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

Wolfram Wingerath
ww@baqend.com
October 26, 2017
Who I Am

Research & Teaching

Wolfram Wingerath

PhD studies:
• Real-Time Databases
• Stream Processing
• NoSQL Databases
• Database Benchmarking
• ...

Universität Hamburg

Software Development

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

www.baqend.com
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?
Push-Based Data Access
Traditional Databases

No Request? No Data!

What’s the current state?

circular shapes

Query maintenance: periodic polling
→ Inefficient
→ Slow
Ideal: Push-Based Data Access
Self-Maintaining Results

Find people in Room B:

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

1. Erik (5/10)
2. Wolle (22/8)
3. Wolle (19/13)
4. Flo (4/3)
5. Erik (5/10)

Ideal: Push-Based Data Access
Self-Maintaining Results
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

Discussion
What are the bottlenecks?
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

repeat query every 10 seconds

monitor incoming writes

forward CRUD

CRUD
Oplog Tailing

**Basics: MongoDB Replication**

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

![Diagram of MongoDB cluster](image)

- **MongoDB cluster** (3 shards)
- **Primary A**, **Primary B**, **Primary C**
- **Secondary C1**, **Secondary C2**, **Secondary C3**
- Write operation
- Apply
- Propagate change
Oplog Tailing
Tapping into the Oplog

mongoDB cluster (3 shards)

Primary A  Primary B  Primary C

Oplog broadcast

query
(when in doubt)

App server

push relevant events

App server

CRUD
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

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**


[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

Illustration taken from:
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 Envolve is founded
  → was often used for cross-device state synchronization
  → state synchronization is separated (Firebase)
- 2012: Firebase is founded
- 2013: Firebase is acquired by Google
Firebase
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!*

Illustration taken from: Frank van Puffelen, *Have you met the Realtime Database?* (2016)  
Firebase
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 Firebase Google Group (2015)
https://groups.google.com/forum/#!topic/firebase-talk/d-XjaBVL2Ko (2017-02-27)

Illustration taken from: Frank van Puffelen, *Have you met the Realtime Database?* (2016)
Honorable Mentions
Other Systems With Real-Time Features

GRAPHCOOL
rapid.io
BETA
CouchDB
OrientDB®
elasticsearch
mongodb
realm
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
Quick Comparison
DBMS vs. RT DB vs. DSMS vs. Stream Processing

- **Meteor**
  - RethinkDB
  - Parse
  - Firebase

**Real-Time Databases**

- evolving collections

**pull-based**  |  **push-based**
Wrap-Up
Direct Comparison

<table>
<thead>
<tr>
<th></th>
<th>Meteor</th>
<th>RethinkDB</th>
<th>Parse</th>
<th>Firebase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Poll-and-Diff</td>
<td>Oplog Tailing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scales with write TP</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Scales with no. of queries</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Composite queries (AND/OR)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Sorted queries</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Limit</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Offset</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
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

Low Latency

Less Processing
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.“
InvaliDB
Invalidating DB Queries

Real-Time Queries (Websockets)

Create
Update
Delete

Server

Pub-Sub

Fresh Caches

Pub-Sub
Baqend Real-Time Queries
Real-Time Decoupled

Keeps data up-to-date!
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
Baqend Real-Time Queries
Low Latency + Linear Scalability

Linear Scalability
- High throughput
- Linear scalability

Stable Latency Distribution
- Consistent and predictable latency

**Quaestor**: Query Web Caching for Database-as-a-Service Providers
VLDB ‘17
Programming Real-Time Queries
JavaScript API

