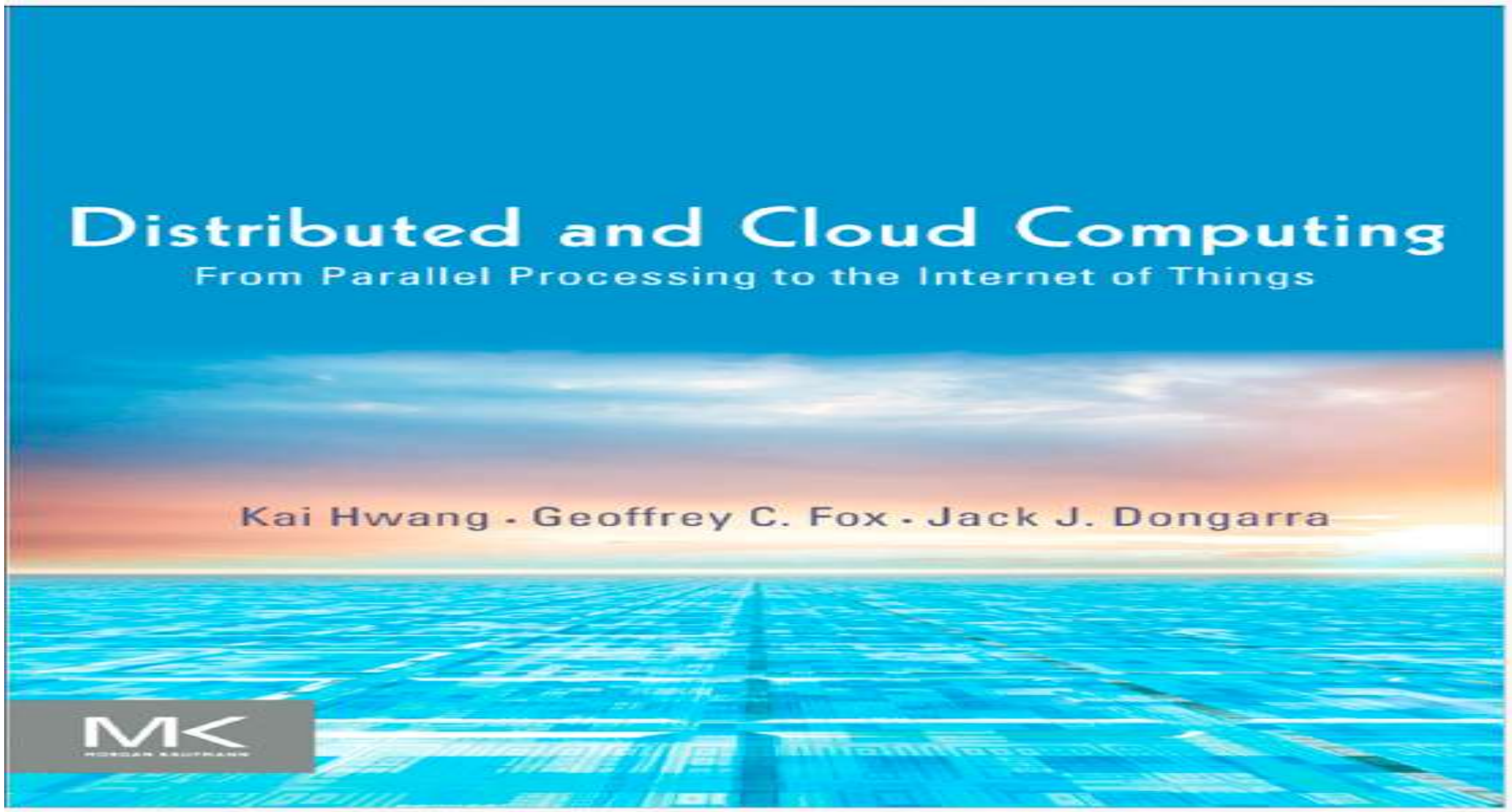
Cloud Computing

BCS601



Distributed and Cloud Computing

From Parallel Processing to the Internet of Things

Kai Hwang · Geoffrey C. Fox · Jack J. Dongarra

MK

Course objectives:

- Introduce the rationale behind the cloud computing revolution and the business drivers
- Understand various models, types and challenges of cloud computing
- Understand the design of cloud native applications, the necessary tools and the design tradeoffs.
- Realize the importance of Cloud Virtualization, Abstraction`s, Enabling Technologies and cloud security

COURSE OUTCOMES

At the end of the course, the student will be able to:

- Describe various cloud computing platforms and service providers.
- Illustrate the significance of various types of virtualization.
- Identify the architecture, delivery models and industrial platforms for cloud computing based applications.
- Analyze the role of security aspects in cloud computing.
- Demonstrate cloud applications in various fields using suitable cloud platforms.

Text Books:

1. Kai Hwang, Geoffrey C Fox, and Jack J Dongarra, Distributed and Cloud Computing, Morgan Kaufmann, Elsevier 2012
2. Dan C. Marinescu, Cloud Computing Theory and Practice, Morgan Kaufmann, 2nd Edition, Elsevier 2018
3. Google Cloud Teaching Resources – LMS [for practical component]
4. AWS Cloud Developing – AWS Academy Courses [for practical component]

Reference Books:

1. Rajkumar Buyya, Christian Vecchiola, and Thamrai Selvi, Mastering Cloud Computing McGrawHill Education, 1st Edition, 2017
2. Toby Velte, Anthony Velte, Cloud Computing: A Practical Approach, McGraw-Hill Education, 2017.
3. George Reese, Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, O'Reilly Publication, 1st Edition, 2009
4. John Rhoton, Cloud Computing Explained: Implementation Handbook for Enterprises, Recursive Press, 2nd Edition, 2009.

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.

Textbook 1: Chapter 1: 1.1 to 1.5

Module-2

Virtual Machines and Virtualization of Clusters and Data Centers: Implementation Levels of Virtualization, Virtualization Structure/Tools and Mechanisms, Virtualization of CPU/Memory and I/O devices, Virtual Clusters and Resource Management, Virtualization for Data Center Automation.

Textbook 1: Chapter 3: 3.1 to 3.5

Module-3

Cloud Platform Architecture over Virtualized Datacenters: Cloud Computing and Service Models, Data Center Design and Interconnection Networks, Architectural Design of Compute and Storage Clouds, Public Cloud Platforms: GAE, AWS and Azure, Inter-Cloud Resource Management.

Textbook 1: Chapter 4: 4.1 to 4.5

Module-4

Cloud Security: Top concern for cloud users, Risks, Privacy Impact Assessment, Cloud Data Encryption, Security of Database Services, OS security, VM Security, Security Risks Posed by Shared Images and Management OS, XOAR, A Trusted Hypervisor, Mobile Devices and Cloud Security

Cloud Security and Trust Management: Cloud Security Defense Strategies, Distributed Intrusion/Anomaly Detection, Data and Software Protection Techniques, Reputation-Guided Protection of Data Centers.

Textbook 2: Chapter 11: 11.1 to 11.3, 11.5 to 11.8, 11.10 to 11.14

Textbook 1: Chapter 4: 4.6.

Textbook 1: Chapter 3: 3.1 to 3.5

Module-5

Cloud Programming and Software Environments:
Features of Cloud and Grid Platforms, Parallel and Distributed Computing Paradigms, Programming Support for Google App Engine, Programming on Amazon AWS and Microsoft, Emerging Cloud Software Environments.

Textbook 1: Chapter 6: 6.1 to 6.5

Practical Components

Sl.NO	**Experiments**
1	Creating a Virtual Machine: Configure and deploy a virtual machine with specific CPU and memory requirements in Google Cloud. OR Exploring AWS CloudShell and the AWS Cloud9 IDE
2	Getting Started with Cloud Shell and gcloud: Discover the use of gcloud commands to manage Google Cloud resources from Cloud Shell. OR Working with Amazon S3Orchestrating Serverless Functions with AWS Step Functions
3	**Cloud Functions**: Create and deploy a Cloud Function to automate a specific task based on a Cloud Storage event. OR Working with Amazon DynamoDB
4	**App Engine**: Deploy a web application on App Engine with automatic scaling enabled. OR Developing REST APIs with Amazon API Gateway
5	**Cloud Storage: Qwikstart: Google** Cloud Storage provides scalable and secure object storage for managing data, accessible via the Cloud Console or gsutil CLI. OR Creating Lambda Functions Using the AWS SDK for Python
6	Cloud SQL for MySQL: Discover how Google Cloud SQL for MySQL provide automated management and high availability for MySQL databases? OR Migrating a Web Application to Docker Containers
7	**Cloud Pub/Sub:** Experiment how Google Cloud Pub/Sub facilitate real-time messaging and communication between distributed applications. OR Caching Application Data with ElastiCache, Caching with Amazon CloudFronT, Caching Strategies

7	**Cloud Pub/Sub:** Experiment how Google Cloud Pub/Sub facilitate real-time messaging and communication between distributed applications. OR Caching Application Data with ElastiCache, Caching with Amazon CloudFronT, Caching Strategies
8	**Multiple VPC Networks:** Explore benefits of using multiple VPC networks in Google Cloud for organizing and isolating resources. OR Implementing CloudFront for Caching and Application Security
9	**Cloud Monitoring:** Discover how Cloud Monitoring help in tracking and analyzing the performance and health of cloud resources? OR Orchestrating Serverless Functions with AWS Step Functions
10	Kubernetes Engine: Qwik Start: Deploy a containerized application to a Kubernetes Engine cluster. OR Automating Application Deployment Using a CI/CD Pipeline

1.SCALABLE COMPUTING OVER THE INTERNET

Scalable Computing over the Internet

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

1.1. The Age of Internet Computing

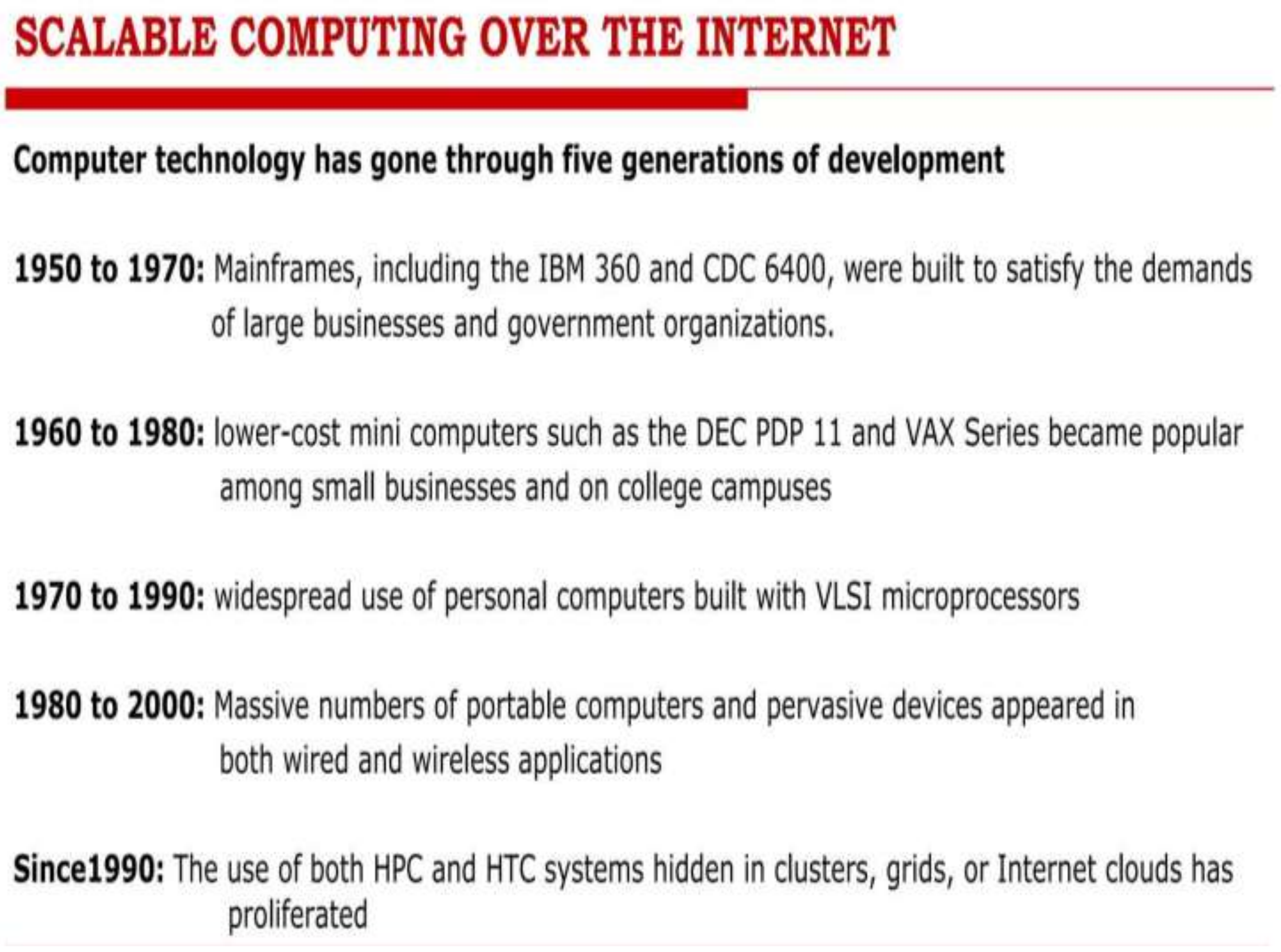
High-performance computing (HPC)

- ✓ applications is no longer optimal for measuring system performance

High-throughput computing (HTC) systems

- ✓ built with parallel and distributed computing technologies
- ✓ upgrade data centers using fast servers, storage systems, and high-bandwidth networks.

The Platform Evolution



SCALABLE COMPUTING OVER THE INTERNET

Computer technology has gone through five generations of development

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

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

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

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

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

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.

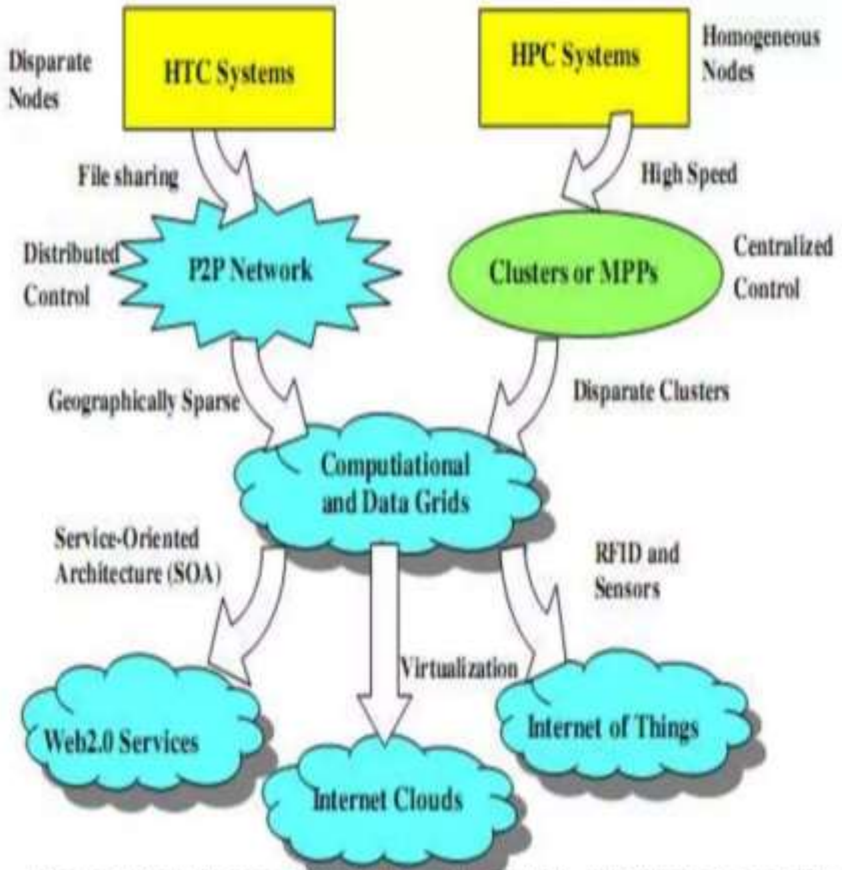


Figure: Evolutionary trend toward parallel, distributed, and cloud

- **On the HTC side, peer-to-peer (P2P) networks are formed for distributed file sharing.**
- A P2P system is built over many client machines . Peer machines are globally distributed in nature.
- P2P, cloud computing, and web service platforms are more focused on HTC applications than on HPC applications.
- **Clustering and P2P technologies lead to the development of computational grids or data grids.**

The maturity of radio-frequency identification (RFID), Global Positioning System (GPS), and sensor technologies has triggered the development of the Internet of Things (IoT).

High-Performance Computing (HPC)

High-Performance Computing (HPC) has historically prioritized raw speed, increasing from GFLOPS in the early 90s to PFLOPS in 2010, driven by scientific, engineering, and manufacturing needs. The TOP500 list, using the Linpack benchmark, reflects this focus. However, HPC is only used by a small fraction (under 10%) of computer users, with most people relying on desktops and servers for everyday tasks like web browsing and market-driven computing.

