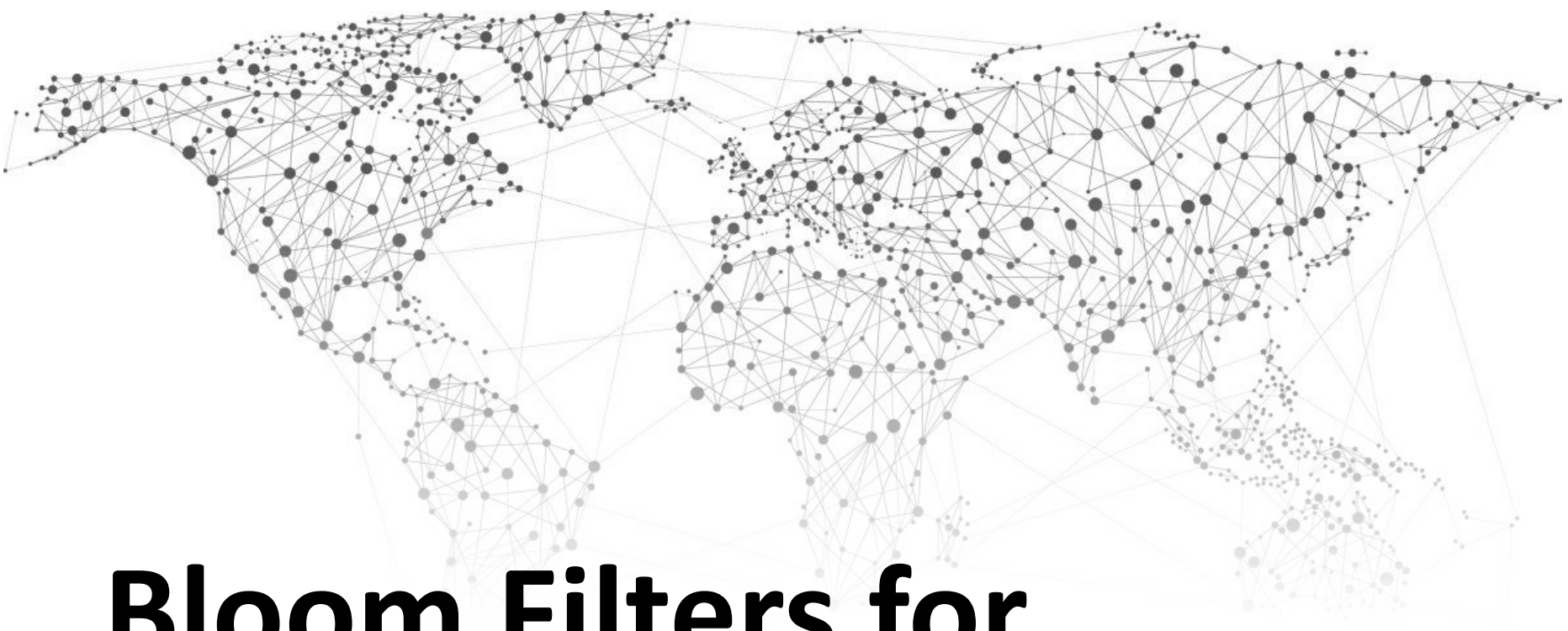




Bloom Filters for Web Caching

Felix Gessert

Felix.gessert@baqend.com



[@baqendcom](https://twitter.com/baqendcom)

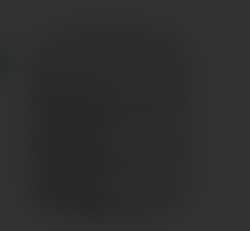
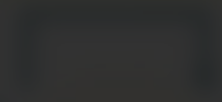
Backend Abstracts

