Project Report

A Middlebox specialized Hypervisor

Author:
Mihir J. Vegad

Guide:
Prof. Purushottam Kulkarni

A report submitted in partial fulfilment of the requirements

for the degree of Master of Technology in the

Computer Science and Engineering

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

Acknowledgement

I would like to thank my guide, **Prof. Purushottam Kulkarni** for giving me the opportunity to work in this field. I really appreciate the efforts which he put in throughout the project, to understand the work done by us and then to guide us to the next step. During this process, I learned a lot and overall it has created strong base for me in the field of NFV/SDN/Virtualization. I would also like to thank fellow SYNERG mates for extending their support whenever it was required.

Abstract

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the key factor in the rise of the Data center networking in recent years is, the core system infrastructure such as storage, network devices has become software-defined. Network Function Virtualization decouples the network applications from proprietary hardware and virtualize them. These network functions are usually referred as Middleboxes. To run middleboxes on virtual machines, we require a Virtual Machine Monitor or Hypervisor which allow multiple commodity virtual machines to share conventional hardware in a safe and resource managed manner. In the era of virtualization and Software Defined Networking, Data centres are being flooded with various middleboxes and we want to virtualize them efficiently without compromising the overall performance or resource utilization.

1.1 Middleboxes and Hypervisors

By definition a middlebox is a networking function that transforms, inspects, filters or manipulates network traffic for some purpose other than packet forwarding. [**?**] NFV has attracted many service providers to virtualize their network. Virtualized network functions would be located in those data centres along with other network nodes. According to a survey, number of middleboxes deployed in varying size data centres are on par with number of L3 routers and number of L2 switches. [**?**] Wan optimizers, proxies, application layer gateways, load balancers, intrusion detection system, intrusion prevention system are widely used middleboxes in data centres. Middleboxes play an important role in meeting service level agreements(SLAs) with client. Traffic is normally routed through chain of such middleboxes. All the middleboxes perform some entry level task followed by specific middlebox function followed by packet forwarding. They perform some redundant tasks which increases packet processing time through a middlebox chain and overall response time to user. To deal with this issue we have to leverage virtualization as in the transition from traditional network, hypervisor is still one component which operates in traditional manner.

The sole aim of virtualization is to run many virtual systems on a single physical system efficiently. Hypervisors are softwares/firmwares that creates or destroys virtual machines and manages physical resources among them. Two different types of hypervisors, Bare metal hypervisor and Hosted hypervisors are widely used in the market. Bare metal hypervisor runs directly on the hardware and manages virtual machines. Hosted hypervisor runs as an application on the operating system, usually known as host operating system. Oracle VM Server for x86, Xen server, KVM, Hyper-v are examples of bare metal hypervisors. VmWare workstation/player, virtual box are example of hosted hypervisors. We will focus on Bare metal hypervisors (Type-1) as they are widely used in data centres. A hypervisor mainly provides virtual hard disk, virtual network interface card, and life cycle management facility to virtual machines. We will discuss what more a hypervisor can offer to middleboxes, apart from this traditional stuff. Mostly used hypervisors in data centre virtualization are vSphere

