Big Data in a Nutshell

Dr. Olaf Flebbe
of ät oflebbe.de

Zu mir

Bigdata Projekt, benutzt Apache Bigtop

Linux ‚seit Anfang‘ vor Minix/ATARI

Linuxtag 2001 ?

Promoviert in Computational Physics in Tü

Seit Jan 15 Apache Bigtop Committer

Hadoop

Big Data: Wie groß?

Big Data: So groß!

Viele PetaByte

Big Data

- Scale Up : Größere Computer
- Scale Out: Mehr Computer

Scale Up

- Immer schnellere, mehr CPU, shared RAM
- Problem wird irgendwann sehr sehr teuer
  - Compute (Crossbars: etwa 500 CPU, mehrere TB Hauptspeicher)
  - Storage (z.B. Fiberchannel) PB Proprietäre HW
- Programmiermodell: Threads, Shared Memory

Scale Out

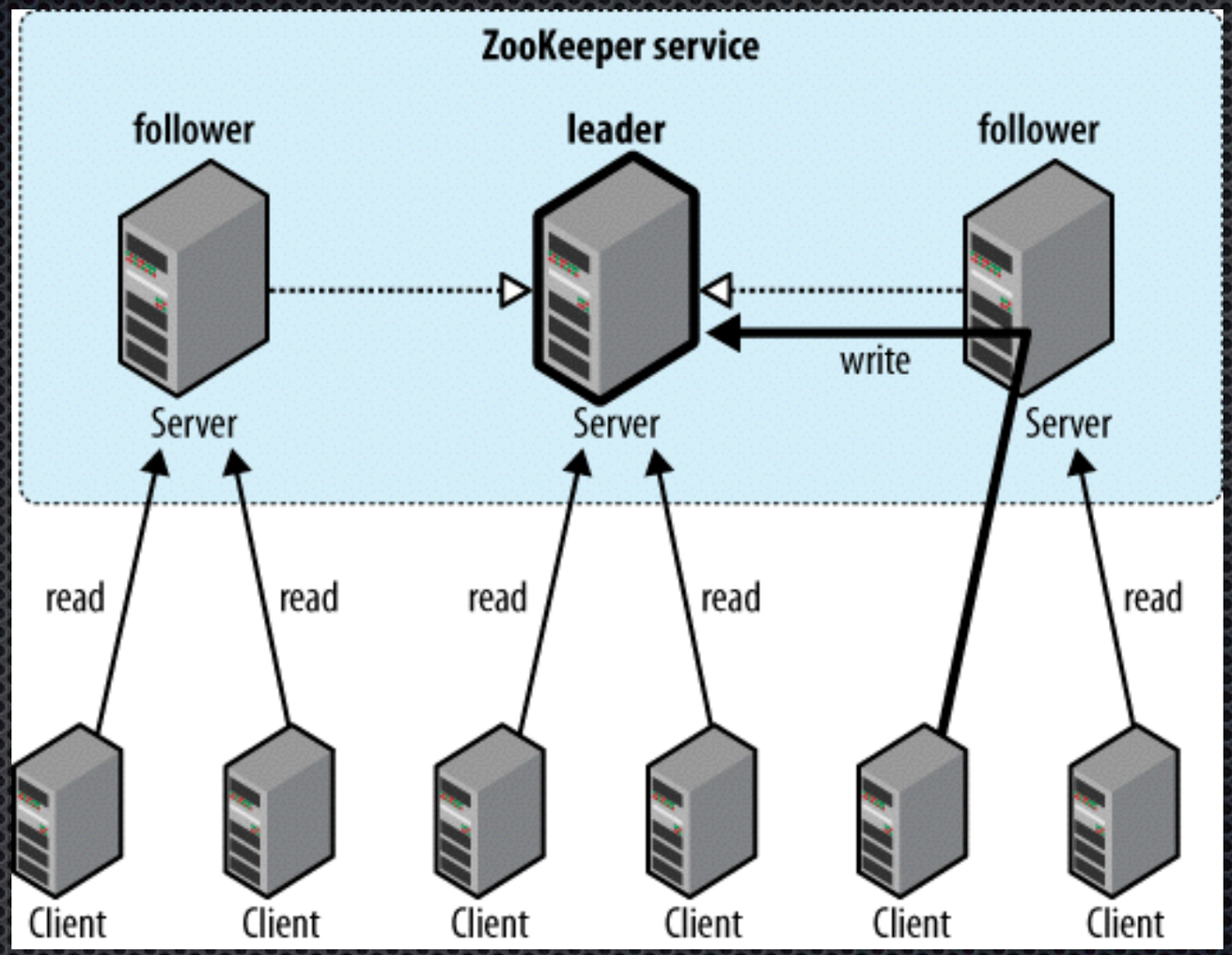
- Viele Commodity Rechner:
  - D.h. 2-Wege Server (CPU 6 Kerne) 10Gb Ethernet mit > 8 SATA Platten
- Facebook: Mehrere 1000 Server (Thema OpenCompute)
- Programmiermodell: naja…

