Konzepte der Datenbanktechnologie

Prof. Dr. U. Hoffmann
FH Wedel

NOSQL-DBMS

#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Gliederung

NOSQL

- Vertikale Skalierung
- Warum NOSQL?
- Horizontale Skalierung
- CAP-Theorem
- NOSQL-Systeme
- MongoDB
- Techniken

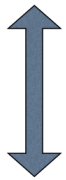
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

NOSQL

- steht für eine neue Familie von sehr gut skalierbaren Datenbanksystemen
- neuer Begriff (2009)
- ist Akronym für **N**ot **O**nly **SQL**
- Antwort auf die Anforderungen großer Internet-Sites (Facebook, Twitter, . . .)

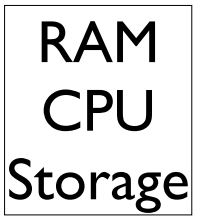
Relationale Datenbanksysteme

SQL-Datenbankensystem

- Ausrichtung auf **komplexe** Datenbanken mit
  - großen Datenmodell
  - ACID, Transaktionen, Integritätsbedingungen
  - Normalisierung (JOINs)
- Skalierung: **vertikal** 
- Software bleibt gleich, Hardware wächst

#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Vertikale Skalierung

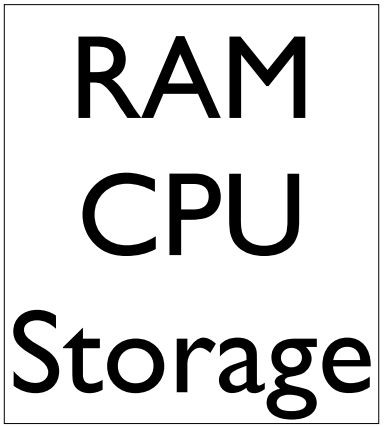


#lecture

NOSQL

Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Vertikale Skalierung

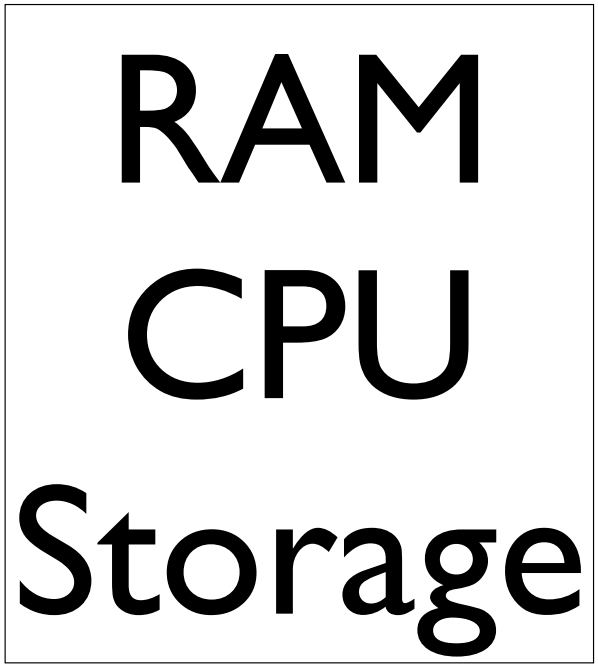


#lecture

NOSQL

Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

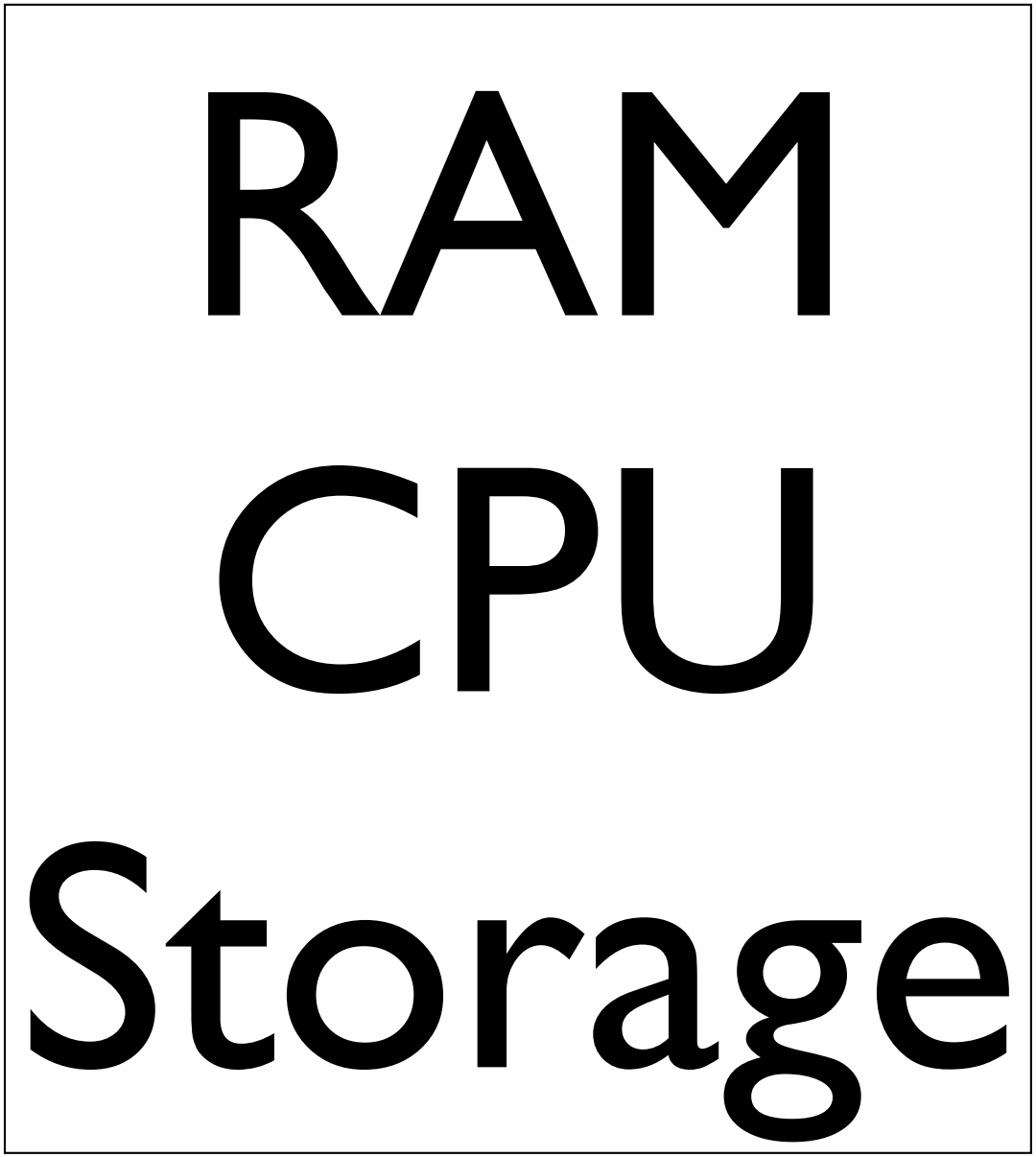
Vertikale Skalierung



#lecture

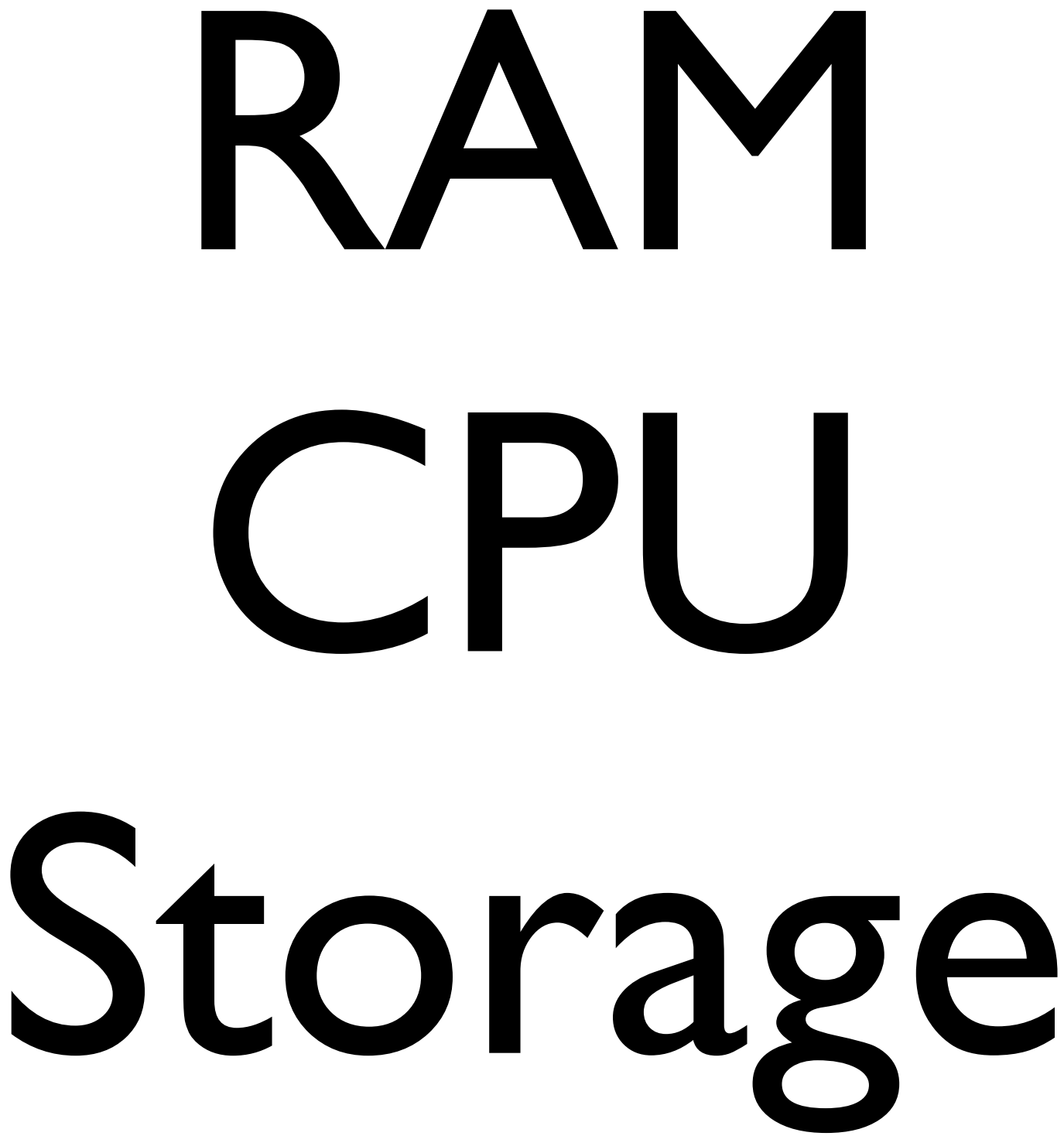
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Vertikale Skalierung



#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken



#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Vertikale Skalierung

- führt zu Groß-Systemen
- überproportional teurer
- erzeugt *single point of failure*

Warum NOSQL?

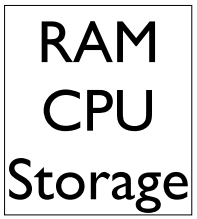
Internet-Anwendungen des 21. Jahrhunderts

- Google, Amazon, MySpace, Flickr, YouTube
- Facebook
  - 30.000 Server
  - 25 Terabyte Logdaten täglich
  - 300.000.000 Nutzer
  - 230 Ingenieure
- rasantes Wachstum
- stabile Zugriffszeiten
- hohe Verfügbarkeit
- Skalierung: **horizontal** 

