The Improvement for Virtual Machines Utilization on Cloud Computing Environment

Hsu Mon kyi, Thinn Thu Naing
University of Computer Studies, Yangon
hsumonkyi.ucsy@gmail.com, ucsy21@most.gov.mm

Abstract

Cloud computing is a scalable distributed computing environment in which a large set of virtualized computing resources, different infrastructures, various development platforms and useful software are delivered as a service to customers as a pay-as-you-go manner usually over the Internet. In cloud computing, virtual machines offer unique advantages to the computing community, such as Quality of Service (QoS) guarantee, performance isolation, easy resource management, and the on-demand deployment of computing environments. Virtual machines need to be schedule on the cloud for maximize utilization, do the job faster and consume less energy. This system presents the VM scheduling algorithms using backfilling and gang scheduling approaches to maximize the VM resource utilization.

1. Introduction

Cloud computing delivers infrastructure, platform, and software as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. These services in industry are respectively referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The importance of these services is highlighted in a recent report from Berkeley as: "Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service" [16]

Clouds [3] aim to resource management and scheduling to user applications for the high performance computing community. So, Virtual machine technology is adopted for high end computing to achieve efficient computing resource usage. Some technical work has reported that virtual machine could be used for scientific applications with tolerable performance punishment [14], [19]. A virtual machine (VM) is a software based machine emulation technique to provide a desirable, on demand computing environments for users. In virtual machine, an effective scheduling strategy is important to meet the desired quality of service parameters from both user and system perspectives. Specifically, we would like to reduce response and wait times for a job, maximize the throughput and utilization of the system, and be fair to all jobs regardless of their size or execution times.

Various scheduling algorithms such as the round-robin, backfilling, and gang scheduling algorithms [20] can be implemented in the virtual machine deployment process. In this paper, we focus on implementing a utilization efficient scheduling algorithm for high performance computing where virtual machines are dynamically provided for executing jobs. With these results, we construct a backfilling gang scheduling system, called BGS, which fills in holes in the Ousterhout scheduling matrix [12] using backfilling. We demonstrate that this combined strategy of backfilling with gang scheduling (BGS) consistently outperforms the other strategies (backfilling and gang scheduling separately) from the perspectives of responsiveness, slowdown, fairness, and utilization.

The rest of this paper is organized as follows. Section 2 describes the related work. In Section 3 presents virtualization technology on cloud system. Section 4 presents scheduling strategies in virtual machine. Section 5 describes modeling the VM allocation. Section 6, we describe our proposed system. Finally section 7 concludes the paper.

2. Related Work

To deal with underutilization in batch queuing systems, backfilling techniques such as EASY are often used. Jobs can jump ahead in the queue if they do not delay the start time of the first job in the queue. Conservative backfilling approaches [8] require that upon a backfill operation, no job in the queue is delayed in order to maintain fairness. A problem with these approaches is their reliance on user estimates of job runtimes which are often incorrect [15]. Several techniques have been proposed to model this runtime in order to tackle this problem [17].

Our work is related to gang scheduling [12], which also departs from batch scheduling by allowing time-sharing of compute resources. In gang scheduling, tasks in a parallel job are

executed during the same synchronized time slices across cluster nodes. This requires distributed synchronized context-switching, which may require significant overhead and thus long time slices, although solutions have been proposed [3]. In this work we simply achieve time-sharing in an uncoordinated and low-overhead manner via VM technology. A second drawback of gang scheduling is memory pressure, i.e., the overhead of swapping to disk [1].

In this paper we present a new scheduling strategy consisting of a combination of gang scheduling together with backfilling approach. Moreover, we show that the performance of gang scheduling is dependent on the selected parameters of the scheduler and that significant gains are achievable. In addition, we address the problem of finding a suitable criterion for measuring the scheduler performance.

