Building Twitter Analytics in the Cloud

Omar El Samad(749907), **Samuel Josei Jenkins** (389975), **MubashirMunawar**(713627), **Xuan Fan** (653226)**, DinniHayyati**(666967)
Computing and Information Systems
Melbourne School of Engineering/ Science
The University of Melbourne

Abstract

The use of social analytics in the cloud has grown to support businesses and government. Therefore, we developed a system with similar purpose which is facilitated by Twitter data to get how people express their feelings over certain products on this social media. We combined NeCTAR, an OpenStack based cloud platform with IaaS (Infrastructure as a Service) model, and CouchDB as the database along with its MapReduce to perform sentiment analysis. This paper provides the details of the tools implemented, system design and architecture and how they are applied to build analytics in the cloud. The system was designed for research purpose and the results show strength and weakness of the NeCTAR cloud platform. Therefore, we provide future improvement and other possible designs to be implemented in building more robust social analytics in the cloud.

Keywords: *Sentiment Analysis, Cloud computing, Social Analytics, Data Mining*

1. Introduction

Paying attention on how people think is a way to get different point of views. Companies, governments, and political campaigns have gained point of views in social media [6] as an evaluation. It is sentiment analysis, a technique in data mining to evaluate textual content as a method in marketing research [5]. A food company, Kraft Foods, implemented this technique as a part of its rapid response systems to handle a controversy of its product in 2003 and cut trans-fat from its products based on the result of the analysis[7 in 6]. As companies and other sectors have widely used this technique, sentiment analysis has been the hottest research areas in computer science and several techniques in sentiment analysis have been developed [8].

We developed a pilot system to perform sentiment analysis over Twitter data by using harvested tweets. We use Twitter because people today express their opinion on almost everything in social media. While some people use Twitter to express their feelings, businesses use these feelings expressed on Twitter to increase revenue. Companies are doing research through social media for reputation monitoring or for marketing [3]. Various researches related to Twitter has been conducted from how people express their feeling to how to do opinion mining in the social media [1, 2, 5]. To help the research, business intelligences have been developed to analyse Twitter data as it is found that this social media can give a real-time of brand sentiment [4].

Despite abundant information on Twitter, performing analysis based on Twitter data may overcome numbers of difficulties. Below are some characteristics of Twitter data which requires thoughtful design in data mining:

1) Each tweet contains a very short message, up to 160 characters. Although it mentions the targeted product, the content may not be valuable and is not a useful response [2].
2) The data grows every day and the harvest tweets can be arbitrarily large [1].
3) A person may tweet the same emotion of the same thing/product/branch over and over again.

Besides Twitter data characteristics, it has to be noted that analysing Twitter data could mean working with big data. Therefore, most systems which perform sentiment analysis over Twitter data which has been reported are cloud based systems.
Cloud computing provides services for their users to do parallel processing and analytics Jobs [19]. This service provides cloud providers' customers a facility to process big data in shorter time. A real example is provided by EC2 (Elastic Compute Cloud), a web service released by Amazon to launch instances of an application under several operating system [15]. EC2 let Washington Post to use 200 EC2 instances (1,407 server hours) to convert thousand pages of documents into a form that can be used in the internet within nine hours after the event occur [20 in 19]. Processing big data faster has been a convincing solution to perform analysis, particularly Twitter sentiment analysis in this project.

As EC2 is commercial, we use NeCTAR in this project instead. Introducing NeCTAR, a single integrated OpenStack based cloud which is funded by Australian Government for research purpose. OpenStack (Nova) is an open source that analogues to EC2 and can be managed in similar way [21]. NeCTAR is chosen by our system as it has similar functions to EC2 and other Amazon web services. Table 1 describes OpenStack technologies NeCTAR uses which some of them similar to Amazon Web Services.

OpenStack Technologies NeCTAR Uses	Amazon Web Services
Openstack Compute (NOVA)	EC2
Openstack Object Storage (Swift)	Simple Storage Service (S3)
Openstack Identity Service (Keystone)	Identity And Access Management (IAM)
Heat	Amazon Cloudformation
Openstack Block Storage (Cinder)	Elastic Block Store (EBS)
Openstack Dashboard (Horizon)	AWS Management Console
Openstack Image Service (Glance)	Amazon Machine Image
Ceilometer	

Table 1: NeCTAR Services and The Analogues To Amazon Web Services [22]

Performing sentiment analysis in the cloud overcomes certain issues. This paper explains the experience in implementing certain technologies to develop data mining to perform sentiment analysis upon Twitter data by using

NeCTAR as well as how NeCTAR support our system. In performing sentiment analysis in the cloud, there are several challenges to answer and they are:

1) Harvesting tweets as many as possible. To perform sentiment analysis, adequate numbers of data have to be available. A pilot data sample in this project is tweets from Boston, Massachusetts, United States.
2) Building robust and fault tolerant architecture to ensure data availability by exploiting a multitude of virtual machines (VMs) across the NeCTAR for harvesting tweets to get more tweets and handle possible errors occur.

The objectives of this paper are to:

1) Present other possible design systems to perform sentiment analysis which have been reported and used as the background of this project (Section 2).
2) Give appropriate details of implemented technologies and architecture to build the system and the reason why these technologies are applied (Section 3)
3) Present the scenarios and tools to harvest data and perform sentiment analysis and the result of sentiment analysis that the system performs (Section 4).
4) Present the issues occur during implementation and how errors are handled (Section 5).
5) Summarise and give future direction for similar projects (Section 6).

2. Related Work and Issue

Generally, twitter analysis is similar to analytics feature. Analytics is a popular technology of cloud services that makes several kinds of analysis, including sentiment analysis, possible. However, to run analytics in the cloud, complex and multiple tools are required. A cloud-based data analytics implementing the combination of MapReduce, SOA, and HBase in private cloud was reported in 2014 [29]. The project proved MapReduce as an effective solution since results of the mapper are divided on each processing node and data are taken from all the nodes [28, 29]. These data then combined into a single group to get the final merged results. This method in MapReduce is called “shuffling” and the partitions are called “buckets” [28]. There are several similar research which developed systems to perform sentiment analysis over Twitter data in the cloud. To gain more details about the infrastructures which are used to build data mining, business analytics and similar system, this paper refers to several projects in deploying data mining in the cloud. Some of them are not success story and some examples of cloud failures is provided in table 3.

In 2012, a system to perform sentiment analysis of U.S presidential election style by using information provided by Twitter was developed [16]. They avoid Using Twitter API in their system as it only provides 1% or less of its entire traffic. Instead, they used Gnip Power Track, a commercial Twitter data provider to gain relevant information they required. In terms of API, there are several possible API to use. Table 2 below are some examples of API to filter Twitter data which can be used in searching over Twitter data and the comparison.

API	**Details**
GNIP’s PowerTrack	Filter data by matching Tweets based on a wide variety of attributes, such as user attributes, geo location, and language. Data is delivered through constraint stream.

API	Details
Twitter Streaming API	Give complete answers of the filtered data in stream, even if it is not relevant. Low latency. For monitoring purpose.
GNIP's Search API	Provide answers of Twitter data searched in streaming form. Focus on historical, mostly to recent tweets in searching.
Search API	Focus on relevance of searching. The answers provided will be exact as if it is searched by using queries.

