nffa-di

Piano Nazionale di Ripresa e Resilienza

AREA SCIENCE PARK

Data Infrastructure

Lecture 3: File System & Processors

Federica Bazzocchi
20 and 24/4/2026

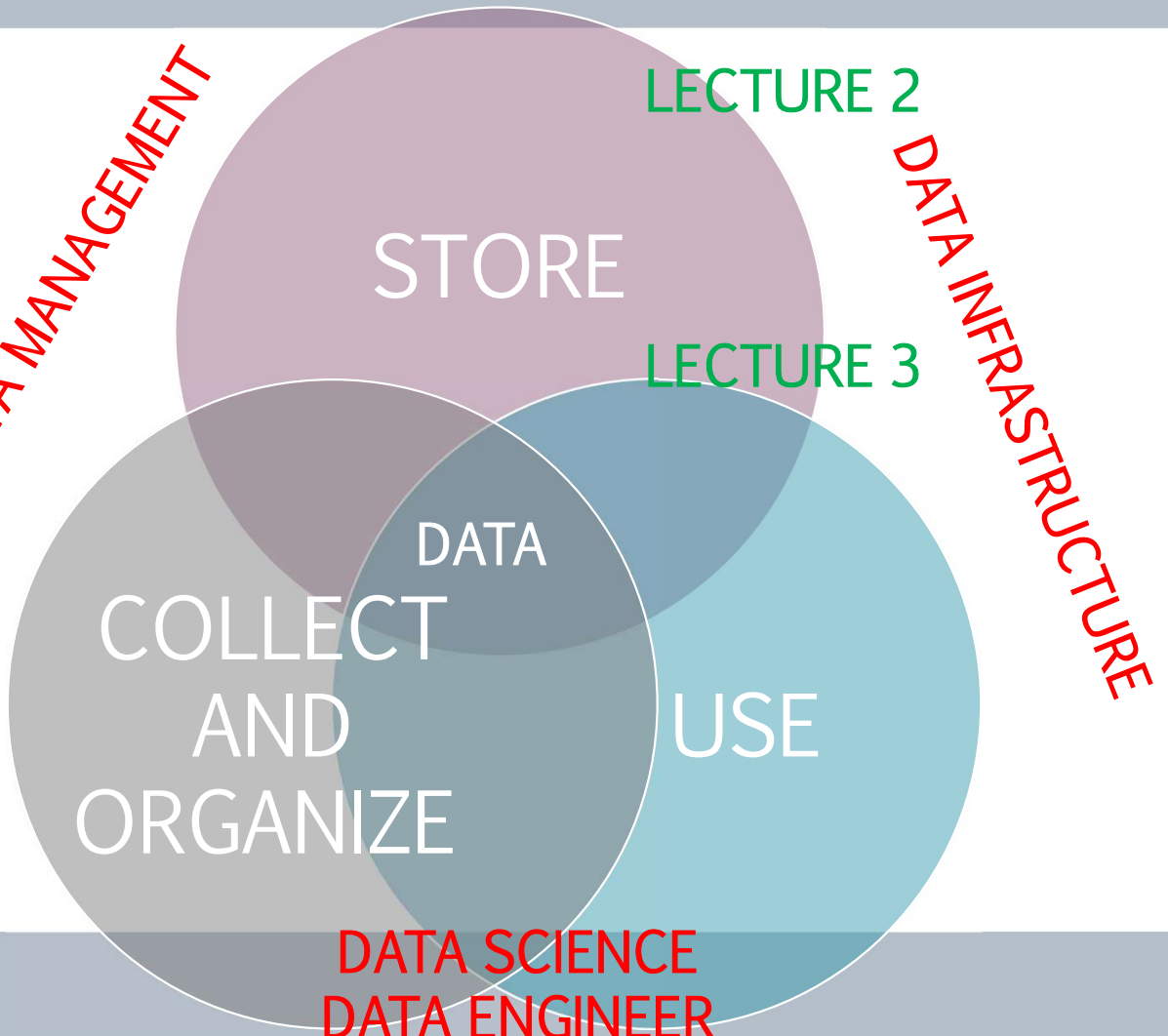
π

LECTURE 1

RESEARCH DATA MANAGEMENT

LECTURE 2

DATA INFRASTRUCTURE

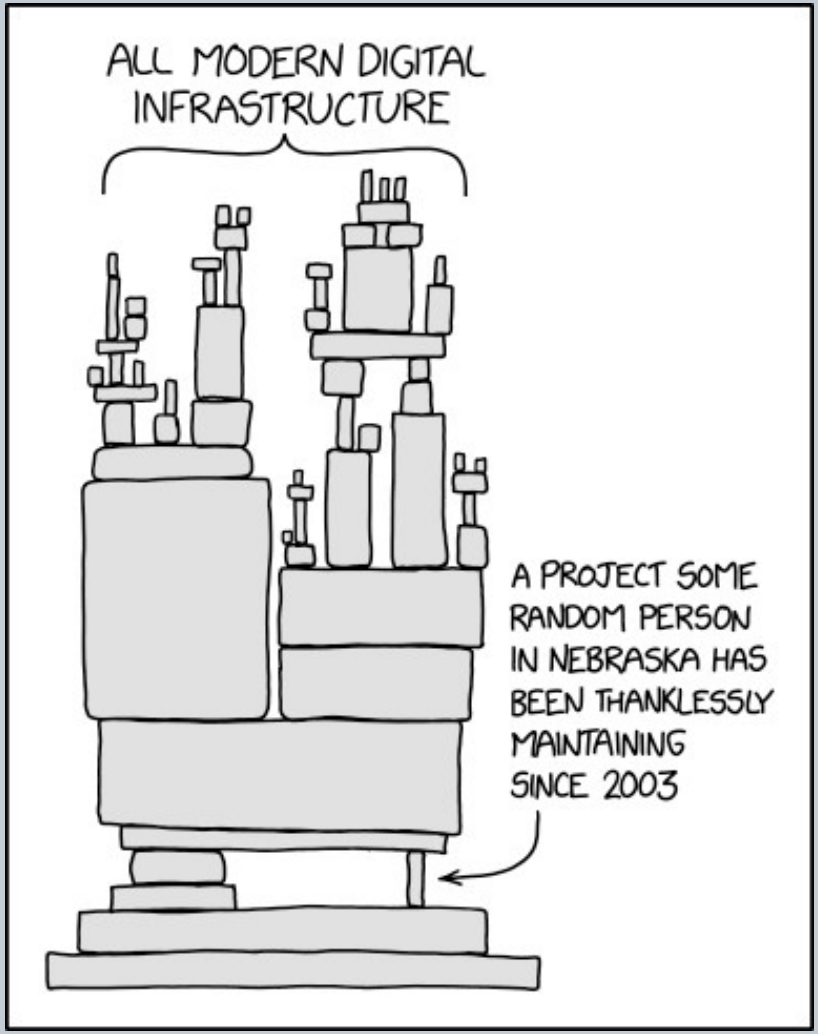


DATA SCIENCE
DATA ENGINEER
AI ENGINEER

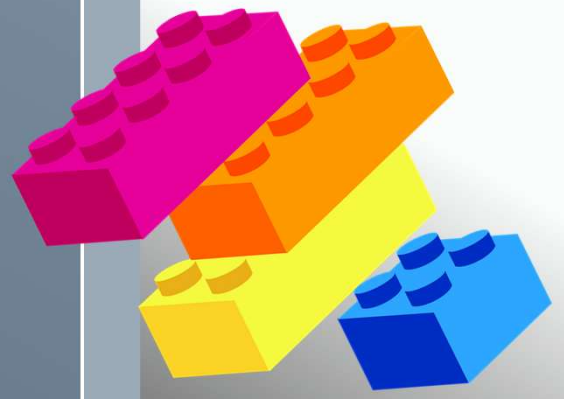
π

LECTURE 3 OUTLINE:

- File System (and bucket storage)
- Hardware components: processing units
- Networking and communication (2 words)



Generic Data Infrastructure



SOFTWARE COMPONENT

HARDWARE COMPONENT

Core Hardware Components

- Storage Systems: HDD, SSD, NAS, SAN
- Servers: Compute power for running applications and processing data
- Networking: Switches, routers, firewalls, load balancers
- Data Centers: Physical facilities for computing and storage
- Additional Systems: Backup and Disaster Recovery Systems

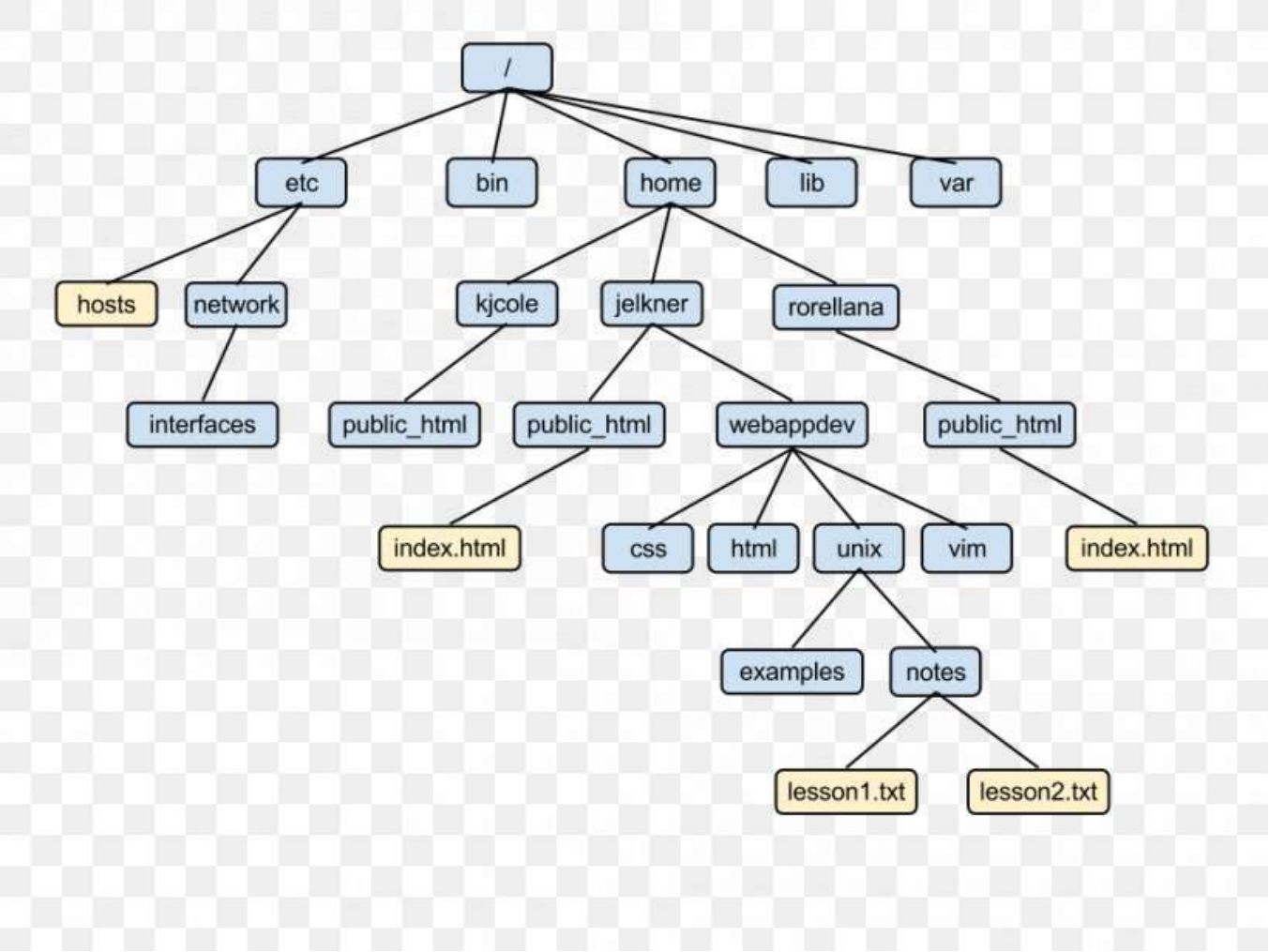
There are rules to organize the warehouse...

