International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 22651-22662

http://www.ripublication.com

Enriching Packet Aggregation Mechanism Based on Cloud Services I/O Virtualization

Dr. S. Selvakani[1], J.Nelson [2]
[1] Professor and Head, [2] PG Scholar
Department of Computer Applications, Francis Xavier Engineering College, Tirunelveli, Tamilnadu, India

Abstract

The key technology that power's cloud computing virtualization. Aggregation algorithm combine several small packets into larger packet and forward to an aggregation target. In the network I/O virtualization model, to increase the throughput and decrease the delay, a packet aggregation mechanism is used. It reduces the memory creation in virtual machine monitor. Because the packets are combined using the MAC address of the packets. A survey of the packet aggregation mechanism has been analyzed in this paper.

Keyword: Virtualization, XEN, Driver Domain, Cloud Computing, Networking Performance

Introduction

Cloud computing is changing the way of computer world operates and it's a profit by providing infrastructure and software as chargeable services delivered over the internet. Virtualization enables the cloud computing. Virtualization helps to achieve greater system utilization and total cost of ownership, responding more effectively to changing conditions in government and organizations. Virtualization technique to run multiple and isolated virtual machines on a single physical machine.

The driver domain is a special virtual machine that is in charge of managing the shared access to the devices, especially the network interface card(NIC).The driver handles networking by multiplexing incoming traffic. The additional layer in the packets path produces an extra overhead. The I/O mechanism of virtual machine monitor consists in copying the packet to the shared memory between the driver domain and the virtual machine.

We proposed packet aggregation for networking performance enhancement. In the packet aggregation mechanism the detail study of throughput and delay jitter are provided.

Terminologies:

Virtualization
In computing refers the creation of virtual. Version of operating system, Storage devices.

Network Virtualization
Network Virtualization is the process combining hardware and software network resources and network functionality into a single , software-based administrative entity a virtual network.

Packet Aggregation
Packet aggregation is the process of joining multiple packets together into a single transmission unit. In order to reduce the overhead associated with each transmission.

Queuing Theory
Queuing theory is a collection of mathematical models of various queuing system. exceeds the capacity Queues or waiting lines are arise when demand for a service facility of that facility.

Packet Aggregation Algorithm
The packet aggregation algorithm used to aggregate the packet based on the MAC address. Based on the network we can assign the MAC address for packet and then aggregate the packet.

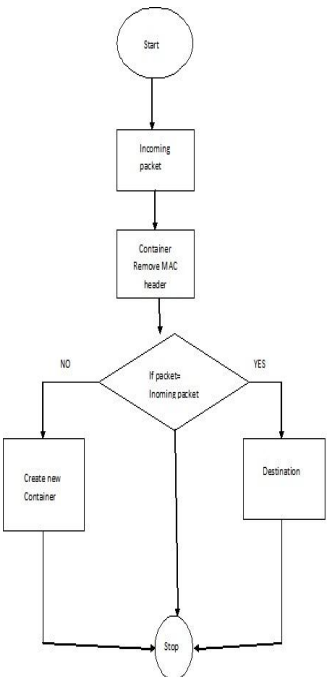


Figure: Flow chart for packet Aggregation

Assume that packets are transferred from the driver domain to virtual machine. Packet aggregation is based on the MAC destination address. The packet aggregation is divided into two parts. They are container and unloader. Based upon the incoming packet the container generator module remover its MAC header and it checks if a container with same MAC destination is already waiting to be transferred to the net front. The container generator module checks the total size of the container. It examine the maximum allowed size of the container after adding the incoming packet, the packet is transferred to the destination through the shared memory. Then the generator creates a new container with the same MAC header of that incoming packet.

Packet Aggregation Model

Packet aggregation algorithms combine several small packets into one larger packet and forward to an aggregation target. It is used to construct trains of packets. The packet aggregation mechanism consists of two modules. They are container and module. Containers used to transfer the packets between netback and net front in both directions. In this method more data can be transferred with less memory request for memory grants and less copies and less notifications. In packet aggregation methods transferred packets first assembled in containers. Then containers are transferred through the event channel to destination. Packet aggregation is based on their MAC destination address. The original packets will be retrieved in the destination. The packet aggregation is then performed in one side and their extraction in the other one.

