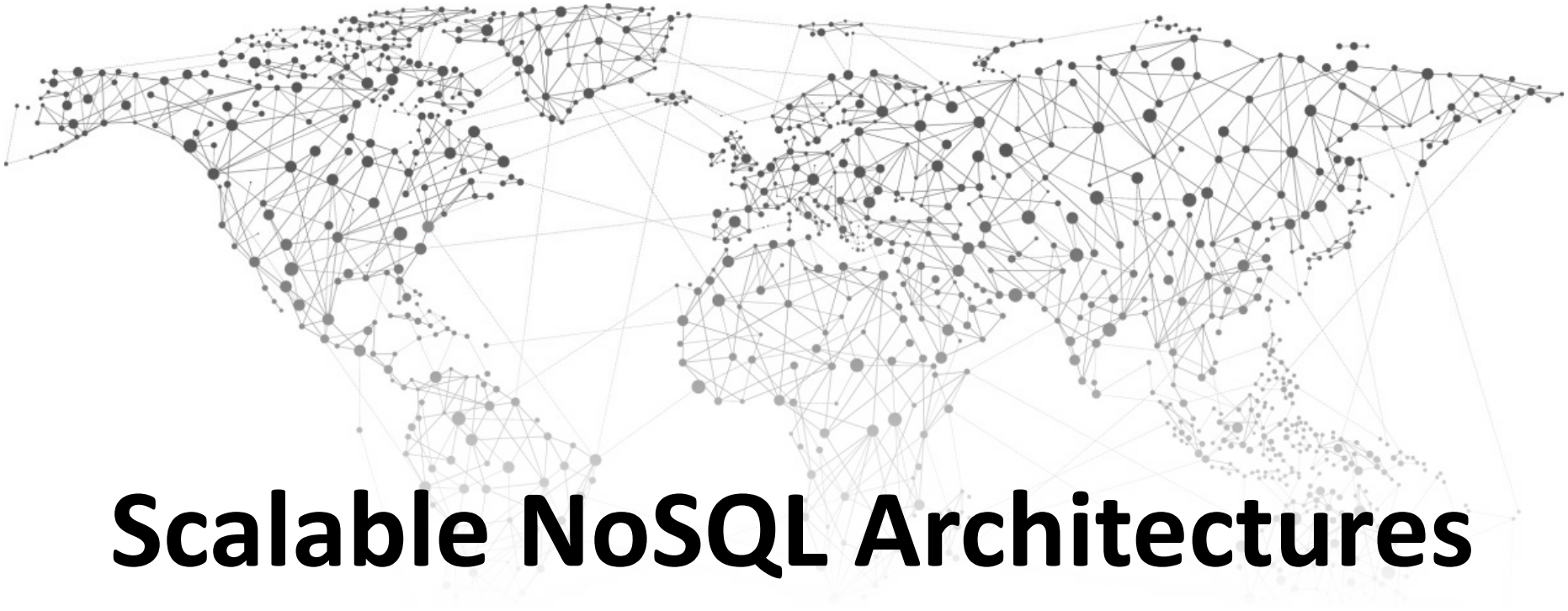


Scalable NoSQL Architectures

Lessons Learned

Felix Gessert
felix.gessert@baqend.com
May 18, AWS Summit 2017, Berlin

BaQend

@baqendcom

About me

Felix Gessert

PhD Thesis & Research

CEO & Co-founder

NoSQL Technologies:

- Polyglot Persistence
- Web Caching
- Cloud Data Management

Universität Hamburg

Baqend:
High-Performance
Backend-as-a-Service

BaQend

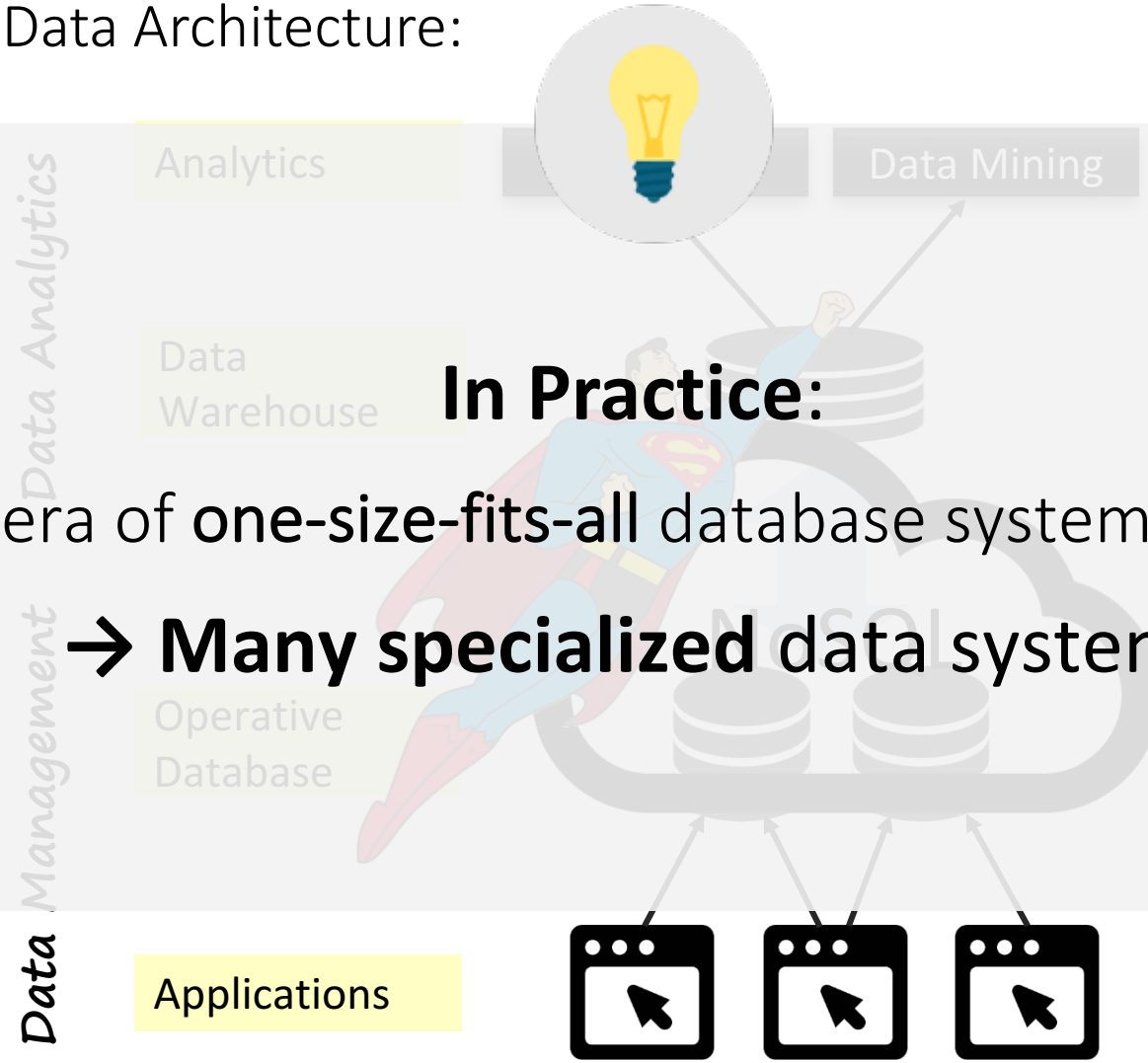
www.baqend.com

INTRODUCTION

What are NoSQL data stores?

Architecture

Typical Data Architecture:



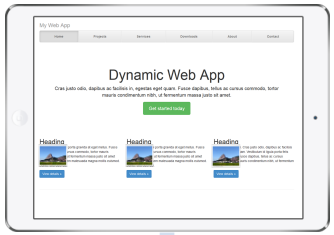
In Practice:

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

→ **Many specialized** data systems

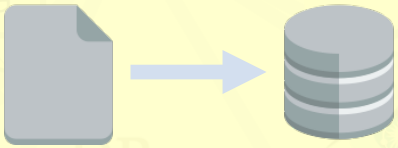
How to choose a database system?

Many Potential Candidates



Question for this talk:

How to approach the



requirements database

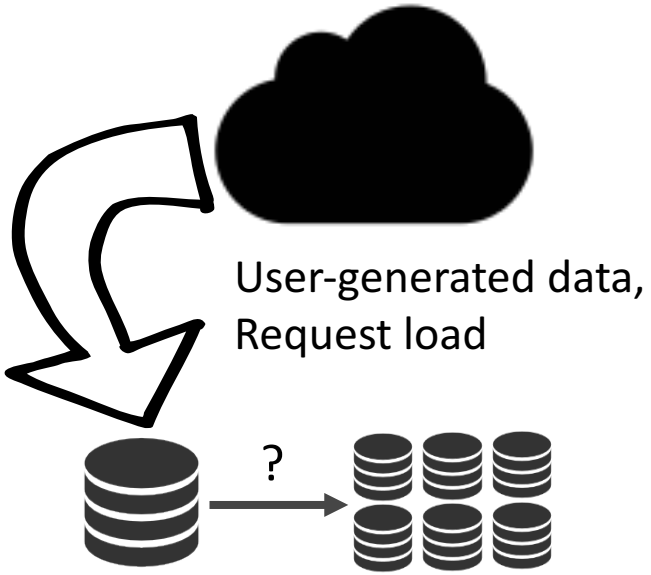
decision problem?

NoSQL Databases

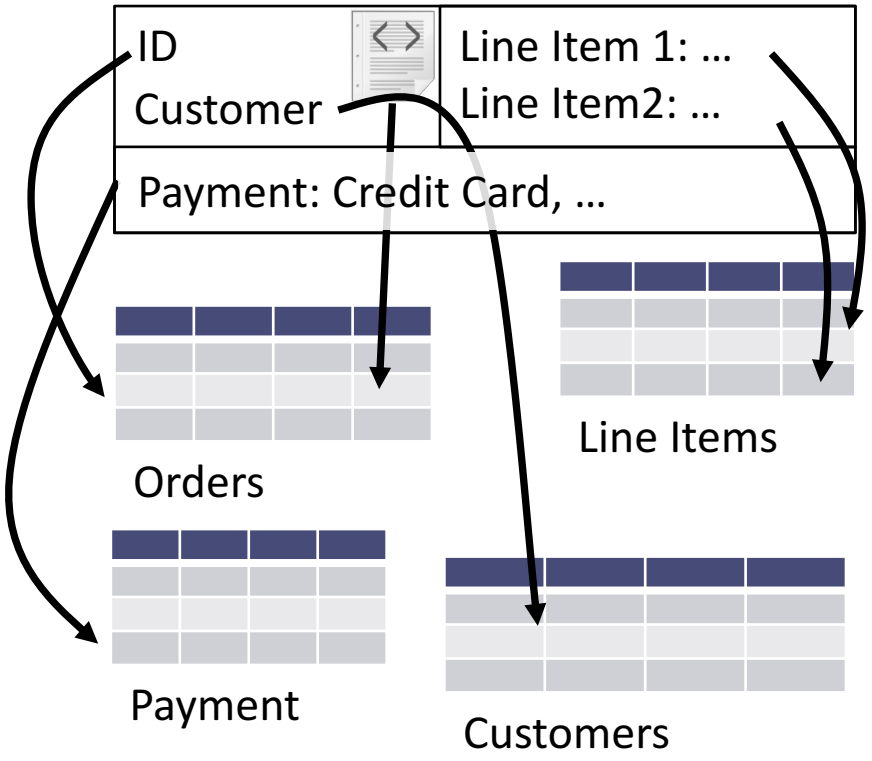
Motivation

- Two main drivers:

Scalability

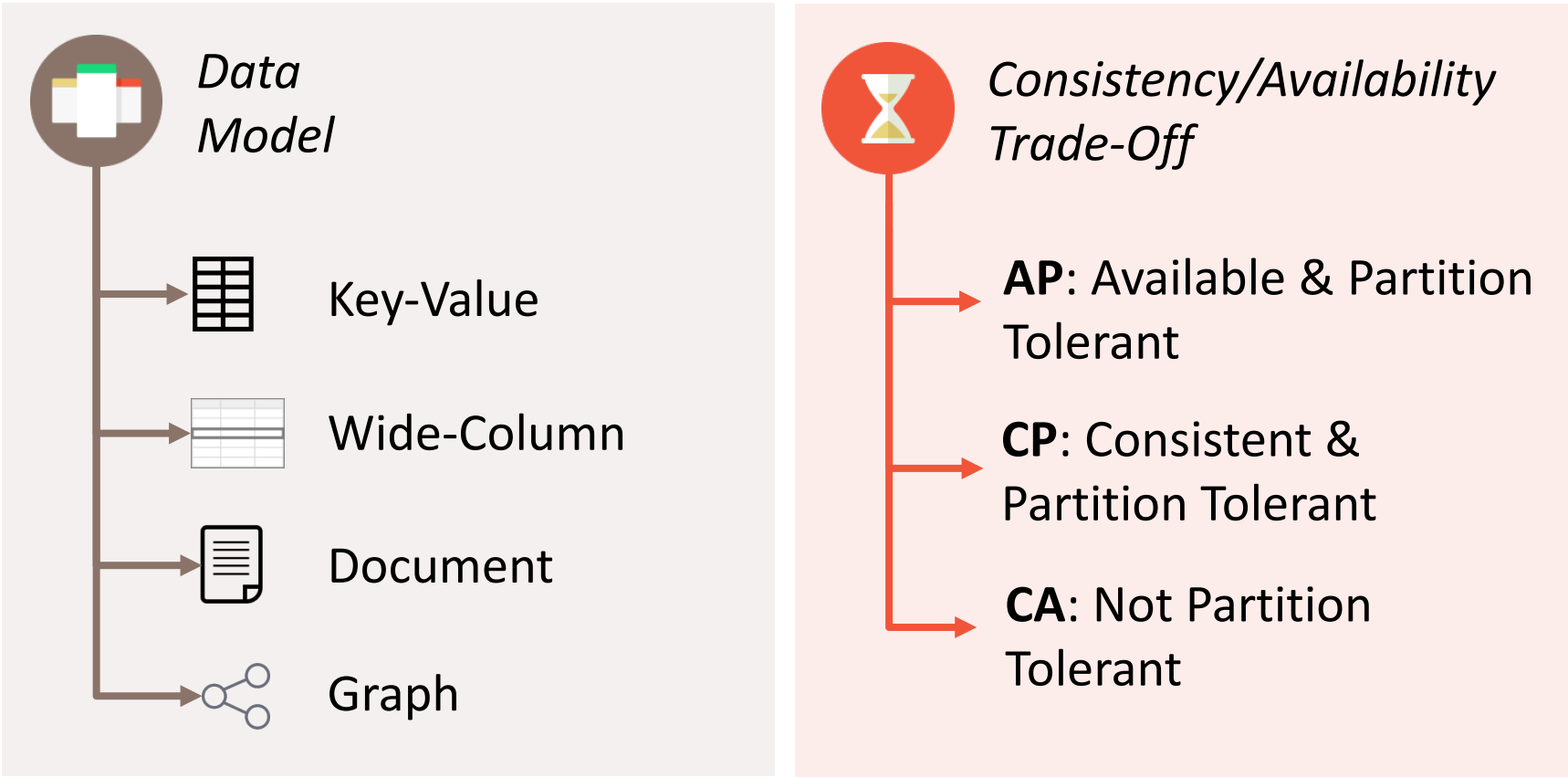


Impedance Mismatch

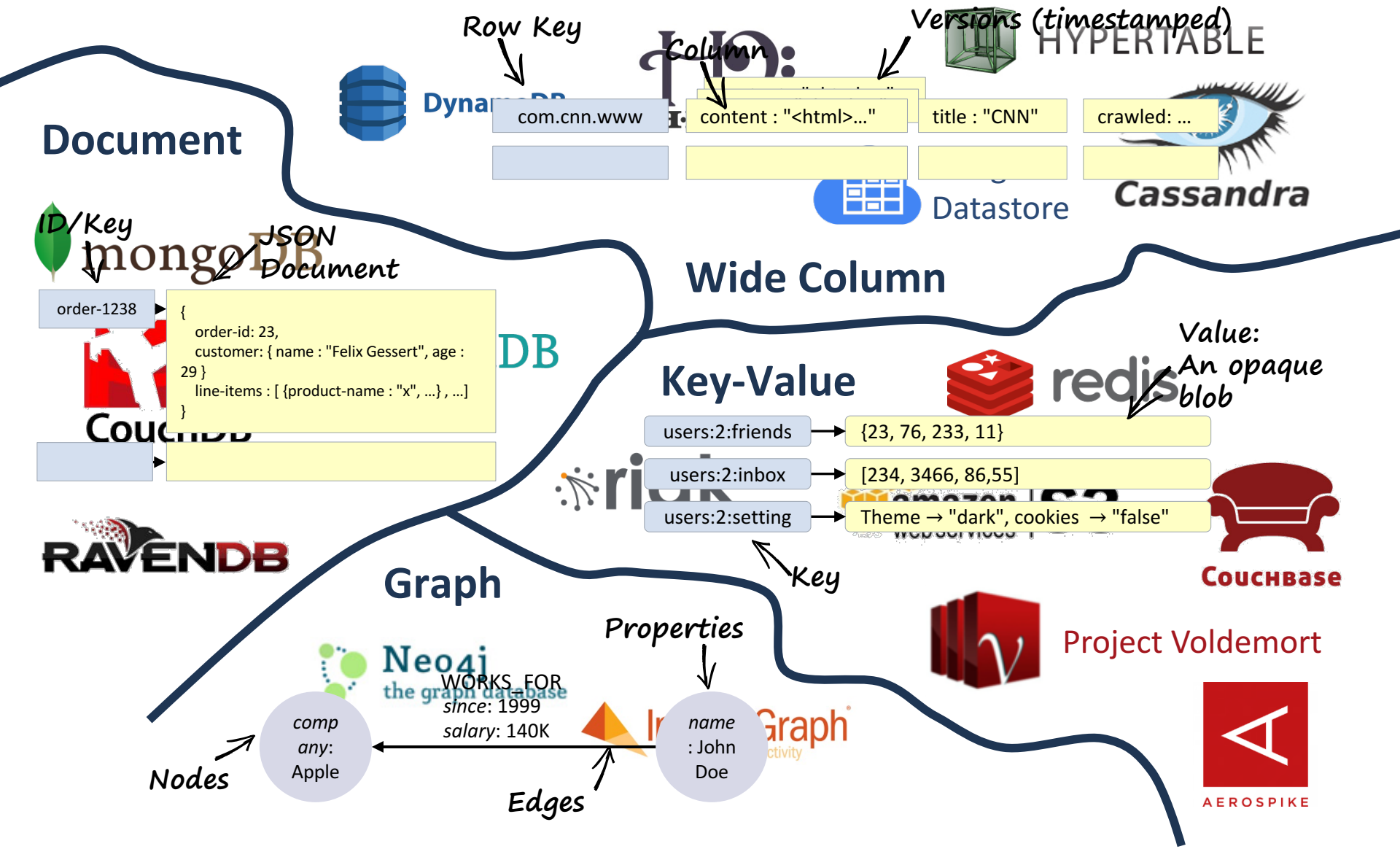


NoSQL System Classification

- Two common criteria:

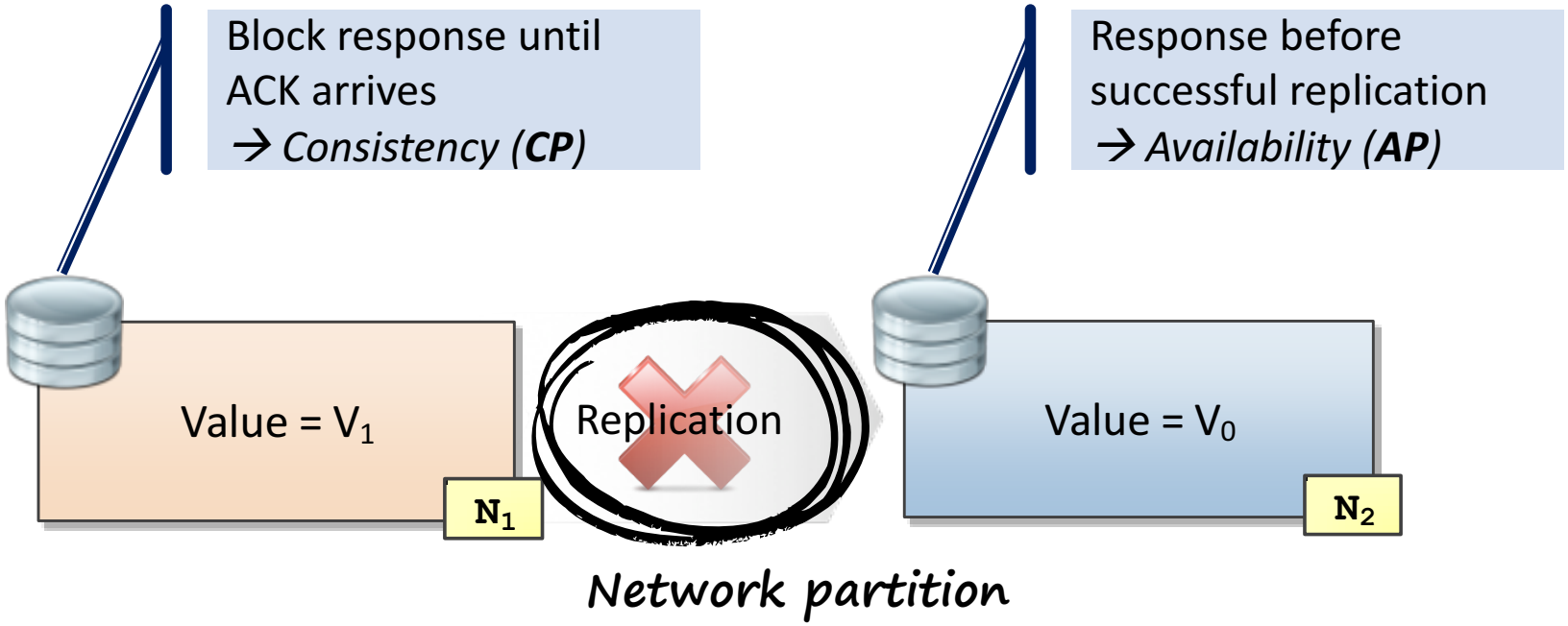


Data Model Classes



CAP-Theorem Classes

Consistency, **Availability**, **Partition Tolerance** - pick two:



Eric Brewer, ACM-PODC Keynote, July 2000

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

BEYOND CAP AND DATA MODELS

What about concrete requirements, e.g. query capabilites?

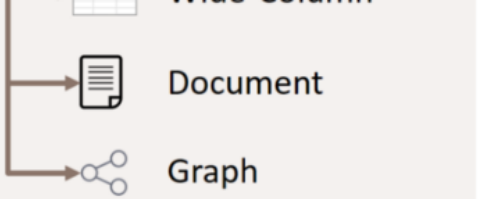
enable

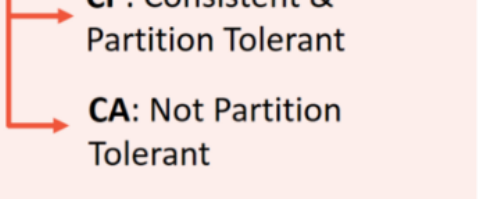
enable

Functional Require-ments from the application

Central techniques NoSQL databases employ

Operational Require-ments





2.1 Different Data Models

The most commonly employed distinction between NoSQL databases is the way they store and allow access to data. Each system covered in this paper can be categorised as either key-value store, document store or wide-column store.

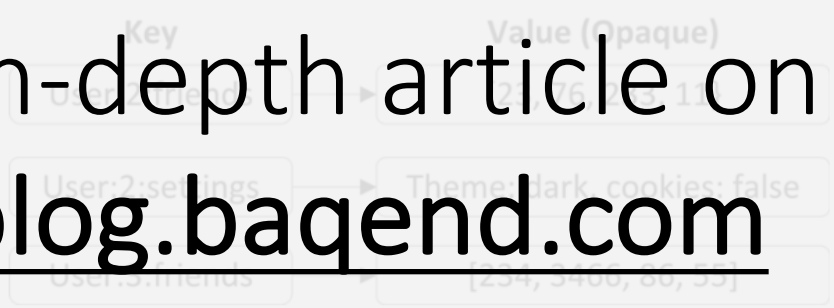
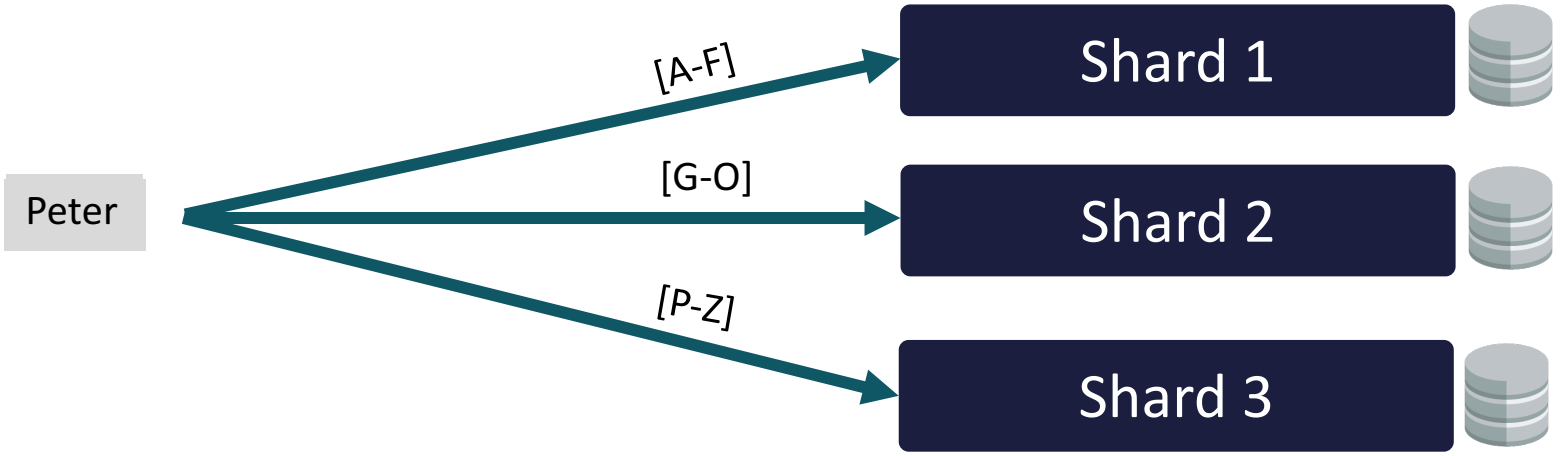


Figure 1: Key-value stores offer efficient storage and retrieval of arbitrary values.

2.1.1 Key-Value Stores. A key-value store consists of a set of key-value pairs with unique keys. Due to this simple structure, it only supports get and put operations. As the nature of the stored value is transparent to the database, pure key-value stores do not support operations beyond simple CRUD (Create, Read, Update, Delete). Key-value stores are therefore often referred to as **schemaless**: Any assumptions about the structure of stored data are implicitly encoded in the application logic (*schema-on-read*) and not explicitly defined through a data definition language (*schema-on-write*).

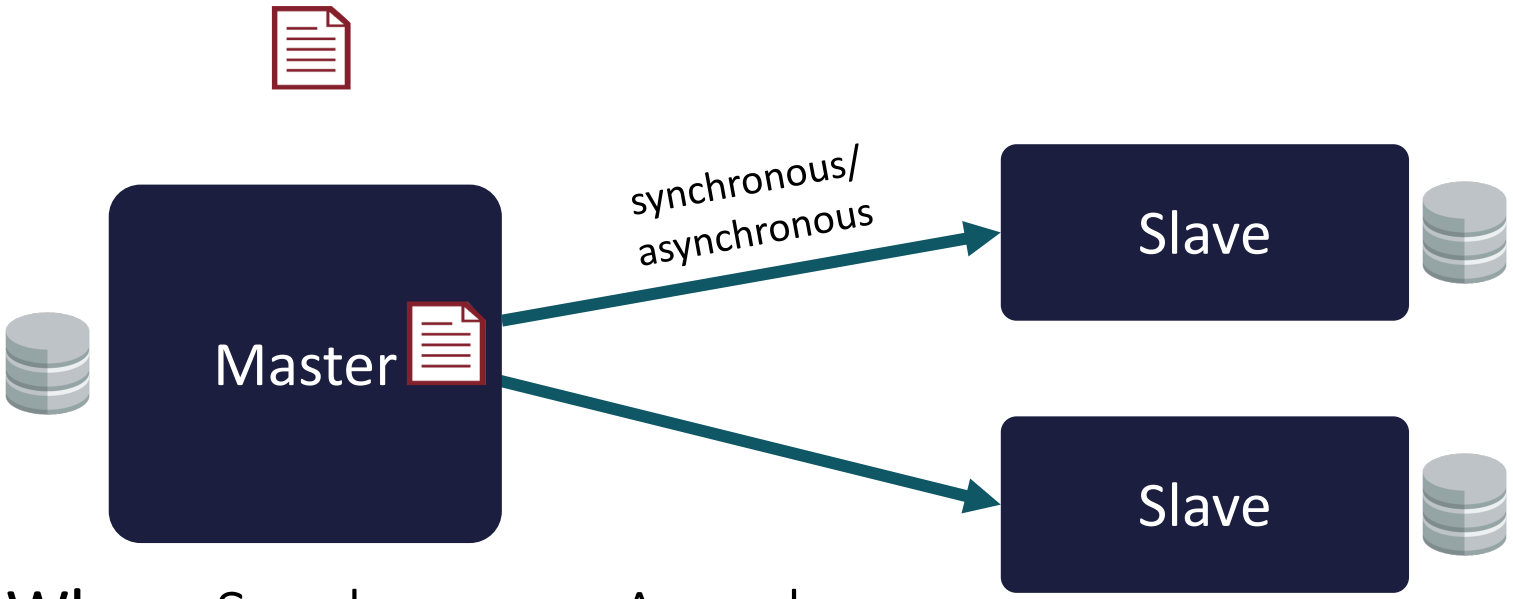
The obvious advantages of this data model lie in its simplicity. The very simple

- Scaling **Storage** and **Throughput**



- **Partitioning strategies**:
  Hash-based, Range-based, Entitiy Group Sharding

- Read Scalability and Fault Tolerance



- **When**: Synchronous, Asynchronous
- **Where**: Multi-Master, Master-Slave, Quorum-based

