

NoSQL

Datenbanken für Big und Fast Data

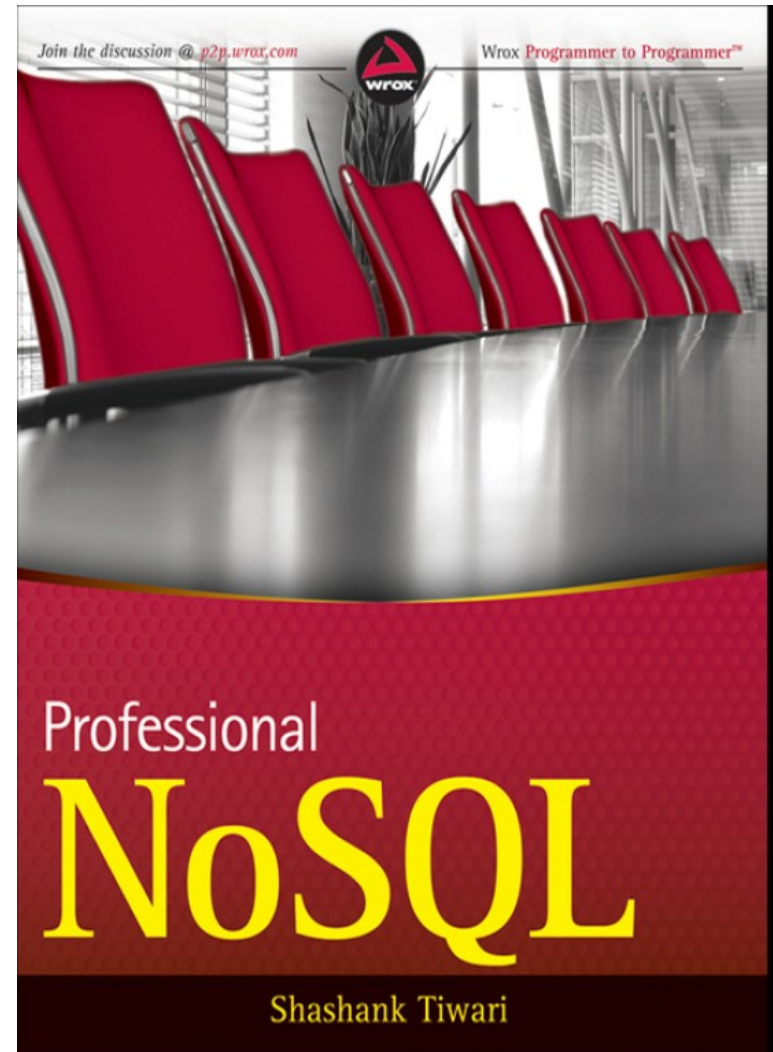
The
Pragmatic
Programmers

Seven Databases in Seven Weeks

A Guide to Modern Databases
and the NoSQL Movement

Eric Redmond
and Jim R. Wilson

Edited by Jacquelyn Carter



- Die benutzten Datenbanken sind entweder Open Source oder werden in der Community-Edition benutzt
- Weitere in diesem Seminar verwendete Werkzeuge sind Open Source
 - LPGL Lizenzmodell
- Benutzt wird eine Vielzahl von Produkten und Technologien
 - Konsolen
 - Abfragesprachen
 - Programmiersprachen
- Programmierung
 - Damit werden die Inhalte durch Übungen vertieft und verinnerlicht
 - Eine Musterlösung wird in elektronischer Form angeboten
 - Diese muss jedoch weder die eleganteste noch beste Lösung sein!

