Journal of Computer and Communications, 2015, 3, 1-6
Published Online May 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.35001

Deep Packet Inspection Based on Many-Core Platform

Ya-Ru Zhan[1], Zhao-Shun Wang[2]

[1]Department of Software Engineering, University of Science & Technology Beijing, Beijing, China
[2]Department of Computer Science & Technology, University of Science & Technology Beijing, Beijing, China
Email: zhanyr111@163.com, zhswang@sohu.com

Received February 2015

Abstract

With the development of computer technology, network bandwidth and network traffic continue to increase. Considering the large data flow, it is imperative to perform inspection effectively on network packets. In order to find a solution of deep packet inspection which can appropriate to the current network environment, this paper built a deep packet inspection system based on many-core platform, and in this way, verified the feasibility to implement a deep packet inspection system under many-core platform with both high performance and low consumption. After testing and analysis of the system performance, it has been found that the deep packet inspection based on many-core platform TILE_Gx36 [1] [2] can process network traffic of which the bandwidth reaches up to 4 Gbps. To a certain extent, the performance has improved compared to most deep packet inspection system based on X86 platform at present.

Keywords

Many-Core Platform, Deep Packet Inspection, Application Layer Protocol, TILE_Gx36

1. Introduction

Openness of the Internet leads to diversity of network information, which has provided convenient experience to users, but meanwhile contains a variety of security risks, such as network attacks. Only by having a profound understanding of the Internet traffic, simultaneously performing effective inspection and control on it could make a sound development of the Internet. The traditional packet inspection technology has been confined to data inspection on the network layer and transport layer. It lacks the recognition of the application layer protocol and cannot meet the modern network environment. The emergence of deep packet inspection [3] has played an important role in the development and improvement of the traditional packet inspection technology. But under the huge data flow environment, the deep packet inspection deployment often tends to a large scale, which arouses the urgent need for high performance and low consumption. The many-core processor is the trend of the development of processor technology. Compared to the traditional multi-core processor, it has a higher performance power ratio, making it become the ideal choice. Therefore, the transplantation of open source software—Suricata in many-core platform and expanding the application layer protocol identification on the basic of Open DPI is researched in this paper. Through this way, a deep packet inspection based on many-core platform with high performance and low consumption was built.

How to cite this paper: Zhan, Y.-R. and Wang, Z.-S. (2015) Deep Packet Inspection Based on Many-Core Platform. *Journal of Computer and Communications*, **3**, 1-6. http://dx.doi.org/10.4236/jcc.2015.35001

2. Related Works

At present, most researches on deep packet inspection are based on X86 platform. The X86 platform has more traits in common, which enables its flexibility and strong extendibility. Although the platform can meet the demands to realize the functions of deep packet inspection, but in terms of performance it will be limited by architecture. The X86 platform has more levels of structure, making the optimization become difficult. The performance optimization under X86 platform mainly works through specialized hardware unit to accelerate the specific function modules. But this approach cannot guarantee the power consumption of the whole system. S.D harmapurikar has adopted the bloom filter [4] to realize a deep packet inspection system based on hardware acceleration [5]. It mainly utilized bloom filter and dedicated hardware unit to improve the throughput and detection speed of deep packet inspection. Inspection on HTTP flow under X86 platform indicated that the throughput was merely 1.333 Gbps [6].

Beyond that, there are other two kinds of deep packet inspection technology based on ASIC and NP platform, respectively. ASIC platform uses special chips to perform hardware accelerated processing. It built detection logic into the chips in order to improve the efficiency of inspection, and through this way optimizing the system performance. However, the method resulted in poorer expansion, higher research costs and also a longer development cycle, which makes the ASCI platform only applicable to the place with higher demands on the throughput and delay.

NP platform is specially designed for network equipment. Its architecture and instruction set perform a particular optimization on usual network data, which can achieve data packet filtering and forwarding efficiently. NP platform facilitates the development and application, support the extensible service. Meanwhile, the research cycle is shorter and the cost is lower. But compared to X86 platform, the application development and function expansion of NP platform is limited by its supporting software, making the firewall based on NP got a poorer flexibility. Therefore, NP is beneath ASIC in terms of performance. The difficulty and flexibility of NP development is between ASIC and X86. So far the main provider of NP platform is Intel and Motorola internationally and Lenovo in the domestic.

