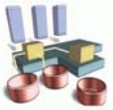
9. Cloud Data Management

- Einführung
- Verteilte Datenhaltung
  - Dateisysteme (GFS)
  - Hbase (Google Big Table)
- Amazon Dynamo
- MapReduce
- MongoDB

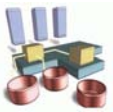


Cloud Computing

"Cloud computing is using the internet to access someone else's software running on someone else's hardware in someone else's data center"

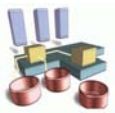
- Lewis Cunningham

- Externe Bereitstellung von IT-Infrastrukturen sowie Applikations-Hosting über das Internet (bzw. Intranet)
  - Public Cloud vs. Private Clouds
- Charakteristika:
  - Illusion unendlicher, „on demand" verfügbarer Ressourcen
  - Virtualisierung: gemeinsame Nutzung v. Ressourcen durch viele Nutzer
  - „Elastizität" - schnelle Belegung/Freigabe von Ressourcen nach Bedarf („Hinzuschalten" weiterer Rechner)
  - Abrechnung nach Verbrauch (CPU Zyklen, Speicherplatz, Übertragungsvolumen)



Cloud Data Management

- Cloud-Anwendungen erfordern Datenhaltung / Austausch von Daten in der Cloud
- Effiziente und persistente Verwaltung großer Datenmengen die Speicherkapazität eines Rechners übersteigen (mehrere TB-PB)
- Möglichkeiten:
  - Verteiltes Filesystem (Google File System, Hadoop DFS)
  - Cloud-Datenbanken: statt RDBMS oft „NO SQL“, z.B. **Key-Value Stores** (Bigtable/Hbase, Dynamo, Cassandra, CouchDB, …)
- Map/Reduce-Paradigma zur Parallelisierung
- Unterstützung für parallele Datenauswertungen / Data Mining
  - OLAP - multidim. Aggregation großer Datenmengen
  - ACID nicht erforderlich



NoSQL-Datenbanken

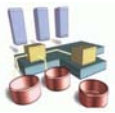
Definition von ***www.nosql-database.org***

Next Generation Databases mostly being
- non-relational
- distributed,
- open-source and
- horizontally scalable.

The original intention has been modern web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply as:

- schema-free, easy replication support, simple API,
- eventually consistent / BASE (not ACID) …

"nosql" is now mostly translated with "not only sql"



NoSQL Produkte/Projekte

www.nosql-database.org listet mehr als 100 NoSQL-Systeme

Key Value Stores / Tuple Stores

- Amazon Dynamo, Voldemort, Yahoo! Sherpa/PNUTS
- Membase, LevelDB …

Wide Column Store / Column Families

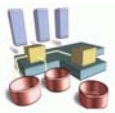
- Hadoop & Hbase
- Cassandra, Hypertable …

Document Stores

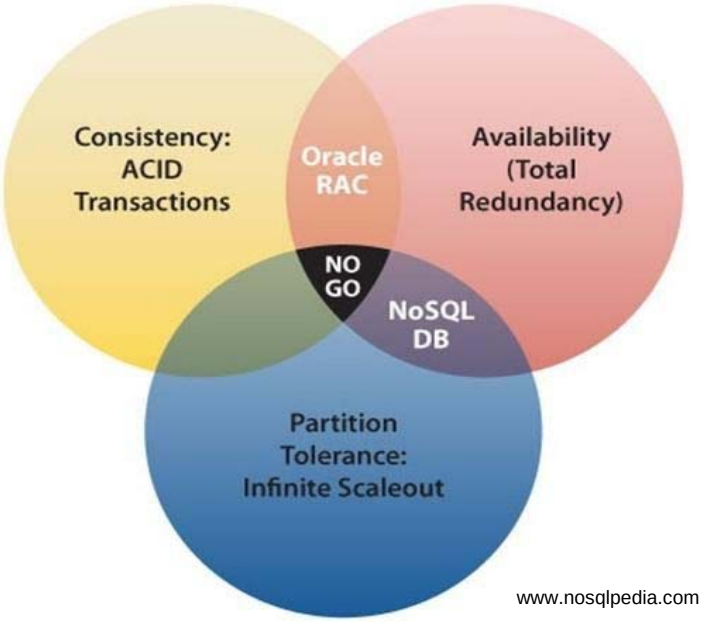
- CouchDB, MongoDB …

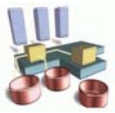
Graph Databases

- Neo4J …



CAP Theorem von Brewer

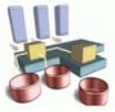




Google File System[1] (GFS)

