• Cloud Computing •

Module 2:

1) • Virtualization: Software used to run multiple Virtual computing resources on Single Hardware platform.

• VM: Software Virtualization of a hardware that executes & communicates like a Virtual Computer.

Files:

(a) .vmx: Contains configuration of the VM (such as RAM, CD-ROM, port info, power settings etc.). It is a plaintext file that can be edited using appropriate UTF-8 supporting program.

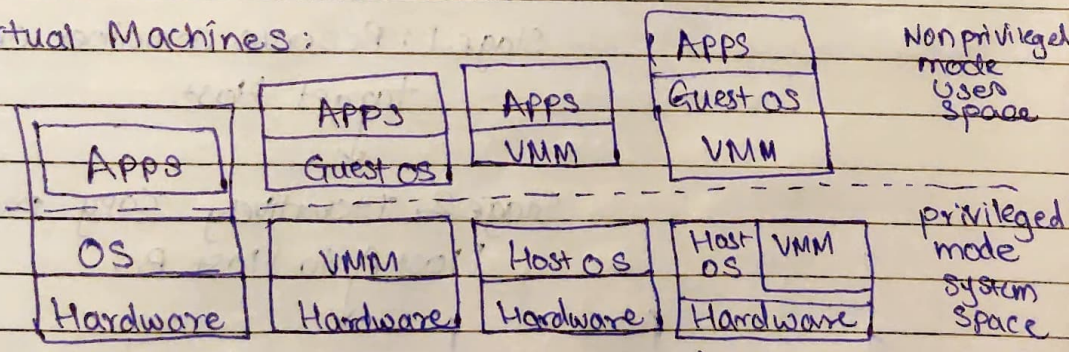
(b) .vmtx: A Template configuration (.vmx gets converted)

(c) .nvram: Contains BIOS settings (virtual) and can be accessed by pressing F2 while booting the VM.

etc.

Purpose of VM is to enhance computer's resource sharing & application flexibility.

2) Virtual Machines:



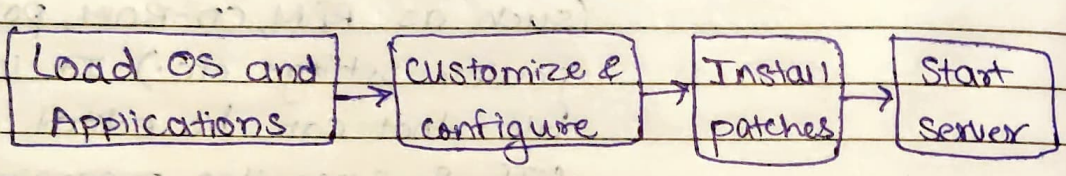
(a) Physical Machine (b) Native VM (c) Hosted VM (d) Dual-mode VM

- Primitive operation in VMs:

A VM can be suspended and stored in storage. Once resumed the applications that were running before VM was suspended will resume in their running state.

VM can be migrated from one hardware platform to other.

- Provisioning a VM:



- Live Migration:

Movement of a Live VM from ~~that~~ Host to other. Live migration takes place w/o noticable effect and it is automatically started by strategies like load balancing & server consolidation.

Stages: Stage 0 : Pre migration

↓

Stage 1 : Reserve container on target Host

↓

Stage 2 : Iteratively copy pages from Host A to Host B.

↓

Stage 3 : Suspend VM at A. ~~and start at B~~, ~~&~~ redirect network traffic to B.

↓

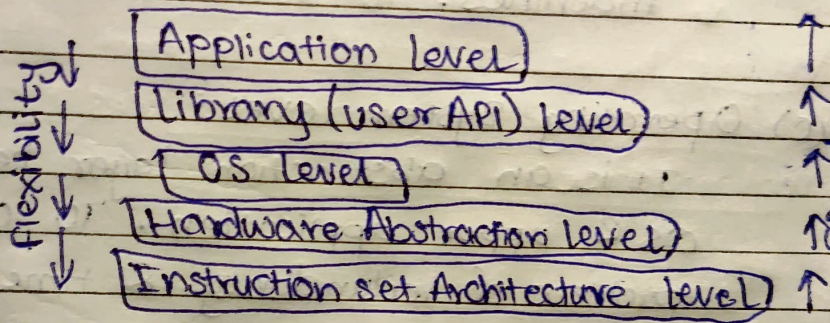
Stage 4 : B indicates A about successful transaction. B becomes primary Host

Stage 5: VM on B is activated, reattach drivers, recover programs (apps) etc.

