Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 7, July 2018, pp. 1383-1390
DOI: 10.23940/ijpe.18.07.p1.13831390

Data Packet Processing Model based on Multi-Core Architecture

Xian Zhang[a,b], Dong Yin[a,b,*], Taiguo Qu[a,b], Jia Liu[a,b], and Yiwen Liu[a,b]

[a]*School of Computer Science and Engineering, Huaihua University, Huaihua, 418008, China*
[b]*Key Laboratory of Intelligent Control Technology for Wuling-Mountain Ecological Agriculture in Hunan Province, Huaihua, 418008, China*

Abstract

According to the characteristics of pipeline structure and multi-core processor structure for packet processing in network detection applications, the horizontal-based parallel architecture model and tree-based parallel architecture model are proposed for packet processing of Snort application. The principle of a tree-based parallel architecture model is to use pipelining and flow-pinning technology to design a processor that is specifically used to capture data packets, and other processors are responsible for other stages of parallel processing of the data packets. The experimental comparison and analysis show that the tree-based parallel architecture model has higher performance on the second-level cache hit ratio, throughput, CPU utilization, and inter-core load balancing compared to the horizontal-based parallel architecture model for packet processing of Snort application.

Keywords: multi-core; network detection; data packets; horizontal-based model; tree-based model

(Submitted on March 20, 2018; Revised on April 12, 2018; Accepted on June 23, 2018)

1. Introduction

With further development of processor technology, the PC based on multi-core architecture has become more popular. The processing performance has improved, and the multi-core processor architecture also provides a hardware platform for parallel processing of data streams for network detection applications. However, it is very difficult for parallel processing of fine-grained applications on the multi-core processor architecture platform, because it not only needs the hardware to support inter-processor communication and synchronization, but it also needs to detect the parallel processing and application of software [9,11]. Fortunately, for the network detection application, we can decompose the packets processing into pipeline hierarchy, so that the data packets processing can realize a parallel processing based on flow. For the implementation of packet processing based on pipelined parallel architecture on the multi-core processor platform, it needs to balance the packet load balancing, resource sharing, local data cache, communication and synchronization between processor cores. [3,10].

For the design and research of the data processing model for network detection application on the multi-core processor architecture platform, an open source network intrusion detection software called Snort is used to analyze the packet processing. The pipeline structure of packet processing in the Snort system can be divided into several stages, such as packet capture, packet decoding, packet preprocessing, detection engine, and result output. [6]. By designing the data packet processing model of network detection application on multi-core processor architecture, the processing performance of the network detection application can be effectively improved, taking full advantage of the parallel processing of the multi-core processor architecture.

In this article, in view of the flow line structure of packet processing for open source network intrusion detection system Snort, horizontal-based parallel architecture model and tree-based parallel architecture model are designed for packet processing on the multi-core processor architecture platform. The performance indexes of packet processing performance and CPU utilization are tested by the horizontal-based parallel architecture model and tree-based parallel architecture model using Snort software.

* Corresponding author.
E-mail address: zhangxian314@sina.com

2. Data Processing Model Architecture Design

2.1. Data Processing of Network Detection Application

Network detection is used to capture the data packet information in the network, and then, according to the content of the data packet, the characteristics of the network link and the real-time operation of the network can be understood. The measurement process of a specific network detection application can be divided into several stages. (1) At a network detection point, the network card is set as a hybrid mode, and the corresponding packet capture function API is used to capture the data packet flow in the network of observation points; (2) The captured packets flow can be sampled and filtered according to the bandwidth of the link bandwidth and the requirements of the detection application; (3) The data packet stream is classified to make the corresponding data packets map to the same class flow record, and then the corresponding flow records are passed to the upper-level application; (4) The corresponding data flow records are processed according to the corresponding applications program [4,12].

The packet processing of network detection application is shown in Figure 1.