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6

81373 München

eMail: training@rainer-sawitzki.de

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

Die Problemstellung	6
NoSQL	36
Produkte	53

1

DIE PROBLEMSTELLUNG

1.1

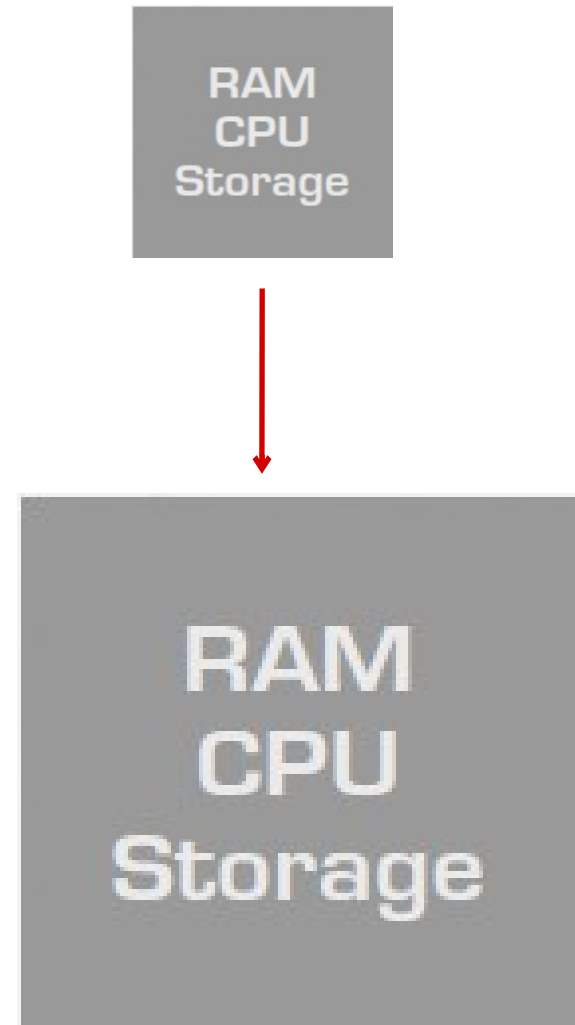
BIG & FAST DATA

- 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

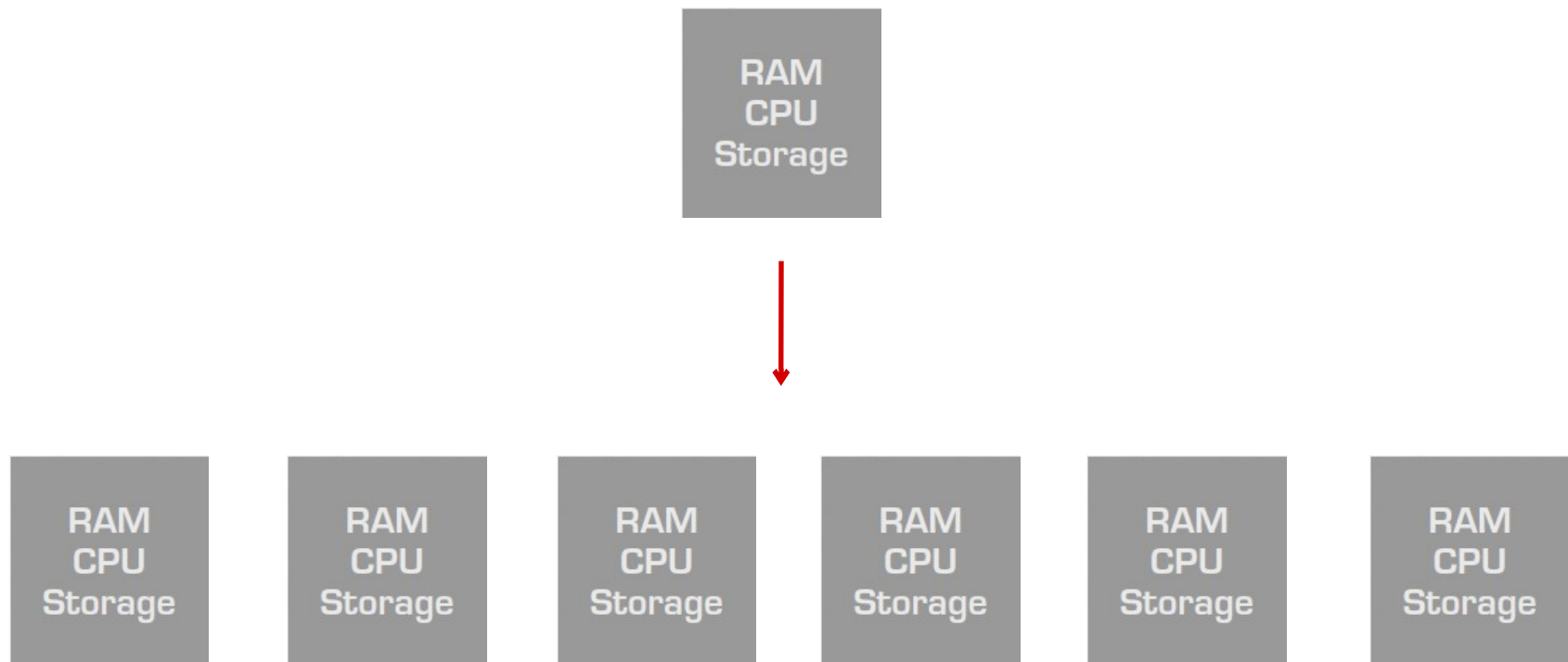
- Völlig verschiedene Bereiche:
 - DNS-Server für Internet
 - Indizierung von Web Seiten für Suchmaschinen
 - Google
 - Monitoring von Seitenzugriffen, aber auch Metrik-Informationen in großen Server-Systemen
 - Content Management und Waren-Systeme
 - Amazon
 - Social Media
 - Facebook

- 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 beliebiger skalierbarer Rechnerkapazität
 - Damit eine Vorstufe zur "Cloud"