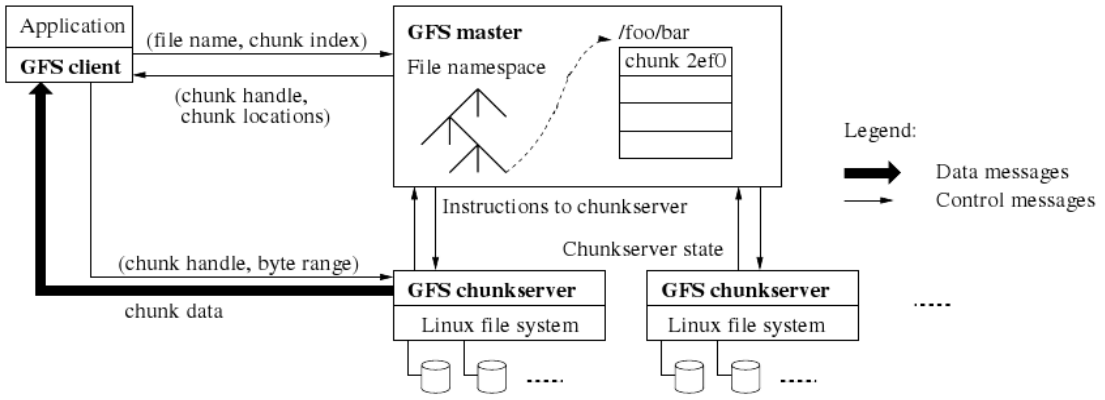
- Proprietäres, verteiltes Linux-basiertes Dateisystem
  - Hochskalierend: tausende Festplatten, mehrere 100TB
  - Open Source-Alternative - Hadoop Distributed File System
- Netzknoten: „billige“ Standardhardware (kein RAID)
  - Hardwarefehler- und ausfälle sind Regelfall
  - Gewährleistung von Datensicherheit
- optimiert für Streaming Access
  - File-Änderungen durch Anhängen: write once - read many times
  - Verteiltes, sequentielles Lesen (blockweise)
- Hoher Durchsatz statt geringer Latenz

[1] S.Ghemawat, H. Gobioff, S. T. Leung. The Google File System. SOSP 2003

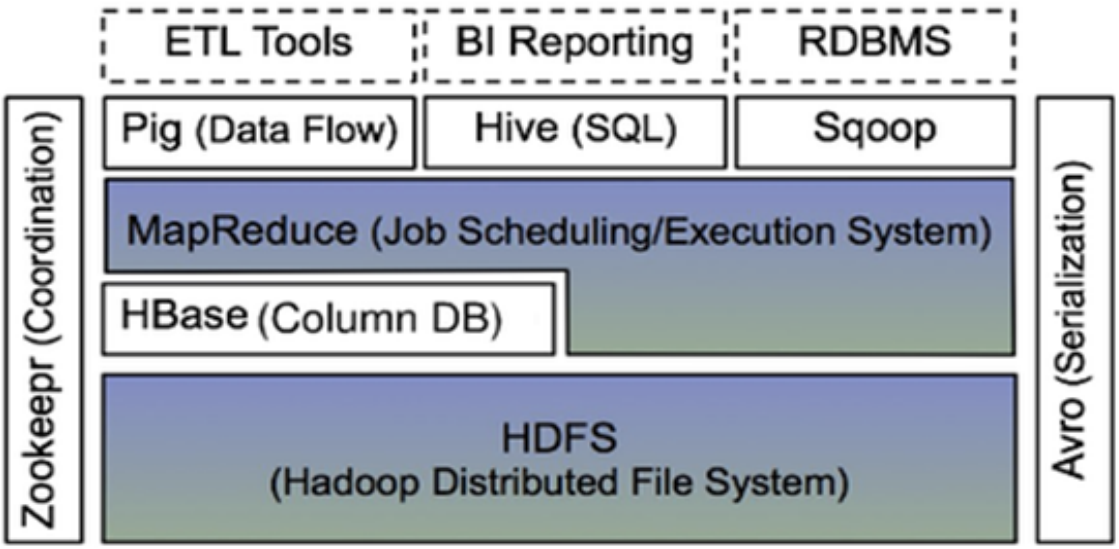


Google File System

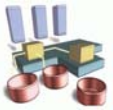
- Physische Datenpartitionierung in Chunks (Default: 64 MB)
  - Verteilung über mehrere Chunkserver (und Replizierung jedes Chunks)
- Master-Server (Metadaten)
  - Mapping: Datei->Chunk->Node
  - Replikat-Management: Default 3 Chunk-Kopien (in 2 Racks)
  - Anfragebearbeitung, Zugriffsverwaltung
- Architektur



