Distributed Processing of Snort Alert Log using Hadoop

JeongJin Cheon [#1], Tae-Young Choe [2]
[#] Department of Computer Engineering, Kumoh National Institute of Technology
Gumi, Gyeongbuk, Korea
[1] cjjjjjin@gmail.com
[2] choety@kumoh.ac.kr (corresponding author)

***Abstract*—Snort is a famous tool for Intrusion Detection System (IDS), which is used to gather and analyse network packet in order to decide attacks through network. Until now, although processing a number of warning messages in real time, Snort is executed mainly in single computer systems. Unfortunately, current amount of network messages exceeds processing capacity of single computer systems. In order to embrace the huge amount of network messages, we have constructed a distributed IDS using Hadoop, HDFS, and 8 working nodes. Experimental results show that our distributed IDS has 426% performance compared to a single computer system.**

Keyword-Intrusion Detection System, Snort, Distributed Framework, Hadoop, HDFS

I. INTRODUCTION

Both the number of required server computers and the amount of network traffic have been on the rise. Since attacks on the computer systems also have been increasing, exact and fast analysis on attack packets are urgent. Many server systems adopt network intrusion detection systems (NIDS) in order to analyse network packets and warn against suspicious packets. Snort is an open source NIDS system and is widely used by various servers [1][2]. Unfortunately, a single computing system could not accommodate the amount of warning message analysis if the warnings occurs in multiple distributed servers. For example, we have observed that a network intrusion detection system generates about 8,700 warning messages when it watches 100Mbps network line. A personal computer system with Pentium 4 3.2GHz CPU processes about 12,000 warning messages per second. Thus about 72.5% of CPU resource is consumed for processing warning messages and it hinders the computer system from servicing the given tasks.

In order to improve performance of warning message process, researchers have improved string match algorithms or have adopted distributed processing. Since string match is one of the main process in warning message analysis, improved string match algorithms reduces the percentage of warning message analysis time [3]-[6]. Unfortunately, the methods do not improve the performance of the analysis over the portion of performance improvement of string match. Distributed processing methods divide a list of warning messages into multiple parts, distribute them to multiple computers, order the computers to process the list parts, and make a result by gathering the partial results from the computers. Although the distributed processing methods require additional operation time such as dividing the list and merging the partial results, they can improve the performance as the number of working computers increases.

In the paper, we propose a distributed Snort system that collects warning messages from distributed servers and analyse alerts using Map-Reduce function of Hadoop. While the proposed distributed Snort system with one node shows about 60% performance compared to a single computing system that does not uses Hadoop, it shows about 426% performance if the number of nodes is 8.

The remainder of this paper is organized as follows: in Section 2, related previous works are presented. Section 3 describes our proposed distributed Snort system that uses Hadoop in order to reduce warning analysis time. Experimental results and analysis are shown in Section 4. Finally, Section 5 draws a conclusion and future works.

II. RELATED WORKS

A. Snort and multi-core adaptation

Snort is one of the most famous network intrusion detection systems supported as open source and has GNU GPL. It has added detection rules through its homepage continuously. Also any system manager can add personalized detection rules by herself. Currently, Snort has been ported into multiple operating systems like Windows, Mac OS, and Linux.

Since Snort does not support multi-threading, its performance is not scalable to multi-core systems. In order to adapt Snort to multi-core systems, many application methods and modifications to Snort are proposed. Intel proposed a method that adapts Snort into multi-core processors using pipeline and flow-pinning [7], where flow-pinning is to classify packets and to send them to specified core. Such flow-pinning reduces the amount of data

to be reloaded into L2 cache. It has better performance if the number of TCP connection is high, while it has similar performance when the number of TCP connection is less than 200.

Another method is to modify Snort in order to run it as multi-threading [8][9]. Derek et al. proposed a method that allocates modified Snort to each thread in two directions: one for conservative parallelism using locking, the other for optimistic parallelism [8]. The performance shows 4.1 speedup on 8 processors at best for conservative parallelization and 3.0 speedup on 6 processors for optimistic parallelization. Jiang et al. noticed that matching fingerprint in pre-filter deteriorates performance in parallel systems and proposed a fingerprint extraction strategy [9]. The strategy is to train patterns using probability of occurrences. Since the strategy tries to reduce execution time of pre-processing in sequential steps, various parallel processing methods can be applied. Open Information Security Foundation (OISF) has developed Suricata an open source-based intrusion detection system [10]. Suricata supports multi-threading and can uses Snort rule set.

