Cloud Computing Systems

Lecture 8

Nuno Preguiça

(nuno.preguica_at_fct.unl.pt)

OUTLINE

Virtualization.

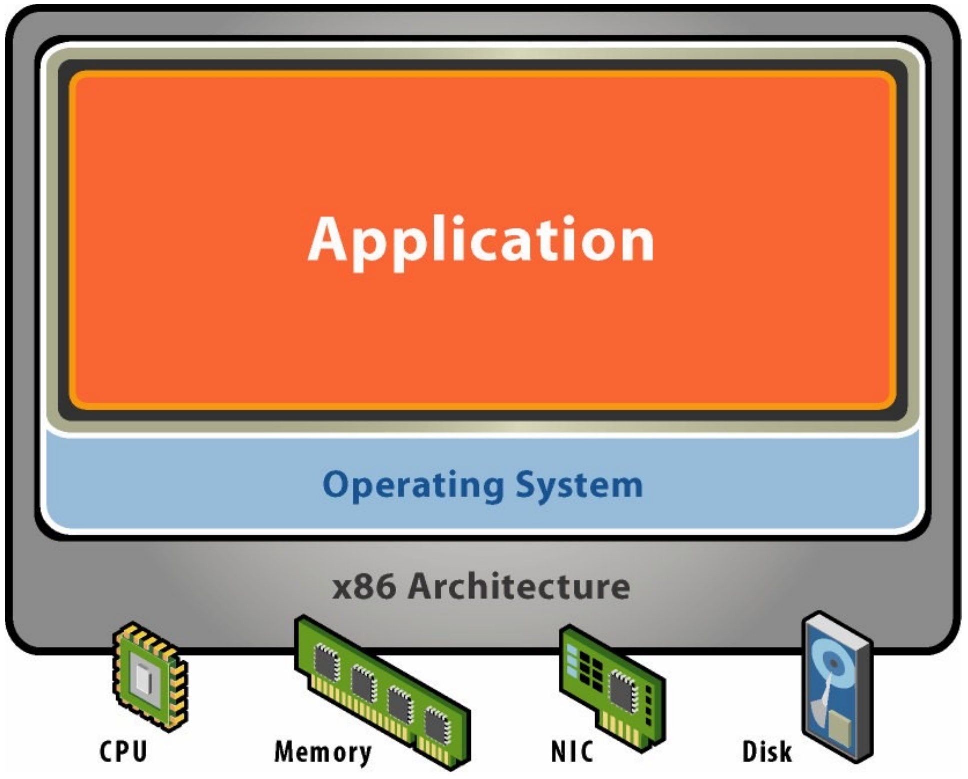
IaaS. Azure VM.

What is virtualization?

Virtualization consists in creating simulated (or virtual) computing resources. These computing resources can include network, storage, computing (CPU+memory).

Virtualization is one of the key techniques used to realize cloud computing platforms.

WHAT IS VIRTUALIZATION? (2)



Physical Hardware

Processors, memory, chipset, I/O bus and devices.

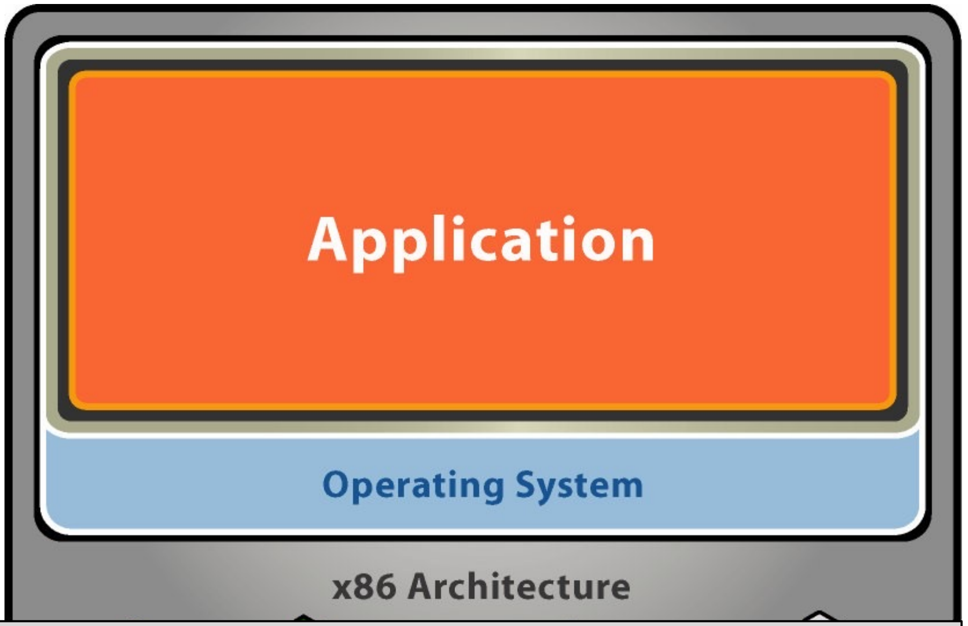
Software

Single OS image.

OS controls hardware.

One or multiple applications.

WHAT IS VIRTUALIZATION? (2)



OS provides, for each application, the illusion that the application is the only one using the computer. This can also be considered a form of virtualization.

Physical Hardware

Processors, memory, chipset, I/O bus and devices.

