COMP90024 Cluster and Cloud Computing Assignment 2 Report

Team 26, Melbourne, Shixun Liu, 766799

Team 26, Melbourne, Yuan Bing, 350274

Team 26, Melbourne, Renyi Hou, 764696

Team 26, Melbourne, Mark Chun Yong Ting, 805780

Team 26, Melbourne, Kaiqing Wang, 700275

Table of Contents

1.Introduction

The users of social media have been growing exponentially in the past ten years due to ubiquity of smartphone and other devices. Social media platforms such as Facebook, Twitter, Instagram now plays a significate role in people’s life. General public has been sharing activities, events, personal feelings and opinions on social media, which makes the social media data valuable to be studied for varieties of analytics scenarios. According to internetlivestats.com, 500 million tweets are tweeted per day. As results, large and informative datasets of tweets would be able to provide us with infinite possibilities to obtain insights of residents’ daily lives and opinions. This Cloud-based solution provides an integrated analysis system from harvesting real-time and historical tweets and executing several stages of analysis to delivering results by data visualisation on a web-based front-end. The majority of this system was developed with Java and Python language then the system can be automatically deployed on the NeCTAR Research Cloud. CouchDB has been used as a centralised database to store all harvested tweets and all final analysis results datasets. This report briefly introduces the role of each team member in the development of the designed system. The report also outlines the design and architecture of the system and then provides the method we used in twitter harvester and series of analytic scenarios that have been considered. The implementations of the front-end web applications are disclosed in this report. Finally, the advantages and disadvantages of the using NeCTAR Research Cloud are discussed. A deployment method using scripts and a simple user guide for testing are included in the end of this report.

There are many new technologies we have never used before, so all our team members spent a great amount of time on research solutions and each team member has picked a part of the process to work on. From the very beginning Shixun (766799) worked on how to run the instances through the NeCTAR and the configuration of the virtual machines, then later he worked on how to set up cluster of the CouchDB. Mark (805780) worked on using Twitter API to build harvester. Then he improved the efficiency of the harvester and developed a method to remove the duplicates tweets. Kaiqing (700275) worked on the deployment and improvement of the twitter harvesting application, setting up a RESTful server and visualisation of data on a web page. Renyi (764696) worked on processing sentiment analysis on all twitters in collections, Bing (350274) worked on using the datasets from AURIN and sentiment analysis to develop a range of analytical scenarios then display results visually on a map by using OpenMap and leaflet. Additionally, Shixun and Mark have been working together on how to use Jgroups and Ansible to deploy the system.

2.System architecture and design

This Cloud-based analysis system includes three necessary application: Data harvester, scenarios analysis, and visual result web page. The overall architecture of the system is shown in figure 1 below. In our team, each group member is good at different programming language, so we use different language to implement our system. In Twitter harvest part, we use Twitter4J library which encapsulated the Twitter API and Java as the basic programming language developed a twitter harvester running on the instance created by the NeCTAR Research Cloud. Using Search API and Steaming API harvest the historical and real-time tweets as much as possible and save the collected data in the corresponding CouchDB, these CouchDB also created on the instance works as a cluster. Then, based on the big dataset harvested in the first part also using the AURIN data and statistic data from the internet we designed a range of analytic scenarios, using Python as the programming language, and save the analysis result into the CouchDB. Finally, the front-end web application would provide the user with a visualizing the result of the analyzed scenarios. In the server side, we use the RESTful design based on the Jersey framework. The web server would response the GET request from the client with the JSON data, the result of scenarios analysis would be displayed through the graph plug-in's interface. The detailed function implementation would describe in section 3.

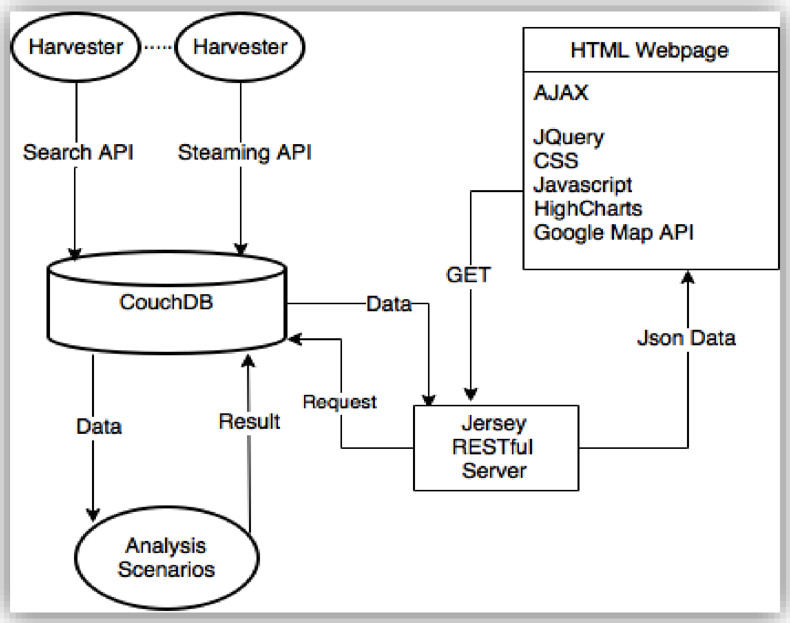


Figure 1 System Architecture

3.System Functionalities

In this section, we will be focusing on explaining the tools that we used to implement the functionality as well as the methodology of each implementation together with the reason we choose to use it. Some figure will be illustrated to support the explanation of the functionality.
We split this section into three subparts:

- Twitter Harvest
- Analytic Scenarios
- Front End Web Application

3.1 Twitter Harvest

Based on the official Twitter API web page[1], they mentioned a lot of twitter library which supported for each different programming luggage, such as tweepy[2] for Python, twitteroauth[3] for PHP, twitter4j[4] for Java, etc. Since we use choose to use Java as the base programming language throughout most of the system implementation, we choose twitter4j library for supporting our twitter harvesting tools. Although twitter4j is an unofficial library, it provided us most of the required features in the twitter harvesting process.

