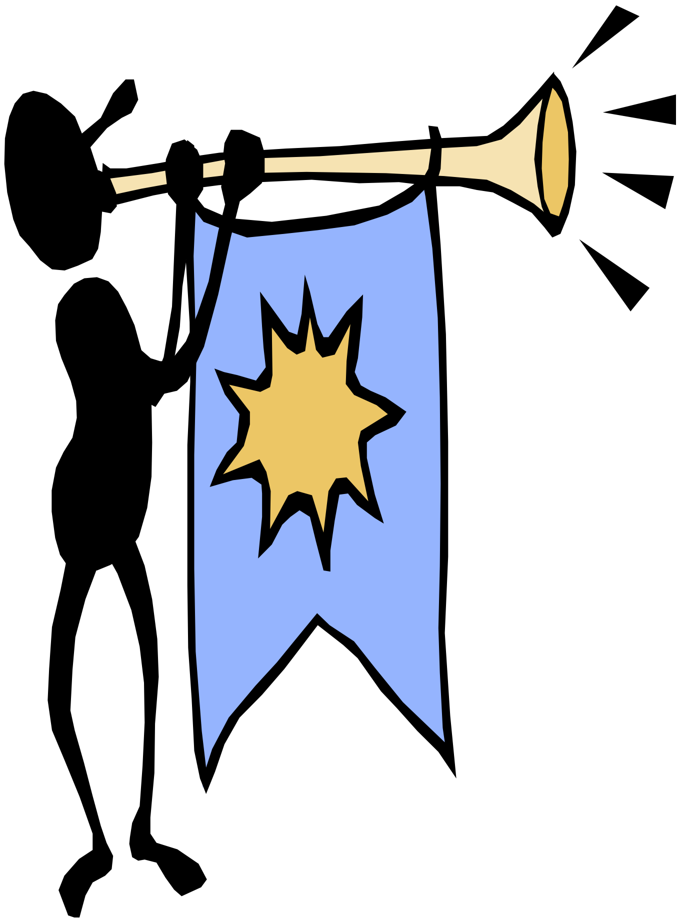
Virtualization Technologies

Roch Glitho, PhD
Full Professor
Ericsson / ENCQOR-5G Senior Industrial Research Chair
Cloud and Edge Computing for 5G and Beyond
My URL - http://users.encs.concordia.ca/~glitho

Concordia University
Engineering and Computer Science
Concordia Institute for Information Systems Engineering

