*v*Testkit: A Performance Benchmarking Framework for Virtualization Environments

Kejiang Ye, Jianhua Che, Xiaohong Jiang, Jianhai Chen, Xing Li

College of Computer Science, Zhejiang University
Hangzhou 310027, China
{yekejiang,chejianhua,jiangxh,chenjh919,lix}@zju.edu.cn

***Abstract*—Virtualization technology has attracted wide attention in recent years as a method to improve resource utilization, reduce costs, and ease server management. However, the performance penalty resulting from virtualization is an unneglectable problem and should be carefully evaluated. To our knowledge, there are few performance evaluating tools developed for virtualization environments. We propose a configurable framework and implement a prototype vTestkit to provide a platform to do performance evaluation for virtualization environments easily, flexibly, and automatically. In this paper, we first discuss the requirements and challenges of performance measurement in virtualization environments, and then present a methodology for characterizing the performance of single virtual machine (VM) scenario and multi-VM scenario. Then we introduce the architecture of vTestkit framework, implement details, and the testing process with vTestkit. Finally, three typical case studies are presented to show that *v*Testkit can meet the complex testing requirements well and is propitious to various scenarios.**

***Keywords*-Virtualization; Performance; Monitoring; Server Consolidation**

I. Introduction

Virtualization technology has of late attracted wide attention as an approach to improve resource utilization, reduce costs, and ease server management. It provides an abstraction of hardware resources enabling multiple instantiations of operating systems (OSes) to run simultaneously on a single physical machine. Besides, virtualization plays an important role in cloud computing [1] to provide on-demand computing capability. Currently, there are several virtualization solutions such as VMware [2] and KVM [3] for full virtualization, Xen [4] for both para-virtualization and full virtualization, OpenVZ [5] for OS-level virtualization, etc.

Due to the introduction of Virtual Machine Monitor (VMM), the performance of virtualization environments is highly concerned by academia and industry community. A lot of focuses are fixed on the performance overheads of pivotal system components in virtual machines, e.g., CPU, memory, disk and network etc [6, 7]. However, with the performance testing requirements for these components getting more complex, the traditional manual testing methods do not comply with testing requirements. And the parameters of different benchmarks are varied and not easy to be adjusted manually. What's more, automated testing, including quick configuration and automatic deployment, is essential for facilitating performance evaluation, and also useful for large-scale deployment and real-time monitoring in virtualization environments. Besides, real-time monitoring of the system resources during the whole testing is also an important requirement. In this paper, we present a configurable framework and implement a prototype called *v*Testkit for performance benchmarking in virtualization environments. *v*Testkit supports the following functions: 1) flexible benchmark type change, 2) easy benchmark parameter adjusting, 3) quick test plan deployment, 4) automatic test processing, 5) parallel testing, and 6) real-time resource utilization monitoring in both single-VM and multi-VM scenario. In addition, it also supports storing and loading the test template for reuse in future tests.

The rest of this paper is organized as follows. We first describe the problem statement and introduce two existing tools of performance measurement for virtualization environments in Section 2. And then in Section 3, we present the architecture of the *v*Testkit framework and describe its testing process. In Section 4, we discuss the design and implementation issues. Section 5 shows three case studies. The related work is introduced in Section 6 and we conclude with the future work in Section 7.

II. Problem Statement and Background

In this section, we describe the problem statement and requirements that we have identified for performance benchmarking framework to assess various virtualization testing scenarios, especially the complex scenarios such as server consolidation and parallel testing in virtual cluster. After that, we introduce two existing tools for server consolidation: VMmark and *v*Consolidate.