B. ICAS (IDS log Cloud Analysis System) and Hadoop related studies

Yang et al. proposed ICAS that analyses log files generated by Snort in cloud systems using Hadoop. In ICAS, IDS scans network packets and generates alert logs [11]. Next, regular parser picks out worthwhile logs and stores the selected logs into Hadoop distributed file system (HDFS). HDFS is a distributed file system for Hadoop [12]. HDFS duplicates file blocks and stores the duplications to different storages. ICAS concurrently analyses log files and merges them into a result file using MapReduce function supported by Hadoop. ICAS with 6 nodes shows about 89% performance improvement compared to a system with single node.

Lee et al. proposed a binary input format `PcapInputFormat` to find start points of packets in large log files stored in HDFS [13]. Since HDFS divides a file in 64Kbytes sized blocks, each block contains lots of merged packet messages where some parts are binary format and some messages are divided by blocks. The main idea in PcapInputFormat is to find time-stamp field in order to distinguish each packet messages in a message dump file. Unfortunately, the format is required because message dump file is input for the system. If Snort scans network packets directly, it does not require the format `PcapInputFormat`.

III. DISTRIBUTED SNORT SYSTEM

Since regular parser of ICAS selects meaningful logs from logs in the node, the node under attack is troubled by heavy loads from attack and regular parser. Also the amount of logs stored in HDFS is too small for an IDS to take advantage of parallel processing in Hadoop. In order to balance load, we propose a distributed Snort system that stores warning logs generated by Snort into HDFS without any pre-processing. In order to describe the proposed IDS system, some properties of Hadoop and a package Chuckwa are explained.

A. Hadoop

Hadoop is an open-source distributed framework written in Java programming language under Apache License [14][15]. It supports functions for processing huge size data. Hadoop system is constructed with a master server and multiple slave nodes as shown in Fig. 1. Hadoop has two logical modules: HDFS (Hadoop File System) and MapReduce. HDFS manages duplicated blocks for data and meta-data in order to tolerate fault in a node. A file is composed of 64Kbytes blocks and a block is duplicated to three replicas in general. When a block is created, the first replica of the block is written in the DataNode where the block is created. The second replica is written in the different DataNode in the different rack. Rack is a group of DataNodes that shares the same network switch. The store location for the replica is chosen randomly in order to increase concurrency. Thus a file is distributed to multiple nodes which can be accessed concurrently. Namenode process in the master server stores metadata and locations of the replicas.

