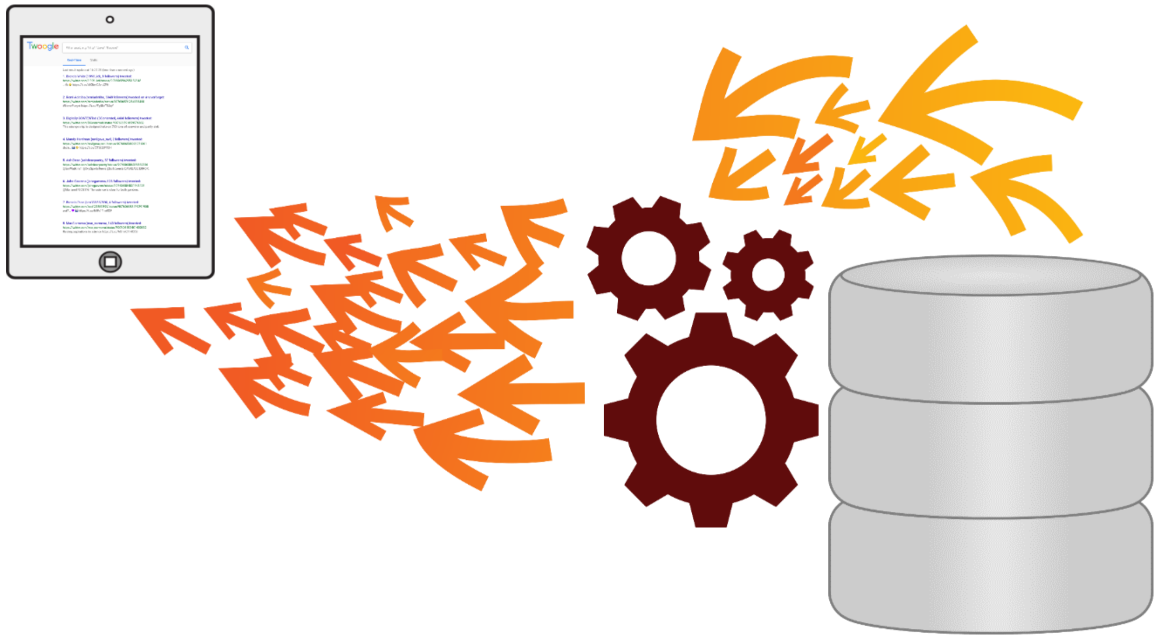
Scalable Push-Based Real-Time Queries
on Top of Pull-Based Databases



Universität Hamburg

Wolfram Wingerath

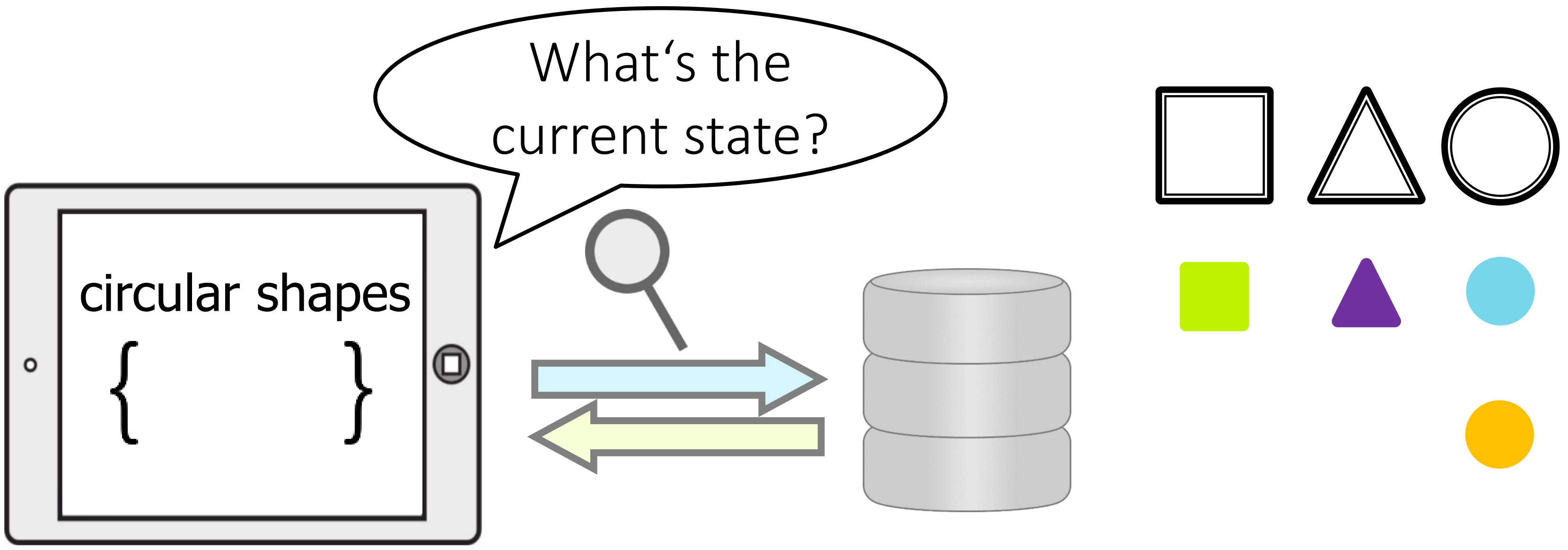
May 8, 2019, Disputation, Hamburg

Outline

- **Pull vs. Push**
  - Traditional DB Queries
  - Why Real-Time Queries?
  - How to Provide Them?
- **The Primary Challenges**
  - C_1 Scalability
  - C_2 Expressiveness
  - C_3 Legacy Support
  - C_4 Abstract API
- **Research Question**

Traditional Databases

The Problem: No Request – No Data!



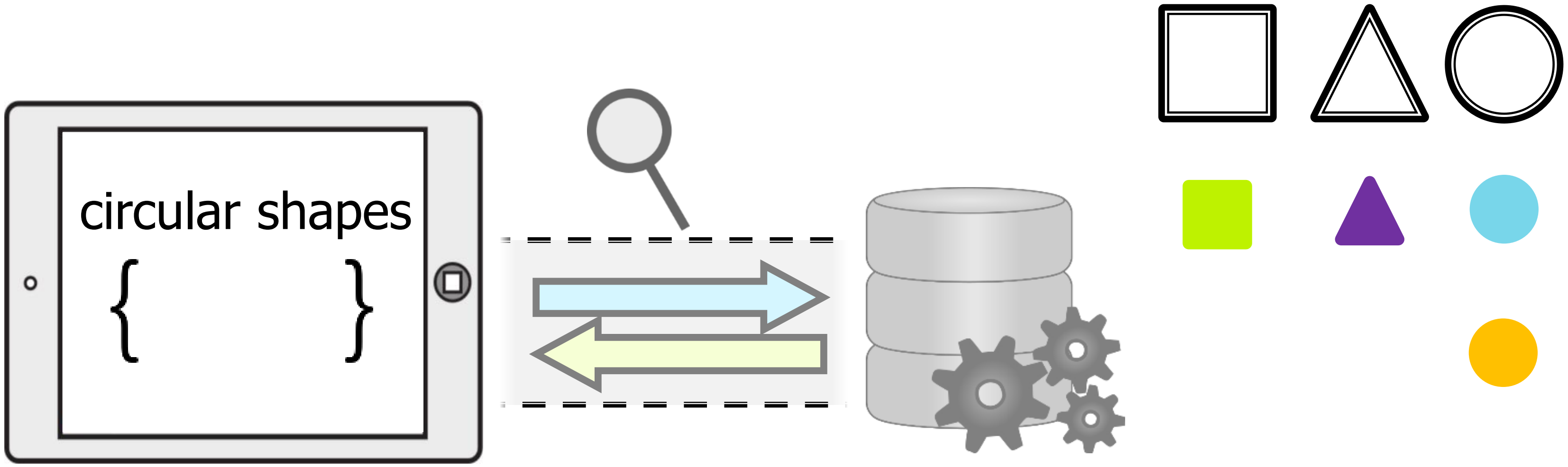
Periodic Polling for query result maintenance:

