ISSN: 2586-7652 (Print) Vol. 03 No. 02, September 2020
ISSN: 2635-7607 (Online)
International Journal of Advanced Engineering
Source: http://ictaes.org
Manuscript received: July 20, 2020; Revised: August 23, 2020; Accepted: September 05, 2020

Big Data Pipeline for Vehicle GPS Data Analysis using Spark and Elasticsearch

Pradip Bhandari[1], Dipak Kumar Nidhi[2], Basanta Joshi[3], Surendra Shrestha[4]

Department of Electronics and Computer Engineering, Pulchowk Campus, Institute Of Engineering, Tribhuvan University, Pulchowk, Lalitpur, Nepal

075mscsk013.pradip@pcampus.edu.np[1], 075msice005.dipak@pcampus.edu.np[2], basanta@ioe.edu.np[3], surendra@ioe.edu.np[4]

Abstract

Today data structuring, processing and transforming data from one system to another regardless of source is a big need. Need of big data was felt almost when first GPS technology was introduced. GPS technology has evolved significantly since big data technology became widely used. Big Data has long been the subject of interest for Computer Science experts, and has gained popularity and even more being important in the recent days with the endless eruption of data resulting from social media, healthcare, satellite, and the quest for tech giants to gain access to deeper analysis of their data. Use of the internet and all the types of computer related systems generates tremendous amount of data in different forms. Due to significant large volume, high velocity, and different types of varieties of this type of data emerges the Big Data Problem. Spark and Elasticsearch programming frameworks provide an effective open source solution for managing and analyzing the Big Data. Today researchers are using both the frameworks and making many interpretations. In this paper, we are discussing the working model of spark and elasticsearch frameworks and by experimental analysis.

Keywords: *Big Data; Spark; elasticsearch; Kibana; GPS; HDFS; MapReduce.*

1. Introduction

Traditional Systems normally have a centralized server to store, process and manipulate data. Spark was initially introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process. Hadoop is basically utilized by Industries to investigate data sets. The explanation is that Hadoop relies on MapReduce, a straightforward programming model and it enables scalable, flexible, fault-tolerant and price effective computing solutions. As against a standard belief, Spark could be a standalone

Corresponding Author : Dipak Kumar Nidhi
Author's affiliation : Department of Electronics and Computer Engineering, IoE, TU, Nepal
Email: 075mscice005.dipak@pcampus.edu.np
Copyright © ICT-AES

system, not a modified version of Hadoop and isn't, really, keen about Hadoop because it's its own cluster management. Hadoop is simply one in every of the ways to implement Spark. Spark uses Hadoop in two ways – one is storage and second is processing. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only. Apache Spark is also a lightning-fast cluster computing technology, designed for fast computation. It supports Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more forms of computations, which has interactive queries and stream processing. The foremost feature of Spark is its in-memory cluster computing that increases the processing speed of an application. Big-data system faces a series of technical challenges, including: First, due to the shape of disparate data sources and the sheer volume, it is difficult to assemble and integrate data with scalability from distributed locations, as an example, more than 175 million tweets containing text, image, video, social relationships are generated by numerous accounts distributed globally [4]. RDBMS support services that have a schema. Unstructured or semi- structured data may have to be indexed. it's become important to form a brand new platform to fulfil the demand of organization because of the challenges faced by traditional data. Big data helps to analyse growing volumes of structured transaction data, plus other varieties of data that are often left untapped by conventional business intelligence (BI) and analytics programs. Elasticsearch may be a big data technology which is schema less. It's a tool accustomed to searching big data. Elasticsearch uses the concept of denormalization for search. Elasticsearch uses the indexing concept. it's a document oriented tool. Once the document is added, it is often searched within a next second as Elasticsearch is real time. The principal of Elasticsearch's intelligent search program is basically another software project called Lucene. It's possibly easiest to know elasticsearch as a component of infrastructure built using Lucene's Java libraries. In elasticsearch everything is expounded to the actual algorithms for matching text and storing optimized indexes of query terms is executed by Lucene. Elasticsearch itself provides a more functional and compact API, scalability, and operational tools overhead Lucene's search implementation. Lucene is old in internet years, dating back to 1999. It's also extremely widespread and established. Lucene is employed by inexpressible numbers of companies, running the scope from huge corporations like Twitter, to small startups. Lucene is demonstrated, tested, and is widely considered best-of-breed in open-source search software [11,16].