Backend Abstracts

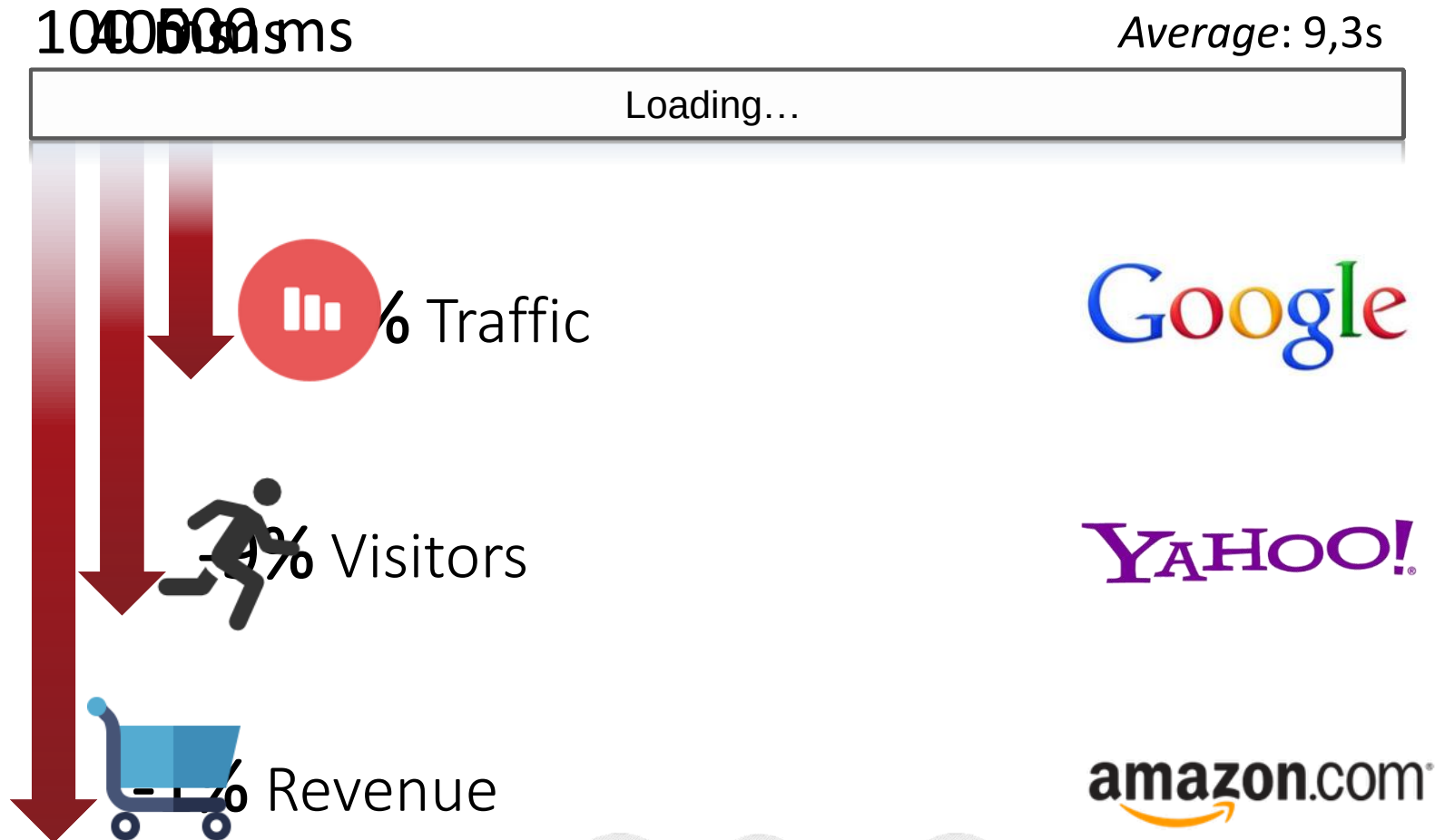
Backend Abstracts is a web application that provides a platform for users to create and manage their own abstracts. The application is built using a combination of Python, JavaScript, and CSS. It features a user interface that is both intuitive and easy to use, allowing users to quickly and easily create and manage their abstracts. The application is also highly scalable, allowing it to handle a large number of users and abstracts simultaneously.