→ 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

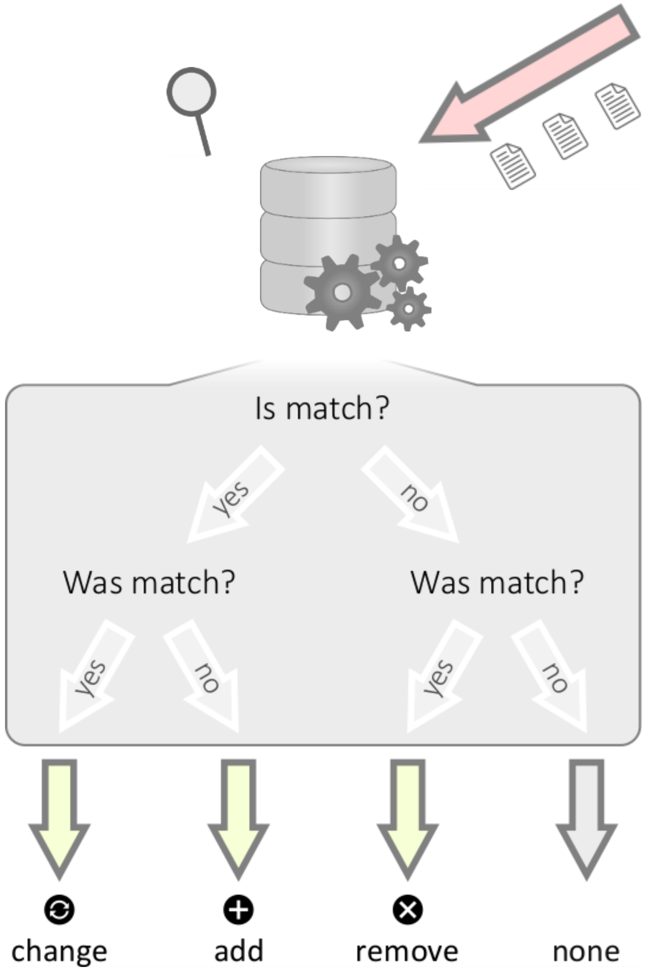
Matching Every Query Against Every Update

→ Potential *bottlenecks:*

- *Number of queries*
- *Write throughput*
- *Query complexity*

Similar processing for:

- Triggers
- ECA rules
- Materialized views



Challenges

Real-Time Databases: Major challenges

C_1: Scalability:

- Handle additional queries
- Handle increasing throughput

C_2: Expressiveness:

- Content search? Composite filters?
- Ordering? Limit? Offset?

Research Question: *„How can expressive push-based real-time queries be implemented on top of an existing pull-based database in a scalable and generic manner?“*

C_3: Legacy Support:

- Real-time queries for *existing databases*
- *Decouple* OLTP from real-time workloads

C_4: Abstract API

- Data independence
- Self-maintaining queries

Outline

Problem Statement
Intro & Research Question

Related Work
State of the Art & Open Issues

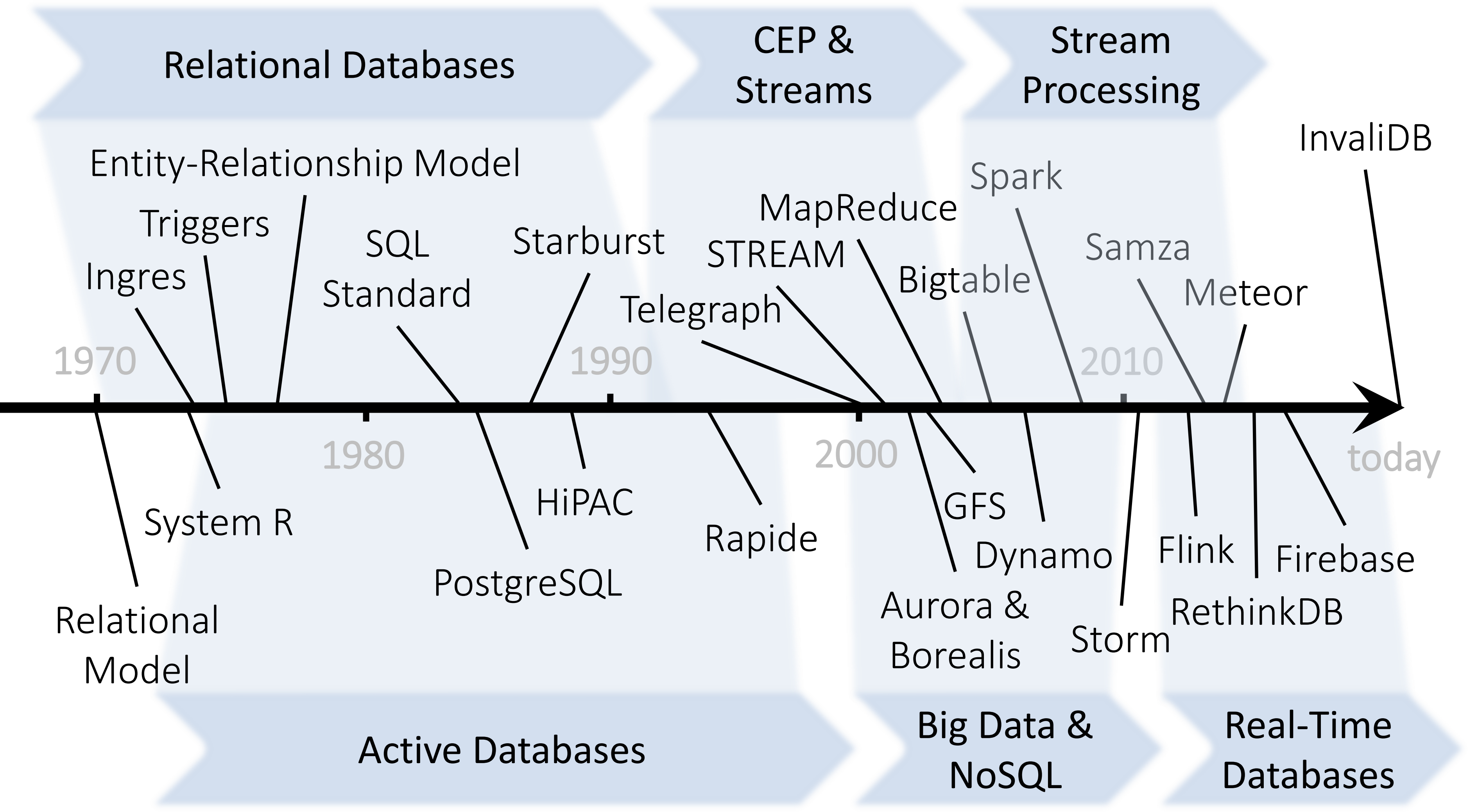
A Scalable RTDB Design
InvaliDB: Concept & Prototype

Discussion
Applications & Outlook

- **Data Management Classes**
  - Historical Overview
  - 4-Part Categorization
- **Real-Time Databases**
  - Poll-and-Diff
  - Oplog Tailing
- **System Comparison**
  - Meteor
  - RethinkDB
  - Parse
  - Firebase
  - InvaliDB

A Short History of Data Management

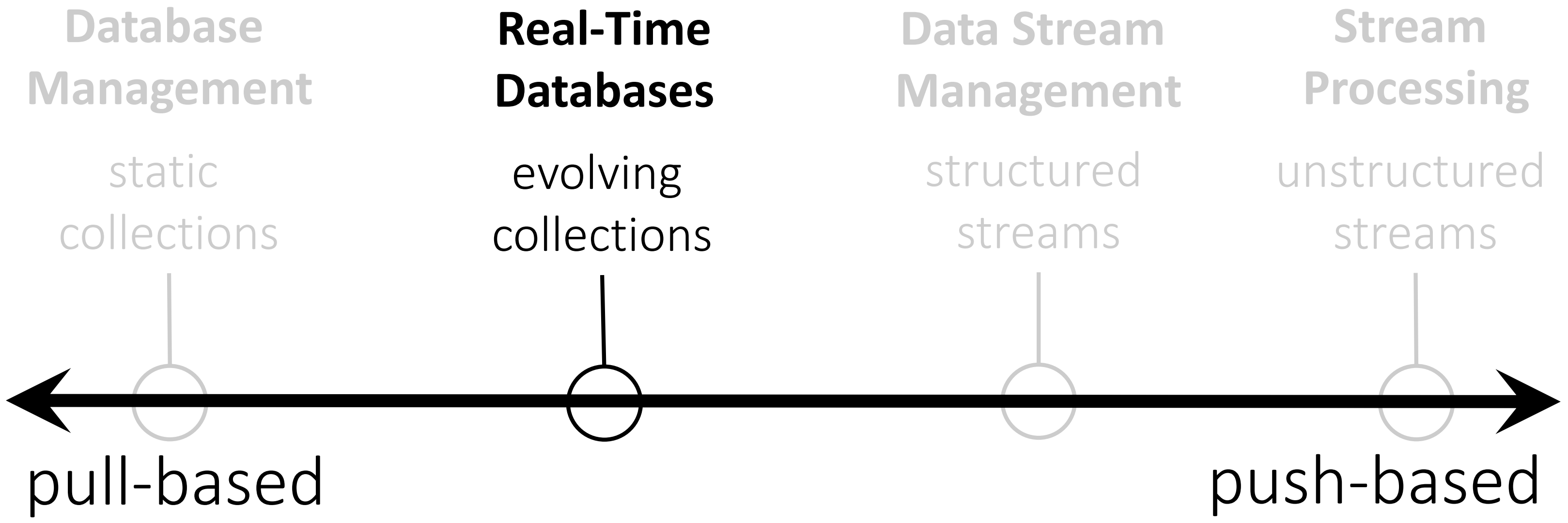
Hot Topics Through The Ages



[WRG19, WGW+18]

