

Cloud Computing Project Report

James Clackett

15732749

4th December 2023

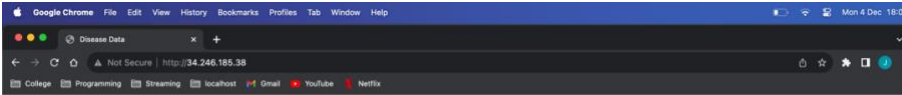
<http://3.254.64.219/>

Overview:

For this assignment, I opted for 'project 2'. The reason for this choice was that I wanted to explore Hadoop and Map Reduce in more depth than I had already. I was particularly interested in writing the Hadoop Map Reduce functions but also with using Map Reduce to create machine learning classifiers.

My initial project goal was to parallelise the creation of a fetal_health classifier I built in another module. Unfortunately after a number of days pursuing this goal, I had to give up. Mahout caused me nothing but trouble with various class path and version errors, and I was not interested in implementing Naïve Bayes in Map Reduce myself. So, instead I chose a new project idea. I found an interesting dataset on Kaggle that contained a number of diseases and patients' verbal descriptions of their symptoms. I decided to build an application that would find the most common words used to describe a disease, and present these on a dashboard that would be hosted publicly on the AWS cloud.

The application dashboard is running on an EC2 server and uses output from a Map Reduce job that was run in an AWS EMR cluster. This EMR cluster copies a Map Reduce JAR from my S3 bucket and sends the output back to S3. It also merges this output into a single file for the dashboard to use.



Diseases and their most commonly occurring descriptors:	
Fungal Infection:	Skin, Itching, Body, Itchy, Spots, Skin, Red, Patches, Rashy, Itch
Hypertension:	Headache, Experiencing, Feeling, Chest, Pain, Balance, Issues, Day, Experienced, Woke
Pneumonia:	Chills, Feeling, High, Chest, Mucus, Sweating, Throat, Coughing, Catch, Phlegm
Diabetes:	Throat, Mouth, Frequently, Healing, Dry, Dry, Skin, Infections, Hands, Taste
Gastroesophageal Reflux Disease:	Throat, Eating, Heartburn, Chest, Indigestion, Frequently, Eat, Food, Acid, Taste
Acne:	Skin, Pimples, Blackheads, Rash, Pus-Filled, Skin, Nasty, Developed, Pus, Additionally
Cervical Spondylosis:	Pain, Experiencing, Struggling, Suffering, Severe, Cough, Muscles, Weak, Weakness, Coughing
Dimorphic Hemorrhoids:	Anus, Difficult, Bowel, Painful, Constipation, Recently, Experiencing, Trouble, Hard, Lately
Jaundice:	Experiencing, Scratchy, Itching, Weight, Feeling, Lost, Intense, Throwing, Losing, Skin
Malaria:	High, Fever, Itching, Severe, Itchiness, Experiencing, Intense, Temperature, Strong, Scratching
Typhoid:	Constipation, Pain, Fever, Experiencing, Stomach, Diarrhea, High, Vomiting, Abdominal, Belly
Varicose Veins:	Legs, Causing, Veins, Calves, Skin, Swollen, Long, Periods, Cramps, Blood
Allergy:	Eyes, Skin, Occasionally, Lips, Nose, Throat, Sore, Sneezing, Sneeze, Breathing
Common Cold:	Sneezing, Throat, Nose, Tired, Sinuses, Red, Feeling, Exhausted, Eyes, Coughing

Figure 1: The application dashboard

Project Objectives:

The project 2 task had several objectives. I will go through my work for these in order:

Identification of dataset and problem to solve:

It took me a while to find a dataset I wanted. As already discussed I initially wanted to use a dataset from another module but this did not go to plan. As instructed on the project specification, I next looked in Kaggle. I wanted to implement a counting map reduce job, but not the simplest example found in many tutorials. I found the Symptom2Disease.csv[1] dataset. This provided the perfect opportunity. It had several diseases with a corresponding description. I then came up with the idea of a most common symptom calculator. Rather than a simple count of words, I would need to find a way of isolating each disease, count its words, throw away irrelevant 'stop words', sort the results, and finally remove all but the most popular.