3. Virtualization Technology on Cloud System

The increasing availability of VM technologies has enabled the creation of customized environments atop physical infrastructures. The use of VMs in distributed systems brings several benefits such as: (i) server consolidation, allowing workloads of several under-utilized servers to be placed in fewer machines; (ii) the ability to create VMs to run legacy code without interfering in other applications' APIs; (iii) improved security through the creation of sandboxes for running applications with questionable reliability; (iv) dynamic provision of VMs to services, allowing resources to be allocated to applications on the fly; and (v) performance isolation, thus allowing a provider to offer some levels of guarantees and better quality of service to customers' applications.

Virtualization technology has two main concepts, virtual machine and virtual machine monitor (Hypervisor). Virtual Machine (VM) is a software artifact that executes other software in the same manner as the machine for which the software is developed and executed. Virtual Machine Monitor is software that supports multiple virtual machines on the same resource. Existing systems based on virtual machines can manage a cluster of computers by enabling users to create virtual workspaces [7] or virtual clusters [9] atop the actual physical infrastructure. Clusters of nodes managed by a resource allocation system that relies on VM technology. The system responds to job requests by creating collections of VM instances on which to run the jobs. Each VM instance runs on a physical node under the control of a VM monitor that can limit its resource usage. All VM Monitors are in turn under the control of a VM Manager that specifies resource usage constraints for all instances. The VM Manager can also preempt instances, and migrate instances among physical nodes.

VM technology allows for accurate sharing of hardware resources among VM instances while achieving performance isolation. The Xen VM monitor [13] enables CPU-sharing and performance isolation in a way that is low-overhead, accurate, and rapidly adaptable. Furthermore, sharing can be arbitrary. For instance, the Xen Credit CPU scheduler can allow three VM instances to each receive 33.3% of the total CPU resource of a dual-core machine [4]. This allows a multi-core physical node to be considered as an arbitrarily time-shared single core. Virtualization of other resources, such as I/O resources, is more challenging but is an active area of research [11].

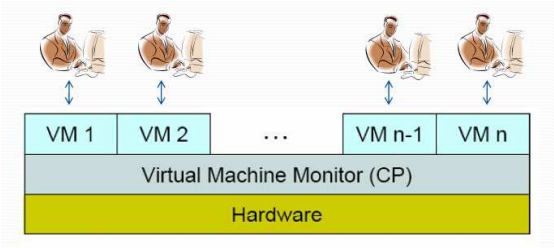

Figure 1.Basic concepts of virtualization technology

4. Scheduling Strategies in Virtual Machine

The following strategies are considered for scheduling user request that arrive at the system.

4.1Backfilling and Gang Scheduling

Backfilling only considered space sharing scheduling strategies. Backfilling can be used with queuing policies such as First come first serve (FCFS), Shortest job first (SJF), Best fit (BFit), Worst fit (WFit). With backfilling, we can bypass the priority order imposed by the policy, as long as the execution of a lower priority job does not delay the start time of higher priority jobs. However, at higher utilizations FCFS performs better than the other policies [18]. A weakness of backfilling is difficult to estimate job execution time and fragmentation may occur.

Another approach gang scheduling, tasks in a parallel job are executed during the same synchronized time slices across cluster nodes. Note that it is possible to schedule some jobs in multiple rows (multiple virtual machines).This can be applied together with any prioritization policy. In particular, we have shown in previous work [10] that gang scheduling is very effective in improving the performance of FCFS policies. This is in

agreement with the results in [18]. We have also shown that gang scheduling is particularly effective in improving system responsiveness, as measured by average job wait time. However, gang scheduling alone is not as effective as backfilling in improving average job response time.

Gang scheduling and backfilling are two optimization techniques that operate on orthogonal axes, space for backfilling and time for gang scheduling. It is tempting to combine both techniques in one scheduling system. In principle this can be done by treating each of the virtual machines created by gang scheduling as a target for backfilling. Backfilling with gang scheduling strategy can compare different scheduling strategies. This strategy will consistently outperforms the other strategies (backfilling and gang scheduling separately) from the perspectives of responsiveness, slowdown, fairness, and utilization.