Data Management Systems

A High-Level Categorization

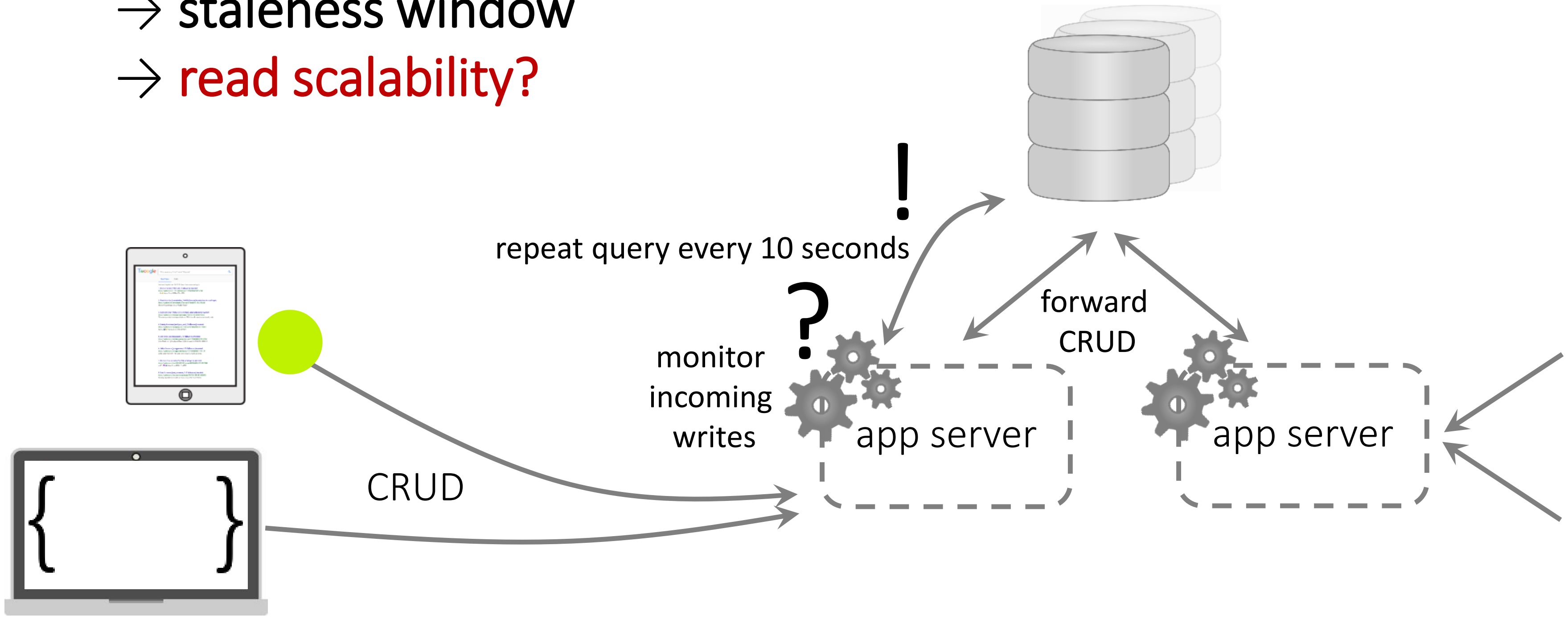


[WRG19, WGW+18]

Typical Maintenance Mechanisms (1/2)

Poll-and-Diff

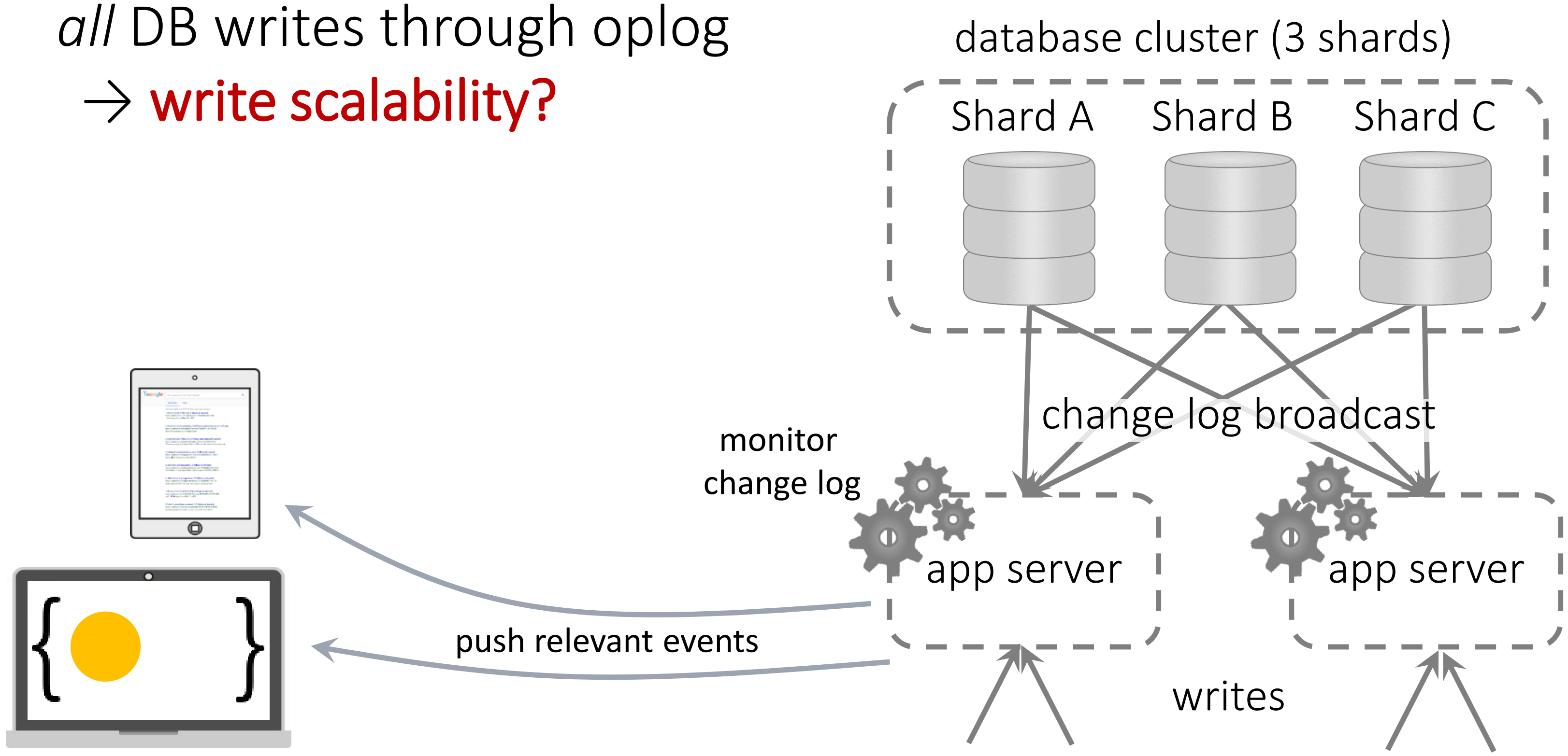
- **Local change monitoring**: app servers detect local changes
  → *incomplete* in multi-server deployment
- **Poll-and-diff**: global changes are discovered through polling
  → **staleness window**
  → **read scalability?**



[GWR17, Win17]

Typical Maintenance Mechanisms (2/2)
Change Log Tailing

- *Every* application server receives *all* DB writes through oplog
  → write scalability?



