Das große Einmaleins der NoSQL Datenbanken

Felix Gessert
felix.gessert@baqend.com
@baqendcom

Slides: slideshare.net/felixgessert

Article: medium.com/baqend-blog

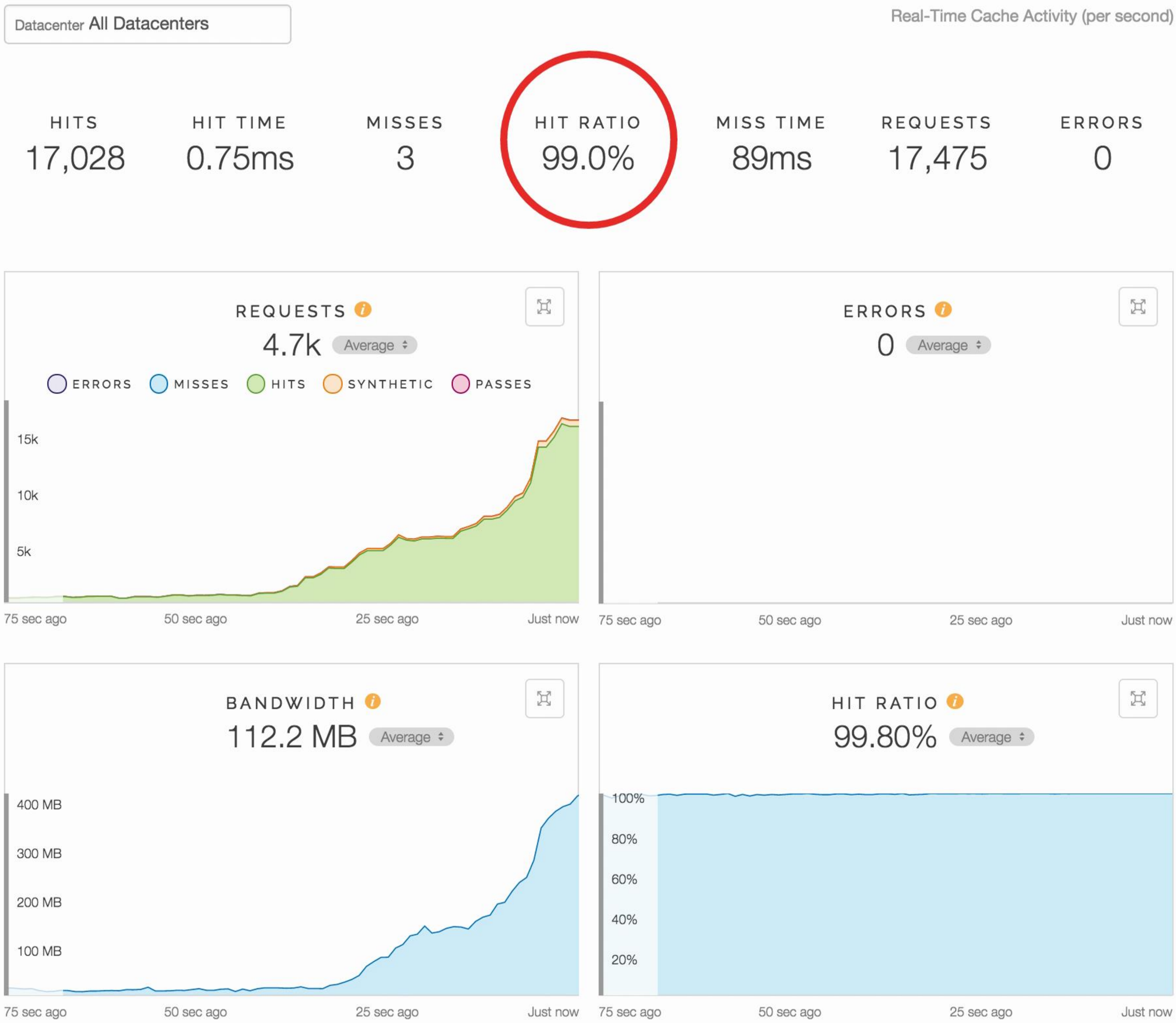
About me

NoSQL research for
PhD dissertation

Orestes

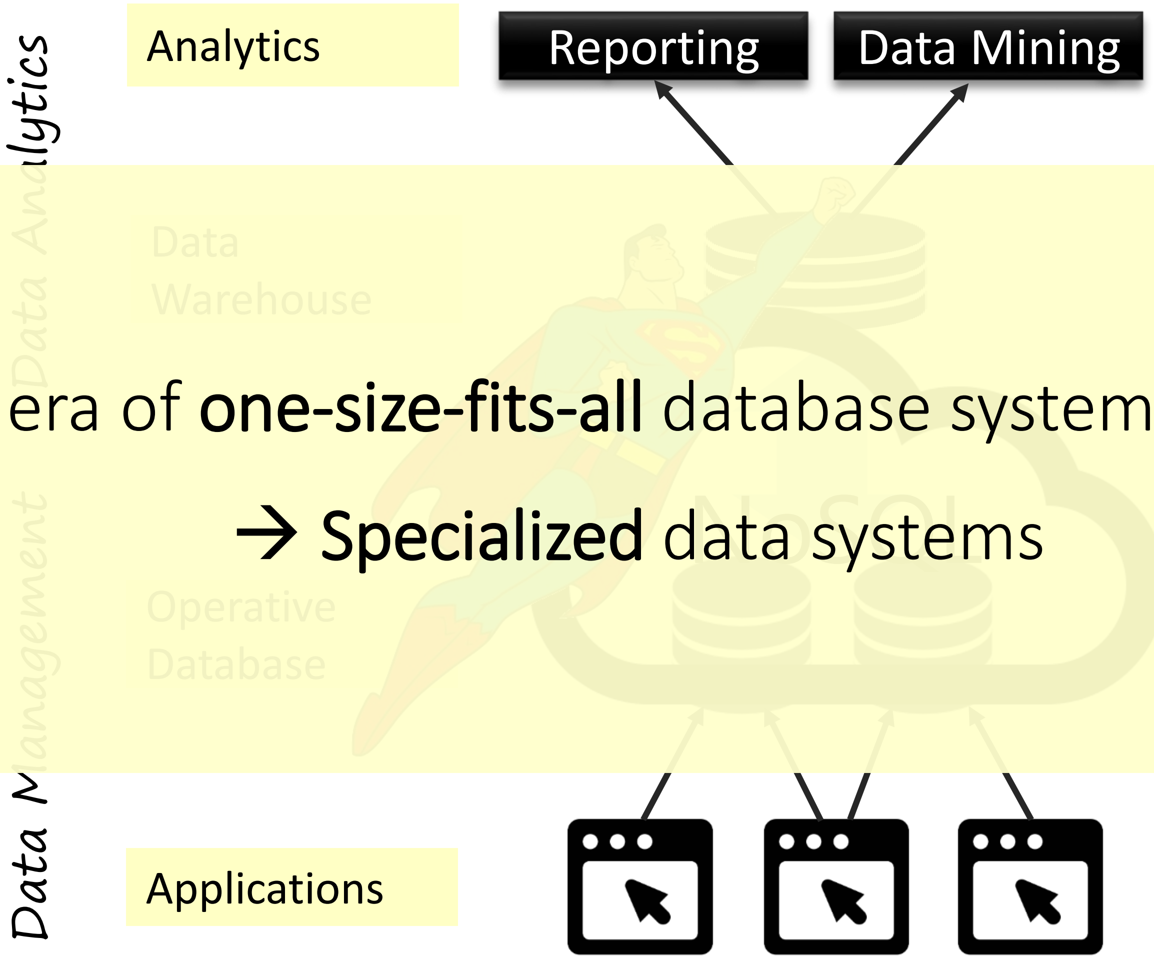
BaQend
Build faster Apps faster.

CEO of **startup** for high-performance
serverless development based on
caching and NoSQL


Datacenter All Datacenters
Real-Time Cache Activity (per second)
HITS 17,028
HIT TIME 0.75ms
MISSES 3
HIT RATIO 99.0%
MISS TIME 89ms
REQUESTS 17,475
ERRORS 0
REQUESTS 4.7k Average
ERRORS MISSES HITS SYNTHETIC PASSES
15k
10k
5k
75 sec ago
50 sec ago
25 sec ago
Just now
ERRORS 0 Average
75 sec ago
50 sec ago
25 sec ago
Just now
BANDWIDTH 112.2 MB Average
400 MB
300 MB
200 MB
100 MB
75 sec ago
50 sec ago
25 sec ago
Just now
HIT RATIO 99.80% Average
100%
80%
60%
40%
20%
75 sec ago
50 sec ago
25 sec ago
Just now

Architecture

Typical Data Architecture:



The era of **one-size-fits-all** database systems is over

→ **Specialized** data systems

The Database Explosion

Sweetspots

IBM DB2

RDBMS

General-purpose ACID transactions

Greenplum

Parallel DWH

Aggregations/OLAP for massive data amounts

VoltDB

NewSQL

High throughput relational OLTP

HBASE

Wide-Column Store

Long scans over structured data

mongoDB

Document Store

Deeply nested data models

riak

Key-Value Store

Large-scale session storage

Neo4j the graph database

Graph Database

Graph algorithms & queries

redis

In-Memory KV-Store

Counting & statistics

cassandra

Wide-Column Store

Massive user-generated content

The Database Explosion

Cloud-Database Sweetspots

Firebase

Realtime BaaS

Communication and collaboration

Amazon RDS

Managed RDBMS

General-purpose ACID transactions

Amazon ElastiCache

Managed Cache

Caching and transient storage

Azure Tables

Wide-Column Store

Very large tables

Amazon DynamoDB

Wide-Column Store

Massive user-generated content

Parse

Backend-as-a-Service

Small Websites and Apps

bonsai

Managed NoSQL

Full-Text Search

Google Cloud Storage

Object Store

Massive File Storage

Amazon Elastic MapReduce

Hadoop-as-a-Service

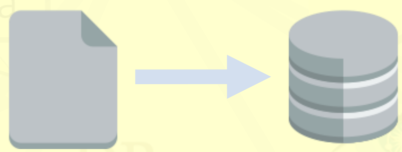
Big Data Analytics

How to choose a database system?

Many Potential Candidates

Question in this tutorial:

How to approach the



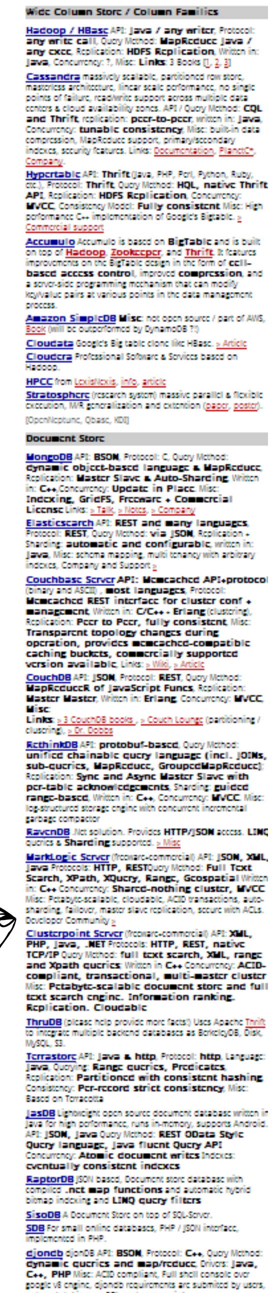
requirements database

decision problem?

NoSQL Databases

- „NoSQL“ term coined in 2009
- Interpretation: „**N**ot **O**nly **SQL**“
- Typical properties:
  - Non-relational
  - Open-Source
  - Schema-less (*schema-free*)
  - Optimized for distribution (clusters)
  - Tunable consistency

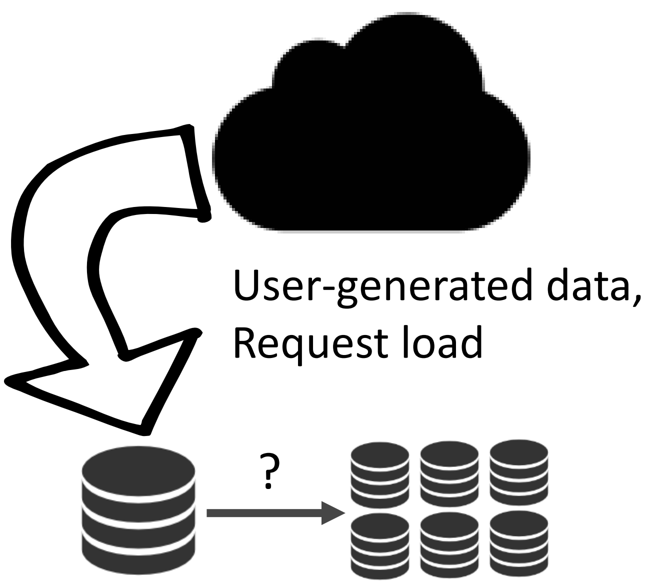
***NoSQL-Databases.org*:**
Current list has over 150
NoSQL systems



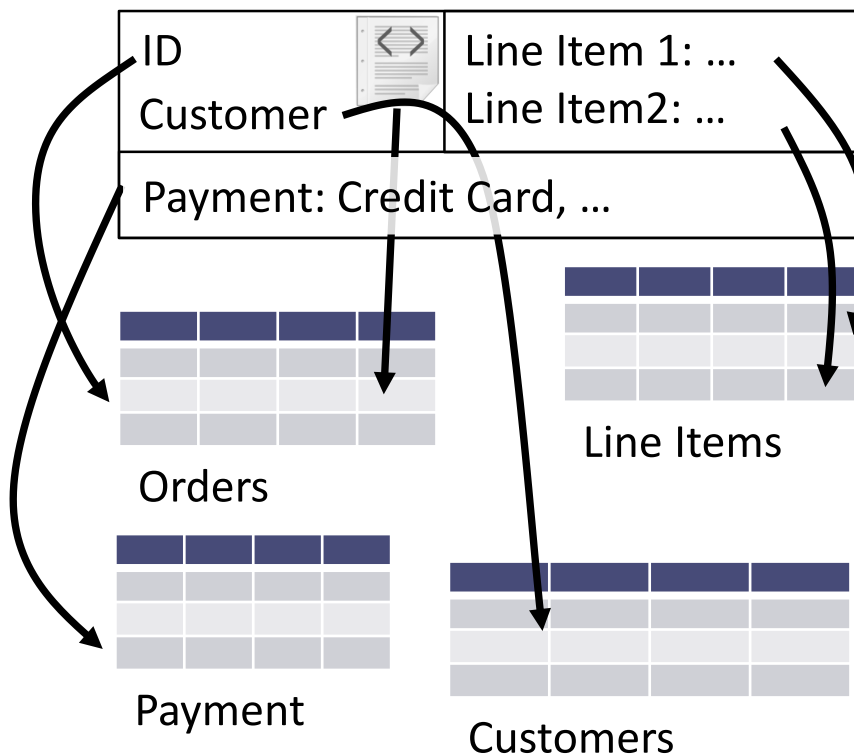
NoSQL Databases

- Two main motivations:

Scalability

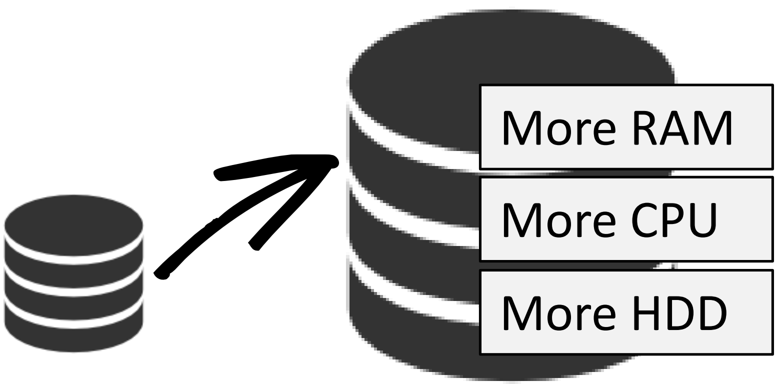


Impedance Mismatch

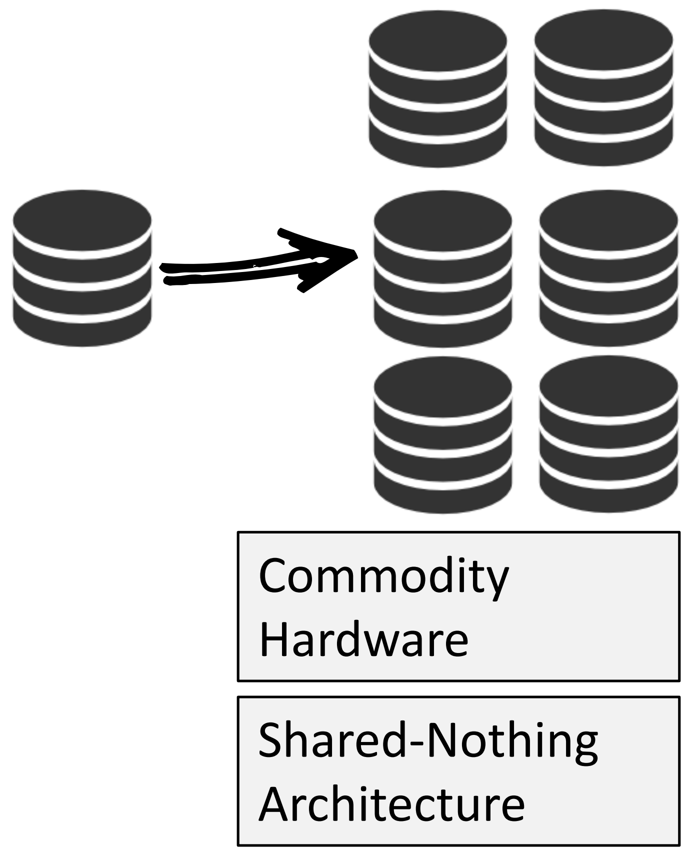


Scale-up vs Scale-out

Scale-Up (*vertical* scaling):

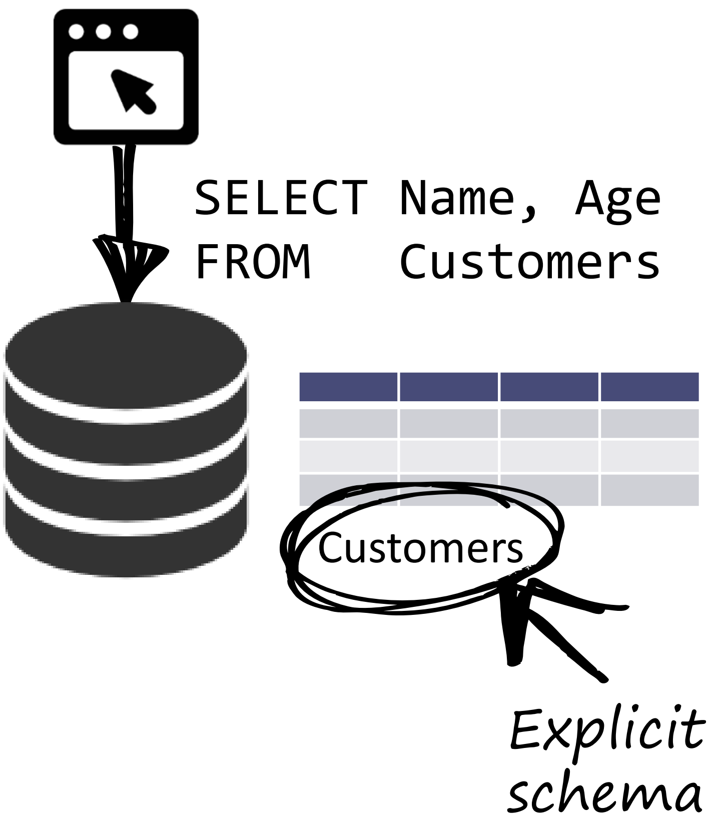


Scale-Out (*horizontal* scaling):

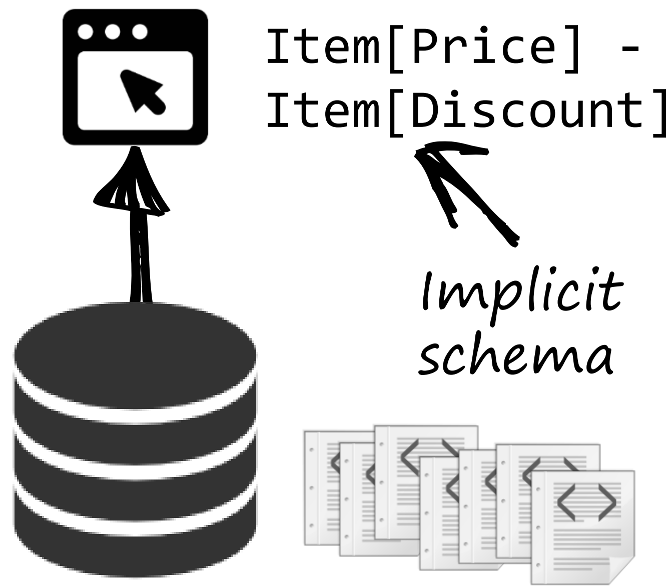


Schemafree Data Modeling

RDBMS:



NoSQL DB:



Open Source & Commodity Hardware

Commercial DBMS	Open-Source DBMS
Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Highly Available Storage (SAN, RAID, etc.)	Commodity drives (standard HDDs, JBOD)

NoSQL System Classification

- Two common criteria:

Data Model

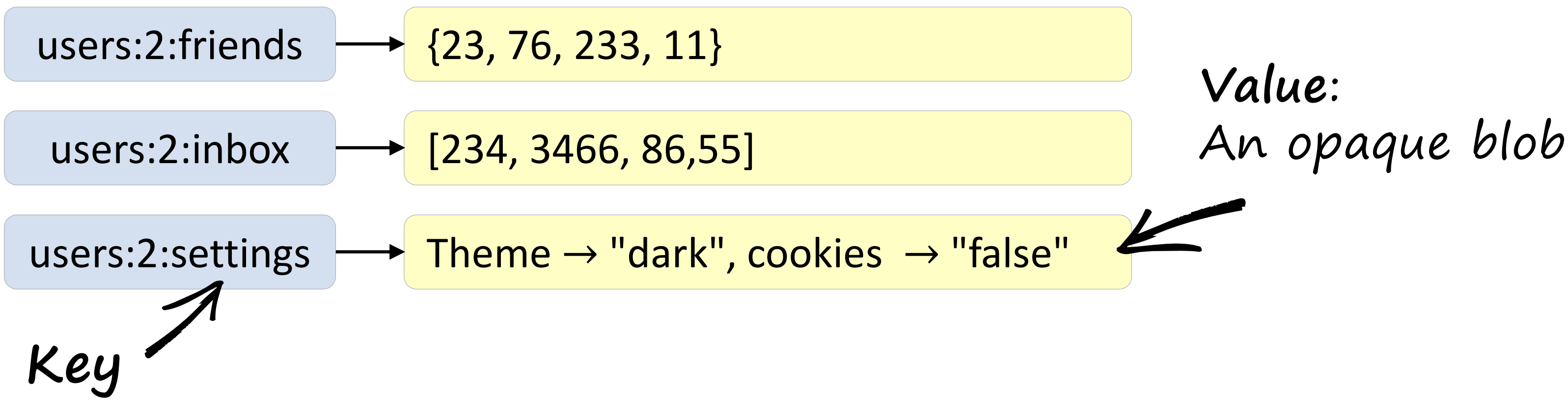
- Key-Value
- Wide-Column
- Document
- Graph

Consistency/Availability Trade-Off

- **AP**: Available & Partition Tolerant
- **CP**: Consistent & Partition Tolerant
- **CA**: Not Partition Tolerant

Key-Value Stores

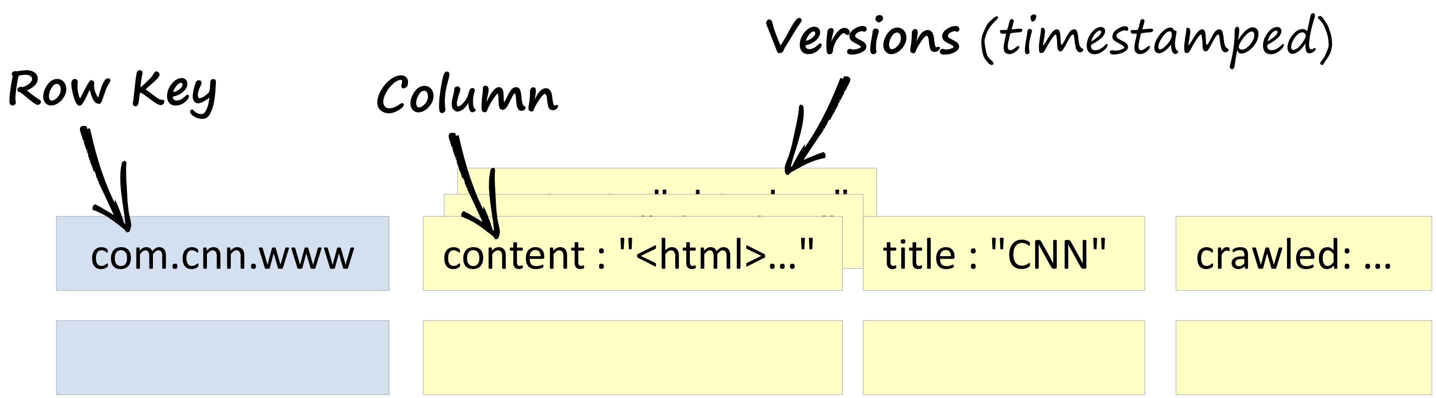
- **Data model:** (key) -> value
- **Interface**: CRUD (Create, Read, Update, Delete)



- Examples: Amazon Dynamo (AP), Riak (AP), Redis (CP)

Wide-Column Stores

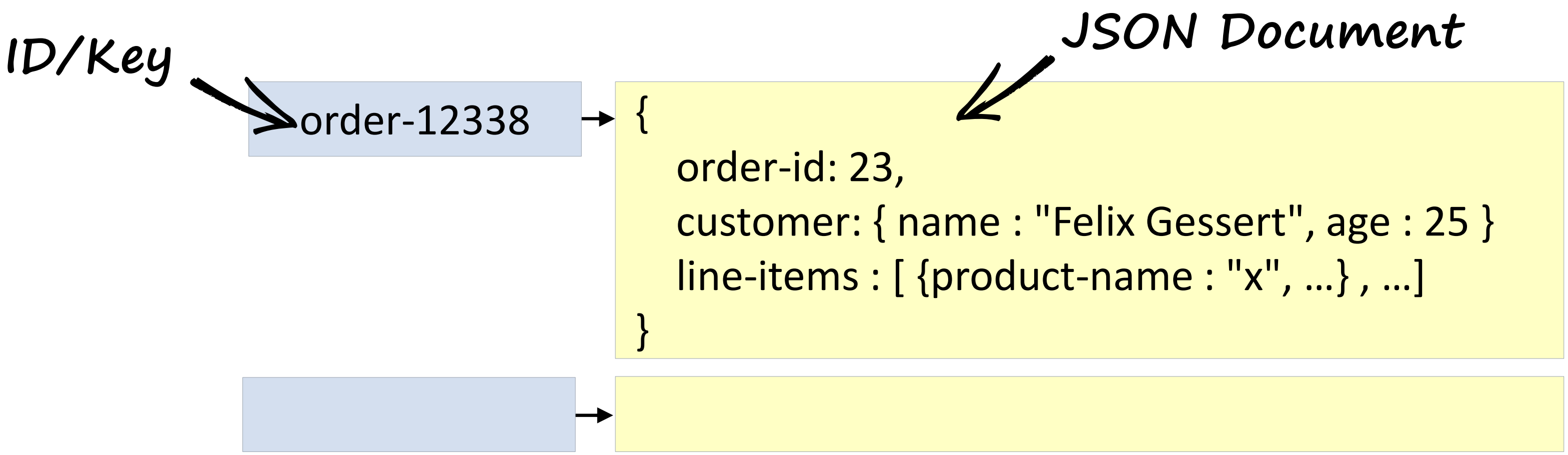
- **Data model:** (rowkey, column, timestamp) -> value
- **Interface**: CRUD, Scan



- Examples: Cassandra (AP), Google BigTable (CP), HBase (CP)

Document Stores

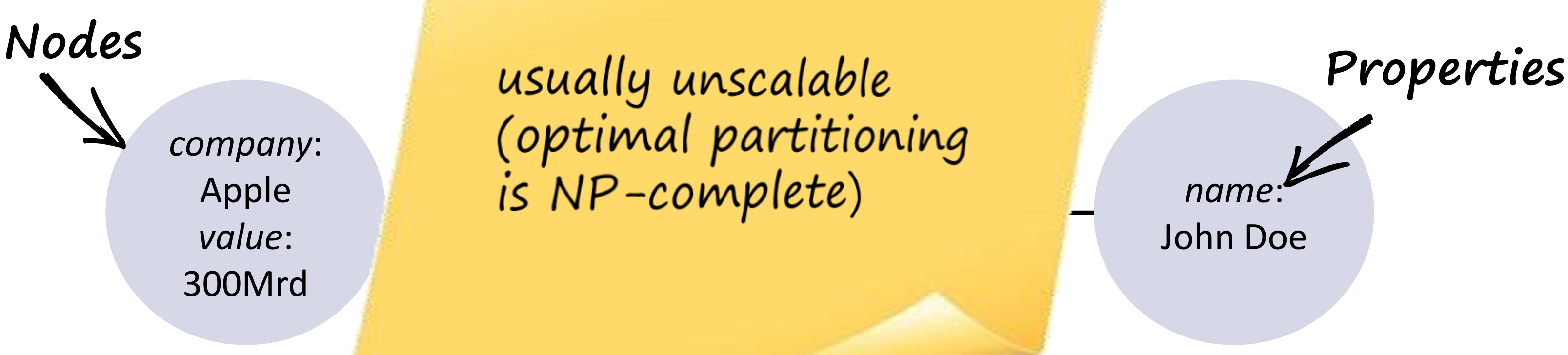
- **Data model:** (collection, key) -> document
- **Interface**: CRUD, Querys, Map-Reduce



- Examples: CouchDB (AP), Amazon SimpleDB (AP), MongoDB (CP)

Graph Databases

- **Data model:** G = (V, E): Graph-Property Modell
- **Interface**: Trave… , transactions



- Examples: Neo4j (CA), InfiniteGraph (CA), OrientDB (CA)

Soft NoSQL Systems

Not Covered Here

Search Platforms (Full Text Search):

- No persistence and consistency guarantees for OLTP
- *Examples*: ElasticSearch (AP), Solr (AP)

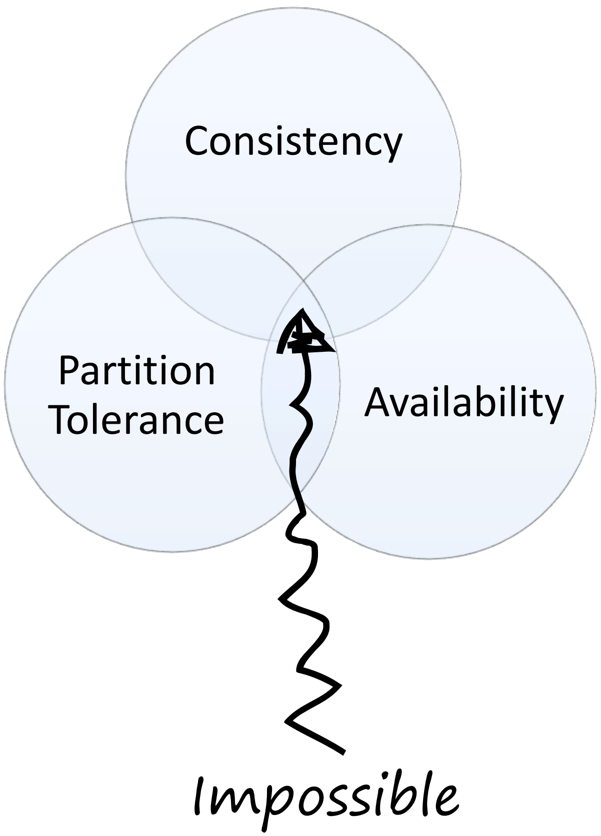
Object-Oriented Databases:

- Strong coupling of programming language and DB
- *Examples*: Versant (CA), db4o (CA), Objectivity (CA)

XML-Databases, RDF-Stores:

- Not scalable, data models not widely used in industry
- *Examples*: MarkLogic (CA), AllegroGraph (CA)

CAP-Theorem



Only 2 out of 3 properties are achievable at a time:

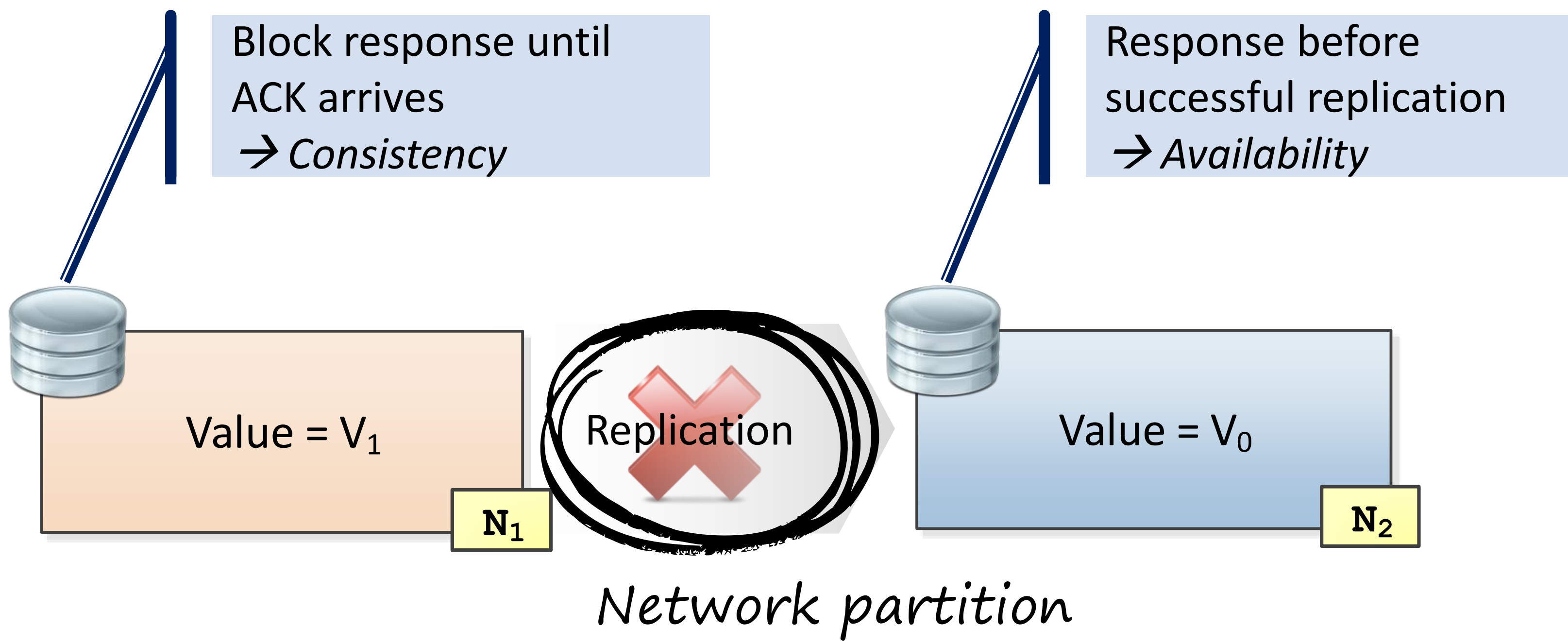
- **Consistency**: all clients have the same view on the data
- **Availability**: every request to a non-failed node most result in correct response
- **Partition tolerance**: the system has to continue working, even under arbitrary network partitions

Eric Brewer, ACM-PODC Keynote, Juli 2000

Gilbert, Lynch: Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services, SigAct News 2002

CAP-Theorem: simplified proof

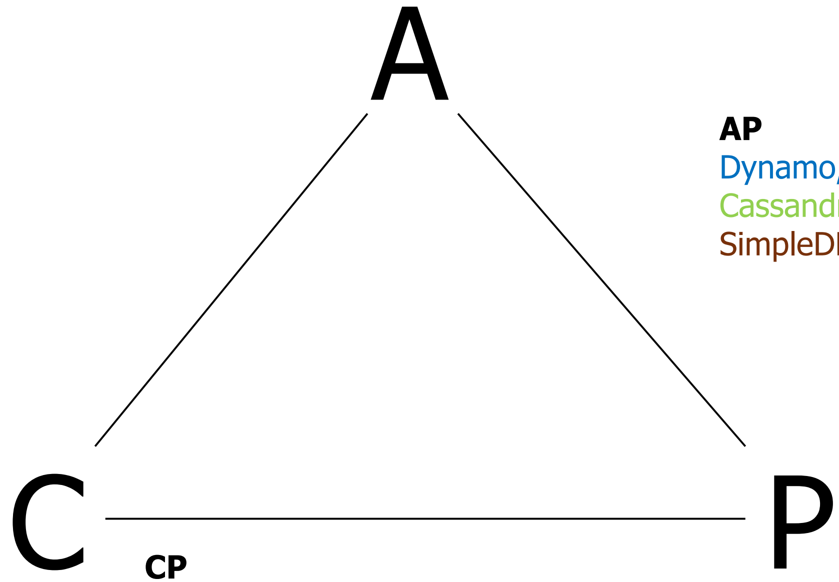
- **Problem**: when a network partition occurs, either consistency or availability have to be given up



NoSQL Triangle

Data models
Relational
Key-Value
Wide-Column
Document-Oriented

Every client can always read and write



CA
Oracle, MySQL, …

AP
Dynamo, Redis, Riak, Voldemort
Cassandra
SimpleDB,

All clients share the same view on the data

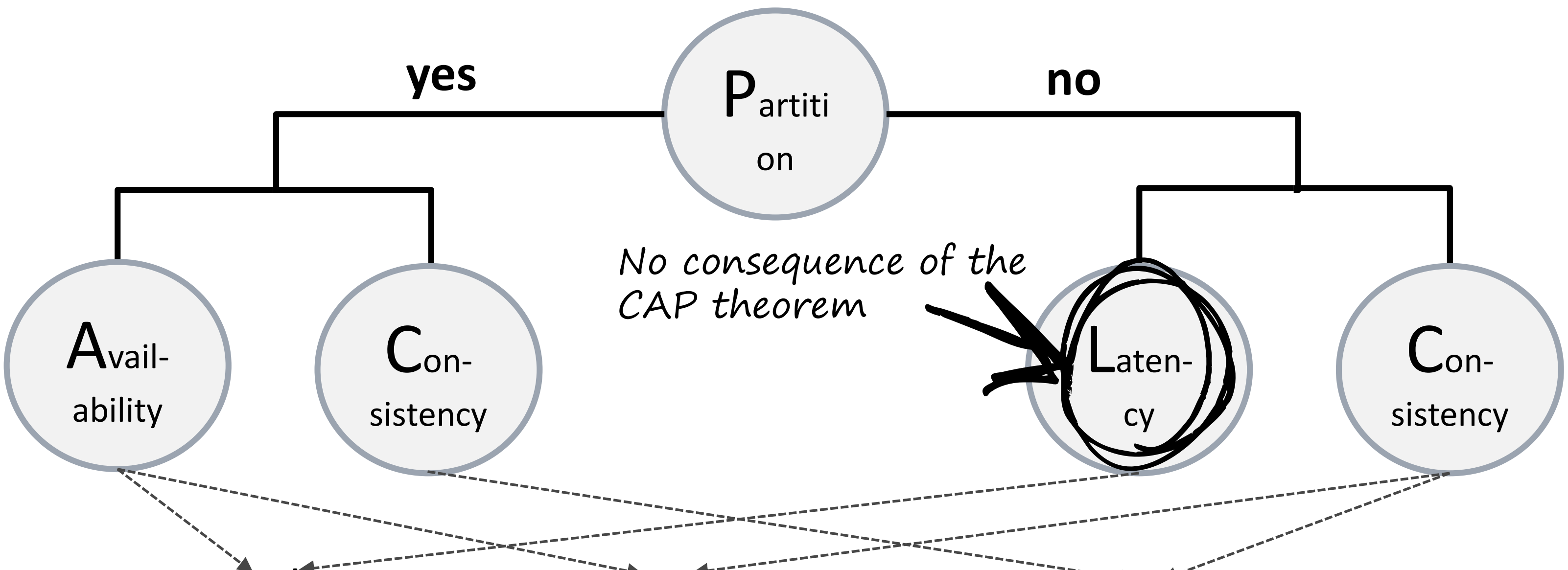
CP
Postgres, MySQL Cluster, Oracle RAC
BigTable, HBase, Accumulo, Azure Tables
MongoDB, RethinkDB

All nodes continue working under network partitions

Nathan Hurst: Visual Guide to NoSQL Systems
http://blog.nahurst.com/visual-guide-to-nosql-systems

PACELC – an alternative CAP formulation

- **Idea**: Classify systems according to their behavior during *network partitions*



AL - Dynamo-Style
Cassandra, Riak, etc.

AC - MongoDB

CC – Always Consistent
HBase, BigTable and ACID systems

Abadi, Daniel. "Consistency tradeoffs in modern distributed database system design: CAP is only part of the story."

Serializability

Not Highly Available Either

Global serializability and availability are incompatible:

Write A=1
Read B

Write B=1
Read A

$w_1(a = 1)\, r_1(b = \bot)$

$w_2(b = 1)\, r_2(a = \bot)$

- Some weaker isolation levels allow high availability:
  - RAMP Transactions (P. Bailis, A. Fekete, A. Ghodsi, J. M. Hellerstein, und I. Stoica, „Scalable Atomic Visibility with RAMP Transactions", SIGMOD 2014)

S. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in partitioned networks. ACM CSUR, 17(3):341–370, 1985.

Where CAP fits in

Negative Results in Distributed Computing

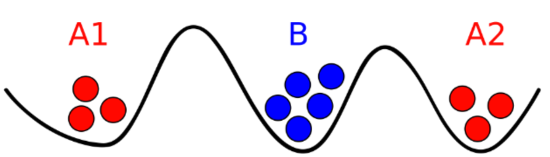
Asynchronous Network, _Unreliable_ Channel

Atomic Storage

Impossible:
CAP Theorem

Consensus

Impossible:
2 Generals Problem



Asynchronous Network, _Reliable_ Channel

Atomic Storage

Possible:
Attiya, Bar-Noy, Dolev (ABD) Algorithm

Consensus

Impossible:
Fisher **L**ynch **P**atterson (FLP) Theorem

Lynch, Nancy A. *Distributed algorithms*. Morgan Kaufmann, 1996.