Both ASCI and NP platform are not open to deep research. It can be seen that the deep packet inspection system based on X86 architecture can be limited by general hardware, while the system based on special hardware filter chip has a poor expansion. This paper will study the performance of the deep packet inspection system on many-core platform TILE_Gx36 under large flow network environment.

3. Overall Structure

This deep packet inspection system based on TILE_Gx36 is primarily composed of three parts: package receiving module, packet processing module and packet forwarding module. The mPIPE unit provided by TILE_Gx36 platform receives message from Ethernet port and sends it to packet processing module, which deals with the data packet. Then packet forwarding module sends the processed packet to host via PCIe. The system in host forwards the data to upper-layer applications. The overall structure of the system is shown in **Figure 1**.

3.1. Packets Receiving and Sending by mPIPE

The mPIPE is a programmable engine of packet processing supplied by TILE_Gx36 platform, with the function of receiving and sending network packets from Ethernet port. It has strong ability of data flow processing, can provide data flow throughput up to 40 Gbps [7]. It has two components: hardware cache manager and package distribution engine. They can perform packets classification, load balancing and buffer management on data packets from Ethernet port in order to achieve high speed of sending and receiving data packets. The system receiving process using mPIPE hardware unit is as follows: mPIPE puts the data packets received from Ethernet port in a receiving package cache, and uses the load balancer to choose a suitable thread for packets to wait for packet processing.

3.2. Packet Processing and Forwarding

3.2.1. Model Select

In the many-core environment, there are two kinds of packet processing model: pipeline model and worker model. Pipeline model refers to that every core is responsible for one stage of the packet processing. That is, after the packet is received, it will be distributed for the second decoding processing on the core and then sent to

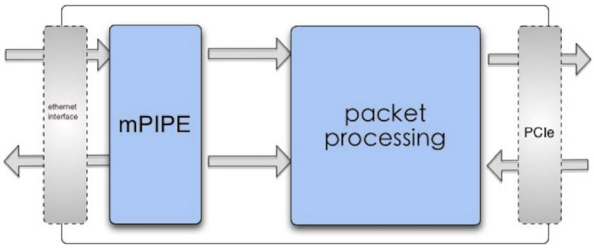

Figure 1. The overall structure of the system.

the next core for IP fragmentation restructure processing in turn. That calls for allocating the same packet for many times and each core has to wait for the former core before its own processing. In this case, the parallelism is poorer and it has a great influence on performance. In addition, the flow reverting will not perform processing until it stores all the packets which are belong to this stream. The pipeline model can't meet the need of flow reverting. The worker model distributes all the packets that belong to the same stream to the same core, which is used to complete all the processing. The core can also conduct flow reverting with a better performance than pipeline model. So, the worker model is adopted to this system.

3.2.2. Core Distribution

This system adopts a multi-threading architecture and besides one Linux system specific core, each core runs a thread. With the principle of meeting the performance and other functions, as many as possible of packet processing threads will be distributed and the default distribution of cores is shown in **Table 1**. The configuration files can also be modified according to different situation to adjust the number of cores.

3.2.3. Packet Processing and Forwarding

Packet processing module, which is based on the software architecture of Suricata, processes the packet received from mPIPE. Then sends it to packet forwarding module through PCIe, and sends the results to the upper layer application after processing. The process is shown in **Figure 2**.

The mPIPE receives packets from the Ethernet port and forwards them to the processing thread of packet processing module. Processing thread decodes second layer protocol of the packet according to the link layer protocols and works out the third and fourth layer information of the packet. In the third-layer decoding process, if the results are IP packets, then judge whether they are IP fragmentation. If so, the slice will be cached until all fragmentations have been received. Then the packets will be processed subsequently. In the case that the collection procedure is not within required time, all fragment nodes of the packet will be aged. Considering the five-element-array information of the third and fourth layer as a key to perform hash lookup on flow table. Then refresh the timestamp of flow table nodes after a successful lookup, at the same time inherit the information like application layer protocol and action preserved in flow table. If the packet is the first package of this stream, the lookup will fail. Then the system build a new flow table and write in timestamp and quintuple information. If the transport layer protocol is TCP, system will operate TCP flow restructuring. When receiving TIN/RST packets or the cache reaches the threshold, the flow restructuring will be stopped and packets will be processed subsequently.

