Wide Column Stores

Felix Bruckner

Hochschule Mannheim
Fakultät für Informatik
Paul-Wittsack-Straße 10
68163 Mannheim
felix.bruckner@gmail.com

Zusammenfassung Wide Column Stores sind verteilte nicht-relationale Datenbanksysteme, welche ihre Daten in einem für den Petabyte-Bereich ausgelegten Datenmodell abspeichern. Diese Art von Datenbanken werden mit den sich ändernden Anforderungen an moderne Datenbanksysteme hinsichtlich Verfügbarkeit und Perfomanz unter Schreiblast sowie der Möglichkeit sehr große Datenmengen effektiv auswerten zu können immer populärer. Diese Arbeit soll einen Überblick über die Technik und Funktionsweise von Wide Column Stores und Schnittstellen zur Auswertung ihrer Daten liefern.

Seit einigen Jahren geht der Trend immer mehr in Richtung Web 2.0 (von Benutzern erzeugte Daten) und sozialen Plattformen. Hierbei gehen die Benutzerzahlen in die Millionen und sind stetig wachsend bis nahe an die Milliarde heran, z.B. verzeichnete Facebook Ende 2011 eine Benutzerbasis von ca. 800 Millionen Menschen[1]. Damit wächst die Menge von generierten und potenziell auswertbaren Daten gleichermaßen; tagtäglich speichert Facebook über 10 Terabyte (Stand: 2010) an Daten in seinem Rechenzentrum[2]. Für dieserlei Datenmengen reichen herkömmliche, relationale Systeme bei Weitem nicht mehr aus. Aus diesem Grund sind vor allem soziale Plattformen bedeutend für die rapide Entwicklung im Bereich NoSQL - allen voran Wide Column Stores. [3]

1 Grundlagen

In diesem Kapitel werden die nötigen, fundamentalen Konzepte rund um NoSQL angerissen, um einen Grundstein für die weiteren Kapitel zu legen. Insbesondere wird das CAP-Theorem von Brewer näher betrachtet sowie NoSQL und verwandte Begrifflichkeiten definiert.

[1] http://techcrunch.com/2011/12/22/googlesplus/, abgerufen am 29.05.2012
[2] http://www.slideshare.net/royans/facebooks-petabyte-scale-data-warehouse-using-hive-and-hadoop
[3] http://www.slideshare.net/hurrycane/nosql-in-the-context-of-social-web-4348152

1.1 Begrifflichkeiten

NoSQL beschreibt eine Bewegung, die in den letzten Jahren populär geworden ist. Relationale Datenbanken verursachen aufgrund ihrer ACID-Konformität, refentieller Integretät, starker Typisierung und schlechter Skalierbarkeit für viele Anwendungsfälle unnötig komplexe Probleme. Wenn es zum Beispiel nicht nötig ist, kann ein Datenbanksystem ohne due vorher genannten Eigeschaften ausgestattet viel schneller operieren. Der Verlust der ACID-Konformität jedoch sorgt dafür, dass kein standardisiertes SQL mehr möglich ist. Demnach beschreibt der Term NoSQL nicht zwangsläufig die totale Abstinenz einer SQL-ähnlichen Abfragesprache, sondern vielmehr ein beschränktes Subset der SQL-Syntax - ergo steht NoSQL für Not Only SQL (nach [SSK11]).

Der Begriff **Wide Column Store** beschreibt eine Klasse von verteilten NoSQL-Datenbanken, welche ihre Daten in eher wenigen, unterschiedlich langen Zeilen, d.h. flexibler Anzahl von Spalten speichern. Diese Architektur ermöglicht es zum Beispiel Zeitfolgen (Messdaten zu Gerätenummern, IP-Adresse zu Zeitpunkt) und Einträge auf Webseiten (Nachricht zu Benutzer) zu verknüpfen und besonders perfomant abfragbar zu machen.

1.2 CAP Theorem nach Brewer

Das CAP-Theorem nach Brewer ist ein grundlegender Baustein für das Verständnis der Zusammenhänge und Unterschiede von relationalen und nichtrelationalen Datenbanksystemen. Das Theorem besagt, dass von den drei im Folgenden beschrieben Eigenschaften von Datenbanken immer nur jeweils zwei gleichzeitig eingehalten werden können:

- Consistency / Konsistenz
- Availability / Verfügbarkeit
- Partition Tolerance / Partitionstoleranz

Dieser Zusammenhang wurde 2002 am MIT axiomatisch bewiesen und in [GL02] dargestellt[4].

