Computing Infrastructures
Theory

Christian Rossi

Academic Year 2023-2024

Abstract

The course topics are:

- Hardware infrastructure of datacenters:
  - Basic components, rack structure, cooling.
  - Hard Disk Drive and Solid State Disks.
  - RAID architectures.
  - Hardware accelerators.
- Software infrastructure of datacenters:
  - Virtualization: basic concepts, technologies, hypervisors and containers.
  - Computing Architecture: Cloud, Edge and Fog Computing.
  - Infrastructure, platform and software-as-a-service.
- Methods:
  - Scalability and performance of datacenters: definitions, fundamental laws, queuing network theory basics.
  - Reliability and availability of datacenters: definitions, fundamental laws, reliability block diagrams.

Contents

CHAPTER 1

Introduction

1.1 Computing infrastructures

Definition (*Computing infrastructure*)**.** Computing infrastructure refers to the technological framework comprising hardware and software components designed to facilitate computation for other systems and services.

Data centers encompass servers tailored for diverse functions:

- *Processing Servers.*
- *Storage Servers.*
- *Communication Servers.*

1.1.1 Virtual machines

Virtual machines offer a comprehensive stack comprising an operating system, libraries, and applications. Applications rely on a guest operating system.

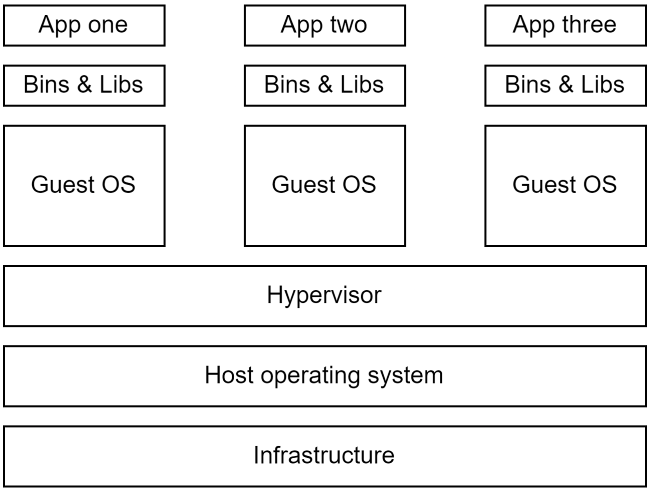


Figure 1.1: Virtual machine's structure

In this configuration, each operating system perceives the hardware as exclusively dedicated to itself. Virtualization results in significant power efficiency gains by consolidating the power consumption of individual machines, typically saving around 60%. Virtualization enables hot swaps, delivering two key advantages:

1. *Maintainability.*

2. *Availability*: overloaded machines can be supplemented by others.

1.1.2 Containers

Containers encapsulate applications along with their dependencies into a uniform unit for streamlined software development and deployment.

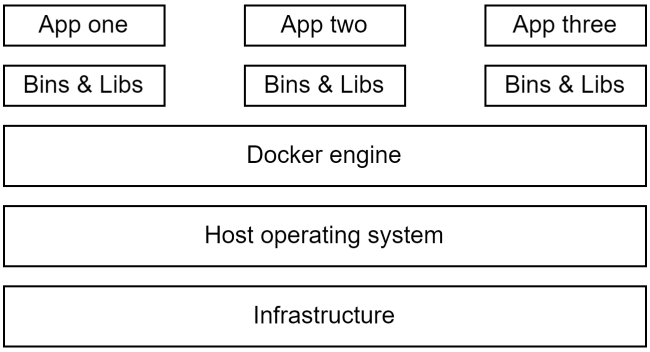


Figure 1.2: Container's structure

Containers are employed to execute specific services and offer a lighter alternative to virtual machines.

1.1.3 Summary

Data centers offer several advantages, including reduced IT costs, enhanced performance, automatic software updates, seemingly limitless storage capacity, improved data reliability, universal document accessibility, and freedom from device constraints. However, it also presents challenges such as the need for a stable internet connection, poor compatibility with slow connections, limited hardware capabilities, privacy and security concerns, increased power consumption, and delays in decision-making.

1.2 Edge computing systems

Edge computing is a distributed computing model in which data processing occurs as close as possible to where the data is generated, improving response times and saving on bandwidth. Processing data near the location where it is generated brings significant advantages in terms of processing latency, reduced data traffic, and increased resilience in case of data connection interruptions. Edge computing systems can be categorized as follows:

- *Cloud*: providing virtualized computing, storage, and network resources with highly elastic capacity.

- *Edge servers*: utilizing on-premises hardware resources for more computationally intensive data processing.
- *IoT and AI-enabled edge sensors*: enabling data acquisition and partial processing at the edge of the network.

Edge computing offers several advantages, including high computational capacity, distributed computing capabilities, enhanced privacy and security, and reduced latency in decision-making. However, it comes with drawbacks such as the requirement for a power connection and dependence on a connection with the Cloud.

1.2.1 Embedded PC

An embedded system refers to a computer system that comprises a computer processor, computer memory, and input/output peripheral devices, all serving a specific function within a larger mechanical or electronic system. Advantages of this approach include its pervasiveness in computing, high-performance units, availability of development boards, ease of programming similar to personal computers, and the support of a large community. On the other hand, it has disadvantages such as relatively high power consumption and the necessity for some hardware design work to be done.

1.2.2 Internet of things

The internet of things (IoT) encompasses devices equipped with sensors, processing capabilities, software, and other technologies. These devices are designed to connect and exchange data with other devices and systems over the internet or other communication networks. Advantages of IoT devices include their high pervasiveness, wireless connectivity, battery-powered operation, low costs, and their ability to sense and actuate. However, these devices also come with several disadvantages, such as their low computing ability, constraints on energy usage, limitations in memory (RAM/FLASH), and difficulties in programming them.

CHAPTER 2

Data centers

2.1 Introduction

Over the past few decades, there has been a significant shift in computing and storage, transitioning from PC-like clients to smaller, often mobile devices, coupled with expansive internet services. Concurrently, traditional enterprises are increasingly embracing cloud computing.

This shift offers several user experience improvements, including ease of management and ubiquitous access.

For vendors, Software-as-a-Service (SaaS) facilitates faster application development, making changes and improvements easier. Moreover, software fixes and enhancements are streamlined within data centers, rather than needing updates across millions of clients with diverse hardware and software configurations. Hardware deployment is simplified to a few well-tested configurations.

Server-side computing enables the swift introduction of new hardware devices, such as hardware accelerators or platforms, and supports many application services running at a low cost per user. Certain workloads demand substantial computing capability, making data centers a more natural fit compared to client-side computing.

