Operating Systems

File and Storage Systems

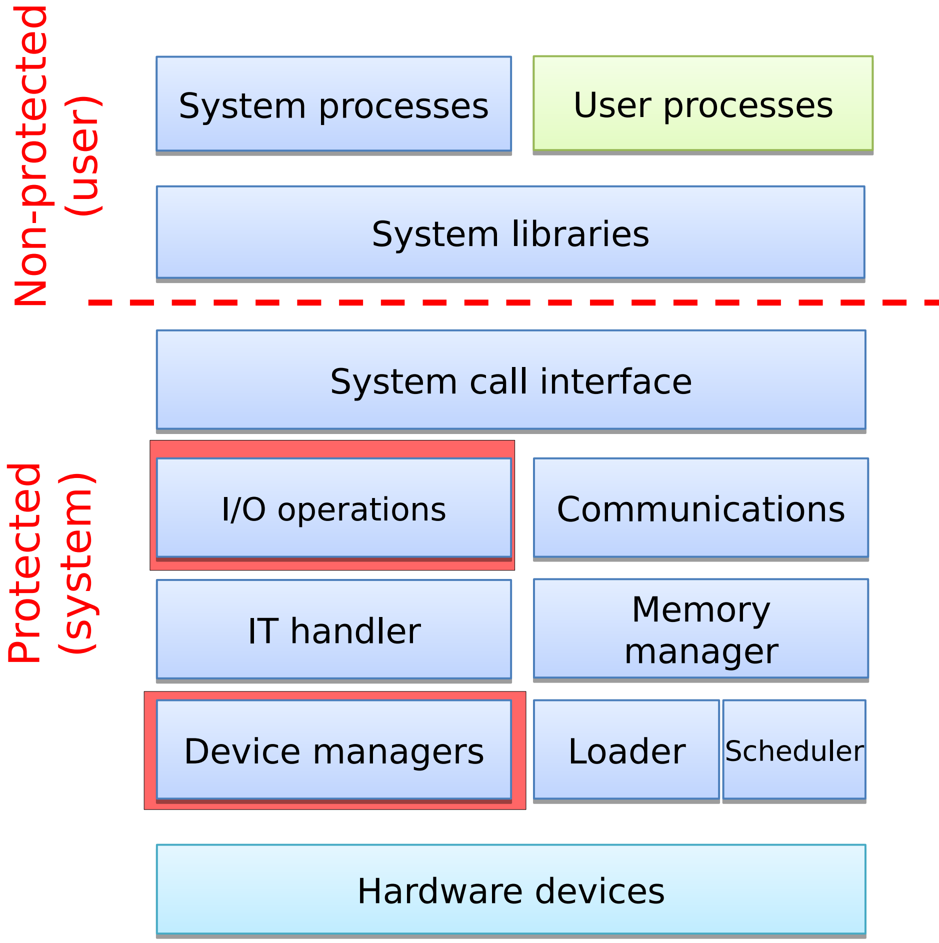
Tamás Mészáros
http://www.mit.bme.hu/~meszaros/

Budapest University of Technology and Economics (BME)
Department of Measurement and Information Systems (MIT)

Previously ...

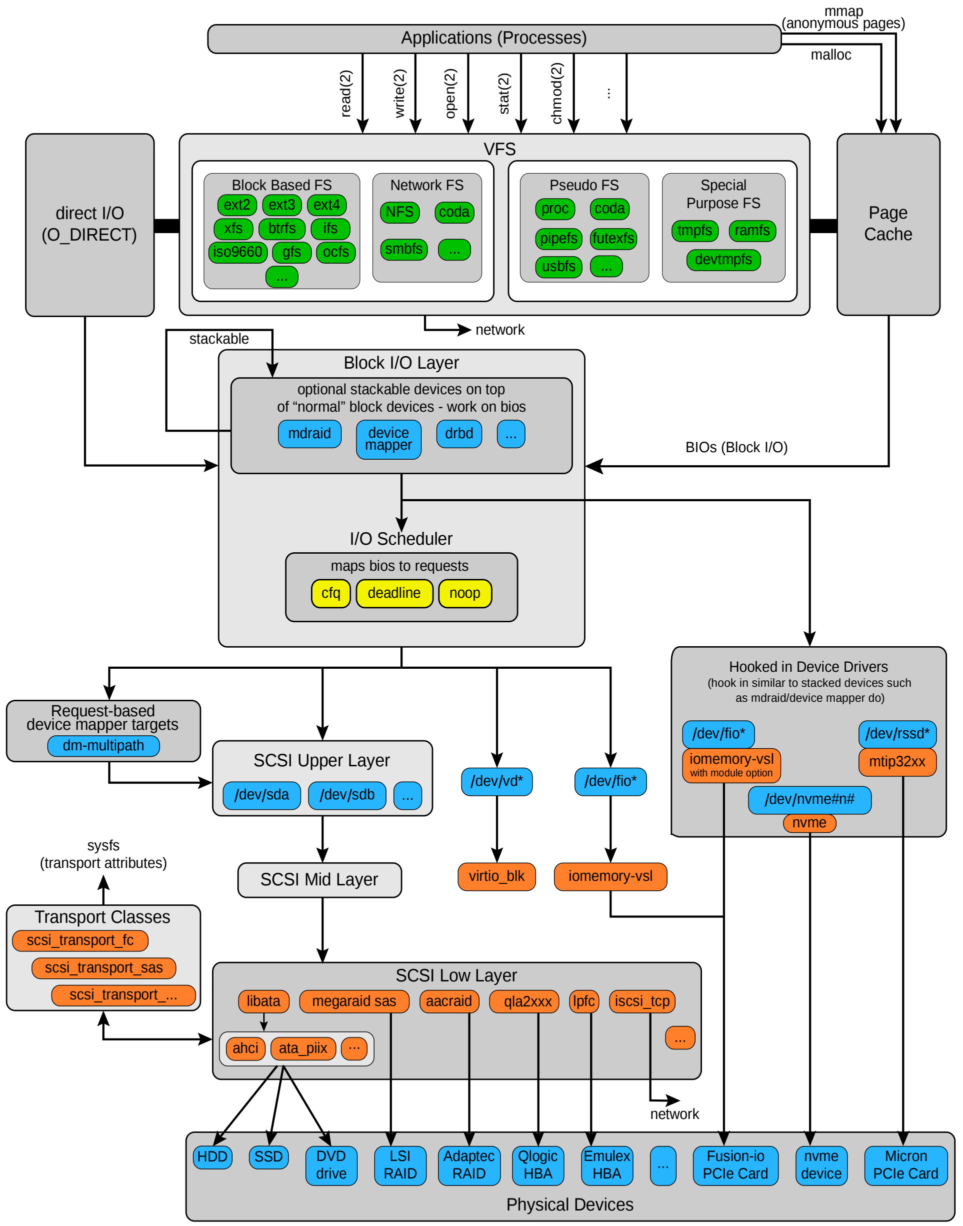
- Tasks
  - typically I/O intensive
  - perform many file operations
  - program code is in the filesystem
- Memory management
  - uses the disk storage to extend the physical memory
- Communication
  - over files (mmap)
- Lab exercises
  - Linux: network filesystem (Samba)
  - Windows: file properties, sharing, network drives

OS architecture



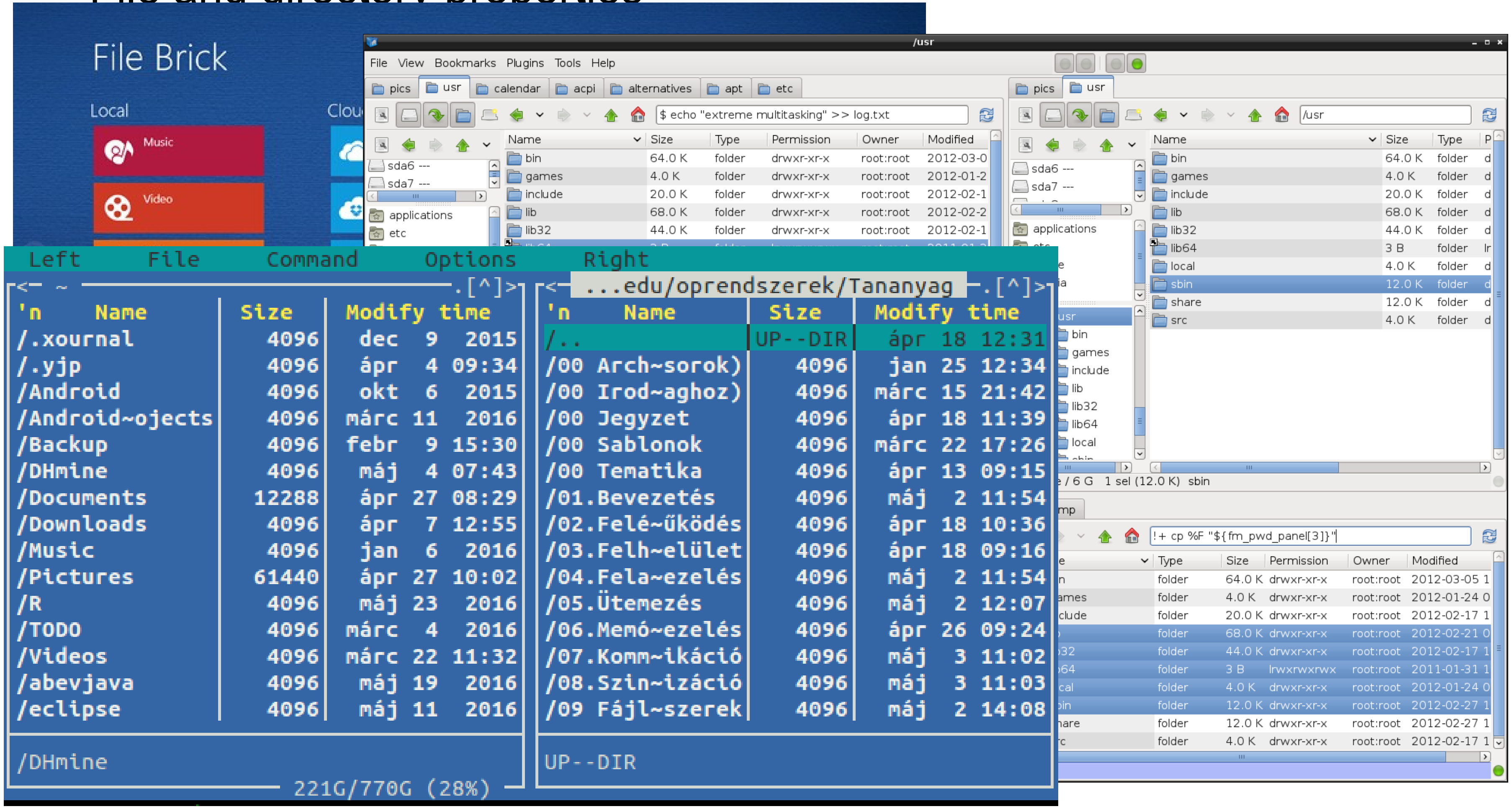
Overview

- The user perspective
  - end user
  - administrator
  - programmer
- Internals
  - filesystem APIs
  - disk organization
  - buffer cache
  - virtual filesystems
- Storage
  - devices (HDD, SSD)
  - I/O scheduling
  - virtualization:
    - local (RAID, LVM)
    - network (SAN, NAS)
  - distributed...


