Design and Implementation of a Locking-Aware Scheduler for Multiprocessor Environments

Yongseob Lee, Wooseok Son, and Sungyong Park

Sogang University, Seoul, Korea
{misteryos,serom81,parksy}@sogang.ac.kr

Abstract. Virtualization technology has known to be an efficient solution to reduce space and costs by using server consolidation and to provide flexible management of system resources. The widely used virtual machine monitor (VMM) such as Xen is not perfectly optimized for running over multi-processor systems. In order to run VMMs over multi-processor systems without performance degradation, the scheduling algorithms running in the VMM should be modified. This paper proposes a Locking-aware Scheduler (LAS) algorithm that has eliminated the lock-holder preemption (LHP) problem, which causes the inefficient use of CPU resources in multi-processor environments. By using LAS algorithm, Xen allows guest kernel to acquire spin-lock and prevents the preemption of related processors during operation. The performance results show that our proposed algorithm solves the LHP problem and reduces the total execution time.

Keywords: virtual machine monitor (VMM), scheduler, multi-processor, Locking-aware Scheduler.

1 Introduction

Virtualization, a technique that enables the sharing of computer resources without software modifications, has gained a lot of attention since 1960s. However, the rise of multitasking operating systems and the reduction of hardware supply costs in the 1980s throughout 1990s have hindered the development of virtualization. The recent development in computing environments has once again attracted much interest in virtual machines [1]. With the advent of multi-core CPUs, the research efforts have been shifted to develop efficient solutions to run VMMs over multi-processor systems. Although previous studies using one CPU with various virtual machines installed have been conducted, there have been unsatisfactory performance results in the efficient use of virtual machines under multi-processor environments. The spinlock, which is known to increase the performance in a single system, can be the cause of inefficient use of CPU resources in multi-processor environments [2]. Since the VMM directly manages hardware resources, there are cases where the virtual machine is unable to be allocated with resources even if the virtual machine uses synchronization methods such as spinlock to acquire resources. Therefore, although virtual machines use spinlocks, the failure to be allotted with hardware resources prevents operations in the critical section and causes time inefficiencies. [2] explains

G. Lee, D. Howard, and D. Ślęzak (Eds.): ICHIT 2011, LNCS 6935, pp. 384–390, 2011.
© Springer-Verlag Berlin Heidelberg 2011

that the LHP problem is caused by the failure of the virtual machine in exclusive use of resources during resource allocation of the virtual machine monitor. Such problem can be easily solved by limiting the number of virtual machines requesting resource allocation. However, this method is not ideal as it reduces the usage rate of resources and limits scalability. This paper attempts to solve this problem by proposing a newly designed algorithm without limitations in performance and scalability by using a widely used virtual machine monitor, Xen [3][4]. The new algorithm has been designed by recording the time that Xen Hypervisor has allotted to the virtual machine, the guest, as to observe performance degradation due to LHP. Furthermore, the saved information in the Hypervisor has been analyzed to determine the scheduling policies. The performance level of the scheduler has been measured using a widely used bench marking tool, kernbench [5] and sysbench [6]. Results have shown that performance level of virtual machines under multi-processor environment can be improved without limitation in scalability and performance.

Chapter 2 describes the details of the study, Chapter 3 explains the newly proposed scheduler LAS, Chapter 4 verifies the validity of the proposed scheduler through performance tests and Chapter 5 states the conclusion.

