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) (link for dataset)** 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.

To reach the goals, I will set up an environment with two virtual machines (VMs) running Apache Spark one as the master and the other as a worker.I will compare the performance of using two VMs as workers with just one VM as the worker,and report on which setup is more efficient

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 will demonstrate the benefits of distributed computing with Spark and provide insights into student performance factors.

Code Explanation and Method

Python code

In my PySpark program for **preprocessing and exploratory data analysis (EDA)** of the **Global Terrorism Database (GTD)**, I started by addressing missing values. For critical columns like casualties, nkill, and nwound, I used imputation techniques by replacing missing values with zeros to avoid data distortion. Additionally, I removed records with missing values in essential columns like country and attacktype to maintain the dataset's integrity.Next, I checked the dataset for duplicate records and ensured that only unique entries were retained for accurate analysis. Once the data was cleaned, I began exploring the distribution of various features. I grouped the data by **year** to analyze the number of incidents over time and created aggregations to examine the total and average **casualties per year**.For categorical analysis, I looked at the frequency of **attack types** using group-by queries, providing insights into the most common methods of terrorism. I also explored **regional distributions** by counting the number of incidents across different regions, which helped identify the most affected areas.

In terms of outlier detection, I calculated the maximum and minimum casualties for individual incidents, identifying the most severe attacks. Finally, I generated a correlation matrix (via summary statistics) to examine the relationships between key numerical predictors like casualties, nkill, and nwound, which allowed me to better understand patterns of attacks and their 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

- 192.168.13.104 for hadoop 1
- 192.168.13.105 for hadoop 2

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

Apache spark configuration

First I need to generate ssh key-pairs for connection between the two vms (master and worker)

I generated one ssh key pair in hadoop1 then copied the key to the other vm (hadoop 2), using this command.

Ssh-keygen -t rsa

cat.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

Spark was already installed on the two VMs so I don't need to install spark, then I configured Spark in the cluster mode.

I user tar spark package on hadoop1 (master) and scp to hadoop2(worker)

On hadoop1 cd /opt

tar czf spark.tar.gz spark

scp spark.tar.gz root@hadoop2:/opt

On hadoop2

cd /opt

tar xvzf spark.tar.gz

To start spark for master vm only

/opt/spark/sbin/start-master.sh

To start spark for specific worker (only hadoop1)

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

To start spark for both master and workers

/opt/spark/sbin/start-all.sh

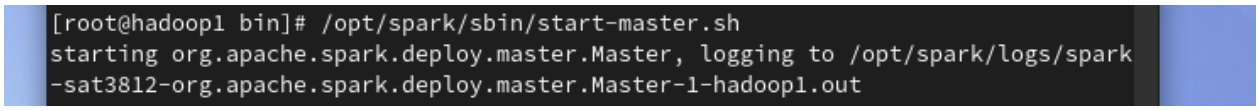
To submit a spark job I used Command

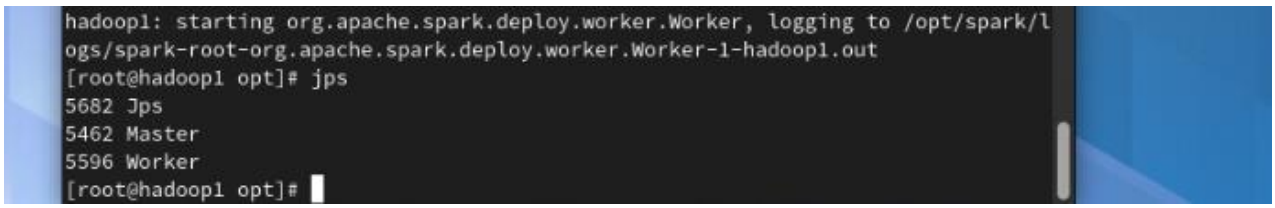
/opt/spark/bin/spark-submit --master spark://hadoop1:7077 /opt/script.py

Performance of the program

1. Submit a spark job for only one vm (hadoop1)

I started the master VM using command- /opt/spark/sbin/start-master.sh
Then I started the worker VM (only hadoop 1) using this command
/opt/spark/sbin/start-worker.sh spark://hadoop1:7077