Applications (Processes)
mmap (anonymous pages)
malloc
read(2)
write(2)
open(2)
stat(2)
chmod(2)
...
VFS
direct I/O (O_DIRECT)
Block Based FS
ext2 ext3 ext4 xfs btrfs ifs iso9660 gfs ocfs ...
Network FS
NFS coda smbfs ...
Pseudo FS
proc coda pipefs futexfs usbfs ...
Special Purpose FS
tmpfs ramfs devtmpfs
Page Cache
network
stackable
Block I/O Layer
optional stackable devices on top of "normal" block devices - work on bios
mdraid device mapper drbd ...
BIOs (Block I/O)
I/O Scheduler
maps bios to requests
cfq deadline noop
Request-based device mapper targets
dm-multipath
SCSI Upper Layer
/dev/sda /dev/sdb ...
/dev/vd*
/dev/fio*
Hooked in Device Drivers
(hook in similar to stacked devices such as mdraid/device mapper do)
/dev/fio* /dev/rssd*
iomemory-vsl with module option
mtip32xx
/dev/nvme#n#
nvme
sysfs (transport attributes)
SCSI Mid Layer
virtio_blk
iomemory-vsl
Transport Classes
scsi_transport_fc scsi_transport_sas scsi_transport_...
SCSI Low Layer
libata megaraid sas aacraid qla2xxx lpfc iscsi_tcp
ahci ata_piix ...
...
network
HDD SSD DVD drive LSI RAID Adaptec RAID Qlogic HBA Emulex HBA ... Fusion-io PCIe Card nvme device Micron PCIe Card
Physical Devices

The end users' perspective

- Command line and GUI tools
- Organizational structure (places)
- File and directory properties


File Brick
Local
Music
Video
Clou
/usr
File View Bookmarks Plugins Tools Help
pics usr calendar acpi alternatives apt etc
sda6 ---
sda7 ---
applications
etc

```
$ echo "extreme multitasking" >> log.txt
```

Name	Size	Type	Permission	Owner	Modified
bin	64.0 K	folder	drwxr-xr-x	root:root	2012-03-0
games	4.0 K	folder	drwxr-xr-x	root:root	2012-01-2
include	20.0 K	folder	drwxr-xr-x	root:root	2012-02-1
lib	68.0 K	folder	drwxr-xr-x	root:root	2012-02-2
lib32	44.0 K	folder	drwxr-xr-x	root:root	2012-02-1

Name	Size	Type	P
bin	64.0 K	folder	d
games	4.0 K	folder	d
include	20.0 K	folder	d
lib	68.0 K	folder	d
lib32	44.0 K	folder	d
lib64	3 B	folder	lr
local	4.0 K	folder	d
sbin	12.0 K	folder	d
share	12.0 K	folder	d
src	4.0 K	folder	d

/ 6 G 1 sel (12.0 K) sbin

```
!+ cp %F "${fm_pwd_panel[3]}"
```

e	Type	Size	Permission	Owner	Modified
n	folder	64.0 K	drwxr-xr-x	root:root	2012-03-05 1
ames	folder	4.0 K	drwxr-xr-x	root:root	2012-01-24 0
clude	folder	20.0 K	drwxr-xr-x	root:root	2012-02-17 1
	folder	68.0 K	drwxr-xr-x	root:root	2012-02-21 0
32	folder	44.0 K	drwxr-xr-x	root:root	2012-02-17 1
64	folder	3 B	lrwxrwxrwx	root:root	2011-01-31 1
cal	folder	4.0 K	drwxr-xr-x	root:root	2012-01-24 0
bin	folder	12.0 K	drwxr-xr-x	root:root	2012-02-27 1
hare	folder	12.0 K	drwxr-xr-x	root:root	2012-02-27 1
c	folder	4.0 K	drwxr-xr-x	root:root	2012-02-17 1

```
  Left     File     Command     Options     Right
┌<─ ~ ─────────────────────.[^]>┐┌<─ ...edu/oprendszerek/Tananyag ─.[^]>┐
│'n     Name        │ Size │Modify time │'n     Name        │ Size │Modify time │
│/.xournal          │  4096│ dec   9  2015│/..                │UP--DIR│ ápr 18 12:31│
│/.yjp              │  4096│ ápr   4 09:34│/00 Arch~sorok)    │  4096│ jan 25 12:34│
│/Android           │  4096│ okt   6  2015│/00 Irod~aghoz)    │  4096│márc 15 21:42│
│/Android~ojects    │  4096│márc 11  2016│/00 Jegyzet        │  4096│ ápr 18 11:39│
│/Backup            │  4096│febr  9 15:30│/00 Sablonok       │  4096│márc 22 17:26│
│/DHmine            │  4096│ máj  4 07:43│/00 Tematika       │  4096│ ápr 13 09:15│
│/Documents         │ 12288│ ápr 27 08:29│/01.Bevezetés      │  4096│ máj  2 11:54│
│/Downloads         │  4096│ ápr  7 12:55│/02.Felé~űködés    │  4096│ ápr 18 10:36│
│/Music             │  4096│ jan   6  2016│/03.Felh~elület    │  4096│ ápr 18 09:16│
│/Pictures          │ 61440│ ápr 27 10:02│/04.Fela~ezelés    │  4096│ máj  2 11:54│
│/R                 │  4096│ máj 23  2016│/05.Ütemezés       │  4096│ máj  2 12:07│
│/TODO              │  4096│márc  4  2016│/06.Memó~ezelés    │  4096│ ápr 26 09:24│
│/Videos            │  4096│márc 22 11:32│/07.Komm~ikáció    │  4096│ máj  3 11:02│
│/abevjava          │  4096│ máj 19  2016│/08.Szin~izáció    │  4096│ máj  3 11:03│
│/eclipse           │  4096│ máj 11  2016│/09 Fájl~szerek    │  4096│ máj  2 14:08│
├───────────────────────────────────────┤├───────────────────────────────────────┤
│/DHmine                                ││UP--DIR                                │
└──────────── 221G/770G (28%) ──────────┘└───────────────────────────────────────┘
```

The Admin's and Programmer's perspective

- System Administrator
  - create, check and remove file systems
  - mount local and networked drives
  - performance tuning
  - disk usage
  - backup
- Programmer
  - APIs
    - system calls
    - system libraries
  - file descriptors and operations
    - create, open, read, write, seek, close, delete
  - file locking

Basic concepts

- File
  - logical unit for data storage
  - name (extension), properties
- Directory
  - logical unit for organizing files and directories
  - may contain files and directories
- Volume, drive
  - a set of files and directories
  - typically assigned to a partition on a disk

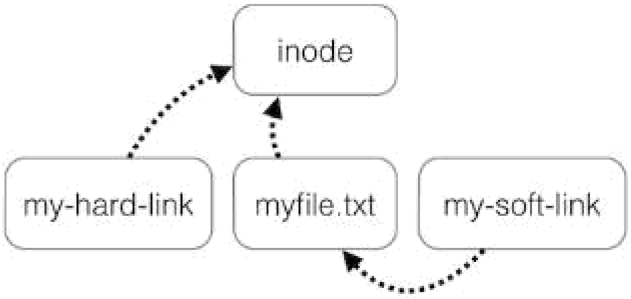
Logical

Physical

- File system
  - stores a coherent set of files and directories
- Partition
  - organizational unit for disk drivers
  - typically contains a file system

Directory structures, volumes and drives

- The basic structure is a directed tree
  - A directory can contain files and other directories
  - The direction of the edges is determined by the containment relation
  - Path: a place of a file or a directory in the tree
    - Absolute: the path from the root of the tree
    - Relative: the path from a specific node in the tree
      - Usually the actual working directory of the user



- Some systems (e.g. Unix) use further edges
  - Hard link
    - linked to the same data
  - Symbolic link (symlink, soft link, shortcut)
    - references a file or directory entry

  How can we delete the link or data?

  What happens if there is directed circle in the graph?