ACID vs BASE

„Gold standard" for RDBMSs

ACID

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

BASE

- **B**asically **A**vailable
- **S**oft State
- **E**ventually Consistent

Model of many NoSQL systems

http://queue.acm.org/detail.cfm?id=1394128

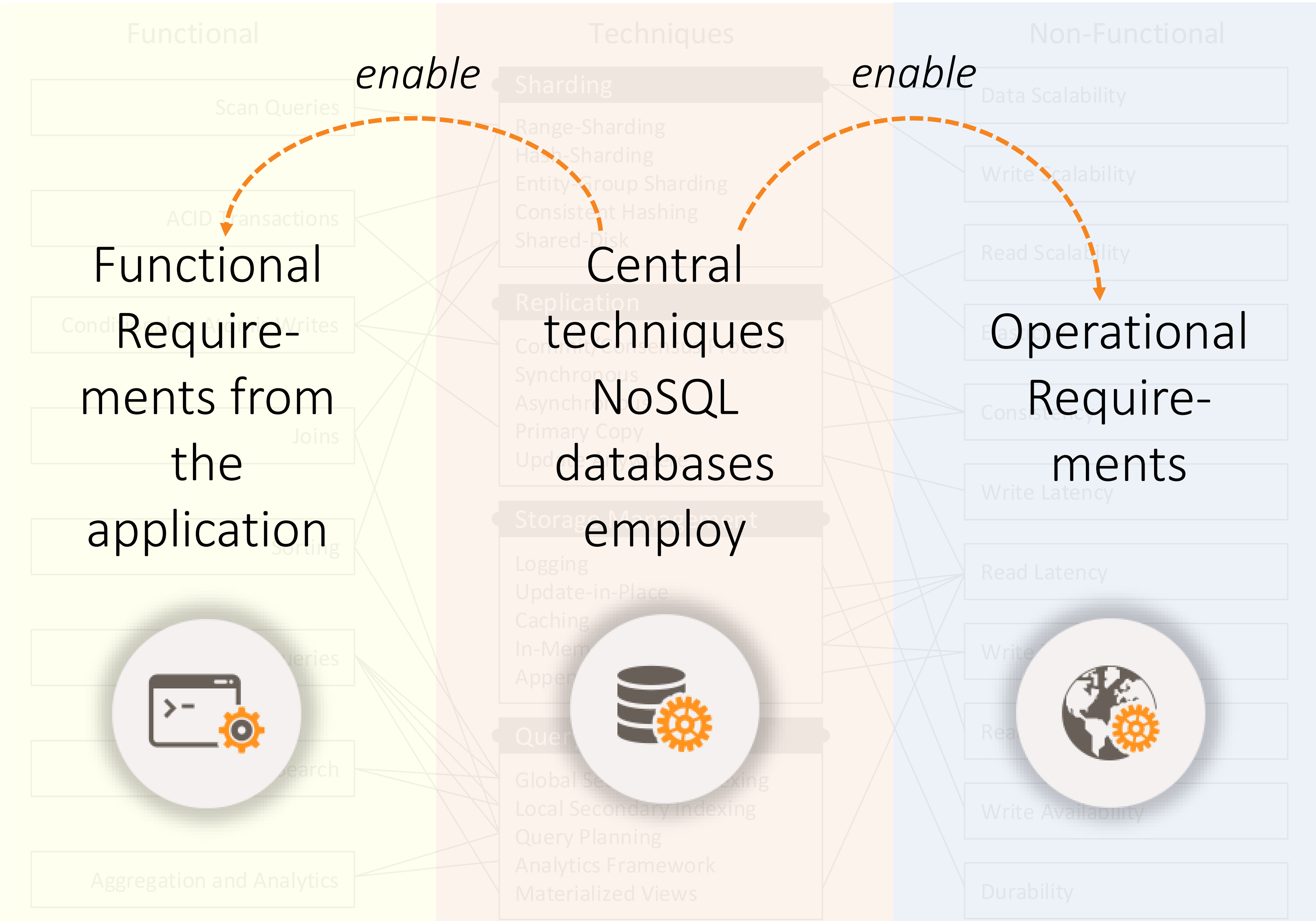
Data Models and **CAP** provide high-level classification.

But what about **fine-grained requirements**, e.g. query capabilites?

Outline

- NoSQL Foundations and Motivation
- The NoSQL Toolbox: Common Techniques
- NoSQL Systems
- Decision Guidance: NoSQL Decision Tree

- Techniques for Functional and Non-functional Requirements
  - Sharding
  - Replication
  - Storage Management
  - Query Processing



enable
enable
Functional Require-ments from the application
Central techniques NoSQL databases employ
Operational Require-ments

NoSQL Database Systems: A Survey and Decision Guidance

Felix Gessert, Wolfram Wingerath, Steffen Friedrich, and Norbert Ritter

Universität Hamburg, Germany
{gessert, wingerath, friedrich, ritter}@informatik.uni-hamburg.de

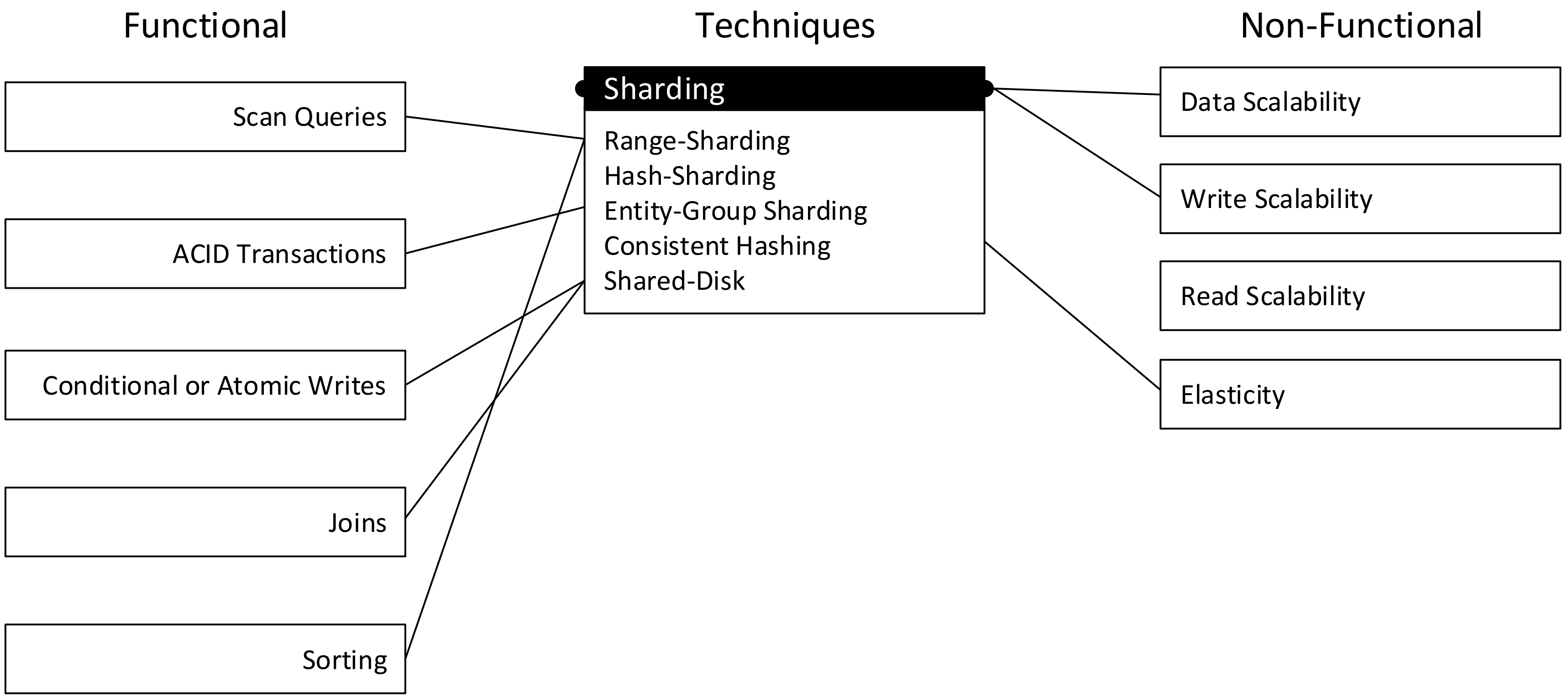
Abstract. Today, data is generated and consumed at unprecedented scale. This has lead to novel approaches for scalable data management subsumed under the term "NoSQL" database systems to handle the ever-increasing data volume and request loads. However, the heterogeneity and diversity of the numerous existing systems impede the well-informed selection of a data store appropriate for a given application context. Therefore, this article gives a top-down overview of the field: Instead of contrasting the implementation specifics of individual representatives, we propose a comparative classification model that relates functional and non-functional requirements to techniques and algorithms employed in NoSQL databases. This NoSQL Toolbox allows us to derive a simple decision tree to help practitioners and researchers filter potential system candidates based on central application requirements.

1 Introduction

Traditional relational database management systems (RDBMSs) provide powerful mechanisms to store and query structured data under strong consistency and transaction guarantees and have reached an unmatched level of reliability, stability and support through decades of development. In recent years, however, the amount of useful data in some application areas has become so vast that it cannot be stored or processed by traditional database solutions. User-generated content in social networks or data retrieved from large sensor networks are only two examples of this phenomenon commonly referred to as **Big Data** [35]. A class of novel data storage systems able to cope with Big Data are subsumed under the term **NoSQL databases**, many of which offer horizontal scalability and higher availability than relational databases by sacrificing querying capabilities and consistency guarantees. These trade-offs are pivotal for service-oriented computing and as-a-service models, since any stateful service can only be as scalable and fault-tolerant as its underlying data store.

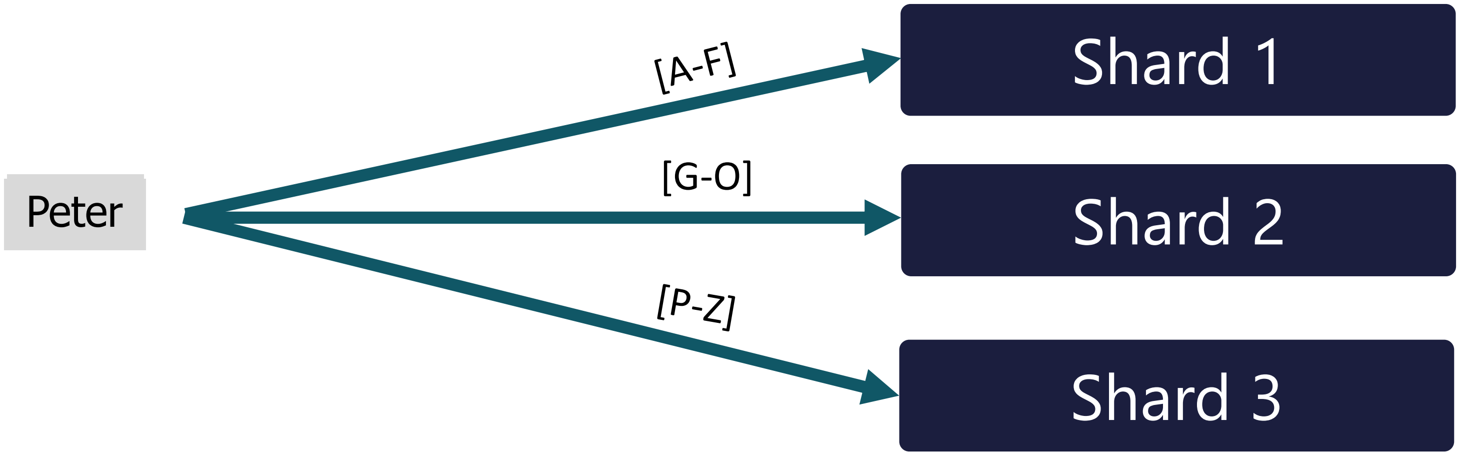
There are dozens of NoSQL database systems and it is hard to keep track of where they excel, where they fail or even where they differ, as implementation details change quickly and feature sets evolve over time. In this article, we therefore aim to provide an overview of the NoSQL landscape by discussing employed concepts rather than system specificities and explore the requirements typically posed to NoSQL database systems, the techniques used to fulfil these requirements and the trade-offs that have to be made in the process. Our focus lies on key-value, document and wide-column stores, since these NoSQL categories

http://www.baqend.com/files/nosql-survey.pdf


Functional
Techniques
Non-Functional
Scan Queries
ACID Transactions
Conditional or Atomic Writes
Joins
Sorting
Sharding
Range-Sharding
Hash-Sharding
Entity-Group Sharding
Consistent Hashing
Shared-Disk
Data Scalability
Write Scalability
Read Scalability
Elasticity

Sharding (aka Partitioning, Fragmentation)

- Horizontal distribution of data over nodes



- **Partitioning strategies**: Hash-based vs. Range-based
- Difficulty: Multi-Shard-Operations (join, aggregation)

Sharding

Hash-based Sharding

- Hash of data values (e.g. key) d
- **Pro**: Even distribution
- **Contra**: No data locality

Range-based Sharding

- Assigns ranges defined over fie
- **Pro**: Enables *Range Scans* and S
- **Contra**: Repartitioning/balancin

Entity-Group Sharding

- Explicit data co-location for sin
- **Pro**: Enables *ACID Transactions*
- **Contra**: Partitioning not easily c

Implemented in

MongoDB, Riak, Redis, Cassandra, Azure Table, Dynamo

Implemented in

BigTable, HBase, DocumentDB Hypertable, MongoDB, RethinkDB, Espresso

Implemented in

G-Store, MegaStore, Relation Cloud, Cloud SQL Server

David J DeWitt and Jim N Gray: "Parallel database systems: The future of high performance database systems," Communications of the ACM, volume 35, number 6, pages 85–98, June 1992.

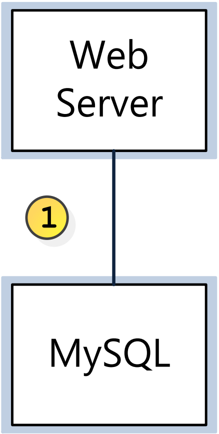
Problems of Application-Level Sharding

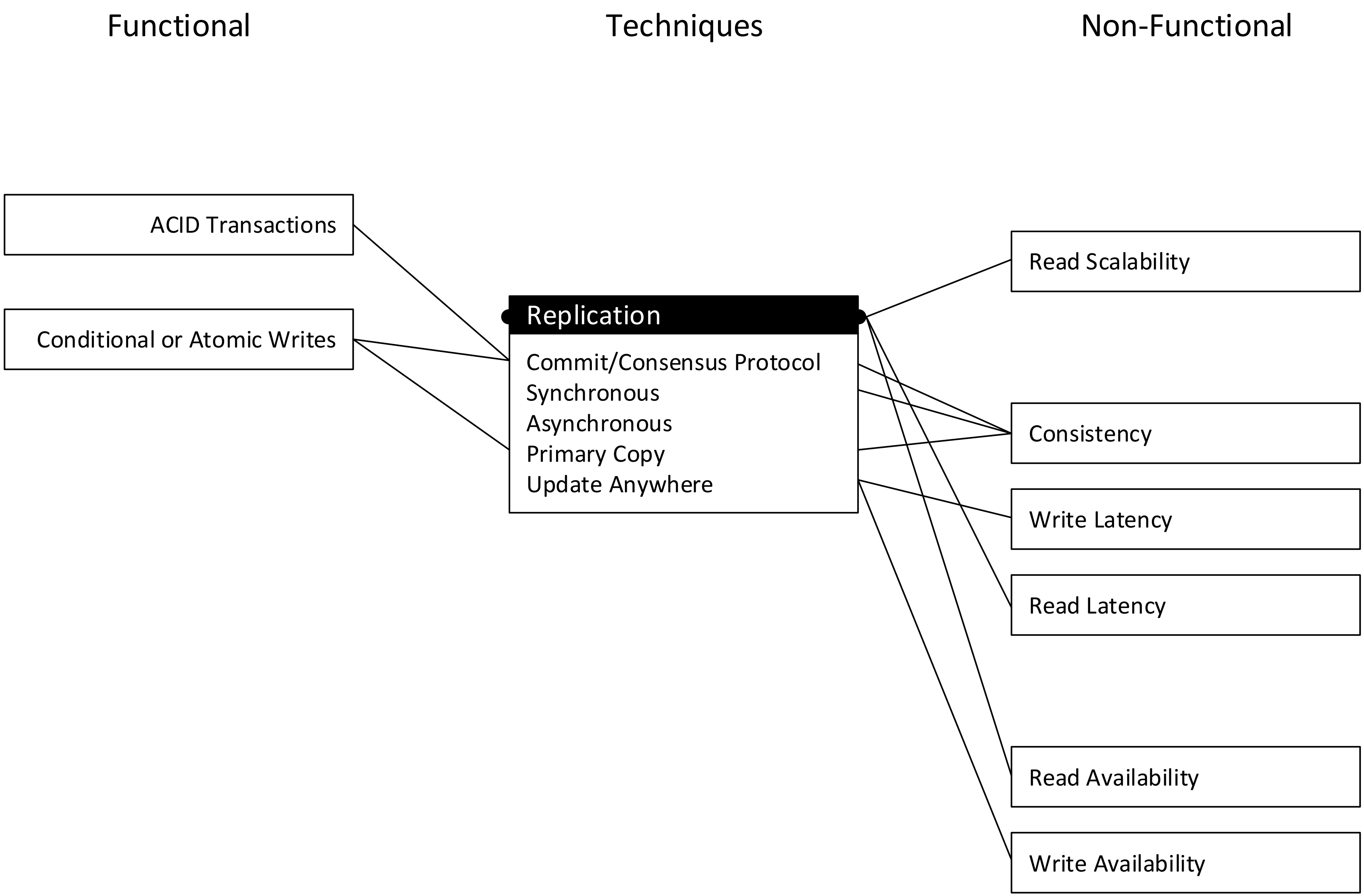
Example: **Tumblr**

- Caching
- Sharding from application

Moved towards:

- Redis
- HBase

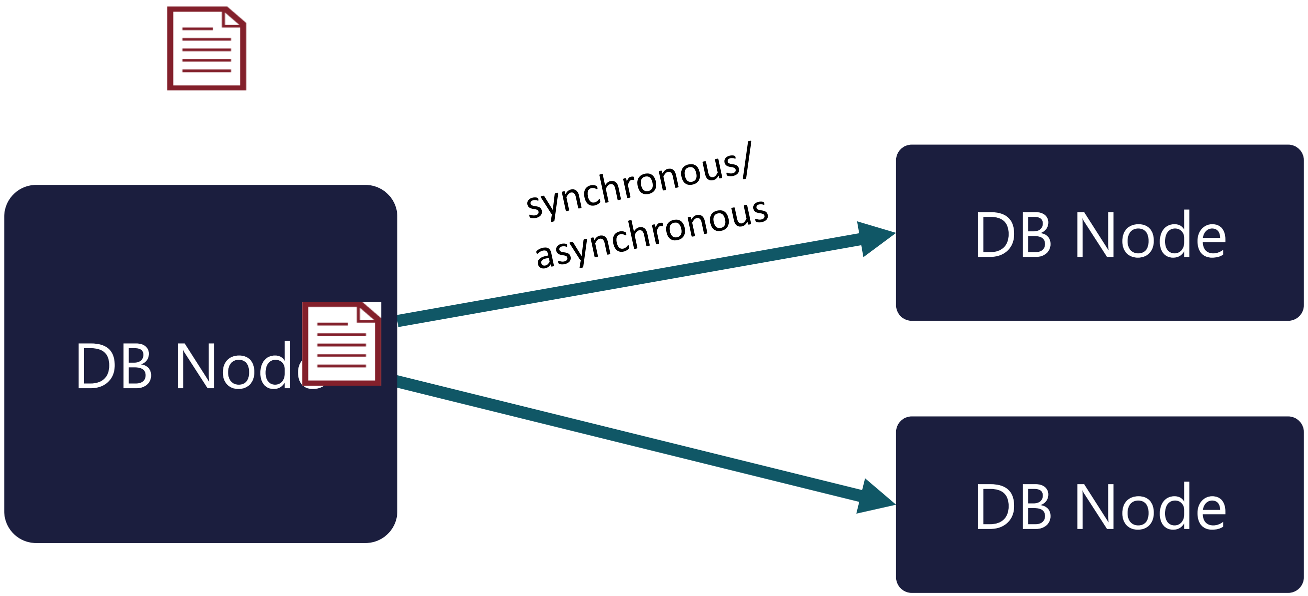




Functional
Techniques
Non-Functional
ACID Transactions
Conditional or Atomic Writes
Replication
Commit/Consensus Protocol
Synchronous
Asynchronous
Primary Copy
Update Anywhere
Read Scalability
Consistency
Write Latency
Read Latency
Read Availability
Write Availability

Replication

- Stores *N* copies of each data item



- **Consistency model**: synchronous vs asynchronous
- **Coordination**: Multi-Master, Master-Slave

Özsu, M.T., Valduriez, P.: Principles of distributed database systems. Springer Science & Business Media (2011)

Replication: When

Asynchronous (lazy)

- Writes are acknowledged imm
- Performed through *log shippi*
- **Pro**: Fast writes, no coordinati
- **Contra**: Replica data potential

Synchronous (eager)

- The node accepting writes syn
  updates/transactions before a
- **Pro**: Consistent
- **Contra**: needs a commit proto
  unavaialable under certain network partitions

Implemented in

Dynamo , Riak, CouchDB, Redis, Cassandra, Voldemort, MongoDB, RethinkDB

Implemented in

BigTable, HBase, Accumulo, CouchBase, MongoDB, RethinkDB

Charron-Bost, B., Pedone, F., Schiper, A. (eds.): Replication: Theory and Practice, Lecture Notes in Computer Science, vol. 5959. Springer (2010)

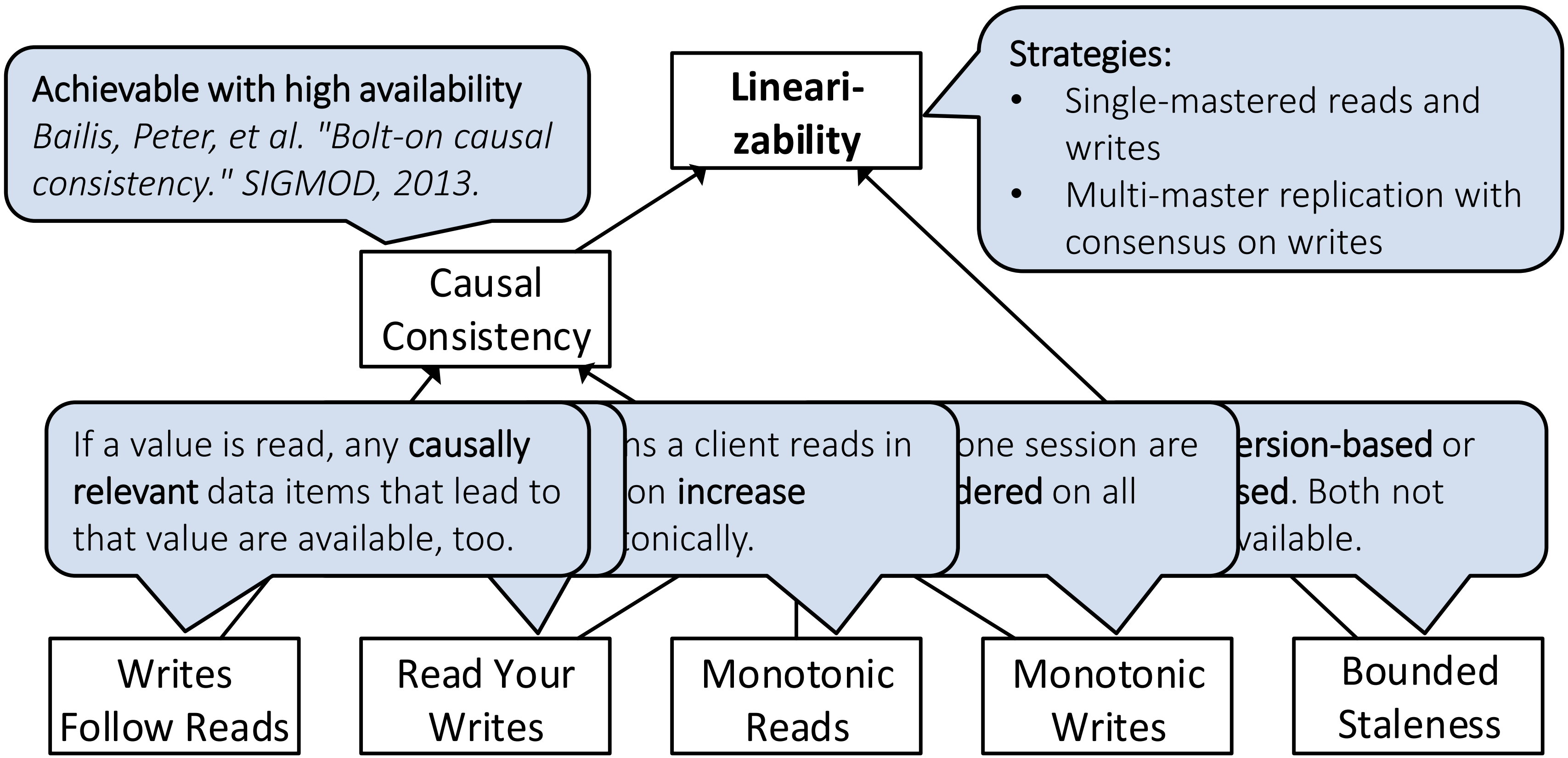
Replication: Where

Master-Slave (*Primary Copy*)
- Only a dedicated master is allowed to accept writes, slaves are read-replicas
- **Pro**: reads from the master are consistent
- **Contra**: master is a bottleneck and SPOF

Multi-Master (*Update anywhere*)
- The server node accepting the writes synchronously propagates the update or transaction before acknowledging
- **Pro**: fast and highly-available
- **Contra**: either needs coordination protocols (e.g. Paxos) or is inconsistent

Charron-Bost, B., Pedone, F., Schiper, A. (eds.): Replication: Theory and Practice, Lecture Notes in Computer Science, vol. 5959. Springer (2010)

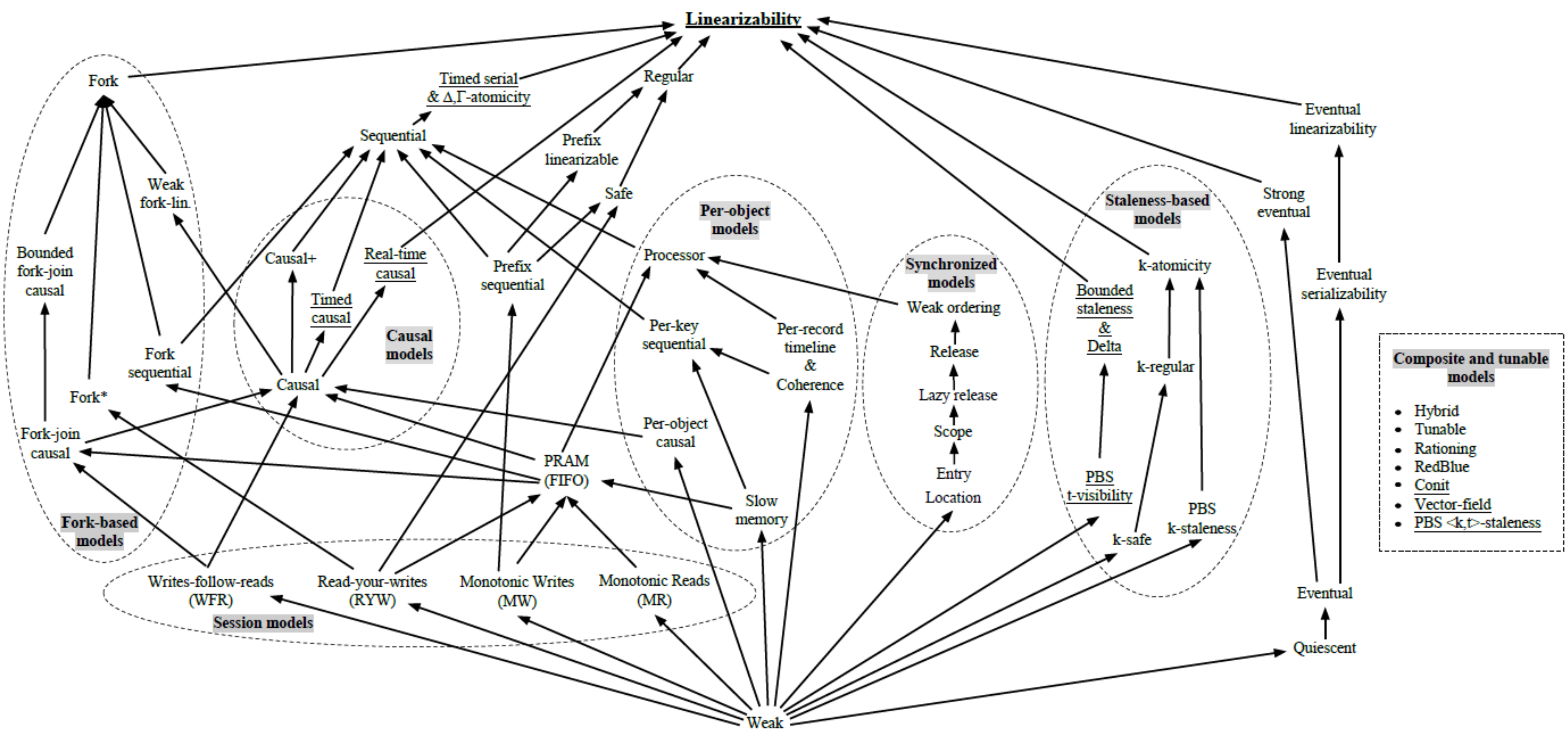
Consistency Levels



Viotti, Paolo, and Marko Vukolić. "Consistency in Non-Transactional Distributed Storage Systems." arXiv (2015).

Bailis, Peter, et al. "Highly available transactions: Virtues and limitations." Proceedings of the VLDB Endowment 7.3 (2013): 181-192.

Problem: Terminology



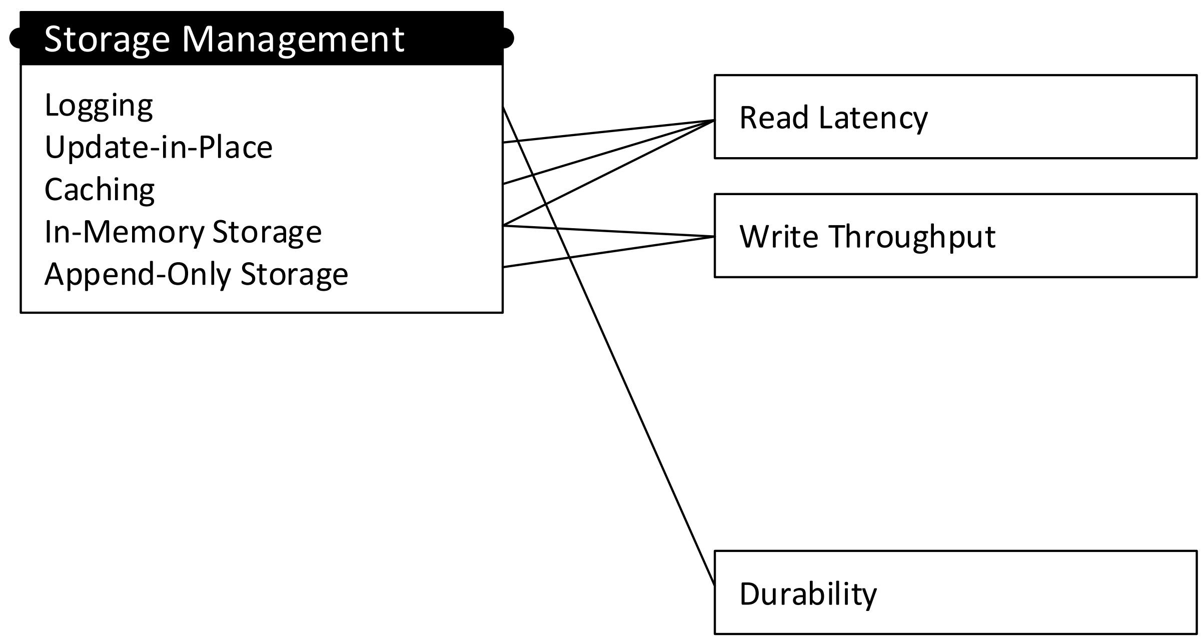
Viotti, Paolo, and Marko Vukolić. "Consistency in Non-Transactional Distributed Storage Systems." arXiv (2015).

Bailis, Peter, et al. "Highly available transactions: Virtues and limitations." Proceedings of the VLDB Endowment 7.3 (2013): 181-192.

Functional

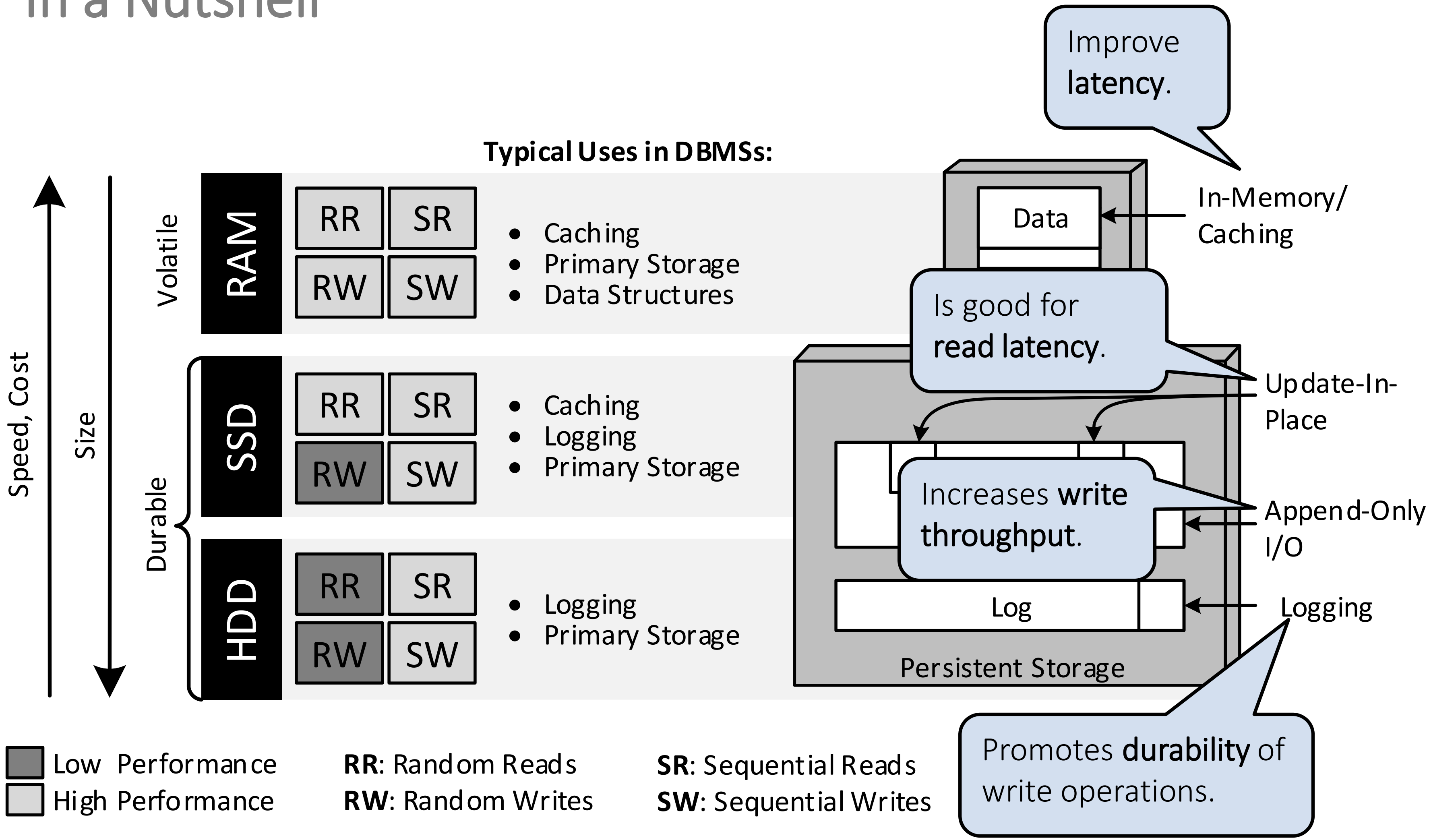
Techniques

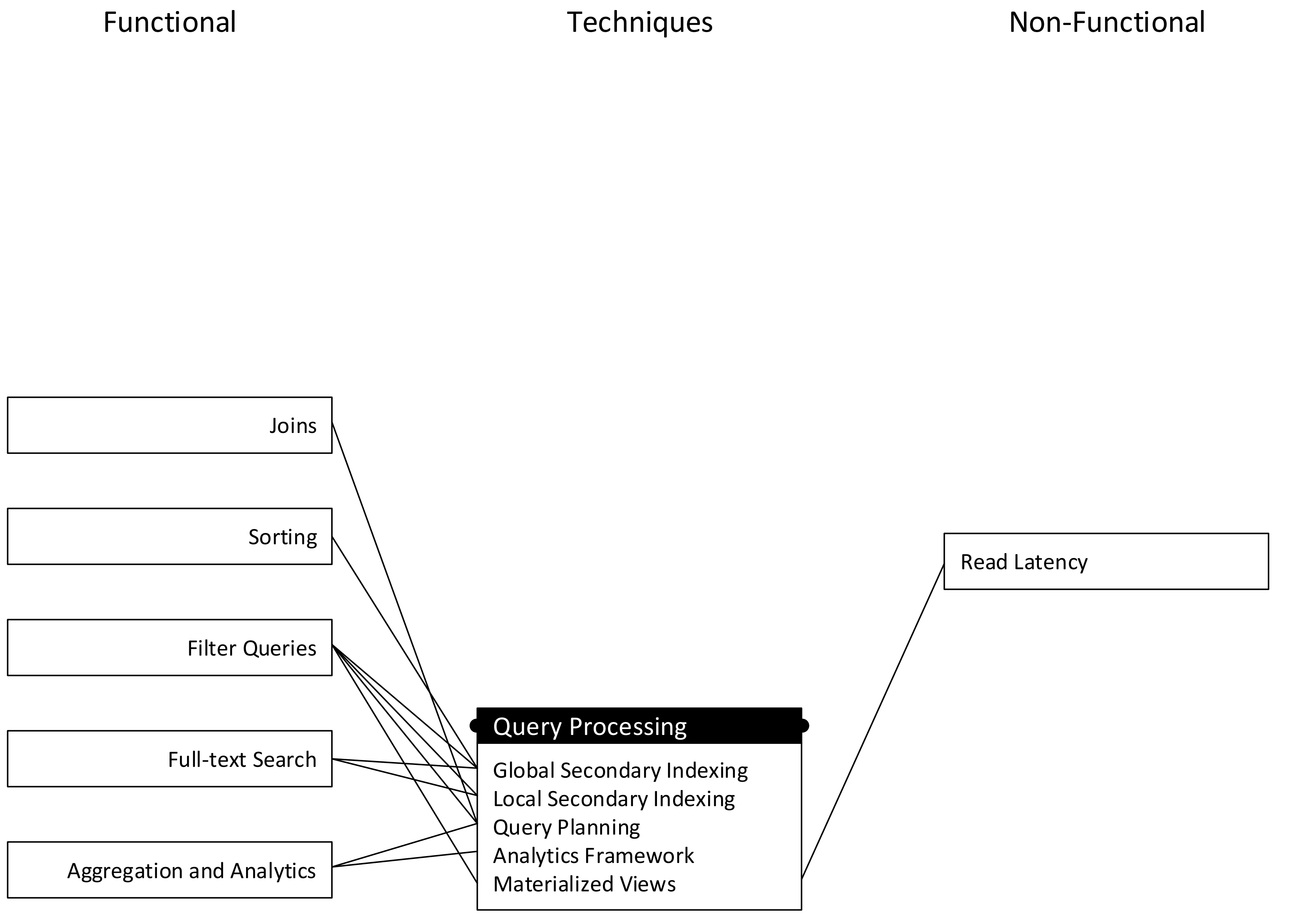
Non-Functional



NoSQL Storage Management

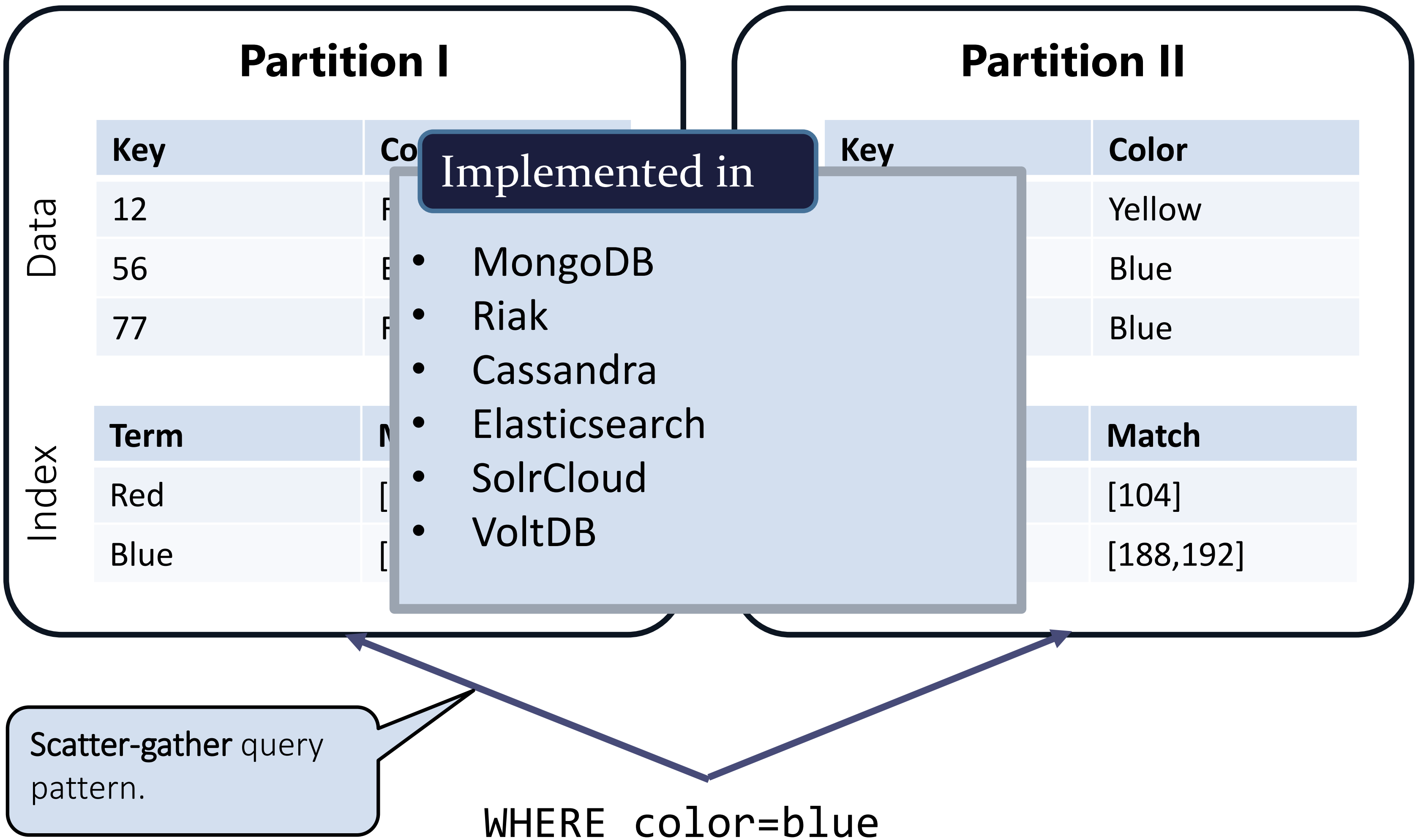
In a Nutshell




Functional
Techniques
Non-Functional
Joins
Sorting
Filter Queries
Full-text Search
Aggregation and Analytics
Query Processing
Global Secondary Indexing
Local Secondary Indexing
Query Planning
Analytics Framework
Materialized Views
Read Latency

Local Secondary Indexing

Partitioning By Document



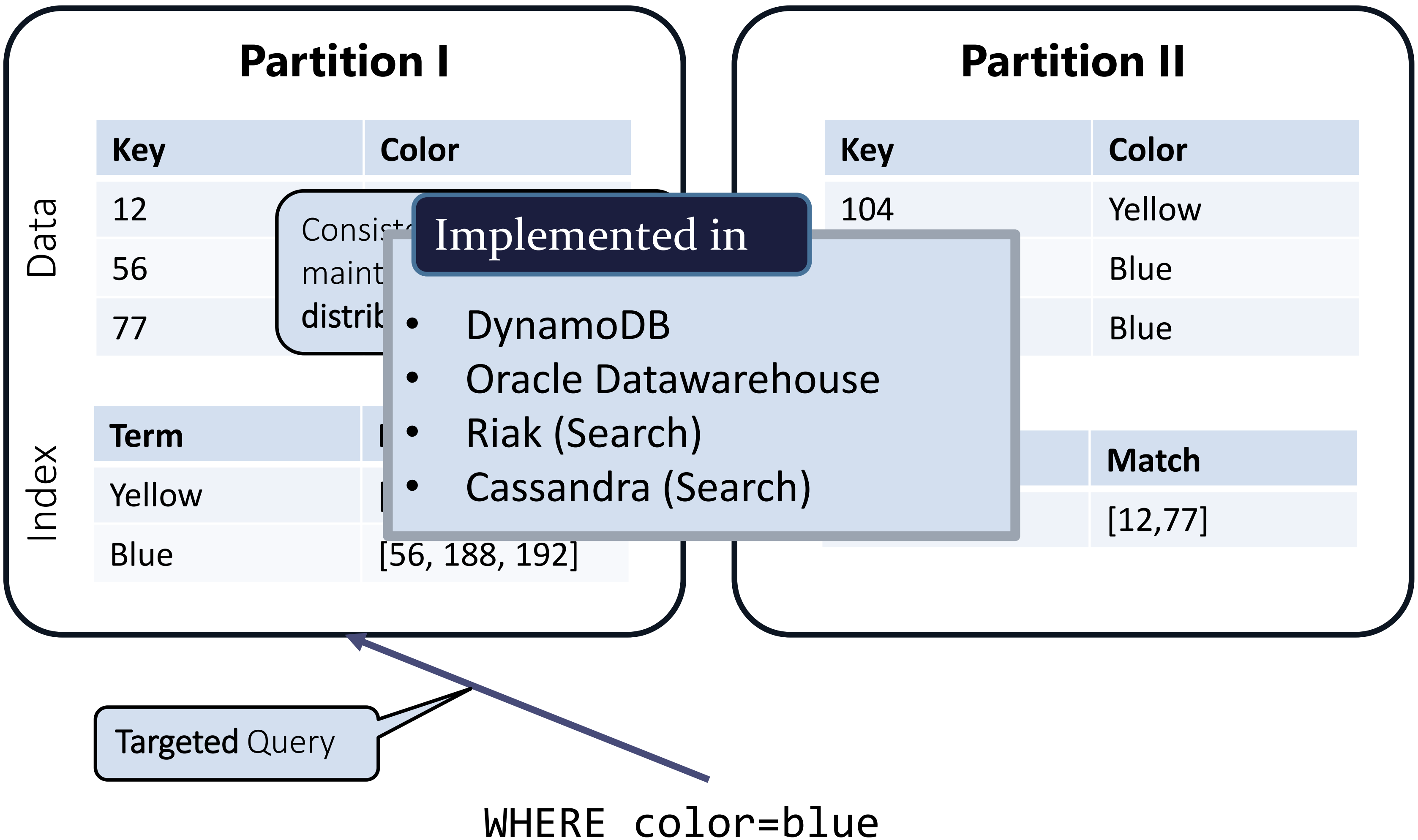
Implemented in

- MongoDB
- Riak
- Cassandra
- Elasticsearch
- SolrCloud
- VoltDB

Kleppmann, Martin. "Designing data-intensive applications." (2016).

