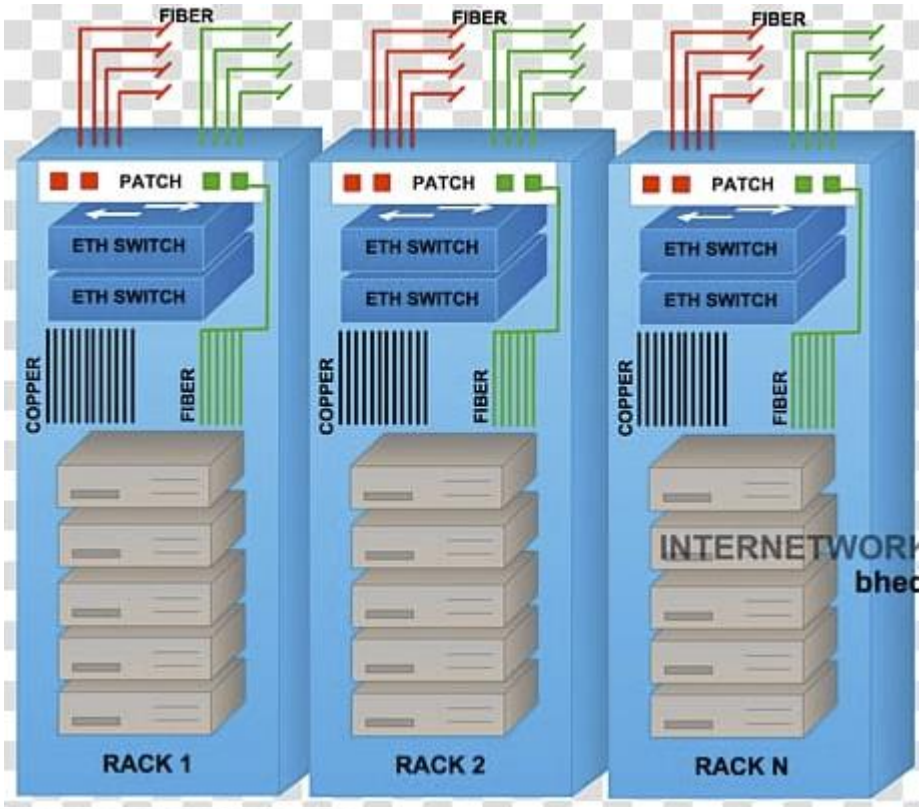
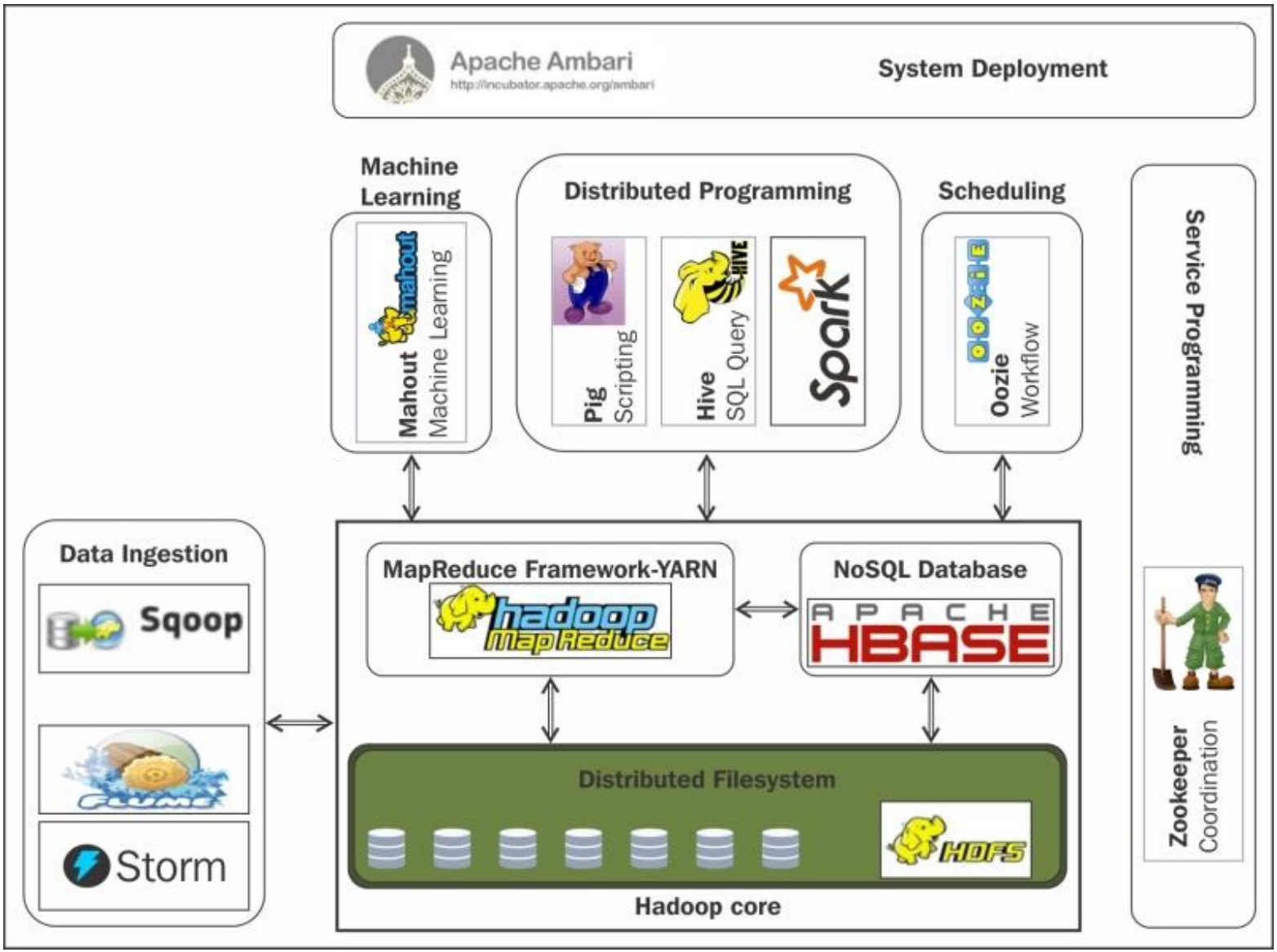
Hadoop File System