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

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

query.resultStream(result => ...);
```
1. Conjure (conjure_re, 3840 followers) tweeted:
https://twitter.com/conjure_re/status/859767327570702336
Congress Saved the Science Budget—And That’s the Problem
https://t.co/UdrgNidAcK
https://t.co/xINjEpKZG

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のけいったくんがリブのやり取りしてる...

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 #TeamSCSCodingChallenge.
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 #TeamSCSCodingChallenge.
https://t.co/vx1o0WgJrZ #SCSchallenge
Accelerating Legacy Websites
Testing Future Performance

<table>
<thead>
<tr>
<th>Domains</th>
<th>Requests</th>
<th>Response Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>145</td>
<td>2.37 MB</td>
</tr>
</tbody>
</table>

Your Website


1016ms | 2.10x Faster | Speed Index | 484ms
216ms  | 3.22x Faster | Time To First Byte | 67ms
1201ms | 2.97x Faster | DOMContentLoaded | 404ms
2153ms | 1.42x Faster | FullyLoaded | 1511ms
1.7s   | 1.75x Faster | Last Visual Change | 1s

Your Website with Speed Kit

https://makefast-staging.com/open-in-new-tab

1.0
Page Speed Analyzer

Enter URL here...

Test the performance of your site!

Choose how to test:
- Region of client: USA, EU
- Mobile: NO, YES

Go
Speed Kit
Baqend Caching for Legacy Websites

Website with Snippet

Requests

Service Worker

Fast Requests

Baqend Service

Pull

Push

3rd Party Services

Existing Backend
Adding Speed Kit to a Site
Speed Kit
How to Use It

1. Configure Domain
Set which URLs Baqend should accelerate.
2. Include Code Snippet

Add the Speed Kit Service Worker to the website.
3. Requests Accelerated

Speed Kit routes the requests through Baqend’s CDN.
Speed Kit
Works Across Tech Stacks

Does it work for you? Try it: https://test.speed-kit.com/
Speed Kit
Works For Publishers

Does it work for you? Try it:
https://test.speed-kit.com/

kicker.de

<table>
<thead>
<tr>
<th>Your Website</th>
<th>Your Website with Speed Kit</th>
</tr>
</thead>
<tbody>
<tr>
<td>42 Domains</td>
<td>42 Domains</td>
</tr>
<tr>
<td>243 Requests</td>
<td>243 Requests</td>
</tr>
<tr>
<td>4.4 MB</td>
<td>4.4 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Time</th>
<th>Metric</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed Index</td>
<td>3332ms</td>
<td>4.11x Faster</td>
<td>810ms</td>
</tr>
<tr>
<td>Time To First Byte</td>
<td>638ms</td>
<td>10.13x Faster</td>
<td>63ms</td>
</tr>
<tr>
<td>DOMContentLoaded</td>
<td>5163ms</td>
<td>4.91x Faster</td>
<td>1051ms</td>
</tr>
<tr>
<td>FullyLoaded</td>
<td>13850ms</td>
<td>3.67x Faster</td>
<td>3770ms</td>
</tr>
</tbody>
</table>

3.5s Last Visual Change
3.98x Faster

Print Report
Speed Kit
Works For Landing Pages

molsoncoors.com

Does it work for you? Try it:
https://test.speed-kit.com/

Your Website

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed Index</td>
<td>1493ms</td>
</tr>
<tr>
<td>Time To First Byte</td>
<td>298ms</td>
</tr>
<tr>
<td>DOMContentLoaded</td>
<td>820ms</td>
</tr>
<tr>
<td>FullyLoaded</td>
<td>1753ms</td>
</tr>
<tr>
<td>Last Visual Change</td>
<td>1.6s</td>
</tr>
</tbody>
</table>

Your Website with Speed Kit

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed Index</td>
<td>476ms</td>
</tr>
<tr>
<td>Time To First Byte</td>
<td>2ms</td>
</tr>
<tr>
<td>DOMContentLoaded</td>
<td>248ms</td>
</tr>
<tr>
<td>FullyLoaded</td>
<td>486ms</td>
</tr>
<tr>
<td>Last Visual Change</td>
<td>0.5s</td>
</tr>
</tbody>
</table>

1.6

Open in new tab

Print Report

Unexpected result?
Speed Kit
Works For E-Commerce

Does it work for you? Try it:
https://test.speed-kit.com/

alibaba.com

[Comparison of domains, requests, and response size between the original and Speed Kit-optimized websites]

Print Report
Platform

- Platform for building (Progressive) **Web Apps**
- **15x** Performance Edge
- Faster **Development**

Speed Kit

- Turns Existing Sites into **PWAs**
- **50-300% Faster** Loads
- **Offline** Mode

Does it work for you? Try it: [https://test.speed-kit.com/](https://test.speed-kit.com/)
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**

Learn more at [blog.baqend.com](http://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