Computing Infrastructures - Notes

260236

July 2024

Preface

Every theory section in these notes has been taken from two sources:

- The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition. [1]
- Quantitative System Performance: Computer System Analysis Using Queueing Network Models. [2]

About:

GitHub repository

Contents

1 Introduction: definition of Data Center and Computing Infrastructure 4

2 Hardware Infrastructures 5
2.1 System-level . . . 5
2.1.1 Computing Infrastructures and Data Center Architectures 5
2.1.1.1 Overview of Computing Infrastructures . . . 5
2.1.1.2 The Datacenter as a Computer . . . 10
2.1.1.3 Warehouse-Scale Computers . . . 11
2.1.1.4 Multiple Data Centers . . . 12
2.1.1.5 Warehouse-Scale Computing / Data Centers Availability . . . 13
2.1.1.6 Architectural overview of Warehouse-Scale Computing . . . 13
2.2 Node-level . . . 15
2.2.1 Server (computation, HW accelerators) . . . 15
2.2.1.1 Rack Servers . . . 17
2.2.1.2 Blade Servers . . . 18
2.2.1.3 Machine Learning . . . 19
2.2.2 Storage (type, technology) . . . 22
2.2.2.1 Files . . . 23
2.2.2.2 HDD . . . 26
2.2.2.3 SSD . . . 29
2.2.2.4 RAID . . . 29
2.2.3 Networking (architecture and technology) . . . 29

3 Software Infrastructure 30
3.1 Virtualization . . . 30
3.2 Computing Architectures . . . 31

4 Methods 32
4.1 Reliability and availability of datacenters . . . 32
4.2 Disk performance . . . 33
4.2.1 HDD . . . 33
4.3 Scalability and performance of datanceters . . . 38

Index 40

1 Introduction: definition of Data Center and Computing Infrastructure

There's no single definition of a Data Center, but it can be summarized as follows.

Definition 1

Data Centers are buildings where multiple servers and communication gear are co-located because of their common environmental requirements and physical security needs, and for ease of maintenance. [1]

Definition 2

A **Computing Infrastructure** (or IT Infrastructure) is a technological infrastructure that provides hardware and software for computation to other systems and services.

Traditional data centres have the following characteristics:

- **Host a large number** of relatively small or medium sized **applications**;
- Each **application is running on a dedicated HW infrastructure** that is de-coupled and protected from other systems in the same facility;
- **Applications tend not to communicate each other**.

Those **data centers host** hardware and software **for multiple organizational units** or even **different companies**.

2 Hardware Infrastructures

2.1 System-level

2.1.1 Computing Infrastructures and Data Center Architectures

2.1.1.1 Overview of Computing Infrastructures

A number of computing infrastructures exist:

- **Cloud** offers virtualized computing, storage and network resources with highly-elastic capacity.
- **Edge Servers** are on-premises hardware resources that perform more compute-intensive data processing.

  In other words, an edge server is a piece of hardware that performs data computation at the end (or "edge") of a network. Like a regular server, an edge server can provide compute, networking, and storage functions.[1]
- **IoT** and **AI-enabled Edge Sensors** are hardware devices where the data acquisition and partial processing can be performed at the edge of the network.

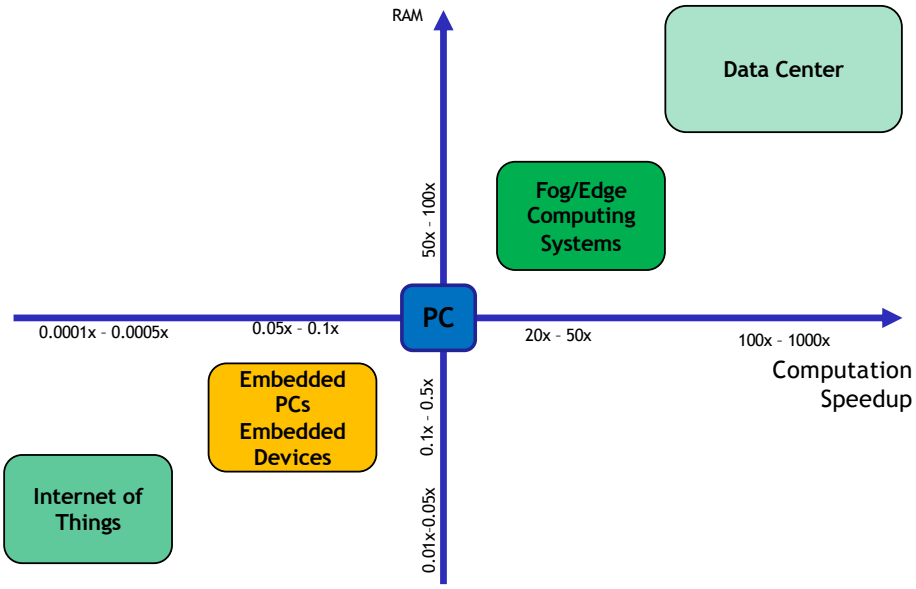


Figure 1: An **example** of Computing Infrastructures. [4]

The **Computing Continuum**, a novel paradigm that extends beyond the current silos of cloud and edge computing, can enable the seamless and dynamic deployment of applications across diverse infrastructures. [3]

[1] More info here.

The Computing Continuum

IoT/Edge — Fog — HPC/Cloud/Instrument

Size	Nano	Micro	Milli	Server	Fog	Campus	Facility
Example	Adafruit Trinket	Particle.io Boron	Array of Things	Linux Box	Co-located Blades	1000-node cluster	Datacenter
Memory	0.5K	256K	8GB	32GB	256G	32TB	16PB
Network	BLE	WiFi/LTE	WiFi/LTE	1 GigE	10GigE	40GigE	N*100GigE
Cost	$5	$30	$600	$3K	$50K	$2M	$1000M

Count = 10^9 Size = 10^1 … Count = 10^1 Size = 10^9

Figure 2: The Computing Continuum. [4]

In the following pages, we analyze the computing infrastructures mentioned in the previous example.

Data Centers

The definition of a Data Centers can be found on page 4.

