An Investigation into Environmental Sustainability
COMP90024 - Team 24

Matthias Bachfischer* Parikshit Diwan †

Liam Simon ‡ Rejoy Benjamin §

Colin McLean ¶

COMP90024 Cluster and Cloud Computing

The University of Melbourne

May 26, 2020

Abstract

In this report, the system and application architecture that was implemented for assignment 2 of the COMP90024 Cluster and Cloud Computing course at the University of Melbourne is presented. The scenario that was chosen for this assignment is related to the topic of environmental sustainability and how it can be linked to user activity in online social media networks such as Twitter.
The report at hand includes an in-depth discussion of each individual system component that is part of the architecture and also outlines the collaboration within the team during the implementation of the system.
The accompanying source code for this assignment is publicly available and can be found under the following Git repository URL:
`https://github.com/Bachfischer/cluster-and-cloud-computing-assignment-2`.

* Student ID: 1133751 - m.bachfischer@student.unimelb.edu.au
† Student ID: 1110497 - pdiwan@student.unimelb.edu.au
‡ Student ID: 1128453 - ljsimon@student.unimelb.edu.au
§ Student ID: 1110935 - rbenjamin@student.unimelb.edu.au
¶ Student ID: 1139518 - cmmclea@student.unimelb.edu.au

Contents

1 Introduction

The topic of environmental sustainability has gained significant momentum within the public discourse over the recent years with various environmental organizations such as Greenpeace or the World Resource Institute which have long been advocating for a more sustainable way of living. Over the last 5+ years, new movements such as "Fridays for Future" protests have emerged and gained a lot of attention in the news and media. Given the increasing importance of online social media networks such as Twitter, it is not surprising that a tremendous amount of the discussions and sharing of information has also occurred in such networks.

The objective of the scenario that was chosen for this assignment was to analyze the Twitter activity of the geographic region of Australia with respect to environmental sustainability. In particular, the focus of the scenario was to analyze whether the amount of online social media activity on Twitter in a particular region correlates with the number of solar panel installations in that region.
By comparing the amount of tweet activity with the number of solar panel installations, the authors of this paper wanted to evaluate whether people really choose to "put their money where their mouth is" when it comes to protecting the environment and living a more sustainable life. Section 2 provides further context about this scenario.

The general system architecture as well as the deployment of the various components on the University of Melbourne Research Cloud is described in section 3. The system architecture is designed for operation in a cloud environment - it is highly scalable and can be fully orchestrated via the Ansible scripting technology (Red Hat Inc., 2020). Docker (Docker Inc., 2020) is used to manage and deploy application components across the four computing nodes that were allocated by the course instructors to the project team. Section 4 describes in detail how those technologies were leveraged for the assignment.

For the study, a large Twitter dataset consisting of tweets from five different cities across Australia (Sydney, Melbourne, Adelaide, Perth and Canberra) was collected. The dataset was extracted from Twitter based on static keyword filtering and exclusively contains tweets that mention the topic of environmental sustainability. Further implementation notes on the architecture of the Twitter data harvesting system can be found in section 5.

The system makes use of the CouchDB database technology (The Apache Software Foundation, 2020) to provide storage and persistence functionality. The CouchDB setup and the most relevant databases are further described in section 6

For comparison of the collected tweet data with statistical data from the Australian Urban Research Infrastructure Network (AURIN), the Clean Energy Regulator (CER) dataset was used. Various processing steps, e.g. conversion of GPS coordinates to postcodes as well as MapReduce-based analytics had to be applied in order to process both the collected tweet data as well as the CER dataset. These processing and analytics steps are further presented in section 7.

Users of the system have the ability to interactively query and extract information from our database using a web-based data visualization tool. The visualization tool consist of a Backend component used for fetching and processing the data from CouchDB database as well a Frontend components which is responsible for visualizing the results in a web-based application. The visualization tool was built using React and Node.JS - section 8 provides further technical details on this implementation.

Section 9 provides a short user guide of the system capabilities, while section 10 outlines how the work effort for the development of the system was split up. It further states the individual contribution of each team member. A conclusion of this paper is presented in section 11.

2 Scenario Overview: Environmental Sustainability

Climate change is an issue that everyone around the world is facing. Over the last decade, it has become more and more popular. People often talk about actions that individuals can take to aid the climate, but it is important for people to follow through and take part in these changes.
The power grid in most of the cities in Australia is predominantly powered by coal and gas, but some Australians are more interested in living a "greener" lifestyle and therefore choose to invest in renewable sources of energy. One of the main aims of renewable energy is to reduce carbon footprint, that is to avoid a tonne of CO2 emissions per unit by replacing it with

a more reliant and environment friendly green energy.

Solar energy is one of the top choices for renewable energy. Within Australia, it is growing in popularity and the rooftop solar panels, which is the data looked at through Aurin, account for most of the solar panel energy generation (Australia Renewable Energy Agency, 2020). These solar panels do not cause air pollution, which means a reduced amount of toxins injected into the air. Solar energy also avoids water pollution as extraction of fossil fuels can affect marine habitats and other aquatic ecosystems.

The emphasis when discussing sustainability can be put on the effects that corporations have on the environment, but individuals can also make a change. Because the installations that are being looked at are small scale, this means that individuals have the power to take matters into their own hands and install these types of energy. There are many organisations who help alleviate the difficulties when attempting to install solar panels, such as Clean Energy Council, who provide a list of recommended companies to work with (Clean Energy Council, 2020).

Social media has impacted human lifestyle in many ways, ranging from diversified information from different individuals to initiating social movements towards a greater cause like global warming. Various online movements and discussion groups have formed since and the main aim of these groups is to generate an awareness among the people to stop using sources of energy that causes pollution and to switch to a more environmental friendly form of energy.
The rise of information through social media and the awareness being brought to the forefront has lead to mass movements across the world and also a change in the human mindset to lead a greener lifestyle.

