Name: SIVAKRISHNA MEDA

UID: U01001538

COURSE: CS 6970

Application Problem:

Detection of Network Traffic Anomalies Using Apache Spark

Description of Problem:

Network anomaly detection systems have become a popular and useful approach for detecting anomalies, assaults, and intrusions as the volume of internet and IoT traffic on the network grows. Network traffic problems can be identified with reasonable classification accuracy using machine learning approach. However, the majority of previous work has been focused on finding abnormalities in a standard machine learning setting. Due to the ever-increasing volume of data and high-speed networks, traditional machine learning environments are no longer capable of dealing with the current situation. In this paper, we look at the viability of using Apache Spark, a big data technology, to solve a problem.

Introduction:

The utilization of AI, network traffic anomalies has been identified with sensible forecast exactness. Nonetheless, with the expanding measure of organization traffic and with the rise of fast organizations, conventional method of preparing the AI model and forecast or arrangement becomes infeasible with wide appropriation of web and IoT innovation, the measure of traffic that cross the organization is developing from time to time. An organization traffic peculiarity identifier has been used to identify the peculiarities from huge measure of network traffic with the goal that security assaults and oddity condition can be distinguished and make a move progressively. As abnormality discovery is significant in network security, there has been dynamic exploration regions since numerous years prior. As of now, there are a few proposed arrangements, to be specific, measurable examination and limit based abnormality discovery and AI/profound learning based methodology. Through the use of AI, network traffic peculiarities have been recognized with sensible forecast exactness. In any case, with the expanding measure of organization traffic and with the development of high velocity organizations, customary method of preparing the AI model and forecast or arrangement becomes infeasible.

Key Observations:

(a) Multi-class network assault abnormality identification framework utilizing Apache Spark's structure.

(b) Apache Spark structure adds huge execution gains than of customary strategies by changing Spark's design boundaries; agent centers and memory.

(c) Few unique AI calculations under various flash designs setting and talk about the suggestion dependent on discoveries.

The following topics are the basic building blocks of our real-time anomaly detection framework.

Data Center: A data center is the store house of data and it has a cloud storage appliance for its storage infrastructure. In addition to, it also has resource management software to manage its storage and infrastructure with VMware, OpenStack and Microsoft. Some advanced data centers have a dynamic and compatible operating environment allowing users to leverage internal and external resources to mitigate their extra demands.

Dynamic Resource Scheduling: VMware Dynamic Resource Scheduling (DRS) permits clients to characterize the guidelines and strategies that choose how virtual machines share assets and how these assets are focused on among various virtual machines. Some different guidelines like liking and against fondness rules [18] have likewise been characterized to work on the planning. Every one of the standards are static and can't be changed powerfully. To do ongoing booking, we need AI procedures. In the center of the scheduler, unaided (no preparation information), and regulated (preparing information) learning can be utilized to dissect stream information.

Spark Stream Data: If a user runs a resource intensive application (e.g., CPU or memory hungry), the respective counters in the performance metrics rise beyond their threshold. As a result, the application requires more resources to complete its task smoothly. Our real-time distributed framework should diagnose this event and reschedule the appropriate resources dynamically. VMware has some authoritative orders like [TOP], also, [ESXTOP], which give execution information of the virtual machines (VMs). Occasionally, we catch these insights as a depiction. Subsequent to preprocessing and highlight choice, an element vector is produced that goes about as an information point for the proposed information investigation calculation.

Apache Spark and Real-time Issues: Data center automation may require breaking down the exhibition information in real-time to distinguish inconsistencies. As the stream information comes consistently and is voluminous, we require an adaptable appropriated system. To address the adaptability issue, we can utilize an appropriated arrangement like Hadoop, Map Reduce, and so on Hadoop runs in clump mode and can't deal with ongoing information. As we are searching for constant information examination in an appropriated system, we have chosen to utilize Apache Spark which is fault tolerant, and supports dispersed ongoing calculation framework for preparing quick, huge floods of information.

Apache Spark is an open-source dispersed structure for information investigation with a straightforward engineering. It utilizes Hadoop for the dispersed document framework and can deal with the highest point of YARN, a cutting edge Hadoop bunch. Sparkle evades the I/O bottleneck of ordinary two-stage Map Reduce programs. It likewise gives in-memory group figuring that permits a client to stack information into a bunch's memory and inquiry it proficiently. This builds the presentation up to multiple times quicker than Hadoop Map Reduce.

Sparkle upholds both bunch and stream handling [23]. Its streaming segment is profoundly adaptable, and deficiency lenient. It utilizes a miniature clump method what separates the information stream as an arrangement of little bunched pieces of information from a little time span. It then, at that point conveys these little pressed pieces of information to a group framework for handling. Sparkle Streaming has two sorts of administrators:

- **Transformation administrator** - makes another D Stream from at least one parent streams. It very well may be either stateless (free on every stretch) or stateful (share information across spans).
- **Output Operator-**an activity administrator that permits the program to compose information to outer frameworks (e.g., save or print D Stream). Like Map Reduce, map is a change work that takes each dataset component and returns another RDD. On the other hand, decrease is an activity work that totals all the components of the RDD and returns the end-product (reduceByKey is a special case that profits a RDD). Sparkle Streaming acquires every one of the changes and activities of average group structures, including map, decrease, group by, and join, and so forth.

