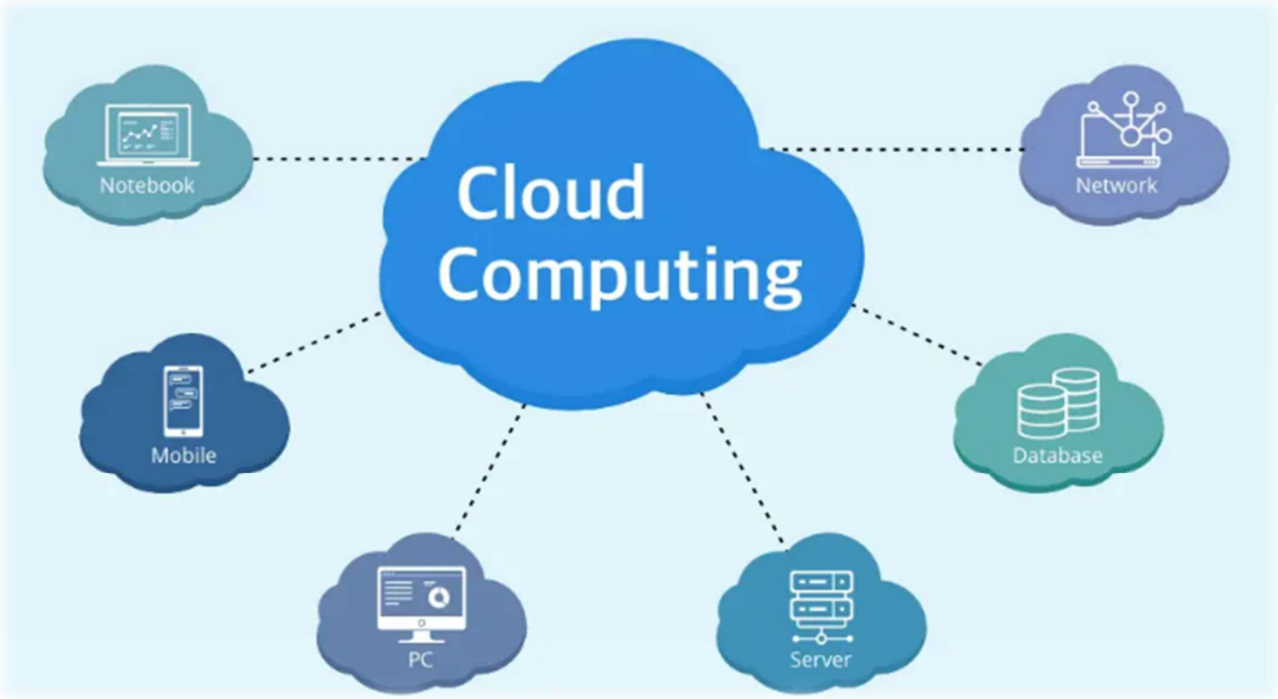
COURSE OBJECTIVES_01

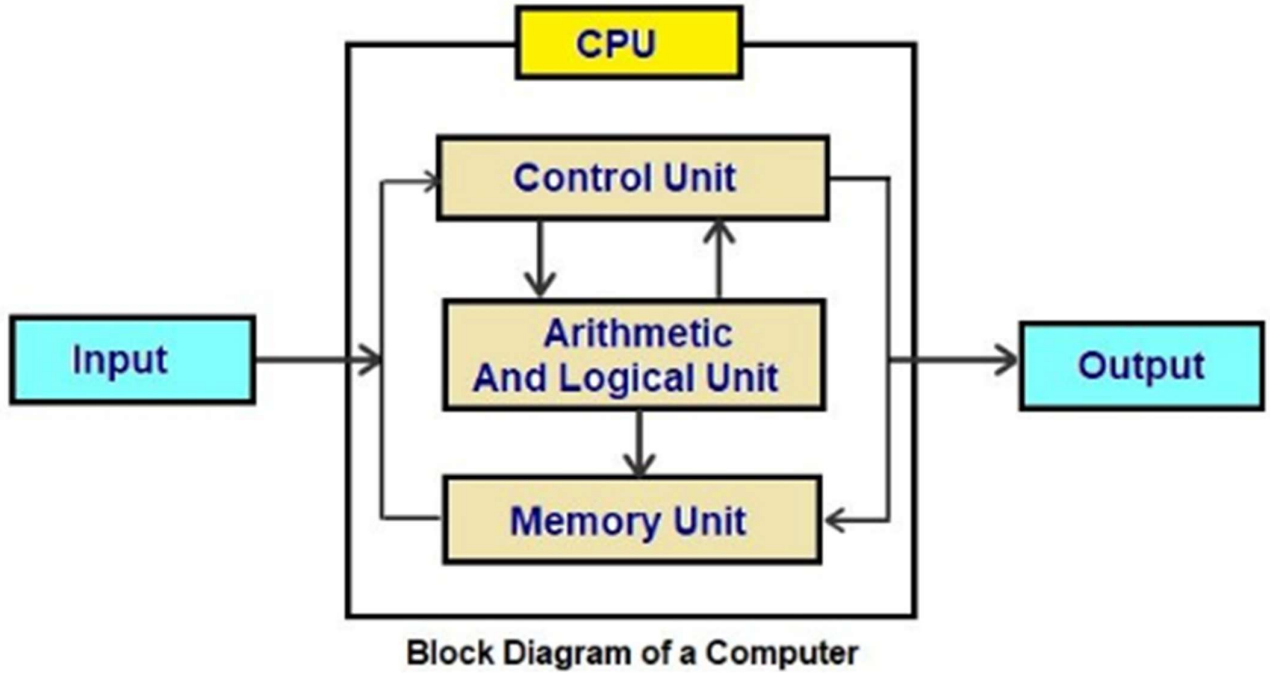
History of Computing Paradigms

- Overview of Distributed Computing
- Cluster Computing
- Grid Computing
- Ubiquitous Computing
- Peer-to-Peer Computing
- Distributed System Models
- Enabling Technologies



Definition of Computer

A computer is an electronic device that can accept data (input), process it using instructions (programs), store the data, and produce meaningful results (output).



Block Diagram of a Computer

Key Features:

1. Electronic – works using electricity and electronic components.
2. Programmable – follows instructions (software/programs).
3. Automatic – once given instructions, it works without human intervention.
4. Versatile – can perform a wide variety of tasks (calculation, communication, multimedia, AI, etc.).
5. Fast & Accurate – processes data quickly and correctly (errors usually come from humans, not the machine).
6. Storage – can store vast amounts of information for future use.

Paradigm – Definition

A paradigm is a model, pattern, or way of thinking/doing something that serves as an example or framework.

It's like a set of rules, methods, and concepts that guide how we approach and solve problems.

What is a Computing Paradigm?

A **computing paradigm** is a model, style, or approach to computing that defines how problems are solved using computers. Over time, computing paradigms have evolved alongside advances in technology, architecture, and problem-solving needs.

HISTORY OF COMPUTING PARADIGMS

1. Mechanical Computing (Pre-1940s)

- ✓ Devices: **Abacus**, **Pascal's Calculator**, **Babbage's Analytical Engine**.
- ✓ Characteristics:
  - No electricity, mechanical gears and levers.
  - Basis for later programmable machines.
- ✓ Paradigm: **Manual mechanical computation**.

2. Electromechanical Computing (1930s–1940s)

- ✓ Devices: **Zuse Z3**, **Harvard Mark I**.
- ✓ Characteristics:
  - Used relays and mechanical switches.
  - Slow, bulky, but programmable.
- ✓ Paradigm: **Early automation of computation**.

3. First Generation: Vacuum Tube Computers (1940s–1950s)

Examples: **ENIAC**, **UNIVAC**.

- ✓ Characteristics:
  - Vacuum tubes for processing.
  - Machine language programming.
  - Very large and power-hungry.
- ✓ Paradigm: **Numerical computation & basic stored-program concept**.

4. Second Generation: Transistor Computers (1950s–1960s)

- ✓ Characteristics:
  - Faster, smaller, more reliable than vacuum tubes.
  - Assembly language & early high-level languages (COBOL, FORTRAN).
- ✓ Paradigm: **Commercial data processing + scientific computation**.

5. Third Generation: Integrated Circuit (IC) Computers (1960s–1970s)

- ✓ Characteristics:
  - Use of ICs improved speed, cost, and reliability.
  - Multiprogramming and time-sharing OS introduced.
  - Minicomputers became popular.
- ✓ Paradigm: **Batch & time-sharing computing**.

6. Fourth Generation: Microprocessor-based Computers (1970s–1990s)

- ✓ Examples: **IBM PCs, Apple II, personal computers**.
- ✓ Characteristics:
  - Microprocessors allowed small, cheap, personal computing.
  - Graphical user interfaces, networks.
  - Rise of software industry.
- ✓ Paradigm: **Personal computing & client-server model**.

7. Fifth Generation: Parallel and Distributed Computing (1980s–2000s)

✓ Characteristics:
- Parallel processors, clusters, and supercomputers.
- Distributed systems & Internet growth.
- Object-oriented programming, AI research.

✓ Paradigm: **Networking, parallelism, and knowledge-based computing**.

8. Sixth Generation: Cloud & Mobile Computing (2000s–2010s)

✓ Characteristics:
- Cloud services (AWS, Azure, GCP).
- Mobile devices and ubiquitous computing.
- Virtualization, SaaS, and big data.

✓ Paradigm: **On-demand computing & pervasive access**.

9. Seventh Generation: AI, Quantum & Edge Computing (2010s–Present)

✓ Characteristics:
- AI-driven computing (machine learning, deep learning).
- Quantum computing research.
- Edge & IoT devices for real-time processing.

✓ Paradigm: **Intelligent, distributed, and quantum-enhanced computing**.

🌀 **Summary of Paradigm Evolution**

✓ **Mechanical** → Manual aids to calculation.
✓ **Electromechanical** → Automated mechanical computation.
✓ **Electronic (Vacuum Tubes)** → Stored-program concept.
✓ **Transistor-based** → Commercial and scientific expansion.
✓ **IC-based** → Multi-user systems and minicomputers.
✓ **Microprocessors** → Personal computing revolution.
✓ **Parallel & Distributed** → Networking and Internet era.
✓ **Cloud & Mobile** → On-demand global services.
✓ **AI & Quantum** → Future of intelligent and scalable computing.

OVERVIEW OF DISTRIBUTED COMPUTING

Distributed Computing is a **computing paradigm** in which multiple independent computers (often geographically separated) work together as a single system to solve a problem or perform tasks. Instead of relying on a single powerful machine, the workload is divided across several computers (called **nodes**) that communicate and coordinate through a network.

1. **Definition**:
   Distributed computing is the study, design, and implementation of systems where components located on different networked computers communicate and coordinate their actions by passing messages.

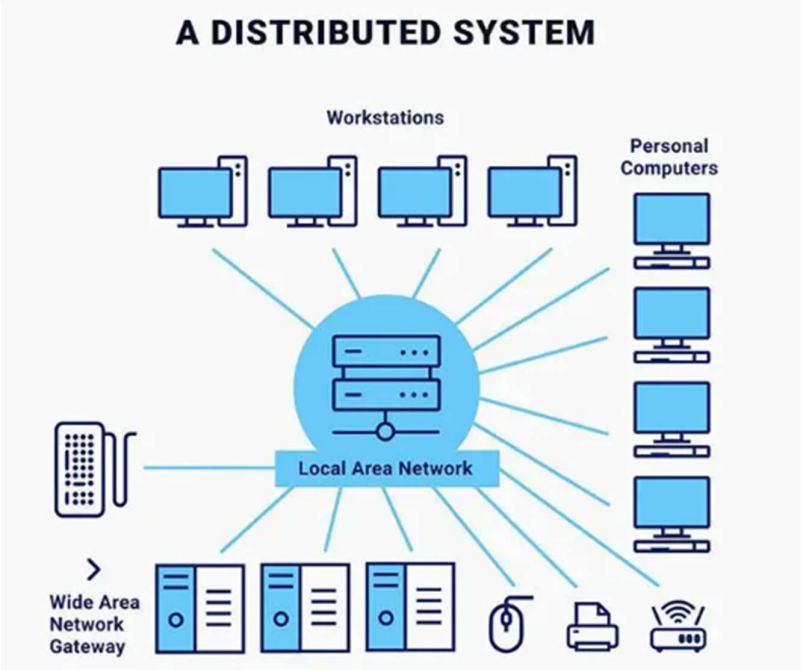
2. **Main Idea**:
   Divide a large problem into smaller subtasks, process them on multiple machines, and combine the results.

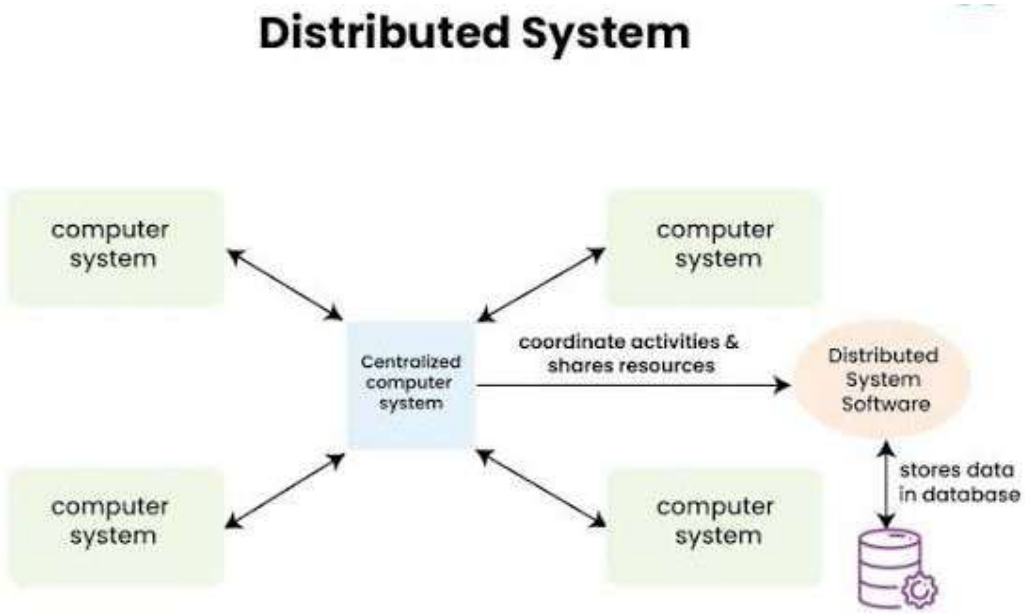
3. **Features**:
   - **Multiple independent computers** (nodes).
   - **Resource sharing** (CPU, memory, storage, etc.).
   - **Concurrency** (tasks run in parallel).
   - **Scalability** (can add more nodes to handle bigger tasks).
   - **Fault tolerance** (system continues even if some nodes fail).

4. **Examples**:
   - Google Search uses distributed systems to process billions of queries.
   - Online multiplayer games.
   - Cloud computing platforms (AWS, Azure, GCP).
   - Blockchain networks.

5. **Difference from Centralized Computing**:
   - *Centralized*: All computation happens on one powerful machine.
   - *Distributed*: Workload is spread across multiple machine

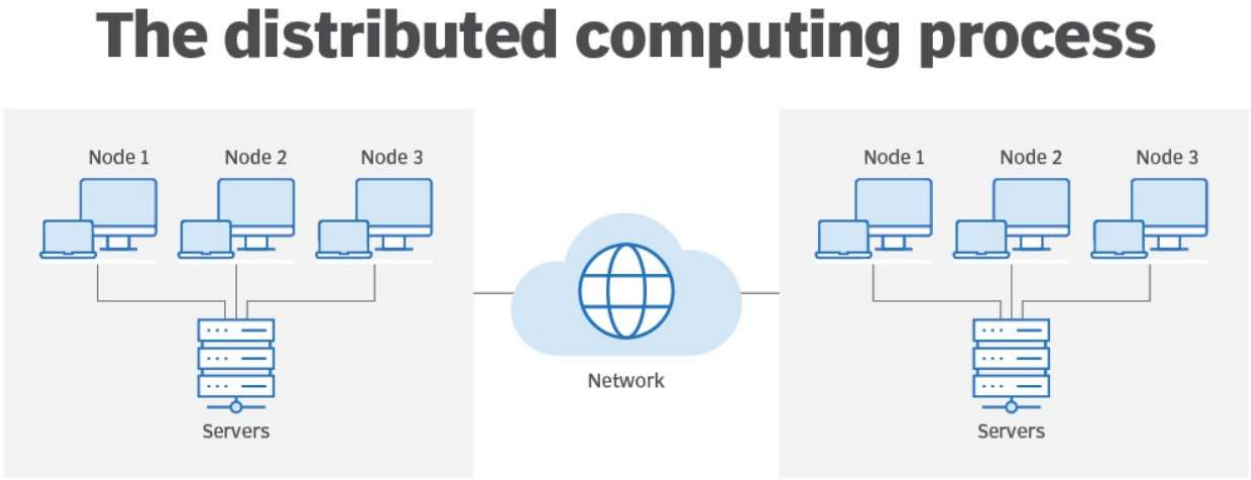
Distributed computing is a field of computer science in which components of a software system are shared among multiple networked computers to achieve a common goal. This approach treats a network of computers as a single, powerful system, enabling the efficient processing of large workloads and providing several advantages over monolithic systems.





How Distributed computing works

Instead of relying on a single, powerful machine, a distributed system splits large tasks into smaller subtasks, which are then assigned to different computers, or nodes. The nodes communicate with each other over a network to coordinate their work and share resources. After completing their tasks, they send the results back to a central node or aggregator, which combines them to produce the final output.



