Performance Analysis of Ethernet Packet Transmission in a Multi-kernel OS

Qitao Xu

Washington University in St. Louis, St. Louis, Missouri, 63130
[1]qitao@wustl.edu

1 Introduction

Multi-kernel environment means that hardware resources of a node can be decomposed into partitions that are fully managed by independent system software stacks. Since Multi-kernel environment can not only provide performance isolation for applications running on separate operating system to keep performance consistent in a single computing node, but also offer customized operating system and hardware resources to each application to achieve optimal performance, respectively, Multi-kernel environment has emerging as an active area for High Performance (HPC) application execution.
Since Multi-kernel environment is designed to make operating systems operating in separate address spaces to provide performance isolation, processes on separate Multi-kernels cannot communicate with each other. However, such communication is required to specific applications. For example, if a HPC application is running on an independent kernel and generates data simultaneously while another application is running on another independent kernel in the computing node to make data analysis for the HPC application in real time, communication between these two applications are required. And we noticed that traditionally, two virtual machines hosted by Linux kernel can have communication with each other by taking advantage of Linux Bridge and TAP device. Basically, Linux Bridge is a Linux Kernel Module, acting as a virtual switch. Even though it is a virtual switch, both physical Ethernet port and virtual Ethernet port can be added to Linux Bridge. Terminologically, we call virtual Ethernet ports added on Linux Bridge as tap device or tap interface.

However, communication between processes

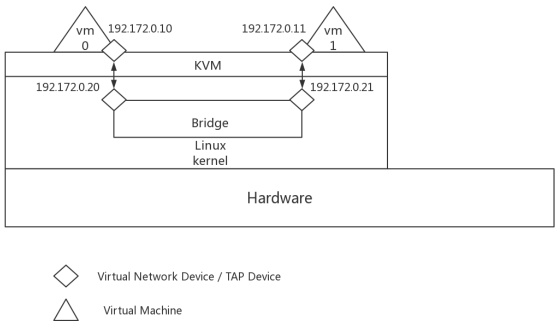

Fig. 1. Communication Model through Linux Bridge and TAP Device

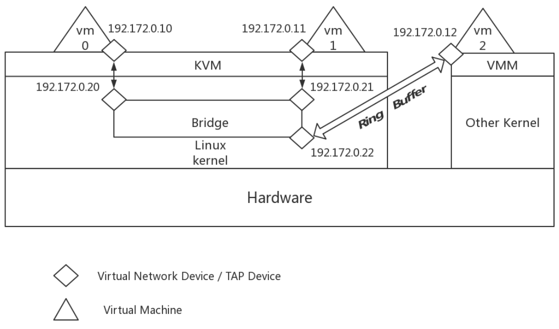

Fig. 2. Ring Buffer Objects managed by XEMEM

on separate Multi-kernels cannot be achieved only through Linux bridge and TAP device in Multi-kernel environment. Since virtual machines can be hosted by other hypervisor whose kernel is not Linux kernel, those virtual machines cannot get access to TAP devices and Linux Bridge to

realize cross-enclave communication. To fully take advantage of Linux Bridge and TAP device communication model in Multi-kernel environment to realize communication between processes on separate Multi-kernels, ring buffer objects are implemented to build a link between TAP device and virtual network devices of virtual machine hosted by arbitrary kernel. And those ring buffer objects are managed by a kernel module, we call it XEMEM [1]. And in this project, we perform a performance evaluation of this mechanism to realize cross-enclave communication.
Consequently, to verify if performance of cross-enclave communication can be approaching to traditional communication model through Linux Bridge and TAP device, in this report, iperf benchmark is used to testify the performance of Ethernet packet transmission in a muiti-kernel operating system. Specifically, we will evaluate a packet transmission system that leverages Linux TAP devices and bridge forwarding framework in the context of the Hobbes multi-kernel OS. There are four cases we seek to evaluate: (1) communication between virtual machines on Linux kernel; (2) communication between virtual machines on Linux kernel and lightweight Kitten kernel; (3) communication between virtual machines on the same lightweight Kitten kernel; (4) communication between virtual machines on different lightweight Kitten kernel.

