ScaDS DRESDEN LEIPZIG

UNIVERSITÄT LEIPZIG

AUS TRADITION
GRENZEN ÜBERSCHREITEN

Cloud Data Management

Kapitel 3: Verteilte Dateisysteme – Teil2

Dr. Eric Peukert

Wintersemester 2017

Universität Leipzig, Institut für Informatik
http://dbs.uni-leipzig.de

News

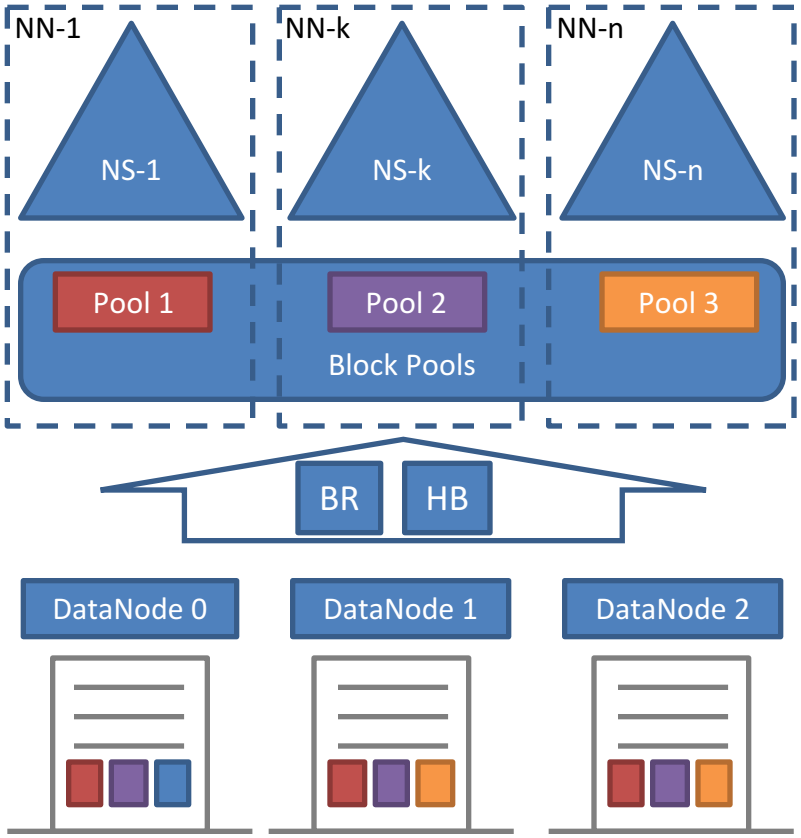
- https://www.networkworld.com/article/3236046/internet-of-things/future-proofing-your-career-with-iot.html

Fragen zu HDFS Teil1

- Footprint eines Blocks auf der Festplatte bei kleinen Dateien (kleiner als Blockgröße)

- Ablage von großen Blöcken im Dateisystem vs. Fragmentierung

HDFS – Federation
Überblick



- HDFS1: Cluster Speicher skaliert horizontal - der Namespace (NS) nicht
- Problematisch für große Installationen mit vielen (kleinen) Dateien
- Dateisystem-Operationen begrenzt durch NN Performanz
- Keine Isolation in Mehrbenutzerumgebung
- HDFS2: NS wird in mehrere Namespaces aufgeteilt
- Hinzufügen von NameNodes (NN-1, …, NN-n), die je einen Teil des NS (NS-1, …, NS-n) verwalten
- NS volume = NS und Block Pool (= Blöcke für NS)
- NS Volumes sind unabhängig
- DNs speichern Blöcke aus allen Pools und senden HB/BR an alle NNs

