

WITS UNIVERSITY

SCHOOL OF ELECTRICAL AND INFORMATION
ENGINEERING

ELEN7046 - SOFTWARE TECHNOLOGIES AND TECHNIQUES

Big Data Visualization using Commodity Hardware and Open Source Software

Solution: Twit-Con-Pro

Authors:

Gareth STEPHENSON

Matsobane KHWINANA

Sidwell MOKHEMISA

Dave CLOETE

Kyle TREHAEVEN

Student Number:

778919

779053

1229756

1573016

0602877N

https://github.com/garethstephenson/ELEN7046_Group2_2016

July 2, 2016

DECLARATION OF ORIGINALITY

ELEN7046 Group 2 team members hereby declare the following:

- We are aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- We confirm that ALL the work submitted for assessment for the above course is our own unaided work except where we have explicitly indicated otherwise.
- We have followed the required conventions in referencing the thoughts and ideas of others.
- We understand that the University of the Witwatersrand may take disciplinary action against each one of us if there is a belief that this is not our own unaided work or that we have failed to acknowledge the source of the ideas or words in our writing.

| Member Name | Primary Task | Time Spent | Discretionary | Member Signature |
|--------------------|-----------------------------|------------|---------------|------------------|
| Sidwell Mokhemisa | Architecture | | ..% | |
| Kyle Trehaeven | Development - Streaming | | ..% | |
| Matsobane Khwinana | Development - History | | ..% | |
| Gareth Stephenson | Development - MapReduce | | ..% | |
| Dave Cloete | Development - Visualization | | ..% | |

ABSTRACT

The project aims to provide a low cost solution to big data processing problems, enabling a commercially viable option to individuals, small businesses and academia using commodity hardware. This report explores the use of Open Source technologies Apache Spark and Scala as well as the low cost and scalable Raspberry Pi's. After building a prototype system to source, process and visualize data from twitter, it was concluded that commodity hardware is a viable small-scale solution to working with Big Data.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 4 |
| 1.1 | Problem Statement | 4 |
| 1.2 | Solution Summary | 4 |
| 1.3 | Approach | 4 |
| 2 | Literature Review | 5 |
| 2.1 | Data Sourcing | 5 |
| 2.2 | Big Data Processing Algorithm | 5 |
| 3 | Lifecycle Methodology | 6 |
| 4 | Assumptions and Constraints | 7 |
| 4.1 | Pros and Cons determinations | 8 |
| 5 | Design Decisions | 8 |
| 6 | Success Criteria | 9 |
| 7 | Solution Design | 9 |
| 7.1 | High Level Design: Component Architecture | 9 |
| 7.2 | Detailed Designs | 11 |
| 7.2.1 | Data Acquisition (Batch) | 11 |
| 7.2.2 | Data Acquisition (Streaming) | 12 |
| 7.2.3 | Data Processing | 13 |
| 7.2.4 | Data Visualization | 13 |
| 7.3 | Operational Model: Infrastructure Design | 14 |
| 7.4 | Possible Extensions | 15 |
| 8 | Conclusion | 16 |
| 9 | Appendices | 17 |

1 Introduction

This report presents the work done by Group 2 in response to the project brief for ELEN7046: Software Technologies and Techniques.

The report will broadly focus on the following topics:

- The Methodology followed to execute the project;
- The Architecture of the solution developed for the project; and
- The different technologies used to deliver the solution.

1.1 Problem Statement

There is an abundance of big data available to individuals and companies with very limited capacity to refine this data into meaningful information, particularly for small scale endeavours. Big Data processing is often locked behind high cost barriers to entry, and individuals, start ups and academics may find it difficult to be involved in Big Data processing.

1.2 Solution Summary

Commodity hardware is available to provide a means by which the barrier to entry for Big Data projects can be overcome. Simple, low cost components can be leveraged to address each of the parts of a big data processing solution, whether it be data sourcing, transformation, or visualization; and can be scaled according to needs or as required.

1.3 Approach

As the team was not co-located, challenges around communication and general collaboration was raised before the first team meeting. The team made use of the following, online collaboration tools to meet their needs as well as drive the project delivery:

- **Trello:** To manage collaborative tasks and to provide an understanding of the scope of work.

- **WhatsApp:** To initiate any discussion that required an immediate response.
- **Slack:** For knowledge sharing and lengthy, more general discussions.
- **GitHub:** Source Control and Content Management.
- **Google Hangouts:** For collaborative, interactive knowledge sharing .
- **Contact Sessions:** To ensure frequent, face-to-face communication and ensuring the project moved forward, the team decided to meet every Sunday for the duration of the project.

