Nettverkslagring (SAN)

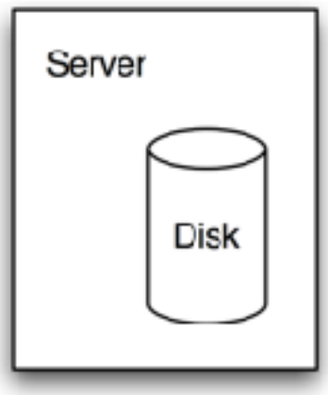
Lagring: Serdeles viktig

- Informasjon (data) er ...
  - ekstremt verdifullt
  - regulert av lovverket (DLD)
  - en operativ avhengighet
  - omtrent fullstendig digitalisert
    - ... og dermed er det blitt ditt problem

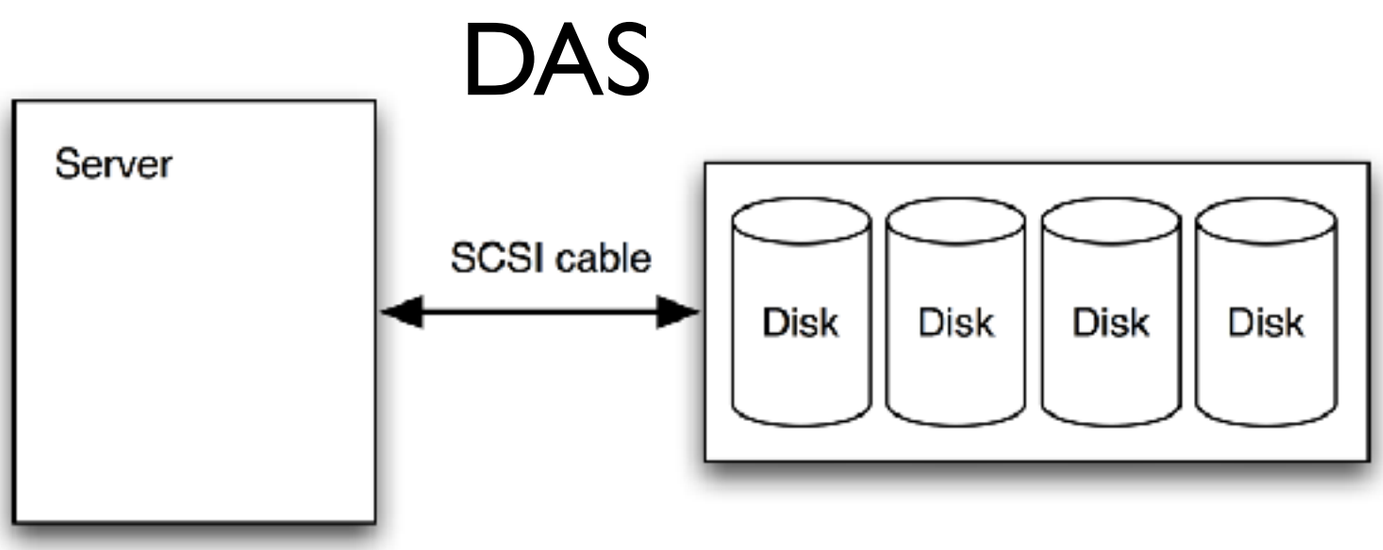
Enkleste tilnærming: lokal disk

- Å installere store mengder lagring direkte på en server har visse ulemper:
  - Systemfeil betyr at dataene er utilgjengelige
  - Ingen lastbalansering
  - Få muligheter til å endre ting i farta (unntatt kanskje hot-swap)
  - Store disker blir ikke utnyttet
  - Dyrt
  - Følger livssyklusen til serveren (som ikke er spesielt lang)