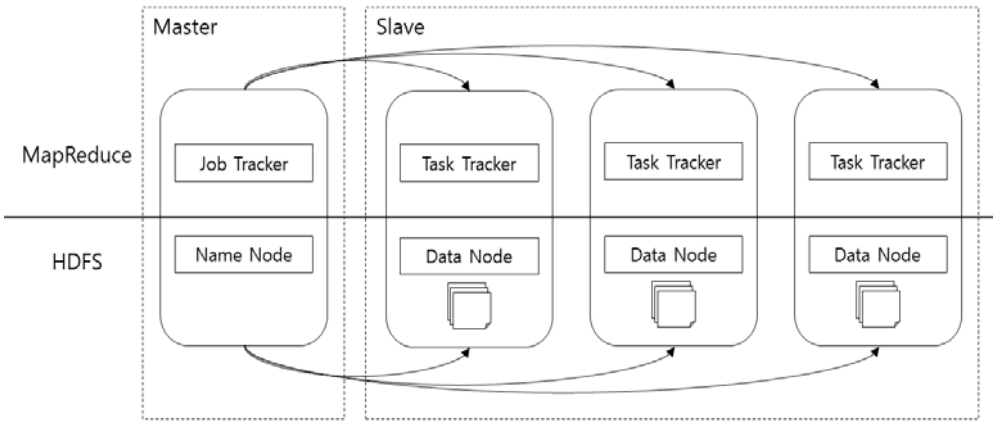

Fig. 1. Structure of a Hadoop cluster

MapReduce component in Hadoop controls job processing in distributed manner. JobTracker process in the master server allocates jobs to TaskTrackers in slave nodes and merges results received from the TaskTrackers. When a task process a file in HDFS, each block of the file can be processed by an allocated task in DataNode in each slave node. For example, assume a task `foo` tries to analyse 200Kbytes sized file `log.txt` stored in HDFS. Since the file is composed of 4 blocks with size 64Kbytes or less, replicas of the blocks are scattered over DataNodes. The blocks of the file can be located in 4 different DataNodes at maximum. JobTracker obtains the locations of nodes that has the replicas and allocates task `foo` to the nodes. In order to maintain availability of replicas, DataNodes send heartbeats to the NameNode every three seconds. If there is no heartbeat from a DataNode in ten minutes, NameNode creates new replica on another DataNode. Also the node receives jobs in the broken node.

In order to process the file, jobs are allocated to the slave nodes using *map* function provided by Hodoop. Result of the jobs are transmitted to the node where *reduce* function runs as shown in Fig. 2. Reduce function arranges the result and generates a result file. The number of nodes that run reduce function is determined by the number of blocks in the file and the number of reduce functions is decided by user parameter.

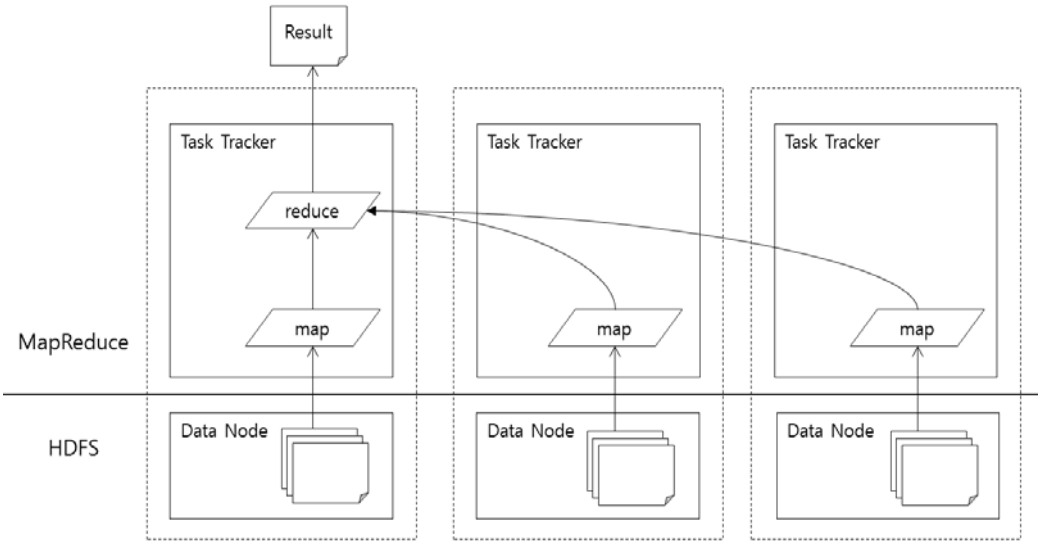

Fig. 2. Map and Reduce function in Hadoop

B. Chukwa

In order to analyse warning messages issued by working computers in a company, the messages should be collected and stored in HDFS. Chukwa is a package that collects logs in various computers and stores them in HDFS [16][17]. Chukwa is an open source subproject of Hadoop. A *collector* and multiple *agents* are constructed in order to collect logs into a big log file in HDFS. Each agent locates in a computer system where log messages are generated. An agent sends logs to a collector over HTTP because a collector is implemented as a Java servlet. A collector gathers logs from agents and stores them into a file in HDFS. There could be multiple collectors for fault tolerance of collector. Data sources that generate data to be collected is called *adaptors* in Chukwa. Since a Snort generates logs and the logs are collected by Chukwa, Snort processes are adaptors in the system.

C. Proposed Distributed Snort System

We propose a NIDS composed of 4 components: Snorts, Chukwa, HDFS, and MapReduce as shown in Fig. 3. Each Snort monitors a working server where it resides and sends alert logs to Chukwa agent as an adaptor. Chukwa agents send logs to a Chukwa collector, which writes logs into a single *sink file*. Periodically, the file is closed and the next sink file is created. The sink files are analysed by map functions in DataNodes where block replicas of the files are stored. Results of map functions are transmitted to a reduce function and a merged result is stored in a file as shown Fig. 4.