Key characteristics

- **Resource sharing:** Nodes can share hardware (like printers), software, and data, increasing resource utilization and performance.
- **Scalability:** Systems can be easily scaled horizontally by adding more nodes to the network. This allows them to handle increased demand and larger workloads without requiring a single, costly machine upgrade.
- **Fault tolerance:** If one node fails, other nodes in the system can take over its work, preventing a single point of failure from causing a total system crash.
- **High availability:** Because services are often replicated across multiple nodes, the system can continue to operate even if some components fail, ensuring continuous service.
- **Concurrency:** Multiple tasks can be executed simultaneously across different nodes, which significantly improves overall processing speed and efficiency.

Common architectural models

- **Client-server:** The most traditional model, where clients (e.g., web browsers) request resources from central servers. This is used for applications like email and web browsing.
- **Peer-to-peer (P2P):** A decentralized model where each node can function as both a client and a server. It is known for its resilience and is commonly used for file sharing and blockchain networks.
- **N-tier architecture:** An extension of the client-server model that divides an application into multiple logical layers, such as the presentation layer, application layer, and data layer. This separation of concerns improves scalability and maintainability.
- **Microservices:** An architectural style where an application is built as a collection of independent, loosely coupled services. Each service handles a specific business function and can be developed, deployed, and scaled independently.

Challenges of distributed computing

While powerful, distributed systems introduce unique complexities and challenges:

- **Complexity:** Managing multiple interconnected components across a network is inherently more complex than managing a single system. Debugging, monitoring, and maintaining these systems require specialized tools and expertise.
- **Network dependency:** The system's reliance on network communication means that performance and reliability can be affected by network latency, bandwidth limitations, and connection issues.
- **Data consistency:** Ensuring that replicated data remains consistent across all nodes, especially with concurrent updates, can be difficult. The CAP theorem highlights the trade-offs between consistency, availability, and partition tolerance.
- **Security:** With data and processes spread across multiple nodes and networks, distributed systems present a larger attack surface. Implementing robust security measures is more challenging than in a centralized system.
- **Coordination:** Coordinating tasks and maintaining synchronization across independent nodes without a global clock is challenging and requires complex algorithms.

Applications and use cases

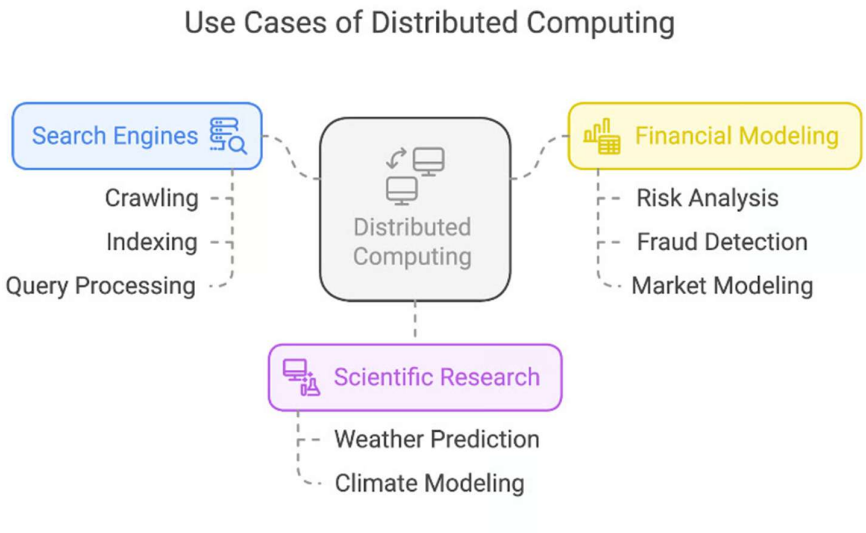
Distributed computing is a foundational technology for many modern applications and services:

- **Cloud computing:** Cloud service providers like AWS, Azure, and Google Cloud use distributed systems to provide scalable and resilient services.
- **Big data:** Frameworks like Apache Hadoop and Spark use distributed processing to analyze massive datasets.
- **Financial services:** Distributed systems support high-frequency trading, risk management, and secure financial transactions.
- **Content delivery networks (CDNs):** CDNs distribute content across servers located closer to users, reducing latency and improving access speeds.
- **Scientific research:** Large-scale projects, such as the Human Genome Project, use distributed computing to analyze massive amounts of data.

Use Cases of Distributed Computing

Distributed computing powers some of the most impactful applications in our world today. Let's look at three common use cases: search engines, scientific research, and financial modelling.

01.Search engines



Search engines like Google rely heavily on distributed computing to crawl and index billions of web pages. Instead of assigning the task to a single machine, these systems divide the workload among many, many nodes.

Some nodes focus on crawling web pages, others handle indexing, and another group processes user queries in real time. This division of labor ensures fast and efficient search results, no matter the scale of the task.

02. Scientific research

In scientific research, distributed computing enables breakthroughs by running complex simulations and analyzing huge datasets. For instance, climate scientists use distributed systems to model and predict weather patterns or simulate the effects of global warming.

03. Financial modeling

The financial industry relies on distributed computing for tasks like risk analysis, fraud detection, and market modelling

CLUSTER COMPUTING

Cluster Computing – Definition

Cluster computing is a type of **distributed computing** where a group of interconnected, usually **homogeneous (similar)** computers work together as a single system.

These computers (called **nodes**) are tightly coupled and connected through a **local area network (LAN)**. They cooperate to perform high-performance tasks like simulations, data analysis, or scientific computations.

Key Features of Cluster Computing

1. **Homogeneous Systems** – Nodes usually have similar hardware and operating systems.
2. **Tightly Coupled** – Nodes are located close to each other (same data center/lab).
3. **High Performance** – Used for scientific research, engineering, weather forecasting, etc.
4. **Single System Image (SSI)** – Appears as one powerful computer to the user.
5. **Fault Tolerance** – If one node fails, others can take over.

Types of Clusters

1. **Load-Balancing Clusters** – Distribute workloads across nodes to optimize performance.
2. **High-Performance Clusters (HPC)** – Used for complex computations that need lots of processing power.
3. **High-Availability Clusters (HA)** – Provide redundancy so the system keeps working even if some nodes fail.

Examples

- Google's early search engine was built on Linux clusters.
- Weather forecasting supercomputers.
- Banking and stock exchange servers for fault tolerance.
- Universities' supercomputers for scientific simulations.

Introduction :

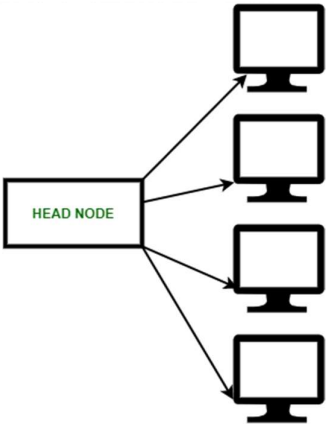
Cluster computing is a collection of tightly or loosely connected computers that work together so that they act as a single entity. The connected computers execute operations all together thus creating the idea of a single system. The clusters are generally connected through fast local area networks (LANs)

Cluster Computing

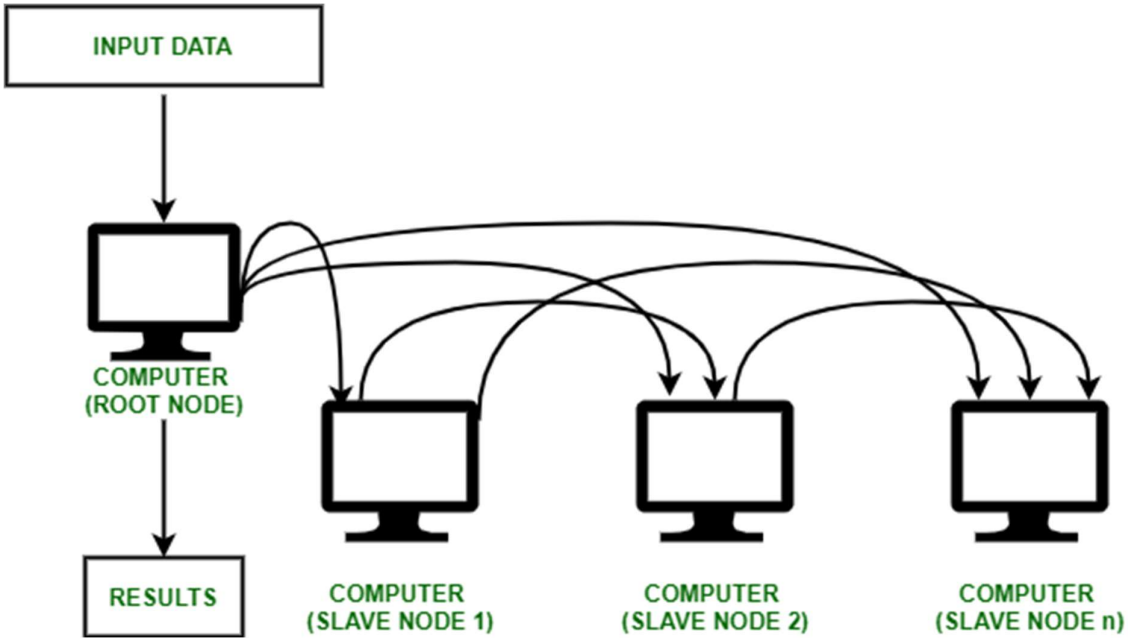
Why is Cluster Computing important?

Cluster computing gives a relatively inexpensive, unconventional to the large server or mainframe computer solutions.

1. It resolves the demand for content criticality and process services in a faster way.
2. Many organizations and IT companies are implementing cluster computing to augment their scalability, availability, processing speed and resource management at economic prices.
3. It ensures that computational power is always available.
4. It provides a single general strategy for the implementation and application of parallel high-performance systems independent of certain hardware vendors and their product decisions.



A Simple Cluster Computing Layout



Types of Cluster computing :

1. High performance (HP) clusters :
HP clusters use computer clusters and supercomputers to solve advance computational problems. They are used to performing functions that need nodes to communicate as they perform their jobs. They are designed to take benefit of the parallel processing power of several nodes.

2. Load-balancing clusters :
Incoming requests are distributed for resources among several nodes running similar programs or having similar content. This prevents any single node from receiving a disproportionate amount of task. This type of distribution is generally used in a web-hosting environment.

3. High Availability (HA) Clusters :

HA clusters are designed to maintain redundant nodes that can act as backup systems in case any failure occurs. Consistent computing services like business activities, complicated databases, customer services like e-websites and network file distribution are provided. They are designed to give uninterrupted data availability to the customers.

Classification of Cluster :

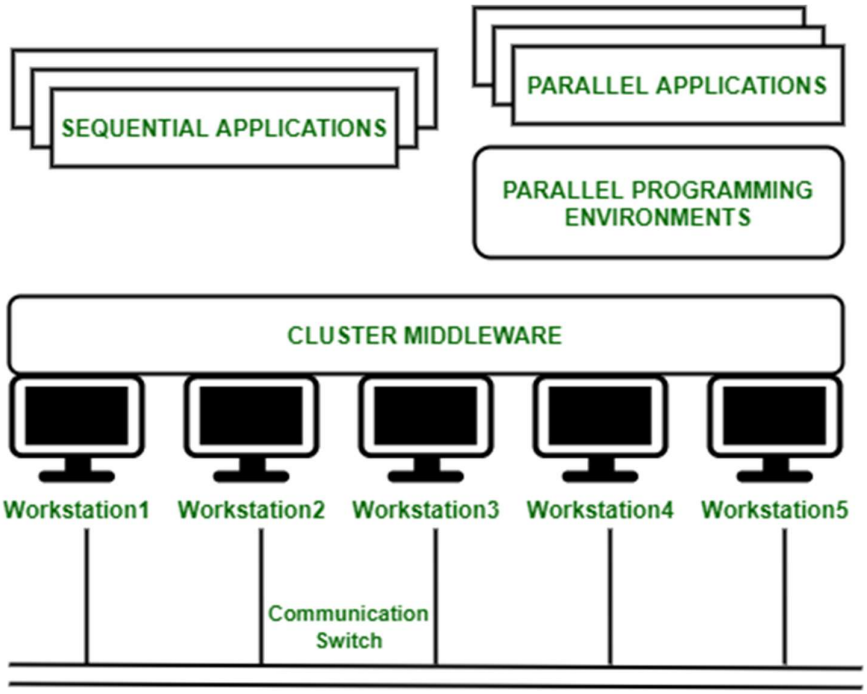
1. Open Cluster :

IPs are needed by every node and those are accessed only through the internet or web. This type of cluster causes enhanced security concerns.

2. Close Cluster :

The nodes are hidden behind the gateway node, and they provide increased protection. They need fewer IP addresses and are good for computational tasks.

Cluster Computing Architecture :

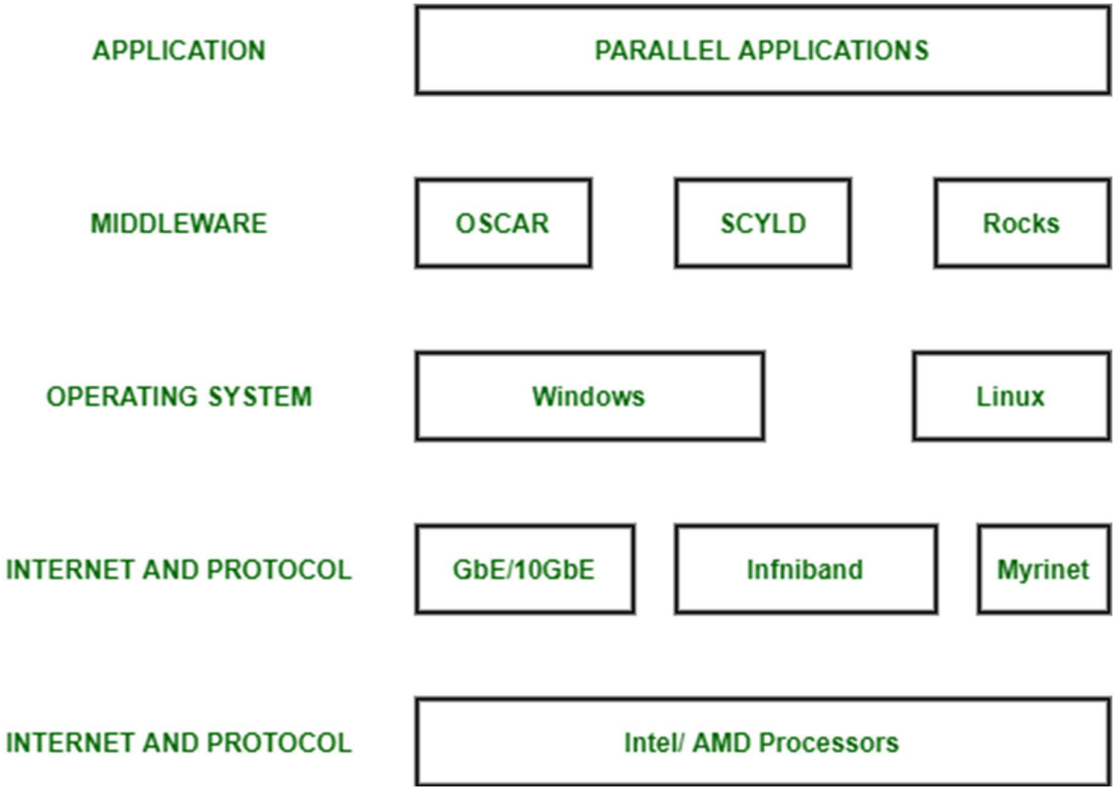


- It is designed with an array of interconnected individual computers and the computer systems operating collectively as a single standalone system.
- It is a group of workstations or computers working together as a single, integrated computing resource connected via high speed interconnects.
- A node – Either a single or a multiprocessor network having memory, input and output functions and an operating system.
- Two or more nodes are connected on a single line or every node might be connected individually through a LAN connection.

Components of a Cluster Computer :

1. Cluster Nodes
2. Cluster Operating System
3. The switch or node interconnect
4. Network switching hardware

Cluster Components:



Advantages of Cluster Computing :

1. High Performance :
The systems offer better and enhanced performance than that of mainframe computer networks.
2. Easy to manage :
Cluster Computing is manageable and easy to implement.
3. Scalable :
Resources can be added to the clusters accordingly.
4. Expandability :
Computer clusters can be expanded easily by adding additional computers to the network. Cluster computing is capable of combining several additional resources or the networks to the existing computer system.

5. Availability :

The other nodes will be active when one node gets failed and will function as a proxy for the failed node. This makes sure for enhanced availability.

6. Flexibility :

It can be upgraded to the superior specification or additional nodes can be added.

Disadvantages of Cluster Computing :

1. High cost :

It is not so much cost-effective due to its high hardware and its design.

2. Problem in finding fault :

It is difficult to find which component has a fault.

3. More space is needed :

Infrastructure may increase as more servers are needed to manage and monitor.

Applications of Cluster Computing :

- Various complex computational problems can be solved.
- It can be used in the applications of aerodynamics, astrophysics and in data mining.
- Weather forecasting.
- Image Rendering.
- Various e-commerce applications.
- Earthquake Simulation.
- Petroleum reservoir simulation.

GRID COMPUTING

Grid Computing – Definition

Grid computing is a type of distributed computing where resources (like processing power, storage, or data) from multiple heterogeneous computers, often spread across different geographical locations, are combined to work on a large task.