High-Throughput Computing (HTC)

High-end computing is shifting from High-Performance Computing (HPC), focused on raw speed, to High-Throughput Computing (HTC), which prioritizes high-flux computing. HTC is designed for applications like web services and internet searches used by millions concurrently. Performance is measured by the number of tasks completed per unit of time (throughput). Beyond processing speed, HTC development must address cost, energy efficiency, security, and reliability in data centers.

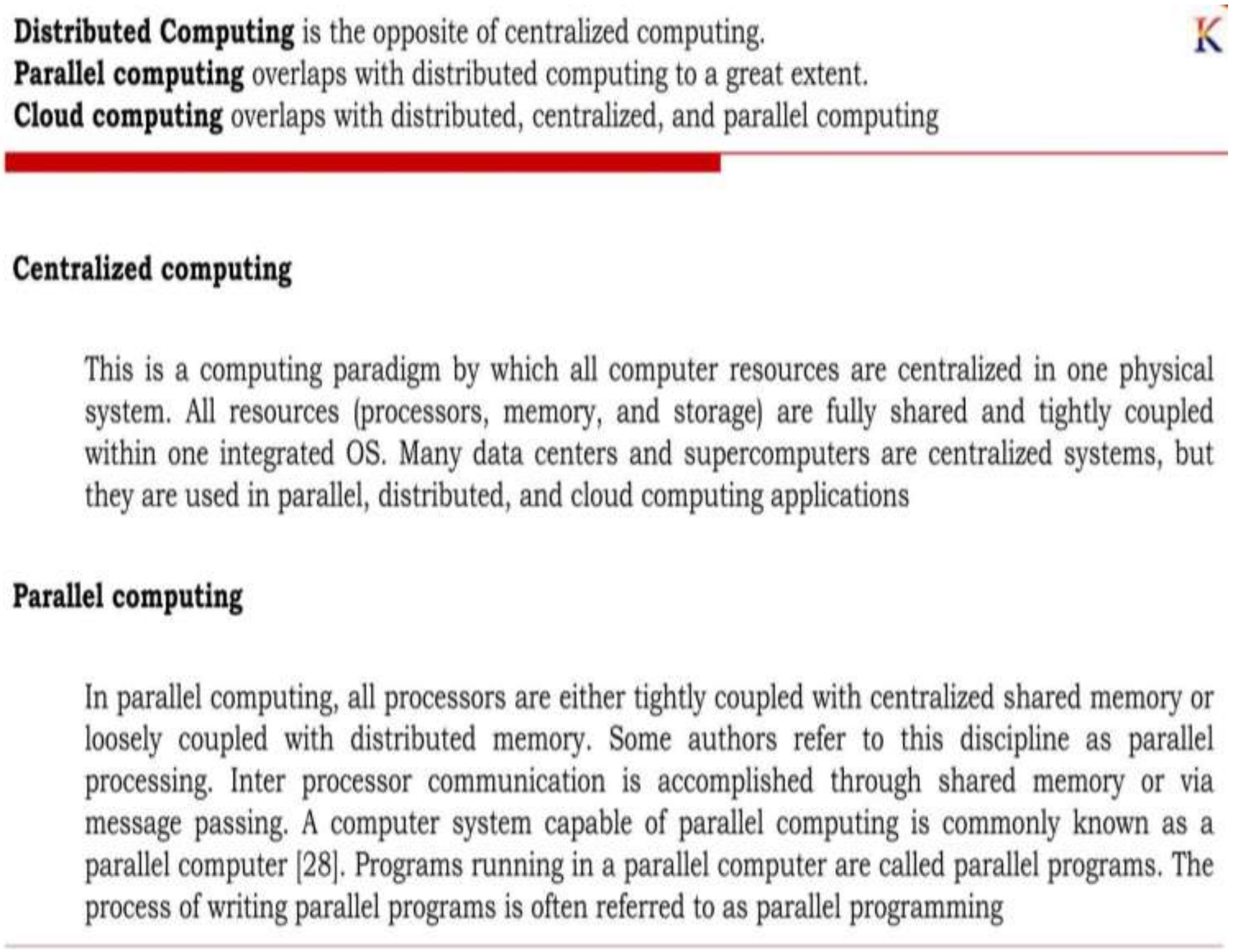
Three New Computing Paradigms

Three new computing paradigms that have emerged alongside advancements in technology:
Service-Oriented Architecture (SOA): Enabled by the introduction of SOA, Web 2.0 services have become widely available.
Internet Clouds: Advancements in virtualization technology have facilitated the growth of Internet clouds as a new computing model.
Internet of Things (IoT): The development of RFID, GPS, and sensor technologies has led to the rise of the IoT.

Three New Computing Paradigms

Three New Computing Paradigms
SOA, Web 2.0 services become available. Advances
in virtualization make it possible to see the growth of Internet clouds as a new computing paradigm.
The maturity of radio-frequency identification (RFID), Global Positioning System (GPS), and sensor
technologies has triggered the development of the Internet of Things (IoT). These new paradigms
are only briefly introduced here. We will study the details of SOA in Chapter 5; virtualization in
Chapter 3; cloud computing in Chapters 4, 6, and 9; and the IoT along with cyber-physical systems
(CPS) in Chapter 9.
When the Internet was introduced in 1969, Leonard Klienrock of UCLA declared: "As of now,
computer networks are still in their infancy, but as they

Computing Paradigm Distinctions



Distributed Computing is the opposite of centralized computing.
Parallel computing overlaps with distributed computing to a great extent.
Cloud computing overlaps with distributed, centralized, and parallel computing

Centralized computing

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

Parallel computing

In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Some authors refer to this discipline as parallel processing. Inter processor communication is accomplished through shared memory or via message passing. A computer system capable of parallel computing is commonly known as a parallel computer [28]. Programs running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming

Distributed Computing is the opposite of centralized computing.
Parallel computing overlaps with distributed computing to a great extent.
Cloud computing overlaps with distributed, centralized, and parallel computing

Distributed computing:

This is a field of computer science/engineering that studies distributed systems. A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network. Information exchange in a distributed system is accomplished through message passing. A computer program that runs in a distributed system is known as a distributed program. The process of writing distributed programs is referred to as .

Cloud computing

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

Paradigm Shift:

High-end computing is moving away from High-Performance Computing (HPC) and towards High-Throughput Computing (HTC).

Focus on Throughput: HTC prioritizes processing a large number of tasks (high throughput) rather than focusing solely on the speed of individual tasks (as in HPC).

High-Flux Computing: HTC emphasizes "high-flux computing," meaning it's designed to handle a massive flow of data and tasks.

Key Applications: HTC is ideal for applications used by millions of people concurrently, such as:

- Internet searches
- Web services

Performance Metric: The primary measure of success in HTC is "throughput," which is the number of tasks completed within a given time frame. This is different from HPC, where the focus is often on how fast a single, complex task can be completed.

Developing HTC involves addressing crucial issues like:

- Cost: Keeping the infrastructure and operations affordable.
- Energy Savings: Minimizing energy consumption for environmental and cost reasons.
- Security: Protecting data and systems from threats.
- Reliability: Ensuring consistent and uninterrupted

- In the future, both HPC and HTC systems will demand multi core or many-core processors that can handle large numbers of computing threads per core.

- Both HPC and HTC systems emphasize parallelism and distributed computing. Future HPC and HTC systems must be able to satisfy this huge demand in computing power in terms of throughput, efficiency, scalability, and reliability.

- The system efficiency is decided by speed, programming, and energy factors. Meeting these goals requires to yield the following design objectives:

  - ✓ Efficiency
  - ✓ Dependability
  - ✓ Adaptation in the programming model
  - ✓ Flexibility in application deployment

Efficiency: measures the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput, data access, storage, and power efficiency.

- Dependability measures the reliability and self-management from the chip to the system and application levels. The purpose is to provide high-throughput service with Quality of Service (QoS) assurance, even under failure conditions.
- Adaptation in the programming model measures the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workload and service models.
- Flexibility in application deployment measures the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

The interplay between technological trends and computing applications, emphasizing the desire to predict future system capabilities. It mentions:

Jim Gray's work: His paper exemplifies how technology influences applications and vice versa.

Moore's Law: Processor speed doubles every 18 months (though its future validity is uncertain).

Gilder's Law: Network bandwidth doubles annually (future trend also uncertain).

Commodity Hardware: The price/performance ratio of commodity hardware, driven by personal computing markets, has influenced large-scale computing adoption.

Evolution of parallelism in computing, moving from finer to coarser
granularity:
Bit-Level Parallelism (BLP): Early computers used bit-serial processing. BLP shifted to word-level processing, progressing from 4-bit to 64-bit CPUs.
Instruction-Level Parallelism (ILP): Processors began executing multiple instructions simultaneously (pipelining, superscalar, VLIW, multithreading). ILP requires sophisticated techniques like branch prediction and dynamic scheduling.
Data-Level Parallelism (DLP): SIMD and vector machines enabled processing multiple data elements with a single instruction. DLP also needs hardware and compiler support.

Task-Level Parallelism (TLP): With multicore processors, the focus shifted to parallel execution of larger tasks. TLP is more challenging to implement efficiently due to programming and compilation difficulties.

Job-Level Parallelism (JLP): In distributed processing, even larger units of work (jobs) are executed in parallel.

The passage concludes by noting that modern processors utilize all these types of parallelism and that coarse-grain parallelism (like JLP) builds upon fine-grain parallelism (BLP, ILP, DLP).

1.2. SCALABLE COMPUTING TRENDS AND NEW PARADIGMS

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 highlights few key applications that have driven the development of parallel and distributed systems over the

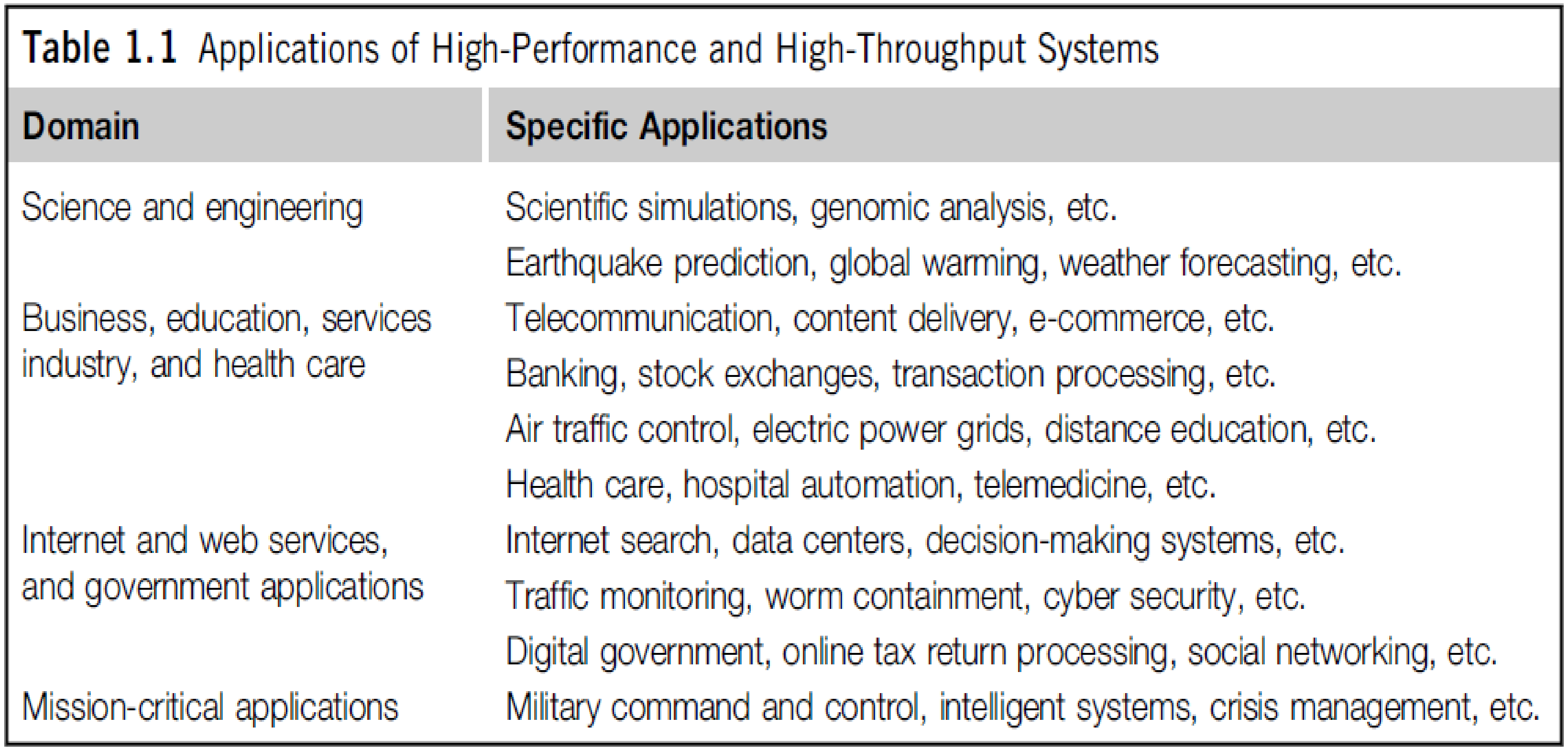


Table 1.1 Applications of High-Performance and High-Throughput Systems

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.

Several key computing paradigms and their shared characteristics

Common Characteristics of Paradigms: These paradigms share three key traits:

- **Ubiquity:** They are pervasive in daily life.
- **Reliability and Scalability:** They are designed to be dependable and able to handle increasing demands.
- **Autonomic Operation:** They aim for self-organization and dynamic resource discovery.
- **Composability:** They can be combined with Quality of Service (QoS) and Service Level Agreements (SLAs).

Computer Utility Vision: These paradigms realize the vision of computing as a utility, similar to electricity or telephone services.

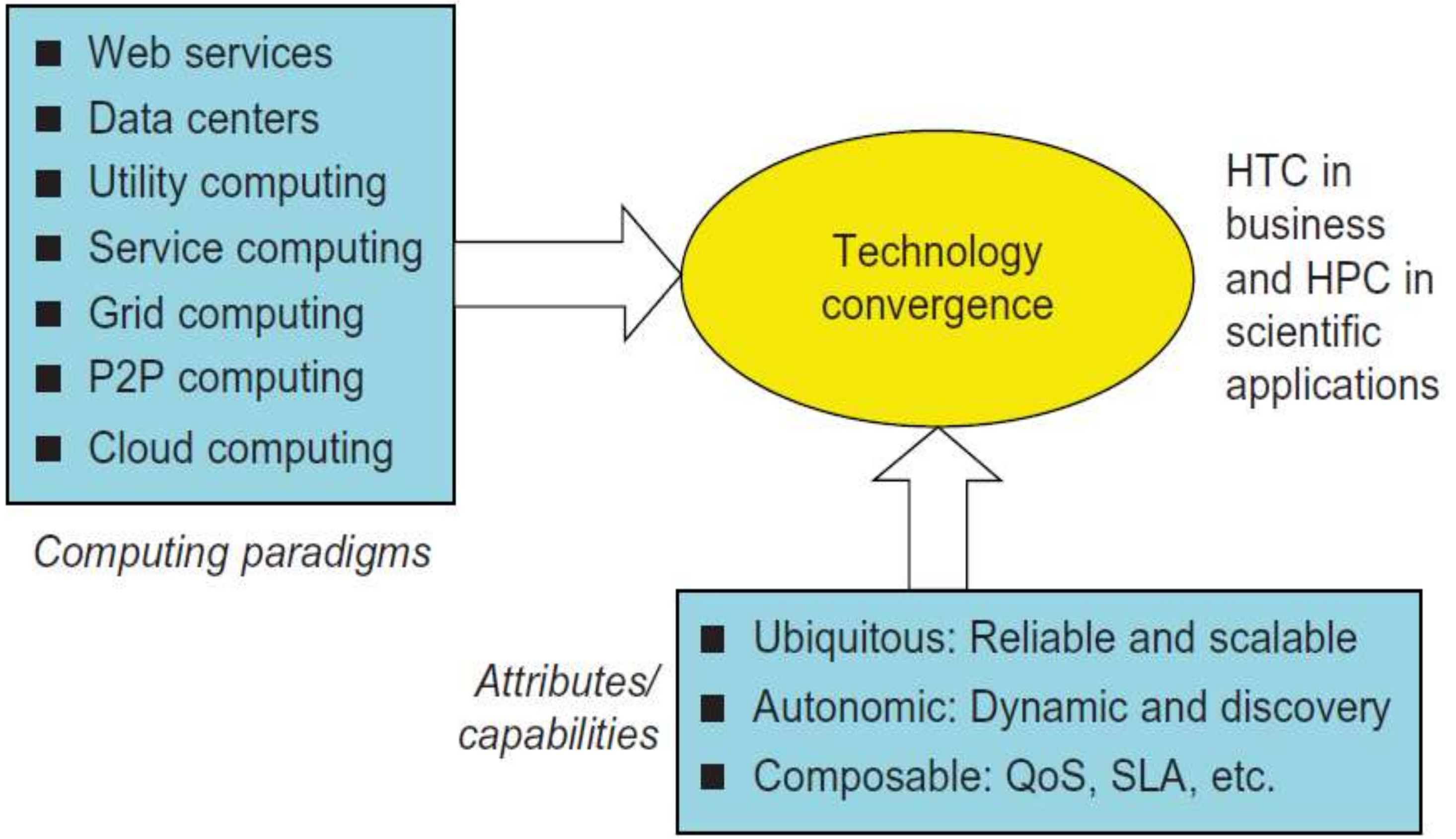
Utility Computing: This model focuses on a business relationship where customers pay a service provider for computing resources. Grid and cloud platforms are considered utility providers.

