DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ENGINEERING AND TECHNOLOGY
PESHAWAR, PAKISTAN

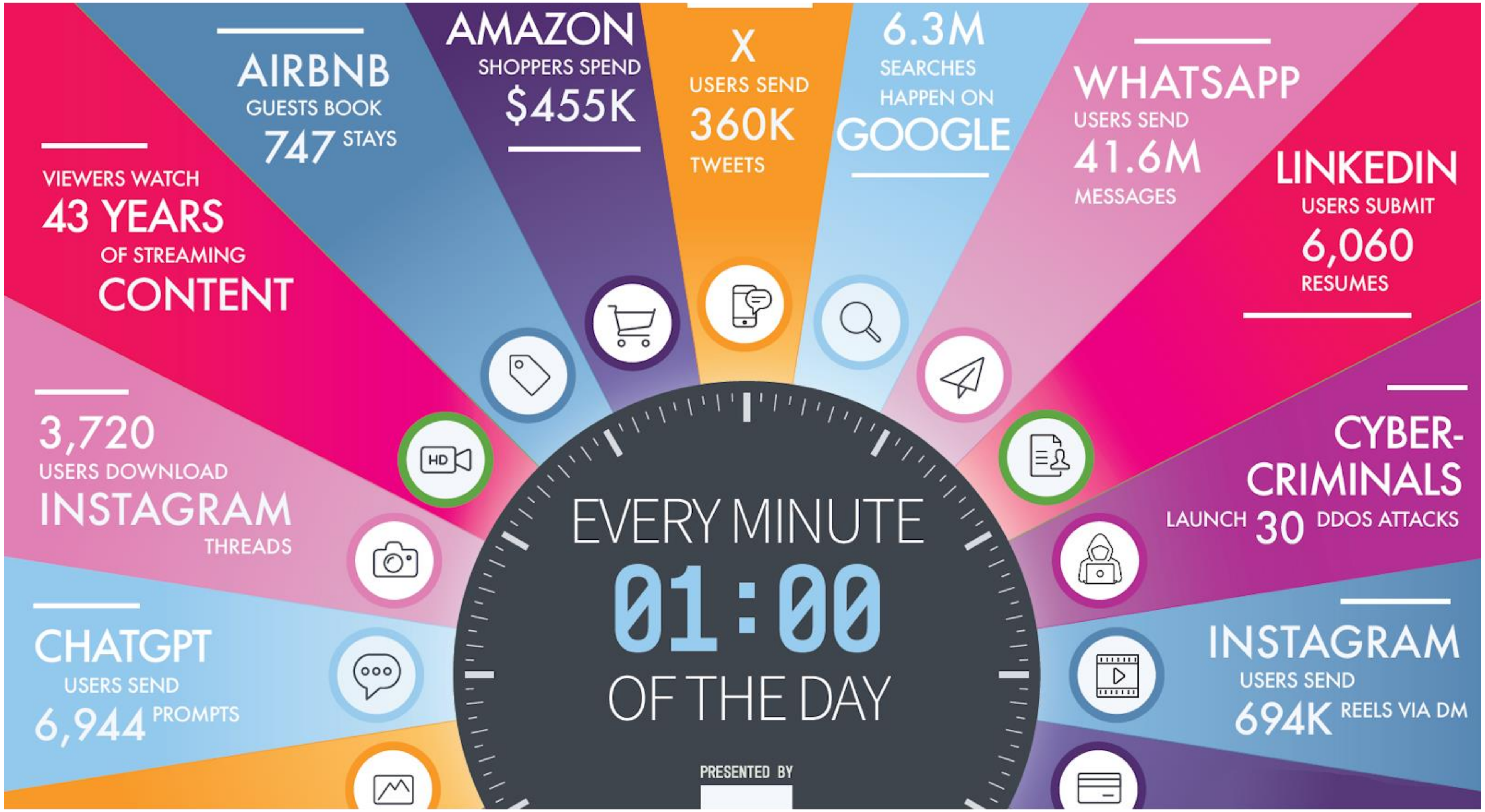
CS-401
Parallel & Distributed Computing (PDC)

MOHAMMAD
Ph.D. Computer Science
Lecturer
Department of Computer Science
University of Engineering & Technology Peshawar

