On Parallel Honeynet Log Mining Algorithm using P-Apriori in Cloud Environment

Jun Luo[1, a], Jing Liu[2,b] and Song Han[1,c]

[1] Department of IW, Naval Command College,10 Zhuhai Road, Nanjing, China

[1]Naval CA Station, Xi Sanhuan Road, Beijing, China

[a]zyqs1981@163.com, [b]lj384@hotmail.com, [c]HS0101@vip.163.com

Keywords: Honeynet, Log Mining, Cloud Computing, P-Apriori

Abstract. The current analysis on honeynet log mining was facing malpractice of horizontal partitioning and insufficient data problems. By which we use Apriori algorithm with parallel processing to decrease the frequency of database scan operation and set up an improved algorithm S-Apriori, and then P-Apriori algorithm was put forward through better case analysis and vertical partition data origination, which can be integrated with Hadoop parallel Computing environment. Simulations prove that the S-Apriori, P-Apriori algorithm improved compute efficiency significantly, the simulation results are also analyzed by the minimum support, the impact of the number of nodes on the P-Apriori algorithm running time.

Introduction

There are important linkages between the attackers' information & behavior where we can locate the real cyber threat, e.g. an attacker would first open port scanning system before next the specific attack step, therefore, it is difficult to judge the attacker's behavior under isolated analysis of the separate information, and we need to link the attackers' behavior to the overall analysis. With the growing number & type of the cyber-attacks, it will consume more in analyzing honeynet logs, where the real attack behaviors are located.

The aim of Honeynet log Mining is to discover unknown attacks. Network security personnel can using honeynet log mining system to discovery methods and models for hackers before take measures to promptly stop network attacks, targeted to strengthen the network security defense, and reduce losses. But till now there is no good Mining solution for this problem in detecting real attack in a large amount of extracted network stream.

Current Log Mining Algorithm

Comparisons. Relevance algorithms including STEM, AIS, DHP are designed on a single machine environment, so we choose Apriori algorithm which has excellent parallelism, running in a private cloud parallelization computing environment [1].Apriori algorithm is a classical association rule mining algorithm which often used in the network security field in digging out the rules for the invasion of the event. Each record can be seen as a transaction, we define all log records constitutes a transaction database *D*, but the amount of log records is very large, so that database scanning operation will be performed frequently , by which the efficiency of the algorithm will seriously affected. Therefore, to improve the honeynet log mining algorithms is a viable and necessary way to improve efficiency.At present, there are two main aspect of improvement, the reduction of the scan operation and decrease the generation of candidate set.

Hadoop, the main framework of big data parallel computing, is a distributed system which can be run on the basis of large-scale cluster, it is an open source framework, complete using Java Runtime, with scalability, low cost, high efficiency parallel processing data, and strong reliability[2].

MapReduce parallel processing combines fault tolerance, localized computing, data distribution, load balancing together, can be used in the application of Apriori algorithm, where the large data files

can be easily divided for parallel processing. Hadoop integrated into the honeynet log mining system, can improve efficiency of the analysis.

Definitions.

Definition 1: Define the dataset *D* for association rule mining which denoted called transactional database, set up $D = \{T_1, T_2, T_3, \mathbf{L}\, T_n\}$, $T_k = \{I_1, I_2, I_3, \mathbf{L}\, I_m\}$,which $T_k (k = 1,2,3, \mathbf{L}\, n)$ become Affairs, $I_i (i = 1,2,3, \mathbf{L}\, m)$ became the project.

Definition 2: Let $I = \{I_1, I_2, \mathbf{L}\, I_m\}$ be the set of all items in *D*. Item set is a subset of each transaction T_k included in the correlation analysis, if an item set contains *k* entries, then called *k* term set.

Definition 3: Support of itemsets

For Itemsets X , $X \subset I$, if the number of transactions contained in the database, then the support of item sets (support) is:

$$Support(X) = \frac{count(X \subseteq T)}{D}. \tag{1}$$

Support (X) denotes the probability of item sets that appear in need itemsets association rules meet the minimum generated when the support threshold, called min_sup item sets. Frequent itemsets are those who support greater than or equal to Itemsets min_sup composed, if the item set to meet support is greater than min_sup, called frequent itemsets, denoted by L [k].

