in cloud computing, has defined it as follows: "A cloud is a pool of virtualized computer resources.

Internet Clouds

- Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically (see Figure 1.18).

The Cloud Landscape

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.

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.

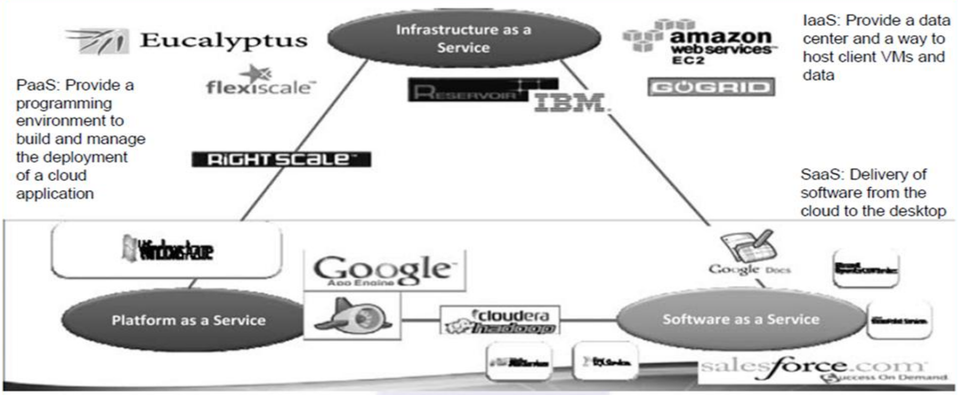


FIGURE 1.19

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

(Courtesy of Dennis Gannon, keynote address at Cloudcom2010 [19])

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

2. Computer Clusters for Scalable Parallel Computing

2.1 Clustering for Massive Parallelism

- A computer cluster is a collection of interconnected stand-alone computers which can work together collectively and cooperatively as a single integrated computing resource pool.

2.1.1 Cluster Development Trends

- Milestone Cluster Systems
- hot research challenge
- fast communication

2.1.2 Design Objectives of Computer Clusters

- Scalability
- Packaging
- Cluster nodes can be packaged in a compact or a slack fashion.

2.1.3 Fundamental Cluster Design Issues

- Scalable Performance
- Single-System Image (SSI)
- Availability Support

- Clusters can provide cost-effective HA capability with lots of redundancy in processors, memory, disks, I/O devices, networks, and operating system images.
- Cluster Job Management
- Internode Communication
- Fault Tolerance and Recovery
- Cluster Family Classification
- Load-balancing clusters
- These clusters shoot for higher resource utilization through load balancing among all participating nodes in the cluster.

2.2.3 Cluster System Interconnects

- High-Bandwidth Interconnects

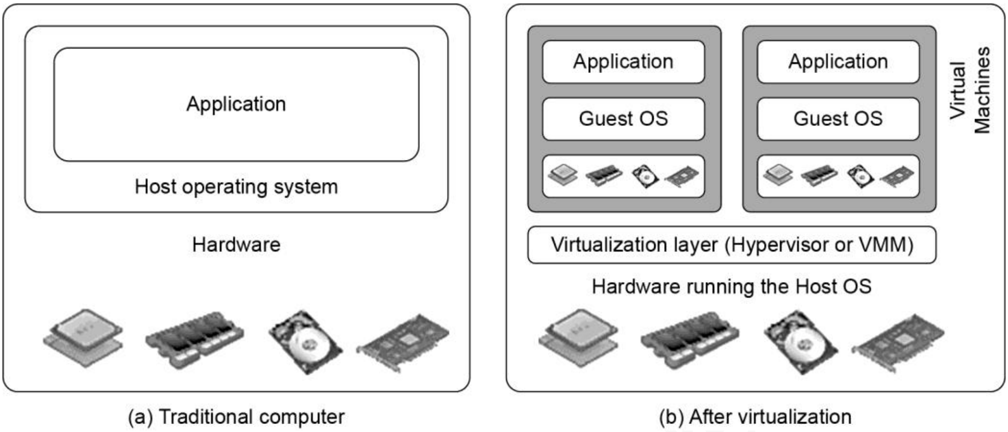
2.2.4 Hardware, Software, and Middleware Support

- The middleware, OS extensions, and hardware support needed to achieve HA in a typical Linux cluster system (Fig. 2.10).