Global Secondary Indexing

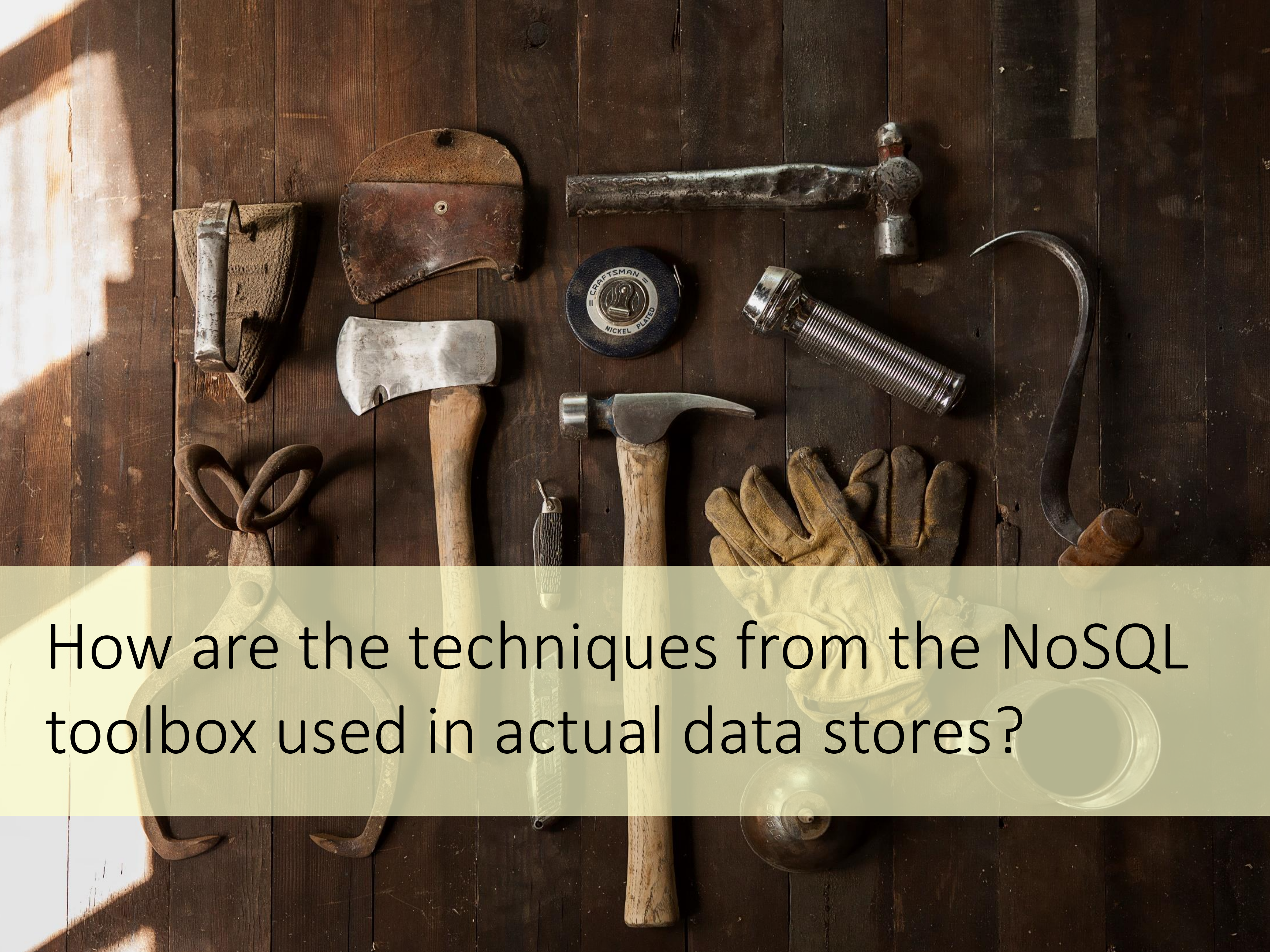
Partitioning By Term



Kleppmann, Martin. "Designing data-intensive applications." (2016).

Query Processing Techniques
Summary

- **Local Secondary Indexing:** Fast writes, scatter-gather queries
- **Global Secondary Indexing:** Slow or inconsistent writes, fast queries
- **(Distributed) Query Planning**: scarce in NoSQL systems but increasing (e.g. left-outer equi-joins in MongoDB and θ-joins in RethinkDB)
- **Analytics Frameworks**: fallback for missing query capabilities
- **Materialized Views**: similar to global indexing



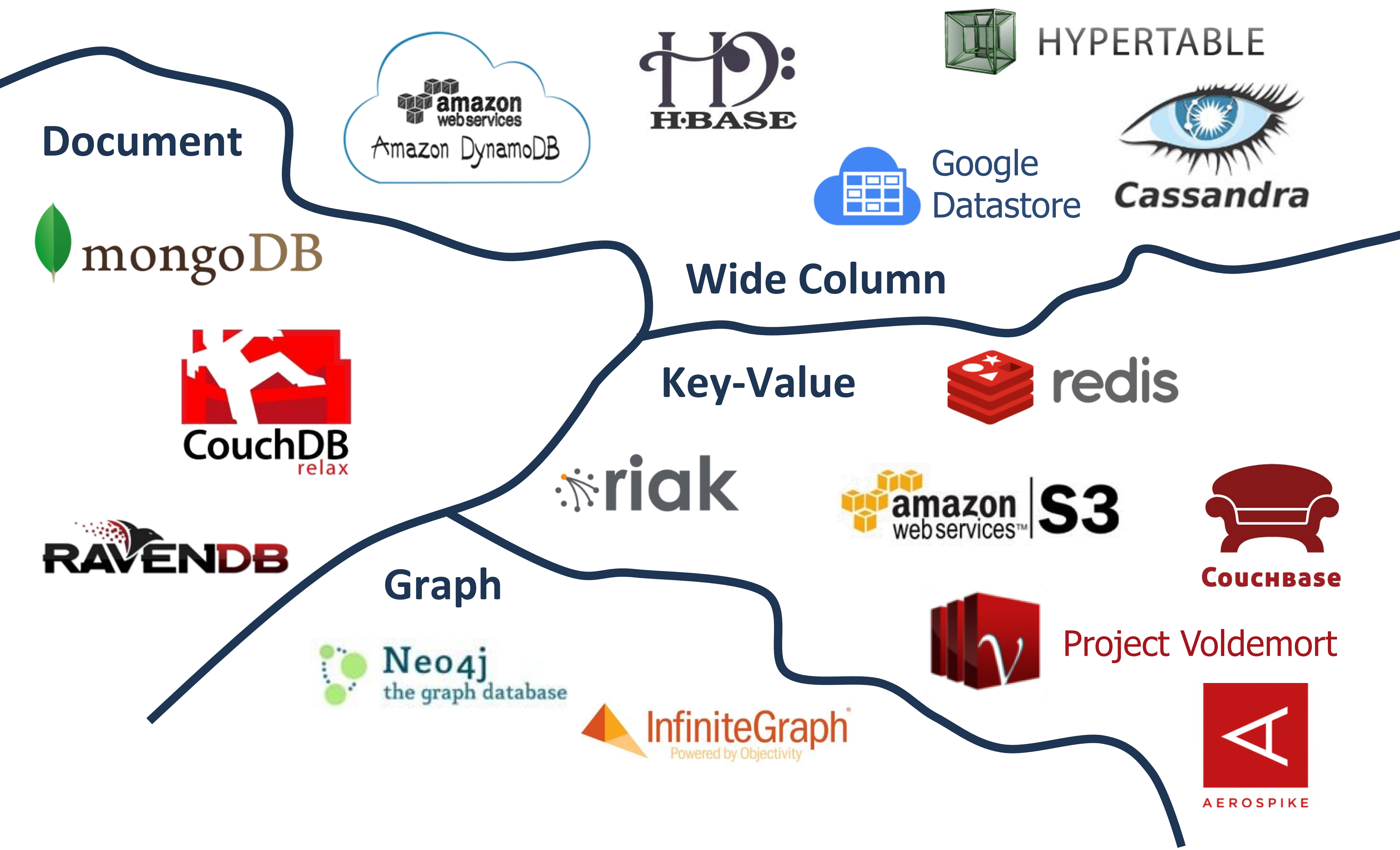
How are the techniques from the NoSQL toolbox used in actual data stores?

Outline

- NoSQL Foundations and Motivation
- The NoSQL Toolbox: Common Techniques
- NoSQL Systems
- Decision Guidance: NoSQL Decision Tree

- Overview & Popularity
- Core Systems:
  - Dynamo
  - BigTable
- Riak
- HBase
- Cassandra
- Redis
- MongoDB

NoSQL Landscape



Popularity

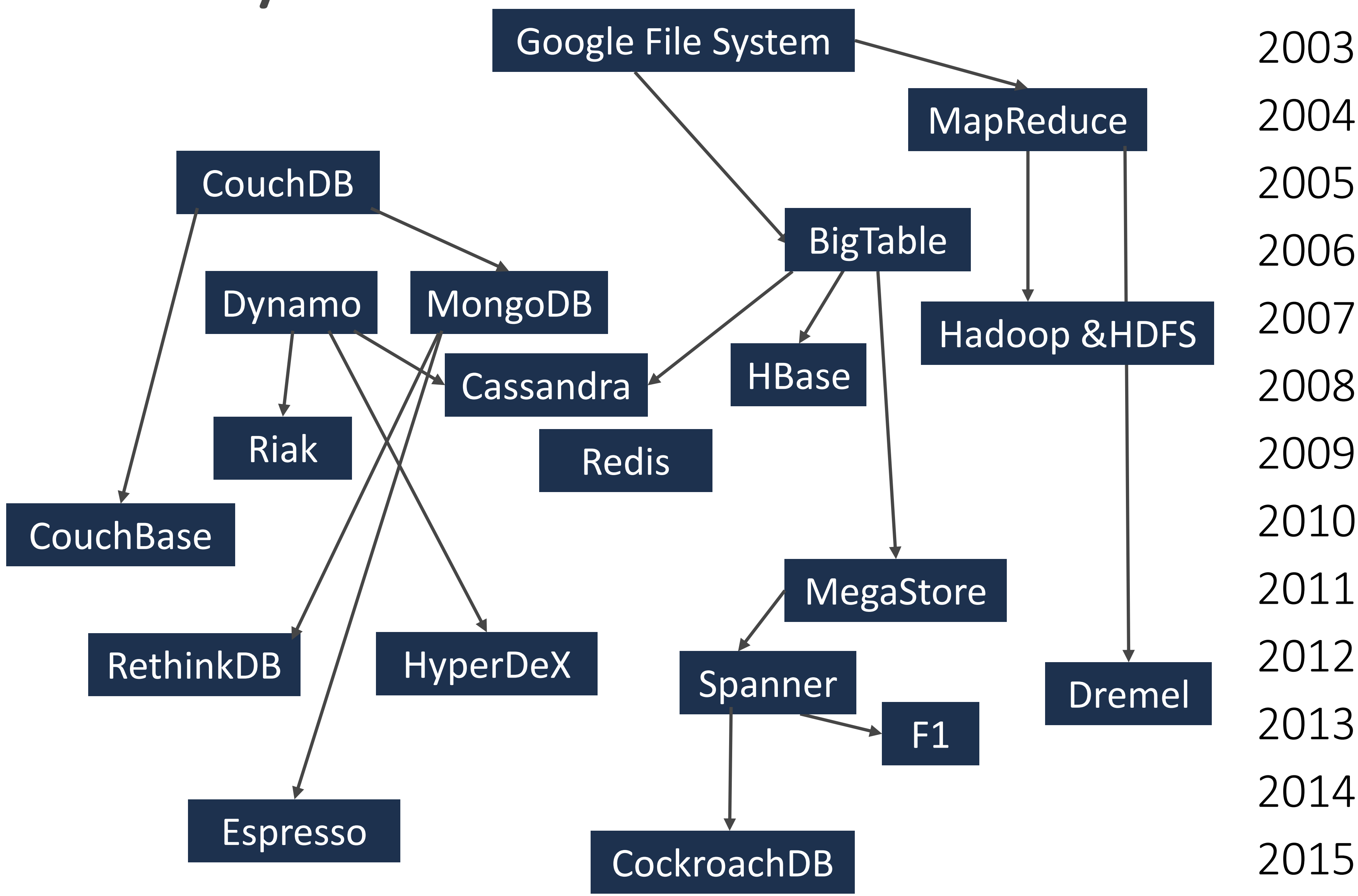
http://db-engines.com/de/ranking

#	System	Model	Score
1.	Oracle	Relational DBMS	1462.02
2.	MySQL	Relational DBMS	1371.83
3.	MS SQL Server	Relational DBMS	1142.82
4.	**MongoDB**	**Document store**	**320.22**
5.	PostgreSQL	Relational DBMS	307.61
6.	DB2	Relational DBMS	185.96
7.	**Cassandra**	**Wide column store**	**134.50**
8.	Microsoft Access	Relational DBMS	131.58
9.	**Redis**	**Key-value store**	**108.24**
10.	SQLite	Relational DBMS	107.26

11.	**Elasticsearch**	**Search engine**	**86.31**
12.	Teradata	Relational DBMS	73.74
13.	SAP Adaptive Server	Relational DBMS	71.48
14.	**Solr**	**Search engine**	**65.62**
15.	**HBase**	**Wide column store**	**51.84**
16.	Hive	Relational DBMS	47.51
17.	FileMaker	Relational DBMS	46.71
18.	**Splunk**	**Search engine**	**44.31**
19.	SAP HANA	Relational DBMS	41.37
20.	MariaDB	Relational DBMS	33.97
21.	**Neo4j**	**Graph DBMS**	**32.61**
22.	Informix	Relational DBMS	30.58
23.	**Memcached**	**Key-value store**	**27.90**
24.	**Couchbase**	**Document store**	**24.29**
25.	**Amazon DynamoDB**	**Multi-model**	**23.60**

Scoring: Google/Bing results, Google Trends, Stackoverflow, job offers, LinkedIn

History



Google File System
MapReduce
CouchDB
BigTable
Dynamo
MongoDB
Hadoop &HDFS
Cassandra
HBase
Riak
Redis
CouchBase
MegaStore
RethinkDB
HyperDeX
Spanner
Dremel
F1
Espresso
CockroachDB
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

NoSQL foundations

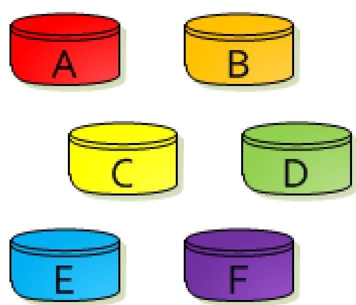
- **BigTable** (2006, Google)
  - Consistent, Partition Tolerant
  - Wide-Column data model
  - Master-based, fault-tolerant, large clusters (1.000+ Nodes), **HBase**, **Cassandra**, **HyperTable, Accumolo**
- **Dynamo** (2007, Amazon)
  - Available, Partition tolerant
  - Key-Value interface
  - Eventually Consistent, always writable, fault-tolerant
  - **Riak**, **Cassandra**, **Voldemort**, **DynamoDB**

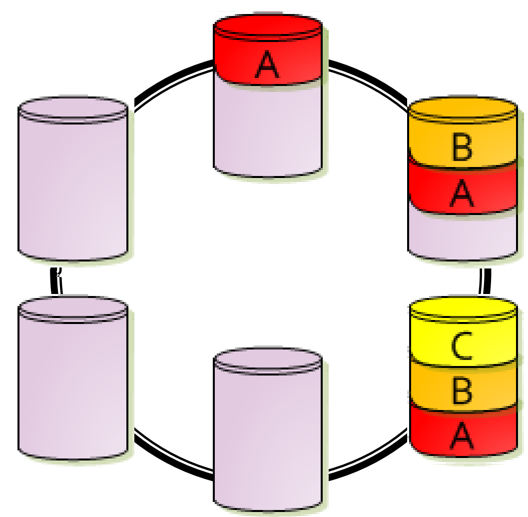
Chang, Fay, et al. "Bigtable: A distributed storage system for structured data."

DeCandia, Giuseppe, et al. "Dynamo: Amazon's highly available key-value store."

Dynamo (AP)

- Developed at Amazon (2007)
- Sharding of data over a ring of nodes
- Each node holds multiple partitions
- Each partition replicated **N** times

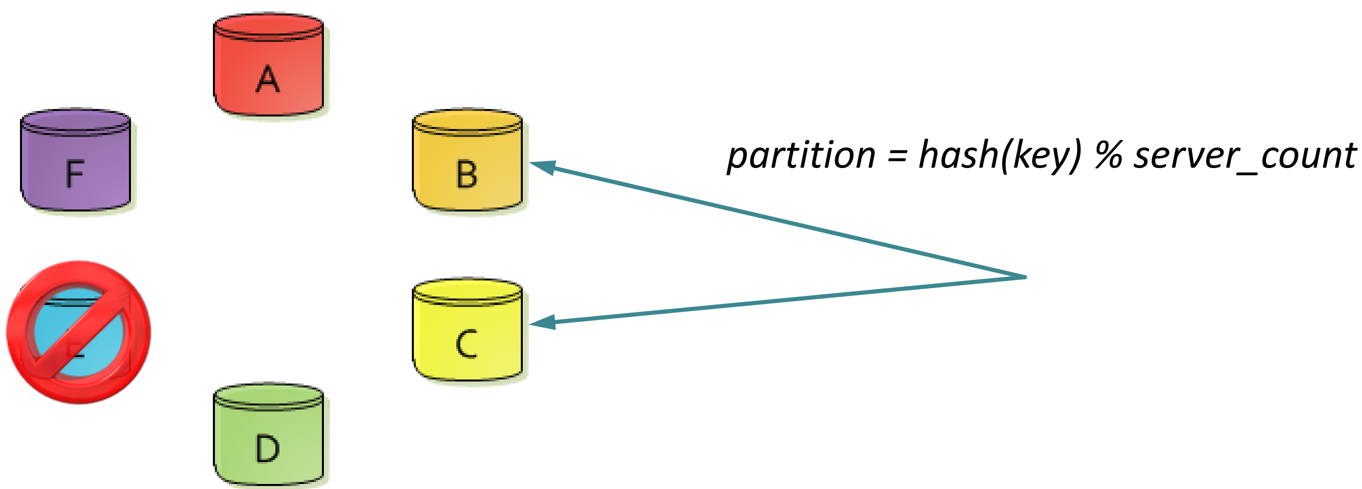




DeCandia, Giuseppe, et al. "Dynamo: Amazon's highly available key-value store."

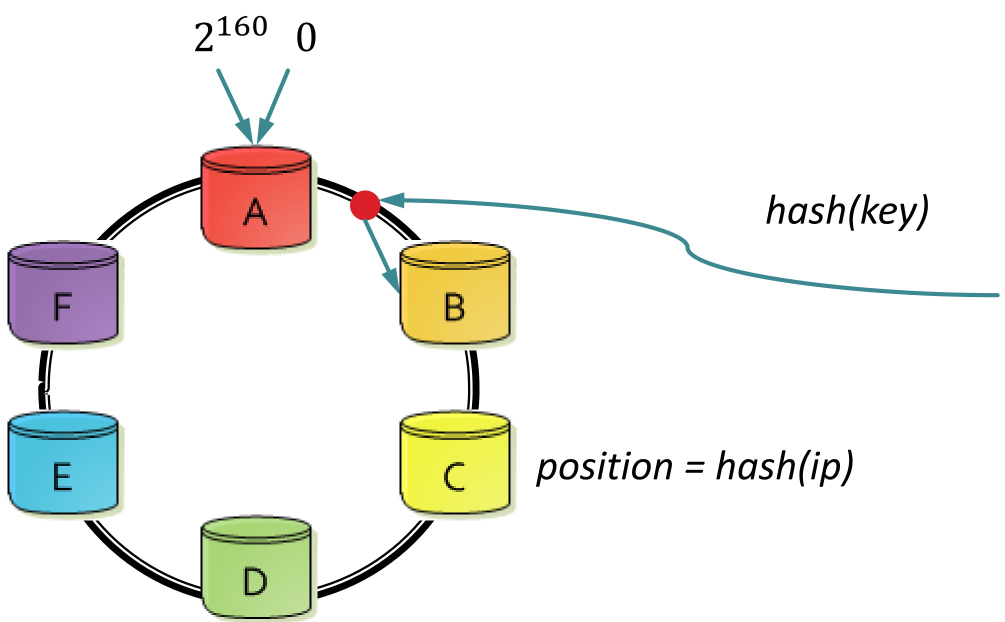
Consistent Hashing

- Naive approach: **Hash-partitioning** (e.g. in Memcache, Redis Cluster)



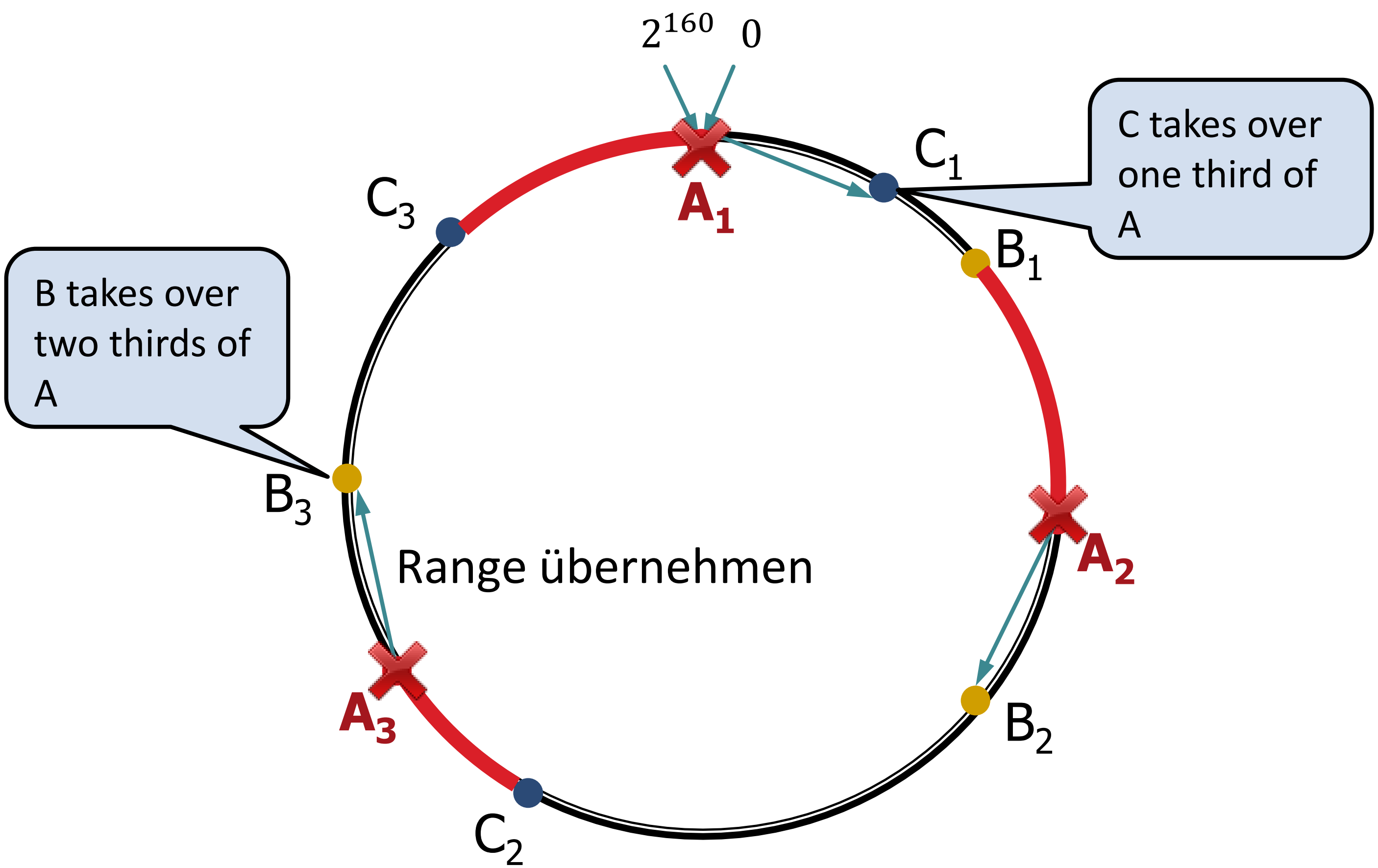
Consistent Hashing

- Solution: **Consistent Hashing** – mapping of data to nodes is stable under topology changes



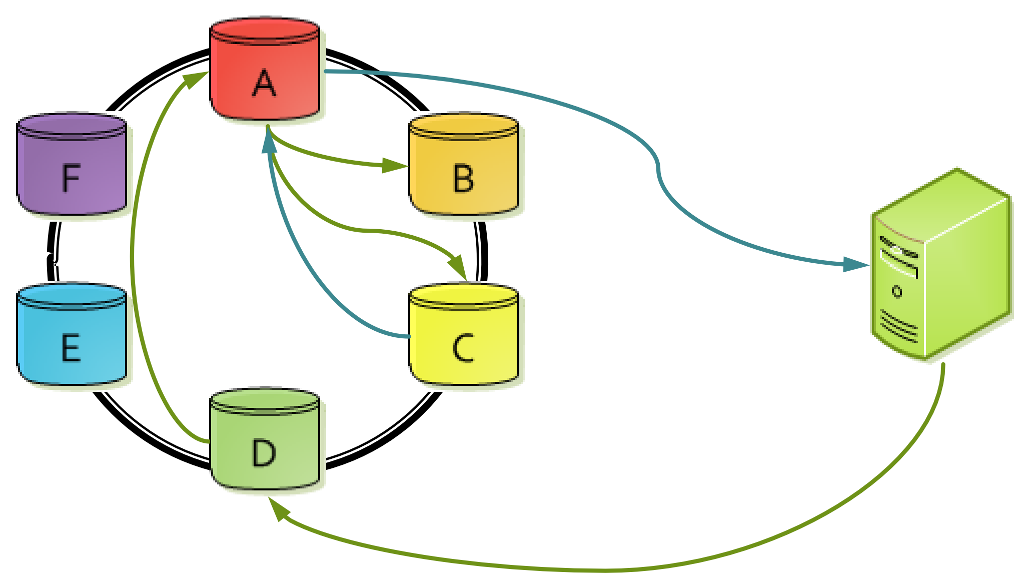
Consistent Hashing

- Extension: **Virtual Nodes** for Load Balancing



Reading

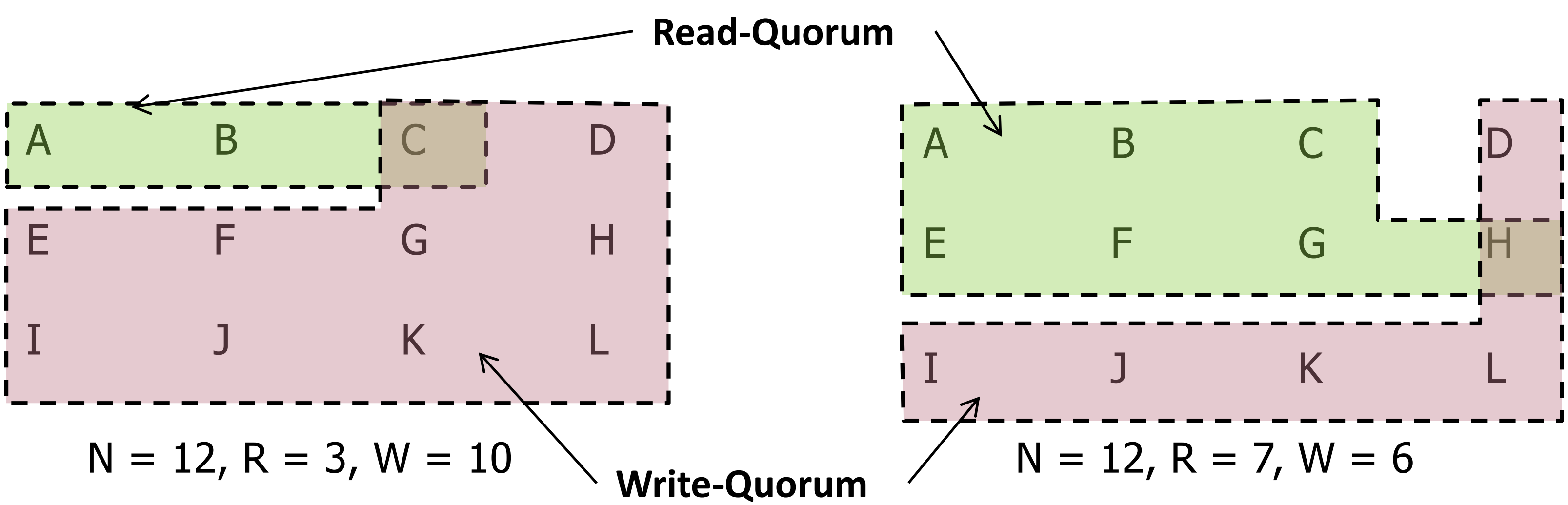
- An arbitrary node acts as a coordinator
- **N**: number of replicas
- **R**: number of nodes that need to confirm a read
- **W**: number of nodes that need to confirm a write



N=3
R=2
W=1

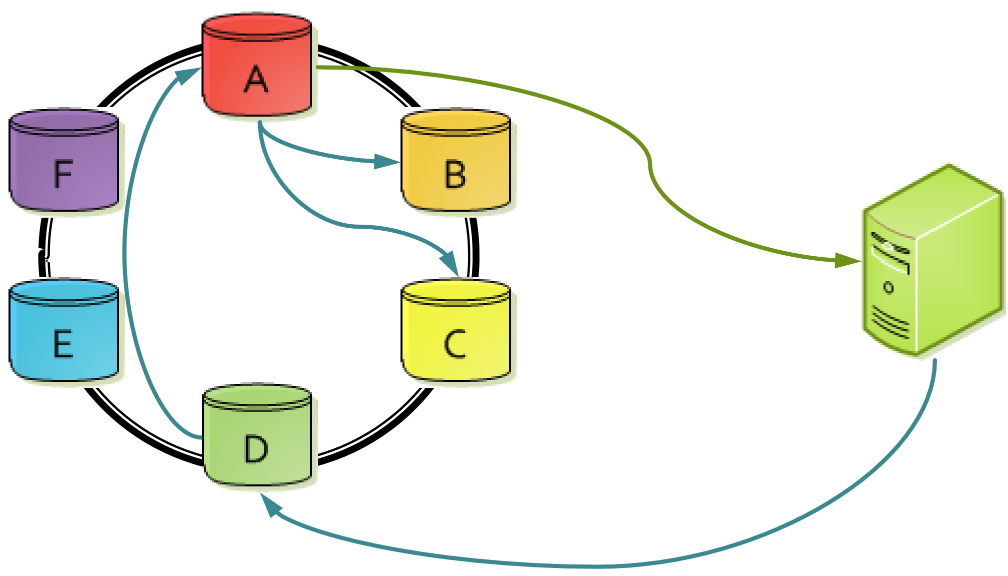
Quorums

- **N** (Replicas), **W** (Write Acks), **R** (Read Acks)
  - $R + W \leq N \Rightarrow$ No guarantee
  - $R + W > N \Rightarrow$ newest version included



Writing

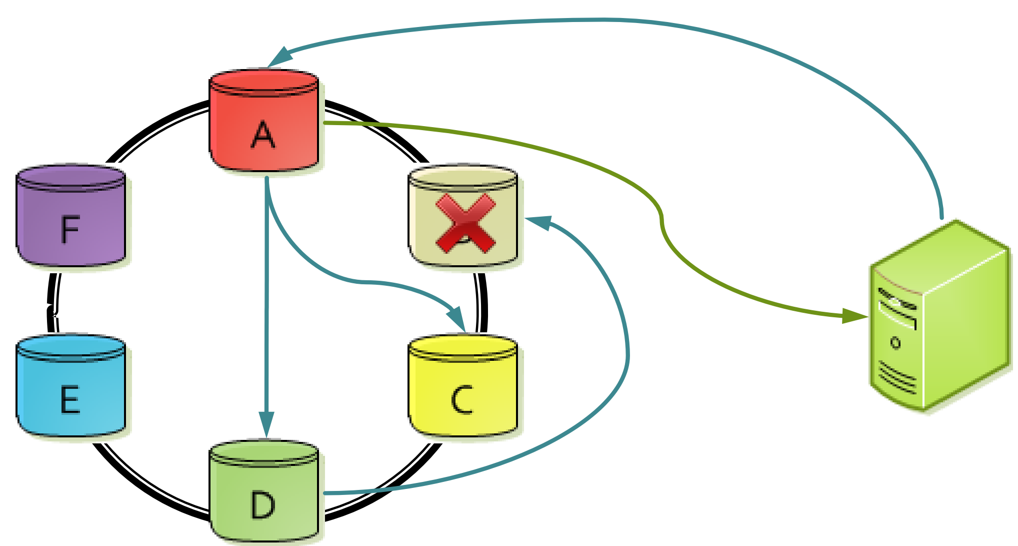
- **W** Servers have to ack



N=3
R=2
W=1

Hinted Handoff

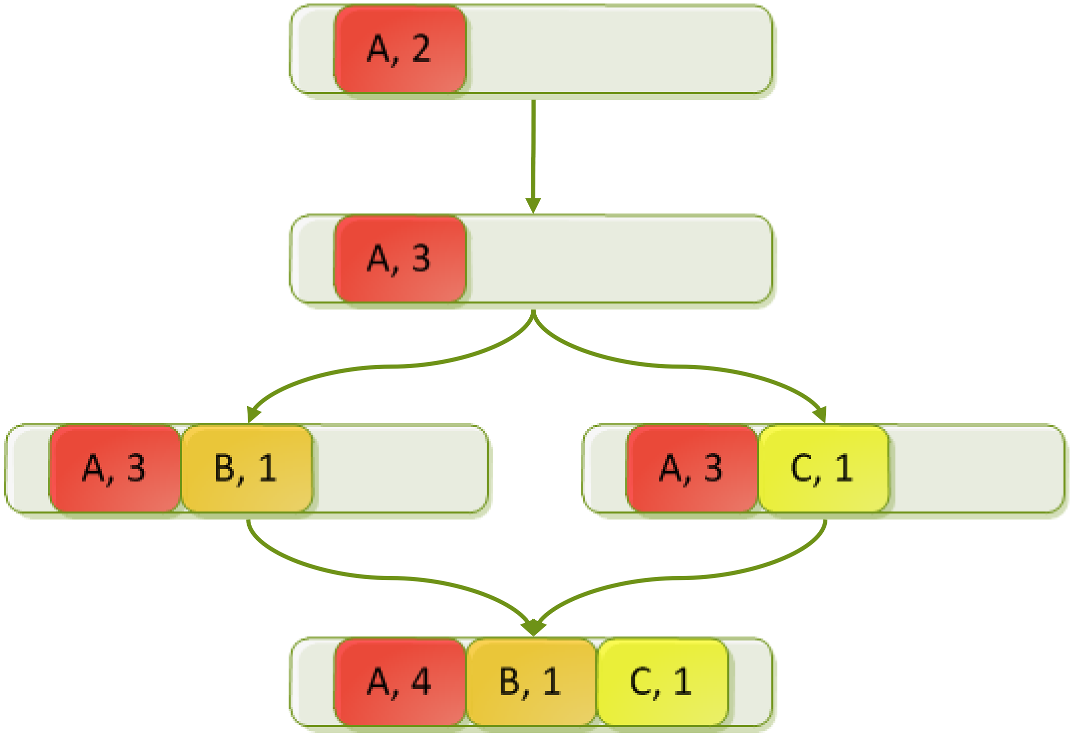
- Next node in the ring may take over, until original node is available again:



N=3
R=2
W=1

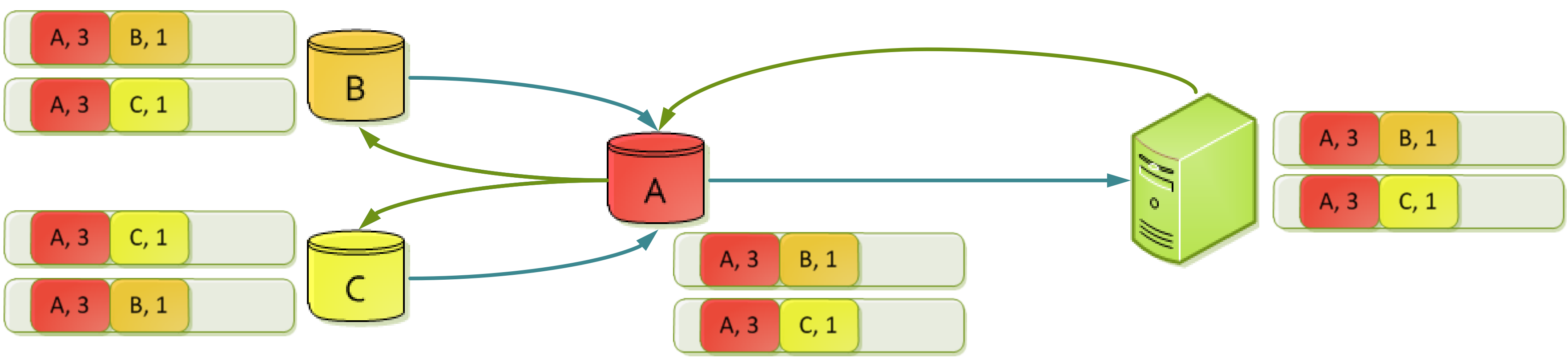
Vector clocks

- Dynamo uses **Vector Clocks** for versioning



Versioning and Consistency

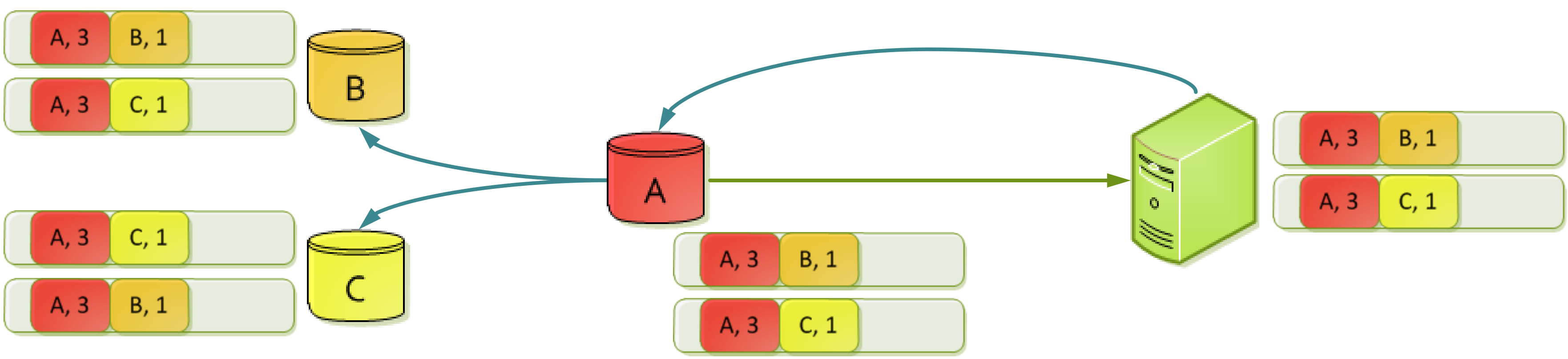
- $R + W \leq N \Rightarrow$ no consistency guarantee
- $R + W > N \Rightarrow$ newest acked value included in reads
- **Vector Clocks** used for versioning



Read Repair

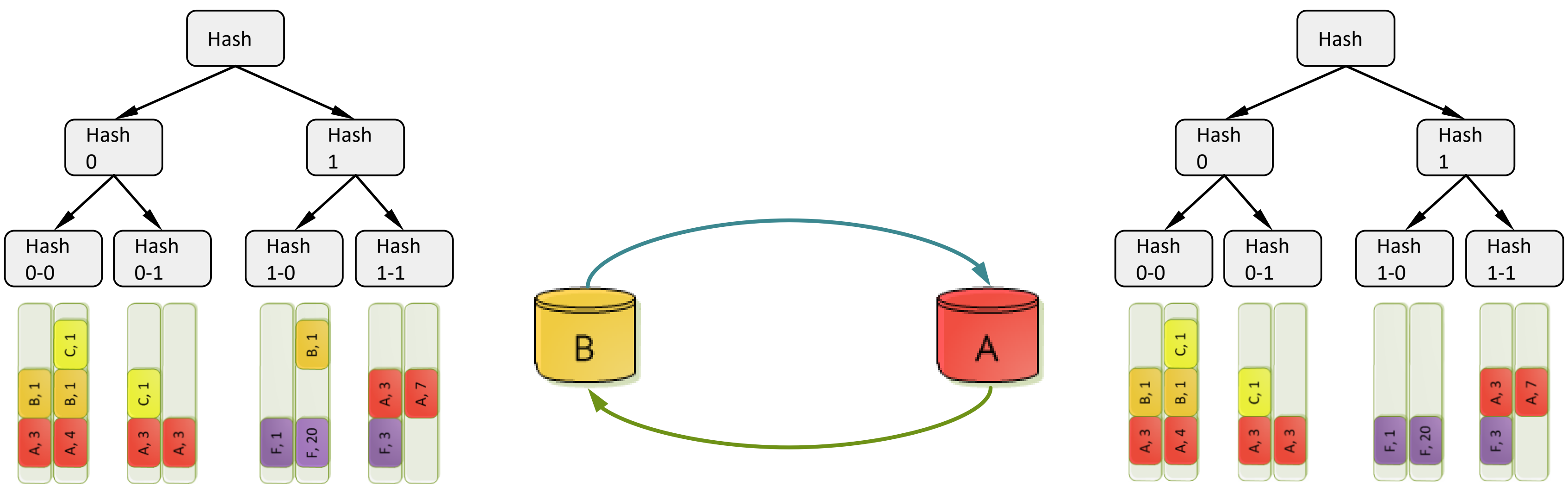
Conflict Resolution

- The application merges data when writing (*Semantic Reconciliation*)