Cloud Computing's Broader Scope: Cloud computing encompasses a wider concept than simple utility computing. Cloud applications can run on diverse servers across edge networks.

Technological Challenges: Realizing these paradigms presents significant challenges across computer science and engineering, including:

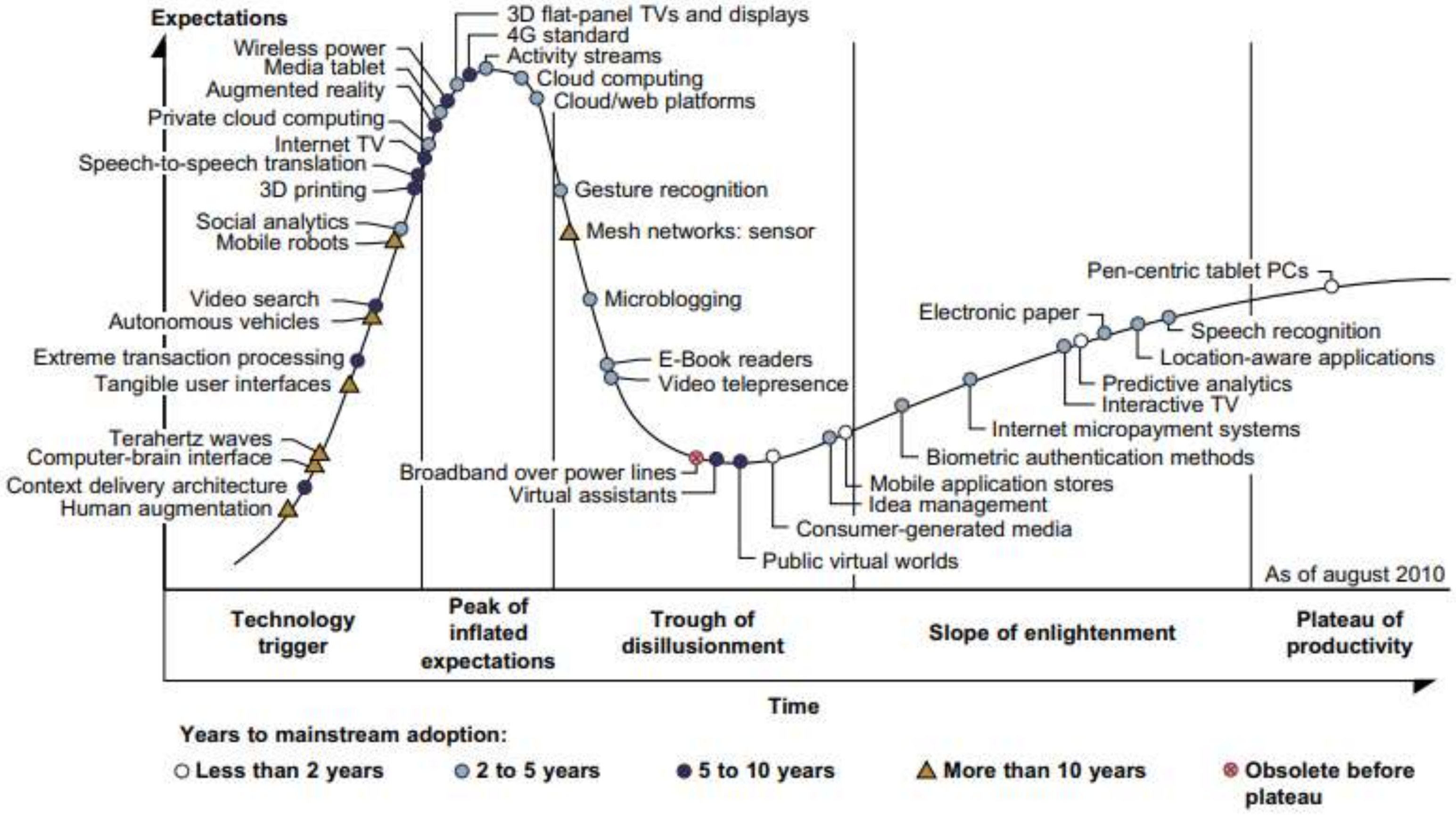
Network-efficient processors
Scalable memory and storage
Distributed operating systems
Middleware for virtualization
New programming models
Effective resource management
Application development tools

1.2.2.TREND TOWARDS UTILITY COMPUTING



1

The Hype Cycle of New Technologies as on Aug 2010:



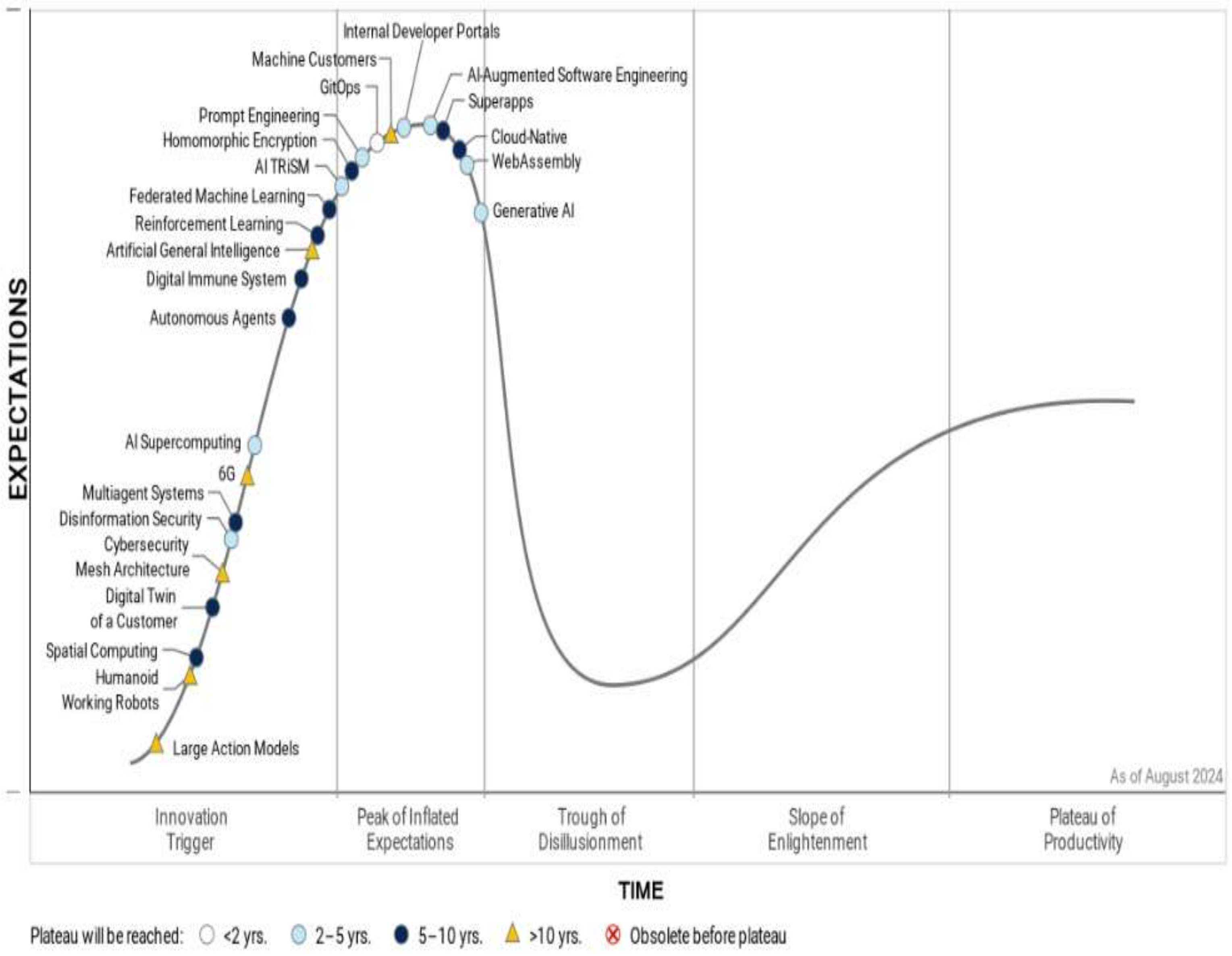
The Hype Cycle as on Aug 2010: Emerging technologies progress through five stages:

1. **Trigger:** The initial spark or invention.
2. **Peak of Inflated Expectations:** Enthusiasm and hype build, often exceeding realistic potential.
3. **Trough of Disillusionment:** Reality sets in, and expectations fall as limitations become apparent.
4. **Slope of Enlightenment:** The technology matures, finding practical applications and overcoming initial hurdles.
5. **Plateau of Productivity:** The technology is widely adopted and integrated into mainstream use.

- **Time to Adoption:** The passage uses symbols to indicate the predicted timeframe for technologies to reach mainstream adoption:
- Hollow circles: <2 years
- Gray circles: 2-5 years
- Solid circles: 5-10 years
- Triangles: >10 years
- Crossed circles: Obsolete before reaching plateau

•**Examples from 2010:** The passage provides examples of where various technologies were in the hype cycle as of August 2010:
•**Consumer-generated media:** Disillusionment stage, <2 years to plateau.
•**Internet micropayments:** Enlightenment stage, 2-5 years to plateau.
•**3D printing:** Rising expectations, 5-10 years to plateau.
•**Mesh network sensors:** Inflated expectations, >10 years to plateau.
•**Cloud technology:** Just past peak of expectations, 2-5 years to plateau.
•**Broadband over power line:** Obsolete.
•**Many other technologies:** At peak of expectations, 5-10 years to plateau.
•**General Trend:** Technologies climbing the "slope of enlightenment" are likely to reach the "plateau of productivity" within 2-5 years.

The Hype Cycle of New Technologies as on Aug 2024:



Gartner

1.3.THE INTERNET OF THINGS AND CYBER-PHYSICAL SYSTEMS

- Internet of Things is about connecting "Things" (Objects and Machines) to the internet and eventually to each other;

- Cyber Physical Systems (CPS) are integration of computation, networking and physical process.

Internet of Things (IoT)

IoT expands the traditional internet by interconnecting everyday objects, devices, and computers through **wireless sensor networks, RFID, and GPS technologies**.
Initially introduced in **1999 at MIT**, IoT aims to seamlessly integrate the digital and physical worlds.
With the **IPv6 protocol**, the IoT can assign $\mathbf{2^{128}}$ **unique IP addresses**, ensuring that **100 trillion objects** can be tracked globally. Researchers estimate that each individual will be surrounded by **1,000 to 5,000 smart objects**, leading to **universal addressability and large-scale data management challenges**.

Internet of Things (IoT)

The IoT enables three key **communication models**:
H2H (Human-to-Human)
H2T (Human-to-Thing)
T2T (Thing-to-Thing)
This paradigm shift envisions an **intelligent, interconnected world** where objects communicate autonomously, reducing costs and increasing efficiency. Current IoT projects are still in **early experimental stages**, primarily in **Asia and Europe**..

Future IoT advancements will be driven by **cloud computing, AI, and emerging internet technologies**, fostering **smart cities, efficient infrastructure, and intelligent resource management**. The vision of a **"Smart Earth"** includes sustainable environments with **green IT, smart governance, and intelligent services**, though achieving this goal globally remains a long-term challenge.

Cyber-Physical Systems (CPS)

A **Cyber-Physical System (CPS)** integrates **computational processes** with the **physical world**, creating an intelligent feedback loop between **computation, communication, and control ("3C" technologies)**. It combines **heterogeneous, asynchronous cyber elements** with **concurrent, data-intensive physical objects** to enhance real-world interactions.

Unlike the **Internet of Things (IoT)**, which focuses on **networking physical objects**, CPS is more concerned with **virtual reality (VR) applications** and how digital intelligence **interacts with and transforms the physical world**. This concept is particularly being explored in the **United States**, where CPS research aims to revolutionize **human-environment interactions**, much like the Internet reshaped digital communication.

2. TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

Technologies for Network-Based Systems

- 2.1 Multicore CPUs and Multithreading Technologies
- 2.2 GPU Computing to Exascale and Beyond
- 2.3 Memory, Storage, and Wide-Area Networking
- 2.4 Virtual Machines and Virtualization Middleware
- 2.5Data Center Virtualization for Cloud Computing

Multicore CPUs and Multithreading Technologies

Over the past **30 years**, advancements in **processor speed** and **network bandwidth** have been critical to the evolution of **High-Performance Computing (HPC)** and **High-Throughput Computing (HTC)** systems.

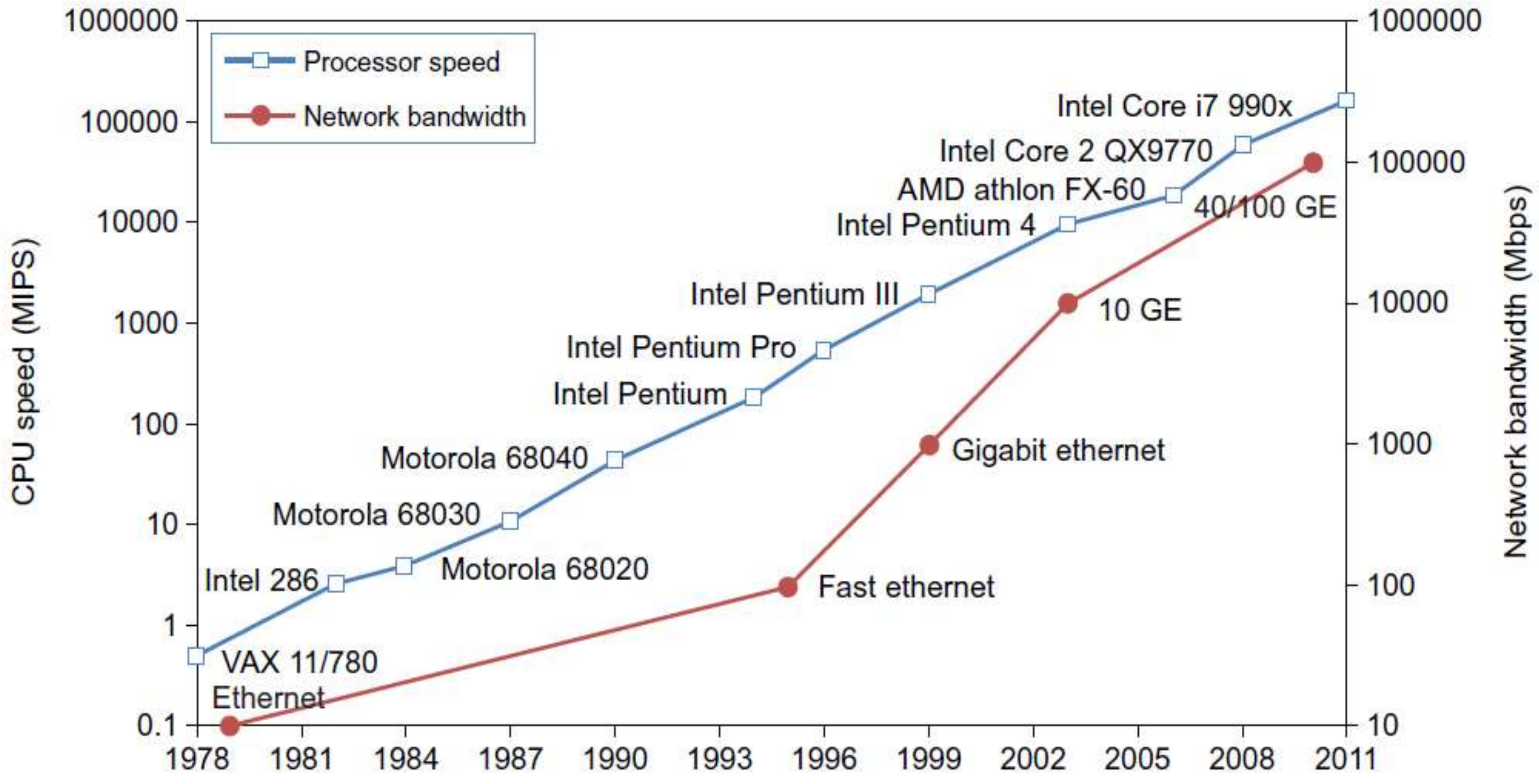
Processor Speed (Measured in MIPS)

- Increased significantly with the transition from **single-core** to **multi-core** and **many-core architectures**.
- Enhancements in **parallel processing and deep learning accelerators** have further improved computational power.

Network Bandwidth (Measured in Mbps and Gbps)

- Evolved from **megabit per second (Mbps) speeds** to **gigabit (Gbps) and beyond**.
- The introduction of **1 Gbps Ethernet (GE)** and **high-speed interconnects** like **InfiniBand and NVLink** has optimized data-intensive applications.

