Assignment 5 Report

Map-Reduce With Cloud Functions

The Submitted files contains: Web UI Files, function zip files, script to download project Gutenberg files in batch and report.pdf
Programming languages used: Node JS, JavaScript, HTML, CSS
Link for web UI: https://avinashpawar.ml/A5-ECC/index.html

Design Details.
Design of script and Implementation
The simple design of the system is very elegant one.
The Architecture can be divided into 4 simple modules of:

1. **Mapper Function**
   The Mapper function Takes the input of book links, then downloads the books from internet into local disk then performs map() operation on it and stores the data to mapper.json file, Which again is uploaded to a shred cloud bucket.
   Languageb used: Node JS

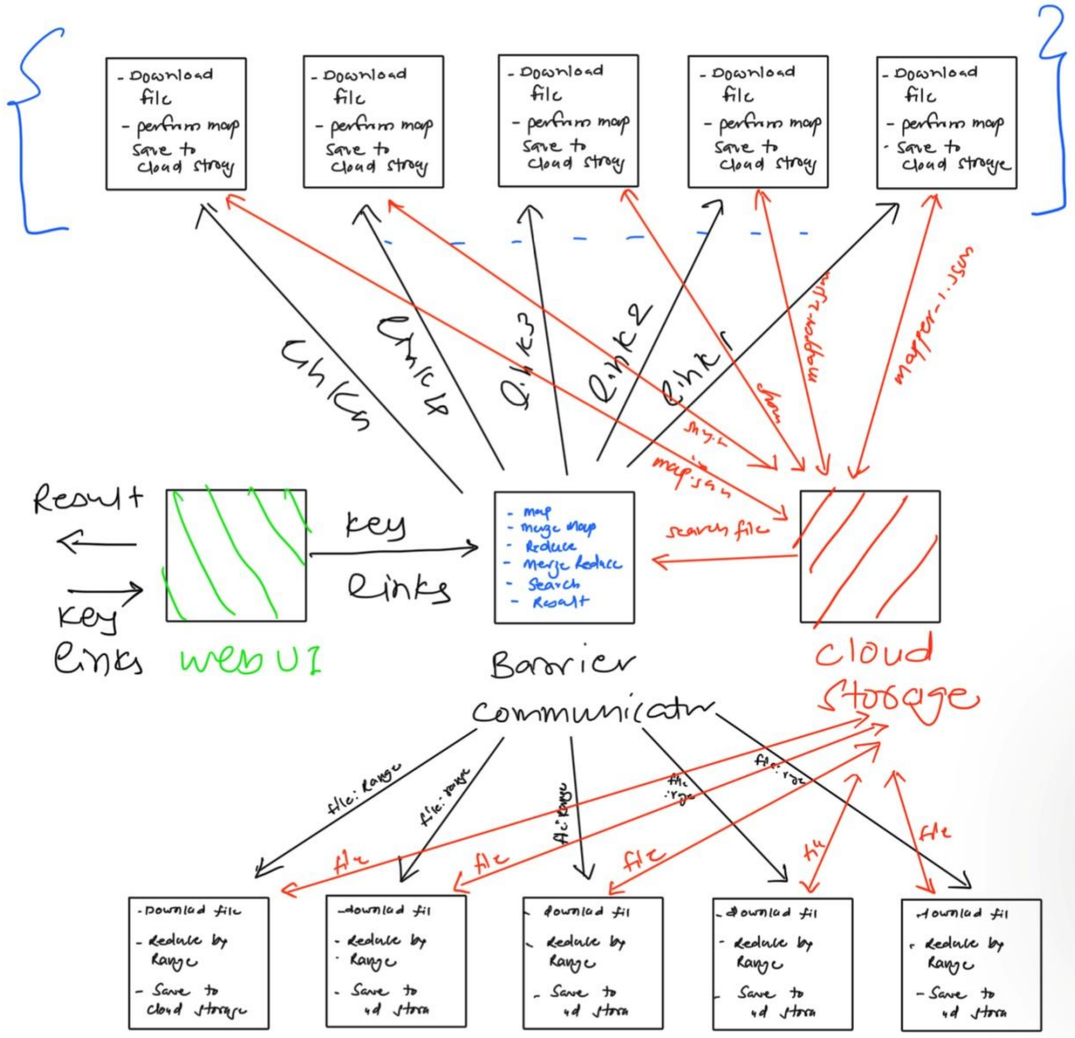
2. **Reducer Functions**
   The Reducer function Takes the input of Mapper.json file , then downloads the file from shared disk into local disk then performs reduce() operation on it and stores the data to reducer.json file, Which again is uploaded to a shred cloud bucket.
   Languageb used: Node JS

