International Journal of Future Generation Communication and Networking
Vol. 8, No. 6 (2015), pp. 123-134
http://dx.doi.org/10.14257/ijfgcn.2015.8.6.12

A Queuing based Network I/O Scheduling with QoS Guarantee for Virtual Machine

Xiaodong Liu, Zhengying Wen and Miao Wang

School of Computer, Henan Institute of Engineering, Zhengzhou, 451191, China
liuxiaodongxht@qq.com, 3094207@qq.com, wmscan@tom.com

Abstract

Since multiple VMs share the same network interface card (NIC). The I/O operation of one virtual machine is affected by others. The quality of service (QoS) of the network is hardly guaranteed. This paper presents a queuing based I/O scheduling for big data processing via the cloud, an efficient scheduling strategy with quality of service (QoS) guarantee. Our proposed scheme uses a queuing system to control the I/O bandwidth and provide QoS guarantee for each application. The proposed techniques are implemented on the xen virtual machine monitor (VMM) and evaluated with micro-benchmarks on Linux operation system.

Keywords: *I/O scheduling, virtual machine, queuing system*

1. Introduction

Many fields, such as Internet applications, Internet of Things, e-commerce, telecom, health care, financial services, etc, generate large amounts of data [1, 2]. Deep analyzing and processing data can help us gain great value from big data. Nowadays, parallel computing model, such as MapReduce [3], BSP [4], has been widely used in big data processing. Big data is divided into a set of sub-data. These sub-data are processed respectively and they exchange data information by means of communication. So, I/O has an important affect on the performance of big data processing.

Meanwhile, computing systems have undergone a fundamental transformation from single-processor devices to the large scale data center, such as cloud computing [5]. In the virtualization based cloud computing environment, multiple computers are allowed to run as virtual machines (VM) on a single physical computer. These VMs share the underlying physical resources, such as I/O bandwidth. So, the I/O bandwidth of one application is affected by others. How to guarantee the QoS of network I/O has been a key issue.

Several techniques have been proposed for improving I/O performance [6-9]. Software based [6-8] and hardware based [9] I/O virtualization technology are applied in virtualization to improve network I/O performance. Such I/O technologies are advantageous in high performance computing, and cloud computing environment, where I/O is a major performance bottleneck.

Although the I/O performance of VMs is improved by these technologies, the QoS of network I/O cannot be guaranteed. Since VMs share the same network equipment, the I/O performance of an application running on the VM will be impacted by other applications' I/O operation running on the same physical computer.

In an effort to guarantee QoS of network I/O, some techniques have been proposed to provide QoS guarantee for network I/O. The virtualization technology, such as xen [10], provides I/O network bandwidth limiting strategy to guarantee each VM's I/O bandwidth. Building separated virtual router for VMs [11, 12] or limiting the network I/O rate or time of VMs [13-15] are both used to guarantee the QoS of network. However, the limiting strategy is static allocate. The I/O network bandwidth cannot be adjusted dynamically. The I/O network bandwidth utilization is low. Firstly, some applications, such as big data

ISSN: 2233-7857 IJFGCN

processing using parallel computing method [3, 4], are not always use network I/O. When these applications are not communicates with each other, the network bandwidth will be wasted. Secondly, diverse applications inside VMs may have different network requirements. Unpredictable I/O workloads worsen the semantic gap between bandwidth allocation and I/O requirement of VMs [16], leading to inefficient I/O bandwidth allocation and low bandwidth utilization. Thirdly, the I/O bandwidth cannot be reallocated when we create new VMs or destroy old VMs.

This paper presents a queuing based I/O scheduling for big data processing via the cloud. I/O requests of applications running on VMs are considered as guest and network driver is considered as server system. VMs are allocated a certain number of network credits periodically. VMs will consume the network credits when they execute I/O operation and they cannot execute I/O operation when network credits reduce to zero. Queuing rules are established according to network credits and workloads of VMs. the I/O workloads are predicted through network credits information including allocated credits, consumed credits and remained credits. Network credits are allowed to lend or borrow among VMs to improve network bandwidth utilization.

The proposed techniques have been implemented on the xen VMM [10]. The distinguished features of VCF compared to previous work are as follows: Firstly, the proposed technique schedule network I/O resources in a formal method and it can effectively reason about CPU resources scheduling. Secondly, because the proposed technique use queuing model, performance analysis of network I/O is easy. Thirdly, our scheme is workload-aware. The credits of VMs can be lent or borrowed according to the I/O workload.

