School of Computing and Information Systems
University of Melbourne
Australia

Australian Social Media Analyze

Team 37
Huan Cao
985151 Melbourne, Australia
Mindong Xu
1022525 Adelaide, Australia
Yaojia Zhang
1018668 Chongqing, China
Zhixuan Wei
988348 Melbourne, Australia

27-May-2020

Contents

1. Introduction

In March 2020, the World Health Organization (WHO) declared Coronavirus disease 2019 (COVID-19) a pandemic, pointing to over 110 countries and territories around the world where the coronavirus illness is present. Infectious disease outbreaks such as COVID-19, as well as other public health events, can cause emotional distress and anxiety. In facing the outbreak of COVID 19, life of every individual in Australia has been greatly impacted by the pandemic. While public health procedures such as social distance and working from home is effectively slowing down the spread of the virus, it is certain that more people choose to spend their time online for all kinds of activity.

Twitter is a popular social media platform where the number of daily active Twitter users reached 152 million[1]. People illustrate and communicate their views and attitudes about extensive topics. Then information volumes are exploding, data analytics is becoming more compute intensive. Then, we decide to take advantages of its data and analysis the different expressions of emotion before and after the outbreak.

This report focuses on telling stories of life in Australian impacted by COVID 19, and we are interested in the emotional change of people who live in Australia, and how does it relate with the income and employment rate offered by AURIN. To search the solution, a dynamic and working balancing cloud computing system has been deployed on the *UniMelb Research Cloud*. Also, to improve the efficiency of data analyzing, Tweeter Harvester, a database cluster and a web server have been allocated on *UniMelb Research Cloud*.

First, this exhaustive report describes the data and technology which used in building cluster computing systems, generalizing the function of each technology. Second, strategy and the details of *System Architecture Design* a are involved. In **Chapter 5**, we illustrate how life impacted by COVID -19 by analyzing Tweet. Lastly, to improve, in **Chapter 6** an self-evaluation of system is displayed.

[1] https://www.statista.com

2. Technologies Used

2.1 Ansible

Ansible is an IT automation engine that automates configuration management, cloud provisioning, application deployment, intra-service orchestration, and many other IT needs. With Ansible Playbook (plain-text YAML files), application intrastate is perfectly described and automation is human readable. Also, systems resources, packages files are managed in Modules. In this project, Ansible is used to deploy instances on *UniMelb Research Cloud* and install necessary software packages on it.

2.2 Docker

Docker is a containerization technology, which requires instances to share a single host OS to reduce wasted resources.Docker is an open platform and a containerization technology, improving the software delivering speed by separating users' applications from their infrastructure. Docker images, containers and swarm are used in our system to build a CouchDB clustering database and achieve the load balance.

2.2.1 Docker Platform

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security enable containers simultaneously on a given host. Considering that no extra load of a hypervisor needs and could be run directly within the host machine's kernel, containers are lightweight.

2.2.2 Swarm

The cluster management and orchestration features embedded in the Docker Engine are set up depending on swarmkit.[1]A swarm composed by multiple Docker hosts, which run in swarm mode and behave as workers and manager (who are responsible to manage membership and delegation). Optimal state should be clarified when service is created, and then docker works obligate to maintain that desired state. If any worker node is unavailable, Docker assign that node's tasks to other nodes. Comparing with standalone containers, the primary advantage of swarm services is that service's configuration could be modified without restarting the service manually.

2.3 CouchDB

Apache CouchDB is a document-oriented NoSQL database, storing data in Json and querying language by MapReduce. And the cluster architecture is easy to set up by http API. Moreover, with multi-version concurrency control (MVCC), without locking the database, CouchDB could handle a high volume of readers and writers concurrently. In this project, an automated deployment of CouchDB has been implemented to a *UniMelb Research Cloud* instance by Docker swarm.

2.3.1 Views: Query Documents with MapReduce

After querying a view, every document in the database would be run based on the view was defined in. Users query the view to retrieve the view result. Furthermore, CouchDB is aimed at promising no extra costs: all documents only are run once when the first query is done. The map function which is only run once could handle all the changes in the document, to recompute the keys and values.

2.4 Twitter API

A Python library *tweepy* is used to crawl tweets on Twitter, and the amount and efficiency are depending on the type the Twitter API. For the research API, there is a rate limit which are spited into a 15-minute interval. Thus, to ensure there would be large amount of tweet in our database, two tweet harvesters are deployed with different authentication. And each harvester use *tweepy* to listen on tweets stream.

[1]Swarmkit:A separate project which implements Docker's orchestration layer and is used directly within Docker.

2.5 TextBlob

A sentiment analysis implemented by using a Python library *TextBlob*. The function of sentiment analysis in *TextBlob* is supported by Naive Bayes Method. After calculating the conditional probability of each token in the tweet, a polarity score which indicates the emotion of this tweet could be return.There are three situations of the score and represents three kinds of emotion: score lager than 0 means the sentiment is positive, whereas smaller than 0 is negative, and equals to 0 demonstrates neutral.