Merkle Trees: Anti-Entropy

- Every Second: Contact random server and compare



Quorum

- Typical Configurations:

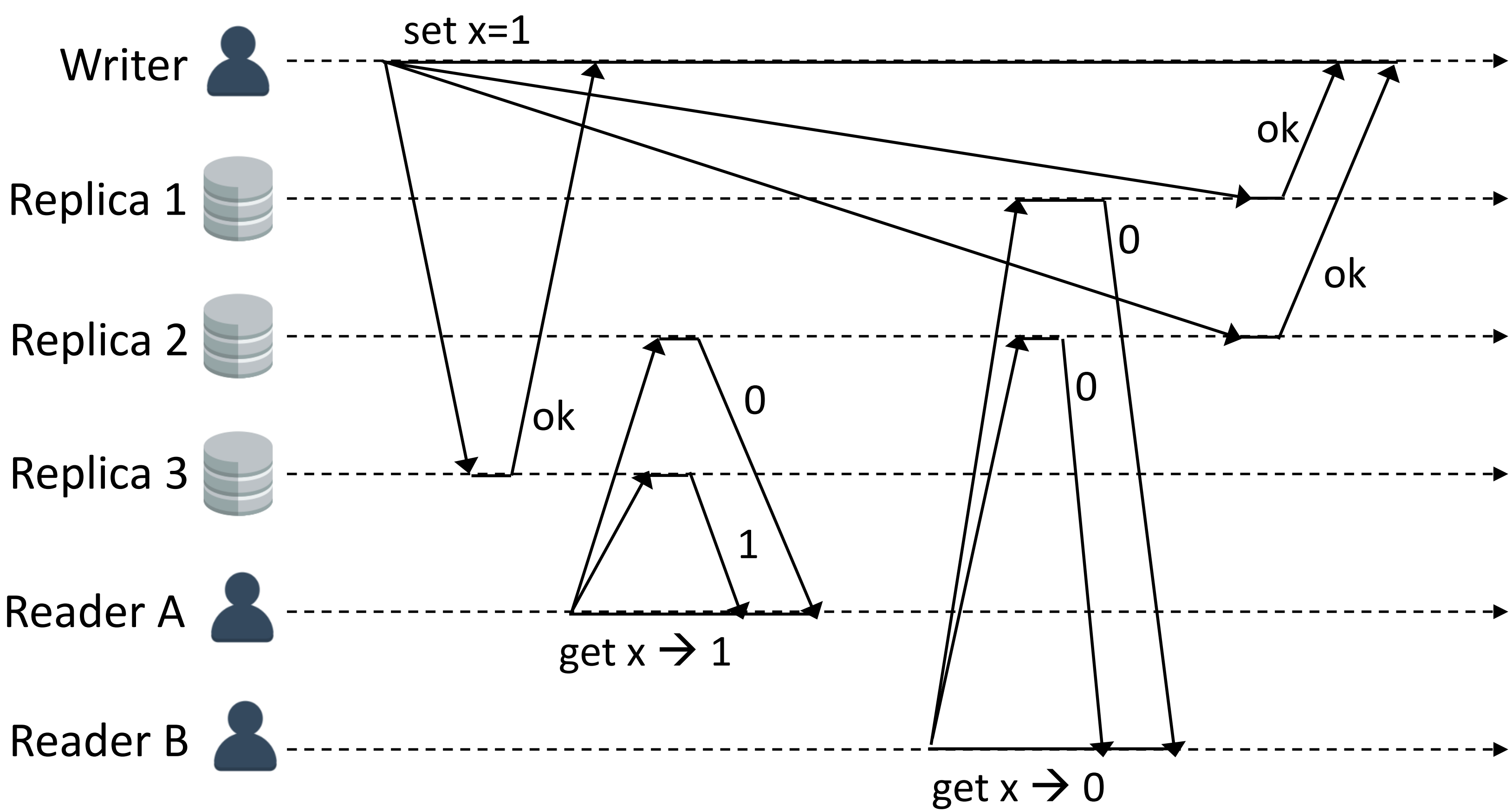
LinkedIn (SSDs):
$P(consistent) \geq 99.9\%$
nach $1.85\ ms$

Performance (Cassandra Default)	N=3, R=1, W=1
Quorum, fast Writing:	N=3, R=3, W=1
Quorum, fast Reading	N=3, R=1, W=3
Trade-off (Riak Default)	N=3, R=2, W=2

P. Bailis, PBS Talk: http://www.bailis.org/talks/twitter-pbs.pdf

$R + W > N$ does not imply linearizability

- Consider the following execution:

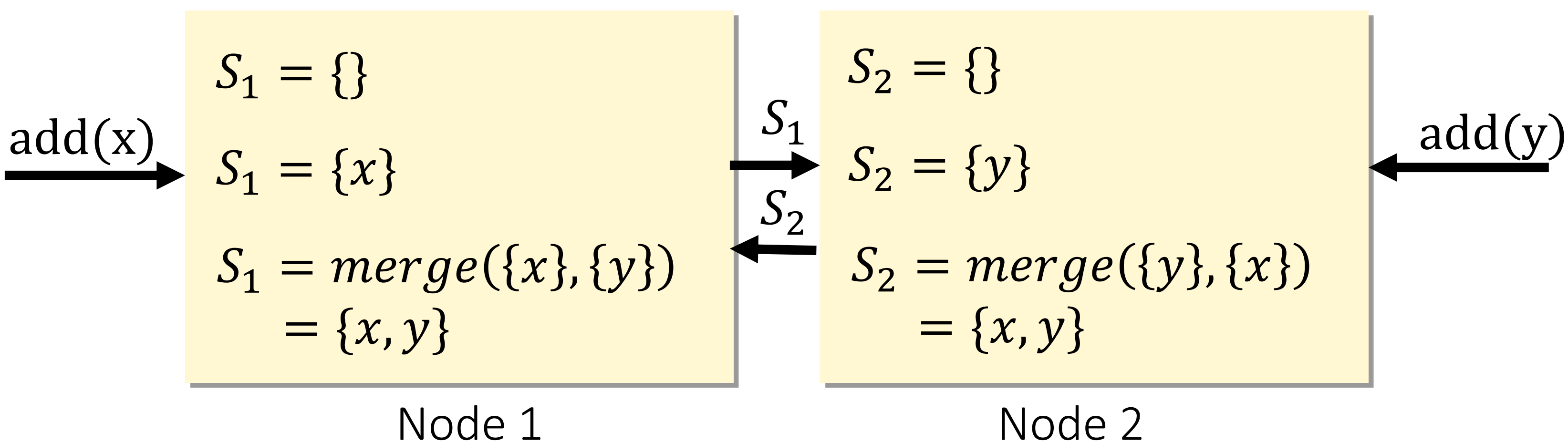


Kleppmann, Martin. "Designing data-intensive applications." (2016).

CRDTs

Convergent/Commutative Replicated Data Types

- **Goal**: avoid manual conflict-resolution
- **Approach**:
  - **State-based** – commutative, idempotent merge function
  - **Operation-based** – broadcasts of commutative upates
- Example: State-based Grow-only-Set (G-Set)

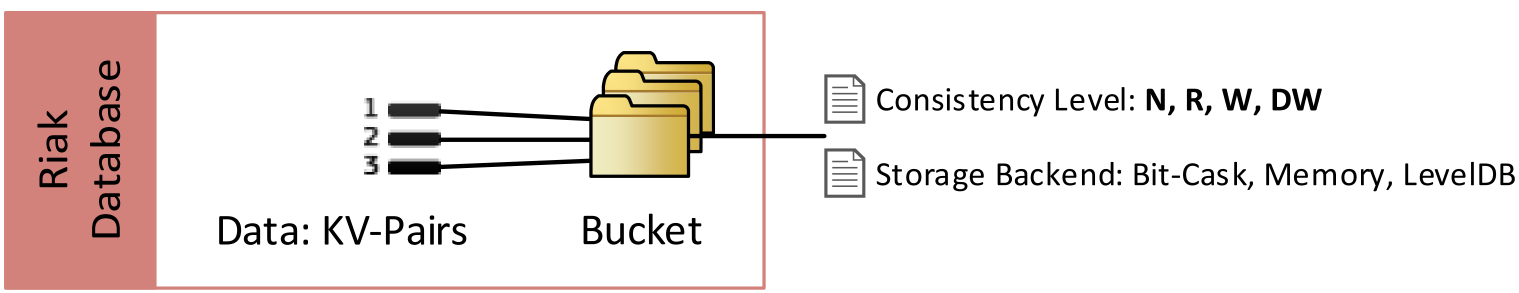


Marc Shapiro, Nuno Preguica, Carlos Baquero, and Marek Zawirski "Conflict-free Replicated Data Types"

Riak (AP)

- Open-Source Dynamo-Implementation
- Extends Dynamo:
  - Keys are grouped to **Buckets**
  - KV-pairs may have **metadata** and **links**
  - Map-Reduce support
  - Secondary Indices, Update Hooks, Solr Integration
  - **Riak CS:** S3-like file storage, **Riak TS**: time-series database

Riak
Model:
Key-Value
License:
Apache 2
Written in:
Erlang und C



Riak Data Types

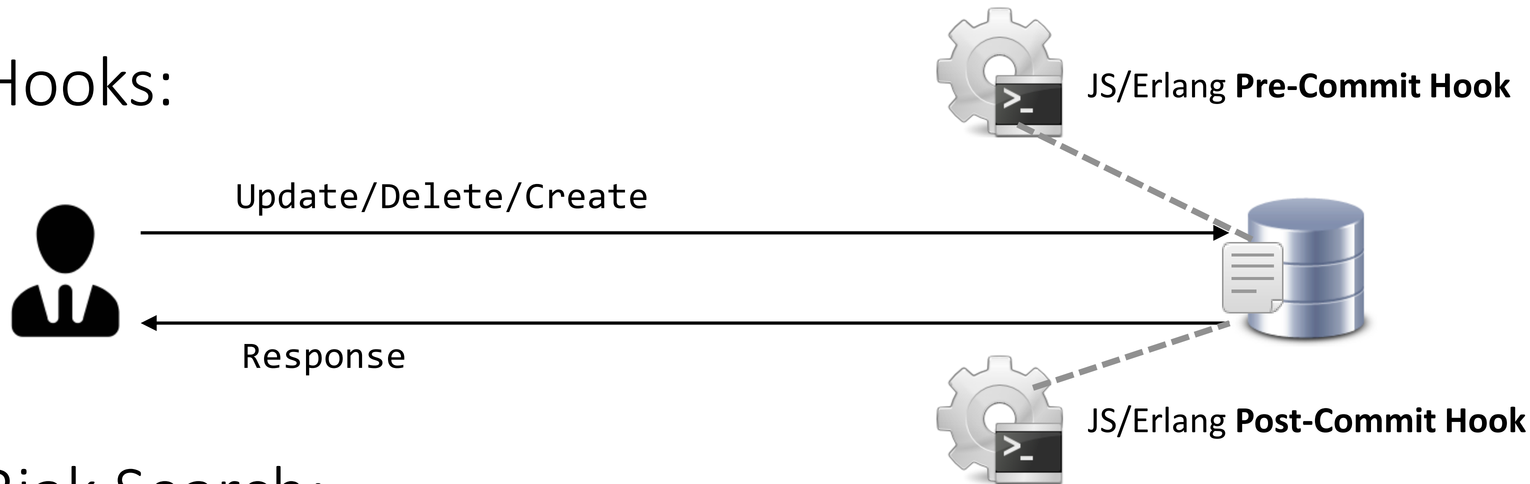
- Implemented as *state-based CRDTs*:

Data Type	Convergence rule
Flags	enable wins over disable
Registers	The most chronologically recent value wins, based on timestamps
Counters	Implemented as a PN-Counter, so all increments and decrements are eventually applied.
Sets	If an element is concurrently added and removed, the add will win
Maps	If a field is concurrently added or updated and removed, the add/update will win

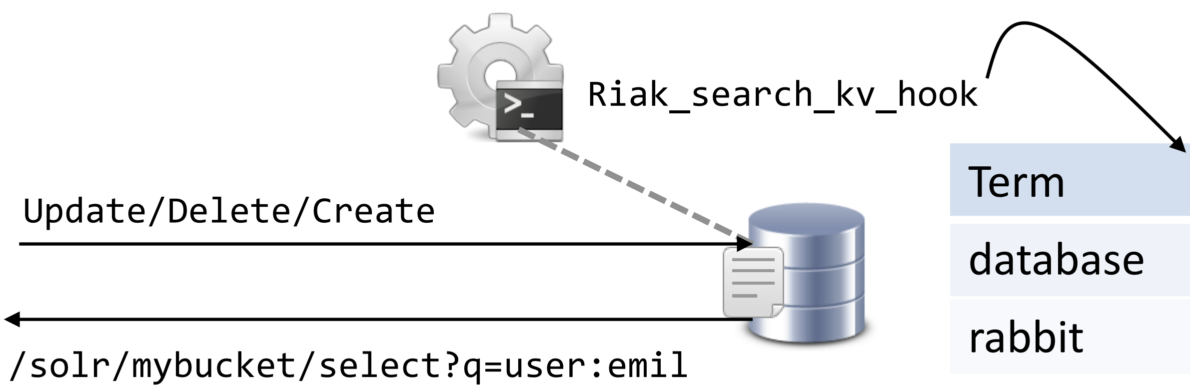
http://docs.basho.com/riak/kv/2.1.4/learn/concepts/crdts/

Hooks & Search

- Hooks:



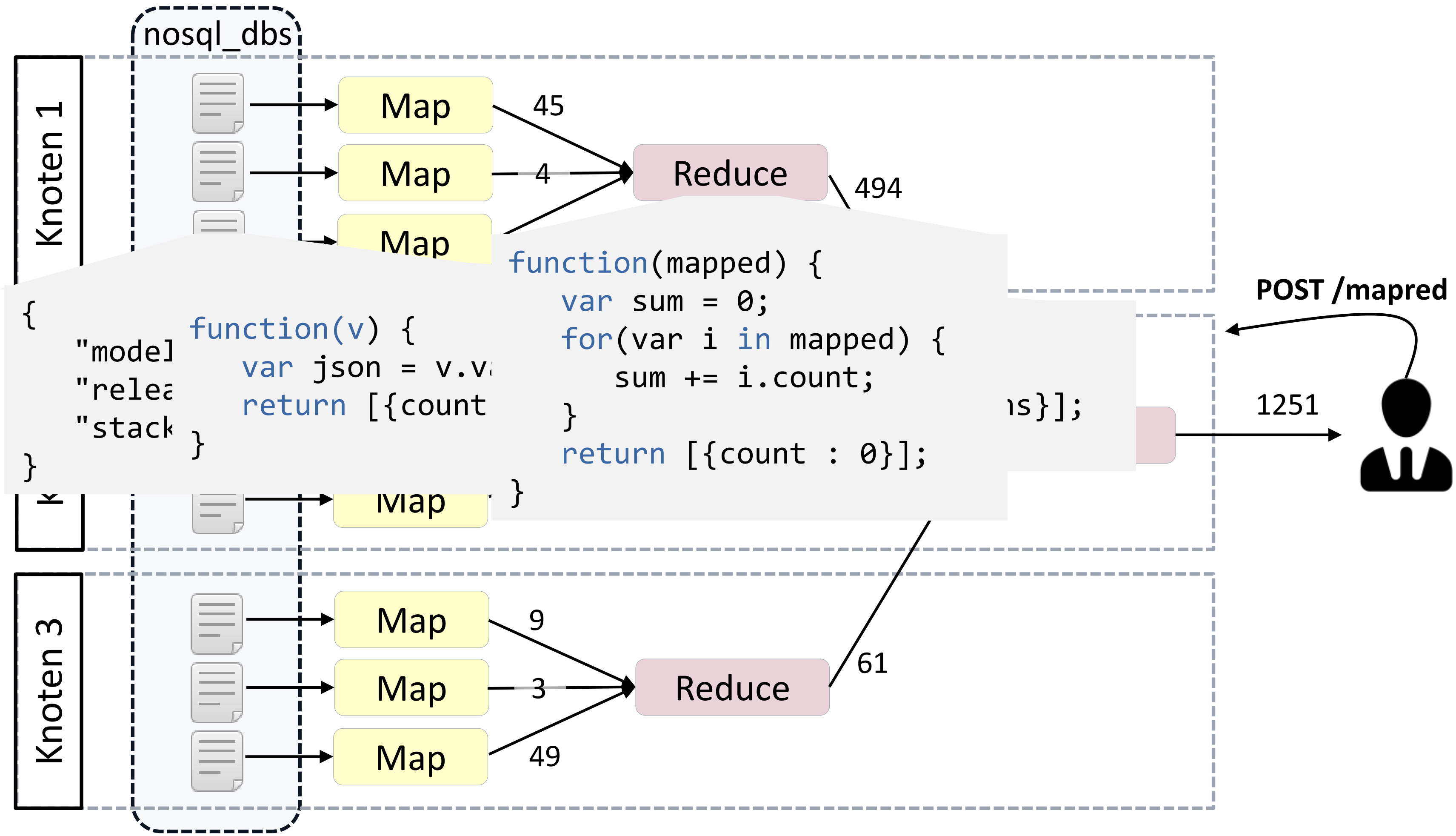
- Riak Search:



Term	Dokument
database	3,4,1
rabbit	2

Search Index

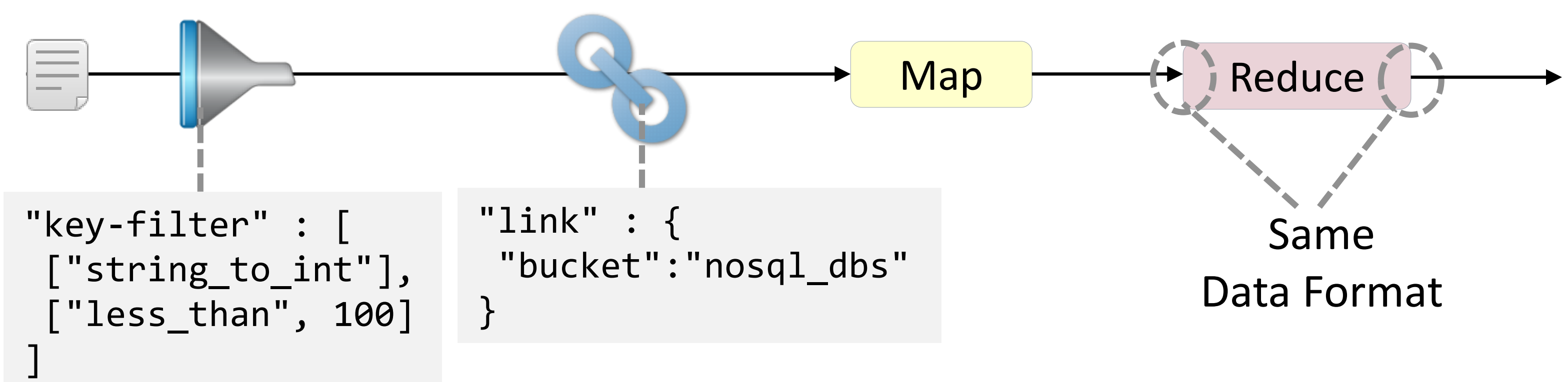
Riak Map-Reduce



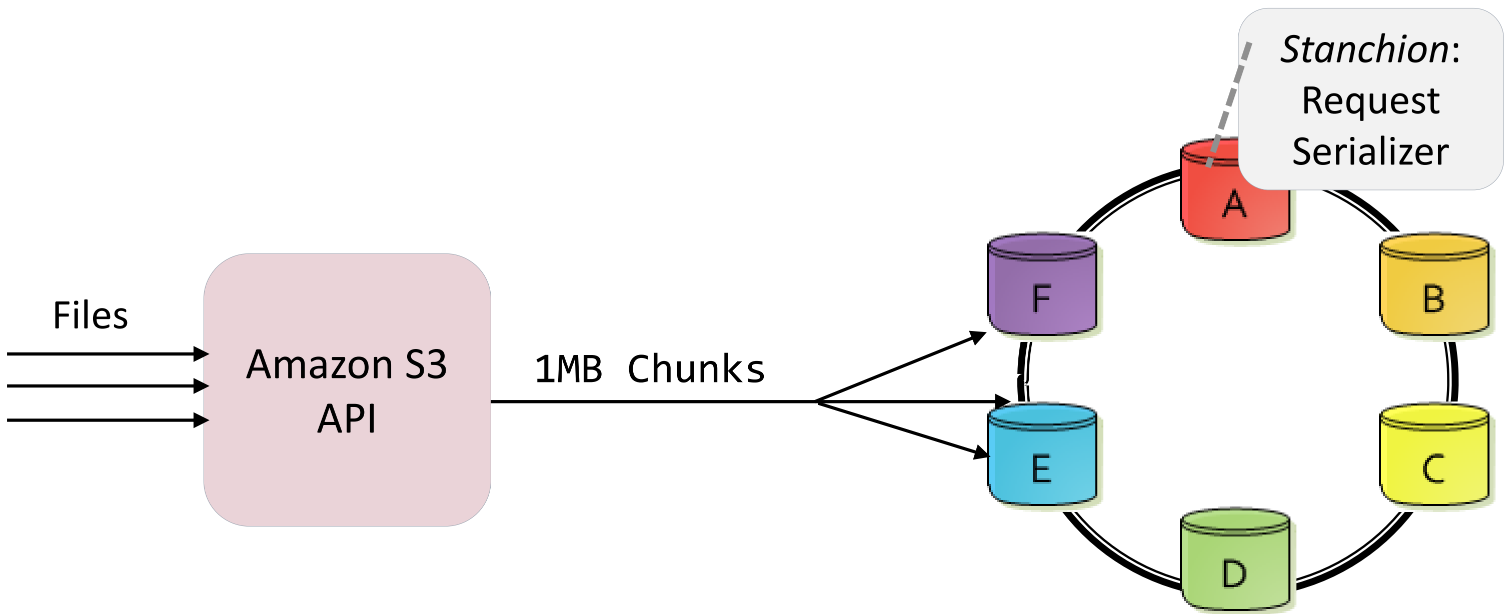
http://docs.basho.com/riak/latest/tutorials/querying/MapReduce/

Riak Map-Reduce

- JavaScript/Erlang, stored/ad-hoc
- Pattern: Chainable Reducers
- **Key-Filter**: Narrow down input
- **Link Phase**: Resolves links



Riak Cloud Storage



Summary: Dynamo and Riak

- Available and Partition-Tolerant
- **Consistent Hashing**: hash-based distribution with stability under topology changes (e.g. machine failures)
- Parameters: **N** (Replicas), **R** (Read Acks), **W** (Write Acks)
  - N=3, R=W=1 → fast, potentially inconsistent
  - N=3, R=3, W=1 → slower reads, most recent object version contained
- **Vector Clocks**: concurrent modification can be detected, inconsistencies are healed by the application
- **API**: Create, Read, Update, Delete (CRUD) on key-value pairs
- **Riak**: Open-Source Implementation of the Dynamo paper

Dynamo and Riak

Classification

Sharding	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
Replication	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
Storage Management	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
Query Processing	Global Index	Local Index	Query Planning	Analytics	Materialized Views

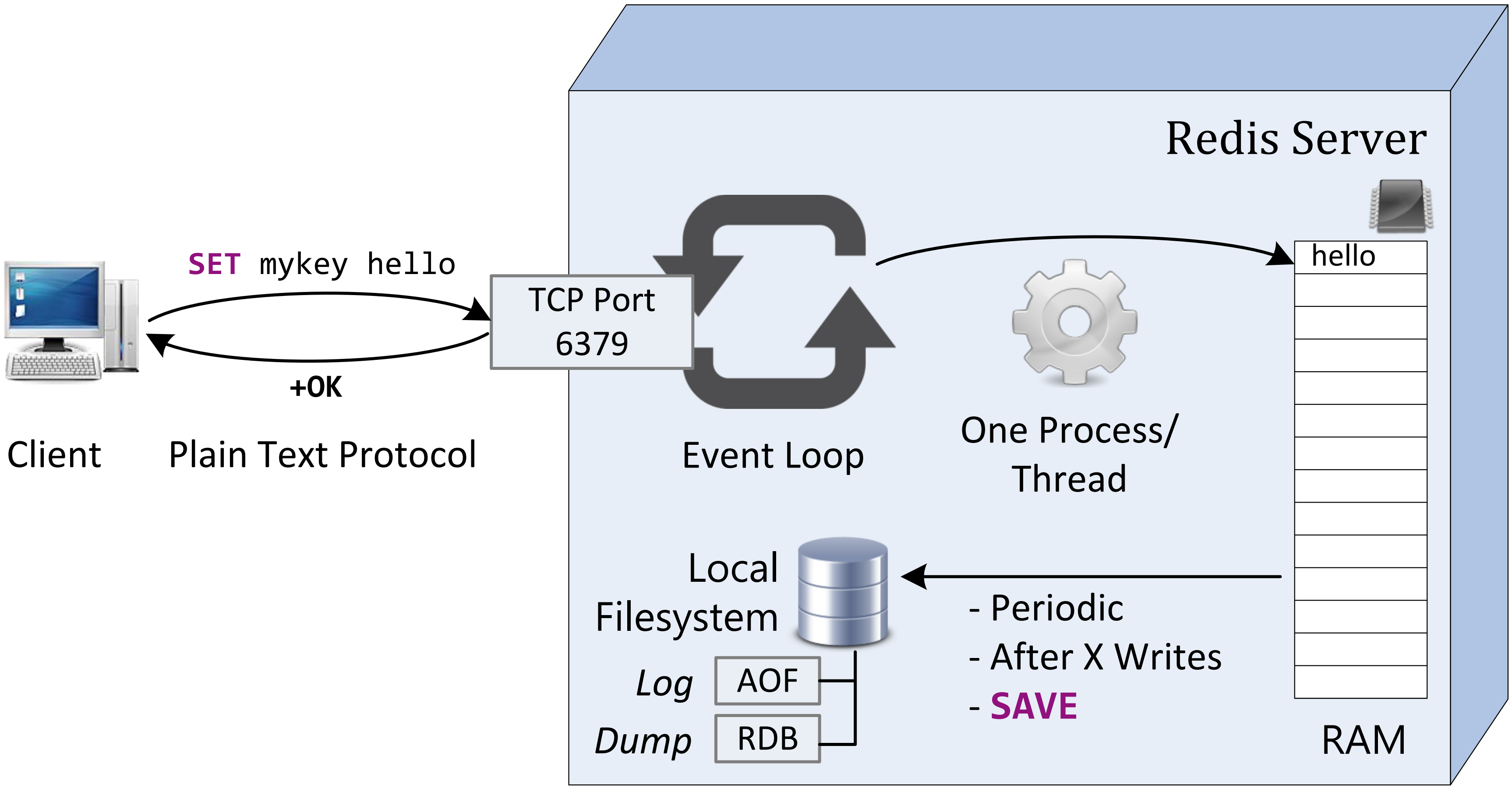
Redis (CA)

Redis
Model:
Key-Value
License:
BSD
Written in:
C

- **Re**mote **Di**ctionary **S**erver
- In-Memory Key-Value Store
- Asynchronous Master-Slave Replication
- Data model: rich data structures stored under key
- **Tunable persistence**: logging and snapshots
- Single-threaded event-loop design (similar to Node.js)
- Optimistic **batch transactions** (*Multi blocks*)
- Very high performance: >100k ops/sec per node
- Redis Cluster adds sharding

Redis Architecture

- Redis Codebase ≅ 20K LOC

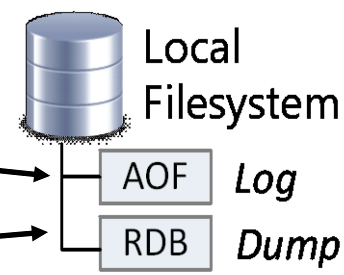


Persistence

- Default: „Eventually Persistent“
- **AOF**: Append Only File (~Commitlog)
- **RDB**: Redis Database Snapshot

```
config set appendonly everysec
```

`fsync()` every second

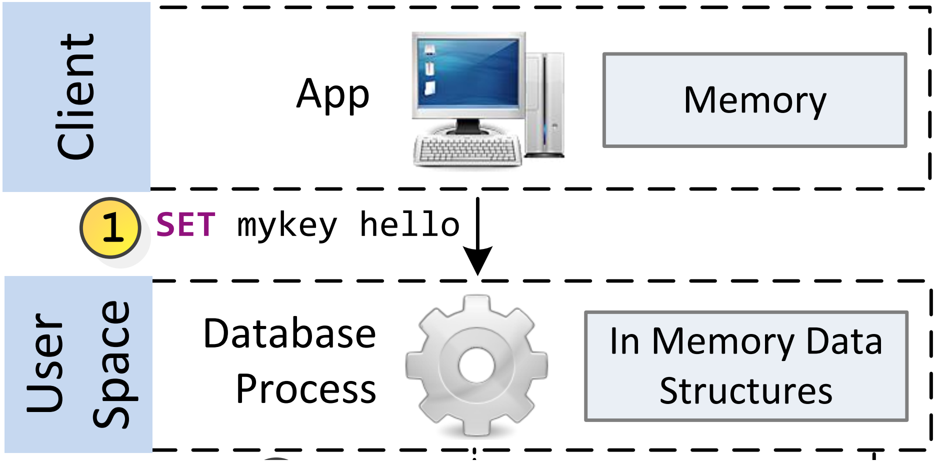


Snapshot every 60s,
if > 1000 keys changed

```
config set save 60 1000
```

Persistence

1. Resistence to client crashes
2. Resistence to DB process crashes
3. Resistence to hardware crashes with *Write-Through*
4. Resistence to hardware crashes with *Write-Back*

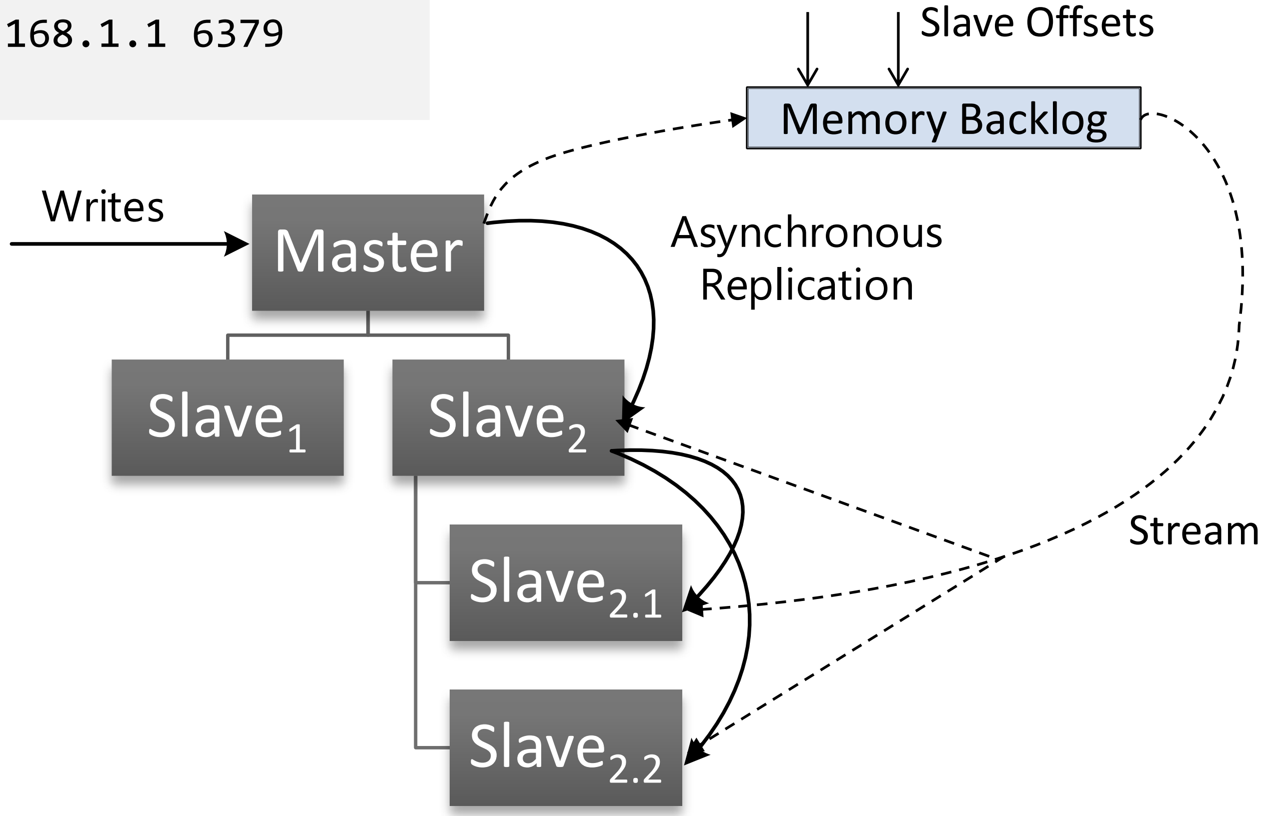


Persistence: Redis vs an RDBMS

PostgreSQL:	Redis:
`synchronous_commit on`	`appendfsync always`
Latency > Disk Latency, Group Commits, Slow	
`synchronous_commit off`	`appendfsync everysec`
periodic `fsync()`, data loss limited	
`fsync false`	`appendfysnc no`
Data corruption and loss possible	Data loss possible, corruption prevented
`pg_dump`	`save` oder `bgsave`

Master-Slave Replication

```
> SLAVEOF 192.168.1.1 6379
< +OK
```



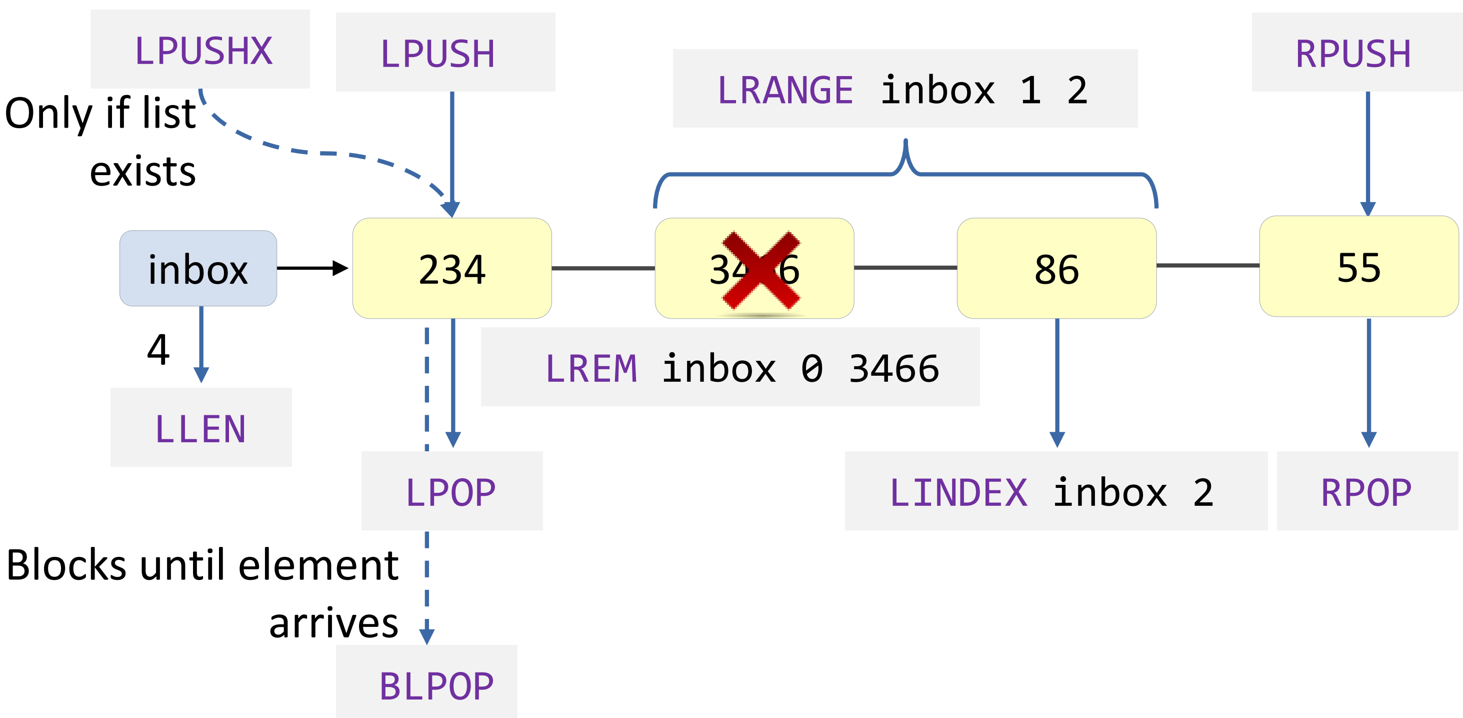
Data structures

- String, List, Set, Hash, Sorted Set

String	web:index	"<html><head>…"
Set	users:2:friends	{23, 76, 233, 11}
List	users:2:inbox	[234, 3466, 86,55]
Hash	users:2:settings	Theme → "dark", cookies → "false"
Sorted Set	top-posters	466 → "2", 344 → "16"
Pub/Sub	users:2:notifs	"{event: 'comment posted', time : …"

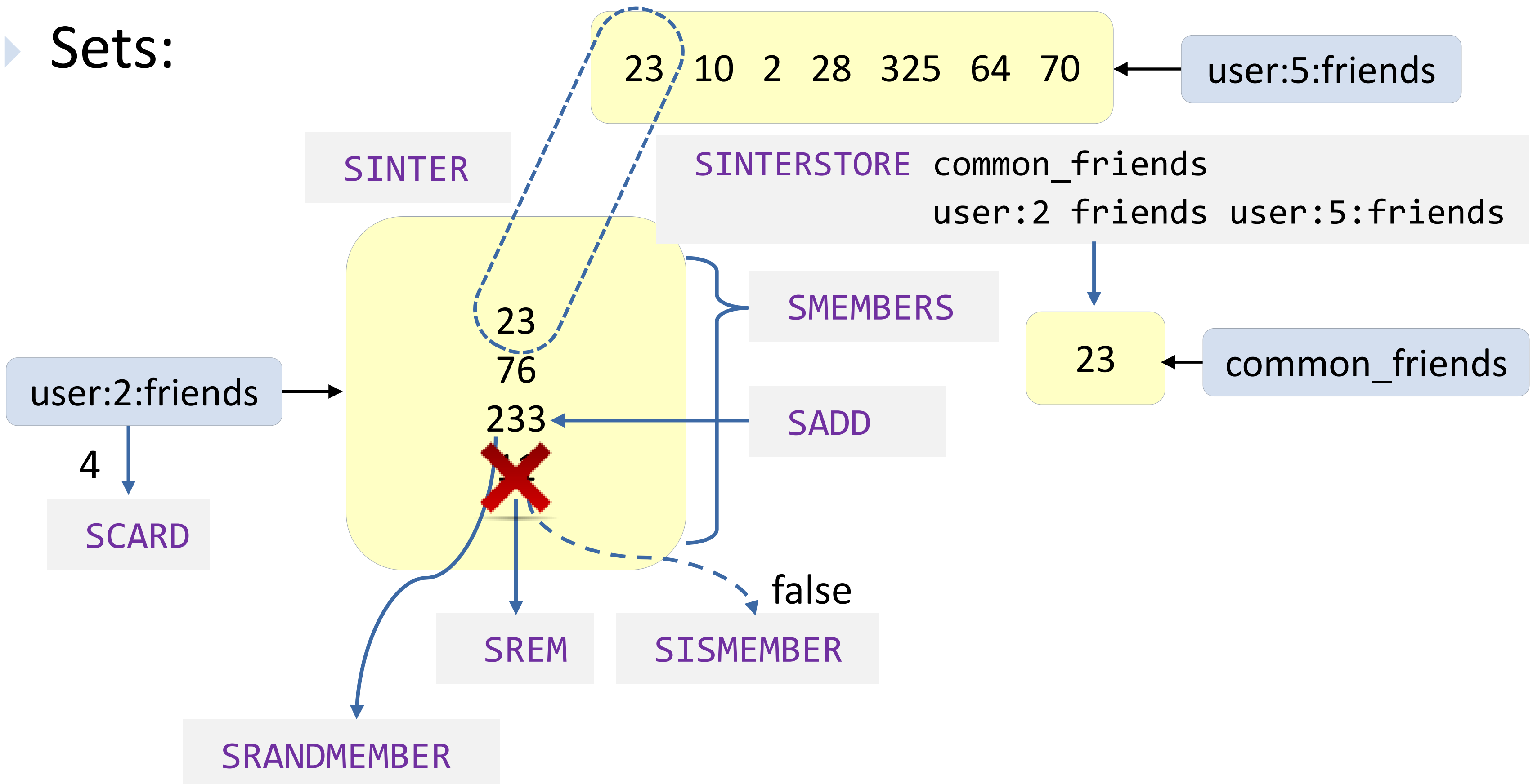
Data Structures

- (Linked) Lists:



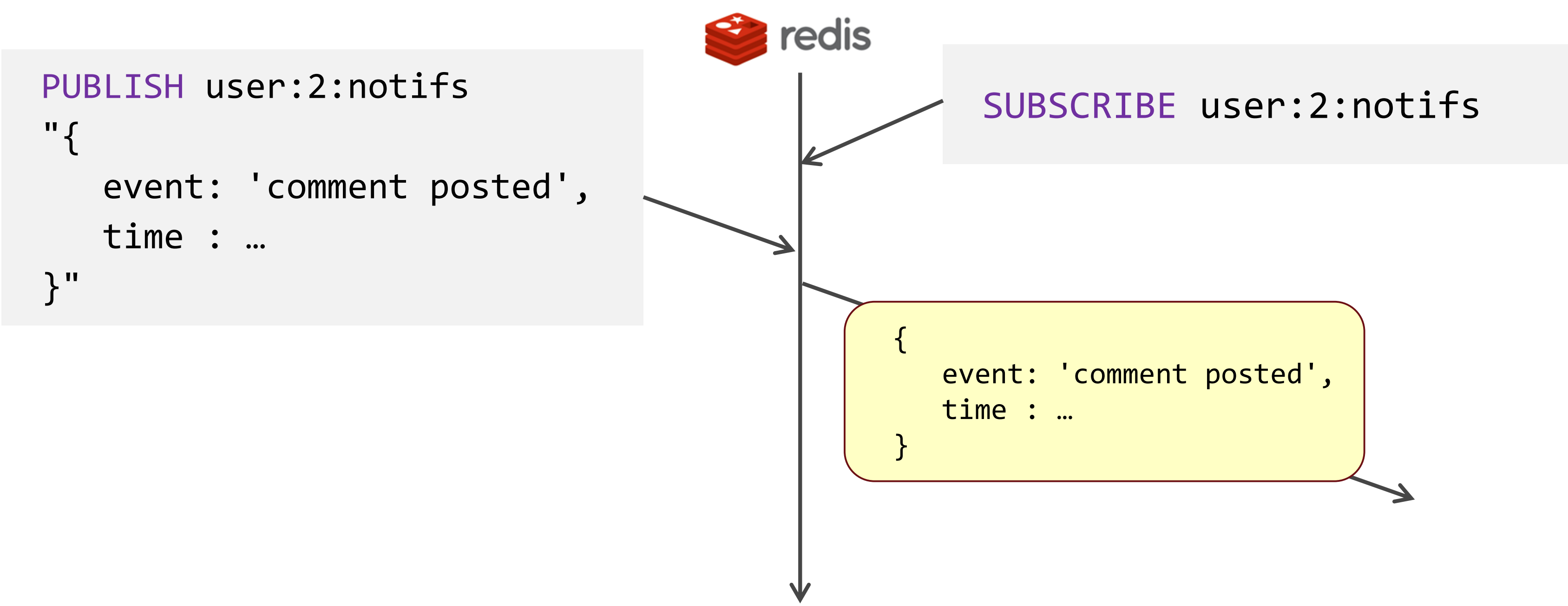
Data Structures

- Sets:



Data Structures

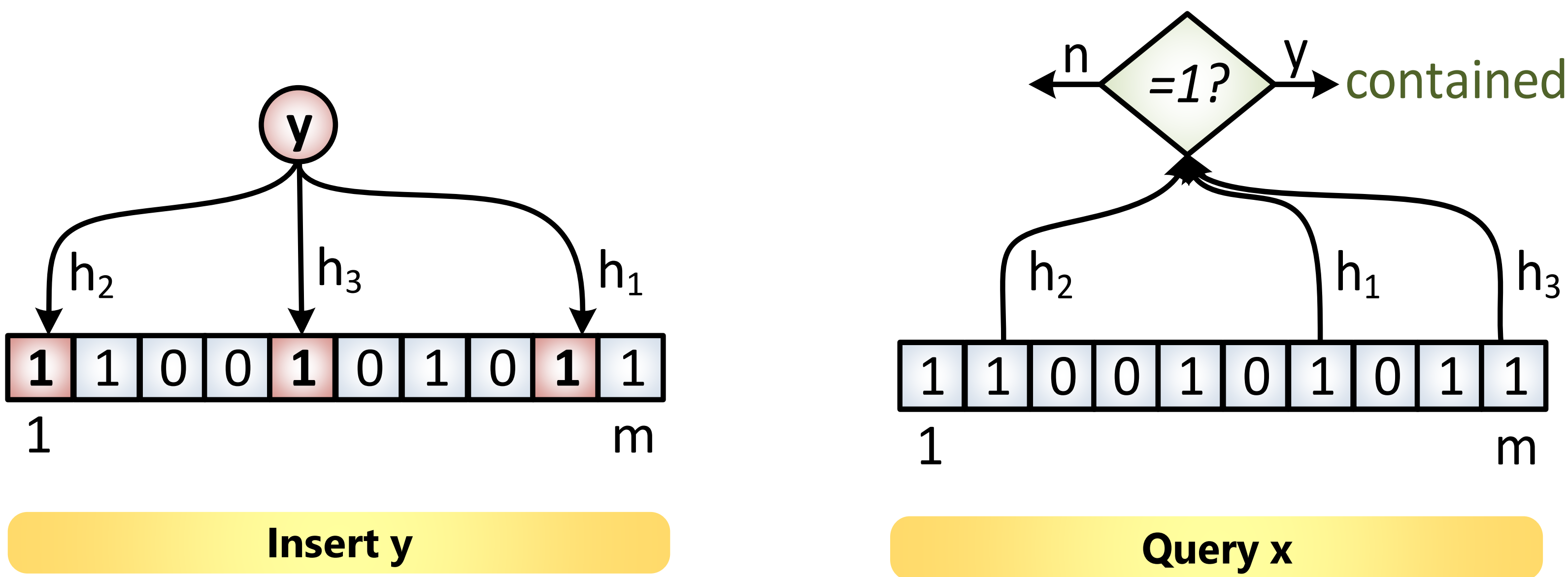
- Pub/Sub: users:2:notifs → "{event: 'comment posted', time : …"



Example: Bloom filters

Compact Probabilistic Sets

- Bit array of length **m** and **k** independent hash functions
- **insert(obj)**: add to set
- **contains(obj)**: might give a false positive



https://github.com/Baqend/Orestes-Bloomfilter

Bloomfilters in Redis

- Bitvectors in Redis: String + SETBIT, GETBIT, BITOP

```
public void add(byte[] value) {
  for (int position : hash(value)) {
    jedis.setbit(name, position, true);
  }
}
```

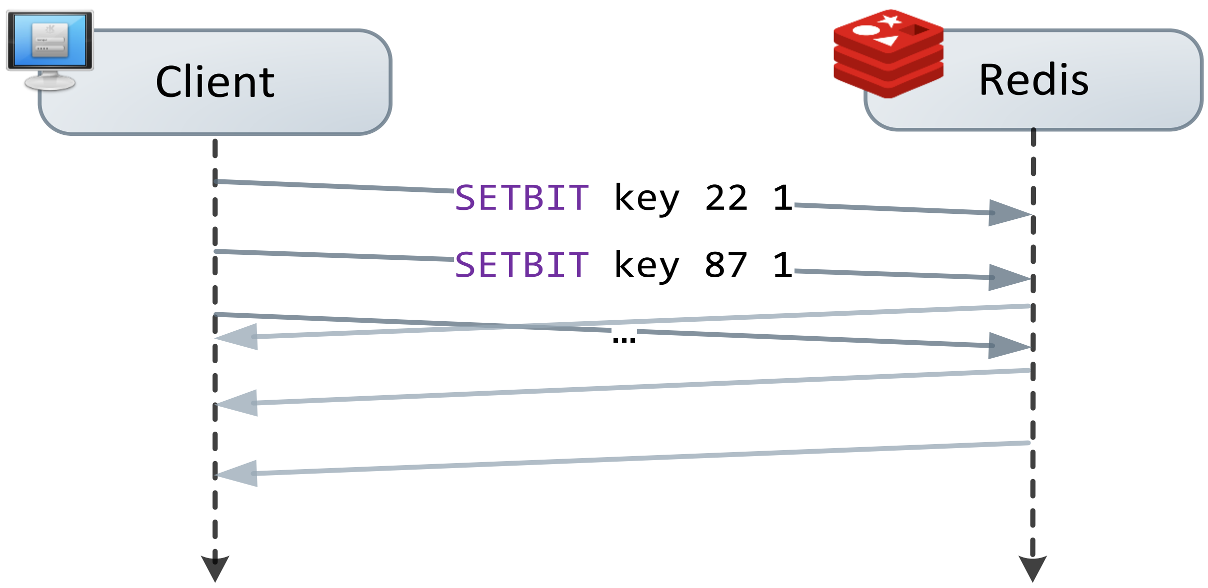
Jedis: Redis Client für Java

SETBIT creates and resizes automatically

```
public void contains(byte[] value) {
  for (int position : hash(value))
     if (!jedis.getbit(name, position))
        return false;
  return true;
}
```

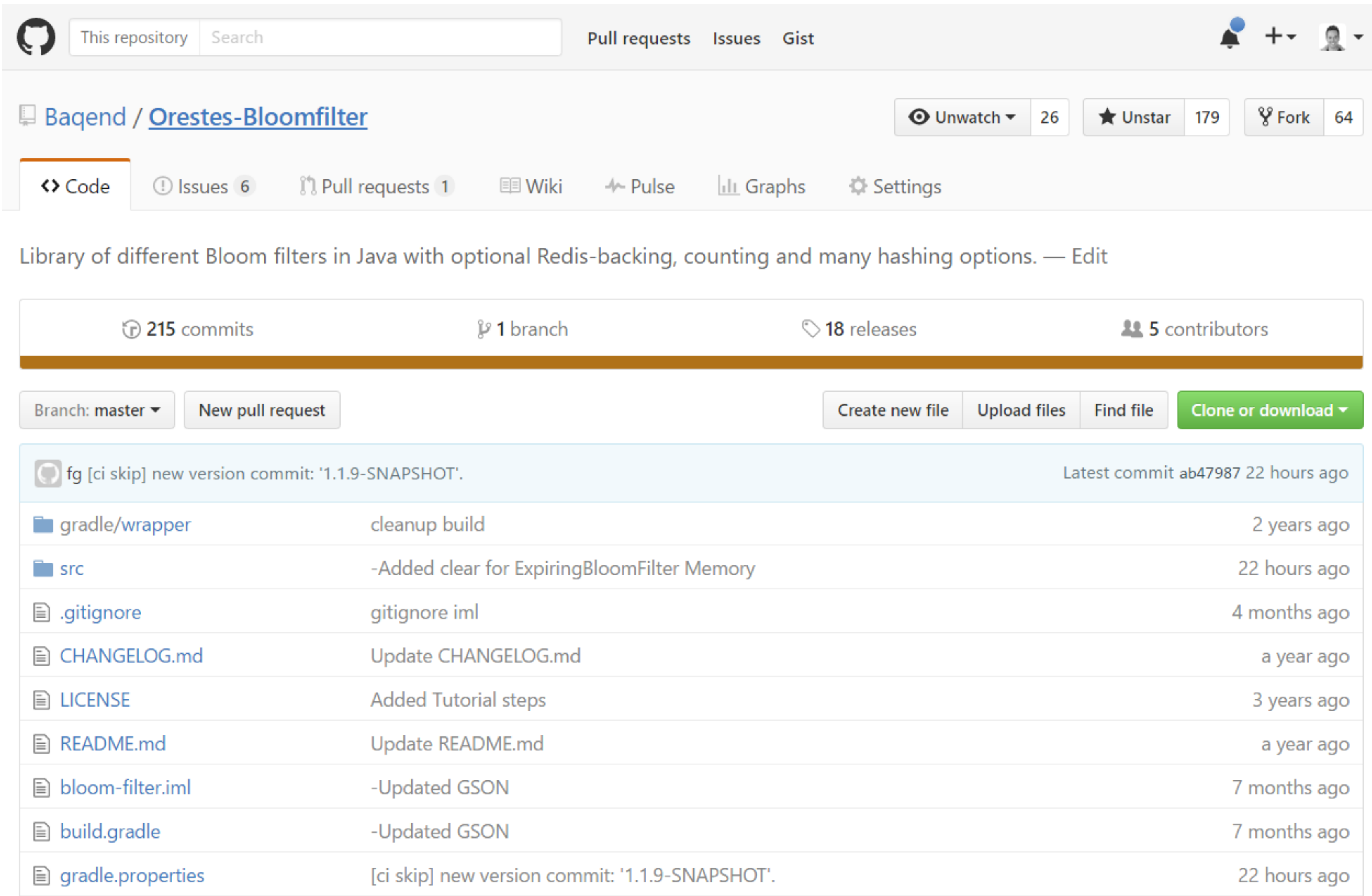
Pipelining

- If the Bloom filter uses 7 hashes: 7 roundtrips
- **Solution**: Redis Pipelining



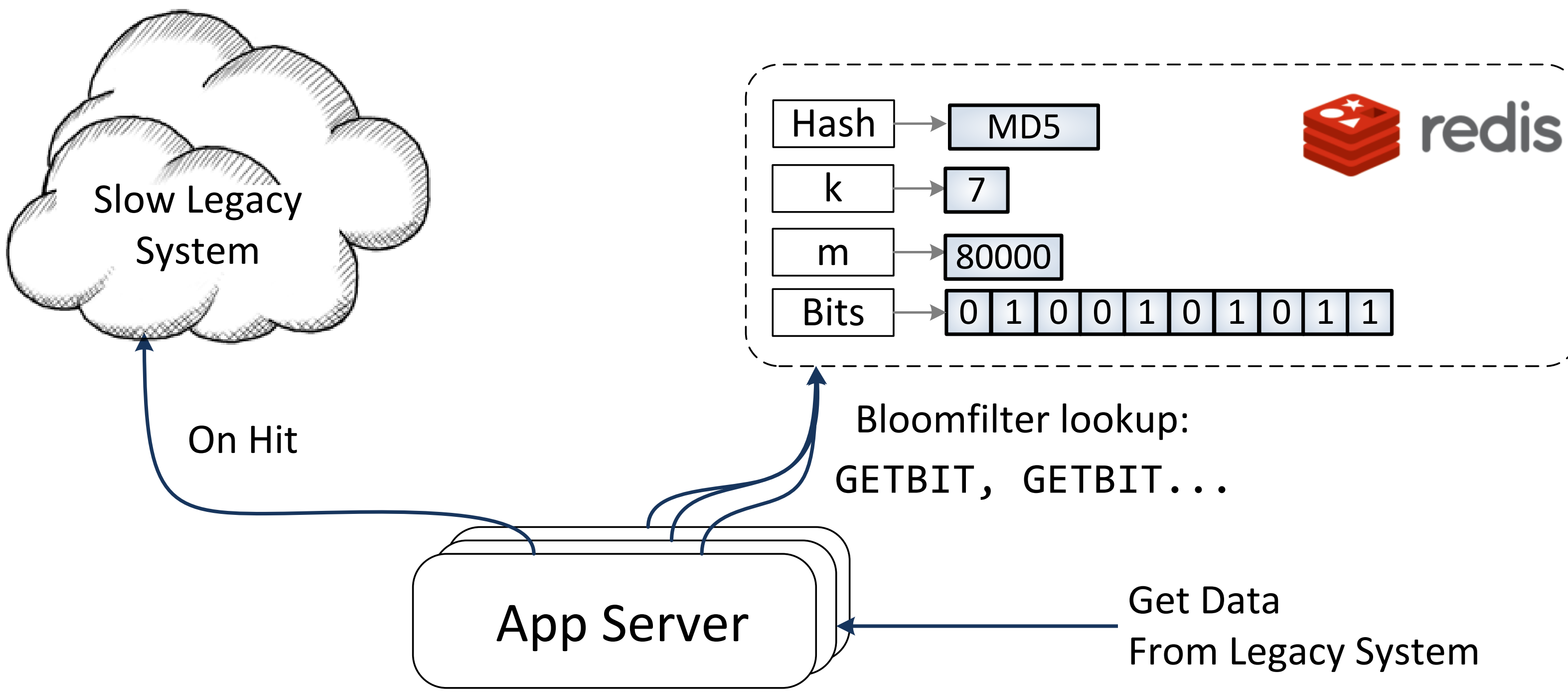
Our Bloom filters

Open Source Implementation

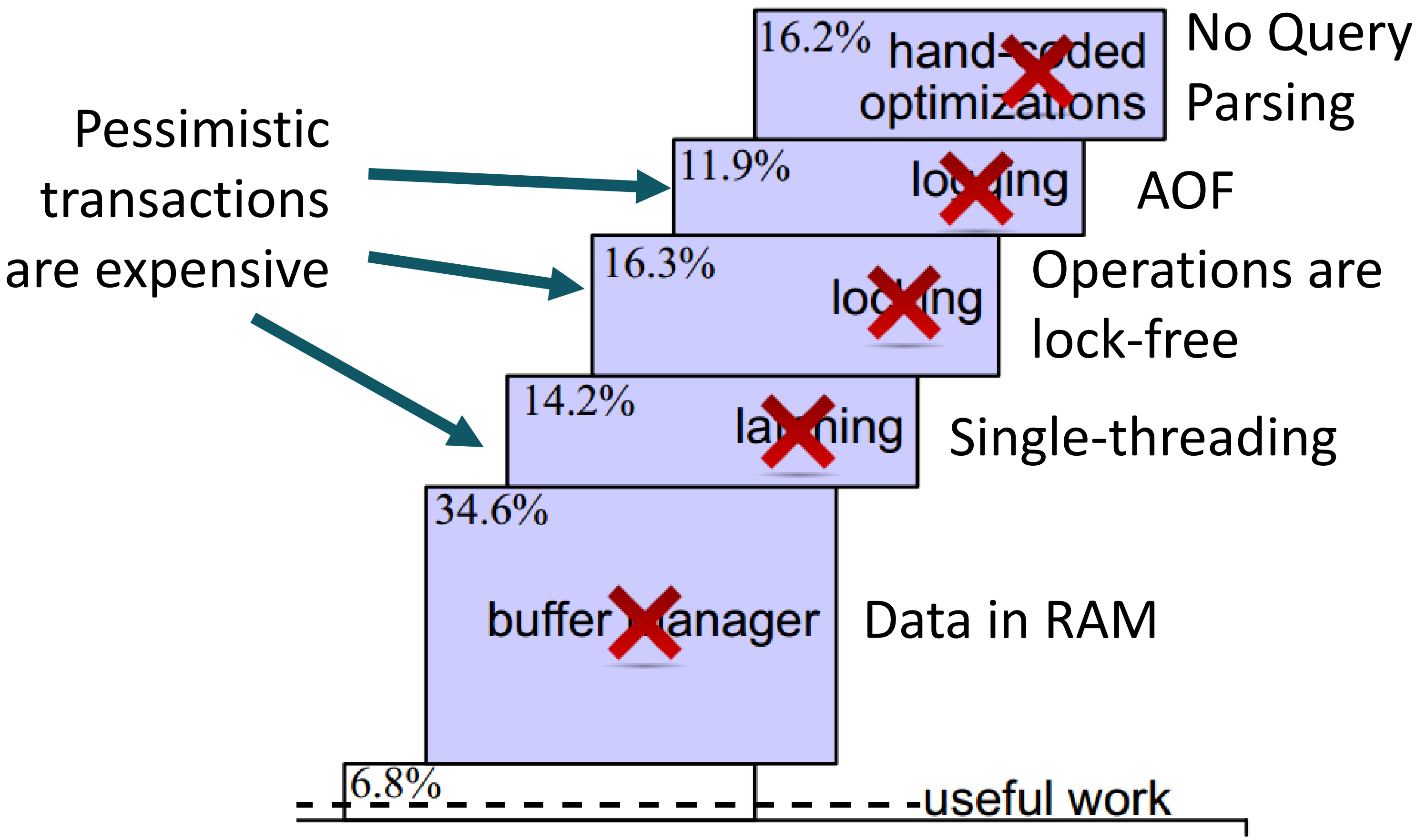


Redis for distributed systems

- Common Pattern: distributed system with **shared state** in Redis
- Example - Improve performance for legacy systems:



Why is Redis so fast?



Harizopoulos, Stavros, Madden, Stonebraker "OLTP through the looking glass, and what we found there."

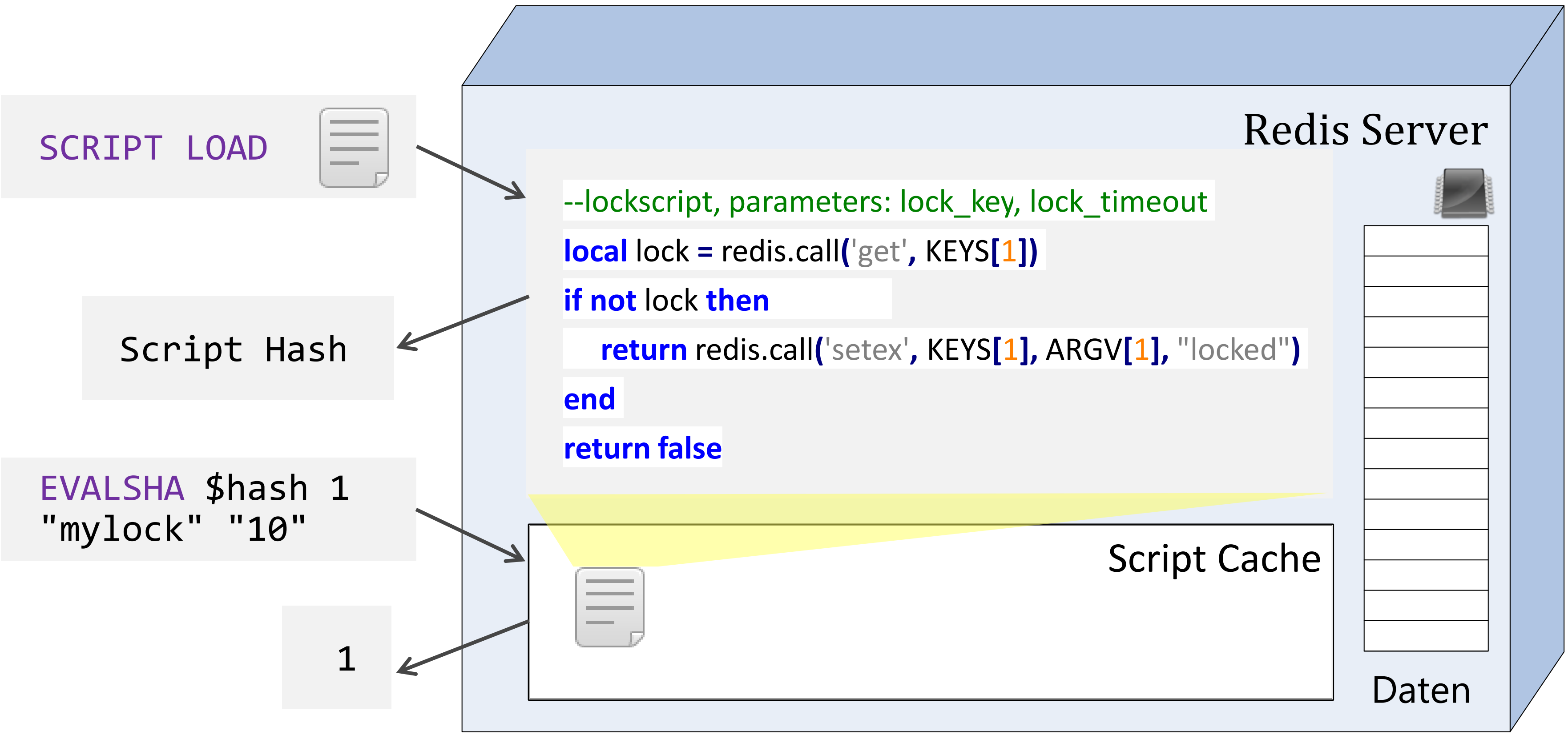
Optimistic Transactions

- MULTI: Atomic Batch Execution
- WATCH: Condition for MULTI Block

Only executed if bother keys are unchanged

```
WATCH users:2:followers, users:3:followers

MULTI

   SMEMBERS users:2:followers ⟶ Queued

   SMEMBERS users:3:followers ⟶ Queued

   INCR transactions ⟶ Queued

EXEC ⟶ Bulk reply with 3 results
```

Lua Scripting

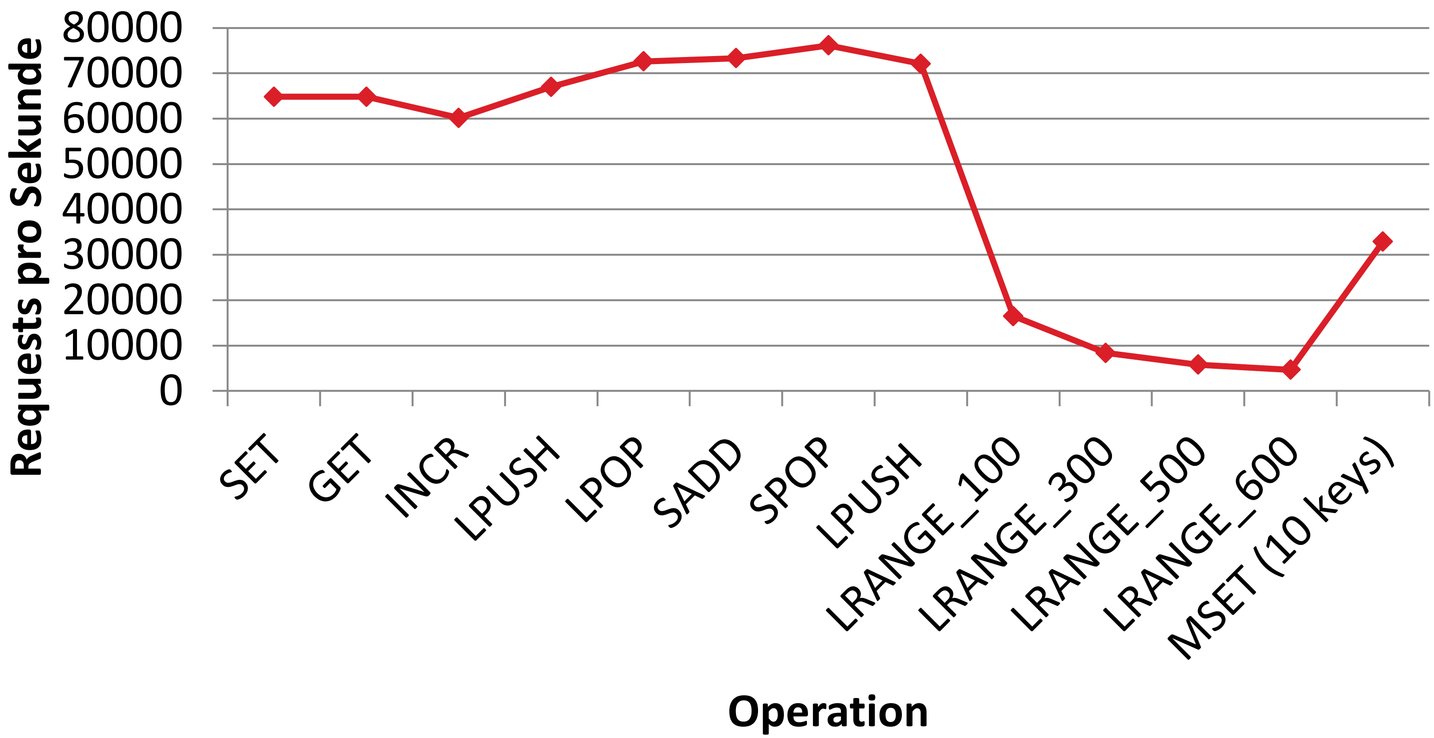


Ierusalimschy, Roberto. Programming in lua. 2006.

Performance

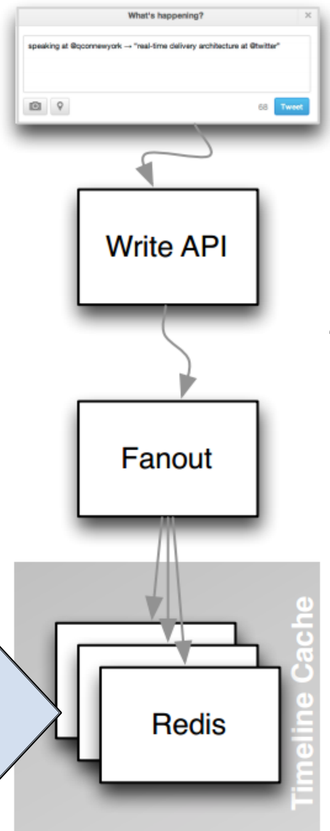
- Equivalent to Memcache

```
> redis-benchmark -n 100000 -c 50
```



Example Redis Use-Case: Twitter

- Per User: one materialized timeline in Redis
- Timeline = List
- Key: User ID



>150 million users
~300k timeline querys/s

```
RPUSHX user_id tweet
```

http://www.infoq.com/presentations/Real-Time-Delivery-Twitter

Classification: Redis

Techniques

Sharding	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
Replication	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
Storage Management	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
Query Processing	Global Index	Local Index	Query Planning	Analytics	Materialized Views

Google BigTable (CP)

- Published by Google in 2006
- Original purpose: storing the Google search index

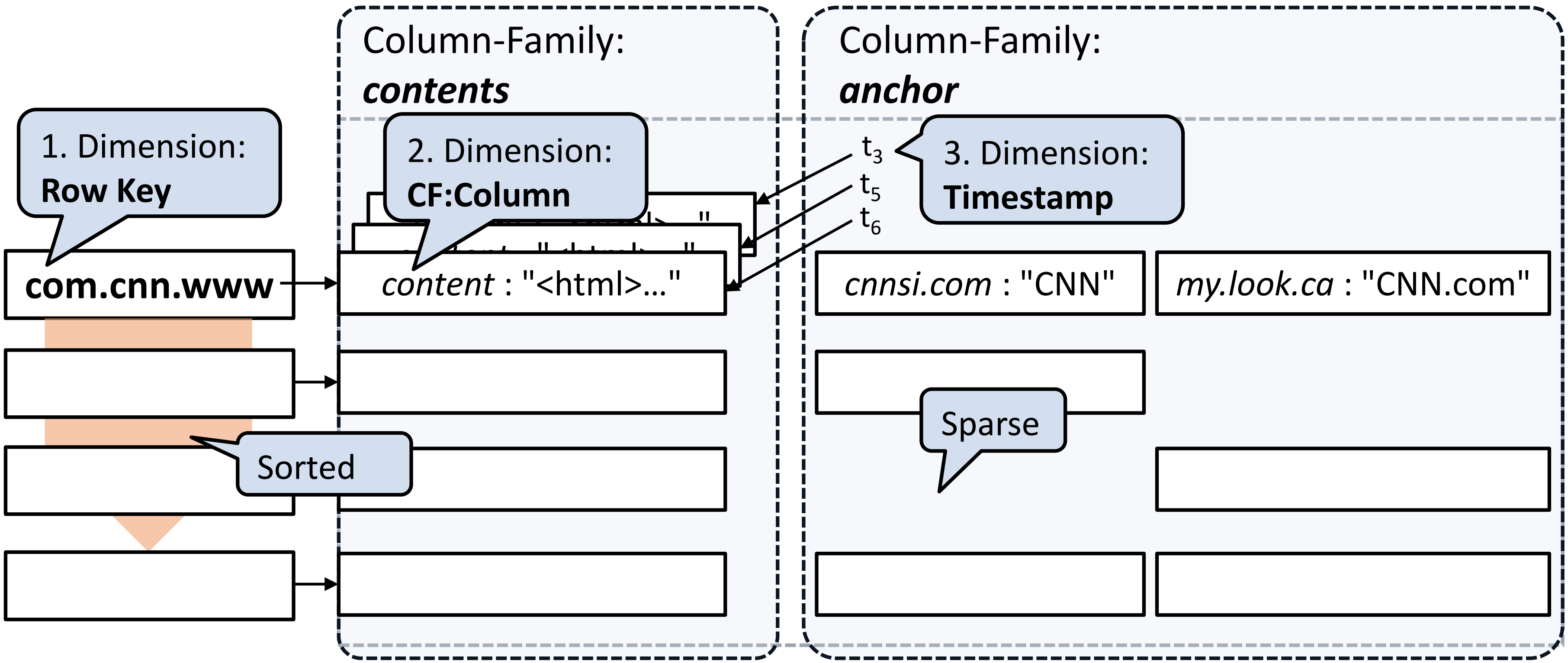
A Bigtable is a sparse, distributed, persistent multidimensional sorted map.

- Data model also used in: **HBase**, **Cassandra**, **HyperTable, Accumulo**

Chang, Fay, et al. "Bigtable: A distributed storage system for structured data."

Wide-Column Data Modelling

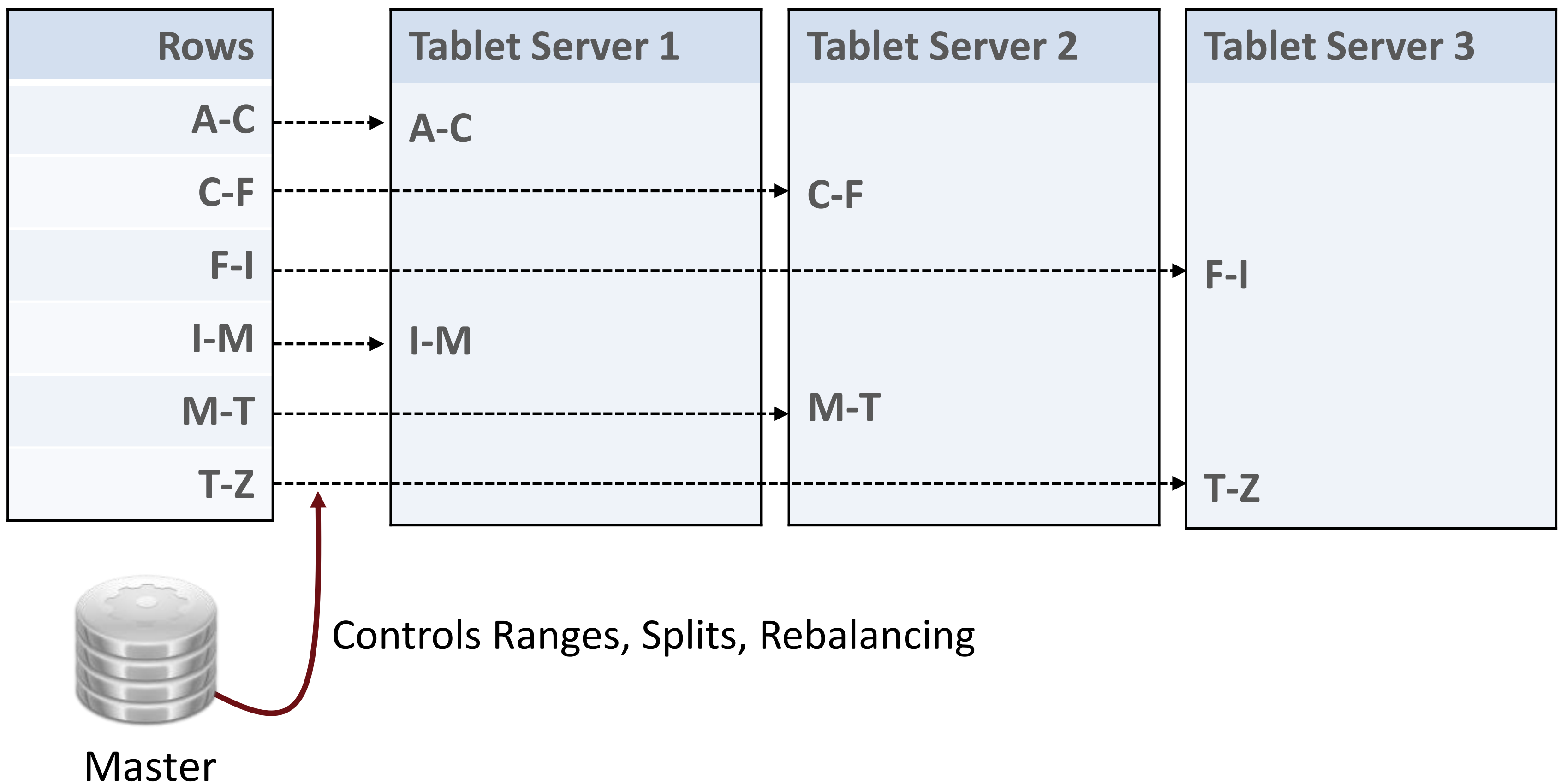
- Storage of crawled web-sites („Webtable“):



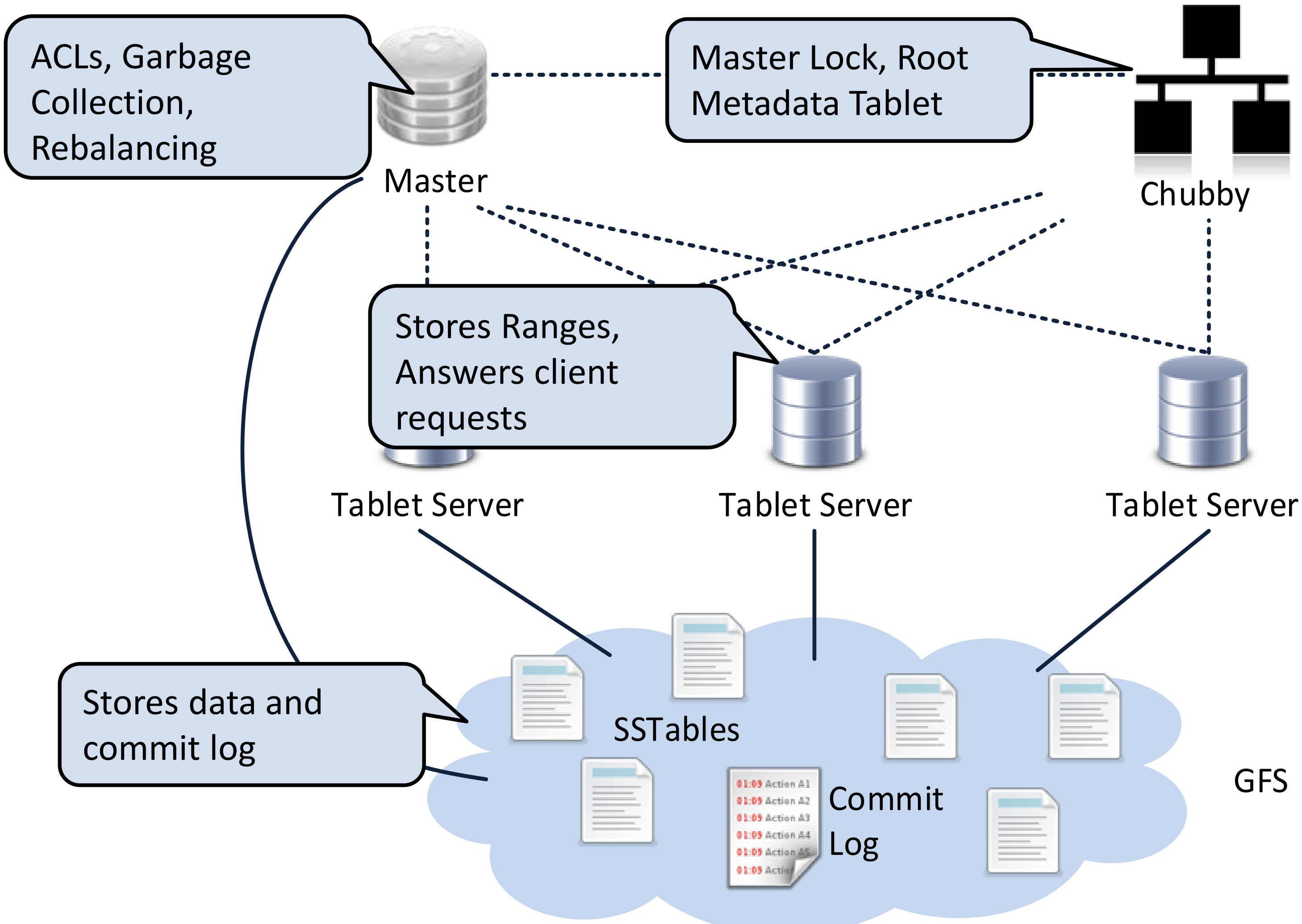
Range-based Sharding

BigTable Tablets

Tablet: Range partition of ordered records

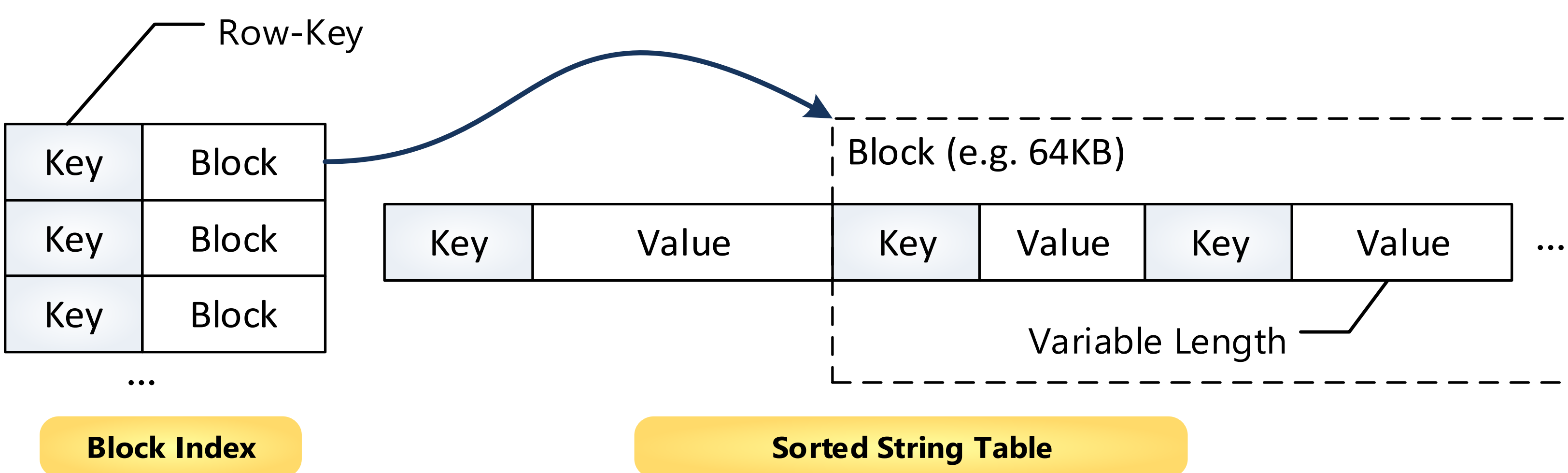


Architecture



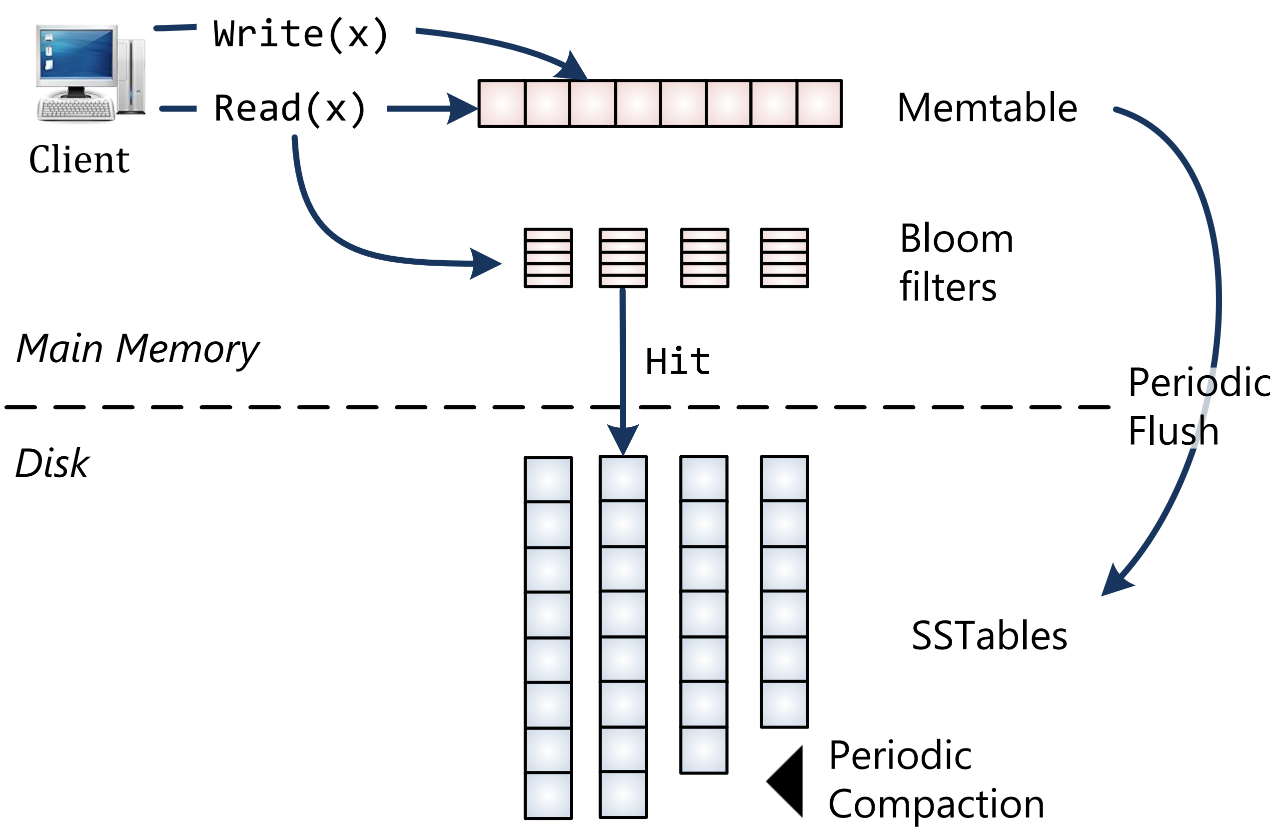
Storage: Sorted-String Tables

- **Goal**: Append-Only IO when writing (no disk seeks)
- Achieved through: **Log-Structured Merge Trees**
- **Writes** go to an in-memory *memtable* that is periodically persisted as an *SSTable* as well as a *commit log*
- **Reads** query memtable and all SSTables



Storage: Optimization

- Writes: In-Memory in **Memtable**
- SSTable disk access optimized by Bloom filters



Apache HBase (CP)

HBase
Model:
Wide-Column
License:
Apache 2
Written in:
Java

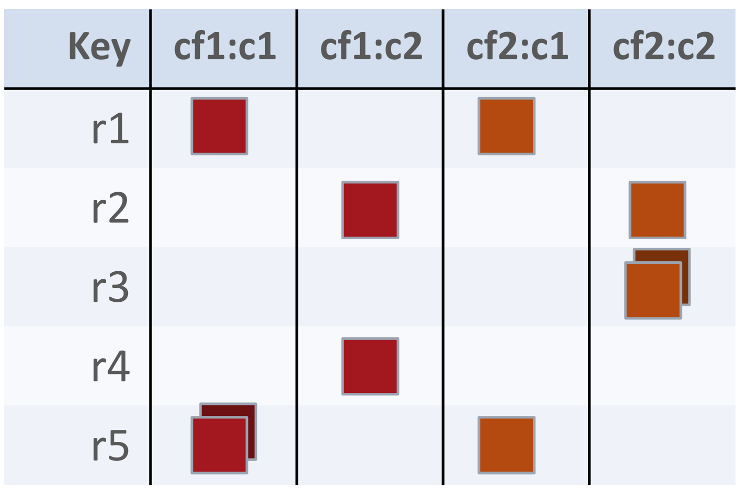
- Open-Source Implementation of BigTable
- Hadoop-Integration
  - Data source for Map-Reduce
  - Uses Zookeeper and HDFS
- Data modelling challenges: key design, tall vs wide
  - **Row Key**: only access key (no indices) → key design important
  - **Tall**: good for scans
  - **Wide**: good for gets, consistent (*single-row atomicity*)
- No typing: application handles serialization
- Interface: REST, Avro, Thrift

HBase Storage

- Logical to physical mapping:

In Value	r2:cf2:c2:t1:<value>
In Key	r2-<value>:cf2:c2:t1:_
In Column	r2:cf2:c2<value>:t1:_

Key Design – where to store data:
r2:cf2:c2:t1:<value>
r2-<value>:cf2:c2:t1:_
r2:cf2:c2<value>:t1:_



r1:cf2:c1:t1:<value>
r2:cf2:c2:t1:<value>
r3:cf2:c2:t2:<value>
r3:cf2:c2:t1:<value>
r5:cf2:c1:t1:<value>

HFile cf2

r1:cf1:c1:t1:<value>
r2:cf1:c2:t1:<value>
r3:cf1:c2:t1:<value>
r3:cf1:c1:t2:<value>
r5:cf1:c1:t1:<value>

HFile cf1

George, Lars. HBase: the definitive guide. 2011.

