Data Storage Organization

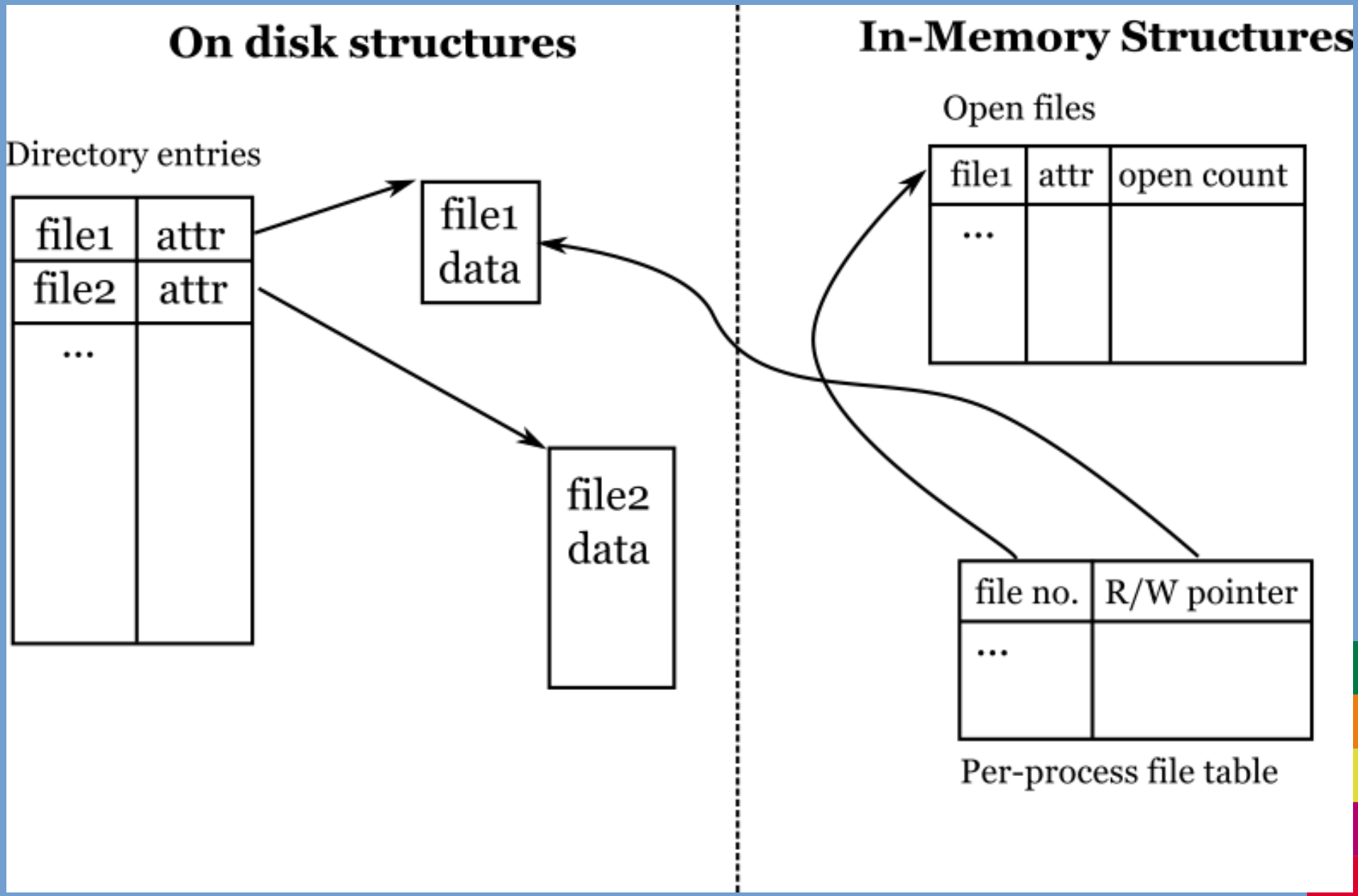
- It is clear we want to store data, in fact *lots* of it.
- We are given a disk which can hold several terabytes of data.
- How do we organize/structure this storage?