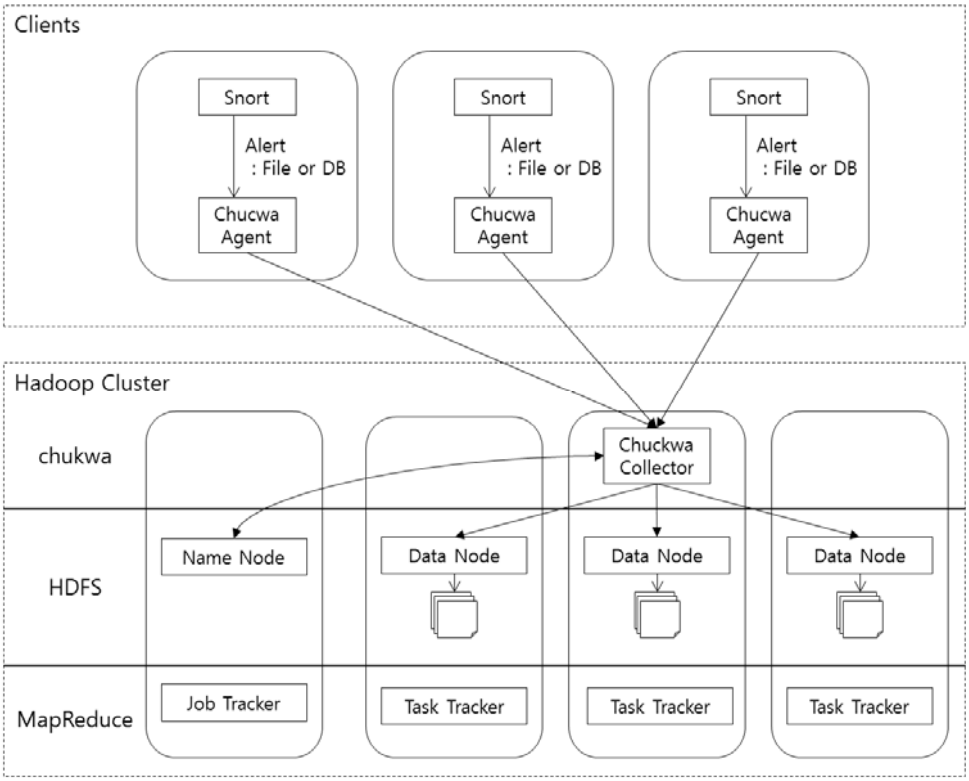

Fig. 3. Structure of proposed distributed Snort system

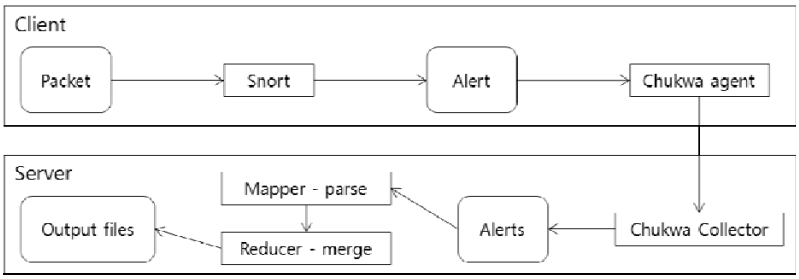

Fig. 4. Data flow in distributed Snort system

In each slave node where a duplicated block of the sink files is stored, a map function (or a mapper) parses warning messages in the block. It analyses the messages using regular expression. The regular expression is designed based on *snort-stat* [18], a warning message analysis tool programmed in Perl script language.

