Virtualization Techniques

Full Virtualization:

- Here, the virtual machine simulates the hardware & is independent (**Binary translation/Emulation)**
- Furthermore, the guest OS doesn't require any modification.
- VM simulates “enough” hardware to allow an unmodified guest OS to be run in isolation

Para-Virtualization/ OS assisted:

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

Hypervisors

Full Virtualization vs. Para-Virtualization

Full virtualization

- Does not need to modify guest OS, and critical instructions are emulated by software through the use of binary translation.
- VMware Workstation applies full virtualization, which uses binary translation to automatically modify x86 software on-the-fly to replace critical instructions.

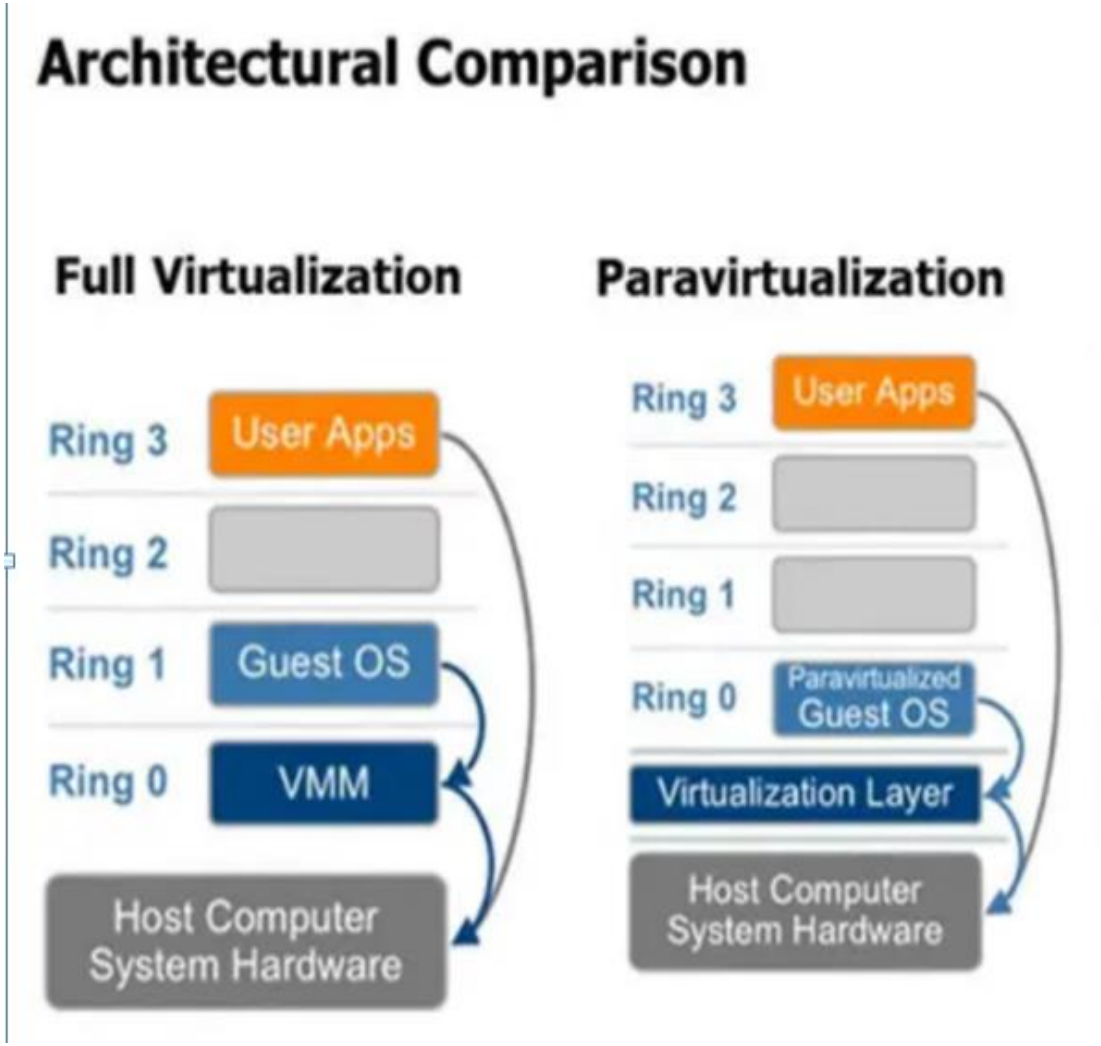
Advantage: no need to modify OS.