2 Related Work

Spinlock in virtual machine can degrade the throughput of the service. Scheduling in Xen refers to the mapping VCPUs (virtual CPU) to PCPUs (physical CPU). VCPU is considered as a process in the virtual machine monitor and therefore virtual machine monitor does not consider the VCPU's status during scheduling. The assumption that spinlock is not preempted during execution may be violated by the virtual machine monitor. The attempt to acquire spinlock by another guest process causes LHP(lock-holder preemption) problem by spending the entire CPU time in acquiring spinlock [2]. Co-scheduling is a technique used in concurrent systems for scheduling related processes to run on different processors at the same time [7]. It can be a solution to the LHP problem however this may cause fragmentation in the PCPU and have negative effects in CPU usage rate. However, it does not provide a clear solution to the LHP problem. To reduce the affection of LHP problem, [2] proposed intrusive solution and non-intrusive solution. In [8], by allowing the preemption of spinlock equipped VCPU, PCPU has been yielded to VCPUs that spin for an unusual time as to propose the Tolerating Lock-holder Preemption algorithm that supports the operation of critical sections by reallocating preempted lock-holders in shortest time.

The algorithms proposed in [2] and [8] relatively improve the performance of virtualization systems but do not completely solve the LHP problem.

3 Locking-Aware Scheduler

LHP problem is caused as the scheduler of the virtual machine monitor conducts scheduling without consideration to the guest status. (Fig. 1) shows the structure of LAS (locking-aware scheduler), which enables interaction between hypervisor (virtual machine monitor of Xen) and the guest. LAS writes the CPU allotted time

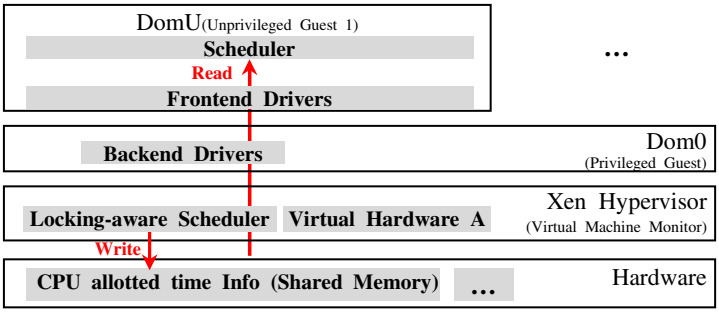

Fig. 1. Lock-aware Scheduler's Structure

data to the shared memory and the guest determines whether to take spinlock or not using the information in the shared memory. Atomic operation is guaranteed for the spinlock of LAS algorithm attained by using Xen Linux 2.6.18 as they have identical movements of a general Linux kernel. Consistency is maintained when failure to take spinlock prevents access to the data that record performance time of critical sections. On the other side, there is the possibility of LHP problem caused by the CPU allotted time method as it does not guarantee atomic operation between kernel process during the process of the guest OS comparing CPU allotted time information from the hypervisor and the time records in the spinlock data structure as to determine spinlock process. In order to solve this problem, the renewal of the maximum time spent had been disallowed when the time spent in critical sections go over n ms. The credit scheduler schedules VCPU in 30 ms intervals in which spinlock time over 30ms have been regarded as an occurrence of the LHP problem as to disallow renewal of maximum time spent. LAS records allotted termination time of PCPU in shared memory as to enable guests to read the information. The guest utilizes the PCPU allotted time information to determine spinlock status at the current point. In order to maintain fairness of scheduling, when the guest domain kernel has yielded CPU as it has determined that the acquisition of spinlock will be unavailable, additional CPU time will be allotted to the to the related VCPU during the following scheduling. Although the use of LAS may solve the LHP problem, it would be an inefficient algorithm if it leads to performance degradation at the overall system level. In order to show that LAS does not cause performance degradation of the system, the performance improvement should be larger than the time costs, the overhead. The expectation time to take a spinlock of the existing scheduler is shown below in (1).

$$O_{\text{time}} = (n+T/2)(S/T) + n(1-S/T) = n+S/2 \ . \tag{1}$$

T represents the length of time (quantum) that the scheduler of the virtual machine monitor allots to the VCPU at once and S represents the length of critical section protected by spinlock. n represents the execution time of spinlock code when LHP problem does not occur. An average of T/2 amount of time is spent to clear spinlock during upon rise of the LHP problem in which the probability of LHP problem can be shown by the ratio of the length of critical section to time length allotted to the VCPU

during scheduling. On the other side, the expectation time to hold a spinlock of LAS is shown in (2).

$$L_{time} = n + h\ . \tag{2}$$

h represents the additional time delay due to shared memory access. As LAS does undergo the LHP problem, the expectation time for acquisition of spinlock for LAS can be shown as the sum of performance time of spinlock acquisition code and time delay due to shared memory access. 2h<S must be satisfied in order for L_{time} to have a smaller expectation value than O_{time} in the two formulas above.