Table 2: Examples of API to Filter Twitter Data

Technically speaking, the fundamental aspect in sentiment analysis is classification of the data [2]. Naive Bayes model was used as a statistical classifier in presidential election analysis [16]. The same model was also used in many Twitter sentiment analysis researches [9, 11, 12] as Naive Bayes is a simple model and very common to use in text classification [9]. Techniques have gone further to determine whether or not opinions are subjective. Moreover, it is possible to know when an opinion on Twitter is related to a certain topic [25]. Above all, a technique that is used the most is by classifying the sentiment into three categories, which are positive, negative, and neutral then rank the filtered tweets in terms of these three categories [25, 26, 13].

In terms of data management, despite RDBMS (Relational Database Management System) is stated to be useful when dealing with a variety of data [29], managing big data in the cloud requires more than traditional RDBMS. To tackle RDBMS problem in cloud environment, a research in 2015 for geospatial data analytics combined and proposed NoSQL databases in distributed environment with traditional RDBMS [29]. This proposal was supported by another research which proposed Hbase, an example of NoSQL, to store massive imagery data [30 in 29]. Furthermore, similar collaboration of open source HBase and MySQL Cluster was proposed for semantic web management on a cluster [31]. However, combining HBase and Hadoop costs the ease in integrating and analysing various and separated data source [32].

To increase the performance, a project [33] used virtualisation to run several servers in the same virtual machine in cloud environment. This research stated that a layer between hardware resources and OS influence the performance of the virtual guest in virtualized environment. Despite it concludes that different virtualization solution can be implemented in the cloud, the use of virtualisation has other contradiction that affects the performance. However, virtualization introduces a degradation of performances as it gives additional overhead [33].

Cause	Effect	Details
Security Vulnerability	Restarted and shutdown instances	Suffered by Nectar, 13 May 2015
Heavy load on storage	Poor performance	A common issue for Hadoop users [38]
Expires SSL Windows [35]	HTTPS traffic. Unavailable service for hours	Suffered by Microsoft Azure in 2013 [35]

Cause	Effect	Details
Change in data centers	Unavailable service for hours	Suffered by Google Cloud in 2011 [37]
Unidentified cause	Unavailable service for hours	Suffered by Microsoft Azure in 2012 [36]

Table 3: Examples of Cause-Effect of Cloud Platform's Failures

3. System Design

3.1. Architecture

The system is built under NeCTAR which offers Infrastructure-as-a-Service (IaaS) service model in the cloud as this allows its user to apply different kinds of platforms and develop system in various programming languages [15]. NeCTAR cloud behaves like a real-life machine in a remote location and allows this "machine", which in NeCTAR is called instance, to run in it [18]. Cluster nodes are built with virtual machines. Considering that running multiple instances at a time reduces the loads of processing [14], this system applies three nodes with 8GB RAM, 2 Virtual CPU, 10GB Disk and 60GB Ephemeral Disk for each instance.

The system designed is known as Chirp, a peer-to-peer distributed platform. Chirp is used for collaboration across distributed systems such as clusters. Details on how it is implemented is illustrated in figure 1 with the components installed are summarised in Table 4. Three nodes run parallel by implementing load balance to avoid a single machine exhausts for fault tolerant purpose, specifically high availability.This allowsthe system to still run when a node failsand to enable the system to harvest faster with more machines. Chirp is developed for the following properties:
1) Declarative configuration.
2) Adjustable scaling.
3) Centralised access.
4) Load balancing.
5) Security aware.
6) Self-healing.
7) High availability.
8) No Single Point of Failure.

Declarative configuration and adjustable scaling is conducted through the use of a configurable custom platform launcher including infrastructure, such as number of nodes, system and application details. This system orchestrates Terraform (infrastructure configuration management software) and Ansible (system and application configuration management software). All operations are idempotent or in other words, can work one or more times with the same effect. In effect, the platform launcher's goal is to reach the platform's state to what is defined in its associated configuration files, realising the 'infrastructure as code' mantra.

Centralised access is provided through a third-party DNS provider, specifically DNSimple.com. Each node is registered on the same wildcard A record, which in this project is. '*.chi.t19.ninja'. If a user wants to reach the service monitor application at mon.chi.t19.ninja, for example, this user will be redirected to one of the nodes who in turn redirects to what node provides it.

Load balancing is conducted twofold. Since there are multiple A records on a domain, most clients will round-robin on the IP addresses, servicing as a 'poor man' capability for rudimentary load balancing and failover. Once a node is reached through DNS, HAProxy will read what subdomain is being accessed, and will redirect to the node it believes has the least number of connections that can provide the associated application. Since the applications currently installed are stateless, no sticky connections are required although it can be configured.

Security access is managed by HAProxy through the use of basic access HTTP authentication. Since the nodes expose itself to the internet through HAProxy, this has been done to provide the bare minimum in avoiding the compromise of any application by adversaries.

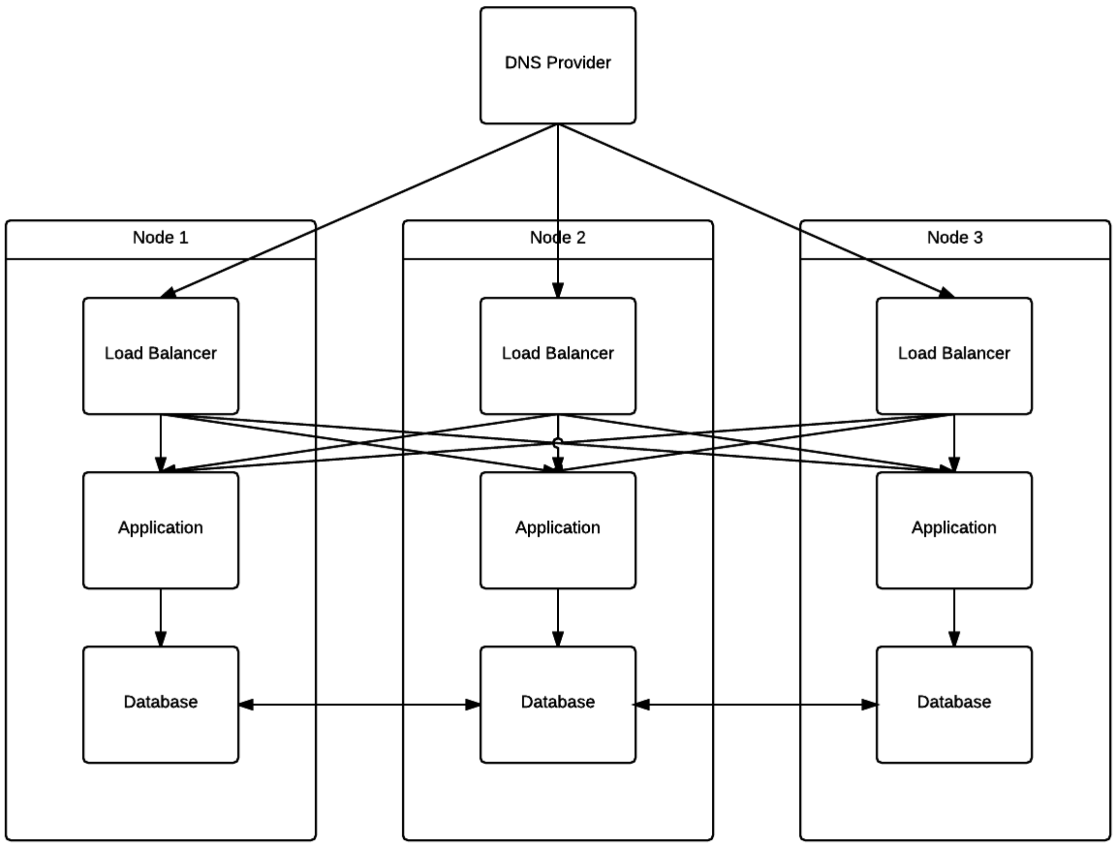


Figure 1: System Architecture

Self-healing is conducted at the application and system level. The monitoring service, Monit, is configured to supervise all applications and services on excess CPU and memory usage and program crashes, as shown in Figure 2. At this stage, Monit will automatically start the target after a timed delay to avoid CPU thrashing from constantly crashing targets. At the infrastructure level, if a node goes down, the platform launcher will need to be run again, provisioning a new instance to replace the old in the worst case and as such requires manual intervention.