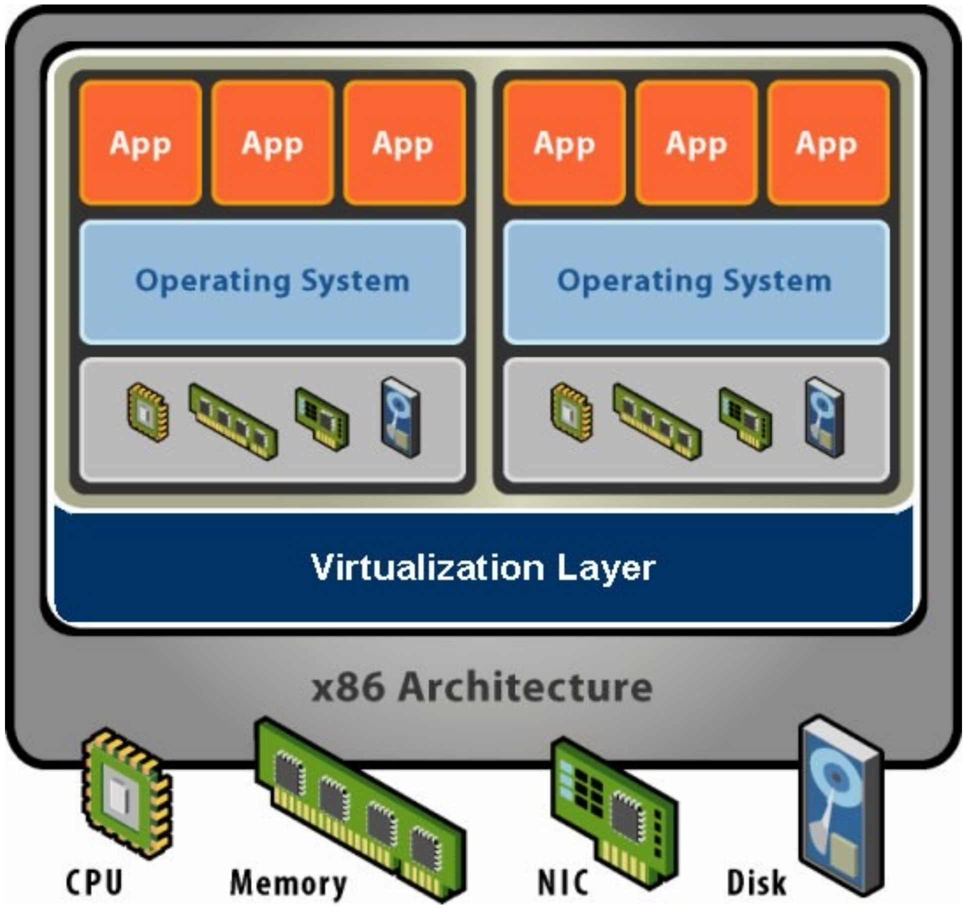
Software

Single OS image.

OS controls hardware.

One or multiple applications.

What is Virtualization? (3)



Virtualization Software

Extra level of indirection decouples hardware and OS

Multiplexes physical hardware across multiple guest VMs

Manages physical resources.

Hardware Level Abstraction

Guest VMs use virtualized hardware.

Why to virtualize?

Provide a better abstraction for users of shared resources.

- E.g.: a virtual machine provides the abstraction of a machine to each of the users of a physical machine.

Higher and more flexible utilization of resources.

- Allows to share the same physical resources among multiple users.
- Decoupling between the physical and virtual resources.
- Allows to share the resources among multiple users.

ALTERNATIVE FORMS OF VIRTUALIZATION

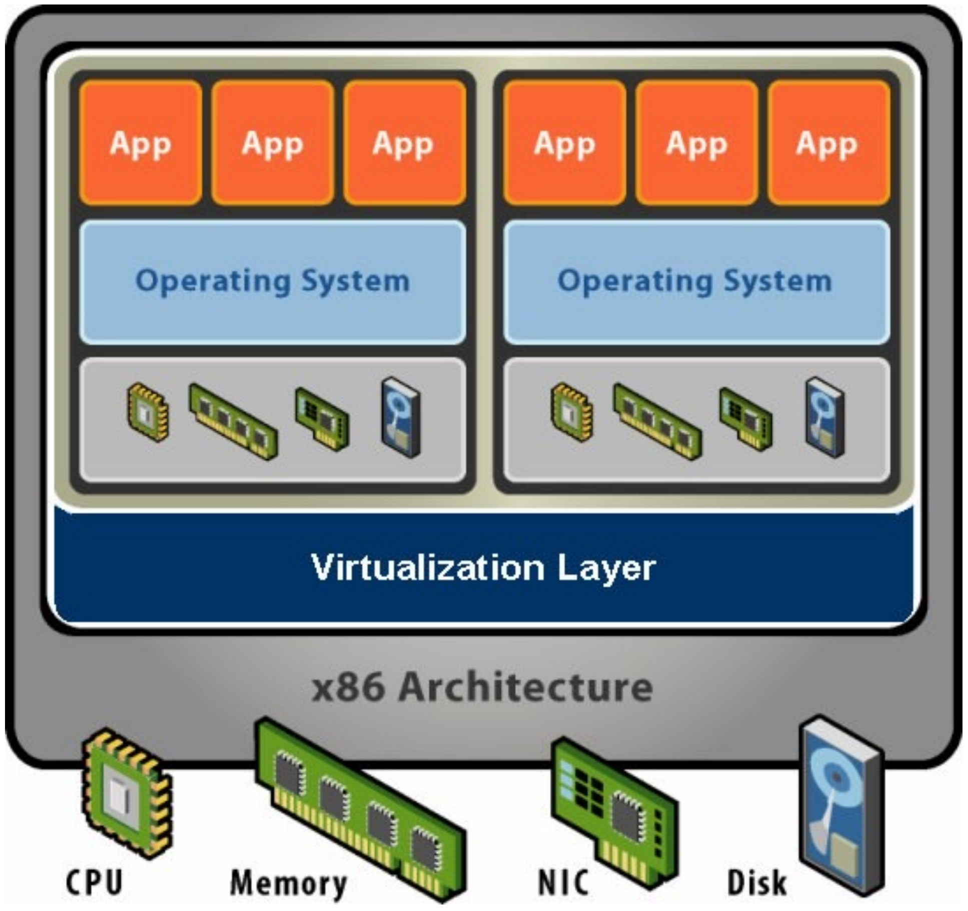
- System virtual machine.

