A Network Traffic Collection System for Space Information Networks Emulation Platform

Chengpeng Kuang, Dongxu Hou, Qi Zhang, Kanglian Zhao(✉), and Wenfeng Li

School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, People's Republic of China
{mfl923032, mgl923075}@smail.nju.edu.cn, zhaokanglian@nju.edu.cn

Abstract. This paper presents a network traffic collection system for space networks emulation platform. The system uses virtualization technology and Elasticsearch database technology to collect the network traffic of gigabits per second. On this basis, a parallel processing method is proposed to further utilize the resources and improve the collection performance. Corresponding experiments are carried out to evaluate the collection performance of the system. The experimental results show that the system can collect massive and high-speed network traffic in real time and improve the collection rate.

Keywords: Space information networks emulation · Network traffic collection · Elasticsearch · Parallel processing

1 Introduction

The space information networks system is a network system which takes the space platform as the carrier to acquire, transmit and process spatial information. It is an indispensable guarantee to serve the national economic construction and national security. In order to verify the relevant space technologies and networking communication protocols, it is necessary to develop an emulation platform to emulate the space networks application scenarios.

The Nanjing University team has constructed a laboratory scale network emulation platform which applies virtualization technologies to emulate space information networks [1]. As shown in Fig. 1, the space networks emulation platform is divided into four planes, including the logic plane, the control plane, the data plane and the analysis plane. The logical plane is responsible for the actual network scenario. The control plane receives the parameters of the logic plane and creates the emulation scene. The data plane generates real data flow between network nodes to ensure the authenticity of emulation. The analysis plane defines the specific parameters of the network according to the actual situation of the network. And it monitors, statistics and feeds back the emulation results.

Q. Guo et al. (Eds.): WiSATS 2021, LNICST 410, pp. 217–225, 2022.
https://doi.org/10.1007/978-3-030-93398-2_22

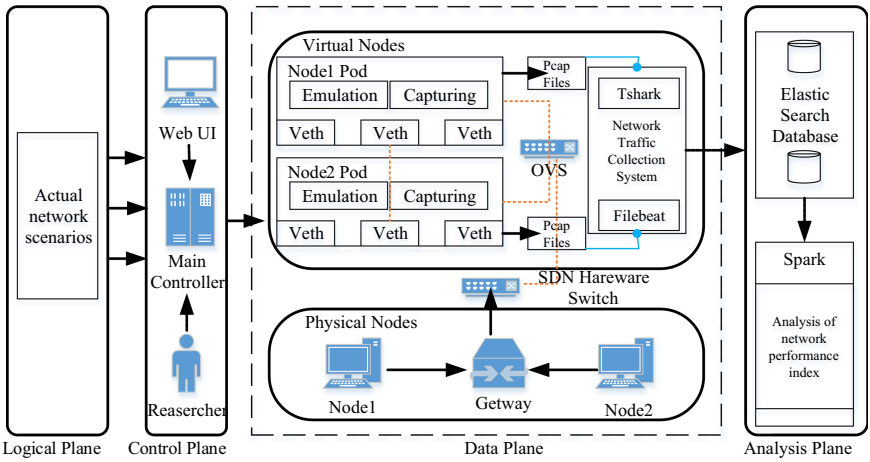

Fig. 1. Architecture of space information networks emulation platform.

In order to evaluate the performance of space networks scenarios, it is necessary to conduct in-depth analysis of traffic data in the emulation network environment. Network traffic collection system is an important part of the emulation platform. Under the limited platform resources, the network traffic collection system must be able to meet the collection, storage and analysis requirements of massive high-speed traffic, such as the acquisition of gigabit traffic per second. The system can analyze the specific protocol to facilitate the subsequent analysis. At the same time, the network traffic collection system should also have the functions of high stability, expandability, easy maintenance and easy recovery after failure.