3. Design of Experiment

3.1 Cloud System Architecture

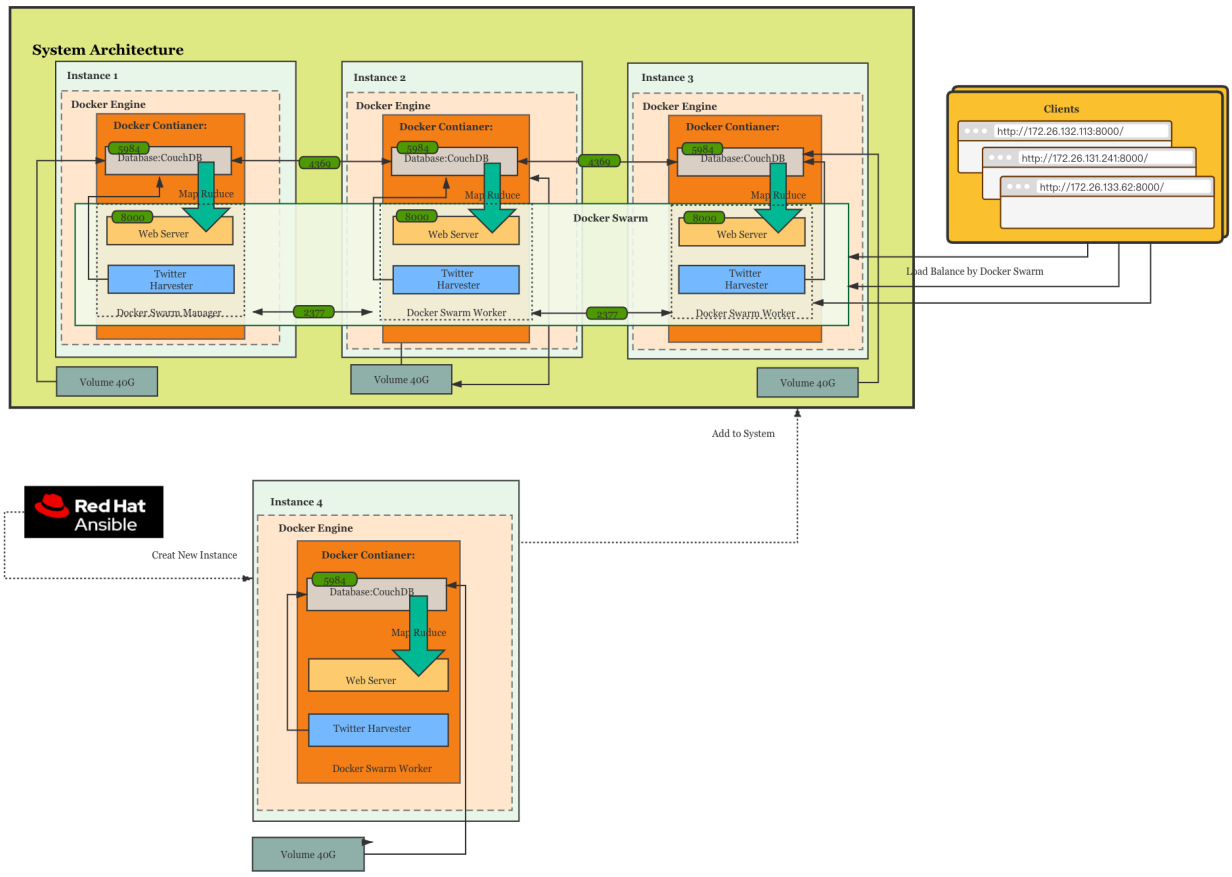

Figure 3.1: Architecture of Cloud

After abstracting a detailed function list of our system and considering the limited resources[1], we proposal our architecture. There are three main essential parts in this system and the details of each function are illustrated in Section 4.2: *Tweet Harvester* is in Section 4.2.1,*Database:CouchDB* is in Section 4.2.2, and *Web Server* is in Section 4.2.3. The processed public Tweets which crawled by Twitter Harvester are stored into CouchDB. And Views that generated based on MapReduce algorithm pass the analyzed data to Web Server.

To achieve the automated deployment and have the scripted deployment capabilities, Ansible which has been introduced in Section 3.2.1 is the main tool. As Figure 3.1 shows, there are 3 working instances in our system and more instances could be added manually through Ansible script, which satisfy scaling up and down according to the actual demand. Clients could reach our system by access the IP address through *8000* port, and the page view could be load balanced by Docker Swarm. After installed Docker environment on each instance, Instance 1 is treated as Swarm manager. And other instances are joined to our Docker Swarm as worker. Each instance could be connected through a unique IP address. With Docker Swarm, the requests sent by browser would be assigned to each node by swarm manager.

For each instance, CouchDB is deployed by Docker Engine through port *5984*, and Web Server and Tweet Harvester is deployed by Docker Swarm. Consider there are multiple Gigabyte of Tweet data stored in database, a CouchDB Cluster is need. Also, to improve the stability and reliability, Web server and Twitter Harvester are deployed in Docker Swarm. After deploying the Docker Swarm, we could apply rolling updated to nodes incrementally. Our Swarm manger could control the delay between service deployment to various nodes. When we are updating our system, we could handle all the unexpected situation by rolling back to a previous version. Moreover, if any scale up or down is needed, our Swarm manger automatically adapts the new demand by adding or removing tasks to keep the ideal state.

3.2 Deployment

3.2.1 Docker Container

CouchDB

Tweets stored as JSON format in CouchDB with organizing key-value pairs. After analysing around 100 GB Tweet data provided by third party and crawling serveral decades days, there are around 1GB JSON documents in CouchDB.