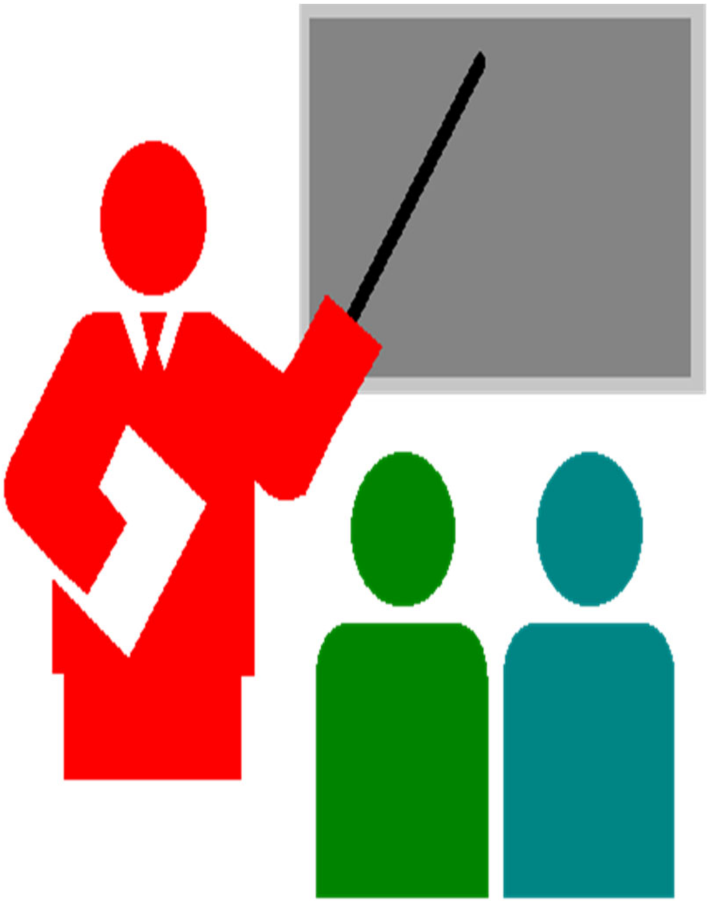
Outline



1. Hypervisor based – virtualization
2. Containers
3. Uni-kernel
4. Virtualization and serverless computing

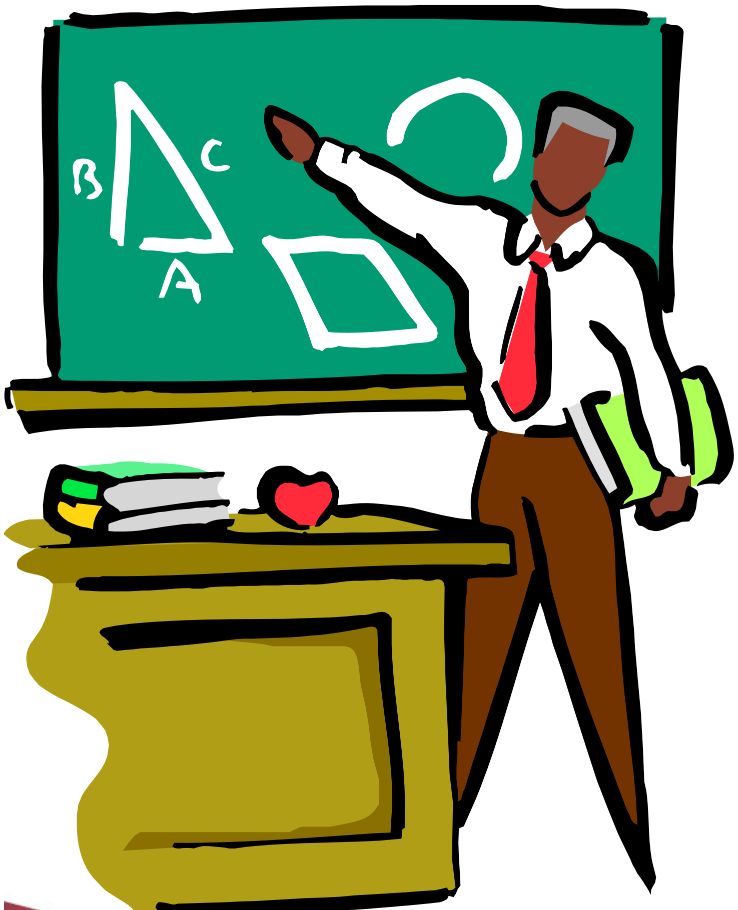
Hypervisor Based - Virtualization

On Systems Virtualization



- Key concepts
- Type I (bare metal) vs. Type 2 (hosted)
- Full virtualization vs. para-virtualization

Basic concepts



1. On operating systems

2. Virtual machine, hypervisor

4. Examples of benefits

Operating systems

Some of the motivations

- Only one single tread of CPU can run at a time on any single core consumer machine
- Machine language is tedious

Operating systems

Operating systems bring a level of abstraction on which multiple processes can run at a time – Deal among other things with:

- Multiplexing
- Hardware management issues

However only one operating system can run on a bare single core consumer machine

virtual machines and hypervisors

- Systems virtualization dates back to the 60s
- IBM experimentation with “time sharing systems”
  - Need for virtual machines to test how applications / users can time share a real machine

virtual machines and hypervisors

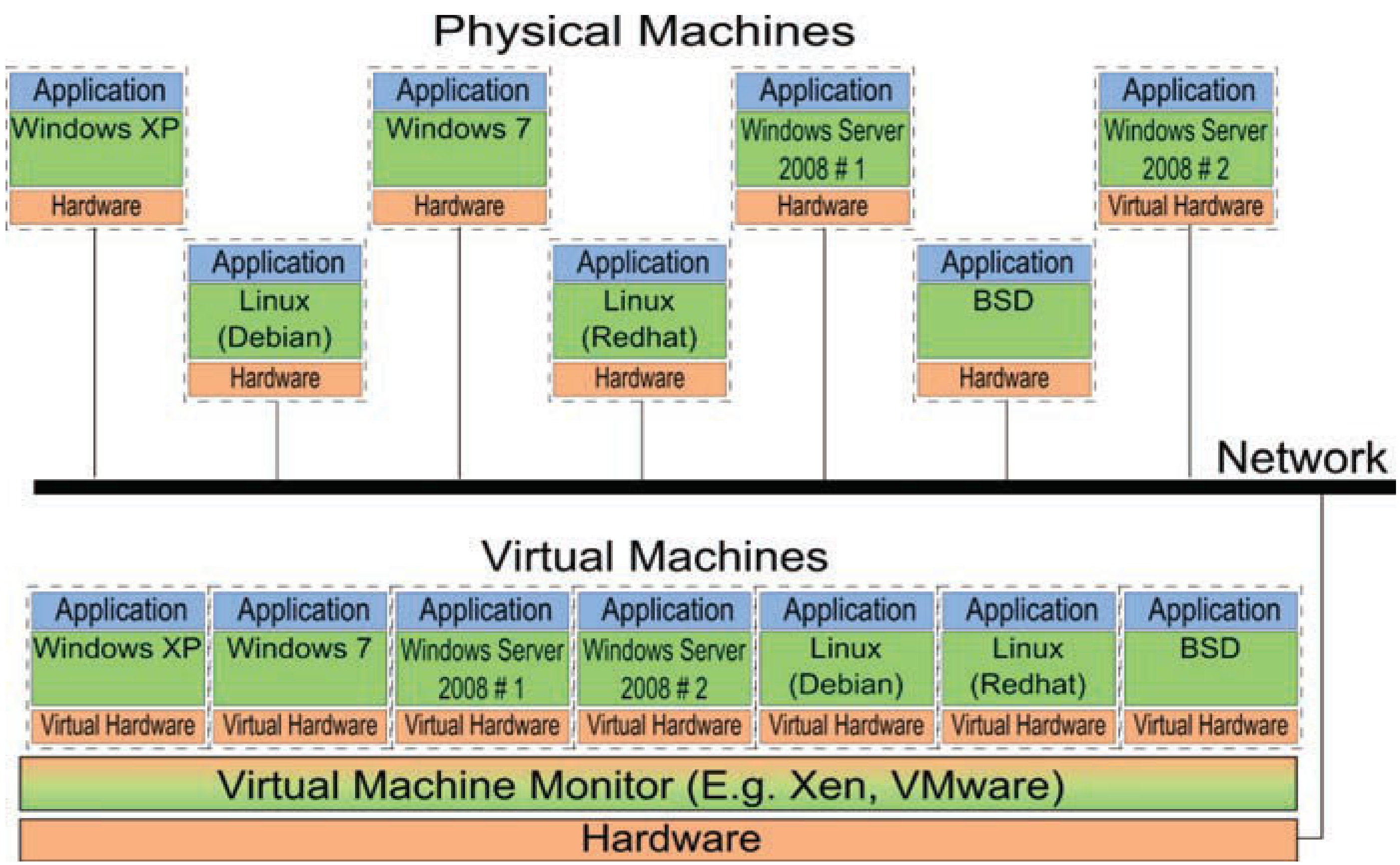
Virtual machine (VM)

- Software that provides same inputs / outputs and behaviour expected from hardware (i.e. real machine) and that supports operations such as:
  - Create
  - Delete
  - Migrate
  - Increase resources

