ZFS

Linuxadministration 1 1DV417

Överblick

- Lagringspooler
- Transaktionsbaserat objektsystem
- Dataintegritet
- Enkel administration

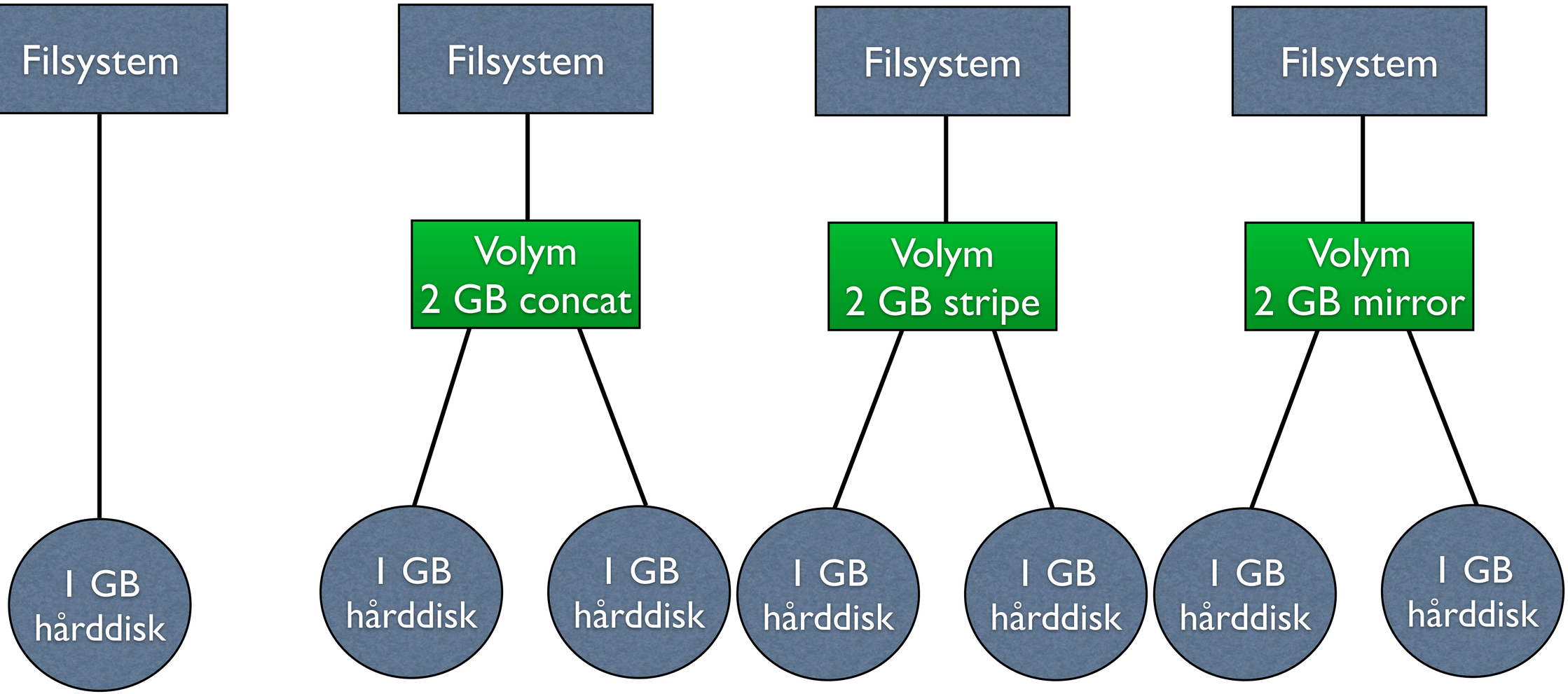
Problem med nuvarande filsystem

- Inget skydd mot "silent data corruption"
- Jobbigt att administrera
- Lååångsamt

