Module-1
Distributed System Models and Enabling Technologies: 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. **Textbook1: Chapter1:1.1 to 1.5**

EVOLUTION OF DISTRIBUTED COMPUTING

- Grids enable access to shared computing power and storage capacity from your desktop.
- Clouds enable access to leased computing power and storage capacity from your desktop.
- Grids are an **open source**technology. Resource users and providers alike can understand and contribute to the management of their grid
- Clouds are a **proprietary** technology. Only the resource provider knows exactly how their cloud manages data, job queues, security requirements and so on.
- The concept of grids was proposed in 1995. The Open science grid (OSG) started in 1995 The EDG (European Data Grid) project began in 2001.
- In the late 1990`s Oracle and EMC offered early private cloud solutions .However the term cloud computing didn't gain prominence until 2007. o 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
- We have to upgrade data centers using fast servers, storage systems, and high-bandwidth networks.
- From 1950 to 1970, a handful of mainframes, including the IBM 360 and CDC 6400

1.1 SCALABLE COMPUTING OVER THE INTERNET

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.

The Age of Internet Computing

<u>The Platform Evolution</u>

o From 1960 to 1980, lower-cost minicomputers such as the DEC PDP 11 and VAX Series

o From 1970 to 1990, we saw widespread use of personal computers built with VLSI microprocessors.

o From 1980 to 2000, massive numbers of portable computers and pervasive devices appeared in both wired and wireless applications

o Since 1990, the use of both HPC and HTC systems hidden in clusters, grids, or Internet clouds has proliferated

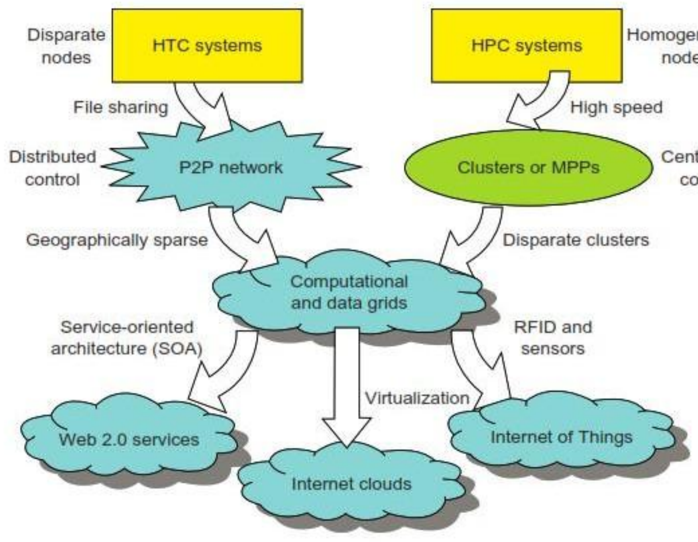


FIGURE 1.1

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

High-Performance Computing (HPC) and High-Throughput Computing (HTC) have evolved significantly, driven by advances in **clustering, P2P networks, and cloud computing**.

- **HPC Evolution**:
  - Traditional **supercomputers (MPPs)** are being replaced by **clusters of cooperative computers** for better resource sharing.
  - HPC has focused on **raw speed performance**, progressing from **Gflops (1990s) to Pflops (2010s)**.
- **HTC and P2P Networks**:
  - HTC systems prioritize **high-flux computing**, emphasizing **task throughput** over raw speed.
  - **P2P networks** facilitate **distributed file sharing and content delivery** using globally distributed client machines.
  - **HTC applications** dominate areas like **Internet searches and web services** for millions of users.
- **Market Shift from HPC to HTC**:
  - HTC systems address challenges beyond speed, including **cost, energy efficiency, security, and reliability**.
- **Emerging Paradigms**:
  - Advances in **virtualization** have led to the rise of **Internet clouds**, enabling **service-oriented computing**.
  - Technologies like **RFID, GPS, and sensors** are fueling the growth of the **Internet of Things (IoT)**.
- **Computing Model Overlaps**:
  - **Distributed computing** contrasts with **centralized computing**.
  - **Parallel computing** shares concepts with **distributed computing**.
  - **Cloud computing** integrates aspects of **distributed, centralized, and parallel computing**.

The transition from **HPC to HTC** marks a strategic shift in computing paradigms, focusing on **scalability, efficiency, and real-world usability** over pure processing power.

Computing Paradigm Distinctions

Centralized computing

A computing paradigm where all computer resources are centralized in a single physical system. In this setup, processors, memory, and storage are fully shared and tightly integrated within one operating system. Many data centers and supercomputers operate as centralized systems, but they are also utilized in parallel, distributed, and cloud computing applications.

• **Parallel computing**

In parallel computing, processors are either tightly coupled with shared memory or loosely coupled with distributed memory. Communication occurs through shared memory or message passing. A system that performs parallel computing is a parallel computer, and the programs running on it are called parallel programs. Writing these programs is referred to as parallel programming.

• **Distributed computing** studies distributed systems, which consist of multiple autonomous computers with private memory communicating through a network via message passing. Programs running in such systems are called distributed programs, and writing them is known as distributed programming.

Cloud computing refers to a system of Internet-based resources that can be either centralized or distributed. It uses parallel, distributed computing, or both, and can be established with physical or virtualized resources over large data centers. Some regard cloud computing as a form of utility computing or service computing. Alternatively, terms such as concurrent computing or concurrent programming are used within the high-tech community, typically referring to the combination of parallel and distributed computing, although interpretations may vary among practitioners.

• **Ubiquitous computing** refers to computing with pervasive devices at any place and time using wired or wireless communication. The Internet of Things (IoT) is a networked connection of everyday objects including computers, sensors, humans, etc. The IoT is supported by Internet clouds to achieve ubiquitous computing with any object at any place and time. Finally, the term Internet computing is even broader and covers all computing paradigms over the Internet. This book covers all the aforementioned computing paradigms, placing more emphasis on distributed and cloud computing and their working systems, including the clusters, grids, P2P, and cloud systems.

Internet of Things The traditional Internet connects machines to machines or web pages to web pages. 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.

• It allows objects to be sensed and controlled remotely across existing network infrastructure

Distributed System Families

Massively distributed systems, including **grids, clouds, and P2P networks**, focus on **resource sharing** in hardware, software, and datasets. These systems emphasize **parallelism** and **concurrency**, as demonstrated by large-scale infrastructures like the **Tianhe-1A supercomputer** (built in China in 2010 with over 3.2 million cores).

Future **HPC (High-Performance Computing)** and **HTC (High-Throughput Computing)** systems will require **multicore and many-core processors** to support large-scale parallel computing. The effectiveness of these systems is determined by the following key **design objectives**:

1. **Efficiency** – Maximizing resource utilization for HPC and optimizing job throughput, data access, and power efficiency for HTC.
2. **Dependability** – Ensuring reliability, self-management, and **Quality of Service (QoS)**, even in failure conditions.
3. **Adaptability** – Supporting large-scale job requests and virtualized resources across different workload and service models.
4. **Flexibility** – Enabling HPC applications (scientific and engineering) and HTC applications (business and cloud services) to run efficiently in distributed environments.

The future of **distributed computing** depends on **scalable, efficient, and flexible** architectures that can meet the growing demand for computational power, throughput, and energy efficiency.

Scalable Computing Trends and New Paradigms

Scalable computing is driven by technological advancements that enable **high-performance computing (HPC)** and **high-throughput computing (HTC)**. Several trends, such as **Moore's Law** (doubling of processor speed every 18 months) and **Gilder's Law** (doubling of network bandwidth each year), have shaped modern computing. The increasing affordability of **commodity hardware** has also fueled the growth of large-scale distributed systems.

Degrees of Parallelism

Parallelism in computing has evolved from:

- **Bit-Level Parallelism (BLP)** – Transition from serial to word-level processing.
- **Instruction-Level Parallelism (ILP)** – Executing multiple instructions simultaneously (pipelining, superscalar computing).
- **Data-Level Parallelism (DLP)** – SIMD (Single Instruction, Multiple Data) architectures.
- **Task-Level Parallelism (TLP)** – Parallel execution of independent tasks on multicore processors.
- **Job-Level Parallelism (JLP)** – Large-scale distributed job execution in cloud computing.

Coarse-grained parallelism builds on fine-grained parallelism, ensuring scalability in **HPC and HTC systems**.

Innovative Applications of Distributed Systems

Parallel and distributed systems support applications in various domains:

Domain	Applications
Science & Engineering	Weather forecasting, genomic analysis
Business, education, services industry, and health care	E-commerce, banking, stock exchanges
Internet and web services, and government applications	Cybersecurity, digital governance, traffic monitoring
Mission-Critical Systems	Military, crisis management

HTC systems prioritize **task throughput** over raw speed, addressing challenges like **cost, energy efficiency, security, and reliability**.

The Shift Toward Utility Computing

Utility computing follows a **pay-per-use** model where computing resources are delivered as a service. Cloud computing extends this concept, allowing **distributed applications to run on edge networks**.

