Kelvin Hernandez Cruz

COMP.5800

Big Data System Design

Project:

“Big Data Storage and Visualization”

Assignment due date:

11-Dec-2018

Table of Contents

[**Introduction:** 3](#_Toc532233423)

[**Motivation:** 3](#_Toc532233424)

[**Related Work:** 3](#_Toc532233425)

[**Proposed Approach:** 3](#_Toc532233426)

[**Experimental Results and Discussions:** 5](#_Toc532233427)

[**Conclusions:** 10](#_Toc532233428)

[**Statement Code:** 10](#_Toc532233429)

[**References:** 11](#_Toc532233430)

# **Introduction:**

The research is going to be based on how to store dataset and analyze it with the end goal to visualize the dataset over a well structure big data architecture using already existing services. The dataset selected for this project is National Footprint Accounts (NFAs) that I fount from “data.world”. Therefore, what is the NFAs? The NFAs measure the ecological resource use and resource capacity of nations. The calculations in the NFA are primarily based on United Nations data sets, including those published by the Food and Agriculture Organization, United Nations Commodity Trade Statistics Database, and the UN Statistics Division, as well as the International Energy Agency [1]. This contain/includes the total and per capita national biocapacity, the ecological footprint of consumption, the ecological footprint of production and total area in hectares. The dataset is going to use as a test and prove of concept that the architecture being implemented actually works. However, the end goal is to store and use this dataset to analyze and identify which country are using more ecological resources and services than they have, and those with more resources than they are using.

# **Motivation:**

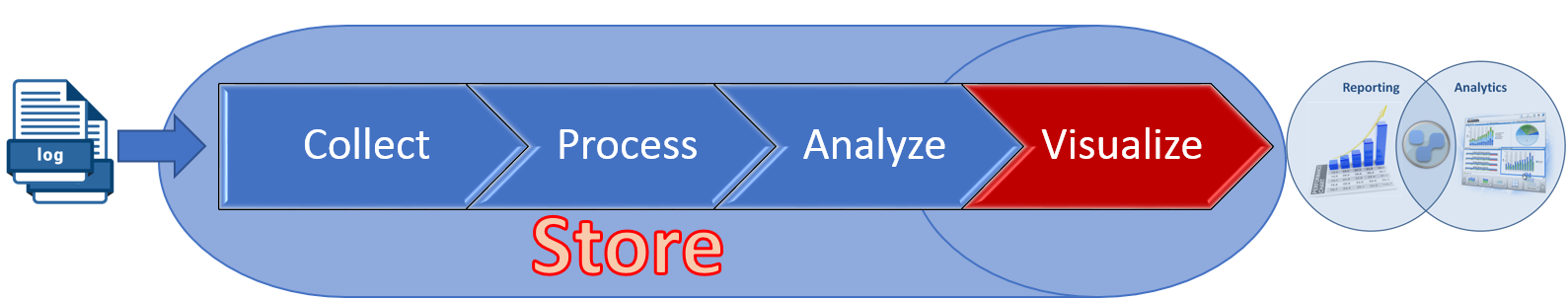
The motivation of this project is developing a big data system that can optimize and be reliable for collecting all the NFAs data, being able to store them, analyze them and generate accurate visual reports. Implementing this architecture will be quite interesting since it could beneficial useful to be able to produce valuable information base on what we had collected. The overview of this project is more like an End to End big data storage with data analytics solution.

# **Related Work:**

Yes, many research related to big data storage. However, almost all of them are more based on the best practical way of redundant, scalable, and performance of storing the data, but not quite much on how to analyze the data to generate reports.