2 Background

Kitten Lightweight Kernel
Kitten [2, 3] is an operating system to better leverage multicore processors and hardware virtualization to provide an efficient environment for massive parallel HPC application execution.

Palacios Virtual Machine Monitor
Palacios [1, 3], designed for embedding into multiple host operating system, is an open source and OS-independent virtual machine monitor capable of virtualizing existing and unmodified applications without overhead of porting. The combination of Palacios and Kitten can be a lightweight hypervisor to support virtualization of full system.

Pisces Lightweight Co-kernel Architecture

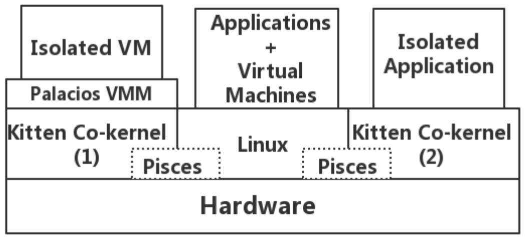

Fig. 3. The Pisces Co-kernel Architecture [3]

Co-kernel means a node's hardware resources can be partitioned into parts independent OSes are running on, respectively. And Pisces allows mutiple Kitten kernels instances running on the same node alongside an unmodified Linux host operating system.

Pisces
Pisces [1] is a kernel module that allows a node to deploy multiple co-located Kitten instances as multi-kernels executing alongside an unmodified Linux host OS.

Enclave
Enclave [1] is an exascale OS/Rs composed of a wide range of heterogeneous hardware and software environments, where each enclave is specified in supporting a particular class of exascale workload, but still remains strictly isolated from the other enclaves on the system.

Linux Bridge, TAP
Linux Bridge is a Linux Kernel Module, acting as a virtual switch. Even though it is a virtual switch, both physical Ethernet port and virtual Ethernet port can be added to Linux Bridge. Terminologically, we call virtual Ethernet ports added on Linux Bridge as *tap device* or *tap interface*. When a virtual machine is created, a corresponding tap device on behalf of this virtual machine can be also created and added to Linux Bridge. Hence, if a Client process on a virtual machine 0 wants to communicate with Server

process running on another virtual machine 1, it can put data into its virtual network device and these data can be paketized as Ethernet frame and sent by vm_0_kernel thread to its tap device. And such frame can be forwarded by Linux Bridge to tap device on behalf on virtual machine 1. There is another process, v3_tap_rx, forwarding this frame to the virtual Ethernet port of virtual machine 1.

XEMEM
XEMEM [1]: a shared memory system that can efficiently contruct memory mappings across enclave OSes to support composed workloads while allowing diverse application components to execute in strictly enclaves.

3 System Architecture

In this section, we present four different system architecture on which we have computer network performance test and computer network communication mechanism, respectively.