Hadoop-Plattform



Source: Cloudera.com

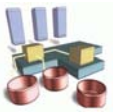


HBase[1]

- Verteilte Datenspeicherung nach Vorbild von Google Bigtable[2]
  - Spaltenorientierter Key-Value-Store
  - Multi-Dimensional
  - Versioniert
  - Hochverfügbar
  - High-Performance
- Ziele
  - Milliarden von Zeilen, Millionen von Spalten, Tausende von Versionen
  - Real-time read/write random access
  - Große Datenmengen (mehrere PB)
  - Lineare Skalierbarkeit mit Anzahl Nodes

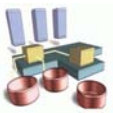
[1] http://hadoop.apache.org/hbase/

[2] Fay Chang, Jeffrey Dean, Sanjay Ghemawat et al. Bigtable: A Distributed Storage System for Structured Data. OSDI'06



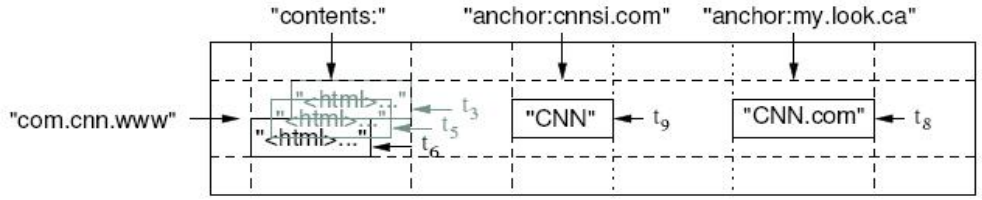
Hbase: Einsatzfälle

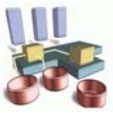
- Web-Tabelle
  - Tabelle mit gecrawlten Webseiten mit ihren Attributen/Inhalten
  - Key: Webseiten-URL
  - Millionen/Milliarden Seiten
- Random-Zugriff durch Crawler zum Einfügen neuer/geänderter Webseiten
- Batch-Auswertungen zum Aufbau eines Suchmaschinenindex
- Random-Zugriff in Realzeit für Suchmaschinennutzer, um Cache-Version von Webseiten zu erhalten



Datenmodell Hbase / Bigtable

- Verteilte, mehrdimensionale, sortierte Abbildung
  (row:string, column:string, time:int64) → *string*
  - Spalten- und Zeilenschlüssel
  - Zeitstempel: mehrere Versionen pro Zelle
- Zeilen
  - (nur) Lese- und Schreiboperationen auf eine Zeile sind atomar
  - Datenspeicherung in lexikographischer Reihenfolge der Zeilenschlüssel
- Spaltenfamilien (column families)
  - Benachbarte Speicherung verwandter Spalten (ähnliche Inhalte)
    Spaltenschlüssel = Spaltenfamilie:Kennzeichen
  - innerhalb Familie: flexible Erweiterbarkeit um neue Spalten





Datenmodell

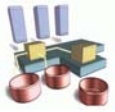
- Konzeptionelle Sicht (alternativ)

Row Key	Time Stamp	Column Contents	Column Family Anchor	
"com.cnn.www"	T9		Anchor:cnnsi.com	CNN
	T8		Anchor:my.look.ca	CNN.COM
	T6	"<html>.. "		
	T5	"<html>.. "		

- Physische Speicherung (Hstore)

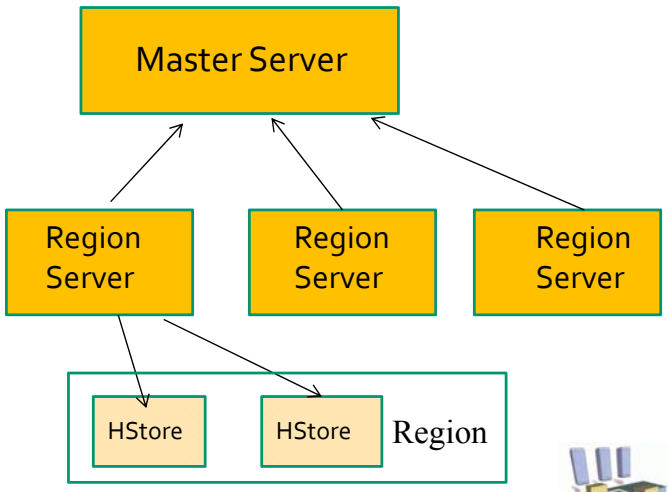
Row Key	Time Stamp	Contents
com.cnn.www	T6	"<html>.."
	T5	"<html>.."

