Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

Survey on Software Solution for High Performance Packet Processing

Nanda Kishore
Computer Science and Engineering
Ramaiah Institute of Technology
Bangalore-560054

Dr. S. Rajarajeswari
Assitant Professor
Computer Science and Engineering
Ramaiah Institute of Technology
Bangalore-560054

Dr. Anita Kanavalli
Professor
Computer Science and Engineering
Ramaiah Institute of Technology
Bangalore-560054

***Abstract*— With the evolution of internet, we see huge spike in number of network users. The range of generated internet traffic in most of the various locations is increasing rapidly. Latest networking environment should deliver high throughput, high speed, high bandwidth and less delay properties to cope up with the new era of internet. The traditional data packet acquisition and distribution operations and packet processing capabilities may fail to agree with the new standards of faster data packet processing, in turn leads to data loss, which is costly in packet processing. This survey contributes on finding a faster packet processing framework to overcome the challenges faced by traditional network packet processing frameworks in networking environments. This paper aims at packet acquisition and distribution method based on Data plane development kit (DPDK) and other traditional packet processing models, which can productively decrease packet loss and successively enhance the performance rate in networking environment. It can also help in reducing resource waste in packet acquisition and distribution. Comparison between the packet processing frameworks show that Data plane development kit-based network data processing has better performance compared to Netmap Netsclice and PF-RING in faster packet processing.**

Keywords—Batch processing, dpdk, netslice, netmap, pf_ring

I. INTRODUCTION

Packet processing is a collection of steps and algorithms which are performed on an incoming or outgoing packet of information or data which keeps moving between different network entities. With the rapid and continuous growth of networking environment and evolution to future internet, the demand for the efficiency and network traffic analysis is high priority. But traditional network traffic packet processing is facing acute efficiency and performance fall due to platform hardware limitations. One more cause to performance bottleneck is due to the kernel network protocol stack low performance. Using traditional network traffic packet processing also leads to waste in network resources and storage space. So, demand for high performance solutions to network traffic processing is high.

Network communication methods have increased their performance. However, it does not guarantee high packet processing at high packet arrival rate. There are restrictions on the construction of network stacks that must be used with standard hardware. Package performance can work well by switching network stacks and without the need for very high-performance hardware. This led to research on how to integrate hardware and high paced communication systems towards package performance. 10 Gbit / s includes the highest number of packets per second, requiring high CPU usage, as well as support for processing at high packet rates. Current limitations have to be understood in order to understand what strategies can help improve the standard hardware for fast network applications. After that, the introduction of widely suggested strategies aimed at avoiding or minimizing the problems caused by these factors makes sense. Two proposed strategies, the first one is the zero-copy process, which removes more than one copy of data between the kernel and the user's location. Most fast package packaging frameworks are used. The second most recent and emerging process has identified GPU exploitation to do package processing things. Properties aimed at enabling faster packet processing are presented and how the offered strategies to be implemented.

Correspondingly, we study high performance network traffic processing technologies, which focus on enhancing the performance of network traffic processing, i.e. Netmap, Netslice, Pf_ring and DPDK.

II. RELATED WORK

[1] On Manycore systems, it uses NAPI technique to generate bunch hardware interrupts for batch processing. It provides an analysis of the ability to process on a standard operating system and identify major operational barriers in the process of processing the Linux kernel network stack and also gives insight to understanding of different operational and overtime issues that need to be come across. [2] Tests performance can be improved that can be made with the use of huge pages or large pages within applications based on the netmap. [3] This presents the potential impact on the realization of one-way communication through performance tests and provides an introductory way to one-way communication using the Commercial Off the Shelf NIC modified device driver. After that, to ensure the profitability of the Commercial Off The Shelf based on a single communication method, it presented the sample implementation using Intel 82580 NIC and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

PF_RING ZC (Zero Copy). [4] This paper mainly compared between two trends PF_RING and DPDK on processing capabilities which focused on receiving and learning more without referring to the pipeline model and the run to completion model. [5] This paper reads package processing based on the Data plane Development kit for ARM-based systems and Xilinx Zynq and sees improvements in throughput in packet forwarding. [6] This paper introduces analysis to assess the capability of the two network packet processing methods (PF_ring and Netmap) with network card models (Intel and Chelsio). [7] This paper suggests the speed of the NFV application can elevate using CPU and FPGA architecture. The acceleration sotware used here is DPDK, commonly used in network software, as a visual interface for CPU and FPGA, by setting ring operation and table viewing to better utilize the bus between FPGA and CPU.

