

Research on Virtual Device Management in Real-time Environment

Wenjia Gong^{1, a}, Yongchao Tao^{1, b}, Wencheng Xiang^{1, c}, Xianghu Wu^{1, d}

¹ Shenzhen Academy of Aerospace Technology, Shenzhen, China

^asjjg_hit@163.com, ^btaoyongchao652@163.com, ^cHPEC_SAAT@163.com ^dwxh_hit@126.com

Keywords: Virtualization, scheduling algorithm, mixed load, Xen

Abstract. In this paper, we present improved methods of credit scheduling algorithm of Xen. By using gray-box knowledge reasoning technology, we are able to identify the IO task in the internal workload of each virtual machine. By using this information of the internal workload, we proposed a Boosting mechanism based on the selection. Selectively raise the priority of the virtual processor to give priority to the corresponding field of the I/O Tasks improve the response speed, and reduce the response time to achieve real-time purposes. In addition, we propose four complementary mechanisms for the shortcomings of the proposed BOOST scheduling algorithm based on the selection. The four complementary mechanisms are the cancellation BOOST priority mechanism, reducing Credit precise mechanism, setting Rate value mechanisms and UNDER queue mechanisms. These four mechanisms can effectively fill the insufficient of BOOST scheduling algorithm based selection, and improve the stability of the algorithm.

1 Introduction

Currently mainstream virtual machine monitors include Xen, Denali, and VMWare. Xen is an open source virtual monitor developed by Cambridge University that is based on quasi-virtualization technology. Xen has given up full virtualization to achieve high performance.

The main function of Xen is to provide a virtualized operating environment (mainly hardware resources) for the guest operating system running on it, and to manage the allocation of these resources to ensure that the virtual machines running on Xen are not influences. A domain is a description of all the hardware environments that are required for the guest operating system to run. Xen uses a mixed mode, so Xen has a privileged domain Domain0 to facilitate managing user VMs.

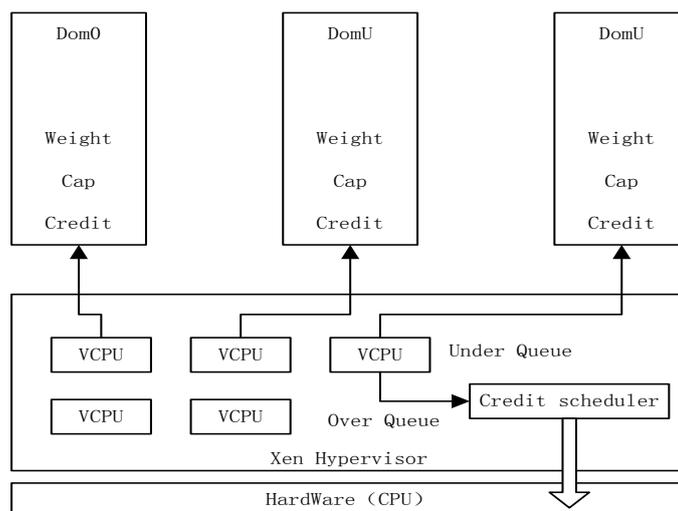


Fig.1. Credit scheduling algorithm

Xen allows domain0 to select three scheduling methods: BVT (borrowed virtual time) scheduling algorithm, Simple Earliest Deadline First algorithm (SEDF) and Credit scheduling algorithm (including improved BOOST mechanism). Among them, Xen has been using the Credit scheduling algorithm as the default scheduling algorithm since version 3.0[1]. The Credit scheduling algorithm is a non-preemptive scheduling algorithm that is fair and shared in proportion.

Figure 1 shows the structure of the Credit scheduling algorithm[2].

The shortcomings of the Credit scheduling algorithm are as follows: 1) the algorithm can't guarantee high response, can't achieve real-time[3]. In the Credit scheduling algorithm, the response delay of the process is highly correlated with the location of the running queue (UNDER queue and UNDER queue). This also led to such applications are relatively large response time, and very prone to response fluctuations, that is, the size of the response time is uneven. Therefore, the Credit scheduling algorithm can't be applied to applications that require high responsiveness and real-time performance[4]. 2) From the perspective of global load balancing, the Credit scheduling algorithm is too simple, which will cause the processor to be idle in some cases, such as the process is not ready and so on[5].