- There might be more than one trees in a system
  - There can be more volumes in the system, each one contains one tree
  - On Windows, the drives are named with C, D, E, etc. letters

Overview of the Windows 10 folder structure

- More than one folder structures (trees)
  - Physical storages are assigned with logical units, drives
    -
- The boot drive (usually C:) is the starting point (C:\)
  - `\Program Files` – installed applications
  - `\Program Files (x86)` – installed applications (32-bit)
  - `\ProgramData` – user independent data of the applications
  - `\Users` – user folders (files, folders, user dependent application data, …)
  - `\Windows` – the OS files and directories
    - »
- Further drives (D:, E:, …)
  - CD/DVD/USB drives
  - Further partitions on the disk
  - Network file systems
    - »
- Versions, trends
  - In the newer Windows systems the physical storages can be assigned to folders also (not just to volumes), but it isn't a widely-used feature

Overview of the UNIX directory structure

- It is organized into a single tree (no drives)
- The root directory is the starting point (/)
  - `/bin` – binary files for the system
  - `/sbin` – similar to /bin, usually programs with root permissions
  - `/dev` – hardware devices
  - `/etc` – system and application configuration files
  - `/home` – user directories and files
  - `/lib` – basic shared system libraries
  - `/mnt` – the mount point of physical partitions
  - `/tmp` – temporary files (for apps. and users)
  - `/usr` – user programs and libraries, documentation, etc.
  - `/var` – dynamic files of the system, logs, databases, …

  More details: `man hier`
- Disk usage

  `df, du, xdu, baobab, kdiskstat, filelight`
- File system „standards”, changes
  - Filesystem Hierarchy Standard (FHS) is just a recommendation
  - UsrMove: the /bin, /sbin is moved under /usr (Solaris11, Fedora)

Overview of the Android directory structure

- Similar to the Unix structure with additional directories
  - `/cache` – cache for applications
  - `/data` – user programs and data
  - `/data/app` – applications installed by the user
  - `/data/anr` – app-not-responding: error logs
  - `/data/tombstones` – memory dumps of the terminated apps.
  - `/data/dalvik-cache` – optimized binary files of the apps.
  - `/data/misc` – user configuration files
  - `/data/local` – temporary files
  - `/mnt` or `/storage` – mounted file systems, e.g. SD card
  - `/mnt/asec` – unsecured copies of the apps. running from SD card
  - `/system` – preinstalled apps., system libraries, configuration files

- Remarks
  - File is system access is limited, root user is inaccessible by default.
  - Apps. stored on the SD card are encrypted (`.android_secure`), these files are mounted under the `/mnt/asec` directory while running

File properties (with Unix examples)

- List the content of the actual directory (ls -la)

```
drwx------   6 root root    4096 Feb 23 14:20 .
drwxr-xr-x 22 root root    4096 Nov 21  2014 ..
-rw-r--r--   1 root root     570 Jan 31  2010 .bashrc
-rw-r--r--   1 vps  vps    71103 Nov  5  2013 package.xml
-rwxrwxrwx   1 root root      35 Feb 23 14:21 test.sh
lrwxrwxrwx   1 root root       8 Nov 24  2014 www ->
/var/www
```

- What is in the list?
  - Type of the entry: - d p l b c s
  - POSIX permissions (see next slide)
  - Number of links
  - Owner and group
  - Size
  - Timestamp (ctime: change of the metadata, mtime: data modification, atime: access time)
  - Name of the entry
- The OS also stores
  - Unique identifier (for internal identification)
  - Location (where the file is stored on the disk)

The Unix permission systems

- POSIX permissions
  - 3x3 bits: owner, group, others X read, write, execute
  - Values: read-4, write-2, execute-1, no access-0
    - E.g.: 740 = owner: RWX, group: R, others: no access
  - In the case of directories, the execute means „list”
  - Setting: chmod <permissions> <file/directory>
    - E.g.: chmod 750 /home/me   chmod u+rwx,g+rx,o-rwx /home/me
- Special permissions: SETUID, SETGID, StickyBit
  - SETUID/GID: set user ID upon execution" and "set group ID upon execution
    - The executed file will have the same permission as the owner (not the user which executed the file)
    - It is usually set to files which require root permissions
  - StickyBit: only the owner (and root) can delete/rename the files or directories

```
drwxrwxrwt 44 root root     12288 máj    9 15:25 /var/tmp
```

- POSIX ACL (access control list) (*extended permissions*)
  - flexible, can store several access control lists for an entry

```
setfacl -m u:student:r file
```

Sysadm tasks

- Create (format)
  - type (next slide)
  - properties
  - name (for humans), ID (for machines)
  - storage (disk, network etc.)
- Mount
  - physical → logical assignment
  - mount point
  - mounted file system covers a part of the file system tree
- Check, tune
  - offline checking
  - change size of storage has changed
  - tune for performance, compression etc.
- Backup (see later)

An overview of the widely used file systems

- FAT32
  - Typically used on portable storage devices because the compatibility
  - Originally 8+3 character file names extended to 255 characters, maximum file size: 4GiB (!)
- NTFS
  - Default file system in Windows
- UFS/ Berkeley FFS
  - Traditional UNIX file system, currently rarely used
- ext2,3,4 (cased on UFS)
  - Currently used file systems in Linux systems
- XFS
  - Default in RedHat Linux 7
- HFS+
  - Default in MacOS
- Integrated file + virtual storage systems (see later)
  - ZFS: Designed for Solaris, later it become open source, popular in BSD-s also
  - Linux btrfs: newer, currently under development
- Many more file systems
  - CD/DVD file systems

Practice in Linux

- Basic file and directory operations

  `cp, mv, cd, pwd, mkdir`

  How to rename a file?
- File attributes: `ls -la`
- Managing file systems: `mount, umount, df, mkfs, fsck`
- Example: create a file system in a file

  ```
  dd if=/dev/zero of=filesystem.img bs=1k count=1000
  losetup /dev/loop0 filesystem.img
  mke2fs /dev/loop0
  mount /dev/loop0 /mnt
  ```

  An annoying error: device is busy

  while unmounting a file system

  check what is used: `lsof /mnt`
- What's happening in the file system?

  ```
  iotop, sar, dstat, vmstat, …
  sudo sysctl vm.block_dump=1
  tail -f /var/log/kern.log
  ```

Backing up and restoring data

- Multiple causes of data loss
  - Uncorrectable fault in the file systems
    - The error in the physical storage (disk error)
    - Inconsistency caused by power failure or other HW error
  - User mistakes (not rare)
    - Accidental deleting of files or whole file systems, partitions
  - Malwares (sadly these are also not rare)
    - Deleting or encrypting data (ransomware)
- The type of data loss
  - Limited (e.g.: disk error, user mistakes, …)
  - Total (e.g.: SSD sudden death)
- Creating a backup
  - How: automated (regular), manual (casual)
  - What: files or whole file system
    - A consistent state has to be backed up – problematic when the FS is in use
  - Where: high capacity disks, CD/DVD, tape systems
- Restoring the system from a backup (recovery)
  - Bare metal recovery: restoring the whole system
  - Data recovery: only recovering specific files

The programmer's perspective

Programming interfaces

- Opening (and creating) files
  - `open()` system call and its arguments
  - File descriptor and the opened file object (metadata)
  - File opened by multiple processes?
- Read, write, seek: `read()`, `write()`, `fseek()`
  - Sequential access: the data is accessed in the stored order
  - Direct access: given sized blocks can be read in any order
- Close files: `close()`
- Managing directories:
  - `opendir()`, `readdir()`, `rewinddir()`, `closedir()`

Locking files

- Locking files
  - a file may be a shared resource
  - synchronization problem: keep the file content consistent
  - we may use any synchronization method
  - the kernel also provides efficient and better locking methods for files
  - note: deadlocks are also possible
- Advisory locking
  - the OS provides tools for the tasks but they may ignore these
  - tools used → locking works
  - tools not user → no locking
- Mandatory locking
  - locking enforced by the kernel
- The scope of locking
  - whole or a part, see Windows LockFileEx(), Unix fcntl()

Shared access to files through memory (mmap)

- Communicate through a file
  - It is problematic with the standard op.-s (read(), write(), fseek())
  - Can we use a file like the shared memory?
- UNIX mmap, Windows: CreateFileMapping
  - An open file object (open()) can assigned to an address: mmap(addr, size, prot, flags, fd, offset)
    - addr: the assigned address, 0: the kernel choses
    - size: the accessed data range
    - prot: the mode of access: R, W, X
    - flags: own or shared file, etc.
    - fd: file descriptor returned by the open() system call
    - offset: the start position
  - Return value: the assigned virtual memory address
  - Close the assignment: munmap(addr, len)

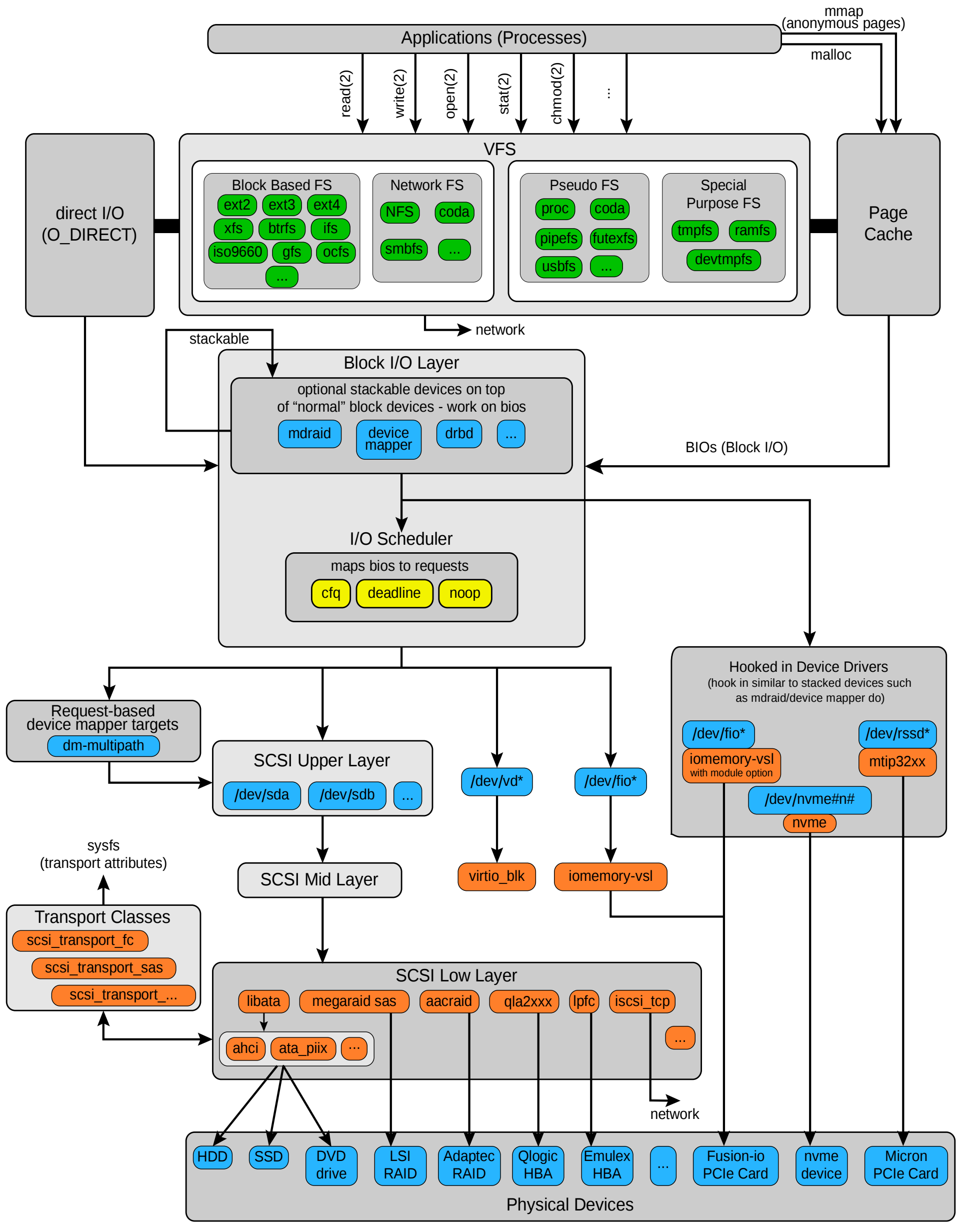
I/O operations without waiting

- Non blocking I/O
  - syscall options not to block the task
  - the call returns immediately
    - with the data
    - or with „no data” error
  - the task may/should retry the operation any time

- Asynchronous I/O
  - the task initiates the I/O and sets a buffer for the data
  - the asynchronous I/O is performed in the background
    - the system call returns immediately
  - the task can perform other instructions
  - when the I/O is done, the kernel notifies the caller
    - e.g.: with a signal with custom handler
  - see POSIX aio, Windows I/O Completion ports

Overview

- The user perspective
  - end user
  - administrator
  - programmer
- Internals
  - file system implementations
  - disk organization
  - buffer cache
  - virtual file systems
- Storage
  - devices (HDD, SSD)
  - I/O scheduling
  - virtualization:
    - local (RAID, LVM)
    - network (SAN, NAS)
  - distributed...