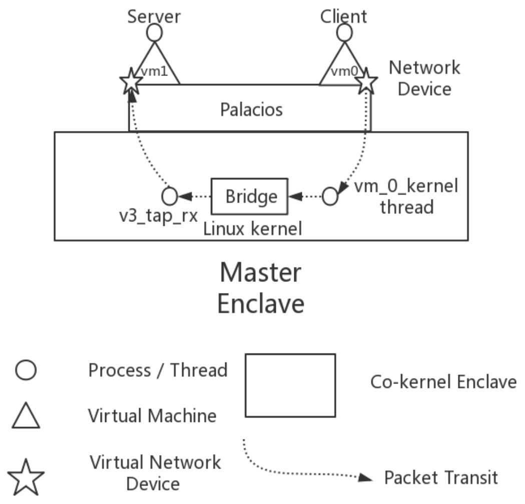

Fig. 4. Case LNX configuration

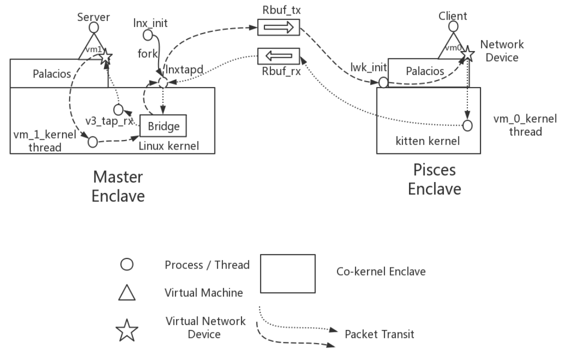

Fig. 5. Case LNX-LWK configuration

3.1 Case LNX: Two vms hosted by one Linux enclave

In this configuration, two virtual machines are hosted by Master enclave whose kernel is Linux kernel. Hence, two virtual machines rely on Linux bridge to realize communication. Similarly, message from process Client running on virtual machine 0 is passed to its virtual network device. And vm_0_kernel thread transits this message to Linux bridge. v3_tap_rx on behalf of virtual machine 1 receives message and passes it to virtual network device of virtual machine 1.

3.2 Case LNX-LWK: Two vms hosted by Pisces enclave and Linux enclave

In this configuration, one virtual machine is hosted by Pisces enclave while the other is hosted by Master enclave, whose kernel is Linux kernel instead of kitten kernel. First, process Client running on virtual machine 0 passes message with IP address of virtual machine 1 to its virtual network device. Second, vm_0_kernel thread takes this message through virtio and put it into Rxbuf_rx, which is a ring buffer serving for virtual machine 0 to queue messages whose source is virtual machine 0. Third, a process running on Master enclave takes message from ring buffer and transmits it to Linux bridge. Fourth, Linux bridge passes message to another process, v3_tap_rx, which receives packets on behalf of virtual machine 1. Finally, this message can be passed to virtual

network device through virtio and read by Server process running on virtual machine 1.
However, it is a different story to pass message back. First, message with IP address of virtual machine 0 is passed to network device and taken by vm_1_kernel thread through virtio. Second, this message is passed to Linux bridge and according to its destination IP address, it is put to Rxbuf_rx by process lnxtapd, where Rxbuf_rx is a ring buffer serving for virtual machine 0 to contain messages whose destination is virtual machine 0. Third, a process lwk_init serving to receive packets on behalf of virtual machine 0 can take message and pass it to virtual network device of virtual machine 0. Finally, this message can be read by process Client running on virtual machine 0.

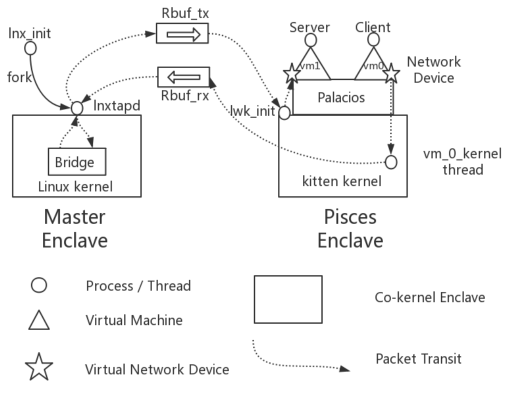

Fig. 6. Case LWK configuration

3.3 Case LWK: Two vms hosted by one Pisces enclave

In this configuration, two virtual machines are hosted by one Pisces enclave and they have to take advantage of rxbuf, a ring buffer data structure, to pass messaages via Linux bridge in Master enclave. For example, a userspace process , Client, on virtual machine 0, is trying to pass message to virtual machine 1 according to IP address of virtual machine 1. First, Client passes message to its network device implemented by virtio. Through virtio, this message is handled by a kernel thread, vm_0_kernel thread. Second, this kernel thread puts passed message to a ring buffer exclusively serving for virtual machine 0, Rbuf_rx. Third, a process forked by lnx_init running on Master enclave, lnxtapd, takes this message in Rbuf_rx, and passes it to a tap device on behalf of virtual machine 0 and attached on Linux bridge in Master enclave. Linux bridge can pass message to another tap device according to destination IP address pointing to virtual machine 1. Third, lnxtapd process can handle this message from tap device and put it to another ring buffer, Rbuf_tx, exclusively serving for virtual machine 1. Fourth, a userspace process running on Pisces enclave, lwk_init, takes message from Rbuf_tx and passes it to network device of virtual machine 1. Finally, this message is read by a process running on virtual machine 1, Server.