✔ Data Centers Advantages

- **Lower IT costs.**
- **High Performance.**
- **Instant software updates.**
- **"Unlimited" storage capacity.**
- **Increased data reliability.**
- **Universal data access.**
- **Device Independence.**

👎 Data Centers Disadvantages

- **Require a constant internet connection.**
- **Do not work well with low-speed connections.**
- **Hardware Features might be limited.**
- **Privacy and security issues.**
- **High power Consumption.**
- **Latency in taking decision.**

Internet-of-Things (IoT)

An **Internet of Things (IoT) device is any everyday object embedded with sensors, software, and internet connectivity**.

This allows to collect and exchange data with other devices and systems, typically over the internet, with limited need of process and store data.

Some **examples** are Arduino, STM32, ESP32, Particle Argon.

✔ **Internet-of-Things Advantages**

- **Highly Pervasive**.
- **Wireless connection**.
- **Battery Powered**.
- **Low costs**.
- **Sensing and actuating**.

👎 **Internet-of-Things Disadvantages**

- **Low computing ability**.
- **Constraints on energy**.
- **Constraints on memory (RAM/FLASH)**.
- **Difficulties in programming**.

Embedded (System) PCs

An **Embedded System** is a computer system, a combination of a computer processor, computer memory, and input/output peripheral devices, that has a dedicated function within a larger mechanical or electronic system.

A few **examples**: Odroid, Raspberry, jetson nano, Google Coral.

✔ **Embedded System Advantages**

- **Persuasive computing**.
- **High performance unit**.
- **Availability of development boards**.
- **Programmed as PC**.
- **Large community**.

👎 **Embedded System Disadvantages**

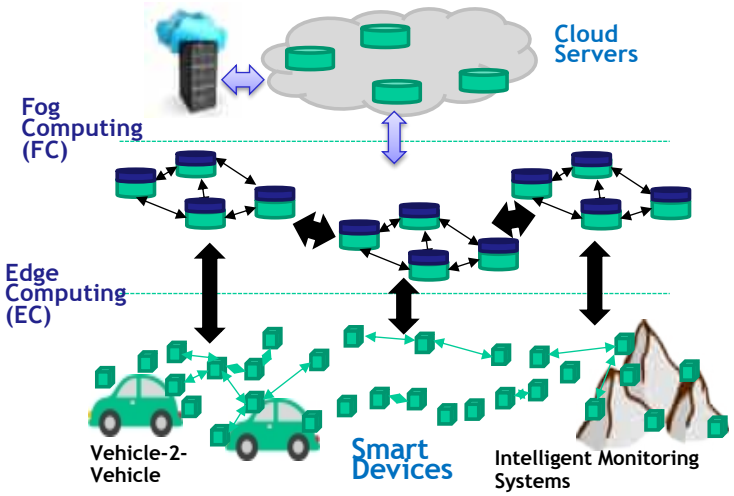
- **Pretty high power consumption**.
- **(Some) Hardware design has to be done**.

Edge/Fog Computing Systems

The key **difference** between **Fog Computing** and **Edge Computing** is associated with the location **where the data is processed**:

- In **edge computing**, the data is processed closest to the sensors.
- In **fog computing**, the computing is moved to processors linked to a local area network (IoT gateway).

Edge computing places the intelligence in the connected devices themselves, whereas, fog computing puts in the local area network.



✔ Fog/Edge Advantages

- **High computational capacity.**
- **Distributed computing.**
- **Privacy and security.**
- **Reduced Latency in making a decision.**

👎 Fog/Edge Disadvantages

- **Require a power connection.**
- **Require connection with the Cloud.**

Feature	Edge Computing	Fog Computing
Location	Directly on device or nearby device.	Intermediary devices between edge and cloud.
Processing Power	Limited due to device constraints, sending data to central server for analysis.	More powerful than edge devices. However, sending data to a central server for analysis.
Primary Function	Real-time decision-making, low latency. However, central server analyzing combined data and sending only relevant information further.	Pre-process and aggregate data, reduce bandwidth usage. However, central server analyzing combined data and sending only relevant information further.
Advantages	Low latency, reduced reliance on cloud, security for sensitive data.	Bandwidth efficiency, lower cloud costs, complex analysis capabilities.
Disadvantages	Limited processing power, single device focus.	Increased complexity, additional infrastructure cost.

Table 1: Differences between Edge and Fog Computing Systems.

2.1.1.2 The Datacenter as a Computer

In the last few decades, computing and storage have moved from PC-like clients to smaller, often mobile, devices combined with extensive internet services. Furthermore, traditional enterprises are also shifting to Cloud computing.

The **advantages** of this migration are:

- **User-side**:
  - **Ease of management** (no configuration or backups needed);
  - The **availability of the service is everywhere**, but we need connectivity.
- **Vendors-side**:
  - SaaS (Software-as-a-Service) allows **faster application development** (more accessible to make changes and improvements);
  - **Improvements and fixes** in the software are **more straightforward inside their data centers** (instead of updating many millions of clients with peculiar hardware and software configurations);
  - The **hardware deployment** is restricted to a **few well-tested configurations**.
- **Server-side**:
  - **Faster introduction of new hardware devices** (e.g., HW accelerators or new hardware platforms);
  - Many application **services can run at a low cost per user**.

Finally, another advantage is that **some workloads require so much computing capability that they are a more natural fit in the datacenter** (and not in client-side computing). For example, the search services (web, images, and so on) or the Machine and Deep Learning.

2.1.1.3 Warehouse-Scale Computers

The trends toward server-side computing and widespread internet services created a new class of computing systems: **Warehouse-Scale Computers**.

Definition 1

Warehouse-Scale Computers (WSCs) is intended to draw attention to the most distinctive feature of these machines: **the massive scale of their software infrastructure, data repositories and hardware platform**.

What is a *program* at a WSC?

In Warehouse-Scale Computing **the program is an internet service**, which may **consist of tens or more individual programs that interact to implement complex end-user services** such as *email*, *search*, or *maps*. These programs might be implemented and maintained by different teams of engineers, perhaps even across organizational, geographic, and company boundaries.

Difference between WSCs and Data Centers

WSCs currently power the services offered by companies such as Google, Amazon, Microsoft, and others. The main difference from traditional data centers (see more on page 4) is that **WSCs belong to a single organization, use a relatively homogeneous hardware and system software platform, and share a common systems management layer**. In contrast with the typical data center that belongs to multiple organizational units or even different companies, use dedicated HW infrastructure in order to run a large number of applications (more details on page 4).

