MongoDB

...in the NoSQL and SQL world.

Horst Rechner horst.rechner@fokus.fraunhofer.de
Berlin, 2012-05-15

MongoDB
...in the NoSQL and SQL world.

- NoSQL – What? – Why? - How?
- Say goodbye to... ACID, hello BASE
- You cannot have everything... CAP
- Types of NoSQL databases
- mongoDB highlights
- Some examples in SQL and mongo‘s query language

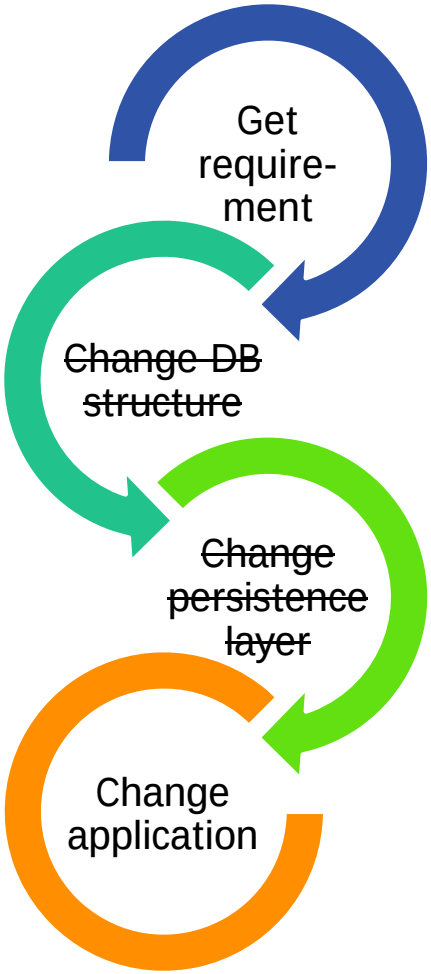
if there is time...

- Map-Reduce example with MongoDB

What is NoSQL

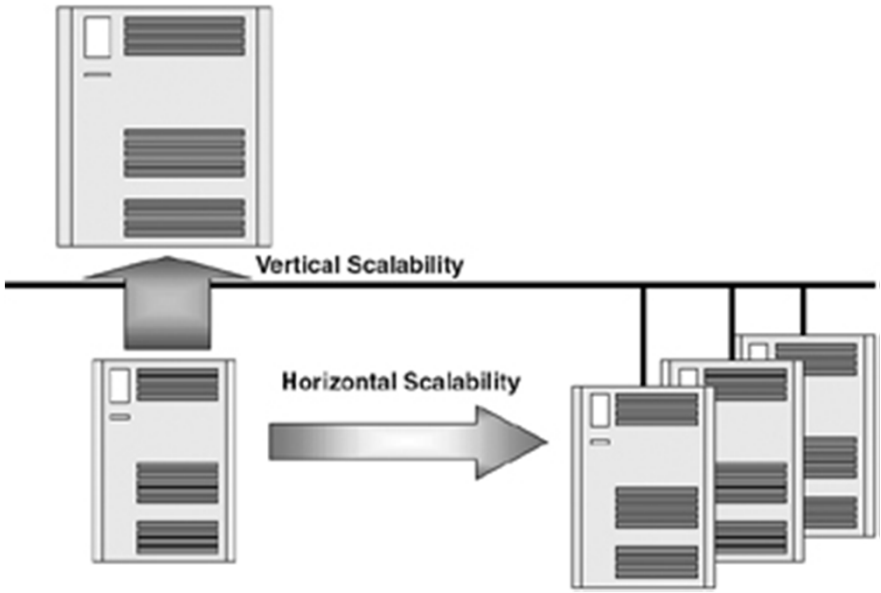
- 2009 Johan Oskarsson from last.fm coined the term NoSQL = Not Only SQL
- It's not a DBMS, but a class of them or even a movement.

- Not necessarily using/giving...
  - SQL as query language
  - a fixed schema (vs. agile) →
  - ACID guarantees (vs. scalability)
- Adding
  - easy distribution (machines / cloud)
  - query methods for big data (many times map-reduce)



Why NoSQL – “throughput first!”

- RDBMS are tuned towards
  - frequent but small reads / writes
  - large transactions with rare writes
- NoSQL databases are tuned towards
  - heavy read / write workload
  - so we need
    - good horizontal scalability (vs. vertical scalability)
  - How to do that?



1) Vallath M. - Oracle Real Application Clusters

Say goodbye to...