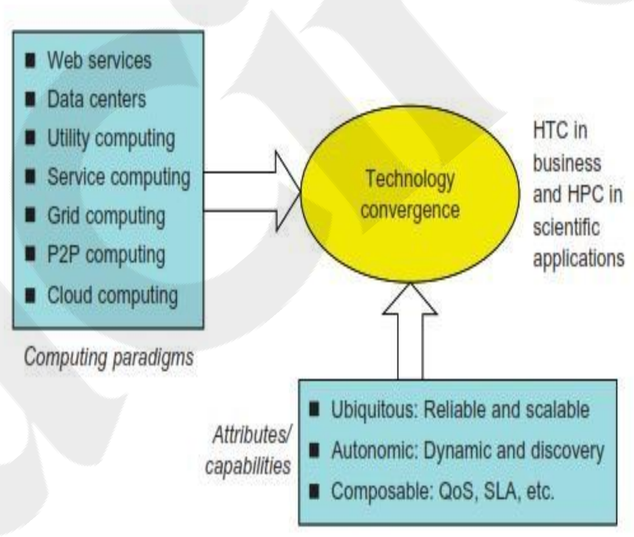


FIGURE 1.2

Challenges include:

- Efficient **network processors**
- Scalable **storage and memory**
- **Virtualization middleware**
- New **programming models**

The Hype Cycle of Emerging Technologies

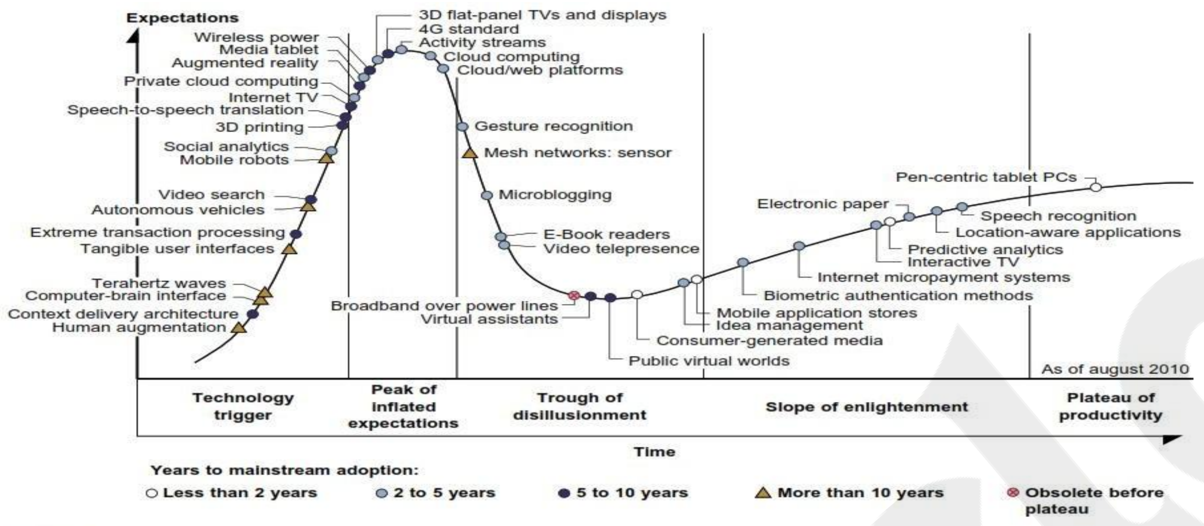


FIGURE 1.3
Hype cycle for Emerging Technologies, 2010.

New technologies follow a **hype cycle**, progressing through:

1. **Technology Trigger** – Early development and research.
2. **Peak of Inflated Expectations** – High expectations but unproven benefits.
3. **Trough of Disillusionment** – Realization of limitations.
4. **Slope of Enlightenment** – Gradual improvements.
5. **Plateau of Productivity** – Mainstream adoption.

For example, in **2010**, **cloud computing** was moving toward mainstream adoption, while **broadband over power lines** was expected to become obsolete.

The Internet of Things (IoT) and Cyber-Physical Systems (CPS)

- **IoT**: Interconnects **everyday objects** (sensors, RFID, GPS) to enable real-time tracking and automation.
- **CPS**: Merges **computation, communication, and control (3C)** to create intelligent systems for **virtual and physical world interactions**.

Both **IoT and CPS** will play a significant role in future **cloud computing and smart infrastructure development**.

1.2 Technologies for Network-Based Systems

Advancements in **multicore CPUs** and **multithreading technologies** have played a crucial role in the development of **high-performance computing (HPC)** and **high-throughput computing (HTC)**.

Advances in CPU Processors

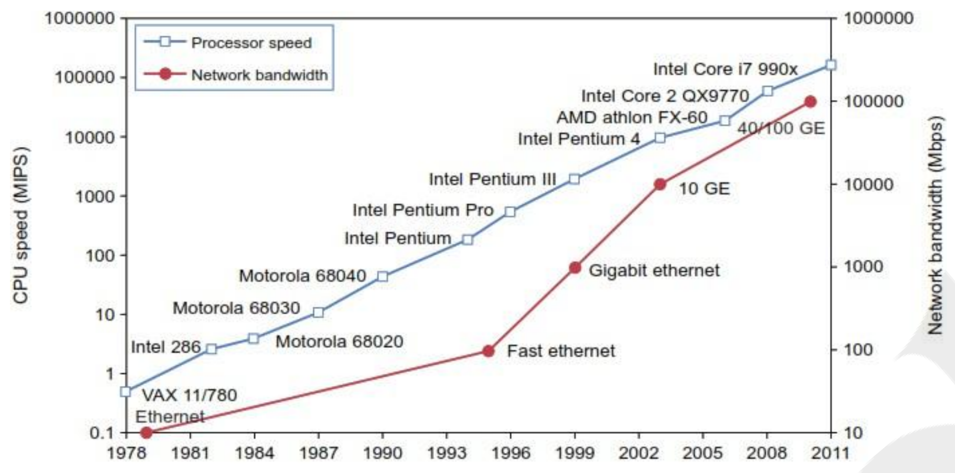


FIGURE 1.4

Improvement in processor and network technologies over 33 years.

- Modern **multicore processors** integrate **dual, quad, six, or more processing cores** to enhance parallelism at the **instruction level (ILP)** and **task level (TLP)**.
- **Processor speed growth** has followed **Moore's Law**, increasing from **1 MIPS (VAX 780, 1978)** to **22,000 MIPS (Sun Niagara 2, 2008)** and **159,000 MIPS (Intel Core i7 990x, 2011)**.
- **Clock rates** have increased from **10 MHz (Intel 286)** to **4 GHz (Pentium 4)** but have **stabilized** due to **heat and power limitations**.

Multicore CPU and Many-Core GPU Architectures

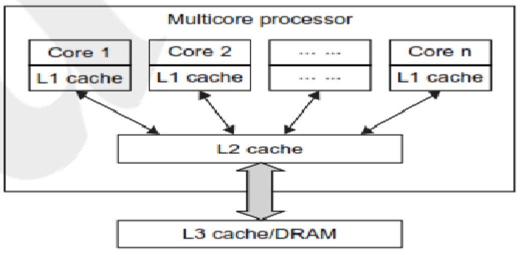


FIGURE 1.5

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 processors** house multiple processing units, each with **private L1 cache** and **shared L2/L3 cache** for efficient data access.
- **Many-core GPUs** (e.g., NVIDIA and AMD architectures) leverage **hundreds to thousands of cores**, excelling in **data-level parallelism (DLP)** and **graphics processing**.
- **Example: Sun Niagara II** – Built with **eight cores**, each supporting **eight threads**, achieving a **maximum parallelism of 64 threads**.

Key Trends in Processor and Network Technology

- **Multicore chips** continue to evolve with improved caching mechanisms and increased processing cores per chip.
- **Network speeds** have improved from **Ethernet (10 Mbps)** to **Gigabit Ethernet (1 Gbps)** and beyond **100 Gbps** to support high-speed data communication.

Modern **distributed computing systems** rely on **scalable multicore architectures** and **high-speed networks** to handle massive parallelism, optimize efficiency, and enhance overall performance.

<u>**Multicore CPU and Many-Core GPU Architectures**</u>

Advancements in **multicore CPUs** and **many-core GPUs** have significantly influenced modern **high-performance computing (HPC)** and **high-throughput computing (HTC)** systems. As CPUs approach their **parallelism limits**, GPUs have emerged as powerful alternatives for **massive parallelism and high computational efficiency**.

<u>**Multicore CPU and Many-Core GPU Trends**</u>

- **Multicore CPUs** continue to evolve from **tens to hundreds of cores**, but they face challenges like the **memory wall problem**, limiting data-level parallelism (DLP).
- **Many-core GPUs**, with **hundreds to thousands of lightweight cores**, excel in DLP and **task-level parallelism (TLP)**, making them ideal for **massively parallel workloads**.
- **Hybrid architectures** are emerging, combining **fat CPU cores and thin GPU cores** on a single chip for optimal performance.

<u>**Multithreading Technologies in Modern CPUs**</u>

- **Different microarchitectures** exploit parallelism at **instruction-level (ILP) and thread-level (TLP)**:
  - **Superscalar Processors** – Execute multiple instructions per cycle.
  - **Fine-Grained Multithreading** – Switches between threads every cycle.
  - **Coarse-Grained Multithreading** – Runs one thread for multiple cycles before switching.
  - **Simultaneous Multithreading (SMT)** – Executes multiple threads in the same cycle.