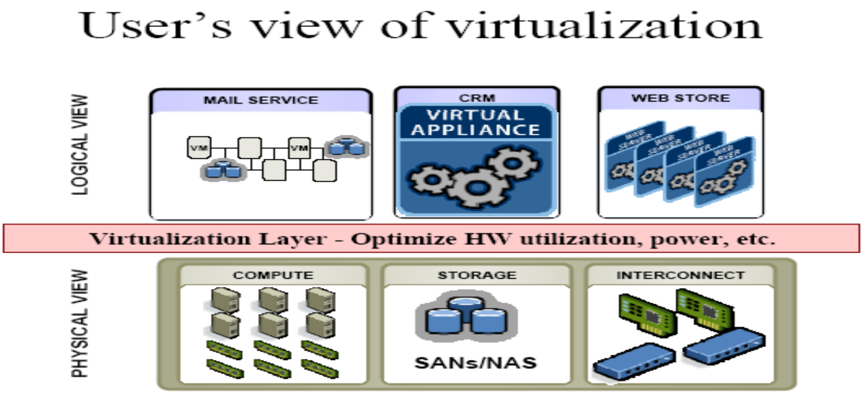
2.2.5 GPU Clusters for Massive Parallelism

- A GPU cluster is often built as a heterogeneous system consisting of three major components: the CPU host nodes, the GPU nodes and the cluster interconnect between them.

Difference between Traditional Computer and Virtual machines



Virtual Machine, Guest Operating System, and VMM (Virtual Machine Monitor) :



Virtualization at ISA (Instruction Set Architecture) level:

Emulating a given ISA by the ISA of the host machine.

- e.g, MIPS binary code can run on an x-86-based host machine with the help of ISA emulation.
- Typical systems: Bochs, Crusoe, Quemu, BIRD, Dynamo Advantage:

Virtualization at OS Level

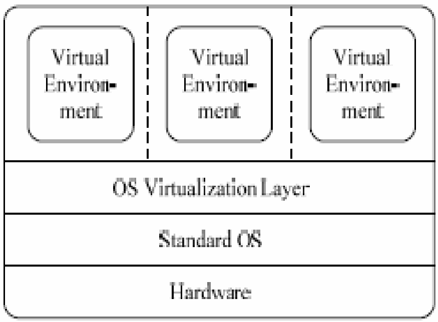


Figure 6.3 The virtualization layer is inserted inside an OS to partition the hardware resources for multiple VMs to run their applications in virtual environments

Virtualization for Linux and Windows NT Platforms

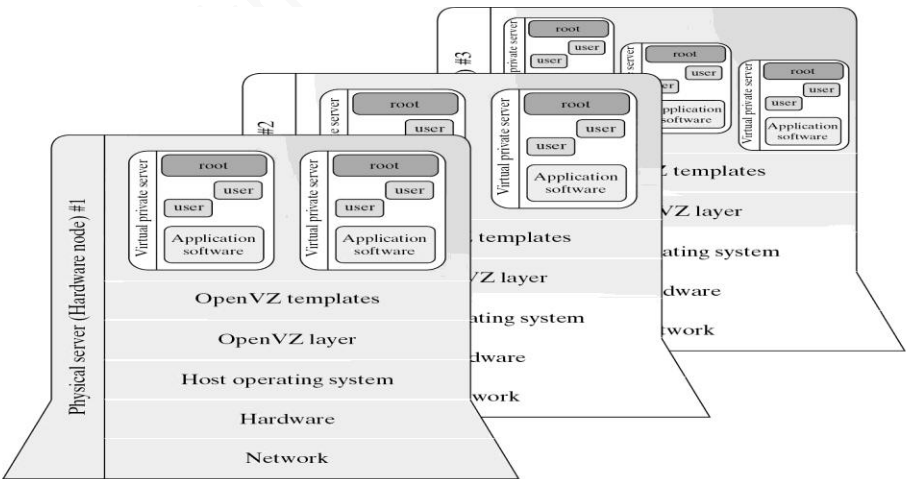


Table 3.3 Virtualization Support for Linux and Windows NT Platforms

Virtualization Support and Source of Information	Brief Introduction on Functionality and Application Platforms
Linux vServer for Linux platforms (http://linux-vserver.org/)	Extends Linux kernels to implement a security mechanism to help build VMs by setting resource limits and file attributes and changing the root environment for VM isolation
OpenVZ for Linux platforms [65]; http://ftp.openvz.org/doc/OpenVZ-Users-Guide.pdf)	Supports virtualization by creating *virtual private servers (VPSes)*; the VPS has its own files, users, process tree, and virtual devices, which can be isolated from other VPSes, and checkpointing and live migration are supported
FVM (Feather-Weight Virtual Machines) for virtualizing the Windows NT platforms [78])	Uses system call interfaces to create VMs at the NY kernel space; multiple VMs are supported by virtualized namespace and copy-on-write

Library Support level*:*

It creates execution environments for running alien programs on a platform rather than creating VM to run the entire operating system.