Selecting and implementing Map Reduce algorithm:

This was the part of the project that took the most amount of time. The finished algorithm can be found in my project folder in 'AWS EMR/Java Projects/CC_Project'. I found various sources online that had created a counting algorithm for Hadoop Map Reduce. I studied these in detail and implemented them myself in a number of test applications. When I was confident that I fully understood how to write Map Reduce functions in Java, I set about creating the more complex version for my app.

The mapper was relatively simple, it separates each line of csv data into a pair of disease name (key), and a symptom (value). One extra aspect is that a text file containing stop words is imported and used to filter out unnecessary words from the algorithm like 'this', 'I', 'then' etc.

The reducer is responsible for taking a key and set of values, and doing some work on them. In the case of my algorithm, it takes a count of the occurrences of each word, and adds this to a hash map of <word: count>. It then creates a new sorted list from this map, where the order is of descending count. It then removes all but the most common 10 words. The final product of the Map Reduce job is a series of output files that contain data in the format:

Disease Name topWord1: count, topWord2: count, , topWordN: count

Building a Dashboard:

Building a dashboard for the app was the easiest part of the project as it is something we have done many times before. The code used to build the dashboard was common boilerplate Flask code. There was a route for the "/" path, and a template that would be rendered with data received. It used Bootstrap CSS to create a nicer UI but not much more was needed. The app simply reads from the merged output file created by Hadoop. It creates a dictionary of { disease_name: [word] } and passes this to the template. Each disease name is also used to dynamically create a Wikipedia URL that the user can click on. The app runs in an AWS EC2 instance. It is publicly available as I have used nginx to serve the app on the servers port 80 (where public http and https traffic has been enabled). In order to copy the merged output file from my S3 bucket, it was necessary to create a new IAM role

that gave READ permission for that bucket to my EC2. Git was used to clone the Flask app from my private repo into the instance. Finally, nohup was used to run the Flask app in the background so that I can leave the instance and not have the process killed. Please refer to my project source files and screenshots/videos for more information. There is much more detail than I can sensibly provide here.

Problem and Dataset:

I believe that the idea behind the project was to become familiar with Hadoop Map Reduce, and working in a cloud environment. For these reasons I opted not to spend much time on creating a beautiful interactive dashboard. This seemed to be besides the point. Secondly, I believe that the project wanted us to showcase the ability to create a parallel processing solution, but not necessary use an enormous dataset to showcase its benefits. It is also true that I no longer have the free tier of AWS and need to be mindful of the resources I use. Throughout the project, I aimed to act as if I was using an very large dataset, and avoided implementing any measures that you would not take when dealing with one. For example, there is much sample code and tutorials online where only 1 core is used in AWS EMR, or examples where the number of reducers are set to zero. While these measures make sense in a local environment, they do not in the context of parallel processing of big data. For this reason I avoided doing the same.

The dataset I found was reasonably sized (230KB) and from initial inspection I could tell there were no missing or invalid values. This was great as I did not need to spend any time cleaning or preparing the data. The only necessary change was removing extra quotations from the symptoms description. As discussed earlier, stop words were also removed but I do not believe these were indicative of an unclean dataset. There were 24 unique diseases in the dataset, each with different descriptions. It provided the perfect clarity and information needed to achieve my project goal.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		label	text											
2		0 Psoriasis	I have been experiencing a skin rash on my arms, legs, and torso for the past few weeks. It is red, itchy, and covered in dry, scaly patches.											
3		1 Psoriasis	My skin has been peeling, especially on my knees, elbows, and scalp. This peeling is often accompanied by a burning or stinging sensation.											
4		2 Psoriasis	I have been experiencing joint pain in my fingers, wrists, and knees. The pain is often achy and throbbing, and it gets worse when I move my joints.											
5		3 Psoriasis	There is a silver like dusting on my skin, especially on my lower back and scalp. This dusting is made up of small scales that flake off easily when I scratch them.											
6		4 Psoriasis	My nails have small dents or pits in them, and they often feel inflammatory and tender to the touch. Even there are minor rashes on my arms.											
7		5 Psoriasis	The skin on my palms and soles is thickened and has deep cracks. These cracks are painful and bleed easily.											
8		6 Psoriasis	The skin around my mouth, nose, and eyes is red and inflamed. It is often itchy and uncomfortable. There is a noticeable inflammation in my nails.											
9		7 Psoriasis	My skin is very sensitive and reacts easily to changes in temperature or humidity. I often have to be careful about what products I use on my skin.											
10		8 Psoriasis	I have noticed a sudden peeling of skin at different parts of my body, mainly arms, legs and back. Also, I face severe joint pain and skin rashes.											