Die Klassifizierung von Datenbanksystem lässt sich auch anhand ebendieser Eigenschaften durchführen, indem man Tupel aus jeweils zwei Eigenschaften bildet[5]

- **A-P** - Key-Value Stores, nicht-relationale Datenbanken (verfügbar - dabei ist Konistenz von eher geringer Priorität)
- **C-A** - Relationale Datenbanken (konsistent sowie verfügbar)

[4] `http://de.wikipedia.org/wiki/CAP-Theorem`

[5] Nach: `http://2.bp.blogspot.com/_S1OQqsgO8Vs/S-NwsyqJwPI/AAAAAAAACfU/Lo44suCk_uw/s1600/UPick2-NoSQL.GIF`

– **C-P** - Einige Nicht-relationale Datenbanken (mit Priorität auf hohe Konsistenz)

Dabei bezeichnet man relationale Datenbanken aufgrund ihrer Eigenschaften als ACID (Atomicity, Consistency, Isolation, Durable) und NoSQL-Datenbanken wie von Brewer in [Bre00] defininert als BASE (Basically Available, Soft State, Eventual consistency)

2 Geschichtlicher Überblick

Dieses Kapitel soll einen kurzen Abriss bezüglich der doch interessanten Entstehungsgeschichte von Wide Column Stores liefern.

Im Jahre 2005 veröffentlichte Google das Paper *Bigtable: A Distributed Storage System for Structured Data*, in welchem Chang et al beschrieben, wie bei Google für viele Dienste ein verteiltes Datenbanksystem entwickelt und für viele Dienste eingesetzt wurde. Heute bildet Bigtable die Grundlage für nahezu jeden Dienst von Google, darunter Google Websuche, Google Maps, Google Mail, Youtube und weitere, wie in [CDG+08] aufgezeigt wurde. Das besondere an Bigtable war zu jener Zeit, aufbauend auf ein verteiltes Dateisystem namens GFS (Google File System) eine Art verteilte Hash Map zu implementieren, welche weitgehend selbstständig die Daten für effiziente, sequentielle Festplattenlesevorgänge bzw. In-Memory Pufferung zur Verfügung stellen konnte.

Dies inspirierte viele Entwicler zu Bigtable-Klonen - z.B. Hypertable (2008) oder Apache HBase (2008), beide Open Source.

Amazon veröffentlichte im Jahre 2007 ein Paper zu Dynamo, ebenfalls ein verteiltes Dateisystem ähnliche GFS welches im Gegensatz zu Googles Lösung aber auf höchstmögliche Ausfallsicherheit ausgelegt war. In [DHJ+07] beschreiben die Autoren das innovative Peer to Peer Prinzip, welches ein autarkes Dasein für jeden Knoten im Cluster sorgt. Damit wird die Beschränkung von GFS - die Abhängigkeit von redundanten Masterknoten - abgeschafft.

Facebook schuf im Jahr 2008 mit dem Open Source Projekt Cassandra eine hocheffiziente verteilte Datenbank basierend auf einem ausfallsicheres verteilten Dateisystem inspiriert von Dynamo und Bigtable.

3 Funktionsweise von Wide Column Stores

In diesem Kapitel soll die Funktionsweise von Wide Column Stores näher erläutert werden, insbesondere das Datenmodell und die Zusammenhänge auf technischer Ebene. Des Weiteren wird erläutert, wie diese Datenbanken Ausfallsicherheit und Skalierbarkeit gewährleisten.

