An Interrupt Scheduling Mechanism of Virtual Network Interface Card for Xen-Based System

Liuzheng Lu [1, a], Runze Wan [2,1, b] * and Jinrong He [3, c]

[1]School of Computer, Hubei University of Education, Wuhan 430205, China;

[2]Co-Innovation Centre of Elementary Education Information Technology Services of Hubei, Wuhan 430205, China.

[3] Science Computing and Intelligent Information Processing of Guangxi Higher Education Key Laboratory, Guangxi Teachers Education University, Nanning 530299

[a] lzlu@mail.hue.edu.cn, [b] runzewan@126.com, [c] hejinrong@163.com

*In this paper, Runze Wan is the corresponding author.

Keywords: virtualization; interrupt; latency; virtual buffer.

Abstract. As one of typical paravirtualization hypervisors, Xen has received widespread attentions especially its scaling capability under some kinds of workload. In this paper, we focus on the problem that the most CPU resources are occupied by frequent interrupt from the NIC(Network Interface Card) and it will cause bottleneck of the system for Xen. To alleviate this problem, this paper proposes an adaptive interrupt latency scheduling mechanism based on XEN, which use the polling or interrupt method in accordance with the queue length of virtual buffer without supplementing any additional processing unit. Also, the method can guarantee different quality of service to some extent by means of the definition of the two types of priority virtual buffers. Simulation results show that the mechanism can reduce CPU overhead significantly and improve system performance effectively.

I. INTRODUCTION

Virtualization technology is being widely used in the test of hardware and software, server consolidation, security and disaster recovery. Virtualization technology refers to the realization of utilization of hardware, which can maximize a hardware entity into a set of logical entity to meet the requirements of different business. From the point of view of hardware, the research of virtualization technology such as virtual processor, virtual device and virtual memory are developing rapidly[1].

The DMA virtualization is built in the North Bridge Chipset for the hardware realization of virtual IRQ. VT-d[2] technology is a kind of hardware assisted virtualization technology, which can be generated from the I/O equipment interrupt request accurately to specify different virtual machine.

Taking into account the behavior of a virtual machine may affect the allocation of resources of other virtual machines, Gupta et al. [3] put forward a set of primitives to ensure performance isolation, which use XenMon to measurement the resource consumption of each VM accurately and adopt SEDF-DC scheduler to allocate the resource of CPU. VPE[4] proposes one of the nuclear separated to handle the I/O operation request scheme for the low rate of utilization of multi-core CPU, which can improve the performance of virtual I/O. NCQ[5] applies the multi-end queue NIC to reorder the received packet for improving I/O throughput. Credit[6] is a kind of proportional fair sharing of non preemptive scheduling algorithm, which set the parameter domain scheduling weights for client operating system domain to handle the I/O request.

Due to the use of interrupt mechanism, CPU will stop the current thread as the external network data arrived. However, frequent interruption lead to the fully occupied of processing resources as the packets arrived continually, which will result in the poor performance. To resolve the problem, this paper proposes a according to network busy degree automatic adjustment. An automatic switching mode between I/O mechanism and Virtual system is implemented in Xen, which can deal with the

data in the buffer queue length of virtual machine rather than need to increase additional processing units.

II. VIRTUALIZATION ARCHITECTURE OF XEN

Xen is one of the most widely virtualization technology for virtual service application, the logic structure as shown in figure 1. The framework allows any non-modification device driver to drive the domain performs independently in the host. Figure 1 shows the architecture of the Inter-Domian on Xen virtual machine. Dom 0 is a privileged domain, and Dom U is a client domain for customers. IDD is the driver domain, where provides the device driver for customers to visit the hardware. Figure 2 shows the Xen network operating mode. While Dom U need to receive the packets from network, Dom 0 immediately write this part of the data in a shared data structure inside, and get through the event channel to inform Dom U. Then, the Hypervisor checks the event and set up an analog channel to the Dom U. Next, the sub operating system regards it as an external interrupt, and call the NIC driver interrupt handler which can be captured by the Hypervisor. Eventually the packets can be read in shared memory.