[GWR17, Win17]

Real-Time Database Comparison

	METEOR	RethinkDB	Parse	Firebase	InvaliDB
	Poll-and-Diff	Change Log Tailing		Unknown	2-D Partitioning
Write Scalability	✓	✗	✗	✗	✓
Read Scalability	✗	✓	✓	? (100k connections)	✓
Composite Filters (AND/OR)	✓	✓	✓	○ (AND In Firestore)	✓
Sorted Queries	✓	✓	✗	○ (single attribute)	✓
Limit	✓	✓	✗	✓	✓
Offset	✓	✗	✗	○ (value-based)	✓
Self-Maintaining Queries	✓	✗	✗	✗	✓
Event Stream Queries	✓	✓	✓	✓	✓

[GWR17, Win17]

Outline

Problem Statement
Intro & Research Question

Related Work
State of the Art & Open Issues

A Scalable RTDB Design
InvaliDB: Concept & Prototype

Discussion
Applications & Outlook

- **System Model & Overview**
  - Query Subscription
  - Write Ingestion
  - Change Propagation
- **Real-Time Query Processing**
  - Two-Dimensional Workload Partitioning
  - Processing Stages
- **Performance Evaluation**
  - Read Scalability
  - Write Scalability
  - Multi-Tenancy

InvaliDB: A Scalable Real-Time Database Design

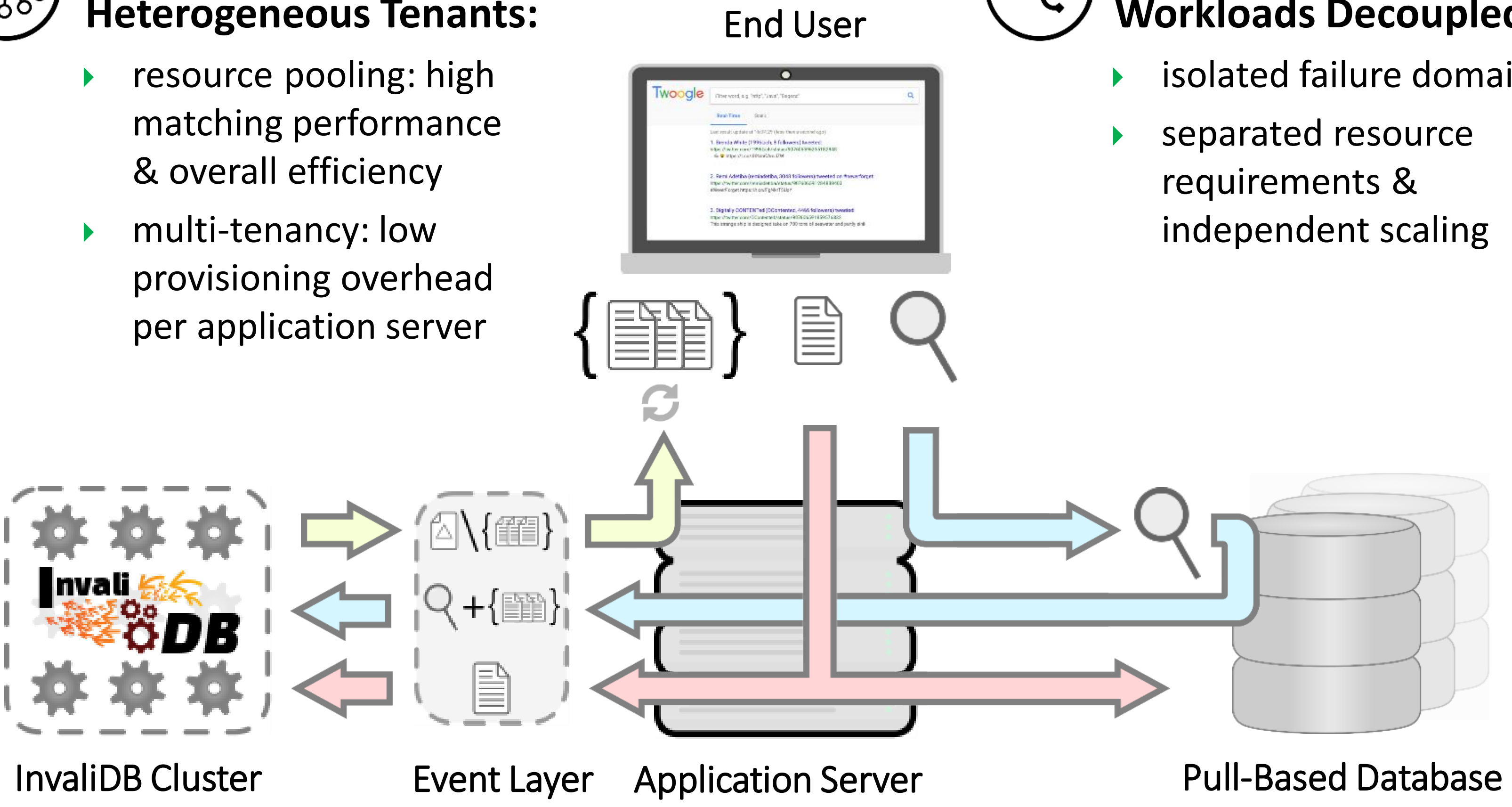
System Model & Overview

Realtime-as-a-Service For Heterogeneous Tenants:

- resource pooling: high matching performance & overall efficiency
- multi-tenancy: low provisioning overhead per application server

Real-Time & OLTP Workloads Decoupled:

- isolated failure domains
- separated resource requirements & independent scaling



[WGF+17, GSW+17]

InvaliDB: A Scalable Real-Time Database Design

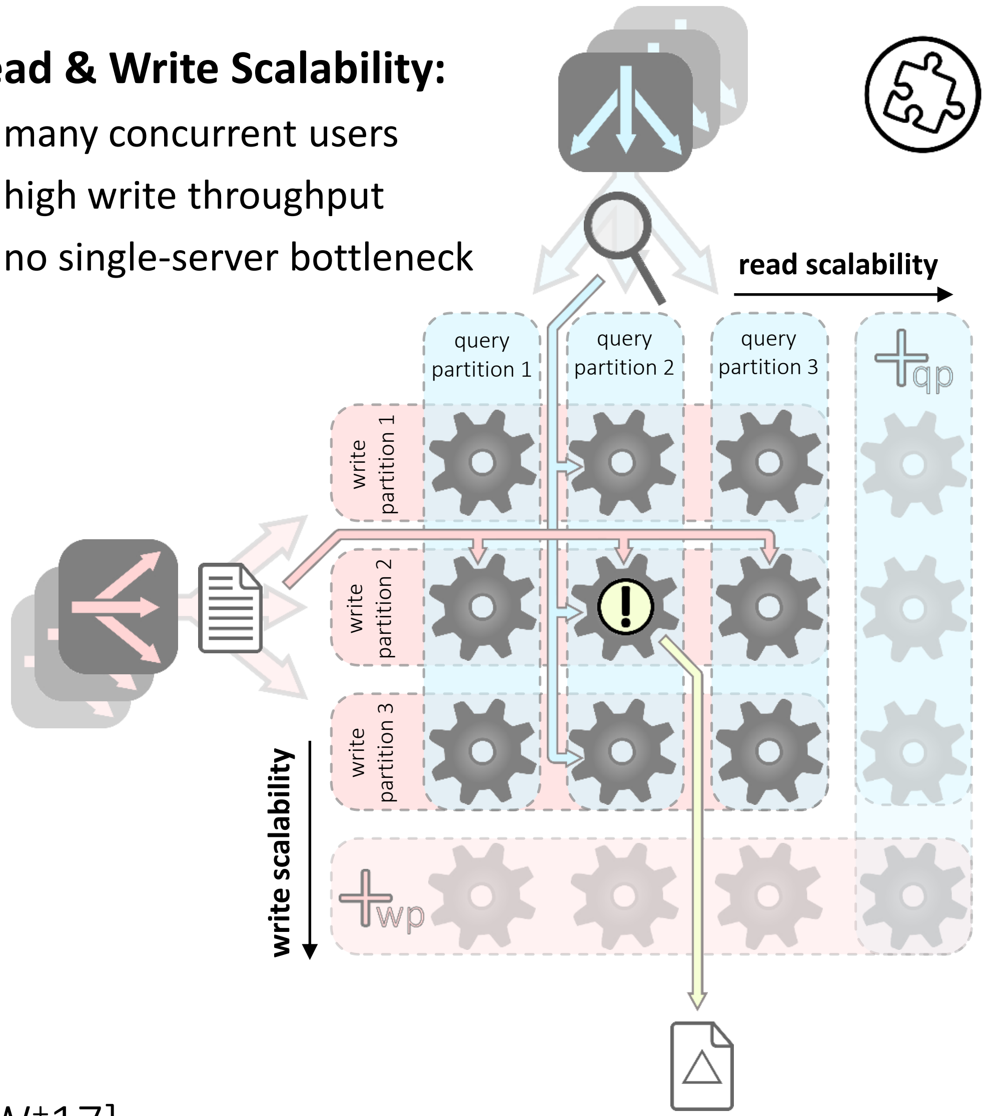
Two-Dimensional Workload Partitioning

Read & Write Scalability:

