Operating Systems – File systems part 2

Péter Györke

http://www.mit.bme.hu/~gyorke/

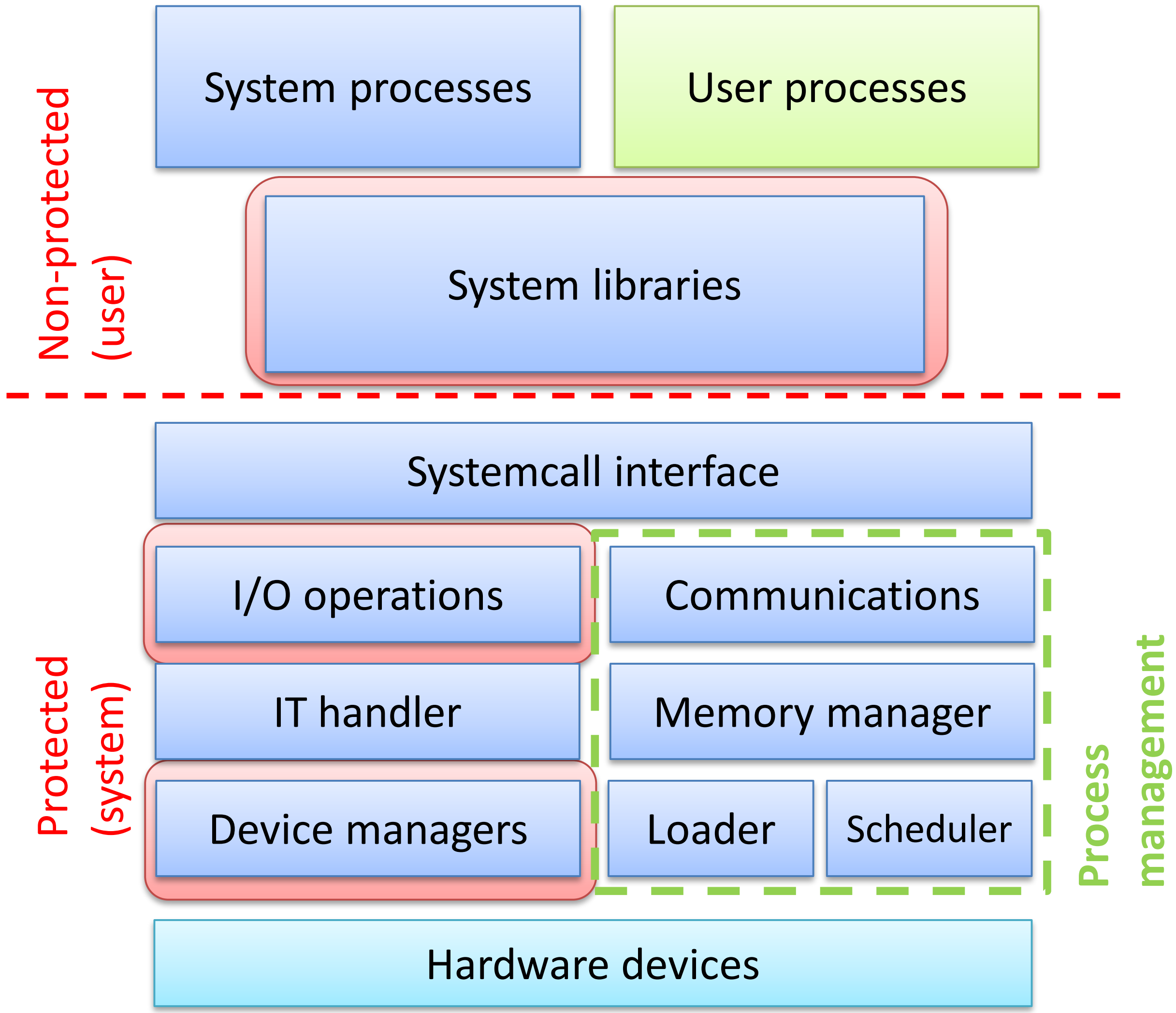
gyorke@mit.bme.hu

Budapest University of Technology and Economics (BME)

Department of Measurement and Information Systems (MIT)

The slides of the latest lecture will be on the course page. (https://www.mit.bme.hu/eng/oktatas/targyak/vimiab00)

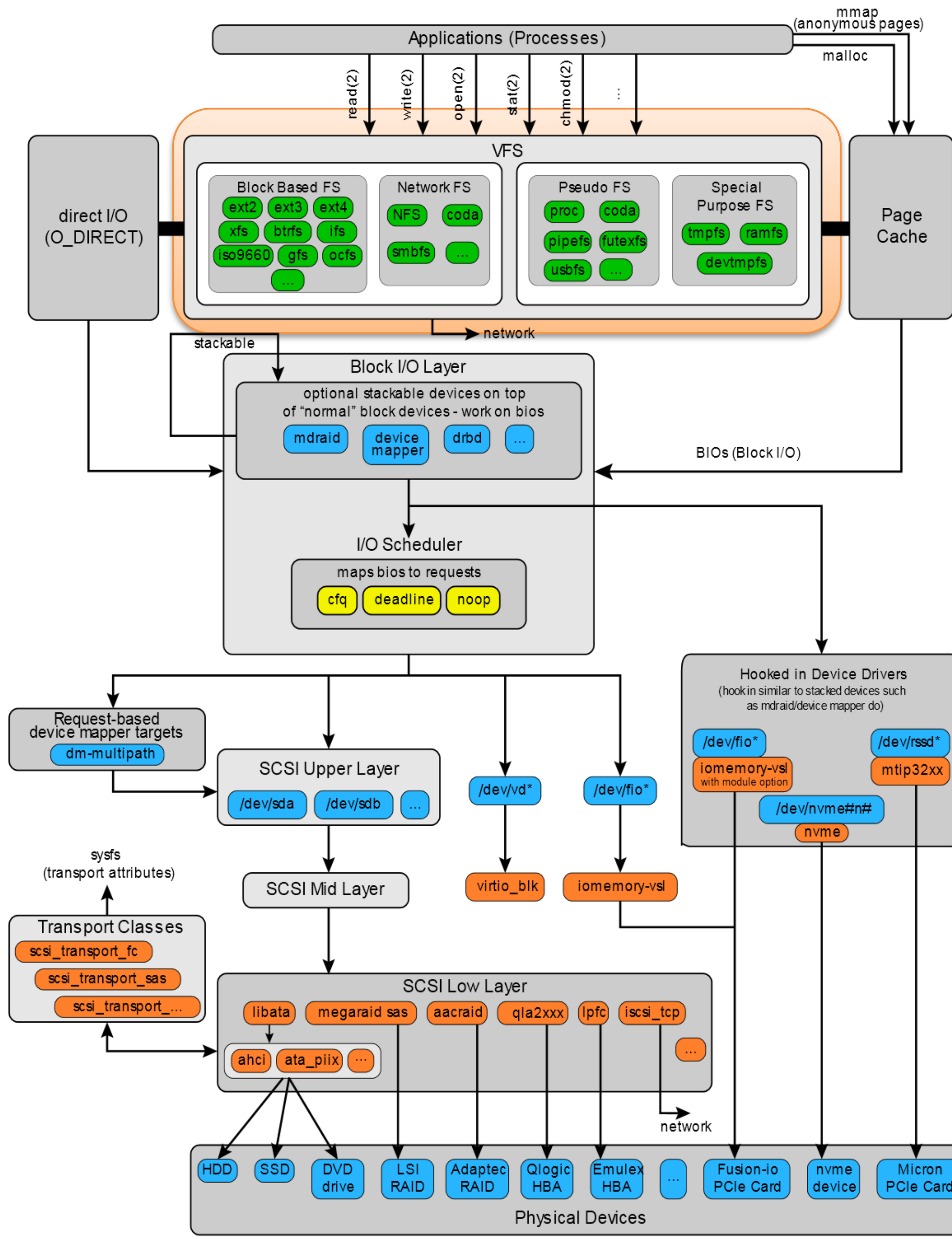
The main blocks of the OS and the kernel (recap)