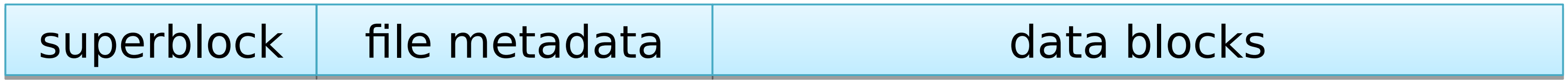


Implementation of file systems (overview)

- Operation from the user’s point of view (already discussed)
  - Files, directories, tree/graph structure
  - Format, mount, unmount
  - Check, repair, create, modify, tune
- Organizing the file system in the storage
  - The logical units are assigned to physical devices
  - The data is stored in blocks
  - Beside the file contents, metadata is also stored
  - Managing the free (unused) blocks in the storage device
- Run-time operations
  - File system descriptors (metadata of the mounted file systems)
  - Descriptors (metadata) of the files
    - Access to opened files
  - Managing the data in the memory, buffering

File system structure

- The stored data
  - File system metadata (superblock, master file table, partition control block)
  - File metadata (inode, file control block, on Windows: it is part of master file table)
  - Stored data



- The file system metadata

On disk

- Type and size
- List of free blocks
- The location of the file metadata
- State
- Modification information
- …

In memory

- everything that is on the disk
- mount information
- „dirty" bit
- locking info
- ...

- The file system is sensitive to metadata loss (e.g. hardware error)
  - Therefore backups are made
  - See: `dumpe2fs /dev/sda1 | grep -i superblock`

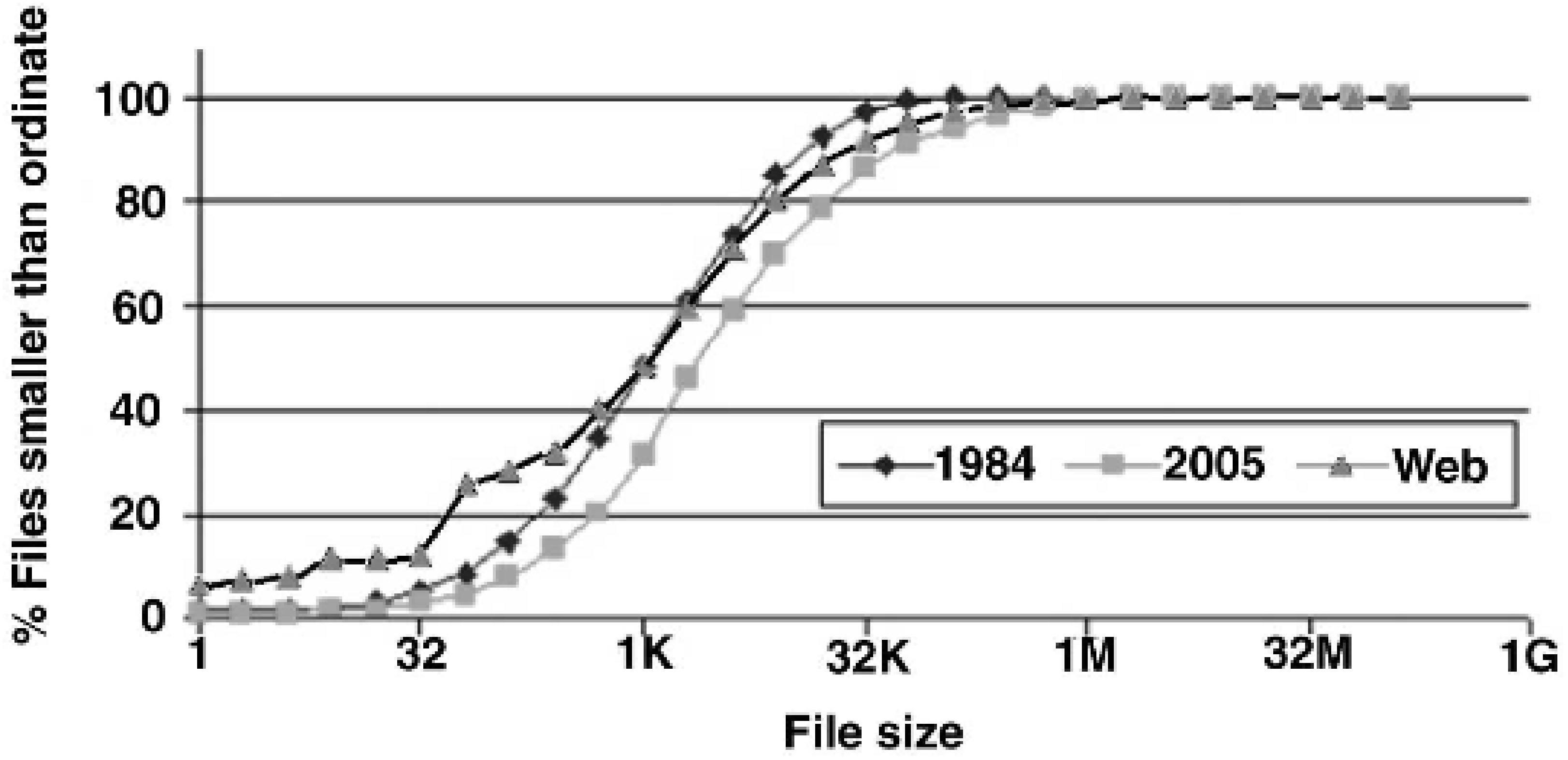
File metadata

- **On disk**
  - Authentication information (UID, GID)
  - Type
  - Permissions
  - Timestamps
  - Size
  - Data block locations
  - Example: UNIX inode (index node), Windows Master File Table entry
- **Runtime extensions** (in memory)
  - State (locked, modified, etc.)
  - Disk/file system identifier
  - Reference counter (file descriptors)
  - Mounting point descriptor

Storing data blocks (allocation methods)

- Continuously on the disk…
  - simple but causes serious fragmentation over time
- Chained list (sequential access), e.g. FAT
  - data is divided into in smaller chunks (blocks)
  - data blocks are stored in a linked list
    - the address of the first part is in the metadata
    - every part contains the address of the next part
  - efficient for sequential access, not good for random access
  - very sensitive to errors
- Indexed storage (direct access)
  - data is divided into in smaller chunks (blocks)
  - the location/map of the blocks: **the index** (see next slide)
  - block may be stored sequentially if possible
  - efficient, good for random and also for sequential access
  - sensitive to loosing the index (may be duplicated)

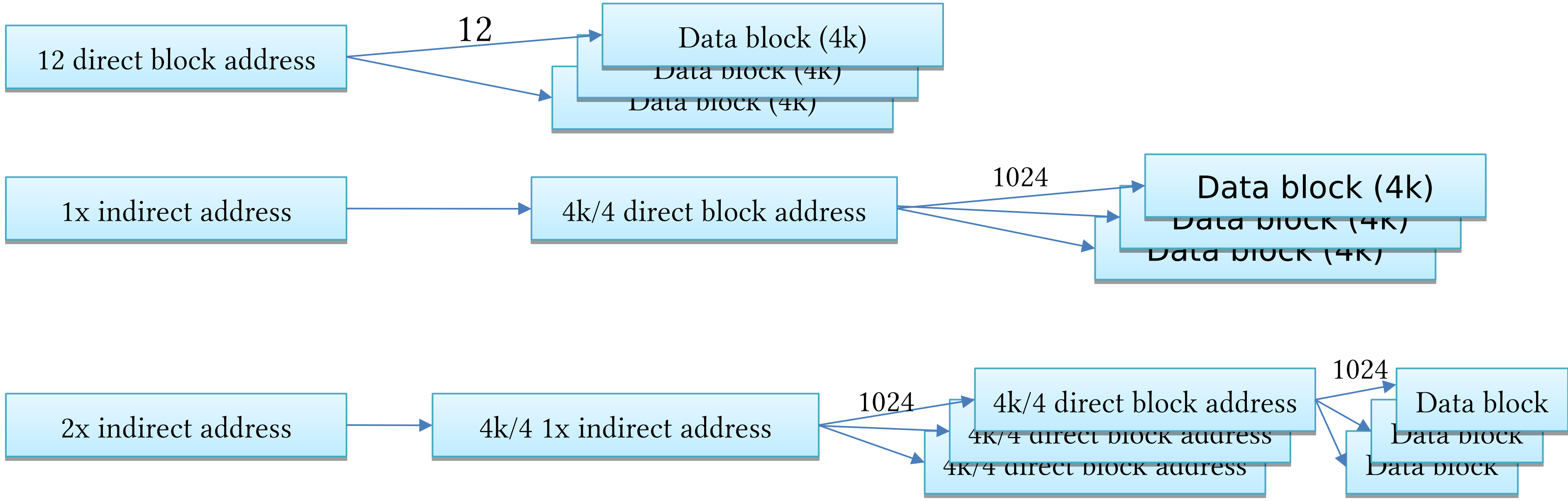
How to determine the block size?