Unlike cluster computing (where nodes are tightly coupled and usually homogeneous), grid computing uses loosely coupled, diverse systems that are connected through the internet or wide-area networks.

Key Features of Grid Computing

1. Heterogeneity – Different types of computers (PCs, servers, mainframes) can participate.
2. Geographical Distribution – Resources are spread across different locations.
3. Resource Sharing – Participants share unused CPU cycles, memory, or storage.
4. Loose Coupling – Nodes are not tightly bound; they may join or leave anytime.
5. Scalability – Can scale to thousands or millions of machines worldwide.
6. Collaboration – Often used by organizations/universities to share resources.

Examples of Grid Computing

- SETI@home (Search for Extraterrestrial Intelligence) – Volunteers' computers analyzed radio signals from space.
- Folding@home – Uses volunteer PCs to simulate protein folding for medical research.
- Large Hadron Collider (CERN) – Uses a worldwide grid for processing particle physics data.
- NASA projects – Space research and image processing.

Cluster vs Grid Computing (Quick Table)

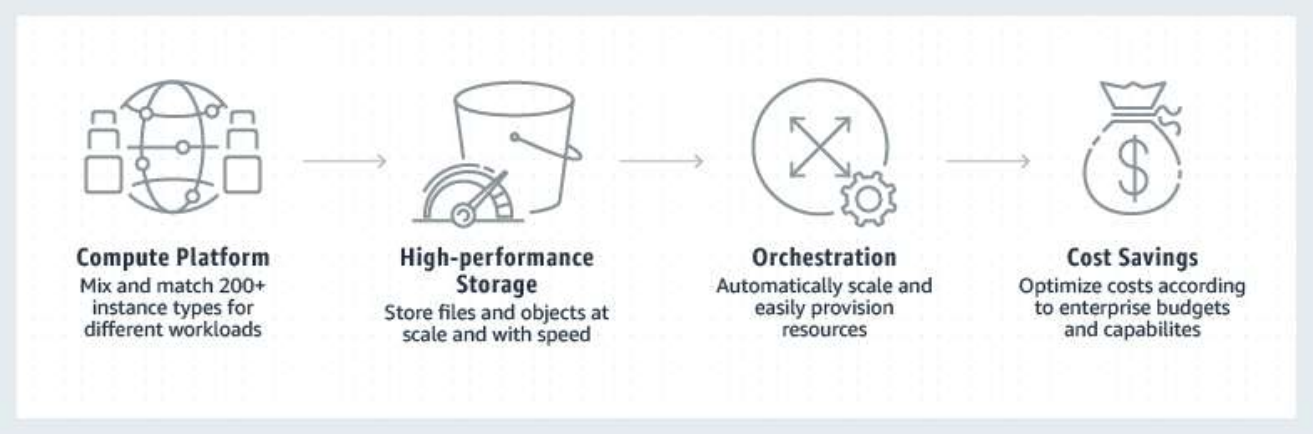
Feature	Cluster Computing 🖥	Grid Computing 🌍
System Type	Homogeneous, similar nodes	Heterogeneous, diverse nodes
Location	Same physical place (LAN)	Spread across the globe (WAN/Internet)
Coupling	Tightly coupled	Loosely coupled
Scalability	Limited (within one site)	Very high (worldwide)
Control	Single organization	Multiple organizations/volunteers
Best Use	High-performance computing	Large-scale collaboration & resource sharing

---

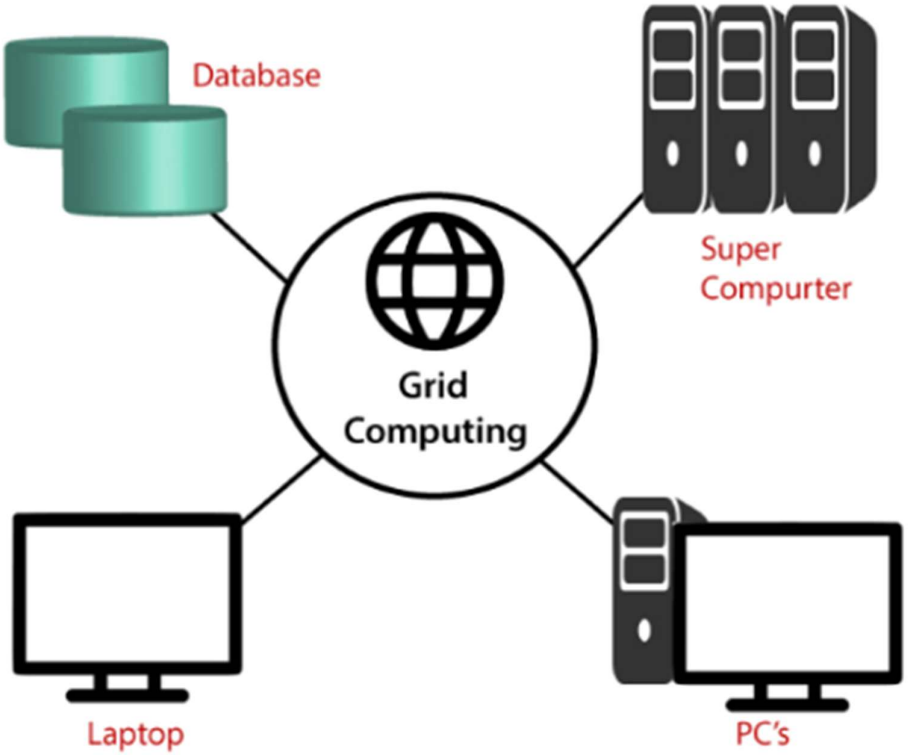
☑ In short:
Grid Computing = A network of geographically distributed heterogeneous computers that work together to solve a big problem by sharing resources.

What is grid computing?

Grid computing is a computing infrastructure that combines computer resources spread over different geographical locations to achieve a common goal. All unused resources on multiple computers are pooled together and made available for a single task. Organizations use grid computing to perform large tasks or solve complex problems that are difficult to do on a single computer.



For example, meteorologists use grid computing for weather modeling. Weather modeling is a computation-intensive problem that requires complex data management and analysis. Processing massive amounts of weather data on a single computer is slow and time consuming. That's why meteorologists run the analysis over geographically dispersed grid computing infrastructure and combine the results.



Why is grid computing important?

Organizations use grid computing for several reasons.

01.Efficiency

With grid computing, you can break down an enormous, complex task into multiple subtasks. Multiple computers can work on the subtasks concurrently, making grid computing an efficient computational solution.

02.Cost

Grid computing works with existing hardware, which means you can reuse existing computers. You can save costs while accessing your excess computational resources. You can also cost-effectively access resources from the cloud.

03. Flexibility

Grid computing is not constrained to a specific building or location. You can set up a grid computing network that spans several regions. This allows researchers in different countries to work collaboratively with the same supercomputing power.

What are the use cases of grid computing?

The following are some common applications of grid computing.

01.Financial services

Financial institutions use grid computing primarily to solve problems involving risk management. By harnessing the combined computing powers in the grid, they can shorten the duration of forecasting portfolio changes in volatile markets.

02.Gaming

The gaming industry uses grid computing to provide additional computational resources for game developers. The grid computing system splits large tasks, such as creating in-game designs, and allocates them to multiple machines. This results in a faster turnaround for the game developers.

03.Entertainment

Some movies have complex special effects that require a powerful computer to create. The special effects designers use grid computing to speed up the production timeline. They have grid-supported software that shares computational resources to render the special-effect graphics.

04. Engineering

Engineers use grid computing to perform simulations, create models, and analyze designs. They run specialized applications concurrently on multiple machines to process massive amounts of data. For example, engineers use grid computing to reduce the duration of a Monte Carlo simulation, a software process that uses past data to make future predictions.

What are the components in grid computing?

In grid computing, a network of computers works together to perform the same task. The following are the components of a grid computing network.

01.Nodes

The computers or servers on a grid computing network are called nodes. Each node offers unused computing resources such as CPU, memory, and storage to the grid network. At the same time, you can also use the nodes to perform other unrelated tasks. There is no limit to the number of nodes in grid computing. There are three main types of nodes: control, provider, and user nodes.

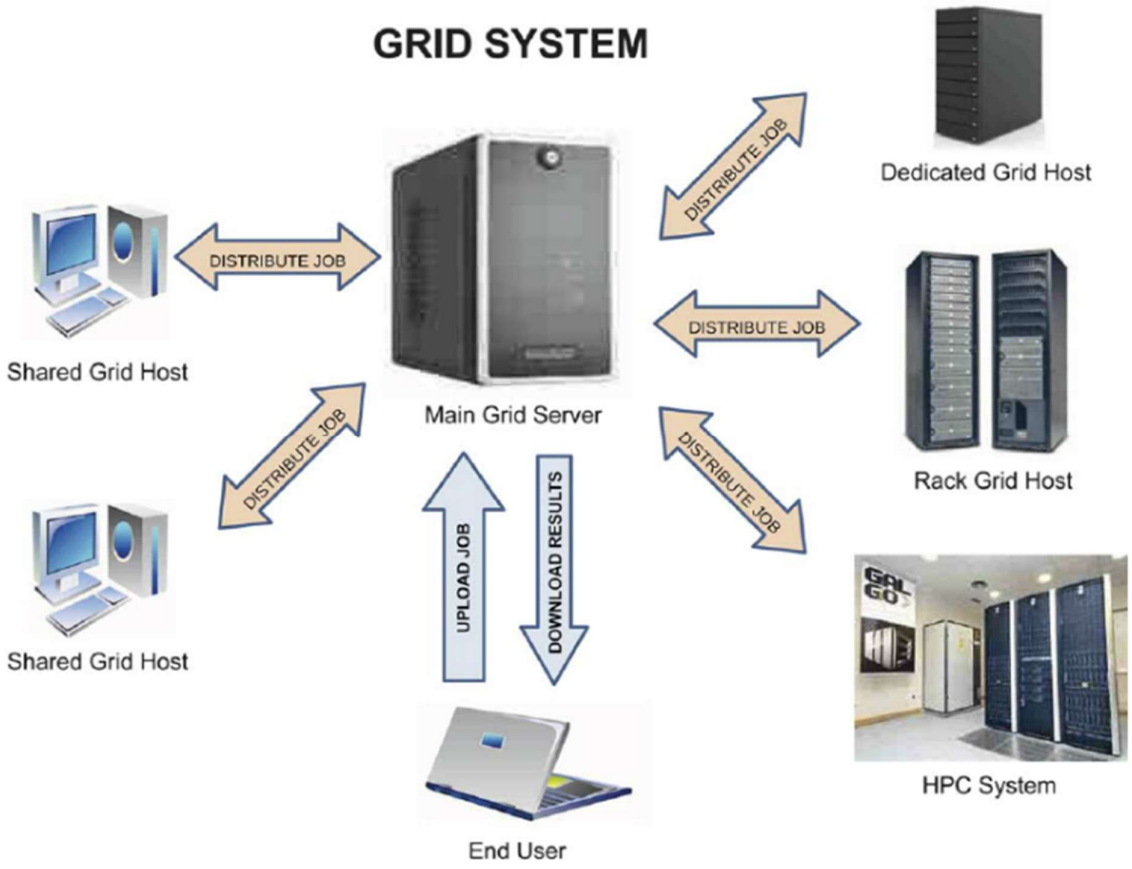
02.Grid middleware

Grid middleware is a specialized software application that connects computing resources in grid operations with high-level applications. For example, it handles your request for additional processing power from the grid computing system.

It controls the user sharing of available resources to prevent overwhelming the grid computers. The grid middleware also provides security to prevent misuse of resources in grid computing.

Grid computing architecture

Grid architecture represents the internal structure of grid computers.
The following layers are broadly present in a grid node:



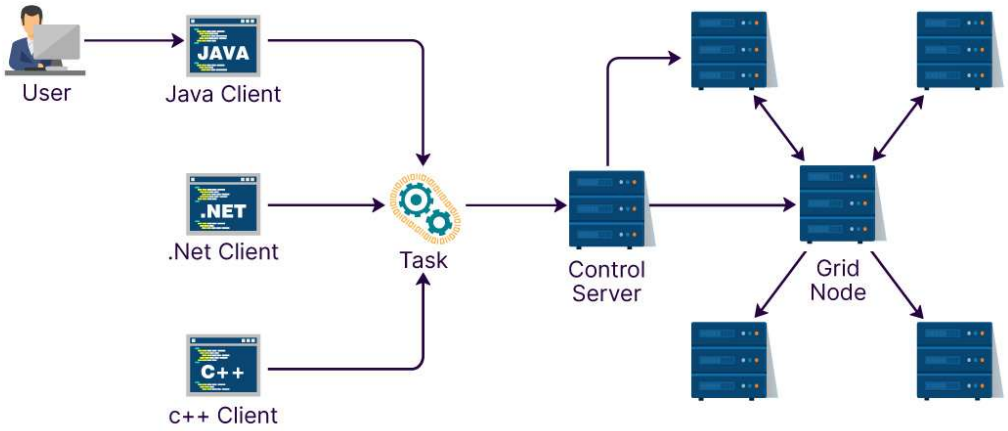
Architecture of Grid Computing

The top layer consists of high-level applications, such as an application to perform predictive modelling.

1. The second layer, also known as middleware, manages and allocates resources requested by applications.
2. The third layer consists of available computer resources such as CPU, memory, and storage.
3. The bottom layer allows the computer to connect to a grid computing network.

How does grid computing work?

Grid nodes and middleware work together to perform the grid computing task. In grid operations, the three main types of grid nodes perform three different roles.



HOW GRID COMPUTING WORKS

01. User node

A user node is a computer that requests resources shared by other computers in grid computing. When the user node requires additional resources, the request goes through the middleware and is delivered to other nodes on the grid computing system.

02. Provider node

In grid computing, nodes can often switch between the role of user and provider.

A provider node is a computer that shares its resources for grid computing. When provider machines receive resource requests, they perform subtasks for the user nodes, such as forecasting stock prices for different markets. At the end of the process, the middleware collects and compiles all the results to obtain a global forecast.

03. Control node

A control node administers the network and manages the allocation of the grid computing resources. The middleware runs on the control node. When the user node requests a resource, the middleware checks for available resources and assigns the task to a specific provider node.

What are the types of grid computing?

Grid computing is generally classified as follows.

01. Computational grid

A computational grid consists of high-performance computers. It allows researchers to use the combined computing power of the computers. Researchers use computational grid computing to perform resource-intensive tasks, such as mathematical simulations.

02.Scavenging grid

While like computational grids, CPU scavenging grids have many regular computers. The term *scavenging* describes the process of searching for available computing resources in a network of regular computers. While other network users access the computers for non-grid–related tasks, the grid software uses these nodes when they are free. The scavenging grid is also known as CPU scavenging or cycle scavenging.

03.Data grid

A data grid is a grid computing network that connects to multiple computers to provide large data storage capacity. You can access the stored data as if on your local machine without having to worry about the physical location of your data on the grid.

Key Components of Grid Computing

A grid computing environment consists of a set of primary grid components. As grid designs and their expected usage vary, specific components may or may not always be a part of the grid network. These components can be combined to form a hybrid component in specific scenarios. Although the combination of elements may differ depending on use cases, understanding their roles can help you while developing grid-enabled applications.

Let’s understand the key components of a grid computing environment.

1. User interface

Today, users are well-versed with web portals. They provide a single interface that allows users to view a wide variety of information. Similarly, a grid portal offers an interface that enables users to launch applications with resources provided by the grid.

The interface has a portal style to help users query and execute various functions on the grid effectively. A grid user views a single, large virtual computer offering computing resources, similar to an internet user who views a unified instance of content on the web.

2. Security

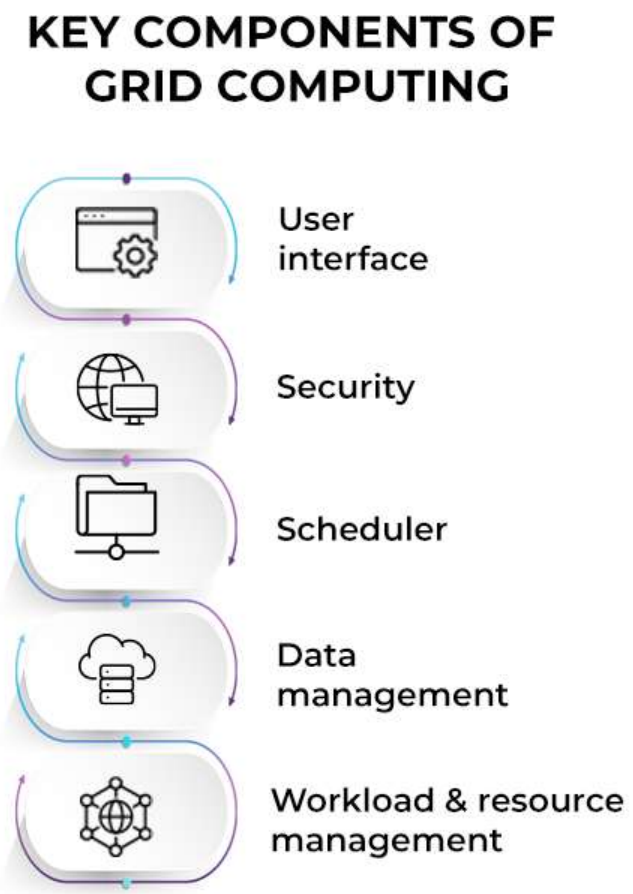
Security is one of the major concerns for grid computing environments. Security mechanisms can include authentication, authorization, data encryption, and others. Grid security infrastructure (GSI) is an important ingredient here. It outlines specifications that establish secret and tamper-proof communication between software entities operating in a grid network.