Gammeldags alt-i-ett



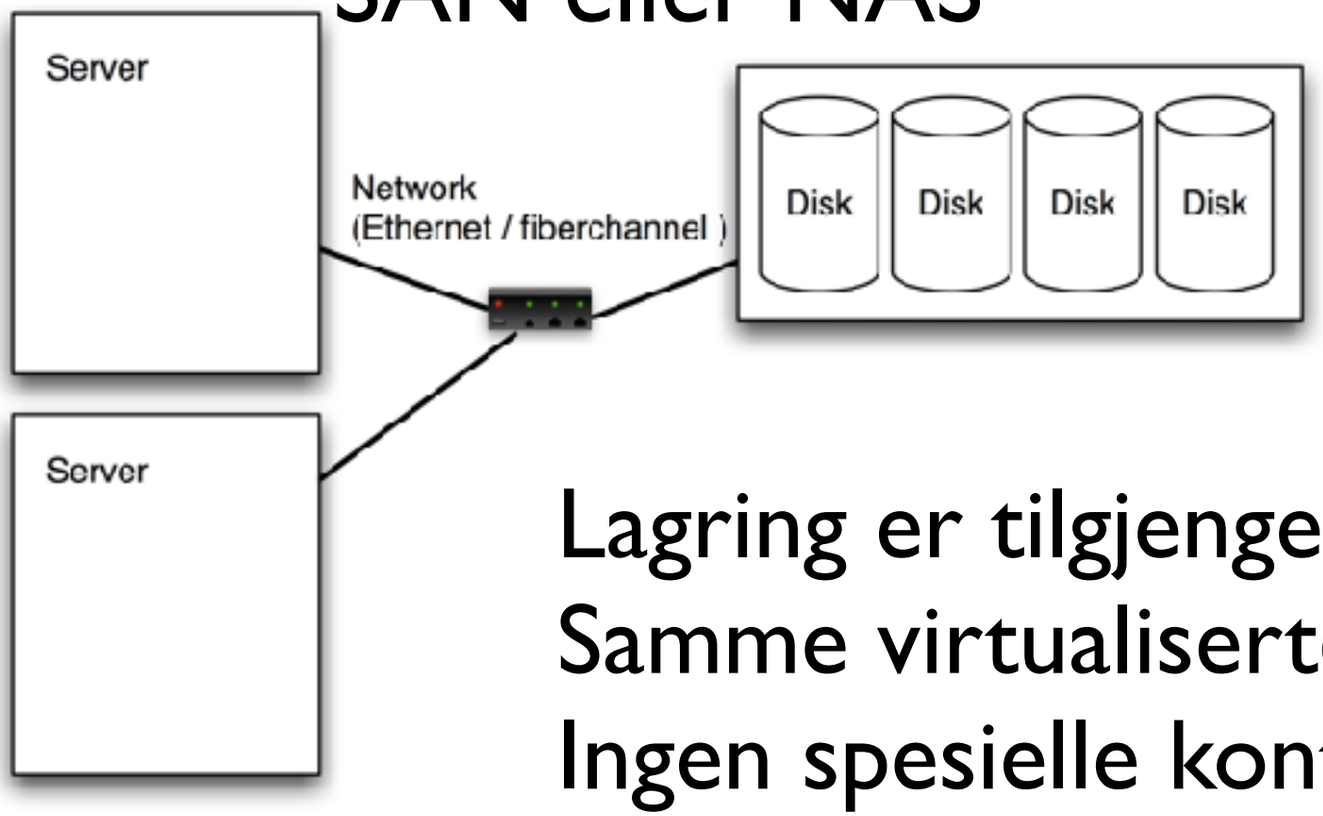
Neste steg: Direct attached storage



Serveren har en SCSI kontroller med eksterne tilkoblingsmuligheter. DAS er koblet direkte til serveren via en SCSI kabel.
Serveren ser kun en “virtuell disk”.
Kun korte distanser.
Lite fleksibelt, få servere kan være koblet til på likt.
Renéssanse med eSATA

Dagens praksis: Nettverkslagring

SAN eller NAS



Lagring er tilgjengelig via nettverket.
Samme virtualiserte disker som i DAS.
Ingen spesielle kontrollere.
Uavhengig av nettverkstopologi. (Bortsett fra FiberChannel og ATA over Ethernet)
Skalerer til mange klienter.
Ingen begrensning mht. fysisk distanse.

