vGrouper: Optimizing the Performance of Parallel Jobs in Xen by Increasing Synchronous Execution of Virtual Machines

Peng Jiang[1], Ligang He[1], Shenyuan Ren[1], Junyu Li[1], and Yuhua Cui[2]

[1] Department of Computer Science, University of Warwick, Coventry, CV4 7ES, UK
Ligang.He@warwick.ac.uk
[2] Shandong Worldwide Byte Security Co.ltd cuiyh@jzxtsec.com

Abstract. Xen is one of the most popular virtualization platforms nowadays, which has been broadly used by the industry. Credit scheduler, the default scheduler of Xen, was initially designed for serial jobs, which achieves good performance overall for serial jobs. Unfortunately, the parallel jobs are likely to co-exist with serial jobs in the same host in practice, the resource contention between virtual machines results in severe performance degradation of the parallel jobs. In this paper, we propose vGrouper, a progressive solution to enhance the performance of the parallel jobs. The vGrouper focuses on synchronizing the execution time of the parallel nodes in order to achieve the best performance of the parallel job. Moreover, the vGrouper guarantees that the parallel job nodes are able to run concurrently on pCPUs for the entire time slice, which maximizes the efficiency of communication between parallel nodes. A prototype of vGrouper is implemented, the experimental results demonstrate that the performance of the parallel job and resource utilization in Xen have been significantly improved.

Keywords: Xen · Virtual Machine · Virtualization · Parallel Jobs · Scheduling.

1 Introduction

The Xen virtualization platform has been embraced by industry nowadays due to its impressive scalability and outstanding performance. However, the credit scheduler, which is the default scheduling strategy of Xen, has been identified to be less capable of scheduling parallel jobs. A parallel job usually relies on communication between nodes, which is a completely different working fashion from the serial job. Due to the lack of knowledge of the parallel job, the credit scheduler treats the nodes of each parallel job node as a normal serial job, where the particularity of the parallel job is completely disregarded. The shortcoming of credit scheduler has been discussed in several studies[3][4][2][5].

In this paper, we propose vGrouper, a progressive solution to further enhance the performance of the parallel job to a new level. The vGrouper focuses on synchronizing the execution time of the parallel nodes in order to improve the

performance of the parallel job. Moreover, the vGrouper guarantees that the parallel job nodes are able to run concurrently on pCPUs for the entire time slice, which maximizes the efficiency of communication between parallel nodes.

2 Background and Related Work

2.1 Credit Scheduler

The credit scheduler, which is a proportionally sharing scheduling strategy based on fair allocation of resource, was initially designed for scheduling serial jobs. Each VM will be given credits to consume during execution, which indicates that how many physical resources a VM can have. There two parameters *weight* and *cap* can be used for customizing the bias of resource allocation according to user's need, where the *weight* indicates the relative proportion of execution time and *cap* stands for the maximum amount time of execution time of a VM. In credit scheduler, two priorities are used to indicate the status of a VM. $UNDER$ priority is given to those VMs which are remaining credits, while a VM running out of credits is given $OVER$ priority. Each vCPU of the VM is allowed to executed for a certain time, and the VMs with $UNDER$ priority will be scheduled one by one.

2.2 Related Work

Several optimizations have been made to improve the performance of the parallel job in Xen. Chen and et al. found that overcommitted vCPUs brings performance degradations to concurrent jobs in [1], they mitigate the negative impact by adjusting the time length of execution of VMs according to the type of the workload. Shao and et al. also reveal the problem of overcommitted vCPUs in their research and indicate there is potential penalty on the performance of the parallel job in Xen [3], they choose to expose the workload types of VMs to Xen hypervisor to alleviate the decrease of the performance of the parallel job.

