COMP90024
Cluster and Cloud Computing

Assignment 2: #TheDeadlySins#2019

Aaron Robins	694098	a.robins3@student.unimelb.edu.au
Yishan Shi	883166	s.yishan@student.unimelb.edu.au
Huiya Chen	894933	h.chen81@student.unimelb.edu.au
Tong He	867488	the2@student.unimelb.edu.au
Yao Wang	869992	y.wang453@student.unimelb.edu.au

Contents Page:

1. Abstract

The following document is a report on Assignment 2 in COMP90024. Firstly, providing the problem statement and what the system wishes to achieve as well as a high level description of the assignment requirements. Following this the system functionalities will be discussed alongside some graphical analysis and reporting of the results. A detailed description of the deployment process and intended usage of the system will provide a simple guide for testing. Then we will go into a discussion on the system architecture, technologies used and overall system performance. This report will also include a risk registry, discussion on the UniMelb Research Cloud and then conclude with some final comments.

2. Problem Statement

Twitter alone has 326 million active users and 500 million tweets per day, combine this with Facebooks two billion active users as well as Instagram, Google, Youtube and many more social media platforms and it becomes fairly obvious that the amount of data generated from Social Media alone is monumental. What makes social media data specifically interesting is the amount and quality of sentiment analysis that can be achieved to understand people and society through data analytics of social media data. For example, each tweet is written by a user, that user has an ID and a location, simply using the tweet and location we can begin to understand weather people are generally happy in certain areas, based on the words they use in their tweets, or if they are particularly angry in a certain location, perhaps because their Footy team has just lost the grand final.

What makes this problem specifically interesting and challenging is dealing with the large amounts of data generated by users, known as Big Data. For a system to be able to get, store and analyse Big Data, technologies used, architecture and use of computing resources must be specifically chosen, designed and deployed. The notion of Big Data does not just involve volume of data but also includes:

1. Velocity: The rate at which new data is important into the stystem
2. Variety: The variability of DB schema, how well is the schema at adapting to changing environments
3. Veracity: The level of trust in the data accuracy

One widly used solution to this issue is using a Cluster and Cloud based architecture. The cloud services utilized for this project is the National eResearch Collaboration Tools and Resources research cloud (NeCTAR) which operates based on the openstack protocol.

The purpose of this assignment is to build a cloud based system which uses the computing resources of multiple VMs and nodes, this system will be used to analyse and explore large sets of social media data as well as validate these findings with data available within the

AURIN platform. This must all be achieved in a scalable manor which allows for elastic scaling. The system will have three main components, twitter harvesting application, CouchDB database and a front end wed application. The following five activities outline the five key steps involved in building this system.

1. Cloud based system: Exploits a multitude of machines, can run across any node to harvest and store tweets
2. Harvesting of social media data
3. Storage of social media data
4. Analysis of social media data
5. Validation against real data from AURIN

The following two sections explore the system functionalities based on the above five key activities as well as a user guide for testing. In section five the way different software components are distributed on computers, and the way in which they interact with each other will be discussed.

Lastly, The entire system needs separate components to integrate seamlessly, which involves different network services, different software and frameworks, and different programming languages. This requires the team to have a varied level of programming and members can be involved in different skill directions. The team has to not only consider the completion of individual functions, but also the compatibility issues between them and the way data is delivered. The team needs to have good communication skills and collaboration skills if the final system is to be delivered, making this assignment just as much about teamwork in software development as well as technical skills such as IT and programming skills.

3. **System Functionalities**

3.1 Overview

Broadly speaking the system has three main functions. Firstly to harvest tweets from twitter, secondly to store tweets from twitter and thirdly to analyse the data stored in the DB and A front-end web visualising these data sets/scenarios. Due to the large amount of data that is being written and read from the DB a NoSQL database, CouchDB is chosen. The main reason for NoSQL databases performing faster in regards to read and write is because they are horizontally scalable. All of the data for this assignment as stored in CouchDB including tweets harvested from Twitter and raw data taken from AURIN.

NeCTAR is an infrastructure which offers computing resources as a service, for the purpose of this system four VM's have been utilised for a range of functions. For automated deployment and configuration of VM's ansible playbook has been used. At a high level, one VM is for CouchDB, one for a web server and two for twitter harvesting. A python program is utilised for the purpose of harvesting tweets and saving them in CouchDB.

