Infrastructures et Stockage et de Traitement de Données - IST

Leonard Cseres | June 20, 2025

Block level storage Provides low-level persistent storage. Most are random access, some for backup are serial access (magnetic tape). Block level abstraction of fixed size (typically 4 kiB). Ex.: *Hard disk (SSD or magnetic), floppy disk, tape, CD, DVD, Blu-ray.*

Magnetic Disk / Hard Drive Spindle rotates platters at 5'400 - 12'000 rounds per minute. Data stored as magnetic orientation on surfaces. Lifespan 3-6 years.

- **Track:** Circular path on a surface.
- **Cylinder:** Tracks across all surfaces at the same distance from the spindle.
- **Sector:** Division of a track, typically stores **4 kiB**.
- **Logical Block Addressing (LBA):** Specifies starting block number and count for read/write.
- Controller optimizations:
  - **Command queueing and out-of-order execution:** Up to **32 commands**.
  - **Write caching and coalescing:** Buffers writes in RAM.
  - **Intelligent seek optimization:** Minimizes head movement.
  - **Read-ahead caching:** Anticipates sequential reads.
  - Remaps failing sectors.
- Communication with host OS using **AHCI protocol**.
- Secure Erase: by OS.

Solid-State Disks Memory cells with floating gates store electrical charge (2-10 years retention).

- Information encoding: **2 levels (SLC - 1 bit), 4 levels (MLC - 2 bits), 8 levels (TLC - 3 bits), 16 levels (QLC - 4 bits)**.
- Trade-off: more levels = higher capacity, lower reliability (**reduced Program/Erase cycles**).
- **Page:** Cells of a word line (**4-8 KB**), smallest unit for read/write.
- **Block:** Group of **128-256 pages**, smallest unit for erase.
- **Plane:** Group of **1024 blocks**, independent I/O.
- **Die:** Piece of wafer with **1-4 planes**.
- **TSOP:** Circuit board with **1-8 dies**.
- Issues compared to magnetic disks:
  - **Coarse-grained erase:** Entire block must be erased before writing.
  - **Cell wear out:** Limited erase cycles (**1k-100k cycles**).
- Controller functions:
  - **Garbage collection:** Moves valid pages, erases full dirty blocks.
  - **Wear levelling:** Rotates data to wear cells uniformly.
  - **Over-provisioning:** Extra **10%** memory cells.
- **TRIM command:** OS informs SSD when pages are no longer used.
- States of a page: **free, used, dirty**.
- Secure Erase: with ATA Secure Erase command.

File Systems Layer on top of block storage, provides abstraction of files, directories, metadata, links. Ex.: *ext2, ext3 (deprecated), ext4 (standard Linux), Btrfs, ISO 9660, ReiserFS, XFS, ZFS, NTFS, FAT/VFAT, HFS+, APFS.*

Virtual File System (VFS) Manages files and directories inside the kernel.

Data structures on disk (Ext2/3/4)

- **Block groups:** Reduce fragmentation.
- **Superblock:** Basic file system info (magic number, block size - **1024-4096 bytes**).
- **Group descriptor:** Block numbers of bitmaps.
- **Directory:** Special file with file names and **Inode numbers**.
- **Inode:** Metadata (permissions, size, times) and pointers to data blocks (direct, single, double, triple indirection).

```
fdisk /dev/sdX                # Basic interactive disk partitioning
sfdisk /dev/sdX < script.txt  # Scriptable disk partitioning
parted /dev/sdX               # Powerful command-line partitioning
gparted                       # Graphical partitioning tool (GUI)
mkfs -t ext4 /dev/sda1  # Create a file system of type ext4
mount /dev/sda1 /mnt    # Mount a file system to /mnt
umount /mnt             # Unmount the file system from /mnt
df -h           # Show mounted file systems with human sizes
findmnt /mnt    # Show mount hierarchy and options for /mnt
fsck /dev/sda1  # Check and repair file system (must be unmounted)
ln file_a file_b               # Hard link
ln -s /path/to/file file_link  # Symbolic (soft) link
```

Hard Links New directory entry pointing to the same Inode. Limited to the same file system/partition.

- Original deleted/recreated, link breaks.

Symbolic Links (symlinks) Special file with its own Inode, containing the path to the original file. Can span file systems.

- Different file system. Point to directory. Distinguish original and link. Point to another link. Original moved, link breaks.

Virtual Storage Blocks replaced by pointers in an index (table) to physical blocks. Physical disks can be local (LVM) or remote (SAN).

- Motivation: Big capacity, dynamic resizing, flexibility, sharing.
- **Snapshots:** Read-only, near-instant copy of a virtual disk (only index copied initially).
- **Copy-on-write:** Before modifying a block in the active disk after a snapshot, a copy is made.