High availability is conducted through the unison of multiple domain A records and HAProxy. HAProxy will automatically monitor other nodes and their associated endpoints, and remove them temporarily from the balancing schedule as necessary. At the DNS level, clients have a choice of multiple IP addresses, ensuring they can connect to at least one healthy node. Although this means the client may momentarily perceive a degraded system until it retries on another IP address, of which even domain record deletion on node failure may not resolve due to non-honouring time-to-live DNS providers. Ideally, the use of Virtual Router

Redundancy Protocol with an associated system such as 'heartbeatd' on a single A record to elastic/virtual IP would decrease the time the system is in a degraded state. However NeCTAR does not provide the Neutron, OpenStack project to provide "networking as a service" [40], and as such this option was out of consideration.

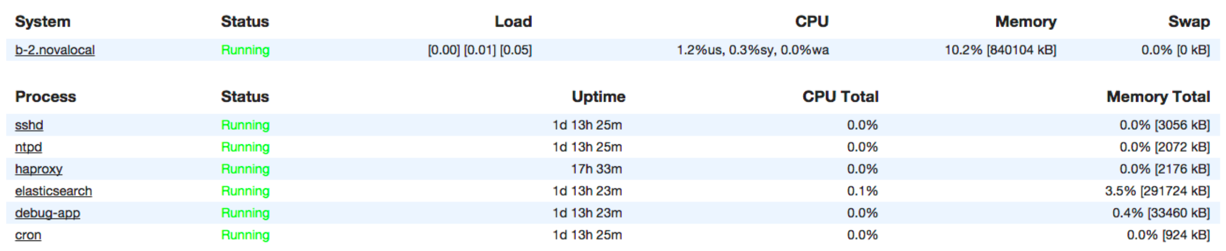
No Single Point of Failure becomes a natural property in a peer-to-peer architecture with redundant centralised access. Redundancy is at the access point is conducted through using a third party distributed DNS provider. The load balancers, applications and services run each node (peer). All applications and services installed that are cluster-aware are peer-to-peer architectures themselves (e.g. CouchDB with replication).

3.2. Programming Language

Despite the possibility of running several programming languages in NeCTAR, the system is best supported by Python and its libraries. A python library named `Tweepy` is used to download targeted timelines and save them in a database. Analytics is also performed by implementing a Python plugin named `TextBlob` for natural language processing (NLP) and the application is developed by using `Flask` framework. RESTful web service implemented is also developed by using Python.

Components	Details	Description
Host operating system	Ubuntu	Free, security, versatility and policy
Database	CouchDB	Text oriented. Consideration for data type is completely unnecessary.
Object storage	CouchDB	Provide MapReduce for analytics purpose
Web Service	RESTful	1. Use JSON which is identical to CouchDB 2. Enable interaction with users in a web application
Virtualization	Emulation	Run a complete virtual machine on an emulated server
Load Balancer	HAProxy	HAProxy operates as a reverse proxy and load balancer.
Service Management	Monit	Enable self-healing in the system.
Infrastructure Configuration Management	Terraform	1. Simple file based configuration 2. For Safely changing infrastructure [40]
Automation Tool	Ansible	1. Configure systems 2. Deploy software 3. Orchestrate more advanced IT.

Table 4: Summary of System Components and Why They Are Chosen



System	Status	Load	CPU	Memory	Swap
b-2.novalocal	Running	[0.00] [0.01] [0.05]	1.2%us, 0.3%sy, 0.0%wa	10.2% [840104 kB]	0.0% [0 kB]

Process	Status	Uptime	CPU Total	Memory Total
sshd	Running	1d 13h 25m	0.0%	0.0% [3056 kB]
ntpd	Running	1d 13h 25m	0.0%	0.0% [2072 kB]
haproxy	Running	17h 33m	0.0%	0.0% [2176 kB]
elasticsearch	Running	1d 13h 23m	0.1%	3.5% [291724 kB]
debug-app	Running	1d 13h 23m	0.0%	0.4% [33460 kB]
cron	Running	1d 13h 25m	0.0%	0.0% [924 kB]

Figure 2: Monit's Interface To Monitor Services and Application

3.3. Security and Fault Tolerance

There are several possible failures and security issues in running systems in the Cloud. Below are some of them [15]:

1) Authentication and authorization, where an individual may access some levels outside his/her privilege.
2) System failures, power outages, and other catastrophic events.
3) Data loss or leakage.

Other than fault tolerant architecture, security aspects implemented in the system to handle possible failures are:

1) Implementing a service healing monitoring, named Monit. Monit monitor running system, each process and file systems. if services of the system fails, it sends emails as alerts.
2) Replication. CouchDB is configured to replicate (master-master replication).

3.4. Simple Steps Guide in Running the System

Brief explanation about the system is listed below.

1) The system can be run by running our Ansible orchestration for cluster deployment and nodes configuration as follow: \TwitterAPI-master\devops\platform\build-cluster.sh, This will make sure all services are up and running.
2) Detailed readme deployment on this link: https://github.com/fanxeon/Twitter API/blob/master/devops/platform/README.md
3) Full project details on github: https://github.com/fanxeon/Twitter API