- many concurrent users
- high write throughput
- no single-server bottleneck

Pluggable Query Engine:

- legacy-compatibility
- multi-tenancy across databases



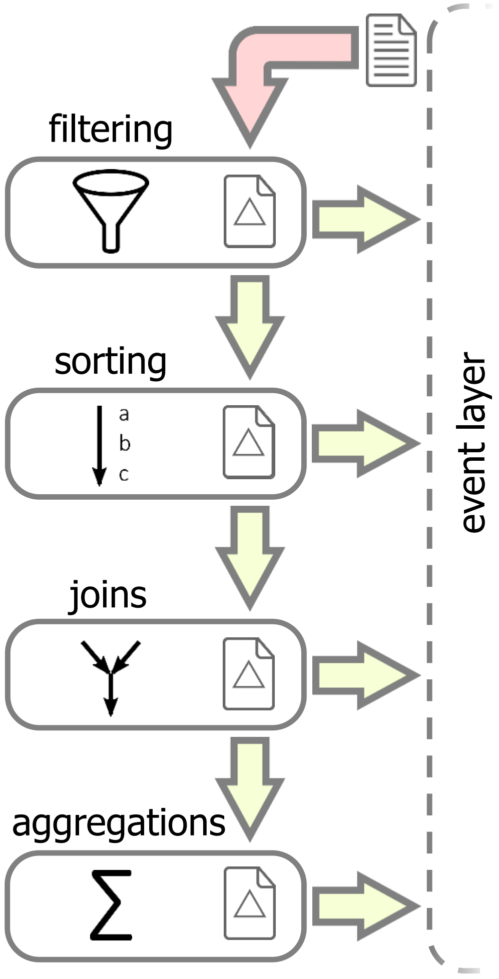
[WGF+17, GSW+17]

InvaliDB: A Scalable Real-Time Database Design

Staged Real-Time Query Processing

Change notifications go through different 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



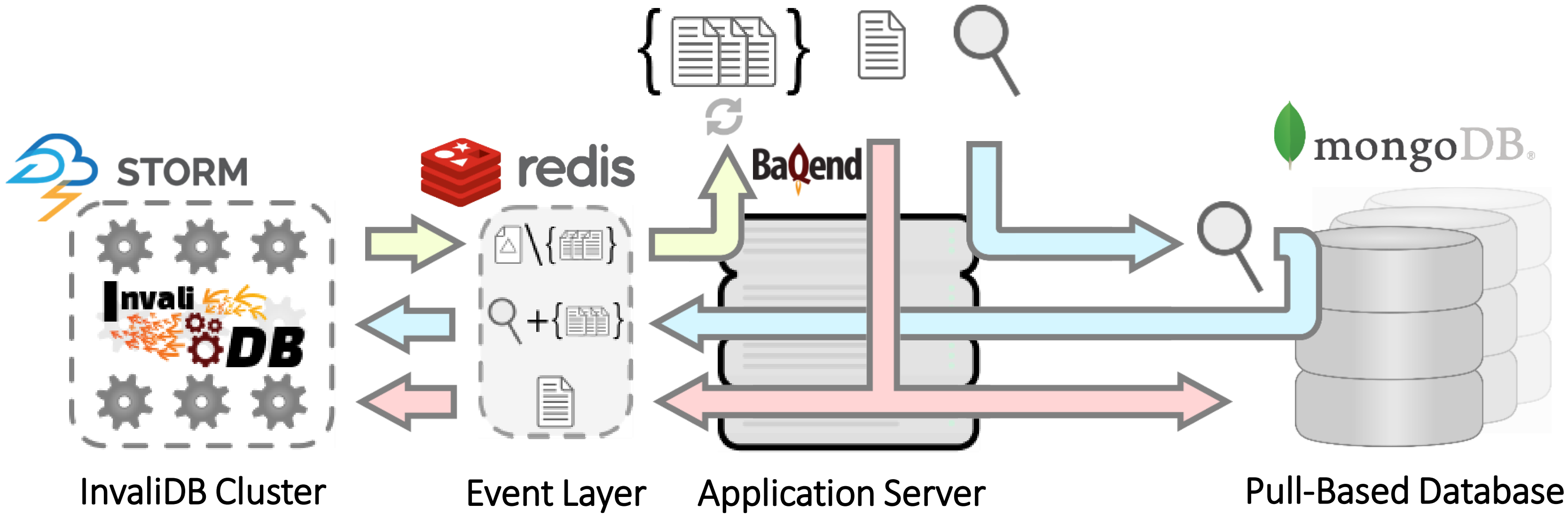
[WGF+17, GSW+17]

Evaluation: Performance & Scalability
Prototype Implementation

Query Processing
- low latency
- customizability
- tried & tested

Event Layer
- low latency
- high per-node throughput
- ease of deployment

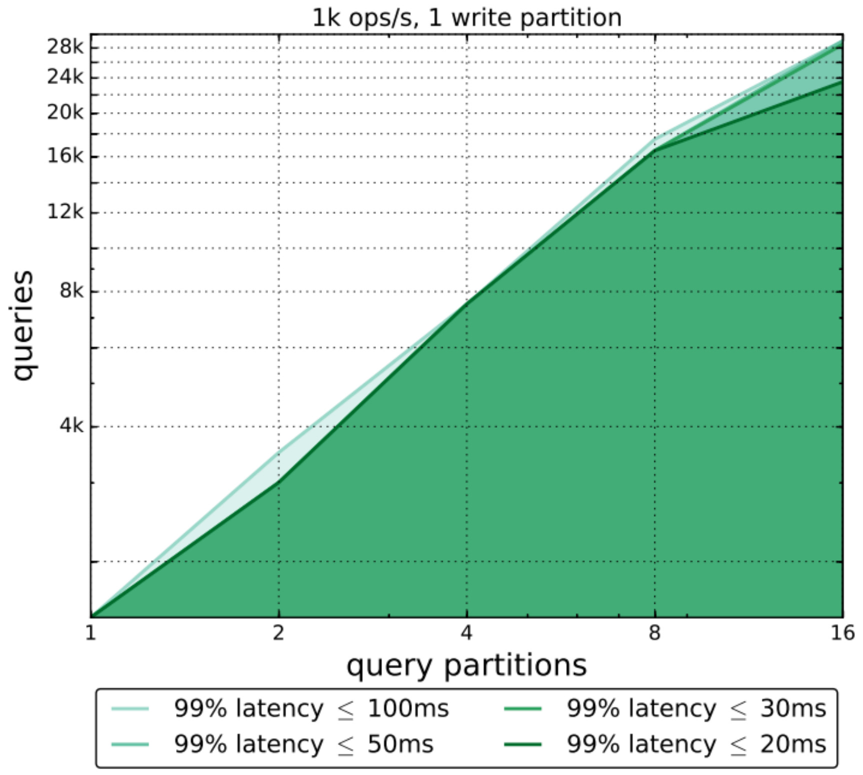
Database
- typical RTDB expressiveness
- typical NoSQL datastore
- wildly popular

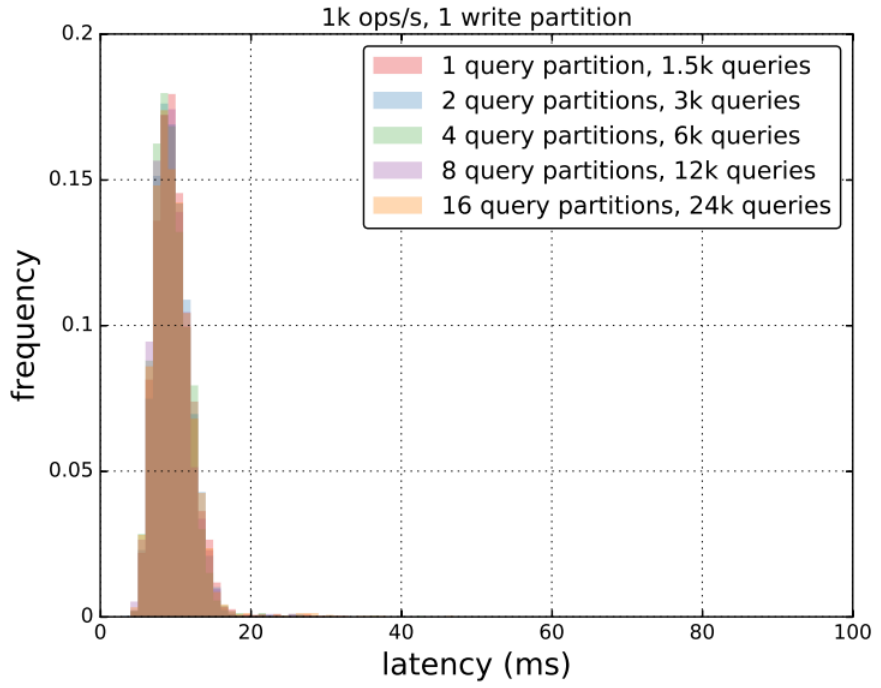


[WGF+17, GSW+17, Win16, WGFR16, GWFR16]

