Grokking Modern System Design Interview for Engineers & Managers
0% completed

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

Concluding Remarks

Course Certificate

Mark Course as Completed

Requirements of a Distributed Task Scheduler's Design

Learn about the functional and non-functional requirements of the task scheduler.

> **We'll cover the following**

- Requirements
  - Functional requirements
  - Non-functional requirements
- Building blocks we will use

Requirements#

Let's start by understanding the functional and non-functional requirements for designing a task scheduler.

Functional requirements#

The functional requirements of the distributed task scheduler are as follows:

- **Submit tasks**: The system should allow the users to submit their tasks for execution.

- **Allocate resources**: The system should be able to allocate the required resources to each task.

- **Remove tasks**: The system should allow the users to cancel the submitted tasks.

- **Monitor task execution**: The task execution should be adequately monitored and rescheduled if the task fails to execute.

- **Efficient resource utilization**: The resources (CPU and memory) must be used efficiently in terms of time, cost, and fairness. Efficiency means that we do not waste

resources. For example, if we allocate a heavy resource to a light task that can easily be executed on a cheap resource, it means that we have not efficiently utilized our resources. Fairness is all tenants' ability to get the resources with equally likely probability in a certain cost class.

- **Release resources**: After successfully executing a task, the system should take back the resources assigned to the task.

- **Show task status**: The system should show the users the current status of the task.

Functional requirements of the distributed task scheduler

Non-functional requirements#

The non-functional requirements of the distributed task scheduler are as follows:

- **Availability:** The system should be highly available to schedule and execute tasks.

- **Durability:** The tasks received by the system should be durable and should not be lost.

- **Scalability:** The system should be able to schedule and execute an ever-increasing number of tasks per day. Fault-tolerance: The system must be fault-tolerant by providing services uninterrupted despite faults in one or more of its components.

- **Bounded waiting time:** This is how long a task needs to wait before starting execution. We must not execute tasks much later than expected. Users shouldn't be kept on waiting for an infinite time. If the waiting time for users crosses a certain threshold, they should be notified.

Non-functional requirements of distributed task scheduler

So far in this lesson, we have learned about task schedulers in general and distinguished between centralized and distributed task schedulers. Lastly, we listed the requirements of the distributed task scheduler system.

Building blocks we will use#

We'll utilize the following building blocks in the design of our task scheduling system:

Building blocks of a task scheduler

- **Rate limiter** is required to limit the number of tasks so that our system is reliable.
- **A sequencer** is needed to uniquely identify tasks.
- **Database(s)** are used to store task information.
- **A distributed queue** is required to arrange tasks in the order of execution.
- **Monitoring** is essential to check the health of the resources and to detect failed tasks to provide reliable service to the users.

We've identified the requirements of the task scheduler. In the next lesson, we'll design our task scheduling system according to these requirements.

Back

System Design: The Distributed Task …

Next

Design of a Distributed Task Scheduler

Mark as Completed

Report an Issue