

[CSC 583-Midterm essay

Design architecture of web search engine



October 25, 2022

Name: Muhammad Zaid Kamil; Student ID: 211890395

1. **Introduce the context and background of architecture design**

**Abstract**

The essay reports on the design of the architecture for a web search engine. The goal of the web search engine is to return correct and precise results based on the user’s search request. The two main architecture patterns represented in this essay are the shared data pattern which is the central repository, which is able to add, update, delete and refresh data in the repository. The second architecture patter is the map reduce pattern which first use the map function to filter the key words and process the key-values then the reduce function to aggregate and search the key values which the output data format is then stored in the repository. From the repository the data is sent to the indexer which ranks the index and displayed to the user. The main quality attributes considered in this study are performance, availability, and modifiability.

**Introduction**

Web search has become a part of our daily life. It was just yesterday, I was searching in Google “How to Elon Musk” People, then in seconds the search results were displayed from Wikipedia to Forbes all different resources were accessible just by one click. Same as Google, there are number of different popular web search engines which include: Yahoo, Bing, DuckDuckGo etc.… which allows users to search and navigate daily tasks.

In our previous project part 1 we were required to design a KWIC search engine which searches based on the keywords. The architecture patterns we used were Pipe and Filter pattern, MVC pattern, and Client- Server pattern. [1] Regardless of the architecture the basics of the functionality of a web search engine contains the following main components:

The search engine is made up of a few essential modules, as stated below:

* Crawler
* Indexer
* Repository (Database)
* User interface
* Query parser
* Ranker

Crawler:

Web crawlers work by visiting each specified website address then getting and storing all information contained in the website.

Extract the data from webpage URLs. Based on the user keywords, the crawler crawls through the billion webpages in the Internet. Based on our requirements, we can design the crawler’s crawl rate to extract URL from webpages. For example, we can set the crawler at a crawl rate of 5 pages per second. The Web Crawler collects the URLs text and metadata through the internet which will be stored in the shared data web repository.

A search engine's main component is collecting information from different web pages based on topics or keywords. This collected information or URL from the webpage is then given input to an indexer.

Index:

Index works similar to finding an index in a reading book. For example if we want to search the word “Apple” in a cooking book we go to the back of the book and check the Apples index and retrieve the page number (location) of the word inside the book.

When it comes to search engine, the Index main function is to index the webpage URLs given by the crawler frontier. The data from the webpage is also extracted during the indexing process. The data includes the Title from the webpage, Text of webpage, Images, time. [2] The index data is stored in text files.

* Indexer extracts the word and point to location of webpage. For example ‘book’ 00> [[3,[4,6],7,[7,8]. Webpage number 3. Location of words 4 and 6. [2]

Repository:

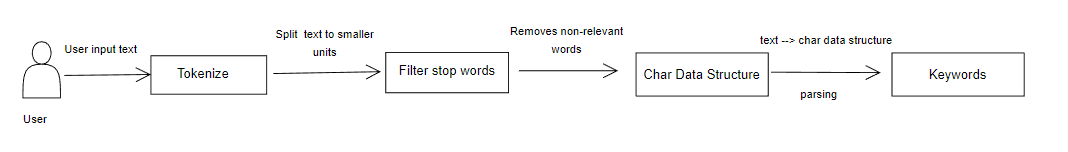
This is a shared data pattern which stores the URLs from Crawler, indexed URLs, key values from architecture patterns.

User Interface:

The UI is the search toolbar where the user input the search text to request for search.

Query Parser:

This takes input from the user through the user interface, and keywords are obtained with processes like stemming, stop-words and punctuation removal, which is then used to search the repository. Query parser simplifies the searching and helps in the effective retrieval of information.



**Figure 1: Process of Query parser**

*Tokenizer and Filter Principle in search query :*

User: Example user input search text: “I need to know how to develop a web search engine”

* *Tokenize:* Split the input user input text into smaller units.

Tokenized text with stop words:

[‘I’, ‘need’, ‘to’, ‘know’, ‘how’, ‘to’, ‘develop’, ‘a’, ‘ web’, ‘search’, ‘engine’, ‘.’]