The proposed techniques are evaluated for various workloads in terms of guaranteeing QoS of the network. We demonstrate that QoS of VMs can be guaranteed when multiple VMs share the same physical network. In addition, we show that the unused bandwidth of the VM can lend to VMs which need more bandwidth.

The remainder of this paper is organized as follows: Section 2 presents the background and related work. The system model is described in Section 3. Section 4 presents our queuing based I/O scheduling strategy. Section 5 shows the experimental results. Finally, Section 6 summarizes our conclusions and suggests future work.

2. Background and Related Work

Our work is motivated by previous work on improving I/O performance of VMs and providing QoS guarantee for VMs. This section presents the background and related work.

2.1. The I/O Virtualization of Xen VMM

Xen adopts split virtual driver model by introducing a privileged driver domain (Dom0). Figure 1 describes the xen I/O architecture. A virtual driver consists of two split sub-drivers, respectively called front-end driver (FE) and back-end driver (BE). The FE driver provides interfaces of the virtual driver to Guest OS, but the FE driver cannot conduct real I/O operations. The Dom0 conducts real I/O operations on behalf of Guest OS. Communications between FE and BE are implemented by the I/O ring mechanism. The I/O ring adopts producer/consumer communication mode. When Guest OS need perform I/O operation, they put read/write requests into I/O ring. The BE will read these requests, process them and put response information into I/O ring. Processed response information can be read by the FE. FE and BE notify each other of an I/O event via an event channel. This I/O model improves the efficiency of the I/O device and enhances the reliability of an entire system.

Since multiple VMs' I/O requests are put into the same response queue and BE process these I/O requests uses first come and first serve (FCFS) strategy. The I/O performance of

one VM is affected by others, and the QoS of network I/O can hardly be guaranteed. This paper will research on providing QoS of network I/O for VMs which share the same physical network.

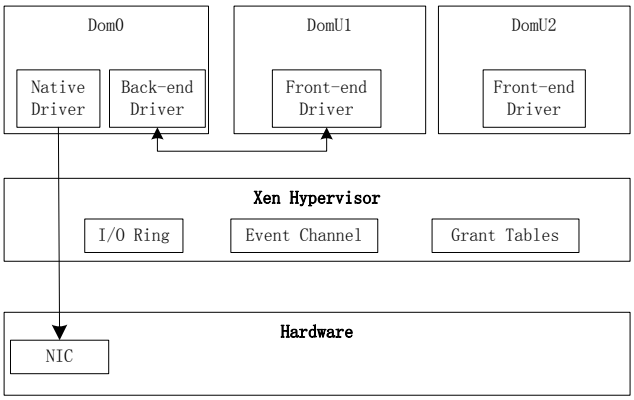


Figure 1. Xen I/O Architecture

2.2. Improve I/O Performance for Vms

The VCPU scheduling has an important affect on I/O performance. Ludmila Cherkasova and Diwaker Gupta *et al.*, [17] analyze the impact of the choice of scheduler and its parameters on I/O performance. Diego Ongaro and Alan L. Cox *et al.*, [18] study the impact of scheduler on I/O performance using multiple virtual machines concurrently running different types of application. Memory and cache also affect I/O performance. Guangdeng Liao *et al.*, [7] proposed a cache-aware scheduling and credits steal strategy to improve I/O performance. Their mechanism reduces inter-domain communication cost and accelerates packet processing. Jose Renato Santos *et al.*, [19] reduce memory management and data copy overheads to improve I/O performance. There also has been recent work on improving I/O performance through optimize I/O channel. Aravind Menon *et al.*, [8] optimize I/O channel for transferring network packets between the guest and driver domains. This strategy avoids a data remap or copy in the common case. J. Lakshmi and S.K. Nandy [20] propose an extension to the I/O architecture. The architecture allows native access of I/O device. The semantic gap between the VMM and VMs can lead to inefficient I/O scheduling. HwanjuKim *et al.*, [16] present a virtual machine scheduling technique for transparently bridging the semantic gap between the VMM and VMs in order to improve I/O performance without compromising CPU fairness. Task-aware scheduling [21] and Communication-aware scheduling [22] are both scheduling strategies to improve I/O performance by bridging the semantic gap between the VMM and VMs.