This project hopes to identify whether Australians are simply talking about working towards a more environmentally friendly future, or taking action. By comparing the amount of tweets that are coming from a particular area, the authors of this paper wanted to study whether it's possible to make any interpretations about the amount of renewable energy that the same area has installed.

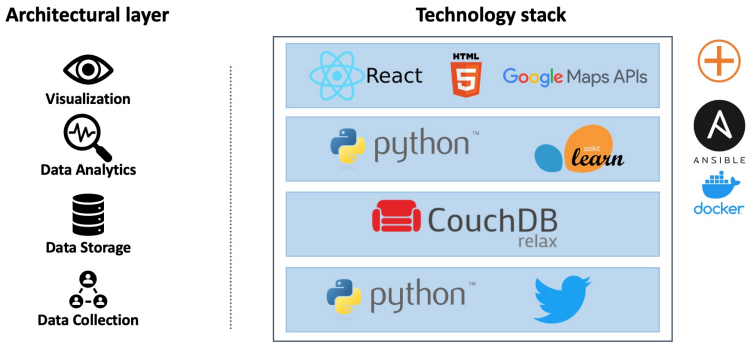


Figure 1: Reference architecture and core technology stack

3 System Architecture

The system architecture developed for this assignment was designed with a focus on scalability and reliability. It comprises a multi-layer architecture consisting of layer responsible for:

- Data Collection
- Data Storage
- Data Analytics
- Data Visualization

The architectural layers, as well as the respective technology stack (technologies and programming frameworks used for the assignment) are shown in figure 1 and are further described throughout this paper.

3.1 General System Design

Core focus of this assignment was to build up a data pipeline which is capable of handling data being streamed into the system in real-time. The data pipeline that was built for this purpose is shown in figure 2. It shows how data is being collected from various twitter harvesters and stored in a database called *twitter_raw_data*. In order to analyze the collected tweet data and compare it with the CER dataset extracted from AURIN, the GPS coordinates of each individual record first had to be converted to postcodes (as the dataset was only available on a postcode level).
To accomplish this, a data processing component was created which extracts

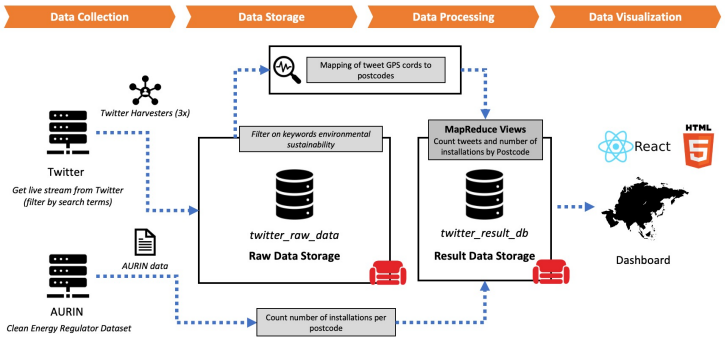


Figure 2: Data Pipeline

tweets from the raw data storage, processes it and stores the resulting information in a new database *twitter_result_db*. This information is subsequently accessed and displayed in the web-based visualization frontend.

3.2 System Deployment

The whole system is deployed on a total of four computing instances running on the University of Melbourne Research Cloud. An overview of the computing nodes as well as their respective services is given in figure 3.
Three of the four computing instances that were deployed on the cloud (*server-01, server-02 and server-03*) have been designated as "Data Nodes". Each Data Node runs an instance of the Twitter Harvester and the CouchDB database in a Docker container environment, thereby providing the opportunity for future scaling in case it is required.
Server-04 has been designated as the "Processing Node" which takes care of processing and analyzing the data. It also hosts the components that are required for the running the web-based visualization frontend. All components again are being run a Docker container environment.

4 Cloud Infrastructure

The architecture described in this report makes use of the University of Melbourne Research Cloud infrastructure that was provided the course instructor. In addition to that, various cloud management and orchestration tools such as Ansible and Docker were used to facilitate the setup and deployment of the system architecture described in this assignment.

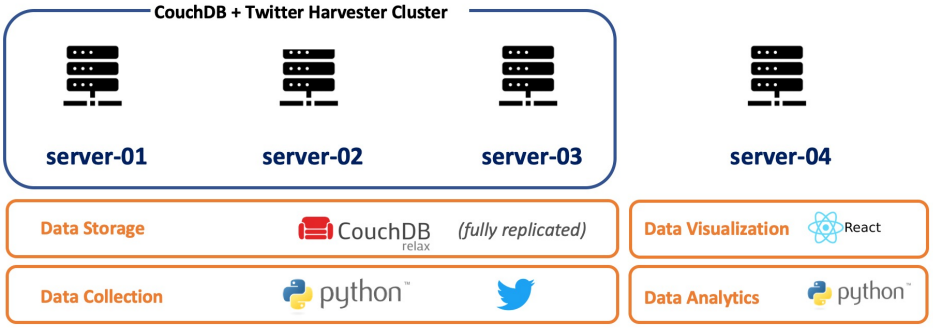


Figure 3: Reference architecture and core technology stack

4.1 Unimelb Research Cloud Infrastructure

The University of Melbourne operates its own cloud computing infrastructure which "comprises almost 20000 virtual cores, available across a range of instance sizes" (The University of Melbourne, 2020). The cloud infrastructure itself is built using the open-source implementation of OpenStack (The OpenStack Foundation, 2020) which allows the usage of existing libraries to access the OpenStack Application Programming Interface (API).
Throughout the project and as a result of our continuous interaction with the University of Melbourne Research Cloud, various benefits and shortcomings were identified and are further listed below:

Benefits

- *Cross-cloud portability:* The deployment scripts and source code for this assignment are interoperable with any hosting provider that runs the OpenStack software, thereby limiting lock-in effects and allowing for future migration in case necessary
- *Usability:* The University of Melbourne Research Cloud provides a visually-appealing frontend interface via the OpenStack Horizon service as well as an API to allow programmatic provisioning of new resources

Shortcomings

- *Limitations in available services:* Commercial public cloud offerings such as Amazon AWS or Microsoft Azure provide a vast amount of services which can be used to speed-up application development (e.g.