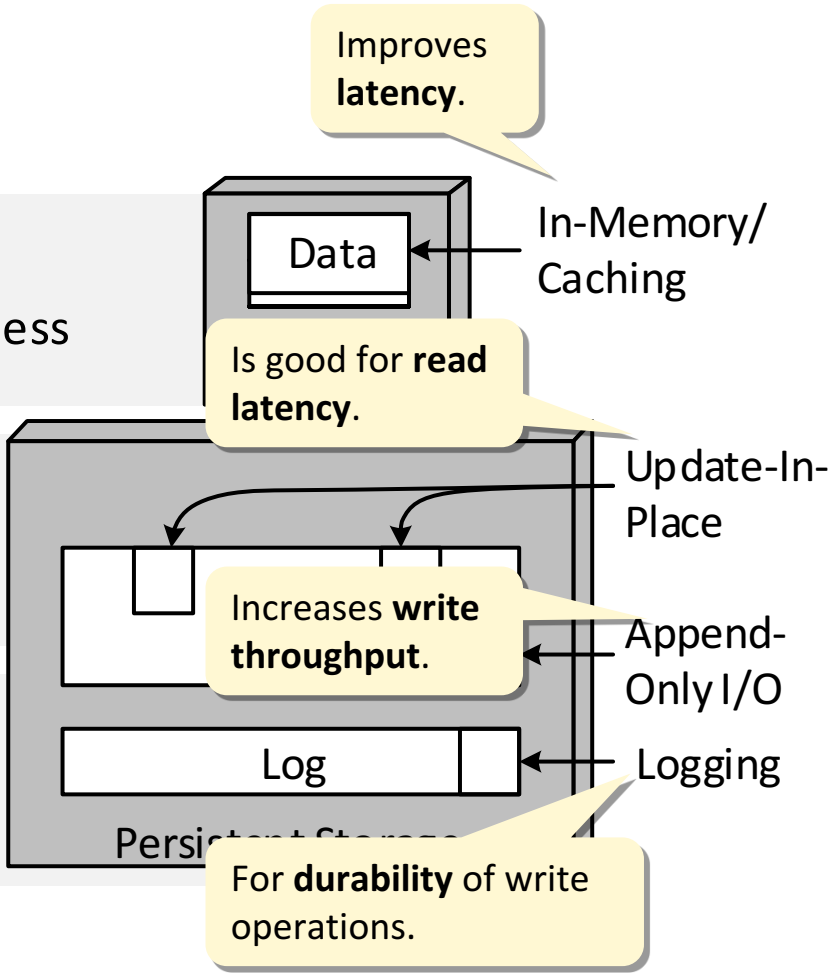
Özsu, M.T., Valduriez, P.: Principles of distributed database systems. Springer Science & Business Media (2011)

Storage Management:

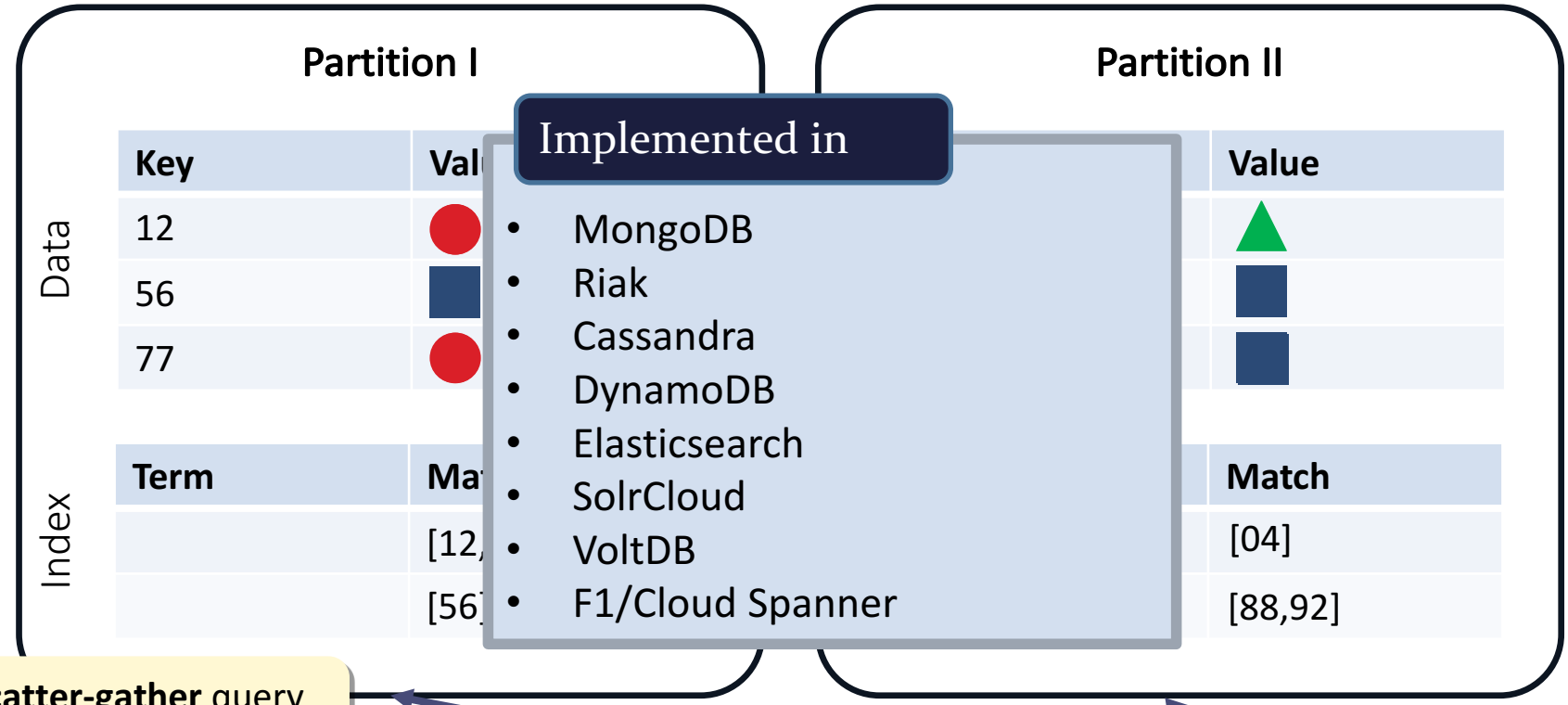
RAM
- Fast but small
- Random and sequential access

SSD
- Limited write cycles
- Bad for random writes

HDD
- Large capacities
- Only fast sequentially

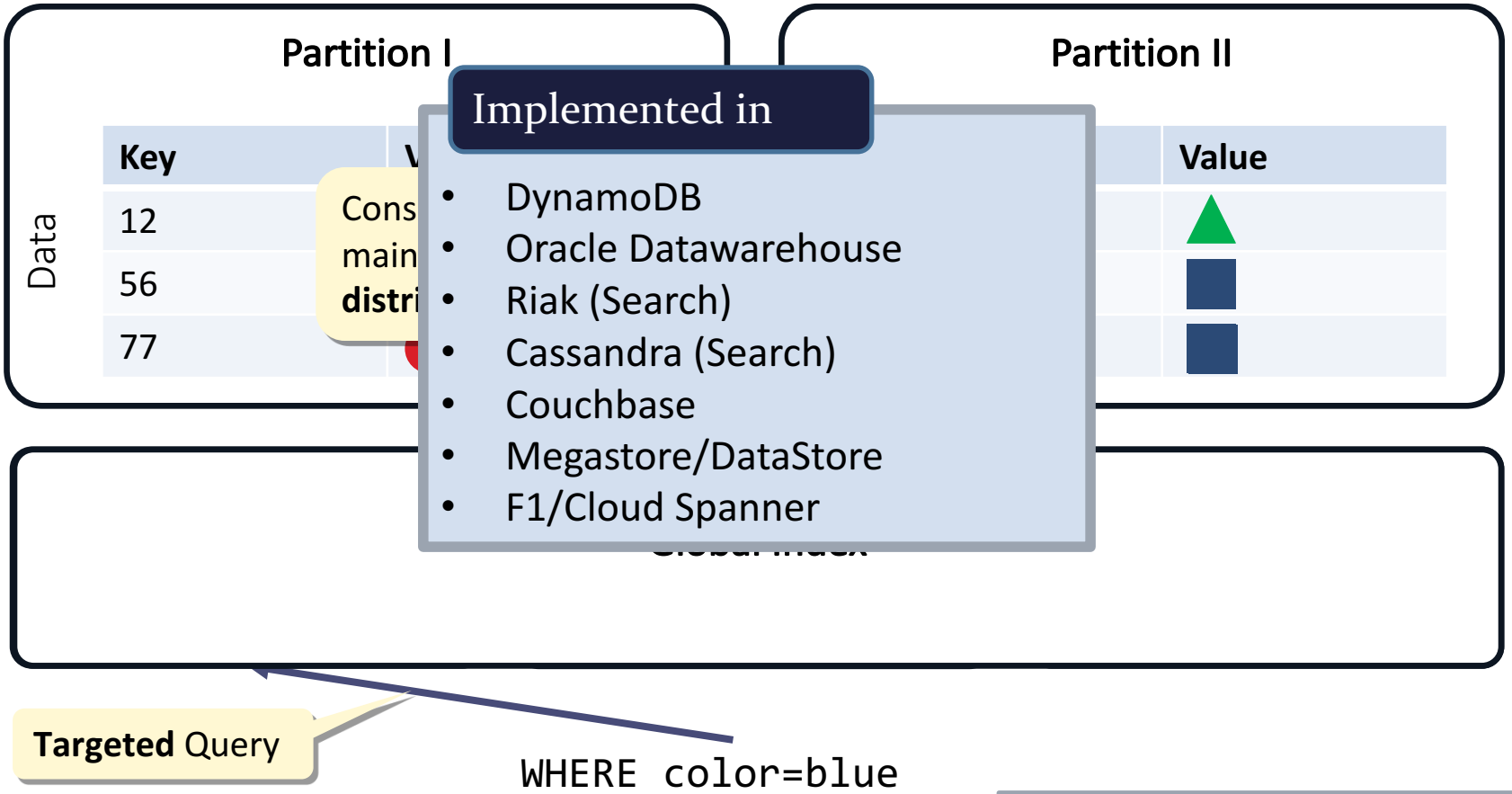


Local Indexing – Partition by document:



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

Global Indexing – Partition by term:



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

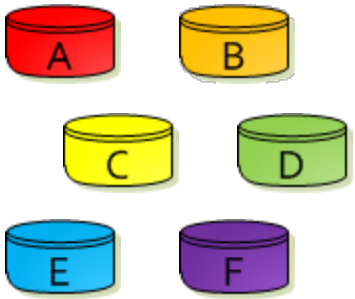
DECISION TREE

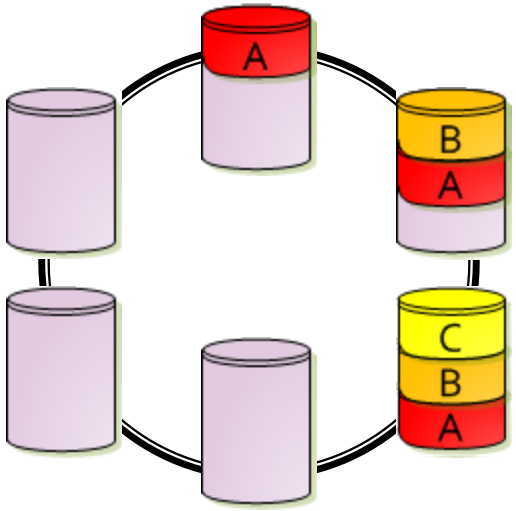
How are the techniques from the NoSQL toolbox used in **data stores**?

Dynamo (AP)

Distributed, Highly Available KV-Store

- Developed at **Amazon** (2007)
- **Techniques**: Consistent Hashing & Quorums
- Inspired **Riak**, **Cassandra**, **Voldemort**, **DynamoDB**





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

Classification: Dynamo-style Systems

Techniques

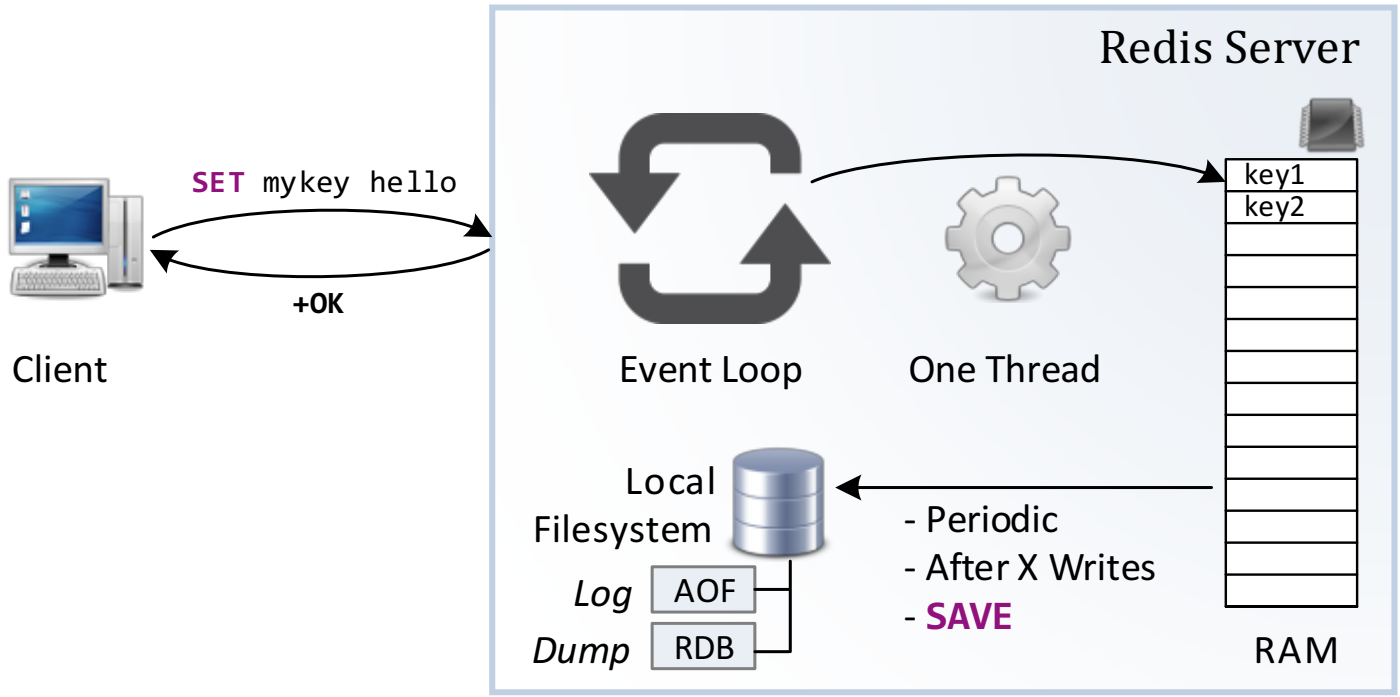
 **Sharding**	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
 **Replication**	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
 **Storage Management**	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
 **Query Processing**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

Redis (CA)

In-Memory Key-Value Store

- **Re**mote **Di**ctionary **S**erver
- **Data model**: rich data structures (list, sets, ...)
- Hash-Sharding & Asynchronous Replication

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



Classification: Redis

Techniques

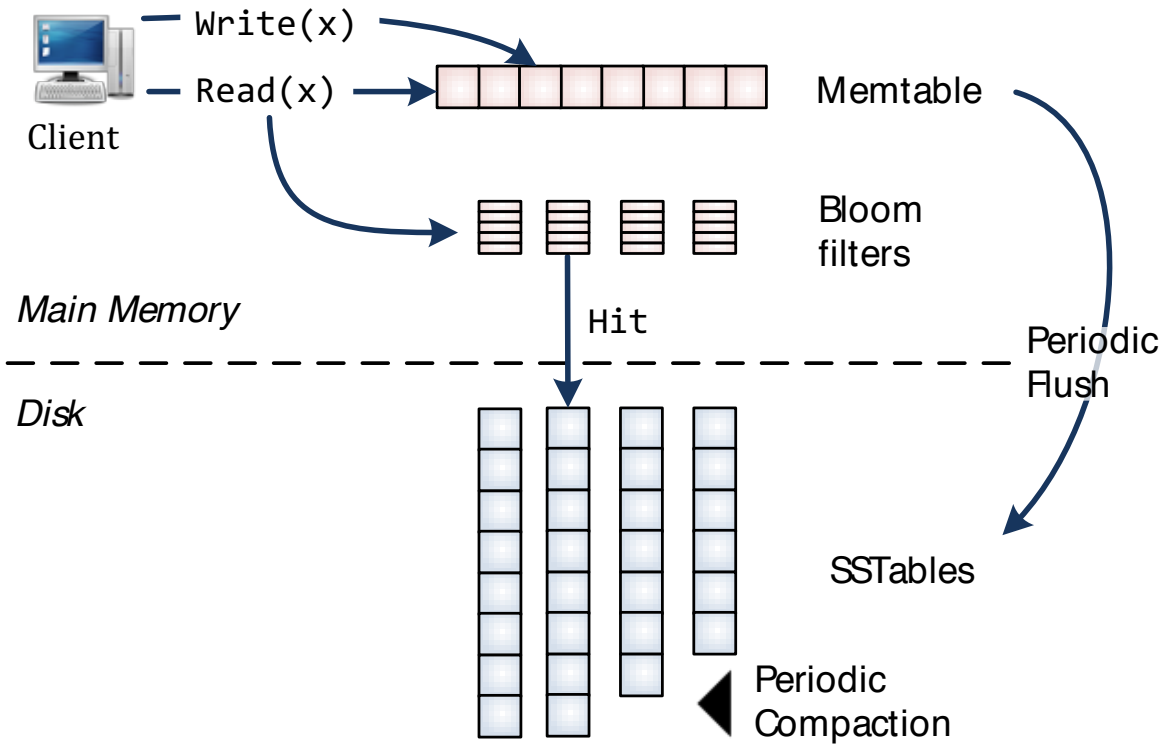
 **Sharding**	Range-Sharding	**Hash-Sharding**	Entity-Group Sharding	Consistent Hashing	Shared Disk
 **Replication**	Trans-action Protocol	Sync. Replica-tion	**Async. Replica-tion**	**Primary Copy**	Update Anywhere
 **Storage Management**	**Logging**	Update-in-Place	**Caching**	**In-Memory**	Append-Only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