First, from the past research or experience, the total tweet data that Twitter database owned consider to be substantial, and a number of tweets that we gather for the purpose of analysis are required to be massive to form an accurate result, which predicted to be more than around 100GB in size or to be said more than 10 million tweets in total. To make sure the system meet the effectiveness mentioned above, we need to implement a low-cost solution so that we can gather a large amount of tweet data in a short amount of time. In the following section, we will be explaining how we meet the expected result by splitting the twitter harvesting process into two sub-process, as well as how we set it up in the cloud environment to achieve the best outcome.

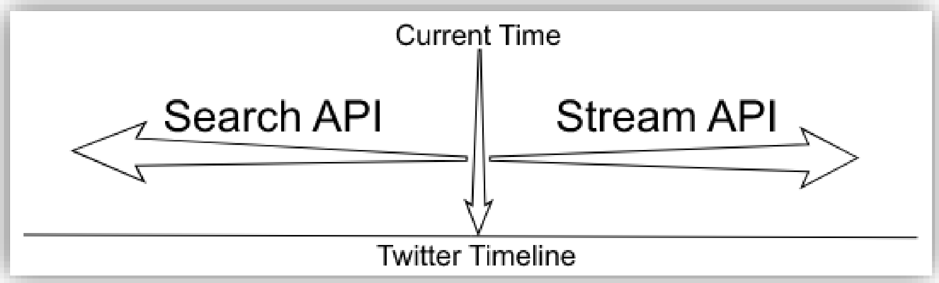


Figure 2 Twitter Harvest method based on Timeline

[1] Twitter API - https://dev.twitter.com/resources/twitter-libraries
[2] Tweepy - https://github.com/tweepy/tweepy
[3] Twitteroauth - https://github.com/abraham/twitteroauth
[4] Twitter4j - http://twitter4j.org/en/index.html

Before we head forward to the detail explanation of the background process, I will first roughly describe and differentiate between "Search" method of tweet data and "Stream" method of tweet data. As we all know about the timeline, all the data happen to have a timestamp and data from the past or future will be using a different method to gather. For easier understanding, I illustrate out the figure below to explain the purpose and reason of using search API and stream API.

During the twitter harvesting, it will have a timestamp attached to each of the tweets, which we imagine that these massive amount tweets are irregularly spread through the twitter timeline. So, whenever we start the twitter harvesting process, we suppose that will be the current time event in the timeline. From the figure, everything on the right from the current time will be the future tweet data from twitter, and without a doubt, everything on the left will be the past tweet data that was stored in twitter database. On the official twitter API documentation, they provide stream API for gather future tweets and search API for past tweets. The detail description of each API will be elaborate later.

3.1.1 Twitter Token Authentication

For any Twitter API library mentioned in the documentation, the first step of the process is to connect to the twitter API through the account to get the token for accessing the twitter database. These accounts can be created through twitter developer website[5] and required us to note down both the consumer and token key after account creation. Then, connect to the Twitter API through account configuration builder together with enabling Application Only Authentication settings. The purpose of this step is to increase the rate limit from 180 query per 15 minutes to 450 query per 15 minutes during gathering past tweets from search API. So that, we can gather more tweets compare to normal usage in the amount of time. The result comparison chart will be provided below for clearer differentiation.

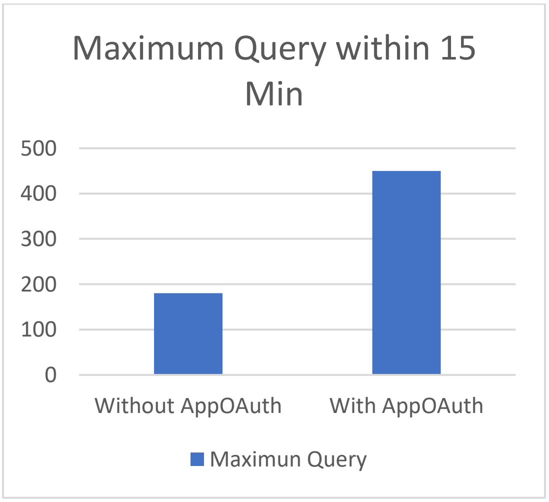


Figure 3 Query Called

[5] Twitter Developer - https://dev.twitter.com/

3.1.2 Search API

Twitter search API is a part of the Twitter REST API. It provides the functionality to request the tweets from twitter database for the system. The library Twitter4j search API using the logic of executing a “GET” request URL through HTTP socket and compiled it into JAVA programming language based method. To request for the tweets, we need to use the twitter4j “Query” class together with our required information such as location or keywords as the parameters and retrieve for the related accurate result.

As we previous mentioned about the efficiency of the system is the most important key for us to get the most tweets out from Twitter API in short amount, we will need to add in some of the technique to resolve the problem. In the first place, we find that official Twitter API have previously announced some limitation to the search API due to prevent some internet traffic problem or security attack. However, the limit is only set for 15 minutes, if the event or any process exceed the limit, it will be reset after 15 minutes. From figure 2 in section 3.1.1, we show the difference of amount that query can be made with or without application authentication. The user could make up to 180 query per 15 minutes by default, but after the application authentication, we can make around 450 query within this 15 minutes. Although, we still need to set some restriction to prevent the system getting to this limit.

Secondly, another part that can be tuned to enhance the performance is to set the count limit of each query. For example, the twitter API provide 15 count limit for each query by default, which means we will be able to get 15 tweets per query. Thus, we try to increase the amount of the count limit to the maximum limitation of Twitter API which is 100 count limit per query. In the result, we may get around 450 query times 100 count limit in total for every 15 minutes instead of 18000 per 15 minutes. The figure below shows the differ between the tweet amount we could gather in 15 minutes after doing the technique stated above.

