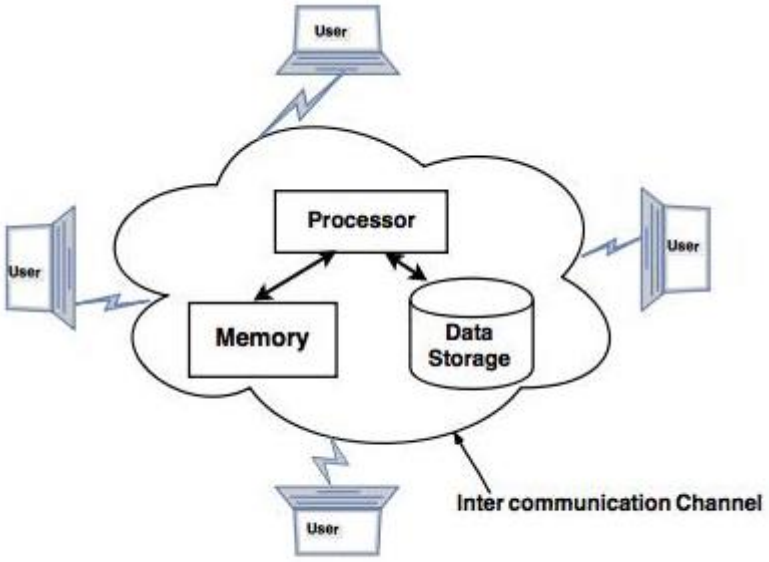
# Module 4

**Parallel Database:**

- A **parallel database** is one which involves multiple processors and working in parallel on the database used to provide the services.
- A parallel database system seeks to improve performance through parallelization of various operations like loading data, building index and evaluating queries parallel systems improve processing and I/O speeds by using multiple CPU's and disks in parallel.




Parallel database system

**Working of parallel database:**

Let us discuss how parallel database works in step-by-step manner −

**Step 1** − Parallel processing divides a large task into many smaller tasks and executes the smaller tasks concurrently on several CPU's and completes it more quickly.

**Step 2** − The driving force behind parallel database systems is the demand of applications that have to query extremely large databases of the order of terabytes or that have to process a large number of transactions per second.

**Step 3** − In parallel processing, many operations are performed simultaneously as opposed to serial processing, in which the computational steps are performed sequentially.

**Features of Parallel Database:**

1. **Performance Improvement –**
   By connecting multiple resources like CPU and disks in parallel we can significantly increase the performance of the system.

2. **High availability –**
   In the parallel database, nodes have less contact with each other, so the failure of one node doesn't cause for failure of the entire system. This amounts to significantly higher database availability.

3. **Proper resource utilization –**
   Due to parallel execution, the CPU will never be idle. Thus, proper utilization of resources is there.

4. **Increase Reliability –**
   When one site fails, the execution can continue with another available site which is having a copy of data. Making the system more reliable.

**Performance Measurement of Databases:**
Here, we will emphasize the performance measurement factor-like Speedup and Scale-up. Let's understand it one by one with the help of examples.

**Speedup –**
The ability to execute the tasks in less time by increasing the number of resources is called Speedup.

**Speedup=time original/time parallel**
Where ,
time original = time required to execute the task using 1 processor
time parallel = time required to execute the task using 'n' processors

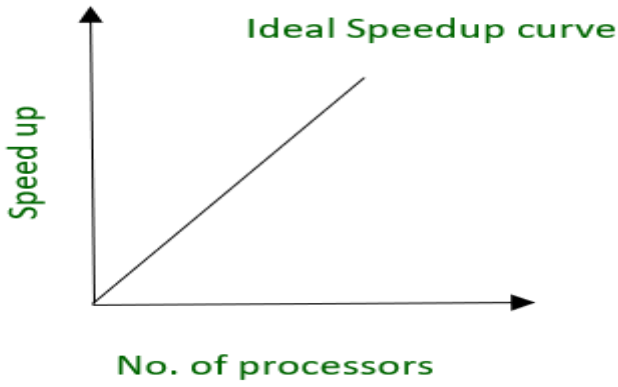



*fig. Ideal Speedup curve*

**Example –**

*fig. A CPU requires 3 minutes to execute a process*

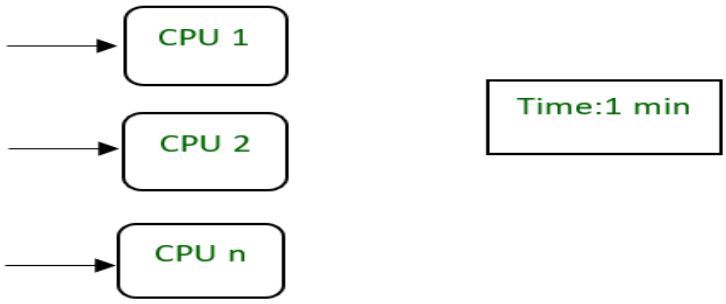



*fig. 'n' CPU requires 1 min to execute a process by dividing into smaller tasks*

**Scale-up** –
The ability to maintain the performance of the system when both workload and resources increase proportionally.