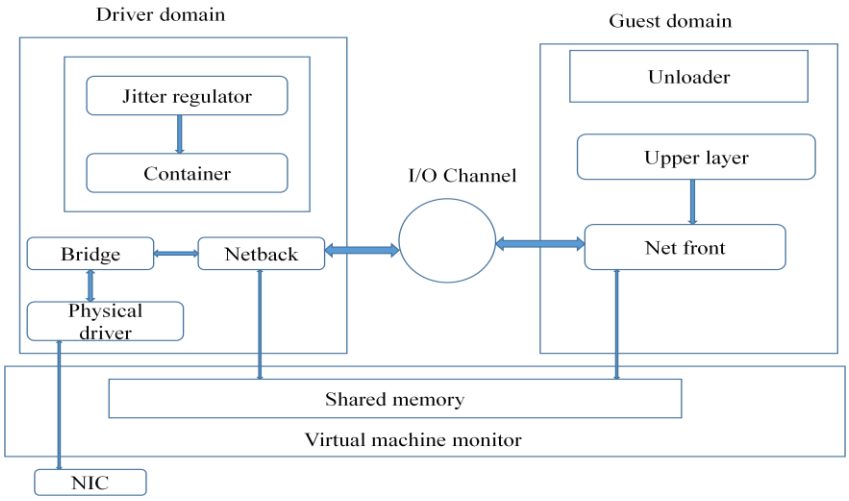


Figure: Packet Aggregation Model

Figure shows the architecture of packet aggregation model. It consists of two modules container and unloader. When the network interface card has the packets it raises the interrupt, if the physical driver needs the interrupt of the NIC directly it must pass through the virtual machine monitor. So the virtual machine monitor raises the virtual interrupt for the NIC in the driver domain. Upon receiving the virtual interrupt the physical driver sends the packet to the Ethernet Bridge. The Ethernet Bridge demultiplexes the packets based on its Ethernet address and delivers to the

appropriate netback interface. The netback driver sends the packet to the net front, over an I/O channel.

Packets aggregation is based on their MAC destination address. Upon the arrival of a packet, the container generator module removes its MAC header and checks whether a container with that same MAC destination is already waiting to be transferred to the net front. The container generator checks whether the total size of the container would exceed the maximum allowed size of the container after adding the incoming packet. The container is transferred to the destination through the shared memory according to event channel mechanism. In the case where no container is available to carry the packet, the container generator creates a new container with the same MAC header of the arriving packet. The Unloader is the component that unloads the received container at the destination to extract the packets. It removes its MAC header and retrieves the packets one by one based on their offset. Packets are then processed by the upper layer. Packets transferred from the VM to the driver domain follow the opposite path.

Software Router Architecture

Software router architecture [7] that parallelizes router functionality both across multiple servers and across multiple cores with single server. In current networking equipment the important goal are high performance and programmability. Next goals are high-end routers, because they rely on specialized and closed hardware and software, and they are difficult to expand and program. Software routers perform packet processing in software running on general-purpose platform. They are easily programmable. But they are only suitable for low-packet rate environments.

There are multiple challenges in building a high speed out of pc. They are performance, power and space consumption, then choosing the light programming model .Para virtualization across servers allows us to incrementally scale out router capacity by adding more servers. The role of the router is to receive the packets arriving at all thee ports, process them and transfer each incoming packet from its input port to the corresponding output port. The routers functionality can be divided into two tasks. They are packet processing and packet switching. Existing software routers, follow an single server as router. The requirement of parallelism is to allow each individual server can met with existing or at least upcoming server models. The server and link technology narrow options to solutions based on load-balanced interconnects. In this each node can independently make packet routing and dropping decisions for a subset of the router's overall traffic. A witching solution involves selecting an interconnect topology with adequate capacity and a routing algorithm that selects the path each packet takes from its input to output port.

