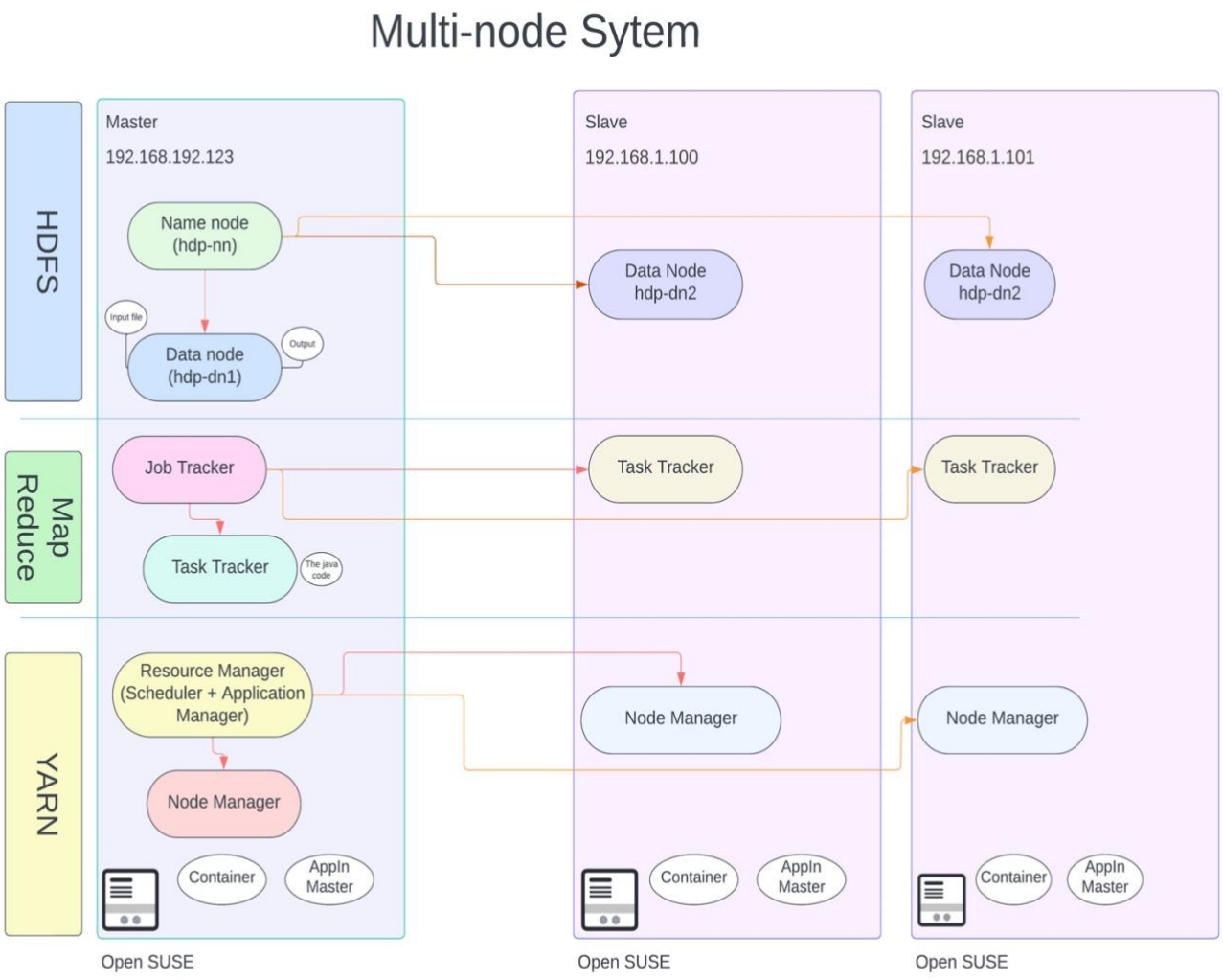
LAB -2 : Expanding Your Hadoop Cluster



Hadoop Cluster multi-node

Overview

A multi-node Hadoop cluster is a distributed system that consists of multiple nodes that work together to store and process large datasets. **Hadoop uses a master-slave architecture**, with one node acting as the master node (NameNode) and the other nodes acting as slave nodes (DataNodes). The NameNode is responsible for storing the metadata for the Hadoop Distributed File System (HDFS), while the DataNodes are responsible for storing the actual data.