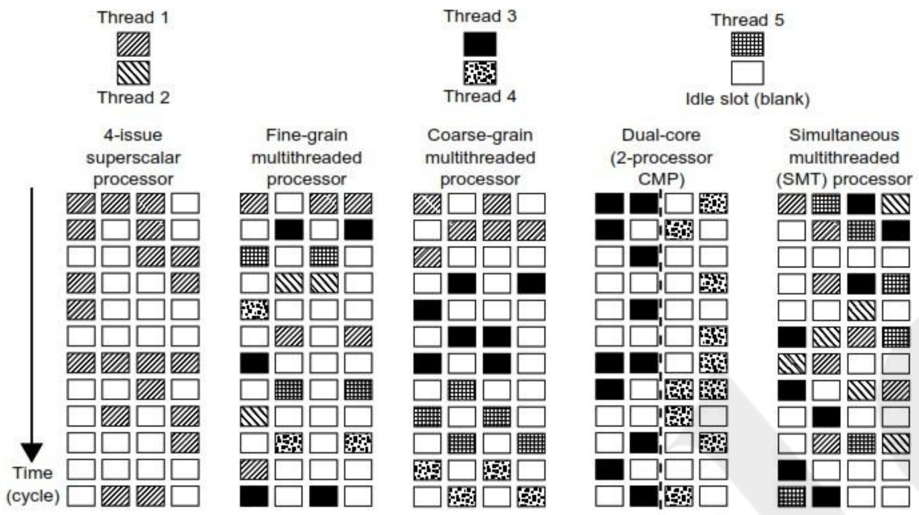

Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Idle slot (blank)
4-issue superscalar processor
Fine-grain multithreaded processor
Coarse-grain multithreaded processor
Dual-core (2-processor CMP)
Simultaneous multithreaded (SMT) processor
Time (cycle)

FIGURE 1.6

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

GPU Computing to Exascale and Beyond

- **GPUs** were initially designed for graphics acceleration but are now used for **general-purpose parallel computing (GPGPU)**.

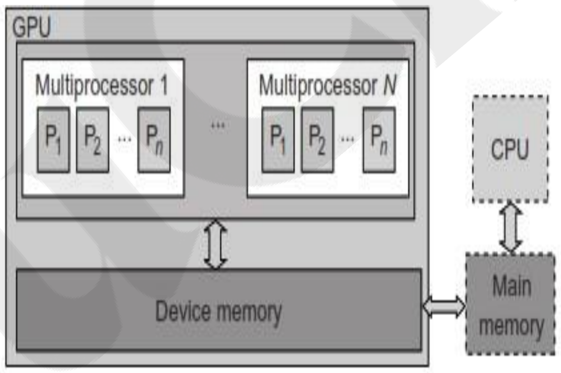


FIGURE 1.7

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

- **Modern GPUs (e.g., NVIDIA CUDA, Tesla, and Fermi)** feature **hundreds of cores**, handling **thousands of concurrent threads**.
- **Example:** The **NVIDIA Fermi GPU** has **512 CUDA cores** and delivers **82.4 teraflops**, contributing to the performance of **top supercomputers like Tianhe-1A**.

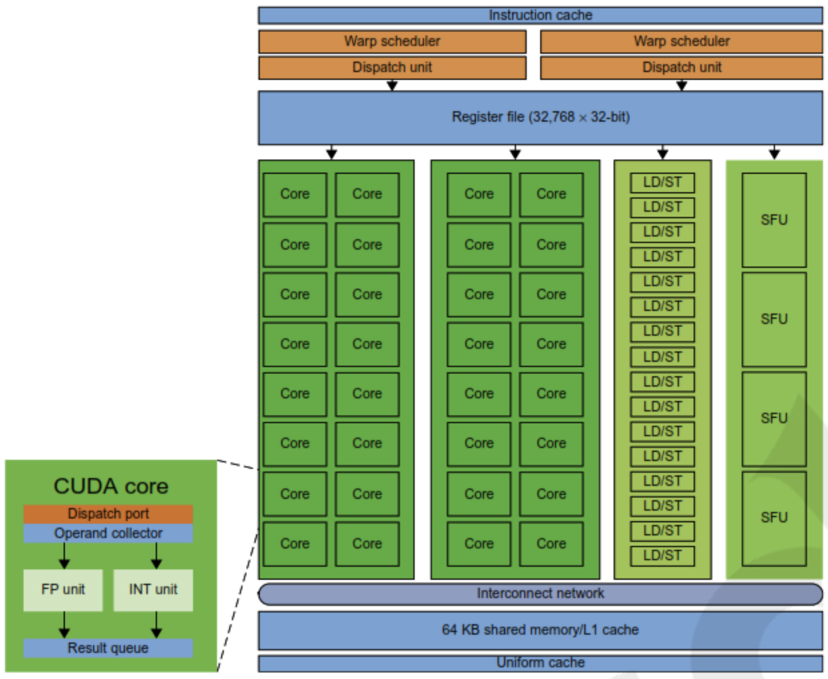


FIGURE 1.8

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].

GPU vs. CPU Performance and Power Efficiency

- **GPUs prioritize throughput**, while **CPUs optimize latency** using cache hierarchies.
- **Power efficiency** is a key advantage of GPUs – GPUs consume **1/10th of the power per instruction** compared to CPUs.
- **Future Exascale Systems** will require **60 Gflops/W per core**, making power efficiency a **major challenge** in **parallel and distributed computing**.

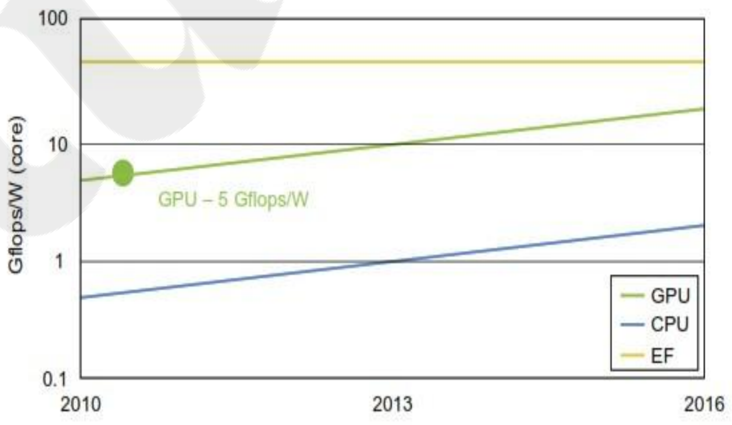


FIGURE 1.9

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.

Challenges in Future Parallel and Distributed Systems

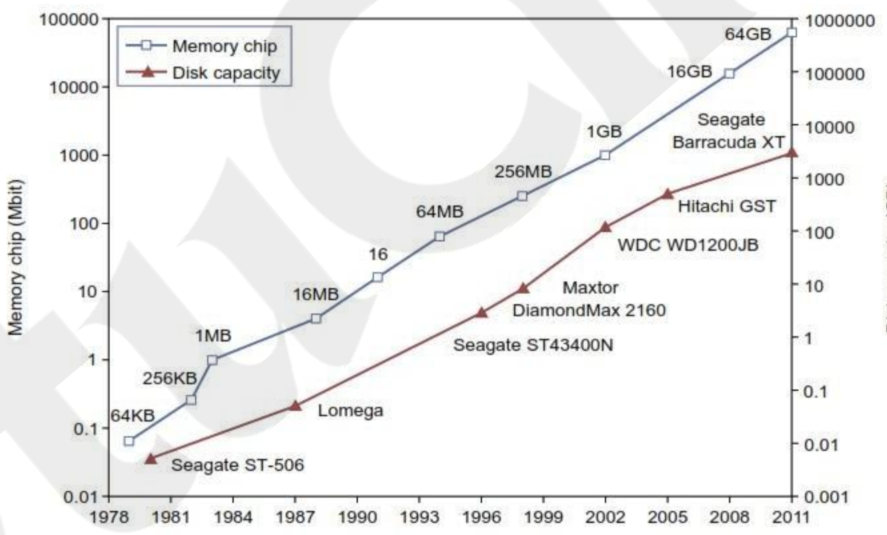
1. **Energy and Power Efficiency** – Reducing power consumption while increasing performance.
2. **Memory and Storage Bottlenecks** – Optimizing data movement to avoid bandwidth limitations.
3. **Concurrency and Locality** – Improving software and compiler support for parallel execution.
4. **System Resiliency** – Ensuring fault tolerance in large-scale computing environments.

The shift towards **hybrid architectures (CPU + GPU)** and the rise of **power-aware computing models** will drive the next generation of **HPC, HTC, and cloud computing systems**.

1.2.3 Memory, Storage, and Wide-Area Networking

Memory Technology

- **DRAM capacity** has increased **4x every three years** (from **16 KB in 1976 to 64 GB in 2011**).
- **Memory access speed has not kept pace**, causing the **memory wall problem**, where CPUs outpace memory access speeds.



•

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

- **Hard drive capacity** has grown **10x every eight years**, reaching **3 TB (Seagate Barracuda XT, 2011)**.
- **Solid-State Drives (SSDs)** provide significant speed improvements and durability (300,000 to 1 million write cycles per block).

- **Power and cooling challenges** limit large-scale storage expansion.

System-Area Interconnects & Wide-Area Networking

- **Local Area Networks (LANs)** connect clients and servers.
- **Storage Area Networks (SANs) & Network Attached Storage (NAS)** support large-scale data storage and retrieval.
- **Ethernet speeds** have evolved from **10 Mbps (1979) to 100 Gbps (2011)**, with **1 Tbps links expected in the future**.
- **High-speed networking** enhances distributed computing efficiency and scalability.