Storage Area Network (SAN)

- En måte å koble *lagringsenheter* til maskiner over nettet.
- Mange forskjellige teknologier (iSCSI, AoE, FC, DRBD, CEPH)
- Dedikerte "bokser" eller hjemmelagde
- Mange leverandører og prosjekter
  - Dell, HP, Sun, IBM, NetApp, openfiler
- Stor business

SAN vs NAS

SAN

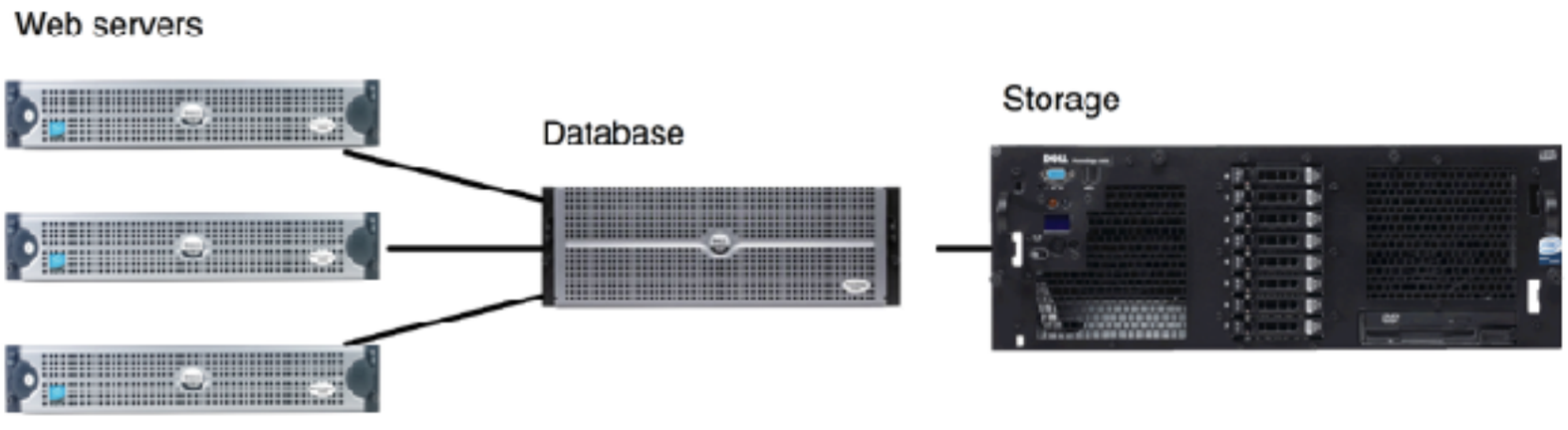
- Nettverksbasert ‘block-device’
- Kun lese/skrive operasjoner over nettet
- Sepparate nettverk for økt ytelse og sikkerhet

NAS (Network Attached Storage)

- Nettverksbasert filsystem (SMB, AFS, pvfs, NFS/CIFS)
- Fil-operasjoner (åpne/lukke) over nettet
- Låsing
- Lokalt nettverk med innebygd tilgangskontroll