Example: Facebook Insights

MD5(Reversed Domain) + Reversed Domain + URL-ID

Row Key

6PM Total	6PM Male	...	01.01 Total	01.01 Male	...	Total	Male	...
10	7		100	65			567	

CF:Daily | CF:Monthly | CF:All

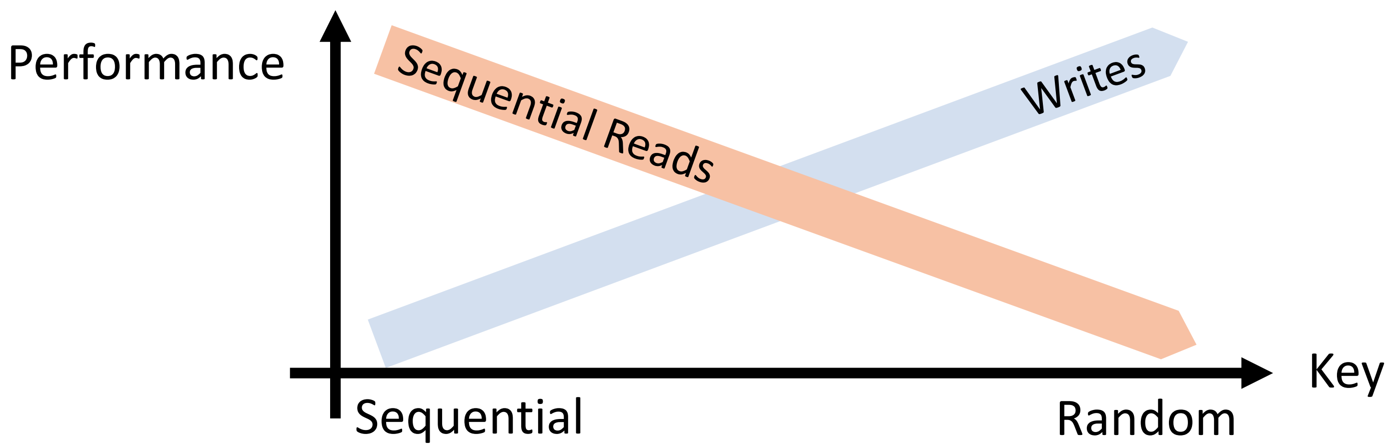
Atomic HBase **Counter**

TTL – automatic deletion of old rows

Lars George: "Advanced HBase Schema Design"

Schema Design

- Tall vs Wide Rows:
  - **Tall**: good for Scans
  - **Wide**: good for Gets
- Hotspots: Sequential Keys (z.B. Timestamp) dangerous



George, Lars. HBase: the definitive guide. 2011.

Schema: Messages

User ID	CF	Column	Timestamp	Message
12345	data	5fc38314-e290-ae5da5fc375d	1307097848	"Hi Lars, ..."
12345	data	725aae5f-d72e-f90f3f070419	1307099848	"Welcome, and ..."
12345	data	cc6775b3-f249-c6dd2b1a7467	1307101848	"To Whom It ..."
12345	data	dcbee495-6d5e-6ed48124632c	1307103848	"Hi, how are ..."

vs

ID:User+Message	CF	Column	Timestamp	Message
12345-5fc38314-e290-ae5da5fc375d	data		: 1307097848	"Hi Lars, ..."
12345-725aae5f-d72e-f90f3f070419	data		: 1307099848	"Welcome, and ..."
12345-cc6775b3-f249-c6dd2b1a7467	data		: 1307101848	"To Whom It ..."
12345-dcbee495-6d5e-6ed48124632c	data		: 1307103848	"Hi, how are ..."

Wide:
Atomicity
Scan over Inbox: **Get**

Tall:
Fast Message Access
Scan over Inbox: **Partial Key Scan**

http://2013.nosql-matters.org/cgn/wp-content/uploads/2013/05/HBase-Schema-Design-NoSQL-Matters-April-2013.pdf

API: CRUD + Scan

Setup Cloud Cluster:

```
> elastic-mapreduce --create --
hbase --num-instances 2 --instance-
type m1.large
```

```
> whirr launch-cluster --config
hbase.properties
```

Login, cluster size, etc.

```
HTable table = ...
Get get = new Get("my-row");
get.addColumn(Bytes.toBytes("my-cf"), Bytes.toBytes("my-col"));
Result result = table.get(get);

table.delete(new Delete("my-row"));

Scan scan = new Scan();
scan.setStartRow( Bytes.toBytes("my-row-0"));
scan.setStopRow( Bytes.toBytes("my-row-101"));
ResultScanner scanner = table.getScanner(scan)
for(Result result : scanner) { }
```

API: Features

- Row **Locks** (MVCC): `table.lockRow()`, `unlockRow()`
  - Problem: Timeouts, Deadlocks, Ressources
- **Conditional Updates**: `checkAndPut()`, `checkAndDelete()`
- CoProcessors - registriered Java-Classes for:
  - Observers (`prePut`, `postGet`, etc.)
  - Endpoints (Stored Procedures)
- HBase can be a Hadoop **Source**:

```
TableMapReduceUtil.initTableMapperJob(
  tableName,  //Table
  scan,  //Data input as a Scan
  MyMapper.class, ...  //usually a TableMapper<Text,Text> );
```

Summary: BigTable, HBase

- Data model: $(rowkey, cf\!:\!column, timestamp) \rightarrow value$
- **API**: CRUD + Scan(*start-key, end-key*)
- Uses distributed file system (GFS/HDFS)
- Storage structure: **Memtable** (in-memory data structure) + **SSTable** (persistent; append-only-IO)
- **Schema design**: only primary key access → implicit schema (key design) needs to be carefully planned
- **HBase**: very literal open-source BigTable implementation

Classification: HBase

Techniques

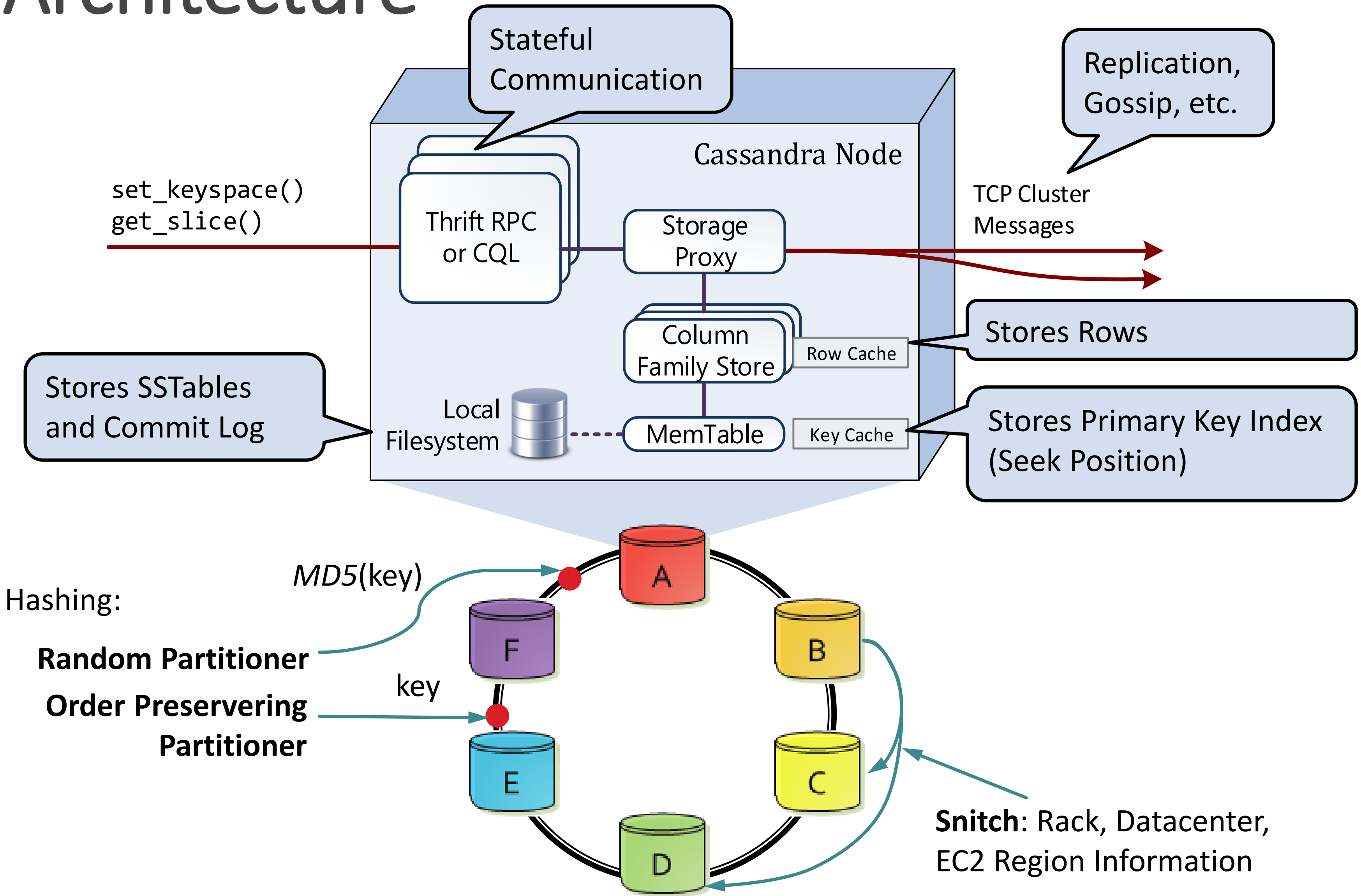
Sharding	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
Replication	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
Storage Management	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
Query Processing	Global Index	Local Index	Query Planning	Analytics	Materialized Views

Apache Cassandra (AP)

Cassandra
Model:
Wide-Column
License:
Apache 2
Written in:
Java

- Published 2007 by Facebook
- **Idea**:
  - BigTable's wide-column data model
  - Dynamo ring for replication and sharding
- Cassandra Query Language (CQL): SQL-like query- and DDL-language
- **Compound indices**: *partition key* (shard key) + *clustering key* (ordered per partition key) → Limited range queries

Architecture



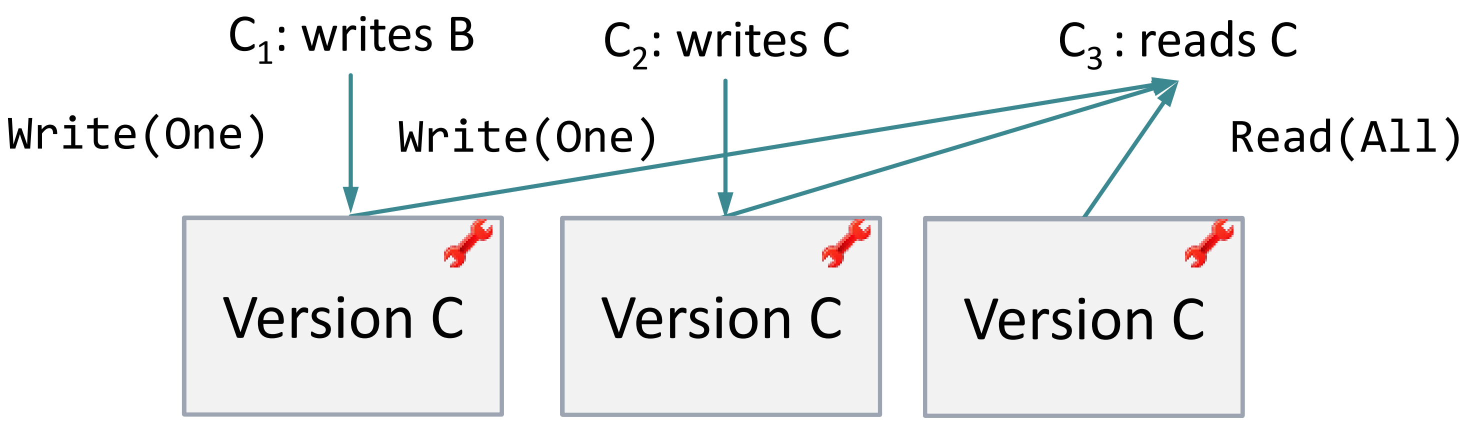
Consistency

- No Vector Clocks but **Last-Write-Wins**
  - ➔ Clock synchronisation required
- No Versionierung that keeps old cells

Write	**Read**
Any	-
One	One
Two	Two
Quorum	Quorum
Local_Quorum / Each_Quorum	Local_Quorum / Each_Quorum
All	All

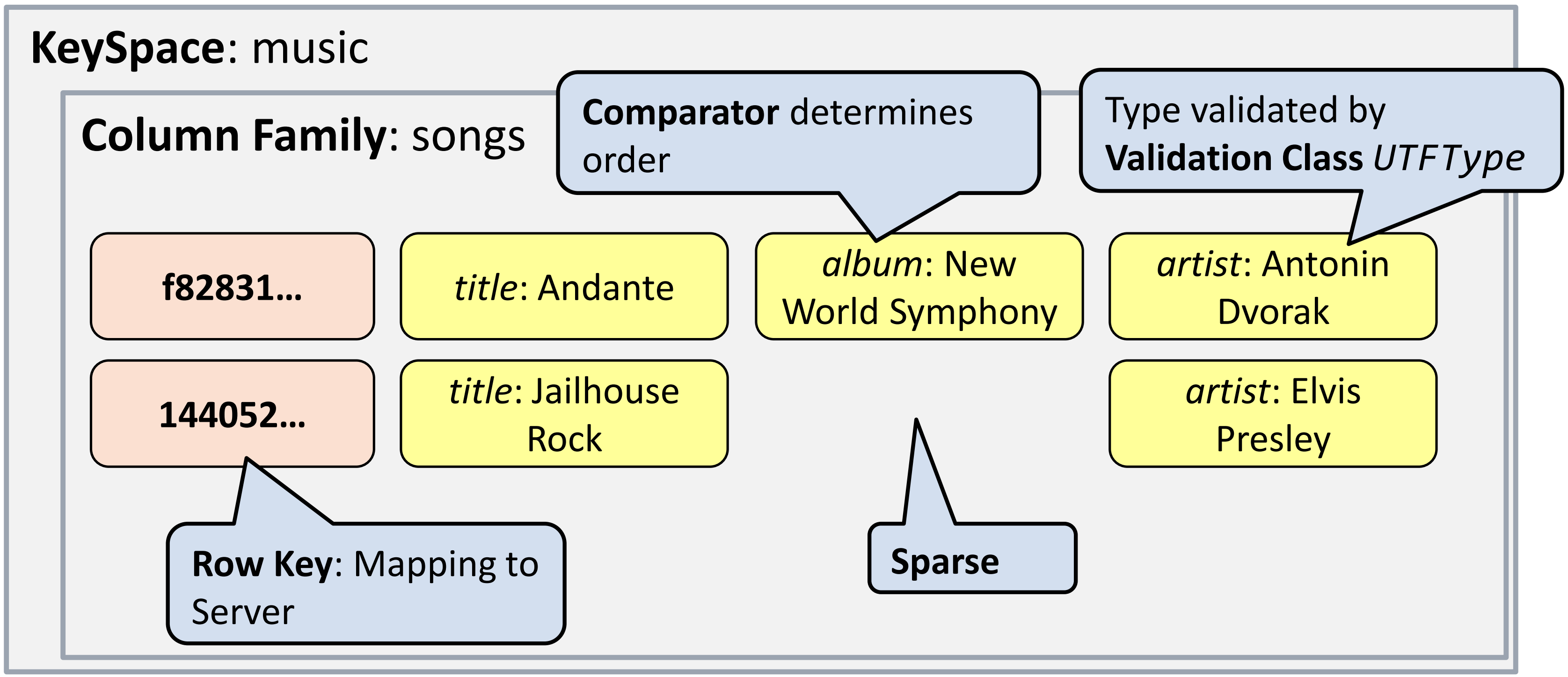
Consistency

- Coordinator chooses newest version and triggers *Read Repair*
- **Downside**: upon conflicts, changes are lost



Storage Layer

- Uses BigTables Column Family Format



http://www.datastax.com/dev/blog/cql3-for-cassandra-experts

CQL Example: Compound keys

- Enables Scans despite *Random Partitioner*

```
CREATE TABLE playlists (
  id uuid,
  song_order int,
  song_id uuid, ...
  PRIMARY KEY (id, song_order)
);
```

```
SELECT * FROM playlists
WHERE id = 23423
ORDER BY song_order DESC
LIMIT 50;
```

Partition Key

Clustering Columns:
sorted per node

id	song_order	song_id	artist
23423	1	64563	Elvis
23423	2	f9291	Elvis

Other Features

- **Distributed Counters** – prevent update anomalies
- **Full-text Search** (Solr) in Commercial Version
- **Column TTL** – automatic garbage collection
- **Secondary indices**: hidden table with mapping
  → queries with simple equality condition
- **Lightweight Transactions**: linearizable updates through a Paxos-like protocol

```
INSERT INTO USERS (login, email, name, login_count)
values ('jbellis', 'jbellis@datastax.com', 'Jonathan Ellis', 1)
IF NOT EXISTS
```

Classification: Cassandra

Techniques

Sharding	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
Replication	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
Storage Management	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
Query Processing	Global Index	Local Index	Query Planning	Analytics	Materialized Views

MongoDB (CP)

MongoDB
Model:
Document
License:
GNU AGPL 3.0
Written in:
C++

- From hu**mongo**us ≅ gigantic
- Tunable consistency
- Schema-free document database
- Allows complex queries and indexing
- **Sharding** (either range- or hash-based)
- **Replication** (either synchronous or asynchronous)
- Storage Management:
  - **Write-ahead logging** for redos (*journaling*)
  - **Storage Engines:** memory-mapped files, in-memory, Log-structured merge trees (WiredTiger)

Basics

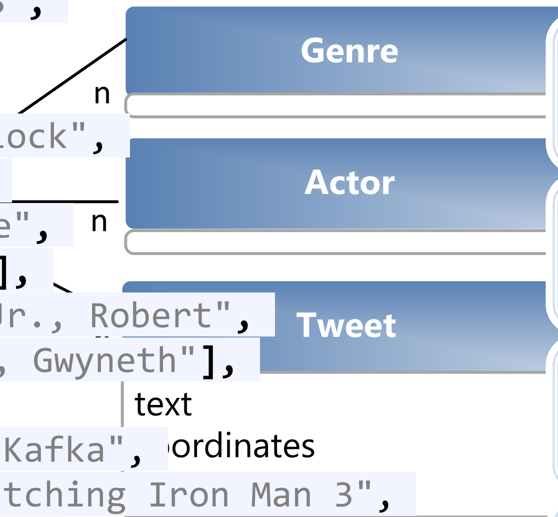
```
> mongod &
> mongo imdb
MongoDB shell version: 2.4.3
connecting to: imdb
> show collections
movies
tweets
> db.movies.findOne({title : "Iron Man 3"})
{
        title : "Iron Man 3",
        year : 2013 ,
        genre : [
                "Action",
                "Adventure",
                "Sci -Fi"],
        actors : [
                "Downey Jr., Robert",
                "Paltrow , Gwyneth",]
}
```

Properties

Arrays, Nesting allowed

Data Modelling

```
{
        "_id" : ObjectId("51a5d316d70beffe74ecc940")
        title : "Iron Man 3",
        year : 2013,
        rating : 7.6,
        director: "Shane Block",
        genre : [ "Action",
                 "Adventure",
                 "Sci -Fi"],
        actors : ["Downey Jr., Robert",
                 "Paltrow , Gwyneth"],
        tweets : [ {
           "user" : "Franz Kafka",
           "text" : "#nowwatching Iron Man 3",
           "retweet" : false,
           "date" : ISODate("2013-05-29T13:15:51Z")
        }]
}
```



Movie Document

Denormalisation instead of joins

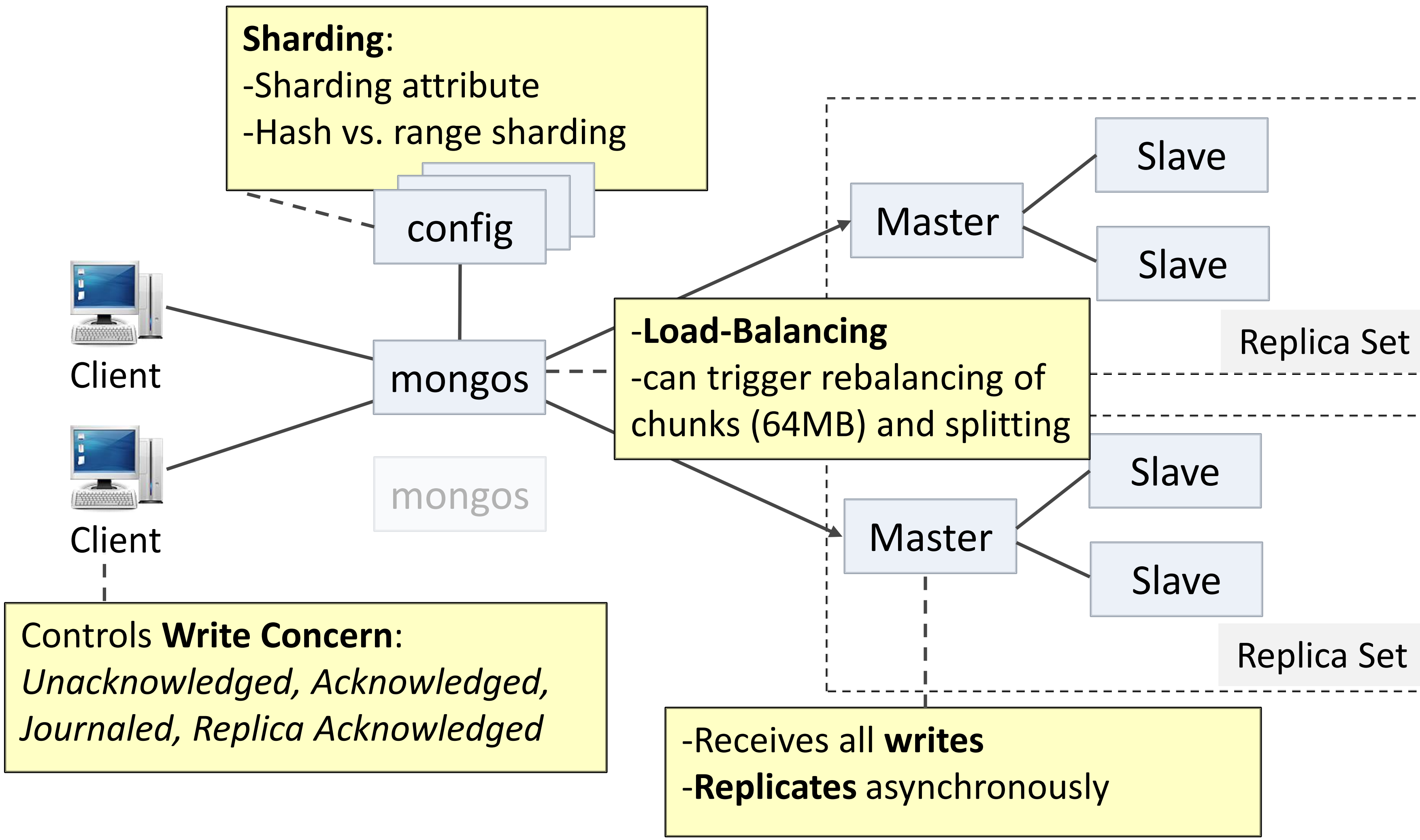
Nesting replaces 1:n and 1:1 relations

Schemafreeness: Attributes per document

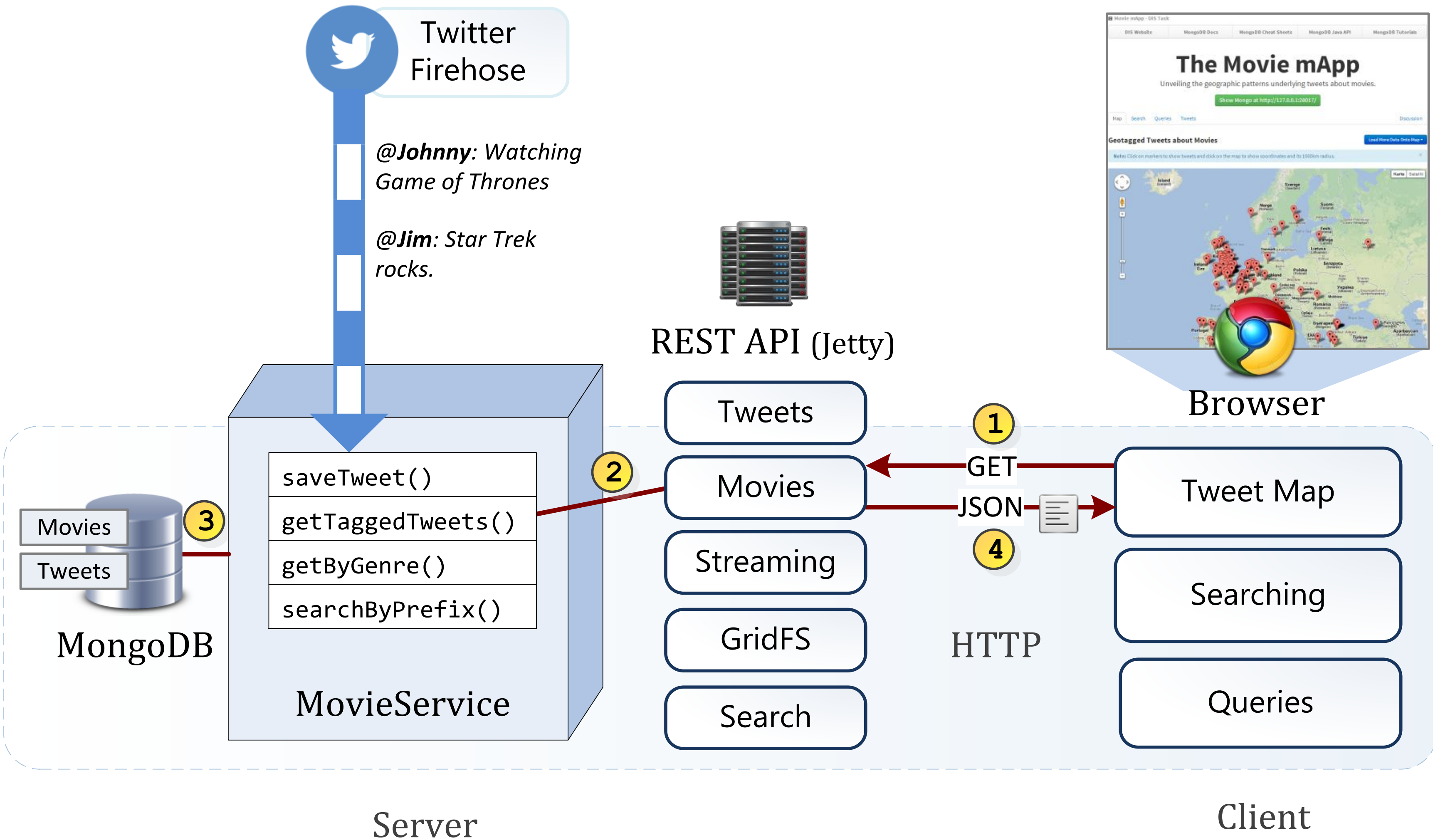
Unit of atomicity: document

Principles

Sharding und Replication



MongoDB Example App



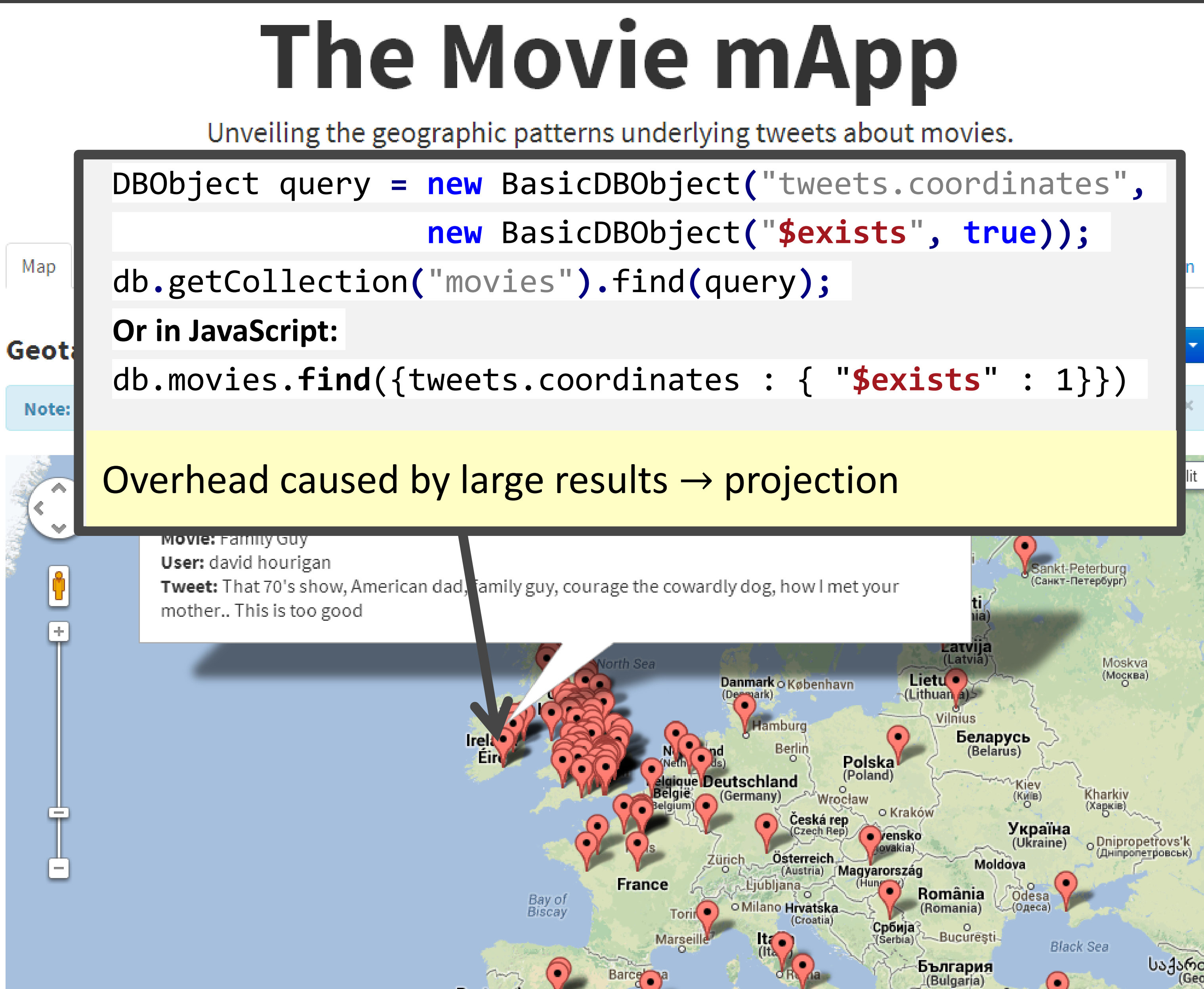
MongoDB by Example

```
DBObject query = new BasicDBObject("tweets.coordinates",
                 new BasicDBObject("$exists", true));
db.getCollection("movies").find(query);
```

Or in JavaScript:

```
db.movies.find({tweets.coordinates : { "$exists" : 1}})
```

Overhead caused by large results → projection



Movie mApp - DIS Task

DIS Website | MongoDB Docs | MongoDB Cheat Sheets | MongoDB Java API | MongoDB Tutorials

The Movie mApp

Unveiling the geographic patterns underlying tweets about movies.

Show Mongo at http://127.0.0.1:28017/

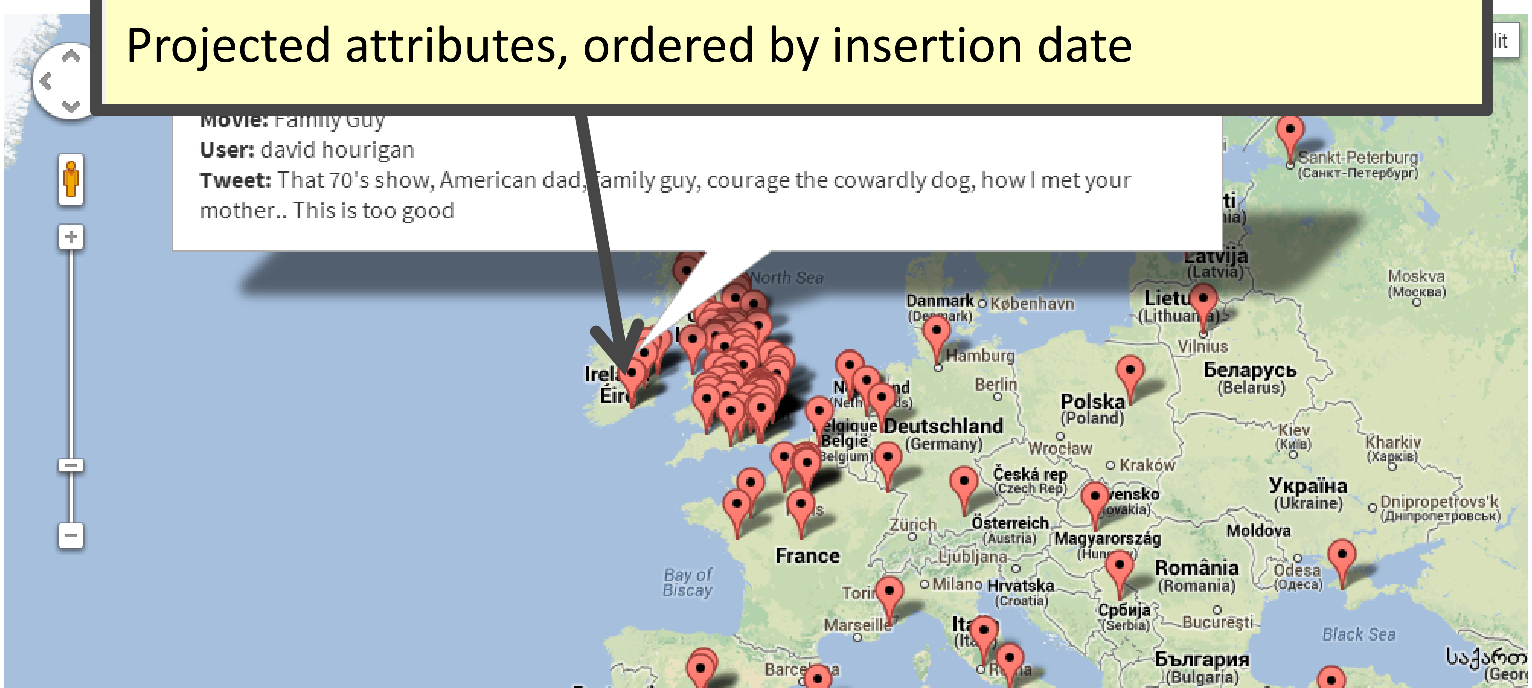
Map

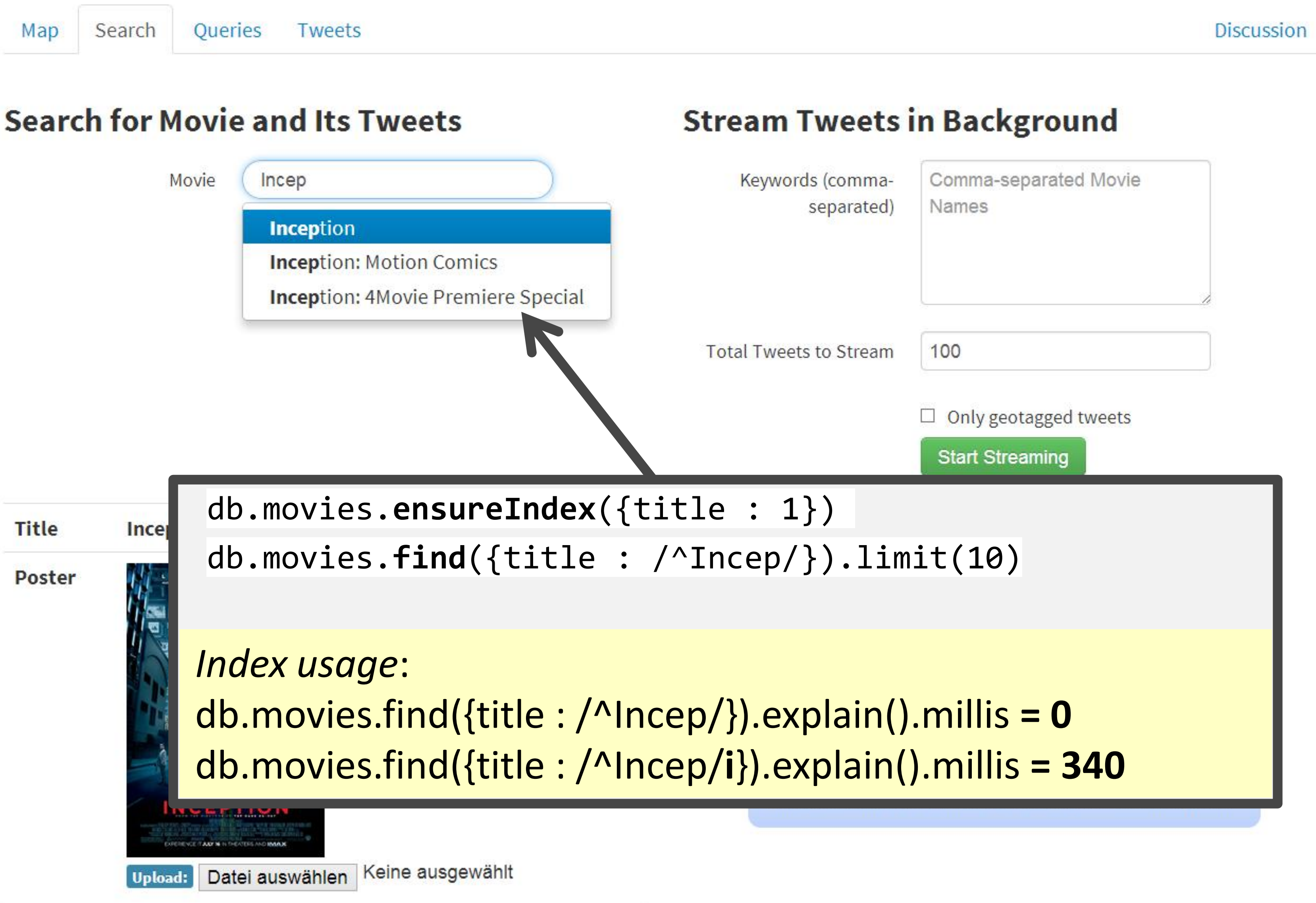
Geot

Note:

```
db.tweets.find({coordinates : {"$exists" : 1}},
  {text:1, movie:1, "user.name":1, coordinates:1})
  .sort({id:-1})
```

Projected attributes, ordered by insertion date






```
db.movies.ensureIndex({title : 1})
db.movies.find({title : /^Incep/}).limit(10)
```

Index usage:

db.movies.find({title : /^Incep/}).explain().millis **= 0**

db.movies.find({title : /^Incep/**i**}).explain().millis **= 340**

Title	Inception
Poster	 Import Poster from IMDB Upload:
Comment	**Editable.** You can edit and save this comment. One of the best movies, that Save
Year	2010
Rating	8.8
Votes	542921
Runtime	148 minutes
Genre	Action,Adventure,Sci-Fi,Thriller
Plot	Dom Cobb is a skilled thief, the absolute best in the dangerous art of extraction, stealing valuable secrets from deep within the subconscious during the dream state, when the mind is at its most vulnerable. Cobb's rare ability has made him a coveted player in this