2.3. Providing QOS Guarantee for VMS

Some techniques have been proposed for guaranteeing QoS of network I/O. These techniques can be classified into two types: (1) separated I/O channel based. Norbert Egi *et al.*, [12] propose a technology that builds of a virtual router for each VM to guarantee each VM's I/O performance. The authors of [11] propose a binding technique which static building of virtual routers to cores and dynamic mapping forwarding path to core. (2) Limiting strategy. The authors of [13] propose a rate limiting mechanism which can stop any VMs from sending or receiving more than its fair share. It guarantees each VM's I/O performance by limiting I/O rate of each VM. VIOS [15] controls the allocation of time to the different VMs. It guarantees each VM's I/O performance by limiting I/O operation time of each VM.

3. System Model

This section describes the system model of our queuing based I/O scheduling.

3.1. System Model

We suppose there are N guest domains (DomUs) and one privileged domain (Dom0). As is shown in Figure 2, Each DomU has a front-end driver (FE) and the privileged domain (Dom0) has a back-end driver (BE) which has privileged access to physical I/O device. When DomUs need perform I/O operations, DomUs put I/O request put into the response queue which located in the Dom0. These I/O request will be processed by BE using first come first serve (FCFS) strategy. Since all DomUs' I/O requests are put into the same response queue. The I/O performance of one DomU is affected by others. The network QoS of DomUs are hardly guaranteed.

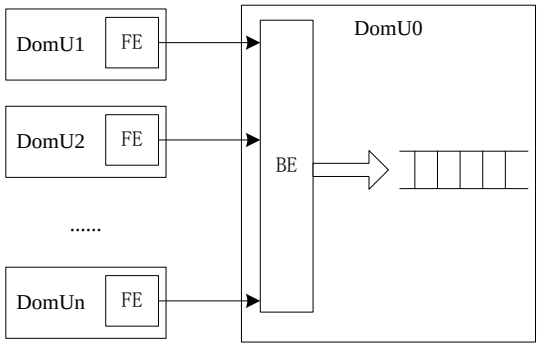


Figure 2. I/O Device System Model

3.2. Problem Description

The queuing based I/O scheduling is the problem of providing QoS guarantee for VMs. queuing based I/O scheduling guarantees that each DomU can use its allocated I/O bandwidth. Meanwhile, when the DomU's network is idle, it can lend its network bandwidth to those VMs which need more network bandwidth.

The queuing based I/O scheduling is a periodically scheduling algorithm. The terminology and notations used are as follows: we denote (1) the network bandwidth of physical computer by B, (2) the i-th DomU by VM_i, (3) the applied network bandwidth of the VM_i by B_i^{apl}, (4) the actual network bandwidth of the VM_i by s_i (5) the scheduling period by ΔT, (6) the i-th scheduling period by t_i.

The I/O bandwidth is converted into network credits in our queuing based I/O scheduling. Network credits are periodically allocated to VMs. when VMs perform I/O operation, they will consume network credits. The VM cannot perform I/O operation when the credits of the VM are no more than zero unless it gets network credits again.

Definition 3.1. We define the total credits in each scheduling period are $B \times \Delta T$, and we define the allocated network credits of the VM_i in each scheduling period are $B_i^{apl} \times \Delta T$.

Definition 3.2. Let s be the size of data sent by the VM_i, we define the consumed network credits of the VM_i are s.

Considering the above situations, the goal of our queuing based I/O scheduling is to providing QoS guarantee of I/O bandwidth for VMs. Meanwhile, the I/O requirements of VMs are different, and the I/O requirements of the VM may vary. The goal of our queuing based I/O scheduling also should be flexible. It should schedule I/O according to VMs requirements.

The network I/O scheduling problem can then be formulated as following:

(1) If the needed network bandwidth B_i^{ned} of the VM_i less than B_i^{apl}, the allocated bandwidth credits of the VM_i in the next scheduling period will be $B_i^{ned} \times \Delta T$.

(2) If the needed network bandwidth of the VM_i and VM_j are both more than their applied network bandwidth, the goal of I/O bandwidth guarantee can be described using the following definition.

Definition 3.3. The QoS of I/O bandwidth can be guaranteed if there exists a small constant θ that in any time intervals Δt, the actual bandwidth of any pair of virtual machines (VM_i, VM_j) meet the following formula.

$$\left| \frac{B_i(\Delta t)}{B_i^{apl}} - \frac{B_j(\Delta t)}{B_j^{apl}} \right| \leq \theta \quad (1)$$

Table 1 gives the list of symbols we use in this paper along with their meaning.