A. Problem Statement

Although virtualization technology holds a lot of benefits, it incurs some performance loss to a certain extent. To use this emerging technology better, it becomes crucial to measure various virtualization overheads and identify potential performance bottlenecks. However, there is a lack of performance benchmarking tools for evaluating the performance

978-0-7695-4106-8/10 $26.00 © 2010 IEEE
DOI 10.1109/ChinaGrid.2010.12

of various virtualization environments. We summarize the deficiencies of traditional manual testing methods as follows:

- Complex configuration process. Currently, there are many benchmarks developed by different organizations or individuals for performance evaluation. And the benchmark parameters are rich and various. When users want to test the CPU performance of virtual machine, they need to spend a lot of time studying different benchmarks (SPEC CPU, linpack, sysbench, etc) and related parameters. Only after that, can they run the benchmarks.
- Inefficient testing process. The users need to download the different types of benchmarks first, then install the benchmarks in VMs. Only after that can they start to do test by running the benchmarks. Obviously, this is an inefficient way.
- Lack of support for parallel testing. It's difficult to start multiple tests or to run benchmarks simultaneously in multi-VM scenario in manual control way.

Based on the above analysis, development of an innovative performance benchmarking framework for virtualization environments is really necessary, which can simplify the configuration process, automate the test process, and enable parallel testing.

B. Requirements for Virtualization Performance Evaluation

To evaluate the performance of virtualization environments especially in the complex multi-VM scenario with various demands, a performance benchmarking framework should satisfy three main requirements:

1) The framework should support a whole automatic testing process, including creating test plans, generating test tasks, deploying test tasks onto target machines (virtual machines or physical machines), controlling the running process, and returning the test reports. All configuration parameters, such as benchmark types and benchmark parameters should be set up or changed flexibly. All information of a test plan, including test purpose, VM configuration, and benchmark suits with their parameters, can be stored and loaded for future use.
2) The framework should also bear complex testing requirements, such as parallel testing that running parallel benchmarks in virtual machines via MPI communication. Three scenarios often occur in virtualization environments: (i) Component testing. In general, it is required to test a specific component of a virtual machine with varies benchmarks and parameters to observe how the performance changes correspondingly. This calls for the framework to support flexible adjustment of benchmarks and parameters. (ii) Server consolidation. It's common to analyze the resource requirements for servers respectively and what is the optimal consolidation for given servers in server consolidation scenario. (iii) Parallel testing. When running parallel applications in virtualized environment, we should maintain the stability of the entire cluster and generate usable results.
3) Real-time monitoring of the resource utilization of virtual machines is another concernful requirement, e.g. the utilization of CPU, memory, disk I/O and network bandwidth and so forth. It's also expected to find performance bottleneck via investigating the resource utilization.

C. VMmark and vConsolidate

To our knowledge, there are two existing benchmarking tools for performance evaluation of server consolidation, namely VMmark and *v*Consolidate. VMmark [8] is a free tool developed by VMware to measure the scalability and individual performance of several typical workloads running simultaneously in virtualization environments, i.e., mail server, java server, web server, database server, file server and standby server. Each workload is called a "tile" and the final performance score of the whole consolidated system is determined by the tile number. *v*Consolidate [9] is another tool to measure the performance of server consolidation. The workloads of *v*Consolidate are similar to VMmark, i.e., database server, web server, java server, mail server, and another extra idle virtual machine without any workload. Only a little difference exists between the workloads running in VMmark and *v*Consolidate.

Obviously, VMmark and *v*Consolidate can only be used for performance measurement of server consolidation. Neither benchmark types nor benchmark parameters can be changed according to specific test requirements. What's more, the two above consolidation benchmarks cannot be used for other test scenarios, such as HPC performance testing. And the lack of real-time monitoring is also a limitation.

III. The *v*Testkit Framework

In this section we first describe the system architecture and testing process of the *v*Testkit framework, and then explore the implementation issues of this framework.

A. System Architecture

*v*Testkit is designed to test the performance of virtualization environments in which users can customize benchmark types and benchmark parameters under different requirements. The architecture of *v*Testkit is illustrated in Figure 1. It consists of *Management Center*, *Configuration Model*, and *Monitoring Model*.