HDFS – Federation: Motivation

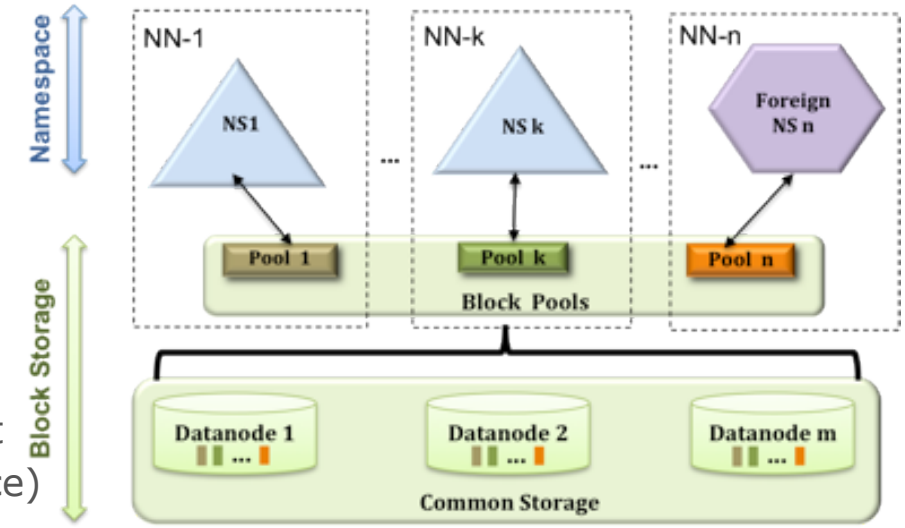
- Hinzufügen von DataNodes -> Speicherkapazität erhöhen
- Ein NameNode verwaltet Namespace des ganzen Clusters
  - Dateibaum+Metadaten und Block Locations aus Performanzgründen
  - komplett im RAM
  - Yahoo!
    - Blockgröße: 128MB
    - Durchschnittliche Datei ≈ 600Bytes Metadata (1 File+1-2 Blockobjekte)
    - 100Mio Dateien ≈ 60GB Metadata
    - 1PB Daten ≈ 1GB Metadata
- HS des Namenodes begrenzt horizontale Skalierbarkeit des Clusters
  - Vertikale Skalierung des Namenodes
- Durchsatz von Lese/Schreiboperationen durch 1 Knoten bestimmt
  - Mehr Clusterknoten
    - Mehr Block Reports
    - Mehr TaskTracker (agieren selbst als HDFS clients)

Bsp:
120K read ops/sec
6K write ops/sec

Quelle: [Federation]

HDFS Federation: Grundidee

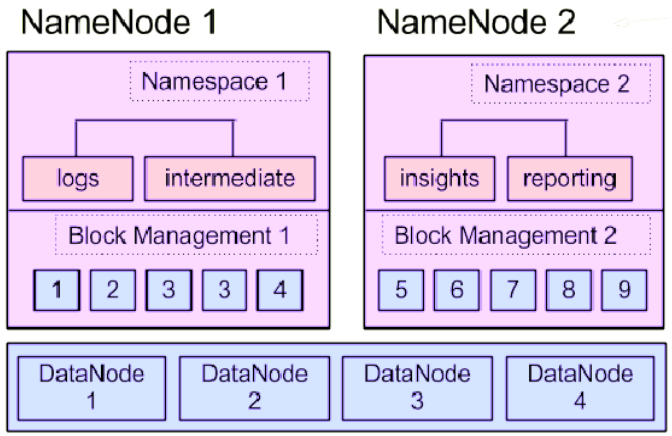
- Mehrere unabhängige Namenodes
  - Keine Kommunikation, jeder NN verwaltet einen eigenen Namespace
  - 1 Block Pool pro Namespace
  - DNs speichern Blöcke aller Namesspaces
- Vorteile
  - "Echte" Horizontale Skalierbarkeit
  - Performanz
    - Höherer Clusterdurchsatz ermöglicht höheren Parallelitätsgrad (MapReduce)
  - Multi-tenancy & Isolation
    - Verschiedene Mandanten können sich ein Cluster teilen
    - Verschiedene Anwendungen (z.B. Produktivanwendungen wie HBase) haben verschiedene Anforderungen → "eigener" NameNode
  - Block Pool Abstraktion ermöglicht Realisierung anderer verteilter Dateisysteme



