2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing

DCSim:Design analysis on Virtualization Data Center

Chia-Jung Chen and Yi-Sheng Liu and Rong-Guey Chang
Department of Computer Science
Advanced Institute of Manufacturing with High-TECH Innovations
National Chung Cheng University, Chiayi, Taiwan, R.O.C
ccj98p@cs.ccu.edu.tw, lys100m@cs.ccu.edu.tw, rgchang@cs.ccu.edu.tw

***Abstract*—Virtualization has become a core technique to support a data center, which usually consists of hardware, virtual machine(VM), and operating system(OS) etc. With virtualization, users can make good use of resources provided in a data center with resource protection and efficient hardware utilization. However, as data amount has increased very rapidly, many issues such as low power and resource sharing have become very critical. To resolved these issues, this paper presents a novel simulation environment based on a tiny OS, Hard Disk Drive (HDD), and Solid State Drive(SSD) to help users and designers understand a data center via simulation and analysis. Compared with the previous work, our tool can simulate and analyze homogeneous and heterogeneous systems with a multi-layer approach.**

Although each part of a data center has been studied by researchers, the analysis for data centers with multiple layers or heterogeneous devices is still an important issue. If programmers cannot analyze a data center, they are not easy to handle resources efficiently or understand cross-layer influence. They may encounter unpredictable challenges arising from the integration of various components.

To address these challenges, programmer and system designer will be need thorough understanding of multi-layer heterogenous system. We propose a new simulation tool called Data Center Simulator(DCSim). We have used small-scale os and storage simulation to achieve multi-layer system simulation to help us to understand multi-layer heterogenous system. Therefore, programmer can simulate and analysis the system that we will analysis the parallel program and different storage deices on DCSim.

I. INTRODUCTION

Data center has began to focus several issues on performance and resource efficiency, such as resource guarantee. Virtualization will be one of the critical technique of system, because resource protection and hardware utilization. The common virtualization tool is Xen[1]. However, it also bring the complexity and system influence which are difficult to solve without detailed system analysis. Due to facing challenges, the full testing which help to analyze the system must be in progress constantly. Not to mention how to test the optimization of single layer will cause any chain reaction on the other layer. In order to help system analysis, they usually use all kinds of system profiling tool. But, most tool are already not suitable for virtualization data center, because the VM hardware protection and multi-layer system isolation. It causes the lack of system information that we can not make a detailed experimental.

Without profiling tool, they can not early face the challenge of power consumption and reliability. Figure 1 is shown the multi-layer architecture. Each layer has lots of research, such as VM layer[1, 2]. Although they had described in detail, but it is still difficult to understand. Not only VM , but also os is combination of multiple fields techniques. As we know, VM had control all hardware device through virtual machine management(VMM). The system will be segregation of each OS's security. But, the first challenge be caused by VM is some application which need resource guarantee can not provide some guarantee of computing priority. Second challenge is also bring the another problem is overhead. Nowadays, VM had stressed less overhead, but we still need to know the relation of multi-layer system.

Third challenge is large data storage. Different hardware have different feature, like OS had design new mechanism for SSD ,but they not consider the price of SSD. In data center, it need a lot of storage capacity and evaluate the price of SSD and Hard Disk Drives(HDD) Thus, we must use them both to cost down, but the storage mechanism is not design for hybrid device. The file placement will be a chance to proper use the hybrid devices.

