Salvestussüsteemid

- Pöörlevad kettad ja SSD
- Ketaste ühendamine
- Kettapöörduste planeerimine
- Ketaste ette valmistamine
- Saalimisala haldus
- RAID struktuurid
- LVM

Pöörleva ketta struktuur

- Ketas kui loogiliste plokkide jada (plokkseade)
- Plokkidele seatakse vastavusse sektorid
- 512 (520) baiti, sisemine plokisuurus on tänapäeval enamasti 4096 baiti
  - Enamasti näidatakse ühilduvuse huvides opsüsteemile 512-baidist sektorit
  - On ka 4096 välja paista laskvaid kettaid (4Kn — *4K native*)
- Sektorite ringi paigutamine vea korral (*reallocation*)
- Silindrid, rajad, sektorid (C/H/S) — pöörleva ketta siseasi

Ketta parameetrid

parameeter	360 kB floppy	WD 18300 ketas
silindrite arv	40	10601
radu silindril	2	12
sektorit rajal	9	281 (keskm.)
sektorit kettal	720	35742000
baiti sektoris	512	512
ketta maht	360 kB	18.3 GB
positsioneerimine kõrvalrajale	6 ms	0.8 ms
positsioneerimine keskmiselt	77 ms	6.9 ms
ühe täispöörde aeg	200 ms	8.3 ms
mootori käivitusaeg	250 ms	20 s
1 sektori ülekande aeg	22 ms	17 μs

SSD-seadmed

- SSD — *Solid State Drive* (flash-mälu plokkseadme liidesega)
- Seek tasuta, ülekandekiirus arvestatav kuni väga hea
- Piiratud kirjutamiste arv samale sektorile (DWPD kui mõõt)
- Toite kadumisel terve *erase block* kaotsis ühe sektori asemel
- *Wear leveling* — meetodid kogu kettapinna ühtlaselt vananemise tagamiseks
- Fragmenteerumine *wear levelingu* tegemisel vs kehv *wear leveling*
- TRIM lisakäsk mittevajalike alade vabastamiseks (et osa kirjutamisel ei oleks vaja vana seisu lugeda)
- Vajalik plokkide joondamine (*alignment*) *erase block* mahule
- OS: unustada pöörlemist eeldavad IO planeerijad

Ketaste ühendamine

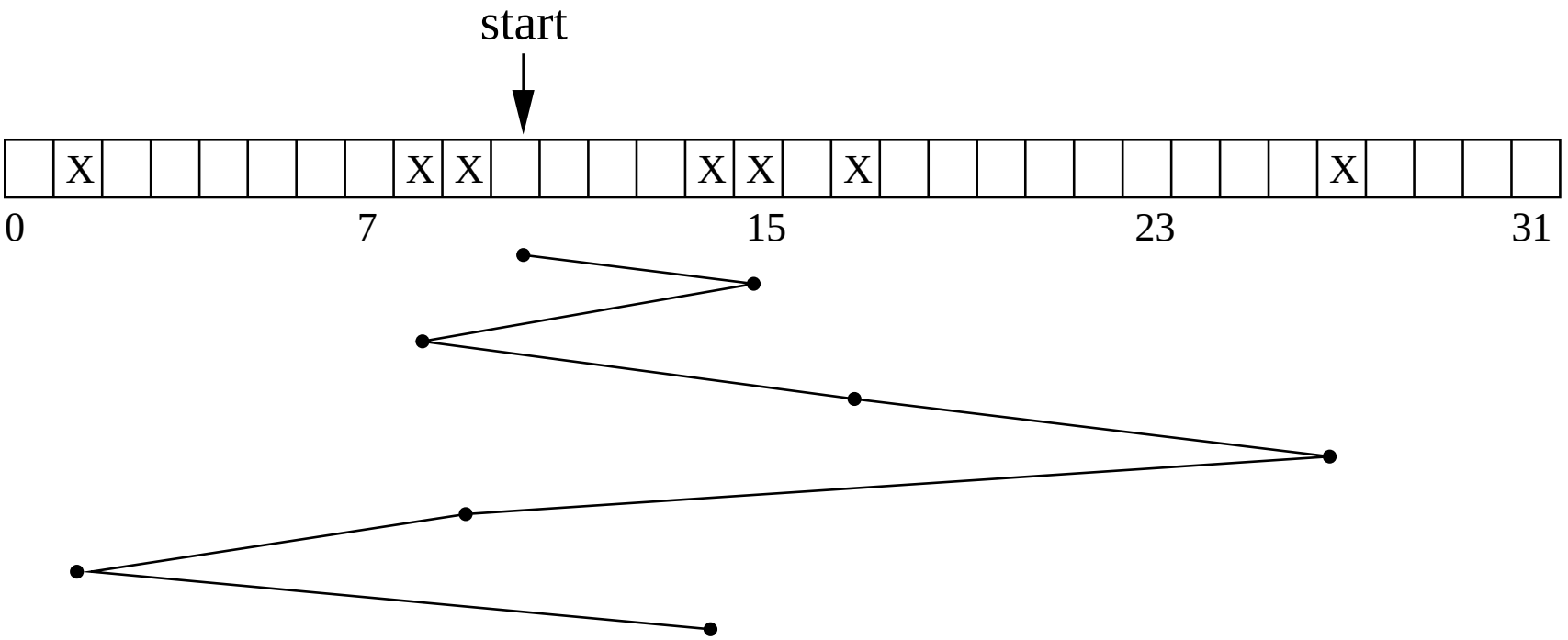
- Otse arvuti külge
  - (S)ATA, SAS/SCSI, NVMe, FireWire, USB, . . .
- Salvestivõrgu kaudu
  - SAN — *Storage Area Network*
  - Oma protokollistikuga võrk salvestusseadmete ja serverite vahel
  - Switched FibreChannel, FC-AL (kuni 32G, standard kuni 128G)
  - IP võrk (+1G/10G/40G/100G Ethernet jms) ja iSCSI
  - Infiniband (kuni 300G, madal latents) + RDMA