Goal: least Downtime (the time betn switching hosts)

- Cold Migration: Migration of a powered off VM. Much easier than Live migration as it does not need a constant shared storage. It also does not need CPU compatibility checks.

3) Levels of Virtualization implementation: (options).



Virtualization on:

(a) Instruction Set level: Provides commands to the processor (such as addressing modes, native datatypes, registers, external I/O etc...)

eg: X86 found on Windows nowadays.

Virtualization here is performed by emulating the ISA given by host machine. A translation layer makes it efficient by performing optimized binary translation.

With this approach, we will be able to run large amount of legacy binary code written for different processors just on any given new hardware host machine.

Emulator eg: QEMU (open source emulator & virtualizer)

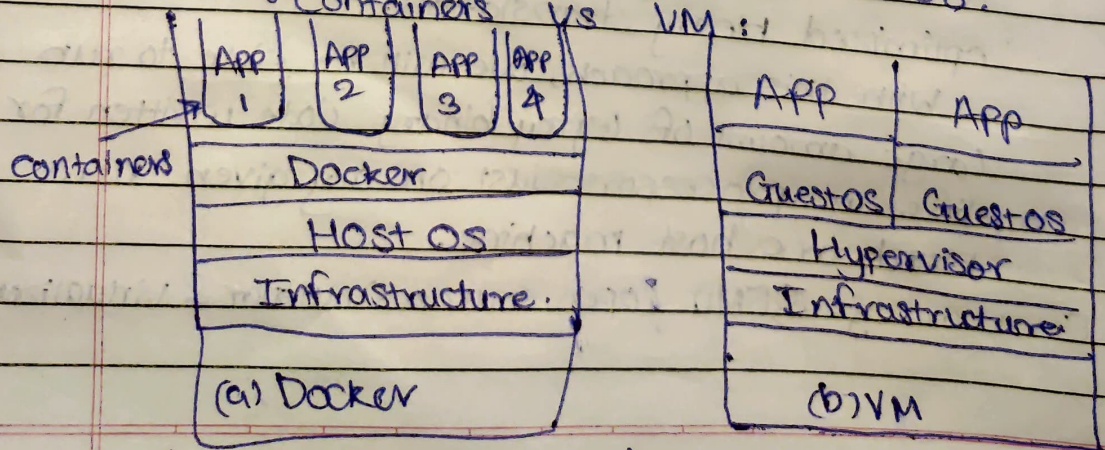
(b) Hardware Abstraction Level: (Hypervisor).
Generates hardware for VM.
Intention: TO upgrade hardware utilization rate by multiple users, concurrently.
A privileged instruction is something that can run only in kernel mode & needs CPU's full attention. VMM must handle privileged instructions issued by VM as there is possibility of multiple VMs being present.
Hypervisor (VMM) supports multiple OSs, gives appearance of multiple seeparate machines.

(c) Operating System Level:
It is an abstraction layer betn traditional OS & upper application. TO overcome problem of redundancy & time consumption.

Server: the physical server used to host multiple VMs.

Dockers:
A container is an environment which runs an application which is OS independant.
Basically, kernel of Host OS.

• Containers vs VM:

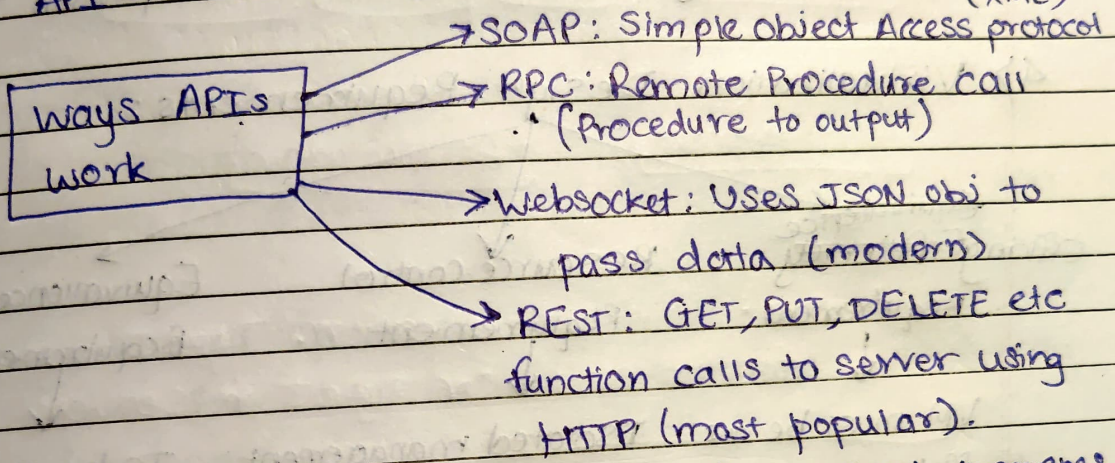


(no config. entanglement).

