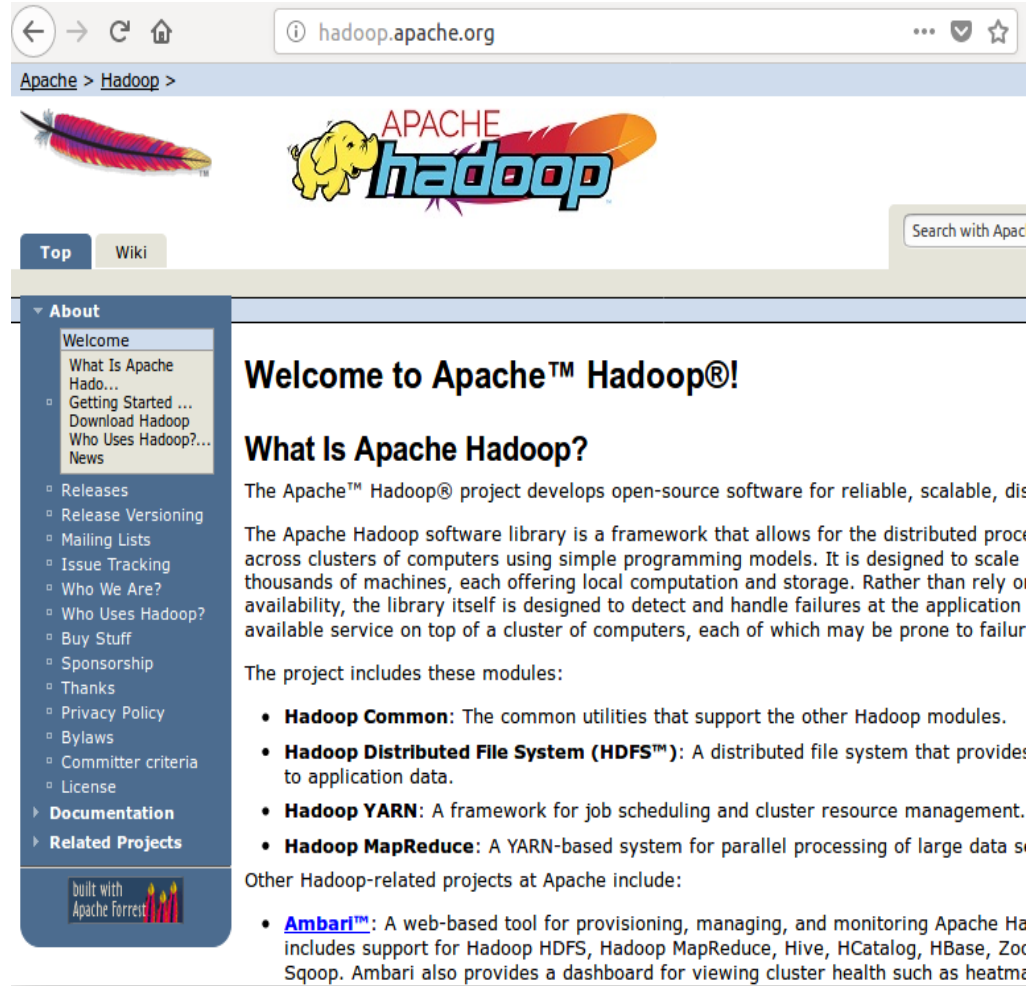


Tom White



← → ↺ 🏠 hadoop.apache.org ... 🔍 ☆

[Apache](#) > [Hadoop](#) >

 **APACHE**
hadoop

[Top](#) [Wiki](#)

▼ **About**

- Welcome
 - What Is Apache Hadoop?
 - Getting Started ...
 - Download Hadoop
 - Who Uses Hadoop?...
 - News
- Releases
- Release Versioning
- Mailing Lists
- Issue Tracking
- Who We Are?
- Who Uses Hadoop?
- Buy Stuff
- Sponsorship
- Thanks
- Privacy Policy
- Bylaws
- Committer criteria
- License
- Documentation
- Related Projects

built with Apache Forrest

Welcome to Apache™ Hadoop®!

What Is Apache Hadoop?

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale to thousands of machines, each offering local computation and storage. Rather than rely on hardware to provide high availability, the library itself is designed to detect and handle failures at the application level, so that the service continues to run even if some machines go down.

The project includes these modules:

- **Hadoop Common:** The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™):** A distributed file system that provides high throughput streaming access to application data.
- **Hadoop YARN:** A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce:** A YARN-based system for parallel processing of large data sets.

Other Hadoop-related projects at Apache include:

- **Ambari™:** A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, and Sqoop. Ambari also provides a dashboard for viewing cluster health such as heatmaps.

- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
 - LPGL Lizenzmodell
- Dies ist ein Programmier-Seminar
 - Damit werden die Inhalte durch Übungen vertieft und verinnerlicht
 - Musterbeispiele werden zur Verfügung gestellt
 - Diese können am Ende des Seminars als ZIP-Datei kopiert werden
 - USB-Stick oder ähnliches
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
- Konventionen
 - Befehle werden in `Courier-Schriftart` dargestellt
 - Dateinamen werden in *`kursiver Courier-Schriftart`* dargestellt
 - Links werden in `unterstrichener Courier-Schriftart` dargestellt
 - Zitate werden in *"Anführungszeichen kursiv"* formatiert, die Quellenangabe steht eingerückt darunter

Apache Hadoop

Programmierung

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6

81373 München

Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.

Big Data, Cluster und Skalierbarkeit	1
Einführung in Apache Hadoop	2
HDFS	3
Map Reduce	4
Werkzeuge und Produkte des Hadoop-Ökosystems	5

1

DIE PROBLEMSTELLUNG

1.1

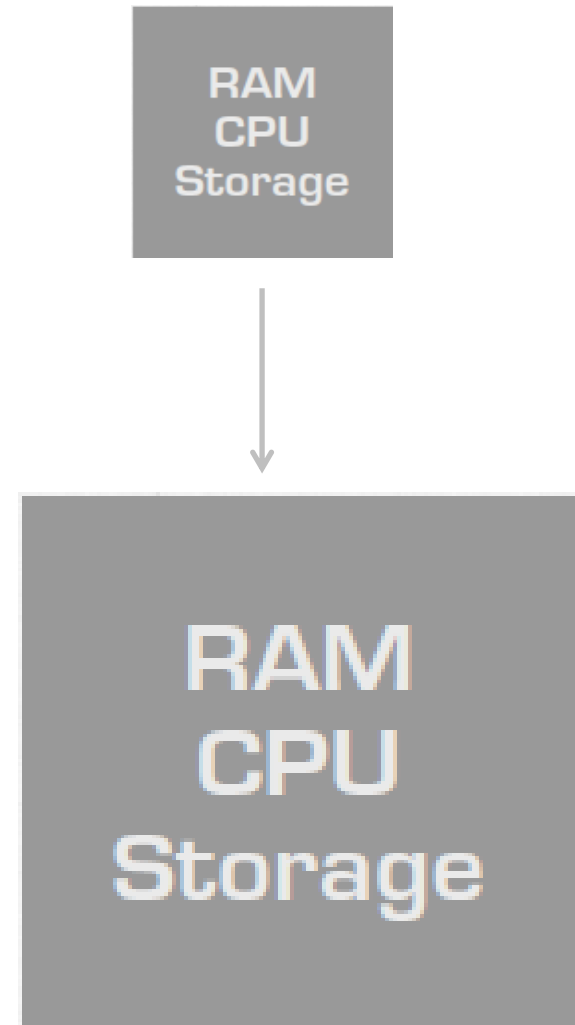
BIG & FAST DATA

- Definitionsversuche:
 - Physikalisch
 - Terabyte
 - System
 - Übersteigt den Bedarf eines normalen Speichermediums
 - Technisch
 - Datenverarbeitung beginnt, auf Grund der Größe Probleme zu bereiten
- Allerdings:
 - Keine wirklich eindeutige Definition erkennbar
 - Grenzen können sich verschieben

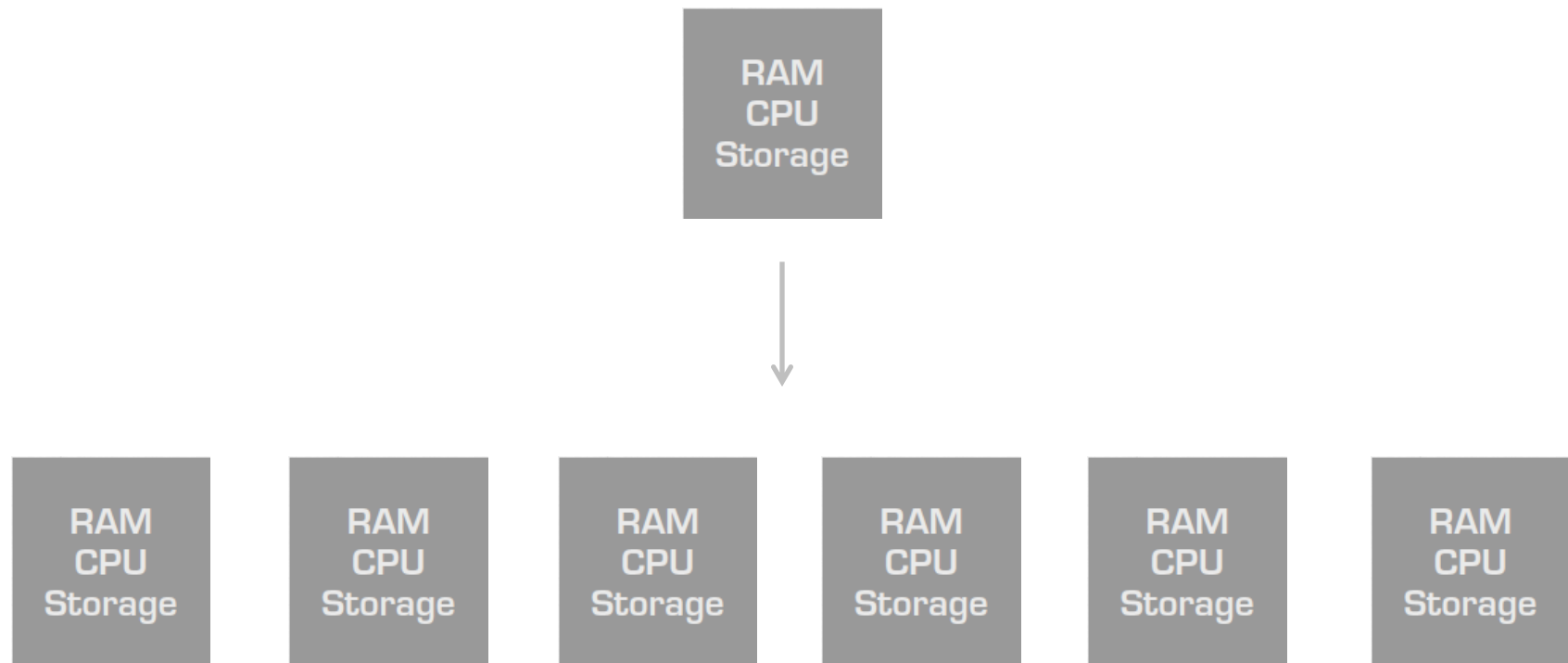
- Völlig verschiedene Bereiche:
 - DNS-Server für Internet
 - Indizierung von Web Seiten für Suchmaschinen
 - Google
 - Monitoring von Seitenzugriffen, aber auch Metrik-Informationen in großen Server-Systemen
 - Content Management und Waren-Systeme
 - Amazon
 - Social Media
 - Facebook

