Container Based Processing of Large Scale Geospatial Data

Tejaswini B N
PG Student, M.Tech (CSE)
City Engineering College, Bangalore, India
tejaswini.begur@gmail.com

Ambika P R
Assistant Professor, Dept of CSE
City Engineering College, Bangalore, India
ambikatanaji@gmail.com

***Abstract*— Processing of geospatial data is compute intensive. If this processing is done on a large scale which has huge data sets like OpenStreetMap then the process of running a filter and the process of aggregation operations on huge records of data can be benefited by performing parallel processing on massive parallel processing platforms. The focus of the project is to describe and build a distributed geospatial processing framework based on one of the evaluated open source stream and batch processing engines like Apache Hadoop, Spark and Pachyderm to perform Spatial ETL functionalities which Extract, Transform and Load like extraction of data from different data sources, transformation of data to correct errors, cleanse, fuse, make them compliant to defined standards and load the transformed data into target spatial database (postgresql) or GIS file or Geospatial web services.**

Keywords— *Geospatial Data, OpenStreetMap, Pachyderm, Java.*

I. INTRODUCTION

It is highly challenging to compute geospatial in large scale these days as the data is huge. Hence processing has to be done in parallel to achieve higher computation if done on large scale records like OpenStreetMaps. Gif file format is a type of file format that is used to encode geographical information and data in our computer file system. It can be represented in the form Rastor data format or Vector data format. Raster data is a digital image which is usually represented in the form of grids that can be enlarged as well as reduced. Raster data is stored in the form of JPEG, BLOB, TIFF images. Vector data is a type of data which is represented in the form of geometrical shapes such as points, lines or polylines and polygons. These vector data can be stored in the form of spatialite, GeoJSON, shapefile, TIGER etc. The map data which we obtain after processing large scale geospatial data will then be stored in the cloud as the benefits are many. Cloud can be defined as a huge network of multiple networks that provide remote access to group of decentralized resources. As we know that there are three types of cloud services namely, IaaS, PaaS and SaaS we will be using another advanced version of PaaS which is CaaS. CaaS is Container as a Service provides a very good method to do orchestration of containers. Docker containers are used in our system which are built on Pachyderm data processing systems. Kubernetes is the CaaS paltform which provides orchestration of containers in processing large scale geospatial data.

Earlier there was an option only to demonstrate the architectural design of container based big data processing system. The automated provisioning of two popular big data processing system like Hadoop and Pachyderm was illustrated. This simulation is based on single machine consisting of Docker swarm, VirtualBox and MacOS X and over simplifies the running of containerized systems. At large scale it cannot be done by just selecting the appropriate driver names. It requires choosing the right cloud compute node types like AWS EC2 will optimize to handle CPU, memory and I/O intensive workloads. It also will need configuration management to automate the provisioning and upgrading of the cluster. Tuning the system to domain specific workload is required. We need to seamlessly scale the system to crunch the large data workload.

The sizing and capacity management issues like how many containers to run on a machine and how to allocate resources to those containers are a primary concern. The role and responsibility of a map is to describe and represent spatial relationships of certain features like roadways, water ways etc. There are many kinds of maps that make an attempt to represent specific features. Maps could represent and show political boundaries of different countries, population of each area or a country, physical features like roads, highways, waterways, natural resources, climates, elevation, social relationships, weather etc.

The manuscript explores about the concept of using containers for processing geospatial data which is generally always in a large scale along with discussing about the limitations present in the previous discussions, problems during the research, the procedures used in the research paper, methodologies adopted, etc. The concepts explored in section 2 are related to the approaches associated in the previous researches which is followed by the current technologies used. Section 3 explains about the methodologies used in building the pipeline of containers using Pachyderm processing framework along with the support of Kubernetes. Finally, the section 4 confers about the results and the assumptions associated with it.

II. LITERATURE SURVEY

The design and adoption of multi cloud infrastructure for the design of complex and distributed software systems are embraced in the software industry always. This new multi cloud infrastructure makes it possible to mix and match platforms for various software development activities and phases. Docker has introduced container based software development which overcomes the limitations of multi cloud infrastructure such as complexity due to different technologies. Docker has also invented distributed system development tool called Swarm which extends the Docker container based software development process [1].

Docker Swarm has even also introduced many of the dependability attributes to support the development of a multi-cloud dependable system. But Swarm cluster to be mande always available needs minimum of three nodes that are active managers which can protect from failure. This situation for the dependability is one of the main limitations because if two manager nodes fail suddenly due to the failure of their hosts, then Swarm cluster cannot be made available for routine operations. This paper presents an approach that overcomes this limitation by detecting and predicting the possible failures earlier [2].