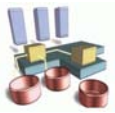
Row Key	Time Stamp	Anchor	
com.cnn.www	T9	Anchor:cnnsi.com	CNN
	T5	Anchor:my.look.ca	CNN.COM



HBase

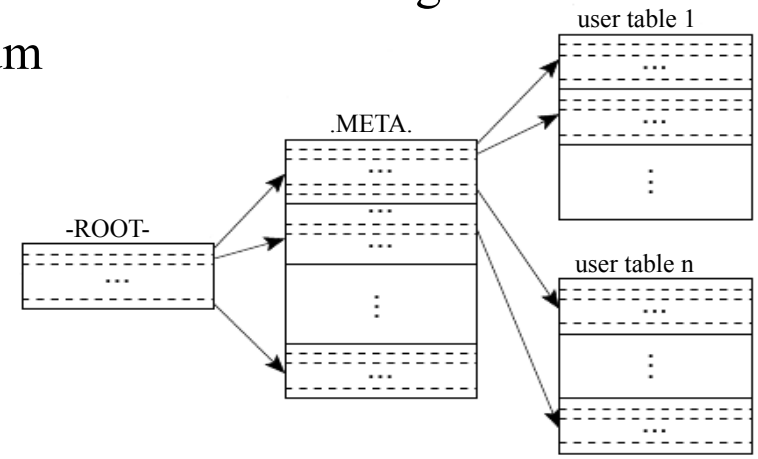
- Regions (Tablets in BigTable)
  - Horizontale Partitionierung von Tabellen in Regions
  - mehrere Region-Server zur Aufnahme der Regions → Lastbalancierung
  - Region-Split in 2 gleich große Regions, wenn max. Größe (z.B. 128 MB) erreicht
- Architektur
  - Datenspeicherung in Region-Server
  - Master
    - weist Regions Region-Servern zu
    - Recovery für Region-Server





HBase

- Zweistufige Katalogverwaltung: Tabellen ROOT, META
- META verwaltet Liste aller (user table) Regions
  - Eintrag enthält 1st Row Key, Location, Status, …
  - Tabellen sind sortiert nach Start-Row-Key
  - pro Tabelle ca. 128.000 Regionen adressierbar (Eintragsgröße 1 KB, Regiongröße 128 MB)
- ROOT: 1 Region mit Liste der META-Regions
- Sehr großer Adressraum

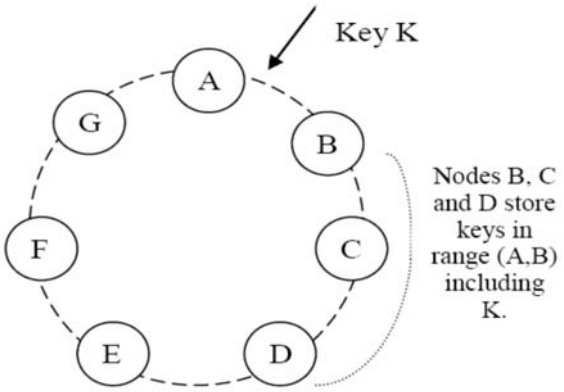


Amazon Dynamo

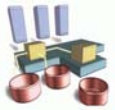
- Verteilter, skalierbarer Key-Value-Speicher
  - v.a. für kleine Datensätze (z.B. 1 MB/Key), z.B. Warenkörbe
  - hochverfügbar
- "Eventually consistent data store"
  - Schreibzugriffe sollen immer möglich sein
  - reduzierte Konsistenzzusicherungen zugunsten Verfügbarkeit
- Performance SLA (Service Level Agreement)
  - "response within 300ms for 99.9% of requests for peak client load of 500 requests per second"
- P2P-artige Verteilung
  - keine Master-Knoten
  - alle Knoten haben selbe Funktionalität

Amazon Dynamo

- Knoten bilden logischen Ring
  - Position entspricht zufälligem Punkt im Wertebereich einer Hashfunktion



