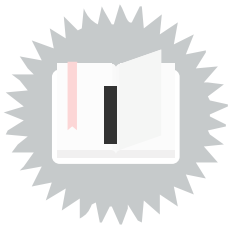
Big Data and Cloud Computing

Dr. Walid Miloud Dahmane khemis miliana University SMI Department Email : w.miloud-dahmane@univ-dbkm.dz

1.0 March 2025

Table of contents

Chapter objectives



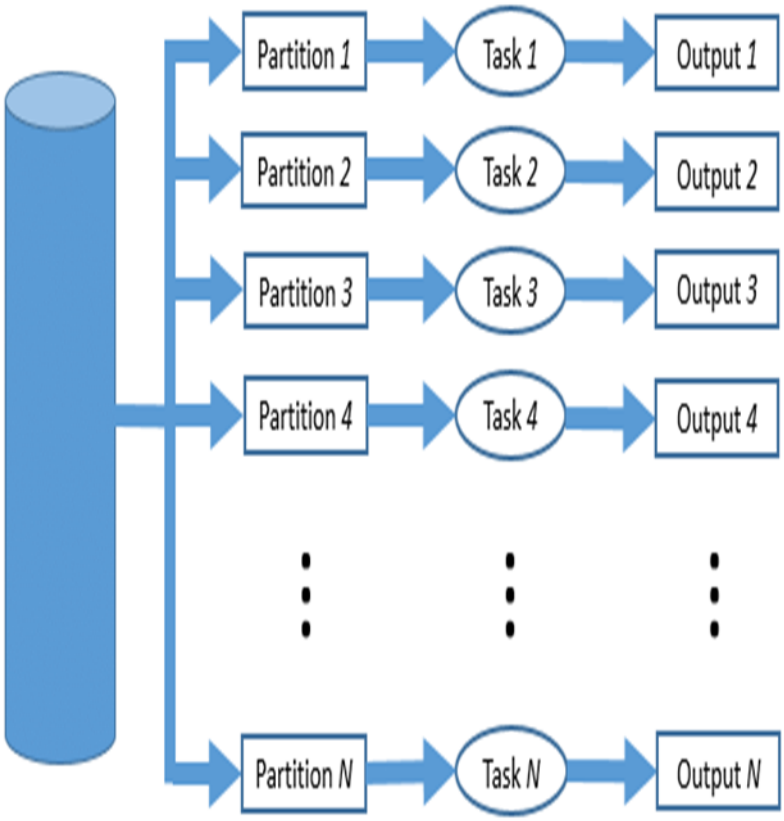
By the end of this chapter, the student will be able to:

- Understand how data can be processed efficiently by using multiple computers at the same time (parallel and distributed processing).
- Learn about tools and systems like Hadoop and clusters that help manage and process large amounts of data.
- Explore different processing methods (workload management, stream processing, and batch processing).

Chapter 3: Big Data Processing Concepts

1. Parallel data processing

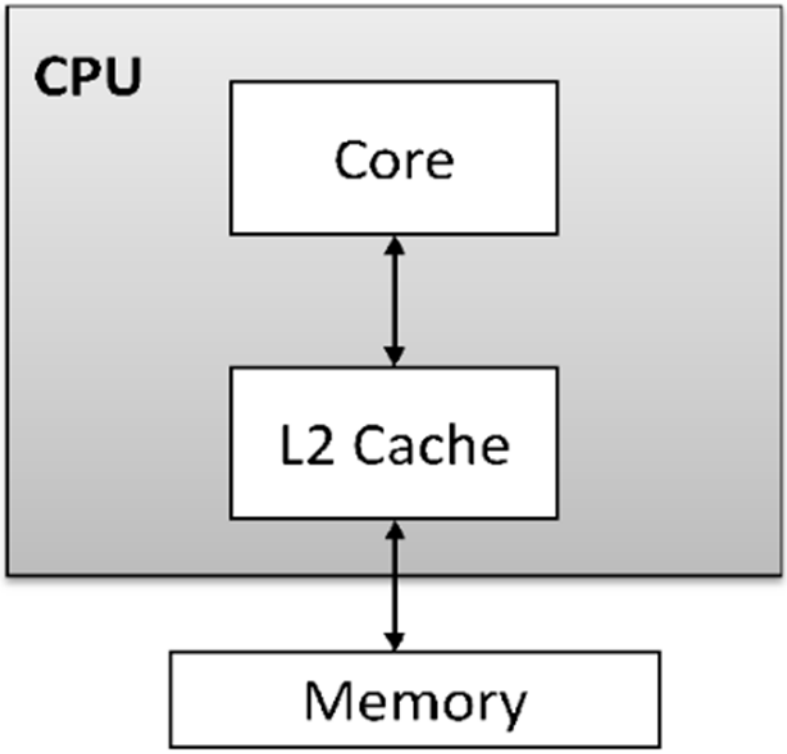
- Parallel data processing saves time, allowing the execution of applications in a shorter wall-clock time. Solve Larger Problems in a short point of time.
- Compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real-world phenomena.



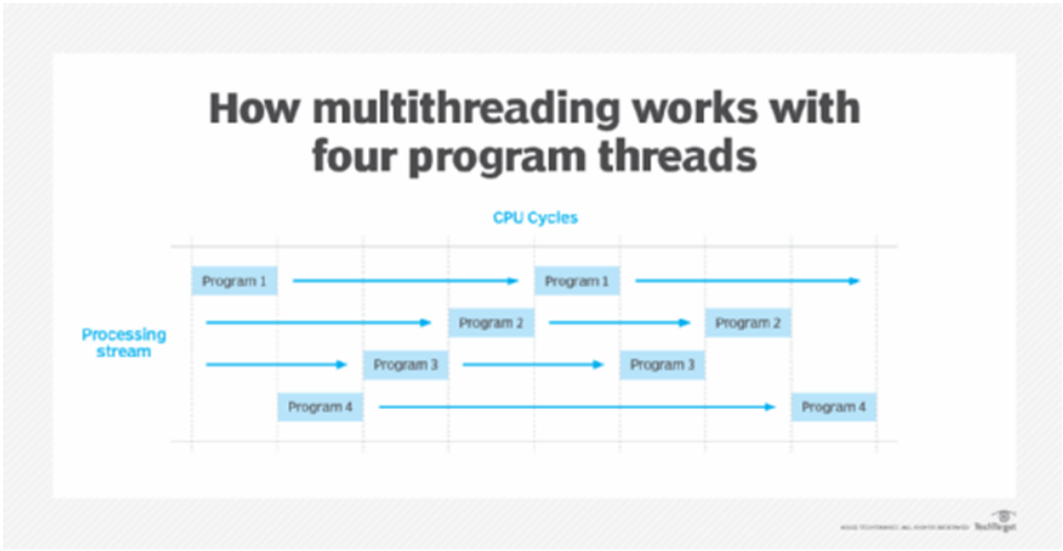
- Parallel processing is a computing technique when multiple streams of calculations or data processing tasks co-occur through numerous central processing units (CPUs) working at the same time.
- Parallel computing is becoming critical as more Internet of Things (IoT) sensors, and endpoints need real-time data. Given how easy it is to get processors and GPUs (graphics processing units) today through cloud services, parallel processing is a vital part of any micro service rollout.
- The concepts of processing are represented in the following types:

1- A single-core processor is a type of CPU (Central Processing Unit) that has only one processing unit or core. It can execute one instruction or task at a time, meaning it processes instructions sequentially. While this type of processor can handle basic computing tasks effectively, it is less

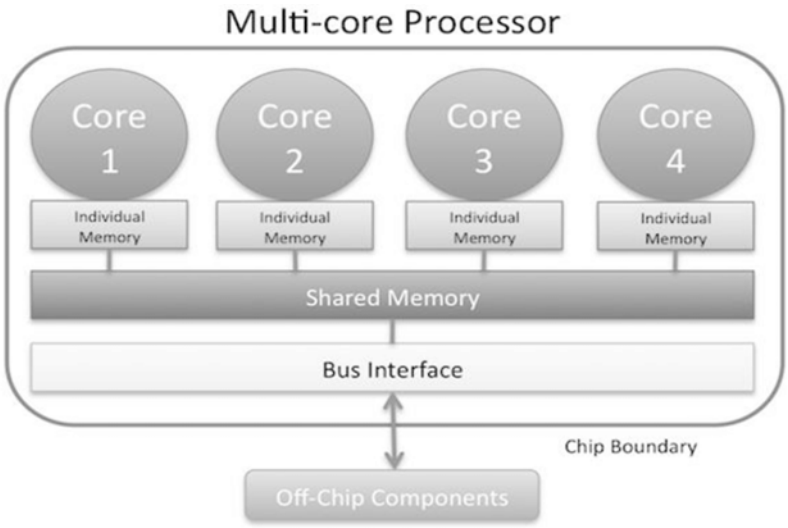
efficient than multi-core processors, which can handle multiple tasks simultaneously. Single-core processors were more common in early computers, but modern systems often use multi-core processors to improve performance and multitasking capabilities.



