Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 3, March -2017

DPDK-Based Implementation Of Application : File Downloader

Prof. Anup Kadam[1],Vinay Singh[2], Rituraj Singh[3], Virendra Singh Rawat [4], Sandeep Kumar Singh[5]

[1,2,3,4, 5]*Department of Computer Engineering, Army Institute of Technology, Pune, India*

Abstract — *Implemented a file downloader using the DPDK network interface for rump kernel. The combined result is a userspace TCP/IP stack doing packet I/O via DPDK. DPDK is a framework used to provide a simple, complete framework for fast processing of packets in data plane development applications and the framework creates a set of libraries for specific environments. The DPDK implements a model known as run to compilation for processing of packets, where all resources must be allocated before processing pakects by calling Data Plane applications, running on logical cores as execution unit.DPDK also uses a pipeline model which passes packets or messages between different cores via the rings.*

Keywords-- *Qemu/KVM,DPDK(Data Plane Development Kit),Rump Kernel,Open v-Switch,TCP/IP Stack*

I. INTRODUCTION

DPDK is used to provide complete framework for fast processing of packets in data plane applications[1]. DPDK framework creates an Environment Abstraction Layer (EAL) with the help of set of different libraries for specific environments, which is mainly be specific to a mode of the Intel architecture, Linux user space or a specific platform [1]. Make files and configuration files are used to creating and building these environments. To create applications using DPDK,once the EAL library is created, user links his application with the EAL library [1].
The DPDK implements a model known as run to completion model for processing of packets [1]. DPDK also uses a pipeline model which passes packets or messages between cores via the rings. This allows different types of work to be performed in stages via pipeline and may allow more efficient use of code on cores. Interrupts are not used in this model because of the performance overhead imposed due to interrupt processing.
For DPDK enabled application a DPDK network interface for rump kernel is created and the combined result is a user space TCP/IP stack doing packet I/O via DPDK.A rump kernel employs a mechanism for taking an monolithic operating system kernel(existing), leaving everything out except drivers, and those drivers are used as a library components.

II. GOALS AND OBJECTIVE

The main goal of this project is to improve the performance of network application by fast packet processing using Data Plane Development Kit and better utilization of resources.At the end we will analyse and compare the Performance of Network Application working on traditional environment and a DPDK enabled environment.

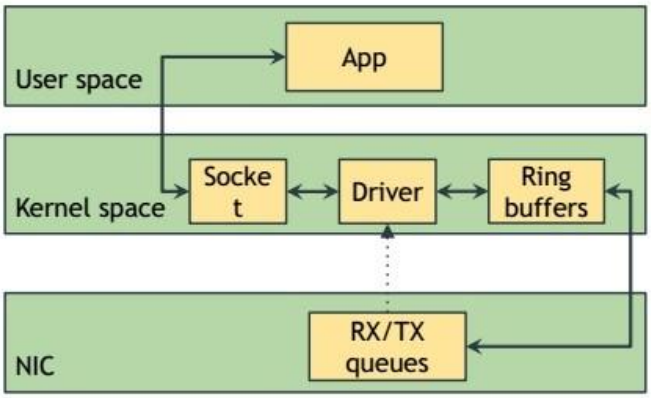

Figure 1. Packet Processing in Linux

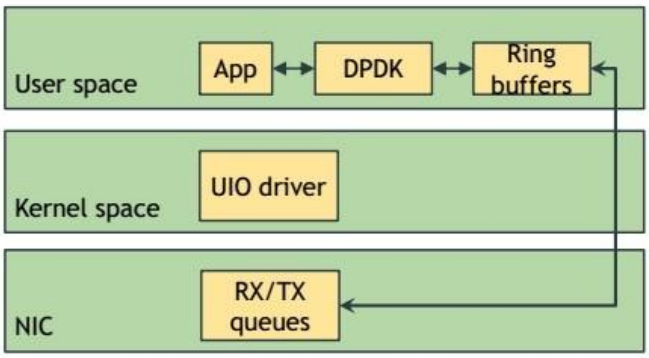

Figure 2. Packet Processing in Linux using DPDK

III. SYSTEM REQUIREMENT

For the most of platforms, no special type of BIOS settings are needed to use basic DPDK functionality [1] . The Kernel version that is used should be 2.7.34 or newer and glibc library version should be also needed to be 2.8 or latest. In the Fedora Operating System and other common distributions os, such as Ubuntu, or Red Hat Enterprise Linux, the vendor supplied kernel configurations can be used to run most of the DPDK applications. For other kernel builds UIO support and HUGETLBFS should be enabled. For the large memory pool allocation used for packet buffers Hugepage support is required.

