



# Cache Sketches

## Using Bloom Filters and Web Caching Against Slow Load Times

Felix Gessert, Florian Bücklers

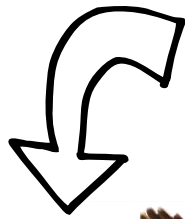
{fg,fb}@baqend.com



@baqendcom

# Who we are

- ▶ Felix Gessert, Florian Bücklers



Research Project since 2010



Orestes



Backend-as-a-Service **Startup** since 2014

## Introduction

## Main Part

## Conclusions



Web Performance:  
State of the Art



Cache Sketch:  
Research Approach



Using Web Caching in  
Applications

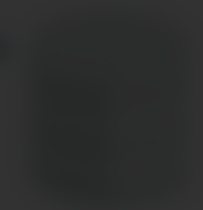
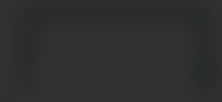
# Backend Abstracts

## Backend Abstracts

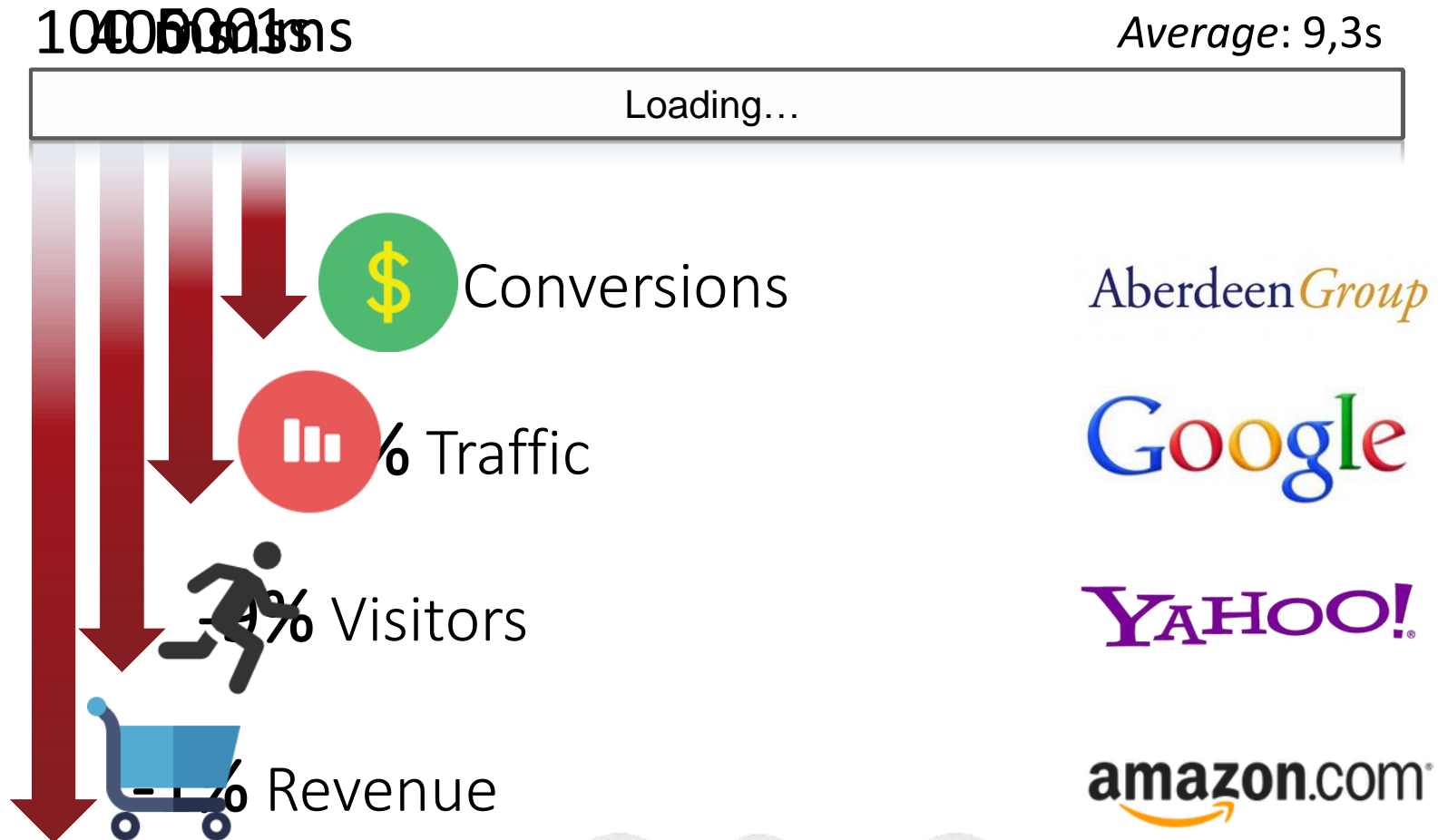
Backend Abstracts is a web application that provides a platform for users to create and manage their abstracts. The application is built using a modern web framework and is designed to be user-friendly and easy to use.



Presentation  
is loading



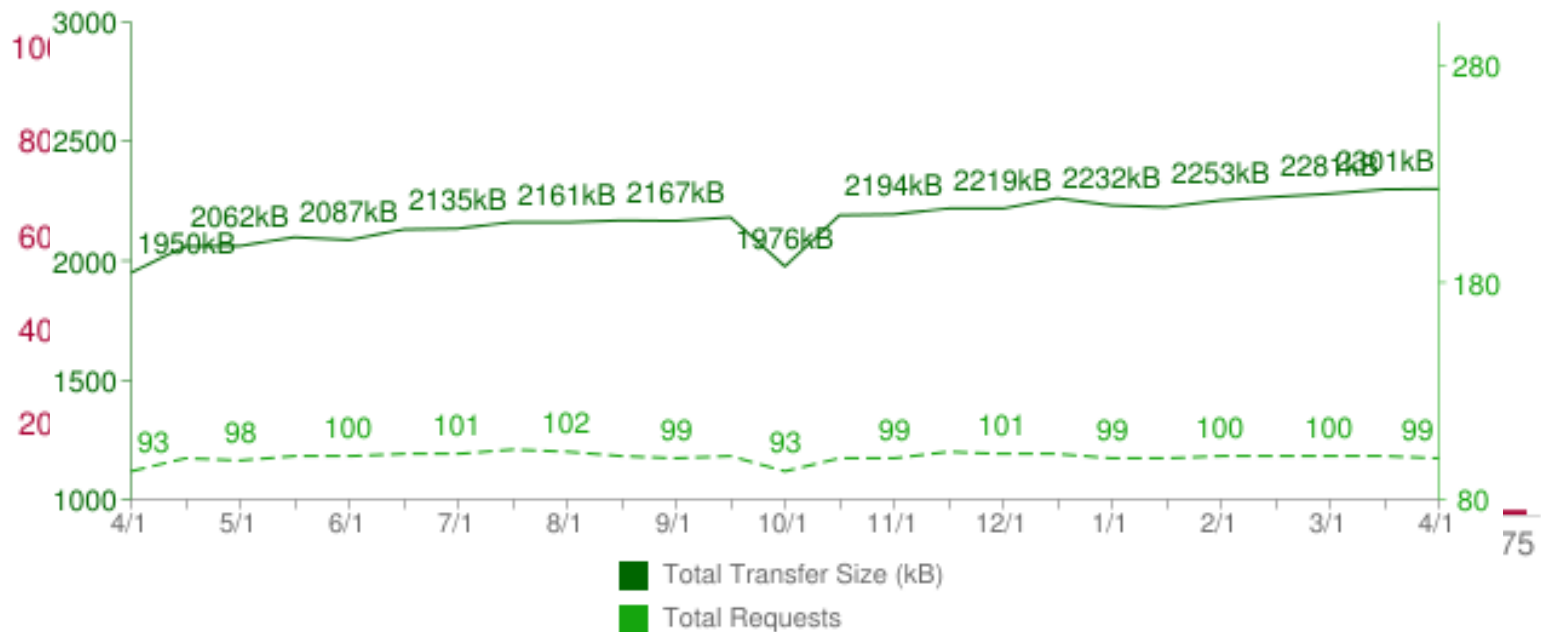
# Why performance matters

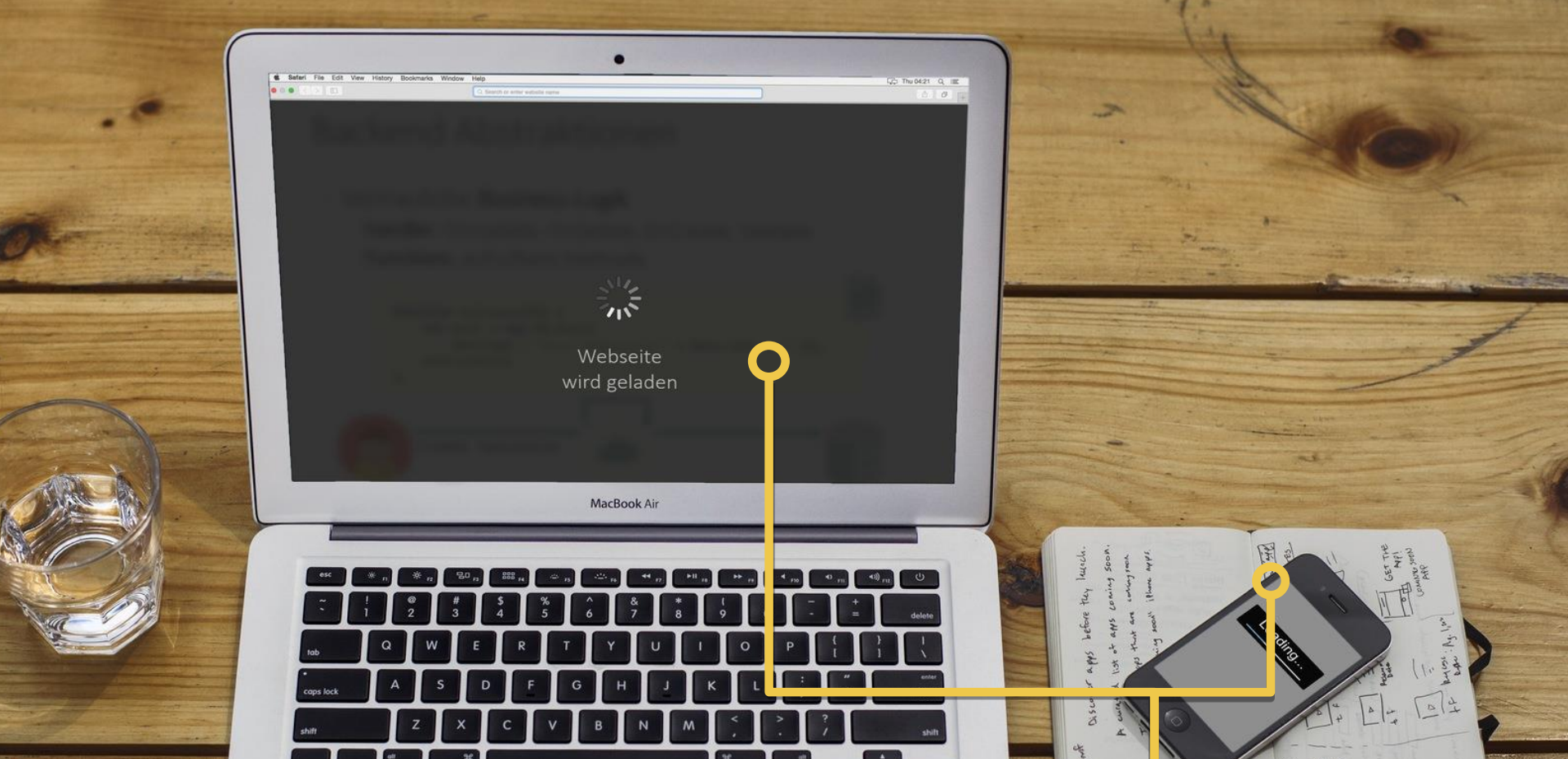


# An Average Website

## Some Statistics

Total Transfer Size & Total Requests





If perceived speed is such an important factor

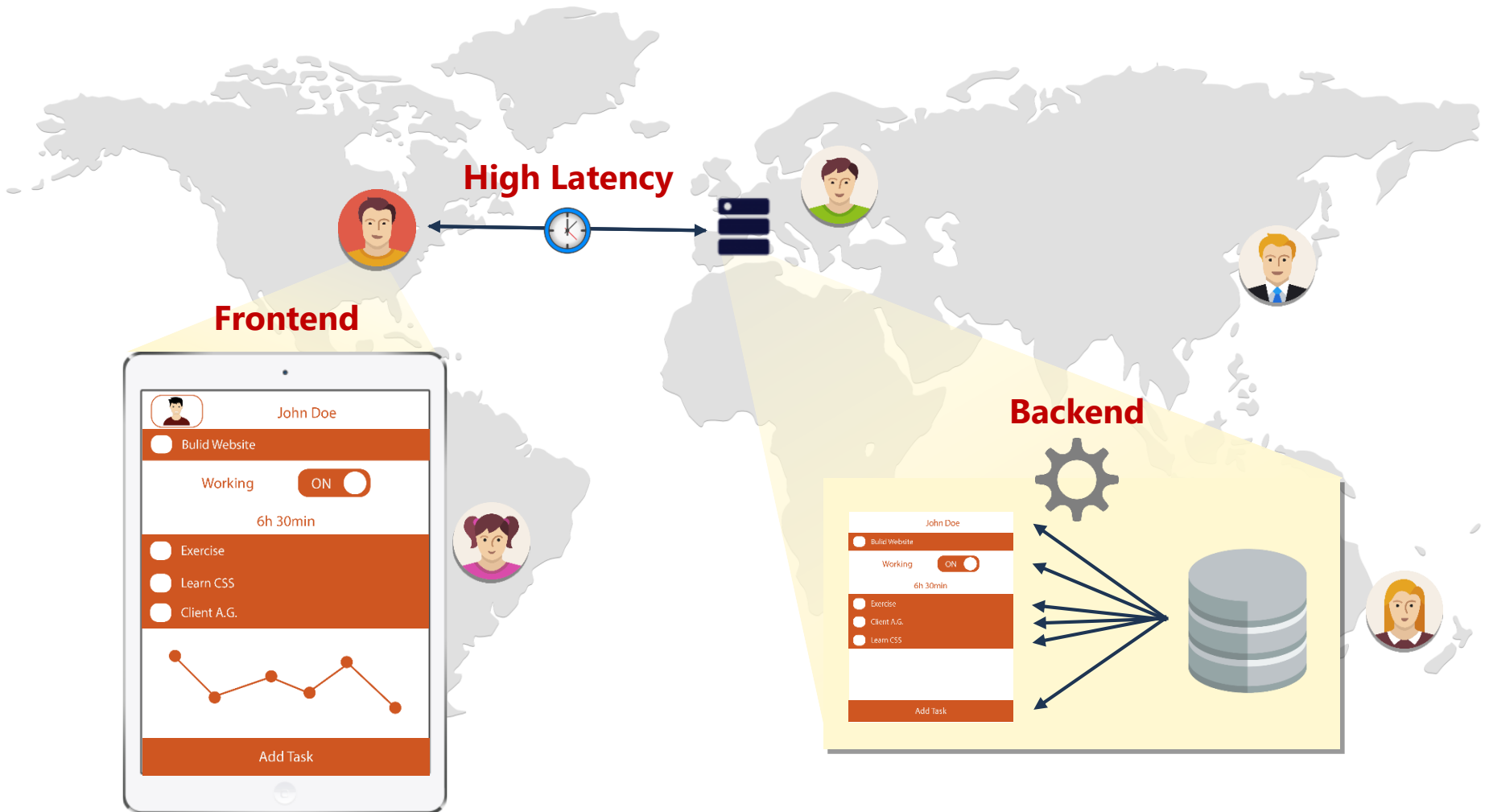


...what causes slow page load times?



