A study about migrating from HTTrack to Apache Nutch considering the Math Information Retrieval

Caio M. M. Leite[1], **Eduardo S. Saccardo**[1], **Flávio B. Gonzaga**[1]

[1]Instituto de Ciências Exatas - Ciência da Computação
Universidade Federal de Alfenas (UNIFAL-MG)

{caio.leite,eduardo.saccardo}@sou.unifal-mg.edu.br

Abstract. *Web crawlers are essential for the functioning of search engines. This paper aims to analyze and effectively migrate an already implemented HTTrack crawler to Apache Nutch. Apache Nutch was adequately configured and connected to Apache Kafka, and its crawl data was then stored in a MongoDB database using a Springboot Application. The results show that four layers were necessary to index the entire domain, taking between 21 and 23 minutes, compared to HTTrack, which took almost 3 hours more. The use of Apache Nutch for crawling and Kafka for message queuing proved effective, as it enabled distributed and scalable data processing.*

1. Introduction

It is fair to say that Internet-based information retrieval would collapse if search engines were not available [Gordon and Pathak 1999]. Furthermore, Google alone had more than 86 billion accesses between July and August of 2022 [Similarweb 2022]. In that regard, search engines make optimal and target searches so that a user finds their goal, or at least comes close to it.

The use of search engines in today's digital age has become an integral aspect of internet-based information retrieval. Without search engines, the process of finding relevant and accurate information would be greatly hindered [Gordon and Pathak 1999]. Search engines make it possible for users to find the information they are looking for in a quick and efficient manner by sorting through vast amounts of data and presenting relevant results in a matter of seconds. In 2022, the high number of accesses to search engines, such as Google, further highlights the importance and reliance on these tools in today's society.

1.1. Web Crawler

Web crawling is the main technique used by search engines to store and provide insight. This technique is the process used by search engines to collect pages from the Web [Castillo 2005] and, as such, it can be referred to as systematically browsing the World Wide Web for Web indexing. Indexing is mapping a document with all the words on the page to organize each web page efficiently. Thus, when a user searches for something, the search engine matches each query word with the words of the indexed documents, returning the documents with greater relevance. A query can be composed of one or more words.

The Internet can be viewed as a directed graph, where web pages are considered nodes, and hyperlinks, edges. Thus the search operation may be summarized as a process of traversing directed graph [Kausar et al. 2013]. A crawler operates from an initial seed (a text file containing one or several URLs). Like so, the crawler enters all the hyperlinks of the visited pages, downloading and indexing them to be later processed by the search engine.

1.1.1. HTTrack and Apache Nutch

HTTrack is a free and open-source web crawler that also serves as an offline browser [HTTrack 2023]. Developed by Xavier Roche and licensed by GNU(General Public License), HTTrack was initially released in 1998. Over the years, several scientific applications started to use this web crawler. For example, SearchOnMath's search engine[1] can be cited, which has been using HTTrack since 2008. The most recent and stable HTTrack version was released in 2017.

Despite being robust and with many features, the HTTrack is a web crawler where the result of its execution will be a copy of the files written locally to disk. This means that if some additional actions need to be performed on the downloaded files, like inserting them into a database, one or more software must be developed to perform such actions.

In this sense, there are more modern alternatives, such as Apache Nutch.Apache Nutch is a highly extensible and scalable open-source web crawler software project. [Nutch 2023]. According to Yadav and Goyal [Yadav and Goyal 2015], Nutch excels in scalability, as it is dynamically scalable and fault-tolerant through Hadoop. Moreover, it is flexible with its plugin system that can be integrated into different software, such as Apache Kafka. This possibility means that the output of Apache Nutch can be automated so that different tasks are performed, for example, pre-processing and data insertion directly into a database. The fact of being possible to compose an execution pipeline with the Apache Nutch considerably reduces the need to develop more software.

1.2. Objectives

This work is an initial study to verify the possibility of migrating the SearchOnMath [Gonzaga 2022] web-crawler infrastructure from the HTTrack to Apache Nutch. The specifics goals are:

- Index pages containing mathematical formulas using Apache Nutch.
- Develop a Spring Boot application to integrate the Apache Kafka messaging service with Apache Nutch.
- Compare the whole process, using the Apache Nutch, with the current SearchOnMath's crawling approach (based on the HTTrack).

2. Related Work

Previous research in this area revealed a scarcity of literature on techniques for prioritizing and indexing web pages containing mathematical content. This section presents some recent work using HTTrack or Apache Nutch.

The paper [Sharma and Tribhuvan 2021] describes a web crawler named "Smart Crawler" and compares its performance with that of a widely used web crawler, HTTrack.

[1] `https://www.searchonmath.com`

The experimental evaluation was performed by crawling five selected domains and experimental findings indicate the efficacy of the smart crawler, which is higher than other crawlers in minimum running time.

Furthermore, [Lopez et al. 2015] discusses how to optimize the open-source web crawler Apache Nutch for specific domain crawling at a large scale. The paper covers techniques for improving the efficiency and performance of Nutch, such as configuring the software for specific types of content or using distributed computing to handle a large number of pages. It also discusses how to handle the specific challenges that come with crawling at a large scale, such as dealing with large amounts of data or managing the resources required to crawl a large number of pages.