IV. ECOSYSTEM SETUP

Creation an Ecosystem means installing and configuring following elements.

1. Building the DPDK 17.02 Target for OVS. [2]
2. Building Open vSwitch with DPDK-17.02. [2]
3. Create Open vSwitch DataBase and Start Daemon 'ovs-vswitchd'. [2]
4. Configuring for OVS-DPDK Usage the Host. [2]
5. KVM/QEMU: Kernel-based Virtual Machine(KVM) is a virtualization framework that is used for the kernel that turns it into a hypervisor. We have used Quick Emulator(QEMU) to create two virtual machines for running our applications. QEMU is fast, para-virtualized hypervisor emulator running with KVM.Because of Software and Hardware Limitations of standard systems VMs are created, as DPDK is very heavily Intel hardware reliant.
6. Creating Virtual Machine using DPDK enabled ports with QEMU-KVM. [2]
7. Ecosystem Architecture: Following is the system which is developed after setup (figure 3).

V. BENCHMARKING OF ECOSYSTEM

iPerf is a tool used for active measuring of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (UDP, SCTP, TCP, with IPv4 and IPv6). The iPerf tool reports the network bandwidth, packet loss, and other parameters like speed of data flow. For benchmarking of ecosystem iPerf is used. Iperf is run on client side and on server side there are two ports: one is DPDK enabled and one is without DPDK enabled. Server side is ping from client side and speed is measured using iPerf tool.

Sr. No.	Parameter	Before	After
1	Bandwidth (in percent)	1.8(GBits/sec)	3.36(GBits/sec)
2	Transfer in 0-10sec	1.88(GBytes)	3.91(GBytes)

