Introduction to Virtualization
Part. 1

Dr Stella Bitchebe
PostDoctoral Researcher
McGill University
April 4, 2023

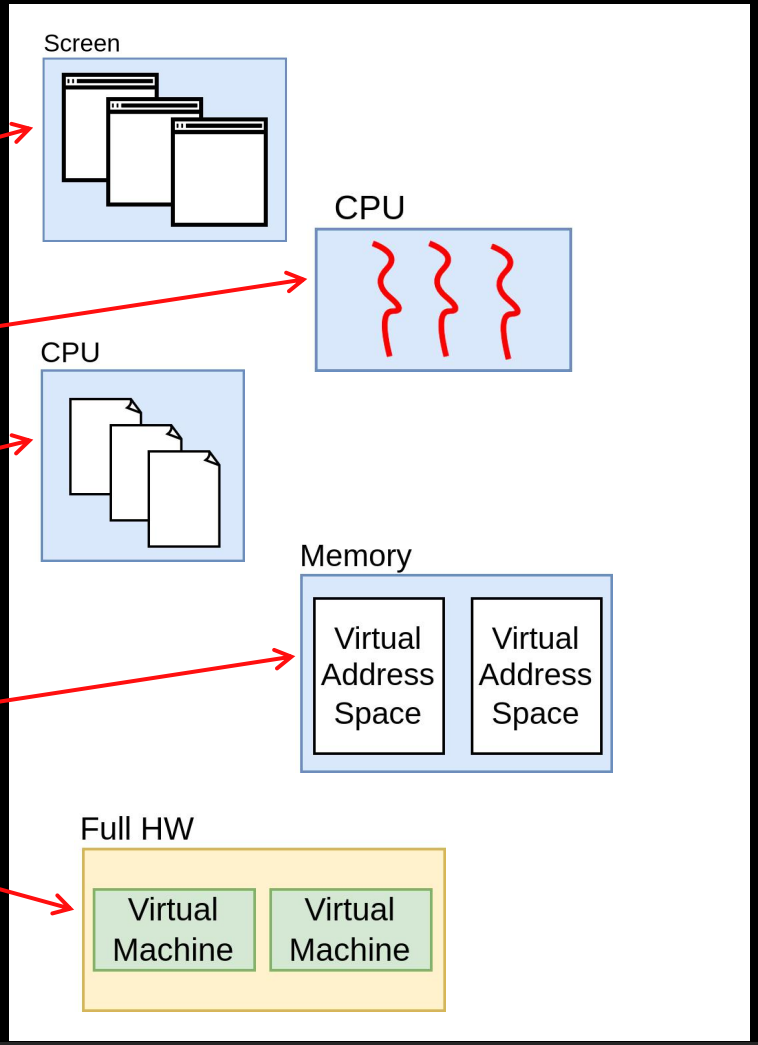
Virtualization

Definition

Virtualize: create an abstraction of something that is real/physical, see [1]

In CS, we can virtualize:

- Screen --> Window
- CPU --> Process
- Disk --> File
- Memory --> Virtual Memory
- Full HW --> Virtual Machine (VM)



Virtualization

Definition

Virtualize: create an abstraction of something that is real/physical, see [1]

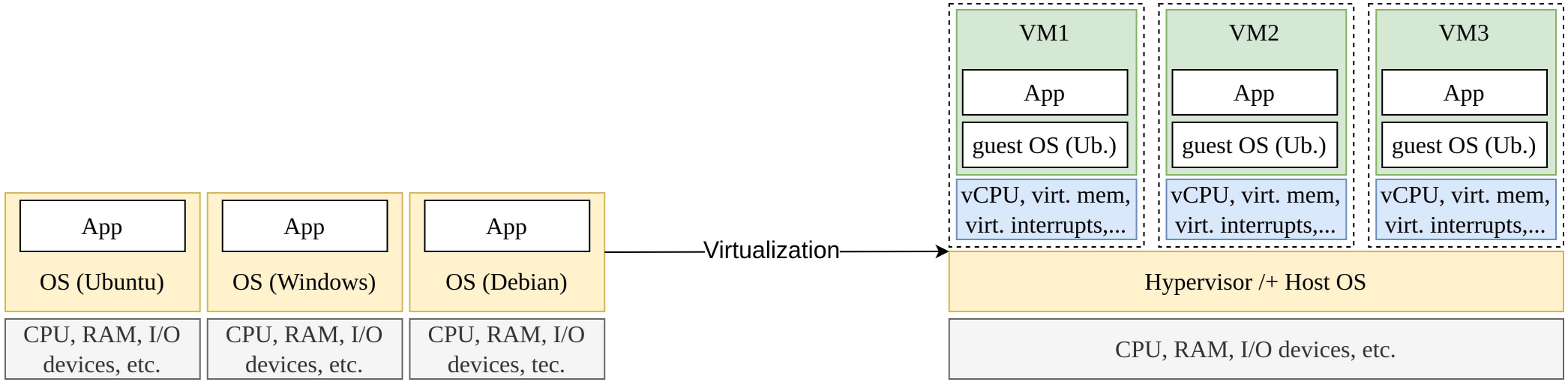
In CS, we can virtualize:

- Screen --> Window
- CPU --> Process
- Disk --> File
- Memory --> Virtual Memory
- Full HW --> Virtual Machine (VM)