The *Management Center* subsystem is the key component of this framework that is responsible for managing the whole testing process such as creating testing plan, generating testing tasks, deploying the testing tasks to the target machines,

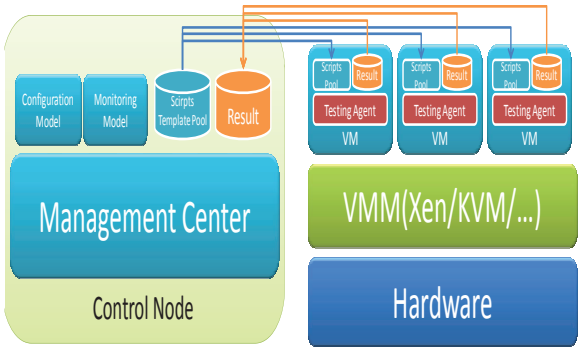


Figure 1. The Architecture of *v*Testkit

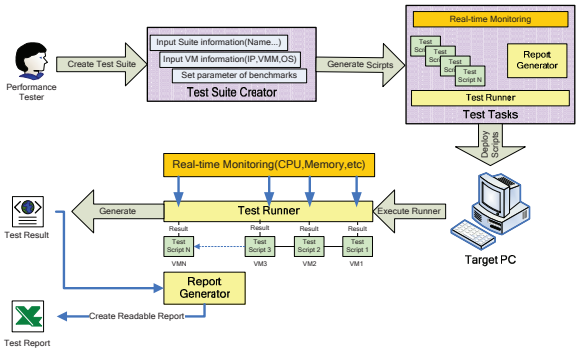


Figure 2. The Test Process of *v*Testkit from user perspective

controlling the running process, and collecting the testing reports from target machines.

The *Configuration Model* subsystem is an easy-to-use component that contains three sub-modules. In *TestPlan* sub-module, users can create a test plan, including the information like the number of virtual machines needs to be created. In *VM Base Information* sub-module, users can define the basic VM information, such as VMM types, Guest OS, VM IP, etc. In *Test Item Information* sub-model, users can customize the testing components, benchmark types, and their parameters.

The *Monitoring Model* subsystem is responsible for monitoring the virtual machine resource utilization when running a benchmark. Through this model, users can monitor the status of each virtual machine. Currently, the framework supports for monitoring the utilization of CPU, memory, and network.

B. *vTestkit Testing Process*

Figure 2 shows the whole testing process from user's perspective. When users begin to test, they should firstly create a test plan, which involves several virtual machines. Each virtual machine is installed with specific benchmarks. When the test plan is created, all test scripts will be generated and sent to the target machines, and the test runner can start to execute the benchmarks. Finally, the test reports will be generated by the report generator. In the whole testing process, the real-time monitor will monitor the utilization of system resource.

We show the process flow of a typical testing using the *v*Testkit framework in detail from the system architect's perspective in Figure 3 (The numbers below correspond to the numbers in the figure):

① Firstly, *Management Center* calls the *Configuration Model* to create a testing plan, and configure the information of virtual machines, and then define the testing components, benchmark types and benchmark parameters.

② After configuration, all information about testing plan, benchmarks and virtual machines will be stored into the *Test Template Pool* so as to be reused in future tests.

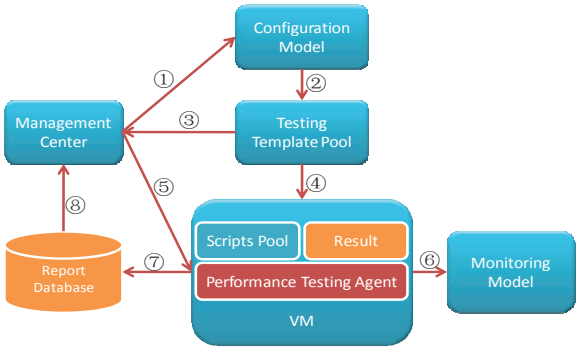


Figure 3. The Test Process of *v*Testkit from system perspective

③ Users can also load an existing testing plan from *Test Template Pool*, and change the parameters according to new requirements, which saves time and improves the testing efficiency.

④ The test templates are deployed to the target virtual machines, and stored into the *Script Pool* at the same time.

⑤ The *Management Center* sends an instruction to the *Performance Test Agent* to start the execution process.

⑥ In the execution process, users can monitor the performance of target virtual machines through *Monitoring Model*.

⑦ After finishing the execution process, a test report will be generated and stored into the *Result Database*.

⑧ The *Management Center* can fetch and view the test reports whenever needed.

