NoSQL und MongoDB

Übersicht

- NoSQL - Überblick
- MongoDB
  - Installation und Benutzung
  - Eigenschaften: In-place updates etc.
  - Redundanz durch Verteilung
  - Skalierung durch (Auto-)sharding

NoSQL I

Unter dem Schlagwort “NoSQL” versteht man DB-systeme die meist

- vom relationalen Modell abweichen, insbesondere join-operationen vermeiden
- Vorteile bei Verteilung im Netz haben
- Bei ACID-Eigenschaften Kompromisse eingehen

Oft motiviert von großen Anwendungen im Netzwerk

NoSQL II

Grundeigenschaften von Datenbanken

- A *atomicity*
- C *consistency*
- I *isolation*
- D *durability*

SQL-Systeme bieten alles davon – ein Grund für den Erfolg, aber es hat einen Preis, Verteilung ist schwierig

CAP - Theorem

Mehr als zwei von den dreien geht in verteilten Systemen nicht:

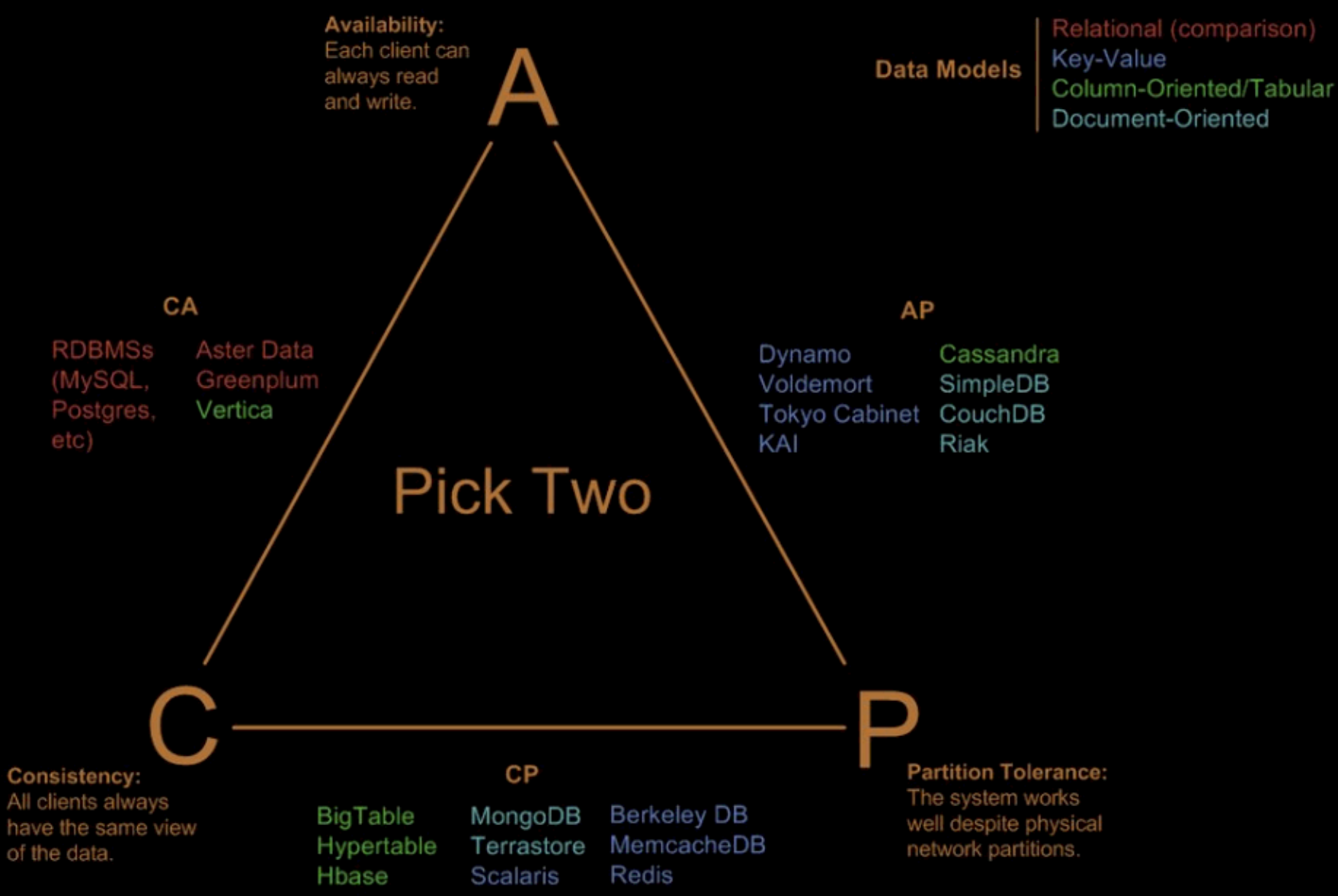
C	Consistency	Alle sehen das selbe
A	Availability	Es läuft auch wenn Knoten fehlen
P	Partition Tolerance	System läuft auch bei Netzfehlern