2. Basic Concept

Elasticsearch is a near real-time, open-source full-text search and analytics engine which is accessible from RESTful Web Service Interface. Elasticsearch, at its heart, may be a program. It's great at taking documents, understanding their language (word stemming, cleaning stop words, etc) and storing them during a way which can allow in no time fetching. But it'll be fast just for fetching the primary 100s-1000s of documents. When querying for 10000 results - Elasticsearch are going to be relatively slow. Elastic search is made for searching datasets, it has some visualization tools and some basic analysis functions There are a few concepts that come with Elasticsearch and their understanding is critical to fully understand how Elasticsearch works and operates [6,8].

2.1. Index

Indices, the foremost important unit of data in Elasticsearch, are logical partitions of documents and will be compared to a database within the planet of relational databases. it's accustomed store the documents and browse them from it. Elasticsearch uses Apache Lucene library to place in writing and browse the data from the index. Elasticsearch index could even be built of quite one Apache Lucene Index by using "Shards". you will be ready

to have as many indices defined in Elasticsearch as you'd like. These successively will hold documents that are unique to each index. Indices are identified by lowercase names that sit down with actions that are performed actions (such as searching and deleting) on the documents that are inside each index [20].

2.2. Document

Documents are JSON objects that are stored within an Elasticsearch index and are considered the base unit of storage. In the world of relational databases, documents can be compared to a row in table. Document is the main entity in the Elasticsearch world. At the end, all use cases of using Elasticsearch can be carried at a point where it is all about searching for documents and analyzing them. Document consists of fields, and each field identified by its name and can contain one or multiple values. Each document may have a different set of fields; there is no schema or imposed structure [20].

2.3. Types

Elasticsearch types were used within documents to subdivide similar types of data wherein each type represents a unique class of documents. Types consist of a name and a mapping and are used by adding the type field. This field can then be used for filtering when querying a specific type. An index could have had any number of types, and you can store documents belonging to these types in the same index [20].

2.4. Mapping

Like a schema in the world of relational databases, mapping defines the different types that reside within an index. It defines the fields for documents of a specific type — the data type (such as keyword and integer) and how the fields should be indexed and stored in Elasticsearch. This includes meta-fields, such as _index, _type, and _id. The input text is divided into tokens, which tokens should be filtered out, or what additional processing, such as removing HTML tags, is needed. This is where mapping comes into play; it holds all the information about the analysis chain [20].

2.5. Node

The single instance of the Elasticsearch server is called a node. A single node in Elasticsearch deployment can be sufficient for many simple use cases.Elasticsearch is designed to index and search our data, so the first type of node is the data node. Such nodes hold the data and search on them. The second type of node is the master node, a node that works as supervisor of the cluster controlling other nodes' work. The third node type is the Tribe node, which is new and was introduced in Elasticsearch. The tribe node can join multiple Clusters and thus act as a bridge between them, allowing us to execute almost all Elasticsearch functionalities on multiple clusters[20].

2.6. Cluster

Elasticsearch cluster is comprised of one or more Elasticsearch nodes. As with nodes, each cluster has a unique identifier that must be used by any node attempting to join the cluster. By default, the cluster name is "elasticsearch," but this name can (and should) be changed, of course. Be sure not to use the same name for one cluster in different environments, otherwise nodes might be grouped with the wrong cluster. One node in the cluster is the "master" node, which is in charge of cluster-wide management and configuration actions (such as

adding and removing nodes). This node is chosen automatically by the cluster, but it can be changed if it fails. As a cluster grows, it will reorganize itself to spread the data. Even though clusters are designed to host multiple nodes, you can assign only node to a cluster if it is so desired [20].