Overview of the topic

Last lecture

- User interfaces
  - User
  - Administrator
  - Programmer
- File systems
  - Kernel data structures
  - File system interfaces
  - Data arranged in blocks on disks
  - Virtual file systems
- Storing the data
  - Physical storages (HDD, SSD)
  - I/O scheduling
  - Local storage system virtualization (RAID, LVM)
  - Network and distributed file systems



The Virtual File System (VFS)

- There are many types of file systems
  - Typically under UNIX systems, multiple types used at the same time
  - We can't except that the programmers manage them separately
- VFS is an implementation independent file system abstraction
  - The basis of the modern UNIX file systems
- Goals
  - Supporting multi type file systems running simultaneously
  - Standard programming interface (after mounting)
  - Provide the same interface also for special FS (e.g. network)
  - Modular structure
- Abstraction
  - fs (file system metadata) → vfs
  - inode (file metadata) → vnode

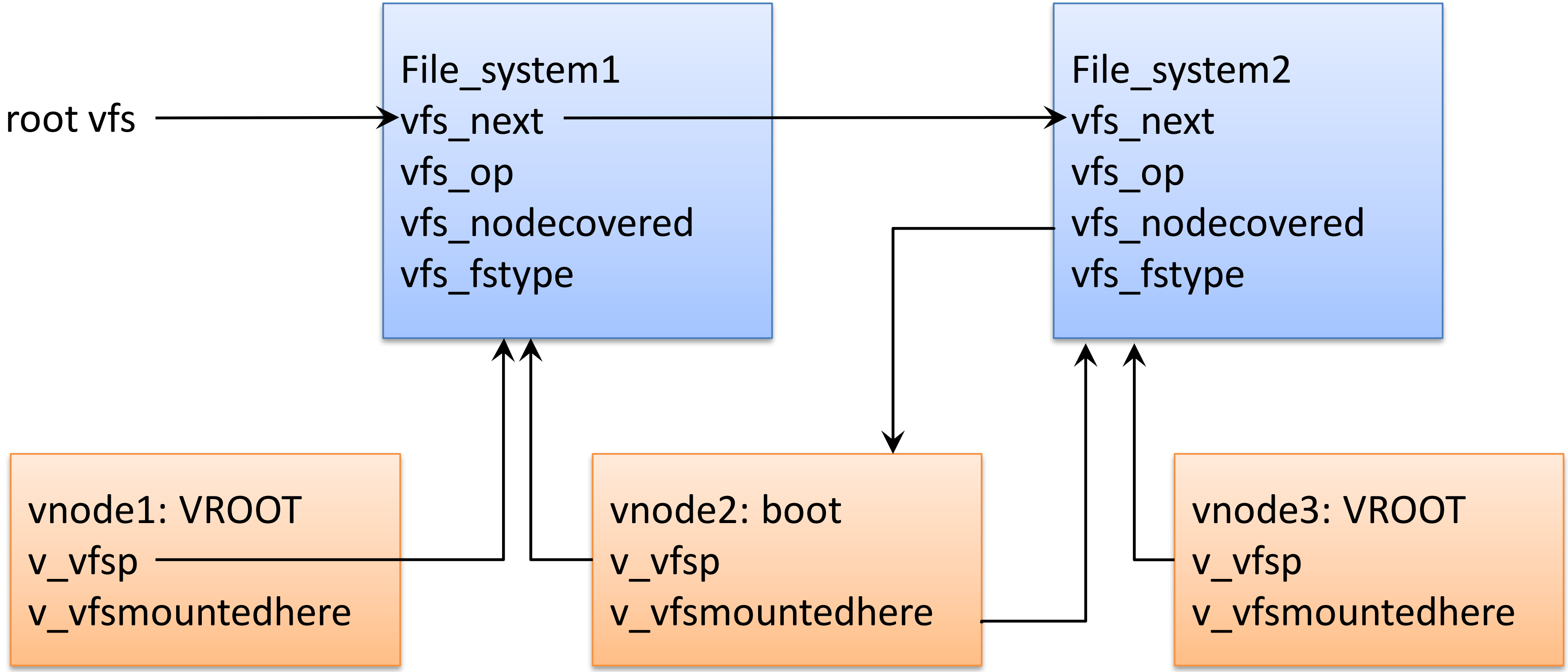
vnode and vfs

- vnode data fields
  - Common data (type, mounting, link counter)
  - `v_data`: file system dependent data (inode)
  - `v_op`: table of the file methods (operations)

- vfs data fields
  - Common data (FS type, mounting, `vfs_next`)
  - `vfs_data`: file system dependent data
  - `vfs_op`: table of the FS methods (operations)

- Virtual functions
  - vnode: `vop_open(), vop_read(), …`
  - vfs: `vfs_mount, vfs_umount, vfs_sync, …`
  - These are translated to the FS dependent methods

The connection between vfs and vnode



Special virtual file systems (examples)

- Which file systems are supported?