VIRTUAL MACHINE MONITOR (VMM)

The VMM provides provides the illusion of a machine identical with the original machine. The VMM controls the hardware.

Software runs in one VM in complete isolation from other VMs, with only minor decreases in speed.

This idea has been around since the 60s in IBM mainframes.



VVM, VM, HYPERVISOR...

Virtual Machine (VM)

- It is the simulation of the "real machine", with a "virtual hardware" created/managed by the VMM.
- There may exist several VM instances running, and they:
  - behave as if they were distinct, isolated, machines;
  - run their applications unmodified and efficiently.

The Virtual Machine Monitor (VMM)

- Provides the virtual hardware abstractions to the VM;
- A (VM+VMM) pair constitutes an hypervisor "process".

The hypervisor

- It is an operating system that runs the (VM/VMM) processes.
- Sometimes VMM and hypervisor terms are used interchangeably.

ALTERNATIVE FORMS OF VIRTUALIZATION

- System virtual machine.
- Process virtual machine.

LANGUAGE VIRTUAL MACHINE

Provides an environment with a specific instruction set and standard libraries for accessing the computer resources.

Runs as an application in an OS.

Programs are compiled for the language VM code and run in the language VM.

Examples?

Java VM, CLR, etc.

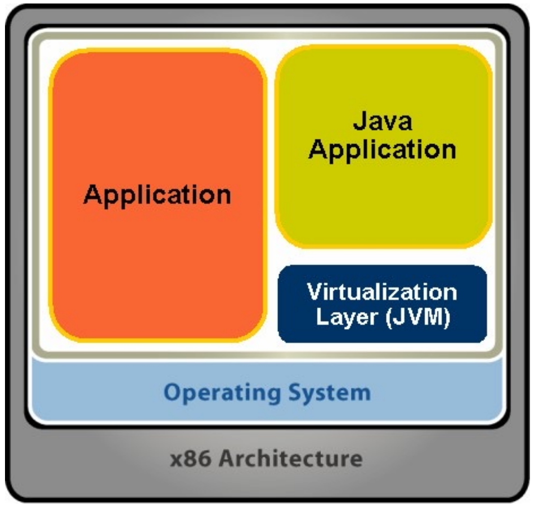
ALTERNATIVE FORMS OF VIRTUALIZATION

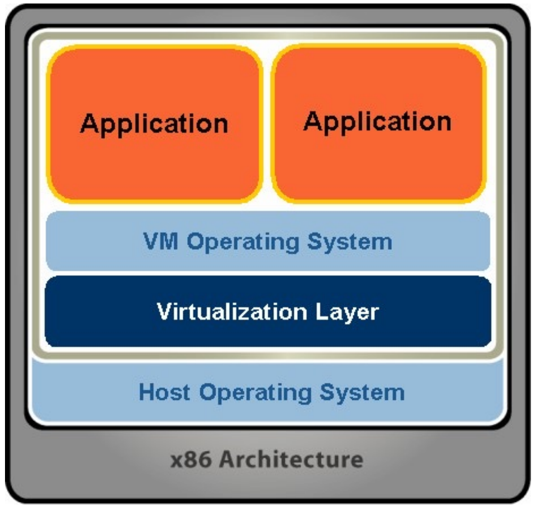
Virtual machines abstract using a layer at different places.

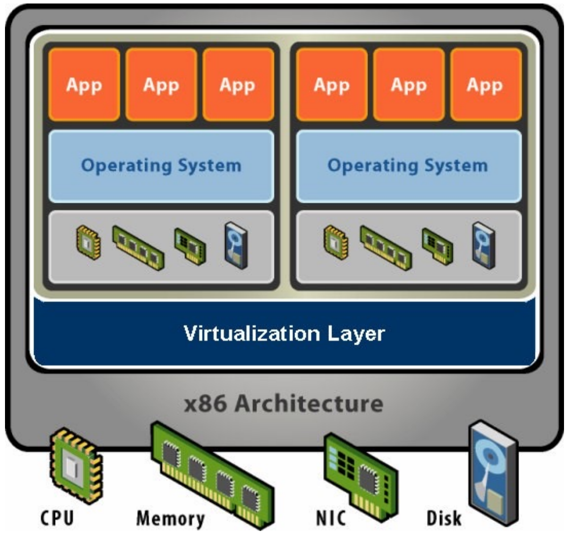
Language Level

OS Level

Hardware Level







Taxonomy from Jerry Breecher.

ALTERNATIVE FORMS OF VIRTUALIZATION TAXONOMY

System Virtualization

Hardware Level

Bare-Metal/ Hypervisor

- HP Integrity VM
- IBM zSeries z/VM
- VMware ESX Server
- Xen

Hosted

- Microsoft Virtual Server
- Microsoft Virtual PC
- Parallels Desktop
- VMware Player
- VMware Workstation
- VMware Server