2 Literature Review

2.1 Data Sourcing

This project was intended to deliver a system or solution for the visualization of Big Data. For this reason it is important that we start by defining what Big Data is.

According to Maden[1], Big Data can be defined as "data that is too big, too fast, or too hard for existing tools to process."

The above definition further supports our group's decision to focus on Twitter as the source of data for the project. Statistics have shown the following average figures with regards to the amount of data that one can get from Twitter[2]:

- 6000 per second;
- 350 000 per minute;
- 500 million per day; and
- 200 billion in a year.

2.2 Big Data Processing Algorithm

The team sought to have all the processing of the Big Data received from Twitter done on multiple nodes following the principles of MapReduce.

Apache Spark was used for this project together with MapReduce which in the main delivers the MapReduce functionality based on the algorithm that is broken down into the Mapper class and the Reducer class[3].

3 Lifecycle Methodology

In order for the team to successfully deliver this project, a development methodology based on IBM Rational Unified Process (RUP) was followed albeit tailored to cater for the specific needs of this project.

The diagram below depicts the IBM RUP model:

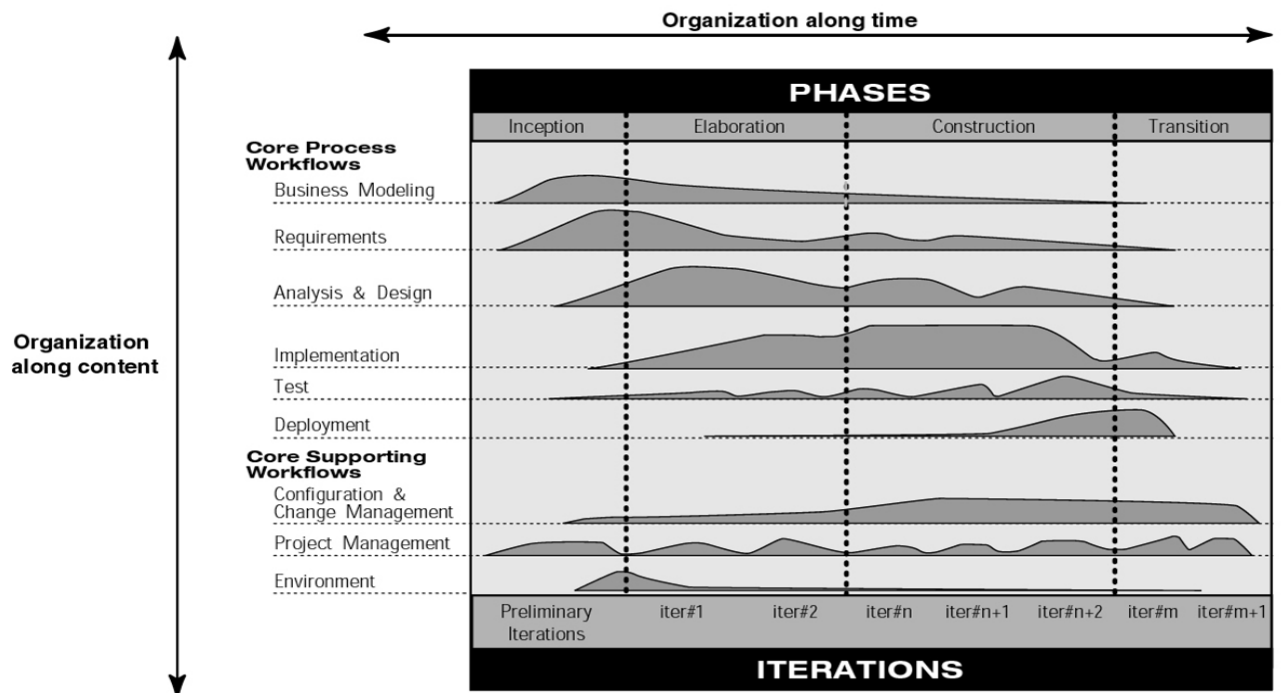


Figure 1: IBM Rational Unified Process (Source: RUP, Best Practices for Software Development Teams)

4 Assumptions and Constraints

Early ideas in the project were based on the Twitter location data; for example, in the inception one of the project the first idea was to visualise the world's reaction to Oscar Pistorious' murder accusation over the world map. Given this idea, the location at which each Twitter postings (tweets) were made was paramount to the success of that idea. Unfortunately due the amount of Twitter data found on this subject, the team decided it was not big enough to qualify as Big Data. The project then decided to rather visualise the currently most topical of subjects, the upcoming United States general elections and also the upcoming South African municipal elections. The idea of using locations where tweets were posted stuck with the project.