Components

The main components of a multi-node Hadoop cluster are:

- NameNode: The NameNode is the master node for HDFS. It stores the metadata for all the files in HDFS, including the location of each file block and the replication factor for each block. The NameNode also coordinates the read and write operations to HDFS.
- DataNode: DataNodes are the worker nodes in HDFS. They store the actual data blocks. DataNodes also replicate the data blocks across the cluster to ensure that the data is not lost if a node fails.
- ResourceManager: The ResourceManager is the master node for YARN. YARN is a resource management framework that allocates resources to applications running on the cluster. The ResourceManager keeps track of the resources available on each node in the cluster and assigns them to applications based on the applications' needs.
- NodeManager: NodeManagers are the worker nodes in YARN. They are responsible for executing tasks on behalf of applications. NodeManagers also monitor the resource usage of applications and report back to the ResourceManager.
- ApplicationMaster: An ApplicationMaster is a special process that is launched for each application running on YARN. The ApplicationMaster is responsible for coordinating the execution of the application's tasks across the cluster.

Process Diagram

The process diagram of a multi-node Hadoop cluster shows how the different components interact with each other. The main interactions are as follows:

- **NameNode and DataNodes:** The NameNode communicates with the DataNodes to store and retrieve metadata for the HDFS. For **example,** when a client writes a file to HDFS, the NameNode tells the DataNodes where to store the file blocks. When a client reads a file from HDFS, the NameNode tells the client which DataNodes to read the file blocks from.
- **ResourceManager and NodeManagers:** The ResourceManager communicates with the NodeManagers to allocate resources to applications and monitor their progress. For example, when an ApplicationMaster requests resources for an application, the ResourceManager tells the NodeManagers how many resources to allocate to the application. The NodeManagers then execute the application's tasks and report back to the ResourceManager on the progress of the tasks.
- **ApplicationMaster and NodeManagers:** The ApplicationMaster communicates with the NodeManagers to execute tasks and manage resources for the application. For **example**, when an ApplicationMaster needs to execute a task, it tells a NodeManager to execute the task. The NodeManager then allocates resources to the task and executes it.

Scaling

Hadoop clusters can be scaled to handle large datasets by adding more DataNodes and NodeManagers. The number of DataNodes required depends on the amount of data to be stored and the replication factor. The number of NodeManagers required depends on the number of applications to be run and the resource requirements of the applications.

Pipeline Optimization

The following pipeline optimizations can be used to improve the analysis and delivery of data from the multi-node system:

- **Data partitioning:** The input data can be partitioned into smaller chunks, which can then be processed in parallel by multiple nodes. This can significantly improve the performance of data processing tasks.
- **Data caching:** Frequently accessed data can be cached in memory, which can reduce the number of disk reads and improve performance.
- **Data compression:** The data can be compressed before transmission, which can reduce network bandwidth usage and improve performance.
- **Data aggregation:** The data can be aggregated at intermediate stages of the pipeline, which can reduce the amount of data that needs to be transmitted and processed.
- **Data visualization:** The data can be visualized in real time using dashboards or other tools, which can make it easier for data scientists to identify insights and patterns.

MapReduce Search

To perform a MapReduce search to count the number of instances of each word, the following steps can be taken:

1. First data will be stored into data-node.

2. The JobTracker schedules the job's tasks on the TaskTrackers (another legacy component).

3. The TaskTrackers will execute below mentioned tasks and send the results to the JobTracker.

    - Split the input data into smaller chunks. This can be done by line or by file.
    - Map each chunk to a key-value pair, where the key is the word and the value is 1.
    - Reduce the key-value pairs by summing the values for each key.
    - Output the key-value pairs, where the key is the word and the value is the number of instances of the word.
4. The JobTracker merges the results and returns them to the user.

Benefits

There are several benefits to using a multi-node Hadoop cluster:

- Scalability: Hadoop clusters can be scaled to handle large datasets by adding more DataNodes and NodeManagers.
- High availability: Hadoop clusters are highly available because the data is replicated across multiple DataNodes. If a DataNode fails, the data can still be accessed from the other DataNodes.
- Cost-effectiveness: Hadoop clusters can be built using commodity hardware, which makes them relatively cost-effective.