Apache HBase (CP)

Consistent Wide-Column Store

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

- Open-Source Implementation of **BigTable**
- Range-Sharded & Synchronous Replication
- Optimized for **Write Throughput**



Classification: HBase

Techniques

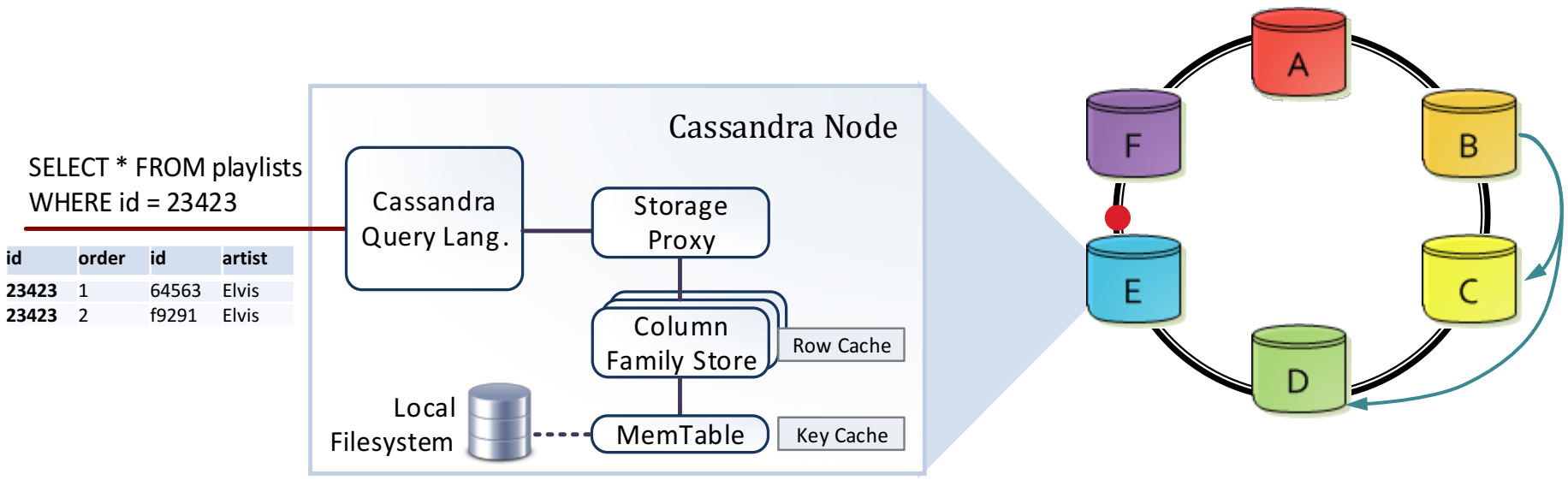
 **Sharding**	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
 **Replication**	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
 **Storage Management**	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
 **Query Processing**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

Apache Cassandra (AP)

Available Wide-Column Store

- Published 2007 by Facebook
- **BigTable data model** with **Dynamo sharding**
- SQL-like (CQL), but **limited complexity**

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



Classification: Cassandra

Techniques

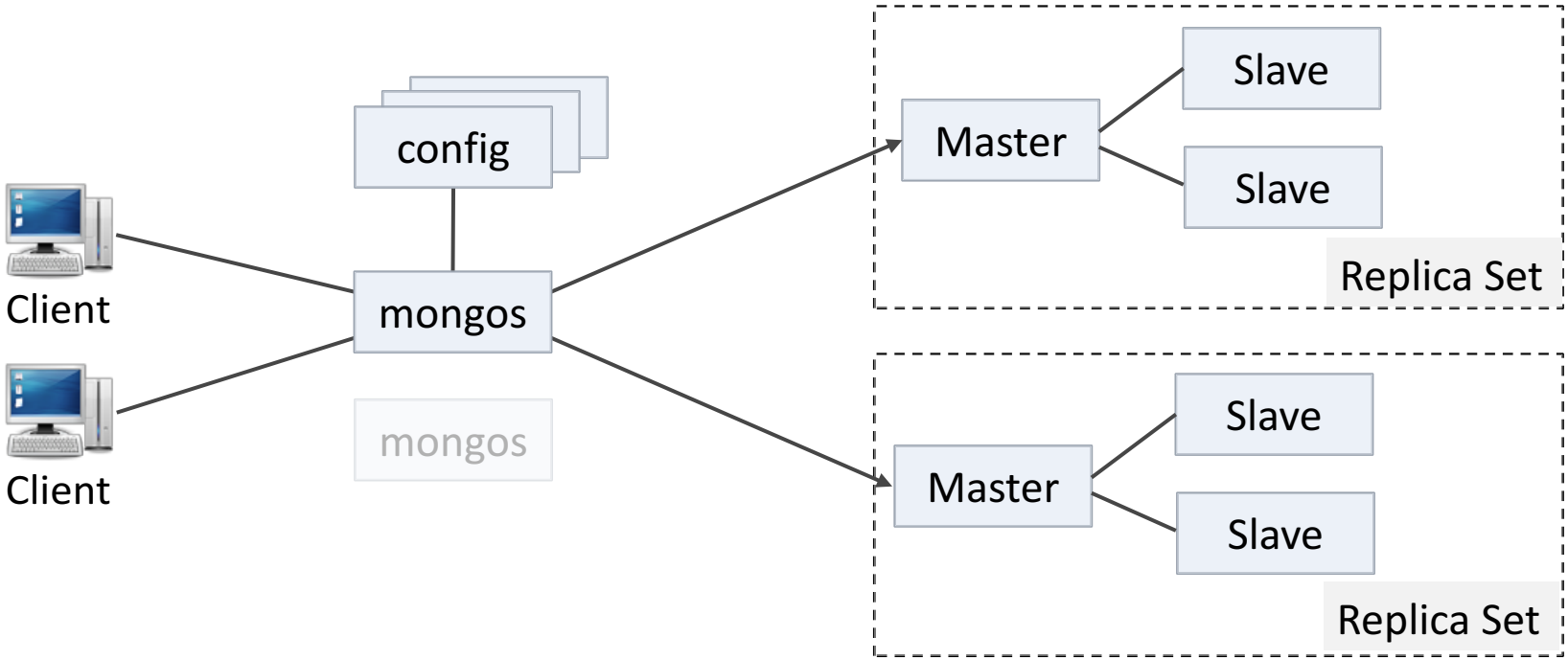
 **Sharding**	Range-Sharding	Hash-Sharding	Entity-Group Sharding	Consistent Hashing	Shared Disk
 **Replication**	Trans-action Protocol	Sync. Replica-tion	Async. Replica-tion	Primary Copy	Update Anywhere
 **Storage Management**	Logging	Update-in-Place	Caching	In-Memory	Append-Only Storage
 **Query Processing**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

MongoDB (CP)

Document Store

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

- Allows **complex queries** and indexing
- Sharding (range- or hash-based)
- Replication (synchronous or asynchronous)
- **Pluggable** Storage Engine



Classification: MongoDB

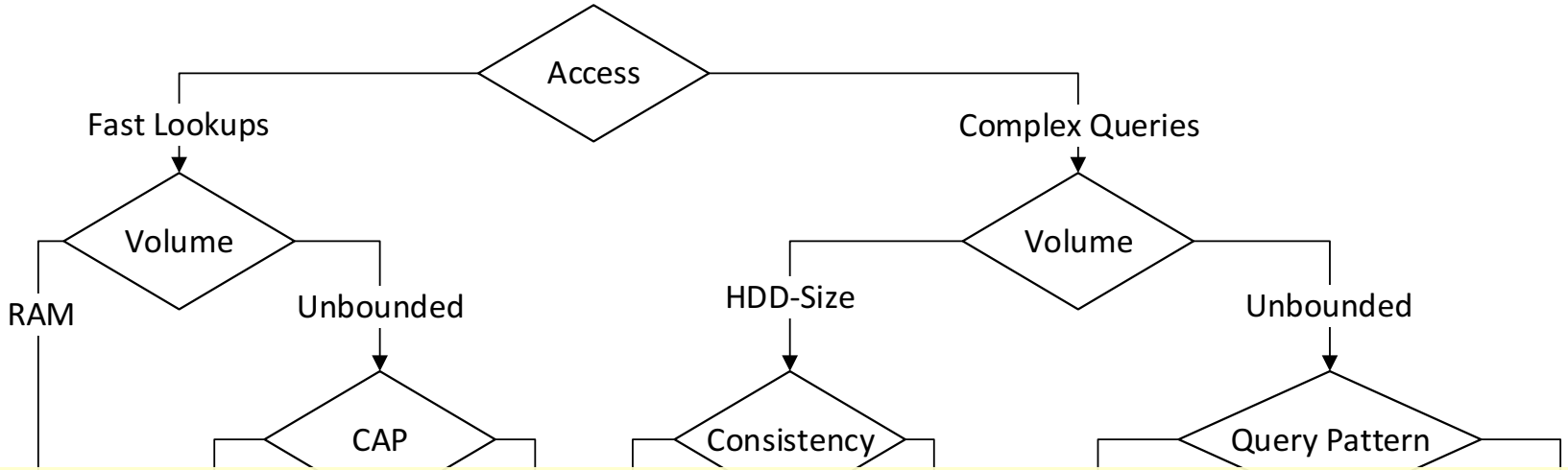
Techniques

 **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

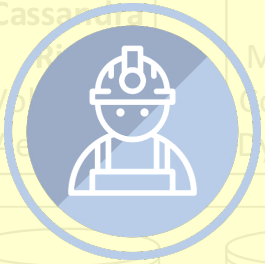
DECISION TREE

How can the choices for an appro-priate system be narrowed down?