# The Problem

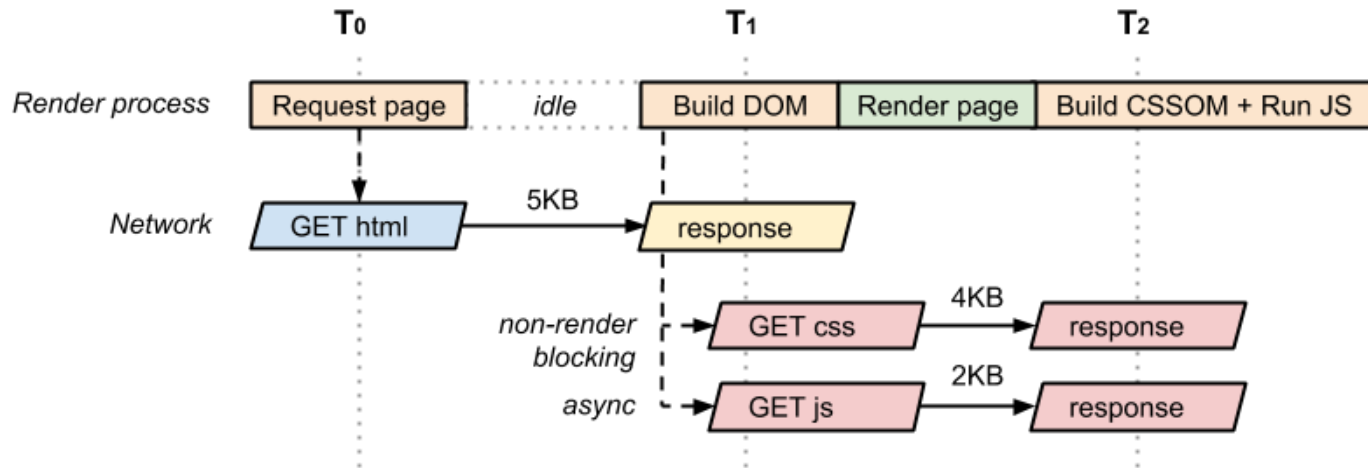
Three Bottlenecks: Latency, Backend & Frontend





# Frontend Performance

## Break-down of the Critical Rendering Path



- ▶ Achieve a fast render of the page by:
  - Reducing the **critical resources** needed
  - Reducing the **critical bytes** which must be transferred
  - Loading JS, CSS and HTML templates **asynchronously**
  - Rendering the page **progressively**
  - **Minifying & Concatenating** CSS, JS and images



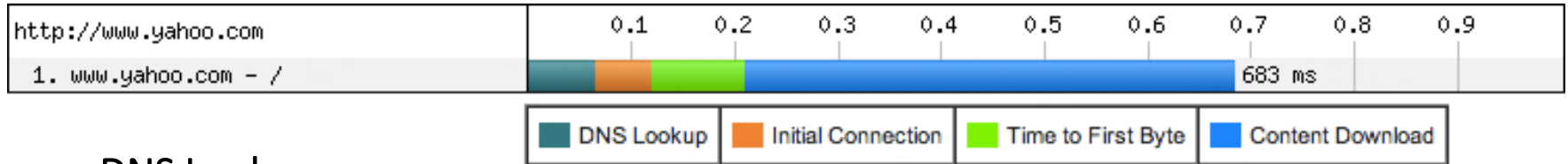
# Frontend Performance

## Tools to improve your page load

- ▶ Well known problem & good tooling:
  - Optimizing CSS (*postcss*)
  - Concatenating CSS and JS (*processhtml*)
  - Minification and Compression (*cssmin*, *UglifyJS*, *Google Closure*, *imagemin*)
  - Inline the critical CSS (*addyosmani/critical*)
  - Hash assets to make them cacheable (*gulp-rev-all*)

# Network Performance

## Break down of a single resource load



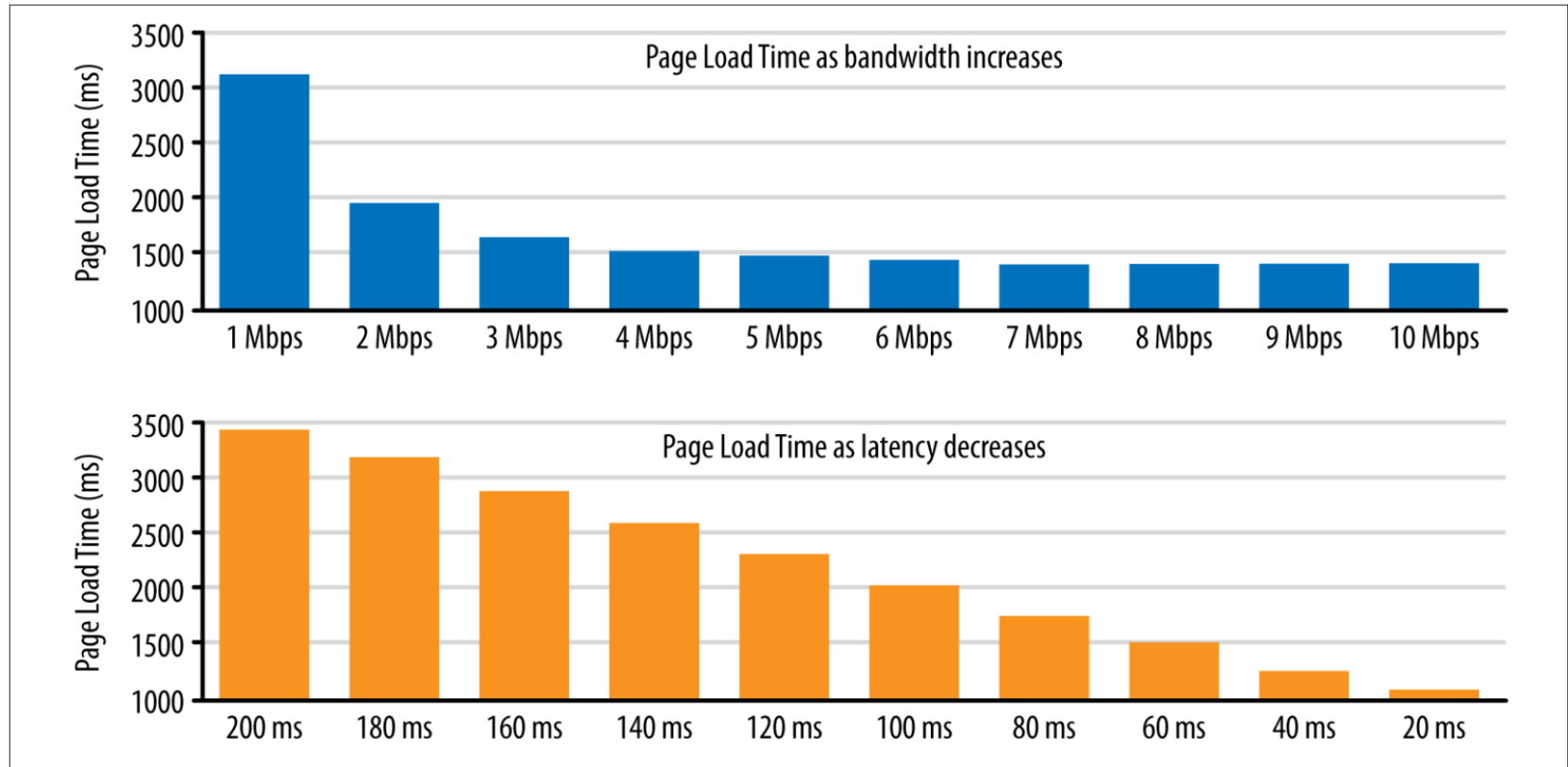
- ▶ **DNS Lookup**
  - Every domain has its own DNS lookup
- ▶ **Initial connection**
  - TCP makes a three way handshake → 2 roundtrips
  - SSL connections have a more complex handshake → +2 roundtrips
- ▶ **Time to First Byte**
  - Depends heavily on the distance between client and the backend
  - Includes the time the backend needs to render the page
    - Session lookups, Database Queries, Template rendering ...
- ▶ **Content Download**
  - Files have a high transfer time on new connections, since the initial congestion window is small → many roundtrips

# Network Performance

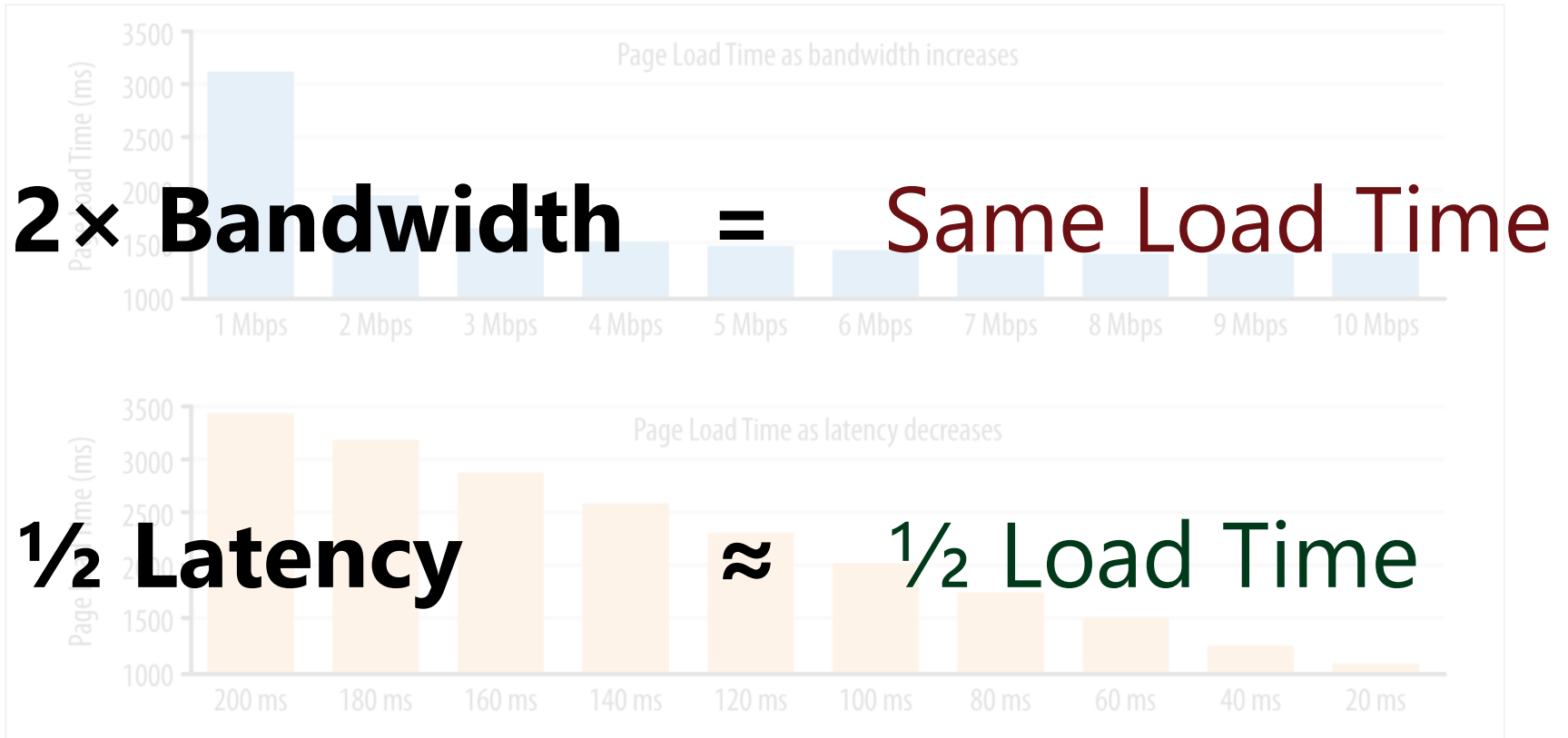
## Common Tuning Knobs

- ▶ Persistent connections, if possible HTTP/2
- ▶ Avoid redirects
- ▶ Explicit caching headers (no heuristic caching)
- ▶ Content Delivery Networks
  - To reduce the distance between client and server
  - To cache images, CSS, JS
  - To terminate SSL early and optimized
- ▶ Single Page Apps:
  - Small initial page that loads additional parts asynchronously
  - Cacheable HTML templates + load dynamic data
  - Only update sections of the page during navigation

# Network Latency: Impact

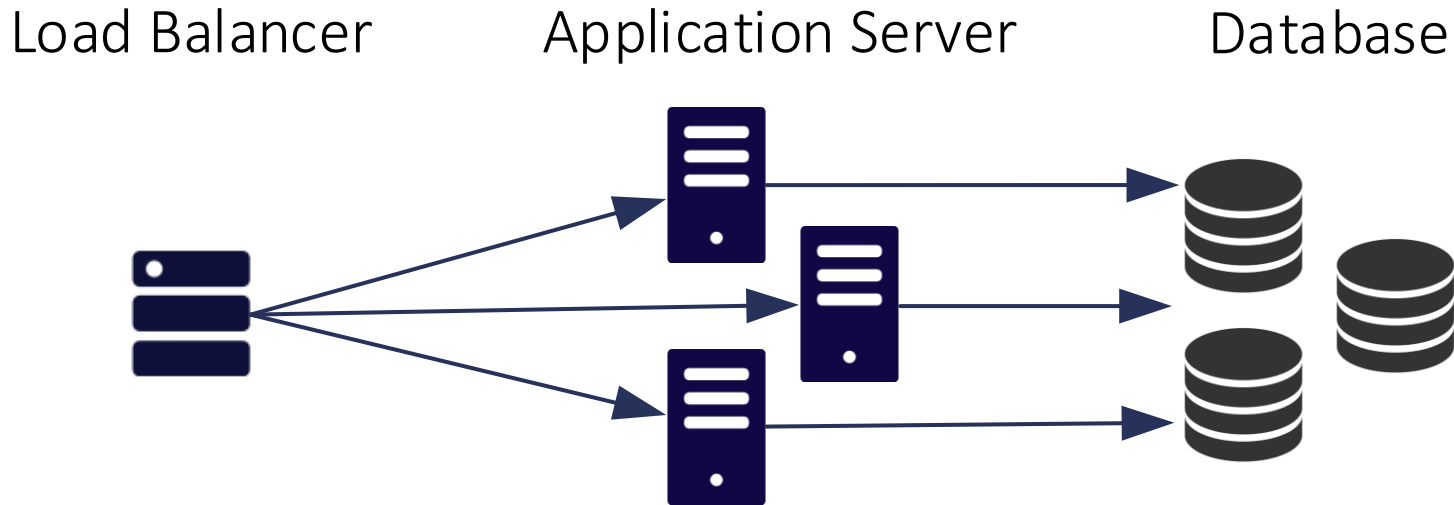


# Network Latency: Impact



# Backend Performance

## Scaling your backend



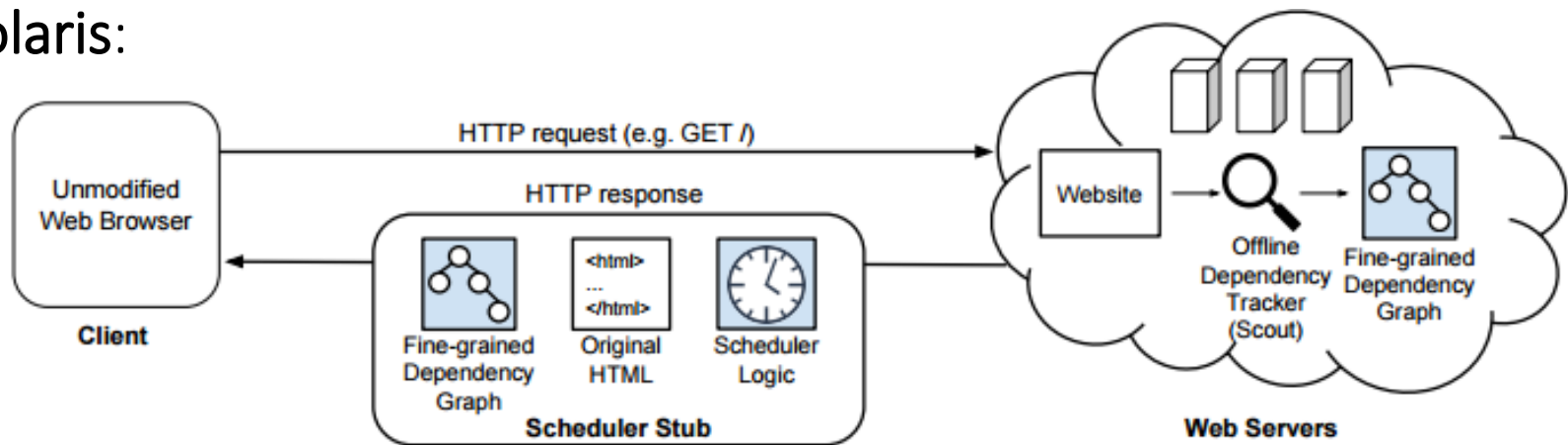
- ▶ Load Balancing
- ▶ Auto-scaling
- ▶ Failover
- ▶ Stateless session handling
- ▶ Minimize shared state
- ▶ Efficient Code & IO
- ▶ Horizontally scalable databases (e.g. “NoSQL”)
  - Replication
  - Sharding
  - Failover



# Research Approaches

## Two Examples

### Polaris:



**Idea:** construct graph that captures real read/write and write/write JS/CSS *dependencies*

**Improvement:** ~30% depending on RTT and bandwidth

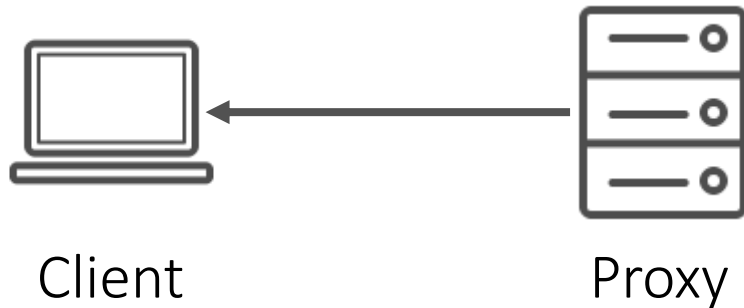
**Limitation:** cannot deal with *non-determinism*, requires server to generate a dependency graph *for each client view*



# Research Approaches

## Two Examples

Shandian:



0.json

```
{
  "loadTimeState": {
    "css": ["#main{font-size:12px;}"],
    "html": {
      "children": [
        {
          "tagName": "body",
          "children": [
            {
              "tagName": "div",
              "id": "main",
              "css": [0]
            }
          ]
        }
      ]
    },
    "postLoadState": {
      "res": ["1.css", "d3.js", "2.js"],
      "heap": "var bar=2;",
      "listeners": [{"load": "done();"}]
    }
  }
}
```

1.css

```
#main{font-size:12px;}
.small{font-size:9px;}
```

2.js

```
var bar = 0;
function foo(){bar++;}
foo(); foo();
d3.select("#main")...
```

**Idea:** Proxy is more powerful than browser, especially mobile

→ evaluate page on proxy

**Improvement:** ~50% for slow Android device

**Limitation:** needs *modified browser*, only useful for *slow devices*



# Other Research Approaches

## Two Examples

Shandian:



Client

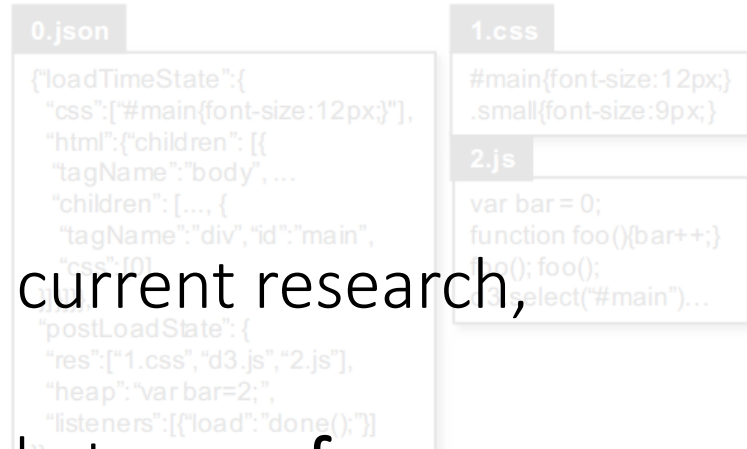
Many good ideas in current research,  
but:

- Only applicable to **very few use cases**
- Mostly require **modified browsers**
- **Small** performance improvements

Idea: Proxy is more powerful than browser especially mobile -> evaluate page on proxy

Improvement: ~50% for slow Android device

Limitation: needs *modified browser*, only useful for slow devices



Wang, Xiao Sophia, Arvind Krishnamurthy, and David Wetherall.  
"Speeding up Web Page Loads with Shandian." NSDI 2016.

