Technische Universität Dresden

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

ADVANCED OPERATING SYSTEMS

CLOUD COMPUTING AND VIRTUALIZATION

https://tud.de/inf/os/studium/vorlesungen/aos

HORST SCHIRMEIER

Overview

- Basic Terminology
- Cloud System Software
- Virtualization Basics
- CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- Summary

Literature

Silberschatz:
--- :-(

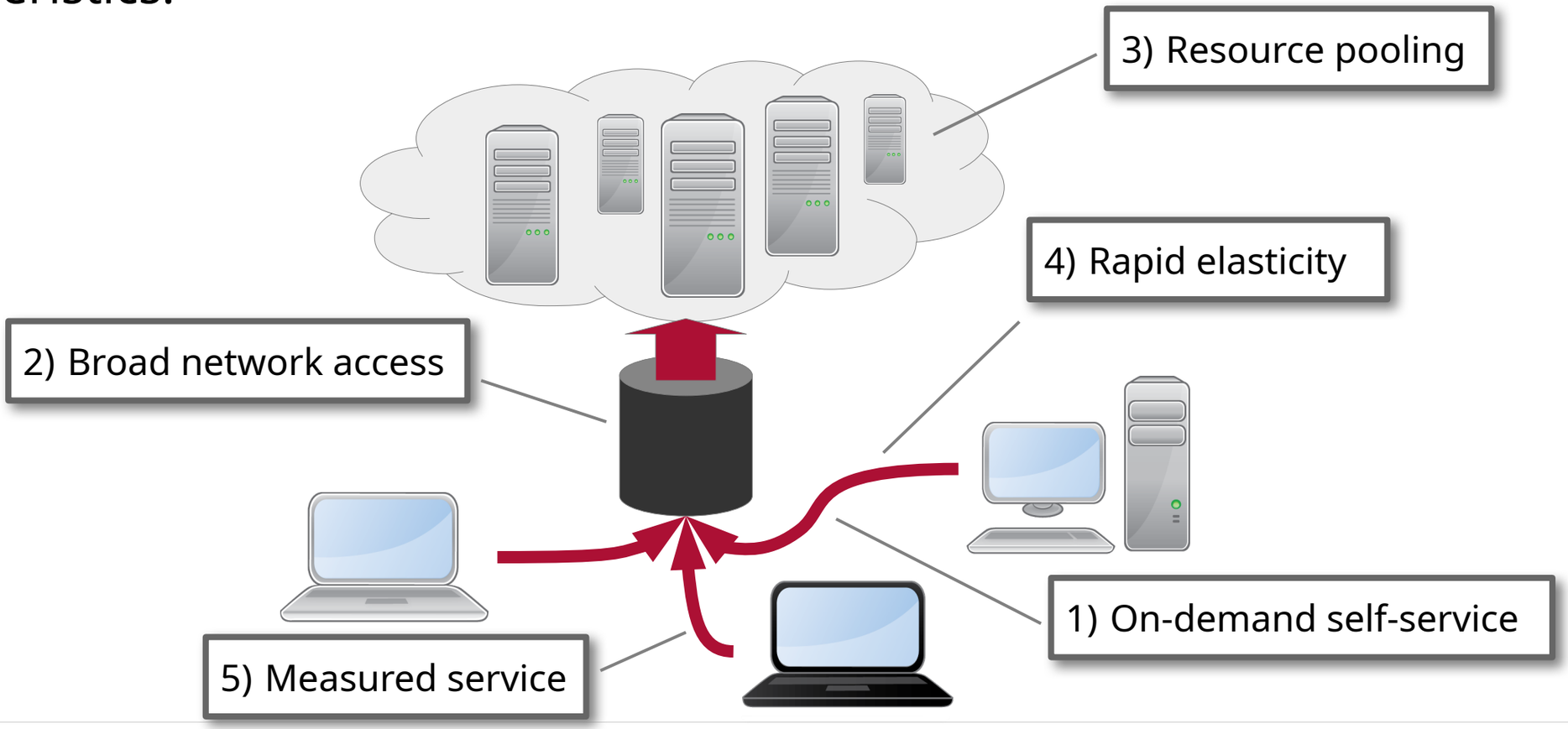
Tanenbaum: Chap. 7 … “Virtualization and the Cloud”

Overview

- **Basic Terminology**
- Cloud System Software
- Virtualization Basics
- CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- Summary