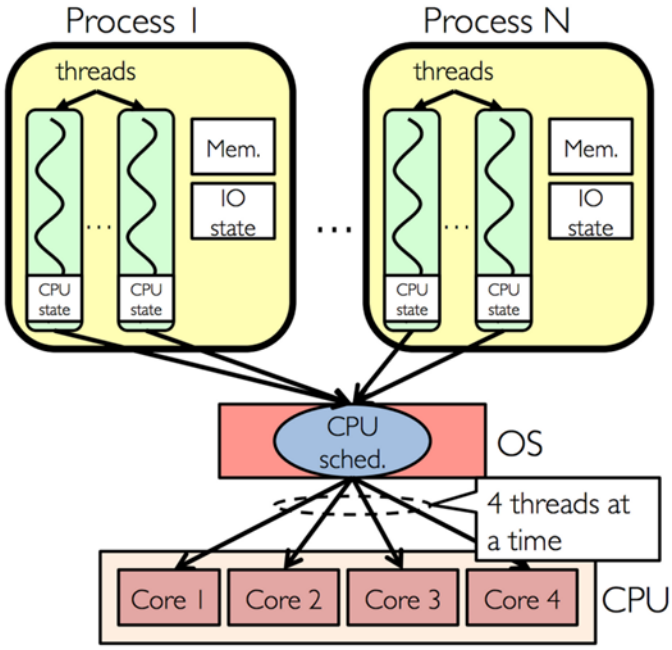
2- Multi-threading is a technique used in computing that allows a single process to be divided into multiple threads, which can be executed concurrently or in parallel. Each thread represents a separate unit of execution within a process, and they share the same memory space. The main advantage of multi-threading is that it enables better utilization of a system's CPU resources, allowing multiple tasks to be processed simultaneously.



3- A multi-core processor is a type of CPU that contains two or more independent processing units, called cores, within a single chip. Each core can execute instructions independently of the others, allowing the processor to handle multiple tasks simultaneously. This architecture improves performance, especially for tasks that can be parallelized.

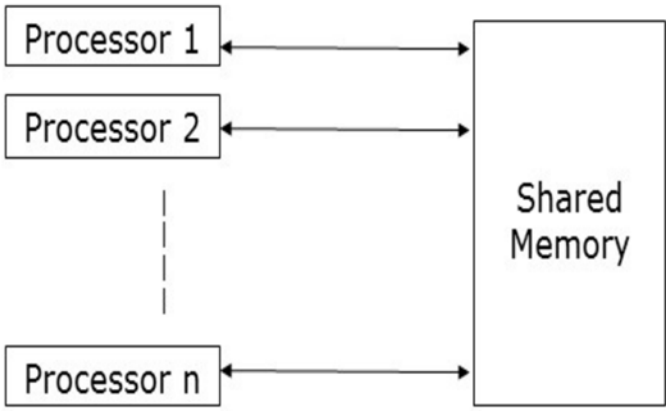


4- Multi-core with multi-threading refers to a system architecture that combines both multi-core processors and the ability to execute multiple threads per core. This combination allows for even more efficient parallel processing and multitasking, maximizing the performance of both hardware and software.

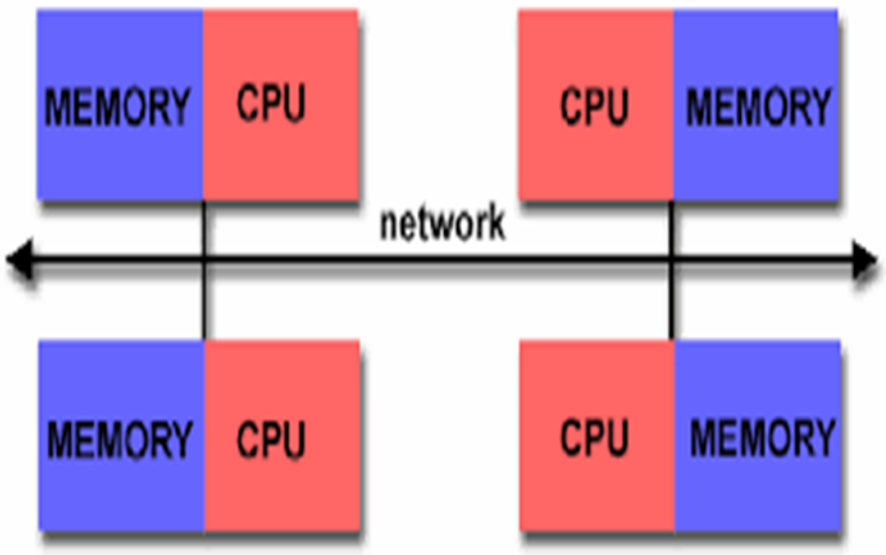


Shared memory refers to a memory architecture where multiple processing units (such as cores or nodes in a distributed system) can access a common memory space. This concept is particularly important in parallel processing systems, where large datasets need to be processed efficiently by multiple processors working in parallel.

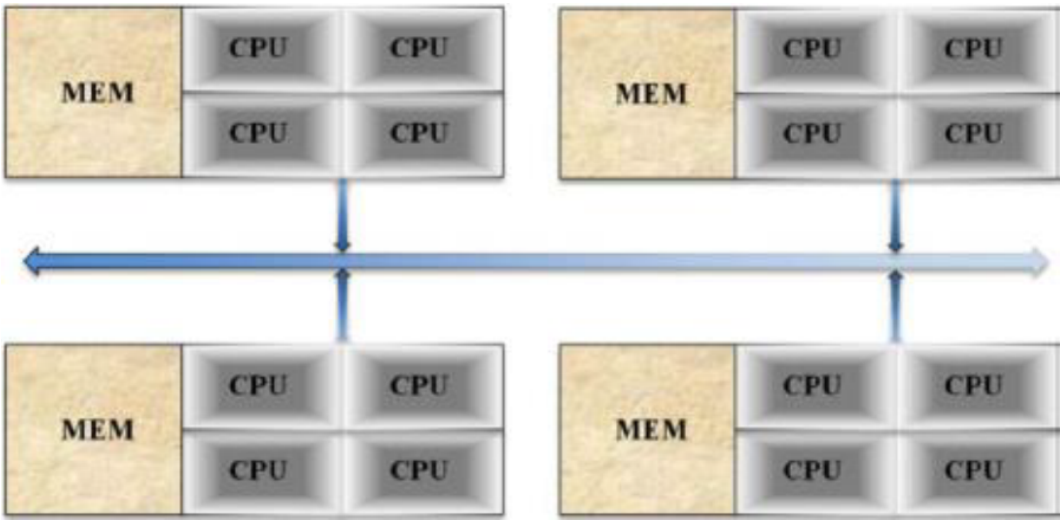
Parallel processing is becoming critical as more Internet of Things (IoT) sensors, and endpoints need real-time data. Given how easy it is to get processors and GPUs (graphics processing units), today, through cloud services, parallel processing is an essential part of implementing any new microservice.



Distributed memory parallel computers use multiple processors, each with their own memory, connected over a network. Examples of distributed systems include cloud computing, distributed file systems, online multiplayer games, etc.

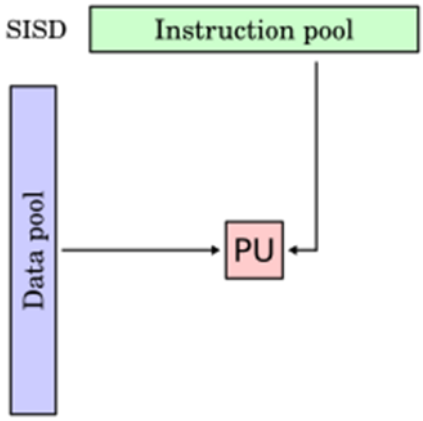


Hybrid memory parallel systems combine shared-memory parallel computers and distributed memory networks. Most “distributed memory” networks are actually hybrids. You may have thousands of desktops and laptops with multi-core processors all connected in a network and working on a massive problem.