#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

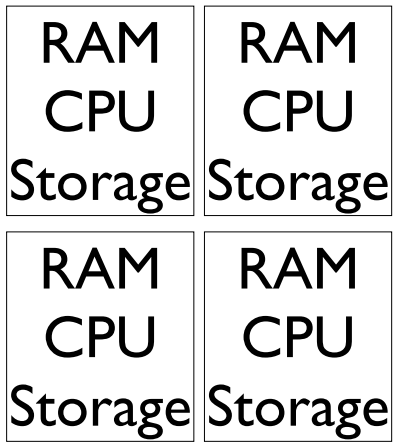
Horizontale Skalierung



#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

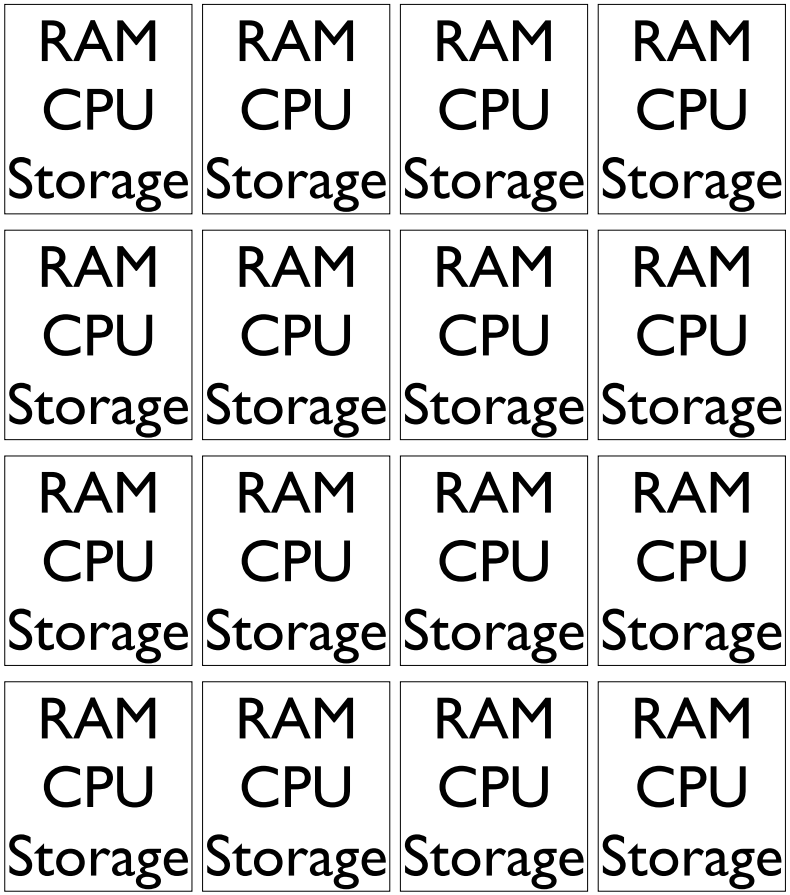
Horizontale Skalierung



#lecture

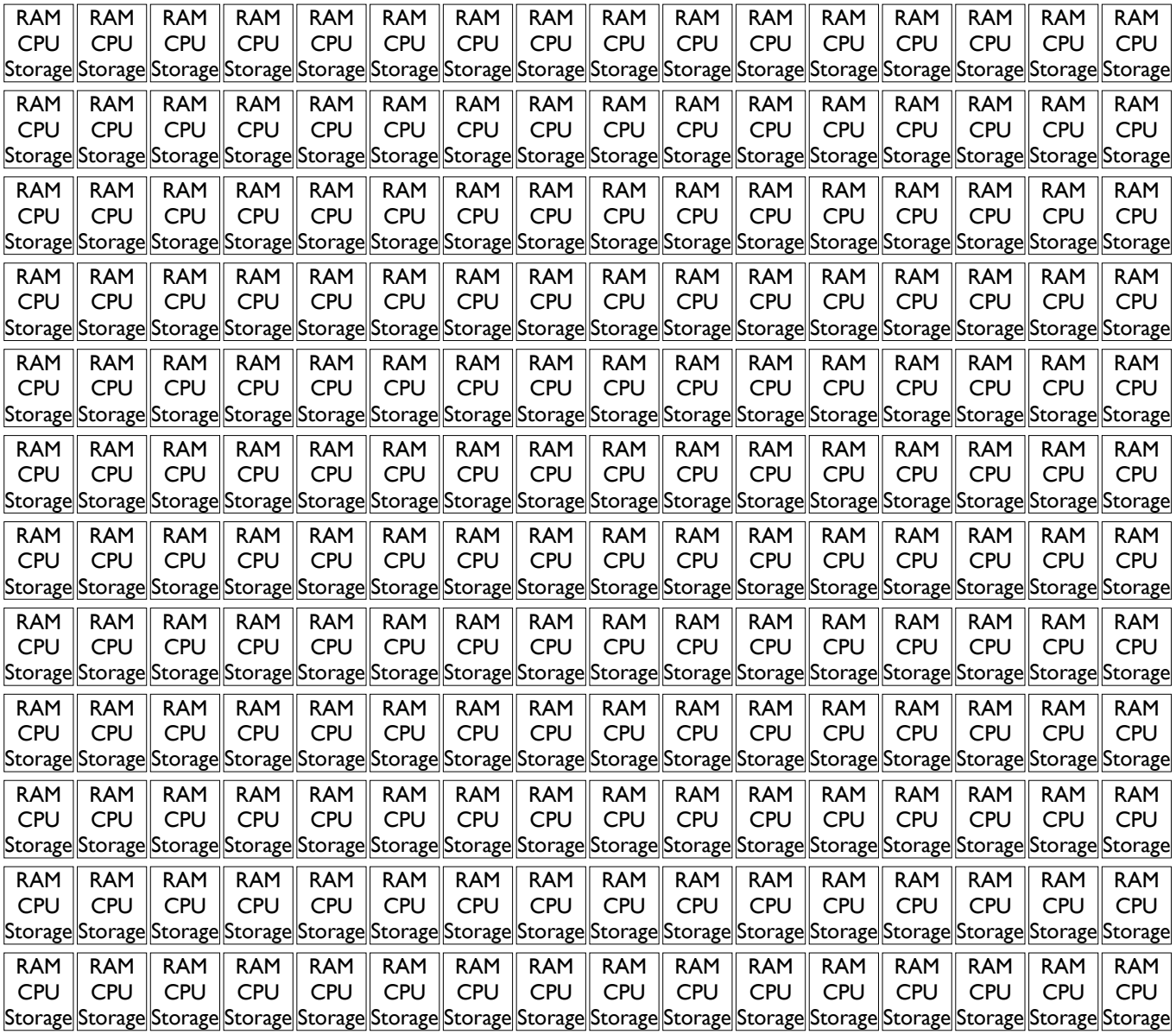
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Horizontale Skalierung