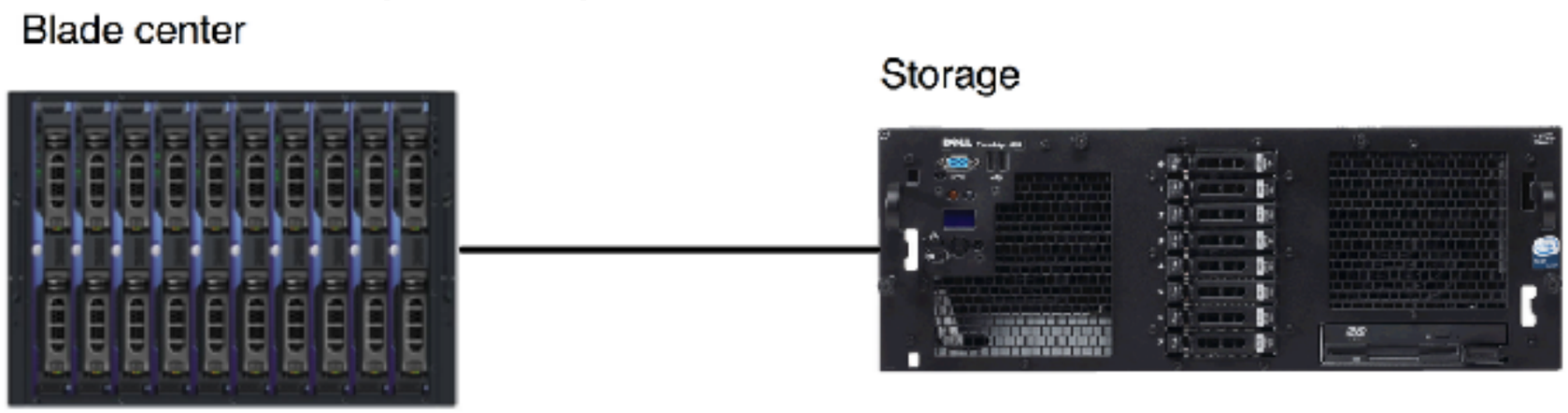
Eksempel bruk: Flerlags Applikasjoner

- Lagring er koblet til en database server
- Serverne er koblet til databasen
  - Ikke direkte tilgang til SAN
- Typisk for webservere og spillservere



Eksempel bruk: Generell lagring

- Servernene er små med minimalt av disk, men mye minne og regnekraft.
- Hovedinnholdet på serverne befinner seg på et SAN som er montert opp som diskpartisjoner
- Typisk bruk: Epost servere, Virtualisering, fil servere (NAS)



iSCSI

Hva er en 'block-device'?

- Tradisjonelle UNIX systemer har to måter å kommunisere med maskinvare:
  - Block device
    - Operasjonene er bufret (bruker minne som cache)
    - Tilfeldig aksess
    - Filsystemer kan kun plasseres på block-device
    - Eksempel: Disker
  - Character devices
    - Operasjonene er ikke bufret
    - Normalt sett ikke tilfeldig aksess
    - Eksempler: Keyboard, TTY, seriell linjer

iSCSI

- SCSI instruksjoner over et IP-basert nettverk
- Populært alternativ til den kostbare FiberChannel arkitekturen
- Finnes båre Open Source og kommersielle produkter
  - "Hardware" og "Software" løsninger
- Støttet på både Linux og Windows
  - Mac OS med tredjeparts programvare (f.eks globalSAN)

Portals, Targets og Initiators

iSCSI bruker en egen terminologi

- Selve iSCSI serveren kalles for et portal
- De delte diskene kalles for ‘targets’
- Klient-programvaren kalles for ‘initiators’

Iscsitarget

- iSCSI Enterprise Target
  - http://iscsitarget.sourceforge.net
- Linux implementasjon av en iSCSI tjeneste
  - Dokumentasjonen er "som forventet" for open source prosjekter ...
    - Aktiv utvikling, men dokumentasjonen mangler
  - Bruker kjernemoduler for økt ytelse
  - En nyere iSCSI tjeneste for linux er tgtd

Installasjon av iscsitarget

- På de fleste Linux distribusjonene finner man iscsitarget som pakker

  `apt-get install iscsitarget`

  - Egenkompilerte kjerner må muligens kompilere modulene selv
- Etter installasjonen, må man tillate at tjenesten startes. I filen: /etc/default/iscsitarget:

  `ISCSITARGET_ENABLE=true`
- Start tjenesten:

  `/etc/init.d/iscsitarget start`

Forbered et "target"

- Konfigurasjonsfilen er /etc/iet/ietd.conf
- Man trenger en partisjon eller fil man ønsker å dele
- Editer konfigurajsonsfilen og legg til:

```
Target identifier:name

Lun 0 Path=/path/to/file,Type=(fileio|blockio)
```

Eksempel:

```
Target 10.0.0.252:mln.windows.vm87
Lun 0 Path=/dev/san/windows.vm87,Type=fileio
```

- Restart tjenesten

blockio vs fileio

- 'blockio' vil kommunisere direkte med block device uten lokal caching.
  - "Lavere" ytelse enn fileio, men mer forutsigbart
- 'fileio' vil la maskinen bruke egen filsystem cache til å mellomlagre innholdet
  - Bedre ytelse ved store mengder minne
  - Kommer til å bryte alt opp i 4k deler (page size)
  - Data skrives ikke til disk umiddelbart (økt risiko)
- Man kan bruke fileio på block device
- Kan være policy-basert (noen på blockio, andre på fileio)

