# Communication-Driven Scheduling for Virtual Clusters in Cloud

Haibao Chen
Serv. Comp. Tech.&Sys. Lab
Cluster & Grid Comput. Lab
School of Computers
Huazhong Univ. of Sci.&Tech.
Wuhan, 430074, China
chenhaibao@hust.edu.cn

Song Wu
Serv. Comp. Tech.&Sys. Lab
Cluster & Grid Comput. Lab
School of Computers
Huazhong Univ. of Sci.&Tech.
Wuhan, 430074, China
wusong@hust.edu.cn

Sheng Di
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439
INRIA, Grenoble, France
sheng.di@inria.fr

Bingbing Zhou
School of Information Tech.
The University of Sydney
NSW 2006, Australia
bing.zhou@sydney.edu.au

Zhenjiang Xie
Serv. Comp. Tech.&Sys. Lab
Cluster & Grid Comput. Lab
School of Computers
Huazhong Univ. of Sci.&Tech.
Wuhan, 430074, China
xiezhenjiang@hust.edu.cn

Hai Jin, and Xuanhua Shi
Serv. Comp. Tech.&Sys. Lab
Cluster & Grid Comput. Lab
School of Computers
Huazhong Univ. of Sci.&Tech.
Wuhan, 430074, China
hjin@hust.edu.cn

## ABSTRACT

Recent research already confirmed the feasibility of running tightly-coupled parallel applications with virtual clusters. However, such types of applications suffer from significant performance degradation, especially as the overcommitment is common in cloud. That is, the number of executable Virtual CPUs (VCPUs) is often larger than that of available Physical CPUs (PCPUs) in the system. The performance degradation mainly results from that the current Virtual Machine Monitors (VMMs) cannot co-schedule (or coordinate at the same time) the VCPUs that host parallel application threads/processes with synchronization requirements.

We introduce a communication-driven scheduling approach for virtual clusters in this paper, which can effectively mitigate the performance degradation of tightly-coupled parallel applications running atop them in overcommitted situation. There are two key contributions. 1) We propose a communication-driven VM scheduling (CVS) algorithm, by which the involved VMM schedulers can autonomously schedule suitable VMs at runtime. 2) We integrate the CVS algorithm into Xen VMM scheduler, and rigorously implement a prototype. We evaluate our design on a real cluster environment, and experiments show that our solution attains better performance for tightly-coupled parallel applications than the state-of-the-art approaches like Credit scheduler of Xen, balance scheduling, and hybrid scheduling.


*HPDC'14,* June 23–27, Vancouver, BC, Canada.

ACM 978-1-4503-2749-7/14/06 ...$15.00.
http://dx.doi.org/10.1145/2600212.2600714.

## Categories and Subject Descriptors

D.4.1 [**Operating System**]: process management—*Scheduling*

## Keywords

Virtual cluster; Scheduling; Cloud computing

## 1. INTRODUCTION

Virtualized cloud datacenter, because of its flexibility and cost-effectiveness, is increasingly being explored as an alternative to local clusters by academic and commercial users to run tightly-coupled parallel applications [1]. However, these users still face the performance degradation problem when running such applications in cloud. This problem is mainly due to the fact that current Virtual Machine Monitors (VMMs) do not simultaneously coordinate/schedule Virtual CPUs (VCPUs) that host threads/processes of parallel applications with synchronization requirements.

Despite some works [2, 3, 4] on scheduling in virtualized environment, the focus has so far been primarily on a *single* symmetric multiprocessing (SMP) VM that runs multi-thread application with synchronisation operations, yet there is no existing research about scheduling on virtual clusters hosting tightly-coupled parallel applications. Moreover, in order to maximize cloud resource utilization, overcommitment is a fairly common phenomenon in cloud. For example, recent research from VMware shows that the average VCPU-to-core ratio is 4:1, based on the analysis of 17 real-world datacenters [5]. Such overcommitted situation aggravates the performance degradation problem of parallel applications running in cloud. This paper targets the challenge of how to efficiently schedule virtual cluster hosting parallel applications in overcommitted cloud environment. We introduce a communication-driven approach for scheduling virtual clusters, which can effectively mitigate the performance degradation of tightly-coupled parallel applications running in overcommitted cloud environment.