**Scaleup = Volume Parallel/Volume Original**
Where,
Volume Parallel = volume executed in a given amount of time using 'n' processor
Volume Original = volume executed in a given amount of time using 1 processor

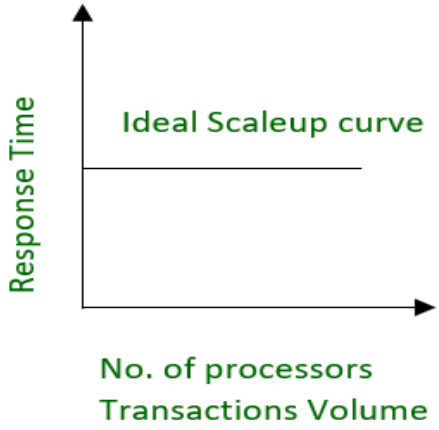
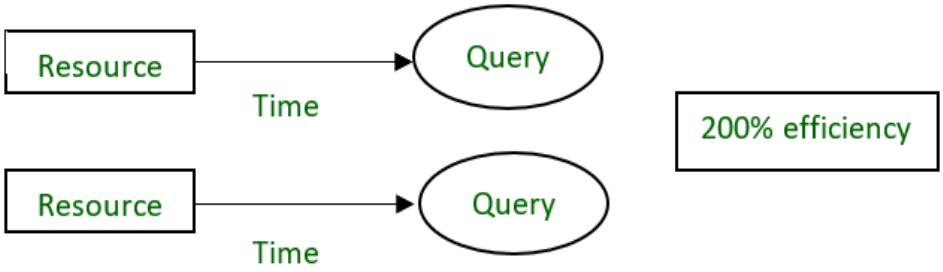



*fig. Ideal Scaleup curve*

**Example** –
20 users are using a CPU at 100% efficiency. If we try to add more users, then it's not possible for a single processor to handle additional users. A new processor can be added to serve the users parallel. And will provide 200% efficiency.



## Shared Memory:

Shared memory is an operating-system feature that allows the database server threads and processes to share data by sharing access to pools of memory. The database server uses shared memory for the following purposes: To reduce memory usage and disk I/O. To perform high-speed communication between processes.

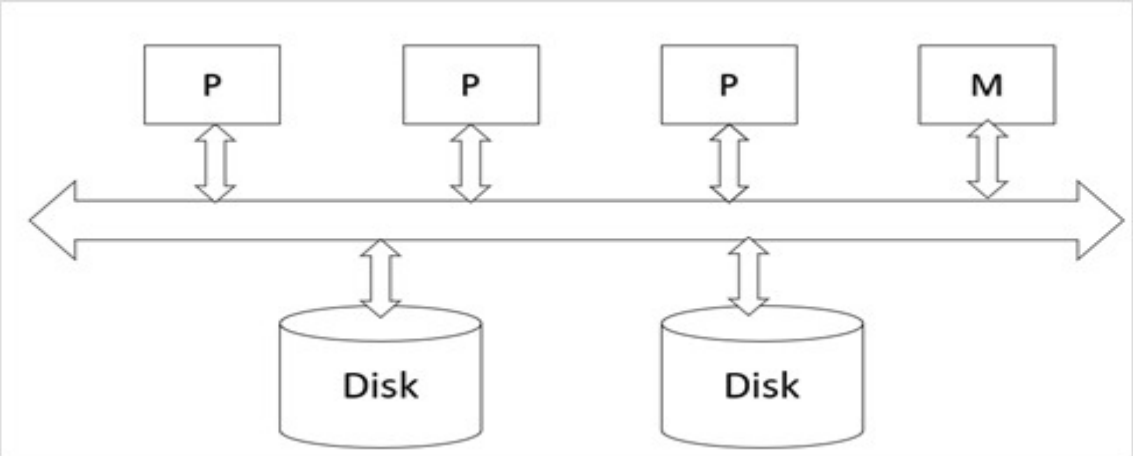
**Architectural Models:**

There are several architectural models for parallel machines, which are given below −

- Shared memory architecture.
- Shared disk architecture.
- Shared nothing architecture.