These improvements have enabled **faster data processing, scalability, and efficiency**, driving advancements in **scientific computing, AI, and cloud technologies**.



Improvement in processor and network technologies over 33 years.

Advances in CPU Processors

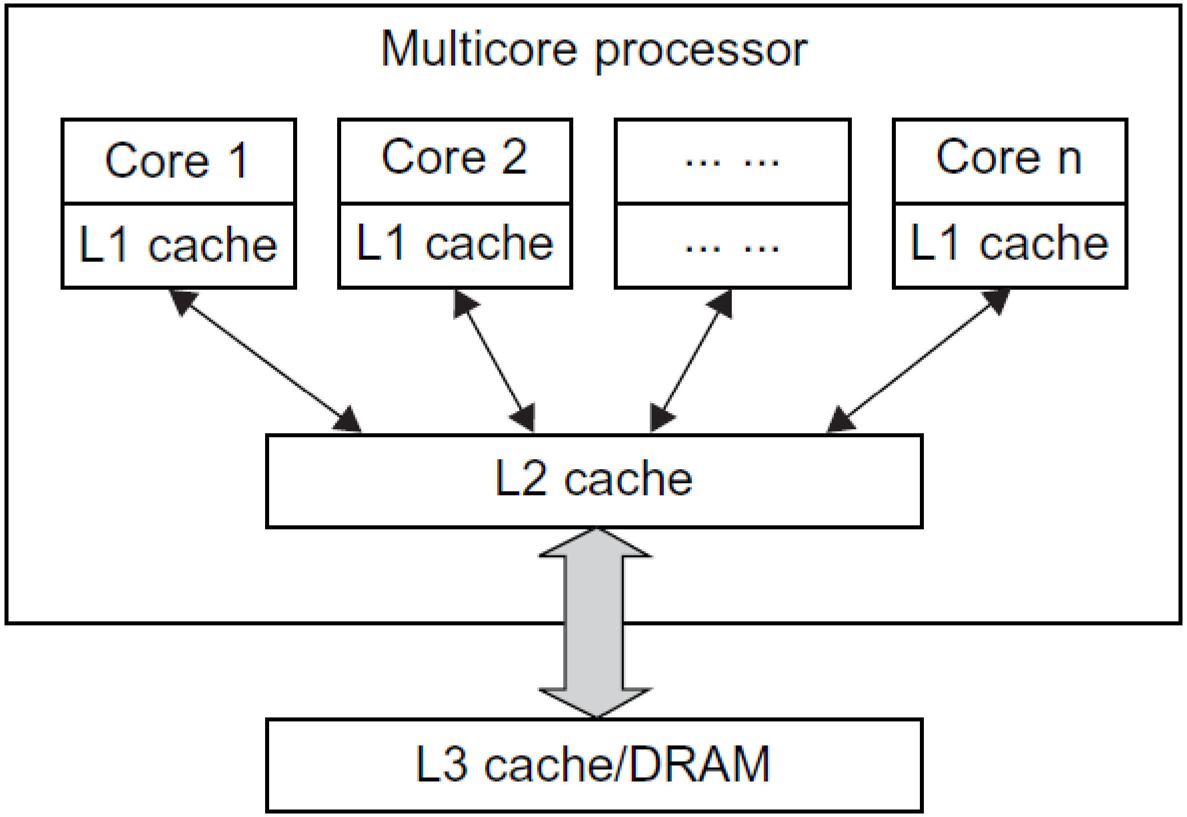
Modern CPUs have evolved into **multicore architectures**, featuring **dual, quad, six, or more cores** to enhance processing power through **Instruction-Level Parallelism (ILP) and Thread-Level Parallelism (TLP)**.

Processor Speed Growth

Increased from **1 MIPS (VAX 780, 1978)** to **1,800 MIPS (Intel Pentium 4, 2002)**, reaching **22,000 MIPS (Sun Niagara 2, 2008)**.

Moore's Law held true for decades, with clock speeds rising from **10 MHz (Intel 286)** to **4 GHz (Pentium 4)** in 30 years.

Clock speed limitations emerged due to **power constraints and excessive heat generation**, preventing most CPUs from exceeding **5 GHz**.



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.

Advances in CPU Processors

Parallel Processing and Multicore Design

ILP techniques: Superscalar architecture, branch prediction, speculative execution, requiring **hardware and compiler support**.

DLP and TLP in GPUs: Many-core GPUs (e.g., NVIDIA GPUs) integrate **hundreds to thousands of cores** to optimize parallel workloads.

Multicore CPU Architecture

Each core has its **own L1 cache**, while multiple cores share an **L2 cache**. Future designs may integrate **L3 cache and multiple Chip Multiprocessors (CMPs) on a single chip**.

Advances in CPU Processors

Advanced **multithreaded CPUs**, such as **Intel i7, Xeon, AMD Opteron, IBM Power6, and Sun Niagara**, handle **multiple threads per core**.

- **Sun Niagara II (2008)** had **8 cores with 8 threads per core**, reaching **64-way parallelism**.
- **Intel Core i7 990x (2011)** achieved **159,000 MIPS**, demonstrating exponential growth in processing capabilities.

These advancements have **revolutionized computing performance, enabling high-speed processing, AI acceleration, and efficient multitasking**.

•**Growth of Multicore CPUs**

•Future CPUs may scale from **tens to hundreds of cores**, but their ability to exploit **massive Data-Level Parallelism (DLP)** is limited by the **memory wall problem**.

•This limitation has driven the development of **many-core GPUs**, which incorporate **hundreds or more thin cores** optimized for parallel processing.

•**Instruction Set and HPC/HTC Applications**

•**IA-32 and IA-64 instruction sets** are now standard in commercial CPUs.

•**x86 processors** have evolved to power **High-Performance Computing (HPC) and High-Throughput Computing (HTC) systems**, replacing traditional **RISC architectures** in many **Top 500 supercomputers**.

•GPUs have also been integrated into **large clusters and massively parallel processing (MPP) systems** for supercomputing.

•**Future Processor Innovations**

•The industry is shifting toward **heterogeneous chip multiprocessors (CMPs)** that combine both **fat CPU cores (optimized for complex tasks)** and **thin GPU cores (optimized for parallel workloads)** on the **same chip**.

•This hybrid approach aims to improve **performance efficiency, scalability, and power consumption** in future **data centers and supercomputers**.

Multicore and Many-Core Processor Trends

- **Expansion of Multicore CPUs**
    - Future CPUs may scale from **tens to hundreds of cores**, but they face limitations in exploiting **massive Data-Level Parallelism (DLP)** due to the **memory wall problem**.
    - This limitation has driven the development of **many-core GPUs**, featuring **hundreds or more lightweight cores** optimized for parallel processing.
- **x86 Processors and HPC/HTC Adoption**
    - **IA-32 and IA-64 instruction sets** are standard in commercial CPUs, and **x86 processors** have been extended to power **High-Performance Computing (HPC) and High-Throughput Computing (HTC) systems**.

- Many **RISC processors** have been replaced by **multicore x86 CPUs and many-core GPUs**, especially in **Top 500 supercomputers**.
- **GPUs** have also been widely deployed in **massively parallel processing (MPP) clusters** for supercomputing applications.

- **Future Trends: Heterogeneous Processors**
  - The industry is moving toward **asymmetric or heterogeneous chip multiprocessors (CMPs)** that integrate both **fat CPU cores** (for complex sequential tasks) and **thin GPU cores** (for highly parallel workloads) on the **same chip**.
  - This hybrid design aims to **enhance performance, efficiency, and scalability** for future **supercomputers and data centers**.

2.1 MULTICORE CPUS AND MULTITHREADING TECHNOLOGIES

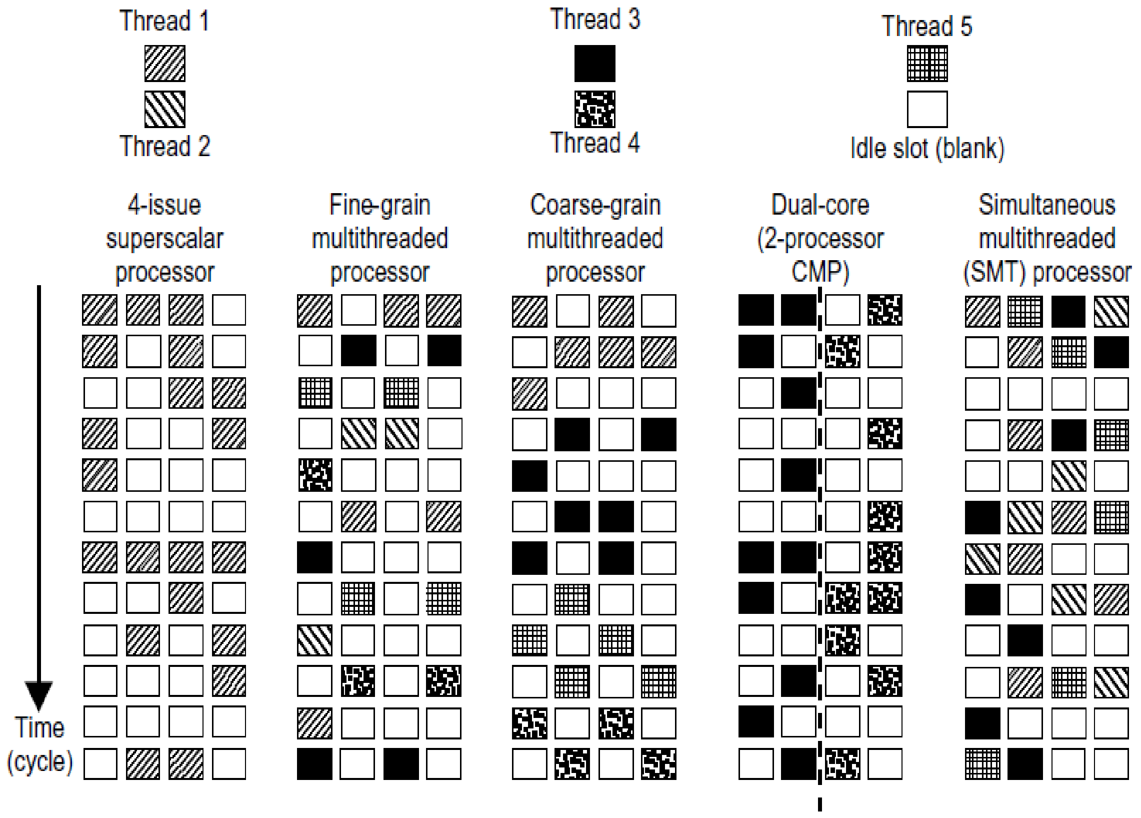

Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Idle slot (blank)
4-issue superscalar processor
Fine-grain multithreaded processor
Coarse-grain multithreaded processor
Dual-core (2-processor CMP)
Simultaneous multithreaded (SMT) processor
Time (cycle)

FIGURE 1.6

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

MULTITHREADING TECHNOLOGY

Five independent threads of instructions to **four** pipelined data paths in each of the following **five** processor categories

2.1 MULTICORE CPUS AND MULTITHREADING TECHNOLOGIES

Modern processors handle **instruction dispatch** differently based on their **microarchitecture**. The execution efficiency depends on **instruction-level parallelism (ILP)** and **thread-level parallelism (TLP)** across different processor types:

Superscalar Processor (Single-threaded, Four-Issue)

Uses **four functional units** to execute **instructions from a single thread** in parallel.

ILP-focused, but does not switch between threads.

If no instructions are available for execution, **pipeline stalls** (blank cycles) occur, reducing efficiency.

2.1 MULTICORE CPUS AND MULTITHREADING TECHNOLOGIES

Fine-Grain Multithreaded Processor

Switches execution between different threads every cycle to reduce stalls.

Maintains **consistent execution efficiency** by keeping functional units occupied.

Prioritizes **TLP** over **ILP** by ensuring multiple threads share processing time.

Coarse-Grain Multithreaded Processor

Executes **several instructions from the same thread** before switching to another.

Reduces thread-switching overhead but may introduce stalls if a thread **lacks ready instructions**.

Balances **ILP and TLP**, but context switches are more expensive than in fine-grain models.

2.1 MULTICORE CPUS AND MULTITHREADING TECHNOLOGIES

Chip Multiprocessor (CMP) with Two Cores

Has **two separate cores**, each a **two-way superscalar processor**.

Each core executes instructions independently, providing **true parallel execution**.

Threads do not interfere with each other, improving overall throughput.

Simultaneous Multithreading (SMT) Processor

Executes multiple threads in the same cycle, utilizing **all available functional units** efficiently.

Maximizes TLP by allowing instructions from different threads to **coexist within a single cycle**.

Achieves **high utilization**, but achieving maximum **ILP and TLP** simultaneously is challenging.

GPU Computing to Exascale and Beyond

A **Graphics Processing Unit (GPU)** is a specialized **graphics coprocessor** designed to **offload graphics-intensive tasks** from the **CPU**, improving performance in video editing and gaming applications.

History and Development

The first **GPU, GeForce 256**, was introduced by **NVIDIA in 1999**, capable of processing **10 million polygons per second**.

GPUs are now **integrated into nearly all modern computers** and have **influenced CPU designs** by incorporating some GPU features.

GPU Computing to Exascale and Beyond

Architectural Differences Between CPU and GPU

Traditional CPUs (e.g., **Xeon X5670**) have only a **few cores** (typically 4–12).

Modern GPUs contain **hundreds to thousands of cores**, optimized for **parallel execution**.

CPUs focus on **fast execution of single-threaded tasks**, whereas GPUs follow a **throughput architecture**, executing **many concurrent threads more slowly but in parallel**.

GPU Computing to Exascale and Beyond

GPUs in High-Performance Computing (HPC)

Parallel GPUs and GPU clusters are increasingly used in **HPC**, outperforming CPUs in **parallel computing tasks.**

General-Purpose Computing on GPUs (GPGPUs) has enabled GPUs to handle non-graphics tasks, such as **scientific simulations and AI training.**

NVIDIA CUDA is a leading **GPGPU computing model,** widely adopted for **parallel computing and HPC applications.**

How GPUs Work

Early **GPUs functioned as coprocessors** attached to the CPU, primarily handling graphics tasks. However, **modern GPUs** have evolved into **highly parallel processors**, capable of **general-purpose computing (GPGPU)**.

Massive Parallelism in Modern GPUs

NVIDIA GPUs now feature up to 128 cores per chip, with each core handling **8 instruction threads**.

This enables **1,024 concurrent threads** on a single GPU, far surpassing the limited parallelism of conventional CPUs.

CPUs are optimized for **low-latency caching**, whereas **GPUs focus on high-throughput processing** with **explicit on-chip memory management**.

How GPUs Work

Expanding Beyond Graphics Processing

GPUs are no longer restricted to **graphics and video encoding**; they now power **HPC systems and supercomputers**.

Designed for floating-point-intensive operations, GPUs are crucial for **scientific simulations, AI training, and massive data processing**.

NVIDIA's **CUDA Tesla and Fermi architectures** are widely used in **GPU clusters and HPC systems** for **parallel computing**.

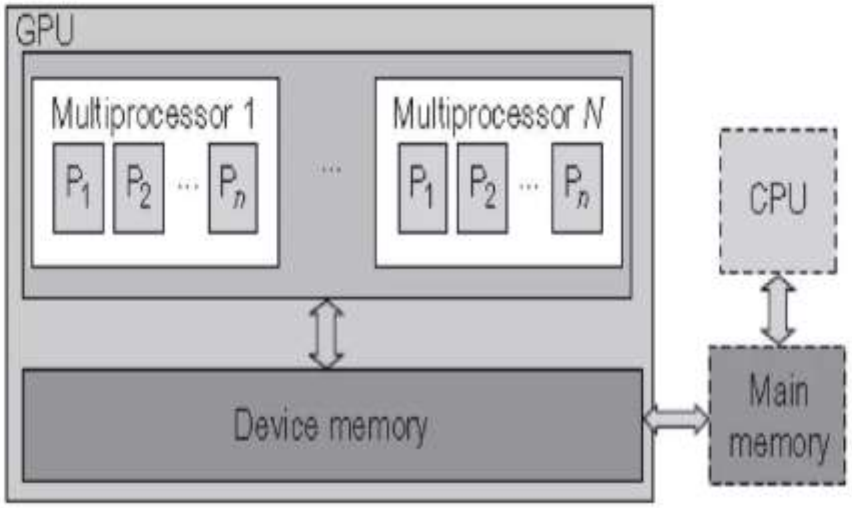
Applications Across Industries

Conventional GPUs are now **integrated into mobile phones, game consoles, embedded systems, PCs, and servers.**

Their versatility has made them essential for **high-performance computing, AI, and data analytics**.

2.2 GPU COMPUTING TO EXASCALE AND BEYOND

- GPU was marketed by NVIDIA in 1999
- Unlike CPUs, GPUs have a throughput architecture that exploits massive parallelism by executing many concurrent threads slowly, instead of executing a single long thread in a conventional microprocessor very quickly.