Hypervisor

- Software environment that enables operations on virtual machines (e.g. XEN, VMWare) and ensures isolation

virtual machines, hypervisors



1. M. Pearce et al., Virtualization: Issues, Security, Threats, and Solutions, ACM Computing Survey, February 2013

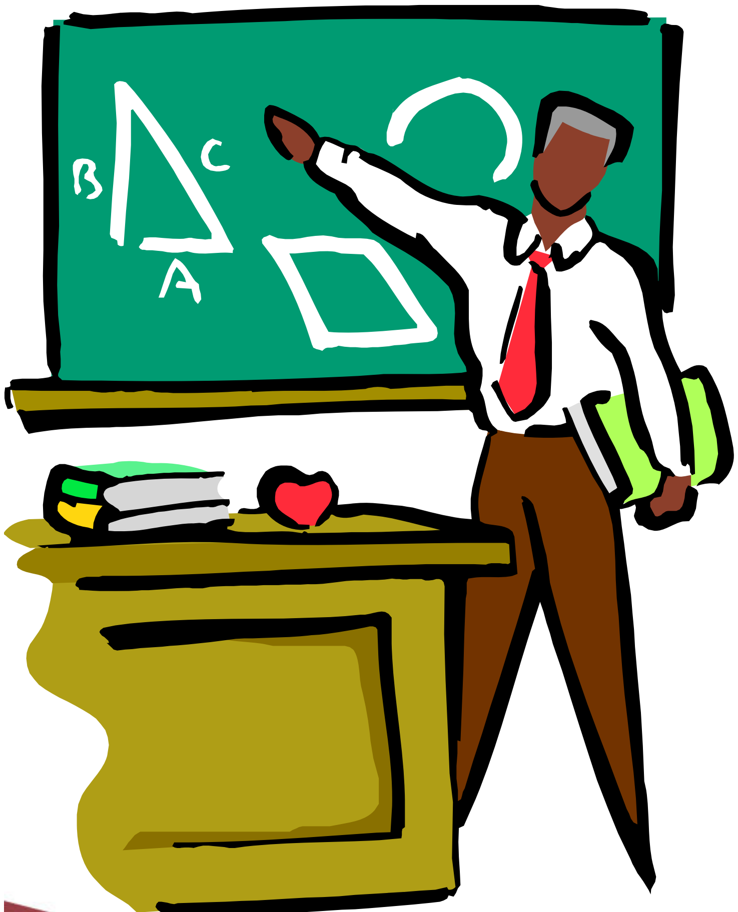
From reference [1] – Note: There is a small error in the figure

Examples of Benefits

All benefits are due to the possibility to manipulate virtual machine (e.g. create, delete, increase resources, migrate), e.g.

- Co-existence of operating systems
- Operating systems research
- Software testing and run-time debugging
- Optimization of hardware utilization
- Job migration

Advanced concepts



1. Bare metal vs. hosted hypervisor

2. Full virtualization vs. Para-virtualization

Type I vs Type II Hypervisor

Types of hypervisor

- Type I – bare metal
  - Installed on bare hardware
  - Examples
    - Citrix XEN server
    - VMWARE ESX/ESXI

Type I vs Type II Hypervisor

Types of hypervisor

- Type 2 – hosted
  - Runs on top of host operating system
  - Examples:
    - VMWare workstation
    - VirtualBox

Type I vs Type II Hypervisor

Type I - Bare metal

- Hypervisor installed on bare hardware
  - Advantages (compared to type II)
    - Performance (No additional software layer to go through)
    - Security (No possible attack through host operating system)
  - Drawbacks (compared to type II)
    - Host operating system needs to be "ported" on top of hypervisor
    - Complexity depends on the type of virtualization (Full virtualization vs. para-virtualization)