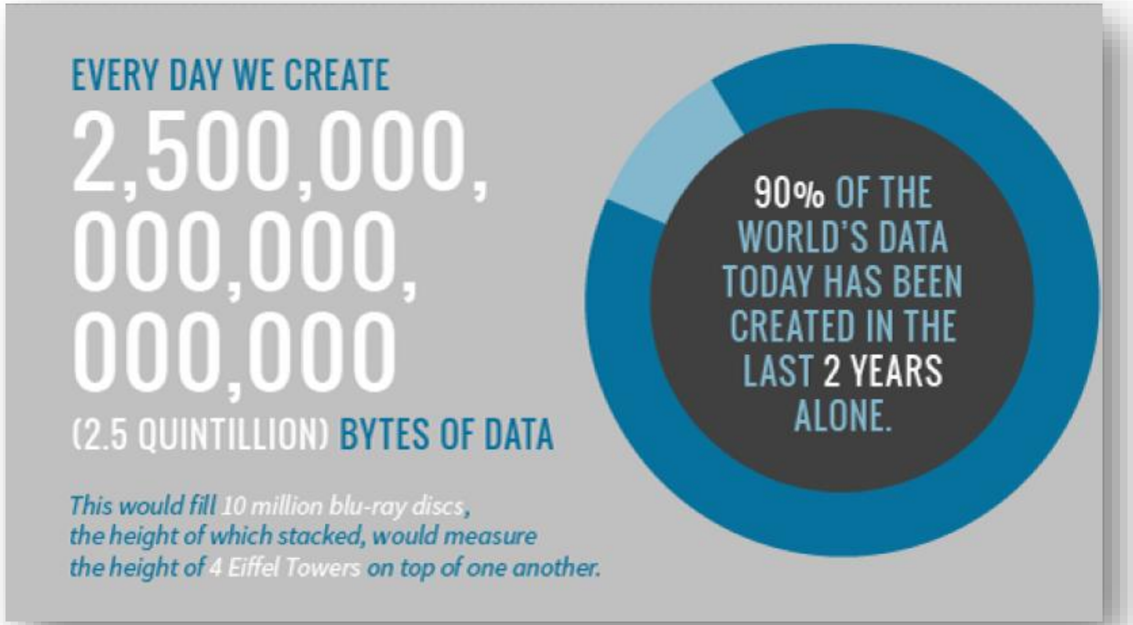
Lecture 01

60s of The Internet

60s of The Internet



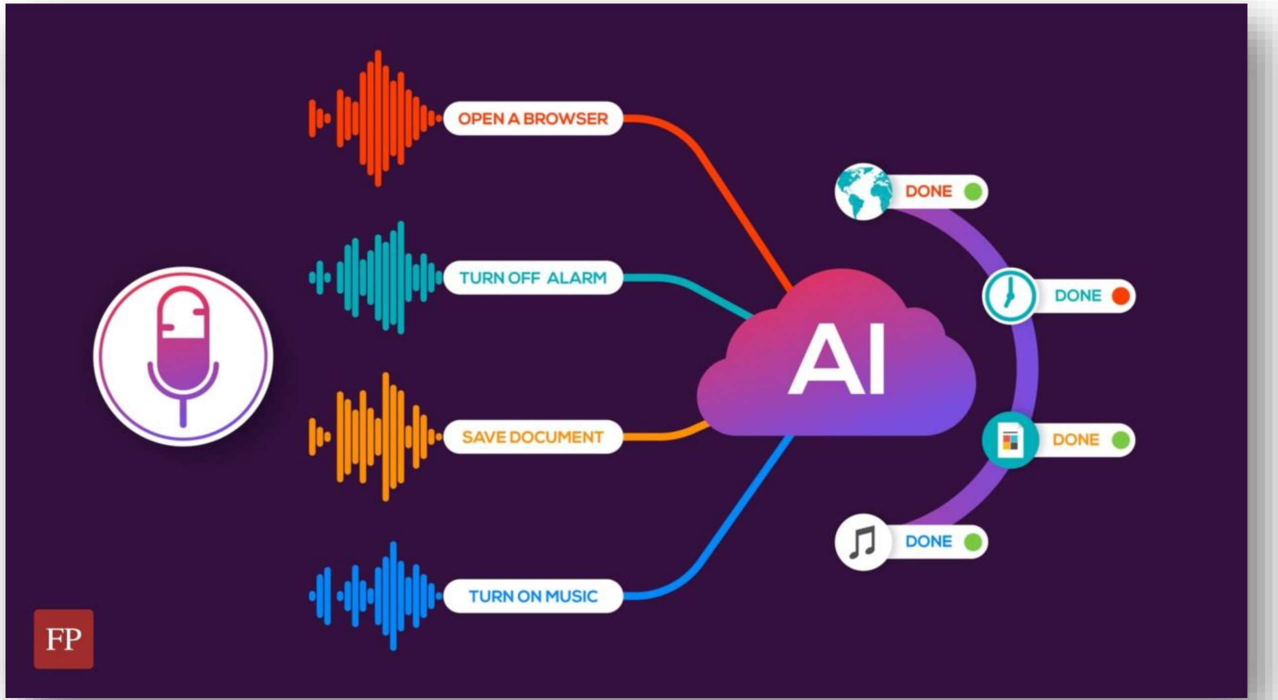
Motivation for PDC



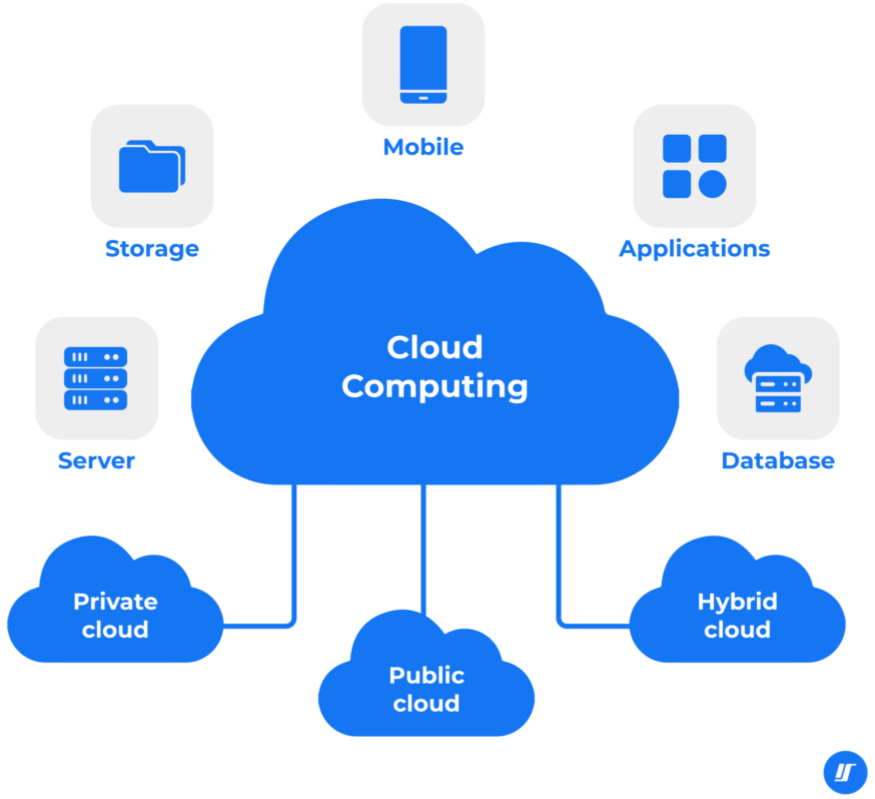
- The world is generating vast amounts of data every second.
- Traditional, single-processor computing struggles with handling big data and complex tasks efficiently.
- As problems grow larger and more complex (e.g., climate simulations, machine learning, real-time systems), parallel and distributed computing offers scalable solutions.

Real-World Applications Driving PDC - I

Real-World Applications Driving PDC - II



Real-World Applications Driving PDC - III



Why Efficiency Matters?

- Faster computations lead to faster innovation.
- Organizations rely on quick data processing to gain a competitive edge.
- Scientific advancements, like understanding protein structures or predicting natural disasters, depend on quick and accurate data analysis.

Human Genome Project

- The Human Genome Project was an international research initiative aimed at mapping and understanding all the genes of the human species.
- Launched in 1990 and completed in 2003, the project involved collaboration among scientists from around the world.

Objectives/Significance:

- **Sequence the Entire Human Genome:** Identify and map all the estimated 20,000-25,000 genes in human DNA.
- **Understand Genetic Diseases:** Analyze the genetic basis of diseases to improve diagnosis, treatment, and prevention.
- **Innovations in Biotechnology:** Led to developments in genetic testing, gene therapy, and biopharmaceuticals.
- **Data Processing:** Demonstrated the importance of parallel and distributed computing in handling large-scale genomic data, reducing what would have taken years to complete to just a few months.

Why Learn Parallel and Distributed Computing?

Demand for Skills:

- Parallel and distributed computing is one of the most in-demand skills in tech and research today.
- It is central to areas like cloud computing, AI, cybersecurity, and big data.

Career Growth:

- Learning PDC opens opportunities in cutting-edge industries, including data science, high-performance computing, system architecture, and more.

Innovation:

- Future advancements—from quantum computing to autonomous AI—are dependent on innovations in parallel and distributed systems.

Parallel computing is not just for today’s challenges; it’s the foundation for future technologies, including quantum computing and advanced AI.

What We Study in PDC?

Fundamentals of PDC

- Definitions, historical context, and evolution of computing paradigms.

Parallel Computing Concepts

- Types of parallelism and performance metrics.

Distributed Computing Concepts

- Principles, characteristics, and challenges of distributed systems.

Programming Models

- OpenMP, MPI, CUDA, and MapReduce.

Synchronization and Coordination

- Mechanisms for process coordination (locks, semaphores).

Performance Analysis and Optimization

- Techniques for profiling and optimizing applications.

Applications of PDC

- Real-world use cases across various domains.

Real-World Applications and Projects After Learning PDC

Scientific Research

- **Example**: Climate modeling, simulations in physics and chemistry.
- **Usage**: Implementing parallel algorithms for data processing and analysis.

Big Data Analytics

- **Example**: Analyzing massive datasets in healthcare or finance.
- **Usage**: Using MapReduce and distributed systems to process and derive insights.

Artificial Intelligence and Machine Learning

- **Example**: Training deep learning models on large datasets.
- **Usage**: Leveraging GPUs with CUDA for faster model training and inference.

Real-World Applications and Projects After Learning PDC

Cloud Computing Solutions

- **Example**: Developing scalable applications using cloud platforms.
- **Usage**: Designing distributed systems that utilize cloud resources efficiently.

Real-Time Systems

- **Example**: Autonomous vehicles and real-time data processing.
- **Usage**: Implementing synchronization techniques to ensure timely responses.

Blockchain Technology

- **Example**: Cryptocurrencies and decentralized applications (dApps).
- **Usage**: Utilizing distributed ledger technology for secure and transparent transactions.

What is Computing?

What is Computing?

Computing is the process of using computational devices (such as computers, servers, and specialized hardware) to perform operations on data, execute algorithms, and solve problems through systematic manipulation of information. It encompasses a range of activities, including:

- **Data Processing:** The collection, storage, and transformation of data into meaningful information.
- **Algorithm Execution:** The implementation of step-by-step procedures or formulas to perform calculations or solve problems.
- **Software Development:** The design, creation, and maintenance of applications that perform specific tasks.
- **System Operations:** The management of hardware and software resources to ensure efficient computing processes.
- **Networking:** The interconnection of computing devices to share resources and communicate.