DevOps integration etc.). In comparison to that, the OpenStack technology appears to be less mature.

- *Limitations in available resources:* The resource allocation for the project did not allow the provisioning of computing instances with public (i.e. externally-accessible) IP addresses. The deployed infrastructure is therefore only available from within the University network, hence limiting its potential user base

4.2 Ansible Cloud Orchestration

To enable dynamic scaling of the computing resources within the cloud environment, the Ansible technology stack was used for orchestration of all components within the system. Ansible enables its users to "scale automation, manage complex deployments and speed productivity with enterprise automation platform" (Red Hat Inc., 2020) and is therefore highly suited for this kind of tasks.
The Ansible playbook that was created for this assignment enables the deployment and setup of the whole system from scratch (a process which is often referred to as "bootstrapping"). Overall, a total of twelve tasks are part of the playbook. The full overview of the Ansible playbook tasks that have been created for this assignment is presented in figure 4.

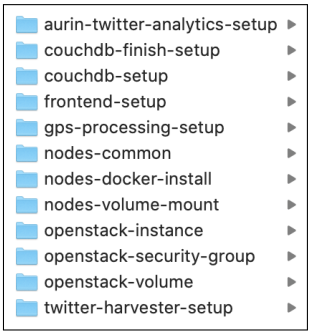


Figure 4: Ansible Playbook Tasks

The tasks are responsible for executing the following steps:

Ansible Bootstrapping Steps

Step 1:
Provisioning of new cloud resources (computing instances, storage volumes, security groups)
Step 2:
Mounting and configuration of storage volumes on computing instances
Step 3:
Installation of software packages (e.g. Python, Docker etc.)
Step 4:
General configuration steps (e.g. configuration of proxy settings for Docker)
Step 5:
Deployment and configuration of CouchDB and twitter harvester source code on DataNodes
Step 6:
Deployment and configuration of data analytics scripts and web-based visualization frontend on Processing Node

Ansible follows the Infrastructure as Code (IaaC) paradignm which refers to the automatic management and deployment of computing infrastructure via machine-readable configuration files. This approach ensures repeatable and predictable deployments, while simultaneously reducing the need for interactive, tedious configuration steps.

4.3 Docker Container Technology

The system implementation leverages a Microservice-based architecture using the Docker container technology for application deployment. Each service runs within its own container and exposes specific functionality to the applications that are interacting with it.
Whenever a new service is deployed, an Ansible task is executed which clones the Git repository[1] containing the application source code for this assignment. Each service has it's own directory within the repository which contains a Dockerfile. The Ansible task subsequently builds the Docker Container as instructed in the Dockerfile, stores the image inthe local repository of the node and executes the Docker image.
This approach has allowed the authors of this paper to frequently deploy new increments of the individual services as more and more features were

[1]Team 24 - Assignment 2 Git Repository: `https://github.com/Bachfischer/cluster-and-cloud-computing-assignment-2`

being implemented. In addition to that, having all the components that are required for the deployment of the system available in isolated Docker containers further facilitated local development work, as part of the system could be quickly set-up for testing and troubleshooting purposes.

5 Data Collection

This section describes the process that is involved in data collection. Section 5.1 describes the setup that was used to harvest data from Twitter, while section 5.2 describes the various ways in which data from AURIN was mined. Section 5.3 further introduces the GeoJSON standard that was leveraged to display information on maps.

5.1 Twitter Data Harvesting

The Twitter harvester has been developed using the Python programming language. For the implementation, the authors have used the Tweepy library (Tweepy, 2020) to access the Twitter API methods. The harvester has been developed as a dynamic and scalable solution with a tight integration into the CouchDB storage environment. This allows to expand the scale of the harvester without needing to taking the system offline.
This features of "zero-downtime update" is achieved by storing the Twitter API credentials , the cities to scrape the tweets from and the search terms we are looking for in 3 databases in the CouchDB setup, allowing us to expand the number of credentials we have, number of cities we are searching in and the the search terms we are looking for in our targeted tweets on the fly.

The overall Harvesting process can be defined as the following:

1. The harvester searches for an available Twitter API credential (i.e. credentials which are not being used by any other harvester instance or whose rate limit has not been reached). If a valid set of credentials is found, it blocks the credentials using a flag so as to not allow any other harvester to access it concurrently. If no credentials are found , then the harvester goes to sleep.

2. Similar to the Twitter API credentials, the harvester looks for cities which are not being used for harvesting .

3. Next, the harvester randomly searches for 20 key search terms from a list of search terms and constructs the search query.

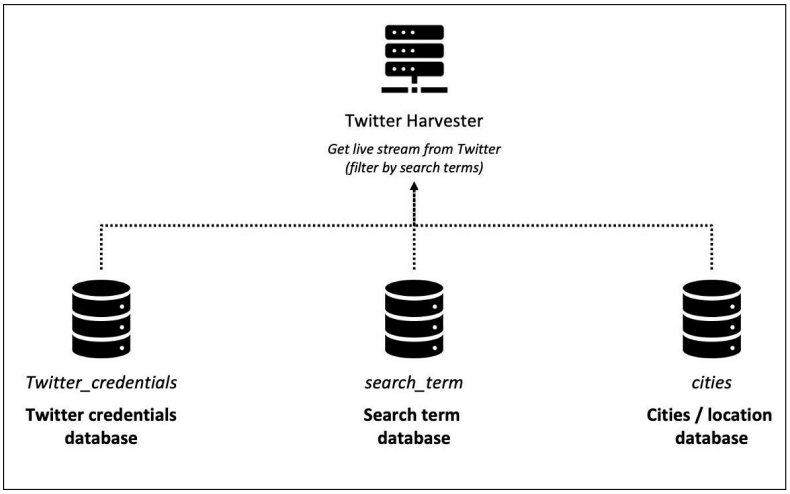


Figure 5: Database support for Twitter Harvester

4. Through randomization, the harvester decides which configuration (the function arguments) of the search API followed by the streaming API.If a satisfactory tweet is found, it is stored in the database with the corresponding tweet ID as the record ID. However, before storing, a check is executed to see whether a record with the same ID already exists in the database or not. This way, it can be ensured that no duplicate data is stored.