Mål med ZFS

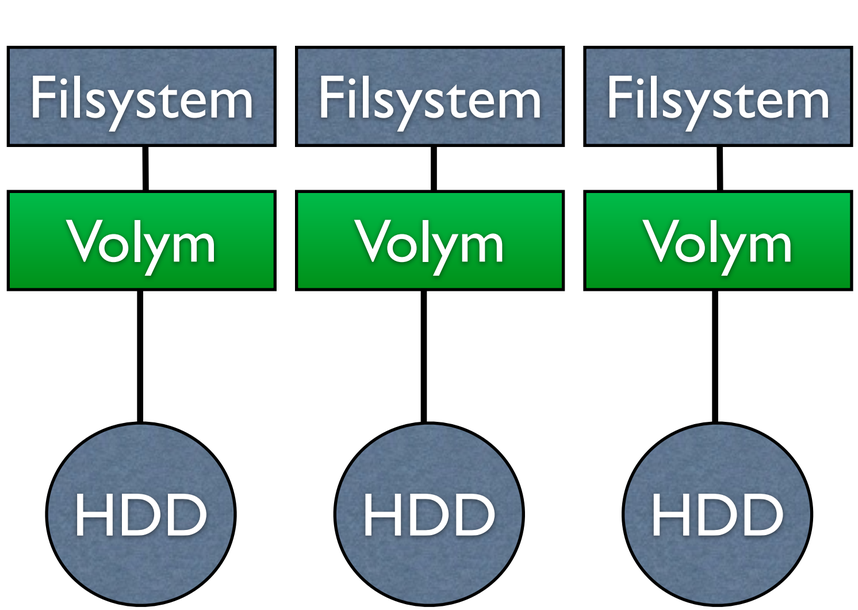
- Få slut på lidandet
- Varför behöver lagring vara så komplicerat?
- Kasta ut alla gamla antaganden
- Designa ett integrerat filsystem från början

Varför volymer?