Definition 4: association rules

Association rules can be expressed as a implication $R : X \Rightarrow Y$.

Wherein $X \subset I, Y \subset I$, and $X \cap Y = \Phi$. It appears that the item set, item collection will appear in the same transaction in accordance with a certain probability.

Definition 5: Rules of support

Association rules, which, and the support of the regulations is the number of X and Y co-workers appeared with all the ratio of number of transactions, that is $Support(X \Rightarrow Y) = \frac{count(X \cup Y)}{|D|}$. Since the association rules by frequent item sets to come, so support and frequent item sets the rules of equal support:

$$Support(X \Rightarrow Y) = Support(X \cup Y) = \frac{count(X \cup Y)}{|D|} \text{ viz.} \tag{2}$$

Definition 6: Confidence

Association rules, which, and the rule R confidence (confidence) and the number refers to the simultaneous occurrence of the ratio of the number only appear. Denoted that $confidence(X \Rightarrow Y)$.

P-Apriori Mining Algorithm

First improved Apriori algorithm was named S-Apriori(Sequential Apriori), the following steps for the S-Apriori algorithm description:

S-Apriori Procedure. In this paper all examples are using the transaction database *D*, for example, the feasibility of the algorithm is described. Examples of analysis before determining the minimum support required, the minimum support threshold = set, namely.

Algorithm (frequent itemsets generation process) of specific implementation process is as follows:

(1) D scanning, vertical format generation transaction database while the number of records for each project transaction support, namely the support count. Support delete support count is less than the threshold value of the project, generate frequent item se.

(2)Goto produce 2- itemsets frequent. In order to take the connection, namely, ... ,, ..., ..., generating 2 items candidate set. Compare Remove support count (project support count is just to support the transaction set for this project intersects the intersection after the number of transactions

for the project is to support the count) is less than the threshold value of project support, generate frequent item set 2- L [2].

(3) Generated by the frequent itemsets candidate set. Generate candidate item sets, in order to connect, just take the same item between two items connected to the preceding paragraph (frequent item generated when two items in the preceding paragraph is consistent, shaped like), compared to delete support count is less than support threshold of entry generate frequent item sets. Examples of items herein generate a candidate set, compared to less than support threshold delete items generate frequent item sets L [3].

(4) Set to give a final take frequent item sets. This article example L [4] is empty will meet the conditions of frequent item sets the output to give a final frequent item sets.

P-Apriori Procedure.

(1)Scan files, get frequent 1 item set. Suppose dealing with a size M of large data files, the number of transactions is N, the size of each data block HDFS is B, Hadoop system maximum parallelism Map task MaxMap months, the data is divided into M / B block, the first step to Each task is assigned a data block Map, scanning each item and calculate the support count, the next task if the current task is completed, the request Map. Reduce process is responsible for the output of all Map tasks to integrate them, delete support count is less than the minimum support threshold of 1 items candidate set, which is equivalent to a database format, such as a vertical format to generate database format D.

(2) The vertical division of the transaction database *D*. MapReduce by implementing the transaction database D in accordance with the scope of the transaction ID is vertically divided into n blocks of data sent to the node performs Map task.

(3) The formatted data blocks. Divide the data block type, responsible for parsing each data block is shaped like a Map to the transfer to perform tasks specific format, where ID represents the transaction database project, TID indicates the transaction identifier contains the item.

(4) Run Map function. Map function will achieve S-Apriori algorithm to generate local frequent sets k- items. But here need a little change is that when you delete a lot of useless candidate set cannot be compared deleted in accordance with the minimum support threshold, if the entry in the Map1 mission support count of 1, while at Map2 mission support count is 1, then the total The support count of 2, any delete-item operation in Map1 will result in frequent itemsets missing. Therefore, when you delete a lot of useless candidates only need to remove support for the item count is 0, where the S-Apriori algorithm is different.

Simulation & Results

Using Data Factory generator system generates test data provide experimental data set association rules, Apriori algorithm, S-Apriori algorithm, P-Apriori algorithm three algorithms were compared and analyzed data the number of transactions as a unit, chapter simulation, the maximum size of data most 200,000 transactions.