FILE SYSTEM

WHAT IS
A FILE SYSTEM?

- it is a storage system on a memory support that structures files and directories and organizes different actions on the files (writing, reading, modifying, deleting,...etc)
- The structure is such that file identification and access are as fast as possible
- It maintains data integrity and it is essential for file operations, access control, metadata management, and backups

A file system is the set of rules, structures, and tools that turn "chaos" into accessible files.



We are so used to see data organized in this way that we apply it even in other context (S3)

Some definitions

•**Disk:** a permanent storage medium of a given size

•**Block**: This is the smallest unit of storage that a file system (FS) can manage. It is used universally in file systems to organize and store data.

•**Partition:** A partition is a logical division of a physical storage device. It allows a single disk to be split into multiple sections, each of which can have its own file system. A subset of all the blocks on a disk.

•**Volume**: A volume can refer to a single partition or a collection of partitions that are managed together.

•**Metadata**: data that describes other data. In FS context it includes information about files such as their names, sizes, creation dates, modification dates, permissions, and locations on the disk. Metadata helps the file system manage and access files efficiently.

•**Journaling:** a technique used by FS to keep track of changes not yet committed to the main FS. It maintains a log (journal) of changes, which helps in recovering the FS to a consistent state in case of a crash or power failure. This enhances the reliability and integrity of the file system.

•**Attribute**: Attributes are properties or characteristics of files and directories in a FS. They can include metadata like read only status, hidden status, system file status, and more. Attributes help define how files and directories are treated by the operating system and applications. (subtle difference between metadata and attributes, in FS contexts attributes are those metadata that defines behavior of or restriction on a file)

Storage Management and Data Retrieve I/II

- Storage Allocation:
  - The file system (FS) divides the available storage space on a device into the smallest manageable units, called blocks or sectors.
- File Storage:
  - When a file is saved, the FS allocates these units to store the file's data. Larger files are divided into smaller segments, distributed across multiple non contiguous units.
- Metadata management:
  - FS controls for each file/directory metadata as name, dimension, creation date, last modified date, permissions. Metadata helps the file system manage and access files efficiently.
- Directories structure:
  - Hierarchical structure to allow users and application to look for files easily.
- Access control:
  - FS manages users file and directories permissions.

Storage Management and Data Retrieve II/II

- File operations:
  - FS gives mechanism to operated on files and. These operations may imply metadata update.
- Consistency and Integrity:
  - Advanced FSs implement measures to ensure data consistency and integrity, as well as error detection and correction in case of system failure.
- Optimization:
  - FSs use various mechanisms to improve performance. For example, they may perform caching to speed up access to frequently used data or organize data to minimize disk seek times.

Types of File Systems I/II

- Disk based File Systems:
  - meet various storage requirements, including the need for large files, high storage capacities, and robust data recovery capabilities
  - include features to manage disk space efficiently, such as support for large blocks, which reduces overhead, and compression, which makes better use of available space.
  - NTFS (win), ext4 (linux), HFS+, (macOS) , exFAT (multiplatform)
- Flash File Systems:
  - designed for flash memory storage devices, such as USB flash drives,
  - customized to integrate the specific characteristics and limitations of flash memory,
  - F2FS, JFFS)
- Tape File Systems:
  - designed to manage data in magnetic tape storage.
  - primarily used for backup, archiving, and data recovery due to its cost efficiency.
  - optimized for large sequential data transfers and incorporate robust error correction mechanisms to ensure data integrity over time.
  - Modern tape FSs allow users to interact with tape storage similarly to other removable media enabling file drag and drop.
  - Linear Tape FS (LTFS)

Types of File Systems II/II

- Network File Systems:
  - allow multiple users and systems to share files over a computer network
  - provide a seamless interface for reading, writing, and executing files.
  - sharing and management of resources across different systems and platforms.
  - Network File System (NFS) for Unix and Linux, server Message Block (SMB) for Windows, Apple Filing Protocol (AFP) for macOS.
- Shared Disk File Systems (also Cluster File Systems):
  - provide simultaneous access to a portion of disk storage from multiple file system servers or nodes
  - enable a high level of data sharing and collaboration among clustered systems,(suitable for environments requiring high availability, scalability, and performance).
  - GFS

Types of File Systems II/II

Feature	Network File System (NAS/NFS)	Shared/Parallel File System
Architecture	Client-Server (Centralized)	Distributed (Clustered / P2P)
Bottleneck	NAS Controller (Metadata)	Consistency Protocols (Locking)
Scalability	Vertical (Scale-up)	Horizontal (Scale-out)
Complexity	Low (Standard Configuration)	High (Requires expert tuning)
Typical Use Case	Corporate files, Home dirs, Backup	HPC, AI Training, 8K Video Editing

Examples of Common File Systems

- FAT / FAT32 – legacy systems and USB drives
- exFAT – modern portable devices
- NTFS – Windows OS with security and recovery
- ext4 – Linux default file system
- APFS – optimized for Apple SSDs
- HFS / HFS+ – older Apple systems
- GFS – shared disk access in Linux clusters
- UDF – optical media (CD, DVD, Blu ray)

Inode in Unik like FS

Our warehouse “registry”!

- An inode is a data structure used in Unix like file systems
- It stores metadata about a file, not the file name or content
- Inode number is a unique identifier of a file in the FS
- Directories are special files that map names to inode numbers

Inode structure and type of Meatdata

- address of data block on the storage media description of the file (POSIX)
- Size of the file
- Storage device ID
- User ID of the file's owner.
- Group ID of the file.
- File type
- File access right
- Inode last modification time (ctime)
- File content last modification time (mtime),
- Last access time (atime).
- Count of hard links pointed to the inode.
- Pointers to the disk blocks that store the file's contents

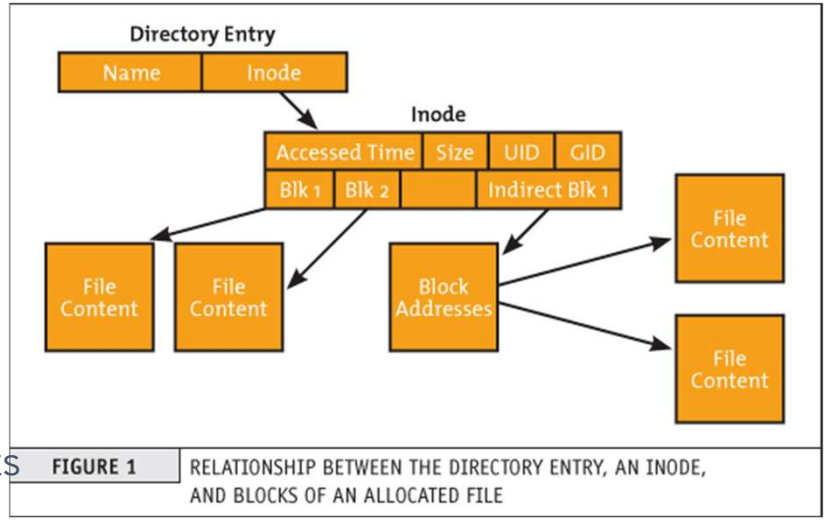


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

How Inodes Work

- File names are stored in directories, not in the inode
- Directory entries link names to inode numbers
- The file system uses the inode to access file metadata and data blocks
- Hard links create multiple directory entries pointing to the same inode

Useful for ...

- Efficient file metadata management
- Support for multiple names (hard links)
- Key to file system integrity and navigation
- Limited number of inodes can impact storage usage
- 'df i' command shows inode usage in Linux

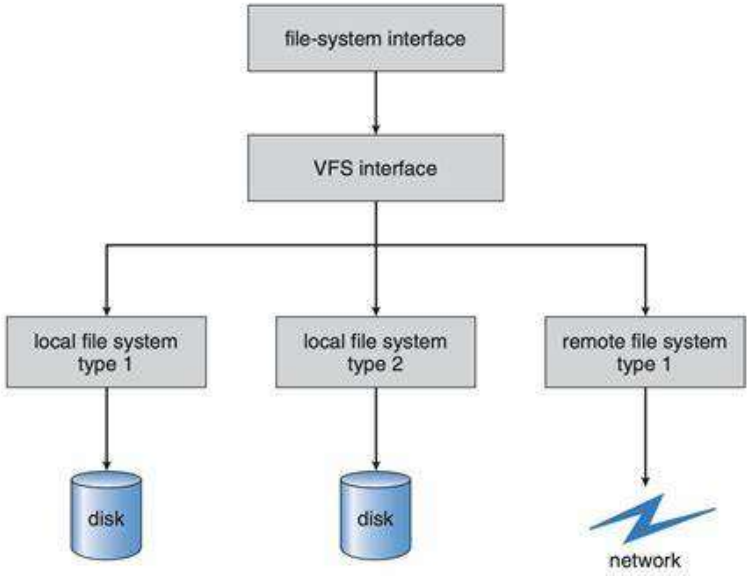
```
➜  ~ sudo tune2fs  l /dev/nvme0n1p2
tune2fs 1.47.0 (5 Feb 2023)
Filesystem volume name:   <none>
Last mounted on:          /boot
Filesystem UUID:          477d3ecb bb5d 44e0 a32a eea94eabcfa0
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index orphan_file
filetype needs_recovery extent 64bit flex_bg metadata_csum_seed sparse_super large_file
huge_file dir_nlink extra_isize metadata_csum orphan_present
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              65536
Block count:              262144
Reserved block count:     13107
Overhead clusters:        12949
Free blocks:              152733
Free inodes:              65492
First block:              0
Block size:               4096
Fragment size:            4096
Group descriptor size:    64
Reserved GDT blocks:      127
Blocks per group:         32768
Fragments per group:      32768
```

Continue.....

```
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Tue May 21 17:34:15 2024
Last mount time:          Wed Apr  2 09:29:24 2025
Last write time:          Wed Apr  2 09:29:24 2025
Mount count:              214
Maximum mount count:       1
Last checked:             Mon Aug 26 09:12:26 2024
Check interval:           0 (<none>)
Lifetime writes:          2810 MB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:              256
Required extra isize:     32
Desired extra isize:      32
Journal inode:            8
Default directory hash:   half_md4
Directory Hash Seed:      d1a60ac2 e26e 46f8 8515 678a2a95f4d4
Journal backup:           inode blocks
Checksum type:            crc32c
Checksum:                 0xd1bd87ec
Checksum seed:            0x6ae0df22
Orphan file inode:        12
```

