NoSQL Datenbank Technologie: HBase

NGOUKAM MONKAM CHARLY ROSSY
UNIVERSITÄT SIEGEN
charly.nmonkam@student.uni-siegen.de

KURZFASSUNG

Die Mengen der zu verarbeitenden Daten haben über die Jahre exponentiell zugenommen. Die Unternehmen und die Organisationen sind mit dem Effizienzproblem konfrontiert. Mit dieser wachsenden Macht haben die relationalen Datenbanken ihre Grenzen erreicht. Um dieses Problem zu lösen, sind viele NoSQL Datenbanken entstanden. Der Begriff NoSQL bedeutet "Not Only Sql", auf Deutsch übersezt "nicht nur SQL". Dies ist ein neuer Ansatz der Datenbankdesigns und ist besonders nützlich für große Mengen verteilter Daten. Heutzutage gibt es eine Vielzahl von NoSQL Datenbanken (HBase, MongoDB, CouchDB ...). Wir werden uns in dieser Ausarbeitung hauptsächlich für die HBase Datenbank interessieren. Im ersten Teil werden wir die Geschichte und das Konzept der NoSQL Datenbanken präsentieren. Wir werden im zweiten Teil die HBase Technologie, seine Geschichte, sein Datenmodell und seine Architektur detaillieren. Im dritten Teil werden wir die wichtigsten HBase-Befehle nutzen, um eine konkrete Datenbank zu erstellen. Mit ein paar Bildern zeigen wir, wie wir Abfragen an den Server schicken und wie die Ergebnisse davon zurückgeliefert werden.

1. EINLEITUNG

In den letzten Jahren hat sich die Welt der Informationsverarbeitung entscheidend verändert. Die Unternehmen haben ein erhöhtes Bedürfnis mit großen Datenmengen umgehen zu können. Dies ist der Fall von Google, Yahoo oder Facebook, die etwa 500 terabytes an neuen Daten pro Tag speichern. Die Anfragen von den ungefähr 890 Millionen täglich aktiver Nutzer auf Facebook müssen gleichzeitig ohne Wartezeit bearbeitet werden. Die relevante Frage ist jetzt, wie man ein solches System entwickeln kann, das diese Anforderungen erfüllt. Die relationale Datenbanken sind für diese Datenmengen nicht geeignet . Es wird praktisch unmöglich für einen Server, solche Daten allein zu verarbeiten, weil die Datenverarbeitungszeiten zu lang sind. Um diese große Datenmengen einfach zu verarbeiten, haben die Computeringenieure geplant, NoSQL Datenbanken zu entwickeln. Die Architektur der NoSQL Datenbanken ist an die Cluster angepasst, das heißt, die NoSQL Datenbanken laufen dann gleichzeitig auf mehreren Servern, die parallel arbeiten. Somit wird die Speicherung und die Verarbeitung der Daten immer einfacher. Im Rest unseres Berichts werden wir den technischen Aspekt von HBase erläutern und wie HBase parallel auf mehreren Servern arbeitet.

2. NOSQL

Dieses theoretische Kapitel bezieht sich auf die Entstehung des Begriffes NoSQL sowie eine Beschreibung der verschiedenen Typen von NoSQL Datenbanken. Da mehrere Arten von NoSQL Datenbanken existieren, ist es wichtig, die Eigenschaften von jeder NoSQL Datenbank zu verstehen. Anhand des CAP-Theorem werden wir diese Eigenschaften einfach präsentieren.

2.1 Die Entstehung der NoSQL Datenbanken

Der Begriff NoSQL bezieht sich auf die Datenbankmanagementsystemen, die die Daten manipulieren, deren Volumen sehr groß sind. Es eignet sich für Websites, die Millionen von Nutzern pro Tag wie Google, Yahoo oder Facebook haben. Der Begriff NoSQL wurde zum ersten Mal 1998 von Carlo Strozzi verwendet. Tatsächlich hat Carlo Strozzi diesen Begriff verwendet, um sein relationales Open Source-Datenbanksystem zu benennen. Im Jahr 2009 an der Entwicklerkonferenz von nicht-relationalen Daten hat Eric Evans diesen Begriff verwendet, um nicht-relationale Datenbanken zu benennen.