Filing Cabinet

Ikea ERIK

Directory Organization

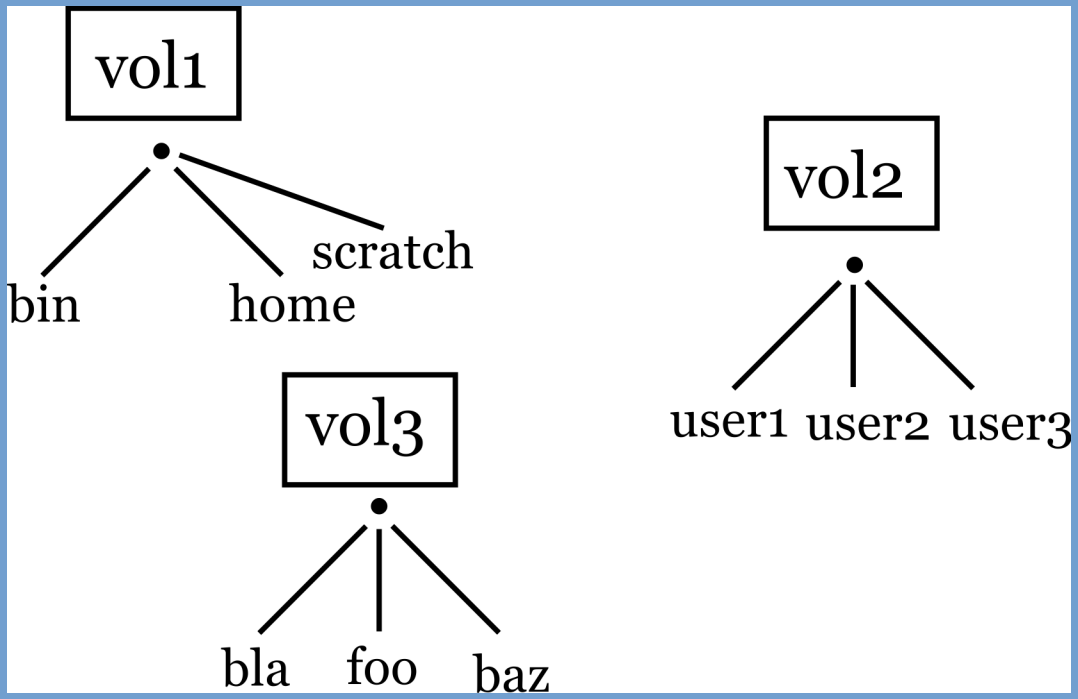
- Filing cabinet.
- Drawers ↔ Disks / volumes.
  - Folders ↔ Directories.
    - Sheets of paper ↔ Files


On disk structures
In-Memory Structures
Directory entries
file1
attr
file2
attr
...
file1
data
file2
data
Open files
file1
attr
open count
...
file no.
R/W pointer
...
Per-process file table