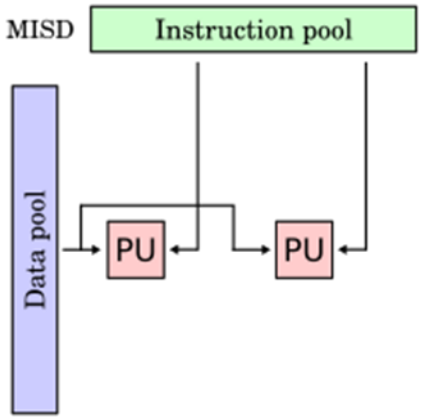


Followig are types of Parallel data processing:

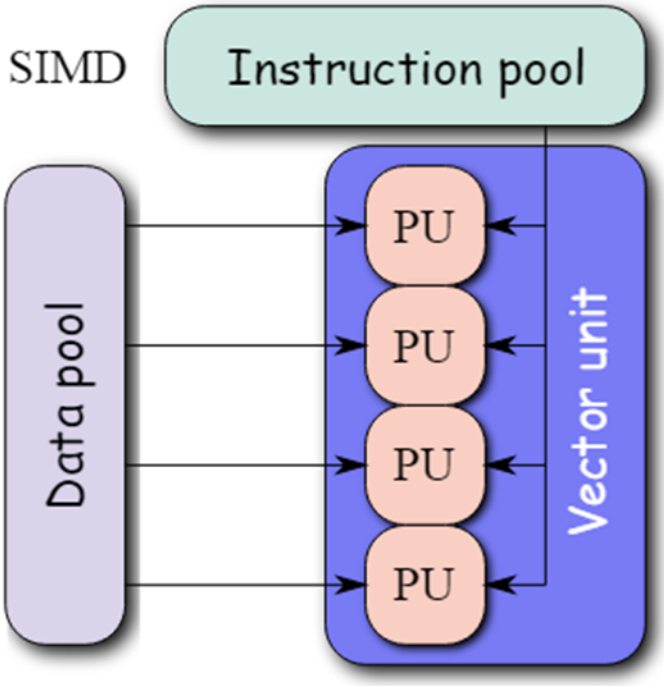
1- Single Instruction, Single Data (SISD): In SISD a single processor is responsible for simultaneously managing a single algorithm as a single data source. A computer organization having a control unit, a processing unit, and a memory unit is represented by SISD. It is similar to the current serial computer. Instructions are carried out sequentially by SISD, which may or may not be capable of parallel processing, depending on its configuration.



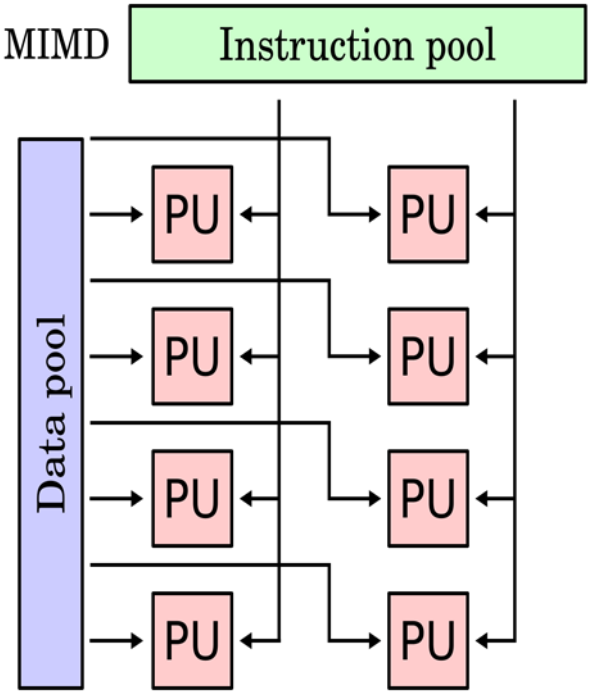
2- Multiple Instruction, Single Data (MISD) : Multiple processors are standard in computers that use the Multiple Instruction, Single Data (MISD) instruction set. While using several algorithms, all processors share the same input data. MISD computers can simultaneously perform many operations on the same batch of data. As expected, the number of operations is impacted by the number of processors available.



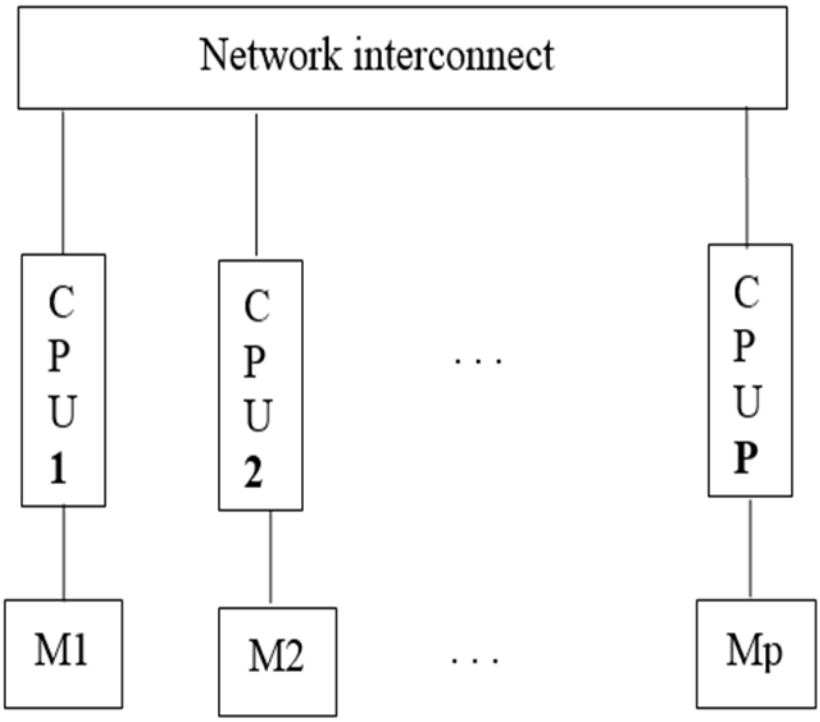
3- Single Instruction, Multiple Data (SIMD): Computers that use the Single Instruction, Multiple Data (SIMD) architecture have multiple processors that carry out identical instructions. However, each processor supplies the instructions with its unique collection of data. SIMD computers apply the same algorithm to several data sets. The SIMD architecture has numerous processing components.



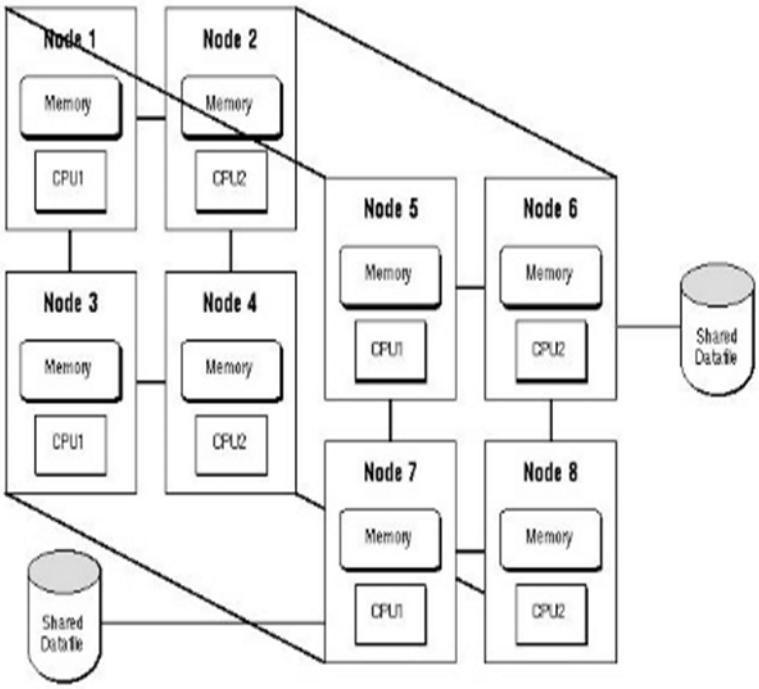
4- Multiple Instruction, Multiple Data (MIMD): Multiple Instruction, Multiple Data, or MIMD, computers are characterized by the presence of multiple processors, each capable of independently accepting its instruction stream. These kinds of computers have many processors. Additionally, each CPU draws data from a different data stream. A MIMD computer is capable of running many tasks simultaneously.



