Cloud Computing II

UNIT II - Virtualization and DevOps

▼ Virtualization

1. **Hardware virtualization/Bare metal virtualization/ Type 1 Hypervisor**
    - VMM acts as traditional OS
2. **Hosted hypervisor/Type 2 hypervisor**
3. **Paravirtualization - Xen**
    - Presents a software interface to VMs that is similar to but not identical to that of the underlying hardware.
    - OS to be explicitly ported to run on top of the VMM
    - Intercepts and emulates privileged and sensitive instructions at compile time.
    - Compatibility and portability is doubtful, cost is high and performance varies. Low performance in binary translation.
    - VMWare ESX is a bare metal hypervisor for paravirtualization, virtualizes physical hardware, resource manager allocates resources and service console is responsible for booting the system.
4. **Full/Transparent virtualization**
    - Provides complete simulation of underlying hardware.
    - OS runs without modification
    - Intercepts and emulates privileged and sensitive instructions at runtime

▼ Virtualization software techniques

- Trap → hypervisor when VM tries to execute instruction that could change the state of the system.
- Emulate → instruction in hypervisor

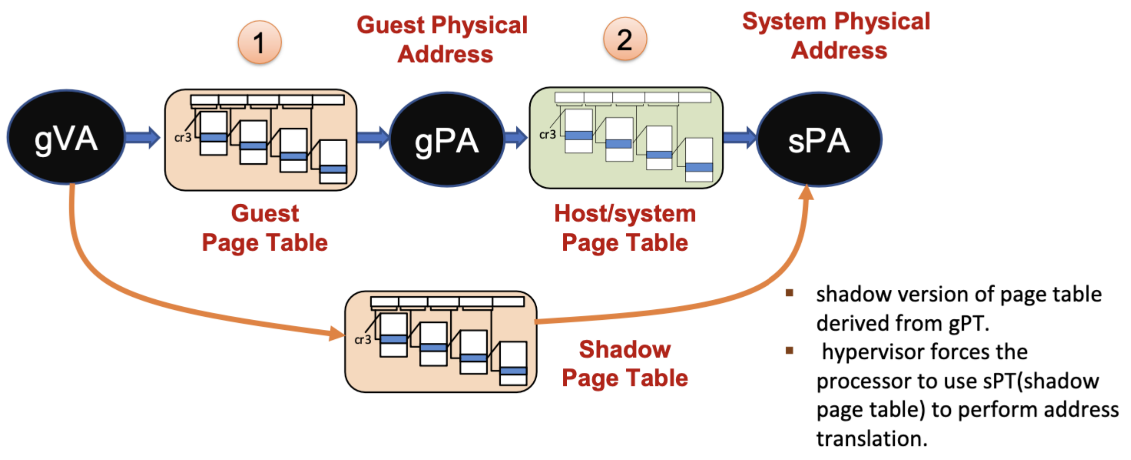
- Ring 0 - privilege mode
- Ring 3 - user mode
- VMM runs in ring 0, Guest OS runs in Ring 1. If guest application has to handle syscall/interrupt, traps it to vmm, vmm jumps to os trap handler, guest handles trap.
- Any privileged action by guest OS is trapped to VMM, VMM emulates it and then goes back to guest OS.
- <u>Issues w trap and emulate -</u>
    1. Performance overhead
    2. Not all architectures support it
    3. Some X86 instructions which change hardware state run in both privileged and unprivileged modes (popf, pushf)
    4. Guest OS recognises its running in a lower privilege level
    5. Memory protection is a challenge
- Binary translation technique - Implementing full virtualization - no trap and emulate, instead examines code for safe and unsafe instructions. Each instruction is translated and copied into translation cache.
  IDENT - instructions which pose no problem, modified and copied into translation cache.
  INLINE - simple dangerous instructions, translated into a short sequence emulation code
  CALLOUTS - dangerous instructions that need to performed by emulation code.

  Complexities of BT include control flow changes, retranslation of branches and keeping track of branches.

▼ Virtualization - Memory and IO

- VMs do not have direct access to physical memory, they only manage Host physical memory

- Implementing 2 level address translation - Shadow page table, Extended or Nested page table.
- Virtual address → mapped via shadow table → physical address



- shadow version of page table derived from gPT.
- hypervisor forces the processor to use sPT(shadow page table) to perform address translation.