Para-virtualization

High-Level Language

- Java
- Microsoft .NET / Mono
- Smalltalk

- FreeBSD Jail
- HP Secure Resource Partitions

Emulators

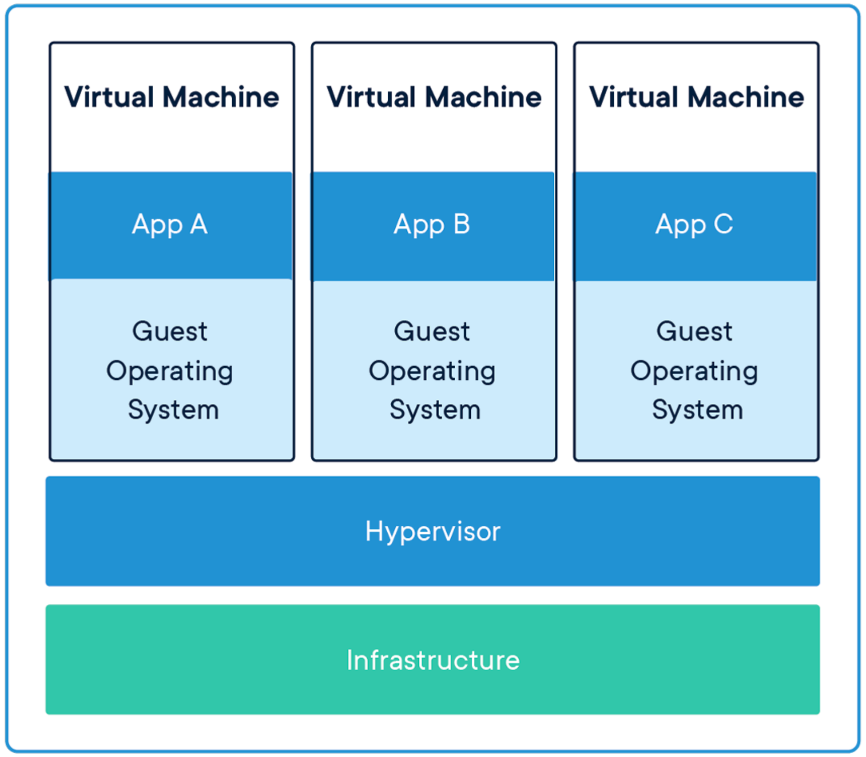
- Bochs
- Microsoft VPC for Mac
- QEMU

Taxonomy from Jerry Breecher.

Virtualization: under the hood

Virtualization allows running one OS inside/on top of another OS

- Host OS/Hypervisor: running on the bare metal machine
- Guest OS: running inside/on top of the Host OS



Requirements for virtualisation (1)

The VMM (or hypervisor) should provide:

- **Safety**: the VM (guest OS/application) should not be able to compromise the VMM, which should have full control of the resources (real and virtual).
- **Fidelity**: when running, the behaviour of software on the VM should be identical to the behaviour on real hardware.
- **Efficiency**: the VMM should interpose itself as little as possible while the VM is running its code.

REQUIREMENTS FOR VIRTUALISATION (2)

How does the VMM protect itself (from VMs)?

- The VMM could maintain the state of the processor, memory, IO and interpret each instruction of the VM, updating the state of the processor/memory/IO accordingly.
- This is safe, but very slooowwww…
- This is called an **emulator**, and it is necessary to run programs compiled for a different processor.

Requirements for virtualisation (3)

How does the VMM protect itself (from VMs)?

- For being fast, it is necessary that the VM software runs directly in the processor, but without compromising the system.
- OS do this to a certain extent – a process cannot arbitrarily access the computer resources. How?

1. Memory isolation between processes;
2. Privileged operations that would allow a process to access computer resources can only be executed in supervisor mode (S-mode) – the OS code is the only code that runs in supervisor mode, guaranteeing that the OS can control the resources accessed by a process.

REQUIREMENTS FOR VIRTUALISATION (4)

Can VMMs use the same approach?

- Not immediately if there are only two modes (S-mode and U-mode/user mode) …
  - If the VM guest OS runs in privileged mode, it could compromise the VMM as it would have access to all instructions;
  - If the VM guest OS runs in user mode, it could not run correctly, as it could not run the necessary privileged instructions.
- What is needed?

1. The VM guest OS needs to think it is running in supervisor mode – need to see the S(upervisor) flag;
2. The VM guest OS needs to be able to issue privileged instruction, but its execution needs to be controlled by the VMM.

VIRTUALISATION WITH TRAP-AND-EMULATE

“Virtualisation friendly” architectures (e.g. IBM 370)

- Privileged instructions trap if executed in user-mode
  1. The VM runs user code directly on the CPU (no performance penalty)
  2. The guest OS runs in User-mode (but the vCPU flags indicate S-mode – so, the guest OS thinks it is running in User-mode)
  3. When the guest OS executes a privileged instruction, a trap changes the execution to VMM (in a trap, the execution of the guest OS is stopped, and a function of the VMM is called).
  4. The VMM interprets (emulates) the instruction and returns the execution to the guest OS (in user-mode).
- All sensitive operations (those that behave differently when executed in U and S-modes) are privileged.

VIRTUALISATION WITH BINARY TRANSLATION

“Virtualisation unfriendly” architectures (e.g. Intel x86)

- Some sensitive instructions are not privileged, e.g., PUSHF, POPF.
- **Problem**: if the guest OS runs in U-mode (with vCPU S-mode flag), the effect of executing the operation is not correct.
- **Idea**: replace the sensitive instruction by a trap (forcing controlled execution in the VMM).

Virtualisation with Binary Translation

“Virtualisation unfriendly” architectures (e.g. Intel x86)

Resorts to **binary translation** (VMware earliest versions)

- The VM runs user code directly on the CPU (no performance penalty).
- The guest OS runs in U-mode (but the vCPU flags indicate S-mode).
- Guest OS binary code is fed to a translator that outputs changed binary code for sensitive instructions - non-sensitive and privileged instructions are not modified.
- Privileged instructions are run as trap-and-emulate.

Drawbacks

- Slowdown in guest OS code – need to translate code; several instructions may need to be trapped in each guest OS function;

VIRTUALISATION WITH PARAVIRTUALISATION

“Virtualisation unfriendly” architectures (e.g. Intel x86)

Problem: Even if code translation can be fast (e.g. VMware implemented a very efficient translator with a translation cache), having multiple instructions of each guest OS function trapped slows execution.

Idea: why not replacing these guest OS functions by a VMM function that does the same functionality? A single trap would be necessary to call the VMM function. Furthermore, the VMM function can access the resources directly (instead of accessing a virtualized version).