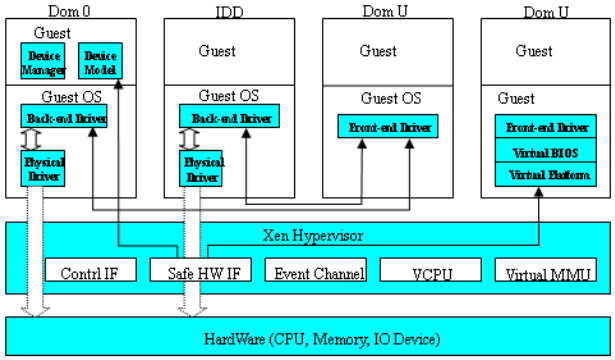

Fig. 1 Inter-Domian on Xen virtual machine

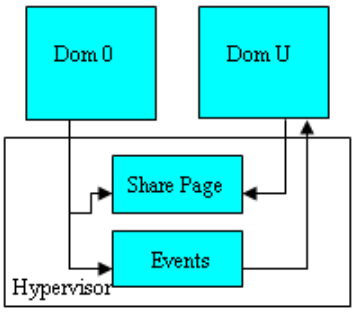

Fig. 2 Virtualization architecture of Xen

At present, the research of the I/O technique often focuses on the network throughput. But for other indicators, i. e. the CPU overhead, is not being paid attention. Actually, in the applications of request-response frequently network services, NIC should carry through the interruption to notify the CPU after receiving the packet. On the one hand, most of the CPU resources will be frequently interrupted by the arrival of the card. On the other hand, lower priority packets may not be timely implemented. This will cause a buffer overflow of the network subsystem, and the received packet may be dropped. If this situation lasts longer, it will enable network service request to reach saturation quickly, and packet forwarding throughput will drop to zero. Finally, the system CPU resources are suspended due to lack of other processes.

Taking into account the interruption caused by CPU context switching overhead is very large, the network operational modes is modified in [7] by the following methods. Firstly, after adding the NIC delay interrupt mechanism, a timer hardware is added to control the frequency of processing requests Dom U. Secondly, only one network I / O request is occurred within the time interval of clock control for sub operating system. Thus, entire CPU overhead can be shorten by controlling I / O

interrupt request. However, there are also the following disadvantages: (1) Modification of physical infrastructure and the hardware cost; (2) Since every context switch CPU need that all the data in the buffer card can be processed completely and then to exit, it is not flexible enough. (3) Differentiated demands are not being taken into account in accordance with the different QoS requirements of packets.

III. MECHANISM OF INTERRUPT LATENCY OF NIC

In this paper, the main idea is that front end of the virtual buffer data fetch time interval can not be set as a fixed time interval, but adjusted dynamically according to the amount of unsent data in buffers. This allows the CPU is able to arrange job execution plan in terms of the interval size for the next cycle. Besides, the time span of CPU processing can be set adaptively according to the overload status of buffers. If the queue is too long and the job execution program of the CPU is not very busy after adjustment. The CPU processing length can be extended to ensure that the buffers not being overflowed during the next cycle.

Definition 1: The number of virtual cards is n .The queue length of i-th buffer is $q_i(i=1,2,\cdots,n)$ with the priority weight w_i .And the waiting time for i-th queue is denoted as T_i .

Definition 2: Each queue is eager for be scheduled as early as possible, then we define the value of aspiration:

$$\varepsilon_i = w_i * \ln(\frac{T_i}{T_0} * \frac{q_i}{\min_{i\in[0,1,\cdots,n]}\{q_i\}}) \tag{1}$$

where the length and the weight of each queue is considered in the calculation.

Assuming that the time interval in j-th cycle is T_j , the CPU time being allocated for i-th buffer can be given as the following rule:

$$t_{ij} = \frac{\varepsilon_j}{\sum_{i=1}^{n}\varepsilon_i} * T_j \tag{2}$$

Therefore, the amount of packets that can be processed by CPU is calculated by $C(j) = t_{ij} * q_i$. We can discuss two cases to design the interrupt latency mechanism:

(1) If $\sum_{i=1}^{n} q_i - C(j) > 0$, the packets in all buffers are unprocessed and the amount of residual packets is $\sum_{i=1}^{n} q_i - C(j)$. Assuming that the average generation rate of packets from physical NIC to virtual hosts is λ , the latest overload is λT_j . Then, the time interval in the next cycle should be set as :

$$T_{j+1} = \frac{\lambda T_j + [\sum_{i=1}^{n} q_i - C(j)]}{C(j)} \tag{3}$$

(2) Otherwise, $\sum_{i=1}^{n} q_i - C(j) \le 0$ denotes that the packets is processed completely in buffers. And the interruption will be occurred while some packets arrived in the buffer. To avoid frequent interruption for CPU as soon as the packets arriving, a threshold value of Q_{Thres} can be considered. When the buffer queue length and the threshold is reached, it will generate an interrupt request to the front end.

