Computing Resources in Infrastructure

Computing Resources

- CPU + Main Memory (RAM)

Computer Machine / Server

- CPU + Main Memory + Disk Storage (OS), NIC, Accelerators (GPU, FPGA, etc.), High Performance Computing (AI, DNN)

Physical Machine
Virtual Machine

Physical Machine

Hardware

- OS ควบคุมการทำงานของ hardware
- OS run on physical or virtual ก็ได้

CPU

- Socket (Physical Package) ถอดออกง่าย เปลี่ยนง่าย ถ้าไม่มี socket ก็ต้องบักรีลง main board (เปลี่ยนยาก)
- Core (Circuit Component) -->Register + ALU/Computing Unit + Control Unit --> วงจร electronic บน chip silicon
- Thread --> กิจกรรมการ execute คำสั่งที่ **กำลัง** เกิดขึ้นใน CPU (เก็บสถานะใน Register) (sometimes call Virtual CPU) (บางทีใช้งาน ALU หรือ Control ร่วมกันระหว่าง Thread หลายอันได้)
- Instruction Set Architecture (ISA) (Machine Code) --> Depends on CPU design (x86/64/amd64, ARM)

socket --> silicon --> core

Memory / RAM, Storage / Disk, NIC, Computing Device (GPU, FPGA), Others

Virtual Machines

รวม --> หลาย physical to 1 virtual (Ex. Cluster แต่ไม่ตรง 100% แต่บะเครื่องยังอิสระต่อกัน)
แบ่ง--> 1 physical to หลาย virtual (ปัจจุบัน)

use physical machine / HW

Virtualization Software

- VMM (Virtual Machine Monitor) (ไม่เรียกชื่อนี้แล้ว)
- Hypervisor (ปัจจุบัน) --> สร้าง virtual machine ได้หลายตัว

Each VM --> CPU, RAM, Disk / Storage, NIC, GPU, Hardware (may pass-through (Direct access))

virtual machine ใช้ hardware แบบ pass through แล้ว ตัวอื่นจะมาใช้ด้วยไม่ได้ ใช้ได้คนเดียว

Virtual Machine Technology

- Virtual Server (Server Side)
- Virtual Desktop Infrastructure (VDI)
  - ประมวลผลออกจาก user interface
  - แยก execute กับ user interface
  - จอ client จะเห็นแค่ interface คือแค่ remote ส่วน execute ที่คอม ปลายทาง
  - literally just remote access
  - Client / User Mechanics
  - At Server / Center
  - Using Remote Desktop Software Connect to VMs at Servers

2 Types of Hypervisor

- หลาย VM รันบน physical เครื่องเดียว
- physical OS and hardware --> Host, VM OS and Apps --> Guest

Type 1 Bare-Metal

- Hypervisor คุม hardware โดยตรง (Ex. VMware ESX, Microsoft Hyper-V, Xen)

Type 2 Hosted

- Hypervisor จะเรียกใช้ hardware ผ่าน OS อีกทีหนึ่ง
- ทำให้ hypervisor เปรียบเสมือน app ชนิดนึง
- มี Process อื่นๆ รันอยู่บน Host ควบคู่ได้ด้วย
- Easy Management, Avoid code duplication (process scheduler, memory management)
- (Ex. VMWare Workstation, Microsoft Virtual PC, Sun VirtualBox, QEMU, KVM)

Combined

- Mostly Hosted แต่บางอย่างไปอยู่ใน OS Kernel เพื่อแก้ปัญหา Performance (Ex. KVM)

VM Techniques

Emulation (or Simulation)

