Design Document on

Log analysis using distributed system using MapReduce and Hadoop.

Submitted by

Group 16

Uddhav Raj B150878CS
Akansh Kumar B150398CS
Meshram Rohit Ajit B150895CS
T.S.Ashutosh B150640CS

Under the Guidance of

Dr RajaSree T

Department of Computer Science and Engineering
National Institute of Technology Calicut
Calicut, Kerala, India - 673 601

April 8, 2018

1 Introduction

A distributed denial of service attacks are the most serious factor among network security risks in distributed computing environment.The attacker often compromises the vulnerable hosts by flooding the large amount of packets to the computer network or server .

An HTTP flood attack is an application layer attack that targets the web applications and web servers. The client sends the request by using HTTP GET or POST method to the target web server. The web server processes the request and returns the response back to the client. An attacker sends a large volume of GET (image or scripts) or POST (file or forms) requests to overwhelm the target server to saturate all their computing resources. The victim server allocates many resources in response to the single request. This prevents legitimate requests from reaching the server causing a flooding attack.[2]

We try to propose a method of integration between HTTP flooding among DDOS attacks and MapReduce processing for a fast attack detection.

2 Architecture/Framework

As the volume of Internet traffic data has highly increased, the processing of the enormous volume of data to detect the attacks will take several days to compute the task by using a single machine.Hadoop is an open-source distributed cluster platform that includes a distributed file system, HDFS and MapReduce.Hadoop is a framework that allows us to store Big Data in a distributed environment,so that, we can process it parallely.

Hadoop enables parallelism by splitting the les into different blocks of specied size. Files smaller than the Hadoop block size are not split over multiple nodes for execution. Therefore, the overall detection time remains the same over different cluster node.[2]

MapReduce model is a parallel processing model proposed for quick operating batch jobs for Internet services. This model is also applied to perform computations on large volumes of data.

Components of Hadoop are:

1. HDFS- It creates an abstraction for distributed storage. We can see it as a single logical unit but actually we are storing our data across multiple nodes.A framework for job scheduling and cluster resource management.[5]

2. Yarn- It performs the processing activities by allocating resources and scheduling tasks.MapReduce is a YARN-based system for parallel processing of large data sets.

A MapReduce program is composed of a Map task that performs ltering and sorting, and a Reduce task that performs a summary operation.

Mapper:

After starting MapReduce, the rsttask is a mapper task which takes input from HDFS as a block.Mapper job takes pair of data as input and returns a list of pairs (key, value). Mapper output type may differ from mappers input type.

After the mapper(s) have nished their jobs, the data or worker nodes perform a shufe step. During shufing the nodes redistribute the data based on the output keys, such that all data belonging to one key is located on the same worker node.

Reducer:

Once the mapper tasks are completed, the reducer will start operating on the list of key/value pairs produced by the mapper functions. The reducers are assigned a group with unique key, it means that all the packets with unique key (unique src IP in our case) will be assigned to one reducer. We can congure Hadoop to run reducer jobs on varying number of data nodes.

3 Methodology

We have proposed a DDoS detection method based on Hadoop . We will use a Hadoop based packet processor and devise a MapReduce based detection algorithm against the HTTP GET ooding attack. We will employ a counter-based DDoS detection algorithm in MapReduce that counts the total trafc volume or the number of web page requests for picking out attackers from the clients. Counterbased method uses time interval, threshold and unbalanced ratio for the detection of attacks.[1]

Step1: Flooding on hadoop datanode
Step2: Capturing the traffic
Step3: copying that file to hadoop user
Step4: job assignment
Step5: map and reduce task
Step6: Collecting results

3.1 Collecting Log files

Before using the server logs we need to collect the logs.The process of collecting the data from the real web servers, and in this work, Apache web server is used.Step1 and Step2 are handled in this part.