Typisk maskinvare-kontekst for Big Data-rammeverk



«In house» eller skybasert

Hadoop-økosystemet

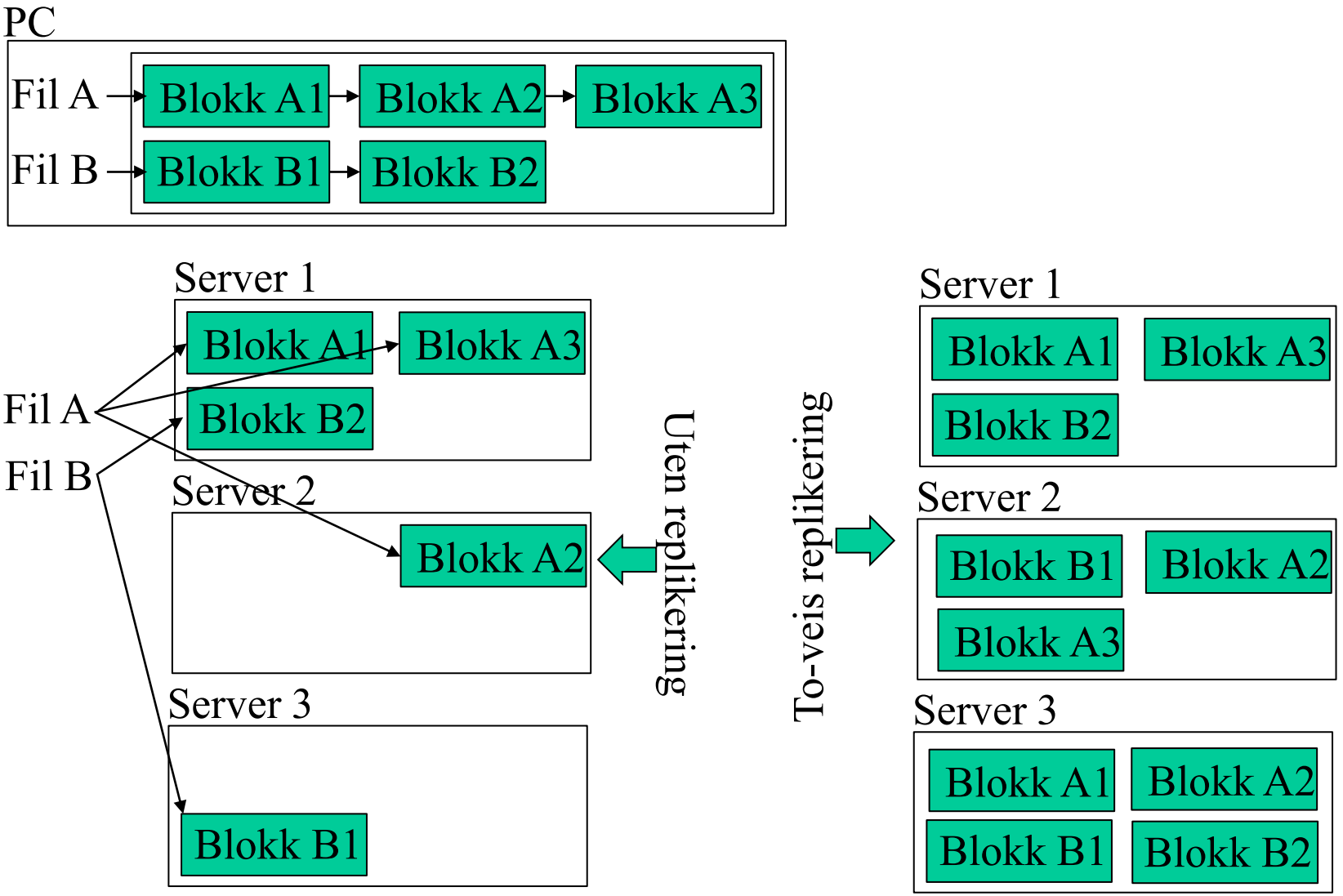


(Frå *Hadoop Essentials*)

Hadoop File system (HDFS)