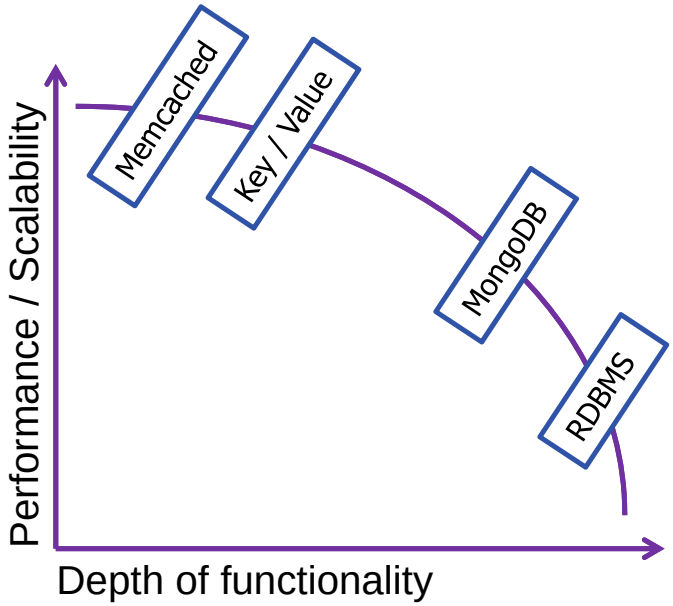
- Relational **joins**
  - vs. adding indexes, materialized views, table partitions, denormalization

  →handle them in your code
- **Normalization**
  - vs. modification anomalies

  → be careful

  → storage is cheap
- Complex multi-statement **transactions** with rollbacks

  → handle them in your code
  (atomic transactions are still there)



(this is true for MongoDB
there are exceptions to the rule - transactions e.g. Redis)

1) Max Schireson – Mongo DB Berlin Keynote. 2012

ACID → BASE

- Move away from
  ACID = **A**tomicy **C**onsistency **I**solation **D**urability
  → if you are **not** running a bank

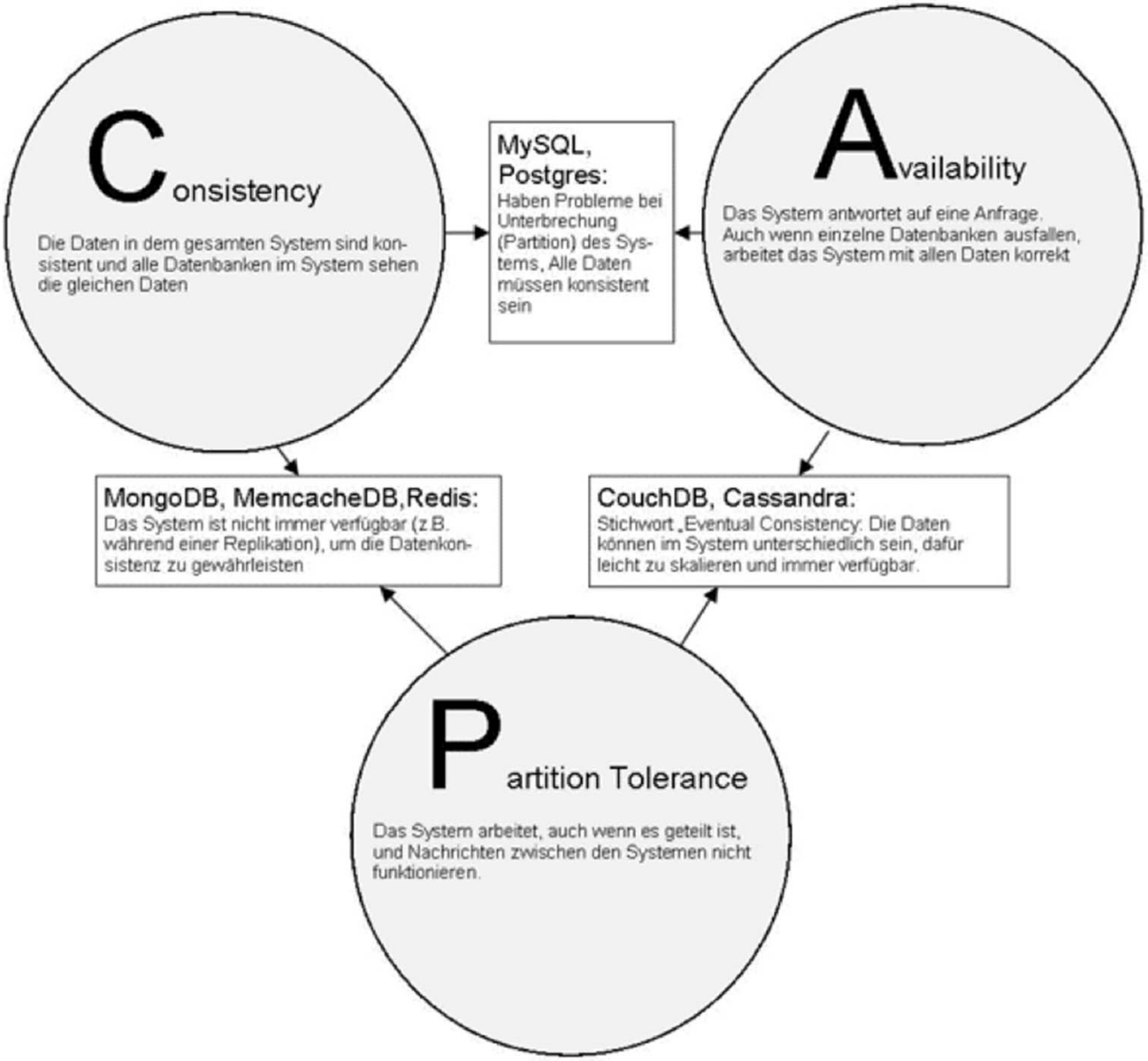
- towards
  BASE = **B**asically **A**vailable **S**oft-State **E**ventual Consistency
  →if you want to run twitter