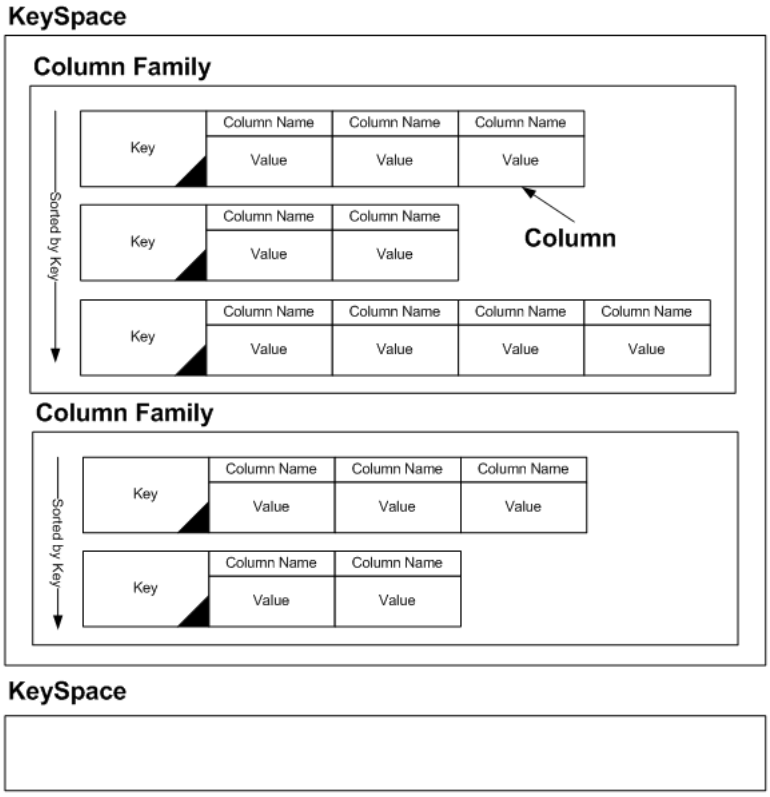

Abbildung 1. Wide Column Store Datenmodell

3.1 Datenmodell

Das Datenmodell[6] von Wide Column Stores unterscheidet sich grundlegend von dem relationalen Ansatz. Zwar existieren die Begriffe Zeile (Row) und Spalte (Column) und bedeuten ungefähr das Gleiche, sind jedoch mit anderen Eigenschaften belegt. Eine Zeile wird anhand eines Schlüssel identifiziert und kann eine beliebige Anzahl an Spalten haben, das heißt also zwei Zeilen können eine unterschiedliche Anzahl an Spalten haben. Wie der Begriff *Wide Column Store* schon impliziert, kann eine Zeile sehr viele Spalten haben; je nach DBMS bis zu einer Million oder mehr ([LM09]) bzw. Zeilen sind in der Regel nicht in ihrer Länge beschränkt [CDG+08].

Zeilen werden anhand ihres Schlüssels mithilfe eines Sortierers sortiert. Diese Sortierer sind meist auf lexikalischer Basis, aber jedoch hinreichend komplex in ihrer Implementierung, da bei verteilten Systemen auf Lokalität geachtet werden

[6] Bildquelle: `http://javamaster.files.wordpress.com/2010/03/cassandra_data_model.png`

muss. Deswegen kann die Wahl des Schlüssels einen erheblichen Geschwindigkeitsvorteil bieten, wenn die Abfragedaten auf möglichst wenigen Knoten liegen.

Eine Menge an Zeilen wird in einem weiteren Konstrukt zusammengefasst. Je nach Implementierung heißen diese Column Family (Cassandra) oder Tablets (Bigtable).

Eine Menge an Column Families wird wiederum in einem Keyspace organisiert. Eine Menge an Keyspaces inkl. dem System-Keyspace bildet das Datenbanksystem.

Ein in dieser Arbeit außen vor gelassenes Konzept sind Super Column Families, welche Innerhalb einer Zeile anstatt Spalten einen weiteren Schlüssel beinhalten, welcher eine Menge von Spalten referenziert; quasi verschacheltete Spalten.

3.2 Technische Aspekte

Wie im Bigtable Paper von Google [CDG+08] beschrieben, besteht eine spaltenorientierte Datenbank aus zwei wichtigen Komponenten:

- Verteiltes Dateisystem (DFS).
- Datenbankaufsatz, welcher die Datendateien auf dem DFS speichert.

Hieraus ergibt sich direkt der Vorteil solcher Datenbanken: Durch die Speicherung von Log- und Datendateien auf einem verteilten Dateisystem, ist die Integrität der Daten zwischen den Maschinen leichter sicherzustellen als zwischen autarken Knoten. Weitere Dienste beinhalten Clustermanagement, Scheduler, Monitoring sowie die eigentliche API für Datenzugriffe.

Skalierbarkeit Die Skalierbarkeit ist in spaltenorientieren DBMS sehr gut, da man vor allem auf Commodity Hardware setzt, d.h. eher viele gewöhnliche Rechner zusammenschließt anstatt wenige sehr leistungsfähige Superrechner. Das ermöglicht einen sehr kostengünstigen Einstieg und des Weiteren eine horizontale Skalierung im Falle des Bedards nach mehr Leistungsfähigkeit.

