Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 3 (2011) 401–407

Procedia Computer Science

www.elsevier.com/locate/procedia

WCIT-2010

A transparent virtual machine monitor level packet compression network service

Ali Hamidi, Hadi Salimi and Mohsen Sharifi

Distributed Systems Labaratory, School of Computer Engineering, Iran University of Science and Technology,

Abstract

Packet compression is a well-known technique for improving the performance of low speed networks such as WANs. This technique is also effective in networks with a high cost per transmitted byte, namely wireless networks. Most implementations of this technique as a network service, like IPComp, are not transparent and require modifications either to applications or to operating systems. Some other implementations of this technique as a network service, e.g. WANProxy, run as user space processes that impose extra overhead due to unnecessary packet hooking mechanisms, involving frequent switches between user and kernel spaces. This paper presents a transparent packet compression network service using virtualization technology. This service transparently compresses network packet payloads and reduces communication overhead of applications running inside virtual machines. This service requires no modifications to applications, operating systems or virtual machine monitors. In addition, it has no extra overhead for switching between user and kernel spaces, because it is implemented as a configurable kernel module that can be activated or deactivated dynamically. A proof-of-concept transparent packet compression service has been implemented on Xen hypervisor. Evaluation results show the feasibility of development and dynamic configuration of such a transparent packet compression network service. Results also show approximately 25% improvement in network performance of applications over slow or congested links. In addition, the results show that this service can tremendously reduce transmitted bytes over high cost networks like wireless networks, in a transparent manner.

© 2010 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of the Guest Editor.

Keywords: Packet Compression; Virtualization Technology; Transparent Network Service

1. Introduction

Compressing network packets is a well-known technique to improve network performance in a low speed network like WAN [1]. In addition, in some networks such as wireless ones, packet compression can improve efficiency especially when the link has high cost per transmitted byte [2]. In a technical view, compressing network packets can help reaching better performance in networks that the time needed for compression or decompression of a packet and transferring it through the network in lower than the time needed to transfer uncompressed packet.

In networking community, there are two classes of packet compression technique: (1) network headers compression and (2) bulk compression. As the names imply, the first class just packs the network packet headers, while the later treats the whole packet (headers and data) as a block of raw bytes and compresses the whole. As far as, TCP/IP is the most common protocol stack nowadays, in this paper we concentrate on approaches that have been proposed on this protocol stack.

There are many approaches proposed to compress network headers [1][2]. In such approaches if the compression is applied to IP header, there must be a compression and a decompression on every physical link in the network, because IP header information is used for routing packets from sender to receiver. We know that every compression

1877-0509 © 2010 Published by Elsevier Ltd.
doi:10.1016/j.procs.2010.12.067

and decompression action has a cost and imposes an overhead to the network performance. Hence, compressing IP header on every physical link has a lot of overhead on the network performance and it must be done on hardware. But, (de)compressing of IP payload that contains headers and data of upper layer network protocols can be done in two nodes that are placed at two sides of a slow or congested link.

On the other hand, in recent years, Virtualization Technology (VT) experiences resurgence due to its support for consolidation, isolation and migration [3]. In addition, the development of high performance Virtual Machine Monitors (VMMs), like Xen [4] and VMware ESX [5] has made VT more applicable to high performance computing (HPC) [6], Grid computing [7] and also pervasive computing [8]. More recently, an innovative concept called Cloud computing [9] has been introduced to the computing community. Generally, Clouds are an evolution of Grids using VT for dynamic provisioning of resources and scalability [10]. VT also allows plugging new transparent services into computing environments without any modifications to applications or operating systems. Many researches use this support in order to get information from applications or operating systems running in virtual execution environments [11][12][13]. VT can also be used to change the way applications use system resources like the network interface [12][14].

In our previous work [15], we proposed a local acknowledgement network service implemented in a virtualized environment. To achieve high performance, we have implemented and deployed this network service as a kernel module at the kernel space. We have used this approach to eliminate the need of switching from kernel-space networking modules into user-space network service and vice versa.

In this paper we have used the mentioned approach to implement a network service that transparently (de)compresses network packets at two ends of a slow or congested link. Using this technique a system administrator can dynamically improve communication performance of nodes over slow or congested link. This means that for enabling or disabling packet compression network service, there is no need to change or reconfigure operating systems and applications running inside them. This technique makes packet compression more common and easy to use. The evaluation results of the proof-of-concept packet compression network service show approximately 25% improvement in network performance of the TCP applications on slow or congested link.

The rest of paper is organized as follows. Section 2 presents some notable related works. Section 3 presents our approach to provide transparent packet compression network service. In Section 4 we report evaluation results of the implemented packet compression network service and finally Section 5 concludes the paper.