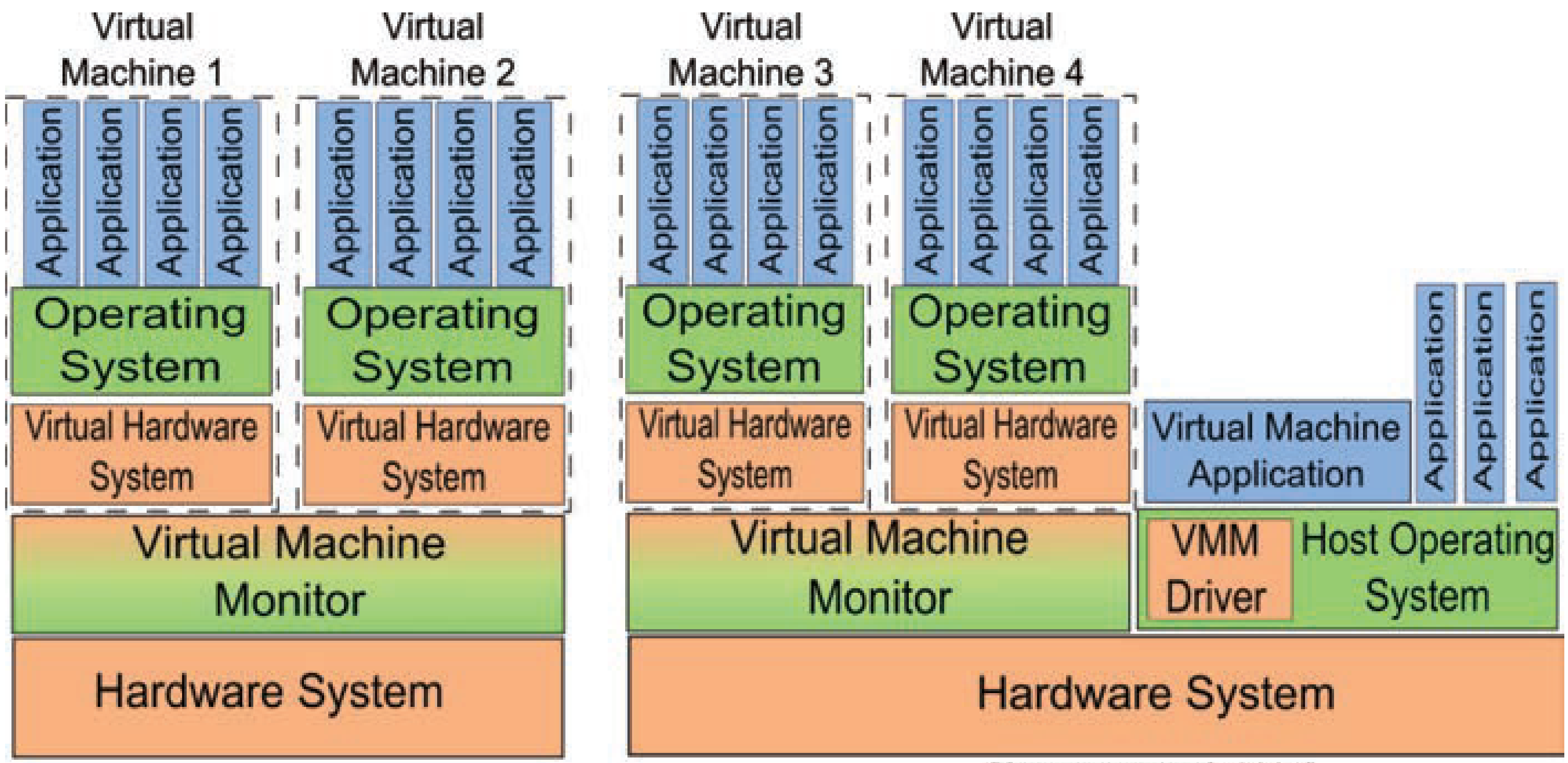
Type I vs Type II Hypervisor

Type II - Hosted

- Hypervisor installed on top of host operating system
  - Drawbacks (compared to type I)
    - Performance (need to go through host operating system)
    - Security (i.e. Possibility to attack through host operating system)
  - Advantages (compared to type I)
    - Host operating system is re-used as it is (No need to port it)
    - No change required to applications running on top of host operating system

Type I vs Type II Hypervisor (Summary)

Types of hypervisor/virtual machine monitor (From ref. 2)



[Sugerman et al. 2001]

Full virtualization vs. Para-virtualization

More on operating systems fundamentals

- Privileged vs. non privileged instruction
  - Privileged
    - If called in user mode, the CPU needs to trap it and switch control to supervisory software (e.g. hypervisor) for its execution

Full virtualization vs. Para-virtualization

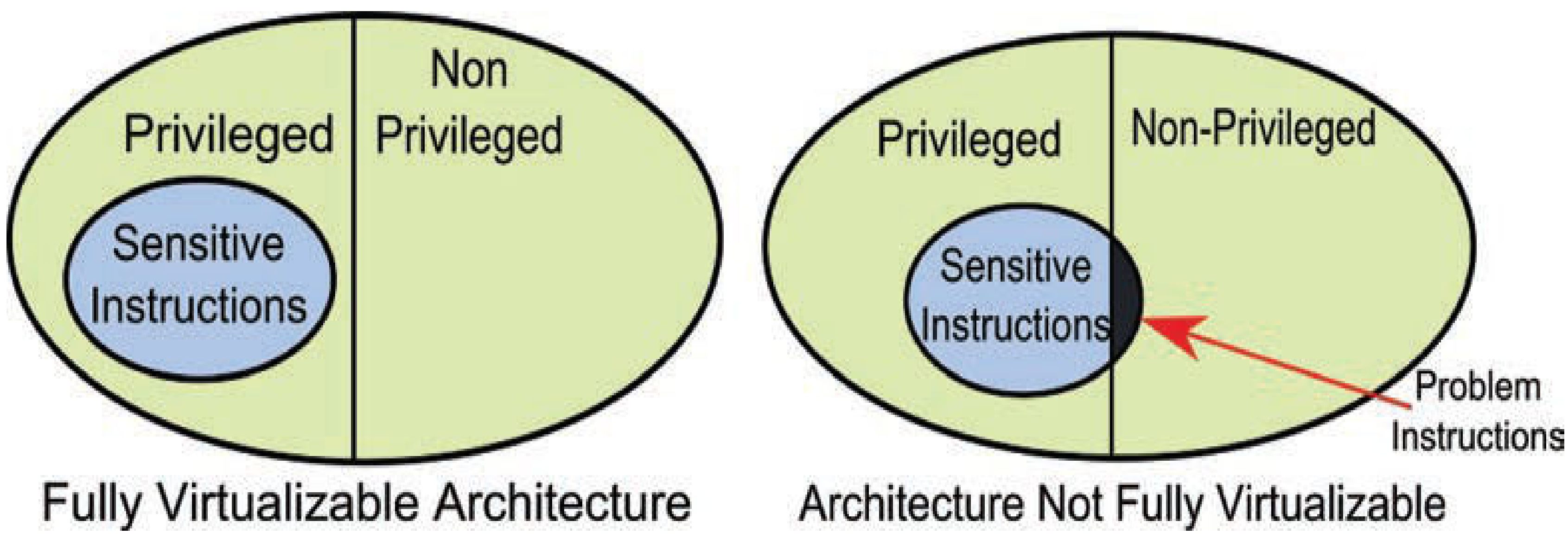
More on operating systems fundamentals

- Sensitive vs. non sensitive instruction
  - Sensitive
    - Has the capacity to interfere with supervisor software functioning (e.g. Hypervisor)
      - Write hypervisor memory vs. read hypervisor memory

Full virtualization vs. Para-virtualization

Could all CPU architectures be fully virtualized ?

- Could be fully virtualized only if the set of sensitive instructions is a subset of the privileged instructions



From reference [1]

Full virtualization vs. Para-virtualization

Could all CPU architectures be fully virtualized ?