Additionally, [Donovan 2021] delves into the use of web crawlers to search and retrieve relevant documents related to health policy automatically. The authors explain that a web crawler can be helpful in this context because it can save time and effort compared to manually searching for information. However, they also note that a web crawler's results may not always be comprehensive or accurate and that it is essential to evaluate the quality of the documents retrieved. The article describes the development and implementation of a web crawler designed explicitly for health policy documents and the results of its use. Additionally, it can be used to track the changes in the policies over time and evaluate the impact of the guidelines on public health.

The article [Tan and Lan 2019] goes over the analysis and improvement of the Chinese index technology used in the open-source search engine Nutch. It suggests methods for improving the accuracy and efficiency of the indexing process for Chinese language content. The paper examines the current limitations of Nutch's Chinese indexing technology, specifically in handling Chinese characters and distinguishing between different forms of a Chinese word. It then presents several proposed improvements to address these limitations, such as implementing a more advanced Chinese word segmentation algorithm and using additional resources like synonym dictionaries to improve the quality of the index. The article also suggests several experiments and tests to evaluate the effectiveness of these proposed improvements. Overall the article tries to improve the Chinese indexing technology of the Nutch search engine.

Also, [Jansson et al. 2020] investigates the use of web crawlers to create a web archive for Swedish trade unions. The authors tested several software programs including HTTrack before settling on NetarchiveSuite (NAS) and the data crawler Heritrix. However, the study's authors found that they had trouble finding a powerful indexer that worked well with HTTrack, which led them to opt for NetarchiveSuite and Heritrix instead.

Lastly, [Yu et al. 2020] studied the fundamental principles of web crawlers, including their various characteristics and classifications, as well as the technical strategies utilized in web crawlers, including webpage acquisition and analysis, data storage methods, and web search strategies.

3. Methodology

As previously mentioned, this work verifies the viability of migrating SearchOnMath's web crawler infrastructure from the HTTrack to Apache Nutch. This section presents a description of the tools as well as the proposed activities.

3.1. HTTrack use case by SearchOnMath

SearchOnMath indexes websites of different types, like Digital Libraries, Forums, and Scientific Papers repositories. Some of its websites make their content available through dump files (generated from databases), being not necessary to use web crawlers. This is the case for the following domains:

- MathOverflow: `mathoverflow.net`
- StackExchange websites:
  - Computational Science: `scicomp.stackexchange.com`
  - Computer Science: `cs.stackexchange.com`
  - Engineering: `engineering.stackexchange.com`
  - Cross Validated: `stats.stackexchange.com`
  - Artificial Intelligence: `ai.stackexchange.com`
  - Quantum Computing: `quantumcomputing.stackexchange.com`
  - Internet of Things: `iot.stackexchange.com`
  - Mathematics Educators: `matheducators.stackexchange.com`
  - Mathematics: `math.stackexchange.com`
  - Physics: `physics.stackexchange.com`
- Wikipedia (English version): `en.wikipedia.org/wiki/Main_Page`

Some of the SearchOnMath indexed websites, however, do not provide their content in the dump file format. For these websites, SearchOnMath uses HTTrack to download the web pages. This is the case for the following domains:

- NIST DLMF: `dlmf.nist.gov`
- nLab: `ncatlab.org`
- Physics Forums: `physicsforums.com`
- PlanetMath: `planetmath.org`
- Socratic: `socratic.org`
- Wolfram MathWorld: `mathworld.wolfram.com`

HTTrack comes with a lot of parameters that can be tuned. A complete list is available on its official website[2]. The parameters used by SearchOnMath in most cases are described as follows:

- -p1: save only html files
- -B: can both go up&down into the directory structure
- -K3: absolute URI links
- -%u: various hacks to limit duplicate URLs (strip //, www.foo.com==foo.com..)
- -#L20 000 000: maximum number of links
- -A1 250 000: maximum transfer rate in bytes/seconds

3.2. Apache Nutch viability study

In this work, both HTTrack and Apache Nutch were configured to fetch the NIST DLMF (Digital Library of Mathematical Functions). The reason for choosing DLMF as the website to be downloaded is due to its size. It is the smallest website considering the set of websites indexed by the SearchOnMath. Since this work aims to perform an initial study

[2]`https://www.httrack.com/html/httrack.man.html`, accessed on 02/13/2023.

to migrate SearchOnMath's web crawler infrastructure from the HTTrack to the Apache Nutch, working with a small website is a good starting point because possible errors can be detected and fixed faster.

The pipeline comprises Apache Nutch, which will fetch the data from DLMF; Apache Kafka, where the downloaded data will be recorded (to be consumed later); a SpringBoot application, which will consume the data from Apache Kafka; process it and persist it into MongoDB.

The following sections offer more details about Apache Kafka and MongoDB.