Goal: sharing / isolation / saving

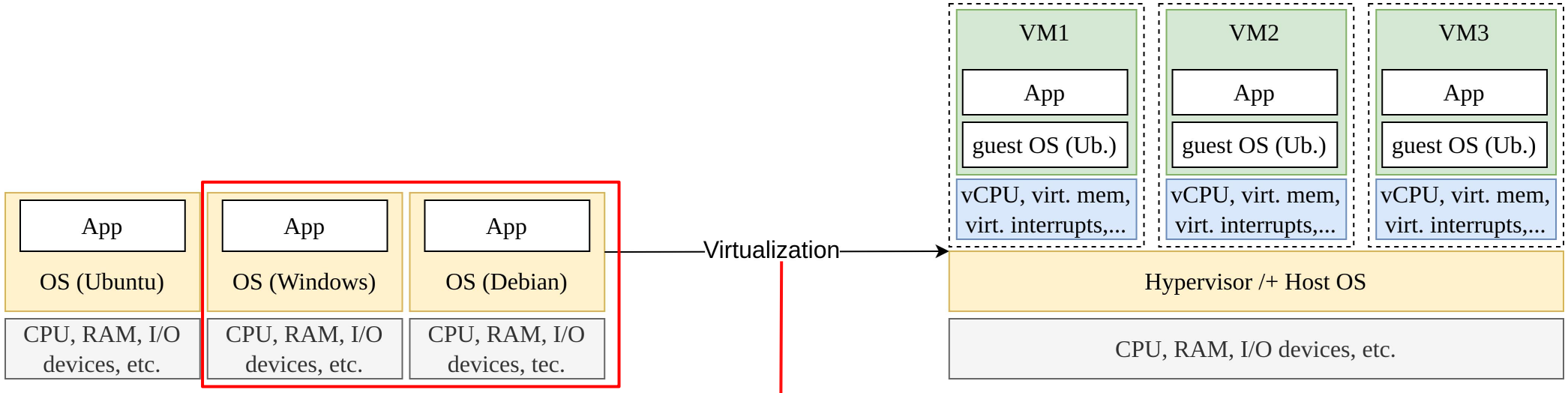
Virtualization

Virtual Machines



Virtualization

Virtual Machines



Allows reducing 50% - 70%
of electrical expenses [10]

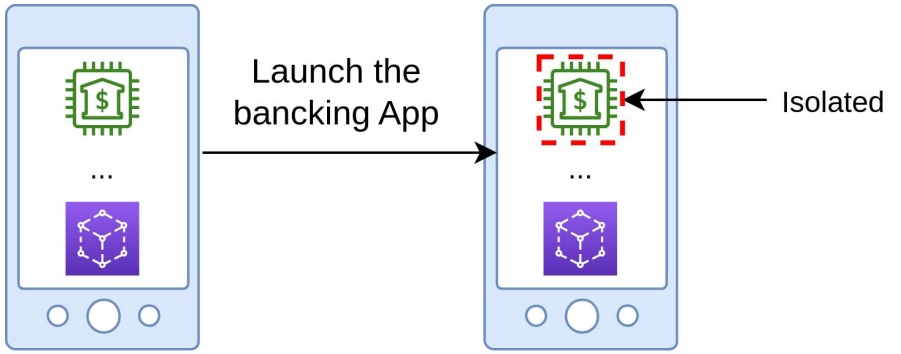
Virtualization

Example Application

A unique whatsapp app possible in a mobile phone

Create a "*virtual phone*" that enables installing another app

Runing many instances of mobile Apps



Mobile phone with many Apps not launched

Create an isolated space to run the banking App for security

Security and Isolation of sensitive Apps

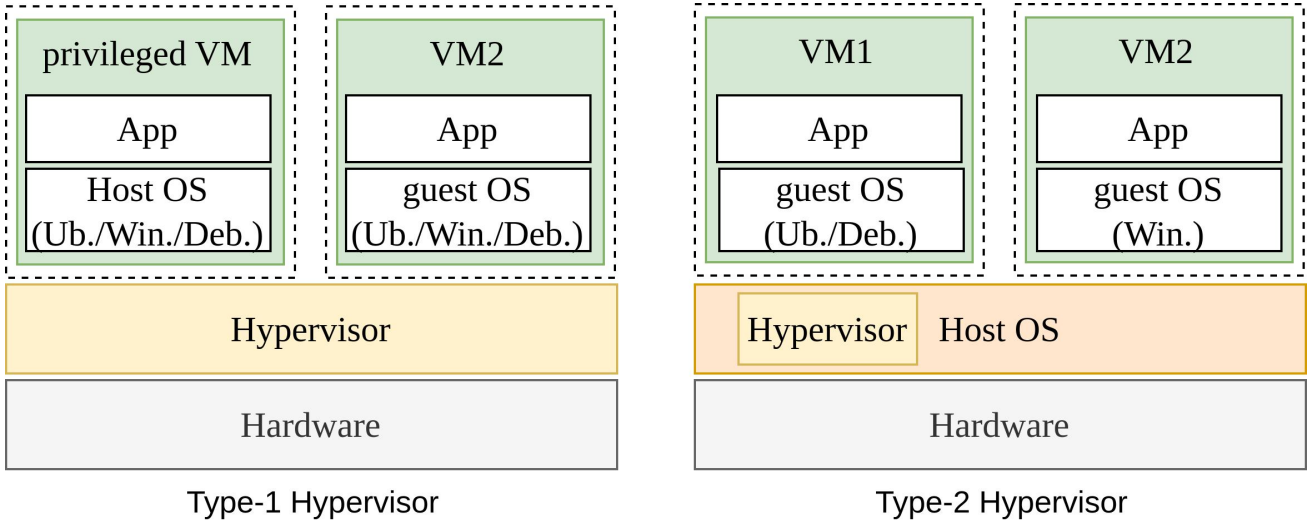
Hypervisors

Definition

- Software layer that enables virtualization.
- Intermediary between the VM and the hardware
- Role:
  - Physical resources virtualization
  - Scheduling of VMs
  - Memory Allocation
  - Interrupt management
  - etc.