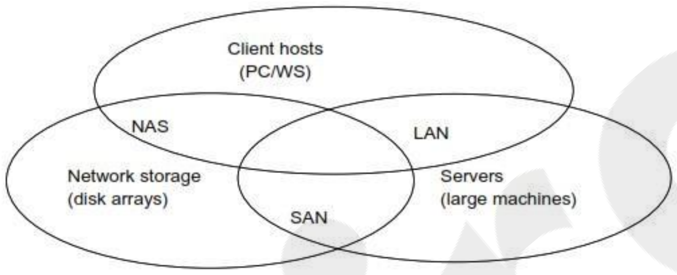


FIGURE 1.11
Three interconnection networks for connecting servers, client hosts, and storage devices; the LAN connects client hosts and servers, the SAN connects servers with disk arrays, and the NAS connects clients with large storage systems in the network environment.

1.2.4 Virtual Machines and Virtualization Middleware

Virtualization in Distributed Systems

- Traditional computing tightly couples OS and hardware, reducing flexibility.
- **Virtual Machines (VMs)** abstract hardware resources, allowing multiple OS instances on a single system.

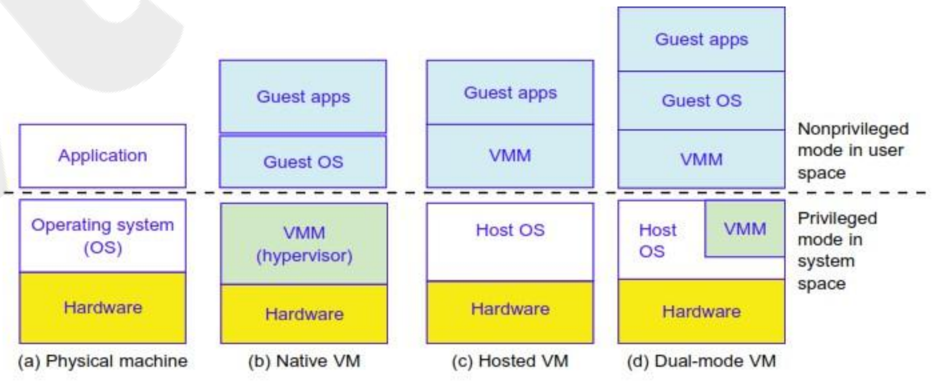


FIGURE 1.12
Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

-

Virtual Machine Architectures

1. **Native VM (Hypervisor-based)** – Direct hardware access via **bare-metal hypervisors** (e.g., VMware ESXi, Xen).
2. **Host VM (Software-based)** – Runs as an application on a host OS (e.g., VirtualBox, VMware Workstation).
3. **Hybrid VM** – Uses a combination of user-mode and privileged-mode virtualization.

Virtual Machine Operations

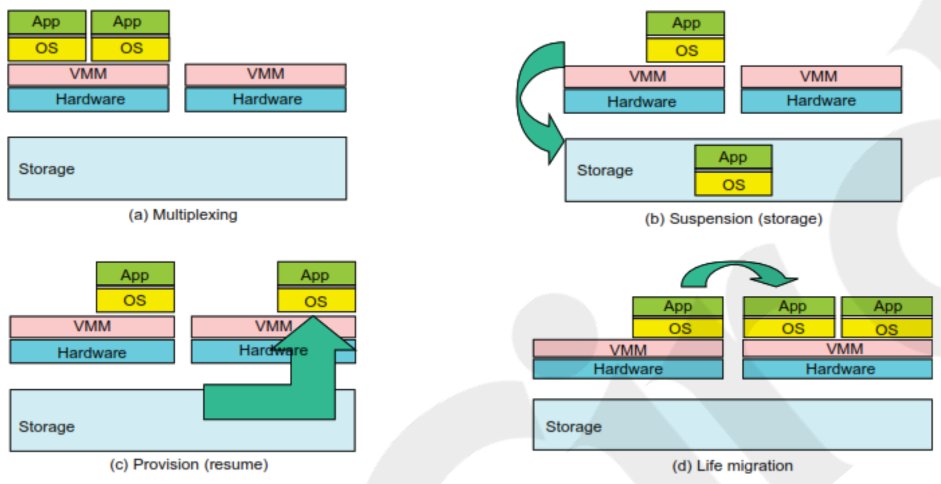


FIGURE 1.13

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

- First,theVMscanbemultiplexedbetweenhardwaremachines,asshowninFigure1.13(a).
- Second,aVMcanbesuspendedandstoredinstablestorage,asshowninFigure1.13(b).
- Third,asuspendedVMcanberesumedorprovisionedtoanewhardware platform,asshowninFigure1.13(c).
- Finally,aVMcan bemigratedfromone hardwareplatformtoanother,as shown inFigure1.13(d).

- **Multiplexing** – Multiple VMs share physical resources.
- **Suspension & Migration** – VMs can be paused, saved, or migrated across different servers.
- **Provisioning** – VMs can be dynamically deployed based on workload demand.

Virtual Infrastructure

- **Separates physical hardware from applications**, enabling flexible resource management.
- **Enhances server utilization from 5–15% to 60–80%** (as claimed by VMware).

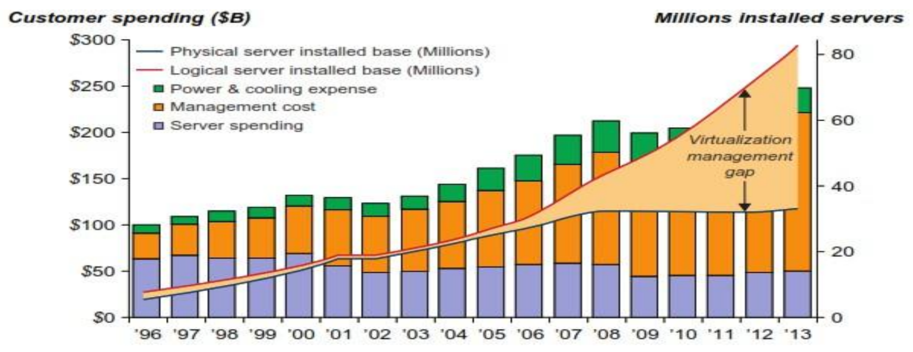


FIGURE 1.14
Growth and cost breakdown of data centers over the years.

1.2.5 Data Center Virtualization for Cloud Computing

Data Center Growth and Cost Breakdown

- **43 million servers worldwide (2010)**, with **utilities (power & cooling) exceeding hardware costs** after three years.
- **60% of data center costs** go toward **maintenance and management**, emphasizing energy efficiency over raw performance.

Low-Cost Design Philosophy

- **Commodity x86 servers & Ethernet** replace expensive **mainframes & proprietary networking hardware**.
- **Software handles fault tolerance, load balancing, and scalability**, reducing infrastructure costs.

Convergence of Technologies Enabling Cloud Computing

1. **Virtualization & Multi-core Processors** – Enable scalable computing.
2. **Utility & Grid Computing** – Provide a foundation for cloud computing.
3. **SOA, Web 2.0, and Mashups** – Facilitate cloud-based service integration.
4. **Autonomic Computing & Data Center Automation** – Improve efficiency and fault tolerance.

The Rise of Data-Intensive Computing

- **Scientific research, business, and web applications** generate vast amounts of data.
- **Cloud computing & parallel computing** address the **data deluge** challenge.
- **MapReduce & Iterative MapReduce** enable scalable data processing for **big data and machine learning applications**.
- The convergence of **data-intensive computing, cloud platforms, and multicore architectures** is shaping the **next generation of distributed computing**.

The **integration of memory, storage, networking, virtualization, and cloud data centers** is transforming distributed systems. By leveraging **virtualization, scalable networking, and cloud computing**, modern infrastructures achieve **higher efficiency, flexibility, and cost-effectiveness**, paving the way for **future exascale computing**.

1.3 SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

• **Distributed and cloud computing systems** are built using large-scale, interconnected **autonomous computer nodes**. These nodes are linked through **Storage Area Networks (SANs), Local Area Networks (LANs), or Wide Area Networks (WANs)** in a hierarchical manner.

- **Clusters**: Connected by **LAN switches**, forming tightly coupled systems with **hundreds of machines**.
- **Grids**: Interconnect **multiple clusters via WANs**, allowing resource sharing across **thousands of computers**.
- **P2P Networks**: Form **decentralized, cooperative networks** with **millions of nodes**, used in file sharing and content distribution.
- **Cloud Computing**: Operates **over massive data centers**, delivering **on-demand computing resources** at a global scale.

These systems exhibit **high scalability**, enabling **web-scale computing** with **millions of interconnected nodes**. Their technical and application characteristics vary based on factors such as **resource sharing, control mechanisms, and workload distribution**.

Functionality, Applications	**Computer Clusters** [10,28,38]	**Peer-to-Peer Networks** [34,46]	**Data/ Computational Grids** [6,18,51]	**CloudPlatforms** [1,9,11,12,30]
Architecture, Network Connectivity,and Size	Networkof computenodes interconnected bySAN,LAN,or WAN hierarchically	Flexiblenetwork ofclientmachines logically connectedbyan overlaynetwork	Heterogeneous clusters interconnectedby high-speed networklinksover selectedresource sites	Virtualizedcluster ofservers overdata centersvia SLA
Controland Resources Management	Homogeneous nodes withdistributed control,running UNIXorLinux	Autonomous clientnodes,free inandout,with self-organization	Centralized control,server-orientedwith authenticated security	Dynamicresource provisioningof servers, storage, andnetworks
Applicationsand Network-centric Services	High-performance computing, search engines, andwebservices, etc.	Mostappealing tobusinessfile sharing,content delivery,and socialnetworking	Distributed supercomputing, globalproblem solving,anddata centerservices	Upgradedweb search,utility computing, andoutsourced Mostappealing to businessfile sharing,content delivery,and socialnetworking computing services
Representative Operational Systems	Googlesearch engine,SunBlade, IBMRoad Runner,Cray XT4,etc.	Gnutella,eMule, BitTorrent, Napster, KaZaA, Skype, JXTA	TeraGrid, GriPhyN,UK EGEE,D-Grid, ChinaGrid,etc	GoogleApp Engine,IBM Bluecloud,AWS, andMicrosoft Gnutella,eMule, BitTorrent, Napster, KaZaA, Skype, JXTA

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

Server Clusters and System Models for Distributed Computing

1.3.1 Server Clusters and Interconnection Networks

Server clusters consist of **multiple interconnected computers** using **high-bandwidth, low-latency networks** like **Storage Area Networks (SANs), Local Area Networks (LANs), and InfiniBand**. These clusters are **scalable**, allowing thousands of nodes to be connected hierarchically.