NoSQL Decision Tree

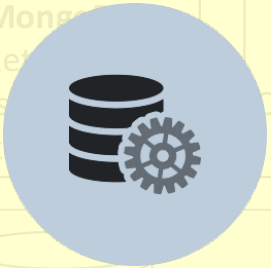


Purpose



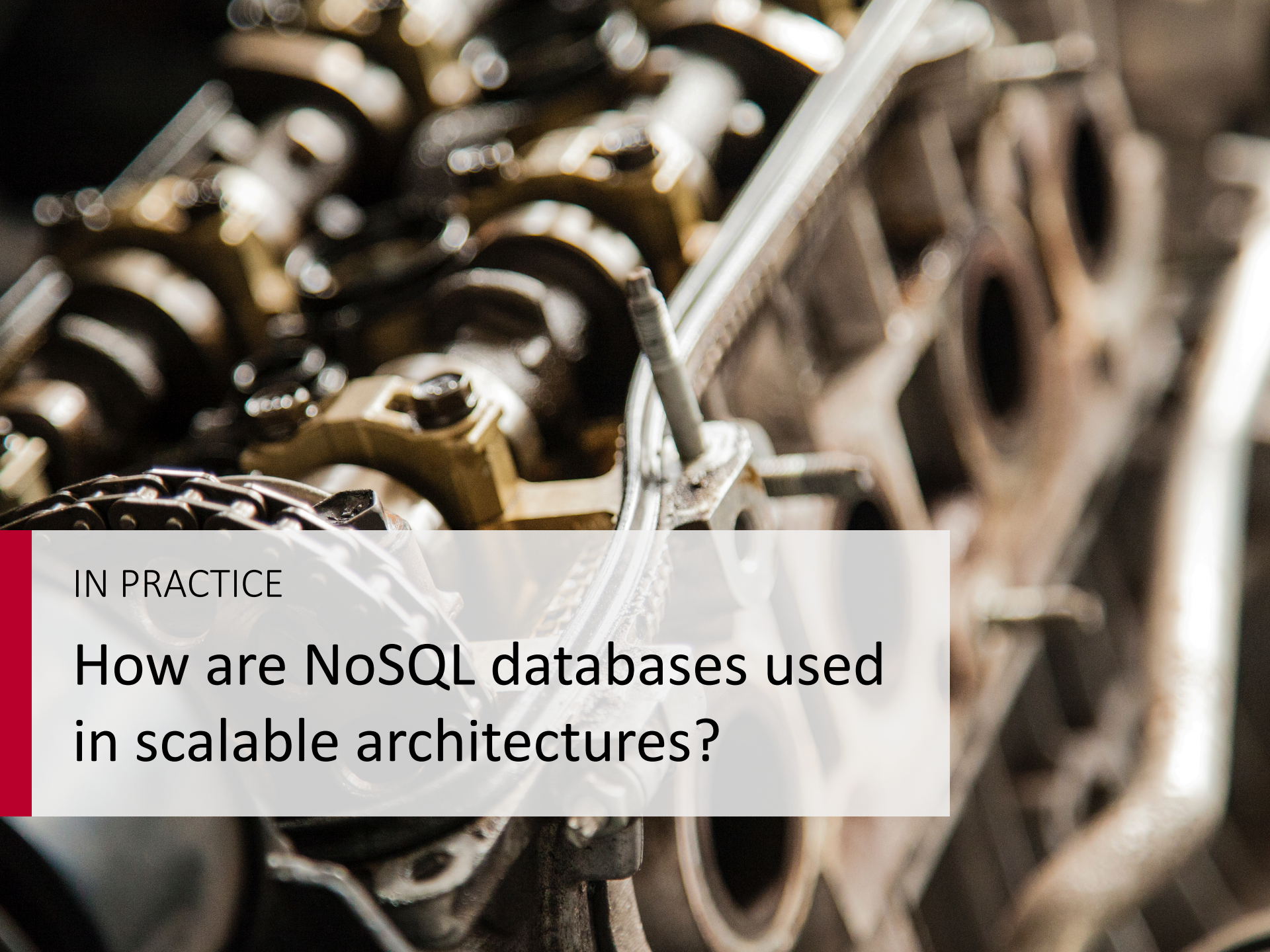
Architects

Find system candidates



DB vendors

Communicate Trade-Offs

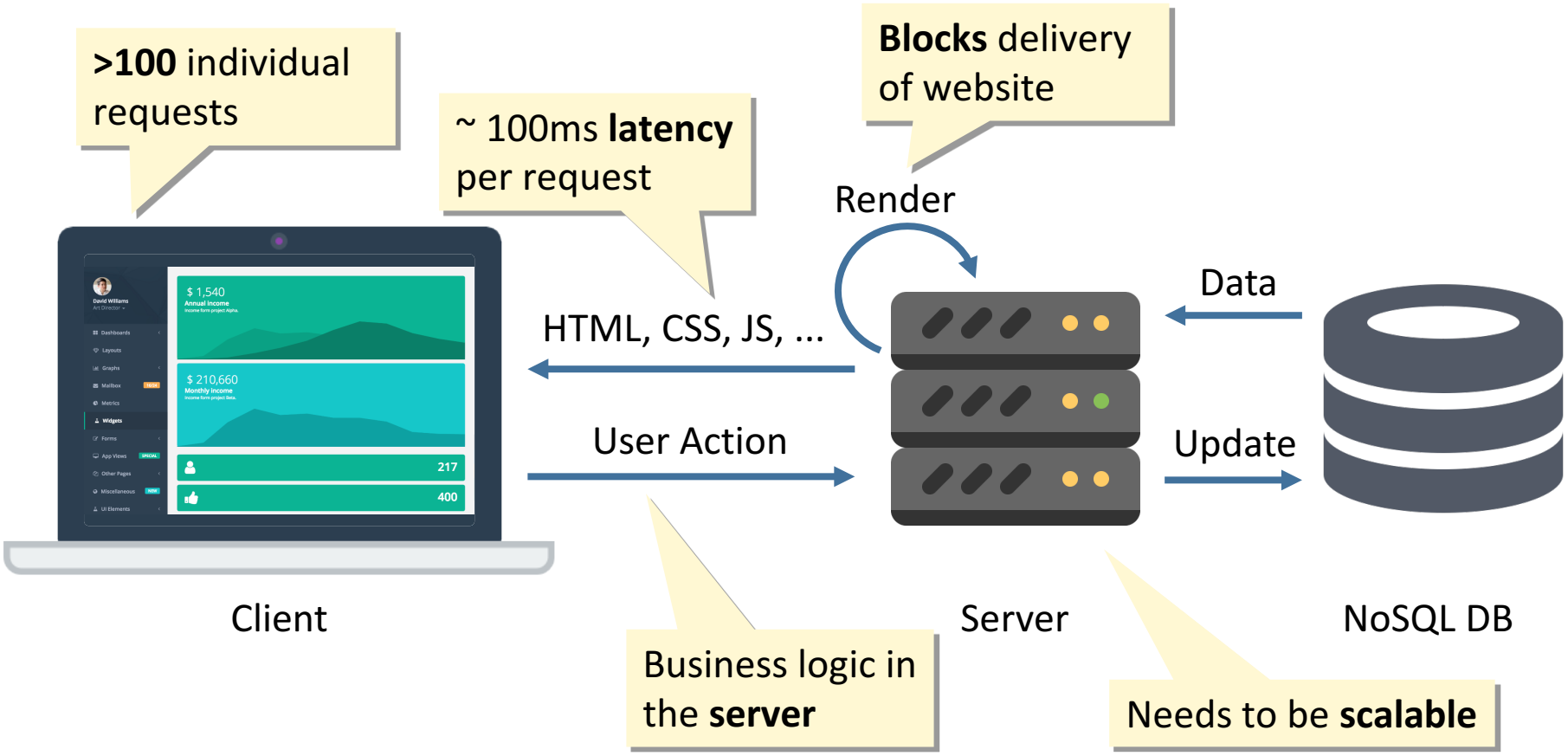


IN PRACTICE

How are NoSQL databases used in scalable architectures?

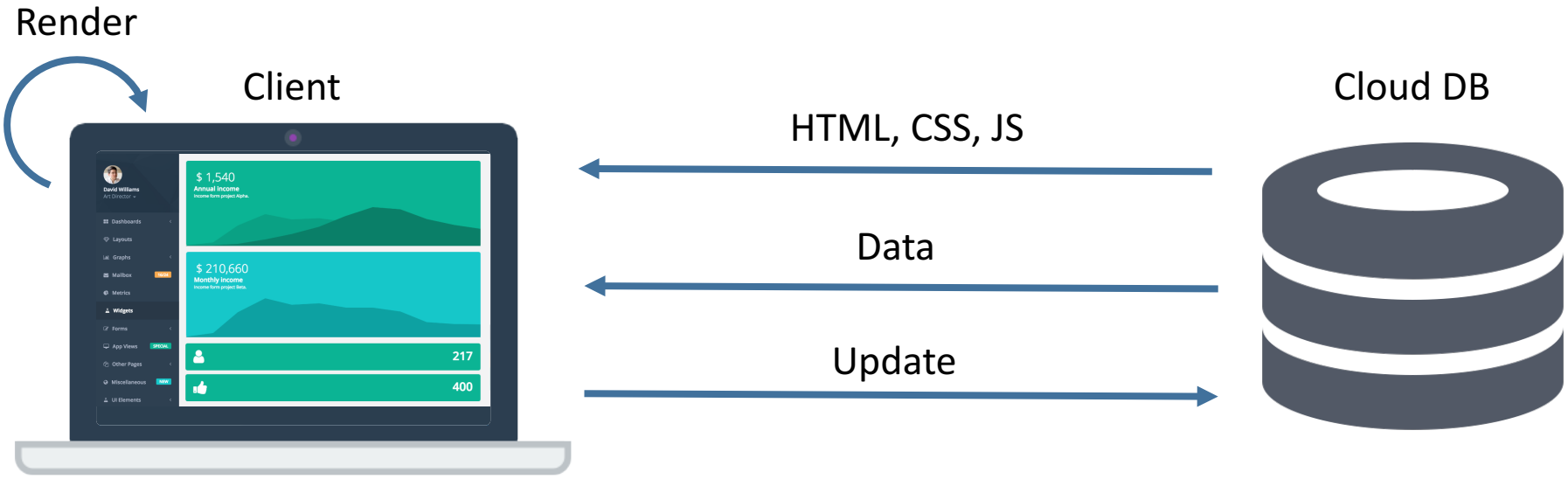
Scalable NoSQL Architectures

The 3-Tier Stack



Scalable NoSQL Architectures

The 2-Tier Stack



Challenges

- User Management
- Access Control
- Server-side Business Logic
- **Latency** and **scalability** still an issue

The Serverless Paradigm

FaaS, BaaS and everything in between

Backend-as-a-Service

Function-as-a-Service
(AWS Lambda)