Hypervisors

Types



Type-1: runs immediately atop the physical hardware and can directly manage the physical resources (CPU, memory, I/O devices) for virtualization

- Xen
- Microsoft Hyper-V
- VMWare ESXi

Type-2: are embedded with the host OS (e.g., as a kernel module)

- KVM
- Virtualbox
- VMWare Workstation

Hypervisors

Example

You can try compiling and installing the Xen hypervisor, creating a VM and making some examples following this tutorial:
https://github.com/bstellaceleste/OoH/tree/SPML/OoH-PML

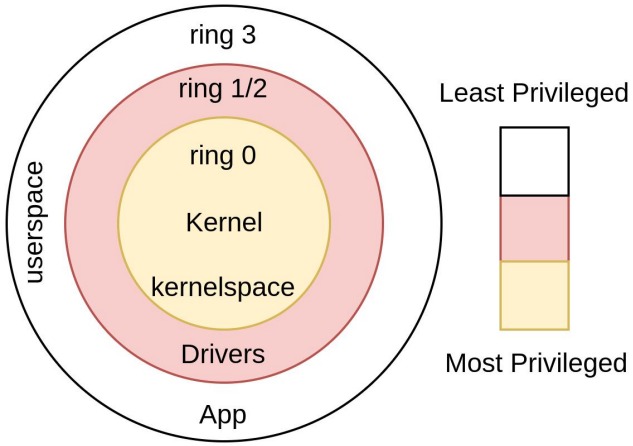
Types of VMs

Execution Rings

Also called CPL (current privileged level): determine what instructions the processor is allowed to perform without trapping

At ring/CPL 3, privileged instructions trap to the kernel e.g.,:

- reading/writing processor registers
- accessing memory
- interrupts
- etc.



Types of VMs

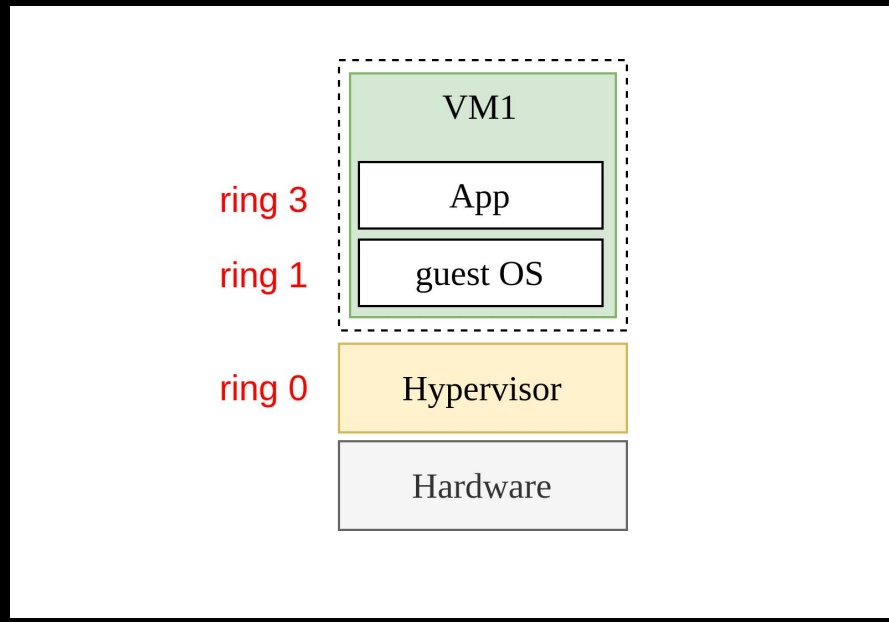
PVM (Paravirtualized VM)

PVM demotes OS to ring 1 and the hypervisor runs at the most privileged level ring 0.

As a consequence, guest OSes must be modified:

- Guest OS register a descriptor table for exception handlers with the hypervisor
- Guest OS has a timer interrupt and aware of *real* and *virtual* time
- Guest attempts to update page tables is handled and validated by the hypervisor

Guest OSes are aware of being virtualized.



Types of VMs

HVM (Hardware VM)

McGill

Hardware Assisted Virtualization (HAV)

Architectural support for virtualization (e.g., Extended Page Table -EPT-, explained in following sections)

Introduced in 2005/2006:

- AMD-V, 2005
- Intel VT-x (Virtualization Technology extensions), 2006

VT-x replicates the original states of the processor and introduces 2 new execution modes:

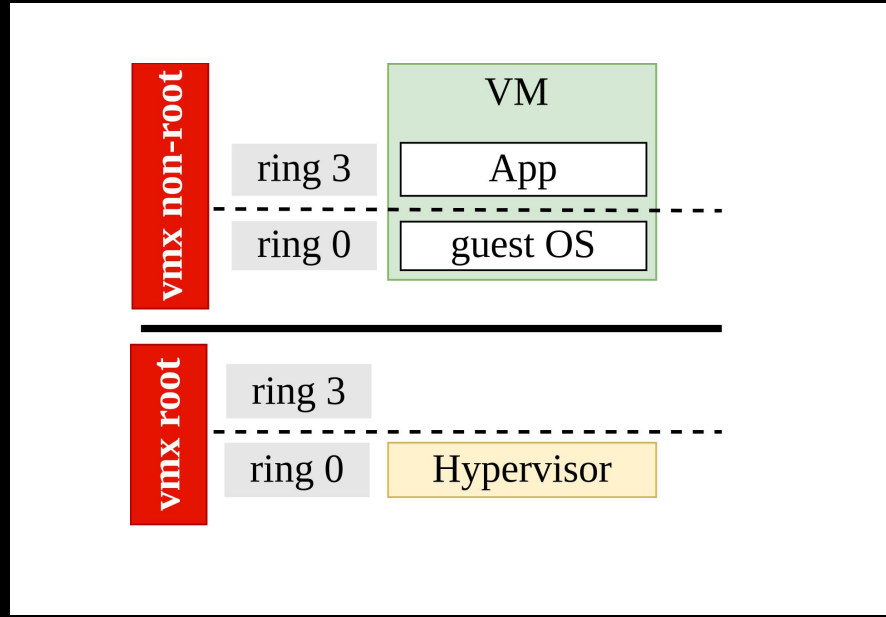
- vmx root
- vmx non-root

Types of VMs

HVM (Hardware VM)

With HAV:

- Guest OSes (HVMs) to run unmodified (in ring 0) thanks to the new processor execution modes
- Guest OS is unaware of being virtualized while the CPU does



MMU Virtualization

Native Page Table (PT)

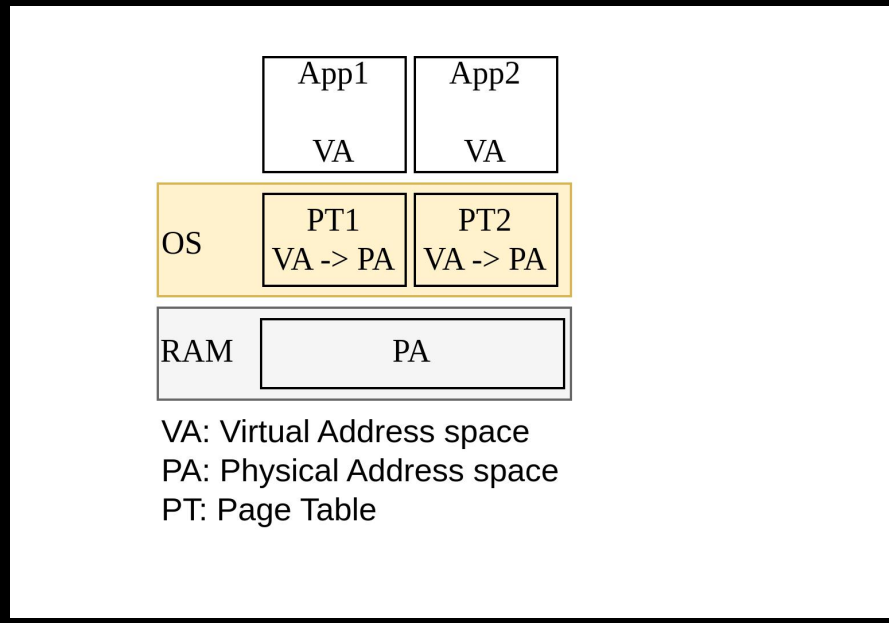
McGill

Bare-metal (non-virtualized) Environment

In native environments, memory is virtualized to ensure process isolation

The OS maintains a per-process page table (PT) that translates virtual address (va) to physical address (pa)

The OS intercepts any attempt from an application to access physical memory and walks the PT

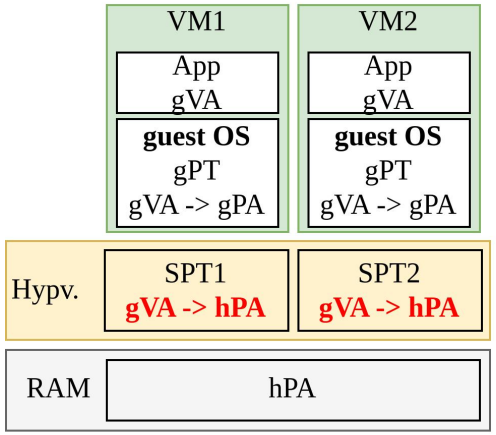


MMU Virtualization

Shadow Page Table (SPT)

Virtualized Environment: PVM

- Before VT, the hypervisor was maintaining gVA - hPA translations so-called SPT because in the processor isn't aware of virtualization => on page fault (#PF), it reads the CR3 register (that points to the SPT) as on bare-metal and walks a unique PT.
- As a consequence, any attempt by the guest OS to update a gPT is captured by the hypervisor to also update the SPT => heavy source of latency.



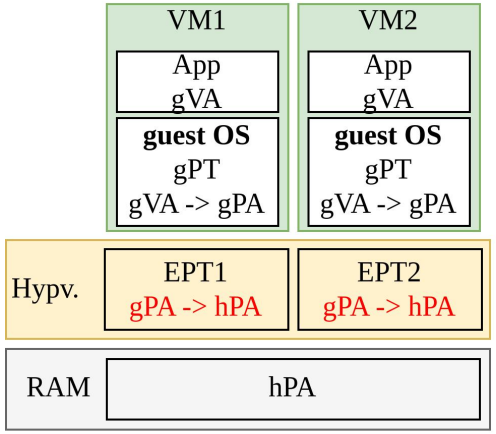
(g)VA: (guest) Virtual Address space
(g/h)PA: (guest/host) Physical Address space
gPT: guest Page Table
SPT: Shadow Page Table

MMU Virtualization

Extended Page Table (EPT)

Virtualized Environment: HVM

- EPT introduces a new CR3 register called nested CR3 (nCR3).
- With EPT, the processor is aware of virtualization => upon #PF, when the CPU is in vmx non-root mode, it knows it has two PTs to walk: the gPT and the EPT.
- The reads CR3 pointing to gPT, then reads nCR3 pointing to EPT.
- If a gPA mapping is missing, the hypervisor simply updates the EPT independently from the guest => the hypervisor no longer needs to trap guest PT updates.



(g)VA: (guest) Virtual Address space
(g/h)PA: (guest/host) Physical Address space
gPT: guest Page Table
EPT: Extended Page Table

Importance of Virtualization

- Virtualization has become the foundation of data centers because it allows:
  - Resource mutualization (optimal resource utilization)
  - Cost reduction
  - Overcommitment (allocating more resources than is physically available)
  - etc.

- Most of cloud providers today sell resources in the form of virtual machines, which is less expensive for customers: AWS, Azure, Google Cloud, etc.

- Costumers use VMs for deployment tests, benchmarking, security isolation, etc.

You can have fun creating VMs on AWS and making some tests following this short tutorial [2].