What is Computing?

Main Characteristics:

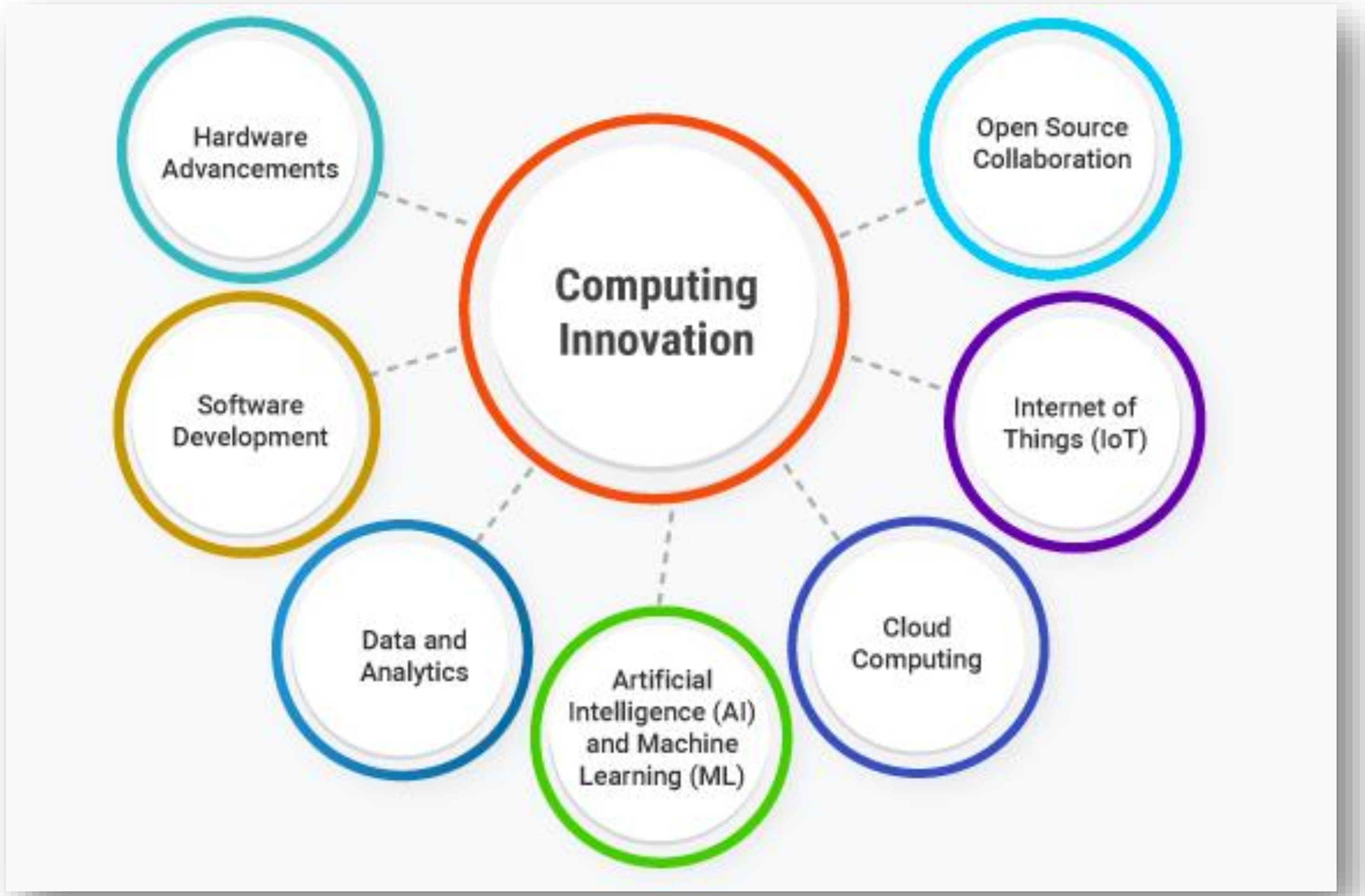
- **Automated**: Involves automated processes that execute tasks without human intervention.
- **Quantitative**: Primarily concerned with numerical and logical calculations.
- **Dynamic**: Capable of adapting to different types of input and processing requirements.
- **Resource-Intensive**: Often requires significant computational power, storage, and memory, especially for complex tasks.

Modern Computing Fields:

- Artificial Intelligence (AI)
- Cloud Computing
- Quantum Computing
- Cybersecurity
- Distributed Systems

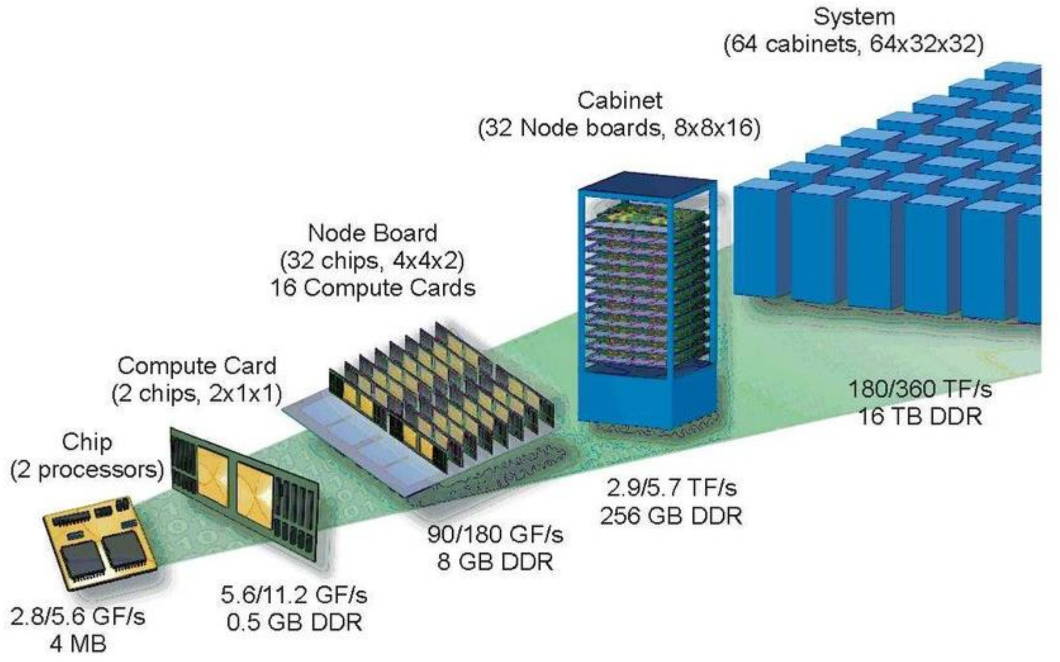
- *Computing forms the foundation of modern technology and powers the digital world.*

Parallel & Distributed Computing



Super Computer

- A supercomputer is a highly advanced computer system designed for processing vast amounts of data and performing complex calculations at extremely high speeds. It consists of thousands, sometimes millions, of processors that work together to solve problems too complex for regular computers.
- Supercomputers are used in fields like scientific simulations, weather forecasting, and advanced research.



Parallel & Distributed

The word **parallel** generally means side by side or running alongside each other without intersecting. In different contexts, it carries specific meanings:

- **Parallel (General):** Side by side, occurring at the same time without intersecting.
- **Parallel (Maths):** Two or more lines that remain equidistant and never intersect, no matter how far extended.
- **Parallel (Computing):** Multiple tasks executed simultaneously on different processors to solve a problem faster.

