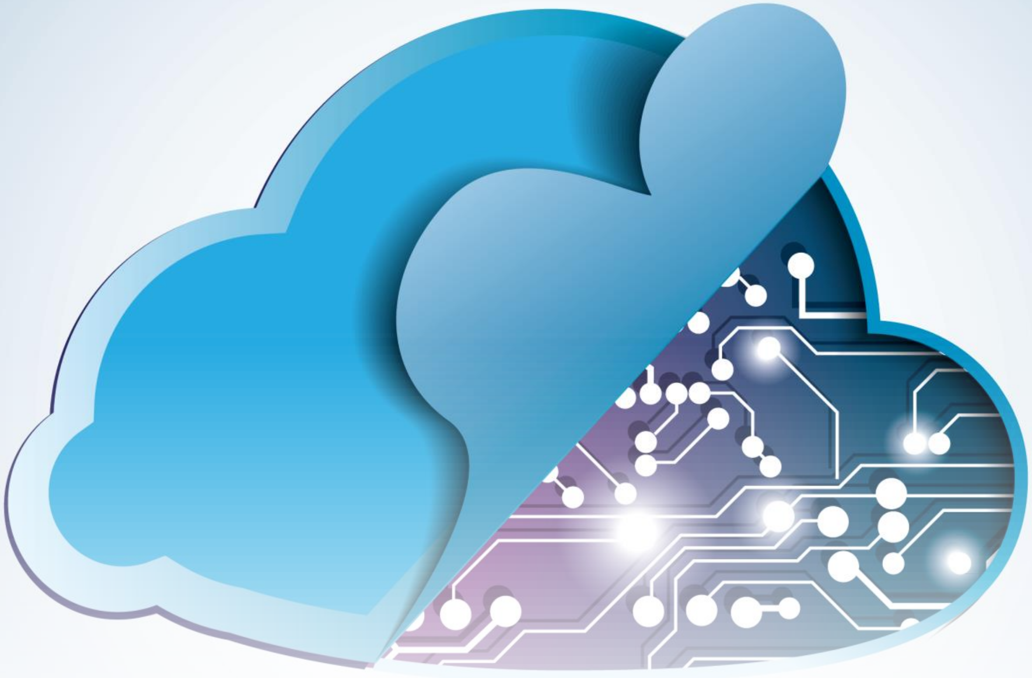


Towards Scalable Cloud Data Management

Felix Gessert, Norbert Ritter
gessert/ritter@informatik.uni-hamburg.de

SCDM 2015

3rd Workshop on Scalable Cloud Data Management

Co-located with the IEEE BigData Conference.
Santa Clara, CA, October 29th 2015.
Starting at **8am in Ballroom C**.

Workshop Schedule

OCTOBER 5, 2015
Camera-ready

October 29, 2015

SCDM 2015

SCDM 2015 Workshop in Santa Clara, CA. The preliminary schedule is online. SCDM starts on Oct 29, 8am (Ballroom C) with a keynote by Russel Sears on "Purity and the future of scalable storage".

Camera-ready
SCDM 2015 announced
Extended P Deadline
SCDM 2015
Notification

BTW
2015
Hamburg

Outline

Motivation

ORESTES: a Cloud-Database Middleware

Solving Latency and Polyglot Storage

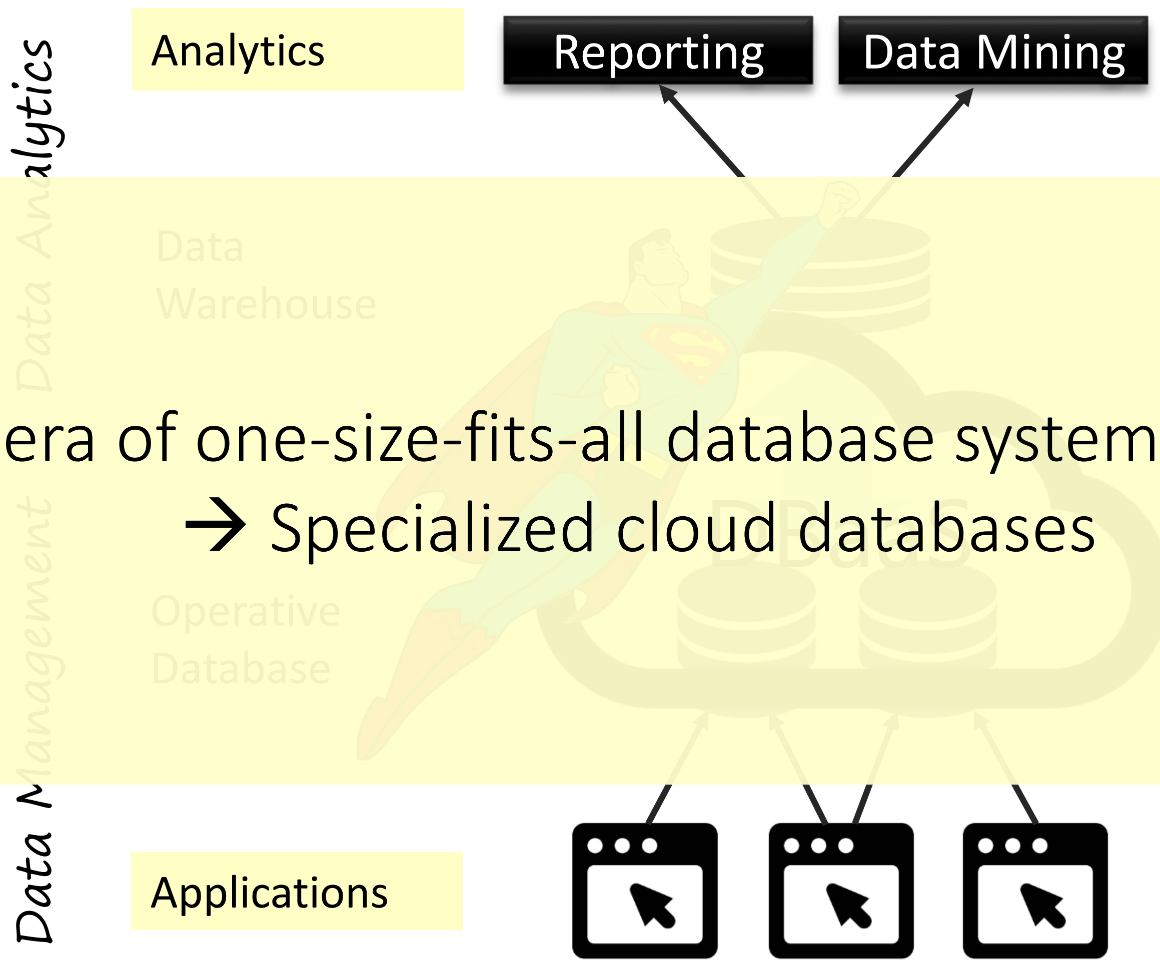
Wrap-up

- *Cloud Data Management*
- Cloud Database Models
- Research Challenges

Introduction: What are the challenges in Cloud Data Management?

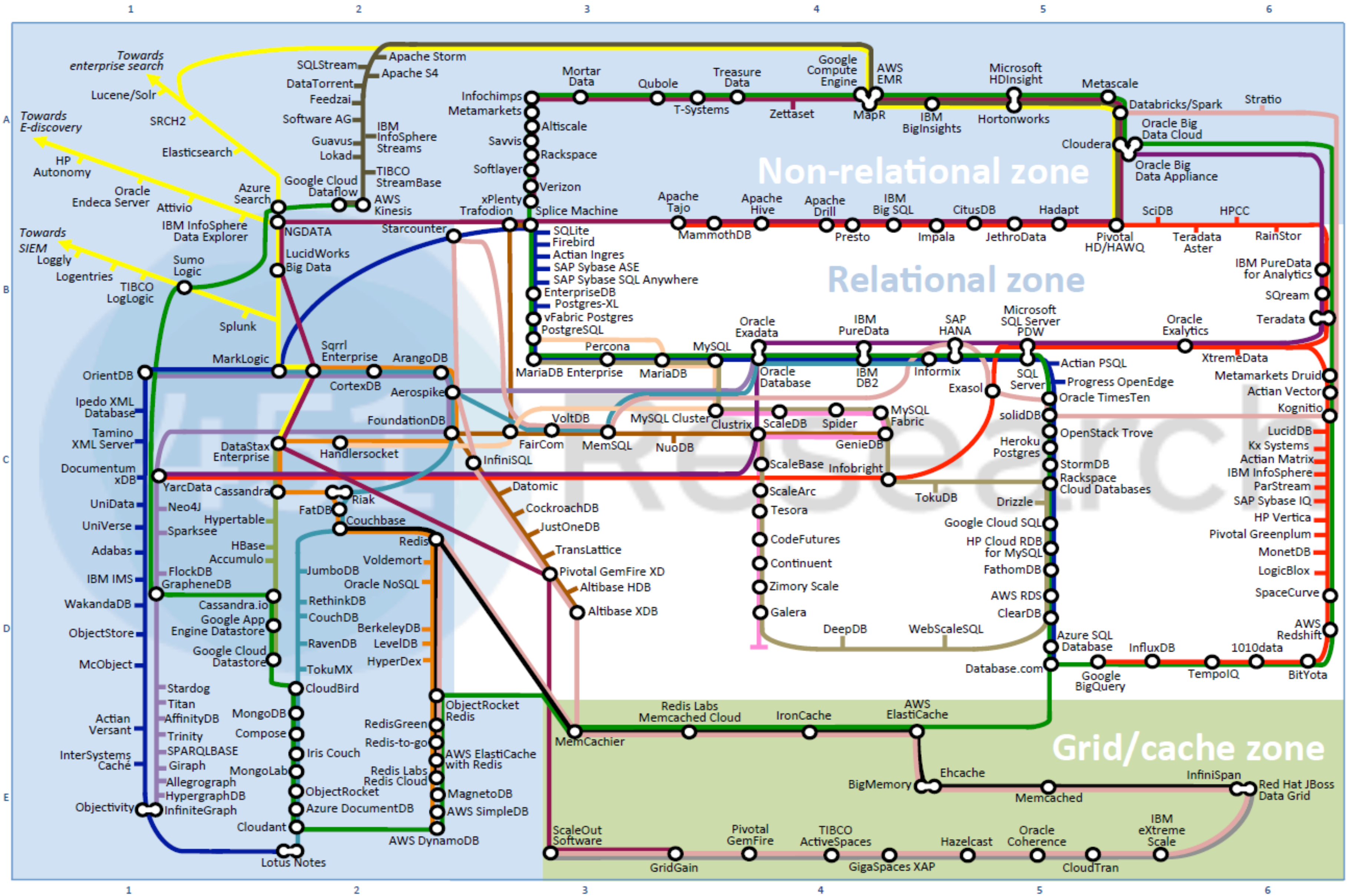
Architecture

Typical Data Architecture:



The era of one-size-fits-all database systems is over

→ Specialized cloud databases



Non-relational zone
Relational zone
Grid/cache zone
Towards enterprise search
Towards E-discovery
Towards SIEM
Lucene/Solr
SRCH2
Elasticsearch
HP Autonomy
Oracle Endeca Server
Attivio
Azure Search
IBM InfoSphere Data Explorer
Loggly
Logentries
Sumo Logic
TIBCO LogLogic
Splunk
SQLStream
DataTorrent
Feedzai
Software AG
Guavus
Lokad
Google Cloud Dataflow
Apache Storm
Apache S4
IBM InfoSphere Streams
TIBCO StreamBase
AWS Kinesis
Infochimps
Metamarkets
Altiscale
Savvis
Rackspace
Softlayer
Verizon
xPlenty
Trafodion
Splice Machine
Mortar Data
Qubole
T-Systems
Treasure Data
Zettaset
Google Compute Engine
AWS EMR
MapR
IBM BigInsights
Microsoft HDInsight
Hortonworks
Metascale
Databricks/Spark
Stratio
Oracle Big Data Cloud
Cloudera
Oracle Big Data Appliance
Apache Tajo
MammothDB
Apache Hive
Apache Drill
Presto
IBM Big SQL
Impala
CitusDB
JethroData
Hadapt
Pivotal HD/HAWQ
SciDB
Teradata Aster
HPCC
RainStor
NGDATA
LucidWorks Big Data
Starcounter
SQLite
Firebird
Actian Ingres
SAP Sybase ASE
SAP Sybase SQL Anywhere
EnterpriseDB
Postgres-XL
vFabric Postgres
PostgreSQL
IBM PureData for Analytics
SQream
Teradata
MarkLogic
Sqrrl Enterprise
ArangoDB
OrientDB
CortexDB
Aerospike
Percona
MySQL
MariaDB Enterprise
MariaDB
Oracle Exadata
Oracle Database
IBM PureData
IBM DB2
SAP HANA
Informix
Microsoft SQL Server PDW
SQL Server
Exasol
Oracle Exalytics
XtremeData
Actian PSQL
Progress OpenEdge
Oracle TimesTen
Metamarkets Druid
Actian Vector
Kognitio
Ipedo XML Database
Tamino XML Server
Documentum xDB
FoundationDB
DataStax Enterprise
Handlersocket
VoltDB
FairCom
MemSQL
MySQL Cluster
NuoDB
Clustrix
ScaleDB
Spider
MySQL Fabric
GenieDB
solidDB
OpenStack Trove
Heroku Postgres
StormDB
Rackspace Cloud Databases
LucidDB
Kx Systems
Actian Matrix
IBM InfoSphere
ParStream
SAP Sybase IQ
HP Vertica
Pivotal Greenplum
MonetDB
LogicBlox
SpaceCurve
AWS Redshift
InfiniSQL
ScaleBase
Infobright
YarcData
Cassandra
Neo4J
Hypertable
Sparksee
HBase
Accumulo
FlockDB
GrapheneDB
Riak
FatDB
Couchbase
Datomic
CockroachDB
JustOneDB
TransLattice
Pivotal GemFire XD
Altibase HDB
Altibase XDB
ScaleArc
Tesora
CodeFutures
Continuent
Zimory Scale
Galera
DeepDB
WebScaleSQL
TokuDB
Drizzle
Google Cloud SQL
HP Cloud RDB for MySQL
FathomDB
AWS RDS
ClearDB
Azure SQL Database
Database.com
Google BigQuery
InfluxDB
TempoIQ
1010data
BitYota
UniData
UniVerse
Adabas
IBM IMS
WakandaDB
ObjectStore
McObject
Actian Versant
InterSystems Caché
Objectivity
Cassandra.io
Google App Engine Datastore
Google Cloud Datastore
Redis
JumboDB
Voldemort
Oracle NoSQL
RethinkDB
CouchDB
BerkeleyDB
RavenDB
LevelDB
HyperDex
TokuMX
CloudBird
Stardog
Titan
AffinityDB
Trinity
SPARQLBASE
Giraph
Allegrograph
HypergraphDB
InfiniteGraph
MongoDB
Compose
Iris Couch
MongoLab
ObjectRocket
Azure DocumentDB
Cloudant
Lotus Notes
RedisGreen
Redis-to-go
Redis Labs Redis Cloud
ObjectRocket Redis
AWS ElastiCache with Redis
MagnetoDB
AWS SimpleDB
AWS DynamoDB
MemCachier
Redis Labs Memcached Cloud
IronCache
AWS ElastiCache
BigMemory
Ehcache
Memcached
InfiniSpan
Red Hat JBoss Data Grid
ScaleOut Software
GridGain
Pivotal GemFire
TIBCO ActiveSpaces
GigaSpaces XAP
Hazelcast
Oracle Coherence
CloudTran
IBM eXtreme Scale