- Zur Ablage und Auswertung von Daten sind selbst mächtige Server-Maschinen auf Dauer überfordert
- Die Datenhaltung wird deshalb auf mehrere Server verteilt
- Die Lastverteilung übernimmt ein simpler Load Balancer oder ein komplexer "Master"
 - Aus Client-Sicht heraus tritt der Cluster damit immer noch als eine Einheit auf
- Im Optimum ist der Cluster damit eine quasi unendliche Datensenke mit beliebige skalierbarer Rechnerkapazität
 - Damit eine Vorstufe zur "Cloud"

- Grenzen sind klar definiert
 - Kosten
 - Aktuelle Hardware-Grenzen
 - Anforderungen an Ausfallsicherheit

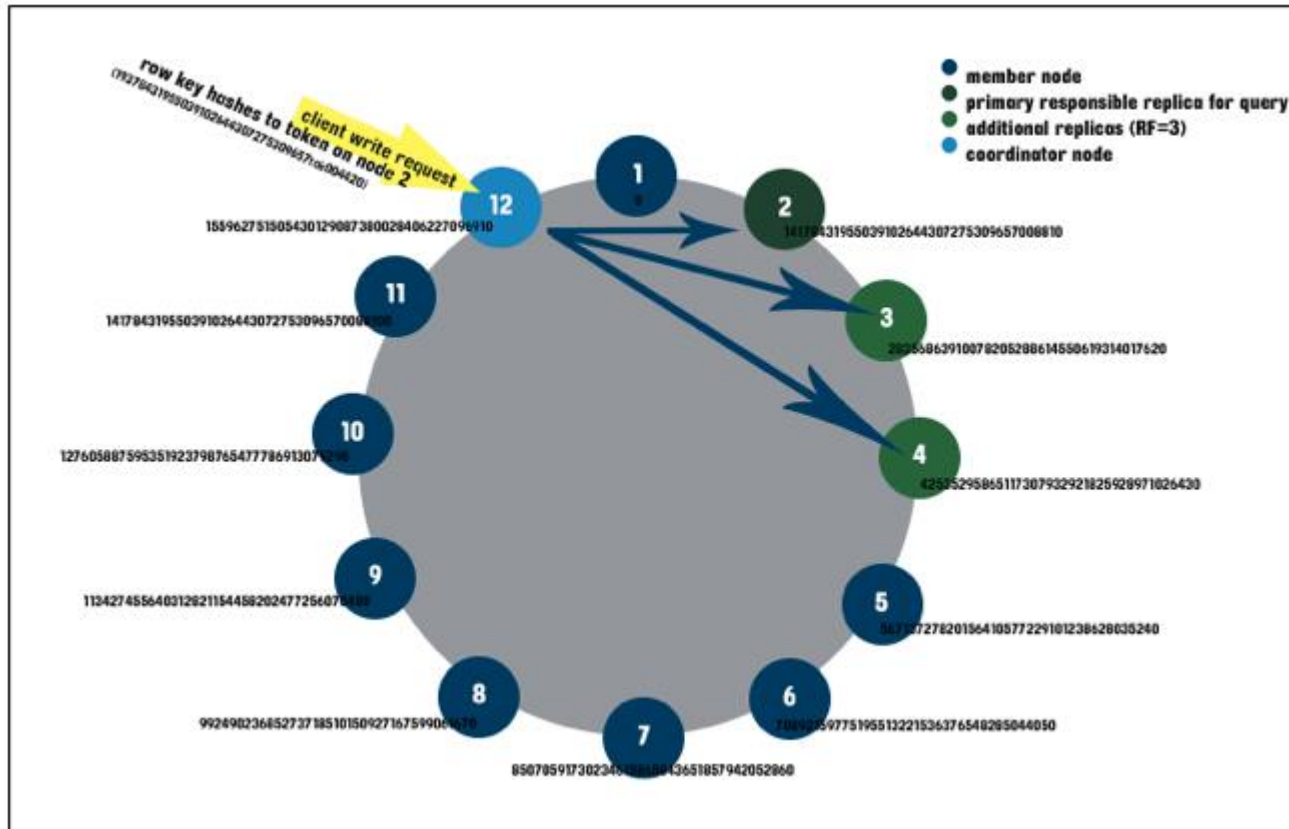


- Grenzen
 - Ausfallsicherheit fordert Replikation über Netzwerk
 - System-Administration und Überwachung



- Die Daten werden auf Grund eindeutiger Hashes auf einen "Ring" von einzelnen Servern verteilt
- Es gibt hierbei keinen Master
 - Jeder Knoten kann auf den Server delegieren, der die Daten wirklich hält

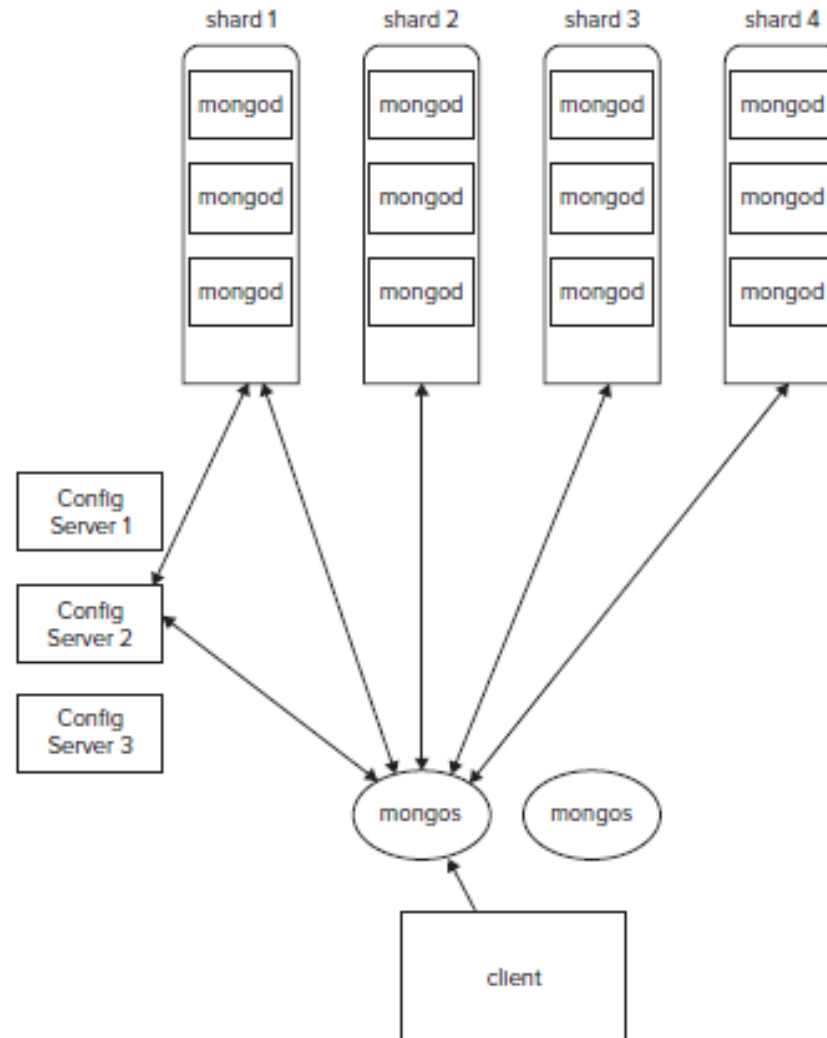
Beispiel: Apache Cassandra Ring



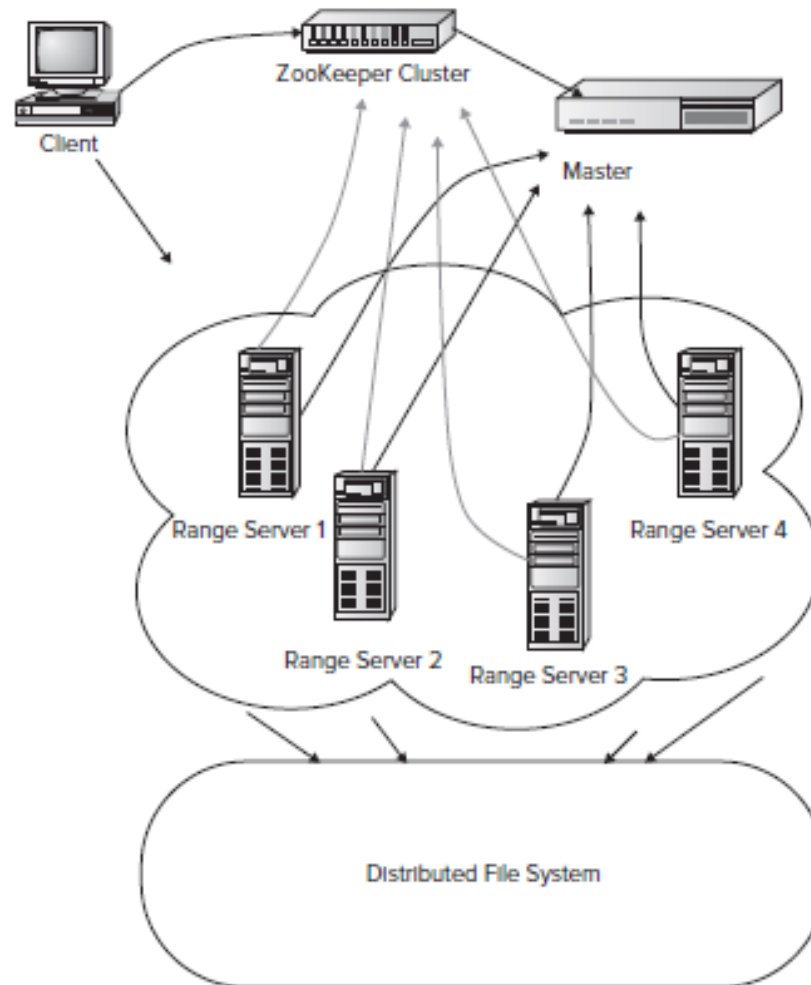
Quelle: <http://www.planetcassandra.org/blog/node-of-the-rings-fellowship-of-the-clusters-or-understanding-how-cassandra-stores-data/>

- Daten werden in sinnvoller, wohl-definierter Art und Weise auf mehrere Rechner verteilt
- Eine besondere Herausforderung stellt sich im Cluster-Betrieb
 - Das Hinzufügen eines Knoten kann ein „Resharding“ auslösen
 - Dies kann zu beträchtlicher Last auf dem Server führen
 - Administrativer Vorgang!