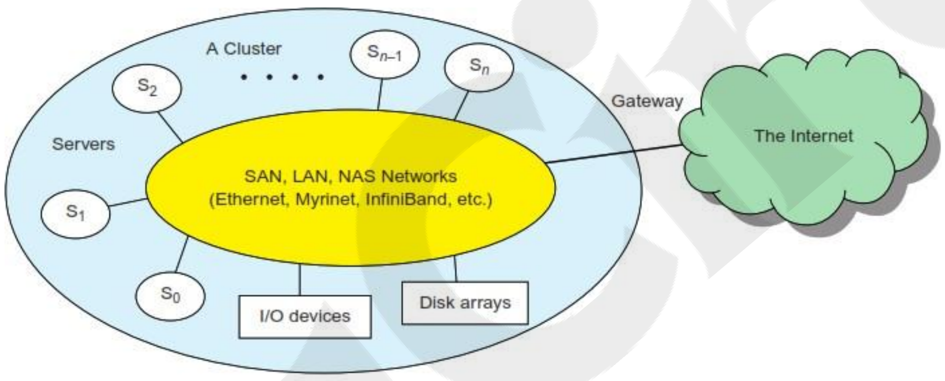


FIGURE 1.15

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.

- **Clusters are connected to the Internet via a VPN gateway**, which assigns an IP address to locate the cluster.
- Each **node operates independently**, with its own OS, creating **multiple system images (MSI)**.
- The cluster manages shared **I/O devices and disk arrays**, providing efficient resource utilization.

1.3.1.2 Single-System Image (SSI)

An **ideal cluster** should merge multiple system images into a **single-system image (SSI)**, where all nodes appear as a **single powerful machine**.

- **SSI is achieved through middleware or specialized OS support**, enabling **CPU, memory, and I/O sharing** across all cluster nodes.

- Clusters without SSI function as a **collection of independent computers** rather than a unified system.

1.3.1.3 Hardware, Software, and Middleware Support

- **Cluster nodes** consist of **PCs, workstations, or servers**, interconnected using **Gigabit Ethernet, Myrinet, or InfiniBand**.
- **Linux OS** is commonly used for cluster management.
- **Message-passing interfaces (MPI, PVM)** enable parallel execution across nodes.
- **Middleware** supports features like **high availability (HA)**, distributed memory sharing (**DSM**), and job scheduling.
- **Virtual clusters** can be dynamically created using **virtualization**, optimizing resource allocation on demand.

1.3.1.4 Major Cluster Design Issues

Features	FunctionalCharacterization	FeasibleImplementations
AvailabilityandSupport	Hardware andsoftware support for sustainedHAincluster	Failover,failback,check pointing, rollbackrecovery,nonstopOS,etc.
Hardware FaultTolerance	Automated failuremanagementto eliminateallsinglepointsoffailure	Componentredundancy,hot swapping, RAID,multiplepower supplies, etc.
SingleSystem Image(SSI)	AchievingSSIatfunctionallevelwith hardware andsoftware support, middleware,orOSextensions	Hardware mechanismsor middlewaresupport toachieveDSM atcoherent cache level
EfficientCommunications	Toreduce message-passingsystem overhead andhidelatencies	Fastmessagepassing, activemessages,enhancedMPIlibrary,e
Cluster-wideJob Management	Usingaglobaljobmanagement system withbetter scheduling and monitoring	Applicationofsingle-job managementsystemssuch asLSF, Codine,etc.
DynamicLoadBalancing	Balancingtheworkloadofall processingnodes alongwithfailure recovery	Workloadmonitoring,process migration,jobreplicationandgang scheduling, etc.
Scalabilityand Programmability	Addingmoreservers toaclusteror adding moreclusters toagridas theworkloadordata setincreases	Useofscalable interconnect, performancemonitoring,distributed executionenvironment,andbetter software tools

- **Lack of a cluster-wide OS** limits full resource sharing.
- **Middleware solutions** provide necessary functionalities like **scalability, fault tolerance, and job management**.
- Key challenges include **efficient message passing, seamless fault tolerance, high availability, and performance scalability**.

Server clusters are **scalable, high-performance computing systems** that utilize **networked computing nodes** for **parallel and distributed processing**. Achieving **SSI and efficient middleware support** remains a key challenge in cluster computing. **Virtual clusters** and **cloud computing** are evolving to enhance cluster flexibility and resource management.

1.3.2 Grid Computing, Peer-to-Peer (P2P) Networks, and System Models

Grid Computing Infrastructures

Grid computing has evolved from **Internet and web-based services** to enable **large-scale distributed computing**. It allows applications running on remote systems to interact in real-time.

1.3.2.1 Computational Grids

- A grid **connects distributed computing resources** (workstations, servers, clusters, supercomputers) over **LANs, WANs, and the Internet**.

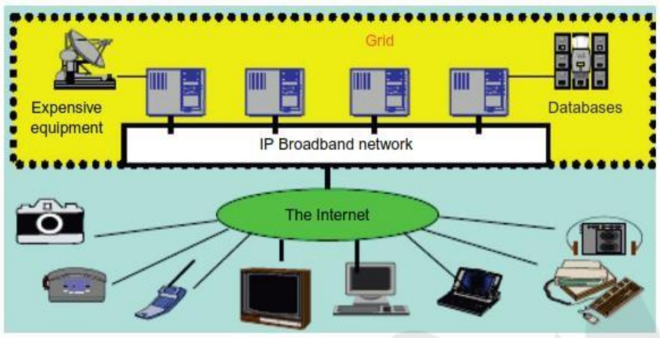


FIGURE 1.16
Computational grid or data grid providing computing utility, data, and information services through resource sharing and cooperation among participating organizations.

- Used for **scientific and enterprise applications**, including **SETI@Home and astrophysics simulations**.
- Provides an **integrated resource pool**, enabling shared computing, data, and information services.

1.3.2.2 Grid Families

DesignIssues	ComputationalandDataGrids	P2P Grids
GridApplicationsReported	Distributedsupercomputing, NationalGridinitiatives,etc.	Open gridwithP2P flexibility,all resourcesfromclientmachines
RepresentativeSystems	TeraGridbuiltinUS,ChinaGridin China,andthee-Sciencegrid builtin UK	JXTA,FightAid@home,SE TI@home
DevelopmentLessonsLearned	Restricted usergroups, middlewarebugs, protocols to acquireresources	Unreliable user-contributed resources,limitedtoafewapps

- **Computational and Data Grids** – Used in **national-scale supercomputing projects** (e.g., **TeraGrid, ChinaGrid, e-Science Grid**).
- **P2P Grids** – Utilize client machines for open, distributed computing (e.g., **SETI@Home, JXTA, FightingAID@Home**).
- **Challenges** include **middleware bugs, security issues, and unreliable user-contributed resources**.

1.3.3 Peer-to-Peer (P2P) Network Families

P2P systems **eliminate central coordination**, allowing **client machines to act as both servers and clients**.

1.3.3.1 P2P Systems

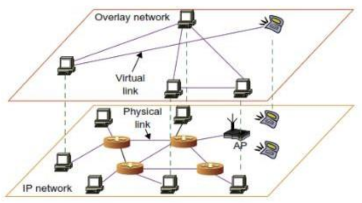


FIGURE 1.17
The structure of a P2P system by mapping a physical IP network to an overlay network built with virtual links.

Decentralized architecture with self-organizing peers.

No central authority; all nodes are independent.

Dynamic membership – peers can join and leave freely.

1.3.3.2 Overlay Networks

- **Logical connections between peers**, independent of the physical network.
- Two types:
  - **Unstructured overlays** – Randomly connected peers, requiring flooding for data retrieval (high traffic).
  - **Structured overlays** – Use predefined rules for routing and data lookup, improving efficiency.

1.3.3.3 P2P Application Families

P2P networks serve **four main application categories**:

Category	**Examples**	**Challenges**
File Sharing	Napster, BitTorrent, Gnutella	Copyright issues, security conc
Collaboration Plat	Skype, MSN, Multiplayer ga	Privacy risks, spam, lack of tru
Distributed Comp	SETI@Home, Genome@Ho	Security vulnerabilities, selfish
Open P2P Platforn	JXTA, .NET, FightingAID@	Lack of standardization and sec

1.3.3.4 P2P Computing Challenges

- **Heterogeneity** – Varying hardware, OS, and network configurations.
- **Scalability** – Must handle growing workloads and distributed resources efficiently.
- **Data Location & Routing** – Optimizing data placement for better performance.
- **Fault Tolerance & Load Balancing** – Peers can fail unpredictably.

- **Security & Privacy** – No central control means increased risk of **data breaches and malware**.