Virtual File System (VFS)

- An abstraction layer on top of actual file systems
- Provides a uniform interface to interact with files (simplifies user and program interaction with files)
- Supports multiple underlying file systems (ext4, NFS, etc.)
- Used by the Linux kernel to manage file operations
- Unifies access to different storage types
- Enables mount points for local and networked file systems



/proc – Process Information

- Virtual file system providing process and system info
- Located at /proc
- Contains files and directories for every running process
- Read only and dynamically generated by the kernel

Common /proc Entries

- /proc/cpuinfo – CPU details
- /proc/meminfo – Memory usage
- /proc/<PID>/status – Info on a specific process

/sys – System and Hardware Info

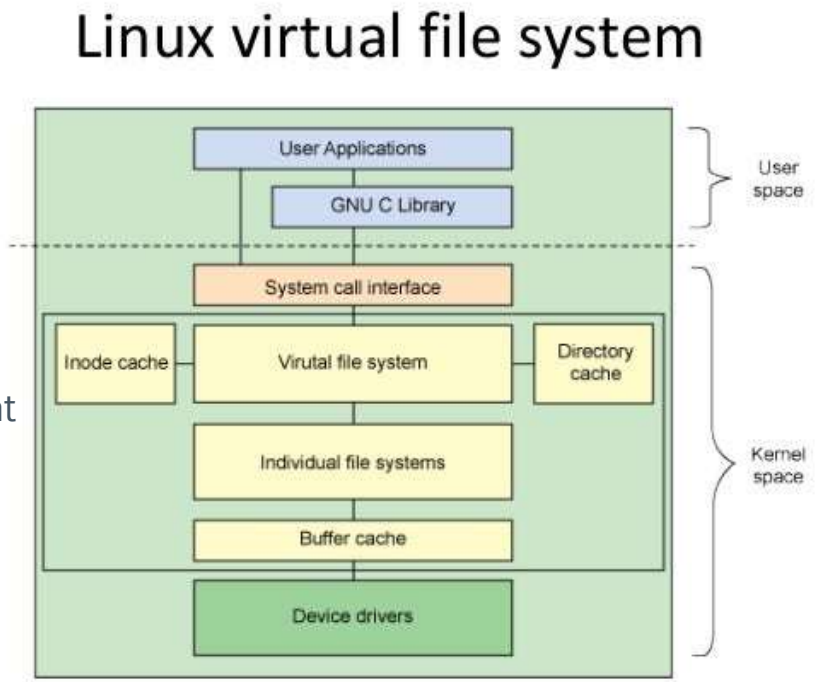
- Exposes kernel devices and modules
- Located at /sys
- Provides runtime access to kernel data structures
- Used for device configuration and system control

How /proc and /sys Work

- Not stored on disk, no actual I/O
- Kernel intercepts file operations and provides content
- Content changes in real time with system state
- Useful for diagnostics and monitoring

Command Line Examples

- cat /proc/cpuinfo
- cat /proc/meminfo
- ls /sys/class/net



```
[fbazzocchi@login02 ~]$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz
stepping        : 4
microcode       : 0x200005e
cpu MHz         : 2194.806
cache size      : 16384 KB
physical id     : 0
siblings        : 8
core id         : 0
cpu cores       : 8
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx
fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon rep_good nopl xtopology
cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma cx16 pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt
tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault pti ssbd
ibrs ibpb stibp tpr_shadow flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2
erms invpcid rtm mpx avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl
xsaveopt xsavec xgetbv1 xsaves arat vnmi umip pku ospke md_clear flush_l1d arch_capabilities
vmx flags       : vnmi preemption_timer posted_intr invvpid ept_x_only ept_ad ept_1gb flexpriority
apicv tsc_offset vtpr mtf vapic ept vpid unrestricted_guest vapic_reg vid shadow_vmcs pml tsc_scaling
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa
mmio_stale_data retbleed gds bhi ibpb_no_ret
bogomips        : 4389.61
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

```
[fbazzocchi@login02 ~]$ ls /sys/class/net/
ens18  ens19  ibs16  lo
```

Show all the network interfaces at disposal

2 Ethernet (on slot)

infiniband

Looback interface (internal network interface)

```
[fbazzocchi@login02 ~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether bc:24:11:42:aa:27 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
3: ens19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether bc:24:11:5d:c0:4d brd ff:ff:ff:ff:ff:ff
    altname enp0s19
4: ibs16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2044 qdisc mq state UP mode DEFAULT
group default qlen 256
    link/infiniband 00:00:09:25:fe:80:00:00:00:00:00:00:b8:59:9f:03:00:dc:af:e1 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    altname ibp0s16
```

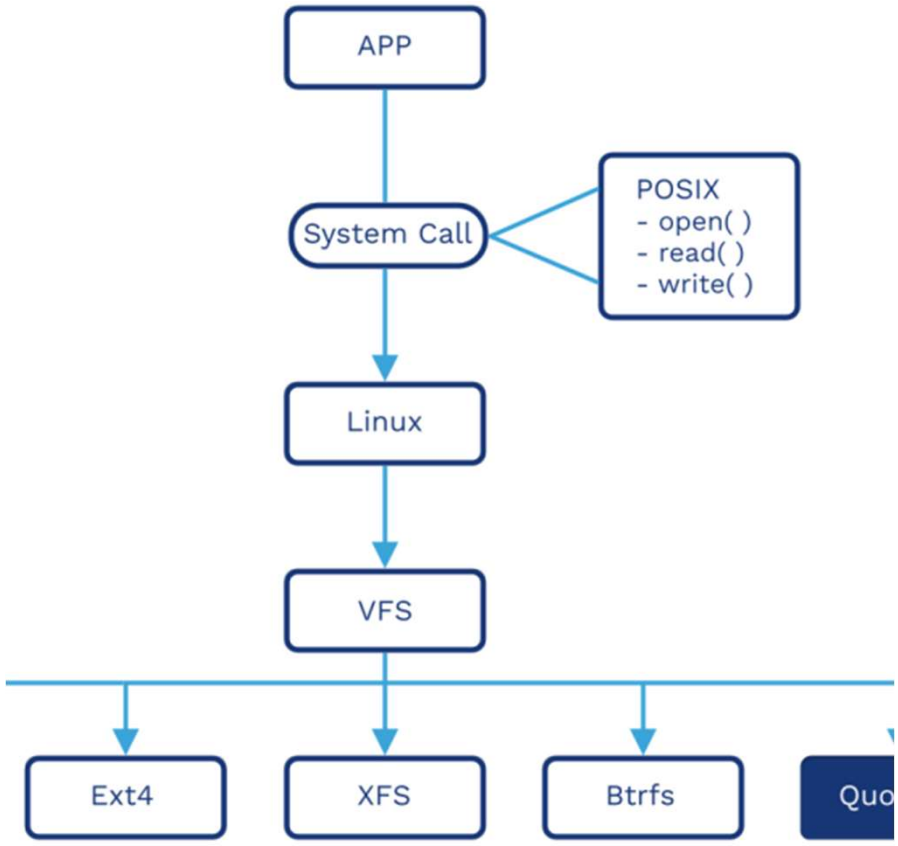
POSIX INTERFACE

- POSIX stands for Portable Operating System Interface
- A family of standards specified by IEEE (world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. Below, you can find IEEE's mission and vision statements.) for maintaining compatibility between operating systems
- Defines API, command line shells, and utility interfaces
- Ensures software developed on one POSIX compliant system runs on another with minimal changes (interoperability among systems)

- Key Components of POSIX
  - System APIs: Standard C function calls (e.g., open, read, write)
  - Shell and utilities: Commands like ls, cp, mv, grep
  - File and directory handling standards
  - Process control and threading interfaces (pthreads)
  - Signals, timers, and inter process communication (IPC)

POSIX

- Promotes software portability across UNIX like systems
- Provides consistent developer interface
- Widely adopted by Linux, macOS, BSD variants
- Supports cross platform application development
- Facilitates open source and standards based system design
- Standard I/O interface across many platforms
- open, read/write, close functions in C/C++/Fortran
- It allows buffered file I/O (streams) within (c/sdtio)
- Posix assumes atomicity and ubiquity ·
- Changes are visible immediately to all clients



POSIX I/O Limitations in Parallel Environments

- Global file pointer: leads to race conditions
- Posix assumes atomicity: difficult to manage concurrent writes
- File viewed as flat byte stream, no parallel layout awareness
- High overhead when synchronizing access manually
- POSIX requires a strict consistency to sequential order :
  - lock
  - Create a directory is an atomic operation with immediate global view
- No support for non continuous I/O

What is MPI IO?

- MPI IO is the I/O interface defined by the MPI 2 standard
- Provides parallel file access for distributed memory systems
- Allows multiple MPI processes to read/write files concurrently
- Uses collective operations and file views for efficiency

MPI IO Advantages Over POSIX

- File views: define non overlapping regions per process
- Collective I/O: optimized group operations (e.g. write_all)
- Decouples logical layout from physical layout
- Leverages parallel file systems (e.g. Lustre, GPFS) efficiently

Parallel File System

- BeeGFS
  - Developed at Fraunhofer Institute, freely available not open
  - Home - BeeGFS - The Leading Parallel Cluster File System
- Lustre
  - open and Free owned by Intel DDN
  - Intel no longer sells tools to manage and support ($$$)
  - Lustre
- GPFS (now known as Spectrum Scale)
  - IBM proprietary $$$
  - Very nice solution and expensive ones
- ...many others (WekaIO/MooseFS/Panasas... etc)
- CephFS (see later)

Feature	Standard FS (e.g., NFS, EXT4)	Parallel FS (e.g., Lustre, GPFS)
Metadata	Centralized (Bottleneck)	Distributed / Separated
Locking	Blocking (Serializes all access)	Distributed (Parallelized)
Data Path	Single server / controller	Multiple storage nodes in parallel
Ideal Use Case	Office, Home, Small App	Simulation, AI Training, Big Data

Ceph:Open source distributed storage system

CephFS

- CephFS is a distributed file system built on top of Ceph
- POSIX-compliant: supports standard file operations
- Designed for scalability, redundancy, and high availability
- Used in HPC, cloud, and data center environments
- Core components:
  - Ceph OSDs (Object Storage Daemons): store data as objects
  - Metadata Servers (MDS): manage namespace and directory hierarchy
  - Ceph Monitors (MONs): maintain cluster state and consistency
  - Clients mount CephFS and access data via the network

Advantages of CephFS

- High scalability and fault tolerance
- No single point of failure
- Seamless integration with Ceph object and block storage
- Open-source and community supported

Use Cases of CephFS

- Scientific computing and HPC
- Backup and archival solutions
- Large-scale content repositories
- Hybrid cloud infrastructure

Flexible!

XFS

- Journaling file system designed for scalability
- Supports extremely large files and file systems
- Metadata journaling enables fast crash recovery
- Can be defragmented and extended while mounted
- Efficient extent-based allocation with delayed and preallocation
- Excellent I/O scalability using B-trees for metadata and data indexing
- Default configurations optimized for most workloads (e.g., Red Hat)

- mkfs.xfs auto-selects segment width and unit matching hardware
- inode64: recommended for multi-terabyte systems (not for legacy 32-bit NFS clients)
- logbsize: ideal for frequently modified filesystems or burst workloads
  - Default 32 KB, maximum 256 KB
- Manual tuning advised for RAID setups
- Delayed allocation improves performance and reduces fragmentation
- Preallocation avoids fragmentation for known-write workloads

Measure raw performance on FS

dd command