Ausfallsicherheit und Redundanz Das verteilte Dateisystem ermöglicht einen ausfallsicheren Betrieb. Das Bestreben von Amazon mit Dynamo war ein so ausfallsicheres System wie möglich zu schaffen, deswegen setzt Cassandra auf das Dynamo-Prinzip beim verteilten Dateisystem. Im Gegensatz zu BigTable (welches Masterknoten benötigt) sind die Knoten in einem logischen Ring angeordnet und über ein P2P-Prinzip verbunden. Damit entfällt der Masterknoten. Ein neuer Rechner im Netz benötigt lediglich die IP-Adresse eines anderen Knotens und kann sich damit in den Ring integrieren.

Durch die hohe Anzahl an gewöhnlicher Rechner, ist es nicht nötig besonders ausgefeilte RAID-Systeme oder SAN-Storage zu verwenden. Meist haben die

Rechner gar keine im RAID-Verbund gesicherten Festplatten, sondern setzen auf Masse statt Sicherheit. Um Datenverlust zu vermeiden, werden die Daten repliziert, also mehrere Kopien auf verschiedenen Knoten gespeichert.

4 Anwendungsfälle

Durch die Eigenschaft, besonders viel Schreiblast und Leselast bei sehr großer Datenmenge bewältigen zu können, werden spaltenorientierte Datenbanken vorallem für Web 2.0 Websites eingesetzt.

Durch die Struktur der Daten in den Zeilen bzw. Spalten eignen sich spaltenorientierte Datenbanken besonders gut, um Daten in Abhängigkeit von Zeit zu erfassen, also z.B. Messungen oder Logeinträge.

Google verwendet die hauseigene BigTable-Technologie um Webseitenaufrufe u.a. zeitlich nach Webseite geordnet abzuspeichern. Hierbei betrug das Volumen im Jahre 2006 bereits über 200 Terabyte an komprimierten Rohdaten bei 80 Milliarden Elementen. Weitere Anwendungsgebiete umfassen die Bereitstellung von Kartendaten für Google Earth und Teile der Websuche welche mit ca. 1 Billion Elementen und 800 TB komprimierten Daten zu Buche schlug, beides im entsprechenden Google Paper[CDG+08] dargelegt.

Ein weiteres Beispiel und als Anregung für die letztendlich im Rahmen der Arbeit entstandene Infrastruktur ist die einige Petabyte umfassende Warehousing-Infrastruktur von Facebook zum durchsuchen von Inbox-Nachrichten[7].

5 Cassandra und Map/Reduce

Die Cassandra-Distribution von DataStax[8] bietet auf der Ebene des verteilten Dateisystems CassandraFS eine Hadoop-kompatible Schnittstelle. Hadoop wiederum ist eine verteiltes Dateisystem nach dem BigTable-Beispiel von Google (GFS) und bietet eine Implementierung des MapReduce-Algorithmus an, welcher bei Google entwickelt wurde. Damit lassen sich effizient Berechnungen über große Mengen (mehrere Petabyte) an Daten durchführen, indem die relevanten Daten in einem Computercluster aufgeteilt und parallel verarbeitet werden. Konkret können damit alle in Cassandra abgelegten Daten mit Map/Reduce verarbeitet werden.

MapReduce ermöglicht es, gewisse *Splits* für große Datenmengen zu definieren, um Daten anhand von gewissen, vom Ersteller/Implementierer Eigenschaften zu trennen. Diese Daten und die auszuführenden Aufgaben werden im Cluster verteilt und von den beteiligten Knoten nebenläufig nach dem folgenden Prinzip abgearbeitet werden:

[7] `http://www.slideshare.net/royans/facebooks-petabyte-scale-data-warehouse-using-hive-and-hadoop`

[8] `http://www.datastax.com/`

– **Map-Phase** Dabei werden alle Untermengen parallel im Cluster jeweils in Form von Schlüssel-Wert-Paaren als *Input* in eine weitere Menge *Output* zu Zwischenergebnissen verarbeitet.
– **Reduce-Phase** Für jedes dieser Zwischenergebnisse werden wiederum parallel Ergebnisse berechnet, wobei mehrere Reduce-Schritte je nach Berechnungskomplexität nötig sein können.