Authentication

Gateway

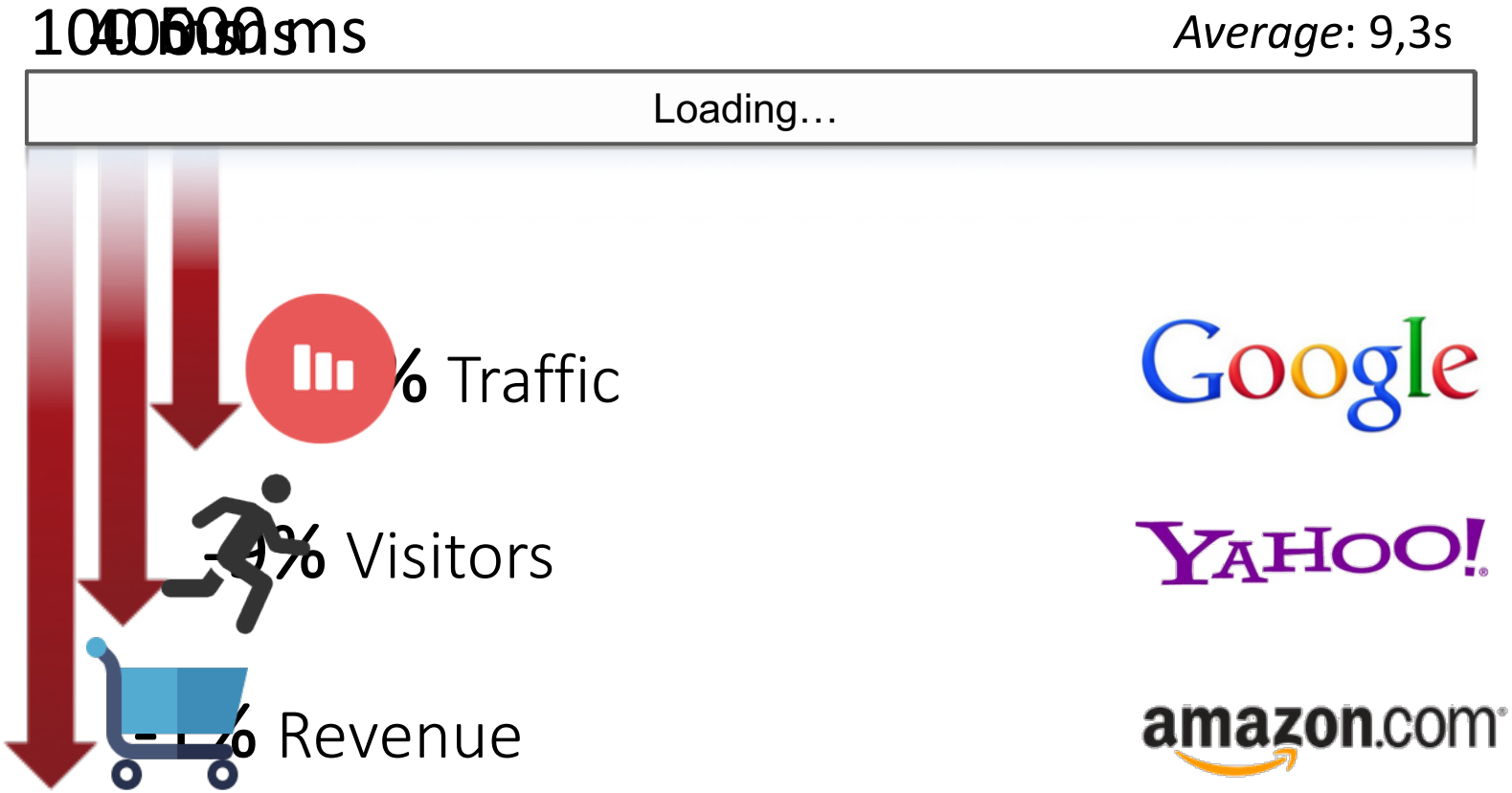
Business Code

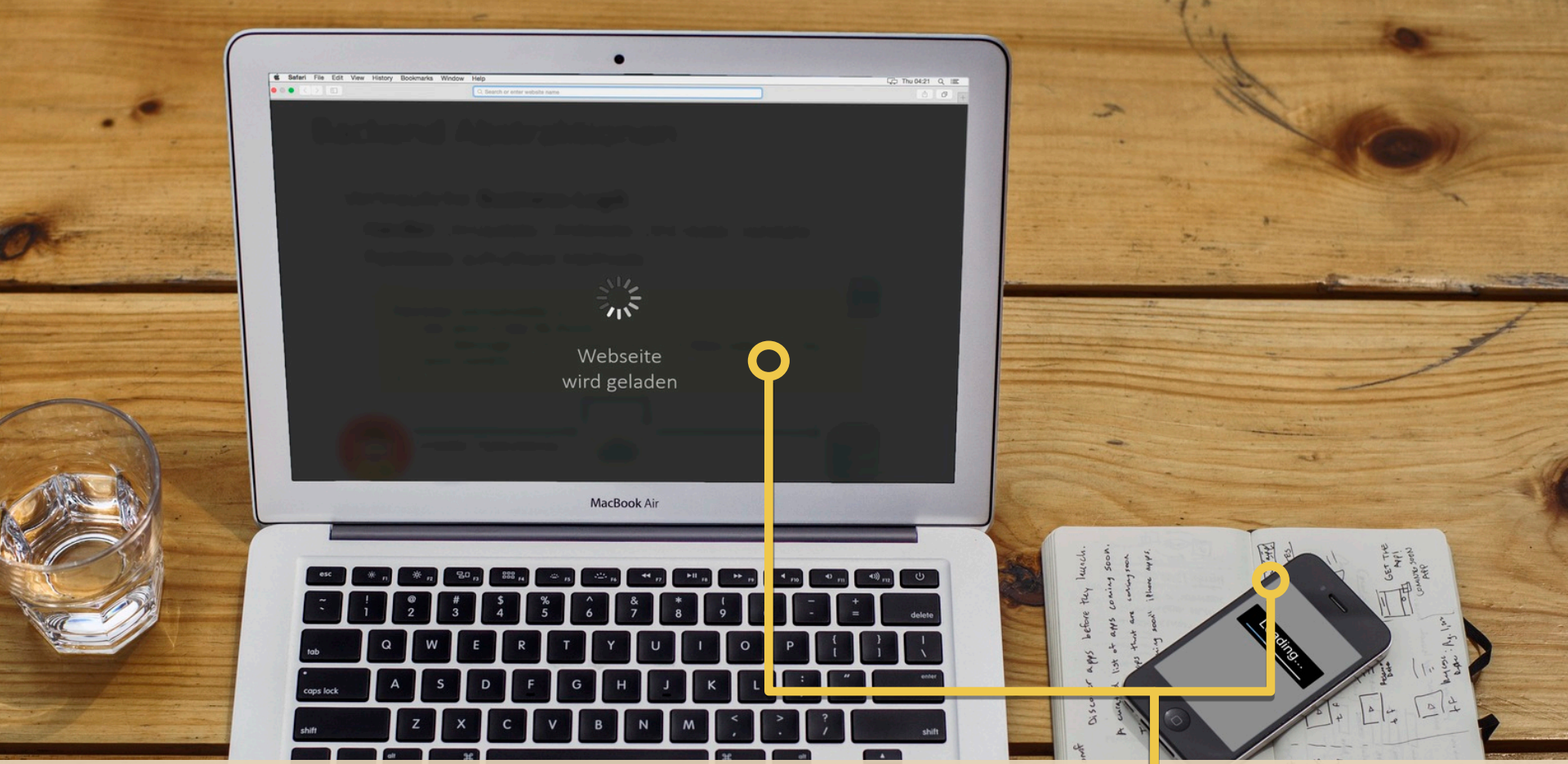
Another DB

Cloud DB

Presentation
is loading

Why latency matters



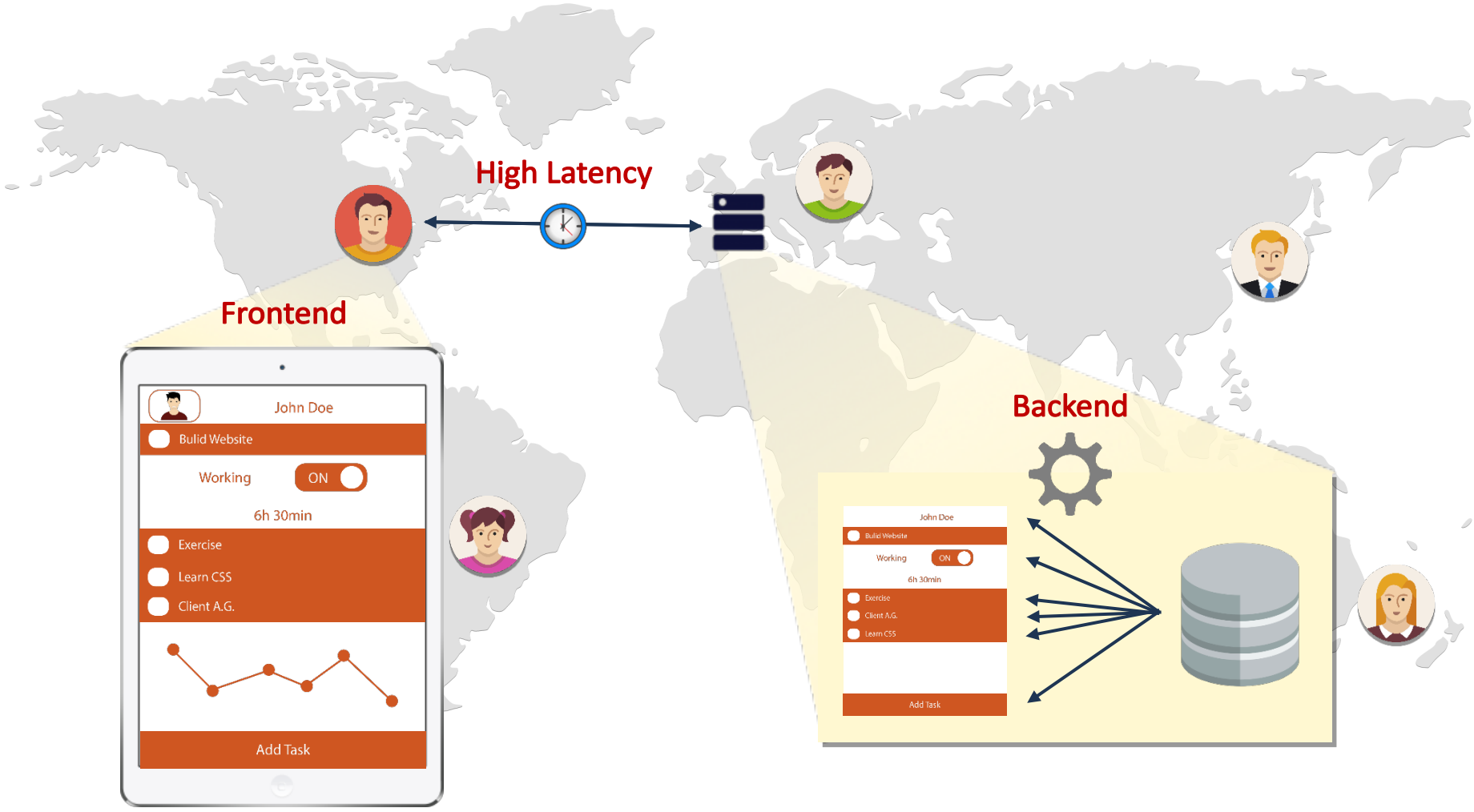


If perceived speed is such an important factor

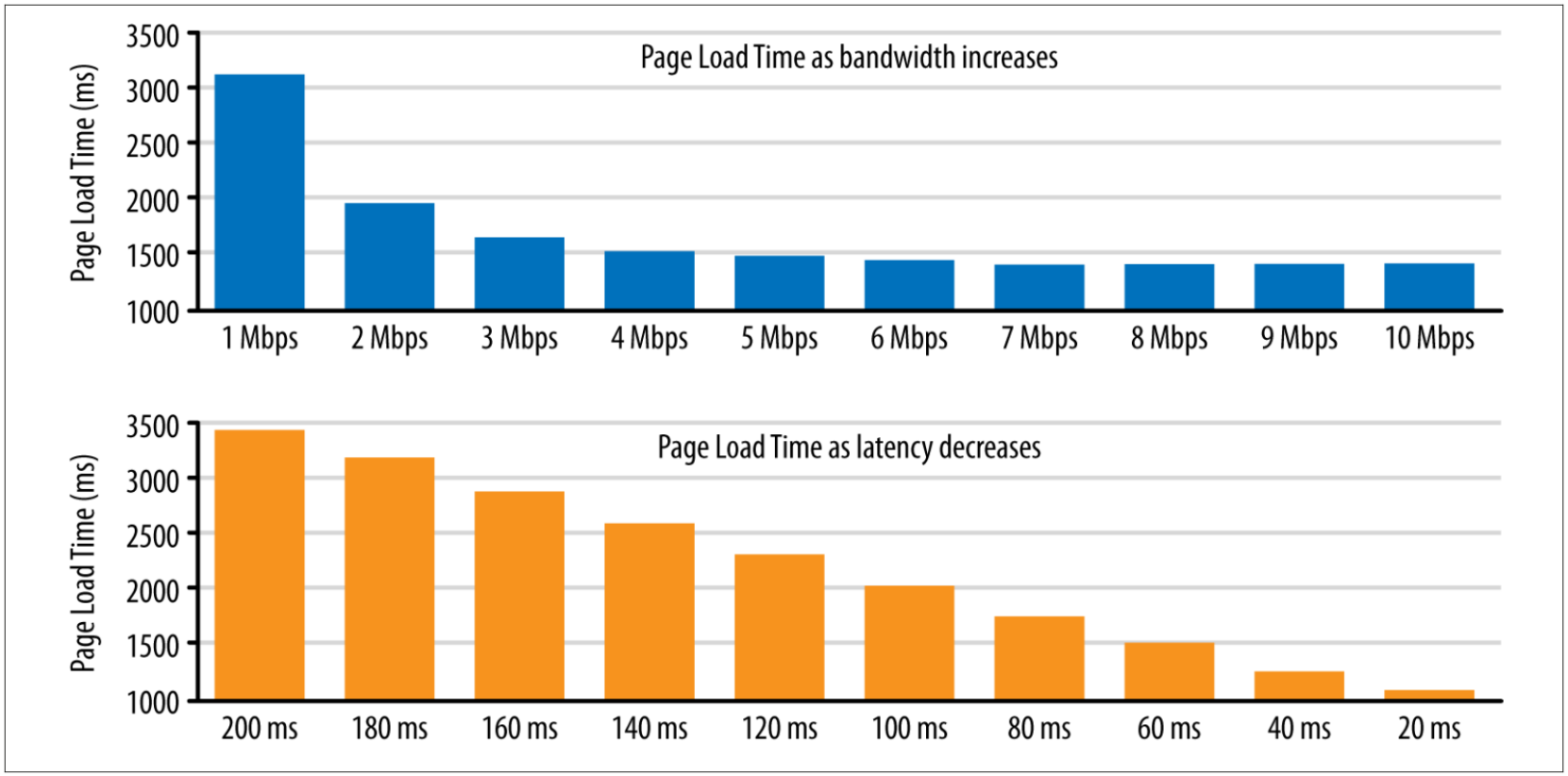
...what causes slow page load times?

The Problem

Three Bottlenecks: Latency, Backend & Frontend

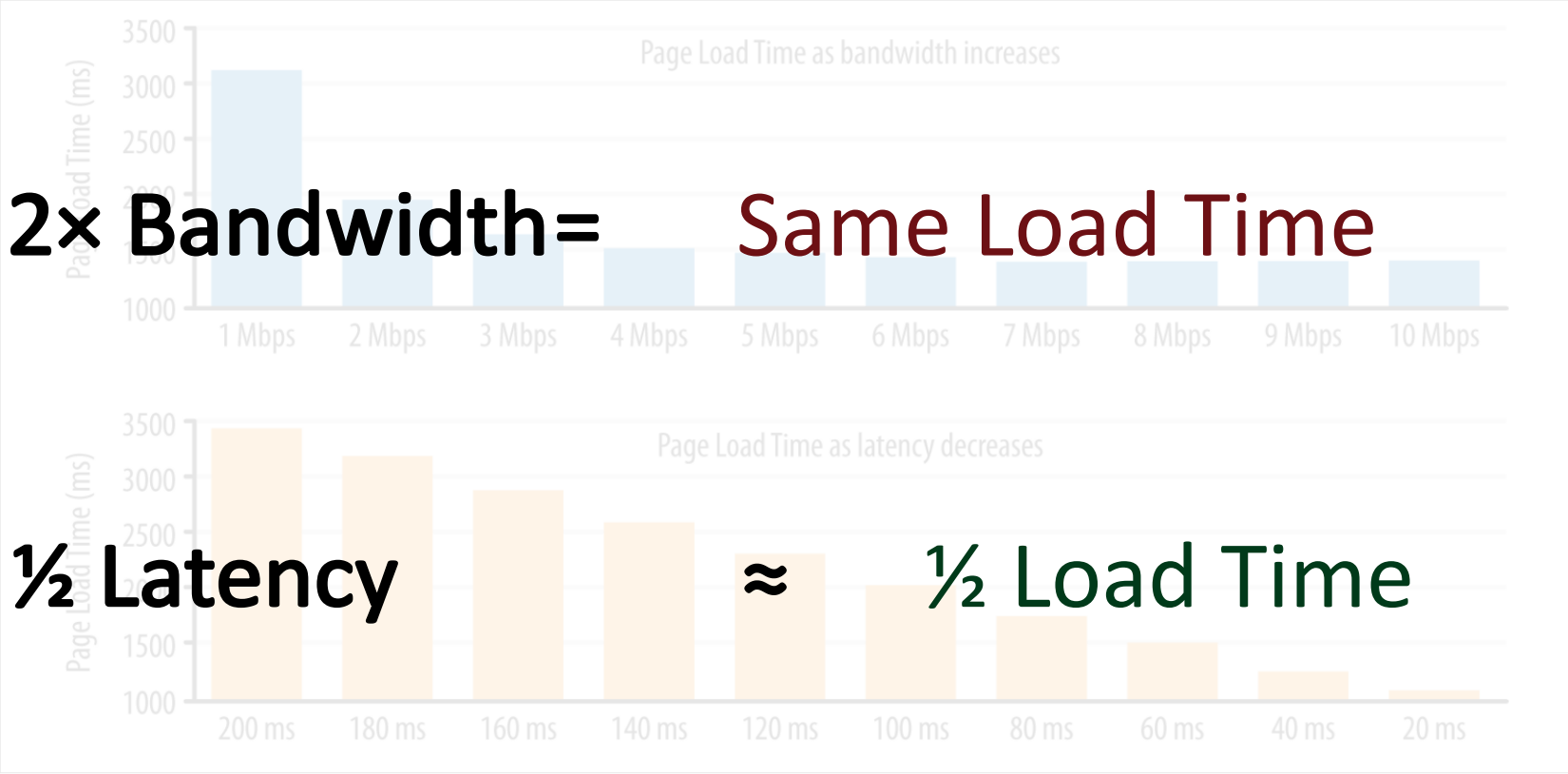


Network Latency: Impact



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

Network Latency: Impact



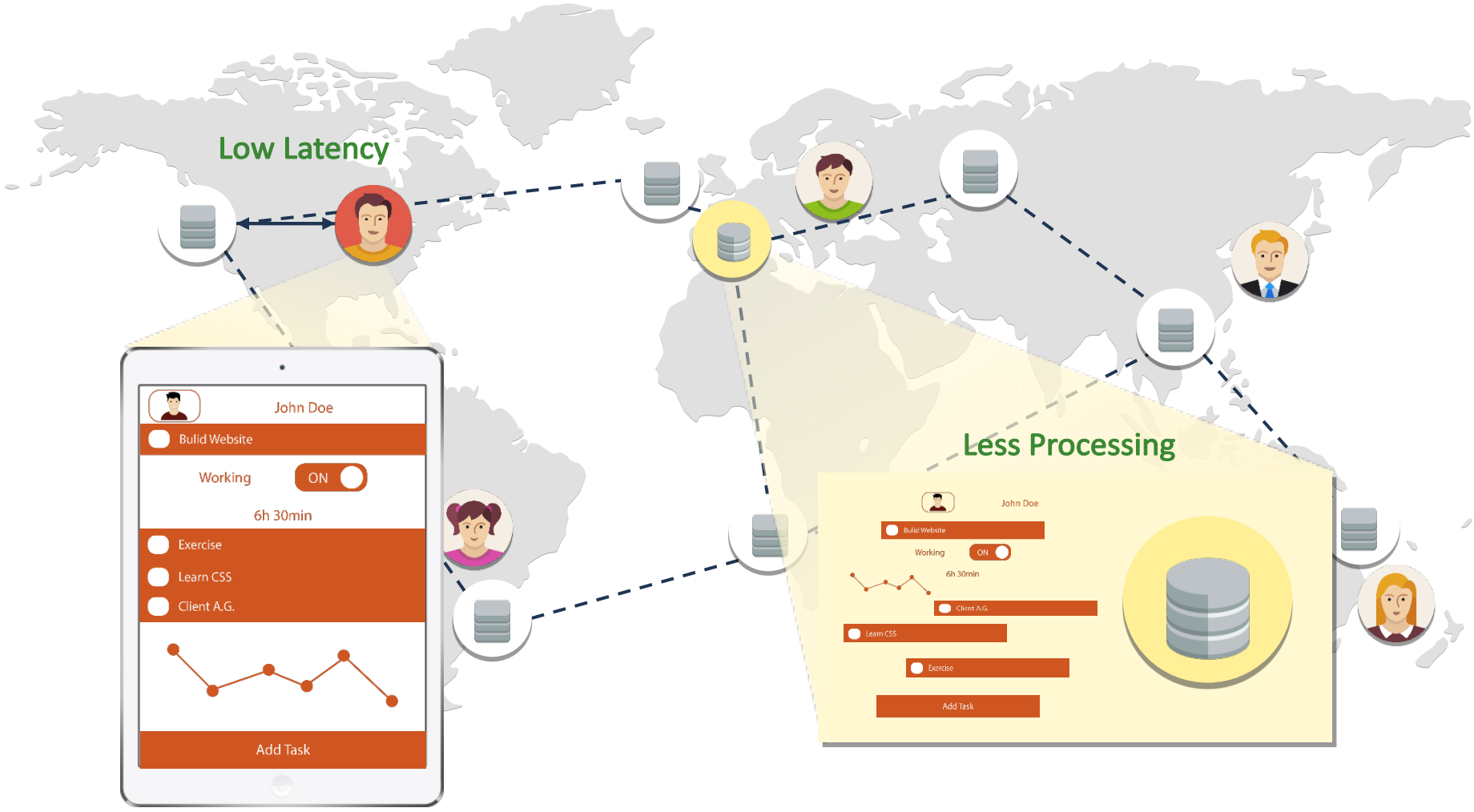
2× Bandwidth= Same Load Time

½ Latency ≈ ½ Load Time

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

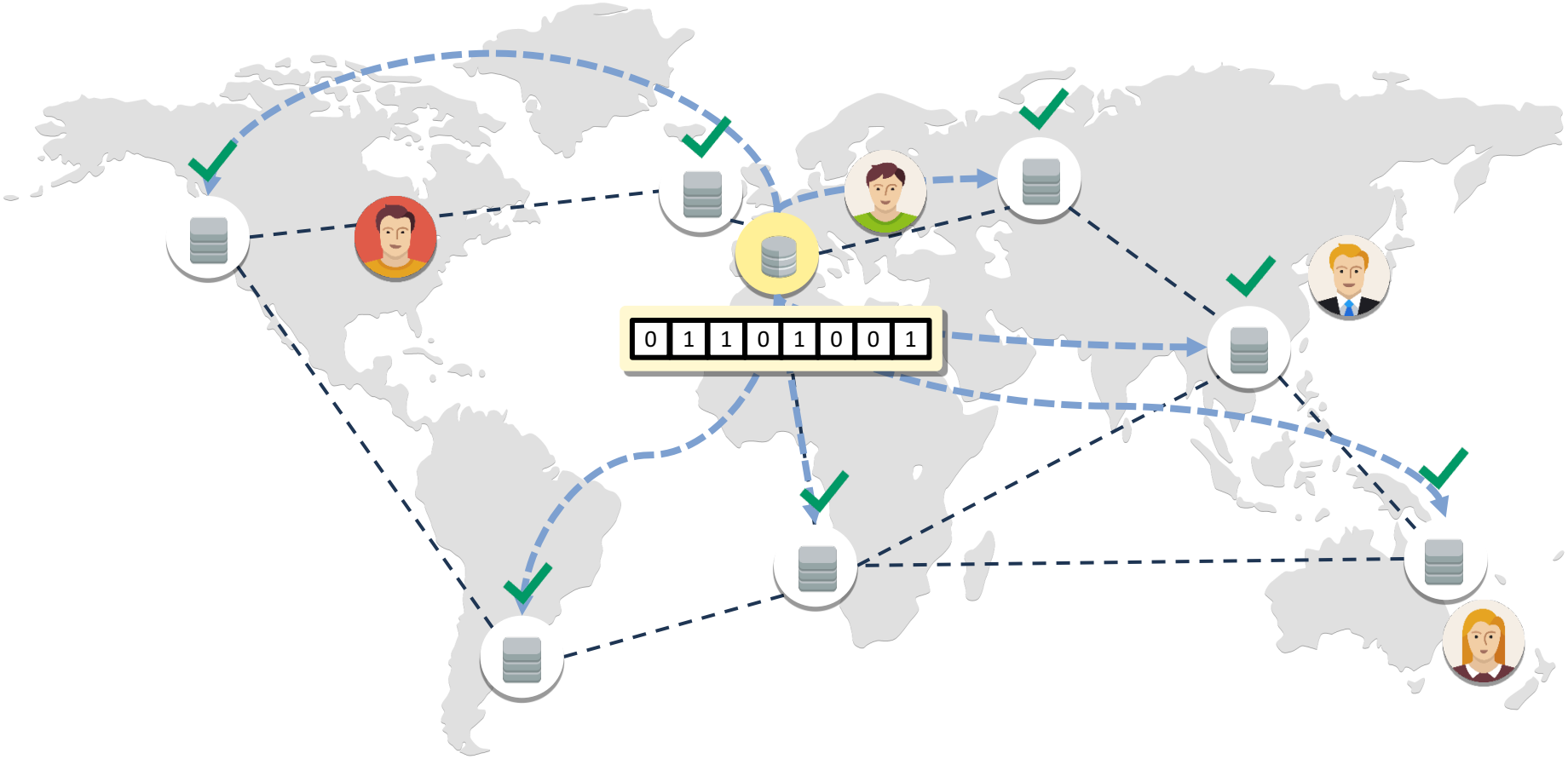
Solution: Global Caching

Fresh Data from Ubiquitous Web Caches



New Caching Algorithms

Solve Consistency Problem



New Caching Algorithms

Solve Consistency Problem

7 Years
Research &
Development

Universität Hamburg

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: e… auf Cloud-Datenbanken“, in Informatik…

F. Gessert und F. Bücklers, *Performanz… vermittels der Web-Caching-Hierarchie…*

M. Schaarschmidt, F. Gessert, und N. R… Persistence“, in BTW 2015.

S. Friedrich, W. Wingerath, F. Gessert, … Survey“, in *44. Jahrestagung der Gesel…* 704.

W. Wingerath, F. Gessert, S. Friedrich, N. Ritter „Real-time stream processing for Big Data“, *Big Data Analytics it - Information Technology,* 2016

F. Gessert, W. Wingerath, S. Friedrich, N. Ritter “NoSQL Database Systems: A Survey and Decision Guidance”, *Computer Science - Research and Development,* 2016

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.