5. Whenever a Twitter API error (e.g. in the form of rate limit) is reached or no more data is available, the harvester looks for the next set of available credentials and city and the whole process begins again.

Besides the more generic functionality to harvest tweet data from Twitter at large-scale, the Tweet harvesting component of the system employs a variety of elaborate features to ensure robustness and ease-of-use.
Some of the salient features are:

- Usage of up to 10 twitter API credentials from various accounts to ensure continuous data colelction when the rate limit is reached

- Ability to scan the present stream of tweets as well as to "go back in time" and search for missed out tweets using the `max_id` and `since_id` information in conjunction with the Twitter API.

- Handling of the Twitter API rate limit in two different ways:
    1. Usage of multiple Twitter API credentials so that if the rate limit is reached, another set of credentials is used
    2. Recording of the timestamp of the last usage of a particular Twitter credentials set (so that only credentials which were last used 15 minutes ago are being selected by the Twitter Harvester).

5.2 AURIN Data Mining

This project used the National Collaborative Research Infrastructure Strategy (NCRIS)-enabled Australian Urban Research Infrastructure Network (AURIN) Portal e-Infrastructure to access CER - Small-scale Installations - Solar Water Heater - Solar Panel (POA) on April 29th, 2020 (Sinnott et al., n.d.). This analysis was undertaken using the Spatialise Aggregated Dataset tool, implemented in the NCRIS-enabled Australian Urban Research Infrastructure Network (AURIN) Portal e-Infrastructure on April 29th, 2020. This tool was used to generate shape files for rendering the cities analyzed on the web app.

The data used from AURIN was published by the Clean Energy Regulator (CER) on a yearly basis. The CER records the number of small scale renewable energy installations throughout Australia (Government of the Commonwealth of Australia - Clean Energy Regulator, 2019). These installations are grouped based on postcodes, and the data is then filtered to contain only the postcodes within the Greater Capital City Statistical Areas (GCCSA) for each state, excluding Tasmania. The data that is used spans from the timeframe from 2001 to 2018.

The data set originally included installation numbers on a month-by-month basis, the total number of installations in a year, and the associated postcode, but the focus was on the total number of installations per year. After downloading this data from AURIN, it was then possible to loop through all of the years of the data and compile a new file that had the total number of installations from the entire 17 year period. These were then used as a measurement of the "action taken" by the people in that postcode. The data was stored via CouchDB and access it at any point we desired.

H

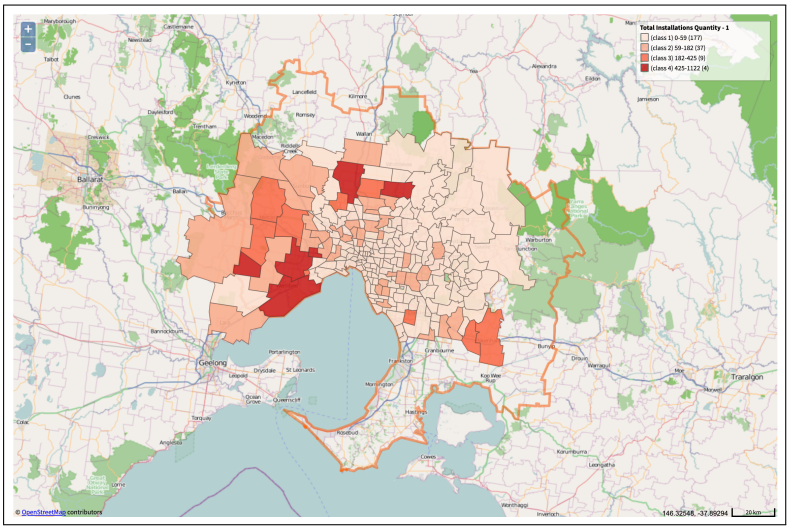


Figure 6: Distribution of installations around Melbourne installations

The other data extracted from Aurin were the shape files that were associated with the installation data. Using the spatialise dataset tool, it was then possible to generate shape files for every postcode. These shape files were then used to extracted the GeoJSONs within them. Once the GeoJSONs were extracted, it was run through the external tool Mapshaper ("mapshaper", 2020) to reduce the granularity of the boundaries. This reduction in boundary detail also led to a reduced file size. After extracting the necessary GeoJSONs, the day was stored in a database with the associated total installations.

After looking at the more granular data grouped by postcodes, we then decided to also look at the data based on the GCCSA as a whole, instead of their respective postcodes. To do this, we retrieved the Aurin data we stored via CouchDB that was the summation of installations in all of the postcodes within GCCSAs spanning 2001-2018. We then looped through all of the different postcodes within these GCCSAs and summed up all of the total installations per year. This resulted in the total number of installations in each GCCSA over the course of the 17 year period.

Through initial data exploration, it was observed that installations were

usually clustered around the outer parts of the GCCSAs, as this is where there were likely new homes being built and new buildings being developed. This can be seen in Figure 5, which is a choropleth map of the Melbourne area.

5.3 GeoJSON Standard

The GeoJSON standard follows the the format specified in RFC 7946 (Butler et al., 2016) stating that "GeoJSON is a geospatial data interchange format based on the JavaScript Object Notation Language (JSON)". The sections of RFC 7946 that were of most interest for this project were that of section 3.1.6 and 3.1.7 which are that of the polygon and multipolygon. These follow certain rules allowing for their standardised use in APIs such as the Maps JavaScript API[2]

For example, a city may comprise of multiple polygons to make up the area that the city encompasses with potentially many different 'holes' for natural features such as lakes where information may be irrelevant. Suburbs, being relatively small in comparison, tend to have just a polygon feature in the GeoJSON but can still contain the 'holes' as mentioned above.