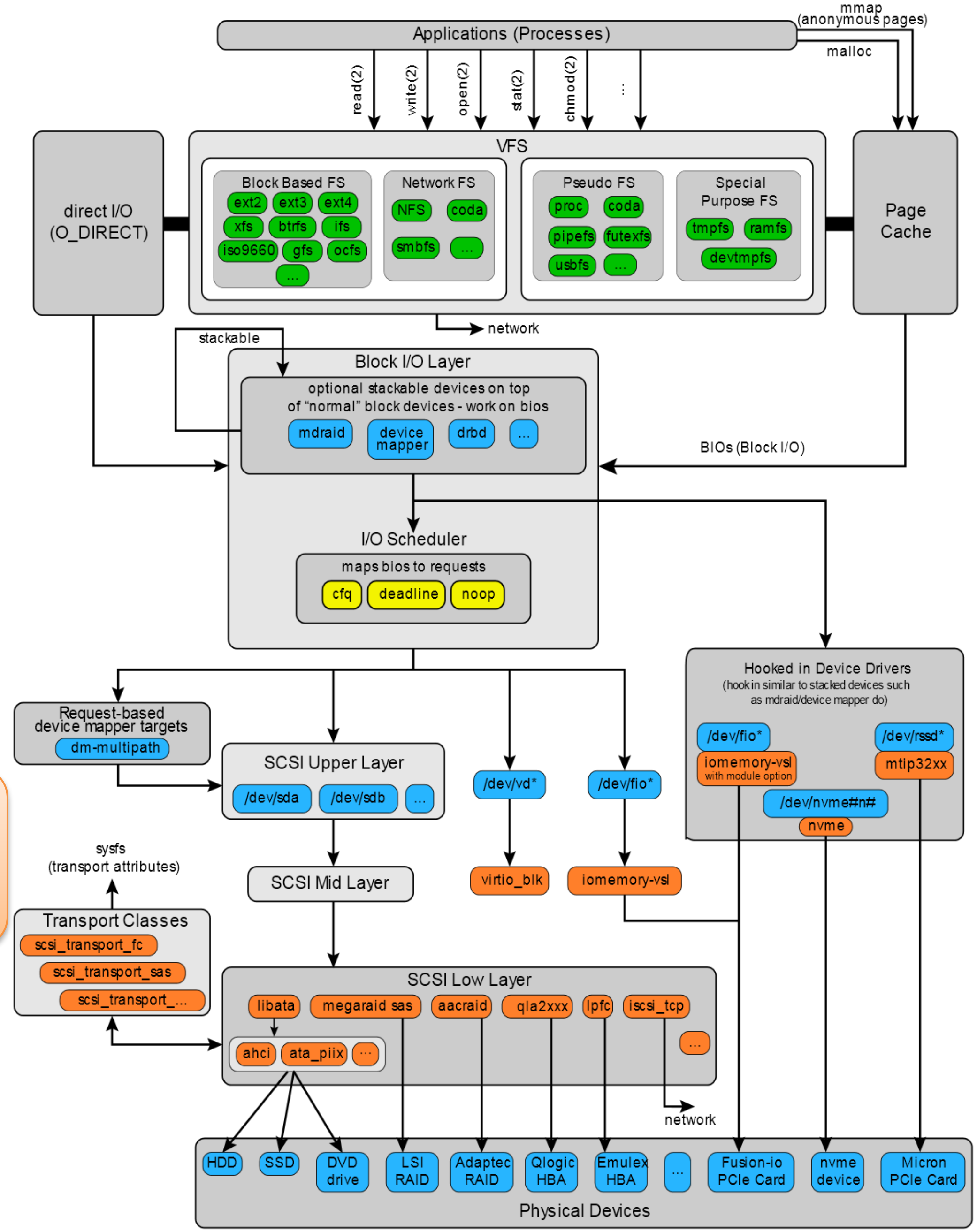
```
cat /proc/filesystems
```

- devtmpfs and devfs
  - accessing the HW devices trough the file system
- procfs
  - accessing to the process metadata and kernel structure through the FS
- sysfs
  - accessing to kernel subsystems through FS
- cgroup, cpuset
  - setting resource allocation for process groups

```
mount | egrep „cgroup|cpuset”
```

Overview of the topic

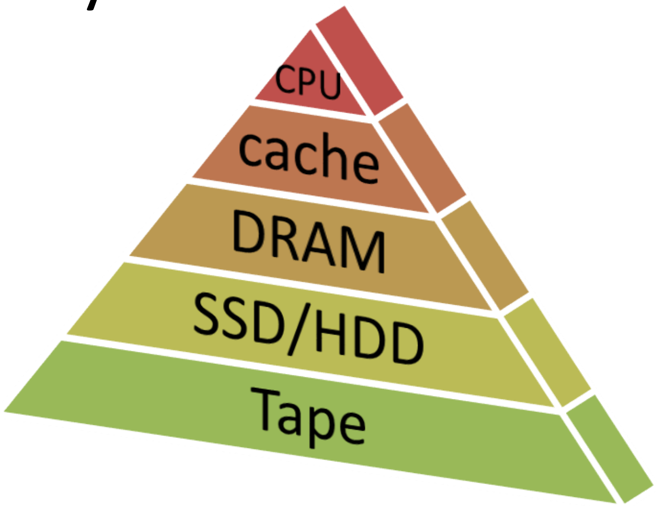
- User interfaces
  - User
  - Administrator
  - Programmer
- File systems
  - Kernel data structures
  - File system interfaces
  - Data arranged in blocks on disks
  - Virtual file systems
- Storing the data
  - Physical storages (HDD, SSD)
  - I/O scheduling
  - Local storage system virtualization (RAID, LVM)
  - Network and distributed file systems



Physical storage solutions behind file systems

- Physical storage devices
  - Magnetic
    - HDD and tape devices
  - Optical
    - CD, DVD, Blu-ray
  - Nonvolatile memories (solid state, integrated circuit based)
    - SSD, USB drive, SD card
- Virtual storage systems
  - extends the services of the physical storage systems with further layers
    - Merging devices
      - To increase storage size or reliability
      - e.g. RAID, LVM
    - Provides network interfaces
      - With file or block level transfer
      - e.g. NAS, SAN
    - Creating a distributed storage system
      - For reliable and scalable storage systems
      - e.g. Ceph, GlusterFS
  - In certain cases these are integrated with the FS
    - e.g. Solaris ZFS, Linux BTRFS, …