Table 1. Description of Symbols Used in this Paper

Symbol	Description
N	The number of VMs
Δt	The time intervals
$B_i(\Delta t)$	The consumed bandwidth credits in the time interval Δt
ΔT	The scheduling period
$C_i^{rem}(t_i)$	The remained network credits of the VM_i in the i-th scheduling period.
$C_i^{alc}(t_i)$	The allocated network credits of the VM_i in the i-th scheduling period
$C_i^{req}(t_i)$	The needed network credits of the VM_i in the i-th scheduling period
s_i	The I/O resources status of the VM_i
$C^b(t_i)$	The total network credits that VMs request borrow in the (i+1)th scheduling period
$C^l(t_i)$	The total network credits that VMs request lend in the (i+1)th scheduling period

4. The Queuing Based I/O Scheduling

In this section, we present our queuing based I/O scheduling, which aims to provide QoS guarantee of network I/O for VMs.

4.1. Queuing System

This sub-section presents our queuing system which is used to control the I/O bandwidth and provide QoS guarantee for VMs. one advantages of queuing system is that it is also a performance analysis model. We will evaluate the performance of I/O and obtain some important performance indicators in the next.

We consider I/O requests of VMs as “customer”. Customers can be single or batches arrive. Network I/O is controlled by queuing rules.

Definition 4.1. If the network credits of the VM not more than zero, the customer from the VM is not allowed to queue. Or else, if the network credits of the VM more than zero, the customer from the VM is allowed to queue.

Customers are served by the BE. The serving time is related to the size of the data. The larger the data is. The long the serving time is.

Considering the above situations, the QoS of the network I/O is related to the network credits. The QoS of the network I/O can be controlled by network credits. The queuing based I/O scheduling provides QoS guarantee for VMs through controlling network credits of VMs. In general, the queuing based I/O scheduling consists of two phases: (1) predicting future I/O resources requirements, in which the running I/O information of VMs is collected. The future I/O resources requirements of VMs can be predicted based on the collected information. (2) I/O scheduling phase, in which the scheduling of I/O resources is done.

4.2. Predicting Future I/O Resources Requirements

We first give the definition of network credits.

Definition 4.2. Let Δt is the scheduling period. We define the total network credits in each scheduling period C_{total} as $B \times \Delta t$. We suppose the applied network bandwidth of the VM_i is B_i^{apl}. Then, the allocated bandwidth credits of the VM_i in each scheduling period should be $B_i^{apl} \times \Delta t$.

Definition 4.3. Let d is the size of data send by the VM_i in a scheduling period. We define the consumed network credits of the VM_i in the scheduling period are d.

The queuing base I/O scheduling periodically collects the running I/O information, including the allocated network credits, the remained network credits and the consumed credits. Based on these collected information, the I/O resources requirements of VMs in the next scheduling period can be predicted.

(1) If $C_i^{rem}(t_i) \le 0$, the allocated network credits of the VM_i are consumed completely. Let t_{con} is the time of these allocated network credits are consumed completely. The needed network credits of the VM_i in the next scheduling period can be calculated as follows.

$$C_i^{req}(t_{i+1}) = \frac{C_i^{alc}(t_i) \times \Delta t}{t_{con}} \tag{2}$$

(2) If $C_i^{rem}(t_i) > 0$, the allocated network credits are not consumed completely, which means that there are surplus network credits in the VM_i. These surplus network credits can be lent to VMs which need more network credits. The needed network credits of the VM_i in the next scheduling period can be calculated as follows.

$$C_i^{req}(t_{i+1}) = C_i^{alc}(t_i) - C_i^{rem}(t_i) \tag{3}$$

The predicting future I/O resources requirements algorithm is formalized in algorithm 1.

Algorithm 1. The predicting algorithm of the I/O resources

1 **for** each VMs **do**
2 record the $C_i^{rem}(t_i)$, $C_i^{alc}(t_i)$ and t_{con}
3 if $C_i^{rem}(t_i) \le 0$ **then**
4 calculate the needed network credits using equation (2)
5 else **if** $C_i^{rem}(t_i) > 0$ **then**
6 calculate the needed network credits using equation (3)
7 **end for**

4.3. I/O Scheduing Phase

This subsection presents our I/O resouces scheduling strategy. To motivate further discussion, we first define the satus of I/O resources.

Definition 4.4. we define the I/O resouces status of the VM_i as follows:

$$s_i = \begin{cases} 1 \\ 0 \qquad \text{mn} \\ -1 \end{cases}$$

After the I/O resouces statuses of VMs are defined, we can then calculate the following two total value.

$$C^b(t_i) = \sum_{\substack{i=1 \\ s_i=1}}^{N} (C_i^{req}(t_{i+1}) - C_i^{alc}(t_i)) \tag{4}$$

$$C^l(t_i) = \sum_{\substack{i=1 \\ s_i=-1}}^{N} (C_i^{alc}(t_i) - C_i^{req}(t_{i+1})) \tag{5}$$

where $C^b(t_i)$ are the total credits that VMs request borrow in the next scheduling period, and $C^l(t_i)$ are the total surplus credits that VMs request lend in the next scheduling period.