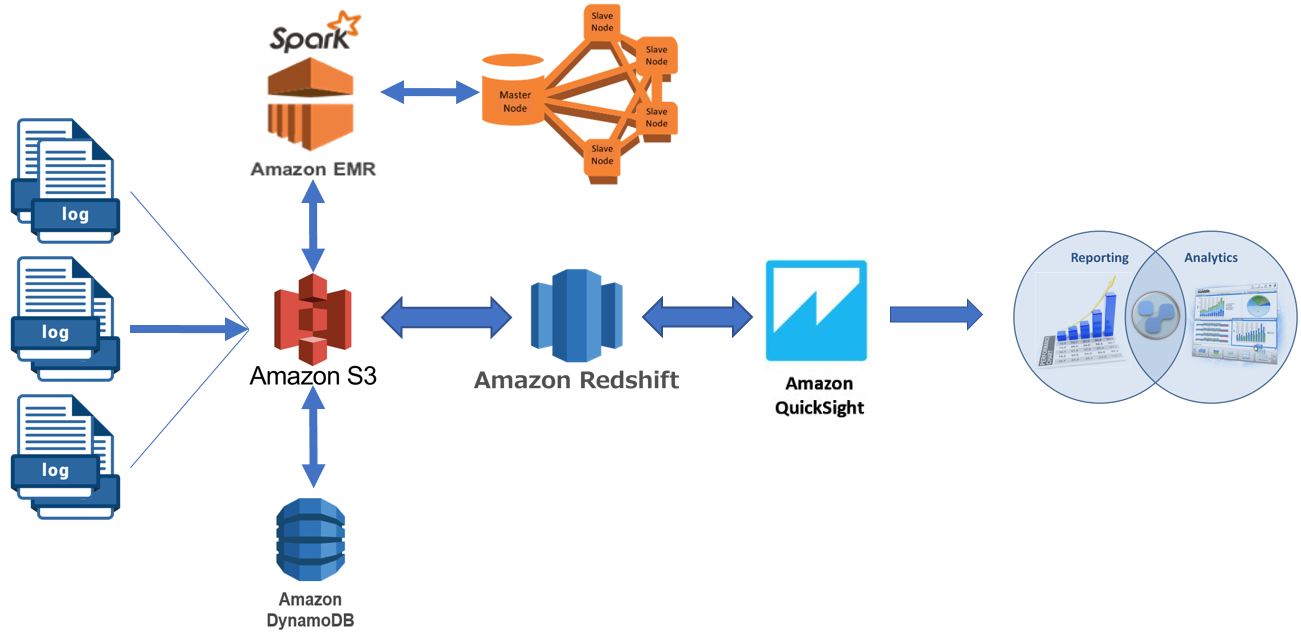
# **Proposed Approach:**

The basic idea is to have data coming in and get a sophisticate analytics out with the data processed. On Figure 1 is shown a primitive pattern of the implementation. The general step to be follow on this implementation are the following: first we basically know that we need to collect the datasets that we being provided. Second the data collected will need to be process it. Next, the data already processed will be analyze and will make sure that the data it is useful. Final step will be to read the data after being analyzed with the purpose of visualization of the data to actually get an easy end user usage of the inputted raw data, which is one of this project focus. However, during the process the ability of the services that are going to be test will show the ability of storing data on a big data scale. Making sure that the data can be store in a secure and reliable way. Having a reliable massive data storage will help us to go thru the rest of each step of the implementation. Since all the instance depend of the storage. Also, being able to manage the data after being store, and understand the data that we have. Which it is important to have a mechanics that can let us run sophisticated big data analytic on our data without having to move the data to a separate analytic platform.



**Figure 1**. Primitive Pattern of the Implementation.

After reading some research and documents, I came up with a intial high-level overview architecture which is shown in Figure 2. In this architecture it can be observe that we don’t have any implementation to stream real time data because that’s not what we are actually doing, we are just going store the dataset selected strait to S3 (Simple Storage Service). To process the data, we will be using Elastic MapReduce (EMR) clusters with at least two nodes (master and Slave). Using Redshift, we are going to run query and analytics against all the data the we have on S3. Also, we going to load data in parallel optimizing it for fast analytics queries. With only two nodes (master and Slave). After a successful implementation of the mentioned services finally going to analyze and visualize data with quick-sight.



**Figure 2**. Initial Proposed High-Level Overview of the System Design

# **Experimental Results and Discussions:**

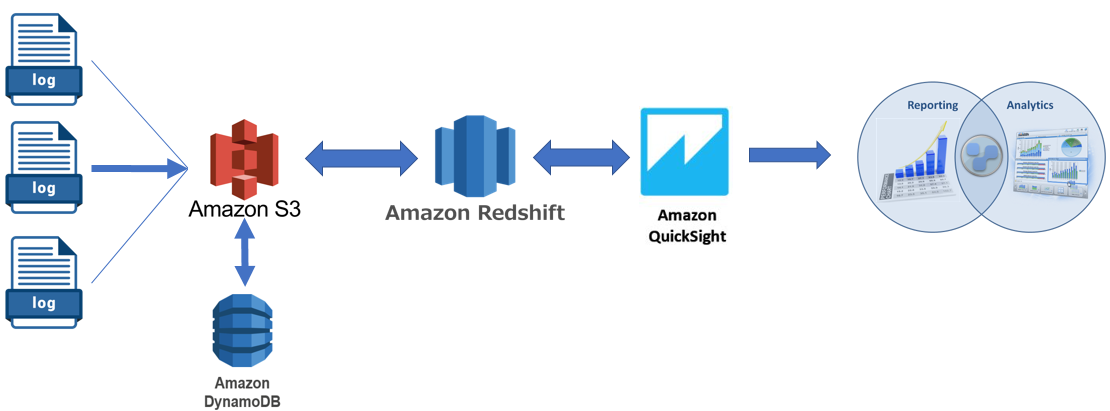
To begging with the implementation of this research and Since most of the work will be done using AWS (Amazon Web Services). I had to take in consideration that was extremely important to learn how to spin up and get know how each one of these services work, specially the part of each services that actually need to be use for this implementation.

