

Multicore Packet Distribution method using Multicore Network Interface Card based on Tile-gx72 Network Processor

Won Seok Choi*, Sang Ju Lee**, Jong Oh Kim**, Seong Gon Choi*

* Information & Communication Engineering, Chungbuk University, Cheongju-si, Chungcheongbuk-do, Korea

**Fisys Inc., 168, Gajeong-ro, Yuseong-gu, Daejeon, Korea

wschoi@chungbuk.ac.kr, angelet86@fisys.co.kr, jokim@fisys.co.kr, choisg@chungbuk.ac.kr

Abstract— We propose a data plane acceleration technology to deliver data from the network to the host system in a high-performance computing environment. In the fourth industrial revolution, server systems are developing into high-performance computing systems through convergence with major technologies such as IoT, cloud, AI, and self-driving cars. The 4th industrial revolution is the convergence of various technologies and IT, requiring various flows and large amounts of data to be processed on servers. When transferring packets from the network interface card to the host server, packet processing in kernel space has a large overhead. Additionally, for fast packet processing by the host server, packets must be processed according to core affinity. Therefore, we propose a load balancing data transmission method to 48 cores based on Tile-Gx72 network processor to transfer data from the network interface card to the host CPU by kernel bypass in a multi-core-based high-performance server system. In addition, the performance of the 48 cores-based load balancing data transmission system based on the Tile-Gx72 network processor is confirmed through implementation.

Keywords— Multicore, Network Interface Card, Network Processor, Packet distribution, Tile-gx72

I. INTRODUCTION

The 4th Industrial Revolution is developing through the convergence of various technologies such as IoT, cloud, AI, and self-driving vehicle, and the use and amount of data required to provide services using technology is increasing. [1].

Internet Data Center (IDC) predicts that the amount of data generated will reach 73 trillion bytes by 2025 due to the development of IoT technology, and the generated data is expected to be transmitted to servers through the network

IDC predicts that the amount of data generated by IoT technology will reach 73 trillion gigabytes in 2025 [2]. This data will be transmitted to servers or devices through the network. This means that the amount of data transmitted through the network and the amount of data processed by the server CPU increases. However, the gap between network line

speeds and the speed at which CPUs can produce and consume data is rapidly widening [3].

In particular, when transmitting data between a Network Interface Card (NIC) and a CPU, a kernel space is a bottleneck, and overhead due to packet processing causes performance degradation. In this case, the problem is usually solved by using the kernel bypass to directly deliver packets to the user space [4].

Solutions that use the kernel bypass include Data Plane Development Kit (DPDK), Single Root I/O Virtualization (SR-IOV), and Jae Woo Ahn et al.'s load distribution method using multicore based network interface card for high-performance computing System.

DPDK is an intel architecture-based packet processing optimization system that uses environment abstraction layer and poll mode driver as software to the bypass kernel space and transmit packets to user space [5]. However, in order to use the DPDK, a NIC driver that supports the DPDK is required, which limits NICs that can be used.

SR-IOV is a technology defined by PCI-SIG that shares I/O devices in a virtualization environment. In a virtualization server using a virtual machine, the SR-IOV maps the virtual NIC of the virtual machine to a Virtual Function (VF) of the physical NIC of the host server, bypassing the host server's kernel space and enabling fast packet transmission and reception [6]. As a result, SR-IOV reduces the overhead of kernel space on the host server and provides the virtual NIC functionality, but cannot reduce the overhead of kernel space on the virtual Machine.

Jae Woo Ahn et al. propose a load distribution method using a multicore based NIC for high-performance computing system that distributes packets of up to 20 Gbps to 28 cores of servers using a NIC based on Tile-gx36 network processor [4]. For fast packet processing in the host server CPU, packets are distributed according to core affinity according to flow. However, as the computing power and number of cores of servers are increasing, there are limitations in transmitting packets over 20 Gbps and distributing packets over 28 cores.