3. **The controller (Barrier)**
   The controller handles the communication between the Mapper function the Reducer function. Also it is the one who is responsible for invoking the mapper and the reducer. Also it merges the intermediate output of both map() and the reduce() on each iteration.
   Languageb used: Node JS

4. **Web UI**
   The web UI the end point of the system which just helps to invoke the cloud functions and to take the input from the user and display the result.
   Languageb used: Node JS

Following is the Diagram of Architecture.

mappers
Dynamic upon given files



Reducers

The cloud functions and their signatures, return values

Mapper does Map() on text files

Declaration:

```
async function Maper(req,res)
```

Input Values : json of

```
{
  link: "https://www.gutenberg.org/files/102/102-0.txt",
  fname: "101",
};
```

Return Values

```
res.status(200).send(ret);
```

Returns the Mapper files public Google cloud bucket link

Language Used: Node JS

MergeMapper Merges the json output files of all Map() output

Declaration:

```
async function MergeMapper(req,res)
```

Input Values:

```
{
  files: [
    "mapper intermediate file name",
    "mapper-103.json",
    "mapper-107.json",
    "mapper-108.json",
    "mapper-110.json",
  ],
};
```

Return Values:

```
res.status(200).send(ret);
```

Returns the MergeMapper files public google cloud bucket link

Language Used: Node JS

Reducer does Reducer() on text files

Declaration:

```
async function Reducer(req,res)
```

Input Values:

```
{
  files: "mapper.json", //Whole Mapper merged file
  range: "^[s-z]+$", //the range for to reduced
  filename: "reducer-5", //filename to be saved as
};
```

Return Values:

```
res.status(200).send(ret);
```

Returns the Reducer files public google cloud bucket link
Language Used: Node JS

MergeReducer Merges the json output files of all Reduce() output

Declaration:

```
async function MergeReducer(req,res)
```

Input Values:

```
{
  files: [
    "reducer-1.json", //list of all files produced by reducers
    "reducer-2.json",
    "reducer-3.json",
    "reducer-4.json",
    "reducer-5.json",
  ],
};
```

Return Values

```
res.status(200).send(ret);
```

Returns the MergeReducer files public google cloud bucket link
Language Used: Node JS

cloudMapReduce

Declaration:

```
async function cloudMapReduce(req, res)
```

Input Values: {

```
   files: [ //files links and file name to be refrenced as
     {
       link: "https://www.gutenberg.org/files/102/102-0.txt",
       fname: "102",
     },
     {
       link: "https://www.gutenberg.org/files/103/103-0.txt",
       fname: "103",
     },
     {
       link: "https://www.gutenberg.org/files/107/107-0.txt",
       fname: "107",
     },
     {
       link: "https://www.gutenberg.org/files/108/108-0.txt",
       fname: "108",
     },
     {
       link: "https://www.gutenberg.org/files/110/110-0.txt",
       fname: "110",
     },
   ],
};
```

Return Values:

```
res.status(200).send(ret);
```

Returns the FinalReducer files public google cloud bucket link

Language Used: Node JS

Data Structure Used:

1. **Mapper function**

   For mapper I have used following data structure to save the data

   ```
   {"text":"a","document":"102","count":1342},
   Text is word found in document and its count
   ```

