

# **Tutorial on Scalable Cloud-Databases in Research and Practice**

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

# Outline



## Motivation



## ORESTES: a Cloud-Database Middleware

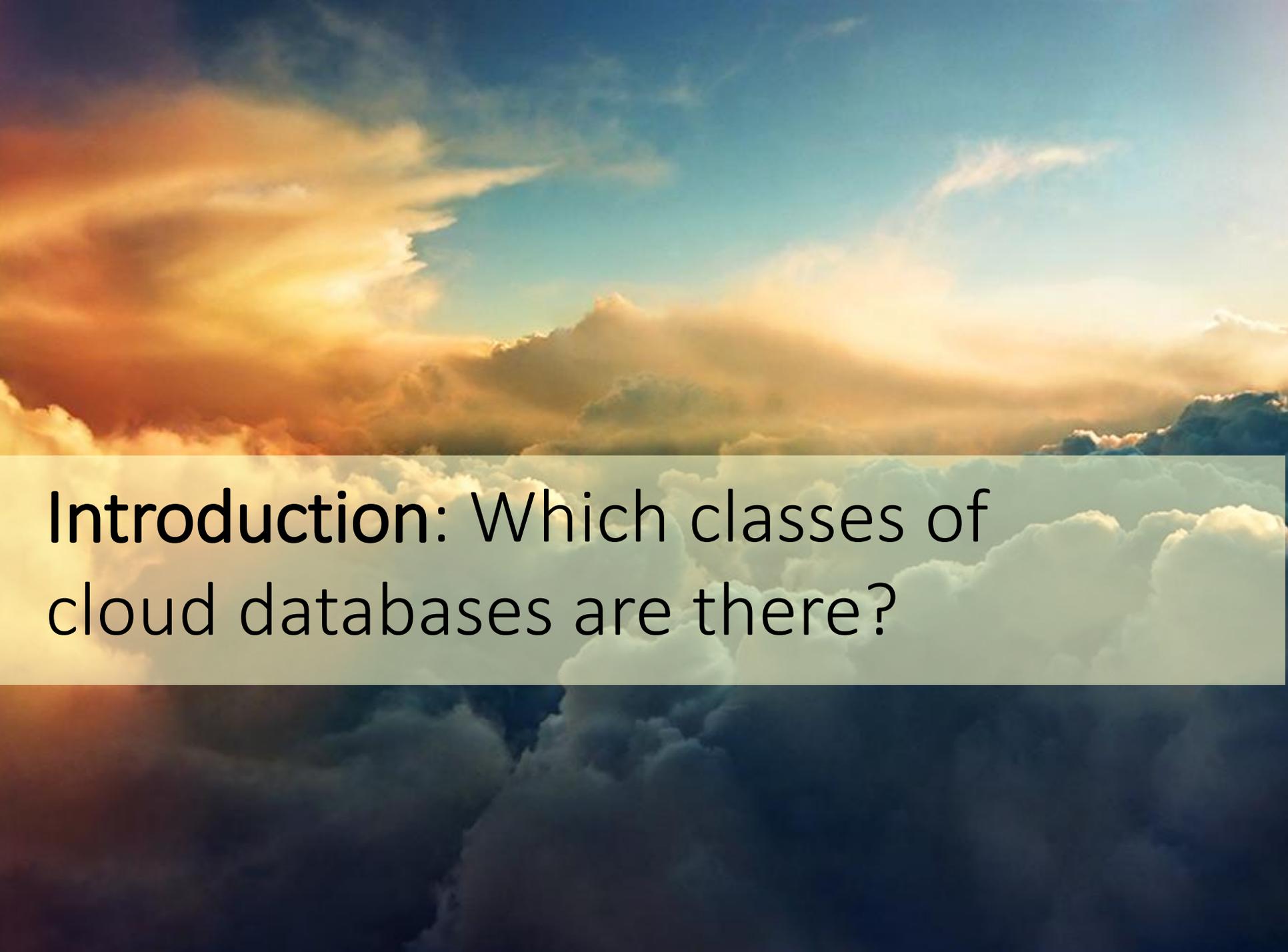