You can find out more details on virtualization in:

- My thesis [3], Chapter 2
- Professor Edouard Bugnion's book [4] on HW and Software Support for Virtualization

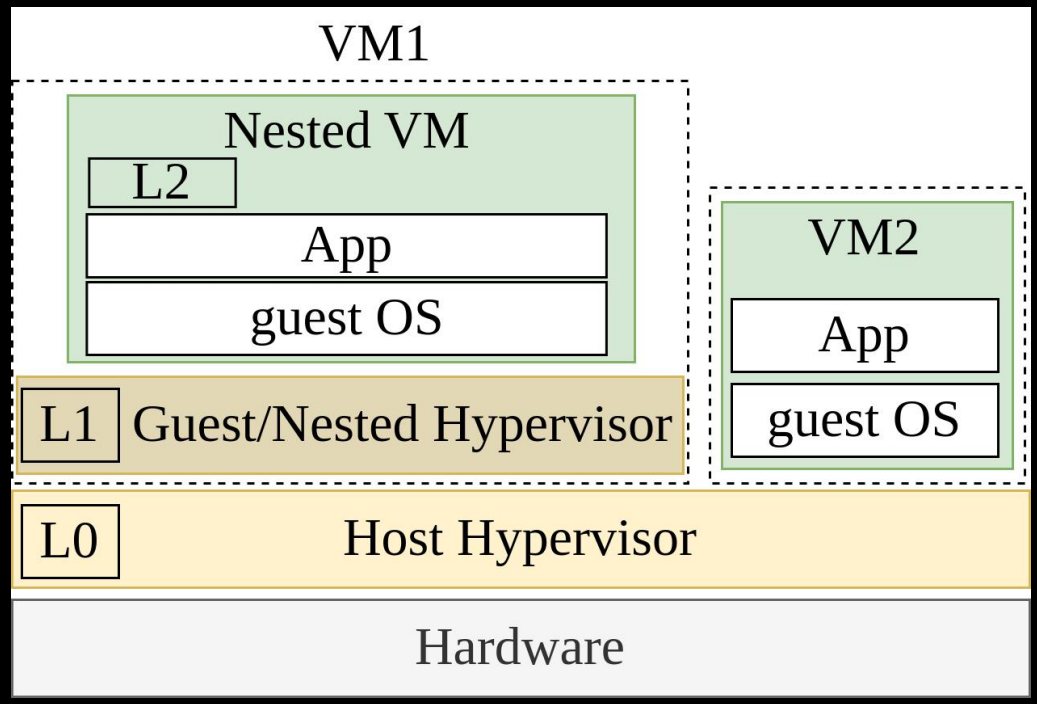
Thesis Research Projet
Out of Hypervisor (OoH)

Nested Virtualization

Definition

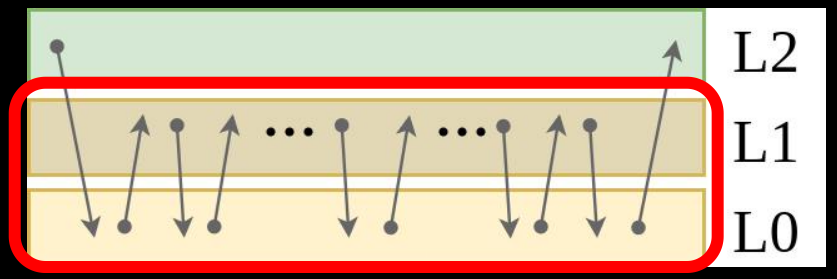
"*Recursive virtualization, where a virtual machine runs under itself a copy of a hypervisor.*"

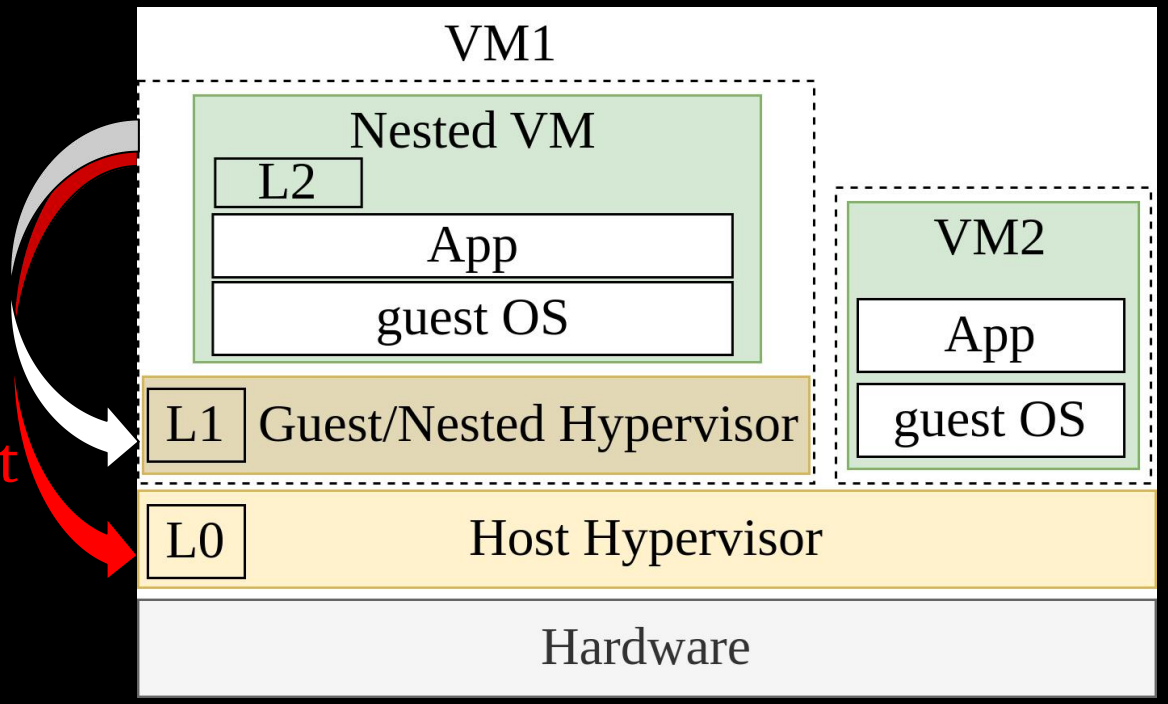
Popek and Goldberg, 1970s



Nested Virtualization

Drawbacks





Single trap from nested VM produces multiple traps to the host hypervisor

Significant number of VM exits (at least ×2)