It includes OpenSSL implementation and provides a single sign-on mechanism for users to perform actions within the grid. It offers robust security by providing authentication and authorization mechanisms for system protection.

3. Scheduler

On identifying the resources, the next step is to schedule the tasks to run on them. A scheduler may not be needed if standalone tasks are to be executed that do not showcase interdependencies. However, if you want to run specific tasks concurrently that require inter-process communication, the job scheduler would suffice to coordinate the execution of different subtasks.

Moreover, schedulers of different levels operate in a grid environment. For example, a cluster may represent an independent resource with its own scheduler to manage the nodes it contains. Hence, a high-level scheduler may sometimes be required to accomplish the task done on the cluster, while the cluster employs its own separate scheduler to handle work on its individual nodes.



4. **Data management**

Data management is crucial for grid environments. A secure and reliable mechanism to move or make any data or application module accessible to various nodes within the grid is necessary. Consider the Globus toolkit — an open-source toolkit for grid computing.

5. Workload & resource management

The workload & resource component enables the actual launch of a job on a particular resource, checks its status, and retrieves the results when the job is complete. Say a user wants to execute an application on the grid. In that case, the application should be aware of the available resources on the grid to take up the workload.

So, it interacts with the workload manager to determine the resource availability and updates the status accordingly. This helps in efficient workload and resource management for various nodes on the grid.

UBIQUITOUS COMPUTING

Ubiquitous Computing (Ubicomp), also called **pervasive computing**, is a concept in computer science where computing is made to appear everywhere and anywhere in everyday life. Instead of having computers as distinct devices (like desktops or laptops), computing power is embedded into everyday objects and environments in a way that people can use them naturally without even noticing.

Key Ideas:

1. **Invisible Integration**
   Computers blend seamlessly into the background of daily activities (e.g., smart homes, wearable devices).
2. **Context Awareness**
   Systems can sense and respond to the environment or user needs (e.g., smartphones detecting location, smart thermostats adjusting temperature).
3. **Mobility and Connectivity**
   Devices are networked and can communicate anytime, anywhere (IoT is a key enabler).
4. **Human-Centric Interaction**
   Interfaces are natural and intuitive—speech, gesture, touch, or even automatic adaptation instead of traditional keyboards/screens.

Examples of Ubiquitous Computing

- **Smart homes**: Alexa, Google Nest, automated lighting.
- **Wearables**: Smartwatches, fitness trackers, AR glasses.
- **Healthcare**: Remote patient monitoring, smart implants.
- **Transportation**: Connected cars, traffic management systems.
- **Retail**: Amazon Go stores (sensor-based checkout).

Benefits

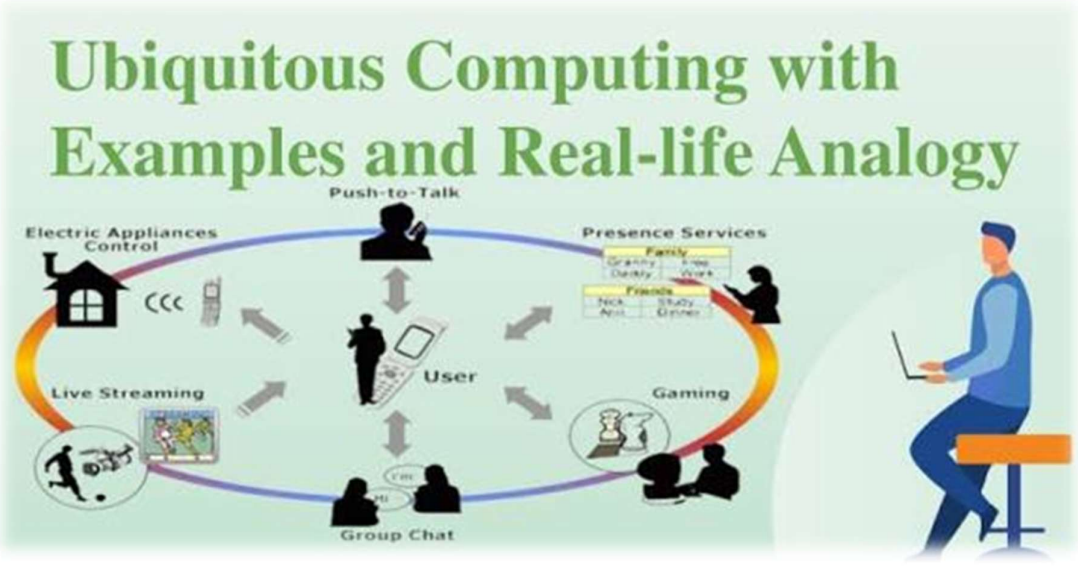
- Convenience and efficiency in daily life.
- Personalized services (health, education, entertainment).
- Improved safety (smart surveillance, health monitoring).
- Automation reduces manual effort.

Challenges

- **Privacy & security** risks (data everywhere).
- **Standardization** across diverse devices.
- **Cost & accessibility** for all.
- **Dependence** on technology.

☞ In short, **ubiquitous computing = computing everywhere, anytime, in everything**.

Ubiquitous computing, or ubicomp, is a paradigm where computing capabilities are seamlessly integrated into everyday objects and environments, making technology pervasive yet invisible to the user. The goal is to create an intelligent and context-aware environment where devices anticipate user needs and respond proactively with minimal explicit human-computer interaction.



Mark Weiser coined the term in 1988 while at Xerox PARC, envisioning it as the "third wave" of computing, following mainframe and personal computing. In this paradigm, computing moves from

one computer shared by many, to one person using one computer, to one person using many computers embedded in their surroundings.

Key characteristics

- **Invisibility and calm technology:** Computing recedes into the background and becomes indistinguishable from the environment, much like successful, commonplace technologies such as the electric motor.
- **Context-awareness:** Ubicomp systems use sensors to detect changes in the environment, including a user's location, and respond accordingly. A smart thermostat, for example, might adjust based on the time of day or whether people are in the room.
- **Interconnectivity:** A huge network of intelligent devices, often part of the Internet of Things (IoT), communicates and shares information to provide seamless, coordinated services.
- **Distributed computation:** Unlike centralized computing, processing power and resources are spread across many different devices. This promotes resilience, efficiency, and flexibility.

Real-world examples

- **Smart homes:** Voice assistants like Alexa, smart thermostats, and automated lighting systems interact with each other to make a home more efficient and responsive to a resident's habits.
- **Intelligent transportation:** Autonomous and self-driving vehicles, smart traffic lights that optimize flow, and electronic toll systems all demonstrate ubicomp in action.
- **Wearable technology:** Smartwatches and fitness trackers are small, unobtrusive computers that continuously monitor health and activity, adapting to the user's needs.
- **Ubiquitous healthcare:** Wearable sensors can enable remote patient monitoring and telemedicine, automatically collecting data and alerting doctors to health issues.
- **Smart cities:** Large-scale networks of sensors and devices manage urban infrastructure, from energy consumption and waste management to public safety and environmental monitoring.

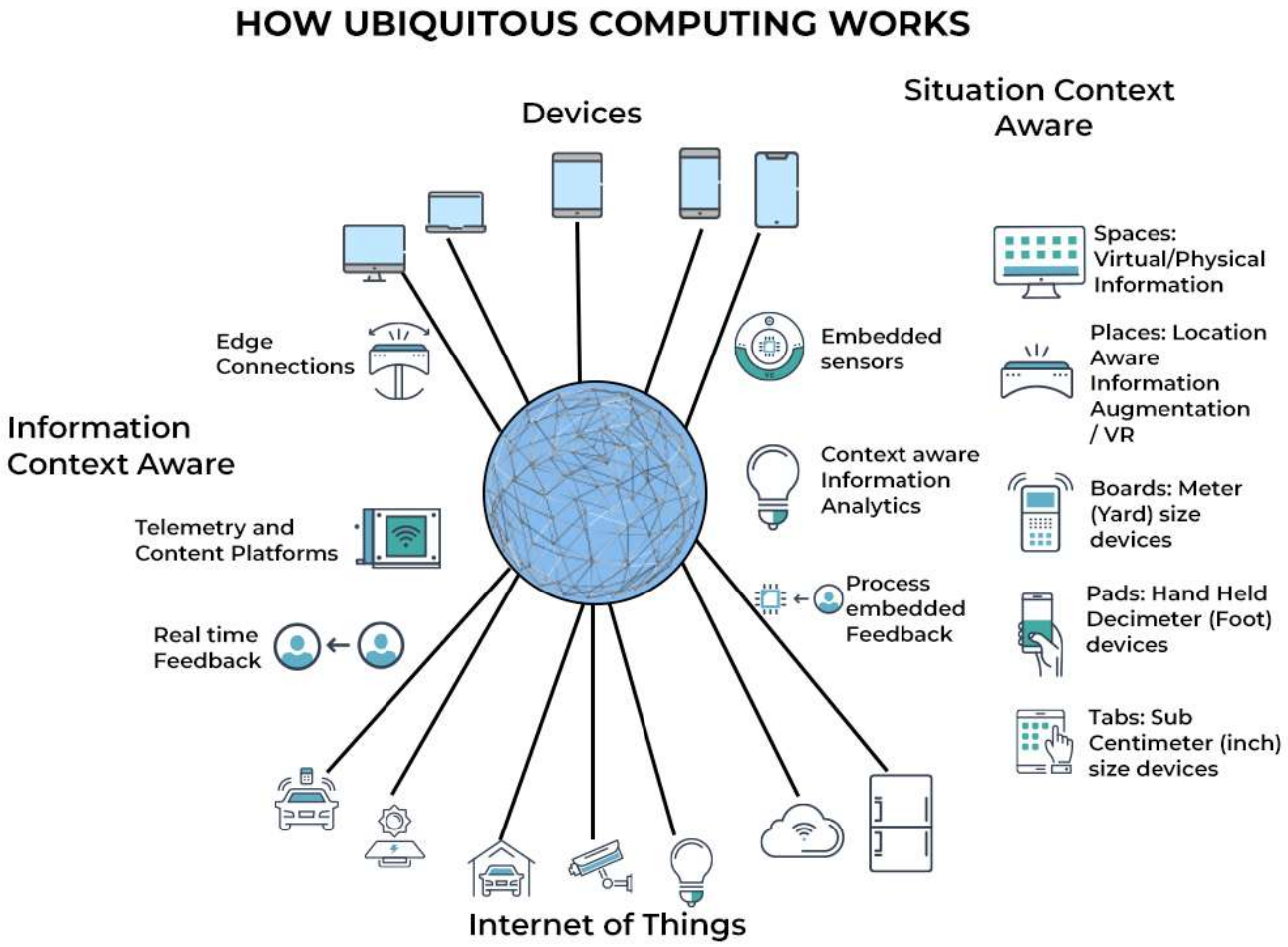
Challenges and the future

- **Privacy and security:** As more devices collect and share data, protecting personal information becomes a major concern. Future solutions will need strong security measures and transparent data practices to build user trust.
- **Interoperability:** Getting countless heterogeneous devices from different vendors to work together smoothly is an ongoing challenge that requires standardization across the industry.
- **Resource limitations:** Many ubicomp devices, such as sensors, are constrained by memory, processor speed, and battery life. Designing efficient systems for these devices is an important area of research.
- **Dependence and reliability:** The increasing reliance on technology embedded in our environment raises questions about what happens when systems fail. Thoughtful design that plans for failure is crucial.

The future of ubiquitous computing will increasingly involve artificial intelligence (AI) and machine learning to make environments even more adaptive and anticipatory. This will allow computing to continue its transition from a focused task to a seamless, integrated part of the human experience.

What Is Ubiquitous Computing?

Pervasive computing, or ubiquitous computing, integrates connectivity functionalities into all of the objects in our environment so they can interact with one another, automate routine tasks, and require minimal human effort to complete tasks and follow machine instructions.



How Ubiquitous Computing Works

Pervasive computing, also known as ubiquitous computing, refers to the growing pattern of embedding computing capacity (generally in the form of microprocessors) into ordinary items to make them communicate effectively and accomplish helpful tasks while reducing the end subscriber's need to communicate with computers. Pervasive computing tools are network-connected and always on.

The late 1980s were the birthplace of ubiquitous computing. The term "ubiquitous computing" was coined in 1988 by Mark Weiser, CTO at Xerox PARC (or the Palo Alto Research Center). He co-

authored some of the first studies on the subject alongside John Seely Brown, chief scientist and director of PARC. Weiser is credited for coining the term “ubiquitous computing” and diving into its underlying difficulties.

The proliferation of interconnected devices in work, home, and transportation environments is called ubiquitous computing or ambient computing. These integrated technologies would make these settings and transportation methods far more engaging and practical via contextual data collection, application, and seamless payment mechanisms.

This is a system in which computer network technology permits itself to exist discreetly in the background of the consumers’ awareness. The seamless integration of computers into our regular activities and physical settings is a core concept in the ubiquitous computing paradigm. Ubiquitous computing tries to make the computer “invisible” by enabling these embedded processors to detect and respond to their application environment autonomously.

Pervasive computing, as opposed to desktop computing, may operate with any device, anytime, in any location, and datatype across networks and can transfer duties from one computer to another when a consumer walks from vehicles to the workplace. Laptops, notebooks, smartphones, tablets, wearable devices, and sensors are ubiquitous computing devices (fleet management and pipeline components, lighting systems, and appliances).

A great example of a ubiquitous computing system is an autonomous vehicle that recognizes its authorized passenger via smartphone vicinity, docks and charges itself when required, and handles the emergency response, toll, and fast-food payments efficiently by interfacing with the infrastructure. It entails linking electrical equipment, including installing microprocessors to transfer data. Devices that employ ubiquitous computing are always available and fully connected.

Ubiquitous computing keeps a close focus on learning by decreasing computer complexity and enhancing efficiency while utilizing computing for everyday activities. Ubiquitous computing, which is sometimes regarded as the successor to mobile computing, usually entails wireless communication and networking technologies, mobile devices, embedded systems, wearable computers, radio frequency ID (RFID) tags, middleware, and software agents.

Internet connectivity, speech recognition, and artificial intelligence (AI) features are frequently added. It incorporates computers into everyday items, allowing people to connect with information-processing equipment more easily and freely than they do now, regardless of place or context.

The expression is further described as a computing environment where each instructor and student has their own internet-connected, private mobile computing device that they may use at home and in the classroom. While computers were formerly large and heavy equipment, ubiquitous computing is based on the downsizing of electronics, which allows for the integration of computer technologies into lightweight handheld devices and various other ubiquitously emerging situations.
Layers of Ubiquitous Computing

It is not as easy as creating a new type of computer with varying abilities from its predecessors to define ubiquitous computing. It's as complex as establishing a unique manner of engaging with and obtaining information, communicating, and a new way of living. One may use wireless sensor networks with the Internet of Things (IoT).

Such sensor networks gather information from selected device sensors before passing it to an IoT server. A system of layers with a specific series of tasks that come together to make up ubiquitous computing might be considered.

These layers include the following:

Layer 1: The task management layer

It examines user tasks, context, and index. It also controls the territory's complicated dependencies. The system uses knowledge about a user's jobs to set up and reconfigure the surroundings on the consumer's behalf. To begin, the infrastructure must understand what to configure for; i.e., what a user requires from surroundings to complete their responsibilities. Second, the infrastructure must understand how to design the environment appropriately: it must have processes to match the user's demands to the resources and capabilities available in the environment.

Task management also analyzes explicit user cues and occurrences in the user's physical environment. It coordinates the two aspects of preserving the user-level state of a suspended task and reinstating the resumed task on receiving a signal from the user – whether to suspend the current task or resume another task. One can also capture complex depictions of user tasks through task management.

It collects information about customer tasks and their related purpose. This information is used to regulate the setup of the environment when the user's work or situation changes. For example, a user seeking to complete a task in a new environment can have task management access to all relevant information and manage task support with the environment management layer.

Layer 2: The environment management layer

It is responsible for mapping service needs, user-level conditions of specific attributes, and resource and capability tracking.The environment's abstract models are kept in the environment management layer. These models bridge the gap between the user's demands, described in environment-independent terms, and the actual abilities of every environment.

Such directions are utilized to handle environmental heterogeneity as well as dynamic change. In terms of heterogeneity, when customers require services, such as voice recognition, environment management will locate and configure a "supplier" for that service from among those accessible in the environment.

In terms of dynamic change, explicit models of the environment's capabilities permit automated reasoning when those abilities change constantly. The environment management automatically updates such a mapping in reaction to variations in the subscriber's wants (adaptation triggered by

task management) and modifications in the environment's resources and capabilities (adoption initiated by environment management). Maximizing a utility function expresses the user's preferences and guides adjustment in both circumstances.

Layer 3: The environment layer

It monitors and maintains essential resources. The environment layer includes programs and devices that one may customize to help a user complete a job. Aside from configuration difficulties, these vendors engage with the user just because they would without the system. The network simply intervenes to set up such vendors on the user's behalf.

The environment manager manipulates the individual abilities of every supplier, acting as an interpreter for the environment-independent definitions of user demands supplied by task management.

The platform allows applications to perform the adaptation strategies suited for every job by factoring models of users' needs and circumstances out of individual applications. This information is challenging to gain at the application level, but once decided at the user level via task management, it is readily transferred to the programs that support the subscriber's task.

Computing systems on which people rely must increasingly adapt to failures in contexts that are not totally within the control of the system implementers. They must modify their run-time features to accommodate the changing loads, resources, and objectives. The field of ubiquitous computing is a particularly relevant sector for self-adaptation.

