# DPDK-based Improvement of Packet Forwarding

Hao BI[1,a], Zhao-Hun WANG[2]
[1]*Postgraduate ,University of Science and Technology Beijing, Beijing P.R. ,China*
[2]*Professor,University of Science and Technology Beijing, Beijing P.R. ,China*
[a]Corresponding author: bihao@baidu.com

**Abstract.**Reel-time processing of packets occupies a significant position in the field of computer network security. With theexplosive growth of the backbone link rate,which is consistent with Gilder's law, many bottlenecks of server performance leave the real-time data stream unprocessed.Thus, we proposedto take use of DPDK(Data Plan Development Kit) framework to achieve an intelligent NIC packet forwarding system. During this research, we deeply analysis the forwarding process of packet in DPDK and improve its DMA mode.According to the results of experiment, the system greatly enhanced the performance of packet forwarding,and the throughput of forwarding 64-byet or random-length packets by 20Gbit NIC reaches13.3Gbps and 18.7Gbps(dual ports forwarding).

## 1 Introduction

High-speed network packet forwarding system plays a vital role in many fields, such as packet analysis, virus detection and traffic management. Universal NIC runs into a big bottleneck in many aspects, such as data buffering, packet copy and frequent interruptions which causes huge overhead. For the host, it causes excessive consumption of the computing, storage, I/O overhead which is too large to meet the requirements of line-speed data capture and handling on the Gigabit link.

In order to reduce the cost of data copies and complexity of kernel protocol stack process,Intel launched a data plane development kit (DPDK) of which the purpose is to provide programmers a simplee and complete fast packet processing framework[1].Useers can create prototypes or add their own protocol stack according to their requirements.DPDK achieves a zero-copy, large pages, lock-free buffer queue, DMA access mode, user-space drivers and other key technologies to enhance the packet processing capability. In the field of high-speed traffic processing, DPDK has a huge advantage: 1.High performance, the data provided by Intel shows that packet with random length forwarding can reach 10Gbit line-speed; 2.It supports almost all common Intel NIC; 3.For specific NIC, DPDK provides full support, users can take advantage of user-space drivers to build their own packet forwarding system, which greatly simplifies the development process [2].

In this paper, we analyze the key technologies of DPDK, and take advantage of DPDK framework to build a packet forwarding system. In the process of DMA transfer,DPDK needs to send descriptor separately, which wasted PCIe bandwidth, resulting in a performance bottleneck, therefore we propose a new method named Minimum Overhead DMA to solve this problem.

## 2 Key Technologies in DPDK

DPDK (Data Plane Development Kit) is an optimized data plane software solution developed by Intel for its multi-core processors. It includes high performance packet processing software formed by application processing, control processing, data plane processing, and signal processing tasks onto a single platform. It focuses on improving performance of packet processing.

### 2.1 Efficient Mmemory Mmanagement

In the DPDK initialization phase, it uses a large space to form a memory pool, which is an object allocator with fix size (storing packets entities require memory alignment to ensure that the object in memory allocated in different memory channels). In DPDK, the memory pool is identified by name, and uses a ring to manage free objects. In case of multiple consumers, you can use pipe-line mode to access the ring. To avoid competition, DPDK uses CAS (compare and swap) between multiple threads so that they can access the ring (lock-free queue) without using lock, which will greatly reducee the huge overhead of lock competition.

When large memory-intensive applications run on Linux platform of operating system, the usage of the default 4kb-size page will cause a lot of page faults and TLB misses, which greatly affect the performance of the application.DPDK uses huge page technology to solve this problem. The use of large pages can significantly reduce the frequency of TLB miss and page fault, reducing unnecessary consumption of memory access.