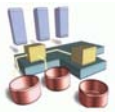
- Knoten speichert Daten, deren Key-Hashwert zwischen Vorgänger und ihm liegt
- jedes Datum über N Knoten repliziert
  - Präferenzliste: Liste der Knoten, die zur Datenspeicherung für einen Key zuständig



Amazon Dynamo

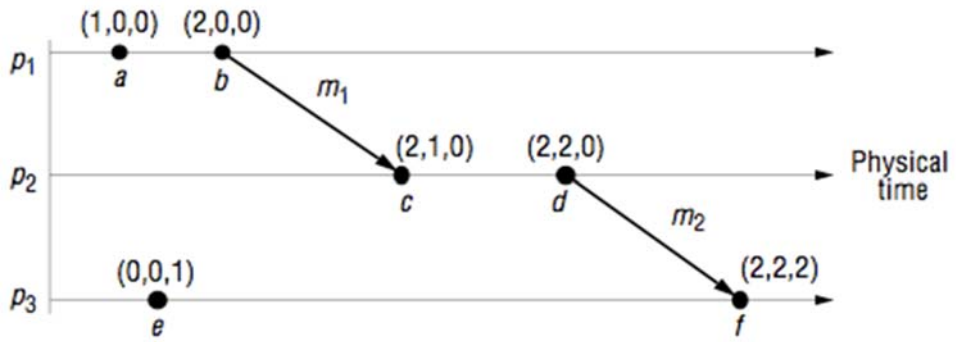
- Verwendung von Read/Write-Quoren
  - R/W minimale Anzahl der N Replikat-Knoten, die für erfolgreiche Read/Write Operation übereinstimmen müssen
  - Anwendung kann (N,R,W) an Bedürfnisse an bzgl. Performanz, Verfügbarkeit und Dauerhaftigkeit einstellen
  - üblich ist (3,2,2)

- Sloppy Quorum
  - strenge Konsistenz erfordert R + W > N, W > R/2 → hohe Zugriffszeiten
  - W=1 → Schreiben möglich, solange mindestens ein Knoten arbeitet
  - Nutzergesteuerte Konfliktbehandlung für abweichende Versionen



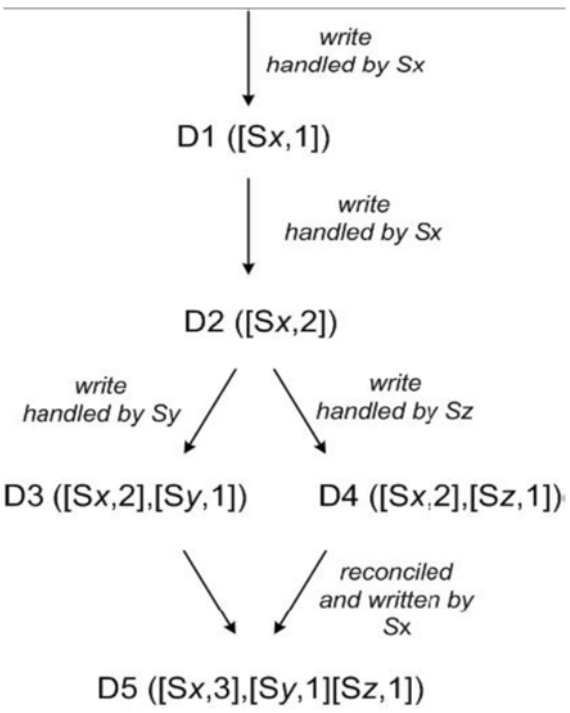
Amazon Dynamo

- Verwenden von "Vector Clocks" um Abhängigkeiten zwischen verschiedenen Versionen eines Objektes darzustellen
  - Versionsnummer/zähler pro Replikat-Knoten
    z.B. **D ([S_x, 1]** für Objekt D, Speicherknoten $S_{x,}$ Version 1
  - Vector Clock : Liste von `(node,counter)`-Paaren zur Anzeige welche Objektversionen bekannt sind
- Beispiel zur Entwicklung von Objektversionen



Amazon Dynamo

- mit Vector Clocks feststellbar, ob 2 Objektversionen aufeinander aufbauen oder nicht
  - Counter der 1. Vector Clock ≤ alle Counter der 2. Vector Clock→ 1.Version ist Vorfahr, kann gelöscht werden
  - sonst Konflikt
- Leseoperation liefert im Konfliktfall alle bekannten Versionen inkl. Vector Clocks
  - darauf folgendes Update führt verschiedene Versionen wieder zusammen
- Anwendung kann Konfliktlösung bestimmen
  - z.B. Mischen verschiedener Warenkorbversionen

