

Wolfram Wingerath
wingerath@informatik.uni-hamburg.de



Universität Hamburg



Real-Time Databases Explained: Why Meteor, RethinkDB, Parse and Firebase Don't Scale

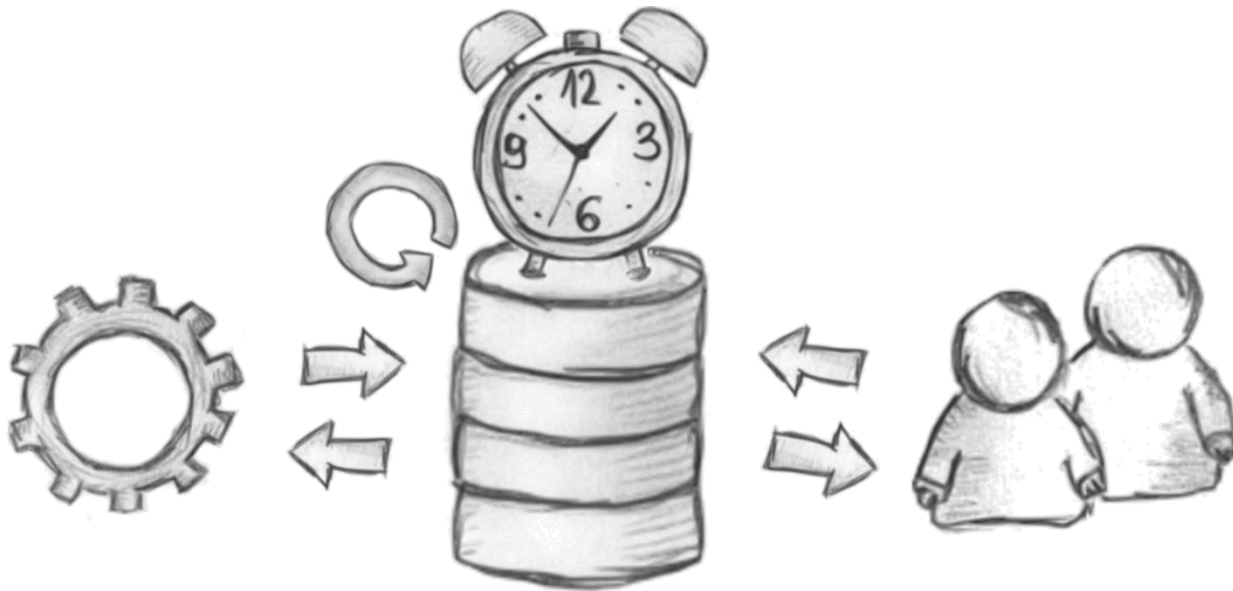
Infrastructure & DevOps

September 28, 2017



Real-Time Databases Explained

Why Meteor, RethinkDB, Parse and Firebase Don't Scale



Wolfram Wingerath

wingerath@informatik.uni-hamburg.de

September 28, 2017

Who I Am

*Research
& Teaching*



Wolfram
Wingerath

*Software
Development*



PhD studies:

- Real-Time Databases
- Stream Processing
- NoSQL Databases
- Database Benchmarking
- ...



Universität Hamburg

Baqend:

High-Performance
Backend-as-a-Service



Outline



Push-Based Data Access

Why Real-Time Databases?

- Pull-based data access
- Self-maintaining results



Real-Time Databases

System survey



Discussion

What are the bottlenecks?



Baqend Real-Time Queries

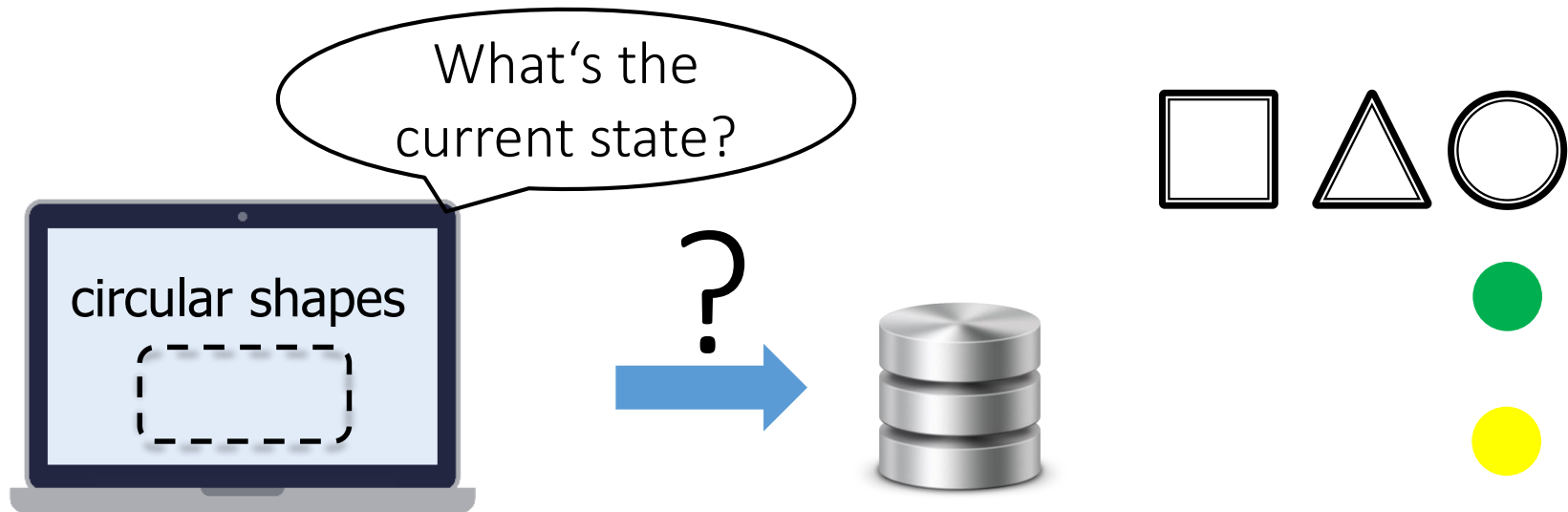
How do they scale?

```
        'replace_interests' => false,  
        'send_welcome'      => false,  
    });  
  
    array('error', $result)) {  
        $result = array ('response'=>'error', 'message'  
        (   
            $result = array ('response'=>'success');
```

Push-Based Data Access

Traditional Databases

No Request? No Data!



Query maintenance: periodic polling

→ **Inefficient**

→ **Slow**

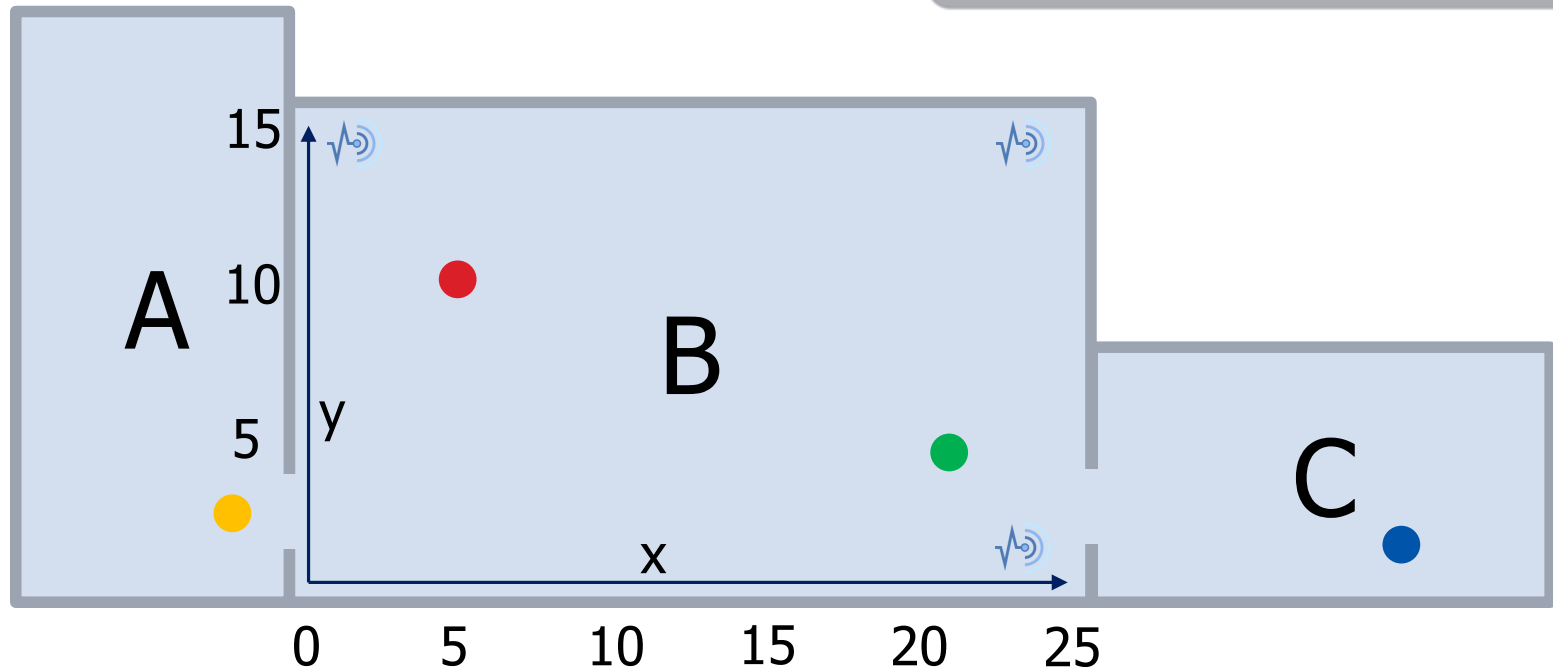
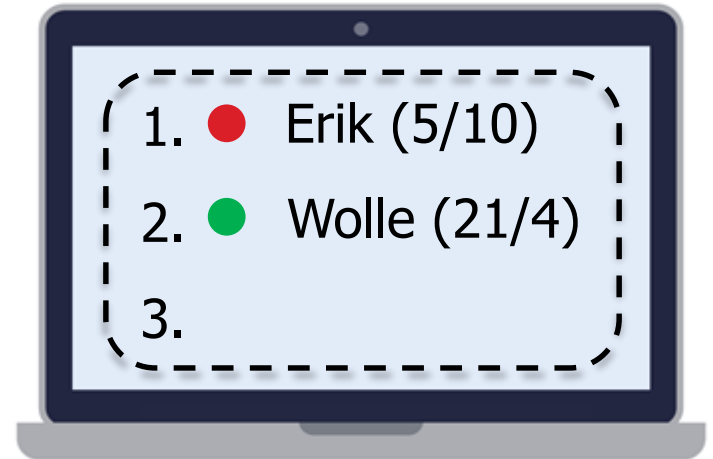