Database Sweetspots

RDBMS

General-purpose ACID transactions

Parallel DWH

Aggregations/OLAP for massive data amounts

NewSQL

High throughput relational OLTP

Wide-Column Store

Long scans over structured data

Document Store

Deeply nested data models

Key-Value Store

Large-scale session storage

Graph Database

Graph algorithms & queries

In-Memory KV-Store

Counting & statistics

Wide-Column Store

Massive user-generated content

Cloud-Database Sweetspots

Firebase

Realtime BaaS

Communication and collaboration

Amazon RDS

Managed RDBMS

General-purpose ACID transactions

Amazon ElastiCache

Managed Cache

Caching and transient storage

Azure Tables

Wide-Column Store

Very large tables

Amazon DynamoDB

Wide-Column Store

Massive user-generated content

Parse

Backend-as-a-Service

Small Websites and Apps

bonsai

Managed NoSQL

Full-Text Search

Google Cloud Storage

Object Store

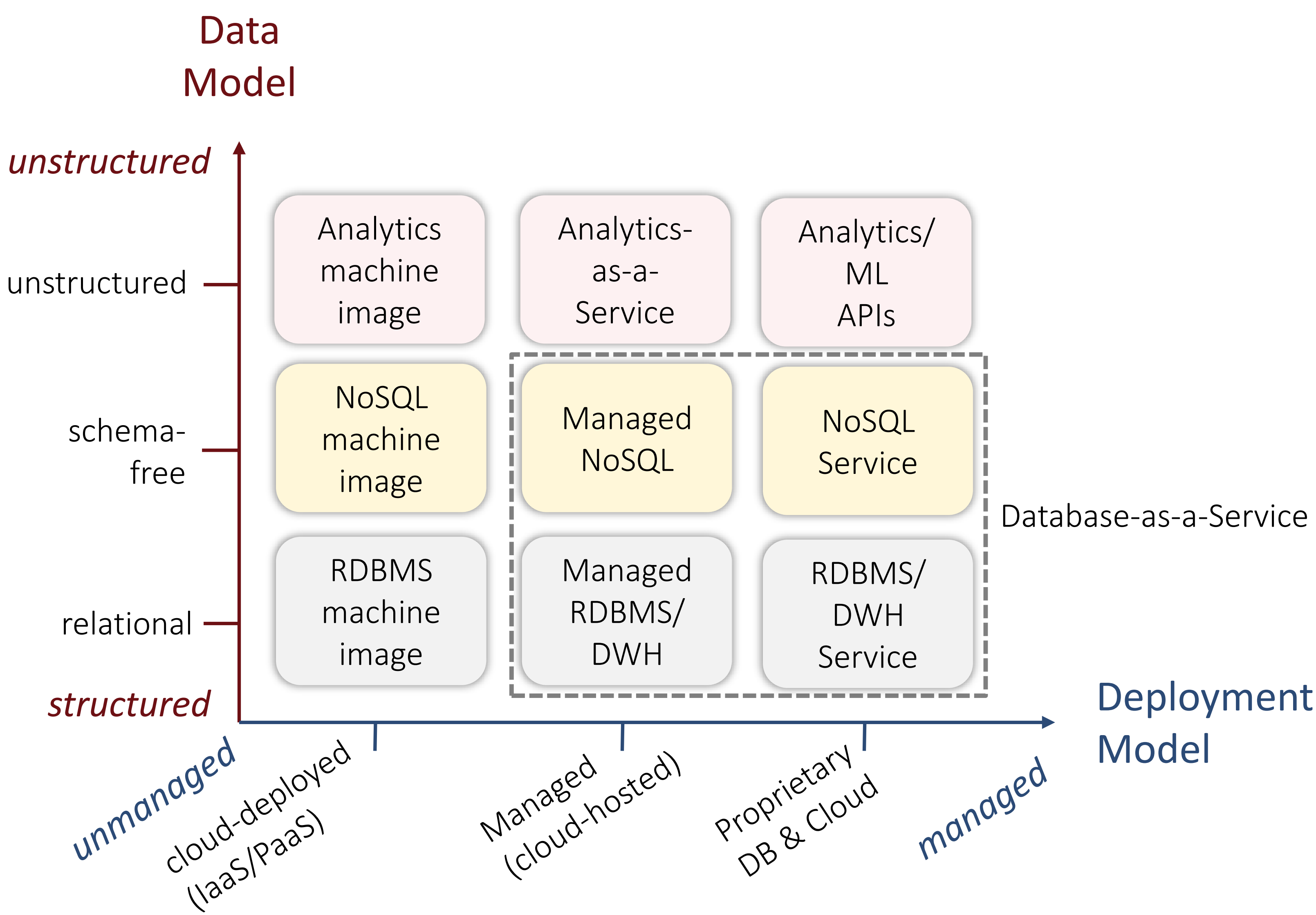
Massive File Storage

Amazon Elastic MapReduce

Hadoop-as-a-Service

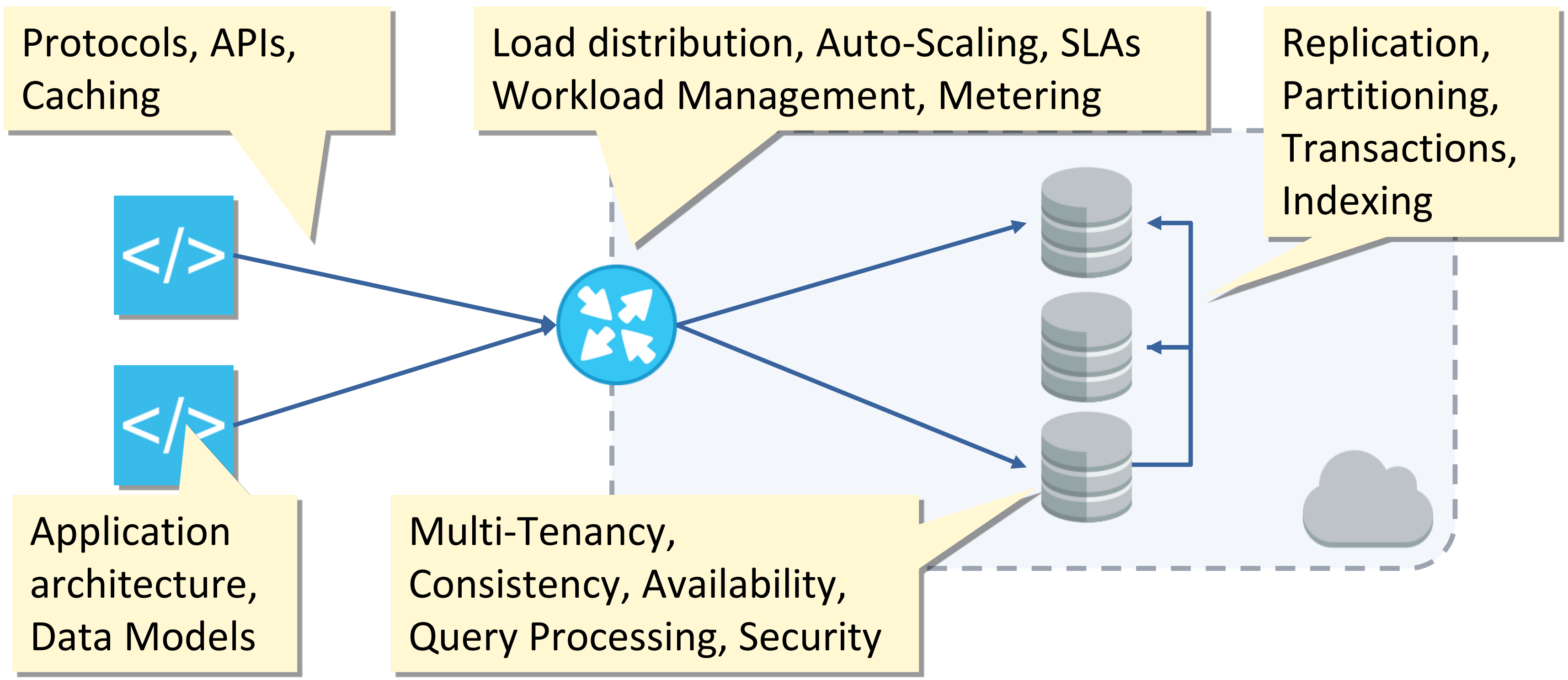
Big Data Analytics

Cloud-Database Models



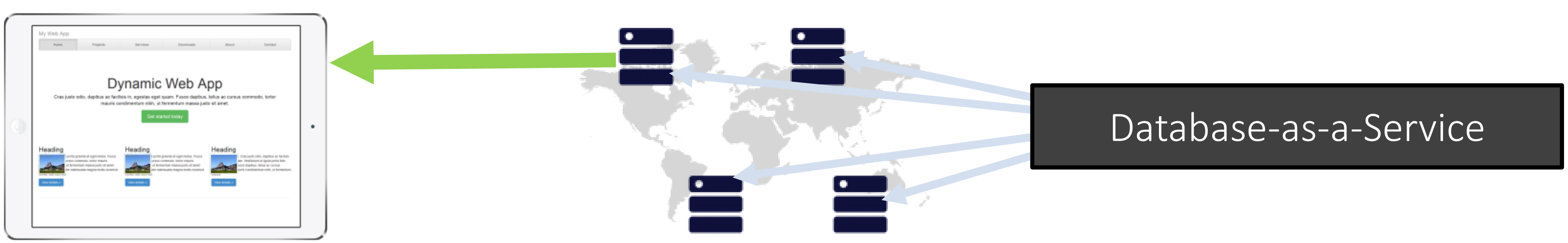
Cloud Data Management

- Research field tackling the *design*, *implementation*, *evaluation* and *application implications* of **database systems in cloud environments**:



Open Research Questions
Performance & Latency

- How can database systems support **novel application architectures** (e.g., single-page or real-time apps)?
- Can the **functionality-performance trade-off** popularized by the NoSQL movement be turned into a tunable runtime configuration?
- How can a DBaaS deliver **low latency** in face of distributed storage and application tiers?



Open Research Questions
Consistency & Transactionality

- Which consistency and transaction guarantees can be provided across (**geo-)replicated, partitioned,** possibly heterogeneous/polyglot database systems?
- How can the **consistency-latency-availability trade-off** be best exposed to applications and developers?
- Can the existing methods (*quorum-based, consensus-based, master-slave*, etc.) be **reconciliated** into a single approach and tied to application requirements?
- How can we **replace CAP** by a more fine-grained and nuanced consistency classification scheme?

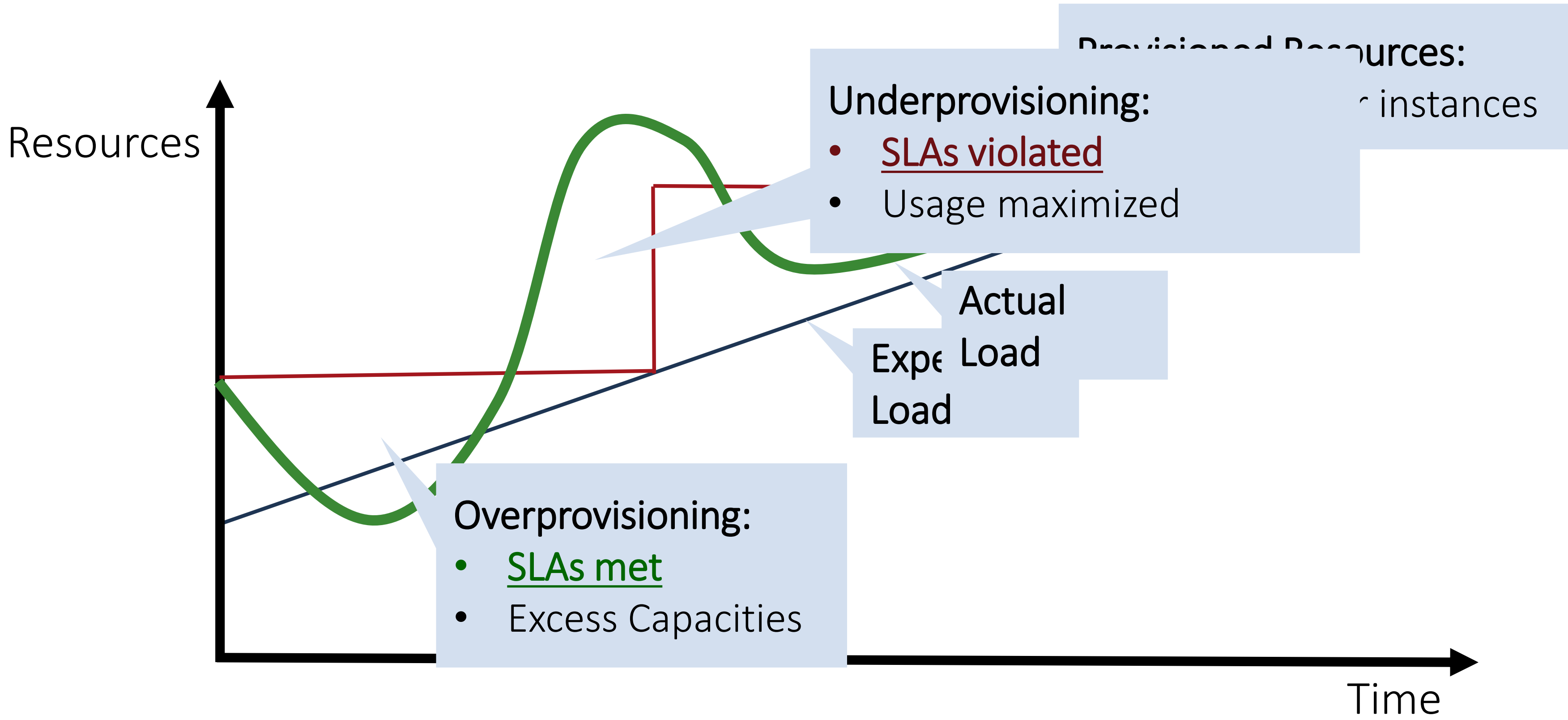
Open Research Questions
Service-Level Agreements

- How can database SLAs be guaranteed in a virtualized, multi-tenant **cloud environment**?
- Can we derive Service-Level-Objectives that are easy enough to understand and maintain to be **practical**?

	Model	CAP	SLAs
SimpleDB	Table-Store	CP	✗
DynamoDB	Table-Store	CP	✗
Azure Tables	Table-Store	CP	99.9% uptime
AE/Cloud DataStore	Entity-Group Store	CP	✗
S3, Az. Blob, GCS	Object-Store	AP	99.9% uptime (S3)

Open Research Questions
Service-Level Agreements

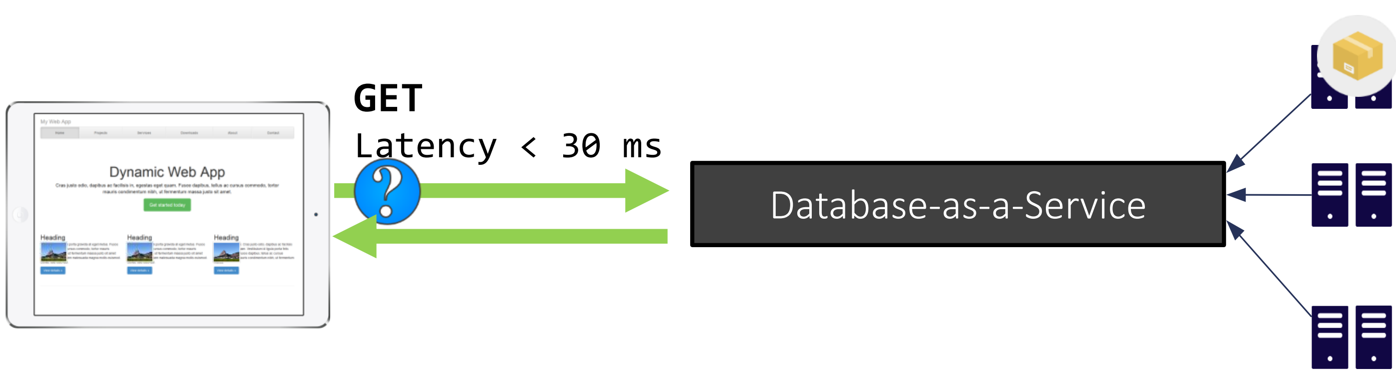
- How can SLAs be incorporated in **autoscaling** to optimize costs and minimize SLA violations?



T. Lorido-Botran, J. Miguel-Alonso et al.: "Auto-scaling Techniques for Elastic Applications in Cloud Environments". Technical Report, 2013

Open Research Questions
Poylgot Persistence

- Can the data system functions of **storage, search, streaming and analytics** be unified and integrated?
- Is it possible to **automate**, optimize and learn the **best choice** of given database systems?
- How can queries and data be **routed to databases,** so that SLAs & performance characteristics are met?



