3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

# Was Architekten bei NoSQL beachten sollten

Dr. Halil-Cem Gürsoy

adesso AG

# Agenda

- Ein Blick in die Welt der RDBMS
- Klassifizierung von NoSQL-Datenbanken
- Gemeinsamkeiten
- CAP-Theorem - BASE - Transaktionen
- Konsistenz
- Suchen – Map/Reduce
- Skalierung
- Welche Datenbank?


http://www.flickr.com/photos/travis_warren123/4229031035/

# Die RDBMS Welt

- Relationale Datenbanken sind ausgereift („mature“)
- Ausgereifte Backup / Recovery-Lösungen
- Tuning
  - Query Optimizer, Query Caches
- Mit SQL ein Standard (ISO 9075-x u.a.) für eine Abfragesprache
  - Schnelles Einarbeiten in verschiedene Dialekte
- Objektrelationale Mapper sind ausgereift
  - JPA als standardisierte API in der Java-Welt
  - (Java/.NET) Entwickler müssen nur noch selten SQL schreiben

# RDBMS

- „One size fits all“
  - Alles muss in „die“ Datenbank
  - Skalierung von schreibenden Zugriffen schwierig
  - Horizontale Skalierung aufwendig (= $$$)
- Schema-Änderungen nicht trivial
- Produktentscheidungen oft Managment- und nicht Anforderungsgetrieben

# NoSQL

- Aktuell > 122 Datenbanken bei nosql-databases.org gelistet
- Laut Gartner ist der NoSQL-Hype aktuell auf dem Höhepunkt
  - > http://my.gartner.com


Quelle: http://www.indeed.com/jobtrends

# Klassifizierung von NoSQL-Datenbanken

- Grobe Klassifizierung in 4 Typen
  - Key/Value
  - Column Family
  - Dokumentenorientierte Datenbanken
  - Graphenorientierte Datenbanken

- Für eine weiterführende Klassifizierung s. Zusammenfassung von C. Strauch: http://www.christof-strauch.de/nosqldbs.pdf

# Key/Value

- Schlüssel/Wert-Paare
- Häufig optimiert auf viele konkurrierende

  Lese- und Schreibzugriffe
- Häufig als in-memory Datenbanken anzutreffen
- Einsatz z.B. als Cache-System


# Column Family

- Schlüssel/Wert-Paare bilden Spalten
- Unterschiedliche Anzahl an Spalten pro Zeile
- Komplexere Datenmodelle als bei Key/Value DB's möglich
- Keine Joins
- Schemalos
- Oft optimiert auf sehr große Datenmengen in einem verteilten System