3.2 Harvesting Tweets

Before storing or analysing Twitter data, the harvesting application will make use of Twitter API's to collect tweets from Australia. One of the key criteria of this component is that it must be scalable, meaning it can be run across and node of the Cloud to harvest and store tweets. The scalability of this component will be discussed in the user guide section of the report.

Twitter has two public API's that can be used for harvesting tweets, Streaming and Searching. To access these API's a Python program, "name of program", is used as well as the tweepy library for Python. The python program will take advantage of both the Search and Streaming APIs. Also, there is a pre-processing of twitter data so that storage is persistent and consistent. Pre-processing ensures no duplicate entries are added to the DB, JSON documents are cleaned so that only relevant and valuable information is stored and also ……………………………………………… Multiple harvesters can be running at the same time and all storing data in the same CouchDB instance.

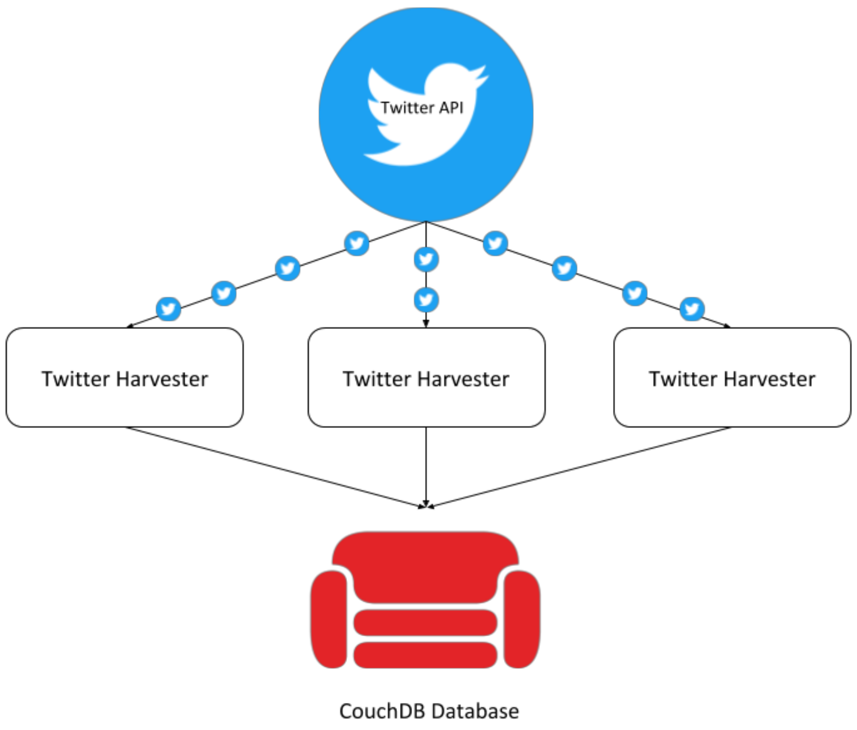


Figure 1: *Architecture: Twitter Harvester*

So, what exactly fits inside a Twitter Harvester? The twitter harvester does some pre-processing before storing files in CouchDB. Visually this looks as follows.

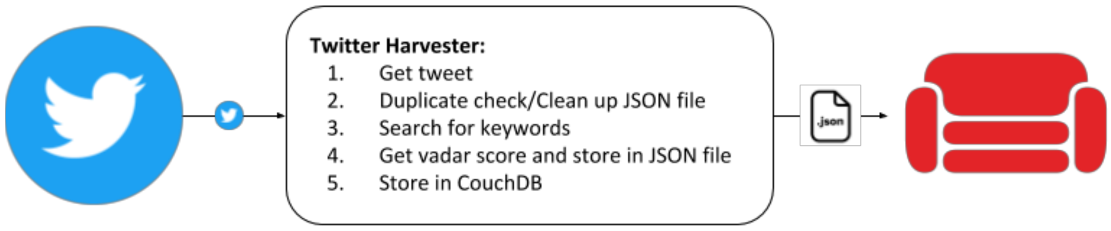

Twitter Harvester:
1. Get tweet
2. Duplicate check/Clean up JSON file
3. Search for keywords
4. Get vadar score and store in JSON file
5. Store in CouchDB
json