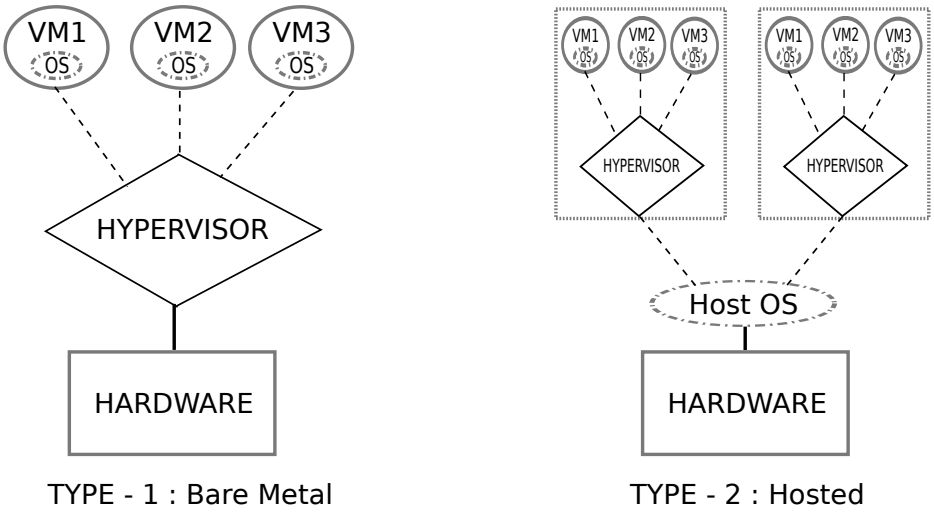

Figure 1.1: Different types of Hypervisor

from VMware, Xenserver from Citrix, hyper-v from Microsoft and KVM from Redhat. Different hypervisors are used as per the requirements of the data centre administrator. We will focus on Kernel Virtual Machine(KVM) as a hypervisor in our experiments. KVM virtualizes the processor and the memory of the system. QEMU is a 'Quick Emulator' which is used to arbitrate hardware resources such as disk and network interface card. QEMU is a type-2 hypervisor by itself, it executes the virtual CPU instruction using host operating system on the physical cpu. But when QEMU combines with KVM, the combination becomes type-1 hypervisor. In that combination, QEMU is used as a virtualizer, and it achieves near native performance by executing guests code directly on hardware. We have used virtio enabled KVM setup for experiments. Virtio is a standard for network device driver while transmitting or receiving a packet from/to that network device. In this case, network device driver just know that it is running in the virtualized environment.

When a packet is received on the physical network interface card, it is hypervisor's duty to forward it to the specific middlebox hosted on that machine or vice versa. In our KVM setup, how this packet travels from the physical NIC to middlebox or viceversa is very important to understand before we move towards design of the solution. When a packet is received on pNIC, It is forwarded to the hypervisor bridge. Bridge forwards it to the tap interface for destination middlebox. Tap interface forwards it to vNIC. And then at last it is received by the middlebox. This is very brief description of the whole process. We will discuss this in more detail in chapter 5.

1.2 Problem Statement

1.3 Organization of the work

Chapter 2

NFV in Data centers: Middleboxes

2.1 Why to focus on MB?

Middleboxes are crucial part of data centre networks as only packet forwarding is not good enough to meet customer requirements, other functionalities like Quality of Service/Quality of Experience, load balancing, security are needed to make customer experience complete. This fact is also supported by a survey done over 57 enterprise data centres, whose size varied from less than 1K hosts to more than 100K hosts. [**?**] The survey shows that number of middleboxes deployed in data centres are on par with number of routers and number of switches in the data centre.

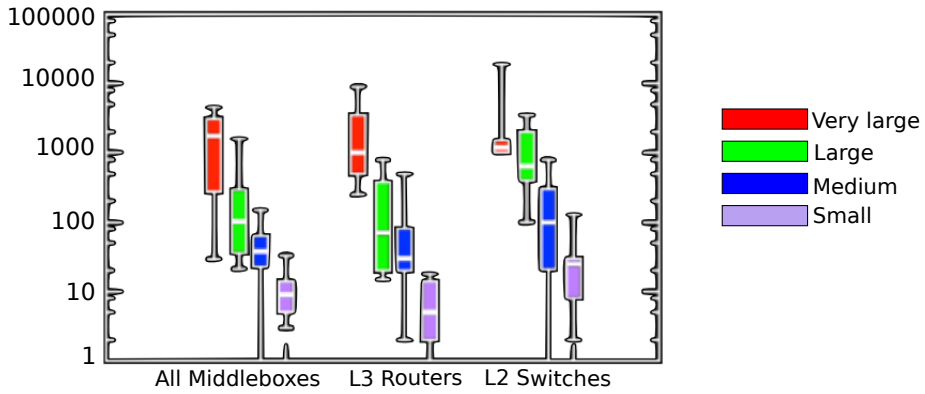

Figure 2.1: MB presence in the data centres [**?**]

2.2 Middlebox examples

2.2.1 General middleboxes

Some of the widely used middleboxes among the data centres are described here.