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

GPU Programming Model

Example 1.1 The NVIDIA Fermi GPU Chip with 512 CUDA Cores

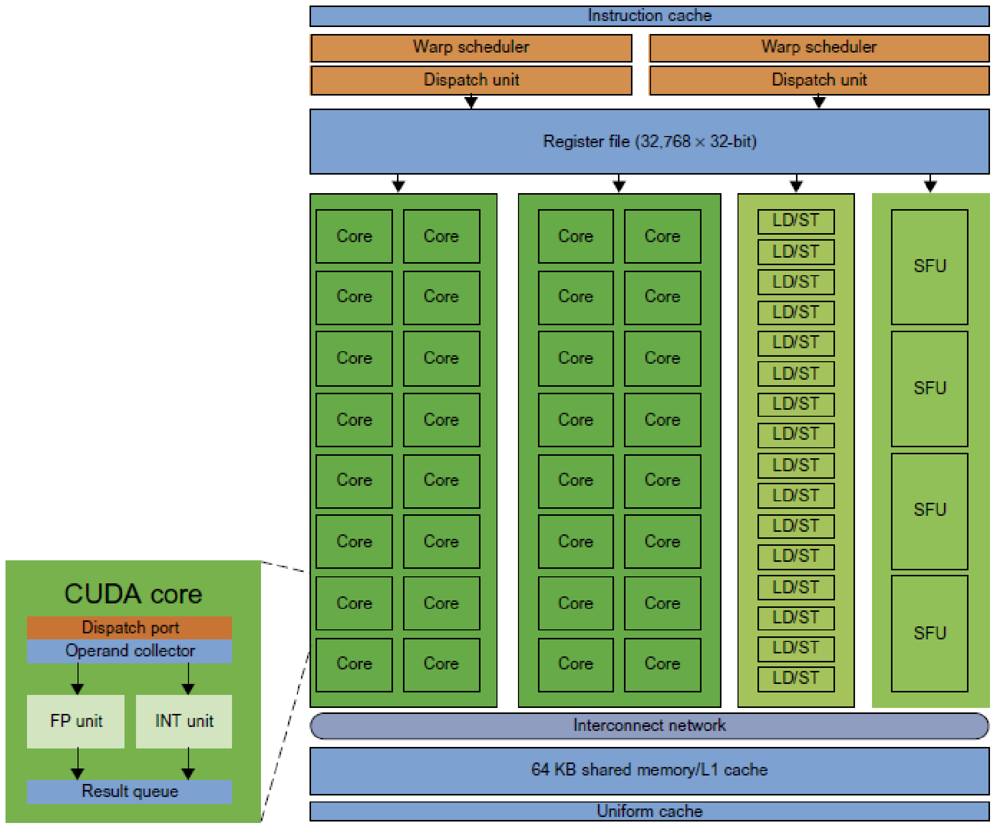


FIGURE 1.8

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

Example 1.1 The NVIDIA Fermi GPU Chip with 512 CUDA Cores

By **November 2010**, **three of the world's five fastest supercomputers (Tianhe-1A, Nebulae, and Tsubame)** utilized **GPU acceleration** for floating-point computations. These systems relied on **NVIDIA Tesla and Fermi GPUs** to enhance performance.

Fermi GPU Architecture (NVIDIA, 2011)

Built with 16 Streaming Multiprocessors (SMs) and 3 billion transistors.

Each **SM contains 32 CUDA cores**, leading to **512 total cores per GPU**.

Tesla GPUs in Tianhe-1A had 448 CUDA cores, following a similar architecture.

Example 1.1 The NVIDIA Fermi GPU Chip with 512 CUDA Cores

Parallel Execution:

Each CUDA core includes a **pipelined Integer ALU and Floating-Point Unit (FPU)**.

16 load/store units allow efficient memory access for **16 threads per clock cycle**.

4 Special Function Units (SFUs) handle complex mathematical operations (e.g., trigonometric functions).

Memory Architecture:

64 KB L1 cache per SM and a **768 KB unified L2 cache** shared across all SMs.

Connected to **6 GB of external DRAM** via **memory controllers**.

Example 1.1 The NVIDIA Fermi GPU Chip with 512 CUDA Cores

Performance Capabilities

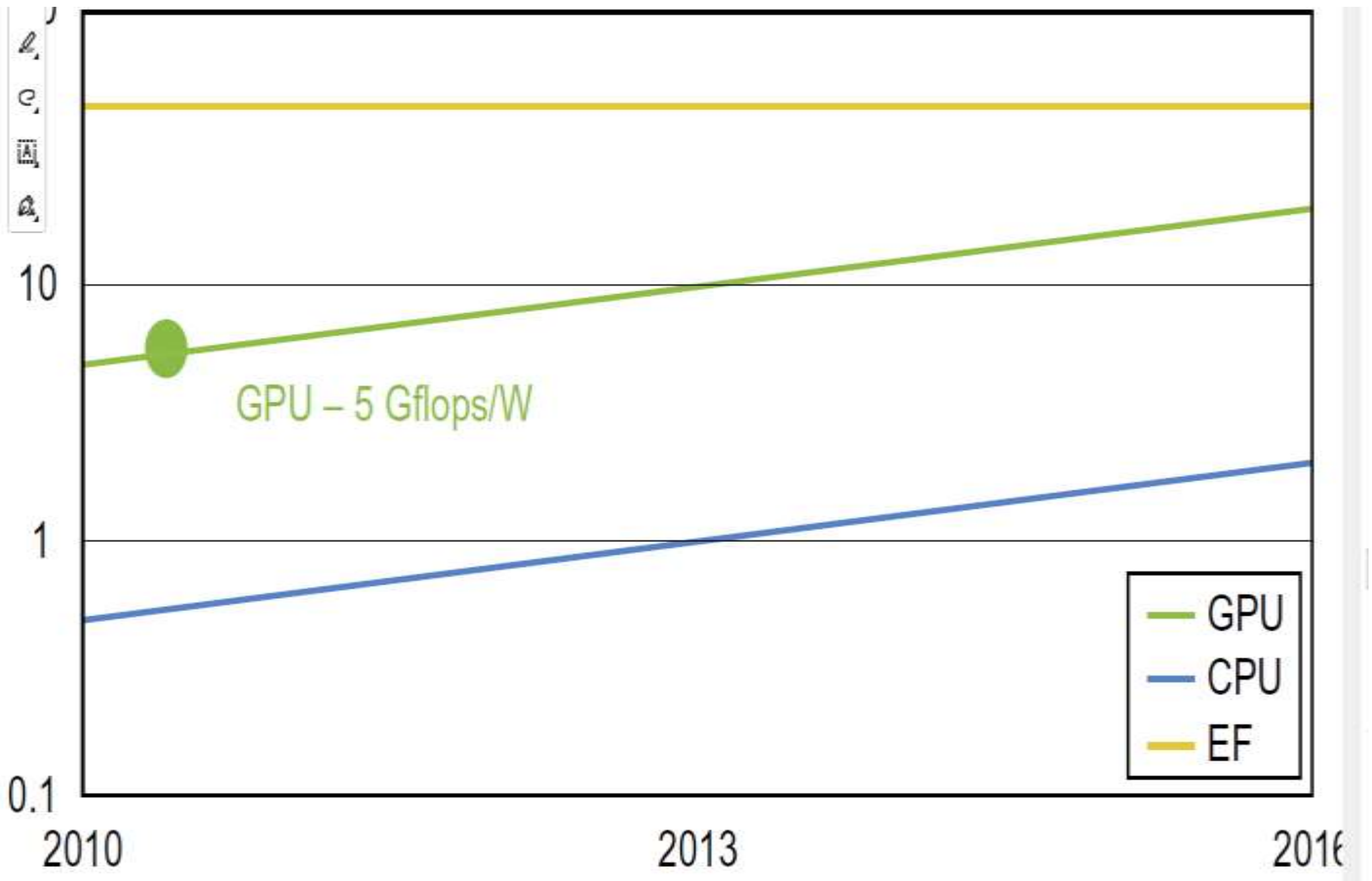
Thread Scheduling: Uses groups of **32 parallel threads (warps)** for efficient processing.

Fused Multiply-Add (FMA) Operations:

- **512 CUDA cores** can operate in parallel, delivering **515 GFlops of double-precision performance**.
- A **single Fermi GPU** can achieve **82.4 TFlops** peak performance.
- **12 Fermi GPUs combined** can potentially reach **petaflop (PFlops) performance**.

Key Takeaways

Power Efficiency of the GPU



Power Efficiency of the GPU

Bill Dally (Stanford University) highlights **power efficiency and massive parallelism** as key advantages of **GPUs over CPUs** for future computing.

Power Consumption and Efficiency

Exaflop Systems require **60 GFlops/watt per core** for feasible operation.

Power Constraints affect CPU and GPU chip design:

- **CPU power consumption**: **~2 nJ per instruction**.
- **GPU power consumption**: **~200 pJ per instruction** (**10× more efficient** than CPUs).

Optimization Differences:

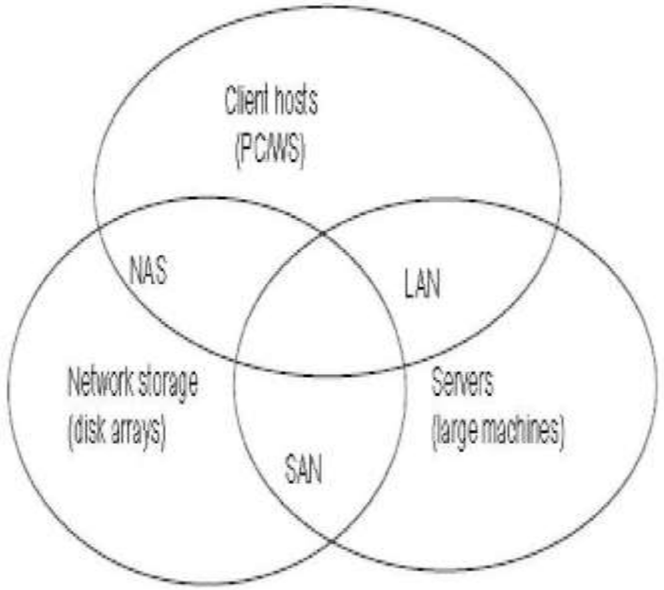
- **CPUs prioritize low-latency memory access (cache-heavy design).**
- **GPUs focus on high-throughput parallelism (explicit memory management).**

Power Efficiency of the GPU

Performance per Watt Comparison (2010)
GPU efficiency: **5 GFlops/watt per core.**
CPU efficiency: **<1 GFlop/watt per core.**
Future scaling challenges: Power consumption may limit **supercomputer growth**, but **GPUs could further close the efficiency gap with CPUs**.

Challenges in Future Computing
Optimizing Data Movement:
Power consumption is dominated by **data transfers, not computation.**
Storage hierarchy must be optimized to application needs.
Software and System-Level Innovations:
Development of **self-aware OS, runtime systems, and locality-aware compilers**.
Auto-tuning tools for GPU-based **Massively Parallel Processing (MPP) systems**.

2.3.MEMORY STORAGE AND WIDE AREA NETWORKS



LAN typically is used to connect client hosts to big servers

A storage area network (SAN) connects servers to network storage such as disk arrays

Network attached storage (NAS) connects client hosts directly to the disk arrays.

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.

Memory Technology

Memory and Storage Growth Trends

DRAM Capacity Growth

Increased from **16 KB (1976) to 64 GB (2011)**, following a **4× increase every three years.**

Despite capacity improvements, **memory access time has not improved significantly**, worsening the **memory wall problem** as processors get faster.

Hard Drive Storage Expansion

Grew from **260 MB (1981) to 250 GB (2004)**, and reached **3 TB (Seagate Barracuda XT, 2011)**.

Storage capacity has increased ~10× every eight years, with future disk arrays expected to grow even further.

Impact on Computing

The increasing gap between **processor speed and memory access time** worsens the **memory wall issue**, potentially limiting future **CPU performance**.

Solutions such as **faster memory architectures, caching strategies, and non-volatile memory innovations** will be crucial in addressing this bottleneck.

Memory Technology

Disks and Storage Technology

Disk Storage Growth

By **2011**, disks and disk arrays surpassed **3 TB** in capacity, with a **7-order magnitude increase over 33 years**.

The rise of **flash memory and solid-state drives (SSDs)** has significantly impacted **HPC and HTC systems**.

SSDs and Flash Memory Advantages

Durability: SSDs can handle **300,000 to 1 million write cycles per block**, making them **long-lasting** even under heavy workloads.

Performance: SSDs provide **significant speedups** for many applications compared to traditional **hard disks (HDDs)**.

Memory Technology

Disks and Storage Technology

Performance: SSDs provide **significant speedups** for many applications compared to traditional **hard disks (HDDs).**

Challenges in Large-Scale Computing Systems

- **Power consumption, cooling, and packaging** constraints will **limit large system development**.
- **Power scaling issues**:
  - Power **increases linearly with clock frequency** and **quadratically with voltage**.
  - **Clock speed cannot increase indefinitely**, driving demand for **lower voltage supplies**.

Future of Storage Technology

- Jim Gray (USC) predicted: *“Tape units are dead, disks are tape units, flashes are disks, and memory are caches now.”*
- **SSDs remain expensive (as of 2011),** preventing them from completely replacing **disk arrays** in the storage market.
- **Flash storage will continue to grow**, potentially **outpacing HDDs** as costs decrease.

System-Area Interconnects

- **Small Cluster Networking**
- **Nodes in small clusters** are typically connected via an **Ethernet switch** or a **Local Area Network (LAN)**.
- **LANs** are used to link **client hosts to large servers** in computing environments.
- **Storage Network Configurations**
- **Storage Area Network (SAN)**: Connects **servers** to **network storage** (e.g., disk arrays).
- **Network-Attached Storage (NAS)**: Directly connects **client hosts** to **disk arrays** for shared access.
- **Both SAN and NAS are commonly found in large clusters** using commercial network components.
- **Building Small Clusters**
- If **no large distributed storage is needed**, a **multiport Gigabit Ethernet switch with copper cables** can efficiently connect machines.
- All three networking types (**LAN, SAN, NAS**) are **commercially available** and widely used in **cluster computing**.

Wide-Area Networking

Ethernet Bandwidth Growth and HPC Networking

Rapid Ethernet Bandwidth Growth

- **10 Mbps (1979) → 1 Gbps (1999) → 40–100 Gbps (2011)**.
- **1 Tbps network links** were projected to be available by **2013**.

Reported Network Speeds (2006)

- **1,000 Gbps (1 Tbps)** – International links.
- **1,000 Gbps** – National links.
- **100 Gbps** – Organizational links.
- **10 Gbps** – Optical desktop links.
- **1 Gbps** – Copper desktop links.

Faster Growth Than Moore's Law

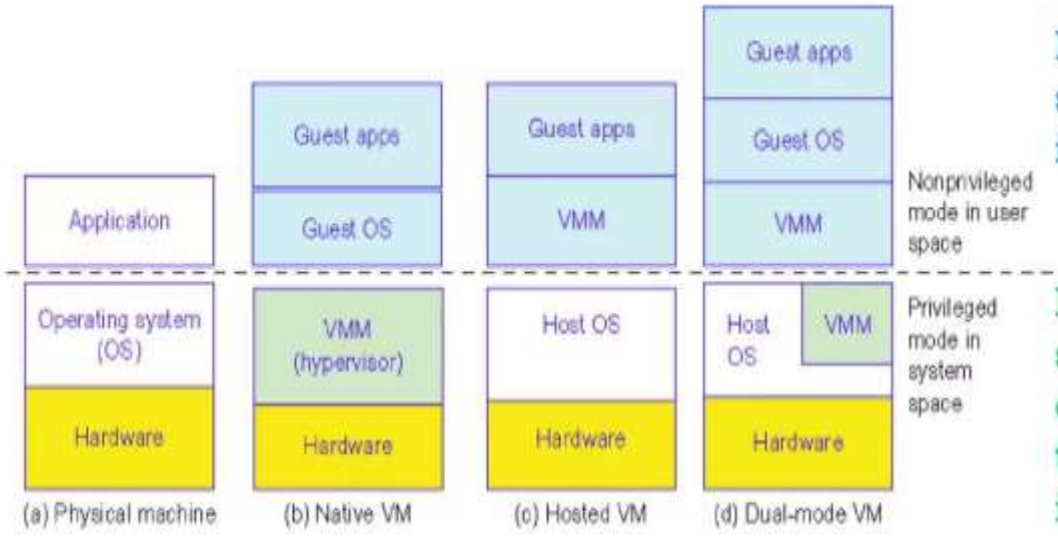
- Network performance **doubled every year**, outpacing **Moore's law**, which predicts **CPU speed doubling every 18 months**.
- Faster networking enables **massively distributed computing** and the use of **more concurrent systems**.

HPC and Data Center Networking Trends

- **InfiniBand and Ethernet** were predicted to dominate **HPC interconnects** (IDC 2010 report).
- **Gigabit Ethernet (GE)** is widely used for **server cluster interconnects in data centers**.