2.7. Shards

Put simply, shards are a single Lucene index. They are the building blocks of Elasticsearch and what facilitate its scalability. Index size is a common cause of Elasticsearch crashes. Since there is no limit to how many documents you can store on each index, an index may take up an amount of disk space that exceeds the limits of the hosting server. As soon as an index approaches this limit, indexing will begin to fail. One way to counter this problem is to split up indices horizontally into pieces called shards. This allows you to distribute operations across shards and nodes to improve performance. When you create an index, you can define how many shards you want. Each shard is an independent Lucene index that can be hosted anywhere in your cluster.

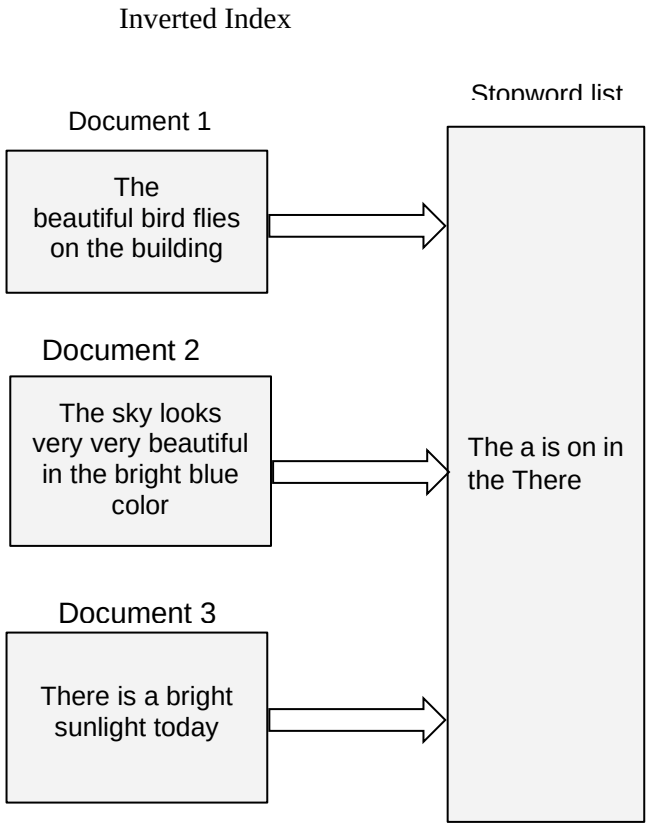
2.8. Replicas

Replica shard is the replica of primary Shard and is used to prevent data loss just in case of hardware failure. As the name suggests, they're Elasticsearch fail-safe mechanisms and are the same copies of your index's shards. This is often a useful backup system — or, in other words, when a node crashes. Replicas also serve read requests, so adding replicas can help to significantly increase search performance. An index can even be replicated zero (meaning no replicas) or more times. To confirm high availability, replicas aren't placed on the same node as the original shards (called the "primary" shard) from which they were replicated. For example, Shard 1 might share Node A with Replica Shard 2, and Node B hosts Shard 2 and Replica Shard 1. In a real use case, there'll be many more shards. Like with shards, the number of replicas can be defined per index when the index is originally created. Unlike shards, however, you may want to change the number of replicas anytime after the index is created. In other words, you'll be able to have over at least one replica for every shard, but there's a balance to be struck between having too few and too many shards. More shards lead to more overhead and resource usage, in addition as can affect performance and speed [20].

2.9. Analyzers

Analyzers are used during indexing to break down – or parse – phrases and expressions into their constituent terms. Defined within an index, an analyzer consists of a single tokenizer and any number of token filters. For example, a tokenizer could split a string into specifically defined terms when encountering a specific expression.