- **Firewalls:** A firewall is a network security system that monitors and controls the incoming and outgoing traffic based on predetermined security rules set by administrator. [**?**]
- **IDS/IPS:** IDS is a software application that monitors the network traffic for malicious activities or policy violation and produces a report to administrator. IPS is proactive approach of monitoring network traffic and identifying malicious activities and prevent them from occuring. [**?**]
- **Load Balancer:** Load balancer distributes the incoming requests among a set of resources available. Efficient load balancer leads to optimized resource utilization, maximized throughput and minimized response time.

- **Wan optimizer:** It improves bandwidth consumption and latency between dedicated end points. It coordinates to cache and compress traffic that traverses the internet. [**?**]
- **Network Address Translators:** It serves network traffic to multiple private machines which are exposed to internet through a public host. It modifies the 4-tuple address fields of the packets to ensure that it reaches to the correct host.
- **Traffic shaper:** Traffic shaper is also known as a rate limiter. It limits the amount of bandwidth for specific traffic.
- **Proxies:** A proxy sits in between client and server, to simplify the client's requests to servers and to provide anonymity to clients. There are various types of proxies based on what specific task they do in addition to the basic task. The simplest proxies which passes the requests and responses unmodified are also known as **Gateways**.
- **VPN :**A virtual private network establishes a private network across public networks. It allows user to send and receive data across public networks maintaining the policy and security enforced by the private network.
- **Wire:** A simple middlebox which sends packets from its input interface to output interface. It is generally used to give a performance baseline.

2.2.2 Application specific middleboxes

Chapter 3

Related Work

Over the past few years, much research has been done in the area of customization of middlebox functionalities, customization of hypervisors which hosts the middleboxes and customization of operating systems for middleboxes. It is very important to understand the work done so far, before moving ahead with our proposed solution.

Based on our goal, which is to improve the performance of the middleboxes in virtualized data centres, we can classify the techniques studied so far into three categories.

- Redesign the Operating System
- Regesign the Hypervisor
- Redesign the hardware

We will briefly discuss some techniques which fall inside these categories. And we will also state how it relates to or contrast with our proposed solution.

3.1 Redesign the Operating System

ClickOS[**?**] proposes replacement of traditional guest operating system with some optimized, light weight operating system. Linux as a guest operating system in VMs is actually over provisioning for middlebox applications. Middlebox uses very few operating system services in addition to network connectivity. Traditional Linux operating system takes around 128MB space on guest VM and also takes around 5 second time to boot, that is very slow in context of middlebox setup time. So, they came up with minimalistic operating system for virtual machines. ClickOS virtual machines which runs MiniOS are very small in size that is only around 5 MB and also very fast to setup, only 30 msec boot time. MiniOS has a single address space, no kernel/user space separation and a cooperative scheduler. Basically ClickOS are single core machines. They have used Xen as a hypervisor for experiments. It also improves scalability of the middleboxes. With traditional full fledged OS running on the VMs, number of tenants supported on a physical machine are very small. But with MiniOs running on ClickOS machine the number increases drastically. This technique improves the scalability and the setup time for middleboxes but it does not address redundancy among middlebox functionalities. Though it can create new ClickOS VM quickly for scaling, ClickOS VMs do not have symmetric multiprocessing support.

OSv[**?**] also offeres optimized operating system for middleboxes in cloud environment. VMs in cloud usually runs linux as an OS. When we say middlebox is running on a VM, it often means that only single application (middlebox application) runs on that VM. In this case, managing multiple VMs on hypervisor only means that managing multiple applications on the hypervisor. Hypervisor provides features like isolation among VMs, hardware abstraction, memory management. In this case hypervisor almost act as an OS for middlebox applications. So, when we use traditional operating system to run middlebox applications,

the above mentioned features become redundant. It affects overall performance of the host machine. OSv is a guest OS specially designed to run a single application on a VM in the cloud environment. It has very small OS image, dedicates more memory to the application and lesser boot time. OSv supports various hypervisors and processors with minimal change in architecture specific code. For 64-bit x86 processors, it supports KVM, Xen, VMware and Virtual Box hypervisors. It also improves scalability of middleboxes just like ClickOS. But it supports Symmetric multi-processing which is an advantage over ClickOS. It also gives throughput and latency improvement for middleboxes.