Why not have it all? Consistency and horizontal scaling?

→Eric Brewer's CAP Theorem (1)

1) Dr. Eric A. Brewer - Principles of Distributed Computing Keynote, 2000
2) BASE Theorem, Eric Brewer, 1997

The CAP Theorem / Triangle



1) Oliver Kurowski – No SQL Einführung, 2012
2) http://blog.nahurst.com/visual-guide-to-nosql-systems, 2010

Types of NoSQL Databases

- **Document-oriented** (e.g. CouchDB, MongoDB, Lotus Notes – the 80's!)
  - Semi-structured documents encoded in XML, JSON, BSON (Queries with cURL)
  - Schemas can be used
  - Associations between data has to be coded manually
- **Key-Value stores** (e.g. Riak, Redis, MemcacheDB, Associative arrays – the 60's!)
  - Think giant distributed hashtable, lists or sets (access only via key)
  - Providing persistence & replication (Redis)
  - All operations in RAM (Redis) or on disk
- **Column-oriented** (e.g. Cassandra, RAPID – the 60's)
  - "next dataset is next row same column"
  - SQL as query language
  - Good for analysis of big data when many fields are not used
- **Graph-database** (e.g. Neo4J, ??? – the ??'s)
  - Store data in a graph (nodes, edges and their properties)

```
1,2,3;
Smith,Jones,Johnson;
Joe,Mary,Cathy;
40000,50000,44000;
```

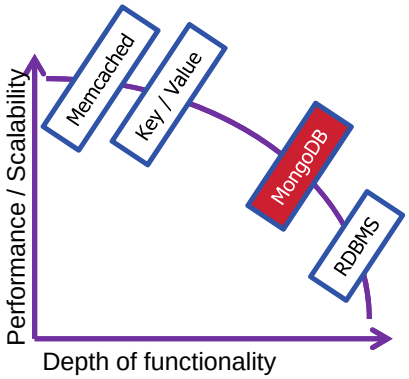
→ So is NoSQL new?

1) http://en.wikipedia.org/wiki/Column-oriented_DBMS

MongoDB - highlights

Things we already know...

- Document-oriented (Binary JSON)
- Consistency and partition tolerant, but not always available.
- No Relational Joins / No multi-statement transactions with rollback



Things we like from the DBMS world...

- Indexes / Geo-Indexes
- **Ad-hoc (without special indices) queries (right now)**
- Atomic transactions (inc, set, findAndModify...)
- Clients for myriad of programming languages (including POJO mappers)

Things you need for scaling...