2.2 Die Funktionsweise der NoSQL Datenbank

2.2.1 Skalierung

Die Skalierung ist die Fähigkeit für ein System, seine Leistung beim Hinzufügen von Hardware-Ressourcen zu halten[1]. Es gibt zwei Arten von Skalierung: Die vertikale Skalierung und die horizontale Skalierung.

Die vertikale Skalierung (Scale Up) Abbildung 1 ist das Hinzufügen von Ressourcen innerhalb einer logischen Einheit, um die Kapazität zu erhöhen. Ein Beispiel könnte das Hinzufügen von CPUs zu einem existierenden Server sein oder die Erweiterung des Speicherplatzes [2]. Ein Vorteil für die vertikale Skalierung ist ihr geringerer Stromverbrauch und sie ist außerdem einfach umzusetzen [3]. Die Nachteile hier sind sehr kritisch, weil sie schnell an ihre Grenzen stößt und der Hardwareausfall Systemstillstand bedeuten kann [3]. Aus diesen Gründen ist die vertikale Skalierung an die NoSQL Datenbanken nicht angepasst.

Die horizontale Skalierung (Scale Out) ermöglicht das parallele Hinzufügen der identischen Server, um die Leistung von Systemen zu erhöhen, die sehr große Datenmengen verarbeiten[4]. Die NoSQL Datenbanken sind an die horizontale Skalierung gut angepasst, weil sie viele Hardware-Ressourcen brauchen. Das Bild Abbildung 2 illustriert dieses Prinzip. Wenn der erste Server seine Grenze erreicht, kann man einen zweiten identischen Server hinfügen. Die Vorteile der hori-

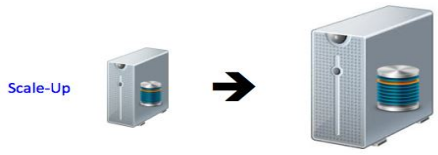

Figure 1: Die vertikale Skalierung [3]

zontalen Skalierung ist ihre hohe Verfügbarkeit durch Lastenverteilung und Replikation. Der Hardwareausfall kann auch besser kompensiert werden. Die Datenbank kann auch theoretisch unendlich skalierbar sein. Ein Nachteil wären zum Beispiel die höheren Stromkosten (Serverbetrieb, Kühlung, etc)[3]. Aus diesen Gründen ist die horizontale Skalierung an die NoSQL Datenbanken gut angepasst.

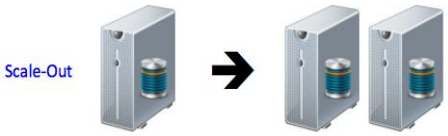

Figure 2: Die horizontale Skalierung [3]

Wir kennen jetzt die Skalierung, die für die NoSQL Datenbanken benutzt wird. In der folgenden Sektion müssen wir die Eigenschaften der NoSQL Datenbanken anhand des CAP-Theorem präsentieren.

2.2.2 CAP-Theorem

Das CAP-Theorem ist eine von Dr. Eric A. Brewer aufgestellte Behauptung, die besagt, dass von drei möglichen Eigenschaften eines Datenbanksystems — Konsistenz, Verfügbarkeit und Partitionierbarkeit — stets nur zwei zugleich erreichbar sind [5].

- *Die Konsistenz (C):* bedeutet, dass alle Server dieselben Daten zur gleichen Zeit sehen.
- *Die Verfügbarkeit (A):* bedeutet, dass die Lese- und Schreibabfragen immer ausgeführt werden.
- *Die Partitionierung (P):* bedeutet, dass der einzige Grund für die Abschaltung des Systems der komplette Netzwerkausfall ist.