Whenever a user sends a request for the information in the server, a log is created and maintained automatically by the server. The combined log format of an access log trace of an Apache server contains different kinds of information such as remote host, remote login name, remote user, timestamp (day/month/year:h:min:s+zone), HTTP request (HTTP method, URL and HTTP version), HTTP status code, length of the data in bytes, referral URL and user agent. An example of the access log trace is given as follows:

105.24.45.178 [17/Mar/2015:07:53:33+700] GET/ scripts/root.exe?/c+dir/HTTP/1.0 200 578 https://www.nitt.edu/OLCLD/view.php?q=book/ Mozilla/4.08 [en] (Win98; I; Nav).

These logs are used to determine the attacks by the analysis of the request strings and other parameters.

On successful transfer of log the le(s), the detection server split the le(s) into same size blocks and starts MapReduce DDoS detection jobs on cluster nodes . Once the detection task is nished, the results are saved into HDFS.

3.2 Mapper and Reducer

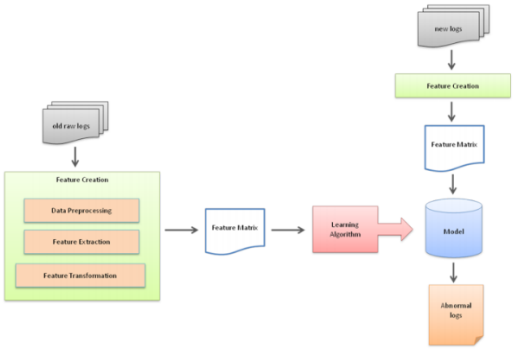
In our case , the map function filters non-HTTP GET packets and generates key values of server IP address, masked timestamp, and client IP address. The masked timestamp with time interval is used for counting the number of requests from a specific client to the specific URL within the same time duration.

Mapper job also use hashing for combining all the logs of data on the basis of src IP address, so that it becomes easier for reducer to analyze the attack trafc.The reduce function summarizes the number of URL requests, page requests, and server responses.

The reducer function takes input in key/value pair and produces a single key/value pair output after counting the number of instances.

4 Flowchart

1. Performing DDoS attack on localhost using python script. 2. Collecting log file from Apache server 3. Feature Extraction, where only some relevant information from each log file is extracted. Example of such extracted features are character bigram ,trigram and n-gram [6] and timestamp statistics. In this step Hadoop is used both as a storage system (HDFS) and as a computation engine for extracting the features. 4. Feature Transformation, where normalization and dimensionality reduction is performed on the extracted features, in an effort to reduce the size of the data. 5. This algorithm is trained on the data, and the produced model will be later used to identify abnormal log entries.



5 Algorithm

5.1 Feature Extraction

The feature extraction is the most important task when using machine learning techniques. Select-

ing features that can differentiate between normal and abnormal logs can lead to better results, while selecting irrelevant features will lower the performance of the learning algorithm. Appropriate features will lead to improved model interpretability, shorter training times, and enhanced generalisation by reducing the chance of over-fitting, a situation where the model learned to recognise the specific training inputs instead of the underlying relationship.

5.1.1 Character bi-gram

Character bi-grams is a very common feature when using machine learning techniques on free text [7]. Character bi-grams belong to the more generic n-gram. An n-gram is a sequence of n items taken from a string. In character bi-gram specifically, the sequence of all two adjacent characters is produced. For each log file, the number of times each bi-grams appears is calculated and then divided by the total number of bi-grams in that log, thus the feature matrix is added with the frequency of all possible bi-grams.

5.1.2 Time-stamp statistics

Log files contain time-stamp information for each of their entries. The total execution time of a test case can definitely help identify abnormal logs. For example, the fact that a test case was running for too long, for example for one hour, most likely indicates that something is not working properly and further investigation is required. Therefore the time difference between adjacent log entries are computed.