Beispiel: Sharding und die Mongo DB



Beispiel: Apache Hadoop

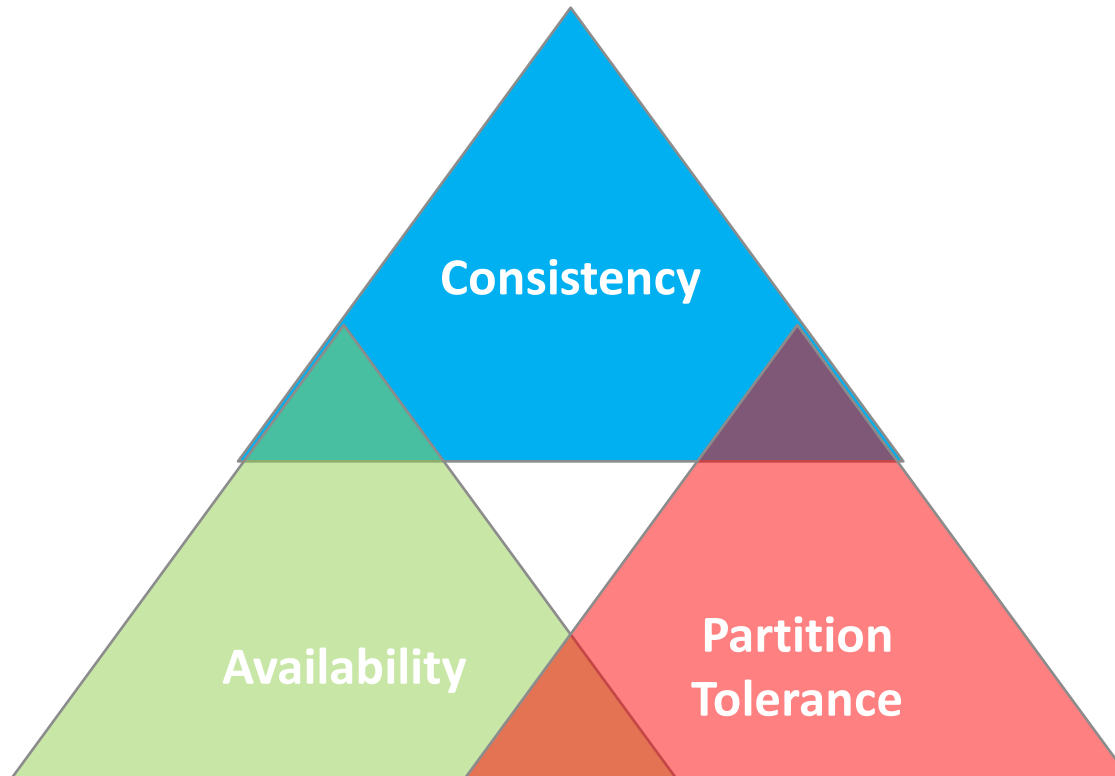


1.2

DAS CAP-THEOREM

- Consistency
 - Alle Knoten haben jederzeit den selben Informationsstand
- Availability
 - Jeder Client bekommt garantiert Antwort
 - Solange zumindest ein Knoten im Cluster läuft
- Partition Tolerance
 - Das System toleriert
 - Den Ausfall von Knoten
 - Den Ausfall des Netzwerks zwischen Knoten

Keine gemeinsame Schnittmenge!



- Consistency
 - Alle Knoten haben **jederzeit** den selben Informationsstand
- Eventually Consistent
 - Änderungen des Datenbestandes werden zeitlich versetzt an die anderen Knoten propagiert
- BASE
 - **B**asically **A**vailable
 - **S**oft state
 - **E**ventual consistency

- ACID
 - Starke Konsistenz der Daten
 - Transaktionssteuerung
 - Transaction-Isolation
 - Bei Bedarf Zwei-Phasen-Commit
 - Relativ komplexe Entwicklung
- BASE
 - Schwache Konsistenz
 - Hohe Verfügbarkeit
 - Schnelligkeit
 - Bei Bedarf "Fire-and-forget"
 - Leichtere Entwicklung

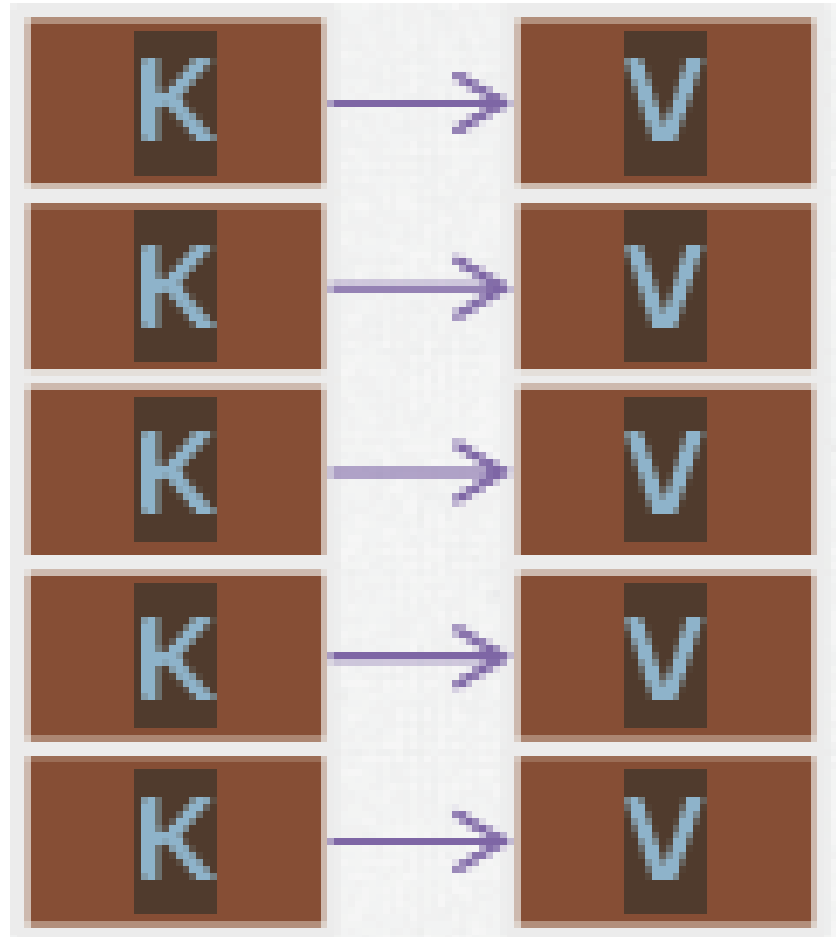
1.3

NOSQL

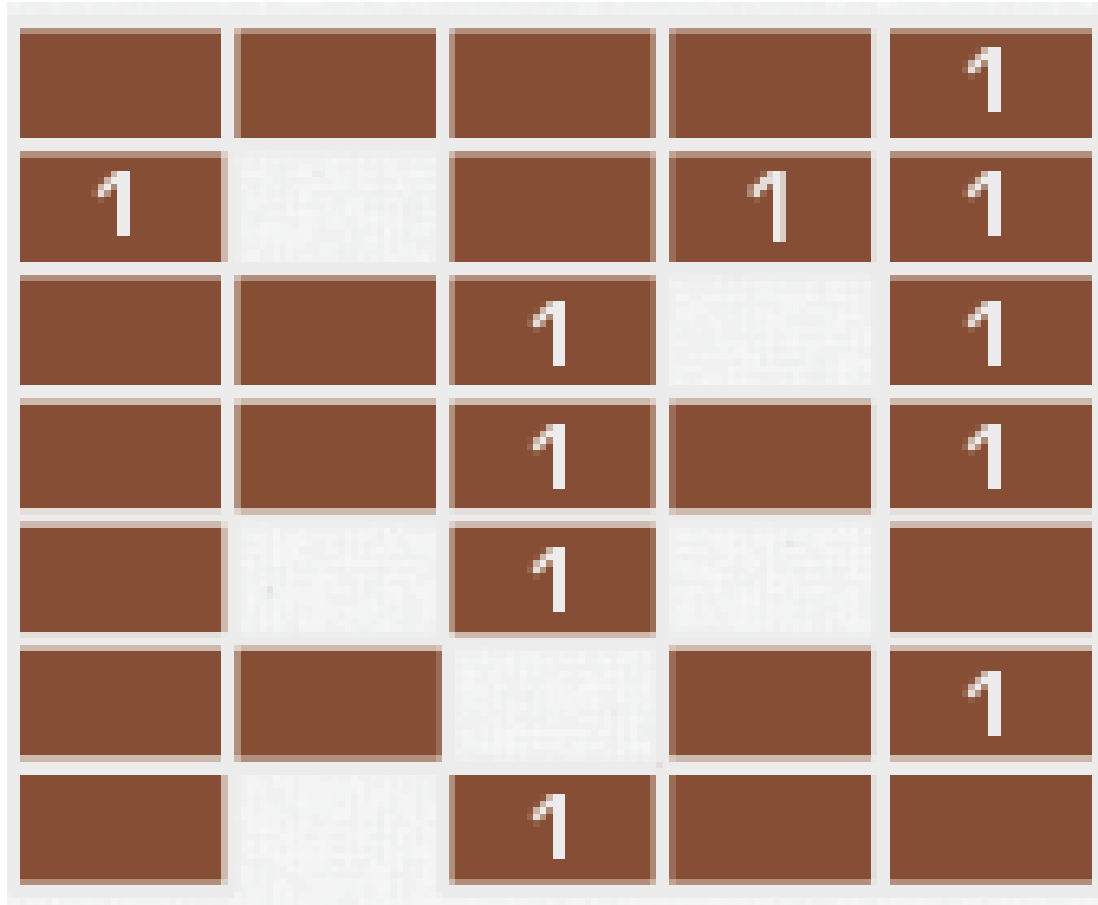
- No SQL!
 - Ursprüngliche Bedeutung
 - Etwas dogmatische Ablehnung relationaler Datenbank-Prinzipien
- Not only SQL
 - Schon deutlich abgeschwächt
 - Reduktion auf geeignete Problemstellungen
- No/ Not only relational
 - Der eigentlich richtige Begriff
 - Fokussierung auf die Daten-Modellierung, nicht auf die Abfragesprache

- Key/Value
- Column-orientiert
- Dokumenten-orientiert
- Graphen-orientiert

- Ablage von Inhalten (=Value) unter einem eindeutigen Schlüssel (key)
 - Flache Struktur
 - Values werden als Byte-Array betrachtet
- Teilweise Verzicht auf Persistierung der Daten
 - Cache-Systeme
- Auslegung auf eine Vielzahl konkurrierender Zugriffe

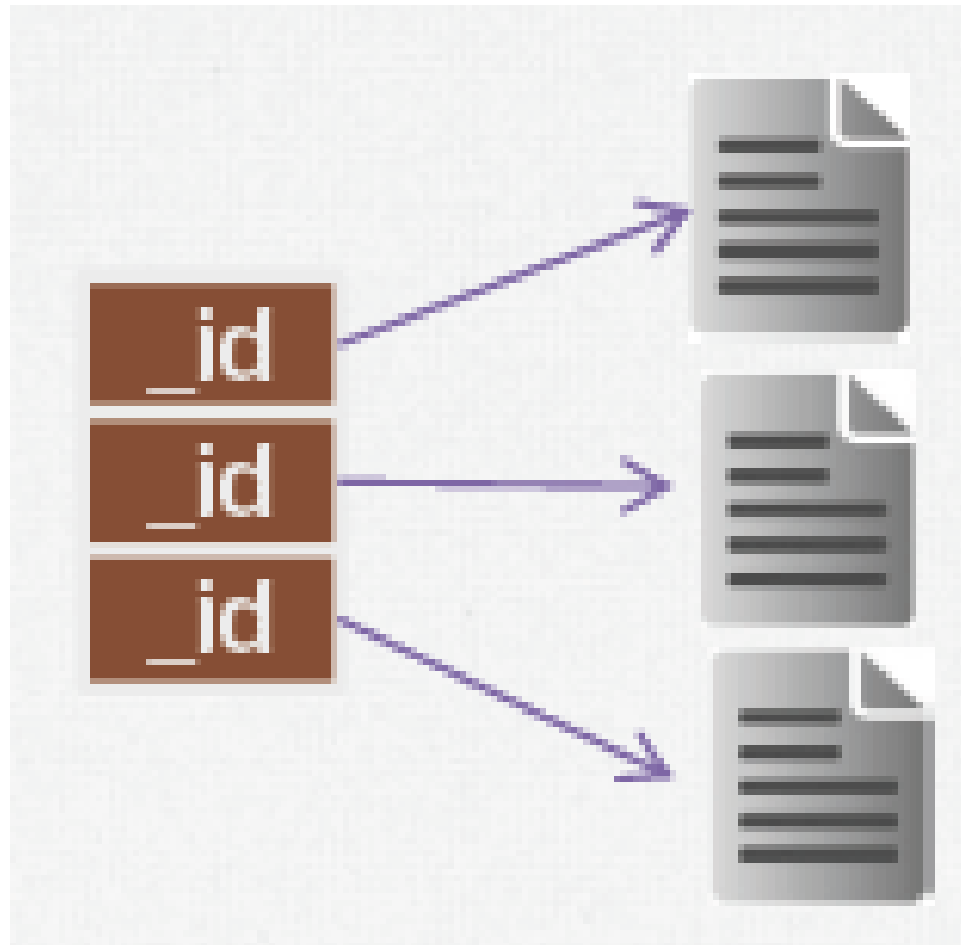


- Daten werden in Columns abgelegt
 - Eine Zeile besteht aber nicht zwangsläufig aus allen Columns
- Diese Columns können noch in Gruppen unterteilt werden
 - Vorteilhaft bei einer verteilten Datenhaltung
- Ausrichtung auf wirklich große Datenmengen



				1
1			1	1
		1		1
		1		1
		1		
				1
		1		

- Ablage der Daten in strukturierten Dokumenten
 - Nicht unbedingt XML!
 - Wesentlich weiter verbreitet ist JSON
 - Bzw. das binäre BSON



- Ablage der Daten in
 - Knoten
 - Beziehungen
 - Attribute
- Knoten und Beziehungen haben Attribute
- Knoten sind mit Beziehungen verknüpft und umgekehrt
- „Whiteboard-Friendly“
 - „Was Sie auf einem Whiteboard zeichnen können, können Sie in einer Graph-DB ablegen.“



2

EINFÜHRUNG IN APACHE HADOOP

2.1

FEATURES

Was ist Apache Hadoop?