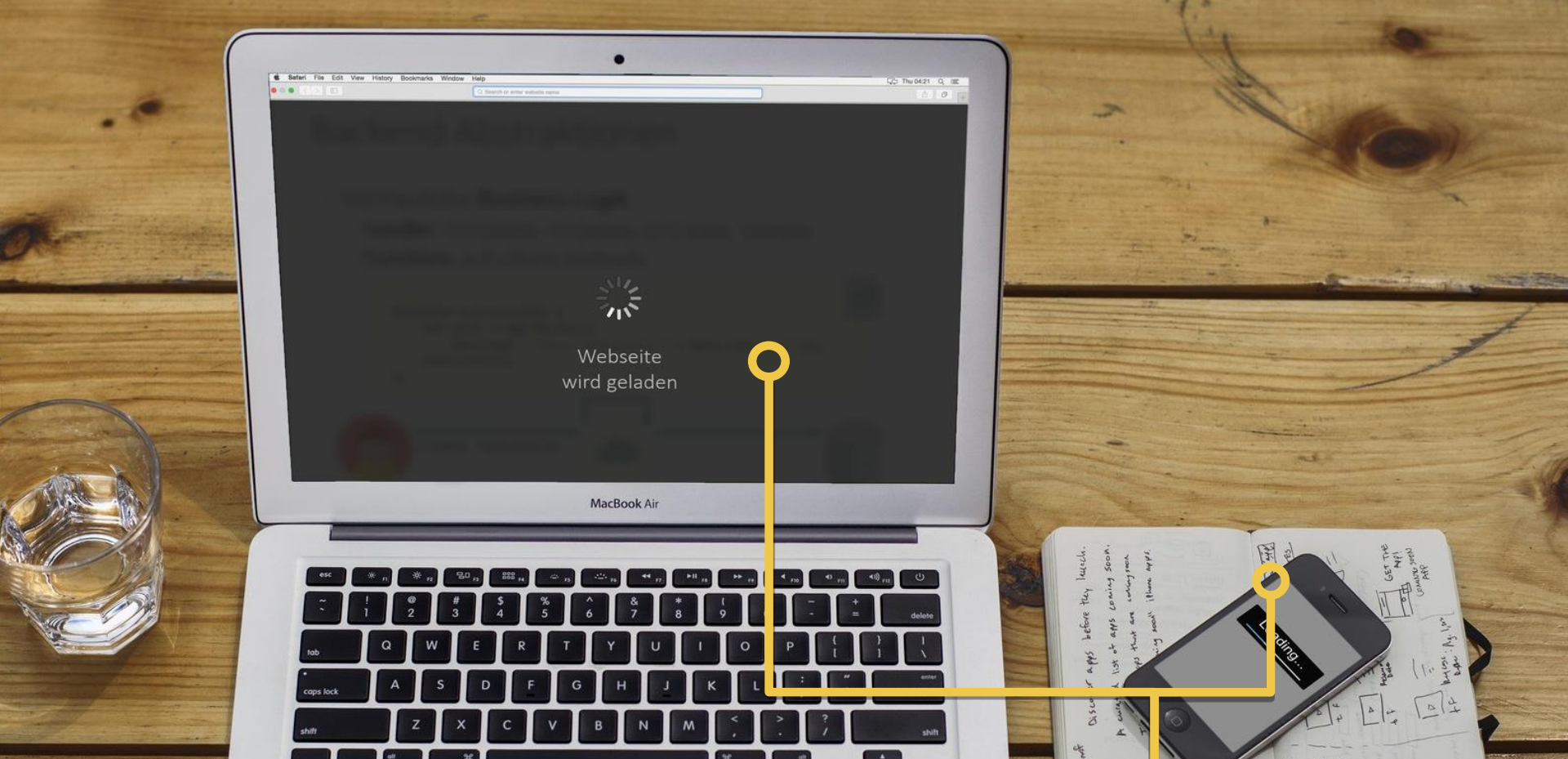


Presentation
is loading



The Latency Problem





If perceived speed is such an
import factor for **business**

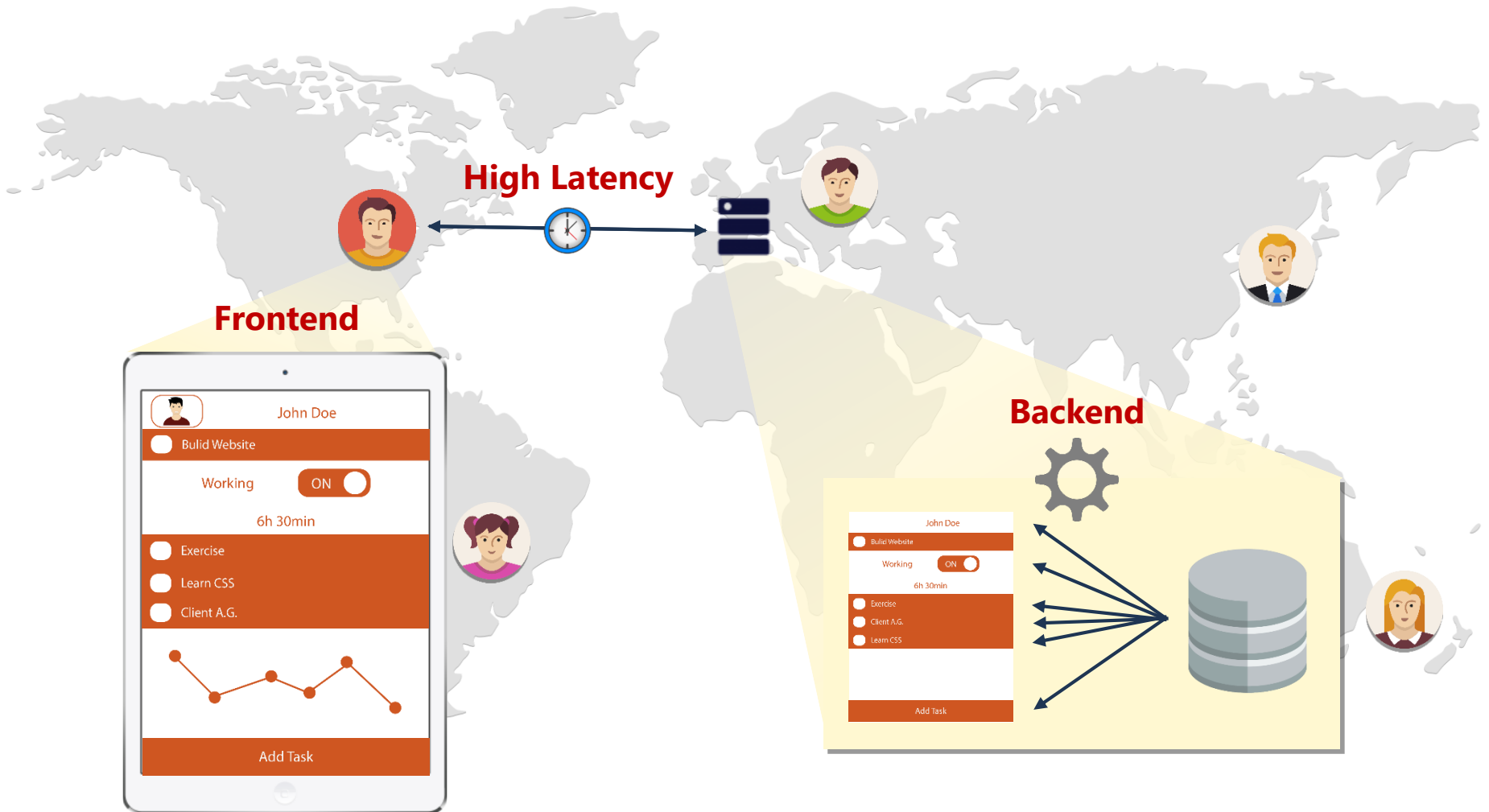


...what causes slow page load times?