Ideal: Push-Based Data Access

Self-Maintaining Results

Find people in Room B:

```
db.User.find()  
  .equal('room', 'B')  
  .ascending('name')  
  .limit(3)  
  .resultStream()
```



Outline



Push-Based Data Access

Why Real-Time Databases?



Real-Time Databases

System survey



Discussion

What are the bottlenecks?



Baqend Real-Time Queries

How do they scale?

- Meteor
- RethinkDB
- Parse
- Firebase
- Others



Real-Time Databases

Overview:

- **JavaScript Framework** for interactive apps and websites
 - **MongoDB** under the hood
 - **Real-time** result updates, full MongoDB expressiveness
- **Open-source**: MIT license
- **Managed service**: Galaxy (Platform-as-a-Service)

History:

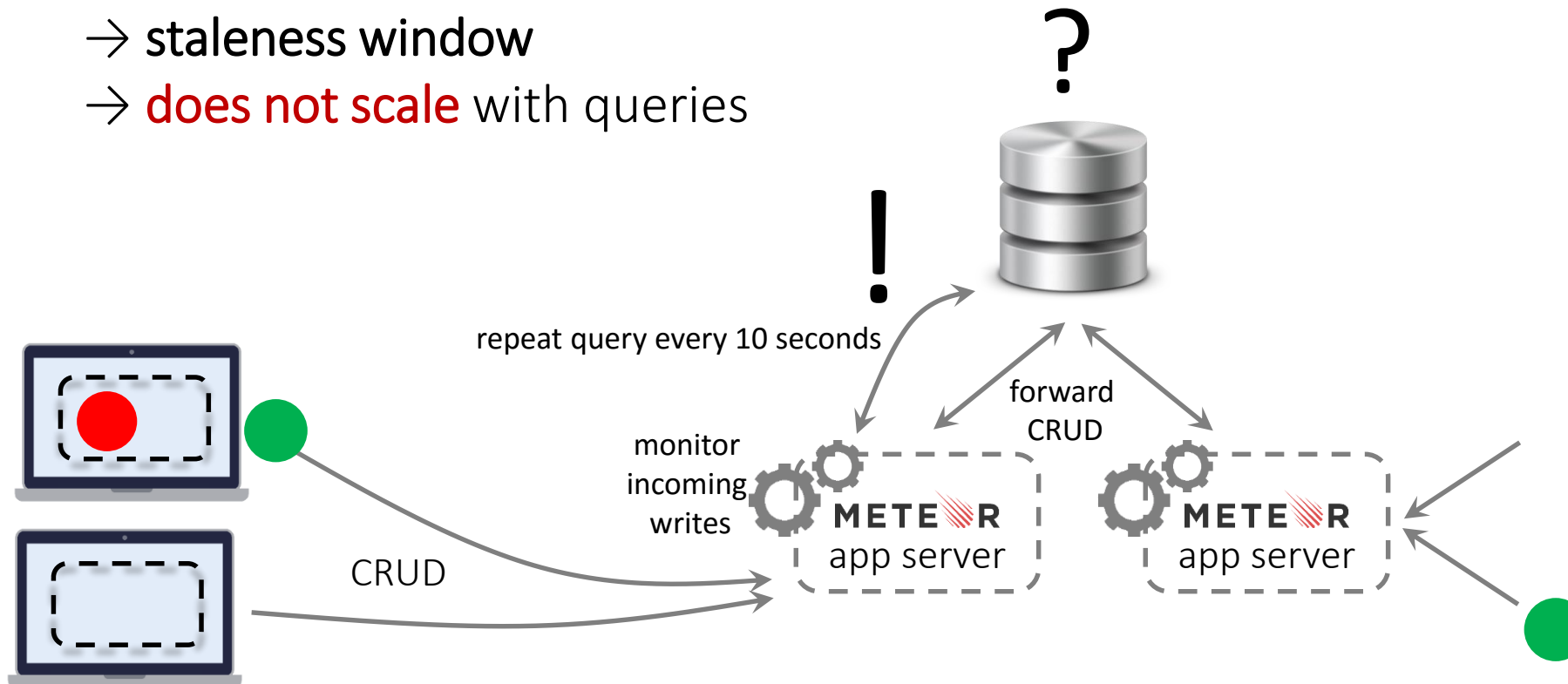
- 2011: *Skybreak* is announced
- 2012: Skybreak is renamed to Meteor
- 2015: Managed hosting service Galaxy is announced

Live Queries

Poll-and-Diff



- **Change monitoring:** app servers detect relevant changes
→ *incomplete* in multi-server deployment
- **Poll-and-diff:** queries are re-executed periodically
→ **staleness window**
→ **does not scale** with queries