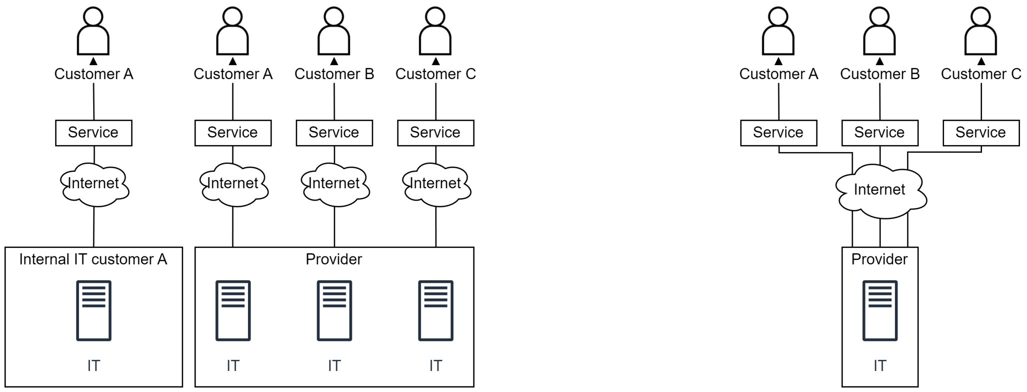
2.1.1 Warehouse-scale computers

The rise of server-side computing and the widespread adoption of internet services have given rise to a new class of computing systems known as warehouse-scale computers (WSCs). In warehouse-scale computing, the program:

- Operates as an internet service.
- Can comprise tens or more individual programs.
- These programs interact to deliver complex end-user services like email, search, maps, or machine learning.

Data centers are facilities where numerous servers and communication units are housed together due to their shared environmental needs, physical security requirements, and for the sake of streamlined maintenance. Traditional data centers typically accommodate a considerable number of relatively small- or medium-sized applications. Each application operates on a dedicated hardware infrastructure, isolated and safe guarded against other systems within the same facility. These applications typically do not communicate with one another. Moreover, these data centers host hardware and software for multiple organizational units or even different companies.

In contrast, warehouse-scale computers are owned by a single organization, employ a relatively uniform hardware and system software platform, and share a unified systems' management layer.



(a) Data center

(b) Data warehouse

Figure 2.1: Structures of data centers and data warehouses

Warehouse-scale computers operate a reduced quantity of highly expansive applications, often internet services. Their shared resource management infrastructure affords considerable deployment flexibility. Designers are driven by the imperatives of homogeneity, single-organization control, and cost efficiency, prompting them to adopt innovative approaches in crafting WSCs.

Originally conceived for online data-intensive web workloads, warehouse-scale computers have expanded their capabilities to drive public cloud computing systems, such as those operated by Amazon, Google, and Microsoft. These public clouds do accommodate numerous small applications, resembling a traditional data center setup. However, all these applications leverage virtual machines or containers and access shared, large-scale services for functionalities like block or database storage and load balancing, aligning seamlessly with the WSC model.

The software operating on these systems is designed to run on clusters comprising hundreds to thousands of individual servers, far surpassing the scale of a single machine or a single rack. The machine itself constitutes this extensive cluster or aggregation of servers, necessitating its consideration as a single computing unit.

2.1.2 Geographical distribution of data centers

Frequently, multiple data centers serve as replicas of the same service, aiming to reduce user latency and enhance serving throughput. Requests are typically processed entirely within one data center.

Definition (*Geographic area*)**.** Geographic areas partition the world into sectors, each defined by geopolitical boundaries.

Within each geographic area, there are at least two computing regions. Customers perceive regions as a more detailed breakdown of the infrastructure. Notably, multiple data centers within the same region are not externally visible. The perimeter of each computing region is defined by latency (with a round trip latency of two milliseconds), which is too far for synchronous replication but sufficient for disaster recovery.

Definition (*Availability zone*)**.** Availability zones represent more granular locations within a single computing region.

They enable customers to operate mission-critical applications with high availability and fault tolerance to datacenter failures by providing fault-isolated locations with redundant power, cooling, and networking. Application-level synchronous replication among availability zones is implemented, with a minimum of three zones being adequate for ensuring quorum.

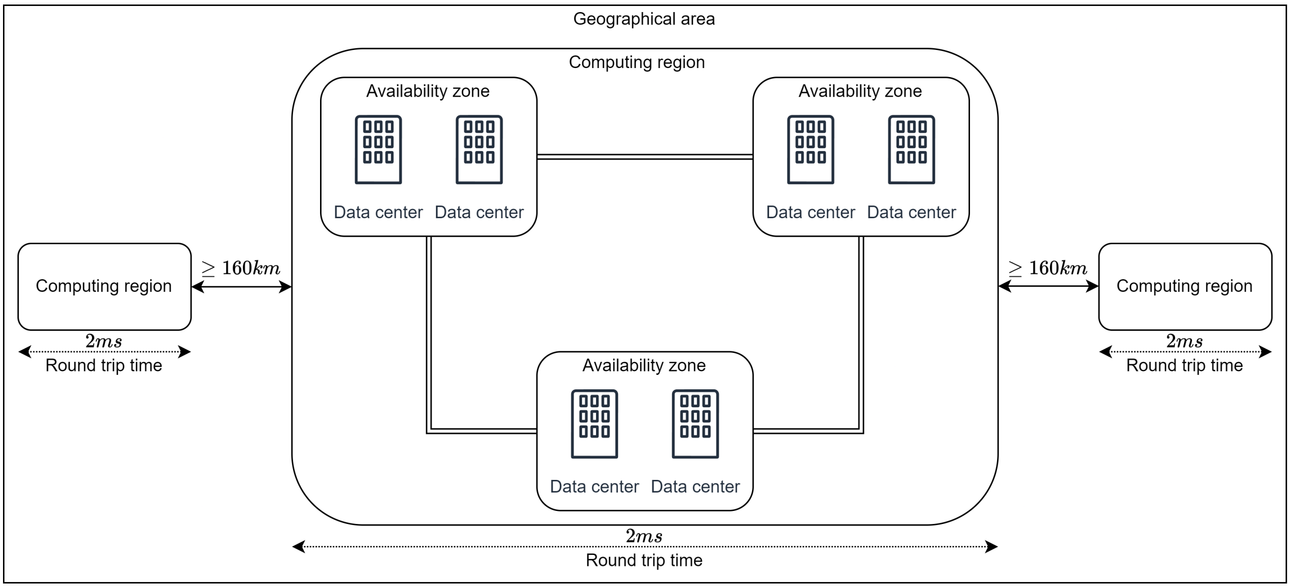


Figure 2.2: Geographical area structure

2.2 Warehouse-scale computers

Services offered by warehouse-scale computers need to ensure high availability, usually targeting a minimum uptime of 99.99%, equating to one hour of downtime per year. Maintaining flawless operation is challenging when managing a vast array of hardware and system software components. Therefore, WSC workloads must be crafted to gracefully handle numerous component faults, minimizing or eliminating any adverse effects on service performance and availability.

2.2.1 Architecture

While the hardware implementation of WSCs may vary considerably, the architectural organization of these systems remains relatively consistent.