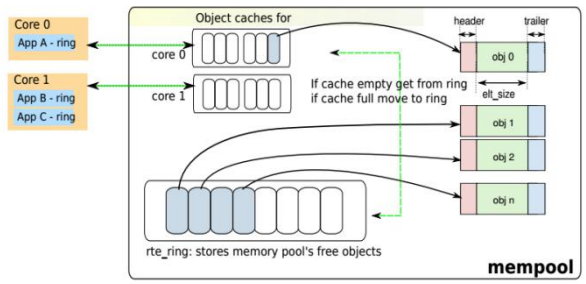


**Figure 1.** Efficient Memory Management

### 2.2 Implementation of Zero-copy

Through aanalysis of traditional packet capture mechanism, we can see how the data stream is flowing. When packets arrive, it will trigger an interrupt; packet is copied from the buffer in NIC to the kernel buffer queue, passes through the kernel then the packet copis from the kernel buffer to the user space, and finally processed the packet in user space. Date flow contains multiple copies of data, resulting in a significant reduction in performance. Since the two newly written data copy operations are inconsistent with localized features, it basically does not have cache hit; therefore the memory access efficiency is very low. Coupled with a large amount of data throughput, the overhead of copy is considerable. Depending on the packet length, a copy of one packet needs to consume about 259-8850 cpu cycles [2].

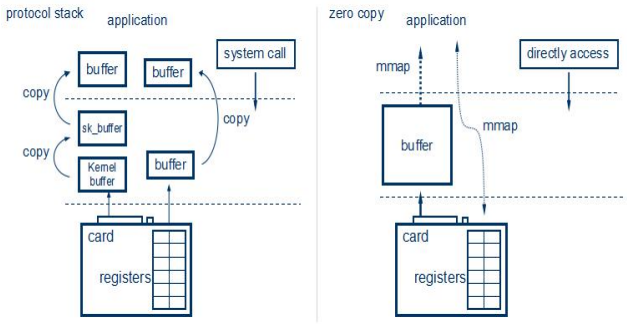


**Figure 2.**Comparison of Traditional Protocol Stack And Zero-Copy

Zero-copy maps kernel space to user space, the underlying driver transfer data directly to the user space through DMA, eliminating the copy from the kernel space to user space. Compared to the protocol stack process, zero-copy is an efficient and special communication technology; it avoids unnecessary data copying operation, and effectively improves the performance of packet processing.DPDK directly mapped buffer of packets in kernel space to user space, avoiding multiple copies of the data by using map.

**Table1.**CPU Writes Descriptor for Received Packet

| | 63 1 | 0 |
|---|---|---|
| 0 | Packet Buffer Address[63:1] | AO |
| 8 | Packet Buffer Address[63:1] | DD |

### 2.3 Pool-Mode driver in user space

Linux provides a kernel module named UIO, the user-space driver can be encoded using this module.DPDK uses module igb_uio to generate a file named uioX in the /dev directory, then the file is mapped to the user space by using map function. It will map the NIC register to user space,therefor users can directly access the NIC registers and that allows user-space driver .

Traditional packet capture requests constant switch between interrupt context and process contexts, In case of the arrival of high-speed packets, interrupt overhead will be enormous.DPDK disables the interrupt, using user-space polling mode driver (PMD), which allows the application access to NIC device without through Linux kernel, and it avoids abnormality of driver causing a risk of kernel crash, while greatly reduces the overhead of interruption [3, 4].

PMD is a direct interface upon DPDK specific NIC,different NICs achieve their own PMD which provides a unified interface to an upper layer application. Making specific card support DPDK, the main job is to write a corresponding PMD.

## 3 Aanalysis DMA Mode in DPDK

From the above description, DPDK manages the packet through by using rings. Each object in the ring is a packet descriptor which is associated with the packet buffer ,by reading the descriptor NIC can obtain the actual location of the packet in memory.PMD completes the interaction between the NIC and CPU by reading and writing the descriptor.