5- Single Program, Multiple Data (SPMD): SPMD systems, which stand for Single Program, Multiple Data, are a subset of MIMD. Although an SPMD computer is constructed similarly to a MIMD, each of its processors is responsible for carrying out the same instructions. SPMD is a message passing programming used in distributed memory computer systems. A group of separate computers, collectively called nodes, make up a distributed memory computer.



6- Massively Parallel Processing (MPP): A storage structure called Massively Parallel Processing (MPP) is made to manage the coordinated execution of program operations by numerous processors. With each CPU using its operating system and memory, this coordinated processing can be pplied to different program sections.



Advantages of parallel Data processing

Note

1. Parallel computing saves time, allowing the execution of applications in a shorter wall-clock time.
2. Parallel computers can be built from inexpensive components, and repair and failure do not cost the overall system.

3. Many problems (e.g., 3D resolutions, AI analysis, etc) are so large and/or complex that it is impractical or impossible to solve them on a single computer, especially given limited computer memory.

Disadvantages Of Parallel Data Processing *Note*

1. Parallel solutions are harder to create, more difficult to fix, and often perform worse than serial solutions.
2. The use of parallel computing lets you solve data-intensive problems using multicore processors, but, sometimes this effect on some of our control algorithm and does not give good results.
3. Managing tasks (like starting, transfers, synchronization, communication, thread creation/destruction, etc) can take a lot of time, especially if they are not programmed properly.
4. Better cooling technologies are required in case of clusters.
5. Power consumption is huge by the multi-core architectures.

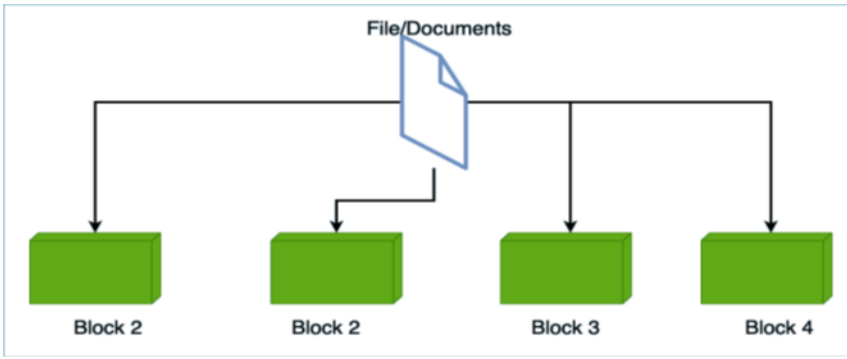
2. Distributed data processing

What It Means

Instead of processing all data on a single computer, the work is divided among several machines (called nodes) connected in a network. Each node processes its part of the data and sends the results back to a central system or combines them for the final output.

Why It's Useful

- Speeds up processing for large datasets.
- Handles more data than a single machine can manage.
- Improves fault tolerance: if one machine fails, others can continue working.



Big data processing use cases *Note*

- Real-time vehicle tracking; traffic management; geofencing.
- Medical IoT.
- Credit card fraud/account takeover detection.
- Real-time stock market quotes management.
- Automated real-time anomaly recognition for manufacturing/oil&gas industries.
- Connected smart appliances.
- Online video games.

- XaaS

Challenges of Distributed Data Processing *Note*

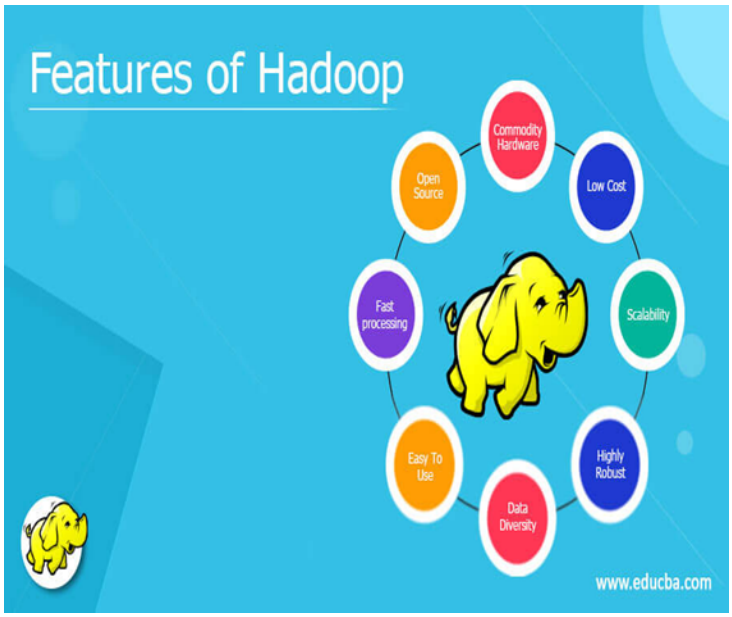
- **Data Distribution and Partitioning**: Deciding how to divide the data across nodes to ensure balanced workload and minimize data transfer between nodes.
- **Data Consistency**: Maintaining data consistency across distributed systems, especially in the face of updates or system failures.
- **Fault Tolerance**: Ensuring the system continues to function despite hardware or software failures.
- **Network Latency and Bandwidth**: High latency and limited bandwidth can hinder performance when transferring data between nodes.
- **Scalability**: Choose scalable frameworks and ensure the architecture supports horizontal scaling.
- **Data Security and Privacy**: Protecting sensitive by the encryption, access control, and secure communication protocols mechanisms.
- **Complexity in Development**: Developing distributed applications is more complex due to the need for synchronization, debugging, and coordination.

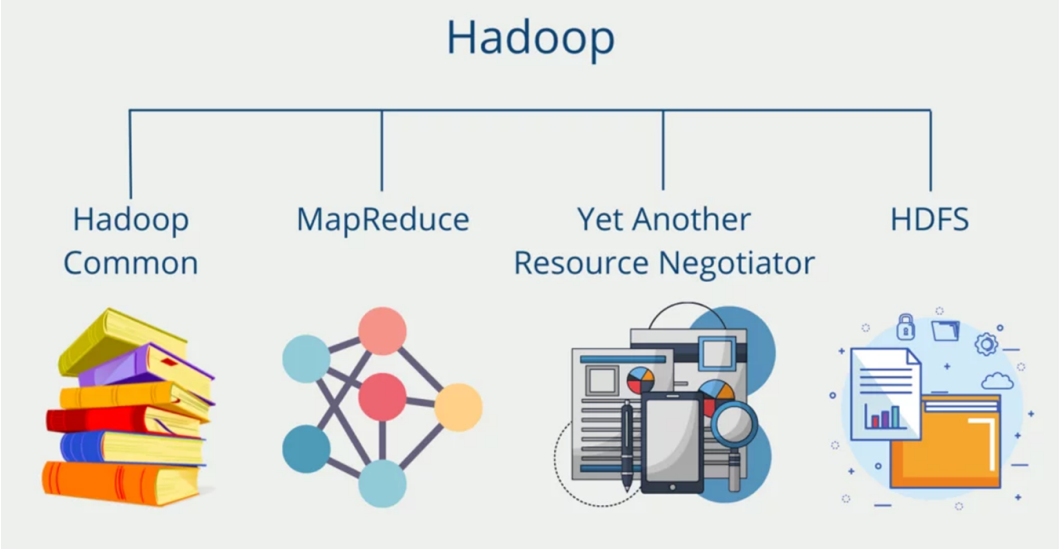
3. Hadoop

Definition

Hadoop is an open source software framework for storage and processing large scale of datasets on clusters of hardware.

Hadoop provides a reliable shared storage and analysis system, here storage provided by HDFS and analysis provided by MapReduce.