Click Modular Router

Click [3] a flexible modular software architecture for creating routers. They are build from fine grained components. The components are packet processing module called elements. To build a router configuration, the user chooses a collection of elements

and connects them into a graph. The graph's edges, which are called connections, represent possible path for packet handoff. Click configuration are modular and easy to extend. Click architecture were directly inspired by properties of routers. They are packet hand off and flow based router context mechanism.

A click element represents a unit of router processing. An element represents a conceptually simple computation such as decrementing an IP packet's time to live ,rather than a large, complex computation such as IP routing. Inside a running router, each element is a c++ object that may contain private state. The important properties of an element are element class ,ports, configuration string, method interface.

Click supports two kinds of connection, PUSH and PULL. On a push connection, packets start at the source element and are passed downstream to the destination. On a pull connection, the destination element initiates packet transfer. It asks the source element to return a packet, or a null pointer if no data packet is available. A click packet scheduler is simply an element with one pull output and multiple pull inputs. An elements responds to a pull request by choosing one of its input, making a pull request to that input and returning the packet receives.

The element is click's unit of CPU scheduling as well as its unit of packet processing. A task can initiate an arbitrary sequence of PUSH and PULL connections or requests. Task handles timer events . Each element can have any number of active timers, each timer calls an arbitrary method when it fails.

Safe Hardware Interface With XEN

Safe hardware interface,[5] is an isolation architecture which allows unmodified device drivers to be shared across isolated operating system instances, while protecting individual operating systems and the system a a whole, from driver failure. Xen ,an X86 –based virtual machine manager designed specifically targeting two utility based computing environments. They are organizational compute data centers and global scale compute utilities. The basic requirements of Xen are reliable execution of operating system instances, hard isolation and accounting and management for the underlying physical resources. Device drivers are used to identify the system bugs and system failures. The sharing of devices raises the stakes of driver dependability in a strong way. To avoid this problem a safe hardware interface has been developed. It allows the containment of practically all driver failures by limiting the driver's access to the specific hardware resources(memory, interrupts, I/O ports)necessary for its operation

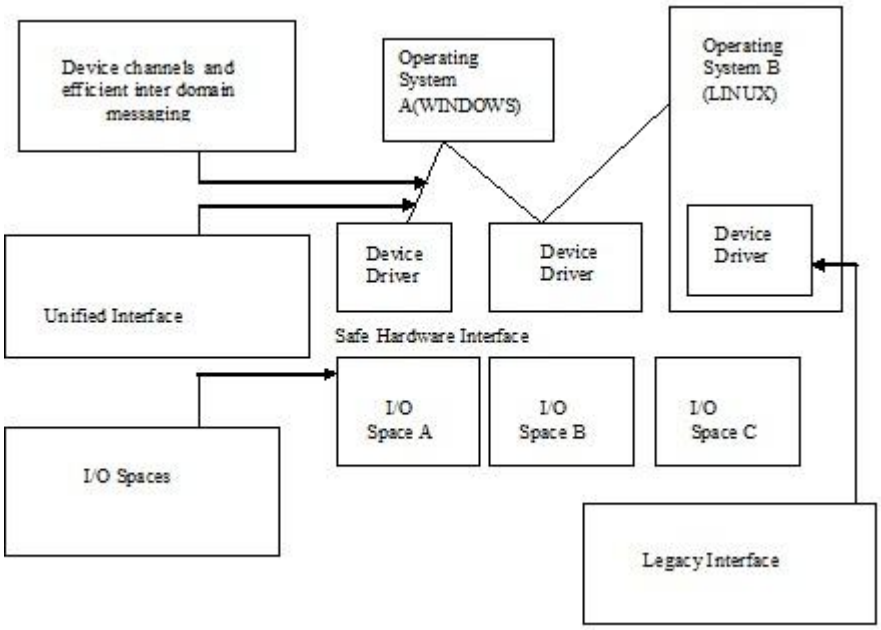


Figure: Design of Safe Hardware Model

Safe hardware model comprises three parts. First I/O spare which arrange that devices perform their work in isolation from the test of the system. It increases the reliability. Second, a set of per-class unified interfaces that are implemented by all devices of a particular type. It provides driver portability. Third, control and management interfaces, it simplifies system configuration and diagnosis and treatment of device problems.