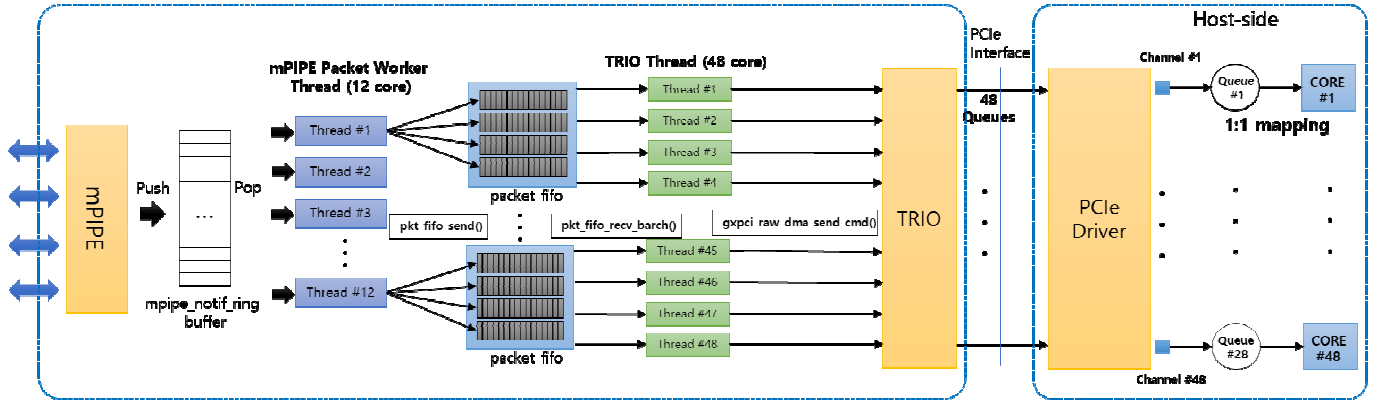


Figure 1. Architecture of the proposed system

Therefore, in order to receive 40 Gbps packets from a Multicore-based high-performance server system, we use Tile-gx72 that is a multicore-based network processor to accept 40 Gbps packet reception. Furthermore, the proposed system distributes packets of 40 Gbps to 48 host server cores by bypassing the server's kernel space. Additionally, performance tests were performed using laboratory-level instruments.

The remainder of this paper is organized as follows: we first introduce data plane acceleration method such as DPDK, SR-IOV in Section 2. In Section 3, we introduce the proposed method in detail. Section 4 reports the implementation and test results. Finally, conclusions and future studies are described in Section 5.

II. RELATED WORK

In this session, we introduce DPDK and SR-IOV as data plane acceleration methods.

A. DPDK

DPDK is a set of data plane development tools provided by Intel that provides library functions and driver support for efficient packet processing in user space on Intel processor architectures. This is different from the purpose of the universal design of Linux system. The DPDK focuses on high-performance processing of packets in network applications. The best feature is that DPDK applications run on user space and use their own data plane libraries to send and receive data packets, bypassing the Linux kernel protocol stack for packet processing [5].

However, in order to use the DPDK, a NIC driver that supports the DPDK is required, which limits NICs that can be used.

B. SR-IOV

SR-IOV proposes a set of hardware improvements to PCIe devices that aim to eliminate key Virtual Machine Monitor (VMM) interventions in performance data movement, such as packet classification and address translation. The SR-IOV inherits Direct I/O technology, which uses I/O Memory Management Unit (IOMMU) to offload memory protection and address translation. The SR-IOV capable devices can create multiple "lightweight" instances of PCI function entities,

called VFs. Each VF can be assigned to a guest for direct access [6].

As a result, the SR-IOV reduces the overhead of kernel space on the host server and provides the virtual NIC functionality, but cannot reduce the overhead of kernel space on the virtual machine because it still performs packet processing on kernel space of the virtual machine.

Therefore, we propose the load balancing data transmission method to 48 cores based on the Tile-Gx72 network processor to transfer data from the network interface card to the host CPU by kernel bypass in a multi-core-based high-performance server system.