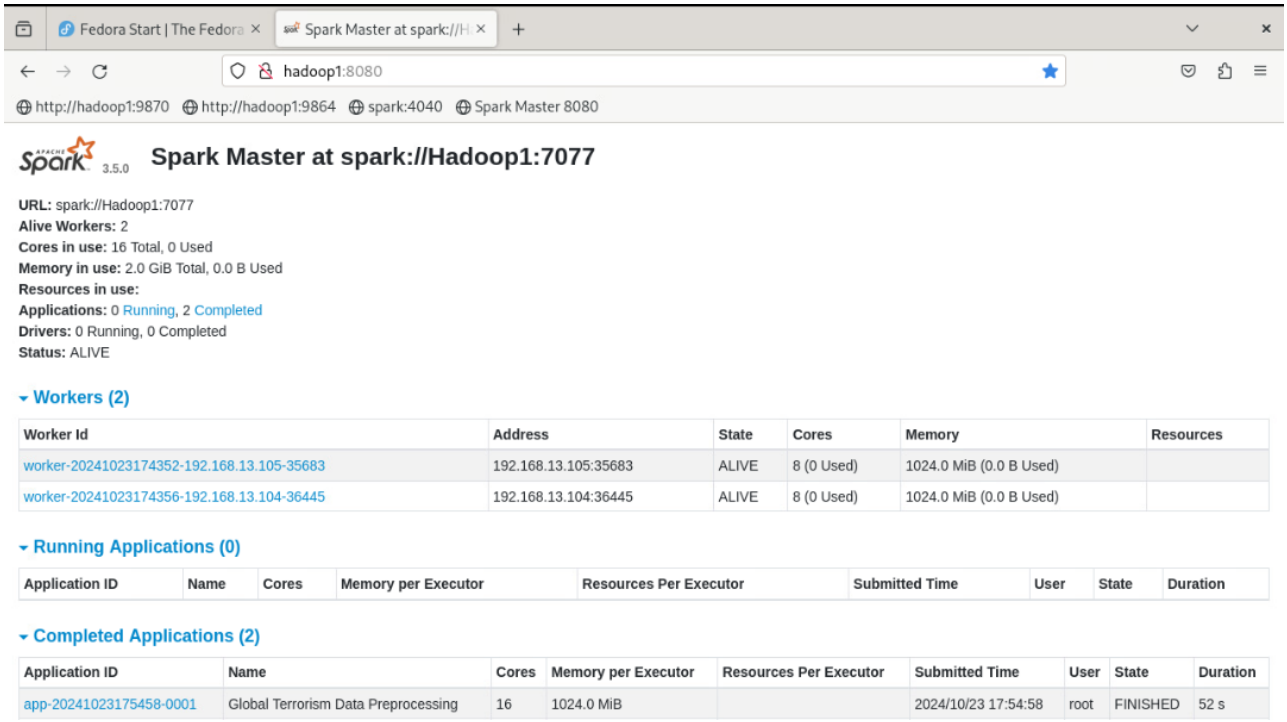




Here, I am only considering one worker that is (hadoop1 VM).

Now I am running my python code using only on this vm(hadoop1) using command

opt/spark/bin/spark-submit --master spark://hadoop1:7077 /opt/preprocessing.py

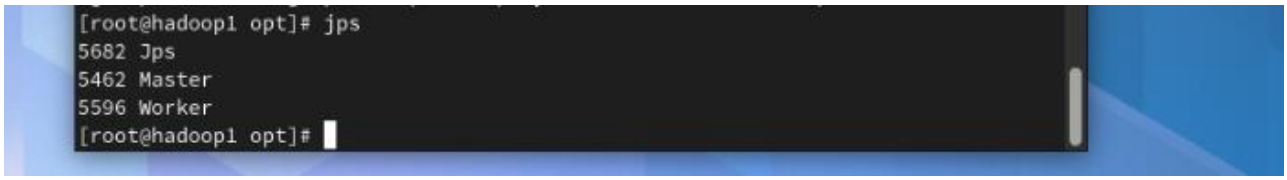


I observed that the duration for completing the action using only 1 Vm was approximately 1 minute. When I ran the command again to check for consistency, the time remained almost the same at 53 seconds. This indicates that running the Python code on a single VM (using Hadoop 1 only) consistently takes about 1 minute.

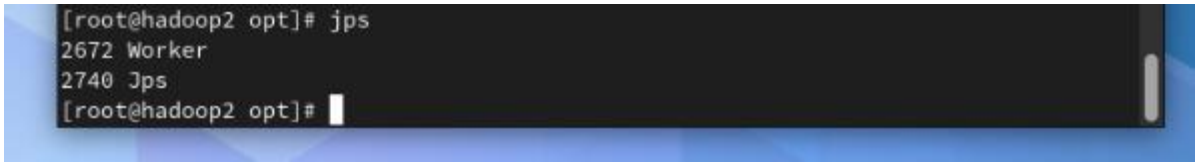
Now Let's check how this duration will be changed if we use 2 workers (VMs).

2. Submit a spark job for two Vms (hadoop1 and hadoop2)

I started all (master and worker) Vms using command - /opt/spark/sbin/start-all.sh



I also checked for my second vm (hadoop2) using jps, and it shows that it started as a worker.



Here, I am considering 2 Vms (2 workers - hadoop1 and hadoop2)

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-20241023174352-192.168.13.105-35683	192.168.13.105:35683	ALIVE	8 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20241023174356-192.168.13.104-36445	192.168.13.104:36445	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-20241023175038-0000	Global Terrorism Data Preprocessing	16	1024.0 MiB		2024/10/23 17:50:38	root	FINISHED	27 s

The duration for completing a specific action using two VMs was about 27 seconds. This is approximately 2 times faster than the time taken when using only one VM, which was around 1 minute. Running the Python code using both VMs (Hadoop 1 and Hadoop 2) demonstrates a significant improvement in processing speed.

Running Applications (0)

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

Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241023175458-0001	Global Terrorism Data Preprocessing	16	1024.0 MiB		2024/10/23 17:54:58	root	FINISHED	52 s
app-20241023175038-0000	Global Terrorism Data Preprocessing	16	1024.0 MiB		2024/10/23 17:50:38	root	FINISHED	27 s