Key-Value Datenspeicher

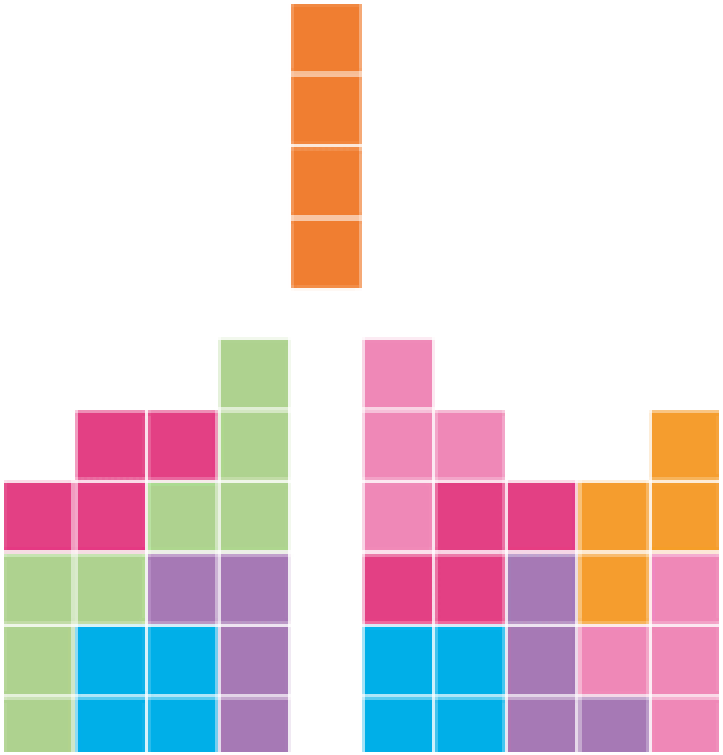
NoSQL für Anwendungen

Hochschule Mannheim
Fakultät für Informatik
Cluster Grid Computing Seminar
SS 2012

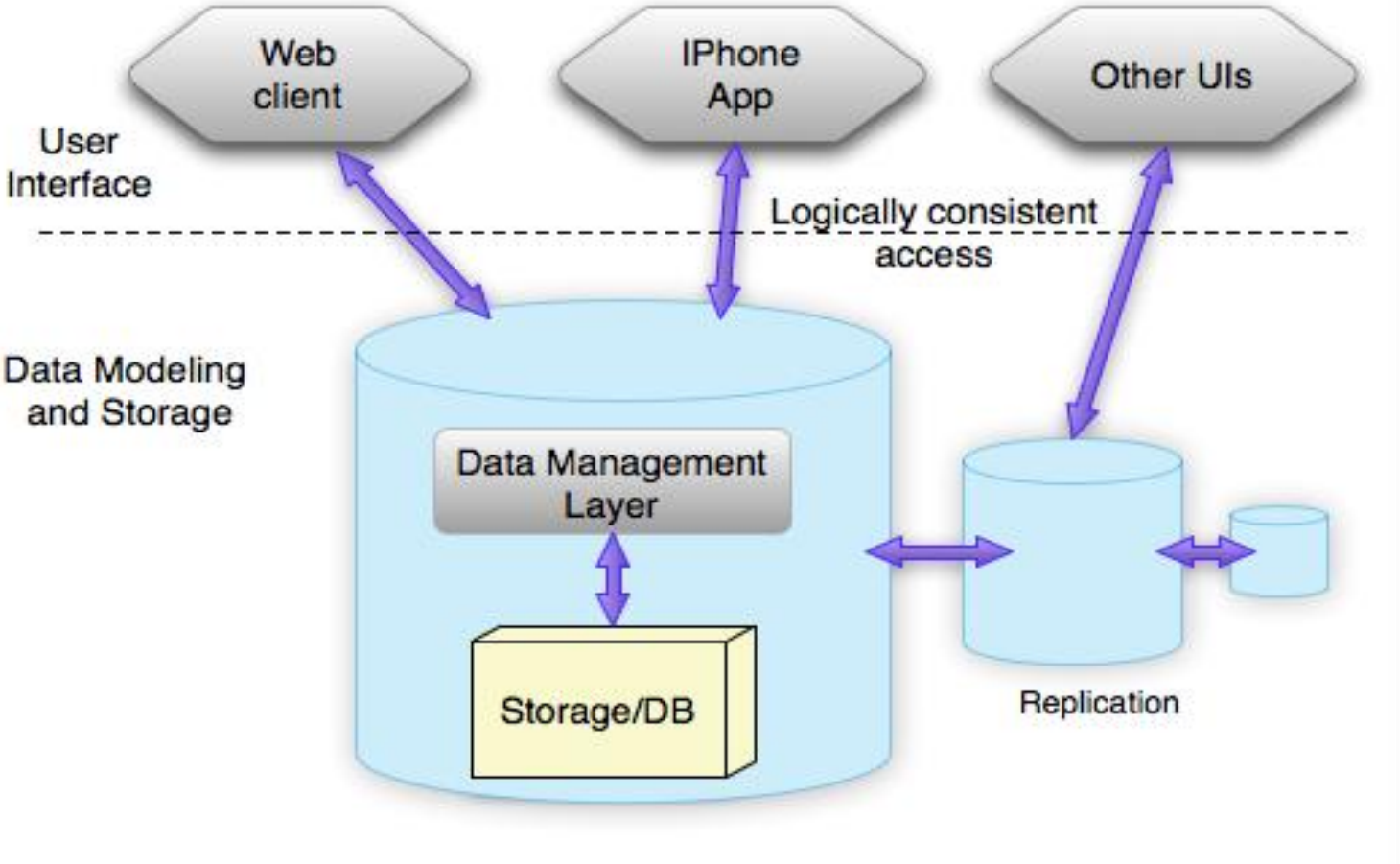
Lemmy Tauer (729400)
lemmy.coldLemonade.tauer@gmail.com