Let t(l, e) be the timestamp for the entry e of the log file l, —l— the number of entries in l. If t(l, e) denotes the time difference between the entry e and e+1, as shown in equation 3.1, then T(l) denotes all time differences in the log file l.

t(l, e) = t(l, e+1)t(l, e)

Then the following statistics are calculated for each log and are added in the feature matrix:

Average: mean(T(l))
Variance: var(T(l))
Sum: sum(T(l))
Maximum: max(T(l))

5.1.3 Number of Records

The length of a log file can also help identify abnormal logs. A very large log might not be considered normal. Therefore, the number of entries each log has is added in the feature matrix.

5.1.4 Feature matrix

To conclude, the feature matrix is constructed by concatenating all the above mentioned vectors. So, each vector of the matrix consists of:

[Character bi-grams — tri-grams — n-grams — Timestamp statistics — Number of Record]

5.2 Feature Transformation

The step following the Feature Extraction is the Feature Transformation. Here the feature matrix is normalized and its dimensionality is reduced in an effort to improve the training process.

5.2.1 Normalization/Scaling

A common problem is having some features with a large range of values, while others having rather small ones. For example, in feature matrix, the frequency of the character bi-grams are in [0,1], while the number of lines, of even the time-stamp statistics have no upper limit. In this case, the larger one becomes dominant. To minimize this problem, data normalization is performed on the feature matrix in order to make it have consistent statistical properties. The normalization is done by subtracting the average of each column of the feature matrix, and then dividing by the columns standard deviation, thus, forcing the ranges to be in [1,1] during training.

5.2.2 Dimensionality Reduction

The way the feature matrix was created, it is in a high dimensional space. This necessitates the reduction of the matrix since it might lead to dimensionality disaster ,a case where the distance between two data points becomes meaningless. Additionally, having higher dimensionality increases the training time.

5.3 Clustering

5.3.1 K-Means Clustering

Given a set of observations (x1, x2, , xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (n) sets S = S1, S2, , Sk so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where i is the mean of points in Si. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

The equivalence can be deduced from identity:

$$\sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_i - \mathbf{y}).$$

Because the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in different clusters (between-cluster sum of squares, BCSS), which follows from the law of total variance.

5.3.2 Agglomerative Clustering

Hierarchical clustering relies using the clustering techniques to find a hierarchy of clusters, where this hierarchy resembles a tree structure, called a dendrogram. Agglomerative clustering uses a bottom-up approach, wherein each data point starts in its own cluster. These clusters are then joined greedily, by taking the two most similar clusters together and merging them.

5.3.3 Affinity Propagation

Let x1 through xn be a set of data points, with no assumptions made about their internal structure, and let s be a function that quantifies the similarity between any two points, such that s(xi, xj) ¿ s(xi, xk) iff xi is more similar to xj than to xk. For this example, the negative squared distance of two data points was used i.e. for points xi and xk, The

$$s(i,k) = -\|x_i - x_k\|^2$$

diagonal of s (i.e. s (i , i)) is particularly important, as it represents the input preference, meaning how likely a particular input is to become an exemplar. When it is set to the same value for all inputs, it controls how many classes the algorithm produces. A value close to the minimum possible similarity produces fewer classes, while a value close to or larger than the maximum possible similarity, produces many classes. It is typically initialized to the median similarity of all pairs of inputs.

The algorithm proceeds by alternating two message passing steps, to update two matrices:

The "responsibility" matrix R has values r(i, k) that quantify how well-suited xk is to serve as the exemplar for xi, relative to other candidate exemplars for xi.

The "availability" matrix A contains values a(i, k) that represent how "appropriate" it would be for xi to pick xk as its exemplar, taking into account other points' preference for xk as an exemplar.