Logical Volume Management (LVM) Layer of indirection between block devices and file system. (LVM on Linux, LDM on Windows).

- Features: Flexible capacity, resizeable pools, online data relocation, convenient naming, disk striping, mirroring, snapshots.
- Components: **Physical Volumes (PVs)**, **Volume Groups (VGs)**, **Logical Volumes (LVs)**.
- **Physical Extents (PEs)** and **Logical Extents (LEs):** Chunks of storage (default **4 MB**).

```
pvscan                            lvreduce -L -2G /dev/myvg/mylv
vgscan                            vgremove myvg
lvscan                            lvremove /dev/myvg/mylv
pvcreate /dev/sda1                vgchange -ay myvg
vgcreate myvg /dev/sda1 /dev/sdb1 vgchange -an myvg
lvcreate -L 10G -n mylv myvg      lvdisplay /dev/myvg/mylv
vgextend myvg /dev/sdc1           vgdisplay myvg
lvextend -L +5G /dev/myvg/mylv    pvdisplay /dev/sda1
```

Networked Storage

Direct-Attached Storage (DAS) Directly connected to a computer.

Network-Attached Storage (NAS) File server providing file-level access over a network (e.g., NFS, SMB).

Storage Area Network (SAN) High-performance network providing block-level access to servers (e.g., iSCSI, Fibre Channel).

Cloud Block Storage - AWS Elastic Block Store (EBS) Must be in the same **Availability Zone (AZ)** as the EC2 instance. One volume per instance.

- Replicated within an AZ for durability.
- **Snapshots:** Incremental backups to S3, replicated across AZs.
- EBS volume types:
  - **General Purpose SSD (gp2/gp3):** Most workloads.
  - **Provisioned IOPS SSD (io1/io2):** High-performance, consistent IOPS (up to **64'000 IOPS**).
  - **Throughput Optimized HDD (st1):** Big data, data warehouses (up to **500 MiB/s**).
  - **Cold HDD (sc1):** Infrequently accessed, lowest cost (up to **250 MiB/s**).
- **Maximum volume size: 16 TiB**.
- Snapshot pricing: **~$50/TB-month**.

Object Storage - AWS Simple Storage Service (S3) Unlimited capacity, supports very big objects (up to **5 TB** per object).

- Accessed via RESTful CRUD API (HTTP).
- Buckets are globally unique, located in one **Region**, cannot be renamed.
- Storage classes: **Standard, Intelligent-Tiering, Standard-Infrequent Access, One Zone-Infrequent Access, Glacier Instant Retrieval, Glacier Flexible Retrieval, Glacier Deep Archive**.
- Cost components: Storage, Requests, Data Transfer OUT.
- S3 bucket URLs: **Path-style** (deprecated) and **virtual hosted-style**.

Cloud File Storage - Amazon Elastic File System (EFS) File storage in the AWS Cloud, shared, elastic. Petabyte-scale, low-latency. Supports NFSv4. Compatible with Linux EC2 instances. Mount targets in VPC subnets, one per AZ.

IAM User An individual or application.

IAM Group Collection of users with the same permissions.

IAM Role Identity that can be assumed by users, applications, or services, provides temporary credentials.

Permission Policy JSON document defining Allow or Deny rules for actions on resources.

Principal IAM entity making a request.

Action Operation to be performed (e.g., s3:GetObject, ec2:StartInstances).

Resource AWS resource to act upon (identified by ARN - Amazon Resource Name).

Effect Allow or Deny. Explicit Deny overrides any Allows.

- Policy evaluation: Implicit Deny by default if no matching Allow policy. Deny overrides Allow.
- **Identity-based policies:** Attached to IAM users, groups, or roles.
- **Resource-based policies:** Attached to a resource (e.g., S3 bucket policy).

Serverless Zero server ops (auto-scaling, no provisioning). No compute costs when idle. Stateless. Asynchronous, concurrent, easy to parallelize.

- Initialization code: code outside the handler method
- Local storage: `/tmp`
- `event`: information about the event
- `context`: func. name, req. id, deployment params, remaining time, env

Deployment Parameters Memory, CPU, Timeout, Concurrency, Environment variables

Pricing Resources consumed (memory × CPU). Number of invocations. Ingress/egress traffic.