Consumers are now exposed to diverse, ubiquitous, and changeable technology. It is diverse because computation may occur on various computer systems, interfaces, networks, and services. It penetrates much of our working and residential surroundings via wireless and cable communication. It is flexible because resources change: users might relocate from resource-rich to resource-poor areas.

Examples of Ubiquitous Computing

The qualities and functions that describe the scope of ubiquitous computing operations are enumerated below. Mobility and ad hoc networking are already widespread in real-world communication. Autonomy, context awareness, and energy autonomy are not projected for another two to five years. Context-awareness and embedding in daily things are definitive and formative features of ubiquitous computing. In comparison, energy autonomy and systems and components autonomy are secondary features.

As a result, it logically follows that pervasive computing will emerge gradually as its properties evolve in several steps. There are ubiquitous computing applications everywhere. It has developed into various devices, including computers, notebooks, smartphones, tablets, wearable gadgets, smart speakers, and sensors.

In other words, examples of ubiquitous computing are unlimited, allowing the technology to be one of the most recognized and efficient types. It is a research-based technology that strives for the appropriate computing whenever and wherever. It is used in several ways and is designed to reduce waste and time-lapses.

Electronic toll systems on roads are examples of ubiquitous computing, as are monitoring programs such as Life360, which can measure the user's position, speed, and smartphone battery life; smart traffic lights; and Fitbit. Pervasive computing apps are intended for consumer usage and assist individuals in their daily activities. An example of ubiquitous computing is an Apple Watch that notifies the customer of a call and enables the call to be finished through the watch.

Another example is when a registered user of Audible, Amazon's audiobook server hosted on AWS cloud, starts their book on the train using the Audible app on a smartphone and continues listening to the book at home using Amazon Echo. A ubiquitous computing environment is one in which devices are present everywhere and are capable of some type of computing. The following industries are investing in research and development for ubiquitous computing: energy, entertainment, military, healthcare, and logistics.

Self-driving cars may be voice-controlled, allowing more efficient trips, and saving time and energy. Smart locks safeguard the home and employ cutting-edge technology to keep the owner informed. Pervasive computing also includes smart clocks and lamps that can be controlled by voice and aid with energy efficiency

.

Because ubiquitous computing systems can gather, analyze, and communicate data, they can adapt to the context and activity of the data. That is a network that can comprehend its environment and improve life experiences and life quality

.

The dystopian sci-fi thriller 'Minority Report' showed a bleak future of widespread computer misuse. The proliferation of gadgets, latent biometric scans, and people monitoring using standard identification methods seemed overwhelming when the Internet of Things (IoT) was only starting to gain traction.

Access to wireless Internet, email on mobile phones, and handheld computers give us the impression that continuous, unrestricted data exchange is commonplace. However, the unique performance characteristics of ubiquitous computing will enable an entirely new level of data, information, and knowledge exchange and processing in the future.

With ubiquitous computing, many tasks can be relegated to the background, and the remaining bulk can then be performed wholly or partially autonomously. However, pervasive computing will not arise simultaneously and uniformly across all socioeconomic sectors.

Applications of Ubiquitous Computing

A wide variety of devices are compatible with ubiquitous computing systems. In this section, let's examine the most prominent uses of ubiquitous computing. Pervasive computing envisions a future where small, potentially portable devices, sensors, and actuators connect. Environment monitoring, health and home care, and intelligent transportation systems are just a few of the many prospective uses for pervasive computing. The various applications include the following.

1. Logistics

The difference in IT control systems between physical movement and information flow is bridged by tracking logistical commodities across the entire transportation cycle of raw materials, semi-finished goods, and completed products (including ultimate disposal). This offers the chance to automate and streamline logistics that are already apparent. Computers may use sensor technology to gather, define, and interpret factual information. Users must translate the data into a higher-level idea, the "scenario."

In the actual world, sensory inputs from the physical surroundings are solved using scenario identification approaches, which can be specification-based like, logic programming, fuzzy logic, ontologies, or learning-based, for instance, neural networks, web mining, and decision trees.

Sensor technology is used in manufacturing and logistics to regulate operations and product quality. A critical problem for smart freight items in the transportation of sensitive commodities is assessing quality changes while considering environmental circumstances.

2. Self-driving vehicles

It is also an excellent use of ubiquitous computing expertise. The system is designed to respond to transmitted information with several degrees of autonomy, either reactively, through basic monitoring, or proactively. The system is intended to provide an in-vehicle system that supports the driver with journey planning (drive panel), provides a comfortable atmosphere for the driver while observing him (wellness panel), and adaptively manages interactions with their mobile phone and the online world.

3. Home automation

There is no shortage of smart home gadgets, and the number will only grow as home automation becomes more popular. Smart lamps, smart locks for doors, windows, and cupboards – we're only just beginning to see how technology can alter our daily lives. Home automation systems may control elements like temperature, entertainment, and appliances in addition to lights.

Ubiquitous computing can thus transform many home technological devices, including heating, lighting, ventilation, and communication equipment, into intelligent objects that automatically respond to the needs of the residents. Ubiquitous computing enables the automation of regular physical chores, freeing us from all the manual labor in the household and allowing us to live more freely.

The usage of these technologies tends to alter our traditional views of time and space, breaking free from the constraints provided by the physical confines of the household. Smart home systems may simply bring ease to daily tasks, in addition to the advantages supplied by mechanical and electrical technology.

4. E-commerce

Utilizing a wide variety of digital services, smart objects employed through ubiquitous computing enable the development of new business models. This includes location-based services, the transition from buying to renting goods, and software agents that will teach pervasive computing components to start and run services and business activities on their own.

In today's corporate world, smart services adapt to changing market conditions. Because market conditions change, so should the design of devices that provide smart services. This aspect of adaptability in creating smart gadgets has inherent issues that may impair the device's quality of service. Nonetheless, the benefits outweigh the slight problems associated with implementing and operating smart devices.

5. National security

Identification devices – for example, electronic passports and smart cards – are examples of ubiquitous computing applications in inner security. Monitoring devices are becoming increasingly crucial in the long term, for example, in environmental protection and surveillance of critical facilities, including airports and power grids.

6. Medical technology

Medical applications provide various options, such as smart implants, for monitoring health of the ill and aged in their homes as pervasive computing becomes more autonomous, adaptable, miniaturized, and connected. The fact that ubiquitous computing integrates effectively with numerous application areas may make it a topic of particular interest.

Ubiquitous technology is already allowing the development and management of tailored health and wellness services that intelligently respond and adapt to consumers' ever-changing demands. In doing so, this field handles a wide variety of issues, including but not restricted to disease monitoring, support diagnosis, behaviour coaching, and others.

7. Military and armed forces

The military field needs information on avoiding and combating potential risks that are as detailed, multidimensional, and interconnected as feasible. This includes information gathering and processing. It also covers the creation of new weaponry systems. The 21st-century war style has evolved into a digitalized cooperative union technique that depends primarily on real-time information technology.

The use of ubiquitous computing and network technologies in military defense is required to conduct many battles in the twenty-first century. Moreover, these techniques can increase strategic abilities

such as sensing and detecting, exchanging and sharing sophisticated real-time information, and strengthening the strategic units' capacity.

Takeaway

Ubiquitous computing is all around us, and as IoT devices become more mainstream, it will be a crucial concept for enterprises and personal use. Eventually, the physical world will comprise the building blocks of IT infrastructure to enable smart homes, smart campuses, and entire smart cities. For ubiquitous computing to succeed, the industry requires affordable microprocessor technology, easy access to the cloud, and ruggedized, energy-efficient hardware – which is the next leap in the evolution of computing.

PEER-TO-PEER (P2P) COMPUTING

Definition:

Peer-to-Peer computing is a **distributed network model** where each computer (called a **peer**) acts as both a client and a server. Unlike traditional client–server architecture, there is **no central authority**—peers share resources (files, processing power, bandwidth, etc.) directly with each other.

Key Characteristics

1. **Decentralization** – No single central server; all nodes are equal.
2. **Resource Sharing** – Each peer contributes (storage, files, processing).
3. **Scalability** – More peers = more resources.
4. **Fault Tolerance** – If one peer fails, others continue working.
5. **Dynamic Connections** – Peers can join or leave the network anytime.

Types of P2P Networks

1. **Unstructured P2P**
   - Random connections between peers.
   - Example: Early file-sharing systems like **Napster**, **Gnutella**.
2. **Structured P2P**
   - Uses algorithms (like Distributed Hash Tables) to organize peers efficiently.
   - Example: **BitTorrent**, **Kademlia** network.
3. **Hybrid P2P**
   - Combination of client–server + P2P.
   - Example: Skype (before Microsoft integration).

Examples

- **File sharing**: BitTorrent, LimeWire.
- **Communication**: Skype, WhatsApp (partially P2P).
- **Cryptocurrency**: Bitcoin, Ethereum (blockchain is P2P-based).
- **Collaborative Platforms**: Peer-to-peer lending, decentralized apps (DApps).

Advantages

- ✅ Efficient resource utilization.
- ✅ Scalable with more peers.
- ✅ No central point of failure.
- ✅ Supports large-scale data distribution.

Challenges

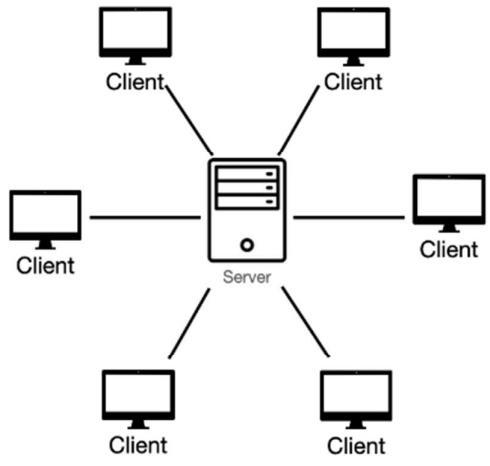
- ⚠️ Security risks (malware spreading easily).
- ⚠️ Data consistency issues.
- ⚠️ Harder to manage than centralized systems.
- ⚠️ Legal/ethical issues (piracy in file sharing).

👉 In short: **P2P computing = users connecting directly, sharing resources without a central server.**

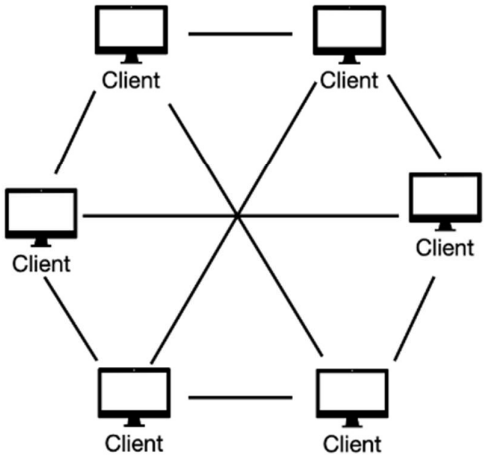
P2P computing

Peer-to-peer (P2P) computing is a decentralized network architecture where interconnected computers, or "peers," share resources and act as both clients and servers. Unlike the traditional client-server model, a P2P network has no central authority or server, making it resilient, scalable, and efficient for specific tasks.

How P2P computing works



Client Server Architecture



P2P Architecture

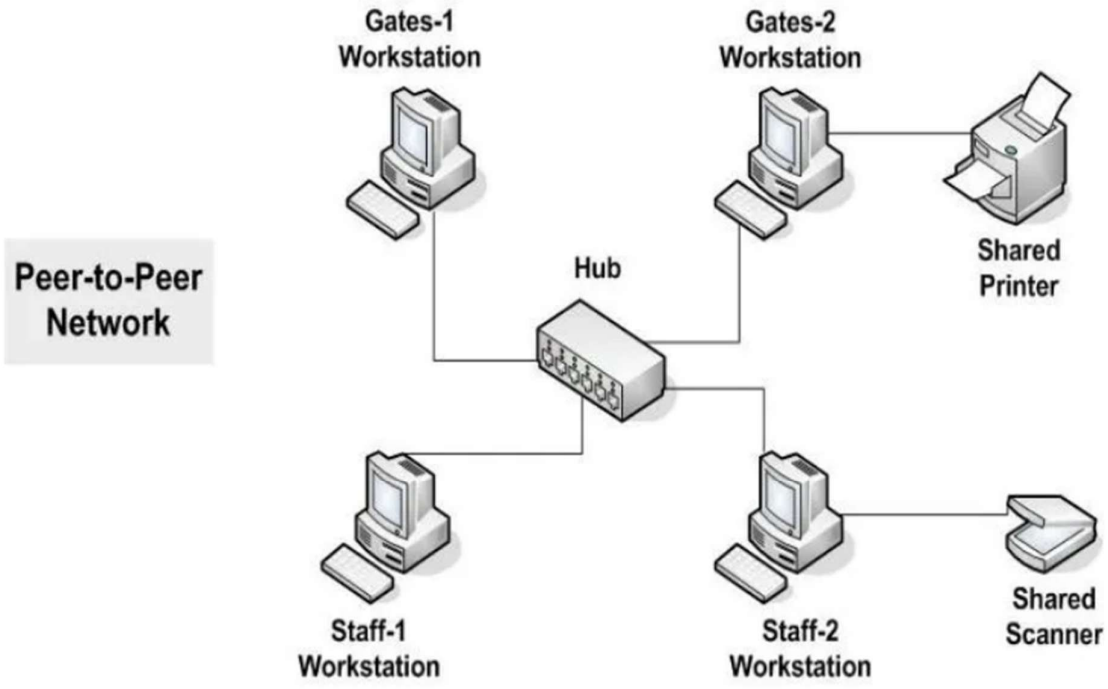
1. **Direct communication:** Each computer running P2P software can connect directly with other devices on the network. A logical "overlay network" is formed on top of the physical internet, allowing peers to find and communicate with each other.

2. **Dual client-server role:** All participants in the network have equal privileges and responsibilities. A computer can request a resource (acting as a client) and also provide a resource to other peers (acting as a server).

3. **Distributed resources:** P2P networks distribute the workload and data storage across all participating devices. If you download a file via a P2P application like BitTorrent, for example, you download small pieces of that file from multiple other users in the network simultaneously

4. **Resource discovery:** To locate a resource like a file, a peer may use different methods depending on the network's structure.
    - In an **unstructured network**, the query is flooded across the network to find a peer with the requested file.
    - In a **structured network**, a distributed hash table (DHT) is used to efficiently map keys (like file identifiers) to specific peers responsible for storing them.

Types of P2P network architectures



Peer — to — Peer Model

P2P networks are typically categorized by their level of centralization.

- **Centralized:** Uses a central server to store metadata, such as user information and file locations, but the files are transferred directly between peers. The original Napster was a centralized P2P network, and its central server made it vulnerable to shut down.

- **Decentralized (pure P2P):** Operates with no central authority at all. Communication and resource discovery happen directly between peers. The Gnutella network is an example.

- **Hybrid:** Combines elements of both client-server and pure P2P models. A central server may help peers find each other, but the data is transferred directly between them. The BitTorrent protocol operates as a hybrid model.

Key advantages

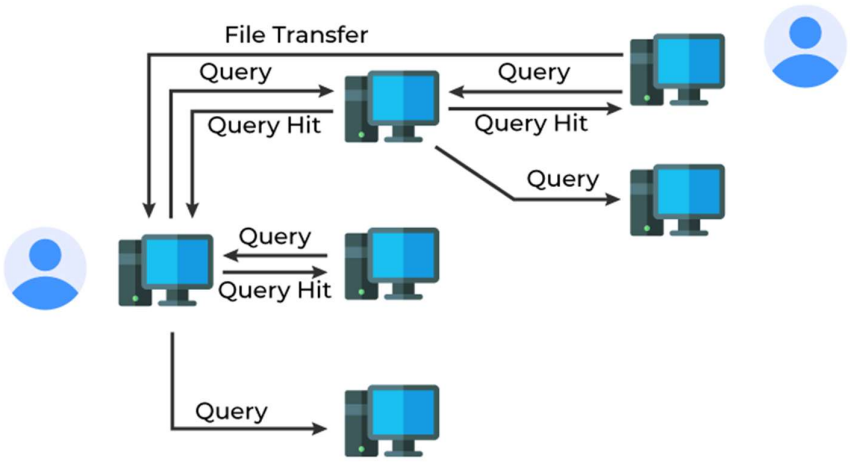
- **Decentralization:** With no central server, there is no single point of failure. The network remains resilient even if many nodes go offline.
- **Scalability:** The total bandwidth and storage capacity increase as more users join the network. This avoids bottlenecks that can occur in centralized systems.
- **Cost-effectiveness:** P2P networks reduce the need for expensive, centralized server infrastructure. Resources are pooled from all participants, lowering the cost of content delivery.
- **Efficiency:** Distributing data across multiple sources can lead to faster downloads, particularly for popular files.

Common applications

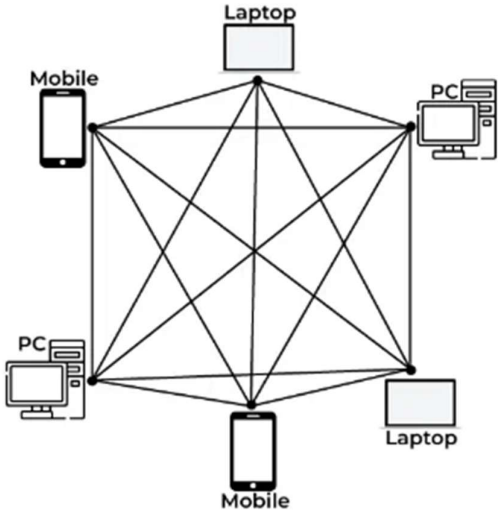
P2P computing has evolved beyond its origins in file-sharing to enable a wide range of modern applications:

- **File sharing:** This is the classic P2P use case, with applications like BitTorrent allowing users to efficiently share large files like software and videos.