The most common analyzers include the Standard Analyzer and the Simple Analyzer, as well as several language-specific analyzers. By default, Elasticsearch will apply the Standard Analyzer, which contains a grammar-based tokenizer that removes common English words and applies additional filters. Elasticsearch comes bundled with a series of built-in tokenizers as well, and you can also use a custom tokenizer.A token filter is used to filter or modify some tokens. For example, a ASCII folding filter will convert characters like ê, é, è to e [5,20].



ID	Term	Document
1	beautiful	Document1, Document2
2	bird	Document1
3	blue	Document2
4	bright	Document2, Document3
5	**building**	**Document1**
6	color	Document2
7	**flies**	**Document1**
8	looks	Document2
9	sky	Document2
10	sunlight	Document3
11	today	Document3

Figure 1. Working of Elasticsearch with Stopword List concept

3. Big Data and Big Data Analytics

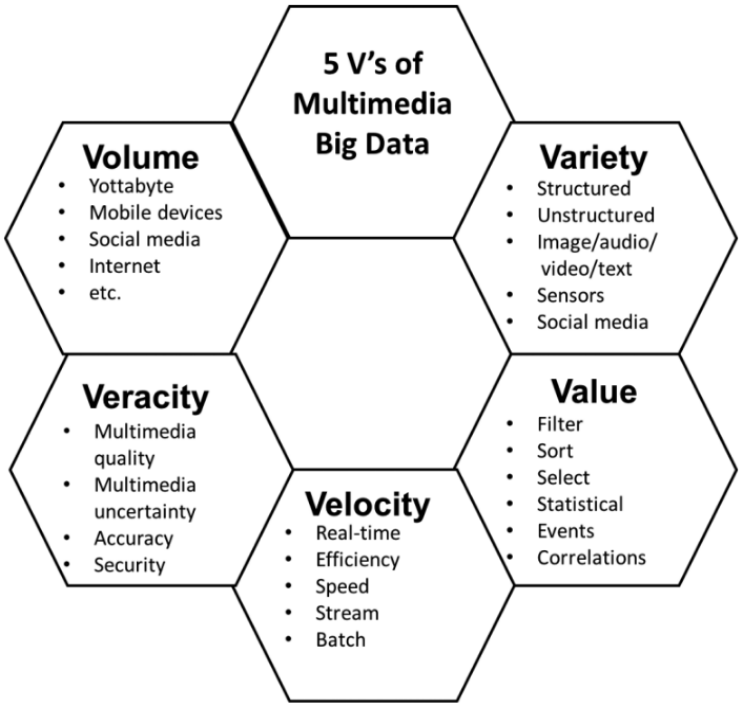
There are endless ever-growing supply of both structured and unstructured information lying in the World Wide Web and this information could have answers to a substantial measure of inquiries, which haven't been encircled within the bothers of business pioneers. On the opposite hand these organizations don't work out the way to utilize Big Data to the simplest point of their interest, the open source technologies like Hadoop and Spark may be accustomed finding knowledge for the organizations. Organizations must discover powerful approaches to seek out knowledge from the massive amount of knowledge being created each moment by making sense of what's in it and what to do with it. And there are a few advancements in technologies which can increase the organizations making benefit out of Big Data [9,10,12,13].

These Big amounts of data allude to five dimensions that are volume, variety, velocity, veracity and value. Volume: Enterprises are overwhelmed with regularly developing information of different kinds, effortlessly accumulating Exabyte—even Zettabytes—of data [1,2,3,4]. According to [4] 12 terabytes of Tweets made every day into improved product sentiment analysis, Convert 350 billion annual meter readings to better predict power consumption.Velocity:Data is increasing at unparalleled speed. We should think of what will happen in the coming years. It is a challenging task. Variety:As the increasing in technology, the data that comes today is in all types of formats. For example, audio, video, picture, numeric and structured data in traditional databases. It is also becoming problem for the organizations nowadays. Veracity: As per the different variety of data business analyst can trust on the information and knowledge generated. Because sometimes this types of data are not authentic as well as sometime this types of data are sponsored or spam. As