Bei den NoSQL Datenbanken läuft das System weiter, wenn ein Teil des Systems ausfällt. Um eine verteilte Architektur zu erstellen muss man zwei von drei verfügbaren Eigenschaften auswählen. Wir erhalten daher die folgenden Architekturen:

- *CP:* Die Server des Datenbanksystem sehen die gleichen Daten zur gleichen Zeit und das System ist fehlertolerant. Das System kann Probleme bei der Schrift- und Wiedergabevorgang der Daten haben.
- *AP:* Das System ist sehr effizient und fehlertolerant, aber die Konsistenz der Daten zwischen den Knoten ist nicht garantiert.
- *CA:* Die Datenkonsistenz sowie die Schreib- und die Rückkehr der Daten sind gewährleistet. Falls ein Problem auf der Netzebene auftritt, bemerkt man sofort ein Datenkonsistenzproblem.

Diese Architekturen erlauben uns, unsere NoSQL Datenbank nach unseren Bedürfnissen auszuwählen. Die Abbildung 3 zeigt, dass HBase zu der Architektur CP gehört.

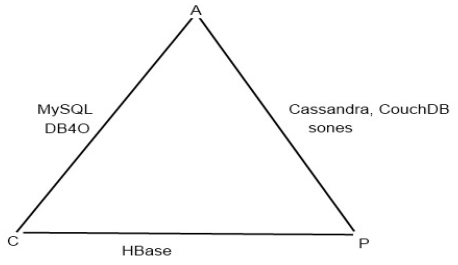

Figure 3: Datenbanksysteme anhand des CAP-Theorems [6]

In diesem Kapitel haben wir einen Überblick über die NoSQL Datenbanken Technologien vorgestellt. Im nächsten Kapitel werden wir ein Beispiel von NoSQL Datenbank präsentieren, insbesondere HBase.

3. HBASE

HBase ist ein spaltenorientierter Key-Value-Data-Store, der entwickelt wurde, um mit dem Hadoop Distributed File System (HDFS) zusammenzuarbeiten [7]. HBase ist ein Datenbankmanagementsystem, das in Java geschrieben ist und hat ein Datenspeichersystem, das auf ein oder mehrere Cluster (eine Anzahl von vernetzten Computern) verteilt ist. HBase wurde im Jahr 2007 von der Firma Powerset erstellt [8]. Im Jahr 2008 kaufte Microsoft die Firma Powerset, die die Entwicklung von HBase abandonniert hat. Im Jahr 2010 wird HBase ein Hauptprojekt der Apache Foundation[9].

In diesem Kapitel werden wir das HBase Datenmodell sowie seine Architektur präsentieren.

3.1 HBase Datenmodell

Das HBase Datenmodell ist nicht schwer zu verstehen. In HBase werden Daten in Tabellen gespeichert, die aus Zeilen und Spalten bestehen. Das HBase Datenmodell wird in sechs Konzepte beschrieben Abbildung 4 :

- *Table:* Die Daten sind in Tabellen zusammengefasst. Die Tabellennamen sind die Zeichenketten.
- *Row:* Ein Row ist eine Zeile in einer Tabelle. Eine Zeile wird durch einen eindeutigen Schlüssel (Rowkeys) identifiziert.
- *Column Family:* dies ist die Gruppierung von Daten innerhalb einer Zeile. Der Colum Family wird bei der Erstellung des Datenbank in HBase definiert. Die Namen der Colum Family sind die Zeichenketten.
- *Column qualifier:* ermöglicht den Zugriff auf Daten in einer Colum Family.
- *Cell:* ermöglicht es die Daten zu speichern.
- *Version (Timestamp):* Die Werte innerhalb einer Zelle können verschiedene Versionen haben. Die Versionen werden durch die Timestamp identifiziert.

In Abbildung 4 haben wir zwei verschiedene Column Families nämlich Customer und Sales. Die Spalten Name und City gehören zur Column Family Customer. Die Spalten Product und Amount gehören zur Column Family Sales. Man bemerkt, dass die Zeilen 1 und 2 gleiche Rowkeys haben, aber sie haben verschiedene Versionen (T1, T2). Die zweite Zeile ist die zweite Version der ersten Zeile. Man kann auch die Versionen konfigurieren, das heißt, wenn die Versionen zu alt werden oder zu viele existieren, werden sie automatisch gelöscht. Ein paar Cells in der Tabelle sind leer und sie verbrauchen keinen Speicherplatz.

