Twitter Analytics

Khushbooba Solanki (*Author*)

Computer Engineering Department

Santa Clara University

Santa Clara, United States

*Abstract*—Nowadays, cloud applications are increasingly being used in industries for business and analytics. Cloud computing offers the benefits of virtualization, the elasticity of resources and elimination of cluster setup cost and time to cloud applications users. On the other hand, big data management on the cloud is very expensive. Let us say that you form a cloud-based startup company with one or two colleagues from the cloud computing course. A client has approached your company and several other companies to compete on a project to build a web service for Twitter data analysis. The client has provided a raw dump of tweets that runs into tens of terabytes. The dataset must be stored within your cloud services. The web service should be able to handle a specific number of requests per second for several hours. The client has a limited budget for this solution. The task is to build an effective and cost efficient solution utilizing Amazon Web Services resources. This paper also compares traditional analytic Web Service and ‘Twitter Web Service’ in terms of cost and efficiency. We also propose the flow, design and procedure that would be followed to implement the same.

*Index Terms*— Analytics; Tweets; Cost Efficient; Web Service; Twitter Analytics

Introduction

These days, all the companies have high volume, high velocity and high variety of the databases. The main motivation is to build the web service for a particular web application called twitter for its big data analytics. All analytical services are very expensive so the future goal is to make it very fast and cost efficient. In this project, we will try to build a cost effective, cost efficient and fast web service solution utilizing AWS.

If we want to implement the same services on different famous tools, then the cost and disadvantages using them are as below:

Salesforce analytical cloud: $900 cloud setup cost + $3000 to Manage, to upload and to combine data = $3900 (Not included the price of per TB computation and map reduce result storage costs. If we will include it then total cost becomes more than $4435.)

Tableau Twitter Analytics: $2,000(year subscription) + $500 (setup cost) = $2500**. Disadvantages are that this analytics service only allows you to perform actions on the structurally formatted files. It is impossible to implement the custom logic or to modify the text and get a sentimental score for each tweet over 12TB of data.**

Traditional cloud computing analytical services: To analysis of the big data solutions, it is highly motivational to use mapper and reducers configured with Hadoop environment. Hadoop configured mapper and reducers are very heavy for the execution and take more than 16 hours to analyze 12TB of data with 1 master and 11 large slave clusters. If we will use this map-reduce system every 15-day time period in then it will take 16\*12\* 0.215 = 41.28 \* 24 = $990.72 per year on just Elastic Map Reduce, AWS. After map reduce jobs, it is very important to use cloud tools very smartly. The inefficient program can lead us to micro to medium instances. Medium instances cost almost double if they would like to use it for a year.

Following is also the example of heavyweight EMR jobs. It cost us $113 for processing 12TB of data twice.

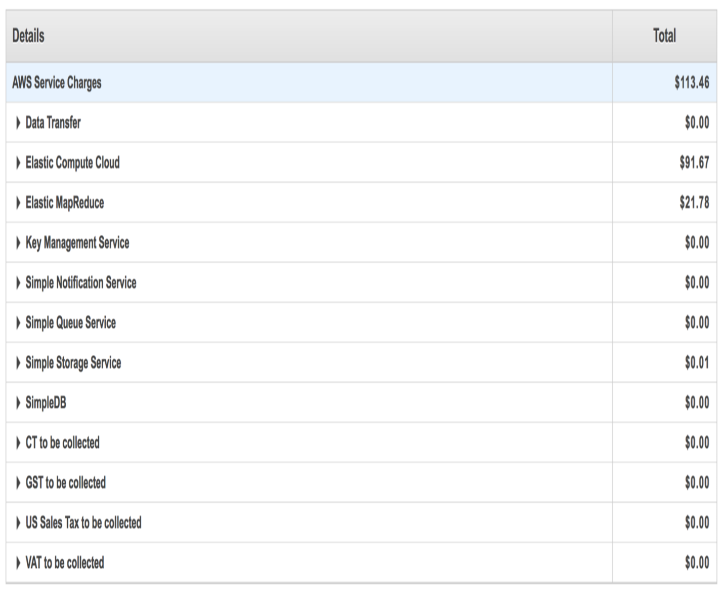


Figure 1 Traditional Analytics Cost

To use big data analytics on the cloud is very expensive. When it comes to 10TB of data most of the web services costs a lot. The big data analytics on the cloud is very expensive because of the computation, abstracting, transforming and loading the data with the time sensitive applications. Web services become costly when it comes to computation and use of the instances. Some instances do only computations and it takes the same time, efforts and cost compared to those instances which are used for actually storing the data and performing different actions on it. So, this problem is very important to get resolved.