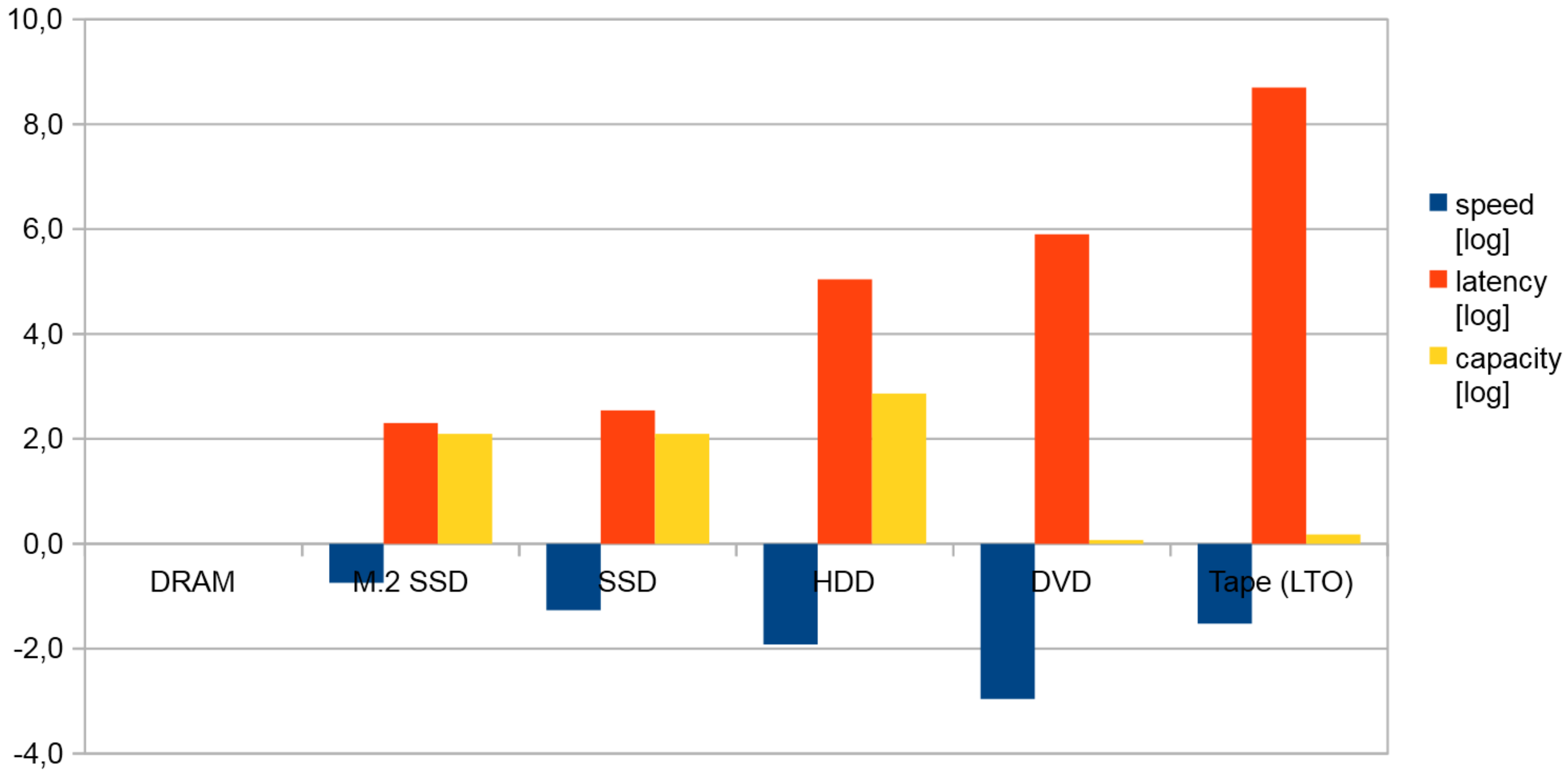
Properties of physical storage systems



- Performance
  - Capacity: 4 B → TB
  - Throughput (read/write)
    - 10 MiB/s → 200 GiB/s
  - Access time: 0.5 ns → 50 ns
- Reliability
  - measures related to the life-time of a device (see SMART)
  - **Annualized failure rate (AFR)**
    - How many devices fail within a year?
    - Typically 2-4%, but sometimes above 10%
  - **Mean time to failure (MTTF)**
    - Millions of operating hours (>100 years), according to vendors
    - It is related to all of the devices averaged, not for a single device
    - Bathtub curve: higher failure chance for old and new devices
    - Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?
  - **Total bytes written** (**TBW**, for memory based devices)
    - The memory pages cannot written infinite times
    - The amount of bytes written, which won't cause a failure
    - It can be decades for a daily 50 GB amount of data (link)

Performance of storage devices

- Compared to DRAM
  - Log scale comparison of the speed, latency and capacity



Trends of storage systems

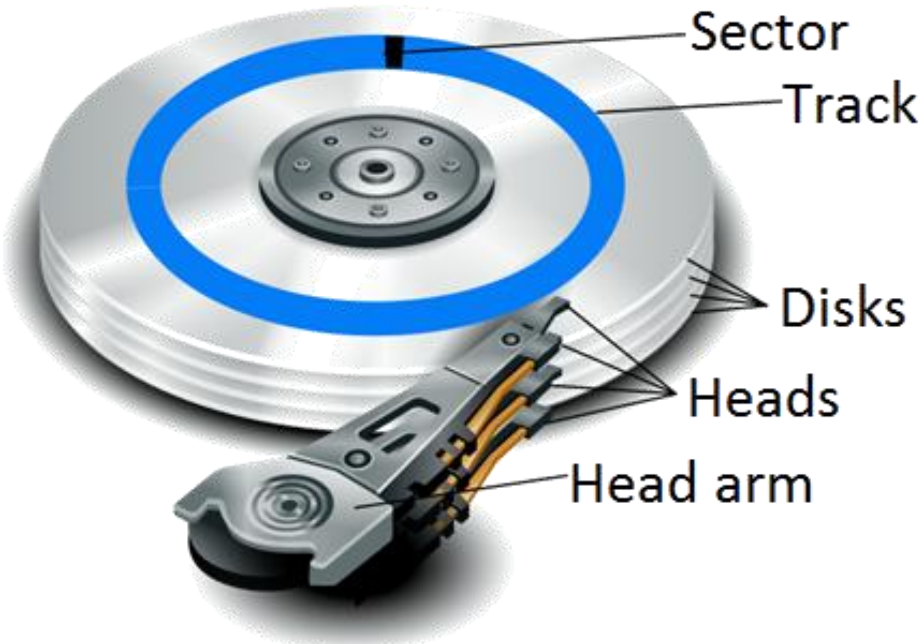
- In the past
  - Significant performance difference between CPU and disks
    - The CPU-s were developed faster than HDD-s
    - The slow I/O (relative) operations defined the principle of operation of the operating systems
- Recently
  - The size of the physical memory is highly increased
    - The size of disk cache is higher
  - There are methods based on fast CPU-s
    - runtime data compression (ZFS, btrfs)
    - Deduplication
      - A type of compression: avoiding the storage of the same data part more than one times
  - Spreading of memory based „disks"
    - Increasing speed and capacity, low latency
    - Storage class memory: Almost DRAM performance
- What's changing?
  - Memory management (faster swapping)
  - Scheduling (lower waiting times)

Tape drives

- Traditional tool for back-ups
  - High capacity
  - Long lifetime
  - slow operation, manual cassette change
- Recent developments
  - Sequential read speed is almost SSD fast
    - Tape – 300 MB/s, SSD – 500 MB/s
  - Can it replace the HDD?
    - Pro-s and con-s
  - Larger caches
    - Almost every data is there
    - Filled with sequential read
  - log-structured file systems
    - sequential read/write

Data allocation on disk drives

- The location of the superblock, inode list, data blocks on the disk
  - Goals: performance, reliability
- Cylinder block
  - Tracks assigned to the same head position
  - The data can be accessed without head movement
  - Collective damage is possible when a head-disk collision happens
- Allocation principles
  - The superblock is stored in every cylinder block
  - inode list and free blocks are in a separate c.block
  - Small files in the same c.block
  - Larger files are distributed between c.blocks
  - The new files will be on a less used c.block

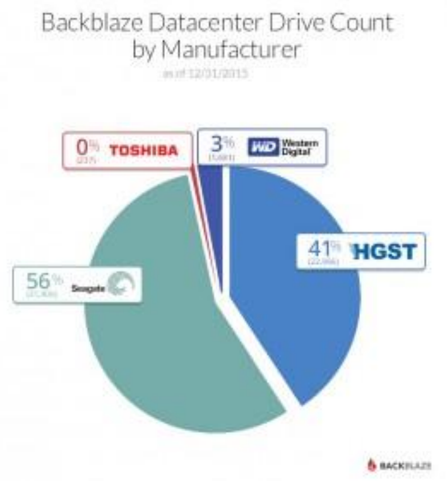


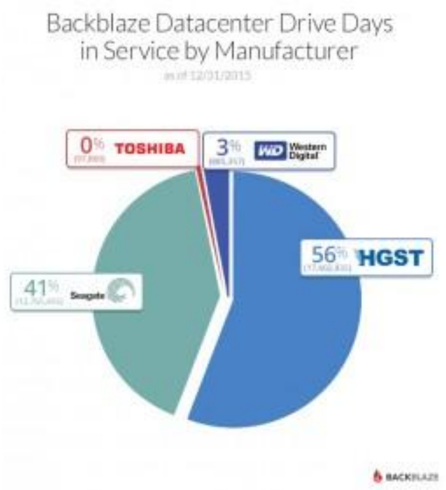
Scheduling of disk operations

- The kernel schedules the requests from the user layer towards the storage devices
- I/O schedulers is LINUX
  - **Noop**: simple FIFO scheduler
    - may concatenate adjacent requests
    - Small overhead
    - It is recommended if the storage system (RAID, NCQ, virtual systems,…) has an internal scheduling, or if scheduling is unnecessary (RAM disk)
    - Best solution for CPU intensive systems (low load on disks)
  - **Deadline**: tries to perform requests before a deadline
    - The requests are ordered by the block address in read and write batches
    - Recommended for I/O intensive systems with many parallel requests
  - **CFQ** (Completely Fair Queuing): equal service for every request
    - Request queues for every process, and a time-slice is assigned
    - With the `ionice` command the following states can be set: real-time, best effort, idle
    - A predictive estimation is also assigned to each queue, for estimating the further load
    - The scheduling is depends in priority and estimation of the queues, not the individual requests
    - Recommended for general usage (usually this is the default)