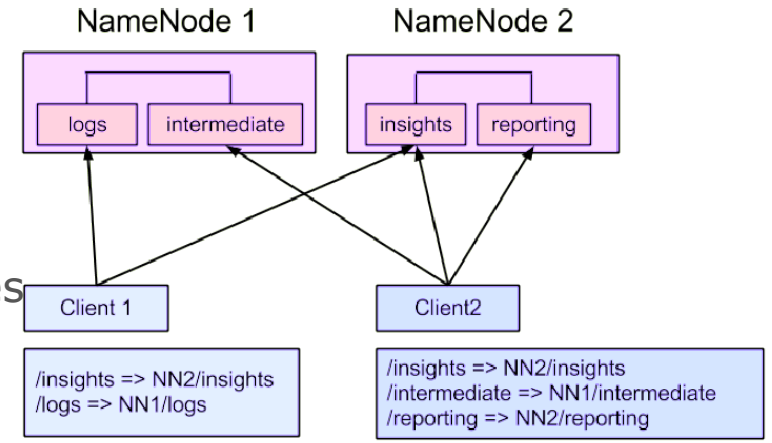
http://hortonworks.com/blog/an-introduction-to-hdfs-federation/

HDFS Federation: Mount Tables

- Aufteilung des globalen Namespaces auf mehrere NNs soll für Clients weitestgehend transparent sein
  - NN verwaltet nur Ausschnitt des NS
  - DN speichern Blöcke verschiedener NS
  - Abwärtskompatibilität

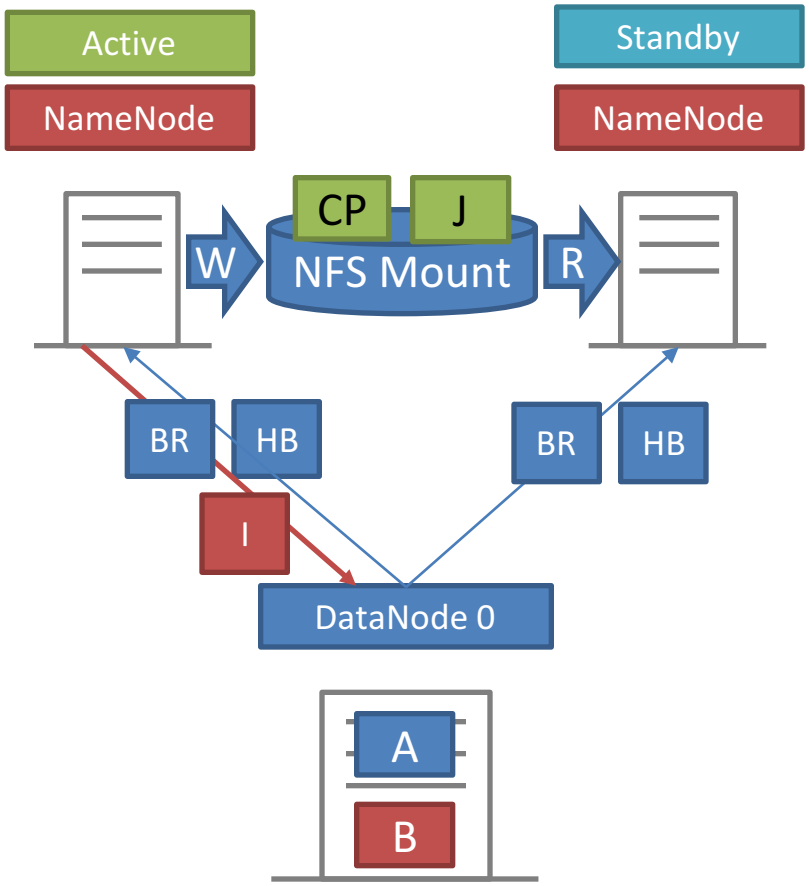


- *Client-side Mount-tables*
  - Bereitstellung einer globalen Sicht
  - Verwenden Client-seitiger Mount Tables ähnlich */etc/fstab* in unixartigen OS
    - Es können auch andere Dateisysteme (S3, Local) "gemounted" werden