-> Inhalt

- NoSQL
- CAP / ACID / Kompromisse
- Key-Value Kandidaten
- Key-Value als Zwischenspeicher
  - Memcached / MemcachedDB
- Key-Value Datenstruktur
- Key-Value Datenspeicher
  - Redis
  - Redis Client
- Key-Value mit Big Data
  - Cassandra
  - Datenmodel
  - Cassandra Query Language
  - Kommandozeilen Schnittstelle
- Resumee
- Quellen



-> Datenzugriff und Speicher



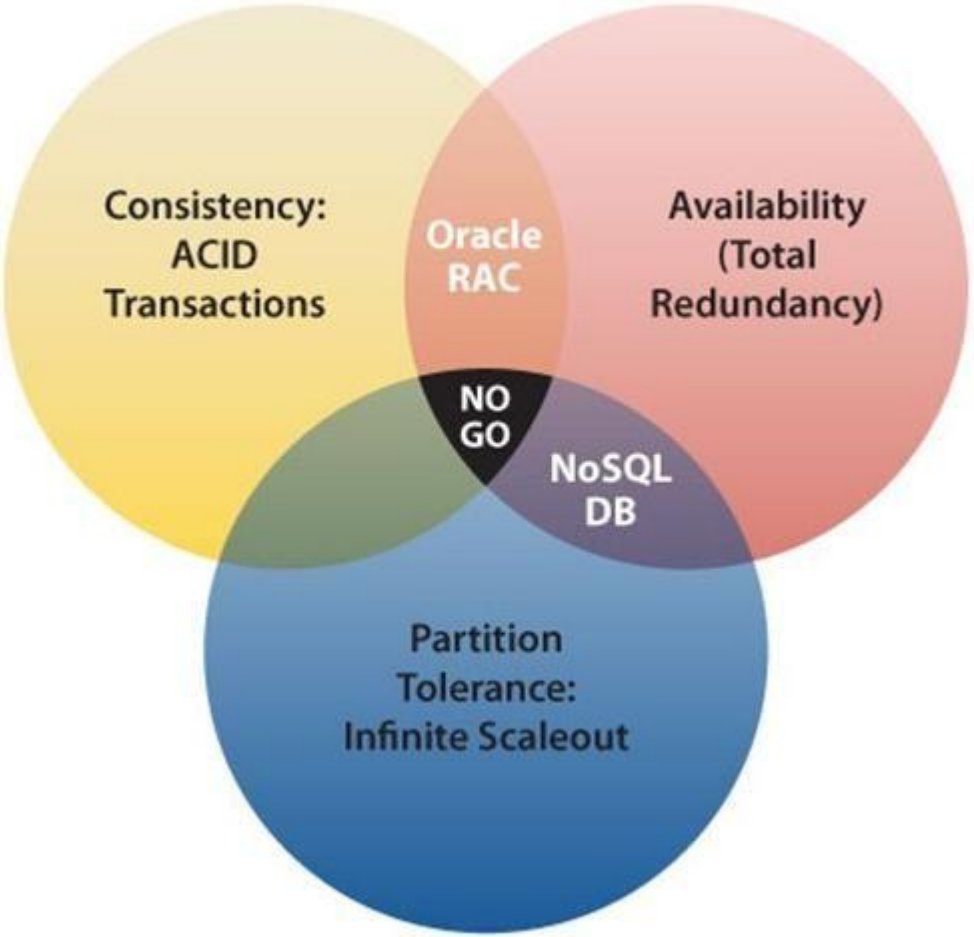
-> Was ist noSQL ?