Thus, the I/O resources scheduling can be divided into three conditions:

(1) if $C^b(t_i) < C^l(t_i)$, there are enough network credits that can satisfy VMs which request more. The I/O resources are scheduled as follows.

(a) The allocated network credits of VMs which s_i is 1 can be calculated as follows.

$$C_i^{alc}(t_{i+1}) = C_i^{req}(t_{i+1}) \tag{6}$$

(b) The allocated network credits of VMs which s_i is 0 can be calculated as follows.

$$C_i^{alc}(t_{i+1}) = C_i^{alc}(t_i) \tag{7}$$

(c) The allocated network credits of VMs which s_i is -1 can be calculated as follows.

$$C_i^{alc}(t_{i+1}) = C_i^{req}(t_{i+1}) + \frac{C^l(t_{i+1}) - C^b(t_{i+1})}{\sum_{\substack{i=1 \\ s_i=-1}}^{N} B_i^{apl}} \times B_i^{apl} \tag{8}$$

Eq.(6) indicates that VMs which need more network credits can be satisfied completely. Eq.(8) indicates that VMs which have idle network credits need not lend all their idle credits.

(2) if $C^b(t_i) = C^l(t_i)$, the total network credits that VMs request borrow are equal to the total network credits that VMs request lend. The allocated network credits of all VMs in the next scheduling period can be calculated using Eq. (7).

(3)if $C^b(t_i) > C^l(t_i)$, there are not enough network credits can satisfy VMs which request more credits. The I/O resources are scheduled are scheduled as follows.

Step 1. Network Qos guarantee phase, in which each VM's network I/O is guaranteed.

The requested network credits of each VM are Recaculated as follows.

$$C_i^{req}(t_{i+1}) = \begin{cases} C_i^{req}(t_{i+1}) & \text{if } C_i^{req}(t_{i+1}) < B_i^{apl} \\ B_i^{apl} \times \Delta t & C_i^{req}(t_{i+1}) \geq B_i^{apl} \end{cases} \quad (9)$$

Eq.(9) indicates that the applied network bandwidth of the VM can be guarantee.

After the requested network credits are recaculated, The I/O resources status of VMs and the total network credits that VMs request borrow in the next scheduling period are also need to be recalculated.

Step 2. The idle network credits are allocated to VMs whose I/O resources status is 1. We suppose the total applied network bandwidth is not more than B, the total idle network credits B_{total}^{idle} can be calculated as follows.

$$B_{total}^{idle} = B - \sum_{i=1}^{N} C_i^{req}(t_{i+1}) \quad (10)$$

These idle network credits will be allocated to VMs whose I/O resources status is 1.

If s_i =1, the allocated network credits of the VM_i in the next scheduling period can be calculated as follows.

$$C_i^{alc}(t_{i+1}) = C_i^{req}(t_{i+1}) + \frac{(C_i^{req}(t_{i+1}) - C_i^{alc}(t_i))}{C^b(t_i)} \times B_{total}^{idle} \quad (11)$$

If s_i =0 or s_i =-1, the allocated network credits of the VM_i in the next scheduling period can be calculated by equation (6).

The predicting future I/O resources requirements algorithm is formalized in algorithm 2.

Algorithm 2. The predicting algorithm of the I/O resources

1 convert bandwidth into bandwidth credits according the definiton 4.2
2 calculate $C^b(t_i)$ and $C^l(t_i)$
3 **for** each VMs **do**
4 **if** $C^b(t_i) < C^l(t_i)$ **then**
5 **if** s_i =1 **then**
6 calculate $C_i^{alc}(t_{i+1})$ according to the Eq.(6)
7 **else if** s_i =0 **then**
8 calculate $C_i^{alc}(t_{i+1})$ according to the Eq.(7)
9 **else if** s_i =-1 **then**
10 calculate $C_i^{alc}(t_{i+1})$ according to the Eq.(8)
11 **else if** $C^b(t_i) = C^l(t_i)$ **then**
12 calculate $C_i^{alc}(t_{i+1})$ according to the Eq.(7)
13 **else if** $C^b(t_i) > C^l(t_i)$ **then**
14 calculate $C_i^{req}(t_{i+1})$ and B_{total}^{idle} according to the Eq.(9) and Eq.(10)
15 **if** s_i =0 or s_i =-1 **then**
16 calculate $C_i^{alc}(t_{i+1})$ according to the Eq.(6)
17 **else if** s_i =1 **then**
18 calculate $C_i^{alc}(t_{i+1})$ according to the Eq.(11)
19 **end for**