[1]Limited Resources: 4 servers, 8 virtual CPUs ,36Gb memory, and 250Gb volume storage

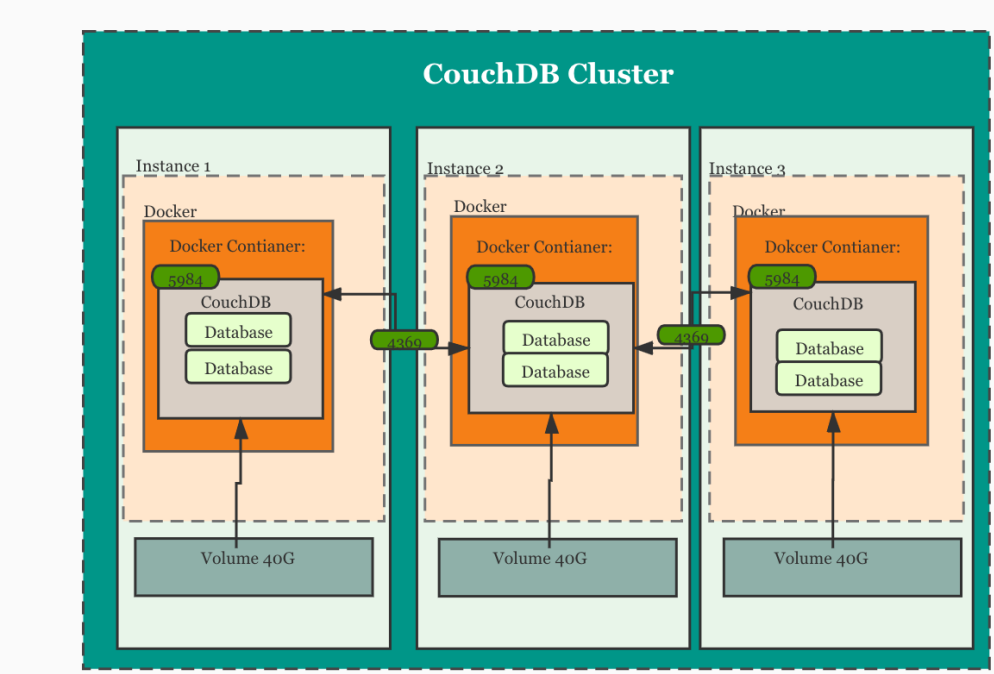

Figure 3.2: Architecture of CouchDB Cluster

An automated deployment of CouchDB has been implemented to *Unimelb Research Cloud* via Docker Container. As Figure 3.2 shows, our CouchDB cluster is deployed on 3 instances in Docker Container, with a 40 Gigabyte volume attached to each node. A straightforward and simple approach for CouchDB backup is applying CouchDB replication to another CouchDB installation. In our system, considering the limitation of the number of nodes in *Unimelb Research Cloud*,two replication are made.

CouchDB Cluster

To ensure the high availability, the CouchDB cluster is built based on partitioning data into shards[2] and allocating copies of each shard [3] to different nodes. With this setting in CouchDB, the strong durability against node loss is ensures. Also as a user-friendly database, shard could be created automatically and the subsets of documents which attribute to shard are distributed automatically. Thus, availability and partition tolerance are obeyed in our system. And CouchDB Cluster is deployed based on Docker container and Ansible playbooks.

[2]A shard is a horizontal partition of data in a database
[3]Copy of shard: shard replicas / replicas

In our system, CouchDB cluster provides two main strengths, especially in working with high-volume database environments. The first one is the capacity of fault tolerance. There are three servers for clients to connect to, clustering offers an alternative, in the case of individual server failure. And load balancing is the second advantage. The high volume of request to different shard could allocated to each node evenly, then the capacity of computing is increased. This growth shows obviously, for example, the time needed to create Views reduces dramatically.

With regard to scale up, Ansible is fundamentally an orchestrator – a new server Instance 4 which also includes a CouchDB in docker container. However, the existed data will not be replicated to new CouchDB automatedly and the new settled database could only be used to store new data. As the User Guide of CouchDB stated, modifying cluster membership and shard behavior should be done manually. To update the cluster metadata, a special and build-in database which maps databases to shards and nodes is used.

In respect of scaling down, it is more complicated than scaling up. To remove one node, we need to move all the shards on this node evenly to other nodes manually. Avoiding any unbalanced load, CouchDB will not redistribute those shards, we must move it manually. All the details of each operation could be found at CouchDB User Guide[4].

3.2.2 Docker Image

In this project, to achieve load balance and improve the stability of system, Tweet Harvester and Web server are deployed on the Docker Swarm by packing them into images. All the deployment finished by the Docker Command, and the details are included in User Guide(Appendix A).

[4]https://docs.couchdb.org/en/stable/cluster/sharding.html

3.3 Service on Cloud

3.3.1 Twitter Harvester

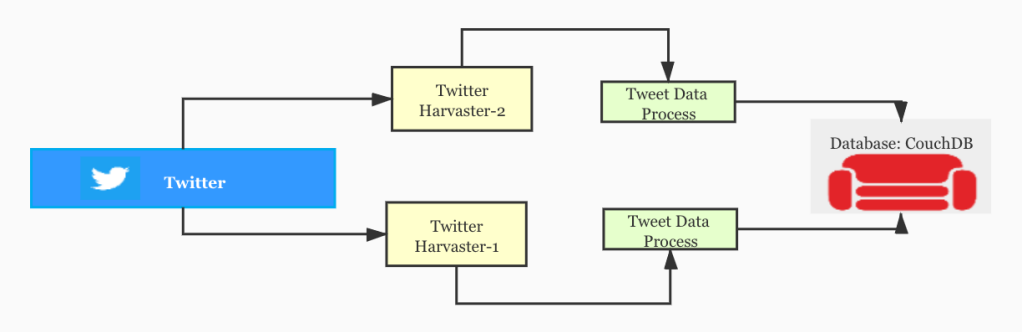

Figure 3.3: Architecture of Twitter Harvester

By connecting directly to Twitter data service, two Tweet harvesters are deployed on *Unimelb Research Cloud*: one is working on crawling public Tweets over Australia, another is responsible for crawling tweets over Melbourne. Once a new real-time public Tweets is caught, it will be processed and stored in database,sequentially. Considering the purpose of this experiment is demonstrating the usage of Twitter under COVID-19, thus only real-time tweet and *Streaming API* are need in this experiment. With *StreamListener*[5] object, *Streaming API* retrieves real-time public Tweets which constrained by *filter()*. The only restraint in our harvesters is the location *geo*[6], based on the coordinates. Then the public Tweets sent by people in Australia and in Melbourne are kept.