```
[fbazzocchi@login02 ~]$ dd if=/dev/zero  count=1
1+0 records in
1+0 records out
512 bytes copied, 4.7513e 05 s, 10.8 MB/s
```

/dev/zero is is a virtual file (whose contents are not stored on mass memory) with the characteristic of returning 0 when read

Predefined blocksize for operation I/O

```
[fbazzocchi@login02 1+0 records in
1+0 records out
512 bytes copied, 0.000156403 s, 3.3 MB/s
```

```
[fbazzocchi@login02 ~]$ dd if=/dev/zero of=~/big write count=1M
1048576+0 records in
1048576+0 records out
536870912 bytes (537 MB, 512 MiB) copied, 4.87525 s, 110 MB/s
```

Why such a difference between the runs? ·
Why copying unit of 512B ? `~]$ dd if=/dev/zero of=~/small write count=1`

Blocksize on FS

- 512 byte is a typical block size of the disk:
- It cannot read less than 512 bytes, if you want to read less, read 512 bytes and discard the rest.
- File System block size can be different

```
[fbazzocchi@login02 ~]$ stat  f small write
  File: "small write"
    ID: b92f1c82fffffffff Namelen: 255     Type: ceph
Block size: 4194304    Fundamental block size: 4194304
Blocks: Total: 1290859070 Free: 1034637851 Available: 1034637851
Inodes: Total: 463198973  Free:  1
```

Predefined allocated blocksize on disk

```
[fbazzocchi@login02 fbazzocchi]$ ls  lh small write
 rw r  r   1 fbazzocchi fbazzocchi 512 Apr  5 20:48 small
write
```

```
[fbazzocchi@login02 fbazzocchi]$ stat NX_managing.ipynb
  File: NX_managing.ipynb
  Size: 3851            Blocks: 8          IO Block: 4194304 regular file
Device: 0,55    Inode: 1099633564111  Links: 1
Access: (0644/-rw-r--r--)  Uid: (10020014/fbazzocchi)   Gid:
(10020014/fbazzocchi)
Access: 2024-01-11 16:40:33.484910795 +0100
Modify: 2024-01-11 17:03:02.641409895 +0100
Change: 2024-01-11 17:03:57.265573121 +0100
 Birth: 2024-01-11 16:40:33.484910795 +0100
```

```
[fbazzocchi@login02 fbazzocchi]$ stat HDF5_ex/
  File: HDF5_ex/
  Size: 13              Blocks: 0          IO Block: 65536  directory
Device: 0,55    Inode: 1099686987573  Links: 9
Access: (0755/drwxr-xr-x)  Uid: (10020014/fbazzocchi)   Gid:
(10020014/fbazzocchi)
Access: 2024-05-20 21:53:56.686580241 +0200
Modify: 2024-09-20 10:40:39.325250064 +0200
Change: 2024-09-20 10:40:39.325250064 +0200
 Birth: 2024-05-20 21:53:56.686580241 +0200
```

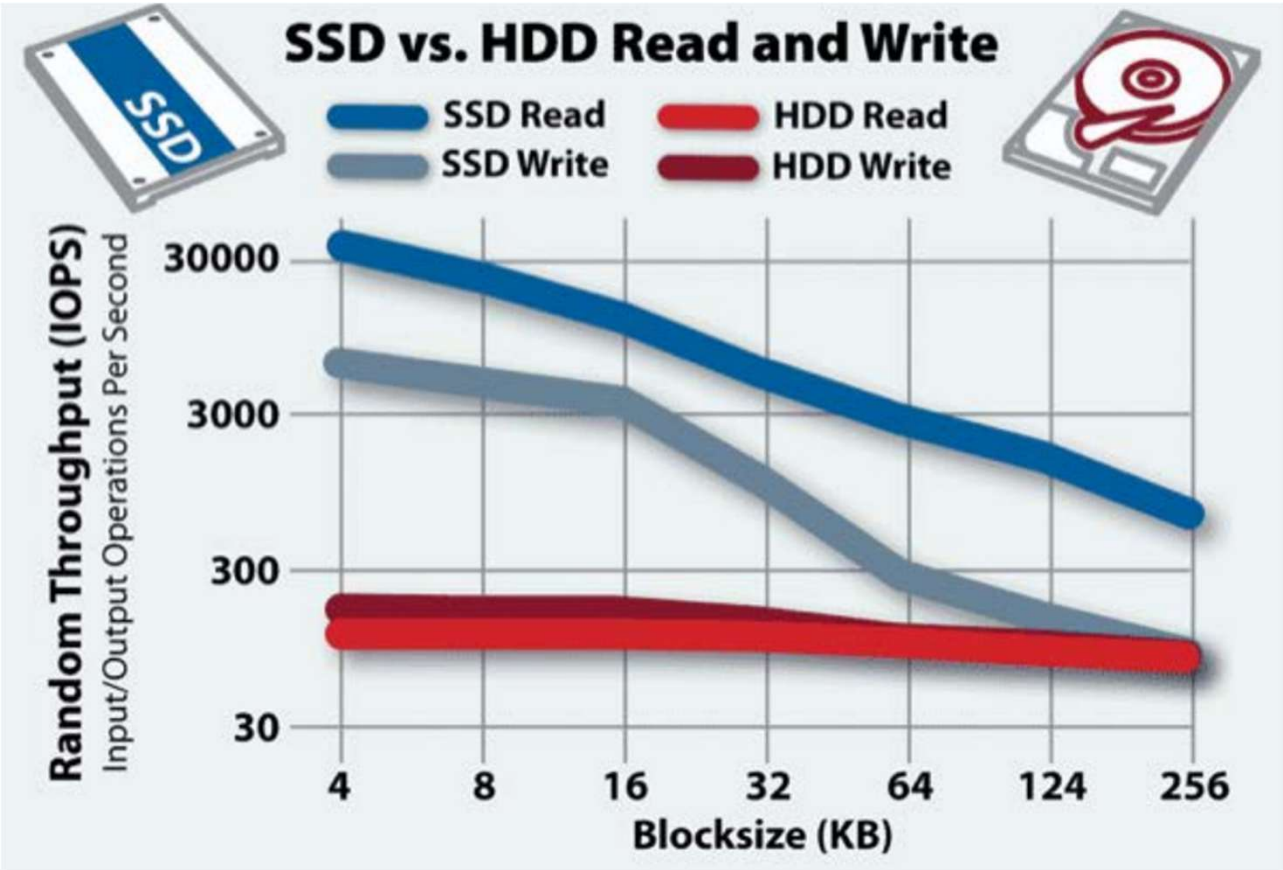
Stat <name> gives file/dir metadata