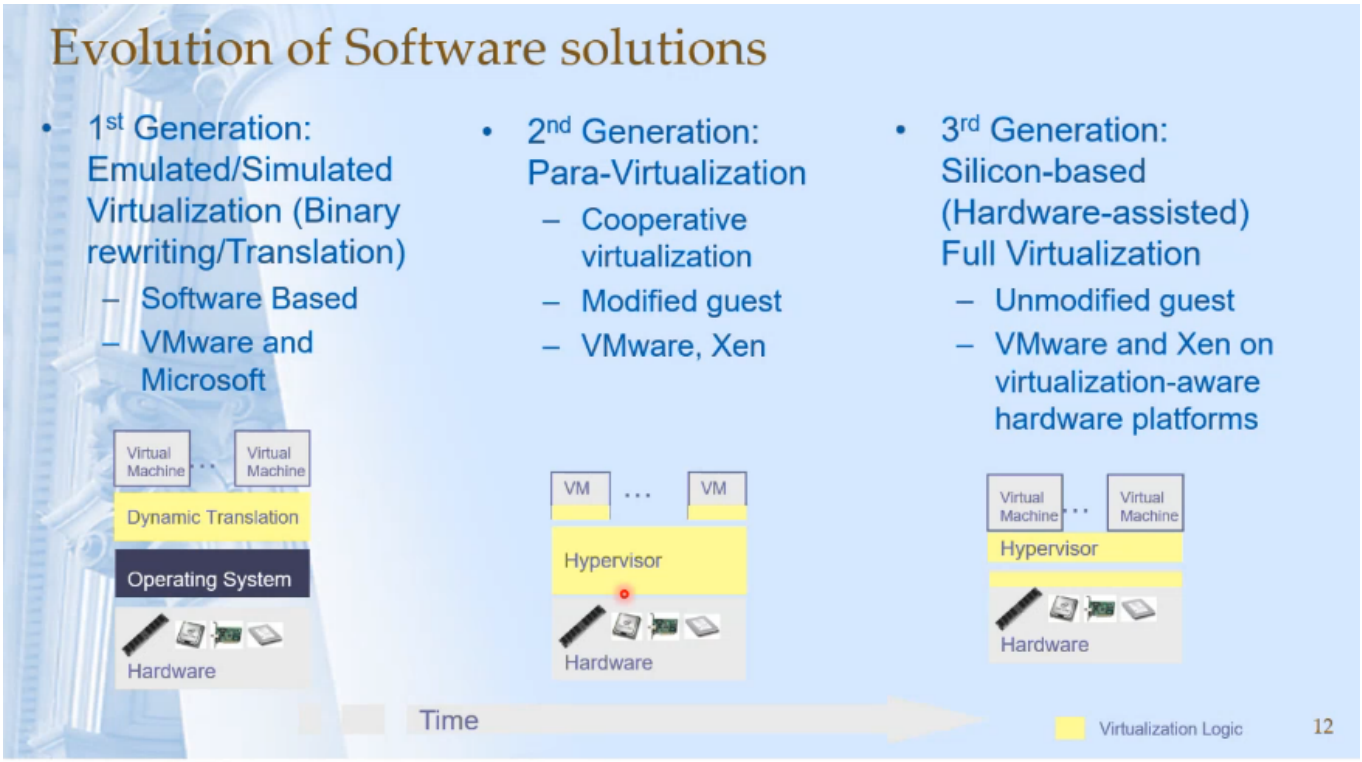
- ใช้กรณี ใช้ใน CPU / Hardware ที่แตกต่างกัน
- จำลองการทำงานของ Virtual Machine บน Software
- Decode and Execute Instructure by Software
  - Fetch Instructions
  - Decode - is it ADD, XOR or MOV
  - Execute - using the emulated registered and memory
- Pro: Simple
- Cons: Slow
- Example: Old game console emulation