Datenbanken werden gelegentlich danach klassifiziert welches Paar sie implementieren, z.B. RDBMS (MySQL): CA

Die strikte Klassifikation der Systeme ist aber dubios, da einzelne Aktionen im selben System anderen Regeln folgen können.

Oft Kompromisse wie ‘gebe ein wenig C auf für etwas mehr A’

Visual Guide to NoSQL Systems



NoSQL Datenmodelle

Es gibt jede Menge:

- Dokument
  - JSON
  - XML
- Key-Value
- Tabellen
- Graph

```
{
        "messwerte" : {
                "innen" : 18,
                "aussen" : 2
        },
        "tags" : [
                "kalt",
                "regen"
        ],
        "zeit" : ”14:00:00"
}
```

MongoDB - start

Fertige Pakete für Linux, Mac, Windows etc.

Der erste Query:

```
[cbook3:~] mnbokaem% mongod --dbpath /tmp -fork
[cbook3:~] mnbokaem% mongo
> use mydb
> bsp = { messwerte: { innen: 18, aussen: 2 },
    tags: [ "kalt", "regen"], zeit: new Date() }
> db.test.save(bsp)
> db.test.find()
{ "_id" : ObjectId("4ce8934f36d467451f367753"),
  "messwerte" : { "innen" : 18, "aussen" : 2 },
  "tags" : [ "kalt", "regen" ],
  "zeit" : ”Fri Nov 19 2010 14:27:26 GMT+0100 (CET)”
}
```

Queries I

Die Struktur von mongo-queries ist SQL ähnlich.

Es gibt cursors, limits, sort, suche, indizes usw.

Datenbank -> Collection -> Dokument

SQL	Mongo
SELECT a,b FROM users WHERE age=33	db.users.find({age:33}, {a:1,b:1})
SELECT * FROM users LIMIT 10 SKIP 20	db.users.find().limit(10).skip(20)
SELECT COUNT(*) FROM users where AGE > 30	db.users.find({age: {'$gt': 30}}).count()
CREATE INDEX myindexname ON users(name)	db.users.ensureIndex({name:1})

Queries II

- Mongo arbeitet mit ‘binary JSON’ = BSON
- Operatoren für JSON: $push, $pop, $in, $addToSet etc.
- In Dokumente hineinreichen: { messwerte.innen: { $lt : 0 } }
- Update operatoren:
  db.users.update({ _id : ... }, { $inc : { age : 1 } }
- Atomares Suchen und Ändern:

```
job = db.jobs.findAndModify( {
        query: {inprogress: false},   sort : {priority:-1},
        update: {$set: {inprogress: true, started: new Date()}});
```

Queries III

- Neben den ‘normalen’ Queries kann server-seitiger Javascript Code ausgeführt
- MapReduce, Group und $where
- JavaScript in Mongo ist zur Zeit single-threaded und vergleichsweise langsam, wenn normale Queries reichen sollte man die einsetzen