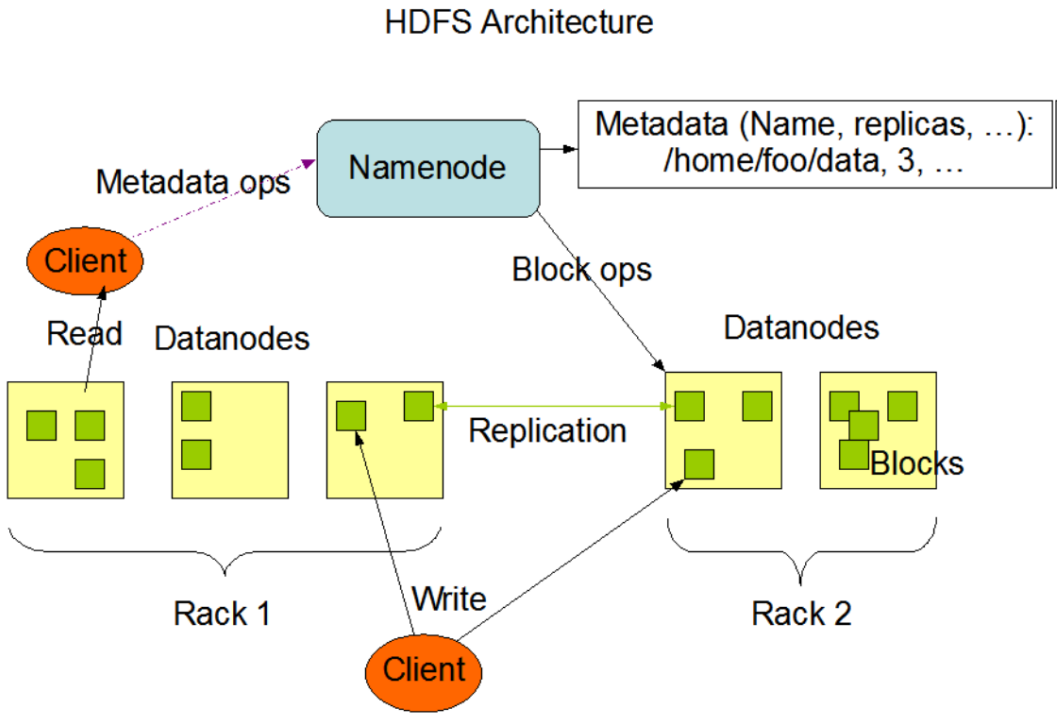
1- HADOOP DISTRIBUTED FILE SYSTEM (HDFS):

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on hardware.

It has many similarities with existing distributed file systems.

HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware.

HDFS provides high throughput access to application data and is suitable for applications that have large data sets.



NameNode and DataNodes

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system name space and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system name space operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. TheDataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation,deletion, and replication upon instruction from the NameNode.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linuxoperating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNodesoftware. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment hasa dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNodesoftware. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case.

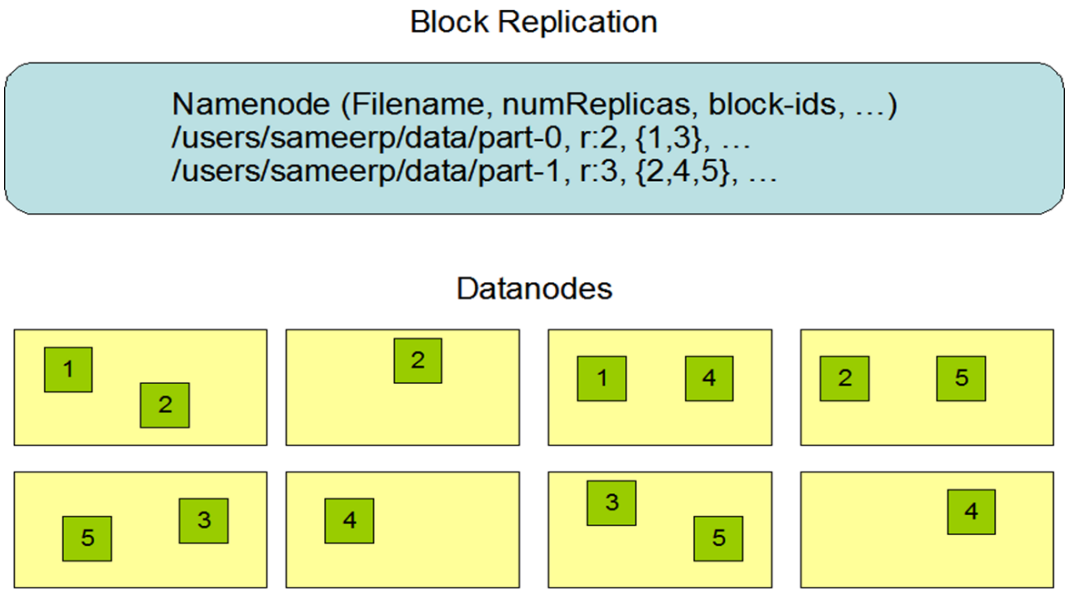
The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator andrepository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode.

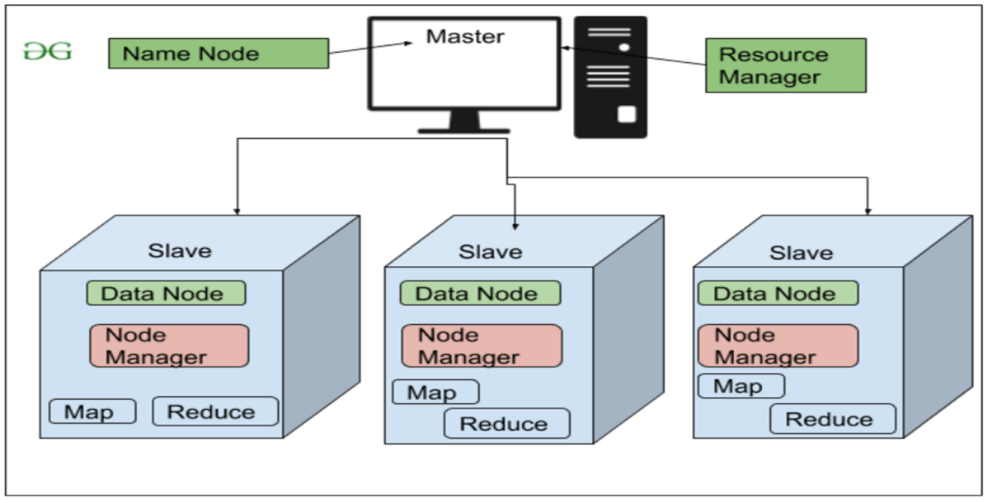
The File System Namespace

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories.The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from onedirectory to another, or rename a file. HDFS supports user quotas and access permissions. HDFS does not support hard links or soft links.However, the HDFS architecture does not preclude implementing these features.

While HDFS follows naming convention of the FileSystem, some paths and names (e.g. /.reserved and .snapshot) are reserved. Features suchas transparent encryption and snapshot use reserved paths.

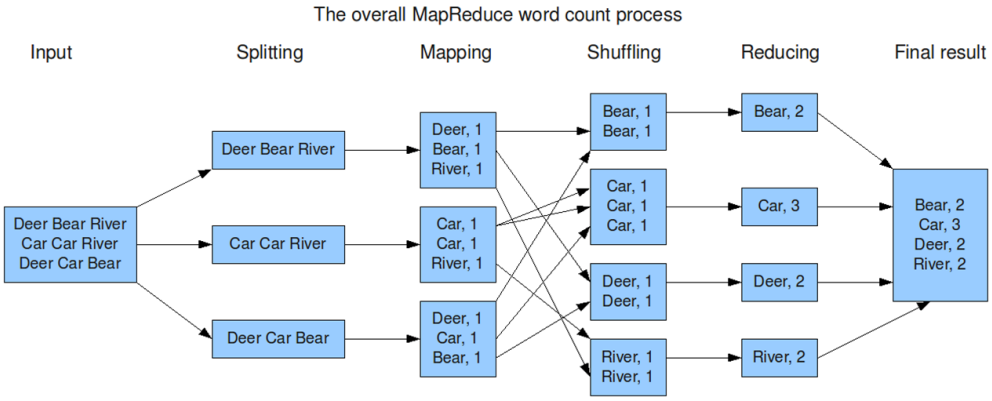
The NameNode maintains the file system namespace. Any change to the file system namespace or its properties is recorded by the NameNode.An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called thereplication factor of that file. This information is stored by the NameNode.





2- MapReduce:

It origins from Google and helps to divide complex computing tasks into more manageable subprocesses and then distributes these across several systems, i.e. scale them horizontally. This significantly reduces computing time. In the end, the results of the subtasks have to be combined again into the overall result.



Input: The system takes a large dataset (e.g., a file or database) as input for processing.