3.3. Apache Kafka

Apache Kafka is an open-source distributed streaming system for stream processing, real-time data pipelines, and large-scale data integration. According to Confluent [Confluent 2022], one of the prominent companies that maintain the project, Apache Kafka, consists of a storage layer and a compute layer that combines efficient, real-time data ingestion, streaming data pipelines, and storage across distributed systems. In short, this enables simplified data streaming between Kafka and external systems. Figure 1 and Figure 2 shows the elements of a Kafka cluster.

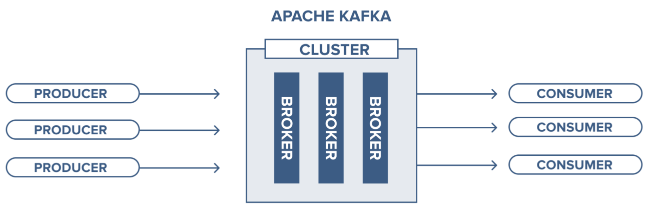


Figure 1. Representation of a Kafka cluster[3].

- **Kafka Clusters:** A Kafka cluster is an ensemble of multiple Kafka brokers
- **Broker:** A broker may be analog to a server and each broker is identified with its ID.

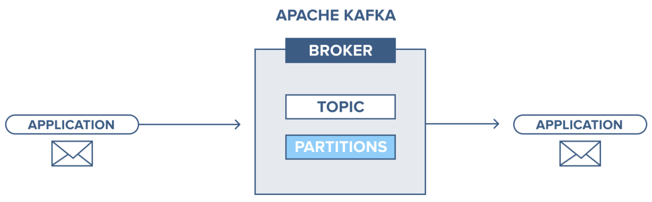


Figure 2. Representation of a Kafka Broker[4].

- **Topics:** Kafka topics are a particular stream of data within the Kafka cluster. A topic is similar to a table in a database but without all the constraints because

[3] `cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html`, accessed on 02/14/2023.

[4] `cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html`, accessed on 02/14/2023.

it is allowed to send anything to a Kafka topic. Moreover, a topic supports any message format, for example, Text file, Binary, and JSON. A data stream is a sequence of messages. It is not possible to query topics. Instead, it is used by Kafka producers to send data and Kafka Consumers to read the data. In addition, Kafka topics are immutable, which means that once data is written to a partition, it cannot be changed
- **Partition:** Topics are split into partitions to facilitate scalability. Messages within each partition are ordered.
- **Off-set:** offset only has meaning for a specific partition. It may be analog to an array in C for example, in this case, partitions are different arrays, and offsets are the indexes of these arrays. Equal offsets may not reflect equal results as they are different partitions and therefore have different content.

A producer can publish messages on a topic. The published messages are then stored at a set of servers called brokers. A consumer can subscribe to one or more topics from the brokers and consume the subscribed messages by pulling data from the brokers [Kreps et al. 2011].

This work created a Kafka topic and designated Apache Nutch as the producer. During web crawling, Apache Nutch wrote the data to the Kafka topic. A Spring Boot application was developed to consume the data from the Kafka topic, with the application serving as a Kafka consumer.

3.4. MongoDB

MongoDB is an open-source, cross-platform, and distributed document-based database designed for ease of application development and scaling. It is a NoSQL database [TutorialsTeacher 2022]. The primary advantage of NoSQL databases is that, unlike relational databases, they handle unstructured data such as documents, e-mail, multimedia, and social media efficiently [Khan and Mane 2013].

The next section describes in detail how the environment was configured to fetch and persist the NIST DLMF website data using the proposed pipeline with Apache Nutch.

4. Results

4.1. Environment Configuration

This project utilized the following software versions:

- **Apache Nutch:** version 1.18
- **MongoDB:** version 4.2.23
- **Apache Kafka:** version 2.13-3.2.3
- **Spring Boot:** 2.7.4 (with OpenJDK version 11.0.17)

The configuration details for each of the software used will be presented below.

4.1.1. Apache Nutch

Considering Apache Nutch, it was necessary to change four configuration files, listed below:

- *index-writers.xml*: connect Nutch to different indexers (Kafka in this case).
- *regex-urlfilter.txt*: filter which URLs to crawl based on regex.
- *nutch-site.xml*: override *nutch-default.xml* configurations.
- *seed.txt*: set up which domains to crawl.

The location of each file within the Apache Nutch directory tree is shown in Figure 3, followed by the description of what was modified on each file.