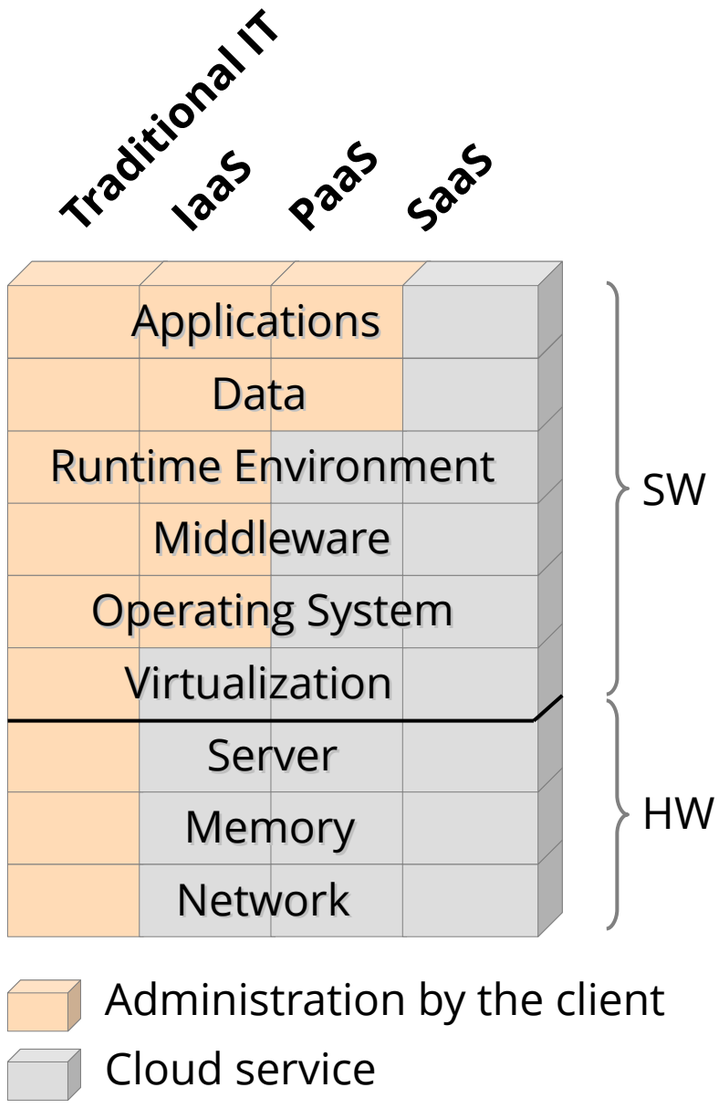
Cloud Computing

- The *National Institute of Standards and Technology* (NIST) defines five essential characteristics:



Cloud Service Models

- **SaaS – *Software-as-a-Service***
  - Cloud service provider makes ready-to-use **application** available.
  - e.g. Office365, Gmail, Zoom
- **PaaS – *Platform-as-a-Service***
  - **Execution environment for applications** including operating system and runtime environment (depending on programming language)
  - e.g. Engine Yard, Google App Engine
- **IaaS – *Infrastructure-as-a-Service***
  - (Virtual) **hardware platform**
  - e.g. Amazon EC2, Microsoft Azure



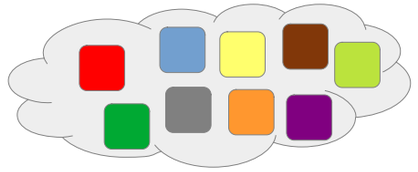
Discussion: Dark sides of the Cloud

Cloud computing offers many advantages, but can also create new problems that need to be taken into account during planning:

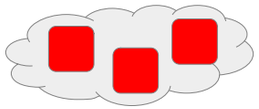
- **Data privacy**
  - **Where is my user's/customer's data stored?** What data protection guidelines apply in the country in question (or the country the CSP resides in – cf. US "Cloud Act")?
  - Is the cloud service provider trustworthy?
- **Vendor lock-in**
  - Can I get my data if I want to **change providers**? If so, in what format?
- **Quality of service**
  - What **guarantees** does the service provider give me?

Provisioning Models

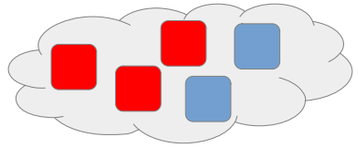
- **Public Cloud**
  - **Cloud Service Provider (CSP)** works for any customer.



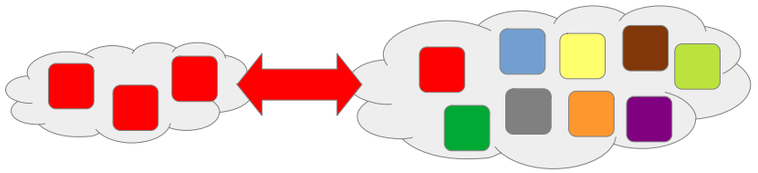
- **Private Cloud**
  - A cloud infrastructure for a (large) company. You can use your own or third-party resources. You have more control.



- **Community Cloud**
  - Several customers with the same requirements share a cloud infrastructure.



- **Hybrid Cloud**
  - Mixed concept



Provisioning Models – Comparison

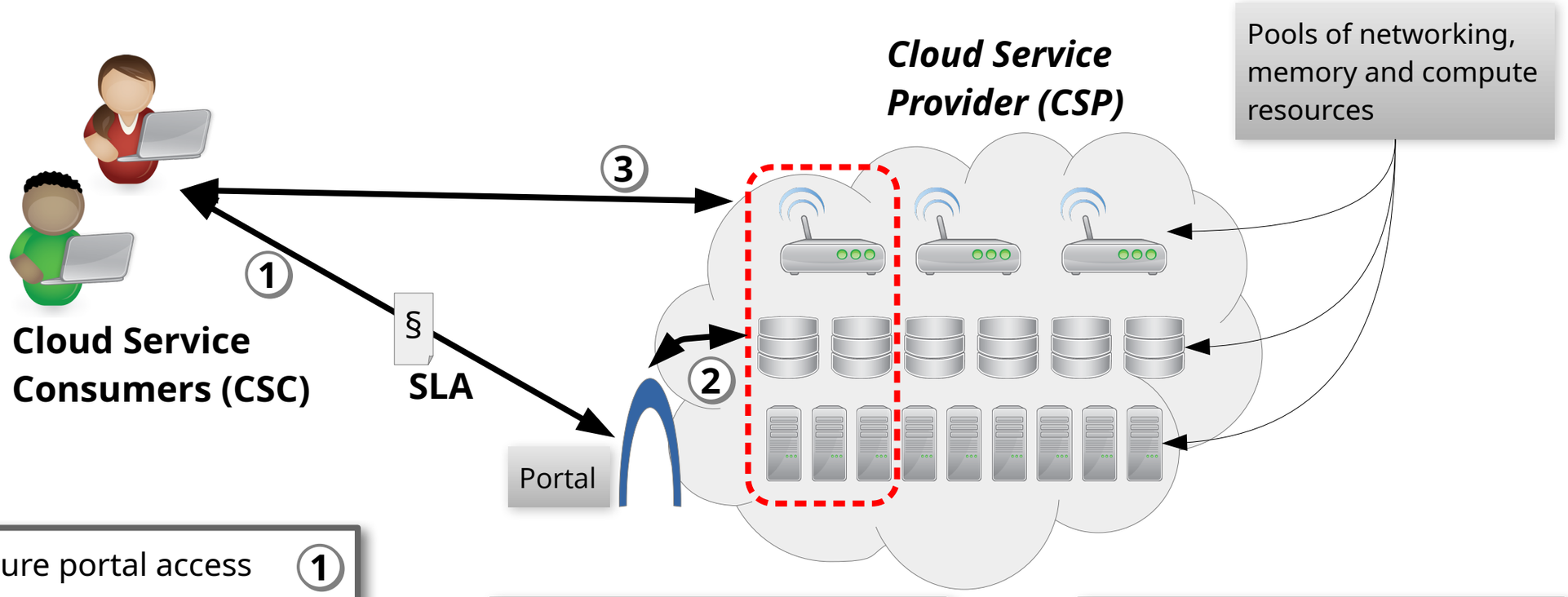
- Stallings: „*Operating Systems: Internals and Design Principles*“