Eine Besonderheit mit HBase ist, dass es keine Verbindung zwischen Tabellen gibt. Es gibt kein festdefiniertes Schema wie in RDBMS (relational database management system). Die Sortierung der Elemente einer Zeile erfolgt zuerst nach dem Namen der Column Family, dann nach dem Namen der Spalten.[6] Diese Sortierung ist alphabetisch erzeugt. In der folgende Sektion werden wir jetzt die HBase Architektur darstellen, die sich hinter der HBase Datenmodell versteckt.

Row Key	Timestamp	Customer		Sales	
Customer Id		Name	City	Product	Amount
101	T1	Suresh	Hyderabad		300
101	T2	Suresh Reddy		Books	
102	T1	Lavya Gavshinde	Indore	Fan	600

Figure 4: HBase Datenmodell [10]

3.2 Aufbau von HBase

3.2.1 *Hadoop Ökosystem*

Bevor man die Architektur von HBase eingeht, ist es für uns unerlässlich, das Ökosystem von Hadoop zu verstehen. Hadoop ist eine freie verfügbare Plattform, die die große Datenmengen auf Standard-Hardware in verteilter Form speichert und verarbeitet. Hadoop wurde entwickelt, um gleichzeitig auf mehreren Servern laufen zu lassen (bis zu mehreren Tausend). Die Abbildung 5 zeigt das Hadoop Ökosystem und in dieses Ökosystem befindet sich Hbase. Es gibt viele Komponenten, die wir in diese Seminar nicht angehen werden. Das Kern von Hadoop besteht aus HDFS und Mapreduce.

Das Hadoop Distributed File System (HDFS) ist das Dateisystem, das für die Speicherung sehr großer Datenmengen in dem Hadoop Ökosystem verantwortlich ist. Um eine höhere Zuverlässigkeit und Geschwindigkeit zu erreichen, wird jeder Daten auf drei verschiedenen Servers geschrieben. Das ist das Prinzip der Redundanz in Hadoop. MapReduce ordnet die verschiedenen Aufgaben zu, die von den Knoten des Clusters ausgeführt werden. Da wir einen Überblick über Hadoop Ökosystem haben, werden wir nun die Architektur der HBase studieren.

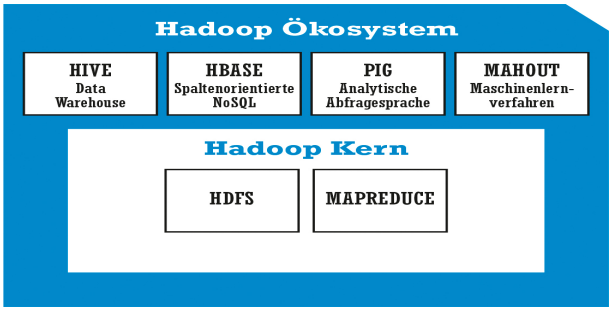

Figure 5: Hadoop Ökosystem [11]

3.2.2 *HBase Architektur*

HBase ist die NoSQL Datenbank des Hadoop Ökosystem. HBase verwendet das HDFS (Hadoop Distributed File System), um seine Daten zu speichern. Es ist praktisch unmöglich, HBase und HDFS zu trennen. HBase verwendet die Algorithmen, um die Lese- und Schreibzufallsdaten durchzuführen. HBase hat viele Komponenten nötig für die Datenspeicherung. Die Abbildung 6 zeigt die Komponenten, die wir nachfolgend im Detail vorstellen werden:

HMaster

Der HMaster ist ein Server, der alle andere Server (Regionserver) auf die Cluster koordiniert und überwacht. Der HMaster weiß, welche Server ihm zur Verfügung stehen und welche Daten bereits verarbeitet wurden. Der HMaster ordnet Aufgaben auf verschiedene Server des Clusters zu. Wenn diese Server ihre Aufgaben erledigen, melden sie sich bei dem HMaster. Der HMaster weiß genau, wo die Ergebnisse auf die Server (Regionserver) sind. In Abbildung 6 arbeit der HMaster mit drei Regionserver.