4. Harvest Tweets and Sentiment Analysis

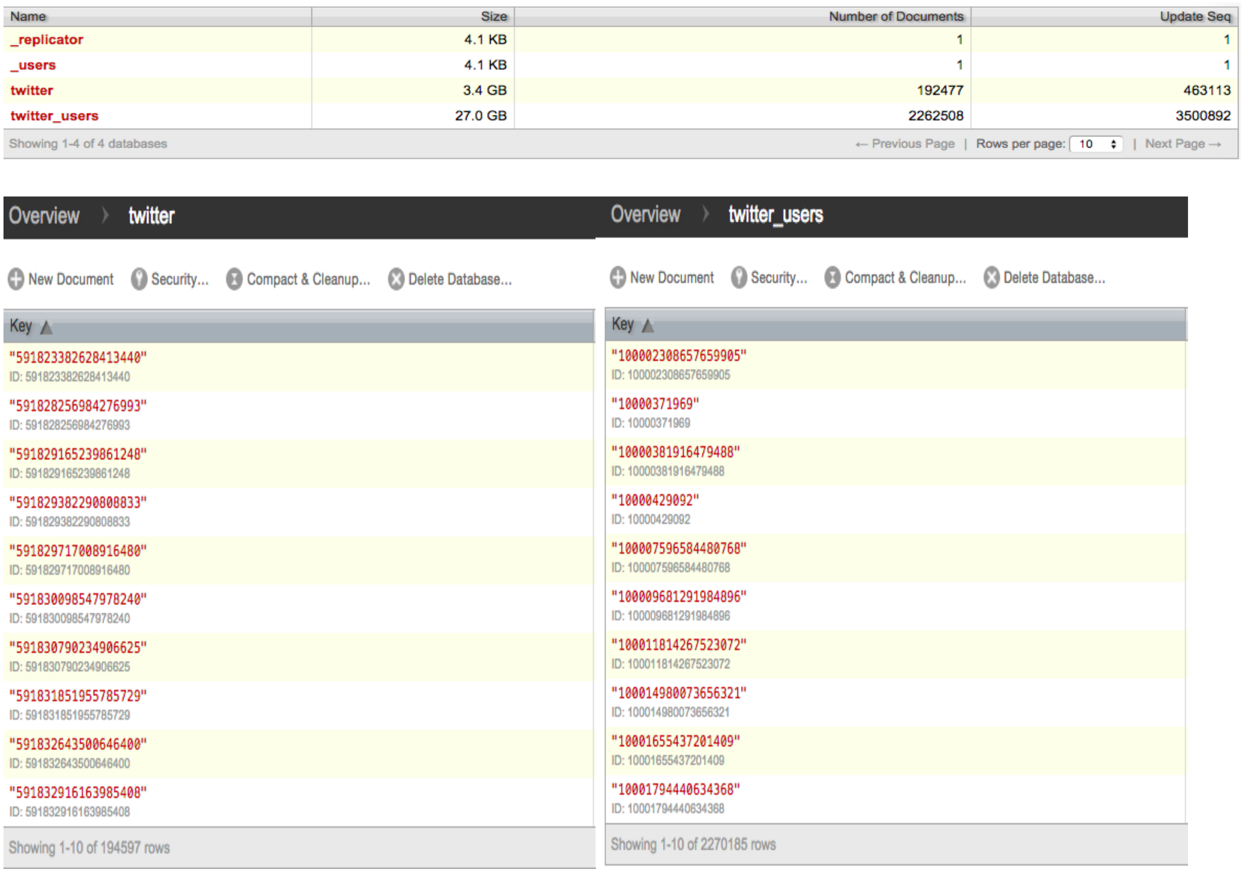
The pilot sample of this project is Twitter data from Boston, Massachusetts, United States. Parallel processing between nodes is applied to optimize the performance in harvesting the tweets in peer to peer configuration. By using Twitter Streaming API then moving to RestAPI, twitter data streams are loaded into storage. Captured image in Figure 3 below is the data we use in this project.

Steps taken in Harvesting Tweet are:

1) Getting the coordinate of location we aim, which are Boston. Latlong (http://www.latlong.net/) is applied in getting coordinate.
2) Using tweepy API for fast search REST requests based on the coordinates.
3) Twitter sampling to go through the city on a node and users on other nodesusing OS environment variables since CouchDB is replicated.
4) TextBlob text sentiments calculating in a new field of each document before saving using the unique tweet id_str parameter.
5) Working on different nodes skewed the number of tweets for user since it starts in ascending order, so shuffling the users was the key.
6) Moving towards the 12-factor app (Methodology for software as a

service) manifesto to support cloud deployment.

The homepage of application running in the cloud to perform sentiment analysis is shown by figure 4. As social analytics are highly used for monitoring politicians and brands [6, 7, 15 19, 20], we put Politics, Schools, Sports, Food, and Clothing in the homepage to make it easier in searching.



Name	Size	Number of Documents	Update Seq
_replicator	4.1 KB	1	1
_users	4.1 KB	1	1
twitter	3.4 GB	192477	463113
twitter_users	27.0 GB	2262508	3500892

Showing 1-4 of 4 databases ← Previous Page | Rows per page: 10 | Next Page →

Overview › twitter

New Document Security... Compact & Cleanup... Delete Database...

Key

"591823382628413440" ID: 591823382628413440
"591828256984276993" ID: 591828256984276993
"591829165239861248" ID: 591829165239861248
"591829382290808833" ID: 591829382290808833
"591829717008916480" ID: 591829717008916480
"591830098547978240" ID: 591830098547978240
"591830790234906625" ID: 591830790234906625
"591831851955785729" ID: 591831851955785729
"591832643500646400" ID: 591832643500646400
"591832916163985408" ID: 591832916163985408

Showing 1-10 of 194597 rows

Overview › twitter_users

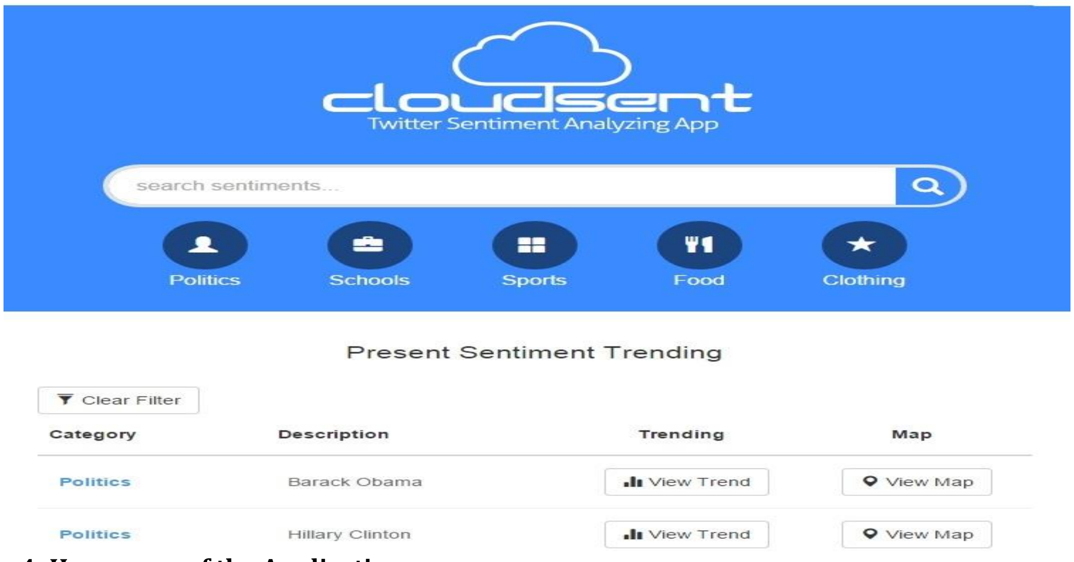
New Document Security... Compact & Cleanup... Delete Database...