Shortcoming & limitation:

- poor application flexibility and isolation

Library Support level:

It creates execution environments for running alien programs on a platform rather than creating VM to run the entire operating system.

Advantage:

- It has very low implementation effort

Shortcoming & limitation:

- poor application flexibility and isolation

Virtualization with Middleware/Library Support

Table 3.4 Middleware and Library Support for Virtualization

Middleware or Runtime Library and References or Web Link	Brief Introduction and Application Platforms
WABI (http://docs.sun.com/app/docs/doc/802-6306)	Middleware that converts Windows system calls running on x86 PCs to Solaris system calls running on SPARC workstations
Lxrun (Linux Run) (http://www.ugcs.caltech.edu/~steven/lxrun/)	A system call emulator that enables Linux applications written for x86 hosts to run on UNIX systems such as the SCO OpenServer
WINE (http://www.winehq.org/)	A library support system for virtualizing x86 processors to run Windows applications under Linux, FreeBSD, and Solaris
Visual MainWin (http://www.mainsoft.com/)	A compiler support system to develop Windows applications using Visual Studio to run on Solaris, Linux, and AIX hosts
vCUDA (Example 3.2) (IEEE *IPDPS* 2009 [57])	Virtualization support for using general-purpose GPUs to run data-intensive applications under a special guest OS

User-Application level:

It virtualizes an application as a virtual machine.

- This layer sits as an application program on top of an operating system and exports an abstraction.
-

Table 3.1 Relative Merits of Virtualization at Various Levels

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

Hypervisor

A hypervisor is a hardware virtualization technique allowing multiple operating systems, called guests to run on a host machine. This is also called the Virtual Machine Monitor (VMM).

Major VMM and Hypervisor Providers

VMM Provider	Host CPU	Guest CPU	Host OS	Guest OS	VM Architecture
VMware Work-station	X86, x86-64	X86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, Netware, OS/2, SCO, BeOS, Darwin	Full Virtualization
VMware ESX Server	X86, x86-64	X86, x86-64	No host OS	The same as VMware workstation	Para-Virtualization
XEN	X86, x86-64, IA-64	X86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, windows XP and 2003 Server	Hypervisor
KVM	X86, x86-64, IA64, S390, PowerPC	X86, x86-64, IA64, S390, PowerPC	Linux	Linux, Windows, FreeBSD, Solaris	Para-Virtualization

The XEN Architecture (1)

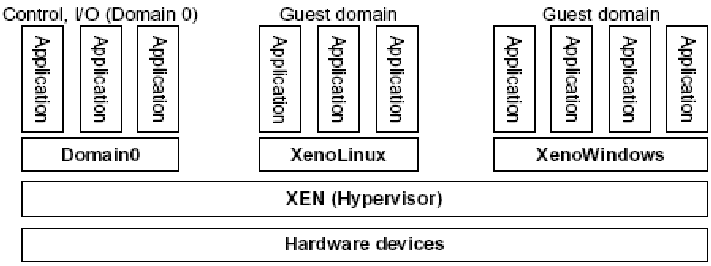


FIGURE 3.5
The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

The XEN Architecture (2)

The XEN Architecture (3)

Full virtualization

• Does not need to modify guest OS, and critical instructions are emulated by software through the use of binary translation.

Para virtualization

• Reduces the overhead, but cost of maintaining a paravirtualized OS is high.

Full Virtualization

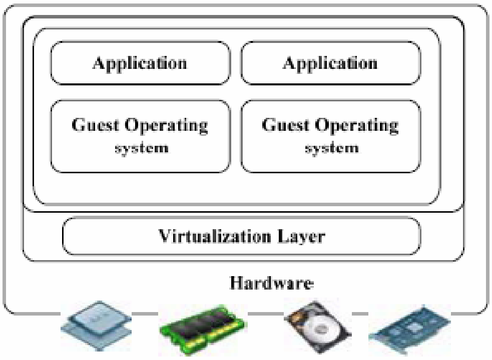


Figure 6.9 The concept of full virtualization using a hypervisor or a VMM directly sitting on top of the bare hardware devices. Note that no host OS is used here as in Figure 6.11.