IV. Design and Implementation of *v*Testkit

In this section, we discuss the key issues of design and implementation of *v*Testkit. We also discuss the design issues of agent and communication mechanisms between virtual machines in *v*Testkit.

A. *Test Templates*

Normally, it is a heavy workload to configure the virtual machine and benchmark manually. In order to improve the efficiency, we adopt a template-based technique to simplify the configuration of VMs via using test templates.

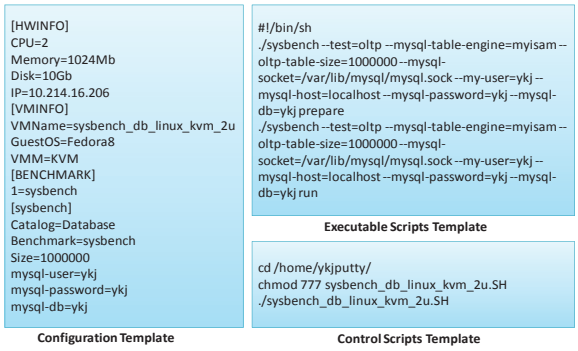


Figure 4. The Test Templates Designed for *v*Testkit

There are three kinds of templates: configuration template, benchmark template, and control script template. The *VM related configuration templates* includes information of VCPU, memory, disk, IP address, VMM and guest OS, and are saved in the template pool that can be loaded in the future. However, these templates can not be executed in the target machines directly. Only when the *benchmark templates* encapsulated with the benchmark types and parameters become the executable scripts, can they be sent to the target machines for running. The *control script template* controls the running time of the executable scripts.

Figure 4 shows the example templates created for testing the database performance in KVM virtual machine. The *Configuration Template* contains the configuration information including information of hardware, virtual machine, and benchmarks. While the *Executable Script* is sent to the target machine it can start to run controlled by the *Control Script*.

B. Agent and Communication Mechanism

We use agent-based communication mechanism to enable communication among VM and *Management Center* of *v*Testkit. There are two kinds of agents in *v*Testkit. The first agent is called "testing agent" responsible for listening and running the test scripts, collecting test results, and returning test results to *Management Center*. The second kind is called "monitoring agent" that is designed to monitor the utilization of system resources including CPU, memory, and network. Through this kind of agent, we can monitor the status of each virtual machine.

When the testing process starts, a secure connection is established between *v*Testkit *Management Center* and target virtual machines. Through this secure connection, users can deploy the test scripts to the target virtual machines, control the running process, and collect test reports. We avoid entering the password by sending the public key to the target virtual machine before establishing the secure connection.

C. Prototype Implementation

Figure 5 shows the main GUI of *v*Testkit. The prototype was implemented under Windows Visual Studio in C++. With this tool, a user can create new testing plan or import existing testing plan, deploy testing tasks to target virtual machines, control the running of testing tasks, and generate testing reports. Also, users can monitor the resources utilization of virtual machines in real time.

D. Extensibility Analysis

*v*Testkit is an extensible and easy-to-use framework. It can adapt to new benchmarks well with no or little change to our framework since we define an unified format for benchmarks. So far, the framework can support several benchmarks including SPEC CPU, linpack for CPU testing, bonnie++, iozone for disk I/O measurement, lmbench for memory measurement, webbench for web server measurement, sysbench for database server measurement, SPEC JBB for java server measurement. When users want to add new benchmarks, they can simply create a new benchmark item according to the defined format. What's more, the framework can support several kinds of virtual machine monitors (VMMs) such as Xen, KVM, OpenVZ. However, the framework doesn't support windows virtual machine environment at present.