Figure 2: A small sample of the dataset

Suitability of Hadoop Map Reduce:

In today's world there are better options than Hadoop Map Reduce. Apache's Spark has essentially become its replacement and is more performant, easier to use, and contains a newer range of tools. It is also more compatible with modern applications and

environments. When researching different approaches for my algorithm, I was suggested Spark at every turn. I also noticed that documentation was lacking or outdated in some areas. This was especially true for libraries/frameworks built for Hadoop like Mahout. This was exactly what frustrated my initial assignment efforts.

While being worse in many regards, the most positive aspect of Hadoop Map Reduce is its lack of complexity. While solutions built with it may be less performant, they are easier to understand. Especially for students. This project was a very useful exercise in understanding parallel programming, at least at a high level. In terms of its suitability for the application itself, this is more difficult to discern. I did not test my application on a large dataset due to resource constraints so I am unable to confirm any obvious performance drawbacks. It is my understanding that Map Reduce does not work well for producing real time results (due to its batch processing nature). This certainly would not have been an issue in my application, as the Map Reduce job was run before the app is even deployed. An app such as mine would typically listen to an update event in the s3 data store and download a new version of the job output. This still would not affect performance as the previous file would be used until the update has completed.

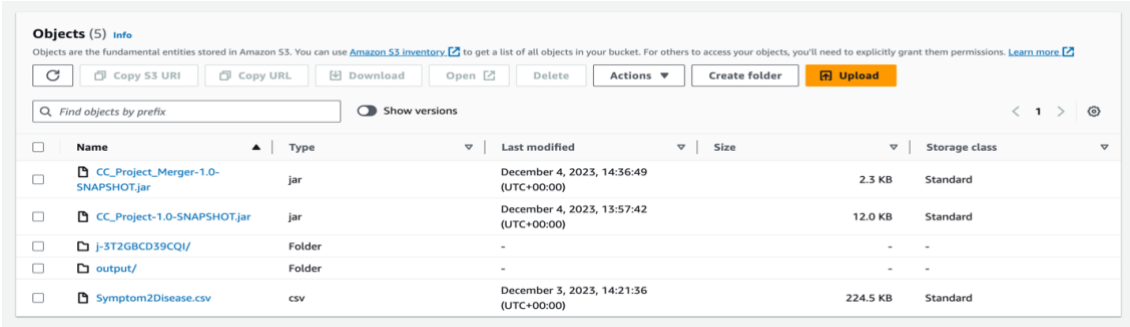
With a simple application such as this, Hadoop Map Reduce is definitely a good choice. It offers simplicity, and real time processing is not necessary. Map Reduce's simplicity depends on the complexity of the algorithm being constructed. It suits simple tasks much better. For a more complicated application, it may be too cumbersome to use which is where something like Spark would be a better option.

Application feature set:

Please review my screenshots and videos in the '/Media' folder for working examples of my application. I show screenshots of various aspects of the app, and have recorded a video and voiceover for key parts of the build.

S3 Datastore:

The AWS S3 Data store is used to store the Hadoop Map Reduce JAR. This Jar is used by EMR to create a Map Reduce Step. It also stores the Symptom2Disease.csv dataset that the EMR used to run its job on. In addition to these, it is the designated location for output files and logs created by EMR.