III. PROPOSED METHOD

We propose a system that delivers packets to 48 cores of the host based on Tile-gx 72 network processor in order to distribute the data collected from the network interface card according to the flow with load balancing.

Figure 1 shows the structure of the proposed 48 cores-based data transmission system with load balancing. The proposed structure consists of tile-side, which consists of a network interface card based on the Tile-Gx72 process, and host-side, where the network interface card is installed. Tile-side and Host-side are connected through the PCIe interface.

Tile-side consists of multicore Programmable Intelligent Packet Engine (mPIPE), mPIPE packet worker thread, TRIO Thread, and Transactional I/O (TRIO).

The mPIPE is a hardware packet processing engine that performs packet collection, packet transmission, packet header parsing, packet distribution, packet buffer management, load balancing, and L4 checksum. It performs the same role as a network interface card in a typical PC or server [3]. Additionally, mPIPE hashes packets transmitted to the host and stores them in the mpipe_notif_ring buffer.

The TRIO handles Direct Memory Access (DMA) data movement and buffer management for data transfer through PCIe between the Tile memory system and the Host memory system [3].

The mPIPE packet worker thread is a thread that processes data collected from the mPIPE, and the transactional I/O (TRIO) thread is a thread that controls the TRIO and delivers packets to the host [3].

The mPIPE packet worker thread is responsible for processing packets collected from mPIPE. The mPIPE packet worker thread uses the packet information stored in the mpipe_notif_ring buffer to match the hashing-based flow affinity and transmits it to the TRIO thread as a packet fifo. The mPIPE packet worker thread uses a total of 12 tile cores, each transmitting with 4 TRIO threads [3].

The TRIO thread is responsible for transmitting data collected from the mPIPE packet worker thread to the host using direct memory access technology. There are 48 TRIO threads, each mapped 1:1 to transmit packets to 48 cores of host.

Host-side consists of a PCIe driver and host core to support the 48-core-based load balancing data transmission software of the network interface card.

Packets with flow affinity from mPIPE packet worker threads are delivered to 48 cores of host by matching core affinity.

IV. IMPLEMENTATION RESULTS

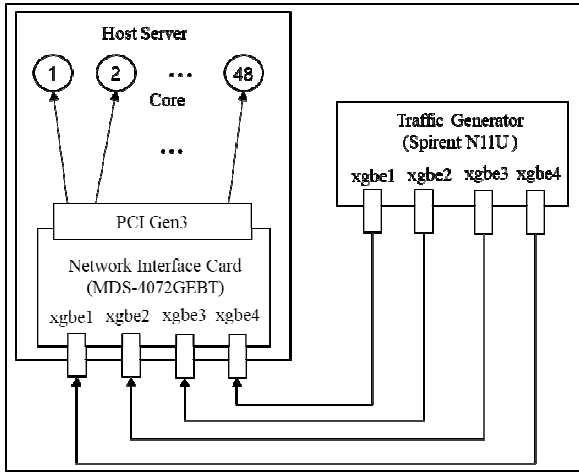


Figure 2. Ideal test environment

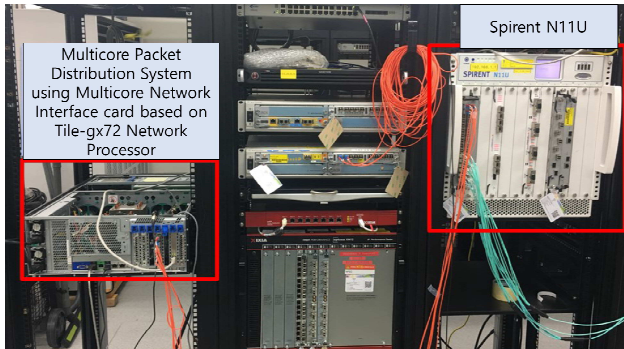


Figure 3. Real test environment

We analyze the performance of the proposed multicore packet distribution method using MDS-4072GEBT and Spirent N11U.