#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Horizontale Skalierung



#lecture

NOSQL

Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Horizontale Skalierung

- führt zu Verteilten Systemen
- komplexer zu beherrschen als zentrale Systeme
- erzeugt *many points of failure*, Fehler ist Normalzustand

Verfügbarkeit

Verfügbarkeitsklassen:

Klasse	Verfügbarkeit	Downtime / Jahr
2	99%	3 Tage 15 Stunden
3	99,9%	8 Stunden 45 Minuten
4	99,99%	52 Minuten
5	99,999%	5 Minuten

- 4 9en liegen im Bereich der Hardwareausfallraten
- bessere Verfügbarkeit dann nur durch Redundanz erreichbar

#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Verteilte Systeme

Verteilte Systeme sind nicht einfach:

- Kommunikation, Bandbreite, Latenz, Transportkosten
- Homogenität, Heterogenität
- Sicherheit
- Administration

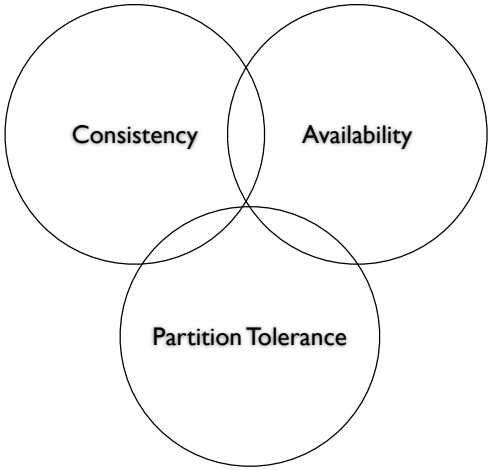
Wie kann man damit Datenbanksysteme betreiben?

#lecture

NOSQL

Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

CAP-Theorem



Theorem (CAP — Eric Brewer, PODC 2000)

In einem verteilten System können jeweils höchsten zwei der Eigenschaften:

1. *Konsistenz (Consistency)*
2. *Verfügbarkeit (Availability)*
3. *Partitionstoleranz (Partition Tolerance)*

gleichzeitig erfüllt sein.