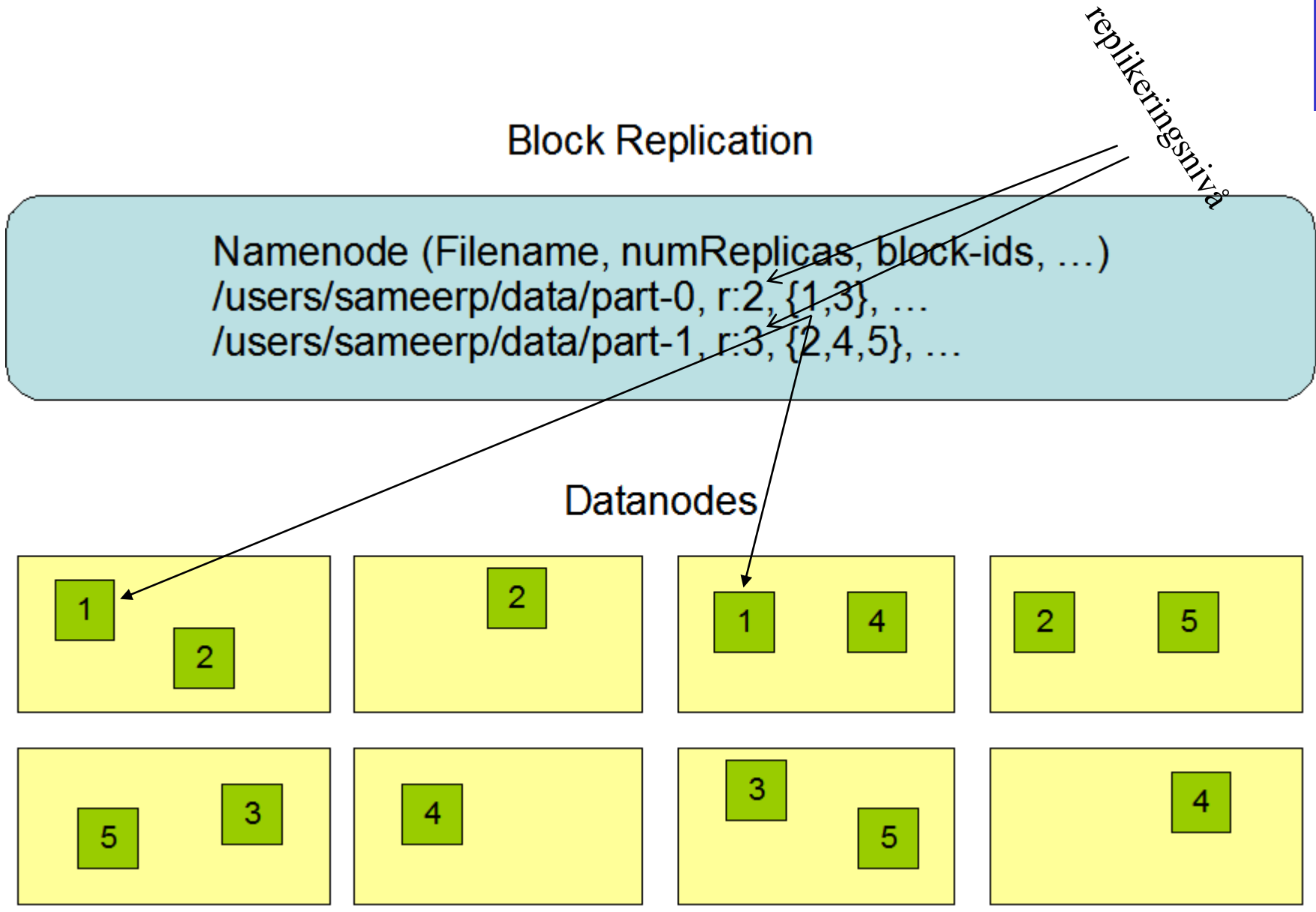
- Sterkt inspirert av Google File System (GFS)
- Standard filsystem i Hadoop
- ”User-level” filsystem
  - På topp av lokalt FS, alle data lagra i «vanlege» filer
- Mål:
  - (Veldig) store filer
  - Datastrøm-aksess
  - Hyllevare (maskiner)
- Ikkje eigna til:
  - Data-aksess med krav til liten forseinking (*low-latency data access)*
  - Mange små filer
  - Fleire samtidige skrivarar (dvs. er *single-writer*)
  - Vilkårlege oppdateringar i filer (dvs. er *append-only*)

Lokalt vs. distribuert filesystem (*veldig* forenkla)



Blokker

- Filer og metadata i HDFS lagra på ”lokalt” filsystem
- Filer delt opp i *blokker* med default størrelse typisk 64MB eller 128MB
  - *Jfr: disk-sektor på 512B og (typisk) 4 eller 8 KB blokkstørrelse i lokalt filsystem*
- Store blokker reduserer aksess-tid (seek)
- Kvar blokk lagra som ei ”lokal” fil
- Feiltoleranse vha. replikering av blokker på fleire nodar
- NB! Filer som er mindre enn 64/128MB bruker kun så mykje disk-plass som reell størrelse på fila
- Blokker (fast størrelse):
  - Mogleg med filer som er større enn det som kan lagrast lokalt på ei maskin
  - Gjer replikering enklare (samanlikna med fil som granularitet)


replikeringsnivå
Block Replication
Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …
Datanodes
1
2
2
1
4
2
5
5
3
4
3
5
4