per [4] 1 in 3 business leaders don't trust on the information or knowledge generated by computer automated systems for making decisions. Value: As value dimension researchers try to identify importance and usability of this huge amount of Big Data for enhancing business and living standards. Big amount of biometric data also follow these five dimensions, that is why we can solve storing, managing, and retrieving issues using Big Data problem.Big data analytics, it is the process in which big data is examined that contains various kinds of data of various data types. The analysts find the hidden pattern in the big data and research the solution.The example of big data analytics done can be through text analysis, data mining etc.The steps involved in Big Data Analysis are: 1) Data Acquisition and Gathering 2) Data Cleaning and Extraction 3) Data Aggregation and Integration 4) Analysis and Modelling 5) Interpretation [15,17].

Figure 2. Five V's of Big Data

4. Hadoop Distributed File System – HDFS



The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is extremely fault-tolerant and is intended to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a some POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject. HDFS supports the rapid transfer of data between compute nodes. At its outset, it was closely coupled with MapReduce, a programmatic framework for data processing [9,14].

When HDFS takes in data, it breaks the information down into separate blocks and distributes them to different nodes in a cluster, thus enabling highly efficient parallel processing. Moreover, the Hadoop Distributed File System is specially designed to be highly fault-tolerant. The file system replicates, or copies, each piece of data multiple times and distributes the copies to individual nodes, placing at least one copy on a different server rack than the others. As a result, the data on nodes that crash can be found elsewhere within a cluster. This

ensures that processing can continue while data is recovered. HDFS uses master/slave architecture. In its initial incarnation, each Hadoop cluster consisted of a single NameNode that managed file system operations and supporting DataNodes that managed data storage on individual compute nodes. The HDFS elements combine to support applications with large data sets.The Hadoop Distributed File System arose at Yahoo as a part of that company's ad serving and search engine requirements. Like other web-oriented companies, Yahoo found itself juggling a variety of applications that were accessed by a growing numbers of users, who were creating more and more data. Facebook, eBay, LinkedIn and Twitter are among the web companies that used HDFS to underpin big data analytics to handle these same requirements [17,18].

5. Apache Spark

Apache Spark is a powerful unified analytics engine for large-scale distributed data processing and machine learning. On top of the Spark core data processing engine are libraries for SQL, machine learning, graph computation, and stream processing. These libraries can be used together in many stages in modern data pipelines and allow for code reuse across batch, interactive, and streaming applications. Spark is helpful for ETL processing, analytics and machine learning workloads, and for batch and interactive processing of SQL queries, machines learning inferences, and artificial intelligence applications. Application developers and data scientists incorporate Spark into their applications to rapidly query, analyze, and transform data at scale. Tasks most often related to Spark include ETL and SQL batch jobs across large data sets, processing of streaming data from sensors, IoT, or financial systems, and machine learning tasks [6,7].

Spark is capable of handling several petabytes of data at a time, distributed across a cluster of thousands of cooperating physical or virtual servers. It has an extensive set of developer libraries and APIs and supports languages such as Java, Python, R, and Scala; its flexibility makes it well-suited for a range of use cases. Spark is often used with distributed data stores such as MapR XD, Hadoop’s HDFS, and Amazon’s S3, with popular NoSQL databases such as MapR Database, Apache HBase, Apache Cassandra, and MongoDB, and with distributed messaging stores such as MapR Event Store and Apache Kafka. A wide range of technology vendors have been quick to support Spark, recognizing the opportunity to extend their existing big data products into areas where Spark delivers real value, such as interactive querying and machine learning. Well-known companies such as IBM and Huawei have invested significant sums in the technology, and a growing number of startups are building businesses that depend in whole or in part upon Spark.