Replication

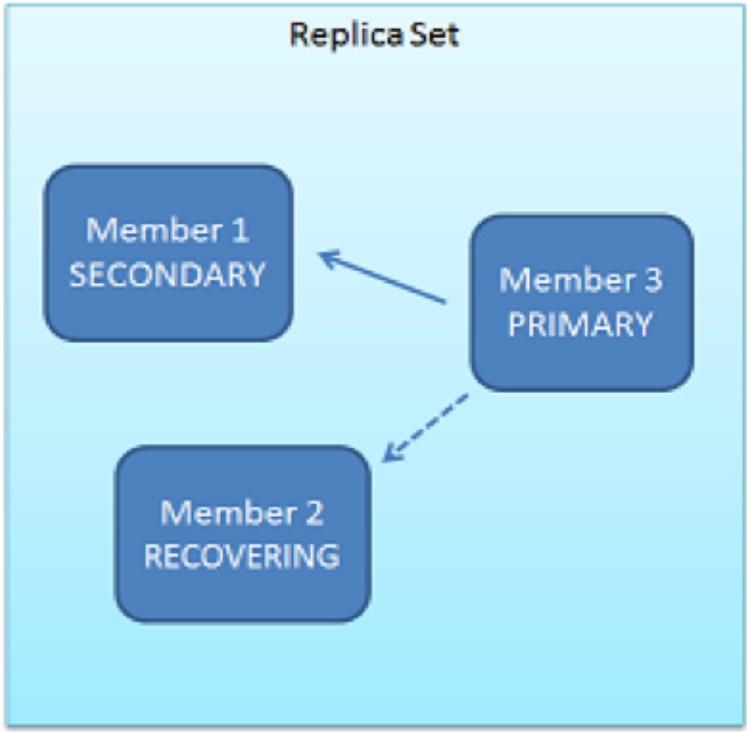
“Replica Sets” wählen einen PRIMARY node, alle anderen sind read-only SECONDARIES

Wenn der PRIMARY ausfällt wird übernimmt ein SECONDARY automatisch die Rolle

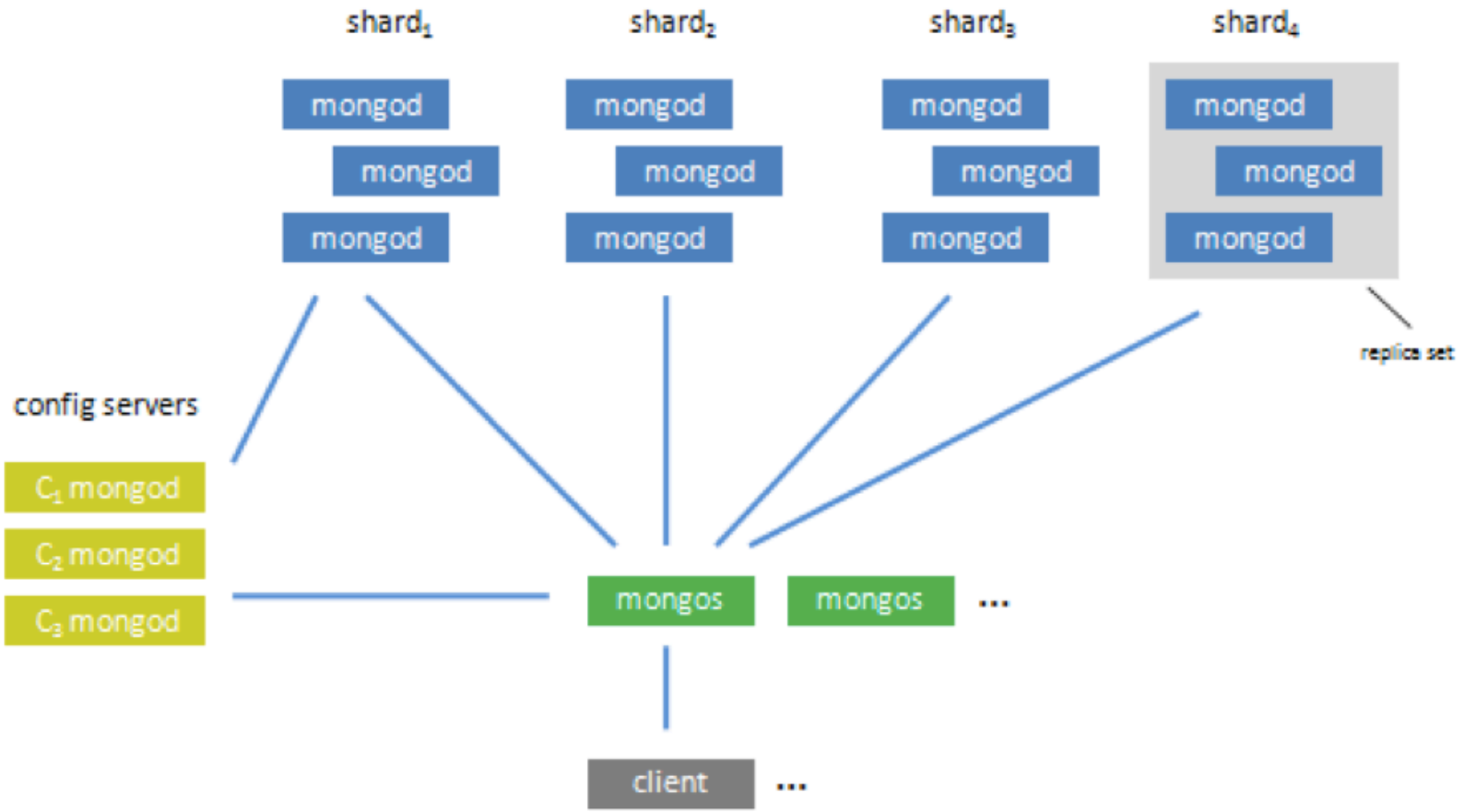
ARBITER nodes halten keine Daten, stimmen aber mit ab