- Verteilte Speicherung
  - Write-Once Read-Many
- Kein (hierarchisches) Datenschema
  - Weniger komplexe Relationen
  - Weniger dynamische Abfragen
- Verzögerte Globale Konsistenz, schnelle Verfügbarkeit
  - Keine ACID Semantik (Transaktionen)
- BASE Semantik (**B**asically **A**vailable **S**oft-state **E**ventually-consistent)
  - Flexible Konsistenzbedingung

- Low-Level Datenmodelierung
- Aufwand und Geschwindigkeit
- Verzicht auf Overhead der SQL Architektur
- Hochskalierfähiger Verarbeitung von Big Data (MapReduce)
  - Bei zunehmend komplexer Datenhaltung

-> CAP und ACID

- *Eric Brewer 2000*



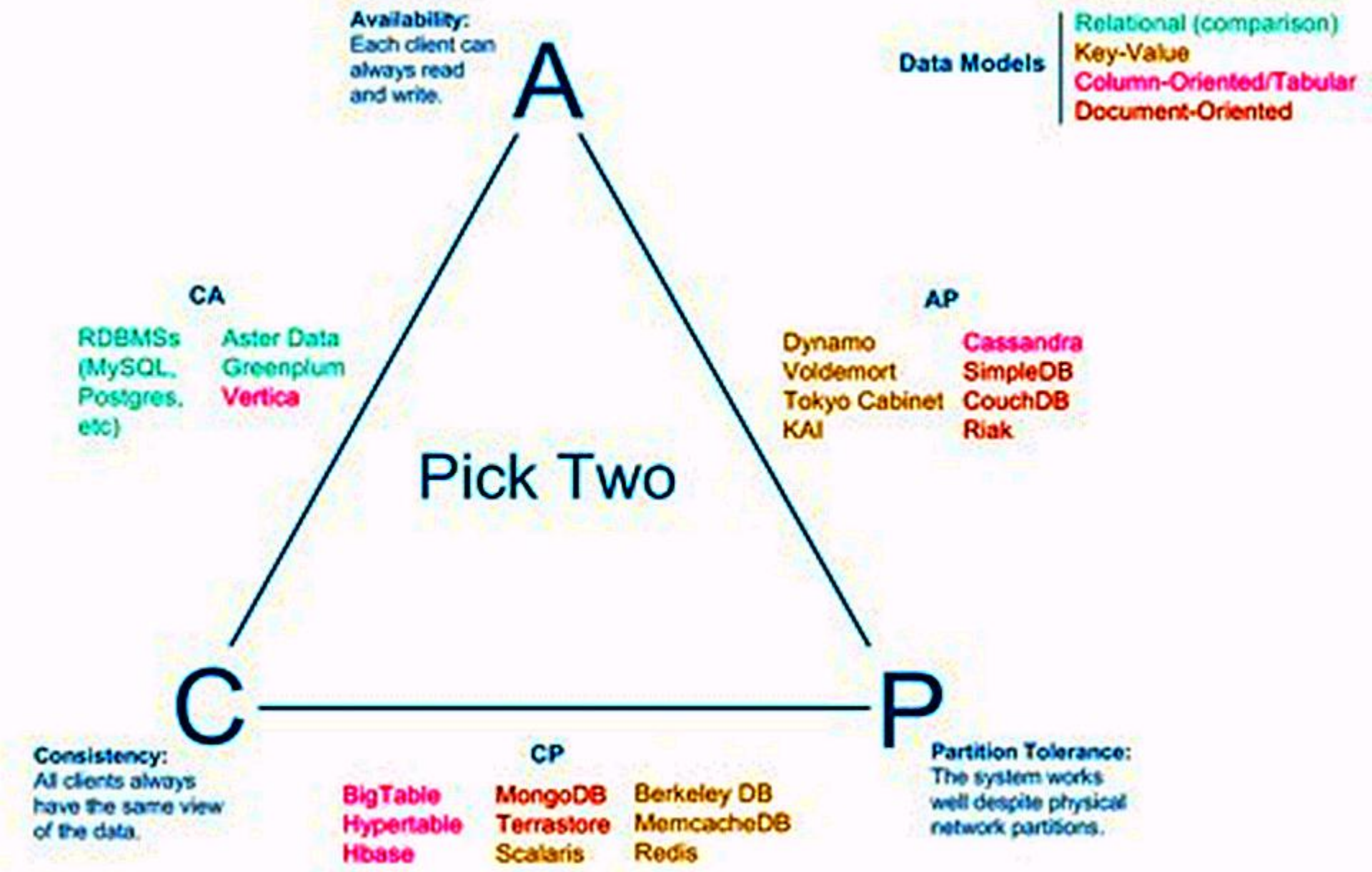
-> Anforderungen

- Annahme
  - Persistente Datenspeicherung
  - Horizontale Skalierung
  - Extensible Datenstruktur
  - Keine beliebigen dynamischen Abfragen
  - Hohe Lese-/Schreibleistung

  - Arrays, Listen, Hashmaps, BLOBs
  - Atomare Operationen
    - Erstellen, Löschen, Zugriff per Schlüssel
    - Konstante Komplexität von O(1) für Zugriffe

  - Viele Anwendungen (Desktop und Web)

Visual Guide to NoSQL Systems



-> Kompromisse

- Nachteile (-)
  - Integrität
    - Fremdschlüssel / Redundanz
  - Denormalisierung
  - Keine dyn. strukturierte Abfrage
    - JOINS
  - Leichtgewichtig (aufwendig)