Both matrices are initialized to all zeroes, and can be viewed as log-probability tables. The algorithm then performs the following updates iteratively:

First, responsibility updates are sent around:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\}$$

Then, availability is updated per

$$a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k))\right) \text{ for } i \neq k \text{ and}$$

$$a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k)).$$

5.3.4 Spectral Clustering

Given an enumerated set of data points, the similarity matrix may be defined as a symmetric matrix A displaystyle A , where A i j 0 represents a measure of the similarity between data points with indices i and j. The general approach to spectral clus-

tering is to use a standard clustering method (there are many such methods, k-means is discussed below) on relevant eigenvectors of a Laplacian matrix of A. There are many different ways to define a Laplacian which have different mathematical interpretations, and so the clustering will also have different interpretations. The eigenvectors that are relevant are the ones that correspond to smallest several eigenvalues of the Laplacian except for the smallest eigenvalue which will have a value of 0. For computational efficiency, these eigenvectors are often computed as the eigenvectors corresponding to the largest several eigenvalues of a function of the Laplacian.

Spectral clustering is well known to relate to partitioning of a mass-spring system, where each mass is associated with a data point and each spring stiffness corresponds to a weight of an edge describing a similarity of the two related data points. Specifically, the classical reference explains that the eigenvalue problem describing transversal vibration modes of a mass-spring system is exactly the same as the eigenvalue problem for the graph Laplacian matrix defined as

$$L := D - A,$$

where D is the diagonal matrix

$$D_{ii} = \sum_j A_{ij}.$$

The masses that are tightly connected by the springs in the mass-spring system evidently move together from the equilibrium position in low-frequency vibration modes, so that the components of the eigenvectors corresponding to the smallest eigenvalues of the graph Laplacian can be used for meaningful clustering of the masses. A popular related spectral clustering technique is the normalized cuts algorithm or ShiMalik algorithm introduced by Jianbo Shi and Jitendra Malik,[2] commonly used for image segmentation. It partitions points into two sets (B 1 , B 2) based on the eigenvector v corresponding to the second-smallest eigenvalue of the symmetric normalized Laplacian defined as

$$L^{\text{norm}} := I - D^{-1/2} A D^{-1/2}.$$

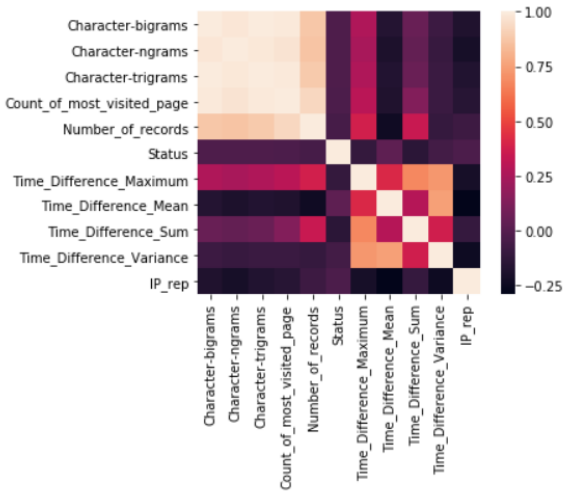
A mathematically equivalent algorithm [3] takes the eigenvector corresponding to the largest eigenvalue of the random walk normalized adjacency matrix

$$P = D^{-1} A.$$

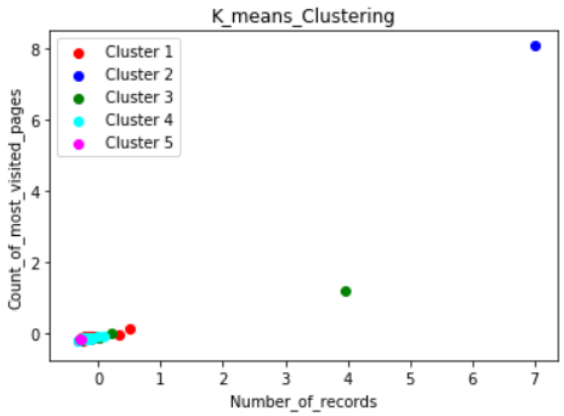
Knowing the eigenvectors, partitioning may be done in various ways, such as by computing the median m of the components of the second smallest eigenvector v , and placing all points whose component in v is greater than m in B_1, and the rest in B_2.