Data System Proposal

Problem

The goal of this data system proposal is to design a system that can be used to analyze customer reviews to identify trends and patterns. The system will be used by a company that sells e-commerce products to improve their understanding of their customers and products.

Input Data

The **input data for the system will be customer reviews.** The reviews will be in a structured format, with each review containing the following fields:

- Product ID
- Customer ID
- Rating
- Review text

Data Structure

The data will be stored in a relational database, with each review stored in a separate row. The following table shows the database schema:

Column	Type	Description
product_id	INT	The ID of the product that the review is for.
customer_id	INT	The ID of the customer who wrote the review.
rating	INT	The rating that the customer gave the product (1-5).
review_text	TEXT	The text of the review.

Data Transformation

The data will be stored in a Hadoop cluster. The following storage techniques will be used:

- The review data will be stored in a **Hive table.**

- The review text will be indexed using **Elasticsearch.**

Pipeline

The following pipeline will be used to analyze the data:

1. The review data will be loaded into the Hive table.
2. The review text will be indexed using Elasticsearch.
3. Spark will be used to analyze the data. The following tasks will be performed:
    - ◊ Count the number of reviews for each product.
    - ◊ Calculate the average rating for each product.
    - ◊ Identify the most common words and phrases in the reviews.
    - ◊ Identify the most common topics discussed in the reviews.
4. The results of the analysis will be stored in a MySQL database.

Data Model

The following data model will be used:

- The Hive table will store the review data.
- Elasticsearch will index the review text.
- The MySQL database will store the results of the analysis.

Output

The output of the system will be a dashboard that visualizes the results of the data analysis. The dashboard will show the following information:

- The number of reviews for each product.
- The average rating for each product.
- The most common words and phrases in the reviews.
- The most common topics discussed in the reviews.

Concerns

The following concerns have been identified:

- The performance of the system may be an issue, as it will be processing a large amount of data.
- The accuracy of the results may be an issue, as the data analysis will be performed using machine learning algorithms.
- The interpretability of the results may be an issue, as it may be difficult for users to understand the output of the machine learning algorithms.

Mitigation Strategies

The following mitigation strategies will be used to address the concerns identified above:

- The system will be designed to be scalable, so that it can handle large amounts of data.
- The system will be validated using a held-out test set to ensure that the results are accurate.
- The system will provide explanations for the results of the machine learning algorithms, so that users can understand how the results were generated.

Additional Infrastructure

The following additional infrastructure will be required to support the system:

- A Hadoop cluster.
- An Elasticsearch cluster.
- A MySQL database server.

Implementation

The system will be implemented using the following technologies:

- Hadoop
- Elasticsearch
- Spark
- MySQL
- Django

Documentation

The following documentation will be used:

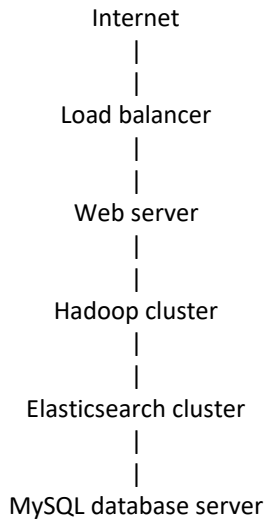
- Hadoop documentation
- Elasticsearch documentation
- Spark documentation
- MySQL documentation
- Django documentation

Networking Components

The following networking components will be used:

- A load balancer to distribute traffic between the web servers.
- A web server to host the Django application.
- A Hadoop cluster to store and process the data.
- An Elasticsearch cluster to index the review text.
- A MySQL database server to store the results of the analysis.

Diagram of the data system architecture



The following is a more detailed description of how Hadoop will be used in the system architecture:

1. The customer review data will be loaded into HDFS.
2. YARN will be used to schedule a Spark job to analyze the data.
3. The Spark job will use Hive to read the review data from HDFS.
4. The Spark job will analyze the data and generate the results.
5. The results will be written back to HDFS.
6. The results will be loaded from HDFS into the MySQL database server.
7. The dashboard will read the results from the MySQL database server and display them to the user.