Driver Domain Virtualization

The driver domain[6] is a virtual machine that runs a largely unmodified operating system. It is able to use all of the device drivers that available for that operating system.

The driver domain model provide a safe execution environment for physical device driver, enabling improved fault isolation over traditional models that locate device driver in the virtual machine monitor. The processing overheads occur in the driver domain virtualization limit overall I/O performance. There are two approach used for reduce the driver domain overheads. The multi-queue network interfaces in Xen is used to eliminate the software overheads of packet multiplexing and copying, second, a grant reuse mechanism is used to reduce memory protection overheads. These mechanism reduce the I/O virtualization in guest domain because these methods shift the bottleneck from the driven domain to the guest domain.

Figure illustrates the operation of driver domain virtualization model. When the network packets received by the network interface card (NIC) it raises an interrupt

when the physical driver access the NIC, the interrupt must passes through the virtual machine monitor. So that the virtual machine monitor raises the virtual interrupt, for the NIC in the driver domain upon receiving the virtual interrupt ,the physical driver sends the packet to the Ethernet bridge.

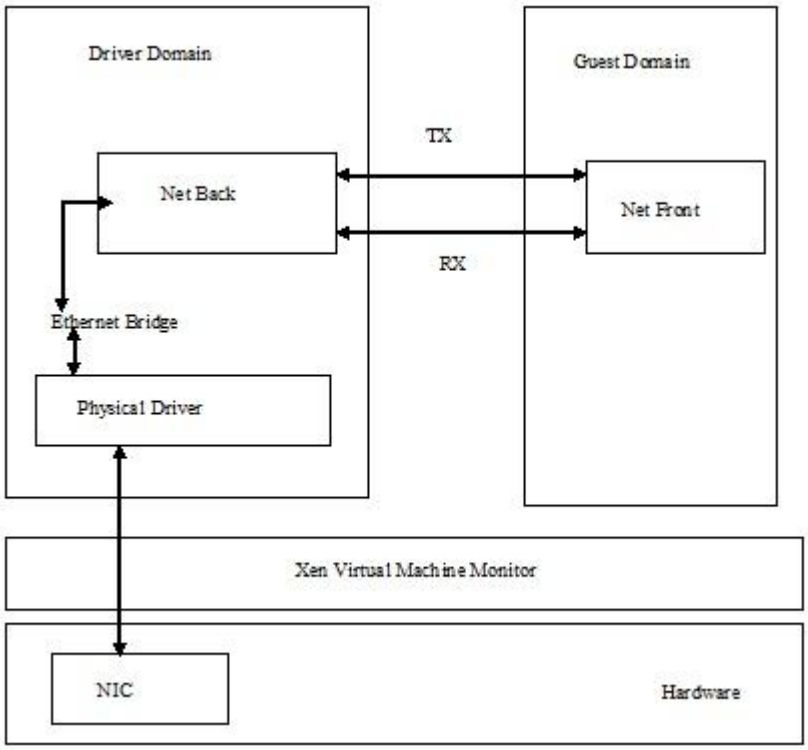


Figure: Xen's Domain Driver Architecture

The Ethernet bridge demultiplexes the packet based on its Ethernet address and deliver it to the appropriate netback interface. The netback driver then sends the packet to the net front, over an I/O channel. The I/O channels allow communication between the frontend and back end drivers using an event based mechanism.

The Ethernet bridge demultiplexes the packet based on its Ethernet address and deliver it to the appropriate netback interface. The netback driver then sends the packet to the net front, over an I/O channel. The I/O channels allow communication between the frontend and back end drivers using an event based mechanism.

Packet Aggregation Dimensioning Tool

The queuing theory will be used to model the system parameters like load, number of concurrent VM, etc.., The analytical model is used to get the average waiting time of the packet. The analytical model for packet aggregation consists of driver domain, the virtual machine and shared memory.

To consider the s/m composed of a driver domain and Network virtual machine denoted by VMi, i=1,2,…..N that exchange containers through shared memory.