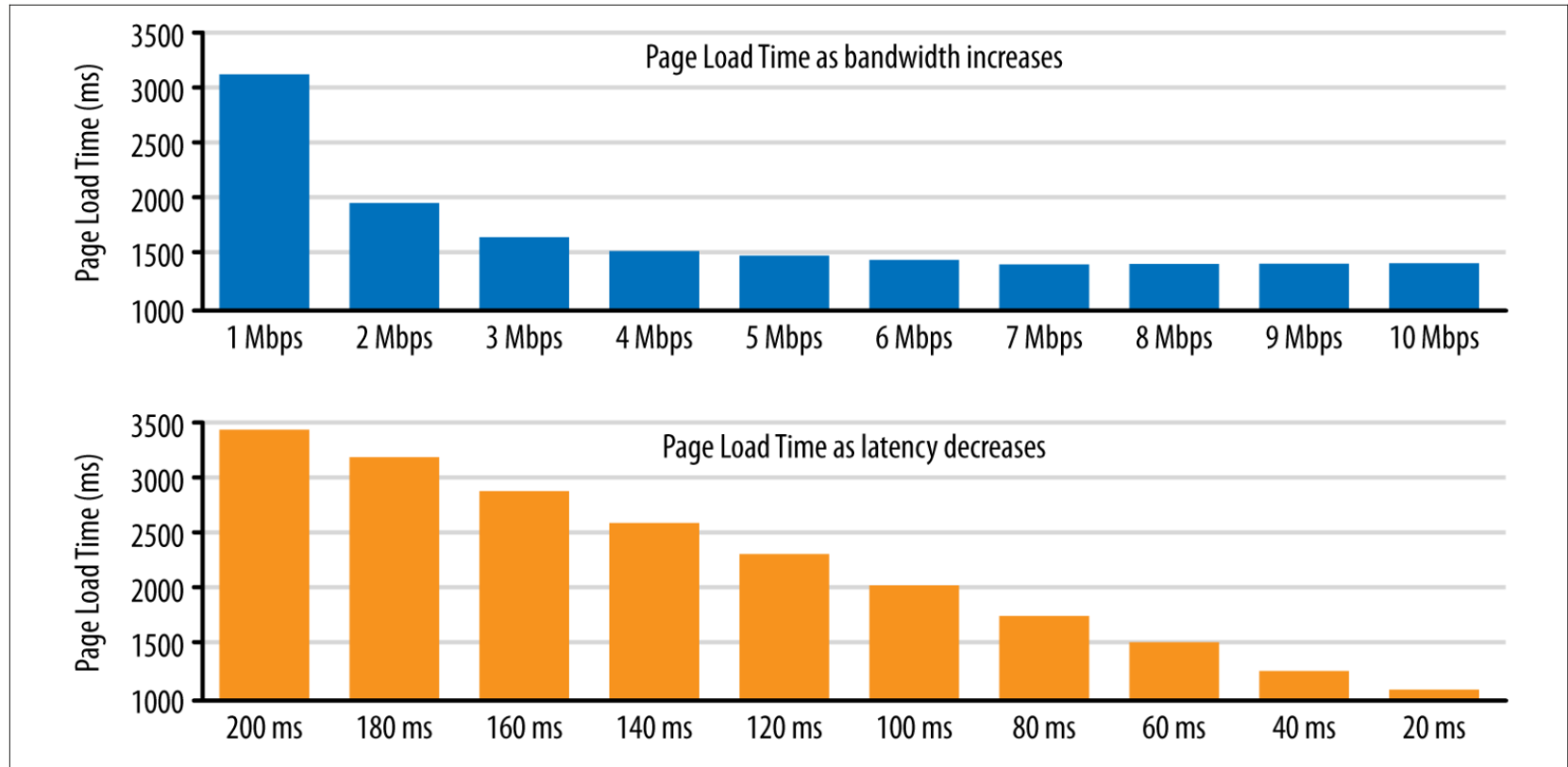


The Problem

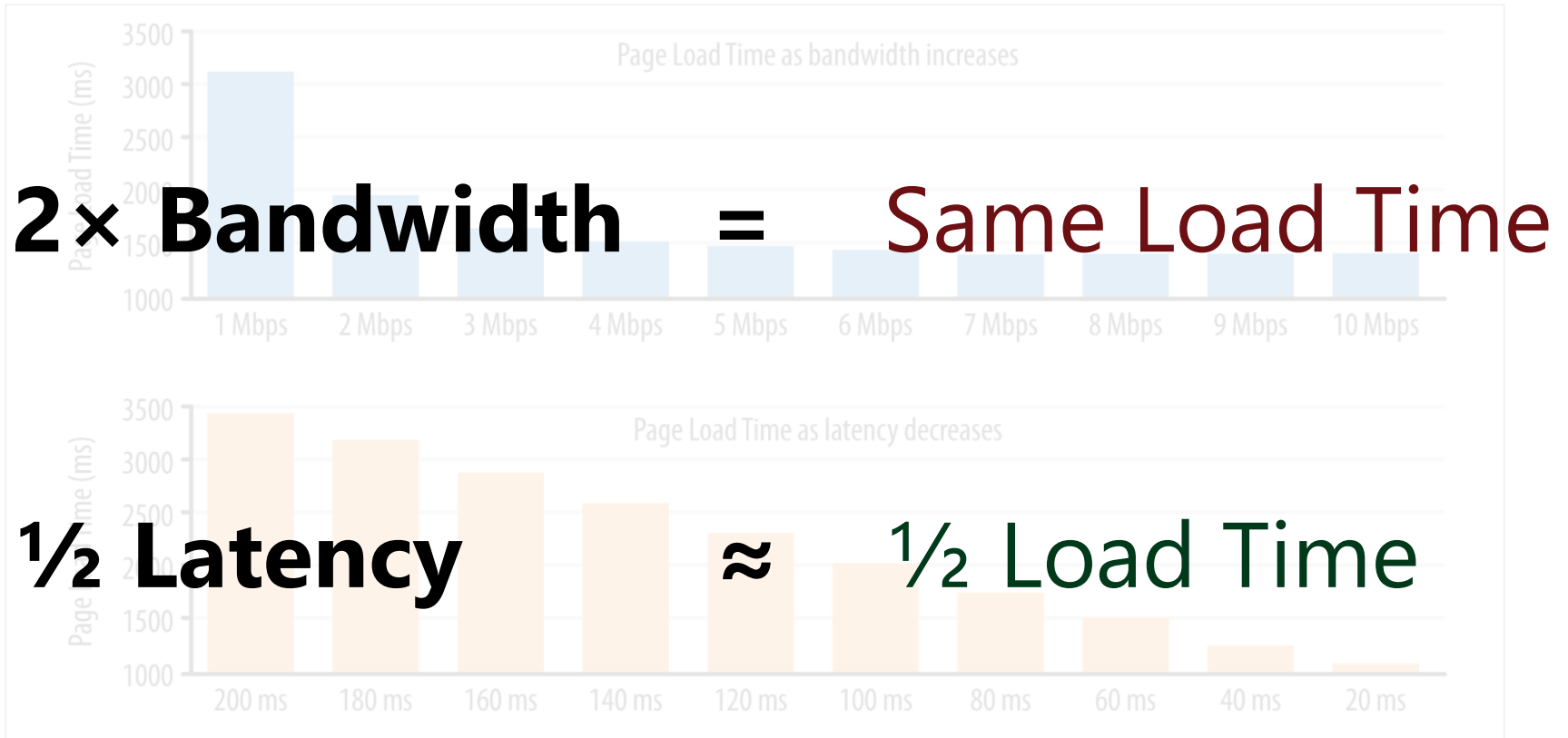
Three Bottlenecks: Latency, Backend & Frontend