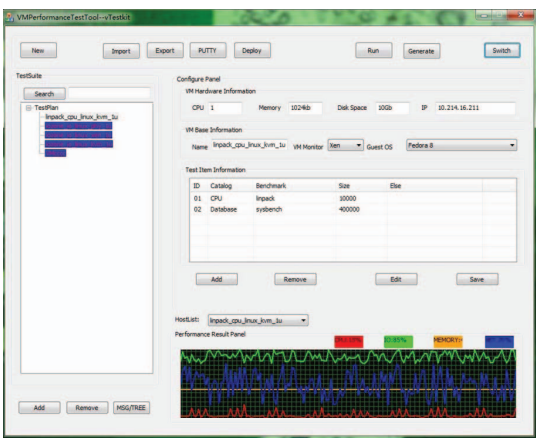
V. Case Studies

In this section, we show three typical cases for evaluating the performance of virtualization environments with *v*Testkit. In the first case, we present a component testing scenario comparing the CPU performance of physical machine, Xen virtual machine and KVM virtual machine using SPEC CPU 2006. In the second case, we present a server consolidation scenario comparing the application performance running in dedicated and consolidated mode. In the third case, we test the performance of virtual cluster with HPCC benchmark running in parallel.

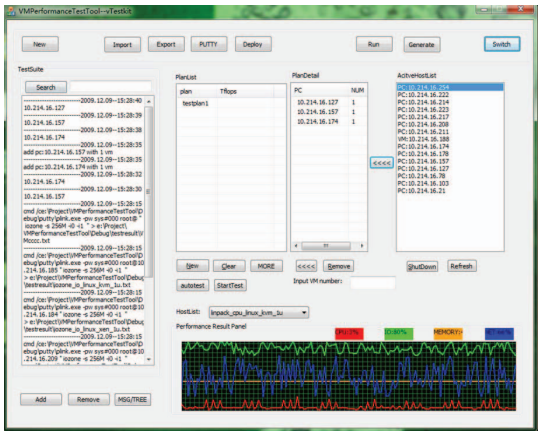
A. Component Testing

In this experiment, we measure the CPU performance overheads in virtualization environment using *v*Testkit. We choose Xen as a para-virtualization solution and KVM as a full virtualization solution, and SPEC CPU 2006 as the benchmark. We create two virtual machines onto two physical machines with one para-virtualized VM and the other full virtualized VM. We also start a third idle physical machine as the comparing baseline. All the three machines are configured with the same benchmark and parameters.

Figure 6 and Figure 7 illustrate the CPU performance of physical machine, Xen virtual machine, and KVM virtual machine. Obviously, physical machine obtains the best performance, while Xen the second and KVM the third. It demonstrates that virtualization will lead to some performance loss but not very obvious. The reason is that full virtualization totally simulates the underlying hardware and all virtualization-sensitive operations for virtual machines, and these sensitive operations must be validated by the