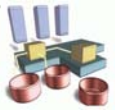


Quelle: G. DeCandia, D. Hastorun, M. Jampani et al. Dynamo: Amazon'sHighly Available Key-value Store. SOSP'07

MapReduce

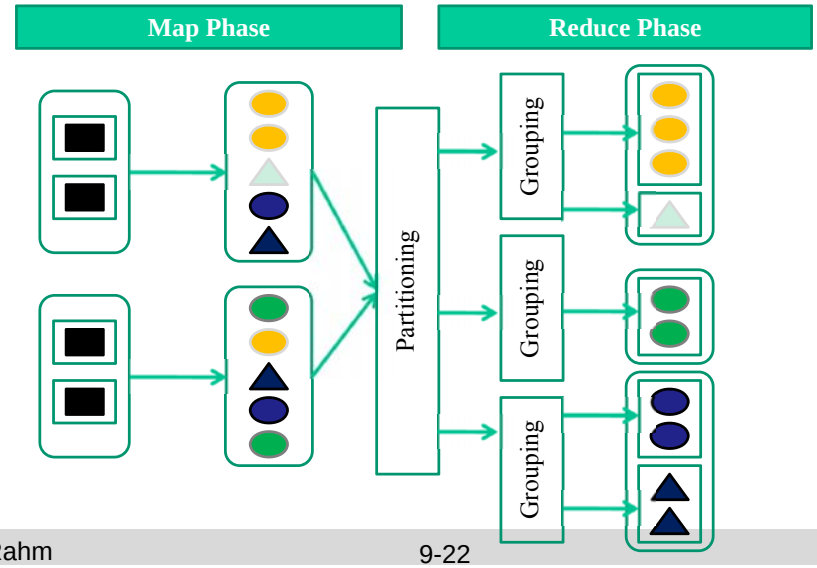
- Framework zur automatischen Parallelisierung von Auswertungen auf großen Datenmengen
  - Entwicklung bei Google
  - Populäre Open-Source-Implementierung: Hadoop
- Nutzung v.a. zur Verarbeitung riesiger Mengen teilstrukturierter Daten in einem verteilten Dateisystem

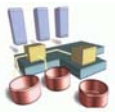
  Konstruktion Suchmaschinenindex, Clusterung von News-Artikeln, Spam-Erkennung …
- Ausnutzung vorhandener Datenpartitionierung (GFS, BigTable)
- Verwenden zweier Funktionen Map und Reduce
  - Map: Verarbeitung von Key-Value-Paaren, Generierung und dynamische Partitionierung von Zwischenergebnissen
  - Reduce: Mischen aller Zwischenergebnisse mit demselben Key; Datenreduktion; Generierung von Key-Value Paaren als Endergebnis