Take the Ixgbe 82599 as an example, we can analysis the process of receiving a packet in DPDK. It uses the ring buffer to manage the queue of packets received, and the ring buffer stores fixed length descriptors of incoming packet, rather than storing the actual packet. Descriptors of received packet are written by cpu into the circular queue.[5]When NIC receives the packets it will trigger a hardware interrupt, and make the CPU give up the control of the bus. Ixgbe NIC will receive descriptor of packet and place them into the FIFO buffer through the DMA read-engine. Then the NIC will sent the packet to the memory area where is specified by the descriptor through DMA write-engine, while updating content of the descriptor .Descriptor length is 16 bytes, using bit multiplexing. It has different meanings when the NIC read or write,it is used to indicate address of the packet or the value of the packet parsed via the NIC(Including identifier for the length of packet, which is used to determine whether release of the packet or not)[6].Take the descriptor for received packet as an example:

**Table 2.**CPU Read Descriptor For Received Packet

| | 63<br>48 | 47<br>32 | 31 | 30 21 | 20<br>17 | 16<br>4 | 3<br>0 |
|---|---|---|---|---|---|---|---|
| 0 | RSS Hash/Fragment Checksum | | SPH | HDR_LEN | RSCCNT | Packet Type | RSS TYPE |
| 8 | VLAN Tag | PEK_LEN | Extended Error | | Extended status | | |

DPDK uses descriptor controlling NIC to send and receive packets, a comparison made with using the entire buffer as a circular queue, which has both advantage and disadvantage [7].

Advantages: 1.Applications could maintain received packets until explicitly been freed. If the entire buffer is used as a circular queue, after reading the packet, the read pointer slides, the pointer to packet which application has stored before may have failed; 2.Buffer is shared by sending and receiving packet, therefore received packets can be sentt directly without copied.3.When CPU receives a packet it just needs to check descriptors instead of reading the NIC registers.

Disadvantage: While the NIC sendd or receive packets the descriptor must be written back to the ring, this increases DMA operation and causes excessive DMA overhead as well as a waste of PCIe bandwidth [8].

## 4 Implementation of Mminimum Over head DMA

Based on the above analysis, in the process of sending and receiving messages we need to write back to the descriptor, which resulted in excess DMA operation and caused unnecessary waste of bandwidth, so we proposed an alternative DMA approach, namely Minimum Overhead DMA [8, 9].

To reduce overhead, after receiving the or sending messages, we do not write back to the descriptor information.When NIC receives a packet, we put the additional information in front of the data packet and do not write the additional information back to the descriptor.When NIC send a packet, we do not write back any information to the host, and judge whether the corresponding data has been sent by the position of the pointer to descriptor.

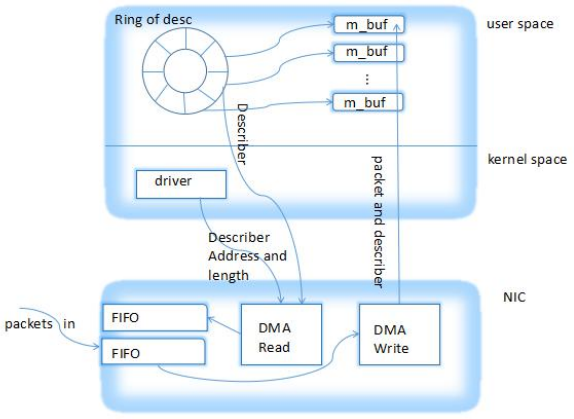


**Figure 3**. Process of Packets Uploading In Dpdk

Based on the analysis of PCIe bus protocol, we can get the PCIe bandwidth utilization before and after

Improvements. According to PCIe 2.0 specification, we can calculate two kinds of DMA modes PCIe theoretical bandwidth overhead. It need to be considered in the calculation include MAX_PAY_LOAD (single transfer maximum payload length, determined by the network card in consultation with the North-Bridge), as well as memory alignment strategy (transmission length is a multiple of cache line size), as well as TLP (transaction layer protocol of PCIe protocol) header overhead.

We assume that the MAX_PAY_LOAD is 256 bytes [10], using 64 byte alignment for transmission. In the process of a single descriptor DMA, descriptor 16 bytes plus the TLP header is less than 64, which needs to be aligned, thereforee the transmission length is 64 bytes in size. During receiving of packets before and after the improvements, it must firstly obtain descriptor, because this part uses the same mechanism so we do not contrast.We aanalyze the process of packet uploading, receiving 64 byte packets for example, each packet has to be encapsulated in the TLP, TLP header plus the 64-byte packet is less than 128 bytes but it needs to be filled to 128 bytes because of the alignment strategy, here for it costs 192 bytes to transmit one packet before improvement. After improvement, It costs 128 bytes to transmit one packet.Accordingly the bandwidth utilization improved by 33.3 percent. The following figure shows that along with the packet length increasing the bandwidth utilization rate after the improvement