Quelle: [Federation]

HDFS – High Availability (HA) Überblick



- HDFS1: NameNode ist SPOF
- HDFS2 HA (NFS): zwei redundante NameNodes in aktiv/passiv Konfiguration
- Nur ein NN ist aktiv und verantwortlich für alle Client Operationen
- Beide NNs haben Zugang zu einem gemeinsamen Directory (NFS)
- Aktiver NN schreibt in Journal, Standby NN überwacht Journal auf Änderungen und wendet Updates auf eigenen Namespace an
- DNs senden Block Reports und Heartbeats an beide NNs
- Standby Node erstellt Checkpoint
- HDFS2 HA (QJM)
- NFS Share ist SPOF
- Quorum Journal Manager
- Journal Nodes anstatt NFS share

HDFS High Availability: Motivation

- NameNode ist Single Point of Failure (SPOF)
  - Hält FS Tree, Metadaten und Block Locations im Hauptspeicher
  - Zusätzlich 2 Dateien im lokalen FS
    - *fsimage:* Schnappschuss des FS beim Start des NNs (**ohne Block Locations**)
    - *edits log:* Protokollierung aller Änderungen gegenüber *fsimage* (WAL)
  - Beim Neustart des NN wird *fsimage* mit *edits log* gemerged
    - NN wird selten neugestartet → großer *edits log* → Merge dauert lange
  - Vor HDFS HA: Secondary NameNode ("Checkpoint Node")
    - Secondary NN kopiert in regelmäßigen Abständen *fsimage* und *edit log* vom NN
    - Mergen von *fsimage* und *edits log*
    - Zurückkopieren der aktualisierten und Ersetzen der alten *fsimage* Datei
    - Im Fall eines geplanten Neustarts muss NN trotzdem auf Block Reports der einzeln DataNodes warten → **mehrere Stunden für große Cluster** (2000 nodes)
  - Defekt des NNs
    - **Änderungen seit letztem Checkpoint verloren**
    - Vor HDFS HA: redundante Speicherung von *fsimage* und *edits log* (versch. Platten /NFS)

HDFS HA: Grundidee

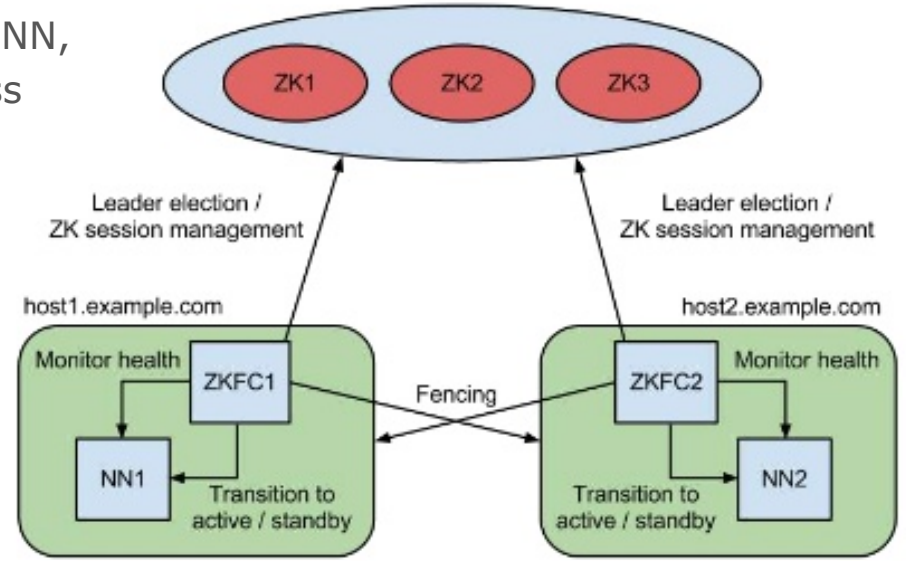
- Verwenden von 2 NN pro Cluster (Namespace bei Federation)
  - *Active NameNode* bearbeitet Client-Anfragen
  - *Standby NameNode* für **schnelles Failover** (“Hot Standby”)
  - Secondary NameNode (“Checkpoint Node”) nicht mehr benötigt
  - DataNodes senden Block Reports periodisch an beide Knoten
- Synchronisation von *fsimage* und *edits log*
  - Active NN schreibt/aktualisiert *edits log* **synchron** in **gemeinsam zugänglichen, hochverfügbaren Speicher** → 2 Varianten
    - NFS
    - Quorum-based
  - Standby NN überwacht diesen Speicher kontinuierlich und wendet Änderungen auf eigenem Filesystem-Abbild im HS an
- Failover
  - Nur wenige Änderungen nachzufahren, Block Locations aktuell
  - Manuell (Wartung): <3s, automatisch: 30-40s