How is the WSC organized?

The **software on WSCs**, such as Gmail, runs on a scale far beyond a single machine or rack: it **runs on clusters of hundreds to thousands of individual servers**. Therefore, the machine, the computer, is itself this **large cluster or aggregation of servers** and must be **considered a single computing unit**.

Most importantly, WSCs run fewer vast applications (internet services). An **advantage** is that the **shared resource management infrastructure allows significant deployment flexibility**. Finally, the requirements of:

- **Homogeneity**
- **Single-Organization Control**
- **Cost Efficiency**

Motivate designers to take new approaches to constructing and operating these systems.

2.1.1.4 Multiple Data Centers

Sometimes the data centers are **located far apart**. **Multiple data centers** are (often) **replicas of the same service**:

- To *reduce user latency*
- To *improve service throughput*

Typically, a request is fully processed within one data center.

The world is divided into **Geographic Areas (GAs)**. Each Area is defined by Geo-political boundaries (or country borders). Also, there are at least two computing regions in each geographical Area.

The **Computing Regions (CRs)** are the smallest geographic unit of the infrastructure from the customer's perspective. Multiple Data centers within the same region are not exposed to customers.

However, they are defined by a latency-defined perimeter, typically less than 2ms for round-trip latency. Finally, they're located hundreds of miles apart, with considerations for different flood zones, etc. It is too far from synchronous replication but suitable for disaster recovery.

The **Availability Zones (AZs)** are finer-grain **locations within a single computing region**. They allow customers to run mission-critical applications with high availability and fault tolerance to Data Center failures. Because there are fault-isolated locations with redundant power, cooling, and networking (they are different from the concept of the Availability Set).

This hierarchical structure ensures efficient data management and compliance with local data laws while optimizing network performance through strategically placing data centers.

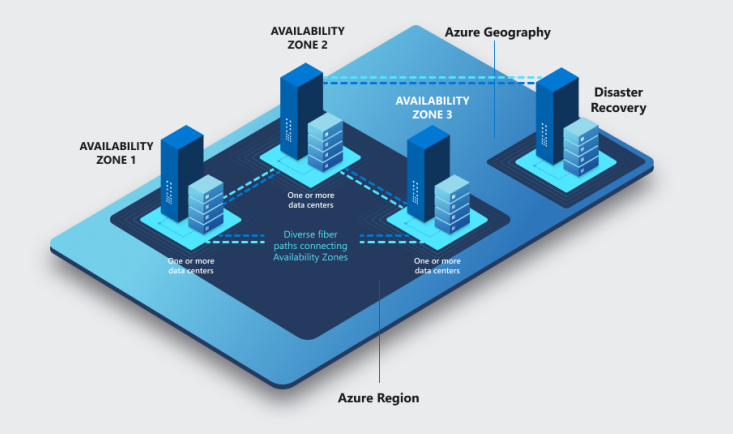


Figure 3: Example of Azure Availability Zones.

2.1.1.5 Warehouse-Scale Computing / Data Centers Availability

The services provided through WSCs (or DCs) **must guarantee high availability**, typically aiming for at least 99.99% uptime (e.g. one hour of downtime per year).

Some **examples**:

- 99,90% on single instance VMs with premium storage for a more accessible lift and shift;
- 99,95% VM uptime SLA for Availability Sets (AS) to protect for failures within a data center;
- 99,99% VM uptime SLA through Availability Zones.

Such fault-free operation is more accessible when an extensive collection of hardware and system software is involved.

WSC workloads must be designed to gracefully tolerate large numbers of component faults with little or no impact on service level performance and availability!

2.1.1.6 Architectural overview of Warehouse-Scale Computing

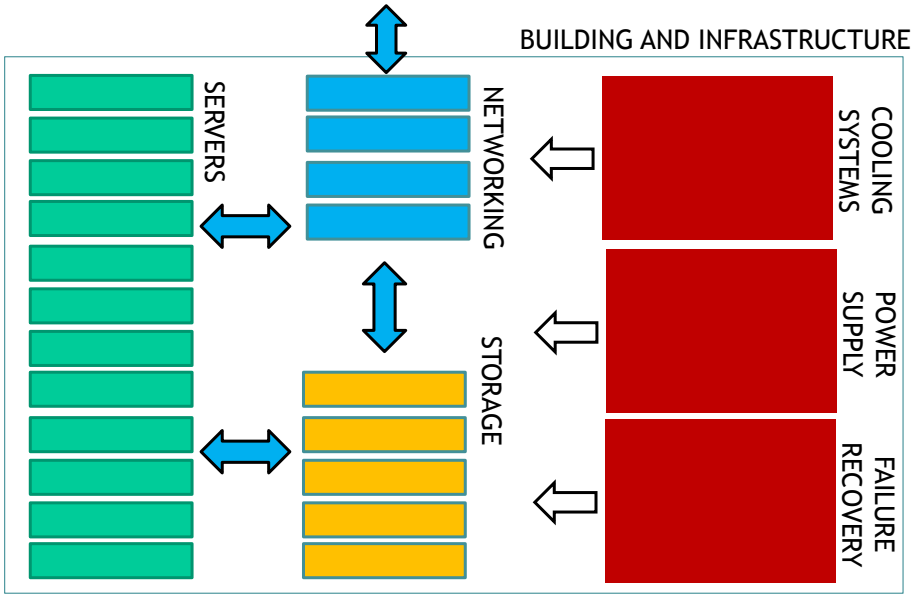


Figure 4: Architectural overview of Warehouse-Scale Computing.

- **Server** (section 2.2.1, page 15). Servers are the **leading processing equipment**: different types according to CPUs, RAM, local storage, accelerators, and form factor. The servers are **hosted on individual shelves** and are the **basic building blocks of Data Centers and Warehouse-Scale Computers**. They are interconnected by hierarchies of networks and supported by the shared power and cooling infrastructure.
- **Storage** (section 2.2.2, page 22). Disks, flash SSDs, and Tapes are the **building blocks** of today's **WSC storage systems**. These devices are **connected to the Data Center network and managed by sophisticated distributed systems**.