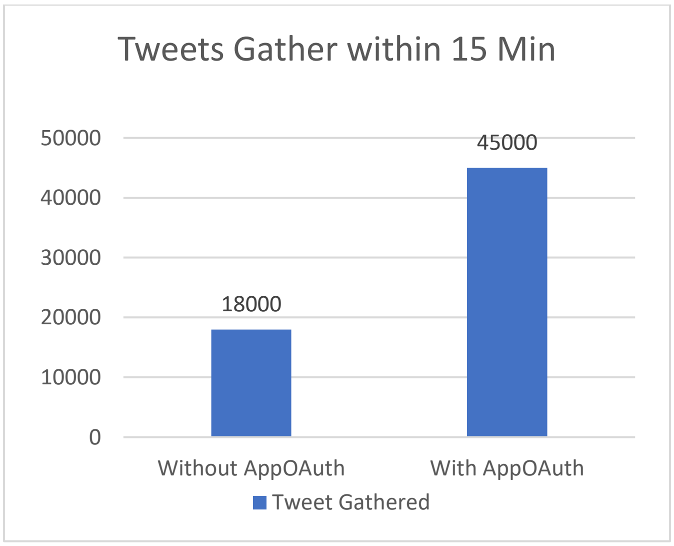


Figure 4 Tweets Gathered

At last, since we are always making a similar query time to time, we might be getting the same tweets, and this will make the system to fail as well as having the worst twitter harvest functionality. Again, referencing from twitter API documentation[6], we find a special method stated that would resolve this problem and highly cut down the repeatability of tweet data. In the introduction section on 3.1, we understand that the tweets data are storing on a database with a timestamp on the timeline, the technique we need to implement will be setting the tweet id range from the "sinceid" method from Query class in twitter4j each time we call the query. In detail, when we get a set of tweets from one query, we retrieve the maximum tweet id among the list. From the next query, we use the tweet id and find every tweet in which id is more than the maximum id that we store previously.

From all the solution that we implement above, we can ensure that we can get the most number of unrepeatable as well as accurate tweets data within a short amount of time.

3.1.3 Stream API

Streaming API[7] is the interface that supports functionality for getting tweets which happen live on the timeline. The implementation of stream API is comparably easier than search API, simply start up the tweets listener and filter all the incoming tweets with information we need, in this situation we will be using location details as our parameter. Then, wait for the live tweets to come in, and we are good to go.

3.1.4 Twitter Harvest Deployment and Store

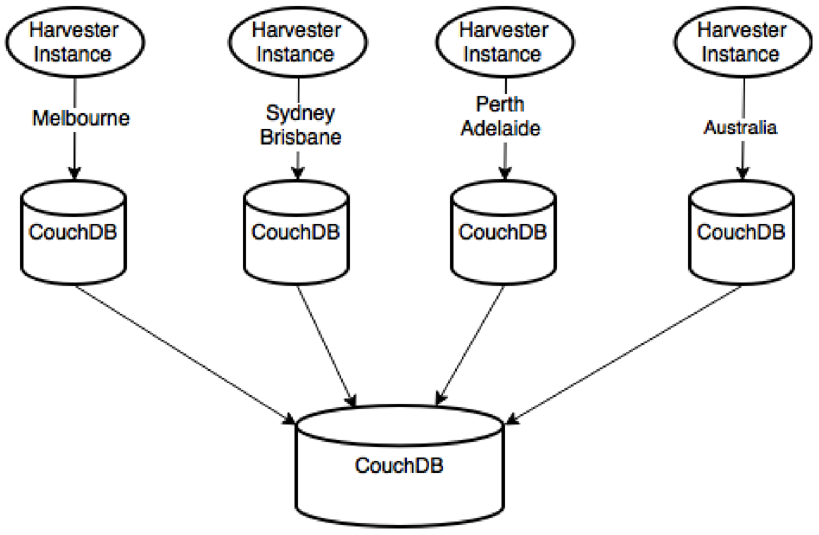


Figure 5 Twitter Harvesting

[6] Working with Timeline - https://dev.twitter.com/rest/public/timelines
[7] Streaming API - https://dev.twitter.com/streaming/overview

As mentioned above, for one Twitter development account Twitter API provided 450 query times and 100 count limit per query. However, if we use just one account for searching tweets, about 2 to 3 minutes the account would be blocked for 15 minutes, so we apply for as many account as we can, while one account has been blocked another account would continue to search the tweets based on the “sinceId” provided by the previous account. This would ensure that the twitter harvesting application would run all the time without waiting for the 15 minutes’ block time. As shown on figure X In our Cloud-based system we offered 4 instances for running the harvesting application; we deployed 5 twitter harvesters in 3 instance using Search API search tweets from 5 metropolis cities in Australia: Melbourne, Sydney, Brisbane, Perth, and Adelaide, save the searched tweets into the corresponding CouchDB. And 1 harvester using Steaming API harvest the tweets within Australia and save the tweets into 5 clustered CouchDB based on the attributes of place.

3.2 Analytic Scenarios

Analytic Scenario is designed to analyze presupposed events with existing known information. It demonstrates the summary of the collection of existing data and the observable trends of the future. In this case, we developed the following topics of analytic scenarios to explore the correlation of collected tweets and AURIN urban information.

- The amount of tweet among five largest cities in Australia in different periods of time:
  In this scenario, the team group explores the number of sent tweet counts in different time slots in five largest cities in Australia. The idea of this analytic scenario is to state the lifestyle and the habit of sending a tweet of Australian. As the raw tweet fetched, the “created_at” is an attribute in the tweet JSON format. The created time could be extracted and analyze it. The specific situation is described in Section 4.

- Comparison of the percentage of different sentiments with per capita GDP in five largest cities in Australia:
  In this case, we explore the two things: the percentage of sentiments, the per capita GDP in five cities. What it’s designed to do is to find the correlation between levels of wealth and the feeling person has. We assume that people would be happier and more confident when they are rich and wealthy. In order to perform sentiment analysis, Stanford CoreNLP software is implemented in this research by using hidden Markov model. Meanwhile, the per capita GDP could be accessed in AURIN data.

