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

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

Universität Hamburg

Wolfram Wingerath

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?

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 (21/4)
3.

Diagram:

- Room A
- Room B
- Room C

Legend:

- Red: Erik (5/10)
- Green: Wolle (21/4)
- Orange: Unknown person

Graph:

- X-axis: 0 to 25
- Y-axis: 0 to 15

Points:

- (5, 5) - Red
- (10, 10) - Green
- (5, 15) - Orange
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
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

[Diagram of MongoDB cluster (3 shards)]
Oplog Tailing
Tapping into the Oplog

MongoDB cluster (3 shards)

Primary A  Primary B  Primary C

Oplog broadcast

App server

CRUD

query (when in doubt)
monitor oplog
push relevant events

Oplog broadcast

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
• **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 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
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

Database Management
Real-Time Databases
Data Stream Management
Stream Processing

static collections
static collections
 persistent/ephemeral streams
 ephemeral streams

pull-based
pull-based
push-based
push-based

ORACLE
PostgreSQL
MySQL
IBM

ME T E R
RethinkDB
Parse
Firebase

_PIPELINE
DB

STORM
samza
Flink
Spark

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

Database Management
Real-Time Databases
Data Stream Management
Stream Processing

static collections
evolving collections
 persistent/ephemeral streams
ephemeral streams

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>
<th>Baqend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scales with write TP</strong></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Scales with no. of queries</strong></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Composite queries (AND/OR)</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Sorted queries</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Limit</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>✓</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
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.“
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
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

Stable Latency Distribution

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 => ...);
```

Static Query

Real-Time Query
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/Ud5nNldk 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 #teamscc, and #scs...
https://twitter.com/LisaMSchmid/status/859767317311500290
Congrats to Matthew Kent, winner of the 26th TeamSCSCoding Challenge.
https://t.co/vx1oWgJrZ #SCSchallenge

5. Brian Martin Larson (Brian_Larson, 40 followers) tweeted on #teamscc, and...
https://twitter.com/Brian_Larson/status/859767317303001089
Congrats to Matthew Kent, winner of the 26th TeamSCSCoding Challenge.
https://t.co/vx1oWgJrZ #SCSchallenge
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**

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
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