The map function reads packet data and checks whether it is a warning message. Since a normal warning message includes a string "`[**]`" as shown in Fig. 5, the map function checks whether there is the string "`[**]`" using regular expression "`\[\*\*\]`". In that case, the function reads ID of Snort rule that corresponds to the warning message and the warning contents using regular expression "`[\\[](\\d+):(\\d+):(\\d+)[\\]]\\s([^\\[]*)`". Next, it reads an attack type using regular expression "`Classification \\:([^\\]]*)`". It reads IP address and port number using regular expression "`([\d\.]+)[\:]*([\d]*) \s[\-\>]+\s([\d\.]+)[\:]*([\d]*)`".

```
03/01-15:12:36.895772 [**] [1:100000160:2] COMMUNITY SIP TCP/IP
message flooding directed to SIP proxy [**] [Classification: Attempted
Denial of Service] [Priority: 2] {TCP} 206.132.214.8:80 ->
192.168.2.5:1801
```

Fig. 5. An example of a warning message

The result of the analysis according to the previous process shows four types of information: priority, event type per IP address, and source and destination IP. The analysis from the message in Fig. 5 is shown in Fig. 6.

```
Priority
   2        1          COMMUNITY SIP TCP/IP message flooding directed to SIP proxy   1
Source
   206.132.214.8   COMMUNITY SIP TCP/IP message flooding directed to SIP proxy   1
Destination
   192.168.2.5     COMMUNITY SIP TCP/IP message flooding directed to SIP proxy   1
IP
   206.132.214.8   192.168.2.5          1
```

Fig. 6. Analysis result from a warning message

The results from map functions are copied to a reducer function, which calculates the count of each type and outputs it as a file. Commonly used functions `Outputformat`, `TextOutputFormat`, or `FileOutputTextFormat` output only one result per one job. Thus function `MultipleOutputFormat` is used in order to generate output data with multiple results.

IV. Experimental Results

In order to measure performance of the proposed system, it is constructed on a cluster computing system composed of 9 Pentium 4 3.2GHz computers. A master node that has a NameNode and a TaskTracker has 2Gbytes RAM. Maximum eight slave nodes that run DataNodes and TaskTrackers has 1Gbytes RAM. Hadoop 1.0.3 is installed on CentOS 3.5 operating system. Java development package JDK 1.6.0_29 is used. The nodes are connected by 1Gbps Ethernet network switch. Block size in HDFS is set to 64Mbytes and the number of duplica is set to 2.

We generated packet dump file based on network packet dump files proposed by Felix Uw Laboratory in University of California, Davis USA [19]. The files are composed of 29 files and total size of the dump files is about 2.42Gbytes. Since the size of warning message from the dump file is 459Kbytes, distributed processing of the warning message file in Hadoop is not efficient. Thus we duplicated the dump file while randomly changing an IP address of each message in the dump file. As the result, we got an expanded 2Gbytes sized warning message file. Analysis performance for various number of slave nodes are compared on the 2Gbytes size warning message file.

Table I shows execution time when a single computer system without Hadoop is used, and when the proposed distributed Snort system with various number of slave nodes is used. Since there is no overhead on resource or task management in a single computer system, single system has better performance than the proposed system with a single slave node does. As the number of nodes increases in the proposed system, execution time decreases. From two nodes system, the performance of the proposed system starts to exceed that of the single system. When the number of nodes in the system is 8, performance of the system is about 4.2 times better than that of the single system. Fig. shows that Gradient of speed up is almost linear, which means that the proposed system is highly scalable.

V. Conclusion

We proposed a distributed Snort system that gathers warning messages from multiple Snort processes which run on different servers, analyses the warning messages in parallel using MapReduce function presented by Hadoop. The proposed system that uses two or more slave nodes has better performance than a single computer system does. The proposed system with 8 slave nodes shows 4.2 times faster speed than that of a single computer system.

Although the proposed distributed Snort system is scalable, it lacks real-time property. In order to add real-time property to the proposed system, we have investigated HBase and Cloudera Impala. HBase is a database that supports real-time read/write access to big data, and Cloudera Impala is a database query package for HBase and Hadoop. We expect that merging the packages to the proposed distributed Snort system will improve the responsiveness of the system.

Acknowledgment

This paper was supported by Research Fund, Kumoh National Institute of Technology.