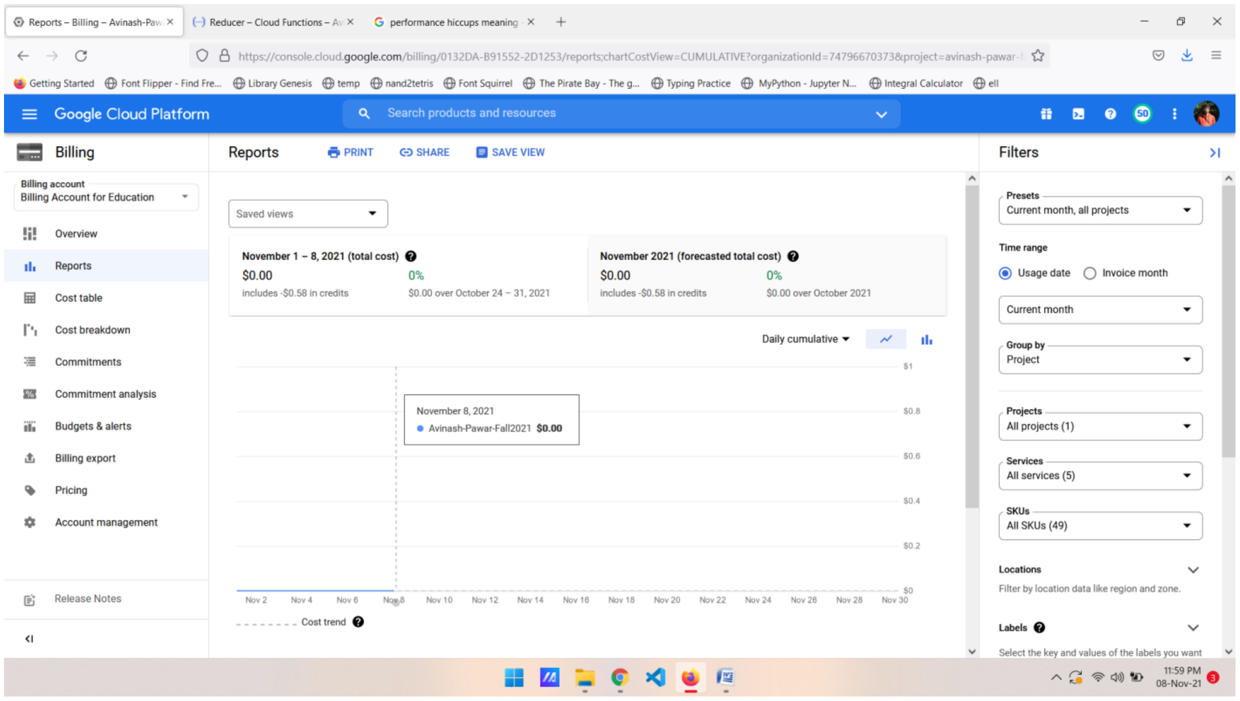
2. **Reducer function**

   For reducer I have used following data structure to save the data

   ```
   "a":[["102",1342],["103",1388],["108",2767],["107",4045],["110",3314]],
   Here "word": ["document", count] is used
   ```

Performance, cost and Charts

Cost: My cloud functions do not have anything as per cost say
As I have not stored any large data in bucket as I am dynamically downloading files from the internet. Also my functions take short time to execute. So cost is in 10usd to 100 usd depending upon concurrent user I would say. (mine now is close to 0)



https://console.cloud.google.com/billing/0132DA-B91552-2D1253/reports;chartCostView=CUMULATIVE?organizationId=747966703738&project=avinash-pawar-

Billing account: Billing Account for Education

Reports

November 1 – 8, 2021 (total cost): $0.00 — includes -$0.58 in credits — 0% — $0.00 over October 24 – 31, 2021

November 2021 (forecasted total cost): $0.00 — includes -$0.58 in credits — 0% — $0.00 over October 2021