HYPERTABLE

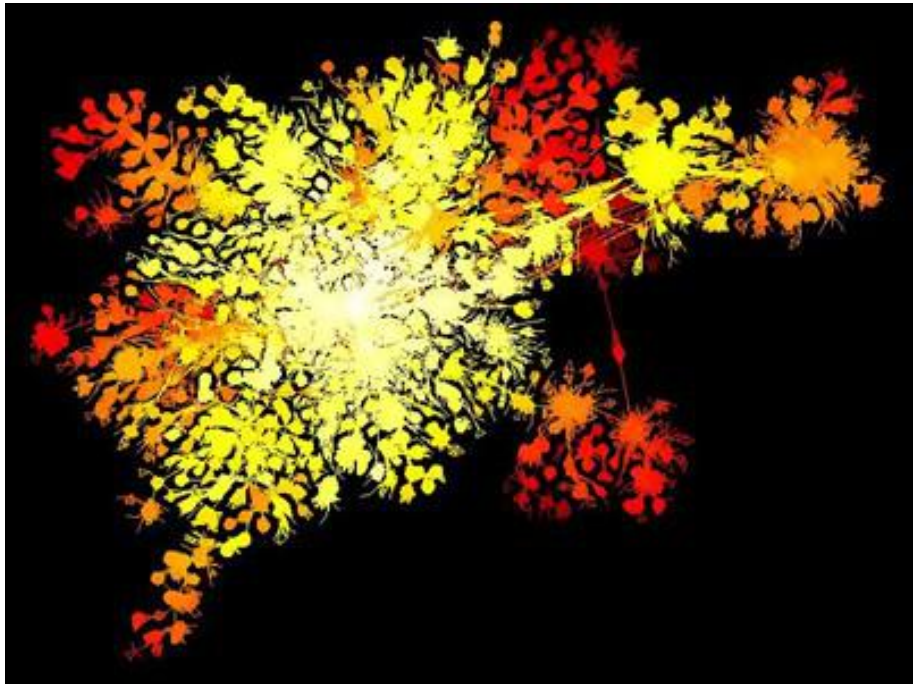
# Dokumentenorientierte Datenbanken

- Daten in Form von semistrukturierten „Dokumenten“
- JSON oder BSON
- Komplexe(re) Datenstrukturen
- Schemalos
- Aber keine Volltextsuche!

```
{"name":"Meier",
 "forename":"Max",
 "adress":{"street":"Deich 7",
           "postcode":28355,
           "city":"Bremen"}
 "comment":"Ok."}
```

# Graphendatenbanken

- Knoten
- (gerichtete) Beziehungen
- Knoten und Beziehungen haben Eigenschaften
- Rekursive Strukturen möglich
- Ebenfalls Schemalos
- Gut geeignet für Abbildung von Netzstrukturen



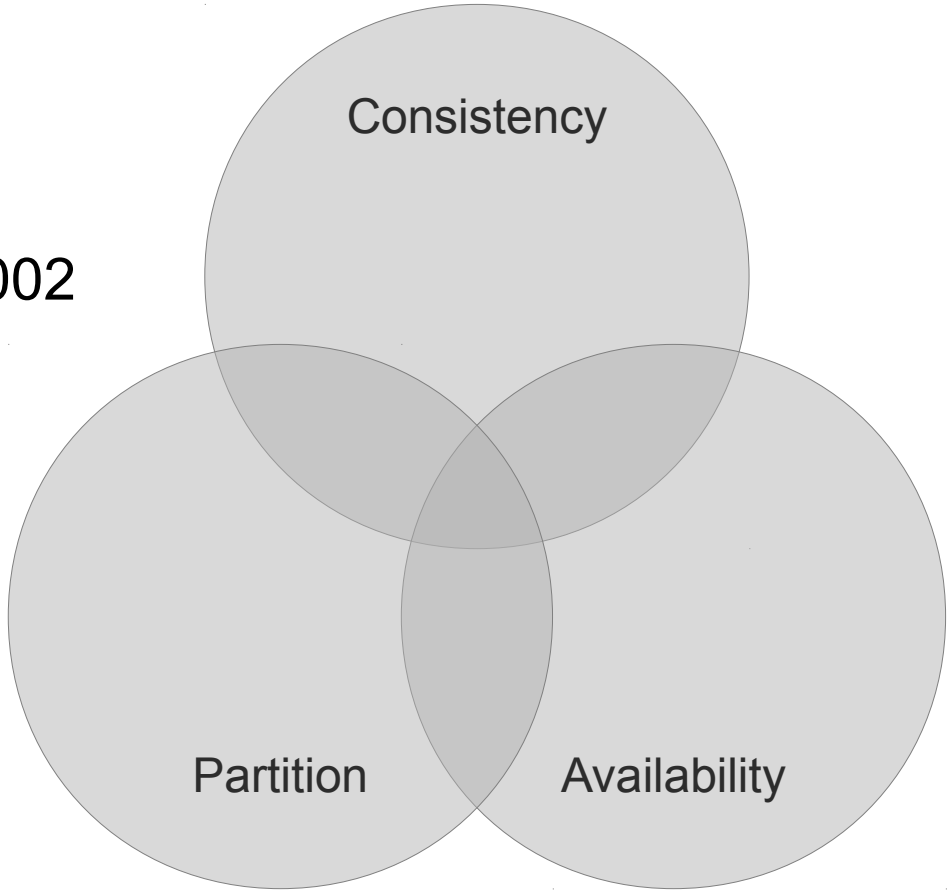
Protein Homology Graph
Quelle:: Alex Adai, ICMB UTex