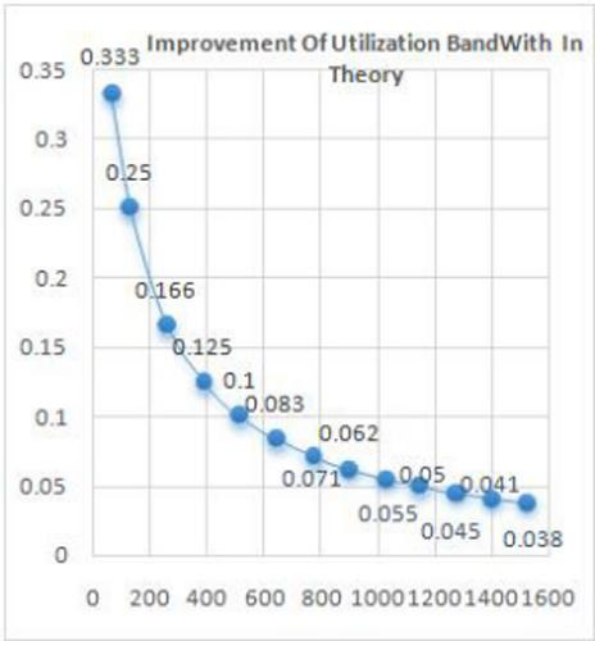


**Figure 4.**Improvement of The Bandwidth Utilization In Theory

### 4.1 Implementation of Packet Uuploading

1. When NIC receives data, it asks the host for descriptor;
2. Host transmit descriptors to NIC descriptor queue using DMA (bulk);
3. Packet and descriptor assembly into the corresponding format;
4. NIC transmits the data using DMA to the memory address (m_buf) where descriptor points to;
5. The CPU reads descriptor (head of the m_buf), and parse the head content;
6. According to the packet length which is parsed before, length> 0 means packet has transferred to memory, release corresponding descriptor;

### 4.2 Implementation of Packet Forwarding

1. When the host has data to sent, it puts the data in the corresponding position and puts descriptor in the ring buffer;
2. When PCIe module finds that a descriptor is valid, it asks the host for descriptor (bulk);
3. Host transmits descriptors to NIC descriptor queue using DMA;
4. The PCIe module requests data according to the information such as the address and size of memory described by the descriptor.
5. Host transmits the data to NIC,PCIe module cache the data to Fifo
6. NIC forwards packets through the port.

## 5. Evaluation

We tested and compared the packet forwarding performance of DPDK original DMA mode and Minimum Overhead DMA mode. It is based on a single Intel Xeon E5-2609 processor (8 logical cores) at 2.4 GHz, 64 GB of RAM. Operating system is Red Hat Enterprise 6.3, and kernel version is 2.6.32.we use a dedicated network tester (SPIRENT SPT-N11U) to transmit and receive packets. Network interface card is Intel 82599 (single-mode optical fibber; dual port Gigabit NIC).Each Ethernet port respectively accept 10Gbps, and forwards back to the network tester via the other port. The test results of the packet forwarding performance of before and after the improvement are shown in figure 5 and figure 6.

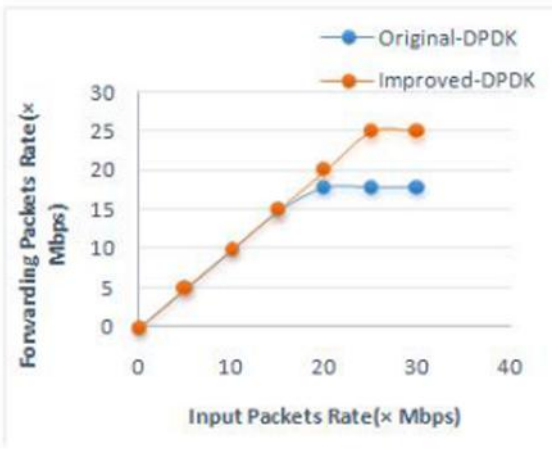


**Figure 5.** 64-Bytes Packets Forwarding

From figure 5we can see, the transfer system has a significant improvements. Especially in case of small packets of 64 bytes, the packet rate peaks at 13.3Gbps (25.3Mpps) after improvement, comparing with original DMA way it has been significantly improved.

