Distributed Database Management System (DDBMS) is a type of DBMS which manages a number of databases hoisted at diversified locations and interconnected through a computer network. It provides mechanisms so that the distribution remains oblivious to the users, who perceive the database as a single database.

## Distributed Database Management System

A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

### Features

* It is used to create, retrieve, update and delete distributed databases.
* It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
* It ensures that the data modified at any site is universally updated.
* It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.
* It is designed for heterogeneous database platforms.
* It maintains confidentiality and data integrity of the databases.

## Factors Encouraging DDBMS

The following factors encourage moving over to DDBMS −

* **Distributed Nature of Organizational Units** − Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
* **Need for Sharing of Data** − The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.
* **Support for Both OLTP and OLAP** − Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.
* **Database Recovery** − One of the common techniques used in DDBMS is replication of data across different sites. Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuous to users.
* **Support for Multiple Application Software** − Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

## Advantages of Distributed Databases

Following are the advantages of distributed databases over centralized databases.

**Modular Development** − If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.

**More Reliable** − In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

**Better Response** − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

**Lower Communication Cost** − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

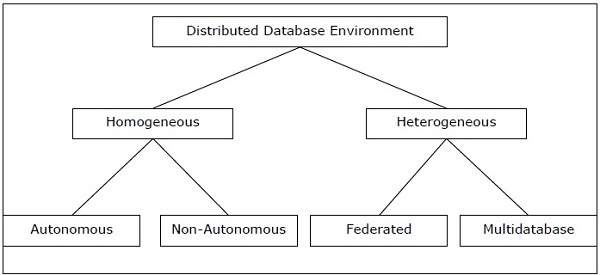
## Adversities of Distributed Databases

Following are some of the adversities associated with distributed databases.

* **Need for complex and expensive software** − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
* **Processing overhead** − Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
* **Data integrity** − The need for updating data in multiple sites pose problems of data integrity.
* **Overheads for improper data distribution** − Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

## Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.



### Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are −

* The sites use very similar software.
* The sites use identical DBMS or DBMS from the same vendor.
* Each site is aware of all other sites and cooperates with other sites to process user requests.
* The database is accessed through a single interface as if it is a single database.

### Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database −

* **Autonomous** − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
* **Non-autonomous** − Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

### Heterogeneous Distributed Databases

In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are −

* Different sites use dissimilar schemas and software.
* The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
* Query processing is complex due to dissimilar schemas.
* Transaction processing is complex due to dissimilar software.
* A site may not be aware of other sites and so there is limited co-operation in processing user requests.

### Types of Heterogeneous Distributed Databases

* **Federated** − The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
* **Un-federated** − The database systems employ a central coordinating module through which the databases are accessed.

## Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters −

* **Distribution** − It states the physical distribution of data across the different sites.
* **Autonomy** − It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
* **Heterogeneity** − It refers to the uniformity or dissimilarity of the data models, system components and databases.

## Architectural Models

Some of the common architectural models are −

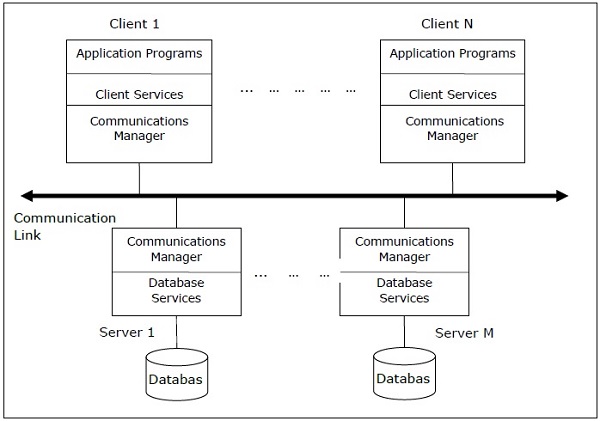
* Client - Server Architecture for DDBMS
* Peer - to - Peer Architecture for DDBMS
* Multi - DBMS Architecture

### Client - Server Architecture for DDBMS

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.

The two different client - server architecture are −

* Single Server Multiple Client
* Multiple Server Multiple Client (shown in the following diagram)



### Peer- to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.

This architecture generally has four levels of schemas −

* **Global Conceptual Schema** − Depicts the global logical view of data.
* **Local Conceptual Schema** − Depicts logical data organization at each site.
* **Local Internal Schema** − Depicts physical data organization at each site.
* **External Schema** − Depicts user view of data.

### Multi - DBMS Architectures

This is an integrated database system formed by a collection of two or more autonomous database systems.

## Design Alternatives

The distribution design alternatives for the tables in a DDBMS are as follows −

* Non-replicated and non-fragmented
* Fully replicated
* Partially replicated
* Fragmented
* Mixed

### Non-replicated & Non-fragmented

In this design alternative, different tables are placed at different sites. Data is placed so that it is at a close proximity to the site where it is used most. It is most suitable for database systems where the percentage of queries needed to join information in tables placed at different sites is low. If an appropriate distribution strategy is adopted, then this design alternative helps to reduce the communication cost during data processing.

### Fully Replicated

In this design alternative, at each site, one copy of all the database tables is stored. Since, each site has its own copy of the entire database, queries are very fast requiring negligible communication cost. On the contrary, the massive redundancy in data requires huge cost during update operations. Hence, this is suitable for systems where a large number of queries is required to be handled whereas the number of database updates is low.

### Partially Replicated

Copies of tables or portions of tables are stored at different sites. The distribution of the tables is done in accordance to the frequency of access. This takes into consideration the fact that the frequency of accessing the tables vary considerably from site to site. The number of copies of the tables (or portions) depends on how frequently the access queries execute and the site which generate the access queries.

### Fragmented

In this design, a table is divided into two or more pieces referred to as fragments or partitions, and each fragment can be stored at different sites. This considers the fact that it seldom happens that all data stored in a table is required at a given site. Moreover, fragmentation increases parallelism and provides better disaster recovery. Here, there is only one copy of each fragment in the system, i.e. no redundant data.