	Private	Community	Public	Hybrid
Scalability	Limited	Limited	Very high	Very high
Security	Most secure option	Very secure	Moderately secure	Very secure
Performance	Very good	Very good	Low to medium	Good
Reliability	Very high	Very high	Medium	Medium to high
Cost	High	Medium	Low	Medium

Overview

- Basic Terminology
- **Cloud System Software**
- Virtualization Basics
- CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- Summary

Use Case / Requirements



- Secure portal access
- Function request (e.g. VM Templates)
- Service selection
- Monitoring and adjustments

1

- Provision of the requested resources
- **Service-Level Agreement (SLA)** and pricing confirmation

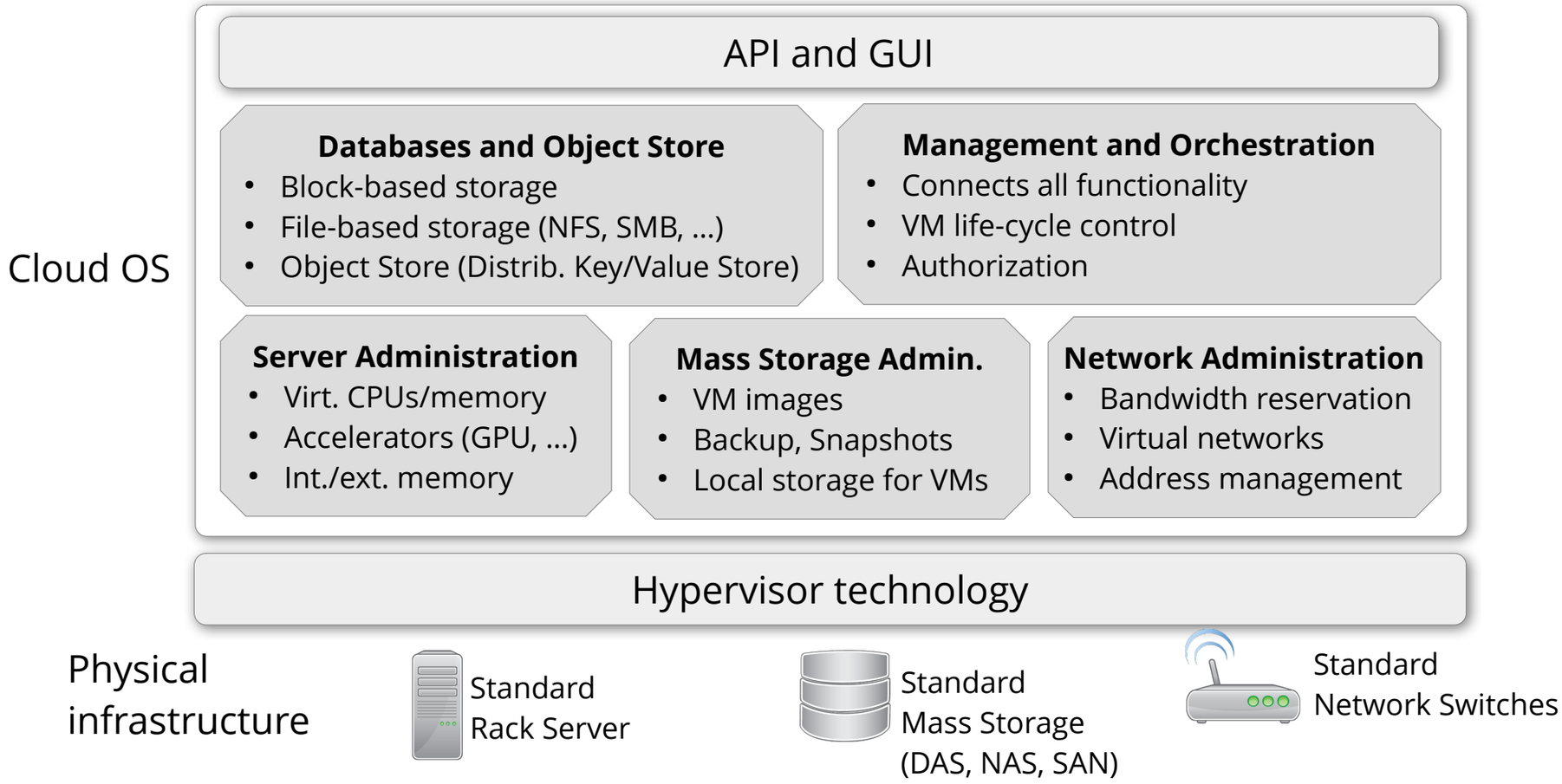
2

- Usage (CSC) and Monitoring (CSP)
- **Management** (CSP) (migration, redundancy, energy saving, extension, ...)

3

General Cloud-OS Architecture

All resources are virtualized → IaaS is the basis for all services



Strategic Questions

- Where to place VMs? When should you migrate?
- How to minimize SLA breaks? How much **oversubscription**?
- Is it worth freeing up and shutting down servers?

More resources are sold than are actually available.