  Some **examples**:

  - Direct Attached Storage (DAS)
  - Network Attached Storage (NAS)
  - Storage Area Networks (SAN)
  - RAID controllers
- **Networking** (section 2.2.3, page 29). The **Data Center Network (DCN) enables efficient data transfer and interaction between various components**. The data processing ecosystem within the DCs needs to reach the DC services from outside. Communication equipment includes switches, Routers, cables, DNS or DHCP servers, Load balancers, Firewalls, etc.
- **Building and Infrastructure**. WSC has other essential components related to power delivery, cooling, and building infrastructure that must be considered. Some interesting numbers:

  - Data Centers with up to 110 football-pitch size.
  - 2-100s MW power consumption (100k houses), and the largest in the world is 650 MW.

2.2 Node-level

2.2.1 Server (computation, HW accelerators)

Servers are like ordinary PCs, usually more powerful, but with a **form factor that allows them to fit into the shelves** (such as rack, blade enclosure format, or tower; the differences are explained later). They are usually built in a tray or blade enclosure format, **housing the motherboard**, the **chipset**, and **additional plug-in components**.

The **motherboard** acts as the central hub, **connecting all the crucial components of the server and enabling them to communicate and work together**.

It provides sockets and plug-in slots to install CPUs, memory modules (DIMMs), local storage (such as Flash SSDs or HDDs), and network interface cards (NICs) to satisfy the range of resource requirements.

The **chipsets** and **additional components** are grouped in the following way:

- Number and type of CPUs:
  - From 1 to 8 CPU socket.
  - Intel Xeon Family, AMD EPYC, etc.
- Available RAM:
  - From 2 to 192 DIMM Slots.
- Locally attached disks:
  - From 1 to 24 Drive Bays.
  - HDD or SSD.
  - SAS (higher performance but more expensive) or SATA (for entry-level servers).
- Other special purpose devices:
  - From 1 to 20 GPUs per node, or TPUs.
  - NVIDIA Pascal, Volta, etc.
- Form factor:
  - From 1 unit to 10 units.
  - Tower.

Differences between Rack, Blade and Tower

Tower Server

A **Tower Server** looks and feels much like a **traditional** tower **PC**.

✔ Advantages

- ✓ **Scalability** and **ease of upgrade**. Customized and upgraded based on necessity.
- ✓ **Cost-effective**. Tower servers are probably the *cheapest of all kinds of servers.*
- ✓ **Cools easily**. Since a tower server has a low overall component density, it cools down easily.

👎 Disadvantages

- ✗ **Consumes a lot of space**. These servers are difficult to manage physically.
- ✗ **Provides a basic level of performance**. A tower server is *ideal for small business that have a limited number of clients.*
- ✗ **Complicated cable management**. Devices aren't easily routed together.

2.2.1.1 Rack Servers

Rack Servers are unique **shelves that accommodate all the IT equipment** and allow their interconnection. The racks are used to store these rack servers.

Server racks are measured in **Rack Units**, or "U". One U is approximately 44.45 millimeters. The main advantage of these racks is that they **allow designers to stack up other electronic devices and servers**.

A rack server is designed to be positioned in a bay by vertically stacking servers one over the other along with other devices (storage units, cooling systems, network peripherals, and batteries).

✔ **Advantages**

- ✓ **Failure containment**. Very little effort to identify, remove, and replace a malfunctioning server with another.
- ✓ **Simplified cable management**. Easy and efficient to organize cables.
- ✓ **Cost-effective**. Computing power and efficiency at relatively lower costs.

👎 **Disadvantages**

- ✗ **Power usage**. Needs of additional cooling systems due to their high overall component density, thus consuming more power.
- ✗ **Maintenance**. Since multiple devices are placed in racks together, maintaining them gets considerably though with the increasing number of racks.

2.2.1.2 Blade Servers

Blade Servers are the latest and the most advanced type of servers in the market. They can be termed hybrid rack servers, where servers are placed inside blade enclosures, forming a blade system.

The **most significant advantage** of blade servers is that these servers are the **most minor types of servers available now** and are **great for conserving space**.

Finally, a blade system also meets the IEEE standard for rack units, and each rack is measured in the units of "U".

✔ Advantages

- ✓ **Size and form factor**. They are smallest and the most compact servers, requiring minimal physical space. Blade servers offer *higher space efficiency* compared to traditional rack-mounted servers.
- ✓ **Cabling**. Blade server don't involve the cumbersome tasks of setting up cabling. Although you still might have to deal with the cabling, it is near to negligible when compared to tower and rack servers.
- ✓ **Centralized management**. Blade enclosures typically come with centralized management tools that allow administrators to easily monitor, configure and update all blades from a single interface.
- ✓ **Load balancing, failover, scalability**. Uniform system, shared components (including network), simple addition/removal of servers.

👎 Disadvantages

- ✗ **Expensive configuration and Higher initial cost**. Although upgrading the blade server is easy to handle and manage, the initial configuration or the setup requires more effort and higher initial investment.
- ✗ **Vendor Lock-In**. Blade servers typically require the use of the manufacturer's specific blades and enclosures, leading to vendor lock-in. This can limit flexibility and potentially increase costs in the long run.
- ✗ **Cooling**. Blade servers come with high component density. Therefore, special accommodations have to be arranged for these servers to ensure they don't get overheated. Heating, ventilation, and air conditioning systems (HVAC) must be carefully managed and designed.

2.2.1.3 Machine Learning

Deepening: Machine Learning (supervised learning)

Machine learning (ML) is a branch of artificial intelligence (AI) and computer science that focuses on the using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy (source).

UC Berkeley breaks out the learning system of a machine learning algorithm into three main parts:

1. A Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate about a pattern in the data.
2. An Error Function: An error function evaluates the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
3. A Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this iterative "evaluate and optimize" process, updating weights autonomously until a threshold of accuracy has been met.

The main goal is to learn a target function that can be used for prediction. Given a training set of labeled examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, estimate the prediction function f by minimizing the prediction error on the training set:

$$y = f(x)$$

Where y is the output, f is the prediction function and the x is an image feature.

Deepening: Artificial Neural Network

The **Artificial Neural Network** is a computational model inspired by the human brain (perceptron). It consists of interconnected nodes (neurons) organized in layers to process and analyze data and used to learn data representation from data (learn features and the classifier/regressor).