- Comparison of different device counts in Melbourne suburbs:
  As the tweet fetched, the source detail would be accessed from the JSON form. According to our study, the source from sending a tweet is various. The IOS, Android devices, and third-party applications could be the source to operate the twitter via mobile

app or twitter API. In this scenario, we would study the popularity of different devices in Melbourne suburbs. With the result of the scenarios, some interesting events can be explored deeply. For example, the average price of iPhone series is about $900, and the average price of Android devices is about $400. Presumably, the people who live in the relatively poor suburb, prefer to choose the cheap mobile phone. However, as the result of we studied, some suburbs which young people live in, have the higher popularity of IOS devices than Android devices. The potential reason we speculated is young people prefer to use the high technology devices.

3.3 Front-End Web Application

As shown in figure 1, we provided a front-end web application. For the server side of the web application, we use the Jersey framework design our RESTful services; the web server would receive the request from the client and return the corresponding JSON data from the CouchDB. In the client part, we use the HTML5 for building the web page. While loading the index page, the web client would send "GET" request to the server and get the JSON data to initialize the chart plug-in: HighCharts in the web page.

The front-end web application is written in a combination of HTML, JavaScript, and CSS, it assists in visualizing the large amount of the datasets that have been collected and the final outcome. in order to present data well, and easy to understand, several plug-ins such as leaflet and Highcharts are used to achieve it. The leaflet is an elegant plugin for displaying a choropleth map with a color scale. Google Map is obtained via google API has been chosen to be the base map, then leaflet is able to present layer of suburbs map at the top of the base map. The suburbs polygons on the map are in different shades of color in order to present the distribution of varieties of socio-economic attributes. Each polygon is a JSON line in the GeoJson file. The major advantages of having a choropleth map are: it is easier for the viewer to detect any relationships between attributes and geographic information or understand a general overall pattern of the data presented relatively quickly.

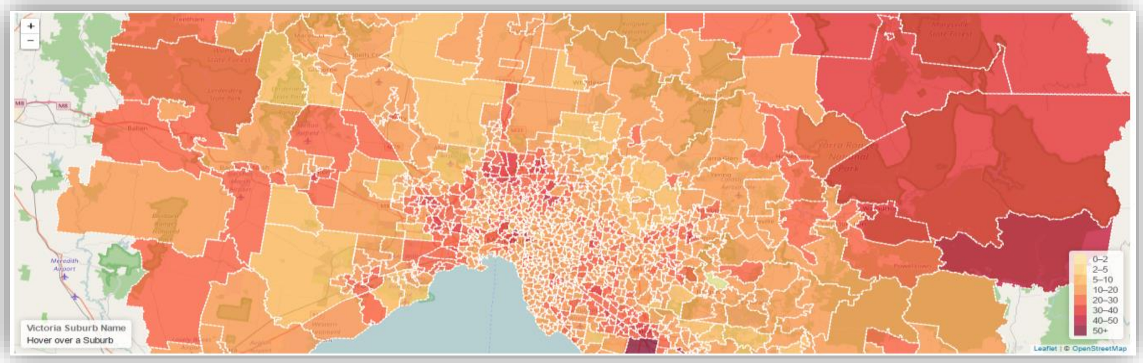


Figure 6 Google Map

The GeoJson dataset has been selected for this project is Socio-Economic Variables by PBC for 2013 Australian federal election which has the 2011 Census of Population and Housing data are spatially allocated. As shown in the caption below: this is a choropleth map for percentage people has low income who earns $600 or less. The darker the shade of the color presents higher the percentage of low-income earners reside in that area. When the mouse hovers over the area, it will also give other socio-economic details, for instance: percentage of residents live in own house, the unemployment rate in that area, etc. as expected, there is a negative correlation in between percentages house owners and low-income earners. Besides, the low-income area also has a connection with high unemployment rate. Additionally, some of the low-income areas are all clustered around university due to the high concentration of student residents.