# Performance: State of the Art

## Summarized

### Frontend



- Doable with the right set of best practices
- Good support through build tools

### Latency



- Caching and CDNs help, but a considerable effort and only for static content

### Backend

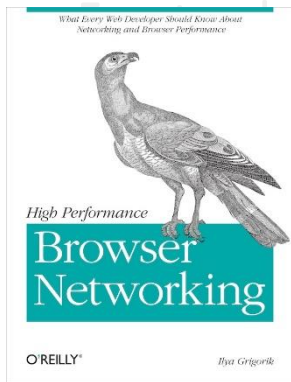


- Many frameworks and platforms
- Horizontal scalability is very difficult

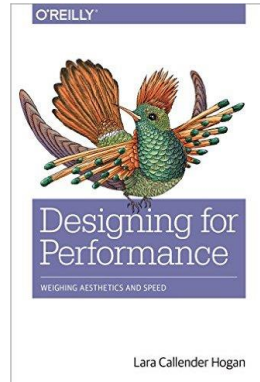
# Performance: State of the Art

## Summarized

### Good Resources:



chimera.labs.oreilly.com/  
books/1230000000545



shop.oreilly.com/product/  
0636920033578.do

Latency

Google Developers

Performance

<https://developers.google.com/web/fundamentals/performance/?hl=en>

Website Performance Optimization  
The Critical Rendering Path

<https://www.udacity.com/course/website-performance-optimization--ud884>

### Good Tools:

PageSpeed Insights 

☒ Mobil ☒ Desktop

[https://developers.google.com/speed/  
pagespeed/](https://developers.google.com/speed/pagespeed/)

GTmetrix

<https://gtmetrix.com>

 WEBPAGETEST

<http://www.webpagetest.org/>

# Performance: State of the Art

## Summarized

### Frontend



- Doable with the right set of best practices
- Good support through build tools

### Latency



- Caching and CDNs help, but large effort and only for static content

### Backend



- Many frameworks and platforms
- Horizontal scalability is very difficult

How to cache & scale  
**dynamic** content?



Introduction

**Main Part**

Conclusions



Web Performance:  
State of the Art



Cache Sketch:  
Research Approach

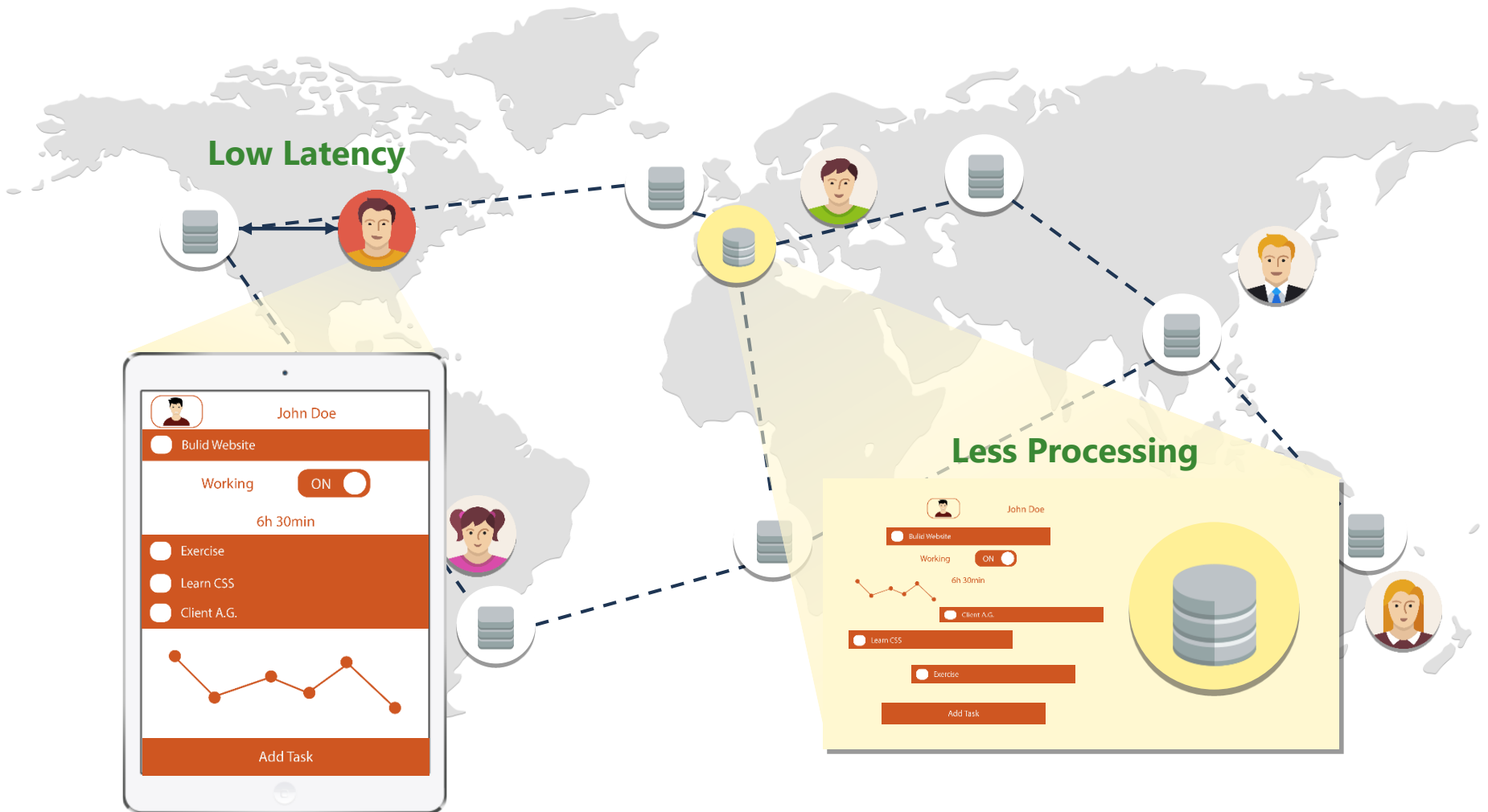


Using Web Caching in  
Applications



# Goal: Low-Latency for Dynamic Content

## By Serving Data from Ubiquitous Web Caches

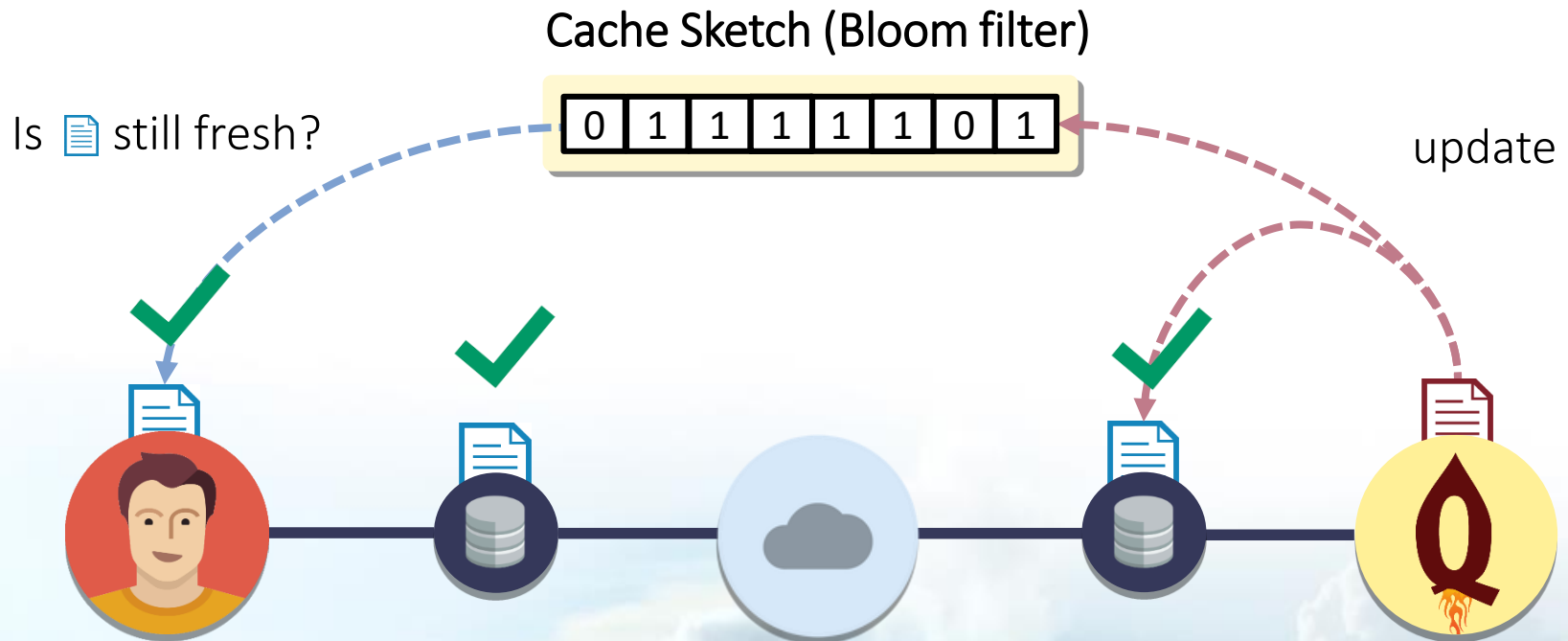


# In a nutshell



# 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 mittels 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*.

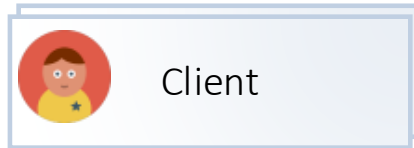


F. Gessert, „Skalierbare NoSQL- und Cloud-Datenbanken in Forschung und Praxis“, *BTW 2015*



# Web Caching Concepts

## Invalidation- and expiration-based caches

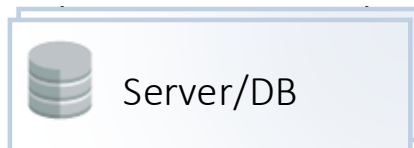


### Expiration-based Caches:

- ▶ An object  $x$  is considered fresh for  $TTL_x$  seconds
- ▶ The server assigns TTLs for each object

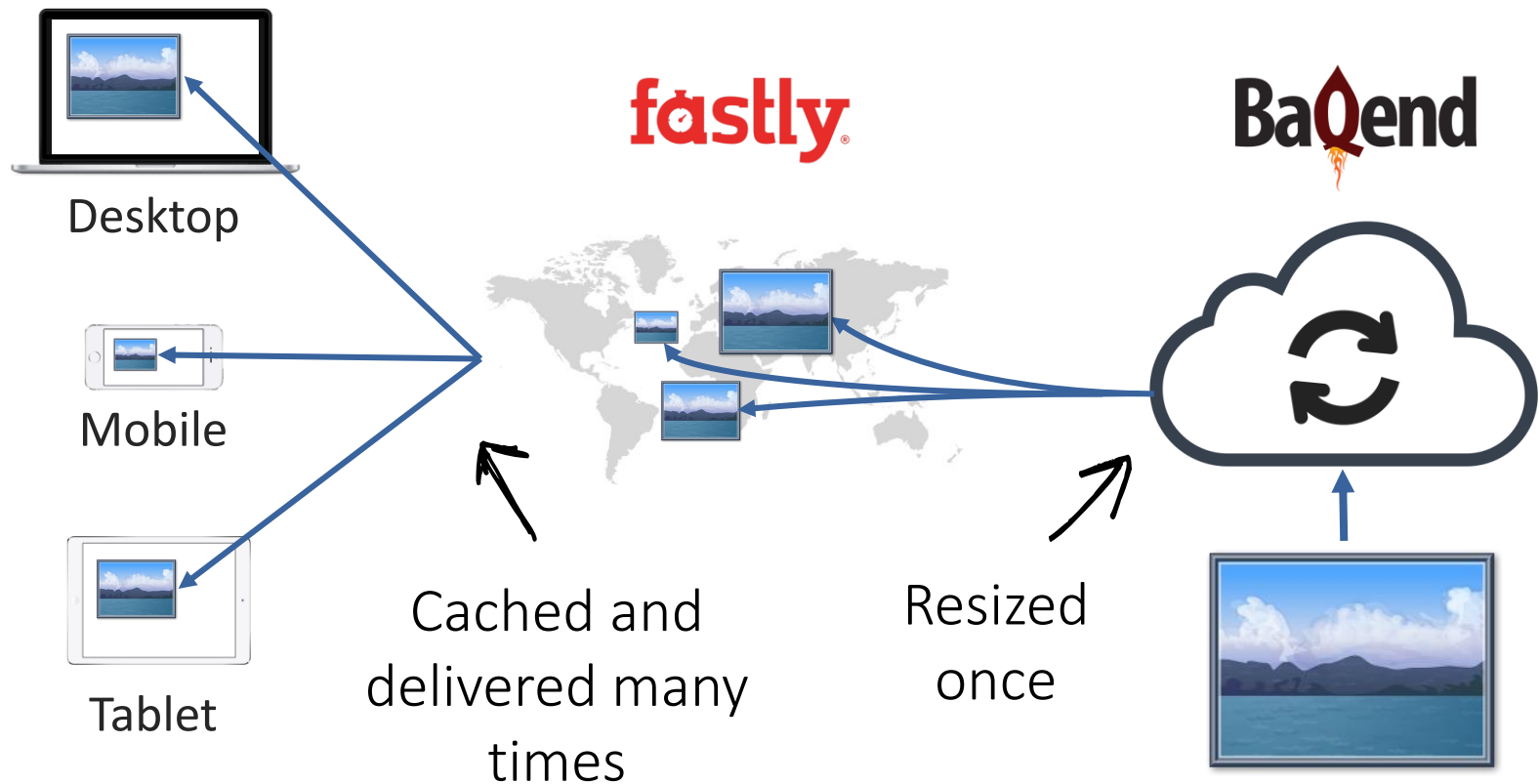
### Invalidation-based Caches:

- ▶ Expose object eviction operation to the server



# Classic Web Caching: Example

A tiny image resizer



# Bloom filter Concepts

## Compact Probabilistic Sets

- ▶ The „Bloom filter principle“:

*“Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated.”*



A. Broder und M. Mitzenmacher, „Network applications of bloom filters: A survey“, Internet Mathematics, 2004.

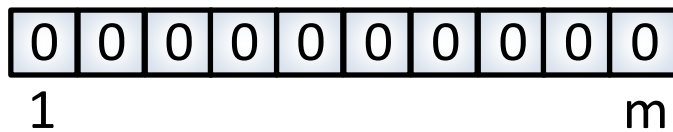
- ▶ Bit array of length **m**
- ▶ **k** independent hash functions
- ▶ **insert(obj)**: add to set
- ▶ **contains(obj)**:
  - ▶ Always returns true if the element was inserted
  - ▶ Might return true even though it was *not* inserted (false positive)

```
def insert(obj):  
    for each position in hashes(obj):  
        bits[position] = 1  
  
def contains(obj):  
    for each position in hashes(obj):  
        if bits[position] == 0:  
            return false;  
    return true
```

