Journal of Network and Computer Applications 39 (2014) 104–116

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

DMVL: An I/O bandwidth dynamic allocation method for virtual networks

Huailiang Tan [a,*], Lianjun Huang [a], Zaihong He [a], Youyou Lu [b], Xubin He [c]

[a] College of Information Science and Engineering, Hunan University, Changsha 410082, China
[b] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[c] School of Engineering, Virginia Commonwealth University, Richmond 23284, VA, USA

ARTICLE INFO

Article history:
Received 13 November 2012
Received in revised form
6 May 2013
Accepted 28 May 2013
Available online 4 June 2013

Keywords:
Virtualization
I/O bandwidth
Fairness
Stability
Dynamic allocation
Lower cost

ABSTRACT

In a consolidated server system that uses virtualization, accesses to physical network devices from guest virtual machines (DomUs) need to be coordinated. In this environment, virtualized network devices are required to service workloads executing concurrently from multiple DomUs, with potentially diverse network data delivery requirements. Although a number of methods have been developed for I/O performance virtualization among multiple DomUs, previously proposed researches focused either on improving network I/O performance and lowering overhead from hardware and software, or on achieving network I/O fairness by directly applying special physical network interface cards. We argue that it is important to allocate network I/O bandwidth fairly and stably among multiple DomUs based on pure software approaches and not to hinder live migration and portability of virtual machines. This paper proposes Dynamic Mapping of Virtual Link (DMVL) method, which prevents the interference between multiple DomUs by introducing separated Logical Data Path (LDP) and I/O request queue for each DomU. In DMVL, several techniques are employed. Firstly, we provide isolated I/O bandwidth guarantees to each DomU by mapping a separate LDP to I/O device for each DomU and using multi-queue. Then, the adjusted LDP bandwidth quantity is converted into adjusted credit count, and credit transferring and updating methods based on shared logs are introduced to adjust LDPs bandwidth dynamically. Finally, we improve lottery scheduling algorithm based on shared logs to implement adaptive bandwidth adjustment to meet proportional multi-queue scheduling at lower cost. The proposed techniques are implemented on the Xen virtual network, and evaluated with micro-benchmarks and simulated workloads on Linux guest operating systems. Experimental results show that DMVL improves fairness by at least 60% and stability by at least 29% in the cases of three or more virtual DomUs running on the same physical host.

1. Introduction

With increased maturity of virtualization technology, virtual machine systems have been widely used in both desktop and enterprise server environments. Virtualization allows multiple OSes to share a single physical device in order to maximize physical resource utilization in a computer. However, it brings on the problem of unfair resource distribution. Virtual I/O is a major performance bottleneck in virtual machine systems (Shafer, 2010). For I/O intensive workloads, CPU cycles are wasted while CPUs are waiting for data available or spinning on idle cycles, and this decreases overall system performance and scalability. Therefore, rational allocation of I/O bandwidth is particularly important.

In general, the I/O subsystem, including the network and the disks, plays a key role in a computer system. Unfortunately, traditional virtual machines focus on process scheduling, which addresses the fairness of processor resources sharing among multiple domains, leaving I/O scheduling as a secondary concern. Meanwhile, I/O scheduling in guest OSes does little to improve global fairness of I/O resources for the following two reasons. One is that each DomU views only its own virtual disk, so the corresponding scheduling algorithms are designed depending on its virtual disk. Due to the lack of inconsideration of I/O characteristics from other DomUs and physical disk status, the disk scheduling algorithms are blind from overall viewpoint. For example, an I/O request scheduled by shortest-seek-time-first (SSTF) spends the shortest seek time on a virtual disk. But the I/O scheduler in Dom0 (control domain) aggregates all I/O requests from multiple DomUs, the request's time is undetermined after the request is turned into a real disk I/O request. In this sense, SSTF cannot be ensured. The other is that traditional virtual machine I/O

* Corresponding author. Tel.: +86 18674823096.
E-mail addresses: tanhuailiang@hnu.edu.cn (H. Tan), hljhnu@hnu.edu.cn (L. Huang), hezaihong@hnu.edu.cn (Z. He), luyy09@mails.tsinghua.edu.cn (Y. Lu), xhe2g@vcu.edu (X. He).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.05.010

scheduler lacks unified management, which leads to competition for disk I/O bandwidth by multiple DomUs. Some real key DomUs' applications cannot be responsed in proper time because of insufficient disk I/O bandwidth when relatively less important DomUs' applications hold superabundant bandwidth.

Network I/O resources have similar problems. Multiple DomUs willfully consume network I/O resources based on respective maximal requirement. Each DomU's maximal bandwidth of network devices is restricted by the credit mechanism in traditional virtual machines. However, the credit mechanism is simple and defective. Firstly, it adopts static deployment mode, in which the credit number allocated to each DomU is fixed on creating and cannot be changed after booting. Secondly, it merely restricts maximal bandwidth, and the credit number cannot be dynamically allocated and adjusted based on network traffic status. As a result, bandwidth waste may be induced in some DomUs. This would cause much degradation of I/O performance such as the bandwidth and latency, and reduce the stability of I/O access, which makes the performance of DomU's applications unpredictable. These negative influences would make virtualization less desirable for I/O intensive applications whose performance is critically dependent on I/O latency or bandwidth (Ongaro et al., 2008). Although a self-virtualization device and the VM-bypass method can effectively efface the I/O performance degradation caused by the scheduling delay (Liu et al., 2006; Raj and Schwan, 2007; PCI-SIG, 2011), they cannot be applied to the traditional device sharing cases.