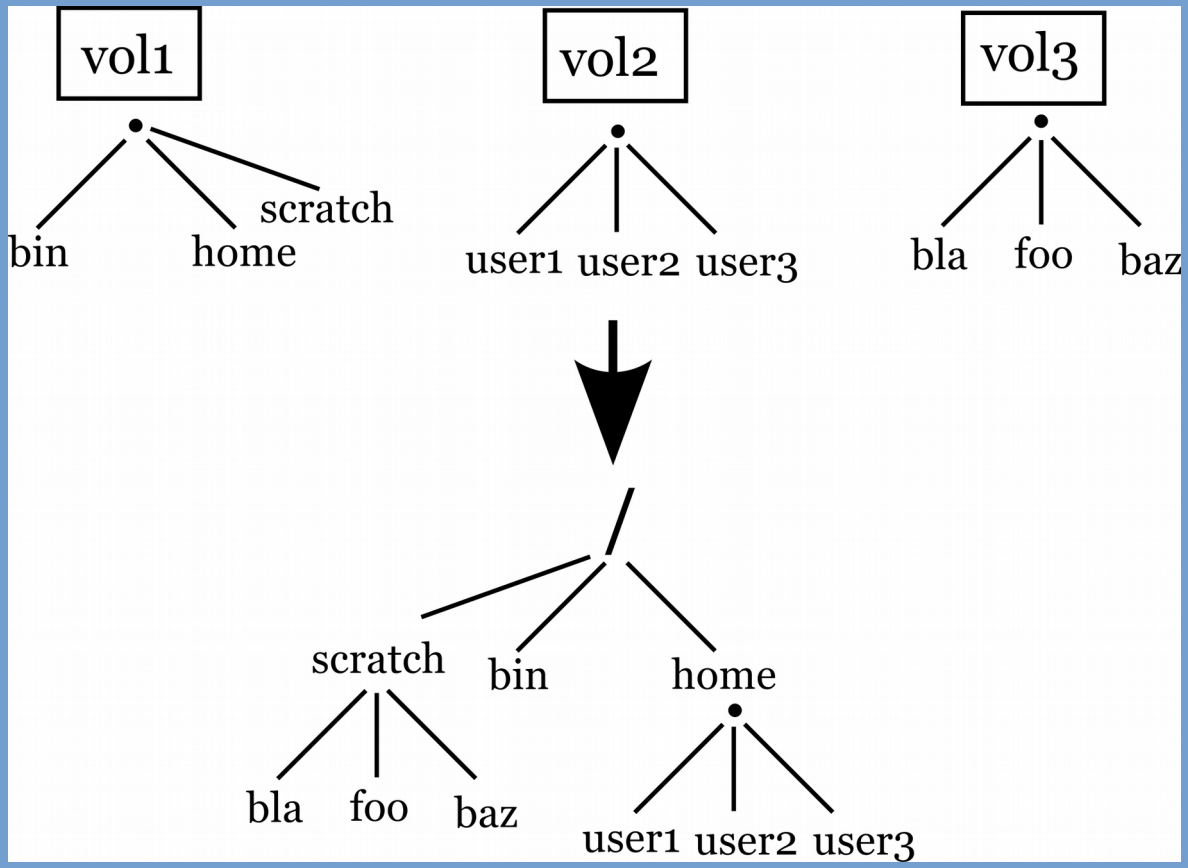
Link types

- Symbolic link
  - New directory entry type.
  - Contains file name (symbolic) of link target.
  - Following this pointer: “resolving the link”.
- Hard link
  - Duplicated directory entry.
  - Directory entries point at “inodes”.
  - In the “inode” a reference count is kept.