The screenshot shows the AWS S3 console interface for a bucket. At the top, there's a header 'Objects (5) info' with a link to 'Amazon S3 inventory'. Below this is a toolbar with buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload'. A search bar 'Find objects by prefix' and a 'Show versions' toggle are also present. The main area is a table with columns: Name, Type, Last modified, Size, and Storage class. The table lists five objects: 'CC_Project_Merger-1.0-SNAPSHOT.jar' (2.3 KB), 'CC_Project-1.0-SNAPSHOT.jar' (12.0 KB), 'j-3T2GBCD39CQ/' (Folder), 'output/' (Folder), and 'Symptom2Disease.csv' (224.5 KB).

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	CC_Project_Merger-1.0-SNAPSHOT.jar	jar	December 4, 2023, 14:36:49 (UTC+00:00)	2.3 KB	Standard
<input type="checkbox"/>	CC_Project-1.0-SNAPSHOT.jar	jar	December 4, 2023, 13:57:42 (UTC+00:00)	12.0 KB	Standard
<input type="checkbox"/>	j-3T2GBCD39CQ/	Folder	-	-	-
<input type="checkbox"/>	output/	Folder	-	-	-
<input type="checkbox"/>	Symptom2Disease.csv	csv	December 3, 2023, 14:21:36 (UTC+00:00)	224.5 KB	Standard

Figure 3: S3 Bucket

AWS EMR:

EMR was used to create a Hadoop environment for this project. It provides a very easy way of creating a Hadoop HDFS in the cloud. JARs were the primary means of running jobs in the cluster, and it was not necessary to run them manually from inside the master node. The AWS EMR Dashboard offers a 'steps' option that lets you specify JAR and behaviour to carry out. It did not provide a means for merging output files which would have been nice. It is possible to create scripts and add these as a different 'step', however it was much easier to SSH into the cluster and merge the files manually. I copied the output folder into the HDFS, merged them into one text file, and sent this back to S3.

The image shows two side-by-side panels from the AWS EMR console. The left panel, titled 'Step settings', shows a 'Customised JAR' step named 'Map Reduce symptom counter job'. The JAR location is 's3://cc-project-disease/CC_Project-1.0-SNAPSHOT.jar'. The arguments are 'SymptomCounter', 's3://cc-project-disease/Symptom2Disease.csv', and 's3://cc-project-disease/output'. The right panel, titled 'Summary', shows the step's configuration: Name 'Cloud Project', Amazon EMR release 'emr-6.15.0', Application bundle 'Custom (Hadoop 3.3.6)', Instance groups 'Primary (m5.xlarge), Core (m5.xlarge)', Provisioning configuration 'Core size: 3 instances', and Networking.

EC2 Dashboard:

The Dashboard allows users to access the information that has been created by Hadoop. It displays a list of each disease, and all the most common words associated with that disease. It has a brief title and headline explaining what it is/does. Each disease name is also a hyperlink that takes the user to the relevant Wikipedia page about the disease. This link is created dynamically using the Jinja2 templating engine within Flask. Finally, as Bootstrap CSS was used to style the dashboard, it is responsive to different device and screen sizes.

The image shows a terminal window with the command 'nano /etc/nginx/conf.d/dashboard-app.conf'. The configuration file content is as follows:

```
server {
    listen 80;
    server_name myprojectdomain;

    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }
}
```

Figure 4: nginx configuration file

Final Thoughts:

This project has been extremely useful for putting concepts discussed in lectures to use. It has covered the cloud, deployment, distributed data stores, and parallel programming. It was unfortunate that much of my time was wasted in the initial few days as I tried to force a Mahout solution to work. However the positive was that I still had to navigate the EMR cluster while trying to implement that solution and therefore gained an understanding of the environment. As a result, when I decided to change my project idea I wasn't started from absolute scratch.

The disease symptom project was just as interesting as the initial idea would have been, and allowed me to write my own Map Reduce jobs. This is something I would not have learned to do had I persevered with Mahout. Since our lectures and practical's placed such an emphasis on Hadoop and map reduce, this was probably for the best. Upon finishing this project I now have a working application that I can showcase and discuss in interviews. I also have a much greater understanding of parallel computing and map reduce. Most importantly I am now familiar with AWS EMR, a cloud service that I will certainly need to use in the future.

References:

- [1] Barman, N R. **Symptom 2 Disease**, April 2023, Available at:
<https://www.kaggle.com/datasets/niyarrbarman/symptom2disease>