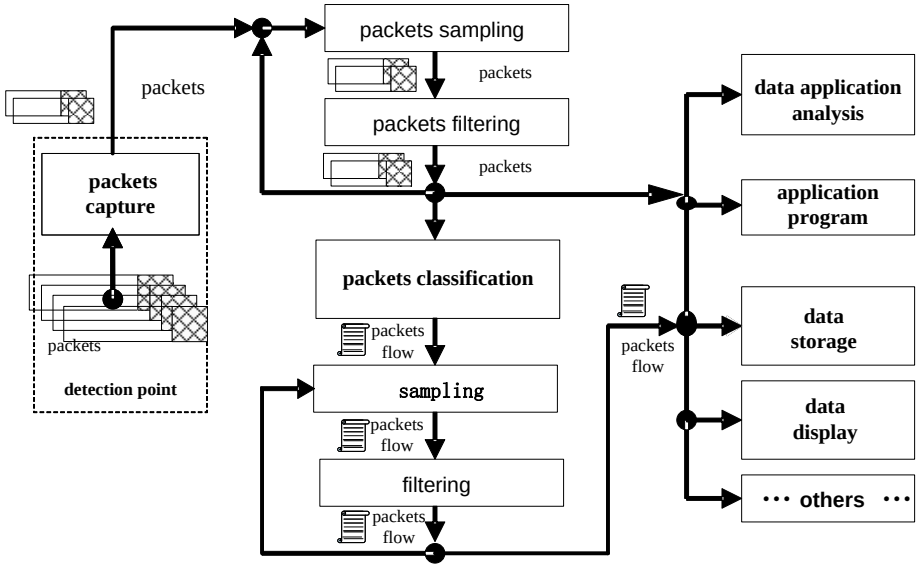

Figure 1. Packet detection and process

2.2. Network Intrusion Detection Application Snort

Snort is an open source network security solution written by Marty Roesch of Sourcefire Company. Today, it has developed into a multi-platform system used for real-time traffic analysis and network IP packet monitoring. Snort is a lightweight intrusion detection system based on Libpcap. “Lightweight” in this context has two meanings: first, it can be easily installed and configured on any node of the network, and it will not have much impact on network operation; second, it should have the capability of cross-platform operations, and the administrator can use it to make real-time security responses in a short time by modifying the configuration. Snort can run on multiple operating system platforms.

Snort software includes three working modes: sniffer, packet recorder, and network intrusion detection system. Under the sniffer mode, the data packets captured from the network are read out and the packet information is displayed on the terminal. The data packet record mode is to record the packet information on the memory device. Network intrusion detection mode is to analyze the collected information. The intrusion behavior is then found. Snort application can analyze the captured packets according to certain rules to determine whether there is a network attack, and then, according to the test results, take certain protective actions.

The Snort system consists of four basic modules: packet sniffer, preprocessor, detection engine, and alarm output module. All of these software modules are integrated with Snort through the plug-in mode, which makes it easy to expand, and developers can add their own modules to extend Snort's functionality. All of these sub-modules are based on the packet intercept library function interface Libpcap, which provides a portable packet capture and filtering mechanism for them [2,5]. The flow diagram of the pipeline structure of packet processing for network intrusion monitoring system is shown in Figure 2.