Disadvantage: binary translation slows down the performance.

Para virtualization

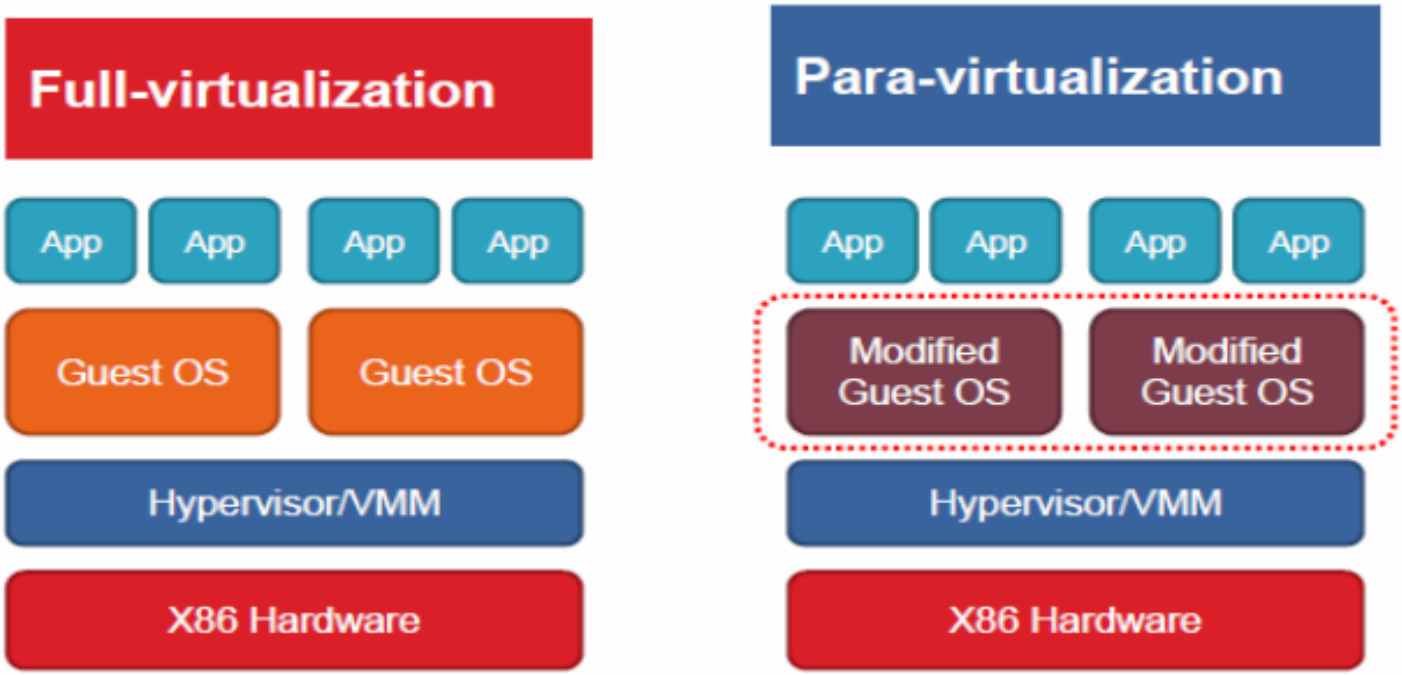
- Reduces the overhead, but cost of maintaining a paravirtualized OS is high.
- The improvement depends on the workload.
- Para virtualization must modify guest OS, non-virtualizable instructions are replaced by hyper calls that communicate directly with the hypervisor or VMM.
- Para virtualization is supported by Xen, Denali and VMware ESX.

Virtualization Techniques



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.

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.

References

Cloud Computing, Sandeep Bhowmik

Cloud computing Black Book by Kailash Jayaswal

https://data-flair.training/blogs/virtualization-in-cloud-computing/

https://www.w3schools.in/cloud-computing/cloud-virtualization/

https://www.slideshare.net/karimalinani/virtualization-4228802

Virtualization in Cloud Computing, Intellipaat

https://www.youtube.com/watch?v=_pPlanX5wQY&t=20s

Virtualization and Cloud Computing Lecture 4: Hardware-assisted CPU virtualization in KVM/QEMU

https://www.youtube.com/watch?v=wiSwGRjJN-w

Virtualization and Cloud Computing Lecture 5: Full Virtualization

https://www.youtube.com/watch?v=MVxBxg_aNk0