Kettapöörduste planeerimine

- Opsüsteem vastutab kettapöörduste efektiivsuse eest
  - Läbilaskekiirus suureks
  - Viivitus väikseks
- Pöördusajal kaks olulist tegurit:
  - Raja otsimine — aeg sõltub sellest, kui palju on pead vaja liigutada
  - Rajalt sektori otsimine (keskmiselt pool pööret)
- Läbilaske maksimaalne kiirus sõltub kettaseadmest
- Opsüsteem tahab liigse positsioneerimise välja optimeerida, et läbilase oleks maksimumi lähedane
- Kui meil on ketta järjekorda kogunenud päringud mingitele sektoritele, mis järjekorras neid täita, et viivitus (lugemispea poolt läbitud vahemaa) vähim oleks?

Klassikalised kettapöörduste planeerimise algoritmid

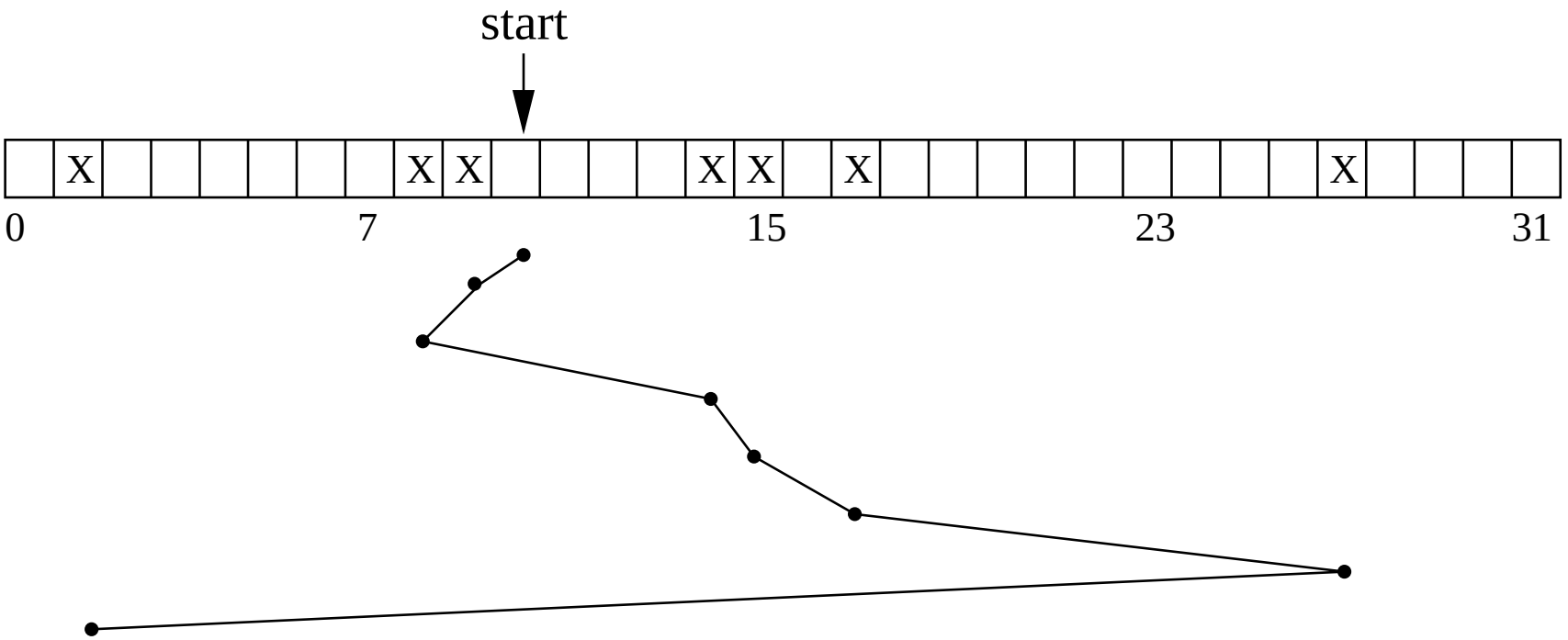
- FCFS — *First Come First Served* (70)