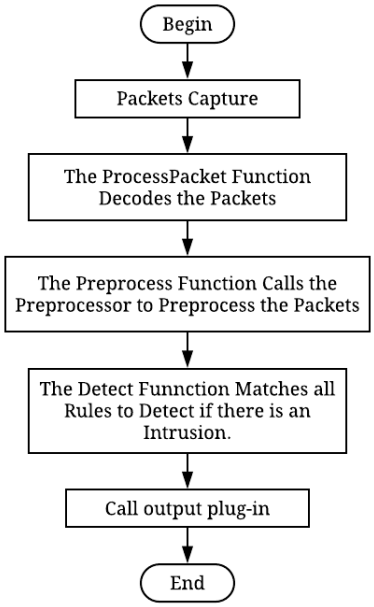

Figure 2. Packet processing of Snort system

The data packet processing procedure is called by the main program in the InterfaceThread function. When the data packet is processed, first the network protocol resolver is invoked to deal with and analyze the captured packets, and the analysis results are saved to the data structure table. The ProcessPacket function calls the DecodeEthPkt function to decode the Etheric frame, while the DecodeIP function decodes the IP protocol and the DecodeTCP function decodes the TCP protocol. Then, the system calls the preprocessor Preprocess function and the main engine Detect function. The main engine Detect function judges the intrusion detection behavior with the given rules according to the order of the data packets captured by the main engine function. If the captured packet is consistent with a specified test rule, apply the network intrusion detection according to the rule with the previously defined response method and the initialization output module, and then choose a certain way to record the results and make the corresponding alarm operation [1].

2.3. Design the Data Processing Model Architecture

The Network Intrusion Detection Application Snort is a program implemented in C language that runs on Unix systems or Linux systems. In order to run the application in parallel on multicore processor platform, we use the POSIX threading library (pthreads), and pthreads defines a set of C program language types, functions and constants that are implemented with the pthread.h header file and a thread library. Pthreads allows programmers to write programs to create and destroy threads as needed, as well as to implement threads using various synchronization primitives and parallel operations.

In the field of network detection, we can make use the characteristics of the network application and multi-core processor architecture to implement the parallel serialize processing of network detection application on the multi-core processor architecture platform. This improves the data processing ability of network detection applications and system performance. There are two inherent advantages of network detection application to realize parallel operation: first, the packet processing of network detection application has a natural pipeline hierarchy so that it can be conveniently organized into a pipeline hierarchical model. Another advantage is that packets belonging to different streams can be processed in parallel in different processor cores to reduce additional inter-core communication and synchronization overhead. Implementing a pipeline structure on a multi-core processor architecture is still a challenging task. First, the network detection application itself is a memory intensive and I/O intensive application, which could further aggravate memory latency between the multi-core processor architecture and computing power of inconsistency. On the other hand, the synchronization and communication between kernels must be handled by software or instruction, which will cost more CPU resources and worsen the system processing performance [7,8].

In this article, we proposed the horizontal-based parallel architecture model and tree-based parallel architecture model of Snort application on multi-core processor architecture, using the pipeline hierarchy of packet processing of the Network Intrusion Detection Application Snort. The experiment is designed in the case of small TCP connection flow and large TCP

connection flow. It tested the second level cache hit ratio, throughput, CPU utilization, and load balance of the Snort application in the above two kinds of data packet processing model.

2.3.1. Design the Data Processing Model Architecture

On a multi-core PC platform with a quad-core processor, the packet flow processing of network detection applications is usually run in parallel to the four processor cores. The horizontal-based parallel architecture model is adopted in the multi-core processor platform for the packet stream processing of the Network Intrusion Detection System Snort. Data packet processing of Snort is run on four processor cores using horizontal-based parallel pipeline structure. Every processor core thread runs the same packet pipelining process for network intrusion detection application Snort. Packets from the network card device are read, the reading packet is decoded and reorganized by the function, and the preprocessor is called to preprocess the packet. Then, the function is called to match all the rules to detect an intrusion behavior, and the output plug-in (log module, alarm module, record access module) is also called.

Packet processing flow of the Network Intrusion Detection Application Snort with the horizontal-based parallel architecture model on multi-core processor platform is shown in Figure 3.