- Outside of these phases the platform freezes the execution environment
- After invocation, the platform will keep the execution environment for some time for optimizing
- The function must be stateless, but the reuse of the exec. env. can be used for caching
  - Objects outside the handler function
  - `/tmp`
  - Background procs. or callbacks initiated by the function and did not complete when it ended resume if the platform reuses the execution environment

Permissions

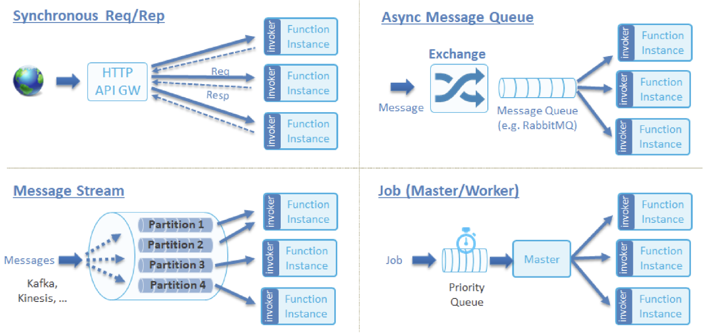
- Policies allowing other AWS services to invoke the function
- (#) Policies allowing the function to access other services

```
Version: "2012-10-17"
Statement:
  - Effect: "Allow"
    Principal:
      Service: "apigateway.amazonaws.com"
    Action: "lambda:InvokeFunction"
    # Action: ["s3:GetObject", "s3:ListBucket"]
    Resource: "*"
    # Resource:
    #   ["arn:aws:s3:::your-bucket-name/*",
    #     "arn:aws:s3:::your-bucket-name"]
```

S3 Object Lambda

- Use cases: redaction, conversion, augmentation, compression, resizing, watermarking, custom row level authorization

S3 Access Point Alias for a bucket with custom permissions



S3 Object Lambda Access Point Enhanced S3 access point

Configured with:

- Standard S3 access point
- AWS Lambda transform function
- (optional) IAM policy to restrict access to this access point

Cold Start Problem Mitigations

- Not only in FaaS, also in auto-scaling
- Reduce dependencies and optimize function initialization
- Fixed allocation of a set of VMs to run the function

FaaS platform

- Control plane: developer-facing API
- Data plane: underlying infrastructure

1. Call hits the LB
2. Frontend Invoke (entrypoint, retrieves metadata)
3. Counting Service (count invocations and concurrent functions)
4. Worker Manager (manages the instances, sandboxes, lifecycle)
5. Worker (compute instance, running the sandboxes)
6. Placement Service (optimizes the intelligent placement of the function, minimize the cold start, ensure optimal utilization, monitors health of workers)

```
Code < Runtime < Sandbox <  GuestOS  < Hypervisor < HostOS < Hardware
#     one function       |one account|          many accounts
```

Functions Security Functions are isolated from each other

- **Environment isolation**: customize the environment without affecting other
- **Security isolation**: data is not shared
- **Performance isolation**: “noisy neighbors” should not infer with other

Sandboxes are built from the same techo. as Docker

- One Sandbox executes several function invocations in serial
- A Sandbox is never reused across several functions

Hundreds or thousands of Guest OS’s may run on a physical host

- Guest OS’s are shared within an account
- A Guest OS is never reused across accounts

Data

- **OLAP**: Online analytical processing
- **OLTP**: Online transaction processing

Data Warehouse (1990s) Central repository of key data ingested from OLTP

- Data is well-integrated, highly structured, highly curated, and highly trusted

Hadoop & Big Data (2006) Data has high velocity, high volume, high variety

Data Lake (2011) “Store everything just in case”. Raw data, without optimizations. Data is unstructured, semi-structured, and structured. Usage of S3.

Data Lakehouse (2020) Data lake with a data warehouse. Optimized binary format (Parquet). Concurrent reads/writes (Delta Lake).

Data mesh (2021) Each company division has its own independent data lake. More agility.

MapReduce (2004) Programming model for parallel processing of large data sets

HDFS Hadoop Distributed File System

- Files are distributed across multiple nodes
- Replicated across multiple nodes 3+ times
- Chunks of 64MB
- **Move the processing to the data**

HDFS design decisions

- Files stored as chunks
- Reliability through replication
- Single master to coordinate access, keep metadata
- No data caching
- Simplify the API
- **HDFS namenode**
  - manages the file system (file structure, metadata)
  - coordinates file operations (no data is moved through the namenode)
  - maintains overall health (block re-replication & rebalancing, GC)
- **HDFS datanode**: manages the chunks

Hadoop 1.x cluster

- Per cluster:
  - One Namenode (NN): master node for HDFS
  - One Jobtracker (JT): master node for job submission
- Per slave machine:
  - One Tasktracker (TT): contains multiple task slots
  - One Datanode (DN): serves HDFS data blocks