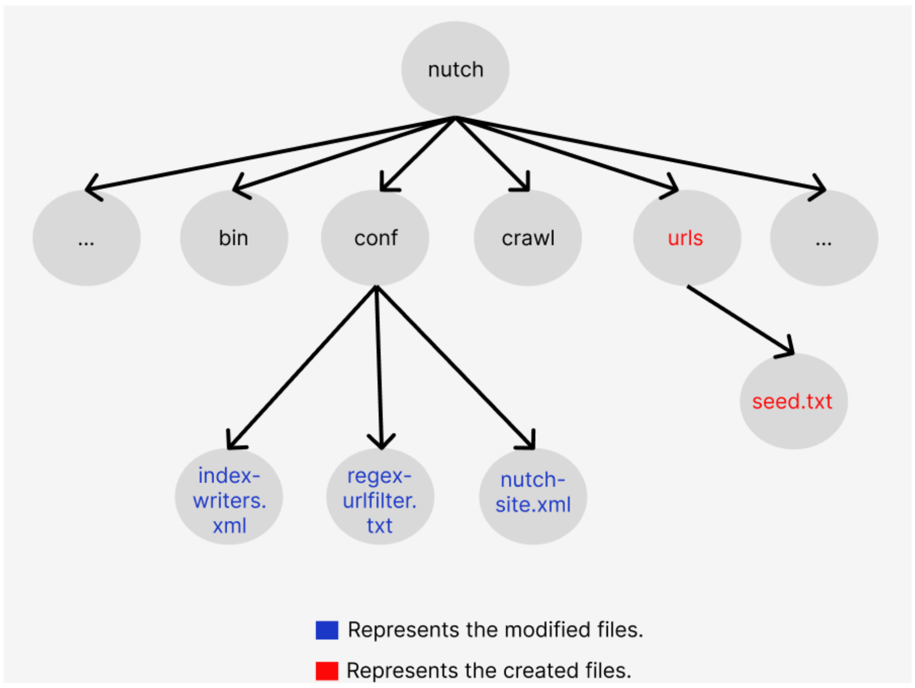


Figure 3. Nutch hierarchical file system. Source: own elaboration.

File *index-writers.xml*

In order to connect Apache Kafka to Apache Nutch, the value of the host, port, and topic must be defined. In this case, the host, port, and topic were defined as follows:

```
<writers ...>
   <writer ...>
       <parameters ...>
           <param name="host" value="localhost" />
           <param name="port" value="9092" />
           <param name="topic" value="main-topic" />
       </parameters>
        ...
   </writer>
</writers>
```

Listing 1. Code for the modified index-writers.xml. Source: own elaboration.

File *nutch-site.xml*

To change a property from Apache Nutch *conf/nutch-default.xml* file, it is necessary to overwrite them in the *nutch-site.xml* file. The description for each property is located in Appendix A. For instance, this project's *nutch-site.xml* file is as follows:

name	value
plugin.includes	protocol-httpclient|urlfilter-(regex|validator)| parse-(html|tika)| index-(basic|anchor)| indexer-(kafka)| scoring-opic| urlnormalizer-(pass|regex|basic)
fetcher.max.crawl.delay	-1
fetcher.server.delay	1.0
http.agent.name	Apache Nutch
http.agent.version	1.18
http.agent.description	Nutchup for DLMF
http.useHttp2	true
http.redirect.max	-1
http.content.limit	-1
db.ignore.external.links	true
ftp.keep.connection	true
indexer.max.title.length	-1
file.content.limit	-1

Table 1. Modified Nutch properties

File *regex-urlfilter.txt*

To restrict the crawl to a specific domain, it is required to configure the *conf/regex-urlfilter.txt* file. As an example for crawling on the DLMF domain that contains a digit on the subdirectory part of the URL (example: https://dlmf.nist.gov/**20**), the following regex was added at the end of the file:

```
    ##### DLMF RULES #####
# dlmf: seeds=[http://dlmf.nist.gov/]

# ok paths
+^https?://dlmf\.nist\.gov/\d*(\.\d+)?$

# reject anything else
-.
```

Listing 2. RegEx filter for DLMF domain. Source: own elaboration.

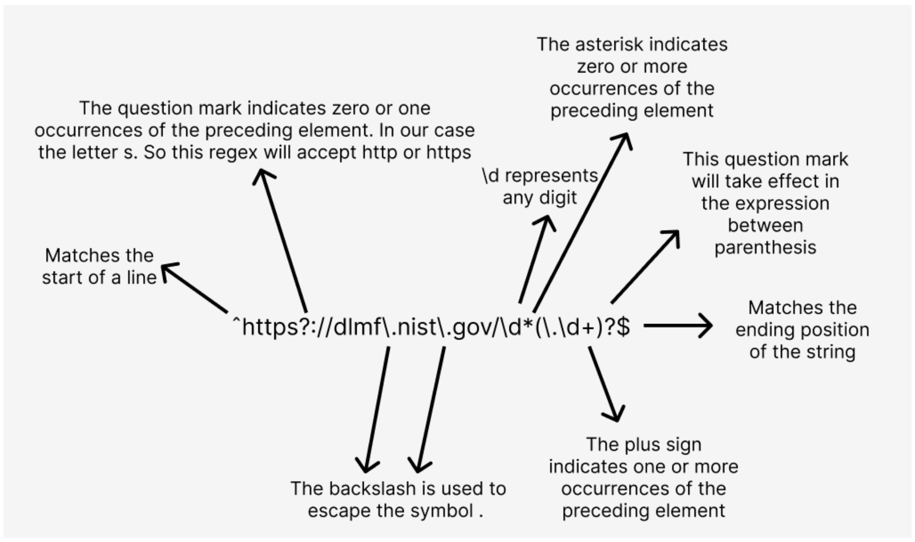

The asterisk indicates zero or more occurrences of the preceding element
The question mark indicates zero or one occurrences of the preceding element. In our case the letter s. So this regex will accept http or https
\d represents any digit
This question mark will take effect in the expression between parenthesis
Matches the start of a line
^https?://dlmf\.nist\.gov/\d*(\.\d+)?$
Matches the ending position of the string
The plus sign indicates one or more occurrences of the preceding element
The backslash is used to escape the symbol .