- Hadoop ist ein verteiltes System zur Datenhaltung und Analyse
- Konzeption als beliebig skalierender Cluster
 - auf Basis normaler Standard-Hardware-Komponenten

2.2

ARCHITEKTUR

- HDFS
 - Hadoop Filesystem
 - Speichert praktisch unbegrenzte (Petabyte und größer) Datenmengen
 - Die Größe einer Festplatte beschränkt die Dateigröße nicht
- Map Reduce
 - Ein Algorithmus zur Analyse der Daten
 - Konzeptuell mit einem typischen Batchlauf vergleichbar
- Scheduling
 - Verteilung und Orchestrierung von Jobs auf die beteiligten Rechner
 - YARN übernimmt diese Aufgabe seit Hadoop 2
- Hadoop I/O
 - Datenzugriff auf das Hadoop-Dateisystem
 - Netzwerkkommunikation zur
 - Verteilung
 - Skalierung
 - Replikation zur Ausfallsicherheit

- **Datengröße**
 - Relational: Terabyte-Bereich
 - Hadoop: Pentabyte und größer
- **Transaktionen**
 - Relational: ACID
 - Hadoop: Keine
- **Datenmanipulation**
 - Relational: Beliebig viele Datenänderungen
 - Hadoop: Einmal schreiben, beliebig oft lesen
- **Skalierung:**
 - Relational: Nicht-linear mit Sättigung, häufig nur vertikale Skalierung
 - Hadoop: Linear
- **Datenstruktur**
 - Relational: Schema on Write, also konsistentes Schreiben von Daten
 - Hadoop: Schema on Read

- Auf Grund der ursprünglichen Konzeption war dies nicht der Fall
 - Eine Verwendung von Hadoop mit relationalen Modellen nutzt die Vorteile kaum und ist aus Performance-Gründen ungünstig
 - Insbesondere die interne Verwaltung der Daten verursacht immensen Netzwerk-Verkehr
- Aber
 - Neue Hardware und Optimierungen lassen diese Probleme immer mehr in den Hintergrund treten
 - Heute gibt es durchaus auch effiziente und schnelle Hadoop-basierte Lösungen, die ein relationales Modell umsetzen

2.3

INSTALLATION

- Hadoop ist Java-basiert
 - Damit läuft Hadoop auf jeder Maschine, die eine Java-Installation aufweist
 - Nach dem Entpacken des Distributions-Archivs steht Hadoop zur Verfügung
 - Zumindest prinzipiell, die Konfiguration sowie die Installation weiterer Komponenten ist nicht trivial
- Umgebungen
 - Standalone
 - Ein einzelner Hadoop-Prozess für Training, Development und Test
 - Pseudodistributed
 - Hier laufen alle Cluster-Daemons auf einer einzelnen Maschine
 - und simulieren somit einen Hadoop-Cluster
 - Fully-Distributed Mode
 - Die einzelnen Hadoop-Prozesse sind auf verschiedene Maschinen verteilt

- Cloudera bietet für Hadoop kommerzielle Unterstützung an
- Weiterhin werden auf Grund der durchaus komplexen Installation fertige Umgebungen angeboten
 - Die "Quickstarts"
 - Virtual Machines
 - Docker-Container

```
@quickstart:/
cloudera
rainer@rainer-Aspire-VN7-572G:~$ docker run --name cloudera --hostname=quickstar
t.cloudera --privileged=true -t -i -p 8888:8888 -p2080:80 cloudera/quickstart /
usr/bin/docker-quickstart
Starting mysqld: [ OK ]

if [ "$1" == "start" ] ; then
    if [ "${EC2}" == 'true' ]; then
        FIRST_BOOT_FLAG=/var/lib/cloudera-quickstart/.ec2-key-installed
        if [ ! -f "${FIRST_BOOT_FLAG}" ]; then
            METADATA_API=http://169.254.169.254/latest/meta-data
            KEY_URL=${METADATA_API}/public-keys/0/openssh-key
            SSH_DIR=/home/cloudera/.ssh
            mkdir -p ${SSH_DIR}
            chown cloudera:cloudera ${SSH_DIR}
            curl ${KEY_URL} >> ${SSH_DIR}/authorized_keys
            touch ${FIRST_BOOT_FLAG}
        fi
    fi
    if [ "${DOCKER}" != 'true' ]; then
        if [ -f /sys/kernel/mm/redhat_transparent_hugepage/defrag ]; then
            echo never > /sys/kernel/mm/redhat_transparent_hugepage/defrag
        fi
    fi
fi
```

Hadoop-Konsole der Cloudera Umgebung

←

→

×

🏠

localhost:8888/home#

⋮

🔒

☆

↓

🔍

📄

⚙️

?

🚫

☰

HUE

🏠

Query Editors ▾

Data Browsers ▾

Workflows ▾

Suche

Security ▾

📄

🔍

⚙️

?

🚫

☰

🏠 Meine Dokumente

AKTIONEN

➕ Neues Dokument

🗑️ trash 0

MEINE PROJEKTE ➕

➕ Sie sind derzeit kein Eigentümer von Projekten. Klicken Sie hier, wenn Sie nun eines hinzufügen möchten.

AN MICH FREIGEGEREN

👤 hue

🔍 example

8

🔍

Search for name, description, etc...

Name	Beschreibung	Zuletzt geändert	Projekt	Wird freigegeben
<div>🔍</div> Sample: Top salary	Top salary 2007 above \$100k	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Salary growth	Salary growth (sorted) from 2007-08	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Job loss	Job loss among the top earners 2007-08	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Customers	Email Survey Opt-Ins, Customers for Shipping ZIP Code, Total Amount per Order	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Top salary	Top salary 2007 above \$100k	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Salary growth	Salary growth (sorted) from 2007-08	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Job loss	Job loss among the top earners 2007-08	02/25/18 08:13:18	example	🔗
<div>🔍</div> Sample: Customers	Email Survey Opt-Ins, Customers for Shipping ZIP Code, Total Amount per Order	02/25/18 08:13:18	example	🔗

© Javacream

Apache Hadoop

45

Usage: `hadoop [--config confdir] COMMAND`

where `COMMAND` is one of:

<code>fs</code>	run a generic filesystem user client
<code>version</code>	print the version
<code>jar <jar></code>	run a jar file
<code>checknative [-a -h]</code>	check native hadoop and compression libraries availability
<code>distcp <srcurl> <desturl></code>	copy file or directories recursively
<code>archive -archiveName NAME -p <parent path> <src>* <dest></code>	create a hadoop archive
<code>classpath</code>	prints the class path needed to get the
<code>credential</code>	interact with credential providers
	Hadoop jar and the required libraries
<code>daemonlog</code>	get/set the log level for each daemon
<code>trace</code>	view and modify Hadoop tracing settings
or	
<code>CLASSNAME</code>	run the class named CLASSNAME

- Eine lokale Hadoop-Installation ist möglich
- Dann werden nur die Kommandozeilen-Tools benutzt
- Die Konfiguration zur Kommunikation mit dem Hadoop-System ist nicht ganz trivial
 - Ein laufender Hadoop-Cluster stellt viele Server-Prozesse zur Verfügung, die in der Konfiguration korrekt eingetragen werden müssen
 - Zookeeper
 - Nameserver
 - Regionserver
 - ...
 - In der Praxis wird die Netzwerk-Konfiguration des Hadoop-Clusters auf den lokalen Client übertragen
 - `core-site.xml`
 - `hdfs-site.xml`
 - `mapreduce-site.xml`

- Viele Hadoop-Distributionen installieren auf dem Cluster einen Web Server, dessen Anwendung den Zugriff auf den Cluster ermöglicht
 - HUE-Konsole von Cloudera
- Weiterhin werden häufig auch http-basierte Protokolle unterstützt
 - REST-API
 - Thrift
 - Kommunikationsprotokoll auf Basis von REST
- Remote Shell (SSH)
- Treiber
 - So kann das Dateisystem über Treiber auch in einem klassischen Date Explorer betrachtet werden
 - allerdings nicht sonderlich effizient...