The learning process of a Neural Network is as follows: Neurons make decisions (activation functions). There are wights, so the connections between neurons are strengthened or weakened through training- randomly initialized.
The (training data) Neural Networks learn from historical data and ex-

amples. Then, labeled data are provided.

Deepening: effects of ML and ANN

Deep learning models began to appear and be widely adopted, enabling specialized hardware to power a broad spectrum of machine learning solutions.
Since 2013, AI learning compute requirements have doubled every 3.5 months (vs. 18-24 months expected from Moore's Law).
To satisfy the growing compute needs for deep learning, **WSCs deploy specialized accelerator hardware**:

- Graphical Processing Units (GPUs) are used for data-parallel computations (the same program is executed on many data elements in parallel). In order to use parallel programming, high-level languages such as CUDA, OpenCL, OPENACC, OpenMP, and SYCL exist. This technique allows up to 1000x faster than CPU.
- Tensor Processing Unit (TPU), where Tensor is a n-dimensional matrix, are used for training and inference.
- Field-Programmable Gate Array (FPGA) are programmable hardware devices. The device user can program an array of logic gates ("configured") in the field instead of the people who designed it. An array of carefully designed and interconnected digital subcircuits that efficiently implement common functions, offering very high levels of flexibility. The digital subcircuits are called configurable logic blocks (CLBs).

  FPGA Applications in Data Centers:

  - Network acceleration: FPGAs can offload specific processing tasks from CPUs, improving overall network performance and reducing CPU workload.
  - Security acceleration: Encryption, decryption, and other security-related tasks can be accelerated using FPGAs, enhancing data centre security while maintaining performance.
  - Data analytics: FPGAs can accelerate specific algorithms in data analytics workloads, leading to faster data processing and analysis.
  - Machine learning: FPGAs can be configured to efficiently implement specific machine learning algorithms, potentially offering performance advantages for specialized tasks.

	Advantages	Disadvantages
CPU	• Easy to be programmed and support any programming framework. • Fast design space exploration and run your applications.	• Suited only for simple AI models that do not take long to train and for small models with small training set.
GPU	• Ideal for applications in which data need to be processed in parallel like the pixels of images or videos.	• Programmed in languages like CUDA and OpenCL and therefore provide limited flexibility compared to CPUs.
TPU	• Very fast at performing dense vector and matrix computations and are specialized on running very fast programming based on Tensorflow.	• For applications and models based on the TensorFlow. • Lower flexibility compared to CPUs and GPUs.
FPGA	• Higher performance, lower cost and lower power consumption compared to other options like CPUs and GPU.	• Programmed using OpenCL and High-Level Synthesis (HLS). • Limited flexibility compared to other platforms.

2.2.2 Storage (type, technology)

Data has significantly grown in the last few years due to sensors, industry 4.0, AI, etc. The growth favours the **centralized storage strategy** that is focused on the following:

- **Limiting redundant data**
- **Automatizing replication and backup**
- **Reducing management costs**

The *storage technologies* are many. One of the oldest but still used is the **hard disk drive (HDD)**, a magnetic disk with mechanical interactions. Due to its mechanical movement, the **solid-state drive (SSD)** is the best solution (quality-price) because there are no mechanical or moving parts, and they are built out of transistors (NADN flash-based devices). The **non-volatile memory express (NVMe)** also exists, which is the **latest industry standard** for running PCIe[2] SSDs.

As for price classification, we can see that the NVMe is the most expensive solution:

1. NVMe (between 100€ and 200€ for 1TB)
2. SSD (between 70€ and 100€ for 1TB)
3. HDD (between 40€ and 60€ for 1TB)

For these reasons, it is reasonable to use a **hybrid solution** (HDD + SSD):

- A speed storage technology (**SSD or NVMe**) as **cache** and **several HDDs for storage**. It is a combination used by some servers: a small SSD with a large HDD to have a faster disk.
- Some HDD manufacturers produce Solid State Hybrid Disks (SSHD) that combine a small SDD with a large HDD in a single unit.

[2] **PCIe (peripheral component interconnect express)**. is an interface standard for connecting high-speed components

2.2.2.1 Files

An **OS can see the disks as a collection of data blocks** that can be read or written independently. To allow the ordering/management among them, **each block is characterized by a unique numerical address** called **LBA (Logical Block Address)**. Typically, the **OS groups blocks into clusters**[3] **to simplify the access to the disk**. Typical cluster sizes range from 1 disk sector (512 B, or 4 KB) to 128 sectors (64 KB).

Each *cluster* contains:

- **File data**. The actual content of the files.
- **Metadata**. The information required to support the file system:
  - File names
  - Directory structures and symbolic links
  - Creation, modification, and last access dates
  - Security information (owners, access list, encryption)
  - **Links to the LBA where the file content can be located on the disk**

The disk can thus contain **several types of clusters**:

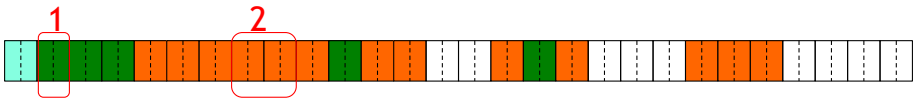
- Metadata:
  - Fixed position (to bootstrap the entire file system)
  - Variable position (to hold the folder structure)
- File data (the actual content of the files)
- Unused space (available to contain new files and folders)

Figure 5: A cluster can be seen visually as an array. In this image, for example, we've shown three types of cluster: metadata fixed position (azure), metadata variable position (green), file data (orange), unused space (white).

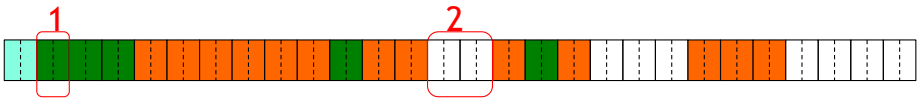
[3] **Clusters** are the minimal units an OS can read from or write to a disk.

The following explanation introduces some basic operations on the files to see what happens inside the disks.

- **Reading**. To read a file:
  1. Access the metadata, variable position (because it contains the folder structure), to locate its block;
  2. Access the blocks to read the contents of the file.



- **Writing**. To write a file:
  1. Access the metadata, variable position (because it contains the folder structure), to find free space.
  2. Write the data in the allocated blocks (cluster type: unused space).



