ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Improve Virtualization Technique Using Packet Aggregation Mechanism

A.Senthilkumar*[1], Mr.G. Sivakumar[2]
*[1]M.E – II Year, Gnanamani College Of Engineering, Namakkal, India
[2]Assistant Professor, Gnanamani College Of Engineering, Namakkal, India
senthilafree@gmail.com

Abstract

Virtualization is a key technology for cloud based data centers to implement the vision of infrastructure as a service (IaaS) and to promote effective server consolidation and application consolidation. A new method is implemented in virtual machine monitor and representative workloads set method in cloud-based data centers does not provide sufficient performance isolation to guarantee the effectiveness of resource sharing, particularly during run time for a program on multiple virtual machines of the same physical machine are competing for computing and communication resources. In this project, we present our performance measurement study of network I/O applications in virtualized cloud. The proposed packet aggregation based mechanism is to transfer packets from the driver domain to the virtual machines. By observational the performance is calculated and documented that our proposal allows the virtual machines throughput to scale up at line rates. The proposed model allowed us to dynamically tune the aggregation mechanism in order to achieve the best tradeoff between the packets delay and throughput. This I/O virtualization model henceforth satisfies the infrastructure providers to offer Cloud computing services.

Keywords: Cloud computing, Resource Sharing,Virtual Machine, cost effective..

Introduction

In recent years, there has been a rapid growth in the adoption of virtual machine technology in data centers and in cluster environments. Commercial virtualization software vendors such as VMware, Xen Source (Citrix) and Microsoft are increasingly making their presence felt in the server market. On the hardware front, Intel and AMD have incorporated support for virtualization in their CPU instruction set [AMD, INT], and the PCI-SIG vendor consortium has proposed standards for adding support for virtualization in the PCI standard for I/O virtualization [ATS, SRI, MRI]. This trend towards server virtualization is driven by two main factors: the savings in hard-ware cost achieved through the use of virtualization, and the greater flexibility of management of virtualized cluster resources.

In a typical server environment, each server application is run on a separate server machine, both in order to isolate it from other server applications, and to customize its software stack (including the operating system). Unfortunately, this results in unnecessary server sprawl, where each server machine runs at a low average utilization rate. Server virtualization prevents this sprawl by consolidating multiple underutilized servers onto a smaller number of physical machines, thereby reducing the hardware costs required to run multiple servers. Each server application is run in its own operating system running on a different virtual machine (VM), and multiple VMs are multiplexed on a single physical machine by the virtual machine monitor (VMM).

Virtualization Approaches

In a virtualized environment, the VMM performs the task of virtualizing and multiplexing the physical resources of the system among the virtual machines. These resources include the CPU, memory and I/O devices. Thus, it provides each VM with its own virtual CPU(s), a subset of the host memory, and a set of virtual I/O devices. The guest OS running in the VM uses these virtual resources in a manner similar to the way it uses physical resources in a native environment.

There are two main distinct approaches to virtualization, full virtualization and per-virtualization. In full virtualization, the hardware interface provided to guest operating systems is exactly identical to the underlying native hardware

ISSN: 2277-9655
Impact Factor: 1.852

interface. The advantage of this approach is that it provides full binary compatibility with existing operating systems, and thus does not require any modification in the guest OS to run in the VM. The downside of this approach is performance degradation incurred for emulating the exact semantics of the native hardware. The full virtualization approach for x86 CPUs was pioneered by VMware. With the introduction of hardware support for virtualization in x86 CPUs, other virtualization products, such as Citrix's XenServer, and Microsoft Hyper-V also support full virtualization for unmodified OSes.

Scope of the Project

An important concern with server virtualization is that the performance of server applications running inside the VMs can degrade significantly relative to their performance in a native environment. This is because of the overheads incurred by the VMM in the virtualization of the physical resources of the system. This overhead can be particularly high for the virtualization of I/O devices [MST+05, SVL01], especially for I/O devices which require frequent servicing by the device driver, for example, interrupt-intensive devices such as network cards.

We address the problem of efficiently virtualizing the network interface in a virtual machine environment. A number of important server applications, such as web servers, and streaming media servers are network-intensive applications, and their overall performance depends crucially on the networking performance of the system. In addition, efficient networking in the VMM is useful for a number of VMM operations, such as virtual machine migration, network file system traffic, etc.

The network performance in a Type-II VMM, using the Xen VMM as an example of this architecture.Xen is an open-source VMM developed at the University of Cambridge [BDF+03], and it uses an I/O architecture similar to the Type-II hosted VMM.

We focus on the network performance in pervirtualized guest operating systems running on Xen. We identify the fundamental performance bottlenecks in the network virtualization stack of the Xen VMM, and propose a number of solutions to address these bottlenecks.