However, during the data collection exercise it was discovered that most of the Twitter posts (tweets) had no location data attached to them. It turns out most Twitter users disable their GPS sensor for the Twitter application, presumably for security and privacy reasons. This meant the project would collect data that did not have locations and that would sabotage ideas of visualising data based on locations. After a further analysis on the Twitter data that can be collected, the team came to a realisation that Twitter users have profile location data specified as a free text. As a workaround the user profile location was collected alongside the tweets. Therefore each tweet that did not have a location, the user profile location data was associated with that tweet. The assumption being made by the project was that location of each tweet is the same as that of the user's profile. The project acknowledges that if the user's GPS sensor is turned on, profile location may not necessarily be the same as the location at which the tweet would be posted.

Both Twitter and Google API have usage limits which presented a constraint to the project. Twitter search API allows returns 100 tweets per request and allows only 450 requests per 15 minute window [1]. Only 2500 requests per day are allowed by the standard Google Maps Geocoding API [2].

We assume a rudimentary algorithm for sentiment assessment is "Good Enough", forgoing complex natural language inference.

We assume that any incidental tweets sourced (such as EFF = Electronic Freedom Foundation) are minimal and acceptable.

We assume that Twitter is providing us with either the full extent of our search criteria, or if this is not the case, then the sample we receive is a representational slice.

We assume information that users provide on twitter (name of their location in particular) is correct and relevant, and that those who do not are in the minority. Failing the above, we assume that of those who do provide location based information, their distribution is representative of the norm.

We acknowledge that our demographics are skewed (younger people more likely to use twitter, poorer South Africans less likely to use twitter).

We acknowledge that application software is not flawlessly optimised.

4.1 Pros and Cons determinations

Rudimentary algorithm for determining Twitter statements (tweets) that are against or for a particular candidate was adopted...

5 Design Decisions

The table below details all the key design decisions made in the delivery of the solution:

- **History Data:** History data/ batch interface to Twitter was designed to provide past data subscribed to based on date range.
- **Streaming Data:** This interface provides for all subscribed data in real-time starting from now and going into the future.
- **GoogleMaps:** Tweet location was derived from the co-ordinates found in a tweet where the person tweeting enabled location services on their device to provide current location information. Where privacy settings were not enabled for current location, an interface to GoogleMaps was developed to read the location information of the person tweeting from their Twitter profile and resolve this to actual co-ordinates.

- **Security Directive(s):** It was decided that where security is concerned, only the Twitter and GoogleMaps security requirements be adhered to where integration is concerned. All data acquired from Twitter is data that is already in the public domain, therefore no effort was required to secure the data while in transit, hence the use of only FTP instead of a lot more secure transport mechanism.

6 Success Criteria

- Build a system that uses commodity hardware to solve for big data problems;
- Provide a solution that covers the data sourcing, transformation and presentation of social media data from Twitter relating to the United States and South African elections. The end result must be visualization that provides insight to the sentiment of election candidates on Twitter.

7 Solution Design

7.1 High Level Design: Component Architecture

The high level component model below depicts the key features of the solution delivered for the project.

This design can be categorised into the following three features from a functionality point of view:

- Acquiring data from Twitter based on topics subscription - Political Party sentiment in USA and RSA for 2016. This was achieved by using JEE for history and C# for online data acquisition while persisting all data in MongoDB.
- Data processing was achieved using Apache Spark with MapReduce algorithms developed in Scala. Data in MongoDB was extracted and FTP'd to our clustered infrastructure where all the necessary transformation happened.

- Extracts from the transformation process were sent to the presentation tier application in JSON format where node.js was used in conjunction with D3 chartz to render the visualized big data content on a web browser.