2 Scheduling Algorithm Based on Selection Strategy

In the original Credit algorithm, if an event is woken up in the virtual machine and the VCPU is idle, it will be raised to the BOOST priority regardless of the type of the event. If a VCPU containing a non-I / O event is BOOST before a VCPU containing an I/O event, then the I/O event is scheduled for delay, which still affects the response speed of the I/O event. In order to improve the responsiveness of I/O events, we have improved from the following:

1) I/O tasks can't make good use of the BOOST mechanism because the VMM scheduling algorithm does not recognize the workloads within each guest operating system, and it is not possible to know whether the tasks performed are I/O tasks.

2) If multiple virtual processors are in the BOOST state at the same time, the original scheduling algorithm still needs to be processed in a first-in, first-out manner, which is detrimental to the I / O type task. For example, if the two events have been BOOST, but the first task raised to BOOST priority is not an I / O task, then the I / O task still have to wait.

3) For the task of UNDER queue scheduling, the algorithm does not take full advantage of the characteristics of I / O, that is, consumption of a small amount of Credit value characteristics.

For the above three cases, we make targeted improvements. For the first case, we selectively improve the priority of events that contain I/O tasks to BOOST. If you find that I/O tasks are included at a time, we selectively increase the VCPU priority that contains this event to BOOST.

In the second case, when multiple VCPUs are in the BOOST state, they are scheduled according to the FIFO. However, if the client operating system is not scheduled to be an I/O type task, the VMM scheduler can revoke the VCPU in the BOOST state and revoke the physical CPU assigned to it.

In the third case, when the scheduler dispatches the UNDER queue task, we sort the UNDER queue by calculating Credit. Because the I/O task consumes a small amount of CPU time, it consumes very little credit.

3 Methods for identifying I/O tasks

If you know that the guest operating system is a pure CPU type task or is an I/O sensitive type task, it is not necessary to recognize the I/O type task. We only need to prioritize the VCPU containing I/O sensitive tasks. However, due to the lack of understanding of each customer operating system, we can't predict the internal operating load of the guest operating system, and the workload is always complex and changeable, our job is how to work in a mixed workload to distinguish between I/O tasks. This article will infer whether I / O tasks are based on tracking the client's operating system's kernel scheduling policy at the virtualization level. This method is non-intrusive, and there is no need to modify the kernel of the user's operating system again.

In order to distinguish I/O types of tasks from mixed workloads, VMM needs to track tasks in the domain at the virtualization level. To get a non-intrusive approach, we track the task at the virtualization level by monitoring the Memory Management Unit's entry.

VMM infers the I / O type task in the guest operating system by observing the underlying

interaction between the client kernel and the hardware through the use of gray box knowledge. In addition, we need VMM to maintain a confidence level for each task on whether it is an I/O type task. But here, we use a relatively simple, only part of the knowledge of confidence, to a simple logic to identify whether the I/O task. The I/O task identification process is shown in Figure 2 below.

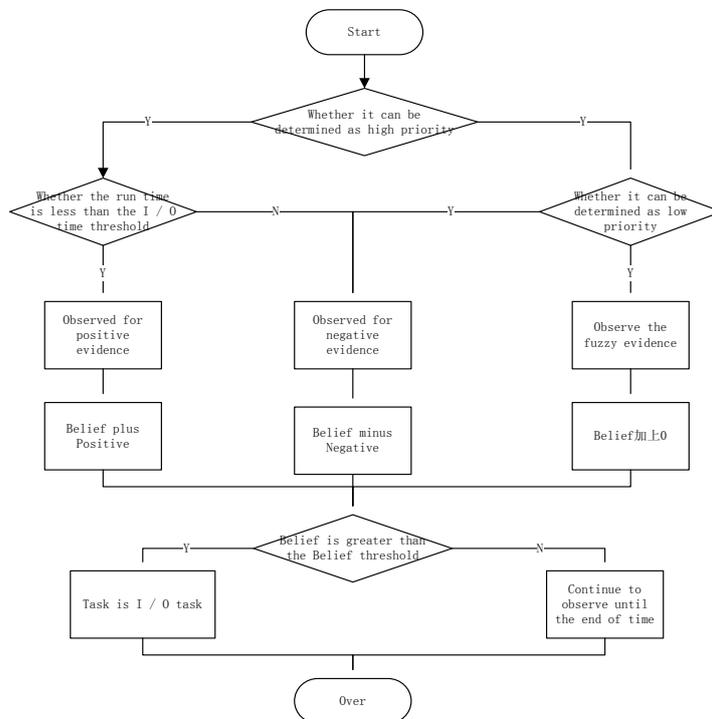


Fig.2. I / O task recognition process

Confidence is a variable that we use to indicate how much a task is likely to be I/O type. At the time of initialization, confidence Belief starts with 0, which means that VMM maintains a fair attitude towards each task at the beginning.