Network collection is actually network measurement. Network measurement includes active measurement and passive measurement. Researchers have proposed a variety of network data collection methods and developed the corresponding system for different networks. The mainstream network collection system is roughly divided into three categories. The first is the network collection system based on SNMP protocol [2]. The basic principle is to exchange MIB table information between the management terminal and the managed terminal through the request answer mode of SNMP protocol [3]. However, SNMP cannot distinguish the proportion and distribution of different types of network services in the total network traffic, and the collected network traffic information is very rough. The second network collection system is based on flow technology [4]. In the network device that starts the flow mode, the data packets will be recorded and counted according to their own characteristics [5]. But this way of working is to use hardware technology to achieve fast forwarding of network data, which is not suitable for network emulation. The third collection system is based on the probe capture network [6], mainly through the use of data collection tools as a probe to monitor the specified network card [7], directly capture network messages for analysis, which is widely used to analyze the low-speed traffic between servers or workstations in simple network. The above methods cannot meet our needs. For the massive high-speed traffic collection of the emulation platform, container technology and big data

technology are used to collect and store all data packets in real time, as well as parse and filter the traffic at the same time.

This paper uses virtualization container technology to share network namespace, and uses Tshark to capture and parse network traffic [8]. Using big data technology and Elasticsearch database to realize data storage function, a method of parallel collection is proposed. At the same time, reasonable allocation of system resources, distributed deployment, load balancing through file sharing, network traffic collection performance of the emulation platform is further studied.

The rest of this paper is organized as follows. In Sect. 2, we will introduce the realization of network traffic collection system architecture. In Sect. 3, we introduce the optimization of system parameters and the parallel processing method. In Sect. 4, we do experiments with the parallel processing method and analyze the experimental results. At last, the conclusion is given in Sect. 5.

2 Architecture of Network Traffic Collection System

The network traffic collection system is integrated in the data plane of the space networks emulation platform. First of all, container technology and big data technology are used. Based on the ELK architecture [9], the collection system can capture, parse and store network traffic data. Secondly, in order to further improve the collection performance, the system parameters are optimized and a parallel processing method is proposed.

The architecture of our designed system is illustrated in Fig. 2, which can be divided into network traffic capturing module, network traffic parsing module, network traffic filtering module, network traffic storage module and visualization module.

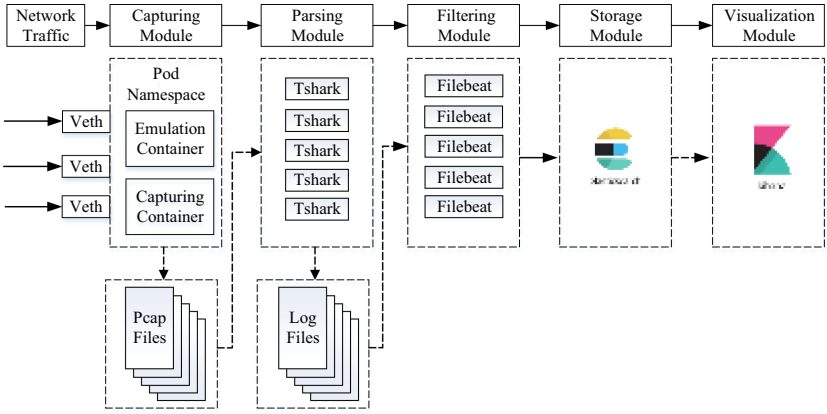

Fig. 2. Architecture of network traffic collection system.

Network Traffic Capturing Module: This module is to ensure that the network traffic data is completely captured without loss of packet, so as to achieve flexible and

configurable network capture. Each virtual node of the emulation platform includes an emulation node and a capture node. The emulation container is responsible for the emulation of network protocols. The capture container is responsible for the capture of network traffic. Docker containers share network resources [10], and the network traffic processed by the emulation container is visible to the capture container. To provide extensibility, the container is managed by orchestration, for example, Kubernetes [11].

Network Traffic Parsing Module: All captured packets have a lot of redundant information. This module implements the parsing function. It parses out the fields we need, such as source IP, destination IP, packet length, protocol type, and other useful information, and writes them to the specified log file. For example, Tshark is primarily used for packet capture and analysis in a command-line environment, where we listen on container ports and implement parsing.

Network Traffic Filtering Module: This module is mainly responsible for filtering information according to rules. The information is written into the database in the form of fields to facilitate the direct use of queries after taking out. Filebeat is used to collect log data [12]. It realizes the function of preprocessing and can continuously collect new contents of log files and write them into the database.

Network Traffic Storage Module: This module stores useful information for subsequent analysis. It requires full utilization of storage resources. Data is easy to deploy, manage, extend, and query, such as Elasticsearch databases [13].