- Sharding for horizontal scaling & replica sets for failover / high availability (and easy to set those up)
- **Map & Reduce queries (if there is time)** & hadoop integration

1) http://stackoverflow.com/questions/7339374/nosql-what-does-it-mean-for-mongodb-or-bigtable-to-not-always-be-available Discussion on CP-A

2) http://www.mongodb.org/display/DOCS/Java+Language+Center

Ad-hoc queries in SQL and „MongoQL“

- `use simtd_versuche`
- Don’t define any schemas.
- `db.fahrer.insert(`
  `{name: 'Horst Rechner',`
  `telefon: '030-3463-7276', …});`
- `db.fahrer.find(`
  `{name: 'Horst Rec`
  `{telefon:1, _id:0`
- `db.fahrer.find({n`
  `Rechner'}, {_id:1`

  `db.gruppestation.find({fahrer:`
  `ObjectId("4fb16727d56ba17b066d25f8`
  `")}, {station_id:1, _id:0});`

If you use arrays, they are treated as first class objects.

['030...', '0177...']

- `CREATE DATABASE simtd_versuche`
- Create tables with their fixed schemas...
- `INSERT INTO  fahrer(id, name,`
  `telefon, …)`
  `VALUES (1, 'Horst Rechner', ‘030-`
  `3463-7276‘, …);`
- `SELECT telefon`
  `hrer`
  `ame = 'Horst Rechner';`

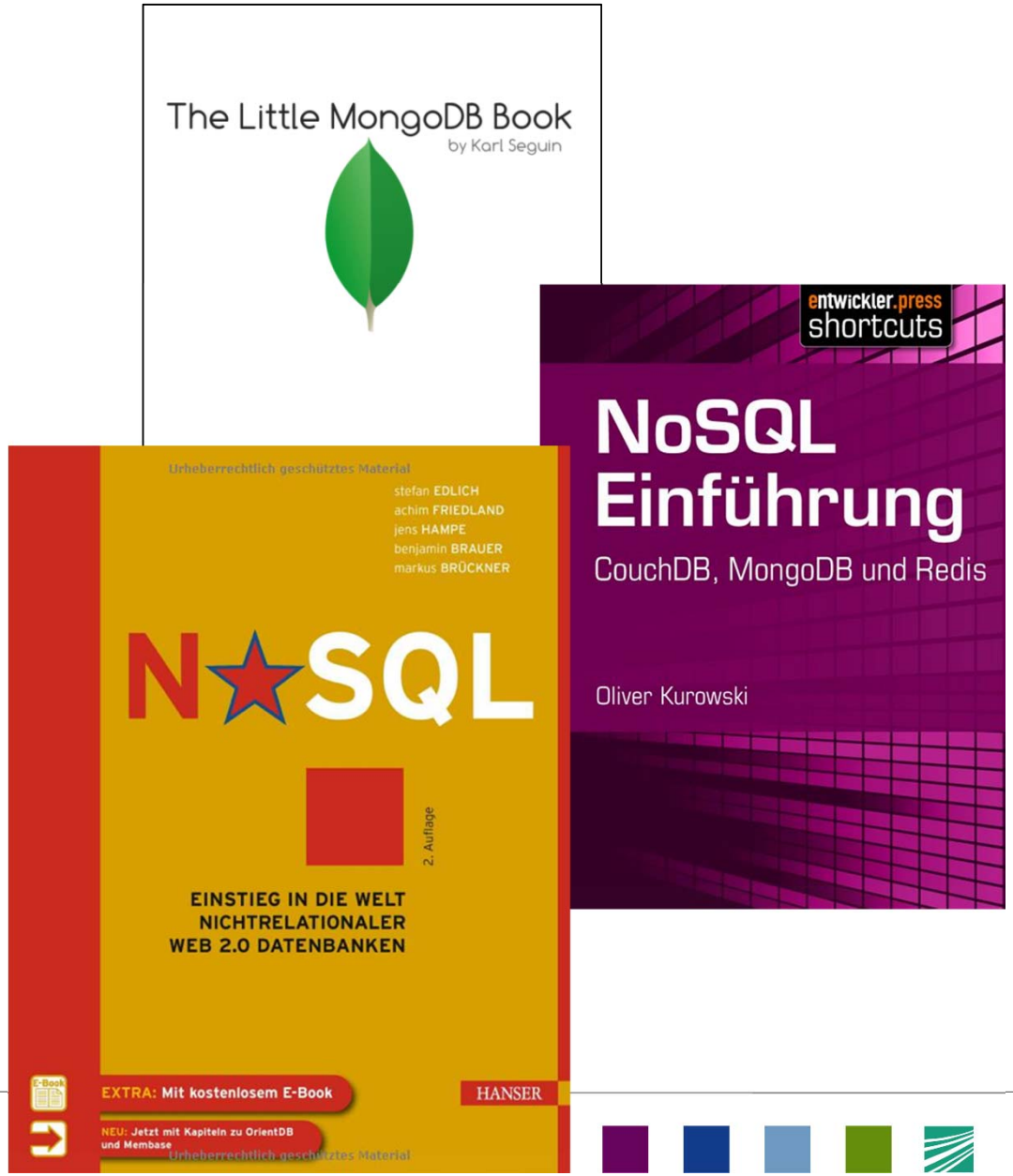
  `gruppestation.station_id`
  `uppestation`
  `LEFT JOIN fahrer`
  `ON gruppestation.fahrer_id =`
  `fahrer.id`
  `WHERE fahrer.name = 'Horst`
  `Rechner';`