Out of the box, Spark can run in a standalone cluster mode that simply requires the Apache Spark framework and a JVM on each machine in your cluster. However, it’s more likely you’ll want to take advantage of a resource or cluster management system to take care of allocating workers on demand for you. In the enterprise, this will normally mean running on Hadoop YARN (this is how the Cloudera and Hortonworks distributions run Spark jobs), but Apache Spark can also run on Apache Mesos, while work is progressing on adding native support for Kubernetes.

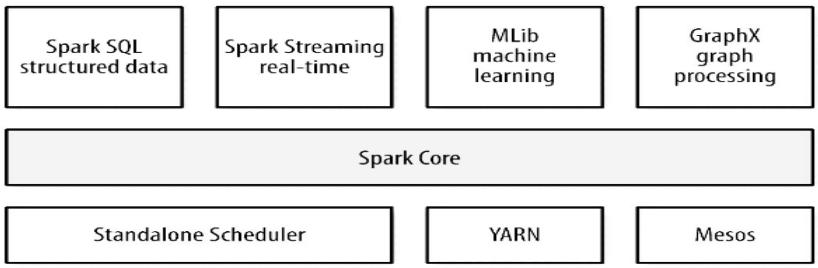


Figure 3. Spark Core

Unlike Hadoop spark maintains the intermediate results in memory rather than writing every intermediate output to disk. This hugely cuts down the execution time of the operation, resulting in faster execution of task, as more as 100X time a standard MapReduce job. Apache Spark can also hold data onto the disk. When data crosses the threshold of the memory storage it is spilled to the disk. This way spark acts as an extension of MapReduce. Spark doesn't execute the tasks immediately but maintains a chain of operations as meta-data of the job called DAG. The action on the DAG happens only when an action operation is called on to the transformation DAG. This process is called as lazy evaluation. This allows optimized execution of the queries on Big Data.

6. Spark RDD

At the heart of Apache Spark is the concept of the Resilient Distributed Dataset (RDD), a programming abstraction that represents an immutable collection of objects that can be split across a computing cluster. Operations on the RDDs can also be split across the cluster and executed in a parallel batch process, leading to fast and scalable parallel processing. RDDs can be created from simple text files, SQL databases, NoSQL stores (such as Cassandra and MongoDB), Amazon S3 buckets, and much more besides. Much of the Spark Core API is built on this RDD concept, enabling traditional map and reduce functionality, but also providing built-in support for joining data sets, filtering, sampling, and aggregation. Spark runs in a distributed fashion by combining a driver core process that splits a Spark application into tasks and distributes them among many executor processes that do the work. These executors can be scaled up and down as required for the application's needs.

There are two ways to create RDDs − parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.Spark Transformation is a function that produces new RDD from the existing RDDs. It takes RDD as input and produces one or more RDD as output. Each time it creates new RDD when we apply any transformation. Thus, the so input RDDs, cannot be changed since RDD are immutable in nature. Transformations create RDDs from each other, but when we want to work with the actual dataset, at that point action is performed. When the action is triggered after the result, new RDD is not formed like transformation. Thus, Actions are Spark RDD operations that give non-RDD values. The values of action are stored to drivers or to the external storage system. It brings laziness of RDD into motion.

7. Implementation and Experimentation

For implementation, we have used the cloud computing from amazon aws services. We have typically implemented Apache Spark on Amazon EMR (Spark 2.4.5 on hadoop 2.8.5 YARN). We uploaded 14.6GB of

raw traffic data to amazon s3. Configured and created our cluster by specifying data inputs, outputs, cluster size, security settings, etc.

We used two instances both are m3.xlarge(RAM:15GB, vCPUs 4), one master node and another core. To process 14.6 GB data it took almost 1 hour and 5 minutes. We converted the raw unstructured data to sql and index it to elasticsearch index. We used elasticsearch 5.4.3 with JAVA api client. Configured and created our cluster by specifying data inputs, outputs, cluster size, security settings, etc. After converting raw csv files to sql format we imported it to mysql database. We have total: 45,398,899(40.5 Millions rows). Counting total rows took almost 10sec in mysql. Same thing we query from elasticsearch using Kibana sense. It took just 50 milliseconds as compared to 9.65 seconds in standard mysql.