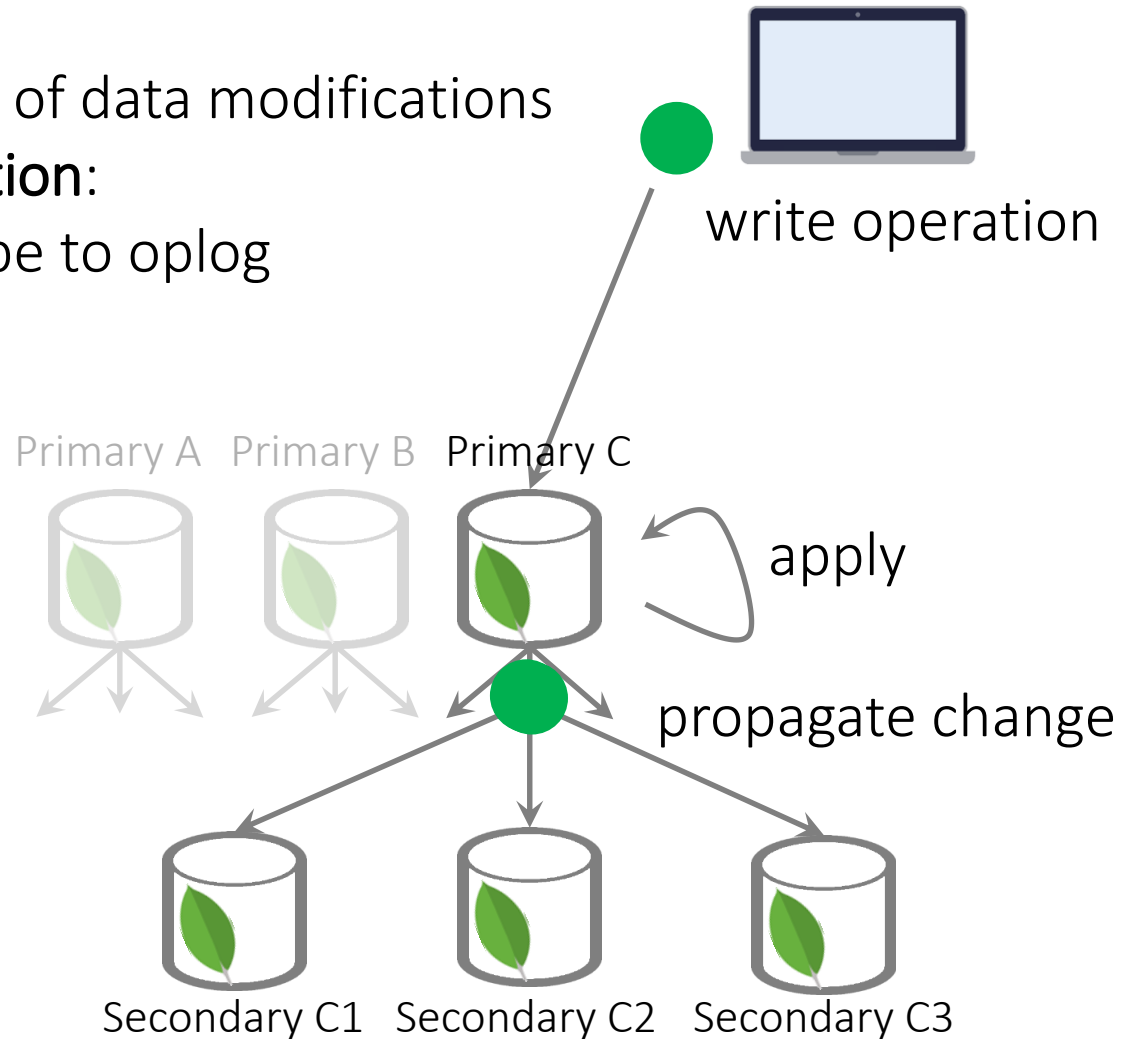
Oplog Tailing

Basics: MongoDB Replication



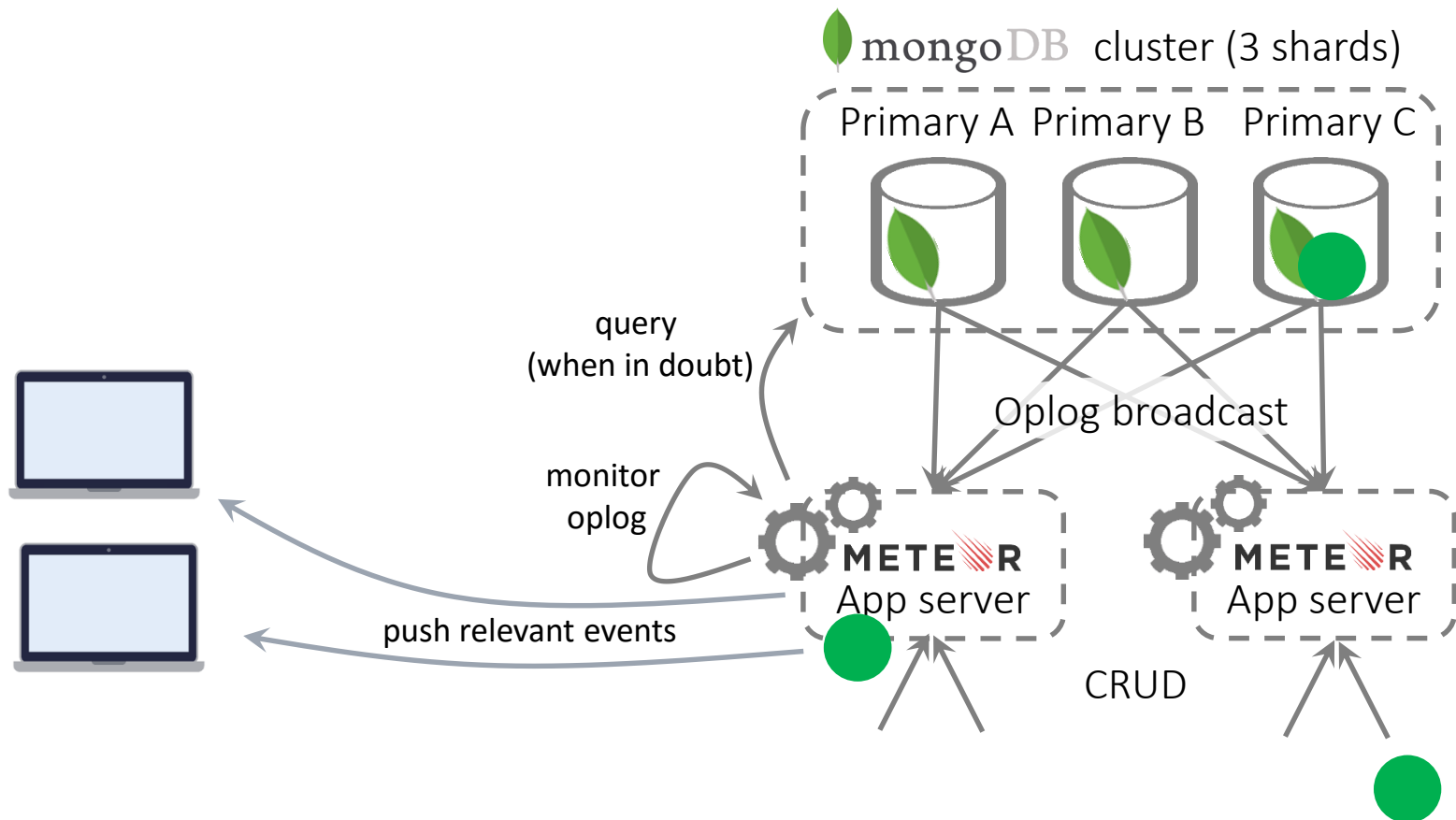
- **Oplog:** rolling record of data modifications
- **Master-slave replication:**
Secondaries subscribe to oplog

 **mongoDB** cluster
(3 shards)



Oplog Tailing

Tapping into the Oplog



Oplog Tailing

Oplog Info is Incomplete



What game does Bobby play?

- if baccarat, he takes first place!
- if something else, nothing changes!

Partial update from oplog:

```
{ name: „Bobby“, score: 500 } // game: ???
```

Baccarat players sorted by high-score

The logo for METEOR, featuring the word "METEOR" in a bold, sans-serif font. The "E" is stylized with three red diagonal lines extending from its top-right corner.

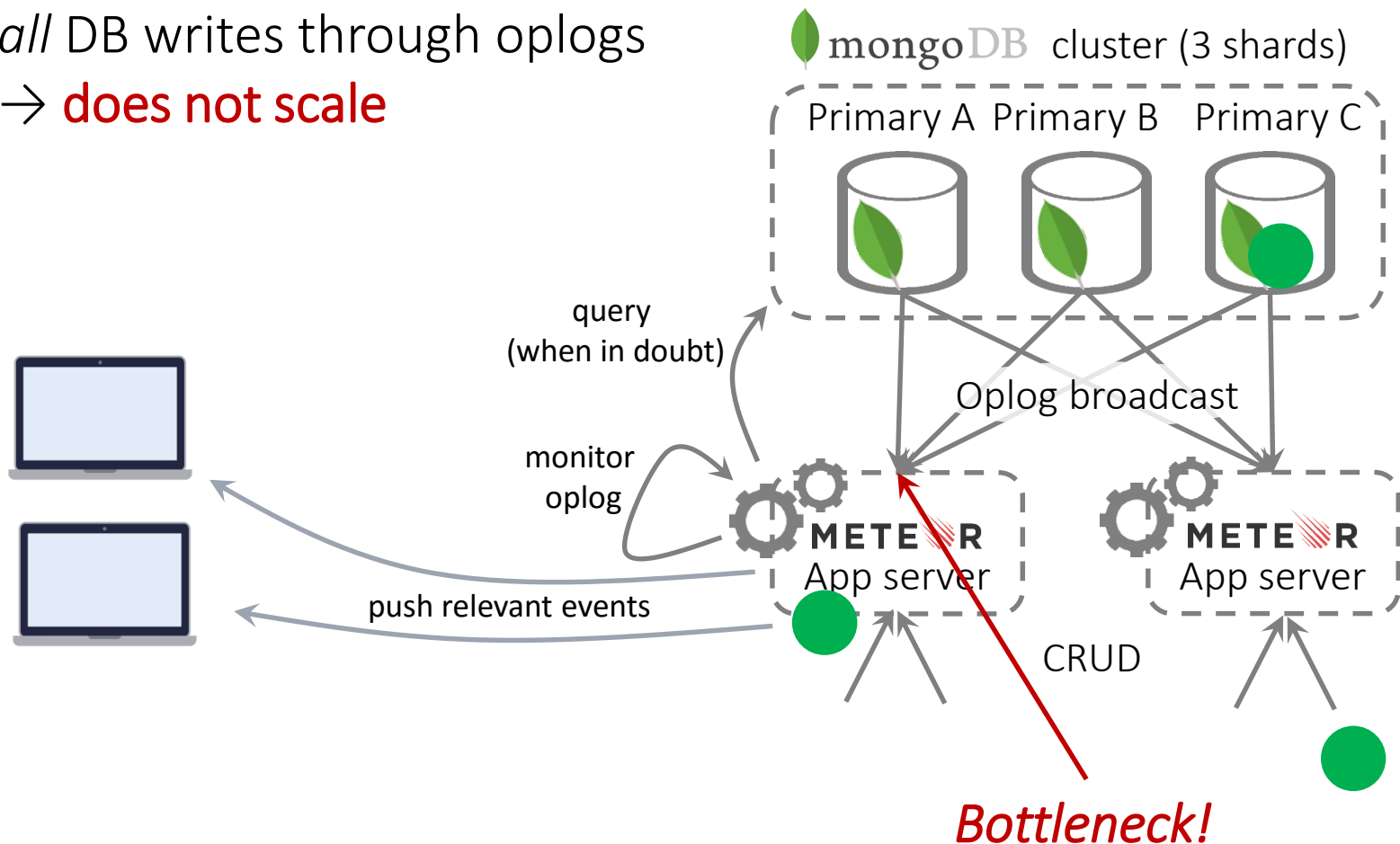
1. { name: „Joy“, game: „baccarat“, score: 100 }
2. { name: „Tim“, game: „baccarat“, score: 90 }
3. { name: „Lee“, game: „baccarat“, score: 80 }

Oplog Tailing

Tapping into the Oplog



- Every Meteor server receives *all* DB writes through oplogs
→ **does not scale**



Overview:

- „MongoDB done right“: comparable queries and data model, but also:
 - Push-based queries (filters only)
 - Joins (non-streaming)
 - Strong consistency: linearizability
- JavaScript SDK (*Horizon*): open-source, as managed service
- Open-source: Apache 2.0 license

History:

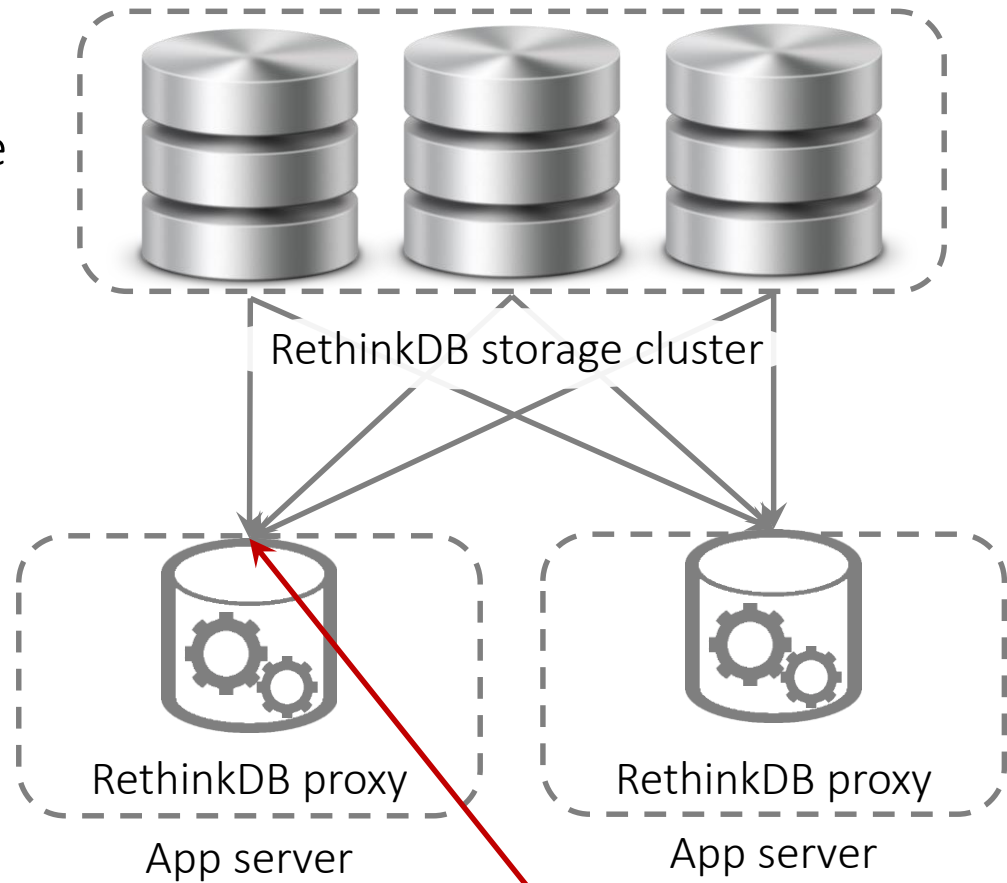
- 2009: RethinkDB is founded
- 2012: RethinkDB is open-sourced under AGPL
- 2016, May: first official release of Horizon (JavaScript SDK)
- 2016, October: RethinkDB announces shutdown
- 2017: RethinkDB is relicensed under Apache 2.0

RethinkDB

Changefeed Architecture



- Range-sharded data
- **RethinkDB proxy**: support node without data
 - Client communication
 - Request routing
 - Real-time query matching
- Every proxy receives *all* database writes
→ **does not scale**



Bottleneck!



William Stein, *RethinkDB versus PostgreSQL: my personal experience* (2017)
<http://blog.sagemath.com/2017/02/09/rethinkdb-vs-postgres.html> (2017-02-27)



Daniel Mewes, *Comment on GitHub issue #962: Consider adding more docs on RethinkDB Proxy* (2016)
<https://github.com/rethinkdb/docs/issues/962> (2017-02-27)

Overview:

- **Backend-as-a-Service** for mobile apps
 - **MongoDB**: largest deployment world-wide
 - **Easy development**: great docs, push notifications, authentication, ...
 - **Real-time** updates for most MongoDB queries
- **Open-source**: BSD license
- **Managed service**: discontinued

History:

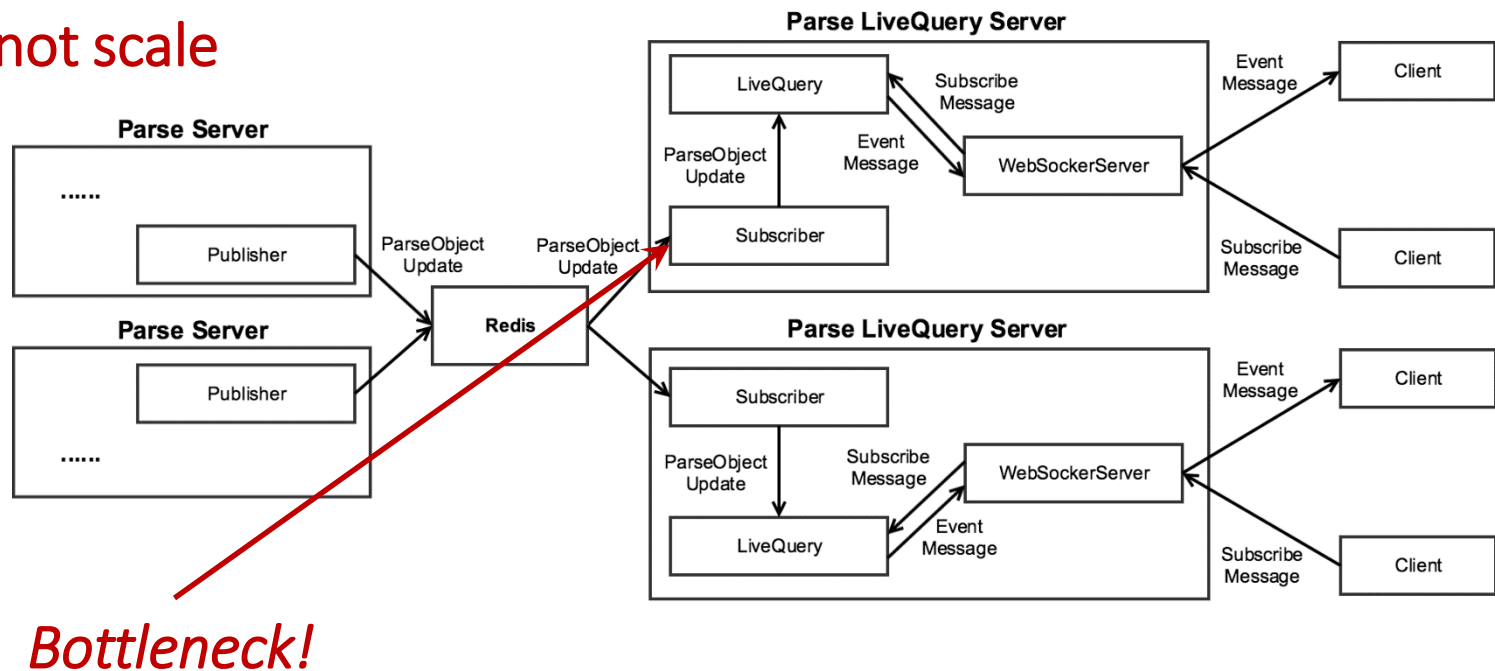
- 2011: Parse is founded
- 2013: Parse is acquired by Facebook
- 2015: more than 500,000 mobile apps reported on Parse
- 2016, January: Parse shutdown is announced
- 2016, March: **Live Queries** are announced
- 2017: Parse shutdown is finalized

Parse

LiveQuery Architecture



- **LiveQuery Server:** no data, real-time query matching
- *Every* LiveQuery Server receives *all* database writes
→ **does not scale**



Overview:

- **Real-time state synchronization** across devices
- **Simplistic data model**: nested hierarchy of lists and objects
- **Simplistic queries**: mostly navigation/filtering
- **Fully managed**, proprietary
- **App SDK** for App development, mobile-first
- **Google services integration**: analytics, hosting, authorization, ...

History:

- 2011: chat service startup Envolv is founded
 - was often used for cross-device state synchronization
 - state synchronization is separated (Firestore)
- 2012: Firestore is founded
- 2013: Firestore is acquired by Google

Firestore

Real-Time State Synchronization



- **Tree data model:** application state ~JSON object
- **Subtree synching:** push notifications for specific keys only
→ Flat structure for fine granularity

→ *Limited expressiveness!*

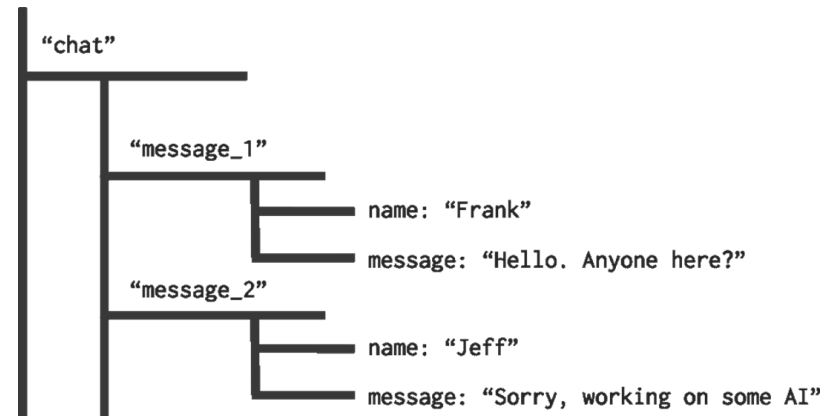


Firestore



Query Processing in the Client

- Push notifications for **specific keys** only
 - Order by a **single attribute**
 - Apply a **single filter** on that attribute
- Non-trivial query processing in client
→ **does not scale!**



Jacob Wenger, on the Firestore Google Group (2015)

<https://groups.google.com/forum/#!topic/firestore-talk/d-XjaBVL2Ko> (2017-02-27)



Illustration taken from: Frank van Puffelen, *Have you met the Realtime Database?* (2016)

<https://firebase.googleblog.com/2016/07/have-you-met-realtime-database.html> (2017-02-27)

Honorable Mentions

Other Systems With Real-Time Features



Outline



Push-Based Data Access

Why Real-Time Databases?