HDFS HA: Client Failover

- Client Failover
  - “Umrouten” der HDFS Clients zum Active NN
  - Konfiguration eines Paares von NameNode-Adressen bei Clients
  - Verwenden der Adressen in konfigurierter Reihenfolge
    - Timeout: Retry anderer NN (vormals Standby jetzt Active)
    - Kontaktiert Client den Standby NN $\rightarrow$ Antwort-Code $\rightarrow$ Retry Active NN

HDFS HA: Automatic Failover

- Automatisches Failover
  - Verwendet Apache ZooKeeper-Quorum
    - Hochverfügbarer Koordinationsservice
    - Verteilte Synchronisation von Konfigurationen, Mitgliedschaft, Leader Election,
  - Jeder NN hat dauerhafte Verbindung zu ZooKeeper-Cluster
    - Active NN Crash → ZooKeeper-Session Timeout
      → ZooKeeper informiert Standby NN, dass Failover initiiert werden muss
  - Dazu ZooKeeper Failover Controller-Prozess (ZKFC) auf jedem NN
    - Active NN Election, ZooKeeper Session Management, Health Monitoring



Quelle:
[HDFSHA]

HDFS HA: Fencing

- Fencing
  - **Split Brain-Szenario**
    - Beide NNs nehmen an sie sind Active NN
    - Widersprüchliche Änderungen am FS Tree
    - Es darf zu einem Zeitpunkt nur einen NN geben
  - Lösung
    - Standby NN verhindert, dass anderer NN Änderungen am gemeinsamen Speicher macht
    - Nur ein NN darf active sein
    - Jeder Node kann anderem den Zugriff entziehen (aber sich nicht selbst wieder erlauben)
      - *sshfence: Kill des Active NN durch SSH Befehl*
      - *shell: Beliebiges Shell Script (STONITH: Shoot The Other Node In The Head)*