November 8, 2021
Avinash-Pawar-Fall2021 $0.00

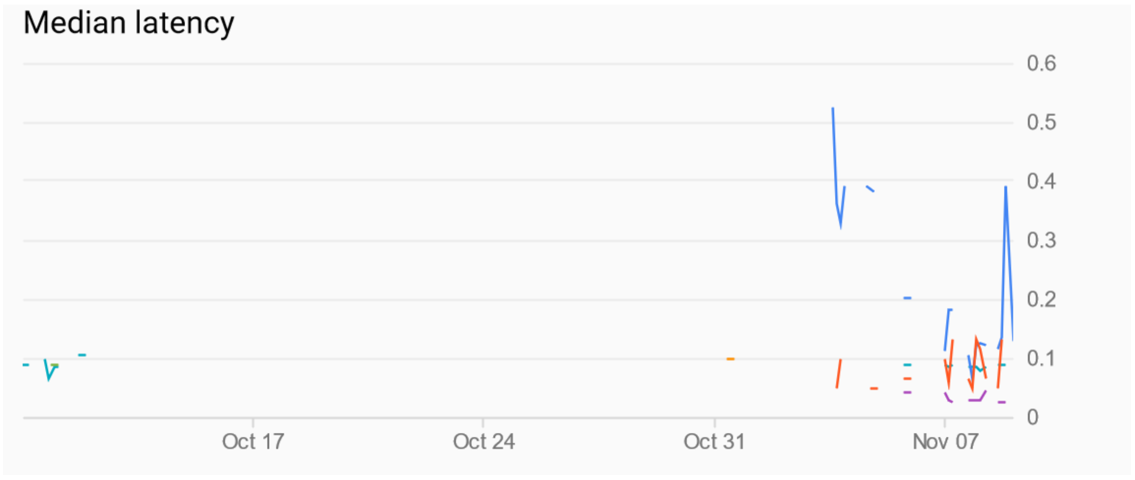
$1, $0.8, $0.6, $0.4, $0.2, $0

Nov 2, Nov 4, Nov 6, Nov 8, Nov 10, Nov 12, Nov 14, Nov 16, Nov 18, Nov 20, Nov 22, Nov 24, Nov 26, Nov 28, Nov 30

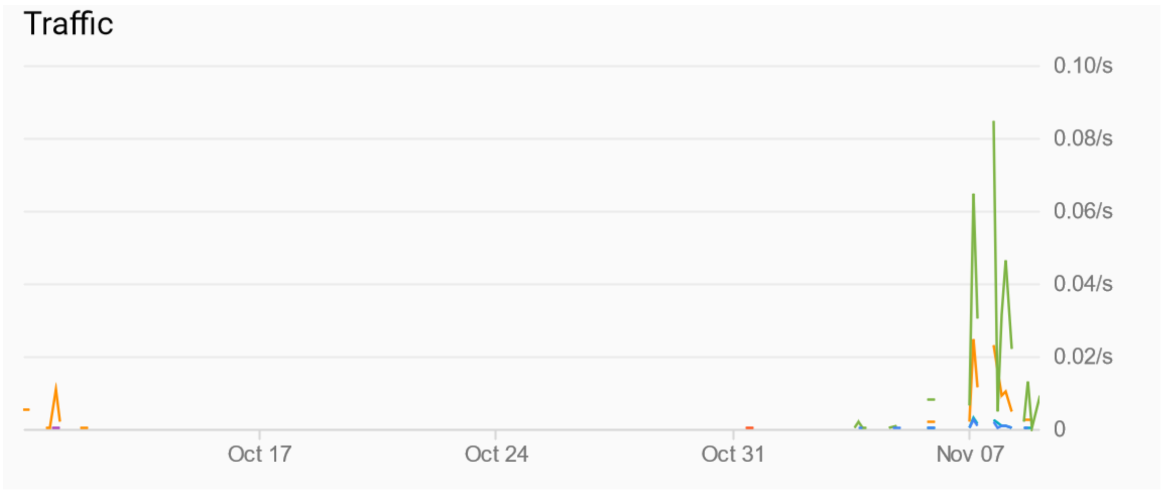
Presets: Current month, all projects
Time range: [✓ Usage date] Invoice month
Current month
Group by: Project
Projects: All projects (1)
Services: All services (5)
SKUs: All SKUs (49)
Locations: Filter by location data like region and zone.
Labels: Select the key and values of the labels you want