Binary Translation of Guest OS Requests using a VMM:

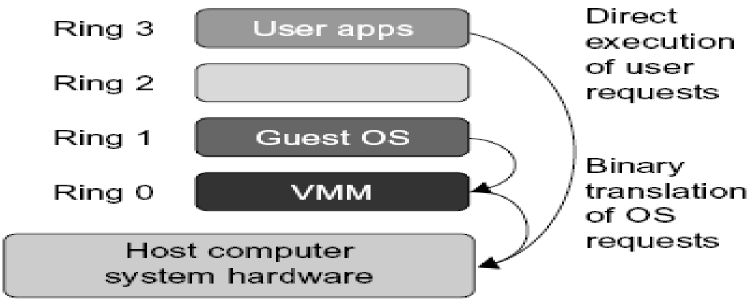
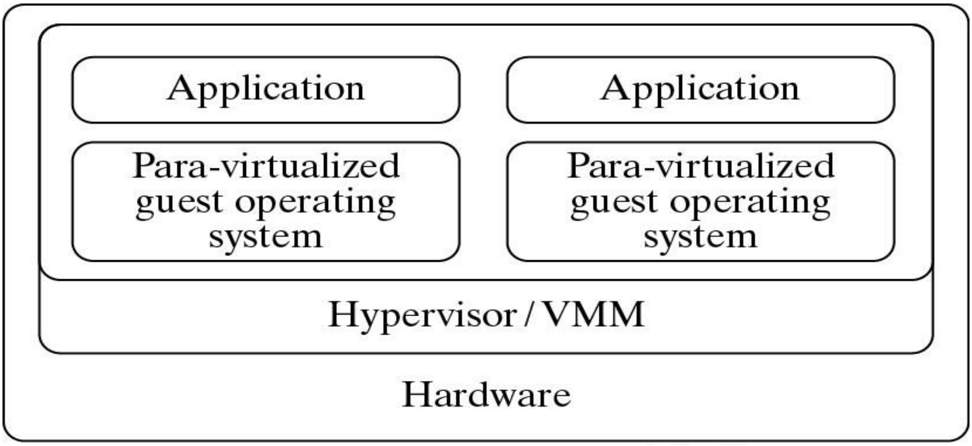


FIGURE 3.6

Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simple instructions on the same host.

Para- Virtualization with Compiler Support.



The KVM builds offers kernel-based VM on the Linux platform, based on para-virtualization

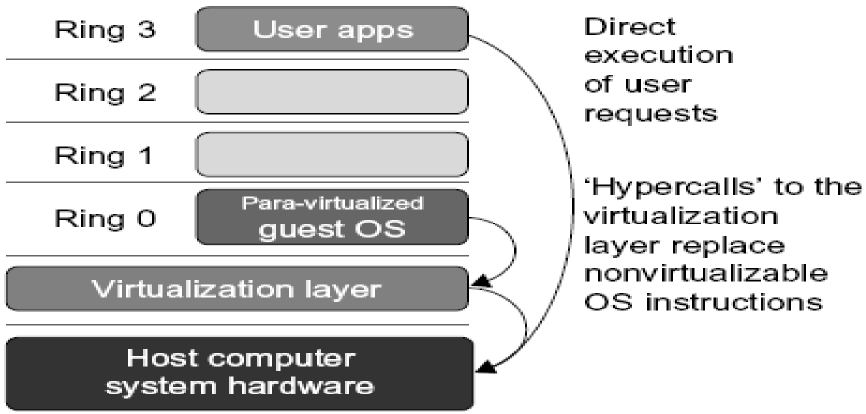
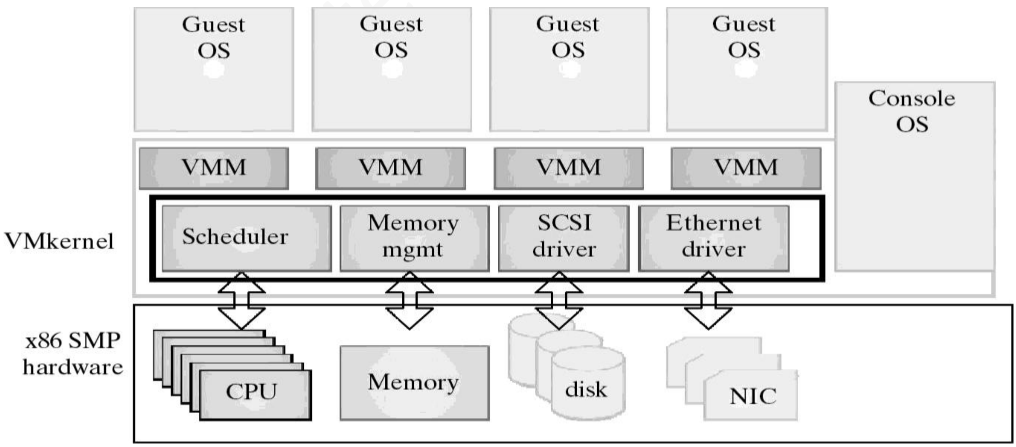
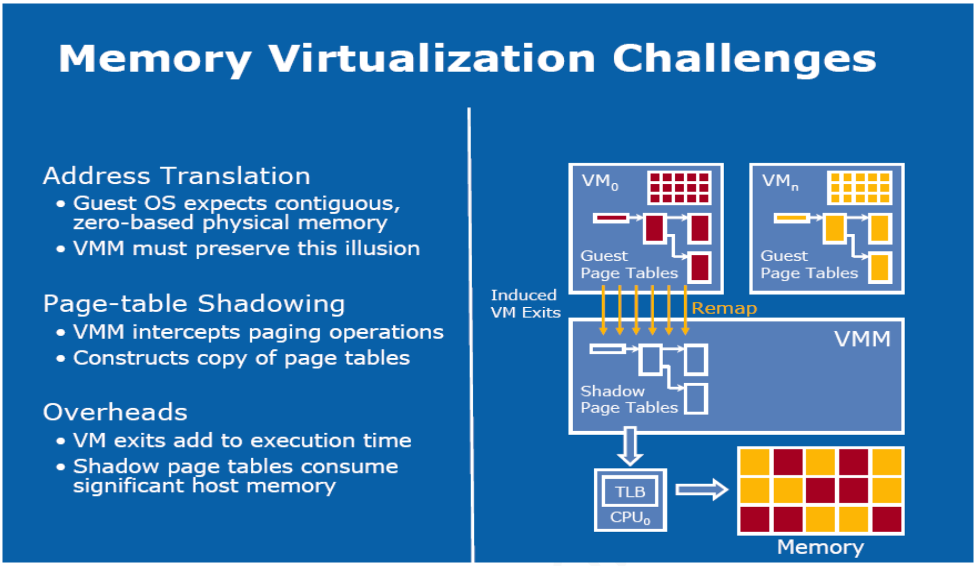