6 Result

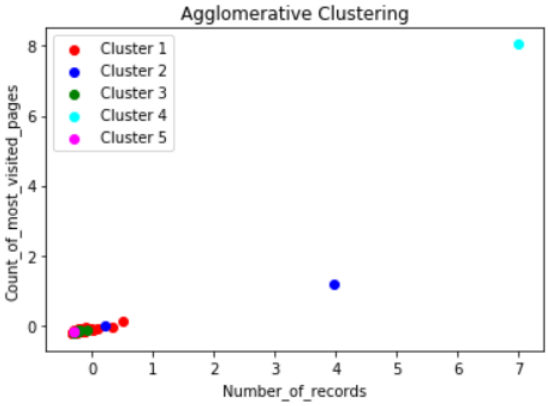
First we found out the correlation amongst the features using heat seaborn's HeatMap to plot the graph. Two features found were Count_of_most_visited_page and Number_of_records.



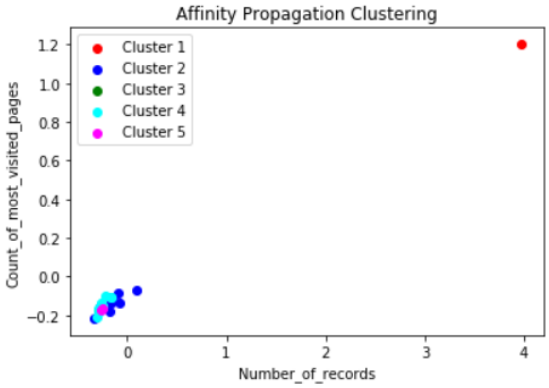
6.1 K-means Clustering



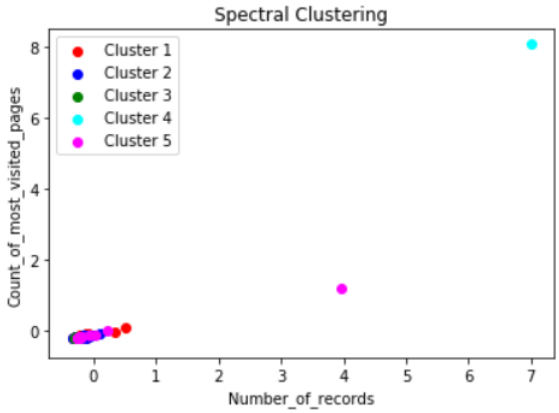
6.2 Agglomerative Clustering



6.3 Affinity Propagation Clustering



6.4 Spectral Propagation Clustering



7 Inference

For this dataset Agglomerative Clustering was found to give best result.

References

[1] Yeonhee Lee and Youngseok Lee, Detecting ddos attacks with hadoop, ACM Student Workshop, December 6 2011, Tokyo, Japan.

[2] Suan Hameed and Usman Ali, Efcacy of Live DDoS Detection with Hadoop.

[3] Thankaraja Raja Sree and Somasundaram Mary Saira Bhanu, HADM: detection of HTTP GET flooding attacks by using Analytical hierarchical process and DempsterShafer theory with MapReduce,Wiley Online Library ,October 17 2016.

[4] Mr.Akshay Dattatray Shete and Mr.Sudhanshu S. Gonge,Review on Detecting DDoS Attacks using MapReduce in Hadoop,International Journal of Innovative Research in Computer Science Technology,ISSN: 2347-5552, Volume-2, Issue-2, March-2014

[5] https://hadoop.apache.org/

[6] W. B. Cavnar, J. M. Trenkle et al., N-gram-based text categorization, Ann Arbor MI, vol. 48113, no. 2, pp. 161175, 1994.

[7] K. Pearson, Liii. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 2, no. 11, pp. 559572, 1901.