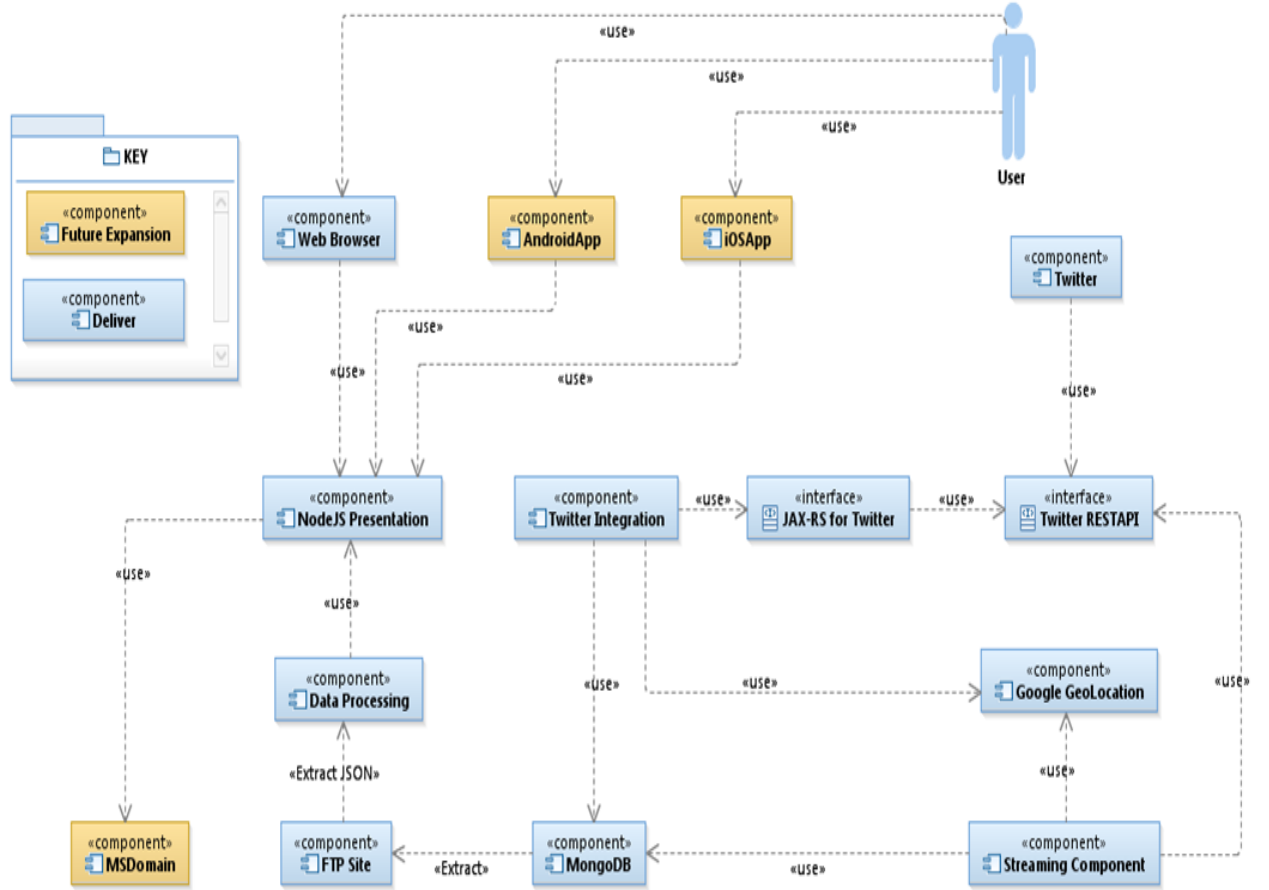


Figure 2: High Level Component Model.