Key

"1000023086576599O5" ID: 100002308657659905
"10000371969" ID: 10000371969
"10000381916479488" ID: 10000381916479488
"10000429092" ID: 10000429092
"100007596584480768" ID: 100007596584480768
"100009681291984896" ID: 100009681291984896
"100011814267523072" ID: 100011814267523072
"100014980073656321" ID: 100014980073656321
"10001655437201409" ID: 10001655437201409
"10001794440634368" ID: 10001794440634368

Showing 1-10 of 2270185 rows

Figure 3: Total Dataset in CouchDB (Upper), Number Tweets (Left), and Number Users (Right)



Present Sentiment Trending

Clear Filter

Category	Description	Trending	Map
Politics	Barack Obama	View Trend	View Map
Politics	Hillary Clinton	View Trend	View Map

Figure 4: Homepage of the Application

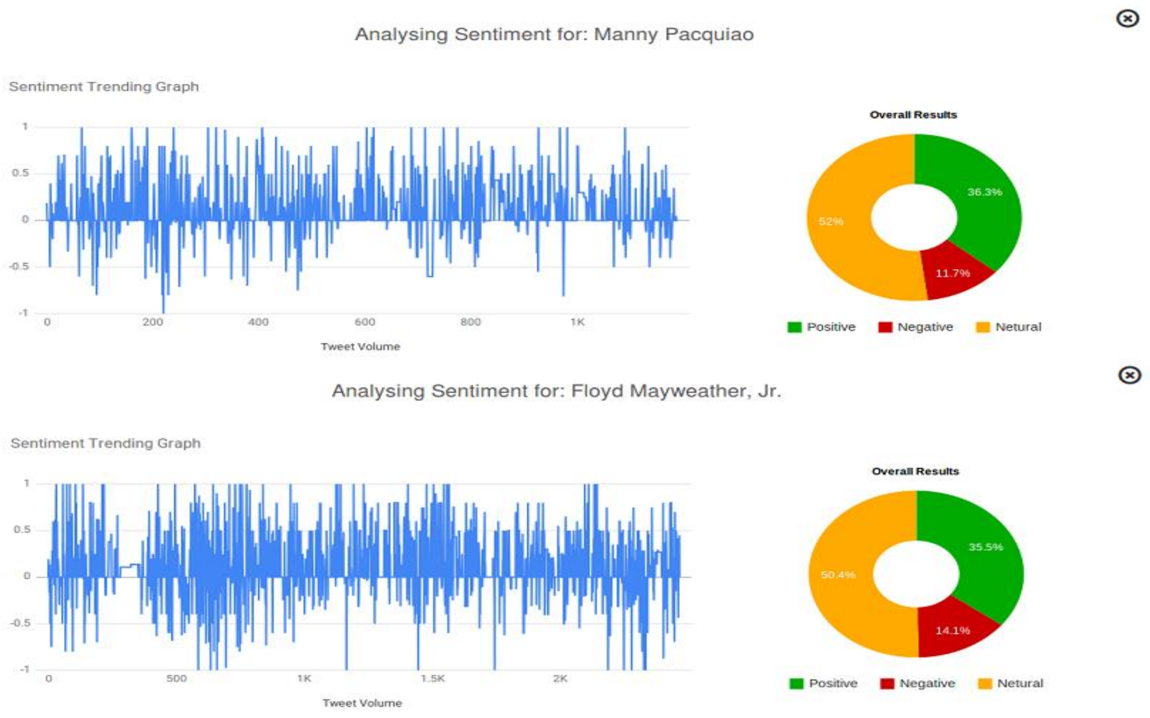


Figure 5: Examples of Analysis Performed

Natural Language Processing / Sentiment Analysis are performed with a tool called TextBlob which is integrated to Flask. CouchDB's MapReduce is applied to build views and make the retrieval faster to show data and graphs. Data is displayed in the form of graphs, pie-charts and map. For example, we made comparisons of one of the famous boxing fight between Floyd Mayweather and Manny Pacquiao as shown in Figure 5. Figure 7 shows sentiment analysis over the players and number of tweets per day. The result shows Mayweather was more popular than Manny in Boston. Figure 6 tells about people sentiment from the location they tweeted and their moods during the week days. Figure 8 is about the number of tweets and the equipment used for tweeting. The result shows that most tweets were from iPhone users. Figure 9 shows analysis performed based on tweets about restaurants.

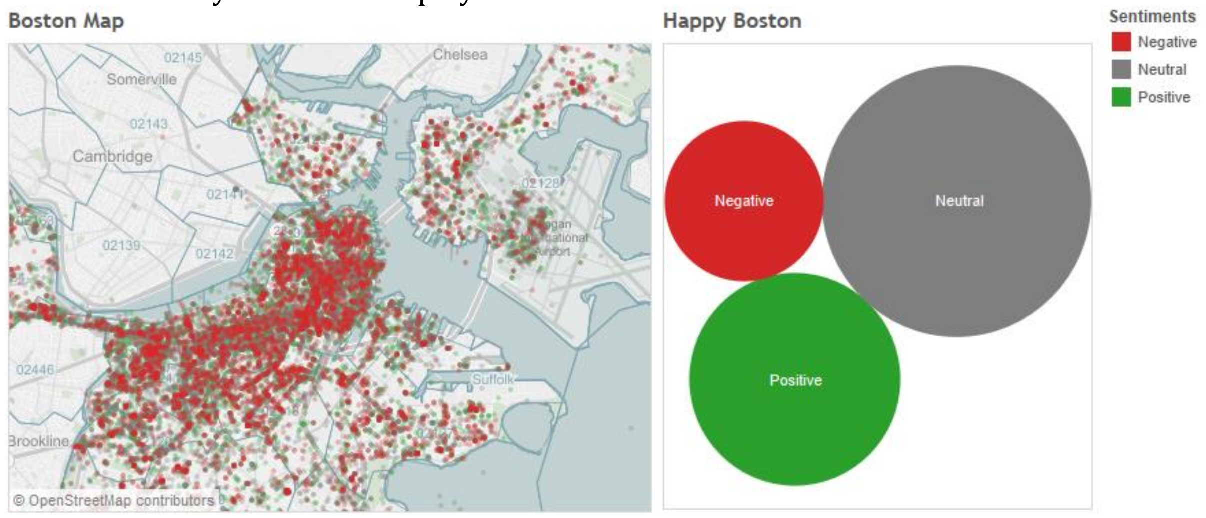
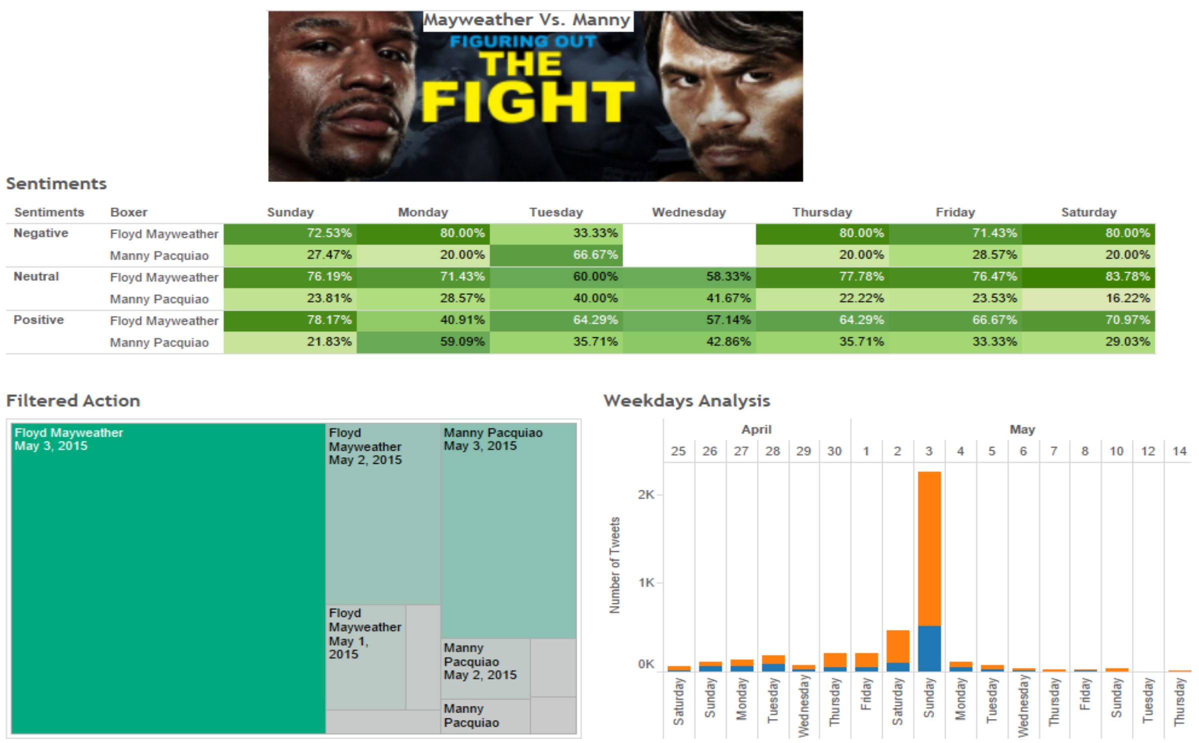