FIGURE 3.8

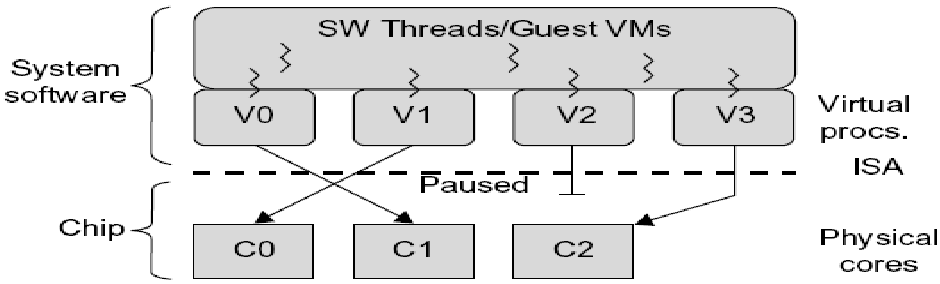
The Use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

VMWare ESX Server for Para-Virtualization



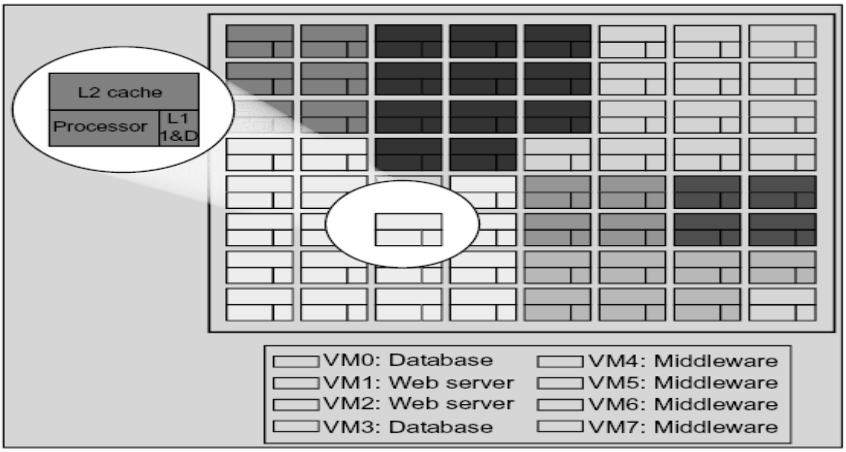


Multi-Core Virtualization: VCPU vs. traditional CPU



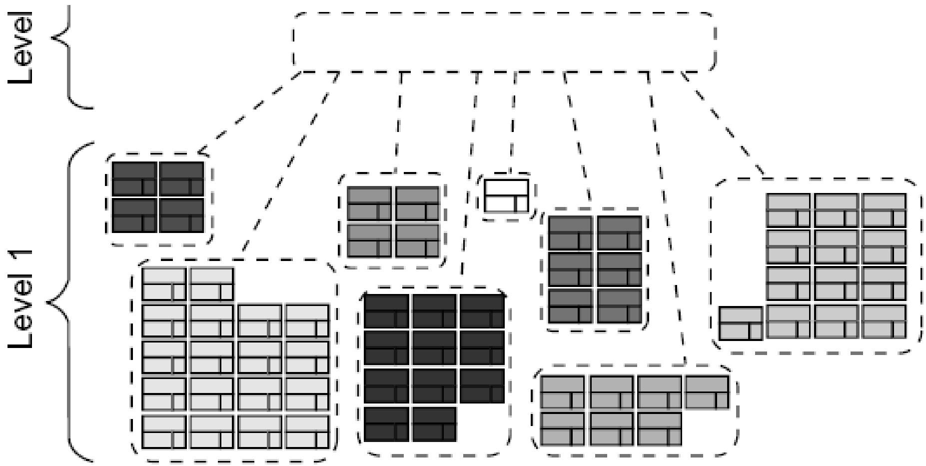
Physical cores	Virtual cores
The actual physical cores present in the processor.	There can be more virtual cores visible to a single OS than there are physical cores.

More burden on the software to write applications which can run directly on the cores.	Design of software becomes easier as the hardware assists the software in dynamic resource utilization.
Hardware provides no assistance to the software and is hence simpler.	Hardware provides assistance to the software and is hence more complex.