Linear Read Scalability

Sustainable Queries at 1k Writes per Second

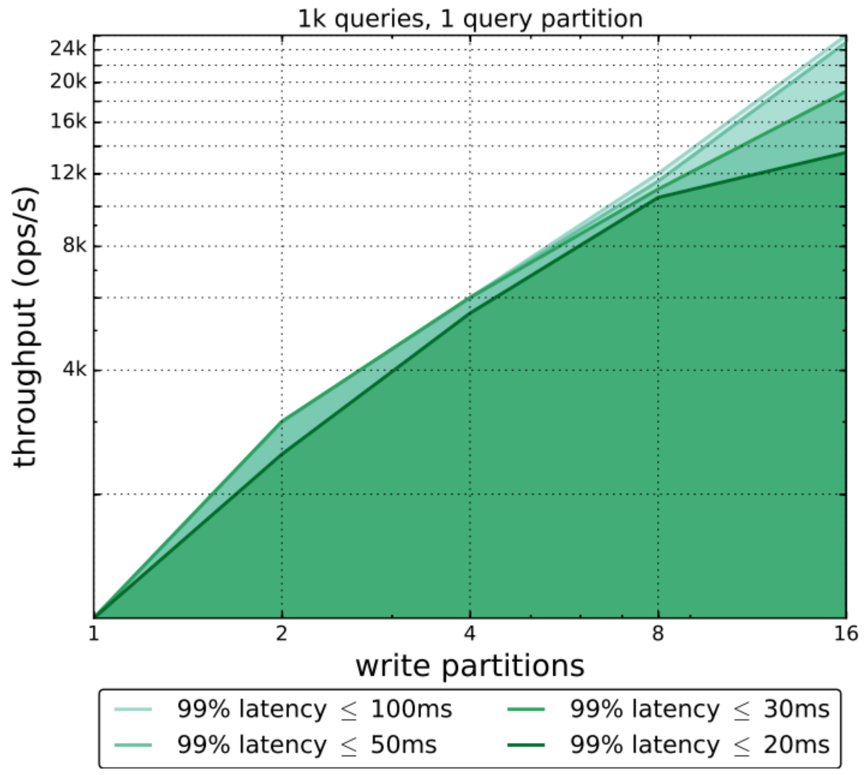


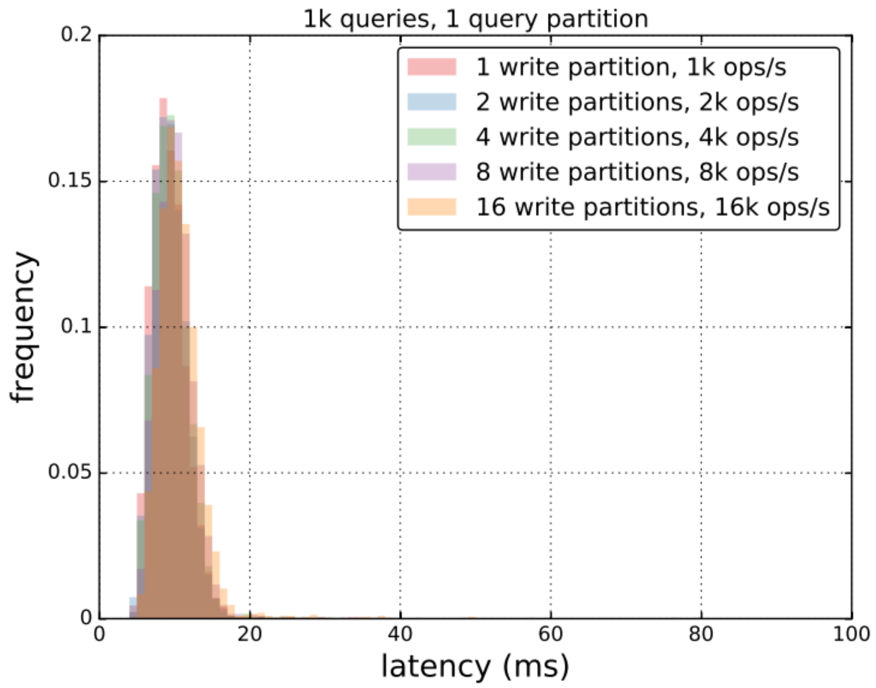


1.5 mio. matching ops/s per node

Linear Write Scalability

Sustainable Throughput With 1k Active Queries





1 mio. matching ops/s
per node

Outline

Problem Statement
Intro & Research Question

Related Work
State of the Art & Open Issues

A Scalable RTDB Design
InvaliDB: Concept & Prototype

Discussion
Applications & Outlook

- **Application Scenarios**
  - Real-Time Queries
  - Query Caching
- **Future Work**
- **Publications**
  - Articles & Papers
  - Tutorials
  - Book
- **Contributions**
  - Data Management Categorization
  - InvaliDB: Design & Impl.
  - Proof of Practicality

Use Case 1: Real-Time Queries

An Easy-to-Use JavaScript API

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

Pull-Based Query

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

Google

Real-Time Query

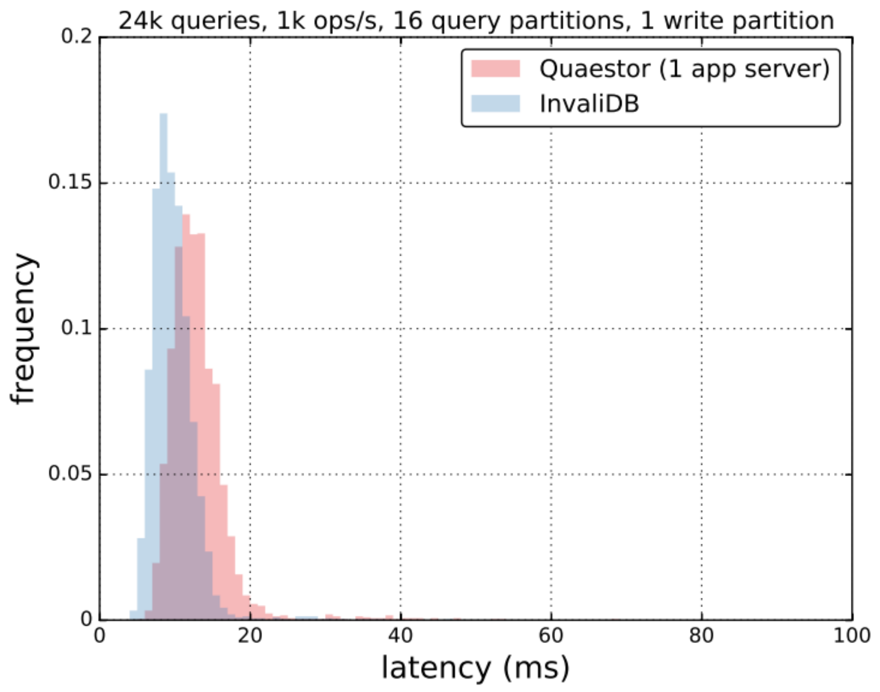
```
query.resultStream(result => ...);
```

Twoogle

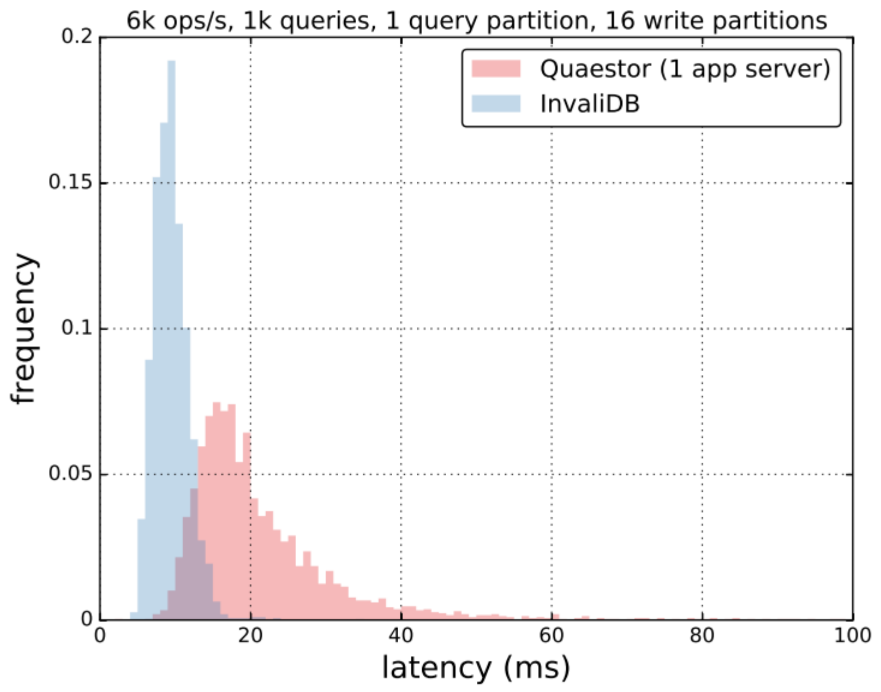
[WGR19a]