11:59 PM
08-Nov-21

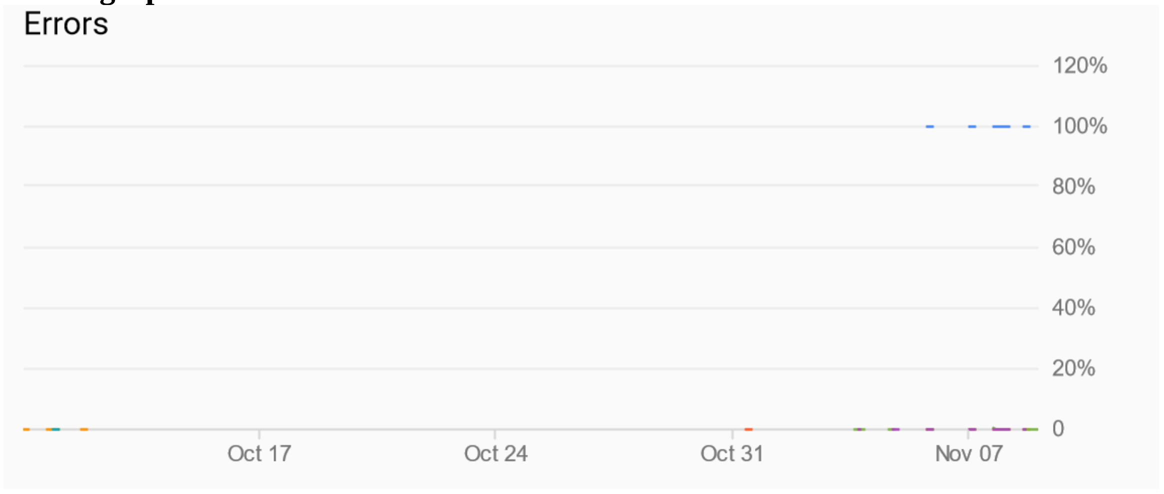
Median latency Graph



Traffic graph:



Errors graph:



The problem of Mapper distribution:
As we know that we need a balanced mapper distribution as we don't want one mapper to do all the work.
So I thought of the unique solution to divide load into 5 Mappers by finding out the probability of most occurring words in English and assigning mappers according to that.

The research is taken from
https://www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html
Hence I found out the balanced alphabetical ranges for 5 mappers are

A-D
E-H
I-M
N-R
S-Z

With average probability of occurring is 20%.

Roadblocks and performance hiccups
For the development of I have used the Node JS which was huge mistake for me as the Node JS does not have the necessary tools for data cleaning and processing.
Also the NodeJs don't have any support for the multithreading.