Overvåkninsmuligheter

- Tilstandsinformasjon kan hentes direkte fra kjernen
- cat /proc/net/iet/volume

```
tid:174 name:10.0.0.252:mln.windows.vm87
            lun:0 state:0 iotype:fileio iomode:wt path:/dev/san/
windows.vm87
```

- cat /proc/net/iet/session

```
tid:174 name:10.0.0.252:mln.windows.vm87
        sid:2187060582583040 initiator:iqn.1993-08.org.debian:
01:c51a3697152b
                cid:0 ip:10.0.0.3 state:active hd:none dd:none
```

Legge til nye targets i farta

- Det hender man ønsker å legge til nye targets
  - Dessverre innebærer editering av ietd.conf og restart av tjenesten at dette vil avbryte tjenesten for alle

1. Opprett devicet eller filen du vil dele
2. Finn en ledig target ID

```
less /proc/net/iet/volumes
```

3. Legg til device manuelt med ietadm kommando

```
ietadm --op new --tid=220 --params Name=sanctum:kyrre

ietadm --op new --tid=220 --lun=0 --params Path=/dev/san/kyrre,Type=fileio
```

Installasjon av open-iscsi

- Open-iscsi er en populær initiator for Linux
  `apt-get install open-iscsi`
- Når det er installert, har man tre komponenter:
  - Kommandoen iscsiadm
  - Demonen iscsid, som må kjøre
  - Kjernemoduler som blir lastet inn av iscsid
- To viktige tilstander:
  - En Node (target eller portal) er oppdaget (discovered)
  - Man er logget inn på et target

Vanlige kommandoer

- List opp kjente targets
  `iscsiadm -m node`
- List opp innloggede sesjoner
  `iscsiadm -m session`
- Manuelt legg til et target
  `iscsiadm -m node -T sansiro.vlab.iu.hio.no:lab30 -p 192.168.0.253 -o new`
- Manuelt logg in på et target
  `iscsiadm -m node -T sansiro.vlab.iu.hio.no:lab30 -p 192.168.0.253 -l`

Vanlige kommandoer II

- Log ut (koble fra) et target

```
iscsiadm -m node -T sansiro.vlab.iu.hio.no:lab30 -p 192.168.0.253 -u
```

- Fjern kjenskapen til target

```
iscsiadm -m node -T sansiro.vlab.iu.hio.no:lab30 -p 192.168.0.253 -o
delete
```

- Oppdagelse, list opp alle tilgjengelige targets og lagre dem

```
iscsiadm -m discovery -t sendtargets -p 192.168.0.253:3260
```

- Få statistikk over sesjonene

```
iscsiadm -m session -s
```

- Få detaljert sesjonsinfo

```
iscsiadm --mode node -T sansiro.vlab.iu.hio.no:lab29 -p 192.168.0.253
```

Hva skjer så?

- Når man har startet en sesjon (logget inn på et target) vil Linux sette opp et nytt device som om det var en lokal disk (sd*)
  - Litt uforutsigbart
- Man kan også finne den her

```
/dev/disk/by-path/ip-192.168.0.253:3260-iscsi-
sansiro.vlab.iu.hio.no:lab29-lun-0
```

  - Ja, den er lengre, men den er lik hver gang
- Deretter kjør mkfs og montér med mount som vanlig

Overvåke ytelsen

- Ingen egne overvåkingsverktøy men nok av data å ta tak i
- På serveren og klienten
  - /proc/diskstats: Sektorer som er lest/skrevet samt vente-køen
  - Nettverksdata
- Verktøy slik som munin kan gi et generelt inntrykk
  - Ekstra plugins kan brukes for å lete etter faresituasjoner
  - ‘iostat -k 5’ kan gi et bra her-og-nå inntrykk

Hva slags ytelse man kan forvente

- Nettverksbasert lagring har tradisjonelt sett lavere ytelse (med mindre det er skikkelig dyrt)
  - Ytelsen varierer mer
- Klienter kan lett påvirke hverandre
- Linux + iscsitarget gir allikevel nokså bra fleksibilitet i forhold til kostnadene