Since the *file system can only access clusters*, the **actual space taken up by a file on a disk is always a multiple of the cluster size**. Given:

- s, the *file size*
- c, the *cluster size*

Then the **actual size on the disk** a can be calculated as:

$$a = \text{ceil}\left(\frac{s}{c}\right) \times c \tag{1}$$

Where ceil rounds a number up to the nearest integer. It's also possible to calculate the **amount of disk space wasted by organising the file into clusters** (**wasted disk space** w):

$$w = a - s \tag{2}$$

A formal way to refer to wasted disk space is **internal fragmentation** of files.

Example 1: internal fragmentation

- File size: 27 byte
- Cluster size: 8 byte

The *actual size* on the disk is:

$$a = \text{ceil}\left(\frac{27}{8}\right) \cdot 8 = \text{ceil}\,(3.375) \cdot 8 = 4 \cdot 8 = 32 \text{ byte}$$

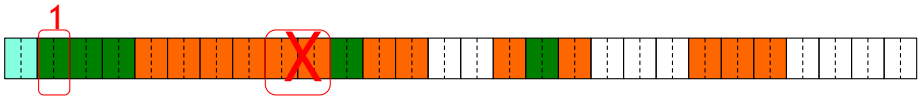
And the internal fragmentation w is:

$$w = 32 - 27 = 5 \text{ byte}$$

- **Deleting**. To delete a file:

  1. Just update the metadata, variable position (because it contains the folder structure), to say that the blocks where the file was stored are no longer used by the OS.

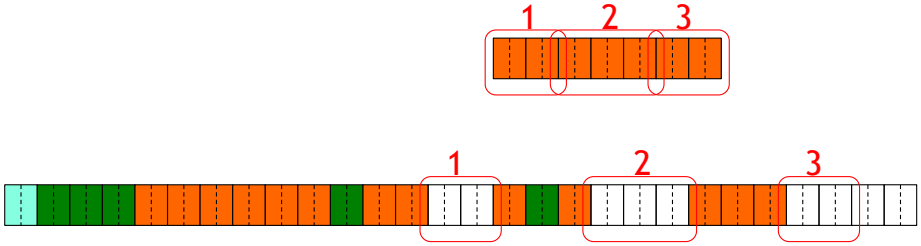
  Deleting a file never actually deletes the data on the disk: if a new file is written to the same clusters, the old data is replaced by the new.



- **External fragmentation**. As the disk's life evolves, there might **not be enough space to store a file contiguously**.

  In this case, the file is split into smaller chunks and inserted into the free clusters spread over the disk.

  The effect of **splitting a file into non-contiguous clusters** is called **external fragmentation**.



2.2.2.2 HDD

A **Hard Disk Drive (HDD)** is a **data storage device that uses rotating disks (platters) coated with magnetic material**.

Data is read randomly, meaning individual data blocks can be stored or retrieved in any order rather than sequentially.

An HDD consists of one or more rigid (*hard*) rotating disks (platters) with magnetic heads arranged on a moving actuator arm to read and write data to the surfaces.

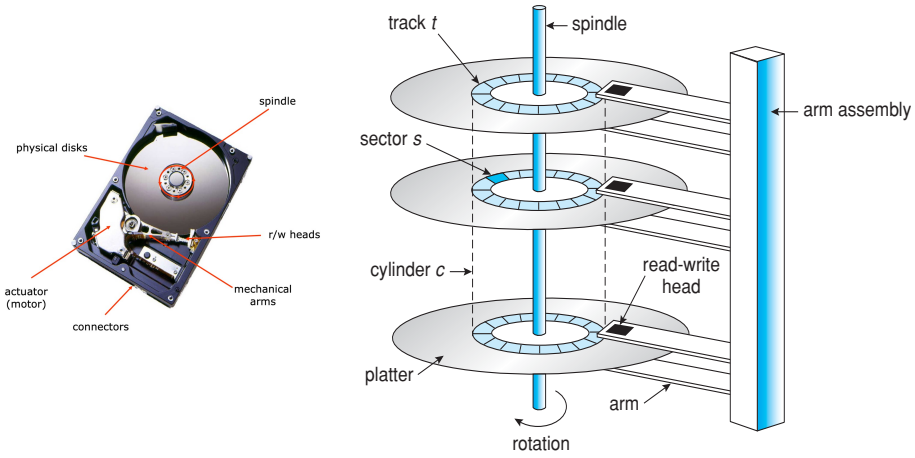


Figure 6: Hard Drive Disk anatomy.

Externally, hard drives expose a large number of **sectors** (blocks):

- Typically, 512 or 4096 bytes.
- Individual **sector writes** are **atomic**.
- Multiple sectors write it may be interrupted (**torn write**[4]).

The geometry of the drive:

- The sectors are arranged into **tracks**.
- A **cylinder** is a particular track on multiple platters.
- Tracks are arranged in concentric circles on **platters**.
- A disk may have multiple double-sided platters.

The **driver motor spins the platters at a constant rate**, measured in **Revolutions Per Minute (RPM)**.

[4]Torn writes happen when only part of a multi-sector update is written successfully to disk.

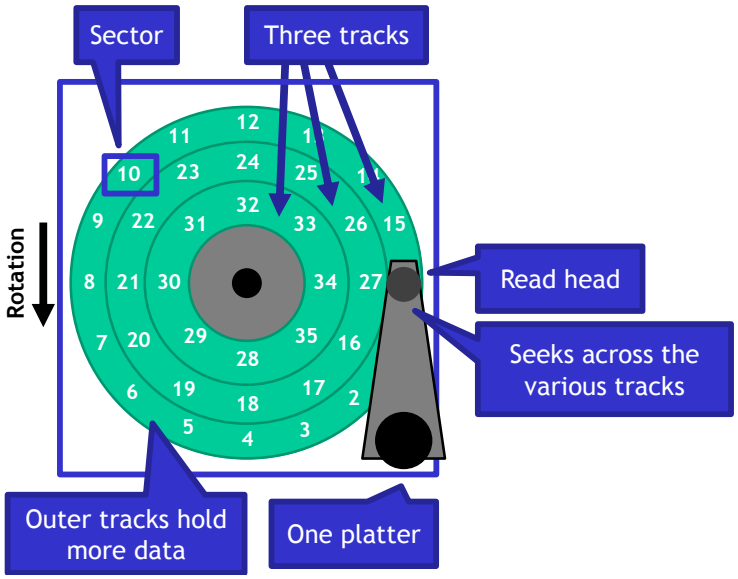


Figure 7: Example of HDD geometry.