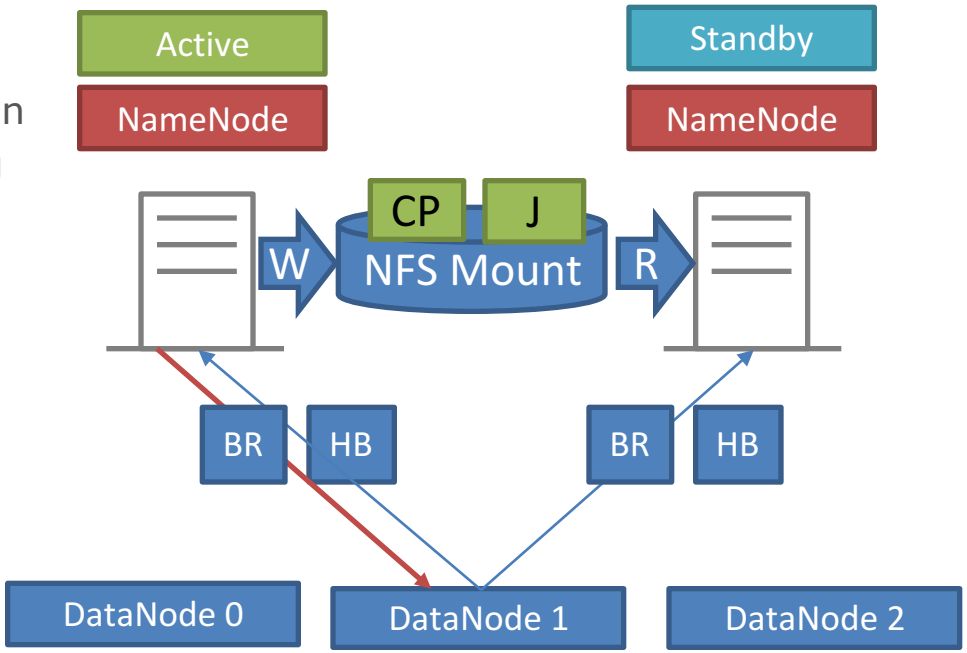
Fencing Beispiel Split-Brain

- Zwei NameNodes: **A** und **B** -> beide mit assoziierten ZKFC
- **ZKFC-A** process crash -> znode auf Zookeeper entfernt
- **NameNode A** process läuft weiter
- ZKFC B erfährt das znode entfernt wurde
- ZKFC B möchte NameNode B zu Active machen
  - ohne Fencing wären beide NameNodes gleichzeitig aktiv!

Quelle:
[HDFSHA]

HDFS HA: NFS

- Hochwertiges NAS
  - Auf Rechnern beider NNs ist ein bestimmtes Verzeichnis auf einem Dateiserver gemounted (Lese- und Schreibzugriff für die NN-Prozesse)
  - Hochverfügbarkeit
    - Mehrere Netzwerkanbindungen
    - Redundante Stromversorgung
    - Mehrere Platten / RAID



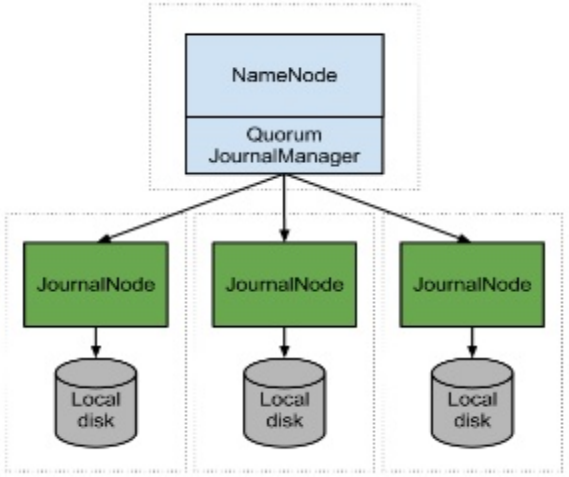
Quelle:
[HDFSHA]

HDFS HA: NFS Nachteile

- Nachteile NAS-Variante
  - Teure Spezialhardware
  - Komplexe Administration (Redundante Netzwerkverbindungen, Mount-Optionen, …)
  - NAS ist noch immer SPOF
- Design-Ziele Quorum-based
  - Verteilter replizierter Speicher (≠HDFS) für *edits log*
  - Ausfall/Nichtverfügbarkeit mehrere Knoten tolerabel, solange die Mehrheit der Knoten erreichbar ist
    - "Anzahl der tolerierbaren Fehler konfigurierbar"
  - Anzahl der Speicherknoten soll ohne Performanzeinbußen erhöht werden können
    - Durchsatz nicht vom langsamsten Knoten bestimmt
  - Verwenden der im Hadoop-Cluster verfügbaren Ressourcen