Node-typar

- *Master-worker*: *NameNode* og *DataNode*
- *NameNode:*
  - Handsamar namnerom til filsystem (filer, katalogar, og metadata for desse)
  - All informasjon kontinuerleg i hovudlager pga. effektivitet, disk kun for persistens
  - Informasjon lagra persistent på lokal disk som to filer: *namespace image* og *edit log*
  - Alle fil-metadata-operasjonar skrivne til log før operasjon stadfesta (write ahead logging)
  - Også oversikt for kvar fil kvar blokker er lagra
    - Ikkje nødvendig å lagre dette persistent (kan rekonstruerast frå datanodar ved restart)
  - *Secondary NameNode* (CheckpointNode): lagar nytt namespace image ved å kombinere noverande namespace image med operasjonane i edit log, kan så hentast av NameNode

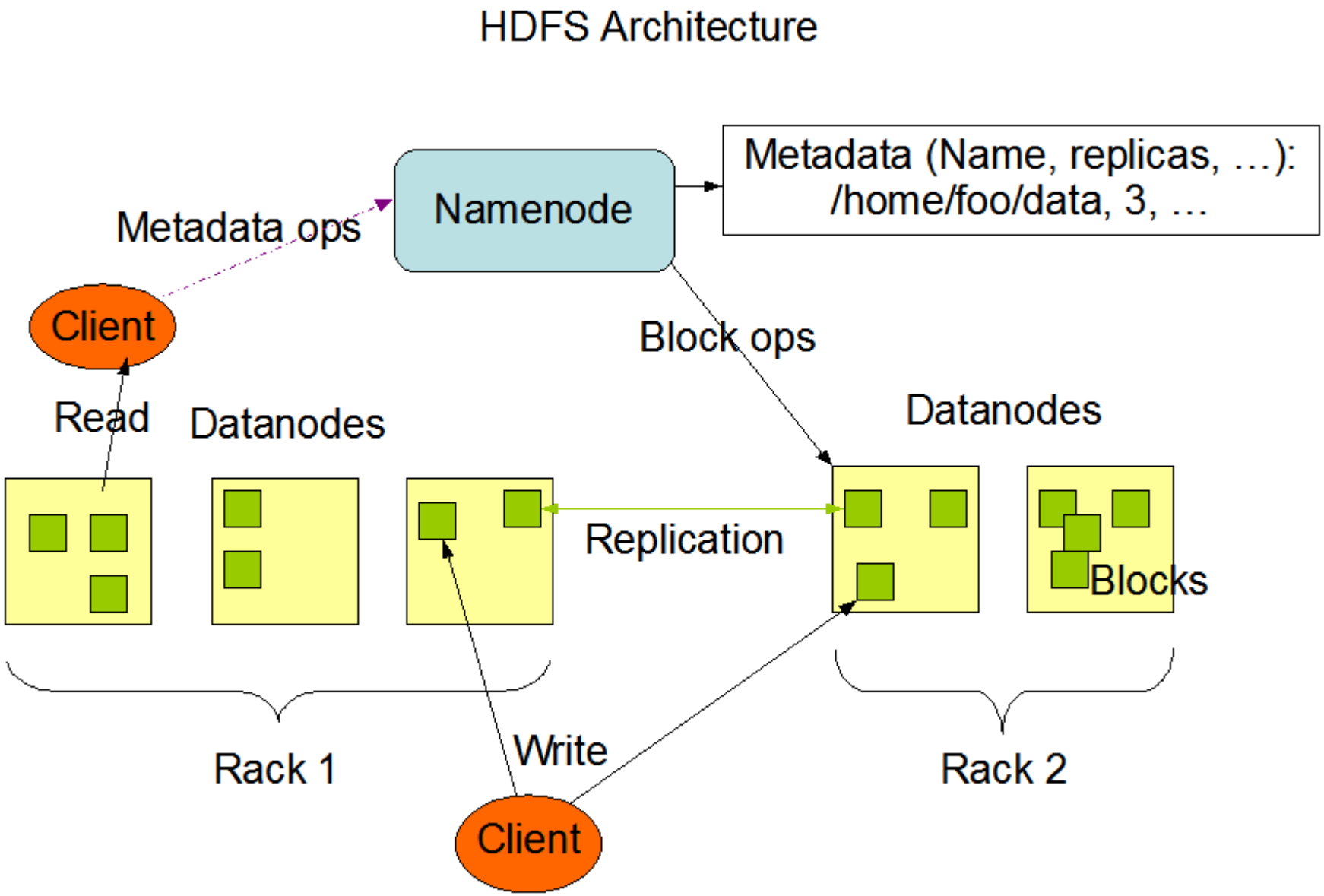
Viktige datastrukturar og filer

- NameNode*:*
  - Persistent i fil: *Namespace image*:
    For kvar fil/katalog (litt forenkla, i praksis organisert som inoder etc.): Path, Replication, ModificationTime, AccessTime, PreferredBlockSize, BlocksCount, FileSize, Permission, UserName, GroupName, [block identifiers]
  - I minne: Også kva datanodar blokkene ligg lagra på
  - I loggfiler: fil-metadata-operasjonar
- DataNode:
  - Ei fil for kvar datablokk, blokk-identifikator som del av filnamn
  - Separat metadata-fil for kvar datablokk, som m.a. inneheld sjekksummar