Results and importance to solve this problem are: (i) Use EC2 spot instances: Spot instances allow you to bid on spare Amazon EC2 computing capacity. Since Spot instances are often available at a discount compared to On-Demand pricing, you can significantly reduce the cost of running your applications, grow the computation capacity and throughput for the same budget, and enable new types of cloud computing applications. (ii) Less number of Main instances: All the operations will be done using just 3 micro instances with the lower use (iv) Load Distribution: (a) Web service should be effective to hold a specific number of requests per second for hours. (b) The web service must be able to balance the varying load and also achieve the minimum throughput as mentioned in the requirement. (c) The Web service must not refuse queries. (d) Meet the requirements of throughput and latency for queries for the workload.

The main contribution of our project is: (i) Robust data foundation (ii) Fast Time to Value (iii) Improved Collaboration (iv) Lower total cost of ownership (v) Fast and efficient recovery system

# Problem analysis and proposed solution

The optimization at all stage is very important. Code and script at any stage should be very optimal, compressed and light weighted. We don’t know the future of cloud computing and big data platforms so optimized code and smart use of platform can help us to save cost at all times.

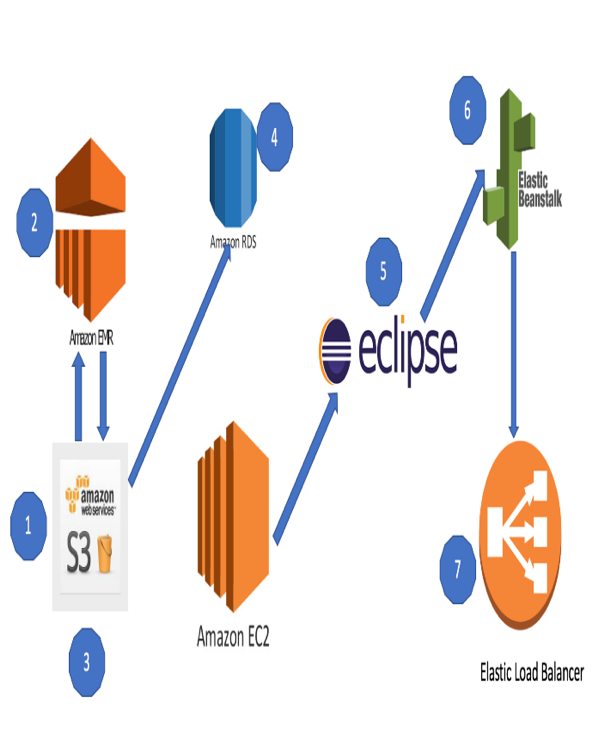


Figure 2 Flow of proposed solution

# Implementation and evaluation:

Implementation has been divided into 7 steps.

Step 1: setting up the s3 bucket.

* We got an S3 bucket from the client which is raw and dump data.
* Here each .gz file is around 153.4MB and total 1.2TB of the zip file.
* When we unzip all the files which are in the s3 bucket we would get **12TB** of approximate data for map reduce.