# NoSQL – Einige Gemeinsamkeiten

- Schemalos
  - Große Freiheit auf der Seite der Datenbank
  - Applikations-Code muss aber dennoch angepasst werden!
- Keine ACID-Transaktionen
  - Atomizität wird meist nur pro Datensatz/Dokument gewährleistet
  - Kein JTA-Support
  - Ausnahme: Neo4J
- Skalierbarkeit
  - Verschiedene Ansätze zur Skalierung und Ausfallsicherheit

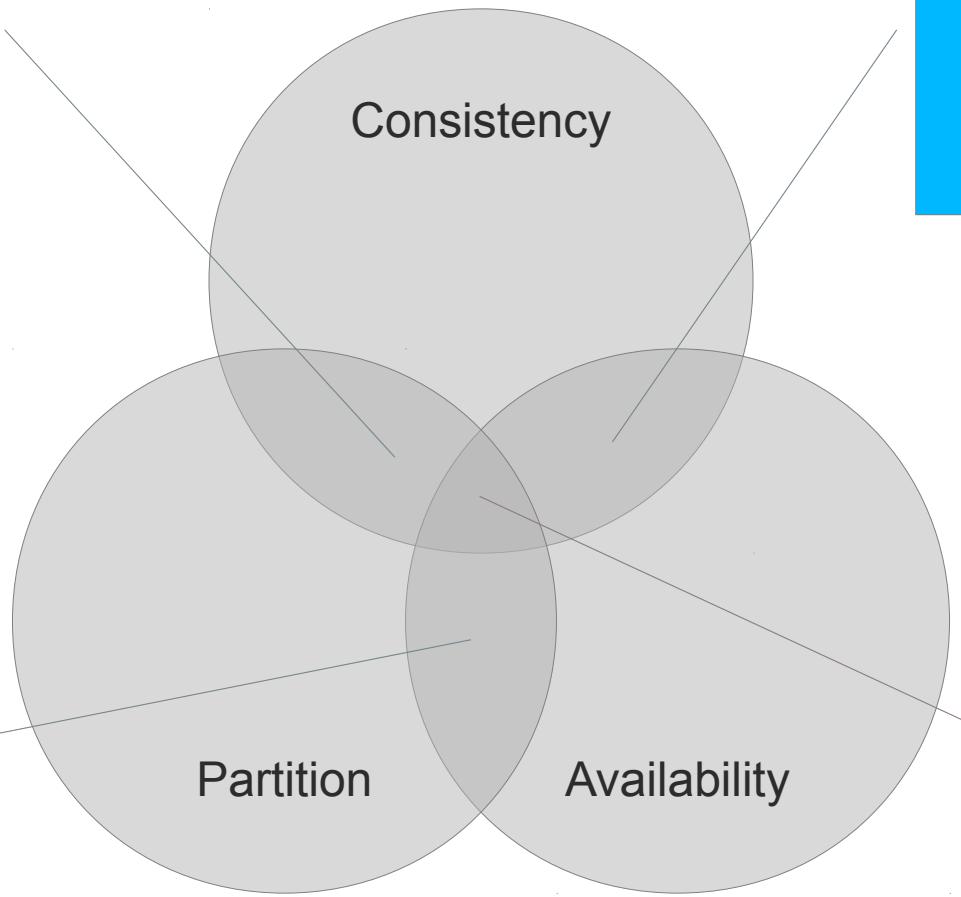
# CAP-Theorem

- Andrew Brewer, 2000
- Gilbert & Lynch, axiomatischer Beweis 2002



# CAP-Theorem

HBase
BigTable
MongoDB




Klassische
RDBMS

DNS
CouchDB
Dynamo
Cassandra

Nicht
Erfüllbar!