The word **distributed** means spread out or shared across multiple locations or entities. It implies that something is divided among different parts rather than being concentrated in one place.

- **Distributed (General):** Spread out or shared across multiple locations or entities.
- **Distributed (Computing):** Tasks or data are divided and processed by different computers working together over a network.

Parallel Computing

Parallel computing refers to the simultaneous execution of multiple tasks on multiple processors to solve a computational problem faster.

Core Concepts:

- **Multiple Processors**: Uses multiple CPUs or cores within a single machine.
- **Simultaneous Execution**: Tasks are divided and executed concurrently to save time.
- **Shared Memory Model**: All processors access the same memory space.

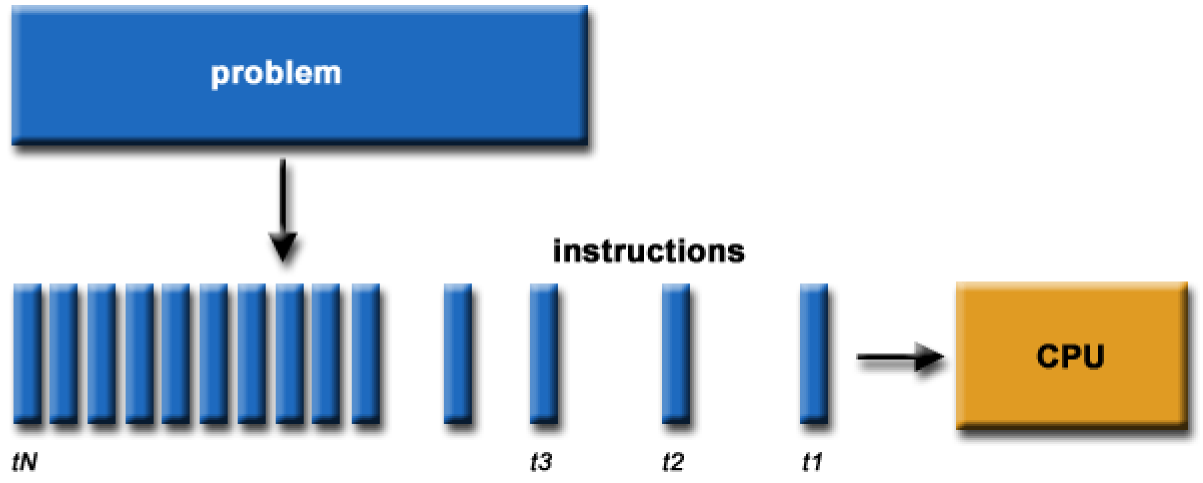
Examples:

- **High-Performance Computing**: Scientific simulations, weather forecasting.
- **Multithreading**: Running multiple threads in a program on different cores.

Parallel Computing - I

Traditionally, software has been written for serial computation:

- To be run on a single computer having a single Central Processing Unit (CPU);
- A problem is broken into a discrete series of instructions.
- Instructions are executed one after another.
- Only one instruction may execute at any moment in time.



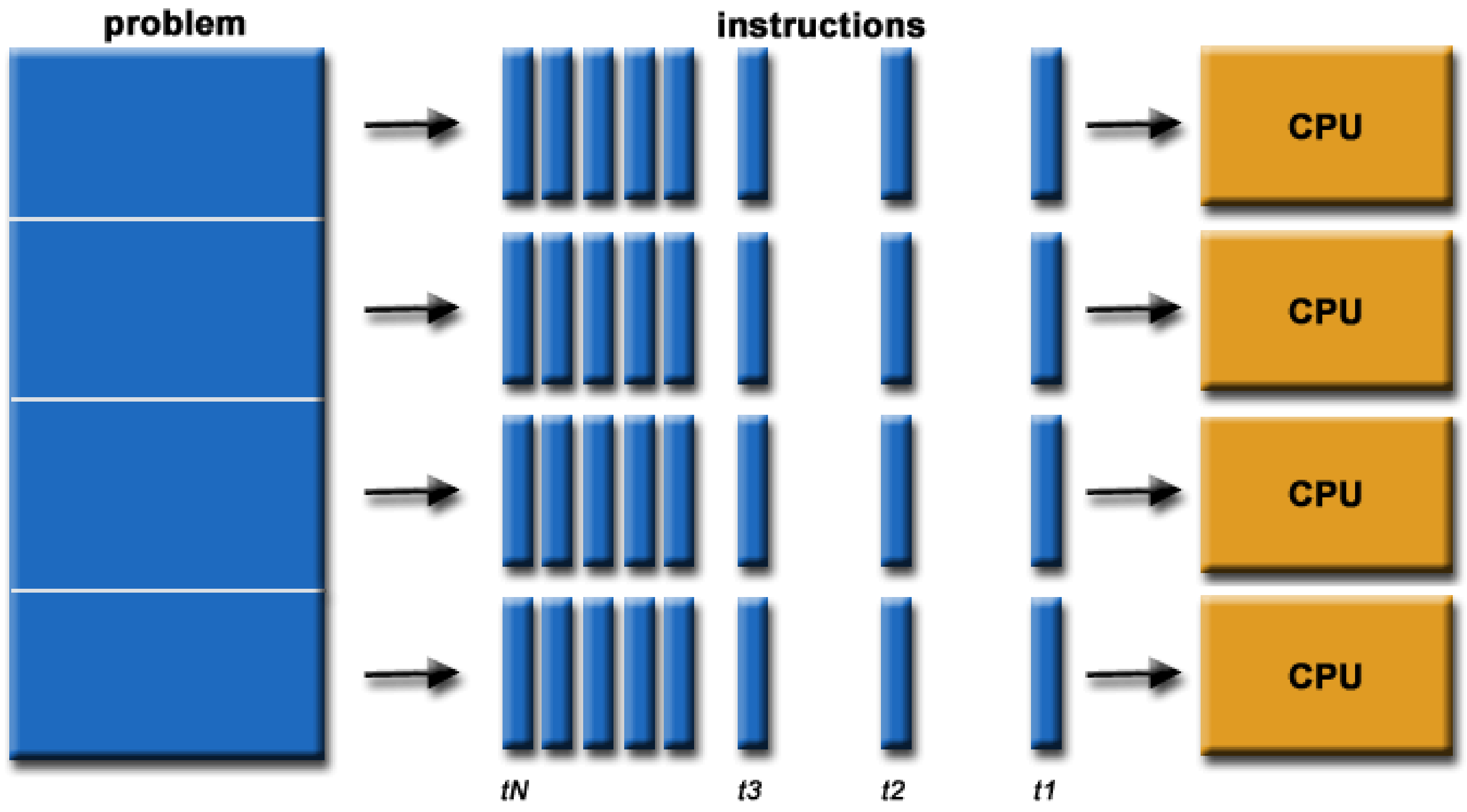
Parallel Computing - II

In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem.

- To be run using multiple CPUs
- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions

Instructions from each part execute simultaneously on different CPUs

Parallel Computing - II



Basic Components in Parallel Computing

Multi-core Processors:

- Modern CPUs have multiple cores, allowing parallel execution of tasks within a single chip.

Threading:

- Threads are the smallest unit of processing that can be scheduled by an operating system, allowing multiple threads to run in parallel within a single process.

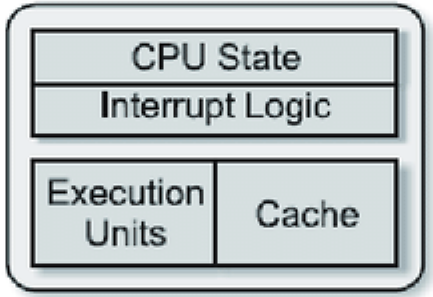
Parallel Algorithms:

- Specialized algorithms designed to efficiently divide and conquer tasks among multiple processors.