Various strategies are possible:

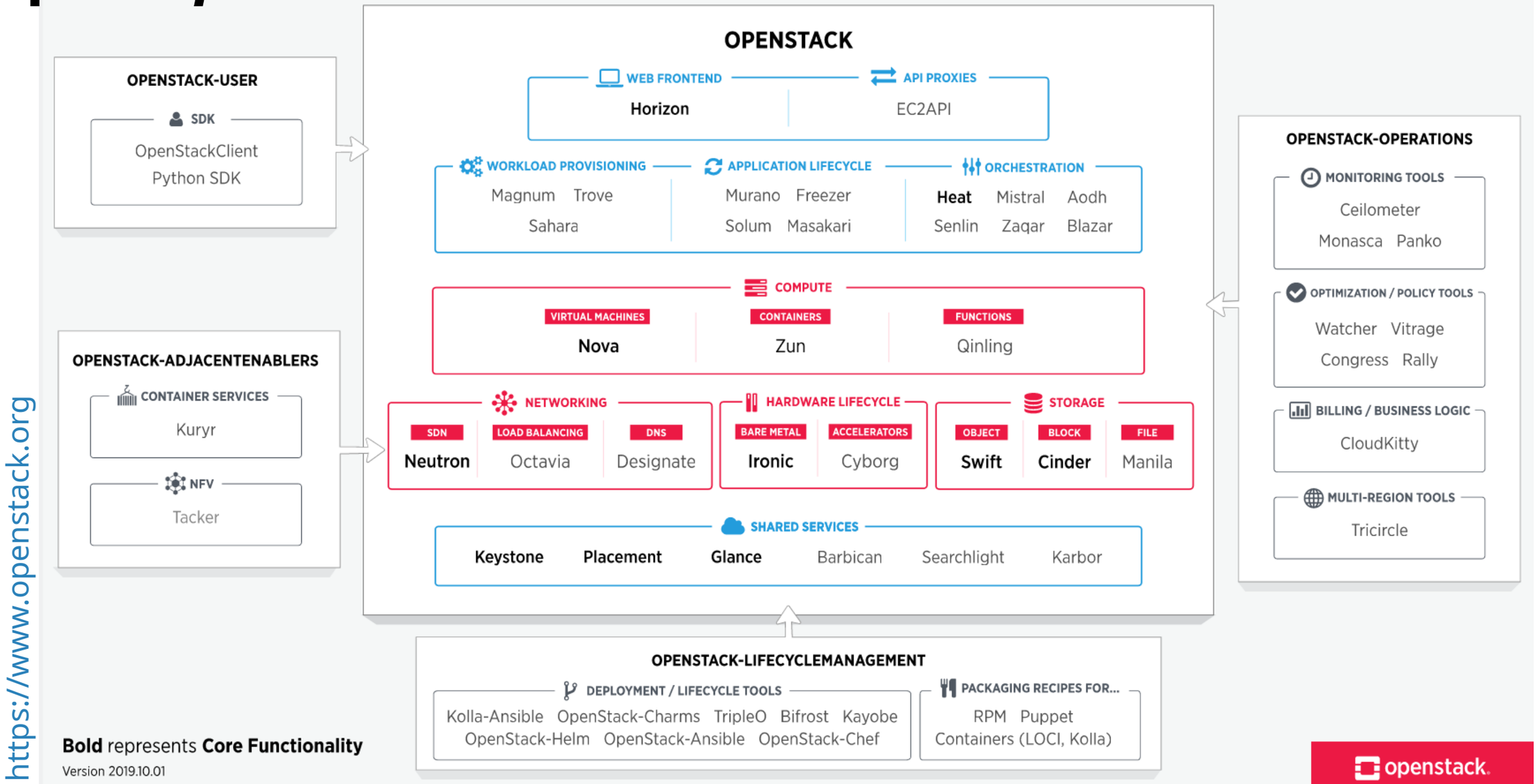
Scheduler	Migrationen	Brüche			Strafen	Kosten	Gewinn	Marge (in %)
		CPU	RAM	UT				
FF	28.326	12	370	3.350	87.792,85	31.537,03	-58.523,54	-96,2
HPGBF	5.355	401	212	60	90.558,46	30.381,79	-60.133,91	-98,9
HPGOP	2.372	114	101	33	30.942,97	31.093,17	-1.229,80	-2,0
HPGWF	3.705	49	141	0	24.760,00	30.781,68	5.264,65	8,7
MMBF	1.111	6	69	40	10.516,92	31.350,98	18.938,43	31,1
MMOP	825	2	34	7	4.934,46	32.066,84	23.805,03	39,1
MMWF	890	2	34	0	4.400,00	31.736,97	24.669,36	40,6
MMWF + LB	871	3	26	0	3.600,00	32.326,96	24.879,37	41,0

Tabelle 10.10: Zusammenfassung der Ergebnisse der Intra-DC-Scheduler inklusive EE-Erweiterung und initialer VM-Verteilung nach RAM-Ressourcen für den Bitbrains RnD Trace (Monat 1) mit 500 VMs (In jeder Simulation wurden 60.806,34 Umsatz erzielt)

We won't discuss the strategies in detail here.

Source:
Dissertation A. Kohne,
„SLA-basierte VM-Scheduling-Verfahren für Cloud-Föderationen“

Example: *OpenStack* Cloud OS

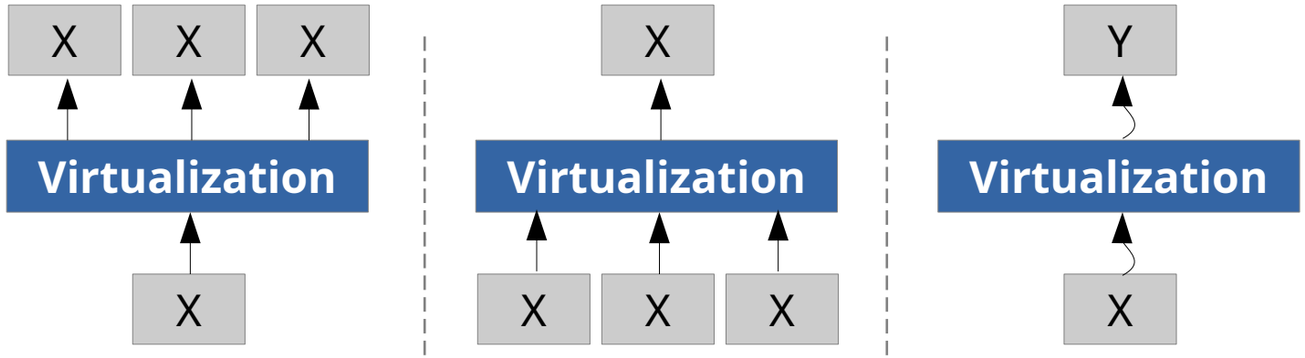


Overview

- Basic Terminology
- Cloud System Software
- **Virtualization Basics**
- CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- Summary

