Disclaimer: "The contents of this document are scribe notes for The University of Texas at Austin EE382V Spring 2007, Computer Architecture: User System Interplay". The notes capture the class discussion and may contain erroneous and unverified information and comments.

GPP virtualization: A comparison of software and hardware techniques for x86 virtualization

Lecture #5: Monday, 5 February 2007
Lecturer: Mattan Erez
Scribe: Denis Gudovskiy
Reviewer: Min Kyu Jeong

1 References

M. Rosenblum, "The Reincarnation of Virtual Machines", ACM Queue vol. 2, no. 5. (background paper).
K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization", ASPLOS-XII (required paper).

2 Introduction

Presentation: George Abraham and Kyle Cullen

2.1 What is virtual machine?

Initially, virtual machine (VM) meant a software abstraction with the looks of a computer system's hardware. Now, VM can be seen as a layered abstraction over various places in the system including following levels of virtualization: hardware–level, operating system–level, and high–level language VMs.
Hardware–level approach was the first attempt to implement virtualization. Several hardware VMs were built between 1960's and 1970's (see classic IBM's VM/370) and then interest to hardware VMs became extinct. Basically, hardware VM is a thin layer of software called the virtual machine monitor (VMM). VMM manages executing VM code

providing illusion of raw hardware execution. Nice thing about hardware VMs is that they can execute any software. VMM at the same time must execute VMs with obeying a safety. VMs have to work in isolation to protect own data and it is VMM's job to arbiter all running VMs.
Operating system–level virtualization concept is that VM layer lies between operating system and actual applications. These applications could be VMs. So, VM interacts with operating system–level virtualization manager with direct access to hardware.
High-level language virtualization technique is just programming language (everyone knows Java) that has means to provide abstraction layer. This virtualization layer resides inside program. Compiled code in this way can run on every machine using abstraction layer on top of the operating system.

2.2 Why use virtual machine?

In the general case, VMs allow to run several OSes on one computer or execute applications complied on one machine on other machines with different hardware and OSes. Due to definition VMs must have several attributes:

- Software compatibility and platform independence. Operating systems, applications can run on other (host) operating system and various hardware platforms.
- Isolation. VMs are isolated from host operating system and between each other. So, they can be potentially safer.
- Encapsulation. VMs provide a level of indirection which can be used to control the execution of the virtual machine. This extra control can be exploited to increase safety and improve management.
- Performance. VM inevitably adds performance overhead. Fortunately, different optimizations are possible in order to increase performance.

3 Paper discussion

3.1 What is the problem being solved?

Briefly, paper solves following couple problems:

- Software and hardware techniques for VMs.
- Implicitly, paper appeals to SW and HW guys to discuss between each other what will be optimal to resolve VM problems e.g. what HW is lacking to handle VMs.

Why did VMware explore hardware/hybrid VMs instead of just improving software ones that they successfully built and sold before? First of all, until recently processors

didn't have good enough hardware support for virtualization. Also, there are several benefits to use hardware VMMs that likely will reincarnate hardware VMMs in future commercial products. Of course, hardware VMs performance potentially could be better than software ones because of natively less overheads. Secondly, compatibility issues on software VMs. Imagine how many operating systems are used today with permanent updates and how many will appear soon. Because of various operating systems it is hard to rewrite and verify software VMs to keep isolation and high performance every time instead of just use thin standardized VMM. Hence, paper compares software and hardware techniques for VMs as well as discusses what hardware is lacking in order to build strong VM.

3.2 Who are the intended users?

Intended users of VMs are just everyone who wants "software and hardware independence". It is great to reuse any software executing on any hardware. More specific users are following:

- Computer architects that design future microprocessors. Since one of the main message of the paper was to review hardware support for VMs, computer architects may find what software guys expect from hardware.
- System programmers who are writing VMs to understand how OS is working and interacting with VMs. Operating–system level VMs resides between OS and applications. So, system programmers could find useful information of how OS interact with VM.
- Programmers/HW designers that want to evaluate their solutions on various SW/HW platforms. Suppose you built something. Then, it is cheaper and faster to test your solution on various configurations using VMs rather than have tens environments.
- Data centers which are interested in VM technology in order to offer various SW environments for their users. Data servers have to provide service for clients that gives an opportunity to run any software with no issues.