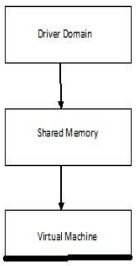


Figure: Packet Aggregation Dimensioning Tool

Assumptions

Assumption 1:
If the system will be an under heavy load conditions, there will always be a container available to be unloaded in guest domain.

Assumption 2:
The packets arrival to the driver domain follows a Poisson process with parameter $\mu 0$.

Assumption 3:
The packets are aggregated according to Poisson process with parameter $\mu i=1,2\ldots N$.

Assumption 4:
Containers receive an exponential service of parameter $\alpha 0$, when they are transferred from driver domain to virtual machine and of parameter αi, when they are transferred from virtual machine to driver domain.

Assumption 5:
The packets extraction time is null in the driver domain as well as in the virtual machine.

Analytical Model

Upon the arrival of a packet from the network device to the driver domain, it creates a container with size k0 and waits for the arrival of the k0-1 packets , before the whole container transferred through the shared memory to the destination virtual machine. The delay between the arrival of two containers generated by the driver domain and transferred to the shared memory corresponds to the sum of k0 random variables. Each random variables follows an Poisson process (i.e) random variable is exponentially distributed with parameter $\mu 0$. The sum of k0 exponential random

variables is characterized by an Erlang distribution with parameters μ0 and k0. In the similar way, the container size and packet delay are calculated in the virtual machine.

Packet Extraction Algorithm

The unloader is the component that unloads the receiver container at the destination to extract the packets. It removes the MAC header and retrieves the packet one by one based on their MAC destination address. Then the packets are processed by the upper layer. Packets are transferred from the virtual machine to driver domain follow the opposite path.

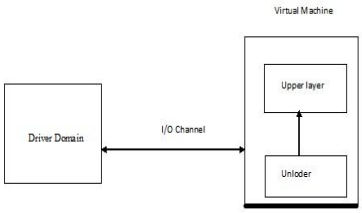


Figure: Packet Extraction Process

Results and Discussion

Experimental Setup

A system under test on which we use the virtual machine's and four machines used as traffic source and sinks.

Throughput Analysis

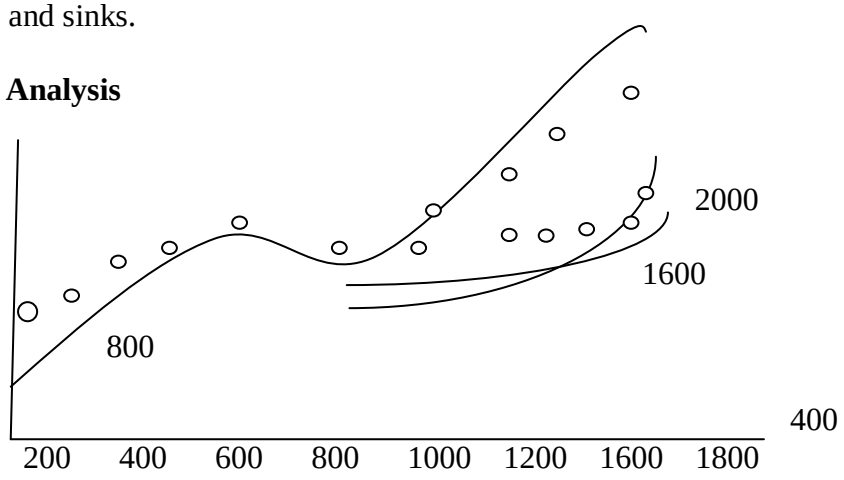
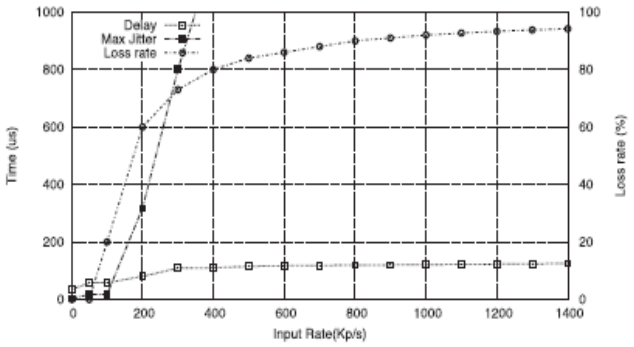


Figure shows the throughput for one virtual machine. The packet aggregation mechanism significantly improves the transmission and forwarding throughputs. The virtual machine achieves up to 1000kp/s in the case of packet forwarding. The throughput can be achieved up to 1100kp/s.