```
[fbazzocchilogin02 orfeo]$ ls
LTS  cephfs  fast  opt  scratch

[fbazzocchi@login02 orfeo]$ stat  f LTS
  File: "LTS"
    ID: fd0000000000 Namelen: 255     Type: xfs
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 3915776    Free: 1820163    Available: 1820163
Inodes: Total: 7864320    Free: 7656589
[fbazzocchi@login02 orfeo]$ stat  f cephfs/
  File: "cephfs/"
    ID: b92f1c82ffffffff Namelen: 255     Type: ceph
Block size: 4194304    Fundamental block size: 4194304
Blocks: Total: 1290859070 Free: 1033832946 Available: 1033832946
Inodes: Total: 463971472  Free:  1@
```

Different file system, different choices of block size

DISK perfomance is function of the blocksize!



ls

- Meta-data : Data to describe data attribute (and extended attribute)
  - size, owner, creation date
- Meta-data are the bottleneck of scalability ·
  - How many times do you type ls in a day?
  - (does good data management help?)
- ls means a scanning of all the files in the directory !
  - ls + filter/options

Proposed exercize

- Identify your FileSystem (df and mount in linux)
- Explore Linux file systems
  - List partitions: lsblk
  - Disk usage: df -h
  - Inode stats: df -i
  - Metadata: stat file
- Measure/Estimate the rough performance of your hard drive
- Compare it with the ramfs on your linux box and on your cluster system

Object Storage System vs FS

- At the physical level, both store bits on HDDs/SSDs using similar block-level operations.
- The difference lies in how the OS manages the mapping between the user's request and the physical sectors.

File System (organized warehouse)

- **Structure:** Hierarchical (Tree-based).
- **Access:** POSIX-compliant (Open, Read, Write, Close).
- **Limit:** Highly dependent on the OS kernel; managing millions of nested folders creates massive metadata overhead.

Object Storage (The Warehouse):

- **Structure:** Flat Namespace (No folders, just "Buckets").
- **Access:** RESTful APIs (HTTP GET, PUT, POST, DELETE).
- **Philosophy:** Data is treated as an "Object" containing the Data + Metadata + Global ID.

Scalability

- For the FS Inode is the Bottleneck
  - To find `/data/logs/2024/april/file.log`, the system must traverse every directory level.
  - Each step requires a metadata lookup, creating a serial bottleneck.
  - Expanding a File System usually requires "scaling up" (bigger controllers).
- The Global ID Advantage (Object Storage):
  - Key-Value Logic: The ID (Key) is hashed to point directly to the object's location.
  - There is no "tree" to climb. Whether you have 1,000 or 1,000,000,000,000 objects, the lookup time remains nearly constant.
- Decoupled Metadata:
  - In Object Storage, metadata is stored *with* the object, not in a central bottlenecked table (like an Inode table).
  - This allows for Horizontal Scalability: you can add thousands of cheap storage nodes, and the system stays fast.
- Eventually Consistent: Unlike File Systems that require instant locking (slowing down writes), Object Storage often uses "Eventual Consistency," allowing for massive global distribution.

Scalability

- **The difference:** FS performance is a function of the structure (how many folders/files) while Object Storage performance is a function of the an slgorithm (calculation, hashing)
- **Decentralized Metadata:** Using consistent hashing, the system maps IDs to specific storage nodes without a central "master" bottleneck.
- **Constant Latency:** The time required to compute a hash is independent of the total number of objects in the bucket.
- **Horizontal Scaling:** As the workload grows, we add nodes. The "Map" expands mathematically, maintaining stable performance across Exabytes of data.

In a File System, as N(number of files) grows, the search time increases as O(log(N). In Object Storage there is no serach but computation! Since the complexity of the math is independent of the number of items the performance remains stable and flat O(1).

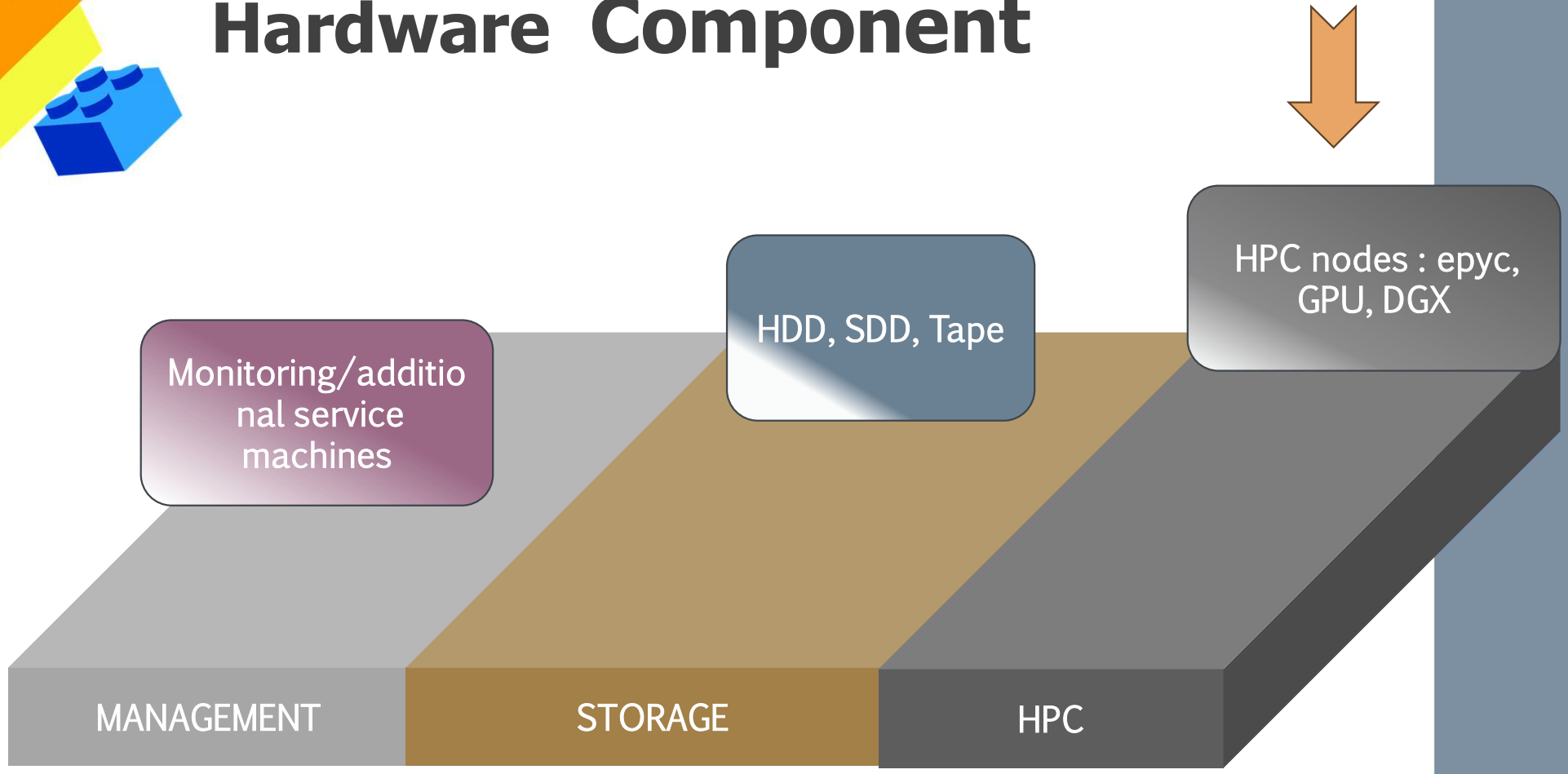
Exercise

https://colab.research.google.com/drive/1DO0RHtDAHtsTMTcZ56KPUJG6OO3vSQoo?usp=sharing

Reflect on

- Which strategy would you use according data temperature?
- Which partitioning strategy would you use if you have a dataset with 10 years of data and you often queried the last 6 months?
- Which data governance strategy would you implement?
- If your storage woud be Amazon S3 (where you pay for every GB scanned), which format would you choose?

Hardware Component



HARDWARE COMPONENTS: PROCESSING UNITS

PROCESSORS AND CORES

- Fundamental components of high-performing-computing (HPC) system