Relevance and Benefits of Virtualization

- Enforces strict adherence to layering structure by **resource control on access**
- Basis for …



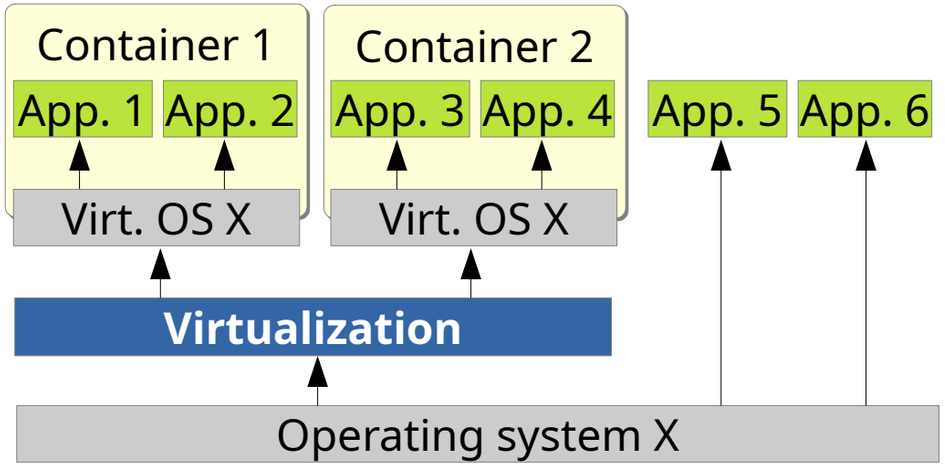
X and Y are resource types, e.g. RAM, disks, CPUs, I/O devices.

Source: [1]

- Construction principle can be repeatedly applied on different layers and for different resources.

Container-based Virtualization

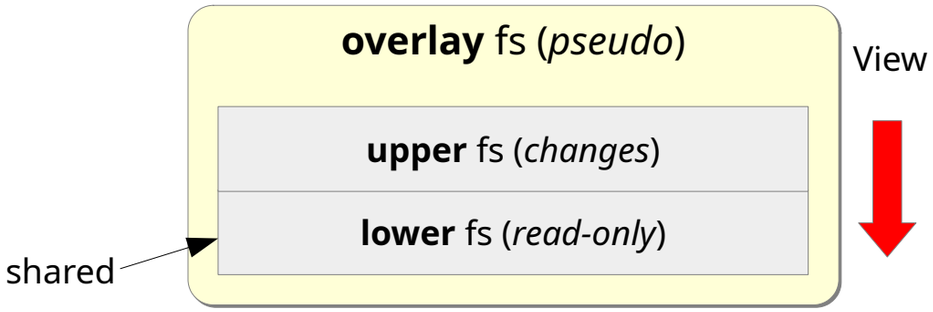
- short: **Container**
- OS **kernel** is virtualized
  - Containers share kernel
  - Libraries/system processes can vary



- Virtualization component ensures …
  - **Separated views**, e.g. each container only “sees” its own processes
  - **Resource partitioning**, e.g. regarding CPU time
  - **Efficient sharing**, e.g. avoiding duplicate files

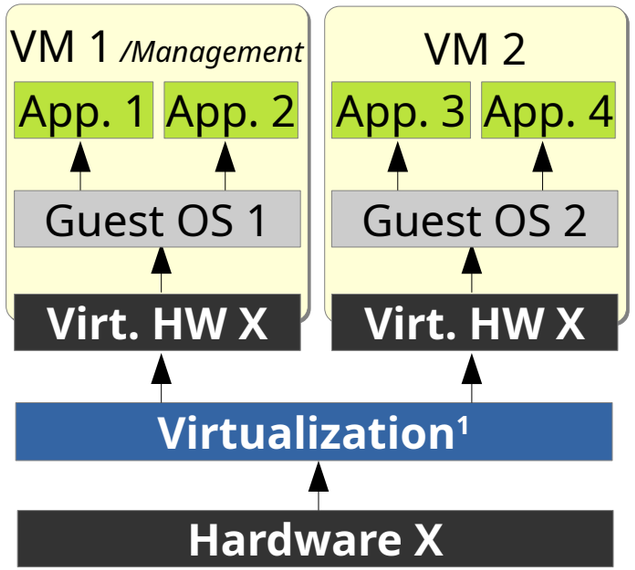
Example: Linux Container Support

- Integrated in the Linux kernel
  - Container
  - Container solutions (e.g. Docker) only have management tasks
- Separated views: **Namespaces** per task
  - … for computer names (“UTS”), processes (“PID”), mount points (“Mount”), network devices and configuration (“Network”), IPC-Objekte (“IPC”), Control Groups (“Cgroup”, see below) and system time (“Time”)
- Resource partitioning: **Control Groups** (cgroups)
  - Containers’ share of CPU time, memory and I/O bandwidth
  - Configuration interface: **cgroupfs** pseudo file system
- Efficient sharing (of files): **Overlay FS**
  - Overlay of directory trees