# BASE vs. ACID

- CA-Systeme sind ACID-Systeme
  - *Atomicity, Consistency, Isolation und Durability*
- Aber für „V“ und „P“ muss Konsistenz „geopfert“ werden
- Logische Folge: BASE
  - ***B**asically **A**vailable, **S**oft state, **E**ventual consistency*
  - Pritchett, 2008 (http://queue.acm.org/detail.cfm?id=1394128)
- *Basically Available*: Verfügbar im Sinne von CAP
- *Soft state*: Der Zustand des Systems kann sich auch ohne weitere Außeneinwirkung ändern

# Konsistenz

- *Eventual consistency*: **Schlussendliche Konsistenz!**
- **Konsistenz?**
  - BASE
    - Auf allen Replikas sind die gleichen Daten vorhanden
  - ACID
    - Ein widerspruchsfreier Datenbestand
- Beispiel: MongoDB
  - Lesen nur aus dem primären Knoten erlaubt
  - Nur „unter Zwang“ Lesen aus sekundären Knoten
    - Ggfs. veraltete Daten oder Daten nicht vorhanden

# CAP und Apache Cassandra / Amazon Dynamo

- Jeder Knoten ist Master eines „Hashraumes“
  - *Consistent Hashing*
- Daten werden auf *N* Knoten repliziert
- Daten werden von *R* Knoten gelesen
- Beim Schreiben müssen *W* Knoten positive Rückmeldung geben
- *N*, *R* und *W* sind konfigurierbar:
  - *W*=*N* & *R*=1 → Schnelles Lesen
  - *R*=*N* & *W*=1 → Schnelles Schreiben
  - *W*+*R* > *N* → Starke Konsistenz
  - *W*+*R* <= *N* → Schlussendliche Konsistenz

# Suchen?

- NoSQL-Datenbanken bieten minimalen Suchsupport
- Keine Joins
  - `select p. * from person p, posts po where po.p_id = p.id and …`
- „Prejoined data“
  - Speichern des „Join-Ergebnisses“ in *person*
- Daten denormalisieren
  - Ggfs. redundante Daten
- „Reverse Lookup“ speichern
  - Logik liegt in der Applikation