- Central processing unit (CPU) primary units executing instructions (the ‘brain” for general instructions)
- Choice of the processor can have significant role for the performance of specific applications
- Cores are the individual processing units within a processor
- Modern CPU are multiple cores (parallel processing)
- A thread is the single sequence of programmed instructions
- Modern processors use multiple threads per core
- CPU core switch between threads to improve utilization
- Multiple threads share core’s resources

Number crunching on CPU: what do we count ?

- Rate of [million/billions of] floating point operations per second ([M|G]flops) FLOPs/S
- Theoretical peak performance:
  - determined by counting the number of floating-point additions and multiplications that can be completed during a period of time, usually the cycle time of the machine

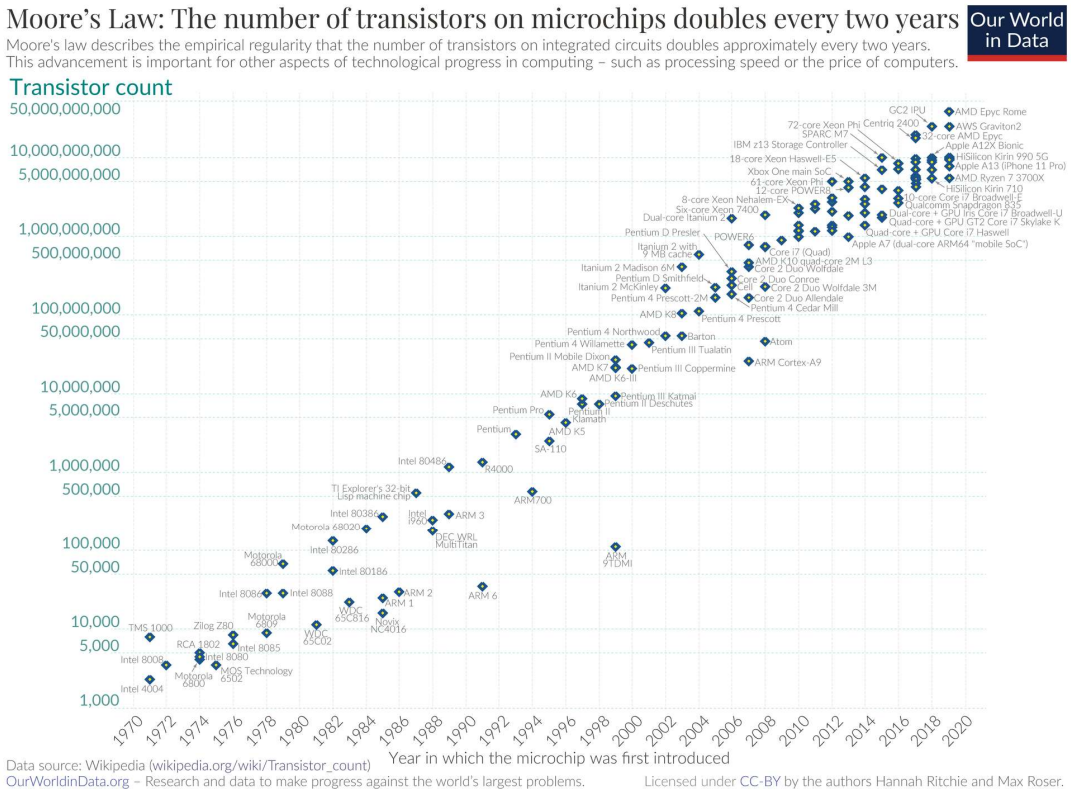
FLOPS=clock_rate*Number_of_FP_operation*Number_of_cores

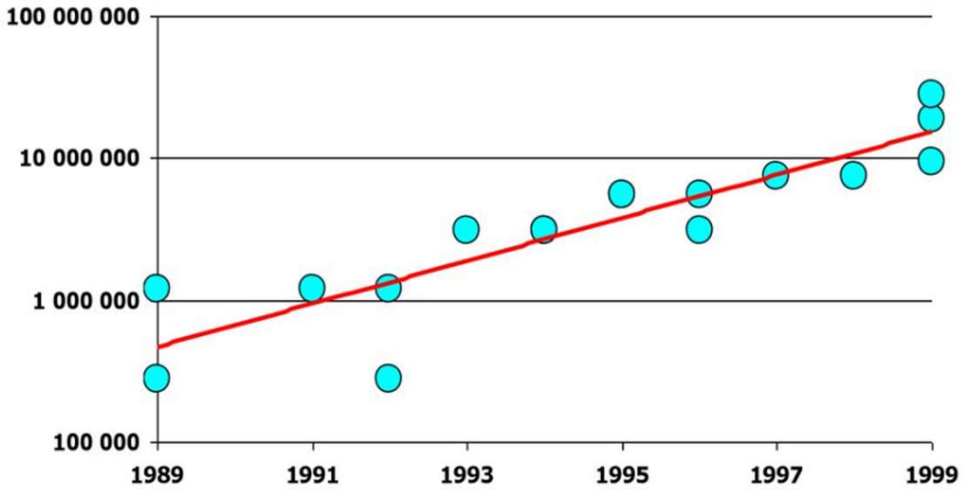
CPU	Year	Frequence (MHz)	Register/bus dim	Transistors' Num
8086	1978	4.77 — 12	8 / 16	29 000
80286	1982	8 — 16	16 / 16	134 000
80386	1986	16 — 33	32 / 32	275 000
80386 SX	1988	16 — 33	32 / 16	275 000
80486	1989	33 — 50	32 / 32	1 200 000
Pentium	1993	60 — 200	32 / 64	3 100 000
Pentium II	1997	233 — 400	32 / 64	7 500 000
Pentium III	1999	450 — 1133	32 / 64	24 000 000
Pentium 4	2000	1600 — 2000	32 / 64	42 000 000

Slides prof. A.Bonifati

Moore Law (1965)

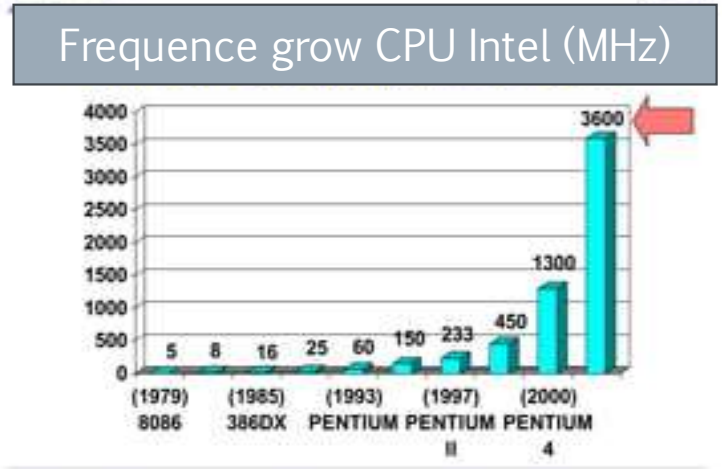
Moore's law is the observation that the number of transistors in a integrated circuit (IC) doubles about every 2 years (now is more roughly 3 years according to P.Gelsinger Intel CEOin 2023 but he declared that)





Power= Capacity x Voltage^2 x frequency

Heat dissipation!

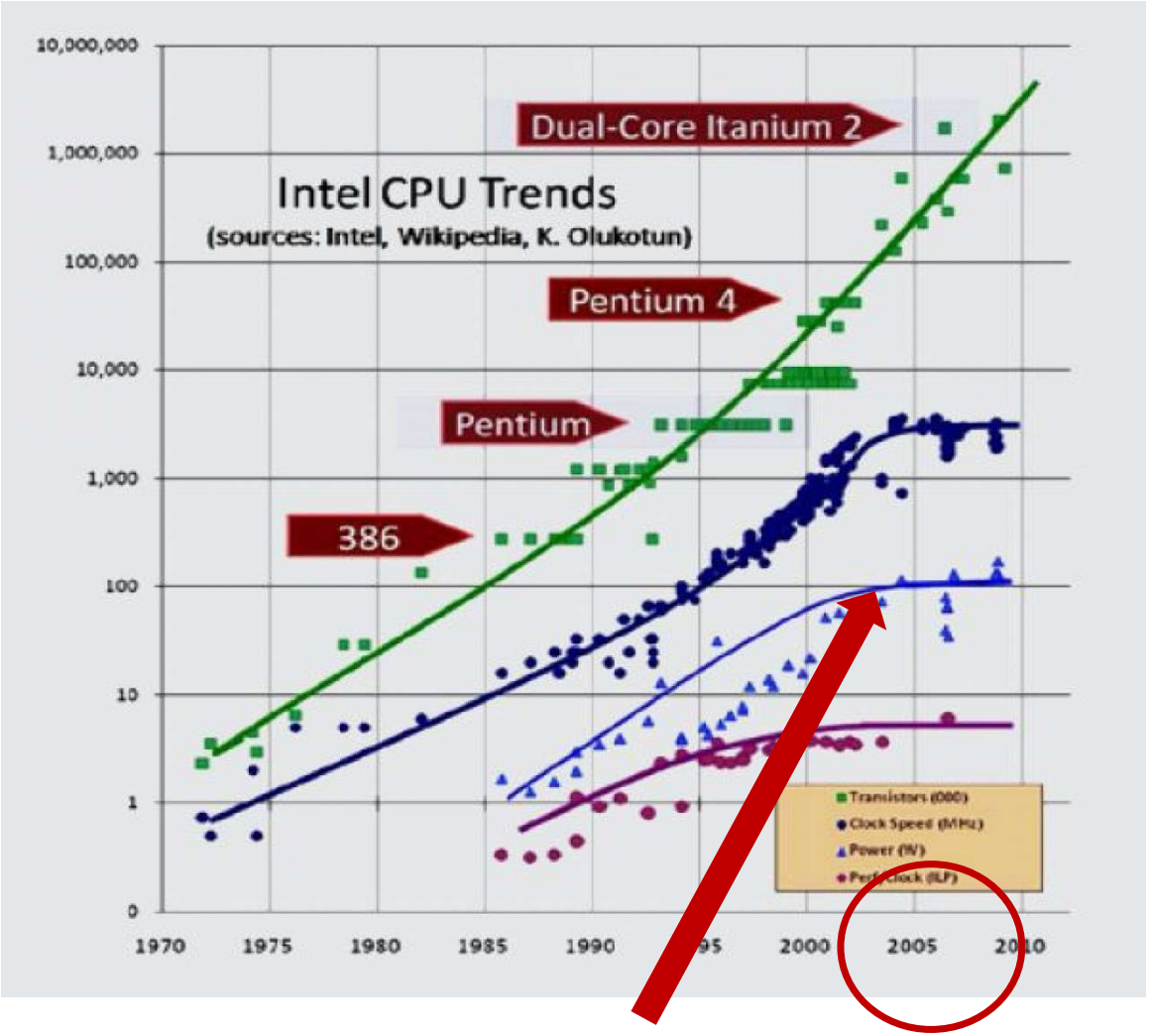


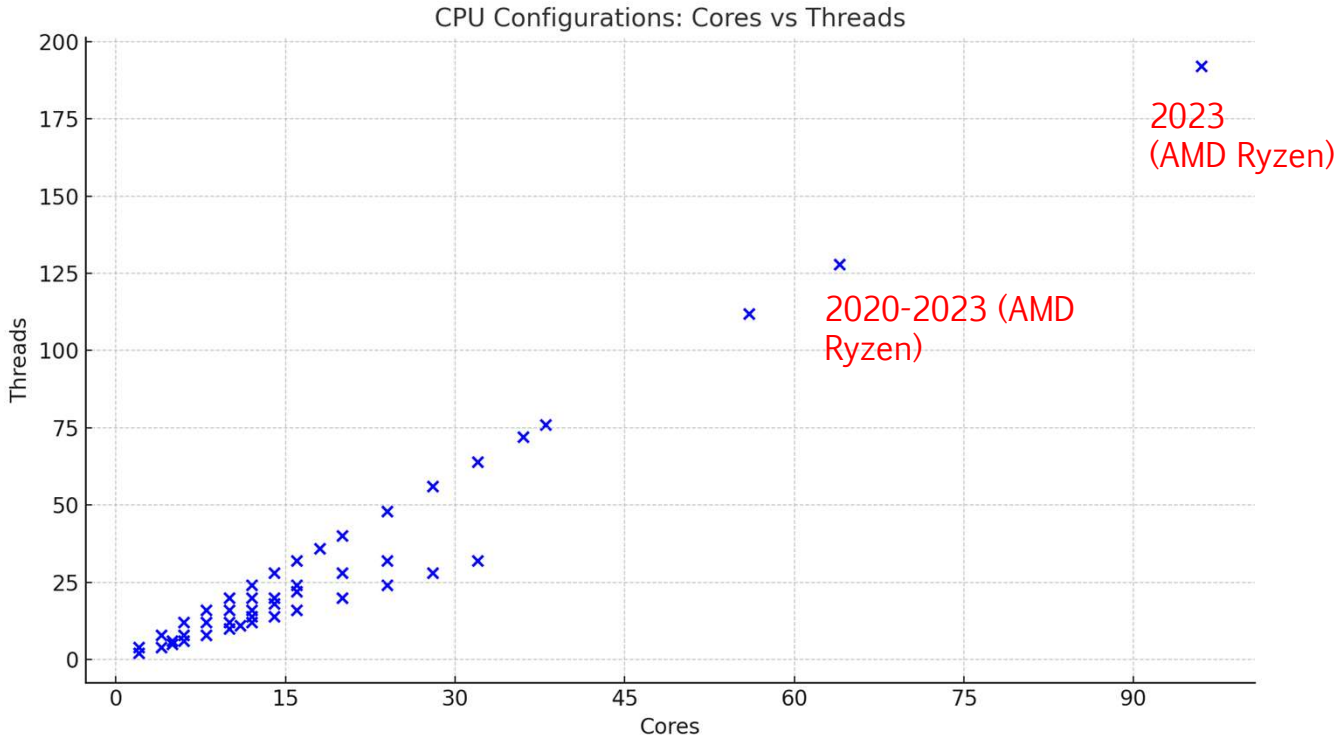
Slides prof. A.Bonifati

Intel Has Double Vision: First Multi-Core Silicon Production Begins Dual-core System Introduction in Second Quarter 2005

SANTA CLARA, Calif., **Feb. 7, 2005** – Intel Corporation today announced it has completed initial production runs of dual-core processors and provided further details about its multi-core plans to its customers, signaling the beginning of an era when PCs will have two or more “brains” inside.

Intel News Release



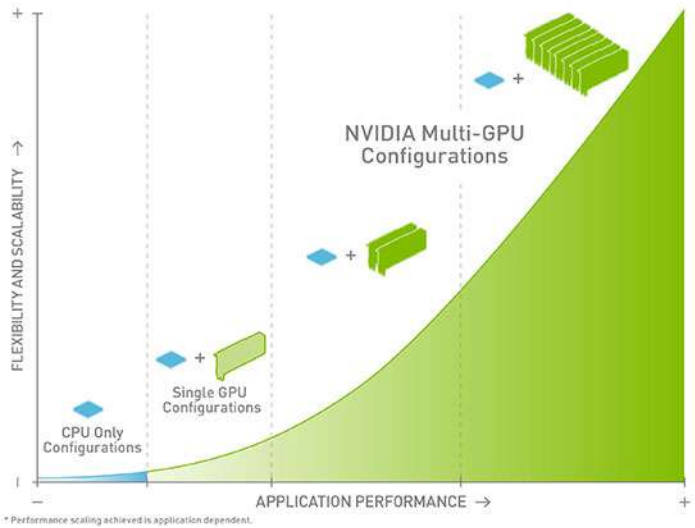


8(8): 2019 (Intel Core i7) 2020-2024 (Apple M1-M4)

EatYourBytes

GPU – Graphics Processing Unit

- Specialized processor designed for parallel data processing
  - Originally for rendering graphics, now widely used for compute tasks
  - Thousands of cores optimized for high-throughput operations
  - Ideal for AI/ML training, scientific computing, and 3D rendering
  - Examples: NVIDIA CUDA, AMD ROCm



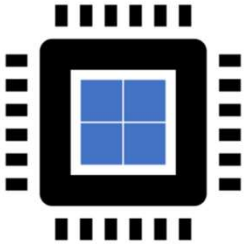
2012

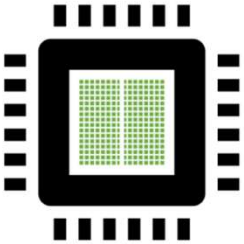
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

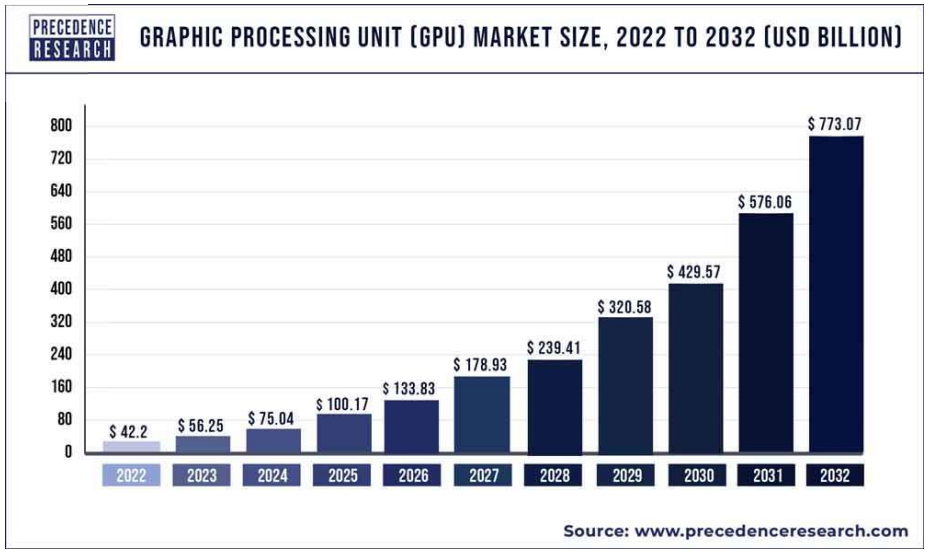
Abstract

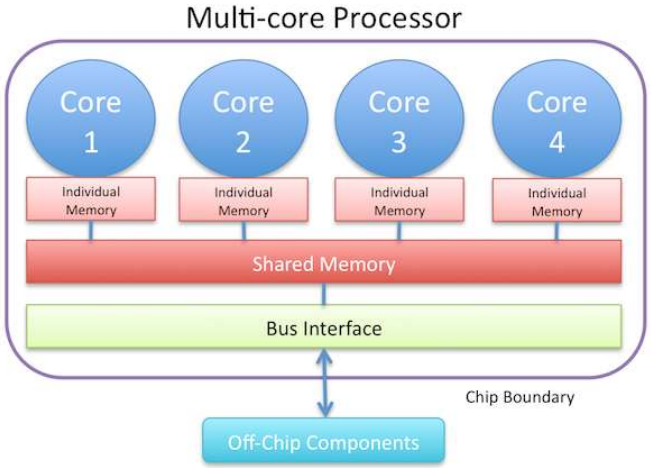
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

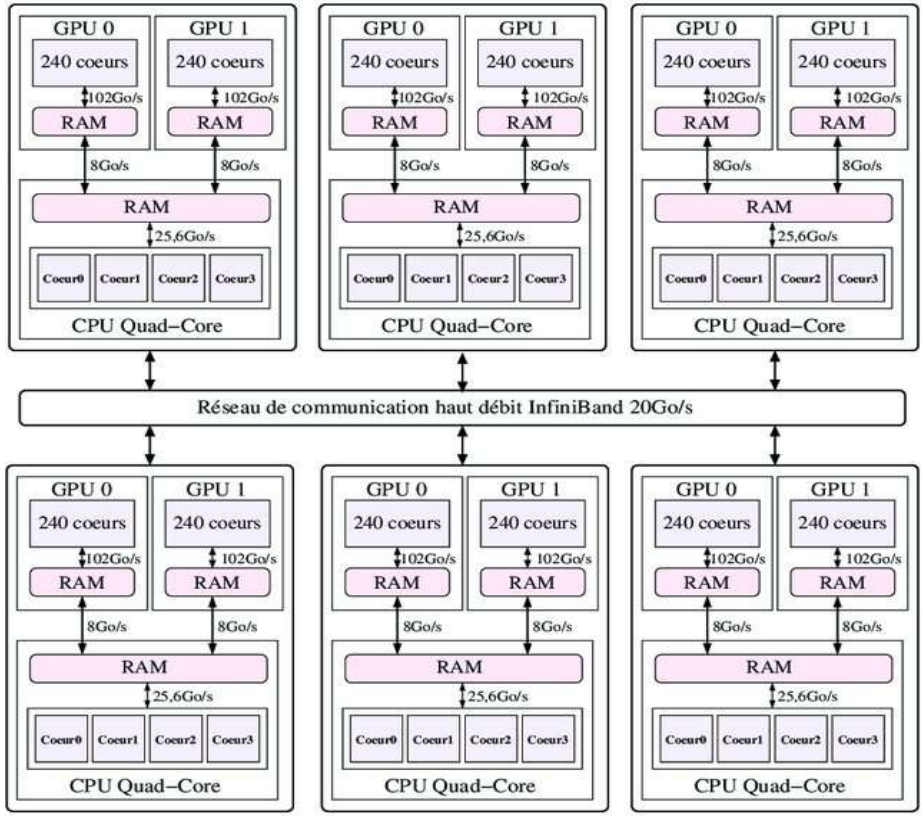




CPU	GPU
Central Processing Unit	Graphics Processing Unit
4-8 Cores	100s or 1000s of Cores
Low Latency	High Throughput
Good for Serial Processing	Good for Parallel Processing
Quickly Process Tasks That Require Interactivity	Breaks Jobs Into Separate Tasks To Process Simultaneously
Traditional Programming Are Written For CPU Sequential Execution	Requires Additional Software To Convert CPU Functions to GPU Functions for Parallel Execution




Multi-core Processor
Core 1
Core 2
Core 3
Core 4
Individual Memory
Individual Memory
Individual Memory
Individual Memory
Shared Memory
Bus Interface
Chip Boundary
Off-Chip Components


GPU 0
GPU 1
240 coeurs
102Go/s
RAM
8Go/s
25,6Go/s
Coeur0
Coeur1
Coeur2
Coeur3
CPU Quad–Core
Réseau de communication haut débit InfiniBand 20Go/s

TOP 500 List

- The TOP500 list www.top500.org
- published twice a year from 1993
  - ISC conference in Europe (June)
  - Supercomputing conference in USA (November)
- List the most powerful computers in the world
- yardstick: Linpack benchmark (HPL)

HPL: some details

- From http://icl.cs.utk.edu/hpl/index.html:
  - The code solves a uniformly random system of linear equations and reports time and floating-point execution rate using a standard formula for operation count.
  - Number_of_floating_point_operations = $2/3n^3 + 2n^2$ (n=size of the system)

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR03R2L2       86000  1024     2     1             191.06              2.219e+03
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0043644 ...... PASSED
```