File System Mounting



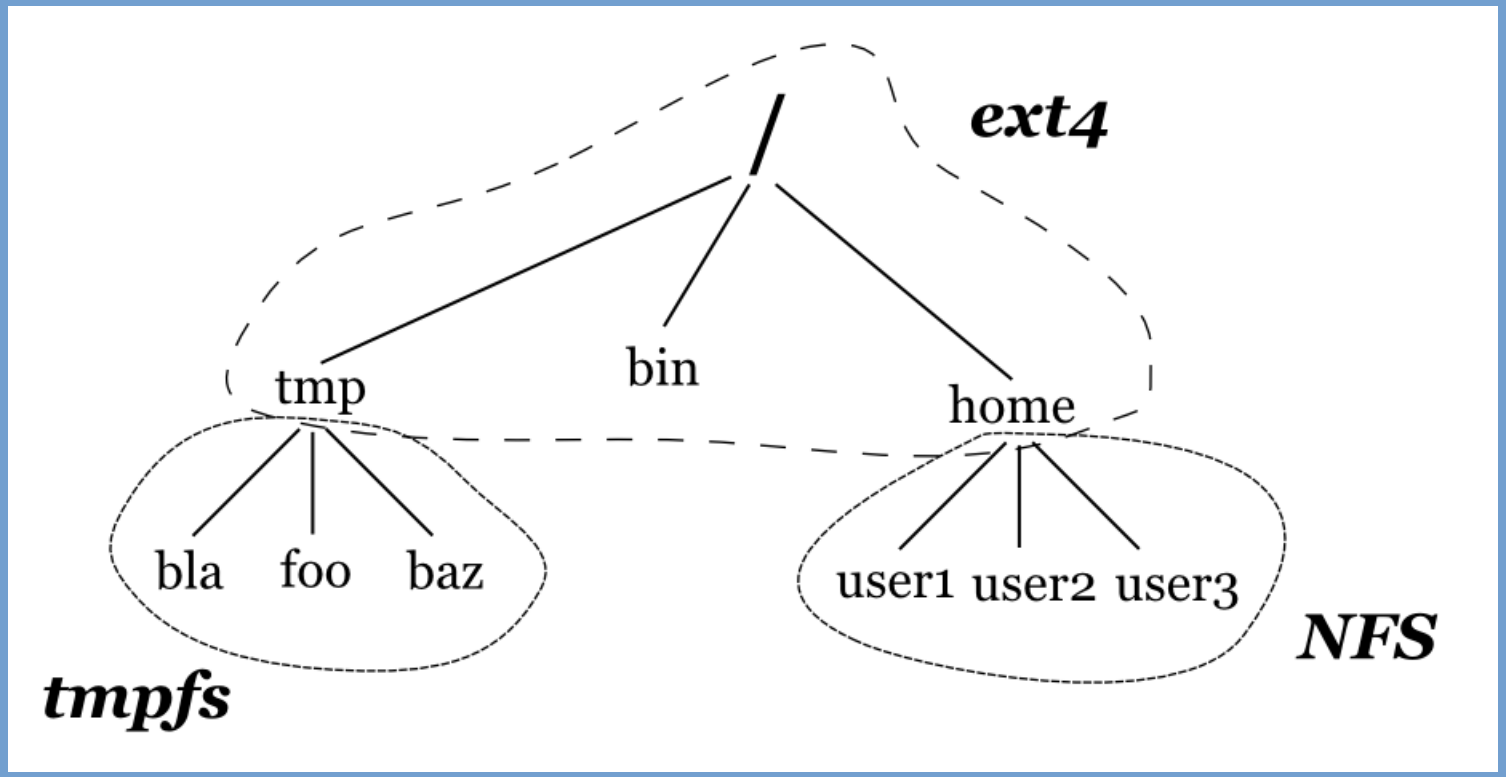
File System Mounting



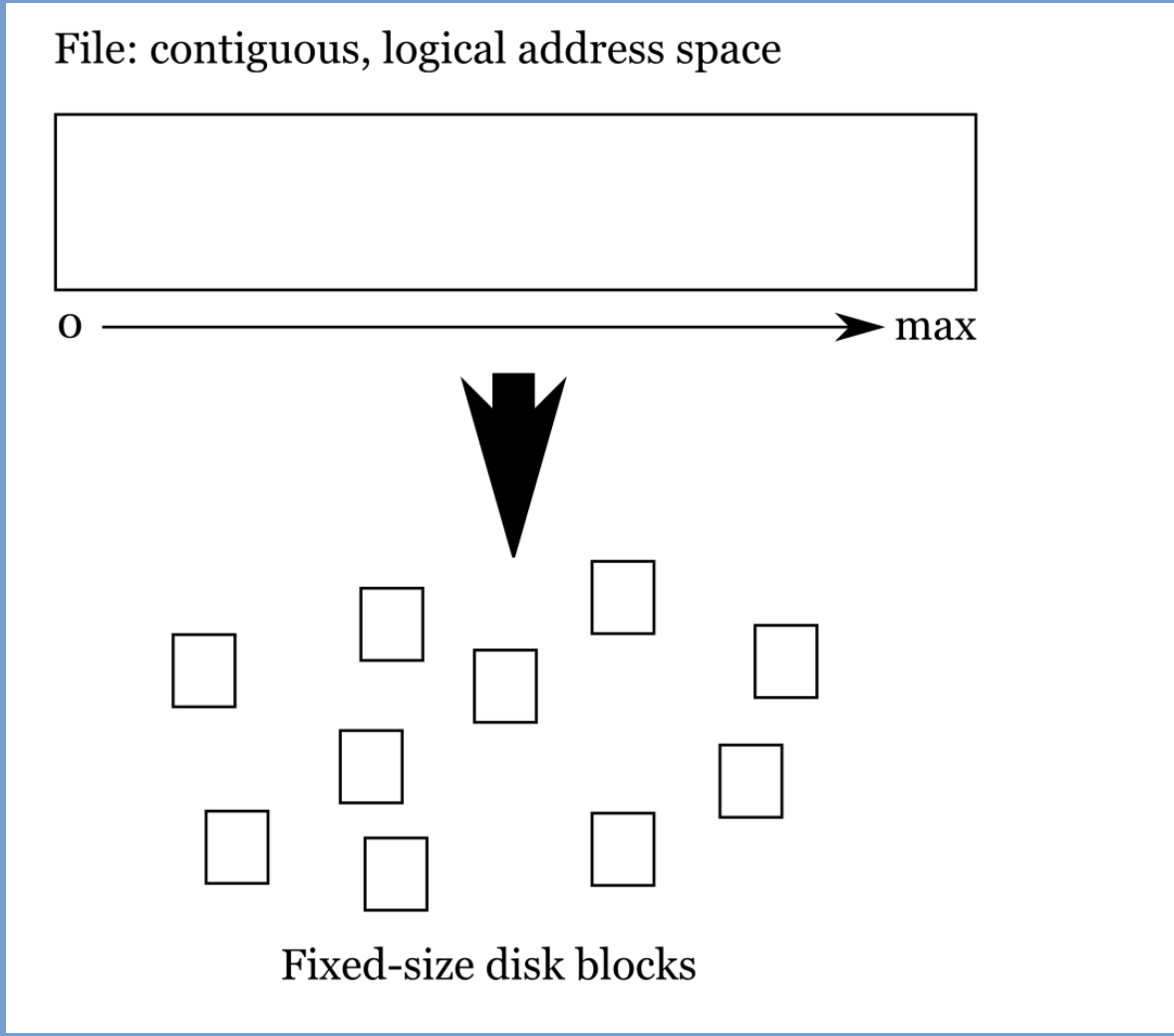
Chapter 11: File system implementation

Why Virtual File Systems?

- At different mount points, different **types** of file systems may be mounted:



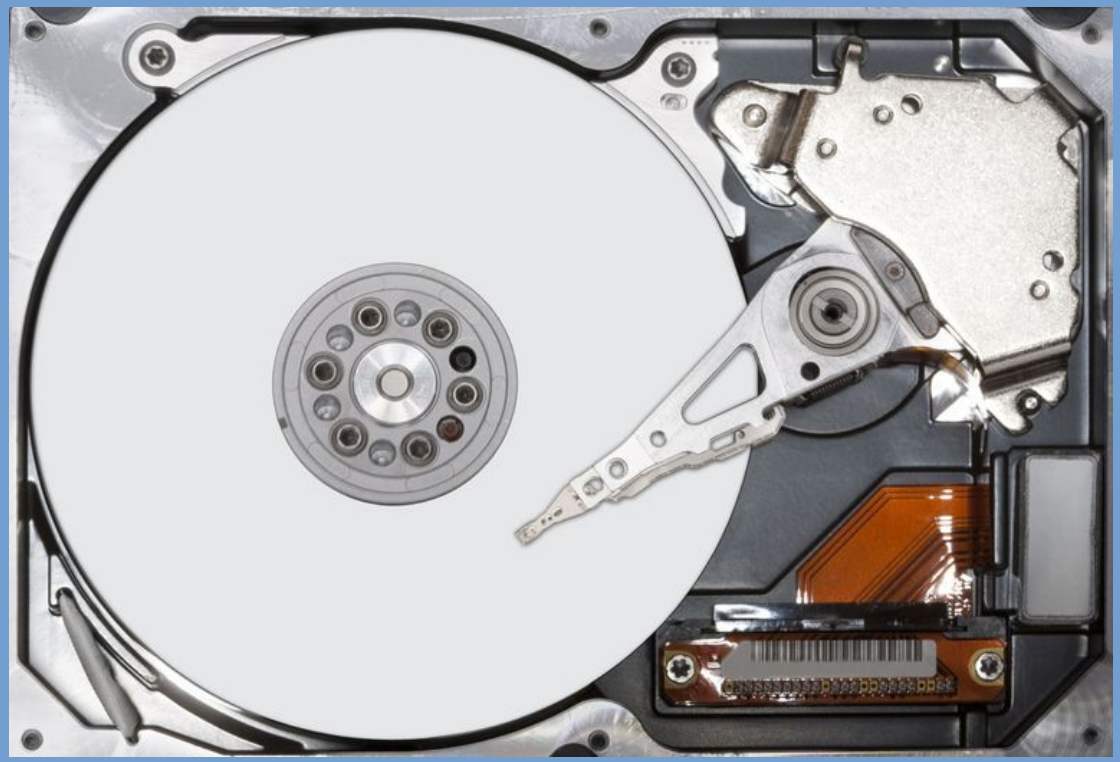
Disk Block Allocation



Chapter 12: Mass storage systems

Modern hard drives

- Available up to ~10 TB.

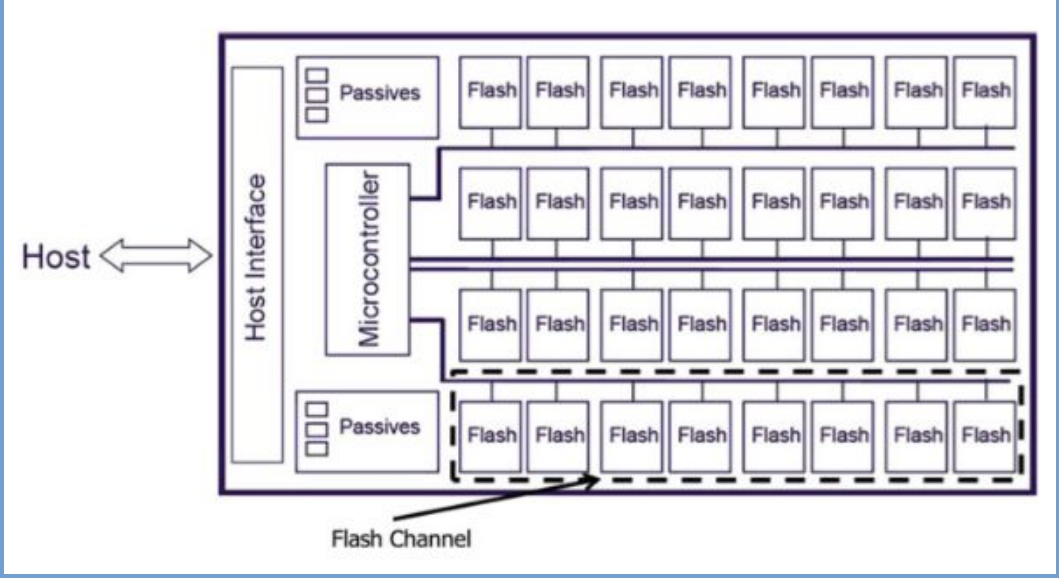


Solid State Disks

- These days available in different form factors, different buses (SATA, SAS, NVMe PCI).

SSD architecture

- SSDs are built from collections of flash memories.
  - The chips are organized into separate channels.
  - Communication on a channel can be interleaved, when one chip is busy, we can continue communicating with another chip on that channel.
- A microcontroller manages the flash memories and communication with the host.



Source: K. Eshghi and R. Micheloni, SSD Architecture and PCI Express Interface, Springer 2013.