RegionServer

Der RegionServer verwaltet die Regionen. Er enthält die Methoden (get, put, splitregion, etc.), die die Daten und Regionen verarbeiten. Man bemerkt in der Abbildung 6, dass jeder RegionServer zwei Regionen enthält. HBase wurde entwickelt, um Milliarden von Daten in seine Tabellen zu speichern. Wenn die Größe der Daten in der Tabelle seinen Grenzwert erreicht, schneidet HBase die Tabelle automatisch nach Region mit Hilfe der methode splitregion()[12]. Es ist eine sehr schnelle Operation. Jede Region ist durch eine minimale und maximale RowKey identifiziert. Jede Region ist für die Verwaltung von Tabellendaten verantwortlich.

Region

Region ist für die Speicherung und Verteilung von Daten in HBase verantwortlich. Seine Implementierung ist HRegion[13]. Jede Region verwaltet eine Partition einer HBase Tabelle. Ein Region besteht aus:

- **Store:** Ein Store verwaltet eine Partition einer Column Family. Seine Implementierung ist HStore und es besteht aus mehreren StoreFiles und ein MemStore.
- **Memstore:** Er speichert alle Einträge und Updates einer Partition.
- **HFile:** Das ist die physische Datei, in der die Daten gespeichert werden.

Zookeeper

Der Zookeeper überwacht den Status des Clusters und informiert regelmäßig den Master Server bezüglich der verschiedenen Zustände des Systems. Zusätzlich speichert er kritische Cluster-Informationen wie die Position der Systemtabelle ROOT. **ROOT** enthält die Liste der Meta-Tabellen und ihren Standort. **META** enthält die Liste der Regionen und ihren Standort.

Zusammenfassend wird eine Tabelle in mehrere Partitionen unterteilt. Ein RegionServer verwaltet mehrere Regionen. Eine Partitionstabelle wird von einer Region verwaltet. Eine Region enthält mehrere Stores. Jeder Store verwaltet die Columnfamily in einer Tabelle. Ein Store verwaltet ein MemStore und mehrere storeFile. Ein storeFile verwaltet eine Partition Speicherdatei[14].

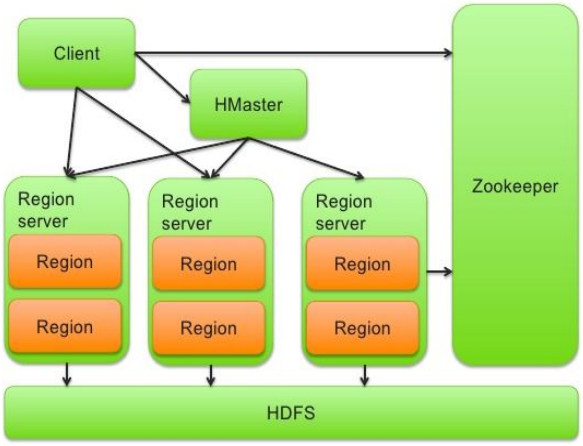

Figure 6: HBase Komponenten [15]

4. DIE SPEICHERUNG MECHANISMUS IN HBASE

In dem vorherigen Kapitel haben wir die HBase Architektur dargestellt. In diesem Kapitel werden wir die Speicherung Mechanismus in HBase eingehen. HBase hat viele Befehle, um seine Daten zu verwalten. Wir präsentieren nur die wichtigsten Befehle, weil die Liste sehr umfassend ist.

4.1 Schreiben von Daten

Das Schreiben von Daten in HBase wird in mehreren Schritten durchgeführt Abbildung 7.Der HBase-Client markiert die Region, die die Tabelle verwaltet, durch die Zookeeper. Die Daten, die vom Client-HBase gesendet werden, sind in der write-ahead log (WAL) geschrieben. Diese Daten werden dann vorübergehend in dem Memstore gespeichert [16]. Die Daten von Memstore sind sortiert und in einer kontinuierlichen Weise indexiert. Die Indexierung wird mit dem Algorithmus Log-Structured Merge Tree (LSM Tree) durchgeführt. Wenn die memstore seine Grenze erreicht, werden alle sortierten Daten in eine neue HDFS-Datei geschrieben.

Der Befehl für die Erstellung der Tabelle ist wie folgt: ***create '<table name>','<column family>'***. Sobald die Tabellen erstellt werden, können wir die Daten lesen und schreiben.