Figure 2 shows ideal test environment and figure 3 shows real test environment to confirm performance of the proposed

method. We performed performance tests using the MDS-4072GEBT that is a Tile-gx72 network processor-based network interface card manufactured by Fisys Inc. The MDS-4072GEBT network interface card was installed on the host server through the PCI Gen3 interface and the proposed 48-core data transmission system was operated. Traffic generation was performed using the Spirent N11U.

TABLE 1. MDS-4072GEBT NETWORK INTERFACE CARD

List	Specification
Network Processor	- Tile-Gx72 - Core: 72
Interface	- 10Gbps SFP+ X 4 Port
PCIe	- PCI Gen3
OS	- Tile OS

TABLE 2. HOST SYSTEM

List	Specification
CPU	- Intel(R) Xeon(R) CPU - E5-2695 v3 @ 2.30GHz - Core: 14 x 2 - Thread: 28 x 2
PCIe	- PCI Gen3
OS	- CentOS 6

The specifications of MDS-4072GEBT are shown in table 1. The network processor is equipped with Tile-gx72 and can support 10 Gbps SFP+ 4 ports. Furthermore, it supports PCI Gen3 and PCIe Gen2.

Table 2 shows the system information of the host system. The host system uses Intel(R) Xeon(R) CPU E5-2695 v3 CPU, and a total of 56 logical cores can be used.

In this performance test, traffic was generated using 4 ports of the Spirent N11U, and the results of data transmission from the MDE-4072GEBT to 48 host cores are shown in figure 4 and figure 5. Traffic settings were set to 400 flows of 512 bytes and different IPs, and traffic was transmitted at 40Gbps.

Figure 4 shows utilization of host cores using top program that is provides a dynamic real-time view of a running system in Linux system. It was confirmed that packets were distributed to 48 host cores, resulting in a load of less than 5 percent.

Figure 5 shows the results of packet transmission from the Spirent N11U to the MDS-4072GEBT. The left side of figure 5 shows the amount of packets transmitted from four SFP+ ports on the Spirent N11U. The right side of figure 5 shows the amount of packets received by 48 host cores. The Spirent N11U transmitted 739,563,206 packets to the host server at 40 Gbps. As a result, it was confirmed that the MDS-4072GEBT installed on the host server received packets and transmitted 739,563,206 packets to 48 host cores. We used a self-developed host application to check received packets on host cores.

Cpu0 : 1.6%us,	Cpu12 : 1.3%us,	Cpu24 : 4.3%us,	Cpu36 : 1.7%us,
Cpu1 : 2.7%us,	Cpu13 : 3.0%us,	Cpu25 : 5.0%us,	Cpu37 : 2.7%us,
Cpu2 : 1.3%us,	Cpu14 : 3.0%us,	Cpu26 : 2.0%us,	Cpu38 : 1.0%us,
Cpu3 : 1.3%us,	Cpu15 : 4.3%us,	Cpu27 : 2.7%us,	Cpu39 : 1.0%us,
Cpu4 : 2.3%us,	Cpu16 : 5.0%us,	Cpu28 : 0.3%us,	Cpu40 : 2.3%us,
Cpu5 : 2.3%us,	Cpu17 : 5.0%us,	Cpu29 : 2.7%us,	Cpu41 : 2.7%us,
Cpu6 : 2.0%us,	Cpu18 : 3.3%us,	Cpu30 : 2.0%us,	Cpu42 : 3.7%us,
Cpu7 : 1.3%us,	Cpu19 : 2.0%us,	Cpu31 : 1.0%us,	Cpu43 : 5.0%us,
Cpu8 : 1.7%us,	Cpu20 : 5.0%us,	Cpu32 : 1.0%us,	Cpu44 : 4.7%us,
Cpu9 : 1.3%us,	Cpu21 : 5.0%us,	Cpu33 : 1.3%us,	Cpu45 : 3.7%us,
Cpu10 : 1.3%us,	Cpu22 : 2.7%us,	Cpu34 : 0.7%us,	Cpu46 : 3.7%us,
Cpu11 : 1.7%us,	Cpu23 : 3.7%us,	Cpu35 : 1.3%us,	Cpu47 : 1.7%us,