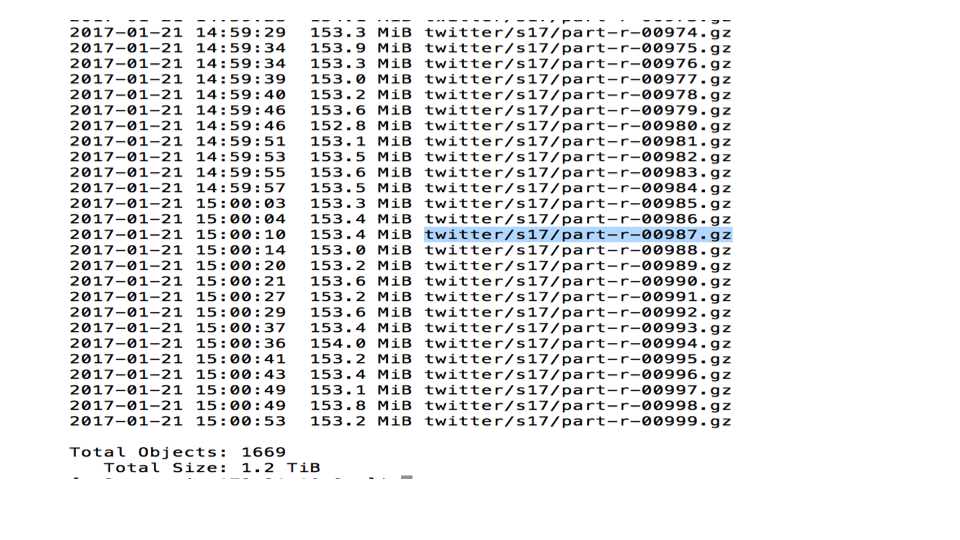


Figure 3 s3 bucket with 1TB zip files

Step 2: Creating mapper and reducers free from Hadoop libraries:

* Now, a client asked for 4 different queries and instructed us for clean data with the sentimental score and removing bombs. This can be possible when only we have our custom mapper reducer logic. If we have used other technologies like hive or pig, then we must run the SQL queries but It is not possible to do custom actions on each row of the data. It will be only helpful to get structural data but not for the other operations.
* This kind of custom logic operations can only be done using java, python or other object oriented languages.
* We are removing bad words to \*\*\*\*\* and find the sentimental score to each query.
* These are the important functions of mapper and reducer classes in java.



Figure 4 Sentimental Score Computation

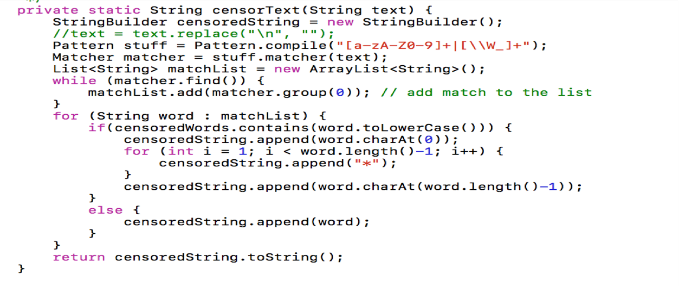


Figure 5 Sensor the text

Please find the entire code into my GitHub account: https://github.com/khushbooba91/TwitterFinalProject

Step 3: Testing mapper and reducer jar

* Once we have tested the small size input by executing java files in development tool we need to create a .jar file for Elastic Map Reduce job. It is very important to test it in local pipelines. Run the jar files with a little big chunk of the data into your console. It will save time and cost because if you directly start performing the map reduce job on EMR, it will fail after several hours when it comes to 12 Tb data. It will waste the money and time. Testing on EMR is very costly.
* Commands used:
  + Java –classpath ETL.jar Mapper input.tar.gz > input1 (Store mapper results into input1 file).
  + Java –classpath ETL.jar Reducer input1 > output1(Store reducer results into output1 file).
  + It is very important to check the results in output1 whether that gave you the proper result of removing bad words and sentimental scores or not.



Figure 6 Result of sentimental score and text conversion

If we won’t do this step, we have to spend lots of money on EMR clusters and S3 and hours of time.

Step 4: EMR map reduce job.