Data Prepossessing

Before storing data in our database, a simple data processing is applied. According to the settings in Twitter APIs, data provided by return data are encoded in *JSON*[7]. There are five main attributes(all 76 attributes, sub included) recorded by each tweet: *an author, a message, a unique ID, a timestamp* of when it was posted, and sometimes *geo* metadata. In this step, three attributes are extracted out and one new attributed("sentiment" which record the polarity score calculated by *TextBlob*)is added in Tweet *JSON*.

[5]StreamListener: a key component in Steaming API
[6]An attribute in Tweet JSON
[7]JavaScript Object Notation: Based on key-value pairs, with named attributes and associated values

created_at	id	id_str	text	truncated
in_reply_to_status_id	in_reply_to_status_id_str	in_reply_to_user_id	in_reply_to_user_id_str	in_reply_to_screen_name
userid	id_str	name	screen_name	location
url	description	translator_type	protected	verified
followers_count	friends_count	listed_count	favourites_count	statuses_count
created_at	utc_offset	time_zone	geo_enabled	lang
contributors_enabled	is_translator	profile_background_color	profile_background_image_url	profile_background_image_url_https
profile_background_tile	profile_link_color	profile_sidebar_border_color	profile_sidebar_fill_color	profile_text_color
profile_use_background_image	profile_image_url	profile_image_url_https	profile_banner_url	default_profile
default_profile_image	following	follow_request_sent	notifications	geo
coordinates	placeid	url	place_type	name
full_name	country_code	country	bounding_boxtype	coordinates
attributes	contributorsnull	is_quote_statusfalse	quote_count	reply_count
retweet_count	favorite_count	entitieshashtags	urls	user_mentions
symbols	favorited	retweeted	filter_level	lang
timestamp_ms				

Figure 3.4: Attributes of Raw Tweet JSON

created_at	id	user_screen_name	text	hashtags
sentiment	geo	place	country	state
city	coordinates			

Figure 3.5: Attributes of Processed Tweet JSON: Australia

created_at	id	user_screen_name	text	hashtags
sentiment	geo	place	full_name	state
city	country	coordinates	suburb	suburb_is_ignore

Figure 3.6: Attributes of Processed Tweet JSON: Melbourne

Error Handling: Disconnection 420

As stated in Guides [8] provided by Twitter Developer, there are five reasons of a streaming connection close, but only two situations are related in this experiment:

1. Too many connections are established with the same credentials.
2. Reading data too slowly.

The first scenario happens when Twitter API receives too many attempts from a same client in a short period of time. To solve this status and reconnect, our Twitter Harvester will sleep 90 seconds and then reconnect with Streaming API.

The second scenario happens when we are trying to interact with database and save data to JSON file. With streaming high volumes of real-time Tweets comes,

[8] https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/connecting

an error handing is needed to maintain both data reliability and data full fidelity. To maximize our connection time and reconnect automatically, an asynchronous process is deployed through a multi-thread program.

Error Handling: Twitter Replication

To solve the problem of duplicate tweets, after analyzing the official document from Twitter, we proposal our solution. There are two steps involved in this solution:

1. Constraints added in Twitter Harvester Program
2. Setting Tweet id to the document id in CouchDB

In step 1, after adding filter requirements, only original Tweets are downloaded. In more detail, for each Tweet JSON, neither of *retweeted status* and *quoted status* existed and *in reply to status id* is null, it would be treated as original Tweet.
In step 2, provided that Tweet id would never be duplicated, then we allow Tweets id to act our document id in our CouchDB database. When a Tweet with duplicate id sent to CouchDB, it would not be recorded.

With this simple and straightforward solution, we could guarantee no duplicated Tweet in our database.

3.3.2 Views: MapReduce

In terms of passing data to web server fast and efficiently, Views are created. Based on the purpose of data analysis, 8 Views are created with MapReduce Algorithm. There are 4 Views for Australia Tweet data, and 4 views for Melbourne Tweet data, the statistic, sentiments of historical Tweet and real-time Tweet.