Visualization Module: Data visualization helps effectively understand and analyze massive amounts of data, which is at the end of the system. Kibana [14], with the help of the powerful search engine Elasticsearch, provides columns that can be used for retrieval and can realize analysis and positioning of massive data.

The above modules work together to finally store traffic data in the database in JSON format. Data can be written and displayed in real time.

3 Optimization Schemes

3.1 Optimized Configuration of System Parameters

We hope that the network traffic filtering and the write rate of the enclosure can keep up with the conversion resolution rate of traffic capture module and resolution module Tshark. However, for the default Filebeat and Elasticsearch cluster configuration parameters, the index rate of data writing to the Elasticsearch database cannot meet the requirements of emulation. While Filebeat and Elasticsearch have no fixed optimization method, so we need to study how to configure parameters and improve the speed. Emulation experiments were carried out to discuss the best settings of Filebeat and Elasticsearch in the system. Performance is judged by the number of packets processed per second.

The experimental scenario is shown in Fig. 3. Two emulation nodes are used. Each node is composed of an emulation container and a traffic collection container. They share a network namespace. The two nodes send 1 Gbps packet traffic. The collection

container collects, captures and filters the traffic, and writes it to Elasticsearch database. By recording the rate of each operation step, we observe the performance under different parameter configurations.

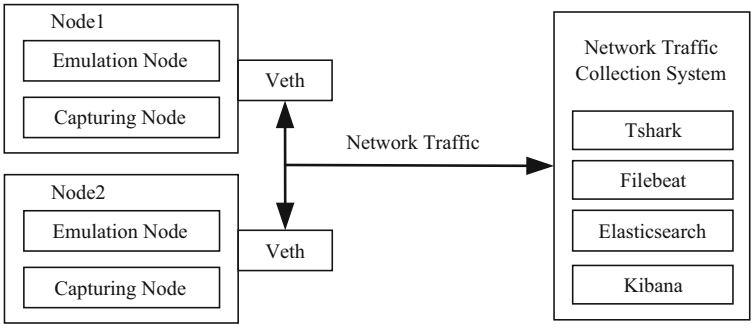

Fig. 3. Topology of test experiment.

Elasticsearch is a distributed database, so the indexing of Elasticsearch is usually broken into different parts. The data distributed on different nodes is called shards. The number of primary shards affects write performance. There are two ways to refresh when writing data. Synchronization means that memory data is brushed to disk at fixed intervals.

Figure 4 shows the impact of different primary shards and different synchronization modes on performance. Performance improves as the number of primary shards increases. Compared with synchronous writing, asynchronous writing has a great improvement in write performance.

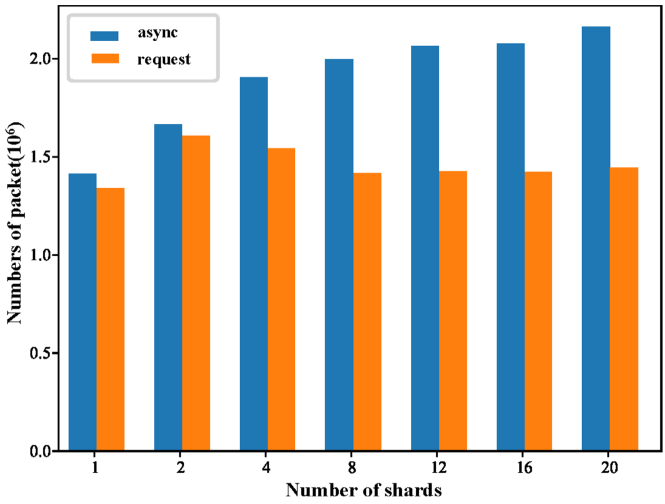

Fig. 4. Number of shards and writing method.

Table 1 shows the performance impact of different replica shards. Replica shards are mainly used to backup and restore data to improve the security and reliability of the system. We can temporarily set the replica shards to 0 when a large amount of data is written, and then restore it later. This not only improves the writing performance, but also ensures the security of the system.

Table 1. The effect of the number of replicas on performance

Numbers of replicas	Elasticsearch database writing speed (packet/s)
0	36123
1	32023
2	30379
3	30175

The results show that when the number of primary shards is 8 and the number of replica shards is 0, the write rate reaches a reasonable value.