## Solving Latency and Polyglot Storage



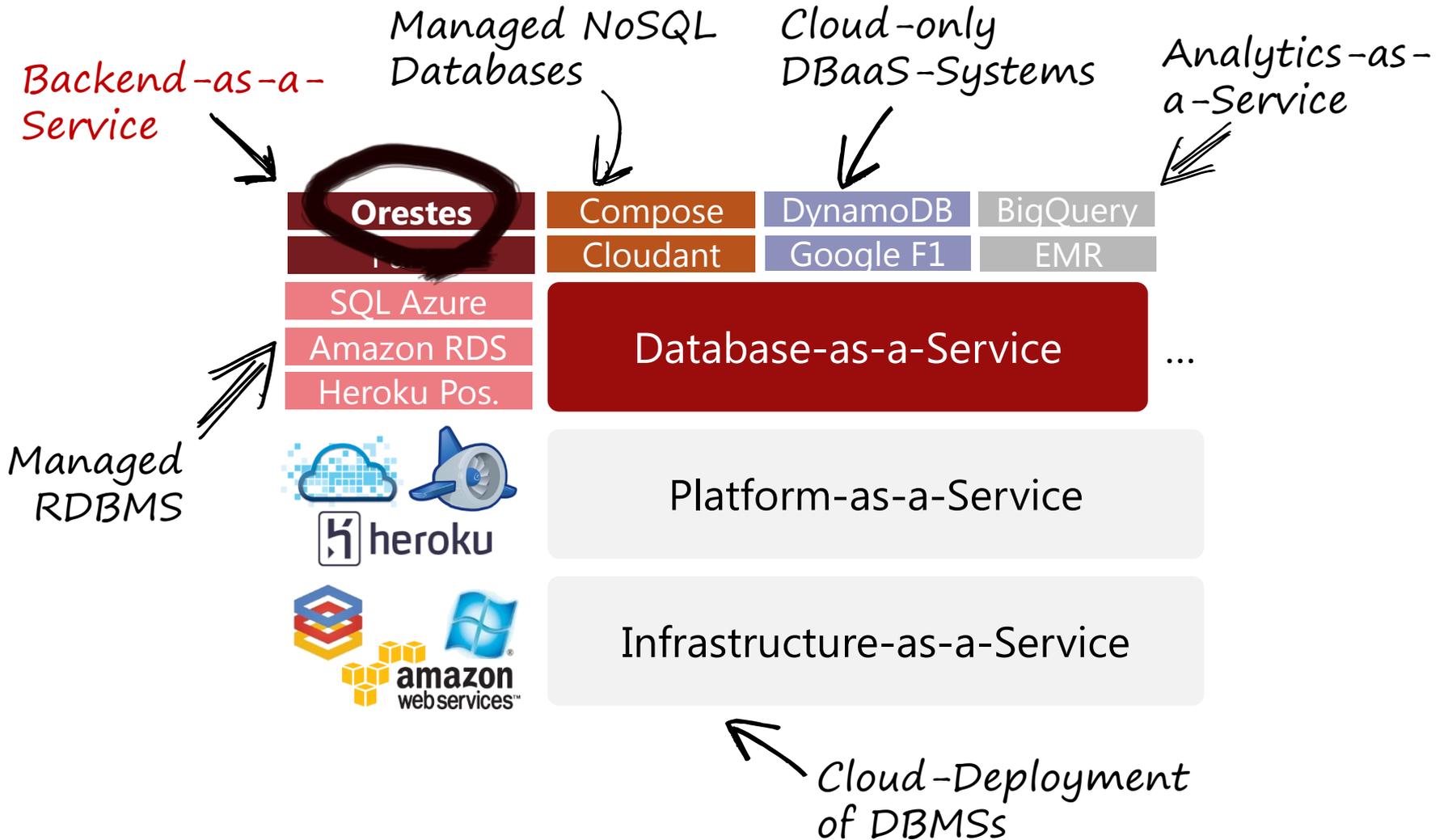
## Wrap-up

- Overview
- The New Field *Cloud Data Management*
- Cloud Database Models
- Research Challenges