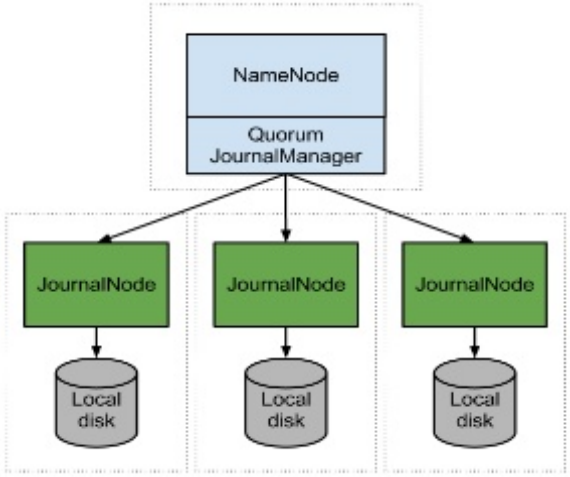
HDFS HA: Quorum-based

- Jeder NN kommuniziert mit einer Gruppe von *JournalNodes* (JNs)
  - Ungerade Anzahl von n JNs auf n Cluster-Knoten
  - Änderungen am *edits log* werden durch Active NN synchron zur Mehrheit der JNs geloggt
  - Standby NN überwacht JNs kontinuierlich
  - Im Falle eines Failovers liest Standby NN alle "ausstehenden" Änderungen von den Journal Nodes bevor er zum Active NN wird



[Quelle: HDFSHA]

HDFS HA: Quorum-based(2)



- Implizites Fencing - JNs erlauben zu einem Zeitpunkt nur einem NN Schreibzugriff
  - Vermeidet split brain (NN darf sich nicht selbst active setzen)
  - Jeder Quorum JournalManager hat eine eindeutige *epoch number, (*tot. Ordnung)
  - Jede Schreiboperation an JN trägt *epoch number*
  - Jeder JN speichert die größte zuletzt gesehene *epoch number*
  - Schreibanforderungen von NNs mit kleinerer *epoch number* werden ignoriert
  - Wenn NameNode zum Active NN wird generiert er mit Zustimmung der Mehrheit der JNs eine neue *epoch number* → implizites **Fencing**
    - Explizites Fencing zur Vermeidung der Beantwortung von Leseanfragen durchvorherigen Active NN

[Quelle: HDFSHA]

HDFS Zusammenfassung

- Verteilte, fehlertolerante Dateisysteme
  - Speicherschicht des Hadoop Ecosystems
  - Für Streaming großer Dateien
  - Fehlertoleranz durch Block-Replikation
  - Block Placement Policy und Balancierung
- HDFS2
  - *High Availability* zur Vermeidung /Reduzierung von Ausfallzeiten
  - *Federation* zur Vermeidung von NS Skalierbarkeitsproblemen

GFS/HDFS: Zusammenfassung

Eigenschaft	Technologie / Idee
Metadaten	
Instanzdaten	
Verfügbarkeit	
Lese-Operation	
Schreib-Operation	

Quellen und Literatur

- [GGL03] Ghemawat, Gobioff, and Leung: The Google File System. Symposium on Operating Systems Principles, 2003
- [W12] White, Tom. Hadoop: The Definitive Guide. 3rd Edition. O'Reilly. 2012.
- [S10] Shvachko et al. The Hadoop Distributed File System. MSST. 2010.
- http://hadoop.apache.org
  - [HDFS] http://hadoop.apache.org/hdfs/
- [HDFSHA]: http://de.slideshare.net/cloudera/ha-phase-2-with-atm-updates
- [Federation]: http://de.slideshare.net/AdamKawa/apache-hadoop-yarn-namenode-ha-hdfs-federation