Network Latency: Impact

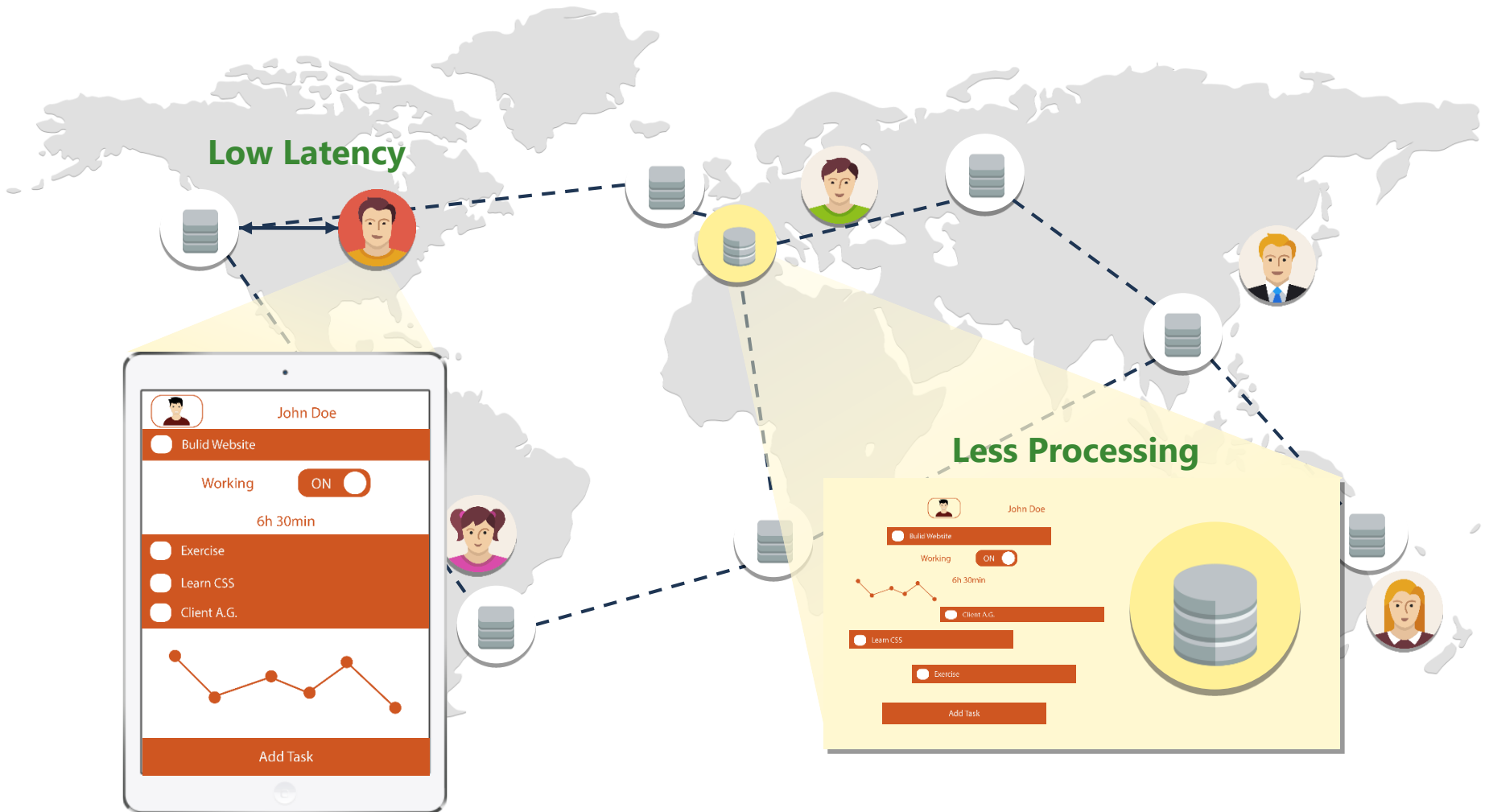


Network Latency: Impact



Goal: Low-Latency for Dynamic Content

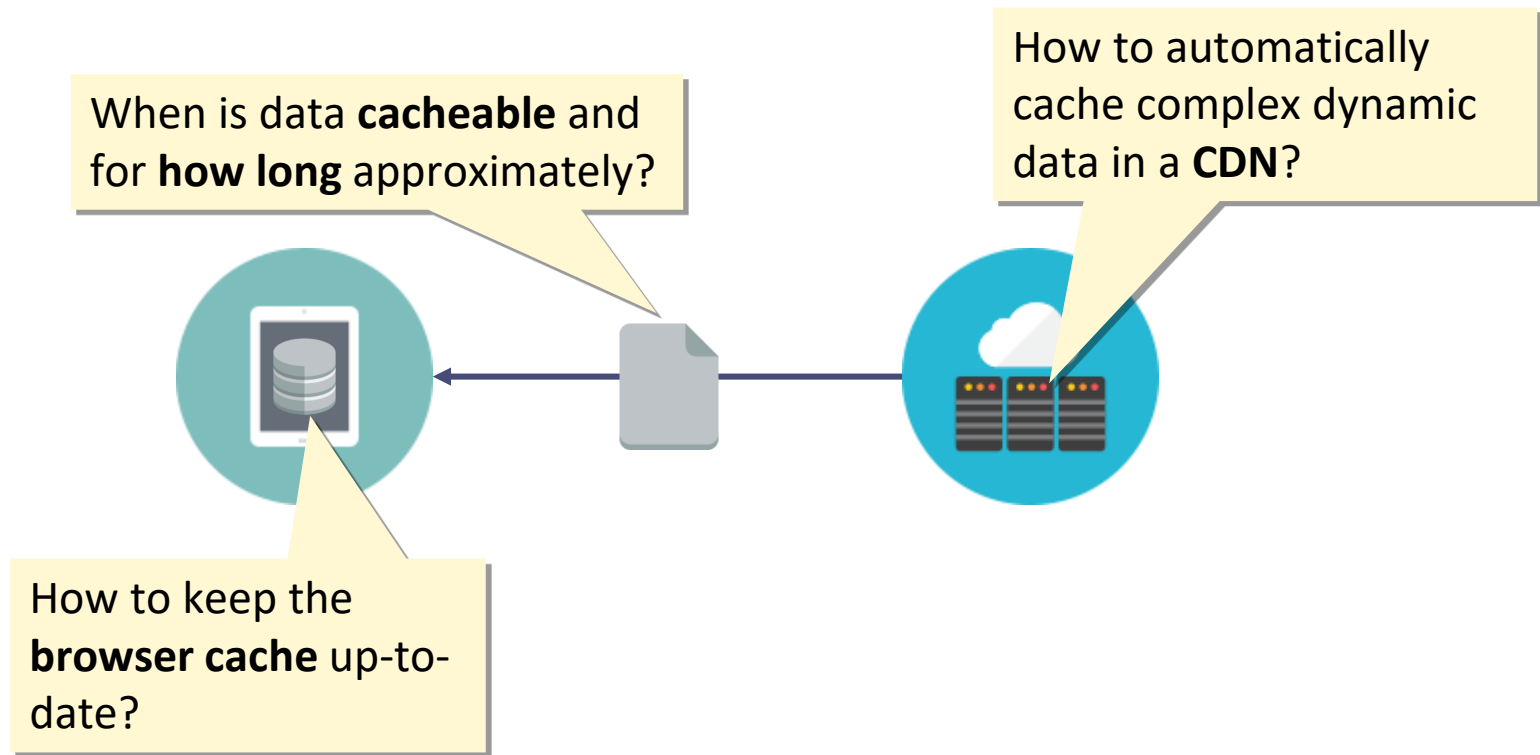
By Serving Data from Ubiquitous Web Caches



Our Approach