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

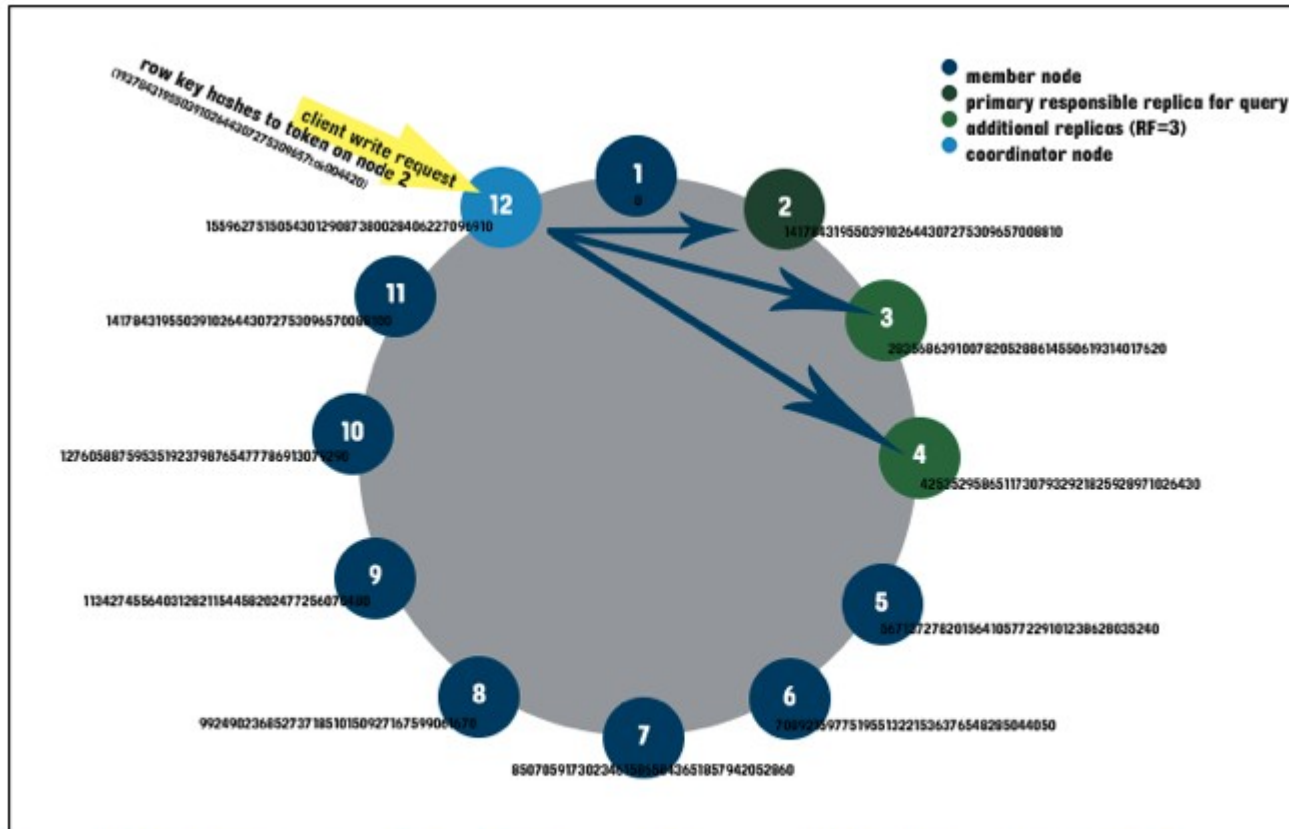


- Grenzen
 - Ausfallsicherheit fordert Replikation über Netzwerk
 - System-Administration und Überwachung