Binary file formats

- Avro (Hadoop)
- Thrift (Facebook)
- Protocol Buffers (Google)
- Column-oriented binary formats: (Optimized for analytics, efficient storage and queries)
  - Parquet (Twitter & Cloudera)
  - ORC (Hadoop)

Parquet *Column-based* format - files are organized by column, rather than by row, which saves storage space and speeds up analytics queries

- Row groups: Each group contains column chunks (contiguous in file)
- Metadata at file end: chunk positions, min/max stats, dictionary filtering
- Supports sorting on one column for efficient filtering
- Encoding: Plain, Dictionary, Run-Length Encoding (RLE), Delta encoding
- Compression: gzip, Snappy, LZO
- Data types: BOOLEAN, INT32/64/96, FLOAT, DOUBLE, BYTE_ARRAY
- Logical types: STRING, ENUM, UUID, DECIMAL, DATE, TIME, TIMESTAMP, JSON, LIST, MAP
- Tools: pqrs to inspect schema, content, metadata, convert to CSV/JSON

Hive Metastore & AWS Glue Data Catalog

- Hive Metastore: Metadata DB for tables, columns, types, file locations, partitions
- SerDe (Serializer/Deserializer): Reads/writes tables in various formats (Avro, ORC, Parquet, CSV, JSON, Regex, custom)
- AWS Glue Data Catalog: Hive-compatible, shared by AWS services (Athena, Glue, Redshift, etc.)
  - Tables point to folders (not files); new files = new data
  - If a file misses a column, value is null
  - Populated by DDL, UI, or Glue crawlers (auto schema inference)

Partitioning & Best Practices

- Partitioning: Split table data into subfolders (e.g., by year/month/day)
- Partition keys become virtual columns
- Queries using partition keys only scan relevant files
- Hive-style: Folders named key=value (recommended)
- Glue crawlers auto-detect partitions and add keys to schema

Data Pipelines & Transformations

- ETL: Extract, Transform, Load
- Typical steps: Ingest → Transform → Store → Catalog → Consume
- Ingestion: Batch (one-time, scheduled), Streaming (real-time)
- Transformations: Format conversion (e.g., CSV→Parquet), standardization, quality checks, partitioning, denormalization, cataloging
- Tools: AWS DMS (Database Migration Service), Kinesis (streaming), Glue (ETL), Elastic MapReduce (Hadoop/Spark)

AWS Glue

- Crawler: Discovers schema, writes to catalog
- Job: ETL (Python, Spark, or visual editor)
- Run: On demand, event, or schedule

Data Consumption

- AWS Athena: Serverless SQL queries on data lake (CSV, JSON, Parquet, ORC, Avro, logs, etc.)
  - Based on Presto/Trino
  - Requires tables in Glue Data Catalog
  - Pay per data scanned
- Amazon Redshift/Spectrum: Data warehouse/lakehouse
- QuickSight: Visualization
- Sagemaker: Machine learning

Presto / Trino

- Distributed SQL query engine (originally by Facebook)
- Interactive queries at petabyte scale
- Used by Athena, Netflix, others
- Coordinator/worker architecture

Delta Lake

- Stores table data as Parquet files + _delta_log/ with JSON commit files (transaction log)
- Supports versioning, time travel, concurrent access
- Checkpoints (every 10 commits) for fast recovery
- Open source, supported by Spark, Presto, Trino, Hive, etc.

```sql
-- Create a table
CREATE TABLE table_name (
  id INT,
  name STRING
);

-- Create an external table with CSV format
CREATE EXTERNAL TABLE table_name (
  col1 INT,
  col2 STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES ('field.delim' = ',')
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
     'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://your-bucket/path/'
TBLPROPERTIES ('classification' = 'csv');

-- Create a Parquet table with partitions
CREATE EXTERNAL TABLE table_name (
  col1 STRING,
  col2 DOUBLE
)
PARTITIONED BY (year STRING, month STRING)
STORED AS PARQUET
LOCATION 's3://your-bucket/path/'
TBLPROPERTIES ('classification' = 'parquet');

-- Select data
SELECT * FROM table_name;
```

```
-- Filter data
SELECT * FROM table_name WHERE col1 = 'value';

-- Insert data
INSERT INTO table_name VALUES (1, 'example');

-- Create a view
CREATE VIEW view_name AS
SELECT col1, col2 FROM table_name;

-- Drop a table
DROP TABLE table_name;

-- Repair partitions (for partitioned tables)
MSCK REPAIR TABLE table_name;

-- Add a partition manually
ALTER TABLE table_name ADD PARTITION (year='2025', month='01')
LOCATION 's3://your-bucket/path/year=2025/month=01/';

-- Show partitions
SHOW PARTITIONS table_name;

-- Describe table structure
DESCRIBE table_name;
```