Gnutella: Gnutella represents a new wave of P2P applications providing distributed discovery and sharing of resources across the Internet. Gnutella is distinguished by its support for anonymity and its decentralized architecture. A Gnutella network consists of a dynamically changing set of peers connected using TCP/IP.



- **Blockchain and cryptocurrencies:** The Bitcoin network operates on a decentralized P2P architecture, where each node maintains a copy of the ledger and validates transactions without a central authority.
- **Distributed computing:** In projects like SETI@home, peers donate unused processing power to collectively solve complex computational problems.
- **Real-time communication:** VoIP and messaging apps, including the early versions of Skype, have used P2P technology to establish direct connections between users.
- **Content delivery:** In some content delivery networks (CDNs), P2P augments traditional servers to handle peak demand more effectively



DISTRIBUTED SYSTEM MODELS IN CLOUD COMPUTING

A **distributed system** is a network of independent computers that work together as a single system. In **cloud computing**, distributed system models are the backbone—they allow **scalability, fault tolerance, resource sharing, and high availability**.

1. Client–Server Model

- **How it works**: Clients request services, and servers provide them.
- **Cloud Example**:
  - Google Drive (client = user device, server = Google's cloud servers).
- **Use Case**: Web services, SaaS (Software as a Service).

2. Peer-to-Peer (P2P) Model

- **How it works**: Each node acts as both client and server, sharing resources directly.
- **Cloud Example**:
  - Blockchain (Ethereum, Bitcoin), P2P file storage (IPFS).
- **Use Case**: Decentralized apps, distributed file systems.

3. Three-Tier / Multi-Tier Model

- **How it works**: Divides system into **Presentation Layer (UI)**, **Application Layer (logic)**, and **Data Layer (database)**.
- **Cloud Example**:
  - Web app hosted on AWS: frontend (React) + backend (Node.js) + database (RDS).
- **Use Case**: Web apps, enterprise cloud apps.

4. Service-Oriented Architecture (SOA)

- **How it works**: Applications are broken into **services** that communicate via standard protocols (e.g., SOAP, REST).
- **Cloud Example**:
  - Amazon Web Services (different services like EC2, S3, Lambda).
- **Use Case**: Large enterprise systems, legacy integrations.

5. Microservices Model

- **How it works**: Application split into **independent microservices**; each runs in its own container (like Docker) and communicates via APIs.
- **Cloud Example**:
  - Netflix (runs hundreds of microservices in AWS).
- **Use Case**: Scalable apps, DevOps, CI/CD pipelines.

6. Distributed File System Model

- **How it works**: Data is stored across multiple servers but appears as a **single storage system**.
- **Cloud Example**:
  - Hadoop Distributed File System (HDFS), Google File System (GFS).
- **Use Case**: Big data, analytics, cloud storage.

7. Virtualization Model

- **How it works**: Multiple virtual machines run on the same physical server. Resources are abstracted and allocated dynamically.
- **Cloud Example**:
  - VMware, Hyper-V, AWS EC2 instances.
- **Use Case**: IaaS (Infrastructure as a Service).

8. MapReduce Model

- **How it works**: Breaks big tasks into smaller chunks:
  - **Map phase** → distribute tasks
  - **Reduce phase** → combine results
- **Cloud Example**:
  - Hadoop MapReduce, Google Cloud Dataflow.
- **Use Case**: Big data processing, analytics, ML pipelines.

☑ In short:
Cloud computing relies on **distributed system models** like **Client–Server, P2P, Multi-Tier, SOA, Microservices, Distributed File Systems, Virtualization, and MapReduce**—each fits different application needs.

DISTRIBUTED COMPUTING SYSTEM MODELS

Distributed computing is a system where processing and data storage is distributed across multiple devices or systems, rather than handled by a single central device. In this article, we will see Distributed Computing System Models.

Types of Distributed Computing System Models

1. PHYSICAL MODEL

A physical model represents the underlying hardware elements of a distributed system. It encompasses the hardware composition of a distributed system in terms of computers and other devices and their interconnections. It is primarily used to design, manage, implement, and determine the performance of a distributed system.

A physical model majorly consists of the following components:

1. Nodes

Nodes are the end devices that can process data, execute tasks, and communicate with the other nodes. These end devices are generally the computers at the user end or can be servers, workstations, etc.

- Nodes provision the distributed system with an interface in the presentation layer that enables the user to interact with other back-end devices, or nodes, that can be used for storage and database services, processing, web browsing, etc.
- Each node has an Operating System, execution environment, and different middleware requirements that facilitate communication and other vital tasks.,

2. Links

Links are the communication channels between different nodes and intermediate devices. These may be wired or wireless. Wired links or physical media are implemented using copper wires, fiber optic cables, etc. The choice of the medium depends on the environmental conditions and the requirements. Generally, physical links are required for high-performance and real-time computing. Different connection types that can be implemented are as follows:

- **Point-to-point links:** Establish a connection and allow data transfer between only two nodes.
- **Broadcast links:** It enables a single node to transmit data to multiple nodes simultaneously.
- **Multi-Access links:** Multiple nodes share the same communication channel to transfer data. Requires protocols to avoid interference while transmission.

3. Middleware

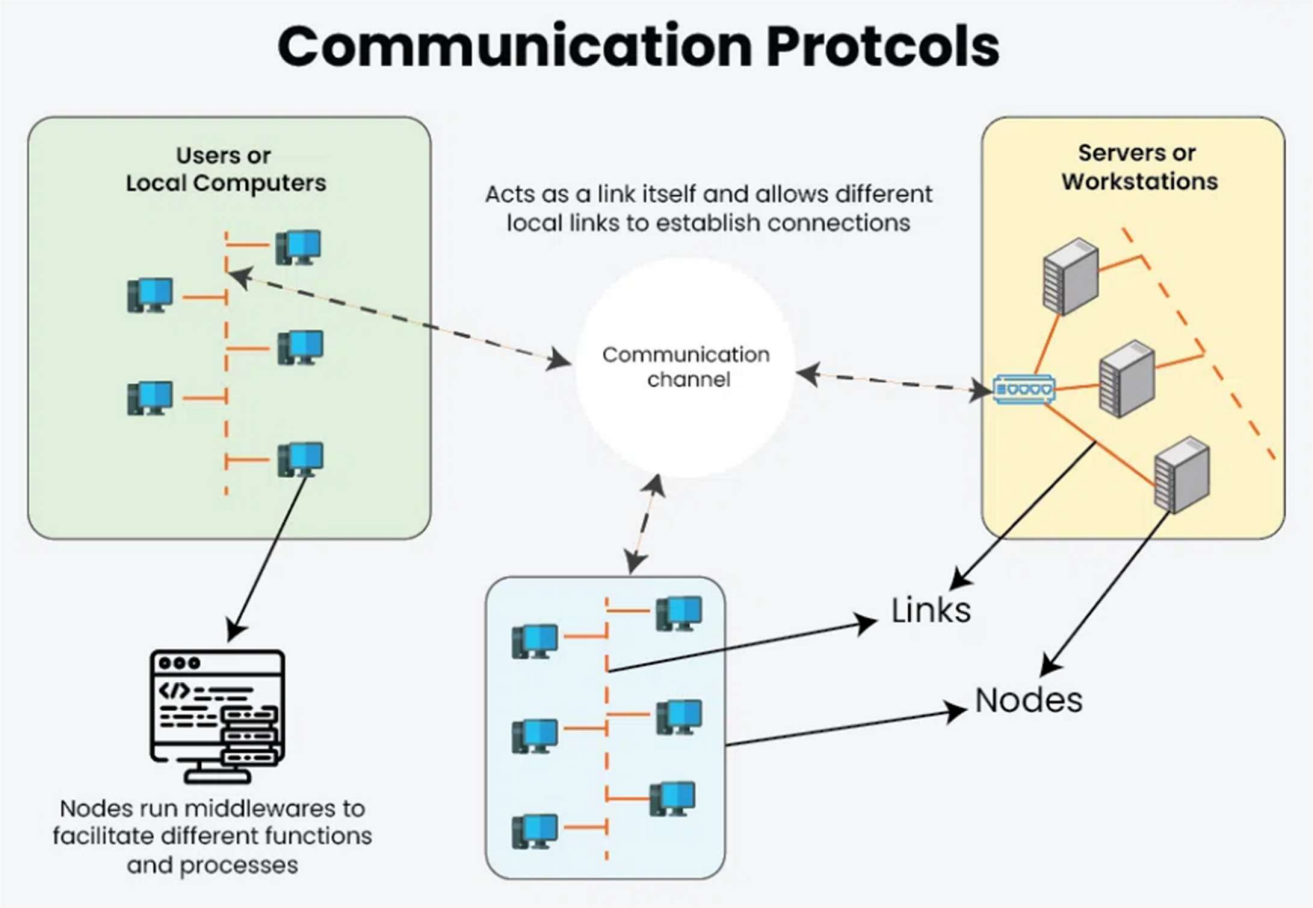
These are the softwares installed and executed on the nodes. By running middleware on each node, the distributed computing system achieves a decentralised control and decision-making. It handles various tasks like communication with other nodes, resource management, fault tolerance, synchronisation of different nodes and security to prevent malicious and unauthorised access.

4. Network Topology

This defines the arrangement of nodes and links in the distributed computing system. The most common network topologies that are implemented are bus, star, mesh, ring or hybrid. Choice of topology is done by determining the exact use cases and the requirements.

5. Communication Protocols

Communication protocols are the set rules and procedures for transmitting data from in the links. Examples of these protocols include TCP, UDP, HTTPS, MQTT etc. These allow the nodes to communicate and interpret the data.



2. ARCHITECTURAL MODEL

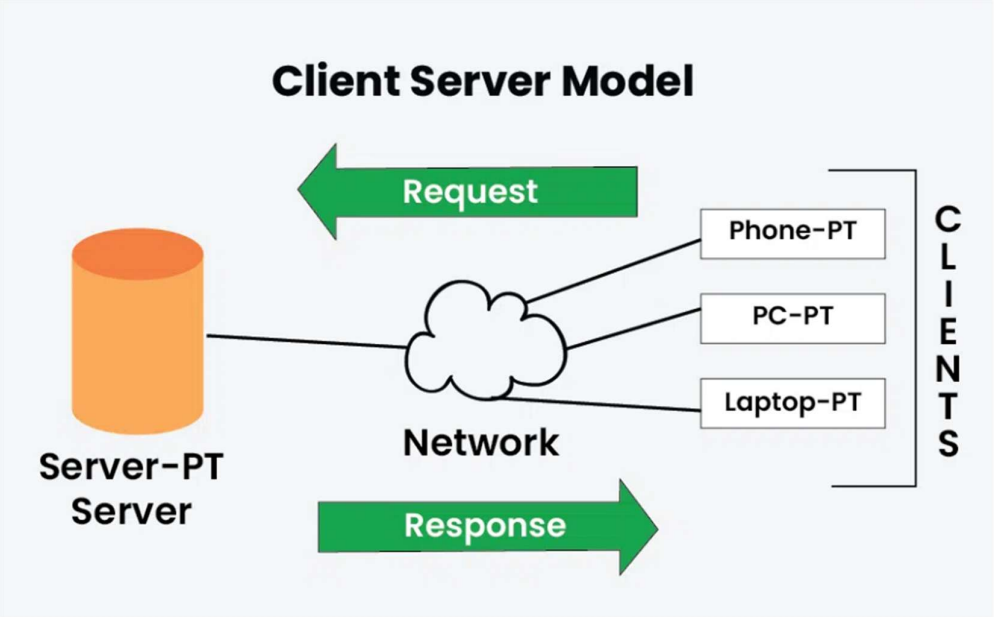
Architectural model in distributed computing system is the overall design and structure of the system, and how its different components are organised to interact with each other and provide the desired functionalities. It is an overview of the system, on how will the development, deployment and operations take place. Construction of a good architectural model is required for efficient cost usage, and highly improved scalability of the applications.

The key aspects of architectural model are:

1. Client-Server model

It is a centralised approach in which the clients' initiate requests for services and severs respond by providing those services. It mainly works on the request-response model where the client sends a request to the server and the server processes it and responds to the client accordingly.

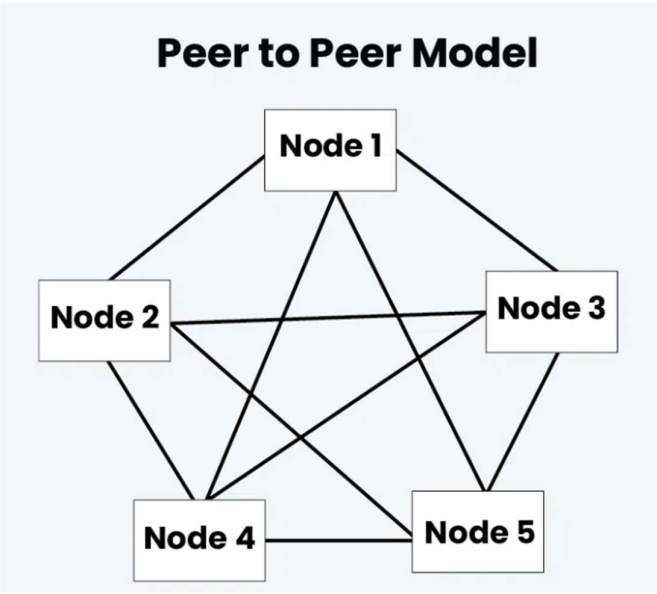
- It can be achieved by using TCP/IP, HTTP protocols on the transport layer.
- This is mainly used in web services, cloud computing, database management systems etc.



2. Peer-to-peer model

It is a decentralised approach in which all the distributed computing nodes, known as peers, are all the same in terms of computing capabilities and can both request as well as provide services to other peers. It is a highly scalable model because the peers can join and leave the system dynamically, which makes it an ad-hoc form of network.

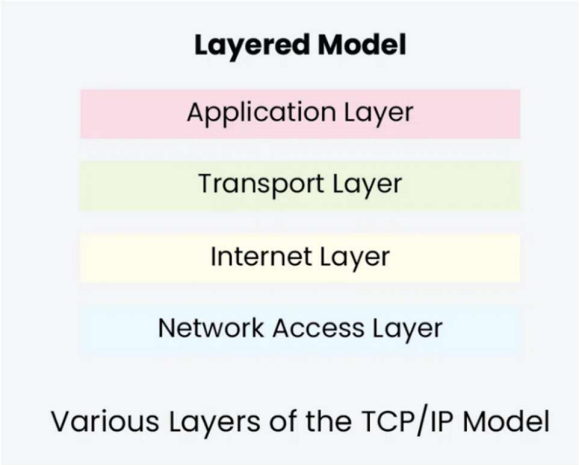
- The resources are distributed, and the peers need to look out for the required resources as and when required.
- The communication is directly done amongst the peers without any intermediaries according to some set rules and procedures defined in the P2P networks.



- The best example of this type of computing is BitTorrent.

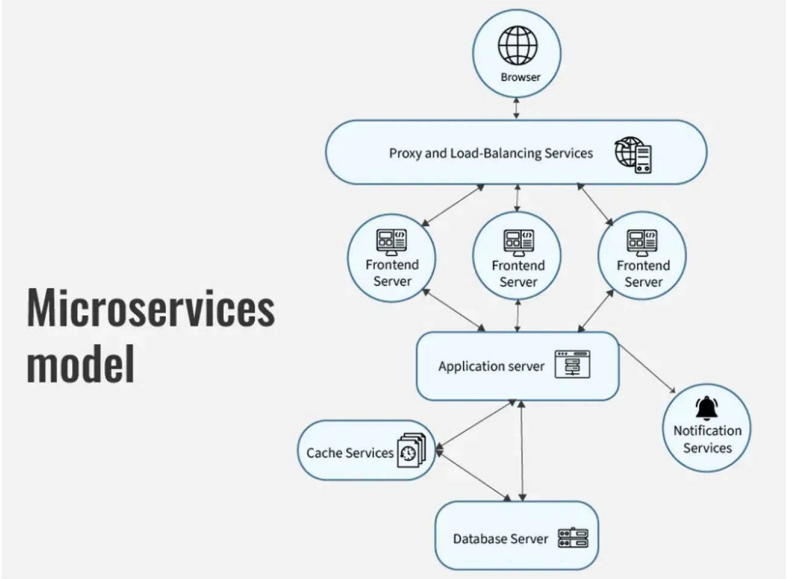
3. Layered model

It involves organising the system into multiple layers, where each layer will provision a specific service. Each layer communicated with the adjacent layers using certain well-defined protocols without affecting the integrity of the system. A hierarchical structure is obtained where each layer abstracts the underlying complexity of lower layers.



4. Micro-services model

In this system, a complex application or task, is decomposed into multiple independent tasks and these services running on different servers. Each service performs only a single function and is focussed on a specific business-capability. This makes the overall system more maintainable, scalable and easier to understand. Services can be independently developed, deployed and scaled without affecting the ongoing services.



3. Fundamental Model

The fundamental model in a distributed computing system is a broad conceptual framework that helps in understanding the key aspects of the distributed systems. These are concerned with more formal description of properties that are generally common in all architectural models. It represents the essential components that are required to understand a distributed system's behaviour.