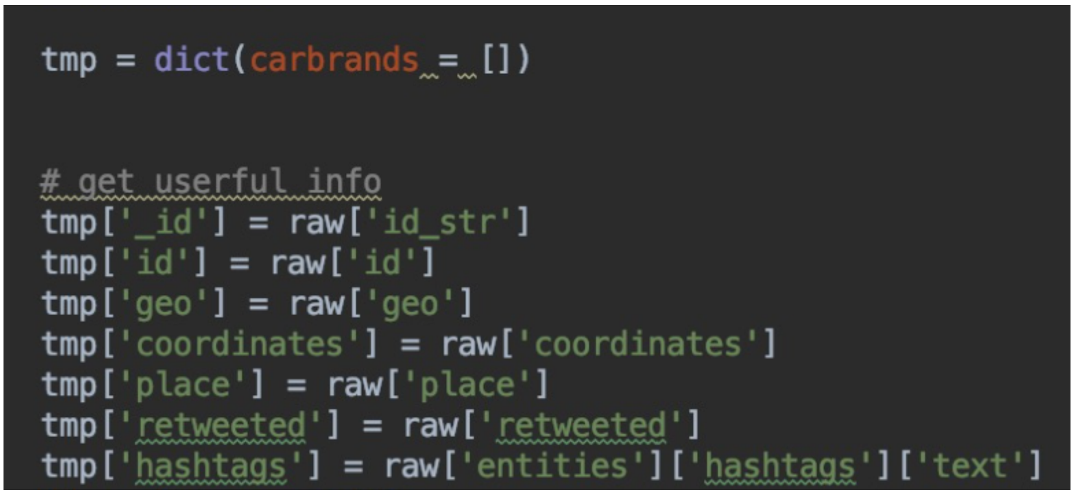
Figure 2: *Processing of Tweets algorithm*

1. Get tweet:

Using Twitter streaming and search API's as well as tweepy, a python library, the python program run.py (https://github.com/Fish-WY/CCCproject2/tree/master/TwitterHarverster) will begin to harvest tweets.

2. Duplicate check/Clean up JSON file:

To check for duplicates we simply use the unique ID provided by Twitter and set it to the primary key of the DB. To clean up the JSON file, as shown in TwitterAPItools.py, create a dictionary and only get the key-value pairs we require from the JSON file and store in the temp dic. After all operations are complete on the dic it is then outputted as a clean JSON file.



```
tmp = dict(carbrands = [])


# get userful info
tmp['_id'] = raw['id_str']
tmp['id'] = raw['id']
tmp['geo'] = raw['geo']
tmp['coordinates'] = raw['coordinates']
tmp['place'] = raw['place']
tmp['retweeted'] = raw['retweeted']
tmp['hashtags'] = raw['entities']['hashtags']['text']
```

3. Search for keywords:

Check each tweet for the following keywords:
Car brand list:Hand-on brands of car in Australia (can buy):
['Abarth', 'Acura', 'Alfa Romeo', 'Alpina', 'Aston Martin','Audi',
'BMW','Bentley', 'Bugatti', 'Buick',
'Chevrolet', 'Chrysler', 'Citroen', 'Cadillac'
'Daihatsu', 'Daewoo', 'Dodge'

```
'Ferrari', 'Ford', 'Foton',
'Holden', 'HSV', 'Honda', 'Hyundai',
'Infiniti', 'Isuzu',
'Jaguar','Jeep',
'Kia',
'Lamborghini', 'Land Rover', 'Lexus',
'Mazda', 'Mercedes-Benz', 'Mitsubishi', 'Maserati', 'Mclaren','MG', 'MINI',
'Nissan',
'Peugeot', 'Porsche',
'Rolls-Royce',
'Subaru','SKODA', 'SsangYong', 'Subaru', 'Suzuki',
'Tesla', 'Toyota'
 'Volkswagen', 'Volvo']
```

Step 4: Get Vadar Score

VadarSentiment is a library in python that allows for sentiment analysis on social media texts. This is an open source library and is widely used and popular because it not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is. The advantages of Vadar include that it does not require training data, only human curated sentiment and importantly for this system it is fast enough to be used online with steaming data. It is also adapt to handling emojis which is a nice touch. A more in depth description of how the scoring is achieved can be found at https://github.com/cjhutto/vaderSentiment

In step 4 each tweet is parsed into the function getVadarScore and that score is stored into the JSON file.