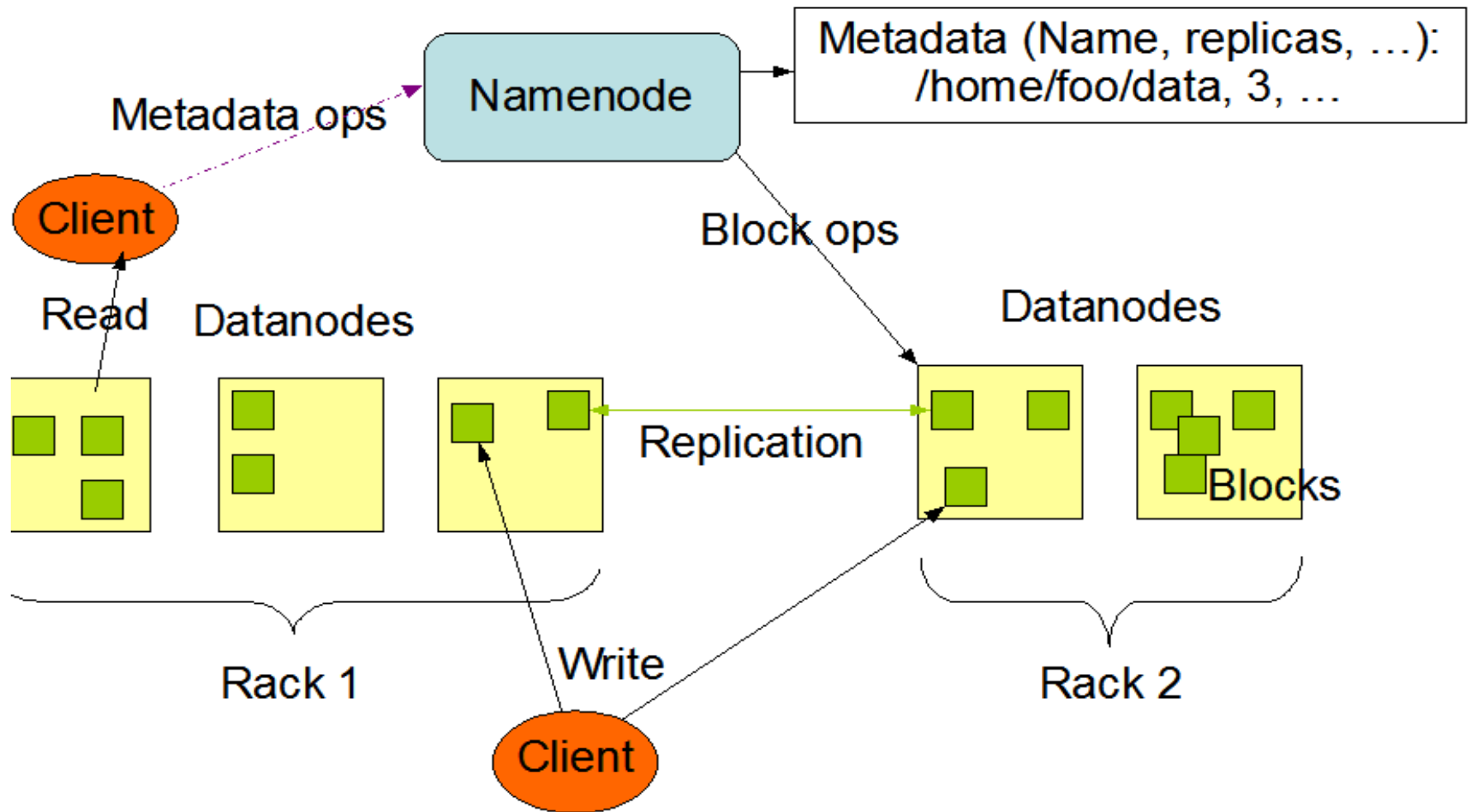
3

DAS HADOOP FILESYSTEM

3.1

DESIGN

- Arbeitet auf Standard-Hardware
- Benutzung Very Large Files
 - HDFS ist nicht optimiert auf
 - Umgang mit vielen kleinen Dateien
 - Schneller Dateizugriff
 - Parallele Schreibvorgänge
 - Updates
 - HDFS ist optimiert auf Batch-Zugriffe
- Streaming API
 - "Ingress" für eingehende Daten
 - "Egress" für ausgehende Daten



3.2

ZUGRIFF UND BEFEHLE

- Das Hadoop File System wird über den `hadoop fs` -Befehl angesteuert
- Eine Liste der Kommandos ist im Folgenden gegeben
 - Referenz unter <https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html>
- Dabei werden die relevanten Dateioperationen unterstützt
 - Zusätzlich werden "Snapshots" unterstützt, die für Backup-Zwecke eine Kopie eines Verzeichnis-Trees erzeugen
- Zum Kopieren von/aus HDFS dienen die Befehle `moveFromLocal` und `moveToLocal`

```
Usage: hadoop fs [generic options]
    [-appendToFile <localsrc> ... <dst>]
    [-cat [-ignoreCrc] <src> ...]
    [-checksum <src> ...]
    [-chgrp [-R] GROUP PATH...]
    [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
    [-chown [-R] [OWNER][:[GROUP]] PATH...]
    [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
    [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-count [-q] [-h] [-v] <path> ...]
    [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
    [-createSnapshot <snapshotDir> [<snapshotName>]]
    [-deleteSnapshot <snapshotDir> <snapshotName>]
    [-df [-h] [<path> ...]]
    [-du [-s] [-h] <path> ...]
    [-expunge]
    [-find <path> ... <expression> ...]
    [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-getfacl [-R] <path>]
    [-getfattr [-R] {-n name | -d} [-e en] <path>]
```

```
[-getmerge [-nl] <src> <localdst>]
[-help [cmd ...]]
[-ls [-d] [-h] [-R] [<path> ...]]
[-mkdir [-p] <path> ...]
[-moveFromLocal <localsrc> ... <dst>]
[-moveToLocal <src> <localdst>]
[-mv <src> ... <dst>]
[-put [-f] [-p] [-l] <localsrc> ... <dst>]
[-renameSnapshot <snapshotDir> <oldName> <newName>]
[-rm [-f] [-r|-R] [-skipTrash] <src> ...]
[-rmdir [--ignore-fail-on-non-empty] <dir> ...]
[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <path>]]
[-setfattr {-n name [-v value] | -x name} <path>]
[-setrep [-R] [-w] <rep> <path> ...]
[-stat [format] <path> ...]
[-tail [-f] <file>]
[-test -[defsz] <path>]
[-text [-ignoreCrc] <src> ...]
[-touchz <path> ...]
[-usage [cmd ...]]
```


- Die Blockgröße, also die minimale Größe einer gespeicherten Einheit ist 128 MByte
 - Diese Größe ergibt sich aus dem Ausbalancieren der Festplatten-Zugriffsgeschwindigkeit: Langsame Seeks werden vermieden
- HDFS speichert die Daten natürlich in einem realen Dateisystem
 - Lokales Dateisystem
 - ftp und ähnliches
 - Cloud-Lösungen wie Amazon S3 oder Windows Azure
- Um Ausfallsicherheit zu erreichen werden die Dateien automatisch repliziert
 - Setzen des Replikationsfaktors mit `setrepl`
- Der `hadoop`-Befehl operiert entweder auf `hdfs` oder einem "realen" System
 - Unterscheidung über die URL
 - `hdfs://`
 - `file://`

3.3

JAVA API

[org.apache.hadoop.examples.pi](#)
[org.apache.hadoop.examples.pi.mat](#)

[org.apache.hadoop.fs](#)
Interfaces
[CanSetDropBehind](#)
[CanSetReadahead](#)
[FsConstants](#)
[PathFilter](#)
[PositionedReadable](#)
[Seekable](#)
[Syncable](#)
[VolumeId](#)
Classes
[AbstractFileSystem](#)
[AvroFSInput](#)
[BlockLocation](#)
[BlockStorageLocation](#)
[ChecksumFileSystem](#)
[CommonConfigurationKeysPublic](#)
[ContentSummary](#)
[FileChecksum](#)
[FileContext](#)
[FileStatus](#)
[FileSystem](#)
[FileUtil](#)
[FilterFileSystem](#)
[FSDataInputStream](#)
[FSDataOutputStream](#)
[FSServerDefaults](#)
[FsStatus](#)
[GlobFilter](#)
[HdfsVolumeId](#)
[LocalFileSystem](#)
[LocatedFileStatus](#)
[Options](#)
[Path](#)
[RawLocalFileSystem](#)
[Trash](#)
[TrashPolicy](#)
Enums

[←](#)
[→](#)
[↺](#)
[🏠](#)

[🔒](#)
[https://hadoop.apache.org/docs/r2.5.2/api/index.html](#)

[⋮](#)
[🔍](#)
[🌟](#)

[Overview](#)
[Package](#)
[Class](#)
[Use](#)
[Tree](#)
[Deprecated](#)
[Index](#)
[Help](#)

[PREV PACKAGE](#)
[NEXT PACKAGE](#)

[FRAMES](#)
[NO FRAMES](#)

Package org.apache.hadoop.fs

An abstract file system API.

See: [Description](#)

Interface Summary	
CanSetDropBehind	
CanSetReadahead	
FsConstants	FileSystem related constants.
PathFilter	
PositionedReadable	Stream that permits positional reading.
Seekable	Stream that permits seeking.
Syncable	This interface for flush/sync operation.
VolumeId	Opaque interface that identifies a disk location.

Class Summary	
AbstractFileSystem	This class provides an interface for implementors of a Hadoop file system (analogous to the VFS of Unix).
AvroFSInput	Adapts an FSDataInputStream to Avro's SeekableInput interface.
BlockLocation	Represents the network location of a block, information about the hosts that contain block replicas, and other block metadata (E.g.
BlockStorageLocation	Wrapper for BlockLocation that also adds VolumeId volume location information for each replica.
ChecksumFileSystem	Abstract Checksummed FileSystem.
CommonConfigurationKeysPublic	This class contains constants for configuration keys used in the common code.
ContentSummary	Store the summary of a content (a directory or a file).
FileChecksum	An abstract class representing file checksums for files

Beispiel: Lesen einer Datei aus HDFS


```
String uri = (args.length == 0) ?  
"hdfs://localhost/user/cloudera/demo.txt" : args[0];  
Configuration conf = new Configuration();  
FileSystem fs = FileSystem.get(URI.create(uri), conf);  
InputStream in = null;  
try {  
    in = fs.open(new Path(uri));  
    IOUtils.copyBytes(in, System.out, 4096, false);  
} finally {  
    IOUtils.closeStream(in);  
}
```


- Das Programm benötigt ausschließlich Hadoop-Client-Bibliotheken
 - Diese werden in der Praxis häufig über ein Dependency Management Tool zur Verfügung gestellt
 - Maven
 - Gradle
 - ...
- Die Ausführung erfolgt dann auf einer beliebigen Maschine
 - Der Hadoop-Cluster muss über alle Ports erreichbar sein
- Die Remote-Konfiguration erfolgt über
 - Konventionen
 - Programmatisch
 - Durch eine Hadoop-Konfiguration



- Hier genügt es, ein normales Java-Archiv zu erzeugen
- Das `hadoop`-Kommando wird dann auf dem Server benutzt, um das Programm auszuführen
 - Die Bibliotheken stehen hier selbstverständlich zur Verfügung
 - Die Konfiguration des Clusters ebenso



- Endpoint `webhdfs://<HOST>:<HTTP_PORT>/<PATH>`
- Darauf werden die Datei-Operationen ausgeführt
 - REST-API
- Dokumentation
 - <https://hadoop.apache.org/docs/r2.7.4/hadoop-project-dist/hadoop-hdfs/WebHDFS.html>
- Beispiel-Request
 - `curl -i`
<http://localhost:50070/webhdfs/v1/user/cloudera/?op=LISTSTATUS>
 - Response:



```
{ "FileStatuses": { "FileStatus": [
  { "accessTime": 1519883870518, "blockSize": 134217728, "childrenNum": 0, "fileId": 18221, "group": "cloudera", "length": 446, "modificationTime": 1519883871322, "owner": "cloudera", "pathSuffix": "demo.txt", "permission": "644", "replication": 1, "storagePolicy": 0, "type": "FILE" }
] } }
```











 Query Editors ▾ Data Browsers ▾ Workflows ▾ Suche Security ▾

 Datei-Browser





 Aktionen ▾  In Papierkorb verschieben ▾

 Startseite / [user](#) / **cloudera** 

 Verlauf  Trash

<input type="checkbox"/>	 Name	 Size	 User	 Gruppe	 Berechtigungen	 Date
<input type="checkbox"/>	 		hdfs	supergroup	drwxr-xr-x	February 28, 2018 12:21 PM
<input type="checkbox"/>	 .		cloudera	cloudera	drwxr-xr-x	February 28, 2018 09:57 PM
<input type="checkbox"/>	 demo.txt	446 Bytes	cloudera	cloudera	-rw-r--r--	February 28, 2018 09:57 PM