For more intelligent reasoning, VMM can dynamically change the weight of positive and negative evidence, that is, the values of Positive and Negative, by learning the workload in VMM. This can more truly reflect whether a task is an I/O task, reducing the chance of judging the error.

4 Selectively improve the BOOST mechanism

Based on the inferred information of I/O task in the previous section, a BOOST mechanism based on selection is proposed to improve the responsiveness of I/O, and the fairness of CPU time allocation can be ensured at the same time. To improve the responsiveness of I/O tasks while maintaining the fairness of CPU allocation, we only allow I/O type tasks to preempt the running VCPU. As a response to the need for immediate I/O processing, the scheduler will hand over the CPU to another VCPU that runs the I/O task.

When an event is waiting for the VCPU and the event contains at least one I/O task, the VMM scheduler prioritizes the VCPU as BOOST, regardless of the VCPU's original priority. The VCPU in the BOOST state can preempt the running VCPU and then run the wait event. If the client operating system to which the VCPU of the BOOST priority belongs is not scheduled to be an I / O type task, the VMM scheduler can withdraw the physical CPU from the VCPU in the BOOST state and revoke its BOOST priority. The VMM scheduler will reassign the original priority of the scheduled VCPU to him and then insert the VCPU into the UNDER or OVER run queue according to the returned priority.

Figure 3 below is the execution step of the selected BOOST mechanism.

Although the idealized BOOST scheduling mechanism based on the selection will continue for a

short time and will be close to the I / O task processing threshold, but there are still some cases that will lead to a much longer time for this mechanism to be processed.

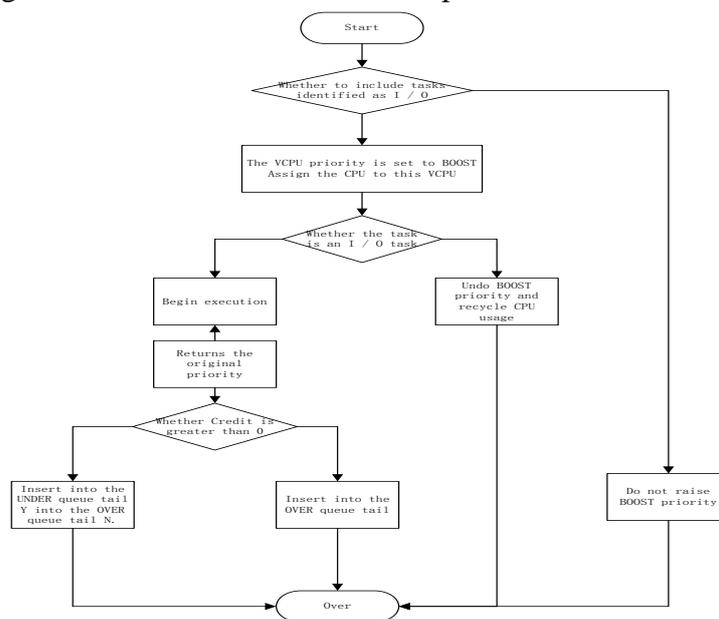


Fig.3. BOOST algorithm flow chart based on the selected

First, in response to the I/O events processed in the kernel, the selective BOOST mechanism can also occur without being delivered to the task. In addition, after a BOOST scheduling mechanism based on the selection, a task that is inferred as an I/O type also consumes the CPU. You can alleviate this unpredictable workload by setting Negative to be larger than Positive. More often, the VMM scheduler can also force the CPU to be allocated to the VCPU to remain based on the selected BOOST mechanism for a period of time. However, this method will result in the overhead of maintaining a separate timer for each VOSTU of BOOST.

5 Test results

The test process is divided into two parts: one is mixed load, the other part is pure I/O load. Mixed load, the virtual three customer domain, through the preparation of the calculation program and send a ping packet to simulate the mixed load situation. Pure I/O load, we send ping packets through three client domains to simulate the test. Each case is run 10 times, taking the average response time.