Caching Dynamic Data with Bloom filters

Idea: use *HTTP Caching* for **dynamic data** (e.g. queries)

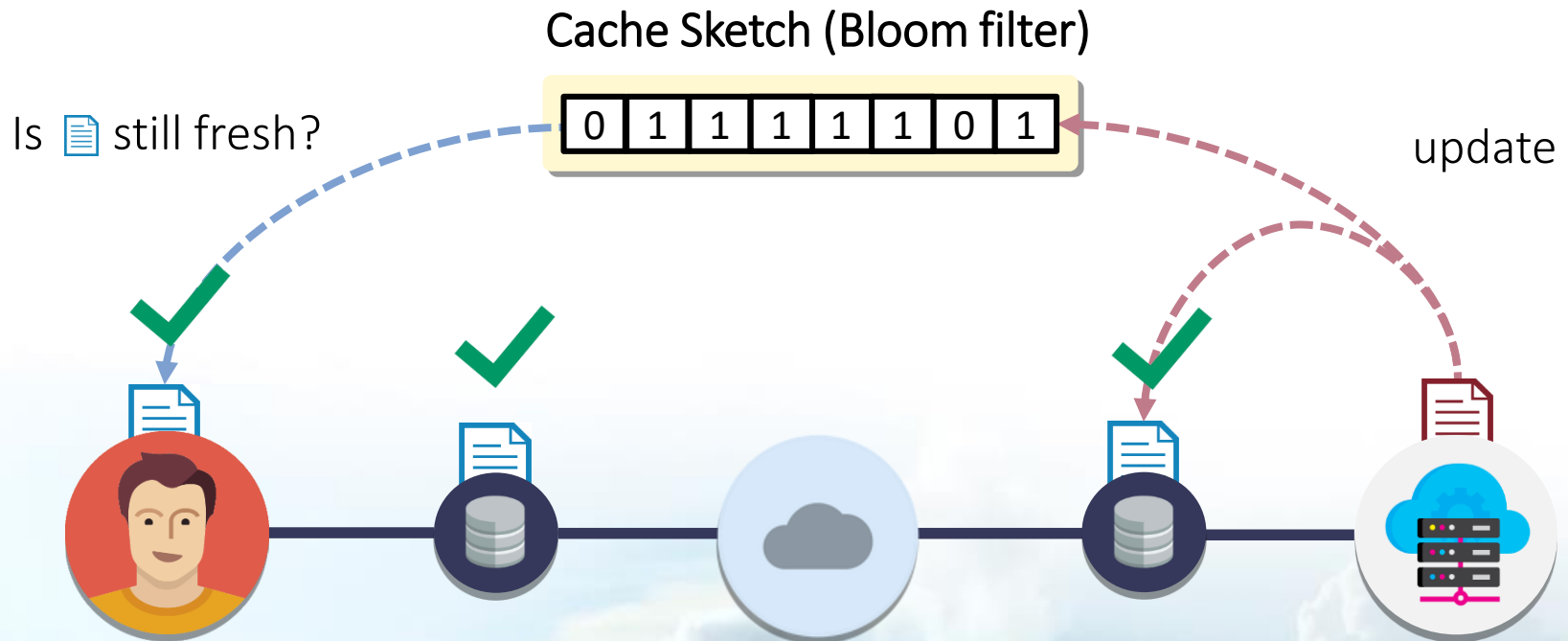


In a nutshell



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*.