- Vorteile (+)
  - Partitionierung
  - Laufzeitflexibilität (schemalos)
  - Leichtgewichtig (schnell)
  - Skalierung (horizontal)
  - Hashmaps & Assoziative Arrays
    - Einfügen, Löschen, Zugriff per Schlüssel

-> NoSQL Kategorien

Graph	Column	Document	Persistent Key/Value	Volatile Key/Value
neo4j	BigTable (Google)	MongoDB (~BigTable)	Dynamo (Amazon)	memcached
FlockDB (Twitter)	HBase (BigTable)	CouchDB	Voldemort (Dynamo)	Hazelcast
InfiniteGraph	**Cassandra (Dynamo + BigTable)**	Riak (Dynamo)	**Redis**	
	Hypertable (BigTable)		Membase (memcached)	
	SimpleDB (AmazonAWS)		Tokyo Cabinet	

-> Key-Value Kandidaten

- Key-Value Objekt Zwischenspeicher (Cache)
  - Memcached
    - Wikipedia, Twitter, Wordpress, Flickr, ...
- Persistenter In-Memory Key-Value Datenspeicher
  - Redis
  - MemcachedDB
- Verteilte, hochverfügbare Key-Value Verwaltungsumgebung
  - Cassandra

-> Zwischenspeicher

- Memcached
  - In-Memory, keine Persistenz
  - Schneller Key-Value Objekt Zwischenspeicher (Cache)
    - LRU (Least Recent Used) Prinzip
- MemcacheDB (2008)
  - Memcached Client-APIs / Protokoll
  - BerkleyDB als persistentes Speichersystem
    - Transaktionen und Replikation
  - Hohe Leistung für Key-Value Objekte
- Perl, C, Python, Java, ...

-> Memcached Beispiel

```
function get_foo(foo_id){
        foo = memcached_get("foo:" . foo_id);
        if defined foo
                return foo;

        foo = fetch_foo_from_database(foo_id);
        memcached_set("foo:" . foo_id, foo);

        return foo;
end
```

- Zwischenspeichern von DB Abfragen
  - Häufig gestellte Benutzeranfragen
- Asynchrone Zugriffe / Abfangen von Serverlatenzen

-> Memcached mit PHP

```
<?php

$memcache = new Memcache;
$memcache->connect('localhost', 11211) or die ("Could not connect");

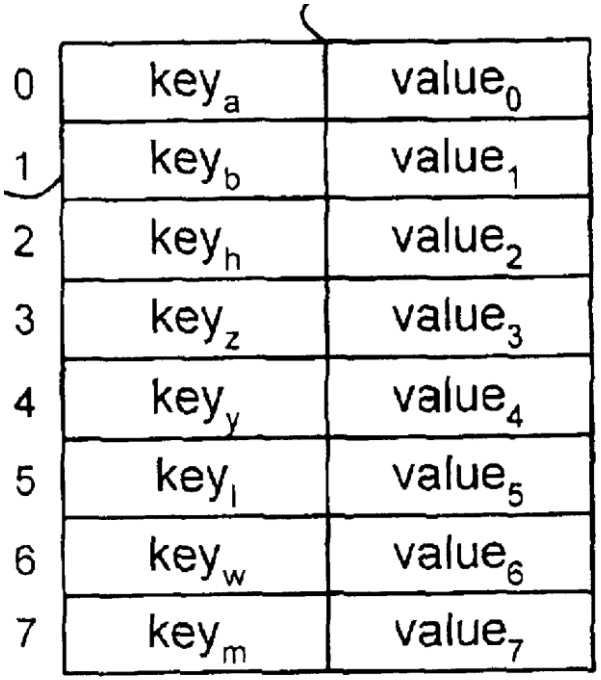
$version = $memcache->getVersion();
echo "Server's version: ".$version."<br/>\n";

$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;
$memcache->set('key', $tmp_object, false, 10) or die ("Failed to save
data at the server");
echo "Store data in the cache (data will expire in 10 seconds)<br/>\n";

$get_result = $memcache->get('key');
echo "Data from the cache:<br/>\n";
?>
```

-> Key-Value Datenstruktur

- Assoziatives Array, Dictionary, Key-Value Map
- Implementierung via Hashmaps, Binärbäume oder Arrays
- Feld von Key-Value Paaren
  - Indizierung der Werte über Schlüssel
  - Schlüssel Datentyp i.d.R. String
  - Wert Datentypen
    - String, List, Hashmap, BLOB ...