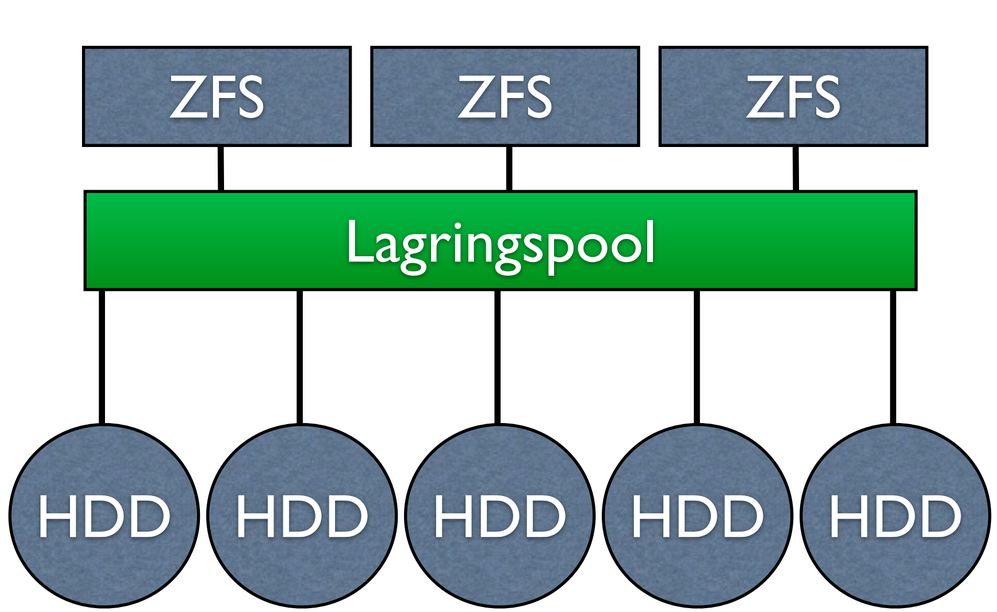


Volymer jämfört med pooler

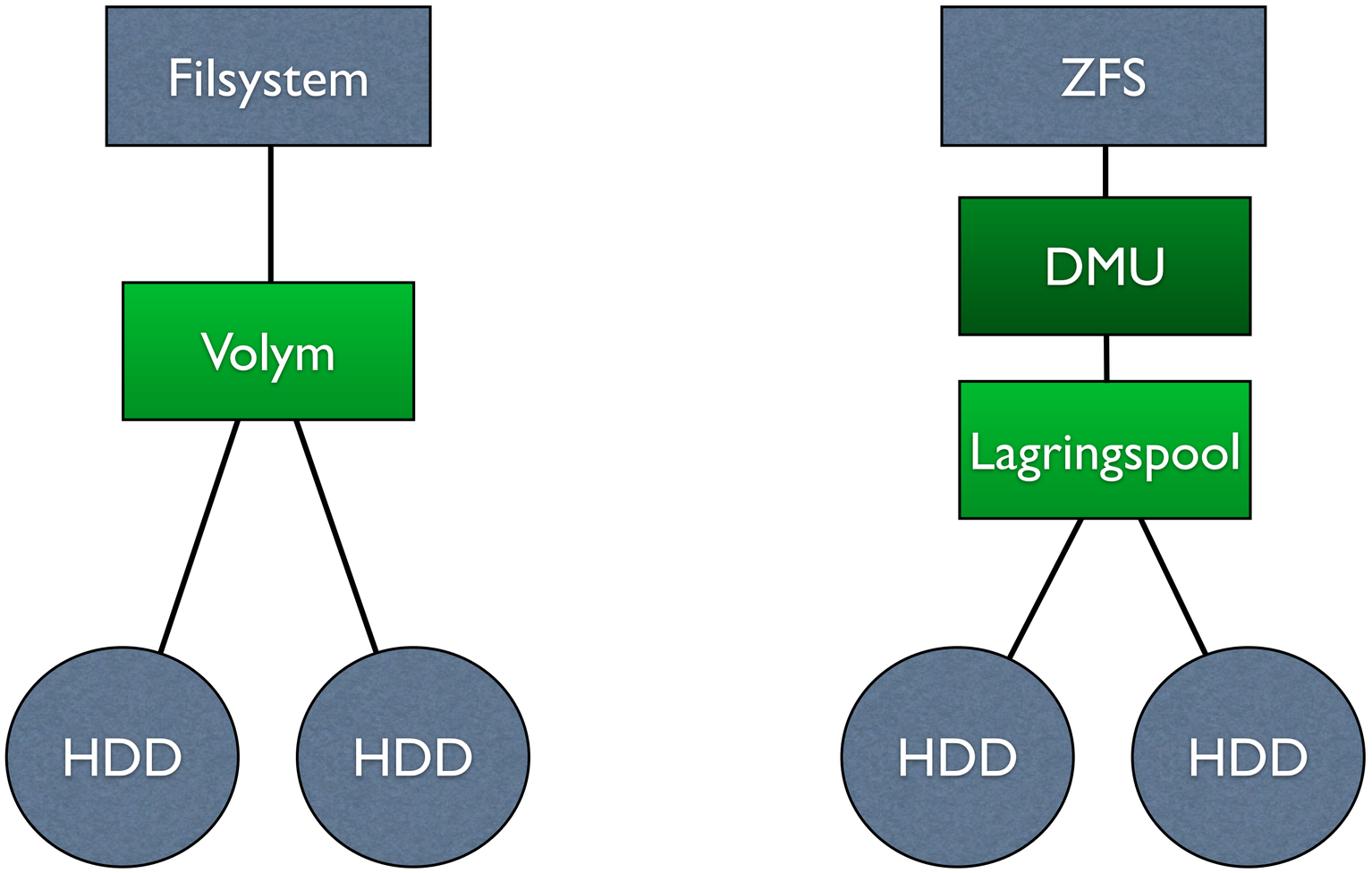
Traditionella volymer



Lagringspooler med ZFS