The fact that GeoJSON follows the standards of a standard JSON meant that very little manipulation had to be performed to ensure communication between the components was standardised.

6 Data Storage

The data storage layer for this system was built using the CouchDB database technology. Three nodes (*server-01, server-02 and server-03*) were chosen to form a CouchDB database cluster described in 6.1. Various components of the system rely on CouchDB for storage and persistence - they are shown in section 6.2.

6.1 CouchDB Cluster Setup

CouchDB is a flexible, modern NoSQL database which can be tailored for a variety of use cases. It can furthermore be configured for operation in a clustered environment in order to "improve on the single-node setup with

[2]Maps Javascript API reference:`https://developers.google.com/maps/documentation/javascript/reference`

higher capacity and high-availability" (The Apache Software Foundation, 2020).
For this assignment, the readily-available Docker images provided by IBM[3] were used, as those particular images already include a variety of features suchas as the Clouseau CouchDB text index plugin. Following this, two Ansible tasks *CouchDB-setup* and *CouchDB-finish-setup* were created which are responsible for installing and configuring the CouchDB Docker containers.
The configuration parameters and credentials that are needed for the setup of the CouchDB cluster are stored as variables within the respective Ansible tasks and are passed to the installer during install time.
In order to enable the operation CouchDB in a clustered mode, various prerequisites need to be fulfilled: The security groups used by the computing instances need to be adjusted to allow inter-node communication on Transmission Control Protocol (TCP) port 4369, 5984 and port range 9100 - 9200, a consistent Universally unique identifier (UUID) needs to be set and default credentials for access to the database need to be defined.

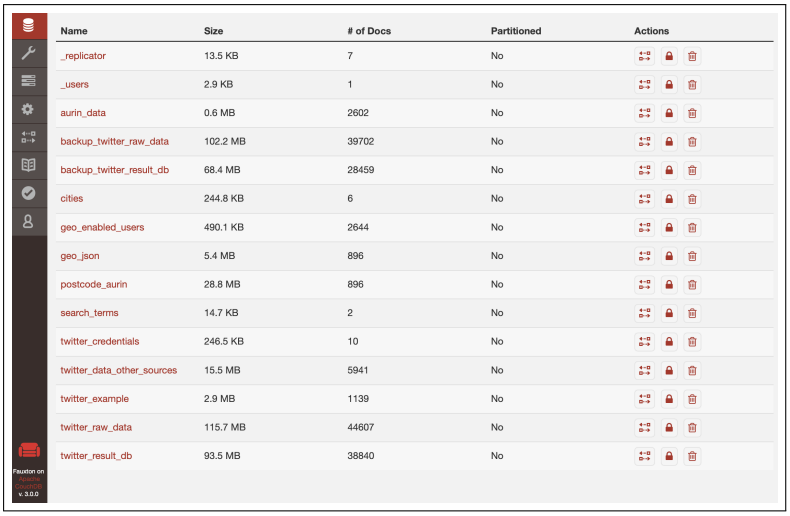
6.2 Database Overview

The CouchDB database is an integral part of various components of the system architecture at hand and is responsible for

1. Storage of raw and processed tweets that were collected from Twitter
2. Storage of configuration parameters for Twitter Harvesters (e.g. Twitter API credentials, Target GPS coordinates of Australia's cities
3. Storage of processed AURIN data for comparison in scenario analysis

An overview of the databases that were used for this assignment is provided in figure 7. Of particular importance is the database *twitter_raw_data* which holds the "raw" tweets that were collected by harvesting the twitter API, as well as the database *twitter_result_db* which holds the tweets after they have been processed and allows aggregation of the data on a per-postcode level.

[3]CouchDB 3.0 Docker Image published by IBM: `https://hub.docker.com/r/ibmcom/CouchDB3`



Name	Size	# of Docs	Partitioned	Actions
_replicator	13.5 KB	7	No	
_users	2.9 KB	1	No	
aurin_data	0.6 MB	2602	No	
backup_twitter_raw_data	102.2 MB	39702	No	
backup_twitter_result_db	68.4 MB	28459	No	
cities	244.8 KB	6	No	
geo_enabled_users	490.1 KB	2644	No	
geo_json	5.4 MB	896	No	
postcode_aurin	28.8 MB	896	No	
search_terms	14.7 KB	2	No	
twitter_credentials	246.5 KB	10	No	
twitter_data_other_sources	15.5 MB	5941	No	
twitter_example	2.9 MB	1139	No	
twitter_raw_data	115.7 MB	44607	No	
twitter_result_db	93.5 MB	38840	No	

Figure 7: Overview of CouchDB databases used as part of this project

7 Data Processing

The system involves two major kinds to data processing. Section 7.1 describes the step which are required to convert GPS coordinates in tweets to postcodes, while section 7.2 further elaborates how the system leveraged CouchDBs built-in MapReduce analytics capabilities. In addition to that, various experiments involving advanced analytics were performed - they are described in section 7.3.

7.1 GPS to Postcode Processing

The conversion of GPS coordinates to postcodes involves the process of taking each record from the "raw" database *twitter_raw_data*, extracting the geo-locations present in these records (i.e. the latitude and longitude from where the Tweet was initially posted) and converting it to an address. From the address we obtain the postcode information and we use that to perform detailed regional analysis of the tweets.

Data processing is achieved with the help of a python script which runs continuously to get all the records from the database which have geo-locations and are not yet converted to postcode. In the case that all the records have been converted the script waits for a minute and then keeps re-checking the

database for newer Tweets. This process continues throughout the lifetime of the project.

For the implementation of the GPS conversion, the Python library Geopy (Geopy, 2020) was used. This library provides a python client for several popular geocoding web-services which allow the conversion the latitude and longitude information to an address from which we obtain the postcode information. This information is subsequently updated with the rest of the data in the record to the "result" database *twitter_result_db* where it is stored eventually for further analysis with the AURIN data.