Step 5: Store in CouchDB

3.2.1 Search API

The Search API provided by Twitter comes in several categories, for the purpose of this system the Standard category will be used which restricts our data to just the last seven days, one of the limitations of the system. The second limitation is a maximum of 180 requests per 15 minutes of use and also only a sample of Tweets are made available, not all tweets matching search criteria will be returned, this is because the standard search API is focused on relevance and not completeness (https://developer.twitter.com/en/docs/tweets/search/overview/standard). If completeness is more important than paid versions such as premium or enterprise would be needed. Although only standard API has been used, for all purposes the system is capable of dealing with larger quantities of data.

3.2.2 Streaming API

As well as the search API, Twitter also provides a Streaming API for streaming real time tweets. As a standard user, this is limited by 400 keywords, 5000 user IDs and 25 location boxes. Unlike the search API, streaming API ensures completeness as it maintains a persistent client-server connection and also does not limit the count of Tweets that can be harvested.

For the purpose of this application, streaming API will be used to continually add the most recent tweets to our analysis on top of historical tweets already stored in CouchDB.

3.3 CouchDB

CouchDB provides the main functionality for storing information from both Twitter and Aurin data as well analysis on this information using MapReduce functionality. CouchDB is open source and focuses on establishing a scalable architecture. It is documented oriented NoSQL database which uses JSON format to store data. The main functionality of CouchDB which makes it suitable for use is that it allows for distributed architecture with replication, Map/Reduce views for analysis of data and an HTTP API for reading and writing updates to the database. Other benefits include the ease of setup, web based UI for admin interface, and inbuilt functionality for sharding, clustering and replication. Version 2.3.0 is the latest current version and the one that will be used for this system.

Two important parameter's to take not of are the number of shards, q, and the number of replicas, n. For the purpose of this system the CouchDB defaults will be used which is eight shards and three replicas. A shard is a horizontal partition of the data, once partitioned into shards, n copies of each shard are distributed to the number of nodes in the system. This provides protection against failing nodes in the cluster. With eight shards and three replicas this leads to 24 shard replicas, meaning that on a three node system there will be 8 shard replicas on each node.

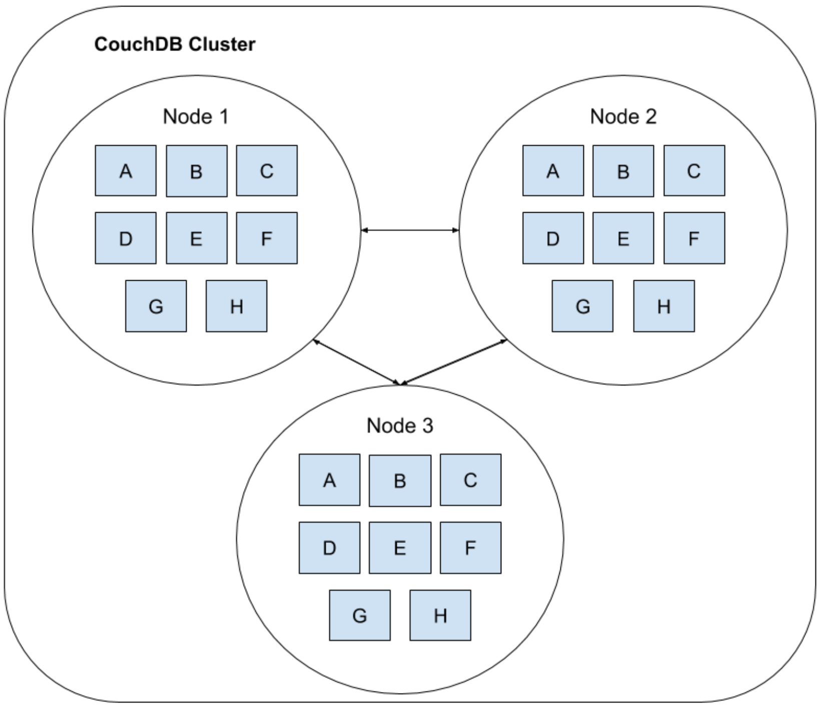


Figure 3: *Shard Replica representation across three nodes with 8 shards and 3 replica*

When deploying the system it is recommended to use a number of nodes which is evenly divisible by the number of shard replicas, in the case of this system that means using either 2, 3, 4 or 6 node cluster would be acceptable. As the number of nodes increases, CouchDB will distribute the shards evenly amongst nodes ensuring that in the event a node fails the complete contents of the database will still be maintained.