7.2 Detailed Designs

7.2.1 Data Acquisition (Batch)

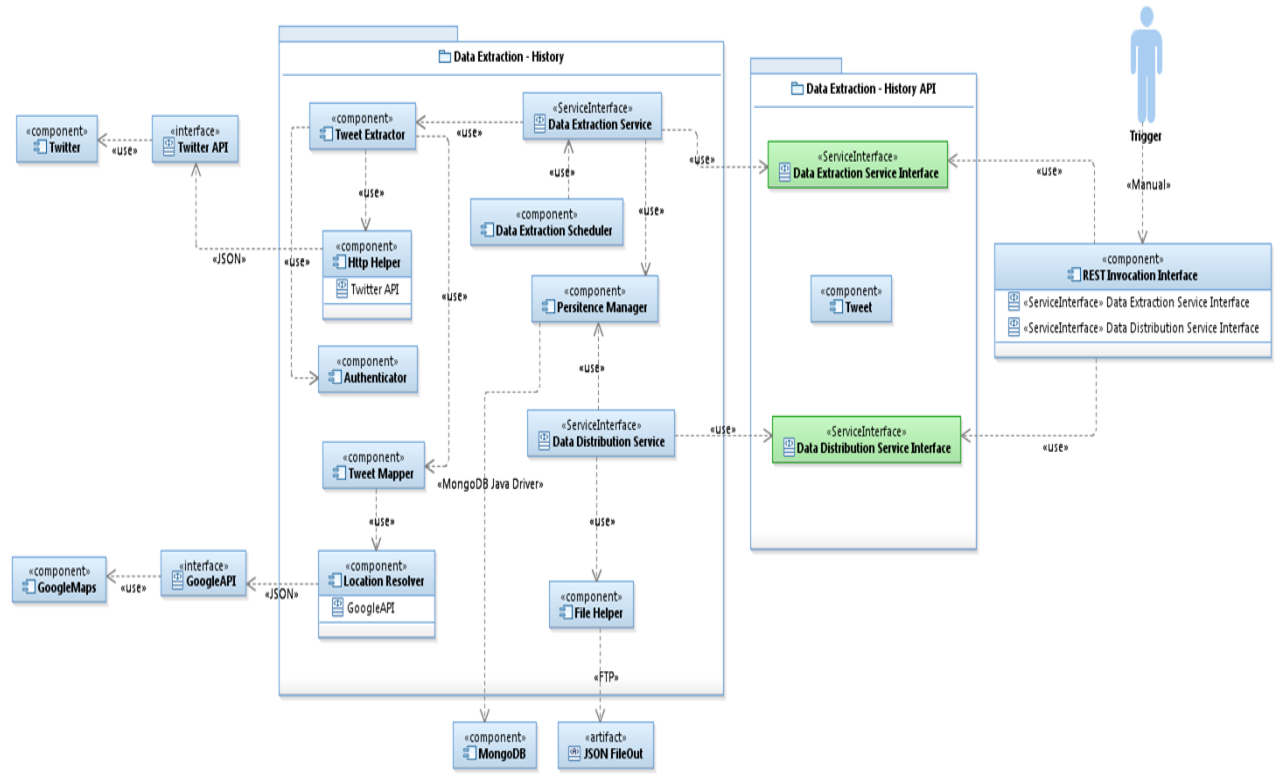


Figure 3: Component Model: Streaming Module

7.2.2 Data Acquisition (Streaming)

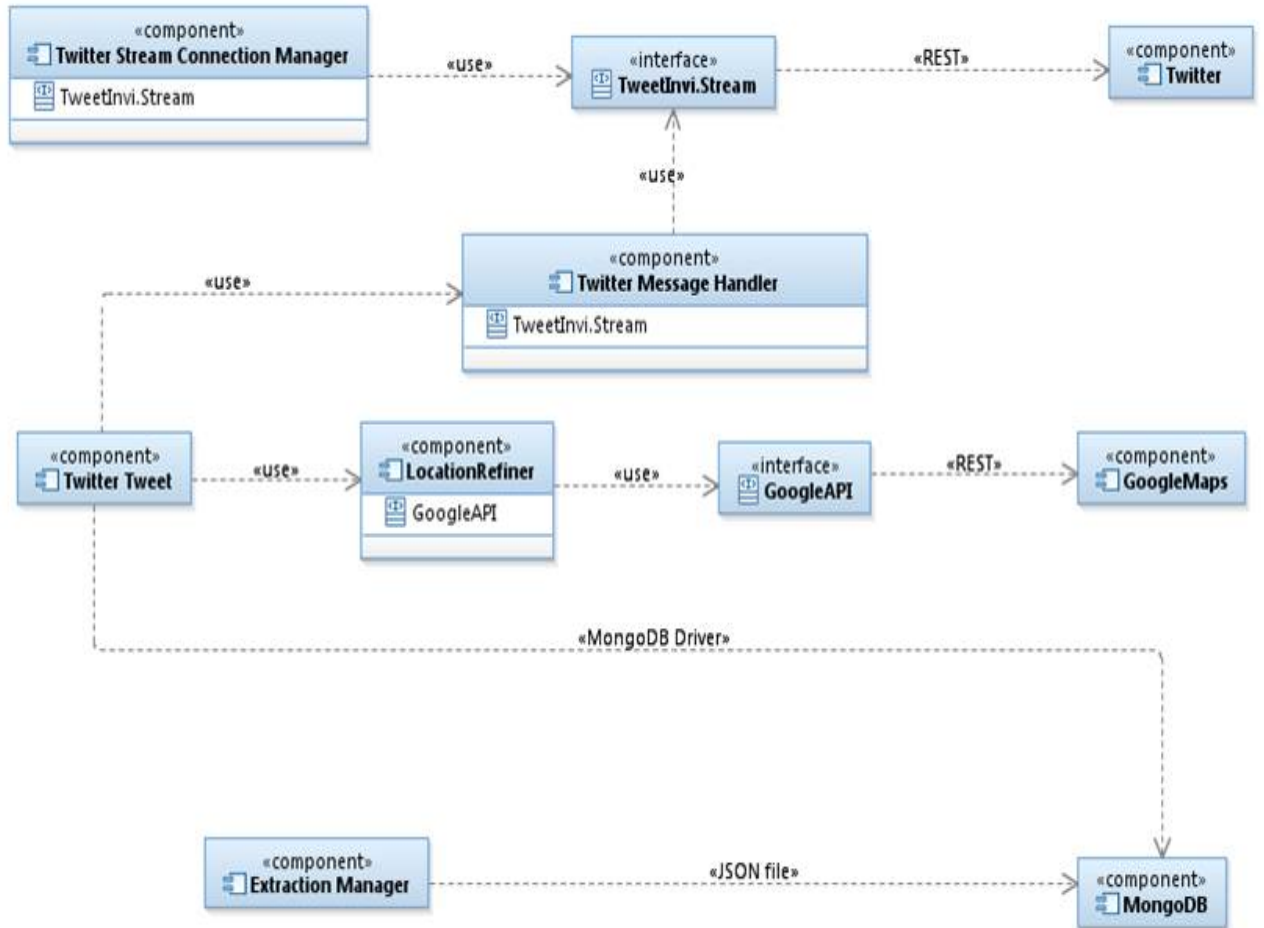


Figure 4: Component Model: Streaming Module

7.2.3 Data Processing

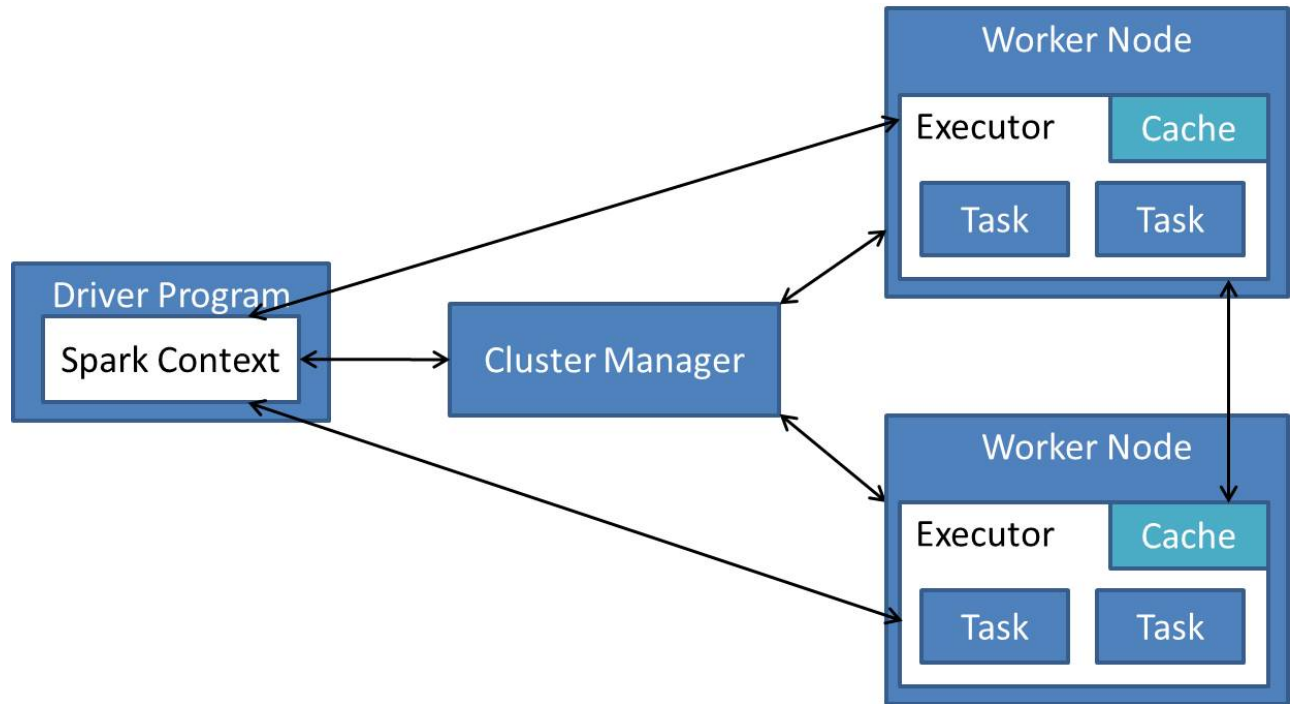


Figure 5: Context Diagram: Data Processing

7.2.4 Data Visualization

The Visual component implements the MVC pattern to separate the View and controlling logic from the Data. This allows for the data components to be developed separately from the Views. Between the Views and Data Models where Controllers are responsible to load the views based on the set configuration and provide the views with the required data model.

The Data Visualisation component was designed to be agnostic of the content and allows for comparison of categories in a topic agnostic of the topic or categories. All charts were designed to be scalable. All graphs will render as the view point changes allowing for a pleasant user experience.