- Python, Perl, JavaScript, Ruby, Lua, PHP, ...
- C/C++, .NET, Java, Groovy, Smalltalk, Lisp , Objective-C

-> Key-Value Datenspeicher

- Erstellen, Lesen, Schreiben, Löschen
  - CRUD (Create, Read, Update, Delete)
- Atomare serverseitige Operationen im Datenspeicher ermöglichen verteilte Zugriffe
  - Redis, MemcachedDB, Cassandra
- Intelligenter Client – Intelligenter Server
  - Hohe Effizienz u.a. bei „kleinen“ Datenvolumen
  - Mehr Entwicklungsaufwand
- CAP / Datenpersistenz
  - Verzögerte Konsistenz (Eventual Consistency)
  - Asynchrone Persistenz

-> Redis

- Schneller In-Memory Key-Value Datenspeicher
  - Salvatore Sanfilippo, 2009
  - VMWare, 2012 (gekauft)
- Asynchrone Persistenz und Journaling
- Master/Slave Replikation (Redundanz, Skalierung)
- Lists, Sets, Sorted Sets, Hashmaps
- Strings und Binärdaten (BLOB)
- Atomare Operationen
  - Union, Intersect, Difference, Create, Delete, Write, ...
- C, C++, C#, Erlang, Go, Java, Lua, Objective-C, Perl, PHP, Python, Ruby, Scala, Smalltalk, ...

-> Redis Clients

- Kommandozeilen Schnittstelle
- ```
  redis> SET erster_schluessel "Hello World"
  OK
  redis> GET erster_schluessel
  "Hello World"
  redis>
  ```

- Java Client API
- ```
  JRedis jredis = new JRedisClient(); // Default-Werte
  fuer Host
  // (localhost) und Port (6379)
  jredis.set("erster_schluessel", "Hello World");
  System.out.println(jredis.get("erster_schluessel"));
  ```

-> Key-Value mit 

- Cassandra
  - Facebook, 2008
- Verteiltes, strukturiertes Key-Value Datenverwaltungsumgebung
  - Amazon Dynamo und Google BigTable

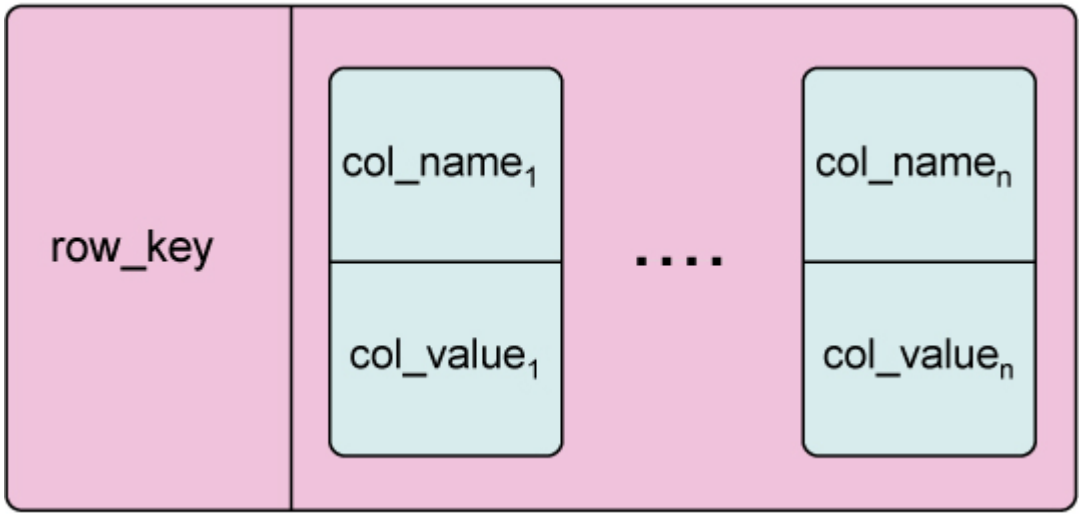
- Hochverfügbarkeit/ Skalierbarkeit
  - Parallelität auf sog. Commodity Hardware
  - Anpassbare Konsistenzstrenge (Tunable/Eventual) Consistency
  - Fehlertoleranz und Geschwindigkeit
  - Master/Master Replikation
- BASE, nicht ACID Semantik

- C++, .NET, Ruby, Perl, PHP, Python, Java,...