Das Einfügen von Daten in HBase wird mit dem Befehl "put"getan. Die allgemeine Syntax ist von der Form: ***put '<table name>','row1',' <colfamily:colname>','<value>'***. Konkreter wäre ein Beispiel: ***put 'emp','1','personal data:name', 'Charly'***. HBase hat viele andere Befehle wie Create Data, Update Data, Scan, usw. Diese Befehle werden während der Demo unserer Anwendung vorgestellt.

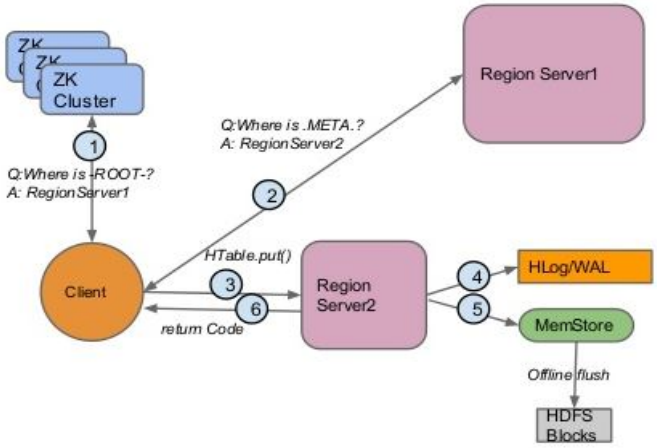

Figure 7: Speicherung Mechanismus in HBase [17]

4.2 Lesen von Daten

Das Lesen von Daten in HBase wird in mehreren Schritten durchgeführt Abbildung 8. Die HBase markiert den Bereich, der für die Partition verantwortlich ist. Der HBase-Client kontaktiert den Zookeeper, um die Root-Tabelle zurückzuholen. Die ROOT-Tabelle gibt dem HBase-Client der Standortbereich des Meta-Tabelle [18]. Der HBase-Client kann jetzt die Region lokalisieren, die er sucht. Die HBase Client sendet eine Abfrage an RegionServer, die die gewünschten Informationen enthält. Die allgemeine Syntax zum Lesen von Daten in HBase ist: ***get '<table name>','row1'***. Um einen bestimmten Column zu lesen, können wir den folgenden Befehl verwenden: ***get 'table name', 'rowid', COLUMN ⇒ 'column family:column name '***

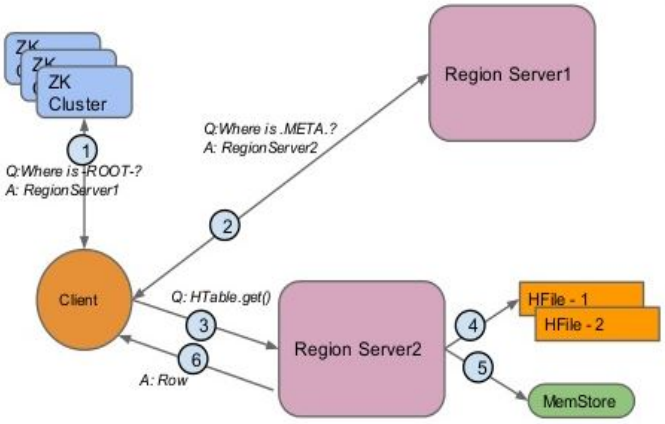

Figure 8: Lesen Mechanismus in HBase [19]

5. FAZIT

Die NoSQL-Datenbanken werden nun zunehmend genutzt. Wir glauben, dass in den kommenden Jahren die NoSQL-Technologien zunehmend genutzt werden, um die Problematik großer Mengen von Daten zu lösen. Im Rahmen unserer Arbeit haben wir uns auf HBase Technologie konzentriert. Diese Arbeit hat uns erlaubt, den allgemeinen Betrieb des HBase Datenbank zu verstehen. Zunächst untersuchten wir die Architektur von HBase. Dann haben wir das Datenmodell von HBase erforscht und wir haben seine Installation durchgeführt, um seine Funktionsfähigkeit effektiv zu verstehen.