- Näide: päringute järjekord 15, 8, 17, 27, 9, 1, 14
- Kokku 70 raja vahetust

Klassikalised kettapöörduste planeerimise algoritmid

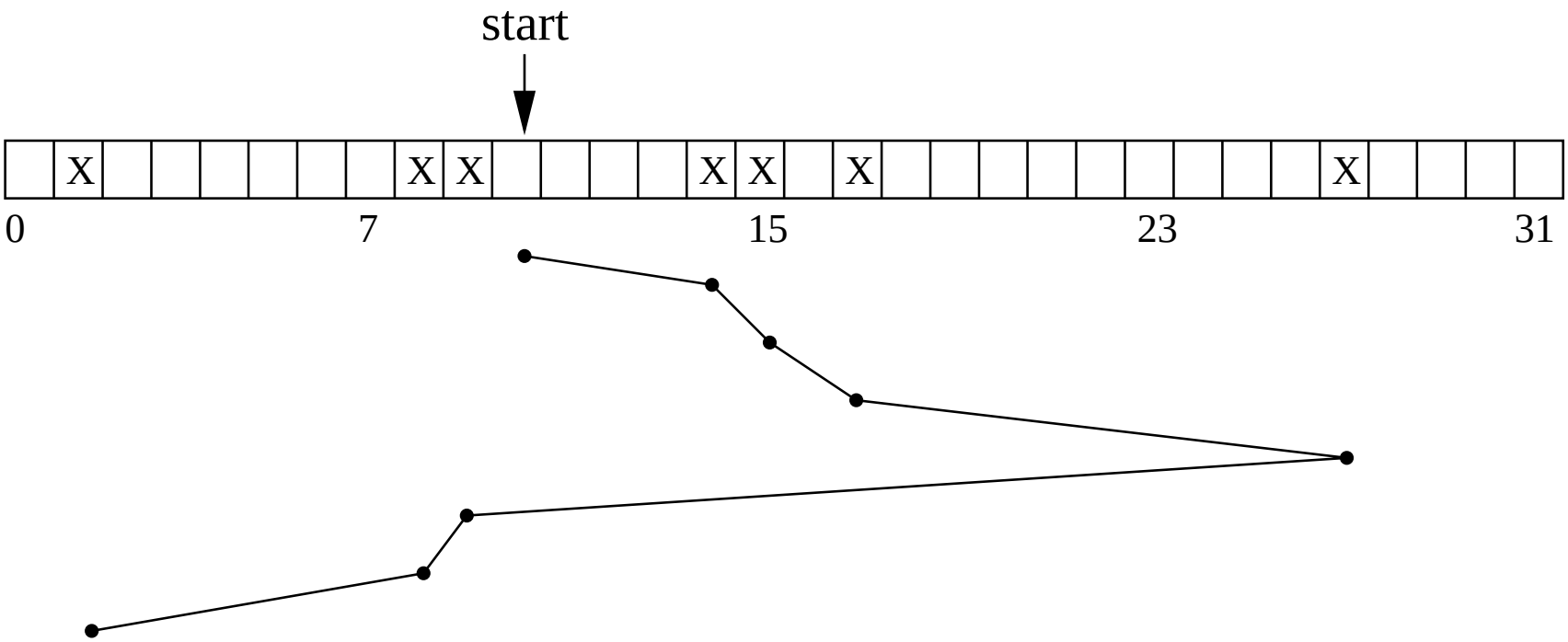
- SSTF — *Shortest Seek Time First* (47)



- Sarnane SJF protsesside planeerijale, aga pole optimaalne
- Võib põhjustada näljutamist