Baqend Real-Time Query Performance

Low Overhead, High Efficiency



Read-Heavy Workload

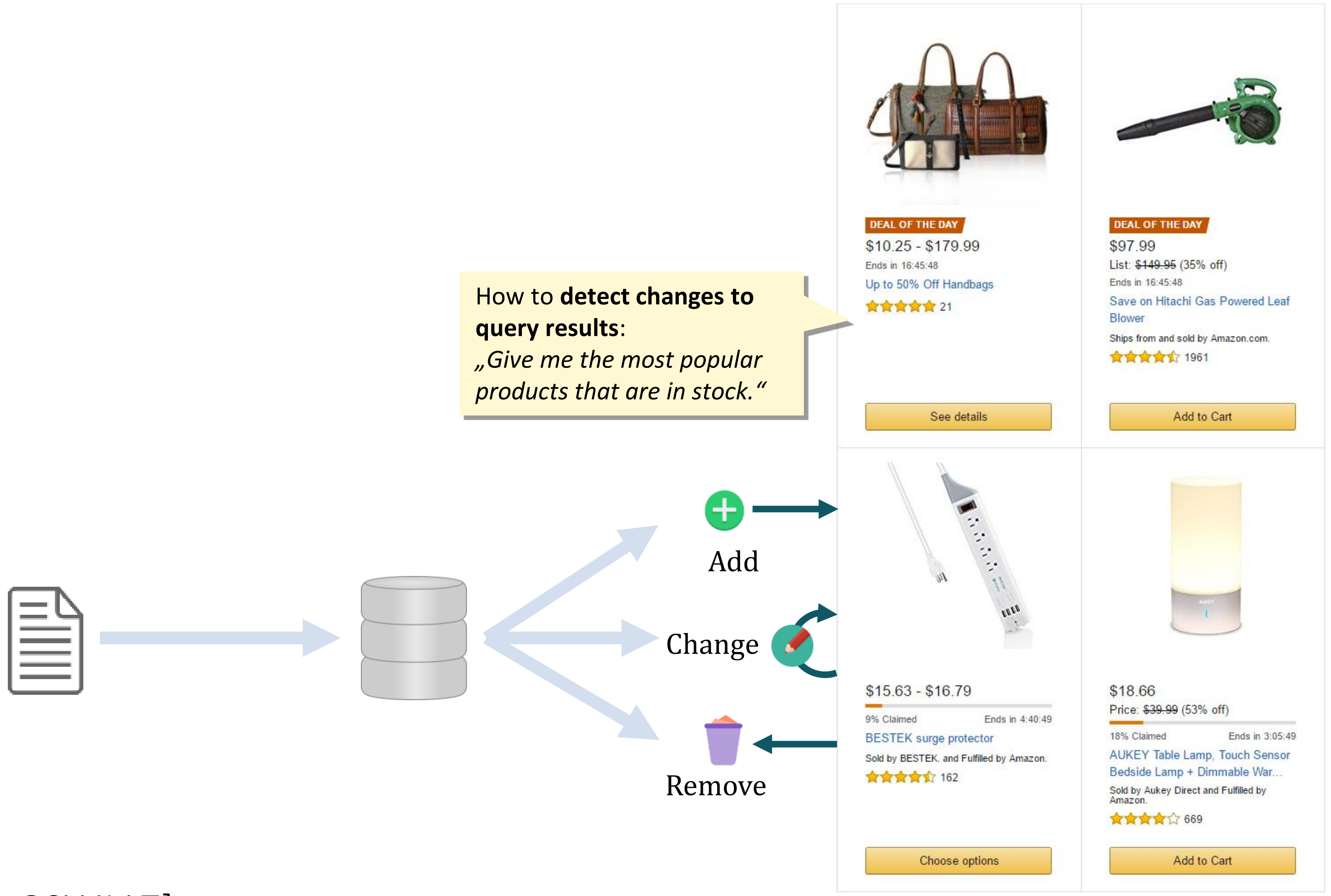


Write-Heavy Workload

Use Case 2: Consistent Query Caching

InvaliDB For Invalidating DB Queries

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



DEAL OF THE DAY
$10.25 - $179.99
Ends in 16:45:48
Up to 50% Off Handbags
21
See details

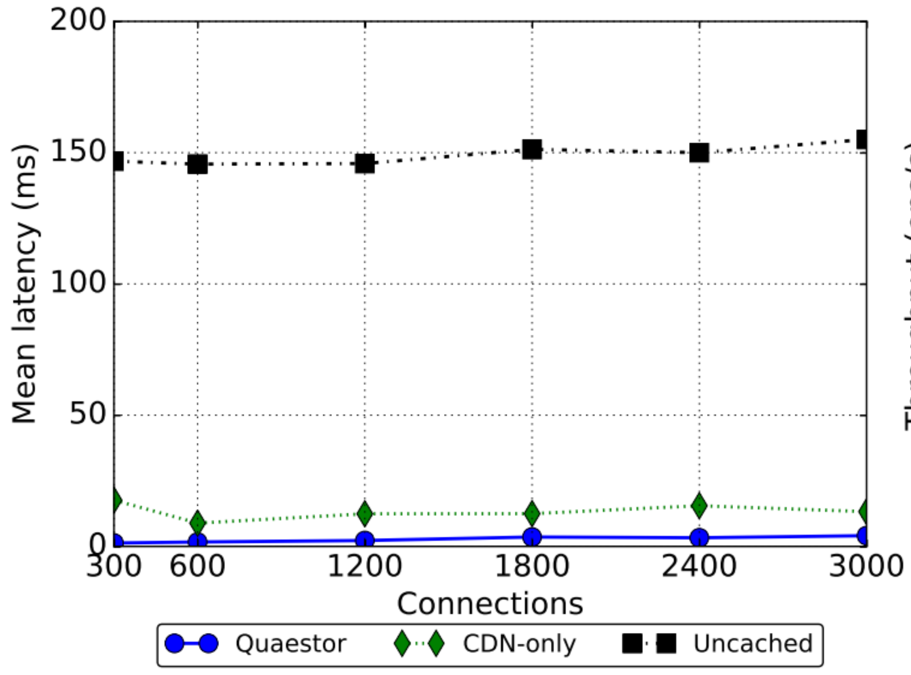
DEAL OF THE DAY
$97.99
List: ~~$149.95~~ (35% off)
Ends in 16:45:48
Save on Hitachi Gas Powered Leaf Blower
Ships from and sold by Amazon.com.
1961
Add to Cart

$15.63 - $16.79
9% Claimed
Ends in 4:40:49
BESTEK surge protector
Sold by BESTEK. and Fulfilled by Amazon.
162
Choose options

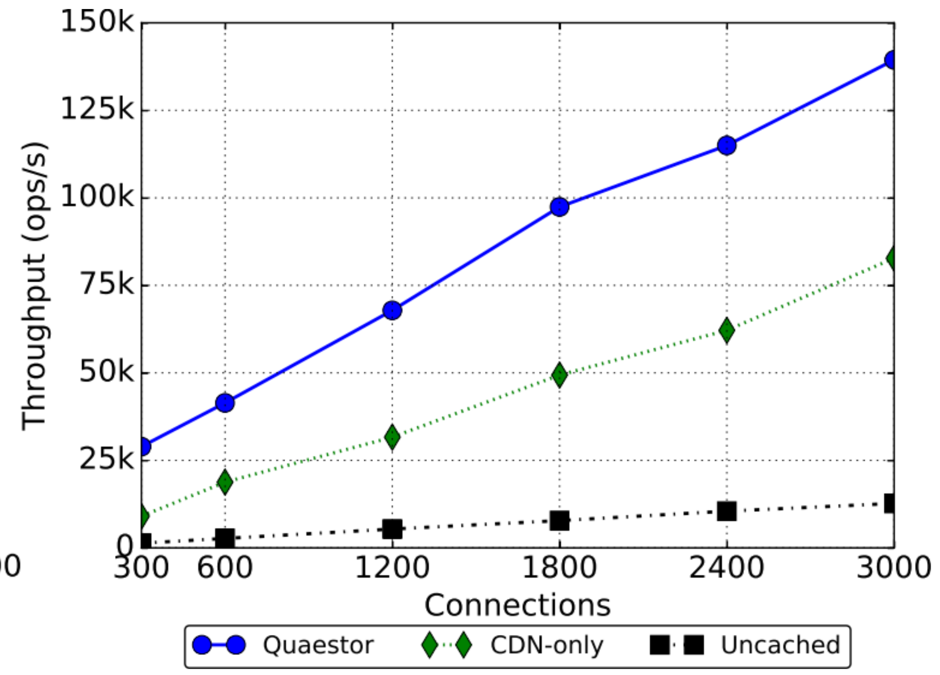
$18.66
Price: ~~$39.99~~ (53% off)
18% Claimed
Ends in 3:05:49
AUKEY Table Lamp, Touch Sensor Bedside Lamp + Dimmable War...
Sold by Aukey Direct and Fulfilled by Amazon.
669
Add to Cart