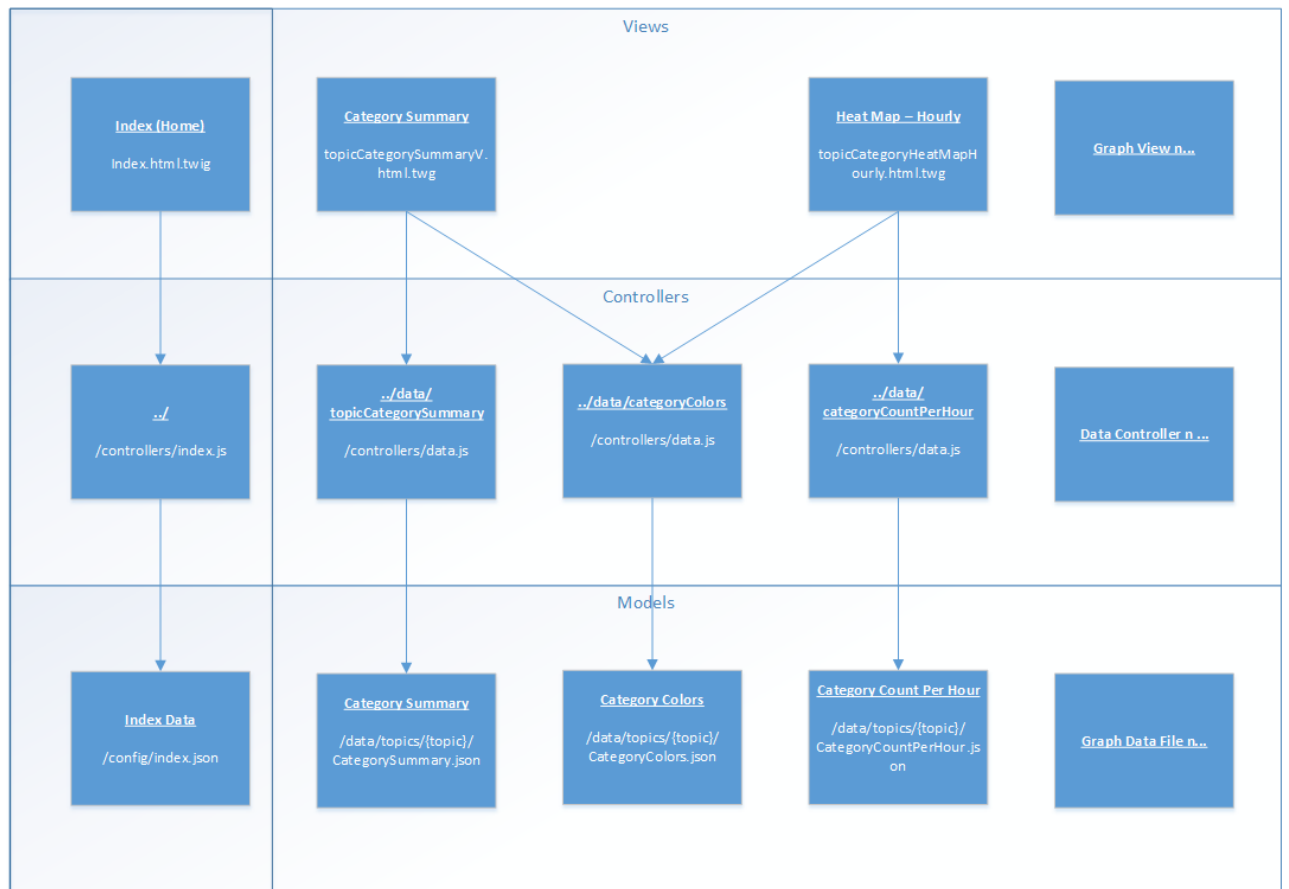


Figure 6: Visualization: MVC Framework

7.3 Operational Model: Infrastructure Design

The Raspberry Pi environment is limited in system resources, specifically RAM. It only has 1GB of RAM which is partitioned for use between the system (accessible by the user) and the GPU.

The Raspberry Pis were configured with the least amount of RAM for the GPU (16MB), leaving the remaining space available for user applications. They were not going to run any sort of GUI or windowing system (like the X Window System), so any larger allocation of RAM for GPU use was unnecessary. The Raspbian Linux operating system (OS) was installed on the Raspberry Pis, as it has been tailored to best utilise the hardware's resources and features. Further configuration was

done on the OS to disable the daemons responsible for controlling the on-board Bluetooth and Wi-Fi modules, thus freeing up even more available RAM. Linux's Network File System (NFS) service was installed and enabled on the Raspberry Pi master node so the slaves in the Spark configuration could operate on a single source of data, with the least amount of memory overhead. When using HDFS, the RAM used by the slave nodes increased from 99MB to 125MB, and from 181MB to 206MB on the master node. In addition to extra RAM use, the time to upload the source data (1GB of plain text files) to HDFS took around 8 minutes, as the data was partitioned across all the data nodes within the cluster.