4.Result of Twitter Analysis

4.1 Sentiment Analysis

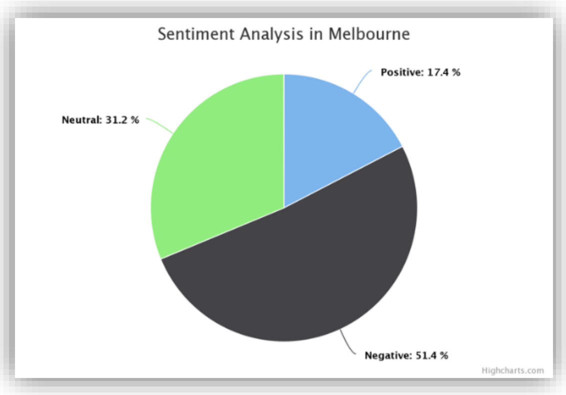


Figure 7Sentiment Analysis in Melbourne

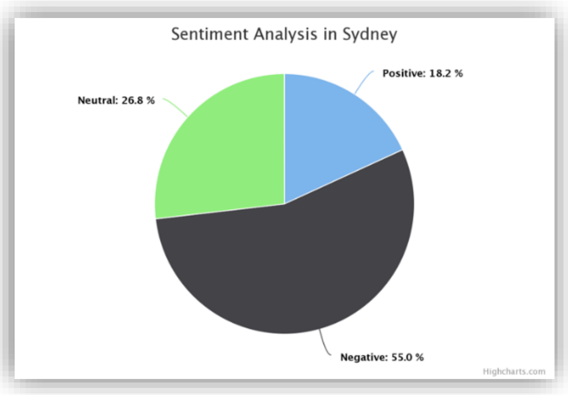


Figure 8 Sentiment Analysis in Sydney

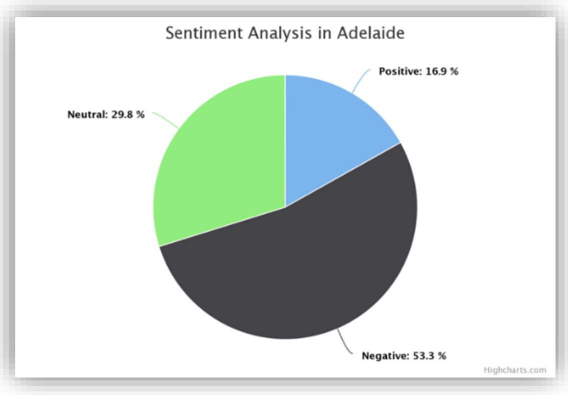


Figure 9 Sentiment Analysis in Adelaide

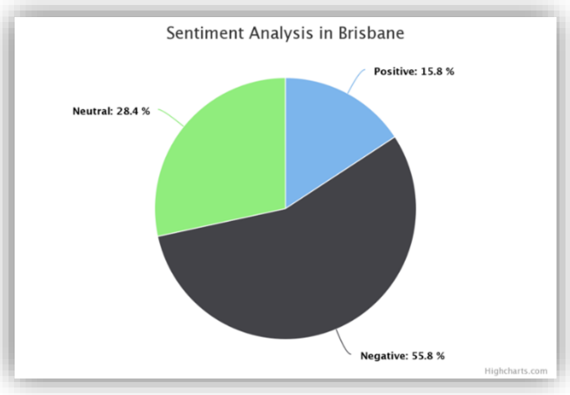


Figure 10 Sentiment Analysis in Brisbane

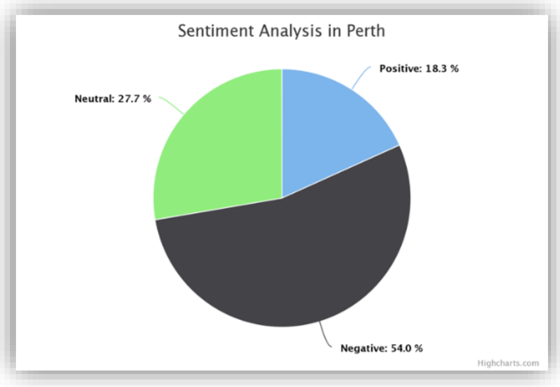


Figure 11 Sentinment Analysis in Perth

All data are presented in the format of percentage in order to be normalized for the comparison. Overall, more than 50% of the twitters are identified and categorized as negative by prediction model in all five major capital cities. The proportion of negative twitters has marginal difference across the five cities that have been studied. On average, people in Sydney and Perth twit more positively than people live in Adelaide and Brisbane which indicates that people in those 2 cities may have joyful lives. Few other socio-economic objects are studied and compared to these 5 cities for correlations and connections.

4.2 Positivity vs. Average Annual income

The positiveness of overall twitters in all five cities has a positive relationship with the average annual income of the residents. In the city of Great Perth, residents earn significantly higher than the rest of the country tend to send more positive twitters. Based on the statistics figure provided, it shows that the proportion of the negative twitters is smaller, and proportion of the positive twitters are slightly larger. Compare to Brisbane, although it is a major tourist city, more negative twitters are published, and its average annual salary is also below Australian average annual salary.

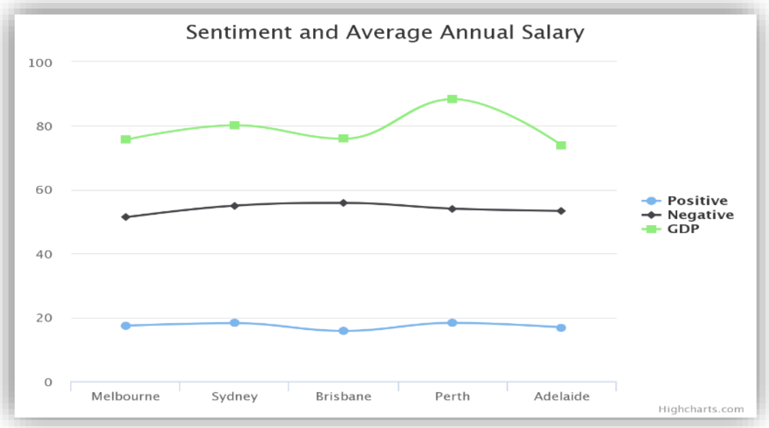


Figure 12 Positive and GDP

4.3 Number of tweets in different section of Time

The number of twitters have been sent out by its users throughout the day has been adjusted according to area time zone, for instance, Melbourne, Sydney, Brisbane fall in Australian Eastern Standard Time Zone (+10:00 ASET), Perth is in (10:00 AWST) and Adelaide is half an hour behind AEST. The graph below presents the volume of twitters that have been sent by users in 3 hours increment throughout 24 hours. The graph implies that people sends less twitters in between 3-6 am and there is an up trending on volumes of twitters from morning 6 am and in between 6 to 9 pm, it reaches its peak volume of the day. All five capital cities have same trends regarding to the number of twitters have been transported which is expected.