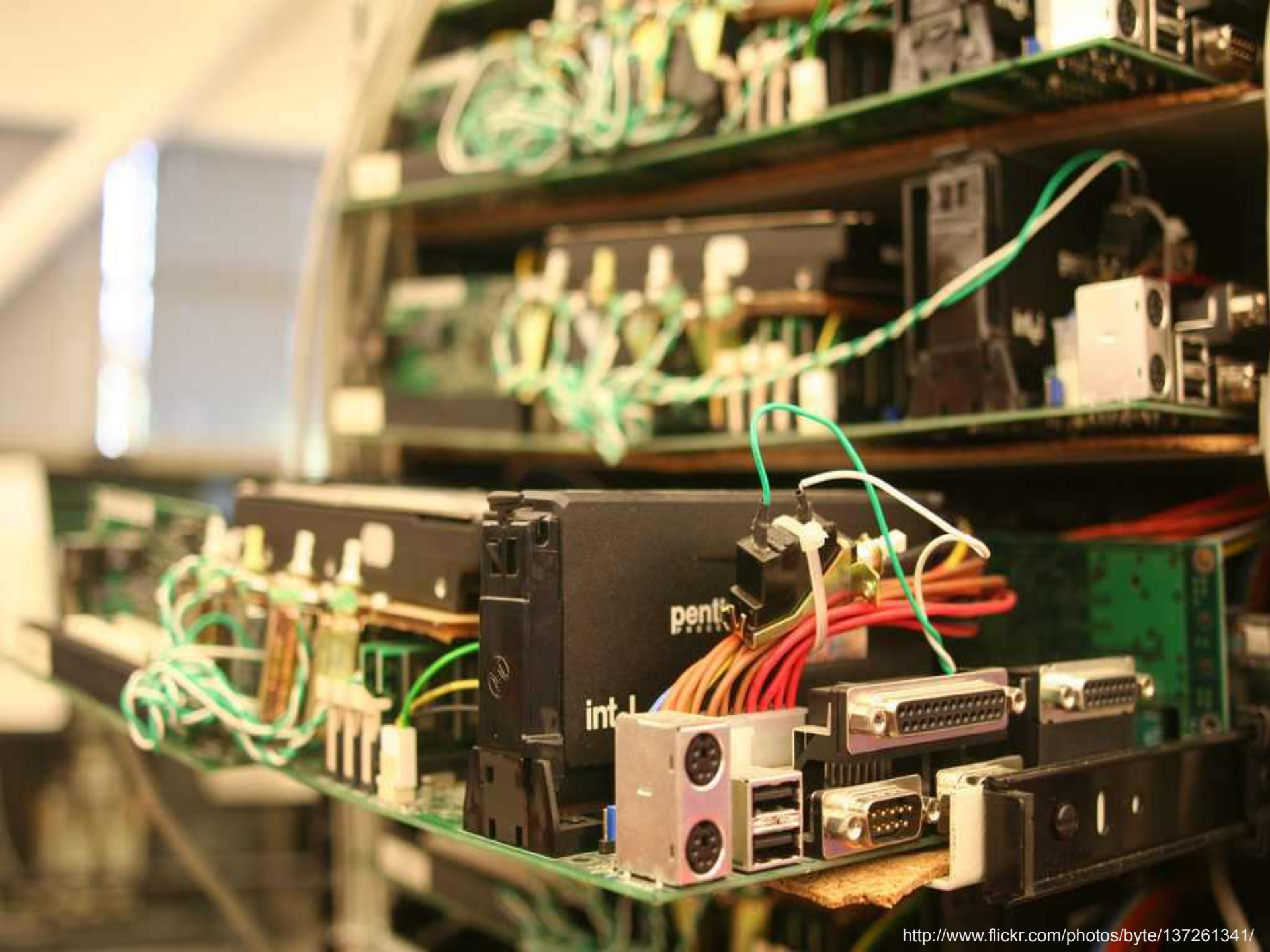
# Suchserver

- Dedizierten Suchserver einsetzen
- Volltextsuche und erweiterte Suchfunktionen
  - „Meinten Sie...“
- Suchserver liefert üblicherweise nur einen Key
- Sehr schnell (optimal index inmemory)



# Map/Reduce

- Schnelles Durchsuchen großer Datenmengen in verteilten DB
- Im Map-Schritt senden einer Abfrage an alle Knoten
- Im Reduce-Schritt erfolgt eine Verarbeitung / Aggregation
  - Ressourcen-Ineffizient (ggfs. lineares Lesen aller Datensätze)
  - hoher Kommunikationsaufwand
  - Abfragezeiten unabhängig von Datenmenge falls Daten verteilt werden
  - Einsatz vieler (günstiger) Server
- Harvest / Yield