Real-Time Databases

System survey



Discussion

What are the bottlenecks?



Baqend Real-Time Queries

How do they scale?

- System classification:
 - Databases
 - Real-time databases
 - Stream management
 - Stream processing
- Side-by-side comparison



Discussion

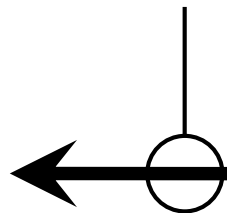
Quick Comparison

DBMS vs. RT DB vs. DSMS vs. Stream Processing



Database Management

static collections

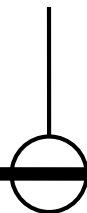


pull-based



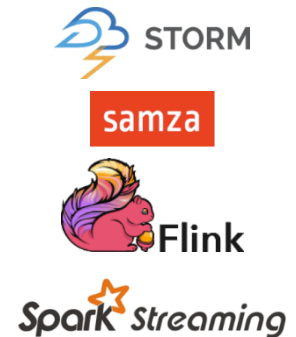
Real-Time Databases

evolving collections



Data Stream Management

persistent/
ephemeral streams



Stream Processing

ephemeral
streams



push-based

Wrap-Up

Direct Comparison



METEOR

 **RethinkDB**

 **Parse**

 **Firebase**

BaQend

	Meteor		RethinkDB	Parse	Firebase	Baqend
	Poll-and-Diff	Oplog Tailing				
Scales with write TP	✓	✗	✗	✗	?	✓
Scales with no. of queries	✗	✓	✓	✓		✓
Composite queries (AND/OR)	✓	✓	✓	✓	✗	✓
Sorted queries	✓	✓	✓	✗	○ (single attribute)	✓
Limit	✓	✓	✓	✗	✓	✓
Offset	✓	✓	✗	✗	✓	✓

Outline



Push-Based Data Access

Why Real-Time Databases?



Real-Time Databases

System survey



Discussion

What are the bottlenecks?



Baqend Real-Time Queries

How do they scale?

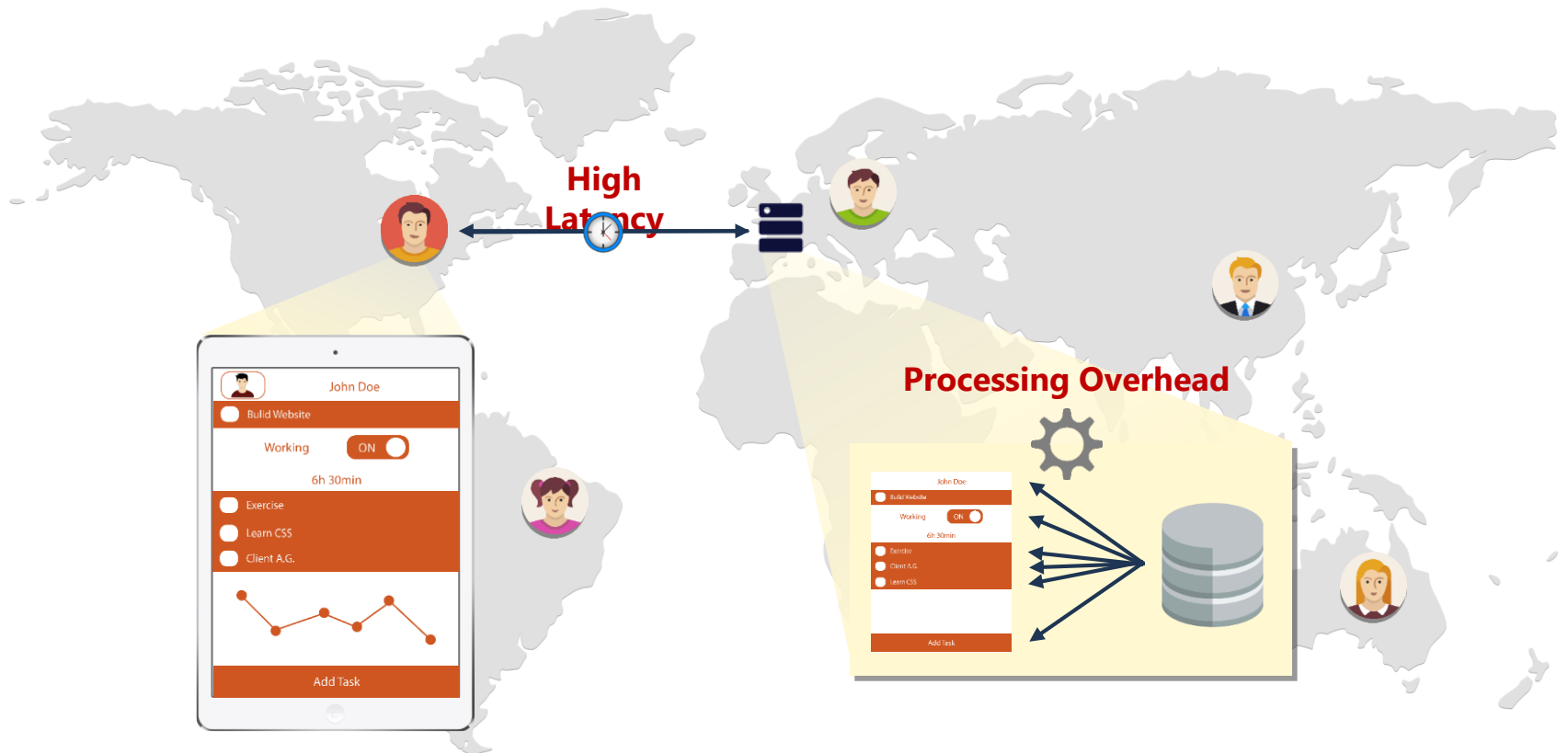
- InvaliDB: opt-in real-time queries
- System architecture
- Query expressiveness
- Performance & scalability
- Example app: Twoogle