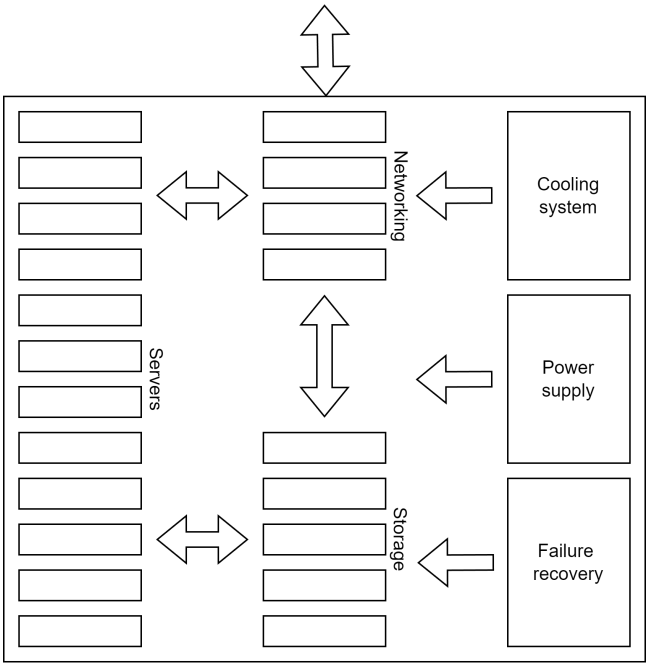


Figure 2.3: Architecture of warehouse-scale computers

Servers Servers resemble standard PCs but are designed with form factors that enable them to fit into racks, such as rack (one or more), blade enclosure format, or tower configurations. They can vary in terms of the number and type of CPUs, available RAM, locally attached disks (HDD, SSD, or none), as well as the inclusion of other specialized devices like GPUs, DSPs, and coprocessors.

Storage Disks and Flash SSDs serve as the fundamental components of contemporary WSC storage systems. These devices are integrated into the data-center network and overseen by advanced distributed systems. Examples of storage configurations include Direct Attached Storage (DAS), Network Attached Storage (NAS), Storage Area Networks (SAN), and RAID controllers.

Networking Communication equipment facilitates network interconnections among devices within the system. These include hubs, routers, DNS or DHCP servers, load balancers, switches, firewalls, and various other types of devices.

2.3 Servers

Servers housed within individual shelves form the foundational components of warehouse-scale computers. These servers are interconnected through hierarchical networks and are sustained by a shared power and cooling infrastructure.

They resemble standard PCs but are designed with form factors that enable them to fit into shelves, such as rack (one or more), blade enclosure format, or tower configurations. Servers are typically constructed in a tray or blade enclosure format, housing the motherboard, chipset, and additional plug-in components.

Motherboard The motherboard offers sockets and plug-in slots for installing CPUs, memory modules (DIMMs), local storage (such as Flash SSDs or HDDs), and network interface cards (NICs) to meet various resource requirements.

The chipset and additional components of these systems include:

- *Number and type of CPUs*: these systems support a varying number of CPU sockets, typically ranging from 1 to 8, and can accommodate processors such as Intel Xeon Family, AMD EPYC, etc.
- *Available RAM*: they offer a range of DIMM slots, typically from 2 to 192, for memory modules.
- *Locally attached disks*: these systems come with between 1 and 24 drive bays for local storage. They support both HDD and SSD options, with specific configurations detailed in lectures. Users can choose between SAS for higher performance (albeit at a higher cost) or SATA for entry-level servers.
- *Other special purpose devices*: these systems can integrate specialized components like GPUs or TPUs, with support for various models including NVIDIA Pascal, Volta, A100, etc.
- *Form factor*: they come in different sizes, ranging from 1U to 10U, and can also be configured as tower systems.

2.3.1 Racks

Racks serve as specialized shelves designed to house and interconnect all IT equipment. They are utilized for storing rack servers and are measured in rack units, denoted as U, with one rack unit (1U) equivalent to 44.45 mm (1.75 inches). One of the advantages of using racks is that they allow designers to stack additional electronic devices alongside the servers. IT equipment must adhere to specific sizes to fit into the rack shelves.

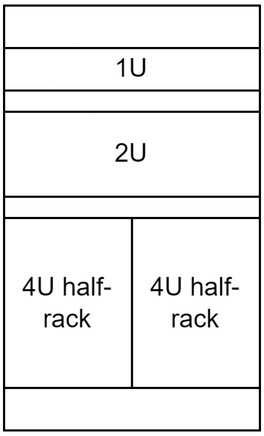


Figure 2.4: Rack elements dimensions

The rack serves as the shelf that securely holds together tens of servers. It manages the shared power infrastructure, including power delivery, battery backup, and power conversion. The width and depth of racks vary across WSCs, with some adhering to classic 19-inch width and 48-inch depth, while others may be wider or shallower. It is often convenient to connect network cables at the top of the rack, leading to the adoption of rack-level switches appropriately called Top of Rack (TOR) switches.

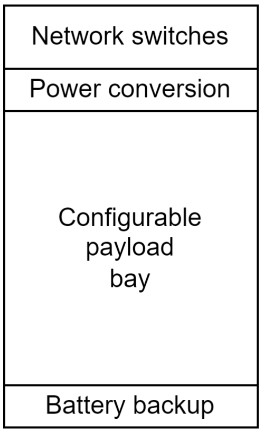


Figure 2.5: Rack structure

Advantages of using racks include the ease of failure containment, as identifying and replacing malfunctioning servers is a straightforward process. Additionally, racks offer simplified cable management, making it efficient to organize cables. Moreover, they provide cost-effectiveness by offering computing power and efficiency at relatively lower costs.

However, there are also drawbacks to consider. The high overall component density within racks leads to increased power usage, requiring additional cooling systems. Consequently, this results in higher power consumption. Furthermore, maintaining multiple devices within racks becomes considerably challenging as the number of racks increases, making maintenance a complex task.

2.3.2 Towers

A tower server closely resembles a traditional tower PC in appearance and functionality. Its advantages include scalability and ease of upgrade, allowing for customization and upgrades as needed. Tower servers are also cost-effective, often being the cheapest option among server types. Additionally, due to their low overall component density, they cool down easily.

However, tower servers have their drawbacks. They consume a significant amount of physical space and can be difficult to manage. Furthermore, while they provide a basic level of performance suitable for small businesses with a limited number of clients, they may not meet the performance needs of larger enterprises. Additionally, complicated cable management is a challenge, as devices are not easily routed together within the tower server setup.

2.3.3 Blade servers

Blade servers represent the latest and most advanced type of servers available in the market today. They can be described as hybrid rack servers, with servers housed inside blade enclosures forming a blade system. The primary advantage of blade servers lies in their compact size, making them ideal for conserving space in data centers.