3.2 Real-Time Tracking Performance

Based on the above optimized configuration, it can be seen from the results listed in Table 2 that the data rate written to Elasticsearch database by the traffic collection system can keep up with the capture and resolution rate of Tshark.

Table 2. Real-time tracking performance.

Time (s)	Capturing speed (packets/s)	Elasticsearch database writing speed (packet/s)	Difference
30	33902	33792	0.62%
60	33211	33343	0.39%
90	33413	33112	0.91%
120	33905	34091	0.55%

3.3 Parallel Processing

Under the limited hardware resources, with the increase of the number of emulation nodes and network traffic data, the performance of serial data collection has been unable to meet our requirements. The parallel data collection method is adopted to improve the collection performance. At the same time, distributed deployment is carried out in order to make reasonable use of system resources.

As shown in Fig. 5, all network traffic captured by a Tshark is serially stored in a log file and monitored by a Filebeat file. This method does not make full use of the resources of the server, and when the traffic scale is small, the rate of serial collection method can meet the demand. So we collect traffic in segments based on a fixed time period. Every once in a while, we turn on a Tshark capture and parse the traffic in the

current time period, and turn on a Filebeat to monitor and filter log files to achieve a parallel effect. It saves the parsed data in JSON format to log files and databases. In this way, in subsequent analysis, the queried data can be used directly without additional processing, which saves processing time.

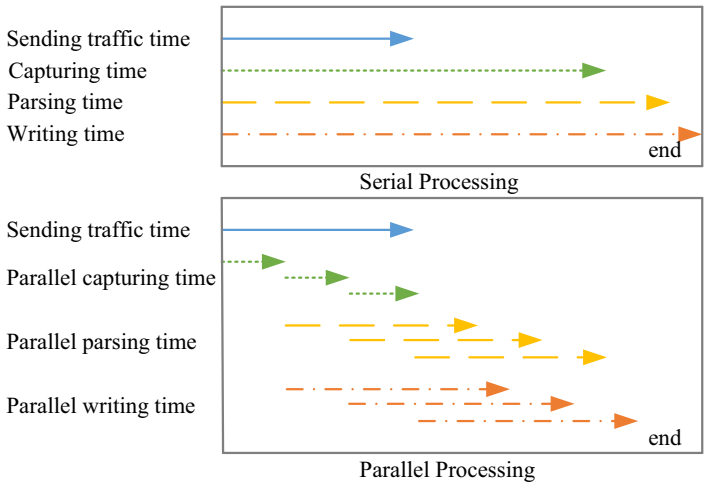

Fig. 5. Serial processing and parallel processing.

At the same time, as shown in Fig. 6, Tsharks and Filebeats are deployed in a distributed way. Tshark is deployed on a server. It shares the log files on another server via NFS files. Files are monitored and filtered by Filebeat. This kind of distributed deployment makes reasonable use of server resources and reduces the contention of CPU and other resources between traffic collection system and emulation system.

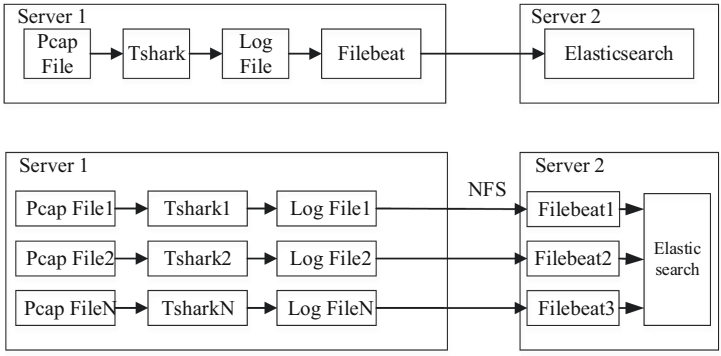

Fig. 6. Distributed deployment of Tshark and Filebeat.

4 Experiment Results

4.1 The Effect of the Number of Tsharks on Performance

The amount of parallel processing is one metric. The number of different Tsharks has an impact on write performance. Table 3 shows that with the increase of the number of Tsharks, the write rate of a single thread will be reduced. But the overall write rate will increase greatly due to the increase of the total number. It can be observed that limited by hardware resources, performance cannot be infinitely linear improved. When the number of Tsharks is 20, the system resource bottleneck is probably reached.

Table 3. The effect of the number of Tsharks on performance.