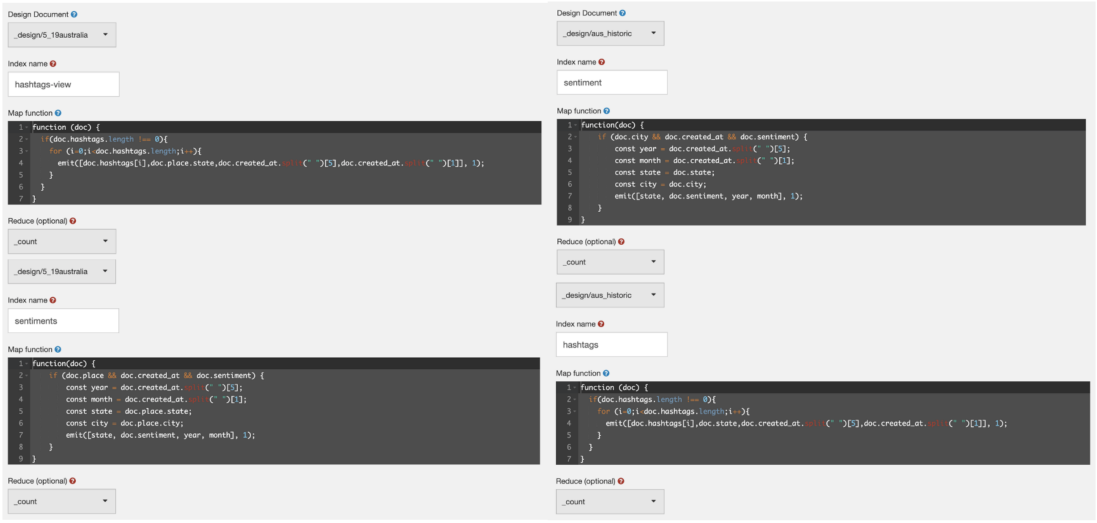

Figure 3.7: Views for Australia Data

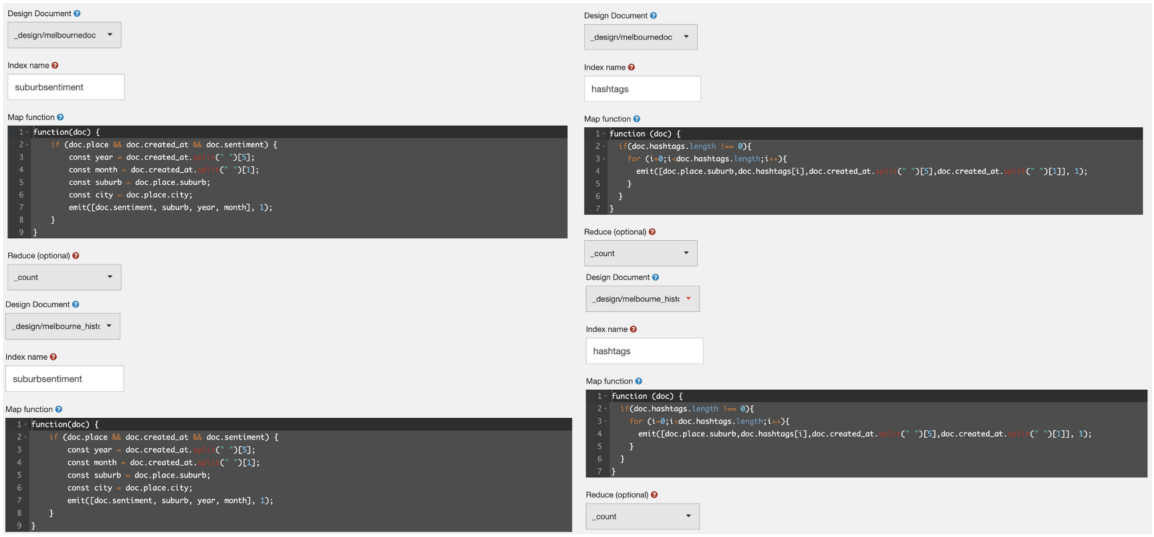

Figure 3.8: Views for Melbourne Data

The most obvious strong point is after Reducing step, the amount of data which satisfies the request from front end decreases dramatically.

3.3.3 Website

Our web server is based on a traditional and flexible frame: CSS + html + JavaScript. In our system, web server works focus on displaying our processed data which passing by MapReduce algorithm. Moreover, to display the real-time data for Tweets in Australia, web server is set to data in read CouchDB automatically once a day.

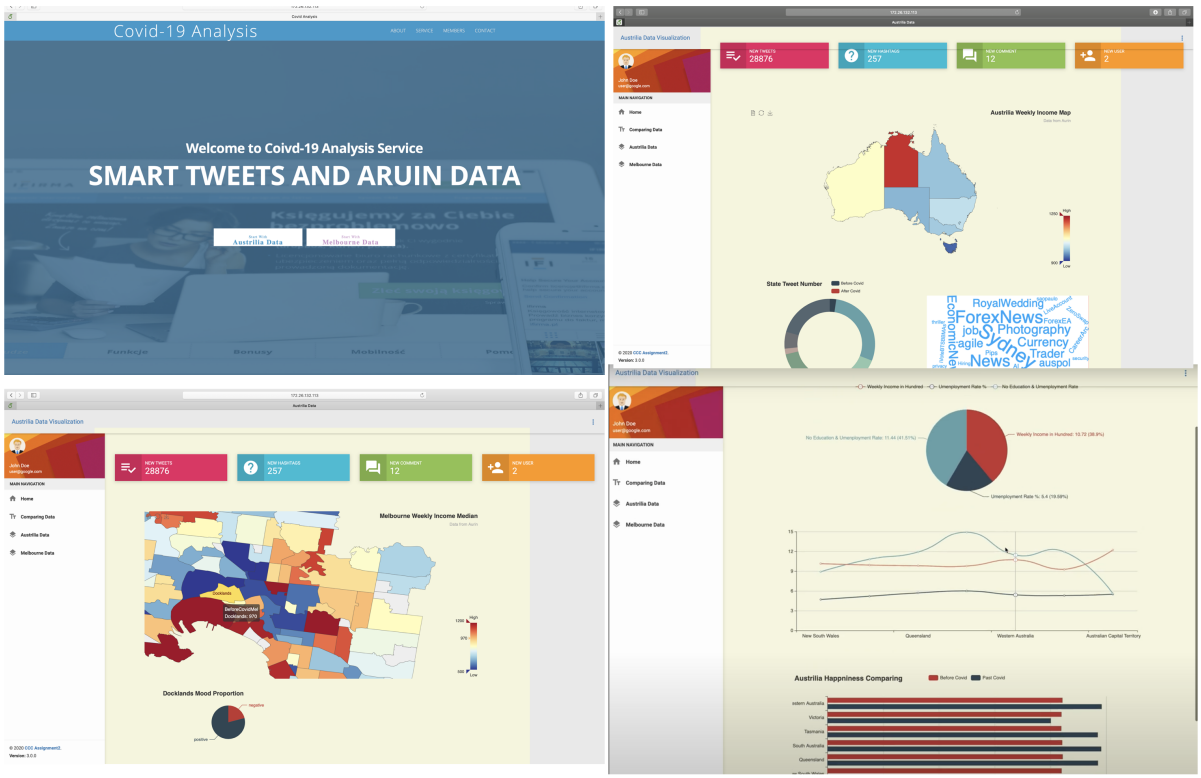

Figure 3.9: Pages of Our Website

As Figure 4.2, there are four main pages in our website: Home page, Australia Tweet page, Melbourne Tweet Page and Data Analysis Page.
And the details of each table which based on Apache Echarts[9] in our project are clarified in next Chapter.