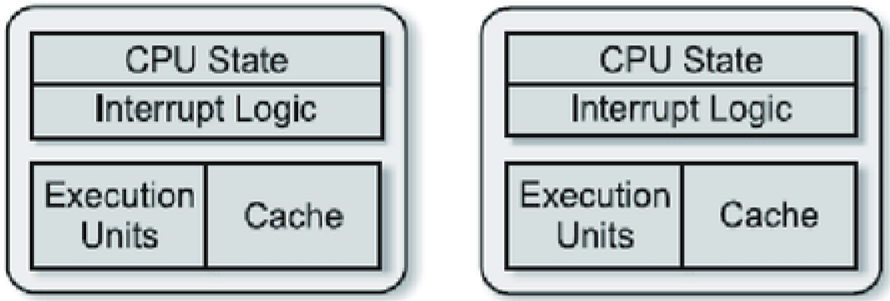
Load Balancing:

- Distributing workloads evenly across processors to optimize resource use and prevent bottlenecks.

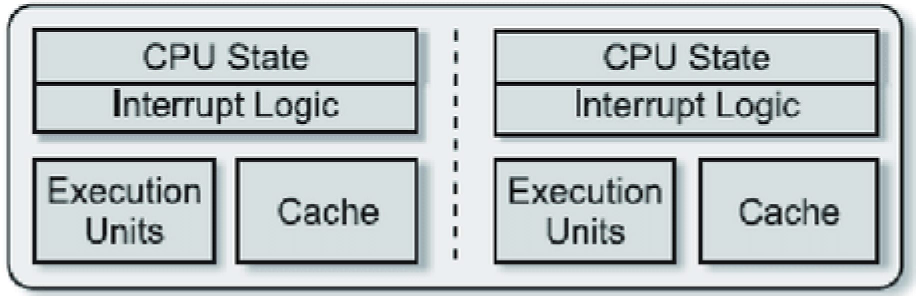
Multi-Core Vs Multi-Processor



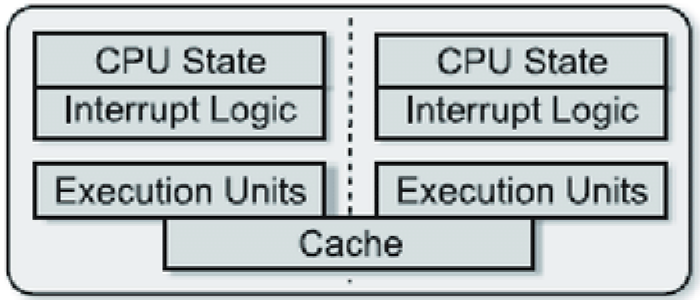
(a) Single core



(b) Multiprocessor



(c) Multi-core



(d) Multi-core with shared cache

Importance of Parallel Computing

Faster Processing:

- By dividing tasks among multiple processors, parallel computing significantly reduces computation time.

Efficiency:

- Maximizes the use of available resources, improving overall system performance.

Tackling Large-Scale Problems:

- Allows complex tasks (e.g., scientific simulations, climate modeling) to be processed efficiently.

Improved Scalability:

- Can handle larger datasets or more complex problems by adding more processors.

Real-Time Processing:

- Enables real-time applications like video processing, 3D rendering, and machine learning.

Distributed Computing

Distributed computing involves multiple independent computers working together over a network to achieve a common goal.

Core Concepts:

- **Multiple Machines:** Different computers collaborate over a network.
- **Distributed Memory Model:** Each machine has its own memory.
- **Network Communication:** Data and tasks are passed over networks between machines.

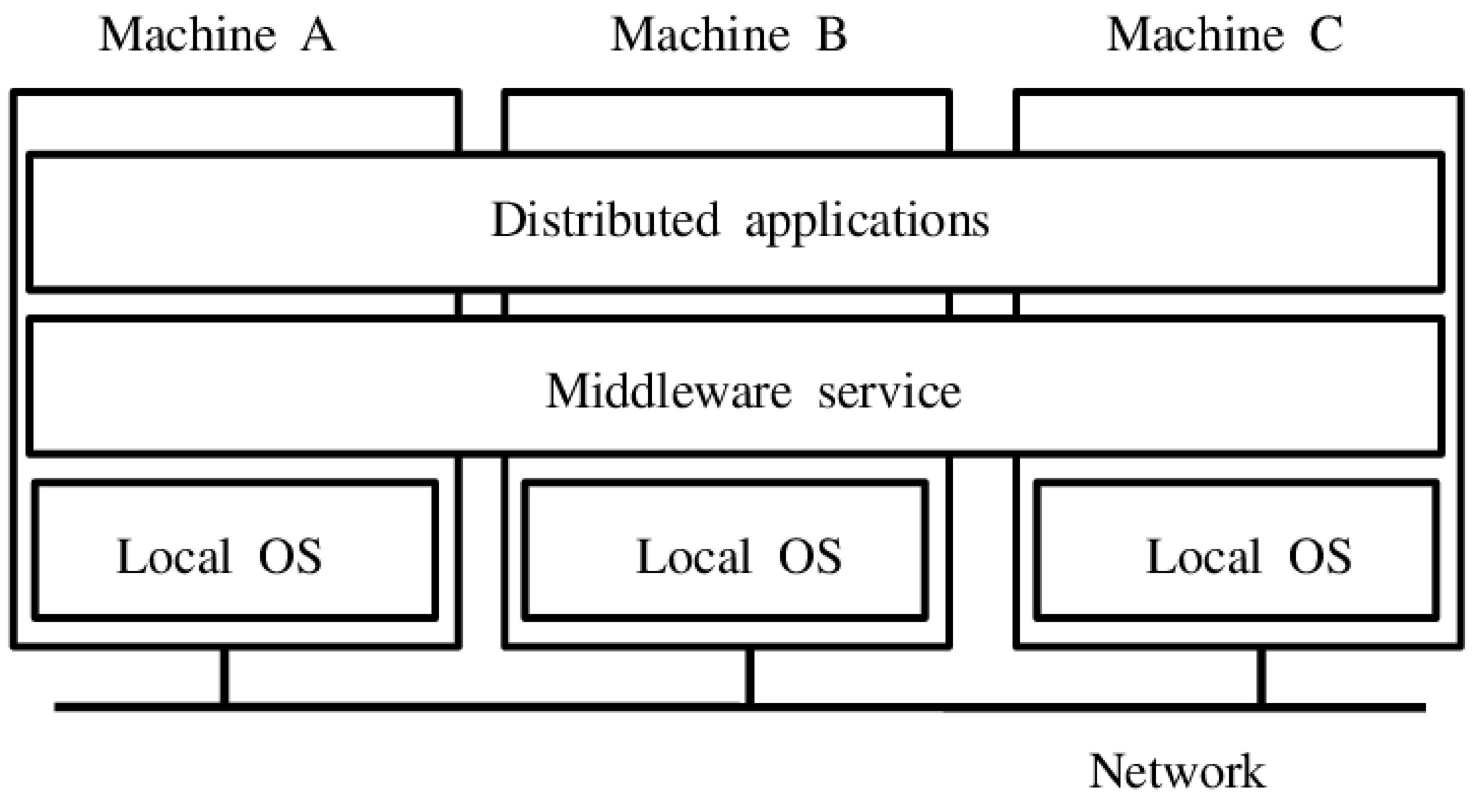
Examples:

- **Cloud Computing:** Services like Google Cloud or Amazon AWS.
- **Blockchain:** Decentralized networks working on a shared ledger.