Source: Andrew S. Tanenbaum, Jorrit N. Herder, Herbert Bos
File size distribution on UNIX systems: then and now. Operating Systems Review 40(1): 100-104 (2006)

Example: Multiple indexed data block address table

- Index table
  - 12 direct block address
  - single and double indirect block address (to cover large files)
  - 4 kB block size
  - 4 byte address



What is the maximal file size?

Managing the free blocks

- Bitmap, bit-vector description
  - Every block is represented by a bit
  - 1=free, 0=used
  - Simple method, easy to find a free block
    - The map can be stored in the memory for smaller FS
    - Typically there is a CPU instruction for getting the first non zero bit location
  - It uses more memory for a larger file system
- Chained list storage
  - The free blocks are marked and the address of the next free block is written there
  - Only the address of the first free block has to be stored
  - Simple, but not so efficient method
  - It can be combined with the chained list block allocation method
- Hierarchical methods
  - Managing the group of (free) blocks
  - The groups can be created based on the size of the FS
  - Within a group, a simpler structure can be used (e.g.: bitmap)

Accelerating data transfers

- Disk buffering
  - to accelerate the access to frequently used data
  - works in coordination with virtual memory management (paging)
    - data is loaded into frames by the VMM
    - page replacement may also free disk buffer frames if needed

- Accelerating..
  - read operations
    - read ahead automatically
    - may be instructed using system calls (see posix_fadvise)
  - write operations (when to write the modified data to the disk)
    - **Write through cache**
      - write immediately to the storage
      - slow but reliable
    - **Buffered write**
      - writes data periodically (flush, sync)
      - significantly faster but may cause data loss

Metadata consistency and journaling file systems

- When write metadata in the memory to the disk
  - similar to data buffering but more complex problem due to transactions
  - a write through cache may cause significant performance loss

    e.g. file access times are changing rapidly

- Journaling file systems
  - changes are saved to a journal, which is always stored on the disk
    - file system operations are grouped into transactions
    - transaction is finished when the it is stored in the journal
    - the transaction data is deleted when it committed to the filesystem
  - the journal is sequential access circular buffer
  - What happed if the system crashes? The journal is processed during reboots.

- Log-structured file system: the log is the file system (e.g. BSD LFS)

- Copy-on-write file system (ZFS, btrfs)
  - a different solution that works on duplicated metadata structures