Advantages of blade servers include their small size and form-factor, requiring minimal physical space. They also simplify cabling tasks compared to tower and rack servers, as the cabling involved is significantly reduced. Additionally, blade servers offer centralized management, allowing all blades to be connected through a single interface, which facilitates easier maintenance and monitoring. Furthermore, blade servers support load balancing, failover, and scalability through a uniform system with shared components, enabling simple addition or removal of servers.

However, blade servers do have drawbacks. Their initial configuration or setup can be expensive and may require significant effort, particularly in complex environments. Blade servers often entail vendor lock-in, as they typically require the use of manufacturer-specific blades and enclosures, limiting flexibility and potentially increasing long-term costs. Moreover, due to their high component density and powerful nature, blade servers require special accommodations to prevent overheating, necessitating well-managed heating, ventilation, and air conditioning systems in data centers hosting blade servers.

2.3.4 Building structure

The IT equipment is housed in corridors and arranged within racks. It's important to note that server racks are never positioned back-to-back. The corridors where servers are situated are divided into cold aisles, providing access to the front panels of the equipment, and warm aisles, where the back connections are located. Cold air flows from the front (cool aisle), cooling down the equipment, and exits the room through the back (warm aisle).

2.4 Hardware accelerators

The rate of complexity in technology doubles approximately every 3.5 months, contrasting with Moore's Law, which historically predicted a doubling of computing capacity every 18-24 months. However, in the present era, Moore's Law has slowed down, with computing capacity doubling every 4 years or longer.

The emergence and widespread adoption of deep learning models have ushered in a new era where specialized hardware plays a crucial role in powering a wide range of machine learning solutions. Since 2013, the computational requirements for AI training have been doubling every 3.5 months, significantly outpacing the traditional Moore's Law projection of 18-24 months. To meet the escalating computational demands of deep learning tasks, WSCs are deploying specialized accelerator hardware such as GPUs, TPUs, and FPGAs. These accelerators are optimized to handle the intensive processing required for training and inference tasks in deep learning applications.

2.4.1 Graphical processing unit

GPUs have revolutionized data processing by enabling data-parallel computations, where the same program can be executed simultaneously on numerous data elements in parallel. This parallelization technique is particularly effective for scientific codes, which are often structured around matrix operations.

Harnessing the power of GPUs typically involves using high-level languages like CUDA and OpenCL. Compared to traditional CPU-based processing, GPUs can deliver remarkable speed boosts, sometimes performing computations up to 1000 times faster.

2.4.2 Neural networks

Neural networks are a computational model inspired by the human brain, specifically the perceptron. They comprise interconnected nodes, or neurons, organized in layers to process and analyze data. Neural networks are utilized to learn data representation, enabling them to learn features and function as classifiers or regressors.

Neural networks have a rich history dating back to the 1940s, with notable developments occurring in the 1980s. In recent years, there has been a resurgence of interest in neural networks due to factors such as increased data availability and computational power, with some regarding them as among the top breakthroughs in 2013.

In natural neurons, information is transmitted through chemical mechanisms. Dendrites gather charges from synapses, which can be either inhibitory or excitatory. Once a threshold is reached, the accumulated charge is released, causing the neuron to fire. Like its biological counterpart, an artificial neuron receives input signals, which are weighted and summed. This sum undergoes an activation function, determining the output signal of the neuron:

$$h_j(x|w,b) = h_j\left(\sum_{i=1}^{I} w_j x_i - b\right) = h_j\left(\sum_{i=0}^{I} w_i x_i\right) = h_j\left(w^T x\right)$$

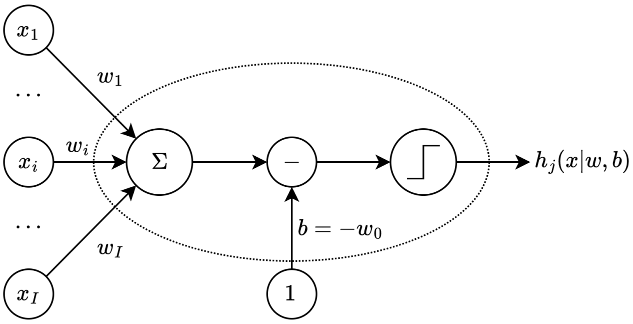


Figure 2.6: Artificial neuron

Neural networks are structured into at least three layers:

- *Input layer*: this is where data is initially introduced into the network.
- *Hidden layers*: intermediate layers that process and transform the input data.
- *Output layers*: the final layer that provides the network's ultimate results or predictions.

These layers are interconnected through weighted connections. Activation functions, which must be differentiable, determine the output of each neuron. Importantly, the output of a neuron is influenced solely by the inputs from the preceding layer.

Neural networks are inherently non-linear models. Their behavior is characterized by various factors, including the number of neurons, the choice of activation functions, and the specific values of the weights assigned to connections within the network. These factors collectively determine the network's ability to learn and make accurate predictions from the input data.

Learning in neural networks involves several key processes:

- *Activation functions*: neurons within the network make decisions based on input data through activation functions.
- *Weights*: connections between neurons are adjusted during training, either strengthened or weakened, to optimize performance.

Neural networks learn from historical data and examples provided during training. This process often involves backpropagation, which utilizes techniques like gradient descent and the chain rule to calculate errors and adjust the model accordingly. Errors are evaluated and used to refine the network's parameters, improving its ability to make accurate predictions or classifications.

Neural networks come in various types, each tailored for specific tasks:

- *Feed forward neural network*: standard neural network where information flows in one direction, from input to output.
- *Convolutional neural networks* (CNNs): designed for processing grid-like data such as images, with specialized layers for feature extraction and pattern recognition.
- *Recurrent neural networks* (RNNs): suitable for sequential data where past information influences the present, often used in natural language processing and time series analysis.

These types of neural networks find applications across diverse domains:

- *Image recognition*: utilized in technologies like FaceNet and YOLO for tasks such as facial recognition, object detection, and instance segmentation.
- *Natural language processing*: leveraged by models like BERT and GPT for applications like chatbots, sentiment analysis, and speech-to-text translation.

The potential for innovation and future development in neural networks is extensive. With the rapid growth of neural network applications, they are continuously expanding into new and diverse areas such as social media, aerospace, e-commerce, finances, and beyond. Additionally, advancements in generative AI are driving innovation in fields like art, music, and content generation, allowing for the creation of novel content through sophisticated generative models.

Neural networks and GPUs GPUs are commonly employed for training neural networks. However, the performance of such a synchronous system is constrained by the slowest learner and the slowest messages transmitted through the network. As the communication phase is a critical component, a high-performance network is essential for expediting parameter reconciliation across learners.

In configurations involving GPUs, a CPU host is typically connected to a PCIe-attached accelerator tray housing multiple GPUs. Within this tray, GPUs are interconnected using high-bandwidth interfaces such as NVlink, facilitating efficient communication and data exchange between the GPUs.

In the A100 GPU, each NVLink lane supports a data rate of $50 \times 4\,Gb/s$ in each direction. The total number of NVLink lanes increases from six lanes in the V100 GPU to 12 lanes in the A100 GPU, resulting in a total bandwidth of $600\,GB/s$. With the H100 GPU, each GPU can have up to 18 lanes, leading to a total bandwidth of $900\,GB/s$.