Outline

- Motivation
- **ORESTES: a Cloud-Database Middleware**
- Solving Latency and Polyglot Storage
- Wrap-up

- Two problems:
  - Latency
  - Polyglot Storage
- Vision: Orestes Middleware

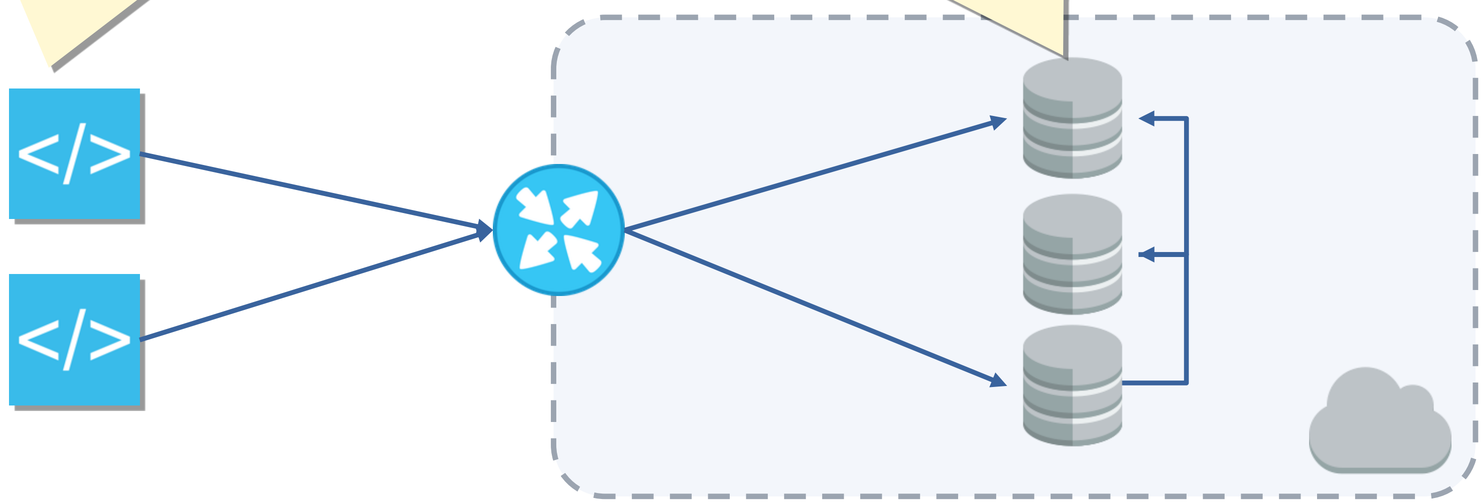
Latency & Polyglot Storage

Two central problems

- Goal of ORESTES: Solve both problems through a scalable **cloud-database middleware**

If the application is *geographically distributed*, how can we guarantee **fast database access**?

If one size *doesn't* fit all – how can **polyglot persistence** be leveraged on a declarative, automated basis?



Problem I: Latency

100ms 400ms 500ms 1s

Average: 9,3s

Loading…

Conversions

% Traffic

% Visitors

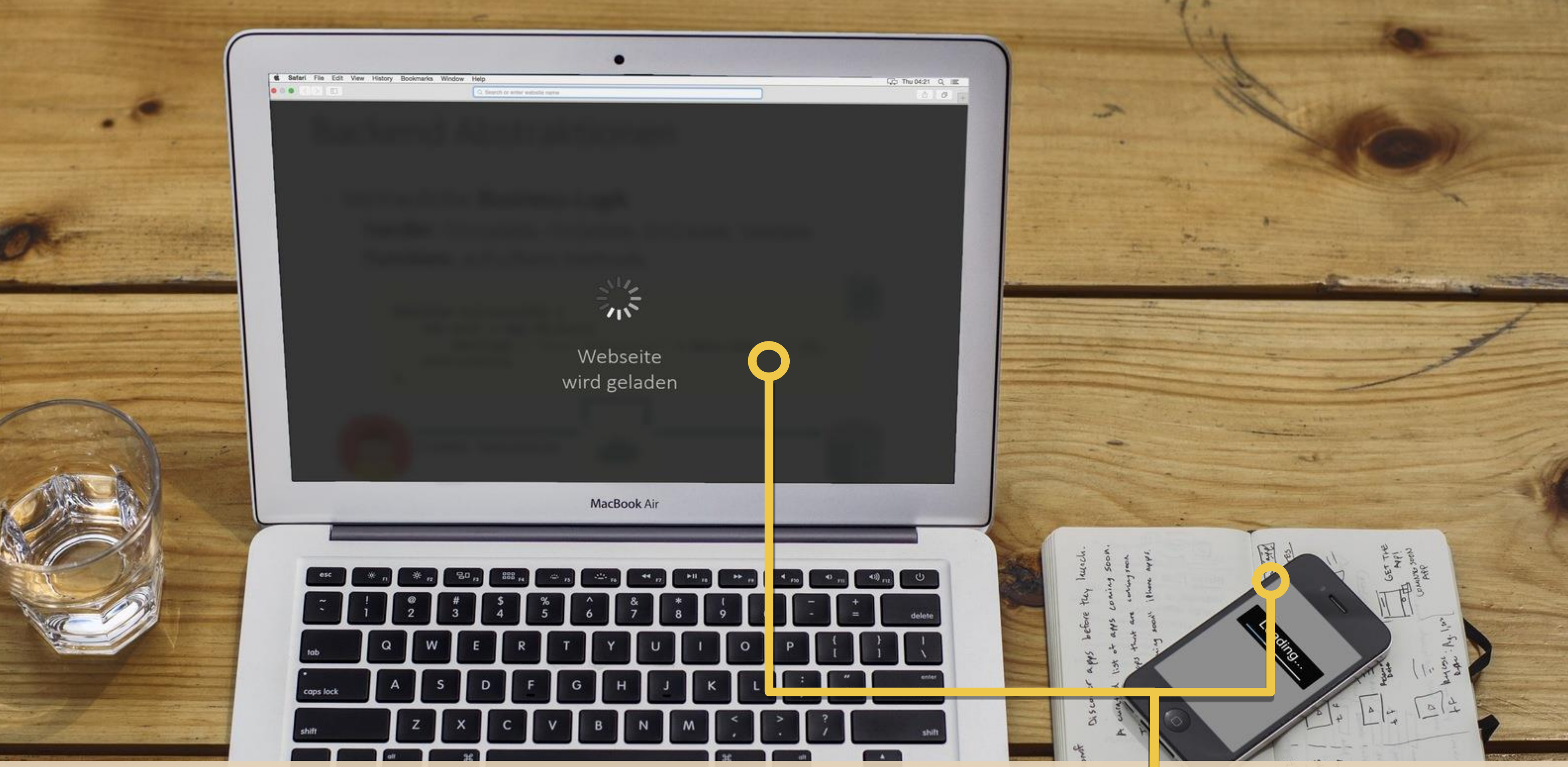
% Revenue

AberdeenGroup

Google

YAHOO!

amazon.com

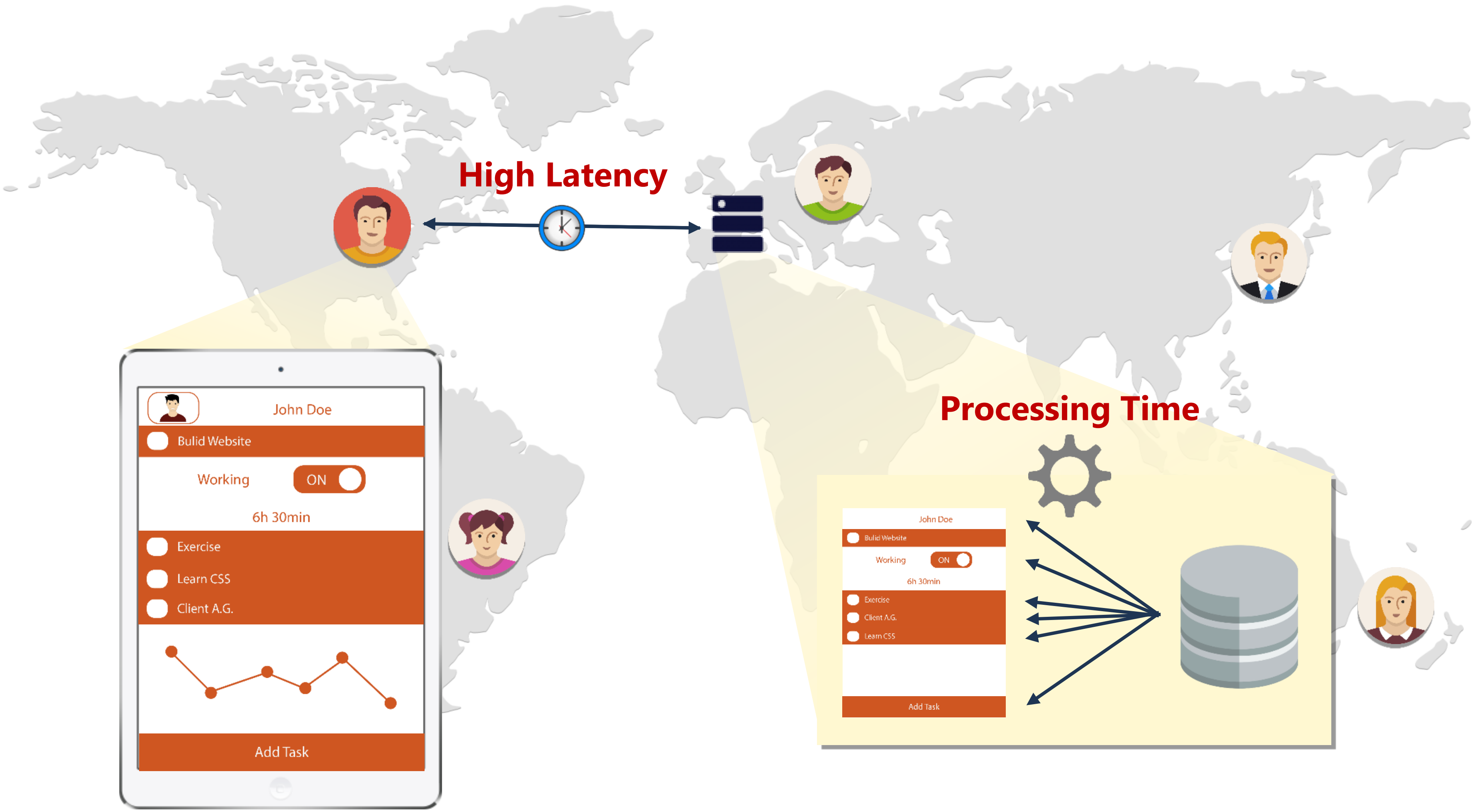


If perceived speed is such an important factor

...what causes slow page load times?

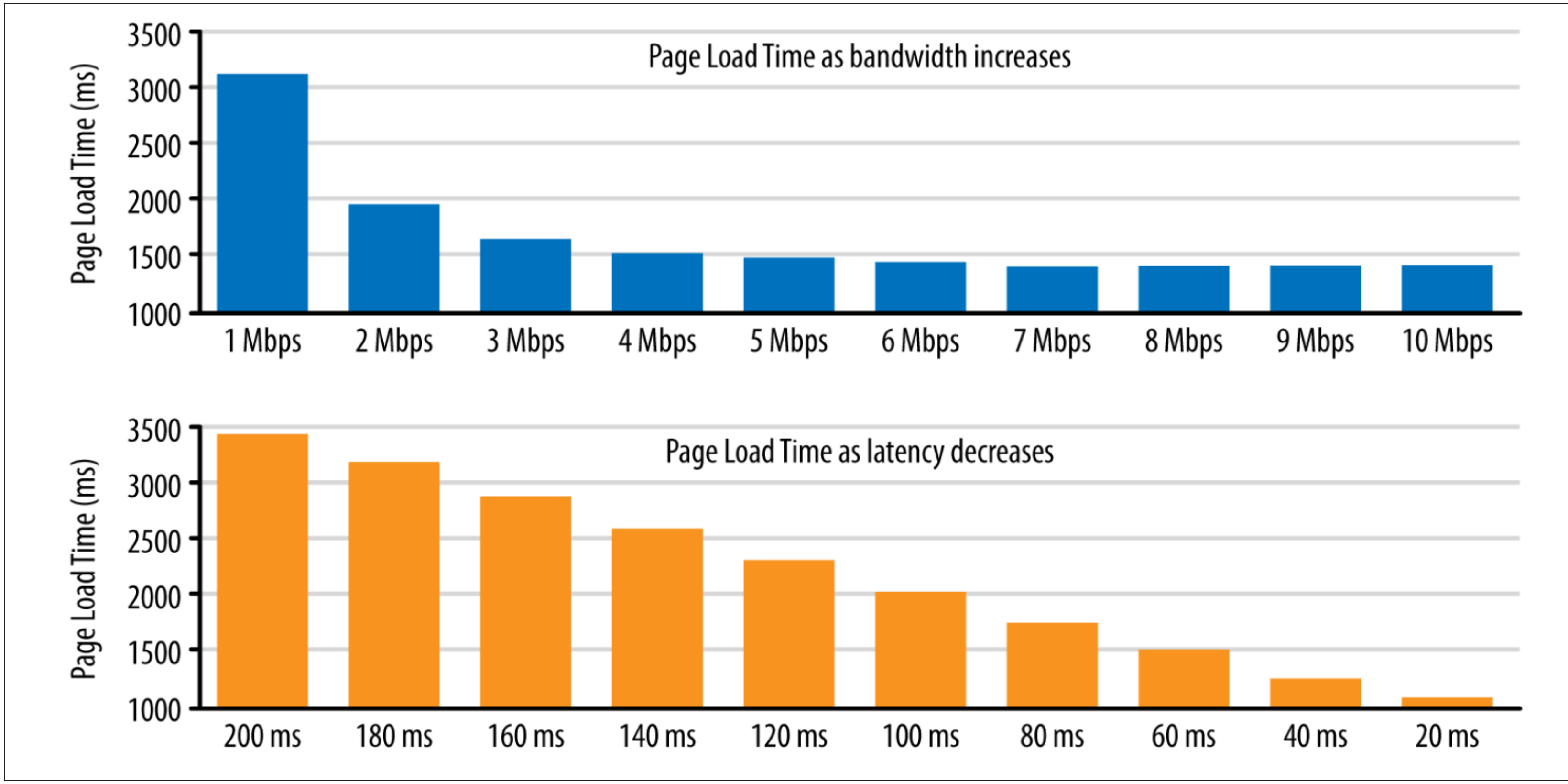
State of the art

Two bottlenecks: latency und processing



Network Latency

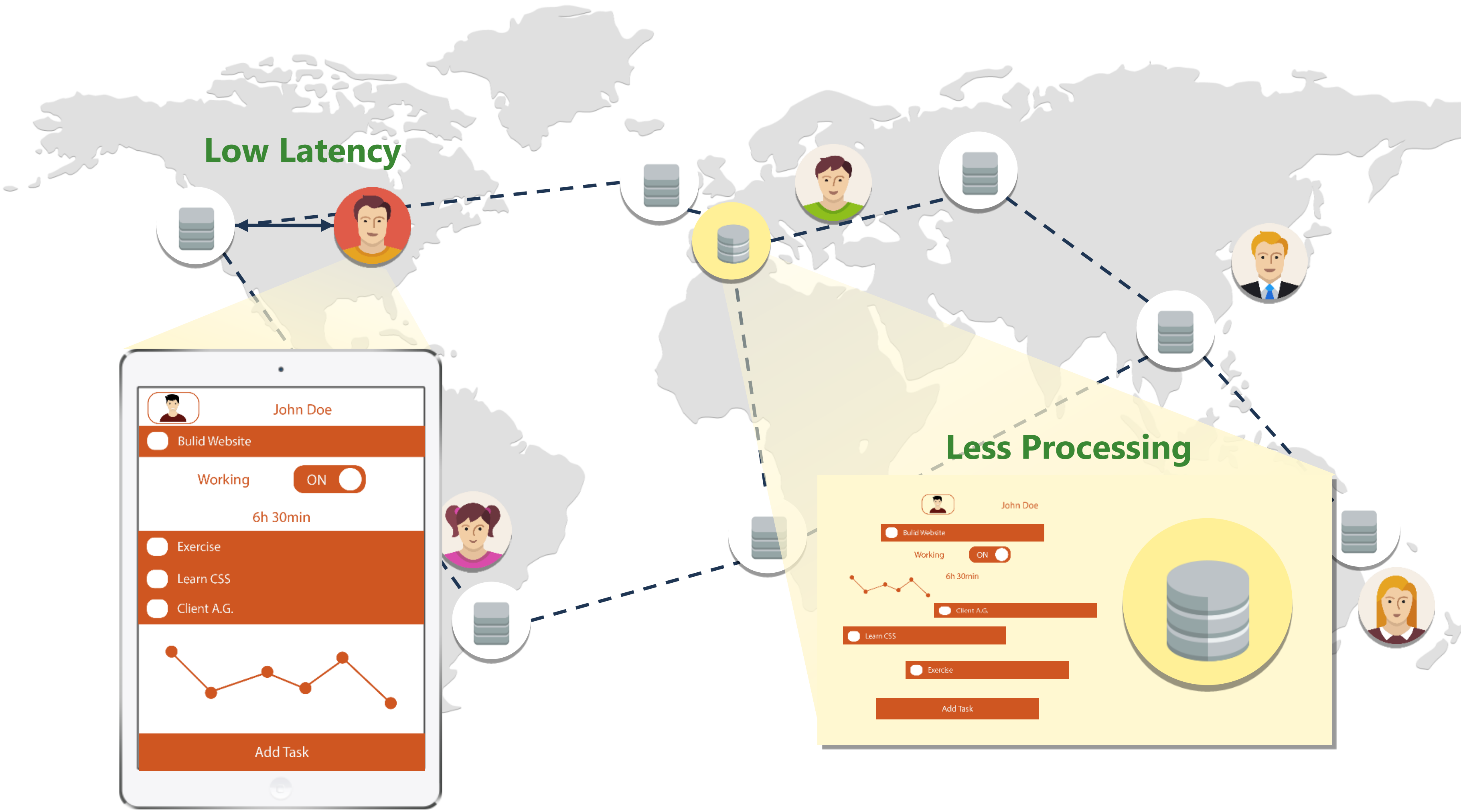
The underlying problem of high page load times