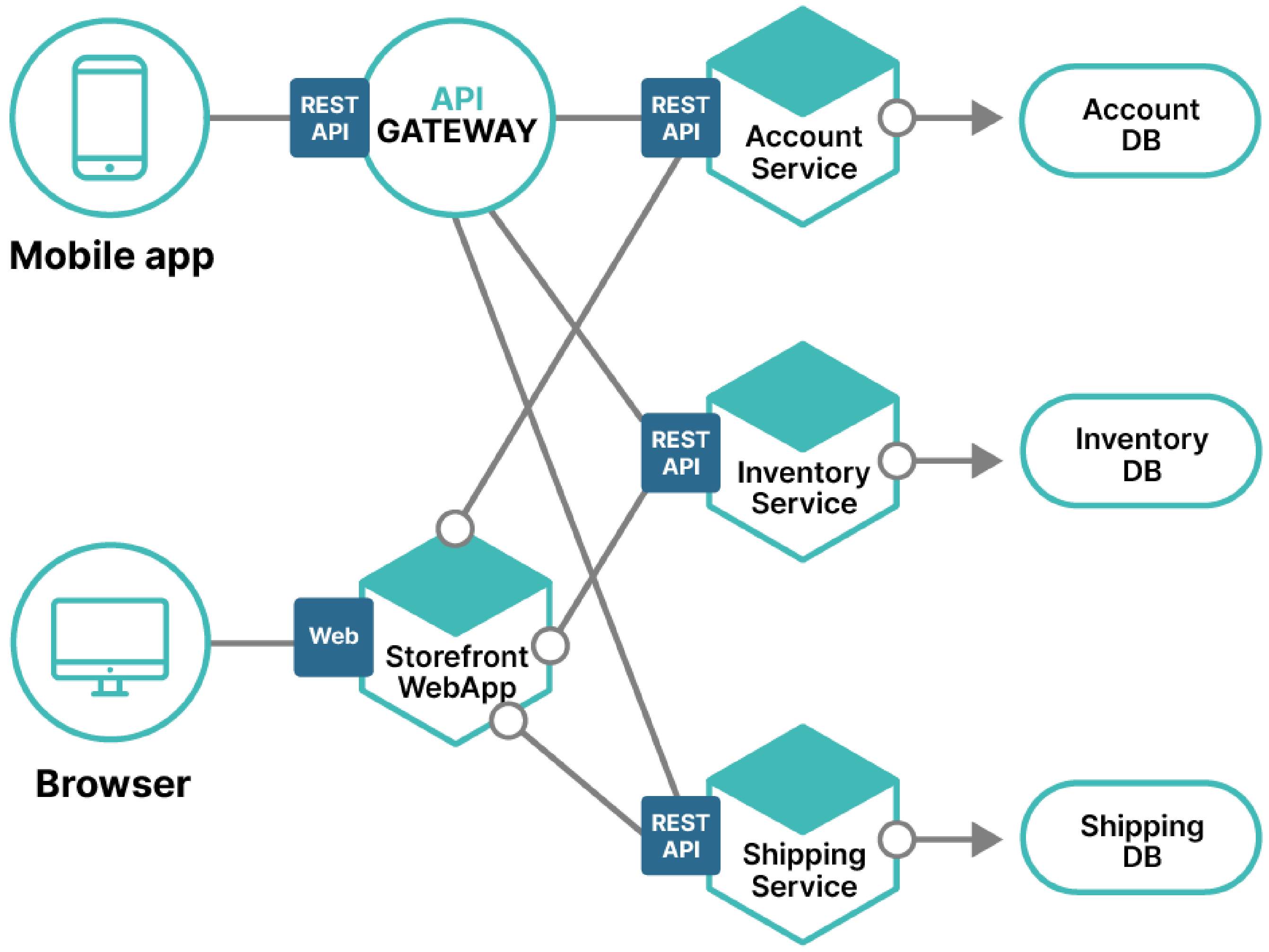
Advantages of Distributed Computing

- **Fault Tolerance:** If one machine fails, the others can continue the task.
- **Scalability:** Easy to add more machines as needed.
- **Geographical Distribution:** Machines can be located in different physical locations.

Distributed Computing Overview



Distributed Computing Overview



Components of a Distributed System

Nodes/Clients:

- Independent machines (computers, servers) that contribute to the system.

Network:

- The communication infrastructure that connects the nodes and facilitates data exchange.

Middleware:

- Software that manages communication between the nodes and ensures they work together seamlessly.

Data Replication:

- Data is replicated across multiple nodes to ensure reliability and fault tolerance.

Types of Distributed Systems

Client-Server Systems:

- Clients request services from servers. Servers process requests and send responses (e.g., web applications).

Peer-to-Peer (P2P) Systems:

- All nodes have equal roles, and they share resources directly with each other (e.g., file-sharing systems like BitTorrent).

Distributed Databases:

- Data is distributed across multiple nodes for high availability and faster access (e.g., Google's Bigtable, Amazon's DynamoDB).

Cloud Computing Systems:

- Cloud-based services where distributed nodes handle storage, processing, and computing (e.g., AWS, Microsoft Azure).

Importance of Distributed Computing

Resource Sharing:

- Distributes workload across multiple systems, utilizing combined computing power from geographically dispersed machines.

Fault Tolerance:

- Increases reliability by ensuring that the system continues to function even if individual machines fail.

Scalability:

- Easily scales by adding more machines to the network, making it suitable for growing applications like cloud computing.

Cost Efficiency:

- Reduces cost by using a network of smaller machines rather than one extremely powerful, expensive machine.

Collaboration and Communication:

- Enables collaborative work across different locations, which is essential for global systems like th internet and cloud-based applications.

Challenges in Distributed Systems

Concurrency

- Managing tasks across multiple machines.

Latency

- Delays in communication across the network.

Coordination

- Synchronizing between processes.

Fault Tolerance

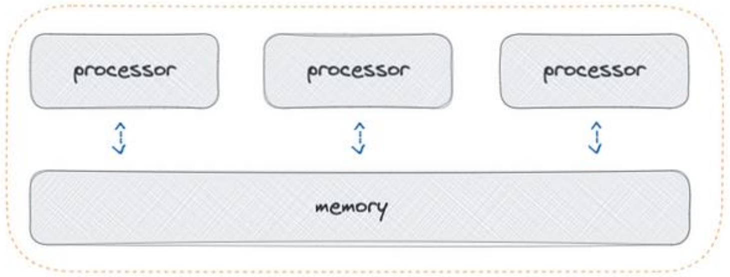
- Handling failures without loss of service.

Security

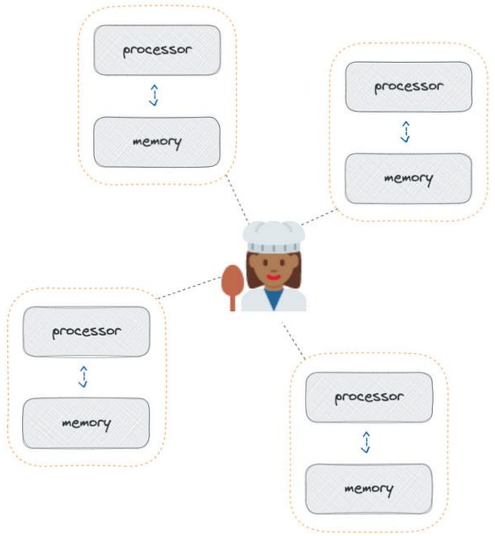
- Ensuring secure communication and data protection.