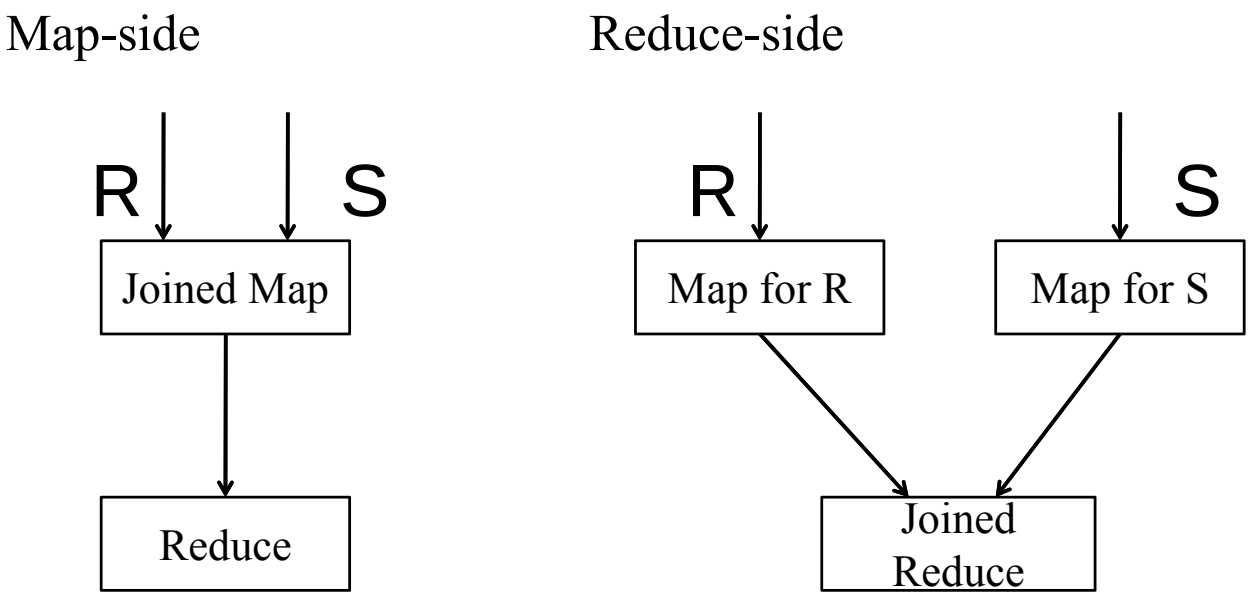
MapReduce

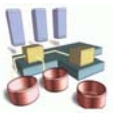
- **Map**-Anwendung pro Eingabeobjekt zur Erzeugung von Key-value Paaren
- Jedes Key-Value-Paar wird einem Reduce-Task zugeordnet
- **Reduce**-Anwendung für jede Objektgruppe mit gleichem Key
- Automatisierte Parallelverarbeitung auf großen Datenmengen



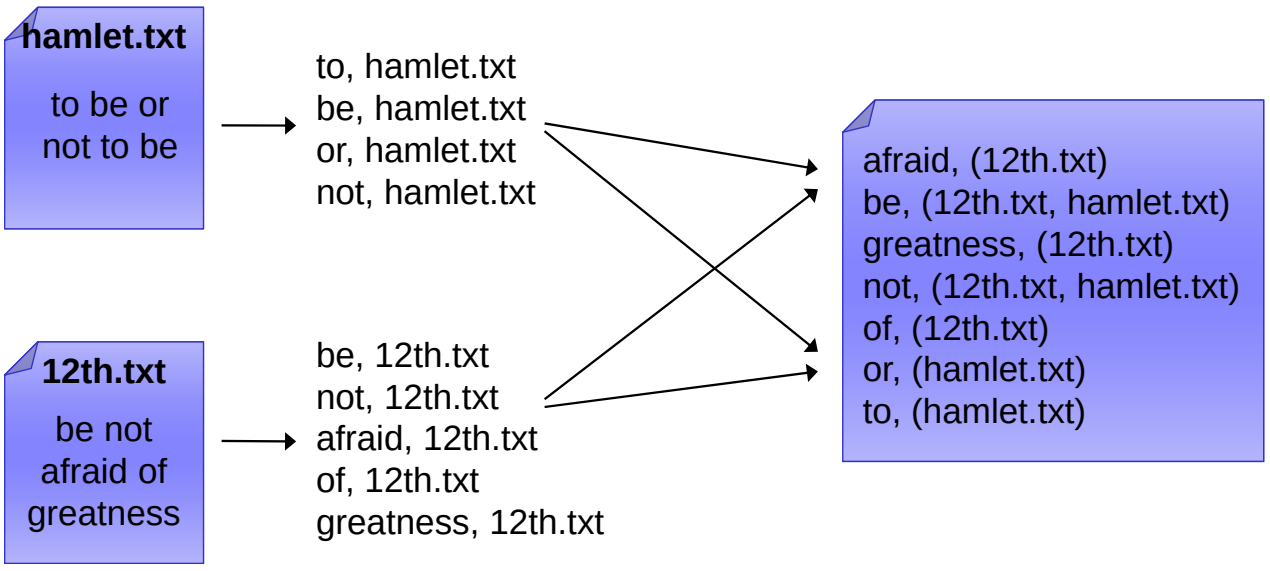


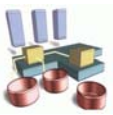
Joins in MapReduce





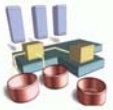
Generierung Text-Index





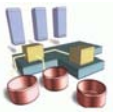
Hadoop/MapReduce vs. Parallele DBS

- Vorteile PDBS
  - Deklarative Anfragesprache (SQL)
  - Automatische Parallelisierung
  - Effiziente Ausführung komplexer Queries und Updates
  - ACID
  - Ausgereifte Technologie für Business-Anwendungen
- Vorteile Hadoop/MapReduce
  - Skalierbarkeit auf sehr große Datenmengen und Cluster
  - Fehlertoleranz
  - Kosten
  - Ständige Weiterentwicklung
- Typische Anwendungsfälle MapReduce
  - ETL, Data Mining
  - Analyse semistrukturierter Daten (Web-Logs, Webseiten, EMail)
  - Einmal-Analysen eines Datenbestandes