-> Cassandra

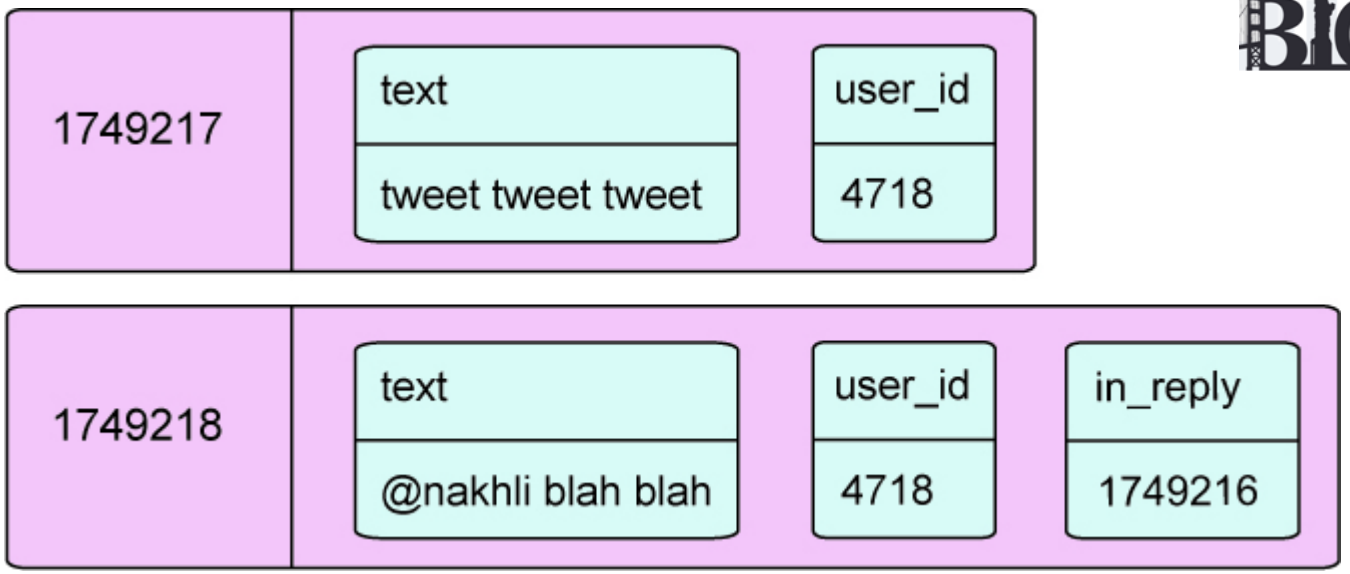
- Anpassbare Konsistenz (CAP / BASE)
  - Strikte und verzögerte Konsistenz , Quorum
  - Anforderungen prüfen
- Replikation zwischen Servern und Clustern
  - Gossip-Protokoll
- Cassandra Query Language
- MapReduce via Hadoop Infrastruktur
- Optional schematisches Datenmodel
  - Flexible Strukturierung der Daten
  - Ausnutzung möglicher Datenlokalität

-> Datenmodel



- Schlüssel verweist auf beliebig viele Spalten (Columns) mit dazugehörigen Werten
- Spalten werden in Spaltenfamilien (Column-Families) geordnet

-> Datenmodel (2)



- Clustering
  - Erhalten der Reihenfolge : OrderPreservingPartitioner
  - Zufällige Verteilung : RandomPartitioner

-> CQL

- SQL artige Syntax
- Keine JOINs und Bereichsabfragen
  - Partitionierung beachten
- Create, Alter, Drop Table, Create Index, Select, Delete, Update, Truncate, Use, ….
- ```
  SELECT * FROM MyColumnFamily;
  UPDATE MyColumnFamily
  SET 'SomeColumn' = 'SomeValue‘
  WHERE
  KEY = B70DE1D0-9908-4AE3-BE34-5573E5B09F14;
  ```
- Speicherparameter und Metadaten
  - Replikationsstrategien Kompression, ...
  ```

-> Cassandra CLI

- Schlüsselraum (Keyspace) anlegen

```
CREATE KEYSPACE demo with placement_strategy =
'org.apache.cassandra.locator.SimpleStrategy' AND
strategy_options = [{replication_factor:1}];
```

- Spaltenfamilie (Column-Family) Users anlegen

```
USE demo;
CREATE COLUMN FAMILY users WITH
comparator = UTF8Type AND
key_validation_class = UTF8Type AND
column_metadata = [
{column_name: full_name, validation_class: UTF8Type}
{column_name: email, validation_class: UTF8Type}
{column_name: state, validation_class: UTF8Type}
{column_name: gender, validation_class: UTF8Type}
{column_name: birth_year, validation_class: LongType}
];
```

-> Cassandra CLI

- Dynamische Spaltenfamilie erstellen

```
CREATE COLUMN FAMILY blog_entry WITH
comparator = TimeUUIDType AND
key_validation_class = UTF8Type AND
default_validation_class = UTF8Type;
```

- Eintrag in der Spaltenfamilie Users speichern

```
SET users['yomama']['full_name']='Cathy Smith';
SET users['yomama']['state']='CA';
SET users['yomama']['gender']='F,;
SET users['yomama']['birth_year']='1969';
```

- Speichern in Spaltenfamilie Blog_entry

```
SET blog_entry['yomama'][timeuuid()] =
'I love my new shoes!';
```

-> Cassandra CLI

- Format wählen

```
ASSUME users KEYS AS ascii;
ASSUME users COMPARATOR AS ascii;
ASSUME users VALIDATOR AS ascii;
```

- Time To Live

```
SET users['bobbyjo'] [utf8('coupon_code')] =
utf8('SAVE20') WITH ttl=864000;
```

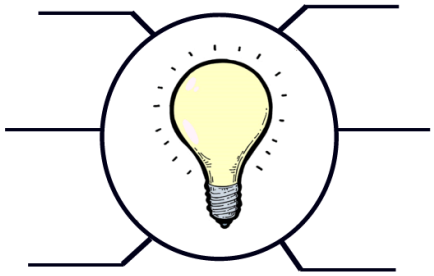
- Sekundäre Indizierung