We can learn that it also improves the performance of packet forwarding in the case of forwarding random length form figure6.The throughput increases with the length of packet, and it eventually stabledat 18.7Gbps.Conclusion confirms our previous analysis, packets of small bytes wasted a great deal of PCIe bandwidth before improvement, resulting in a performance bottleneck.

Testing shows that the improved DMA mode can effectively solve the waste of bandwidth because of packets transmitting separately with descriptors, and it enhanced performance of packet forwarding.

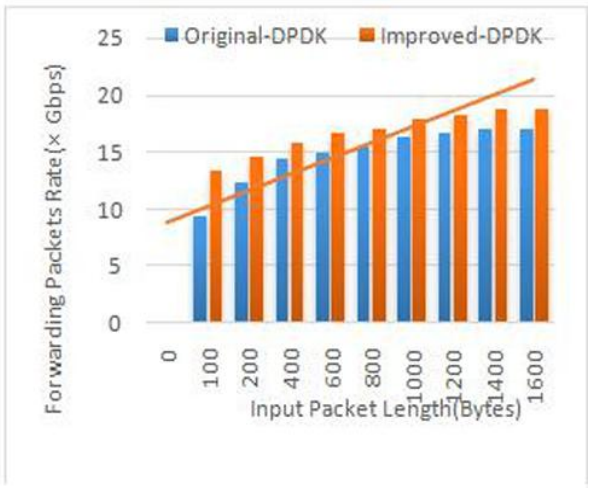


**Figure 6.** Random-Length Packets Forwarding

## 6. Conclusion

In the packet capture system, DPDK is being increasingly used. In this paper, we analyzed the DPDK bottlenecks in DMA mode, and proposed the Minimum Overhead DMA mode using descriptors and packets sending together to

improve the utilization of PCIe bandwidth. Experimental results show that MODMA can effectively solve the performance bottleneck of DPDK packet forwarding, make the system performance, especially in condition of the 64 byte packets improve significantly.

## References

1. DPDKprogramer'sgiude .[Online]http://dpdk.org/doc/intel/dpdk-prog-giude-1.7.0.pdf
2. sample_app_ug-2.1.pdf.[Online].http://www.dpdk.org/doc/pdf-guides/sample_app_ug-16.04.pdf(2014)
3. J.C Mogul and K.K.Ramakrishnan,Eliminating receive lovelock in an interrupt-driven kernel,USENIX Annual Technical Conference,99-112(1996).
4. .E.WANG,Q.ZHANG,B.SHEN,G.ZHANG,X.LU,Q. WU,and Y.Wang,Intel Math Kernel Library, High-Performance Computing on the Intel@Xeon Phi,pp.167-188(2014).
5. K. C. Claffy and T. Monk, What's next for Internet data analysis? Status and challenges facing the community, Proceedings of the IEEE,85, 10, (1997).
6. S.Yamagiwa,K.Aoki and K.Wada,Active Zero-copy: A performance study of non-deterministic messaging, Proceedings of the 4th International Symposium on Parallel and Distributed Computing ,325-332(2005).
7. TeleGeography Research,Overviews:Global bandwidth research products,TeleGeography Company Website(2012)
8. M. Necker, D. Contis, and D. Schimmel, TCP-stream reassembly and state tracking in hardware, IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA(2002).
9. K. C. Claffy and T. Monk, What's next for Internet data analysis? Status and challenges facing the community, Proceedings of the IEEE, 85, 10, (1997).
10. C Krueger, F Valeur, G Vigna, and R A Kemmerer, Stateful Intrusion Detection for High-Speed Networks, In Proceedings of the IEEE Symposium on Security and Privacy, 285-293(2002).