I. Grigorik, High performance browser networking. O'Reilly Media, 2013.

The low-latency vision

Data is served by ubiquitous web-caches



The web's caching model

Staleness as a consequence of scalability

Expiration-based

Every object has a defined Time-To-Live (TTL)

Revalidations

Allow clients and caches to check freshness at the server

Research Question:

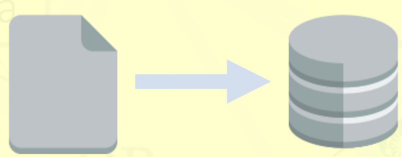
Can database services leverage the web caching infrastructure for low latency with rich **consistency guarantees**?

Problem II: Polyglot Persistence

Current best practice

Research Question:

Can we automate the



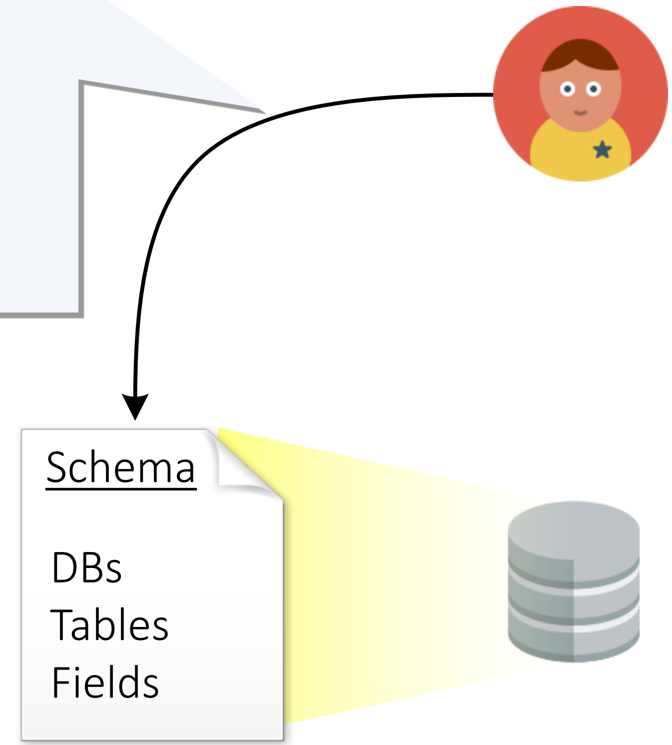
mapping problem?

Felix Gessert, Norbert Ritter: *Polyglot Persistence*. Datenbankspektrum.

Vision

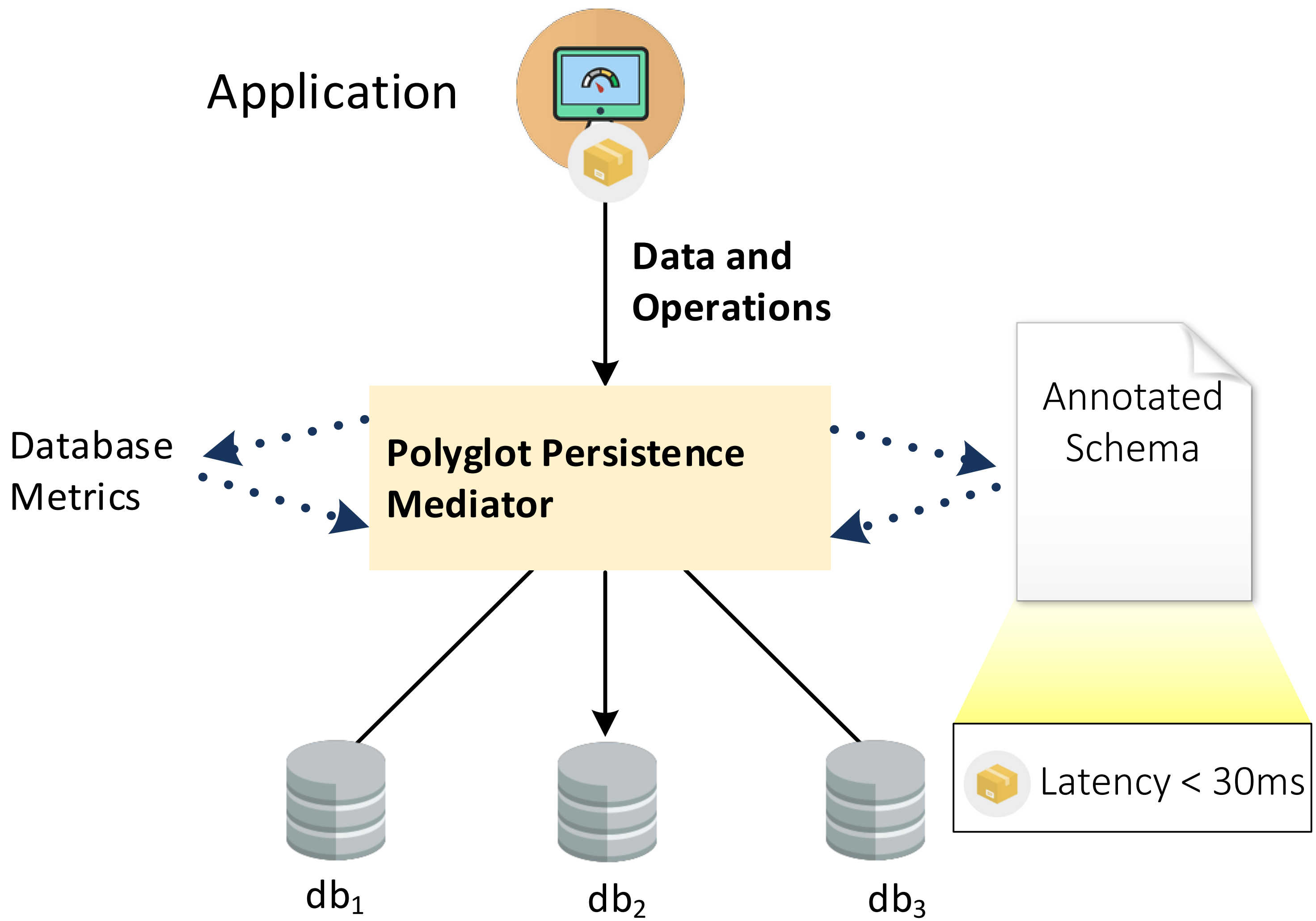
Schemas can be annotated with requirements

- Write Throughput > 10,000 RPS
- Read Availability > 99.9999%
- Scans = true
- Full-Text-Search = true
- Monotonic Read = true



Vision

The Polyglot Persistence Mediator chooses the database



The Big Picture

Implementation in ORESTES

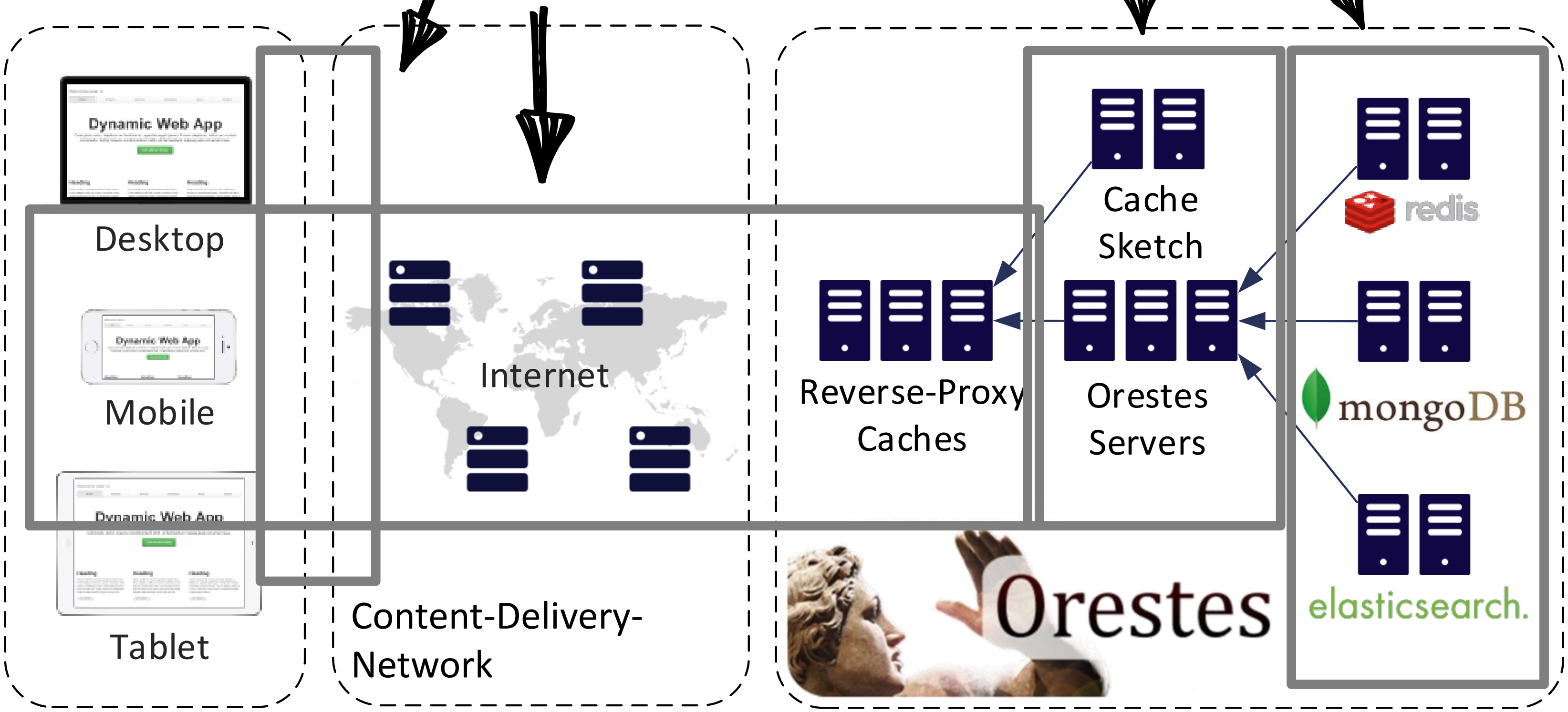
- Polyglot Storage and Low Latency are the central goals of ORESTES

Database-as-a-Service Middleware:
Caching, Transactions, Schemas,
Authorization, Multi-Tenancy

Polyglot Storage

Unified REST API

Standard HTTP Caching



Felix Gessert, Steffen Friedrich, Wolfram Wingerath, Michael Schaarschmidt, Norbert Ritter: *Towards a Scalable and Unified REST API for Cloud Data Stores*. GI-Jahrestagung 2014

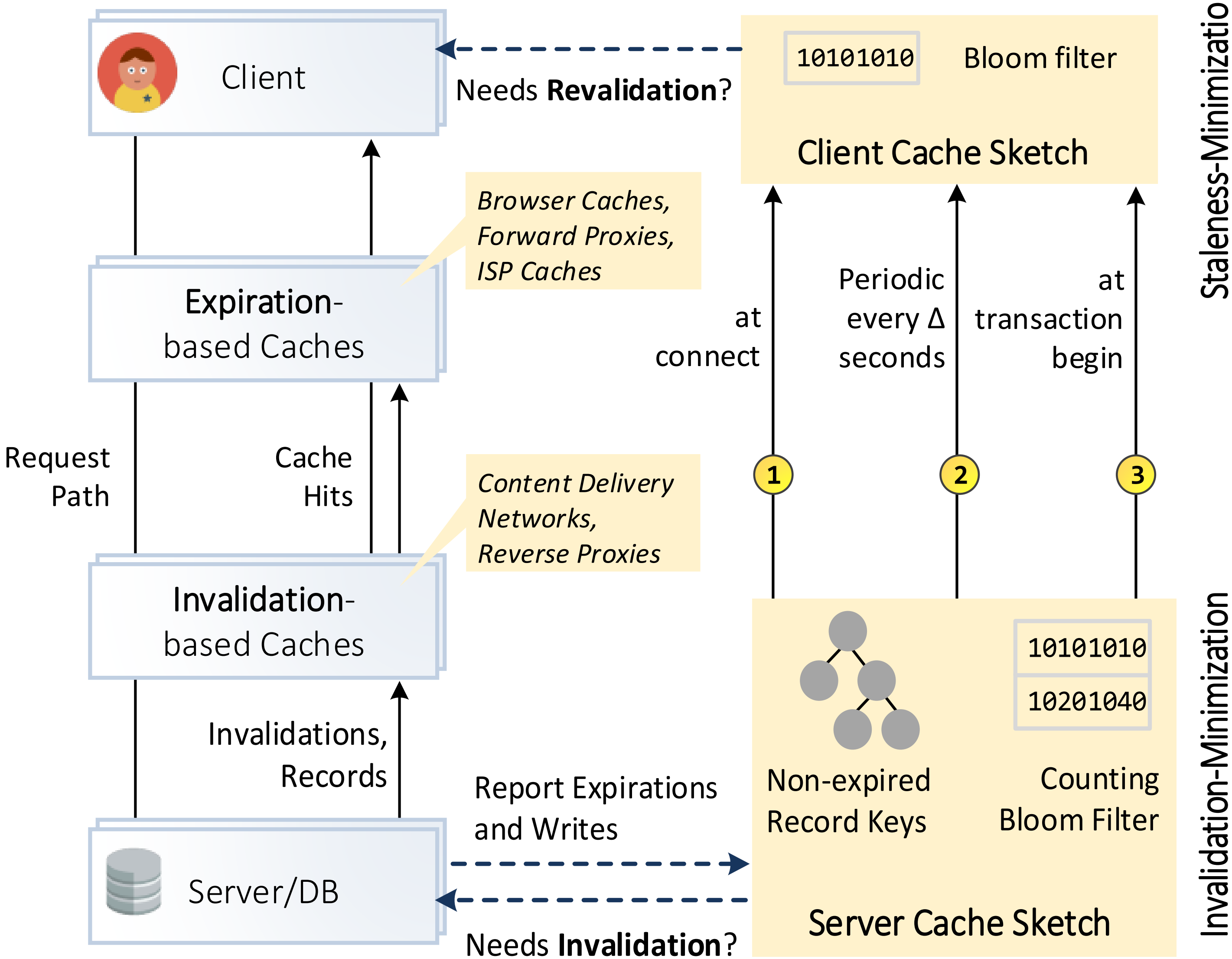
Outline

- Motivation
- ORESTES: a Cloud-Database Middleware
- **Solving Latency and Polyglot Storage**
- Wrap-up