Pachyderm enables efficient and sustainable data science workflow. It also maintains the scalability and reproducibility in parallel. Pachyderm stack uses Docker containers as well as Core OS and Kubernetes for cluster management. Te replaces HDFS with its Pachyderm file system and MapReduce with Pachyderm pipelines. We have an reasonable and graceful style for making systems of systems (SoS) grounded on Google cloud organization. This system in-turn can be secondhanded by any one and anywhere[3]. The key module of cloud computing is virtualization which can also be labelled as the core of cloud computing. The imitation is grounded on Docker Swarm, virtual container and other things [5].

Modern applications usually are accumulated from existing apparatuses and these mechanisms depend on other services and applications. For example, your Java application program might use PostgreSQL as a data stock, Redis for hoarding and Apache as per a Web server. Each and every component like this comes with its own set of addictions that may cause a conflict with other apparatuses. By wrapping each component and its dependencies, Docker resolves this limitation.

III. METHODOLOGY

The distributed framework which is distributed and highly scalable has the following components:

3.1. Spatial Data Source: They are the data sources related to geographical location which is usually stored as coordinates and topology in RDMS database. There are several dissimilarities of foundations of geospatial data. Some of them are normal earth data, exposed topology, NASA earth remarks etc Google map also offers map information which can be cast-off but it is proprietary. [6]

3.2. Vector data: Vector data is built using geometrical shapes such as polygons, lines and points. Discrete data and other types of data such as categorical data such as forests, lakes, oceans can be represented in a very good manner by this vector data. A feature is something that can be seen on the landscape. GIS environment is one type of environment which consists of features that are of real world and this environment can be represented using vector data. All these features can be depicted in the GIS environment. The features of vector data have attributes and these attributes contain numerical information or textual information and these attributes represent the vector data.

3.3. Graph data: Graph data is in the form of maps which consists of roads and landmarks. Roads will be depicted as edges in the graph data and landmarks are depicted in the form of as nodes. Landmarks can also be known as intersections in case of graph data representation.

3.4. Ingest: A scheduler service that periodically triggers task to pull the compressed vector and graph data from external data source, validate and publish the data to processing component.

3.5. Process: A core component responsible for partitioning the incoming workload into smaller chunks, distribute the work across multiple nodes for parallel processing and store the transformed data in spatial database. The process component will be based on one of the open source big data processing engine - Apache Hadoop, Spark, Flink or Pachyderm.

3.6. Data Storage: A spatial relational database management system (SRDMS) that stores the transformed spatial data in the form of Vector tiles. Object storage like AWS S3 id used for this purpose.

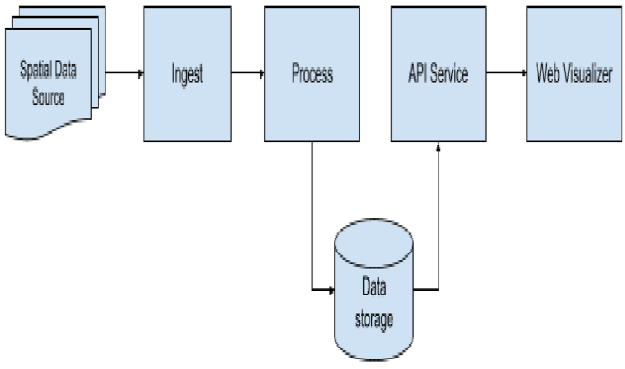

Fig 1. Distributed framework for Geospatial data processing

3.7. API Service: The Rest Application Programming Interface, API that will have to query the spatial data stored in the SRDMS and make it directly available on the web using OGC Web Map Service (WMS) Interface Standard.

3.8. Web Visualizer: A web application to browse through the information in the displayed part of the map aka vector tiles fetched from the API server. The visualize libraries understands GeoJSON format and hence we convert he osm location reference data attributes from JSON file to GeoJSON file. Now that we get the GeoJSON file it needs to be stored in a cloud for which we will be using Amazon Web Services EC2 instances. The basic building of this processed geospatial data in done through containers for which we will be using Docker container maintained in Kubernetes orchestration of containers.

The Pachyderm is that framework system which is open source to work on the flow and does data management along with it. This is accomplished by creating pipelines of data and acts as a layer of versioning of data ahead of projects from a big system which consists of containers.