- IO virtualization - abstract upper layer protocols from physical connections.
  → Full virtualization
  →para-io-virtualization (frontend backend shared memory)
  →direct io
- Advantages of io virtualization:
  → Flexibility
  →Cost minimization
  →Increased density
  →Minimizing cables

▼ Goldberg Popek principles

Terminologies:

1. Virtual machine - Complete compute environment with its own isolated processing capabilities, memory and comm. channels.
2. VMM/Hypervisor - System software that creates and manages VMs that is
   →Efficient

→Omnipotent
→Undetectable

Essential characteristics:

1. Equivalence
2. Resource control
3. Efficiency

Instructions:

1. Privileged - cause trap if processor is not in privileged mode
2. Sensitive - Behaviour and Control sensitive
3. Safe

Theorems:

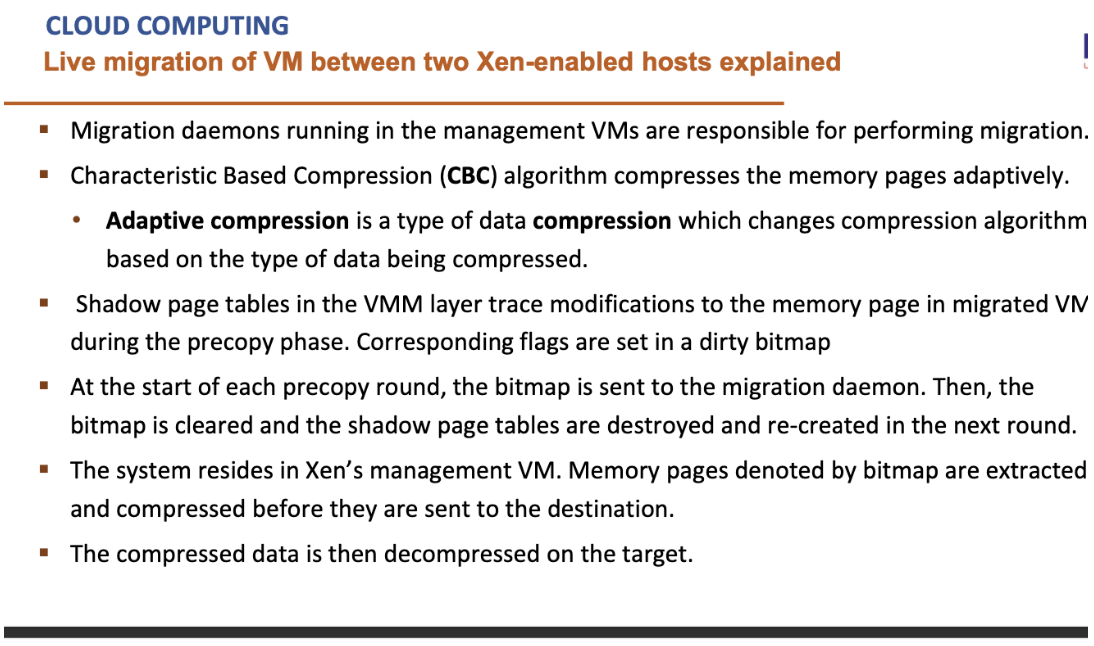
1. For any conventional 3rd gen computer, a VMM may be constructed if the set of sensitive instructions is a subset of the privileged instructions for that computer,
2. A conventional 3rd gen computer is recursively virtualizable if its virtualizable and a VMM without any timing dependencies can be constructed for it.
3. A hybrid VMM may be constructed for any 3rd gen conventional computer in which the set of user sensitive instructions are the subset of the set of privileged instructions.

▼ VM Migration

1. Cold migration (migrate a powered off vm)
2. Non live migration (source host is paused and transferred to target host and then resumed)
3. Live migration (no services disruptions)
    - Pre Copy - Select destination host, reserve resources, iterative precopying rounds, stop and transfer state, commitment, Vm activation
    - Post copy - Processor state is transferred to the destination server before memory content. Uses demand paging (retransmission from source server),

active push and memory prepaging (predicts required pages and makes pushing efficient)

- Disadvantages - Memory, Filesystem and network migration



CLOUD COMPUTING

Live migration of VM between two Xen-enabled hosts explained

- Migration daemons running in the management VMs are responsible for performing migration.
- Characteristic Based Compression (**CBC**) algorithm compresses the memory pages adaptively.
  - **Adaptive compression** is a type of data **compression** which changes compression algorithm based on the type of data being compressed.
- Shadow page tables in the VMM layer trace modifications to the memory page in migrated VM during the precopy phase. Corresponding flags are set in a dirty bitmap
- At the start of each precopy round, the bitmap is sent to the migration daemon. Then, the bitmap is cleared and the shadow page tables are destroyed and re-created in the next round.
- The system resides in Xen's management VM. Memory pages denoted by bitmap are extracted and compressed before they are sent to the destination.
- The compressed data is then decompressed on the target.