[WGF+17, GSW+17]

Query Caching
Improving Pull-Based Query Performance



Latency



Throughput

[GSW+17]

Future Research

Open Challenges & Follow-Up Work

Extending Semantics

- Additional Languages, Joins & Aggregations
- Transactions
- Stream-Based Queries & CEP

Trade-Offs & Optimizations

- Failure Transparency
- Deployment & Adaptive Scaling
- Client Performance

Exploring New Use Cases

- Reactive & Collaborative (Mobile) Apps
- Enhancing UI in Existing Applications
- Augmenting Cache Coherence Schemes

Publications

DMC 2014, Datenbank-Spektrum, BTW 2015

[GFW⁺ 14]	Gessert, Felix; Friedrich, Steffen; Wingerath, Wolfram; Schaarschmidt, Michael; Ritter, Norbert: *Towards a Scalable and Unified REST API for Cloud Data Stores*, Informatik 2014 (**DMC 2014**)
[FWGR14]	Friedrich, Steffen; Wingerath, Wolfram; Gessert, Felix; Ritter, Norbert: *NoSQL OLTP Benchmarking: A Survey*, Informatik 2014 (**DMC 2014**)
[WFR15]	Wingerath, Wolfram; Friedrich, Steffen; Ritter, Norbert: *BTW 2015 – Jubiläum an der Waterkant*. In: **Datenbank-Spektrum** 15 (2015)
[SRS⁺15]	Seidl, Thomas (ed.); Ritter, Norbert (ed.); Schöning, Harald (ed.); Sattler, Kai-Uwe (ed.); Härder, Theo (ed.); Friedrich, Steffen (ed.); Wingerath, Wolfram (ed.): Datenbanksysteme für Business, Technologie und Web (BTW 2015) – *Konferenzband*, **BTW 2015**
[WFGR15]	Wingerath, Wolfram; Friedrich, Steffen; Gessert, Felix; Ritter, Norbert: *Who Watches the Watchmen? On the Lack of Validation in NoSQL Benchmarking*, **BTW 2015**

Publications

…, highscalability.com, it – Information Technology

[RHL^{+}15]	Ritter, Norbert (ed.); Henrich, Andreas (ed.); Lehner, Wolfgang (ed.); Thor, Andreas (ed.); Friedrich, Steffen (ed.); Wingerath, Wolfram (ed.): Datenbanksysteme für Business, Technologie und Web (BTW 2015) – *Workshopband*, **BTW 2015**
[GSW^{+}15]	Gessert, Felix; Schaarschmidt, Michael; Wingerath, Wolfram; Friedrich, Steffen; Ritter, Norbert: *The Cache Sketch: Revisiting Expiration-based Caching in the Age of Cloud Data Management*, **BTW 2015**
[Win16]	Wingerath, Wolfram: *The Joy of Deploying Apache Storm on Docker Swarm*, **highscalability.com** (2016).
[WGFR16]	Wingerath, Wolfram; Gessert, Felix; Friedrich, Steffen; Ritter, Norbert: *Real-Time Stream Processing for Big Data*, **it – Information Technology** 58 (2016).

Publications

…, SummerSOC 2016, SCDM 2017, BTW 2017

[GWFR16]	Gessert, Felix; Wingerath, Wolfram; Friedrich, Steffen; Ritter, Norbert: *NoSQL Database Systems: A Survey & Decision Guidance*, **SummerSOC 2016**
[WGF$^+$17]	Wingerath, Wolfram; Gessert, Felix; Friedrich, Steffen; Witt, Erik; Ritter, Norbert: *The Case For Change Notifications in Pull-Based Databases*, **SCDM 2017**
[FWR17]	Friedrich, Steffen; Wingerath, Wolfram; Ritter, Norbert: *Coordinated Omission in NoSQL Database Benchmarking*, **SCDM 2017**
[Win17]	Wingerath, Wolfram: *Real-Time Databases Explained: Why Meteor, RethinkDB, Parse & Firebase Don't Scale*, **Baqend Tech Blog** (2017).
[GWR17]	Gessert, Felix; Wingerath, Wolfram; Ritter, Norbert: *Scalable Data Management: An In-Depth Tutorial on NoSQL Data Stores*, **BTW 2017**

Publications

…, VLDB 2017, EDBT 2018, Springer Book, BTW 2019

[GSW$^+$17] Gessert, Felix; Schaarschmidt, Michael; Wingerath, Wolfram; Witt, Erik; Yoneki, Eiko; Ritter, Norbert: Quaestor: Query Web Caching for Database-as-a-Service Providers, **VLDB 2017**

[WGW$^+$18] Wingerath, Wolfram; Gessert, Felix; Witt, Erik; Friedrich, Steffen; Ritter, Norbert: *Real-Time Data Management for Big Data*, **EDBT 2018**

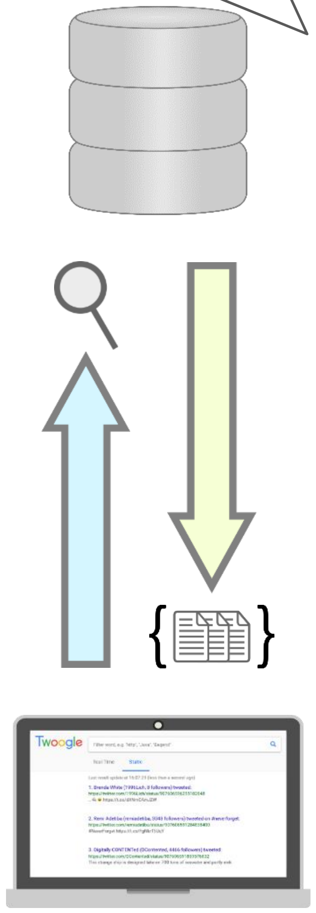
[WRG19] Wingerath, Wolfram; Ritter, Norbert; Gessert, Felix: *Real-Time & Stream Data Management: Push-Based Data in Research & Practice*, Springer International Publishing, **book published in 2019**
ISBN 978-3-030-10554-9

[WGR19a] Wingerath, Wolfram; Gessert, Felix; Ritter, Norbert: *Twoogle: Searching Twitter With MongoDB Queries*, **BTW 2019**

[WGR19b] Wingerath, Wolfram; Gessert, Felix; Ritter, Norbert: *NoSQL & Real-Time Data Management in Research & Practice*, **BTW 2019**

Summary & Contributions

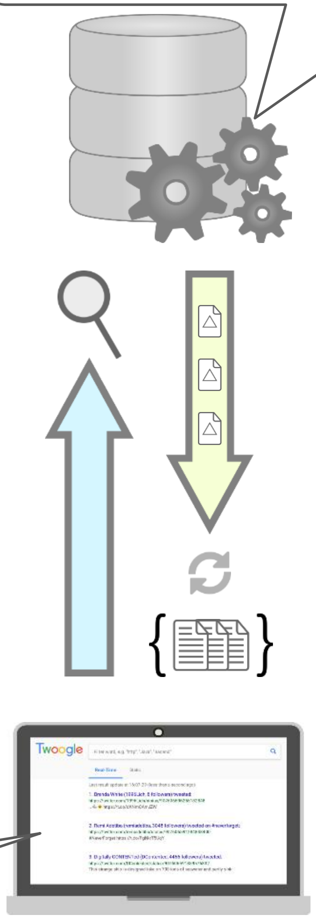
1.) System Categorization



Traditional Databases:
pull-based queries

- inefficient
- slow

3.) A MongoDB-Based Implementation



2.) RTDB System Design for Opt-in Real-Time Queries

With **InvaliDB**:
push-based queries

- scalable & fast
- expressive
- legacy-compatible

4.) Proof of Practicality Through Integration With Orestes