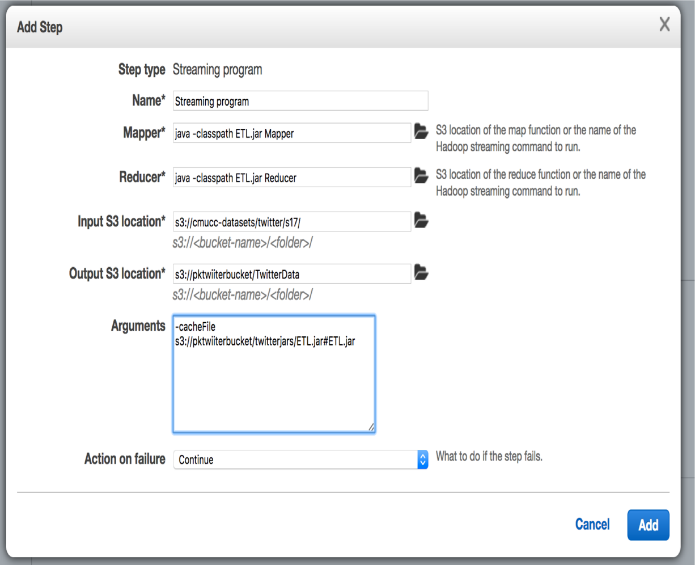


Figure 7 Giving custom mapper reducer into streaming program in EMR

* Store jar files into your s3 bucket.
* All the result data of map reduce will be stored in the s3 bucket specified folder.

Total Cost = Total Cost: 0.190\*6[Master Extra Large M3 Node for 6 hours] = $1.14, 0.147\*6\*6[6 extra-large M4 slave nodes] = $5.292. $1.14 + $5.292 = $6.432 is the total cost for custom map reduce on 12TB data. Let us say in a month you will use it twice for different terabytes of data: $12.864 a month + $ 3 approx. (EC2 instance for virtual memory to the cluster) = $15.864 a month

Step 5: Monitoring the resources. If some container performance keep getting down or hardware usage keep getting high that means it is high chances to get the following error.

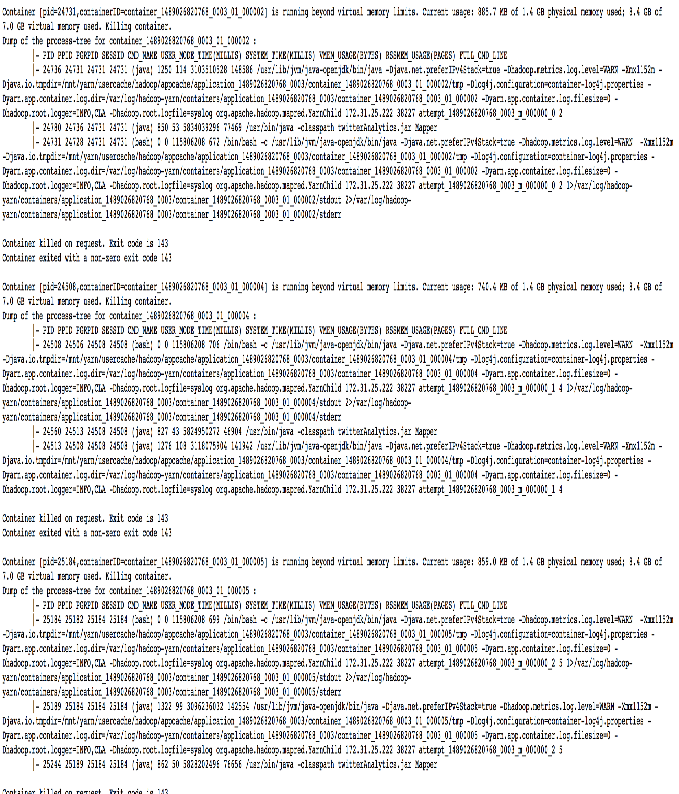


Figure 8 Container Out of Virtual memory Space Error if we have inefficient algorithm for mapper and reducer

This can be fixed in two ways. 1. The increment of slave nodes into the clusters 2. Optimization of the given mapper and reducer cost. I have always chosen to optimization of the mapper and reducer cost. Most of the times this occurs because the data structures used in mapper and reducers are going out of the bound because it is storing data temporarily. So, here in java proper garbage collection and memory management can help us to save cost by saving cost with fewer slave nodes.

These are the charts show us the constant performance while doing the map reduce job.