Shared Memory Architecture

**Shared-memory multiple CPU** − In this a computer that has several simultaneously active CPU attached to an interconnection network and share a single main memory and a common array of disk storage. This architecture is attractive for achieving moderate parallelism because a limited number of CPU's can be exploited.
**Examples** − DBMS on symmetric multiprocessor, sequent, sun.
The architecture of shared memory multiple CPU's is shown below –




Here,
P is the Processor.
M is the Memory.

Advantages

The advantages of the shared memory architecture are as follows −

- Efficient communication: communication overheads are low since data in shared memory can be accessed by any processor without being moved with software.
- CPU to CPU communication is very efficient because a CPU can send data to another CPU with the speed of memory write.
- Shared memory architectures usually have large memory caches at each processor, so that referencing of the shared memory is avoided whenever possible.
- Suitable to achieve moderate parallelism.

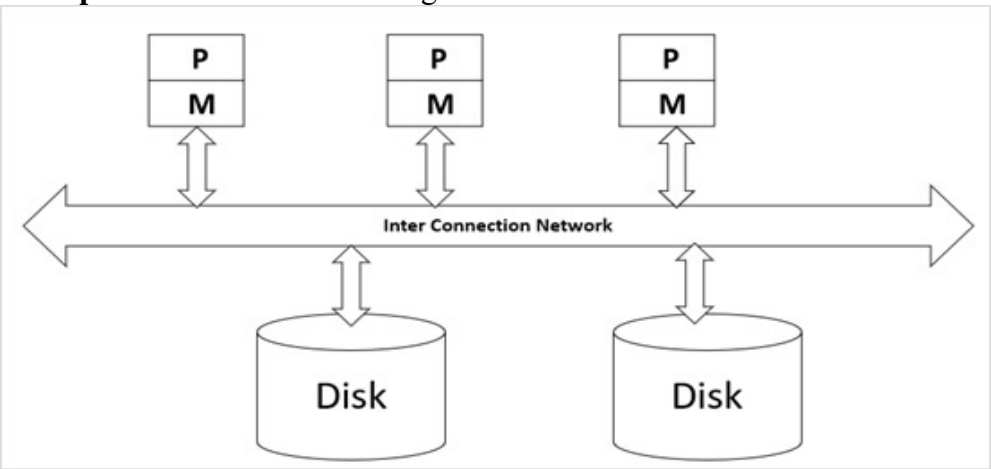
Disadvantages

The disadvantages of the shared memory architecture are as follows −

- The architecture is not scalable beyond 32 or 64 CPU because the processor will spend most of their time waiting for their turn on the bus to access memory.
- Existing CPU's get slowed down, due to bus contention or network bandwidth.
- Maintaining cache memory becomes an increasing overhead with an increasing number of processors.

Shared Disk Architecture

**Shared disk architecture** − In this each node has its own main memory but all nodes share mass storage. In practice, each node also has multiple processors. All processors can access all disks directly via an interconnection network, each processor has their private memories.
**Examples** − IMS/VS data sharing.



Here,
P is the Processor.
M is the Main Memory

Advantages

The advantages of the shared disk architecture are as follows −

- As each processor has its own memory thus the memory bus is not a bottleneck.
- It offers a degree of fault tolerance that is if a processor fails the other processors can take over its task because the database is resident on disks that are accessible from all processors.
- It can scale to a somewhat large number of CPU's.