Show von 1 Elemente

Seite of 1    

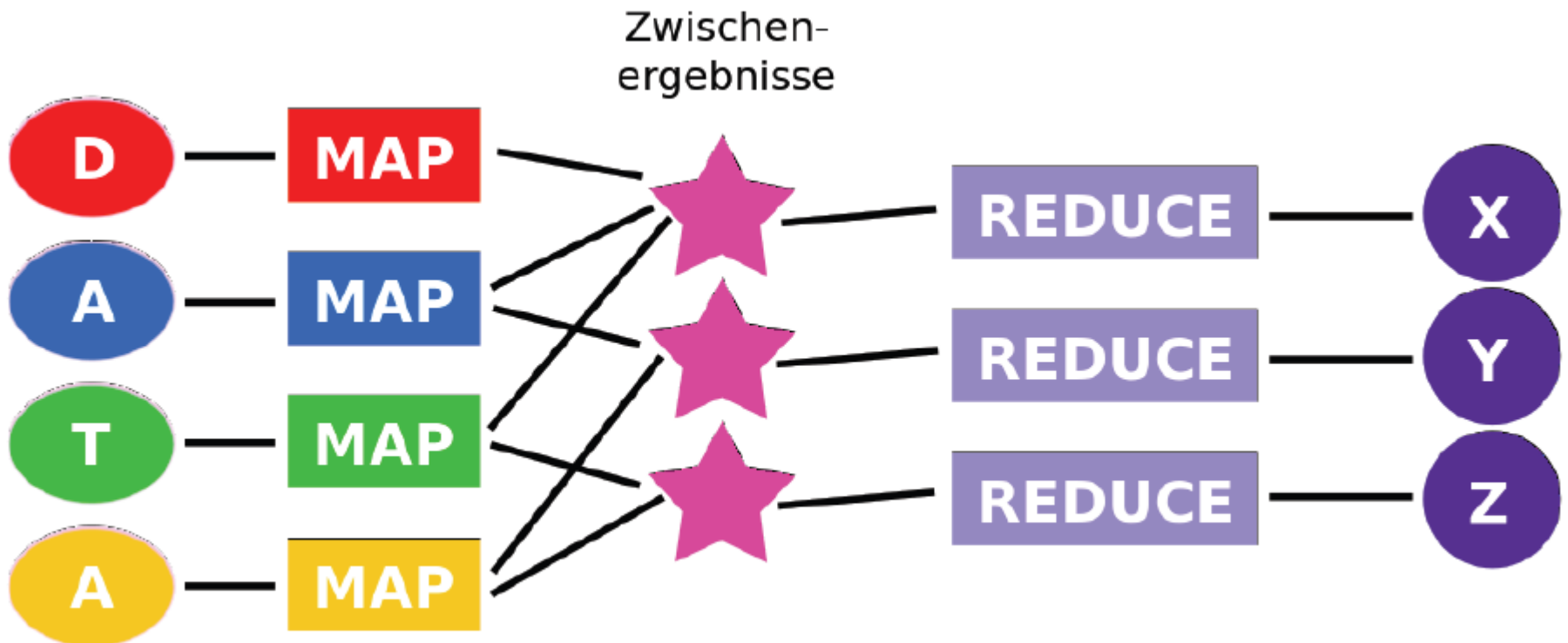
4

MAP REDUCE

4.1

GRUNDLEGENDE ARBEITSWEISE

- Bahnbrechende Entwicklung
 - Idee wird Google zugeschrieben
- Grundsätzliche Arbeitsweise
 - Operiert auf Mengen (Maps)
 - In einem ersten Schritt werden die Daten der Map in einer neuen Map aufbereitet
 - Diese wird in einem oder mehreren Reduce-Schritten zur Ergebnis-Struktur zusammengefasst



Quelle: <http://de.wikipedia.org>

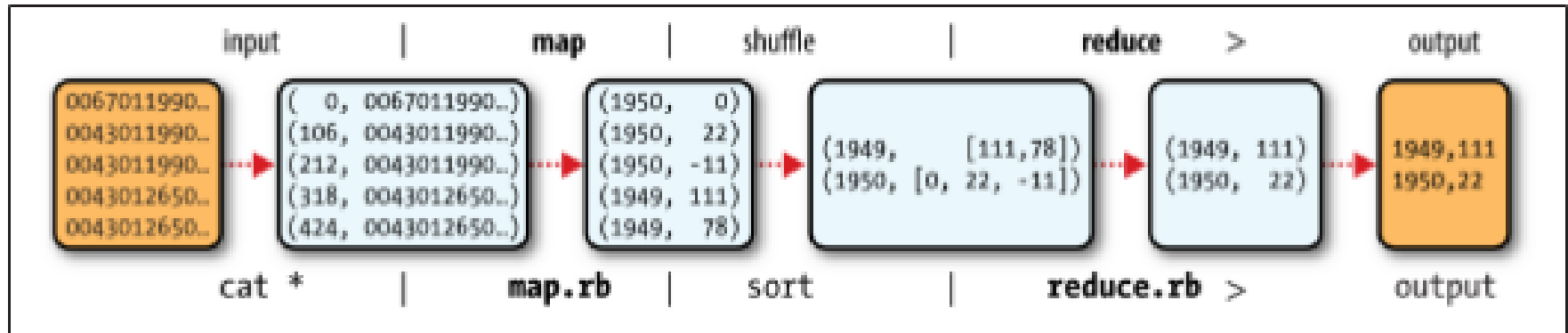
- Map bildet ein Paar, bestehend aus einem Schlüssel k und einem Wert v , auf eine Liste von neuen Paaren (l_r, x_r) ab, welche die Rolle von **Zwischenergebnissen** spielen. Die Werte x_r sind vom gleichen Typ wie die Endergebnisse w_i .
- Bei einem neuen Paar (l, x) verweist der von Map vergebene Schlüssel l dabei auf eine Liste T_l von Zwischenergebnissen, in welcher der von Map berechnete Wert x gesammelt wird.
- Die Bibliothek ruft für jedes Paar in der Eingabeliste die Funktion Map auf.
- All diese Map-Berechnungen sind voneinander unabhängig, so dass man sie nebenläufig und verteilt auf einem Computercluster ausführen kann.

- Bevor die Reduce-Phase starten kann, müssen die Ergebnisse der Map-Phase nach ihrem neuen Schlüssel l in Listen T_l gruppiert werden.
- Wenn Map- und Reduce-Funktionen nebenläufig und verteilt ausgeführt werden, wird hierfür ein koordinierter Datenaustausch notwendig.
- Die Performanz eines Map-Reduce-Systems hängt maßgeblich davon ab, wie effizient die Shuffle-Phase implementiert ist.
- Der Nutzer wird in der Regel nur über die Gestaltung der Schlüssel l auf die Shuffle-Phase Einfluss nehmen. Daher reicht es, sie einmalig gut zu optimieren, und zahlreiche Anwendungen können hiervon profitieren.

Reduce Phase (aus <https://de.wikipedia.org/wiki/MapReduce>)

- Sind alle Map-Aufrufe erfolgt bzw. liegen alle Zwischenergebnisse in T_l vor, so ruft die Bibliothek für jede Zwischenwertliste die Funktion Reduce auf, welche daraus eine Liste von Ergebniswerten w_j berechnet, die von der Bibliothek in der Ausgabeliste als Paare (l, w_j) gesammelt werden.
- Auch die Aufrufe von Reduce können unabhängig auf verschiedene Prozesse im Computercluster verteilt werden.

Optional kann vor der Shuffle-Phase noch eine Combine-Phase erfolgen. Diese hat in der Regel die gleiche Funktionalität wie die Reducefunktion, wird aber auf dem gleichen Knoten wie die Map-Phase ausgeführt. Dabei geht es darum, die Datenmenge, die in der Shuffle-Phase verarbeitet werden muss, und damit die Netzwerklast zu reduzieren.^[5] Der Sinn der Combine-Phase erschließt sich sofort bei der Betrachtung des **Wordcount-Beispiels**: Auf Grund der unterschiedlichen Häufigkeit von Wörtern in natürlicher Sprache, würde bei einem deutschen Text beispielsweise sehr oft eine Ausgabe der Form ("und", 1) erzeugt (gleiches gilt für Artikel und Hilfsverben). Durch die Combine-Phase wird nun aus 100 Nachrichten der Form ("und", 1) lediglich eine Nachricht der Form ("und", 100). Dies kann die Netzwerkbelastung signifikant reduzieren, ist aber nicht in allen Anwendungsfällen möglich.



4.2

UMSETZUNG IN HADOOP

- Hadoop benutzt Java als grundlegende Programmiersprache
 - Genauer: Zur Ausführung eines Map-Reduce-Algorithmusses wird Java-Bytecode benötigt
 - Auch Scala oder andere Sprachen können Bytecode produzieren