I/O i filsystem/volym vs. ZFS

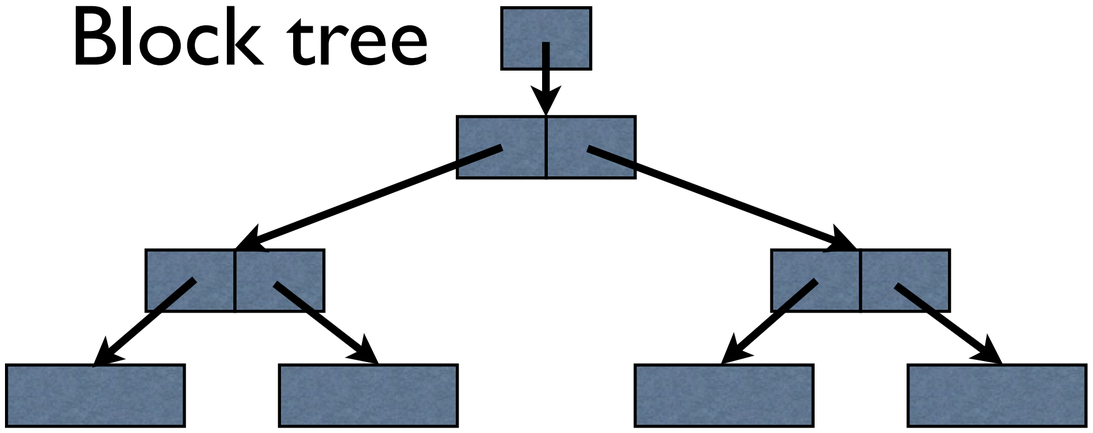


Dataintegritet

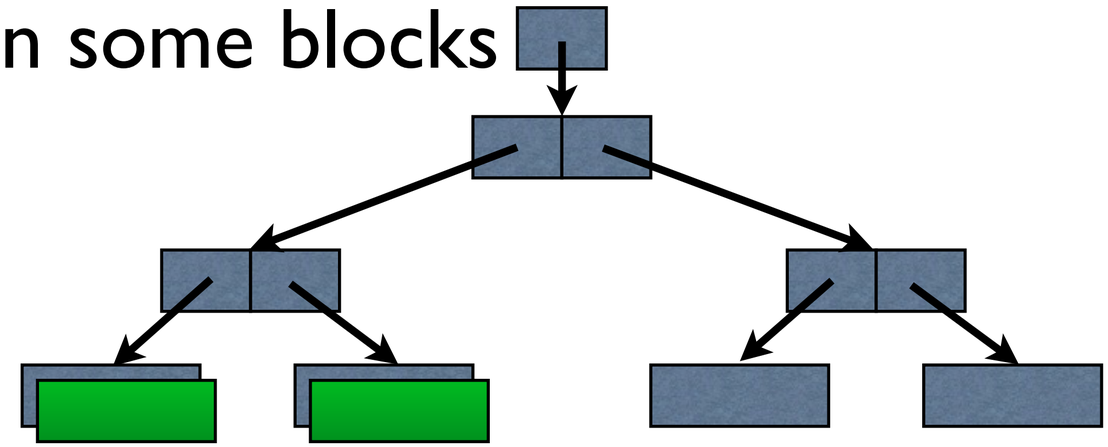
- Copy-on-write
- Transaktionsbaserat
- Checksummor på ALLT