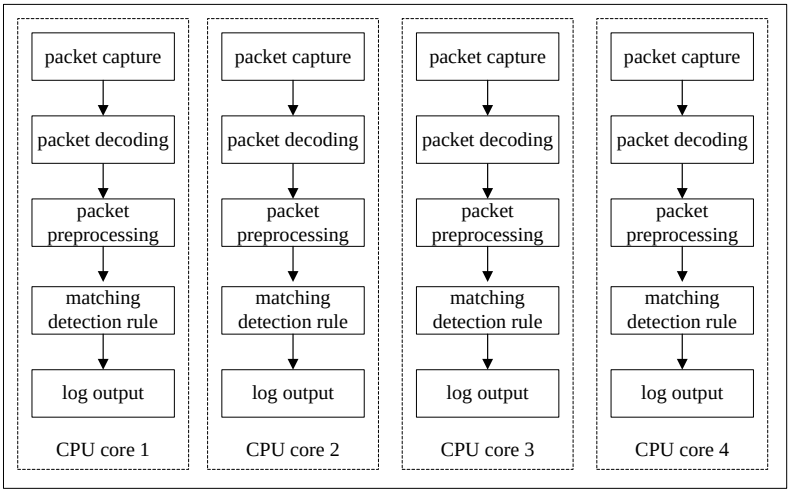

Figure 3. Horizontal-based parallel architecture model

2.3.2. Tree-based Parallel Architecture Model

In order to realize the parallelism of network detection application in flow-level, flow-pinning can be used to determine if flow belonging to the same TCP packets can be processed by the same CPU cores. Since each TCP connection consists of two TCP one-way flows, we bind these two TCP one-way flows into a TCP connection to the same CPU core. Thus, when there is access to the same TCP connection flow, we do not need to consider the resource consumption of inter-core communication and synchronization mechanisms. Another important consideration for multicore packets processing between multi-core is the load balance between processor core. Here, a data flow load balancing method is used to assign the packet stream to the corresponding CPU kernel.

For the packet processing of the Network Intrusion Detection Application Snort, we use the pipelining and flow-pinning technology to make each phase of packet processing for Snort applications respectively run on the four cores on the multicore processor platform. In the tree-based parallel architecture model, we used the pthreads thread library programming to implement the packet capture phase of Snort application running on a single CPU kernel (Core 1), and the corresponding hash algorithm is run in core 1 to classify the captured packets into several queues according to the flow. Then, each queue is mapped to the other three corresponding processor cores. According to the corresponding TCP flow classification, make other CPU cores respectively perform other phases of packet processing of the Snort application. This reads packets from the network card device, then the reading packet is decoded and reorganized by the function, the preprocessor is called to preprocess the packet, the function is called to match all the rules to detect an intrusion behavior, and the output plug-in (log module, alarm module, record access module) is called. Here, the flow classification adopts the principle of CPU connection affinity, and the same TCP connection stream is assigned to the same CPU kernel to avoid unnecessary inter-nuclear communication and synchronization overhead. Therefore, pipelining technology can improve the cache reference of local data, the corresponding cache hit rate, the packet processing ability, and overall performance of the network detection application.

The tree-based parallel architecture model of packet processing of the Network Intrusion Detection Application Snort on the multicore processor platform is shown in Figure 4.

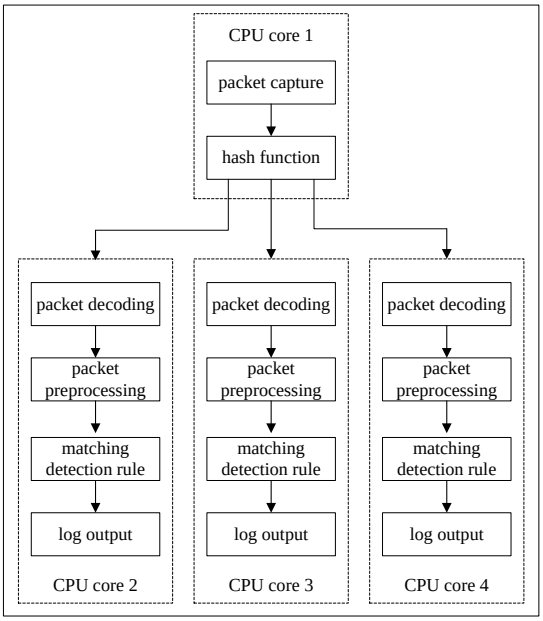

Figure 4. Tree-based parallel architecture model

An important phase in tree-based parallel architecture model is the classification of the packet streams, where the packet classification hash function is used to classify packets into each CPU core. The hash function distributes TCP flow as evenly as possible to each CPU core and reduces load sharing and computational complexity, while also considering the affinity of packet streams to CPU cores. The hashing algorithm uses TCP's five-tuple (source address, destination address, source port number, destination port number, protocol) to classify.

```
hash = (tcp->th_sport) ^(tcp->th_dport) ^(ip->ip_src.s_addr) ^(ip->ip_dst.s_addr) ;
hash=hash %PRIME_NUMBER;
return lookup_table[hash];
```

