Virtualization Techniques

Para-Virtualization/ OS assisted:

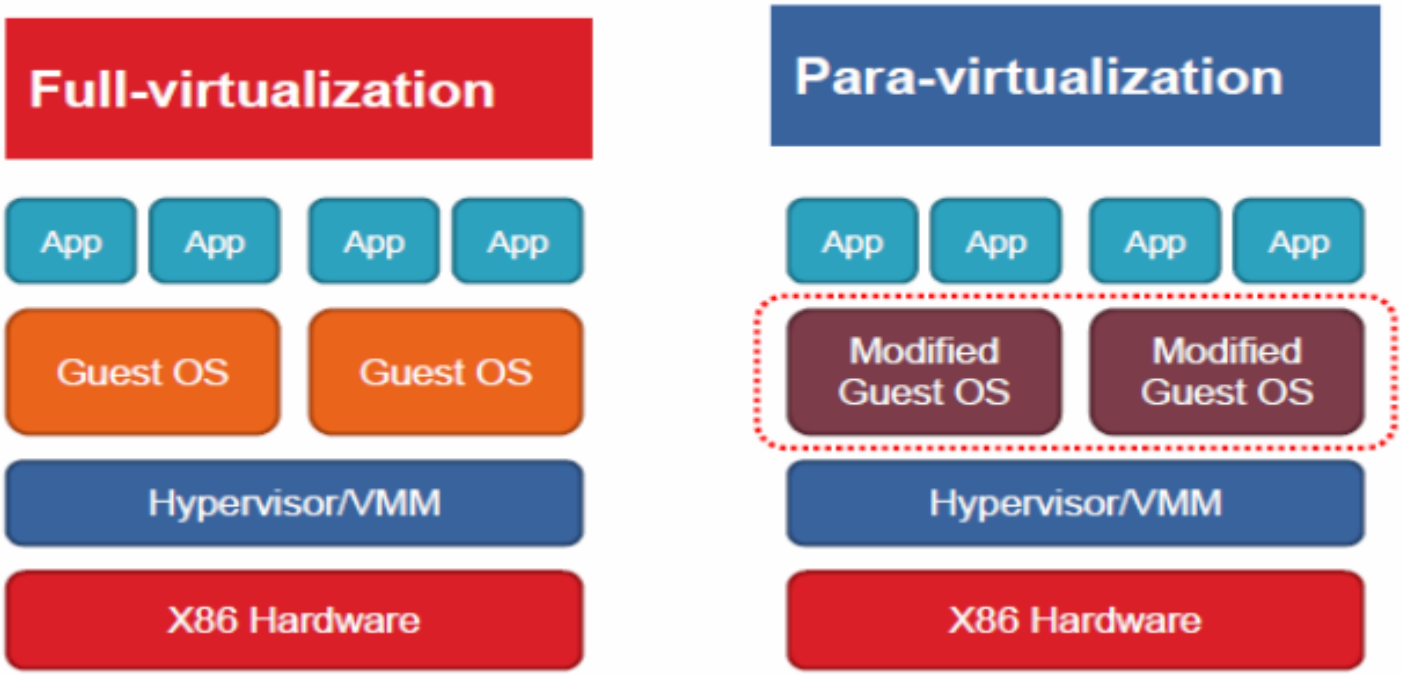
- Here, the hardware is not simulated; instead, the guest software runs its isolated system

Full Virtualization:

- Here, the virtual machine simulates the hardware & is independent (**Binary translation/Emulation)**
- Furthermore, the guest OS doesn't require any modification.

Virtualization Techniques

Full vs. para virtualization



Virtualization Techniques

- How to virtualize unvirtualizable hardware :
  - Para-virtualization
    - Modify guest OS to skip the critical instructions.
    - Implement some hyper-calls to trap guest OS to VMM.
  - Binary translation
    - Use emulation technique to make hardware virtualizable.
    - Skip the critical instructions by means of these translations.
  - Hardware assistance
    - Modify or enhance ISA of hardware to provide virtualizable architecture.
    - Reduce the complexity of VMM implementation.

Para-Virtualization

- Para-Virtualization implementation :
  - In para-virtualization technique, guest OS should be modified to prevent invoking critical instructions.
  - Instead of knowing nothing about hypervisor, guest OS will be aware of the existence of VMM, and collaborate with VMM smoothly.
  - VMM will provide the hyper-call interfaces, which will be the communication channel between guest and host.

Binary Translation

- In emulation techniques :
  - Binary translation module is used to optimize binary code blocks, and translate binaries from guest ISA to host ISA.
- In virtualization techniques :
  - Binary translation module is used to skip or modify the guest OS binary code blocks which include critical instructions.
  - Translate those critical instructions into some privilege instructions which will trap to VMM for further emulation.