Copy-on-write

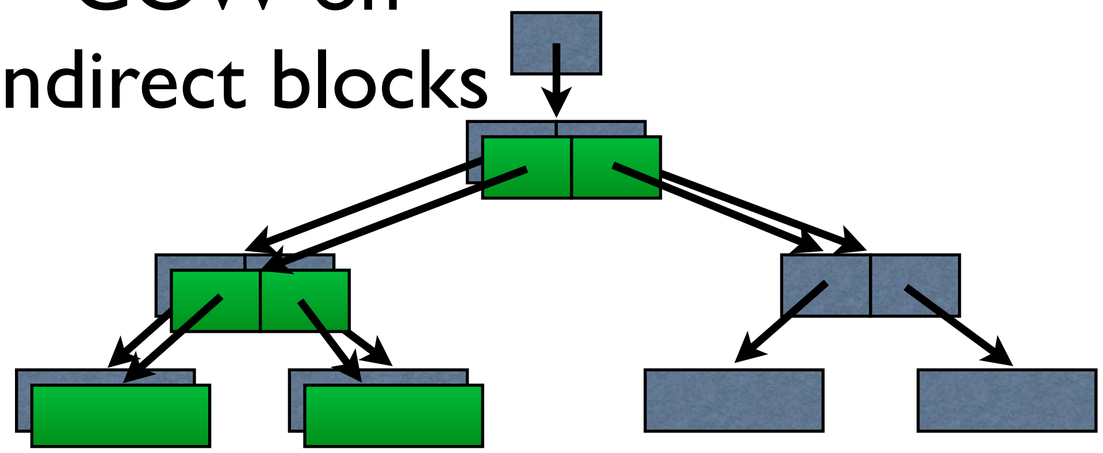
Block tree



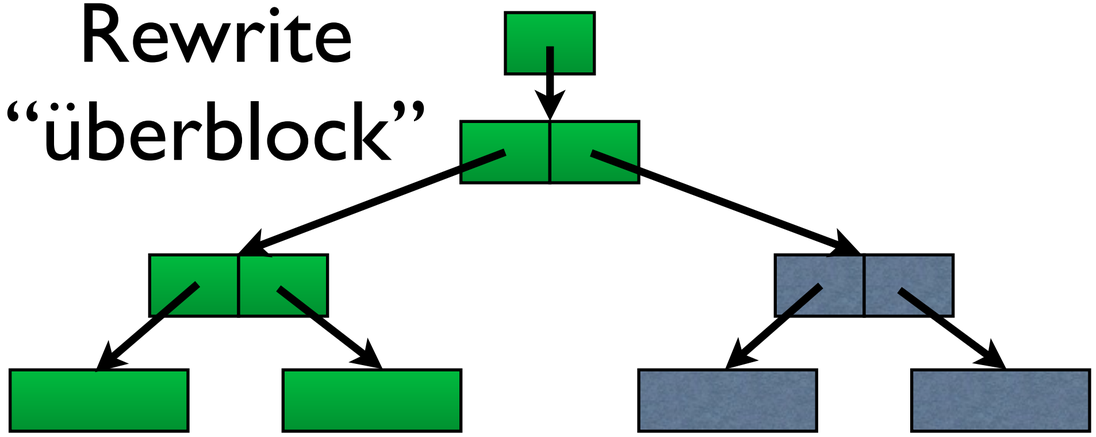
COW on some blocks



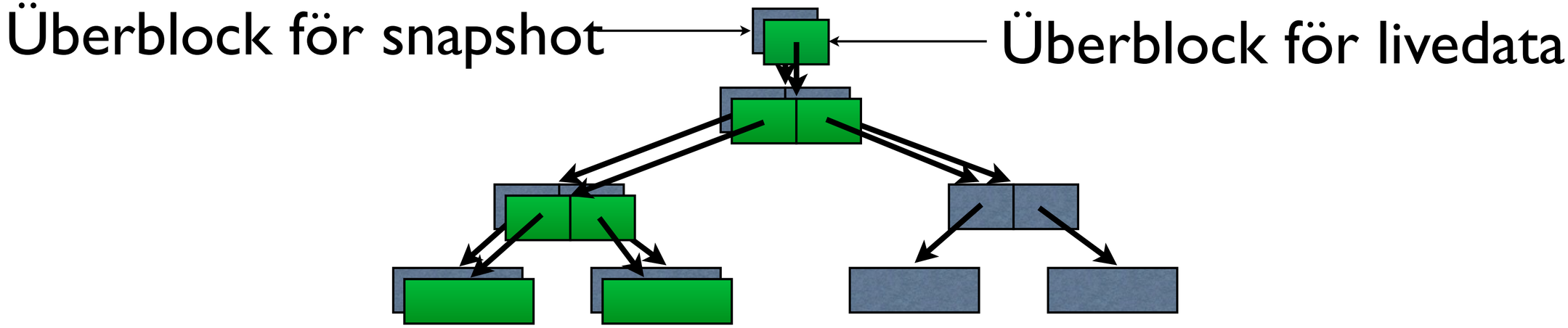
COW on indirect blocks



Rewrite "überblock"