The rest of this paper is organized as follows. We explain our design motivation in Section 2 followed by the description of our approach in Section 3. Section 4 presents the performance evaluation results. Finally, we conclude the paper in Section 5.

# 2. DESIGN MOTIVATION

In this section, we analyze the asynchronous scheduling problem and discuss the disadvantage of existing solutions.

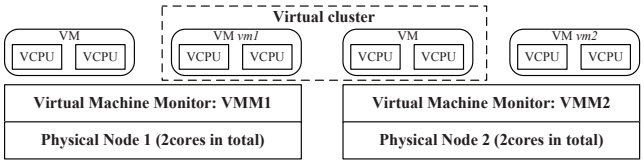



**Figure 1: Virtual cluster deployed among two physical nodes. Each VMM carries out scheduling asynchronously without considering the synchronization requirement of VMs belonging to the virtual cluster.**

Asynchronous VM scheduling method used by VMMs in multi-core physical nodes is simple to implement and beneficial for high-throughput workloads, yet it is inefficient for virtual cluster running parallel applications that require much coordination among tasks, especially in overcommitted environment. We use Figure 1 to illustrate this problem. In this example, a 4-process tightly-coupled parallel application runs on a virtual cluster, which consists of two 2-VCPU VMs (*vm1* and *vm2*) hosted in two different physical machines (*node1* and *node2*). Suppose Xen is used as the VMM, adopting Credit scheduler (a proportional-share scheduling policy). With the Credit scheduler, VCPUs in all run-queues of PCPUs are scheduled asynchronously on each physical machine. This kind of asynchronous scheduling policy usually cannot take over the lock-holder preemption problem [3], which will vastly increase synchronization latency and potentially block the progress of other VCPUs waiting to acquire the same lock.

Recently, most of existing work (e.g., co-scheduling methods of VMware, hybrid scheduling [3], and dynamic coscheduling [4]) on VM-based scheduling is only focused on the performance improvement of concurrent workload processing (multi-thread application with synchronization operations) over SMP VM, instead of the parallel applications on virtual clusters. The problem of these approaches is that all VMs inside a virtual cluster are still scheduled asynchronously from the perspective of virtual cluster, which results in the decreased performance of tightly-coupled parallel application running in virtual cluster. As shown in Figure 1, since *VMM1* and *VMM2* make VM scheduling decisions autonomously, the probability of *vm1* and *vm2* (managed by different VMMs) being scheduled simultaneously is very low. That is, the existing scheduling methods for SMP VMs neglect the synchronization requirement among VMs, belonging to the same virtual cluster.

# 3. OUR APPROACH

We introduce our basic idea in Section 3.1, and propose *communication-driven VM scheduling (CVS)* algorithm and CVS scheduler in this Section 3.2 and 3.3, respectively.

## 3.1 Basic Idea

Based on experiments in virtualized environment, we find that the inter-VM communication (e.g., the number of received packets) can serve as a signal to detect the coordination demands from the viewpoint of VM-level synchronization. That is, VMMs can make VM scheduling decisions based on this signal to satisfy the coordination demands of VMs belonging to the same virtual cluster.

In order to explore the correlation between the number of packets received by VM and the synchronization requirements, we characterize the number of spinlocks (an indicator of synchronization requirement) and the number of packets via experiments. Four 8-VCPU VMs are used to run a set of MPI programs with 32 processes in parallel. We choose three benchmark programs (called *is*, *ep*, and *lu*) from NPB suite of version 2.4, as they exhibit three typical types of parallel executions: communication intensive application with little computation (*is*); CPU intensive application with little communication (*ep*); and the one that lies in between them (*lu*). For each VM, the number of packets and that of spinlocks are recorded every 120 milliseconds (multiplying the 30ms of Xen Credit scheduler by the number of VMs in this experiment) over 60 seconds.

The analysis of experimental results shows that the average Pearson Correlation Coefficients (PCC) between the number of packets and that of spinlocks in tightly-coupled parallel applications (i.e., *lu* and *is*) are 0.89 and 0.97 respectively, while that value in computation-intensive application (i.e., *ep*) is only 0.17. This implies that the inter-VM communication is a fairly good signal to detect potential synchronization requirements.

## 3.2 CVS Algorithm

CVS algorithm helps VMMs select and schedule the VM (running tightly-coupled parallel application) with the largest number of received packets counted from the last de-scheduled moment and promotes its priority. The key idea of CVS is based on such an intuition: when running tightly-coupled parallel application in a virtual cluster, the more packets a VM receives during a scheduling period, the more urgent the synchronization requirement of this VM probably is.