Reliability of hard disk drives

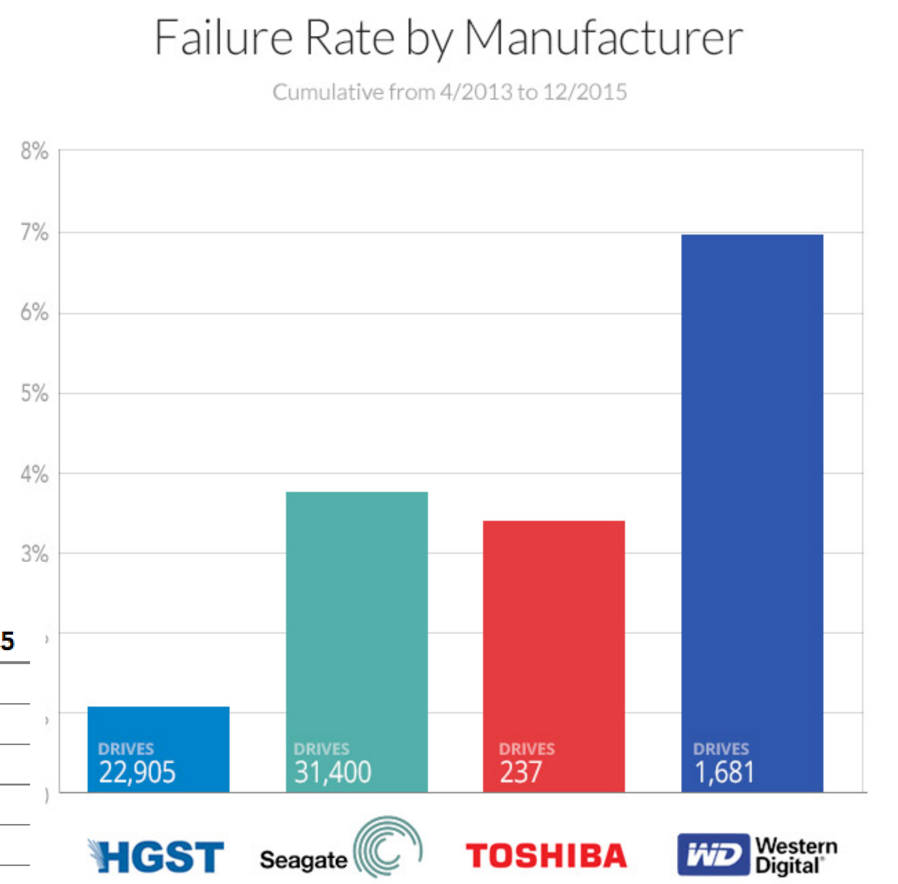
- Statistics for 56K disks of the Backblaze data center





Cumulative Failure Rate through the Period Ending

MFG	Model #	Highest QTY	12/31/13	12/31/14	12/31/15
HGST	HDS5C3030ALA630	4,596	0.9%	0.7%	0.8%
HGST	HDS723030ALA640	1,022	0.9%	1.8%	1.8%
Seagate	ST3000DM001	4,074	9.8%	28.3%	28.3%
Seagate	ST33000651AS	325	7.3%	5.6%	5.1%
Toshiba	DT01ACA300	58	-	4.8%	3.8%
WDC	WD30EFRX	1,105	3.2%	6.5%	7.3%

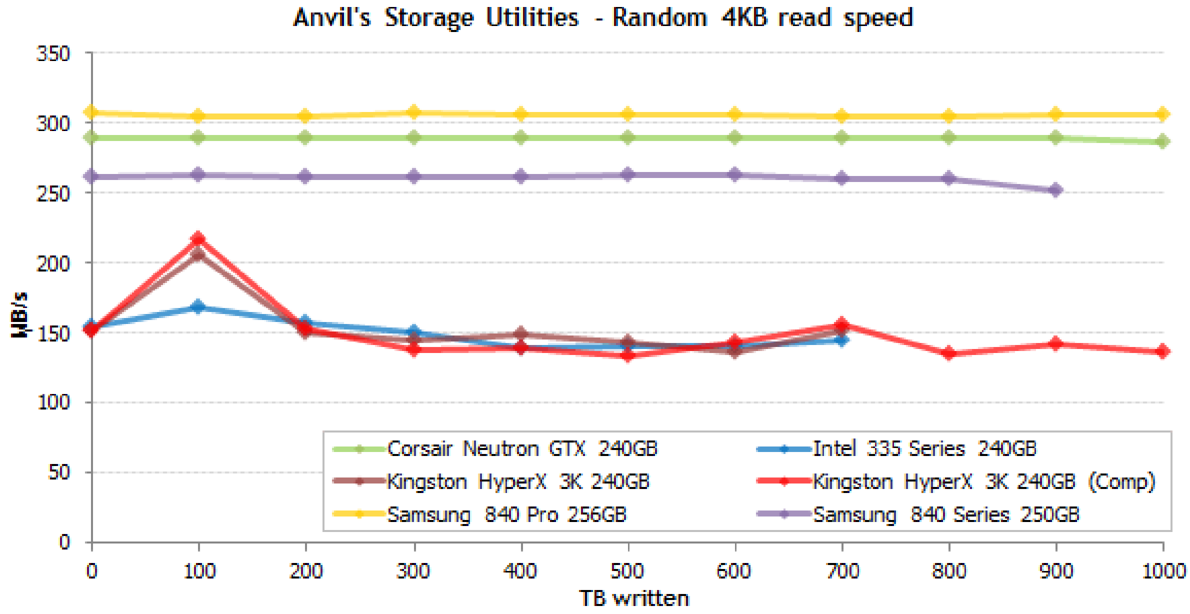


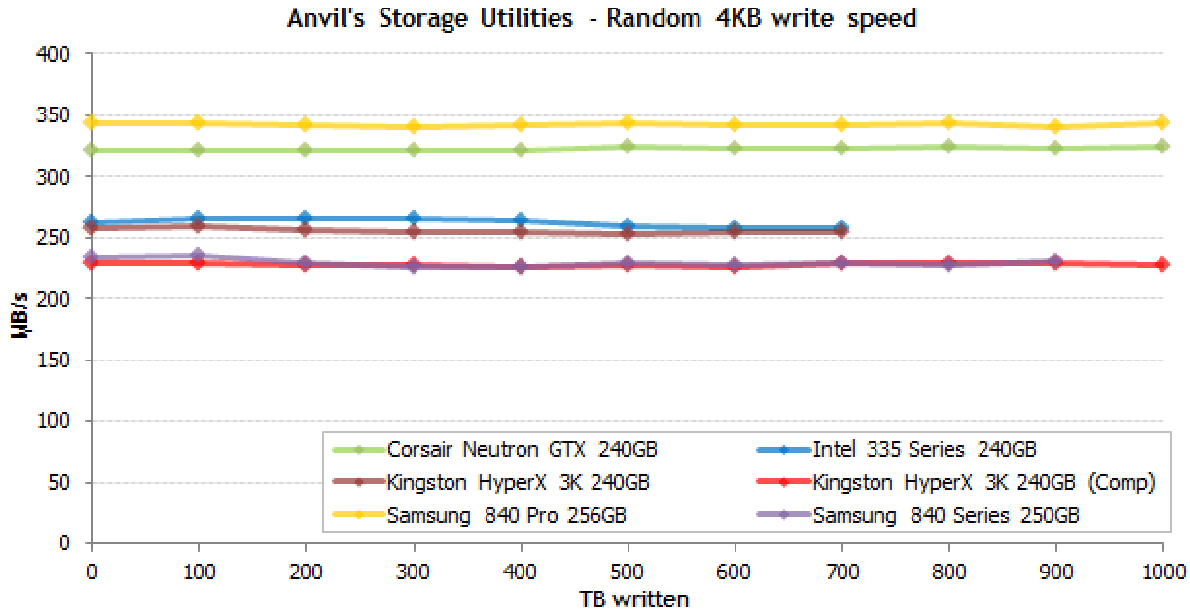
(HGST is the former Hitachi Global Storage Technologies)