Figure 4. Regex explanation. Source: own elaboration

File *seed.txt*

To specify which domains you want to crawl, create a folder named *urls* with a file named *seed.txt* inside. In this example, the file contains only one line: `http://dlmf.nist.gov`.

Crawl output format

Page [
id=["https://dlmf.nist.gov/"],
search=["DLMF: NIST Digital Library of Mathematical Functions"],
tstamp=["Wed Jan 25 18:13:06 BRT 2023"],
segment=["20230125181245"],
digest=["1c52bfb7d156579baf02b5631d556e46"],
host=["dlmf.nist.gov"],
boost=["0.0"],
title=["DLMF: NIST Digital Library of Mathematical Functions"],
url=["https://dlmf.nist.gov/"],
content=["DLMF: NIST Digital Library of Mathematical Functions\nDLMF\nIndex\nNotations\nSearch\nHelp?\nCiting\nCustomize\nAbout the Project\nNIST Digital Library of Mathematical Functions\nProject News\n2022-12-15 DLMF Update; Version 1.1.8; MathML improvements\n2022-12-15 Richard B. Paris, Associate Editor of the DLMF, dies at age...]
]

4.1.2. MongoDB structure

The MongoDB used in this project consists of a single collection that includes multiple documents of a single type referred to as Pages. Each Page document is structured with the following fields: search, tstamp, segment, digest, host, boost, id, title, url, and content, which are in accordance with the output of an Apache Nutch crawl. The structure of the web pages being stored in the MongoDB database can be observed in the "Crawl output format" part.

4.1.3. Apache Kafka

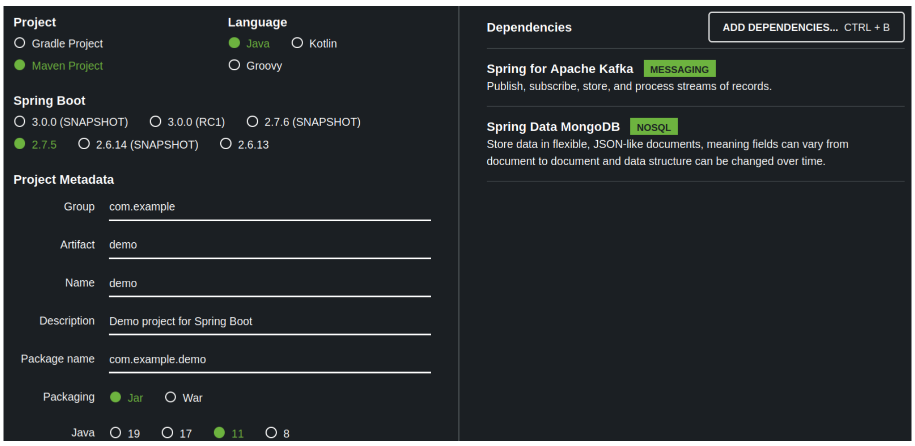
Considering that the main Apache Kafka folder is inside the *home* directory, the following commands will start the Kafka system.

Access the Kafka home directory:
cd ∼/kafka_2.13-3.2.3
Run zookeeper:
./bin/zookeeper-server-start.sh ./config/zookeeper.properties
Run kafka server:
./bin/kafka-server-start.sh ./config/server.properties

bin/crawl -i -s ./urls/ ./crawl/ (profundidade)

4.1.4. Spring Boot

A Spring Boot application through Spring initializr[5] with "Spring for Apache Kafa" and "Spring Data MongoDB" dependencies has been created, as shown in Figure 5.


Project
Gradle Project [✓ Maven Project]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (RC1) 2.7.6 (SNAPSHOT) [✓ 2.7.5] 2.6.14 (SNAPSHOT) 2.6.13

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	19 17 [✓ 11] 8

Dependencies

Spring for Apache Kafka MESSAGING
Publish, subscribe, store, and process streams of records.

Spring Data MongoDB NOSQL
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Figure 5. Spring Initializr example.