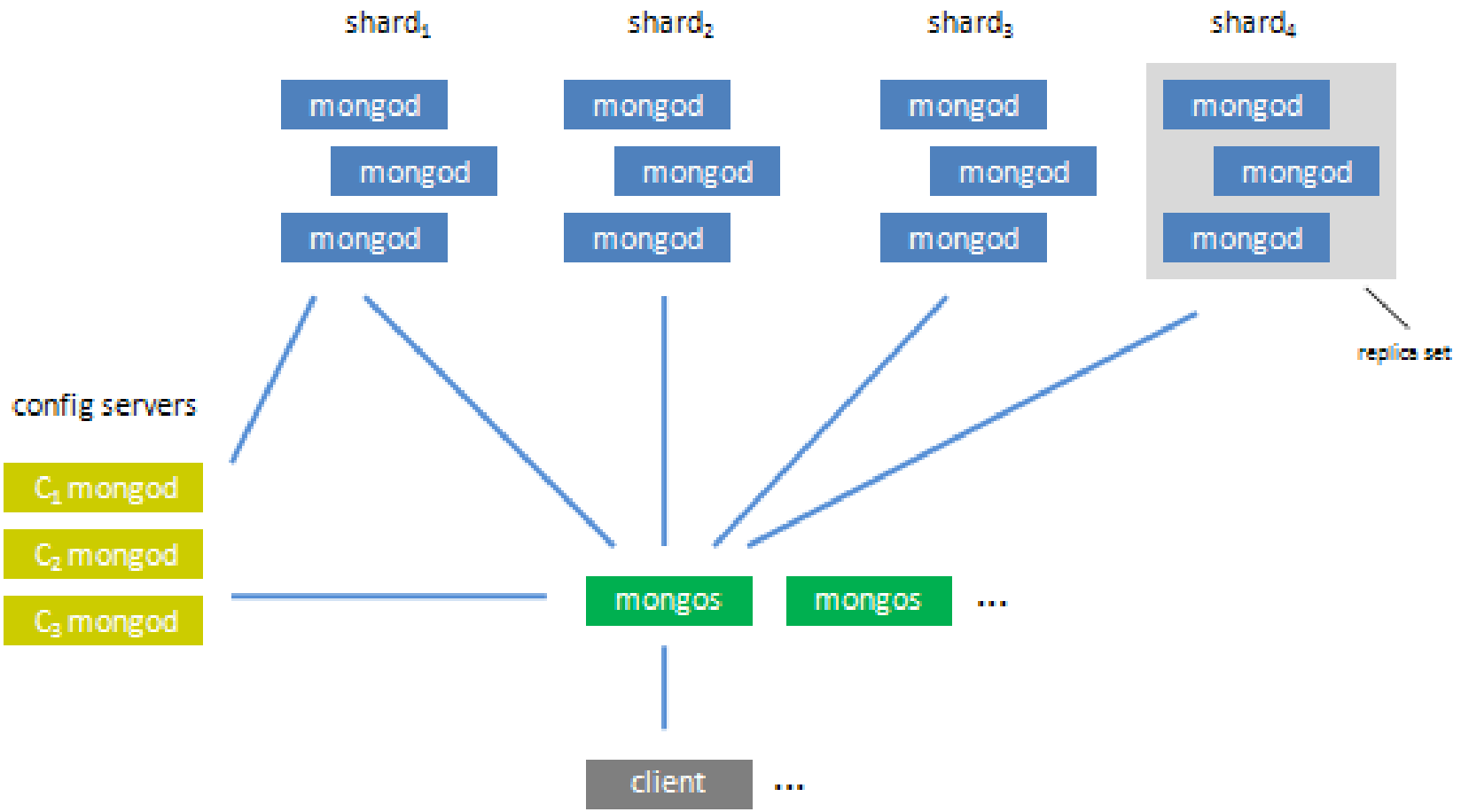

http://www.flickr.com/photos/byte/137261341/

# Skalierung

- Die meisten NoSQL-Datenbanken sind auf einen hohen Grad an **horizontaler** Skalierung ausgerichtet
- Verteilung der Daten durch *Sharding*
- *Consistent Hashing*
  - Verfahren um gleichmäßig Daten auf die Shards zu verteilen
- Klassicher Antipattern mit Shardkeys
  - Fortlaufende ID oder ein Timestamp als Shardkey
- Folge: Ein Shard wird überbelastet, gefolgt von Chunkmoves
- Foursquare Downtime