Parallel vs Distributed Computing

Feature	Parallel Computing	Distributed Computing
System Type	Single machine with multiple CPUs	Multiple independent machines
Memory	Shared memory	Distributed memory
Communication	Internal to system	Over network
Example Applications	Multithreading, GPU computing	Cloud services, Blockchain



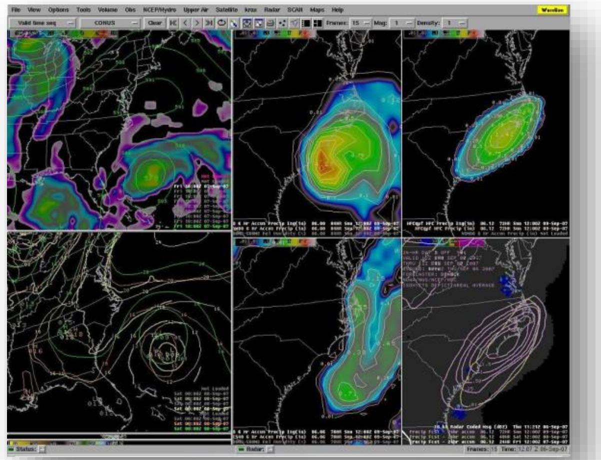
Parallel Computing



Distributed Computing

Applications of Parallel & Distributed Computing

Parallel Computing



Weather Simulations

GPU

Distributed Computing

Large-scale data analysis

Internet-based applications (e.g., Google search, Netflix)

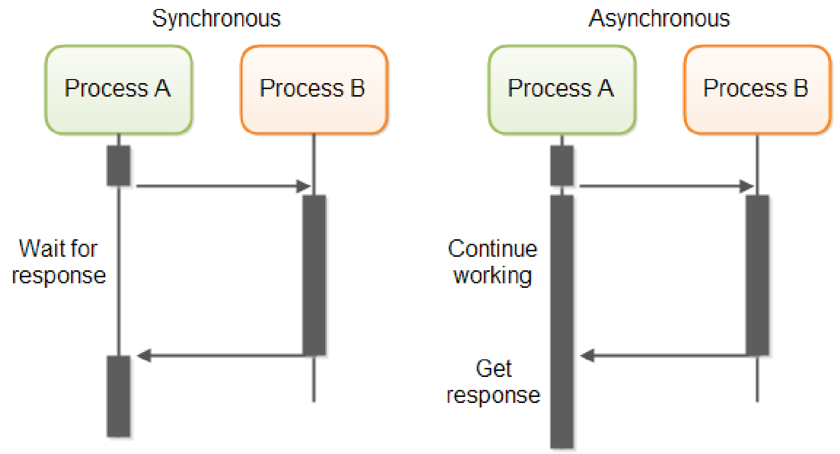
Asynchronous vs. Synchronous Computation/Communication

Synchronous:

- Processes communicate in lock-step.
- Wait for each other at synchronization points.
- Examples: MPI (Message Passing Interface), tightly coupled systems.
- Advantages: Predictability, easier to debug.
- Disadvantages: Idle time, lower efficiency in heterogeneous systems.

Asynchronous:

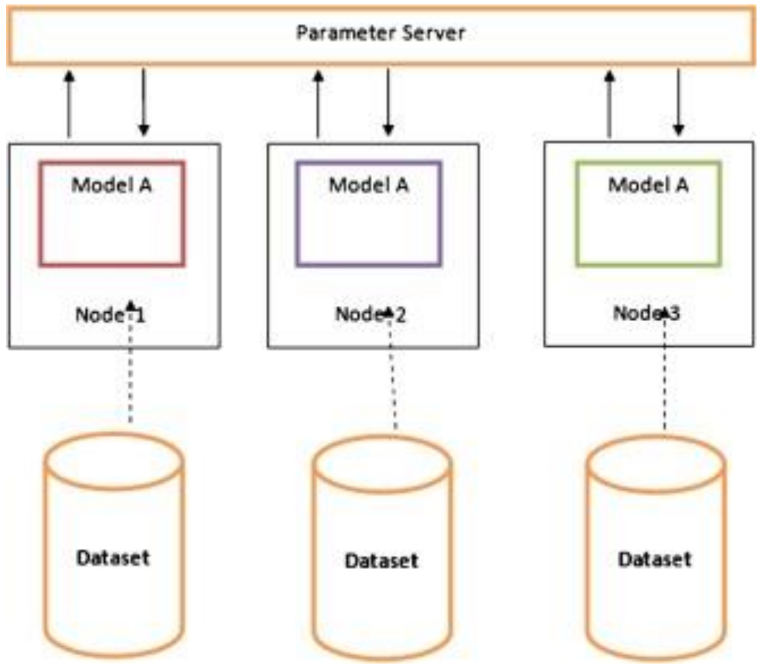
- Processes communicate without waiting.
- More flexible and decoupled.
- Examples: Actor models, Event-driven programming.
- Advantages: Higher efficiency, lower idle times.
- Disadvantages: More complex to design, harder to debug.



Programming Models in Parallel and Distributed Computing

Data Parallelism

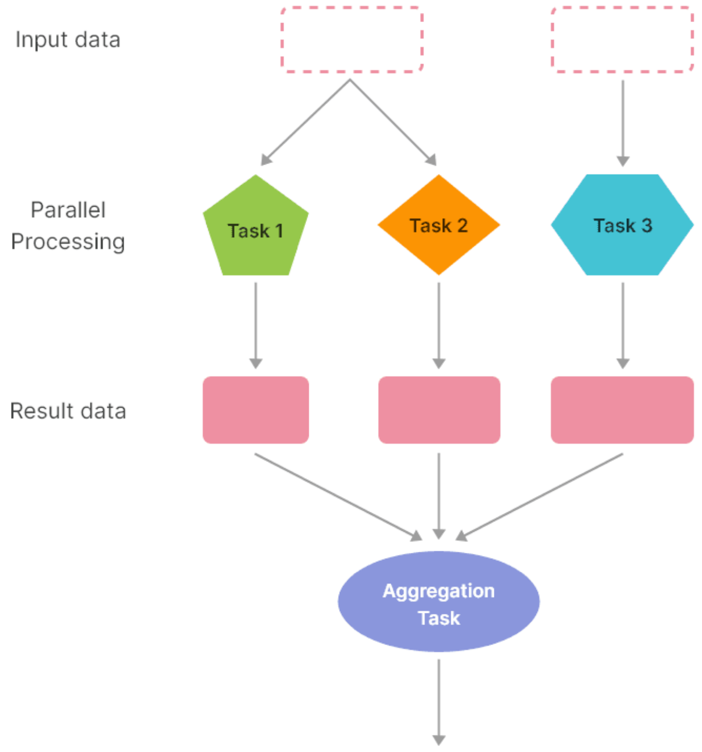
- Involves distributing data across multiple computing nodes and performing the same operation on each piece of data concurrently.
- Using a SIMD (Single Instruction, Multiple Data) approach where the same instruction is applied to different data elements simultaneously.
- Image processing, numerical simulations, and machine learning tasks where large datasets need to be processed.



Programming Models in Parallel and Distributed Computing

Task Parallelism

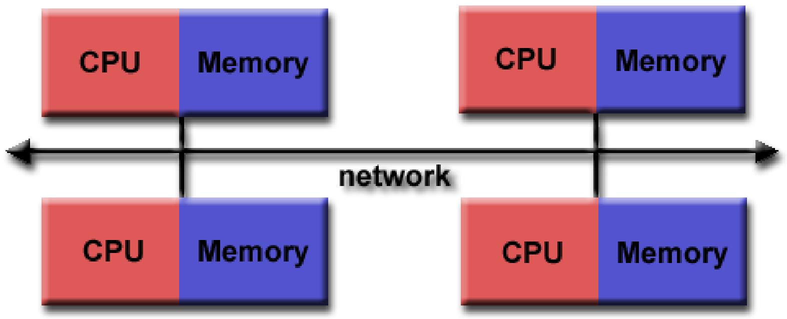
- Focuses on dividing a program into distinct tasks that can be executed independently in parallel.
- In a web server, handling multiple client requests concurrently through separate threads or processes.
- Applications with distinct tasks that can run concurrently, such as data analysis, web services, or simulations with independent components.