- The case of Intel x86 CPU architectures
  - Cannot be fully virtualized
    - "Certain instructions must be handled by the VMM for correct virtualization, but these with insufficient privilege fail silently rather than causing a convenient trap" – Reference [2]

Full virtualization vs. Para-virtualization

Definitions

Full virtualization

- Hypervisor enables virtual machines identical to real machine
  - Problematic for architectures such as Intel x86

Full virtualization vs. Para-virtualization

Definitions

Para-virtualization

- Hypervisor enables virtual machine that are similar but not identical to real machine
  - A solution to the problem of CPU architectures that cannot be virtualized
    - Prevents user programs from executing sensitive instructions
  - Note:
    - Para-virtualization is not the only solution to the problem

Full virtualization vs. Para-virtualization

Full virtualization

- Advantages
  - Possibility to host guest operating systems with no change since virtual machines are identical to real machines
- Disadvantages
  - Not always feasible (e.g. Intel x86)
    - There are work around (e.g. binary translation)
  - Some guest operating systems might need to see both virtual resources and real resources for real time applications

Full virtualization vs. Para-virtualization

Para - virtualization

- Advantages
  - Feasible for all CPU architectures
  - Performance – Compared to:
    - Full virtualization
    - Other approaches to architectures that could not be virtualized (e.g. binary translation)
- Disadvantages
  - Need to modify guest operating systems

Full virtualization vs. Para-virtualization

Para - virtualization

- Alternatives to para-virtualization
  - Binary translation (e.g. VMWare ESX server)
    - Leads to full virtualization
    - No need to re-write “statically” guest operating systems
      - i.e. guest OS can be installed without change
    - Interpretation of guest code (OS + application)
      - “Rewrites” dynamically guest code and insert traps when necessary

Full virtualization vs. Para-virtualization

Para - virtualization

- Alternatives to para-virtualization
  - Binary translation
    - Disadvantages / penalties
      - Performance
      - However, optimization is possible, e.g.
        - » Adaptive translation (i.e. optimize the code being translated)

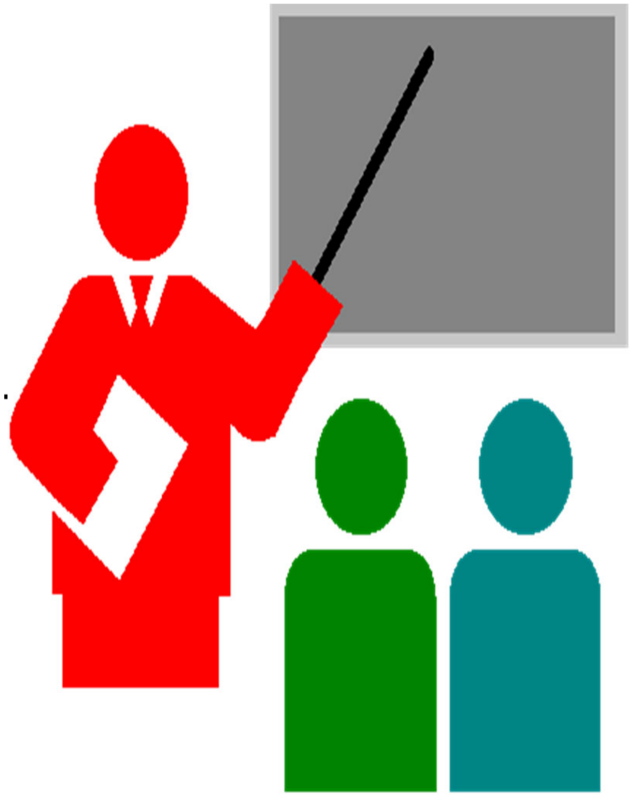
Full virtualization vs. Para-virtualization

Para – virtualization

- A detailed case study on para-virtualization
  - XEN (Reference 2)