Particularly, according to the information of PCPU runqueue, there are three situations that CVS algorithm needs to deal with. If there is no VM that runs tightly-coupled parallel application in the run-queue of PCPU, CVS algorithm will select the VM at the head of run-queue directly, just like the Credit scheduler of Xen does. If there exists only one VM that runs parallel application in the run-queue of PCPU, CVS algorithm will select that VM without any hesitation. When there are more than one VM, which runs tightly-coupled parallel application in the run-queue of PCPU, CVS algorithm will select the VM that receives the largest number of packets counted from the last de-scheduled moment. Further more, if two VMs receive the same number of packets, their remaining CPU shares (e.g., remaining credit values of Xen) will be used to carry out the VM selection, and the more the remaining CPU shares, the higher the priority. If the VMs still cannot be differentiated, CVS algorithm will pick a VM from among all qualified VMs according to their original orders in the run-queue of PCPU.

After CVS algorithm determines which VM should be scheduled for running parallel application, the scheduler of VMM will schedule all VCPUs of the VM simultaneously

by sending Inter-Processor Interrupt (IPI) signal to the involved PCPUs on the same physical machine.

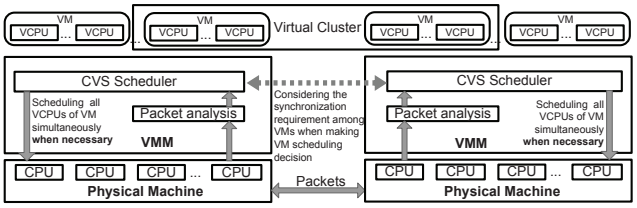



**Figure 2: Overview of CVS scheduler. With CVS algorithm, this scheduler takes the synchronization requirement among VMs belonging to the same cluster into consideration when scheduling VMs.**

## 3.3 CVS Scheduler

Based on CVS algorithm, we design and implement our CVS scheduler by extending the Credit scheduler of Xen 3.2, the overview of which is presented by giving an example as shown in Figure 2.

The CVS scheduler monitors the communication states inside each VMM, and dynamically analyzes the statistics of received packets. The monitored communication state is driven by the running parallel application, and we call it *locally visible synchronization requirement information*. With such information, our CVS scheduler can take the synchronization requirement into consideration when scheduling VMs. Meanwhile, CVS scheduler suffers little overheads, because the coordination information demanded (i.e., the statistics of received packets in VM) is implicitly carried in the communication messages. As for the intra-VM scheduling, all VCPUs of each SMP VM can be scheduled at the same time by sending Inter-Processor-Interrupt (IPI) to involved PCPUs when demanded.

# 4. PERFORMANCE EVALUATION

We first describe our experimental methodology, and then present experimental results in the following sections.

## 4.1 Experimental Methodology

In order to compare our CVS approach to the state-of-the-art scheduling approaches: **CREDIT**: the default scheduler of Xen, **H**ybrid **S**cheduling (HS), and **B**alance **S**cheduling (BS), we devise a test in this section with some restrictions, e.g., using the given configuration (the size, number, and placement) of virtual clusters, which is split into two parts.

## 4.2 Fixed Ratio of VCPU to PCPU

In this experiment, we scale the number of physical nodes (each one is equipped with two Intel Xeon E5345 quad-core CPU) from 2 to 32 (2, 4, 8, 16, and 32), and four 4-VCPU VMs are booted up on each physical node. The fixed VCPU-to-PCPU ratio is 2.5:1. Four identical virtual clusters are built using all VMs in the platform, and the four VMs on each physical node belong to them separately. We run *lu* on these four virtual clusters simultaneously for ten times, and record the execution time of *lu* on each virtual cluster. The same test procedures also go to *is* and *ep*, respectively.