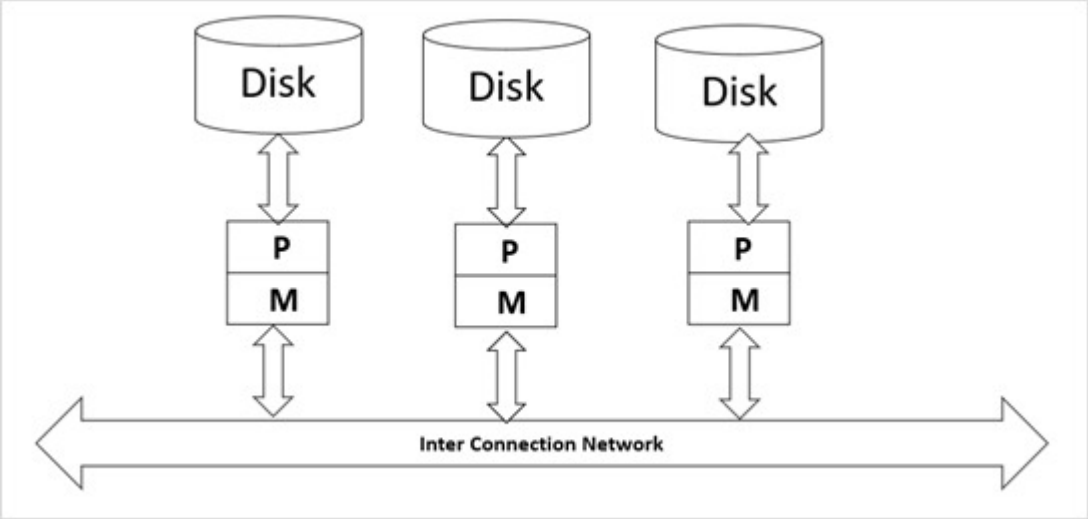
Disadvantages

The disadvantages of the shared disk architecture are as follows −

- The architecture is not scalable, since the interconnection to the disk subsystem is now having the bottleneck.
- CPU to CPU communication is slower since it has to go through a communication network.
- The basic problem with the shared disk model is interface because more CPUs are added. Existing CPU's are slowed down because of increased contention for memory access in network bandwidth.

Shared nothing Architecture

**Shared nothing architecture** − In this each node has its own mass storage as well as main memory. The processor at one node may communicate with another processor at another node by a high Speed interconnection network. The node functions as the server for the data on the disk or disks that the node owns as each processor has its own copy of OS, dbms and data.
**Examples** − Teradata, Gamma, Bubba.
Given below is the diagram of the shared nothing architecture −




Advantages

The advantages of the shared nothing architecture are as follows −

- The interconnection network for shared-nothing models is usually designed to be scalable so that their transmission capacity increases as more nodes are added. Thus these architectures are more scalable and easily support a large number of processors.
- It overcomes the disadvantages requiring all I/O to go through a single intercommunication network.
- It provides linear speed-up and linear scale-up that is time taken for operations decreases in proportion to the increase in the number of CPU's and disks; scale-up means the performance is sustained if the number of CPU's and disks are increased in proportion to the amount of data.

Disadvantages

The disadvantages of the shared nothing architecture are as follows −

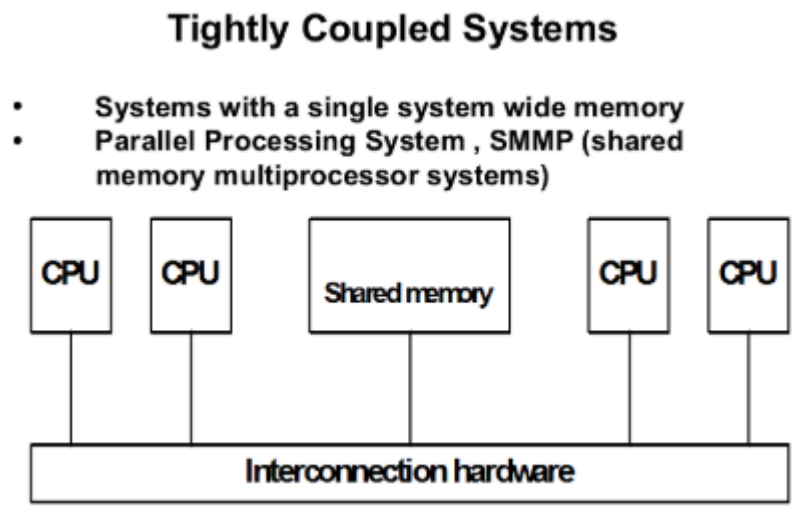
- CPU to CPU communication is very slow.
- The cost of communication and no-local disk access are higher than shared memory or shared disk architecture because sending data involves software interaction at both sides.
- Shared nothing architecture is difficult to load balance.

**Differences between Parallel and Distributed Databases:**

| Parallel Database | Distributed Database |
|---|---|
| In parallel databases, processes are tightly coupled and constitutes a single database system i.e., the parallel database is a centralized database and data reside in a single location | In distributed databases, the sites are loosely coupled and share no physical components i.e., distributed database is our geographically departed, and data are distributed at several locations. |
| In parallel databases, query processing and transaction is complicated. | In distributed databases, query processing and transaction is more complicated. |
| In parallel databases, it's not applicable. | In distributed databases, a local and global transaction can be transformed into distributed database systems |
| In parallel databases, the data is partitioned among various disks so that it can be retrieved faster. | In distributed databases, each site preserve a local database system for faster processing due to the slow interconnection between sites |
| In parallel databases, there are 3 types of architecture: shared memory, shared disk, and shared shared-nothing. | Distributed databases are generally a kind of shared-nothing architecture |
| In parallel databases, query optimization is more complicated. | In distributed databases, query optimization techniques may be different at different sites and are easy to maintain |
| In parallel databases, data is generally not copied. | In distributed databases, data is replicated at any number of sites to improve the performance of systems |
| Parallel databases are generally homogeneous in nature | Distributed databases may be homogeneous or heterogeneous in nature. |
| Skew is the major issue with the increasing degree of parallelism in parallel databases. | Blocking due to site failure and transparency are the major issues in distributed databases. |

**Tightly Coupled:**

In parallel database systems, "tightly coupled" refers to an architecture where multiple processors (CPUs) share a single, common memory space and I/O devices. This means all processors can access the same data and instructions simultaneously, leading to faster performance and scalability.