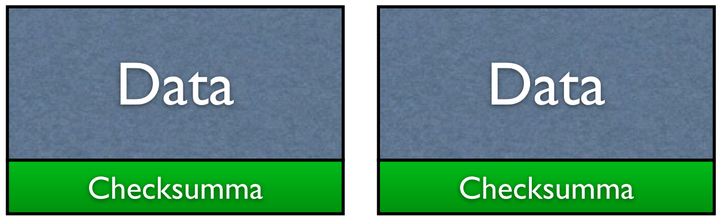


Snapshot-bonus



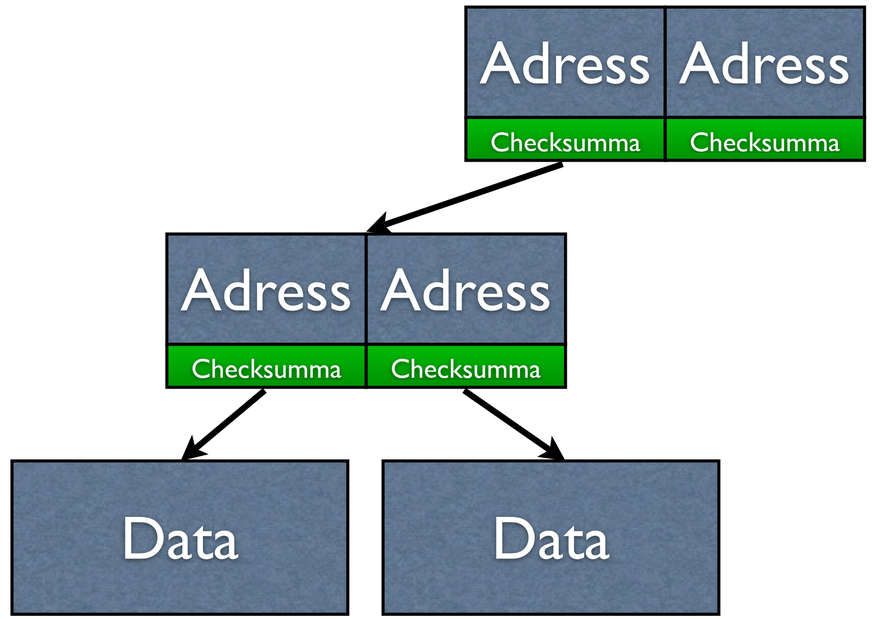
Checksummor på allt

Vanliga checksummor



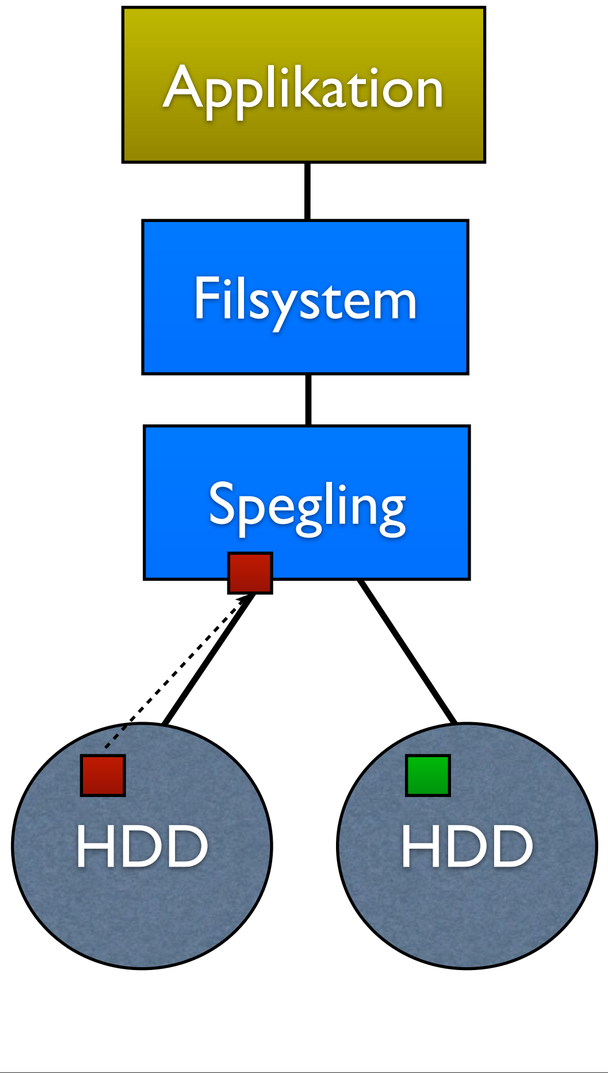
- Bitröta
- Fantomskrivningar
- Feldirigerade skrivningar och läsningar
- Felaktig paritet i DMA
- Drivrutinsbuggar
- Ofrivilliga överskrivningar