III. PACKET PROCESSING IN LINUX NETWORK STACK: MAIN STAGES

When a network interface card (NIC) first receives a data packet, it transmits it to a circular receiving queue (RX), which are also called as rings. The data structure, the receiver descriptor, helps in holding those data packets, till the data packets are copied from NIC to the main memory. The data transfer from NIC to main memory is accomplished through the Direct Memory Access (DMA) operations, without involving CPU. Thereafter, there needs a mechanism which can help in notifying the system about the new data packet received and pass every data packet onto a specially allocated buffer called as the sk_buff struct. This data structure is assigned for each data packet and set off free when a packet enters from kernel space to user space. Disadvantage of this procedure is it consumes a lot of bus cycles. Another problem with the sk_buff structs is, it creates overhead as it designed to contain metadata for all the protocols so that it can be compatible with as many protocols as possible. But that's simply not a requisite for processing specific data packets. Because of this additional struct, processing performance is brought down. Context switching is one more factor which negatively affects performance, because it significantly consumes system resources.

NAPI was brought in in all Linux kernels since version 2.6 and it uses polling mechanism with addition to interrupts to overcome the receive live lock problem. NAPI disables the interrupt and periodically checks for the poll queue for new devices and collects data packets for packet processing. After processing of packets are finished, the cards are deleted, and interrupts.

A. *Bottlenecks*

1) More number of Bus cycles

In standard Linux kernel stack, assigning Sk_buff struct for every data packet coming from NIC to main memory consumes a lot of bus cycles.

2) Direct Memory Access and Memory operation

The operation of network data processing incorporates copying from network interface card or NIC to main memory or MM. After central processing unit or CPU access the metadata of the data packet, it encapsulates the MAC layer and forwards it to second layer. And the data movement from main memory to network card is done using the bus. Moreover, CPU cache are used more considerably because of the memory access speed. The I/O of the input port are not familiarized and if the resources are not utilized properly, the queue lock cost increases and will route into the single output queue.packet queue buffer management and memory operations

3) packet queue buffer management and memory operations

Memory management plays a vital role in any processing environment. In Linux, every allocation of a data structure requires an allocation function to be executed and network cards doesn't use pre-allocated memory buffers to store any data structures. Hence, formations of jitters are possible due to recurring memory allocations. Memory management and its operations are few of the cause.

4) Inability to parallel processing

By one, utilizing serial processing, efficiency of network packet processing decreases. The Linux network stack combine all the packets, this leads network traffic to be present in the same unit and user space created threads cannot acquire the wanted data from the queue, in turn leads to serial processing.

IV. BRIEF DISCUSSION OF TECHNIQUES FOR FASTER PACKET PROCESSING

A. *Netmap:*

Netmap is one such technology, based on zero copy, which provides packet processing at higher speeds by reducing processing slow down costs. Netmap is built on existing OS features and characteristics, which is independent of hardware and few devices. In the figure II, it depicts the Netmap framework. The data stuctures present in Netmap are Netmap ring and Netmap packet buffers.

1) Netmap packet buffers: The Netmap packet buffers are pre-allocated and with predefined size. This helps in reducing cost per packet acquisition and distribution. This data structures are present in shared memory regions to reduce the zero copy between kernel to user space.

2) Netmap ring: The Netmap rings are circular queues which possess buffer related metadata, which are alike to Network card Rings or NIC rings. The metadata carry information related to available buffers, count of slots, that a ring can hold.

3) Netmap_if: It is a data structure which possess all the interface related info, like number of rings present.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

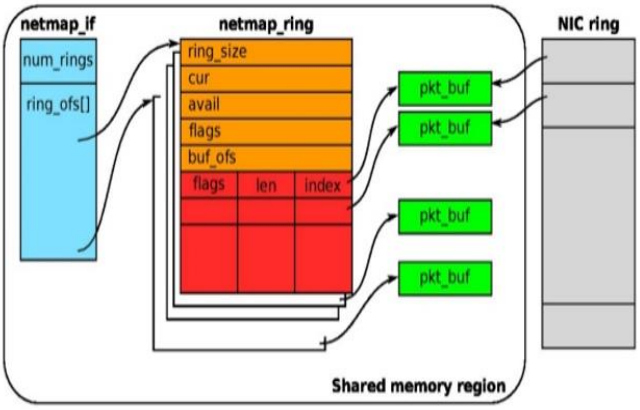

Fig. 1. Netmap Data structure

When a network card runs in this Netmap specified approach, the network card will uncouple from the protocol stack provided by host. A Netmap ring also known as NIC ring is created by the Netmap, which helps in copying. Netmap will create two pair of NIC ring or Netmap ring to communicate between the host protocol stack. The network card directly moves the data packets onto rings referred to cache use, in the shared space. The application running in user space will call a Netmap API, to access the contents of data present in NIC ring or Netmap ring can straightaway read the cache and start writing the data packet, in turn helps in less copy between kernel space to user space and provides zero copy functionality. Key memory regions of the network card register and the kernel to applications are not shared, hence there are no kernel crash caused by the application running in user space. Among frameworks which supports zero copy, Netmap tends to be is more secure.