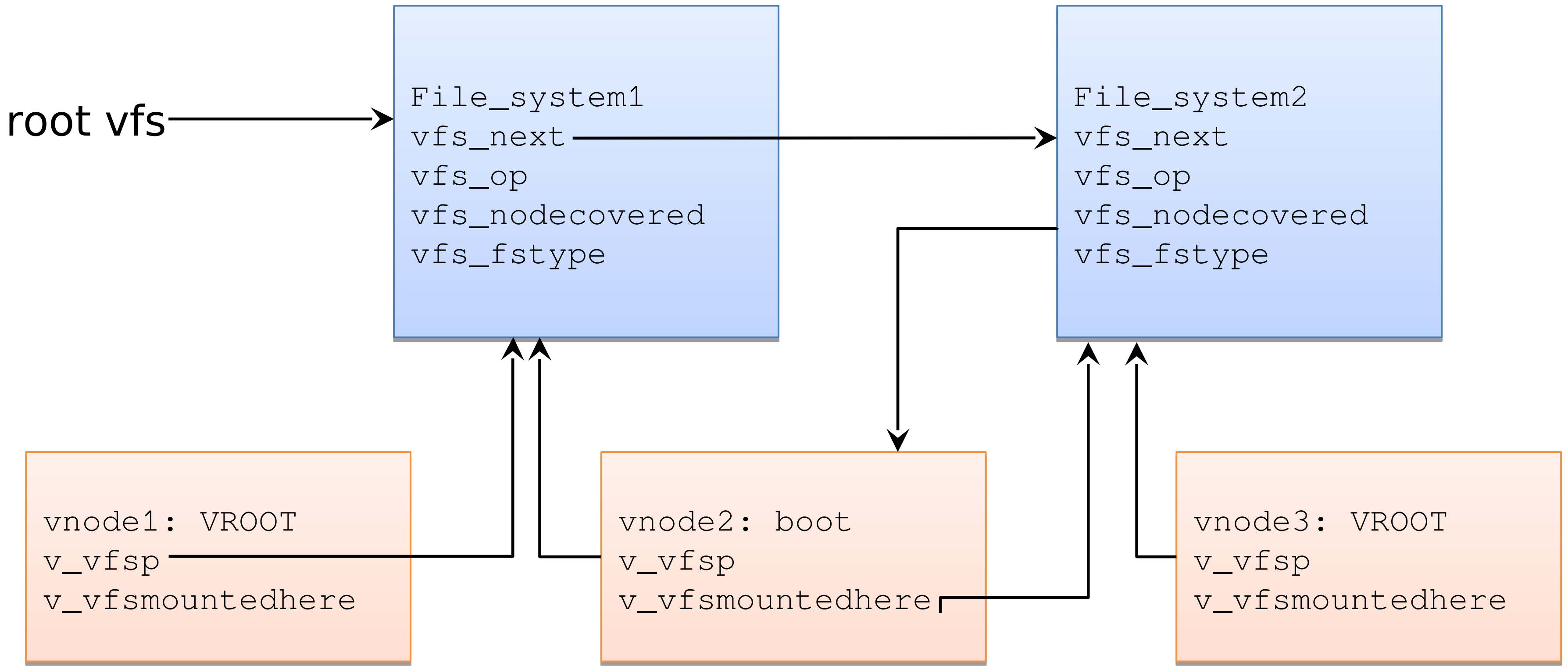
Virtual File Systems

- 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”
```

Implement your own file system using VFS

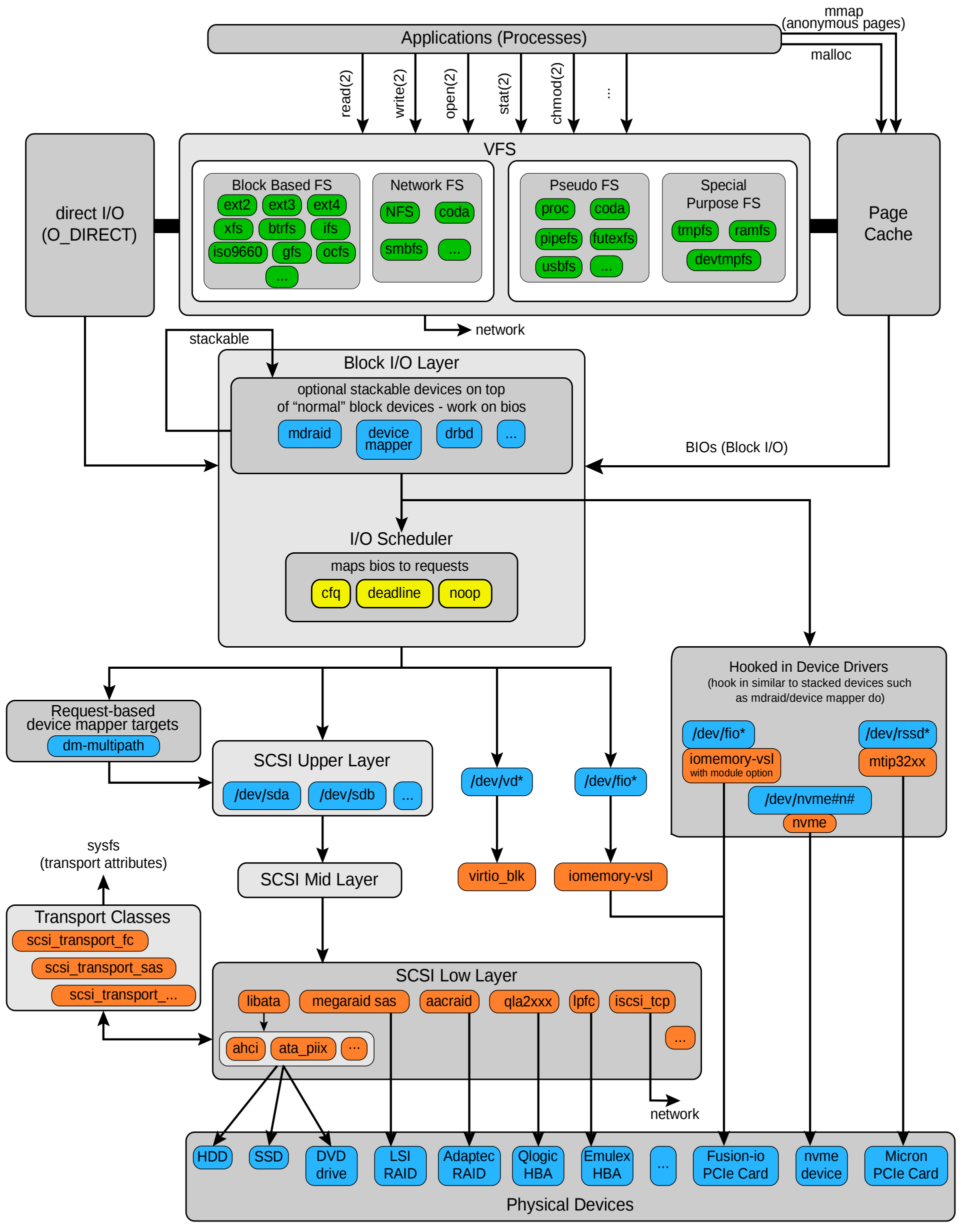
- Is not that hard...
- See

Ravi Kiran, „Writing a Simple File System”

Steve French, „Linux Filesystems 45 minutes” ODP PDF
„A Step by Step Introduction to Writing (or Understanding) a Linux Filesystem”

Overview

- The user perspective
  - end user
  - administrator
  - programmer
- Internals
  - filesystem APIs
  - disk organization
  - buffer cache
  - virtual filesystems
- Storage
  - devices (HDD, SSD)
  - I/O scheduling
  - virtualization:
    - local (RAID, LVM)
    - network (SAN, NAS)
  - distributed...

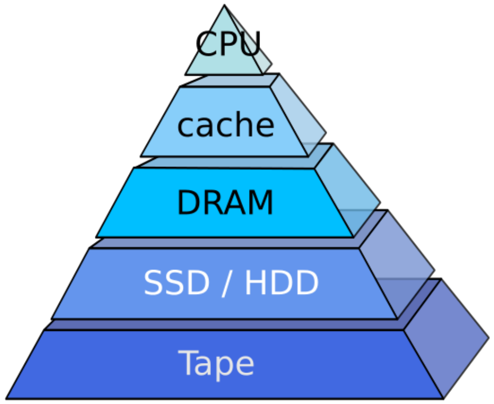


Storage solutions

- Physical 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
  - extend (capacity, services) other storage systems
    - merging
      - e.g. RAID, LVM
    - network access
      - file or block level transfer
      - e.g. NAS, SAN
    - distributed 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, …

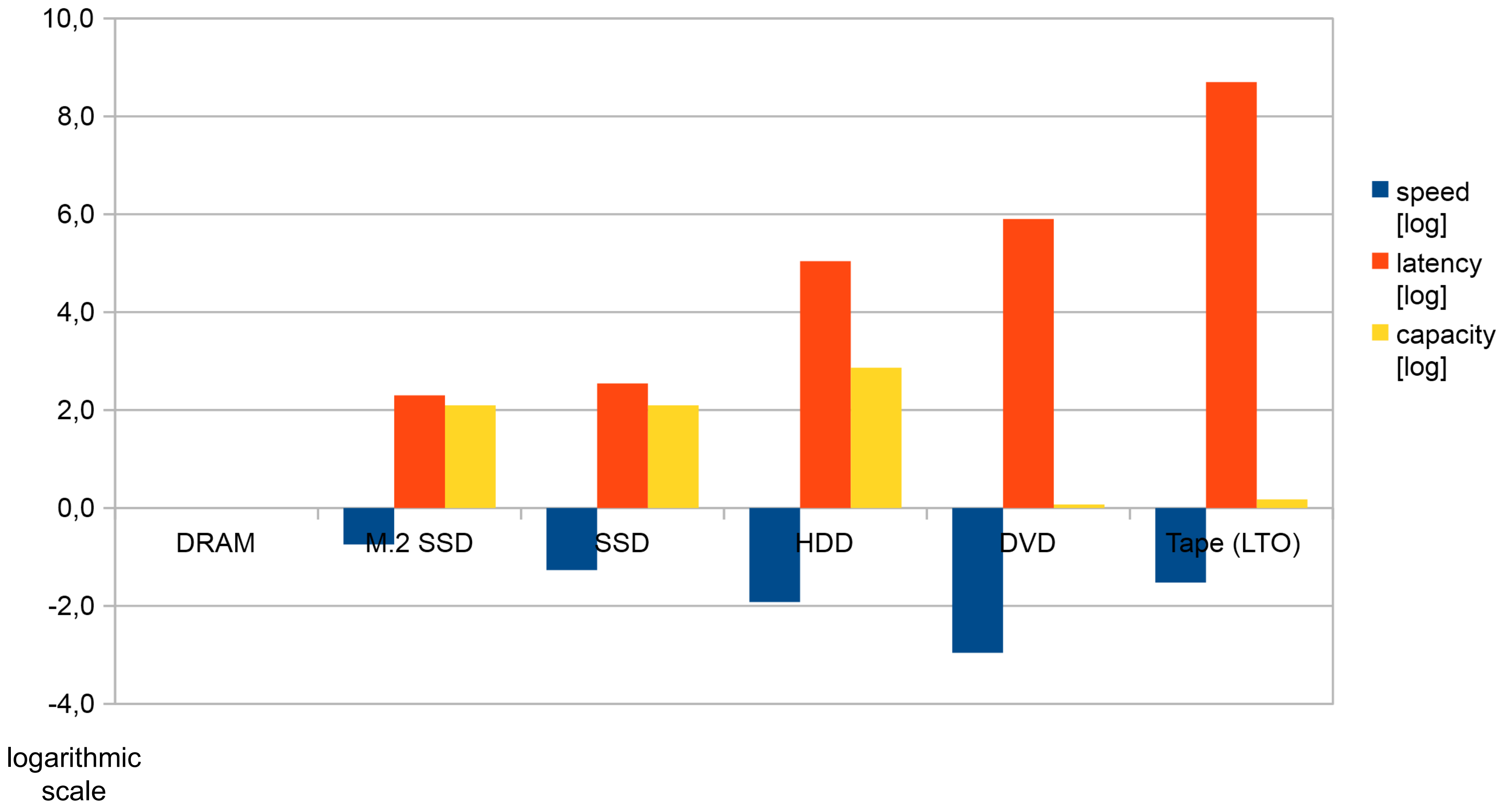
Properties of physical storage systems

- Characteristics
  - capacity from bytes to petabytes
  - throughput (read/write) 10 MiB/s ... 200 GiB/s
  - access time: 0.5 ns ... seconds/minutes



- 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

Performance compared to DRAM



Trends of physical storage systems

- In the past
  - significant performance difference between CPU and disks
    - The CPUs were developed faster than HDDs
    - “slow I/O” was a design principle for operating systems

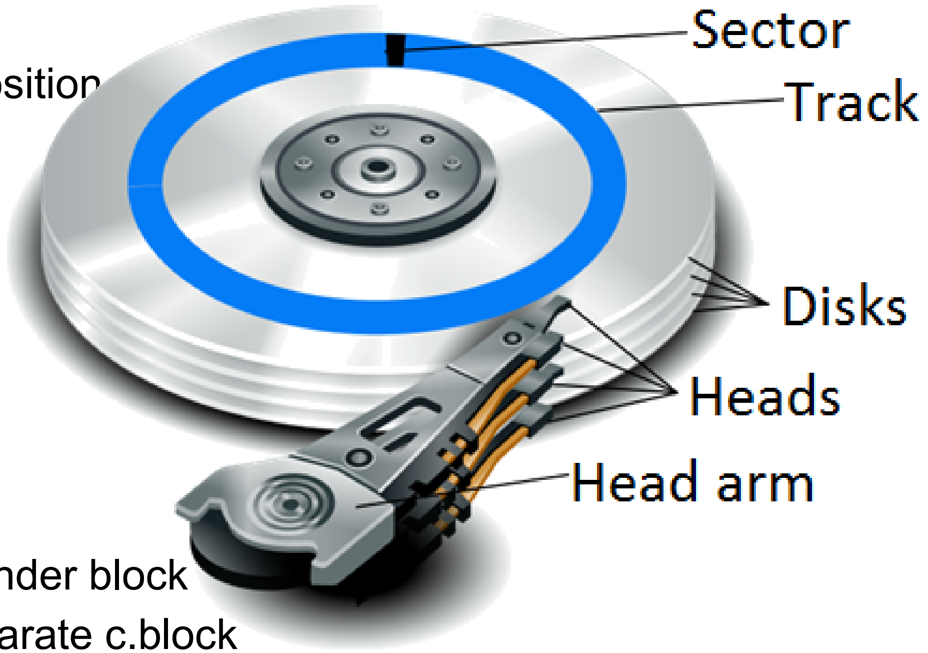
- Present and near future
  - the size of the physical memory is greatly increased
    - the size of disk cache is higher
  - new methods based on fast CPU-s
    - runtime data compression (ZFS, btrfs)
    - deduplication
      - avoiding the storage of the same data part more than one time
  - memory based „disks”
    - increasing speed with very low latency
    - **storage class memory**: almost DRAM performance with large capacity

Tape drives

- Traditional tool for backups
  - High capacity
  - Long lifetime
  - slow operation
  - manual / robotized cassette change
- Recent developments
  - high sequential read speed
    - Tape – 300 MB/s, SSD – 500 MB/s
  - larger caches
    - Almost every data is there
    - Filled with sequential read
  - see log-structured file systems
    - sequential read/write
  - good as a general storage?

Magnetic (mechanical) 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

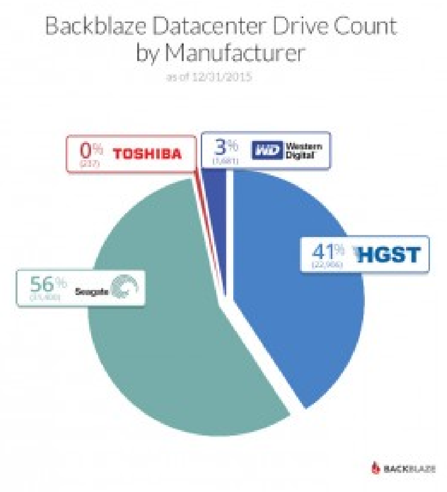


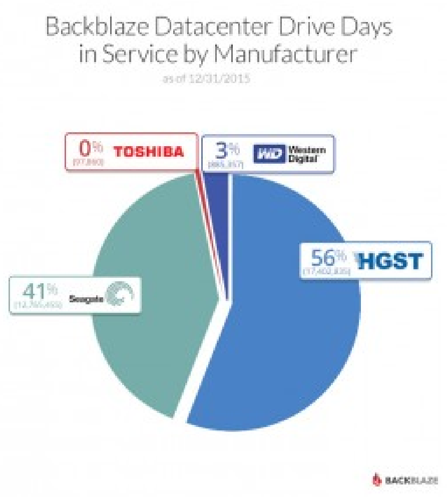