Three fundamental models are as follows:

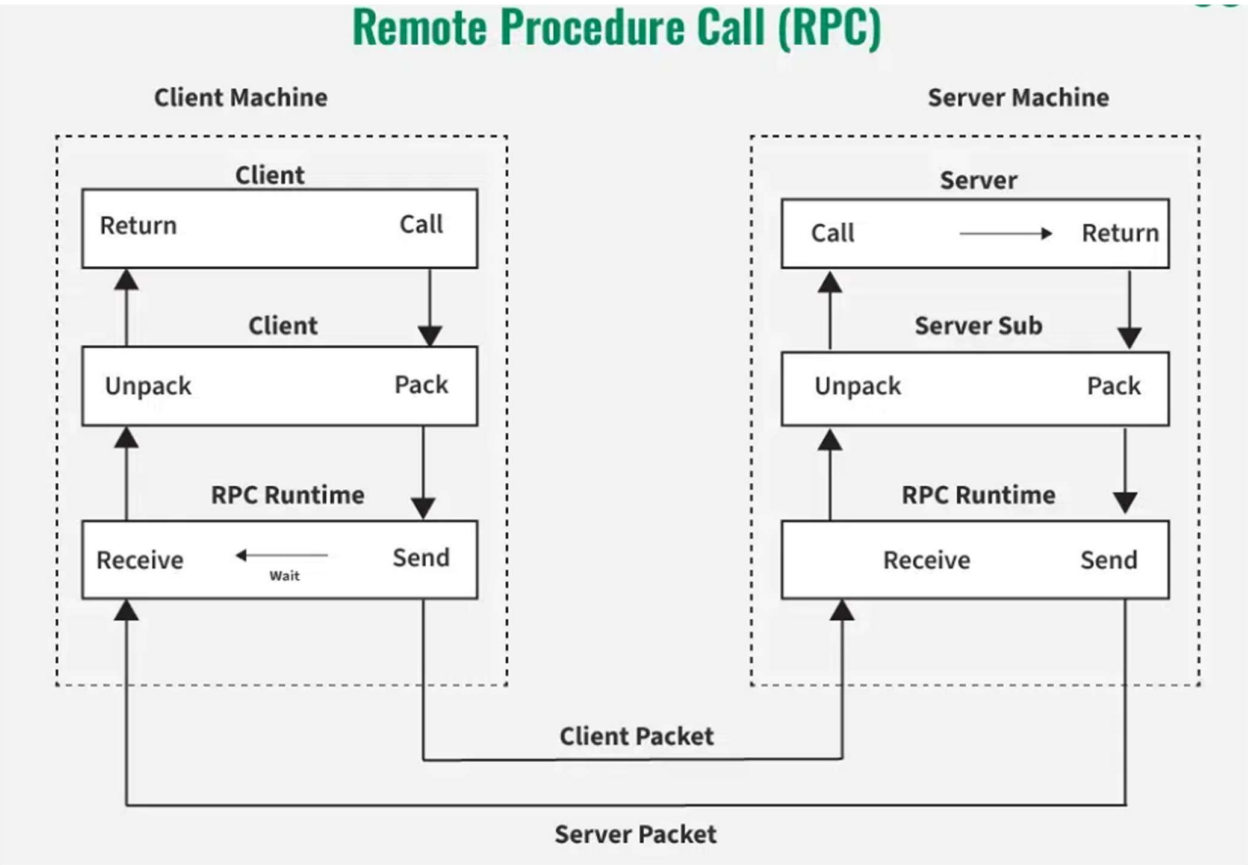
1. Interaction Model

Distributed computing systems are full of many processes interacting with each other in highly complex ways. Interaction model provides a framework to understand the mechanisms and patterns that are used for communication and coordination among various processes. Different components that are important in this model are -

- **Message Passing -** It deals with passing messages that may contain, data, instructions, a service request, or process synchronisation between different computing nodes. It may be synchronous or asynchronous depending on the types of tasks and processes.
- **Publish/Subscribe Systems -** Also known as pub/sub system. In this the publishing process can publish a message over a topic and the processes that are subscribed to that topic can take it up and execute the process for themselves. It is more important in an event-driven architecture.

2. Remote Procedure Call (RPC)

It is a communication paradigm that has an ability to invoke a new process or a method on a remote process as if it were a local procedure call. The client process makes a procedure call using RPC and then the message is passed to the required server process using communication



protocols. These message passing protocols are abstracted and the result once obtained from the server process, is sent back to the client process to continue execution.

1. Failure Model

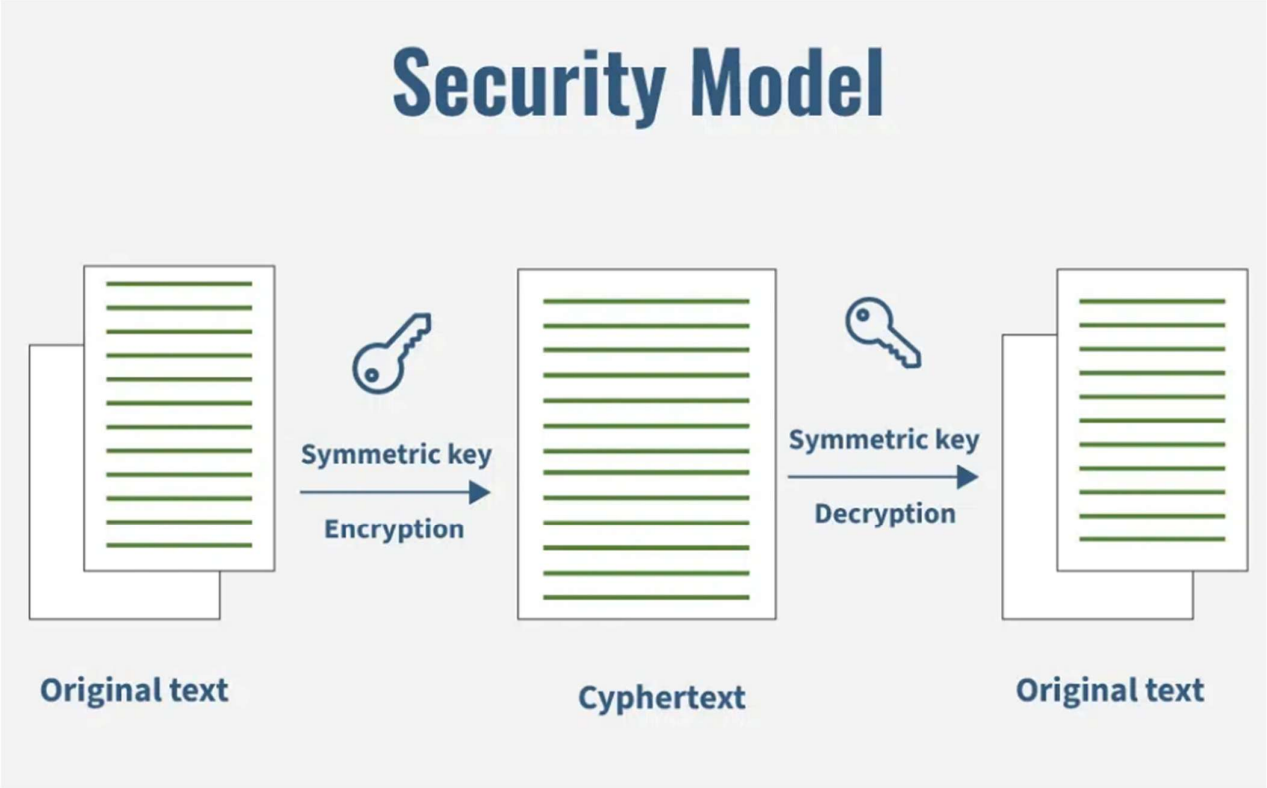
This model addresses the faults and failures that occur in the distributed computing system. It provides a framework to identify and rectify the faults that occur or may occur in the system. Fault tolerance mechanisms are implemented to handle failures by replication and error detection and recovery methods.

Different failures that may occur are:

- **Crash failures -** A process or node unexpectedly stops functioning.
- **Omission failures -** It involves a loss of message, resulting in absence of required communication.
- **Timing failures -** The process deviates from its expected time quantum and may lead to delays or unsynchronised response times.
- **Byzantine failures -** The process may send malicious or unexpected messages that conflict with the set protocols.

2. Security Model

Distributed computing systems may suffer malicious attacks, unauthorised access and data breaches. Security model provides a framework for understanding the security requirements, threats, vulnerabilities, and mechanisms to safeguard the system and its resources. Various aspects that are vital in the security model are:



- **Authentication:** It verifies the identity of the users accessing the system. It ensures that only the authorised and trusted entities get access.

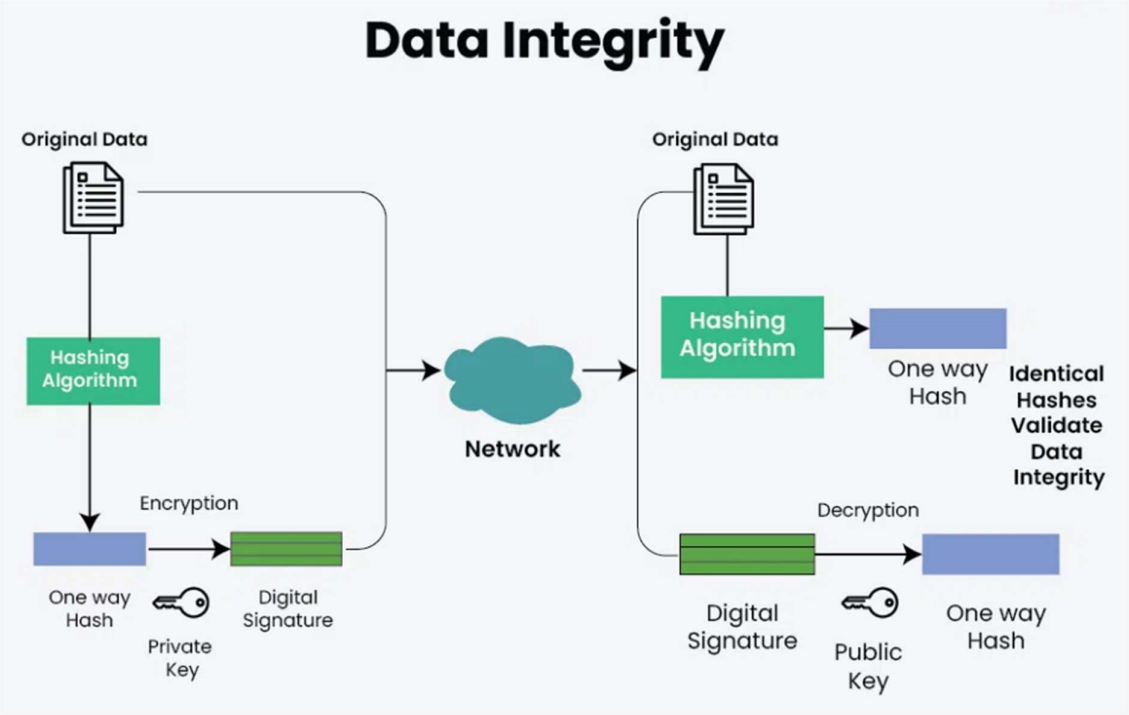
It involves –

✓ **Password-based authentication:** Users provide a unique password to prove their identity.

- ✓ **Public-key cryptography:** Entities possess a private key and a corresponding public key, allowing verification of their authenticity.
- ✓ **Multi-factor authentication:** Multiple factors, such as passwords, biometrics, or security tokens, are used to validate identity.

- **Encryption:**
  - It is the process of transforming data into a format that is unreadable without a decryption key. It protects sensitive information from unauthorized access or disclosure.

- **Data Integrity:**



  - Data integrity mechanisms protect against unauthorised modifications or tampering of data. They ensure that data remains unchanged during storage, transmission, or processing. Data integrity mechanisms include:
    - **Hash functions -** Generating a hash value or checksum from data to verify its integrity.
    - **Digital signatures -** Using cryptographic techniques to sign data and verify its authenticity and integrity.

ENABLING TECHNOLOGIES:

"**Enabling technologies**" are technologies that make it possible to create new products, processes, or services, or to significantly enhance existing ones. They act as **foundational tools or systems** that allow other innovations and applications to be developed.
Here's a structured view:

◇ Definition

Enabling technologies are innovations that provide the means to drive **new capabilities**, **efficiency improvements**, and **transformations** in industries, economies, and societies.

◇ Key Characteristics

- **Foundational**: They serve as building blocks for other technologies.
- **Catalytic**: Enable or accelerate advancements in multiple fields.
- **Transformative**: Often change the way industries operate.
- **Cross-cutting**: Apply across different domains (healthcare, IT, manufacturing, etc.).

◇ Examples

1. **Digital Technologies**
   - Artificial Intelligence (AI) & Machine Learning
   - Cloud Computing
   - Internet of Things (IoT)
   - Blockchain
2. **Physical Technologies**
   - 3D Printing (Additive Manufacturing)
   - Robotics & Automation
   - Advanced Materials (e.g., nanomaterials, composites)
3. **Biological & Life Sciences**
   - Biotechnology & Genetic Engineering
   - CRISPR gene editing
   - Bioinformatics
4. **Energy & Environment**
   - Renewable Energy (solar, wind, hydrogen)
   - Smart Grids
   - Energy Storage (advanced batteries)
5. **Connectivity & Communication**
   - 5G/6G Networks
   - Satellite Internet
   - Quantum Communication

◇ Impact of Enabling Technologies

- Drive **economic growth** and competitiveness
- Enable **new business models** (e.g., sharing economy, Industry 4.0)

- Enhance **quality of life** (healthcare, sustainability, accessibility)
- Lead to **disruption** of traditional industries

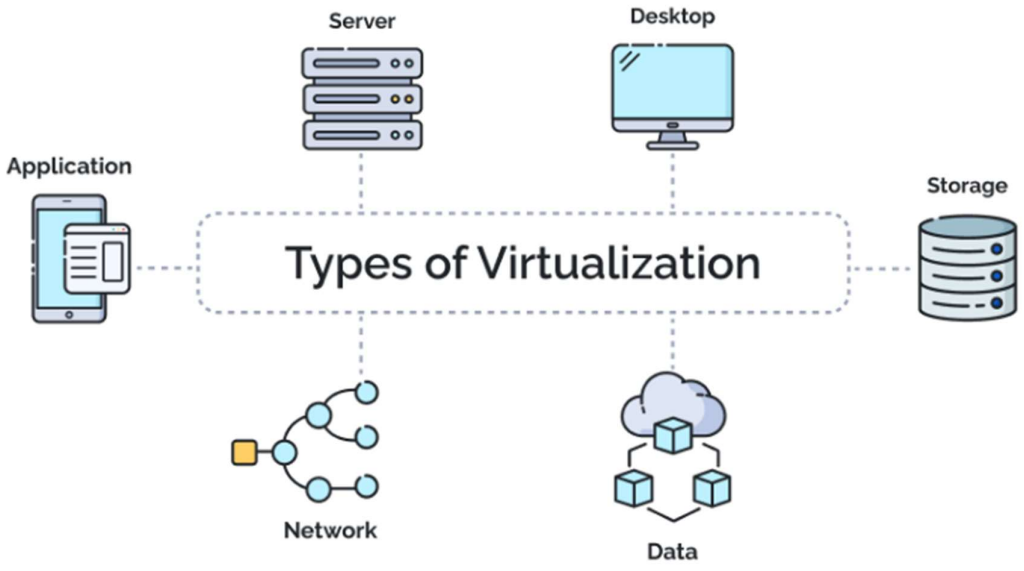
Enabling technologies are the foundational components and methodologies that make cloud computing possible. They allow for the on-demand, scalable, and shared delivery of computing resources over a network.

Key enabling technologies include:

Virtualization

Virtualization is a core cloud technology that abstracts physical hardware resources into multiple virtual instances.

- **How it works:** A hypervisor (a software layer) sits between the hardware and the operating system. It allows a single physical machine to run multiple virtual machines (VMs), each with its own operating system and applications.

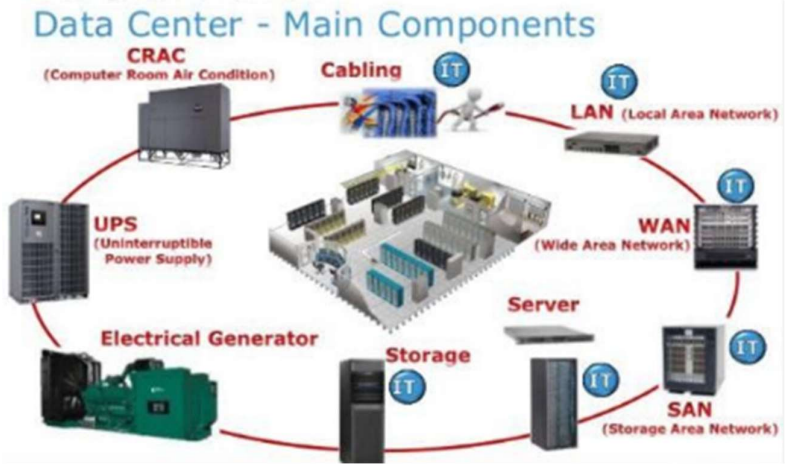


- **Cloud impact:** This enables resource pooling, where a provider can serve multiple customers from the same physical server. It also increases efficiency, optimizes hardware use, and supports rapid provisioning of resources.