(a) The GUI of vTestkit for Component Testing



(b) The GUI of vTestkit for Parallel Testing

Figure 5. The Prototype of *v*Testkit

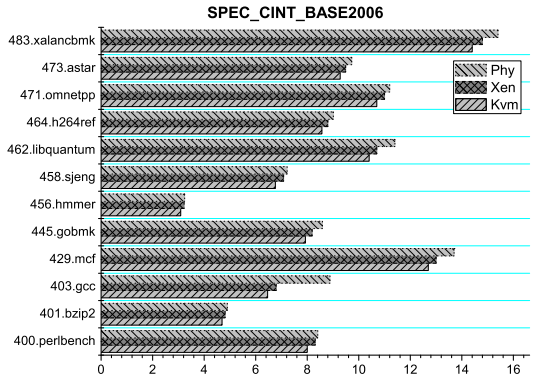


Figure 6. The result of CINT2006 in physical, Xen, and KVM environments

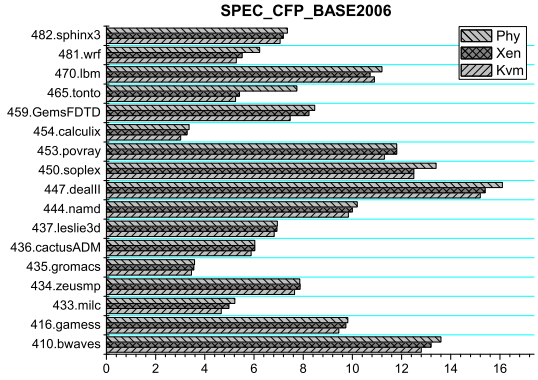


Figure 7. The result of CFP2006 in physical, Xen, and KVM environments

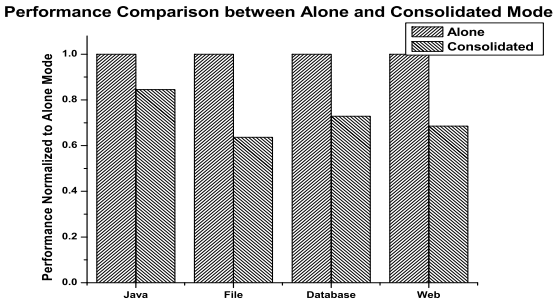


Figure 8. Sever Performance Degradation After Consolidation

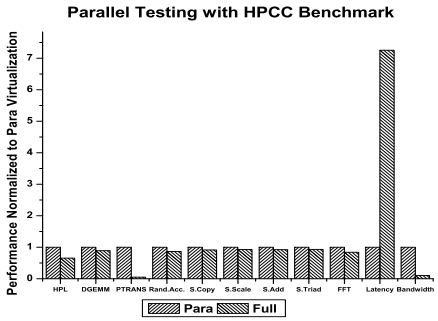


Figure 9. Evaluate the Performance of Para-virtualization and Full Virtualization with HPCC Benchmark

hypervisor that results in some performance penalty. However the para-virtualization technology provides the corresponding fast handler for these sensitive instructions in each virtual machine by the hypervisor in order to obtain higher performance by avoiding frequent traps into the hypervisor.

B. *Server Consolidation*

Server consolidation is a very important function of virtualization that improves resource utilization and reduces cost. In this experiment, we will study server performance after consolidated into a single machine. We create 5 virtual ma-

chines above 5 isomorphic physical machines using *v*Testkit. The first four virtual machines are deployed with java server, file server, database server, and web server respectively, while the last virtual machine is deployed with the four servers running together in a consolidated mode. We select SPEC JBB as a java server workload, IOzone a file server workload, Sysbench a database workload, and WebBench a web server workload.

The result of this experiment is presented in Figure 8. In the consolidation mode, all the servers want to contend the resources and there is also interference with each other. For this reason, each server has a certain degree of performance degradation after consolidation, with java server losing 15.5%, file server 36.36%, database server 27.17%, and web server 31.48% respectively.

C. Parallel Testing

In this experiment, we evaluate the performance overheads in the HPC scenario. We create a 8-node virtual cluster that running HPCC benchmark. We choose MPICH 2.1.0.8 as our MPI environment. We allocate each virtual machine with 4 VCPUs and 256MB memory. Figure 9 shows the test results of para-virtualized and full virtualized cluster. Obviously, in the PTRANS testing, full virtualization achieves very poor performance because PTRANS measures the transfer rate of large arrays of data from multiprocessor's memory and leads to heavy communication overhead. What's more, we also find that the full virtualized cluster has a significant latency and low bandwidth because of the communication overheads. Through such testing case, we can identify the performance bottleneck in the high performance computing environment.

VI. Related Work

A lot of work have been done to evaluate the performance of various virtualization environments [2, 4, 6, 7, 10, 11]. Barham et al. [4] gave a comprehensive introduction to the Xen hypervisor and made a thorough performance evaluation of Xen against VMware ESX Server, UML, Connectix's Virtual PC and Plex86 with SPEC CPU2000, OSDB, dbench and SPEC Web2005. Clark et al. [10] reproduced the results described in [4] with almost identical hardware, and compared Xen with native Linux on a less powerful PC, and evaluated the ability of Xen as a platform for virtual web hosting. Padala et al. [11] extended the evaluation by including OpenVZ for server consolidation. Che et al. [7] studied an initial comparison of Xen and KVM with Linpack, LMbench and IOzone. Recently, Deshane [6] presented an independent work about performance comparison between Xen and KVM, which measured the overall performance, performance isolation and scalability of Xen and KVM. It's mostly noted that VMware and XenSource made their respective experiments to measure and compare the performance of VMware ESX Server and Xen with SPECcpu2000, PassMark, Netperf, SPECjbb2005 as workloads [2].