Baqend Real-Time Queries

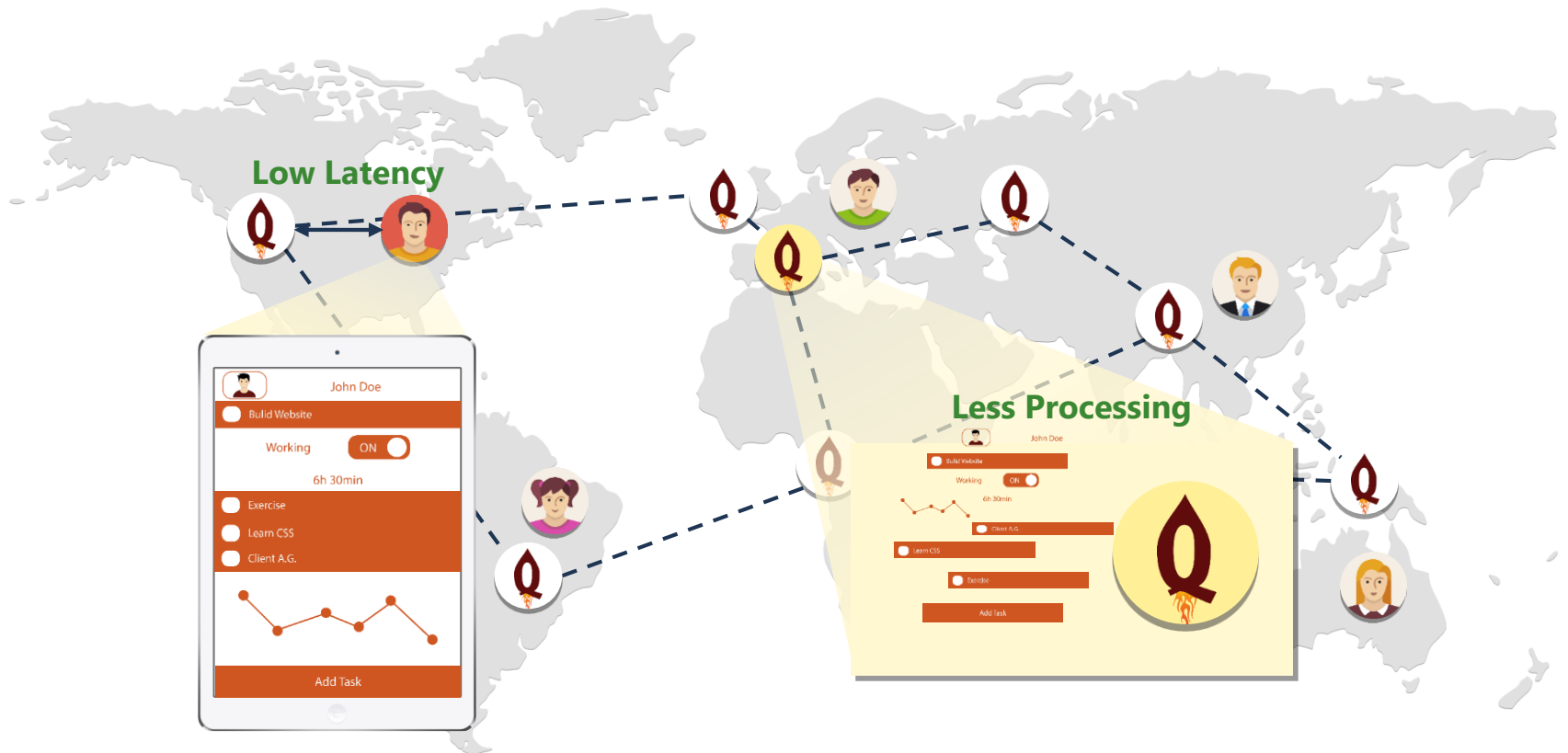
Problem: Slow Websites

Two Bottlenecks: Latency and Processing



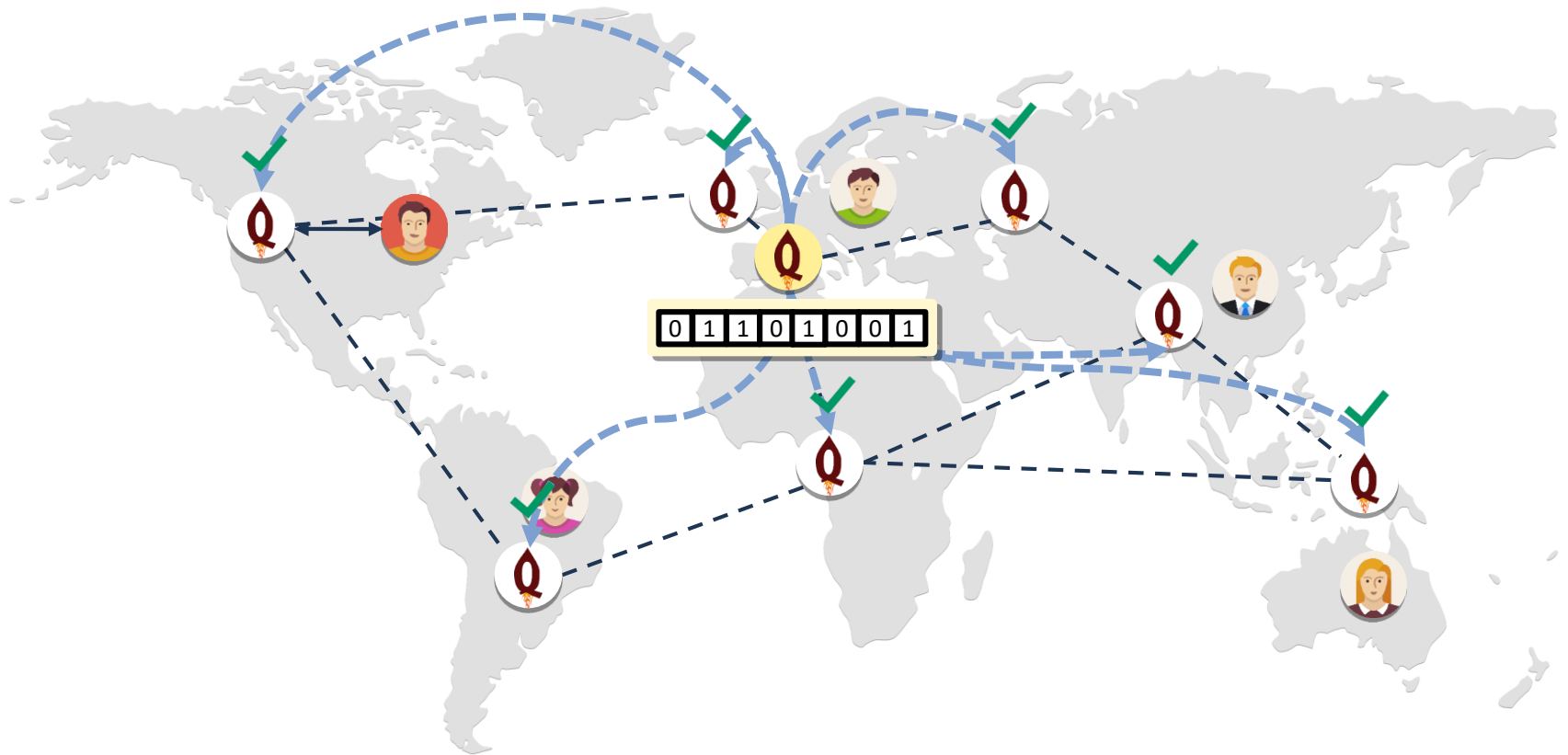
Solution: Global Caching

Fresh Data From Distributed Web Caches



New Caching Algorithms

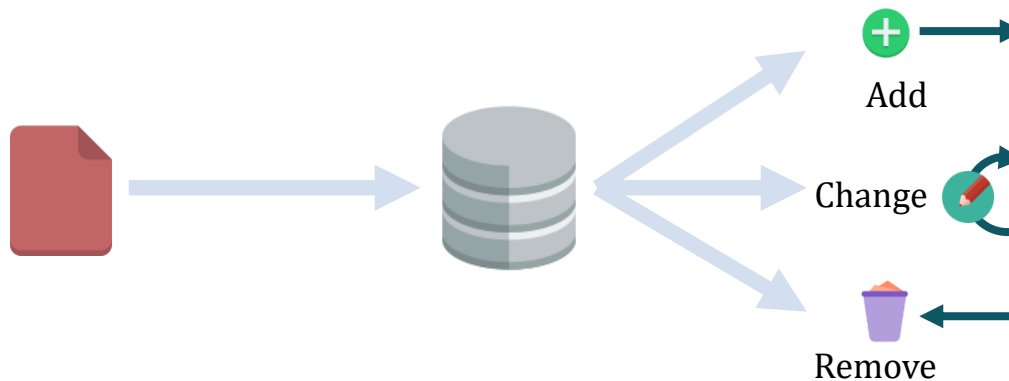
Solve Consistency Problem







InvaliDB

Invalidating DB Queries

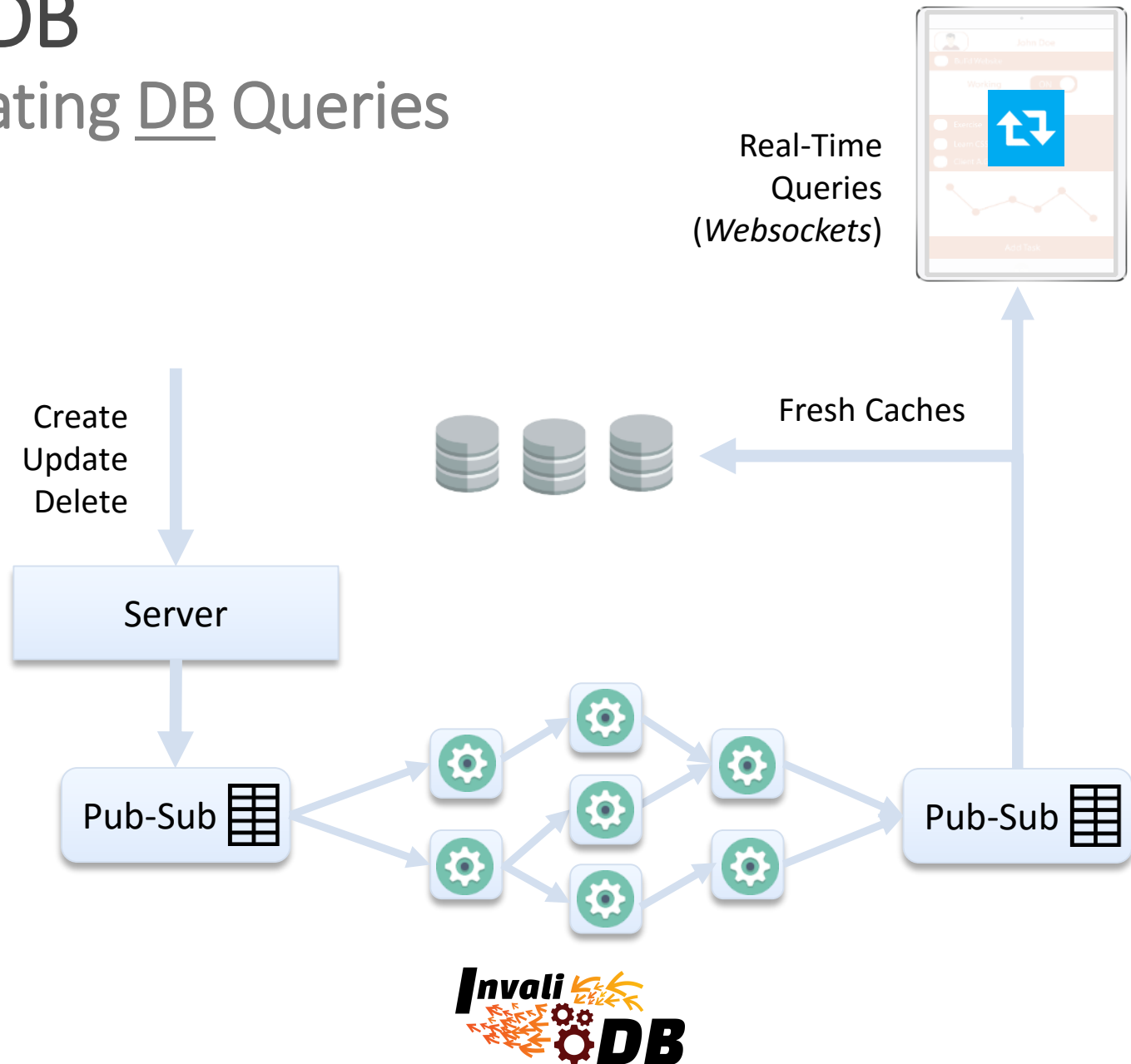
How to detect changes to query results:
„Give me the most popular products that are in stock.“