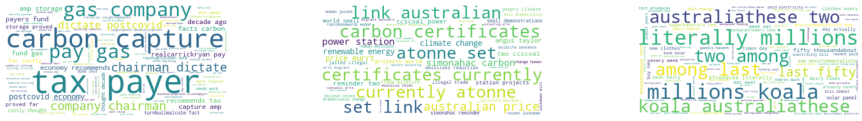
Only those latitudes and longitudes which return a postcode are stored in the Result database while the others are left out. Some of the tweets are "geo-enabled" which means that their latitude and longitude information are highly accurate as their users have attached their current location to their tweets. The tweets that are "geo-enabled" are differentiated from the tweets that are only available on a per-city level in the frontend visualization.

7.2 MapReduce Analytics

MapReduce was used to count the amount of tweets that occurred at each postcode. Since many tweets lacked precise GPS information, the postcodes were further aggregated into self-defined, "city-wide postcode" with a dummy value of the city's central business district postcode + 10,000 (i.e. the postcode that reflects all tweets within Melbourne area is 3000 + 10,000 = 13,000) .
This approach allowed the system to count GPS-enabled data separately to the data which was only available on a per-city level and did not have location data. This not only made it easier both to count but also to display on the website.
MapReduce was further implemented using the view function given by CouchDB where '_count' was the reduce used.

7.3 Advanced Data Analytics

Besides the processing of tweet information in real-time within the system, a variety of additional, additional advanced analytics techniques were applied in order to generate a better understanding of the contents of the data set and verify initial hypotheses.
Amongst others, two of the approaches that were used for his kind of ad-



(a) Sydney (b) Melbourne (c) Perth

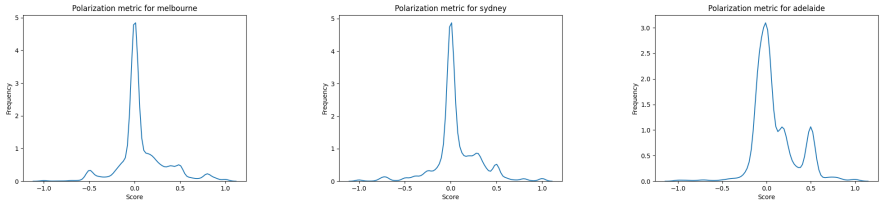
Figure 8: Word clouds for cities in Australia

vanced data analytics are:

1. Generation of word clouds (i.e. most commonly mentioned words)
2. Calculation of polarization and subjectivity scores on a per-city level

By generating word clouds, the authors of this paper wanted to create a visually-appealing representation of the most commonly mentioned words in tweets. An example of the word clouds that were generated for Sydney, Melbourne and Perth is shown in figure 8.

Since environmental sustainability is generally regarded as a highly controversial topic, the authors of this paper were interested in better understanding the sentiment of Australians when tweeting about that topic. Figure 9 shows polarization scores for the cities of Melbourne, Sydney and Adelaide. The polarization scores are calculated using a Python library called TextBlob which returns a float value between -1 and 1 where -1 indicates highly negative statement while 1 indicates positive statement .



(a) Melbourne (b) Sydney (c) Adelaide

Figure 9: Polarization scores for cities in Australia

8 Data Visualization

This section describes the Frontend data visualization component that was implemented for this assignment. It consists of a backend data caching component described in section 8.1 as well as a frontend data visualization component presented in section 8.2.

8.1 Backend Data Interface

The back-end consists of a Node.js application that facilitates the retrieval of information from CouchDB. It provides a Representational State Transfer (REST) API the enables the frontend to easily retrieve specific sets of data from the database (e.g. tweet statistics per city).

The relevant information to be displayed is found in separate databases. The database `geo_json` contains pre-processed AURIN data which is separated into suburbs stored as GeoJSONs, while the database `twitter_result_db` contains the accumulated and processed tweets. These needed to be joined so that the information could be easily gathered from the back-end and passed on to be visualised by the front end. The script uses MapReduce functionality (as discussed above) to gather the count of the tweets from each postcode and join them with the appropriate GeoJSON where it was stored in the properties section where it maintains the standards for the use of a GeoJSON.
From the tweet count and the AURIN data which can be found in the properties, the ratio between the installations of solar panels and the tweet count of a suburb are calculated and stored in the GeoJSON. This reduces the need to do advanced processing on the frontend. This final GeoJSON is subsequently stored in the database `postcode_aurin`' within CouchDB where it could be retrieved.

The back-end does not perform any processing on the data. It is rather used as a way to retrieve data from the database when queried by the front-end. A JavaScript package called nano (Nano, 2020) facilitates the access of CouchDB with the use of views such as the retrieval of all suburbs or cities. This is done through the provided CouchDB REST API.

When attempting to retrieve data from the database, the backend will query a URL that contains the appropriate view which was created on CouchDB. This view provides the frontend with the data is has requested. A similar

process occurs during communication betweenthe frontend and the backend. It queries a URL that resolves to a certain route on the backend where the process of receiving information from the database is implemented..

8.2 Frontend Data Visualization

The front end is built in JavaScript using the web framework React. This technology stack was chosen as it provided easy access to libraries that facilitated the use of google maps as a means to present the data. This data has been displayed using the shape files acquired from AURIN.

The front end queries the backend dependent on what the page attempted to display. The home page shows the distinction between the different capital cities of each state and therefore only requires the retrieval of the GeoJSON on a per-city level. Google maps is then rendered once the file has been retrieved with the focus being initially on the main areas surveyed on a city wide level, Melbourne, Canberra, Sydney and Adelaide.

Information on specific, geo-enabled tweet locations was low and so the use of suburb information has been relegated to a secondary page where individual cities are rendered on google maps. This also allows for better comparison between the local ratios of each suburb against one another.

Polygons are the main method for displaying the shapes on the Google Maps API. The GeoJSON data provides this in the form of polygons as well as multi-polygons which were handled separately but rendered in the same way. As these polygons had their sizes reduced the act of retrieving and displaying them was greatly reduced form the original files.

A white to red scale was used to represent the ratio of each suburb. The more red the suburb was, the higher the ratio. These ratios are shifted from their original range to a distribution between 0 and 255 allowing the conversion to hexadecimal and therefore an RGB colour grade scaling form white to red.