Real Time Anomaly Detection Framework: Data preprocessing, model training, prediction, model updating, and resource scheduling are all part of a real-time anomaly detection framework. The stream data is in an unprocessed format that must be treated before it can be evaluated. It is then turned to multi-dimensional time series data, such as CPU utilization or user program load in a data center over time, after processing. Data normalization and noise removal are sometimes required during preprocessing. Machine learning techniques are used to assess stream data in the real-time framework. It builds clusters on training data using unsupervised clustering methods (e.g., k-means, incremental clustering). Furthermore, the grouping only applies to data that is benign. It predicts data using cluster information after clustering. It is termed an anomaly if any data does not fall into any cluster; otherwise, it is regarded innocuous. The training model should be updated on a regular basis as the stream data changes in nature. Finally, if the framework detects any anomalous activity, it makes a request to the resource scheduler to update its resources**.**

Algorithm: Real Time Framework

procedure IDENTIFYANOMALY method (dataChunk, operation)

if operation == MODEL BUILD then

m ← BuildModelWithClustering(dataChunk)

if operation == PREDICT DATA then

hasAnomaly ← PredictData(dataChunk, m)

NotifyManagerForScheduning (hasAnomaly)

if operation == MODEL UPDATE then

m ← UPDATEModelWithClustering(dataChunk)

- ✓ Initially, we create the training model using the data from the incoming stream It uses the model to predict data after it has been built. Because of the dynamic nature of data, it's critical to update the training model on a regular basis. Following the creation of the training model, various new applications may be run in the virtual machines. which aren't out of the ordinary, but modify the CPU metrics These new CPU measurements might not fit into any of the previous model's clusters, thus they'll be classified as outliers during prediction. As a result, the model must be updated on a regular basis to account for these changes. Finally, if the framework detects any divergence from the expected behavior after prediction, it alerts the resource scheduler for rescheduling.

In this paper, we investigated different machine learning algorithms to detect network traffic anomalies. We describe these machine learning algorithms briefly in this section.

- **Logistic Regression (Multinomial)-LR** is a supervised learning method which can be used for classification for multiclass problems to classify more than two discrete values.
- **Decision Tree (DT)** is a non-parametric machine learning method, which is based on binary splitting features. Each leaf node in a DT represents a specific region or class with different data characteristics. DT is easily interpretable but might be unstable, since it is affected by the variance in the training set.
- **Random Forest (RF)** is a learning method which can be used for the problems such as classification, regression and other tasks that operates by constructing a multitude of decision at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

- **Multi-layer Perceptron (MLP)** is a deep, feed- forward, artificial neural network including more than one perceptron and different layers. It includes an input layer to receive the signal (input data), an output layer to give a probability vector for predictions or only one prediction and a different number of hidden layers in order to represent the input vector in a more abstract form. A single perceptron in each layer calculates a weighted sum of the input and applies a non-linear activation function to this weighted sum. The output of one perceptron is fed as an input to the perceptron of the next layer.
- **Naïve Bayes** is a probabilistic classifier which is based on probability models that incorporate strong independence assumptions. Naïve Bayes is popular among the classification algorithms for its training speed. However, it is more suitable for binary classification and not for multi-class classification.

Dataset: MAWILab is a data source that academics frequently utilize to test their traffic anomaly detection algorithms. It includes network traffic traces data from a backbone link between Japan and the United States. These network traffic traces are available in two formats: CSV and XML. Over 80 features for network traffic data and labels with either normal or a specific type of attack, such as DoS, DDoS, PortScan, Brute force, SYN, FIN j, Ping flood, FTP, SSH, heavy hitter, ptpDoSSYN, alphflHTTP, ntscACK, ntsc TCPRSTACKrp, mptp are included in the data set.

No	Traffic		No of Records
1	Normal		1743179
2	Attack	DoS, DDoS	128027
		PortScan	158930
		Brute force	36
		FTP, SSH	1966
		SYN, FIN j, Ping flood	2180
		heavy_hitter, ptpDoSSYN, alphflHTTP	13835
		ntscACK, ntscTCPRSTACKrp, mptp	252672

Fig. Distribution of MAWI dataset

Technology Stack: For implementation, Python API (PySpark) and Jupyter Notebook with Apache Spark version 2.4.0 is the best programing language. We'll read CSV files as a dataframe as the initial stage in the preprocessing. To speed up the computing process, features like anomaly id, empty columns, and negative-value columns are removed. In large datasets, data is frequently jumbled, and missing values might occur. Instead of deleting records with missing value fields to solve this problem, we filled them with default values to make the dataset full. We used the Apache Spark Machine Learning library to develop a Machine Learning pipeline after dealing with superfluous features and missing values. A pipeline is a collection of steps, each of which is a Transformer or an Estimator. These stages are completed in the order listed. A data frame will be used as the input, which will be changed as it passes through each stage.The data was then separated into training and testing sets in an 80:20 ratio, with a random split across the data set for multiple execution rounds. The model was trained using the training set. The mean of 10 times run is presented as final findings in order to acquire a fair result.