 <p>DEAL OF THE DAY \$10.25 - \$179.99 Ends in 16:45:48 Up to 50% Off Handbags ★★★★★ 21</p> <p>See details</p>	 <p>DEAL OF THE DAY \$97.99 List: \$149.95 (35% off) Ends in 16:45:48 Save on Hitachi Gas Powered Leaf Blower Ships from and sold by Amazon.com. ★★★★☆ 1961</p> <p>Add to Cart</p>
 <p>\$15.63 - \$16.79 9% Claimed Ends in 4:40:49 BESTEK surge protector Sold by BESTEK, and Fulfilled by Amazon. ★★★★★ 162</p> <p>Choose options</p>	 <p>\$18.66 Price: \$39.99 (53% off) 18% Claimed Ends in 3:05:49 AUKEY Table Lamp, Touch Sensor Bedside Lamp + Dimmable War... Sold by Aukey Direct and Fulfilled by Amazon. ★★★★☆ 669</p> <p>Add to Cart</p>

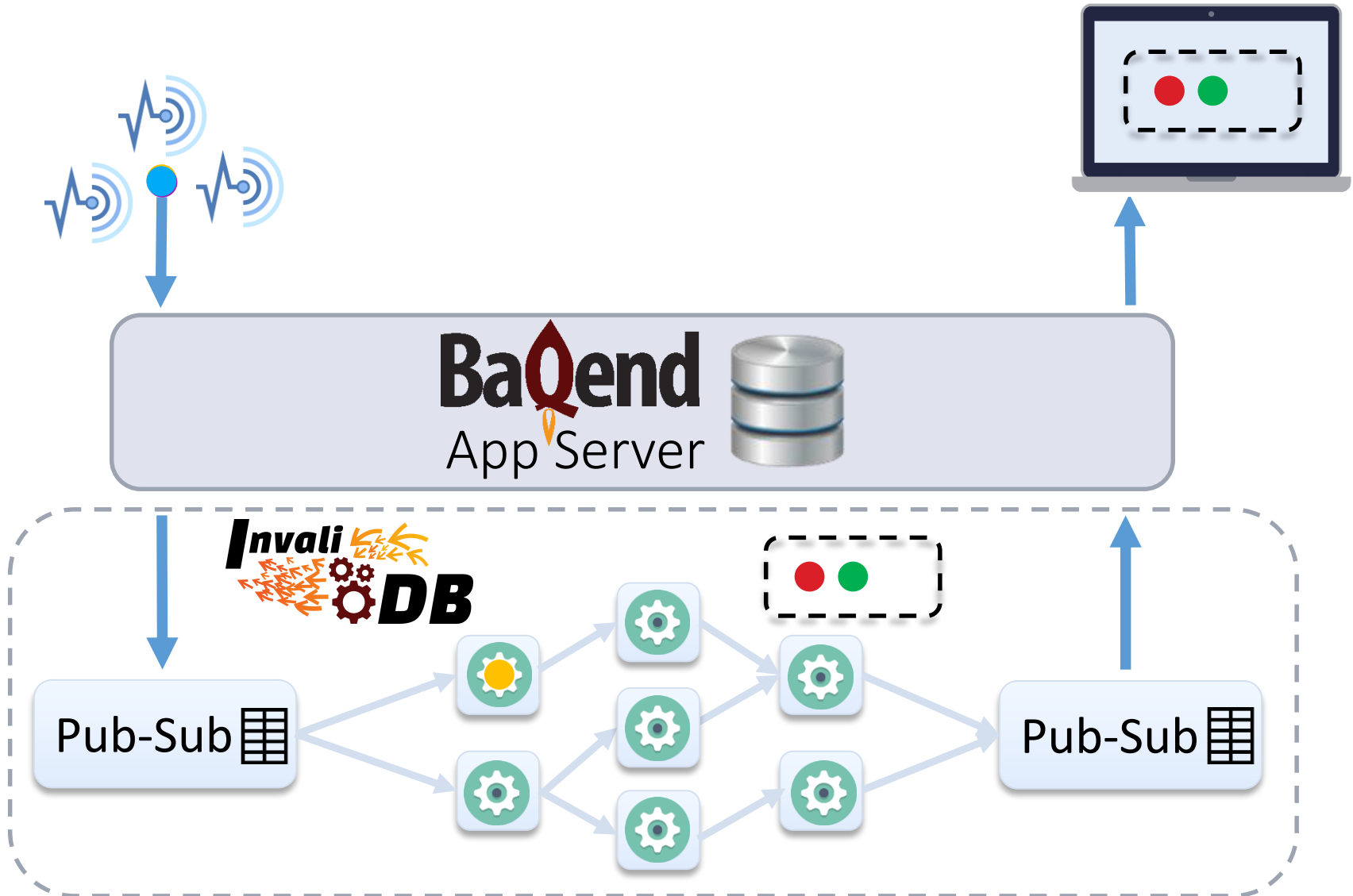
InvaliDB

Invalidating DB Queries



Baqend Real-Time Queries

Real-Time Decoupled

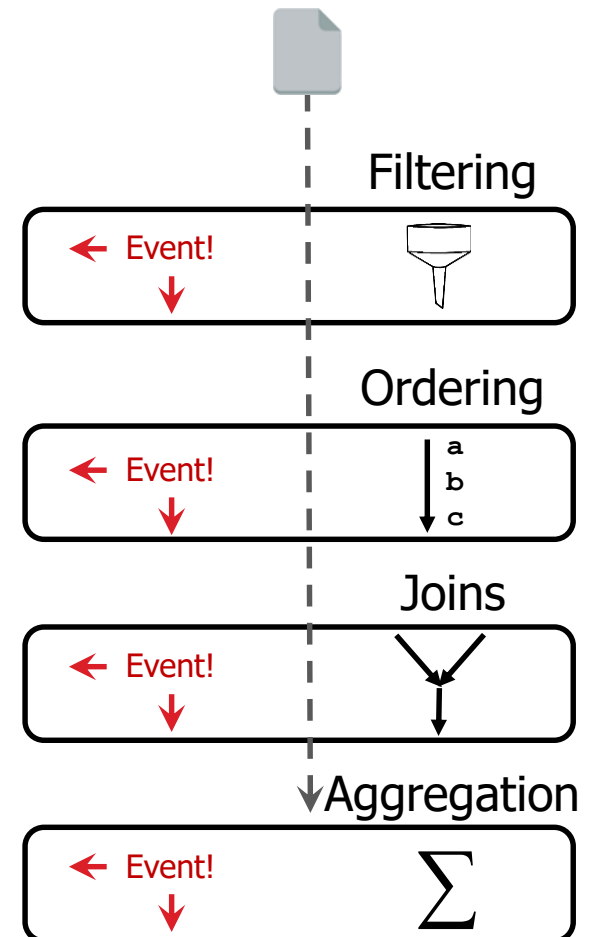


Baqend Real-Time Queries

Staged Real-Time Query Processing

Change notifications go through up to 4 query processing stages:

1. **Filter queries:** track matching status
→ *before-* and *after-*images
2. **Sorted queries:** maintain result order
3. **Joins:** combine maintained results
4. **Aggregations:** maintain aggregations



Baqend Real-Time Queries

Filter Queries: Distributed Query Matching

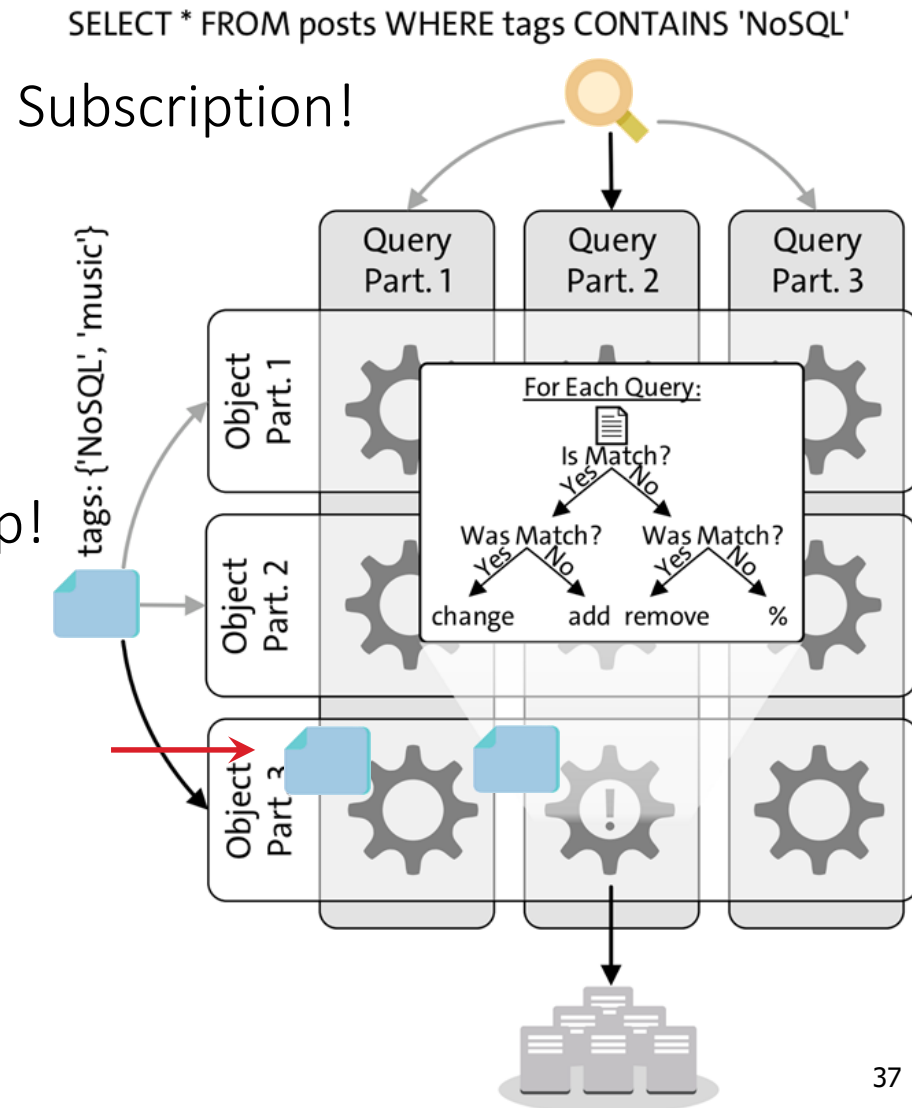
Two-dimensional partitioning:

- *by Query*
 - *by Object*
- **scales with queries and writes**

Implementation:

- Apache Storm
- Topology in Java
- MongoDB query language
- **Pluggable query engine**

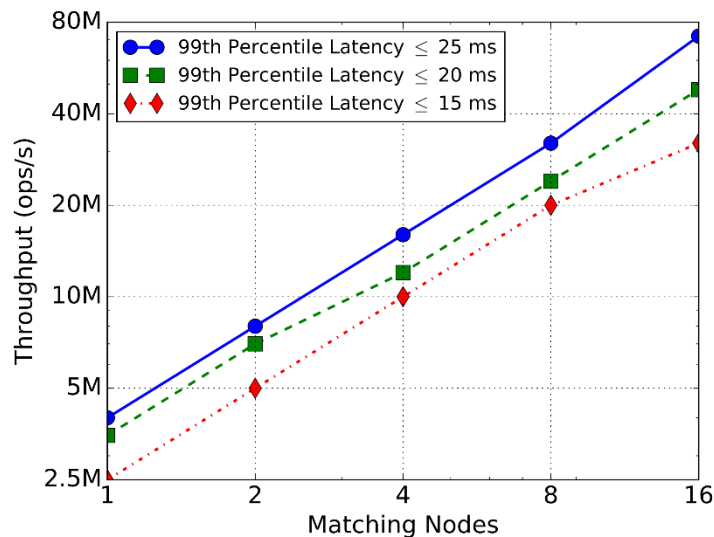
Write op!



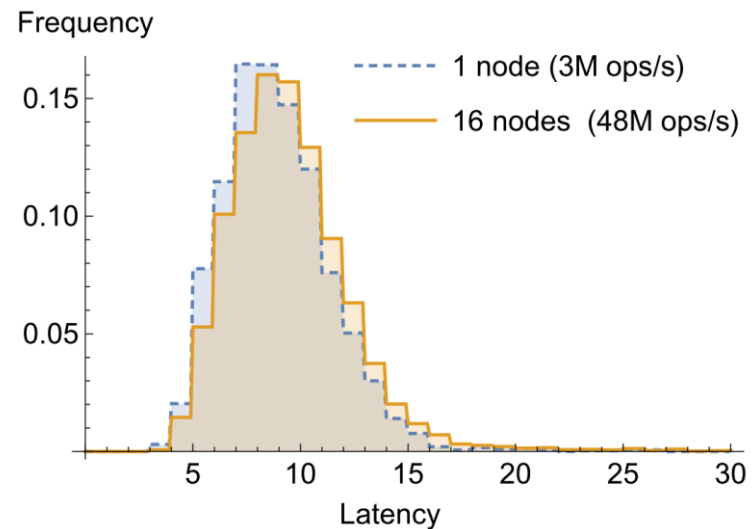
Baqend Real-Time Queries

Low Latency + Linear Scalability

Linear Scalability



Stable Latency Distribution



Programming Real-Time Queries

JavaScript API

```
var query = DB.Tweet.find()  
  .matches('text', /my filter/)  
  .descending('createdAt')  
  .offset(20)  
  .limit(10);
```

Static Query

```
query.resultList(result => ...);
```

Google

Real-Time Query

```
query.resultStream(result => ...);
```

Twoogle

**Real-Time**

Static

Last result update at 15:51:21 (less than a second ago)

1. Conju.re (conju_re, 3840 followers) tweeted:
https://twitter.com/conju_re/status/859767327570702336

Congress Saved the Science Budget—And That's the Problem <https://t.co/UdrjNidakc>
<https://t.co/xlNjpEpKZG>

2. ねぼすけゆーだい (Yuuu__key, 229 followers) tweeted:
https://twitter.com/Yuuu__key/status/859767323384623104

けいきさんと PENGUIN RESEARCHのけいたくん がリプのやり取りしてる...

3. Whitney Shackley (bschneids11, 5 followers) tweeted:
<https://twitter.com/bschneids11/status/859767319534469122>

holy..... waiting for it so long 🍷 © <https://t.co/UdXcHJb7X3>

4. Lisa Schmid (LisaMSchmid, 67 followers) tweeted on #teamscs, and #scs...
<https://twitter.com/LisaMSchmid/status/859767317311500290>

Congrats to Matthew Kent, winner of the 26th #TeamSCSCoding Challenge.
<https://t.co/vx1o0WgJrZ> #SCSChallenge

5. Brian Martin Larson (Brian_Larson, 40 followers) tweeted on #teamscs, a...
https://twitter.com/Brian_Larson/status/859767317303001089

Congrats to Matthew Kent, winner of the 26th #TeamSCSCoding Challenge.

**Real-Time**

Static

Last result update at 15:51:21 (less than a second ago)

1. Conju.re (conju_re, 3840 followers) tweeted:
https://twitter.com/conju_re/status/859767327570702336

Congress Saved the Science Budget—And That's the Problem <https://t.co/UdrjNidakc>
<https://t.co/xlNjpEpKZG>

2. ねぼすけゆーだい (Yuuu__key, 229 followers) tweeted:
https://twitter.com/Yuuu__key/status/859767323384623104

けいきさんと PENGUIN RESEARCHのけいたくん がリプのやり取りしてる...

3. Whitney Shackley (bschneids11, 5 followers) tweeted:
<https://twitter.com/bschneids11/status/859767319534469122>

holy..... waiting for it so long 🍷 © <https://t.co/UdXcHJb7X3>

4. Lisa Schmid (LisaMSchmid, 67 followers) tweeted on #teamscs, and...
<https://twitter.com/LisaMSchmid/status/859767317311500290>

Congrats to Matthew Kent, winner of the 26th #TeamSCSCoding Challenge.
<https://t.co/vx1o0WgJrZ> #SCSChallenge

5. Brian Martin Larson (Brian_Larson, 40 followers) tweeted on #teams...
https://twitter.com/Brian_Larson/status/859767317303001089

Congrats to Matthew Kent, winner of the 26th #TeamSCSCoding Challenge.

Wrap-up



▶ Push-based Data Access

- Natural for many applications
- Hard to implement on top of traditional (pull-based) databases

▶ Real-time Databases


- Natively push-based
- Not legacy-compatible
- Barely scalable


▶ Baqend Real-Time Queries


- No impact on OLTP workload
- Linear scalability
- Low latency
- Filter, sorting, joins, aggregations


Our Related Publications

Scientific Papers:

 *Quaestor: Query Web Caching for Database-as-a-Service Providers*
VLDB '17

 *NoSQL Database Systems: A Survey and Decision Guidance*
SummerSOC '16

 *Real-time stream processing for Big Data*
it - Information Technology 58 (2016)

 *The Case For Change Notifications in Pull-Based Databases*
BTW '17

Blog Posts:

 *Real-Time Databases Explained: Why Meteor, RethinkDB, Parse and Firebase Don't Scale*
Baqend Tech Blog (2017): <https://medium.com/p/822ff87d2f87>

A Real-Time Database Survey: The Architecture of Meteor, RethinkDB, Parse & Firebase

Real-time databases make it easy to implement reactive applications, because they keep your critical information up-to-date. But how do they work and how do they scale? In this article, we dissect the real-time query features of Meteor, RethinkDB, Parse and Firebase to uncover scaling limitations inherent to their respective designs. We then go on to discuss and categorize related real-time systems and share our lessons learned in providing real-time queries without any bottlenecks in Baqend.



A Real-Time Database Survey:
The Architecture of Meteor, RethinkDB, Parse & Firebase

Learn more at blog.baqend.com!



We are hiring.

Frontend Developers

Mobile Developers

Java Developers

Web Performance Engineers

Contact us.



Wolfram Wingerath · ww@baqend.com · www.baqend.com

Questions?

Fr. 10:00 Real-Time Anwendungen mit React und React Native entwickeln

Fr. 14:00 AMP, PWAs, HTTP/2 and Service Workers: A new Era of Web Performance?

Fr. 17:00 Wie man ein Backend-as-a-Service entwickelt: Lessons Learned



Wolfram Wingerath

wingerath@informatik.uni-hamburg.de