- Cache Sketch Approach
  - Caching Objects
  - Caching Query Results
  - Continuous Queries
- Polyglot Persistence Mediator
  - Resolution
  - Mediation
  - Polyglot Materialized Views

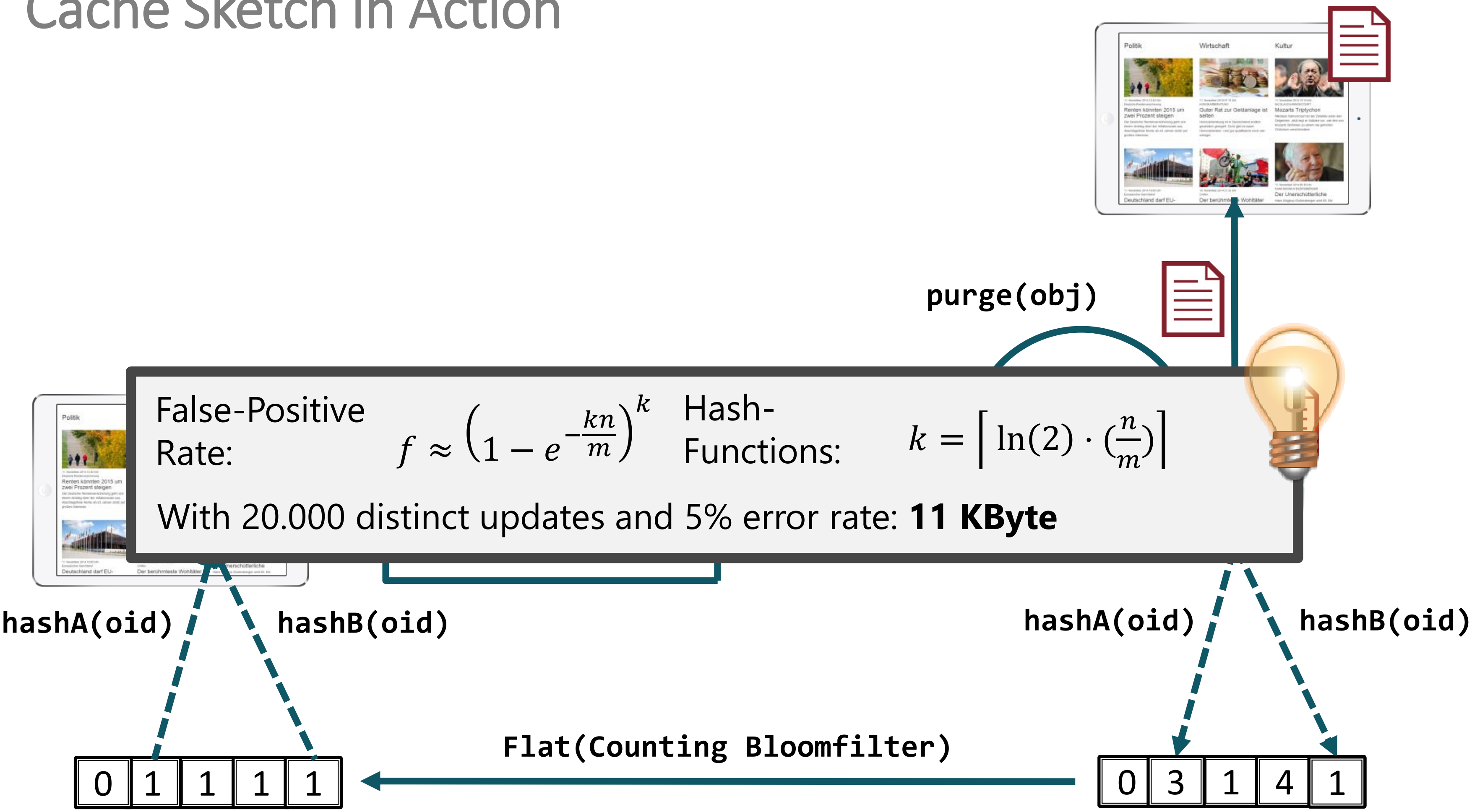
The Cache Sketch approach

Letting the client handle cache coherence



Visually Explained

Cache Sketch in Action



False-Positive Rate: $f \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$

Hash-Functions: $k = \left\lceil \ln(2) \cdot \left(\frac{n}{m}\right) \right\rceil$

With 20.000 distinct updates and 5% error rate: **11 KByte**

Object Caching

Summary of Properties

- *Consistency guarantee*: **Δ-atomicity**
- **Modes:**
  - *Cached initialization*: piggybacked Cache Sketch enables fast page loads
  - *Bounded Staleness*: application refreshes Cache Sketch in fixed intervals
  - *Conflict-Avoidant Optimistic Transactions*: guarantee ACID despite cached reads
- TTL Estimator: learns and (statistically) estimates appropriate expirations

Felix Gessert, Florian Bucklers, Norbert Ritter: Orestes: *A scalable Database-as-a-Service architecture for low latency*. CloudDB2014@ICDE.

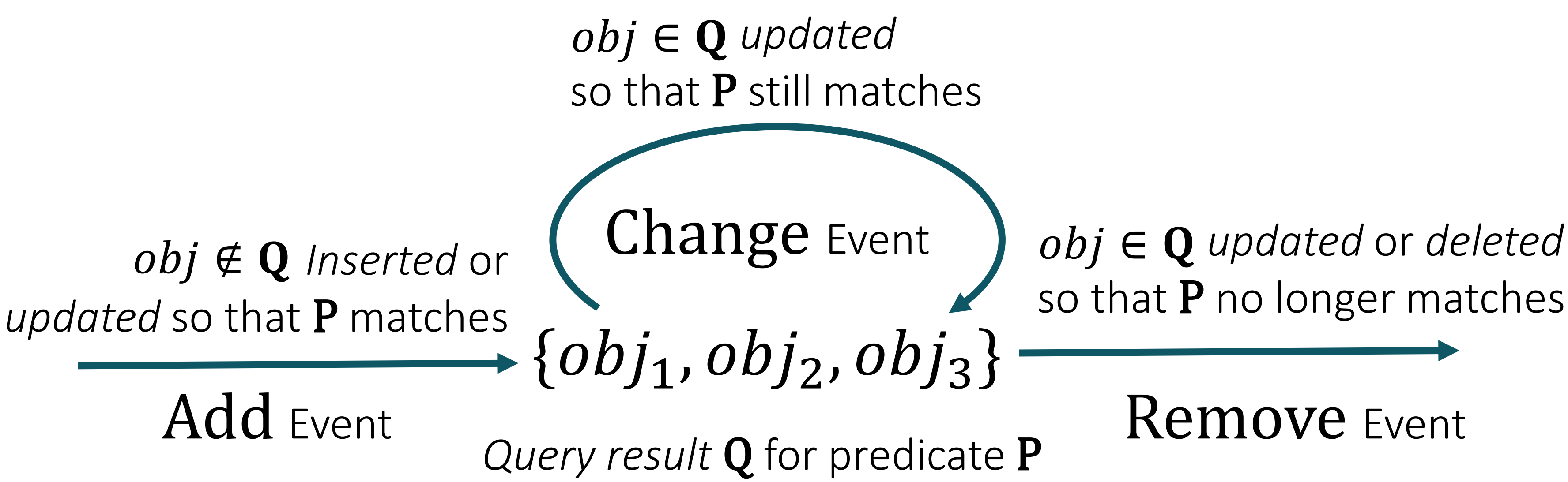
Felix Gessert, Michael Schaarschmidt, Wolfram Wingerath, Steffen Friedrich, Norbert Ritter: *The Cache Sketch: Revisiting Expiration-based Caching in the Age of Cloud Data Management*. BTW 2015

From Object Caching to Query Caching

Generalizing the Cache Sketch to query results

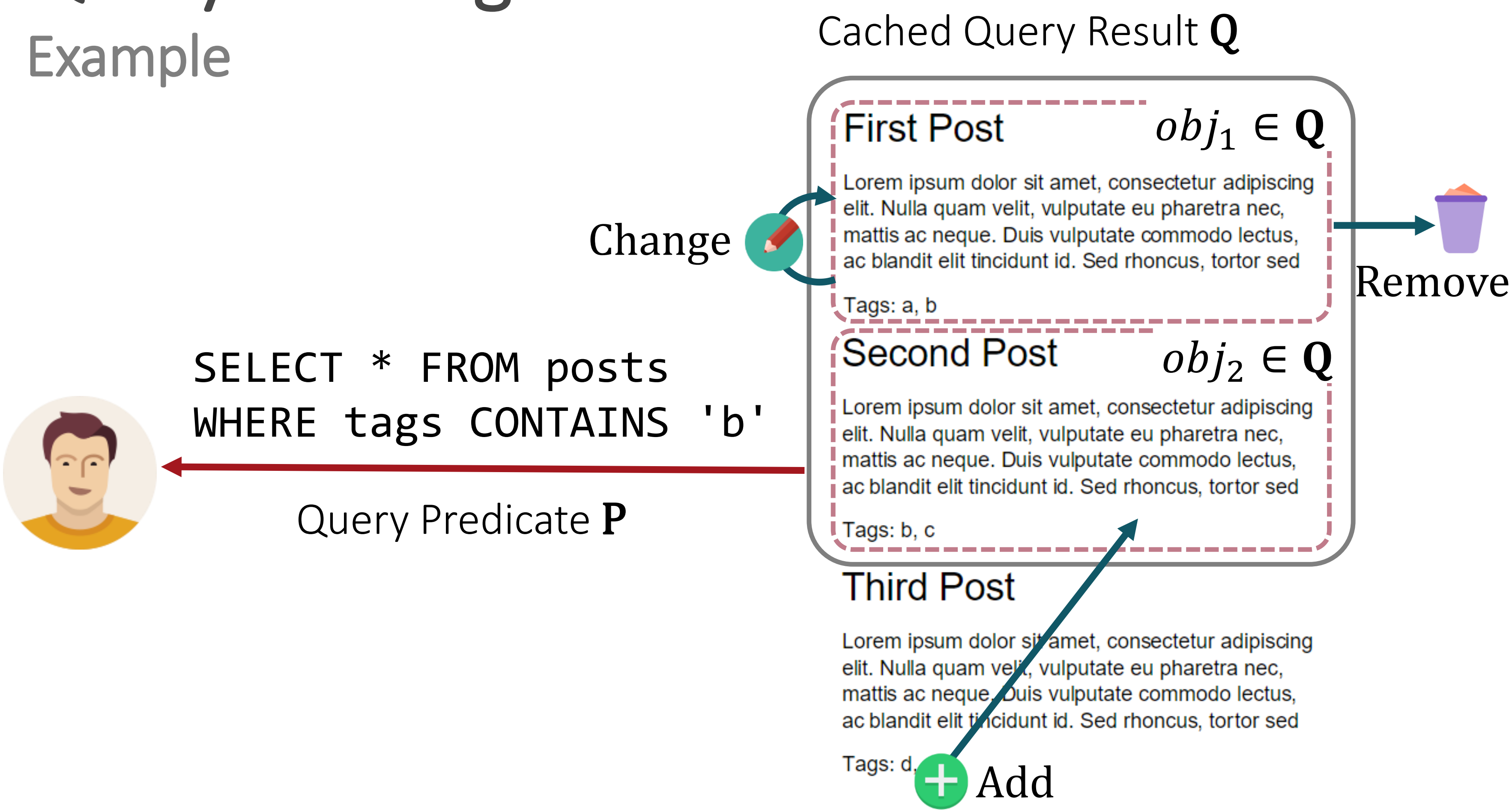
- **Main challenge**: when to invalidate?
  - **Objects**: for every update and delete
  - **Queries**: when the query result changes

→ How to detect query **result changes in real-time**?



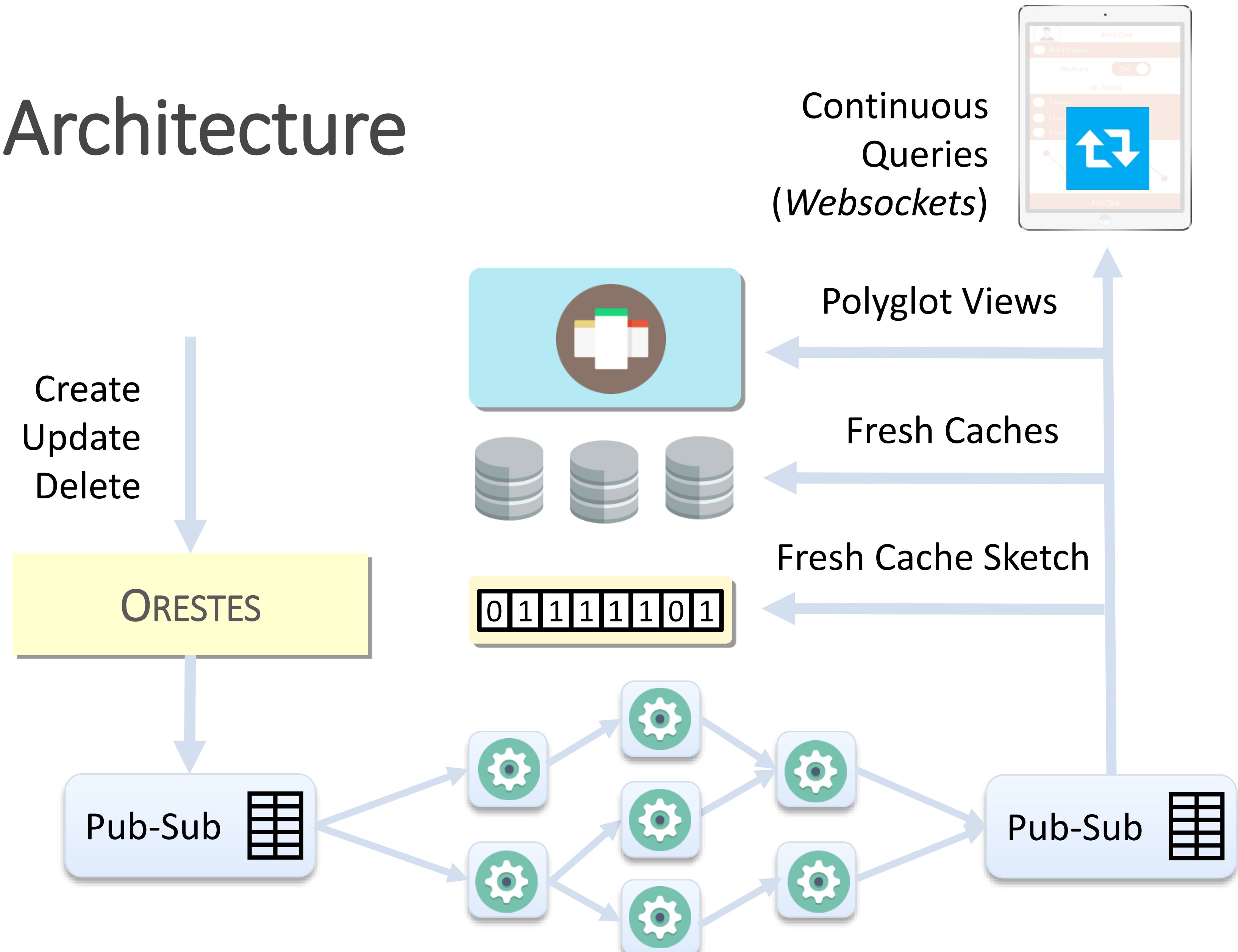
Query Caching

Example



- **Add, Change, Remove** all entail an invalidation and addition to the cache sketch