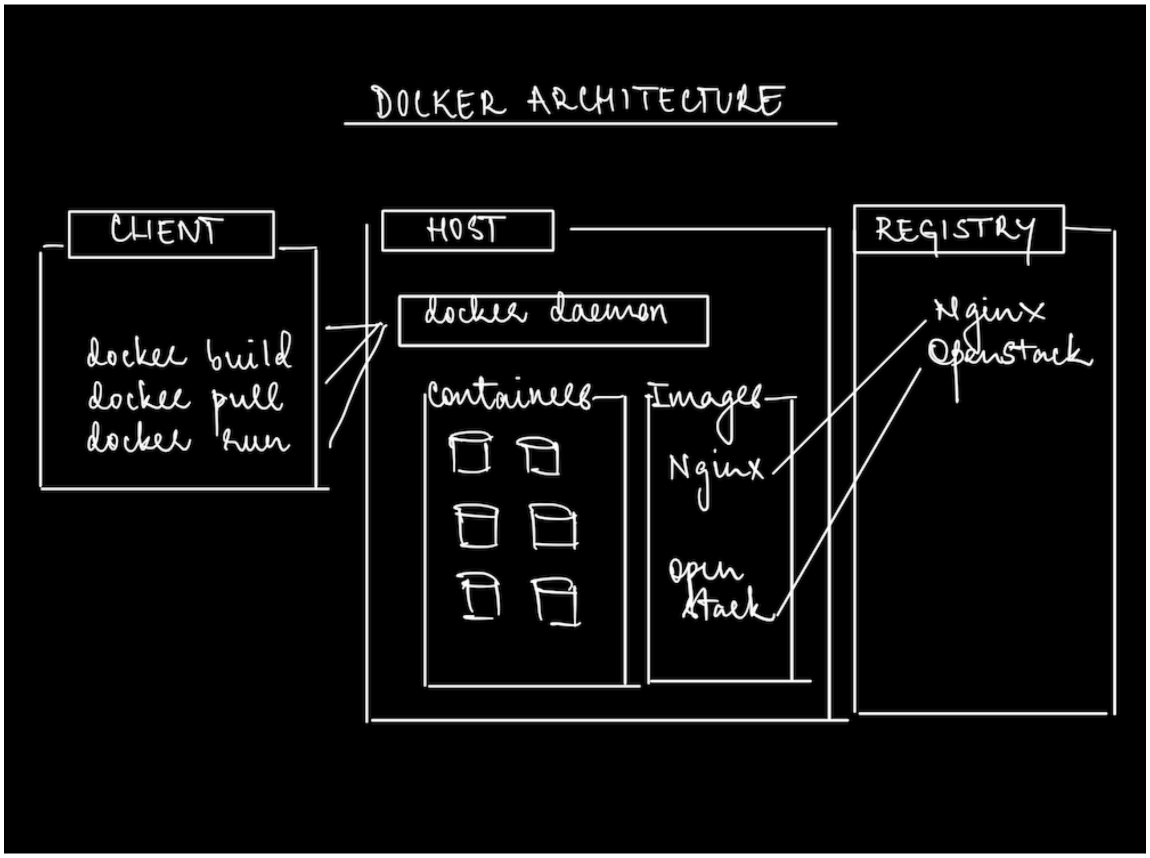
Reasons for VM migration -

- Hardware maintenance
- Balancing of workload
- Scaling to a changing workload
- Consolidating servers for utilization
- Workload mobility
- Performance
- Energy efficiency
- Availability support, Business Continuity

▼ Docker

- Open platform tool which makes it easier to create, test, ship, deploy and execute applications using containers.
- PaaS product that uses OS level virtualization to deliver software packages.
- Docker client → (rest API) Docker daemon → Docker registry

Docker architecture:



- Docker client - interaction bw users and docker, uses docker api, can communicate with more than one daemon.
- Docker host - runs docker daemon and can host registries that store docker images and containers

DOCKER OBJECTS -

1. Images
   Read only template with instructions for creating a docker container. Dockerfile -

script file with syntax. Each instruction is a layer in the image. layer is a set of files and file metadata that is packaged and distributed as an atomic unit.

2. Containers
   Ready applications created from docker images. Runnable instance of an image. Docker creates a set of namespaces when containder is created

▼ Containers, Namespaces and cgroups

Namespaces:

1. PID namespace
2. UTS namespace - multiple hostnames on a single host
3. MNT namespace - mount points
4. IPC namespace - isolation
5. NET namespace - Network access and structure
6. chroot syscall
7. cgroups
8. CAP drop - OS feature restrictions

UnionFS - creates an illusion of merging contents of several directories into one without modifying its physical sources. Operates by creating layers, making it lightweight and very fast. Layering with incremental images

▼ DevOps and Kubernetes

Continuous integration, delivery, testing, deployment

- Container orchestrator - centralized management software that allocates and schedules containers to run on a pool of servers.
- Kubernetes is the most pervasive container orchestration platform to address these challenges.
- Pod
  One or more containers that share storage and network within a K8 config.

Run microservices.
Share storage, linux namespaces, cgroups, ip, port address space

- Service
Coupling a set of pods to a policy by which to access them. Services are used to expose containerized applications to organisation from outside the cluster

Kubernetes architecture:

Master node

- Exposes API, schedules deployments, manages cluster.
- Entry point of all administrative tasks.
- Orchestrates the worker nodes.
- Components of master node:
→ Kube apiserver - exposes api entry points
→ etcd - key value stores all cluster data
→ Kube scheduler - schedules new pods on worker nodes
→ Kube control manager - runs controllers using api servers

Worker node

- Responsible for container runtime
- Worker node components:
→ Kubelet - agent that gets the configuration of a pod from the apiserver
→Kube-proxy - acts as a network proxy and load balancer, takes care of tcp udp network routing
→ Container runtime - runs containers