Push vs. Pull

The Future of Real-Time Databases in the Cloud

Wolfram Wingerath
ww@baqend.com
December 10, SCDM 2018, Seattle
About me
Wolfram Wingerath

PhD Thesis & Research

Research:
- Real-Time Databases
- Stream Processing
- NoSQL & Cloud Databases
- ...

Practice:
- Backend-as-a-Service
- Web Caching
- Real-Time Database
- ...

www.baqend.com

Universität Hamburg
Outline

- A Small History Lesson
- The Problem With Traditional Databases
- Real-Time Databases to the Rescue!

Push-Based Data Access
Why Real-Time Databases?

Real-Time Databases
System survey

Discussion
What are the bottlenecks?

Future Directions
Scalability & Use Cases
A Short History of Data Management

Hot Topics Through The Ages

Relational Databases
- Entity-Relationship Model
- Triggers
- Ingres
- SQL Standard
- Starburst
- Telegraph
- HiPAC
- PostgreSQL

1970

Active Databases
- Relational Model
- System R
- PostgreSQL

1980

CEP & Streams
- MapReduce
- Spark
- STREAM
- Bigtable
- Samza
- Meteor
- Flink
- Storm
- GFS
- Dynamo
- Aurora & Borealis

1990

Big Data & NoSQL
- PostgresQL
- Rapide
- Starburst
- Telegraph

2000

Real-Time Databases
- Baqend
- Baqend
-Firebase
- RethinkDB
- Meteor
- Flink
- Samza
- Spark
- Bigtable
- GFS
- Dynamo
- Aurora & Borealis

2010

today
Traditional Databases
The Problem: No Request – No Data!

What’s the current state?

Periodic Polling for query result maintenance:
→ inefficient
→ slow
Real-time Databases
Always Up-to-Date With Database State

Real-Time Queries for query result maintenance:
- efficient
- fast
Real-Time Query Maintenance
Matching Every Query Against Every Update

→ Potential *bottlenecks*:
  - Number of queries
  - Write throughput
  - Query complexity

Similar processing for:
  - Triggers
  - ECA rules
  - Materialized views
Outline

- Push-Based Data Access
  Why Real-Time Databases?

- Real-Time Databases
  System survey

- Discussion
  What are the bottlenecks?

- Future Directions
  Scalability & Use Cases

- 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

![Diagram of app servers and database with arrows indicating CRUD operations and monitoring incoming writes. Repeat query every 10 seconds, forward CRUD, and monitor incoming writes.]
Oplog Tailing
Basics: MongoDB Replication