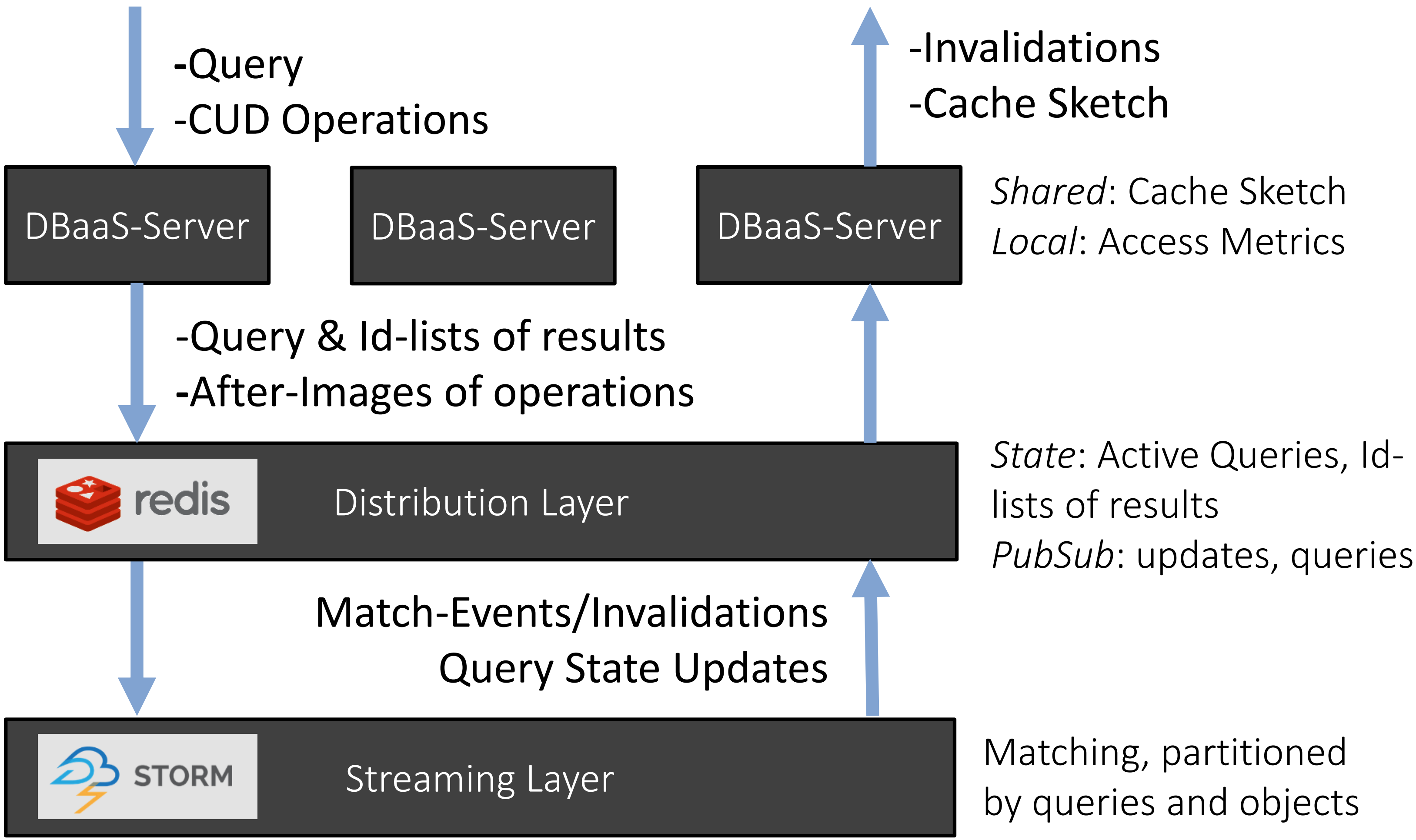
Architecture



Felix Gessert, Michael Schaarschmidt, Wolfram Wingerath, Steffen Friedrich, Norbert Ritter: *Quaestor: Scalable and Fresh Query Caching on the Web's Infrastructure*. Under Submission.

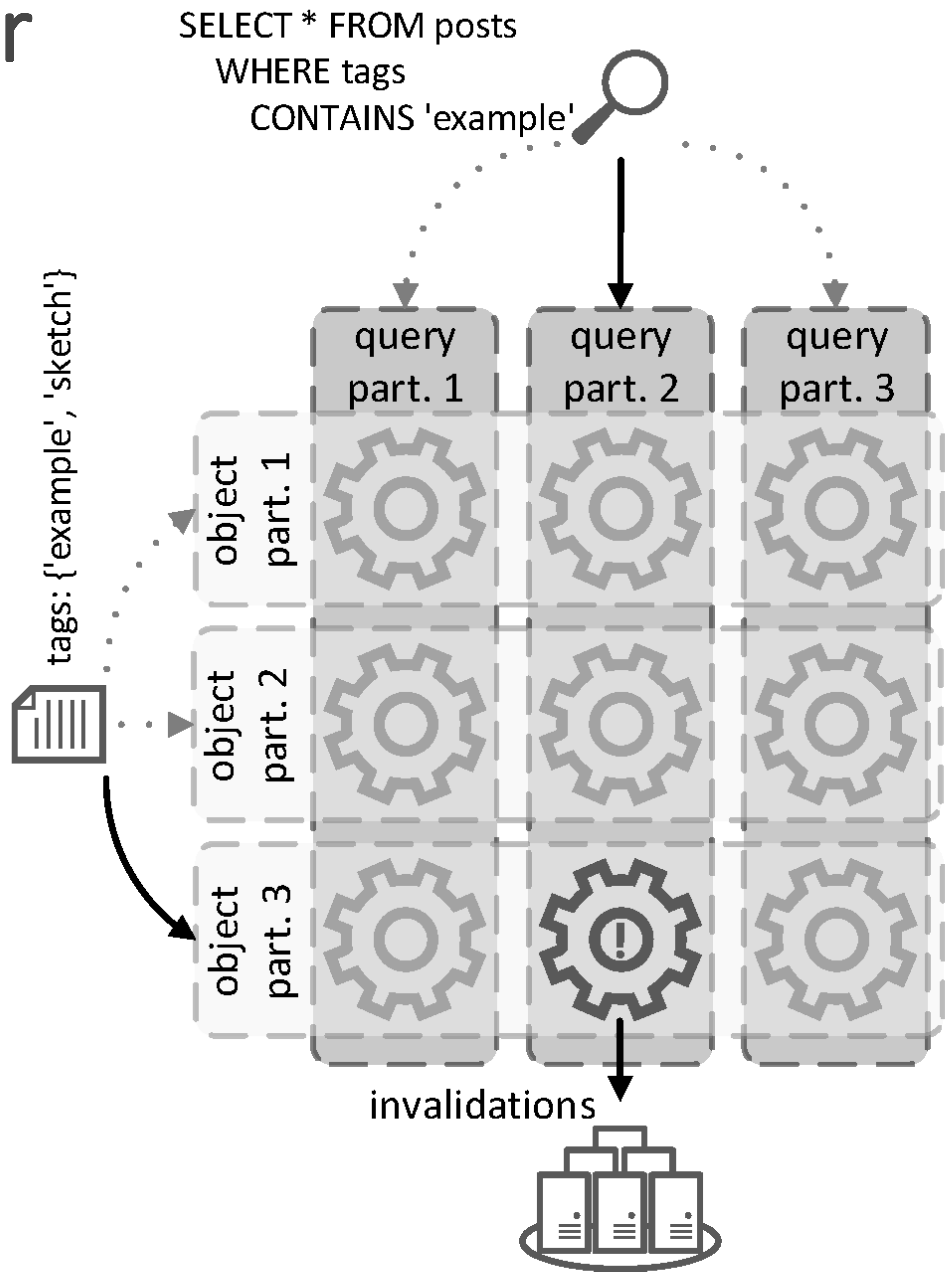
Architecture

Generalizing the Cache Sketch to Query Results



Streaming Layer
Query Matching

Design goals:
- Scalability
- Elasticity
- Low Latency



Optimal Query Representation

Id-Lists

$$\{id_1, id_2, id_3\}$$

Invalidated by: Add, Remove
→less invalidations

Performance: at least two network round-trips

Object-Lists

$$\{\,\{id{:}\,1, tag{:}\,'a'\}, \{id{:}\,2, tag{:}\,'b'\},\\ \{id{:}\,3, tag{:}\,'c'\}\}$$

Invalidated by: Add, Remove, Change

Performance: one round-trip
→lower latency

Cost-based decision model:

$$w\left(\frac{changes}{removes\ +\ adds\ +\ changes}\right) > 1 - \frac{1}{1 + \lceil resultsize / connections \rceil}$$

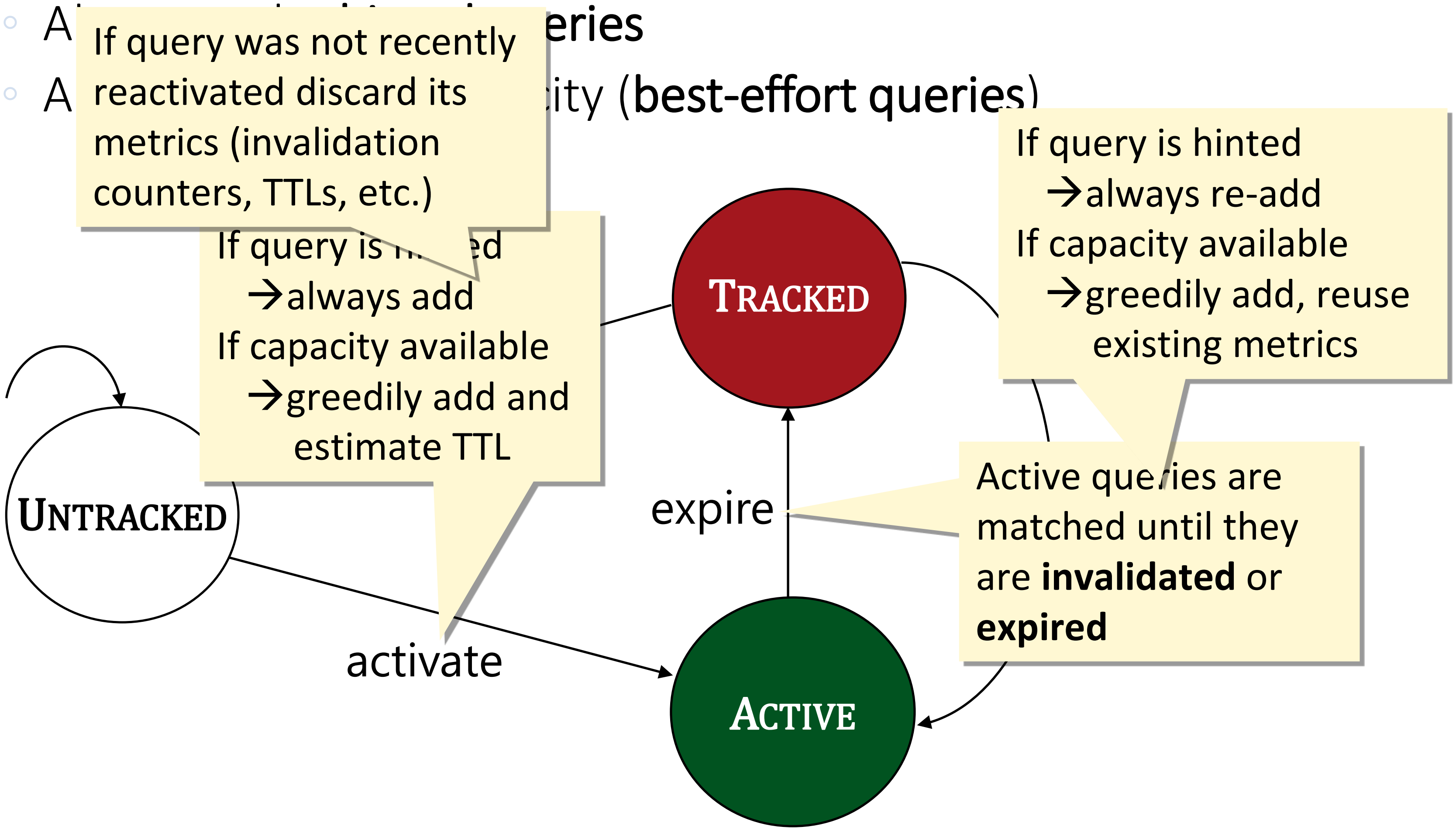
Fraction of avoided invalidations

avoided round-trips

Query Lifecycle

Disitributed Capacity Management

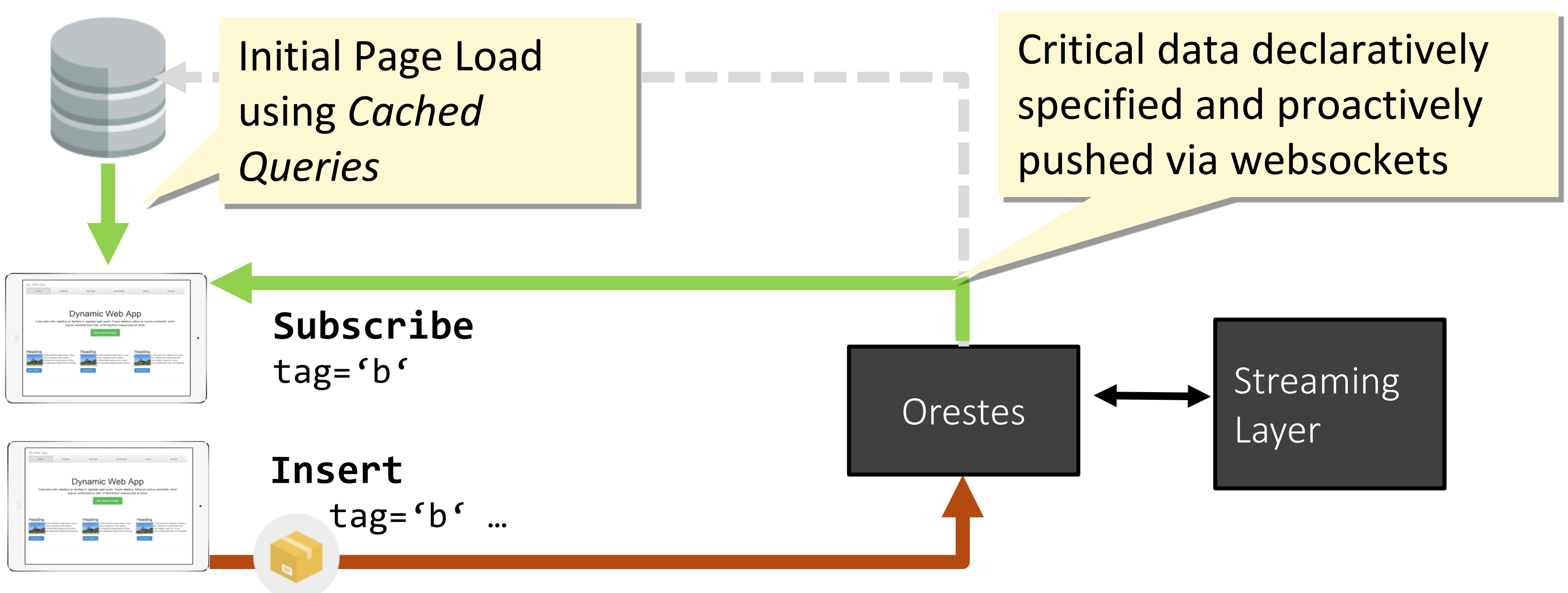
- Matching capacity is limited
  - A … **eries**
  - A … ity (**best-effort queries**)