Poor resource management.	Better resource management.
The lowest level of system software has to be modified.	The lowest level of system software need not be modified.



(a) Mapping of VMs into adjacent cores

Virtual Clusters in Many CoresSpace Sharing of VMs -- Virtual Hierarchy



(b) Multiple virtual clusters assigned to various workloads

Virtual Cluster Characteristics

- The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSs can be deployed on the same physical node.

Virtual Clusters vs. Physical Clusters

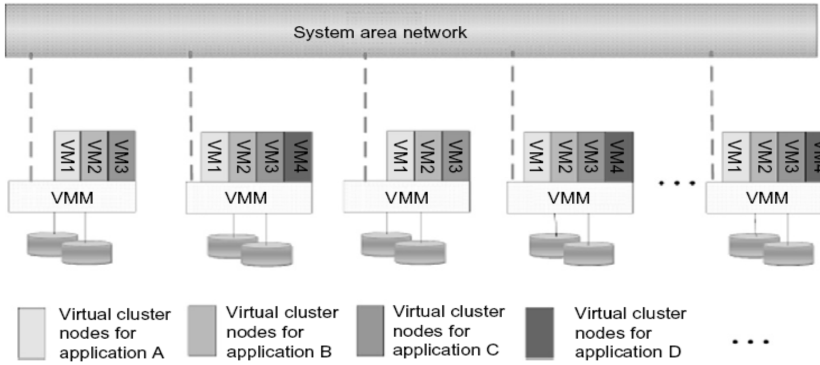


FIGURE 3.19

The concept of a virtual cluster based on application partitioning.

(*Courtesy of Kang, Chen, Tsinghua University 2008*)

Live Migration of Virtual Machines

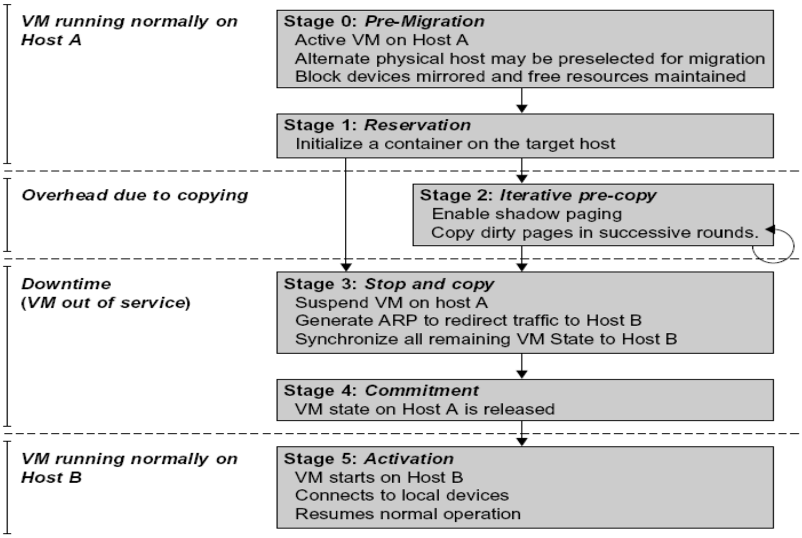


FIGURE 3.20

Live migration process of a VM from one host to another.

