# Kurs administrowania systemem Linux

## Zajęcia nr 15: Systemy plików, cz. 2

Instytut Informatyki Uniwersytetu Wrocławskiego

12 czerwca 2025

# Urządzenia dyskowe

**Zapis na nośniku fizycznym**

- magnetyczny na dysku: CMR, SMR (Shingled Magnetic Recording)
- w komórkach NAND flash: Solid State Drive (SSD)

**Kontroler dysku**

- odwzorowanie sektorów (sector reallocation, Flash Translation Layer)
- korekcja błędów odczytu (Reed-Solomon, Low Density Parity Check, Turbo Codes, BCH)
- diagnostyka (S.M.A.R.T.)
- obsługa magistrali dysku

**Magistrala dysku**

- SATA, NVMe, eMMC, SAS, Fiber Channel

**Host Controller**

- Host Controller Interface (SCSI, ATAPI, AHCI, NVMe, OHCI/UHCI/EHCI/xHCI)

**Device Driver**

- Tunelowanie np. w USB: SAT (SCSI/ATA Translation), UAS (USB Attached SCSI)

# Organizacja dysku

**Atomowa jednostka zapisu: sektor**

- Sektor logiczny: prawie zawsze 512 B (ale są też np. 520 B)
- LBA (Logical Block Addressing), numerowane od 0
- SATA: LBA ma 48 bitów (128 PB); dawniej 22 bity (2 GB), potem 28 bitów (128 GB)
- interfejs jądra używa 64 bitów (8 ZB); dawniej 32 bity (2 TB)
- Sektor fizyczny: Advanced Format (4 kB), flash page (4–8 kB)
- Większe obszary: SMR (strefy zapisu, zwykle 20–40 MB), SSD (erase blocks, zwykle 128–512 stron, tj. 512 kB – 4 MB)

**Wnioski**

- System operacyjny przedstawia dysk jako zbiór sektorów numerowanych za pomocą LBA.
- Wyrównanie (alignment) i lokalność przestrzenna ważne dla efektywności zarówno dla dysków magnetycznych, jak i SSD.

# Warstwy abstrakcji oferowane przez jądro Linuksa

**Partycje**

- Najstarsza metoda podziału dysku na „mniejsze dyski”.
- Obecnie popularny GPT. Dawniej MBR, BSD, Sun, SGI.

**Partycje logiczne: LVM2**

- Można dowolnie „rozcinać”, ale i „sklejać” dyski.
- Jeden z modułów sterownika `dm` (device mapper).

**Softraid**

- Sterownik `md` (multiple device driver)

**Szyfrowanie dysków**

- `dm-crypt`
- `dm-integrity`, `dm-verity`

# Sterownik dm

**Device mapper: ogólny mechanizm transformowania dysków**

- Liczne moduły (poza wymienionymi też `cache`, `delay`, `linear`, `striped`, `error`, `zero`, `flakey`, `mirror`, `multipath`, `snaphot`, `zoned` i in.).
- Sporo starego kodu, który istnieje poza `dm`:
  - partycje (`acorn`, `aix`, `amiga`, `atari`, `efi`, `ibm`, `karma`, `ldm`, `mac`, `msdos`, `osf`, `sgi`, `sun`, `sysv68`, `ultrix`)
  - osobny driver `md`

**Rozwiązanie we FreeBSD: GEOM**

- Struktura obiektowa (moduły — klasy)
- producenci → obiekt GEOM → konsumenci
- klasy implementują wszystkie dostępne transformacje dysków

# Utrata danych z dysku

**Przyczyny**

- Nieatomowość zapisu na dysk
- Uszkodzenie dysku

**Rozwiązanie problemu atomowości (metadanych)**

- Weryfikacja metadanych (`fsck`)
- Księgowanie (*journalling*)
- *Soft updates*
- COW (*copy on write*)

**Rozwiązanie problemu uszkodzeń dysków: redundancja**

- hardware RAID
- soft RAID

Uwaga: RAID zapewnia HA, ale nie zastępuje backupów!

# Uszkodzenia dysków

**Rodzaje nieprawidłowego działania**

- całkowita awaria (dysk nie odpowiada)
- *bad sectors* (dysk zgłasza błędy odczytu)
- *silent errors* (*bit rotting*)

**Bit rotting**

- Teoretycznie niemożliwe, bo dysk używa sum kontrolnych!
- Częsty powód: błędy w oprogramowaniu dysku.