The result of this exclusive or operation is divided by a prime number, and the remainder is the index of the hash table, where 251 is chosen as the prime number. Using the hash function to classify packets make the TCP flow into each of the three CPU cores. Packets from the same TCP flow are assigned to the same CPU kernel, and this can improve the reference rate and cache utilization of local data.

3. Experimental Design and Verification

3.1. Experimental Design

We must test the packet processing performance of the Network Intrusion Detection Application Snort with the horizontal-based parallel architecture model and tree-based parallel architecture model on the multi-core processor platform. In this paper, experiments are designed to test the second level cache hit ratio, throughput, CPU utilization, and load balancing.

In the experimental environment, two PCs with quad-core processors were used to build a test platform, one as a packets generator and the other as a testing machine. The basic configuration information of the PC is as follows: 4 core Intel Xeon processor, Intel E7520 memory controller, 4GB DDRII memory, and gigabit Ethernet card. The tcpreplay software is installed on the packets generator to replay the packet tracking file.

Details of the basic configuration in the experimental platform are shown in Table 1.

In the experiment, Tcpreplay software was used to replay two-packets tracking files, and the test environment was set up to test the performance of the horizontal-based parallel architecture model and tree-based parallel architecture model. A replayed packet tracking file is from the database server, with only a relatively small number of TCP connections (about

175). Another replayed packet tracking file comes from the national laboratory for network research (NLANR), and it has a relatively large number of TCP connections (about 25,000). Install the Network Intrusion Detection Application Snort on the test machine, modify the Snort application to run a multithreaded, and specify the packets of Snort to the specified processor core. Install the performance analyzer VTune on the test machine, and use it to test the second-level cache hit ratio. VTune can collect information in real time without affecting the running of the Snort application.

Table 1. Configuration information of experimental platform

Configuration information	Specifications
CPU frequency	Intel Xeon E3-1231 v3 @3.4GHz
Memory	DDRII 4GB
PCI bus	PCI-X 133 MHz 64b
Network card	RTL8168/8111 PCI-E Gigabit Ethernet NIC
Operating system	ubuntu-16.04
Kernel version	Linux-4.6.4
NIC driver	intel e1000-8.0.16(ixgb-2.0.6)
Snort	snort-2.9.11.1
Tcpreplay	tcpreplay-2.3.5
VTune	vtune_amplifier_xe_2015

3.2. Small Flow Pattern Testing

In order to evaluate the performance of packet processing for the Network Intrusion Detection Application Snort based on the horizontal-based parallel architecture model and tree-based parallel architecture model, the experiment was designed to send a small number of TCP connections (about 175) trace files. Respectively, test the level 2 cache average hit ratio and throughput of the horizontal-based parallel architecture model (as shown in Figure 3) and tree-based parallel architecture model (as shown in Figure 4) on the multi-core processor architecture platform. The experimental results are shown in Table 2.

Table 2. Test results under small flow

Experimental mode (175 TCP connection)	Throughput (Mb/s)	Level 2 cache average hit ratio
Horizontal-based parallel architecture model	543	86%
Tree-based parallel architecture model	576	94%

As shown by the table, for a small number of TCP connections, the tree-based parallel architecture model is slightly higher in both throughput and level 2 cache average hit ratio compared to the horizontal-based parallel architecture model.