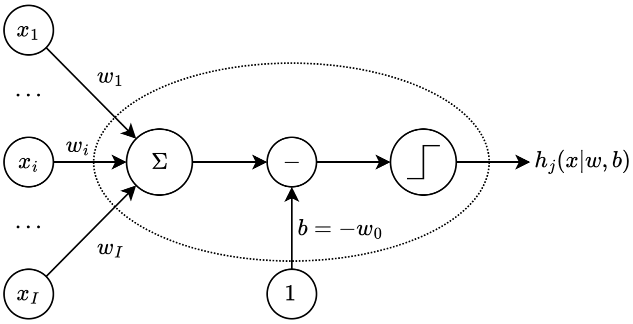


Figure 2.7: Neural network training

2.4.3 Tensor processing unit

Although GPUs are well-suited to machine learning (ML), they are still considered relatively general-purpose devices. However, in recent years, designers have increasingly specialized them into ML-specific hardware. These custom-built integrated circuits are developed specifically for machine learning tasks and are tailored to frameworks like TensorFlow.

These ML-specific hardware units have been powering Google data centers since 2015, alongside CPUs and GPUs. In TensorFlow, the basic unit of operation is a Tensor, which is an n-dimensional matrix.

Tensor Processing Units (TPUs) are extensively used for both training and inference tasks. The main versions of Tensor Processing Units are:

- *TPUv1*: inference-focused accelerator connected to the host CPU through PCIe links.
- *TPUv2*: supports both training and inference operations, providing enhanced flexibility and performance. In TPUv2, each Tensor Processing Unit (TPU) core consists of an array for matrix computations unit (MXU) and a connection to high bandwidth memory (HBM), which is used to store parameters and intermediate values during computation. Specifically, TPUv2 is equipped with 8 GiB of HBM for each TPU core, ensuring ample storage capacity. Additionally, there is one MXU allocated for each TPU core, facilitating efficient matrix computations. The hardware layout of TPUv2 comprises 4 chips, with each chip housing 2 cores. This configuration optimizes performance and scalability for machine learning tasks. Within a rack, numerous TPUv2 accelerator boards are interconnected via a custom high-bandwidth network, facilitating a collective ML compute power of 11.5 petaflops. This network's high bandwidth capabilities ensure swift parameter reconciliation with precisely controlled tail latencies. A TPU Pod, consisting of 64 units, can accommodate up to 512 total TPU cores and $4\,TB$ of total memory. This configuration maximizes both processing power and memory capacity for demanding machine learning tasks.
- *TPUv3*: Google's initial venture into liquid-cooled accelerators within their data centers. With a performance boost of 2.5 times compared to its predecessor, TPUv2, this supercomputing-class computational power enables various new ML capabilities such as AutoML and rapid neural architecture search. The TPUv3 Pod offers an impressive maximum configuration, boasting 256 devices, totaling 2048 TPU v3 cores. This setup delivers an astounding 100 petaflops of computing power and accommodates 32 TB of TPU memory, ensuring enhanced performance and scalability for demanding machine learning tasks.

- *TPUv4*: announced in June 2021, TPUv4 marks the latest advancement in Google's Tensor Processing Unit lineup. Initially deployed to bolster Google's in-house services, TPUv4 is not yet available as a cloud service. A single TPUv4 pod comprises 4096 devices, showcasing a remarkable performance increase of approximately 2.7 times compared to its predecessor, TPUv3. With computing capabilities equivalent to around 10 million laptops, TPUv4 sets a new standard for high-performance machine learning hardware.

- *TPUv5*: it has two versions (v5e and v5p). The latter one was announced in December 2023 and has been available since early this year. Compared to TPUv4, v5p can train large language models like GPT3-175B 2.8 times faster. On the other hand, although v5e is slower, it offers more relative performance per dollar compared to v5p.

On the other hand, AWS offers Trainium2, specifically optimized for Large Language Models (LLMs). Additionally, they provide Graviton4, based on ARM architecture, offering 30% greater energy efficiency for general AI tasks. For AI inference, AWS offers Inferentia2, which boasts 2.3 times higher throughput and up to 70% lower cost per inference compared to comparable EC2 instances.

2.4.4 Field programmable gate array

FPGAs (Field-Programmable Gate Arrays) are arrays of logic gates that users can program or configure in the field, without relying on the original designers. These devices consist of carefully designed and interconnected digital subcircuits that efficiently implement common functions, providing extremely high levels of flexibility. The individual digital subcircuits within FPGAs are known as configurable logic blocks (CLBs).

VHDL and Verilog are hardware description languages used to describe hardware. They enable users to create textual representations of hardware components and their interconnections. HDL code resembles a schematic, using text to define components and establish connections between them.

	Advantages	**Disadvantages**
CPU	Easily programmable and compatible Rapid design space Efficiency	Simple models Small training sets
GPU	Parallel execution	Limited flexibility
TPU	Fast for ML	Limited flexibility
FPGA	High performance Low cost Low power consumption	Limited flexibility High-level synthesis

2.5 Storage solutions

During the 80s and 90s, data was mainly produced by humans. However, in contemporary times, machines generate data at an unprecedented pace.

Various forms of media such as images, videos, audios, and social media platforms have emerged as significant sources of big data. Moreover, the widespread deployment of sensors, surveillance cameras, digital medical imaging devices, and other technologies has further accelerated the accumulation of data. This data deluge is further augmented by the integration of Industry 4.0 technologies and artificial intelligence, ushering in a new era of data-centricity and innovation.

The trend favors a centralized storage approach, which offers several advantages: By limiting redundant data, it streamlines storage efficiency. Automation of replication and backup processes ensures data reliability and security. This centralized model ultimately leads to reduced management costs.

The current trend leans towards favoring a centralized storage strategy. This approach helps in minimizing redundant data, automating replication and backup processes, and ultimately reducing management costs.

HDDs have long dominated the storage technology landscape, characterized by magnetic disks with mechanical interactions. However, recent technological advancements have introduced SSDs, which differ significantly. SSDs have no mechanical or moving parts and are constructed using transistors, specifically NAND flash-based devices. Furthermore, NVMe (Non-Volatile Memory Express) has emerged as the latest industry-standard for running PCIe SSDs. Despite these innovations, tapes persist as a reliable storage solution unlikely to fade away.

Certain large storage servers employ SSDs as caches for multiple HDDs. Similarly, some latest-generation main boards integrate a small SSD with a larger HDD to enhance disk speed. Additionally, certain HDD manufacturers produce Solid State Hybrid Disks (SSHDs) that combine a small SSD with a large HDD within a single unit.