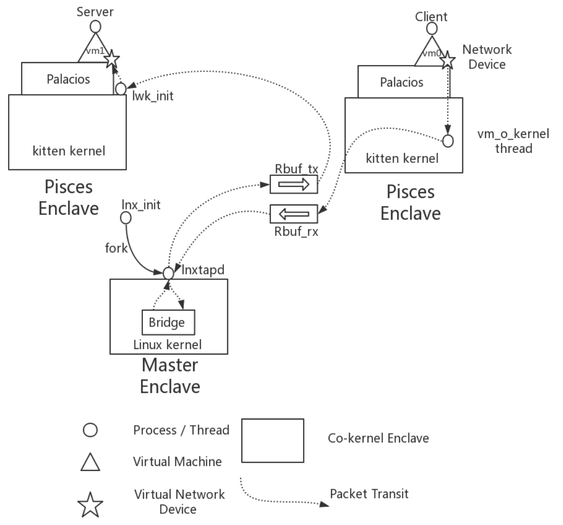

Fig. 7. Case LWK-LWK configuration

3.4 Case LWK-LWK: Two vms hosted by two Pisces enclaves

In this configuration, two virtual machines are hosted by two Pisces enclaves, respectively and they also pass messages to each other via Linux bridge with the help of rxbuf. It is similar to case 0, but after message is put into Rbuf_tx serving for virtual machine 1, lwk_init, a process listening to

Rbuf_tx, takes message and passes it to network device of virtual machine 1 on another Pisces enclave.

4 Experiments and Results

In this section, we take advantage of iperf to testify the performance of communication between virtual machines in four configurations via XEMEM and Linux bridge packet forwarding.

4.1 Network Performace over UDP

In network performance test over UDP protocol on four cases, we record three parameters: datagram loss rate, bandwidth and jitters when change client side sending rate and datagram size.

4.1.1 Datagram Loss Rate

If there is at least one virtual machine hosted by Linux enclave, over UDP, datagram loss rate can be 0 even if sending rate increases, which means that although data sent increase in given time interval whose default value is 10 seconds, datagram loss rate can stay 0 even over UDP. However, as long as two vms are both hosted by Pisces enclaves, no matter hosted by the same enclave or two different enclaves, datagram loss rate can be at most 74 percent. But if we increase datagram size, such issue can be solved.