Hva skal man se etter ved lav ytelse

- Lang kø på de lokale diskene på SAN serveren (bruk /proc/diskstats)
- Tregt nettverk
  - For mange små pakker (jumbo frames?)
- Hyperaktive klienter
  - Må muligens isoleres
  - Dårlig timing av oppgaver? (alle tar backup på likt)

Mulige justeringer

- Hvert target betyr 8 tråder på serveren
  - 50 targets = 400 tråder ...
    - Juster antall tråder med opsjonen “Wthreads x” per target
- Eksperimenter med fileio istedetfor blockio
- Sektor / blokk -plassering

Vanlige feller

- iSCSI gir ikke låsing
  - Vanskelig å dele samme iSCSI target mellom to maskiner
  - Tilnærminger finnes, slik som CLVM, men der er tunge arkitekturer
- For lite tid til design og testing
  - Så snart et SAN er i produksjon er det vanskelig å endre ting
    - Må ta ned veldig mange andre systemer
    - For lite tilgjengelig mellomlagring

GlusterFS

IMT3003 kyrre.Begnum@ntnu.no

Clusterfilsystemer

- Clusterfilsystemer er et NAS/SAN-lignende konsept, der mange maskiner går sammen for å gjøre lagring tilgjengelig
- Nøkkelord:
  - Stor-skala (Enterprise scale)
  - Høy redundans
  - Høy ytelse
  - Skalerbart

GlusterFS.org

- Open Source prosjekt som støttes av bl.a RedHat
- Bred støtte i Linux distroer (og lite i andre)
- Mange bruksområder:
  - Klassiske beregningscluster
  - BigData
  - Tjenestearkitekturer
  - Virtualisering
- Enkelt å bruke