Given the architecture of the HDD, there are **four types of delay**:

- **Rotational Delay** is the **time to rotate the desired sector to the read head**, and it's related to RPM.
- **Seek Delay** is the **time to move the read head to a different track**.
- **Transfer time** is the **time to read or write bytes**.
- **Controller Overhead** is the **overhead for the request management**.

In order to reduce the delay in the HDD, the companies use a high-speed and tiny memory (8, 16 or 32 MB) called **cache** (also called **track buffer**). The cache is used when there is a:

- **Read caching**. It reduces read delays due to seeking and rotation.
- **Write caching**. It is divided into two different implementations:
  - **Write Back cache**: the drive reports that writes are complete after they have been cached. The disadvantage is that it has an inconsistent state if the power goes off before the write-back event.
  - **Write Through cache**: the drive reports that writes are complete after they have been written to disk.

So, the **caching helps improve disk performance**. The critical idea under caching is that if there is a **queue of requests to the disk, they can be reordered to improve performance**. The estimation of the request length is feasible, knowing the position of the data on the disk.

There are several **scheduling algorithms**:

- **First Come, First Serve (FCFC)**. It is the most basic scheduler, serving requests in order. The ***disadvantage*** is that there is much time spent seeking.
- **Shortest Seek Time First (SSTF)**. It is primary purpose is to minimize seek time by always selecting the block with the shortest seek time. The ***advantage*** is that it is optimal and can be easily implemented. The main ***disadvantage*** is that it is prone to starvation.
- **SCAN**, otherwise known as the Elevator Algorithm. The head sweeps across the disk, servicing requests in order. The ***advantage*** is that it performs reasonably well and does not suffer starvation. The ***disadvantage*** is that the average access times are higher for requests at high and low addresses.
- **C-SCAN (Circular SCAN)**. It is like the SCAN algorithm, but only service requests in one direction. The ***advantage*** is fairer than SCAN. However, the ***disadvantage*** is that it has worse performance than SCAN.
- **C-LOOK**. It is a C-SCAN variant that peeks at the upcoming addresses in the queue. The head only goes as far as the last request.

2.2.2.3 SSD

2.2.2.4 RAID

2.2.3 Networking (architecture and technology)

3 Software Infrastructure

3.1 Virtualization

3.2 Computing Architectures

4 Methods

4.1 Reliability and availability of datacenters

4.2 Disk performance

4.2.1 HDD

We can calculate some performance metrics related to the types of delay of HDD (page 27).

- **Full Rotation Delay** R is:

$$R = \frac{1}{\texttt{DiskRPM}} \tag{3}$$

And in seconds:

$$R_{\text{sec}} = 60 \times R \tag{4}$$

From the R_{sec} we can also calculate the **total rotation average**:

$$T_{\text{rotation AVG}} = \frac{R_{\text{sec}}}{2} \tag{5}$$

- **Seek Time**, the **time to move the head to a different track**, which is divided into several phases:
  - Acceleration
  - Coasting (constant speed)
  - Deceleration
  - Settling

The T_{seek} modelling considers a linear dependency with the distance. Also, the **seek average** is:

$$T_{\text{seek AVG}} = \frac{T_{\text{seek MAX}}}{3} \tag{6}$$

- **Transfer time**. It is the **time that data is either read from or written to the surface**. It **includes** the time the **head needs to pass on the sectors** and the **I/O transfer**:

$$T_{\text{transfer}} = \frac{\text{R/W of a sector}}{\text{Data transfer rate}} \tag{7}$$

The **Controller Overhead** is the **buffer management** (data transfer) and **interrupt sending time**.

Transfer time and Controller Overhead are together because they are required to calculate some interesting metrics.

  - **Service Time** $T_{\text{I/O}}$

$$T_{\text{I/O}} = T_{\text{seek}} + T_{\text{rotation}} + T_{\text{transfer}} + T_{\text{overhead}} \tag{8}$$

  - **Response Time**

$$T_{\text{queue}} + T_{\text{I/O}} \tag{9}$$

Where T_{queue} depends on queue-length, resource utilization, mean and variance of disk service time and request arrival distribution.

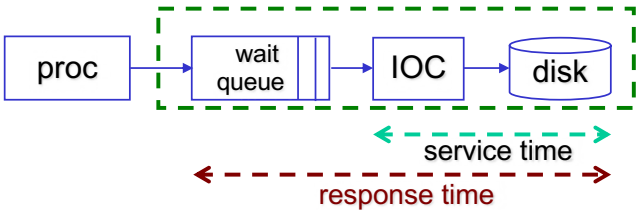


Figure 8: Service and response time.

Exercise 1: mean service time of an I/O operation

The data of the exercise are:

- Read/Write of a sector of 512 bytes (0.5 KB)
- Data transfer rate: 50 MB/sec
- Rotation speed: 10000 RPM (Round Per Minute)
- Mean seek time: 6 ms
- Overhead Controller: 0.2 ms

To calculate the *service time* $T_{\mathrm{I/O}}$, we need the following information:

- T_{seek}, which we already have, and it is 6 ms.
- T_{rotation}
- T_{transfer}
- T_{overhead}, which we already have, and it is 0.2 ms.