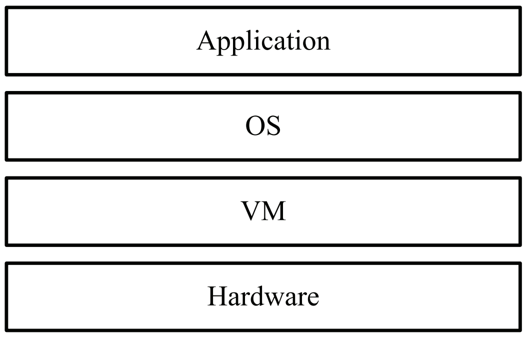

Figure 1. Data Center Architecture

How to understand system influence on a data center is a complicated issue. In general, although OS has a wide variety of optimization techniques, however these optimizations may result in failure running on VM. VM will change the system behavior dynamically because it has protected by the hardware related information, such as power consumption [3] or performance threshold. Roughly speaking, for a cloud or VM, we cannot obtain the hardware-related information, but we may consider the comprehensive factors including performance, power and system information to design a reliable system. Therefore, we have implemented DCSim to simulate the complex behavior for a data center environment, as shown in Figure 2. DCSim is running on linux kernel and use linux POSIX thread to provide services, which can be divided into three layers: virtual hardware (VH), virtual simulator (VS), and virtual OS (VOS). With DCSim, we can verify the behavior of a program and then understand the changes between different layers. DCSim is a system-level emulator composed of small operating system, IO control ring, SSD/HDD (Solid state drive) / (Hard disk drive) simulator, and multi-level monitor. Multi-layer monitor can analyze the system to improve performance based on VS, VOS, and VH. The analysis information can be used to compare VM with Non-VM systems and replace different devices so that we can understand a multi-layer system. We also use cache and memory to verify DCSim for vm challenges with a profiling emulator.

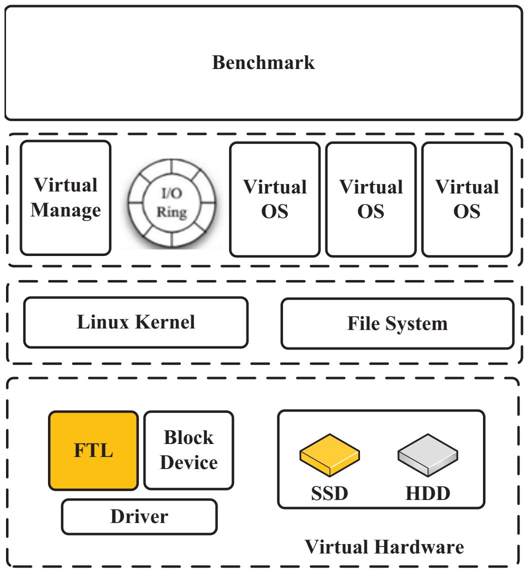

Figure 2. DCSim System Architecture

The rest of paper is organized as follows. Section II is the related work. We first present the hardware support of the novel cache architecture in Section III and then describe the proposed approach in Section IV. The result is shown in Section V and finally we conclude this paper in Section VII.

II. Related work

In the past years, virtualization has become an important issue because modern computers are powerful to use it to improve hardware utilization. In this paper we present a system behavior and mixed hardware virtual machine called DCSim to compare with others in Table I. Although each other simulator has its own feature or functions, they can be classified to four parts: Cloud, Non-OS, System, and Device simulator. The Cloud simulator usually simulates a cloud application on a network environment with a large number of nodes. Thus, they do not take hardware issues into account to simulate an application in detail because the most important thing is to observe network and large node operation [4].

The Non-OS simulator is often designed for a special purpose. For example, SimpleScalar is a simulator for MIPS processors to let users test applications based on their designs, which is similar to a device simulator [7]. For a system designer or server managers, understanding all information of a system is very important. Thus some full system simulators have been designed and developed such as QEMU[2], SimOS[6] and Xen[9], which are executed on a general-purpose operating system. Xen is a virtual machine monitor (VMM) to provide many services and protect instructions and hardware. All guest operating systems are executed like a user-level thread and communicate with each other via inter process communication. In fact, VMM is very similar to micro kernel and thus both of them have some problem about memory page fault and TLB questions[10]. However, in our design, we must understand the drawback of VMM to improve performance [11, 12].

A common feature of all these simulators is that they can execute applications such as large-scale systems, server consolidation, and program optimization. On the other hand, although they can perform a multi-layer or multi-device simulation, their advantages and disadvantages are different from each other. In general, instruction level and device simulator belong to a very detailed design because correction is very important. As a result, their performance will be bad. Following these reasons, a full system simulator is too complicated and thus we must propose the DCSim simulator in this paper to remedy them. DCSim is similar to Xen to simulate a multi-layer virtualization system. In contrast, the real virtualization server built in Xen is hard to study; however, DCSim can meet several standards and simplify the system and multi-level analysis to simulate fast. Moreover, DCSim can also coordinate with a device simulator between the data center simulator and system simulator. Thus, withe the help of DCSim, we can have different views on VM and data center.