BACKBLAZE

Reliability of SSD-s

- With the written amount of 50 GB/day, the expected lifetime is about 40 years

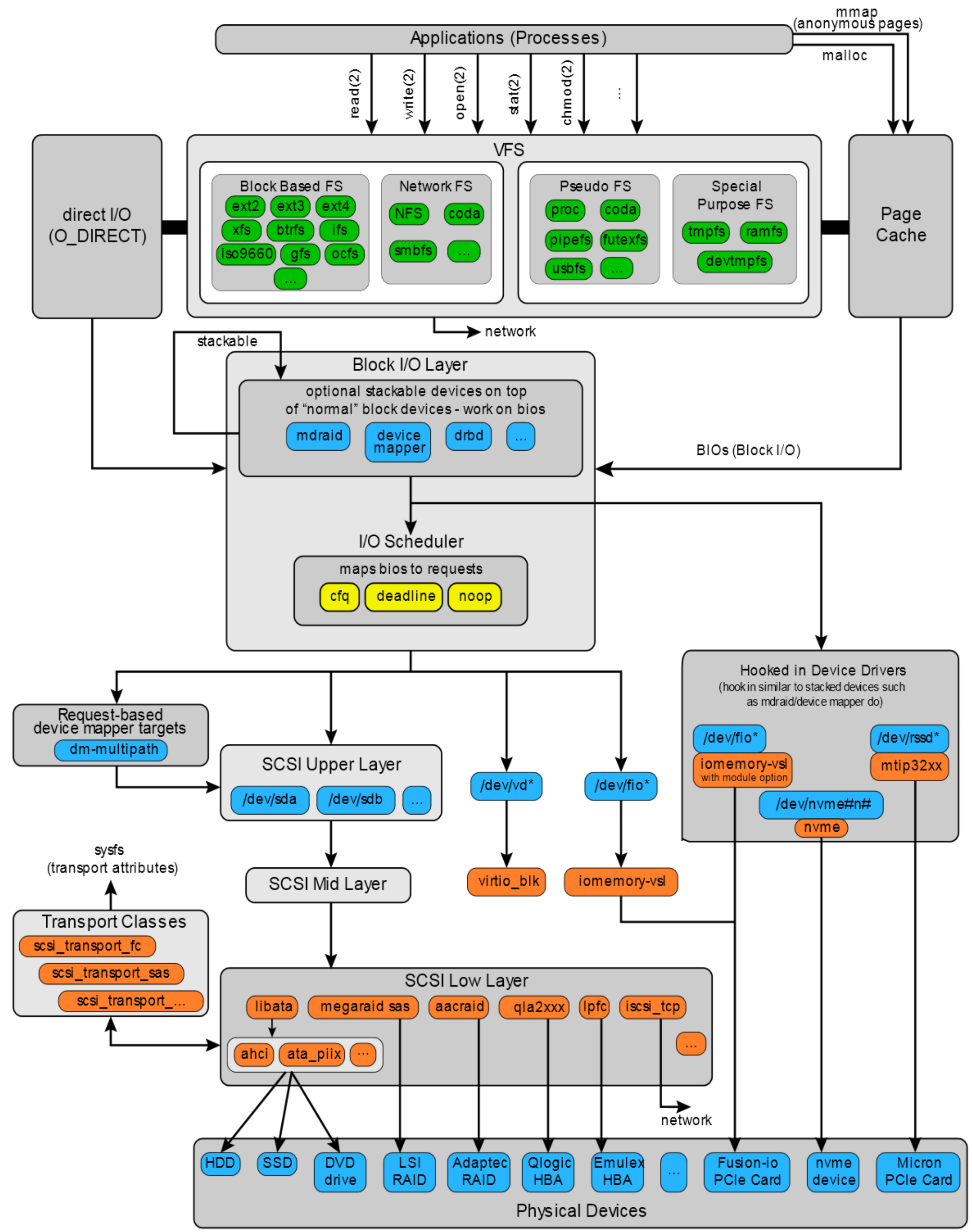




Source: http://techreport.com/review/24841/introducing-the-ssd-endurance-experiment

Overview of the topic

- User interfaces
  - User
  - Administrator
  - Programmer
- File systems
  - Kernel data structures
  - File system interfaces
  - Data arranged in blocks on disks
  - Virtual file systems
- Storing the data
  - Physical storages (HDD, SSD)
  - I/O scheduling
  - Local storage system virtualization (RAID, LVM)
  - Network and distributed file systems



Virtual storage systems: Logical Volume Management (LVM)

- Virtual storage systems can combine/merge more physical storages
  - Increase capacity, performance, reliability
  - Common management for multiple type devices
  - Easier maintenance: replacement of faulty drives, adding new devices

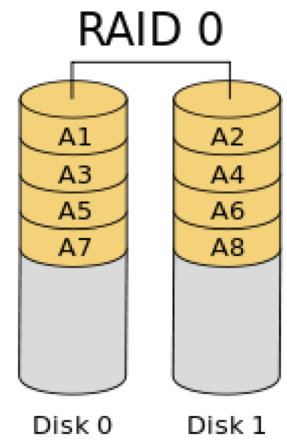
- **Logical Volume Management (LVM)**
  - An allocation system beyond the boundaries of the physical devices
  - More flexible management than partitions
  - Logical volumes can be created from partitions and disks, but other virtual sources also possible (network)
  - E.g. Windows: Logical Disk Manager, Linux: Logical Volume Manager
- Parts of the LVM
  - Physical volumes (PV): disks, partitions, other volumes
  - Logical volumes (LV): virtual disk partitions
  - Logical volume group (VG): a set of LV-s – virtual storage
  - Allocation units
    - Physical extents (PE): parts of the PV-s
    - Logical extents (LE): LE-s are assigned to PE-s (1-N)
      - Usually N=1 → 1 logical unit is stored by 1 physical unit
      - RAID may use it differently (see later)

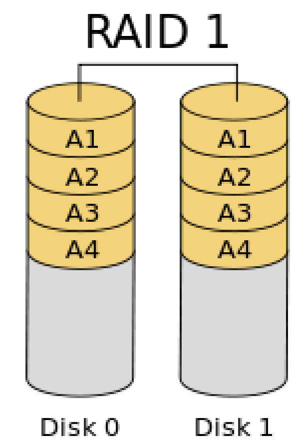
Virtual storage systems: RAID

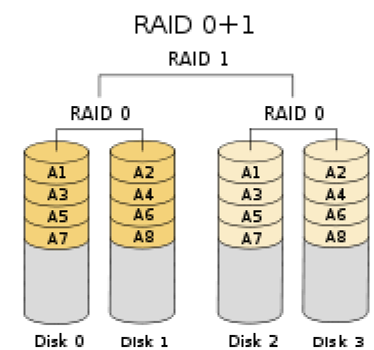
- Redundant Array of Inexpensive Disks
  - „Cheap” (smaller capacity) disks merged together
    - Recently I means Independent, the disks which are supporting RAID by HW are expensive
  - It defines a single common interface for the physical devices
  - Goal: improve redundancy (reliability), performance
  - HW and SW implementations
    - Mainboard RAID → SW (cheap)
    - RAID Disks → HW (expensive)
- Reliability
  - With the increasing number of devices, the possibility of a failure is also increasing
    - 1 disk MTTF: 100 000 hours, 100 disk MTTF: 1000 hours (41 days)
    - How can we increase the reliability with more disks?
  - Using redundancy
    - Storing additional information to correct errors
    - The most simple way is the mirroring: storing the data twice
      - Not so efficient from the capacity point of view
    - Parity: the parity bit can also detect the error and correct it