(d) Library Support Level:

Using APIs exported by user-level libraries rather than using lengthy OS system calls.

API : Client ⟷ Server [service contract] (xML)



Basically, Create execution env. for client programs rather than creating whole ass VM to run it.
eg: WINE (Wine Is Not an Emulator).

(e) User-level:

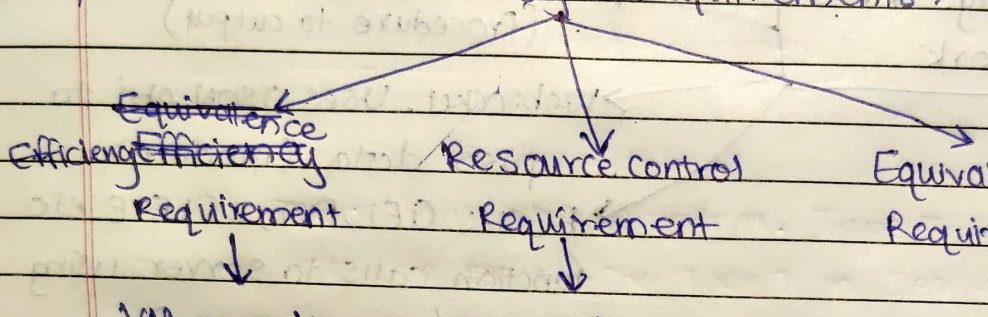
High Level Language (HLL) VMs.
A VM sitting on top of OS to help run programs in HLL. eg: Java in Java Virtual Machine (JVM).
JVM runs Java & programs that are compiled to ~~^~~ java bytecode.
JVM's ~~con~~ interpreter converts the source code into OS compatible machine code, making it platform independant. This therefore gets resulted into System calls.
JVM, .NET CLI are Stack based VMs.

• Other virtualizations such as Register based or program based virtualization also exist.

- Application Virtualization:
  Isolated application which is easier to distribute & remove from workstations.

- Refer to page 131 in PPT for comparison table.

4) Virtualization Design Requirements:



VM must be as efficient as a real machine

Isolated management of different resources

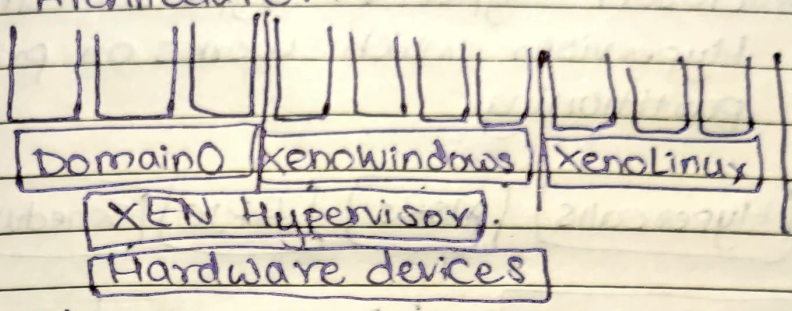
To be able to run all apps that a real machine can run

5) Hypervisor Architecture:
   Type 1: Bare Metal Hypervisor embeded on top of actual components.
   Type 2: Runs on a Host OS which provides Virtualization Services.

- Micro kernel hypervisor: includes basic features such as physical memory management & I/O etc.
- Monolithic hypervisor: All functions of micro kernel as well as of device drivers.

Example: (a) Xen Hypervisor: Type 1 and micro kernel. Used for desktop & server virtualization.

XEN Architecture:



- Domain0: is the guest OS on top of XEN which has control ability. It is privileged.
- DomainU: All other guest OS running on XEN.

6) Full Virtualization: (FV)

With Fv, non critical instructions run directly on hardware. While critical instructions are trapped to be addressed by software separately as they can cause security compromise or damage hardware.

eg: VMware.

Popek Goldberg instructions: → Privileged, → Control sensitive, → Behaviour sensitive.