Vilanova et al. [5, ISCA'19]

Nested Virtualization

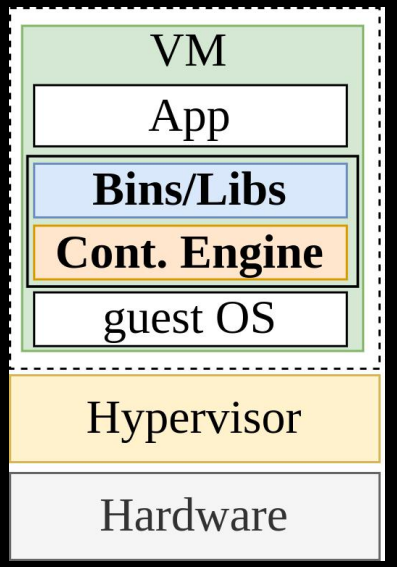
Use Case

Tests/Development/Demo

- No matter the overhead
- Not in production

Isolation

Containers

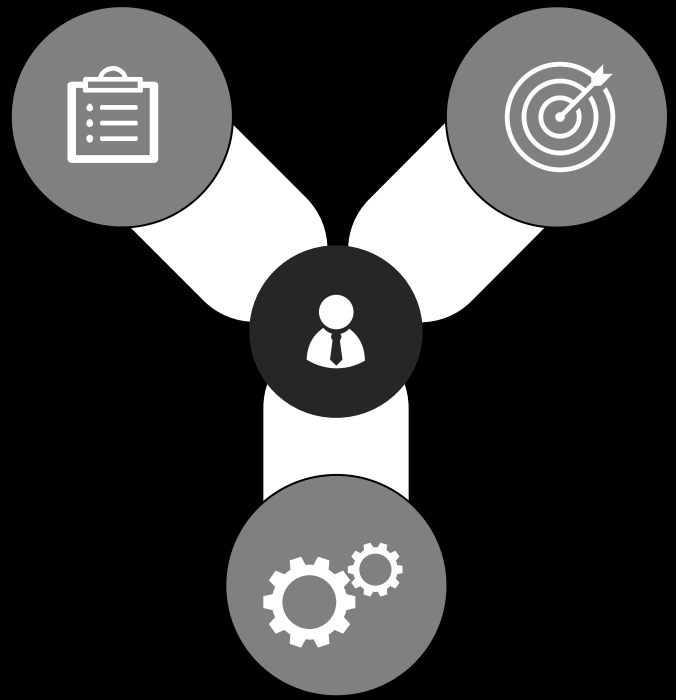


Out of Hypervisor (OoH)

Presentation

What?

New virtualization research axis



Why?

Avoid trying to virtualize full virtual hardware inside a VM

How?

Expose individually hypervisor oriented hardware virt. features to the guest OS so that its processes can benefit from them

Out of Hypervisor (OoH)

Virtualization Technologies Categorization

G1: Resource Multiplexing

- ✓ EPT (*Extended Page Table*): MMU virtualization
- ✓ SRIOV (*Single Root I/O Virtualization*)
- ✓ APICv (*Advanced Programmable Interrupt Controller virtualization*)
- ✓ etc.

G2: VM's Management

- ✓ PML (*Page Modifcation Logging*)
- ✓ CAT (*Cache Allocation Technology*)
- ✓ SPP (*Sub-Page write Permissions*)
- ✓ etc.

Out of Hypervisor (OoH)

Virtualization Technologies Categorization

G1: Resource Multiplexing

- ✓ EPT (*Extended Page Table*): MMU virtualization
- ✓ SRIOV (*Single Root I/O Virtualization*)
- ✓ APICv (*Advanced Programmable Interrupt Controller virtualization*)
- ✓ etc.

G2: VM's Management

- ✓ PML (*Page Modifcation Logging*)
- ✓ CAT (*Cache Allocation Technology*)
- ✓ SPP (*Sub-Page write Permissions*)
- ✓ etc.

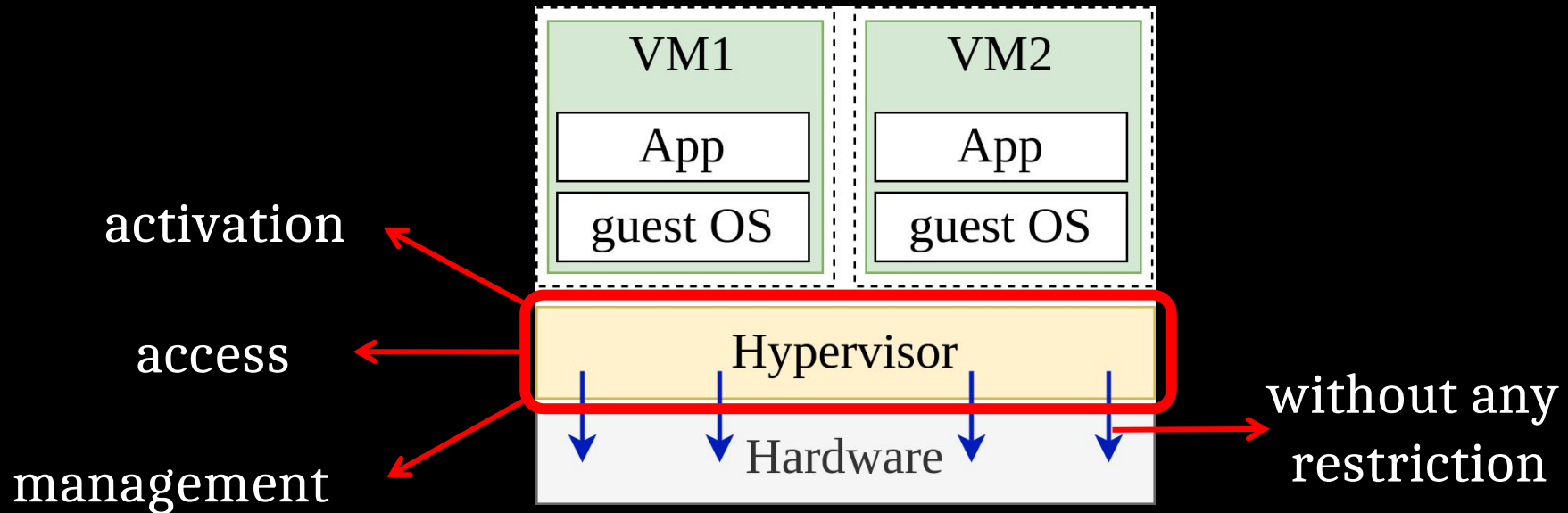
↓

OoH scope: Can be exploited to remove the need for a nested hypervisor to leverage hypervisor properties