III. System Overview

Figure 3 shows the architecture of DCSim. It can support different file systems and execute applications on differ-

Table I
COMPARISON AMONG VARIOUS SIMULATORS.

Type	Name	CPU & Mem	Disk & Network	Application	Sacle
Cloud	NS2	-/-	fine	coarse	Packet-Level
	SSFnet	-	-/fine	coarse	Packet-Level
	GTNetS[4]	coarse	-/fine	coarse	Packet-Level
	GridSim[5]	coarse	fine/fine	coarse	Application
Non OS	Flow	coarse/math	math/math	emulator	System
	TransARM	virtualize/-	-/-	virtualize	CPU
	SimpleScalar	virtualize	-/-	virtualize	CPU
	ISS	virtualize/fine	-/-	virtualize	instruction
System	M5	virtualize/fine	-/fine	virtualize	Full system
	QEMU[2]	virtualize	virtualize	virtualize	Full system
	SimOS[6]	virtualize	virtualize	virtualize	Full system
	Xen[1]	virtualize	virtualize	virtualize	Full system
Device	Disksim[7]	-	virtualize/-	-	Device
	SSDSim[8]	-	virtualize/-	-	Device

ent VMs based on SSDHDD hybrid storage system. With SSD/HDD storage system, each VM can exchange data and communicate with other VMs. It can also trace each I/O operation and collect system-level information to help us understand the address space of an application in storage system. The SSDHDD simulation is similar to a normal hard disk drive that can provide the storage system information. In the upper layer, a VOS is a guest OS in VM with most ordinary functions of an operating system, including semaphore and other services. On the other hand, it also has a basic thread library to exploit the parallelism of an application so that we can easily port various kinds of benchmarks to it. For different layers, we use virtual system calls and threads to integrate them together. Through this cooperation between different layers, we can simulate and record the behavior of cooperative servers. When VOS is working, all I/O requests access I/O through I/O ring and they will be scheduled by I/O ring scheduler.

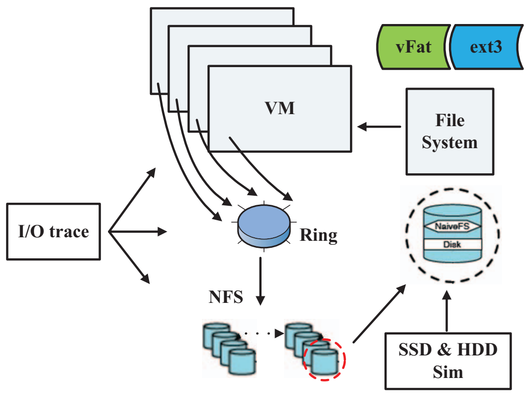

Figure 3. DCSim architecture.

Figure 4 shows the system functions provided by DCSim, where they can be divided into the upper half (user-level monitor) and the bottom half (kernel monitor). In our design, VS and VOS belong to user-level monitor. VH is the block-level device involved in kernel-level monitor to obtain linux kernel support. All user-level functions obtain kernel information through virtual system call. Place management is used to trace file information stored in HDD/SSD memory system. With management functions, we record the status of file sharing by working with user monitor. Thread scheduler is the basis of VOS implementation to arrange the order of all threads.

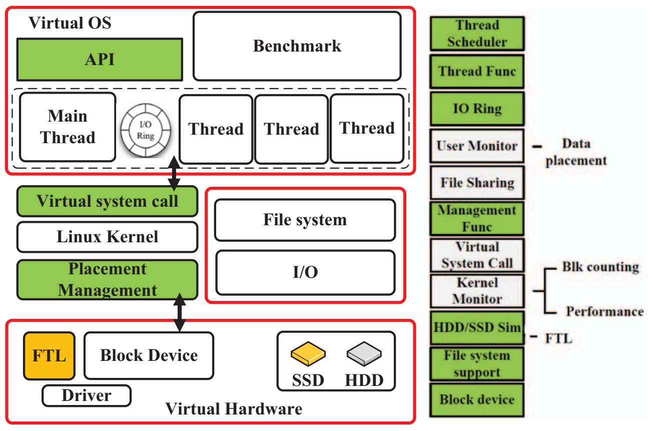

Figure 4. System functions provided by DCSim.