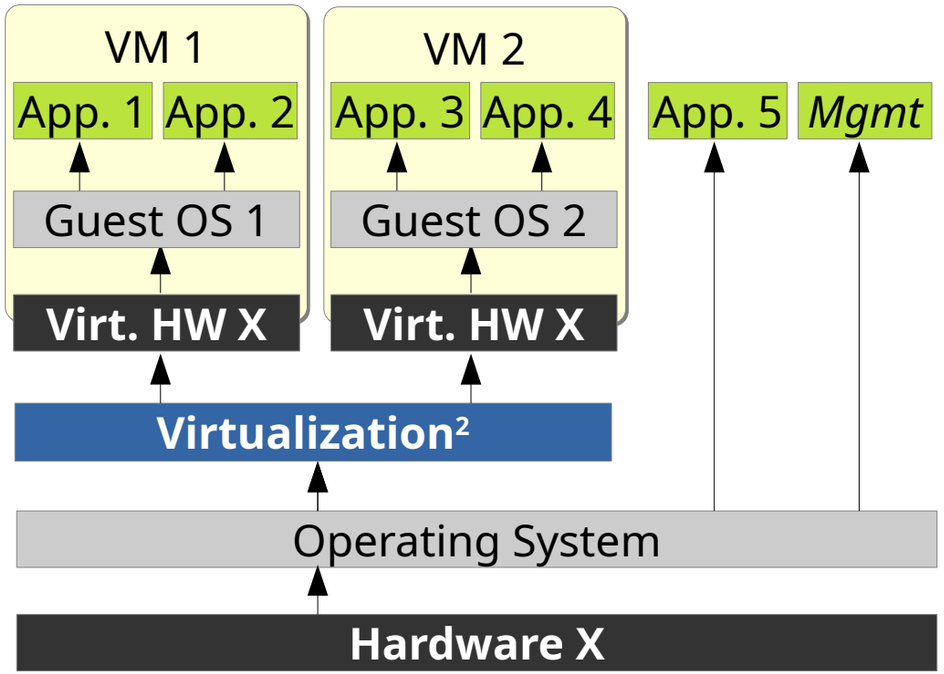


Hardware Virtualization

- A complete server (CPU, memory, I/O devices) is virtualized:



or



¹**Type-1 Hypervisors** provide virtual machines without the help of an operating system (directly on the hardware, bare-metal).

²**Type-2 Hypervisors** operate above a “host operating system”. They can use its services, e.g. virtual memory.

Overview

- Basic Terminology
- Cloud System Software
- Virtualization Basics
- **CPU Virtualization**
- Memory Virtualization
- I/O Virtualization
- Summary

CPU Virtualization (1)

- Simplest possibility: **CPU emulation** (+ multiplexing)
  - **Interpretation** or **just-in-time (JIT) compilation** of instructions of the emulated CPU
  - Examples: Bochs, QEMU (in “TCG” mode), MAME
- Emulation of an arbitrary CPU Y on a CPU X
- Main disadvantage: Execution speed

Execution Mode	T1FAST.EXE time	T1SLOW.EXE time
Native	0.26	0.26
QEMU 0.9.0	10.5	12
Bochs 2.3.5	25	31
Bochs 2.3.7	8	10

Table 3.2: Execution time in seconds of Win32 test program

FAST/SLOW: with/without code optimization

```
static int foo(int i) {
  return(i+1);
}
int main(void) {
  … <start timer>
  for(i=0; i<100000000; i++)
    t += foo(i);
  … <stop timer>
}
```

Source: [2]

Conclusion: Avoid CPU emulation if possible

CPU Virtualization (2)

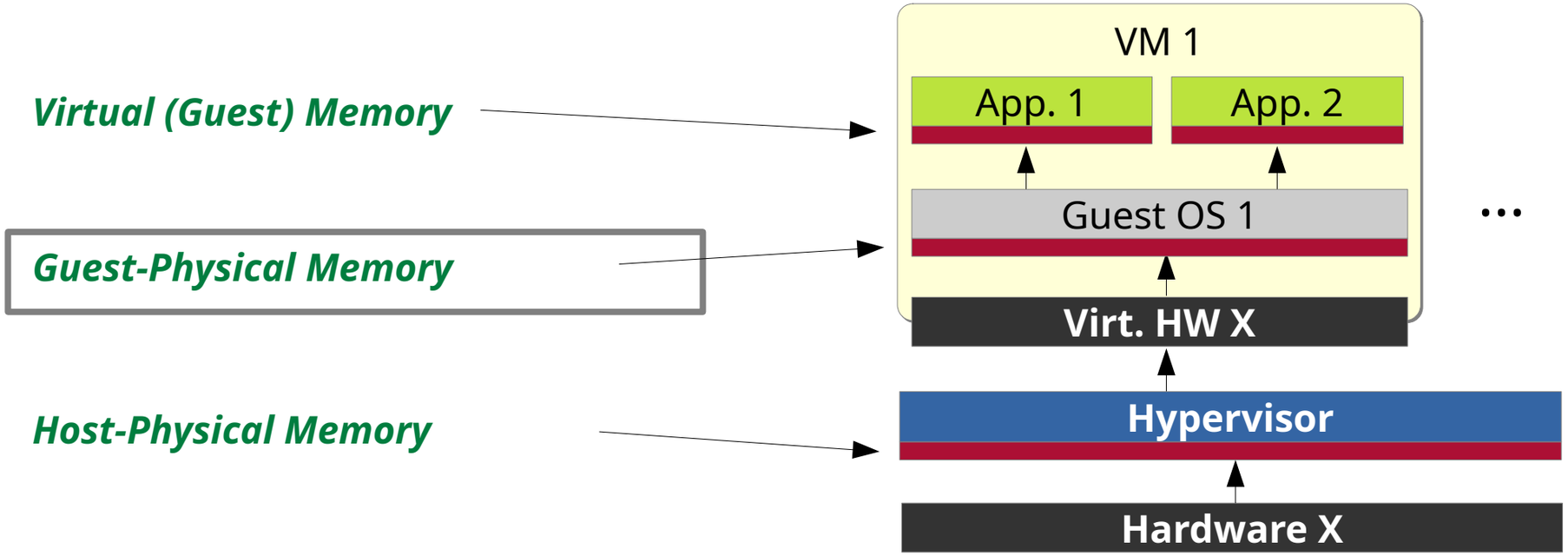
- Efficient alternative: **CPU multiplexing** (CPU X_1, ..., X_N on X)
- Desired properties ("Virtualization criteria")
  - **Equivalence**: A VM behaves like the real machine.
  - **Security**: A VM is isolated. The hypervisor has full control over the hardware.
  - **Performance**: Virtual CPUs are not (significantly) slower than real CPUs.
- Q: Which architectures are "virtualizable" like this?
- A: (Popek und Goldberg, 1974 [3]):
  - There are **"sensitive" instructions** that depend on the privileged state of the CPU (user/supervisor mode, memory mapping, ...) or change it.
  - All sensitive instructions *must lead to a trap when executed in user mode*. The hypervisor can thus emulate the instruction.
- Everything else just as in an OS: VM Scheduling