Alternatives to Hypervisor Based - Virtualization

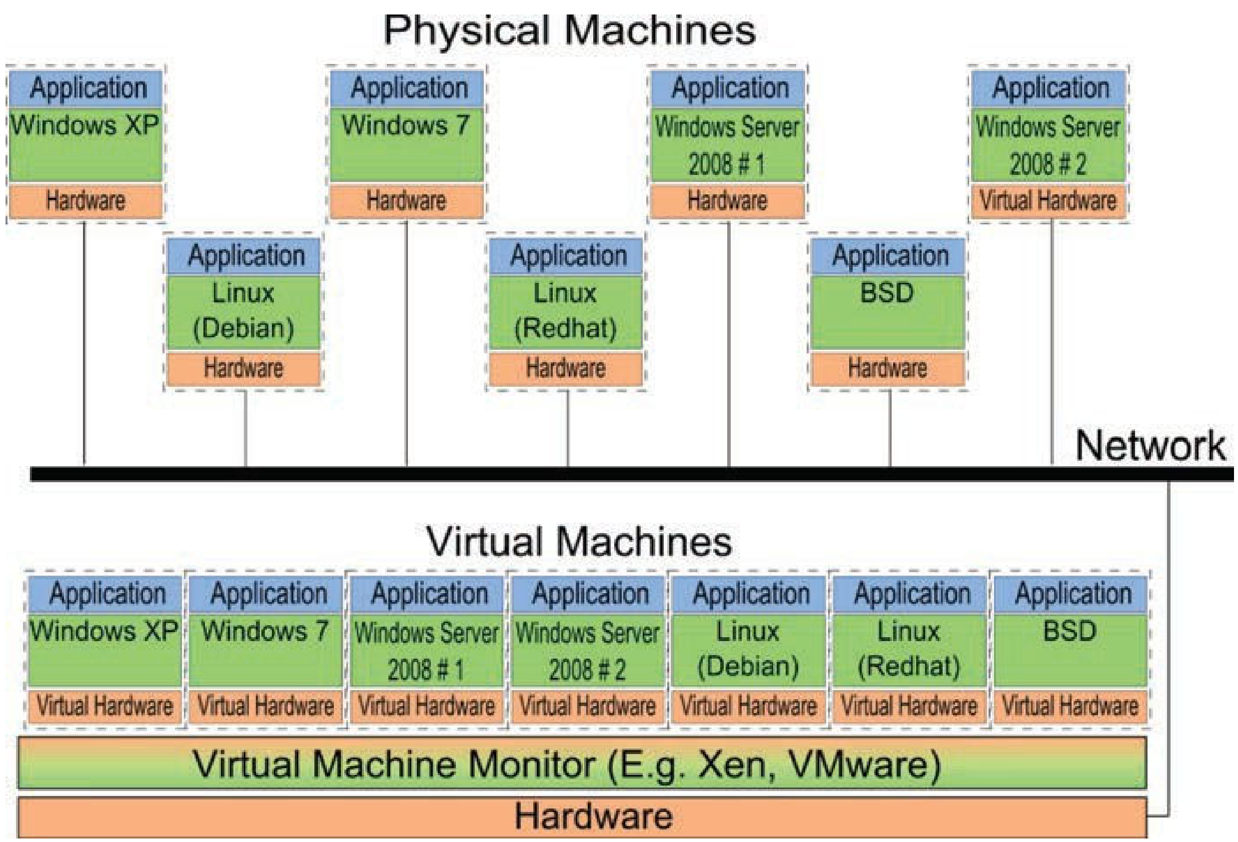
Containers and Unikernels



- Issues with hypervisors

- Alternatives (Containers and unikernels)

virtual machines, hypervisors



From reference [1] – Note: There is a small error in the figure

Hypervisor

In a hypervisor based – approach, a VM includes the application + full blown operating system (e.g. Linux Debian, Linux Red Hat)

- OS on virtual machine needs to boot
  - Slow starting time for application

.

- Resources are not used in an efficient manner
  - Linux kernel replicated in each VM that runs linux.

Proposed Solutions

Back to operating systems basics

- The two components of an operating system
  - Kernel
    - Interacts with the hardware and manages it (e.g. write/read a disk partition)

  - Librairies
    - Set of higher level functions accessible to programs via system calls
      - Enable function like create / read / delete file while hiding the low level operations on the hard disk

Alternatives

VM vs container vs Unikernel

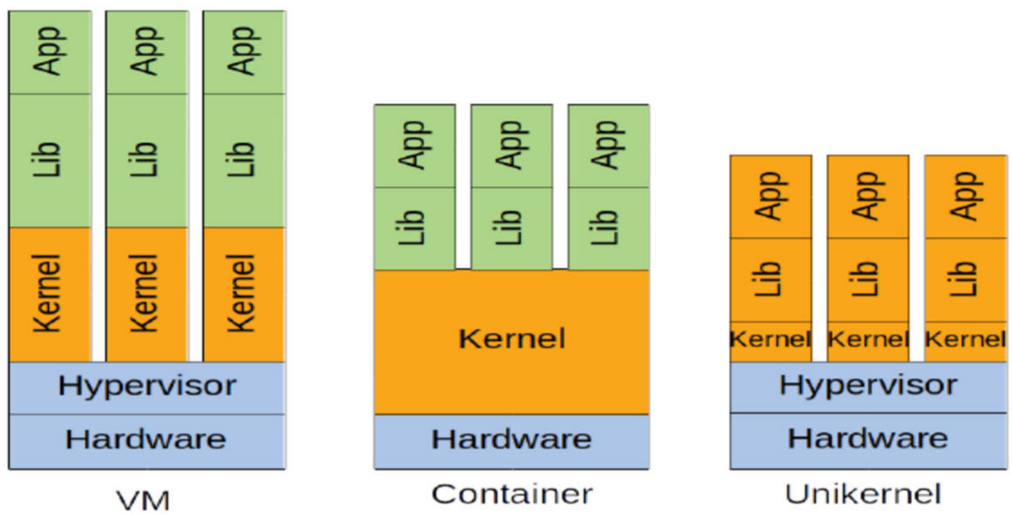


Fig. 1. Comparison of virtual machine, container and unikernel system architecture

T. Goethals et al., Unikernels vs. Containers: An In-Depth Benchmarking Study in the Context of Microservice Application, IEEE SC2 Conference, November 2018

On containers

Operating system (Kernel) virtualization:

- Kernel offers isolated spaces to run containers
  - Containers
    - » Applications packaged with their run time environment that run on a same kernel .
    - » Run as processes, but with isolated file system, networking, CPU and memory resources

On containers

Operating system (Kernel) virtualization:

- Kernel offers isolated spaces to run containers
  - Containers
    - » Hosted by container engine (e.g. Docker Engine)
    - » Need to be deployed, managed and orchestrated (e.g. Kubernetes)

On containers

Operating system (Kernel) virtualization:

- **Kernel offers isolated spaces to run containers**
  - **Some pros / cons**
    - Less memory footprint
      - » Do not include kernel
    - Faster start up time
      - » Kernel does not need to boot

On containers

Operating system (Kernel) virtualization:

- **Kernel offers isolated spaces to run containers**
  - **Some pros / cons**
    - Works only in environments in which you have given operating system kernel + its libraries (e.g. Linux kernel + Linux distributions)
    - Less secure than VM
      - » Challenge:
        - » Trade-off between isolation and performance / efficiency

On Unikernels

Application + Tiny run time:

- **Tiny run time**
  - Not the whole OS like VM
  - Not the whole libraries like containers
    - » Only the function required by the applications
    - » Static binding
  - Can run as a tiny VM or a tiny container