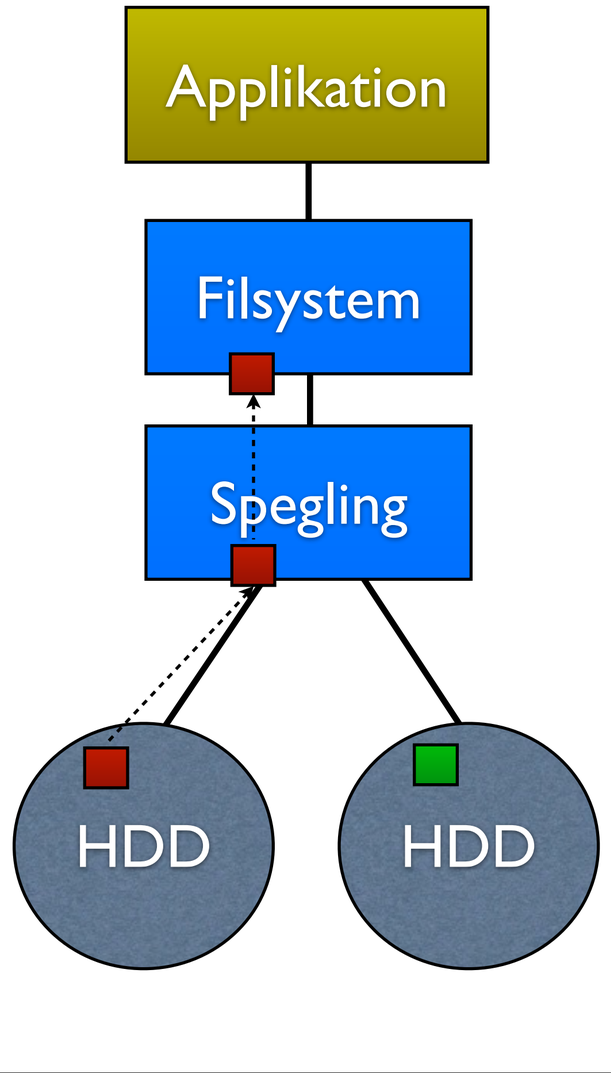
Checksummor i ZFS

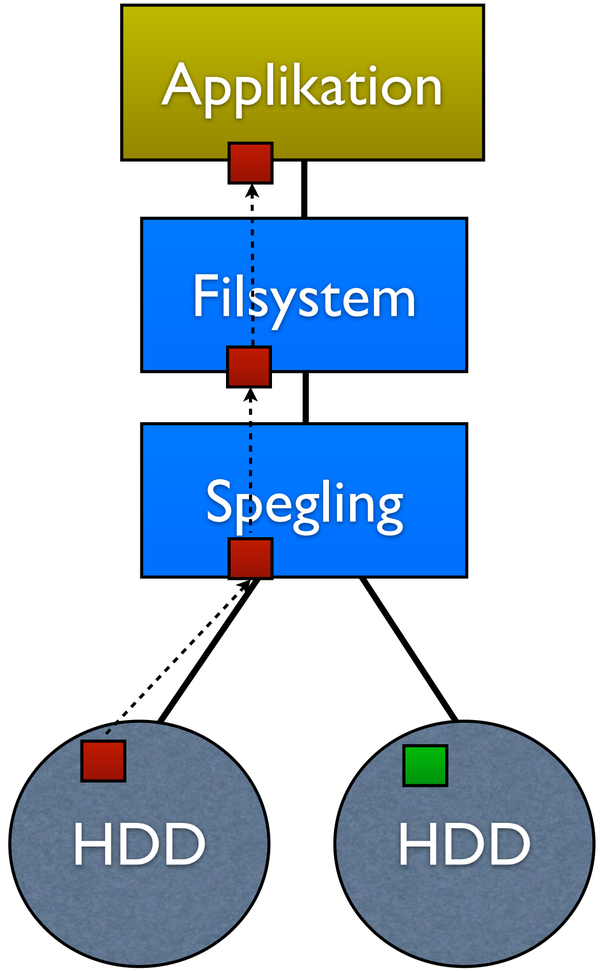


- Bitröta
- Fantomskrivningar
- Feldirigerade skrivningar och läsningar
- Felaktig paritet i DMA
- Drivrutinsbuggar
- Ofrivilliga överskrivningar