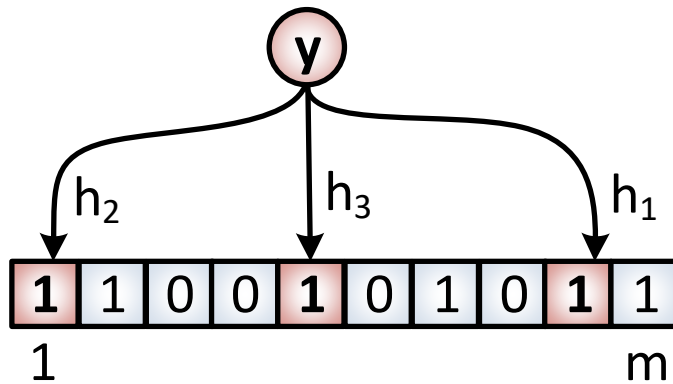
F. Gessert, „Skalierbare NoSQL- und Cloud-Datenbanken in Forschung und Praxis“, *BTW 2015*



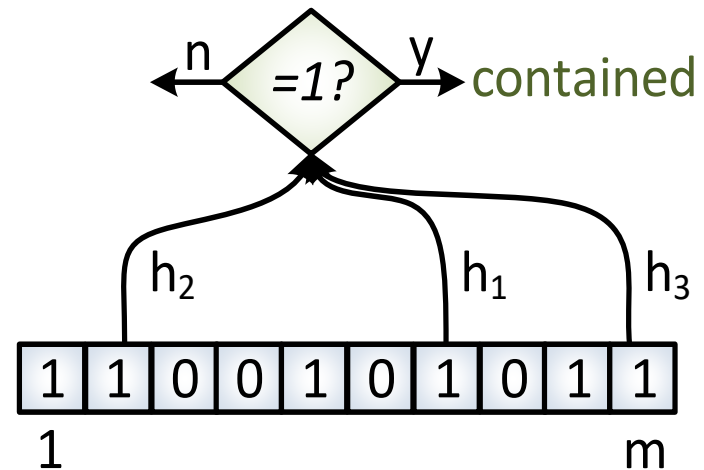
What is a Bloom filter?