4.1.5. Kafka consumer through the spring application

Hereafter is the code for the service class that consumes the topic data and persists in the database. The other project classes are available in Appendix B.

```
package com.example.demo.service;

```

[5] https://start.spring.io/

```
 3 import org.apache.kafka.clients.admin.NewTopic;
 4 import org.json.JSONObject;
 5 import org.springframework.context.annotation.Bean;
 6 import org.springframework.kafka.annotation.KafkaListener;
 7 import org.springframework.kafka.config.TopicBuilder;
 8 import org.springframework.stereotype.Service;
 9
10 import com.example.demo.models.Page;
11 import com.example.demo.repository.MongoDBRepository;
12
13 @Service
14 public class ConsumerService {
15
16     private final MongoDBRepository pageRepository;
17     public ConsumerService(MongoDBRepository pageRepository) {
18       this.pageRepository = pageRepository;
19     }
20     @Bean
21     public NewTopic topic() {
22       return TopicBuilder.name("main-topic")
23         .partitions(0)
24         .replicas(1).build();
25     }
26
27     @KafkaListener(id = "myId", topics = "main-topic")
28     public void listen(JSONObject in) {
29       Page page = new Page();
30       page.setSearch(in.get("search").toString());
31       page.setTstamp(in.get("tstamp").toString());
32       page.setSegment(in.get("segment").toString());
33       page.setDigest(in.get("digest").toString());
34       page.setHost(in.get("host").toString());
35       page.setBoost(in.get("boost").toString());
36       page.setId(in.get("id").toString());
37       page.setTitle(in.get("title").toString());
38       page.setUrl(in.get("url").toString());
39       page.setContent(in.get("content").toString());
40       page.setBase64content(in.get("base64content").toString());
41       pageRepository.insert(page);
42     }
43 }
```

Listing 3. Code for Listening to Kafka Topic and Inserting Pages into MongoDB. Source: own elaboration.

In the code above, a Kafka topic named "main-topic" is created from lines 20 to 25. Moreover, the consumer is created from lines 27 to 42. The consumer fetches the topic data, transforms it into a JSON Object, and then persists in MongoDB.

4.2. Crawl command

Finally, to initiate the crawl process, the following command can be executed within the Apache Nutch home directory, after ensuring the initialization of the ZooKeeper, Kafka, and Spring Boot application:

bin/crawl -i -s ./urls/ ./crawl/ 1000

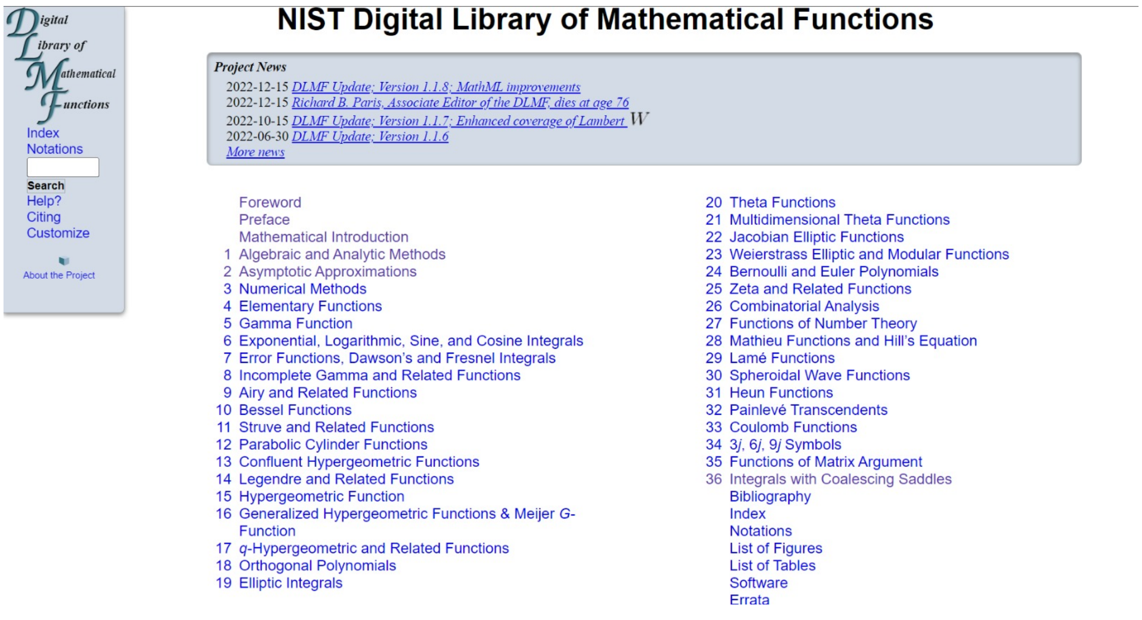
In which:

- **-i** - Indexes crawl results into a configured indexer.
- **-s** - Path to seeds file(s).
- **/urls/** - Directory containing the seed.txt file.
- **1000** - Number of iterations (depth).

4.3. Data analysis

Exploring the crawl's depth on "dmlf" and executing the crawl multiple times showed that four layers were necessary to index the entire domain. The first layer indexed one page, the second indexed 36 pages, the third indexed 856 pages, and the final one indexed 15 pages. The entire process took between 21 to 23 minutes.

Based on Figure 6, it can be exemplified that the first layer belongs to the website's homepage (crawler's seed). As mentioned, the second layer consists of 36 pages. As such, 36 topics can be seen in the figure, corresponding to the links that the nutch will pick up. It is noteworthy that Nutch exclusively considers URLs that conclude with digits due to the regex filter employed in its crawling mechanism for this particular website. Consequently, it refrains from exploring other hyperlinks. To illustrate, upon clicking the hyperlink "1 Algebraic and Analytic Methods", the user is directed to "`https://dlmf.nist.gov/1`", which aligns with the previously-mentioned restriction.