5. Modeling the VM Allocation

This section shows simple scheduling scenario to clear illustrate the scheduling policy. Schedules for space- and time-sharing of a parallel machine can be represented by an Ousterhout matrix, in which the rows represent time slices and the columns represent processors. In this figure, a host with two CPU cores receives request for hosting two VMs, and each one requiring two cores and running four tasks units: t1, t2, t3 and t4 to be run in VM1, while t5, t6, t7, and t8 to be run in VM2.

In Figure 2(a), a space-shared policy is used for allocating VMs, but a time-shared policy is used for allocating individual task units within VM. Hence, during a VM lifetime, all the tasks assigned to it dynamically context switch until their completion. This allocation policy enables the task units to be scheduled at an earlier time, but significantly affecting the completion time of task units that are ahead the queue.

In Figure 2(b), a time-shared scheduling is used for VMs, and a space-shared one is used for task units. In this case, each VM receives a time slice of each processing core, and then slices are distributed to task units on space-shared basis. As the core is shared, the amount of processing power available to the VM is comparatively lesser than the aforementioned scenarios. As task unit assignment is space-shared, hence only one task can be allocated to each core, while others are queued.

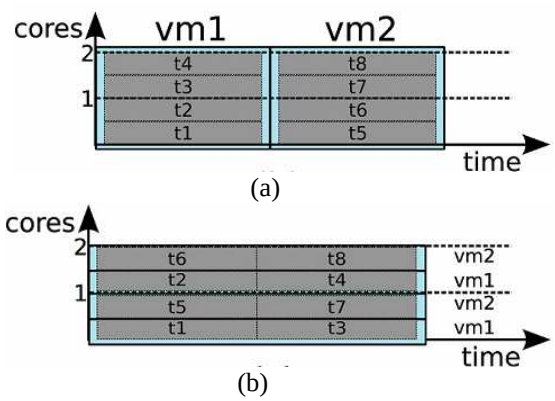

Figure 2. Effects of space and time sharing scheduling policies on task execution: (a) Space-shared for VMs and time-shared for tasks, (b) Time-shared for VMs, space-shared for tasks

6. Proposed System Architecture

This section describes the proposed system for virtual machine scheduling.VM scheduling techniques on cloud environment are power efficient, utilization efficient and cost efficient (VM co-location). This system emphasizes on resource utilization efficient VM scheduling.

The proposed system supports two levels. First, at the VM level and second the host level. At the VM level, the VMs assign specific amount of the available processing power to the individual task units that are hosted within its execution engine. At the host level, it is possible to specify how much of the overall processing power of each core in a host will be assigned to each VM. At each level, this system uses backfilling with gang scheduling policies.

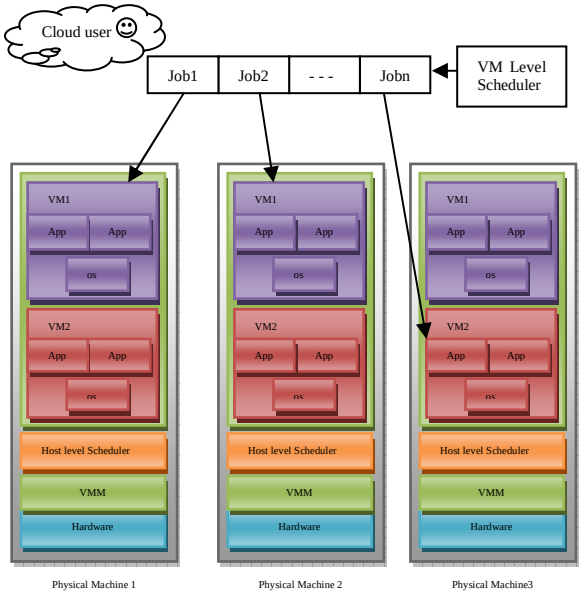

Figure3. System Architecture of the proposed system

7. Conclusion

It has been widely accepted that virtual machines can be employed as computing resources to build a distributed system for high performance computing. So, virtual machine scheduling and resource allocation is essential in cloud computing environment for maximize resource utilization and minimize job's execution time. This paper present an integrated strategy called BackfillingGang Scheduling (BGS), which combines backfilling (on a FCFS job arrival queue) with gang scheduling. We show how this integrated strategy outperforms a system which uses just backfilling or just gang scheduling over sufficient to significantly improve the flow time and increase the machine utilization.