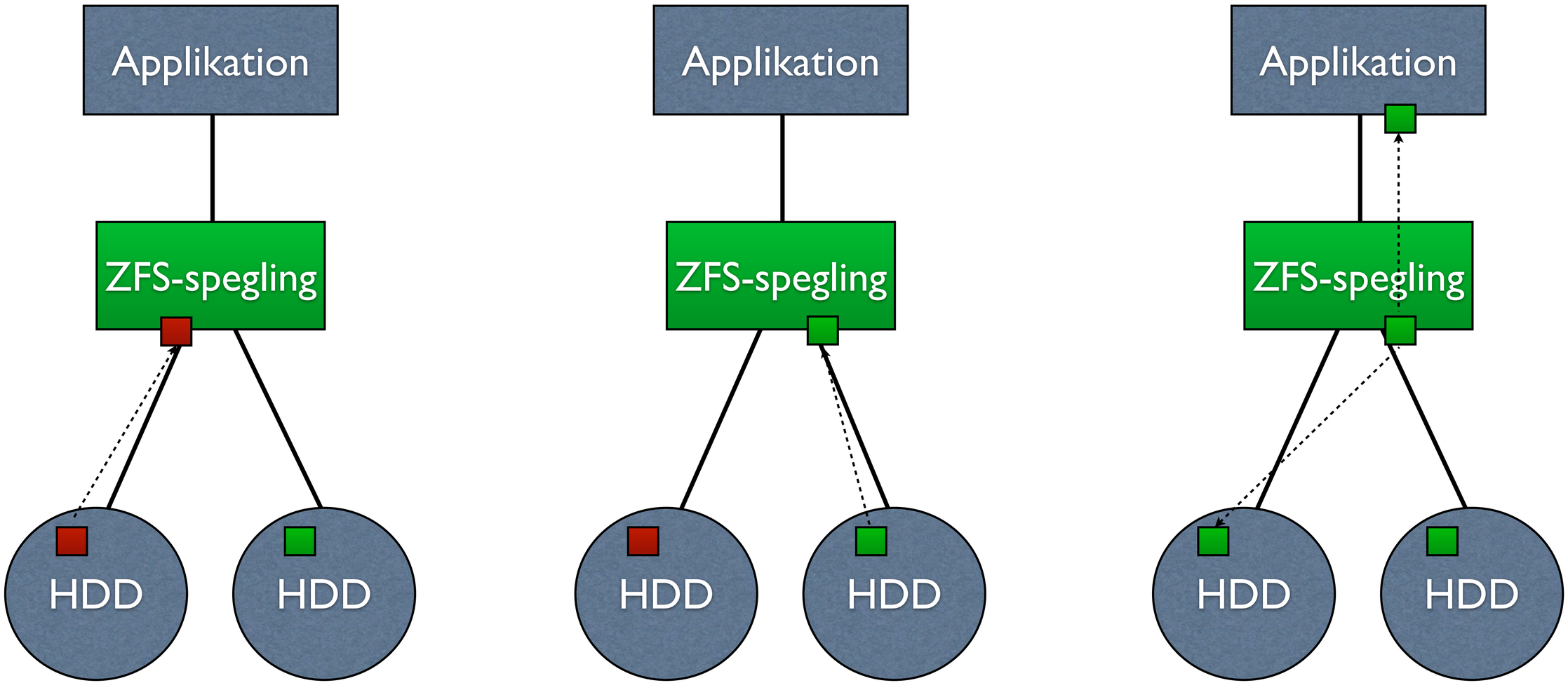
Traditionell spegling av data







Självläkande data i ZFS



Traditionell RAID4 och RAID5

Flera datadiskar + paritet

Skrivhål, förlorar man strömmen mellan skrivning av data och paritet så är RAIDen korrupt

Löses med NVRAM på RAID-kontrollern

Löser INTE tyst datakorruption

RAID-Z

- Dynamisk stripe-vidd
- Alla skrivningar är kompletta stripe-skrivningar
- Upptäcker och korrigerar tyst datakorruption
- Kräver ingen speciell hårdvara, fungerar bra med billiga diskar

Disk scrubbing

- Hittar fel medan de fortfarande går att rätta till
- Verifierar datans integritet
- Ger snabb och pålitligt återskapande av data, sk. resilvering

Skalbarheten hos ZFS

- 128-bitars filsystem
  - 1 ZB = En miljard terabyte
    - Klarar 256 quadriljoner ZB
- 100% dynamisk metadata
  - Ingen gräns på antal filer, filer per katalog, etc.

Prestanda hos ZFS

- Copy-on-write
- Dynamisk storek på stripes över alla hårddiskar
- Multipla storlekar på block
- Intelligent “prefetch”

Administration av ZFS

- Lagringspooler - Inga fler volymer
- Hierarkiskt filsystem med ärvda egenskaper
- Allt görs online, filsystemet behöver aldrig tas ner

Skapa lagringspool och filsystem

Skapa en spegling som heter “tank”

```
# zpool create tank mirror c0t0d0 c1t0d0
```

Skapa filsystem för hemkataloger, monterat till /export/home

```
# zfs create tank/home
# zfs set mountpoint=/export/home tank/home
```

Skapa hemkataloger för några användare

```
# zfs create tank/home/marcus
# zfs create tank/home/stefan
# zfs create tank/home/staffan
```

Lägg till mer utrymme till lagringspoolen

```
# zpool add tank mirror c2t0d0 c3t0d0
```

Sätta egenskaper

Automatisk exportering av alla hemkataloger via NFS

```
# zfs set sharenfs=rw tank/home
```

Använd kompression på all data i poolen

```
# zfs set compression=on tank
```

Ge Stefan en quota på 2 GB

```
# zfs set quota=2g tank/home/stefan
```

Garantera Staffan en reservation på 20 GB

```
# zfs set reservation=20g tank/home/staffan
```

Snapshots

- Readonly-kopia av ett filsystem
- Skapas direkt
- Använder inget extrautrymme, block kopieras bara när de förändras

Ta ett snapshot på Stefans hemkatalog

```
# zfs snapshot tank/home/stefan@tisdag
```

Rulla tillbaka till ett tidigare snapshot

```
# zfs rollback tank/home/stefan@mandag
```

Titta på onsdagens version av filen foo.c

```
$ cat ~stefan/.zfs/snapshot/onsdag/foo.c
```

Övriga funktioner

- Deduplicering
- Kryptering
- L2ARC
- ZIL