The three fragmentation techniques are −

* Vertical fragmentation
* Horizontal fragmentation
* Hybrid fragmentation

### Mixed Distribution

This is a combination of fragmentation and partial replications. Here, the tables are initially fragmented in any form (horizontal or vertical), and then these fragments are partially replicated across the different sites according to the frequency of accessing the fragments.

## Data Replication

Data replication is the process of storing separate copies of the database at two or more sites. It is a popular fault tolerance technique of distributed databases.

### Advantages of Data Replication

* **Reliability** − In case of failure of any site, the database system continues to work since a copy is available at another site(s).
* **Reduction in Network Load** − Since local copies of data are available, query processing can be done with reduced network usage, particularly during prime hours. Data updating can be done at non-prime hours.
* **Quicker Response** − Availability of local copies of data ensures quick query processing and consequently quick response time.
* **Simpler Transactions** − Transactions require less number of joins of tables located at different sites and minimal coordination across the network. Thus, they become simpler in nature.

### Disadvantages of Data Replication

* **Increased Storage Requirements** − Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.
* **Increased Cost and Complexity of Data Updating** − Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.
* **Undesirable Application – Database coupling** − If complex update mechanisms are not used, removing data inconsistency requires complex co-ordination at application level. This results in undesirable application – database coupling.

Some commonly used replication techniques are −

* Snapshot replication
* Near-real-time replication
* Pull replication

## Fragmentation

Fragmentation is the task of dividing a table into a set of smaller tables. The subsets of the table are called **fragments**. Fragmentation can be of three types: horizontal, vertical, and hybrid (combination of horizontal and vertical). Horizontal fragmentation can further be classified into two techniques: primary horizontal fragmentation and derived horizontal fragmentation.

Fragmentation should be done in a way so that the original table can be reconstructed from the fragments. This is needed so that the original table can be reconstructed from the fragments whenever required. This requirement is called “reconstructiveness.”

### Advantages of Fragmentation

* Since data is stored close to the site of usage, efficiency of the database system is increased.
* Local query optimization techniques are sufficient for most queries since data is locally available.
* Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

### Disadvantages of Fragmentation

* When data from different fragments are required, the access speeds may be very high.
* In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
* Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

## Vertical Fragmentation

In vertical fragmentation, the fields or columns of a table are grouped into fragments. In order to maintain reconstructiveness, each fragment should contain the primary key field(s) of the table. Vertical fragmentation can be used to enforce privacy of data.

For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

STUDENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Regd\_No | Name | Course | Address | Semester | Fees | Marks |

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows −

CREATE TABLE STD\_FEES AS

SELECT Regd\_No, Fees

FROM STUDENT;

## Horizontal Fragmentation

Horizontal fragmentation groups the tuples of a table in accordance to values of one or more fields. Horizontal fragmentation should also confirm to the rule of reconstructiveness. Each horizontal fragment must have all columns of the original base table.

For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows −

CREATE COMP\_STD AS

SELECT \* FROM STUDENT

WHERE COURSE = "Computer Science";

## Hybrid Fragmentation

In hybrid fragmentation, a combination of horizontal and vertical fragmentation techniques are used. This is the most flexible fragmentation technique since it generates fragments with minimal extraneous information. However, reconstruction of the original table is often an expensive task.

Hybrid fragmentation can be done in two alternative ways −

* At first, generate a set of horizontal fragments; then generate vertical fragments from one or more of the horizontal fragments.
* At first, generate a set of vertical fragments; then generate horizontal fragments from one or more of the vertical fragments

Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users. The DDBMS designer may choose to fragment tables, replicate the fragments and store them at different sites. However, since users are oblivious of these details, they find the distributed database easy to use like any centralized database.

The three dimensions of distribution transparency are −

* Location transparency
* Fragmentation transparency
* Replication transparency

Database control refers to the task of enforcing regulations so as to provide correct data to authentic users and applications of a database. In order that correct data is available to users, all data should conform to the integrity constraints defined in the database. Besides, data should be screened away from unauthorized users so as to maintain security and privacy of the database. Database control is one of the primary tasks of the database administrator (DBA).

The three dimensions of database control are −

* Authentication
* Access rights
* Integrity constraints

When a query is placed, it is at first scanned, parsed and validated. An internal representation of the query is then created such as a query tree or a query graph. Then alternative execution strategies are devised for retrieving results from the database tables. The process of choosing the most appropriate execution strategy for query processing is called query optimization.

## Query Optimization Issues in DDBMS

In DDBMS, query optimization is a crucial task. The complexity is high since number of alternative strategies may increase exponentially due to the following factors −

* The presence of a number of fragments.
* Distribution of the fragments or tables across various sites.
* The speed of communication links.
* Disparity in local processing capabilities.

Hence, in a distributed system, the target is often to find a good execution strategy for query processing rather than the best one. The time to execute a query is the sum of the following −

* Time to communicate queries to databases.
* Time to execute local query fragments.
* Time to assemble data from different sites.
* Time to display results to the application.
* Database management system is any software that manages and controls the storage, the organization, security, retrieval and integral of data in a specific database, whereas DDBMS consist of a single database that is divided into many fragments. Each fragment is integrated on one or more computer and controlled by independent database (DBMS) (Connolly & Begg, 2004).
* In centralized DBMS the data is distributed across the network computers, and the data is stored on many sites and under the management responsibility of DDBMS. But in the DBMS data is stored and controlled in a central site.
* Both of DDBMS and centralized DBMS provide the access to database using the same interface, but for this function centralized DBMS faces less complication than DDBMS.
* For distributing data over network we can use replication or fragmentation. The objective of replication and fragmentation is to make a transparency of this allocation to make the details of implementation hidden on users. In centralized DBMS is not need to make transparencies.
* In DDBMS design we can find three issues which are not in centralized DBMS design. These issues are: How to split the database to fragments, and fragments to replicate, and in which locate we can find these fragments.
* Consequently, centralized DBMS is less sophisticated than DDBMS because it not supports the organizational structure of today’s widely distributed enterprises, and DDBMS more reactive and reliable (Blurtit, 2010).