With respect to the benchmarks and tools for performance evaluating of virtualization systems, Hai Jin et al. [12] presented VSCBenchmark to evaluate the dynamic performance of virtualization systems in server consolidation, which holds the merit that simplifies the construction and configuration of testing scenarios to the utmost extent. Menon et al. [13] described a system-wide statistical profiling toolkit-Xenoprof and its performance measuring and debugging on Xen with network applications. VMmark [8] and *v*Consolidate [9] are two open source tools to measure the server consolidate performance with several typical server applications, and provide an accordant methodology to compare different virtualization systems. Most recently, several performance analysis frameworks [14] and monitoring tools [15, 16] were developed for cloud environments. However, they focus on the performance monitoring, and cannot support the whole testing process automatically.

VII. Conclusion and Future Work

In this paper, we have presented *v*Testkit, an extensible and easy-to-use framework prototype for performance evaluation of virtualization environments. It is extensible in sense since we have defined a unified benchmark configuration template and can easily add new benchmarks to this framework without any change. It is easy-to-use in sense because it supports to configure the benchmark types and benchmark parameters flexibly, deploy the testing tasks automatically, start multiple testing processes concurrently, and monitor their real-time performance. It also supports complex testing scenarios like parallel testing and server consolidation. We also present three case studies to show the effectiveness of *v*Testkit.

In the future, we will add profiling model for *v*Testkit to measure micro performance by monitoring the hardware events such as CPU cycle, L2 cache miss and TLB miss, and identify the performance bottlenecks of multi-VM environments. Additionally, we plan to extend it to other distributed environments such as grid and cloud environment.

Acknowledgment

This work is funded by the National 973 Basic Research Program of China under grant NO.2007CB310900 and National Natural Science Foundation of China under grant NO. 60970125.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "Above the clouds: A berkeley view of cloud computing," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[2] C. A. Waldspurger, "Memory resource management in vmware esx server," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 181–194, 2002.

[3] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," in *Linux Symposium*, 2007.

[4] 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," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[5] "Openvz: Server virtualization open source project," *http://openvz.org/*, 2009.

[6] T. Deshane, Z. Shepherd, J. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao, "Quantitative comparison of Xen and KVM," *Xen Summit, Boston, MA, USA*, pp. 1–2, 2008.

[7] J. Che, Q. He, Q. Gao, and D. Huang, "Performance Measuring and Comparing of Virtual Machine Monitors," in *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008. EUC'08*, vol. 2, 2008.

[8] V. Makhija, B. Herndon, P. Smith, L. Roderick, E. Zamost, and J. Anderson, "VMmark: A scalable benchmark for virtualized systems," Tech. Rep.

[9] J. Casazza, M. Greenfield, and K. Shi, "Redefining server performance characterization for virtualization benchmarking," *Intel Technology Journal*, vol. 10, no. 3, pp. 243–251, 2006.

[10] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews, "Xen and the art of repeated research," *USENIX annual Technical Conference*, pp. 135–144, 2004.

[11] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. Shin, "Performance evaluation of virtualization technologies for server consolidation," *HP Laboratories Technical Report*, 2007.

[12] H. Jin, W. Cao, P. Yuan, and X. Xie, "VSCBenchmark: benchmark for dynamic server performance of virtualization technology," in *Proceedings of the 1st international forum on Next-generation multicore/manycore technologies*, 2008, p. 5.

[13] A. Menon, J. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel, "Diagnosing performance overheads in the xen virtual machine environment," in *VEE: Proceedings of the 1st ACM Conference on Virtual Execution Environments*, 2005, pp. 13–23.

[14] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann, "C-meter: A framework for performance analysis of computing clouds," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid-Volume 00*, 2009, pp. 472–477.

[15] L. Litty, H. Lagar-Cavilla, and D. Lie, "Computer Meteorology: Monitoring Compute Clouds," in *Proceedings of the 12th Workshop on Hot Topics in Operating Systems*, 2009.

[16] J. Boulon, A. Konwinski, R. Qi, A. Rabkin, E. Yang, and M. Yang, "Chukwa, a large-scale monitoring system," in *Proceedings of Cloud Computing and Its Applications*, 2008.