* *Filter stop words*:

Tokenized text filtered stop words:

[‘develop’, ‘a’, ‘ web’, ‘search’, ‘engine’]

We can see the stop words in this text includes 50% of the user input text.

Changes the input text file as characters data structure.

* By parsing through the character data structure input. Will be able to recognize the set of characters that forms words which form the keywords which are sent to the crawler to retrieve the URLs based on the keywords.

*Early Decision making for web search architecture design*

In our early-stage design of the web search engine. We can brainstorm questions. This will include the direction of our decision for the architecture design of the web search engine. [3]

Questions for our early-stage design include:

* What type of architecture patterns can we include in the search engine?
* Functional requirements of the search engine?
* Which IDE and template will the search engine be developed?
* What protocols the search engine can handle?

1. **Design the overall architecture**

Diagram

Description automatically generated

**Figure 2: Overall Architecture design**

1. **Sketch the repository (Shared-data) pattern that accomplishes the administration of the webpage base, such as the repository structure, repository refreshing and updates, addition to and delete from the repository, repository inquiry, etc.**

Diagram

Description automatically generated

**Figure 3: Shared-data architecture pattern**

* Initially, the URL frontier contains the seed set; as pages are fetched, the corresponding URLs are deleted from the URL frontier.
* DNS resolution module determines the web server on the specified URL to be fetched.
* Then Parser that extracts the text: Header, Images, text from the links of the web pages
* The URLs that are extracted are sent to the crawler frontier
* The URL links with text and metadata are stored in the shared data repository.
* This is done in a loop format which keeps updating the shared data repository with URL links.

[repository update] – The repository is constantly being updated with new URL links retrieved from the Crawler frontier

[repository addition] – The repository is being added with new key value pairs in output data format from reducer function in map reducer pattern.

[repository refresh] – The repository is constantly being refreshed with indexer based on the user query parser.

[repository inquiry] – The repository is constantly being inquired by the query parse when the user request a search.

1. **Sketch the map-reduce pattern by designing a map function and a reduce function that together provide a basic search engine capability.**

Diagram

Description automatically generated

**Figure 4: Map-Reduce Architecture design**

* Input Format: The Input format is a TextInputformat which is a text file of all the URLs from crawler. Reads the inputformat and forms key-value pairs.
* for our architecture is the KeyValueInputFormat(how input files are split). Class that provides functionality (Selects the file, defines the input partition, provides a record for reading file) Text format.
* Record reader converts the input file to key value pair values.
* Mapper: In the Mapper there is a record reader which convert input file to key value pairs. Multiple map instances which works on data. Process input records and generates key-value pairs
* Partitioner: Decide based on the key and values. Determine how many reducers instances should be required to work on map tasks.
* Merge: Merge result of key value pairs

Reducer()

* Shuffling and sorting: Data is sorted and shuffled and fed into the reducer. Aggregation becomes easier. [4]
* Reducer: Multiple reduce tasks based on partitioner. All the data is aggregated to give the final output. [4]
* Output Format: Which is in a format to be sent to the shared data repository.

1. **Present at least three quality attributes that your architecture addresses.**
2. Availability

The central repository in the shared data architecture pattern should be available at all times. Since the web search engine availability to the user is based on the repository availability. The downtime should be very less especially during the peak hours. If there is downtime it should only occur during non-peak hours. We should expect the repository to be available 95% of the times. Availability states how often the function fails. In this case how often the search fails. And how many times is the search engine acceptable to fail. The system failure means that the search engine provides no service to users

3.21 Concrete Scenario



Recover from fault: Update, reconnect

DETECT fault:

log, notify

Response

Repair time

95% availability

Communication channels

Response Measure

Artifact



Internal to system – Communication issue

Network connection unresponsive

Stimulus

Source

1. Performance

The key-value patterns obtained from map and reduce architecture pattern must be efficient. This will allow the user’s keywords to be searched more effectively. We should expect the map() and reduce() function to perform well. The testing requirement is to check the amount of stop words present in the output data format from the map and reduce pattern.