NIST Digital Library of Mathematical Functions

Project News
2022-12-15 *DLMF Update; Version 1.1.8; MathML improvements*
2022-12-15 *Richard B. Paris, Associate Editor of the DLMF, dies at age 76*
2022-10-15 *DLMF Update; Version 1.1.7; Enhanced coverage of Lambert* W
2022-06-30 *DLMF Update; Version 1.1.6*
More news

Foreword
Preface
Mathematical Introduction
1 Algebraic and Analytic Methods
2 Asymptotic Approximations
3 Numerical Methods
4 Elementary Functions
5 Gamma Function
6 Exponential, Logarithmic, Sine, and Cosine Integrals
7 Error Functions, Dawson's and Fresnel Integrals
8 Incomplete Gamma and Related Functions
9 Airy and Related Functions
10 Bessel Functions
11 Struve and Related Functions
12 Parabolic Cylinder Functions
13 Confluent Hypergeometric Functions
14 Legendre and Related Functions
15 Hypergeometric Function
16 Generalized Hypergeometric Functions & Meijer G-Function
17 q-Hypergeometric and Related Functions
18 Orthogonal Polynomials
19 Elliptic Integrals
20 Theta Functions
21 Multidimensional Theta Functions
22 Jacobian Elliptic Functions
23 Weierstrass Elliptic and Modular Functions
24 Bernoulli and Euler Polynomials
25 Zeta and Related Functions
26 Combinatorial Analysis
27 Functions of Number Theory
28 Mathieu Functions and Hill's Equation
29 Lamé Functions
30 Spheroidal Wave Functions
31 Heun Functions
32 Painlevé Transcendents
33 Coulomb Functions
34 $3j$, $6j$, $9j$ Symbols
35 Functions of Matrix Argument
36 Integrals with Coalescing Saddles
Bibliography
Index
Notations
List of Figures
List of Tables
Software
Errata

Figure 6. DLMF: NIST home page

Executing the crawl process with HTTrack on "dlmf" took 3 hours and 17 minutes. During the crawling execution, it was possible to verify that the HTTrack checked links pointing to non-html files (like formulas). However, only html files were downloaded (as expected). The HTTrack execution was based on the following parameters:

httrack https://dlmf.nist.gov/ -dlmf.nist.gov/bib/* -dlmf.nist.gov/about/* -*#* -*?* -*.php* -p1 -B -K3 -%u -L20000000 -F “Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.1) Gecko/20020826” -A1250000 -*.mag -*.viz -*.webgl -*.pmml -*.x3d

Where:

- https://dlmf.nist.gov/ - site to be downloaded
- -F - browser identity
- don’t crawl URLs matching the following patterns:
  - -dlmf.nist.gov/bib/*
  - -dlmf.nist.gov/about/*
  - -*#*
  - -*?*
  - -*.php*
  - -*.mag
  - -*.viz
  - -*.webgl
  - -*.pmml
  - -*.x3d

The rest of the parameters (like -p1) was explained in Section 3.1.

5. Conclusion

In conclusion, Apache Nutch is a powerful tool for web crawling and indexing, providing a scalable and efficient solution for data acquisition. Its integration with Apache Kafka and MongoDB allows for a distributed and fault-tolerant architecture that can handle large amounts of data. Moreover, Apache Nutch has the ability to scale and provide more advanced data analysis.

HTTrack, on the other hand, is a more straightforward tool that is easy to use and requires minimal setup. It is a good choice for small to medium-sized web crawling projects and does not require extensive technical knowledge. However, it lacks the advanced data analysis capabilities of Apache Nutch and may not be suitable for more complex projects.

In summary, when crawling the DLMF domain, the Apache Nutch demonstrated better results than the Httrack crawler used by SearchOnMath. For future works, we recommend the complete migration from the HTTrack to Apache Nutch considering the domains indexed by SearchOnMath.

Appendix A: *nutch-site.xml* properties explanation

Most of the explanations in this appendix were taken from the *nutch.default.xml* file.

- **plugin.includes:** Regular expression naming plugin directory names to include. Any plugin not matching this expression is excluded. Solr indexer was replaced by the Kafka indexer.

- **fetcher.max.crawl.delay:** If the Crawl-Delay in robots.txt is set to greater than this value (in seconds) then the fetcher will skip this page, generating an error report. If set to -1 the fetcher will never skip such pages and will wait the amount of time retrieved from robots.txt Crawl-Delay, however long that might be.

- **fetcher.server.delay:** The number of seconds the fetcher will delay between successive requests to the same server. Note that this might get overridden by a Crawl-Delay from a robots.txt and is used ONLY if fetcher.threads.per.queue is set to 1.

- **http.agent.name:** This field must not be empty, as this is the HTTP 'User-Agent' request header.

- **http.agent.version:** A version string to advertise in the User-Agent header.

- **http.agent.description:** Further description of our bot- this text is used in the User-Agent header. It appears in parenthesis after the agent name.