According to [10], the waiting time from spinning to acquire spinlock during the kernbench process has shown to be 2^9~2^{16}cycle in most cases, which was similar to the results of the our experiment. The access time to the shared memory was 65.1 cycle. Therefore, when $S=2^9$ is at a critical level, $2^6<h(=65.1)<2^8(=S/2)$ is satisfied in which it demonstrates that LAS leads to performance improvement not only on average but in critical levels as well.

4 Experimental Results

A server with two Intel Xeon E5345 processors attached has been used in which LAS has been installed in the Hypervisor of the experimental group (marked as Linux-LAS) and a modified spinlock that can be interlocked with LAS has been installed in the guest(Fig. 2). The unmodified distributed version of Xen has been installed in the Hypervisor of the control group (marked as Linux-xen) and unmodified Xen Linux has been installed in the guest. The LHP problem occurs when the number of VCPUs requesting resource allotment exceeds the number of available PCPU. Therefore, each virtual machine has 8 VCPUs, and shares the memory for evaluation conditions. In order to minimize the effects on the experiment, Dom0 has been set to operate minimum processes.

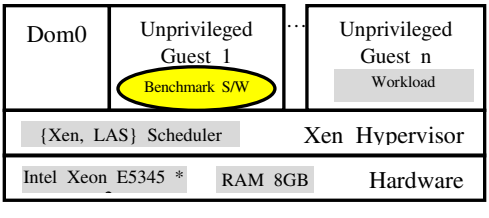

Fig. 2. Experimental Environment

The first experiment has been conducted using a well-known CPU intensive workload performance measuring tool, kernbench[5]. The results present the time spent in compiling the kernel in which smaller values represent higher performance. Experiments have been conducted where 1, 2, 4, 8 guests installed. The performance has been measured with kernbench in one guest in which other guests were used as workload generators which perform infinite loops. This experiment also enables evaluation of performance level and system scalability according to guest increase. The test results are as follows (Fig. 3).

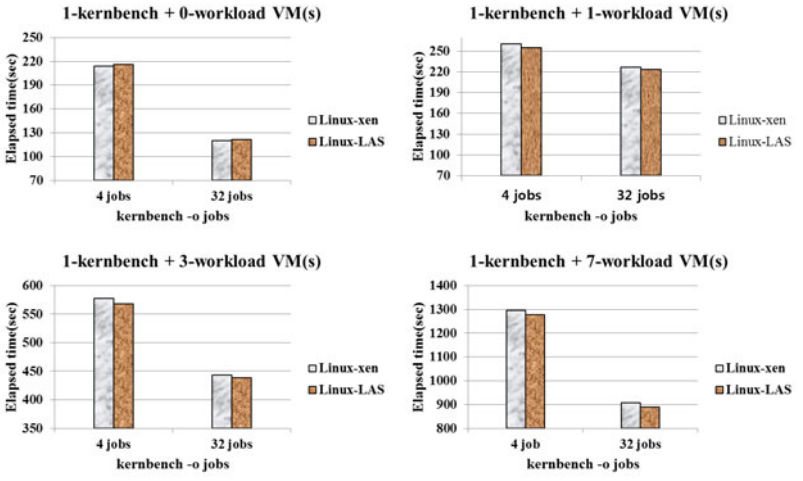

Fig. 3. Kernbench test results

If the LHP problem constantly occurs upon simultaneous operation of various tasks, the costs caused by the LHP problem would exceed the benefits of parallel processing. However, test results have shown that performance can improve with the use of LAS and various number of virtual machines installed. Furthermore, the increase in the performance ratio, even in increase in the number of virtual machines, show that LAS is able to contribute to the scalability of the system.