#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Partitionstoleranz

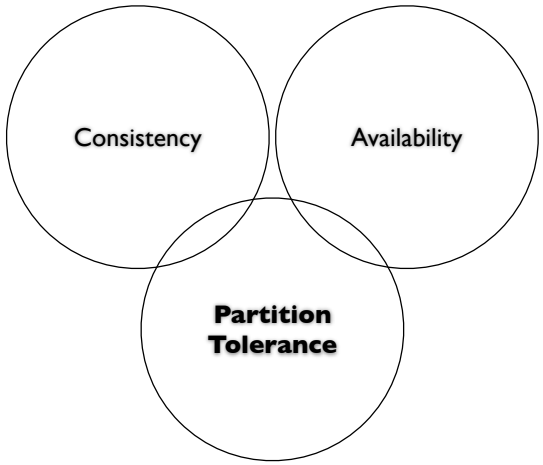
- Was geschieht mit Konsistenz, Verfügbarkeit wenn sich das verteilte System aufspaltet (partitioniert)?
- Partitionen sind isoliert:
  - keine Kommunikation zwischen Partitionen
  - Partitionen arbeiten für sich weiter
- Partitionstoleranz: Gesamtsystem kann weiterarbeiten.
- CAP:
  - Teile abschalten, um Konsistenz zu erhalten, Information dann nicht überall verfügbar, oder
  - weiterarbeiten, Verfügbarkeit gesichert, Konsistenz aufgegeben

#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

In larger distributed-scale systems, network partitions are a given; therefore, consistency and availability cannot be achieved at the same time

Werner Vogels, Amazon.com

- Damit besteht nur noch die Wahl zwischen Konsistenz und Verfügbarkeit.



#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

NOSQL-Datenbanksysteme

- Schemafrei
- keine Normalisierung, kein JOIN
- kein SQL
- Transaktionen (ACID) eingeschränkt
- weiche Konsistenz: *schließliche Konsistenz* (eventual consistency)
- einfache APIs
- horizontal skalierbar
- Replikation

Durch weiche Konsistenz besser skalierbar als bei Erzwingung harter Konsistenz

NOSQL-Systeme

- CouchDB
- MongoDB
- Redis
- Memcachedb
- Tokyo Cabinet
- Google BigTable
- Amazon Dynamo
- Apache Cassandra
- Project Voldemort
- Mnesia (Erlang)
- Hbase (Apache Hadoop)
- Hypertable
- Twitter Gizzard

Kategorien

1. Key-Value-Stores
2. Bigtable-Clones
3. Dokumentendatenbanken

#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

MongoDB

- in C implementierte Dokumentendatenbank
- Schemalos
- Bindings für viele Sprachen
- MongoDB-Shell in JavaScript, Daten extern in JSON-Repräsentation
- CRUD
- Verarbeitung mit Map/Reduce

Quelle: Karl Seguin, The Little MongoDB Book,

`http://openmymind.net/2011/3/28/The-Little-MongoDB-Book`

MongoDB

- Datenbanken,
- Collections
- Dokumente (JSON-Objekte)
- Felder (Strings, Zahlen, Arrays, Object-IDs)

Beispiel (insert)

```
db.unicorns.insert({name: 'Aurora', gender: 'f', weight: 450})
db.unicorns.insert({name: 'Leto', gender: 'm', home: 'Arrakeen', worm:
false})
```

MongoDB–Abfragen: Selectoren

- Datenbankanfragen in JSON-Form

Beispiel (find)

```
db.unicorns.find({gender: 'm', weight: {$gt: 700}})

db.unicorns.find({gender: {$ne: 'f'}, weight: {$gte: 701}})

db.unicorns.find({vampires: {$exists: false}})

db.unicorns.find({gender: 'f',
       $or: [{loves: 'apple'},
             {loves: 'orange'},
             {weight: {$lt: 500}}]})
```

MongoDB–Abfragen

- Auswahl der Ergebnis-Felder
- Sortieren,
- Ausschnitte wählen
- Zählen

Beispiel (find)

```
// Nur das Feld "name" im Ergebnis
db.unicorns.find(null, {name: 1});

// Schwerste Einhörner zuerst
db.unicorns.find().sort({weight: -1})

// erst Vampir-Name dann Vampire-Morde:
db.unicorns.find().sort({name: 1, vampires: -1})

// Ausschnitt
db.unicorns.find().sort({weight: -1}).limit(2).skip(1)

// Zählen
db.unicorns.count({vampires: {$gt: 50}})
db.unicorns.find({vampires: {$gt: 50}}).count()
```