- 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

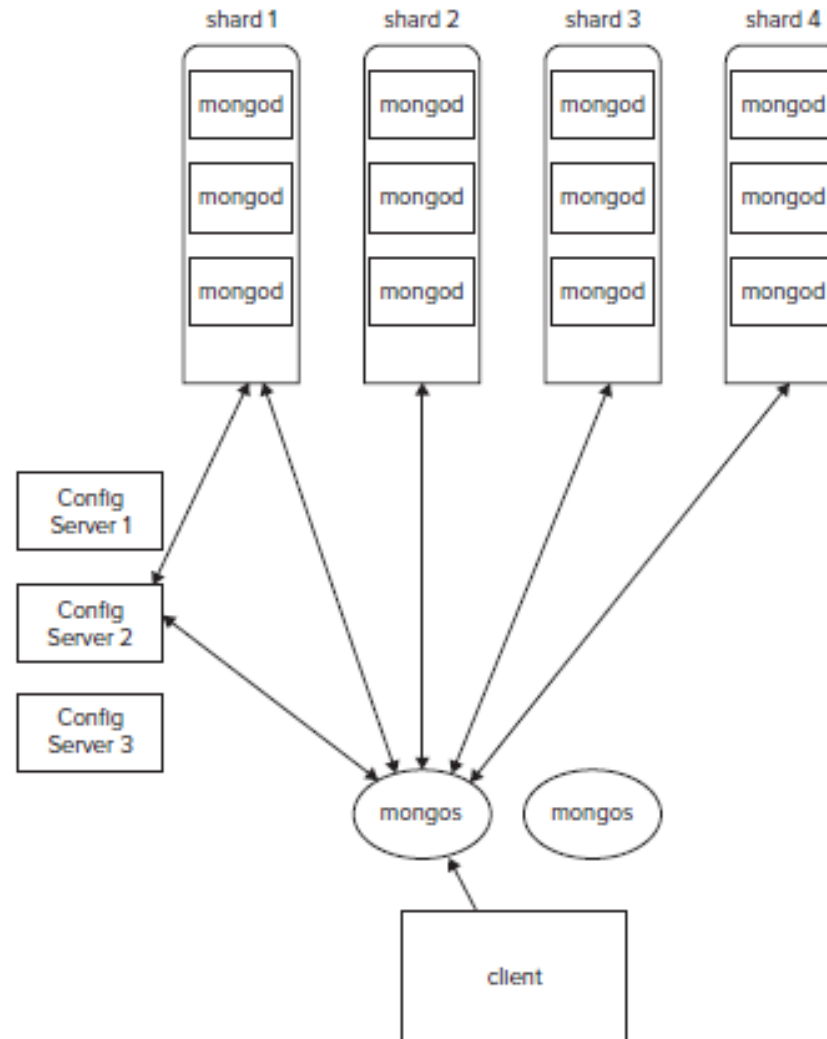
Beispiel: Apache Cassandra Ring

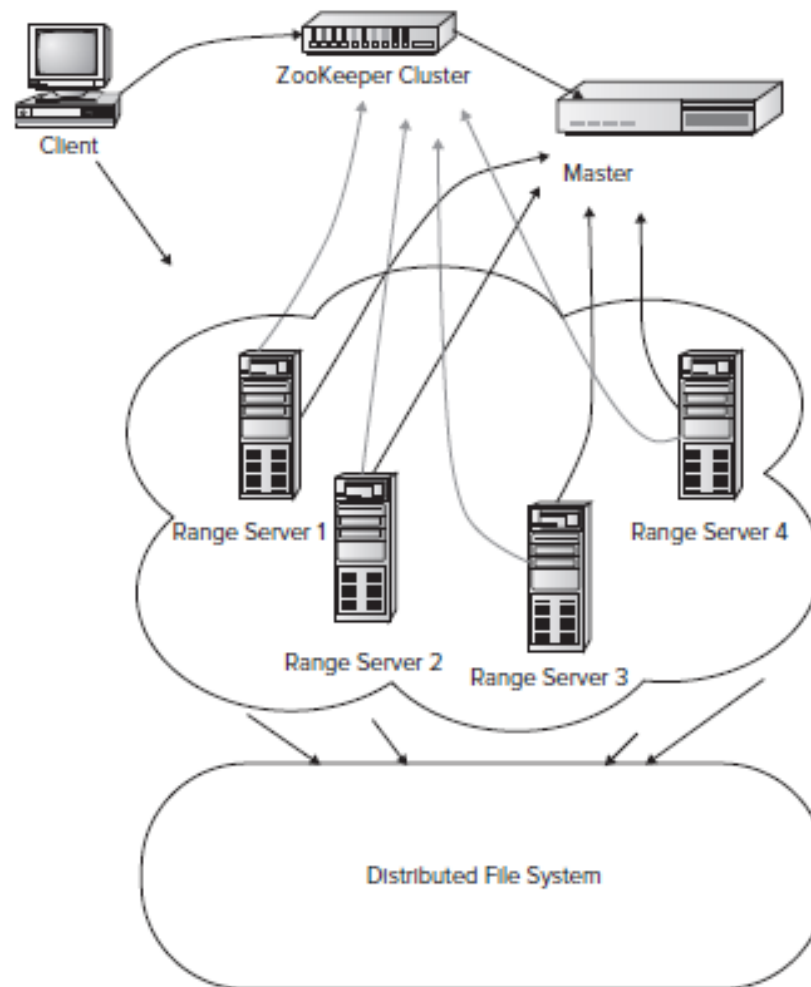


Quelle: <http://www.planetcassandra.org/blog/node-of-the-rings-fellowship-of-the-clusters-or-understanding-how-cassandra-stores-data/>

