Multi-Threaded Web Crawler with File System Indexing

Deepshika Chand (2210110250)
Chahit Luthra (2210110236)
Aakrisht Narang (2210110097)

Project Description:

The project focuses on building a web crawler that fetches and indexes web pages using multiple threads to optimize performance. The crawler will implement threading and synchronization techniques to efficiently retrieve and store data while avoiding conflicts. Additionally, a basic file system will be designed to organize and manage the fetched web pages. This project provides hands-on experience in concurrent programming, synchronization mechanisms, and file system management, making it an advanced and practical learning experience.

The first step is to implement a basic single-threaded web crawler. This involves fetching web pages using HTTP requests, parsing the HTML content to extract URLs, and storing the retrieved data in a local file system.

Once the basic crawler is functional, it needs to be upgraded to a multi-threaded architecture. A thread pool will be implemented to allow concurrent fetching of web pages, while a queue will manage URLs to be crawled. Synchronization mechanisms will be introduced to ensure that multiple threads do not process the same URL, preventing duplication and race conditions.

To further enhance efficiency, a URL filtering and deduplication system will be established. This system will track visited URLs using a set or a database, ensuring that pages are not re-crawled unnecessarily. The crawler will also include restrictions based on allowed domains and depth limits to control the extent of its operations.

For data storage and indexing, a structured file system will be implemented. Crawled web pages will be organized by domain or content type, and a basic indexing mechanism will be developed. This can be achieved using file-based storage formats such as JSON, plain text, or HTML files, while an optional database integration with SQLite or MySQL can be considered for efficient search and retrieval.

Error handling and logging will be integrated to ensure robust performance. The crawler will be equipped with mechanisms to manage network failures, broken links, and timeouts. A logging system will record important events, aiding in debugging and performance monitoring.

To optimize performance, asynchronous request handling will be explored, reducing wait times for HTTP responses. Thread usage will be carefully managed to prevent system overloads, and caching techniques will be implemented to avoid redundant downloads of web pages.

Testing and debugging will be conducted by running the crawler on a small test domain, ensuring proper functionality and fixing any concurrency-related issues. Data storage and indexing mechanisms will be verified to confirm that fetched content is correctly organized and accessible.

If time permits we would like to develop a basic front-end interface to display and manage the crawled content.