Figure 6: Sentiment Analysis on Happiness



Sentiments	Boxer	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Negative	Floyd Mayweather	72.53%	80.00%	33.33%		80.00%	71.43%	80.00%
	Manny Pacquiao	27.47%	20.00%	66.67%		20.00%	28.57%	20.00%
Neutral	Floyd Mayweather	76.19%	71.43%	60.00%	58.33%	77.78%	76.47%	83.78%
	Manny Pacquiao	23.81%	28.57%	40.00%	41.67%	22.22%	23.53%	16.22%
Positive	Floyd Mayweather	78.17%	40.91%	64.29%	57.14%	64.29%	66.67%	70.97%
	Manny Pacquiao	21.83%	59.09%	35.71%	42.86%	35.71%	33.33%	29.03%

Figure 7: Examples of Analysis Performed for Boxing Scenario

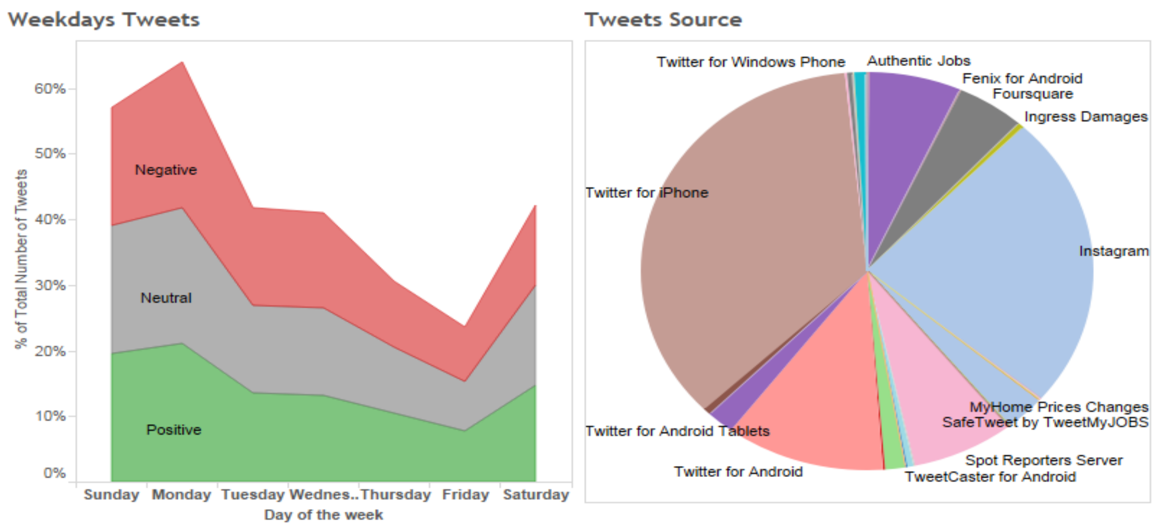


Figure 8: Numbers of Tweets over The Week and Type of Devices Used

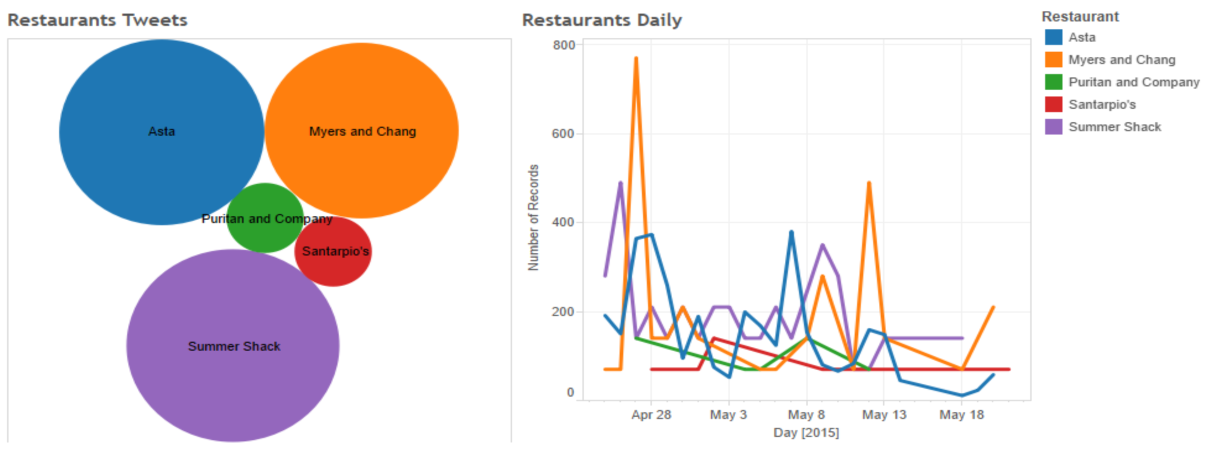


Figure 9: Sentiment Analysis over Restaurants

5. Implementation, Results, and Issues

5.1. API

In gaining data to be harvested, we used several API's, such as Twitter Search API and Streaming API. Harvesting by using Twitter Search API gave us 120000 tweets (0.7GB) in 6 days. Twitter's end also caused the harvesting to be slow with a geocode parameter paginating which only shows 0-20 in Boston due to search requests searching of tweets within 1000km of Boston [43]. We tried a different approach which gave us 1.4 GB in a half day by crawling users. There are other approaches possible depend on the objectives of data filtering. Some of these approaches are briefly described in Table 2. Twitter API has some limitations in terms of the requests per window which is called rate limit that gives 450 search calls app authentication per 15 minutes with a maximum of 100 tweets per call. We also reported another issue to Twitter where the coordinates overlap for filtration and it requires an extra check on place parameter. To keep the system safe when it hits the rate limit, Tweepy API was applied to automatically set slot to sleeps for 2 minutes.

5.2. Security

A security issue was captured during our research. In May, 13 2015 NeCTAR was attacked by Venom (Virtualized Environment Neglected Operations Manipulation) a security issue which attacks computer virtualization platforms [34]. Consequently, all instances were being shutdown and restarted.

5.3. Cloud Platform

As NeCTAR has not fully matured yet a number of issues were faced during system development of which may most likely be resolved in the future. Most were intermittent, the ones listed below tend to be recurrent.The lack of Neutron networking service required us to reach a compromise in utilising multiple domain records and HAProxy hybrid for load balancing and fault tolerance instead of more robust methods such as VRRP (Virtual Router Redundancy Protocol) failover and HAProxy.