MongoDB–Aktualisierungen

- Aktualisieren des gesamten Dokuments
- Aktualisieren von Feldern (mit `$set`)
- Weitere Operatoren u. a. `$inc`, `$push`

Beispiel (update)

```
db.unicorns.update({name: 'Roooooodles'},
                   {weight: 590})

db.unicorns.update({weight: 590},
     {$set: {name: 'Roooooodles',
             dob: new Date(1979, 7, 18, 18, 44),
             loves: ['apple'],
             gender: 'm',
             vampires: 99}})

db.unicorns.update({name: 'Roooooodles'},
                   {$set: {weight: 590}})

db.unicorns.update({name: 'Pilot'},
                   {$inc: {vampires: -2}})
```


#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

MongoDB–Aktualisierungen

- Upserts — insert oder update
- Multiple Updates — mehr als ein Dokument ändern

Beispiel (update)

```
# tut nichts, wenn Dokument nicht vorhanden
db.hits.update({page: 'unicorns'}, {$inc: {hits: 1}})

# erzeugt neues Dokument, wenn noch nicht vorhanden
db.hits.update({page: 'unicorns'}, {$inc: {hits: 1}}, true);


# Ändert nur ein Dokument
db.unicorns.update({}, {$set: {vaccinated: true }});

# Ändert alle Dokumente
db.unicorns.update({}, {$set: {vaccinated: true }}, false, true);
```

MongoDB–Verarbeitung: MapReduce

- 2 Funktionen, Map und Reduce, in JavaScript
- map: Relevante Daten extrahieren/berechnen

Beispiel (map)

```
function() {
  var key = {
    resource: this.resource,
    year: this.date.getFullYear(),
    month: this.date.getMonth(),
    day: this.date.getDate()
  };
  emit(key, {count: 1});
}

{resource: 'index', year: 2010, month: 0, day: 20}
  => [{count: 1}, {count: 1}, {count:1}]
{resource: 'about', year: 2010, month: 0, day: 20}
  => [{count: 1}]
{resource: 'about', year: 2010, month: 0, day: 21}
  => [{count: 1}, {count: 1},
{count:1}]
{resource: 'index', year: 2010, month: 0, day: 21}
  => [{count: 1}, {count: 1}]
```

MongoDB–Verarbeitung: MapReduce

- 2 Funktionen, Map und Reduce, in JavaScript
- reduce: Extrahierte Daten zusammenfassen

Beispiel (reduce)

```
function(key, values) {
    var sum = 0;
    values.forEach(function(value) {
    sum += value['count'];
  });
  return {count: sum};
};
```

```
{resource: 'index', year: 2010, month: 0, day: 20} => {count: 3}
{resource: 'about', year: 2010, month: 0, day: 20} => {count: 1}
{resource: 'about', year: 2010, month: 0, day: 21} => {count: 3}
{resource: 'index', year: 2010, month: 0, day: 21} => {count: 2}
{resource: 'index', year: 2010, month: 0, day: 22} => {count: 1}
```

Techniken in NOSQL-Datenbanksystemen

- Verteilte Hash-Tabellen
  - Partitionierung des Schlüsselraums
  - Abbildung auf Netzwerk (Overlay Network)
  - Abbildung von Daten und Servern: Consistent Hashing
- Vektor-Uhren
  - Verallgemeinerung der logischen Uhren nach Lamport
  - kausale Abhängigkeiten
- Redundanz, Replikation (Sloppy Quorum, Hinted Handoff)
- Indexstrukturen (z. B. Merkle Trees)
- Abstimm/Koordinationsverfahren (z. B. Paxos, Gossip, . . .)

#lecture
NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken

Ausblick

NOSQL

- Vertikale Skalierung
- Warum NOSQL?
- Horizontale Skalierung
- CAP-Theorem
- NOSQL-Systeme
- MongoDB
- Techniken

- Fragen?

#lecture

NOSQL
Vertikale Skalierung
Warum NOSQL?
Horizontale Skalierung
CAP-Theorem
NOSQL-Systeme
MongoDB
Techniken