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

```
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

monitor oplog

push relevant events

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](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!**
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)

Illustration taken from: Frank van Puffelen, *Have you met the Realtime Database?* (2016)
“Scale to around 100,000 concurrent connections and 1,000 writes/second in a single database. Scaling beyond that requires sharding your data across multiple databases.”
Firebase

Firestore: New Model

documents

collections

references

Firebase
Firestore: New Model

finer access granulates

tree-like structure

Firebase

Firestore: Summary

- More specific data selection
- Logical AND for some filter combinations

... But:
- Still Limited Expressiveness
  - No logical OR
  - No logical AND for many filter combinations
  - No content-based search (regex, full-text search)
- Still Limited Write Throughput:
  - 500 writes/s per collection
  - 1 writes/s per document

Firebase, Firestore: Quotas and Limits (2018)
https://firebase.google.com/docs/firestore/quotas (2018-03-10)
Honorable Mentions
Other Systems With Real-Time Features

- realm
- GRAPHCOOL
- rapid.io (BETA)
- CouchDB
- OrientDB
- MongoDB
Outline

Push-Based Data Access
Why Real-Time Databases?

Real-Time Databases
System survey

Discussion
What are the bottlenecks?

Future Directions
Scalability & Use Cases

• System Classification:
  • Databases
  • Real-Time Databases
  • Stream Management
  • Stream Processing
  • Side-by-Side Comparison
Wrapup & Discussion
Data Management Overview

DBMS vs. Real-Time DB vs. Stream Management

Database Management

- static collections
- pull-based

Real-Time Databases

- evolving collections

Data Stream Management

- persistent/ephemeral streams
- push-based
# Real-Time Database Comparison

<table>
<thead>
<tr>
<th></th>
<th><strong>METEOR</strong></th>
<th><img src="https://example.com/rethinkdb" alt="RethinkDB" /></th>
<th><img src="https://example.com/parse" alt="Parse" /></th>
<th><img src="https://example.com/firebase" alt="Firebase" /></th>
<th><img src="https://example.com/baqend" alt="BaQend" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Poll-and-Diff</strong></td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Log Tailing</strong></td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td><strong>Unknown</strong></td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
</tr>
<tr>
<td><strong>2-D Partitioning</strong></td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Write Scalability</strong></td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td><strong>Read Scalability</strong></td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Composite Filters (AND/OR)</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td><img src="https://example.com/question-mark" alt="?" /> (100k connections)</td>
</tr>
<tr>
<td><strong>Sorted Queries</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td><img src="https://example.com/question-mark" alt="?" /> (AND In Firestore)</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><img src="https://example.com/question-mark" alt="?" /> (value-based)</td>
</tr>
<tr>
<td><strong>Self-Maintaining Queries</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td><strong>Event Stream Queries</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?

- Future Directions
  Scalability & Use Cases

- Performance & Scalability
- Query Expressiveness
- Use Cases
  - Real-Time Apps
  - Query Caching
- Summary
Making Real-Time Databases Scale
Baqend Real-Time Queries
Real-Time Decoupled

Keeps data up-to-date!
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

Subscripton!
SELECT * FROM posts WHERE tags CONTAINS 'NoSQL'

Write op!
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
Bagend 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

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

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

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

2. Yuuu_key (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 がリップのやり取りしてる...

3. Whitney Shackley (benschneids11, 5 followers) tweeted:
https://twitter.com/benschneids11/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/vx1o0WgJr/Z #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.
https://t.co/vx1o0WgJr/Z #SCSchallenge
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.“
Summary
Real-Time Databases: Major challenges

Scalability:
- Handle increasing throughput
- Handle additional queries

Expressiveness:
- Content-based search? Composite filters?
- Ordering? Limit? Offset?

Legacy Support:
- Real-time queries for existing databases?
- Decouple OLTP from real-time workloads?
Our Related Publications

Book, Papers, Articles & Tutorials:

<table>
<thead>
<tr>
<th>Title</th>
<th>Publisher/Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quaestor: Query Web Caching for Database-as-a-Service Providers</td>
<td>VLDB ‘17</td>
</tr>
<tr>
<td>NoSQL Database Systems: A Survey and Decision Guidance</td>
<td>SummerSOC ‘16</td>
</tr>
<tr>
<td>Real-time stream processing for Big Data</td>
<td>Information Technology 58 (2016)</td>
</tr>
<tr>
<td>The Case For Change Notifications in Pull-Based Databases</td>
<td>BTW ‘17</td>
</tr>
<tr>
<td>Real-Time &amp; Stream Data Management: Push-Based Data in Research &amp; Practice</td>
<td>Springer 2019</td>
</tr>
<tr>
<td>Real-Time Data Management for Big Data</td>
<td>EDBT 2018</td>
</tr>
<tr>
<td>Scalable Push-Based Real-Time Queries on Top of Pull-Based Databases</td>
<td>PhD thesis, Wolfram Wingerath, 2018</td>
</tr>
<tr>
<td>Low Latency for Cloud Data Management</td>
<td>PhD thesis, Felix Gessert, 2018</td>
</tr>
</tbody>
</table>

Blog Posts:

<table>
<thead>
<tr>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-Time Databases Explained: Why Meteor, RethinkDB, Parse and Firebase Don’t Scale</td>
<td><a href="https://medium.com/p/822ff87d2f87">https://medium.com/p/822ff87d2f87</a></td>
</tr>
</tbody>
</table>

Learn more at [blog.baqend.com](http://blog.baqend.com)!
Thank you

wingerath@informatik.uni-hamburg.de

Blog: blog.baqend.com
Slides: slides.baqend.com

@baqendcom