Diese Darstellung nach [DG08] ist vereinfacht, Details sind unter `http://de.wikipedia.org/wiki/MapReduce` zu finden.

6 Anwendungsbeispiel: Analyse Sozialer Netzwerkdaten

Im Anwendungsbeispiel für dieses Seminar wird eine exemplarische Analyse über Benutzerinformationen in Sozialen Netzwerken durchgeführt. Hierbei sollen die Seitenaufrufe einzelner Benutzer geloggt und eine demographische Analyse über das Benutzerverhalten durchgeführt werden, wobei Altersdurchschnitt der Besucher sowie das Geschlecht und die Gesamtzahl der Besuche pro Seite ausgegeben werden sollen.

6.1 Datenmodell

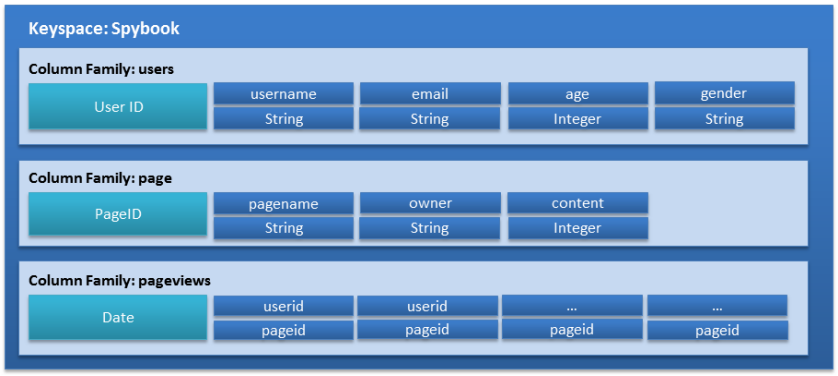

Abbildung 2. Datenmodell desAnwendungsbeispiels

Das Datenmodell besteht aus einem Keyspace namens Spybook (in Anlehnung an Facebook) mit drei Column Families:

– Benutzer
– Seiten
– Seitenaufrufe

Benutzer besitzen einen Benutzernamen, eine E-Mail, ein Alter und Geschlecht als Daten. Seiten haben eine ID, einen Besitzer bzw. Seitennamen und Inhalt während in den Seitenaufrufen einfach die Seiten-ID auf Benutzer-ID nach Tagen unterteilt gespeichert wird.

6.2 Eingangsdaten

Die Eingangsdaten werden mit einem Java-Werkzeug generiert. Hierbei werden einfach diverse vorgenerierte Fantasienamen als Benutzer in die Datenbank eingetragen, einige Seiten erzeugt und dann 100.000 Aufrufe darauf mit zufälligen Benutzer-Kennnummern generiert. Die Daten werden mithilfe der Astyanax-API[9] für Cassandra in die Datenbank geschrieben. Das Beispielprogramm ist in der Dropbox zu finden.

6.3 Verbinden der Daten zwischen Hadoop, Cassandra und Hive

Damit man auf die Daten in Cassandra zugreifen kann, benötigt man entweder wieder eine API, mit welcher man jedoch nur simple Abfragen (keine Aggregationen bzw. Joins) möglich.

Um auch komplexere Abfragen an die Datenbank stellen zu können, benötigt man Map/Reduce. Da Map/Reduce nur mit Hadoop läuft, bietet Cassandra einen HDFS-Ersatz namens CassandraFS. Map/Reduce Jobs wiederum lassen sich mit Hive abstrahieren. Damit man auf die Wide Column Store Daten zugreifen kann, muss man die Daten auf das quasi-relationale Abbild von Hive umbiegen. Ein Mapping kann wie folgt aussehen:

```
CREATE EXTERNAL TABLE
    users(userid string, login string, gender string, age string)
STORED BY
    'org.apache.hadoop.hive.cassandra.CassandraStorageHandler'
WITH SERDEPROPERTIES
     ("cassandra.columns.mapping" = ":key,Login,Gender,Age")
TBLPROPERTIES
    ("cassandra.range.size" = "1000",
     "cassandra.slice.predicate.size" = "1000");
```

Dies erzeugt eine externe Tabelle (eine Art View nach [TSJ+09]) auf die Cassandra-Tabelle namens users.