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



Insert y

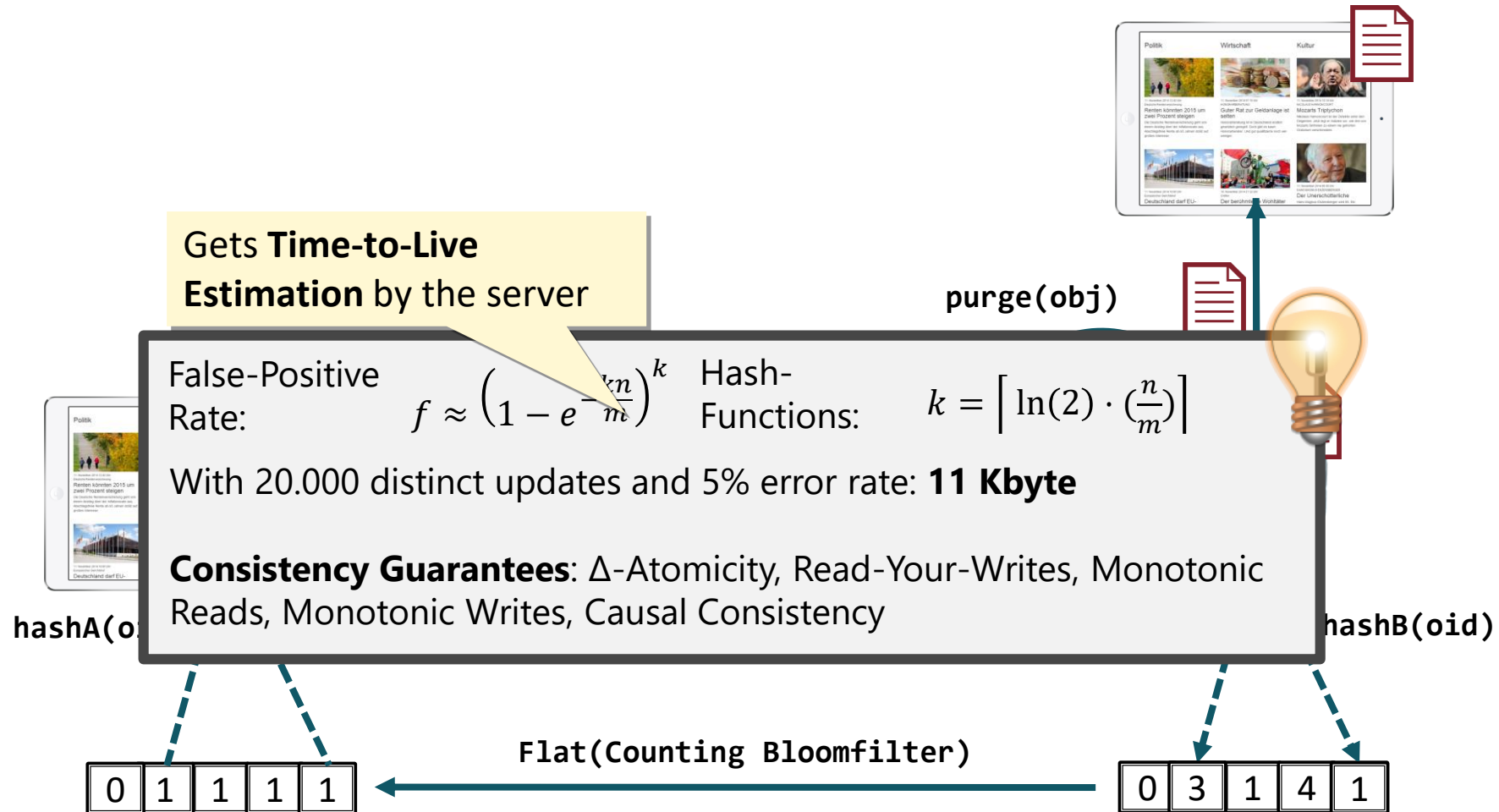


Query x



Bloom filters for Caching

End-to-End Example



Show me some Code!

End-to-End Example



Product About us Blog Whitepaper Docs

Log in

Querying Data

1 2 3 4 5 6 7 8 9

If we do not know the ids, we use queries to specify which objects we are looking for. The following query for instance gets us all the Todos in a specific list that are not marked as done:

```
DB.Todo.find()  
  .equal("listId", "myList")  
  .notEqual("done", true)  
  .resultList(myCallback); //Returns a list of matching objects
```

I did a query >

`resultList` returns all matching objects, whereas `singleResult` only returns the first match.

Baqend supports many query operators, which allow very sophisticated lookups, including ordering, geospatial queries, text search and boolean combinations of query predicates.

HTML CSS JS Result

EDIT ON CODEPEN

LIVE

```
1 var Todoservice = (function() {  
2   return {  
3     //load unfinished Todos  
4     unfinished: function(listId) {  
5       return DB.Todo.find()  
6         .equal("listId", listId)  
7         .notEqual("done", true)  
8         .resultList();  
9     }  
10  }})();  
11  
12 //Load data  
13 function onReady() {  
14   insertData().then(function() {  
15     return Todoservice.unfinished("my-list"); //query  
16   }).then(function(todos) {  
17     printItems("Unfinished Todos:", todos);  
18   });  
19 }  
20  
21 //Insert some data  
22 function insertData() {  
23   var todo1 = new DB.Todo();  
24   todo1.listId = "my-list";  
25   todo1.done = false;  
26   return DB.Todo.create(todo1);  
27 }  
28  
29 //Insert some data  
30 function insertData() {  
31   var todo2 = new DB.Todo();  
32   todo2.listId = "my-list";  
33   todo2.done = false;  
34   return DB.Todo.create(todo2);  
35 }  
36  
37 //Insert some data  
38 function insertData() {  
39   var todo3 = new DB.Todo();  
40   todo3.listId = "my-list";  
41   todo3.done = false;  
42   return DB.Todo.create(todo3);  
43 }  
44  
45 //Insert some data  
46 function insertData() {  
47   var todo4 = new DB.Todo();  
48   todo4.listId = "my-list";  
49   todo4.done = false;  
50   return DB.Todo.create(todo4);  
51 }  
52  
53 //Insert some data  
54 function insertData() {  
55   var todo5 = new DB.Todo();  
56   todo5.listId = "my-list";  
57   todo5.done = false;  
58   return DB.Todo.create(todo5);  
59 }  
60  
61 //Insert some data  
62 function insertData() {  
63   var todo6 = new DB.Todo();  
64   todo6.listId = "my-list";  
65   todo6.done = false;  
66   return DB.Todo.create(todo6);  
67 }  
68  
69 //Insert some data  
70 function insertData() {  
71   var todo7 = new DB.Todo();  
72   todo7.listId = "my-list";  
73   todo7.done = false;  
74   return DB.Todo.create(todo7);  
75 }  
76  
77 //Insert some data  
78 function insertData() {  
79   var todo8 = new DB.Todo();  
80   todo8.listId = "my-list";  
81   todo8.done = false;  
82   return DB.Todo.create(todo8);  
83 }  
84  
85 //Insert some data  
86 function insertData() {  
87   var todo9 = new DB.Todo();  
88   todo9.listId = "my-list";  
89   todo9.done = false;  
90   return DB.Todo.create(todo9);  
91 }  
92  
93 //Insert some data  
94 function insertData() {  
95   var todo10 = new DB.Todo();  
96   todo10.listId = "my-list";  
97   todo10.done = false;  
98   return DB.Todo.create(todo10);  
99 }  
100
```

Unfinished Todos:

- My Todo [id:/db/Todo/my-todo] from list my-list
- My other Todo [id:/db/Todo/bfc54e9e-e558-4031-aad2-4e4d7ba4cd86] from list my-list
- My other Todo [id:/db/Todo/my-todo-2] from list my-list
- My Todo [id:/db/Todo/7c8bb7f9-0992-4fc6-9d3d-1f6ec40c6ea5] from list my-list
- My Todo [id:/db/Todo/e6dc959b-5b6b-4f22-90c6-8bcd41ef7f] from list my-list
- My Todo [id:/db/Todo/6e687190-5c6f-4aaf-acc9-cccf27051380] from list my-list

Try this: www.baqend.com#tutorial

Our **Mission** at **Baqend**:

„Kick load times in the
ass using Bloom filters“

www.baqend.com

@Baqendcom