Continuous Queries

Complementing Cached Queries

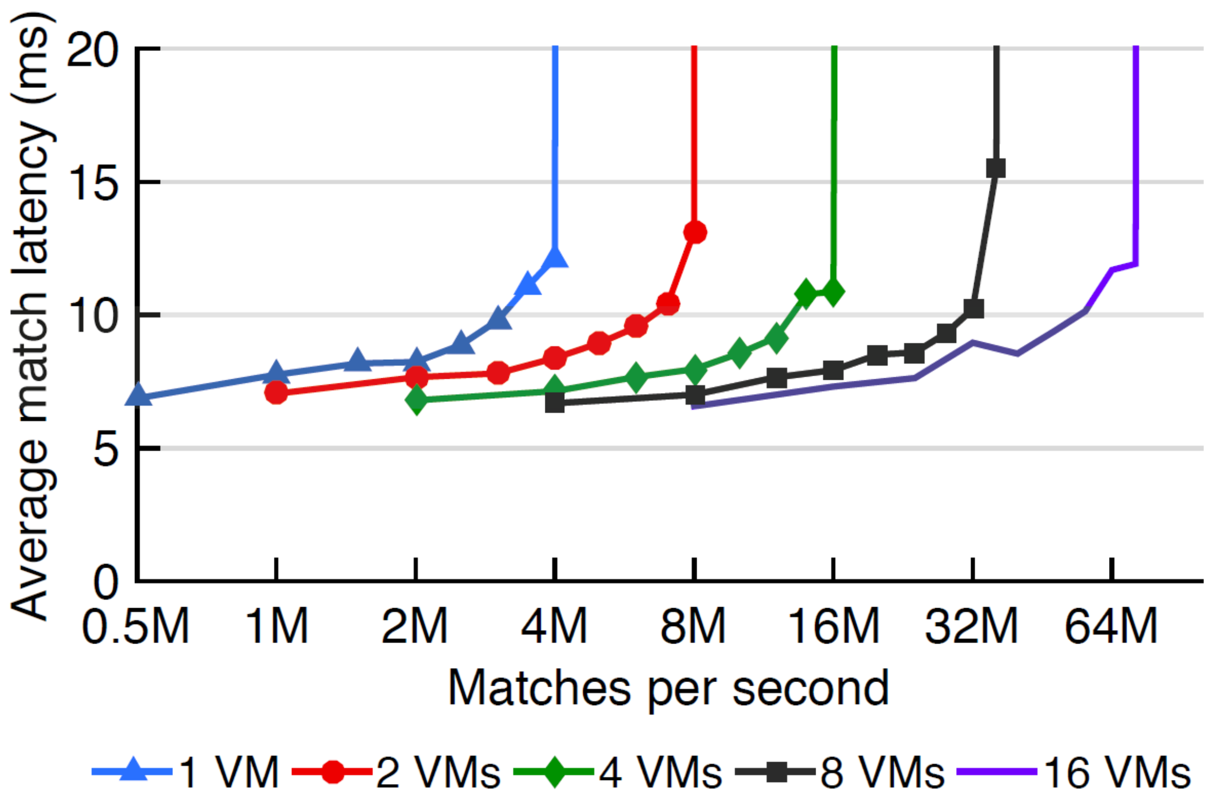
- Same streaming architecture can similarly notify applications (browsers) about query result changes
- **Application Pattern:**



Matching Performance

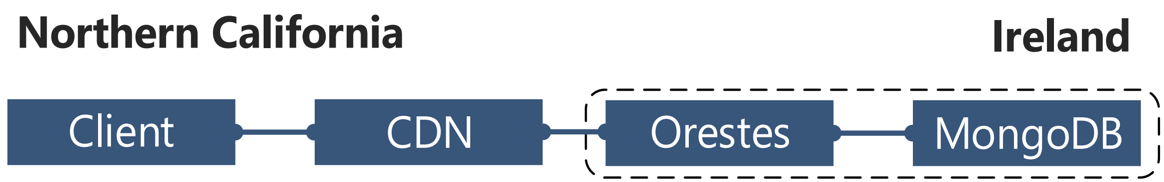
Latency of detecting invalidations

- Latency mostly < 15ms, scales linearly w.r.t. number of servers and number of tables

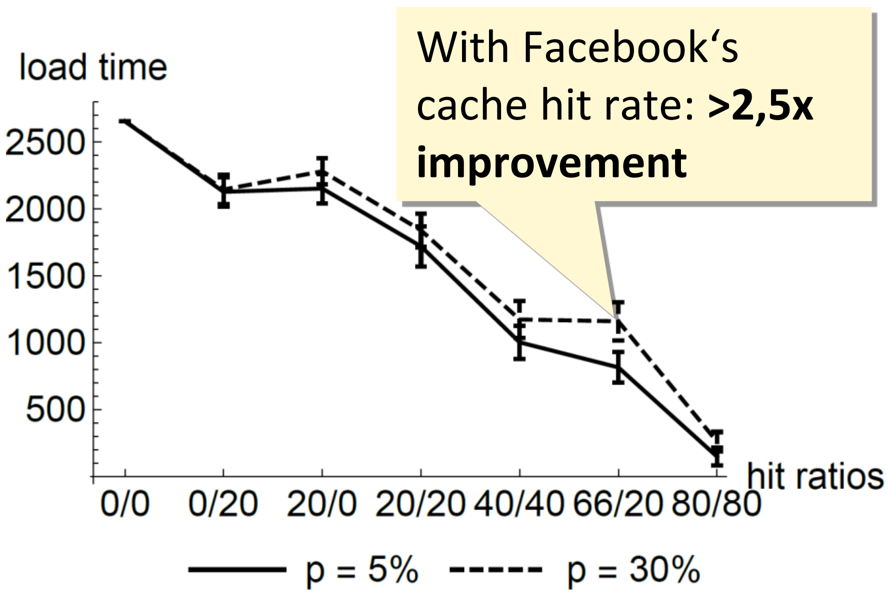


Performance

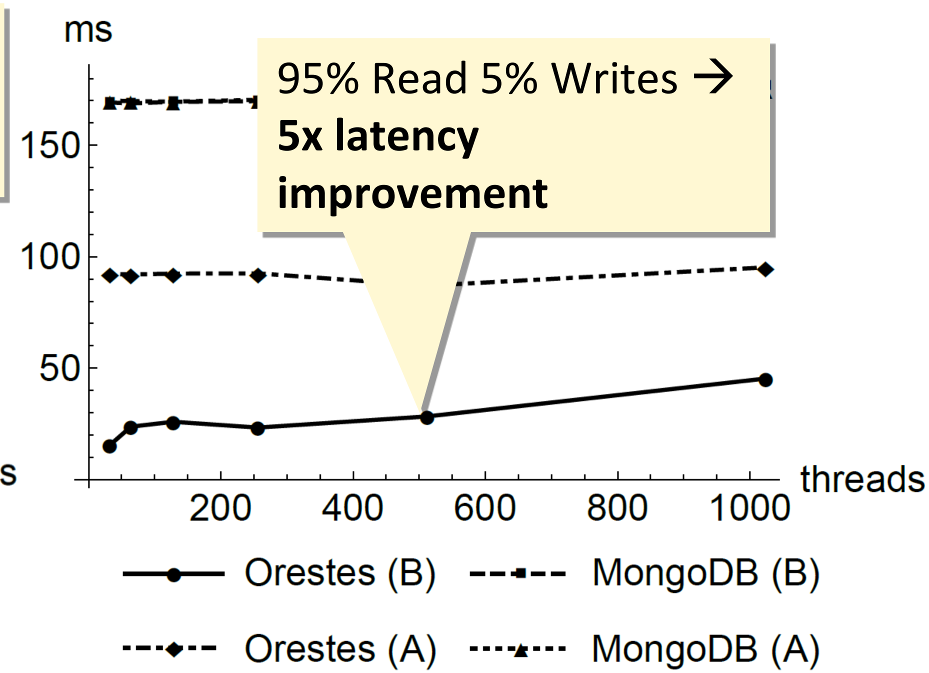
Setup:



Page load times with **cached initialization** (simulation):



Average Latency for YCSB Workloads A and B (real):

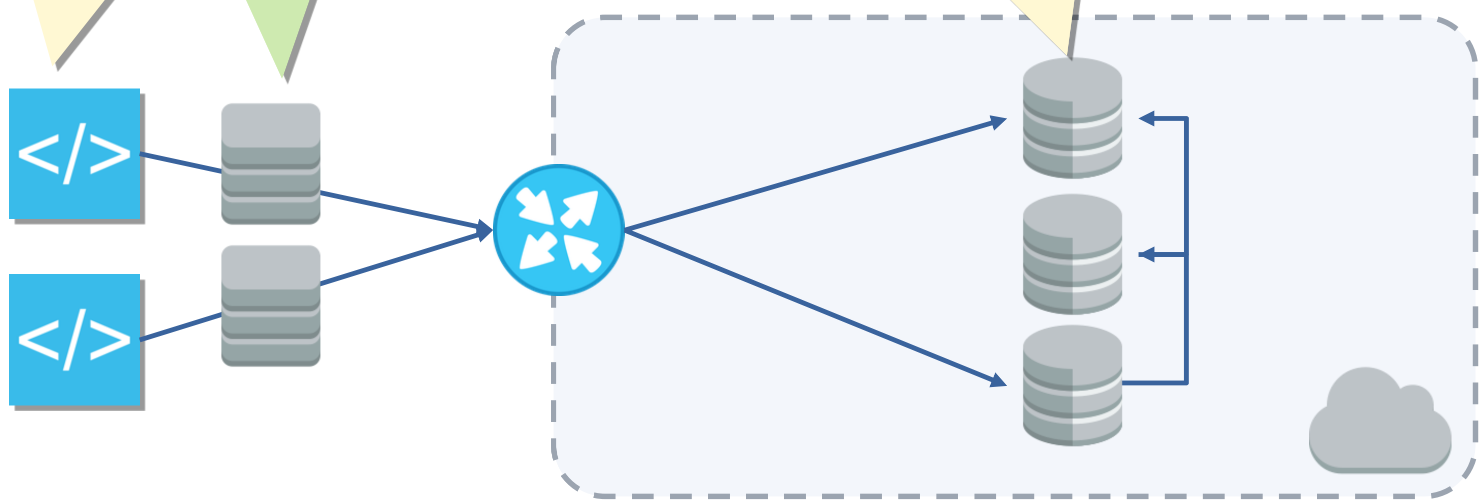


Low Latency

If the appl
distributec
fast datab

Transparent **end-to-end caching** using the Cache Sketch.

If one size *doesn't* fit all – how can **polyglot persistence** be leveraged on a declarative, automated basis?



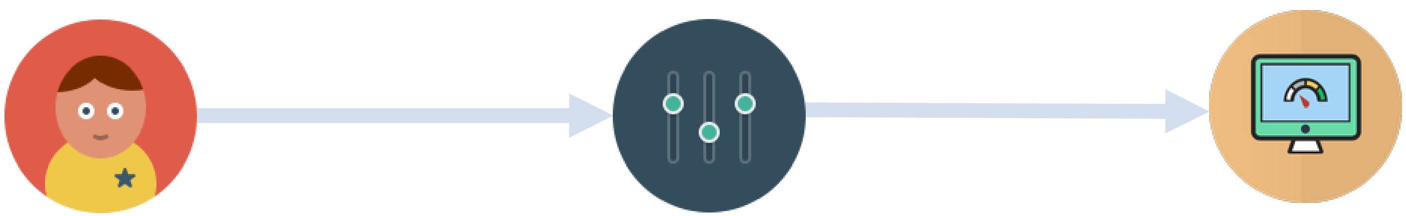
Towards Automated Polyglot Persistence
Necessary steps

- **Goal:**
  - Extend classic workload management to *polyglot persistence*
  - Leverage heterogeneous (NoSQL) databases

1. Requirements

2. Resolution

3. Mediation



Tenant specifies requirements as Service-Level-Agreements

Find or provision a suitable combination of databases

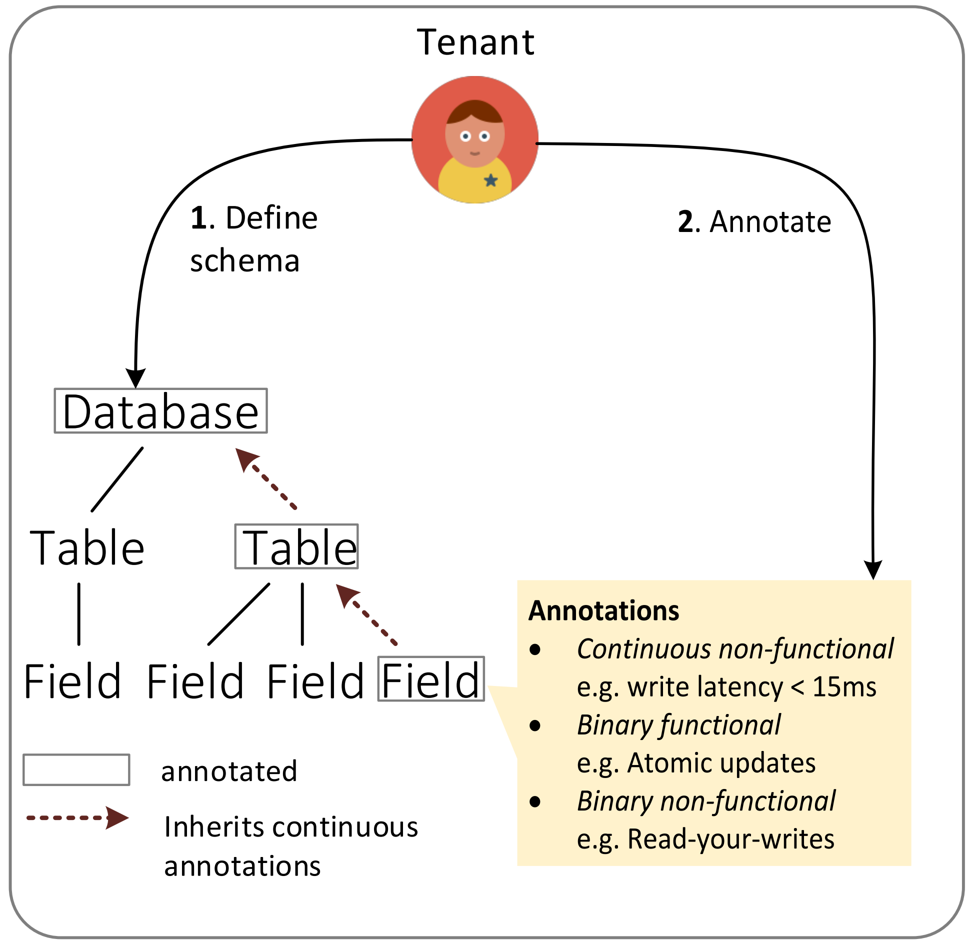
Mediate data and database operations

Michael Schaarschmidt, Felix Gessert, Norbert Ritter: *Towards Automated Polyglot Persistence*. BTW 2015

Step I - Requirements

Expressing the application‘s needs

Annotation	Type	Annotated at
Read Availability	Continuous	*
Write Availability	Continuous	*
Read Latency	Continuous	*
Write Latency	Continuous	*
Write Throughput	Continuous	*
Data Vol. Scalability	Non-Functional	Field/Class/DB
Write Scalability	Non-Functional	Field/Class/DB
Read Scalabilty	Non-Functional	Field/Class/DB
Elasticity	Non-Functional	Field/Class/DB
Durability	Non-Functional	Field/Class/DB
Replicated	Non-Functional	Field/Class/DB
Linearizability	Non-Functional	Field/Class
Read-your-Writes	Non-Functional	Field/Class
Causal Consistency	Non-Functional	Field/Class
Writes follow reads	Non-Functional	Field/Class
Monotonic Read	Non-Functional	Field/Class
Monotonic Write	Non-Functional	Field/Class
Scans	Functional	Field
Sorting	Functional	Field
Range Queries	Functional	Field
Point Lookups	Functional	Field
ACID Transactions	Functional	Class/DB
Conditional Updates	Functional	Field
Joins	Functional	Class/DB
Analytics Integration	Functional	Field/Class/DB
Fulltext Search	Functional	Field
Atomic Updates	Functional	Field/Class