Klientar

- *Klient* aksesserer filsystemet på vegne av brukar/program gjennom kommunikasjon med NameNode og datanodar
- Java API
- Kommandolinjenivå (eksempel):

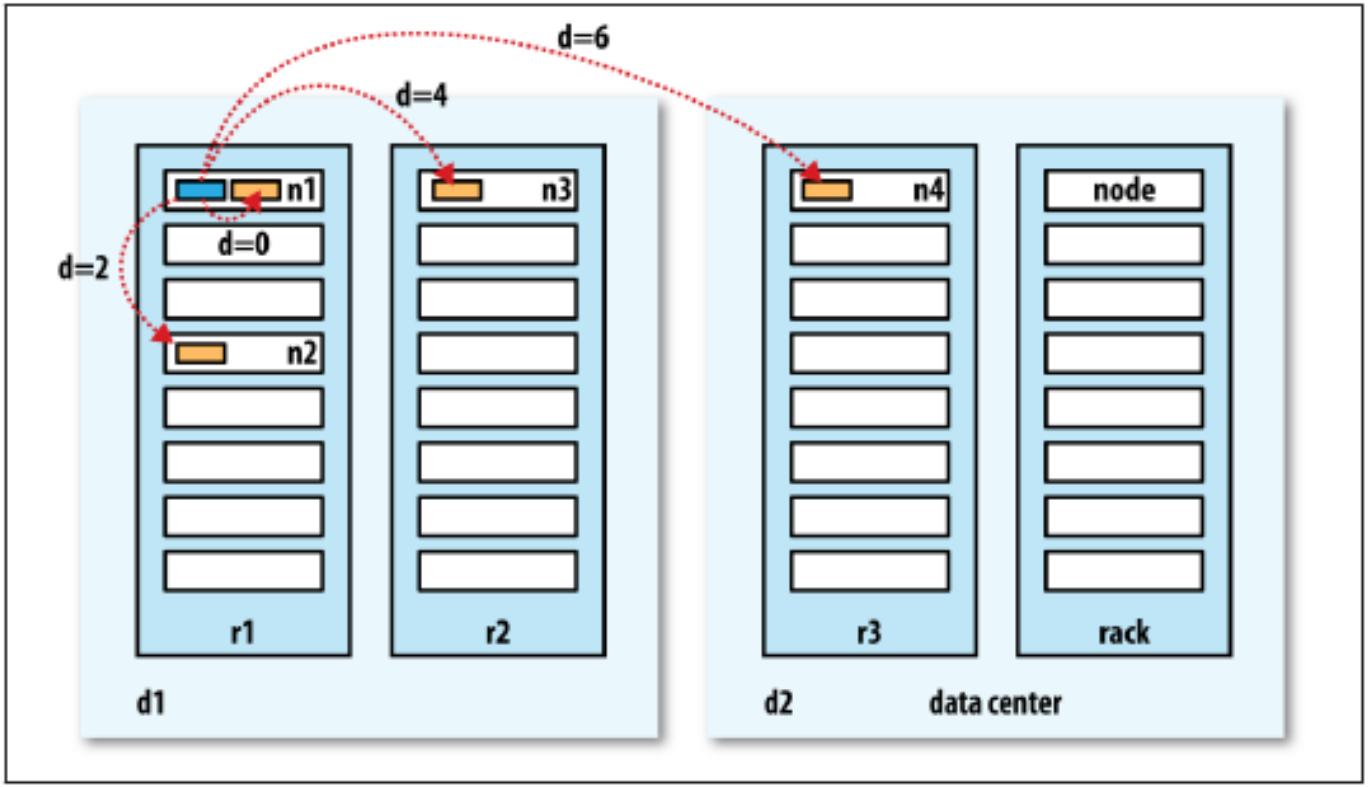
```
hadoop fs -ls
hadoop fs -mkdir testdir
hadoop fs -rmdir testdir
hadoop fs  -copyFromLocal test.txt
hadoop fs  -copyToLocal test.txt
sudo -u hdfs hdfs dfsadmin –report
```