2. Related Work

IPComp [16] is the most famous packet compression technique that has been proposed in RFC 3173. It is actually a new protocol for compressing IP packets payload. This protocol has been designed to increase overall communication performance between nodes over slow or congested links. IPComp has been practically used inside IPSec [17] protocol family in case of compressing packet size before applying encryption to the IP packets, because encryption causes the data to be random in nature and compressing that data in lower protocol layers is ineffective. The most important issue in using IPSec protocol stack is the need of changing default protocol stack of the operating system. In other words, when an administrator decides to use IPSec, she should compile operating system kernel and configure it to use IPSec protocol stack. As we said before, one of the goals of this paper is provisioning of packet compression network service transparently. This means that administrators can dynamically enable or disable this service, without any change in application or operating system.

In [18] there are some techniques that have been proposed to do bulk compression on the packets and overcoming its limitation like the need of large memory for memory and synchronizing dictionary at the compressor and decompressor. Bulk compression has no benefit in compressing network headers that are inside the packet, because the information inside the packets varies from a packet to packet. If a compression algorithm understands the syntax and semantic of headers inside a packet, it can be successful in exploring redundancies and it can gain good ratio in reducing the headers size [1]. This sub-domain of packet compression is called Header Compression and there are some researches like Van Jacobson Header Compression [19], IP Header Compression [2] and Robust Header Compression [20] that have been done in this sub-domain. In addition there are also some researches that have been proposed to perform header compression on the low speed serial links and their protocols like PPP [19][21].

Some other works [22][23] concentrate on limited bandwidth of wireless networks and uses compression to improve communication performance of the application over these networks. For example in [22] a new header compression scheme has been proposed that efficiently compress UDP/IP and TCP/IP headers.

WANProxy [24] is a user space application that acts as a network accelerator for applications communicating through WAN by compressing TCP packets. Beside the features and performance improvement that WANProxy could gain, we believe that (1) Applications must be modified to use WANProxy as their proxy to communicate through WAN and we know that this is not possible for all applications like legacy and closed source applications, (2) Because of user space nature of WANProxy, for every transferred packet, it have to go through network stack two times more. So we believe that using our approach, we can gain better improvement in network performance because this approach does not suffer from passing packet between user and space modes. In addition, this approach can achieve a high level of transparency in which neither the application nor the operating system has to be modified.

3. Transparent Packet Compression Service

In virtualized environments, VMM has the most privileged access to resources. This means that all resource accesses made by guest operating systems running inside VMs must go through VMM. This level of isolation creates the opportunity to provide several services for virtual machines transparently. We have chosen the Xen hypervisor as our testbed VMM. Fig 1 shows the typical networking path of an application running inside a virtual machine (DomU) in a Xen-based environment. In this environment, there is a special domain called driver domain or Dom0 that has direct access to hardware and the Xen hypervisor implements a ring-based shared memory mechanism for communication of front-end and back-end drivers.

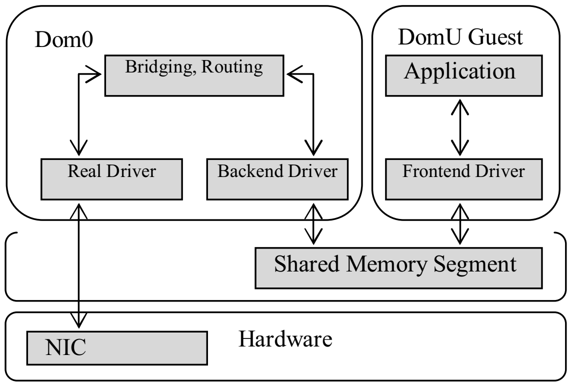

Fig. 1. Typical networking path of an application running inside DomU in a Xen-based environment [15].

To make the performance overhead of switching between user and kernel spaces lower, we use this fact that all network traffic of VMs must go through Dom0, so we can place our packet compression network service inside Dom0 kernel as a pluggable kernel module. We have implemented packet compression network service using netfilter [25] kernel module, because it is a flexible callback based packet filtering framework in the Linux kernel. We used a set of netfilter hooks in our implementation to capture the packets. Using these hooks, our packet compression network service was able to capture all traffics entering/exiting VMs. In addition, to provide dynamic configuration for the packet compression network service we used ProcFS [26] that is a RAM-based file system inside Linux kernel. Using this file system we built a configuration management system for packet compression network service. System administrators can simply manipulate configuration attributes of the service by opening and reading a virtual file, as well as changing the configuration attributes by writing new values to this file.