Clients finden den MASTER nach einem Verbindungsabbruch selbst wieder

Clients können auch von secondaries lesen, Flag im Query: ‘slaveOK’.



Sharding I


shard1
shard2
shard3
shard4
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
replica set
config servers
C1 mongod
C2 mongod
C3 mongod
mongos
mongos
...
client
...

Sharding II

- Verteilt die Dokumente einer Collection basierend auf einem ‘shard key’ – einem beliebigen Attribut (kann _id sein, muss aber nicht)
- Bei queries mit shard key kann mongos direkt den passenden ‘shard’ ansprechen, ansonsten geht der query an alle.
- Verteilte Queries laufen parallel auf allen shards
- Shards haben Bereiche des ‘shard keys’
- Daten werden automatisch zwischen shards ‘balanciert’

In-Place

Mongo blendet alle Daten mit mmap() in den Speicher

- 64-bit Addressraum ist essentiell
- Updates werden direkt ‘im Objekt gemacht’, wenn es nicht wächst
- Geänderte Daten gehen alle 60s (default) auf disk oder per Kommando oder wenn der Speicher knapp wird

⟹Ein einzelner mongod ist nicht ‘durable’, ein Systemabsturz kann eine inkonsistente DB hinterlassen !

- Lösung: Replica Sets
- In Planung: Optionales binlog

Features I

- GridFS
  - Files in MongoDB – erlaubt es viele und grosse Files repliziert zu halten
  - Kein FS-Treiber, Zugriff über Kommando / API
- Geospatial
  - Neben den üblichen Datentypen gibt es ‘Koordinaten’
  - Indizierung und Suche auf ‘Nähe’ ist möglich

Features II

- Capped Collections
  - Feste Größe der Collection
  - Erlauben nur kein delete, eingeschränkte updates
  - Schreibt sequentiell in den Speicher, roll-over wenn vol

$\Rightarrow$ Eigentlich als internes Feature für replication gebaut aber extrem nützlich wenn es um log oder queue-artige Daten geht

Vielen Dank!

Gibt es noch Fragen ?

Home: http://www.mongodb.org/

Interessantes zu CAP:
http://dbmsmusings.blogspot.com/2010/04/problems-with-cap-and-yahoos-little.html