The equivalent of S3, being object storage Swift has immediate issues with the latest Python Boto library 2.38.0. Swift uses a variant of EC2 signature V4 API, of which no longer conforms with the latest EC2 API Boto is targeted to, e.g. retrieving a 'bucket' name from EC2 now uses a HEAD HTTP request, something Swift on NeCTAR is incapable of. Even if Boto is switched to signature V4 mode, Boto is hard-coded to look for named regions for storage of which NeCTAR does not have. Looking into the source code of Boto, a workaround is to retrieve all buckets at once (method 'get_all_buckets'), which uses the GET HTTP request, of which should then be manually filtered for the right bucket. The most relevant issue to this is #2916 on Boto'sGitHub bug tracking system.

There are cases where resources provisioned on NeCTAR do not delete on request through the dashboard. This state will last indefinitely without intervention. A workaround is to use the Nova CLI client directly to delete the target resource.

NeCTAR officially has stated the limitation that using OpenStack Heat requires admin access to a project. As much as we wanted auto-scaling and self-healing at the infrastructure level, we decided to not use Heat as it is not fully developed on NeCTAR yet.

Pro	Contra
1. Free 2. Customisable resources (admin access to instances) 3. Scalable 4. Accessible 5. Relatively easy-to-use dashboard	1. Immature (number of standing issues) 2. Hardware behind each instance is third generation or worse for many AZs 3. Slower support compared to commercial alternatives 4. Less features than alternative IaaS providers

Table 5: Summary of Pro and Contra in Using NeCTAR

A number of issues were faced with provisioning on NeCTAR. NeCTAR's availability zones (AZ) often run out of available hosts to provision. This happened the most in the 'melbourne-*' availability zones. As this happened during most of our development schedule, we utilised the 'sa' AZ. Near the end of the schedule adjustments were made to the project quota which no longer allowed us to provision volumes outside of 'melbourne-*' AZs. Fortunately around a week later the 'melbourne-qh-uom' AZ was released, albeit with a series of immediate issues of which we worked with NeCTAR support to resolve: routing issues to the provisioned instances, lack of available hosts and no corresponding 'melbourne-qh-uom' volume AZ. On NeCTAR support team's guidance, we used 'melbourne-qh' volumes with 'melbourne-qh-uom' instances due to using the same shared storage at implementation level. It is uncertain if this follows best practice as availability zones tend to be logically isolated, of which in this case the property is violated by known implementation details."

The naming of resource endpoints are custom, i.e. 'nova' compute service is renamed to 'Compute Service', of which breaks some development operation programs such as previous versions of the 'gophercloud' library and its associated client libraries. This is not a shortcoming of NeCTAR itself, as it is related more to external tools that can be hard-coded to look for certain properties of OpenStack deployments.

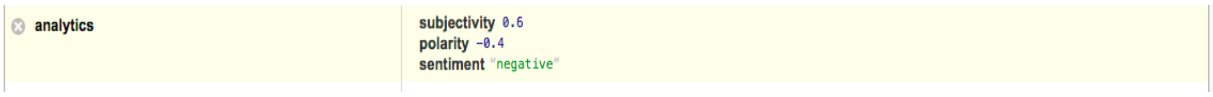
Being allocated 4 nodes with 2 vCPU each restricted some of our designs. For instance a high availability master-slave setup will require at least 5 nodes (3 for master using quorum-based master election, 2 slave nodes to see scheduling is functioning). Using 8 nodes with 1 vCPU each does not suffice, as there are numerous development operation programs such as Consul that require at least 2 cores to function correctly.

Someother possible issues in using cloud platform are captured in table 3, including its causes and effects which have ever recorded. Although it has several issues, NeCTAR in particular purpose can be the proper cloud platform to use. Table 5 summarises the strength and weakness of using NeCTAR.

5.4. Database

During harvesting, a problem other than API occurred due to the null values in the retrieved JSON that has to be checked, such as the place which caused lots of errors at the start. It is handled by passing through some exceptions.Duplication was not an issue from the start since we used the twitter id_str which is always unique as a document id before insertion to couchdb which makes it very efficient, on the other hand after collecting many of the tweets and thinking about the best way to calculate sentiment analysis was to add the analysis when harvesting using TextBlob, therefore we run again through all the old data for a massive update.

We adjusted the harvester to remove sampling bias by removing duplicate twitter id. A result in figure 10 below is an example of how we do analytics for tweet “Katherine and Min just kicked me out” (and some more harsh words in the end). Textblob is used to analyse positive, negative, and neutral trends for specific topic [13].The problem of working in CouchDB is that its instances in isolation.



```
analytics        subjectivity 0.6
                 polarity -0.4
                 sentiment "negative"
```

Figure 10: View in CouchDB to Store Analytics Result which Perform by Using TextBlob

6. Conclusion and Future Improvements

The system relies on TweetMining RESTful (python script) for streaming processing to harvest tweets plus calculating analysis on fly and to perform streaming processing, the system has to be highly available [42]. To be highly available, replication is applied in the current system to guarantee the system is highly fault tolerant. The implemented architecture makes the system robust as if one node dies, the other nodes keep running the system. However, it costs inefficiency in memory and resources used as all data is duplicated and redundant works are processed in each node.

In harvesting data with the characteristics as shown in Table 6, problem may arise due to Twitter API, database it uses and CouchDB. The speed of harvesting tweets can be increased by choosing the right API unless the problem comes from Twitter like the one we experienced. Another aspect to make it faster is by harvesting on different nodes.

Volume	Velocity	Variety	Veracity
Low (27 GB)	Medium	Low	High (exact tweets)

Table 6: Characteristics of Dataset

For future improvements, there are various cloud platforms and tools to build social analytics in the cloud. Careful consideration will focus on the objectives of the system which run in the cloud. The current running system was not literally designed to handle enterprise data and perform sophisticated analysis. However, we have implemented several tools during our research time that could work in building more sophisticated analysis and perform better in handling big data.

One of our endeavours was running Apache Spark. In terms of big data processing, Spark has Spark Streaming for streaming processing. Spark Streaming use HDFS for high availability and fault tolerant. Its features make Spark more suitable for deploying machine learning and more sophisticated analytics. Spark can collaborate with NeCTAR’s swift for data storage. Once they collaborate, data can be set to duplicate in each node. Spark has MapReduce functionality implicitly. Data can be interactively visualized by integrating Tableau and Spark SQL. The difference of HadoopMapReduce and Spark is that the latter utilises smart dataset caching to speed up iterative computations.

"Put your datacentre and cloud on autopilot.[41]" If this is the aim, Mesosphere can be applied in the system. Mesosphere is a data centre operating system (DCOS) which organizes all machines, virtual machines, and cloud instances into a single pool of shared resources [41]. However, there are constraints on how the application has to be developed, such as restrictions on how the web-service and analyticsapplications are developed.
CoreOS, a similar system to MesoSphere with differing distributed properties, is also applicable. Implementing Mesosphere and CoreOS gain better self-healing, migratable applications, and resource scheduling than the current running system. However, for example, it is required to approach the '12 factor app' manifesto in application design, of which would have slowed down development time unless previously experienced with it.

References

[1] Pak, A., &Paroubek, P. (2010, May). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *LREC* (Vol. 10, pp. 1320-1326).

[2] IvanaMamic, L., & Arroyo Almaraz, I. (2013). How the Larger Corporations Engage with Stakeholders through Twitter. *International Journal Of Market Research*, *55*(6), 851-872. doi:10.2501/IJMR-2013-070.

[3] Shaw, R. (2015). The New Innovation. *Marketing Insights*, *27*(1), 36-40.