Kubernetes acts as a backbone for all this by providing orchestration of these containers through the pachyderm pipelines. The following diagram shows the architecture of Kubernetes. Kubernetes is a cloud computing service system which is open source. It is that system which manages applications that are containerized across multiple numbers of hosts. It provides necessary mechanisms such that for an application to deploy manage and scale. The highly incorporated Kubernetes architecture is built within PaaS cloud computing service. This is called CaaS cloud computing service. The users can use the container services even if the required infrastructure is not present. CaaS encapsulates each and every application from the underlying infrastructure and hence enables the operation to perform on any platform which supports containers and its technologies. A free programming platform is provided by CaaS and hence software developers need not depend on the programming languages provided by the vendors. CaaS is used to build PaaS.

Following is a brief description of the components of CaaS:

- Physical host or VM: It is a PaaS server which in our case is Amazon Web Service EC2 instances.
- Container networking: It is the networking phase for the containers to communicate in an asynchronous manner. Each container has its own IP address and we can ping them to communicate.
- Container scheduling: It does the scheduling part for the containers to provide the necessary resources to the pachyderm. Through the API services, the scheduler makes container A to communicate to container B Pachyderm language is converted in Kubernetes language prior to this.
- Service management, container cluster management also does the necessary part of the CaaS services.

.

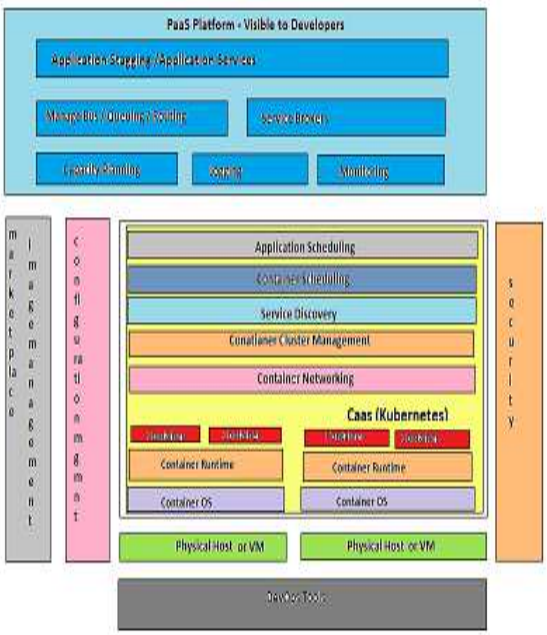

Fig2. Cloud service provider

For an application we need login, application life cycle management, monitoring, security and other necessary services for it to work on a whole which are provided by the other components of PaaS cloud service provider. In our project the class roles or participants are map data, osm2osmlr converter, json2geojson converter and geojson required output. The map data we consider here is the OSM data for Bangalore region. The protobuf file extracted from the map data of the particular region is converted into JSON file which has all the attributes of the road map of that particular region. This is represented in the form of graphhopper data. It has attributes such as number of nodes, points and links.

The map data is fed as input which represents the geospatial representation of a particular area required. We consider only road map in our project model. Osm2osmlr converter is another class role which converts the protobuf file of Bangalore region into our desired JSON file. It consists of all the necessary attributes of the road of Bangalore region. JSON2GeoJSON is the next class role which converts our JSON file to GeoJSON file which is the required format for the vizualizer to access. GeoJSON required output is the next role through which we can see the attributes of the location reference of Bangalore region.

The road map of a particular area is extracted to form a protobuf file. The proto file is extracted, massaged and transformed to form a JSON file which consists of the necessary attributes of the road map data. The JSON file in turn is fed as input to convert in into GeoJSON file. This can be viewed in a vizualizer. The vizualizer used here is leaflet or mapshaper.

The flow chart of the proposed system as shown in fig 3 works as follows:

- Kubernetes is CaaS cloud computing service which is incorporated in a PaaS service. We start the kubernetes which prepares the Kubernetes virtual environment to build up.
- The next step after this is to start pachyderm environment. The containers are built in CaaS which is Kubernetes. Each container contains the necessary application to run. Containers get started.
- While Pachyderm is started it stores the map input data in the database which is object storage.
- Through the container-A map data is converted into osm location reference data which contains our required attributes in JSON format.
- This JSON file is passed on to next container say container B which in turn converts the location reference data into GeoJSON format in order for the visualizer to understand.
- Container exits after its computation.