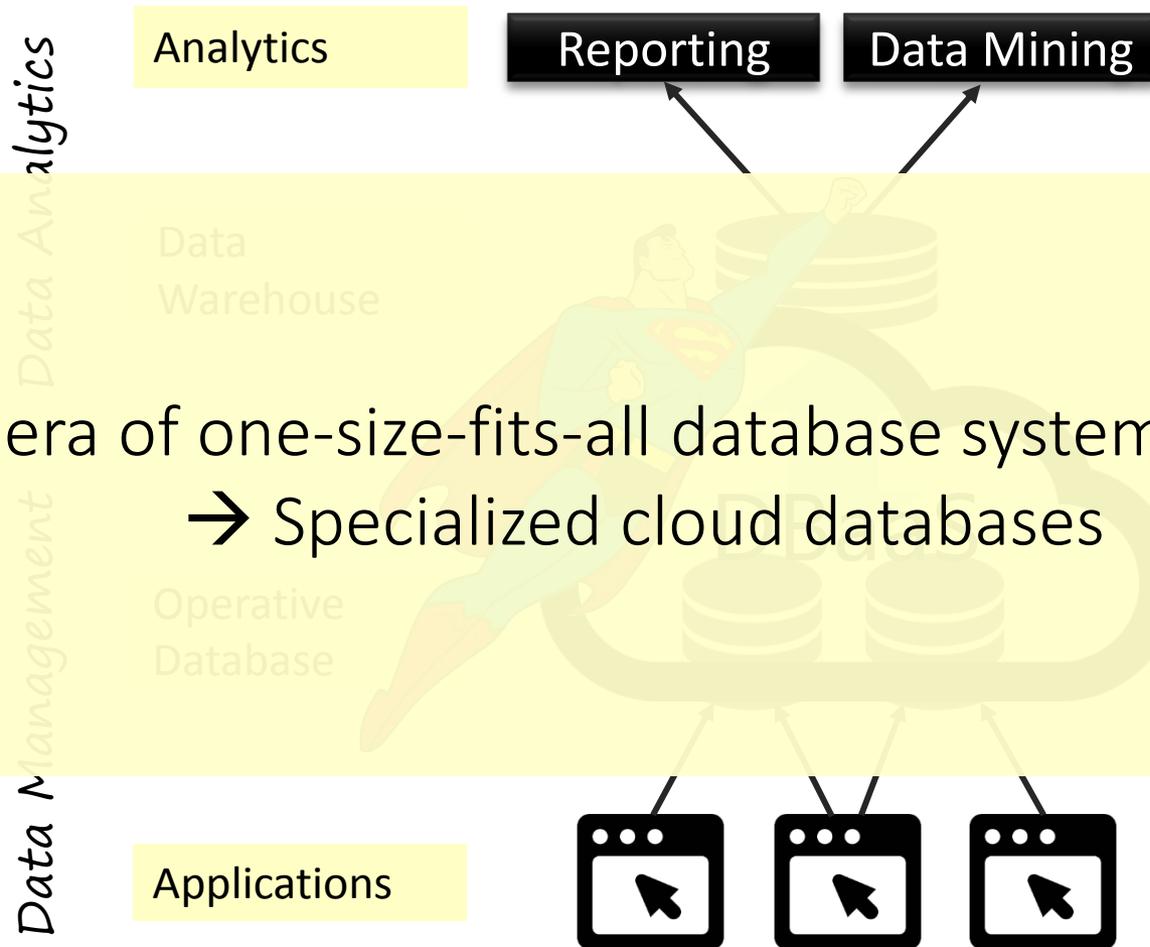
**Introduction:** Which classes of cloud databases are there?

# Cloud Databases



# Architecture

Typical Data Architecture:



The era of one-size-fits-all database systems is over  
→ Specialized cloud databases