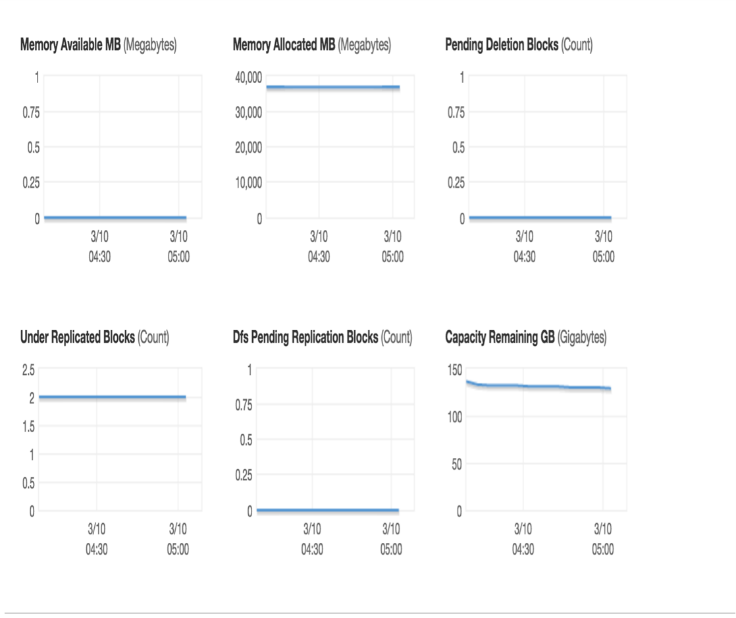


Figure 9 Memory Management performance

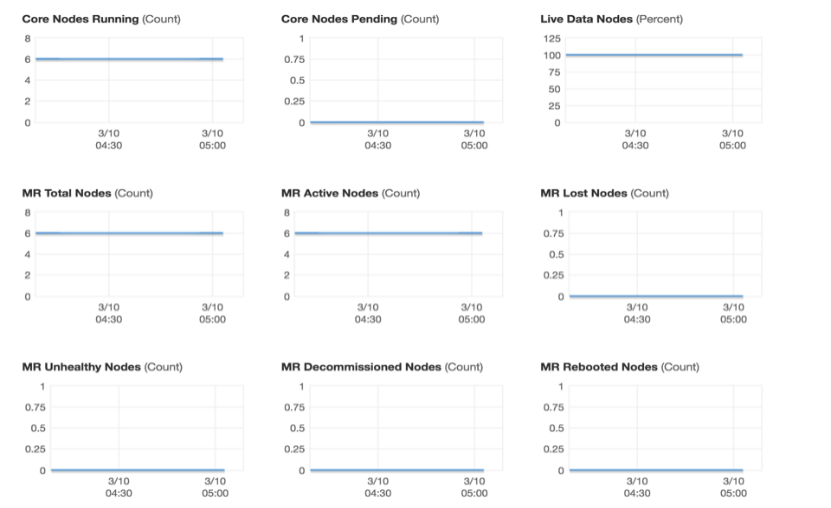


Figure 10 Container Node Steady Performance

Step 6: S3 to RDS:

The map reduce results will be stored into the specified S3 bucket.

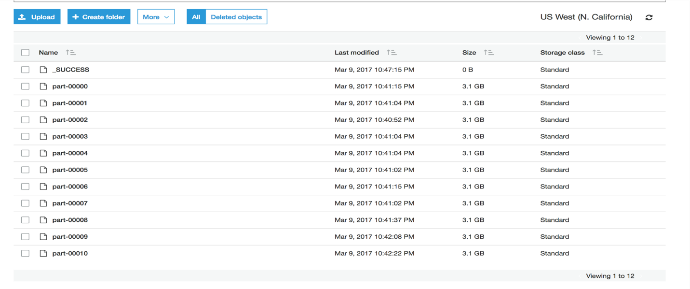


Figure 11 Results of map and reduce

Now we have to load data from s3 to RDS without storing it on the local machine. So, that we have to configure the Amazon – Relational Database Services with 23GB of storage space with the medium sized instance. The backup and recovery managed beautifully by the RDS services.