[9]Echarts: https://echarts.apache.org/zh/index.html

4. Life of People in Australia

To lustre how life of every individual in Australia has been greatly impacted by the pandemic, there are two kind of data used in out project, the official data provided by AURIN, and social media data crawled on Twitter or provided by third party. The AURIN data we used is Labour Force status data for 15-24 year old released by Australian Bureau of Statistics on Dec. 20, 2018 and 2016 Australian census.

4.1 Australia

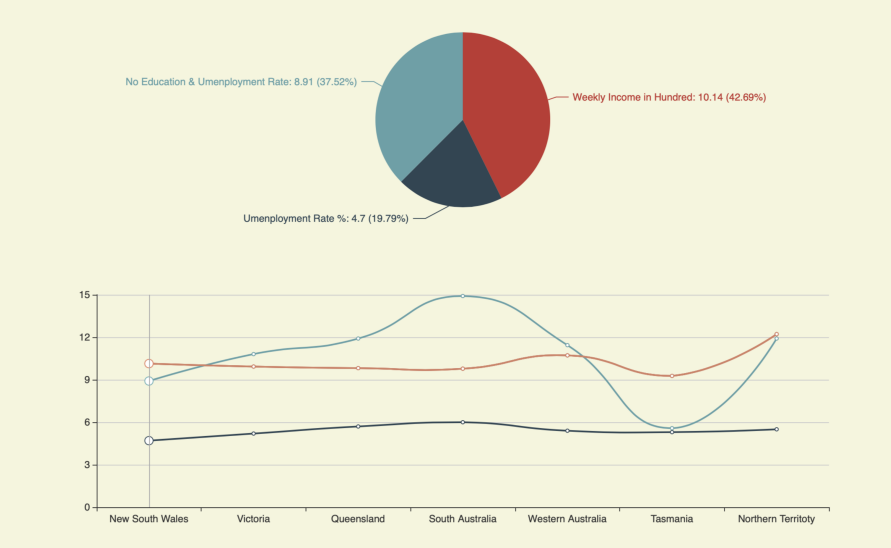

Figure 4.1: Official Data of Each State

The official data provided by AURIN is displayed in a pie chart and a line chart. The Figure 4.1 is used to illustrates the change of Tweet impacted by Income, Employment and Education.

As Figure 4.1 shows, the economics distributed differently in Australia. And highest median weekly income hap pend in Northern Territory, and with lowest employment rate. From this data inspired, we tried to do more research on the Tweet data comparison of this area, however, the data provided by the third party dose not cover any historical Tweet data for Northern Territory.

4.1.1 Tweets Impacted by COVID-19

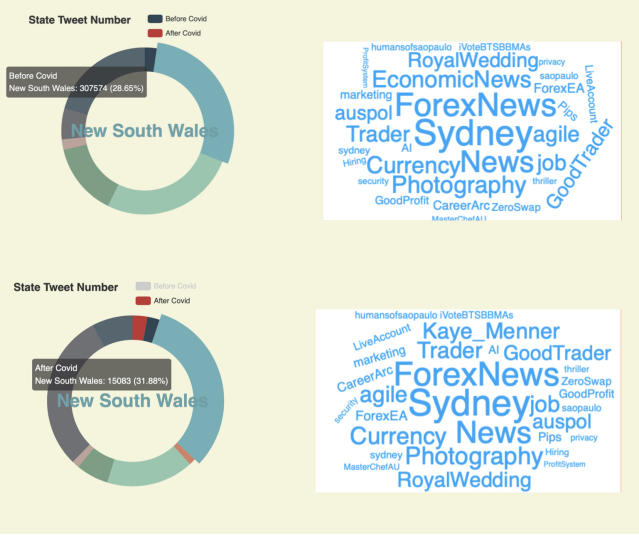

Figure 4.2: Hashtag Comparison and Amount of Tweets Comparison

Affected by COVID-19 and with the self-isolation, as we can see, the number of the Tweet sent by individuals in Australia decreased significantly. Although the percentage of each state is roughly the same, the whole amount of Tweet shrink to 1/10 of the original.
Sentiments of tweets is a standard that was chosen by us to value the impact of COVID-19. When dividing the whole Australian into states, we found out that the COVID-19 did not effect of the positive rate of the tweets that people sent. It's 84.2% before the pandemic, and 83.9% after, only a slightly decrease.