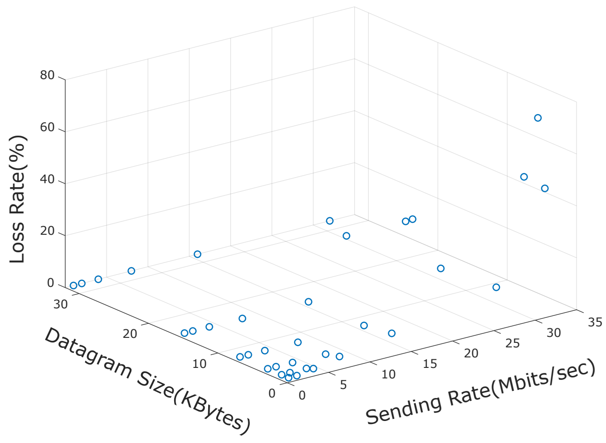

Fig. 8. Case LWK Datagram Loss Rate over UDP

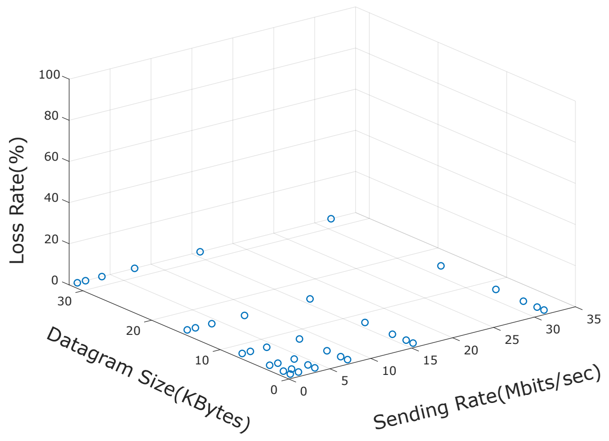

Fig. 9. Case LNX-LWK Datagram Loss Rate over UDP

4.1.2 Bandwidth

Based on argument of datagram loss rate, it seems that the larger datagram size is, the more confidence we can have to guarantee datagram loss rate to be 0. However, when datagram size is set to 64k bytes, there is no data can be received in the other end.
And when datagram size increasing, the bandwidth can be decreased. Hence, we have to find a good datagram size which is the smallest size among those who can have 0 datagram loss rate given specific sending rate.

4.1.3 Jitters

If two virtual machines both hosted by Pisces enclave, no matter hosted by the same enclave or two different enclaves(case LWK and case LWK-LWK), jitters value can be much larger than that where at least one virtual machine hosted by Linux enclave. And if two virtual machines are hosted by Linux enclave, the jitters value is even smaller than that where one virtual machine is hosted by Linux enclave while the other is hosted by Pisces enclave.
If two virtual machines both hosted by Pisces enclave, no matter hosted by the same enclave(case LWK) or two different enclaves(case LWK-LWK), when sending rate increases to be 32Mbits/sec, it's likely that jitters value can be much larger than others. And with larger packets,

we can partly decrease jitters value. However, there are also outliers where lower datagram size with good jitters value when sending rate is 32 Mbits/sec.

However, in case LNX-LWK, where one virtual machine is hosted by Linux vm while the other is hosted by Pisces enclave, when datagram size becomes 32 KBytes, jitters value will fluctuate.

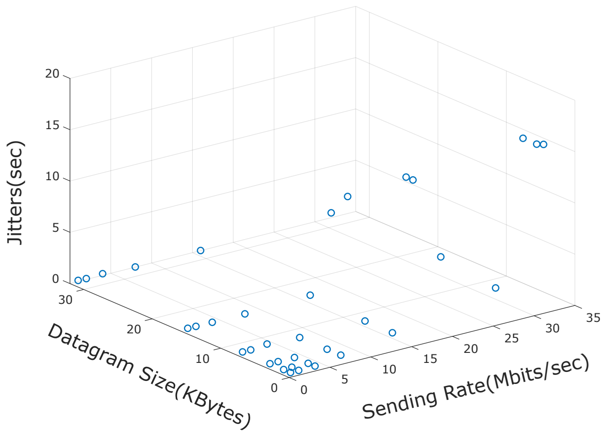

Fig. 10. Case LWK Datagram jitters over UDP

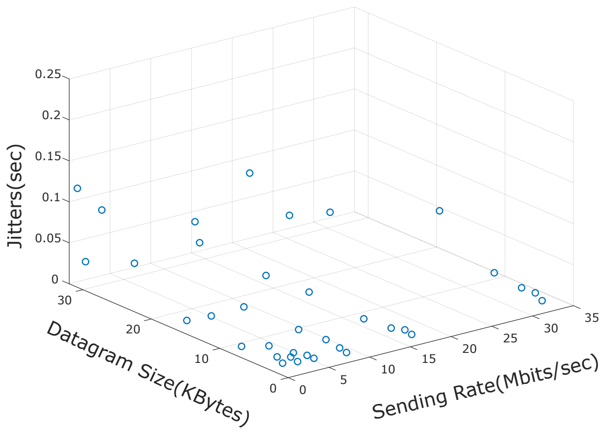

Fig. 11. Case LNX-LWK Datagram jitters over UDP

4.2 Network Performance over TCP

For TCP experiment, we found that when two virtual machines are both hosted by Pisces enclave, they can have approaching bandwidth to control group, where two virtual machines are both hosted by Master enclave, whose kernel is Linux kernel.