1) http://drorbr.blogspot.de/2010/02/migrating-springhibernate-application.html
Migrating a Spring/Hibernate application to MongoDB

Thank you for your attention.

- **Horst Rechner**
  horst.rechner@fokus.fraunhofer.de

Further reading...



1) Picture credit: Simran Jindal's blog

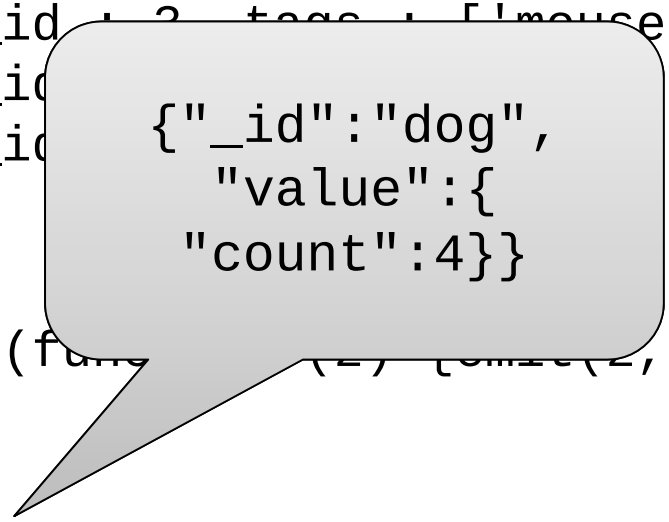
Backup slides

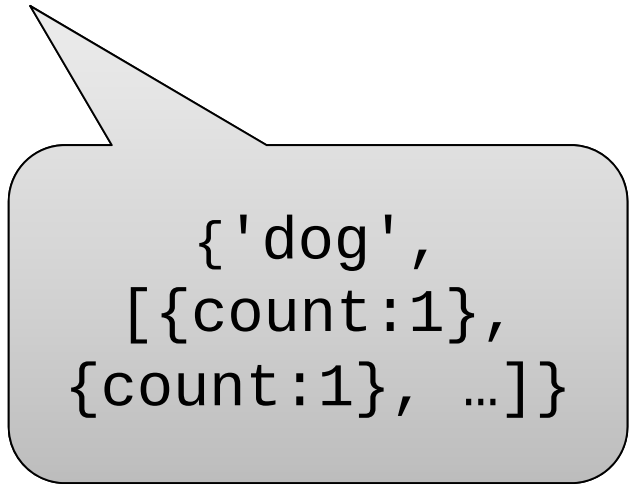
Map-Reduce Example - Definition

- ```
  db.things.insert( { _id : 1, tags : ['dog', 'cat'] } );
  db.things.insert( { _id : 2, tags : ['cat'] } );
  db.things.insert( { _id : 3, tags : ['mouse', 'cat', 'dog'] } );
  db.things.insert( { _id : 4, tags : ['dog'] } );
  ```

- ```
  m = function () {
      this.tags.forEach(function(z) {emit(z, {count:1});});
  }
  ```

- ```
  r = function (key, values) {
      var total = 0;
      for (var i = 0; i < values.length; i++) {
          total += values[i].count;
      }
      return {count:total};
  }
  ```