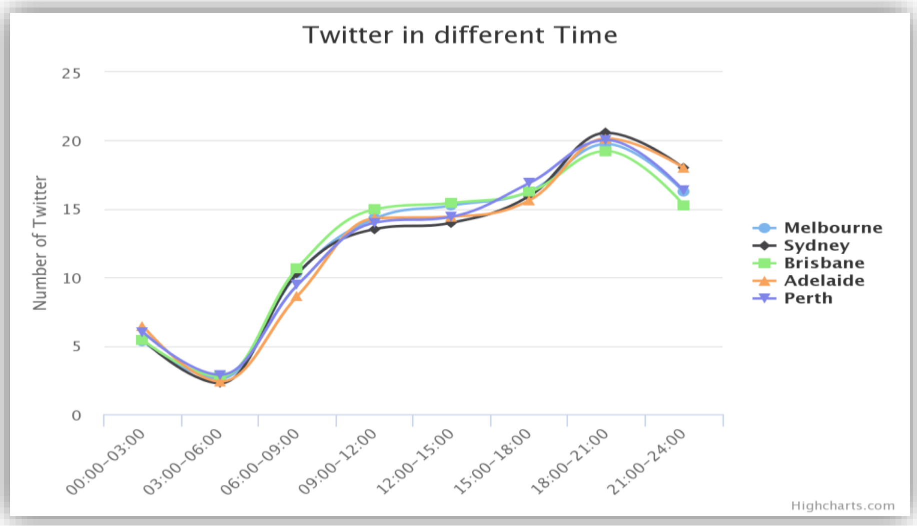


Figure 13 Twitter in different Time

4.4 Correlation between users' smart devices and tweets source

The figure 14 pie chart presents the percentage of platform residents are using when they send twitters. Almost 50% of the twitters come from iPhone device and a small proportion of the twitters come from iPad or other iOS devices. Therefore, more than half of the twitters are sent from iOS devices and approximate one fifth of twitters come from android devices. Some other popular applications such as FourSquare and Instagram contribute approximate another one fifth of the twitters volumes. From figure 15, there are approximately same number of android and IOS active devices in Australia. After compared these two pie charts, we are able to conclude that people using Apple devices are twice more likely to send twitters than people using android devices. IOS platform is the most popular platform for sending out twitters.

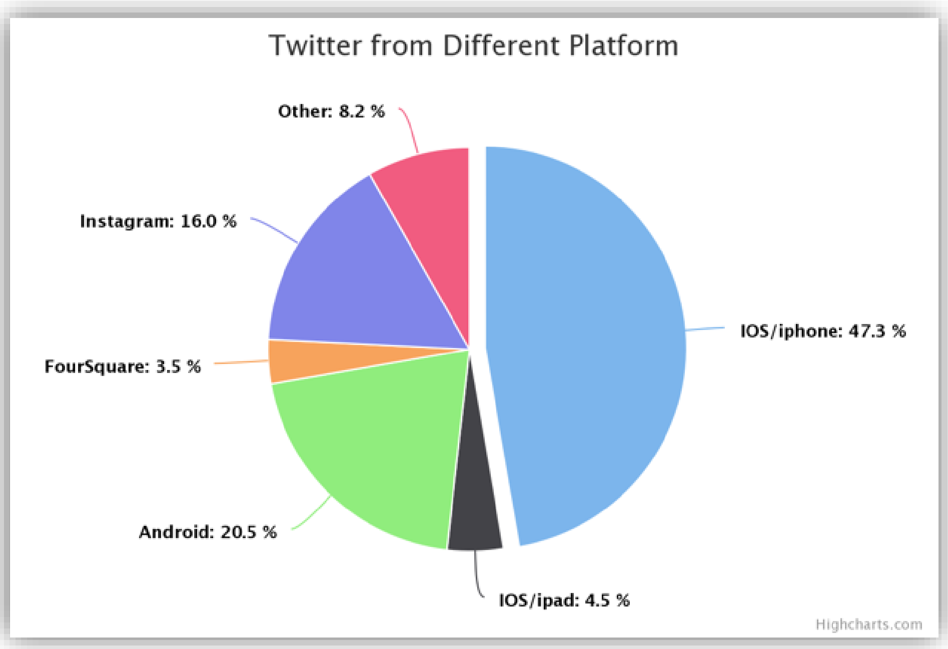


Figure 14 Device User

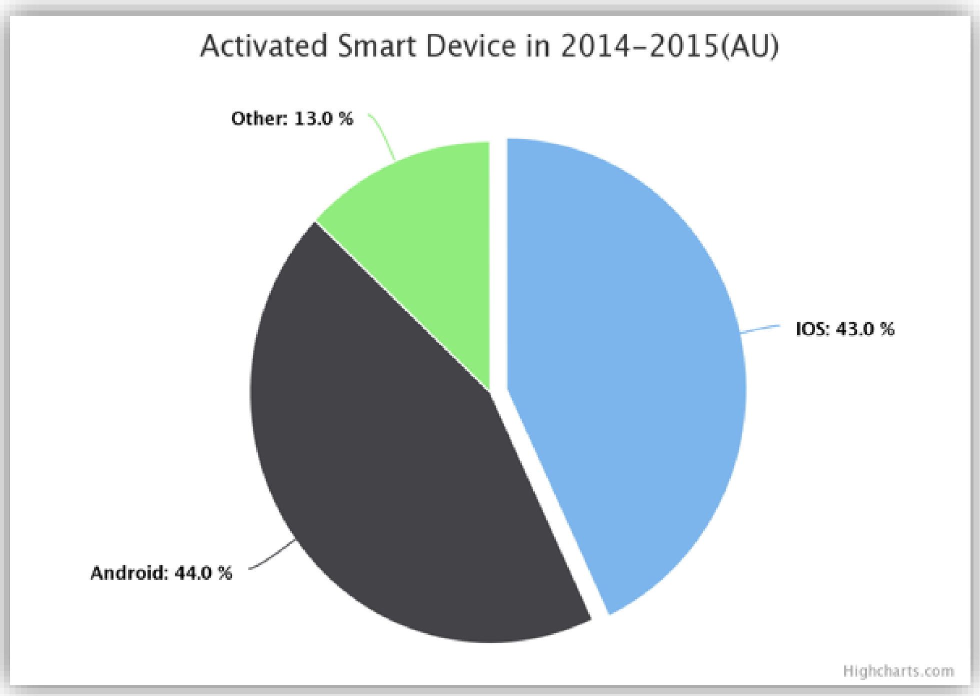


Figure 15 Platform user

5. NeCTAR Research Cloud

Nectar research cloud is a cloud computing facility that provides infrastructure, software, and services which allow Australis's researcher to enjoy the functionality of cloud technology. It is using OpenStack [8] cloud computing as a framework, which is the fastest growing cloud computing facility in the moment.

5.1 Automated Configuration

At the beginning stage of the development, most of our implementation is manually setting up, such as create and modify the instance through Nectar official dashboard. Once created, download the key file and connect to the instance by using SSH path. Besides, when setting up the instance environment, we need to download the required kit manually, for example, JDK or CouchDB as well as transfer runnable software through TCP connection and execute the software. It sounded too complicated and required a lot of testing with adjusting some of the value to make it work perfectly. So, we decide to transfer all of those function into an automated function, simply executed the script or software, and we will get the same result. In this section, we split the configuration into two parts, clouds environment creation, and cloud environment configuration.