Scheduling of disk operations

- Scheduling increases the performance
  - especially on mechanical drivers with slow access times
- E.g. Linux I/O Schedulers
  - **Noop**: simple FIFO
    - may concatenate adjacent requests
    - small overhead
    - recommended if the storage system has internal scheduling (RAID, NCQ, VMs)
    - best for CPU intensive systems (low load on disks)
  - **Deadline**: tries to perform requests before a deadline
    - 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
    - has queues for every process and assigns a time-slice to them
    - estimates the load for each queue
    - scheduling depends on this estimation and the priority of the queues
    - recommended for general usage (usually this is the default)

Reliability of hard disk drives

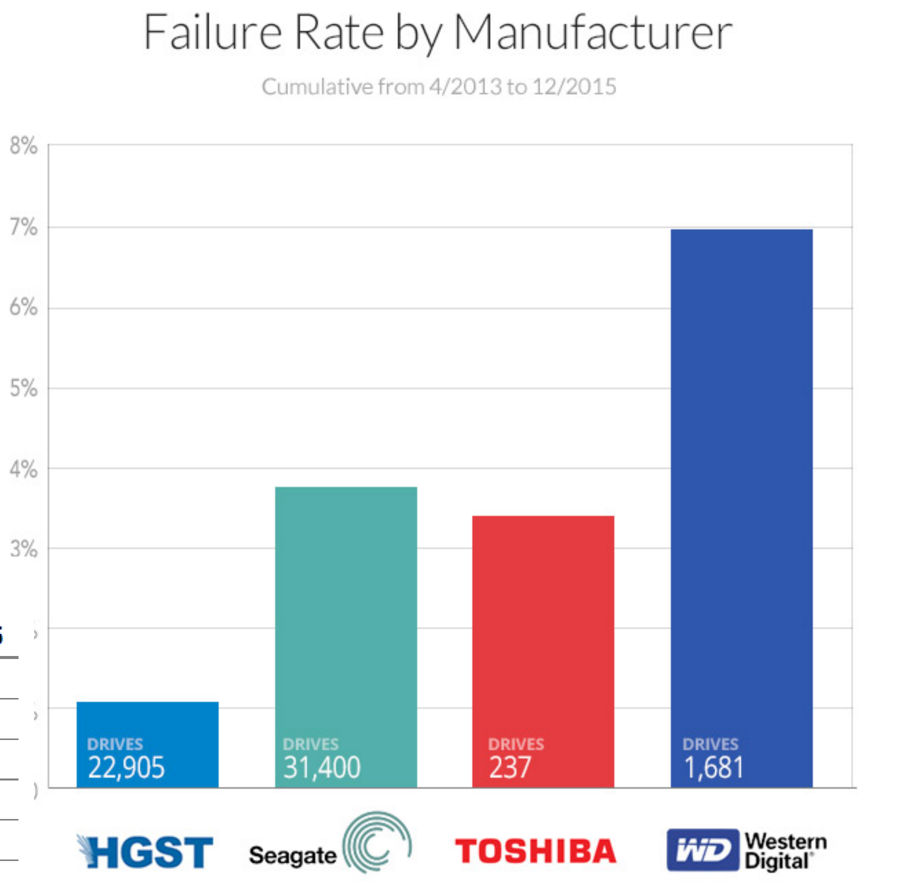
- Statistics for 56K disks of the Backblaze data center





Cumulative Failure Rate through the Period Ending

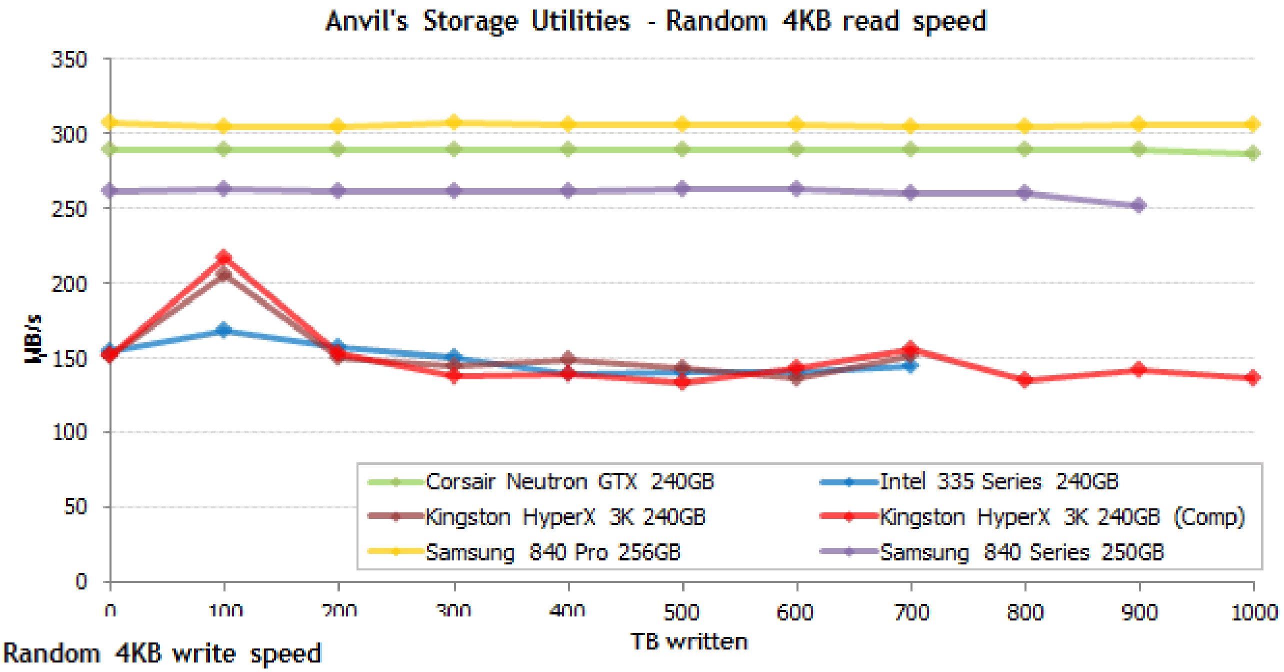
MFG	Model #	Highest QTY	12/31/13	12/31/14	12/31/15
HGST	HDS5C3030ALA630	4,596	0.9%	0.7%	0.8%
HGST	HDS723030ALA640	1,022	0.9%	1.8%	1.8%
Seagate	ST3000DM001	4,074	9.8%	28.3%	28.3%
Seagate	ST33000651AS	325	7.3%	5.6%	5.1%
Toshiba	DT01ACA300	58	-	4.8%	3.8%
WDC	WD30EFRX	1,105	3.2%	6.5%	7.3%

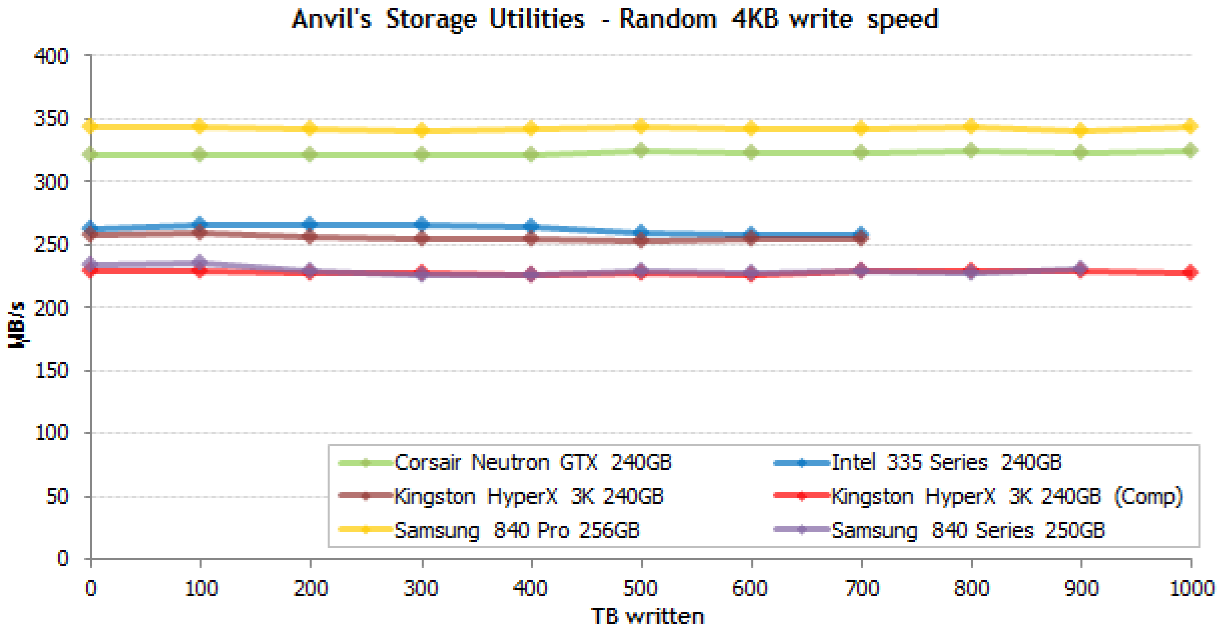


(HGST is the former Hitachi Global Storage Technologies)

BACKBLAZE

SSD reliability





50GB / day
→ > 40 years lifetime

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

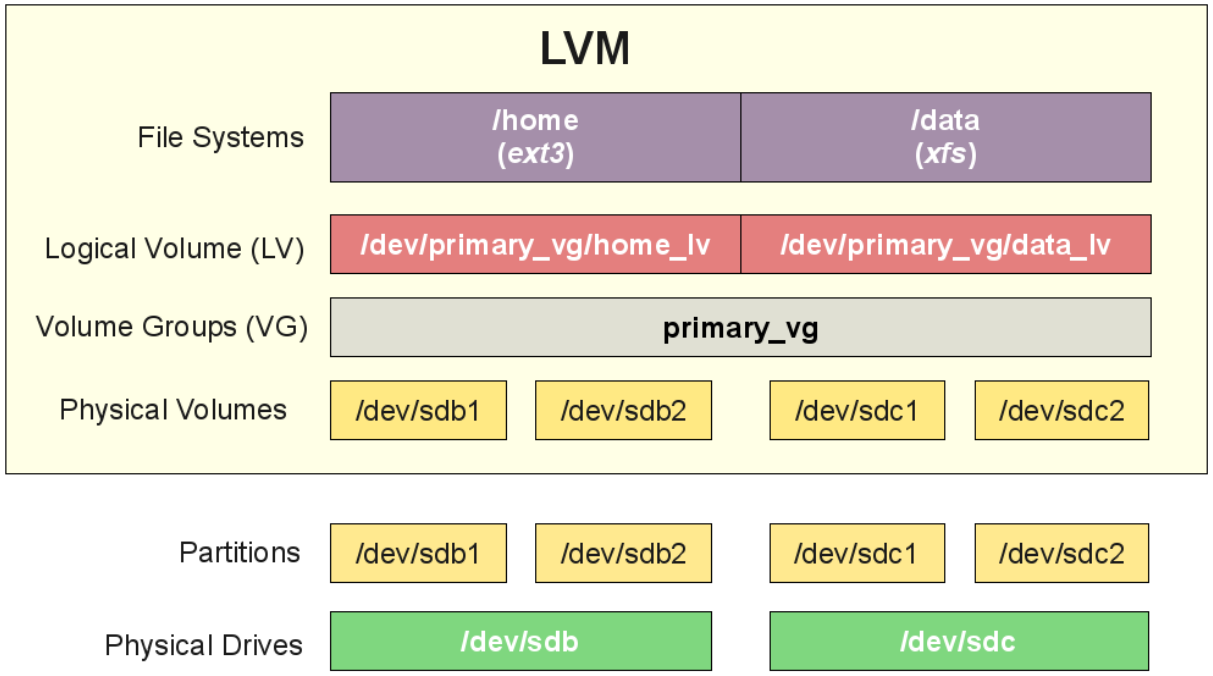
Virtual storage systems

- Overcoming the limits of physical storage solutions
  - capacity, performance, reliability
  - better management
  - better error recovery
  - unifying physical storage devices
- Virtual storage
  - implements a software storage layer that
    - is backed by other storage devices (physical or virtual)
    - unifies the underlying storage into a coherent system
  - examples: RAID, LVM (Linux), LDM (Windows)
  - they are also part of certain file system implementations (e.g. btrfs)

Virtual storage: Logical Volume Management

- An allocation system beyond the boundaries of the physical devices
  - more flexible management than partitions
  - logical volumes can be created from partitions, disks and other resources
  - e.g. Windows: Logical Disk Manager, Linux: Logical Volume Manager

- Parts of the LVM
  - **Physical volumes**: disks, partitions, ...
  - **Logical volumes** virtual partitions
  - **Volume group**: a set of LVs – virtual storage
  - Allocation units
    - **Physical extents**: parts of the PV-s
    - **Logical extents**: 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, RAID disks aren't cheap