Zookeeper

Verteilter Dienst zur Koordination von verteilten Diensten

zookeeper

- Koordinationsservice
- yafs
- znodes, schreibt in garantierter reihenfolge



Thomas Koch, Hadoop und Zookeeper
Linuxtag

Sequential Consistency - Updates from a client will be applied and seen by others in the order that they were sent

Atomicity - Updates either succeed or fail. No partial results

Single System Image - A single client will see the same view of the service regardless of the server that it connects to

Reliability - Once an update has been applied, it will persist from that time forward until a client overwrites the update

Timeliness - The clients view of the system is guaranteed to be up-to-date within a certain time bound

Keine Zombies: Es ZooKeeper Server ist entweder Teil des Quorums oder stumm

Hadoop

- Skalierbare Datenlokale Clusteranwendungen:
  - HDFS (Hadoop File System)
  - Map Reduce
  - Job/Resourcemanagement (YARN)
- Nicht Daten, sondern Algorithmen verteilen

HDFS

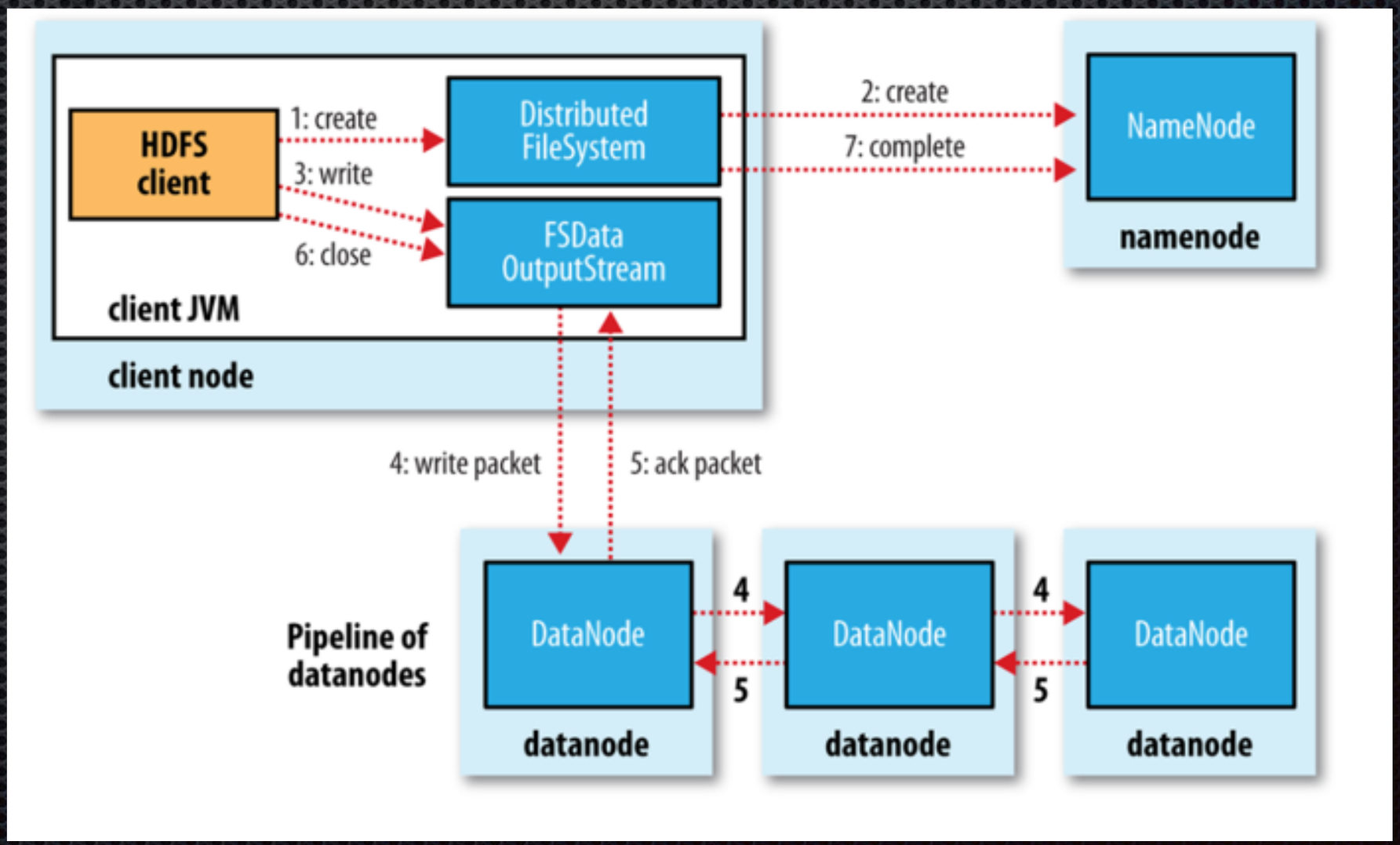
- Clusterfilesystem, POSIX ähnlich
- Snapshots
- ACL
- Nicht besonders schnell, aber extrem robust
- Keine besondere Hardware Anforderungen
- Integration mit Kerberos für Authentisierung
- Sehr Skalierbar
- Besonderheit: Datenlokalität. Ein Client holt die Daten möglichst vom gleichen Node, oder vom gleichen Rack, erst dann von einem entfernten Rack.
- Daten werden redundant gehalten (Meist 3) davon möglichst eine Kopie auf anderem Rack