- Ein Java-basierter Job besteht
 - Einem Parser, welcher die Rohdaten analysiert und in eine Datenstruktur aufbereitet
 - Einem Mapper, der die aufbereitete Datenstruktur verarbeitet
 - Dieser erbt von `Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>`
 - Einem Reducer, der das Endergebnis erzeugt
 - Dieser erbt von `Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT>`
 - Einer Job-Definition, die der Hadoop-Umgebung die notwendigen Informationen zur Ausführung definiert

```
public class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
    private static final int MISSING = 9999;
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        String year = line.substring(15, 19);
        int airTemperature;
        if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
            airTemperature = Integer.parseInt(line.substring(88, 92));
        } else {
            airTemperature = Integer.parseInt(line.substring(87, 92));
        }
        String quality = line.substring(92, 93);
        if (airTemperature != MISSING && quality.matches("[01459]")) {
            context.write(new Text(year), new IntWritable(airTemperature));
        }
    }
}
```

```
public class MaxTemperatureReducer
    extends Reducer<Text, IntWritable, Text,
IntWritable> {
public void reduce(Text key, Iterable<IntWritable>
values, Context context)
    throws IOException, InterruptedException {
int maxValue = Integer.MIN_VALUE;
    for (IntWritable value : values) {
        maxValue = Math.max(maxValue, value.get());
    }
    context.write(key, new IntWritable(maxValue));
}
}
```

```
public static void main(String[] args) throws Exception {
    if (args.length != 2) {
        System.err.println("Usage: MaxTemperature <input path>
<output path>");
        System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(MaxTemperature.class);
    job.setJobName("Max temperature");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(MaxTemperatureMapper.class);
    job.setReducerClass(MaxTemperatureReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

- <https://hadoop.apache.org/docs/r2.7.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

5

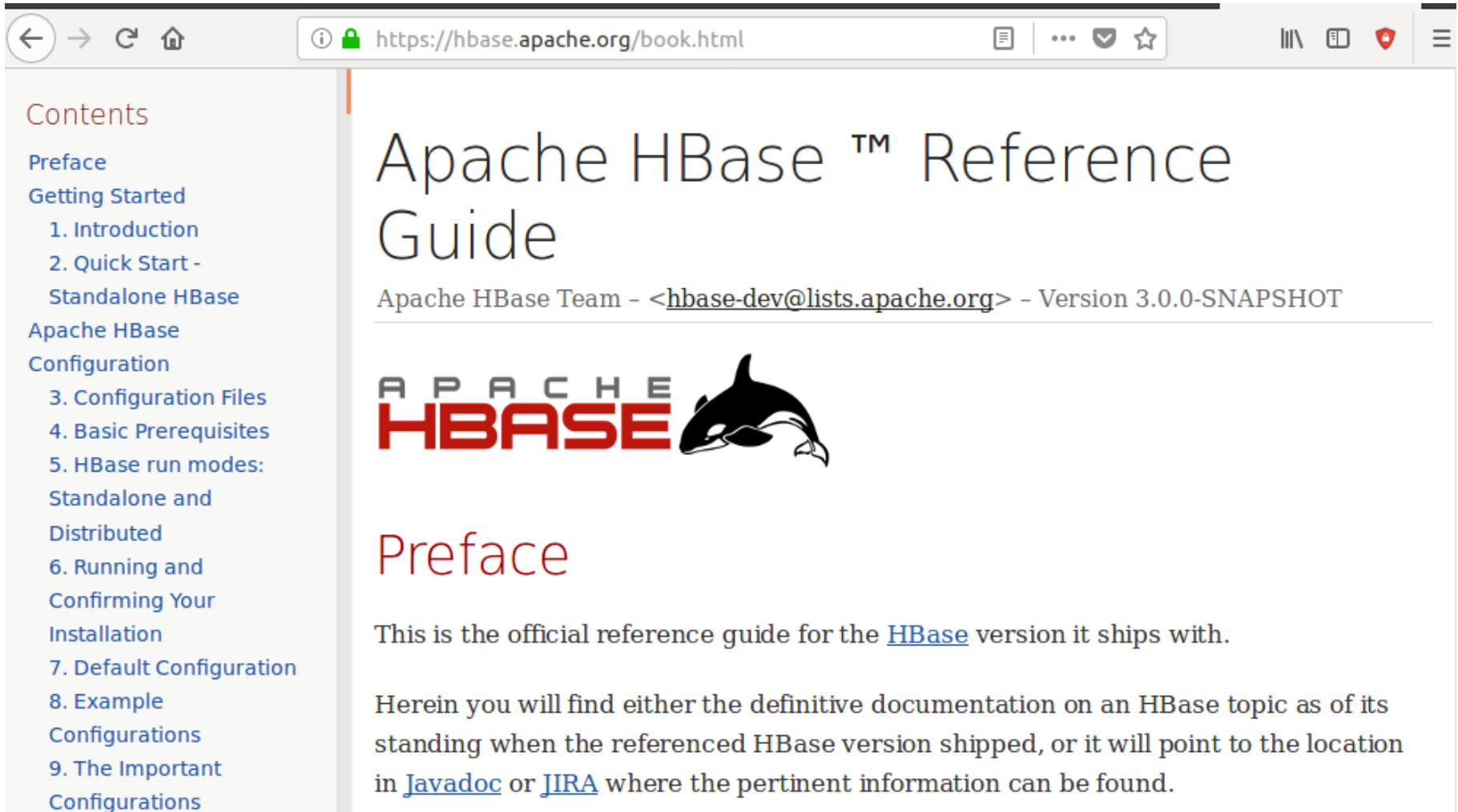
WERKZEUGE UND SERVER DES HADOOP-ÖKOSYSTEMS

5.1

HBASE

- HBase ist eine Spalten-orientierte Datenbank
- Wichtig:
 - Obwohl HBase Tabellen kennt ist jede Analogie mit einer relationalen Datenbank falsch
 - HBase ist eine Map
 - Eine Map kann beliebige Key-Value-Paare aufnehmen
- HBase ist in Java geschrieben

- Ausgerichtet auf BIG Data
- Automatische Sortierung der Schlüssel einer Zelle
- Automatische Versionierung
- Datenablage auf HDFS
- Auswertung durch
 - SQL-nahe Abfragesprache
 - Apache Phoenix
 - Map Reduce




The screenshot shows a web browser displaying the Apache HBase Reference Guide. The browser's address bar shows the URL <https://hbase.apache.org/book.html>. On the left side, there is a navigation menu with the following items: Contents, Preface, Getting Started (with sub-items 1. Introduction and 2. Quick Start - Standalone HBase), Apache HBase Configuration (with sub-items 3. Configuration Files, 4. Basic Prerequisites, 5. HBase run modes: Standalone and Distributed, 6. Running and Confirming Your Installation, 7. Default Configuration, 8. Example Configurations, and 9. The Important Configurations). The main content area features the title "Apache HBase™ Reference Guide" in a large font, followed by the text "Apache HBase Team - <hbase-dev@lists.apache.org> - Version 3.0.0-SNAPSHOT". Below this is the Apache HBase logo, which consists of the word "APACHE" in a grey, spaced-out font above the word "HBASE" in a bold, red font, with a black silhouette of a whale to the right. The section "Preface" is highlighted in red. The preface text states: "This is the official reference guide for the [HBase](#) version it ships with. Herein you will find either the definitive documentation on an HBase topic as of its standing when the referenced HBase version shipped, or it will point to the location in [Javadoc](#) or [JIRA](#) where the pertinent information can be found."

Contents

- Preface
- Getting Started
 - 1. Introduction
 - 2. Quick Start - Standalone HBase
- Apache HBase Configuration
 - 3. Configuration Files
 - 4. Basic Prerequisites
 - 5. HBase run modes: Standalone and Distributed
 - 6. Running and Confirming Your Installation
 - 7. Default Configuration
 - 8. Example Configurations
 - 9. The Important Configurations

Apache HBase™ Reference Guide

Apache HBase Team - <hbase-dev@lists.apache.org> - Version 3.0.0-SNAPSHOT



Preface

This is the official reference guide for the [HBase](#) version it ships with.

Herein you will find either the definitive documentation on an HBase topic as of its standing when the referenced HBase version shipped, or it will point to the location in [Javadoc](#) or [JIRA](#) where the pertinent information can be found.

- **Table**
 - Eine Table besteht aus mehreren Rows
- **Row**
 - Eine Row wird durch einen Row Key identifiziert
 - Jede Row besteht aus Columns
- **Column Family**
 - Hat einen eindeutigen Namen
 - Wird bei der Erstellung der Tabelle angegeben
 - Eine Column Family gruppiert eine beliebige Anzahl von Columns
 - Ein beliebiger Column Qualifier kann zugeordnet werden
- **Column**
 - Eine Column wird einer Column Family zugeordnet
 - Name: `family:column`
- **Cell**
 - Ein Schnittpunkt von Row und Column
 - Besteht aus Inhalt und Timestamp

- Entnommen aus dem Artikel "understanding HBase",
http://jimbojw.com/#understanding_hbase

HBase: Datenablage in einer Map

```
{  
  "zzzzzz" : "woot",  
  "xyz" : "hello",  
  "aaaab" : "world",  
  "1" : "x",  
  "aaaaa" : "y"  
}
```



```
{  
  "1" : "x",  
  "aaaaa" : "y",  
  "aaaab" : "world",  
  "xyz" : "hello",  
  "zzzzz" : "woot"  
}
```

HBase: Datenablage mit Column-Families

```
{
  "1" : {
    "ColumnFamilyA" : "x",
    "ColumnFamilyB" : "z"
  },
  "aaaaa" : {
    "ColumnFamilyA" : "y",
    "ColumnFamilyB" : "w"
  },
  "aaaab" : {
    "ColumnFamilyA" : "world",
    "ColumnFamilyB" : "ocean"
  },
  "xyz" : {
    "ColumnFamilyA" : "hello",
    "ColumnFamilyB" : "there"
  },
  "zzzzz" : {
    "ColumnFamilyA" : "woot",
    "ColumnFamilyB" : "1337"
  }
}
```

HBase: Datenablage mit Column-Families und beliebigen Columns

```
{
  "aaaaa" : {
    "A" : {
      "foo" : "y",
      "bar" : "d"
    },
    "B" : {
      "" : "w"
    }
  },
  "aaaab" : {
    "A" : {
      "foo" : "world",
      "goo" : "moon",
    },
    "B" : {
      "" : "ocean"
    }
  },
  // ...
}
```

HBase: Datenablage mit Column-Families und beliebigen Columns mit Zeitstempel

```
{
  "aaaaa" : {
    "A" : {
      "foo" : {
        15 : "y",
        4  : "m"
      },
      "bar" : {
        15 : "d",
      }
    },
    "B" : {
      "" : {
        6 : "w"
        3 : "o"
        1 : "w"
      }
    }
  },
  // ...
}
```

- **Starten einer Shell**
 - `hbase shell`
 - Eine erweiterte Ruby-Shell
- **Anlegen einer Tabelle für ein WIKI-Projekt**
 - `create 'wiki', 'text'`
 - Nur eine einzige Column Family `text`
- **Anlegen eines Datensatzes mit dem Row Key `home`**
 - `put 'wiki', 'Home', 'text:', 'Welcome to the wiki!'`
 - Check: Auslesen mit `get 'wiki', 'Home', 'text:'`
- **Löschen des Datensatzes**
 - `deleteall 'wiki', 'Home'`
 - Hinweis: `delete` löscht einzelne Zellen

- **Tabelle für weitere Zugriffe sperren**
 - `disable 'wiki'`
- **Wie viele Versionen sollen pro Zelle gespeichert werden?**
 - `alter 'wiki', { NAME => 'text', VERSIONS => org.apache.hadoop.hbase.HConstants::ALL_VERSIONS }`
- **Hinzufügen einer Column Family**
 - `alter 'wiki', { NAME => 'revision', VERSIONS => org.apache.hadoop.hbase.HConstants::ALL_VERSIONS }`
- **Ausgabe der Tabellen-Beschreibung**
 - `describe 'wiki'`
- **Tabelle aktivieren**
 - `enable 'wiki'`

5.2

HIVE

- Hive ist eine Datenbanklösung, die mit der Hive Query Language eine SQL-nahe Abfragesprache definiert
 - Hive operiert auf Dateien im HDFS
- Hive Shell
 - Starten durch `hive`
 - Anschließend können über die Konsole die üblichen Befehle zur Tabellendefinition oder Daten-Manipulation abgesetzt werden
 - Alternativ: Ausführung eines Scripts mit `hive -f`

- **Anlegen durch:**

```
CREATE TABLE records (year STRING, temperature INT, quality  
INT)
```

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
```

- ROW FORMAT DELIMITED FIELDS

- Jede Zeile ist durch Tabs unterteile

- TERMINATED BY '\t'

- Jede Zeile ist durch einen Umbruch getrennt


- Damit hat eine Hive-Tabelle den Aufbau einer simplen Textdatei







- die natürlich in HDFS abgelegt wird

```
LOAD DATA LOCAL INPATH 'input/ncdc/micro-  
tab/sample.txt' OVERWRITE INTO TABLE records;
```



- Dies ist ein simpler Filetransfer!
 - Keinerlei Prüfung der Datei auf Struktur
 - "Schema-less writes"



Die geladene Datei im HDFS



 Query Editors ▾ Data Browsers ▾ Workflows ▾ Suche Security ▾










   ▾   

Datei-Browser



 Aktionen ▾  In Papierkorb verschieben ▾

 Startseite / [user](#) / [hive](#) / [warehouse](#) / **records** 



 Verlauf  Trash

<input type="checkbox"/>	 Name	 Size	 User	 Gruppe	 Berechtigungen	 Date
<input type="checkbox"/>	 ↑		hive	supergroup	drwxrwxrwx	February 28, 2018 07:14 AM
<input type="checkbox"/>	 .		root	supergroup	drwxrwxrwx	February 28, 2018 07:17 AM
<input type="checkbox"/>	 sample.txt	51 Bytes	root	supergroup	-rwxrwxrwx	February 28, 2018 07:17 AM



Metastore Manager






SQL

 default

Tabellen

(1) 

records

	Name	Type	Kommentar
1	 year	string	Add a comment...
2	 temperature	int	Add a comment...
3	 quality	int	Add a comment...