We also know the rotation and transfer information, but we want to know the *mean* service time. Then we calculate the total rotation average $T_{\mathrm{rotation\ AVG}}$:

$$\begin{aligned} R &= \frac{1}{\texttt{DiskRPM}} = \frac{1}{10000} = 0.0001 \\ R_{\mathrm{sec}} &= 60 \cdot R = 60 \cdot 0.0001 = 0.006 \text{ seconds} \\ T_{\mathrm{rotation\ AVG}} &= \frac{R_{\mathrm{sec}}}{2} = \frac{0.006}{2} = 0.003 \text{ seconds} = 3 \text{ ms} \end{aligned}$$

Finally, the transfer time is easy to calculate because we have the R/W of a sector and the data transfer rate. First we do a conversions from

megabytes to kilobytes:

$$\begin{aligned} \text{Data transfer rate:} \quad & 50 \text{ MB/sec} \\ &= 50 \cdot 1024 \text{ KB/sec} \\ &= 51200 \text{ KB/sec} \\ T_{\text{transfer}} &= \frac{0.5 \text{ KB/sec}}{51200 \text{ KB/sec}} \\ &= 0.000009765625 \text{ sec} \cdot 1000 \\ &= 0.009765625 \text{ ms} \approx 0.01 \text{ ms} \end{aligned}$$

The exercise can be completed by calculating the mean I/O service time required:

$$\begin{aligned} T_{\text{I/O}} &= T_{\text{seek}} + T_{\text{rotation}} + T_{\text{transfer}} + T_{\text{overhead}} \\ T_{\text{I/O}} &= 6 + 3 + 0.01 + 0.2 = 9.21 \text{ ms} \end{aligned}$$

The previous service time is supposed to be only for very **pessimistic cases** where **sectors are fragmented on the disk in the worst possible way**. This can happen because the files are tiny (each file is contained in one block) or the disk is externally fragmented.

Thus, each access to a sector requires to pay rotational latency and seek time. This is not the case in many circumstances because the files are larger than one block and stored contiguously.

We can measure the **Data Locality** DL of a disk as the **percentage of blocks that do not need seek or rotational latency to be found**.

Thanks to the Data Locality, it is possible to calculate the **Average Service Time**:

$$T_{\text{I/O AVG}} = (1 - DL) \cdot (T_{\text{seek}} + T_{\text{rotation}}) + T_{\text{transfer}} + T_{\text{controller}} \tag{10}$$

Exercise 2: data locality

The data of the exercise are:

- Read/Write of a sector of 512 bytes (0.5 KB)
- Data Locality: $DL = 75\%$
- Data transfer rate: 50 MB/sec
- Rotation speed: 10000 RPM (Round Per Minute)
- Mean seek time: 6 ms
- Overhead Controller: 0.2 ms

Since the Data Locality is 75%, only 25% of the operations are affected by the DL:

$$(1 - DL) = (1 - 0.75) = 0.25$$

See the exercise on page 34 to understand the values of T_{seek}, T_{rotation}, T_{transfer} and T_{overhead}:

- $T_{\text{seek}} = 6$
- $T_{\text{rotation}} = 3$
- $T_{\text{transfer}} = 0.01$
- $T_{\text{overhead}} = 0.2$

Finally the average time for read/write a sector of 0.5 KB with a DL of 75% is:

$$\begin{aligned} T_{\text{I/O AVG}} &= 0.25 \cdot (6 + 3) + 0.01 + 0.2 \\ &= 0.25 \cdot 9 + 0.21 \\ &= 2.46 \text{ ms} \end{aligned}$$

Exercise 3: influence of "not optimal" data allocation

The data of the exercise are:

- 10 blocks of 1/10 MB for each block (10 blocks of 1/10 MB "not well" distributed on disk)
- $T_{\text{seek}} = 6$ ms
- $T_{\text{rotation AVG}} = 3$ ms
- Data transfer rate: 50 MB/sec

In the exercise you were asked to calculate the time taken to transfer a 1 MB file with 100% and 0% data locality:

- Data Locality equals to 100%:
  - An initial seek (6 ms)
  - A total rotation average (3 ms)
  - Now it's possible to do the 1MB global transfer directly because there are no blocks to seek or rotation latency:

$$1 \text{ MB of } 50 \text{ MB} = \frac{1}{50} = 0.02 \text{ seconds } \cdot 1000 = 20 \text{ ms}$$

  - The total time is:

$$T = 6 + 3 + 20 = 29 \text{ ms}$$

- Data Locality equals to 0%:
  - An initial seek (6 ms)
  - A total rotation average (3 ms)
  - In this case, it's not possible to do a global transfer directly, because each block is affected by the seek or rotation latency. Then we have to transfer block by block and calculate the delay:

$$1 \text{ MB of } 10 \text{ MB} = \frac{1}{10} = 0.1 \text{ seconds } \cdot 1000 = 100 \text{ ms}$$

  - The total time is:

$$T = (6 + 3 + 2) \cdot 10 = 110 \text{ ms}$$

    Where 10 is the number of blocks.

Note: the controller times is not considered.

4.3 Scalability and performance of datanceters

References

[1] L.A. Barroso, U. Hölzle, and P. Ranganathan. *The Datacenter as a Computer: Designing Warehouse-Scale Machines, Third Edition*. Synthesis Lectures on Computer Architecture. Springer International Publishing, 2022.

[2] E.D. Lazowska. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, 1984.

[3] Jacopo Marino and Fulvio Risso. Is the computing continuum already here?, 2023.

[4] Gianluca Palermo. Lesson 1, computing infrastructures. Slides from the HPC-E master's degree course on Politecnico di Milano, 2024.

Index

A

actual size on the disk a 24
Availability Zones (AZs) 12
Average Service Time 35

B

Blade Servers 18

C

C-LOOK 28
C-SCAN (Circular SCAN) 28
cache 27
Clusters 23
Computing Continuum 5
Computing Infrastructure 4
Computing Regions (CRs) 12
Controller Overhead 27, 33

D

data blocks 23
Data Center 4
Data Locality DL 35

E

Edge Computing 8
Embedded System 7
external fragmentation 25

F

First Come, First Serve (FCFC) 28
Fog Computing 8
Full Rotation Delay 33

G

Geographic Areas (GAs) 12

H

Hard Disk Drive (HDD) 26
hard disk drive (HDD) 22

I

internal fragmentation 24
Internet of Things (IoT) 7

L

LBA (Logical Block Address) 23

N

non-volatile memory express (NVMe) 22

P
PCIe (peripheral component interconnect express) 22

R
Rack Servers 17
Rack Units 17
Read caching 27
Response Time 33
Revolutions Per Minute (RPM) 26
Rotational Delay 27

S
SCAN 28
seek average 33
Seek Delay 27
Seek Time 33
Server 15
Service Time 33
Shortest Seek Time First (SSTF) 28
solid-state drive (SSD) 22

T
torn write 26
total rotation average 33
Tower Server 16
track buffer 27
Transfer time 27, 33

W
Warehouse-Scale Computers (WSCs) 11
wasted disk space w 24
Write Back cache 27
Write caching 27
Write Through cache 27