2.5.1 Disk drives

In the view of an operating system, disks are perceived as a compilation of data blocks capable of independent reading or writing. To facilitate their organization and management, each block is assigned a unique numerical address known as the Logical Block Address (LBA). Usually, the operating system groups these blocks into clusters, which serve as the smallest unit that the OS can read from or write to on a disk. Cluster sizes typically vary from one disk sector (512 bytes) to 128 sectors (64 kilobytes).

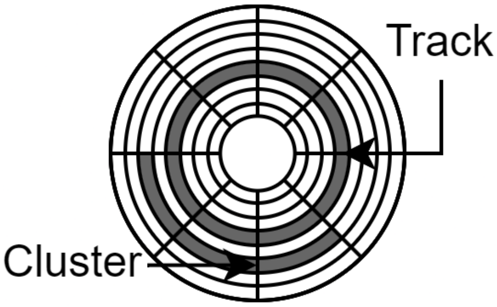


Figure 2.8: Hard disk drive structure

Clusters encompass two crucial components:

1. *File data*: this refers to the actual content stored within files.
2. *Metadata*: this includes essential information necessary for supporting the file system, which consists of:
   - File names.
   - Directory structures and symbolic links.

- File size and file type.
- Creation, modification, and last access dates.
- Security information such as owners, access lists, and encryption details.
- Links directing to the Logical Block Address (LBA) where the file content is located on the disk.

Hence, the disk can harbor various types of clusters:

- *Fixed-position metadata*: reserved to bootstrap the entire file system.
- *Variable-position metadata*: used to store the folder structure.
- *File data*: housing the actual content of files.
- *Unused space*: available to accommodate new files and folders.

Reading To read a file, the process involves:

1. Accessing the metadata to locate its blocks.
2. Accessing the blocks to read its content.

Writing The process of writing a file involves the following steps:

1. Accessing the metadata to locate free space.
2. Writing the data into the assigned blocks.

Since the file system operates on clusters, the actual space occupied by a file on a disk is always a multiple of the cluster size, denoted by c. This is given by the formula:

$$a = \left\lfloor \frac{s}{c} \right\rfloor \cdot c$$

Here, a is the actual size on disk, s is the file size, and c is the cluster size. The wasted disk space, denoted by w, due to organizing the file into clusters can be computed as:

$$w = a - s$$

This unused space is referred to as the internal fragmentation of files.

> **Example:**
> Let's consider a hard disk with a cluster size of 8 bytes and a file with a size of 27 bytes. The actual size on the disk would be:
>
> $$a = \left\lfloor \frac{s}{c} \right\rfloor \cdot c = \left\lfloor \frac{27}{8} \right\rfloor \cdot 8 = 32\,byte$$
>
> And the wasted disk space would be:
>
> $$w = a - s = 32 - 27 = 5\,byte$$

Deleting Deleting a file involves updating the metadata to indicate that the blocks previously allocated to the file are now available for use by the operating system. However, the deletion process does not physically remove the data from the disk. Instead, the data remains intact until new data is written to the same clusters. In other words, when new files are written to the same clusters, the old data is overwritten by the new data.

External fragmentation As the lifespan of the disk progresses, there may not be sufficient contiguous space available to store a file. In such instances, the file is divided into smaller chunks and distributed across the free clusters scattered throughout the disk. This process of splitting a file into non-contiguous clusters is known as external fragmentation. It's important to note that external fragmentation can significantly degrade the performance of a hard disk drive (HDD).

2.5.2 Hard disk drives

A hard disk drive (HDD) utilizes rotating disks (platters) coated with magnetic material for data storage. Information can be accessed in a random-access manner, allowing for the storage or retrieval of individual data blocks in any order, rather than sequentially. The HDD comprises one or more rigid ("hard") rotating disks (platters) with magnetic heads positioned on a moving actuator arm, facilitating the reading and writing of data onto the surfaces.

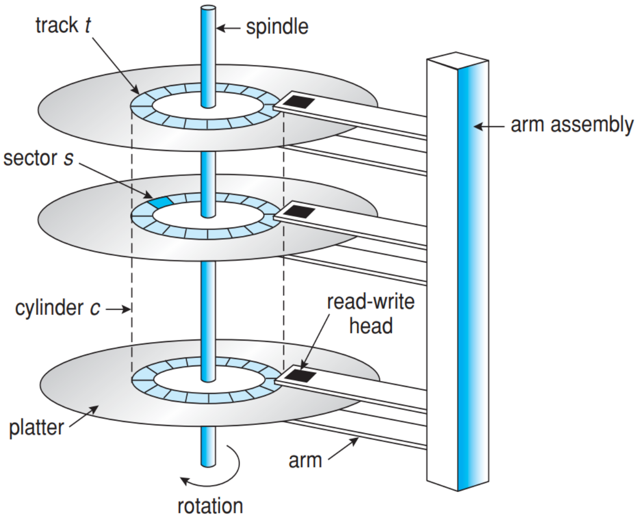


Figure 2.9: Hard disk drive structure

Externally, hard drives present a multitude of sectors (blocks), typically comprising 512 or 4096 bytes each. Each sector write is indivisible and comprises a header along with an error correction code. However, the writing of multiple sectors can be susceptible to interruption, resulting in what is known as a torn write scenario, where only a portion of a multi-sector update is successfully written to the disk.

Regarding drive geometry, sectors are organized into tracks, with a cylinder representing a specific track across multiple platters. These tracks are arranged in concentric circles on the platters, and a disk may consist of multiple double-sided platters. The drive motor maintains a constant rate of rotation for the platters, typically measured in revolutions per minute (RPM).

Present-day technology specifications include:

- *Diameter*: approximately 9 cm (3.5 inches), accounting for two surfaces.

- *Rotation speed*: ranges from 7200 to 15000 revolutions per minute (RPM).
- *Track density*: reaching up to 16,000 Tracks Per Inch (TPI).
- *Heads*: can be parked either close to the center or towards the outer diameter, particularly in mobile drives.
- *Disk buffer cache*: embedded memory within a hard disk drive, serving as a buffer between the disk and the computer.

Cache Numerous disks integrate caches, often referred to as track buffers, which typically consist of a small amount of RAM ranging from 16 to 64 MB. These caches serve several purposes:

1. *Read caching*: This minimizes read delays caused by seeking and rotation.
2. *Write caching*: There are two primary types of write caching:
    - Write back cache: In this mode, the drive acknowledges writes as complete once they have been cached. However, this feature can be risky as it may lead to an inconsistent state if power is lost before the write back event.
    - Write through cache: Here, the drive acknowledges writes as complete only after they have been written to the disk.

Today, some disks incorporate flash memory for persistent caching, resulting in hybrid drives.