…ath, S. Friedrich, und N. Ritter, „The …d Caching in the Age of Cloud Data…

…eb-Caching von Datenbankobjekten im…

…t, „Who Watches the Watchmen? On …rking“, in BTW 2015.

…d-Datenbanken in Forschung und…

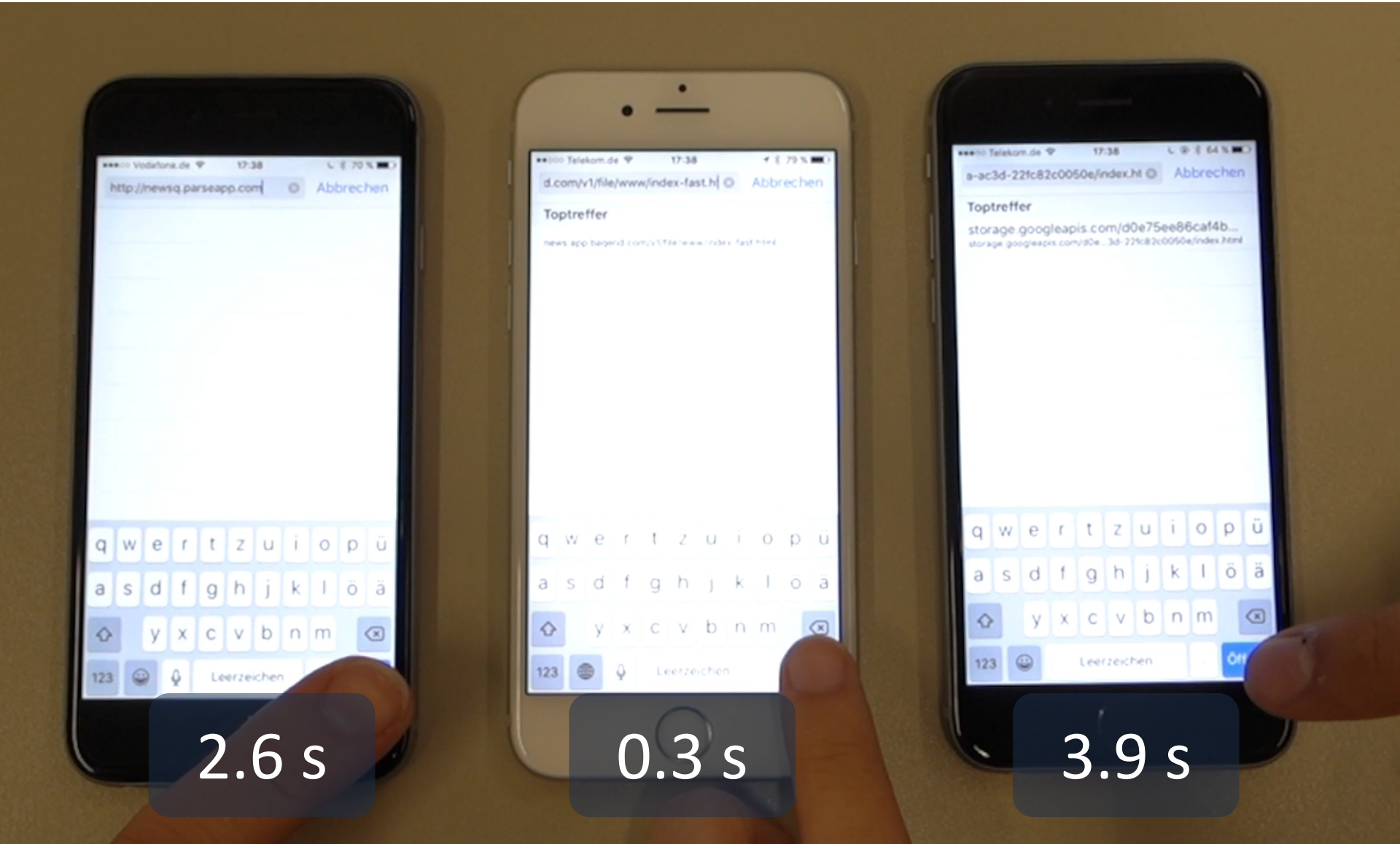
F. Gessert, N. Ritter „Scalable Data Management: NoSQL Data Stores in Research and Practice“, *32nd IEEE International Conference on Data Engineering, ICDE,* 2016

F. Gessert, N. Ritter „Polyglot Persistence“, *Datenbank Spektrum*, 2016.

Parse

Baqend

Kinvey

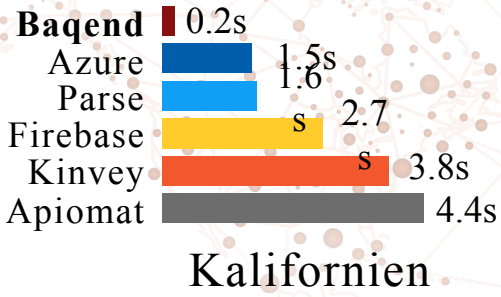


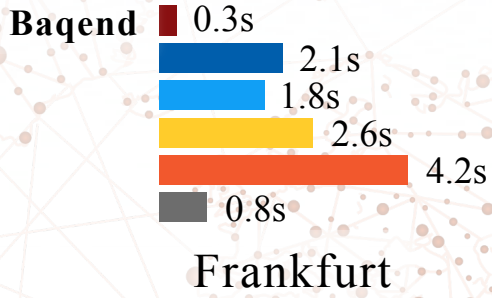
Advantage: 15x Faster

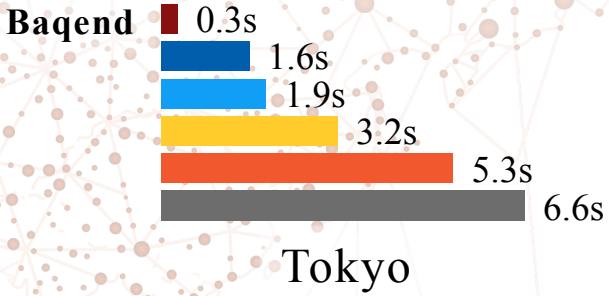
What impact does Baqend have in practice?