Außerdem hat HBase auch Einschränkungen. Es gibt keine Verbindug zwischen die Tabellen und keine Datentypen für Columns. Wenn Daten sehr homogen und einfach strukturiert sind, ist das HBase Datenmodell nicht angepasst.

Schließlich können wir sagen, dass HBase eine ideale Lösung für die Verwaltung von großen Datenmengen ist, denn sein Datenmodell ist einfach zu bedienen. So haben die großen Websites wie Facebook die HBase-Datenbank gewählt, um ihre Daten zu verarbeiten.

6. ABBILDUNGSVERZEICHNIS

[1] Definition. http://michaelmorello.blogspot.de/2011/08/glossaire-lusage-de-ceux-qui-decouvrent.html. Besucht am: 2015-11-04.

[2] Skalierung command. http://habacht.blogspot.de/2007/10/was-ist-skalierbarkeit.html. Besucht am: 2016-02-03.

[3] Skalierbarkeit command. http://wi-wiki.de/doku.php?id=bigdata:skalierung. Besucht am: 2016-02-03.

[4] Publications. http://publicationslist.org/data/a.april/ref-377/GTI792_E12_BONJ06048709_rapport-final.pdf, note = Besucht am: 2015-11-20.

[5] Seminar nosql. http://dbs.uni-leipzig.de/file/seminar0910_Brekle_Ausarbeitung.pdf. Besucht am: 2015-10-30.

[6] Skalierbarkeit command. http://www.hanser-elibrary.com/doi/pdf/10.3139/9783446441774.005. Besucht am: 2016-02-03.

[7] Apache-HBase. http://www.searchenterprisesoftware.de/definition/Apache-HBase. Besucht am: 2015-11-04.

[8] Seminar nosql. http://dbs.uni-leipzig.de/file/seminar0910_Brekle_Ausarbeitung.pdf. Besucht am: 2015-11-04.

[9] Projekt Hadoop. http://blog.octo.com/votre-premier-projet-hadoop/. Besucht am: 2015-11-10.

[10] Hbase command. http://www.linusnova.com/2013/09/hbase-la-base-nosql-de-hadoop/. Besucht am: 2015-11-20.

[11] Skalierbarkeit command. http://blog.isreport.de/wp-content/uploads/2013/11/white-duck-hadoop-grafiken-2.jpg. Besucht am: 2016-02-03.

[12] Seminar nosql. http://www-inf.it-sudparis.eu/COURS/ASR/10-11/projets/Chen-Gerlier/RapportGerlierChen.pdf. Besucht am: 2015-11-15.

[13] Hadoop. http://mbaron.developpez.com/tutoriels/bigdata/hadoop/introduction-hdfs-map-reduce/. Besucht am: 2015-11-04.

[14] Seminar. http://dspace.univ-tlemcen.dz/bitstream/112/8138/1/Etude_comparative_des_bases_de_donnees_NoSQL.pdf. Besucht am: 2015-11-20.

[15] Skalierbarkeit command. http://image.slidesharecdn.com/integrationofapachehiveandhbasefinal-120504182226-phpapp02/95/integration-of-hive-and-hbase-8-728.jpg?cb=1336156216. Besucht am: 2016-02-03.

[16] Seminar nosql. http://dna.fernuni-hagen.de/Lehre-offen/Seminare/1912-SS13/Seminarband.pdf. Besucht am: 2015-10-30.

[17] Skalierbarkeit command. http://image.slidesharecdn.com/storageforbigdata-140207132641-phpapp01/95/storage-systems-for-big-data-hdfs-hbase-and-intro-to-kv-store-redis-jpg?cb=1391782647. Besucht am: 2016-02-03.

[18] Hbase nosql. http://blog.mahamadoutoure.com/2015/01/hbase-un-systeme-de-base-de-donnees.html. Besucht am: 2015-10-30.

[19] Skalierbarkeit command. http://image.slidesharecdn.com/storageforbigdata-140207132641-phpapp01/95/storage-systems-for-big-data-hdfs-hbase-and-intro-to-kv-store-redis-jpg?cb=1391782647. Besucht am: 2016-02-03.