2.4 VIRTUAL MACHINES AND VIRTUALIZATION MIDDLEWARE

- Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.
- To build large clusters, grids, and clouds, we need to access large amounts of computing, storage, and networking resources in a virtualized manner



Multiple OS environments can exist simultaneously on the same machine is called Virtual Machine

If you want to run multiple operating systems on one machine, or multiple copies of the same operating system, then you only have two ways to do it: dual boot or virtual machine

Virtual Machines

Virtual Machines (**VMs**) provide **hardware independence** by enabling **multiple OS environments** on a single physical machine. **Figure 1.12** illustrates different VM architectures:

Host Machine and Virtualization Layers

The **host machine** contains the **physical hardware** (e.g., an **x86 desktop running Windows**).

A **Virtual Machine (VM)** is built with **virtual resources**, managed by a **guest OS** to run applications.

A **Virtual Machine Monitor (VMM)** (or **hypervisor**) is required between the **VMs** and the **host hardware**.

Types of Virtual Machine Architectures

Native (Bare-Metal) VM [Figure 1.12(b)]

The **VMM (Hypervisor)** runs in **privileged mode**, directly managing hardware resources (**CPU, memory, I/O**).

Example: **XEN hypervisor (Cambridge University)**, where an **x86 system runs Windows as the host OS** and **Linux as the guest OS**.

Host-Based VM [Figure 1.12(c)]

The **VMM runs in non-privileged mode**, relying on the host OS without modifications.

Hybrid VM (Dual-Mode) [Figure 1.12(d)]

The **VMM operates partly at the user level and partly at the supervisor level**.

The **host OS may require slight modifications** for better virtualization support.

Advantages of Virtual Machines

Hardware independence: Applications and OS can be **bundled as a virtual appliance**, making them **portable** across different hardware platforms.

OS flexibility: A VM can run a **different OS** than the host system.

VM Primitive Operations

A **Virtual Machine Monitor (VMM)** provides a **virtual machine abstraction** to the **guest OS**, allowing standard OSes (e.g., **Windows 2000, Linux**) to run as if on physical hardware (**full virtualization**).

Key VMM Operations (Figure 1.13)

VM Multiplexing (*Figure 1.13a*) – Multiple **VMs can share a single physical machine**, improving resource utilization.

VM Suspension (*Figure 1.13b*) – A **VM can be paused and stored** in stable storage for later use.

VM Resumption & Migration (*Figure 1.13c, 1.13d*) – A **suspended VM** can be **resumed or moved to a different hardware platform**, ensuring flexibility in distributed environments.

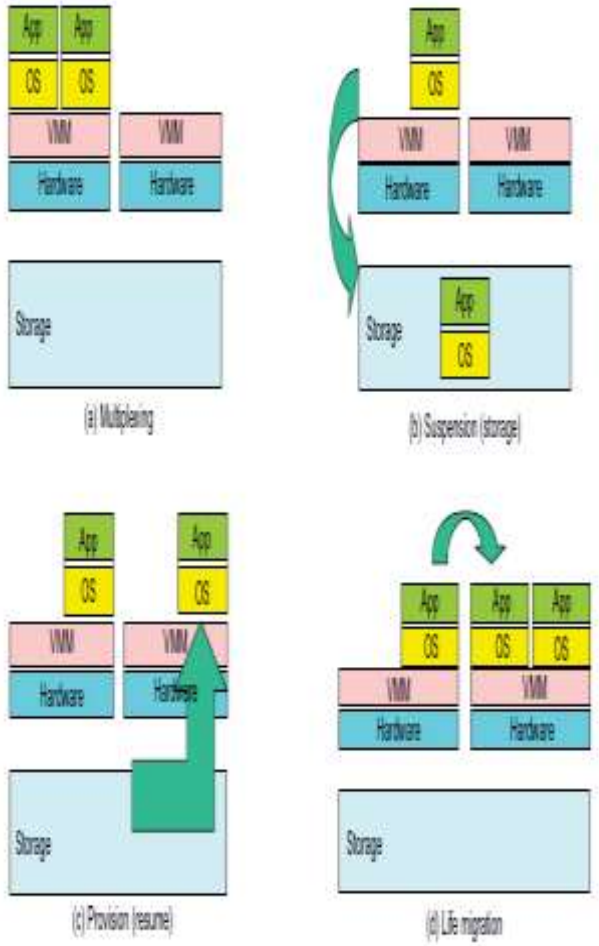


FIGURE 1.13

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

(Courtesy of M. Rosenblum, Keynote address, ACM ASPLOS 2006 [41])

VM Primitive Operations

Benefits of VM-Based Computing

Increased Resource Utilization: VMs **optimize server resources**, consolidating multiple server functions onto fewer machines.

Eliminating Server Sprawl: Traditional **underutilized servers (5–15% efficiency)** can be replaced with **VM deployments achieving 60–80% utilization** (as per VMware).

Seamless Application Portability: VMs enable **transparent workload migration** across hardware platforms, enhancing **cloud computing and data center efficiency**.

Virtual Infrastructures

Separation of Hardware and Software

Physical resources (compute, storage, networking) are **abstracted and mapped** to applications running in **Virtual Machines (VMs)**.

This creates a **virtual infrastructure**, dynamically **connecting resources to distributed applications**.

Benefits of Virtualization

Lower Costs: Reduces hardware expenses by consolidating resources.

Increased Efficiency: Optimizes **server utilization** and **resource allocation**.

Improved Responsiveness: Dynamically provisions resources based on application demand.

Server Consolidation

Virtualization enables **server consolidation and containment**, reducing the number of physical servers while maintaining high availability and performance.

.

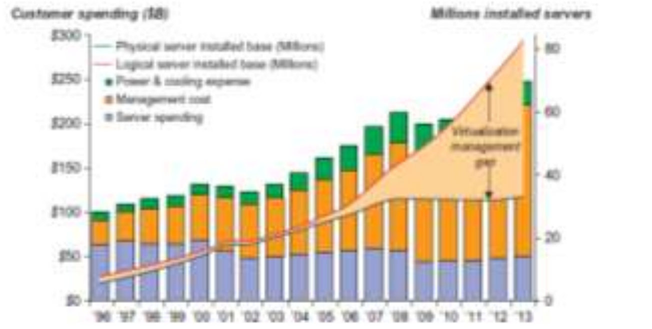


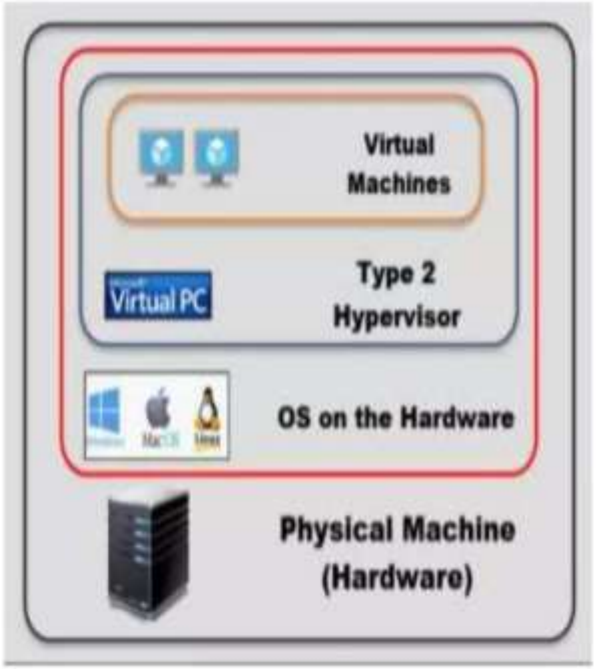
FIGURE 1.14

Growth and cost breakdown of data centers over the years.

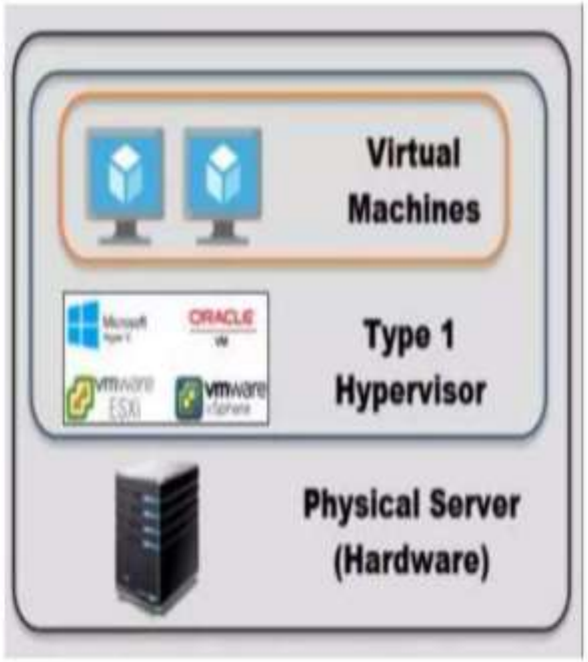
(Source: IDC Report, 2009)

VIRTUAL MACHINES AND VIRTUALIZATION MIDDLEWARE

- a native VM installed with the use of a VMM called a hypervisor in privileged mode The guest OS could be a Linux system and the hypervisor is the server system



HOSTED VIRTUAL MACHINE



NATIVE VIRTUAL MACHINE

2.5.DATA CENTER VIRTUALIZATION FOR CLOUD COMPUTING

- A large data center may be built with thousands of servers.
- Smaller data centers are typically built with hundreds of servers.
- High-end switches or routers may be too cost-prohibitive for building data centers
- Currently, nearly all cloud computing data centers use Ethernet as their fundamental network technology

- 30% of data center cost: **IT Equipment Servers/Disks**
- 33% of data center cost for **chillers**
- 18% of data center cost for **UPS**
- 9% of data center cost for **A.C**
- 7% of data center cost for **lighting in room**

2.5.DATA CENTER VIRTUALIZATION FOR CLOUD COMPUTING

- A large data center may be built with thousands of servers.
- Smaller data centers are typically built with hundreds of servers.
- High-end switches or routers may be too cost-prohibitive for building data centers
- Currently, nearly all cloud computing data centers use Ethernet as their fundamental network technology

- **30% of data center cost: IT Equipment Servers/Disks**
- **33% of data center cost for chillers**
- **18% of data center cost for UPS**
- **9% of data center cost for A.C**
- **7% of data center cost for lighting in room**

Data Center Virtualization for Cloud Computing

Basic architecture and design considerations of data centers

Cloud Architecture Components

Built using **commodity hardware and network devices** for cost efficiency.

Most cloud platforms use **x86 processors**, **low-cost terabyte disks**, and **Gigabit Ethernet**.

Design Priorities

Performance/price ratio is prioritized over raw speed.

Storage capacity and **energy efficiency** are more critical than maximum processing power.

Data Center Growth and Cost Trends

As of **2010**, there were **43 million servers** in use worldwide.

Operational costs (utilities, maintenance, energy) surpass hardware costs within **three years**.

Data Center Growth and Cost Breakdown

Data Center Size and Growth

•**Large data centers** house **thousands of servers**, while **smaller ones** typically have **hundreds**.

•The **cost of building and maintaining** data centers has **increased over time**.

•**Cost Breakdown (IDC Report, 2009)**

•**30%** – IT equipment (**servers, disks**).

•**33%** – **Chiller** (cooling systems).

•**18% – Uninterruptible Power Supply (UPS)**.

•**9% – Computer Room Air Conditioning (CRAC)**.

•**7% – Power distribution, lighting, transformers**.

•**60% of total costs** go toward **management and maintenance**.

•**Rising Power and Cooling Costs**

•**Electricity and cooling costs** increased from **5% to 14% over 15 years**, making energy efficiency a growing concern.

•**Server purchase costs have remained stable**, but **operational expenses continue to rise**.

Data Center Growth and Cost Breakdown

Low-Cost Design Philosophy

Network and Hardware Choices in Data Centers

Cost Constraints in Data Centers

- **High-end switches and routers** are **too expensive** for large-scale cloud infrastructure.
- **Commodity hardware (x86 servers and switches)** is preferred over **mainframes** to balance **cost and scalability**.

Networking in Cloud Data Centers

- **High-bandwidth networks** are often **not economically viable** for cloud environments.
- Instead, **Ethernet** is the **primary network technology** used in almost all cloud data centers.

Role of Software in Managing Infrastructure

- **Software-based solutions** handle **network traffic balancing, fault tolerance, and scalability**, reducing reliance on expensive hardware.

Data Center Growth and Cost Breakdown

Convergence of Technologies

Cloud computing has emerged through the **convergence of four key technology areas**:

Hardware Virtualization & Multi-Core Chips – Enables **dynamic cloud configurations**.

Utility & Grid Computing – Lays the **foundation for cloud infrastructure**.

SOA, Web 2.0, & Mashups – Advances **service integration and scalability**.

Autonomic Computing & Data Center Automation – Improves **efficiency and self-management**.

The Challenge of Data Deluge

Jim Gray highlighted that **science and society face a data explosion** from sensors, simulations, experiments, and web archives.

Managing massive datasets requires **high-performance storage, databases, workflows, and visualization tools**.

Cloud computing is critical for supporting **data-intensive scientific research (e-science/e-research)** across disciplines like **biology, chemistry, physics, and social sciences**.

Data Center Growth and Cost Breakdown

Convergence of Technologies

Cloud Computing as a Transformational Model

Cloud services are provided **on-demand** at different levels:

- **Infrastructure-as-a-Service (IaaS)**
- **Platform-as-a-Service (PaaS)**
- **Software-as-a-Service (SaaS)**

MapReduce provides a **scalable parallel computing model** with **fault tolerance**, widely used for **big data analytics**.

Iterative MapReduce extends this model for **data mining and scientific computing**.

The Future: Convergence of Cloud, Multicore, and Data Science

Cloud computing operates on **large clusters of commodity hardware**, integrating **many-core GPUs** and **multithreading** for high efficiency.

The pipeline of **data → information → knowledge → machine wisdom** is transforming computing into an intelligent **service-oriented architecture (SOA)**.

3.SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

System Models for Distributed and Cloud Computing

- 3.1 Clusters of Cooperative Computers
- 3.2 Grid Computing
- 3.3 Peer-to-Peer Network
- 3.4 Cloud Computing over the Internet

SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

Distributed and cloud computing systems consist of **large-scale autonomous computer nodes**, interconnected through **Storage Area Networks (SANs), Local Area Networks (LANs), or Wide Area Networks (WANs)** in a hierarchical manner.

Scalability and Connectivity

A **few LAN switches** can connect **hundreds of machines** into a **working cluster**.

WANs link multiple local clusters to form **large-scale distributed systems**.

Massive systems can reach **millions of connected computers**, supporting **web-scale applications.**

SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

Classification of Massive Systems (Table 1.2)

Four main categories of distributed systems:

- **Clusters** – Tightly coupled computers working as a single system.
- **Peer-to-Peer (P2P) Networks** – Decentralized systems where nodes communicate directly.
- **Computing Grids** – Distributed computing environments that share resources across organizations.
- **Internet Clouds** – Large-scale, virtualized data centers offering computing and storage as services.

System Collaboration and Technical Aspects

- These systems operate **cooperatively** or **collaboratively** at various levels, depending on the **application and architecture**.
- They involve **hundreds, thousands, or even millions of nodes**, working together to provide **scalable and efficient computing resources**.

Clusters in Supercomputing

•**Clusters dominate supercomputing**, with **417 out of the Top 500 supercomputers (2009)** using cluster architecture.
•Clusters serve as the **foundation for large-scale grids and cloud computing**.

•**Peer-to-Peer (P2P) Networks in Business**
•**P2P networks** are widely used in **business applications**.
•However, the **content industry resisted P2P adoption** due to **copyright concerns in ad hoc networks**.

•**Grid Computing Challenges**
•Many **national grid projects** were **underutilized** due to **lack of reliable middleware** and **well-optimized applications**.
•**Advantages of Cloud Computing**
•**Lower costs** and **ease of use** make cloud computing attractive for **both providers and users**.