Out of Hypervisor (OoH)

Problem

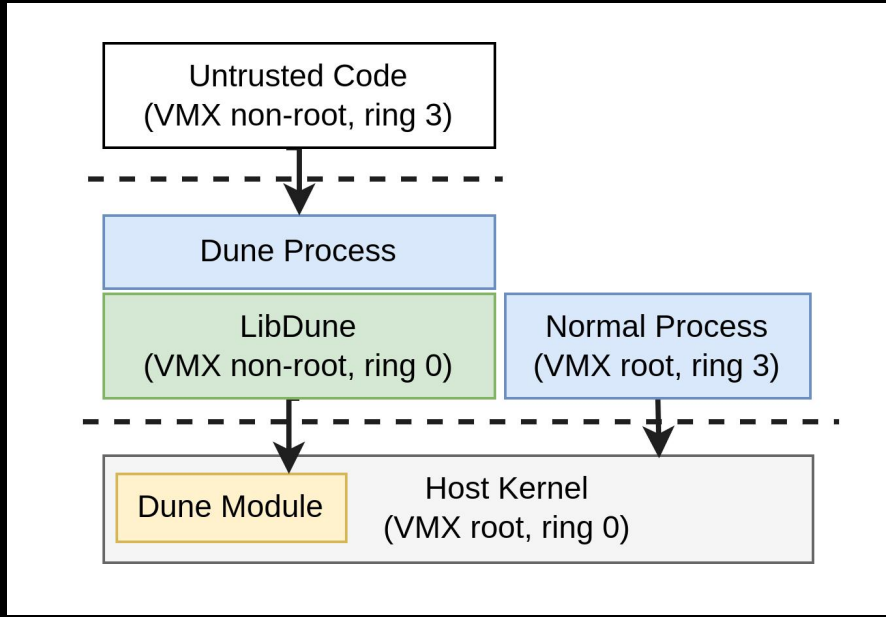
All hardware functionalities are hypervisor-oriented



Out of Hypervisor (OoH)

State-of-the-art

Dune [7, OSDI'12]: leverages hypervisor-oriented hardware virtualization features (EPT) to make privileged instructions, usually only accessible in kernel mode (ring 0), available to processes.

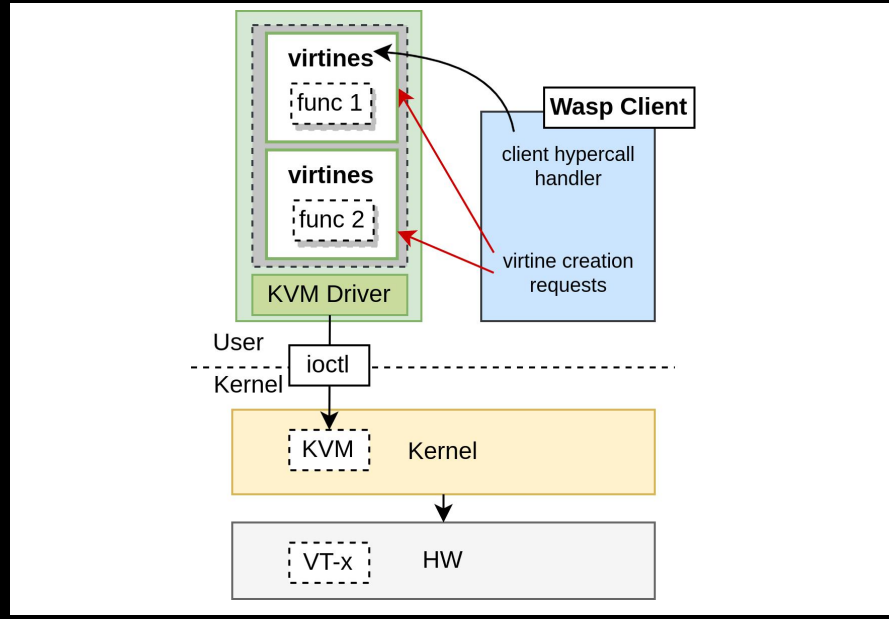


Out of Hypervisor (OoH)

State-of-the-art

Dune [7, OSDI'12]

Virtines [8, Eurosys'22]: uses Vt-x (via the KVM API [6] on the Linux kernel) to propose a leightweight VM-like abstraction for intra-process isolation at a function scale.