VIRTUALISATION WITH PARAVIRTUALISATION

“Virtualisation unfriendly” architectures (e.g. Intel x86)

Resort to **paravirtualisation** (Xen earliest versions)

- Guest OS source code is changed so that portions that use sensitive unprivileged and privileged instructions are replaced with hypervisor calls that execute the same functionality – e.g. a guest OS function that would access the virtualized virtual memory controller is replaced by a VMM/hypervisor call that accesses the virtual memory controller directly;
- The VM runs user code directly on the CPU (no performance penalty).
- The guest OS runs in U-mode (but the vCPU flags indicate S-mode).
- Hypervisor calls are implemented as traps.

Pros and cons of Paravirtualization

Pros:

- Performance within a few percent of un-virtualized case.

Cons:

- Requires changes to the guest OS: ok when source is available (e.g. Linux).

VIRTUALIZATION TODAY: HARDWARE SUPPORT

Intel VT and AMD-V

- VMs are run in HW-provided containers.
- ALL sensitive instructions are privileged:
  - trap-and-emulate
- Lots of CPUs have it (since ~2006)

HW support for memory virtualization

- Intel EPT (Extended Page Tables)
- AMD RVI (Rapid Virtualization Indexing)

VIRTUALIZATION TODAY

Paravirtualization solved the problem of performance when binary translation was used.

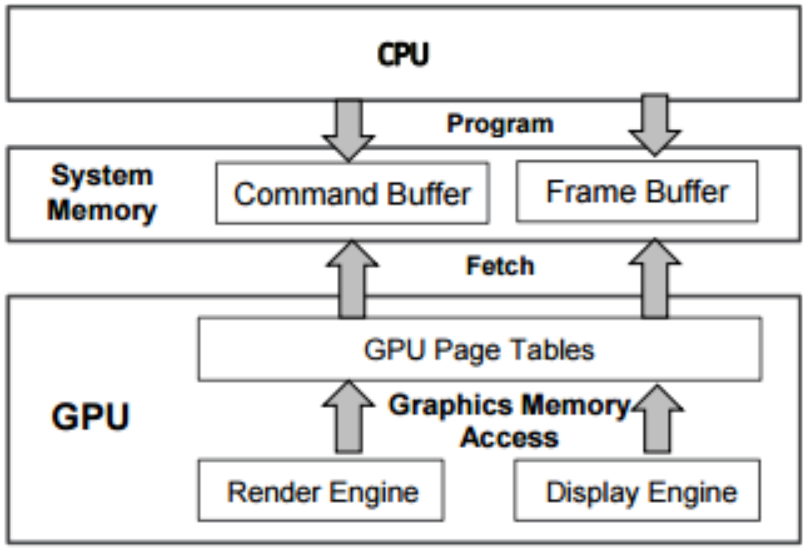
Not needed with the hardware support for virtualization.

- Used only to the few cases where the trap-and-emulate overhead is too high for IO or storage.

GPUs

GPUs have become important for several applications: gaming, ML, HPC, etc.

The CPU and GPU “communicate” by using the system’s memory for sending commands and data.



VIRTUALIZATION AND GPUS

Different techniques have been proposed for GPU virtualization:

- Backend virtualization

  “Back-end techniques run the graphics driver stack inside the virtual machine with the virtualization boundary between the stack and physical GPU hardware”

- Frontend virtualization

  “Front-end virtualization introduces a virtualization boundary at a relatively high level in the stack, and runs the graphics driver in the host/hypervisor”

GPU PCI PASSTHROUGH

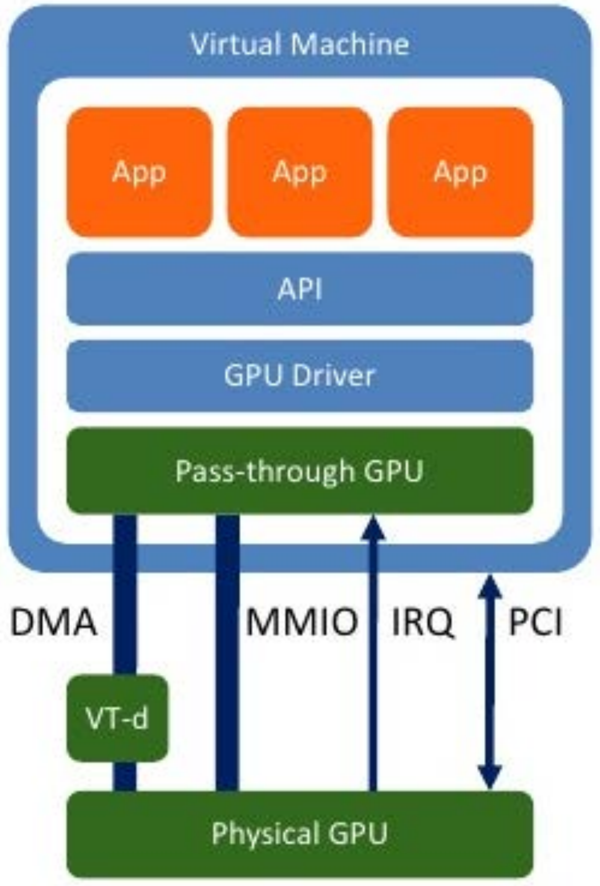
PCI passthrough is a backend technique that allows a virtual machine to have direct access to the GPU by mapping its memory regions into the VM’s address space.

Advantage:

- Near-native performance for GPUs

Problems:

- No way of multiplexing.

