Virtualization

Dr. Prasenjit Chanak
Assistant Professor

Department of Computer Science and Engineering
Indian Institute of Technology (BHU), Varanasi-221005

Properties of VMM

- Three properties have to be satisfied by VMM:
  - Equivalence
  - Resource control
  - Efficiency

Hardware-Assisted Virtualization

- This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation
- This technique was originally introduced in the IBM System/370
- At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with Intel VT and AMD V
- Early products were using binary translation to trap some sensitive instructions and provide an emulated version
- Products such as VMware Virtual Platform, introduced in 1999 by VMware, which pioneered the field of x86 virtualization, were based on this technique
- After 2006,Intel and AMD introduced processor extensions, and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others

Full Virtualization

- Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware
- To make this possible, virtual machine managers are required to provide a complete emulation of the entire underlying hardware
- The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform
- A simple solution to achieve full virtualization is to provide a virtual environment for all the instructions, thus posing some limits on performance
- A successful and efficient implementation of full virtualization is obtained with a combination of hardware and software, not allowing potentially harmful instructions to be executed directly on the host
- This is what is accomplished through hardware-assisted virtualization.

Paravirtualization

- This is a not-transparent virtualization solution that allows implementing thin virtual machine managers
- Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified
- The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution

Paravirtualization

- This allows a simpler implementation of virtual machine managers that have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host
- This technique has been successfully used by Xen for providing virtualization solutions for Linux-based operating systems specifically ported to run on Xen hypervisors
- Other solutions using paravirtualization include VMWare, Parallels, and some solutions for embedded and real-time environments such as TRANGO, Wind River, and XtratuM.

Partial Virtualization

- Partial virtualization provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation.
- Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported, as happens with full virtualization.
- An example of partial virtualization is address space virtualization used in time-sharing systems; this allows multiple applications and users to run concurrently in a separate memory space, but they still share the same hardware resources (disk, processor, and network)
- Historically, partial virtualization has been an important milestone for achieving full virtualization, and it was implemented on the experimental IBM M44/44X

Operating System-Level Virtualization

- Operating system-level virtualization offers the opportunity to create different and separated execution environments for applications that are managed concurrently
- Differently from hardware virtualization, there is no virtual machine manager or hypervisor, and the virtualization is done within a single operating system, where the OS kernel allows for multiple isolated user space instances
- The kernel is also responsible for sharing the system resources among instances and for limiting the impact of instances on each other

Operating System-Level Virtualization

- This virtualization technique can be considered an evolution of the chroot mechanism in Unix systems
- The chroot operation changes the file system root directory for a process and its children to a specific directory
- As a result, the process and its children cannot have access to other portions of the file system than those accessible under the new root directory
- This technique is an efficient solution for server consolidation scenarios in which multiple application servers share the same technology: operating system, application server framework, and other components
- Examples of operating system-level virtualizations are FreeBSD Jails, IBM Logical Partition (LPAR), SolarisZones and Containers, Parallels Virtuozzo Containers, OpenVZ, iCore Virtual Accounts, Free Virtual Private Server (FreeVPS), and others.

Programming Language-Level Virtualization

- Programming virtualization is mostly used to achieve ease of deployment of applications, managed execution, and portability across different platforms and operating systems
- It consists of a virtual machine executing the byte code of a program, which is the result of the compilation process
- Compilers implemented and used this technology to produce a binary format representing the machine code for an abstract architecture
- The characteristics of this architecture vary from implementation to implementation

Programming Language-Level Virtualization

- Programming language-level virtualization has a long trail in computer science history and originally was used in 1966 for the implementation of Basic Combined Programming Language(BCPL), a language for writing compilers and one of the ancestors of the C programming language
- Other important examples of the use of this technology have been the UCSD Pascal and Smalltalk
- The Java virtual machine was originally designed for the execution of programs written in the Java language, but other languages such as Python, Pascal, Groovy, and Ruby were made available.

Programming Language-Level Virtualization

- The ability to support multiple programming languages has been one of the key elements of the Common Language Infrastructure(CLI), which is the specification behind .NET Framework
- Both Java and the CLI are stack-based virtual machines:
  - The reference model of the abstract architecture is based on an execution stack that is used to perform operations
  - An example of a register-based virtual machine is Parrot, a programming-level virtual machine that was originally designed to support the execution of PERL and then generalized to host the execution of dynamic languages

Programming Language-Level Virtualization

- The main advantage of programming-level virtual machines, also called process virtual machines, is the ability to provide a uniform execution environment across different platforms
- Security is another advantage of managed programming languages; by filtering the I/O operations, the process virtual machine can easily support sandboxing of applications
- As an example, both Java and .NET provide an infrastructure for pluggable security policies and code access security frameworks