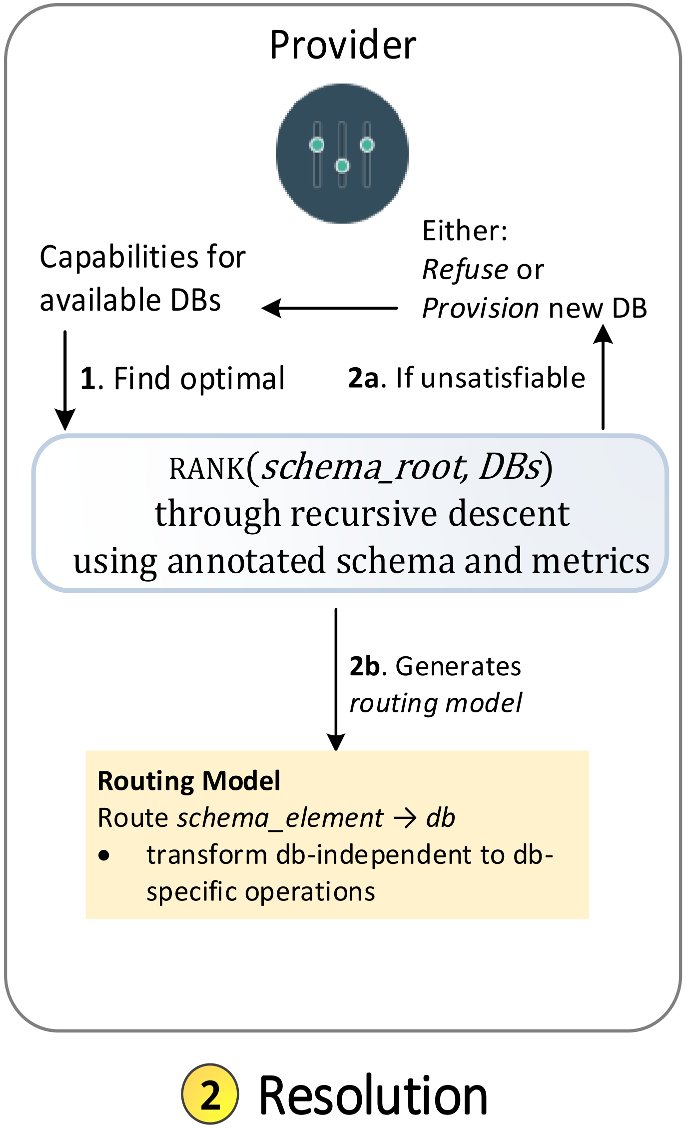


1 Requirements

Step II - Resolution

Finding the best database

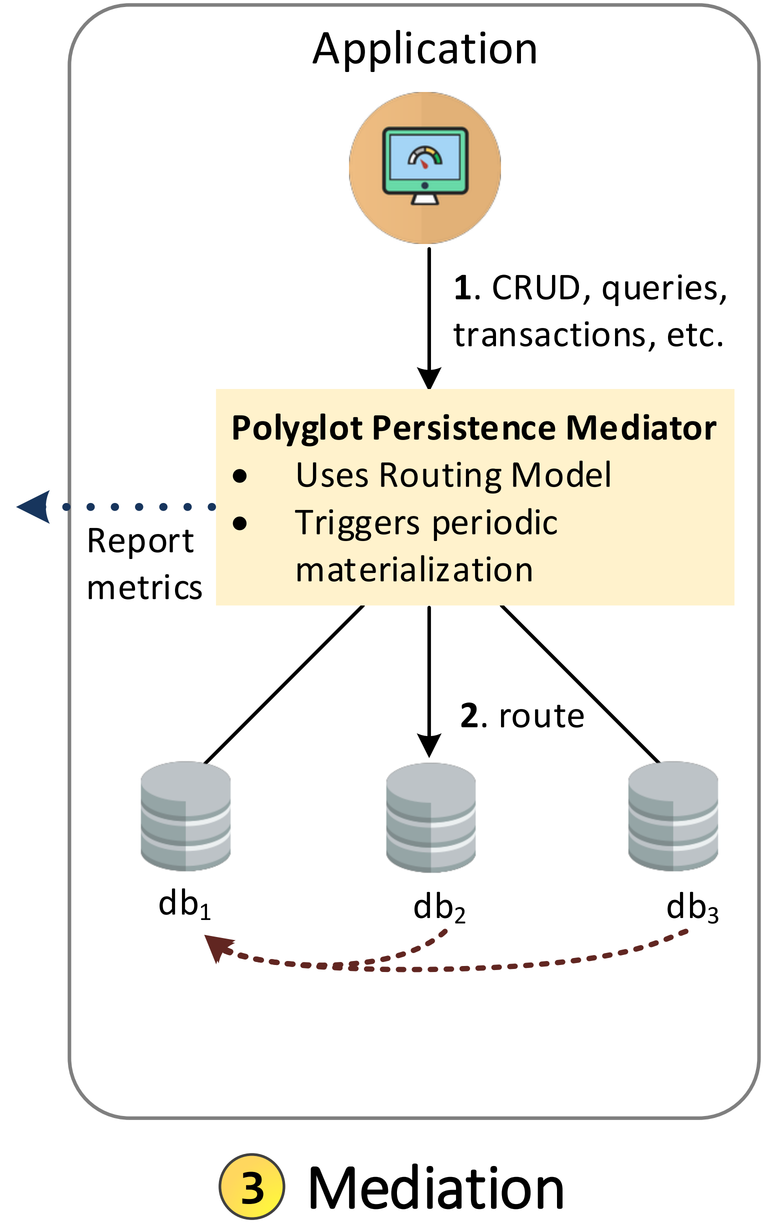
- The Provider resolves the requirements
- RANK: scores available database systems
- **Routing Model**: defines the optimal mapping from schema elements to databases



Step III - Mediation

Routing data and operations

- The PPM routes data
- **Operation Rewriting:** translates from abstract to database-specific operations
- **Runtime Metrics**: Latency, availability, etc. are reported to the resolver
- **Primary Database Option**: All data periodically gets materialized to designated database



Evaluation: News Article

Prototype of Polyglot Persistence Mediator in ORESTES

Scenario: news articles with impression counts
Objectives: low-latency top-k queries, high-throughput on counter, article-queries

Evaluation: News Article

Prototype built on Orestes

Scenario: news articles with impression counts
Objectives: low-latency top-k queries, high-throughput on counter, article-queries

Counter updates kill performance

Evaluation: News Article

Prototype built on ORESTES

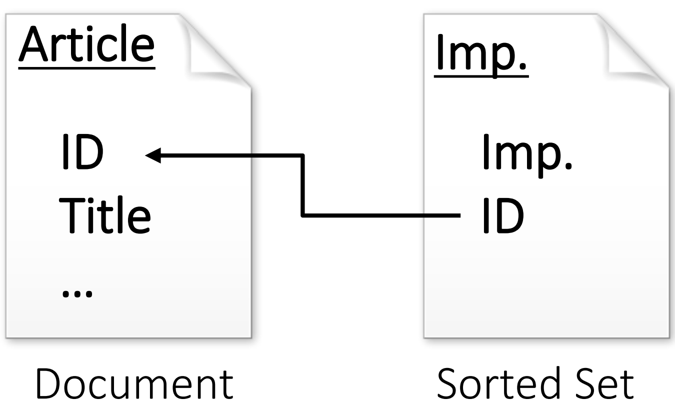
Scenario: news articles with impression counts
Objectives: low-latency top-k queries, high-throughput on counter, article-queries

No powerful queries

Evaluation: News Article

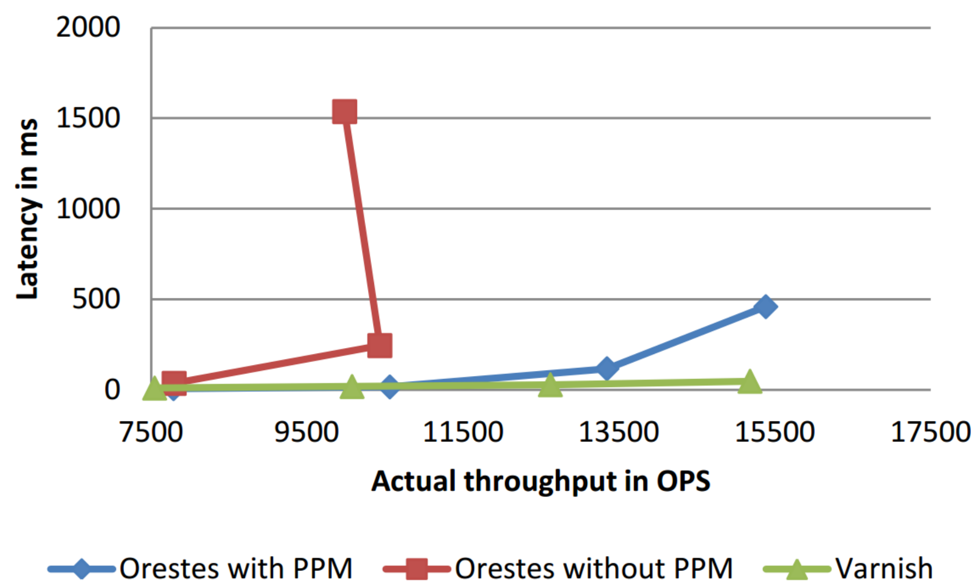
Prototype built on ORESTES

Scenario: news articles with impression counts
Objectives: low-latency top-k queries, high-throughput on counter, article-queries



mongoDB redis

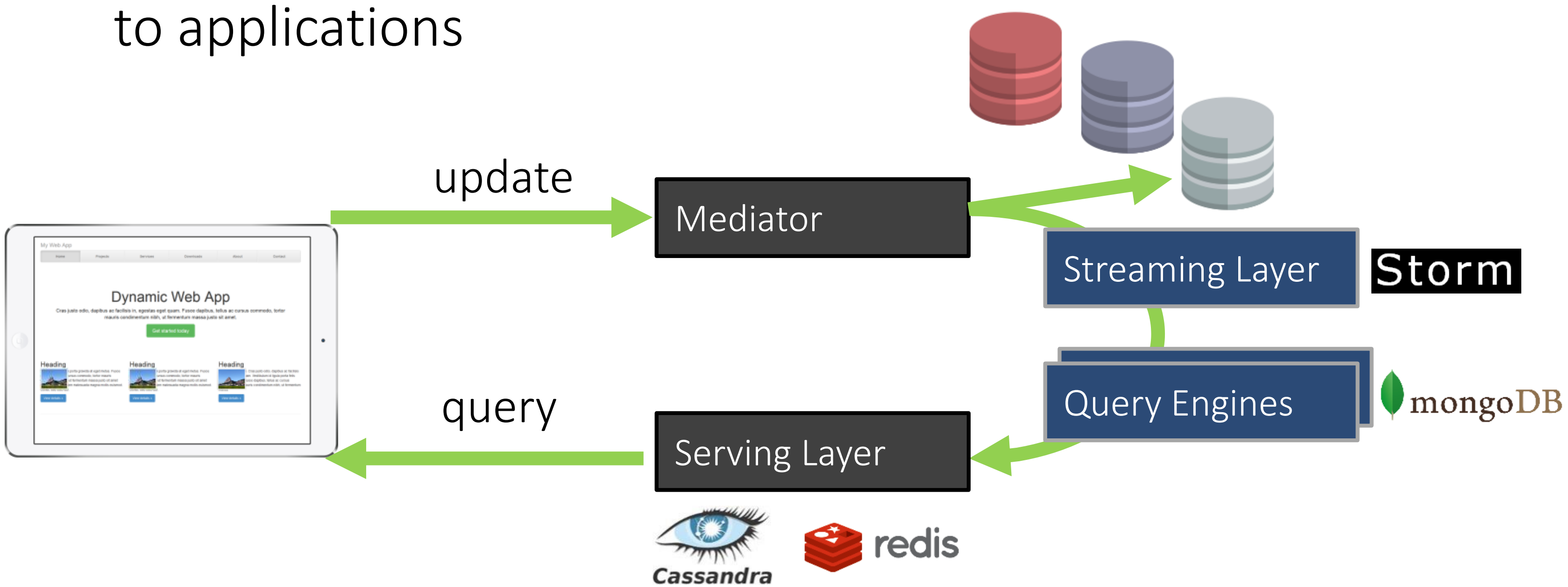
Found Resolution



Polyglot Materialized Views

Arbitrary Queries over arbitrary databases

- Approach:
  - **Mediator** emits change data stream (after-images)
  - **Streaming layer** maintains registered materialized views using pluggable query engines
  - **Serving layer** stores materialized views and serves them to applications



Outline

- Motivation
- ORESTES: a Cloud-Database Middleware
- Solving Latency and Polyglot Storage
- **Wrap-up**
  - Current/Future Work
  - Summary
  - Putting ORESTES into practice

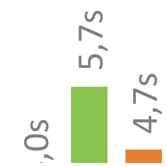
Summary

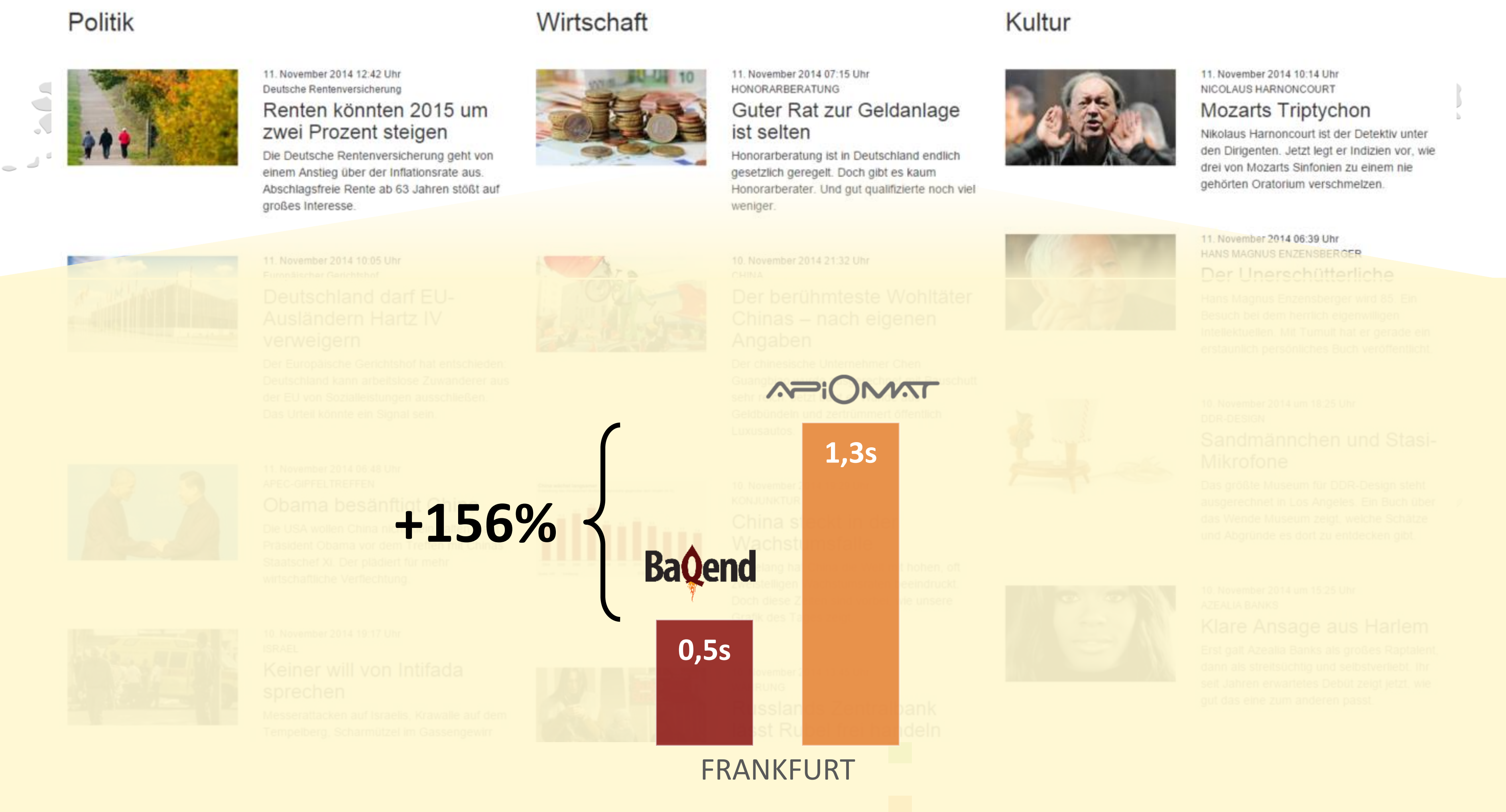
- **Cache Sketch** (web caching for database services):
  - Consistent (Δ-atomic) *expiration-based* caching
  - *Invalidation-based* caching with minimal purges
- **Query Caching:**
  - Invalidations and Cache Sketch updates in real-time
  - Cache-optimal representation of results
- **Continuous & Materialized Queries**
  - Real-time updates to query results
- **Polyglot Persistence Mediator**:
  - SLA-based routing of queries and data to appropriate database systems


BaQend
Build faster Apps faster.
www.baqend.com

Page-Load Times

What impact does the Cache Sketch have?





Thanks a lot!

gessert/ritter@informatik.uni-hamburg.de

baqend.com, orestes.info