2- Advantages of DDBMSs

The DDBMS have many advantages over DBMS. We can gather them in the following points:

Reflects organizational structure: We can distribute the data base over any organization offices which has distributed locations

Improved shareability and local autonomy: Users can use the data of other sites, which mean the data can be near of the users who use it. Also by this way the data can be controlled by the user.

Improved availability: Unlike centralized DBMS, the failure at one site or link of communication makes only some parts of system inaccessible, which means the entire system is still working.

Improved reliability: The replication system make the data exist in many site. So this insures the possibility of accessing to this data if there is any failure happened.

Improved performance: We can improve the accessing speed to data base if we use remote centralized database. Also, there may not be same conflict for CPU and using the services, like DBMS.

Economics: For making the organization systems more cost-effective to obtain separate computers, DBMS allows us to create systems of smaller computer, its power equal the power of one large computer.

Modular growth: This one refers to the flexibility of DDBMS, where we can add a new site without any affects on the operation of other sites.

3- Disadvantages

Complexity: The features of DDBMS like replication, availability flexibility make the implementation of DDBMS more complex than centralized DBMS.

Cost: The cost will increase due to the complexity in DDBMS than centralized DBMS. It requires the maintenance and additional hardware to create a network between the posts.

Security: Unlike centralized system, access to data not only has to be secured, but the entire network also has to be made secure.

Integrity control more difficult: In a DDBMS, the costs of processing and communication that we have to make to implement integrity constraints may be prohibitive.

Lack of standards: Despite there is effective communication when we use DDBMS, There are no methodologies or tools to assist users convert a centralized DBMS into a distributed DBMS.

Lack of experience: Unlike what we have with centralized DBMSs, still our experiences are lack in industry of DDBMS.

Database design more complex: The design of DDMBS is more complex than centralized database, because the data fragmentation has to be implemented in this design, and also data replication, and fragments allocation to specific sites

4- Possibility of using the replication server:

The replication is a mechanism enables the companies to provide their users accessing to current data at the time they need it (Connolly & Begg, 2004).

INTRODUCTION

• In centralized database:

• Data is located in one place (one server)

• All DBMS functionalities are done by that server

• Enforcing ACID properties of transactions

• Concurrency control, recovery mechanisms

• Answering queries

In Distributed databases:

• Data is stored in multiple places (each is running a DBMS)

• New notion of distributed transactions

• DBMS functionalities are now distributed over many machines

• Revisit how these functionalities work in distributed environment

**PARALLEL VS. DISTRIBUTED DATABASES**

• Distributed processing usually imply parallel processing

(not vise versa)