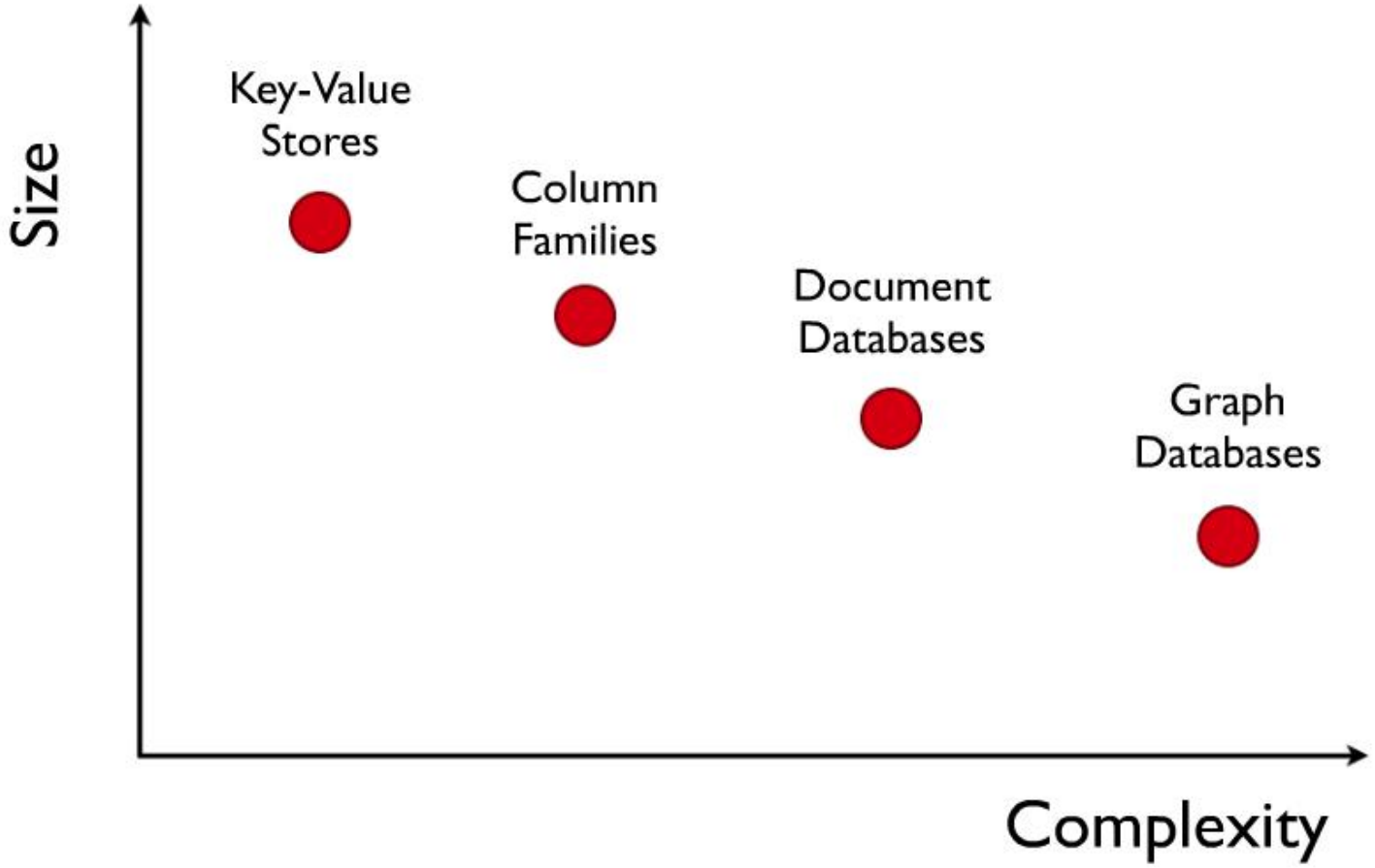
  http://blog.foursquare.com/2010/10/05/so-that-was-a-bummer/

# Skalierung am Beispiel MongoDB




Quelle: http://www.mongodb.org/display/DOCS/Sharding+Introduction

# Welche Datenbank?




Quelle: http://www.slideshare.net/emileifrem/nosql-east-a-nosql-overview-and-the-benefits-of-graph-databases

