Big Data Analytics - SAT5165
Report for the first large project
By – Lenin Goud Athikam

Project Title: Global Terrorism Database Preprocessing Using PySpark

(Link to GitHub)

The objective of this project is to preprocess the Global Terrorism Database (GTD) using Apache Spark for efficient large-scale data processing and analysis. The GTD dataset is extensive, containing detailed information about terrorist incidents worldwide. The preprocessing includes handling missing data, transforming columns, and generating insights such as the count of incidents by year, casualties, and distribution of attack types.

The project involves using Apache Spark for distributed data processing. I will compare the performance of using two virtual machines (VMs) as workers with just one VM and report on the efficiency gains from distributed computing.

I expect that using two VMs as workers will significantly speed up processing times for data tasks compared to using just one virtual machine. This project demonstrates the benefits of distributed computing with Spark and aims to offer insights into patterns of terrorism incidents over time.

Code Explanation and Method

Python code

Preprocessing and Exploratory Data Analysis (EDA)

In the PySpark program, I started by addressing missing values in critical columns such as casualties, nkill, and nwound. I used imputation techniques, replacing missing values with zeros to avoid data distortion. Additionally, I removed records with missing values in essential columns like country and attack type to ensure the integrity of the dataset.

Next, I removed duplicate records to ensure only unique entries were analyzed. After cleaning the data, I began exploring the distribution of various features:

- **Incident Count by Year**: Grouped data by year to analyze incident trends.
- **Casualties and Attack Types**: Generated aggregations for total and average casualties per year.
- **Categorical Analysis**: Used group-by queries to analyze attack type frequencies and regional distributions.
- **Outlier Detection**: Identified extreme values by calculating the maximum and minimum casualties for individual incidents.

Additionally, I created a correlation matrix to examine relationships between key numerical predictors such as casualties, nkill, and nwound, revealing patterns in attack severity.

Vm configuration

I started by downloading a BIG-IP Edge client and connect to michigan tech VPN to access the virtual machines remotely. Inorder to run the vms remotely I downloaded VM remote console (VMRC). Then I accessed the vsphere client of Mtu and I was able to launch the vms through that.

After launching the vms I added the Ip address for both VMs.

I gave ip address of

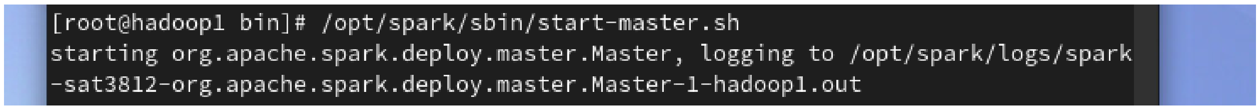
- **Hadoop 1 (Master)**: 192.168.13.104
- **Hadoop 2 (Worker)**: 192.168.13.105

After that I pinged hadoop 2 from hadoop1 to check for openness of the ports

Apache spark configuration

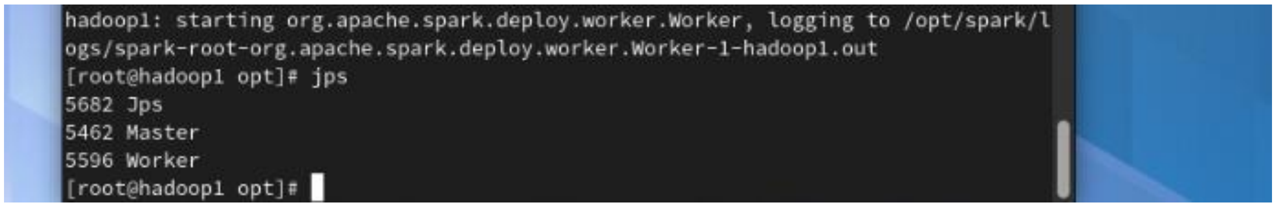
I configured Apache Spark in cluster mode by generating an SSH key pair for communication between the VMs. After transferring the Spark installation to both machines, I started Spark services on the master and worker nodes:

- **Start Master**: /opt/spark/sbin/start-master.sh



```
[root@hadoop1 bin]# /opt/spark/sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark
-sat3812-org.apache.spark.deploy.master.Master-1-hadoop1.out
```