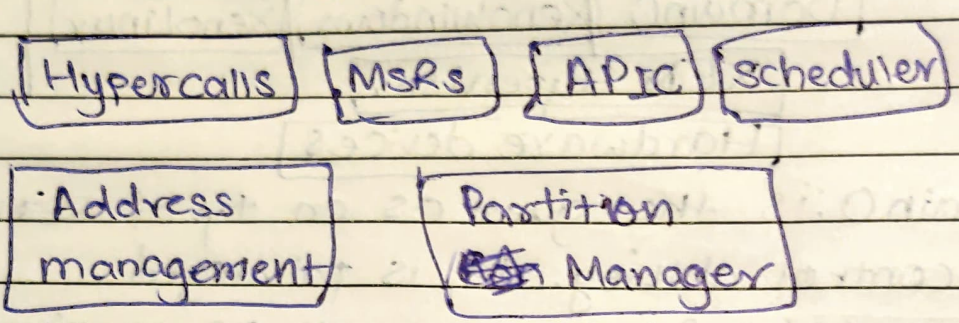
* • Translation: (Binary Translation).

The privileged, control & behaviour sensitive instructions are trapped in VMM and are translated into equivalent but harmless instructions.

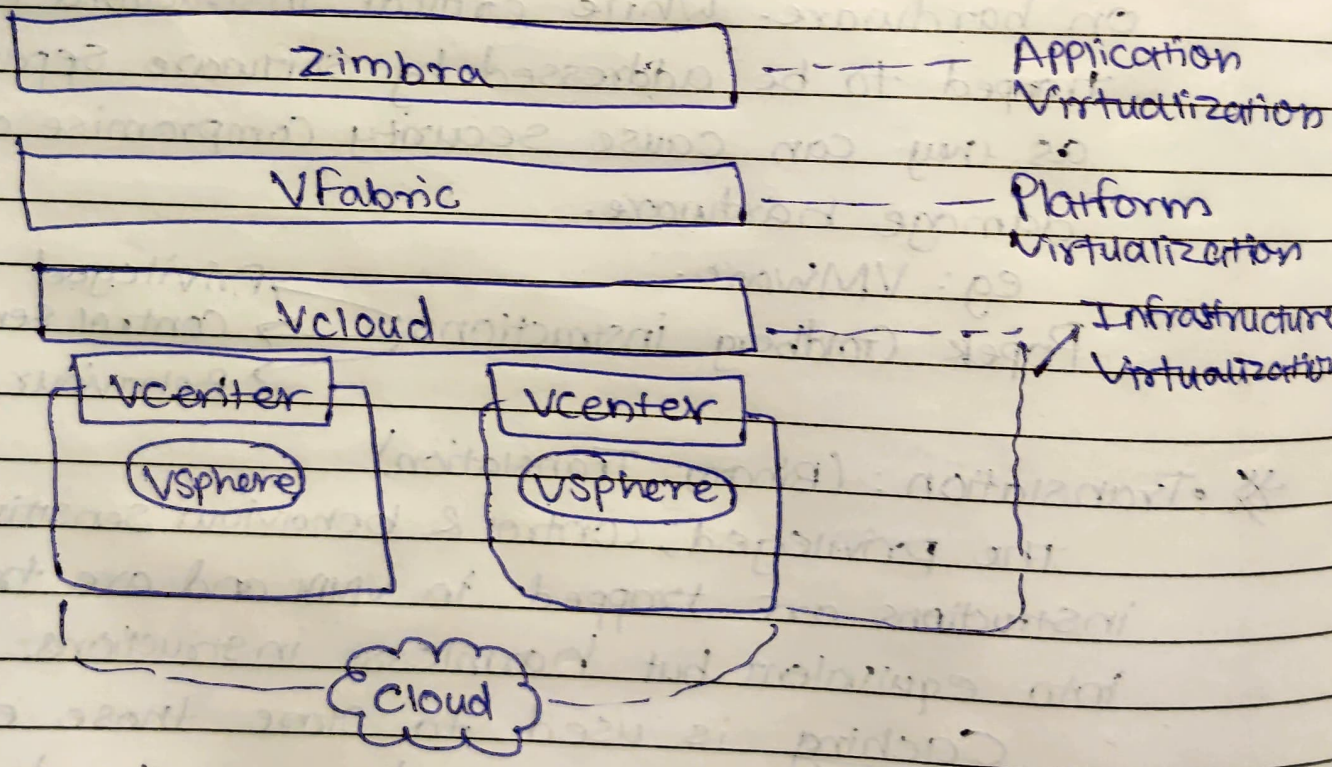
Caching is used to store these equivalent set of instruction to avoid repeatative translation.

• A para virtualized VM: Not Fv, it needs to modify the guest OS. Hypercalls (like sys calls) are made for critical instructions by compiler to hardware. Faster than Fv but more expensive.

- Microsoft Hyper-V: Type 1 Para Virtualization
  . Hypervisor which works on principal of partitioning.



- VMware: Full virtualization. Type II hypervisors in desktop env. or Type I in servers.



Refer to fig on Page 238 for Architecture of entire cloud model.