3.3. Large Flow Pattern Testing

The second experiment was designed to send a large number of TCP connections (about 25000) trace files. Respectively, test the level 2 cache average hit ratio and throughput of the horizontal-based parallel architecture model (as shown in Figure 3) and tree-based parallel architecture model (as shown in Figure 4) on the multi-core processor architecture platform. The experimental results are shown in Table 3.

Table 3. Test results under large flow

Experimental mode (25000 TCP connection)	Throughput (Mb/s)	Level 2 cache average hit ratio
Horizontal-based parallel architecture model	29	42%
Tree-based parallel architecture model	180	70%

It can be seen that, for a large number of TCP connections, the tree-based parallel architecture model has a much higher throughput than the horizontal-based parallel architecture model (more than 6X greater). The level 2 cache average hit ratio is also much higher, with an increase of about 30%. The experimental results show that the tree-based parallel architecture model of the Network Intrusion Detection System Snort can significantly improve the packet processing capability on the multicore processor.

3.4. CPU Performance Testing

To evaluate the CPU performance and inter-core load balancing performance of the packet processing of the Network Intrusion Detection Application Snort with the horizontal-based parallel architecture model and tree-based parallel architecture model on the multi-core architecture platform, these experiments are designed to test the CPU utilization and load balancing performance under the condition of sending a large number of TCP connections (about 25,000). The experimental results are shown in Figure 5.

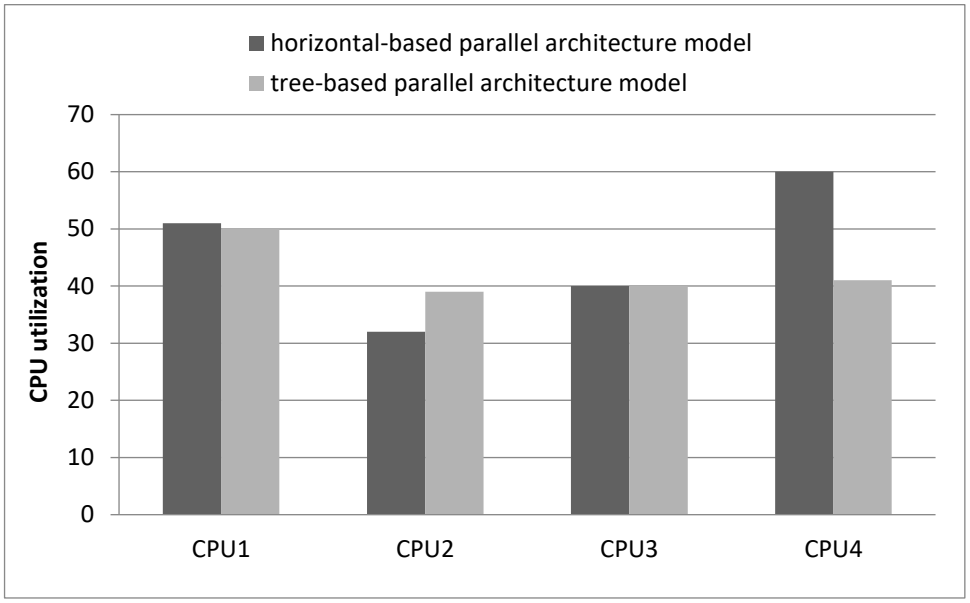

Figure 5. CPU utilization and load balancing performance testing.

From the experimental results, it can be seen that the packet processing with tree-based parallel architecture model performs better than that with horizontal-based parallel architecture model in terms of CPU utilization performance, especially CPU core load balancing. This is because the Network Intrusion Detection Application Snort requires a relatively large amount of CPU resources during the packet capture phase, and the subsequent data processing phase requires relatively few resources. Thus, the tree-based parallel architecture model can improve system performance.

