

Log In

Join

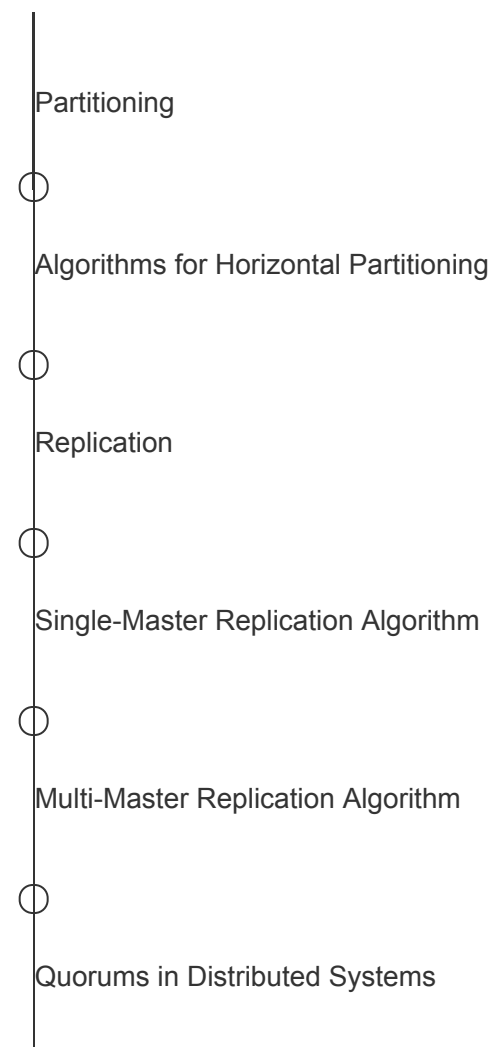
Back To Module Home


Distributed Systems

0% completed

Introduction to Distributed Systems

Basic Concepts and Theorems





○	Safety Guarantees in Distributed Systems
○	ACID Transactions
○	The CAP Theorem
○	Consistency Models
○	CAP Theorem's Consistency Model
○	Isolation Levels and Anomalies
○	Prevention of Anomalies in Isolation Levels
○	Consistency and Isolation
○	Hierarchy of Models
○	Why All the Formalities?
○	Quiz

Conclusion

Mark Module as Completed

Partitioning

See how we can make our system scalable by partitioning.

We'll cover the following

- Scalability
- Mechanism to achieve scalability
 - Partitioning
 - Vertical partitioning
 - Horizontal partitioning
- Limitations of partitioning

One of the major benefits of distributed systems is **scalability**.

Scalability#

Scalability lets us store and process datasets much larger than what we could with a single machine.

Increasing resources

Mechanism to achieve scalability#

One of the primary mechanisms of achieving scalability is **partitioning**.

Partitioning#

Partitioning is the process of splitting a dataset into multiple, smaller datasets, and then assigning the responsibility of storing and processing them to different nodes of a distributed system. This allows us to add more nodes to our system and increase the size of the data it can handle.

There are two different variations of partitioning:

1. Vertical partitioning
2. Horizontal partitioning (or **sharding**)

The terms “vertical” and “horizontal” originate from the era of relational databases that established the notion of a tabular view of data.

In this view, data consists of rows and columns, where each row is a different entry in the dataset, and each column is a different attribute for every entry.

The following illustration contains a visual depiction of the difference between **vertical partitioning** and **horizontal partitioning**.

Vertical partitioning vs horizontal partitioning

Vertical partitioning#

Vertical partitioning involves splitting a table into multiple tables with fewer columns and using additional tables to store columns that relate rows across tables. We commonly refer to this as a **join operation**. We can then store these different tables in different nodes.

Normalization is one way to perform vertical partitioning. However, general vertical partitioning goes far beyond that: it splits a column, even when they are normalized.

Horizontal partitioning#

Horizontal partitioning involves splitting a table into multiple, smaller tables, where each table contains a percentage of the initial table's rows. We can then store these different sub-tables in different nodes.

We can perform this split through multiple strategies.

A simplistic approach for this is an alphabetical split. For instance, we can horizontally partition a table that contains the students of a school by using the students' surnames. The following illustration shows how.

Horizontal partitioning using alphabetical split

Limitations of partitioning#

In a **vertically partitioned system**, requests that need to combine data from different tables (i.e., join operations) become less efficient. This is because these requests may now have to access data from multiple nodes.

In a **horizontally partitioned system**, we can usually avoid accessing data from multiple nodes because all the data for each row is located in the same node. However, we may still need to access data from multiple nodes for requests that are searching for a range of rows that belong to multiple nodes.

Another important implication of horizontal partitioning is the potential for loss of transactional semantics.

When we store data in a single machine, we can easily perform multiple operations in an atomic way, where either *all* or *none* of them succeed. However, this is much harder to achieve in a distributed system.

As a result, it's much harder to perform atomic operations—when partitioning data horizontally—over data that resides in different nodes.

This is a common theme in distributed systems; there's no silver bullet. We have to make trade-offs to achieve the property we desire.

Vertical partitioning is mainly a data modeling practice, which can be performed by the engineers designing a system—sometimes independently of the storage systems used. However, horizontal partitioning is a common feature of distributed databases. So, to use these systems properly, engineers need to know how the system works under the hood. Therefore, we will mostly focus on horizontal partitioning.

Back

Quiz

Next

[Algorithms for Horizontal Partitioning](#)

[Mark as Completed](#)

[Report an Issue](#)