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

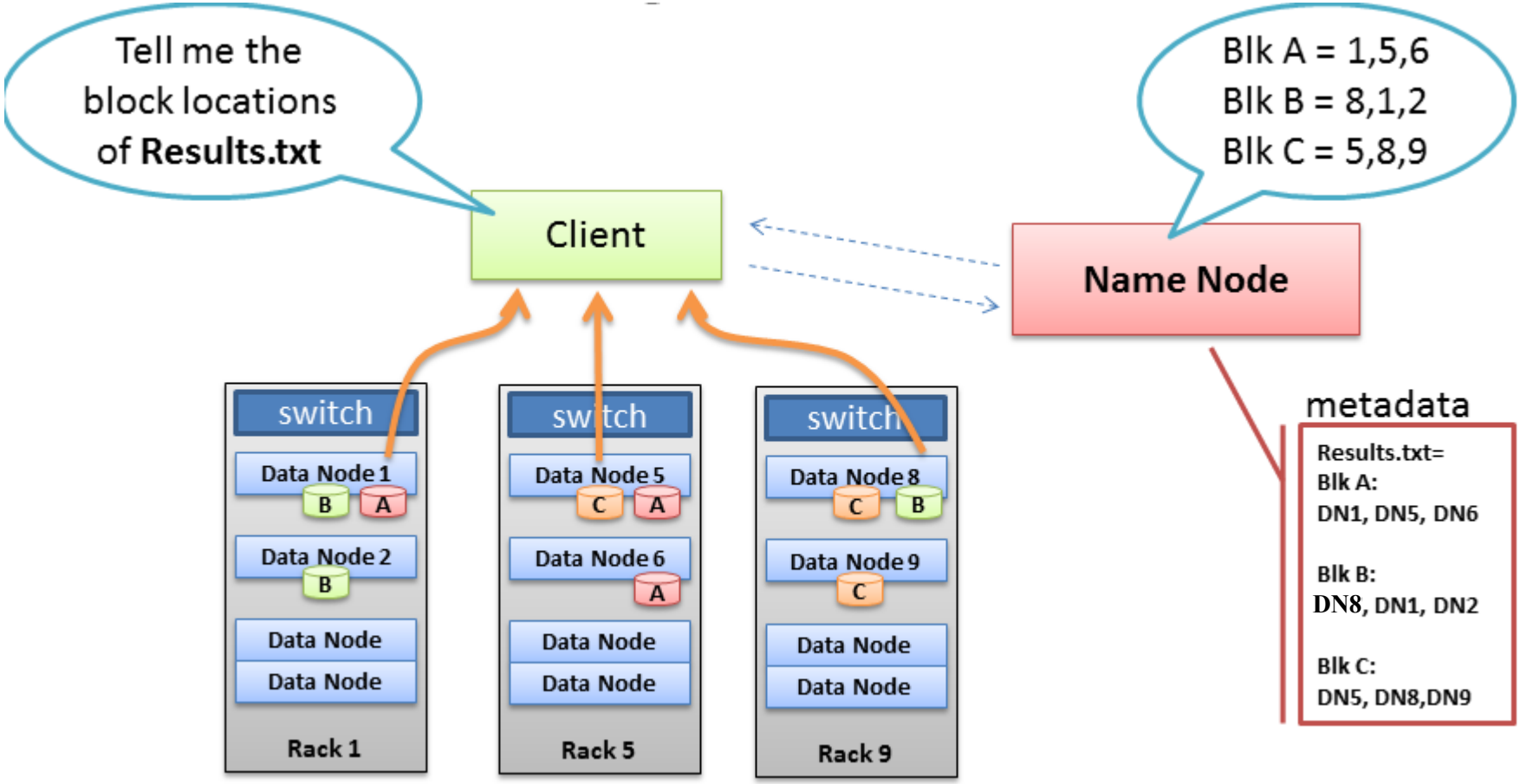
Feiltoleranse

- NameNode feiltoleranse vha:
  - Lagring av persistent tilstand på meir enn ein disk
  - Lagring av persistent tilstand på ikkje-lokal disk (andre nodar), evt. NFS-filsystem
  - Bruk av namespace image på Secondary NameNode ved gjenoppretting (recovery)
  - Hadoop 2: aktiv standby-NameNode for redusert nede-tid
- DataNode feiltoleranse:
  - Ikkje nødvendig pga. replikering av blokker
  - Bruk av sjekksum på kvar blokk for å detektere diskfeil

Nettverks-distanse i Hadoop



Lesing av fil



1) Finn blokk-lokasjon(ar)

2) Les fil (blokkar) sekvensielt, velg blokk som er ”nærmast” (jfr. nettverks-distanse)