- 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



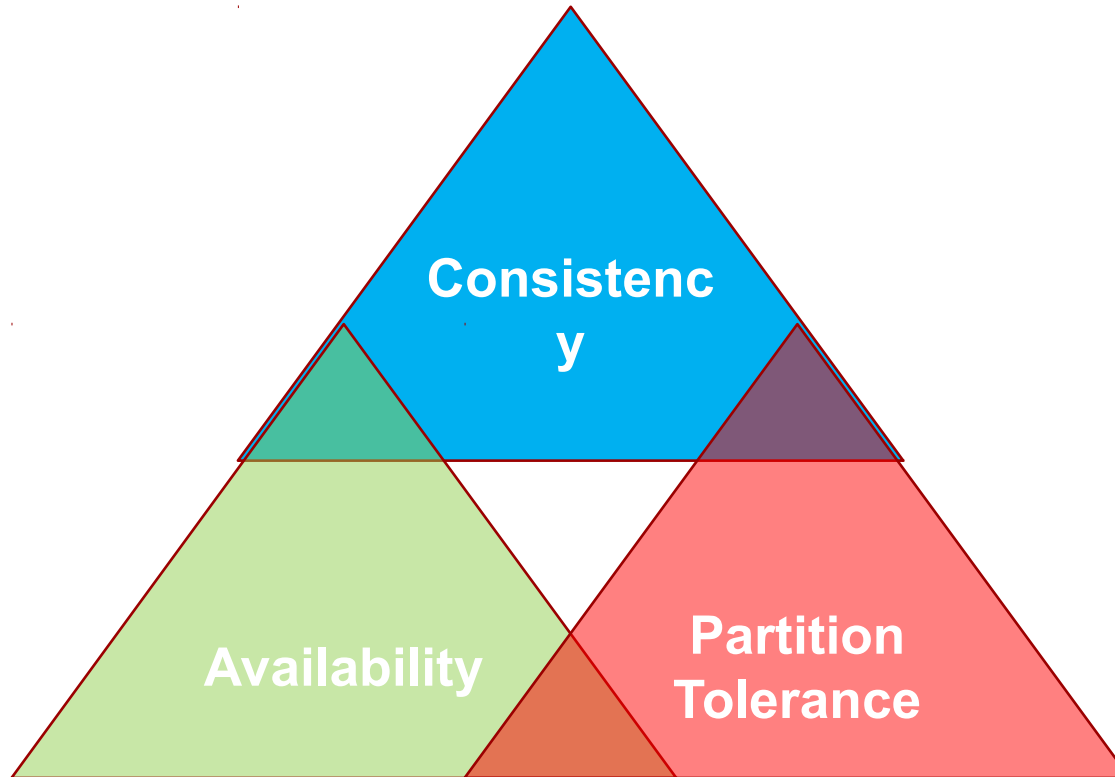


1.2

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!



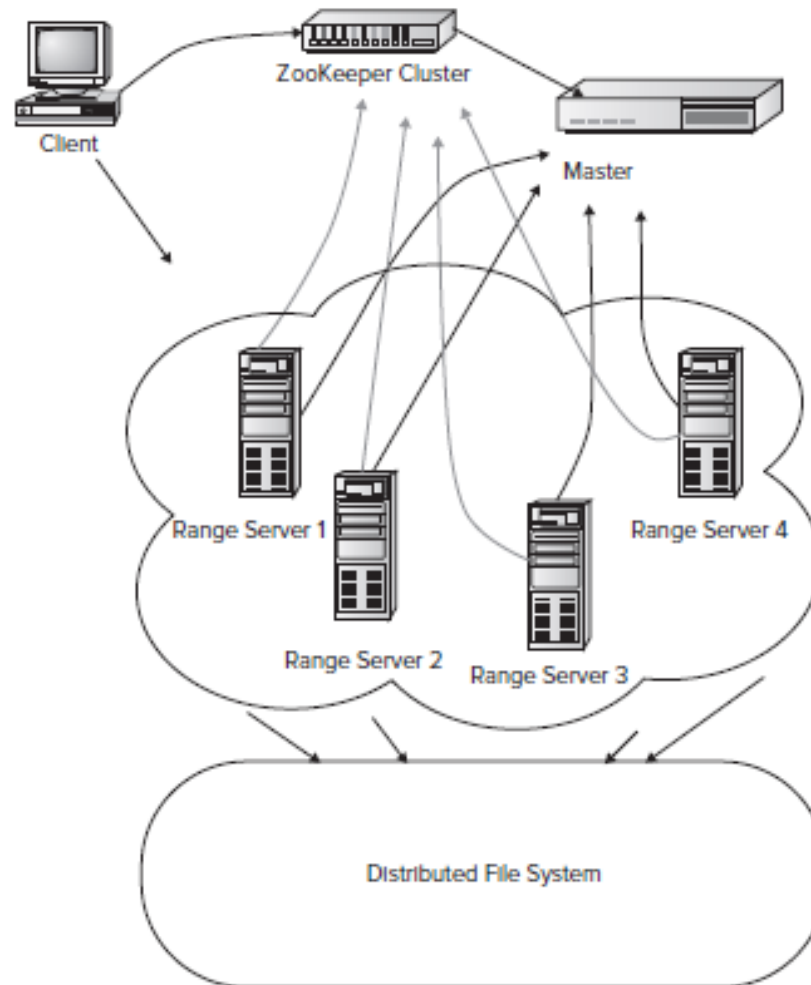
- Consistency
 - Alle Knoten haben **jederzeit** den selben Informationsstand
- Eventually Consistent
 - Änderungen des Datenbestandes werden zeitlich versetzt an die anderen Knoten propagiert
- BASE
 - **B**asically **A**vailable
 - **S**oft state
 - **E**ventual consistency