IV. IMPLEMENTATION

As mentioned before, DCSim consists of Virtual OS (VOS), Virtual Simulator(VS), and Virtual Hardware(VH). VOS is the guest operating system, VS is the platform running on VOS, and VH is used to simulate storage system. Based on this architecture, DCSim can be used to observe the status of how to use all resources on virtual device and virtual entity such as OS behavior and perform optimizations to improve system performance. The details of VOS, VS, and VH are presented below.

A. *VOS and VS*

VOS and VS are designed and developed on MicroC/OS-II [13] and then we port MicroC/os-II to Linux. By porting MicroC/os-II to linux, we can provide many useful functions to simulate complicated a virtualization system. This design help us understand changes caused by scheduler and synchronization mechanism. VS and VOS are responsible for the simulation of OS and VM. VOS simulates the behavior of an OS, including scheduler, interrupt, semaphore, thread library with thread APIs and so on. Each VOS is treated as a guest OS running on VM. When an application has been ported to VOS and executed, VS will monitor it to figure out the performance bottleneck. On the other hand, VS behaves like a VM hypervisor to to connect VOS with interrupt service routines and assign VOS to run on a virtual CPU.

We use threads to construct all requirements of the system and use extra threads to perform virtual interrupts. The threads in VS can be classified as three classes: main thread, interrupt thread, and VOS thread. Main thread is the virtual CPU that aims at scheduling threads of VOS and running correct tasks in a VOS thread. Interrupt thread will support some kinds of interrupt that works with an interrupt of VOS, and then VOS threads becomes a running thread on behalf of other threads in VOS.

For server consolidation, data center usually runs multiple VMs. Thus we run multiple VOSs to simulate a complicated VM environment. Figure 5 is an example to execute multiple VOSs. Each VOS is an independent MicroC/OS-II to share or compete the same VS. Therefore, we can use a real OS to understand OS behavior and the transparent analysis of a parallel program. Notice a major benefit of our approach is that we do not need using complicated tool to trace or log the underlying system.

In Xen, I/O request of each guest OS is under the control of I/O ring. Thus, the proposed VS has a similar mechanism to Xen, which is called virtual I/O ring. All I/O requests have be done through virtual I/O ring, which is implemented in VS layer. VOS and VS in VM send I/O requests to be executed by virtual I/O ring and then the behavior of I/O ring will be monitored by DCSim. VS is designed to monitor and verify the simulator by using two threads as scheduler and interrupts to support VOS. In this way, VS can handle all operations and logs necessary information.

B. *Virtual Hardware*

As the data amount has grown very rapidly, storage information will become very critical to a data center. Moreover, SSD and HDD have been used widely in data centers. To improve performance of virtualization in a data center, we have implemented virtual hardware by taking advantage of their advantages. Our design is SSDHDD based simulator to monitor block-level information in Linux kernel and user-level applications by coordinating with VS. Based on the information above, we can understand the effect of a multi-layer system.

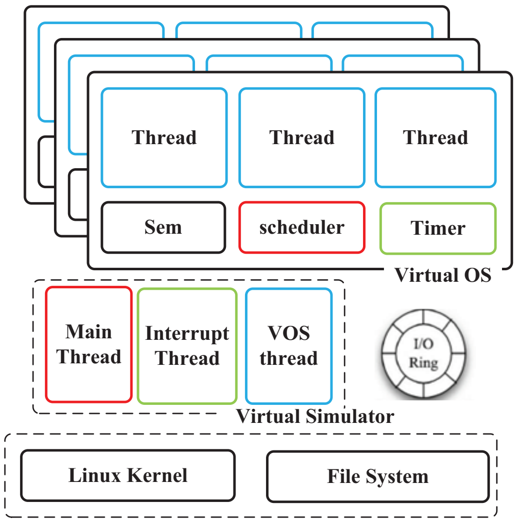

Figure 5. Virtual OS.

We use real HDD blocks to perform simulation and use a hard disk to simulate two devices. In the HDD simulator we add some monitor functions and implement the FTL layer. In our FTL design, SSD block is built on hard disk block and then we create the address mapping.