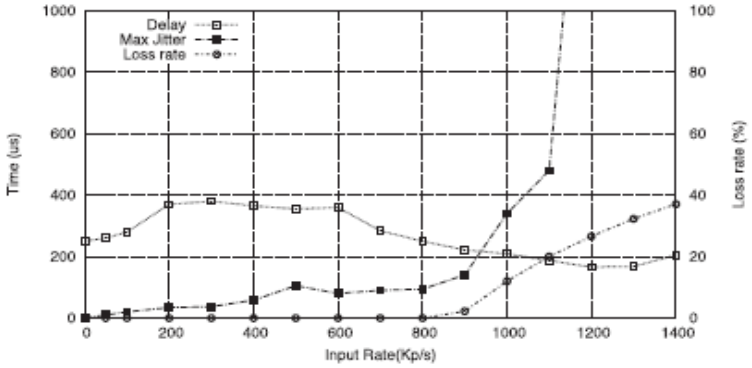
For the four virtual machine, The throughput achieved up to 2400kp/s. The forwarding throughput is almost increases up to 3600kp/s. For the aggregation mechanism, the throughput increases with the number of virtual machine's.

Delay and Jitter

Packet Delay, Jitter and Loss Rate With Native System



Packet Delay, Jitter and Loss Rate With Aggregation Based System



Future Work

In the future work, to arrange the packets in a queuing based on the priority size. So to improve the throughput. And also to apply the segmentation scheme for the virtual machine, When the bottleneck problem will reduced**.** And also to improve the networking performance.

Conclusion

In the next generation networking, the network virtualization place a important role in enhance the performance. Network Virtualization is being adopted in both telecommunication and the internet as the key attribute. Virtualization is becoming a key technology to enable deploying efficient and cost effective cloud computing platforms. However, current network I/O virtualization models still suffer from performance and scalability limitations. In this paper, the survey has been provided for improving the scalability problems to overcome this limitation, a packet aggregation mechanism based on their MAC destination address I developed. A packet aggregation mechanism that allows to transfer containers of packet at once. In this survey, the packet aggregation based I/O virtualization model, packet aggregation mechanism , and queuing theory for improving the performance of I/O virtualization are discussed.

References

[1] A. Menon, A.L Cox, and W. Zwaenepoel, "Optimizing Network Virtualization in Xen," Proc. USENIX Ann. Technical Conf. (USENIX' 06), 2006.

[2] A. Menon and W. Zwaenepoel, "Optimizing TCP Receive Performance," Proc. USENIX Ann. Technical Conf. (USENIX '08), 2008..

[3] E. Kohler, R. Morris, B. Chen, J. Jahnotti, and M.F. Kasshoek, "The Click Modular Router," ACM Trans. Computer Systems, vol. 18, no. 3, pp. 263-297, 2000.

[4] J.R. Santos, Y. Turner, G. Janakiraman, and I. Pratt, "Bridging the Gap between Software and Hardware Techniques for I/O Virtualization," Proc. USENIX Ann. Technical Conf. (USENIX '08), 2008

[5] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield, and M. Williams, "Safe Hardware Access with the Xen Virtual Machine Monitor," Proc. First Workshop Operating System and Architectural Support for the On Demand IT Infrastructure (OASIS '04), 2004

[6] K.K. Ram, J.R. Santos, Y. Turner, A.L. Cox, and S. Rixner, "Achieving 10 Gb/s Using Safe and Transparent Network Interface Virtualization," Proc. ACM SIGPLAN/SIGOPS Int'l Conf. Virtual Execution Environments (VEE '09), 2009.

[7] M. Dobrescu, N. Egi, K. Argyraki, B.G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "Route Bricks: Exploiting Parallelism to Scale Software Routers," Proc. ACM SIGOPS Symp. Operating Systems Principles (SOSP '09), 2009]

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03), Oct. 2003.