Skriving til fil

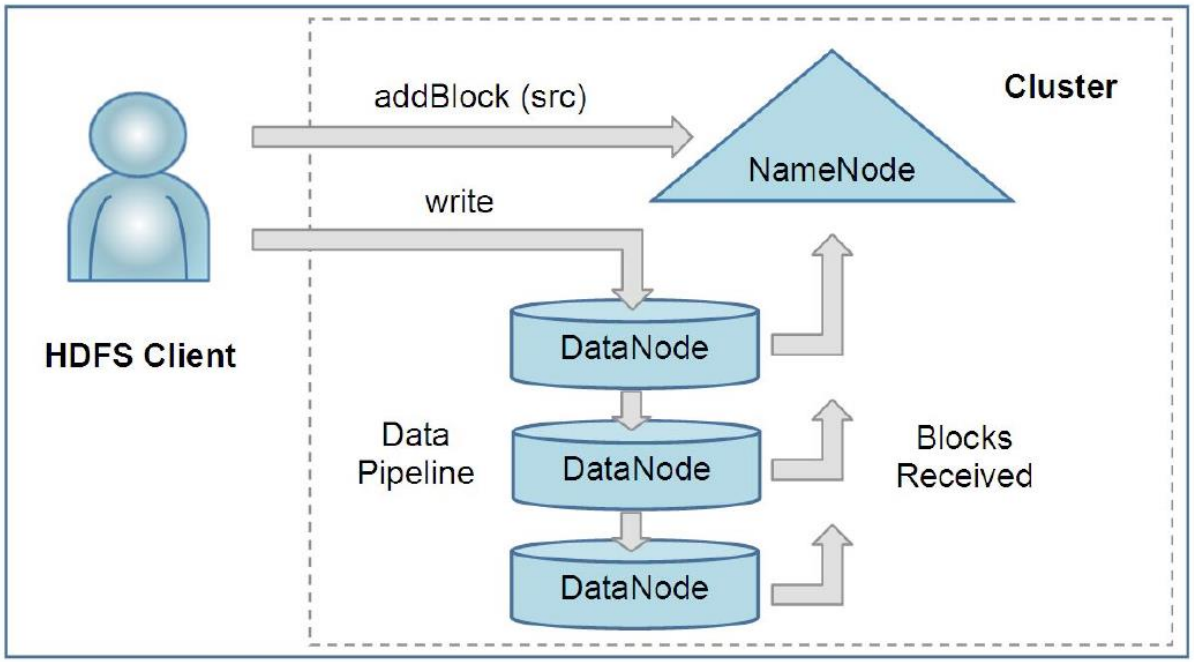
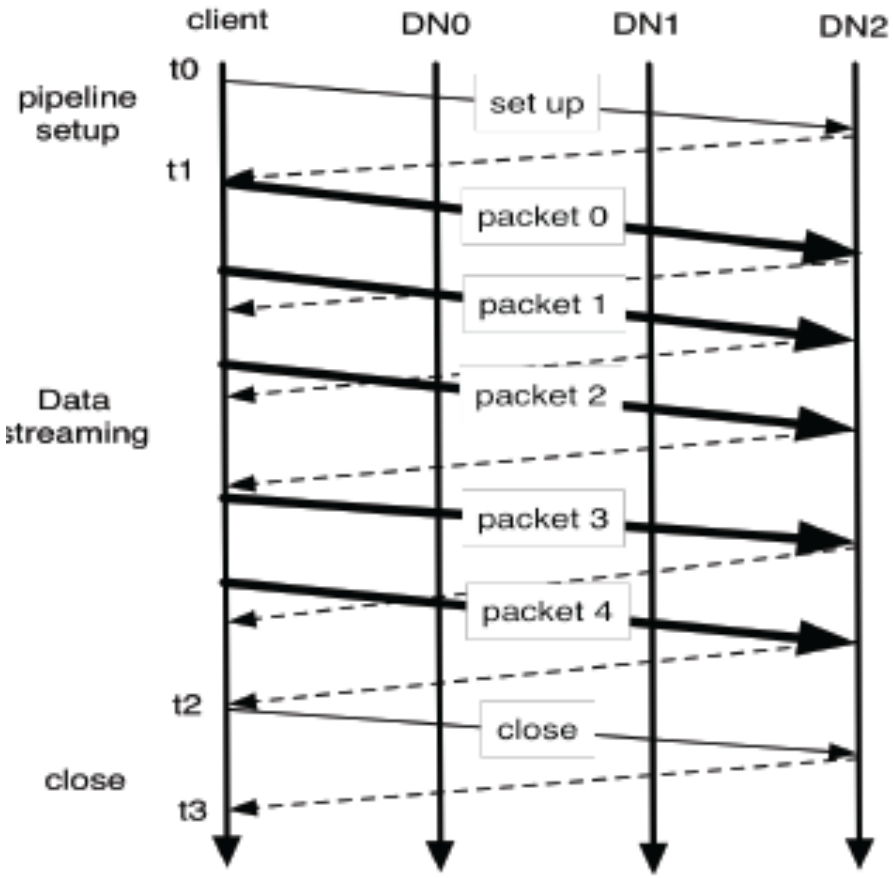


Figure 1. An HDFS client creates a new file by giving its path to the NameNode. For each block of the file, the NameNode returns a list of DataNodes to host its replicas. The client then pipelines data to the chosen DataNodes, which eventually confirm the creation of the block replicas to the NameNode.