V. DCSim Analysis and Overhead

Our experiment has been performed with respect at storage system and VM impact. In our approach, VOS and VS work seamlessly to lest us understand the relationship between different layers. Under the multi-layer scheduler, we have evaluated the fair-sharing scheduler on all layers. Although it is a fair mechanism, it still result in different throughputs, as shown in Figure 6.

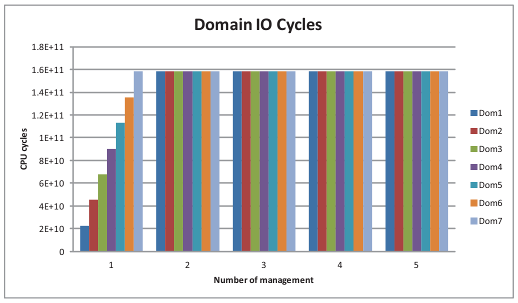

Figure 6. Evaluation of a fair scheduler.

Figure 7A shows the result of I/O delay caused by queuing delay, because I/O burst will put many I/O requests in virtual ring. Domain 1 is the first send I/O request and it may spend much time on I/O burst. If the total execution time has been taken into account, I/O requests of every vos is fair, as shown in Figure 7B, because different VOSs have different response times. How to reduce I/O delay is a challenge of virtualization, which happened in Xen. Now, our DCSim is built as an environment to minimize the same challenge.

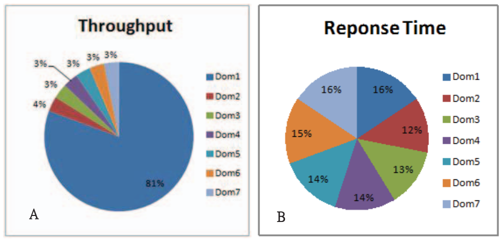

Figure 7. Evaluation of I/O delay.

If there are many I/O bursts, queuing delay will affect throughput significantly. Server consolidation that executes different applications will cause a lot of overhead due to some I/O bursts. Since DCSim is a light-weight design to perform similar virutalization like Xen, it will not produce too much redundant information. Besides, Figure 8 shows the result of running many domains without higher CPU workload. For a VM that has almost consume 100% loading, DCSim only needs 24% machine utilization to execute this workload. Thus, we can understand the challenge of multi-layer system very quickly and effectively to simulate with DCSim.

VI. Hybrid Storage Analysis

To measure the information arising from how to design application or hardware control on collaborative server, we perform the experiment to measure file accesses as follows. The system access a file of size 1,000,000 MB randomly and logs the information at the same time. Then the system will perform migration controlled by a state machine mechanism based on read/write count. In our approach, a popular file will be migrated to SSD and the less used file will be moved back to HDD with respect to various read/write ratios. Figure 9 shows the result for different combinations of read/write with respect to SSDHDD, HDD, and SSD. From the result, it is easy to see that we can obtain a significant performance improvement for the case that SSD to HDD ratio is 15% for R-SSD&RWHDD.

For a write-intensive application, we can acquire the similar performance result to the above. The RW-SDD/HDD will be much slower than the pure HDD because the state machine controller needs to move 15% reads to improve performance. In write intensive state, file placement performance for SSDHDD mix is usually lower.

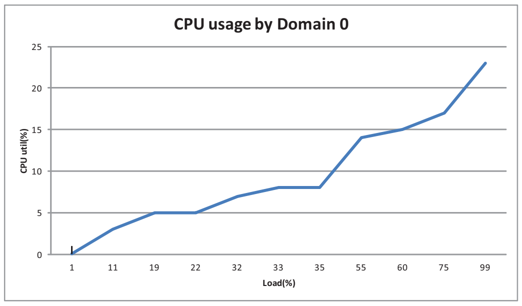

Figure 8. CPU usage for Domain 0.

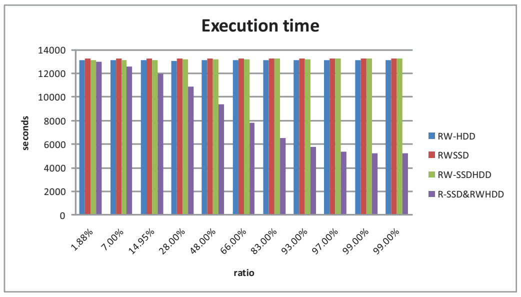

Figure 9. Placement result for SSD/HDD storage system.