Also we tried plotting data(geo location) to map using Kibana visualize tool (Map) and google map for Kathmandu valley.

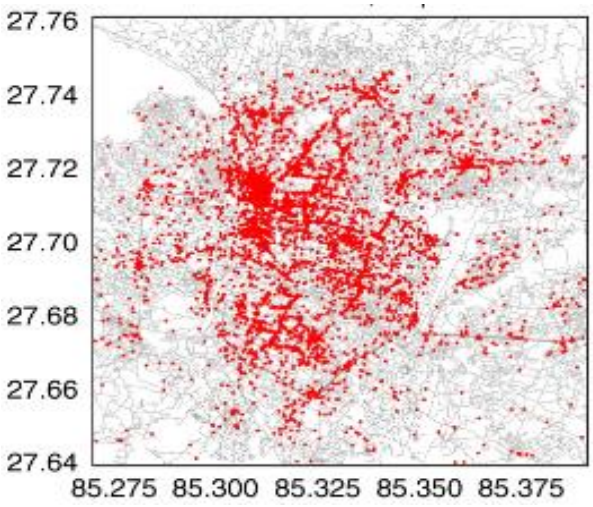


Figure 4. Vehicles geo location

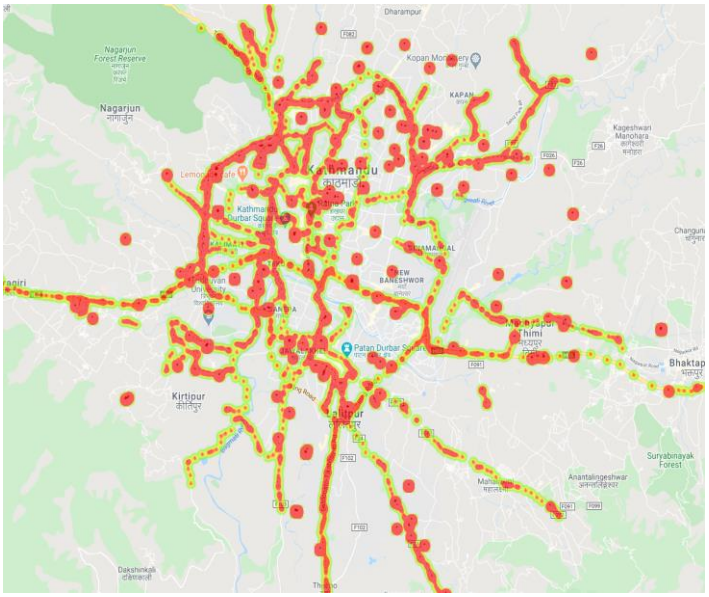


Figure 5. Heatmap of kathmandu city

8. Conclusion

Elasticsearch provides a proper system to store, pre-index, search and query very large scale data in near real-time. In particular, the capability of expanding the cluster size without stopping service as per user's requirement makes it suitable for this application. This research provides insights on how to process data using spark and standardize and configure the processes of Elasticsearch which result in increased analysis efficiency. To show the functionality and interactivity for normal users, the Kibana plugin was used as a user interface. In conclusion, a proper configuration of Elasticsearch and Kibana makes analysis of large scale data efficient and can help policy makers see the results instantaneously and in an accessible format that permits for decision making.

Several aspects are worth further investigation. Firstly, spark framework cannot analyze real-time data, a real-time data streaming tool, e.g., Apache Spark Streaming, can be introduced into the framework for near real-time analysis and tracking. Secondly, the network transportation cost within the cloud-based framework should be experimented for the data process when both the Elasticsearch and Spark cluster are integrated as an entire. Thirdly, although the framework reported within the gps data focused on large data, one can accelerate the analysis process with a customized partitioner for other kinds of data, e.g., geo-tagged. We'll continue this research to boost the framework with more specific performance-improving techniques.