(1) The execution time of three algorithms comparison, the same minimum support, the case of a different number of transaction:

I. Apriori algorithm and S-Apriori algorithm is shown in Figure 1 at the time of a single computing node needs to run.

II. The same as the case shown in a different number of transactions, respectively, in a single computing node S-Apriori algorithm runs in time and Hadoop cluster running P-Apriori algorithm needed contrast in Figure 1. Experimental data shows that with the increase in minimum support to reduce intermediate qualifying frequent item sets, the greater the minimum support was, the less the algorithm running time will be. In support of the same case, the time required to process the data the greater the longer. At the same minimum support, the amount of data from 10 000 to 150 000 transactions, transactions increased by 15 times, but the P-Apriori mining algorithm running time did not increase 15-fold, but increased by about 6 times this indicates that P-Apriori parallel mining algorithm has good parallelism.

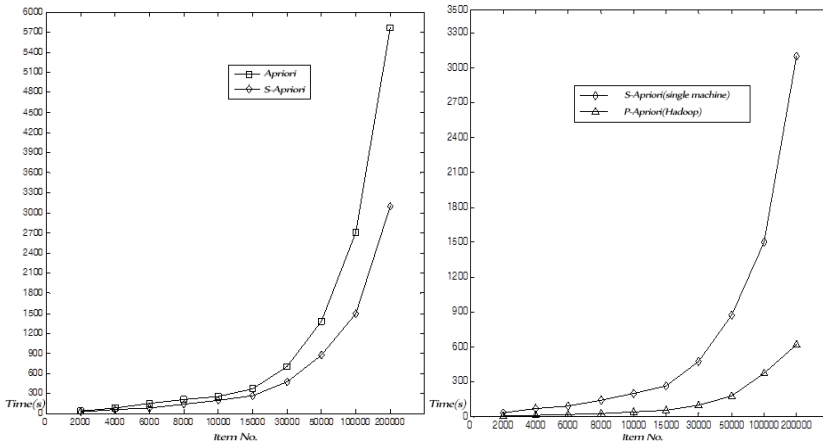

Figure 1. Comparison: Apriori, S-Apriori in Single machine& S-Apriori, P-Apriori in Hadoop

In the minimum support of 0.5 percent of the cases, the transaction increased by 15 times, but the running time is increased by about 8 times, the cause of this result is that P-Apriori algorithm vertical partition data, when performing tasks Map frequent deletion candidates only cut support for the item count is 0, therefore, with the support of smaller, complicated by the efficiency of the algorithm performance will be reduced.

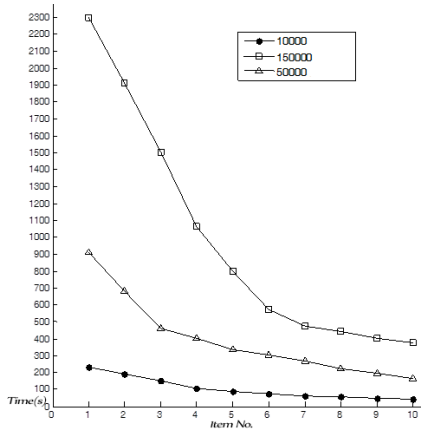

Figure 2 P-Apriori in different minimum support case.

(2) Under the same circumstances min_sup transform computing the number of nodes, the size of the data comparing different clusters running P-Apriori time required, shown in Figure 2. when the data in the single-node data mining, P-Apriori and S-Apriori handle the same amount of the transaction, P-Apriori mining algorithm running time is greater than the single-node direct run S-Apriori algorithm, which because Hadoop parallel mining task scheduling more time, so in a case of P-Apriori compute nodes running time is longer than a single node S-Apriori running time. Can also be seen from Figure 2, with the compute nodes to 6, the running time to reduce the rate leveled off. For smaller data, increase the number of computing nodes is difficult to enhance the efficiency of mining.

References

[1] Takabi H, Joshi J B D, Ahn G J. Security and Privacy Challenges in Cloud Computing Environments[J]. IEEE Security & Privacy, 2010, 8(6): 24-31.

Reference to a book:

[2] Martin G, Lippold A. Forge. mil: A case study for utilizing open source methodologies inside of government[M]. Open source systems: Grounding research. Springer Berlin Heidelberg, 2011: 334-337.

[3] Huang W, Yang J. New network security based on cloud computing[C]. 2010 Second International Workshop on Education Technology and Computer Science. 2010, 3: 604-609