MongoDB

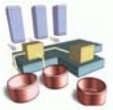
- Verteilter Dokumentenspeicher
  - open source
  - JSON-Dokumente, gespeichert als BSON (Binary JSON)
- Einfache Anfragesprache
  - Indexierung von Attributen möglich
  - Keine Joins
- Automatische Replikation mit Konsistenzwahrung
- Skalierbarkeit auf große Kollektionen durch horizontale Partitionierung (“Sharding”)



MongoDB: Replikation

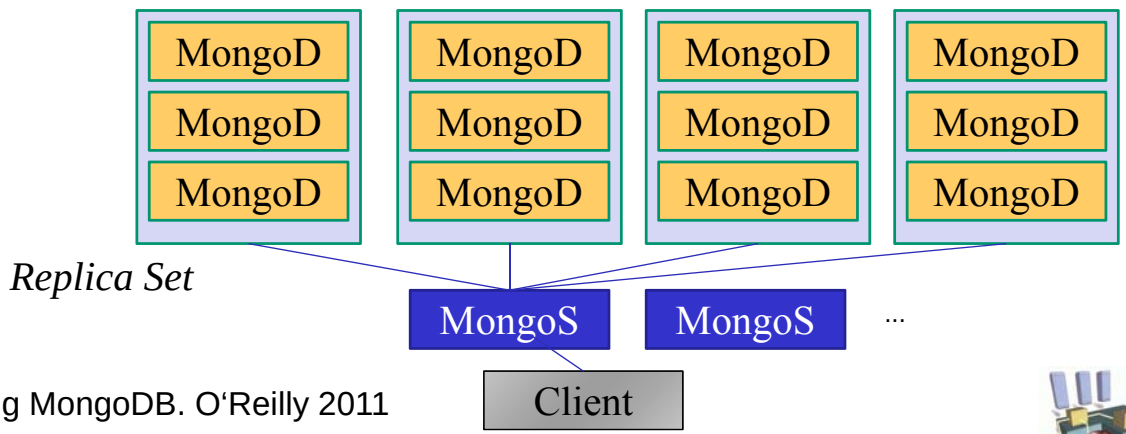
- Asynchrone Master/Slave-Replikation ähnlich Primary-Copy-Ansatz
  - Wählbare Anzahl von Replika (Replica Sets)
  - Leseverfügbarkeit steigt mit wachsender Replikatanzahl
  - Commit von Änderung, wenn Mehrheit der Knoten Daten aktualisiert haben (asynchrone Replikation für noch ausstehende Knoten)
  - Lesen veralteter Daten möglich

- Knotenausfälle
  - Falls Primary ausfällt, dann wählen verbleibende Knoten neuen Primary
  - Neue / wiederkehrende Knoten werden durch Replikation aktualisiert

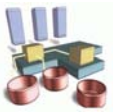


MongoDB: Sharding*

- Horizontale Partitionierung (bzgl. eines Attributs) in Shards
  - jedes Shard ist selbst Replica Set (mehrere MongoDB-Knoten)
- Range-Partitionierung
  - Mehrere Fragmente (Chunks) pro Knoten
  - Automatische Anpassung durch Range-Split wenn Chunk zu groß (Default: 200 MB)
  - Ggf. Migration von Chunks durch Balancer-Prozess
  - Günstige Wahl des Partitionierungs-Attributs entscheidend

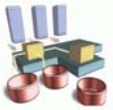


* K. Chodorow: Scaling MongoDB. O'Reilly 2011



Zusammenfassung

- Cloud Computing - Illusion unendlicher, „on demand“ verfügbarer Ressourcen
- Verteilte Dateisysteme, z.B. Google FileSystem
  - replizierte Speicherung großer Datenmengen
- verteilte Speicherung semistrukturierter Daten: Key-Value-Stores
  - Beispiele: Google Bigtable/Hbase, Dynamo
  - Replikation, Lastbalancierung, sehr einfache Operationen
- MapReduce - automatische Parallelisierung von Auswertungen auf großen Datenmengen
- Hohe Fehlertoleranz/Skalierbarkeit auch auf anderen NoSQL-Systemen, z.B. MongoDB



Vorschau SS2012

- Datenbanksysteme 2
- Relationales Datenbankpraktikum (IBM DB2)

- Data Warehousing
- Bio-Datenbanken
- Cloud Data Management