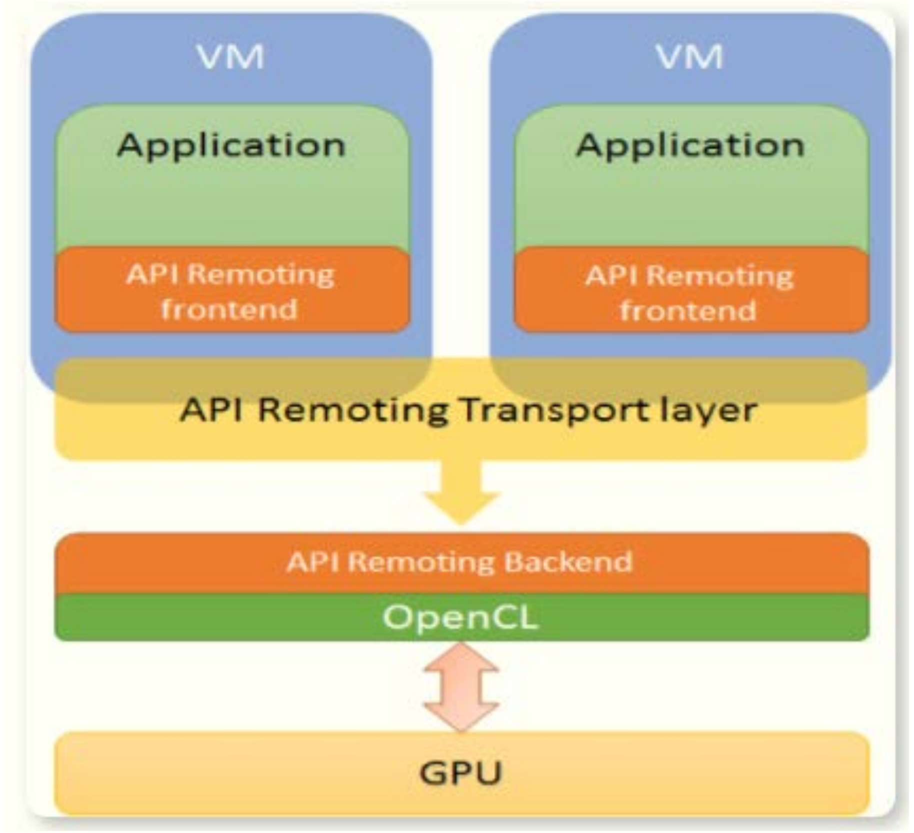


This is currently supported with NVIDIA GPUs in cloud platforms from Alibaba, AWS, Azure, Google and Oracle.
https://docs.nvidia.com/grid/cloud-service-support.html

FRONTEND TECHNIQUES

In frontend techniques, the GPU is controlled by the host/hypervisor. These techniques typically support multiplexing by relying on the independent contexts that can be maintained by GPUs.

In **API remoting**, the graphics API calls in the guest are forwarded to the external graphics stack via remote procedure calls.



FRONTEND TECHNIQUES

In **device emulation**, a virtual GPU is emulated and the emulation synthesizes host graphics operations in response to actions by the guest device drivers.

These solutions allow the GPU to run in a different machine.

OUTLINE

Virtualization.

IaaS. Azure VM.

Types of cloud services: IaaS vs. PaaS vs. SaaS

Infrastructure as a Service (IaaS)

- Provides processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software. [NIST]
- E.g.:
  - VMs, disks, private virtual networks.
  - IaaS public providers: Amazon (EC2); Google (Compute Engine); Microsoft (Azure); etc.
  - Software for private IaaS: OpenStack (open software), vCloud (VMware), etc.

AZURE IAAS

Supports the creation of virtual machines running Linux and Windows.

Supports the creation of VMs specially tailored for running specific software (e.g. SAP).

- Recent numbers show that there are more Linux VMs in Azure than Windows VMs.

AZURE VMS

The user selects the VMs he needs. Needs to specify:

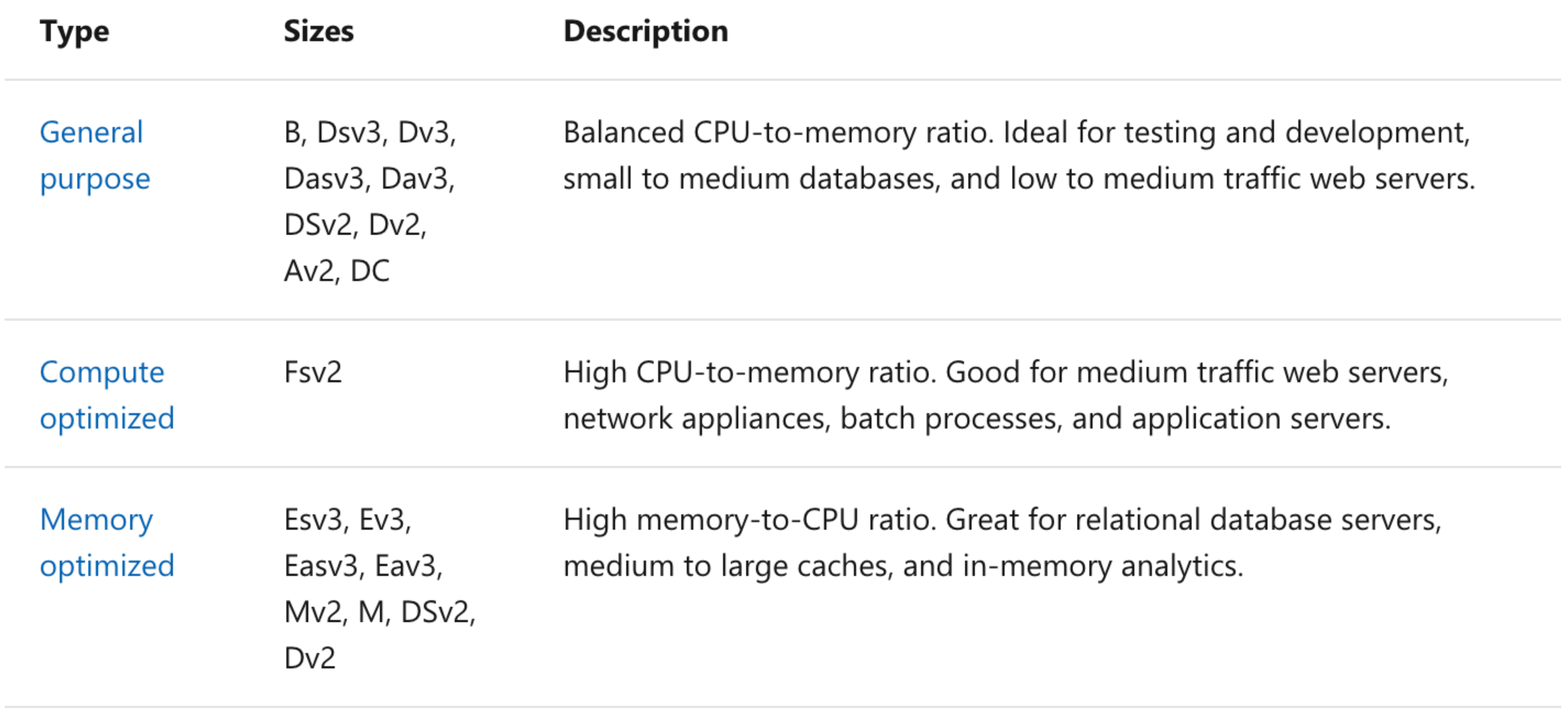
- Type of each VM;
- Select the regions to deploy;
- Select the availability options;
- Disks and network options.