**Jak się zabezpieczyć przed „gniciem bitów”?**

- RAID nie pomaga
- Sumy kontrolne jedynym sprawdzonym rozwiązaniem
- `ext4` ma sumy kontrolne metadanych; ZFS — wszystkiego
- *scrubbing* — wczesne wykrywanie problemów

# Redundant Array of Independent (or Inexpensive) Disks

**RAID levels**

- **RAID 0** — block level stripes: $n\times$ zapis, $n\times$ odczyt, redundancja: 0, pojemność: 1
- **RAID 1** — mirrors: $1\times$ zapis, $n\times$ odczyt, redundancja: $n-1$, pojemność: $1/n$
- **RAID 5** — block-level stripes with distributed parity: $(n-1)\times$ zapis, $n\times$ odczyt, redundancja: 1, pojemność: $1-1/n$
- **RAID 6** — block-level stripes with double distributed parity: $(n-2)\times$ zapis, $n\times$ odczyt, redundancja: 2, pojemność: $1-2/n$

**Stare, niepopularne wersje**

- **RAID 2** — bit level stripes with Hamming code: bardzo duże szybkości transmisji, praca synchroniczna (tylko jedna transakcja na raz), kody Hamminga nie mają dużej przewagi nad bitami parzystości
- **RAID 3** — byte level stripes with parity
- **RAID 4** — block level stripes with parity

# RAID

**Poziomy zagnieżdżone**

- **RAID 01** — mirror of stripes
- **RAID 10** — stripes of mirrors
- **RAID 03** — byte-level stripes with parity of stripes
- **RAID 50** — block-level stripes of block-level stripes with distributed parity
- **RAID 60** — block-level stripes of block-level stripes with double distributed parity
- **RAID 100** — stripes of stripes of mirrors

**Poziomy niestandardowe**

- wiele niestandardowych typów

# Zalety i wady hardware RAID

**Zalety**

- System operacyjny „widzi" macierz jako pojedynczy dysk (nie potrzeba oprogramowania, nie ma kłopotów z bootowaniem itd.).
- Obsługa RAID nie obciążą procesora.
- Podtrzymanie bateryjne — kontroler RAID potrafi dokończyć transakcję mimo utraty zewnętrznego zasilania.

**Wady**

- Mniej elastyczne, niż rozwiązania software'owe.
- System operacyjny nie jest świadom działania macierzy, nie może planować ułożenia danych na dyskach.

**Softraid: sterownik md i interfejs dm-raid**

- Zaimplementowany w jądrze Linuksa.
- Nie potrzebuje wsparcia sprzętowego.
- Oferuje RAID 0,1, 4, 5, 6, 10.
- Konfiguracja: `mdadm(8)`.

# ext4 — podstawowy system plików Linuksa

**Systemy plików często działają latami**

- Każdy system plików jest piękny, gdy jest młody.
- Wiele cykli zapisu i kasowania (total writes przekracza wielokrotnie pojemność dysku) — praca alokatora jest bardzo ważna!
- Systemy plików starzeją się w różny sposób (jak wino lub jak mleko).
- `ext4` jest jednym z najlepszych klasycznych systemów plików.

**Awarie**

- Bardzo dobry `fsck`.
- Bardzo dobre księgowanie (JBD2) — na poziomie bloków.
- Od niedawna sumy kontrolne metadanych i scrubbing.
- Snapshotting — obecnie poprzez LVM2.

# RAID a system plików

**Wady stosu protokołów md/dm/lvm2/ext4**

- RAID nie wie, które sektory są w użyciu, a które nie.
- W razie niespójności RAID nie wie, która wersja jest prawdziwa (nie radzi sobie z gniciem bitów).
- System plików nie wie, na który z fizycznych dysków dane będą zapisane.

**Rozwiązanie**

- Połączyć system plików z *volume managerem*.

# ZFS: The Z File System

- Prace rozpoczęto w 2001 w Sun Microsystems.
- Ogłoszono 14/09/2004, pierwsza wersja 31/10/2005 (Open Solaris).
- Licencja CDDL (open source, ale niekompatybilna z GPL i BSD).
- W 2010 Oracle zamknęło kod.
- Forki open source dalej rozwijane niezależnie.
- Paweł Jakub Dawidek: port do FreeBSD 7 (2008).
- FUSE dla Linuksa (2006), natywny port do Linuksa: ZoL (2008).
- Illumos: fork OpenSolarisa z ZFS.
- Inicjatywa OpenZFS (2013).
- FreeBSD 13 (2021): rebase z Illumosa na OpenZFS. Teraz Linux i FreeBSD mają ten sam upstream.