Advantages offered by Netmap are:

a) supports multiple queue network interface cards

b) better transmission between network stack, network interface card and interfaces and

c) reduced cost

B. *NetSlice:*

Contrary to the fast packet solutions introduced in the preceding sections, this method, NetSlice is not taking advantages of zero copy methods. It is an OS abstraction, which provides faster package performance and works in the user-land. It attempts to synchronize the advantages of package processing systems operating in the user-land, e.g. the separation of error and configuration. NetSlice deploy on the advanced integration of components of hardware and software associated with packet processing. It is based on local architecture, rather than the temporary fragmentation of components of computer software and hardware suitable for package processing, namely memory, CPU cores and NICs. By making such a distinction, NetSlice reduces the conflict of resources shared. The working idea of NetSlice is its state of action, called NetSlice. In figure, a queue of NetSlices are shown. NetSlice uses a number of multi-lines NICs. To support the same practice for many believers, multi-line NICs maintain more than one transmission and receipt queue.

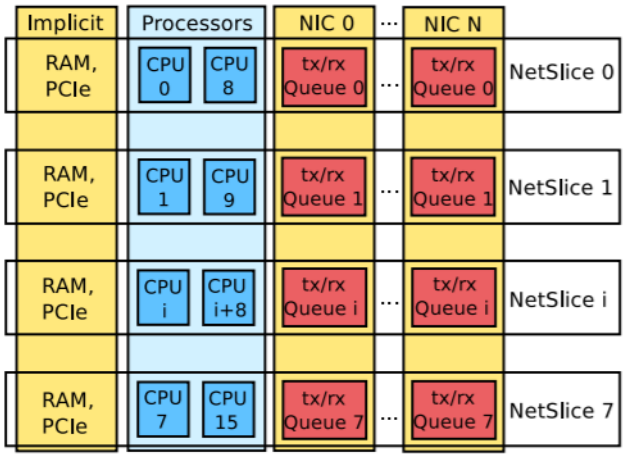

Fig. 2. NetSlice spatial partition

The context of making NetSlice consists of completely separated resources. Even the NICs and the CPU cores are treated as separate resources. At-least two CPU cores contribute to one Netslice. The kernel and user mode operations done with dividing CPU cores into u-peer and k-peer, which are Netslice output states.The u-peer CPU helps perform the user mode functionality. Whereas k-peer CPU is used for in kernel network stack. The k-peer helps detect contextual cables and also summaries the required Netslices. The Netslice help decide which data packets to flow from NICs to user-land and from user-land to NICs. The packets received from NICs are lightly processed on k-peer cores and forwarded to user-land for application use and later using pipe, packets are processed.NetSlice uses socket's standard API's to read, write and vote different streams of data for other Netslices. Using ioctl format Netslice expands the API's. NetSlice uses encryption and hence reduced delay in system calls and in number of system calls. Extended API's can be used for setting up the system calls. Batch processing helps reduce overhead caused by each packet. Netslice is not taking advantages of zero-copy operations. It does copy from kernel space to user space. Establishing zero-copy also improve NetSlice performance. However, it is not included in the frame, as it will limit the load.

C. PF_RING

PF_RING is one more such framework which implements zero-copy method between the kernel space and the user space. It was invented by Luca. It provides fast network packet acquisition and distribution framework. It attains fast packet processing with the help of PF_RING buffers, which are present in shared memory region which is common to user space and kernel space. This PF_RING buffers are pre-allocated, in which helps in reduce cost per network packet memory assignment and un-assignment.

The main element present in PF_RING'S framework are:

1) kernel module: It helps in copying data from network card to PF_RING circular queue.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

2) PF_Ring aware drivers: It helps in increasing the performance by using exclusive drivers to access the network cards.

3) PF_Ring user-space library: It helps in accessing PF_RING module present in kernel from applications running in user space.