TABLE I
Comparison of execution time and speed up

Number of nodes	Execution time (sec)	Speed up based on single machine	Speed up based on Hadoop with single node
Single	1089.3	-	-
1	1835.1	59%	-
2	920.2	118%	199%
3	643.5	169%	285%
4	491.9	221%	373%
5	425.3	256%	432%
6	378.6	288%	485%
7	341.6	319%	537%
8	256.8	424%	715%

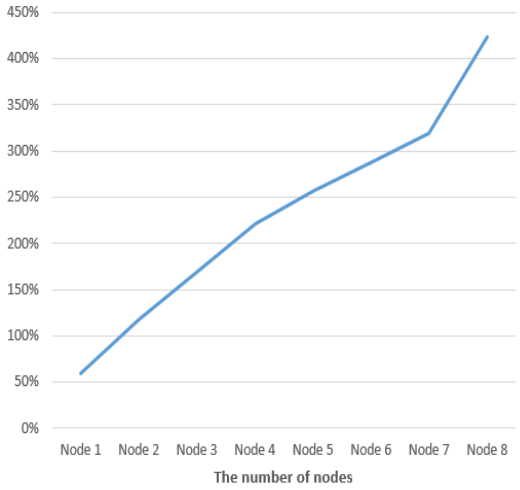

Fig. 7. Speed up for the number of nodes

REFERENCES

[1] Jay Beale, James C. Foster, Jeffrey Posluns, and Brian Caswell, *Snort Intrusion Detection 2.0*, Elsevier Inc., 2003.
[2] Toby Kohlenberg, Jay Beale, and Andrew R. Baker, *Snort IDS and IPS Toolkit*, Syngress Publishing, 2007.
[3] Alfred V. Aho and Margaret J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM* vol. 16, no. 6, pp.333-340, 1975.
[4] Rong-Tai Liu, Nen-Fu Huang, Chia-Nan Kao, and Chih-Hao Chen, "A fast pattern matching algorithm for network processor-based intrusion detection system," *2004 IEEE International Conference on Performance, Computing, and Communications*, pp. 271-275, 2004.
[5] C. Jason Coit, Stuart Staniford, and Joseph McAlemey, "Towards faster string matching for intrusion detection or exceeding the speed of Snort," *In Proceedings on DARPA Information Survivability Conference Exposition II, 2001, (DISCEX'01)*, vol. 1, pp. 367-373, 2001.
[6] Mike Fisk and George Varghese, *Applying fast string matching to Intrusion detection*, DTIC Document, 2002.
[7] Intel, "Supra-linear Packet Processing Performance with Intel Multi-core Processors," *Intel white paper*, 2006.
[8] Derek L. Schuff, Yung Ryn Choe, and Vijay S. Pai, "Conservative vs. Optimistic Parallelization of Stateful Network Intrusion Detection," *IEEE International Symposium on Performance Analysis of Systems and software*, pp. 32-43, April, 2008.
[9] Haiyang Jiang, Jianhua Yang, and Gaogang Xie, "Exploring and Enhancing the Performance of Parallel IDS on Multi-core Processors," *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, (TrustCom)*, pp.673-680, 2011.
[10] OISF homepage, http://www.openinfosecfoundation.org/.
[11] Shun-Fa Yang, Wei-Yu Chen, and Yao-Tsung Wang, "ICAS: An inter-VM IDS log cloud analysis system," *IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS2011)*, 2011.
[12] Dhruba Borthakur, "HDFS architecture guide," *Hadoop Apache Project (http://hadoop.apache.org/)*, 2008.
[13] Yeonhee Lee, Wonchul Kang, and Youngseok Lee, "A Hadoop-based packet trace processing tool," *Traffic Monitoring and Analysis LNCS 6613*, Springer, 2011.
[14] Apache™ Hadoop® homepage, http://hadoop.apache.org/.
[15] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler, "The Hadoop Distributed File System," *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp.1-10, 2010.
[16] Apache™ Chukwa homepage, http://incubator.apache.org/chukwa/.
[17] Jerome Boulon, Andy Konwinski, Runping Qi, Ariel Ribkin, Eric Yang, and Mac Yang, "Chukwa: A large-scale monitoring system," *The first workshop on Cloud Computing and its Applications (CCA'08)*, Chicago USA, October 22-23, 2008.
[18] Snort-stat Project Webpage, http://code.google.com/p/snort-state/.
[19] Felix Wu, "TCPdump Data Sets," http://www.cs.ucdavis.edu/~wu/tcpdump/, 2005.