5.1.1 Jclouds

In the official nectar documentation, they listed out some of the pre-supported third party libraries which provide the usage of the connecting system to the cloud technology, for example, boto API that used for Python language, euca tools priority for Amazon AWS service, etc. All of this using OpenStack API as the reference as we mentioned before, Nectar is implement based on OpenStack. Due to the decision of programming language that we used, we research through the net and decide to use Jclouds API. Jclouds [9] is an open source multi-cloud toolkit that implemented by Apache group. It is a Java-based platform which gives us the ability to create or access applications that are store online across clouds environment.

For our system, we will be using jclouds to handling usage of Nectar Research Clouds. Refer to cloudwatt dev[10]; they clearly demonstrate the use of Jclouds as well as detail tutorial for each of the functionality. Based on the requirement of the system, we implement five of the feature which is creating a new SSH public key, creating a new instance on Nectar cloud, creating new partition volume as well as attach it to the instance and finally list out all the instance that previously created.

[8] Openstack - https://www.openstack.org/
[9] Jclouds - http://jclouds.apache.org/
[10] Cloudwatt Dev - https://dev.cloudwatt.com/en/doc/sdk/sdk-java-jclouds.html

5.1.2 Ansible

In this project, the configuration of each cloud instance is tedious and similar. Instead of manually repeating the deploying work one by one, a powerful IT automation tool, Ansible[11], is utilized to remotely implement the essential operating environment, configuration management, etc. in all instances in Nectar from our local host. It will reduce the time cost and avoid any potential errors made during the manual operation.

According to our solution for this project, the following important aspects will be deployed through Ansible script for cloud instance, including JDK for Twitter harvest program, CouchDB installation, and cluster establishment, binding volume for each instance, Tomcat and XAMPP for the web application. The tasks executed by Ansible are listed below:

- **Wait for SSH ready**: the local host is unable to connect to instance immediately after it has been created, and thus the Ansible need to wait for SSH port come up and start other tasks.

- **Install CouchDB and setup cluster** : the Ansible script downloads and executes the shell script from GitHub[12] for CouchDB installation and automatically modify the local.ini, vm.args and sys.config to enable cluster function of each host. Then the chosen main host will add other hosts to nodes to create the cluster.

- **Create database**: there are five pre-defined databases - 'twittermel,' 'twittersyd,' 'twitterbri,' 'twitterper' and 'twitterade' which will be created after the cluster is setup. The name of the database is not editable as they need to be aligned with the database name defined in the Twitter harvest program.

- **Modify the storage path of data:** the instance consists of two disks, one disk with 10GB and one ephemeral disk with 60GB. With CouchDB first installation, the data will be saved in the default 10GB disk which space is inadequate compared with the potential data size. Thus we will modify the local.ini file to direct the saving path to ephemeral disk**.**

- **Install JDK:** the Twitter harvest program is written in Java, and thus we will set up the JDK for later program execution.

- **Install Tomcat and XAMPP**: the web application is running on Tomcat and XAMPP, and thus Ansible will install Tomcat and start the service of XAMPP.

[11] Ansible - https://www.ansible.com
[12] CouchDB installation - https://github.com/afiskon/install-couchdb

5.2 Pros and Cons

On the late 1950s, computer scientist invented the first project using the centralized system for resources sharing. Since then, this type of technology has progressively improved and become as a core technology in the networking field.

In this section, we will be discussing the pros and cons of Nectar Research Clouds, which is the product of Australian based cloud computing company.

Pros:

- Fee: Since cloud computing was built on the internet, it saves a lot of cost from building a physical computer unit and maintenance. Using the cloud service only require the user to pay how many resources they need. However, for all Australian Researcher, nectar research cloud provides free service with requested resources.

- Personalization: Nectar research cloud let the user customize their own server, for example, the operating system, they volume size, availability zone, etc.

- Resource sharing: When multiple users are joining together as a group, they can share resources or have the right to control the server with each other anywhere and anytime.

- Saving electricity: While using the cloud services, most of the action required less power assumption due to the remote processing. Other than saving cost from power bill, it also helps to create a green environment.

- Access independence: As cloud service is built on the net, the user can access from anywhere with any devices without any fault. Besides, most of them have a special communication protocol and don't require installation to enjoy the services.

Cons:

- Internet Connection: As mentioned before, cloud services are built on the internet. So, one the internet services drop, the user is unable to access or control the cloud server.

- Out of service: When the cloud service provider shut down the services, the user won’t be able to continue accessing the services and the service processed may be cut off some time.

- Indirect access: The internet services provide by ISP may influence the user experiences of using cloud services. Slow bandwidth may cause the communication between cloud to fail without acknowledgment.

6. Testing Guideline

In this section, we will be providing the guideline on how to use the system all the way from setting up the cloud instance to display the result in the front-end web application.

6.1 Nectar Handling (Jclouds)

When starting up the nectar handler software, we will immediately get the following menu option:

```
Nectar Handling Menu
--------------------
1. Create Key Pair.
2. Create Instance.
3. Create Volume.
4. Attach Volume to instance.
5. List instance.
6. Auto creation.
0. Exit.
```

Figure 16 Nectar Handler Menu

Choose any option by typing in the number and hit enter. Now we will be explaining how each of the options works and how to handle it.

6.1.1 Create Key Pair

In this option, the system will let us create a new keypair to be attached to the new instance when we created. This key will be used to access through SSH.

```
Please provide keypair name:
Please provide public key directory:
```

Figure 17 Key Pair Information Request

We will be prompt to input a name to be used for key pair creation as well as a directory public key which we need to create beforehand. For windows user, the public key can be created with "PuttyGen.exe." For Linux/OSX user, the public key can be created through "ssh-keygen -t RSA -f " together with public key directory name from the terminal.