Klassikalised kettapöörduste planeerimise algoritmid

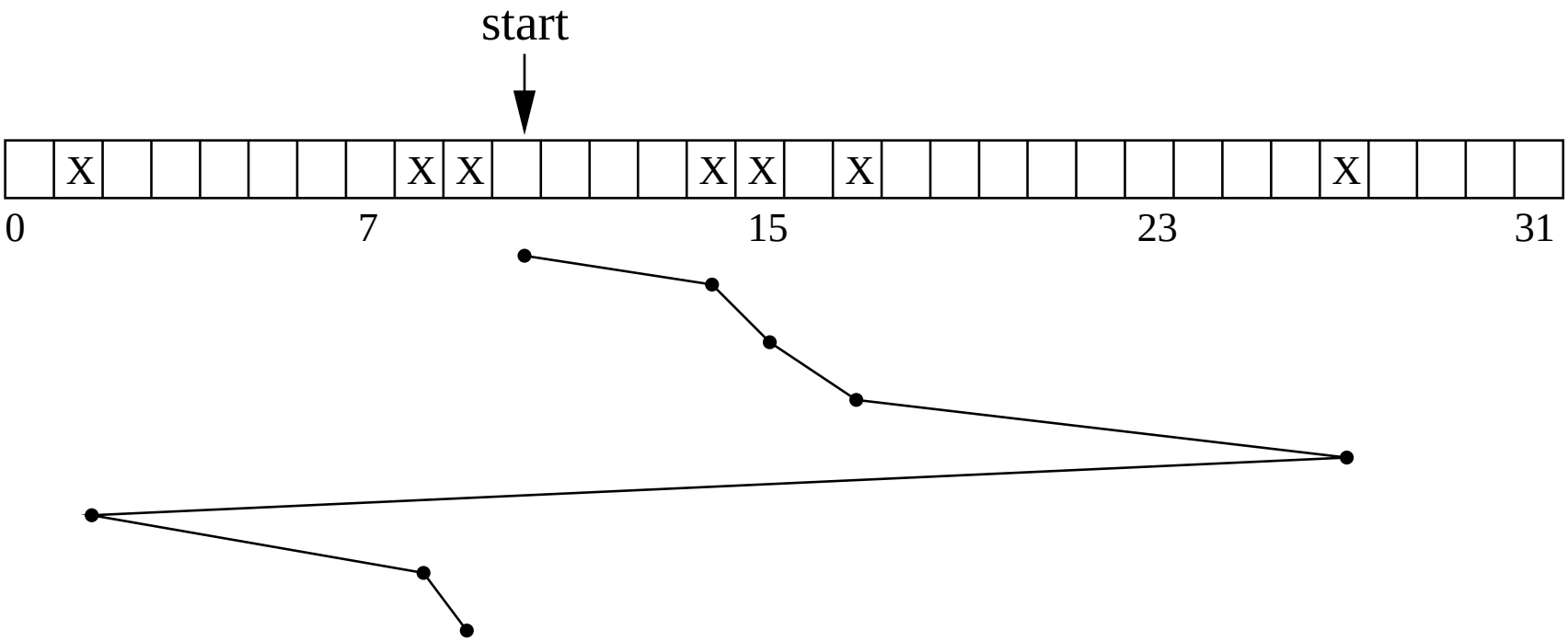
- SCAN (ka elevaatoralgoritm) — pea käib ketta ühest äärest teise ja teenindab teele jäävaid päringuid (51)
- LOOK — pea viiakse otstes ainult äärmise päringuni (43)



- Keskel asuvaid sektoreid teenindatakse kiiremini

Klassikalised kettapöörduste planeerimise algoritmid

- C-SCAN — nagu SCAN, aga tagasiteel päringuid ei teenindata (61)
- C-LOOK — pea viiakse otsest ainult äärmise päringuni (51)



- Päringute teenindamise ooteajad on ühtlasemad

Kettapöörduste planeerimise algoritmid Linuxis

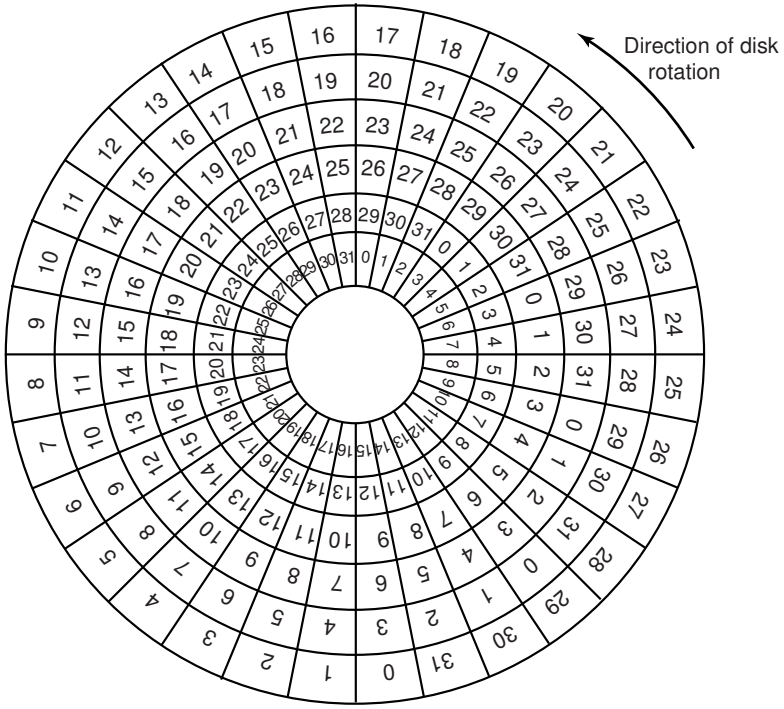
- NOOP (ainult päringute kokku kleepimine)
- Deadline (prioriteediklassid vastavalt interaktiivsusele, näljutamise vastu tähtaeg)
- CFQ (päringud jagatakse klassidesse ja võetakse klassidest vaheldumisi; oskab arvestada I/O prioriteetidega)
- MQ-Deadline, Kyber, BFQ — mitme järjekorraga variandid
- Sõltumata opsüsteemist: mitme järjekorrahalduse kasutamine üksteise otsas võib hoopis aeglustada

Ketaste ette valmistamine

- Madala taseme formaatimine (füüsiline formaatimine, *low level formatting*) — sektorimärkide radadele kandmine
  - Tänapäeval ei paista reeglina opsüsteemile enam kätte
- Partitsioneerimine — ketta osadeks jaotamine
  - MBR partitsioonitabelid — katavad kuni 2T, ketta alguses
  - EFI GUID Partition Table ehk GPT — katavad üle 2T, ketta lõpus
  - VTOC, Unixite *disklabelid*, . . .
- Loogiline formaatimine ehk failisüsteemi tekitamine
- Bootloaderi paigaldamine
- Vigaste kettaplokkide leidmine ja meelde jätmine

