

Log In

Join

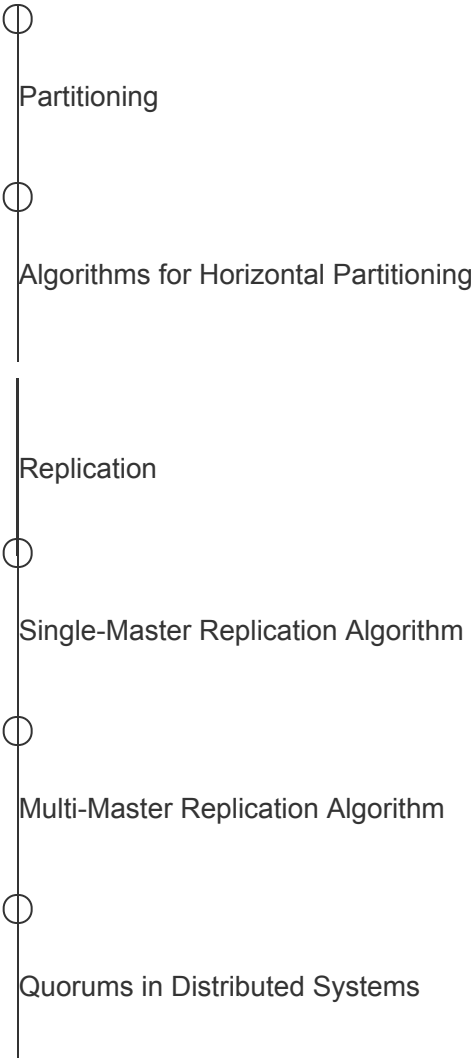
Back To Module Home


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

Replication

Learn what replication is and why it is used in distributed systems.

We'll cover the following

- Availability
- Mechanism to achieve availability
 - Replication
 - Pessimistic replication
 - Optimistic replication

Partitioning can improve the scalability and performance of a system by distributing data and request load to multiple nodes.

Another dimension that benefits from using a distributed system is known as **availability**.

Availability#

Availability refers to the ability of the system to remain functional despite failures in parts of it.

Mechanism to achieve availability#

The technique we use to achieve availability is **replication**.

Replication#

Replication is the main technique used in distributed systems to increase availability. It consists of storing the same piece of data in multiple nodes (called replicas) so that if one of them crashes, data is not lost, and requests can be served from the other nodes in the meanwhile.

Copies of the same data

However, the benefit of increased availability from replication comes with a set of new complications.

Replication implies that the system now has multiple copies of every piece of data. These copies must be maintained and kept in sync with each other on every update.

Ideally, replication should function transparently to the end-user, or engineer. This is to create the illusion that there's only one copy of every piece of data. This makes a distributed system look like a simple, centralized system of a single node that is much easier to reason about and develop software around.

Of course, this is not always possible. We may require significant hardware resources or need to give up other desirable properties to achieve this ideal. For instance, engineers sometimes willingly accept a system that provides much higher performance, but occasionally gives a non-consistent view of the data. Hence, they only do this under specific conditions—and in a specific way—they can account for when they design the application.

Therefore, there are two main strategies for replication:

1. Pessimistic replication
2. Optimistic replication

Pessimistic replication#

Pessimistic replication tries to guarantee from the beginning that all the replicas are identical to each other—as if there was only one copy of the data all along.

Optimistic replication#

Optimistic replication, or lazy replication, allows the different replicas to diverge. This guarantees that they will converge again if the system does not receive any updates, or enters a quiesced state, for a period of time.

Replication is a very active field in research, so there are many different algorithms for it.

Back

Algorithms for Horizontal Partitioning

Next

Single-Master Replication Algorithm

Mark as Completed

Report an Issue