- 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

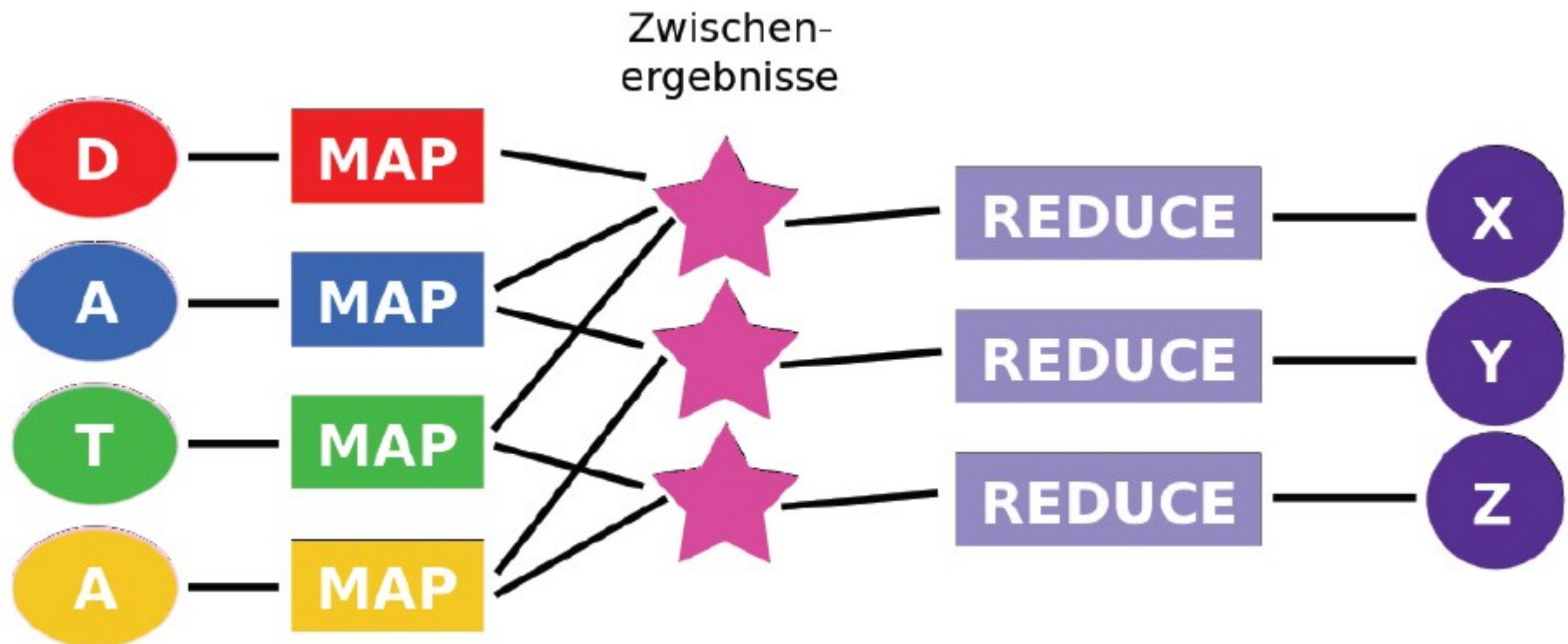
1.3

ANALYSE VON BIG DATA

- Verlangt „neue“ Algorithmen
 - Aufteilung in Jobs
 - Diese werden parallelisiert
 - Operieren auf einem verteilten Datenbestand
- Alle genannten Unternehmen haben in den letzten 10 Jahren Algorithmen entwickelt bzw. eingesetzt
 - Und „netterweise“ der Open Source Community zur Verfügung gestellt
- Fast Data
 - Schneller Zugriff auf Daten-Selektionen
 - Schnelle Umsetzung neuer Abfrage-Anforderungen



- 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>

1.4

EINIGE KLARSTELLUNGEN

NoSQL ist kein Komplettpaket!

- NoSQL-Produkte benötigen
 - Netzwerk-Infrastruktur
 - Netzwerk
 - Router/Load Balancer
 - Dateisystem
 - Überwachung
 - Administration
 - System-Design

Hadoop ist nicht NoSQL!

- Hadoop wird von einigen NoSQL-Produkten benutzt
 - Oder kann benutzt werden
- Ist Bestandteil der Infrastruktur
 - Job-Manager
 - Distributed File System

- Einige Teile der Historie von NoSQL entstammen aus Problemstellungen von Web Anwendungen
 - Session Management
 - DNS-Server
 - Content Repositories
- Viele andere Aufgabengebiete sind aber ohne Bezug hierzu
 - Content Management Systeme
 - Caches
 - Tagging/Indizierung von Big Data
 - ...