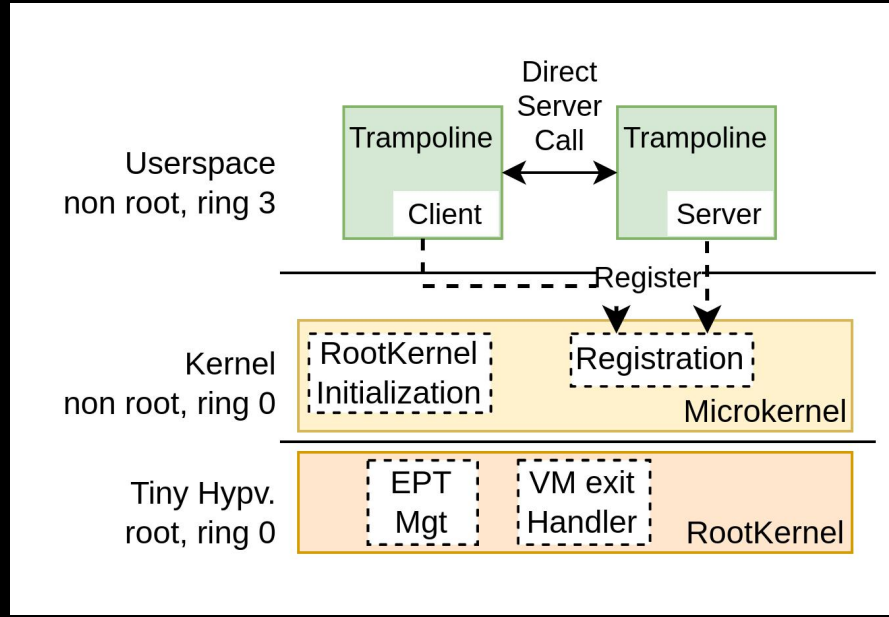
Out of Hypervisor (OoH)

State-of-the-art

Dune [7, OSDI'12]

Virtines [8, Eurosys'22]

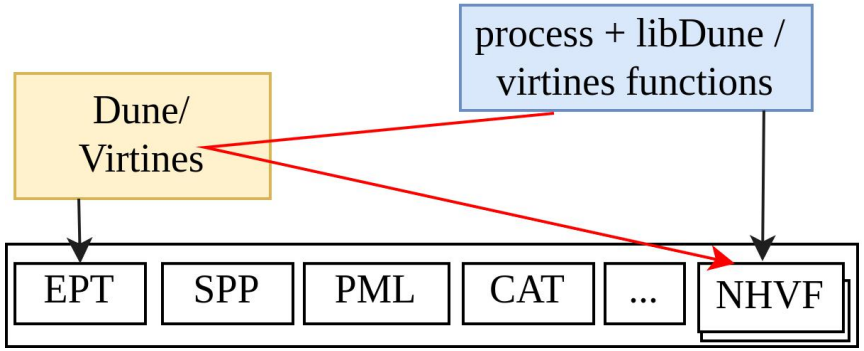
SkyBridge [9, Eurosys'19]: uses Vt-x (via the KVM API [6] on the Linux kernel) to propose a leightweight VM-like abstraction for intra-process isolation at a function scale.



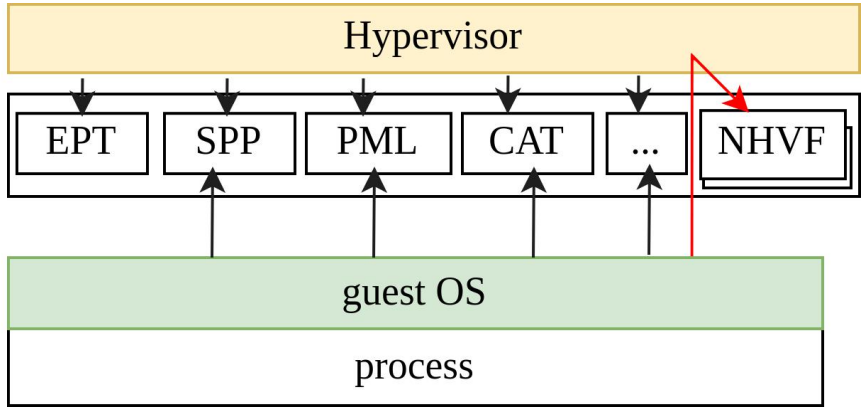
Out of Hypervisor (OoH)

State-of-the-art

State-of-the-art



OoH



Unlike those solutions that use HAV features for isolating guest processes (vmx non-root ring 3), OoH seeks at making those processes to access and use hardware virt. features

Out of Hypervisor (OoH)

Illustration

Details about OoH's illustrations can be found in my thesis [3].

2 Virtualization Features

Intel PML

Intel SPP

4 Application' Examples

In privileged VMs (Xen's dom0)

Working set size estimation of VMs

In Unprivileged VMs (Xen's domU)

Checkpoint/Restore

Garbage Collection

Buffer Overflow Mitigation

References

[1] https://lig-membres.imag.fr/krakowia/Files/Enseignement/Histoire-Informatique/Cours/8-Systemes/8-systemes.key-2pp.pdf
[2] https://github.com/bstellaceleste/ScientificDaysCameroon/blob/main/tutorials/README.md
[3] https://perso.ens-lyon.fr/celestine-stella.ndonga-bitchebe/doc/main.pdf
[4] https://bit.ly/3Gi2Yeb
[5] Vilanova, L., Amit, N., and Etsion, Y. Using smt to accelerate nested virtualization. In Proceedings of the 46th International Symposium on Computer Architecture (New York, NY, USA, 2019), ISCA '19, Association for Computing Machinery, p. 750–761. (Cited on pages 1, 2, 20 and 68.)
[6] https://lwn.net/Articles/658511/
[7] Belay, A., Bittau, A., Mashtizadeh, A., Terei, D., Mazières, D., and Kozyrakis, C. Dune: Safe user-level access to privileged CPU features. In 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12) (Hollywood, CA, Oct. 2012), USENIX Association, pp. 335–348. (Cited on pages 5, 15 and 25.)
[8] Wanninger, N., Bowden, J. J., and Hale, K. C. Virtines: Virtualization at function call granularity. CoRR abs/2104.11324 (2021). (Cited on pages 9, 15 and 25.)
[9] Mi, Z., Li, D., Yang, Z., Wang, X., and Chen, H. Skybridge: Fast and secure inter-process communication for microkernels. In Proceedings of the Fourteenth EuroSys Conference 2019 (New York, NY, USA, 2019), EuroSys '19, Association for Computing Machinery. (Cited on page 15.)
[10] e-Energy '11: Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking. New York, New York, USA: ACM, 2011. I S B N: 978-1-4503-1313-1