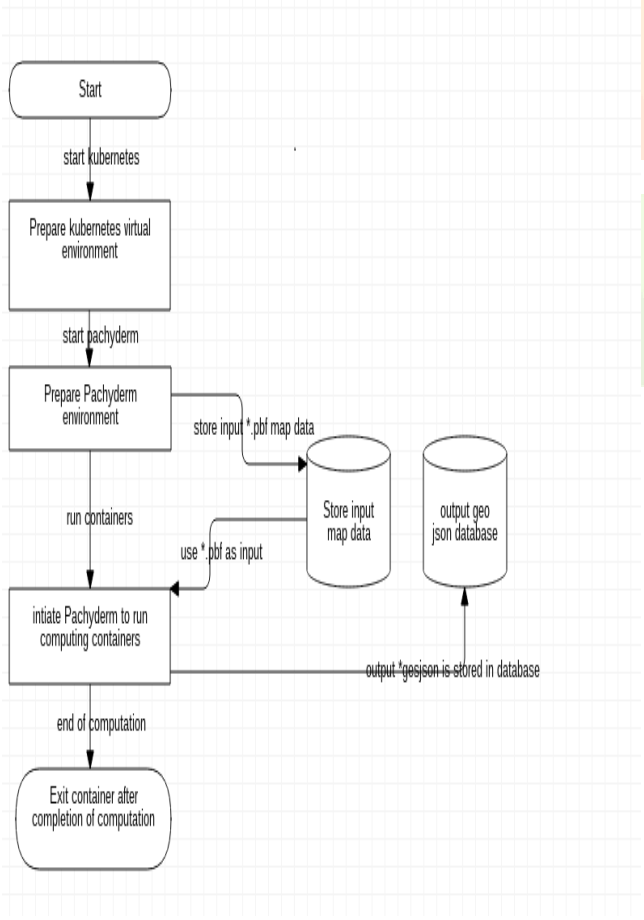

Fig3. Flow chart of the proposed system

Kubernetes is CaaS cloud computing service which is incorporated in a PaaS service. We start the kubernetes which prepares the Kubernetes virtual environment to build up. The next step after this is to start pachyderm environment. The containers are built in CaaS which is Kubernetes. Each container contains the necessary application to run. Containers get started. While Pachyderm is started it stores the map input data in the database which is object storage.

Through the container-A map data is converted into osm location reference data which contains our required attributes in JSON format. This JSON file is passed on to next container say container B which in turn converts the location reference data into GeoJSON format in order for the visualizer to understand. Container exits after its computation.

IV. RESULTS AND DISCUSSION

The map data is extracted and represented in the protobuf file format. This data is processed and transformed. Only the road attributes through which the vehicles can navigate from place to another location is extracted and stored. The final representation of this data will be stored in the JSON file Since JSON file format is not understandable by visualizer to represent the map data this JSON file has to be converted into GeoJSON file fomat. Hence the JSON file is converted into GeoJSON file. This finally is represented in this visualizer. The final road map data seen when we extract Bangalore region is as shown in the fig 4.

Fig4. The final road map data seen when we extract Bangalore region

V. CONCLUSION

The manuscript explores about the concept of using containers for processing geospatial data which is generally always in a large scale. The project describes and builds a distributed geospatial processing framework based on one of the evaluated open source stream and batch processing engines like Apache Hadoop, Spark and Pachyderm. These tools are used to perform Spatial ETL functionalities like extraction of data from different data sources, transformation of data to correct errors, cleanse, fuse, make them compliant to defined standards and load the transformed data into target spatial database (postgresql) or GIS file or Geospatial web services. It uses container based processing for doing so for large scale geospatial data. The cloud based processing engine used is CaaS. Kubernetes is used which processes the data. Container service is used even if the given infrastructure is not present. Use of pachyderm helps us in computing of large scale real time map data. Use of Kubernetes helps us in orchestration of containers. The protobuf format gives us the most efficient serialization of data because of which the total size of the data file gets reduced. The GeoJSON format is required for visualizer. This experimentation shows the proof of concept using distribute computation for processing large scale real time data.

REFERENCES

[1]. N. Naik, P. Jenkins, N. Savage, and V. Katos, "Big data security analysis approach using computational intelligence techniques in R for desktop users," in IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2016

[2]. N. Naik, "Building a virtual system of systems using Docker Swarm in multiple clouds," in IEEE International Symposium on Systems Engineering (ISSE). IEEE, 2016.

[3]. "Connecting Google cloud system with organizational systems for effortless data analysis by anyone, anytime, anywhere," in IEEE International Symposium on Systems Engineering (ISSE). IEEE, 2016..

[4]. N. Naik, "Applying computational intelligence for enhancing the dependability of multi-cloud systems using Docker Swarm," in IEEE Symposium Series on Computational Intelligence (SSCI), 2016.

[5]. "Migrating from Virtualization to Dockerization in the cloud: Simulation and evaluation of distributed systems," in IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments, MESOCA 2016. IEEE, 2016.

[6]. M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," Journal of Big Data, vol. 2, no. 1, pp. 1–21, 2015.