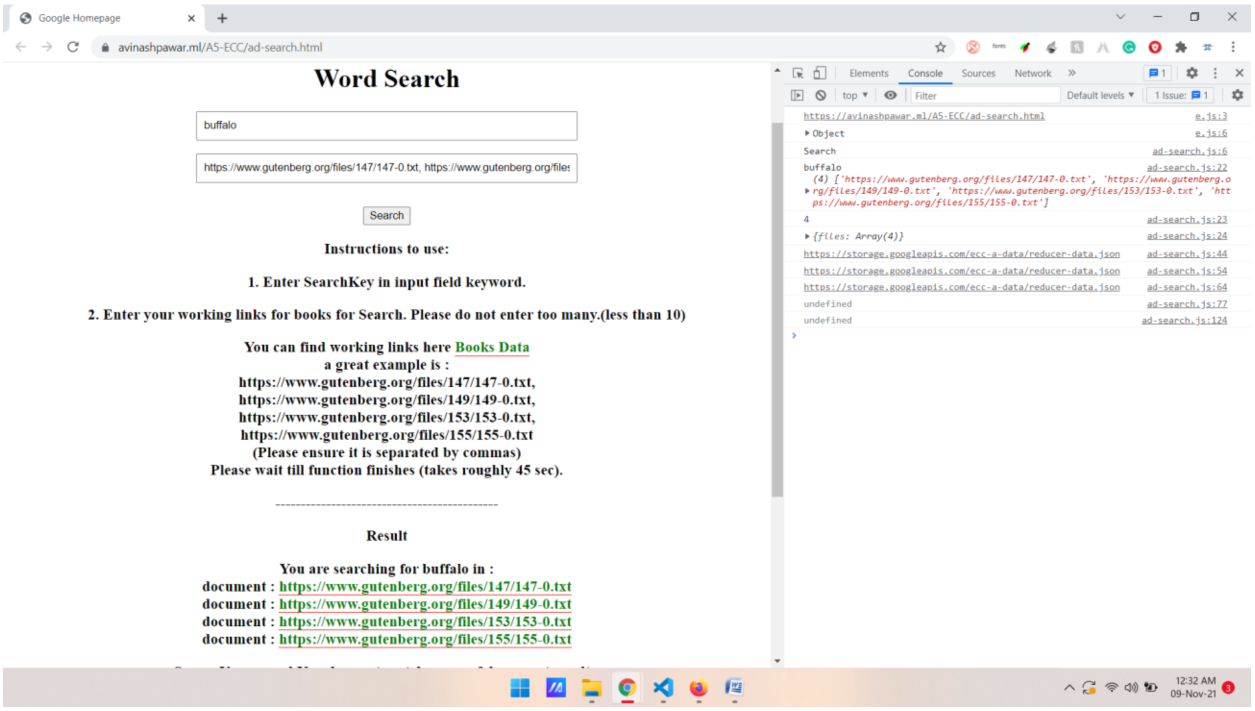
The Google cloud functions don't like the NodeJS environment at all. Sometimes the Cloud functions will work as expected but at some time they don't work at all. I spent 2 days debugging the issue without any luck.
Some of major roadblocks :

1. JSON.parse() :
   The NodeJS JSON.parse() functions reads chunks from the memory if the files become large as 25 mbs the method simply will not work hence halting the execution of function. I spend a solid day for this error without any luck resolving. Identifying the issue was the big achievement for me.
2. CORS :
   Even if you give the pubic access to the all the cloud buckets and files. When you try to acess from the Javascript or NodeJS you will simply faced by CORS (Cross-Origin Resource Sharing).
3. Unstable await execution:
   The NodeJs notorious await and other bugs will make your functions laggy and not even work sometimes.

Future improvements
As Discussed above will like to move away from the NodeJS and do all in python.
Also like to work on some of performance issues.

Result:
Following are the some of results screen from the Web UI

Advanced Search: you can provide your files:



Google Homepage

avinashpawar.ml/A5-ECC/ad-search.html

Word Search

buffalo

https://www.gutenberg.org/files/147/147-0.txt, https://www.gutenberg.org/files

Search

Instructions to use:

1. Enter SearchKey in input field keyword.

2. Enter your working links for books for Search. Please do not enter too many.(less than 10)

You can find working links here Books Data
a great example is :
https://www.gutenberg.org/files/147/147-0.txt,
https://www.gutenberg.org/files/149/149-0.txt,
https://www.gutenberg.org/files/153/153-0.txt,
https://www.gutenberg.org/files/155/155-0.txt
(Please ensure it is separated by commas)
Please wait till function finishes (takes roughly 45 sec).