6.4 Auswertungen via Hive

Die Auswertungen können nun via Hive ausgeführt werden, dabei ist die Syntax sehr ähnlich zu einer SQL-Abfrage, weswegen die Syntax im Rahmen dieser Arbeit nicht näher erläutert werden soll:

[9] `https://github.com/Netflix/astyanax`

```
SELECT
    pv_users.gender, pv_users.pagename,
    COUNT(DISTINCT pv_users.userid), AVG(pv_users.age)
FROM (
    SELECT u.*,p.owner AS pagename
FROM users u
    JOIN pageviews pv ON (pv.userid = u.userid)
    JOIN pages p ON (p.pageid = pv.pageid)
WHERE pv.pv_date = '2012-05-28')pv_users
GROUP BY pv_users.gender,pv_users.pagename;
```

7 Nutzen in der Cloud

Wide Column Stores sind aufgrund der guten Skalierbarkeit eigentlich prädestiniert für Cloudanwendungen. Die Entwicklung geht jedoch aufgrund der immer noch sehr komplizierten Benutzung und entsprechend notwendigen APIs relativ langsam voran.

Erst im Januar 2012 hat Amazon die Wide Colum Store Datenbank DynamoDB freigegeben[10]. DynamoDB skaliert nahtlos mit steigender Last und verwendet dafür von den anderen Amazon Webservices abweichendes Abrechnungsmodell; Es werden Schreib- und Lesedurchsatz abgerechnet. Die Einheiten (Stand: Juni 2012) hierbei sind 0.01 Dollar pro Stunde pro 10 Schreibeinheiten bzw 50 Leseeinheiten[11]. Eine Einheit bezeichnet hierbei Anzahl Schreib-/Lesezugriffe pro Sekunde pro Kilobyte. Steigt also die Last der Andwendung an, erhöhen sich die Lese- bzw. Schreibzugriffe und die Datenbank skaliert durch Zuschalten von Instanzen automatisch (und wird entsprechend teurer.

Weitere Cloudhosting-Angebote mit Wide Column Stores sind zu diesem Zeitpunkt (Juni 2012) rar gesät; cassandra.io bietet als derzeit einzige Plattform Hosting für Cassandra an und Google bietet BigQuery an[12].

8 Ausblick

In dieser Arbeit wurden weitere Möglichkeiten der Auswertung von Daten mithilfe von diversen Schnittstellen außen vorgelassen. Beispielsweise bietet Cassandra die Möglichkeit, statistische Analysen mithilfe von R[13] oder Textsuche mithilfe von Apache Solr[14] zu realisieren.

[10] http://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html
[11] http://aws.amazon.com/dynamodb/pricing/
[12] https://developers.google.com/bigquery/
[13] http://www.datastax.com/dev/blog/big-analytics-with-r-cassandra-and-hive
[14] http://www.datastax.com/docs/datastax_enterprise2.0/search/dse_search_about

9 Schlusswort

In dieser Arbeit wurden die Grundlagen sowie Technik und Schnittstellen von Wide Column Stores erörtert und mit einem Anwendungsbeispiel praktisch dargestellt; insbesondere die Open-Source Lösung Cassandra wurde wegen ihres unkonventionellen Peer-to-Peer Ansatzes näher betrachtet. Eine Erkenntnis dieser Arbeit war ebenfalls, dass die Auswertung sehr großer Datenmengen in den nächsten Jahren immer wichtiger werden wird[15]. Aufgrund ihrer Beschaffenheit, flexiblen Architektur und Schnittstellen sind Wide Column Stores sehr gut für die Anforderungen der Zukunft gerüstet. Wie das Anwendungsbeispiel jedoch gezeigt hat muss noch viel Arbeit in die Benutzbarkeit der Schnittstellen einfließen, um die Datenbanken nicht nur für ausgewiesene Experten bedienbar zu machen.

Literatur

[Bre00] E.A. Brewer. Towards robust distributed systems. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, volume 19, pages 7–10, 2000.

[CDG+08] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[DG08] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DHJ+07] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220, 2007.

[GL02] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):51–59, 2002.

[Hew10] E. Hewitt. *Cassandra: the definitive guide*. O'Reilly Media, Inc., 2010.

[LM09] A. Lakshman and P. Malik. Cassandra: A structured storage system on a p2p network. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 47–47. ACM, 2009.

[SSK11] C. Strauch, U.L.S. Sites, and W. Kriha. Nosql databases. *Lecture Notes, Stuttgart Media University*, 2011.

[TSJ+09] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[15] Hadoop Summit 2012, `http://hadoopsummit.org/program/`