Fig 2 shows the architecture of our proof-of-concept packet compression network service that is installed on two physical machines. According to this figure, each physical machine has a Xen-based virtualized environment. As stated before, packet compression network service is beneficial when applied to two or more machines that are communicating through a low speed or congested link. For simplification, we decided to only compress TCP packets. As depicted in Fig 2, there are two TCP applications running inside different virtual machines running on top of two physical machines. Each TCP application that is run inside a virtual machine communicates to the other TCP application using the WAN. In this figure, the arrows show the path of packet communication. Solid arrows show the path of unmodified packets and dashed arrows show the path of compressed packets. This shows that packet compression network service compresses outgoing packets that are going through WAN and decompresses the packets that are coming from WAN.

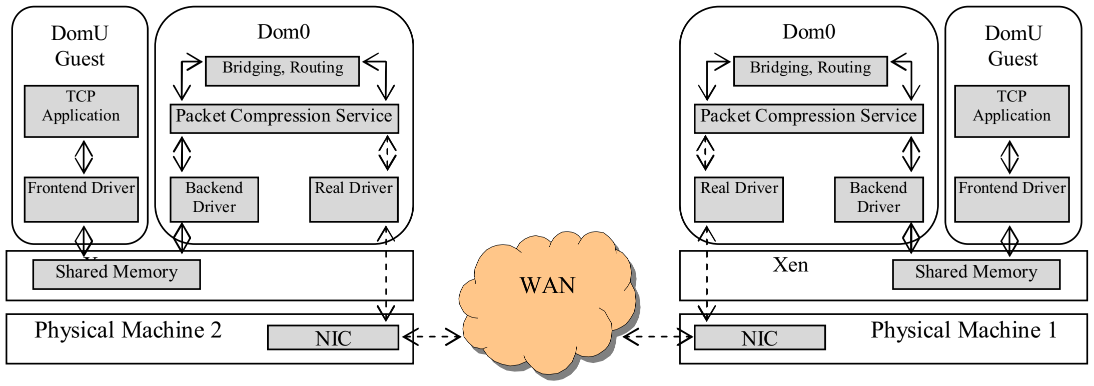

Fig. 2. Packet compression network service installed inside Dom0 of two physical machines.

4. Evaluation

To measure the performance of the implemented packet compression network service, we have built a testbed like the one shown in Fig 2. Each physical machine was equipped with an Intel Core™ 2 Quad Q6600 2.4 GHz processor, 4GB of main memory, and a Realtek RTL-8029(AS) 100Mb/s NIC. The Xen hypervisor (version 3.2) was installed on one of these nodes. The Dom0 operating system of this environment was a CentOS 5.3 operating system with the kernel version 2.6.18-128-Xen.

We have used Iperf [27] benchmark to evaluate network performance of TCP connection between two applications. Iperf offers several parameters to change the test conditions. We have used Iperf parameters to evaluate the network performance in three different tests. We ran the client version of Iperf in DomU of the first machine and the server version of Iperf in DomU of the second machine. In each test, we calculated the network performance in two modes: (1) Existence of packet compression network service and (2) without packet compression network service.

We did three different tests by configuring parameters of Iperf. In the first test, we ran Iperf with its default parameters. Using these parameters it calculates the network bandwidth based on number of bytes that could be transferred in ten seconds. In the second test, we executed Iperf with a 30-second execution time and in third test we configure Iperf to calculate the network bandwidth for transferring about 200MB of data. Fig 3 shows the results of these tests.

Fig. 3. Evaluation results of packet compression network service.

The results show that in all of the three tests we can achieve approximately 25% improvement in network performance. In addition, we have measured the saved sent bytes in the case of using our service. These measurements that are shown in Table 1 show the amount of transmitted bytes that have been saved using packet compression service. Based on these statistics, we can conclude that using packet compression service it is possible to reduce the transmitted bytes over a congested or high cost link.

Table 1. Statistics of packet compression service about the amount of transmitted bytes have been saved.

Test	Saved transmitted bytes in the case of using packet compression service
Iperf (default parameters)	9.61 MB
Iperf (a 30-second execution time)	18.30 MB
Iperf (transferring 200 of data)	32.31 MB

5. Conclusion

In this paper, we showed how virtualization technology, could help implementing a useful network service for compressing network packets. Other similar approaches to implement the packet compression technique either need to alter the application or the operating system's kernel. In addition, they may bring extra overhead due to transferring network packets upwards, compress them and then send them back to the kernel. To avoid these pitfalls, we proposed to implement such a network service inside the virtual machine monitor and using a configurable kernel module. To proof the feasibility of the proposed approach, we implemented this service and installed it on a Xen-based virtual environment. Evaluation results show that not also implementing and deploying of such a service is possible in a virtual environment but also it helps the compression process to perform faster according to the lack of unnecessary upcalls. The measured also results showed that using this service may lead to 25% increase in network bandwidth.