(*Courtesy of C. Clark, et al. [14]*)

Virtual Cluster Projects

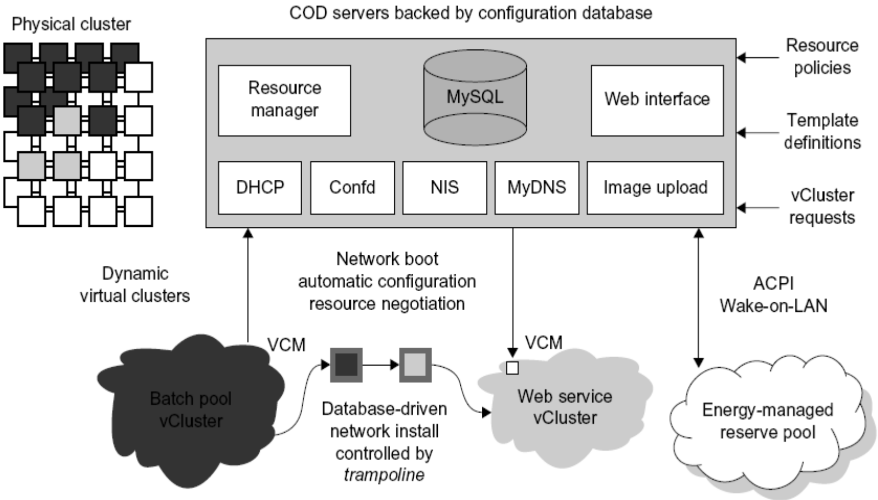


FIGURE 3.23

COD partitioning a physical cluster into multiple virtual clusters.

(*Courtesy of Jeff Chase, et al, HPDC-2003 [12]*)

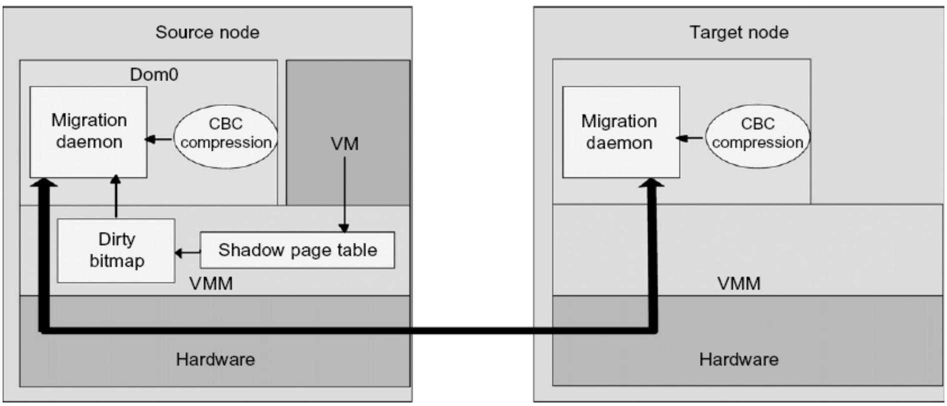


FIGURE 3.22

Live migration of VM from the Dom0 domain to a Xen-enabled target host.

VIOLIN Project at Purdue University

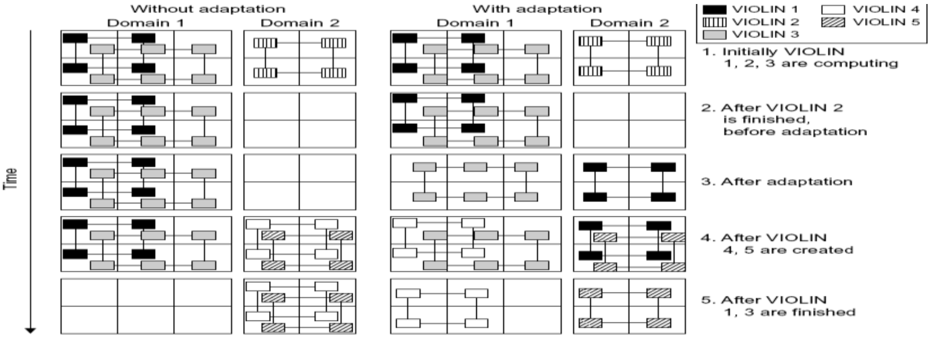


FIGURE 3.25

VIOLIN adaptation scenario of five virtual environments sharing two hosted clusters; Note that there are more idle squares (blank nodes) before and after the adaptation.