Splitting: The dataset is divided into smaller chunks (splits) to be processed in parallel across multiple nodes.

Mapping: Each chunk is processed to generate key-value pairs, where the key organizes the data, and the value provides relevant information.

Shuffling: Key-value pairs are redistributed across nodes so that all pairs with the same key are sent to the same node. This step ensures data is grouped logically for the next stage.

Reducing: Each reducer processes the grouped data (by key), performing operations like summing, counting, or aggregating to generate a result.

Final Result: The reduced outputs from all nodes are combined into the final output dataset, ready for use or analysis.

1. Map: Mappers in containers execute the task using the data block in slave nodes. This is a part of the actual data manipulation for the job requested by the client. All mappers in the containers execute the tasks in parallel. The performance of the mapper depends on scheduling, data locality, programmer skills, container's resources, data size and data complexity.

2. Sort/Spill: The output pair which is emitted by the mapper is called partition. The partition is stored and sorted in the key/value buffer in the memory to process the batch job. The size of the buffer is configured by resource tracker and when its limit is reached, the spill is started
3. Shuffle: The key/value pairs in the spilled partition are sent to the reduce nodes based on the key via the network in this step. To increase the network performance.
4. Merge: The partitions in the partition set are merged to finish the job. This step has usually been studied along with the shuffle step, such as in-memory with compression.
5. Reduce: The slave nodes process the merged partition set to make a result of the application. The performance of reduce depends on scheduling, locality, programmer skills, container resources, data size, and data complexity, as was the case in the map step. However, unlike in the map step, the reduce step can be improved by in-memory computing.
6. Output: The output of reduce nodes is stored at HDFS on the slave nodes.

3- Hadoop YARN

YARN (Yet Another Resource Negotiator) is a core component of Hadoop. Its main job is to manage resources and coordinate the execution of applications across the system. It ensures that all programs get the CPU and memory they need without conflicts.

Roles in Hadoop

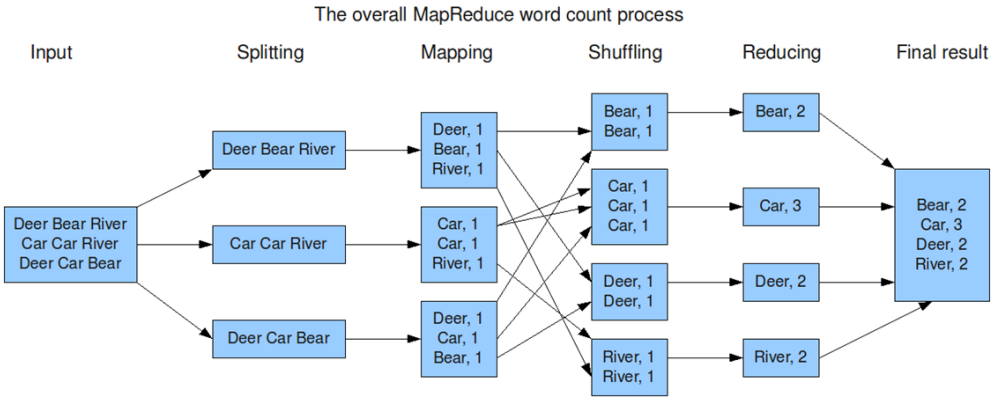
- **Resource Management:** *Example:* In MapReduce, the appropriate nodes are identified while ensuring balance in the system.
- **Application Execution:** *Example:* Running a MapReduce program and monitoring it until it finishes.
- **Fault Tolerance:** *Example:* YARN monitors nodes and restarts failed tasks.

4- Hadoop Common:

Hadoop Common is the foundation of the Hadoop ecosystem. It provides the essential tools, libraries, and utilities required by other Hadoop components (like HDFS and MapReduce) to function properly.

Key Roles

- **Provides Shared Libraries:** *Example*: Libraries for file handling, logging, or network communication.
- **Manages Configuration:** *Example*: Setting the block size in HDFS or specifying the replication factor.
- **Enables Communication Between Components:** *Example*: Ensuring that a MapReduce job can access data stored in HDFS.
- **Provides Utility Tools:** *Example*: Decompressing large files before processing them.



4. Workload processing

Workload

Definition

A workload represents the amount of computational power or resources required for a specific task, such as the processing power, memory, storage, or network capacity needed to run applications or processes. It can also refer to the distribution of tasks across multiple systems, especially in cloud computing or distributed environments.

Types of Workloads

Note

1- CPU-Intensive Workloads

These workloads rely heavily on the processing power of the CPU. They involve computations, logic operations, or data processing tasks that demand significant CPU cycles.

Examples:

- Scientific computations (e.g., simulations, mathematical modeling).
- Data analytics and AI/ML model training.
- Gaming engines and 3D rendering.
- Cryptographic tasks like encryption and hashing.

2- I/O-Intensive Workloads

These workloads are limited by input/output operations such as reading/writing to disk or network communication. They typically require a lot of data movement.

***Examples*:**

- Database transactions (e.g., SQL queries).
- Web servers and file transfers.
- Data backup and restoration.
- Streaming and downloading large files.

3- Memory-Intensive Workloads

These workloads require large amounts of memory (RAM) to store and manipulate data temporarily during processing.

Examples:

- In-memory databases (e.g., Redis, Memcached).
- Big data processing frameworks (e.g., Apache Spark).
- Real-time analytics and caching mechanisms.
- High-resolution image or video editing.

4- Network-Intensive Workloads

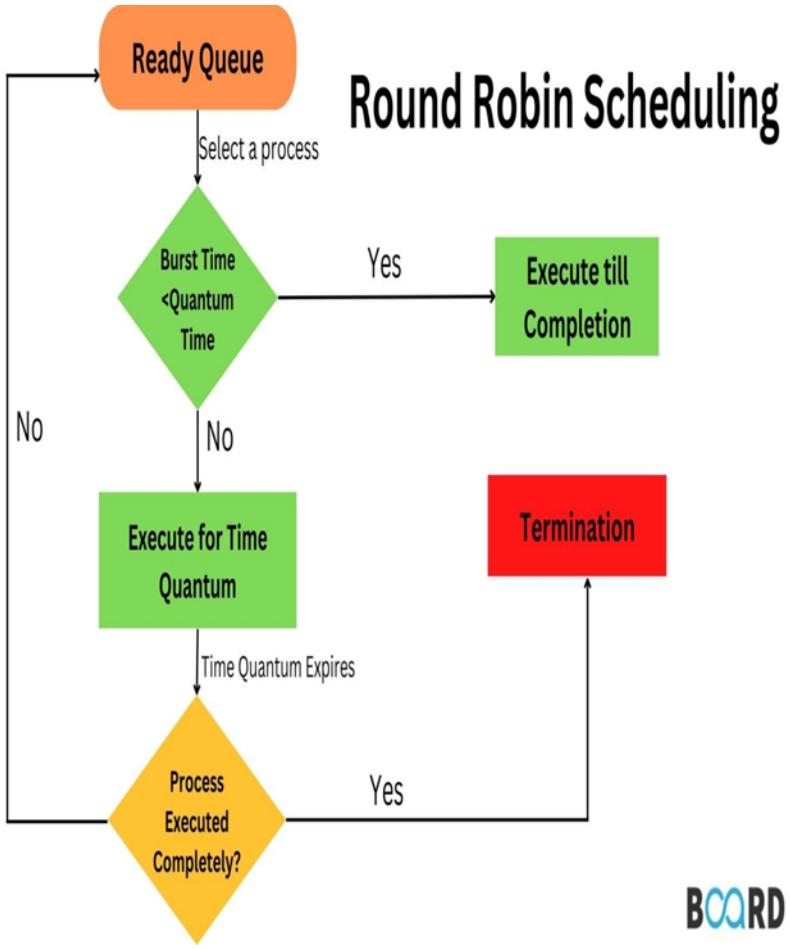
These workloads depend on high-speed network communication and require large bandwidth for data transmission.

***Examples*:**

- Video conferencing and live streaming.
- Online gaming.
- Distributed applications and microservices.
- IoT systems with high-frequency data transfers.

Workload Distribution Techniques

The Round-Robin (RR) algorithm is a simple and widely used scheduling technique in computer systems. It allocates resources or executes processes in a cyclic order, ensuring that each task gets an equal and fair share of the resource for a specific time slice, called a quantum (or time slice).