Para-Virtualization

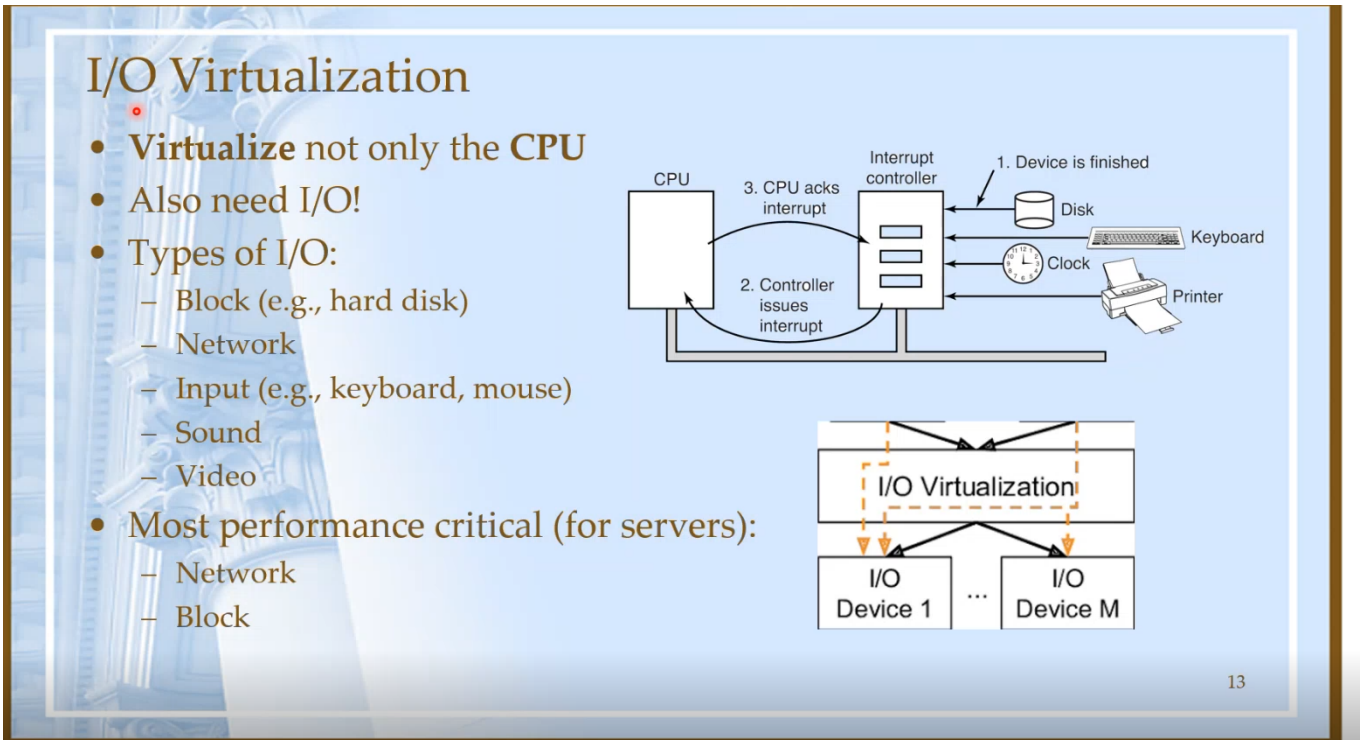
- Does not run unmodified guest OSes
- Require Guest OS to "know" it is running on top of hypervisor
- Guest co-operate with Hypervisor
  - Modified or additional software
  - ถ้า CPU ว่างก็บอก Hypervisor ว่าว่าง คนอื่นจะได้ใช้ได้ เพื่อปะสิทธิภาพสูงสุด
- Pros: No hardware support required, Performance better than emulation
- Cons: Requires specifically modified guest, same guest OS cannot run in the VM and bare-metal
- Example Hypervisor: Xen

Full Virtualization:

- Hardware assisted
- VM simulate enough hardware for unmodified Guest OS to run on isolation
- Designed for the same CPU (Host and Guest)
- Pro: Performance and Control
- **Current Technology**
- Ex. VMware Workstation, VMware Server (ESX Server)
- Hardware especially CPU ถูกออกแบบมาให้รองรับการสร้าง Virtual Machine
- Hypervisor ควบคุม VM ได้เด็ดขาด
- CPU ต้องถูกออกแบบให้มีการควบคุมสิทธิการเข้าใช้ CPU แบบหลายลำดับชั้น
- ยึด CPU, Hardware บางส่วนไปเลย (Virtual Resurces)



I/O Virtualization



Emulate

- **Hypervisor** สร้าง **Virtual NIC (Network Interface Card)** ตามสเปคของ NIC จริง เช่น Intel, Realtek, หรือ Broadcom
- **Register ของ NIC** จะถูกแทนด้วยตัวแปรในหน่วยความจำของ **Hypervisor (host)**

- เมื่อ **Physical NIC** เกิดการส่งสัญญาณขัดจังหวะ (TX complete - ส่งข้อมูลเสร็จสิ้น) Hypervisor จะส่งสัญญาณขัดจังหวะนี้ไปยัง **Guest VM**
- **ข้อดี**: รองรับ **Guest** ที่ไม่ได้ถูกปรับแก้ไข (Unmodified guest)
- **ข้อเสีย**: ทำงาน **ช้า** เนื่องจากทุกครั้งที่ Guest VM เข้าถึง register ของ NIC จะทำให้เกิด **VM exit** (การส่งไปยัง Hypervisor)
    - Hypervisor ต้องจำลองฮาร์ดแวร์ที่ซับซ้อน (emulate complex hardware)
- ตัวอย่าง Hypervisor ที่รองรับการจำลองแบบนี้:
    - **QEMU**, **KVM**, **VMware** (หากไม่ได้ติดตั้ง VMware Tools)

