**Week 8 Assignment**

**Hadoop Framework**

Monroe College

CS 675: Big Data Management and Analytics

**Group 1**

Gita Gurung

Madeline (Zhiyang) Lin

Priyanka Chaudhari

Sukriya Gurung

Professor Syed Farid

November 2th, 2022

*For this assignment you should have already a Hadoop framework and ready to explain on a document how do you implement the framework. Then I will continue giving you guided steps for the following weeks to complete your knowledge and understanding of Big Data.*

*After you create the document you will upload your pdf file with snap shots of your framework.*

**ANS:**

Hadoop is a framework for storing and processing big data on multiple machines. Using multiple machines to process Big Data is very different from using one machine to process regular data. If we use the same methods, we won’t be making the most out of our greater processing power. The process needs to be divided up into smaller tasks for each machine and that is essentially what Hadoop does. On the storage end, Hadoop uses the Hadoop Distributed File System (HDFS) and from here it uses a processing model called MapReduce. Apache Hadoop is an open-source software framework that allows large sets of data to be processed using commodity hardware. Hadoop is designed to run on top of a large cluster of nodes that are connected to form a large distributed system.

**Hadoop Framework works on the following two core components:**

(1) **HDFS** – Hadoop Distributed File System is the java-based file system for scalable and reliable storage of large datasets. Data in HDFS is stored in the form of blocks and it operates on the Master Slave Architecture. HDFS (Hadoop distributed file system) designed for storing large files of the magnitude of hundreds of megabytes or gigabytes and provides high-throughput streaming data access to them. it supports the write-once-read-many model. A file once created, written, and closed need not be changed although we can append the data in the file. HDFS easily scales to thousands of nodes and petabytes of data. HDFS is highly fault-tolerant and is designed 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 few POSIX requirements to enable streaming access to file system data. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access. HDFS has a master/slave architecture. An HDFS cluster consists of**NameNode**, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system’s clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode. HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time. The NameNode makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a BlockReport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A BlockReport contains a list of all blocks on a**DataNode**.

(2**) Hadoop MapReduce** - This is a java-based programming paradigm of Hadoop framework that provides scalability across various Hadoop clusters. MapReduce distributes the workload into various tasks that can run in parallel. Hadoop jobs perform 2 separate tasks job. The map job breaks down the data sets into key-value pairs or tuples. The reduce job then takes the output of the map job and combines the data tuples to into smaller set of tuples. The reduce job is always performed after the map job is executed.

* **YARN:** YARN (Yet Another Resource Negotiator) is a resource manager that knows how to allocate distributed compute resources to various applications running on a Hadoop cluster. In other words, YARN itself does not provide any processing logic that can analyze data in HDFS. Hence various processing frameworks must be integrated with YARN. YARN provides a framework for managing both MapReduce and non-MapReduce tasks of greater size and complexity. YARN, similarly, to HDFS, follows the master-slave design with single Resource Manager daemon and multiple **NodeManagers** daemons having different responsibilities.
* **Resource Manager:**Keeps track of live NodeManagers and the amount of available compute resources that they currently have; Allocates available resources to applications submitted by clients; Monitors whether applications complete successfully. Resource Manager is a Per-Cluster Level Component. Resource Manager is again divided into two components:
  1. Scheduler
  2. Application Manager Resource Manager’s Scheduler is: Responsible to scheduler required resources to Applications (that is Per-Application Master).
* **MapReduce:**MapReduce on YARN is a framework that enables running MapReduce jobs on the Hadoop cluster powered by YARN. It provides a high-level API for implementing custom Map and Reduce functions in various languages as well as the code-infrastructure needed to submit, run, and monitor MapReduce jobs. Hadoop MapReduce is a data processing framework that can be utilized to process massive amounts of data stored in HDFS. Distributed processing of a massive amount of data in a reliable and efficient manner is not an easy task. Hadoop MapReduce aims to make it easy for users by providing a clean abstraction for programmers by providing automatic parallelization of the programs and by providing framework managed fault tolerance support. MapReduce programming model consists of Map and Reduce functions. The Map function receives each record of the input data (lines of a file, rows of a database, and so on) as key-value pairs and outputs key-value pairs as the result. By design, each Map function invocation is independent of each other allowing the framework to use divide and conquer to execute the computation in parallel. This also allows duplicate executions or re-executions of the Map tasks in case of failures or load imbalances without affecting the results of the computation. Typically, Hadoop creates a single Map task instance for each HDFS data block of the input data. The number of Map function invocations inside a Map task instance is equal to the number of data records in the input data block of the particular Map task instance. Hadoop MapReduce groups the output key-value records of all the Map tasks of a computation by the key and distributes them to the Reduce tasks. This distribution and transmission of data to the Reduce tasks is called the Shuffle phase of the MapReduce computation. Input data to each Reduce task would also be sorted and grouped by the key. The Reduce function gets invoked for each key and the group of values of that key (reduce<key, list\_of\_values>) in the sorted order of the keys. In a typical MapReduce program, users only have to implement the Map and Reduce functions and Hadoop takes care of scheduling and executing them in parallel. Hadoop will rerun any failed tasks and also provide measures to mitigate any unbalanced computations.