The PF_RING uses exclusive device drivers, called as Direct NIC access or DNA, for gaining fast network data packet processing without the involvement of central processing unit or CPU, by using system calls. PF_RING uses mapping from NIC memory to user space memory and helps transmission between network card and applications running in user space. The network data packets are sent from network card to user land, without the operations of PF_RING module or Linux kernel. Then zero copy happens by doing Direct memory access from the NIC Network process unit and kernel data packet buffer is copied.

Some of the drawbacks with using PF_RING are:

a) System crash are possible due to misusing memory addresses by NIC's direct memory access.
b) Only one application can run at a time

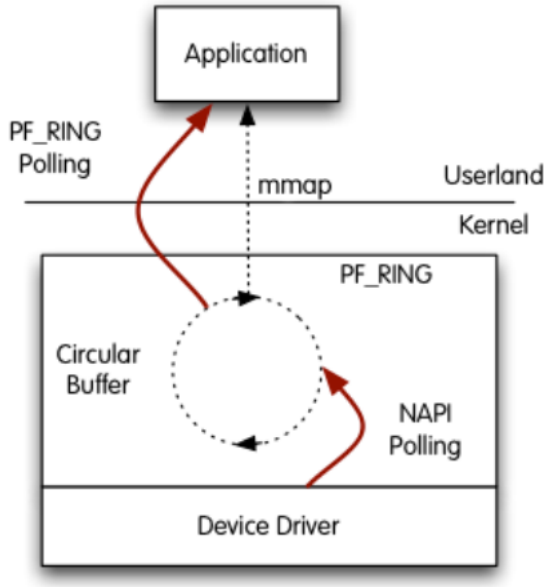

Fig. 3. PF_RING with DNA driver

D. *DPDK*

This framework provides set of libraries, developed by Intel under the open source to accelerate packet processing workloads. The DPDK framework is different from Linux kernel network stack. The DPDK framework works excellently at user space by not entering into the Linux kernel network protocol stack to decrease information copying and make full dominance over provided hardware. The applications run at user space and they use DPDK set of libraries and poll mode drivers at user space to acquire and distribute data packets, when new data packets enter network card in the system, the new data packets are transferred directly from network card to the applications running in user space without using Linux network stack.

DPDK implements either in run to completion model or pipeline model. With the system with many processing cores. The idea in run to completion model is to poll the network card and process packets and transmit from and to the same network ports, attached to the cores. In run to completion model all the cores work to complete a common application code. Whereas in pipeline model, only one core is used for receiving and transmitting the data packets from network card. It makes use of ring queue from Librte_ring to pass this data to other cores to process the packets and in pipeline model, cores can work on different application codes.

1.1) Environment Abstraction Layer (EAL): EAL builds environmental specific libraries. It acts as the abstraction Layer, so that the applications running in the user land have no clue on the environments it is running on. By using the interface provided by EAL, applications can gain access to hardware and the resources.

More specifically, services provided by EAL are

a) DPDK loading and launching.
b) Does core affinity
c) System memory allocation and reservation
d) Helps in communicating with the PCI bus
e) Interrupt handling

1.2 Core Components: The core components help in attaining faster packet processing provided by DPDK with the help of set of libraries offered.

a) Librte_eal: meant to hide system or OS specifics from common upper layers.
b) Librte_ring: It is a circular buffer, which is the queue used to pass data between threads and processes.
c) Librte_mempool: It is a memory pool manager, which sets up pool of memory buffer and uses for packet processing.
d) Librte_mbuf: It is the structure used to hold the packet information.
e) Librte_pmd: It helps in adding and configuring all the supported network cards in DPDK.
f) Librte_Timer: It is a timer manager, which uses Hpatch timers for various routines and helps in scheduling functions.

1.3 DPDK Strategies for high performance packet processing:

a) It uses pre-supplied memory bufferes called as mbuf. It stores both meta-data aswell as the actual data. And it requires only one allocation per packet.

b) Provides cores with their own cache memory and help reduce the access to the shared pool ring memory, which can be more efficient with the CPU.

c) It uses circular rings, used as a queue. It has more advantages over linked list. Besides, it reduce the time required to do more time-consuming tasks.

d) Reduces further disruption by using Poll Mode Drivers.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