# Założenia projektowe

- „The Enterprise's computer on Star Trek probably runs ZFS" (Michael W. Lucas).
- Liczniki nie 64-bitowe (16 EB, jak w ext4 i Btrfs), ale 128-bitowe. System plików może mieć $288 \times 10^{15}$ ZB = 288 biliardów ZB = 256 PiZiB.
- Każdy blok (zarówno danych, jak i metadanych) posiada sumy kontrolne.
- Transakcyjność poprzez COW.
- Zalety COW: tanie snapshoty, klony i mirrory, duża lokalność zapisów.
- Wady COW: mała lokalność odczytów.
- Volume manager pozwala na striping, mirroring i stripes with distributed parity (redundancja od 1 do 3 dysków) — odpowiedniki RAID 0, 1, 5 i ich kombinacje.
- Gnicie bitów: problemy w razie uszkodzenia RAM, dlatego zaleca się ECC.
- Wiele niezależnych systemów plików i urządzeń blokowych w jednej przestrzeni dyskowej.
- Kłóci się z hardware RAID.

# Trwałe struktury danych

**Idea trwałych struktur danych**

- Neil Sarnak, Robert E. Tarjan, Planar point location using persistent search trees, CACM 29(7):669–679, July 1986.
- James R. Driscoll, Neil Sarnak, Daniel D. K. Sleator, Robert E. Tarjan, Making data structures persistent. J. Comp. System Sci. 38(1):86-124, February 1989.

**Copy on Write**

- COW na drzewach jest bardzo efektywny
- Implementacje: ZFS, Btrfs

# Virtual devices (vdev)

**Rodzaje vdev-ów**

- disk
- file
- mirror
- raidz1, raidz2, raidz3
- log (ZIL SLOG)
- cache (Level 2 Adaptive Replacement Cache, L2ARC)

Ponadto:

- spare
- draidz1, draidz2, draidz3; non-default draid (hot spare)
- dedup
- special

Przykład (odpowiednik RAID 10):

```
zpool create mypool mirror da0 da1 mirror da2 da3
```

# Działanie ZFS-a

**Datasety i zvole**

- Dataset — system plików umieszczony na pewnym zpoolu.
- Mountpoint: domyślnie `/zpoolname/datasetname/`.
- Można mieć datasety niemontowalne.
- Zvol — urządzenie blokowe umieszczone na pewnym zpoolu.
- Uwaga: nie konfigurować swap-a jako zvola!

**Administrowanie ZFS-em**

- ZFS przypomina SystemD: zamiast kompatybilnie dodawać — zastępuje.
- Administrowanie LVM-em zrobione na wzór ZFS.
- Zarządzanie pool-ami: polecenie `zpool`.
- Zarządzanie datasetami: polecenie `zfs`.

# Administrowanie ZFS-em we FreeBSD — zpoole

- `geom disk list`
- `camcontrol devlist`; CAM (Common Access Method) — tylko dyski SCSI
- `gpart show`; dokładniej: `gpart list`
- `diskinfo -v <dysk>`
- `zpool create -m none tank da0 da1 da2 da3`
- `zpool list`
- `zpool status tank`
- `zpool destroy tank`
- `zpool create -m none tank mirror da0 da1`
- `zpool add tank mirror da2 da3`
- `zpool online tank da2`
- `zpool scrub tank`

# Administrowanie ZFS-em we FreeBSD — datasety, snapshoty i klony

- `zfs create -o mountpoint=/test1 tank/test1`
- `zfs list`
- `zfs create tank/test1/test2`
- `zfs snapshot tank/set1@snap1`
- `zpool set listsnapshots=on tank`
- `zfs clone tank/set1@snap1 tank/set2`
- `zfs set mountpoint=/nowy tank/stary`

- `beadm list`

# Administrowanie ZFS-em we FreeBSD — mirrory i szyfrowanie

- `zfs send tank/set1@snap1 > snap1`
- `cat snap1 | zfs receive barrel/set1`

- `zfs create -o encryption=on -o keylocation=prompt -o keyformat=passphrase pool/dataset`
- `zfs get encryption pool/dataset`
- `zfs load-key -r pool/dataset; zfs mount pool/dataset`
- albo: `zfs mount -l pool/dataset`
- `zfs unload-key -r pool/dataset`
- `zfs get keystatus pool/dataset`