Table 1. Benchmarking Metrics using iperf Tool

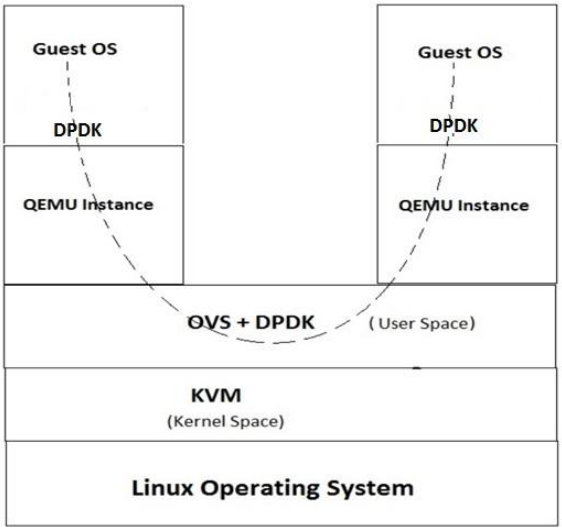

Figure 3. Ecosystem

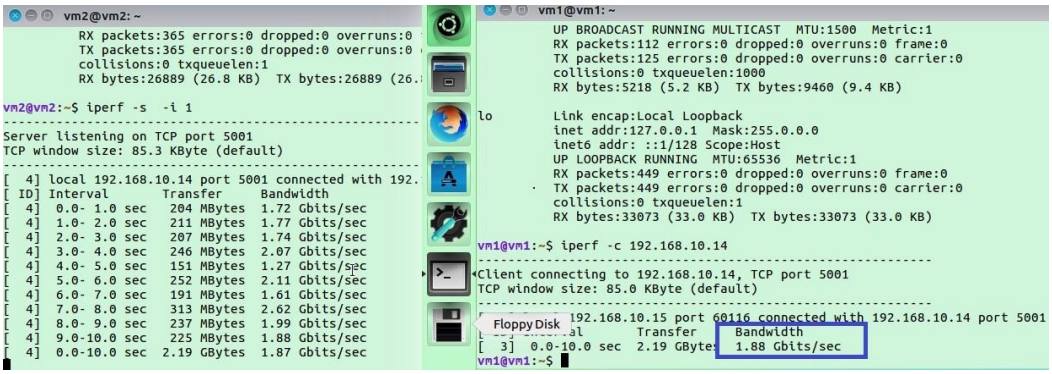

Figure 4. iperf benchmarking for VM without DPDK

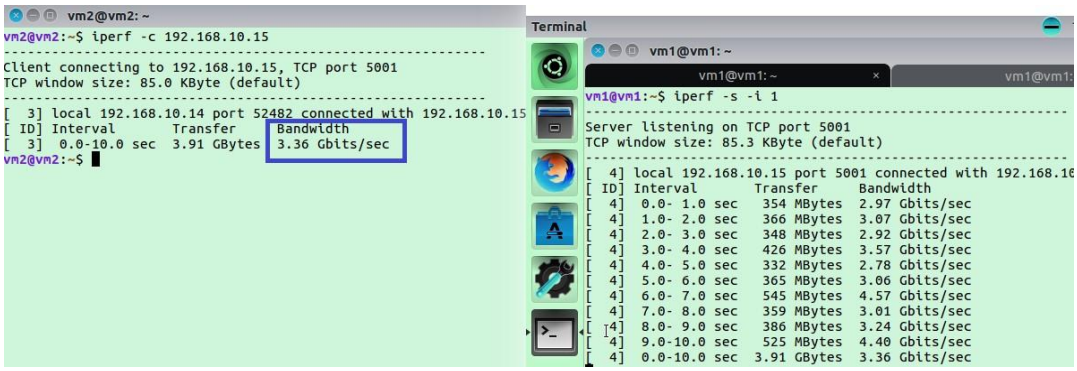

Figure 5. iperf benchmarking for VM with DPDK

VI. IMPLEMENTATION

Using Qemu/KVM two Virtual Machines , VM1 and VM2 are created .On VM1 a traditional environment is set and a HTTP downloader without DPDK is run , i.e. packet processing is being done in kernel space .

6.1. Compiling Rump-Kernel with DPDK

Rump kernel is a light weight kernel which has free, componentized,reusable, kernel quality drivers such as PCI device drivers, file systems, POSIX system calls , TCP/IP and SCSI protocol stacks[3]. Unlike unix kernel, rump kernel makes

hypercalls directly to the hypervisor. On VM2 to run our DPDK enabled application the port which is being used by the application is set down using the following command:

ifconfig eth0 down

Rump kernel and DPDK are compiled on VM2 and the application port is bound to DPDK using the following command:

sudo modprobe uio_pci_generic

sudo modprobe uio sudo

insmod dpdk/build/kmod/igb_uio.ko

insmod dpdk/buildkmod/rte_kni.ko

sudo dpdk_nic_bind.py -b igb_uio eth0

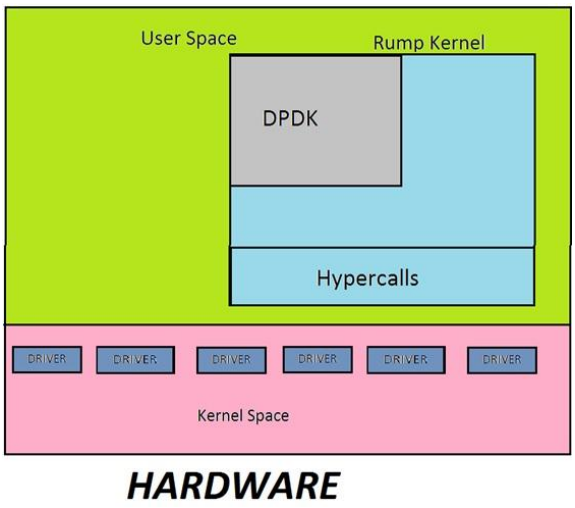

Figure 6. Architecture of Rump Kernel

6.2. Application Development: File Downloader

Two application i.e. HTTP Downloader, one using DPDK libraries and one without it are developed. The application which is developed without using DPDK libraries is happening there. Using DPDK libraries and rump kernel a HTTP downloader is developed to run on VM2. This application is running on a VM2 compiled as a client. Apache server is running on Hypervisor(Host Machine) where file to be downloaded is kept.