Madala taseme formaatimine

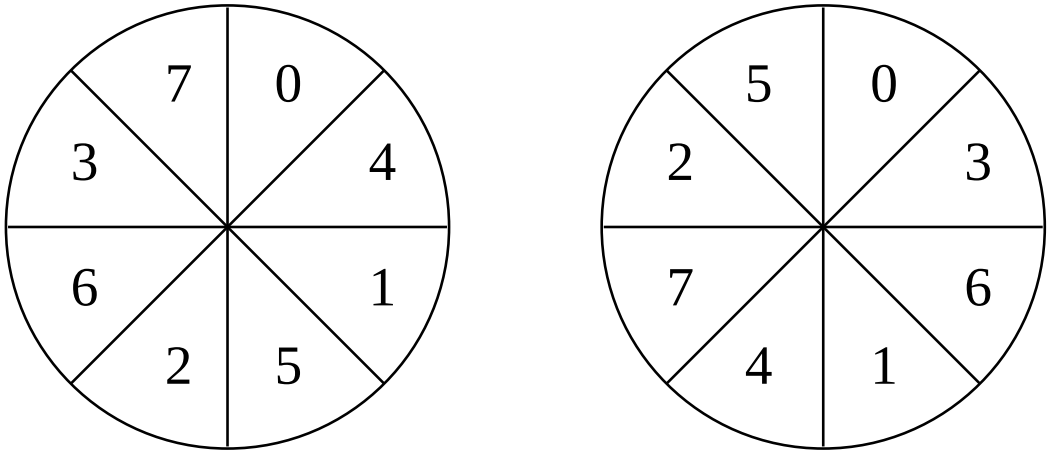
- *Cylinder skew* — erinevate radade sektorid on üksteise suhtes nihkes



- Aitab vähendada pöördumisviivitust järjestikuste sektorite lugemisel mitmelt rajalt

Madala taseme formaatimine

- Vahelejätt (*interleaving*) — sektorid paigutatakse üle ühe



- Aitab vältida olukorda, kus eelmise sektori andmete ülekandmise ajal liigub lugemispea üle järgmise sektori
- Aeglasema ülekandekanali puhul kasutatakse ka üle kahe vahelejättu (*double interleave*)
- Pole vajalik, kui kontroller puhverdab kogu raja tervikuna

Saalimisala haldamine

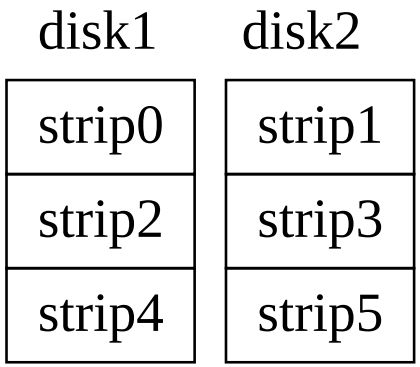
- Saalimisala — protsesside või lehekülgede salvestusruum virtuaalmälule
- Eraldi kettal/partitsioonil
- (Eelallokeeritud) failis
- Saalimisala protsessi kogu virtuaalmälu kohta (BSD Unix)
- Saalimisala reaalselt välja saalitud mälulehekülgede kohta (Solaris, Linux, . . .)
- RAM-is peab saalimisala kohta kaart olema

RAID massiivid

- RAID — *Redundant Array of Inexpensive Disks*
- Paralleelsuse kaudu kiiruse ja töökindluse suurendamine
- Peegeldamine (*mirroring, shadowing*)
- Paarsusinfo laiali jagamine (*interleaved parity*)
  - Bitikaupa
  - Plokikaupa
- Parandusaeg peab mõistlik olema
- Kuumvaru (*hot spare*), ketaste kolimine
- Tarkvaraline vs riistvaraline RAID
- NVRAM vahemälu

RAID 0

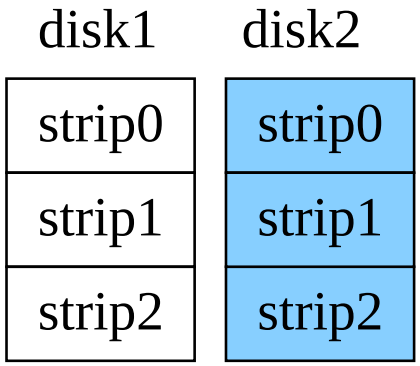
- RAID 0 — Mitmest kettast ühe suure tegemine (*striping*)
- Virtuaalne ketas koosneb vöötidest (*stripe*), igas vöödis k sektorit
- Vöödid on jagatud n ketta vahel



- Kõrvuti olevate vöötide lugemine võib toimuda paralleelselt
- Parandab jõudlust, kuid kahandab töökindlust
- Kui süsteem loeb ketast sektorhaaval, siis pole efektiivsuses võitu