Para-Virtualization

- เพิ่ม **virtual NIC driver** เข้าไปใน **Guest (frontend)**
- สร้าง **virtual NIC** ใน **Hypervisor (backend)**
- ใช้ **protocol** ในการติดต่อระหว่าง **frontend** และ **backend**
- **Paravirtual protocol**:
    - **Guest** เรียกใช้ **hyper calls** โดยส่งที่อยู่เริ่มต้นและความยาวเป็นอาร์กิวเมนต์
    - **Hypervisor** จะรู้ว่าต้องทำอะไร (Hypervisor knows what it should do)
    - โปรโตคอลแบบ Para-Virtualization สามารถทำงานได้ในระดับสูง (high-level)
- **ข้อดี**:
    - **รวดเร็ว** (Fast) – ไม่จำเป็นต้องจำลองฮาร์ดแวร์จริง (no need to emulate physical device)
- **ข้อเสีย**:
    - ต้องการ **Guest driver**
- ตัวอย่าง **Hypervisor** ที่รองรับ:
    - **QEMU**, **KVM**, **VMware (with VMware Tools)**, **Xen**

Direct Access / Direct Assignment

- ดึง (**Pull**) NIC ออกจาก **Host** แล้วเชื่อมต่อ (**Plug**) เข้ากับ **Guest** โดยตรง
- **Guest** สามารถเข้าถึง **NIC registers** ได้โดยตรงโดยไม่ต้องผ่าน **Hypervisor**
- **Host** จะไม่สามารถเข้าถึง **NIC** ได้อีกต่อไป
- **ข้อดี**: เร็วที่สุดเท่าที่เป็นไปได้ (**As fast as possible!**)
- **ข้อเสีย**:
    - ต้องการ **NIC แยกต่อ Guest 1 ตัว**
    - และต้องมี **NIC อีกตัวสำหรับ Host**
    - ไม่สามารถทำฟีเจอร์พิเศษได้ เช่น
        - การ encapsulate (ครอบห่อ) packet ของ Guest
        - การตรวจสอบ (monitor) และแก้ไข packet ในระดับ Hypervisor
- ตัวอย่าง Hypervisor ที่รองรับ:
    - **KVM**, **Xen**, **VMware**

Single Root I/O Virtualization - SR-IOV Standard

- ประกอบด้วย Physical Function ที่ควบคุมโดย Host ใช้สร้าง Virtual Functions
- แต่ละ Virtual Functions ถูกกำหนดให้กับ Guest (คล้ายกับการกำหนดแบบตรง)
- Guest แต่ละตัวคิดว่าตนเองมีการควบคุม NIC อย่างเต็มที่ และเข้าถึง registerโดยตรง
- NIC ทำหน้าที่มัลติเพล็กซ์/ดีมัลติเพล็กซ์การรับส่งข้อมูล

ข้อดี:

- เร็วที่สุดเท่าที่จะเป็นไปได้!
- ต้องการ NIC เพียงตัวเดียว (ต่างจากการกำหนดแบบตรง)

ข้อเสีย:

- เป็นมาตรฐานที่เพิ่งเกิดขึ้น
- ไฮเปอร์ไวเซอร์ส่วนน้อยที่รองรับอย่างเต็มที่
- ต้องการการรองรับจากฮาร์ดแวร์
- ไม่สามารถทำ "สิ่งเจ๋งๆ" ได้

ตัวอย่างไฮเปอร์ไวเซอร์: KVM, Xen, VMware

Comparison

- SR-IOV เร็วสุด แต่แพงสุด
- Paravirtual I/O ถูก แต่ประสิทธิภาพ แย่กว่า

Conclusion

- Computing Resources for Infrastructure
  => Computer Machines
- Physical Machine => Hardware
- Many Virtual Machines use one or many Physical Machines
- Virtual Machine => Servers or Desktops
- Hypervisor Makes physical into virtual machines
  - Type 1 – Bare-Metal vs Type 2 – Hosted
- Virtualization Techniques
  - Translate vs Para-Virtual. vs HW-Assisted