After the accomplishment of the above processing, system will perform DPI and the identify the application layer, then carry out rule matching on packets (System conducts rules filtering based on snort [8]. It supports keywords matching for quintuple and packets within 256 bytes). The current system version can temporarily support actions including blocking, forwarding, logging. It perform actions based on the rules matching. Meanwhile, for further detailed analysis, the results can be sent along with the packets to the upper application via PCIe.

4. Optimization and Test

4.1. Protocol and Expansion of Application Identification

The application layer protocol identification module of Suricata identifies the application layer protocol through feature code and port number. Firstly, the module makes use of multiple pattern matching algorithm and the

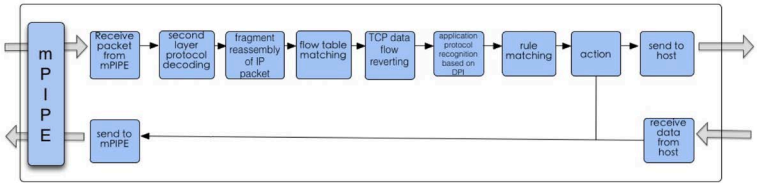

Figure 2. Packet processing.

Table 1. Cores numbers distributed for each function.

Function	Distribution of cores
Linux system	1
packet processing thread: process packet received from mPIPE	19
system manage thread: log sending, admin command processing, heartbeat sending	1
RCVCMD thread: command processing, like issued rules SendCMD thread: send commands to upper software system	2
Forward thread: assemble data from PCIe into packet, then forward them to mPIPE	4
Action thread: send processed packet to upper application	4
MpipeXmit thread: send packet to internet via mPIPE	4
flow manage thread: scan flow table and IP fragment and delete node that out of aging time	1

feature code to identify the application layer protocol. If the application layer protocol cannot be identified, then compare the port information of the packet stream with the application layer protocol detection port registered. After analysis and testing, the application layer protocol identification module of Suricata was found to only supports several mainstream application layer protocols, such as HTTP, DNS, FTP, etc. And the accuracy rate of protocol recognition is very low.

In contrast, the Open DPI System can support 118 kinds of application layer protocol, and the precision of protocol recognition is relatively higher [9]. Open DPI calls detection function of application layer protocol according to the stream information of received message directly to identify the application layer protocol. In order to enhance the ability of application protocol recognition, the application protocol module is added to this system after TCP data flow reverting. The module is realized by dynamic-link library of Open DPI.

Since there are already management operations such as flow counting, packet counting, flow establishing, flow deleting in the system, it is not necessary to use the management structure of Open DPI. Only the protocol identification module is needed. The protocol identification module supplied by Open DPI is an interface, which is ipoque_detection_process_packet() fuction.The function has seven parameters: ipoque_struct, ipq_flow, iph, ipsize, time, src and dst. Among of them, iph, ipsize, time, src, dst has been obtained by previous packet processing, while ipoque_struct, ipq_flow are defined in Open DPI. So, function getparam() is built in system to get the two parameters and pass the parameters to ipoque_detection_process_packet() function through calling DLL.

4.2. Test

4.2.1. Function Test

Every module in the system is tested by corresponding data packet from Ixia packet sending machine. The system can achieve functions of link layer protocol identification like IP fragmentation and reassembly, TCP stream reassembly, keyword matching, rules filter, PPPOE, VLAN, GRE, MPLS, and application protocol recognition such as FTP, HTTP, SSH, TLS. The test shows that the system has achieved the functions of deep packet inspection.

4.2.2. Performance Test

The performance test includes two aspects: power test and throughput test.

Nineteen cores were assigned to package processing in the system. The system can be tested by using different numbers of cores to conduct statistical analysis on the corresponding data.

In power test, using a wattmeter to monitor power consumption of different numbers of cores being used, as shown in **Figure 3**.