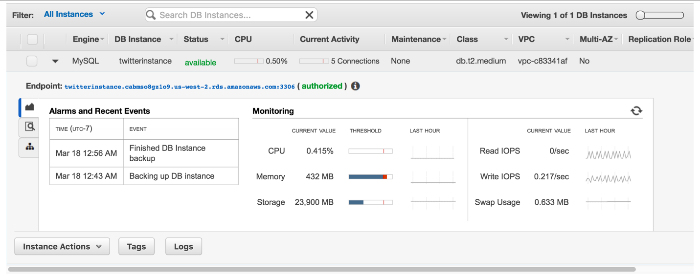


Figure 12 RDS configurations

We will use this command to load data directly from s3 to RDS.

LOAD DATA LOCAL INFILE ”s3://khushboobabucket/part-00000' INTO TABLE q FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'(@col1,@col2,@col3,@col4,@col5) set tweet\_id=@col1,user\_id=@col2,created\_at=@col3,score=@col4, text = @col5;

It will take 87 seconds to save 22 Million records approximately.

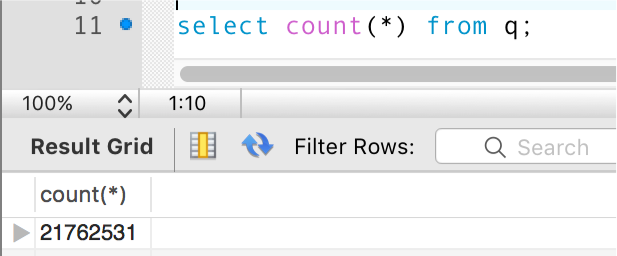


Figure 13 Millions of data on RDS

Step 6: RDS table and index creation with connection on MYSQL workbench.

We must give RDS endpoint URL, username and password to MySQL workbench. It will directly connect the RDS inside your local system.

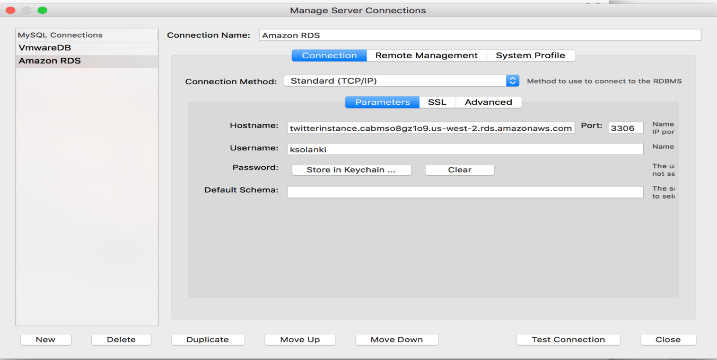


Figure 14 Connection with MySQL to RDS

Now we have created three tables for 3 web services. $0.108 \* 24 = $2.592 (This price is for two days because we need to check load balancing for 12 hours only with our custom inputs and a number of requests.)

Step 7: Creating the database and indexes on relational database services

Created 3 tables which will execute 3 queries. All tables have indexes which will help us to select data faster.

Q1: This Will check the health of entire web services.

Q2: This will return all the tweets on each 10 second. For example, if time now is 11:00 AM PT then it will return all the tweets of the 2014-2015 year which were tweeted at this time.

Q3: This will return all the information between given two user\_ids as a request.

Q4: This will return all the information about the users who has tweeted more than the given count in a year.

# create database

CREATE DATABASE tweets;

# create tables

# q2

CREATE TABLE q2(tweet\_id bigint primary key, text text, created\_at varchar(255));

# q3

CREATE TABLE q3(tweet\_id bigint primary key, user\_id bigint);

# q4

CREATE TABLE q3(tweet\_id bigint primary key, user\_id bigint, score int, text text);

CREATE UNIQUE INDEX q2index ON q2 (tweet\_id, created\_at);

CREATE INDEX q3index ON q3 (user\_id);

CREATE INDEX q4index ON q4 (user\_id, text);

Step 8: Connecting java code with EC2 instances using beanstalk.

