EC 6013 Adv. Comp. Sys. Arch.

Virtual Machines : a virtual machine is taken to be an efficient, isolated duplicate of the real machine. This idea can be explained through a virtual machine monitor (VMM) which has three essential characteristics. First, the VMM provides an environment for programs which is essentially identical with the original machine. Secondly, programs run in this environment show at worst only minor decreases in speed. Thirdly, the VMM is in complete control of system resources.

• Virtual machines (VM) were first developed in mid-1960s, and they remained an important part of mainframe computing over the years. Although mostly ignored in the single user PC era in the 1980s and 1990s, they have recently gained popularity due to i) increasing importance of isolation and security in modern systems; ii) failures in security and reliability of standard operating systems; iii) sharing of a single computer among many isolated users, in particular for ~~cloud~~ cloud computing, and iv) dramatic increases in raw speed of processors over the decades, which made the overhead of VMs more acceptable.

• In broad sense, definition of VM includes all emulation methods that provide a standard software interface, such as Java VM. One may be interested in VMs that provide a complete system-level environment at the binary instruction set architecture (ISA) level. Although some VMs run different ISAs in the VM from the native hardware, one assumes that they always match the hardware. Such VMs are called (operating) System Virtual Machines. IBM VM/370, VirtualBox, VMware ESX Server, and Xen are examples.

• System virtual machines present the illusion that the users have an entire computer to themselves, including a copy of the operating system. A single computer runs multiple VMs and can support a number of different operating systems (OSes). On a conventional platform, a single OS "owns" all the h/w resources, but with a VM, multiple OSes all share the h/w resources.

P.T.O

• The software that supports VMs is called a virtual machine monitor (VMM) or hypervisor; the VMM is the heart of virtual machine technology. The underlying hardware platform is called the host, and its resources are shared among the guest VMs. The VMM determines how to map virtual resources to physical resources; a physical resource may be time-shared, partitioned, or even emulated in software. The VMM is much smaller than a traditional OS; the isolation portion of a VMM is perhaps only 10,000 lines of code.

• Apart from improving protection, virtual machines (VMs) provide two more commercially significant benefits:

1) Managing software: VMs can run multiple operating system environments on a single physical computer, saving physical space, time and management costs. VMs support legacy applications, reducing the cost of migrating to a new OS. VMs provide an abstraction that can run the complete software stack, even including old operating systems like DOS. A typical deployment may be some VMs running legacy OSes, many running the current stable OS release, and a few testing the next OS release.

2) Managing hardware: a reason for multiple servers is to have each application run under the compatible version of the OS on separate computers; this separation can improve dependability (for example, even if one machine malfunctions, some other computer can well and take up the load of the machine which is out of order). Also, VMs allow separate s/w stacks to run independently yet share hardware, thus consolidating the no. of servers. Also, VMMs support migration of a running VM to a different computer, either to balance load or to evacuate from failing or malfunctioning hardware

---

Virtual m/cs: are virtual computers within (physical) computers — A VM is no different than any other physical computer like a laptop, smart phone or server. It has a CPU, memory, disks to store your files, and connects to the Internet, if needed. VMs are often thought of as virtual computers or software-defined computers within physical servers, existing only as a code

Virtual Machines (Cont.)

- Amazon Web Services (AWS) uses the virtual machines in its cloud computing operations offering EC2 for five reasons:

< Note: Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an 'instance', containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. >

1) It allows AWS to protect users from each other while sharing the same server.
2) It simplifies software distribution within a warehouse-scale computer. A customer installs a virtual machine image configured with the appropriate software, and AWS distributes it to all the instances a customer wants to use.
3) Customers (and AWS) can reliably "kill" a VM to control resource usage when customers complete their work.
4) VMs hide the identity of the hardware on which the customer is running, which means AWS can keep using old servers and introduce new, more efficient servers. The customer expects performance for instances to match their ratings in "EC2 Compute Units", which AWS defines: to "provide the equivalent CPU capacity of a 1.0–1.2 GHz 2007 AMD Opteron or 2007 Intel Xeon processor." Newer servers usually offer more EC2 Compute Units than older ones, but AWS can keep renting old servers as long as they are economical.
5) VM Monitors can control the rate at which a VM uses the processor, the network, and disk space, which allows AWS to offer many price points of instances of different types running on the same underlying servers. For example, in 2020 AWS offered

P.T.O.

more than 200 instance types, from less than half a cent per hour (t3a.nano at $0.0047) to more than $25 (memory optimized x1e.32xlarge at $26.69), a price range of over 5000:1 (as $1 ≡ 100 cents)

- In general, the cost of processor virtualization depends on the workload. User-level processor-bound programs have zero virtualization overhead, because the OS is rarely involved; so, everything runs at native speeds. I/O-intensive workloads are generally also OS-intensive, executing many system calls and privileged instructions that can result in high virtualization overhead. On the other hand, if the I/O-intensive workload is also I/O-bound, the cost of processor virtualization can be completely hidden, since the processor is often idle waiting for I/O.
- The overhead is determined by both the number of instructions that must be emulated by VMM and by how much time each takes to emulate them. Hence, when the guest VMs run the same ISA as the host, as one may assume here, the goal of the architecture and the VMM is to run almost all instructions directly on the native hardware.

Requirements of a Virtual Machine Monitor (VMM)

- A virtual machine monitor must i) present a software interface to guest software, ii) isolate the state of guests from each other, and iii) protect itself from guest software (including guest OSes)
- The qualitative requirements are: i) Guest software should behave on a VM exactly as if it were running on the native hardware, except for performance-related behavior or limitations of fixed resources shared by multiple VMs, and ii) Guest software should not be able to change allocation of real system resources directly.
- To 'virtualize' the processor, the VMM must control the following: access to privileged state, I/O, exceptions, and interrupts - even though the guest VM and OS presently running temporarily use them.