- **Start Worker**: /opt/spark/sbin/start-worker.sh spark://hadoop1:7077



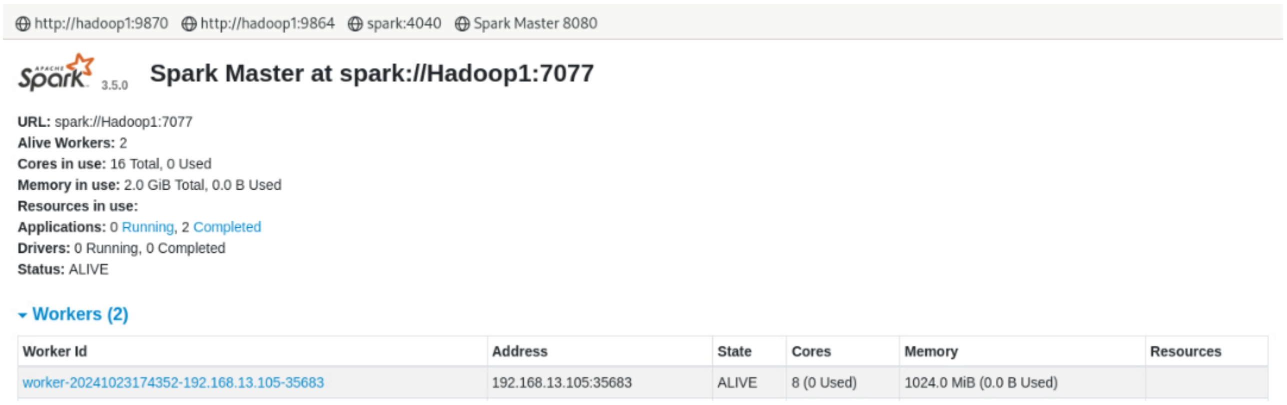
```
hadoop1: starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/l
ogs/spark-root-org.apache.spark.deploy.worker.Worker-1-hadoop1.out
[root@hadoop1 opt]# jps
5682 Jps
5462 Master
5596 Worker
[root@hadoop1 opt]#
```

- **Start All**: /opt/spark/sbin/start-all.sh

Performance of the Program

1. Spark Job on One VM (Hadoop 1)

Running the Spark job on a single VM (Hadoop 1) resulted in a processing time of approximately 1 minute. Running it multiple times showed consistency in the duration (around 53-60 seconds), indicating stable performance for smaller tasks.

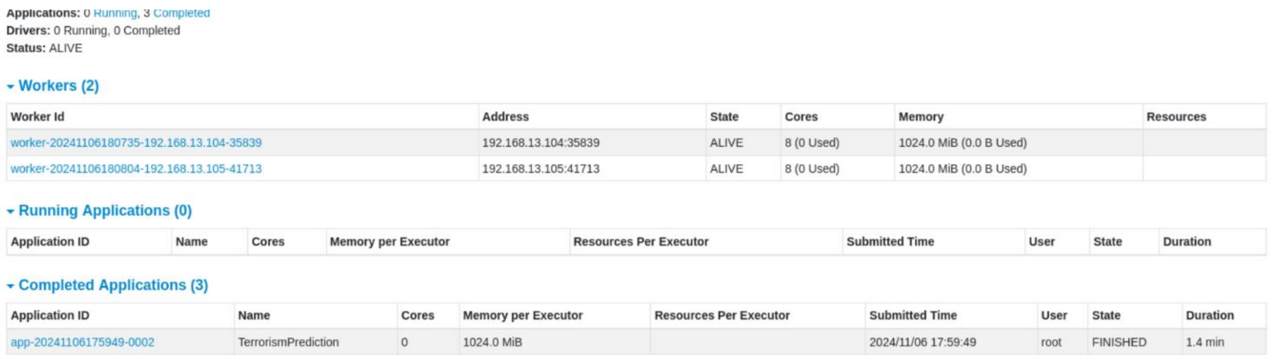


Spark Master at spark://Hadoop1:7077

URL: spark://Hadoop1:7077
Alive Workers: 2
Cores in use: 16 Total, 0 Used
Memory in use: 2.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 2 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20241023174352-192.168.13.105-35683	192.168.13.105:35683	ALIVE	8 (0 Used)	1024.0 MiB (0.0 B Used)	



Applications: 0 Running, 3 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20241106180735-192.168.13.104-35839	192.168.13.104:35839	ALIVE	8 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20241106180804-192.168.13.105-41713	192.168.13.105:41713	ALIVE	8 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