3 Problem Analysis

In the virtualized environment, a pCPU is proportionally shared among several vCPUs, and each vCPU in the local job queue is scheduled for a certain length of time periodically. We first make some assumptions on the environment of the virtual system based on the policies of credit scheduler and common experience of configuration. Firstly, the size of each parallel job is smaller than the number of pCPUs. Secondly, VMs of serial workloads are assumed to be in busy status. Thirdly, the VMs of the parallel job are Uniprocessor systems and only committed to execute the parallel jobs. Forthly, the total number of the VMs in system is much bigger than the number of the pCPUs of the host.

Let p be the number of pCPUs, s be the size of a parallel job and j be the total number of VMs. As the credit scheduler targets for global workload balanced,

the VMs will be evenly and randomly allocated to the pCPUs. Therefore, we know the probability of the parallel nodes being allocated to different pCPUs is:

$$P_A = \frac{\prod_{i=0}^{i<s} p - i}{p^s} \tag{1}$$

Moreover, the nodes of the parallel job should be placed at the same positions of their run queues so that they can be scheduled simultaneously. We know the size of each run queue is:

$$q = j/p \tag{2}$$

Thus, the probability of all parallel nodes being placed at the head of the run queues is:

$$P_B = \frac{1}{q^{s-1}} \tag{3}$$

Therefore, the probability of a parallel job being simultaneously placed on the head of run queues by the credit scheduler is:

$$P_C = \frac{\prod_{i=0}^{i<s} p - i}{p^s} \cdot \frac{1}{q_{min}^{s-1}} \tag{4}$$

We introduce execution efficiency E_s to indicate the percentage of the parallel sub-tasks being simultaneous executed in a time slice. Additionally, as $dom0$ processes the I/O request in Xen, it is compulsory to schedule $dom0$ along with the parallel nodes. Thus, overall evaluation on the efficiency of scheduling a parallel job in Xens:

$$E_{overall} = \frac{\prod_{i=0}^{i<s+1} p - i}{p^{s+1}} \cdot \frac{1}{q_{min}^{s}} \cdot E_s \tag{5}$$

As can be seen, the probability of a parallel job being properly scheduled by credit scheduler is extremely low. Therefore, a co-scheduler is required to assist the credit scheduler to make the appropriate decision on scheduling the parallel job. The problem can be illustrated by Fig 1.

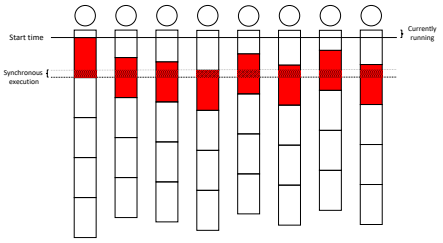

Fig. 1: Problem of parallel job in Xen

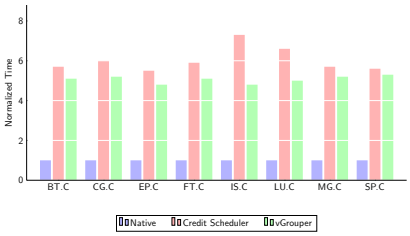

Fig. 2: NPB performance of a parallel job with size of four

4 Co-scheduling solution

The co-scheduling solution solve the problem using three steps. Firstly, it identifies the parallel job in Xen, which requires the VMs of parallel job to be labeled by the users so that co-scheduler is able to identify parallel workload. Secondly, the co-scheduler relocates the VMs of the parallel jobs to avoid overcommitted pCPUs. All VMs of the parallel jobs will be examined in this step, if multiple VMs of a parallel lie on the same job queue of pCPU, the vGrouper is expected to redistribute then to different job queues by migrating. Notably, this step is only taken one time when a parallel job is created. Thirdly, for each parallel job, we choose a flag VM which indicates the parallel job is encountered. When the flag VM is about to be scheduled online, the vGrouper schedule all related VMs of the same parallel job together with the flag VM as a group by boosting the parallel nodes. The boost mechanism of credit scheduler allows current running vCPUs to be preempted by others to accelerate the responding time, which can be used by vGrouper to simultaneously scheduled the VMs of a parallel job together.