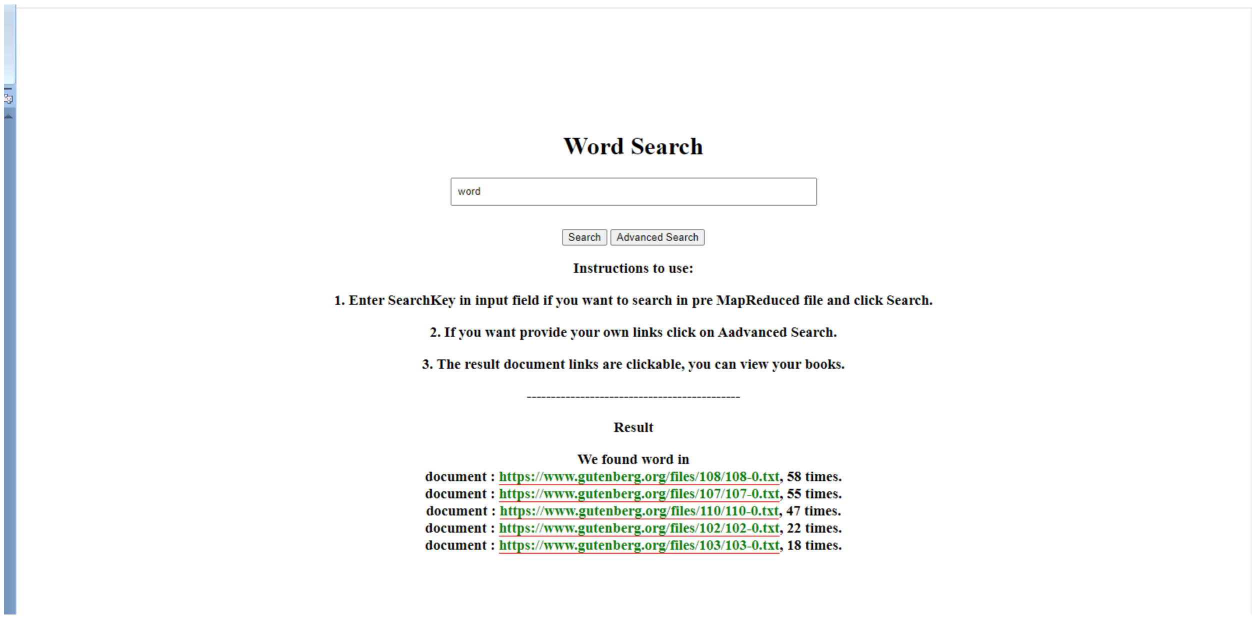
Result

You are searching for buffalo in :
document : https://www.gutenberg.org/files/147/147-0.txt
document : https://www.gutenberg.org/files/149/149-0.txt
document : https://www.gutenberg.org/files/153/153-0.txt
document : https://www.gutenberg.org/files/155/155-0.txt

```
https://avinashpawar.ml/A5-ECC/ad-search.html                          e.js:3
▸Object                                                                e.js:6
Search                                                         ad-search.js:6
buffalo                                                       ad-search.js:22
  (4) ['https://www.gutenberg.org/files/147/147-0.txt', 'https://www.gutenberg.o
▸ rg/files/149/149-0.txt', 'https://www.gutenberg.org/files/153/153-0.txt', 'htt
  ps://www.gutenberg.org/files/155/155-0.txt']
4                                                             ad-search.js:23
▸{files: Array(4)}                                            ad-search.js:24
https://storage.googleapis.com/ecc-a-data/reducer-data.json   ad-search.js:44
https://storage.googleapis.com/ecc-a-data/reducer-data.json   ad-search.js:54
https://storage.googleapis.com/ecc-a-data/reducer-data.json   ad-search.js:64
undefined                                                     ad-search.js:77
undefined                                                    ad-search.js:124
```

12:32 AM
09-Nov-21

Simple word search:



Word Search

word

Search | Advanced Search

Instructions to use:

1. Enter SearchKey in input field if you want to search in pre MapReduced file and click Search.

2. If you want provide your own links click on Aadvanced Search.

3. The result document links are clickable, you can view your books.

Result

We found word in
document : https://www.gutenberg.org/files/108/108-0.txt, 58 times.
document : https://www.gutenberg.org/files/107/107-0.txt, 55 times.
document : https://www.gutenberg.org/files/110/110-0.txt, 47 times.
document : https://www.gutenberg.org/files/102/102-0.txt, 22 times.
document : https://www.gutenberg.org/files/103/103-0.txt, 18 times.