8. References

[1] A. Batat and D. G. Feitelson, "Gang scheduling with memory considerations," in Proc. of the 14th Intl. Parallel and Distributed Processing Symp., 2000, pp. 109–114.

[2] A. Hori, H. Tezuka, and Y. Ishikawa, "Overhead analysis of preemptive gang scheduling," in Proc. of the 4th Workshop on Job Scheduling Strategies for Parallel Processing, ser. LNCS, 1998, vol. 1459, pp. 217–230.

[3] A. Weiss. Computing in the clouds. NetWorker, 11(4):16–25, Dec. 2007.

[4] D. Gupta, L. Cherkasova, and A. Vahdat, "Comparison of the Three CPU Schedulers in Xen," ACM SIGMETRICS Performance Evaluation Review, vol. 35, no. 2, pp. 42–51, 2007.

[5] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and Practice in Parallel Job Scheduling.In IPPS'97 Workshop on Job Scheduling Strategies for Parallel Processing, volume 1291 of Lecture Notes in Computer Science, pages 1–34. Springer-Verlag, April 1997.

[6] D. G. Feitelson and A. M. Weil. Utilization and predictability in scheduling the IBM SP2 with backfilling . In 12th International Parallel Processing Symposium, pages 542–546,April 1998.

[7] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the Grids. Scientic Programming, 13(4):265-275, 2006.

[8] Feitelson, D.G., Weil, A.: Utilization and predictability in scheduling the ibm sp2 with backlling. In: IPPS '98: Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium, Washington, DC, USA, IEEE Computer Society (1998) 542

[9] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual clusters for Grid communities. In 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006), pages 513-520, Washington, DC, USA, May 2006. IEEE Computer Society.

[10] H. Franke, J. Jann, J. E. Moreira, and P. Pattnaik. An Evaluation of Parallel Job Scheduling for ASCI Blue-Pacific . In Proceedings of SC99, Portland, OR, November 1999. IBM Research Report RC21559.

[11] D. Ongaro, A. L. Cox, and S. Rixner, "Scheduling I/O in Virtual Machine Monitors," in Proc. of the ACM SIGPLAN/SIGOPS Intl. Conf. on Virt. Exec. Environments, 2008.

[12] J. K. Ousterhout. Scheduling Techniques for Concurrent Systems. In Third International Conference on Distributed Computing Systems, pages 22–30, 1982.

[13] 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," in Proc. of the 19th ACM Symp.on Operating System Principles, 2003, pp. 164–177.

[14] L. Wang, M. Kunze, and J. Tao, "Performance evaluation of virtual machine based Grid workflow system," Concurrency and Computation: Practice and Experience, vol. 20, no. 15, pp. 1759–1771, 2008.

[15] Mualem, A.W., Feitelson, D.G.: Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. IEEE Transactions on Parallel and Distributed Systems 12 (2001) 529-543.

[16] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

[17] Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel Distrib. Syst. 18(6) (2007) 789-803.

[18] U. Schwiegelshohn and R. Yahyapour. Improving First-Come-First-Serve Job Scheduling by Gang Scheduling. In IPPS'98 Workshop on Job Scheduling Strategies for Parallel Processing,March 1998.

[19] W. Huang, J. Liu, B. Abali, and D. K. Panda, "A case for high performance computing with virtual machines," in Proceedings of the 20th Annual International Conference on Supercomputing, 2006, pp.125–134.

[20] Y. Wiseman and D. Feitelson, "Paired gang scheduling," IEEE Transactions on Parallel and Distributed Systems, vol. 14, no. 6, pp. 581–592,2003.