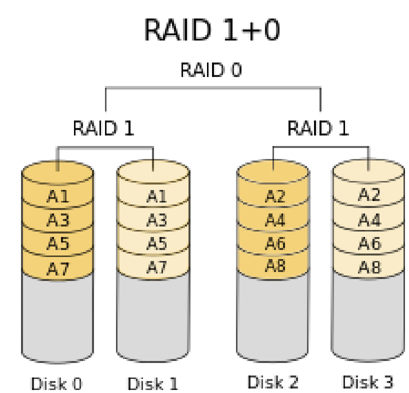
RAID levels: 0 - 1

- RAID level: the mode of merging the physical devices
  - How the data is distributed on the N disks
- **RAID 0 (stripe)**: the data is distributed on the N disks equally
  - Goal: improve performance
  - It can increase the throughput and the latency also
  - The disks capacities are combined
  - Failure of 1 disk → data loss
- **RAID 1 (mirror)**: the same data are stored on multiple disks
  - Goal: improve reliability
  - The combined capacity is the size of a single disk
  - Slower write operations, read can be faster
- Hybrid (nested) RAID solutions
  - **RAID 01** (0+1): mirror of stripes
    - Rather a theoretical possibility, not used in practice
  - **RAID 10** (1+0): stripe of mirrors
    - Great performance, improved reliability
    - Recommended for I/O intensive systems



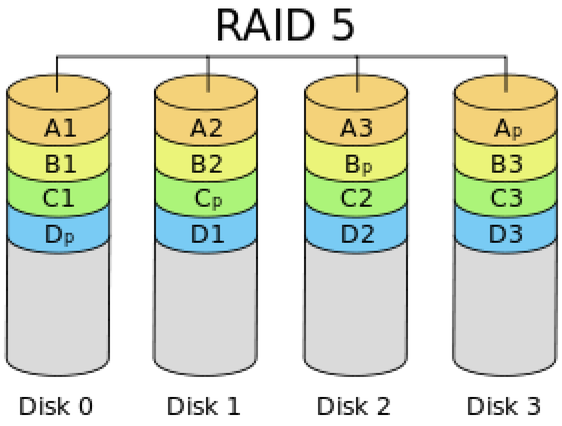




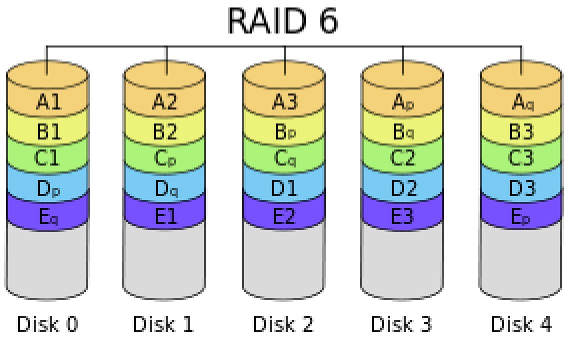


Widely used RAID levels

- Levels 2-3-4 are not used in practice
- RAID 5 and 6 are using parity for redundancy rather than mirroring



- RAID 5: block-level striping with distributed parity (N+1 disk fault tolerance)
  - A parity block is assigned to a group of data
  - This block is distributed among the disks
  - The performance is close to RAID0
  - the capacity is smaller with a size of 1 disk



- RAID 6: block-level striping with double distributed parity (N+2 disk fault tolerance)
  - Extension of RAID5 with an additional parity block
  - No significant performance degradation
  - The capacity is smaller with a size of 2 disks

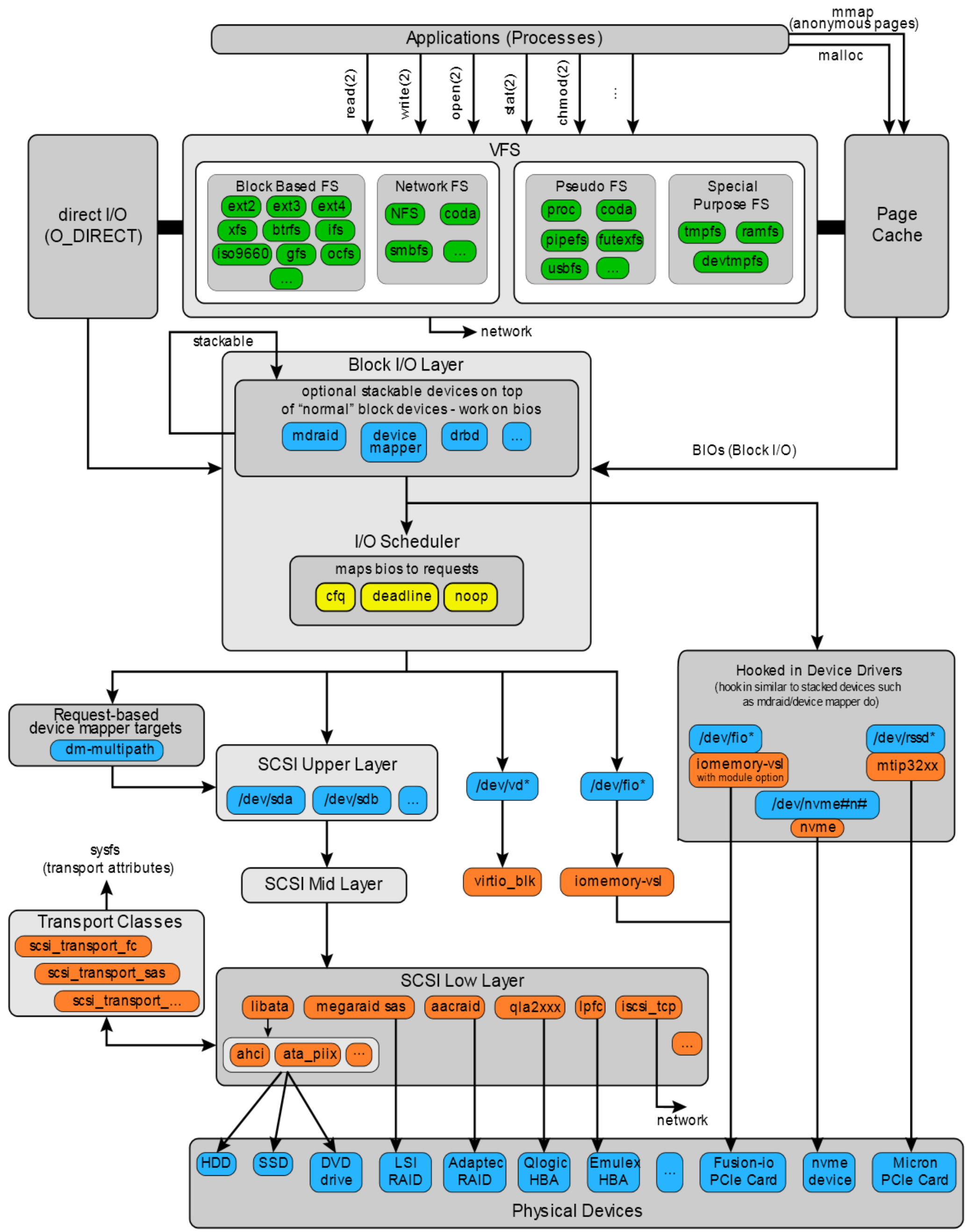
The limits of RAID (drawbacks)

- RAID is almost 3 decades old
  - When developed, the disk capacity was the fraction of today's disks
- How long does is take to correct an error?
  - In the case of 4+1 disks (RAID5)
    - 150 GB disks: ~10 hours
    - 6 TB disks: ~80 hours
  - Disk errors are not rare, a system cannot spend days with error correction
    - Hot spare and RAID6 may improve the situation
- RAID needs the same type of disks
  - After years, the replacement can be difficult
  - Moving the whole RAID array to new disks is a long time → long system downtime
- RAID is a bonded structure, not flexible
  - Cannot upgrade a RAID5 system to RAID6
- Limited combined storage capacity
  - The HW and SW solutions only managing 6-8 disks maximum

- RAID only protects against disk errors
  - What happens if the motherboard, CPU, RAM, power supply has an error?