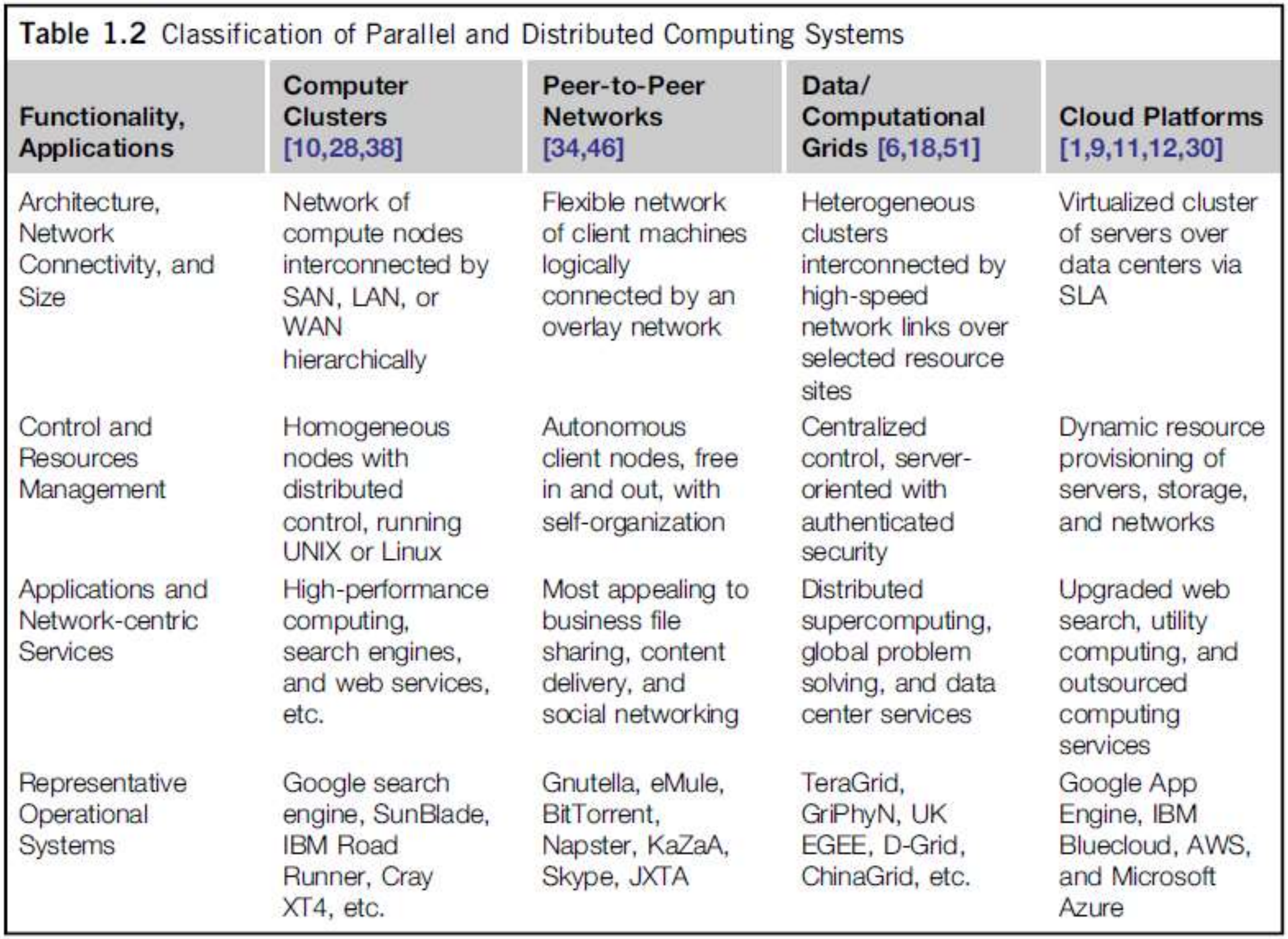


Table 1.2 Classification of Parallel and Distributed Computing Systems

Functionality, Applications	**Computer Clusters** [10,28,38]	**Peer-to-Peer Networks** [34,46]	**Data/ Computational Grids** [6,18,51]	**Cloud Platforms** [1,9,11,12,30]
Architecture, Network Connectivity, and Size	Network of compute nodes interconnected by SAN, LAN, or WAN hierarchically	Flexible network of client machines logically connected by an overlay network	Heterogeneous clusters interconnected by high-speed network links over selected resource sites	Virtualized cluster of servers over data centers via SLA
Control and Resources Management	Homogeneous nodes with distributed control, running UNIX or Linux	Autonomous client nodes, free in and out, with self-organization	Centralized control, server-oriented with authenticated security	Dynamic resource provisioning of servers, storage, and networks
Applications and Network-centric Services	High-performance computing, search engines, and web services, etc.	Most appealing to business file sharing, content delivery, and social networking	Distributed supercomputing, global problem solving, and data center services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc.	Google App Engine, IBM Bluecloud, AWS, and Microsoft Azure

3.1 Clusters of Cooperative Computers

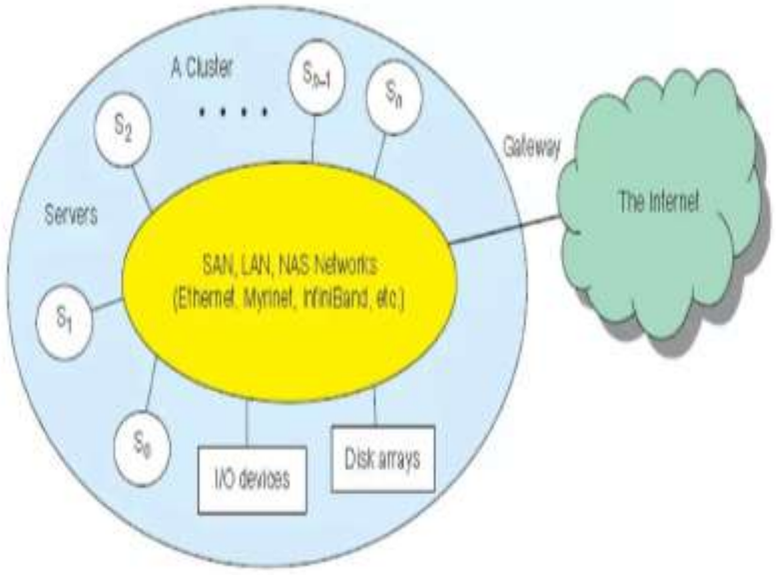


FIGURE 1.15

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.

Clustering means that multiple servers are grouped together to achieve the same service

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

Clusters of Cooperative Computers

Cluster Architecture

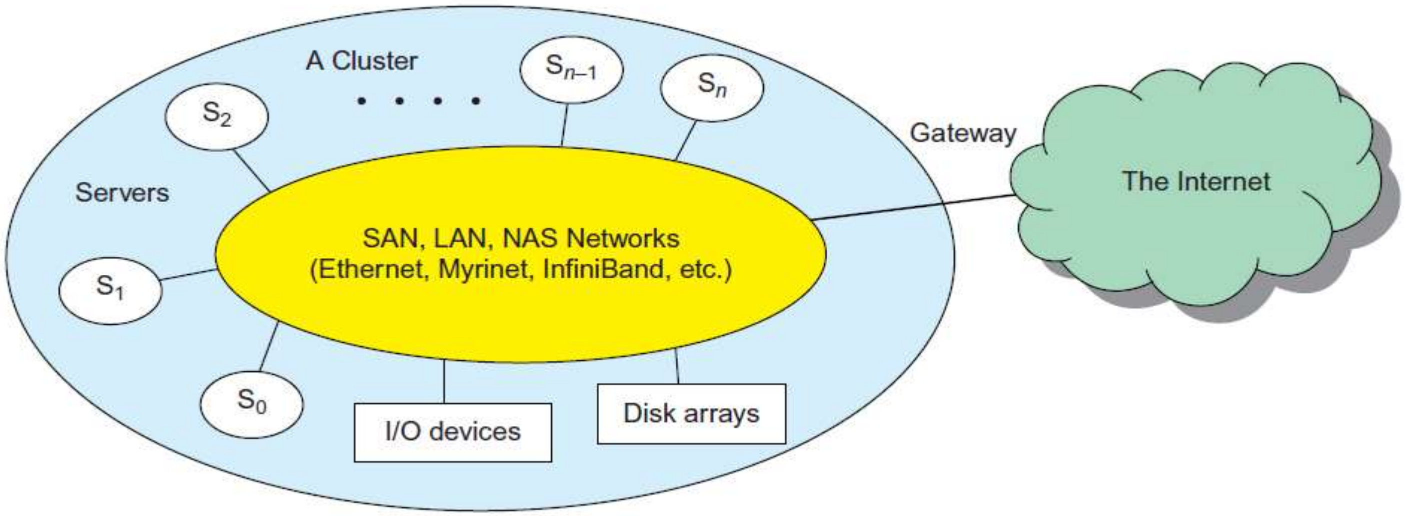


FIGURE 1.15

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.

A **server cluster** is built around a **low-latency, high-bandwidth interconnection network**, which can be:

Storage Area Network (SAN) (e.g., Myrinet).

Local Area Network (LAN) (e.g., Ethernet).

Scalability and Network Hierarchy

Larger clusters use **multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches.**

Clusters of Cooperative Computers

Cluster Architecture

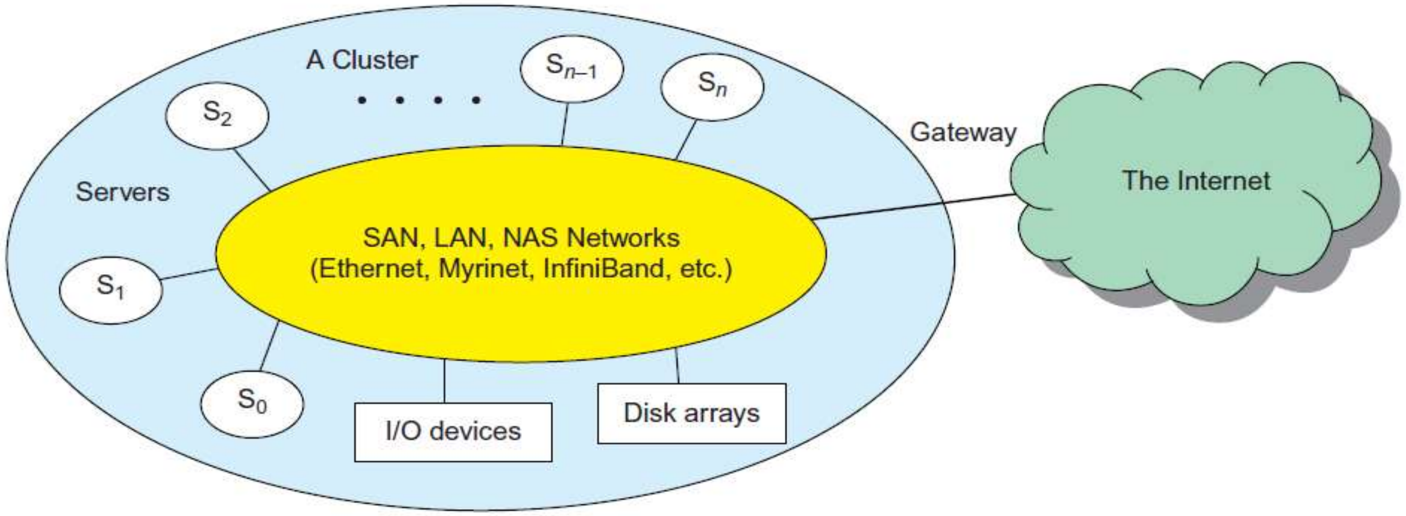


FIGURE 1.15

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 can be **scaled hierarchically** using **SAN, LAN, or WAN.**

The cluster connects to the **Internet via a VPN gateway**, identified by an **IP address**.

System Image and OS Management

A cluster's **system image** is determined by how the **OS manages shared resources**.

Most clusters are loosely coupled, meaning each **server node has its own OS**. This results in **multiple system images**, as each node operates **autonomously**.

Single-System Image

Ideal Cluster Design

Greg Pfister proposed that an ideal cluster should integrate multiple system images into a Single-System Image (SSI).

SSI clusters create an illusion of a unified system, making multiple nodes appear as one powerful machine.

Benefits of SSI

Centralized resource management: Enables CPU, memory, and I/O sharing across all nodes.

Simplifies user experience: The cluster appears as a single entity rather than separate independent machines.

Clusters Without SSI

A cluster with multiple system images operates as a collection of independent computers, lacking seamless resource integration.

SSI improves cluster efficiency, resource sharing, and user experience, making clusters function as a single, unified system rather than loosely connected independent nodes.

Hardware, Software, and Middleware Support

MPP Clusters in HPC

Clusters designed for massive parallelism are called Massively Parallel Processors (MPPs).

Nearly all Top 500 HPC clusters are MPP-based.

Building Blocks of Clusters

Computer Nodes: PCs, workstations, servers, or Symmetric Multiprocessing (SMP) systems.

Networking: High-bandwidth networks like Gigabit Ethernet, Myrinet, InfiniBand.

Communication Software: Uses PVM (Parallel Virtual Machine) or MPI (Message Passing Interface).

Operating System: Most clusters run Linux.

Cluster Middleware and Resource Management

Middleware is required for Single-System Image (SSI) and High Availability (HA).

Supports both sequential and parallel applications.

Distributed memory clusters use Distributed Shared Memory (DSM) to provide a unified memory view.

Achieving full SSI is costly and complex, leading many clusters to remain loosely coupled.

Hardware, Software, and Middleware Support

Virtual Clusters and On-Demand Scalability

Virtualization enables dynamic virtual clusters to be created on demand, improving resource flexibility and user customization.

MPP clusters dominate HPC, requiring special networking, middleware, and parallel environments. While SSI is challenging, virtual clusters offer scalable and flexible computing solutions.

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. Without this middleware, cluster nodes cannot work together effectively to achieve cooperative computing.
- The software environments and applications must rely on the middleware to achieve high performance. The cluster benefits come from scalable performance, efficient message passing, high system availability, seamless fault tolerance, and cluster-wide job management,

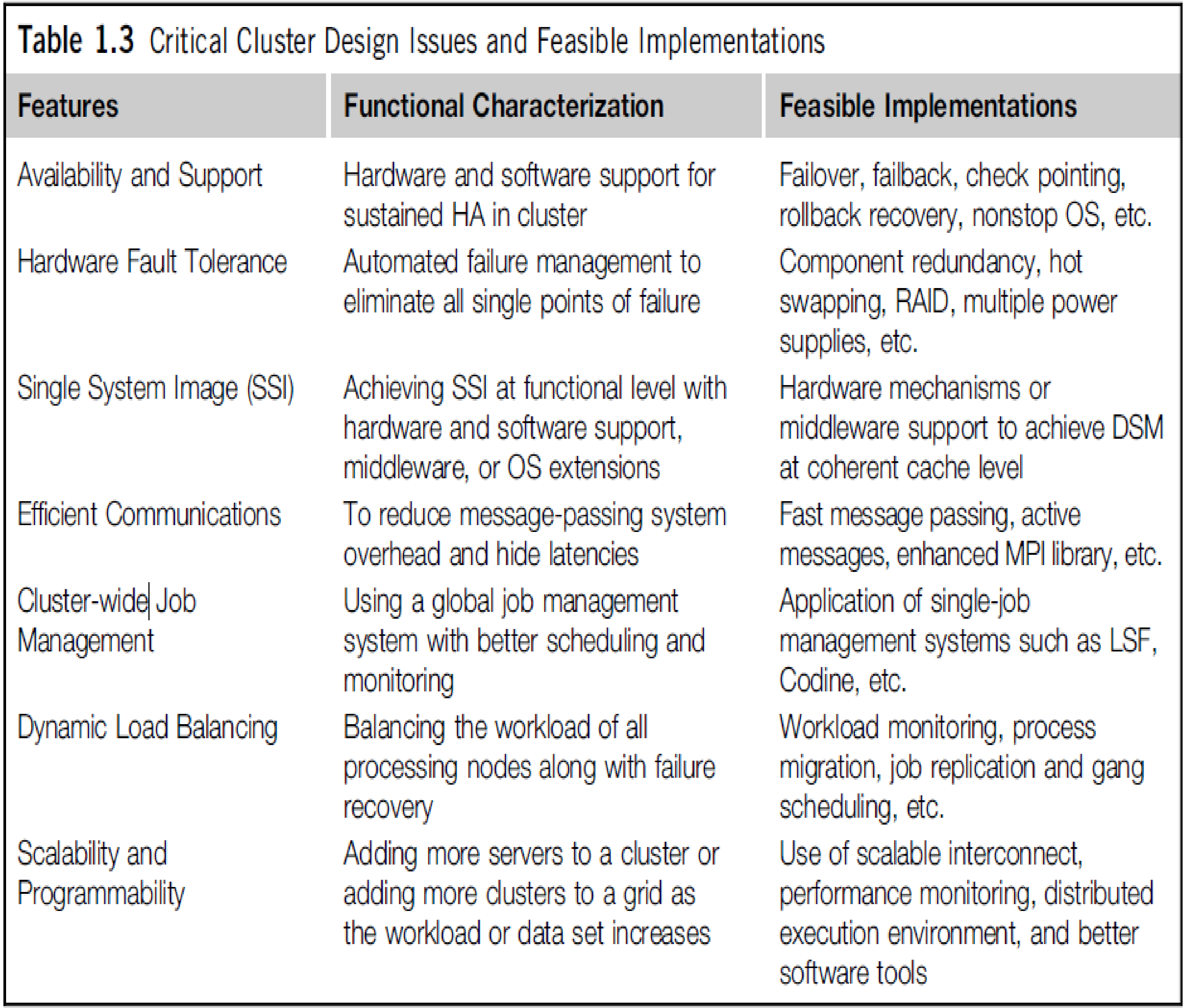


Table 1.3 Critical Cluster Design Issues and Feasible Implementations

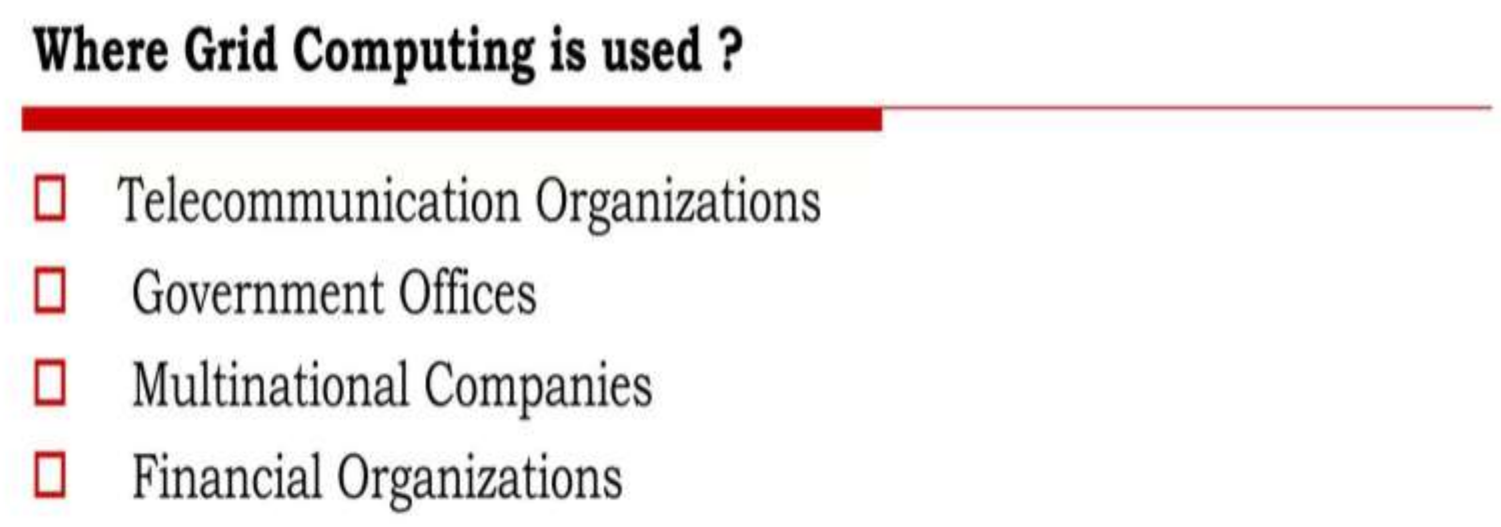
Features	Functional Characterization	Feasible Implementations
Availability and Support	Hardware and software support for sustained HA in cluster	Failover, failback, check pointing, rollback recovery, nonstop OS, etc.
Hardware Fault Tolerance	Automated failure management to eliminate all single points 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

3.2 Grid Computing

Grid Computing is a subset of distributed computing

Grid Computing is a technique in which the idle systems in the Network and their " wasted " CPU cycles can be efficiently used by **uniting pools of servers, storage systems and networks** into a single large virtual system for resource sharing dynamically at runtime.

