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

Fallacies of Distributed Computing

Let's see what false assumptions developers make while developing software for the distributed systems.

We'll cover the following

- The global clock fallacy

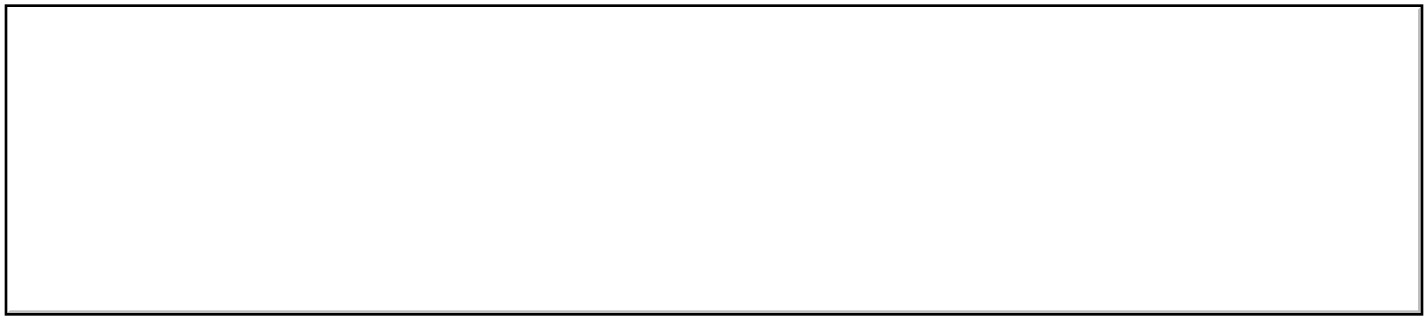
The difference in developing software for distributed systems#

Distributed systems are subject to many more constraints than software systems that run on a single computer. As a result, the development of software for distributed systems is also very different. However, those who are new to distributed systems make assumptions based on their experience with software development for systems that run on a single computer. Of course, this creates a lot of problems down the road in the systems they build.

To eliminate this confusion and help people build better systems, L Peter Deutsch and others at Sun Microsystems created a collection of these false assumptions. These are the fallacies of distributed computing.

Fallacies#

There are eight such fallacies of distributed computing. The following illustration lists them.



False assumptions made by developers while developing software for distributed systems

As you progress through the course, you'll gain a deeper understanding of why these

statements are fallacious.

However, we'll give you a sneak preview here by quickly going over them and explaining where they fall short.

The network is reliable#

The abstractions developers learn from various technologies and protocols often enforce this common fallacy. As we will see in a later chapter, networking protocols like TCP can make us believe that the network is reliable and never fails. However, this is just an illusion with significant repercussions. Also, we build network connections on top of hardware that will also fail at some point. Hence, we should design our systems accordingly.

Latency is zero#

Libraries that attempt to model remote procedure calls as local calls, such as gRPC or Thrift, enforce this assumption. We should always remember that there's a large difference (from milliseconds to nanoseconds) in latency between a call to a remote system and that to local memory access. This gets even worse when we consider calls between data centers on different continents. Thus, this is another thing to keep in mind when deciding how to geo-distribute our system.

Bandwidth is infinite#

This fallacy is weaker nowadays. This is because the bandwidth we can achieve has significantly improved in the last few decades. For instance, we can now build high-bandwidth connections in our own data centers. However, this does not mean we can use all of it if our traffic needs to cross the Internet. This is important to consider when we make decisions about our distributed system's topology, and when requests travel

through the Internet.

The network is secure#

This fallacy shows that the wider network used by two nodes to communicate is not necessarily under their control. Thus, we should consider it insecure.

The course dedicates a portion to security, where it explains the various techniques we can use to securely utilize an insecure network. This network also comprises many different parts that different organizations may manage with different hardware. Moreover, failures in some parts of this network may require us to change its topology to keep it functional.

The following fallacies highlight this:

- Topology doesn't change
- There is one administrator
- The network is homogeneous

Transport cost is zero#

The transportation of data between two points incurs financial costs. We should factor this in when we build a distributed system.

The global clock fallacy#

There's one fallacy that's not a part of the above set, but still often causes confusion amongst people new to distributed systems. . If we follow the same style as above, we can phrase this fallacy as:

"Distributed systems have a global clock, which we can use to identify when events happen."

This assumption is quite deceiving since it's somewhat intuitive and holds true even in non-distributed systems. For instance, an application that runs on a single computer can use the computer's local clock to decide when events happen, and in what order. However, this is not true in a distributed system, where every node in the system has its own local clock that runs at a unique rate.

While there are ways to keep the clocks in sync, some are very expensive and don't completely eliminate these differences. Physical laws also bind this limitation. An example of this is the TrueTime API built by Google, which exposes the clock uncertainty explicitly as a first-class citizen.

> However, as we'll see in the upcoming lessons that discuss cause and effects, there are other ways to reason about time using logical clocks.

Getting Started

Next

Difficulties Designing Distributed Syst…

Mark as Completed

Report an Issue