The most interesting finding is that case LNX-LWK, where one vm is hosted by Linux enclave while the other is hosted by Pisces enclave, the bandwidth it can achieve is even better than control group. And we propose a hypothesis that since kernel of Master enclave is Linux kernel, to Master enclave, two virtual machines running on it have no difference with other processes running on it. Hence, the competition of all processes on Master enclave may decrease the performance of our control group compared to case LNX-LWK, where only one virtual machine is running on Master enclave.

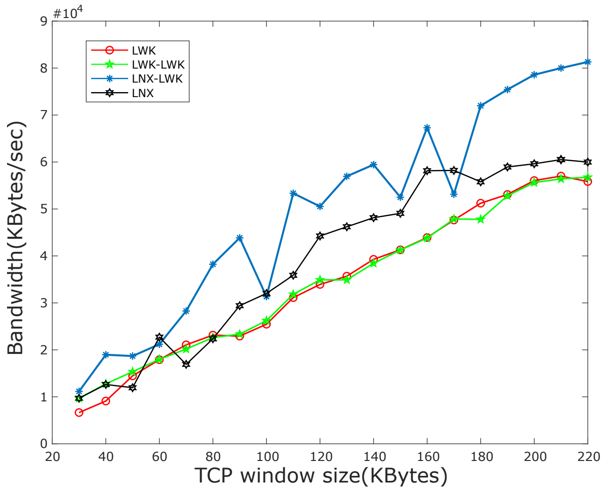

Fig. 12. Bandwidth over TCP

5 Conclusion

According to our experiments, we can draw a conclusion that performance of cross-enclave communication taking advantage of XEMEM and Linux Bridge in different system configuration can be similar to traditional communication method through Linux Bridge, especially over TCP protocol in transport layer, although our solution for cross-enclave communication has some disadvantages such as UDP datagram loss rate can be not zero in specific scenario.

6 Future Work

For UDP datagram loss rate, we want to increase ring buffer size to see what will happen since right now the size of all ring buffer is fixed. And when sending rate is too large, it is likely that ring buffer is full and drop incoming packets. Hence, increasing size of ring buffer may decrease datagram loss rate.
For TCP bandwidth, we are also glad to design experiments to verify if our hypothesis is correct that since unmodified Linux kernel takes virtual machine instances running on it as normal processes like any others, competition of all processes may decrease performance of cross-enclave communication compared to that only one virtual machine is deployed in Master enclave even if another virtual machine has to get access to Linux Bridge via XEMEM.
Besides, if more than one cross-enclave communication instances happen simultaneously, it is unknown whether performance of cross-enclave communication will decrease. Since there is only one process running on Master enclave to serve all ring buffer objects, it would be a bottleneck for multiple cross-enclave communication happening simultaneously.

7 Acknowledgement

I would like to express my sense of appreciation to Professor Kocoloski. Without his generous help, technological guide and moral support, I cannot finish this Master Project. Hence, I want to say ”Thank you” sincerely here.

References

1. Brian Kocoloski and John R. Lange. Xemem: Efficient shared memory for composed applications on multi-os/r exascale systems. In *HPDC*, 2015.

2. John R. Lange, Kevin T. Pedretti, Trammell Hudson, Peter A. Dinda, Zheng Cui, Lei Xia, Patrick G. Bridges, Andy Gocke, Steven Jaconette, Michael Levenhagen, and Ron Brightwell. Palacios and kitten: New high performance operating systems for scalable virtualized and native supercomputing. *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1–12, 2010.

3. Jiannan Ouyang, Brian Kocoloski, John R. Lange, and Kevin Pedretti. Achieving performance isolation with lightweight co-kernels. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC ’15, pages 149–160, New York, NY, USA, 2015. ACM.