# Database Sweetspots



*RDBMS*

General-purpose  
ACID transactions



*Wide-Column Store*

Long scans over  
structured data



*Graph Database*

Graph algorithms  
& queries



*Parallel DWH*

Aggregations/OLAP for  
massive data amounts



*Document Store*

Deeply nested  
data models



*In-Memory KV-Store*

Counting & statistics



*NewSQL*

High throughput  
relational OLTP



*Key-Value Store*

Large-scale  
session storage



*Wide-Column Store*

Massive user-  
generated content

# Cloud-Database Sweetspots



*Realtime BaaS*

Communication and collaboration



*Wide-Column Store*

Very large tables



*Managed NoSQL*

Full-Text Search



*Managed RDBMS*

General-purpose  
ACID transactions



*Wide-Column Store*

Massive user-generated content



*Object Store*

Massive File  
Storage



*Managed Cache*

Caching and  
transient storage



*Backend-as-a-Service*

Small Websites  
and Apps

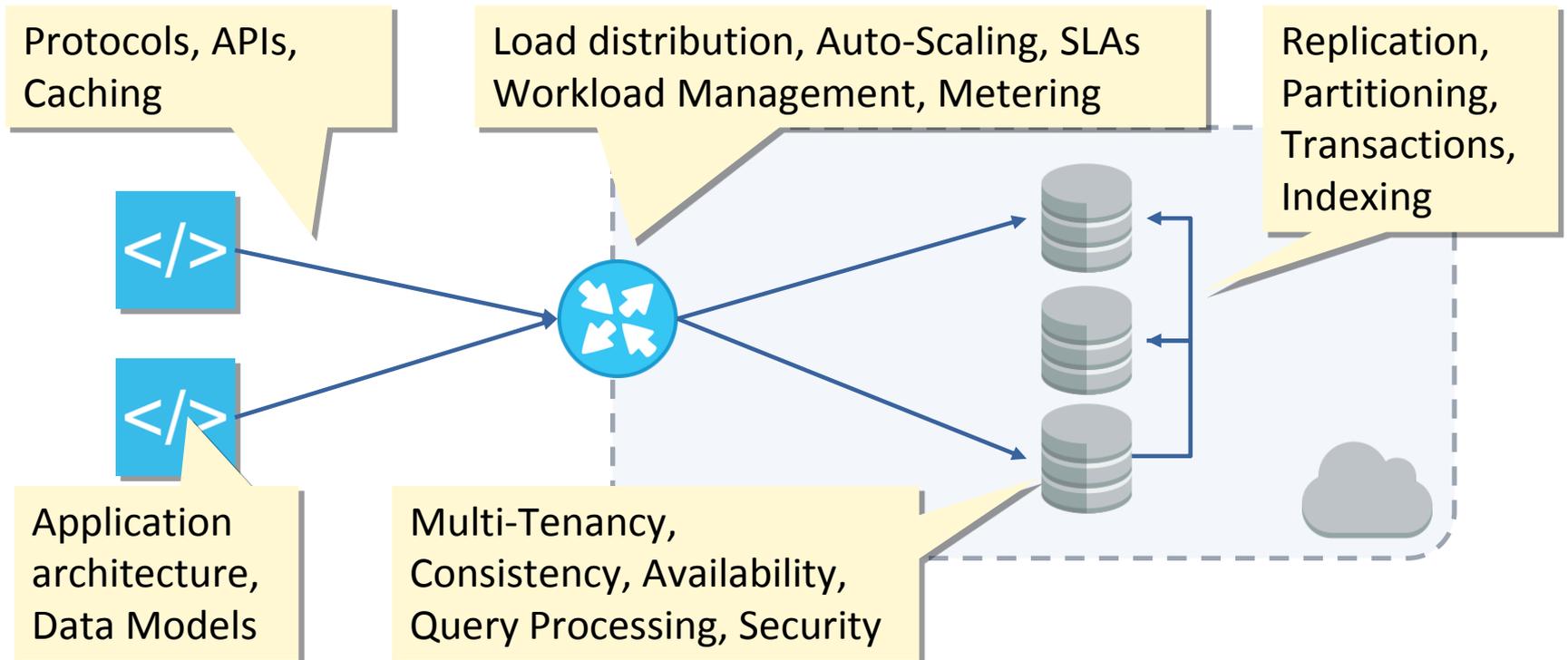


*Hadoop-as-a-Service*