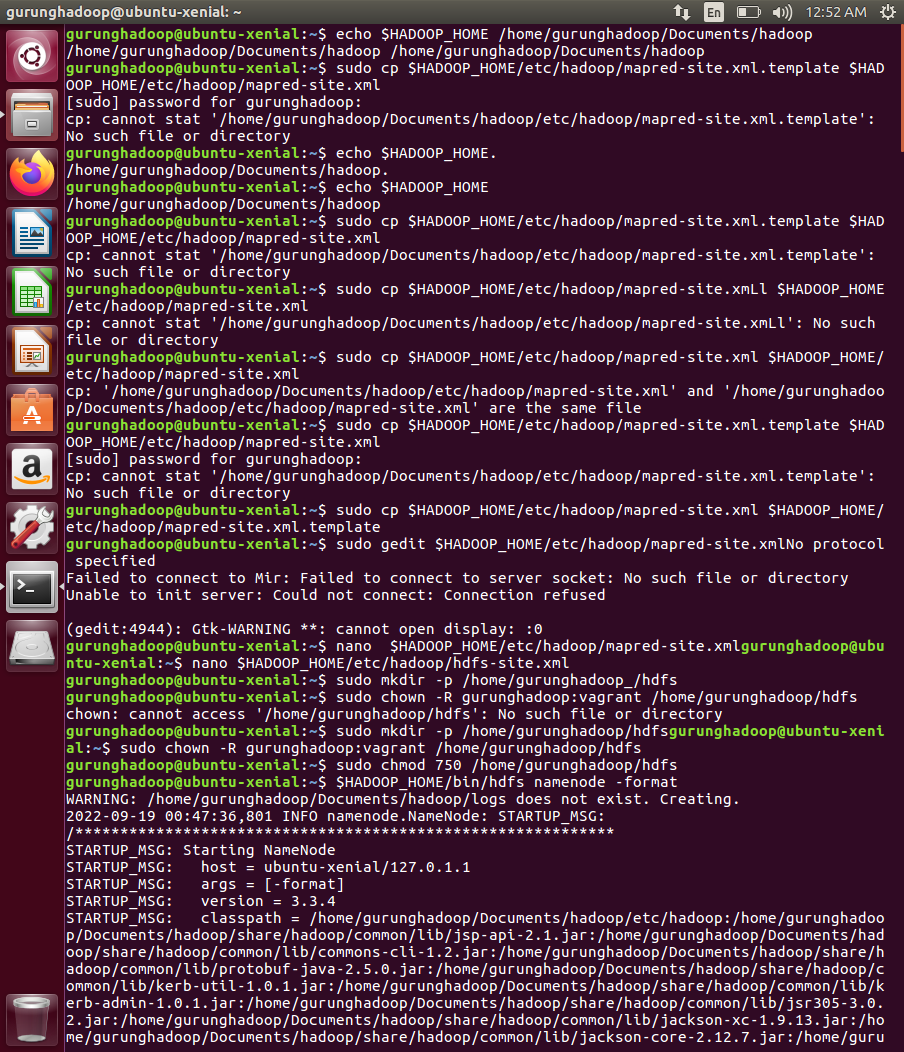
We successfully, completed the Hadoop installation even facing some issues while installing. The following are some screenshots of the work.

Text

Description automatically generated

Text

Description automatically generated



Text

Description automatically generated

Text

Description automatically generated

**Documentation of Week 8 Materials**

# **Real world application project for Big Data – with Apache Spark and AWS-EMR**

This article discusses about how to use Amazon S3 as data lake and schedule big data ETL through Airflow to process the high-volume logs and expose this data to analyst and scientist of the organization by consuming the data as SQL or Python.

# **Hadoop Examples: 5 Real-World Use Cases**

This article talks about how the industries analyze the big data to solve business problems and for decision making. Specifically, it talks about financial services companies, retailers, asset-intensive energy industry, telecommunications companies. Hadoop is used in these industries because it is effective, scalable, and is well supported by large vendor and user communities.

**Ch9 Big Data Governance Best Practices**

This lecture note discusses about the big data governance. Hadoop was initially developed without security or privacy considerations. Hence, there has been a huge gap in data management tools, structure, and operations of Hadoop in the past. Without proper data governance tools and measures, the Hadoop data lake can become a data swamp.

