Ask a Question

System Design: Web Crawler

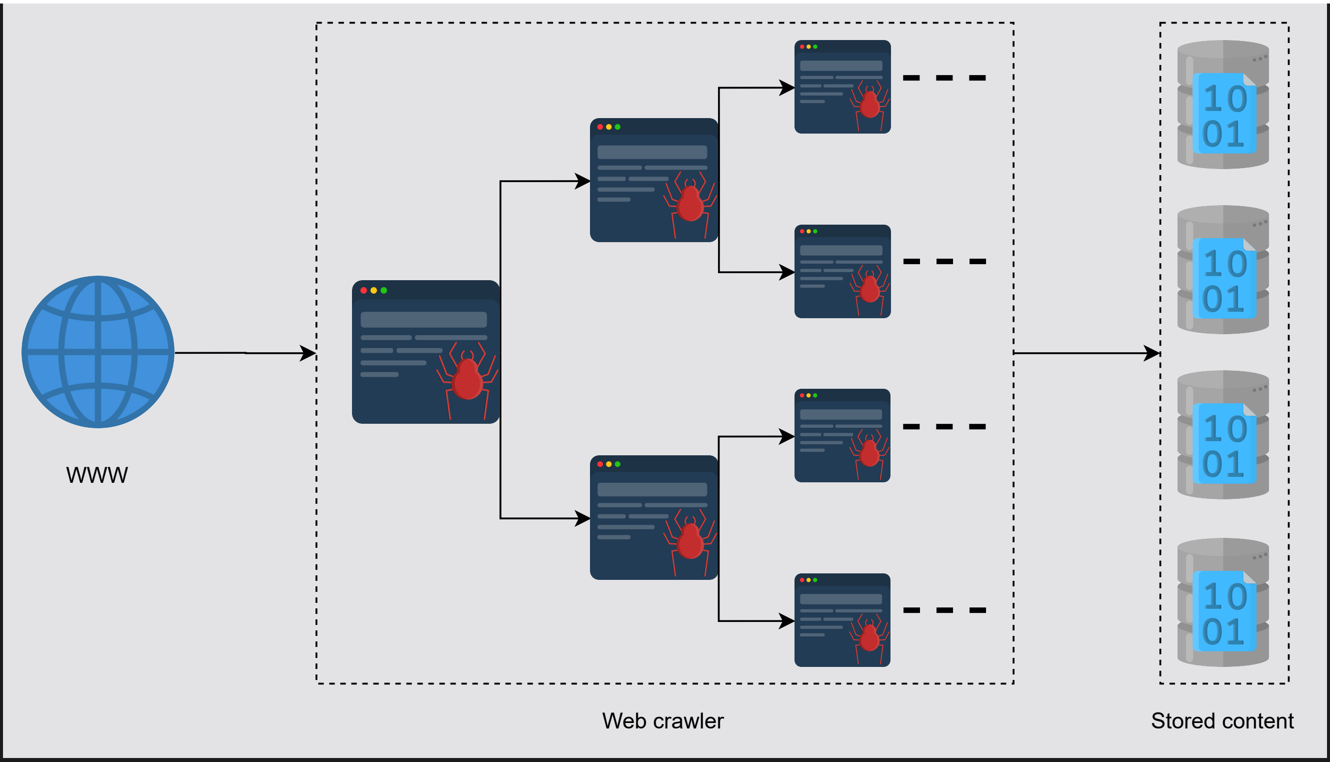
Learn about the web crawler service.

We'll cover the following

Introduction

A **web crawler** is an Internet bot that systematically scours the world wide web (WWW) for content, starting its operation from a pool of seed URLs. This process of acquiring content from the WWW is called **crawling**. It further saves the crawled content in the data stores. The process of efficiently saving data for subsequent use is called **storing**.

This is the first step that's performed by search engines; the stored data is used for indexing and ranking purposes. This specific design problem is limited to web crawlers and doesn't include explanations of the later stages of indexing and ranking in the search engines.



An overview of the web crawler system

educative

The additional utilities of a web crawler are as follows:

- **Web pages testing**: We use web crawlers to check the validity of the links and structures of web pages.
- **Web page monitoring**: We use web crawlers to monitor the content or structure updates on web pages.
- **Site mirroring**: Web crawlers are an effective way to mirror popular websites.
- **Copyright infringement check**: Web crawlers fetch content and check for copyright infringement issues.

In this chapter, we'll design a web crawler and evaluate how it fulfills the function and non-functional requirements.

The output of the crawling process is the data that's the input for the subsequent processing phases—data cleaning, indexing, page relevance using algorithms lik page ranks, and analytics. To learn about some of these subsequent stages, refer to our chapter on distributed search.

How will we design a Web crawler?

This chapter consists of four lessons that encompass the overall design of the web crawler system:

1. **Requirements**: This lesson enlists the functional and non-functional requirements of the system and estimates calculations for various system parameters.
2. **Design**: This lesson analyzes a bottom-up approach for a web-crawling service. We get a detailed overview of all the individual components leading to a combined operational mechanism to meet the requirements.
3. **Improvements**: This lesson provides all the design improvements required to counter shortcomings, especially the crawler traps. These crawler traps include links with query parameters, internal links redirection, links holding infinite calendar pages, links for dynamic content generation, and links containing cyclic directories.
4. **Evaluation**: This lesson provides an in-depth evaluation of our design choices to check if they meet all the standards and requirements we expect from our design.

Let’s begin with defining the requirements of a web crawler.