• Can have parallel processing on a single machine

• Assumptions about architecture

**Parallel Databases**

• Machines are physically close to each other, e.g., same server room

• Machines connects with dedicated high-speed LANs and switches

• Communication cost is assumed to be small

• Can shared-memory, shared-disk, or shared-nothing architecture

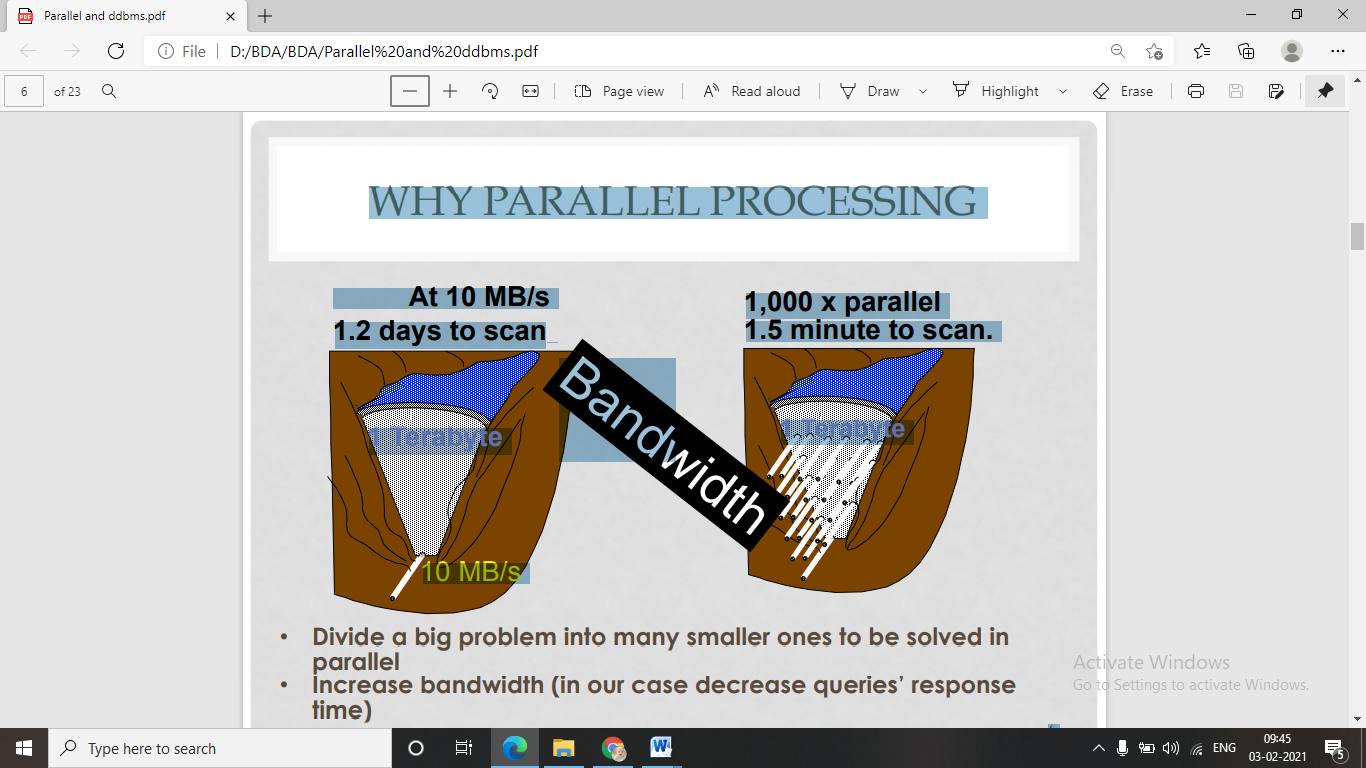
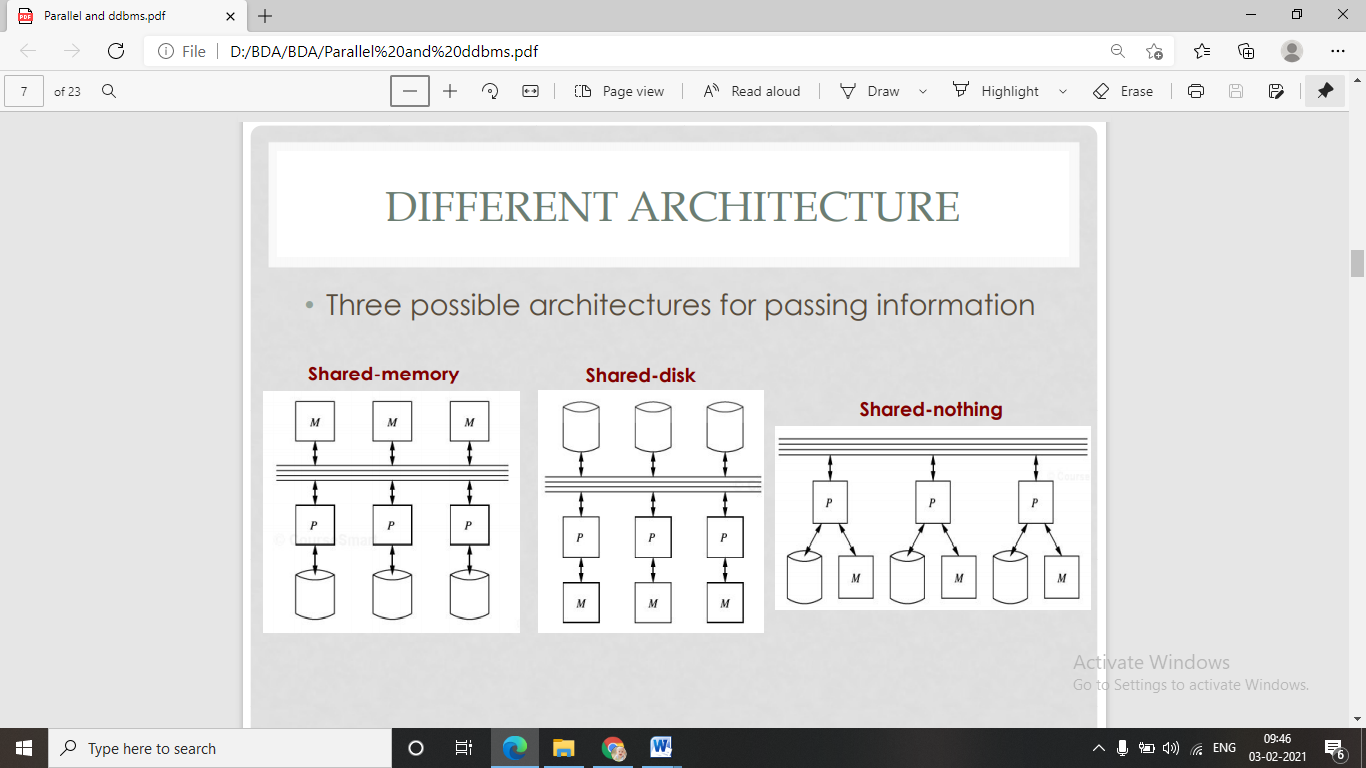
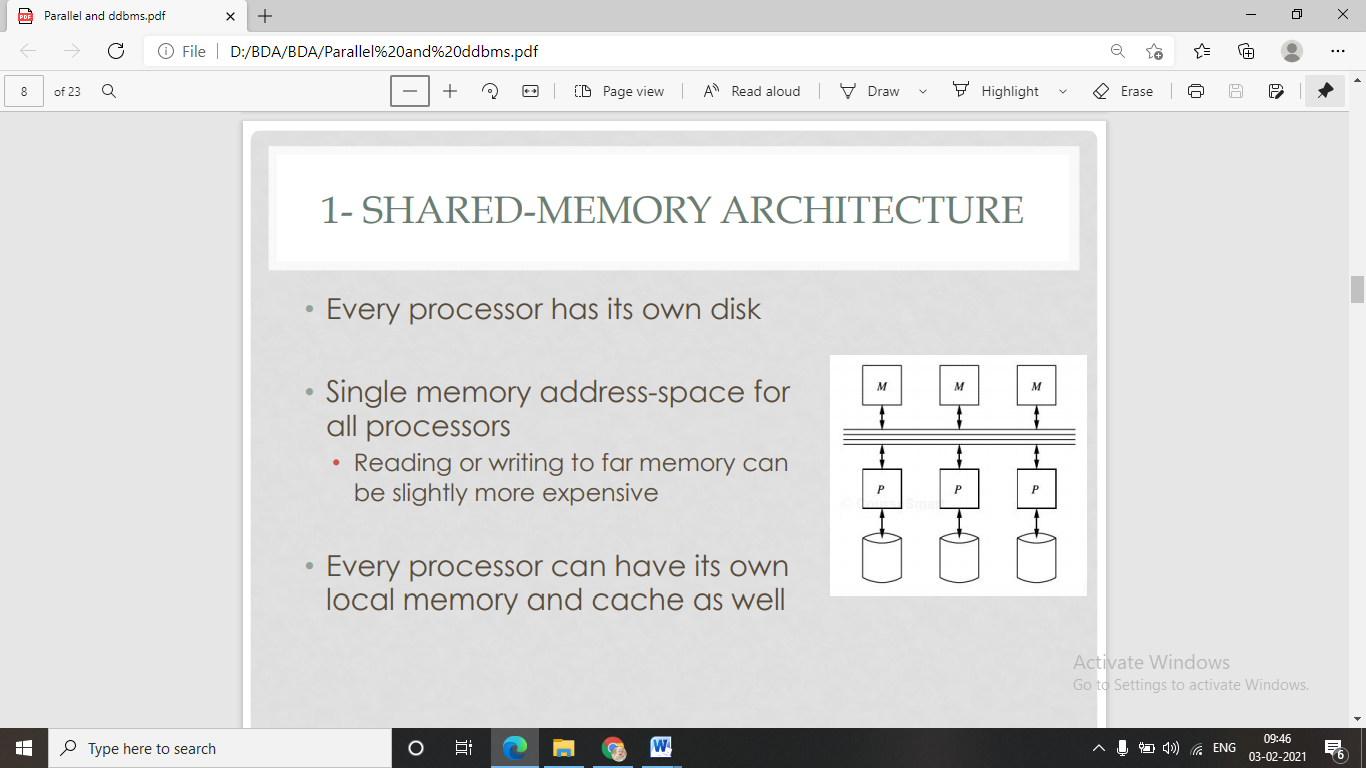
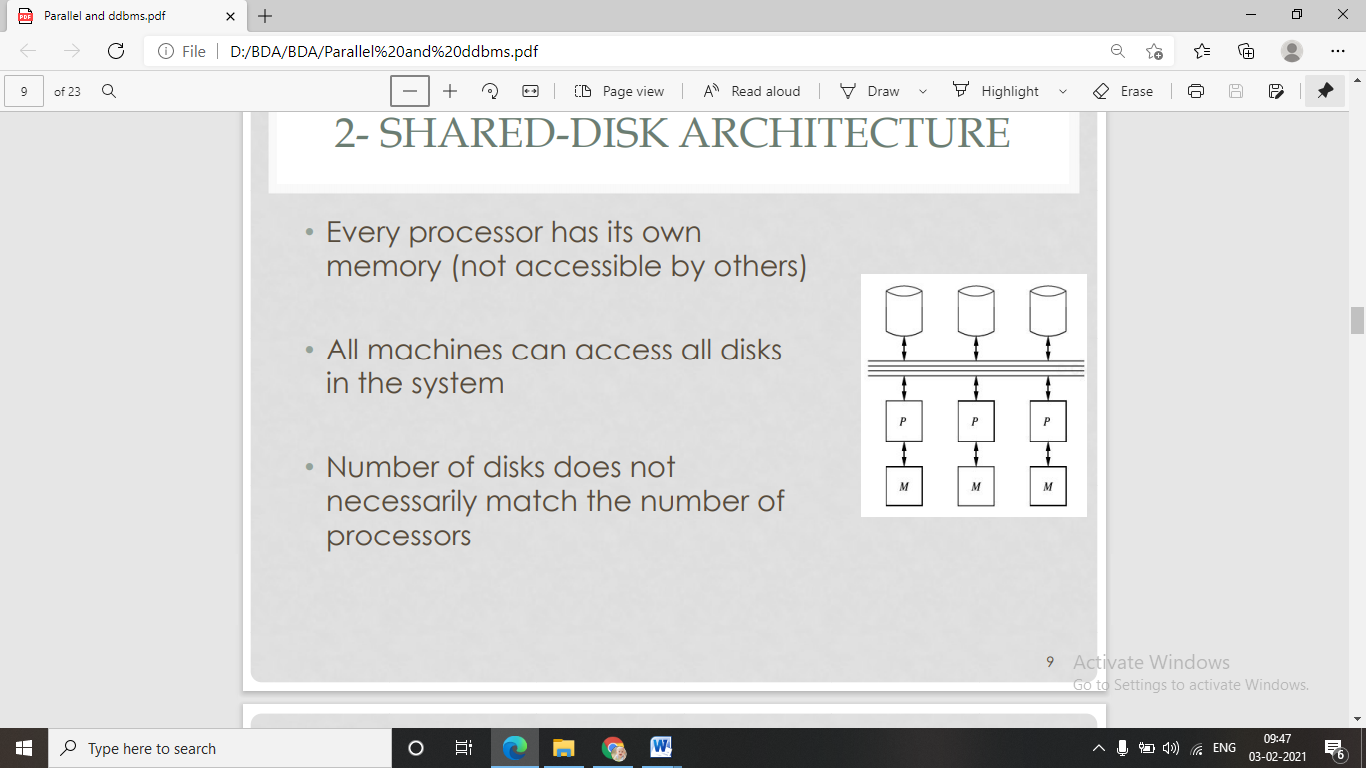