[4] Lu, Y., Wang, F., &Maciejewski, R. (2014). Business intelligence from Social Media: a Study from The Vast Box Office Challenge. *IEEE Computer Graphics And Applications*, *34*(5), 58-69. doi:10.1109/MCG.2014.61

[5] Rambocas, M., & Gama, J. (2013). *Marketing research: The role of sentiment analysis* (No. 489). Universidade do Porto, Faculdade de Economia do Porto.

[6] Schweidel, D. A., & Moe, W. W. (2014). Listening In on Social Media: A Joint Model of Sentiment and Venue Format Choice. *Journal Of Marketing Research (JMR)*, *51*(4), 387-402.

[7] Terdiman, Daniel (2006), "Why Companies Monitor Blogs," CNET, (January 3), [available at http://news.cnet.com/2100- 1030_3-6006102.html].

[8] Feldman, R. (2013). Techniques and Applications for Sentiment Analysis. *Communications Of The ACM*, *56*(4), 82-89. doi:10.1145/2436256.2436274

[9] Go, A., Huang, L., & Bhayani, R. (2009). Twitter Sentiment Analysis. *Entropy*, *17*.

[10] Kouloumpis, E., Wilson, T., & Moore, J. (2011). Twitter Sentiment Analysis: The Good The Bad and The Omg!.*ICWSM*, *11*, 538-541.

[11] Go, A., Bhayani, R., & Huang, L. (2009). Twitter Sentiment Classification Using Distant Supervision. *CS224N Project Report, Stanford*, 1-12.

[12] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., &Passonneau, R. (2011, June). Sentiment Analysis of Twitter Data. In *Proceedings of the Workshop on Languages in Social Media* (pp. 30-38). Association for Computational Linguistics.

[13] LarteyLaryea, B. N., Chi-Hwan, C., In-Sun, J., Kyung-Hee, L., & Wan-Sup, C. (2015). Web Application for Sentiment Analysis Using Supervised Machine Learning. *International Journal Of Software Engineering & Its Applications*, *9*(1), 191-200. doi:10.14257/ijseia.2015.9.1.17.

[14] Khajeh-Hosseini, A., Greenwood, D., &Sommerville, I. (2010, July). Cloud migration: A case study of migrating an enterprise it system to iaas. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on* (pp. 450-457). IEEE.

[15] Marinescu, D. C. (2013). *Cloud Computing. [electronic resource] : Theory and Practice*. Burlington : Elsevier Science, 2013.

[16] Wang H., Can D., Kazemzadeh A., Bar F, and Narayanan S. (2012). A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 115–120, Jeju, Republic of Korea

[17] CouchDB Websites ()

[18] NeCTAR Websites (https://www.NeCTAR.org.au/sites/)

[19] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ...&Stoica, I. (2009). Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, *28*, 13.

[20] Washington Post Case Study: Amazon Web Services [online]. Available from: http://aws.amazon.com/ solutions/case-studies/washington-post/.

[21] Jackson, K. (2012). *OpenStack Cloud Computing Cookbook. [electronic resource]*. Birmingham :Packt Publishing, 2012.

[22]https://www.openstack.org/user-stories/NeCTAR/

[23] http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[24] Penchikala, Sprini. (2015, January). Big Data Processing with Apache Spark: Introduction. *Infoq*. (available at: http://www.infoq.com/articles/apache-spark-introduction).

[25] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis."*Foundations and trends in information retrieval* 2.1-2 (2008): 1-135.

[26] Baccianella, S., Esuli, A., &Sebastiani, F. (2010, May). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *LREC* (Vol. 10, pp. 2200-2204).

[27]https://support.rightscale.com/12-Guides/Designers_Guide/Cloud_Solution_Architectures/Cloud_Computing_System_Architecture_Diagrams/

[28] Mohanty, S., Jagadeesh, M., &Srivatsa, H. (2013). *Big data imperatives : enterprise big data warehouse, BI implementations and analytics*. [Berkeley, California] :Apress, [2013].

[29] Li, Z., Yang, C., Jin, B., Yu, M., Liu, K., Sun, M., & Zhan, M. (2015). Enabling Big Geoscience Data Analytics with a Cloud-Based, MapReduce-Enabled and Service-Oriented Workflow Framework. *Plos ONE*, *10*(3), 1-23. doi:10.1371/journal.pone.0116781.

[30] Chen, J., Zheng, G., & Chen, H. (2013, June). ELM-MapReduce: MapReduce accelerated extreme learning machine for big spatial data analysis. In *Control and Automation (ICCA), 2013 10th IEEE International Conference on* (pp. 400-405). IEEE.

[31] Franke, C., Morin, S., Chebotko, A., Abraham, J., & Brazier, P. (2013). Efficient Processing of Semantic Web Queries in HBase and MySQL Cluster. *IT Professional*, *15*(3), 36-43. doi:10.1109/MITP.2012.42.

[32] Taylor, R. C. (2010). An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC Bioinformatics*, *11*1-6. doi:10.1186/1471-2105-11-S12-S1

[33] Suresh, S., & Kannan, M. (2014). A Performance Study of Hardware Impact on Full Virtualization for Server Consolidation in Cloud Environment. *Journal Of Theoretical & Applied Information Technology*, *60*(3), 556-567.

[34] http://venom.crowdstrike.com/

[35] Miller, Rich. (February 25, 2013). Windows Azure Cloud Crashed by Expired SSL Certificate. Data Center Knowledge. (available at http://www.datacenterknowledge.com/archives/2013/02/25/windows-azure-cloud-crashed-by-expired-ssl-certificate/).

[36] Miller, Rich. (July 26, 2012). Windows Azure Hit by Cloud Outage in Europe. Data Center Knowledge. (available at http://www.datacenterknowledge.com/archives/2012/07/26/windows-azure-down-europe/).

[38] Scarpati, Jessica. Big Data Analysis in The Cloud: Storage, Network and Server Challenge. SearchTelecom. (available at http://searchtelecom.techtarget.com/feature/Big-data-analysis-in-the-cloud-Storage-network-and-server-challenges).

[39] http://ccl.cse.nd.edu/software/manuals/chirp_protocol.html

[40] https://www.terraform.io/
[41] https://mesosphere.com/
[42] Stonebraker, M., Çetintemel, U., & Zdonik, S. (2005). The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, *34*(4), 42-47.
[43] https://twittercommunity.com/t/search-api-returning-very-sparse-geocode-results/27998

Details

Tell how we harvest as many tweets as possible from Boston.	Section 4
Provide a variety of social media dataanalytic scenarios	Section 4
Issues and challenges in using the NeCTAR Research Cloud	Section 5.3
Describe in detail the limitations of mining twitter content and language processing (e.g. sarcasm) and outline any solutions developed to tackle such scenarios.	Section 4 and Section 5.4
How removing duplicates of tweets be handled.	Section 5.4
Describe how the system was designed to be robust and provide any degrees of fault tolerance.	Section 2.1 and Section 2.5
Describe the system functionalities.	Section 3
Explain scenarios supported and why, together with graphical results, e.g. pie-charts/graphs of Tweet analysis and snapshots of the web apps/maps displaying certain Tweet scenarios.	Section 4 in Figures 5 -- 9
Provide simple user guide for testing.	Section 3.4 and Github
Systemdesign and architecture and how/why this was chosen	Table 4 and Section 3
Give discussion on thepros and cons of the NeCTARResearch Cloud and tools and process for image creation and deployment.	Section 5.3