Overview

- Basic Terminology
- Cloud System Software
- Virtualization Basics
- CPU Virtualization
- **Memory Virtualization**
- I/O Virtualization
- Summary

Memory Virtualization (1)

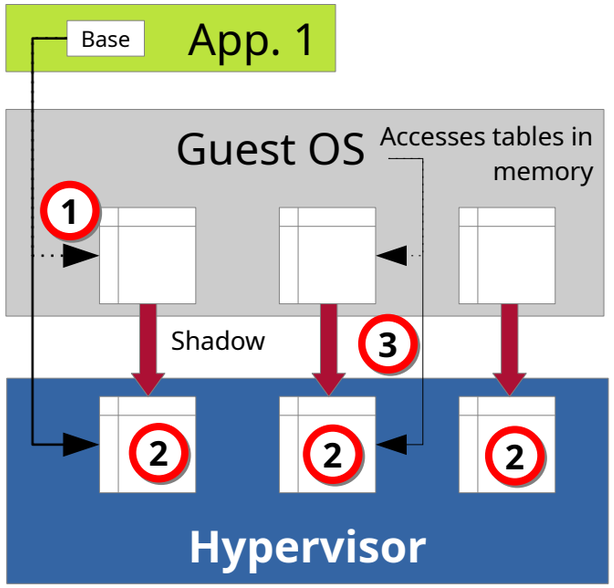
- Problem: Additional address-mapping level



Guest operating systems "believe" they have full control over the hardware. They use (guest-)physical page frames at will. Without an additional mapping step, overlaps with other guest OSs would be possible!

Memory Virtualization (2)

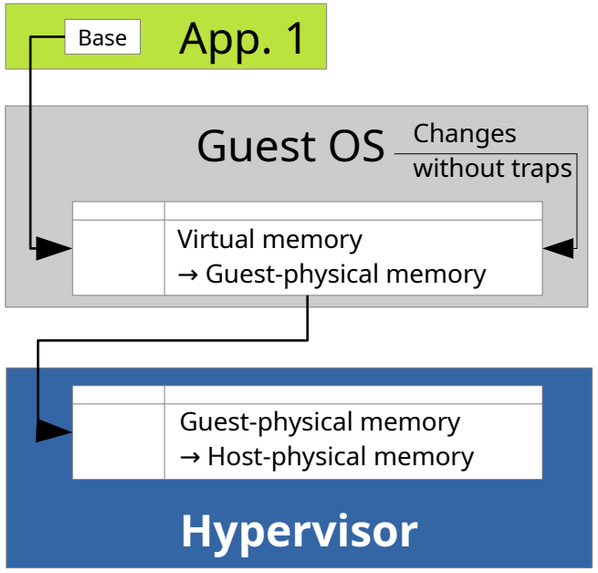
- **Solution 1**: **Shadow Page Tables**
  - No special virtualization support in hardware necessary
- **Approach**:
  1. Guest-OS page tables are not used at all
  2. Hypervisor: Shadow table for each page table
  3. Shadow table must be kept up-to-date!
     - Variant 1: Every write access to memory in which a page table is located must be intercepted and interpreted.
     - Variant 2: Ignore changes; synchronize tables in the event of page faults
     - Both variants lead to many traps to the hypervisor → *Overhead*

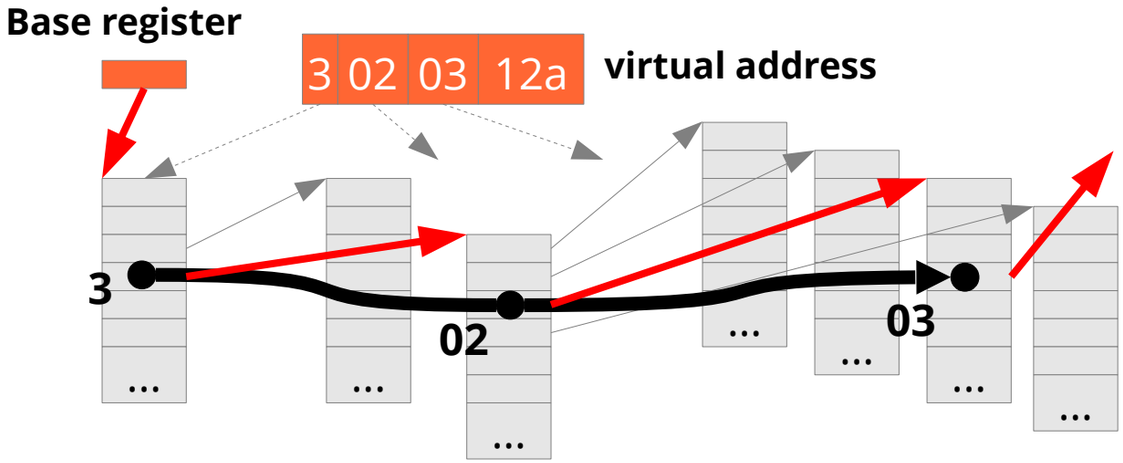


Shadow page tables are expensive. The only remedy is paravirtualization or hardware support.