Based on Figure 3(a) and 3(b), it is clearly observed that our CVS approach exhibits the best performance and scalability for *lu* and *is*. The performance and scalability of HS is much better than that of BS. We analyze the key reasons as follows. First, for tightly-coupled parallel applications (e.g., *lu* and *is*), our CVS scheduler outperforms the other three approaches (BS, HS and CREDIT) and scales better because it considers the synchronization requirements of the VMs that belong to the same virtual cluster when making VM scheduling decisions. Second, BS is a probabilistic co-scheduling approach, and the probability of co-scheduling VCPUs of virtual cluster will become lower and lower with increasing number of physical nodes (VMs of virtual cluster). Thus, BS has a slight performance gain over CREDIT when the number of physical nodes is small (e.g., 2), while the performance gain is not clear with large number of nodes. Third, although HS co-schedules all VCPUs of SMP VMs on single physical node, all VMs belonging to the same virtual cluster are still scheduled asynchronously because involved VMMs neglect the synchronization requirements among VMs when making scheduling decision. Therefore, the performance and scalability of HS are between these of CVS and BS.

From Figure 3(c), we can observe that these four approaches have almost the same performance and scalability. The reason is that CVS and HS will gracefully degrade into CREDIT with respect to the CPU intensive applications with little communication (e.g., *ep*).

## 4.3 Varying Ratios of VCPU-to-PCPU

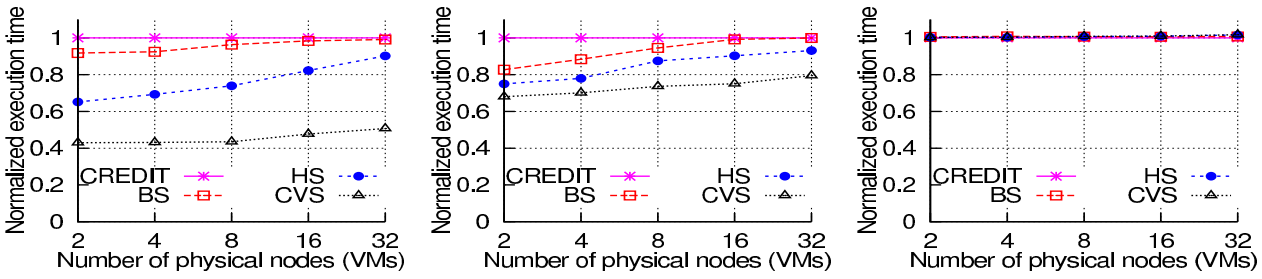
In this experiment, we dynamically adjust the ratio of VCPU-to-PCPU from 2.5 to 4 by changing the number of VMs hosted on each physical node of platform from 4 to 7. As the configuration of virtual clusters in Section 4.2, VMs on each physical node belong to different virtual clusters separately. Figure 4 presents the average execution time of *lu*, *is*, and *ep* when running on virtual clusters with BS, HS and CVS, respectively.

From Figure 4(a) and 4(b), we can easily observe that our CVS approach has the best performance for *lu* and *is* in scenarios with different ratios of VCPU-to-PCPU, and the performance of HS are between these of CVS and BS. Specifically, the performance of CVS and HS become better with increasing ratio of VCPU-to-PCPU, while the performance gain of BS over CREDIT is not obvious at all. The reasons behind these figures are as follows. First, as the ratio of VCPU-to-PCPU increasing, the probability of co-scheduling VCPUs of virtual cluster using BS approach becomes lower and lower. Therefore, the performance gain over CREDIT is not clear in such situation. Second, HS outperforms BS and CREDIT because it can co-schedule the VCPUs of SMP VM that runs tightly-coupled parallel application. However, it is still worse than our CVS approach due to the fact that involved VMMs with HS neglect the synchronization requirements among VMs when making scheduling decision.

From Figure 4(c), we observe that these four approaches have almost the same performance and scalability for *ep* as the ratio of VCPU-to-PCPU changing, which is due to the same reasons for Figure 3(c) in Section 4.2.