Workflow of Xen Network I/O

Each guest domain running on Xen is provided with a number of virtual network interfaces which it uses for all its network operations. Each virtual interface in a guest domain (also called frontend interface) has its own MAC address and is connected to a corresponding 'backend' interface in the driver domain through an 'I/O channel'. The I/O channel provides mechanisms for exchanging network packets between the frontend and backend interfaces. The frontend and backend interface can signal each other using virtual interrupts when there are network packets to be transferred between the two.

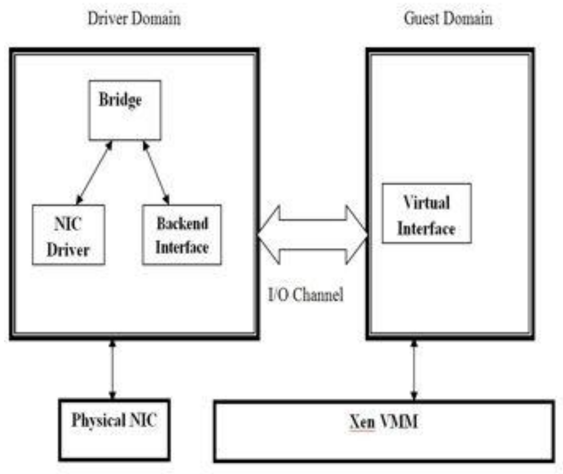

Figure.1: Xen Network I/O Architecture

The above figure shows a high level picture of the network I/O virtualization architecture of Xen.All the backend interfaces in the driver domain (corresponding to the guest domain's virtual interfaces) are connected to the physical NIC and to each other through a network bridge. The combination of the bridge and the I/O channel allows the physical interface and the guest domain's virtual interfaces to transfer packets to each other based on the destination MAC address of the packet. Thus, for instance, on the transmit path, packets are transmitted by the guest domain on its virtual interface, which are then transferred over the I/O channel to the backend interface, which then transfers them to the physical interface over the network bridge, and finally the NIC sends them out on the network. The receive path is similar, except in the reverse direction.

The network virtualization overheads in the Xen I/O architecture are similar to the I/O overheads incurred in a Type-II VMM. Like in a Type-II VMM, network I/O operations in the guest domain require an address space switch to the driver domain in order to invoke the NIC driver. The frequent switching for I/O operations and the cost of operations in the I/O virtualization stack (I/O channel transfers, bridging) significantly degrade network performance in Xen guest domains.

XEN Virtualization Overheads

The benchmark measures the maximum transmit/receive throughput achievable (in Mb/s) over a small number of TCP connections, where each TCP connection is used to send/receive traffic over a different network card. The first set of histograms in the figure shows the transmit performance and the second set shows the receive performance.

The network performance when running the workloads in the driver domain is very close to the performance in a native Linux system. Thus, transmit performance (3760 Mb/s) is the same as the Linux transmit performance (except that it incurs higher CPU overhead), and receive performance (1738 Mb/s) is within 70% of the native receive performance (2508 Mb/s). In contrast, when the workloads run in the guest domain, the performance achieved is significantly lower. The transmit performance in the guest domain (750 Mb/s) is only 20% of the native transmit performance, and the receive performance (820 Mb/s) is roughly 33% of native.

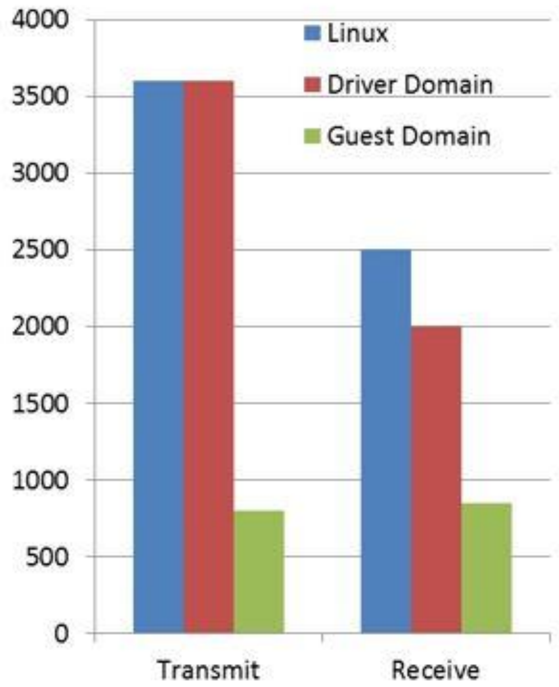

Figure 2: Network performance in guest and driver domains running Xen

Fig 2.shows the transmit and receive network performance for a netperf [NET] like bench-mark running in three configurations: a Xen guest domain, the Xen driver domain, and a native Linux system.The reason for the large difference between guest domain and driver domain performance is that the driver domain invokes the NIC driver directly for network I/O, whereas the guest domain needs to switch to the driver domain, and transfer its packet data to the driver domain address space, for all network I/O operations. We note that in a Type-I VMM, no switching is incurred for invoking the device driver in the hypervisor from the guest domain, and the packet remains in the same address space, thus the performance of a Type-I VMM is comparable to the performance of the Xen driver domain.