RAID 1

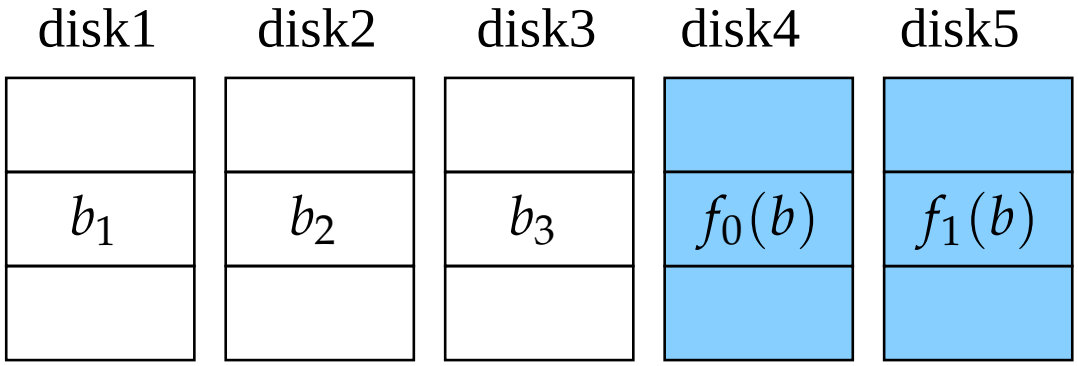
- RAID 1 — Peegeldamine (sama info mitme seadme peal)



- Säilivad RAID0 eelised (paralleelselt lugemine on võimalik)
- Vööt loetakse sellelt kettalt, kust parasjagu kiirem on
- Kirjutamised peavad toimuma kõigil ketastel, aga saab teha korraga

RAID 2

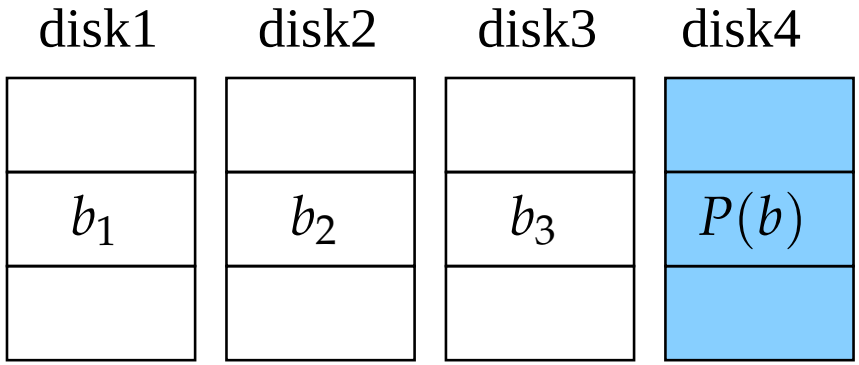
- RAID 2 — Mälu stiilis veaparanduskoodid (ECC)



- Vöödid on väikesed (biti, baidi või sõna suurused)
- Lisaks salvestatakse Hammingi kood
- Ühe ketta veast suudab taastuda
- Vajab log lisakettaid
- Harva kasutusel

RAID 3

- RAID 3 — Bitihaaval paarsuskontroll



- Vöödid on väikesed (biti, baidi või sõna suurused)
- Kasutab ära seda, et lugemisveast saab ketas ise aru, mitte ei anna valesid bitte
- Liiasuse saavutamiseks leitakse tegelike andmete XOR (paarsusinfo)
- Paarsusinfo jaoks on vaja 1 lisaketast
- Suudab taastuda ühe ketta veast

RAID 4

- RAID 4 — Plokihaaval paarsuskontroll

disk1	disk2	disk3	disk4	disk5
strip0	strip1	strip2	strip3	P(0−3)
strip4	strip5	strip6	strip7	P(4−7)
strip8	strip9	strip10	strip11	P(8−11)

- Sarnane RAID 3-ga, kuid suurema vöödisuurusega
- *Read-modify-write*
- Iga kirjutamisoperatsioon nõuab paarsusinfo korrigeerimiseks kõikide ketaste lugemist
- Alternatiivselt arvutatakse paarsusinfo vanast andme- ja paarsusinfost (2 lugemist)
- Paarsusketas võib osutuda pudelikaelaks

RAID 5

- RAID 5 — Plokihaaval hajutatud paarsuskontroll

disk1	disk2	disk3	disk4	disk5
strip0	strip1	strip2	strip3	P(0−3)
strip4	strip5	strip6	P(4−7)	strip7
strip8	strip9	P(8−11)	strip10	strip11
strip12	P(12−15)	strip13	strip14	strip15
P(16−19)	strip16	strip17	strip18	strip19

- Nagu RAID 4, aga paarsussektorid on hajutatud ketaste vahel
- Väldib eraldi paarsuskettaga seotud jõudlusprobleemi
- *Read-modify-write* on endiselt jõudlusprobleemiks
  - dünaamilise vöödisuurusega RAID — iga kirjutamine on täisvööt