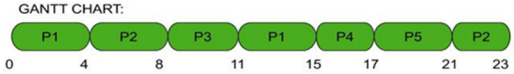
TIME SLICE = 4

PROCESS	ARRIVAL TIME	TOTAL TIME
P1	0	8
P2	1	6
P3	3	3
P4	5	2
P5	6	4

READY QUEUE:

P1	P2	P3	P1	P4	P5	P2

GANTT CHART:



Waiting time?

Execution time?

Uses of RR in reality

- CPU Scheduling: Time-sharing systems to distribute CPU time among processes.
- Network Packet Scheduling: Handling data packets in routers.
- Resource Allocation in Virtualized Environments: Allocating CPU or network resources among virtual machines.
-
- **Advantages**
- Fair Resource Allocation: All processes are treated equally.
- Simplicity: Easy to implement and understand.
- Prevents Starvation: Each process gets a chance to execute.

Disadvantages

- Context Switching Overhead:Frequent process switches require the system to save and load the state of each process, which uses extra time and resources, reducing efficiency.
- Performance Dependent on Quantum:
- If the quantum is too small, overhead increases.
- If the quantum is too large, response time for short tasks may increase.

Max-Min Fairness is a resource allocation principle used to ensure a fair distribution of resources among users or processes. It aims to maximize the allocation for the process or user with the minimum resource share, thereby promoting equity while avoiding resource starvation.

Applications

- Network Bandwidth Allocation: Distributing network bandwidth among multiple users or devices.
- CPU Scheduling: Allocating processor time among competing processes.
- Cloud Computing: Fairly distributing computational or storage resources among tenants.

	26 cpu			
	User1	User2	User3	User4
Needs(Ascending order)	3 cpu	6 cpu	9 cpu	12 cpu
	26/4 = 6 cpu			
Receiving	3 cpu (fully satisfied)	6 cpu (fully satisfied)	6 CPUs (Remaining 3 cpu)	6 CPUs (Remaining 6 cpu)
	Remaining resources: 26–21=5, 5 cpu/2 users=2			
Receiving	3 cpu (fully satisfied)	6 cpu (fully satisfied)	8 cpu (Remaining 1 cpu)	8 cpu (Remaining 4 cpu)
	Remaining resources: 26–25=1			
Receiving	3 cpu (fully satisfied)	6 cpu (fully satisfied)	9 cpu (fully satisfied)	8 cpu (Remaining 4 cpu)

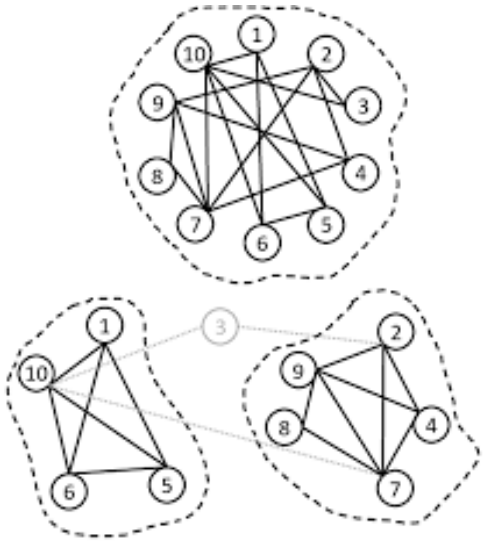
5. Cluster

Definition

A **cluster** is a group of computers (called nodes or servers) that work together like one big system.

- Each node is a separate computer with its own CPU, memory, and storage.
- All the nodes are connected by a network so they can share tasks and data.
- If one node fails, the others can keep the system running.

- Clusters are used to handle big workloads, store large amounts of data, and keep services running without interruption.



Extra

Why use a cluster?

- To make work faster (tasks are divided between nodes).
- To avoid downtime (if one node stops, others keep working).
- To handle more users and more data at the same time.

Types of clusters:

1. High Performance Cluster (HPC) – focuses on speed for heavy calculations.
2. High Availability Cluster (HA) – focuses on keeping services always online.
3. Load Balancing Cluster – spreads requests evenly so no node gets overloaded.

How it works:

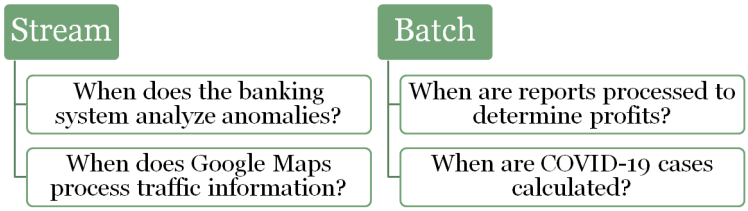
- Tasks are sent to different nodes.
- Nodes talk to each other over a fast network.
- A software layer manages which node does what.

Examples in real life:

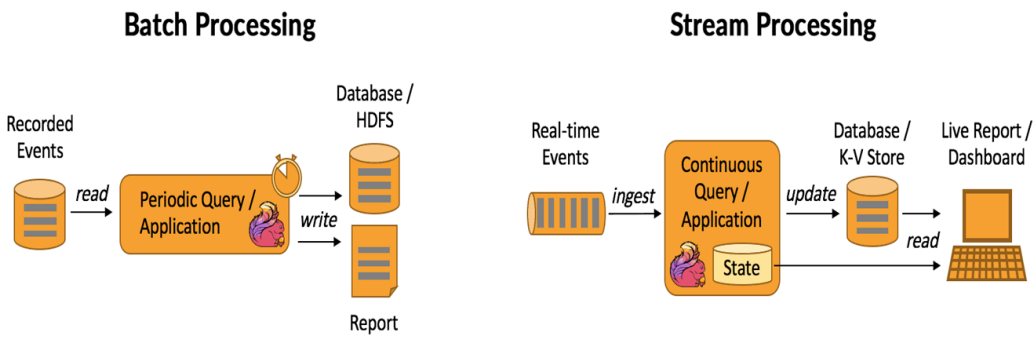
- Google search runs on clusters of thousands of servers.
- Banks use clusters so their services never stop.
- Scientific research uses clusters to process huge data sets.

6. Stream and Batch processing

In data processing, stream processing and batch processing represent two fundamental approaches to handling and analyzing data. Both methods are essential for modern systems, but they are tailored for different scenarios and types of workloads. Understanding their differences and use cases is crucial for designing efficient and scalable systems.

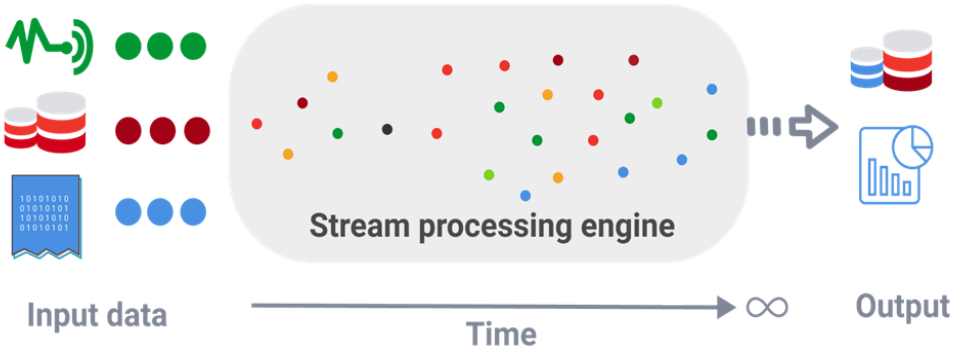


Type of processing *Note*



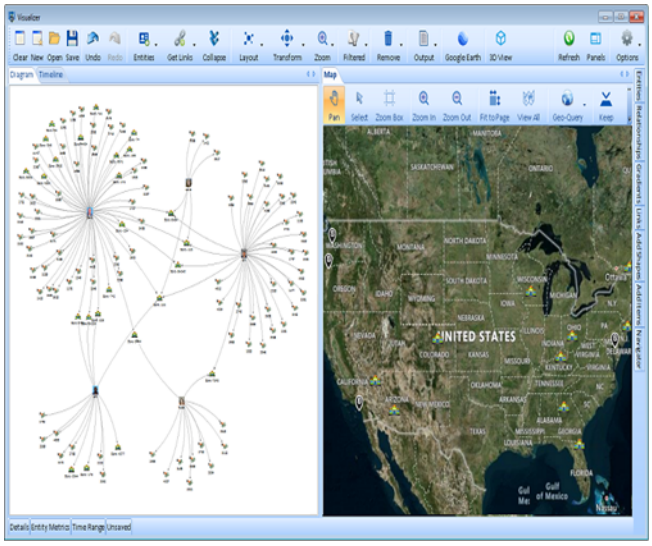
Stream Processing

Stream (real-time) processing is a method of continuously analyzing and processing data as it is generated in real-time. Instead of waiting for all the data to be collected, it processes data events one by one or in small batches, making it ideal for scenarios that require immediate insights or actions.