- Durch das fehlende statische Schema ist ein Mapping von NoSQL-Strukturen in eine statisch kompilierte Programmiersprache tatsächlich schwierig
 - Aber nicht unmöglich
- Skriptsprachen wie Ruby, Python oder auch JavaScript erzeugen deshalb meistens deutlich kompaktere Programme
 - Man muss diese Sprachen aber definitiv nicht benutzen
 - Mappings nach Java, C# etc. können benutzt werden
- Viele NoSQL-Datenbanken ermöglichen es, MapReduce-Algorithmen beispielsweise mit JavaScript zu formulieren
 - Dies wird jedoch stets auf Server-seite optimiert
 - Es existieren auch native Alternativen
- Das aus JavaScript stammende JSON-Format hat sich unabhängig als Fast-Standard für den Datentransfer entwickelt

- Die Verarbeitung unstrukturierter Daten in einer NoSQL kann eine sehr große Stärke darstellen
 - „Ich muss nicht detailliert wissen, wie etwas strukturiert ist, wenn mich nur bestimmte Teile interessieren“
 - Genaues Format einer Log-Meldung
 - Auswertung von Server-Statistiken verschiedenster Produkte
 - Agile Software-Entwicklung
- Aber auch NoSQL kann Struktur vorgeben
 - Verwendung von XML-Schema
 - Mandatory Properties eines Dokuments



- MapReduce-Funktionen
 - Werden häufig als JavaScript definiert
 - Die Programmlogik einer MapReduce-Funktion ist sehr überschaubar!
- Datenformat
 - Hier hat sich JSON etabliert
- Client-Zugriff
 - RESTful Web Services

2

NOSQL

2.1

DEFINITION

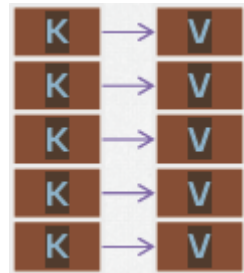
- No SQL!
 - Ursprüngliche Bedeutung
 - Etwas dogmatische Ablehnung relationaler Datenbank-Prinzipien
- Not only SQL
 - Schon deutlich abgeschwächt
 - Reduktion auf geeignete Problemstellungen
- No/ Not only relational
 - Der eigentlich richtige Begriff
 - Fokussierung auf die Daten-Modellierung, nicht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