[View more...](#)

BEISPIEL

	records.year	records.temperature	records.quality
1	1950	0	1
2	1950	22	1
3	1950	-11	1

[View more...](#)

localhost gelesen

- `select year from records`
- Wie zu erwarten werden die Jahreszahlen aus der Datei gelesen und ausgegeben
 - Hier wird das Schema der Tabelle aber sehr wohl geprüft
 - "Schema on Read"

```
SELECT year, MAX(temperature) FROM records  
WHERE temperature != 9999 AND quality IN (0, 1, 4, 5,  
9)  
GROUP BY year;
```

- Wichtig: Diese Abfrage wird von Hive automatisch in einen Map Reduce-Job übersetzt und ausgeführt
 - Ein Group By ist eigentlich immer eine Form von Map Reduce

Function	MySQL	HiveQL
Retrieving information	<code>SELECT from_columns FROM table WHERE conditions;</code>	<code>SELECT from_columns FROM table WHERE conditions;</code>
All values	<code>SELECT * FROM table;</code>	<code>SELECT * FROM table;</code>
Some values	<code>SELECT * FROM table WHERE rec_name = "value";</code>	<code>SELECT * FROM table WHERE rec_name = "value";</code>
Multiple criteria	<code>SELECT * FROM table WHERE rec1="value1" AND rec2="value2";</code>	<code>SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";</code>
Selecting specific columns	<code>SELECT column_name FROM table;</code>	<code>SELECT column_name FROM table;</code>
Retrieving unique output records	<code>SELECT DISTINCT column_name FROM table;</code>	<code>SELECT DISTINCT column_name FROM table;</code>
Sorting	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2;</code>
Sorting backward	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>	<code>SELECT col1, col2 FROM table ORDER BY col2 DESC;</code>
Counting rows	<code>SELECT COUNT(*) FROM table;</code>	<code>SELECT COUNT(*) FROM table;</code>
Grouping with counting	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>	<code>SELECT owner, COUNT(*) FROM table GROUP BY owner;</code>
Maximum value	<code>SELECT MAX(col_name) AS label FROM table;</code>	<code>SELECT MAX(col_name) AS label FROM table;</code>
Selecting from multiple tables (Join same table using alias w/"AS")	<code>SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;</code>	<code>SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);</code>

Function	MySQL	HiveQL
Selecting a database	USE database;	USE database;
Listing databases	SHOW DATABASES;	SHOW DATABASES;
Listing tables in a database	SHOW TABLES;	SHOW TABLES;
Describing the format of a table	DESCRIBE table;	DESCRIBE (FORMATTED EXTENDED) table;
Creating a database	CREATE DATABASE db_name;	CREATE DATABASE db_name;
Dropping a database	DROP DATABASE db_name;	DROP DATABASE db_name (CASCADE);

Function	Hive
Run script inside shell	<code>source file_name</code>
Run ls (dfs) commands	<code>dfs -ls /user</code>
Run ls (bash command) from shell	<code>!ls</code>
Set configuration variables	<code>set mapred.reduce.tasks=32</code>
TAB auto completion	<code>set hive.<TAB></code>
Show all variables starting with hive	<code>set</code>
Revert all variables	<code>reset</code>
Add jar to distributed cache	<code>add jar jar_path</code>
Show all jars in distributed cache	<code>list jars</code>
Delete jar from distributed cache	<code>delete jar jar_name</code>

Hive Command Line

Function	Hive
Run query	<code>hive -e 'select a.col from tab1 a'</code>
Run query silent mode	<code>hive -S -e 'select a.col from tab1 a'</code>
Set hive config variables	<code>hive -e 'select a.col from tab1 a' -hiveconf hive.root.logger=DEBUG,console</code>
Use initialization script	<code>hive -i initialize.sql</code>
Run non-interactive script	<code>hive -f script.sql</code>

- **Numeric Types**
 - TINYINT (1-byte signed integer)
 - SMALLINT (2-byte signed integer)
 - INT/INTEGER (4-byte signed integer)
 - BIGINT (8-byte signed integer)
 - FLOAT (4-byte single precision)
 - DOUBLE (8-byte double precision)
 - DOUBLE PRECISION (alias für DOUBLE)
 - DECIMAL
 - NUMERIC (wie DECIMAL)
- **Date/Time Types**
 - TIMESTAMP
 - DATE
 - INTERVAL

- String Types
 - STRING
 - VARCHAR
 - CHAR
- Verschiedene
 - BOOLEAN
 - BINARY (Note: Only available starting with Hive 0.8.0)
- Komplexe Typen
 - ARRAY<data_type>
 - MAP<primitive_type, data_type>
 - STRUCT<col_name : data_type [COMMENT col_comment], ...>
 - UNIONTYPE<data_type, data_type, ...>

- Unterscheide
 - Managed Tables
 - Werden in den Hive-Workspace kopiert
 - External
 - Hier referenziert Hive über den Metastore die externen Daten

■ Partitions

- Unterteilen die Daten
 - In der Verzeichnisstruktur entspricht dies Unterverzeichnissen
- Dazu wird eine Partition Column eingesetzt
- ```
CREATE TABLE logs (ts BIGINT, line STRING)
 PARTITIONED BY (dt STRING, country STRING);
```

## ■ Bucket

- Fügen eine weitere fein-granulare Unterteilung durch
- ```
CREATE TABLE bucketed_users (id INT, name STRING)  
    CLUSTERED BY (id) INTO 4 BUCKETS;
```

- **Sortieren und Aggregieren**

```
FROM records2
SELECT year, temperature
DISTRIBUTE BY year
SORT BY year ASC, temperature DESC;
```

- **MapReduce**

```
FROM (
FROM records2
MAP year, temperature, quality
USING 'is_good_quality.py'
AS year, temperature) map_output
REDUCE year, temperature
USING 'max_temperature_reduce.py'
AS year, temperature
```

■ Joins

```
SELECT sales.*, things.*  
FROM sales JOIN things ON (sales.id = things.id);
```

■ Subqueries

```
SELECT station, year, AVG(max_temperature)  
FROM (  
  SELECT station, year, MAX(temperature) AS max_temperature  
  FROM records2  
  WHERE temperature != 9999 AND quality IN (0, 1, 4, 5, 9)  
  GROUP BY station, year  
) mt  
GROUP BY station, year;
```


- **Multitable Insert**
 - Eine Input-Datei wird in verschiedene Tabellen unterteilt
- **CREATE TABLE ... AS SELECT**
 - Das Ergebnis einer Query wird in eine neue Tabelle geschrieben
- **Verändern der Tabellenstruktur**
 - In Hive eine triviale Operation
 - Schema onread...

- **Views**

```
CREATE VIEW valid_records AS  
SELECT * FROM records2 WHERE temperature != 9999 AND  
quality IN (0, 1, 4, 5, 9);
```

5.3

SPARK

- Eine Lösung zur Analyse großer verteilter Datenmengen
 - Gleiche Problemstellung wie Map Reduce
 - Aber Spark implementiert eigene Algorithmen
- Spark operiert auf einem In-Memory-Cache von Daten
 - Und hat damit Performance-Vorteile zum Beispiel bei
 - Iterativen Verfahren
 - Interaktive Verfahren
 - Dieser Cache sind die Resilient Distributed Datasets, kurz RDD
 - Eine über den Cluster verteilbare Collection
 - Etwas vereinfacht operiert eine Spark-Abfrage auf diesen RDDs und produziert weitere
 - Vorsicht: RDDs werden lazy erzeugt!
 - Erst eine Auswertung darauf führt echte Aktionen aus

- REPL mit der Spark-Shell
 - `spark-shell`
 - Ein Scala-Interpreter

- `val lines = sc.textFile("/input/sample.txt")`
- **Laden der Daten**
- `val records = lines.map(_.split("\t"))`
- **Transformation der Daten und erzeugen eines neuen RDD**
- `val filtered = records.filter(rec => (rec(1) != "9999" && rec(2).matches("[01459"])))`
- **Filtern der transformierten Daten**
- `val tuples = filtered.map(rec => (rec(0).toInt, rec(1).toInt))`
- **Aggregieren der Daten, der Map-Anteil**
- `val maxTemps = tuples.reduceByKey((a, b) => Math.max(a, b))`
- **Erzeugen des Ergebnisses, der Reduce-Anteil**
- **Ausgabe des Ergebnisses**
 - `maxTemps.foreach(println(_))`

```
import org.apache.spark.SparkContext._
import org.apache.spark.{SparkConf, SparkContext}
object MaxTemperature {
  def main(args: Array[String]) {
    val conf = new SparkConf().setAppName("Max Temperature")
    val sc = new SparkContext(conf)
    sc.textFile(args(0))
      .map(_._split("\t"))
      .filter(rec => (rec(1) != "9999" && rec(2).matches("[01459]")))
      .map(rec => (rec(0).toInt, rec(1).toInt))
      .reduceByKey((a, b) => Math.max(a, b))
      .saveAsTextFile(args(1))
  }
}
```

5.4

WEITERE PRODUKTE

- **Flume**
 - Eine Datensenke, die Daten nach HDFS verschiebt
 - Auch andere Ziele sind möglich
- **Parquet**
 - Ein Format zum Speichern verschachtelter Daten
- **Pig**
 - Eine Hochsprache (Pig Latin) zur einfacheren Definition von Datenanalysen mit Map Reduce
- **Crunch**
 - Vereinfachte Programmierung Java-basierter Datenanalysen mit Map Reduce
- **Sqoop**
 - Ein generischer Datenextraktor
- **Avro**
 - Ein Daten-Serialisierer

6

MACHINE LEARNING

6.1

ÜBERSICHT

- **Supervised Learning**
 - **Klassifizierung**
 - Bei der Klassifizierung von Daten werden Eingangsdaten in eine diskrete Menge von Ergebnissen umgewandelt
 - **Recommendation**
 - Empfehlungen auf Basis vorhandener Beziehungen
 - Eigentlich eine Untermenge der Klassifizierung
 - **Regression**
 - Hier werden beliebige Werte produziert
- **Unsupervised Learning**
 - **Clustering**
 - Erkennen und Gruppieren von Zusammenhängen
 - **Density Estimation**
 - Erkennen von Zusammenhängen, die Datensätze kompakt representieren

- Classification
 - k-nearest neighbours
 - Decision Trees
 - Ensemble Methods
 - (Naive) Bayes
 - Support Vector Machines
- Regression
 - (Weighted) Linear Regression
 - Ordinary Least Squares Regression
 - Logistic Regression
 - Tree based Regression
- <https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html>

- Clustering
 - Centroid-based algorithms
 - Connectivity-based algorithms
 - Density-based algorithms
 - Probabilistic
 - Dimensionality Reduction
 - Neural networks / Deep Learning
- Principal Component Analysis
- Singular Value Decomposition
- Independent Component Analysis
- <https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html/2>

- Mustererkennung
 - Bilder
 - Schriften
 - Gesichter
- Spam-Detection
- Produkt-Empfehlungen
- Social Networking

- Beispiel: Klassifizierung
 - Ein Algorithmus bestimmt aus einem Feature-Set eine Target-Variable
 - Dieser Algorithmus wird
 - Trainiert
 - Hierzu werden dem Algorithmus Features und erwartete Targets präsentiert
 - Nachvollzogen
 - Die "knowledge representation" des Algorithmus weist nach, nach welchen Kriterien die Entscheidungen nach dem Training getroffen werden
 - Getestet
 - Nun werden Test-Daten benutzt und das berechnete Target mit dem erwarteten verglichen

- Statistische Analyse
- großer Datenmengen
- führt zu Ergebnissen, die wiederum statistisch relevant "korrekt" sind
 - Der Anspruch von Machine Learning kann nicht sein, fehlerfreie Ergebnisse zu produzieren
 - Ebenso können selbst kleine Änderungen große Ergebnisschwankungen produzieren
 - Der bekannte Schmetterlingseffekt

- Data Collection
- Aufarbeitung der Daten
- Prüfen der Daten-Qualität
- Training
- Test
- Anwendung

ZUSAMMENHANG MIT HADOOP

- Der Flow eines Algorithmus' kann auf MapReduce passen
 - Graphen-orientierte Methoden sind allerdings ebenfalls gebräuchlich
- Ebenso gebräuchlich ist hier jedoch die Verwendung von Spark
 - Ein Zwischenspeichern von Maps/RDDs über verschiedene Schritte des Ablaufs ist hier sicherlich vorteilhaft

- Mit Mahout (deutsch: "Elefantentreiber") steht ein Werkzeug zur Verfügung, mit dem direkt Machine Learning auf Hadoop ausgeführt werden kann
 - <https://mahout.apache.org/>
- Das Projekt definiert sich primär als Sammlung für Implementierungen von Machine Learning Algorithmen, die auf einem verteilten System ausgeführt werden können.

6.2

VERFAHREN

6.3

BEISPIEL: KLASSIFIZIERUNG MIT K NEAREST NEIGHBORS

- Ein Classification-Algorithmus
- Prinzip:
 - Für eine Menge von Datensätzen mit gemeinsamen Attributen ist jeweils das Target bekannt
 - Ein unbekannter Datensatz wird Attribut für Attribut mit diesem Datensatz verglichen und die Gesamt-Abweichungen für jeden bekannten Datensatz berechnet
 - Diese Abweichungen werden nach Größe sortiert und mit den Target-Werten belegt
 - Die Majorität der Target-Werte der obersten k Einträge bestimmt das Target des zu bestimmenden Datensatzes