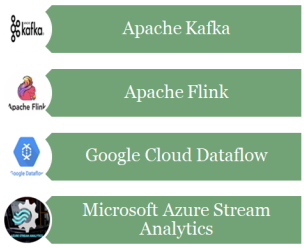
Examples:

1. **Fraud Detection in Banking:** Identifying unusual patterns in transactions, such as large withdrawals or suspicious account activities, as they occur.
2. **Traffic Monitoring (e.g., Google Maps):** Analyzing live traffic data from sensors and user devices to provide real-time updates and suggest alternative routes.
3. **IoT Systems:** Monitoring and reacting to sensor data in smart homes or factories, such as turning off a machine if overheating is detected.
4. **Healthcare:** Real-time surveillance sends immediate alerts for critical patient conditions



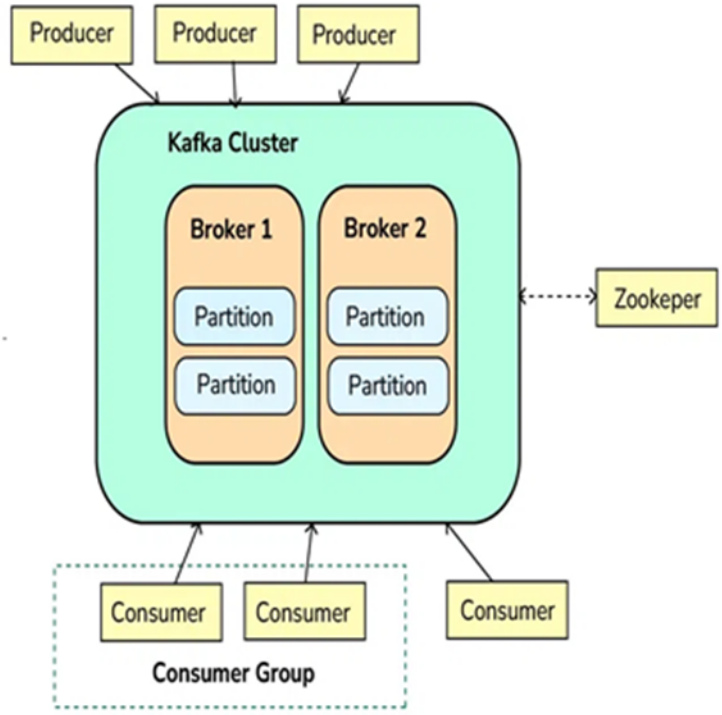
Technologies of Stream Processing

Extra



Kafka

- Apache Kafka is an open-source distributed event streaming platform used to handle high-throughput, low-latency, real-time data streams. It is designed to handle large volumes of data efficiently and allows systems to publish, subscribe, store, and process streams of records (events or messages) in a fault-tolerant manner.
- Kafka is often used for building real-time data pipelines, streaming analytics, and event-driven architectures. It can handle a variety of use cases, such as log aggregation, real-time analytics, data replication, and more.



Step 1: Producers Send Messages (Events)

- Producers are applications or services that send data (events/messages) to Kafka.
- Producers push data into topics in Kafka, which are logical channels to organize messages. A producer can publish messages to one or more topics.

Step 2: Kafka Brokers Store Messages

- Kafka brokers are the servers that receive, store, and manage messages.
- Kafka topics are divided into partitions, which are distributed across different brokers to ensure scalability and fault tolerance.
- Each partition stores messages in a sequential order, and messages within a partition are given an offset, which is a unique identifier.

Step 3: Kafka Consumers Read Messages

- Consumers subscribe to topics or specific partitions to consume the messages.
- Consumers read the messages in the order they were written (based on the offset). A consumer can either read from the latest message or from an earlier offset to reprocess the data.
- Kafka allows multiple consumers to work in parallel, scaling horizontally.

Step 4: Kafka Zookeeper Coordinates Kafka Cluster

- Kafka uses Zookeeper (or KRaft mode in newer versions) to manage and coordinate the Kafka cluster.
- Zookeeper helps track the metadata of topics, partitions, and offsets, and ensures high availability and fault tolerance of Kafka brokers.

Step 5: Kafka Handles Fault Tolerance and Replication

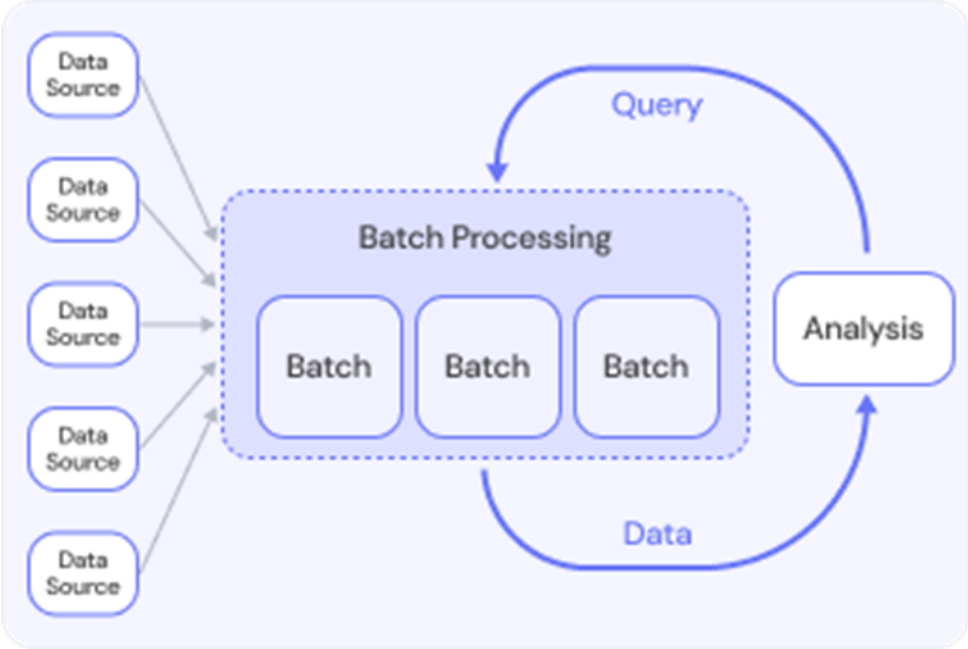
- Kafka provides replication to ensure that the data is not lost. Each partition of a topic has multiple replicas stored on different brokers.
- If a broker fails, Kafka will automatically recover and continue functioning by redirecting to replicas, ensuring data durability and fault tolerance.

Step 6: Kafka Provides Real-Time Processing

- Kafka is commonly integrated with stream processing frameworks like Kafka Streams or external systems like Apache Flink or Apache Spark for real-time data processing.
- The system processes data in real-time, allowing for quick insights and reactions to incoming data.

Batch processing

- Batch processing refers to the execution of a series of data processing tasks in a group or "batch," rather than processing data in real-time. In batch processing, data is collected over a period of time, stored, and then processed all at once. This method is typically used for tasks that do not require immediate feedback or action, and the processing is done at scheduled intervals (e.g., daily, weekly, or monthly).
- Batch processing is often used when dealing with large volumes of data, where real-time processing isn't necessary, and the system can offer a delay in getting results.



Exampels:

- **Lab Test Results:** When patients receive multiple lab tests, results may be processed in batches at regular intervals. Instead of notifying patients individually as results come in, the system sends batch notifications to patients when all their test results are ready.
- **Payroll Processing:** A company has to pay employees every month. Instead of calculating and paying each employee individually, the payroll department processes all employee salaries in a batch, usually at the end of the month.
- **Social Media Post Scheduling:** A social media manager schedules posts for a week. Instead of posting content in real time, they prepare and schedule all posts in a batch at once, to be published at specific times.
- **Telecommunications Billing:** A telecom company processes call data for its users. Instead of processing each call immediately, the system batches all calls and processes them in bulk for billing at the end of the month.

Technologies of Batch Processing

Extra