```
UPDATE COLUMN FAMILY users WITH comparator =
UTF8Type AND column_metadata =
[{column_name: birth_year,
  validation_class: LongType,
  index_type: KEYS}];
GET users WHERE birth_date = 1969;
```

-> Resumee

- Anforderungen der Anwendung prüfen
- Datenvolumen / Big Data
- Statisches/ Dynamisches Datenschema
- Verteilung, Sharding, Clustering
- Zugriffsgeschwindigkeit / Zugriffsmuster
- Prototyping

- Strikte Konsistenz notwendig ?
- Stapelverarbeitung (Map Reduce)




http://192.168.178.25/ClusterAdmin/
Suchen:
Zurück
Weiter
Optionen
Cassandra Cluster Admin
Cluster Name: Test Cluster
Cluster Partitioner: org.apac
mPartitioner
Cluster Snitch: org.apache.
itch
Thrift API Version: 19.32.0
Schema Versions:
127.0.0.1
5f69f
a47b8ae2c18
Create New Keyspace
Keyspaces and Column Families

-> Quellen

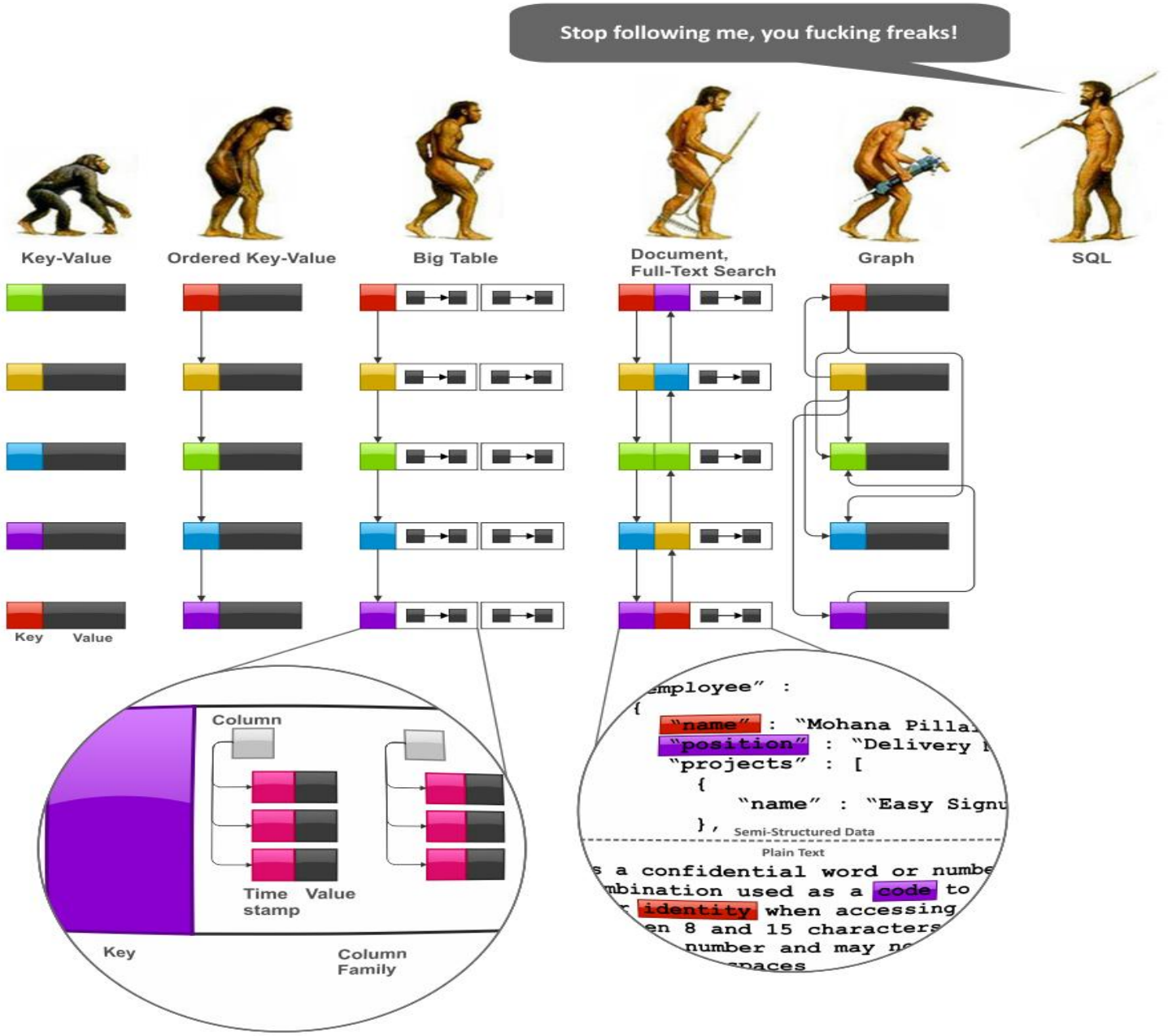
- Key-Value Datenspeicher
- http://www.slideshare.net/marc.seeger/keyvalue-stores-a-practical-overview