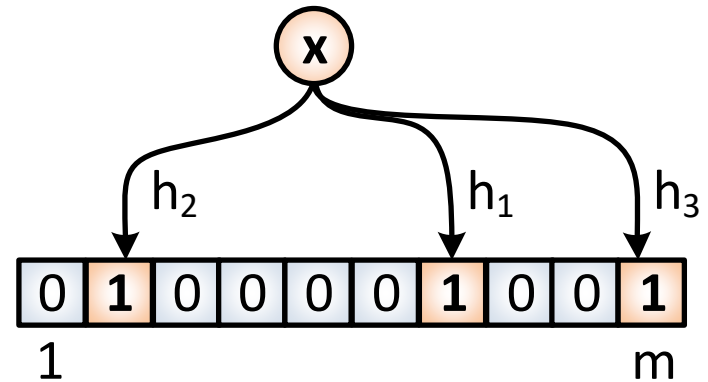


# Bloom filter Concepts

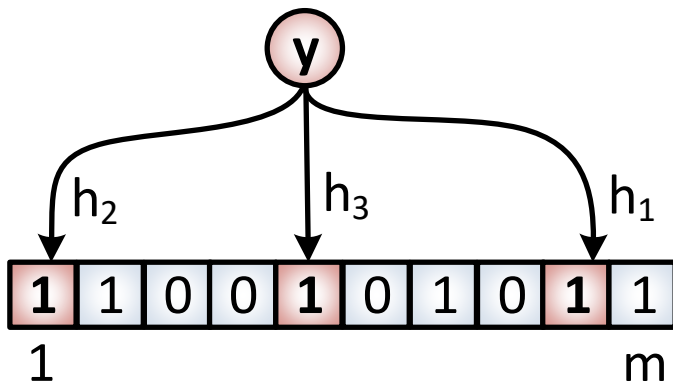
Visualized



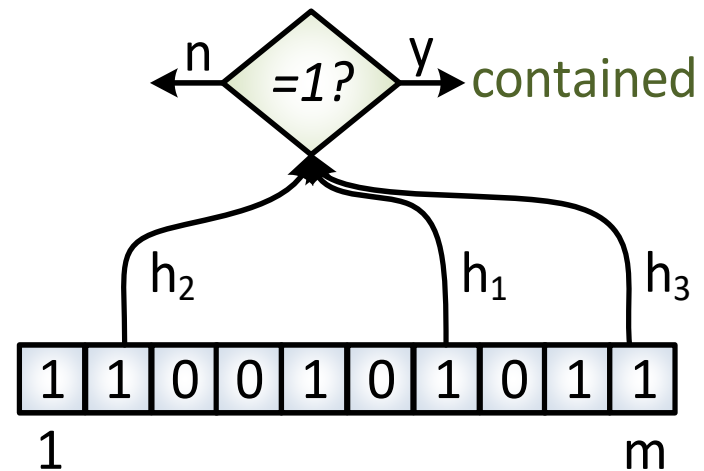
Empty Bloom Filter



Insert x



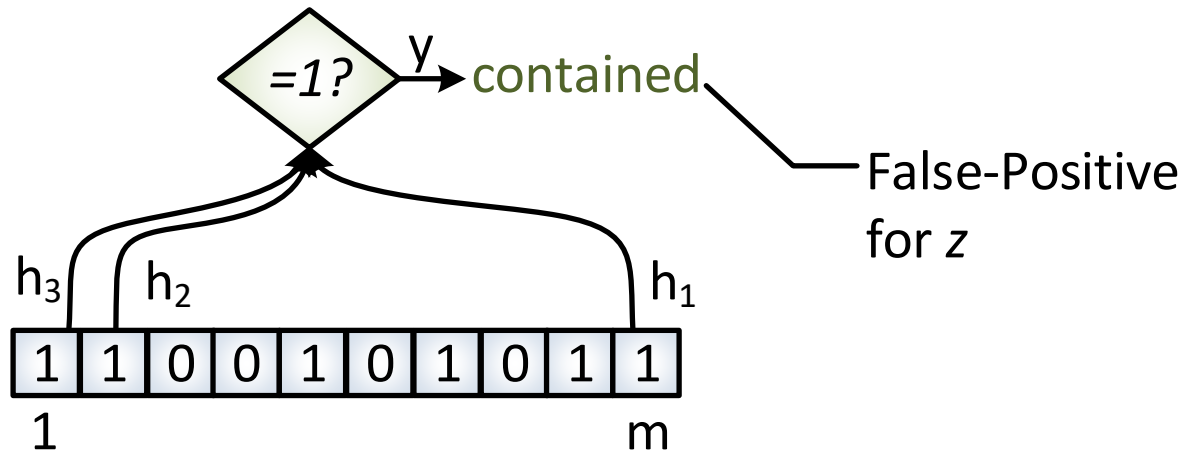
Insert y



Query x

# Bloom filter Concepts

## False Positives



**Query z**


The false positive rate depends on the bits  $m$  and the inserted elements  $n$ :

$$f \approx \left(1 - e^{-\ln(2)}\right)^k \approx 0.6185 \frac{m}{n}$$




For  $f=1\%$  the required bits per element are:  $2.081 \ln(1/0.01) = 9.5$

# Our Bloom filter

## Open Source Implementation

 This repository Search

Pull requests Issues Gist

Baqend / Orestes-Bloomfilter



Unwatch 26 Unstar 153 Fork 50











[Code](#) [Issues 4](#) [Pull requests 0](#) [Wiki](#) [Pulse](#) [Graphs](#) [Settings](#)

Library of different Bloom filters in Java with optional Redis-backing, counting and many hashing options. — Edit

212 commits 1 branch 17 releases 5 contributors

Branch: master [New pull request](#)

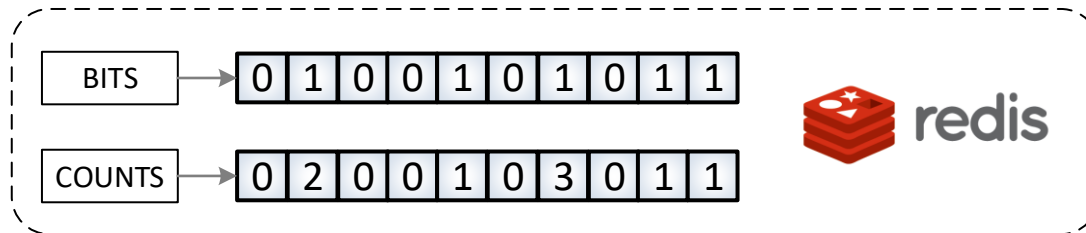
[New file](#) [Upload files](#) [Find file](#) [SSH](#) `git@github.com:Baqend/Orestes`   [Download ZIP](#)

 fbuecklers	gitignore iml	Latest commit 37a0d05 22 hours ago
 gradle/wrapper	cleanup build	a year ago
 src	-Implemented and tested Bloom filter for JS	a day ago
 .gitignore	gitignore iml	22 hours ago
 CHANGELOG.md	Update CHANGELOG.md	8 months ago
 LICENSE	Added Tutorial steps	3 years ago
 README.md	Update README.md	8 months ago
 bloom-filter.iml	-Updated GSON	2 months ago
 build.gradle	-Updated GSON	2 months ago
 gradle.properties	[ci skip] new version commit: '1.1.8-SNAPSHOT'.	14 days ago

# Our Bloom filters

## Example: Redis-backed Counting Bloom Filter

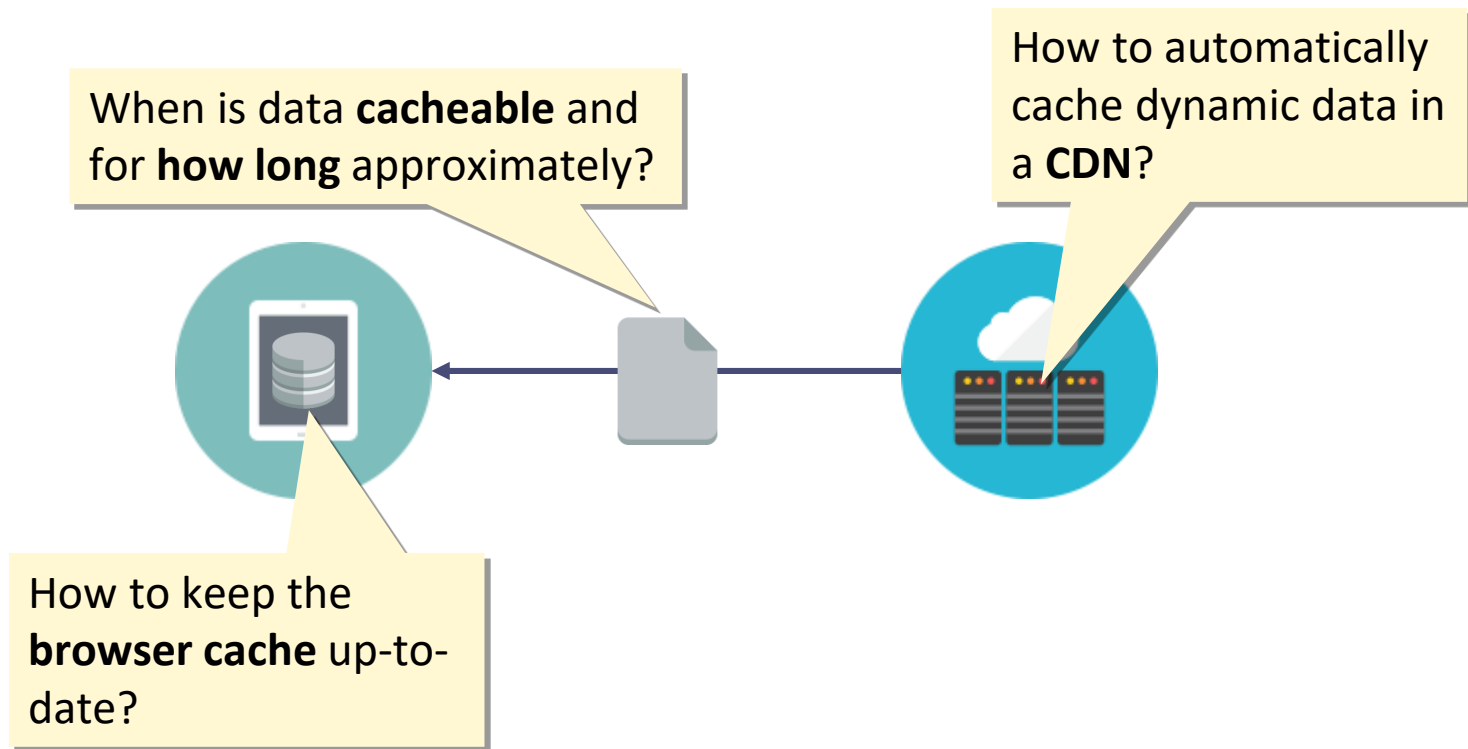
- ▶ Redis-backed Bloom filters:
  - Can be **shared** by many servers
  - Highly **efficient** through Redis' bitwise operations
  - Tunable **persistence**
- ▶ Counting Bloom Filters: use counters instead of bits to also allow **removals**
  - Stores the materialized Bloom filter for fast retrieval



# The Cache Sketch approach

## Caching Dynamic Data

- ▶ **Idea:** use standard *HTTP Caching* for query results and records
- ▶ **Problems:**



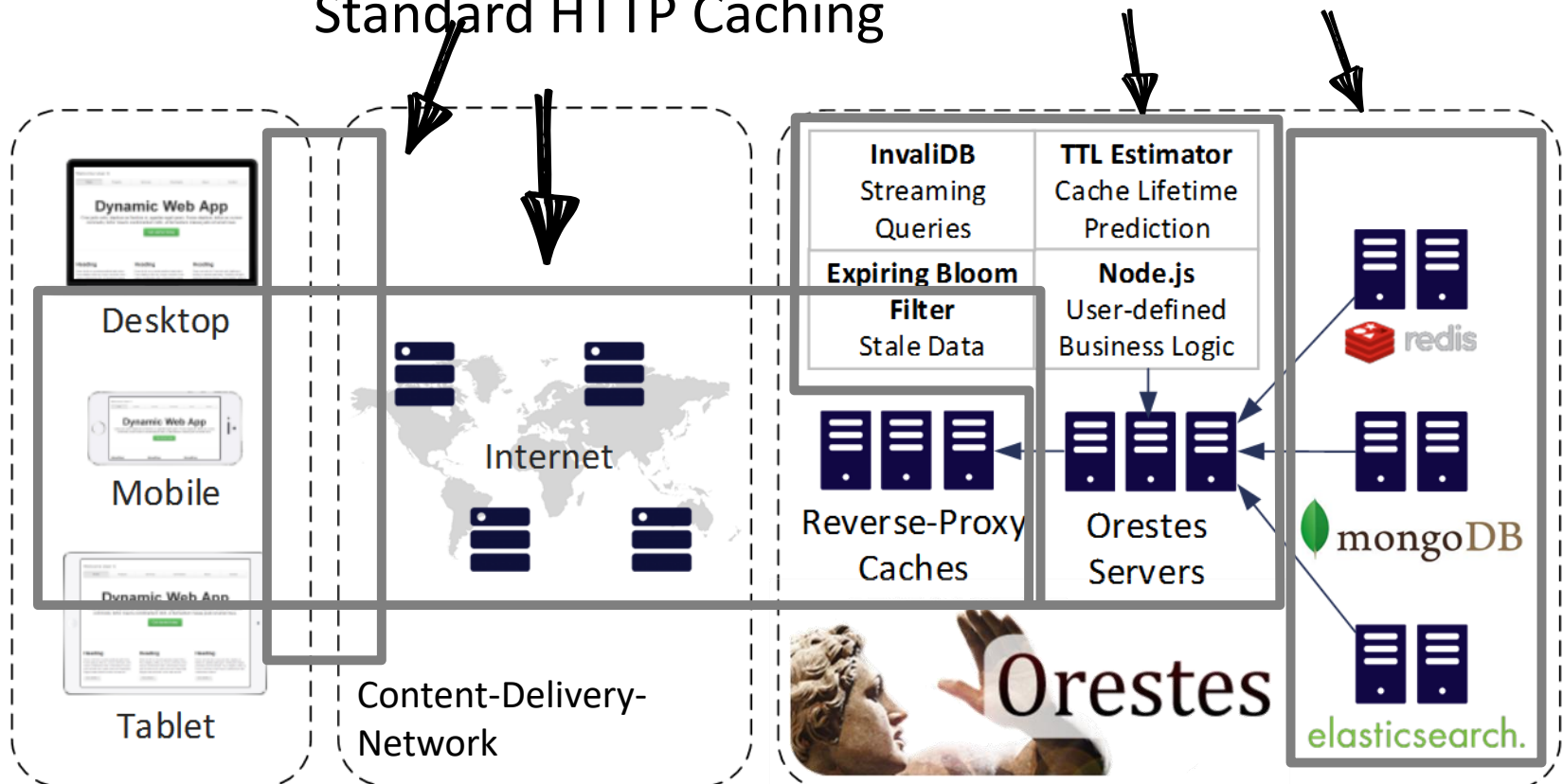
# Orestes Architecture

## Infrastructure

Backend-as-a-Service Middleware:

Caching, Transactions, Schemas,

Unified REST API Invalidation Detection, Not.Storage  
Standard HTTP Caching



# Baqend Architecture

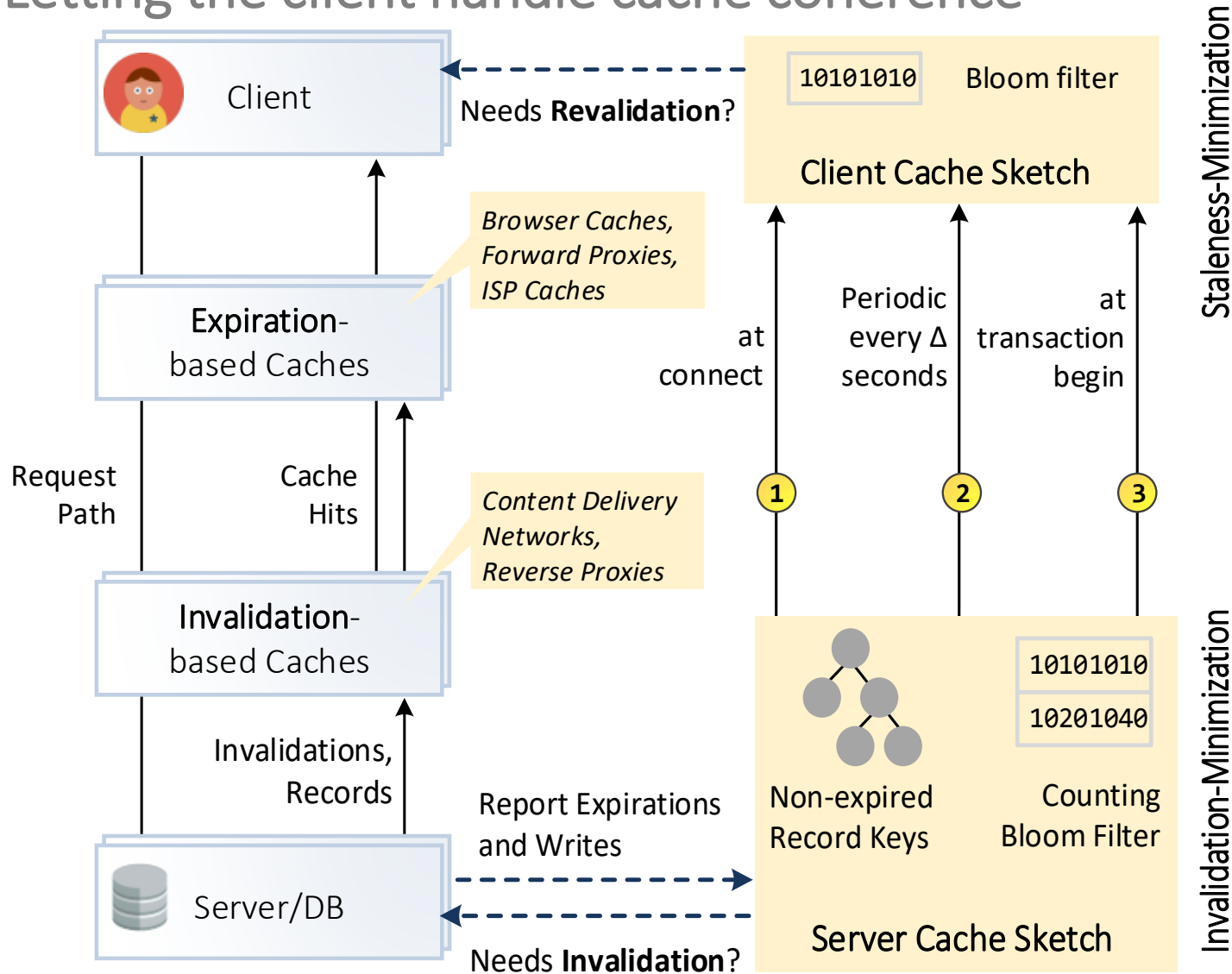
## Infrastructure





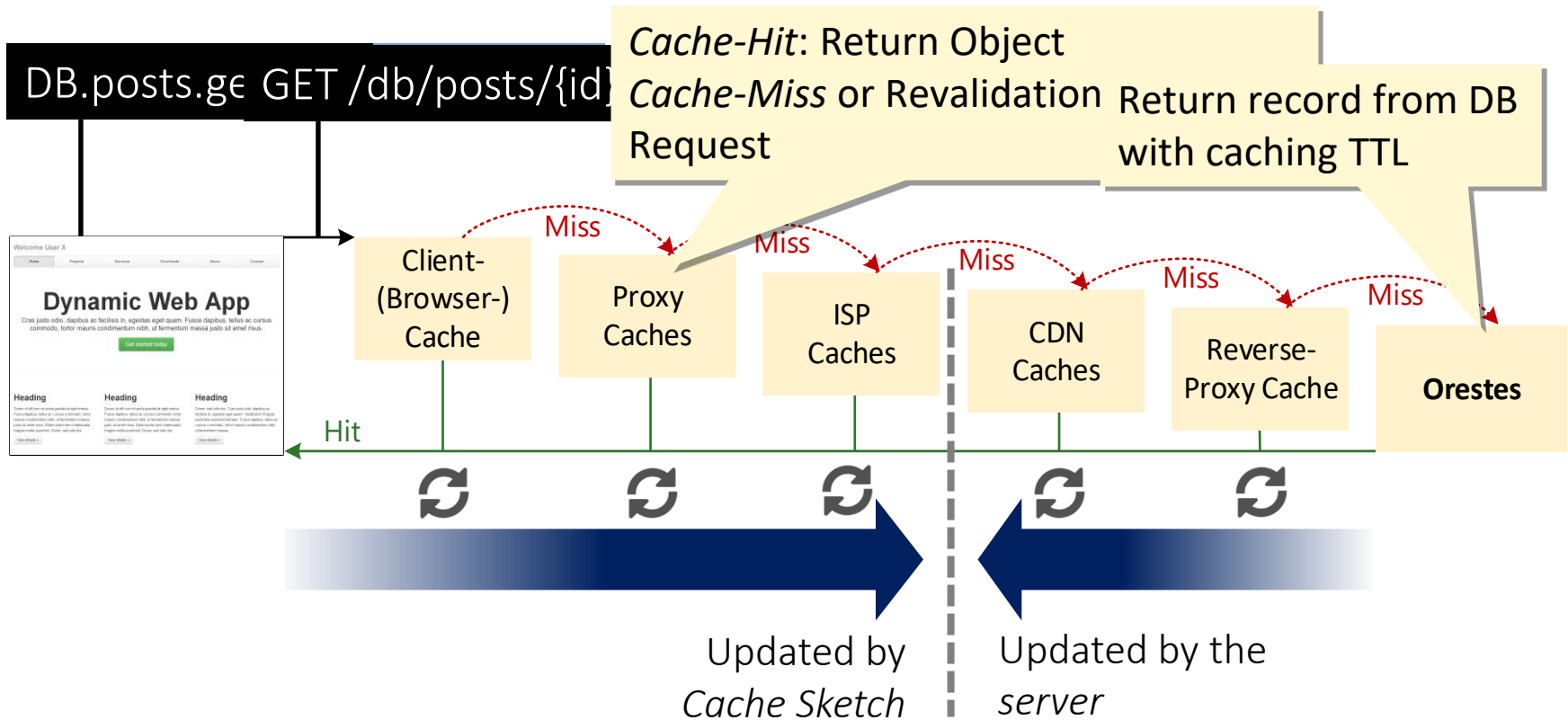
# The Cache Sketch approach

Letting the client handle cache coherence



# The End to End Path of Requests

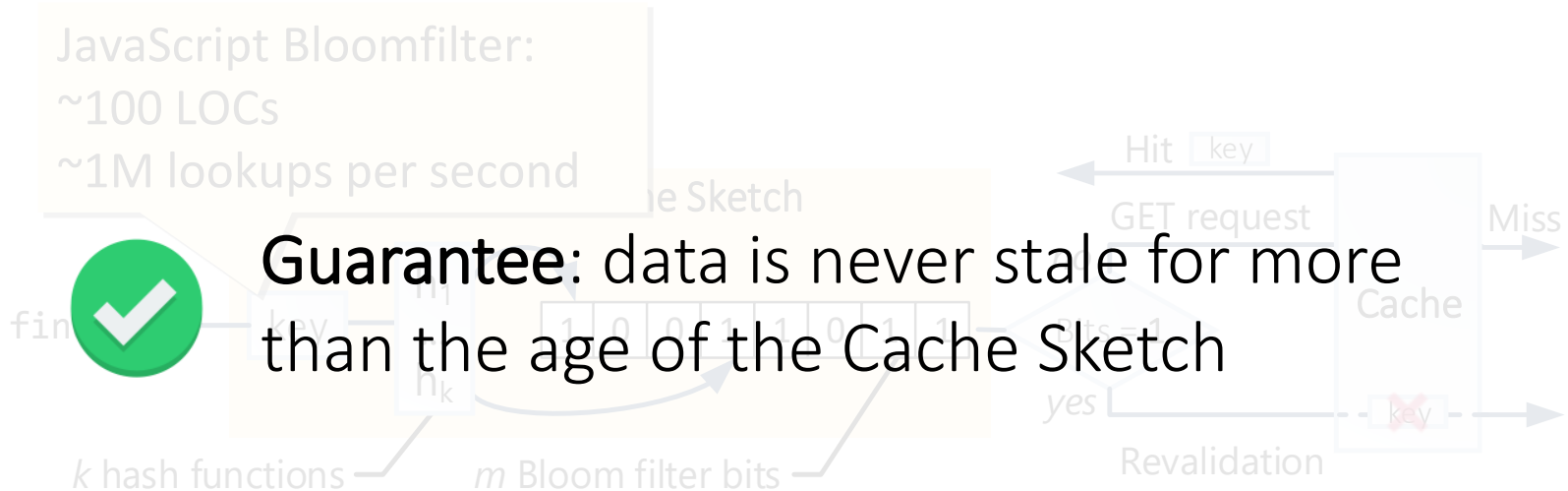
## The Caching Hierarchy



# The Client Cache Sketch

- ▶ Let  $c_t$  be the client Cache Sketch generated at time  $t$ , containing the key  $\mathbf{key}_x$  of every record  $x$  that was written before it expired in all caches, i.e. every  $x$  for which holds:

$$\exists r(x, t_r, TTL), w(x, t_w) : t_r + TTL > t > t_w > t_r$$



# The Server Cache Sketch

## Scalable Implementation

Add  $key_x$  if  $x$  unexpired  
and write occurred

Remove  $x$  from Blom  
filter when expired

Load Bloom filter

BITS → 0 1 0 0 1 0 1 0 1 1

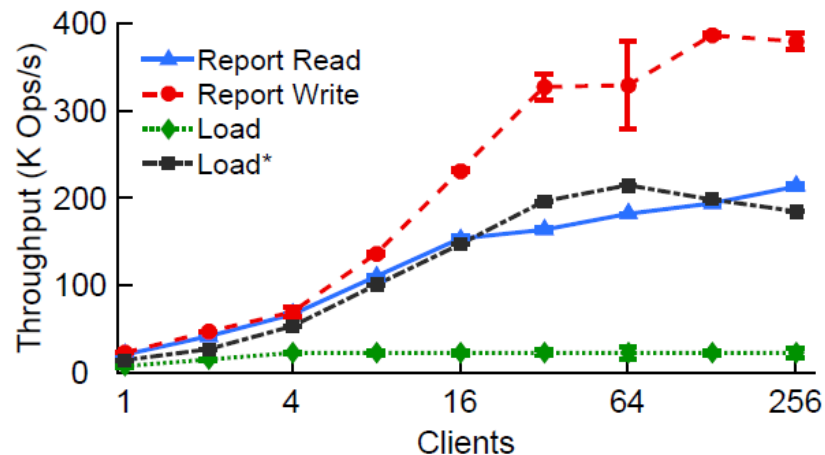
COUNTS → 0 2 0 0 1 0 3 0 1 1

KEY<sub>1</sub> → 1460832302

KEY<sub>6</sub> → 1460836199



Performance > 200k  
ops per second:



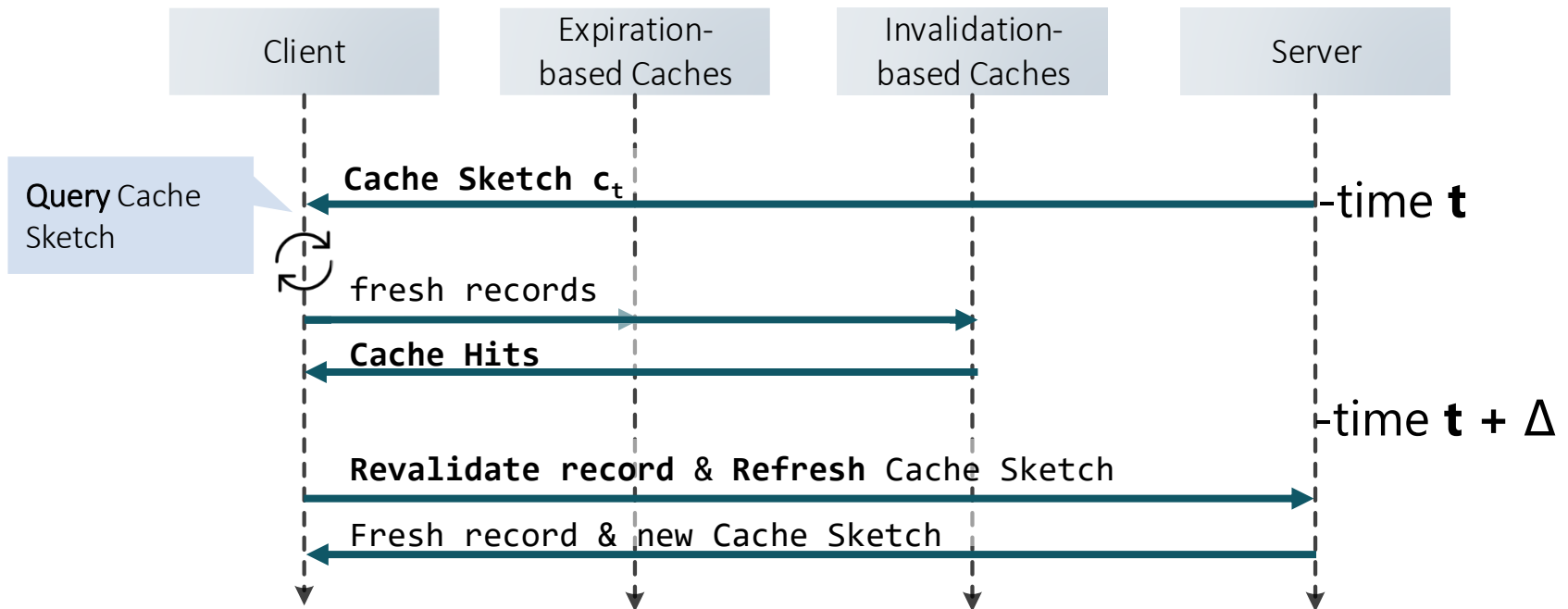
# 1 Faster Page Loads

- ▶ Clients load the Cache Sketch at connection
- ▶ Every non-stale cached record can be reused without degraded consistency

## 2 Faster CRUD Performance

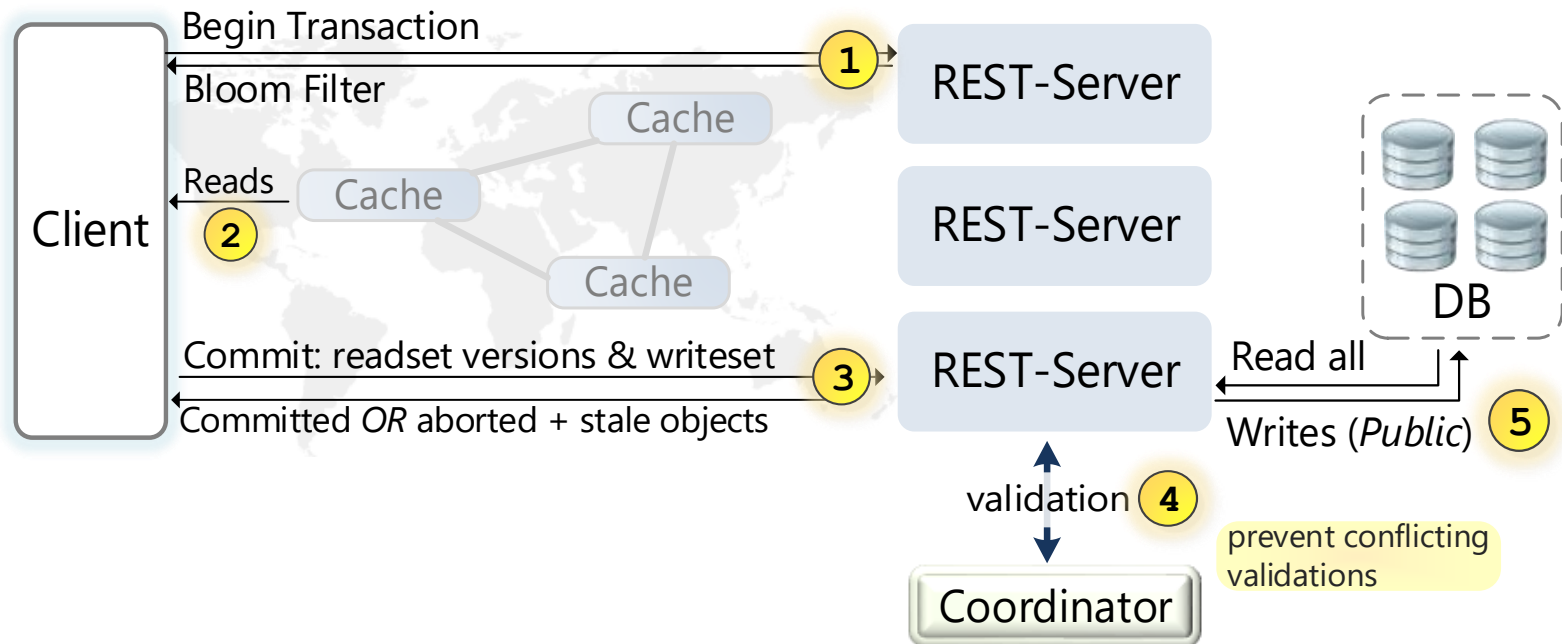
### ► Solution: $\Delta$ -Bounded Staleness

- Clients refresh the Cache Sketch so its age never exceeds  $\Delta$   
→ *Consistency guarantee:  $\Delta$ -atomicity*



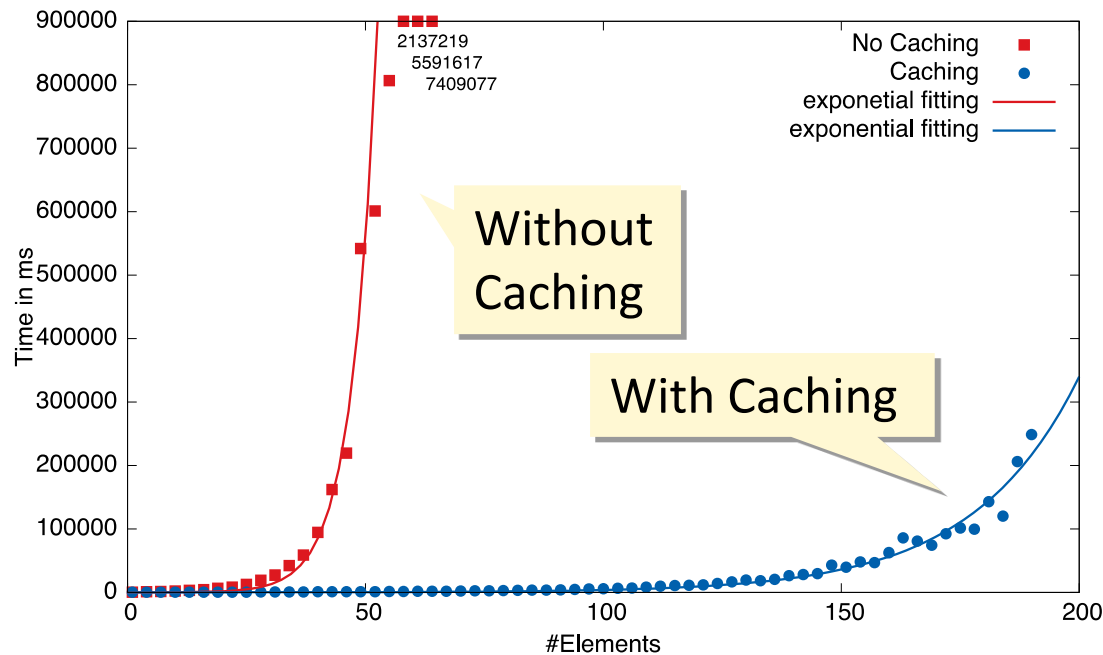
# 3 Scalable ACID Transactions

- ▶ Solution: **Conflict-Avoidant Optimistic Transactions**
  - Cache Sketch fetched with transaction begin
  - **Cached reads** → Shorter transaction duration → less aborts



# 3 Scalable ACID Transactions

- ▶ **Novelty:** ACID transactions on sharded DBs like MongoDB



- ▶ Current Work: **DESY** and **dCache** building a scalable namespace for their file system on this



# TTL Estimation

Determining the best TTL and cacheability

- ▶ **Problem:** if  $TTL \gg$  time to next write, then it is contained in Cache Sketch unnecessarily long
- ▶ **TTL Estimator:** finds „best“ TTL
- ▶ Trade-Off:

Shorter TTLs



- less invalidations
- less stale reads

Longer TTLs



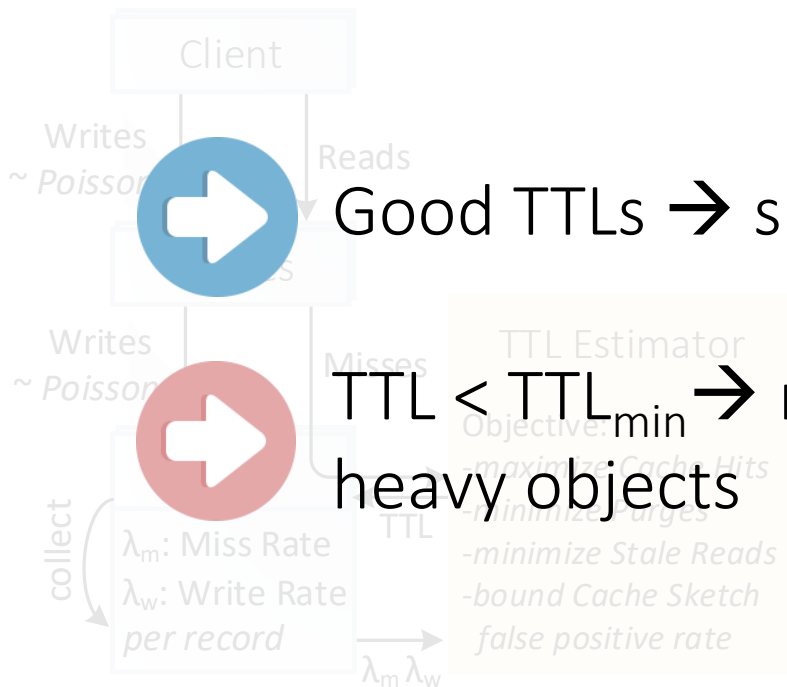
- Higher cache-hit rates
- more invalidations