Example : A small LAN that consists of around 20 systems out of which 10 systems are idle and 5 systems are using less amount of CPU and their CPU cycles are wasted. **Now grid computing** efficiently utilize those "wasted CPU cycles" into "working cycles".



Where Grid Computing is used ?

- Telecommunication Organizations
- Government Offices
- Multinational Companies
- Financial Organizations

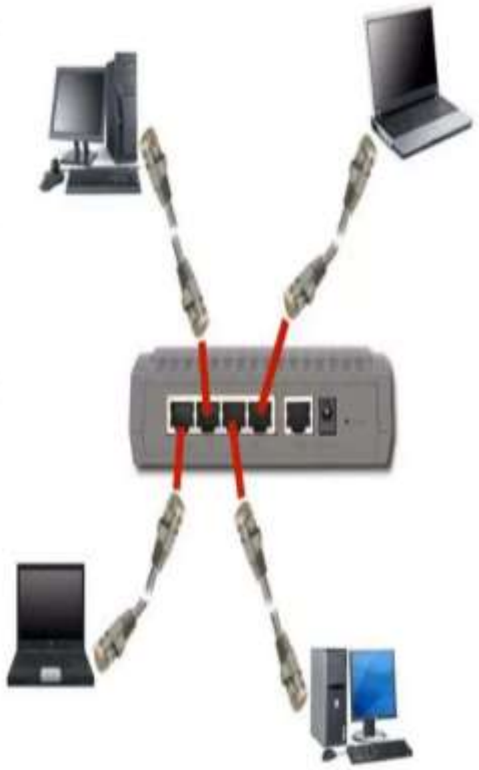
In past 30 years, users have experienced a natural growth path from Internet to web and grid computing services. Internet services such as the Telnet command enables a local computer to connect to a remote computer. A web service such as HTTP enables remote access of remote web pages. Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.

Advantages of Grid computing

- It can solve larger and complex problems in a short time.
- No computer be idle in this system.
- Unit works makes all computer like a super computer.
- Make better uses of hardware.

3.3 Peer-to-Peer Network

- In a **P2P network**, the "**peers**" are computer systems which are connected to each other via the Internet.
- Files can be shared directly between systems on the **network** without the need of a central server.
- In other words, each computer on a **P2P network** becomes a file server as well as a client
- when you create an ad-hoc **network** between two computers, you create a **peer-to-peer network** between them

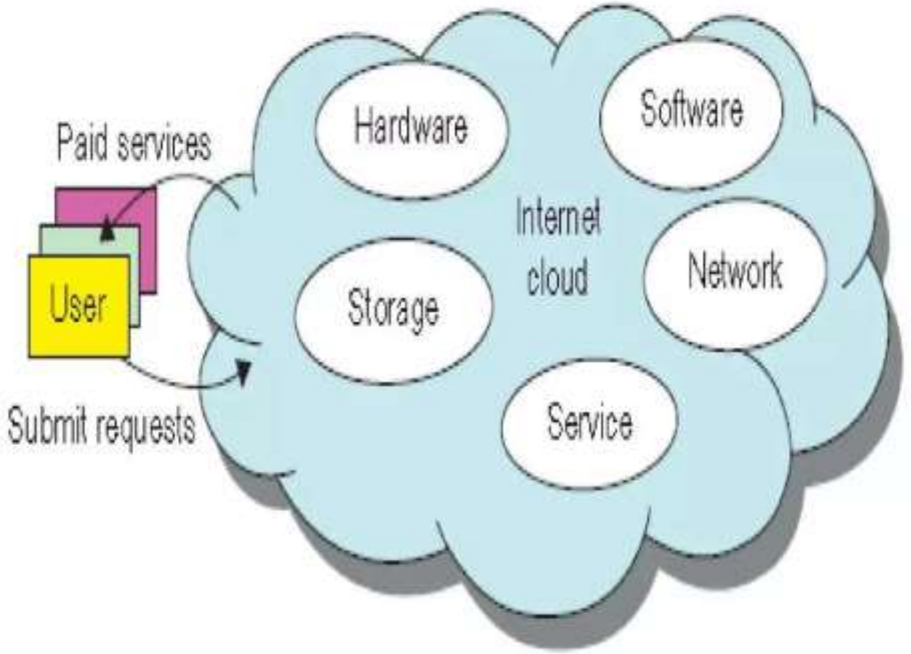


Peer-to-Peer Network

Key advantages of a P2P network

- **Easy file sharing:** An advanced P2P network can share files quickly over large distances.
- **Reduced costs:** There is no need to invest in a separate computer for a server when setting up a P2P network.
- **Adaptability**: P2P network extends to include new clients easily.

3.4 Cloud Computing over the Internet



Infrastructure as a Service (IaaS)

servers, storage, networks, and the data center

Platform as a Service (PaaS)

Middleware, databases, development tools, and some runtime support such as Web 2.0 and Java

Software as a Service (SaaS)

Applied to Business, Industry applicaions, ERP,HR,CRM

Cloud Computing over the Internet

Internet clouds offer Three deployment modes:

- ✓ Private Cloud
- ✓ Public Cloud
- ✓ Hybrid Cloud

4 Software Environments for Distributed Systems and Clouds

Software Environments for Distributed Systems and Clouds

Popular software environments for using distributed and cloud computing systems

- Service-Oriented Architecture (SOA)
- Layered Architecture In web services
- Java RMI and CORBA

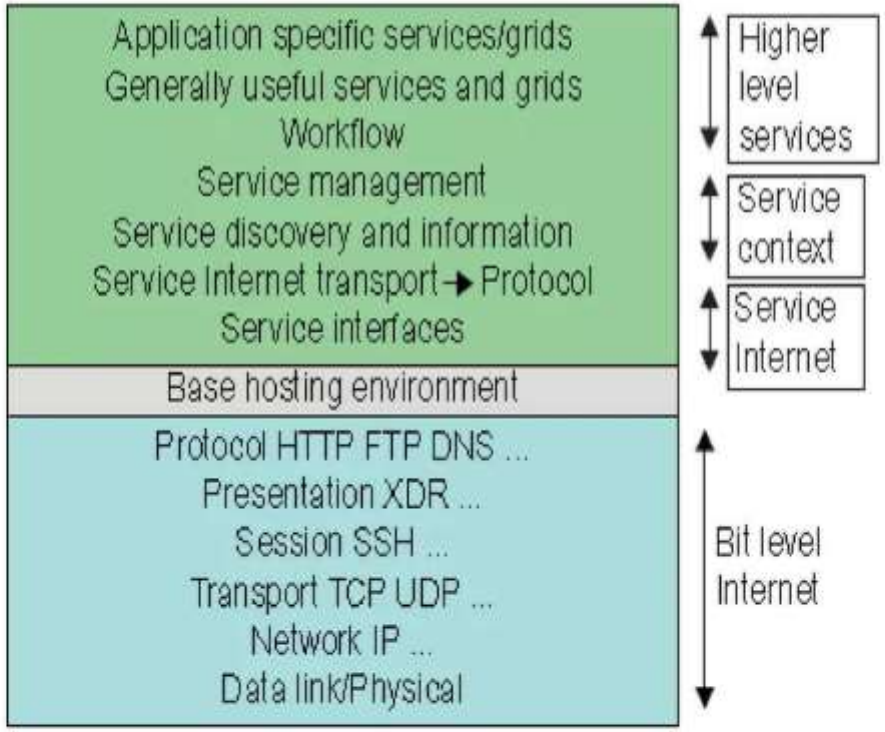
Software Environments for Distributed Systems and Clouds

Service-oriented architecture (SOA)

A collection of services. These services communicate with each other

- SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds

Layered Architecture In web services



4.2 Trends toward Distributed Operating Systems

- One interface to all resources in the network

4.3 Parallel and Distributed Programming Models

Message Passing Interface

MapReduce

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples.

Hadoop Library

Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models

5. Performance, Security, and Energy Efficiency

5.1 Performance Metrics and Scalability Analysis .

5.2 Fault Tolerance and System Availability

5.3 Network Threats and Data Integrity

5.4 Energy Efficiency in Distributed Computing

5 .1.Performance Metrics and Scalability Analysis

- Performance metrics are needed to measure various distributed systems
- Measure CPU speed in MIPS and network bandwidth in Mbps

Dimensions of Scalability

- ✓ Size scalability
- ✓ Software scalability
- ✓ Application scalability
- ✓ Technology scalability

5.2 Fault Tolerance and System Availability

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure some of its components.

System Availability =MTTF/MTTF +MTTR

mean time to failure (MTTF)

mean time to repair(MTTR)

5.3 Network Threats and Data Integrity

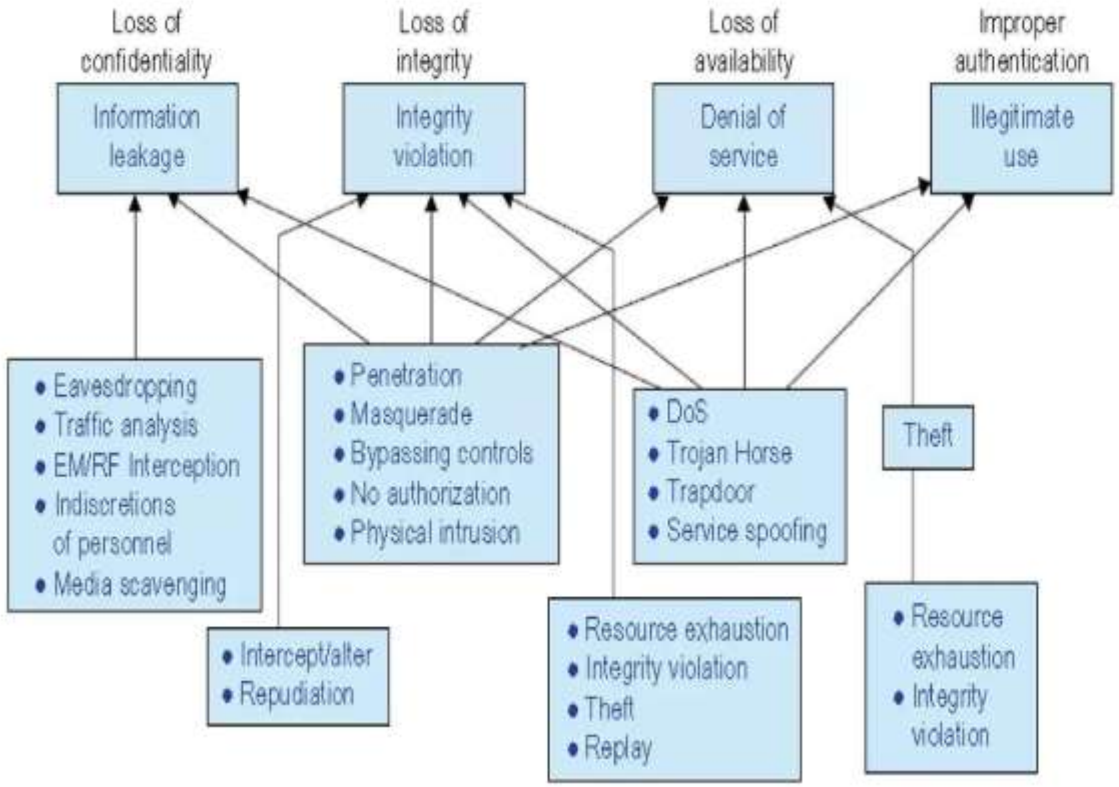
Three security requirements are often considered:

confidentiality,

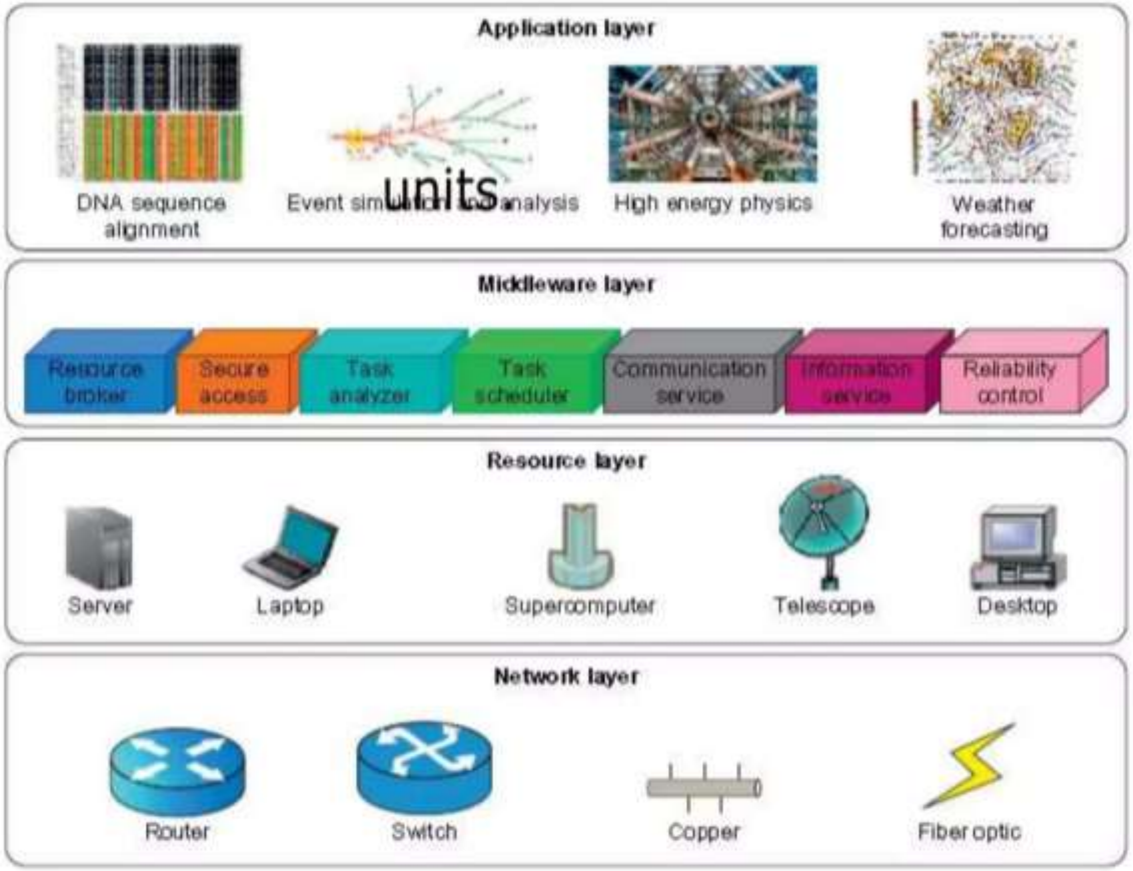
integrity,

availability

for most Internet service providers and cloud users.



5.4.Energy Efficiency in Distributed Computing



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

This layer provides resource broker, communication service, task analyzer, task scheduler, security access, reliability control, and information service

The resource layer consists of a wide range of resources including computing nodes and storage