Some Difficulties

- Difficulties of binary translation :
  - Self-modifying code
    - If guest OS will modify its own binary code in runtime, binary translation need to flush the responding code cache and retranslate the code block.
  - Self-reference code
    - If guest code need to reference(read) its own binary code in runtime, VMM need to make it referring back to original guest binaries location.
  - Real-time system
    - For some timing critical guest OS, emulation environment will lose precise timing, and this problem cannot be perfectly solved yet.
- Difficulty of para-virtualization :
  - Guest OS modification
    - User should at least has the source code of guest OS and modify its kernel; otherwise, para-virtualization cannot be used.

Hardware Solution

- Let's go back to trap model :
  - Some trap types do not need the VMM involvement.
    - For example, all system calls invoked by application in guest OS should be caught by gust OS only. There is no need to trap to VMM and then forward it back to guest OS, which will introduce context switch overhead.
  - Some critical instructions should not be executed by guest OS.
    - Although we make those critical instructions trap to VMM, VMM cannot identify whether this trapping action is caused by the emulation purpose or the real OS execution exception.

- Solution :
  - We need to redefine the semantic of some instructions.
  - We need to introduce new CPU control paradigm.

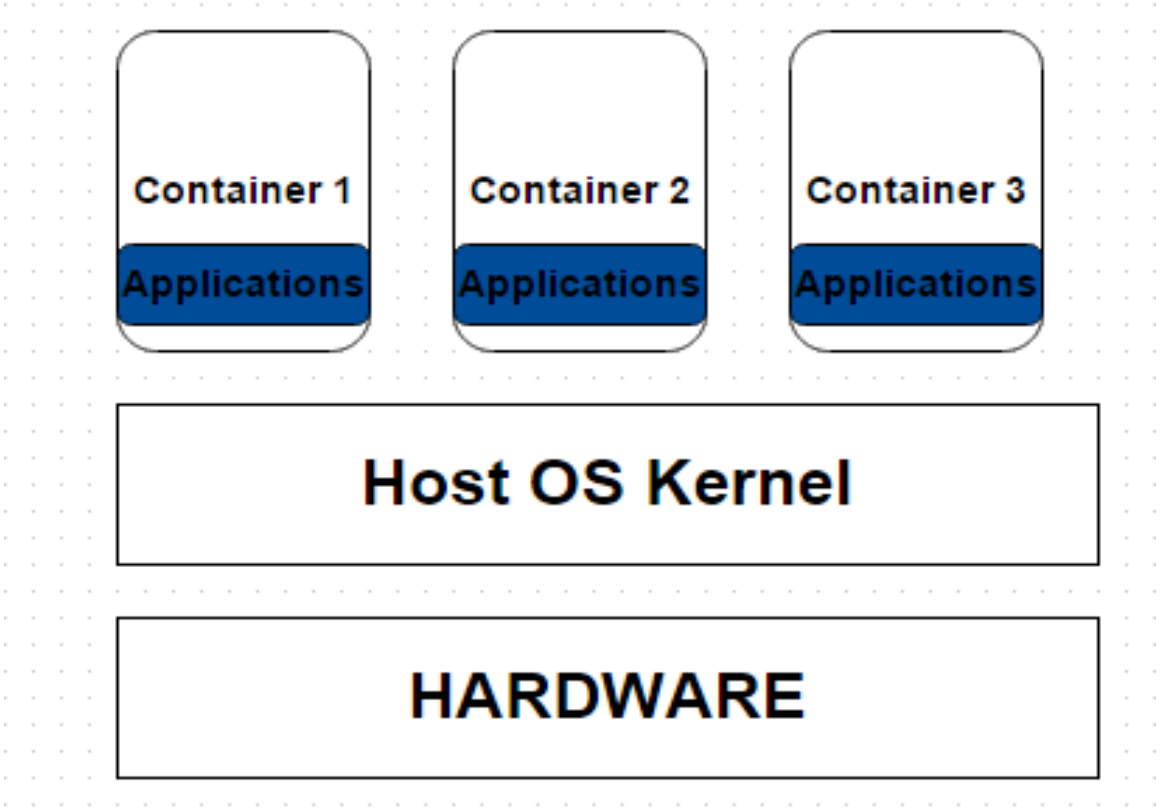
OS Virtualization

- Virtualization technology which work on OS layer
- Here the kernel of an OS allows more than one isolated user-space instances to exist.
- Such instances are called containers/software containers or virtualization engines.
- In other words, OS kernel will run a single operating system & provide that operating system's functionality to replicate on each of the isolated partitions.

Containers vs VMs: What's the difference?

https://www.youtube.com/watch?v=cjXI-yxqGTI

OS Virtualization



References

- *Cloud Computing*, Sandeep Bhowmik
- Containers vs VMs: What's the difference? https://www.youtube.com/watch?v=cjXI-yxqGTI
- Server Virtualization: https://slideplayer.com/slide/5103233/