HDFS

- Auf Linux verwendet man ein ext4
- Von Jeder Datenplatte wird Mountpunkt als Datenstore konfiguriert. HDFS verteilt die Datenblöcke auf die Mountpunkte.
- Für jeden HDFS Datenblock wird eine CRC berechnet und im Hintergrund auf Änderungen gescannt.

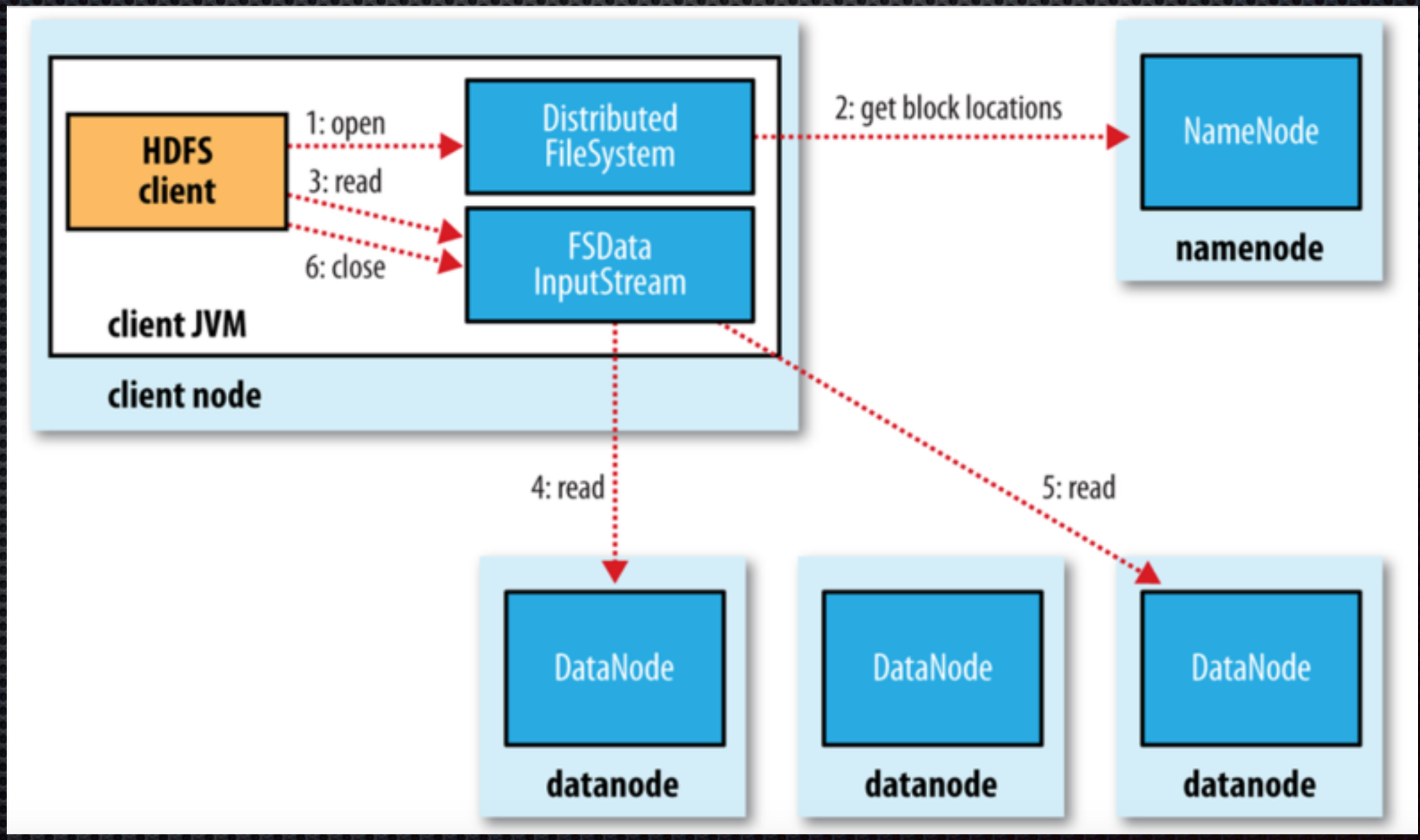
HDFS

- Schreibvorgang:



HDFS

- Lesen:


HDFS client
1: open
3: read
6: close
client JVM
client node
Distributed FileSystem
FSData InputStream
2: get block locations
NameNode
namenode
4: read
5: read
DataNode
datanode
DataNode
datanode
DataNode
datanode

HDFS

- journalling
- HA für namenodes
- Log wird geclustert auf ein Quorum geschrieben, nicht ausgebremst vom langsamsten

HDFS aus Sysadmin Sicht

- Umständlich zu bedienen.
  - hdfs dfs -help
  - Undurchsichtige Verzeichnisse …
- Store vergrößern: Kein Problem: Datanode starten, fertig.
- Festplatte kaputt. Kein Problem, Austausch fertig.
- Datanode kaputt: Kein Problem, wegwerfen.
- Unbalanziert?

HDFS Developer Sicht

- Java API für Filesystem, POSIX nachempfunden
- Maven Integration

HDFS Dienste

- Grundlegend:
  - 1 Namenode (Metadaten: Welcher Block ist wo?)
  - N Datanodes (Nutzdatenservice)

Optimierungen

- Shortcut Reader
- Normal Disk Block -> Datanode -> TCP Sock -> Algorithm
- Shortcut Disk Block -> Data Node (shared mem) -> Algorithm (shared mem)

Hadoop

- HDFS (Hadoop File System)
- YARN (Yet Another Resource Negotiator)

YARN Dienste