For each machine the following numbers are reported using HPL:

Rmax: the performance in GFLOPS for the largest problem run on a machine.

Rpeak: the theoretical peak performance GFLOPS for the machine.

The measure of the power required to run the benchmark

And the winner is.. (both 2024 and 2025)

EL CAPITAN - HPE CRAY EX255A, AMD 4TH GEN EPYC 24C 1.8GHZ, AMD INSTINCT MI300A, SLINGSHOT-11, TOSS

Site:	DOE/NNSA/LLNL
Manufacturer:	HPE
Cores:	11,039,616
Processor:	AMD 4th Gen EPYC 24C 1.8GHz
Interconnect:	Slingshot-11
Installation Year:	2024

11/2025

Performance	
Linpack Performance (Rmax)	1,809.00 PFlop/s
Theoretical Peak (Rpeak)	2,821.10 PFlop/s
Nmax	24,768,000
HPCG	17,406.0 TFlop/s
Power Consumption	
Power:	29,684.62 kW

11/2024

Performance	
Linpack Performance (Rmax)	1,742.00 PFlop/s
Theoretical Peak (Rpeak)	2,746.38 PFlop/s
Nmax	25,446,528
ower Consumption	
ower:	29,580.98 kW
ower Measurement Level:	2
oftware	
perating System:	TOSS
ompiler:	g++ 12.2.1 and hipcc 6.2.0
ath Library:	AMD rocBLAS 6.0.2 and Intel MKL 2016
PI:	HPE Cray MPI

Introduction to HPC, slides from S. Cozzini

Europe? (November 2024)

5	Eni S.p.A. Italy	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9 HPE	3,143,520	477.90	606.97	8,461
6	RIKEN Center for Computational Science Japan	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D Fujitsu	7,630,848	442.01	537.21	29,899
7	Swiss National Supercomputing Centre (CSCS) Switzerland	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS HPE	2,121,600	434.90	574.84	7,124
8	EuroHPC/CSC Finland	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11 HPE	2,752,704	379.70	531.51	7,107
9	EuroHPC/CINECA Italy	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband EVIDEN	1,824,768	241.20	306.31	7,494

Introduction to HPC, slides from S. Cozzini

Europe? (November 2025)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,340,000	1,809.00	2,821.10	29,685
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	1,000.00	1,226.28	15,794

Europe? (November 2025)

6	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
7	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
8	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124
9	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
10	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494

TOP 500 Green List

NOVEMBER 2025

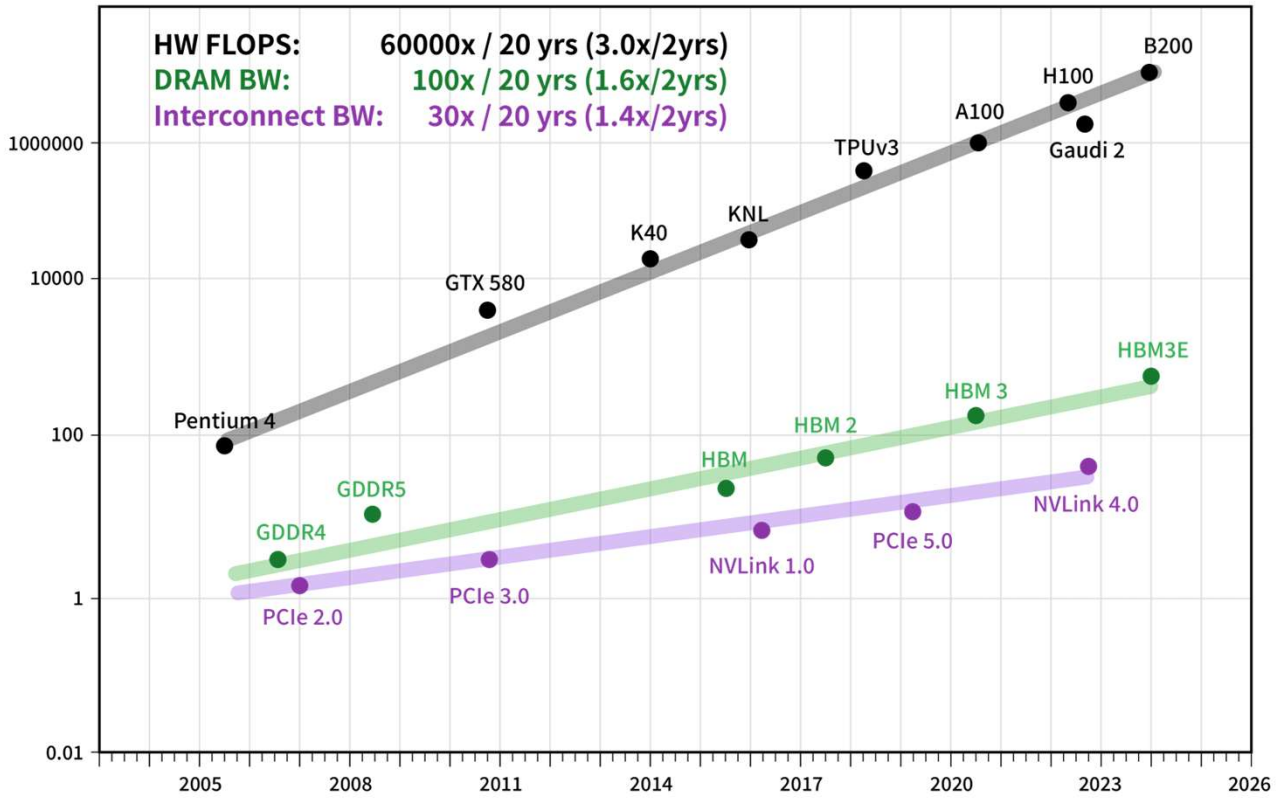
In the Green500 the systems of the TOP500 are ranked by how much computational performance they deliver on the HPL benchmark per Watt of electrical power consumed. This electrical power efficiency is measured in Gigaflops/Watt. This ranking is not driven by the size of a system but by its technology and the ranking order looks therefor very different from the TOP500. The computational efficiency of a system tends to slightly decrease with system size, which among technologically identical system gives smaller system the advantage.

Europe!!!

Rank	Rank	System	Cores	(PFlop/s)	(kW)	(GFlops/watts)
1	420	**KAIROS** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN CALMIP / University of Toulouse - CNRS France	13,056	3.05	46	73.282
2	171	**ROMEO-2025** - BullSequana XH3000, Grace Hopper Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, Red Hat Enterprise Linux, EVIDEN ROMEO HPC Center - Champagne-Ardenne France	47,328	9.86	160	70.912
3	225	**Levante GPU extension** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN DKRZ - Deutsches Klimarechenzentrum Germany	35,904	6.75	110	69.426
4	213	**Isambard-AI phase 1** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE University of Bristol United Kingdom	34,272	7.42	117	68.835

First 6 posiition!s!