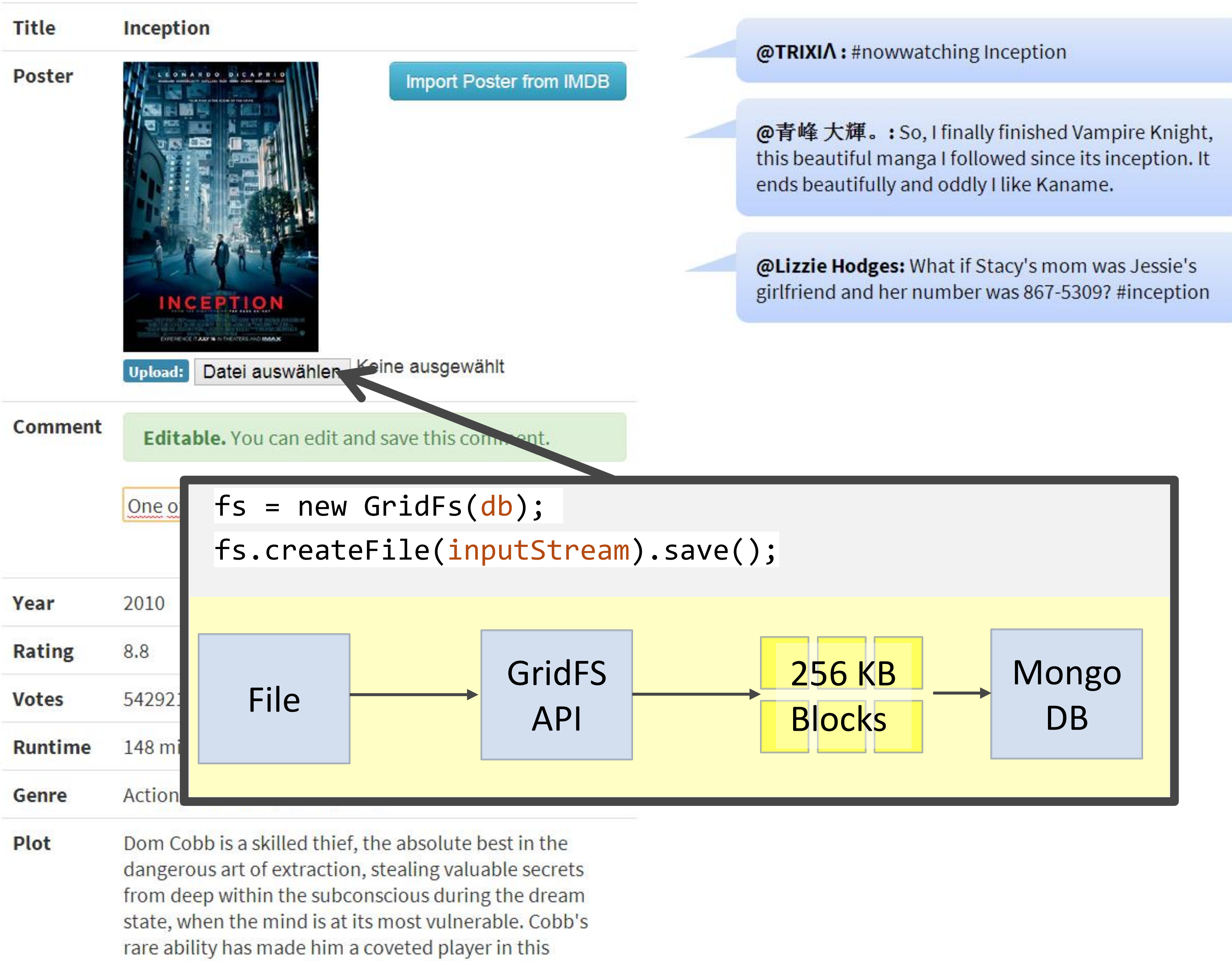
@TRIXIΛ : #nowwatching Inception

@青峰 大輝。: So, I finally finished Vampire Knight, this beautiful manga I followed since its inception. It ends beautifully and oddly I like Kaname.

```
db.movies.update({_id: id), {"$set" : {"comment" : c}})
```

or:

```
db.movies.save(changed_movie);
```



Title Inception
Poster
LEONARDO DICAPRIO
INCEPTION
Import Poster from IMDB
Upload: Datei auswählen Keine ausgewählt
Comment
Editable. You can edit and save this comment.
One o
Year 2010
Rating 8.8
Votes 54292
Runtime 148 mi
Genre Action
Plot Dom Cobb is a skilled thief, the absolute best in the dangerous art of extraction, stealing valuable secrets from deep within the subconscious during the dream state, when the mind is at its most vulnerable. Cobb's rare ability has made him a coveted player in this
@TRIXIΛ : #nowwatching Inception
@青峰 大輝。: So, I finally finished Vampire Knight, this beautiful manga I followed since its inception. It ends beautifully and oddly I like Kaname.
@Lizzie Hodges: What if Stacy's mom was Jessie's girlfriend and her number was 867-5309? #inception
fs = new GridFs(db);
fs.createFile(inputStream).save();
File
GridFS API
256 KB Blocks
Mongo DB

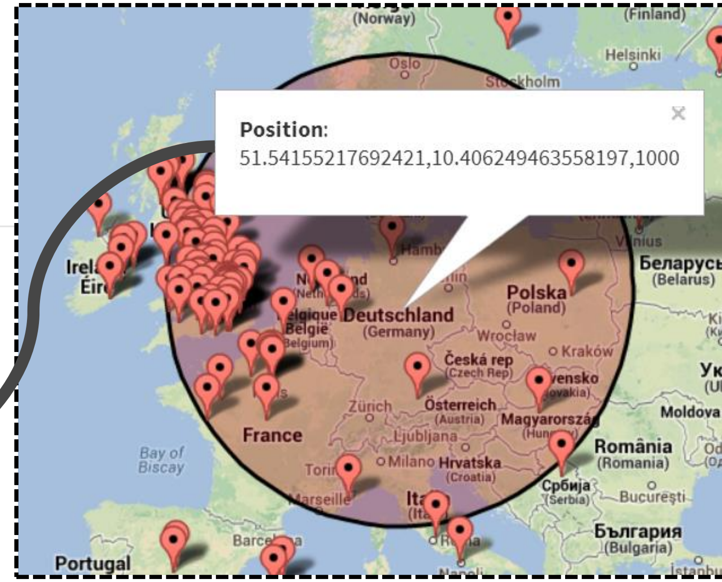
Map Search Queries Tweets

Query Tweets

Query: Get Tweets Near: lat,lng,radius-in-km Go

Parameter: 51.54155217692421,10.406249463558197,1000

Result Limit: 10



User	Tweet	Created at	Coordinates
MitchellyMonica			
J. Z.			
Party Hardy			
nadine stachowiak		2013	
PAIRSonnalites | DE	ES-News : http://t.co/WhBmNr6OM2: In vielen Staaten Afrikas gibt es	Wed May 29	13.46757813,52.5913198

```
db.tweets.ensureIndex({coordinates : "2dsphere"})
db.tweets.find({"$near" : {"$geometry" : … }})
```

Geospatial Queries:

- **Distance**
- **Intersection**
- **Inclusion**

Map Search Queries Tweets Discussion

Query Tweets

Query: Indexed Fulltext Search on Tweets [Go]

Parameter: StAr trek

Result Limit: 100

Show 25 search results per page

Filter search results:

```
db.tweets.runCommand( "text", { search: "StAr trek" } )
```

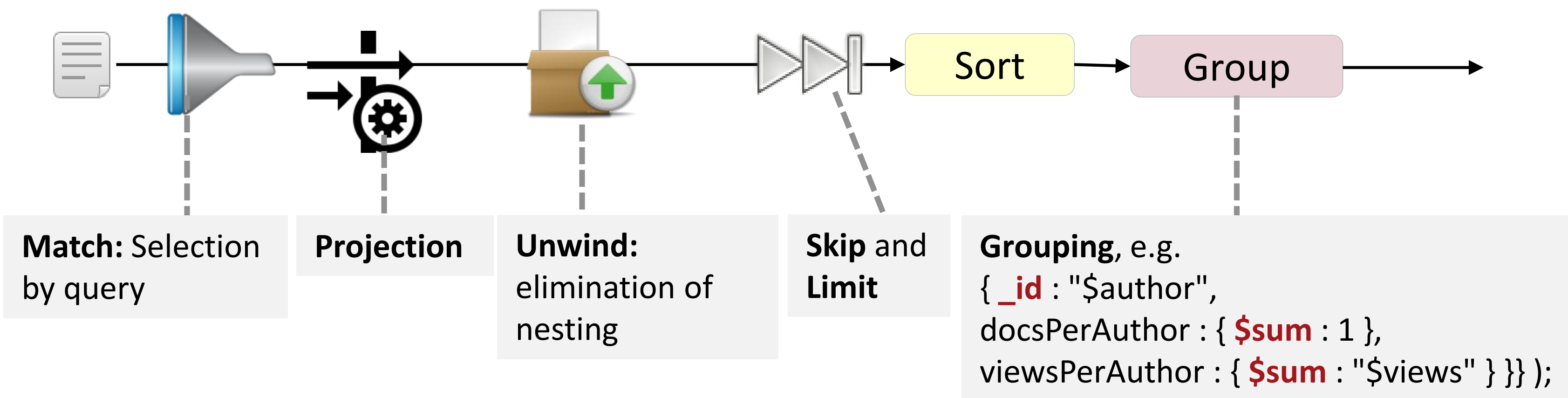
Full-text Search:

- **Tokenization, Stop Words**
- **Stemming**
- **Scoring**

User	Tweet	Created at	Coordinates
manwonman			
Mia Clrss Hrnndz ♡			
A N G G I_			
Stefany Ezra Elvina		2013	
Vanessa Yung	Star Trek into Darkness☐	Wed May 29 19:21:06 +0000 2013	-2.986771,53.404051
tam wilson	Finally getting to see Star Trek! (at @DCADundee Contemporary Arts for Star Trek Into Darkness 3D) http://t.co/0ojg4KMBL5	Wed May 29 18:48:56 +0000	-2.97489166,56.45753477

Analytic Capabilities

- Aggregation Pipeline Framework:

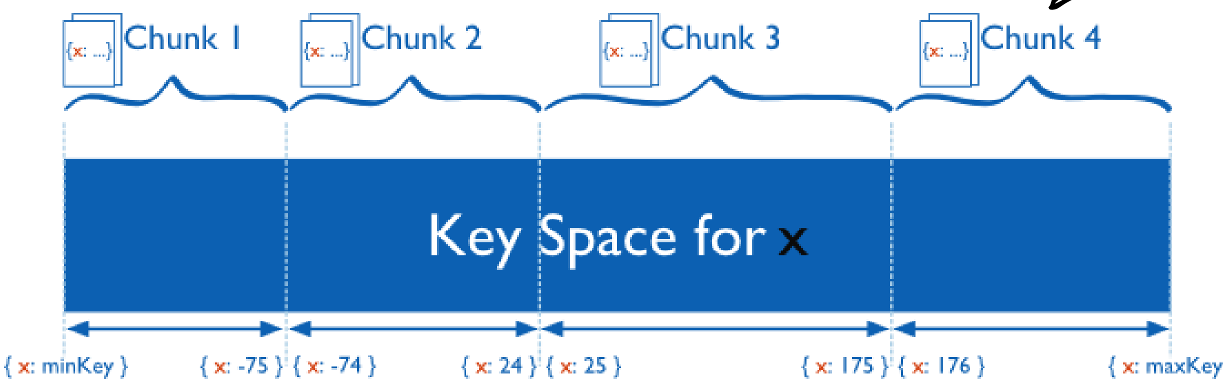


- Alternative: JavaScript MapReduce

Sharding

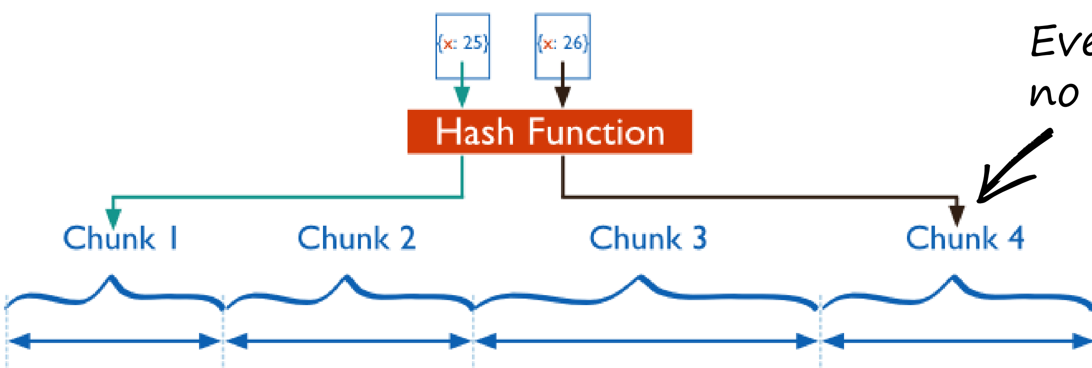
In optimal case only one shard asked per query, else: Scatter-and-gather

- Range-based:



- Hash-based:

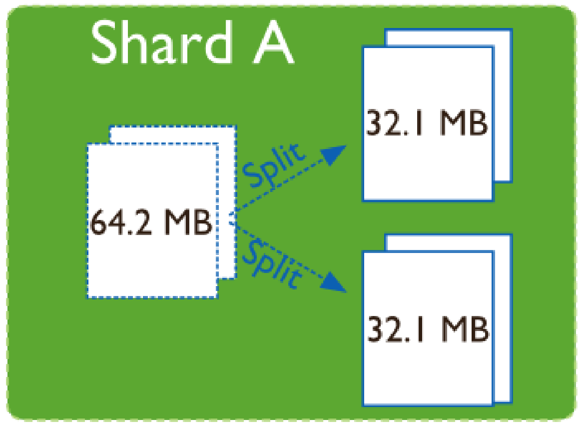
Even distribution, no locality



docs.mongodb.org/manual/core/sharding-introduction/

Sharding

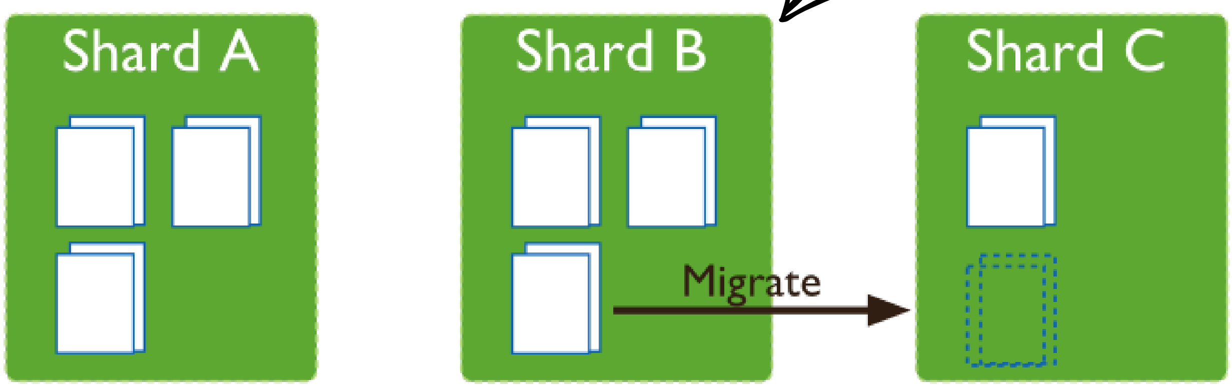
- Splitting:



Split chunks that are too large

- Migration:

Mongos Load Balancer triggers rebalancing

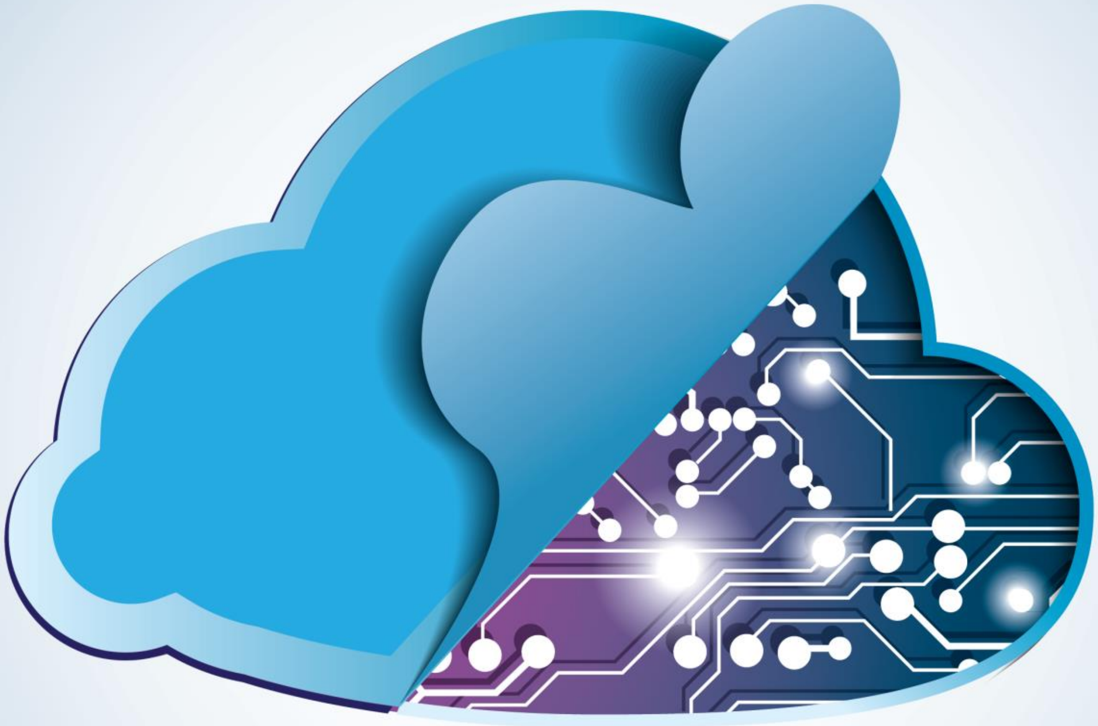


docs.mongodb.org/manual/core/sharding-introduction/

Classification: MongoDB

Techniques

Sharding	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
Replication	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
Storage Management	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
Query Processing	Global Index	Local Index	Query Planning	Analytics	Materialized Views



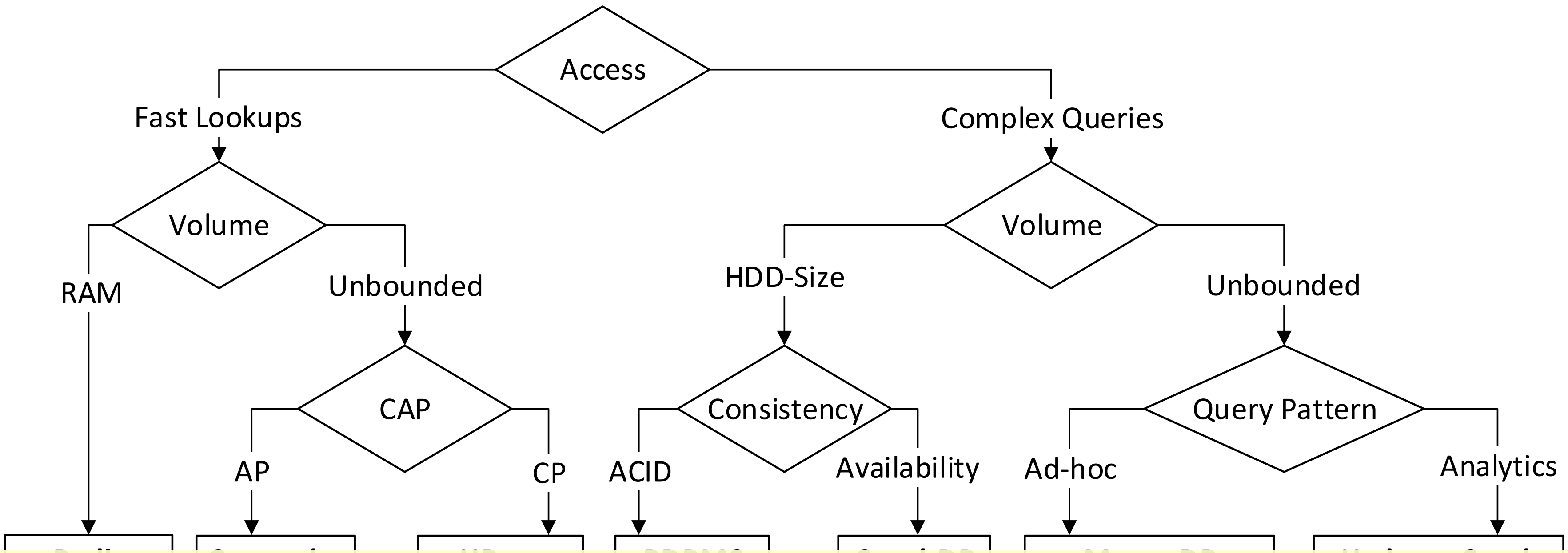
How can the choices for an appropriate system be narrowed down?

Outline

- NoSQL Foundations and Motivation
- The NoSQL Toolbox: Common Techniques
- NoSQL Systems
- Decision Guidance: NoSQL Decision Tree

- Decision Tree
- Classification Summary
- Literature Reommendations

NoSQL Decision Tree



Purpose:

Application Architects: narrowing down the potential system candidates based on requirements

Database Vendors/Researchers: clear communication and design of system trade-offs

System Properties

According to the NoSQL Toolbox

- For fine-grained system selection:

	Functional Requirements							
	Scan Queries	ACID Transactions	Conditional Writes	Joins	Sorting	Filter Query	Full-Text Search	Analytics
Mongo	x		x		x	x	x	x
Redis	x	x	x					
HBase	x		x		x			x
Riak							x	x
Cassandra	x		x		x		x	x
MySQL	x	x	x	x	x	x	x	x

System Properties
According to the NoSQL Toolbox

- For fine-grained system selection:

	Non-functional Requirements										
	Data Scalability	Write Scalability	Read Scalability	Elasticity	Consistency	Write Latency	Read Latency	Write Throughput	Read Availability	Write Availability	Durability
Mongo	x	x	x		x	x	x		x		x
Redis			x		x	x	x	x	x		x
HBase	x	x	x	x	x	x		x			x
Riak	x	x	x	x		x	x	x	x	x	x
Cassandra	x	x	x	x		x		x	x	x	x
MySQL			x		x						x

System Properties

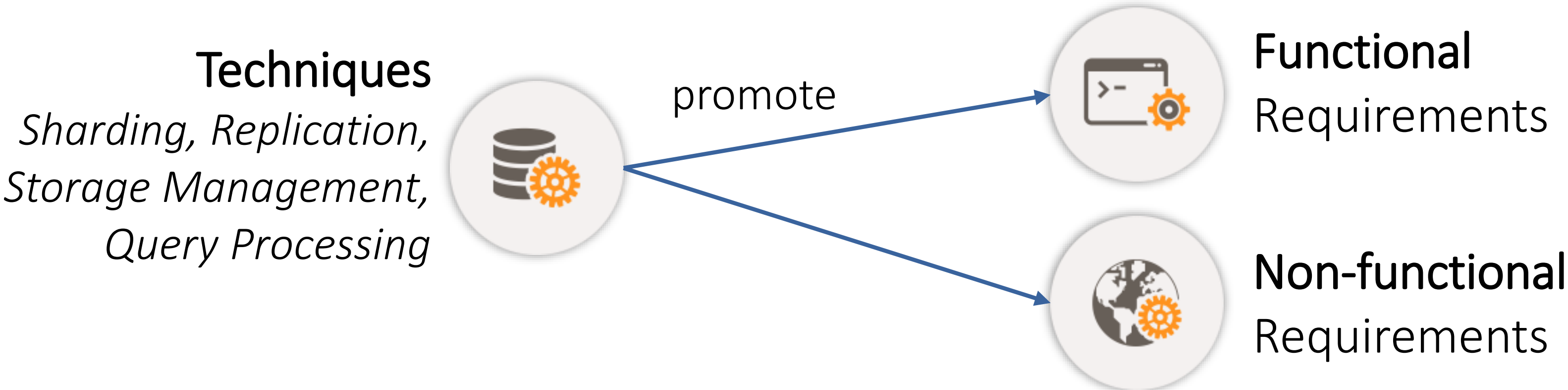
According to the NoSQL Toolbox

- For fine-grained system selection:

	Techniques																			
	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared-Disk	Transaction Protocol	Sync. Replication	Async. Replication	Primary Copy	Update Anywhere	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage	Global Indexing	Local Indexing	Query Planning	Analytics Framework	Materialized Views
Mongo	x	x					x	x	x		x		x	x	x		x	x	x	
Redis								x	x		x		x							
HBase	x						x		x		x		x		x					
Riak		x		x				x		x	x	x	x			x	x		x	
Cassandra		x		x				x		x	x		x		x	x	x			x
MySQL					x			x	x		x	x	x				x	x		

Summary

- High-Level NoSQL Categories:
  - Key-Value, Wide-Column, Docuement, Graph
  - Two out of {Consistent, Available, Partition Tolerant}
- The **NoSQL Toolbox**: systems use similar techniques that promote certain capabilities



- **Decision Tree**

Our NoSQL research at the University of Hamburg

Why performance matters

Average: 9,3s

Loading…

Conversions

% Traffic

% Visitors

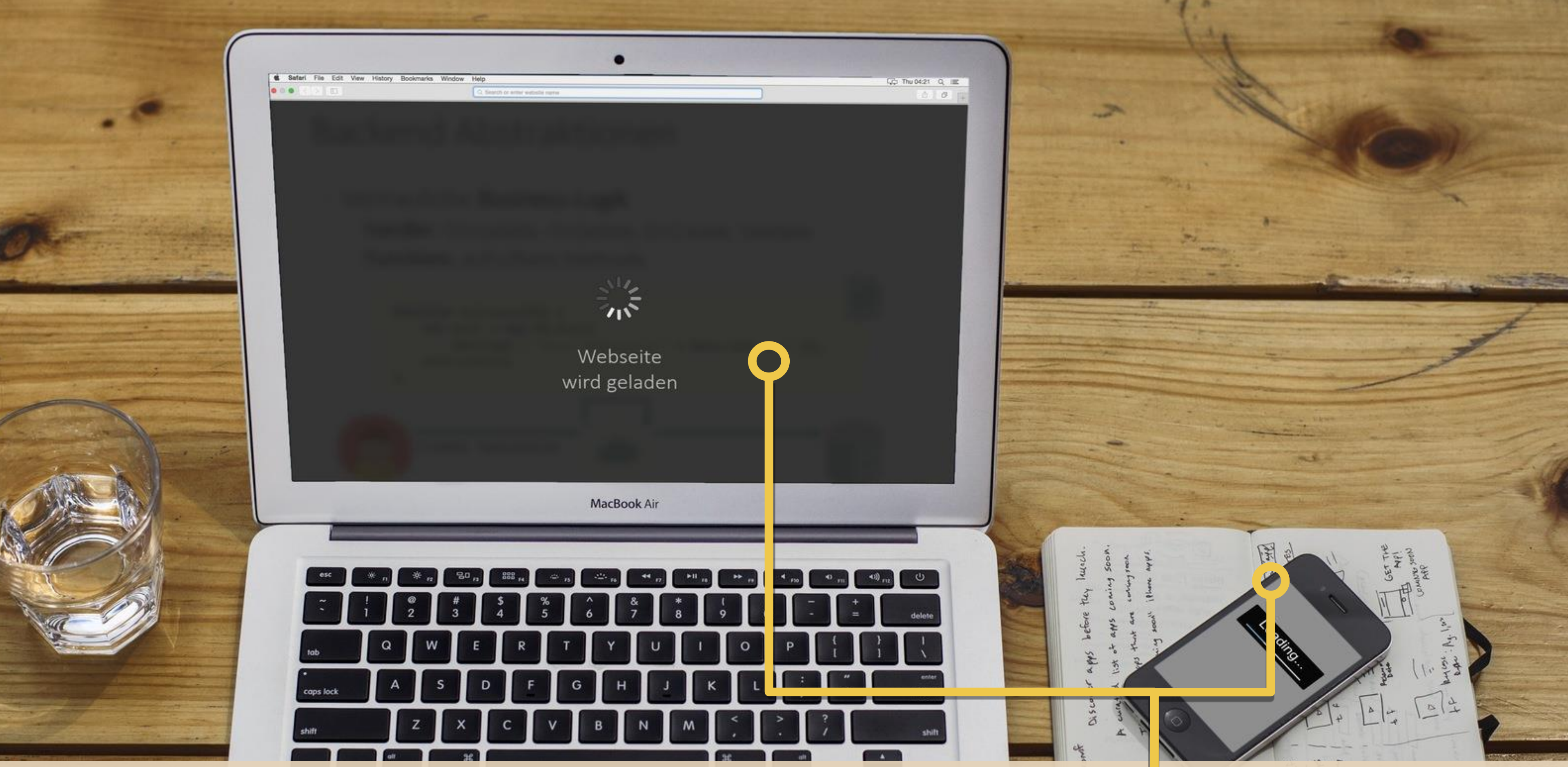
% Revenue

Aberdeen Group

Google

Yahoo!

amazon.com

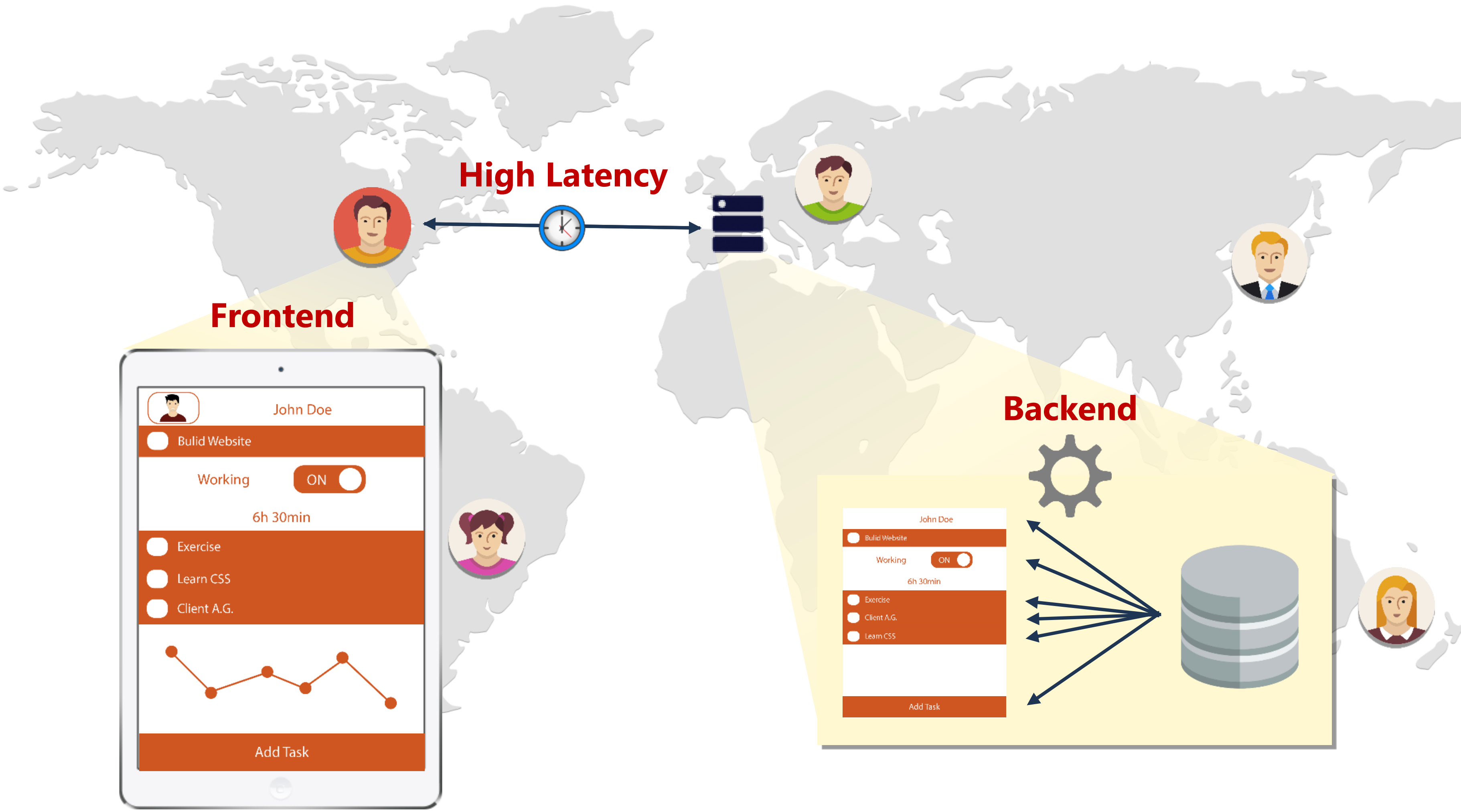


If perceived speed is such an important factor

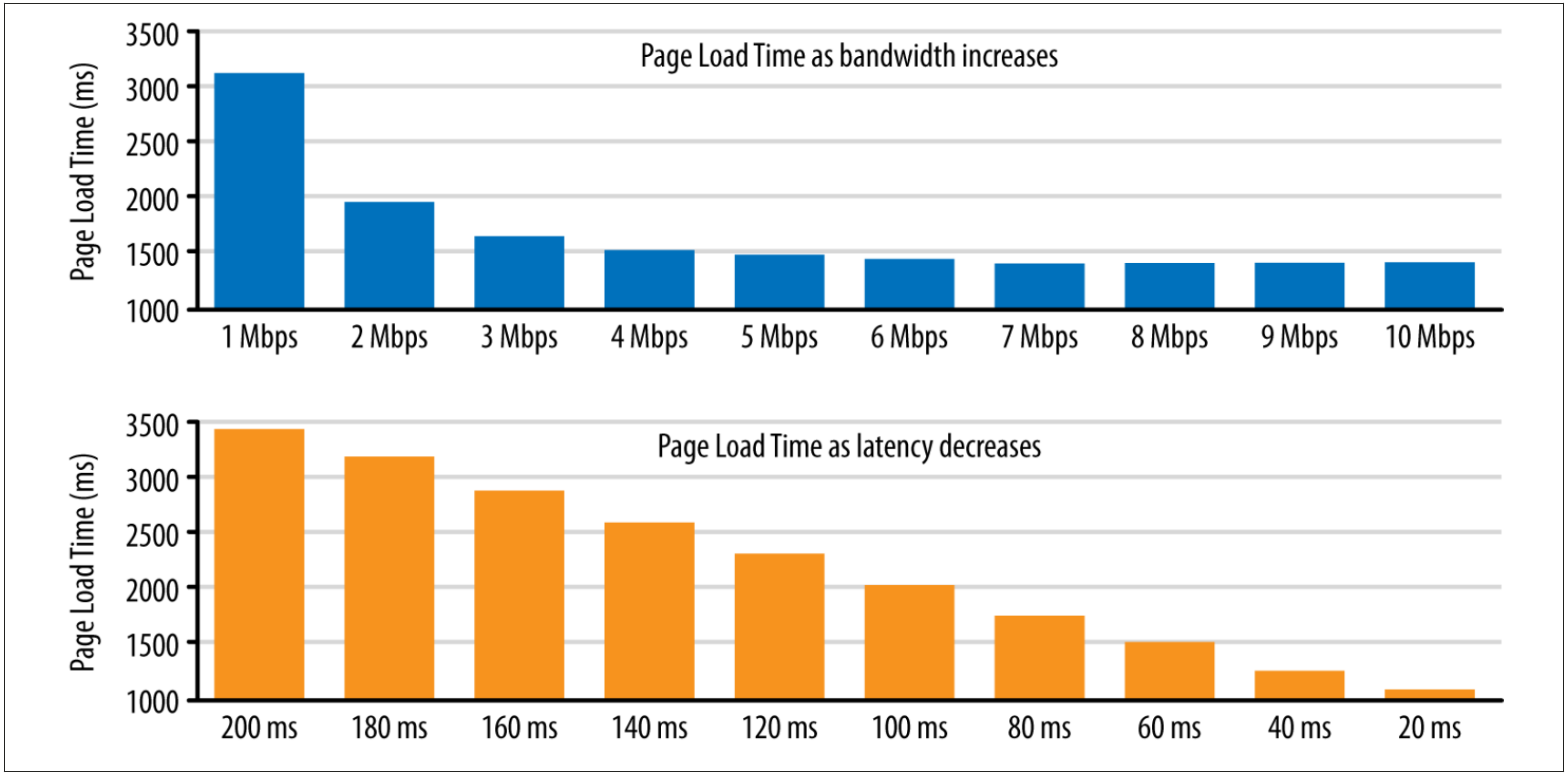
...what causes slow page load times?

The Problem

Three Bottlenecks: Latency, Backend & Frontend

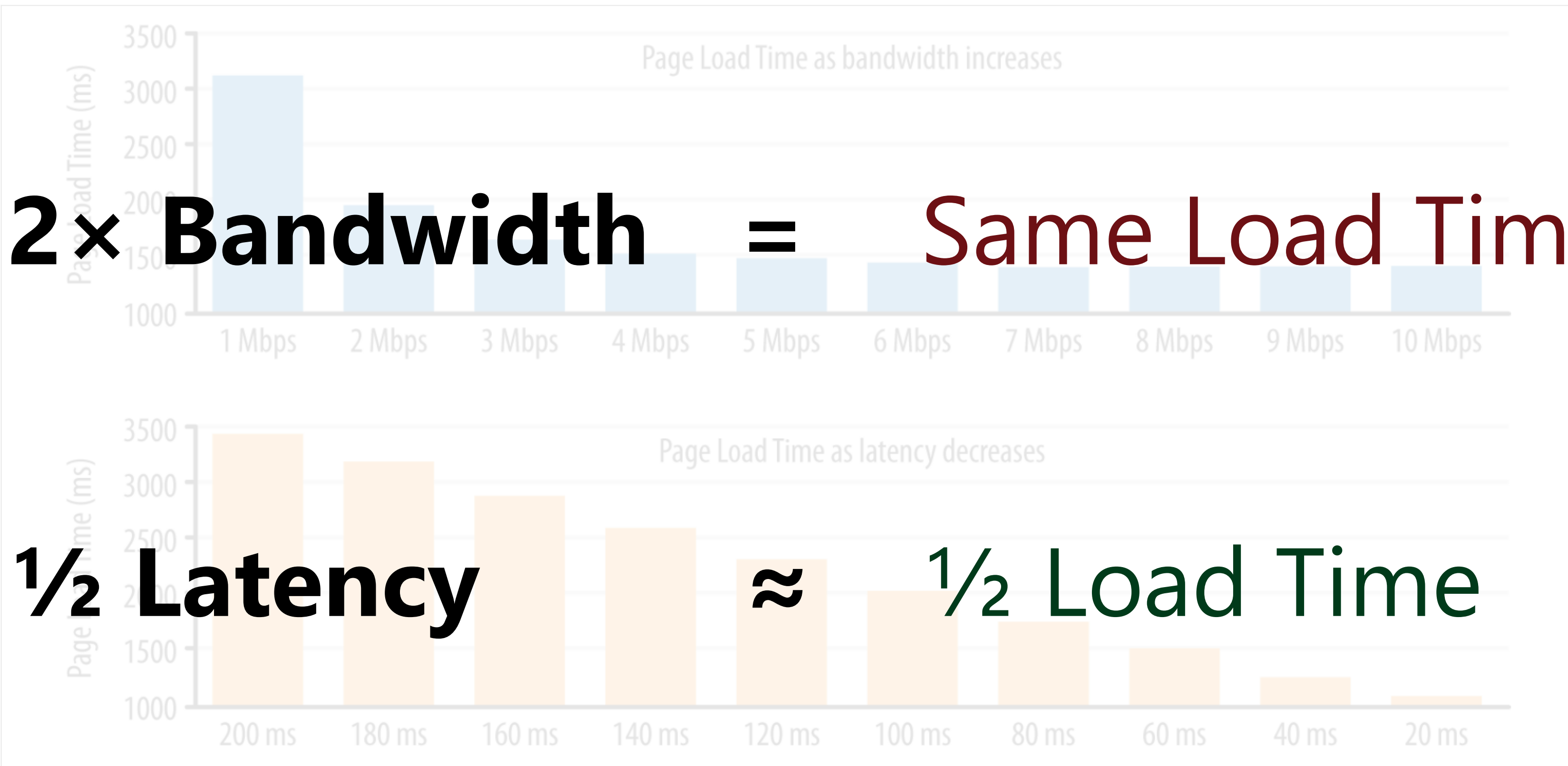


Network Latency: Impact



I. Grigorik, High performance browser networking. O'Reilly Media, 2013.

Network Latency: Impact



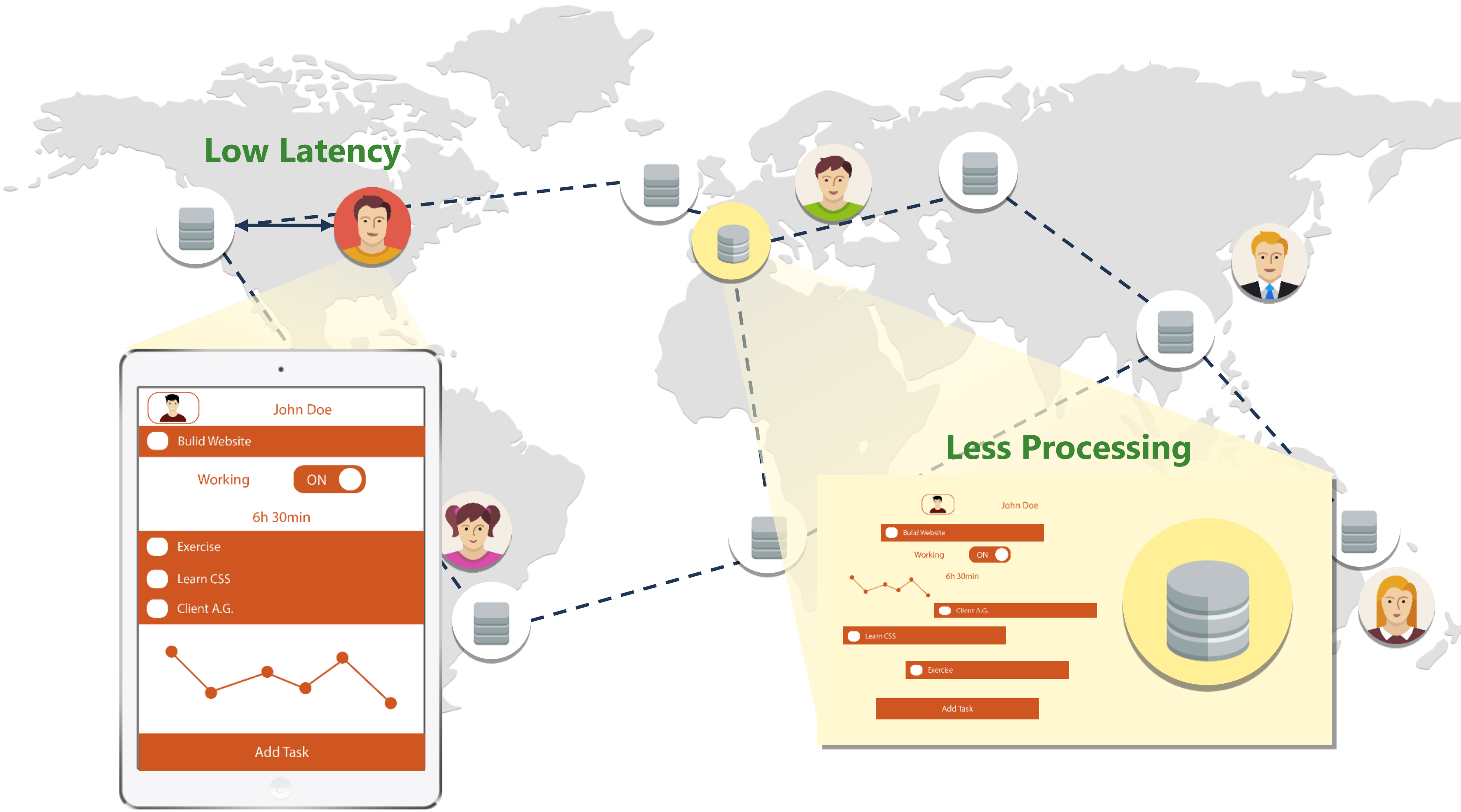
2× Bandwidth = Same Load Time

½ Latency ≈ ½ Load Time

I. Grigorik, High performance browser networking. O'Reilly Media, 2013.

Goal: Low-Latency for Dynamic Content

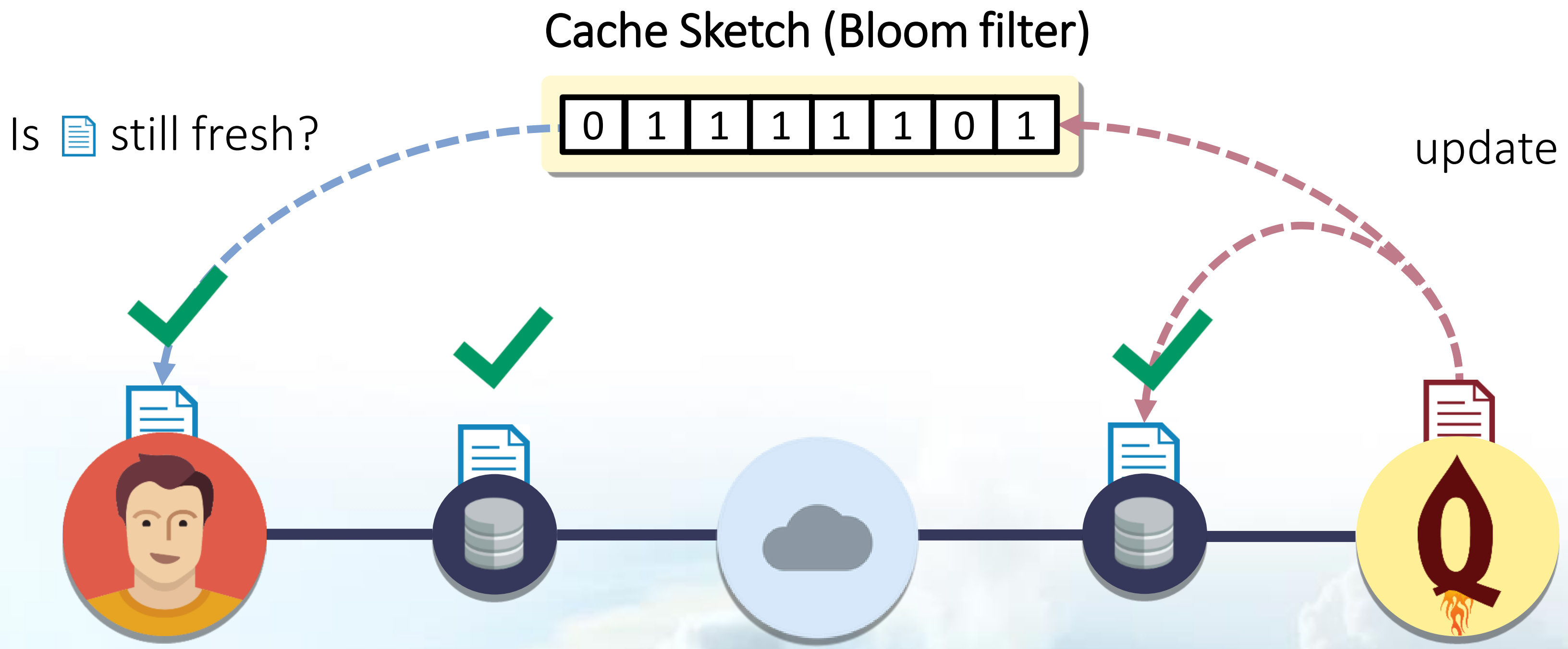
By Serving Data from Ubiquitous Web Caches