# TTL Estimation

## Determining the best TTL

### Idea:

1. Estimate average time to next write  $E[T_w]$  for each record
2. Weight  $E[T_w]$  using the cache miss rate

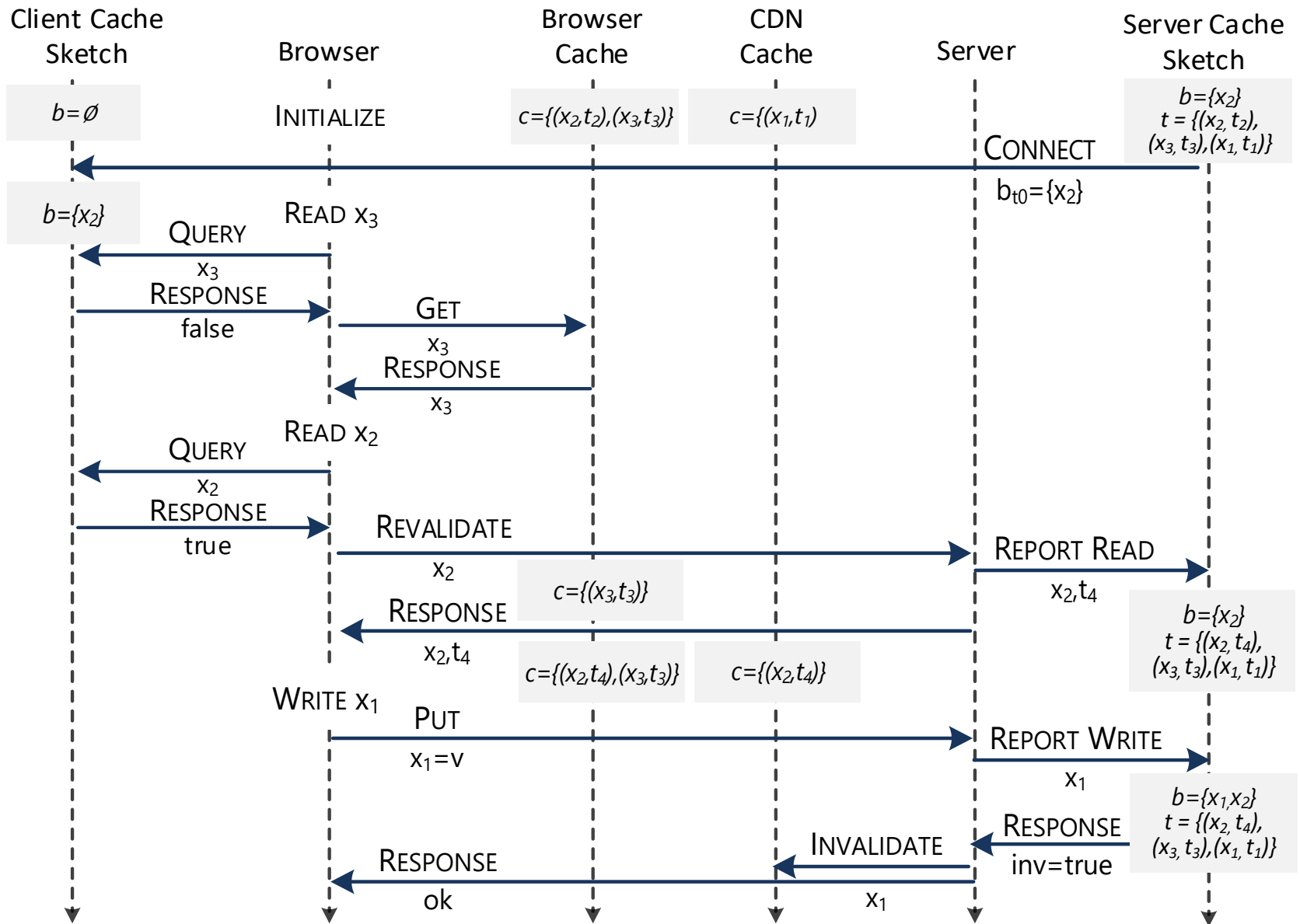


Good TTLs → small Bloom filter

$TTL < TTL_{min} \rightarrow$  no caching of write-heavy objects





# End-to-End Example

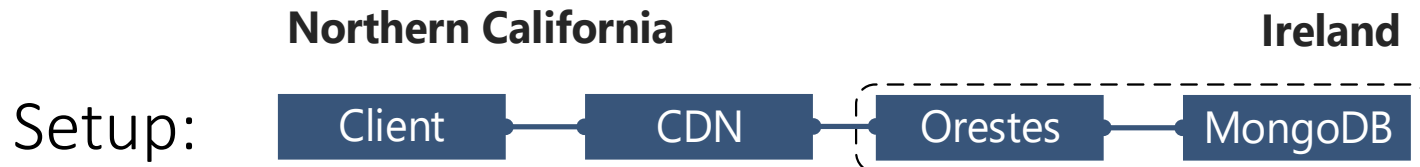


# Consistency

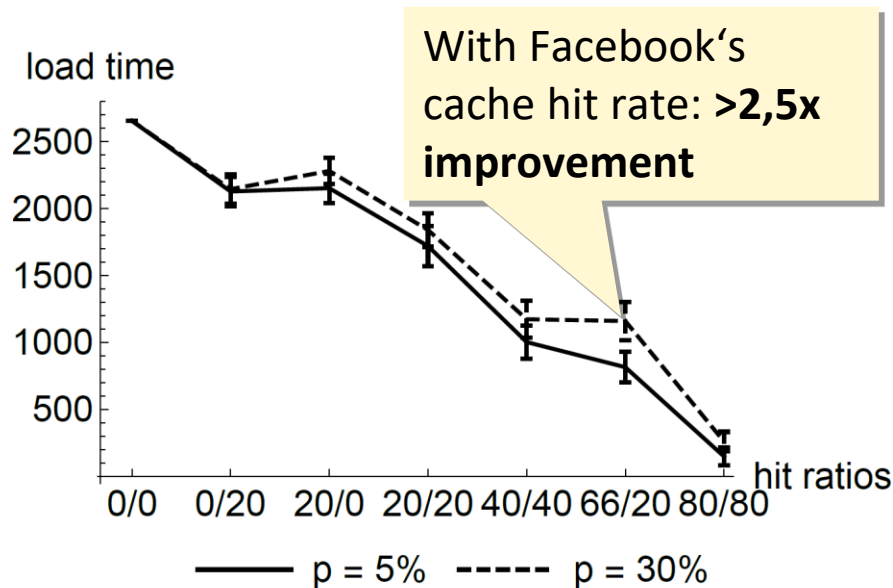
What are the guarantees?

Consistency Level	How	
<b><math>\Delta</math>-atomicity</b> (staleness never exceeds $\Delta$ seconds)	Controlled by age of Cache Sketch	Always 
<b>Monotonic Writes</b>	Guaranteed by database	
<b>Read-Your-Writes</b> and <b>Monotonic Reads</b>	Cache written data and most recent read-versions in client	
<b>Causal Consistency</b>	If read timestamp is older than Cache Sketch it is given, else revalidate	Opt-in 
<b>Strong Consistency</b> (Linearizability)	Explicit revalidation (cache miss at all levels)	

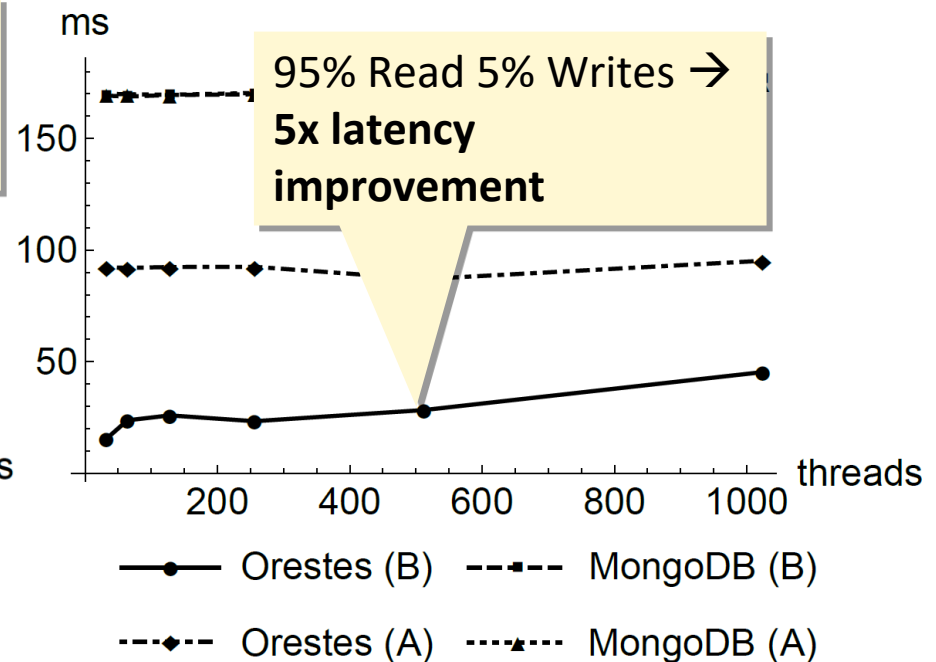
# Performance



Page load times with **cached initialization** (simulation):

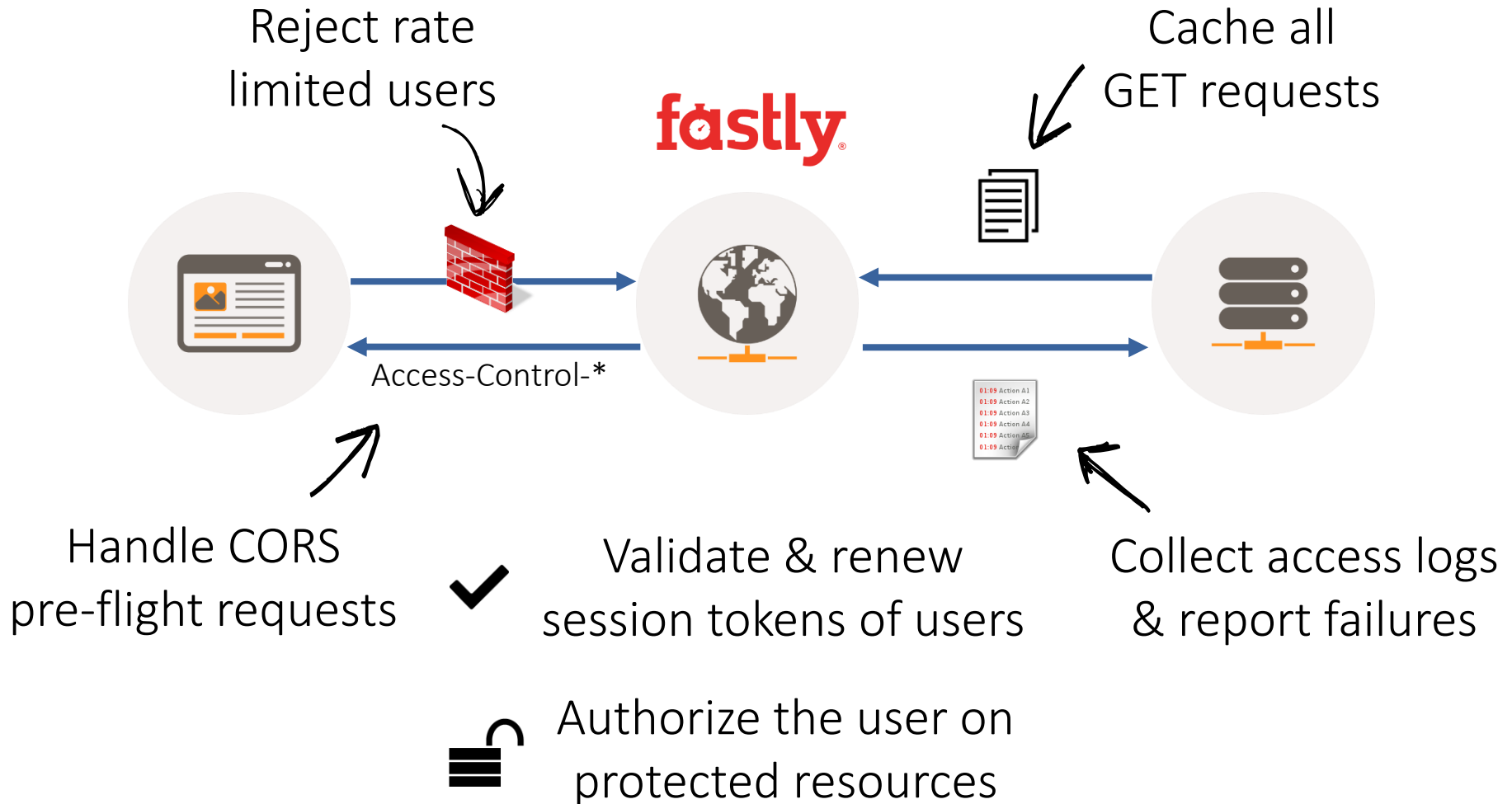


Average Latency for YCSB Workloads A and B (real):



# Varnish and Fastly

What we do on the edge



# The Cache Sketch

## Summary

### Static Data ✓



Immutability ideal for  
static web caching:  
`max-age=31557600`

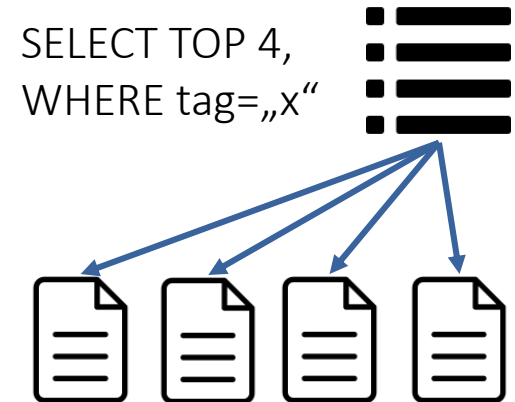
### Mutable Objects ✓

```
{  
  "id": "/db/Todo/b5d9bef9-  
6c1f-46a5-...",  
  "version": 1,  
  "acl": null,  
  "listId": "7b92c069-...",  
  "name": "Test",  
  "activities": [],  
  "active": true,  
  "done": false  
}
```

Cache Sketch for browser  
cache, proxies and ISP  
caches

Invalidations for CDNs and  
reverse proxies

### Queries/Aggregates ?



How to do this?

# Continuous Query Matching

Generalizing the Cache Sketch to query results

**Main challenge:** when to invalidate?

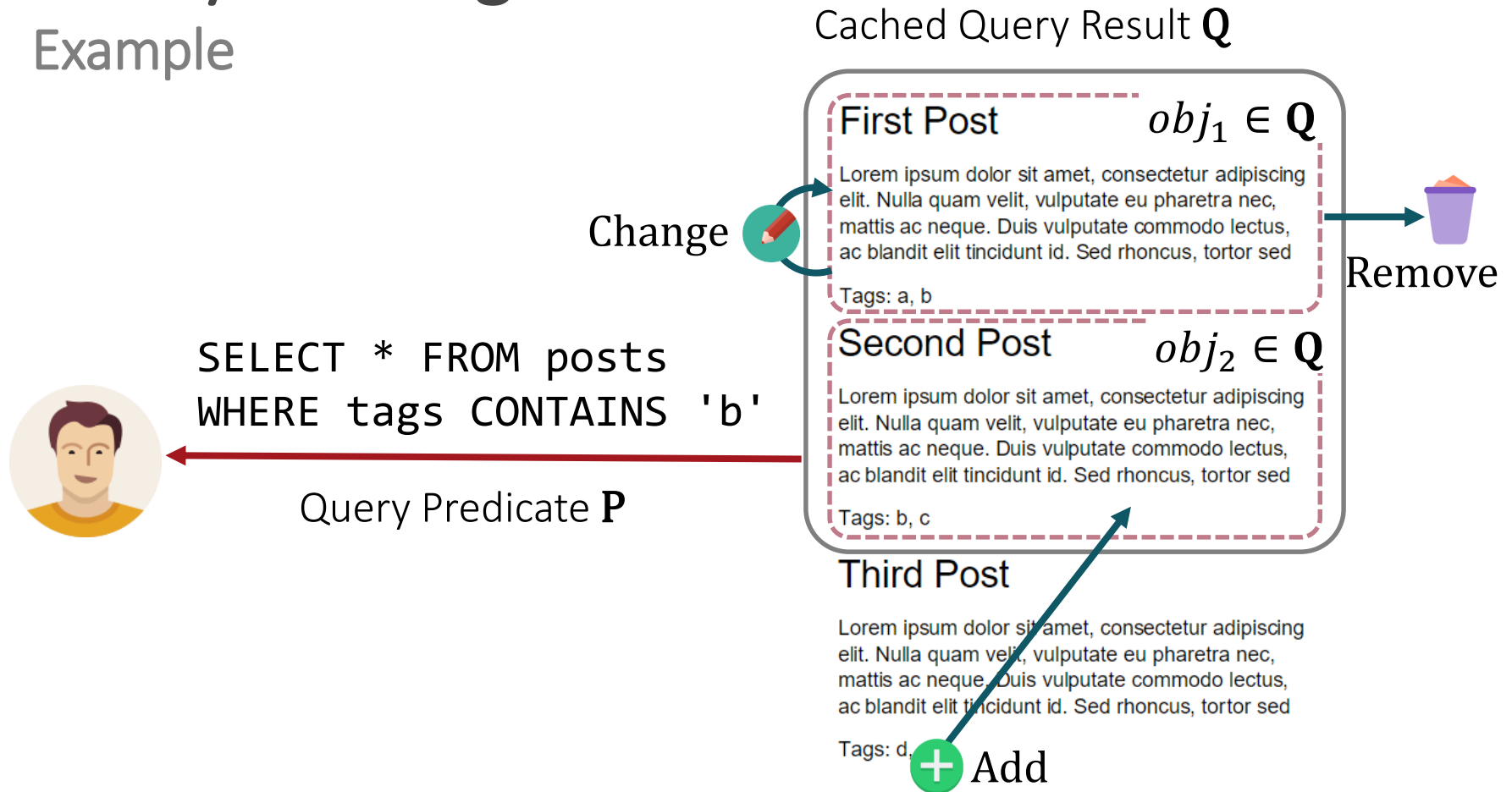
- **Objects:** for every update and delete
- **Queries:** as soon as the query result changes