Overview of the topic

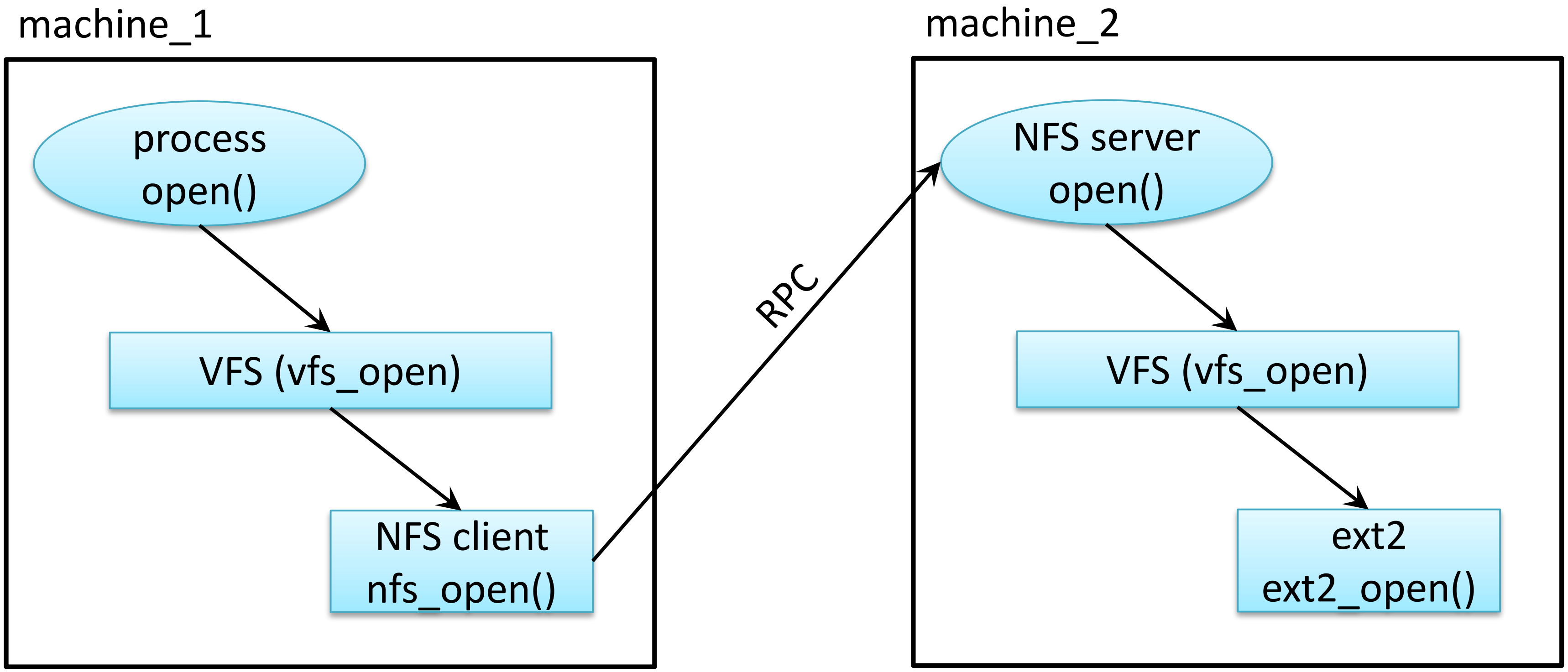
- User interfaces
  - User
  - Administrator
  - Programmer
- File systems
  - Kernel data structures
  - File system interfaces
  - Data arranged in blocks on disks
  - Virtual file systems
- Storing the data
  - Physical storages (HDD, SSD)
  - I/O scheduling
  - Local storage system virtualization (RAID, LVM)
  - Network and distributed file systems



Network and distributed file systems

- Goal: access to files stored in remote machines, sharing files
- Client-server based storage systems
  - Server: provides access to the local storage system
  - Client: connects to the server and grants access to the remote data
  - Network Attached Storage (NAS) file systems
    - High-level, file oriented transmission
    - NFS (Network File System), see next slide
    - SMB/CIFS (Common Internet File System) – Network file system of Windows
  - Block level network storage: SAN (Storage Area Network)
    - Low level data transmission
    - iSCSI (internet SCSI): for transmitting SCSI commands over IP
- Distributed file system
  - Operates as a distributed system
  - The data storage is distributed amongst the nodes of the system
  - Examples:
    - Ceph (Inktank, RedHat, SUSE), Google GFS, RedHat GlusterFS,
    - Windows DFS, PVFS → Orange FS
- Challenges: latency, network errors, consistency

A simple implementation of NFS



Challenges of network file systems

- Location: where is the data stored?
  - Location transparency
    - The name/path of the files are not referring the location
  - Location independency
    - The names and paths don’t change when the data is moved
- Question of network copies
  - The requests are served by remote services
    - Every operation should be performed on a single instance of the data
    - The network introduce latency and possible errors
    - The order of the operations are critical
  - The requests are served with the help of temporary local storages
    - the local machine maintain a copy of the data
    - Size is limited by the local machine
    - Multiple instances → consistency problems
- Operation of the network server
  - stateful: the file operations have a state (faster)
  - stateless: slower, but more reliable

Scalable, distributed storage systems: Ceph

- Universal, virtual storage systems (SW implementation)
  - Block based system (SAN)
  - File based system (NAS)
  - Object store (OSD)

- Scalable, fault tolerant
  - no single point of failure
  - Every component is replaceable at runtime (disc, machine)
  - Dynamic configuration (level of replication)

- Further advantages
  - PB capacity
  - Significantly faster error recovery than RAID
  - No special HW
  - Hot spares are not required (see RAID spare disk)
  - Cooperates with other virtualization systems (OpenStack, Amazon S3)
  - Open source

Further development of storage systems

- Integrated file and storage systems
  - Integrating the file systems with the solutions of RAID and LVM
  - e.g. zfs, btrfs
- Scalability
  - dynamic change of storage capacity (runtime)
- Reliability
  - large capacity → many disks → high possibility of errors
  - The error correction time should be eliminated
- Memory based storages
  - The SSD's speed is reaching the speed of the physical memory → new principles of development
- Data deduplication (e.g. zfs, btrfs)

- Further reading
  - Microsoft ReFS (Resilient File System)
  - Solaris ZFS (Z File System)
  - Linux Btrfs (B-Tree File System, „butter F S")
  - F2FS (Flash-Friendly File System, Samsung)
  - GPUfs (file access on GPU-s, see heterogenous multiprocessor systems)