P2P networks offer **robust and decentralized computing**, but **lack security and reliability**, making them suitable only for **low-security applications** like file sharing and collaborative tools.

Both **grid computing** and **P2P networks** provide **scalable, distributed computing** models. While **grids** are used for **structured, high-performance computing**, **P2P networks** enable **decentralized, user-driven resource sharing**. Future developments will focus on **security, standardization, and efficiency improvements**.

Cloud Computing over the Internet

Cloud computing has emerged as a transformative **on-demand computing paradigm**, shifting computation and data storage from desktops to **large data centers**. This approach enables the **virtualization of hardware, software, and data resources**, allowing users to access scalable services over the Internet.

1.3.4.1 Internet Clouds

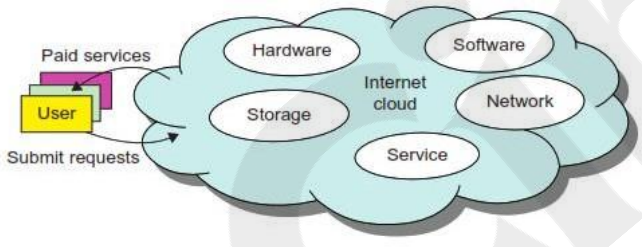


FIGURE 1.18

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.

- **Cloud computing leverages virtualization** to dynamically provision resources, reducing costs and complexity.
- It offers **elastic, scalable, and self-recovering** computing power through **server clusters and large databases**.
- The cloud can be perceived as either a **centralized resource pool** or a **distributed computing platform**.
- **Key benefits:** Cost-effectiveness, flexibility, and multi-user application support.

1.3.4.2 The Cloud Landscape

Traditional computing systems suffer from **high maintenance costs, poor resource utilization, and expensive hardware upgrades**. Cloud computing resolves these issues by providing **on-demand access** to computing resources.

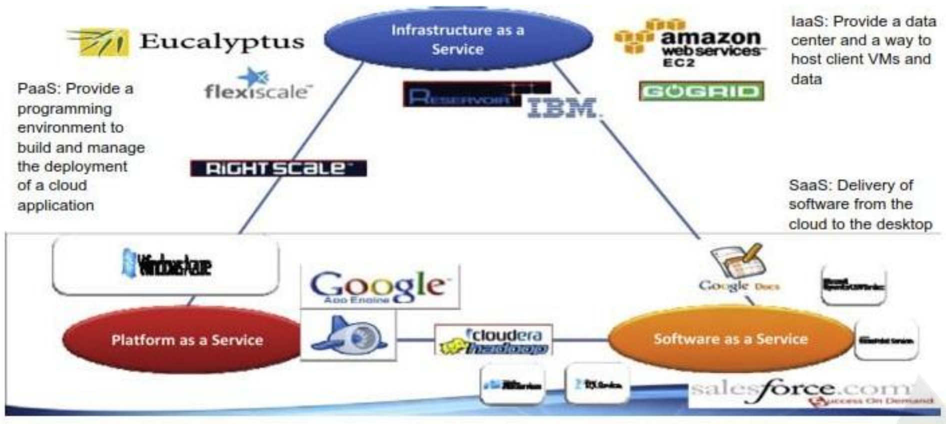

Infrastructure as a Service
Eucalyptus
flexiscale
amazon web services EC2
GOGRID
IBM
Reservoir
IaaS: Provide a data center and a way to host client VMs and data
PaaS: Provide a programming environment to build and manage the deployment of a cloud application
RIGHTSCALE
SaaS: Delivery of software from the cloud to the desktop
Windows Azure
Google App Engine
Google Docs
cloudera hadoop
Platform as a Service
Software as a Service
salesforce.com

FIGURE 1.19

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

Three Major Cloud Service Models:

1. **Infrastructure as a Service (IaaS)**
   - Provides **computing infrastructure** such as **virtual machines (VMs), storage, and networking**.
   - Users **deploy and manage their applications** but do not control the underlying infrastructure.
   - **Examples:** Amazon EC2, Google Compute Engine.
2. **Platform as a Service (PaaS)**
   - Offers a **development environment** with **middleware, databases, and programming tools**.
   - Enables developers to **build, test, and deploy applications** without managing infrastructure.
   - **Examples:** Google App Engine, Microsoft Azure, AWS Lambda.
3. **Software as a Service (SaaS)**
   - Delivers **software applications via web browsers**.
   - Users **pay for access** instead of purchasing software licenses.
   - **Examples:** Google Workspace, Microsoft 365, Salesforce.

Cloud Deployment Models:

- **Private Cloud** – Dedicated to a single organization (e.g., corporate data centers).
- **Public Cloud** – Hosted by third-party providers for general use (e.g., AWS, Google Cloud).
- **Managed Cloud** – Operated by a **third-party service provider** with customized configurations.

- **Hybrid Cloud** – Combines **public and private clouds**, optimizing **cost and security**.

Advantages of Cloud Computing

Cloud computing provides **several benefits** over traditional computing paradigms, including:

1. **Energy-efficient data centers** in secure locations.
2. **Resource sharing**, optimizing utilization and handling peak loads.
3. **Separation of infrastructure maintenance** from application development.
4. **Cost savings** compared to traditional on-premise infrastructure.
5. **Scalability for application development** and cloud-based computing models.
6. **Enhanced service and data discovery** for content and service distribution.
7. **Security and privacy improvements**, though challenges remain.
8. **Flexible service agreements and pricing models** for cost-effective computing.

Cloud computing **fundamentally changes** how applications and services are developed, deployed, and accessed. With **virtualization, scalability, and cost efficiency**, it has become the **backbone of modern Internet services and enterprise computing**. Future advancements will focus on **security, resource optimization, and hybrid cloud solutions**.

1.4 Software Environments for Distributed Systems and Clouds

This section introduces **Service-Oriented Architecture (SOA)** and other key software environments that enable **distributed and cloud computing systems**. These environments define how **applications, services, and data** interact within grids, clouds, and P2P networks.

1.4.1 Service-Oriented Architecture (SOA)

SOA enables **modular, scalable, and reusable software components** that communicate over a network. It underpins **web services, grids, and cloud computing environments**.

1.4.1.1 Layered Architecture for Web Services and Grids

- Distributed computing **builds on the OSI model**, adding layers for **service interfaces, workflows, and management**.

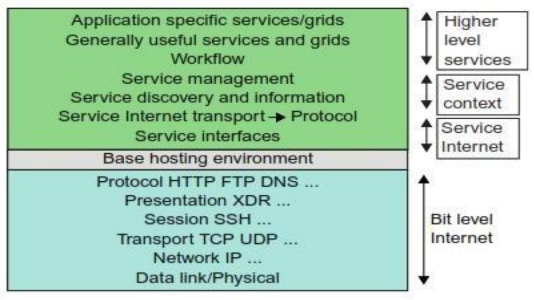


FIGURE 1.20
Layered achitecture for web services and the grids.

-

- **Communication standards** include:
    - **SOAP (Simple Object Access Protocol)** – Used in web services.
    - **RMI (Remote Method Invocation)** – Java-based communication.
    - **IIOP (Internet Inter-ORB Protocol)** – Used in CORBA-based systems.
- **Middleware tools** (e.g., WebSphere MQ, Java Message Service) manage **messaging, security, and fault tolerance**.

1.4.1.2 Web Services and Tools

SOA is implemented via two main approaches:

1. **Web Services (SOAP-based)** – Fully specified service definitions, enabling **distributed OS-like environments**.
2. **REST (Representational State Transfer)** – Simpler, lightweight alternative for **web applications and APIs**.

- **Web Services** provide **structured, standardized communication** but face challenges in protocol agreement and efficiency.
- **REST** is **flexible and scalable**, better suited for **fast-evolving environments**.
- **Integration of Services** – Distributed systems use **Remote Method Invocation (RMI) or RPCs** to link services into larger applications.

1.4.1.3 The Evolution of SOA

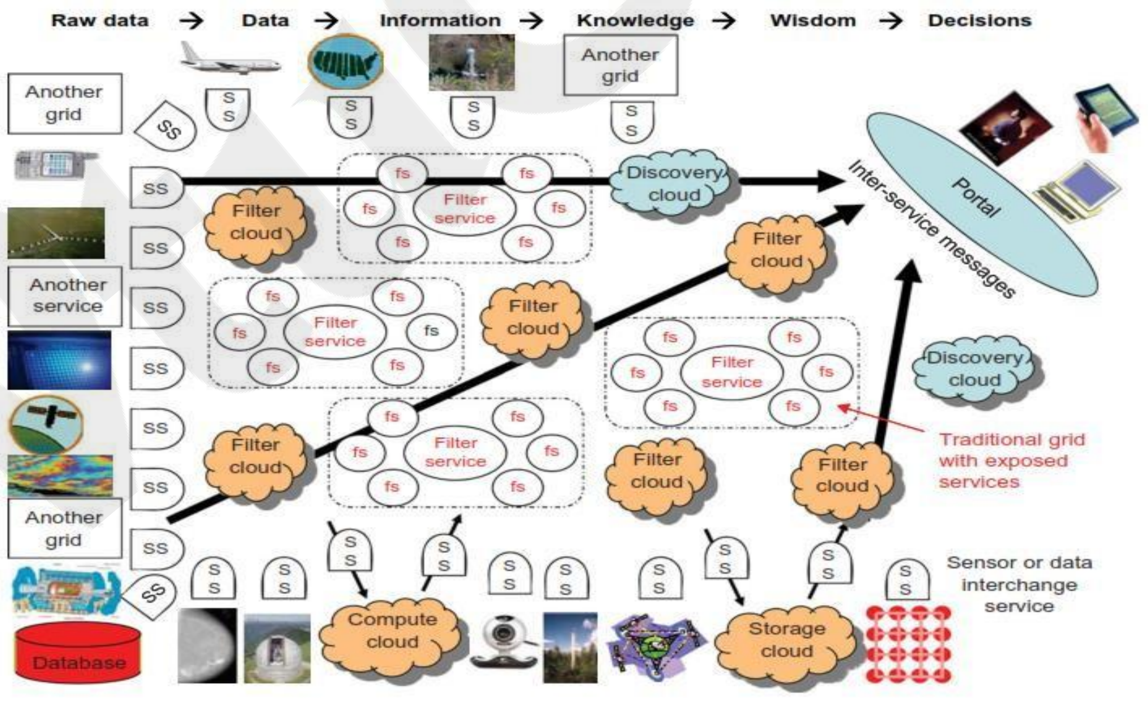


FIGURE 1.21

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

SOA has expanded from **basic web services to complex multi-layered ecosystems**:

- **Sensor Services (SS)** – Devices like **ZigBee, Bluetooth, GPS, and WiFi** collect raw data.
- **Filter Services (FS)** – Process data before feeding into **computing, storage, or discovery clouds**.
- **Cloud Ecosystem** – Integrates **compute clouds, storage clouds, and discovery clouds** for managing large-scale applications.

SOA enables **data transformation** from **raw data → useful information → knowledge → wisdom → intelligent decisions**.

SOA **defines the foundation for web services, distributed systems, and cloud computing**. By **integrating sensors, processing layers, and cloud resources**, SOA provides a **scalable, flexible** approach for **modern computing applications**. The future of **distributed computing** will rely on **intelligent data processing, automation, and service-driven architectures**.

1.4.1.4 Grids vs. Clouds

- **Grids use static resources**, whereas **clouds provide elastic, on-demand resources** via virtualization.
- **Clouds focus on automation and scalability**, while **grids are better for negotiated resource allocation**.
- Hybrid models exist, such as **clouds of grids, grids of clouds, and inter-cloud architectures**.

1.4.2 Trends toward Distributed Operating Systems

Traditional distributed systems **run independent OS instances** on each node, resulting in **multiple system images**. A **distributed OS** manages all resources **coherently and efficiently** across nodes.

1.4.2.1 Distributed OS Approaches (Tanenbaum's Models)

1. **Network OS** – Basic resource sharing via **file systems** (low transparency).
2. **Middleware-based OS** – Limited resource sharing through **middleware extensions** (e.g., MOSIX for Linux clusters).
3. **Truly Distributed OS** – Provides **single-system image (SSI)** with **full transparency** across resources.

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 vs. DCE

- **Amoeba (microkernel approach)** offers a lightweight distributed OS model.
- **DCE (middleware approach)** extends UNIX for **RPC-based distributed computing**.
- **MOSIX2** enables **process migration** across **Linux-based clusters and clouds**.

1.4.2.3 MOSIX2 for Linux Clusters

- **Supports virtualization** for seamless process migration across **multiple clusters and clouds**.
- **Enhances parallel computing** by dynamically balancing workloads across Linux nodes.

1.4.2.4 Transparency in Programming Environments

- **Cloud computing separates user data, applications, OS, and hardware** for flexible computing.
- Users can **switch between OS platforms and cloud services** without being locked into specific applications.

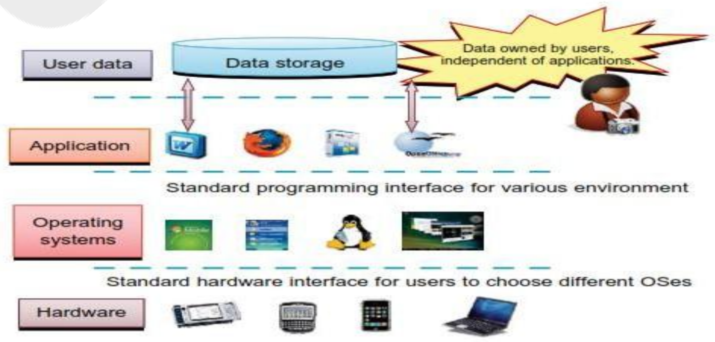

User data
Data storage
Data owned by users, independent of applications.
Application
Standard programming interface for various environment
Operating systems
Standard hardware interface for users to choose different OSes
Hardware

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.

1.4.3 Parallel and Distributed Programming Models

Distributed computing requires **efficient parallel execution models** to process large-scale workloads.

Model	Description	Key Features
MPI (Message-Passing Interface)	Standard for writing **parallel applications** on distributed systems	**Explicit communication between processes** via **message-passing**
MapReduce	Web programming model for **scalable data processing** on large clusters	**Map function generates key-value pairs; Reduce function merges values**
Hadoop	Open-source framework for **processing vast datasets** in business and cloud applications	**Distributed storage (HDFS) and MapReduce-based computing**

1.4.3.1 Message-Passing Interface (MPI)

- Used for **high-performance computing (HPC)**.
- Programs explicitly **send and receive messages** for inter-process communication.

1.4.3.2 MapReduce

- **Highly scalable parallel model**, used in **big data processing** and **search engines**.
- Splits workloads into **Map (processing) and Reduce (aggregation) tasks**.
- **Google executes thousands of MapReduce jobs daily** for large-scale data analysis.

1.4.3.3 Hadoop

- **Open-source alternative to MapReduce**, used for **processing petabytes of data**.
- **Scalable, cost-effective, and fault-tolerant**, making it ideal for cloud services.

1.4.3.4 Grid Standards and Toolkits

Grids use **standardized middleware** to manage **resource sharing and security**.

Standard	Function	Key Features
OGSA (Open Grid Services Architecture)	Defines **common grid services**	Supports **heterogeneous computing, security policies, and resource allocation**
Globus Toolkit (GT4)	Middleware for **resource discovery and security**	Uses **PKI authentication, Kerberos, SSL, and delegation policies**
IBM Grid Toolbox	Grid computing framework for **AIX/Linux clusters**	Supports **autonomic computing and security management**

- **Distributed OS models are evolving**, with MOSIX2 enabling **process migration and resource sharing** across Linux clusters.
- **Parallel programming models like MPI and MapReduce**optimize large-scale computing.
- **Cloud computing and grid computing** continue to merge, leveraging **virtualization and elastic resource management**.
- **Standardized middleware (OGSA, Globus)** enhances **grid security, interoperability, and automation**.

1.5 Performance, Security, and Energy Efficiency

This section discusses **key design principles** for distributed computing systems, covering **performance metrics, scalability, system availability, fault tolerance, and energy efficiency**.

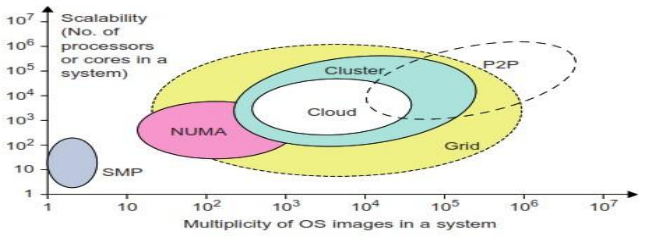
1.5.1 Performance Metrics and Scalability Analysis

Performance is measured using **MIPS, Tflops, TPS, and network latency**. Scalability is crucial in distributed systems and has **multiple dimensions**:

1. **Size Scalability** – Expanding system resources (e.g., processors, memory, storage) to improve performance.
2. **Software Scalability** – Upgrading OS, compilers, and libraries to accommodate larger systems.
3. **Application Scalability** – Increasing problem size to match system capacity for cost-effectiveness.
4. **Technology Scalability** – Adapting to new hardware and networking technologies while ensuring compatibility.

1.5.1.3 Scalability vs. OS Image Count

- **SMP systems** scale up to a few hundred processors due to hardware constraints.
- **NUMA systems** use multiple OS images to scale to thousands of processors.
- **Clusters and clouds** scale further by using virtualization.
- **Grids** integrate multiple clusters, supporting **hundreds of OS images**.
- **P2P networks** scale to millions of nodes with independent OS images.



- FIGURE 1.23
System scalability versus multiplicity of OS images based on 2010 technology.

1.5.1.4 Amdahl's Law (Fixed Workload Scaling)

- Speedup in parallel computing is limited by the sequential portion of a program.
- **Speedup Formula:** $\text{Speedup} = S = T/[\alpha T + (1-\alpha)T/n] = 1/[\alpha + (1-\alpha)/n]$

  where **α** is the fraction of the workload that is sequential.
- Even with **hundreds of processors**, speedup is limited if **sequential execution (α) is high**.
- **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 by Hwang and Xu [14]. To execute a fixed workload on n processors, parallel processing may lead to a system efficiency defined as follows:

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

1.5.1.6 Gustafson's Law (Scaled Workload Scaling)

- Instead of fixing workload size, this model **scales the problem to match available processors**.
- **Speedup Formula:** $S' = W'/W = [\alpha W + (1-\alpha)nW]/W = \alpha + (1-\alpha)n$
- ThisspeedupisknownasGustafson'slaw.ByfixingtheparallelexecutiontimeatlevelW,the followingefficiencyexpression isobtained:

  $E' = S'/n = \alpha/n + (1-\alpha)$
- **More efficient for large clusters**, as workload scales dynamically with system size.