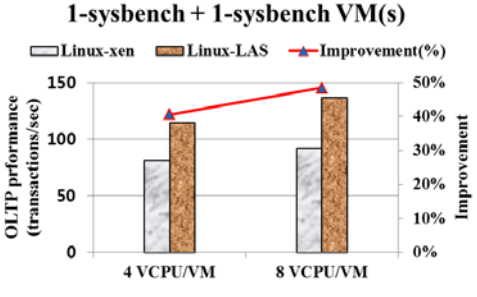

Fig. 4. Sysbench test results

The second experiment has been conducted using sysbench OLTP which enables the evaluation of the overall system and Online Transaction Processing performance level. Since we think that the two guests will be a typical VM configuration in an actual data center, two guests have been installed in the experimental server and tested using the sysbench OLTP test. The measurement values show the maximum number of treatments in which larger values represent higher performance level. The experiment has been done where each virtual machine had 4 or 8 VCPUs. 8 VCPUs have been allotted as to test the performance level with the presence of the LHP

problem and 4 VCPUs have been allotted as to test the performance level of the proposed algorithm without the presence of the LHP problem. Results have shown 48% performance improvement with 8 VCPUs allotted and 41% performance improvement with 4 VCPUs allotted as compared to the virtual machine without any modifications.

5 Conclusion

Results of experiments have shown that LAS is effective in increasing performance level under multi-processor Xen virtualization system as it decreases the LHP problem. Moreover, it has also revealed that the ratio of the performance improvement increases in positive relation with the number of virtual machines, which demonstrates improvement in system scalability as well. OLTP test results, a widely used test not only in intensive CPU tasks but also in data centers as well, have shown that LAS can be utilized in wide scope. In the proposed LAS, the hypervisor acquires guest kernel spinlock and determines the acquisition status of the related spinlock based on the information recorded at the shared access memory. This serves as an advantage, as compared to the existing experiments, in enabling optimization of the spinlock in each kernel. However, although the overhead caused during the information acquisition process of the kernel spinlocks may be small when providing temporary service in short periods, performance improvement may be limited. Although performance improvements have been revealed under conditions similar to the actual environment, the analysis on system operation status and the ability to forecast spinlock acquisition status when there is lack of data on spinlock during initial operation will enable further performance improvement.

Acknowledgment. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency (NIPA-2011-C1090-1101-0008)).

References

1. Rosenblum, M., Garfinkel, T.: Virtual machine monitors: current technology and future trends, vol. 38(5), pp. 39–47. IEEE Computer Society, Los Alamitos (2005)
2. Uhlig, V., LeVasseur, J., Skoglund, E., Dannowski, U.: Towards Scalable Multiprocessor Virtual Machines. In: Proceedings of the 3rd Conference on Virtual Machine Research and Technology Symposium, vol. 3, pp. 4–4 (2004)
3. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the Art of Virtualization. In: SOSP 2003 (2003)
4. Govindan, S., Nath, A.R., Das, A., Urgaonkar, B., Sivasubramaniam, A.: Xen and Co.: Communication-aware CPU Scheduling for Consolidated Xen-based Hosting Platforms, Xen and Co. In: Proceedings of the 3rd International Conference on Virtual Execution Environments, VEE 2007, pp. 126–136. ACM Press, New York (2007)

5. KernBench project, http://freshmeat.net/projects/kernbench
6. Sysbench, http://sysbench.sourceforge.net
7. Ousterhout, J.K.: Scheduling Techniques for Concurrent Systems. In: Proceedings of Third International Conference on Distributed Computing Systems, pp. 22–30 (1982)
8. Friebel, T., Biemueller, S.: How to Deal with Lock Holder Preemption. Presentation at Xen Summit North America (2008)