IV. Simulation And Results

We construct the experimental platform with a virtual host, which carries a quad core processor Intel (R) core (TM) i5 750 2.67GHZ and 4G memory. Dom 0 is bound to a CPU core, and the remaining three nuclear is shared by other subsystems. The network environment is set up by several Gigabit switches, where open source testing tool for network performance is installed. The operating system of Dom 0 is CentOS 5.3, and driver Domain and other three Guest Domain are equipped with Ubuntu 8.10.

Figure 3 shows the comparison of CPU overhead under the circumstances of different number of virtual machines, and we focus on Dom 0 and Dom U respectively. For Dom 0, the discrepancy is not obvious whether make use of the schedule mechanism. For Dom U, with the increase in the number of virtual machines, the proposed scheduling policy significantly better than when not in use. Figure 2 shows the comparison of the throughput with different number of VMs. As can be seen from the figure, the throughput by applying our mechanism is not reduced significantly compared to the circumstance without optimazation. This means that our scheme can decreases the number of CPU interruption, and meanwhile keep high throughput.

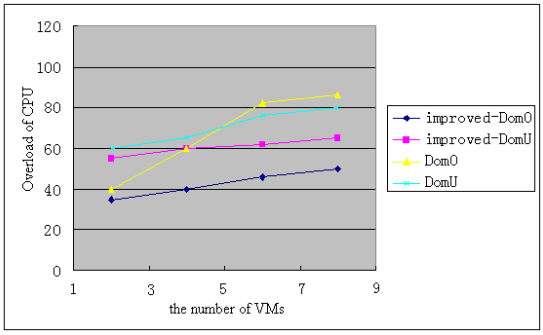

Fig. 3 Comparison of overload of CPU

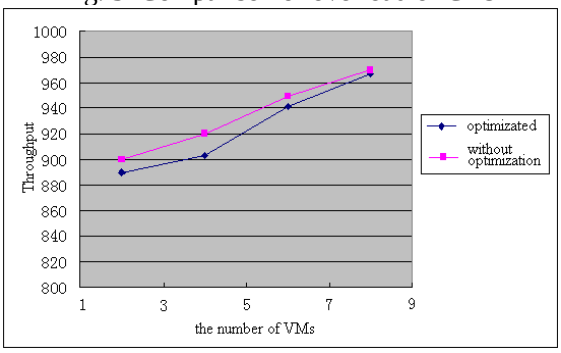

Fig. 4 Comparison of the throughput

V. Conclusion

In this paper, we focus on the problem that the most CPU resources are occupied by frequent interrupt from the NIC(Network Interface Card) . To alleviate this problem, this paper proposes an adaptive interrupt latency scheduling mechanism for Xen-Based System. Simulation results show that the mechanism can reduce CPU overhead significantly and improve system performance effectively.

Acknowledgment

This work is supported by the project of Hubei Co-Innovation Center of Information Technology Service for Elementary Education. This work is also partly supported by Natural Science Foundation of Hubei Province (No. 2015CFB405) and Science Computing and Intelligent Information Processing of Guangxi Hhigher Education Key Laboratory (No. GXSCIIP201406).

REFERENCES

[1] Ludmila Cherkasova,Diwaker Gupta,et al. “Comparison of the three CPU schedulers in Xen”. ACM Sigmetrics Performance Evaluation Review, Sep.2007,42-51.

[2] U Gurav, R Shaikh. “Virtualization-A key feature of cloud computing”. ACM ICWET, Mumbai, Maharashtra, india, 2010:227-229.

[3] Gupta D, Cherkasova L, Gardner R,etal. “Enforcing performance isolation across virtual machines in Xen”. Proceedings of ACM IFIP USENIX 7th International Middleware Conference (Middleware 06). Melbourne, Australia, 2006:342-362.

[4] Jiuxing Liu, Bulent Abali. “Virtualization Polling Engine(VPE): Using Dedicated CPU Cores to A accelerate I/O Virtualization”. ACM Proeeedings of the 23rd international Conference on Supercomputing, 2009.

[5] YoungJin Yu, Dong in Shin, Hyeonsang Eom, Heon Young Yeom. “NCQ vs. I/O Seheduler: Preventing Unexpected Misbehaviors”. ACM Transactions on Storage(TOS), 2010, 6(l):15-27.

[6] Diego Ongaro, Alan L.Cox, Scott Rixner. “Scheduling I/O in virtual machine monitors”. Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2008.

[7] Yang Hongbo. A study of high performance network virtualization technique[D]. Shanghai Jiaotong University, 2011.