1) http://www.mongodb.org/display/DOCS/MapReduce
  ```

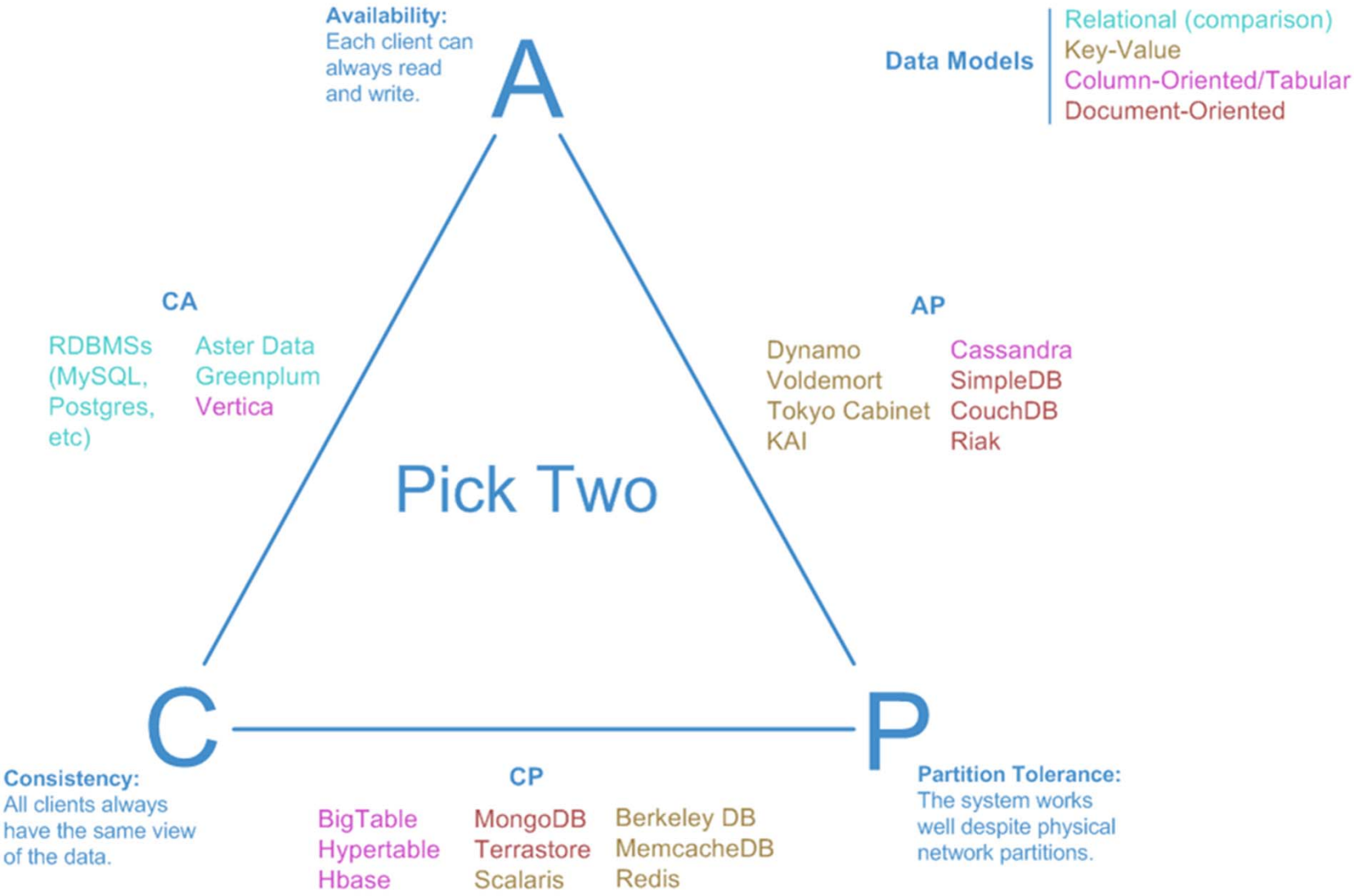
Map-Reduce Example - Execution

```
db.things.insert( { _id : 1, tags : ['dog', 'cat'] } );
db.things.insert( { _id : 2, tags : ['cat'] } );
db.things.insert( { _id : 3, tags : ['mouse', 'cat', 'dog'] } );
db.things.insert( { _id : 4, tags : []  } );
db.things.insert( { _id : 5, tags : ['dog', 'dog'] } );
```

```
res = db.things.mapReduce(m, r, { out : "myoutput" } );
{"result" : "myoutput",
 "timeMillis" : 4,
 "counts" : {
     "input" : 5,
     "emit" : 8,
     "reduce" : 2,
     "output" : 3
 },  "ok" : 1,}
```

```
db.myoutput.find()
{ "_id" : "cat", "value" : { "count" : 3 } }
{ "_id" : "dog", "value" : { "count" : 4 } }
{ "_id" : "mouse", "value" : { "count" : 1 } }
```


Visual Guide to NoSQL Systems



1) http://blog.nahurst.com/visual-guide-to-nosql-systems