If the key name has been used, a warning will be prompted, and the user may need to change another name for key pair creation.

6.1.2 Create Instance

Now, we are going through the step of creating new instance/server in the nectar research cloud.

```
Please provide instance name:
Please provide keypair name:
```

Figure 18 Instance creation Information Request

First, we need to provide two information, which will be the instance name we need to create and the keypair name that we created in section 6.1.1 or which is already exist in the nectar research cloud.

As for the project purpose, by default, we set the OS image as Ubuntu 16.04, flavor id as 1, security group name as default, SSH, and TCP as well as availability zone as “Melbourne-np.”

If the instance name has been used, a warning will be prompted, and the user may need to change another name for instance creation. Otherwise, check the keypair name if it already exists in the nectar cloud.

6.1.3 Create Volume

Create volume to be attached to an instance can help us manage the partition and prevent from overloading the storage.

```
Please provide volume size:
Please provide volume name:
```

Figure 19 Volume Creation

Provide volume size in “Gb” format and volume name to create a new volume. If the error was shown, check if there still have any volume size free or if the volume name has been used.

6.1.4 Attach Volume to instance

After creating volume in section 6.1.3, provide the volume name, instance name to be attached with and the device name of the instance that will be attached.

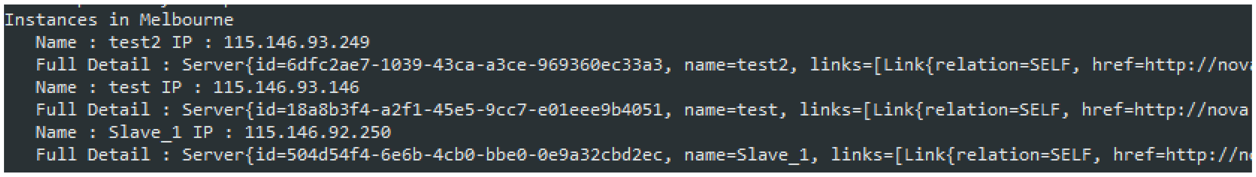
```
Please provide volume name:
Please provide instance name:
```

Figure 20 Volume Attachment

When the creation is completed, it will show which device is attached to the volume, for example, “Volume as attached to /dev/vdc.”

6.1.5 List instance

For this option, simply choose the option, and it will display all the instance which already created.



```
Instances in Melbourne
   Name : test2 IP : 115.146.93.249
   Full Detail : Server{id=6dfc2ae7-1039-43ca-a3ce-969360ec33a3, name=test2, links=[Link{relation=SELF, href=http://nova
   Name : test IP : 115.146.93.146
   Full Detail : Server{id=18a8b3f4-a2f1-45e5-9cc7-e01eee9b4051, name=test, links=[Link{relation=SELF, href=http://nova
   Name : Slave_1 IP : 115.146.92.250
   Full Detail : Server{id=504d54f4-6e6b-4cb0-bbe0-0e9a32cbd2ec, name=Slave_1, links=[Link{relation=SELF, href=http://n
```

Figure 21 Instance List

6.1.6 Auto Creation

Since we have a fix condition for the project, such as four instances with 250Gb volume size in total, we can implement an automated solution for deploy the instance. By choosing this option, the system will create a private key name as “keypair” based on public key provide by a user. Then, create four instances with the key pair created, follow by creating four same size volume and attach it to each instance. In the end, the system will show the device volume attached to as well as each server details.

6.2 Setup Cloud Environment (Ansible)

6.2.1 Install Ansible

Since our local machine is based on MacOS, according to the official manual, we need to use the pip[13] to install Ansible. After successfully installation, we can find that it does not create the default file: /etc/ansible/hosts and /etc/ansible/ansible.cfg. Here we need to create them manually and set the content of ansible.cfg file according to the official guide[14].

Moreover, when we first connect to remote cloud instance, the Nectar will always alarm the host key verification message. Here we can set the following content in ansible.cfg to enable the verification so that the Ansible script will run without errors.

```
[defaults]
host_key_checking = False
```

Figure 22 setting of ansible.cfg

6.2.2 Setup host inventory

Since our solution for the project is based on establishing four cloud instances and making them work as a cluster, thus one instance will be chosen randomly as the main point to manage the cluster setup. In the inventory file of Ansible, these four hosts will be divided into two categories

[13] install Ansible - http://docs.ansible.com/ansible/intro_installation.html#latest-releases-via-pip
[14] ansible.cfg - https://github.com/ansible/ansible/blob/devel/examples/ansible.cfg

like shown below; we need to replace the “Host IP” with the real IP address of each cloud instance.

```
[main]
Host IP

[slave]
Host IP
Host IP
Host IP
```

Figure 23 Setting of host IP

```
[main:vars]
ansible_ssh_user=ubuntu
ansible_ssh_private_key_file=/path/to/key

[slave:vars]
ansible_ssh_user=ubuntu
ansible_ssh_private_key_file=/path/to/key
```

Figure 24 Setting of .key file path

As the local machine connects to each host through ssh, it is essential to assign the path of the .key file. In host inventory, we also need to fulfill the following content by replacing the “/path/to/key” with the real path of the .key file.

6.2.3 Run Ansible script

Open terminal and run the following command: ansible-playbook -i /etc/ansible/hosts /path/to/script/file. Here we need to replace the “/path/to/script/file” with the real path of the Ansible .yml script file. Then the local machine will connect to each remote host and deploy the configuration defined in the script.

6.4 Front-end Web Application

Upload our RESTful server “InfoLoad.war” file to the “web apps” folder in the Tomcat 9 then start up the Tomcat services, after that upload our front web page folder “Result” to the “htdocs” folder in the XAMPP and running the Apache web server. While all the web server is running, we can access the index web page with the URL: “http://130.56.252.7/Result/index.html”.

7. Link

We use github as our version control system and sharing source code as well as executable file among the member.

Github : https://github.com/OrangePurpleThief/CCC-Assignment-2

Youtube : https://www.youtube.com/watch?v=32kqA_G5AW4&feature=youtu.be