- **http.useHttp2:** If true try HTTP/2 and fall-back to HTTP/1.1 if HTTP/2 not supported.

- **http.redirect.max:** The maximum number of redirects the fetcher will follow when trying to fetch a page. If set to negative or 0, fetcher won't immediately follow redirected URLs, instead it will record them for later fetching.

- **http.content.limit:** The length limit for downloaded content using the http/https protocols, in bytes. If this value is non-negative (≥ 0), content longer than it will be truncated; otherwise, no truncation at all.

- **db.ignore.external.links:** If true, outlinks leading from a page to external hosts or domain will be ignored. This is an effective way to limit the crawl to include only initially injected hosts or domains, without creating complex URLFilters.

- **ftp.keep.connection:** Whether to keep ftp connection. Useful if crawling same host again and again. When set to true, it avoids connection, login and dir list parser setup for subsequent URLs. If it is set to true, however, you must make sure (roughly):
  (1) ftp.timeout is less than ftp.server.timeout
  (2) ftp.timeout is larger than (fetcher.threads.fetch * fetcher.server.delay)
  Otherwise there will be too many "delete client because idled too long" messages in thread logs.

- **indexer.max.title.length:** The maximum number of characters of a title that are indexed. A value of -1 disables this check.

- **file.content.limit:** The length limit for downloaded content using the file://protocol, in bytes. If this value is non-negative (≥ 0), content longer than it will be truncated; otherwise, no truncation at all.

Appendix B: project classes

Main.java

```java
package com.example.demo;
import org.apache.kafka.clients.admin.NewTopic;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.
   SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.kafka.config.TopicBuilder;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Repository.java

```java
package com.example.demo.repository;

import org.springframework.data.mongodb.repository.
   MongoRepository;
import org.springframework.stereotype.Repository;

import com.example.demo.models.Page;

@Repository
public interface MongoDBRepository extends MongoRepository<Page,
    String> {
}
```

The Lombok library was employed to mitigate the need for repetitive and verbose code. The integration of Lombok into the project, facilitated by utilizing Maven as the build tool, involved including the corresponding Lombok dependency within the project's pom.xml file.

Lombok dependency

```
<dependencies>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.26</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Page.java

```
package com.example.demo.models;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.mongodb.core.mapping.Field;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@Document(collection = "pages")
public class Page {
  @Id
  private String id;

  @Field("search")
  private String search;

  @Field("tstamp")
  private String tstamp;

  @Field("segment")
  private String segment;

  @Field("digest")
  private String digest;

  @Field("host")
  private String host;

  @Field("boost")
  private String boost;

  @Field("title")
  private String title;

  @Field("url")
  private String url;

  @Field("content")
  private String content;

}
```

References

Castillo, C. (2005). Effective web crawling. page 55–56. Association for Computing Machinery.

Confluent (2022). What is kafka? `https://www.confluent.io/what-is-apache-kafka`. Accessed: 2022-08-18.

Donovan, M. (2021). A web crawler for automated document retrieval in health policy.

Gonzaga, F. B. (2022). Searchonmath. `https://www.searchonmath.com/`. Accessed: 2022-08-18.

Gordon, M. and Pathak, P. (1999). Finding information on the world wide web: the retrieval effectiveness of search engines. volume 35, pages 141–180.

HTTrack (2023). `https://www.httrack.com/html/fcguide.html`. Accessed: 2023-02-15.

Jansson, J., Uba, K., and Karo, J. (2020). Labor gone digital (digifacket)! experiences from creating a web archive for swedish trade unions. volume 7, page 19.

Kausar, M. A., Dhaka, V., and Singh, S. K. (2013). Web crawler: a review. volume 63. Foundation of Computer Science.

Khan, S. and Mane, V. (2013). Sql support over mongodb using metadata. volume 3, pages 1–5. Citeseer.

Kreps, J., Narkhede, N., Rao, J., et al. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, pages 1–7.

Lopez, L. A., Duerr, R., and Khalsa, S. J. S. (2015). Optimizing apache nutch for domain specific crawling at large scale. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1967–1971.

Nutch, A. (2023). `https://cwiki.apache.org/confluence/display/nutch`. Accessed: 2023-02-15.

Sharma, M. and Tribhuvan, P. P. (2021). Smart crawler. volume 12.

Similarweb (2022). `https://www.similarweb.com/pt/website/google.com/#overview`. Accessed: 2022-08-17.

Tan, M. and Lan, H. (2019). Analysis and improvement of chinese index technology of open source search engine nutch. In *Journal of Physics: Conference Series*, volume 1176, page 022032. IOP Publishing.

TutorialsTeacher (2022). What is mongodb? `https://www.tutorialsteacher.com/mongodb/what-is-mongodb`. Accessed: 2022-08-18.

Yadav, M. and Goyal, N. (2015). Comparison of open source crawlers-a review. volume 6, pages 1544–1551.

Yu, L., Li, Y., Zeng, Q., Sun, Y., Bian, Y., and He, W. (2020). Summary of web crawler technology research. In *Journal of Physics: Conference Series*, volume 1449, page 012036. IOP Publishing.