Unikernels[**?**] also provides light weight machines to deploy middlebox applications in cloud environment. Unikernels are single purpose appliances, they strips away functionalities of general purpose system at compile time. It is inherently suitable for middleboxes. It takes into consideration the idea of library OS, in which an application links against separate OS service libraries and unused services from the library are eliminated from the final image by the compiler. For an example, virtual machine image of a DNS server can be as small as 200KB. Mirage OS[*] is an example of such a library OS. It is written in OCaml, a functional programming language and it runs on the Xen hypervisor[*]. It also improves scalability and setup latency for the middleboxes.

3.2 Redesign the Hypervisors

Container[**?**] modifies the hypervisor to eliminate redundant features among the hypervisor and the guest OS. Actually it drops the idea of traditional hypervisor in virtualization. It modifies the host operating system to support isolated execution environments for applications while running on the same kernel. It improves resource utilization among guests and lowers per guest overhead. It also improves the overall performance of the system. I/O related workload, server type workload performs better on container based system compared to hypervisor based system. It also scales well compared to hypervisor setup. But still most of the data centres prefer to use hypervior based system. Hypervisor based system can support multiple kernels but by design container based system can not support the same. Container also does not have support for VM migration. Hypervisors are the industry standard for virtualization.

CoVisor[**?**] is a hypervisor, which uses a completely different approach to host middlebox applications in a software defined network. It uses basic concept of network hypervisor that is to manage virtual networks on a physical network. Along with usual middlebox hosting challenges, it deals with some SDN specific challenges as well. For example, middlebox applications used by different vendors may be built by using different SDN APIs. Covisor makes it possible to run any middlebox irrespective of what SDN API is used. Covisor provides facility to assemble multiple middlebox applications as per the administrator configuration. Each middlebox can be used independently, in parallel or sequential with other middlebox, or conditionally. Administrator can provide abstract virtual topology for each middlebox. It restricts each middlebox's view of the physical network. Configuration file provided by administrator includes policies to assemble middleboxes, mapping for each middlebox's virtual network components to actual physical components and access control limitation for each middlebox. Covisor was tested with respect to its composition efiiciency and devirtualization efficiency. It gives satisfactory results on metrics like policy compilation time, Rule updation time and total devirtualization time. Covisor is still in development phase and as most of the data centres use traditional hypervisors, switching to Covisor needs lots of modification to existing data centre architecture.

We already discussed ClickOS[**?**] phase-I in the first section. ClickOS phase-II falls under this section. ClickOS authors ran ClickOS machines on Xen hypervisor for experiments. They modified the Xen hypervisor to achieve throughput and better resource utilization. In traditional Xen hypervisor networking, when a packet is received on physical NIC, it traverses through network driver(dom0), software switch(dom0), virtual interface(dom0),

netback driver(dom0) and netfront driver(guest machine) before getting processed by a middlebox. ClickOS technique modifies memory grant mechanism, netback driver, software switch and netfront driver to increase the middlebox throughput. Modified version of Xen is still not capable of handling a chain of middleboxes. Longer the chain is, lower the throughput. Even after all the modifications, hypercalls done by Xen for each packet transmission remains the bottleneck in this case.

3.3 Redesign the hardware

Chapter 4

Design of MsH

Chapter 5

Experiments

5.1 Experimentation setup

5.2 Results

5.2.1 Interrupts generated by packets

5.2.2 Time required to process a packet

5.2.3 Throughput of a client application

5.3 Challenges

Chapter 6

Conclusion and Future work