**Major components of parallel databases:**

- **Architectures:**
  - **Shared Memory:** Multiple CPUs share the same main memory and can access data simultaneously.
  - **Shared Disk:** Processors have their own memory but share a common storage system.
  - **Shared Nothing:** Each processor has its own memory and storage, and communication is achieved through network messages.
- **Query Parallelism:**
  - **Interquery Parallelism:** Multiple queries are processed concurrently, each running independently.
  - **Intraquery Parallelism:** A single query is broken down into parts that can be processed simultaneously.
- **Data Partitioning:**

  Dividing data across multiple processors to enable parallel processing.
- **Concurrency Control:**

  Managing concurrent access to data to ensure data integrity and consistency.
- **Interconnection Network:**

  Facilitates communication between processors in shared-memory and shared-disk architectures.
- **Synchronization Mechanisms:**

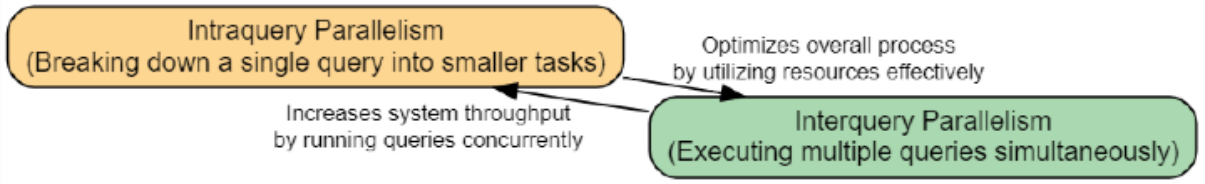
Coordinating the activities of different processors to avoid conflicts and ensure data consistency.

- **Locking Mechanisms:**
  Controlling access to shared resources to prevent conflicts and maintain data integrity.

---

**Types of Parallelism in Query Execution in DBMS/ how parallelism is introduced in query processing?**

There are two types of parallelism in query execution in DBMS: intraquery parallelism and interquery parallelism.




**Intraquery Parallelism**

Intraquery parallelism in Database Management Systems (DBMS) is an efficient and powerful technique that breaks down a single query into multiple smaller tasks, which are then executed simultaneously.

This method significantly boosts the speed of query execution by distributing processing power across different CPUs or workers within a database system.

Let's consider a simple example for clarity's sake- a voluminous database requiring sorting data based on certain parameters. Instead of executing this as one big task (which could be time-consuming), intraquery parallelism divides it into manageable sub-tasks implemented concurrently.

The results from these individual operations are then combined to produce the final output, effectively speeding up the overall process while utilizing resources optimally.

**Interquery Parallelism**

Interquery parallelism is a type of parallelism in query execution in DBMS that focuses on executing multiple queries simultaneously. This feature allows multiple queries to be executed simultaneously instead of having to wait for one query to complete before initiating the next one.

The main goal of interquery parallelism is to increase overall system throughput and reduce query response time.

By allowing multiple queries to run concurrently, interquery parallelism makes more efficient use of system resources. This leads to faster query execution and improved performance. For example, if one query is performing a long-running operation, such as a complex join, other queries can still be processed independently without being blocked.

Implementing interquery parallelism involves various techniques, such as dividing the workload into smaller tasks that can be executed simultaneously across different processors or nodes in a distributed database environment.

Parallel database systems are designed with this capability in mind and employ sophisticated algorithms for distributing workloads effectively.

---

**Questions:**

1. Define Parallelism.
2. Explain different architecture involved in shared memory.
3. Explain the differences between parallel database system and Distributed database system.
4. Define Tightly coupled multiprocessor system.
5. What is parallel database? Explain its features and disadvantages in detail.
6. Differentiate between shared-nothing, shared-disk, and shared-memory architectures.
7. Provide examples of each and explain how they affect performance and scalability.
8. Discuss the major components of parallel databases.
9. Discuss how partitioning affects query performance, load balancing, and fault tolerance.
10. Discuss intra-query and inter-query parallelism.
11. Explain how parallelism is introduced in query processing.