RAID 6

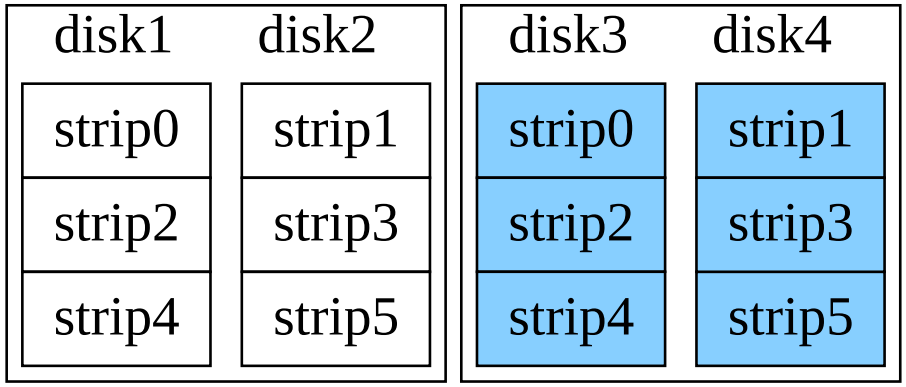
- RAID 6 — Kahemõõtmeline paarsuskontroll

disk1	disk2	disk3	disk4	disk5	disk6
strip0	strip1	strip2	strip3	P(0−3)	Q(0−3)
strip4	strip5	strip6	P(4−7)	Q(4−7)	strip7
strip8	strip9	P(8−11)	Q(8−11)	strip10	strip11
strip12	P(12−15)	Q(12−15)	strip13	strip14	strip15
P(16−19)	Q(16−19)	strip16	strip17	strip18	strip19

- Nagu RAID 5, aga liiasust arvutatakse kahe erineva funktsiooni abil
- Vajab kahte lisaketast
- Suudab taastuda kahe ketta vigade korral

RAID 0+1

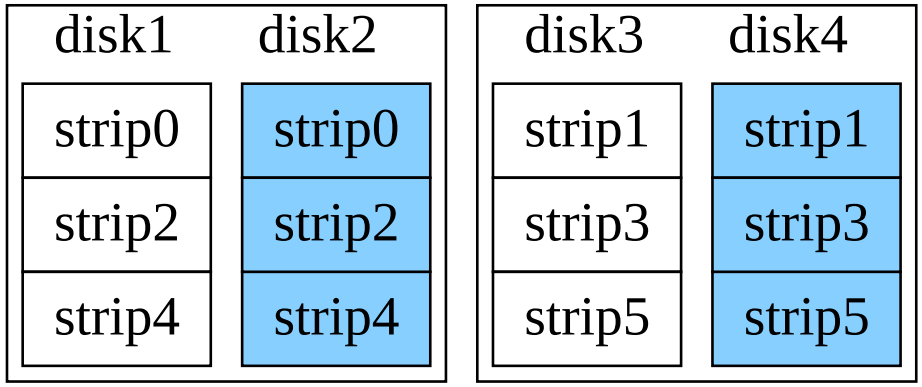
- RAID 0+1 — peegeldatud *stripe*'id



- Kannatab ühest peegli poolest kasvõi mitme ketta viga, aga teine peegli pool peab terveks jääma

RAID 1+0

- RAID 1+0 — *stripe*'itud peeglipaarid



- Kannatab ühe ketta viga igast peeglipaarist
- Keskmiselt kannatab vigu rohkem kui RAID 0+1, aga kaks veaga ketast võivad ka RAID 1+0 maha võtta

RAID riistvaras ja tarkvaras

- Enamus RAID tasemeid saab realiseerida nii riistvaras kui tarkvaras
  - Missuguseid ei saa?
- Riistvaras:
  - CPU koormust vähemaks
  - Liiasusega andmed liiguvad üle siini 1 korra
- Tarkvaras:
  - CPU võib olla kiirem kui RAID kontrolleri protsessor
  - Opsüsteemi RAID on kergemini hallatav ja vähemate ühilduvusprobleemidega
- *HostRAID*, BIOS RAID — tarkvaraline RAID draiveris
  - Võimalus bootida muudelt RAID-idelt kui 1
  - Hädalahendus, kui opsüsteemi RAID-i ei saa kasutada

RAID failisüsteemi tasemel

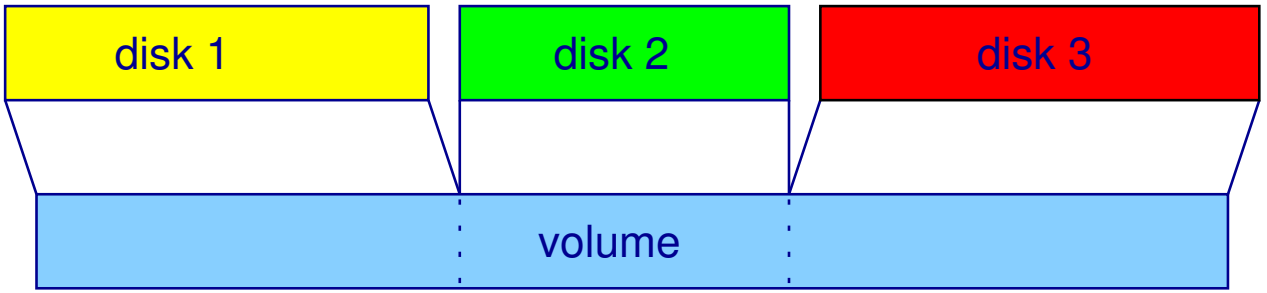
- Liiasust ei pea realiseerima plokkseadme tasemel, vaid võib teha ka failisüsteemi siseselt
- Anname komponent-plokkseadmed otse failisüsteemi hallata
- Failisüsteemi draiver jagab üksikute failide plokke laiali vastavalt liiasuspoliitikale
- Pole fikseeritud vöödisuurust
- Saame ellimineerida *read-modify-write* jõudlusprobleemi *copy-on-write* mehhanismi ja täisplokkide kirjutamisega
- Pioneer: Sun ZFS
- Tänapäeval on sarnane ka näiteks Linuxi Btrfs