I have specified 4 micro instances and deployed the application which is having 4 JSP which will perform 4 queries. Here we have used Amazon EC2 spot instances. Amazon EC2 Spot instances allow you to bid on spare Amazon EC2 computing capacity. Since Spot instances are often available at a discount compared to On-Demand pricing, you can significantly reduce the cost of running your applications, grow your applications’ compute capacity and throughput for the same budget, and enable new types of cloud computing applications. This will help us when there is a high time where you are receiving lots of requests that one instance can handle. This is important because we have 36GB of structural data in MySQL.

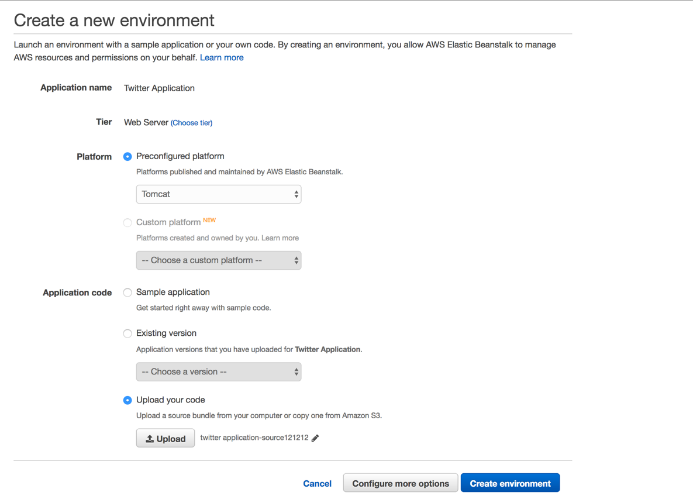


Figure 15 Beanstalk Configuration

Please find the code at my GitHub account:

https://github.com/khushbooba91/TwitterFinalProject

Step:9 Load balancing.

After deploying the application on the beanstalk, we must load balance the incoming requests. Elastic Load Balancing distributes incoming application traffic across multiple EC2 instances, in multiple Availability Zones. This increases the fault tolerance of your applications.

The load balancer serves as a single point of contact for clients, which increases the availability of your application. You can add and remove instances from your load balancer as your needs change, without disrupting the overall flow of requests to your application. Elastic Load Balancing scales your load balancer as traffic to your application changes over time and can scale to the vast majority of workloads automatically.

With load balancing as mentioned, we have ec2 spot instances. These both technique will be great for application health.

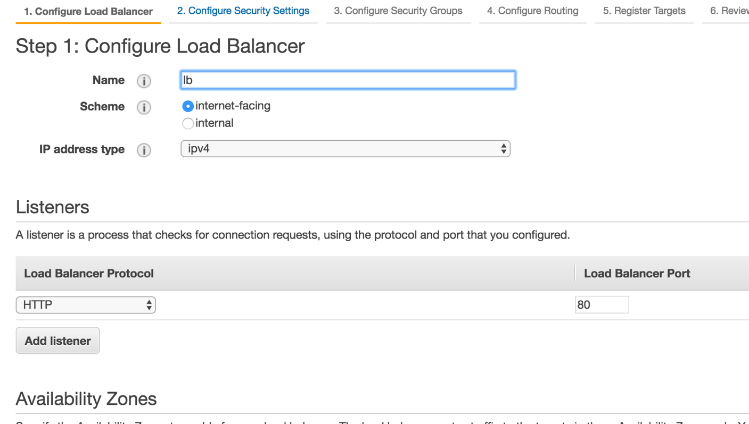


Figure 16 LOAD BALANCER CONFIGURATION

# Discussion and related work

Twitter analysis are done by many companies and organization in various ways. Each and every implementation of analysis is not cost effective. For analysis, companies are spending a lot of money. But, our project is cost effective as well as highly technical. On the other hand, projects implemented by other organization are expensive to the company. (i) We are building a web service to analyze twitter data which is cost-effective. (ii) Distributed NO-SQL database, HBase is used instead of a relational database. (iii) A number of systems used in our project are comparatively very less. (iv) Web services are REST based which is used to analyze the twitter data. (v) Spot instances are used to decrease the cost. Spot instances are much cheaper than EC2 instances. https://aws.amazon.com/ec2/spot/ (vi) Our project supports Batch Job processing. ftp://public.dhe.ibm.com/software/webservers/appserv/extend/c omputegrid/WSW14163-USEN-00\_Final\_Jul21\_11.pdf (vii) We consider the real-time scenario to win the contract of twitter data analysis by minimizing the cost. (viii) We work on huge twitter dataset (100 GB). (ix) Softwares used are mainly open source. (x) Testing is done at each stage to evaluate the cost of testing. https://d0.awsstatic.com/whitepapers/AWS\_Cloud\_Best\_Practi ces.pdf