Standard disk interfaces The standard disk interfaces are:

- *ST-506*: an ancient standard characterized by commands (read/write) and addresses in cylinder/head/sector format stored in device registers.
- *ATA*: advanced Technology Attachment, evolved from ST-506, and eventually standardized as IDE (Integrated Drive Electronics).
- *IDE*: integrated Drive Electronics, also known as Parallel ATA, used primarily in older systems.
- *SATA*: serial ATA, the current standard for connecting storage devices, featuring recent versions that support Logical Block Addresses (LBA).
- *SCSI*: a packet-based interface, akin to TCP/IP, with devices translating Logical Block Addresses (LBA) to internal formats such as cylinder/head/sector. It is transport-independent and utilized in various devices including USB drives, CD/DVD/Blu-ray drives, and Firewire connections.
- *iSCSI*: a variant of SCSI over TCP/IP and Ethernet, enabling storage over network connections.

Transfer time With hard disk drives we can have four types of delay:

- *Rotational delay*: this delay represents the time it takes for the desired sector to rotate under the read head. It's influenced by the disk's rotational speed, measured in revolutions per minute (RPM). The full rotation delay is calculated as:

  $$R = \frac{1}{DiskRPM}$$

  While the average rotation time is typically half of the full rotation delay, given by:

  $$T_{rotation} = \frac{60 \cdot R}{2}$$

  in seconds.

- *Seek delay*: this delay refers to the time it takes for the read head to move to a different track on the disk. The seek time includes components such as acceleration, coasting, deceleration, and settling. Modeling seek time with a linear dependency on distance, we find that the average seek time $t_{seek_{AVG}}$ is one-third of the maximum seek time $t_{seek_{MAX}}$.

- *Transfer time*: this delay is the time required to read or write bytes of data from or to the disk. It encompasses the final phase of the input/output (I/O) process, considering the data being read from or written to the disk surface. This includes the time for the head to pass over the sectors and the I/O transfer process, taking into account factors such as rotation speed and storage density.

- *Controller overhead*: this delay encompasses the additional time incurred for managing the requests sent to the disk controller. It involves tasks such as buffer management for data transfer and the time taken to send interrupts.

The service time, also known as I/O time, is determined by summing up various elements:

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer} + T_{overhead}$$

When factoring in the queue time, which represents the wait time for the resource, we can calculate the response time $\tilde{R}$ as follows:

$$\tilde{R} = T_{queue} + T_{I/O}$$

Here, T_{queue} is influenced by factors such as queue length, resource utilization, the mean and variance of disk service time distribution, and the distribution of request arrivals.

> **Example:**
> Let's consider a hard disk with the following specifications: a read/write sector size of 512 bytes (0.5 KB), a data transfer rate of 50 MB/s, a rotation speed of 10000 RPM, a mean seek time of 6 ms, and an overhead controller time of 0.2 ms. The service time is calculated as:
>
> $$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer} + T_{overhead} = 6 + T_{rotation} + T_{transfer} + 0.2$$
>
> Given:
>
> $$T_{rotation} = 60 \cdot \frac{1000}{2 \cdot DiskRPM} = 60 \cdot \frac{1000}{2 \cdot 10000} = 3.0\, ms$$

$$T_{transfer} = 0.5 \cdot \frac{1000}{50 \cdot 1024} = 0.01\,ms$$

Hence:

$$T_{I/O} = 6 + T_{rotation} + T_{transfer} + 0.2 = 6 + 3 + 0.01 + 0.2 = 9.21\,ms$$

The previous analysis of service times reflects a highly pessimistic scenario, assuming the worst-case conditions where disk sectors are extensively fragmented. This scenario occurs when files are tiny, typically occupying just one block, or when the disk suffers from significant external fragmentation. Consequently, each access to a sector necessitates both rotational latency and seek time.

However, in many practical situations, these extreme conditions are not encountered. Files tend to be larger than a single block and are stored contiguously, mitigating the need for frequent seeks and rotational latency.

We can assess the data locality of a disk by quantifying the percentage of blocks that can be accessed without requiring seek or rotational latency:

$$T_{I/0} = (1 - DL)(T_{seek} + T_{rotation}) + T_{transfer} + T_{overhead}$$

Example:
Let's revisit the previous scenario but now with a data locality of 75%. In this case, the time is computed as:

$$T_{I/0} = (1 - DL)(T_{seek} + T_{rotation}) + T_{transfer} + T_{overhead} = (0.25)(6 + 3) + 0.01 + 0.2 = 2.46\,ms$$

Here, DL represents the data locality factor. Substituting the given values, we find the resulting time to be 2.46 ms.

Disk scheduling Caching is instrumental in enhancing disk performance, although it cannot fully compensate for inadequate random access times. The core concept lies in reordering a queue of disk requests to optimize performance. Estimating the request length becomes viable by considering the data's position on the disk. Various scheduling algorithms facilitate this optimization process:

- *First come, first served* (FCFS): this is the most basic scheduler, serving requests in the order they arrive. However, it often results in a significant amount of time being spent seeking.
- *Shortest seek time first* (SSTF): this scheduler minimizes seek time by always selecting the block with the shortest seek time. SSTF is optimal and relatively easy to implement, but it may suffer from starvation.
- *SCAN*, also known as the elevator algorithm: in SCAN, the head of the disk sweeps across the disk, servicing requests in a linear order. This approach offers reasonable performance and avoids starvation, but average access times are higher for requests at the extremes of the disk.
- *C-SCAN*: Similar to SCAN, but it only services requests in one direction, providing fairer treatment to requests. However, it typically exhibits worse performance compared to SCAN.

- *C-LOOK*: this is akin to C-SCAN, but with a twist: the head of the disk only moves as far as the last request in the queue, effectively reducing the range of movement.

These scheduling algorithms can be implemented in various ways:

- *Operating system scheduling*: requests are reordered based on Logical Block Address (LBA). However, the OS lacks the ability to account for rotation delay.
- *On-disk scheduling*: the disk possesses precise knowledge of the head and platter positions, allowing for the implementation of more advanced schedulers. However, this approach requires specialized hardware and drivers.
- *Disk command queue*: found in all modern disks, this queue stores pending read/write requests, known as Native Command Queuing (NCQ). The disk may reorder items in the queue to enhance performance.

Joint operating system and on-disk scheduling can introduce challenges and potential issues.

2.5.3 Solid state drive

Solid-state storage devices, in contrast to traditional hard disk drives (HDDs), do not contain any mechanical or moving parts. Instead, they are constructed using transistors, similar to memory and processors. Unlike typical RAM, they can retain information even in the event of power loss. These devices include a controller and one or more solid-state memory components. They are designed to utilize traditional HDD interfaces and form factors, although this may be less true as technology evolves. Furthermore, they offer higher performance compared to HDDs.

An SSD's components are arranged in a grid structure. Each cell within this grid can store varying amounts of data, ranging from one bit (single-level cell) to multiple bits (multi-level cell, triple-level cell, etc.).

Internal organization NAND flash memory is structured into pages and blocks. Each page contains several logical block addresses (LBAs), while a block generally comprises multiple pages, collectively holding approximately $128 - 256\ KB$ of data:

- *Blocks* (or erase blocks): these are the smallest units that can be erased, typically consisting of multiple pages. They can be cleaned using the ERASE command.
- *Pages*: these are the smallest units that can be read/written. They are subunits of an erase block and contain a specific number of bytes that can be read/written in a single operation through the READ or PROGRAM commands. Pages can exist in three states:
  - *Empty* (or ERASED): these pages do not contain any data.
  - *Dirty* (or INVALID): these pages contain data, but this data is either no longer in use or has never been used.
  - *In use* (or VALID): these pages contain data that can be read.