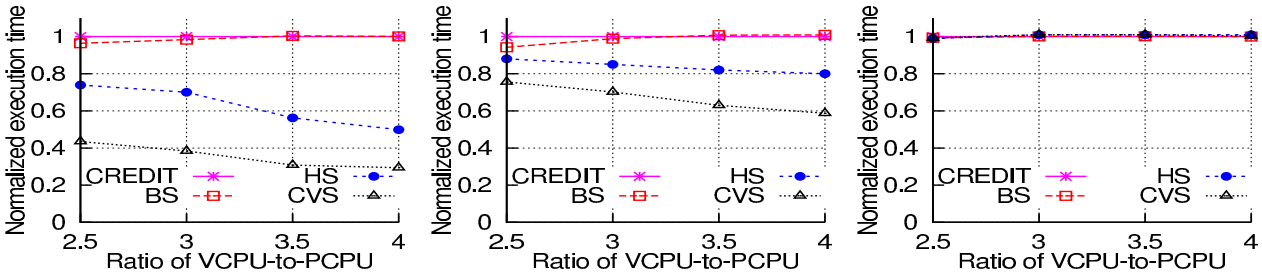
# 5. CONCLUSIONS AND FUTURE WORK

This paper targets the challenge of how to schedule virtual clusters hosting tightly-coupled parallel applications and mitigate performance degradation in overcommitted cloud environment. We introduce a communication-driven VM schedul-




(a) *lu* (b) *is* (c) *ep*

**Figure 3: Performance comparison of approaches (CREDIT, BS, HS, and CVS) with fixed ratio of VCPU to PCPU when running benchmarks on 2, 4, 8, 16, and 32 nodes (VMs).**




(a) *lu* (b) *is* (c) *ep*

**Figure 4: Performance comparison of approaches (CREDIT, BS, HS, and CVS) when running benchmarks with different ratios of VCPU-to-PCPU.**

ing (CVS) approach of virtual clusters. This approach is simple to apply in practice. Meanwhile, it allows participating VMMs to act autonomously, thus retaining the independence of VMMs. For tightly-coupled parallel application, our CVS approach improves the application performance significantly in comparison to the state-of-the-art approaches.

Unrestricted simultaneous scheduling of all VCPUs for SMP VM through sending IPIs may cause excessive numbers of preemptions while repeatedly interrupting other VMs, which results in serious performance degradation [6, 7]. To mitigate this problem with unexpected preemptions, we will devise a VM preemption mechanism to enhance our CVS approach. The prerequisite of all co-scheduling algorithms is to know the type of workload running in VMs. That is, the scheduler must understand whether the workload is parallel application or not. Actually, there are at least two alternative ways to do so. The first one is to adopt the inference techniques using gray-box knowledge. The other is to obtain the type information of workload directly from end users, similar to the work in [3, 8]. In this paper we adopt the latter for simplicity. In the future, we will evaluate the former one for comparison.

## 6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by National Science Foundation of China under grant 61232008, National 863 Hi-Tech Research and Development Program under grant 2013AA01A213 and 2013AA01A208, Guangzhou Science and Technology Program under grant 2012Y2-00040, Chinese Universities Scientific Fund under grant 2013TS094, Research Fund for the Doctoral Program of MOE under grant 20110142130005.

## 7. REFERENCES

[1] Thomas J Hacker and Kanak Mahadik. *Magellan Final Report*. U.S. Department of Energy (DOE), 2011.

[2] Orathai Sukwong and Hyong S Kim. Is co-scheduling too expensive for smp vms? In *Proc. EuroSys*, pages 257–272. ACM, 2011.

[3] Chuliang Weng, Zhigang Wang, Minglu Li, and Xinda Lu. The hybrid scheduling framework for virtual machine systems. In *Proc. VEE*, pages 111–120. ACM, 2009.

[4] Chuliang Weng, Qian Liu, Lei Yu, and Minglu Li. Dynamic adaptive scheduling for virtual machines. In *Proc. HPDC*, pages 239–250, 2011.

[5] Vijayaraghavan Soundararajan and Jennifer M Anderson. The impact of management operations on the virtualized datacenter. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 326–337. ACM, 2010.

[6] Hwanju Kim, Hyeontaek Lim, Jinkyu Jeong, Heeseung Jo, and Joonwon Lee. Task-aware virtual machine scheduling for i/o performance. In *Proc. VEE*, pages 101–110. ACM, 2009.

[7] Hwanju Kim, Sangwook Kim, Jinkyu Jeong, Joonwon Lee, and Seungryoul Maeng. Demand-based coordinated scheduling for smp vms. In *Proc. ASPLOS*, pages 369–380. ACM, 2013.

[8] Cong Xu, Sahan Gamage, Pawan N Rao, Ardalan Kangarlou, Ramana Rao Kompella, and Dongyan Xu. vslicer: latency-aware virtual machine scheduling via differentiated-frequency cpu slicing. In *Proc. HPDC*, pages 3–14. ACM, 2012.