a. core 0~11

b. core 12~23

c. core 24~35

d. core 36~47

Figure 4. Result of top command

Port Traffic and Counters > Basic		
Basic Counters	Errors	Trigg
Port Name	Total Tx Count (Frames)	
Port //1/1	184,869,159	
Port //1/2	184,888,733	
Port //1/3	184,906,602	
Port //1/4	184,898,712	
Σ	739,563,206	

a. Spirent N11U

46	7394961
47	15789311
Total	739563206

b. received packets of host side

Figure 5. Send and received packets

V. CONCLUSIONS

We proposed a 48-core data transmission system based on Tile-gx72 network processor to distribute 48 cores of host by distributing the load with flow affinity. Furthermore, the proposed system was tested through empirical testing. As a result of the test, it was confirmed that the load was distributed across 48 host cores and packets were transmitted with keeping under 5% core load. The proposed system is a system that distributes packet load in a high-performance server system and transmits packets from the network interface to the host. It can be used in high-performance packet processing servers such as self-driving cars and internet of things that have various flow and high-capacity traffic characteristics.

In further study, we plan to conduct performance experiments and research by linking the load-balancing data transmission system with the host's service application.

ACKNOWLEDGMENT

This work was partly supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2024-2020-0-01462, 33%) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). This work was partly supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2020R1A6A1A12047945, 34%). This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1006167, 33%).

*corresponding author is S.G. Choi(choisg@cbnu.ac.kr).

REFERENCES

- [1] Hansen, E. B., and Bøgh, S., "Artificial intelligence and internet of things in small and medium-sized enterprises: A survey," *Journal of Manufacturing Systems*, 58, pp. 362-372, 2021.
- [2] Tomer, V., and Sharma, S., "Detecting IoT Attacks Using an Ensemble Machine Learning Model," *Future Internet*, 14(4), 2022.
- [3] Lin, Jiaxin, et al. "{PANIC}: A {High-Performance} Programmable {NIC} for Multi-tenant Networks," *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020.
- [4] Ahn, J. W., Kim, J. B., Choi, W. S., Kim, J. O., and Choi, S. G., "Load distribution method using multicore based NIC for high-performance computing system," *In 2018 20th International Conference on Advanced Communication Technology (ICACT)*, pp. 90-93, Feb. 2018.
- [5] Zhu, W., Li, P., Luo, B., Xu, H., and Zhang, Y. "Research and implementation of high performance traffic processing based on intel DPDK," *In 2018 9th international symposium on parallel architectures, algorithms and programming (PAAP)*, pp. 62-68, 2018.
- [6] Dong, Y., Yang, X., Li, J., Liao, G., Tian, K., and Guan, H. "High performance network virtualization with SR-IOV," *Journal of Parallel and Distributed Computing*, 72(11), pp. 1471-1480, 2012.



Won Seok Choi received B.S. and Ph.D. degree in the College of Electrical and Computer Engineering, Chungbuk National University, Korea in 2008 and 2014 respectively. He is currently researcher in Research institute of Computer and Information Communication, Chungbuk National University. His research interests include vehicle network, energy saving network, SDN, NFV and NGN.



Sang Ju Lee received B.S and M.S. degree in the College of Electrical and Computer Engineering, Chungbuk National University, Korea in 2014 and 2016 respectively. He is Senior Researcher in Fisys Inc. His research interests include SDN, NFV and CCIX



Jong Oh Kim received B.S. and M.S. degree in Electronics Engineering from Kyeongbuk National University in 1990 and 1992 respectively. He is currently CEO in Fisys Inc. His research interests is SDN, NFV



Seong Gon Choi received B.S. degree in Electronics Engineering from Kyungpook National University in 1990, and M.S. and Ph.D. degree from KAIST in Korea in 1999 and 2004, respectively. He is currently a professor in College of Electrical & Computer Engineering, Chungbuk National University. His research interests include V2X, AI, smart grid, IoT, mobile communication, high-speed network architecture and protocol.