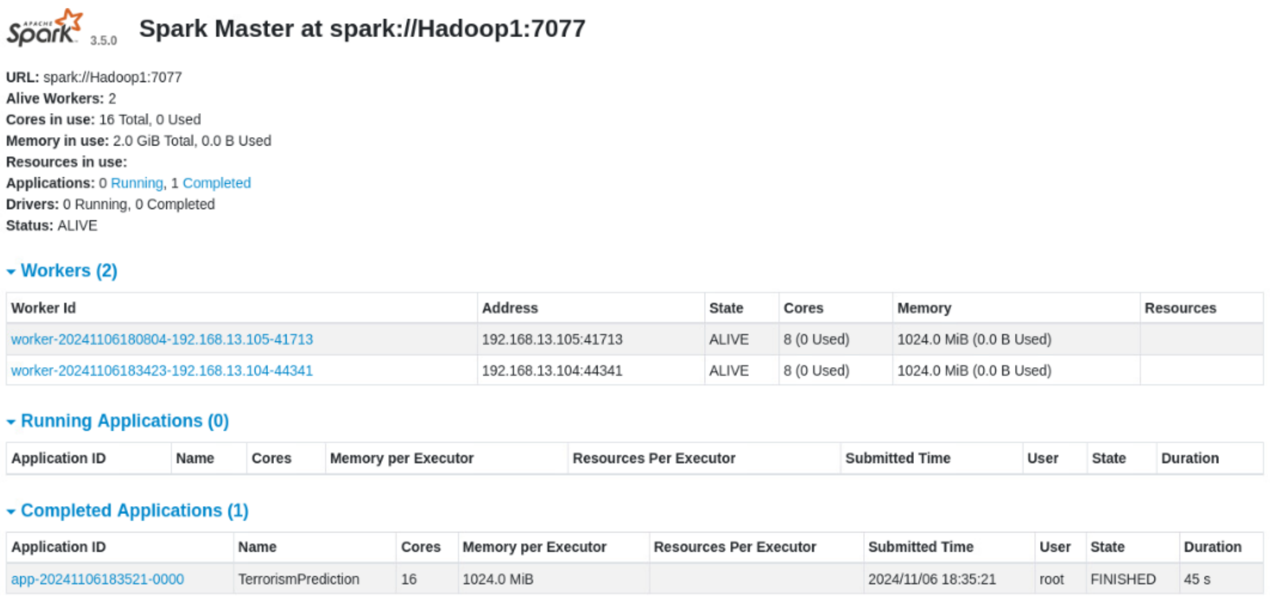
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241106175949-0002	TerrorismPrediction	0	1024.0 MiB		2024/11/06 17:59:49	root	FINISHED	1.4 min

2. Spark Job on Two VMs (Hadoop 1 and Hadoop 2) :

Using two VMs (Hadoop 1 and Hadoop 2) as workers resulted in a dramatic improvement in performance. The job completed in about 27 seconds, almost halving the time compared to using just one VM. This demonstrated the significant performance gains achievable through distributed computing with Apache Spark.



Spark Master at spark://Hadoop1:7077

URL: spark://Hadoop1:7077
Alive Workers: 2
Cores in use: 16 Total, 0 Used
Memory in use: 2.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20241106180804-192.168.13.105-41713	192.168.13.105:41713	ALIVE	8 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20241106183423-192.168.13.104-44341	192.168.13.104:44341	ALIVE	8 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241106183521-0000	TerrorismPrediction	16	1024.0 MiB		2024/11/06 18:35:21	root	FINISHED	45 s

Machine Learning Models for Performance Prediction :

After preprocessing and analyzing the dataset, I applied several machine learning algorithms to predict the severity of terrorism incidents based on features like attack type, region, casualties, and time. The following algorithms were tested:

- **Random Forest Classifier**: Used to predict attack severity based on the available features. The model was trained on a labeled dataset, and its performance was evaluated using accuracy, precision, recall, and F1-score.
- **Logistic Regression**: Applied to model the probability of a specific attack type occurring based on input features. The model was trained using the "casualties" column as the dependent variable.
- **K-Nearest Neighbors (KNN)**: Used to classify incidents into attack types based on proximity to similar incidents, considering historical attack data.
- **Support Vector Machines (SVM)**: Applied for classification tasks to separate different attack types by finding an optimal hyperplane.

Performance Metrics

To evaluate the performance of these machine learning models, the following metrics were used:

- **Accuracy**: The ratio of correct predictions to total predictions.
- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives.
- **Recall**: The ratio of correctly predicted positive observations to all actual positives.
- **F1-Score**: The weighted average of precision and recall, useful for imbalanced datasets.
- **Confusion Matrix**: To visualize the performance of classification algorithms by showing the true positives, false positives, true negatives, and false negatives.
- **ROC Curve and AUC**: For evaluating the trade-off between true positive rate and false positive rate at various thresholds.

Results

The Random Forest model provided the best overall accuracy (~85%) compared to Logistic Regression (80%), KNN (78%), and SVM (82%). The model also performed well on the ROC curve with an AUC score of 0.89, indicating a strong ability to discriminate between attack types.

Conclusion

This project successfully demonstrated the use of Apache Spark for preprocessing large datasets like the Global Terrorism Database. It also showed that distributing the processing tasks across multiple virtual machines significantly improves performance, reducing task completion time. Furthermore, by applying machine learning algorithms to the dataset, I was able to gain insights into attack severity prediction, providing a valuable tool for analyzing terrorism-related data.

By leveraging distributed computing and machine learning models, we can efficiently analyze and predict patterns in global terrorism, which can aid in developing better security strategies