2.2

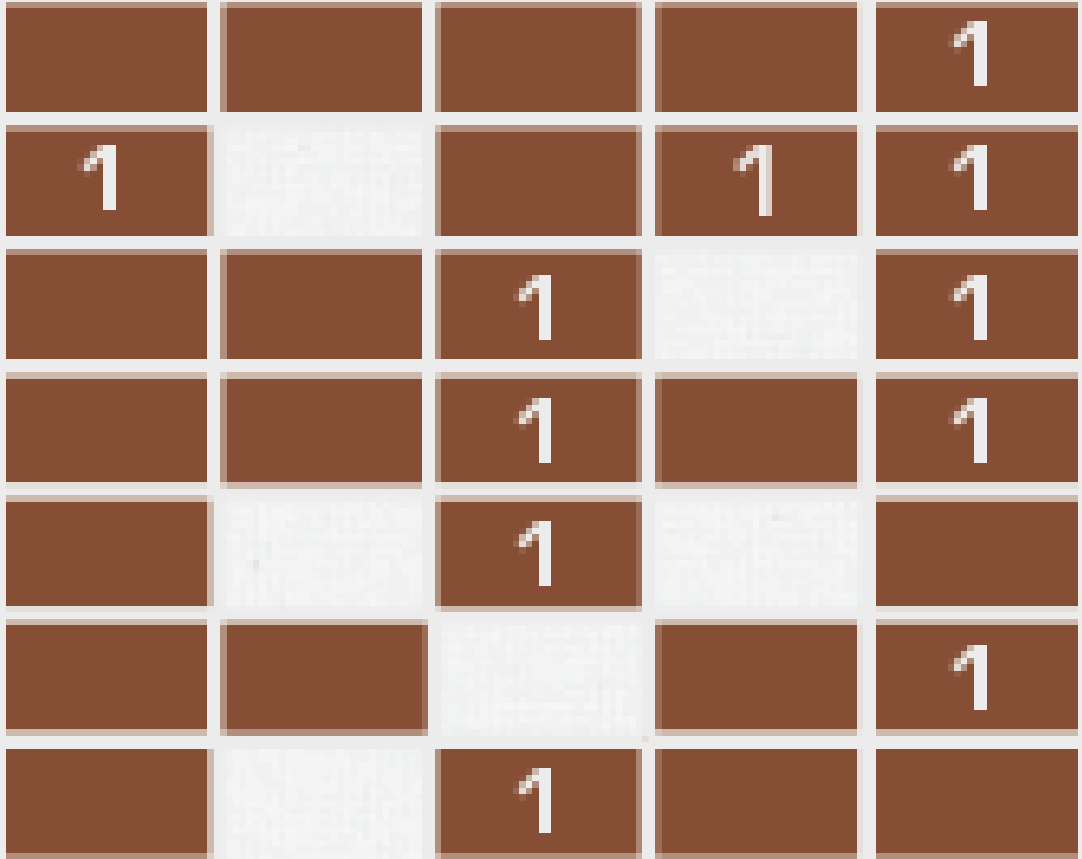
KLASSIFIZIERUNG

- 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

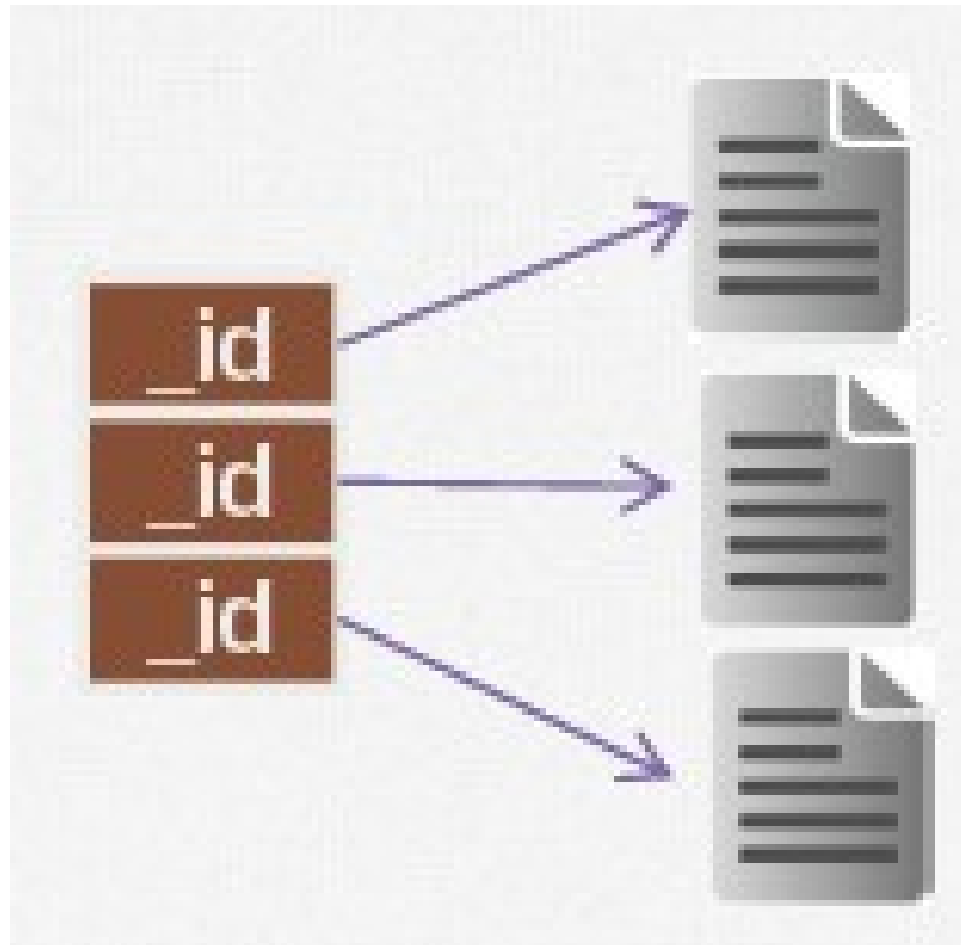


- Daten werden in Columns abgelegt
 - Eine Zeile besteht aber nicht zwangsläufig aus allen Columns
- Diese Columns können noch in Gruppen unterteilt werden
 - Vorteilhaft bei einer verteilten Datenhaltung
- Ausrichtung auf wirklich große Datenmengen



				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.“



2.3

DATENMODELLIERUNG

- Daten werden abstrahiert modelliert
 - Klassenorientierte Programmierung
 - Java, C++/C#, ...
 - Tabellen-Schemata
 - XML-Schema
- Eine Umwandlung von Datentypen erfordert ein Umkopieren
 - „Migration“
- Damit sind Entwicklungszyklen und die Zeiten für die Einführung geänderter Features relativ lang

- „Duck Typing“: Nur vorhandene Eigenschaften und implementiertes Verhalten zählen
 - „Was gelb ist, quakt und watschelt ist eine Ente“
- Dies ist vorwiegend die Domäne von dynamischen Skript-Sprachen
 - Ruby, Python, JavaScript
- Ein Umkopieren von Daten ist nur noch in Ausnahmefällen notwendig
- Damit wird eine agile Software-Entwicklung unterstützt

- Was ist...
 - Besser?
 - Richtig?
 - Komfortabel?
- Leider keine pauschale Antwort möglich
 - „Wer lebend den Raum verlässt hat gewonnen“
- Aber:
 - Der Trend der letzten Jahre geht immer mehr Richtung Dynamil

3

PRODUKTE

3.1

ÜBERSICHT



- **Mongo DB**
 - Eine Dokumenten-orientierte Datenbank
- **Riak**
 - EinKey/Value-Store
- **Neo4J**
 - Eine Graphenorientierte Datenbank
- **HBase**
 - Spalten-orientiert
- Die jeweiligen Dokumentationen stehen bei den Herstellern im Web zur Verfügung
 - Technische Dokumentation
 - Tutorials und Beispiele