3.3 What is unique about the suggested solution?

The main goal is that the paper tries find optimal solution for virtualization by exploring strong points and weaknesses of both software and hardware worlds and proposing solutions or at least highlighting problems. Some other unique solutions:

- Paper explored possible SW/HW techniques typically used for VMs.
- The comparison performs a large amount of quantitative studies on the different workloads a virtual machine may see.

- The discussion highlights adaptive binary translation software technique for VMs.
- Code optimization techniques were reviewed and implemented.

3.4 How is the idea evaluated?

Although, evaluation was not big part of the paper, nice thing about evaluation is that the authors tried to highlight weaknesses using nano-benchmarks as well as doing typical benchmark comparisons. Nano-benchmark is just code that causes certain single event (system calls, page faults, division by zero etc.) that differently works on HW/SW VMs. These events mainly define overall performance of the VM.

The evaluation did a good job of explaining what it is they were doing and why. They started with a qualitative evaluation, moved down to applications, and explained specific behavior with the nano-benchmarks they developed.

The evaluation included following:

- Several benchmarks (SPECint, a JAVA benchmark, and some custom nano-benchmarks) are used to gauge performance of different VM configurations.
- Performance is evaluated compared to both native machine execution, software, and hardware VM techniques.
- Authors measured the microarchitectural improvements over new and previous Intel cores and took the microarchitecture into account to show the overhead of hardware techniques.

3.5 Was the evaluation in line with the stated user requirements?

Yes, the evaluation compares the different techniques and quantitatively analyzed them. Paper came up of what is the problem in HW and SW techniques and highlighted possible future approaches (hybrid VMs etc.). Critique point is that the comparison made was not completely the same. For example, Intel microarchitecture they evaluated did not have support 64-bit segment limits, but that was mentioned in the paper.

3.6 Was technology a factor in the problem or solution?

Yes, while not explicitly stated, the continual improvement of modern x86 processor microarchitecture plays a big part in the support and improvement of VMs. While hardware still doesn't have good support for VMs, future cores highly likely will have better VM support. For example, current microprocessors do not have native support for MMU virtualization, and the extensions that have been recently added only have support for "trap-and-emulate", which does not always improve performance.

3.7 Were new tools or software techniques introduced?

- As mentioned before paper introduced a set of new toy nano-benchmarks.
- The VMware Monitor was discussed and utilized. Remember that hardware VMs must have virtual machine monitor (VMM) that is a thin layer of software which sits between hardware and OS.
- Several software virtualization techniques were introduced. While that wasn't new, paper explains several points about software virtualization that they use in VMware.

3.8 How may users with other requirements be affected?

- Hardware implementation of VMs add overheads to processor design which will affect on power, area, frequency.
- VMs potentially could have security issues. To obey isolation it is necessary that VM may not get access to somebody other data.
- OS has to care about running VMs. OS, while executing VM code, should know that some VM now is running in order to separate VMs and typical processes. This could be beneficial for better management of resources and under some potential issues.
- VMs can be beneficial for other than x86 markets: for cell phone/embedded market, everywhere where users need easily portable to different hardware platforms system software solutions.

3.9 How can the discussion of this paper be generalized in the context of the class?

- VMware engineers appealed to HW companies that they need modified microarchitecture in order to build better VMs. Better means performance, compatibility, development and verification overheads.
- Design without interplay between SW/HW companies does not lead to better satisfaction of user requirements. In this case, users are not able to get good hardware VMs because of lacking better support of VMs in hardware.
- Hybrid hardware/software techniques potentially could be more efficient in future. This point was not reviewed in details, but it was said hybrid technique can use advantages of both worlds.

- The evaluation did a good job of explaining what it is they were doing and why. They started with a qualitative evaluation, moved down to applications, and explained specific behavior with the nano-benchmarks they developed.