Keyword detection

Filter stop words

Key-value pairs

Response

Miss rate

System

Response Measure

Artifact



Users

Search request

Stimulus

Source

Accessibility is the design of products, gadgets, services, transportation, or surroundings so that individuals with impairments can use them. The notion of accessible design and accessible development practice provides both "direct access" (i.e., unaided) and "indirect access," which means compatibility with a person's assistive equipment (for example, computer screen readers).

Testability

In our architecture design, when we want to add a new module/feature to the crawler, it should be possible to identify the flaws of the new module and test its functionality in various circumstances.

How does the new module of the Crawler affect the old Shared data repository pattern?

Identifying the flaws in the previous crawler module by testing with various situations and determining whether the new redesigned system fits with the new aims.

This will aid in the addition of new changes.

1. **Identify tactics that ensure the quality attributes and augment the overall architecture.**

Tactic 1: Active Redundancy

* Hot spare. redundant repository system that is synchronous to the with the active repository database system. This allows recovery with minimum disruption to system.[1]
* If the crawler cannot connect with the repository system. The redundant reposirotry will be working. This allows recovery with minimum disruption to system.

Tactic 2: Coupling reduction:

The coupling refers to the interdependencies between the modules of repository database and Query parser. To avoid the effects of interdependencies in the repository database machines, we can form an intermediary machine

Tactic 3: Exception prevention

Prevent search engine system exception from occurring by masking a fault (hiding the defect)

Tactic 4: Reduce module size:

* For the given modules of search engine, we can split the module. Therefore, each module of the machine is subdivided into smaller modules of machine such as a module specifically to the query parser. This will reduce the cost for further changes

Tactic 5: Passive Redundancy

Warm spare Redundant repository in the shared database system only provide spare when periodic updates.

Tactic 6: Heartbeat

Task tracker periodically sends ‘heartbeat’ signal to the Jobtracker so as to notify him of the current state of the system.

1. **Analyze the advantages and disadvantages of your architecture.**

Table 1: Advantages and disadvantages of selected Architecture patterns

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Perform parallel processing which stores and distributes large data sets across lots of servers. [6] The processing power of the system can be improved with the addition of more servers. [6] | It is not flexible i.e. the MapReduce framework is rigid [7] |
| The programming model divides the tasks to allow the execution of the independent task in parallel which helps to run the program in much less time.[6] | Semantics are hidden inside the map and reduce functions, so it is difficult to maintain, extend and optimize them |
| Shared pattern allows for the fast execution of a command or a query.[8] | Shared pattern is distributed over a large number of geolocations can be susceptible to performance degradation due to excessive network traffic.[8] |

**References**

[1] Group4, "Project 01: Microminer Search Engine."

[2]Tomer Ben David. Design a Search Engine - Part 1. (Nov. 3, 2017). Accessed: Oct. 26, 2022. [Online Video]. Available: <https://www.youtube.com/watch?v=xqzyXicFrTA>

[3] Software Architecture in Practice. Pearson, 2015 p. 53-54

[4] Simplilearn. Mapreduce In Hadoop | MapReduce Explained | MapReduce Architecture | MapReduce Tutorial |Simplilearn. (Apr. 8, 2019). Accessed: Oct. 26, 2022. [Online Video]. Available: <https://www.youtube.com/watch?v=JZGtV278SvE>

[5] Searching the Web - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/General-search-engine-architecture\_fig1\_2523546 [accessed 26 Oct, 2022]

[6]"What is MapReduce? | Learn the Example and Advantages of MapReduce." EDUCBA. <https://www.educba.com/what-is-mapreduce/> (accessed Oct. 26, 2022).

[7] "Advantages and Disadvantages of MapReduce - CS-GY 9223-D: Programming for Big Data." Introduction - CS-GY 6083: Principles of Database Systems. <https://vikram-bajaj.gitbook.io/cs-gy-9223-d-programming-for-big-data/hadoop/advantages-and-disadvantages-of-mapreduce> (accessed Oct. 26, 2022).

[8] "The pros and cons of the Shared architecture pattern." Enable Architect. <https://www.redhat.com/architect/pros-and-cons-sharding> (accessed Oct. 26, 2022).