1.5.2 Fault Tolerance and System Availability

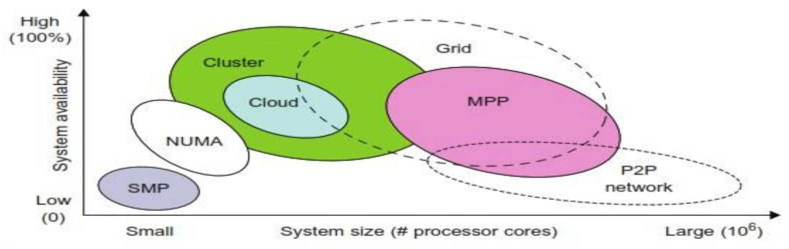
- **High availability (HA)** is essential in **clusters, grids, P2P networks, and clouds**.
- **System availability** depends on **Mean Time to Failure (MTTF) and Mean Time to Repair (MTTR)**: Availability=MTTF/(MTTF+MTTR)
- 

  FIGURE 1.24
  Estimated system availability by system size of common configurations in 2010.
- **Eliminating single points of failure** (e.g., hardware redundancy, fault isolation) improves availability.

- **P2P networks** are highly scalable but have **low availability** due to frequent peer failures.
- **Grids and clouds** offer better **fault isolation** and thus **higher availability** than traditional clusters.
- **Scalability and performance depend on resource expansion, workload distribution, and parallelization.**
- **Amdahl's Law limits speedup for fixed workloads, while Gustafson's Law optimizes large-scale computing.**
- **High availability requires redundancy, fault tolerance, and system design improvements.**
- **Clouds and grids balance scalability and availability better than traditional SMP or NUMA systems.**

Network Threats, Data Integrity, and Energy Efficiency

This section highlights **security challenges, energy efficiency concerns, and mitigation strategies** in distributed computing systems, including clusters, grids, clouds, and P2P networks.

1.5.3 Network Threats and Data Integrity

Distributed systems require **security measures** to prevent cyberattacks, data breaches, and unauthorized access.

1.5.3.1 Threats to Systems and Networks

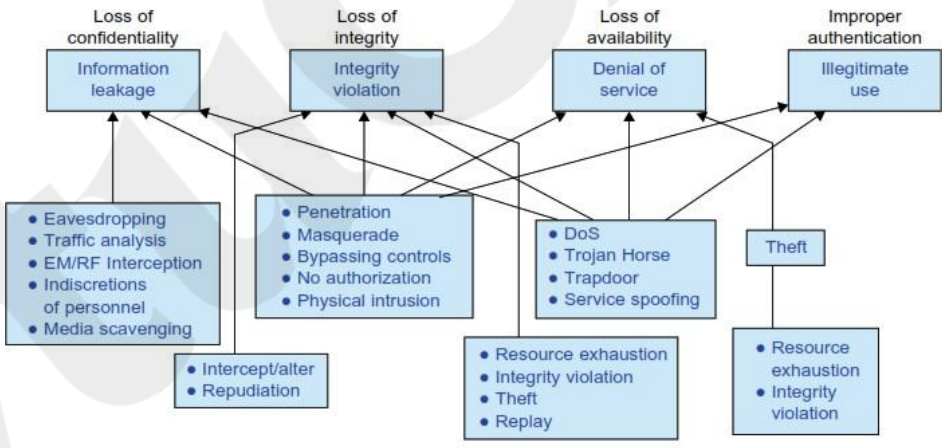


FIGURE 1.25
Various system attacks and network threats to the cyberspace, resulting 4 types of losses.

- **Loss of Confidentiality** – Due to **eavesdropping, traffic analysis, and media scavenging**.
- **Loss of Integrity** – Caused by **penetration attacks, Trojan horses, and unauthorized access**.
- **Loss of Availability** – Denial of Service (DoS) and **resource exhaustion** disrupt system operation.

- **Improper Authentication** – Allows attackers to steal resources, modify data, and conduct replay attacks.

1.5.3.2 Security Responsibilities

Security in cloud computing is divided among different stakeholders based on the cloud service model:

- **SaaS**: Cloud provider handles **security, availability, and integrity**.
- **PaaS**: Provider manages **integrity and availability**, while users control **confidentiality**.
- **IaaS**: Users are responsible for **most security aspects**, while providers ensure **availability**.

1.5.3.3 Copyright Protection

- **Collusive piracy** in P2P networks allows unauthorized file sharing.
- **Content poisoning** and **timestamped tokens** help detect piracy and protect digital rights.

1.5.3.4 System Defense Technologies

Three generations of **network security** have evolved:

1. **Prevention-based** – Access control, cryptography.
2. **Detection-based** – Firewalls, intrusion detection systems (IDS), Public Key Infrastructure (PKI).
3. **Intelligent response systems** – AI-driven threat detection and response.

1.5.3.5 Data Protection Infrastructure

- **Trust negotiation** ensures secure data sharing.
- **Worm containment & intrusion detection** protect against cyberattacks.
- **Cloud security responsibilities vary** based on the service model (SaaS, PaaS, IaaS).

1.5.4 Energy Efficiency in Distributed Computing

Distributed systems must balance **high performance with energy efficiency** due to increasing power costs and environmental impact.

1.5.4.1 Energy Consumption of Unused Servers

- Many servers are left **powered on but idle**, leading to **huge energy waste**.
- **Global energy cost of idle servers**: **\$3.8 billion** annually, with **11.8 million tons of CO_2 emissions**.
- IT departments must **identify underutilized servers** to reduce waste.

1.5.4.2 Reducing Energy in Active Servers

Energy consumption can be managed across **four layers** (Figure 1.26):

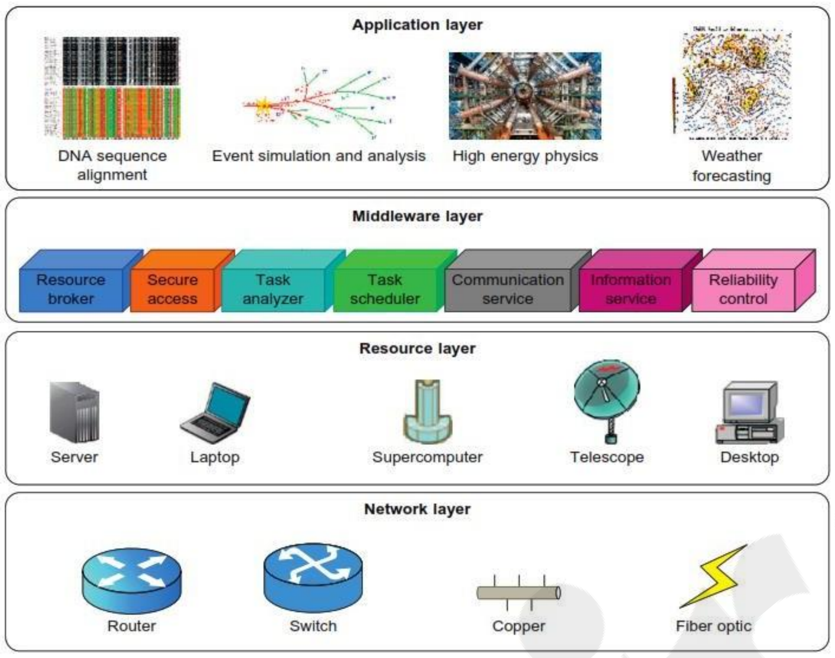


FIGURE 1.26
Four operational layers of distributed computing systems.

1. **Application Layer** – Optimize software to balance performance and energy consumption.
2. **Middleware Layer** – Smart **task scheduling** to reduce unnecessary computations.
3. **Resource Layer** – Use **Dynamic Power Management (DPM)** and **Dynamic Voltage-Frequency Scaling (DVFS)**.
4. **Network Layer** – Develop **energy-efficient routing algorithms** and optimize bandwidth usage.

1.5.4.3 Dynamic Voltage-Frequency Scaling (DVFS)

- **Reduces CPU voltage and frequency** during idle times to save power.
    - **Formula for Energy Consumption in CMOS Circuits**:

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

    - **Lowering voltage and frequency significantly reduces energy usage**.
- **Potential savings**: DVFS can **cut power consumption** while maintaining performance.
- **Energy efficiency** is critical due to **high costs and environmental impact**.
- **Techniques like DPM and DVFS** can significantly reduce power consumption **without compromising performance**.

Example 1.2 Energy Efficiency in Distributed Power Management

Figure 1.27 illustrates the DVFS method. This technique as shown on the right saves the energy compared to traditional practices shown on the left. The idea is to reduce the frequency and/or voltage during workload slack time. The transition latencies between lower-power modes are very small. Thus energy is saved by switching between operational modes. Switching between low-power modes affects performance. Storage units must interact with the computing nodes to balance power consumption. According to Ge, Feng, and Cameron [21], the storage devices are responsible for about 27 percent of the total energy consumption in a data center. This figure increases rapidly due to a 60 percent increase in storage needs annually, making the situation even worse.

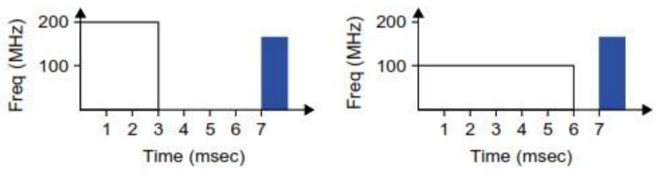


FIGURE 1.27

The DVFS technique (right) saves energy, compared to traditional practices (left) by reducing the frequency or voltage during slack time.