Log In Join

Back To Module Home

Distributed Systems

0% completed

Search Module

Introduction to Distributed Systems

- Getting Started
- Fallacies of Distributed Computing
- Difficulties Designing Distributed Systems
- Measures of Correctness in Distributed Systems
- System Models
- Types of Failures
- The Tale of Exactly-Once Semantics

Mark Module as Completed

Getting Started

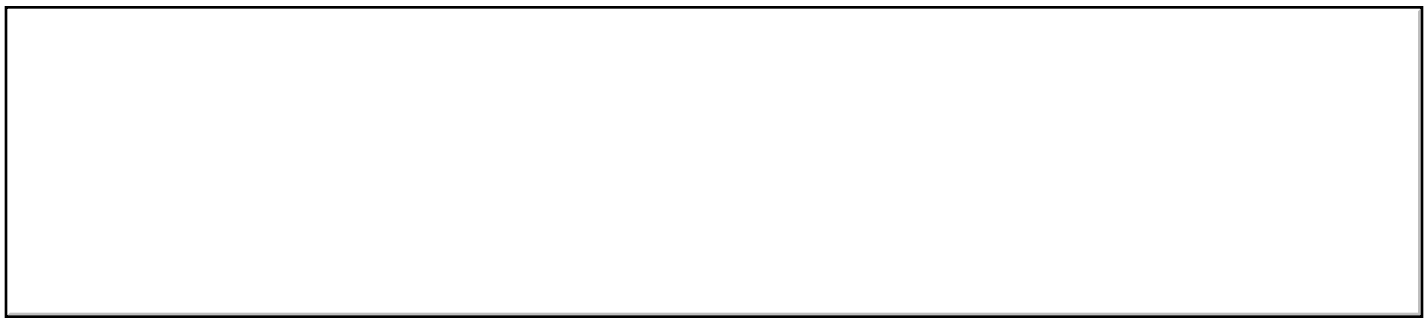
Learn what a distributed system is and why we need it.

We'll cover the following

What is a distributed system?#

According to Coulouris et al., "A **distributed system** is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another."

The components of this system can be thought of as software programs that run on physical hardware, such as computers. These components take many forms; e.g., they can be web servers, routers, web browsers, etc. To keep a generic view, we assume that each program runs on a separate machine. We refer to each of these machines as a **node**.



A distributed system

The above illustration shows that the network either consists of direct connections between the distributed system components, or more components that form the backbone of the network (e.g., if communication is done through the Internet).

> While the generic node view is helpful to understand the diagram above, sometimes real-life examples of how the nodes work may be more helpful... In these cases, we explain the role of each node in the system in detail.

Parts of a distributed system#

There are two categories of the central parts that help distributed systems function:

- The various parts that compose a distributed system: These are located remotely and are separated by a network
- The network that separates the various parts of a distributed system: It acts as a communication mechanism that lets them exchange messages.

> We will see the individual parts in detail later in this course.

Why we need a distributed system#

There are three main benefits of distributed systems, as shown in the illustration below.

The three main benefits of a distributed system

Let's explain each one separately.

Performance#

According to Mohan et al., "**Performance** is the degree to which a software system or component meets its objectives for timeliness."

Problem with a single computer#

The physical constraints of its hardware impose certain limits on the performance of a single computer. Moreover, it is extremely expensive to improve a single computer's performance after a certain point.

Solution#

We can achieve the same performance with two or more low-spec computers as with a single, high-end computer. So, distributed systems allow us to achieve better performance at a lower cost.

> Note that better performance can translate to different things depending on the context, such as lower latency per request, higher throughput, etc.

Scalability#

According to Bondi et al., "Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth."

Problem with a single computer#

Data storage and processing are responsible for most of the value that software systems impart in the real world. As a system's customer base grows, the system needs to handle more traffic and store larger amounts of data. However, a system that comprises a single computer can only scale up to a certain point, as explained earlier.

Solution#

If we build a distributed system, we can split and store the data in multiple computers, and distribute the processing work.

> **Vertical scaling** refers to the approach of scaling a system by adding resources (memory, CPU, disk, etc.) to a single node. Meanwhile, **horizontal scaling** refers to the approach of scaling by adding more nodes to the system.

As a result of this, we can scale our systems to sizes that we could not imagine with a single computer system.

Availability#

In the context of software systems, **availability** is the probability of a system to work as required, when required, during a mission.

Problem with a single computer#

Nowadays, most online services need to operate all the time (also known as “24/7 service”), which is a huge challenge. When a service states that it has five-nine availability, it usually operates 99.999% of the time. This implies that it can be down for only 5 minutes at most per year to satisfy this guarantee.

If we consider how unreliable hardware can be, we can easily understand how big an undertaking this is. Of course, it would be infeasible to provide this kind of guarantee with a single computer.

Solution#

Redundancy is one of the widely used mechanisms to achieve higher availability. It refers to storing data into multiple, redundant computers. So, when one computer fails, we can efficiently switch to another one. This way, we’ll prevent our customers from experiencing this failure.

Given that data are stored now in multiple computers, we end up with a distributed system!

> If we leverage a distributed system, we get all of the above benefits. However, as we will see later on, there is tension between them and several other properties. So, in most cases, we have to make a trade-off. To do this, we must understand the basic constraints and limitations of distributed systems. The first part of this course will help us with this.

Next

Fallacies of Distributed Computing

Mark as Completed

Report an Issue