Besides, Figure 10 shows the result of migration overhead. In general, this overhead increases as SSD ratio becomes larger because it arises from the migration of read intensive files. For mixed SSDHDD we can acquire the similar conclusion. If most reads occurs on SSD, the performance improvement can achieve about 15% and the migration overhead is just a little.

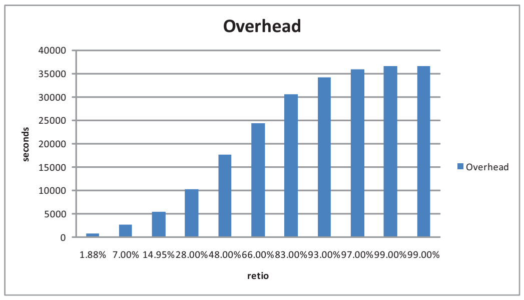

Figure 10. Migration overhead.

VII. Conclusion

In this paper, we have designed a simulator called DCSim to help us understand the design of data center and the factors to affect performance and analyze a data center. DCSim, which consists of VOS, VS, and VH based on the hybrid HDDSDD storage system, is a multi-layer design similar to a cooperative server to provide the analysis on storage, scheduler, and overhead etc. With its help, we can understand and analyze a data center easily and address the issues of a VM through the cooperation of three layers.

For a heterogeneous environment, smart mechanisms is necessary to manage a system by considering factors such as storage capacity and placement mechanism on hybrid storage system. The result shows DCSim can support virtualization well and the performance is better if the capacity ratio of SSD to HDD is 15%. In the near future, we attempt to design some enhancement techniques to improve performance.

References

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf NeugebauerE, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22,*, 2003.

[2] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *In Proceedings of the 2005 USENIX Annual Technical Conference, April.*, 2005.

[3] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *In Proceedings of the 27th Annual International Symposium on Computer Architecture, June*, pages 83–94, 2000.

[4] George F. Riley. Simulation of large scale networks ii: large-scale network simulations with gtnets. In *In Proceeding WSC '03 Proceedings of the 35th conference on Winter simulation: driving innovation.*, 2003.

[5] Rajkumar Buyya and Manzur Murshed. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. In *The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, USA, May.*, 2002.

[6] M. Rosenblum, S.A. Herrod, E. Witchel, and A. Gupta. Complete computer system simulation: the simos approach. In *In Proceeding of Parallel and Distributed Technology: Systems and Applications, Winter.*, 1995.

[7] John S. Bucy, Jiri Schindler, Steven W. Schlosser, and Gregory R. Ganger. The disksim simulation environment version 4.0 reference manual. In *Technical Report CMU-PDL-08-101, Carnegie Mellon University, May*, 2008.

[8] N. Agrawal and V. Prabhakaran. Design tradeoffs for ssd performance. In *In Proceedings of the Usenix Annual Technical Conference, June*, 2008.

[9] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews. Xen and the art of repeated research. In *In Proceedings of USENIX Annual Technical Conference.*, 2004.

[10] Xiantao Zhang, A.X.F Xu, Qi Li, D.K.Y. Yau, Sihan Qing, and Huanguo Zhang. A hash-tlb approach for mmu virtualization in xen/ia64. In *IEEE International Symposium on the Parallel and Distributed Processing, 2008. IPDPS 2008.*, 2008.

[11] Steven Hand, Andrew Warbeld, and Keir Fraser. Are virtual machine monitors microkernels done right? In *Proceedings of the 10th conference on Hot Topics in Operating Systems, p.1-1, June 12-15, 2005, Santa Fe, NM*, 2005.

[12] Aravind Menon, Jose Renato Santos, Willy Zwaenepoel, Yoshio Turner, and G. (John) Janakiraman. Diagnosing performance overheads in the xen virtual machine environment. In *Proceeding of the VEE '05 Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, 2005.

[13] J. Labrosse. Stability and tcp-friendliness and delay performance of aimd/red system. In *MicroC/OS-II: The Real-Time Kernel, 2nd edition , CMP Books*, 1998.