An Implementation of a Crawler for Gutenberg Project

Jose Juan Hernández Gálvez [1], David Cruz Sánchez [2],
Jorge Hernández Hernández [3], Juan Carlos Santana Santana [4]

[1]jose.hernandez219@alu.ulpgc.es
[2]david.cruz107@alu.ulpgc.es
[3]jorge.hernandez12@alu.ulpgc.es
[4]juan.santana176@alu.ulpgc.es

November 2022

Abstract

Information Retrieval deals with searching and retrieving information within the documents and it also searches the online databases. Web crawler is defined as a program or software which traverses the Web and downloads web documents in an automated manner. The extraction of information from websites can be expensive in terms of processor time. Consequently, we proposed an optimal crawler design in order to effectively use idle computing resources and to avoid the need to employ dedicated equipment. We have developed our crawler to extract books from the Gutenberg Project, a website that collects books.[1]

Keywords— Crawler, Event Driven Architecture, Document, Gutenberg Project

1 Introduction

The World Wide Web (WWW) is internet client server architecture. It is a powerful system based on complete autonomy to the server for serving information available on the internet. The information is arranged as a large, distributed, and non-linear text system known as Hypertext Document system. These systems define part of a document as being hypertext- pieces of text or images which are linked to other documents via anchor tags. HTTP and HTML present a standard way of retrieving and presenting the hyperlinked documents. Internet browsers, use search engines to explore the servers for required pages of information. The pages send by the servers are processed at the client side. Beginning in 1990, World Wide Web has grown exponentially in size. As of today, it is estimated that it contains about 55 billion publicly index able web documents spread all over the world on thousands of servers [2].

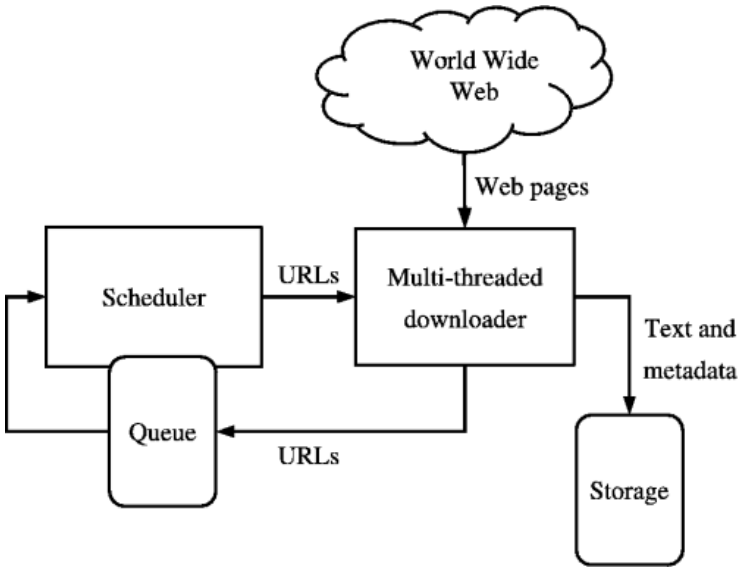

Figure 1: Flow of Web Crawler

The basic algorithm executed by any Web crawler, as shown in the figure 1, takes a list of seed URLs as its input and repeatedly executes the following steps. Remove a URL from the URL list,determine the IP address of its host name, download the corresponding document, and extract any links contained in it. For each of the extracted links, ensure that it is an absolute URL (derelativizing it if necessary), and add it to the list of URLs to download, provided it has not been encountered before. If desired, process the downloaded document in other ways (e.g., index its content) [3].

2 Methodology

In our project we have created two applications, a crawler to download different books from "The Gutenberg Project" website and an inverted index. The crawler will persist files and events in the datalake. The inverted index will be prevailed in disk as a datamart.

We have followed the SOLID principle to make the project maintainable and flexible. Also we wanted or code to be legible, so we decided to make our code clean. In addition, as we used git, we tried to apply the gitflow strategy.

To face the implementation of the crawler, we proposed an event driven architecture. An event-driven architecture uses events to trigger communication between decoupled services as it is common in modern applications built with microservices. An event-driven architecture consists of events producers that generate a stream of events, and event consumers. In this sprint we have created the base for this architecture, as we persist different files and events in our datalake. Also we have created a datamart, the inverted index itself. We hope that in the following iterations we will be able to deploy the full system.

In addition, we have created a bunch of tests to verify the good functionality of

both applications. Furthermore, we have been using TDD to figure out different regex expressions. These will be used by the crawler to deduce different book data, such as the author, the release date or the content itself.

3 Conclusions

The implemented crawler has proven itself capable of extracting book from the Gutenberg project website. This sprint is essential for our event-driven architecture, from the data we extract, many datamarts could be created, giving us a great growth potential. With our methodology, this code is prepared for changes, such as incorporating new websites or implementing a new format in our datalake.

In conclusion, this project is ideal to explore the opportunities that a clean code gives to the programmers, such as the scalability and flexibility.

4 Future Work

Many different adaptations, tests, and experiments are left due to lack of time, but right now our main concerns are:

- Stop depending on untrustworthy third party libraries such us the Unirest.
- Create more tests to be prepared for future major changes.
- Take the full benefit from the different datamarts created.

Additionally, since we proposed an event driven architecture, we must define some consumers, for instance, a listener that consumes from the download events to update the inverted index.

References

[1] https://www.gutenberg.org

[2] Kausar, M., Dhaka, V. & Singh, S. Web crawler: a review. *International Journal Of Computer Applications*. **63** (2013)

[3] Heydon, A. & Najork, M. Mercator: A scalable, extensible web crawler. *World Wide Web*. **2**, 219-229 (1999)