A table underneath describes the total tweets and installations per city or postcode, if you are on the suburbs page, where it shows the ratio received by the front end of each area that has been seen.

9 User Guide

The source code for this assignment is available online under the following repository URL:

Source Code Repository:
`https://github.com/Bachfischer/cluster-and-cloud-computing-assignment-2`.

Demonstration of Web App:
`http://172.26.130.40`.

Video Demonstration:
`https://youtu.be/7oCPjouVqUs`.

It contains all relevant application code as well as the Ansible playbook that is required to deploy and setup the system from scratch.

9.1 System Deployment

To deploy (i.e. bootstrap) the system from scratch, first make sure that you have Ansible installed and available on your machine. Next, clone the source code repository and navigate to the *deployment* directory within the repository.
Execute the *run_bootstrap_infrastructure.sh* script in a Linux shell. The script will ask you for your OpenStack API password, and after entering the password, the Ansible playbook which is defined in *bootstrap_infrastructure.yaml* will be executed and deployed.

9.2 Usage of visualization frontend

Shown below is the main page that a user is greeted with when entering the site. The ratio is displayed at the bottom of the map indicating which colours show the highest and lowest of the cities displayed on the map (figure 10).

The user can then click the suburbs button up the top left where it will take them to the secondary page (figure 11). This shows the user user the ratio of installations per tweet of each suburb in a particular city. The starting city is Melbourne. From there the user can click on one of the 4 other cities that they may wish to check up on.

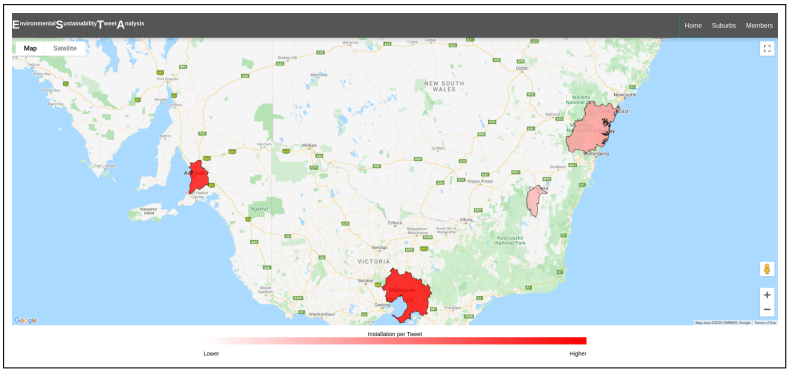


Figure 10: Home page, displaying a ratio comparison between cities

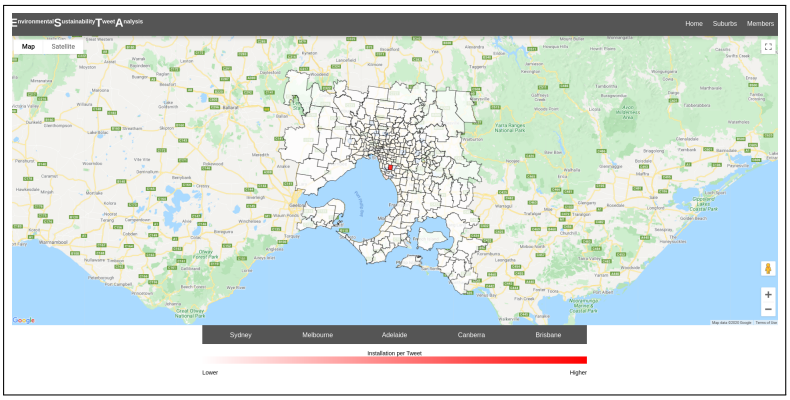


Figure 11: Suburb Page, displaying a ratio comparison between city suburbs

An example of this is the user clicking Adelaide below the map where they will be taken to the vicinity of Adelaide as seen in figure 12.

Under each of the maps the user can find a table displaying the relevant information for the city or cities that they are viewing. For example when visiting the map on the front page then the table displayed is that of all the cities and when visiting a specific city on the suburb page the table that will be displayed is that of the suburbs in the city that have at least one tweet (figure 13).

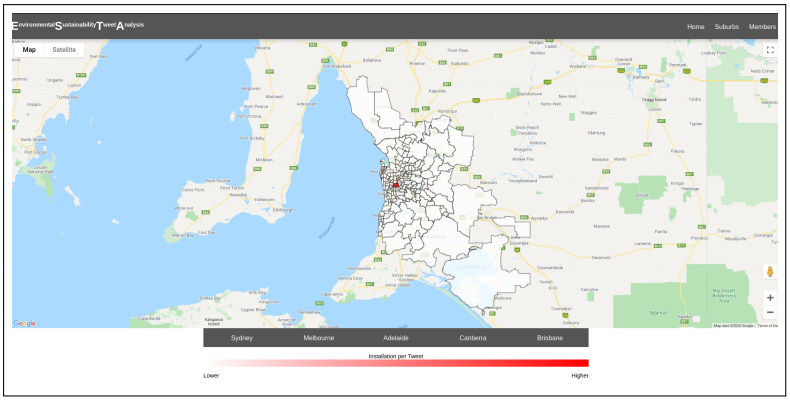


Figure 12: Changing cities will display that cities ratio comparison

Area	Installations	Tweets	Ratio
3000	25	5	5.00
3006	7	1	7.00
3165	1062	1	1062.00
3182	99	1	99.00
3193	329	3	109.67

Figure 13: Table of information on suburbs with a tweet count

10 Team Collaboration

The assignment was delivered by working and collaborating as a highly-integrated, cross-functional team. Each of the team members was tasked with a separate set of responsibilities which are further stated in table 1. The core programming language for this assignment was Python, as all members of the team were familiar with that programming language.
Throughout the implementation period, a variety of digital tools such as Zoom[4] (virtual meetings) or Trello[5] (task tracking) were leveraged.