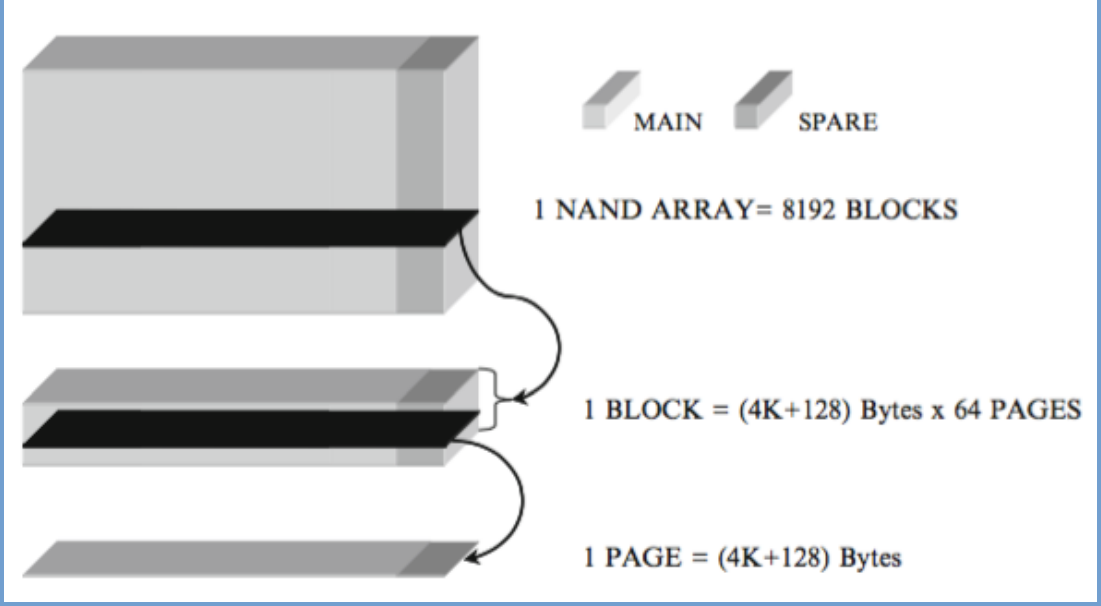
SSD architecture (2)

What is there to manage?

- Translation of host drive geometry (in particular for SATA SSD) to SSD architecture.
  - In fact, although the host thinks it is writing to contiguous blocks, this may not be the case as blocks are dynamically remapped by the SSD microcontroller.
- Wear leveling
  - Flash memory has a limited number of rewrite cycles before it breaks; so distribute the load uniformly.
- Bad block management
  - Maintain a list of bad blocks; already initialized during testing at the factory.

SSD architecture (3)

- The memories are grouped into *pages*.
- Pages are again combined into *blocks*.
- A collection of blocks forms the entire memory array.



Source: K. Eshghi and R. Micheloni, SSD Architecture and PCI Express Interface, Springer 2013.

SSD architecture (4)

- Read and write operations can be done at a page level.
  - (So not at the level of individual memories!).
- Now here's an interesting catch:
  - Before you can write a page, it MUST be empty.
  - If it's not empty, it must first be erased.
  - But: you can only erase entire blocks (!). (Hardware limitation)
- So, page overwrite implies:
  - Read the entire block into a buffer.
  - Erase the block.
  - Write all pages back.