- Grundlegend:
  - 1 Resource Manager (RM: Was für Compute Ressourcen stehen bereit)
  - N node Manager (Jobstarter best practice auf datanode)
  - Application Manager (AM: wird als erster Job gestartet, und fragt die weiteren container an)

Resource Manager

- Verantwortlich die angeforderten Ressourcen zu verwalten
- Platziert Jobs dort, wo Daten lokal sind
  - gleicher Node
  - selbes Rack
  - egal
- Scheduler ist pluggable
  - Eingebaut Capacity Scheduler inkl. Preemption, …

Node Manager

- Startet die Jobs
- Sperrt diese in Container ein (Linux!)

Yarn Jobs

- In der Regel MapReduce Jobs starten/verwalten
- Meistaufgerufene Programm „pi“

Hive

- Erzeugt aus „SQL“ map-reduce jobs
- Daten liegen im HDFS, lokale Algorithmen

Hive Optimierungen

- partitionierung von Daten
- Predicate Pushdown: mapping auf Dateinamen
- Tez (Container reuse)
- File Formate:
  - ORC, Parquet, ...

Bigdata als Distribution

- Distributionen
  - Hortonworks, Cloudera; Pivotal, ODP: Apache Projekte verpacken mit install Tool
  - MapR: Reeimplementierung in native Code
  - Apache Bigtop
- Typischer Inhalt: Zookeeper, Hadoop (HDFS/Yarn/Mapreduce), HBase, Hive, Hue, Flume, Kafka, Sqoop, Spark

Apache Bigtop

- Buildskripte
- Buildumgebung mit puppet aufsetzen
- Paketierung (DEB, RPM)
- Tests (vagrant)
- Beispiel App
- Deployment (puppet)
- Konfiguration (hiera)

Bigtop

- Unglaublich positive Erfahrung in einem Apache Projekt zu arbeiten