In a nutshell

In a nutshell

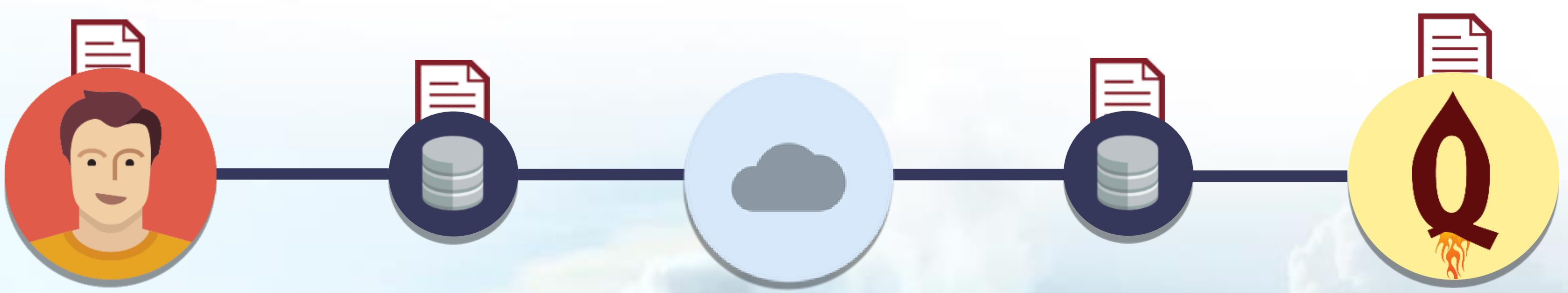
Solution: Proactively Revalidate Data



Innovation

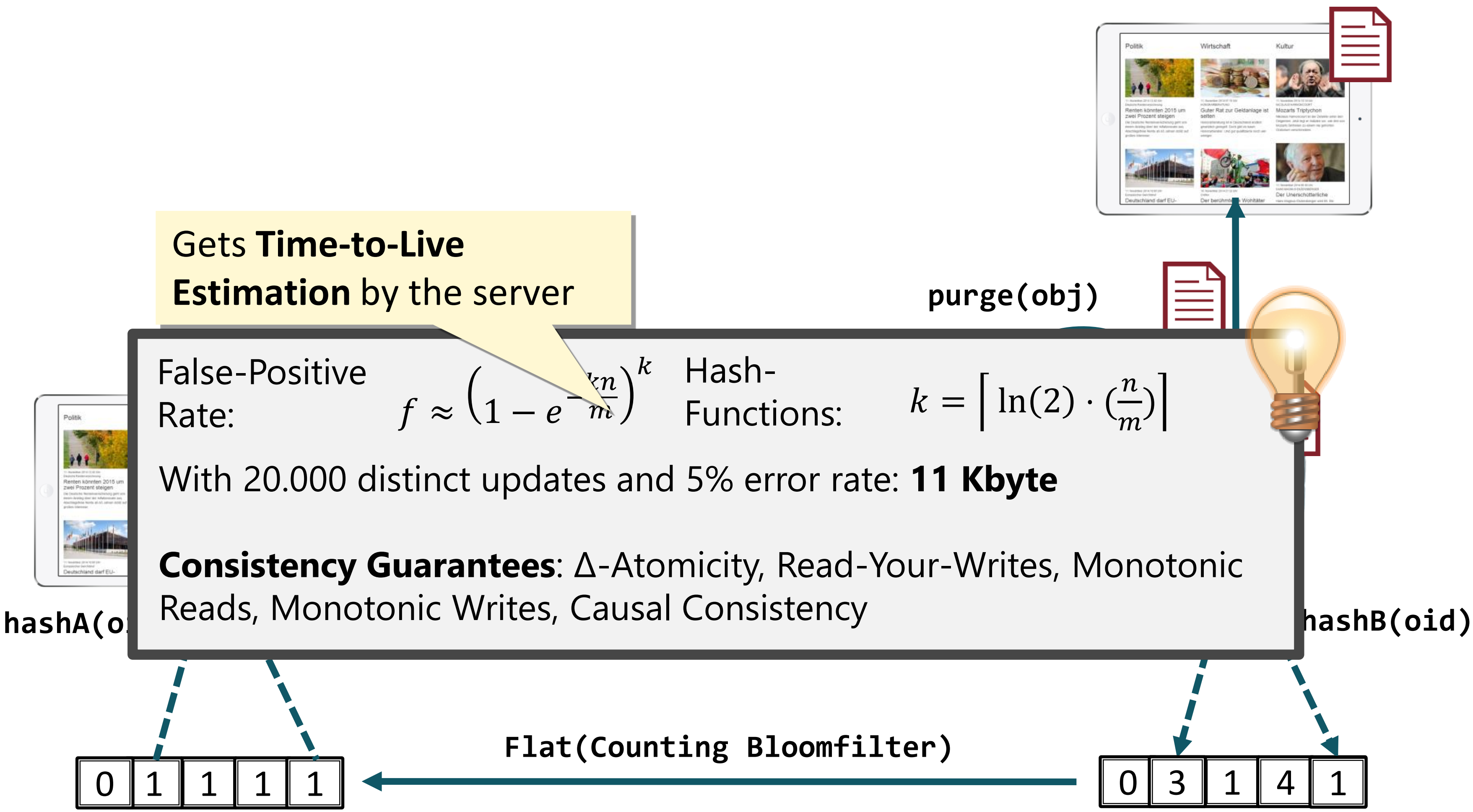
Solution: Proactively Revalidate Data

- F. Gessert, F. Bücklers, und N. Ritter, „ORESTES: a Scalable Database-as-a-Service Architecture for Low Latency“, in *CloudDB 2014*, 2014.
- F. Gessert und F. Bücklers, „ORESTES: ein System für horizontal skalierbaren Zugriff auf Cloud-Datenbanken“, in Informatiktage 2013, 2013.
- F. Gessert und F. Bücklers, *Performanz- und Reaktivitätssteigerung von OODBMS vermittels der Web-Caching-Hierarchie*. Bachelorarbeit, 2010.
- M. Schaarschmidt, F. Gessert, und N. Ritter, „Towards Automated Polyglot Persistence“, in BTW 2015.
- S. Friedrich, W. Wingerath, F. Gessert, und N. Ritter, „NoSQL OLTP Benchmarking: A Survey“, in *44. Jahrestagung der Gesellschaft für Informatik*, 2014, Bd. 232, S. 693–704.
- F. Gessert, S. Friedrich, W. Wingerath, M. Schaarschmidt, und N. Ritter, „Towards a Scalable and Unified REST API for Cloud Data Stores“, in *44. Jahrestagung der GI*, Bd. 232, S. 723–734.
- F. Gessert, M. Schaarschmidt, W. Wingerath, S. Friedrich, und N. Ritter, „The Cache Sketch: Revisiting Expiration-based Caching in the Age of Cloud Data Management“, in BTW 2015.
- F. Gessert und F. Bücklers, *Kohärentes Web-Caching von Datenbankobjekten im Cloud Computing*. Masterarbeit 2012.
- W. Wingerath, S. Friedrich, und F. Gessert, „Who Watches the Watchmen? On the Lack of Validation in NoSQL Benchmarking“, in BTW 2015.
- F. Gessert, „Skalierbare NoSQL- und Cloud-Datenbanken in Forschung und Praxis“, BTW 2015



Bloom filters for Caching

End-to-End Example



Gets **Time-to-Live Estimation** by the server

False-Positive Rate: $f \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$

Hash-Functions: $k = \left\lceil \ln(2) \cdot \left(\frac{n}{m}\right) \right\rceil$

With 20.000 distinct updates and 5% error rate: **11 Kbyte**

Consistency Guarantees: Δ-Atomicity, Read-Your-Writes, Monotonic Reads, Monotonic Writes, Causal Consistency

Orestes Architecture

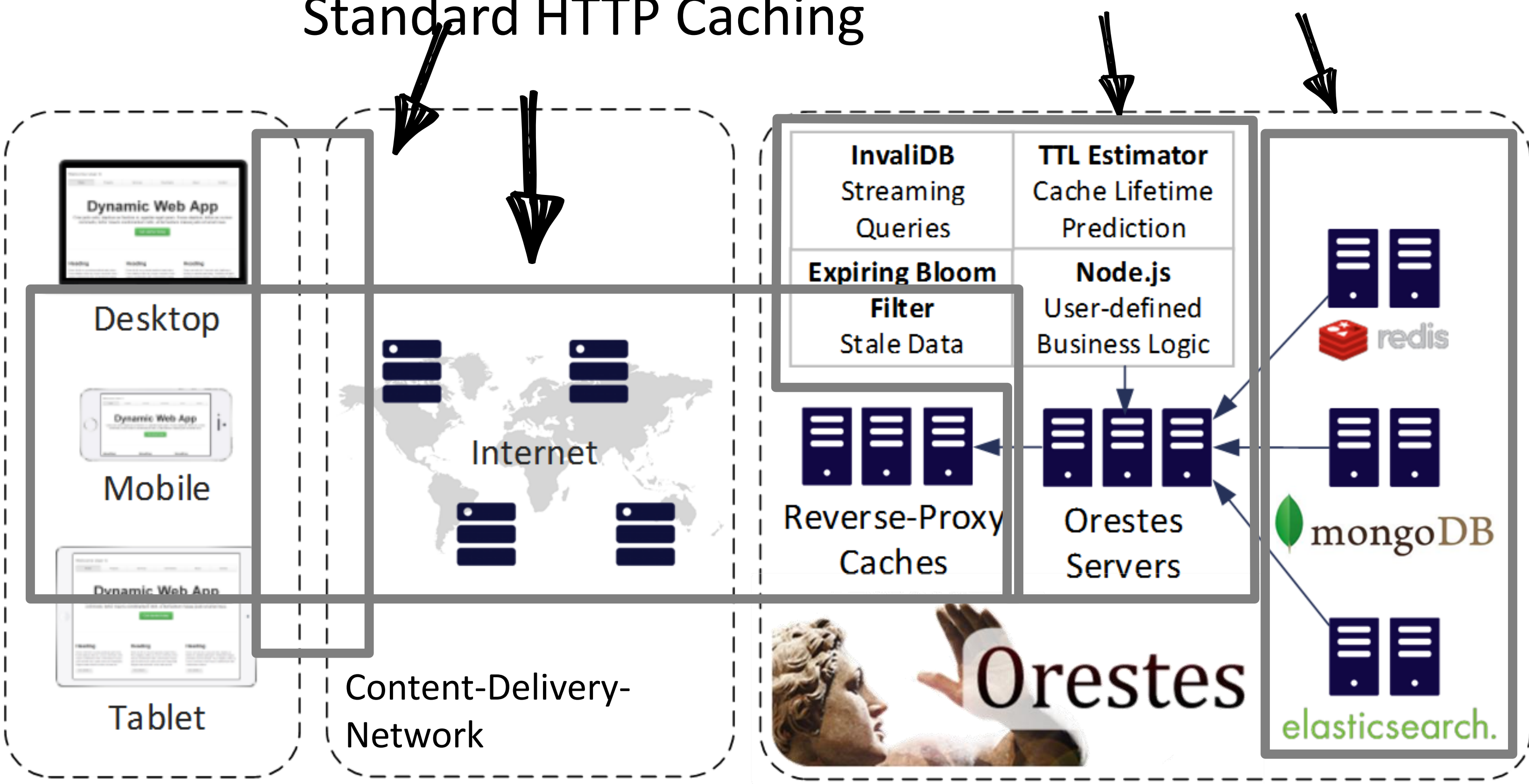
Infrastructure

Backend-as-a-Service Middleware:
Caching, Transactions, Schemas,
Invalidation Detection, …

Polyglot Storage

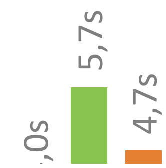
Unified REST API

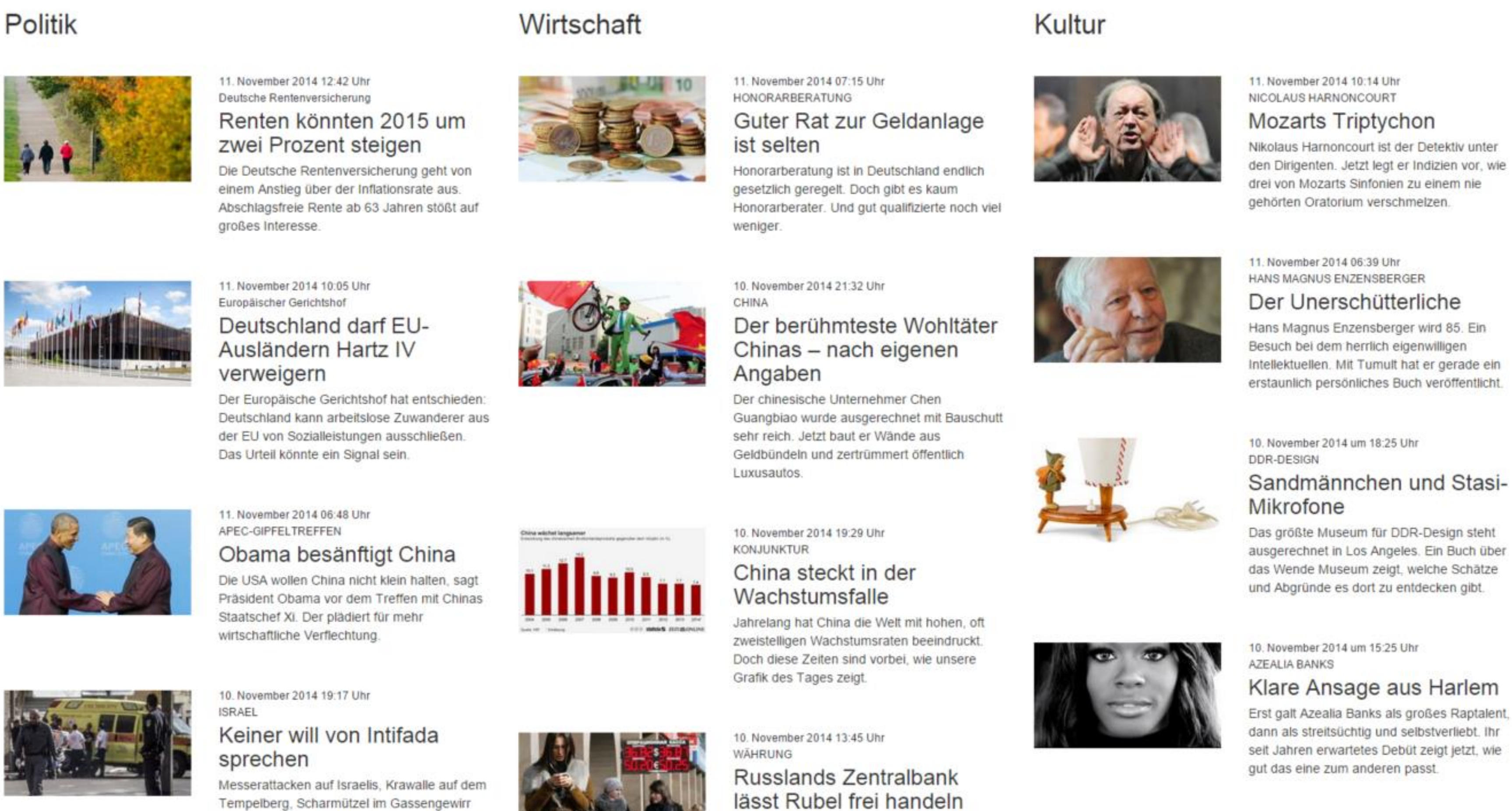
Standard HTTP Caching



Page-Load Times

What impact does caching have in practice?





Politik

11. November 2014 12:42 Uhr
Deutsche Rentenversicherung
Renten könnten 2015 um zwei Prozent steigen
Die Deutsche Rentenversicherung geht von einem Anstieg über der Inflationsrate aus. Abschlagsfreie Rente ab 63 Jahren stößt auf großes Interesse.

11. November 2014 10:05 Uhr
Europäischer Gerichtshof
Deutschland darf EU-Ausländern Hartz IV verweigern
Der Europäische Gerichtshof hat entschieden: Deutschland kann arbeitslose Zuwanderer aus der EU von Sozialleistungen ausschließen. Das Urteil könnte ein Signal sein.

11. November 2014 06:48 Uhr
APEC-GIPFELTREFFEN
Obama besänftigt China
Die USA wollen China nicht klein halten, sagt Präsident Obama vor dem Treffen mit Chinas Staatschef Xi. Der plädiert für mehr wirtschaftliche Verflechtung.

10. November 2014 19:17 Uhr
ISRAEL
Keiner will von Intifada sprechen
Messerattacken auf Israelis, Krawalle auf dem Tempelberg, Scharmützel im Gassengewirr

Wirtschaft

11. November 2014 07:15 Uhr
HONORARBERATUNG
Guter Rat zur Geldanlage ist selten
Honorarberatung ist in Deutschland endlich gesetzlich geregelt. Doch gibt es kaum Honorarberater. Und gut qualifizierte noch viel weniger.

10. November 2014 21:32 Uhr
CHINA
Der berühmteste Wohltäter Chinas – nach eigenen Angaben
Der chinesische Unternehmer Chen Guangbiao wurde ausgerechnet mit Bauschutt sehr reich. Jetzt baut er Wände aus Geldbündeln und zertrümmert öffentlich Luxusautos.

10. November 2014 19:29 Uhr
KONJUNKTUR
China steckt in der Wachstumsfalle
Jahrelang hat China die Welt mit hohen, oft zweistelligen Wachstumsraten beeindruckt. Doch diese Zeiten sind vorbei, wie unsere Grafik des Tages zeigt.

10. November 2014 13:45 Uhr
WÄHRUNG
Russlands Zentralbank lässt Rubel frei handeln

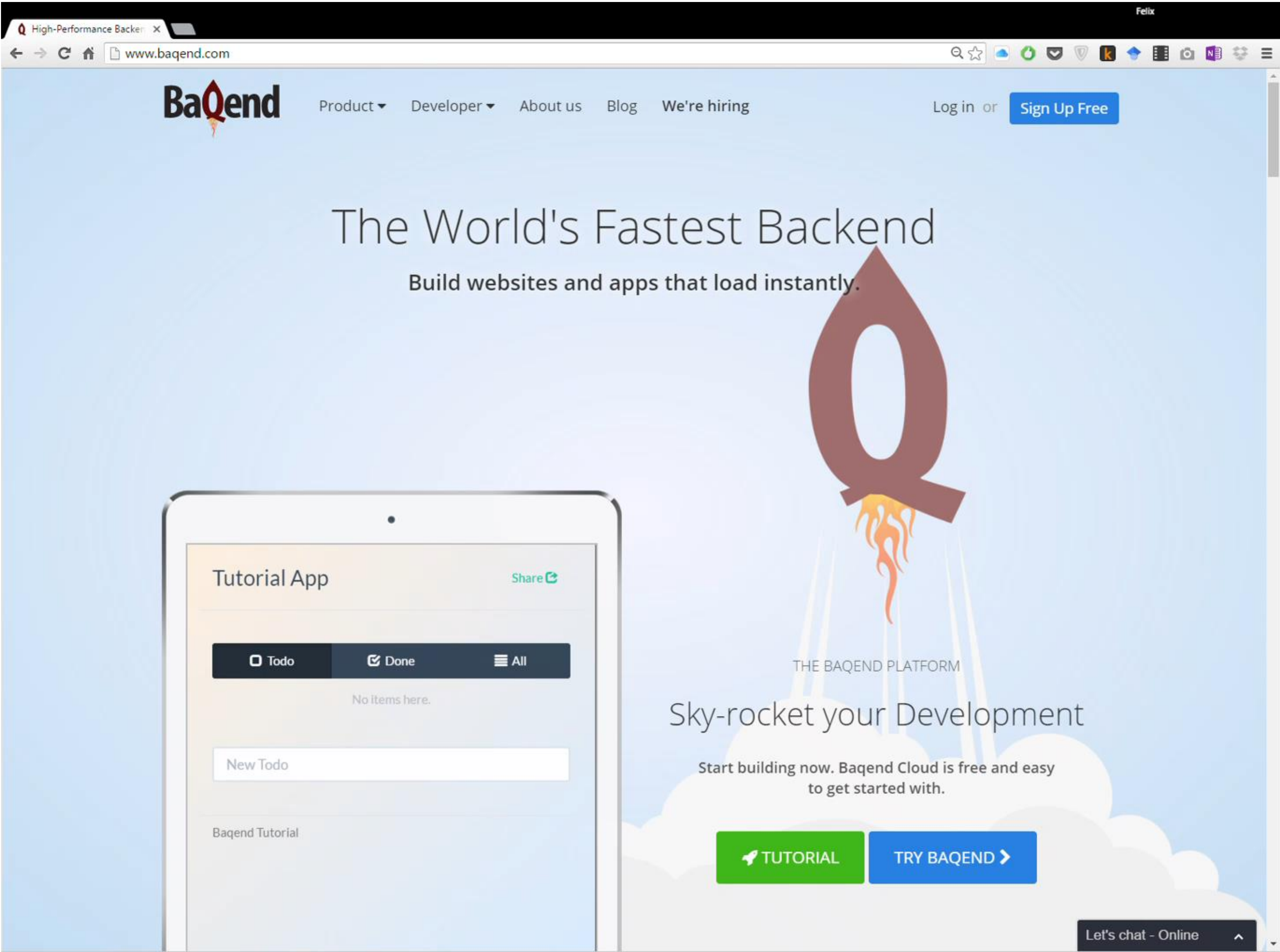
Kultur

11. November 2014 10:14 Uhr
NICOLAUS HARNONCOURT
Mozarts Triptychon
Nikolaus Harnoncourt ist der Detektiv unter den Dirigenten. Jetzt legt er Indizien vor, wie drei von Mozarts Sinfonien zu einem nie gehörten Oratorium verschmelzen.

11. November 2014 06:39 Uhr
HANS MAGNUS ENZENSBERGER
Der Unerschütterliche
Hans Magnus Enzensberger wird 85. Ein Besuch bei dem herrlich eigenwilligen Intellektuellen. Mit Tumult hat er gerade ein erstaunlich persönliches Buch veröffentlicht.

10. November 2014 um 18:25 Uhr
DDR-DESIGN
Sandmännchen und Stasi-Mikrofone
Das größte Museum für DDR-Design steht ausgerechnet in Los Angeles. Ein Buch über das Wende Museum zeigt, welche Schätze und Abgründe es dort zu entdecken gibt.

10. November 2014 um 15:25 Uhr
AZEALIA BANKS
Klare Ansage aus Harlem
Erst galt Azealia Banks als großes Raptalent, dann als streitsüchtig und selbstverliebt. Ihr seit Jahren erwartetes Debüt zeigt jetzt, wie gut das eine zum anderen passt.


Felix
High-Performance Backen
www.baqend.com
BaQend
Product
Developer
About us
Blog
We're hiring
Log in or
Sign Up Free
The World's Fastest Backend
Build websites and apps that load instantly.
Tutorial App
Share
Todo
Done
All
No items here.
New Todo
Baqend Tutorial
THE BAQEND PLATFORM
Sky-rocket your Development
Start building now. Baqend Cloud is free and easy
to get started with.
TUTORIAL
TRY BAQEND
Let's chat - Online

Want to try Baqend?

- Free **Baqend Cloud** Instance at baqend.com
- Download **Community Edition**

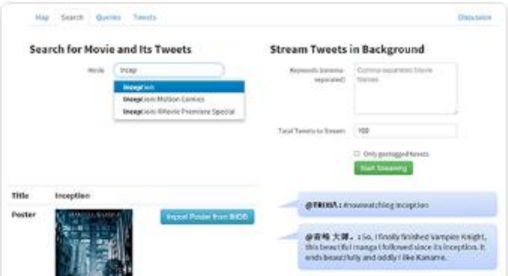
WebLabs.Hamburg

Ideen brauchen Software-Lösungen, die sie zum Fliegen bringen. Wir sind ein Team von Entwicklern und realisieren individuelle Web-Anwendungen und mobile Apps. Unsere Kompetenzen reichen von der Frontend-Entwicklung bis zu hochskalierbaren Backends. Wir arbeiten eng mit Baqend zusammen.

Was wir machen | Was wir bieten | Wer wird sind | Kontakt

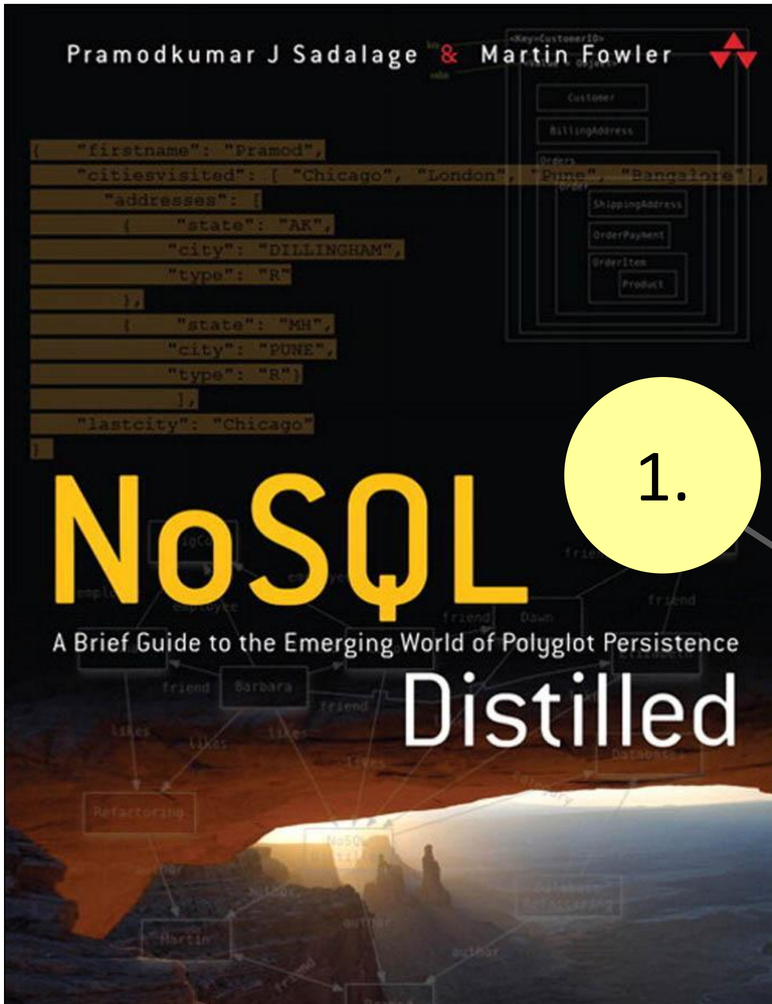
Wir entwickeln Webseiten und Apps.

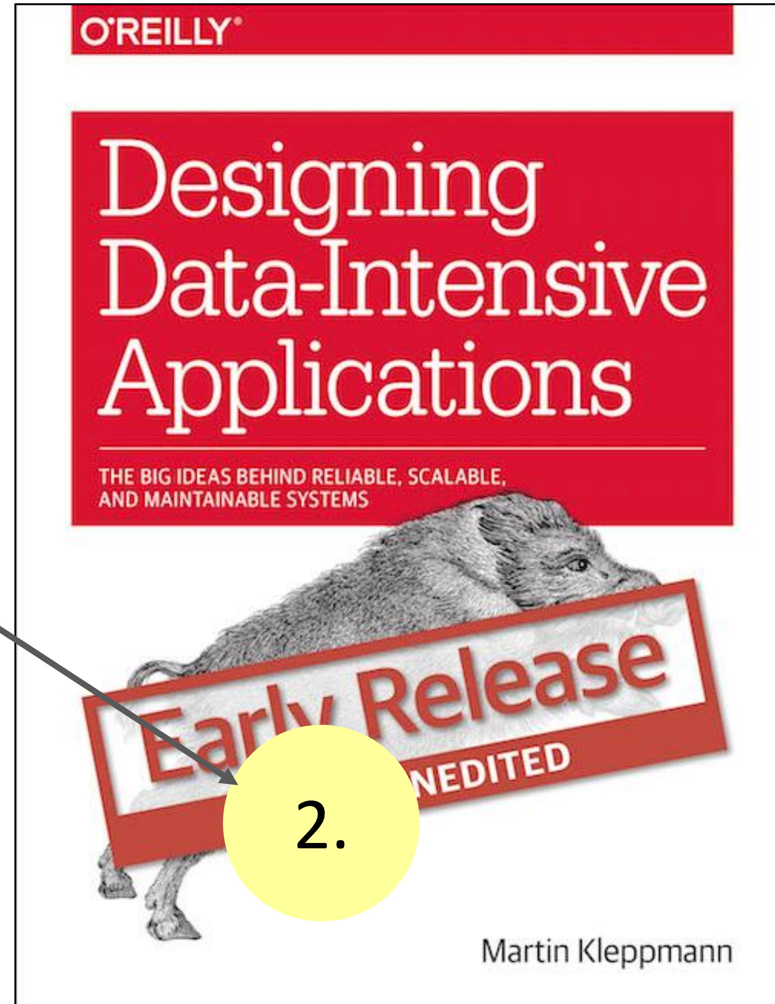
Wir realisieren für Unternehmen Web und App-Projekte. Unsere besonderere Expertise liegt in schnellen Ladezeiten und hoher Skalierbarkeit. Das Team besteht sowohl aus Spezialisten für Datenbanken, Backend- und Big Data



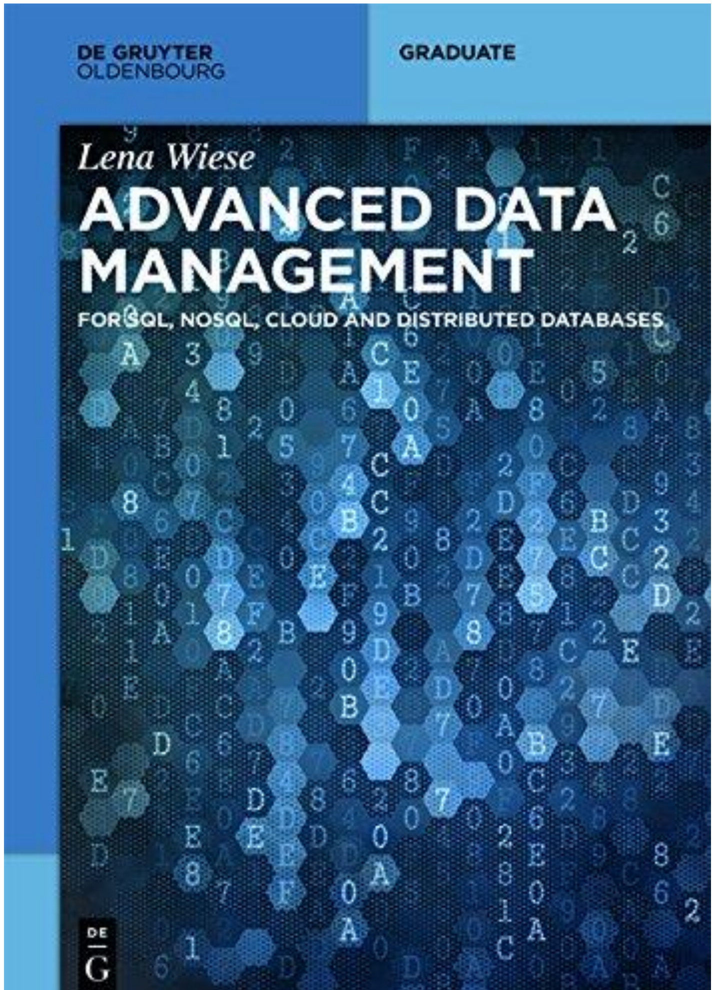
Literature Recommendations

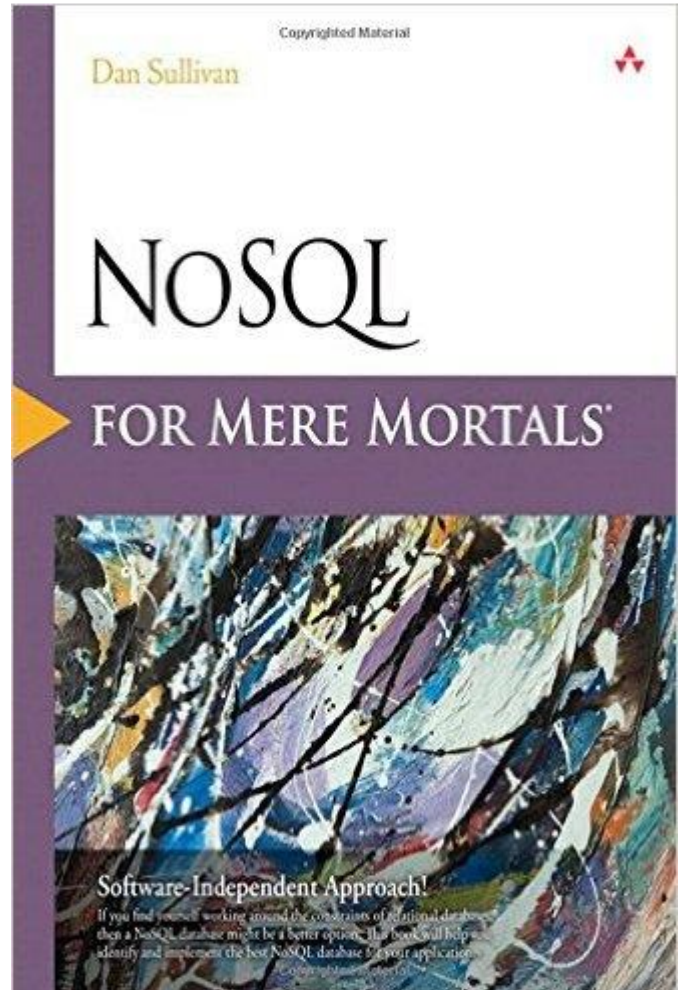
Recommended Literature



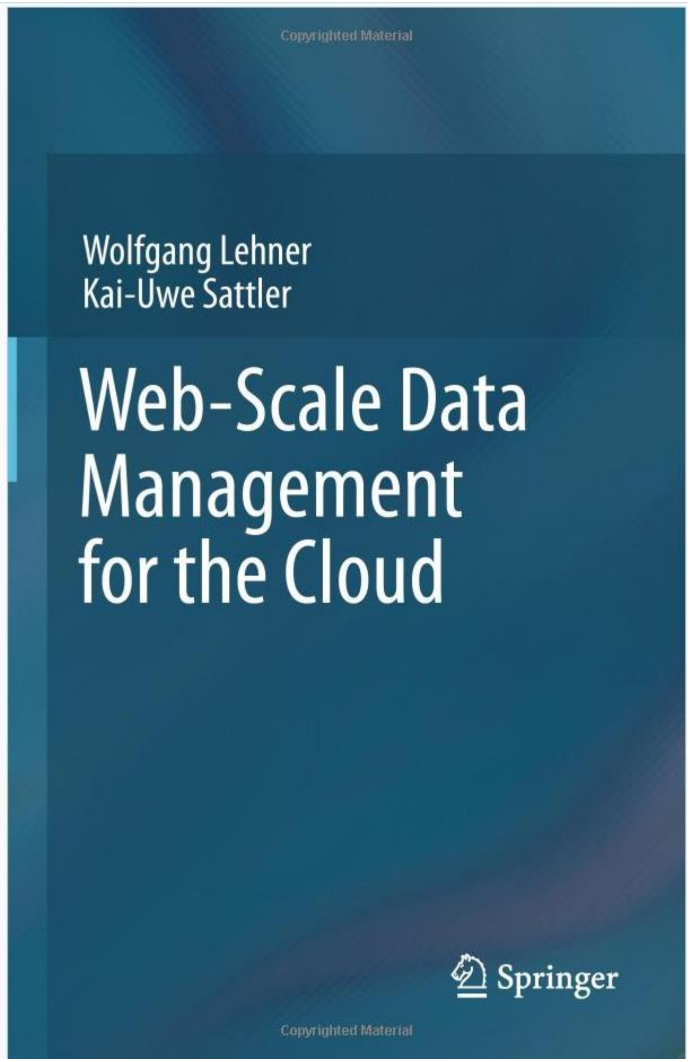


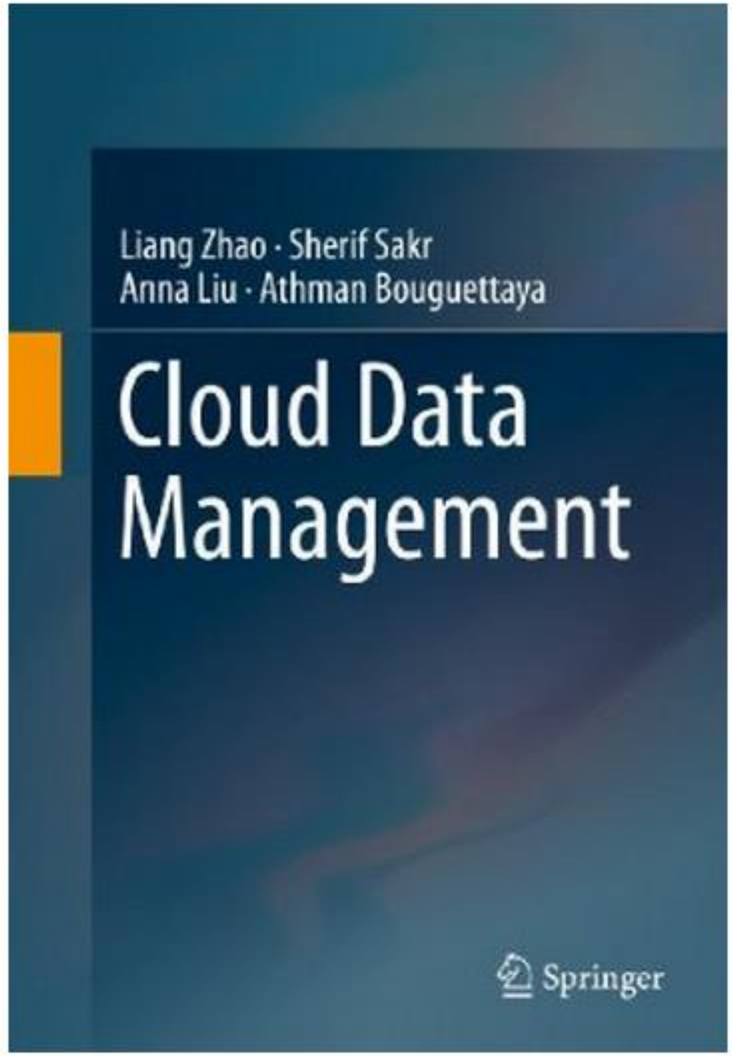
Recommended Literature





Recommended Literature: Cloud-DBs





Recommended Literature: Blogs

http://medium.baqend.com/

http://www.nosqlweekly.com/

http://www.dzone.com/mz/nosql

https://martin.kleppmann.com/

http://www.infoq.com/nosql/

http://highscalability.com/

http://muratbuffalo.blogspot.de/

http://db-engines.com/en/ranking

Seminal NoSQL Papers

- Lamport, Leslie. **Paxos made simple.**, SIGACT News, 2001
- S. Gilbert, et al., **Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services**, SIGACT News, 2002
- F. Chang, et al., **Bigtable: A Distributed Storage System For Structured Data**, OSDI, 2006
- G. DeCandia, et al., **Dynamo: Amazon's Highly Available Key-Value Store**, SOSP, 2007
- M. Stonebraker, el al., **The end of an architectural era: (it's time for a complete rewrite)**, VLDB, 2007
- B. Cooper, et al., **PNUTS: Yahoo!'s Hosted Data Serving Platform**, VLDB, 2008
- Werner Vogels, **Eventually Consistent**, ACM Queue, 2009
- B. Cooper, et al., **Benchmarking cloud serving systems with YCSB.**, SOCC, 2010
- A. Lakshman**, Cassandra - A Decentralized Structured Storage System**, SIGOPS, 2010
- J. Baker, et al., **MegaStore: Providing Scalable, Highly Available Storage For Interactive Services**, CIDR, 2011
- M. Shapiro, et al.: **Conflict-free replicated data types**, Springer, 2011
- J.C. Corbett, et al., **Spanner: Google's Globally-Distributed Database**, OSDI, 2012
- Eric Brewer, **CAP Twelve Years Later: How the "Rules" Have Changed**, IEEE Computer, 2012
- J. Shute, et al., **F1: A Distributed SQL Database That Scales**, VLDB, 2013
- L. Qiao, et al., **On Brewing Fresh Espresso: Linkedin's Distributed Data Serving Platform**, SIGMOD, 2013
- N. Bronson, et al., **Tao: Facebook's Distributed Data Store For The Social Graph**, USENIX ATC, 2013
- P. Bailis, et al., **Scalable Atomic Visibility with RAMP Transactions**, SIGMOD 2014

SCDM 2016

Home | CFP | Committee | Submission | Contact

4th Workshop on Scalable Cloud Data Management

Co-located with the IEEE BigData Conference.
Washington D.C., December 5th 2016.

Submit Paper

June 6, 2016

SCDM 2016 announced

The fourth Scalable Cloud Data Management Workshop (SCDM 2016) will again be held in conjunction with the IEEE BigData 2016 - this year in Washington D.C.

OCTOBER 29, 2015
SCDM 2015

OCTOBER 1, 2016
Paper Deadline

SCDM 2016 announced

Paper Deadline

Thank you

felix.gessert@baqend.com
baqend.com