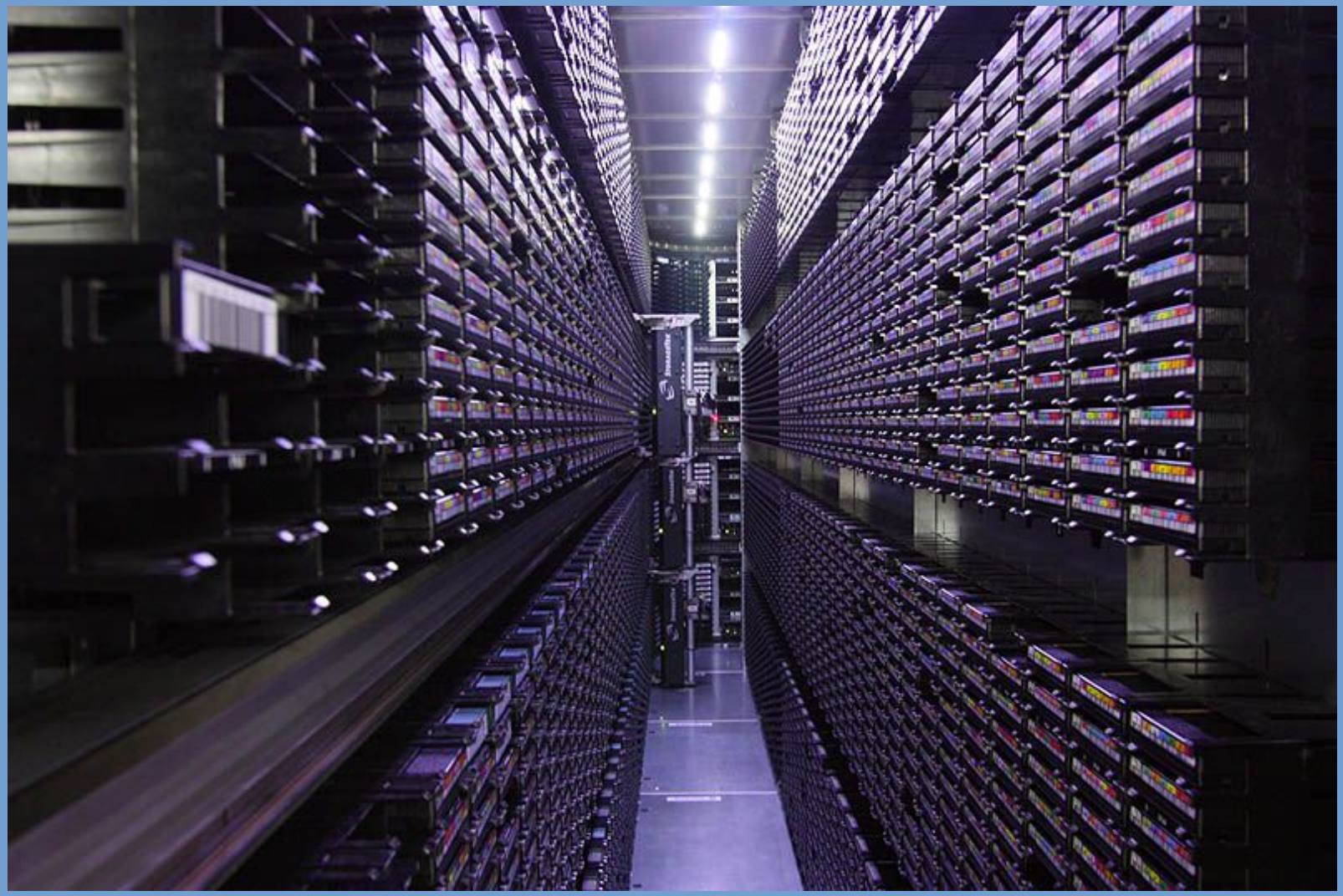
TRIM

- For each overwrite, many more write operations have to be done (write amplification).
- This is not a very good idea, considering the limited number of cycles.
- What happens if a file is deleted?
  - The OS only updates the file system data structures.
  - It can rewrite the blocks when a new file is allocated there, not a problem for magnetic disks.
  - What does this mean for SSD?

TRIM (2)

- Say we have a block of which all pages, except one, contain data of **deleted** files.
- If we overwrite a page on this block, all pages are rewritten, also these blocks that contain **deleted** data.
- To alleviate this, the TRIM command was added to the ATA command set (UNMAP in SCSI).
  - OS support is needed!
  - On file delete, the OS tells the SSD what blocks (pages on the SSD) were deleted. The SSD can then decide to no longer preserve this data in future overwrites of neighboring pages.

Tape silos



NFS vs. iSCSI

- NFS, CIFS, AFP

  → File-based systems. Remote host is accessed using requests for particular files. No knowledge about underlying file system.

- iSCSI

  → Block-based systems. Remote host is accessed using requests for particular disk blocks. Client determines file system.

Parity

- **Odd parity**
  - 0 1 1 0 → 0 1 1 0 1
  - Added parity bit causes number to the odd.
  - We can close any bit and still recover, why?
- **Even parity**
  - 0 1 1 0 → 0 1 1 0 0