Azure VMs

The user selects the VMs he needs. Needs to specify:

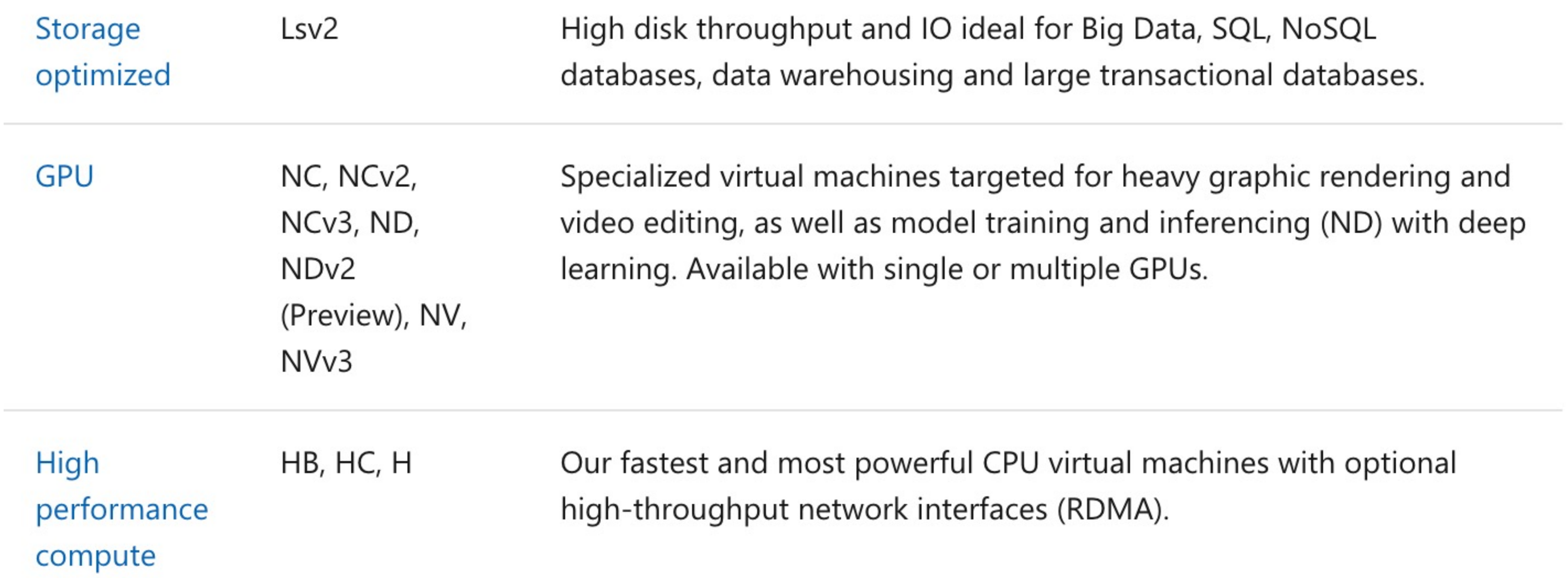
- **Type of each VM;**
- Select the regions to deploy;
- Select the availability options;
- Disks and network options.

Types of virtual machines



Type	Sizes	Description
General purpose	B, Dsv3, Dv3, Dasv3, Dav3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, Easv3, Eav3, Mv2, M, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.

TYPES OF VIRTUAL MACHINES (2)



Storage optimized	Lsv2	High disk throughput and IO ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NC, NCv2, NCv3, ND, NDv2 (Preview), NV, NVv3	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	HB, HC, H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

AZURE VMS

The user selects the VMs he needs. Needs to specify:

- Type of each VM;
- **Select the regions to deploy;**
- Select the availability options;
- Disks and network options.

Azure VMs

The user selects the VMs he needs. Needs to specify:

- Type of each VM;
- Select the regions to deploy;
- **Select the availability options;**
- Disks and network options.

VM AND FAULTS

Unplanned Hardware Maintenance Event occurs when the Azure platform predicts that the hardware or any platform component associated to a physical machine, is about to fail.

In this case, the platform issues an unplanned hardware maintenance event and may use **Live Migration** to migrate the Virtual Machines from the failing hardware to a healthy physical machine.

What is Live Migration?

- It is the process of migrating a running VM to other computer with minimal impact on performance. Idea: pauses the VM; migrates memory, open files, and network connections; and resumes the VM.
- Most advanced systems start migrating memory etc. while still running, pausing only for a very short period in the end to migrate what has been changed meanwhile.