  - goal: improve redundancy (reliability) and performance
  - HW and SW implementations
    - mainboard RAID implemented in software (cheap)
    - RAID boards: HW solution (expensive)

- Reliability
  - more disks mean more failures
    - 1 disk MTTF: 100 000 hours, 100 disk MTTF: 1000 hours (41 days)
    - how can we increase the reliability?
  - introducing redundancy
    - storing error correcting data
    - simple example: mirroring – storing the data twice
    - better: a parity bit can also detect a single error and correct it

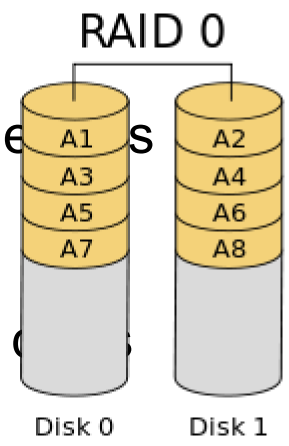
RAID levels: 0 - 1

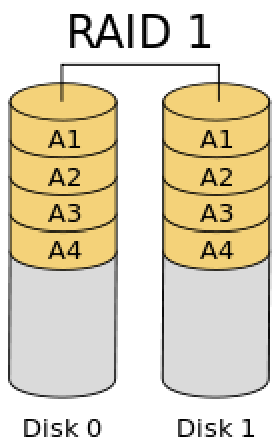
- RAID level: the mode of merging the physical de s
  - How the data is distributed on the N disks

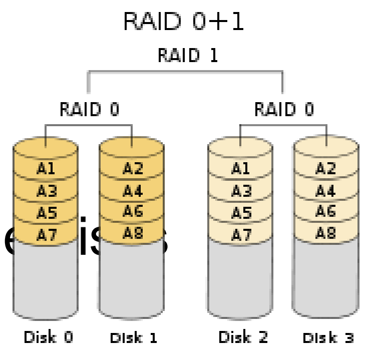
- RAID 0 (stripe): the data is distributed on the N (s
  - Goal: improve performance
  - It can increase the throughput and the latency also
  - The disks capacities are combined
  - Failure of 1 disk ☾ data loss

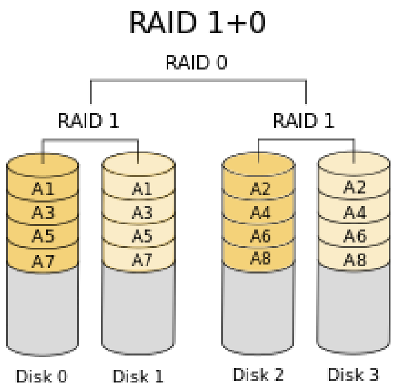
- RAID 1 (mirror): the same data are stored on multiple is s
  - Goal: improve reliability
  - The combined capacity is the size of a single disk
  - Slower write operations, read can be faster

- 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





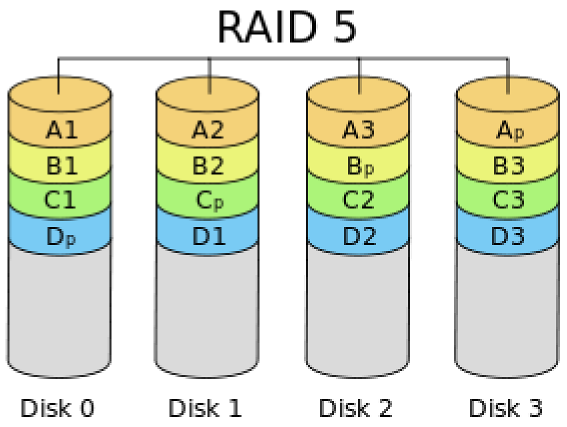




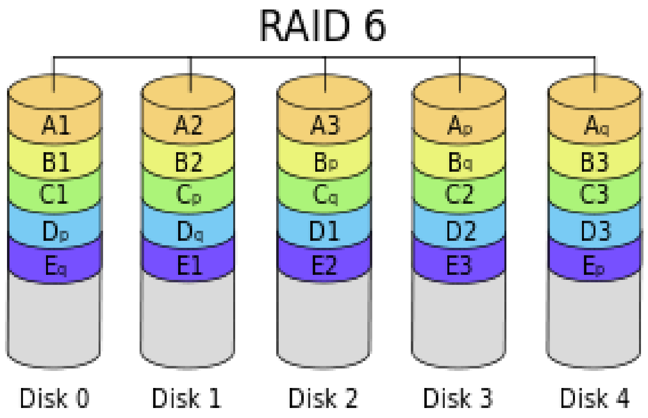
Widely used RAID levels

- 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

  Main problem: **silent error**



- 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
  - handles silent errors



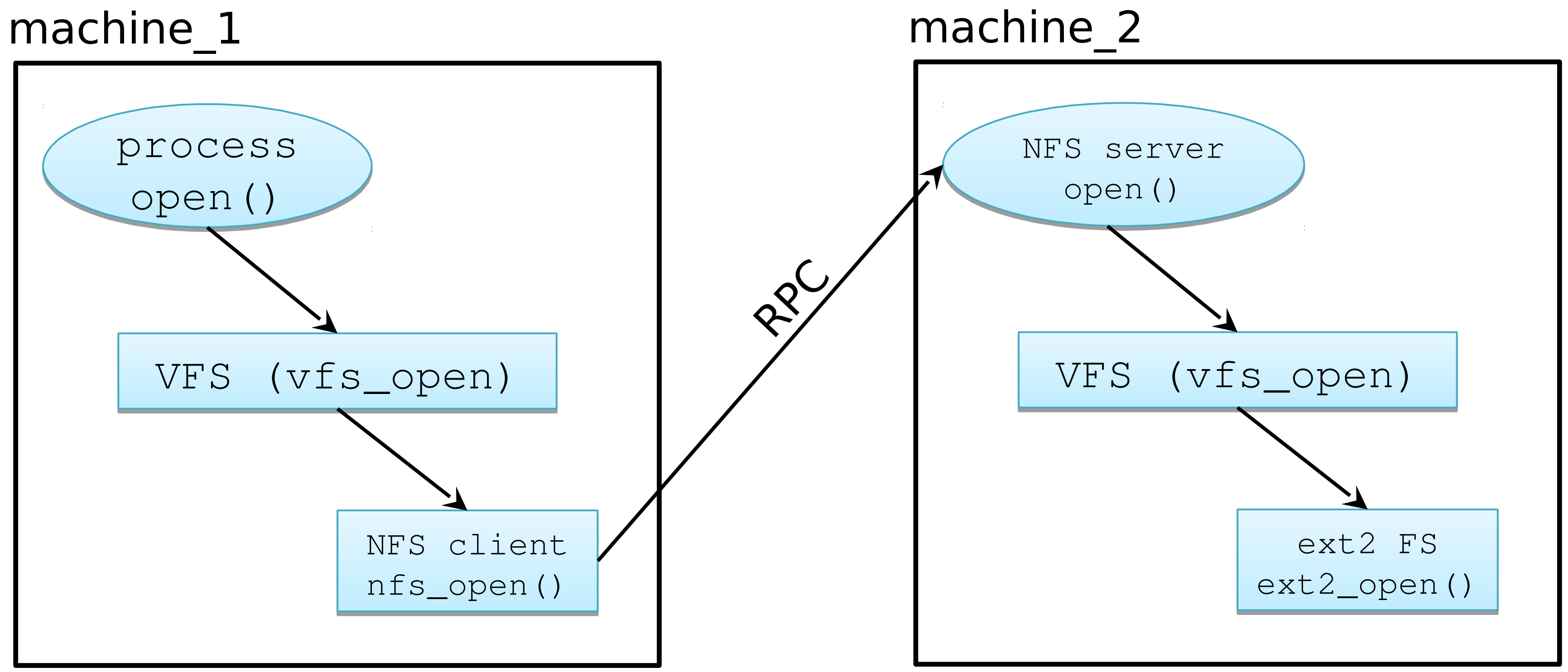
The limits of RAID systems

- How long does it take to correct an error (off-line)?
  - In the case of 4+1 disks (RAID5)
    - 150 GB disks: ~10 hours
    - 6 TB disks: ~80 hours
  - Spending days with error recovery is not acceptable
    - hot spares and RAID6 may improve the situation
- RAID needs the same type of disks
  - the replacement can be difficult
- RAID is a bonded structure, not flexible
  - cannot upgrade a RAID5 system to RAID6
  - transition takes a long time (off-line)
- Limited combined storage capacity
  - 6-8 disks at maximum
- RAID only protects against disk errors
  - What happens if the motherboard, CPU, RAM, power supply has an error?

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)**
    - 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 (RedHat, SUSE), Google GFS, RedHat GlusterFS,
    - Windows DFS, PVFS, Orange FS
- Challenges: latency, network errors, consistency

A simple implementation of NFS using VFS and RPC



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)