Programming Models in Parallel and Distributed Computing

Process-Centric Parallelism

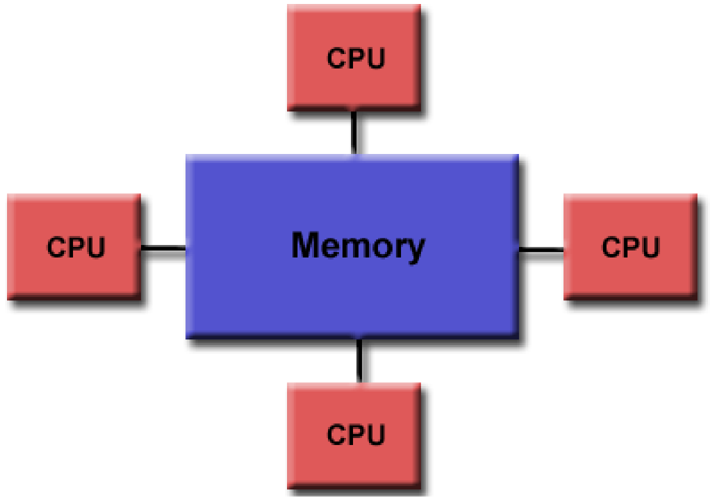
- Centers around the concept of processes, where each process can run on a separate processor or machine, communicating with others as needed.
- Distributed computing frameworks like MPI (Message Passing Interface) where different processes exchange messages to coordinate their actions.
- High-performance computing applications, simulations, and large-scale data processing.



Programming Models in Parallel and Distributed Computing

Shared Memory Model

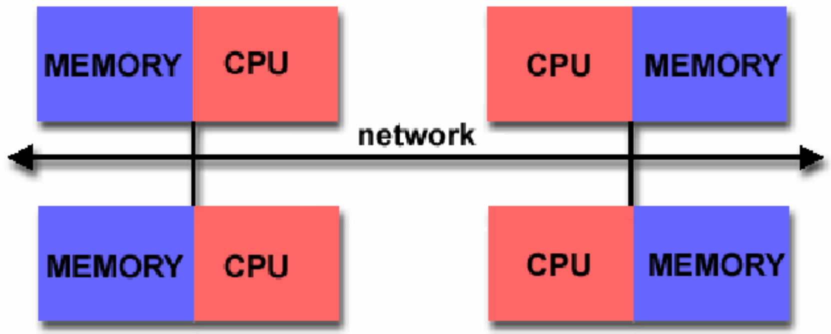
- Multiple processors or threads access a common memory space for communication and data sharing.
- OpenMP, a widely used API for shared-memory programming, where threads can read and write shared variables.
- Applications running on multi-core or multiprocessor systems where fast data access is crucial.



Programming Models in Parallel and Distributed Computing

Distributed Memory Model

- Each processor has its own local memory, and data is exchanged through message passing.
- MPI is commonly used for distributed memory programming, allowing processes to communicate by sending and receiving messages.
- Clusters of computers or grid computing where nodes are connected over a network.



Programming Models in Parallel and Distributed Computing

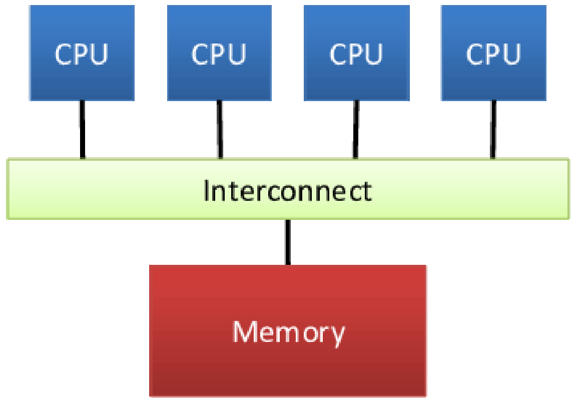
Hybrid Models

- Combine elements of both shared and distributed memory models, allowing flexibility in programming and performance optimization.
- Using MPI for inter-node communication and OpenMP for intra-node threading on a cluster of multi-core processors.
- Complex applications requiring both high-level scalability and efficient memory usage.

Shared Memory vs. Distributed Memory

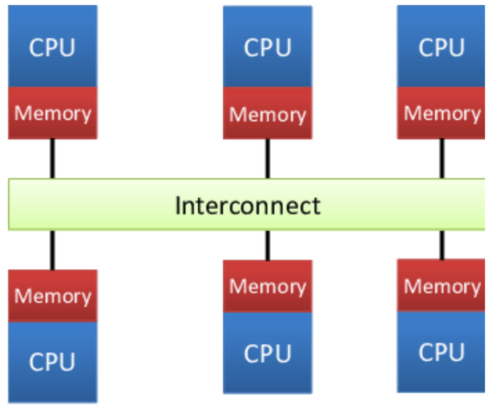
Shared Memory:

- Single memory space shared by multiple processors.
- Advantages: Easy data sharing and communication.
- Disadvantages: Limited scalability, bottlenecks.
- Examples: Multicore systems, SMP (Symmetric Multi-Processing).



Distributed Memory:

- Each processor has its own private memory.
- Advantages: Scalable, efficient in large systems.
- Disadvantages: Complex communication (data must be explicitly passed).
- Examples: Clusters, cloud computing.



Parallel Computing Programming Models

OpenMP:

- A set of compiler directives and libraries for multi-platform shared memory multiprocessing programming.
- Simplifies the process of parallelizing code through pragmas in C, C++, or Fortran.

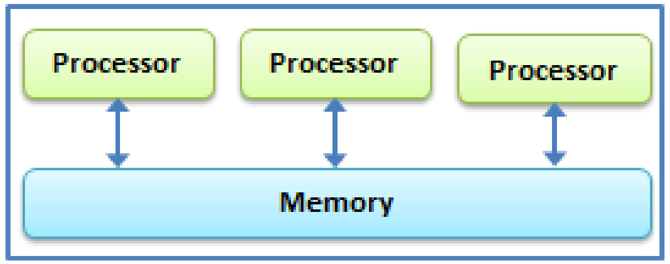
CUDA:

- A parallel computing platform and programming model developed by NVIDIA.
- Designed specifically for programming graphics processing units (GPUs) for high-performance tasks.

Advantages:

- Reduced communication overhead due to shared memory access.
- Easier to program for parallel tasks within a single node.

Parallel Computing



Distributed Computing Programming Models

Message Passing Interface (MPI):

- Allows processes to communicate with one another by sending and receiving messages.
- Commonly used in high-performance computing for parallel tasks.

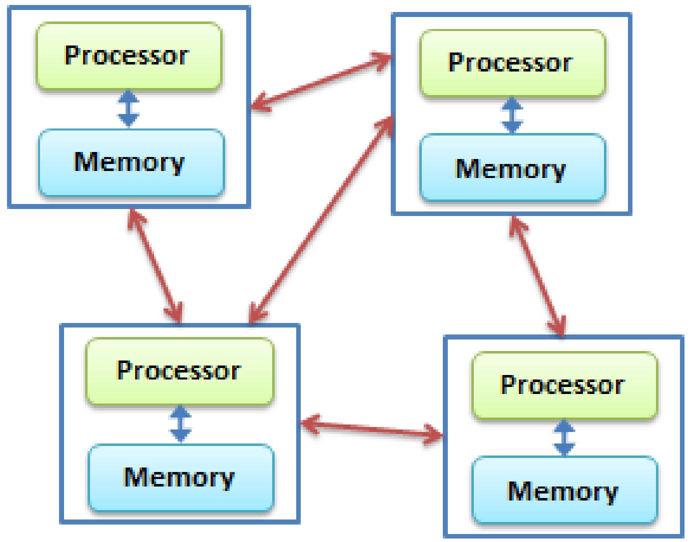
Remote Procedure Call (RPC):

- A protocol that allows a program to execute code on a remote server as if it were a local call.
- Simplifies the process of network communication.

Advantages:

- Adding more nodes can enhance performance.
- Distributes workload across multiple systems.

Distributed Computing



THE END