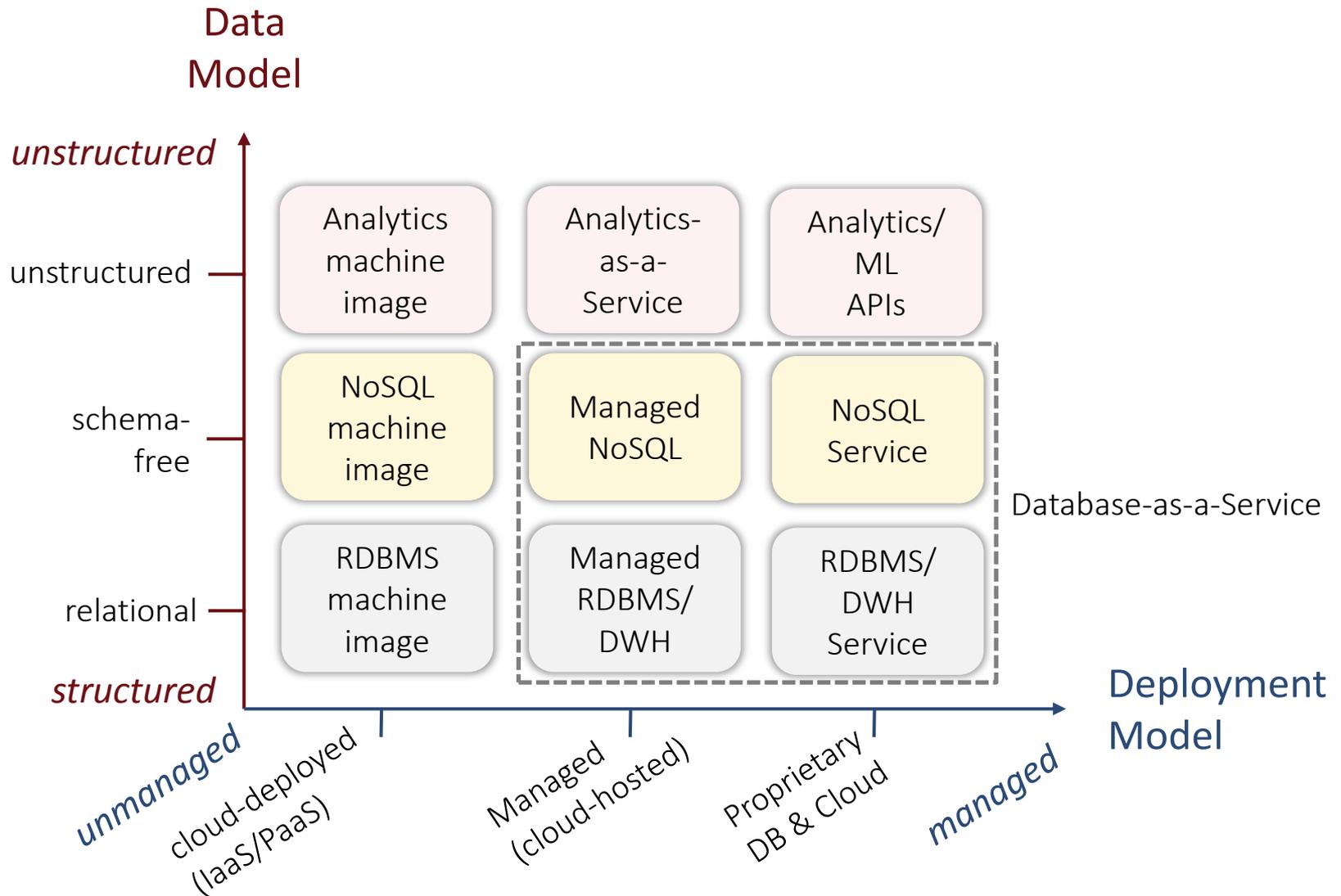
Big Data Analytics

# Cloud Data Management

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

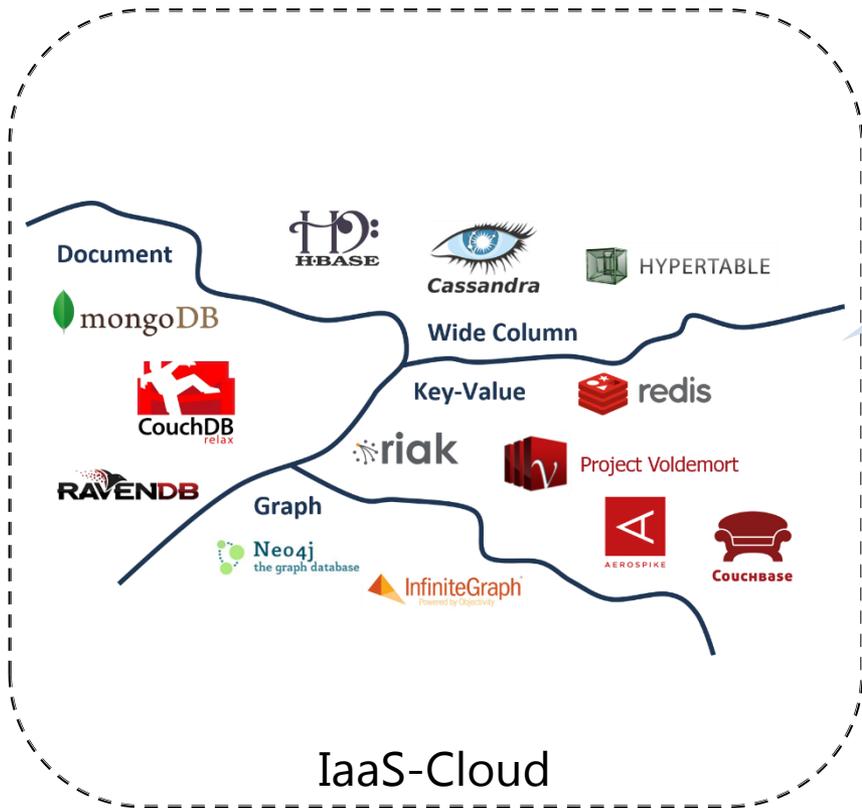


# Cloud-Database Models



# Cloud-Deployed Database

Database-image provisioned in IaaS/PaaS-cloud



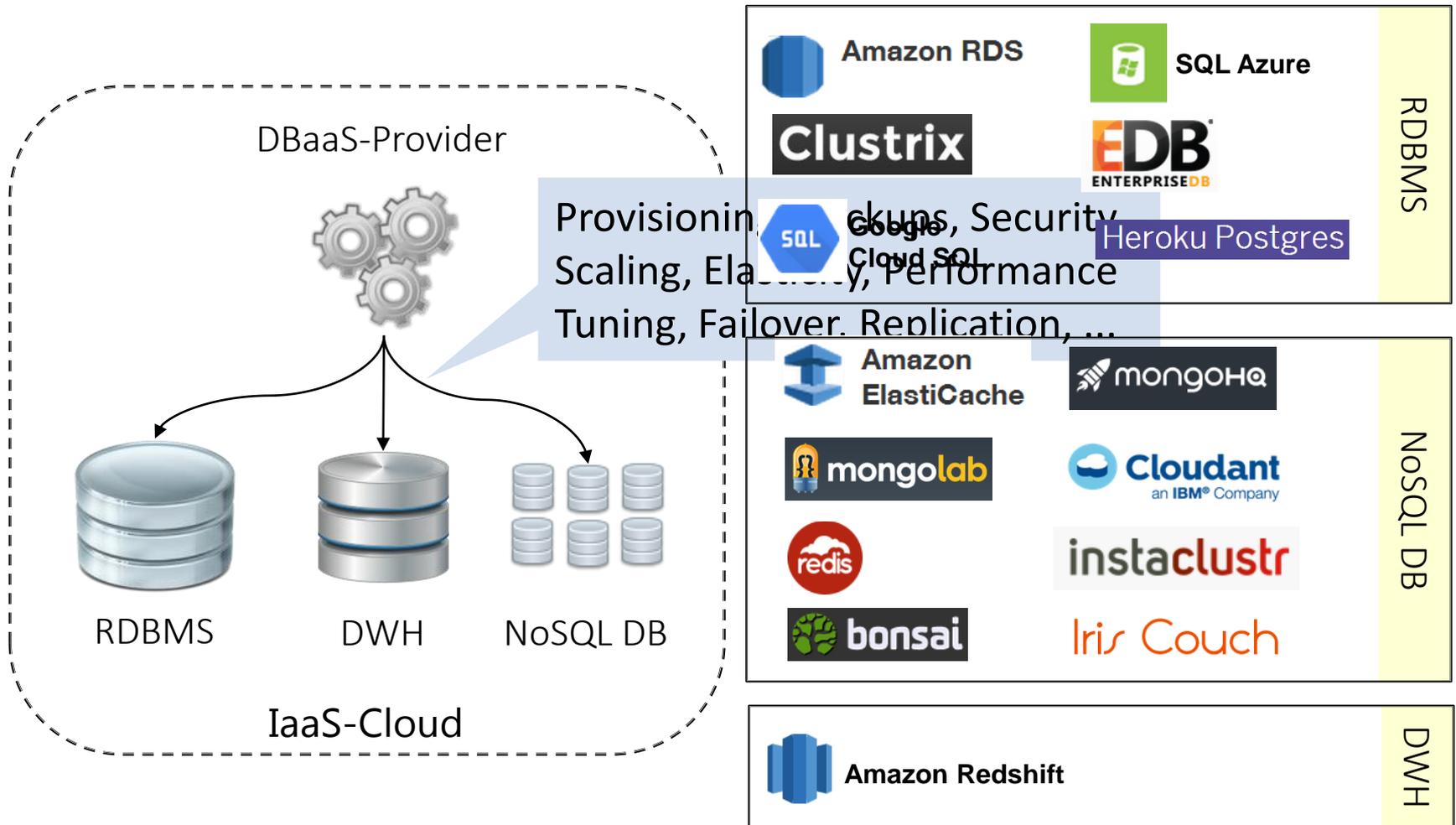
IaaS/PaaS deployment of database system

## Does not solve:

Provisioning, Backups, Security, Scaling, Elasticity, Performance Tuning, Failover, Replication, ...