As can be seen in the figure, the power consumption has only a tiny increase with the cores numbers growing. Although the nineteen threads are all started, the consumption is less than 70w.

There are two cases in throughput test: not issued any rules and issued two thousands rules. Ixia packet sending machine is adopted to conduct the performance test on system respectively by each constructing packet of 200 bytes. And the flow handled on the starting of system dropout is considered as the limit state of the system data processing.

Results of the throughput statistics is shown in **Figure 4**. As can be seen from the figure, whether issued rules or not, to the first eight cores, the system flow processing grows approximately linearly with the numbers of cores. While along with the numbers continue to increase, the curve is no longer linearly but declined. That is because thread locks have a great impact on performance in multithreaded. In circumstance when the system restricts two thousands rules, the data flow handled by system reaches approximately 4 Gbps.

5. Conclusion

This paper discussed Deep packet inspection technology based on many-core platform TILE_Gx36. It transplanted Suricata software framework into the many-core platform, and integrated Open DPI to extend the module of application layer protocol identification. The built system can support the recognition of a hundred kinds of application layer protocol, and process network traffic of which the bandwidth reaches up to 4 Gbps. However, the distribution of the numbers of cores in this system is limited to theoretical analysis. It calls for a more reasonable scheme by experimentation. The multi-pattern matching algorithm was applied in both keyword matching module and application protocol recognition module. The next research direction can be the algorithm

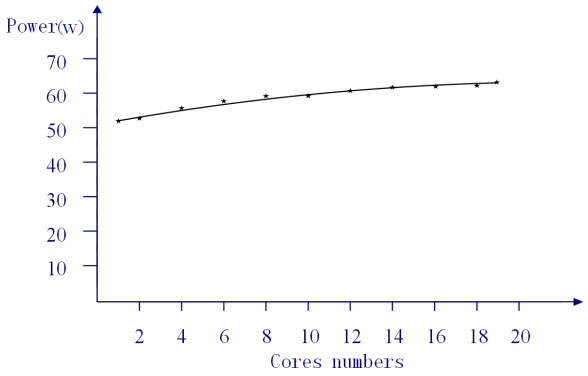

Figure 3. Power of different cores numbers.

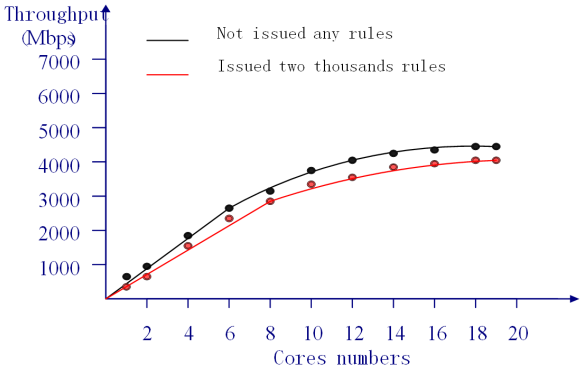

Figure 4. Throughput of different cores numbers.

optimizing to improve the performance.

References

[1] Tilera Inc. (2013) ArchOverview-TILE-Gx.

[2] Tilera Inc. (2013) TIL Encore-Gx-Card-UG.

[3] Lin, Q. (2013) The Design and Implementation of P2P Flow Based on DPI. Zhejiang University of Technology.

[4] Huang, K. and Zhang, D.F. (2011) An Index-Split Bloom Filter for Deep Packet Inspection. *Science China* (*Information Sciences*), **1**, 23-37. http://dx.doi.org/10.1007/s11432-010-4132-4

[5] Dharmapurikar, S., Krishnamurthy, P., Sproull, T. and Lockwood, J. (2004) Deep Packet Inspection Using Parallel Bloom Filters. *IEEE Micro Magazine*.

[6] Ai, X. (2013) The Design and Optimization of DPI System on Many-Core Environment. Harbin Institute of Technology.

[7] Tilera Inc. (2013) mPIPE-Guide.

[8] Snort: An Open Source Network Intrusion Prevention and Detection System (2010).

[9] Wei, Y., Zhou, Y.F. and Guo, L.C. (2011) The Recognition and Analysis of Packet with Open DPI. *Computer Engineering*, **S1**, 98-100.