Pages that are empty are the only ones that can be written to. Erasing is limited to pages that are dirty, and this must occur at the block level, where all pages within the block must be dirty or empty. Reading is only meaningful for pages in the ”in use” or ”valid” state. If there are no empty pages available, a dirty page must be erased. If there are no blocks containing only dirty or empty pages available, special procedures should be followed to gather empty pages across the disk. Resetting the original voltage to neutral is necessary before applying a new voltage to erase the value in flash memory.

It's worth noting that while writing and reading a single page of data from an SSD is possible, an entire block must be deleted to release it. This discrepancy is one of the underlying causes of the write amplification issue. Write amplification occurs when the actual volume of data physically written to the storage media is a multiple of the intended logical amount.

Example:
Let's examine an SSD with these specifications: page size of 4 KB, block size of 5 pages, drive size of 1 block, read speed of 2 KB/s, and write speed of 1 $KB/$s.

First, let's write a 4 : KB text file to the brand-new SSD. The overall writing time will be 4 seconds.

Next, let's write a 8 : KB picture file to the almost brand-new SSD. The overall writing time will be 8 seconds.

Now, suppose the text file in the first page is no longer needed.

Finally, let's write a 12 : KB picture to the SSD. It will take 24 seconds. In this case, we need to perform the following steps:

- Read the block into the cache.
- Delete the page from the cache.
- Write the new picture into the cache.
- ERASE the old block on the SSD.
- Write the cache to the SSD.

The operating system only thought it was writing 12 : KB of data when, in fact, the SSD had to read 8 : KB (2 : KB/s) and then write 20 : KB (1 : KB/s), the entire block. The writing should have taken 12 seconds but actually took $4 + 20 = 24$ seconds, resulting in a write speed of 0.5 : KB/s instead of 1 : KB/s.

As time goes on, write amplification significantly diminishes the performance of an SSD. This degradation occurs due to the wear-out of flash cells, resulting from the breakdown of the oxide layer within the floating-gate transistors of NAND flash memory. During the erasing process, the flash cell is subjected to a relatively high charge of electrical energy.

Each time a block is erased, this high electrical charge progressively degrades the silicon material. After numerous write-erase cycles, the electrical characteristics of the flash cell start to deteriorate, leading to unreliability in operation.

Flash transition layer Establishing a direct mapping between logical and physical pages is not practical. To address this, an SSD component called the FTL (Flash Translation Layer) is utilized to emulate the functionality of a traditional HDD. The FTL manages data allocation and performs address translation efficiently to mitigate the effects of Write Amplification. One method used by the FTL to reduce Write Amplification is through a technique called Log-

Structured FTL. This involves programming pages within an erased block sequentially, from low to high pages.

Garbage collection Additionally, the FTL employs garbage collection to recycle pages containing old data (marked as Dirty/Invalid) and wear leveling to evenly distribute write operations across the flash blocks. This ensures that all blocks within the device wear out at approximately the same rate. Garbage collection incurs significant costs due to the necessity of reading and rewriting live data. Ideally, garbage collection should reclaim blocks composed entirely of dead pages. The expense of garbage collection is directly related to the volume of data blocks requiring migration. To address this challenge, several solutions can be implemented:

- Increase device capacity by over provisioning with extra flash storage.
- Postpone cleaning operations to less critical periods.
- Execute garbage collection tasks in the background during periods of lower disk activity.

During background garbage collection, SSDs operate under the assumption that they can identify which pages are invalid. However, a prevalent issue arises as most file systems do not permanently delete data. To address this challenge, a new SATA command called TRIM has been introduced. With TRIM, the operating system informs the SSD about specific logical block addresses (LBAs) that are invalid and can be subjected to garbage collection. Operating system support for TRIM is available in various platforms such as Windows 7, OSX Snow Leopard, Linux 2.6.33, and Android 4.3.

Mapping table The size of the page-level mapping table can be excessively large, particularly in scenarios like a $1\,TB$ SSD where each $4\,KB$ page requires a 4-byte entry, necessitating $1\,GB$ of DRAM for mapping. To mitigate the costs associated with mapping, several approaches have been developed:

- *Block-based mapping*: employing a coarser grain approach to mapping. This approach encounters a write issue: the FTL needs to read a significant amount of live data from the old block and transfer it to a new one.
- *Hybrid mapping*: utilizing multiple tables for mapping. The Flash Translation Layer (FTL) manages two distinct tables:
  - *Log blocks*: for page-level mapping.
  - *Data blocks*: for block-level mapping.

  When searching for a specific logical block, the FTL consults both the page mapping table and the block mapping table sequentially.
- *Page mapping combined with caching*: implementing strategies to exploit data locality and optimize mapping efficiency. The fundamental concept is to cache the active portion of the page-mapped FTL. When a workload primarily accesses a limited set of pages, the translations for those pages are retained in the FTL memory. This approach offers high performance without incurring high memory costs, provided that the cache can accommodate the required working set. However, there is a potential overhead associated with cache misses.

Wear leveling The Flash memory's Erase/Write (EW) cycle is limited, leading to skewness in the cycles that can shorten the SSD's lifespan. It's essential for all blocks to wear out at a similar pace to maintain longevity. While the Log-Structured approach and garbage collection aid in spreading writes, blocks may still contain cold data. To address this, the FTL periodically reads all live data from these blocks and rewrites it elsewhere. However, wear leveling can increase the SSD's write amplification and reduce performance. A simple policy involves assigning each Flash Block an EW cycle counter, ensuring that the difference between the maximum and minimum EW cycles remains below a certain threshold, denoted as e.

Summary Flash-based SSDs are increasingly common in laptops, desktops, and datacenter servers, despite their higher cost compared to conventional HDDs. Flash memory has limitations, including a finite number of write cycles, shorter lifespan, and the need for error correcting codes and over-provisioning (extra capacity). SSDs also exhibit different read/write speeds compared to HDDs. Unlike HDDs, SSDs are not affected by data locality and do not require defragmentation.

The Flash Translation Layer (FTL) is a crucial component of SSDs, responsible for tasks such as data allocation, address translation, garbage collection, and wear leveling. However, the controller often becomes the bottleneck for transfer rates in SSDs.

2.5.4 Comparison

To assess these two memory types, we can utilize two metrics:

- *Unrecoverable Bit Error Ratio* (UBER): this metric indicates the rate of data errors, expressed as the number of data errors per bits read.

- *Endurance rating*: measured in Terabytes Written (TBW), this indicates the total data volume that can be written into an SSD before it's likely to fail. It represents the number of terabytes that can be written while still meeting the specified requirements.