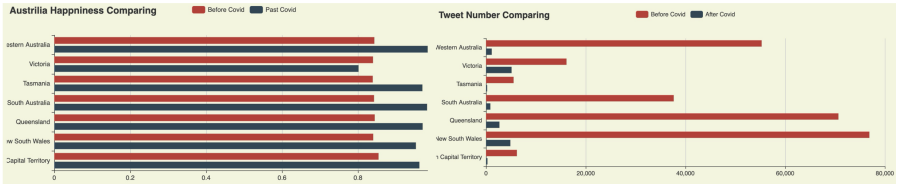

Figure 4.3: Amount of Tweets Comparison

4.2 Melbourne

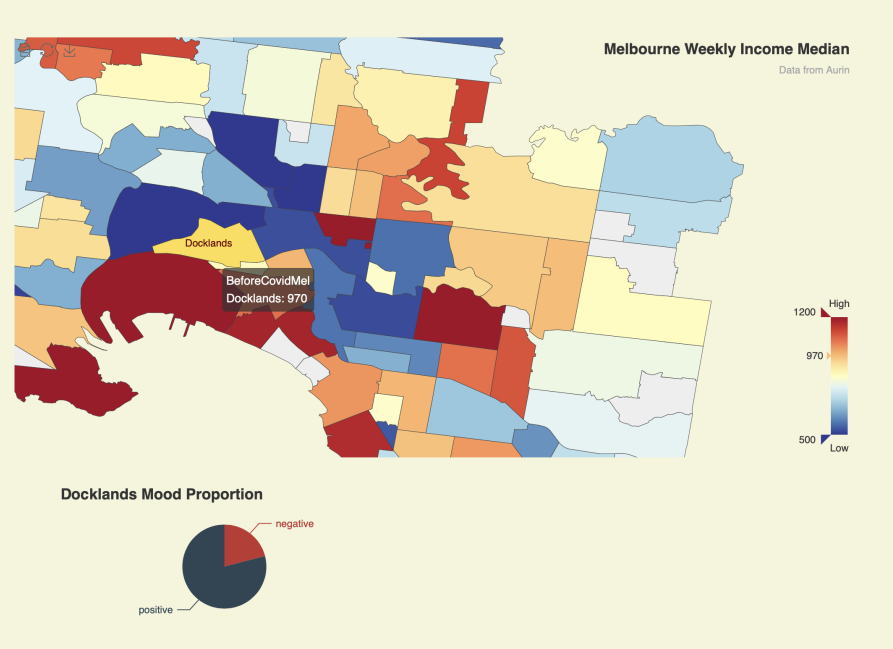

Figure 4.4: Median Weekly Income of Melbourne

The figure 4.4 shows the median meekly income distributed among Melbourne map.However, with the technology and skill constraint, there are around 60 suburbs are displayed in this map (there are close to 370 suburbs provide by AURIN official data. The red color index the richest(highest) weekly income individually, which central at the corner of this map.

As mentioned before,although we already built Views for the Hashtag analysis and statistics of each suburb, the virtualization is not displayed on out web-page. The main reason of giving up doing this virtualization is, impacted by COVID-19, the number of Tweet sent by individuals in Australia shrinks dramatically. Thus, there are not enough data to compute a meaningful word cloud as showed in Australia page.

Focusing on Melbourne, it seems the positive rate decreases more than the whole country's scenario. It's 85.6% before the pandemic and 81.6% after the pandemic. The reason why Melbourne's positive rate turn out like this may be the lockdown policy affects the city life far more than the country life, which leads to less positive tweets of people lives in Melbourne.

5. Evaluation

5.1 Limitation

As mentioned before in Section 4.2.1.2 and 4.2.1.3, there are two solved technical limitation in Mining Twitter: Disconnection and Twitter Replication. Despite those two limitation, the other limitation of research cloud is, it is not that easy for all of the team members to access. For the team member who is currently in China, two VPNs are needed to access to the website that is maintained in the research cloud. The high latency (sometimes even cannot connect) makes the synchronization among the team members difficult to achieve.

Another limitation is the scalability. In this project, the data is expanding all the time. Sometimes, the limitation of the instance's storage can be touched by the expanding data. By that time, if we want to arrange more storage for the instance, shutting down the instance then re-locating the volume is necessary. Which means all the data and existing programs have to be shut down, and re-deploy in the new instance. When the amount of data is large, the job will be almost impossible to achieve. We mitigated the limitation by utilizing the cluster characteristics of CouchDB, which can automatically scale the data from node to node, and mitigate the shortage of the storage space.

5.2 Pros and Con's of UniMelb Research Cloud

UniMelb Research Cloud is a Research Cloud running on the OpenStack, which is a free and open-source software platform for cloud computing. Web-based dashboard, script based tools and RESTful API are used to manage the resources provided by this IaaS[1] flatform.

There are several strengths using the cloud in this project. The first and foremost is the user-friendly: with clear guidance, creating instances and selecting the

[1] Infrastructure-as-a-Service

configuration are straightforward. Also, volumes could be attached or detached accordingly our requirement. The second advantage is the internal network between instances is fast. This high-speed communication ensures the efficiency of each retrieve. Lastly and most importantly, *UniMelb Research Cloud* provides the benefit of quick deployment. The entire system could be completely functional within few minutes. with lots of utilize tools' API (such as Ansible, Docker), there is no need to build our environment instance by instance. And this is the foundation of building a dynamic scaling computing system.

If any deficiency has to be stated, the first drawback is: the preview of data which download on *UniMelb Research Cloud* is not available, which brings some inconvenient in data process step. Moreover, we also feel struggling in debugging on this cloud research platform. To be more specific the error message is too short to help us modify our programs. The third one is the lack of enough computing capacity. The insufficiency of computing capacity is more apparent in MapReduce step. Then inadequate of the supply of the cloud resources is the last disadvantage. In theory, this cloud system should supply an enough and unlimited amount of resources by adding more new nodes. However, in practice, we do meet the overload in building this system.

In summary, as a free cloud platform, *UniMelb Research Cloud* is relatively stable and reliable, which is recommendable for students or researchers. And a monitoring system is required to guarantee all systems are operating smoothly.