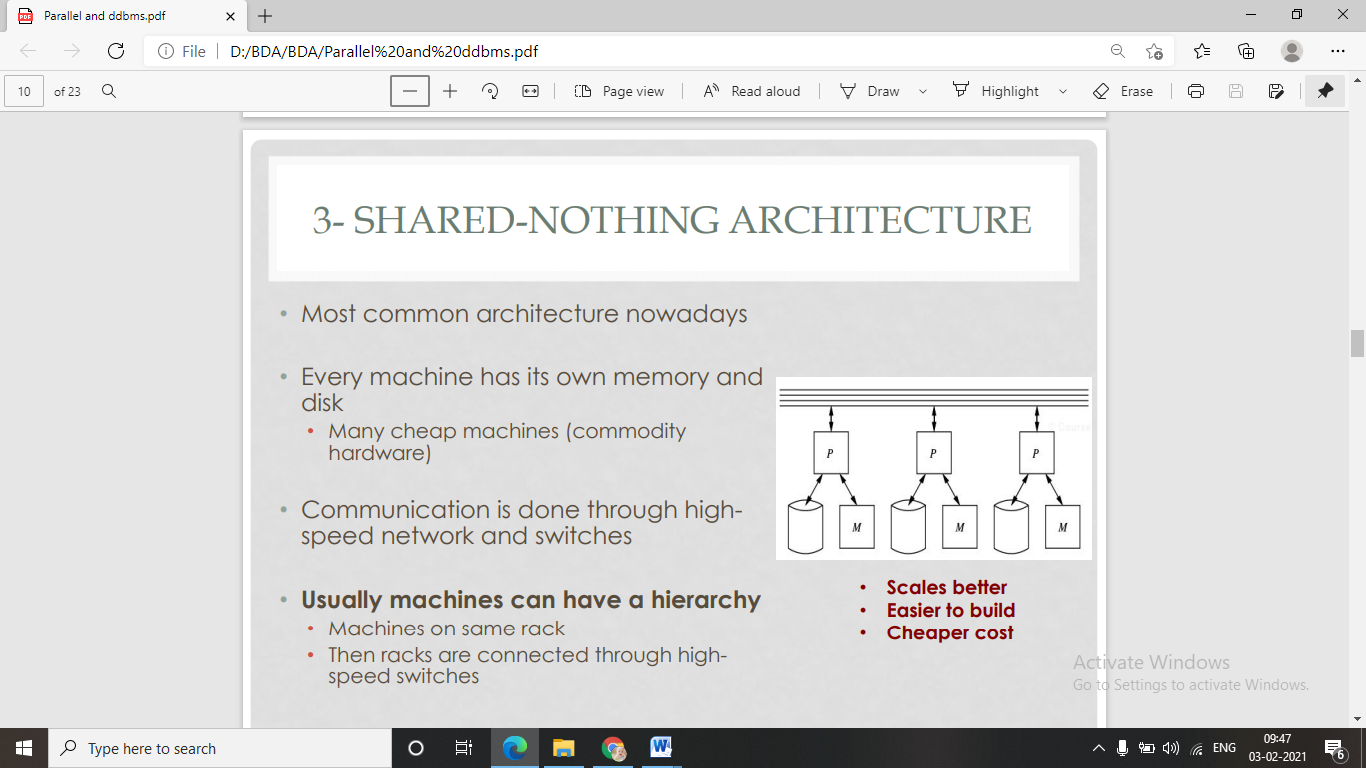
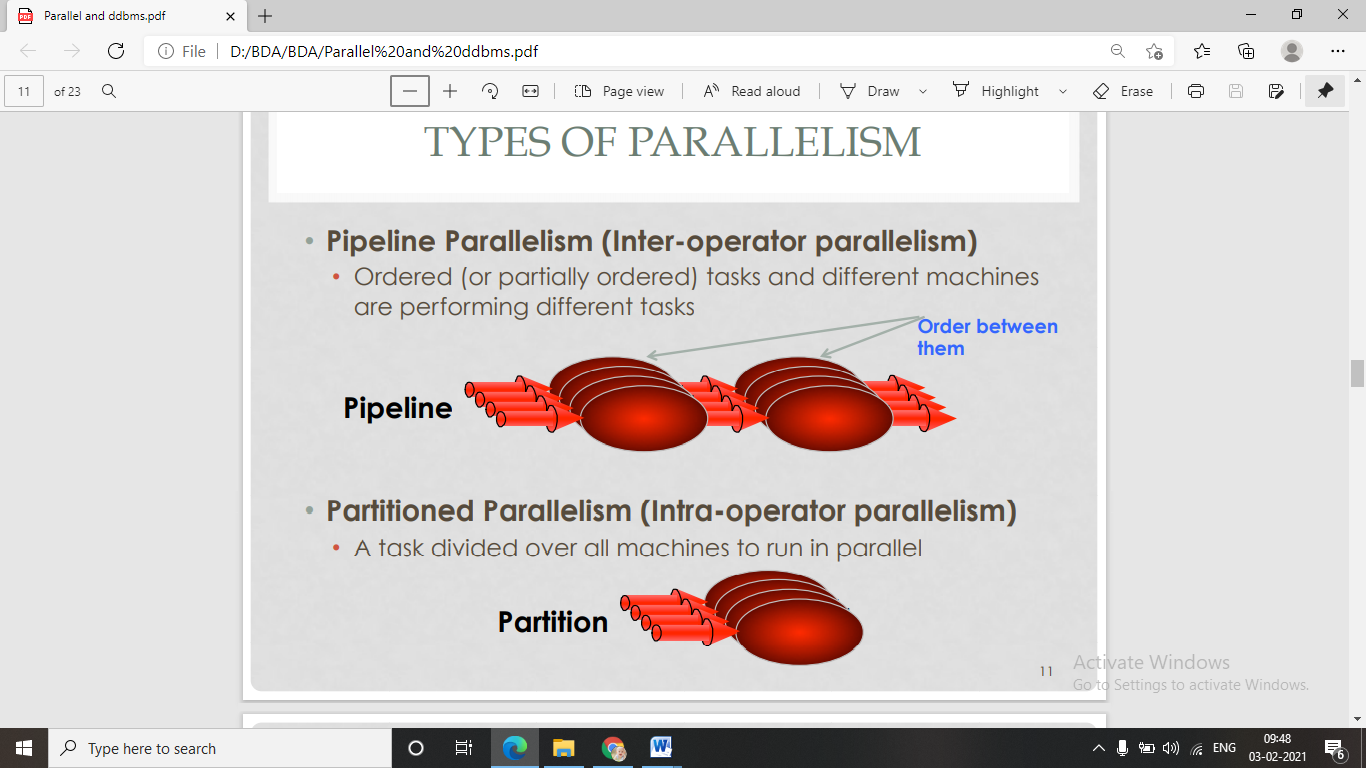
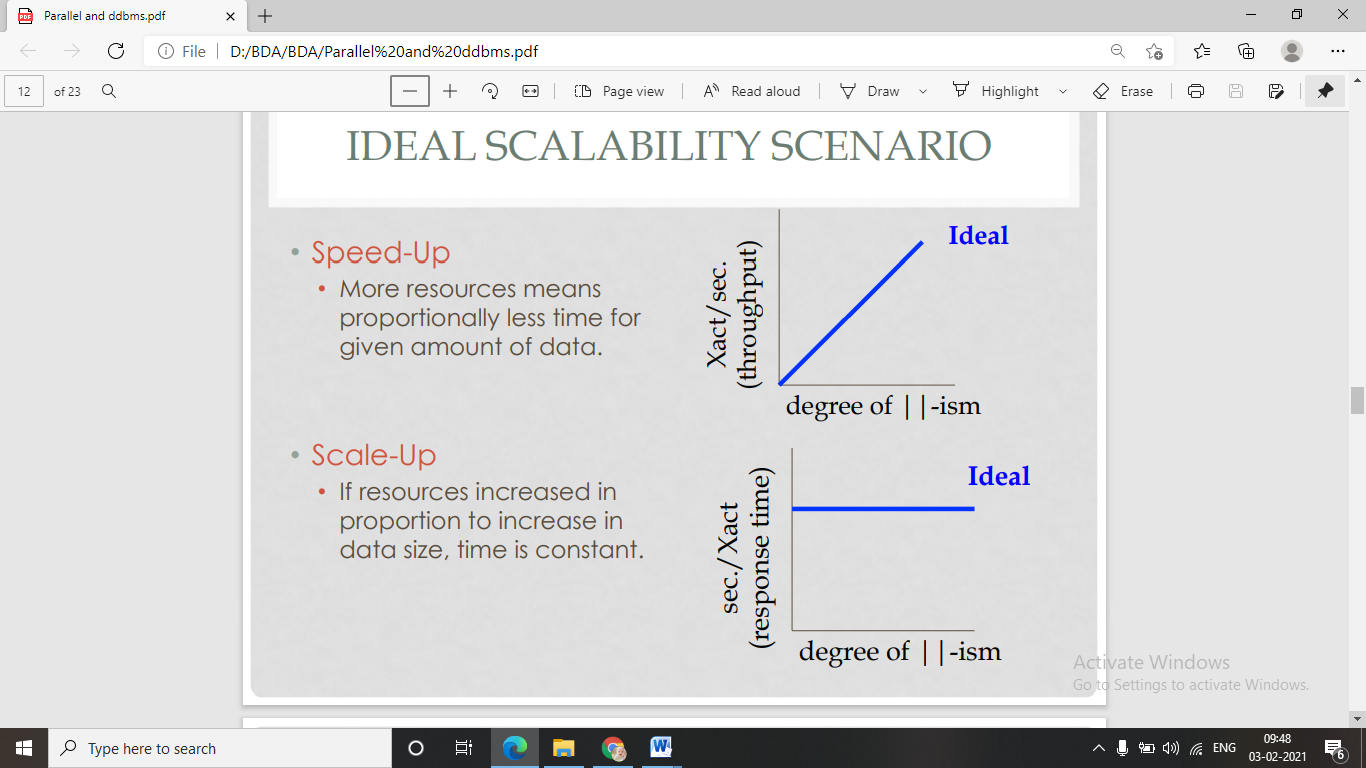
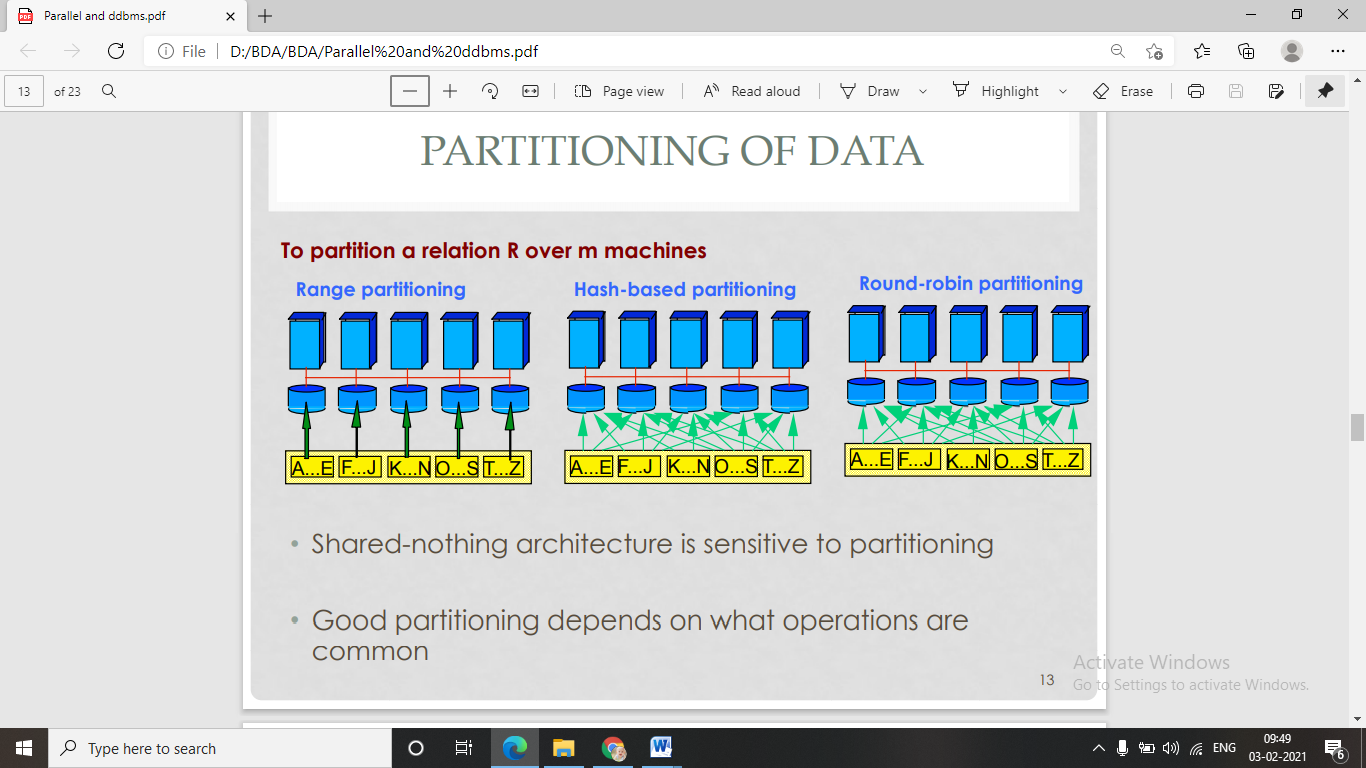
**Distributed Databases**