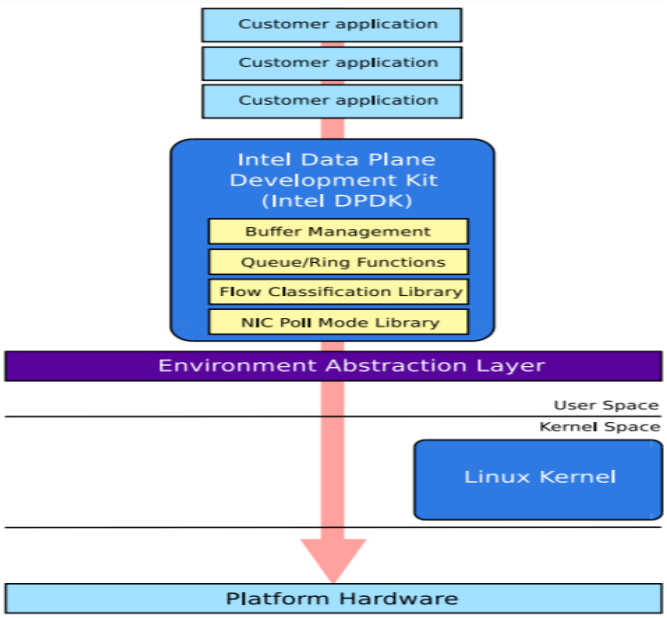

Fig. 4. DPDK major components

V. COMPARISON BETWEEN DIFFERENT FRAMEWORKS

	CPU - Affinity	User - space or kernel space	Batch - processing	Zero - copy	Safety	Support for parallelism
Netmap	Supported	user space	Supported, helps decrease overheads for large batches	Supported, it only requires swapping of Rx and Tx buffer indexes between those interfaces	Supported,	Supported, without lock it spreads load to available multiple cores by mapping between Netmap rings and NIC rings
Netslices	supported	User space, by providing simplified path between user space and NICs for the packets	Supported, by providing packet send and receive operations for bulk packets to decrease overheads	Unsupported,It copies data between kernel and user space	Supported	Supported, it slices the incoming traffic and provides array of independent environment for packet processing
PF_RING	Not supported	It uses ring buffer in the kernel space and creates mmap to share it with the program running in user space	None	Unsupported, but PF_RING ZC version uses it	Not supported	Supported, by scaling down with the no. of cores available
DPDK	supported	User-space	Supported, to support bulk operations and decrease context switch overhead	Supported	supported	Yes, by using per core storage pool to ease allocation or freeing without the use of shared variables

VI. CONCLUSION:

In this paper, we went over the internal structure and principles of software solution for high performance packet processing. we discussed the main stages of Linux network protocol stack in network packet processing. The bottlenecks of data packet processing under traditional network packet processing were discussed. Considering the inherent bottlenecks, DPDK, a software solution to a high-speed and high-performance framework is established. This framework provides significant data processing capabilities. This study and comparison between network data packet processing between Netmap, NetSlice and PF_RING techniques demonstrate that Data plane development kit is magnificent, which can be used in network packet processing and further researching.

VII. REFERENCES

[1] Ramneek;Seung-Jun Cha;Seung Hyub Jeon;Yeon Jeong Jeong;Jin Mee Kim;Sungin Jung "Analysis of Linux Kernel Packet Processing on Manycore Systems", TENCON 2018 - 2018 IEEE Region 10 Conference
[2] Aksić;Hasan Redžović;Aleksandra Smiljanić "Application of huge pages to the netmap platform, Milutin , 2017 25th Telecommunication Forum (TELFOR)
[3] Jin-Hong Kim;Jung-Chan Na "A study on one-way communication using PF_RING ZC", 2017 19th International Conference on Advanced Communication Technology (ICACT)
[4] Haipeng Wang;Dazhong He;Huan Wang "Comparison of highperformance packet processing frameworks on N UMA", 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)
[5] Jan Viktorin;Jan Korenek, "Packet Processing on FPGA SoC with DPDK", 2016 26th International Conference on Field Programmable Logic and Applications (FPL)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

[6] Paulo Rocha;Thiago Pinheiro;Ricardo Macedo;Francisco Airton Silva, "10GbE Network Card Performance Evaluation:
A Strategy Based on Sensitivity Analysis", 2019 IEEE Latin-American Conference on Communications (LATINCOM)

[7] Yoshikazu Watanabe;Yuki Kobayashi;Takashi Takenaka;Takeo Hosomi;Yuichi Nakamura, "Accelerating NFV application using CPU FPGA tightly coupled architecture", 2017 International Conference on Field Programmable Technology (ICFPT)

[8] Wenjun Zhu;Peng Li;Baozhou Luo;He Xu;Yujie Zhang, "Research and Implementation of High Performance Traffic Processing based on Intel DPDK", 2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)