LVM — *Logical Volume Management*

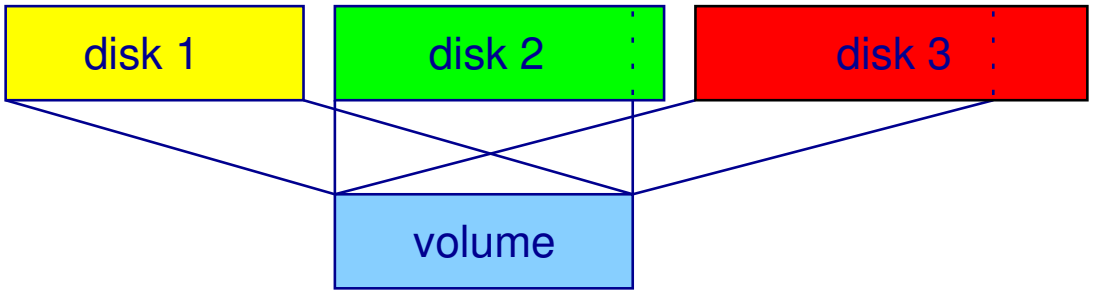
- Idee: abstraheerida salvestussüsteemist välja füüsilised kettad
- Algus: kogu füüsilise ketta peal oli üks failisüsteem
- Areng: füüsilise ketta partitsioonideks jagamine
- (RAID: peegeldamine, *striping*, . . .)
- Samm edasi: peidame füüsilised kettad üldse ära, näitame ülespoole ainult loogilisi "köiteid"
- Need köited võivad tegelike ketaste osade vahel suvaliselt jaguneda
- Tegelikud kettad jagame väikesteks tükkideks, mida saab köideteks grupeerida

LVM: näited

- Ketaste kokku kleepimine



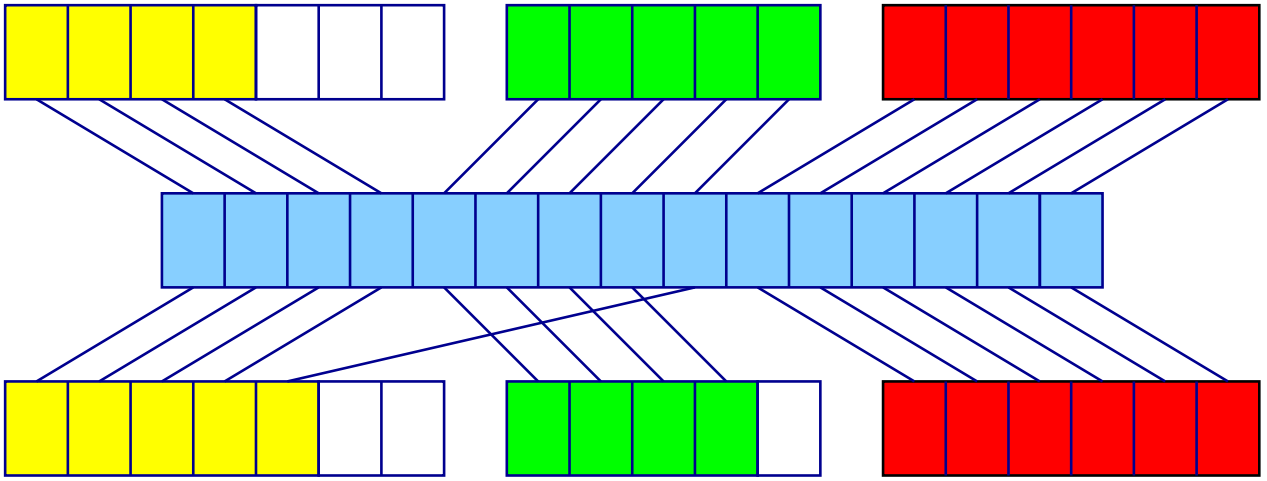
- Peegeldamine



- $\Rightarrow$ esialgu nagu RAID

LVM: näited

- LVM plokkide migreerimine teisele kettale



- Köidete suuruse muutmine käigu pealt
- Krüpteerimine
- Vahemälu SSD peal
- *Multipathing*
- *Thin provisioning*

LVM: näited

- Snapshot