• Machines can far from each other, e.g., in different continent

• Can be connected using public-purpose network, e.g., Internet

• Communication cost and problems cannot be ignored

• Usually shared-nothing architecture

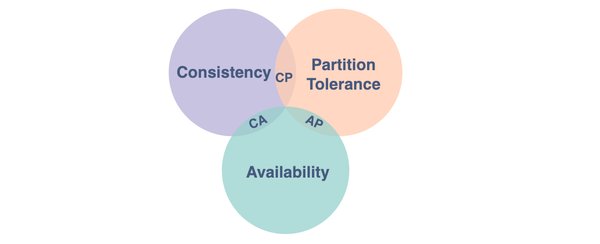
       

# What is the CAP theorem?

**The CAP theorem**, originally introduced as the CAP principle, can be used to explain some of the competing requirements in a distributed system with replication. It is a tool used to makes system designers aware of the trade-offs while designing networked shared-data systems.

The CAP theorem states that it is not possible to guarantee all three of the desirable properties – consistency, availability, and partition tolerance at the same time in a distributed system with data replication.

The **CAP theorem** (also called Brewer’s theorem) states that a distributed database system can only guarantee two out of these three characteristics: Consistency, Availability, and Partition Tolerance.

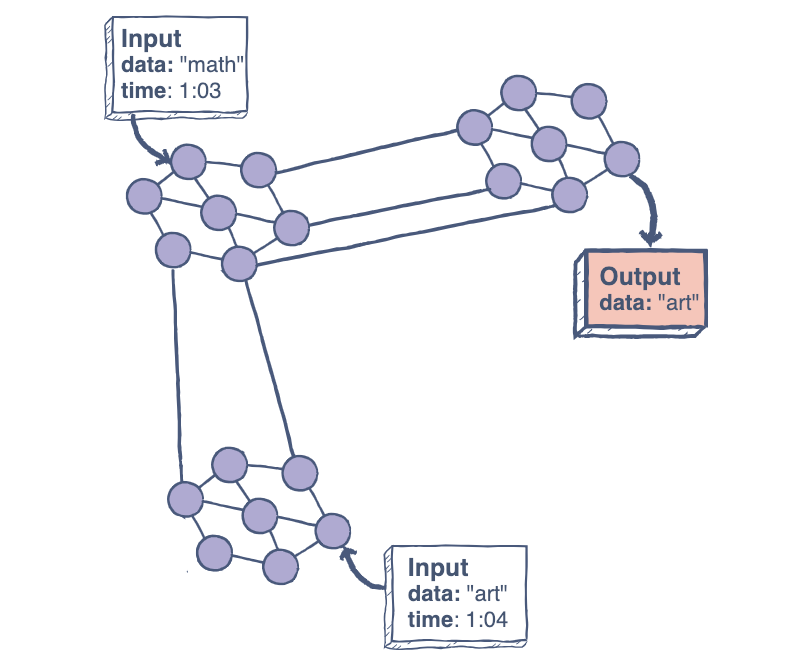


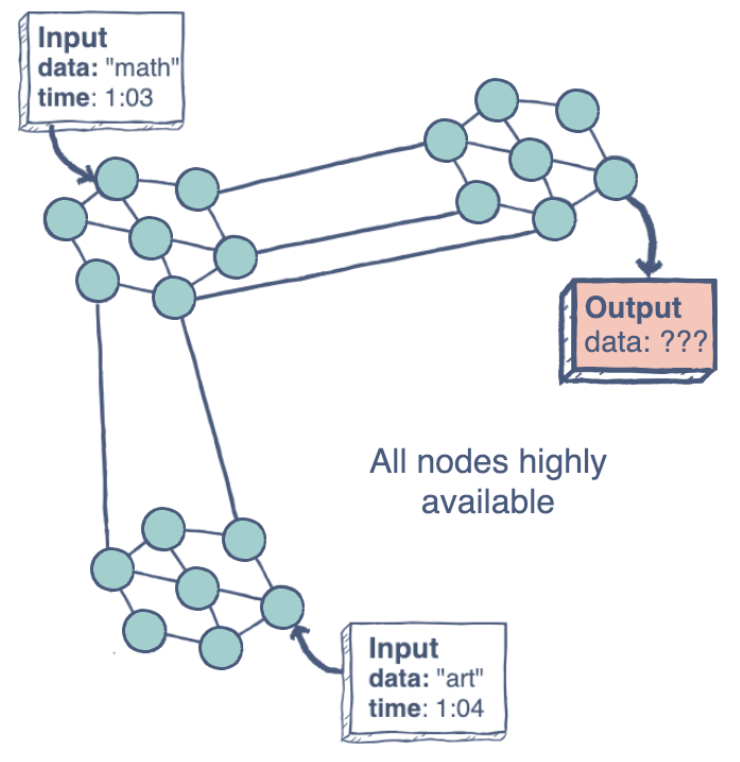


### Consistency

A system is said to be consistent if all nodes see the same data at the same time.

Simply, if we perform a read operation on a consistent system, it should return the value of the most recent write operation. This means that, the read should cause all nodes to return the same data, i.e., the value of the most recent write.





### Availability

Availability in a distributed system ensures that the system remains operational 100% of the time. Every request gets a (non-error) response regardless of the individual state of a node.

Note: this does not guarantee that the response contains the most recent write.

The figure on the left illustrates an “unavailable” system.

### Partition Tolerance

This condition states that the system does not fail, regardless of if messages are dropped or delayed between nodes in a system.

Partition tolerance has become more of a necessity than an option in distributed systems. It is made possible by sufficiently replicating records across combinations of nodes and networks.