CouchDB also provided an HTTP API that can be used to interface between CouchDB and other components in the system which need access to the documents stored in CouchDB. For the purpose of this system this API will be used for the following functions:

1. Harvester will use API to store data in CouchDB
2. Aurin Data will be manually uploaded to CouchDB
3. Website for visualisation will use API to pull data from CouchDB to be visualised.

The last main functionality provided by CouchDB is the ability to perform Map/Reduce functions on the stored data. Map/Reduce functions work by first running a Map function which produces a list of key-value pairs and then a Reduce function that reduces the list to a single value. This family of functions are well suited for jobs that need to run in parallel. The following is a list of MapReduce functions used.

Function	Output	Description

Pros and Cons of CouchDB

Key challenge is deploying nodes

3.4 WebServer

The final component of the system is the Web Server which is used for hosting a website for the purpose of visualising data and results. Tornado, a python based web server framework has been used to make the website. On top of this, bootstrap framework, Highcharts and google maps API have been used for the purpose of design of the front-end web page, making charts and visualising geo-location based data on a map respectively . The Ajax framework has been utilised so that whenever a user clicks or interacts with the webpage a new HTTP request is sent to the Web server, the web server will then connect to the CouchDB, request the most up to date information which will then be dynamically updated on the web page. A more detailed discussion on the libraries, frameworks and technologies used within each component of the system will be provided in section 5 of the report.

3.4 Scenarios supported

- Graphical results and snapshots of user interfaces

4. User guide for testing

[Will need to be given a in person tutorial]

5. Sentiment Analysis

Story

- screenshot of Australia separated into 88 regions
- If you put the cursor over the region it will show the average weekly income
- If you click on a region (show screenshot) it will show the average vadar score per car brand represented on a bar graph
- If you click on the graph there will also be a line graph with Vadar score vs time of day

5. System Architecture

Architecture design along with technologies chosen have been designed with the following main goals in mind. Firstly and most importantly is elasticity and scalability. The system must be able to elastically scale upward and downward with the amount of resources required. For example, the system must be able to choose a number of harvesters to deploy. Secondly the system must be fault and failure tolerant, meaning that from the perspective of the end user, there should be no effect if one node fails. For example, if a CouchDB node fails there should be no loss of data. Lastly, the system must have scripted deployment capabilities, meaning that all virtual machines can be deployed and configured with a script. Please note that although these are the main goals, this does not imply that other goals such as availability, consistency and overall performance of the system have not been considered.

The designed architecture uses all four allocated virtual machines.

- registry of Frameworks, programming languages, databases and any other technology used. Why, pros and cons and for what purpose
- Visual representation of our system and how it all comes together

6. Challenges/risk registry

ID	Name	Description
1	First time using UniMelb Research Cloud	[Aaron to complete]
2	Communication	[Aaron to complete]
3	Speed of communication with twitter API	
4	Difficult to unify geography granularity between tweets and Aurin data	
5	Deploying CouchDB cluster architecture	

7. UniMelb Research Cloud Review

The NeCTAR Research Cloud utilises the Openstack cloud operating system which is used for controlling, administering and metering large pools of compute, storage and networking resources. This is known as the IaaS – infrastructure as a service cloud model. This cloud offers a range of services, pre configured instance types and networking ability. Whilst this discussion will intend to focus specifically on UniMelb research cloud, many of the pros and cons are a direct result of Cloud computing or the Openstack platform.

Pros	**Cons**
Provide infrastructure as a service. Users can deploy a **variety** of underlying resources such as hardware and networking. You can choose your own system, install software and configuration data. More flexible.	Compared to a large number of research users, available resources are limited, such as network IP may not be enough.
There is a UI that can be used to easily create and control instances and set up network security groups. Although it requires a certain amount of basic knowledge, it simplifies the operation.	
NeCTAR has large storage, instant availability and can easily scale.	
Availability of the standard images to boot from streamlines the deployment process	
Several users can access one specific VM, or work on the same data at the same time.	

8. Relevant links

- YouTube Video
- Source Code

9. Conclusion