6. Appendix A:

6.1 Deployment Server User Guide

Our team uses the local machine as a deployment server to deploy our system. Most of works are down automatically by Ansible. The system is deployed to UniMelb research cloud. It may be able to extend to any OpenStack cloud platform but need extra effort to adjust the setup process.
Set up environment: on mac OS Catalina: 10.15.4, Ansible version: 2.9.6, python version: 3.0

Step 0: Create key pair and get OpenStack RC file from the cloud dashboard

Login to UniMelb research cloud dashboard Create key pair in Computer: Key Pairs part. Download your public key to local host as xxx.pem. Reset your OpenStack password in User. Reset Passwd and save it Download OpenStack RC file from User OpenStack RC File and change the file name to openrc.sh.

Step 1: Get the project source from GitHub

Get the project source from:
`https://github.com/acherking/2020S1_CCC_P2.git`
`Command-line: git clone https://github.com/acherking/2020S1_CCC_P2.git`

Step 2: Setup your openrc.sh and change the name of public key file in Ansible part of the project.

Copy your xxx.pem public key file to key_pair directory in the project.
Change the path of public key in the files:
run-prepare-instance.sh and run-prepare-system.sh.
The path is denoted after '-i' option.
Replace the openrc.sh in Ansible to your openrc.sh.
working directory: %Your_Working_Space%/2020S1_CCC_P2/
`Command-line:`
cp [path of xxx.pem]/xxx.pem ./key _pair/
vim ansible/run-prepare-instance.sh
vim ansible/run-prepare-system.sh
cp [path of your openrc.sh] ./ansible/

Step 3: Install Ansible on local machine

Install Ansible on your machine.
Please follow Ansible's web page install it.
[1] working directory: %Your_Working_Space%/2020S1_CCC_P2/
`Command-line:`
brew install ansible vim ansible/run-prepare-instance.sh
vim ansible/run-prepare-system.sh
cp [path of your openrc.sh] ./ansible/

Step 4: Create system instances and prepare them
Step 4.1 Create three instances

Connect to UniMelb's cloud network via VPN.
Using Ansible automatically create three instances:
demo-1 demo-2 demo-3
Every instance contains two volumes: 20G,40G; same security groups
Enter your OpenStack passwd which saved in Step 0
Input your sudo passwd on local machine
Install Ansible on your machine.
working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/
`Command-line:` ./run-create-system.sh/

Step 4.2 Prepare these instances
After creating instances successfully, you need to prepare them to make them ready to construct the system. It will include configure proxy, mount volumes, install docker and couchdb and configure docker proxy.

Edit the hosts file to change the IP addresses in the system part
Get the IP addresses on cloud dashboard working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ Command-line: ./run-prepare-system.sh

Step 5: Construct CouchDb Cluster
There is a container running CouchDB in each instance. You have to execute some commands to make them become cluster. These commands are provided by Couchdb API, you can run it on your local machine.

working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ Command-line: vim ./couchdb-cluster.sh ./couchdb-cluster.sh

Step 6: Configure Docker Swarm

Make demo-1 the swarm manager; demo-2 and demo-3 the swarm worker working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ replace the IP1,IP2,IP3 to your demo-1's IP demo-2's IP demo-3's IP in the command Command-line: vim ./swarm-init.sh ./swarm-init.sh
get the token from swarm-init.sh and save it in swrm-add-workers.sh vim ./swarm-add-workers.sh ./swarm-add-workers.sh

Step 7: Deploy web server and Twitter Harvester on the Docker Swarm
Both Web server and Twitter Harvester are images which store in docker-hub.

working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ replace the IP1,IP2,IP3 to your demo-1's IP demo-2's IP demo-3's IP in the command Command-line: vim ./swarm-run-webserver.sh ./swarm-run-webserver.sh vim ./swarm-run-harvester.sh ./swarm-run-harvester.sh

6.2 System Scale-up Guide

The system can be scale-up by adding new instance to it.
Step 1: Create and prepare new instance
Using Ansible to create and prepare a new instance demo-4.

Using Ansible to create and prepare a new instance demo-4 working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ replace the IP1,IP2,IP3 to your demo-1's IP demo-2's IP demo-3's IP in the command
edit the IP in hosts new part Command-line: ./run-prepare-instance.sh

Step 2: Add new instance to CouchDB Cluster Add the couchdb container in demo-4 to the couchdb cluster.

working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ replace the IP1,IP2,IP3 to your demo-1's IP demo-2's IP demo-3's IP in the command Command-line: vim swarm-add-worker.sh ./swarm-add-worker.sh ssh -i xxx.pem ubuntu@$IP1 sudo docker service scale web-server=4

6.3 System Scale-down Guide

Step 1: Remove a node from CouchDB Cluster
Remove the CouchDB container in demo-4 from the CouchDB cluster.

Remove the CouchDB container in demo-4 from the CouchDB working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ replace the IP1,IP2,IP3,IP4 to your demo-1's IP demo-2's IP demo-3's IP in the command Command-line: vim swarm-add-worker.sh ./swarm-add-worker.sh curl http://IP1/_node/_local/_nodes/node4@IP4 "_id":"node4@yyy.yyy.yyy.yyy","_rev":"1-967a00dff5e02add41820138abb3284d" use the _rev to finish next command Command-line: curl -X DELETE c "http://IP1/_node/_local/_nodes/node4@IP4?rev=1-967a00dff5e02add41820138abb3284d"

Step 2: Remove the node from docker swarm

Remove the instance from Docker Swarm Cluster working directory: %Your_Working_Space%/2020S1_CCC_P2/ansible/ replace the IP1,IP2,IP3 to your demo-1's IP demo-2's IP demo-3's IP in the command `Command-line:` vim swarm-leave.sh ./swarm-leave.sh

Step 3: Delete the instance
Finally, you can release the instance on dashboard. You can also delete the volumes of the instance.

6.4 Links

Link to source code git repository:
`https://github.com/acherking/2020S1_CCC_P2/tree/master`
Link to website video:`https://www.youtube.com/watch?v=w0pdOUFSguc`