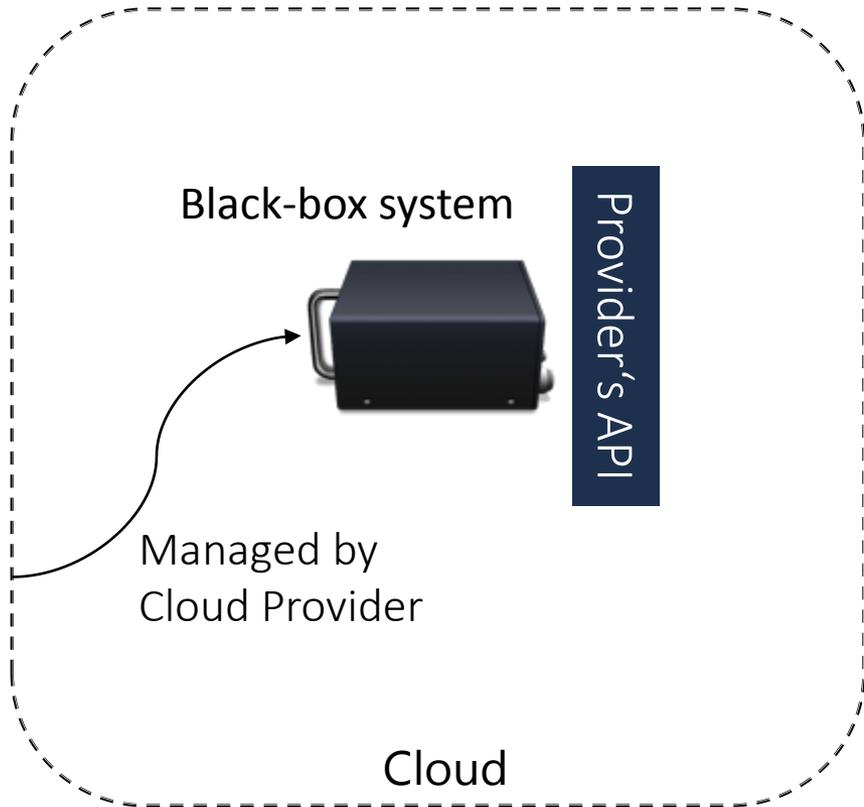
# Managed RDBMS/DWH/NoSQL DB

Cloud-hosted database



# Proprietary Cloud Database

Designed for and deployed in vendor-specific cloud environment

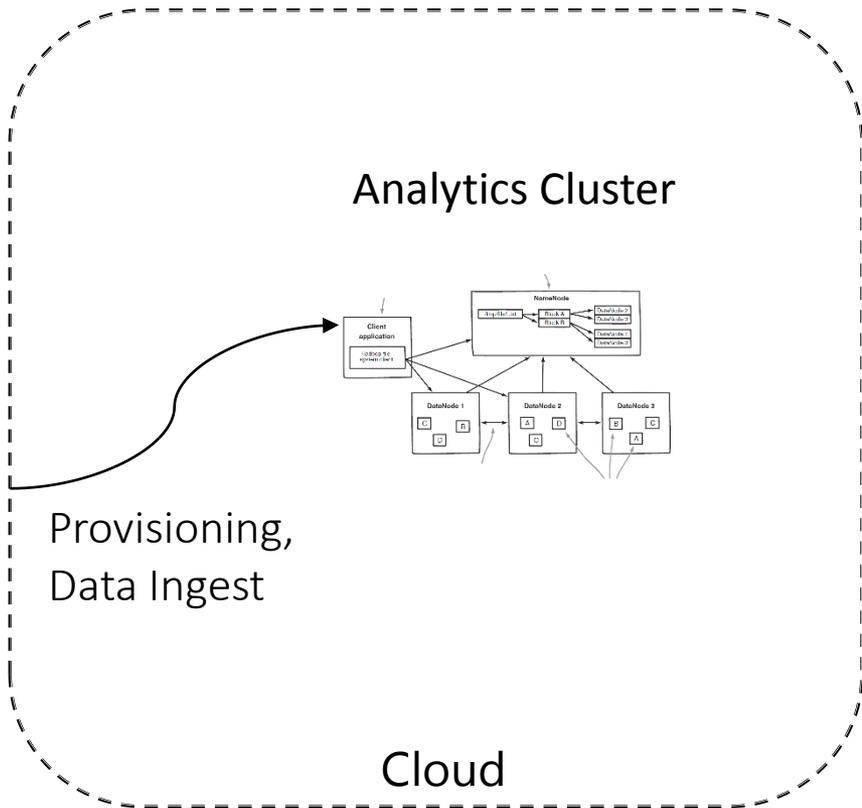


 <b>Amazon SimpleDB</b>	 <b>Amazon DynamoDB</b>	Database
 <b>Google Cloud Datastore</b>	 <b>Azure Tables</b>	
 <b>Database.com</b>	 <b>ORCHESTRATE</b>	
<b>BigTable, Megastore, Spanner, F1, Dynamo, PNuts, Relational Cloud, ...</b>		