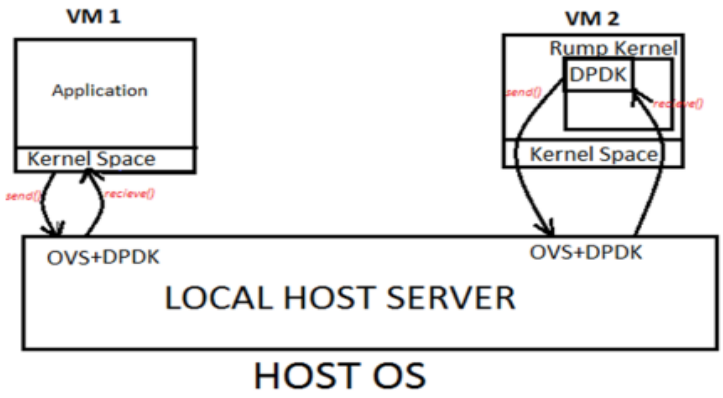

Figure 7. Architecture of file downloader

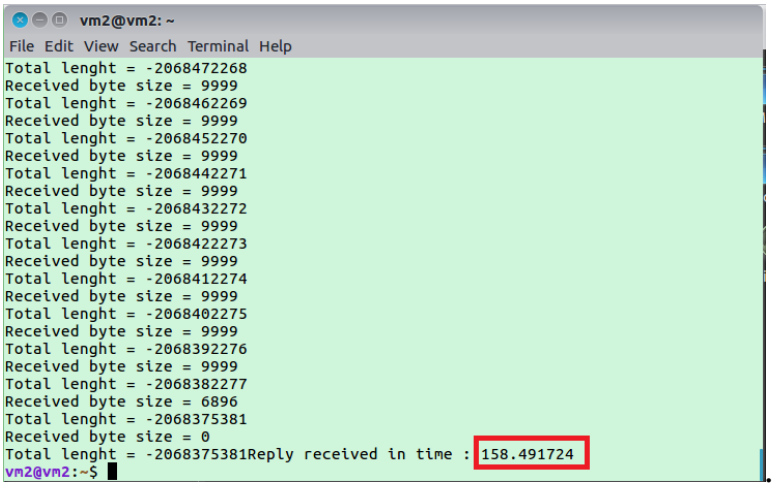

Figure 8. File Downloader Using Traditional Socket Programming

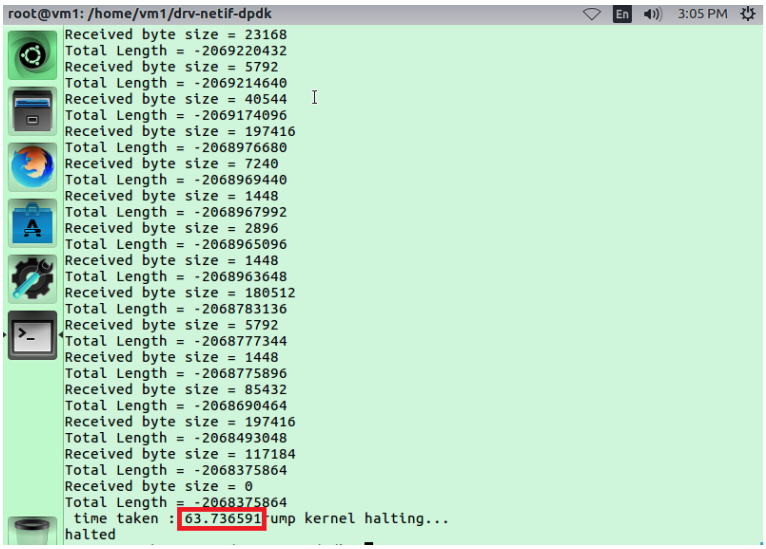

Figure 9. File Downloader Using DPDK network interface for rump kernels

VII. ANALYSIS

Drv-netif-dpdk is used for building a DPDK network interface for rump kernels [4]. The combined result of it is a TCP/IP stack doing packet I/O via DPDK [4]. It is used to build a DPDK enabled file downloader and traditional file downloader is build using traditional socket programming in C.After successful execution of above two application on two separate ecosystem performance analysis is done over time taken to download different file size and we found approximate of 3x times performance improvement.

Sr. No.	File Size	Traditional Downloader(sec)	DPDK enabled Downloader(sec)
1	2.2 GB	158.4917	63.7365
2	4.6 GB	314.5348	117.4863
2	14.4 GB	1151.5348	347.5789

Table 2. Benchmarking Metrics of File Downloader Applications

VIII. CONCLUSION

By using the Intel DPDK library on a common platform, we can experience faster network packet processing, potentially reduce cost by simplifying the hardware to industry standard server architectures, conserve energy by using power-optimized Intel platforms, increase efficiency by maximizing the utilization of your existing environment.

REFERENCES

1. http://dpdk.org/
2. https://software.intel.com/en-us/articles/using-open-vswitch-with-dpdk-for-inter-vm-nfv-applications
3. http://rumpkernel.org/
4. https://github.com/rumpkernel/wiki/wiki/Repo:-drv-netif-dpdk