VM AND FAULTS (2)

An Unexpected Downtime is when the hardware or the physical infrastructure for the virtual machine fails unexpectedly. This can include local network failures, local disk failures, or other rack level failures.

When detected, the cloud platforms automatically migrates the virtual machine to a healthy physical machine in the same datacenter. During the healing procedure, virtual machines experience downtime (reboot) and in some cases loss of the temporary drive. The attached OS and data disks are always preserved.

- Why is it necessary to reboot?
- Memory is lost.

Virtual machines can also experience downtime in the unlikely event of an outage or disaster that affects an entire datacenter, or even an entire region.

VM AND FAULTS (3)

Planned Maintenance events are periodic updates made to the underlying cloud platform to improve overall reliability, performance, and security of the platform infrastructure. Why is this necessary?

Cloud platforms attempt to use VM Preserving Maintenance in all possible occasions, i.e., to keep the VMs running (e.g. by live migrating them to other machines).

AVAILABILITY ZONE

Each Availability Zone has a distinct power source, network, and cooling. An application should use replicated apps and data in different zones for tolerating the loss of a datacenter.

Fault domains

A fault domain is a logical group of underlying hardware that share a common power source and network switch, similar to a rack within an on-premises datacenter.

Update domains

An update domain is a logical group of underlying hardware that can undergo maintenance or be rebooted at the same time.

SCALE SET

Azure virtual machine scale sets allow to create and manage a group of load balanced VMs. The number of VM instances can automatically increase or decrease in response to demand or a defined schedule.

Virtual machines in a scale set can be deployed across multiple regions and fault domains to maximize availability and resilience to outages.

Availability Set

An availability set is a logical grouping of VMs within a datacenter to provide redundancy and availability.

In an availability set, VMs are automatically distributed across different fault and update domains. This approach limits the impact of potential physical hardware failures, network outages, or power interruptions.

Azure VMs

The user selects the VMs he needs. Needs to specify:

- Type of each VM;
- Select the regions to deploy;
- Select the availability options;
- **Disks and network options.**

MANAGED DISK

A managed disk is a virtual hard disk (VHD).

Managed disks are stored as page blobs. They are an abstraction over page blobs, blob containers, and Azure storage accounts.

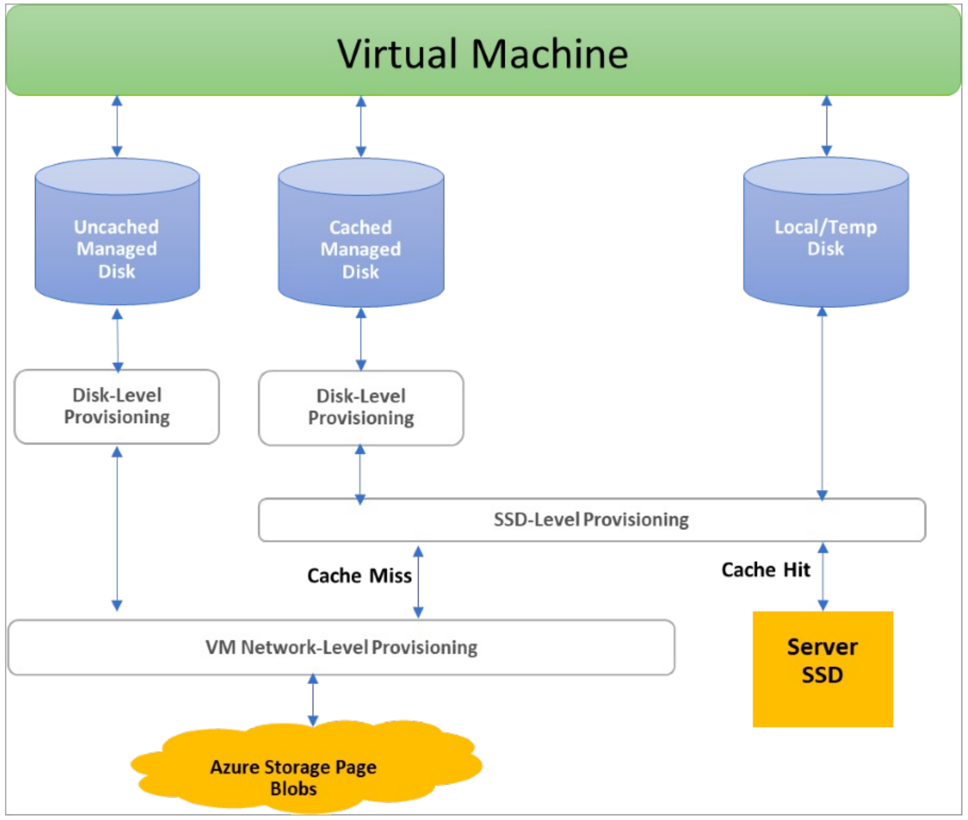
The types of disks are: Ultra disk, Premium solid state drive (SSD), Standard SSD, and Standard hard disk drive (HDD).

So, disks are remote…. that should be very slow, no?

DISK ALLOCATION AND PERFORMANCE

The system provisions per-disk IOPS and bandwidth. Additionally, it provisions network for remote access.

Locally, the server implements SSD provisioning for data stored at server's SDD, including disk with cache and local and temp disks.



To know more

Modern Operation Systems, A. Tannenbaum, H. Bos, 4th edition, sec 7 – sec. 7.4. (the rest of the chapter is also worth reading, but we have not addressed those topics in the course)

https://docs.microsoft.com/en-us/azure/virtual-machines/linux/manage-availability

M. Dowty, et. al. GPU Virtualization on VMware's Hosted I/O Architecture. https://graphics.stanford.edu/~yoel/notes/gpu-wiov-web.pdf

ACKNOWLEDGMENTS

Some text and images from Microsoft Azure online documentation and AWS online documentation.

Some slides based on a previous version by Paulo Lopes and Vitor Duarte.