Related Work

The complexity and cost of network virtualization in the VMM depends on a number of factors, with an important factor being the I/O architecture used by the VMM. The I/O architecture of the VMM defines the way it uses device drivers to control the I/O devices in the system. There are two main approaches to do this, and VMMs are classified as either Type-I or Type-II depending on which approach they take.

Type-I VMM:

In a Type-I VMM, or non-hosted VMM (also called hypervisor-based VMM), the virtual machine monitor completely controls all hardware resources in the system, including the I/O devices, and no guest OS is allowed to access the hardware directly. Thus, the VMM provides its own device drivers for controlling the I/O devices in the system.

Type-II VMM:

In a Type-II VMM, also called a hosted VMM, the VMM does not control all hardware directly. Instead, it relies on a special privileged host operating system (called driver domain in Xen terminology) for controlling and managing the hardware, including all I/O devices. Thus, device drivers for managing the I/O devices are provided by the host OS, and the VMM must switch from the guest to the host OS for every device I/O operation. Thus, this approach entails greater performance overhead for I/O operations.

The VMM architecture of Xen is a combination of Type-I and Type-II VMM. Although the Xen hypervisor controls the physical hardware such as CPU and memory directly, it delegates control of I/O devices to a special, privileged operating system running in a VM, called the driver domain. Thus, it is similar to a hosted VMM architecture in that it relies on a host OS for managing the I/O devices. However, Xen runs the host OS in an isolated VM of its own, thus, the VMM and other VMs are isolated from bugs or crashes causes by the device driver.

ISSN: 2277-9655
Impact Factor: 1.852

Conclusion

To maximize the benefit and effectiveness of server consolidation and application consolidation in virtualized cloud environments, we argue that it is important to conduct in-depth performance measurements for applications running on multiple VMs hosted on a single physical machine. Such measurements can provide quantitative and qualitative analysis of performance bottlenecks that are specific to virtualized environments, offering deeper understanding of the key factors for effective resource sharing among applications running in virtualized cloud environments. We have presented our performance measurement study of network I/O applications in virtualized cloud environments.

References

[1] P. Apparao, R. Iyer, X. Zhang, D. Newell, and T. Adelmeyer, "Characterization &Analysis of a Server Consolidation Benchmark,"Proc. ACM/USENIX Int'l Conf. Virtual Execution Environments,pp. 21-29, 2008.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A.Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M.Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing,"Technical Report UCB/EECS-2009-28,2010.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R.Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art ofVirtualization," Proc. ACM Symp. Operating Systems Principles(SOSP), pp. 164-177, 2003.

[4] Z. Chen, D. Kaeli, and K. Murphy, "Performance Evaluation of Virtual Appliances," Proc. First Int'l Workshop VirtualizationPerformance: Analysis, Characterization, and Tools, Apr. 2008.

[5] L. Cherkasova and R. Gardner, "Measuring CPU Overhead for I/ O Processing in the Xen Virtual Machine Monitor," Proc. USENIXAnn. Technical Conf. (ATC), p. 24, 2005.

[6] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the Three CPU Schedulers in Xen," ACM Sigmetrics PerformanceEvaluation Rev., vol. 35, no. 2, pp. 42-51, Sept. 2007.

[7] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J.N. Matthews, "Xen and the Art of Repeated Research," Proc.USENIX Ann. Technical Conf. (ATC), pp. 135-144, 2004.

[8] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfiel, "Live Migration of Virtual Machines," Proc.USENIX Symp.Network Systems Design and Implementation (NSDI),pp. 273-286, 2005.

[9] T. Deshane, Z. Shepherd, J.N. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao, "Quantitative Comparison of Xen and KVM," Xen Summit Boston 2008.

[10]Y. Dong, X. Yang, X. Li, J. Li, K. Tian, and H. Guan, "High Performance Network Virtualization with SR-IOV," Proc. IEEE16th Int'l Symp. High Performance Computer Architecture (HPCA),pp. 1-10, 2010.

[11]S. Govindan, A.R. Nath, A. Das, B. Urgaonkar, and A.Sivasubramaniam, "Xenand Co.: Communication-Aware CPUScheduling for Consolidated Xen-Based Hosting Platforms," Proc.ACM/USENIX Int'l Conf. Virtual Execution Environments, pp. 126-136, 2007.