[4]Zoom Video Conference Software: `https://zoom.us`

[5]Trello Project Management Tool `https://trello.com/b/mjCUm91f/cluster-and-cloud-computing`

Team member	Responsibilities
Liam Simon	- Implementation of web-based visualization frontend - Implementation of middleware and connection to backend - MapReduce of Tweet information and attachment to AURIN data through scripting
Colin McLean	- Downloading data from Aurin - Feature engineering with AURIN data - Assistance with Twitter harvesters and analzying tweet data
Parikshit Diwan	- Implementation of highly-scalable Twitter harvester setup - Analyzing Twitter data to generate insights among cities
Matthias Bachfischer	- Deployment of cloud infrastructure - Setup of CouchDB - Orchestration of applications
Rejoy Benjamin	- Generic data processing scripts - Conversion of GPS coordinates to postcodes

Table 1: Overview of team responsibilities

11 Conclusion

The observed results were expected for some cities and more surprising for others. The measure of "taking actions" to rhetoric was taken by the ratio of installations per tweet. The results can be observed in figure 14. The highest ratio of installations to tweets is in the greater Adelaide area, followed by Sydney, Melbourne, and lastly Canberra. The top two Greater Capital City Statistical Areas are both over ten, while the bottom two Greater Capital City Statistical Areas are five and under.

Population size and the size of the Greater Capital City Statistical Areass are important to take into consideration when looking at the results. Melbourne and Sydney both have populations hovering around 5 million, while Adelaide has 1.3 million, and Canberra has 300,000 people. Sydney has the largest area in terms of square kilometers, but is still comparable to Melbourne. Adelaide and Canberra have a considerably smaller area in terms of square kilometers, so one would expect to see less installations and less tweets in these two areas.

The inner city suburbs of Melbourne have had elected officials who are a part of the Green Party. Therefore, it makes sense that they also have the highest number of installations. However, Melbourne also has the highest amount of tweets relating to climate change. Because their installations were so much higher than the other areas, their ratio was still the top. This shows that people in Melbourne care about creating a sustainable future, and they

Area	Installations	Tweets	Ratio
Sydney	82126	17169	4.78
Canberra	11065	3304	3.35
Melbourne	173245	14289	12.12
Adelaide	30421	2702	11.26

Figure 14: Ratio of tweet activitiy and solar panel installations

are also acting on their words. Sydney has the second most installations in the analyzed areas. Similarly to Melbourne, they also have a high number of tweets. They have a high number of installations due to the high amount of squared kilometers in the Greater Capital City Statistical Areas, and a high number of tweets due to the large population size. Their low ratio shows that although people in Sydney are talking about climate change and they care about these issues, they are not necessarily taking action to create a more sustainable future.

Adelaide has the second highest ratio of installations per tweet. This means that there are many people acting on their words. Also note that Adelaide has the lowest number of people tweeting about issues related to climate change. An explanation for these results could be the fact that Adelaide has the oldest median age of the four areas that were analyzed (Australian Bureau of Statistics, 2016). It is expected that those who are older also tweet less. In addition to this, Adelaide has a smaller population size than Sydney and Melbourne.

Canberra has a reputation of being a "green" city, and the average income is high due to the bureaucratic nature of the capital city. There are a plethora of well paying jobs, so those who live in this Greater Capital City Statistical Areas have the means to install solar panels if they so desire. However, we can see that Canberra has the lowest ratio of the four analyzed cities. It is worth noting that the Canberra area is also the smallest size in terms of square kilometers, so there is less space to be installing these solar panels.

References

Australia Renewable Energy Agency. (2020). *Solar Energy - Australian Renewable Energy Agency*. Retrieved May 23, 2020, from https://arena.gov.au/renewable-energy/solar/

Australian Bureau of Statistics. (2016). *Australia growing greyer*. Retrieved May 25, 2020, from https://www.abs.gov.au/AUSSTATS/abs@.nsf/mediareleasesbytitle/DF79EBB9029669E8CA2581870037A4D8?OpenDocument

Butler, H., Inc., H., Daly, M., Cadcorp, & A., D. (2016). *The GeoJSON Format* (RFC No. 7946). RFC Editor. RFC Editor. https://tools.ietf.org/html/rfc7946

Clean Energy Council. (2020). *Solar Energy Facts*. Retrieved May 23, 2020, from https://www.cleanenergycouncil.org.au/resources/technologies/solar-energy

Docker Inc. (2020). *Docker Project Website*. Retrieved May 23, 2020, from https://www.docker.com

Geopy. (2020). *Geopy Project Website*. Retrieved May 23, 2020, from https://github.com/geopy/geopy

Government of the Commonwealth of Australia - Clean Energy Regulator. (2019). *CER - Small-scale Installations - Solar Water Heater - Solar Panel (POA) 2001-2018*. Retrieved May 23, 2020, from https://aurin.org.au

Mapshaper. (2020). Retrieved May 25, 2020, from https://mapshaper.org/

Nano. (2020). *Offical Apache CouchDB library for Node.js*. Retrieved May 23, 2020, from https://www.npmjs.com/package/nano

Red Hat Inc. (2020). *Ansible Project Website*. Retrieved May 23, 2020, from https://www.ansible.com/

Sinnott, R. O., Bayliss, C., Bromage, A., Galang, G., Grazioli, G., Greenwood, P., Macaulay, A., Morandini, L., Nogoorani, G., Nino-Ruiz, M., Et al. (n.d.). The Australia Urban Research Gateway.

The Apache Software Foundation. (2020). *CouchDB Project Website*. Retrieved May 23, 2020, from https://couchdb.apache.org

The OpenStack Foundation. (2020). *OpenStack Project Website*. Retrieved May 23, 2020, from https://www.openstack.org

The University of Melbourne. (2020). *The University of Melbourne Research Cloud Website*. Retrieved May 23, 2020, from https://docs.cloud.unimelb.edu.au

Tweepy. (2020). *Tweepy Project Website*. Retrieved May 23, 2020, from http://www.tweepy.org/