Join

Back To Course Home

Grokking Modern System Design Interview for Engineers & Managers

0% completed

Search Course

System Design Interviews

Introduction

Abstractions

Non-functional System Characteristics

Back-of-the-envelope Calculations

Building Blocks

Domain Name System

Load Balancers

Databases

Key-value Store

☐

Content Delivery Network (CDN)

☐

Sequencer

☐

Distributed Monitoring

☐

Monitor Server-side Errors

☐

Monitor Client-side Errors

☐

Distributed Cache

☐

Distributed Messaging Queue

☐

Pub-sub

☐

Rate Limiter

☐

Blob Store

☐

Distributed Search

☐

Distributed Logging

☐

Distributed Task Scheduler

☐

Sharded Counters

☐

Concluding the Building Blocks Discussion

☐

Design YouTube

☐

Design Quora

☐

Design Google Maps

☐

Design a Proximity Service / Yelp

☐

Design Uber

☐

Design Twitter

☐

Design Newsfeed System

☐

Design Instagram

☐

Design a URL Shortening Service / TinyURL

☐

Design a Web Crawler

☐

Design WhatsApp

☐

Design Typeahead Suggestion

☐

Design a Collaborative Document Editing Service / Google Docs

☐

Spectacular Failures

☐

Introduction to Distributed System Failures

Facebook, WhatsApp, Instagram, Oculus Outage

AWS Kinesis Outage Affecting Many Organizations

AWS Wide Spread Outage

Concluding Remarks

☐

Course Certificate

Mark Course as Completed

Introduction to Distributed System

Failures

Learn about the failures in distributed systems and the importance of independent vantage points.

We'll cover the following

- Introduction
- Types of failure in distributed systems
- Vantage points
  - Importance of independent service providers

Introduction#

Once in a while, we encounter the failure of a service that's a household name, and individuals and businesses react to them. As system designers, we might wonder how carefully designed services that have been perfected over years by experienced teams can also fail.

This chapter discusses some of the major failures of well-known services and the measures that can be taken to mitigate such failures.

The following two factors contribute to failures:

- **Diverse users**: Most services have a vibrant user community, and as their needs evolve, so do the software products. If a software doesn't update in the way it provides new features and services, it will become stable over time. However, it might not have the features customers want.

- **Complex systems**: Systems are complex, and they usually have emergent properties where the sum of system components is more complex than the individual pieces.

Diverse users interacting with a complex system

Types of failure in distributed systems#

Most modern services are designed in such a way that failures are contained, and others might be localized for some users. Let’s explore the types of failures we can observe in the distributed systems.

- **System failure**: A software or hardware failure is the most common cause of system failure. The contents in the primary memory are lost when a system fails. However, the data in secondary storage or replicas remains unaffected. The system reboots during such a breakdown.

- **Method failure**: Such failures suspend the working of distributed systems. It may also make the system execute the processes incorrectly or enter a deadlock state.

- **Communication medium failure**: Such failures occur when one component or service of a system can’t reach the other internal or external entities.

- **Secondary storage failure**: In such failures, the secondary storage or replicas are down. The data in these nodes becomes inaccessible, so primary nodes need to generate another replica to ensure reliability and fault tolerance.

Types of failure in a distributed system

Vantage points#

Something is always failing for large services. It’s preferred to have a graceful degradation so that only a small portion of users are impacted for a short time. Therefore, we need globally dispersed vantage points to independently see the service status.

> **Note:** Many independent services, such as Downdetector, exist for crowd-sourced problem reporting. It’s interesting to note that if we visit such a service and see the status of our favorite applications, we’ll see that there’s always someone in the world facing some service problem.

Importance of independent service providers#

One of the design goals of the original Internet was to provide resilience so that if one part fails, the rest can still operate.

With the emergence of a handful of service providers over the last decade, critics have raised concerns about such centralization and the potential impacts of failures. Most companies provide some kind of dashboard to enable users to see the service status. However, some failures might even knock out these dashboards. Affected companies then communicate to their customers via services like Twitter to announce the updates. Independent third-party services are valuable for failure detection and status dissemination.

> **Note:** The broader concept here is to use independent failure domains. A **failure domain** is a concept where anything failing inside a domain or network shouldn’t affect other components and services in other domains. At times, we say that two domains are independent if they’re outside the blast radius of each other.

Importance of independent service

In the following lessons, we’ll discuss the failures of well-known services by giant companies, the causes of failures, and what mitigation techniques can be used to avoid these failures. Although failures are a great way to learn what went wrong and what the original designers could do to avoid such failures, we’d like to prevent them from occurring at all.

Evaluation of Google Docs’ Design

Next

Facebook, WhatsApp, Instagram, Oc…

Mark as Completed

Report an Issue