4. **Conclusions**

In this paper, we first analyze the data processing of network detection and the data packet processing of the network intrusion detection system Snort. According to the characteristics of packet processing of network detection application Snort on the multi-core processor platform, we proposed the horizontal-based parallel architecture model and tree-based parallel architecture model by using pipelining and flow-pinning technology. Through comparison and analysis of the performance indicators of level 2 cache hit ratio, throughput, CPU utilization, and load balancing on data packet processing with the horizontal-based parallel architecture model and tree-based parallel architecture model, the experimental results show that the tree-based parallel architecture model has higher performance than the horizontal-based parallel architecture model. Therefore, this model can improve data processing ability and reduce system resource consumption of network detection applications, offering practical value and potential for further applications.

Acknowledgements

The research was supported by the Education Department of Hunan Province (17C1267) and the Huaihua Science and Technology Agency Project (2017N2207). The research was also supported in part by grants from the Huaihua University Project (HHUY2017-13).

References

1. L. J. Feng. “Parallel Similarity Join of Multi-core”. Computer Technology and Development, 2017, 27(7):43-47.
2. L. T. Han, H. L. Liu, Q.L. Kong, and F. L. Yang. “Parallel Algorithm for Hill-shading under Multi-core Computing Environment”. Journal of Computer Applications, 2017, 37(7):1911-1916.
3. R. L. Ling, J. F. Li, and Li Dan. “Realtime Capture of High-Speed Traffic on Multi-core Platform”. Journal of Computer Research and Development, 2017, 54(6):1300-1313.
4. S. Q Liu, Z. K. Chen, K. Y. Jiang, Y. M. Hu, and H. Xu. “Hybrid Multi-kernels Estimation Method for Data Imputation”. Mini-micro Systems, 2017, 35(7):1523-1527.
5. X. Li, H. Li, C. Gan, and H. D. Zhu. “Comparative Research on High-Performance Network Packet Processing Methods in Servers”. Computer Applications and Software, 2017, 34(11):177-183.
6. Z. Q. Luo, K. Huang, D. F. Zhang, H. T. Guan, and G. G. Xie. “A Many-Core Processor Resource Allocation Scheme for Packet Processing”. Journal of Computer Research and Development, 2014, 51(6):1159-1166.
7. L. Pei, and Q. Liu. “Online Learning Algorithm based on Multi-task and Multi-kernel for Stream Data”. Application Research

of Computers, 2019, 36(3):1001-1008.
8. C. T. Rong, Y. Y. Li, L. J. Feng, and J. M. Wang. “Fine-Granular Parallel Set Similarity Join based on Multicores”. Chinese Journal of Computers, 2017, 40(10):2320-2337.
9. J. Sreeram, and S. Pande. “Exploiting Approximate Value Locality for Data Synchronization on Multi-core processors”. In:2010 IEEE International Symposium on Workload Characterization. Atlanta GA USA,2010,978-987.
10. K. Salah, and A. Kahtani. “Improving Snort Performance under Linux. Published in Institution of Engineering and Technology Communications”,2009,3(12):1883- 1895.
11. J. C. Wang, H. P. Cheng, and B. Hua. “Practice of Parallelizing Network Applications on Multi-core Architectures”. In: Proceedings of the 23rd international conference on Supercomputing. New York,2009,204-213.
12. S. X. Zhu, D. Y. Chen, Z. Z. Ji, and G. L. Sun. “Hardware Data Race Detection Algorithm based on Sliding Windows”. Journal on Communications, 2016, 37(9):10-19.

Xian Zhang graduated from the College of Computer Science and Electronic Engineering at Hunan University with a Master’s degree. His research interests include embedded systems, network management, and network measurement.

Dong Yin graduated from Guizhou University with a Master’s degree. His research interests include data processing, auxiliary engineering, network applications, image processing, and information security.

Taiguo Qu graduated from the College of Computer Science and Electronic Engineering at Central South University with a Ph.D. His research interests include machine learning and bioinformatics.

Jia Liu graduated from Hunan Normal University with a Master’s degree. His research interests include wireless virtual networks, network calculus, and SDN.

Yiwen Liu graduated from Xiamen University with a Master’s degree. His research interests include program language analysis, software tests, and digital image processing.