The first thing that we need to know is to work on the first face, Figure 3), is to move all the datasets to S3 in the same CVS (comma-separated values) format. Setting up S3 for the first time was kind of straight forward, but it can be confusing if you don’t know what you need or want at the time to spin it up. However, once the bucket or buckets are created it is straight forward on how to load the data into the buckets and what kind of privilege to add to it. Step by step of how to set up S3 can be found in referent [2].



**Figure 3**. Data being loaded to S3 Bucket.

The next step from the initial architecture was to spin up EMR and set up the clusters to process the data from S3, however I modified the initial architecture after reading and finding out that Redshift for this project will satisfy the work of this implementation, because of this the initial architecture was modified taking out the EMR service like is show in Figure 4. EMR is use a larger scale that will need multiple cluster to prepare the data for Redshift or another service, this service could be use in another case where the cluster that Redshift offers are not enough and need a process power.



**Figure 4**. Modified High-Level Overview of the System Design

The next step of this implementation was to spin up the Amazon Redshift server. ‘Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. You can start with just a few hundred gigabytes of data and scale to a petabyte or more. This enables you to use your data to acquire new insights for your business and customers’ [3]. This service can be a little confusing at first since need to know how actually you want on your service, like the set of nodes or better call cluster. After spinning up the Redshift another important task is creating the privilege for Redshift to be able upload your data set and then perform data analysis queries from S3 (Figure 5). After creating the necessary rights, a database management system (DBMS), cross-platform SQL query tool needed to be setup to be able to execute command/statement on Redshift to create, retrieve, update and manage data. In this specific case I use Workbench/J on my local machine to interact with Redshift. Here is where the struggle began because the lack of knowledge with database.



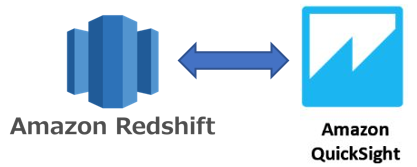
**Figure 5**. Querying data from S3 Bucket to Redshift.

After setting up the DBMS it took me many hours learning and writing the right table and how to retrieve the data from S3 on a correct way. The debugging of this step was exhausted since was my first time working with database. Figure 6 show a high-level view of the relation of the used SWL workbench with Redshift. The final statement code for this implementation can be found in section “Statement Code”.



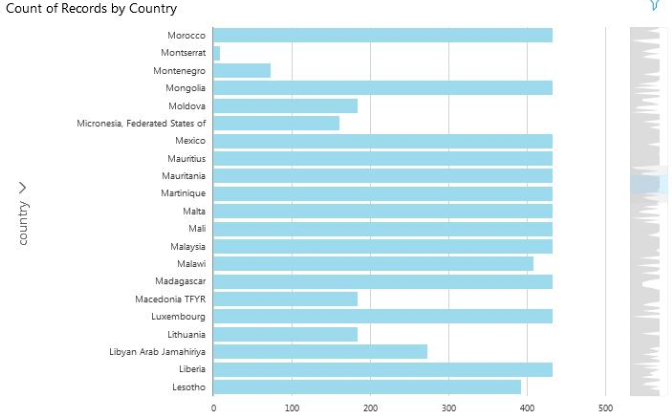
**Figure 6**. High-Level View of the Relation of SWL Workbench w/ Redshift

The last step of the implementation of this architecture is to spin up Quick-Sight connected to Redshift (Figure 7) to retrieve the data from it to generate useful visual reports. The process of setting this step up require of creating a new security group for that instance. This security group contains an inbound rule authorizing access from the appropriate IP address range for the Quick-Sight servers in that region. However, a step by step of enabling Quick Sight access can be found in reference [4]. This service can be a little confusing setting it up at first however noting outside of the box.