5. Performance Evaluation

We have implemented the queuing based scheduling algorithm in the xen 4.1.2 hypervisor. In this section, we compare the performance of the queuing based scheduling algorithm (QBS_xen) with original xen scheduling (Orgi_xen).

Two computers were used to evaluate the QBS scheduler. One was a dual-socket and quad-core Inter Xeon CPU running at 2.40GHZ, 32 GB of RAM (PM1). The operation system is running the 64-bit version of Ubuntu 12.04. This one was acted as host machine and was running xen 4.1.2. The system is configured with one to multiple DomUs and one Dom0. The DomUs and Dom0 are both configured with 1G memory and a single virtual CPU. The operating system of Dom0 and DomUs are the same as the host machine. Another is an Intel dual-core CPU running at 2.93GHZ, 2G of RAM (PM2). The operating system is ubuntu 12.04. This one is used as server during the experiment described below.

The httperf[23] is used in our experiments. In the httperf benchmarks, all DomUs send http requests to PM2.

We first concurrently run three DomUs with the same bandwidth. Figure 2 shows the actual throughput of three DomUs on the machine PM1. Figure 2 (a) shows the throughput of three DomUs in the orginal xen scheduling. When three DomUs sending I/O request in the same time, they will influence each other. The throughput of three DomUs is different, and the difference of DomUs' network can even reach 50%. The QoS of the DomU can hardly be guaranteed. Figure 2 (b) shows the throughput of three DomUs in the queuing based I/O scheduling. The throughput of three DomUs is almost the same, which is caused by limiting network credits for each DomU.

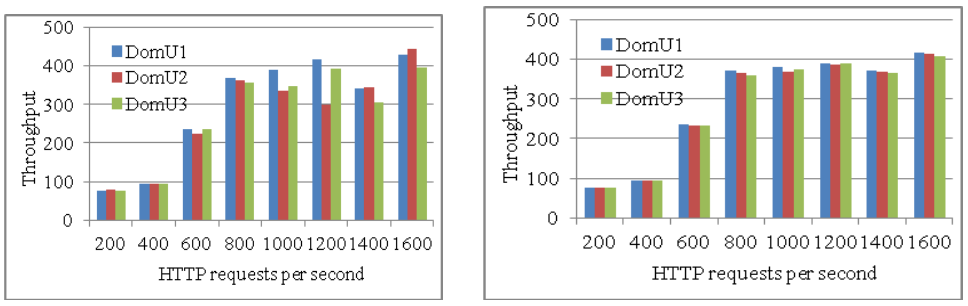
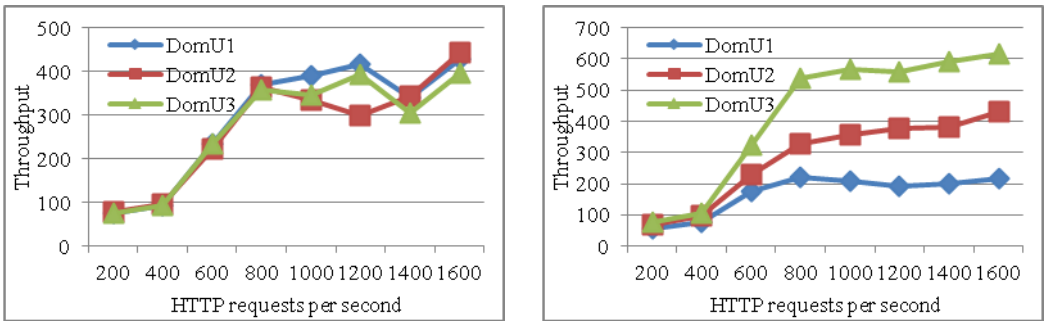