How to detect query **result**  
**changes in real-time?**



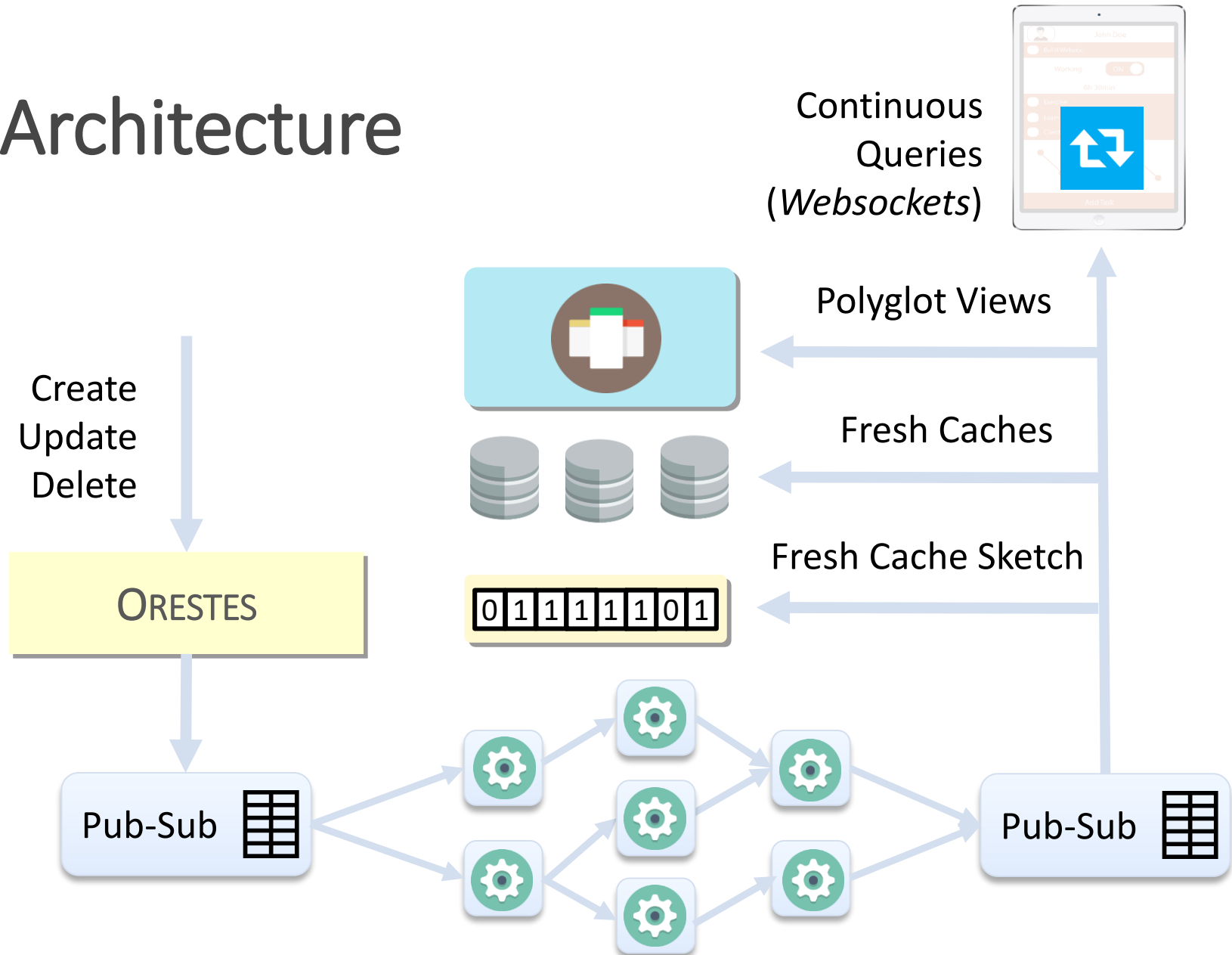
# Query Caching

## Example



- ▶ **Add, Change, Remove** all entail an invalidation and addition to the cache sketch

# Architecture



# InvaliDB

## Matching on Apache Storm

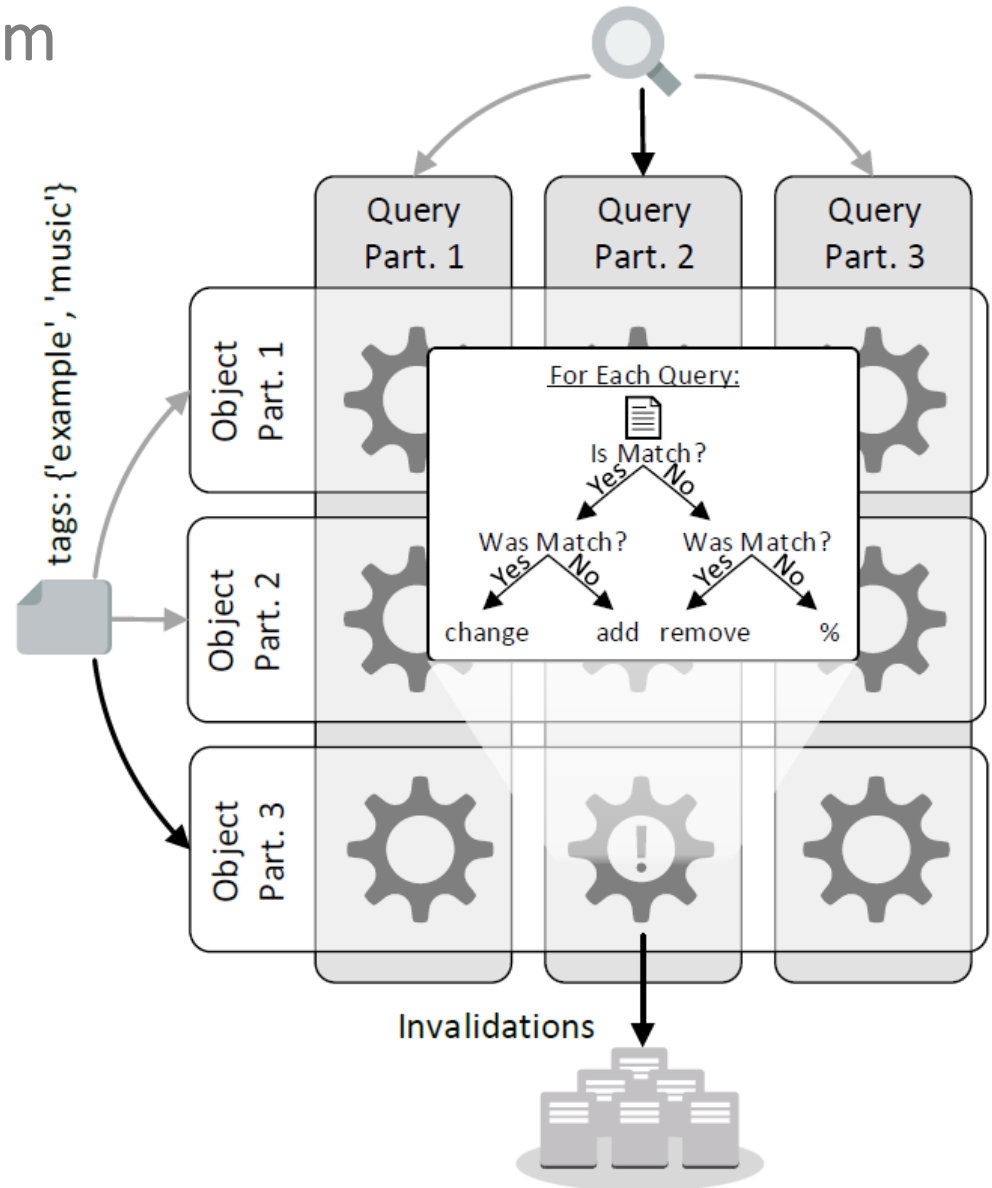
Apache Storm:

- „Hadoop of Real-Time“
- Low-Latency Stream Processing
- Custom Java-based Topologies

InvaliDB goals:

- Scalability, Elasticity, Low latency, Fault-tolerance

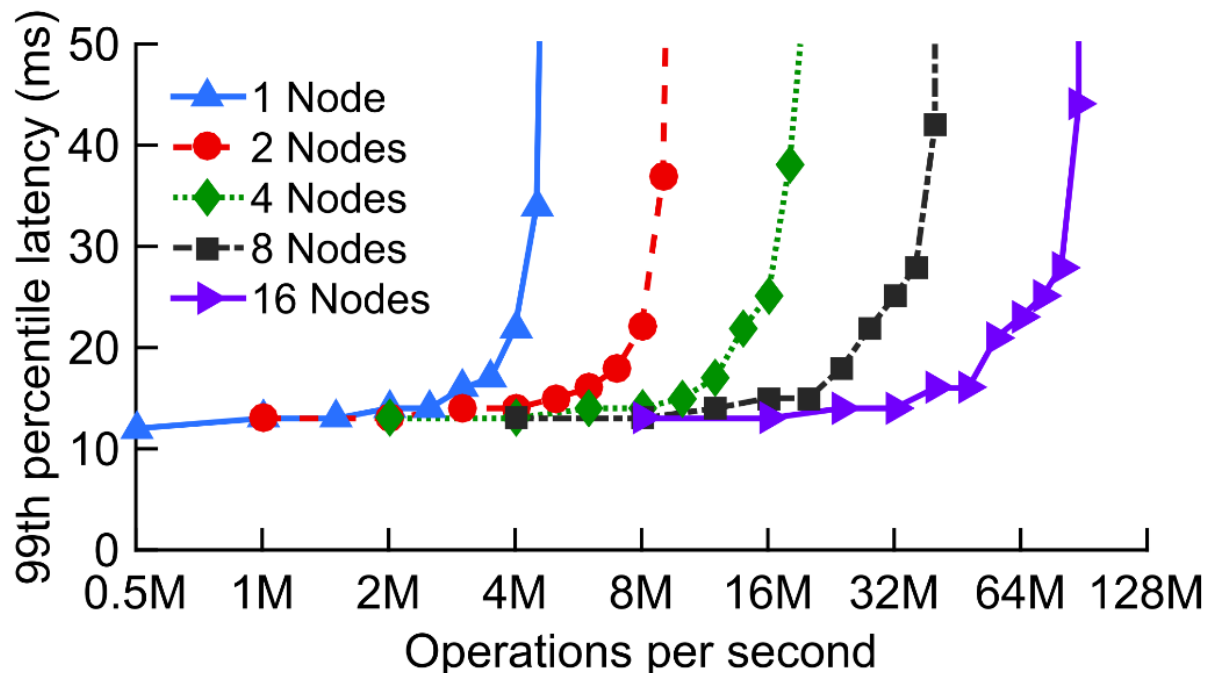
`SELECT * FROM posts WHERE tags CONTAINS 'example'`



# Query Matching Performance

Latency of detecting invalidations

- ▶ Latency mostly < 15ms, scales linearly w.r.t. number of servers and number of tables



# Learning Representations

## Determining Optimal TTLs and Cacheability

**Setting:** query results can either be represented as references (id-list) or full results (object-lists)

Id-Lists

$[id_1, id_2, id_3]$

*Less Invalidations*

Object-Lists

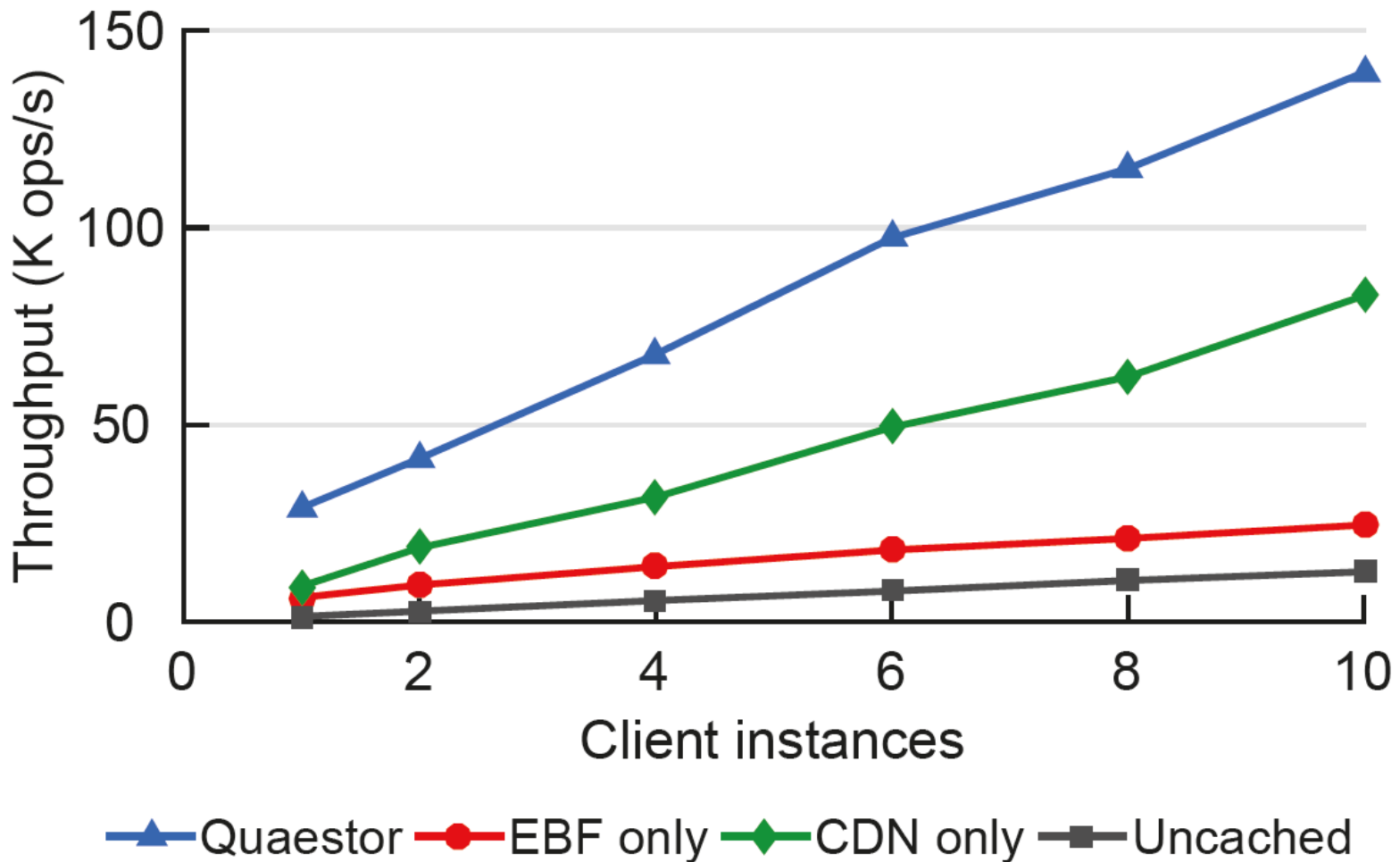
$[\{id: 1, val: 'a'\}, \{id: 2, val: 'b'\}, \{id: 3, val: 'c'\}]$

*Less Round-Trips*

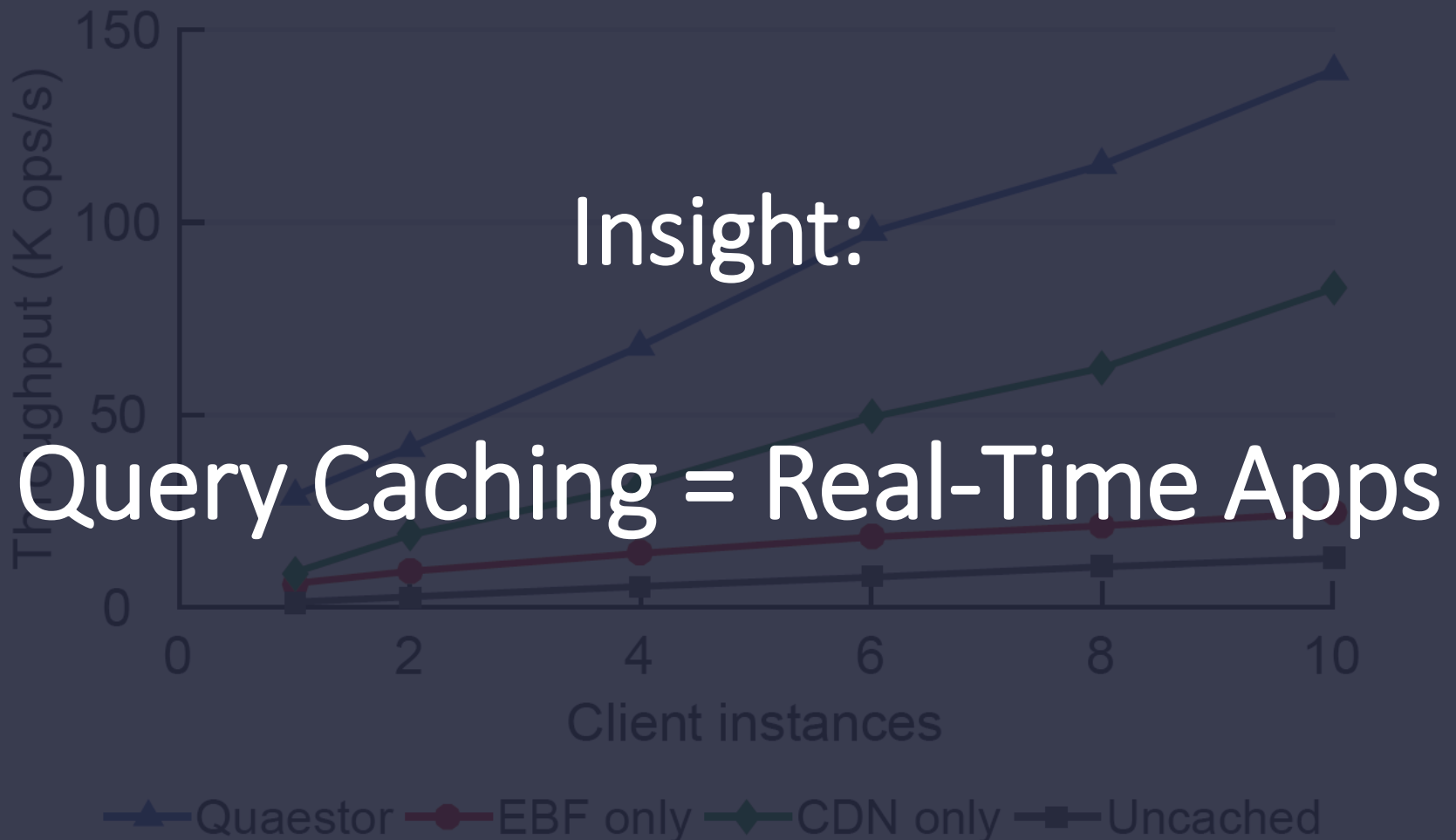
**Approach:** Cost-based decision model that weighs expected round-trips vs expected invalidations