- http://blogupstairs.com/redisnosql-database-%e2%80%93-open-source-advanced-key-value-store/
- http://dbs.uni-leipzig.de/file/seminar_0910_Bin_Brekle.pdf
- http://www.heise.de/developer/artikel/NoSQL-Key-Value-Datenbank-Redis-im-Ueberblick-1233843.html
- Cassandra
- http://techpointers2020.blogspot.de/2011/07/casandra.html
- http://www.datastax.com/docs/0.8/dml/using_cli
- NoSQL
- http://it-republik.de/jaxenter/artikel/Graphendatenbanken-NoSQL-und-Neo4j-2906.html
- http://www.michaelnygard.com/blog/2007/11/architecting_for_latency.html
- http://decrypt.ysance.com/2010/12/sql-and-nosql/

Stand 06/2012

-> Fragen und Antworten

- Vielen Dank für Eure Aufmerksamkeit !

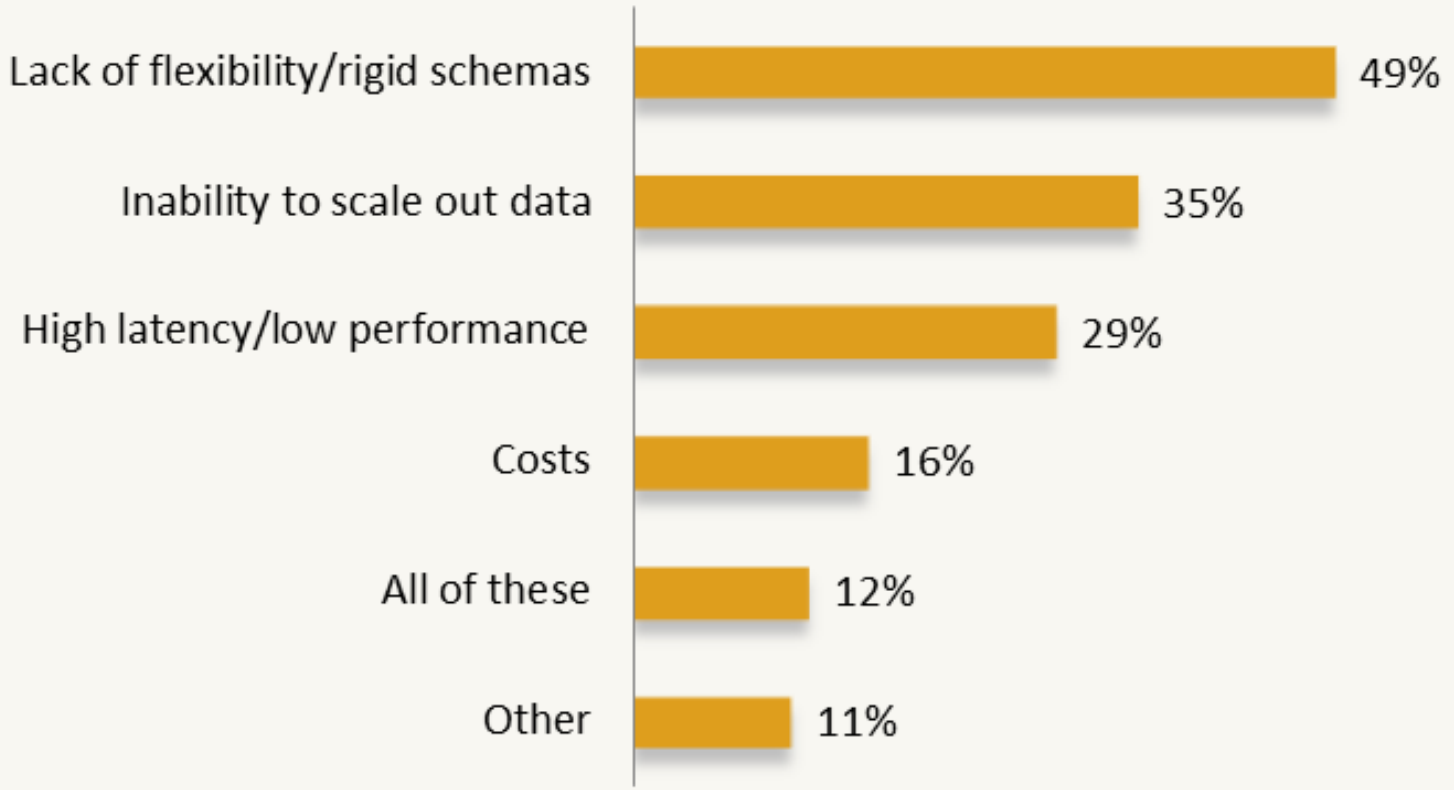
- Gibt es Fragen ?



Key-Value Stores - Lemmy Tauer

http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/

What is the biggest data management problem driving your use of NoSQL in the coming year?



Source: Couchbase NoSQL Survey, December 2011, n=1351