Data protection involves three categories of considerations: sensitive data protection, data sharing considerations, and adherence to governance policies.

Sensitive data protection is concerned with policies that ensure your operations are meeting regulatory and compliance requirement for sensitive data. This is predicated on classifying your data into critical, high-priority, medium priority, and low priority. All critical and high priority are regarded as sensitive. Some organizations refer to such sensitive information as private information(PI).

In terms of data sharing considerations, it is a critical governance consideration – in particular in large organizations, multi-division, and global companies whose data generation and consumption spans not only functional lines but countries, divisions, and affiliates such as corporate partners and franchises.

The flip side of enforcing governance policies is adherence. When it comes to the data protection at the high level, we need to apply tokenization prior to data ingestion into the lake. Data tokenization is an important strategy to protect data. When tokenizing data, it is advisable to store the token keys on the token servers on your premise in a different subnet, in particular if the data resides in the cloud. Tokenization replaces the digits with a randomly generated alphanumeric text ID (known as a “token”) which cannot be identified without de-tokenization using the initial key that was generated in the token server.

Additionally, this article discusses about the big data classification. There are three stages of data formats, including raw data, keyed data, and refined data. Raw data refers to the data with no transformation performed on it. It’s the data that is on-boarded from the source outside of the lake. Keyed data represents data that has been imported into some data structure such as HBase or Hive or similar data format. Keyed data is transformed into standardized file formats and compression. It maintains data keys and scheme are applied. Refined data refers to the data which is merged with other data, filtered, and parsed. In this stage, operational processing is performed, analysis is applied, and reports are generated. Users have access to this data for their analytical work. Refined data is typically created from validated data as users apply structure to the data. They may parse and merge files and records, add attributes, and ship the data out of Hadoop.

The article then talks about four stages of data in Hadoop data lake and metadata rules matched to data stages in the lake and the data usage rules. After that, it talks about data structure design and gives an overview of the sandbox functionality. Users may read data from the Hadoop lake and bring it over to the sandbox. As for detailed access policies, specific access policies can be defined for the sandbox and directory structure. The data governance best practice policies dictate that there should never by any reporting, analysis or publication directly from raw data. There should be no marketing activities, analysis, or reports generated directly from the raw data.

Finally, the article mentions about split data design. Best practices require files to be split physically in Hadoop to separate sensitive data from non-sensitive data.

**Ch10 Big Data Governance Framework**

This lecture note outlines a framework for implementing data governance for big data. Data governance refers to the rules, structures, processes, and practices that allocate the roles, responsibilities, and rights of participants in big data analytics. This governance policy is intended to meet all relevant compliance with applicable laws and objectives of operating the big data platform in a safe and sound manner.

The framework consists of eight capabilities including organization, metadata management standards, data classification, security, privacy, and compliance standards, data usage agreement (DUA), security operations and policies, information lifecycle management, and data quality standards.

The article then talks about the integrity of information, data resources, and assets; benefits of compliance and risks of noncompliance; terms that the program must define including Accountable Executive (AE), Data Council, Data Element, Data Levels, Data Quality Issue, Data Risk Officer (DRO), Data Steward (DS), Data Usage Agreement (DUA), Subject Matter Experts (SME), Third-Party Data Provider, Trusted Data, etc.

Big data governance is federated and organized by the data council. The data council meets regularly, typically monthly, to address changes in policy, manage data issues, audit results and issues related to compliance, security, and privacy. There are four data governance forums that are managed by the data council: 1) Metadata management, 2) Quality management, 3) Data governance, 4) Security/privacy and compliance. When it comes to data classification, it is an important step toward data and application management. As discussed in the Ch9 lecture notes, there are four data types: raw, keyed, validated, and refined.

As for data security, privacy and compliance standards, the security policies and processes outlined in this governance framework are aligned with the enterprise data security, privacy, and compliance rules. Four pillars of big data security are perimeter, access, visibility and protection.

Information lifecycle management governs the standards and policies of data from creation and acquisition to deletion.

As for data quality standards, it defines requirements to classify, document, and manage data to ensure that the big data platform’s critical and high priority data meets established quality requirements. The objective of this standard is to ensure that the platform’s data is managed and is of sufficient quality. All data whether internally or externally obtained is to be classified into four categories including critical data, high priority data, medium priority data, and low priority data. There are five metrics which are used when defining data quality rules for critical and high priority data elements. They are accuracy, validity, completeness, timeliness and consistency.

When data issues are detected from various sources including compliance reporting, control testing, audit findings, data quality and metadata monitoring activities, or from third party data management audits and activities, they must be documented and reported by the data steward for their area of responsibility. Such reports for critical and high priority data must include the remediation plan and target date for remediation. The criteria for triage of data quality issues are from level 1 to level 5, which are low, medium, high, extreme, and severity.