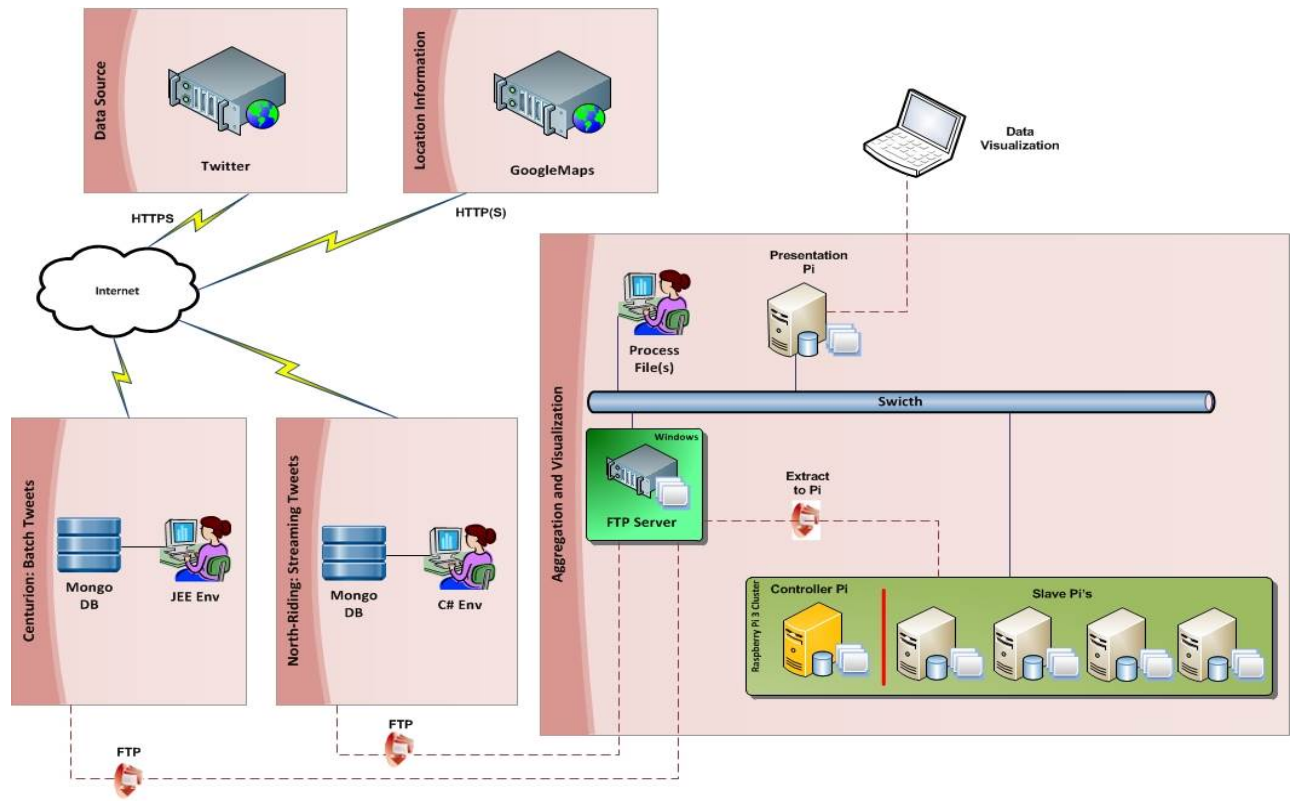


Figure 7: Operational Model: Physical

7.4 Possible Extensions

The project used only Twitter as the Big Data source. According to Statista[6], in April 2016 Facebook ranked number one with 1 590 millions of active users,

while Twitter recorded 320 millions of active users. Other social media networking sites that were ranked included Instagram, WhatsApp, WeChat and LinkedIn among others. Most the social networking service provides an API retrieve social media messages or posts. The project chose Twitter as the data source because Twitter's API seemed to be easier to work with. Due to lack of time only one data source was choosen. One of the possible extension in the project lies on the data collection component. The 2 sub-components can be enhanced to support other social media platforms as the data providers.

Other potential extensions to the solution include developing a smart-phone/tablet App to make the same sort of analytics reports available to a category of user that is always on the move.

The system itself can in future be hooked into LDAP in order to better control user access and security from a central point as well as ease of provision for new users.

8 Conclusion

All this hardware and software is available to anybody interested in Big Data processing.

The hardware is cheap and the software is free.

The learning curve in the beginning can be quite steep but is ultimately very rewarding in terms of what can be achieved with so little financial investment.

References

- [1] . S. Madam. From Databases to Big Data. IEEE Computer Society, 2012.
- [2] . V. Kumar, R. Yuvaraj, C. Anusha. Effective Distribution of Large Scale Datasets Clustering Based on MapReduce. 2016.

9 Appendices