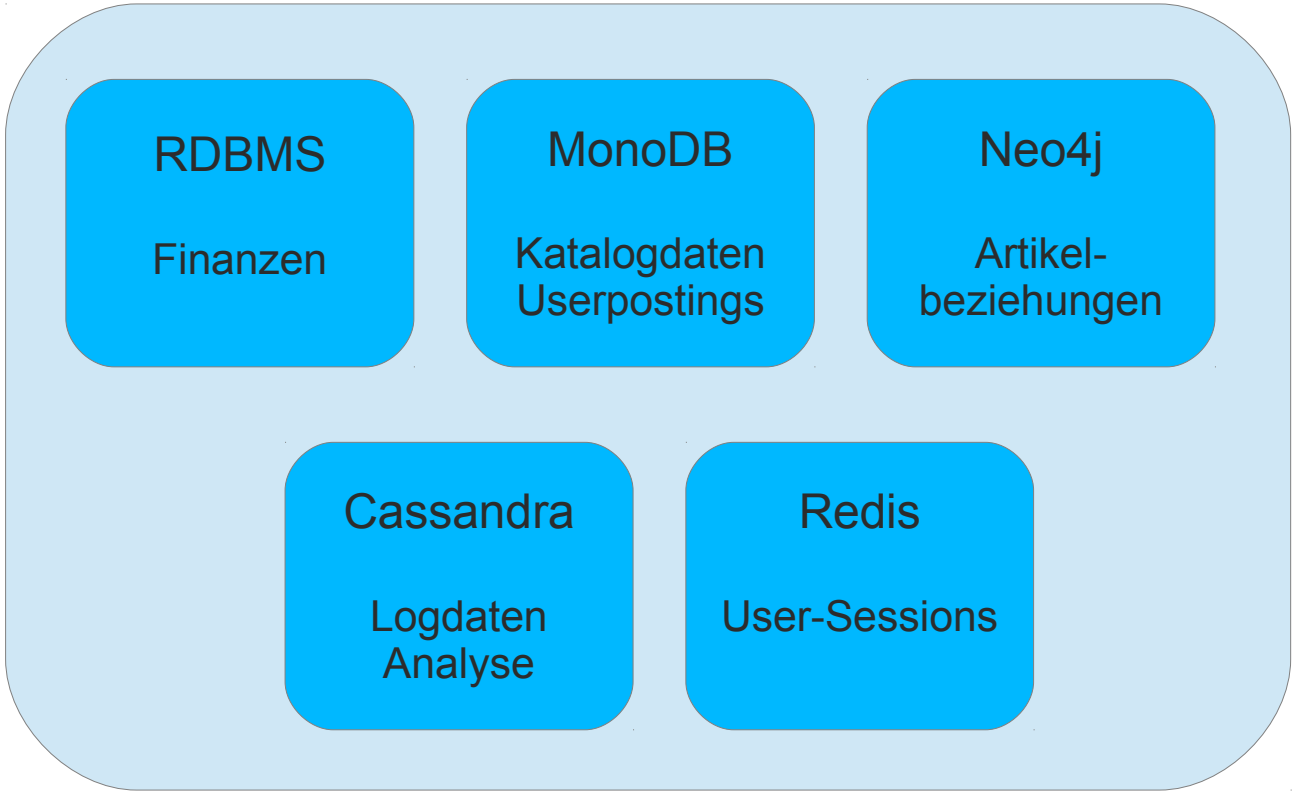
# Eine Datenbank für alles?

- Eine NoSQL-Datenbank für alle Daten einer Anwendung?
- Vielleicht für „spezielle“ Anwendungen passend, aber...

- RDBMS „on size fits all“-Falle!

- Warum nicht die Datenbanken nehmen, die zu den Use-Cases passen?
- *Polyglot Persistence* (Martin Fowler)

# Polyglot Persistence



# Spring Data

- Bisher Problem verschiedener APIs für jede Datenbank
- Mit Spring Data eine einheitliches Programmiermodell
- Unterstützt werden
  - Neo4j
  - MongoDB
  - Hadoop
  - Redis
  - REST
  - JPA




Wir suchen Sie als

- Software-Architekt (m/w)
- Projektleiter (m/w)
- Senior Software Engineer (m/w)


**jobs@adesso.de**
**www.AAAjobs.de**

adesso business. people. technology.

# Vielen Dank für Ihre Aufmerksamkeit.

**Kontakt:**

halil-cem.guersoy@adesso.de

Twitter: @hgutwit

G+ http://goo.gl/hljRS