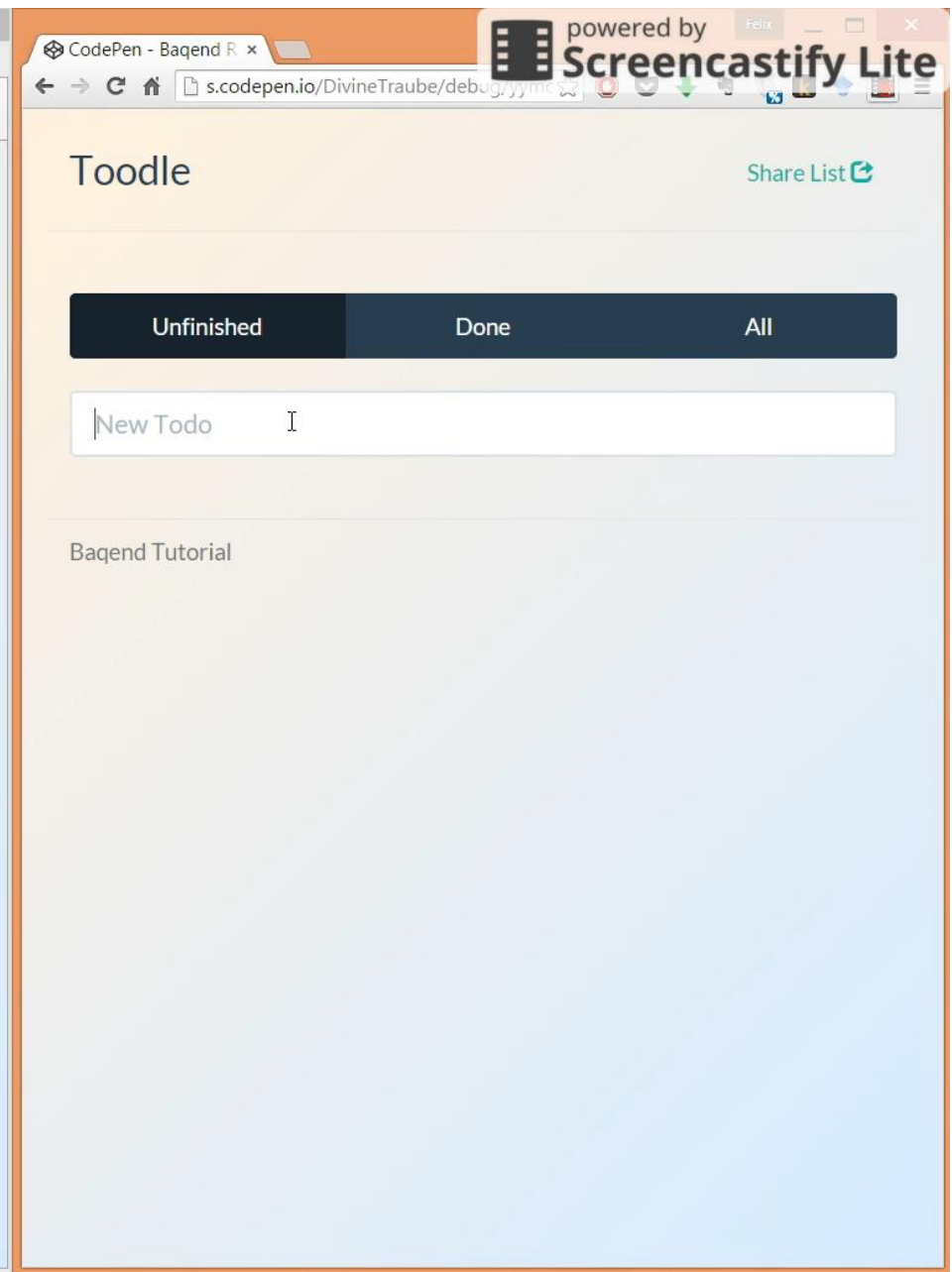
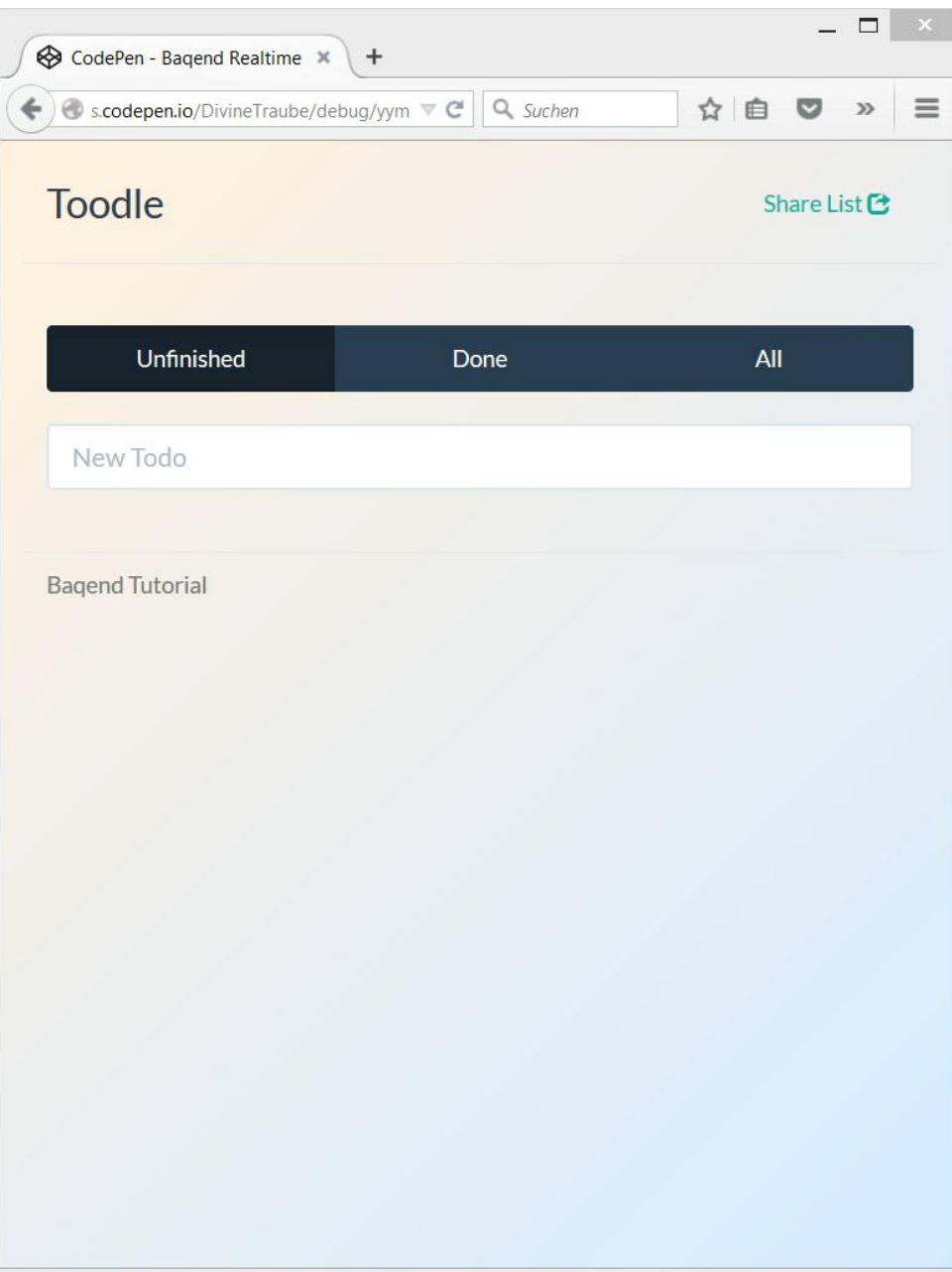
**Ongoing Research:** Reinforcement learning of decisions

# What is the impact of query caching?



# What is the impact of query caching?



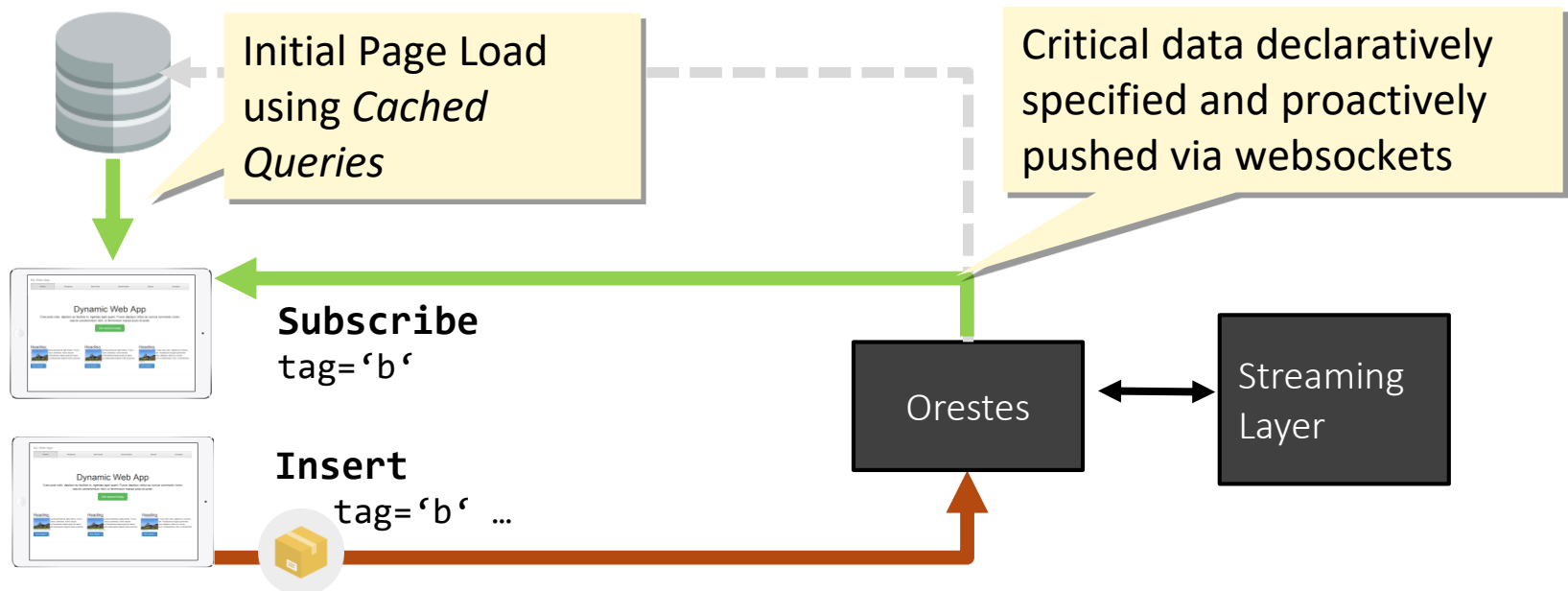




# Continuous Queries

## Complementing Cached Queries

- ▶ Same streaming architecture can similarly notify the browser about query result changes
- ▶ **Application Pattern:**



# Continuous Query API

## Subscribing to database updates

dpa News Demo Baepend

Filter Anwenden ☒ Desktop Benachrichtigungen

Beispiel-Filter

- Wildcard Suche
- Hat eine Summary
- Neuer als
- Tag Filter
- Hat Kategorie

☒ Nur echte Nachrichten anzeigen

**var stream = DB.News.find().stream();**  
**stream.on("add", onNews);**  
**stream.on("remove", onRemove);**

**Panzerbauer KMW und Nexter bleiben zuversichtlich für Fusion** 15:33:28  
Das Wirtschaftsministerium nimmt die Fusionspläne der Panzerhersteller KMW und Nexter genauer unter die Lupe. Doch die Unternehmen sehen ihren Zeitplan nicht in Gefahr. München/Versailles (dpa) - Die Panzerbauer Krauss-Maffei Wegmann (KMW) und Nexter bleiben trotz einer vertieften Prüfung durch die Bundesregierung zuversichtlich für ihre Fusionspläne. «Wir glauben nicht, dass es zu Verzögerungen kommen wird», sagte ein KMW-Sprecher am Donnerstag in München. Auch der französische K...  
Zusammenschl...  
völlig normal»,...  
[Mehr >](#)

**Bruno Ganz**  
Keine Zusammenfassung vorhanden.  
[Mehr >](#)

**Bulgarien stimmt Vorschlag zur Quotenregelung für Flüchtlinge zu** 15:30:48  
Keine Zusammenfassung vorhanden.  
[Mehr >](#)

**Gewerkschaften und Politik wollen Mindestlohn auch für Flüchtlinge** 15:26:52  
Asylbewerber sollen in den Arbeitsmarkt integriert werden. Nur wie? Über den richtigen Weg gibt es unterschiedliche Vorstellungen. Weitere Sonderregeln werden allerdings skeptisch gesehen. Berlin (dpa) - Gewerkschaften und Politik warnen davor, bei Jobs für Flüchtlinge Ausnahmen vom gesetzlichen Mindestlohn zuzulassen. Entsprechende Vorschläge stoßen auf wenig Zustimmung - die Gewerkschaften befürchten, dass Asylbewerber dann als billige Arbeitskräfte ausgenutzt werden könnten. Sowohl die...

03:32:12 PM 03:36:27 PM 03:36:42 PM

# Summary



- ▶ **Orestes**: DB-independent **Backend-as-a-Service**
- ▶ **Cache Sketch Approach**:
  - Client decides when to **revalidate**, server **invalidates** CDN
  - Cache Sketch = **Bloom filter** of stale IDs
  - Compatible with end-to-end ACID **transactions**
  - Query change detection in **real-time**

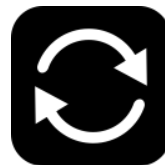


HTTP  
Caching



0	1	0	0	1
0	1	0	1	1
1	1	0	0	0
0	0	0	1	1

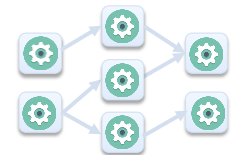
Cache  
Sketch



Invali-  
dations



TTL  
Estimation



RT Query  
Matching

## Introduction

## Main Part

## Conclusions



Web Performance:  
State of the Art



Cache Sketch:  
Research Approach



Using Web Caching in  
Applications

Orestes Caching  
Technology as a  
**Backend-as-a-Service**



# BaQend

Build faster Apps **faster.**



# Page-Load Times

## What impact does caching have in practice?

5,7s  
4,7s

### 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

Messerattaken auf Israelis, Krawalle auf dem Tempelberg, Scharmützel im Gassengewirr

### Wirtschaft



11. November 2014 07:15 Uhr  
HONORARBERATUNG

#### Guter Rat zur Geldanlage ist selten

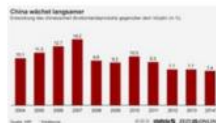
Honorarberatung ist in Deutschland endlich gesetzlich geregelt. Doch gibt es kaum Honorarberater. Und gut qualifizierte noch viel weniger.



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.

# Welcome to Baqend Cloud

Your Baqend account has been created!  
Have a look at these resources to help you get started quickly.

[▶ Tutorial](#)[⚡ Getting Started](#)[📄 Documentation](#)[</> API Docs](#)**bbq**[↑ Enter The App](#)**3,889**

Requests

**19,1** MB

Outgoing Data

**4,1** MB

DB Space

**9.8** %

CDN Cache Hit ratio

Pay as you go

**0** €[Set Limit](#)

Current Plan

**50** €

Medium

[Change Plan](#)

App Status

● **Running****to do**[↑ Enter The App](#)**2,405**

Requests

**2,9** MB

Outgoing Data

**114,8** KB

DB Space

**43.8** %

CDN Cache Hit ratio

Pay as you go

**0** €[Set Limit](#)

Current Plan

**500** €

Business

[Change Plan](#)

App Status

● **Running**[Need help?](#)

## Live Demo: Using Caching in Practice

Want to try Baqend?



Free **Baqend Cloud**  
instance



Download **Community**  
**Edition**





```
graph TD; A((Thank you Questions?)) --- B((baqend.com)); A --- C((fg@baqend.com)); A --- D((Twitter: @baqendcom));
```

Thank you  
Questions?

Twitter:  
@baqendcom

baqend.com

fg@  
baqend.com