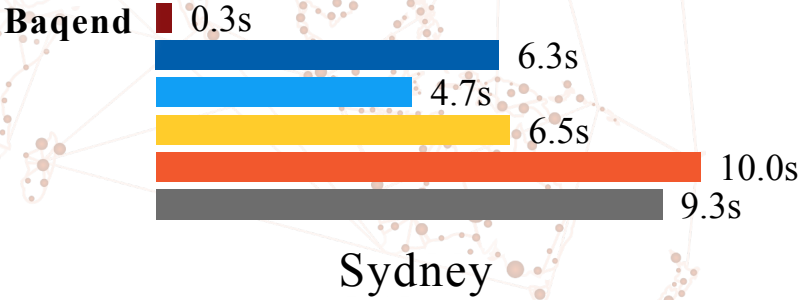
TRY THIS

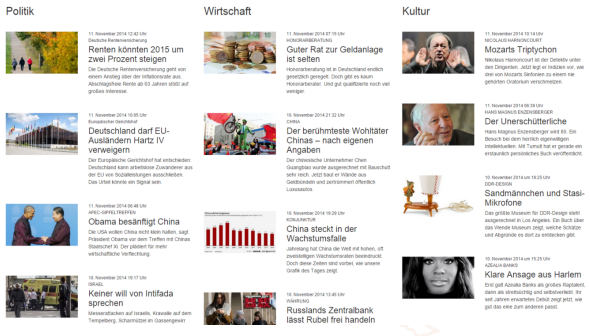
benchmark.baqend.com





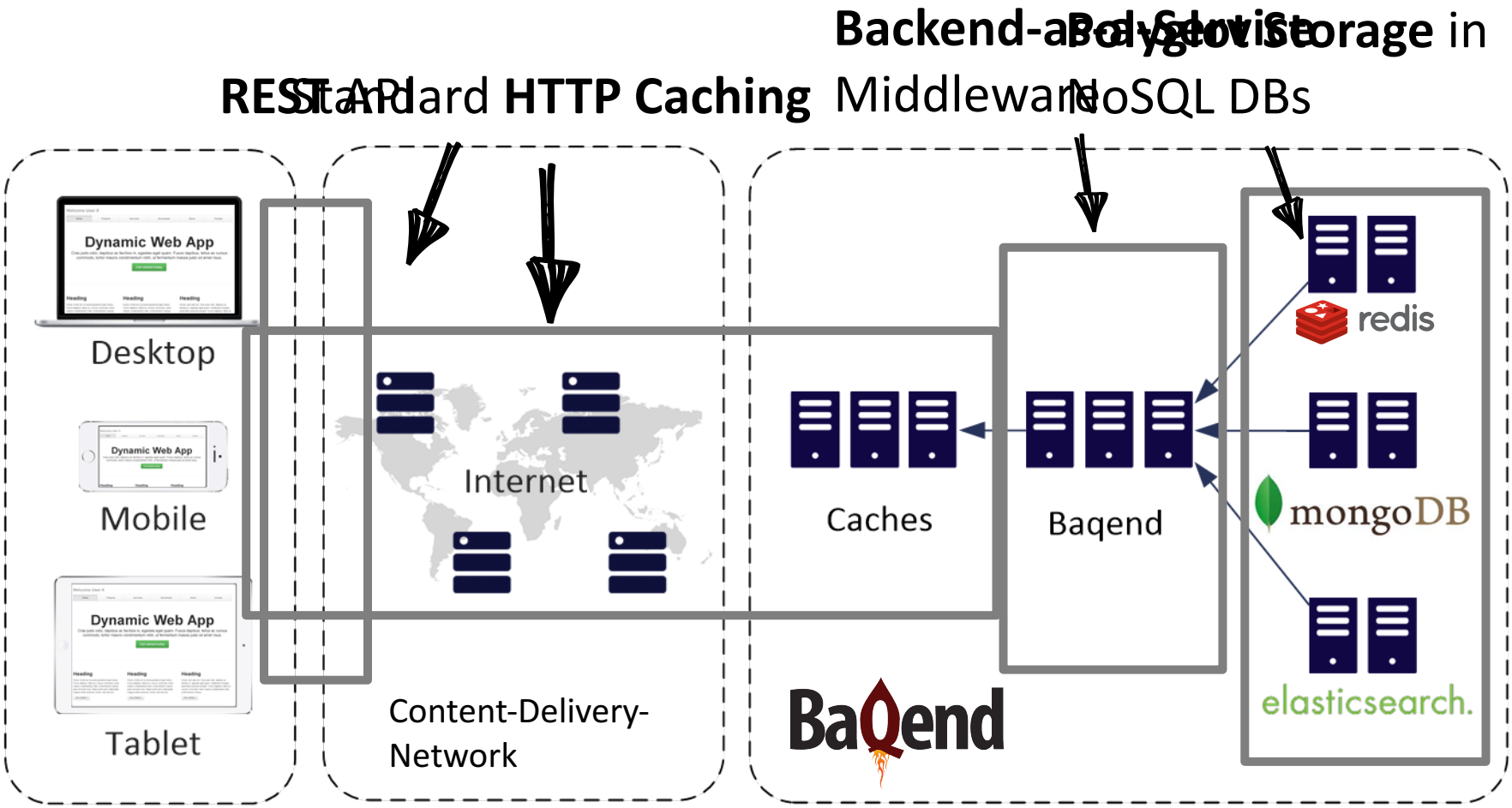






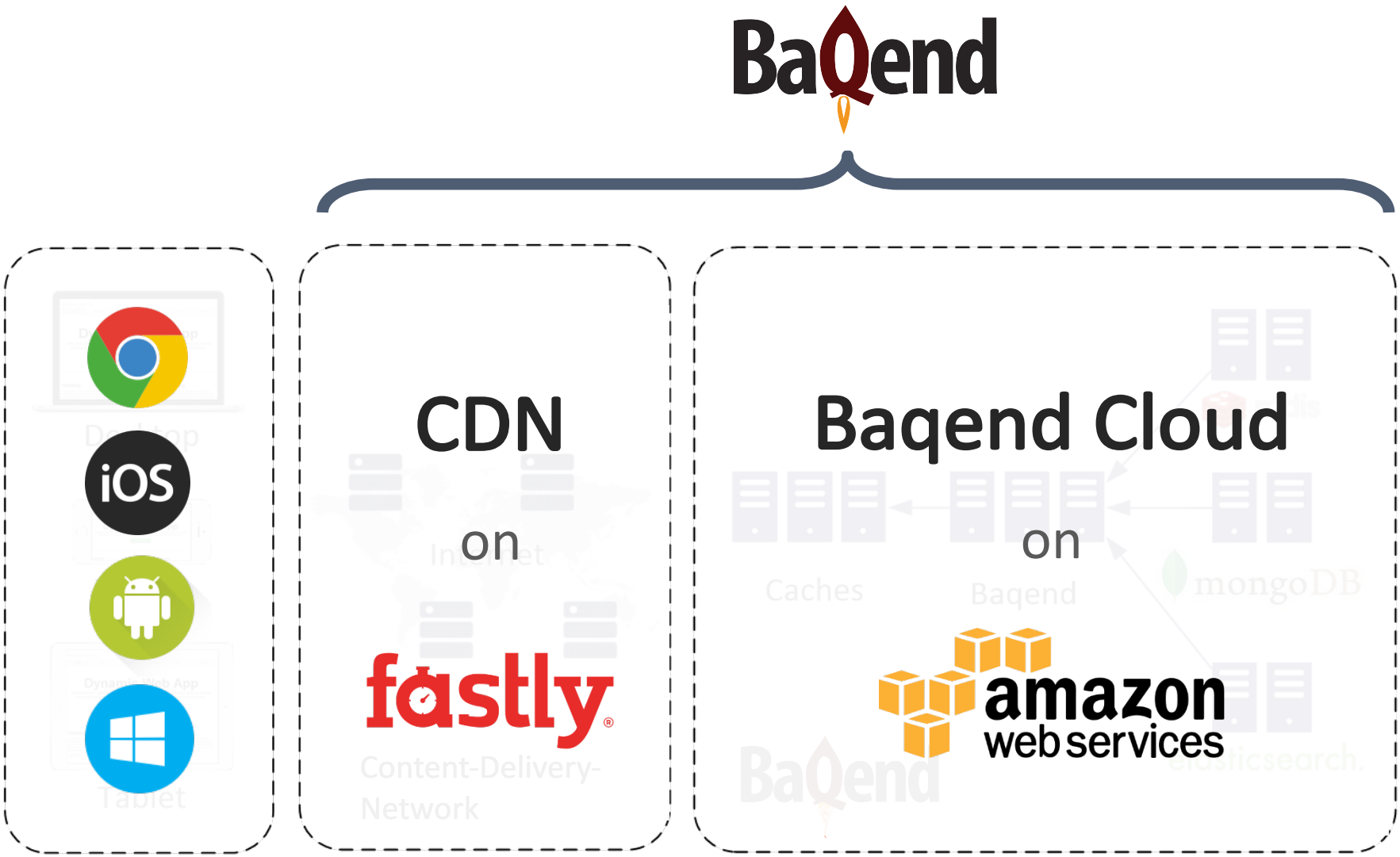
Baqend Architecture

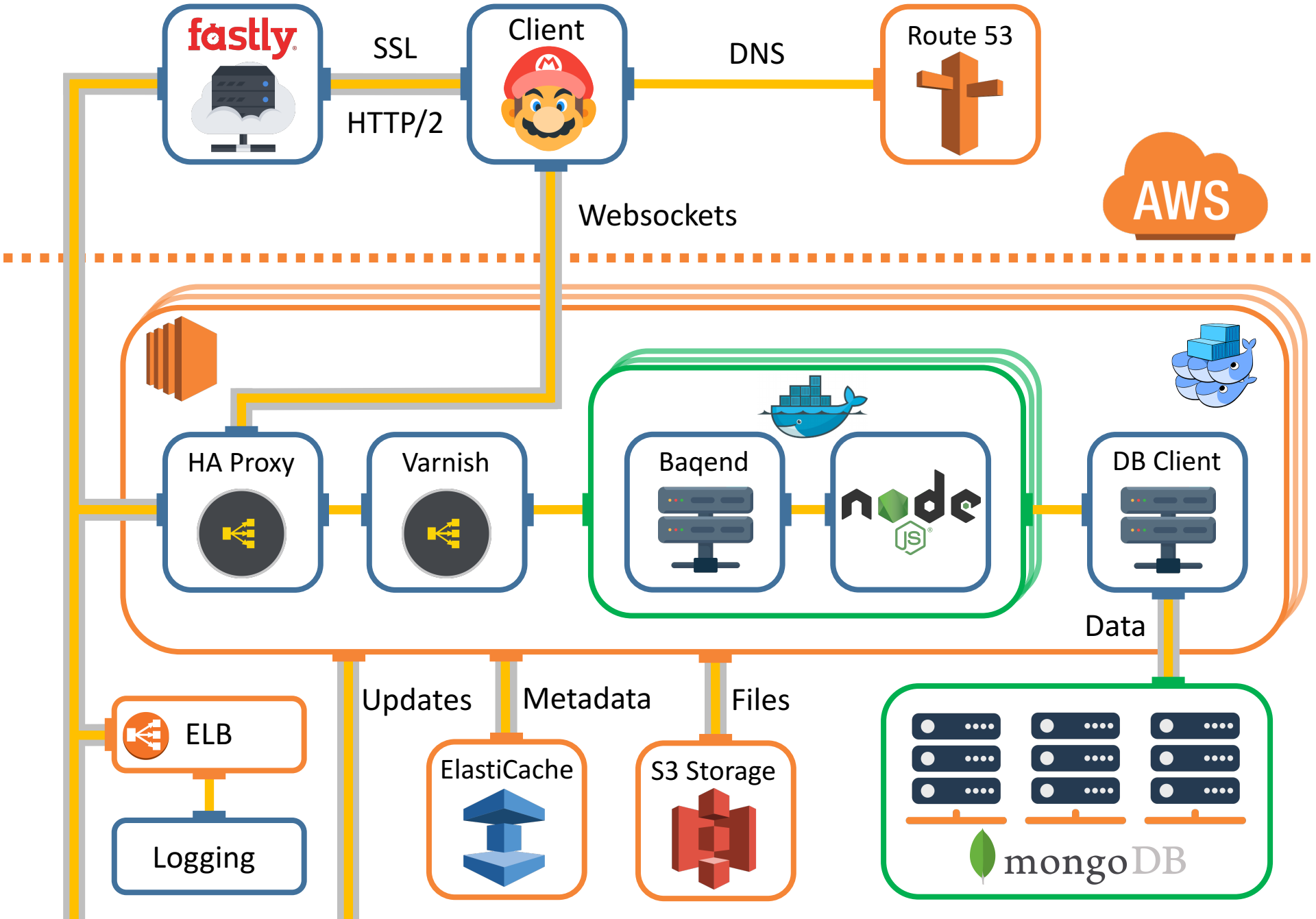
AWS Infrastructure



Baqend Architektur

AWS Infrastructure

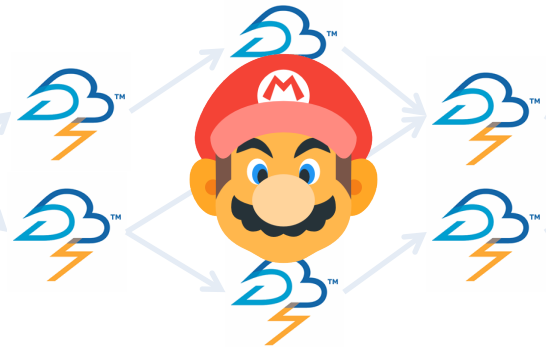



fastly
SSL
HTTP/2
Client
DNS
Route 53
AWS
Websockets
HA Proxy
Varnish
Baqend
node
DB Client
Data
ELB
Logging
Updates
Metadata
Files
ElastiCache
S3 Storage
mongoDB

Updates

Updated Query

Storm Query Matching



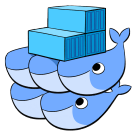
Pub/Sub

Pub/Sub

Management and Orchestration Server

Manager

Swarm



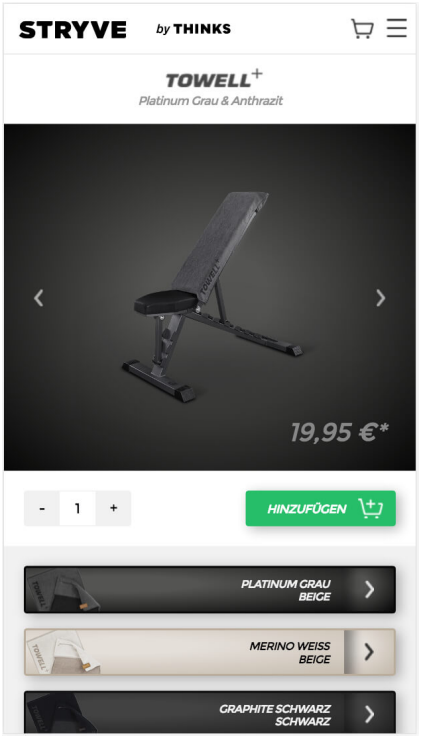
Nimbus

Zookeeper

Example Use Case

Heavy Traffic Peaks

DIE HÖHLE DER LÖWEN



Expected:

- **3.5M** viewers
- **300,000** visitors in 30 minutes
- **20,000** requests per second
- **4 weeks** for development & tests

BaQend

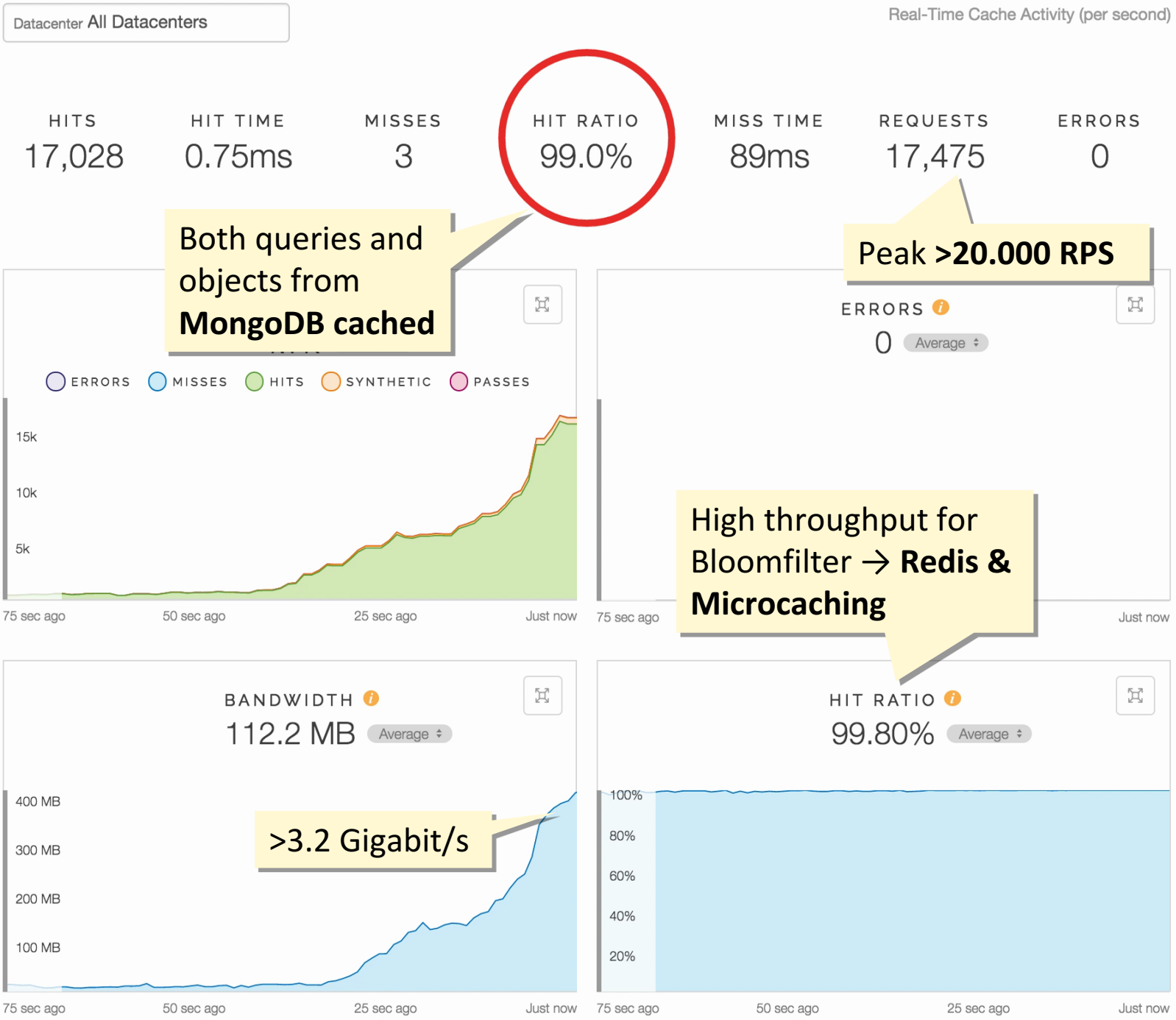
STRYVE by THINKS Shop

< 1 second
Page Load Time

Aktuell
46557
aktive Nutzer auf der Website
MOBILGERÄT TABLET DESKTOP
77% 15% 9%
Concurrent Users

7.8%
Conversion Rate

3%
Server Usage


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
Both queries and objects from **MongoDB cached**
Peak **>20.000 RPS**
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
Just now
High throughput for Bloomfilter → **Redis & Microcaching**
BANDWIDTH 112.2 MB Average
400 MB
300 MB
200 MB
100 MB
>3.2 Gigabit/s
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

High-Performance Backend-as

Hannes

www.baqend.com

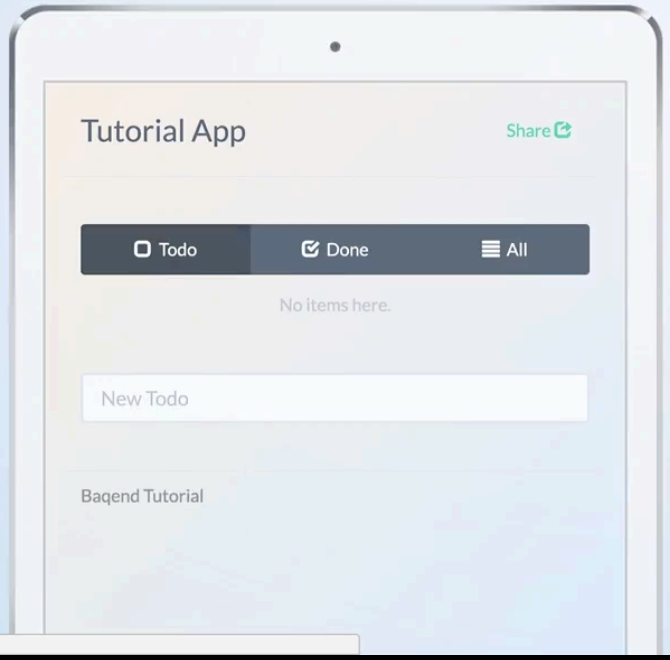
BaQend

Product Developer About us Blog Press

Log in or Sign Up Free

The World's Fastest Backend

Build websites and apps that load instantly.



THE BAQEND PLATFORM

Sky-rocket your Development

Start building now. Baqend Cloud is free and easy to get started with.

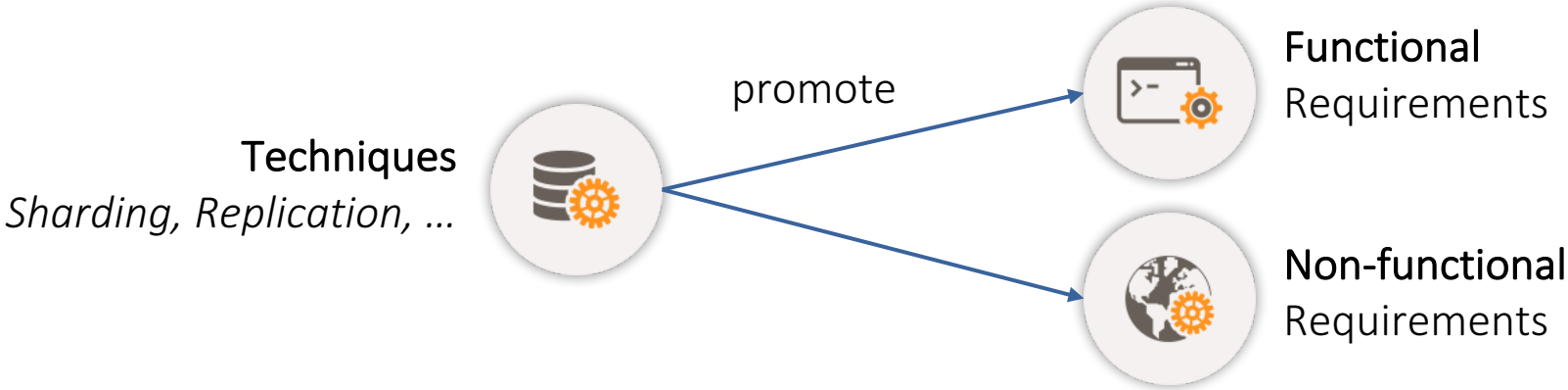
TUTORIAL TRY BAQEND

Warten auf s.codepen.io...

Let's chat - Online

Summary

- Categories **CAP** and **data model** too coarse
- **NoSQL Toolbox:**



- Cloud-DBs for **single-page apps**: Backend-as-a-Service
- **In-depth**: blog.baqend.com