At the same time, virtual machine systems are required to serve different workloads simultaneously with potential diverse data delivery requirements and different I/O request characteristics. Since I/O requests of all applications accessing one I/O device are aggregated to a uniform queue after passing through I/O ring and Back-end driver etc., I/O request queue in Dom0 is shared by all DomUs' virtual devices. Traditional VM I/O schedulers usually employ FIFO algorithm to schedule all kinds of missions serially. In this case, a request is scheduled in its arrival sequence. Since requests from different applications are generated in different frequencies and vary in real-time characteristics, if a lot of requests from other DomUs are backlogged in the queue and wait to be scheduled, the latest arriving requests have to wait. On the contrary, if other DomUs have no I/O requests, the queue is free and the latest arriving requests can be scheduled immediately. So the I/O performance experienced by some DomUs suffers from the variations in I/O request stream characteristics of other DomUs. It is obvious that a shared FIFO queue in Xen cannot meet the requirement of fairness.

Recent research has been conducted on improving virtual I/O performance (Menon et al., 2006; Santos et al., 2008; Menon and Zwaenepoel, 2008; Liao et al., 2008; Liu et al., 2006; Raj and Schwan, 2007; Abramson et al., 2006; Willmann et al., 2007; Dong et al., 2009; Jiuxing, 2010; Yaozu et al., 2010; PCI-SIG, 2011). But they largely focus on improving network I/O performance and lowering overhead from either hardware (Liu et al., 2006; Raj and Schwan, 2007; Abramson et al., 2006; Willmann et al., 2007; Dong et al., 2009; Jiuxing, 2010; Yaozu et al., 2010; PCI-SIG, 2011) or software (Menon et al., 2006; Santos et al., 2008; Menon and Zwaenepoel, 2008; Liao et al., 2008; Cherkasova and Gardner, 2005), or on achieving network I/O fairness by directly applying special physical network interface cards (Anwer et al., 2010; Egi et al., 2008). We believe that it is important to allocate network I/O bandwidth fairly and stably among multiple DomUs based on pure software approaches and not to hinder live migration and portability of virtual machines at lower cost. An I/O bandwidth allocation mechanism for virtual networks should have the following features:

(1) fairness, stability and performance isolation: to provide proportional share fairly to different DomUs' applications and guarantee stable bandwidth allocation;
(2) higher aggregated I/O bandwidth: to achieve high aggregated I/O bandwidth compared with original virtual machines;
(3) lower overhead: not to introduce high overhead and affect DomU's applications execution efficiency;
(4) lower cost: not to bring extra hardware expenses by using pure software approaches.

This paper presents Dynamic Mapping of Virtual Link (DMVL) method to transparently provide an I/O Logical Data Path (LDP) for each virtual machine. First, a theoretical DMVL model is built from fairness and stability aspects of I/O bandwidth allocation, by which the DMVL monitor allocates and adjusts each LDP's I/O bandwidth fairly in a comprehensive view based on the characteristics of each virtual field's workloads. Each LDP includes logically all parts of an I/O path exclusively in order to isolate I/O data stream of each DomU and ensure stable bandwidth of each virtual link. To this end, the method to evaluate the fairness and stability of I/O bandwidth allocation is devised.

Second, the shared I/O request queue is addressed by respectively establishing an I/O request queue for each LDP. The shared I/O request queue stems from native driver I/O scheduling strategy and tends to bring on the issue of fairness. In our approach, the LDP bandwidth is adjusted with the increase or decrease of the credit count, and shared logs are introduced for tracking credit usage and other information to implement adaptive bandwidth adjustment mechanism. With the bandwidth adjustment method based on shared logs, the bandwidth limitation module is improved. Besides, the lottery scheduling algorithm, which maps lottery tickets to the credits and assigns lottery tickets to each DomU's queue, is improved to implement adaptive bandwidth adjustment for meeting proportional multi-queue scheduling at lower cost.

These proposed techniques are implemented in virtual networks with the credit scheduler which is the latest scheduler of Xen VMM. To lower the temporal and spatial cost and simplify the management, our DMVL method uses a bidirectional index link-table to link each LDP's request queue for conveniently adding and removing DomUs. Since the implementation is confined to the Dom0 driver layer without any modification to guest kernels, a variety of OSes can exploit our method. In the evaluation section, we carry out comprehensive experiments to evaluate DMVL performance and make a comparison between DMVL and the original Xen. Results show that DMVL method significantly improves fairness and stability. In the case of concurrent hybrid workloads, fairness is increased by more than 57% when compared with the original Xen. In addition, we demonstrate that DMVL method improves I/O bandwidth utilization by 2.5% with overhead of mere 2–5%. Our DMVL method is completely based on pure software approaches and has no need of special physical network hardware, which does not hinder live migration and portability of virtual machines, so previous software-based optimizations can be directly integrated into DMVL.

The main contributions of our work include

(1) isolated I/O bandwidth is guaranteed by using the multi-queue method and mapping a separate LDP to each I/O device for each DomU;
(2) the LDP bandwidth is quantified and measured with the credit count, and the credit transferring and updating methods based on shared logs are introduced to dynamically adjust LDP bandwidth;
(3) lottery scheduling algorithm based on shared logs is adopted to adaptively adjust the bandwidth with the purpose of proportional multi-queue scheduling at low cost.

The remainder of this paper is organized as follows. Section 2 describes the DMVL architecture and the algorithms. Section 3 illustrates