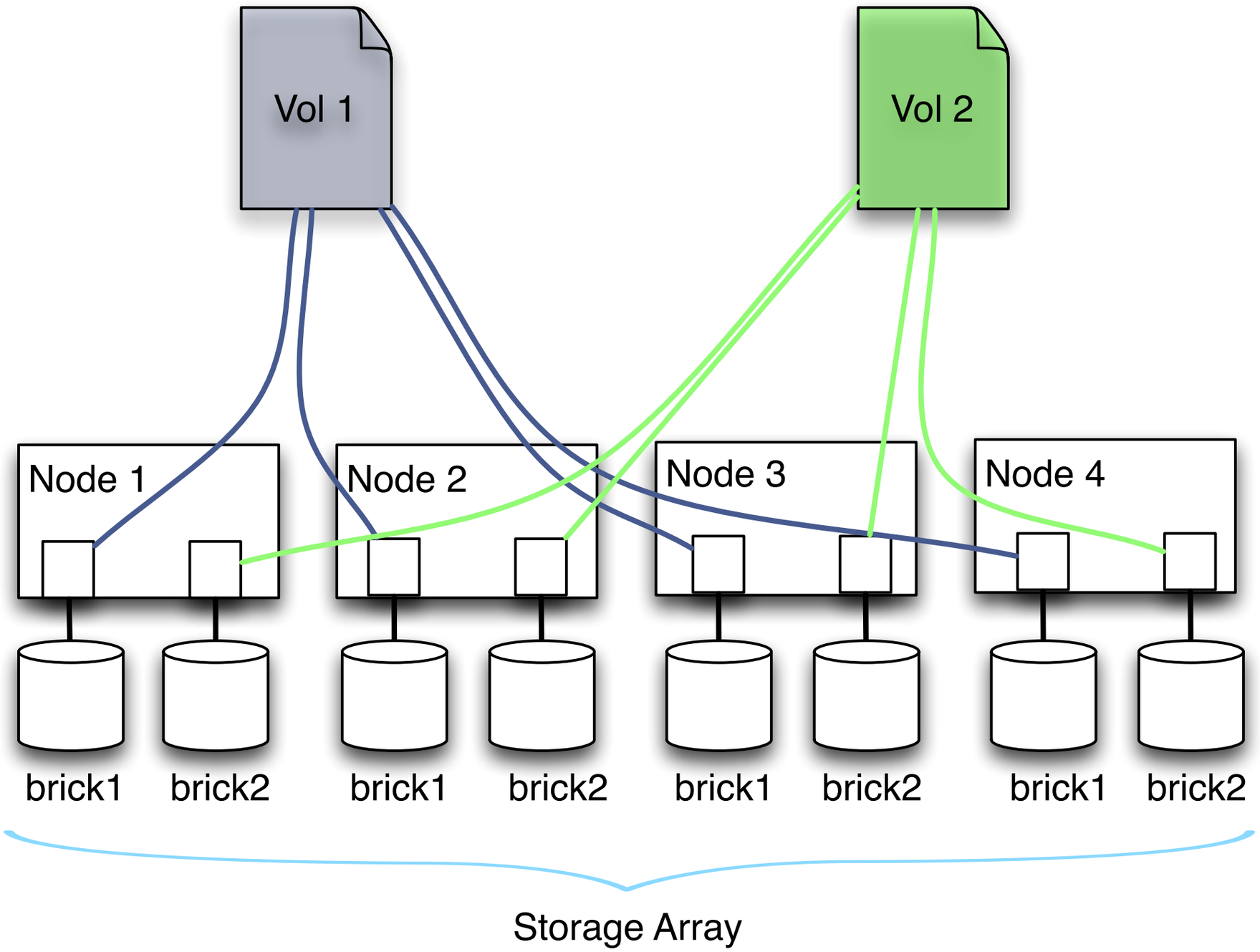
GlusterFS Egenskaper

- Redundans
- Replikering, striping og/eller distribuering
- Fungerer med vanlig maskinvare
- Geo-replikering (over SSH om man vil)
- Skalering
- Kvoter

Oppbyggning

- En gruppe maskiner som er et cluster kalles "storage array"
- Hvert medlem er en "node"
- Hver node har en eller flere disker som utelukkende brukes til GlusterFS og kalles "brick"
- Et "volume" er en gruppe bricks i arrayet som har data spredt ut på en bestemt måte

Illustrasjon



Replikering, striping og distribuering

- GlusterFS har tre måter å sortere data på:
  - Replikering *n*: Data blir replikert mellom *n* bricks
  - Distribuering *n*: Filene blir fordelt mellom *n* bricks
  - Striping *n*: Data fra filene blir splittet opp og fordelt over mellom *n* bricks
- Det er selvfølgelig lov med kombinasjoner av disse, men da må man ha riktig antall bricks

Eksempler

- Anta hver brick er 1 TB
- Replikering 2 + distribuering 3
  - Totalt 6 bricks hvor hver fil havner to steder
  - 3TB lagring
- Replikering 2 + striping 2
  - Totalt 4 bricks hvor hver fil blir spredt ut over to bricks
  - 2 TB lagring
- Replikering 3 + distribuering 2 + striping 2
  - Totalt 12 bricks hvor hver fil lagres over to bricks, tre ganger
  - 4 TB lagring

I bruk

- Klientene mounter et glusterFS volum og ser alle filene
- Klienten vet om alle nodene som er involvert
- Dersom en node faller fra, kommuniserer den videre med en annen
- Ved distribuering og en node som faller fra, forsvinner f.eks halvparten av filene

Installasjon på servere

Vi antar to servere: storage1 og storage2 uten ekstra disker, hvor vi later som om vi har en brick på hver (lager bare en mappe).

- På begge:

  apt-get update
  apt-get install glusterfs-server
  mkdir /brick

- På storage1

  gluster peer probe storage2
  gluster volume create gv0 replica 2 storage1:/brick storage2:/brick
  gluster volume start gv0

- Sjekke status

  gluster volume info
  gluster peer status

Fra en klient

- Følgende kommando fra en klient:
  ```
  apt-get install glusterfs-client
  mount -t glusterfs storage1:gv0 /mappe
  ```
- Dette vil ikke overleve en reboot, da kan det være lurt å sette den siste kommandoen i filen /etc/rc.local eller sørge for at det kjøres ved oppstart på en annen måte
- Ofte ender man opp med en avhengighet, hvor man vil forhindre en tjeneste å starte før volumet er på plass

Designspørsmål

- Nøkkelen til suksess for clusterfilsystem som GlusterFS er den optimale sammensetningen av bricks i forhold til bruk
  - Mange små filer eller få store?
  - Er redundans eller ytelse viktigst?
- Ofte er løsningen å lage flere volumer som er spesialisert til hvert sitt bruksområde