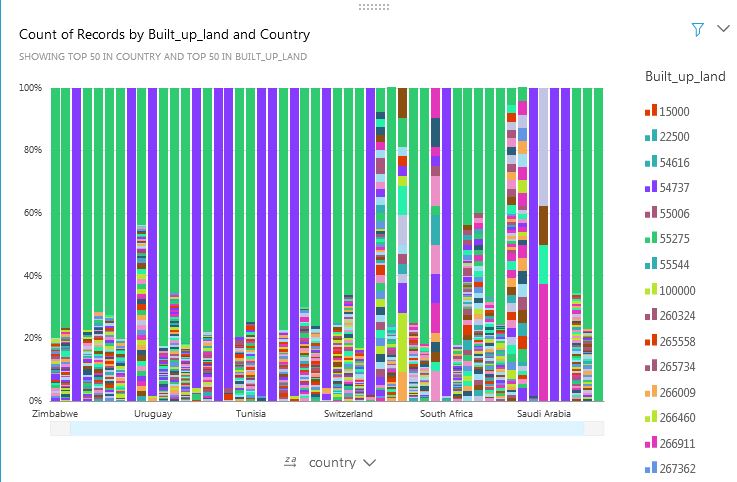


**Figure 7**. Quick-Sight Connected to Redshift

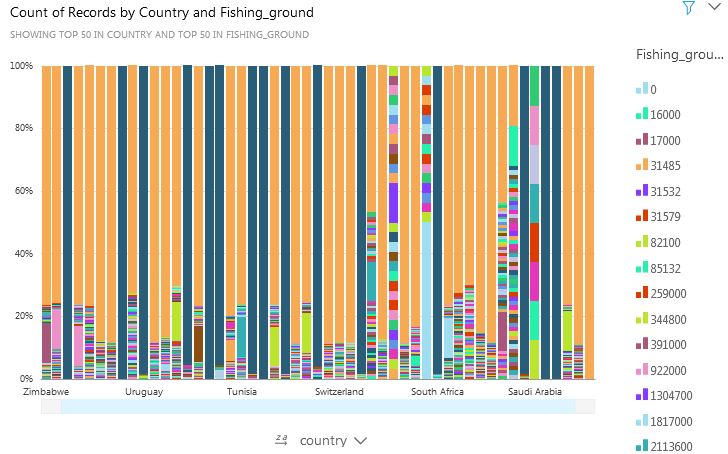
The results were expected, after getting access to the dataset in Redshift using Quick-Sight and getting the analytic of the data by this service are very impressive. Using this service, we can get visual of any kind regarding the dataset, the visual can be modify by any fields list or by applying a filter. For example, it can be simple as Figure 8 and customize as Figure 9 or 10.



**Figure 8**. Count of records by Country.



**Figure 9**. Count of Records by Built Up Land and Country.



**Figure 10**. Count of Records by Country and Fishing Ground.

I have to say that this project was really good since it kind of give me the opportunity to learn and understand more on how the different services and method currently out there that we can use to take care of big data. Since I didn't know much I took this chance to do this implementation using AWS, with the purpose to actually learn and see how other people/companies used it, which I spent many hours learning the essential of each service used in this project. Also learning the implementation part was great which created a picture in my head of how everything actually works.

Automate process data.

# **Conclusions:**

To conclude this project, Figure 8,9,10 proved that using this implementation can not only solve the problem of storing the datasets and analyze them to create useful reports but also reduce the amount of work, time and headache when it comes to scale. another good point here is that this kind of implementation can be adapt it to any target environmental. An improvement of this implementation will be to automate the data to be store and process it so the end user can have the latest reports. A lesson to learn here is that thing can change or take turn during the process and another solution can be found, so never can stick to the initial possible solution.

# **Statement Code:**

DROP TABLE IF EXISTS nfa\_csv\_2018 CASCADE;

CREATE TABLE nfa\_csv\_2018

(

   country         varchar(100),

   year            integer,

   country\_code    integer,

   record          varchar(100),

   crop\_land       varchar(100),

   grazing\_land    varchar(100),

   forest\_land     varchar(100),

   fishing\_ground  varchar(100),

   built\_up\_land   varchar(100),

   carbon          varchar(100),

   total           varchar(100),

   qscore          varchar(100)

);

copy nfa\_csv\_2018

from 's3://mydatabucketsrc/'

iam\_role 'arn:aws:iam::884652343598:role/myRedshiftRole'

region 'us-east-1'

IGNOREHEADER 1

Delimiter as ','

REMOVEQUOTES

select count(\*) from nfa\_csv\_2018;

select \* from stl\_load\_errors order by starttime desc

# **References:**

[1] <https://www.footprintnetwork.org/>

[2] <https://docs.aws.amazon.com/AmazonS3/latest/gsg/CreatingABucket.html>

[3] https://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html

[4] https://docs.aws.amazon.com/quicksight/latest/user/enabling-access-redshift.html