Data-samleband ved skriving av blokk

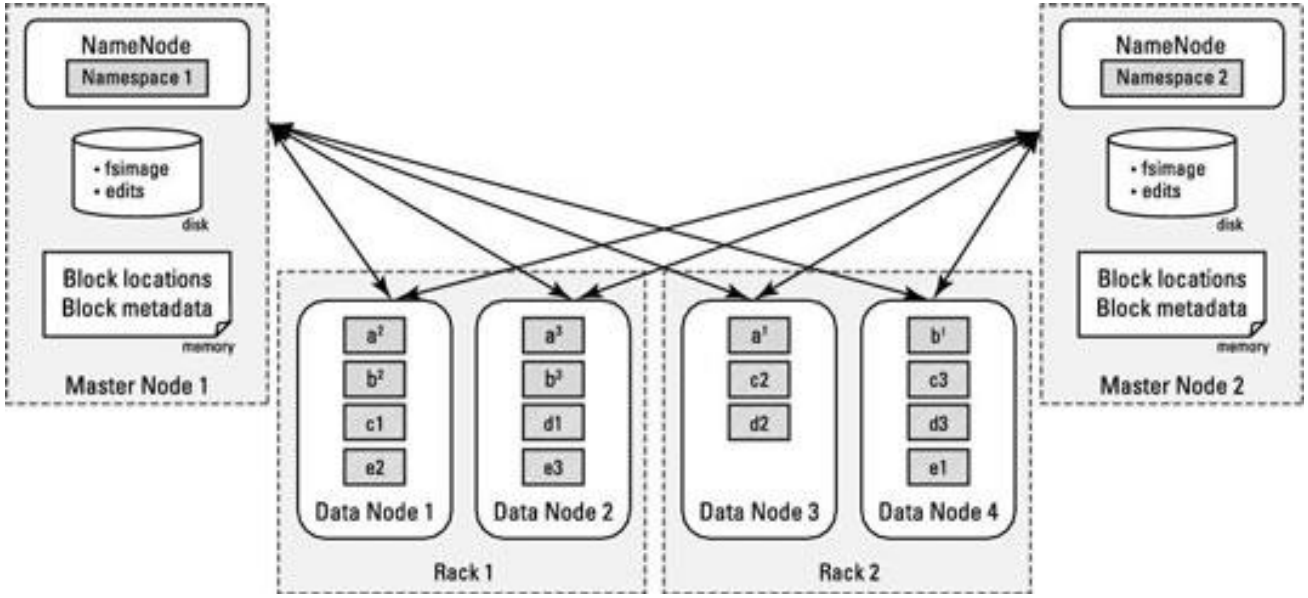


Vedlikehald

- Handsaming av replika
  - Over/under-replikering detektert vha. periodisk info frå datanodar
- Rebalansering
- *Block scanner:* verifisering av sjekksum på datablokker
  - Om feil: merka korrupt og nytt replikat vert etter kvart generert
- Dekommisjonering: kontrollert fjerning av datanodar
  - Først generer nye replikat av blokker som har vore lagra på noda, deretter kan noda fjernast

Nytt i HDFS/Hadoop 2.0 (1)

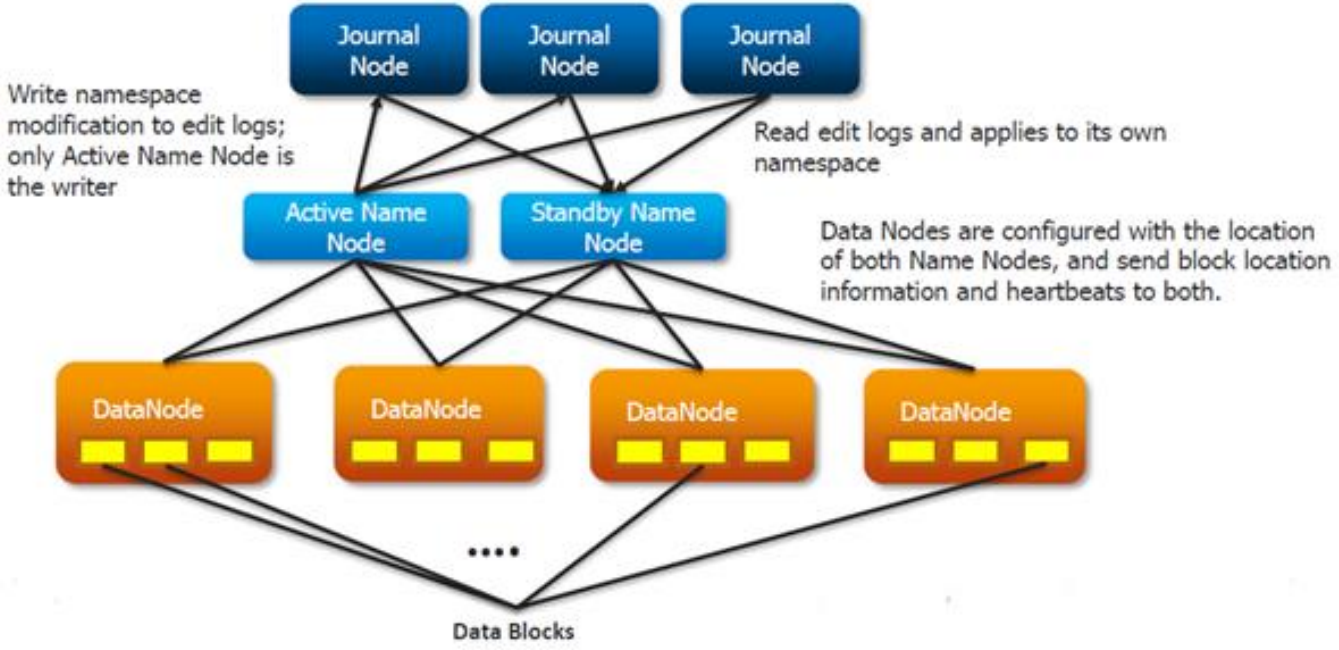
- Federering: distribuert/partisjonert namnerom (uavhengige NameNode'er)



- Namnerom: t.d. /usr og /home

Nytt i HDFS/Hadoop 2.0 (2)

- Betre støtte for høg-tilgjengelegheit:



- Betre støtte for sikkerheit (autentisering av klientar etc.)
- Kontinuerleg forbetring av yting
- Hadoop 3: Multiple namnenoder

Erfaringar

- Svært påliteleg
- Lite ressursar for drift, ca. ein operatør pr. 3K nodar
- Skalerbar: > 100 PB, > 4500 nodar i ei klynge