(*Courtesy of P. Ruth, et al. [24,51]*)

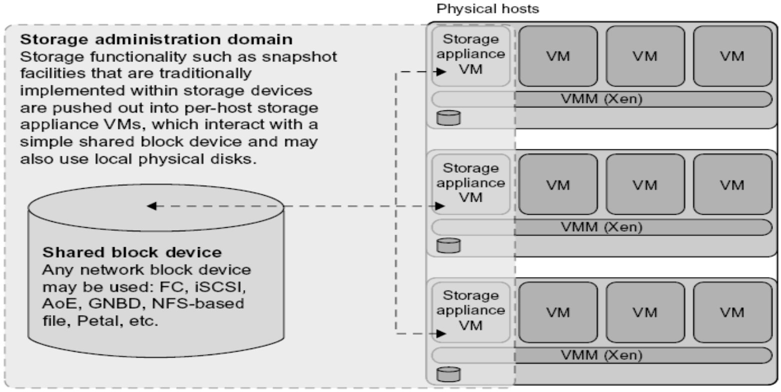


FIGURE 3.26

Parallax is a set of per-host storage appliances that share access to a common block device and presents virtual disks to client VMs.

(*Courtesy of D. Meyer, et al. [43]*)

Eucalyptus : An Open-Source OS for Setting Up and Managing Private Clouds

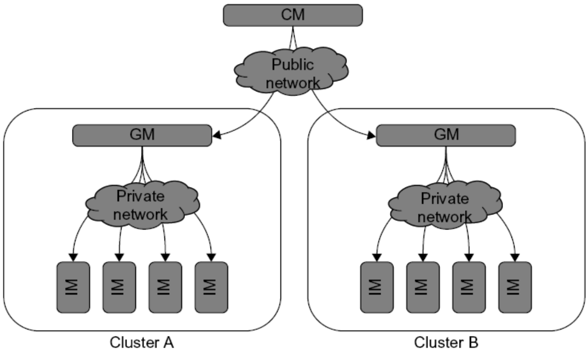
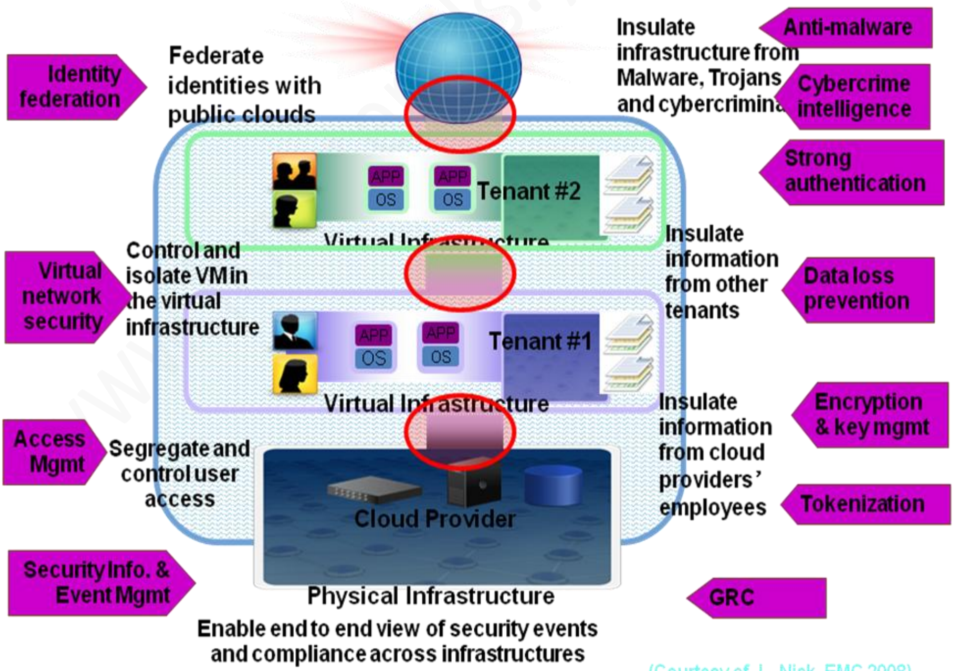


FIGURE 3.27

Eucalyptus for building private clouds by establishing virtual networks over the VMs linking through Ethernet and the Internet.

(*Courtesy of D. Nurmi, et al. [45]*)

Trusted Zones for VM Insulation



(Courtesy of L. Nick, EMC 2008)