Memory Virtualization (3)

- **Solution 2**: **Nested Page Tables**
  - AMD terminology; Intel: *Extended Page Tables*
- **Approach**:
  - Hardware takes over complete address mapping.
  - Guest OS can change its page tables at any time
  - **Page table walk** now more expensive → TLB more relevant
    - Page tables have a tree structure
    - Pointers to page tables are guest-physical addresses
    - Mapping to host-physical addresses necessary (here: **4x**)





Memory Virtualization (4)

What else is possible …

- **Ballooning**: Trick for dynamic memory allocation to VMs
  - Small kernel module communicates with hypervisor,
  - if required it can reserve memory of the OS kernel
  - which can be given to other VMs.
- **Deduplication**: Detection and avoidance of page duplicates across VMs. Saves memory, especially with the same guest OS.
- **VM migration**
  - Complete VM memory state is transferred to other host
  - Optimization: Transfer while the VM is still running
    - While transfer is in progress, further changes are recorded with **dirty bits** in page table.
- **VM replication**
  - State changes in memory are periodically transferred to a backup host. Backup VM quickly takes over in the event of a server failure.

Overview

- Basic Terminology
- Cloud System Software
- Virtualization Basics
- CPU Virtualization
- Memory Virtualization
- **I/O Virtualization**
- Summary

I/O Virtualization (1)

- Simplest possibility: **I/O emulation** (+ multiplexing)
  - I/O register accesses are privileged instructions or can be caught by the hypervisor using the MMU (**“trap and emulate”**)
- Emulation of any I/O device Y using I/O device X
  - Example Oracle VirtualBox: PS/2 **mouse/keyboard**; IDE, SATA, SCSI, … **hard disk**; SVGA **graphics**; several AMD and Intel **networking cards**; **USB** Host Controller; AC‘97, Intel HD or Soundblaster 16 **Audio**
- Main disadvantage: I/O throughput
  - Even simple I/O operations require hundreds or thousands of I/O register accesses!

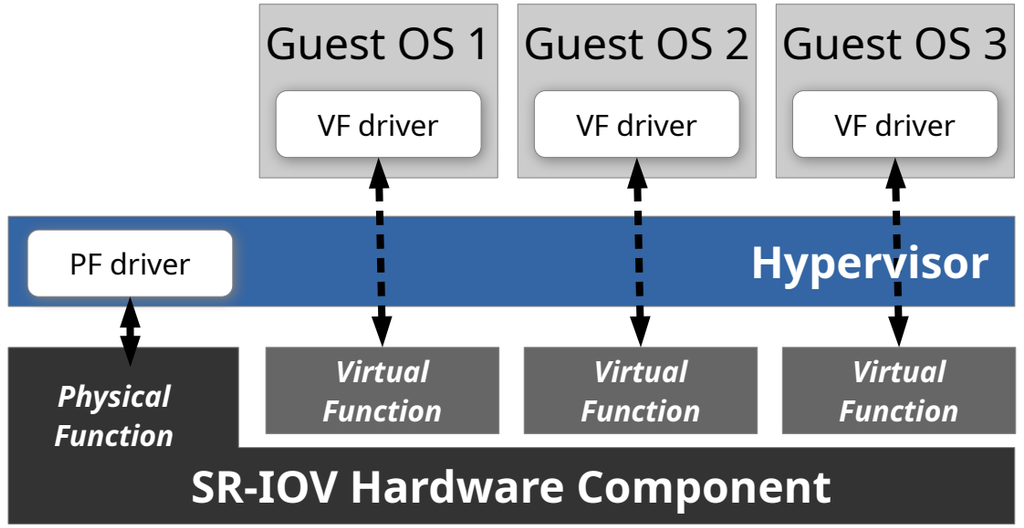
I/O emulation is expensive. Paravirtualization or hardware support provides a remedy (again).

I/O Virtualization (2)

- Alternative 1: No multiplexing but **device passthrough**
  - Device is assigned exclusively to exactly one VM
  - Any register accesses are permitted (without trap)
- Disadvantages:
  - DMA addresses are physical host addresses that the VM does not know
    - Isolation could be broken
  - Interrupts could be triggered on the “wrong” CPU.
- Solution: Input–output memory management unit (**IOMMU**)
  - Hardware extension (CPU / mainboard chipset)
  - With DMA, **address mapping** takes place using tables
    - Acceleration through own TLBs
  - **Interrupt remapping** can change interrupt number and target CPU

I/O Virtualization (3)

- Alternative 2: PCIe **Single Root I/O Virtualization** (SR-IOV)
  - **Hardware mechanism**: One device appears like many
    - Multiple I/O register sets,
      multiple interrupt configurations, …
  - Hypervisor maps one of these devices to a VM and is out of the loop from then on.



- Possible disadvantage:
  - Hardware controls VM prioritization
    - e.g. Round Robin
  - Possible conflicts with/contradictions to the hypervisor's priorities

Overview

- Basic Terminology
- Cloud System Software
- Virtualization Basics
- CPU Virtualization
- Memory Virtualization
- I/O Virtualization
- **Summary**

Summary

- **Virtualization** is an important **recurring architectural concept** in the system software stack
  - Transparent: multiplexing, aggregation, emulation
- Hardware virtualization (in the sense of Popek/Goldberg)
  - removes the rigid link between hardware and software
    - e.g. VM migration and replication at runtime
  - Technical basis of cloud computing
- Operating systems for clouds
  - As before: resource management and abstractions
  - But at a higher level

Literature

[1] Edouard Bugnion, Jason Nieh, and Dan Tsafrir. 2017. *Hardware and Software Support for Virtualization*. Morgan & Claypool Publishers, 2017.

[2] Mihočka, Darek, Stanislav Shwartsman and Intel Corp. *Virtualization Without Direct Execution or Jitting: Designing a Portable Virtual Machine Infrastructure,* 2008.

[3] Gerald J. Popek and Robert P. Goldberg. 1974. *Formal requirements for virtualizable third generation architectures*. Commun. ACM 17, 7 (July 1974), 412–421. DOI: 10.1145/361011.361073