Virtual Machines

Syllabus for Fall 2017

Overview

Virtual machines have transformed the ways in which we build, manage, and interact with modern computer systems. A great deal of the network packets that you send as you sit at your computer are handled by virtual machines that may move to another side of the world in a matter of hours. If you have an Android phone, every application you launch executes in a virtual machine. Have you written a Java or Python program lately? It couldn't run without a virtual machine sitting underneath it. VMs have enabled new development practices for production applications, mobile applications, web applications, and operating system kernels. They allow datacenter operators to more efficiently provision hardware resources, saving money and energy. They allow service providers to quickly react to failures in an efficient and clean way. They enable the construction of portable and platform-agnostic programming languages by decoupling applications from the hardware on which they run. While VMs have been around since the early 1970s, modern developments, from cloudlets to containers, are only increasing the utility of virtualization technologies. One can expect that their importance and relevance will only increase.

This means that a basic understanding of the technological foundations underlying virtual machines should be part of any computer scientist's repertoire. This course will draw back the curtains and expose the magic that makes various types of VMs work. By the end of the course, you will have a deep understanding of hypervisors, system virtualization, machine emulation, language virtual machines, binary translators, virtual resource management, and more. You will gain exposure to a real-world hypervisor code-base. Furthermore, you will actually build a virtual machine and develop an intuition for using VMs to solve problems. The course will involve lectures, written assignments, involved programming projects, and discussions of foundational research papers.

Course Goals

The goals fo this course will be for you to develop a deep understanding of various types of virtualization techniques, their advantages and disadvantage, and to be able to apply them in a practical setting. You will be able to build basic VM constructs and understand how to evaluate them. You will therefore be expected to strengthen your system programming skills. You will also learn about new and upcoming technologies related to virtualization.

In general, you will learn about the following topics (and potentially others):

- General resource virtualization: virtual memory, virtualized CPUs, OS abstractions
- Basic Virtual Machines and historical context. Formal requirements for virtual machines
- Emulation: interpretation and binary translation
- Language VMs and Process VMs: JVM, Python, .NET CLR, Android's Dalvik JVM, VCODE, JIT compilation
- Full system virtualization: Type I & II VMMs, paravirtualization, device virtualization
- Virtualization in practice: virtual resource management, live migration, memory ballooning, fault tolerance
- Network virtualization: virtual overlays, bridged networking, NAT, software-defined networks
- Virtual storage: storage virtualization, virtual disk formats, copy-on-write formats

- Advanced topics in virtualization: hardware assisted virtualization, overheads of virtualization, performant VMs, interrupt virtualization, self-virtualizing devices, security, introspection, the semantic gap
- Current and future directions: VMMs as microkernels, Unikernels, chroot, jails, containers, cloudlets, virtual appliances

Prerequisites

You must have taken CS 351 and CS 450. I will only waive this requirement in truly exceptional cases—this will be a challenging course, and you will need to already be quite familiar with the fundamentals of operating systems. I also recommend having taken CS 551 and CS 470 or 570 (computer architecture).

What you will learn

- You will gain an understanding of the motivation for virtual machines and the history of their development
- You will learn about the different types of virtual machine architectures and their applications
- You will learn about full system virtualization as it is used in practice
- You will gain an understanding of the nuts and bolts that make modern cloud-based services such as Amazon AWS work
- You will get extensive experience developing virtual machine architectures. This experience will range from low-level kernel programming to implementing interpreters for high-level languages and emulators for instruction set architectures
- You will gain experience navigating and understanding a real-world hypervisor code-base
- You will learn about current and future topics in virtualization

Required Texts

The following textbook will be required for this course:

Jim Smith & Ravi Nair. *Virtual Machines: Versatile Platforms for Systems and Proesses*, 1st edition, Morgan Kaufmann, 2005.

Recommended Texts

If you are looking for books to help you in this area, and for good reference texts to have on your shelf, please consider the following:

- Anderson & Dahlin. *Operating Systems: Principles and Practice*, 2nd edition, 2014.
- Remzi & Andrea Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*, available online at `www.ostep.org`.
- Bovet & Cesati. *Understanding the Linux Kernel*, 3rd edition, 2005.

Lectures

You're expected to attend lectures; attendance counts toward the final grade. I will likely post my lecture notes online after lecture, but this is not a guarantee, so come to lecture!

Tentative Lecture Schedule

1. Class overview and introduction to resource virtualization
2. Resource virtualization continued. Types of VMs
3. Computer architecture review
4. Formal requirements for virtualization. History of virtualization, IBM System/370
5. Emulation I: Interpreting instruction sets
6. Emulation II: Theoretical foundations (Turing machines, Turing equivalence, UTMs)
7. Emulation III: Binary translation
8. Emulation IV: Optimizations
9. Process VMs I: Emulating architectures (QEMU, MAME)
10. Process VMs II: Emulating OSes
11. Process VMs III: Implementation
12. Language VMs I: Motivation and Basics, Intermediate representations (IR)
13. Language VMs II: Case studies: JVM, Python VM, .NET CLR, VCODE, Dalvik
14. Language VMs III: dynamic code generation, dynamic binary transformations, JIT compilation
15. Language VMs IV: Memory management & garbage collection
16. System Virtualization I: Privileged features, trap & emulate model. Disco, VMware
17. System Virtualization II: Taxonomy - Type I vs. Type II
18. System Virtualization III: Type I VMMs - ESX
19. System Virtualization IV: Type II VMMs - Palacios, KVM + QEMU
20. System Virtualization V: Semantic gap, paravirtualization, Xen
21. System Virtualization VI: Virtualized paging models: nested paging, shadow paging, TLBs
22. System Virtualization VII: Hardware assisted virtualization
23. System Virtualization VIII: Device virtualization, storage virtualization, virtual disks
24. System Virtualization IX: Virtual timekeeping. Multiprocessor and SMP virtualization
25. Management I: Resource management,
26. Management II: fault tolerance, live migration, memory ballooning
27. Network Virtualization I: Overlays, Network virtualization models
28. Network Virtualization II: Software-defined Networking
29. Advanced Topics: Interrupt virtualization, high-performance VMMs (Palacios)
30. Current and Future Directions: jails, chroot, containers, cloudlets, and virtual appliances

Academic Honesty

Students are responsible for maintaining the highest level of academic integrity, as discussed in the IIT Code of Academic Honesty. The normal penalty for violations of this policy, especially copying or other cheating during tests, is an E for the course, plus notification of the student's advisor and/or department and any appropriate administrators.

Disability Policy

Reasonable accomodations will be made for students with a letter of accomodation from The Center for Disability Resources (3424 S. State Street - 1C3-2; 312-567-5744). Please discuss any necessary accomodations with me, well ahead of time.