1) The default Credit scheduling algorithm runs as follows:

Table.1. Default Credit scheduling algorithm

Load condition	Mixed load	Mixed load	Mixed load	I/O load	I/O load	I/O load
Demon	Dom1	Dom2	Dom3	Dom1	Dom2	Dom3
Response time	69.44	74.75	74.13	3.75	4.56	5.07

2) The default Credit scheduling algorithm contains UNDER sorting runs as follows:

Table.2. Default Credit scheduling algorithm contains UNDER sorting

Load condition	Mixed load	Mixed load	Mixed load	I/O load	I/O load	I/O load
Demon	Dom1	Dom2	Dom3	Dom1	Dom2	Dom3
Response time	39.05	37.43	38.05	3.80	4.53	5.05

3) The BOOST scheduling algorithm based on the selected runs as follows:

Table.3. BOOST scheduling algorithm based on the selected

Load condition	Mixed load	Mixed load	Mixed load	I/O load	I/O load	I/O load
Demon	Dom1	Dom2	Dom3	Dom1	Dom2	Dom3
Response time	5.09	5.69	5.67	5.05	5.95	5.76

4) The BOOST scheduling algorithm based on the selected contains UNDER sorting runs as follows:

Table.4. BOOST scheduling algorithm based on the selected contains UNDER sorting

Load condition	Mixed load	Mixed load	Mixed load	I/O load	I/O load	I/O load
Demon	Dom1	Dom2	Dom3	Dom1	Dom2	Dom3
Response time	5.05	5.56	5.60	3.96	5.09	4.75

The experimental results are shown in Figure 4 and Figure 5. It can be seen that the BOOST scheduling algorithm's average response time is very short in I / O scheduling, and the volatility is very small. It can be applied to real-time environment.

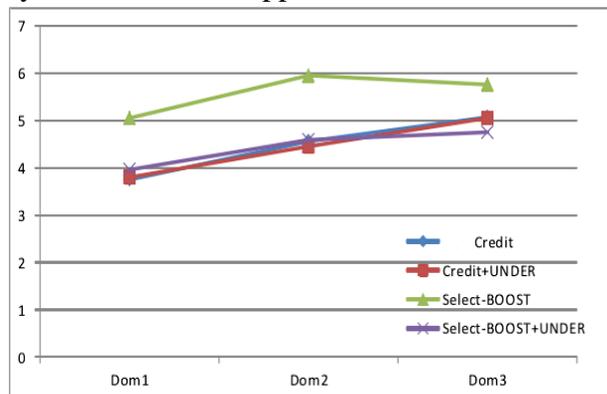


Fig.4. Operation under pure I/O load

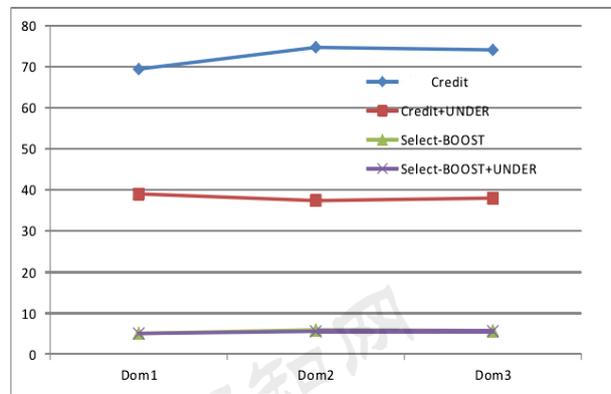


Fig.5. Operation under Mixed load

6 Conclusion

In this paper, the credit scheduling algorithm is improved, and the BOOST scheduling algorithm based on selection is proposed. Selectively improve the priority of the virtual processor containing the I/O task to BOOST, to improve the response speed of the I/O task. In order to identify whether the events in the VCPU contain I / O tasks, we propose an I/O recognition method based on gray box knowledge reasoning. This method can identify I / O types of tasks from complex mixed loads.

Acknowledgement

In this paper, the research was sponsored by the future industry special funds of Shenzhen (Project No. CXZZ20140718154349249).

References

- [1] Jianhua Che, Qinming He, Qinghua Gao et al. Performance Measuring and Comparing of Virtual Machine Monitors[C]. EUC08 Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Volume 2, 2008: 381-386.
- [2] D.Abramson .Jackson, S.Muthrasanallur et al. Intel Virtualization Technology for Directed I/O[J]. Intel Technology Journal, Volume 10 Issue 3, 2006:179-192.

- [3] Kirk L. Kroeker. The evolution of virtualization[J]. Communications of the ACM – Being Human in the Digital Age, Volume 52 Issue 3, March 2009: 18-20.
- [4] Dunlap G.W., King S.T., Cinar S., et al. Re Virt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay[C]. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), New York: ACM, 2002:211-224.
- [5] R.Uhlig, G.Neiger, D.Rodgers, et al. Intel Virtualization Technology[R].IEEE Computer Society, 2005.