On Unikernels

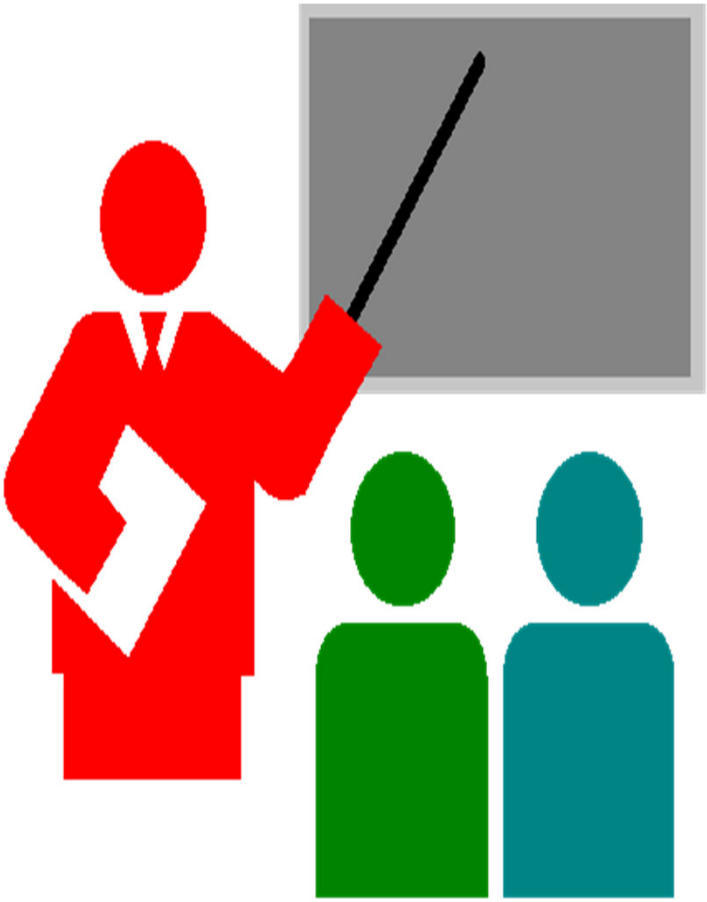
Pros and cons:

- Smaller footprint
- Boot up faster
- Less flexible
  - Addition / removal of functionality requires re-compilation

Virtualization and Serverless Computing

Server-less computing

Server-less Computing (Function as a Service)



- Introduction
- Architecture
- Pros / Cons

Introduction

Server-less does not mean there is no server !!!

- There are indeed servers !!!
  - However the servers are completely transparent to the cloud users, unlike (Virtual Machine (VM), Containers, Uni-kernel)
    - Server-less computing might actual rely on VMs or containers or uni-kernels
  - Cloud users deal with functions
    - thus Functions as a Service (FaaS)

Architecture

Principles

1) Applications built as a set of functions
2) When there is a request for a given function, a run time environment (e.g. VM, container, uni-kernel) is launched with the function code + libraries
3) The run time is terminated after the execution of the function

Architecture (Reference 1)

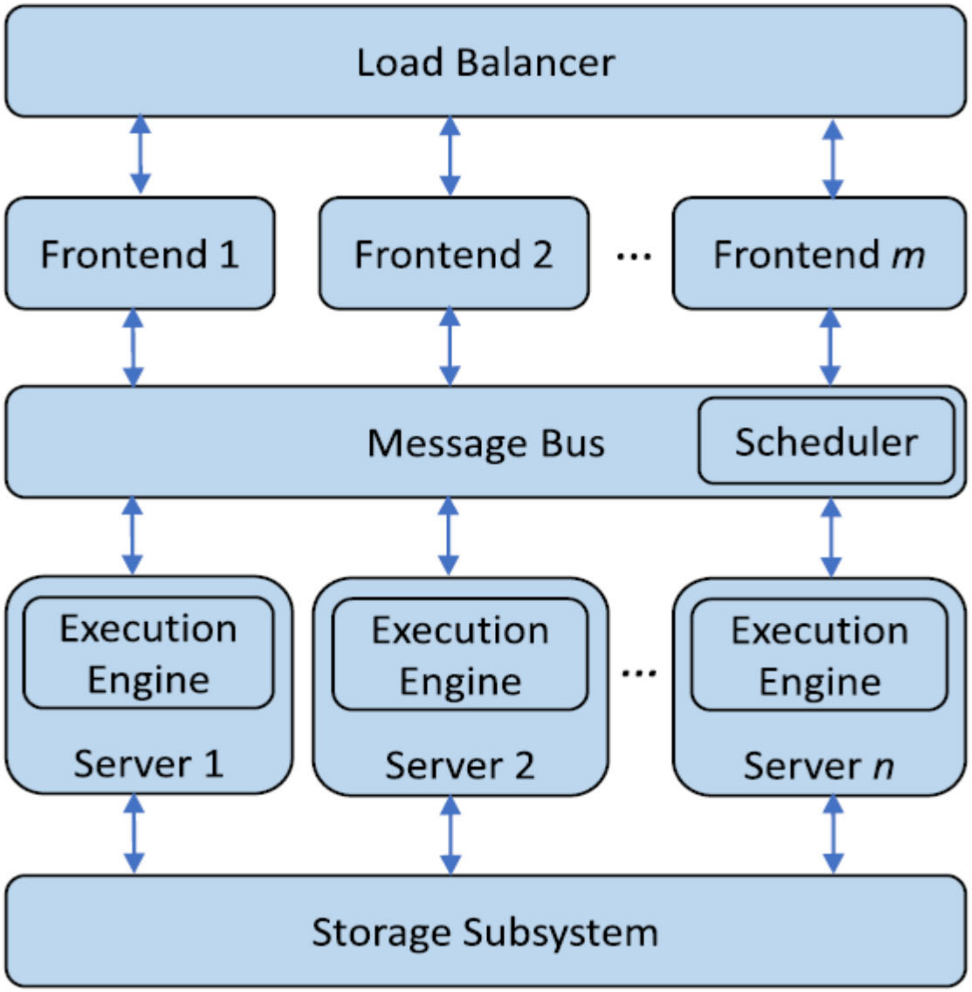


Fig. 1. *Serverless platform architecture.*

Architecture

Load balancer:

- Self explanatory

Front end:

- End user interface

Message bus and scheduler:

- Mediation between front ends and execution engines

Architecture

Load balancer:
- Self explanatory

Front end:
- End user interface

Message bus and scheduler:
- Mediation between front ends and execution engines
  - Relies on a publication / subscription principles

Architecture

Execution engine:

- Self explanatory
  - Might rely on VM, containers and uni-kernels

Storage sub-system:

- States
- Persistent data

Pros (Examples)

- No real / virtual server management by cloud users

- Resource Efficiency and low cost

- Built-in scalability

Cons (Examples)

- **Most cited:**
  - Start up latency
- **Others:**
  - Learning curve of the new programming model (e.g. stateless functions + events)

Pros vs Cons

- Decision to be made on case by case basis (Ref. 1)

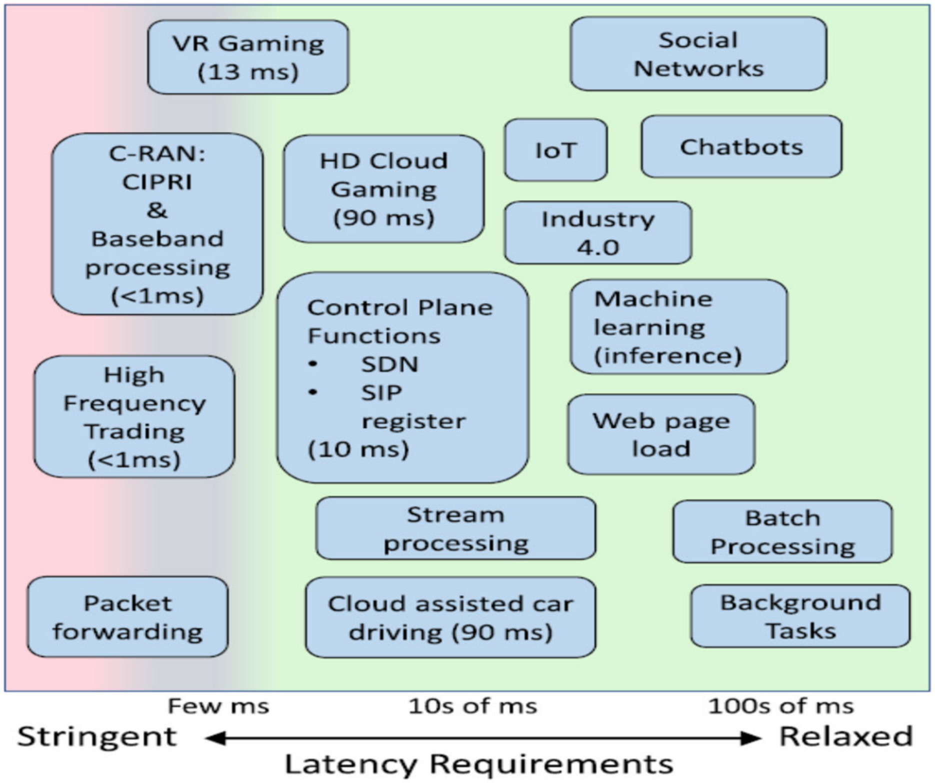


Fig. 3. *Latency requirement ranges for various applications.*

Pros vs Cons

- **Decision to be made on case by case basis (Ref. 1)**

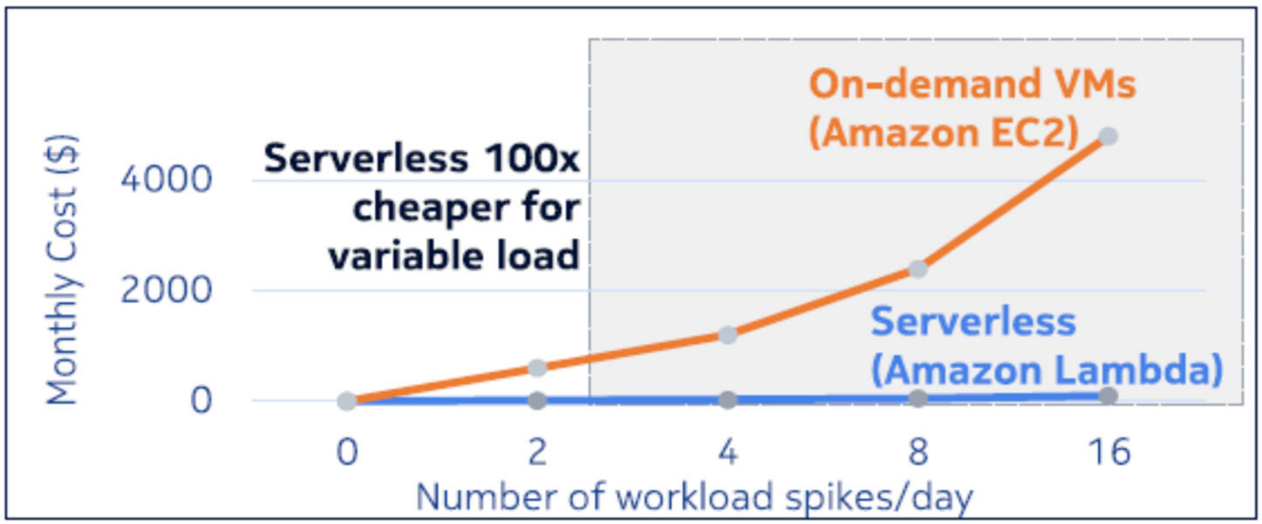


Fig. 4. *Cost comparison between Amazon Lambda (serverless) and Amazon EC2 (VMs) for spiky workload. In the gray region, serverless is 100x cheaper.*

References

- M. Pearce et al., Virtualization: Issues, Security, Threats, and Solutions, ACM Computing Survey, February 2013

- T. Goethals et al., Unikernels vs. Containers: An In-Depth Benchmarking Study in the Context of Microservice Application, IEEE SC2 Conference, November 2018

- P Aditya et al, Will Servless Computing Revolutionize NFV, Proceedings.of the IEEE, April 2019

The End