References

[1] Alex Holms, "Hadoop in Practice", 2012, Manning Publications co.

[2] Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman, "Mahout in Action", 2012, Manning Publications co.

[3] Apache Hadoop. What Is Apache Hadoop?, 2014. http://hadoop.apache.org/, accessed April 2014.

[4] H S. Bhosale1, Prof. D. P. Gadekar2, A Review Paper on Big Data and Hadoop, International Journal of Scientific and Research Publications, 4(10),2014.

[5] Shvachko K., Hairong Kuang, Radia S, Chansler, R The Hadoop Distributed File System Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium.

[6] Spark Job Flow – Databricks https://databrickstraining.s3.amazonaws.com/slides/advanced-sparktraining.pdf

[7] Machine learning with Spark - Spark Summit 2013 https://spark-summit.org/2013/exercises/machinelearning-with-spark.html

[8] M. Dhavapriya, N. Yasodha, Big Data Analytics: Challenges and Solutions Using Hadoop, Map Reduce and Big Table, International Journal of Computer Science Trends and Technology (IJCST) – Volume 4 Issue 1, Jan - Feb 2016.

[9] Ambika P R, Dr. K.N. Narasimha Murthy, Sowmya Naik PT, Aparna J S, Big Data: Towards Next Generation Analytics, International Journal of Innovative Research in Computer and Communication Engineering. Vol.3, Special Issue 5, May 2015.

[10] Reynold Xin, Joshua Rosen, Matei, Zaharia, Michael J. Franklin, Scott Shenker, Ion Stoica. Shark: SQL and Rich Analytics at Scale. SIGMOD 2013. June 2013.

[11] Sara Landset, Taghi M. Khoshgoftaar, Aaron N. Richter* and Tawfiq Hasanin, "A survey of open source tools for machine learning with big data in the Hadoop ecosystem", Landset et al. Journal of Big Data (2015) 2:24 DOI 10.1186/s40537-015-0032-1.

[12] Satish Gopalani and Rohan Arora, " Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means", International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 1, March 2015.

[13] D. M. Blei and J. D. Lafferty, (2006), "Correlated Topic Models," in Advances in Neural Information Processing Systems 18 (Y.Weiss, B. Scholkopf and J. Platt, eds.), MIT Press, Cambridge, MA.

[14] N. K. Nagwani, (2015), "Summarizing Large Text Collection using Topic Modeling and Clustering based on MapReduce Framework," in Journal of Big Data 2, 6.

[15] X. Xu, A. Shimada and R.Taniguchi, (2013), "Correlated Topic Model for Image Annotation," in The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision.

[16] D. Newman, A. Asuncion, P. Smyth and M. Welling, (2008), "Distributed Inference for LatentDirichlet Allocation," J. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), Advances in Neural Information Processing Systems 20, MIT Press, Cambridge, MA, pp. 1081 – 108.

[17] K. Zhai, J.L. Boyd- Graber, N. Asadi and M.L. Alkhouja, (2012), "Mr. LDA: A Flexible Large -ScaleTopic Modeling Package using Variational Inference in MapReduce," in Proceedings of the 21st Int. Conf. on World Wide Web, pp. 879 – 888.

[18] N. Sawant and H. (Software engineer) Shah, Big data application architecture Q & A a problem-solution approach. Apress, 2013.https://doi.org/10.1007/978-1-4302-6293-0.

[19] Vora, Mehul Nalin. "Hadoop-HBase for large-scale data."In Proceedings of 2011 International Conference on Computer Science and Network Technology, vol. 1, pp. 601-605. IEEE, 2011.

[20] Dixit, B. Elasticsearch Essentials; Packt Publishing Ltd.: Birmingham, UK, 2016. [Google Scholar].