Figure 2. http Throughput for Three Domus Running with the same Bandwidth

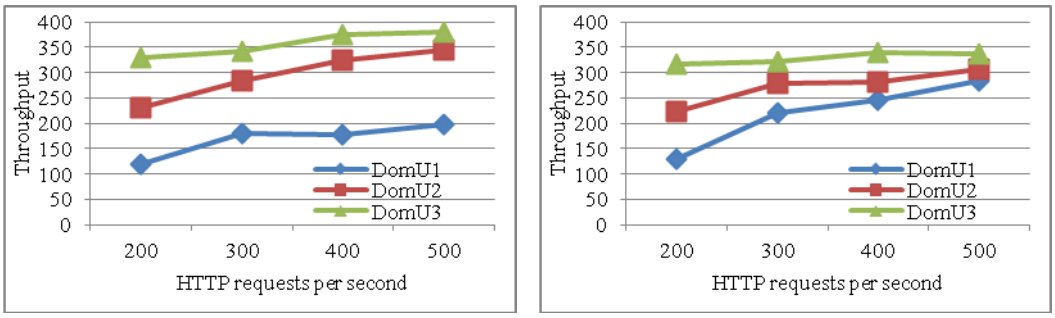
We then concurrently run three DomUs with different bandwidth. The bandwidth ration of DomUs is 1:2:3. Figure 3 shows the plot of throughput of three DomUs against request rate for the httperf. Figure 3 (a) shows the throughput of three DomUs in the orginal xen scheduling. The throughput of three DomUs is almost the same. The throughput of the DomU which applied more bandwidth is the same to the DomU which applied less bandwidth. That is to say, the QoS of the DomU which applied more bandwidth can hardly be guaranteed. The I/O schedule of Orgi_xen use Fist come First Server (FCFS). Multiple DomUs share the same I/O response queue, and they compete for network bandwidth. Figure 3 (b) shows the throughput of three DomUs in the queuing based I/O scheduling. As we can see from the Figure 3 (b), the DomU which applied more bandwidth will get more I/O resources, and the network bandwidth of the VM can be guaranteed. This is because the queuing based I/O scheduling controls network bandwidth by limiting network credits for each VM.



(a) (b)

Figure 3. http Throughput for Three Domus Running with Different Weight

Thirdly, we concurrently run three DomUs with the same bandwidth and different sending rate. In this set of experiments, Three DomUs simultaneously send http request with the rare ration is 1:2:3. Figure 4 shows the experiment result. The x-axis is represents the sending rate of the DomU1. The sending rate of the DomU2 and DomU3 are twice and three times as quick as the DomU1 respectively. As is shown in the Figure 4 (a), because there is no I/O limit in the Orgi_xen, the throughput of the DomU is related to its sending rate. The network bandwidth of the VM which sending rate is lower is affected by the VM which sending rate is quicker. The QoS of the VM which sending rate is lower can hardly be guaranteed. Figure 4 (b) shows the throughput of three DomUs in the queuing based I/O scheduling. As we can see form the Figure 4 (b), when the network is idle, the throughput of the VM is related its sending rate. When the network is busy, the QoS of the VM's network I/O can be guaranteed by queuing based I/O scheduling.



(a) (b)

Figure 4. Http Throughput for Three Domus Running with Different Sending Rate

6. Conclusion and Future Work

In this paper, we have proposed the queuing based I/O scheduling for guaranteeing the QoS of network I/O. Network bandwidth is converted into network credits. The network I/O scheduling is converted into increased or decreased network credits. The queuing based I/O scheduling is a periodical scheduling algorithm. With each periodical round, the I/O resources requirements are diagnosed based on the collected I/O information. Then, VMs are divided into three statuses according to I/O resources requirements and their run information. The queuing based I/O scheduling periodically schedule I/O resources according to VMs' status. Our proposed algorithms have been implemented on the xen hypervisor 4.1.2.

One advantages of our proposed method is that it is also a performance analysis model. In the future, we will evaluate the performance of I/O and obtain some important performance indicators.

Acknowledgments

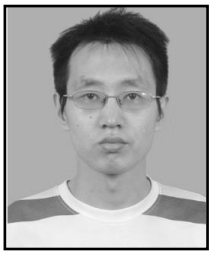
This work is supported by Foundation of He'nan Educational Committee (15A520055,13A520148)