No	Algorithms	Accuracy (%)		
		2	4	8
1	**Decision Tree**	84.2829	84.6629	84.7041
2	**Logistic Regression**	95.8147	95.8150	95.8310
3	**Random Forest**	93.7426	94.9984	95.0559
4	**Multi-layer Perceptron**	75.2488	75.6851	83.0799
5	**Naïve Bayes**	18.0024	18.1120	18.2050

Fig. Attack Detection Accuracy

We analyzed the effect of varying Spark executor cores on execution time. As shown in the Fig.1, execution time of four algorithms except multi-layer perceptron, namely, decision tree, logistic regression, random forest and naïve bayes, is decreasing with the increase of the executor cores. This demonstrates that Spark's parallel execution feature improves the speed performance of these five algorithms. To put it another way, these algorithms can be successfully parallelized on Spark. However, during our numerous runs of tests, the multi-layer perceptron technique, which is a subset of the deep learning model, did not produce consistent results. Each run's execution duration varies dramatically. We believe this is due to the nature of deep learning models and random weight assignment during the first phase of training.

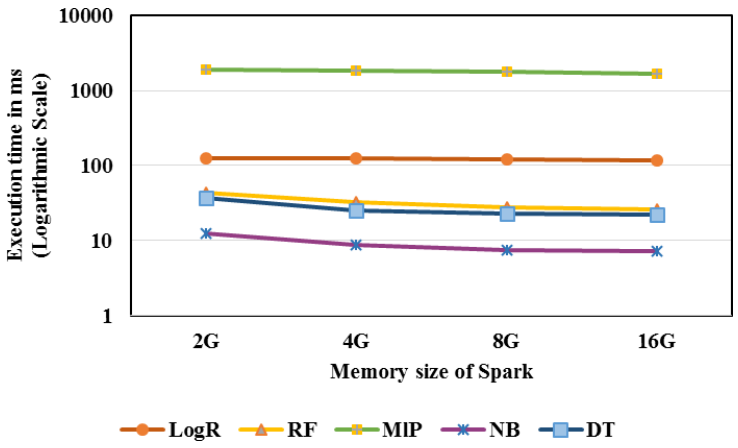

Fig. The impact of different executor cores on the execution time of an algorithm

The executor cores to 8 and altered the memory size, 2G, 4G, 8G, and 16 G, to see what effect increasing Spark's memory size had. The average of ten execution times is given in the results. Prior to doing this experiment, we believed that increasing the memory size would improve speed performance. However, following the experiment, we noticed that increasing the memory amount had no effect on the algorithm execution time. Among these, logistic regression and multi-layer perceptron produce consistent results across a wide range of memory sizes.

Conclusion:

Apache Spark to build a real-time anomaly detection framework in this article. It is capable of managing streaming data and ensuring data delivery. It also has fault tolerance and can scale up in parallelism. Our real-time technology is envisioned as a fundamental component of VMware's dynamic resource management system. Our framework is versatile, and it may be used to monitor system logs to assess a system's operational performance, analyze sensor data from embedded devices, and so on. Other machine learning techniques will be implemented in our framework in the future. Furthermore, we will consider other performance measures

(for example, memory and storage statistics) and attempt to correlate them in order to find anomalies. To categorize multi-class anomalies, several traditional machine learning techniques were applied. The effectiveness of the categorization algorithms was assessed using accuracy in the evaluation procedure. We also tested the detection system's performance with different executor cores and memory sizes. Increasing the number of executor cores in Spark improves prediction outcomes, while altering the memory size has little or no influence on learning algorithm speed and prediction performance. We'll expand on our work with spark streaming, which attempts to detect network traffic irregularities in real time and integrate it into the software defined network's control layer. Furthermore, it is focused on the creation of an anomaly detection system based on a multi-node Spark cluster, as well as the evaluation of its effectiveness.

Discussion:

While working on this case study topic I learned Apache key features such as Hadoop integration, Multiple format support and Real time computation. Similarly, I understand benefits of spark over Map Reduce and Yarn cluster. However, Resilient Distributed Dataset (RDD) explains parallelized collections and Hadoop Datasets these are immutable sets which helps data consistency. Functions of Spark core is the base engine for parallel and data pre-processing helped me to understand the concepts of memory management, fault recovery and how data interacts with storage systems.

References:

- Affinito, A. Botta, L.Gallo, M.Garofalo and G.Ventre, “Spark-based Anomaly detection: The case ofport and net scan detection”, 2018, https://arxiv.org/abs/1806.11047.
- https://www.infoworld.com/article/3236869/what-is-apache-spark-the-big-data-platform-that-crushed-hadoop.html.
- A. McCallum and K. Nigam, “A comparison of event models for Naïve Bayes text classification”, Workshop on learning for text categorization, 1998.
- Apache hadoop nextgen mapreduce (yarn). [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/ hadoop-yarn-site/YARN.html
- J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.