Data center technology

Cloud providers rely on vast, powerful data centers to house the physical infrastructure for cloud services.

- **How it works:** Data centers group large numbers of commodity servers, storage arrays, and networking equipment together. The design focuses on efficiency, modularity, automation, and remote management.

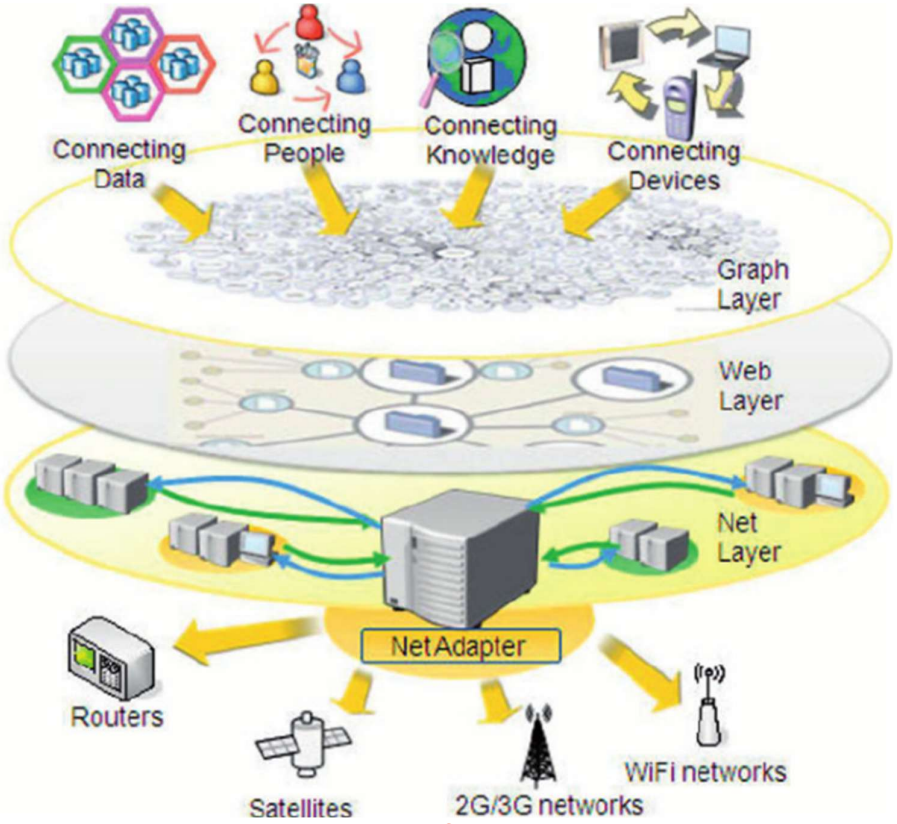


- **Cloud impact:** These centers provide the physical foundation for cloud computing. Standardization and remote management reduce costs, while modularity allows for easy scaling.

Internet architecture and web technologies

Cloud computing fundamentally depends on a robust and reliable internet to deliver services.

- **How it works:** A global network of internet service providers (ISPs) uses technologies like packet switching and router-based interconnectivity to ensure data can be sent reliably over vast distances. Web technologies, such as Uniform Resource Locators (URLs), HTTP, HTML, and XML, provide the basic communication standards for accessing web-based applications.

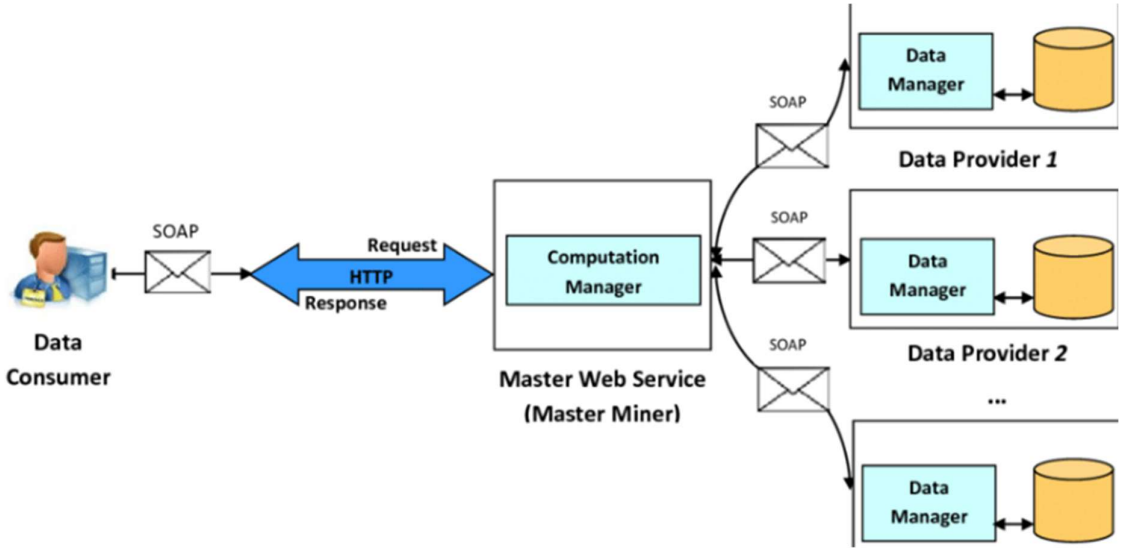


- **Cloud impact:** These technologies provide the broad network access needed for clients to connect to and use cloud services from anywhere.

Service-oriented architecture (SOA) and web services

SOA is a design approach that uses web services to create applications that are independent of any single programming language or platform.

- **How it works:** Applications are built from loosely coupled, interoperable "services" that can be reused and orchestrated to form new applications. Web services technologies like WSDL, XML Schema, and SOAP are foundational to this model.

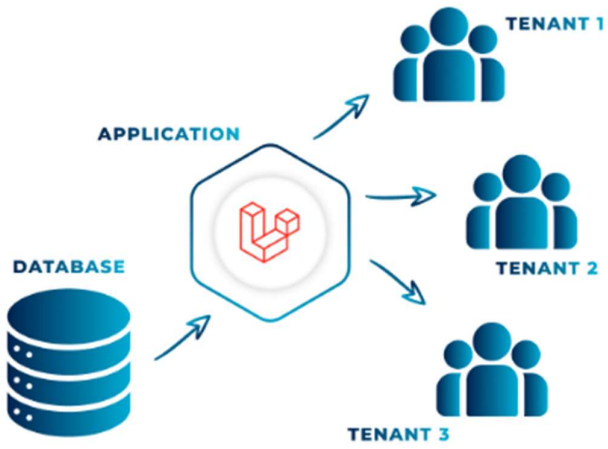


- **Cloud impact:** SOA underlies the key cloud service models (SaaS, PaaS, IaaS) by allowing the creation of complex applications through the integration of simple, standardized, internet-accessible services.

Multitenant technology

This is a software architecture that allows a single instance of an application to serve multiple customers or "tenants".

- **How it works:** The software is designed to logically segregate each tenant's data and usage, while all tenants share the underlying application instance and hardware.

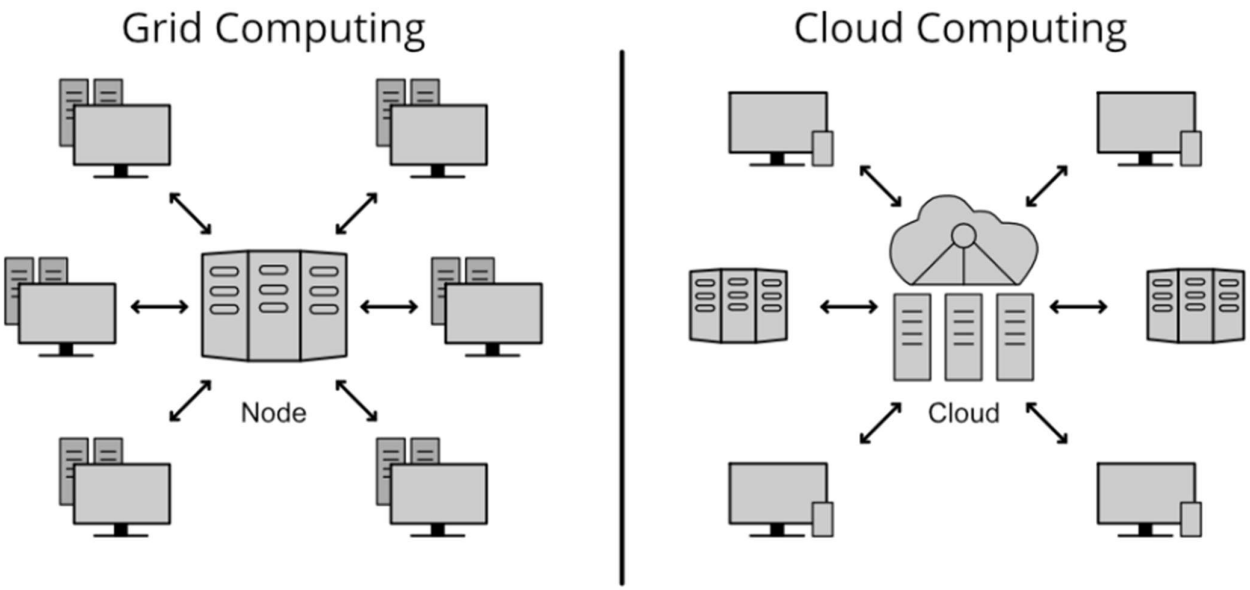


- **Cloud impact:** Multitenancy is essential for the cost-effectiveness and scalability of cloud services. It allows providers to maximize resource utilization and offer services at a lower price point.

Grid computing

Grid computing is a precursor to modern cloud computing that involves connecting a distributed network of computers to work together on a common goal.

- **How it works:** Resources are pooled to handle very large, often computationally intensive, tasks. It relies on middleware to facilitate sharing and manage the network of systems.

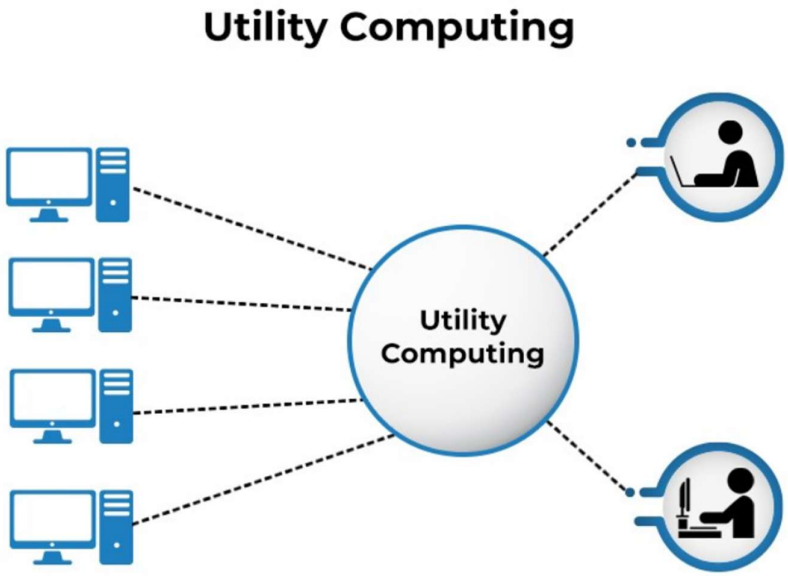


- **Cloud impact:** The concept of pooling and sharing resources in grid computing helped pave the way for the resource pooling and utility computing models used in the cloud today.

Utility computing

Utility computing is a service delivery model where a provider makes resources available and charges customers for specific usage, similar to a traditional utility like electricity.

- **How it works:** A "pay-as-you-go" or consumption-based pricing model is applied to IT services, where the customer is billed for the storage, compute, and other resources they actually use.

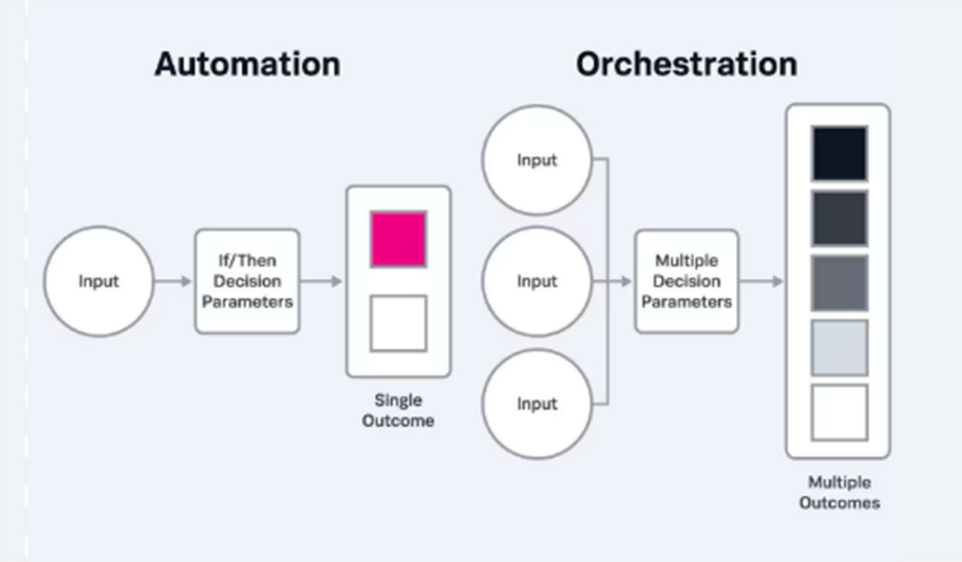


- **Cloud impact:** This economic model drives the cost-effectiveness of cloud computing, allowing businesses to avoid large upfront capital expenditures for hardware.

Automation and orchestration

These technologies automate and coordinate the deployment, management, and scaling of cloud resources with minimal human intervention.

- **How it works:** Specialized tools and platforms automate tasks like provisioning, configuration, and monitoring. This includes advanced capabilities for self-configuration and self-recovery.



- **Cloud impact:** Automation is crucial for the on-demand, rapid elasticity, and self-service characteristics of cloud computing, allowing cloud providers and customers to manage complex environments efficiently.

Enabling technologies are inventions or innovations that can be applied to drive radical change across various industries. Rather than being stand-alone products, they are fundamental tools and capabilities that accelerate the development of other derivative technologies.

A technology can be considered "enabling" if it meets several criteria:

- **Drives radical change:** Enables completely new capabilities that were previously impossible.
- **Cross-cutting application:** Can be applied and integrated across many different fields and industries.
- **Enables further innovation:** Is characterized by a rapid and continuous cycle of subsequent derivative technologies.

Key categories and examples

Enabling technologies can be broadly categorized into several key areas, many of which overlap and converge.

Information and Communications Technology (ICT)

These technologies form the digital backbone that enables widespread connectivity and data management.

- **Cloud computing:** The delivery of computing services—including servers, storage, databases, and software—over the internet. It provides scalable, on-demand resources for a vast range of applications, including Big Data analytics and AI.
- **Wireless sensor networks (WSNs):** Networks of distributed devices with sensors that monitor environmental and physical conditions. They are foundational to the Internet of Things (IoT).
- **Communication protocols:** The standardized rules that allow devices to exchange data over a network. Examples include Wi-Fi, Bluetooth, and cellular networks (like 5G).
- **Edge computing:** Moves computing and data storage closer to the physical location where the data is being generated. This reduces latency and is essential for time-sensitive applications like autonomous vehicles.

Advanced computing and digital intelligence

These technologies provide the intelligence and processing power necessary for sophisticated automation and decision-making.

- **Artificial Intelligence (AI) and Machine Learning (ML):** Used for advanced analytics, automation, and data processing. AI can enable new capabilities like predictive maintenance, autonomous systems, and personalized user experiences.
- **Digital twins:** A real-time virtual simulation of a physical asset, process, or system. Digital twins enable companies to optimize performance, predict failures, and test changes in a virtual environment.
- **Big Data analytics:** Methods for analyzing massive, diverse datasets to reveal hidden patterns, trends, and correlations.

Advanced materials and manufacturing

These innovations improve the fundamental materials and processes used to create products.

- **Nanotechnology:** The engineering of materials and systems at the molecular and atomic scale. Nanomaterials are used in everything from advanced electronics to more effective drug delivery.
- **Advanced manufacturing technologies:** Includes robotics, additive manufacturing (3D printing), and other sophisticated production techniques.
- **Advanced materials:** Lightweight composites, high-performance alloys, and other materials with properties superior to conventional ones.

Biotechnology

This field applies biological processes for technological purposes.

- **Genetic engineering:** The modification of an organism's genes, with applications in medicine, agriculture, and industry.
- **Biosensors and biochips:** Devices that use biological components to detect and measure chemical or biological substances. They are critical for diagnostics, environmental monitoring, and drug discovery.

Broader impact on industries

Enabling technologies drive transformative innovation by initiating new market cycles and changing the fundamental way industries operate.

- **Smart cities:** By combining IoT, AI, and Big Data, cities can manage traffic flow, optimize energy consumption, and improve public safety.
- **Healthcare:** Enabled by biosensors, remote monitoring systems, and AI-powered diagnostic tools, technology can assist people with disabilities to live more independently and improve patient outcomes.
- **Manufacturing:** The integration of AI, robotics, and digital twins is enabling the shift to "Industry 5.0," featuring human-machine collaboration and greater efficiency.
- **Clean energy:** Advances in photovoltaics (solar cells) and smart grid technologies contribute to decarbonization and more efficient energy systems.
- **Aerospace:** Improved materials and design tools enabled by advanced computing have enabled transformative innovations like commercial spaceflight.