References

[1] C. Lynch, "How do your data grow? [J]", Nature, vol. 455, no.4, **(2008)**.
[2] D. Goldston, "Data wrangling [J]", Nature, vol. 455, no.4, **(2008)**.
[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters [J]", Communications of the ACM,vol. 51, no. 1, **(2008)**, pp. 107-13.
[4] X. Liu, W. Tong, Z. Fu and W.Z. Liao, "BSPCloud: A Hybrid Distributed-memory and Shared-memory Programming Model", International Journal of Grid and Distributed Computing, vol. 6, no. 1, **(2013)**, pp. 87-97.
[5] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility[J]", Future Generation Computer Systems, vol. 25, no. 6, **(2009)**, pp. 599-616.
[6] H.J.H. Kim and J.Lee, "XHive: Efficient Cooperative Caching for Virtual Machines", IEEE Transactions on Computers, vol. 60, **(2011)**, pp. 106-119.
[7] G. Liao, D. Guo, L. Bhuyan and S.R. King, "Software techniques to improve virtualized I/O performance on multi-core systems", in Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, **(2008)**, pp. 161-170.
[8] A. Menon, A.L. Cox and W. Zwaenepoel, "Optimizing network virtualization in Xen",in Proc. USENIX Annual Technical Conference (USENIX 2006), **(2006)**, pp. 15-28.
[9] J. Liu, W. Huang, B. Abali and D. K. Panda, "High performance VMM-bypass I/O in virtual machines", in Proceedings of the annual conference on USENIX, **(2006)**, pp. 29-42.
[10] 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", ACM SIGOPS Operating Systems Review, vol. 37, **(2003)**, pp. 164-177.
[11] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield and M. Williamson, "Safe hardware access with the Xen virtual machine monitor", in 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (OASIS), **(2004)**.
[12] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici and L. Mathy, "Fairness issues in software virtual routers", in Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow, **(2008)**, pp. 33-38.
[13] M.B. Anwer, A. Nayak, N. Feamster and L. Liu, "Network I/O fairness in virtual machines", in Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures, **(2010)**, pp. 73-80.
[14] A. Gulati, A. Merchant and P. J. Varman, "mClock: handling throughput variability for hypervisor IO scheduling", in Proceedings of the 9th USENIX conference on Operating systems design and implementation, **(2010)**, pp. 1-7.
[15] S. R. Seelam and P.J. Teller, "Virtual I/O scheduler: a scheduler of schedulers for performance virtualization", in Proceedings of the 3rd international conference on Virtual execution environments, **(2007)**, pp. 105-115.
[16] H. Kim, H. Lim, J. Jeong, H. Jo, J. Lee and S. Maeng, "Transparently bridging semantic gap in cpu management for virtualized environments", Journal of Parallel and Distributed Computing, vol. 71, **(2011)**, pp. 758-773.
[17] L. Cherkasova, D. Gupta and A. Vahdat, "When virtual is harder than real: Resource allocation challenges in virtual machine based it environments", Hewlett-Packard Labs, **(2007)**.
[18] D. Ongaro, A. L. Cox and S. Rixner, "Scheduling I/O in virtual machine monitors", in Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments,**(2008)**, pp. 1-10.
[19] J.R. Santos, Y. Turner, G. Janakiraman, and I. Pratt, "Bridging the gap between software and hardware techniques for i/o virtualization", in Proceedings of the USENIX'08 Annual Technical Conference, **(2008)**.
[20] J. Lakshmi and S. Nandy, "I/O Device Virtualization in the multi-core era", a QoS perspective, in Grid and Pervasive Computing Conference, 2009. GPC'09. Workshops at the, **(2009)**, pp. 128-135.
[21] H. Kim,H. Lim, J. Jeong, H.Jo and J. Lee, "Task-aware virtual machine scheduling for I/O performance", in: Proc. of 2009 ACM SIGPLAN/ SIGOPS International ConferenceonVirtualExecutionEnvironment,VEE'09,Washington,DC,USA, no. 3, **(2009)**.

[22] S. Govindan, J. Choi, A.R. Nath, A. Das, B. Urgaonkar and A. Sivasubramaniam, "Xen and Co.: communication-aware CPU management in consolidated Xen-based hosting platforms", IEEE Transactions on Computers, vol. 58, **(2009),** pp. 1111-1125.
[23] D. Mosberger and T. Jin, "httperf—a tool for measuring web server performance", ACM SIGMETRICS Performance Evaluation Review, vol. 26, **(1998)**, pp. 31-37.

Authors



Xiaodong Liu, he received his Ph.D. degree from Shanghai University. He is currently an associate Professor of Henan Institute of Engineering. His primary research interests cover virtualization, cloud computing and grid computing

Zhengying Wen, she received the master degree in computer science from Huazhong University of Science and Technology. Her research interests include image processing, database, *etc*.

Miao Wang, he received the PhD degree in computer science from Harbin University of Science and Technology. Member of China Computer Federation. His research interests include Spatio-temporal database, Real-time database and Spatial Reasoning.