Numbers of Tsharks	Total speed (10^3*packets/s)	Single speed (10^3*packets/s)
1	31	31
2	60	30
4	115	28.75
6	165	27.5
10	232	23.2
20	302	15.1
30	324	10.8

4.2 Comparsion of Writing Time Between Different Methods

Figure 7 shows the time comparison required for single serial and multi-channel parallel processing when processing the same traffic data. The results show that the parallel mode has improved performance and nearly doubled.

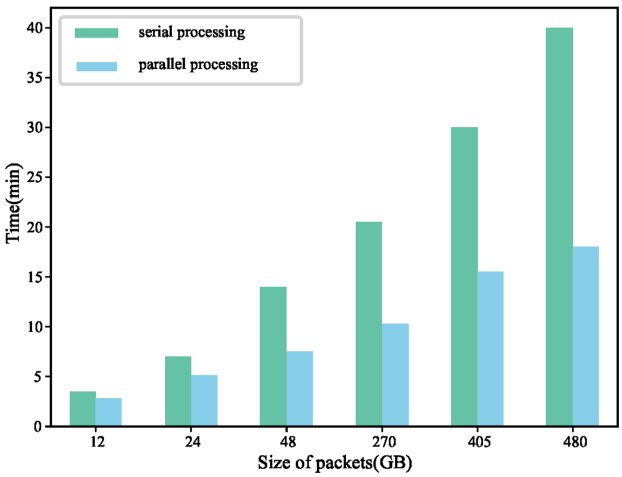

Fig. 7. Comparsion of writing time between different methods.

5 Conclusion

This paper introduces a network traffic collection system based on container technology and Elasticsearch database. Based on the space networks emulation platform, we evaluate the performance of the system to collect massive high-speed network traffic. The system can collect gigabits network traffic per second. Moreover, the system parameters are optimized and a parallel processing method is proposed. The experimental results show that the system can collect massive high-speed network traffic in the space networks emulation platform in real time under the condition of limited hardware resources.

Acknowledgements. This work is supported by the 13th Five-Year Civil Aerospace Technology Pre Research Project, the Fundamental Research Funds for the Central Univesities under Grant 021014380187 and the National Natural Sciences Foundation of China under Grant 62131012.

References

1. Lu, T., Zhang, W., Ni, X., et al.: A scalable network emulation architecture for space internetworking. In: 2016 IEEE International Conference on Communication Systems (ICCS). IEEE (2016)
2. Case, J.D.: Simple Network Management Protocol (SNMP). RFC (1990)
3. Stallings, W.: SNMP and SNMPv2: the infrastructure for network management. IEEE Commun. Mag. **36**(3), 37–43 (1998). https://doi.org/10.1109/35.663326
4. Kapoor, R.D., Calabrese, A.D., Dubey, R.K., et al.: Sample netflow for network traffic data collection. US (2007)
5. Zhao, X.F., Yi-Dong, X.U.: Monitoring system on campus network based on NETFLOW and SNMP. Comput. Technol. Dev. (2008)
6. Harvey, J.P.: Network sniffer for monitoring and reporting network information that is not privileged beyond a user's privilege level. US (2000)
7. Wang, Y., Zhang, N.: Principles and implementations of network sniffer. Appl. Res. Comput. (2003)
8. Tshark. http://www.wireshark.org/docs/man-pages/Tshark.html
9. Bajer, M.: Building an IoT data hub with Elasticsearch, Logstash and Kibana. In: 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW). IEEE Computer Society (2017)
10. Boettiger, C.: An introduction to Docker for reproducible research. Oper. Syst. Rev. **49**(1), 71–79 (2015)
11. Bernstein, D.: Containers and cloud: from LXC to Docker to Kubernetes. Cloud Comput. **1** (3), 81–84 (2014)
12. Ya-Rong, Z., Jin-Gang, Y.U.: Analysis system based on filebeat automated collection of kubernetes log. Comput. Syst. Appl. (2018)
13. Dhulavvagol, P.M., Bhajantri, V.H., Totad, S.G.: Performance analysis of distributed processing system using shard selection techniques on Elasticsearch. Procedia Comput. Sci. **167**, 1626–1635 (2020)
14. Andreassen, O., Alicia, D., Charrondière, C.: Monitoring mixed-language applications with elastic search, Logstash and Kibana (ELK) (2015)