# Conclusion and future work

As we have seen the traditional style twitter analytics costs a lot. On the other hand, if we would like to use other available tools available in the market it will not have custom results flexibility plus costs are high.

By, using this technique we can perform the same actions at $758.9668 + $20 approx. for spot instances = $778.966 a year. These techniques certainly save a lot of money and effort.

Cost calculations are:

EC2 – $5.3324 (EMR, Load balancing, beanstalk) a week => $277.2848 a year

• S3– less than 1 Tb, data are given by client

• RDS - $3.6342 a week => $188.97 a year

 • Elastic Beanstalk: $2.134 a week => $277.28 a year

• EMR: $15.432 a year (calculated above)

# Future work:

• The optimization process is never ending process. We can still reduce the costs using our own scripts and methods on load balancing and RDS performance.

• Custom load balancers and scripts– so that they can request resources with custom requests

• RDS supports scripts for recovery – implement custom recovery algorithm according to our use.

# References and Citations

[1] L.Jaba Sheela, “A Review of Sentiment Analysis in Twitter Data using Hadoop”.

[2] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau, “Sentiment Analysis of Twiter Data”, <http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf>

[3] Jodi Blomberg, “Twitter and Facebook Analysis: It’s Not Just for Marketing Anymore”, [http://support.sas.com/resources/papers/proceedings12/309- 2012.pdf](http://support.sas.com/resources/papers/proceedings12/309-%202012.pdf)

[4] Kaify Rais, “Twitter Analysis”, <http://www.slideshare.net/ajayohri/twitter-analysis-by-kaifyrais>

[5] Deepan Palguna , Vikas Joshi, Venkatesan Chakaravarthy, Ravi Kothari and L V Subramaniam, “Analysis of Sampling Algorithms for Twitter”, https://www.ijcai.org/Proceedings/15/Papers/141.pdf

[6] Shamanth Kumar, Fred Morstatter, and Huan, “Twitter Data Analytics”, <http://tweettracker.fulton.asu.edu/tda/>

[7] LEKHA R. NAIR, DR. SUJALA D. SHETTY, “STREAMING TWITTER DATA ANALYSIS USING SPARK FOR EFFECTIVE JOB SEARCH”, <http://www.jatit.org/volumes/Vol80No2/17Vol80No2.pdf>

[8] Nausheen Azam, Jahiruddin, Muhammad Abulaish, Nur Al-Hasan Haldar, “ Twitter Data Mining for Events Classification and Analysis”, <http://ieeexplore.ieee.org/document/7414678/>

[9] Michał Skuza, Andrzej Romanowski, “ Sentiment analysis of Twitter data within big data distributed environment for stock prediction”, http://ieeexplore.ieee.org/document/7321604/ [10]Ke Tao, Claudia Hauff, Geert-Jan Houben, Fabian Abel, Guido Wachsmuth, “ Facilitating Twitter data analytics: Platform, language, and functionality”, http://ieeexplore.ieee.org/document/7004259/?section=abstract

[11] HBase Snapshots <http://hbase.apache.org/0.94/book/ops.snapshots.html>

[12] Snapshot in distributed computing http://www.ics.uci.edu/~cs237/reading/files/An%20introductio n%20to%20snapshot%20algorithms%20in%20distributed%20 computing.pdf

[13] Spot instances <https://aws.amazon.com/ec2/spot/>

[14]Working of spot instances http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/usingspot-instances.html

[15]Compression in hbase http://hbase.apache.org/0.94/book/snappy.compression.html

[16]Snapshot algorithm http://lamport.azurewebsites.net/pubs/chandy.pdf