References

1. C. Shen Tye and G. Fairhurst, A review of IP packet compression techniques, in PostGraduate Networking Conference, (2003).
2. M. Degermark, B. Nordgren, and S. Pink, IP Header Compression, RFC 2507, (1999).
3. A.I. Sundararaj, A. Gupta, and P.A. Dinda, Increasing application performance in virtual environments through run-time inference and adaptation, in Proc. of the 14th IEEE Int. Symp. on High Performance Distributed Computing (HPDC), (2005) 47-58.
4. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in Proc. of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP'03), (2003) 164-177.
5. VMWare Corporation, http://www.wmware.com.
6. W. Huang, J. Liu, B. Abali, and D. K. Panda, A case for high performance computing with virtual machines, in Proc. of the 20th Annual international Conference on Supercomputing (ICS '06), (2006) 125-134.
7. R. J. Figueiredo, P. A. Dinda, and J. A. Fortes, A case for Grid computing on virtual machines, in Proc. of the 23rd International Conference on Distributed Computing Systems (ICDCS), (2003) 550-561.
8. L. Rudolph, A virtualization infrastructure that supports pervasive computing,IEEE Pervasive Computing, vol. 8, no. 4, (2009) 8-13.
9. R. L. Grossman, The case for cloud computing, IT Professional, vol. 11, no. 2, (2009) 23-27.
10. M. A. Vouk, Cloud computing - issues, research and implementations, in Proc. of the International Conference on Information Technology Interfaces (ITI'08), (2008) 31-40.
11. A. Gupta and P. A. Dinda, Inferring the topology and traffic load of parallel programs running in a virtual machine environment, in 10th Workshop on Job Scheduling Strategies for Parallel Processing (JSPPS 2004), (2004).

12. A. Gupta , M. Zangrilli , A. I. Sundararaj , A. I. Huang , P. A. Dinda and B. B. Lowekamp, Free network measurement for adaptive virtualized distributed computing, in 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), (2006).
13. S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, Antfarm: tracking processes in a virtual machine environment, in Proc. of the Annual Conference on USENIX '06 Annual Technical Conference, (2006) 1.
14. S. Kumar and K. Schwan, Netchannel: a VMM-level mechanism for continuous, transparentdevice access during VM migration, in Proc. of the Fourth ACM SIGPLAN/SIGOPS international Conference on Virtual Execution Environments (VEE '08), (2008) 31-40.
15. A. Hamidi, H. Salimi, and M. Sharifi, Network Service Provisioning using System-Level Virtualization, in The First Workshop on Provisioning and Management of Service Oriented Architecture and Cloud Computing (PROMASC 2010), In conjunction with the NOTERE'2010 Conference, (2010).
16. A. Shacham, B. Monsour, R. Pereira, and M. Thomas, IP payload compression protocol (IPComp), Network Working Group, RFC 3173 (2001).
17. S. Kent and R. Atkinson, Security architecture for the internet protocol, Network Working Group, RFC 2401 (1998).
18. HP Company, HP Case Study: WAN Link Compression on HP Routers, (1995).
19. V. Jacobson, Compressing TCP/IP Headers for Low-Speed Serial Links Status, Network Working Group, RFC 1144 (1990).
20. C. Bormann, et al, RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed, RFC 3095, (2001).
21. M. Engan, S. Casner, and C. Bormann, IP header compression over PPP, Network Working Group, RFC 2509 (1999).
22. M. Degermark, M. Engan, B. Nordgren and S. Pink, Low-loss TCP/IP header compression for wireless networks, Wireless Networks, vol. 3, no. 5 (1997) 375–387.
23. M. Lee, H. Jin, I. Kim and T. Kim, Improving TCP Goodput over Wireless Networks Using Kernel-Level Data Compression, in Proc. of the 18th international Conference on Computer Communications and Networks, (2009) 1-6.
24. WANProxy - A multi-platform open source WAN-optimizing proxy server, http://wanproxy.org.
25. A. Jones, Netfilter and IPTables – a structural examination, SANS Institute Reading Room site (2004).
26. P. J. Salzman, M. Burian, and O. Pomerantz, The Linux kernel module programming guide. (2007).
27. Iperf: Measuring Maximum TCP and UDP Bandwidth Performace, http://iperf.sourceforge.net.