Scaling of Peak Hardware FLOPS, and Memory/Interconnect Bandwidth.

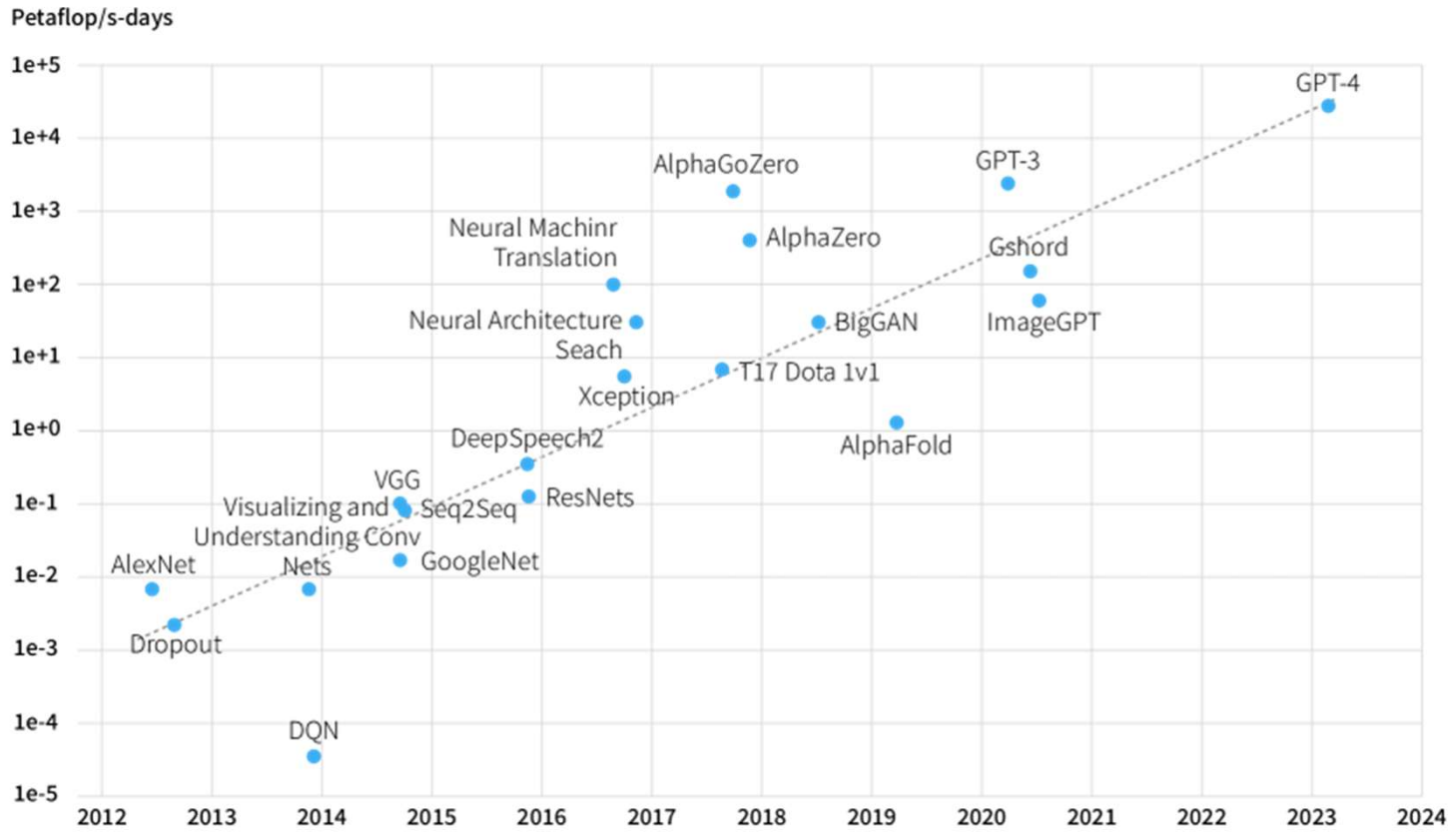


BUT

Huang's law observation that advancements in graphics processing units (GPUs) are growing at a rate much faster than with traditional central processing units (CPUs).

Introduction to HPC, slides from S. Cozzini

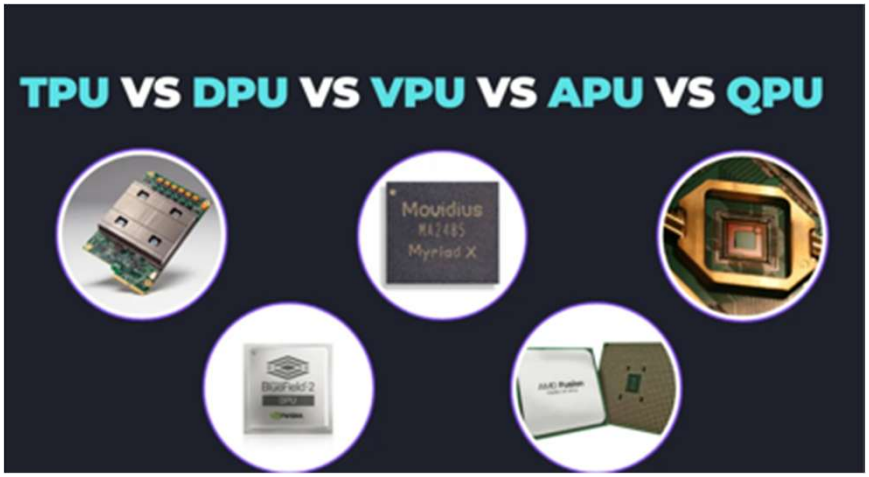
HPC needs for AI



Taken from https://arxiv.org/pdf/2408.14158v2

Specialized AI Hardware

- Traditional CPUs and GPUs are no longer sufficient for AI's demands (ask for energy efficiency and sustainability)
- Specialized processors accelerate machine learning and data intensive tasks.



TPUs – Tensor Processing Units

- Custom built by Google for neural networks
- Use systolic array architecture for fast matrix ops
- Power TensorFlow, Google Search, Photos
- Up to 30x faster than CPUs/GPUs for AI
- Energy efficient, scalable via TPU pods

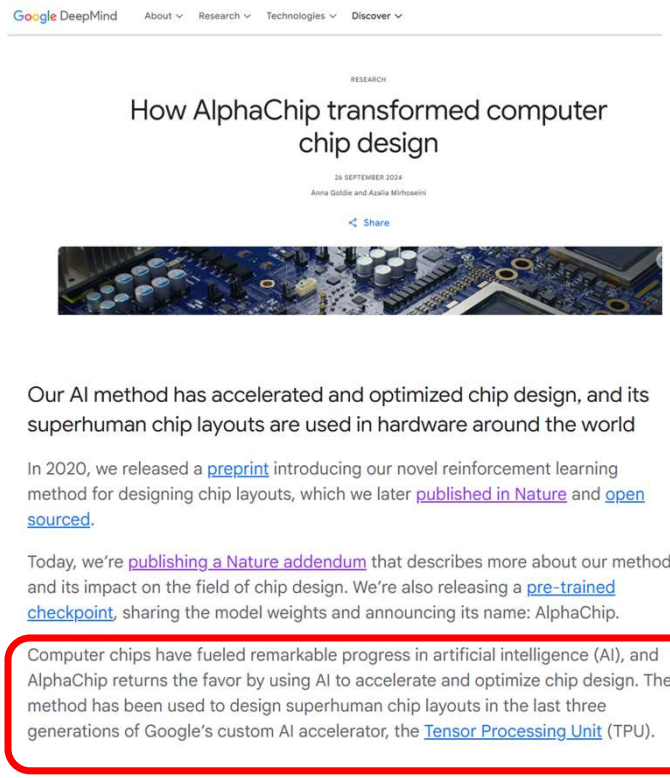
DPUs – Data Processing Units

- Offload networking, storage, and security tasks
- Act as smart NICs in data centers
- Intel IPUs, NVIDIA BlueField accelerate cloud ops
- Reduce CPU load, boost performance & security

Silicon Volley: Designers Tap Generative AI for a Chip Assist

Semiconductor engineers show how a specialized industry can customize large language models to gain an edge using NVIDIA NeMo.

October 30, 2023 by Rick Merritt



NVIDIA Research has hundreds of scientists and engineers worldwide focused on topics such as AI, computer graphics, computer vision, self-driving cars and robotics. Other recent projects in semiconductors include using AI to design smaller, faster circuits and to optimize placement of large blocks.

VPUs – Vision Processing Units

- Designed for computer vision & image processing
- Found in drones, cameras, and smart devices
- Intel Movidius enables real time object detection
- Ideal for edge AI, low power vision tasks

APUs – Accelerated Processing Units

- AMD’s combo of CPU + GPU on a single chip
- Shared memory boosts efficiency
- Great for budget gaming, media centers, entry level AI
- Energy efficient and compact

QPUs – Quantum Processing Units

- Use quantum bits (qubits) for computation
- Leverage superposition & entanglement
- Solve complex problems faster than classical computers
- Potential for revolutionizing AI, optimization, cryptography

At IBM Quantum Summit 2023, IBM Quantum System Two was debuted as the company's first modular quantum computer and cornerstone of IBM's quantum-centric supercomputing architecture. (Credit: Ryan Lavine for IBM)

NEW YORK, Dec. 4, 2023 /PRNewswire/ -- Today, at the annual IBM Quantum Summit in New York, IBM (NYSE: IBM) debuted 'IBM Quantum Heron,' the first in a new series of utility-scale quantum processors with an architecture engineered over the past four years to deliver IBM's highest performance metrics and lowest error rates of any IBM Quantum processor to date.

IBM also unveiled IBM Quantum System Two, the company's first modular quantum computer and cornerstone of IBM's quantum-centric supercomputing architecture. The first IBM Quantum System Two, located in Yorktown Heights, New York, has begun operations with three IBM Heron processors and supporting control electronics.

NETWORKING AND COMMUNICATION

WHY THEY MATTER

- Enable communication between distributed components, internet and other neywork
- Support data sharing, replication, and remote access
- Affect performance, reliability, and scalability of infrastructure
- Critical metrics:
    - Latency: time to begin data transfer
    - Bandwidth: amount of data per unit time

IP Addressing

- A system that assign unique identifiers, IP addresses, to devices connected to a network.
- These serve as primary means for devices to locate and communicate with each other

Protocols

- Standardized rules and conventions that governs how data is trasmitted and received over a network
- Transmission Control Protocol (TCP): ensures reliable, ordered and error-checked delivery of data
- Internet Protocol (IP): routing of data across networks
- HyperText Transfer Protocol (Secure) (HTTP/HTTPS):
  - Used for REST APIs, web-based data services
  - Common in cloud-based and microservices architectures
- File Transfer Protocol (FTP) simple but insecure
- Secure File Transfer Protocol (SFTP): encrypted, uses SSH
  - Used for batch transfers, data ingestion pipelines

Which FAIR principle is strongly related to?