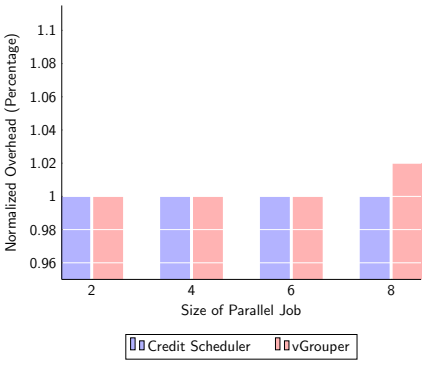

Fig. 3: The overhead of co-scheduling process

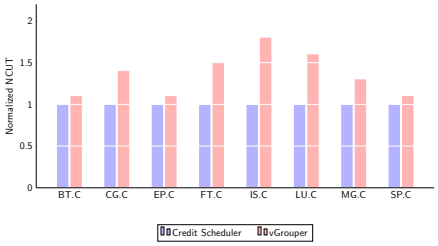

Fig. 4: Number of Communications per Uni Time (NCUT)

5 Experiments and Evaluation

We conduct several experiments to evaluate the performance of vGrouper. Firstly, we test the vGrouper with NPB benchmark suite. As Fig 2 illustrates, the performance of the benchmarking program is significantly increase as expected, especially on IS and LU, which contains lots of communications between the VMs of the parallel job. Secondly, we observe the overhead of the vGrouper. Fig 3 shows that overhead incurred by the vGrouper is negligible, even though that there is a slight increase as the size of the parallel job increases. Finally, we

evaluate the improvement of the utilization. We introduce Number of Communications Per Uni Time (NCUT) as a metric to assess utilization of the system. As can be seen from Fig 4, the frequency of communications in unit time is dramatically increased as the receiving VMs of communications are guaranteed to be online.

6 Conclusion

In this paper, we investigate the reasons for performance degradation of the parallel job in Xen and analyze the importance of simultaneous scheduling to the execution of the parallel job in Xen. we present vGrouper to assist the credit scheduler in handling the parallel application by increasing the length of synchronous execution of a parallel job. The experiments show that vGrouper effectively optimizes the performance of the parallel job in Xen and increases the utilization of the system.

References

1. Chen, H., Jin, H., Hu, K., Huang, J.: Scheduling overcommitted vm: Behavior monitoring and dynamic switching-frequency scaling. Future Generation Computer Systems **29**(1), 341 – 351 (2013). https://doi.org/https://doi.org/10.1016/j.future.2011.08.006, http://www.sciencedirect.com/science/article/pii/S0167739X11001452, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures
2. Huang, W., Liu, J., Abali, B., Panda, D.K.: A case for high performance computing with virtual machines. In: Proceedings of the 20th Annual International Conference on Supercomputing. pp. 125–134. ICS '06, ACM, New York, NY, USA (2006). https://doi.org/10.1145/1183401.1183421, http://doi.acm.org/10.1145/1183401.1183421
3. Shao, Z., Wang, Q., Xie, X., Jin, H., He, L.: Analyzing and improving mpi communication performance in overcommitted virtualized systems. In: 2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. pp. 381–389 (July 2011). https://doi.org/10.1109/MASCOTS.2011.27
4. Weng, C., Wang, Z., Li, M., Lu, X.: The hybrid scheduling framework for virtual machine systems. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. pp. 111–120. VEE '09, ACM, New York, NY, USA (2009). https://doi.org/10.1145/1508293.1508309, http://doi.acm.org/10.1145/1508293.1508309
5. Ye, K., Jiang, X., Chen, S., Huang, D., Wang, B.: Analyzing and modeling the performance in xen-based virtual cluster environment. In: 2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC). pp. 273–280 (Sept 2010). https://doi.org/10.1109/HPCC.2010.79