Requirements of a Virtual Machine Monitor (cont.)

- For example, in the case of a timer interrupt, the VMM should suspend the currently running guest VM, save its state, handle the interrupt, determine which guest VM to run next, and then load its state. Guest VMs that rely on a timer interrupt are provided with a virtual timer and an emulated timer interrupt by the VMM.

- To be in charge, the VMM must be at a higher privilege level than the guest VM, which generally runs in user mode; this also ensures that the execution of any privileged instruction will be handled by the VMM. Basic requirements to enable virtualization of system are:
  i) at least two processor modes, system and user
  ii) a privileged subset of instructions that is available only in system mode, resulting in a trap if executed in user mode; all system resources must be controllable only via these instructions.

(Lack of) Instruction Set Architecture support for Virtual M/Cs

- If VMs are planned for during the design of ISA, it is relatively easy to reduce both the number of instructions that must be executed by a VMM and improve their emulation speed.
- An architecture that allows the VM to execute directly on the hardware may be termed as virtualizable, e.g. IBM 370 architecture
- Since VMs have been considered for PC and server applications only recently, most ISAs were designed without virtualization in mind. Those include x86 and most RISC architectures, including ARMv7 and MIPS (Note: MIPS whose full form is Microprocessor without Interlocked Pipelined Stages) is a family of reduced instruction set computer (RISC) ISA, developed by MIPS computer systems (now MIPS Technologies, based in USA) (64 bit, (32→64), introduced in 1985)
- As the VMM must ensure that the guest system only interacts with virtual resources, a conventional guest OS runs as a user mode program on top of the VMM. Then, if a guest OS attempts to access or modify information related to hardware resources via

RTO

a privileged instruction – e.g. reading or writing a status bit that enables interrupts – it will trap to the VMM. The VMM can then enable appropriate changes to concerned real resources.

- Thus, if any instruction that tries to read or write such sensitive information traps when executed in user mode, the VMM can intercept it and support a virtual version of sensitive information, as the guest OS expects.

Protection and Instruction Set Architecture

- Protection is a joint effort of architecture and OS; however, architects had to modify some details of existing ISA when virtual memory became popular. For example, the x86 instruction POPF loads the flag registers from the top of the stack (in memory). One of the flags is the Interrupt Enable (IE) flag. If one runs the POPF instruction in user mode, it simply changes all the flags except IE. In system mode, it does change the IE. Since a guest OS runs in user mode inside a VM, this is a problem, as it expects to see a changed IE.
- Historically, IBM mainframe h/w and VMM took three steps to improve performance of virtual machines:
  1) Reduce the cost of processor virtualization.
  2) Reduce interrupt overhead cost due to virtualization.
  3) Reduce interrupt cost by steering interrupts to the proper VM without invoking VMM.

<AMD and Intel tried to address the first point in 2006>.

- Final portion of the architecture to virtualize is I/O. This is the most difficult part of system virtualization due to the increasing number of I/O devices attached to the computer and the increasing diversity of I/O device types. Another problem is the sharing of a real device among multiple VMs, and yet another is due to the need to support many device drivers that are required, specially if different guest OSes are supported on the same VM system. The VM illusion can be maintained by giving each VM generic versions of each type of I/O devices drivers, and then leaving it to the VMM to handle real I/O.

google: why is virtual machine necessary?

- Virtual machines (VMs) can run multiple operating system environments on a single physical computer, saving physical space, time and management costs. VMs support legacy applications, reducing the cost of migrating to a new O.S.

⇒ from vmware.com: a VM is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine. Each VM runs its own O.S. and functions separately from the other VMs, even when they are all running on the same host. (for example, a virtual MacOS VM can run on a physical PC).

- VM technology is used for many use cases across on-premises and cloud environments. Public cloud services are using VMs to provide virtual application resources to multiple users at once., for even more cost-efficient and flexible compute

< cloud vs. on-premise s/w comparison: Fundamental difference is where the software resides. On-premise s/w is installed locally, on your business computers & servers, whereas cloud s/w is hosted on the vendor's server and accessed via a web browser >

VMware.com

→ CPU Virtualization emphasizes performance and runs directly on the processor whenever possible. The underlying physical resources are used whenever possible and the virtualization layer runs instructions only as needed to make VMs operate as if they are running directly on a physical machine.

Virtualization (from ibm.com)

- Virtualization is a process that allows for more efficient utilization of physical computer hardware and is the foundation of cloud computing.
- Virtualization uses software to create an abstraction layer over computer h/w that allows the h/w elements of a single computer - processor, memory, storage and more - to be divided into multiple virtual computers, commonly called virtual machines (VMs). Each VM runs its own operating system (OS) and behaves like an independent computer, even though it is running on just a portion of the actual underlying computer hardware.

It follows that virtualization enables more efficient utilization of physical computer hardware and allows a greater return on an organization's hardware investment.

(From Wikipedia): Virtualization is a modern technique (in computer science) developed in late 1950s, and is different from simulation and emulation. Note that simulation models the environment, while emulation replicates the target environment (such as certain kinds of virtual m/c environments). However, virtualization employs techniques used to create instances of an environment. Full virtualization requires that every salient feature of the hardware be reflected into one of several virtual machines - including the full instruction set, input/output operations, interrupts, memory access, and whatever other elements that are used by the software that runs on the bare machine, and that is intended to be run in a virtual machine). In such an environment, any s/w capable of execution on the raw h/w can be run in the virtual m/c, and, in particular, any operating systems. The obvious test of full virtualization is whether an operating system intended for stand-alone use can successfully run inside a virtual machine.