 <b>Azure Blob Storage</b>	 <b>Openstack Swift</b>	Object Store
 <b>Amazon S3</b>	 <b>Google Cloud Storage</b>	

# Analytics-as-a-Service

Analytic frameworks and machine learning with service APIs



**Amazon Elastic  
MapReduce**



**Azure  
HDInsight**

Analytics



**Google  
BigQuery**

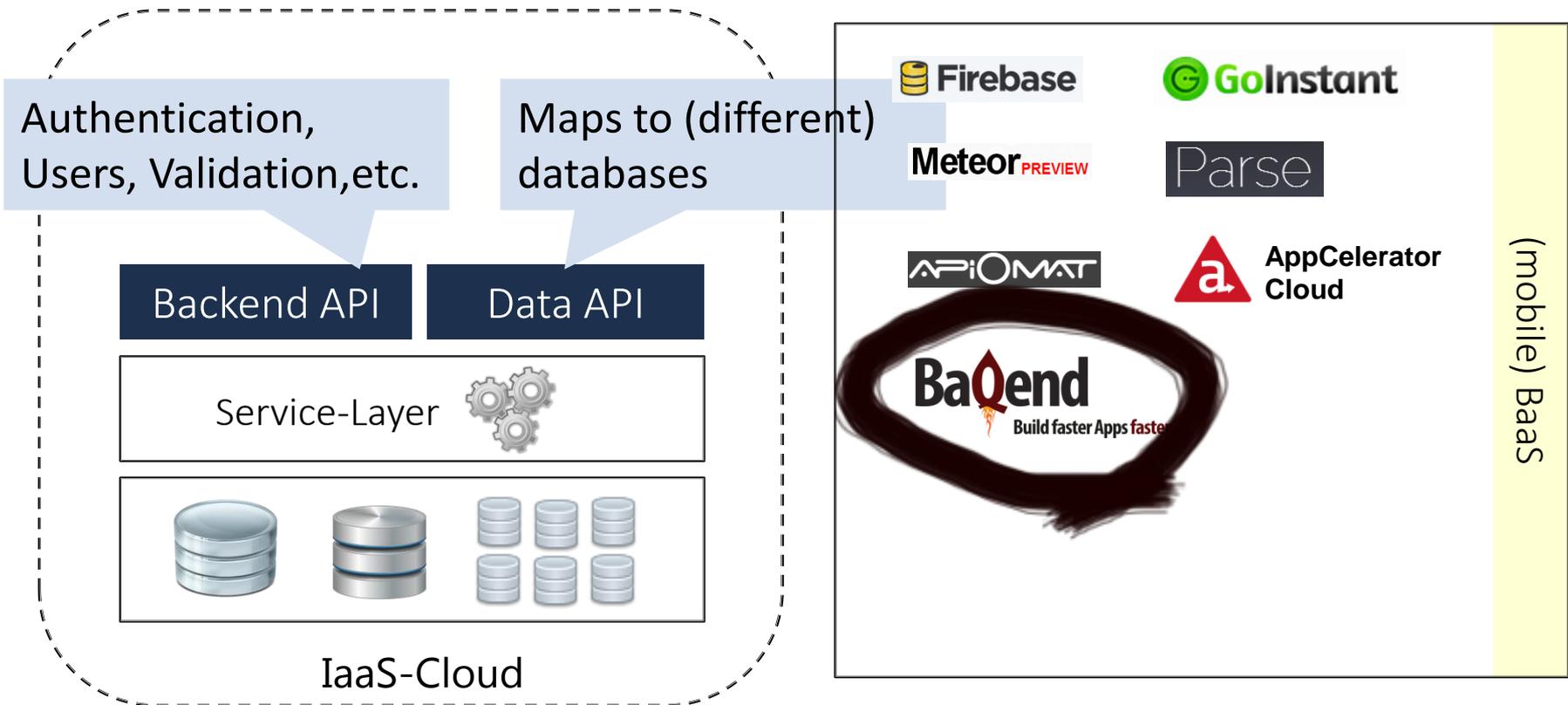


**Google  
Prediction API**

ML

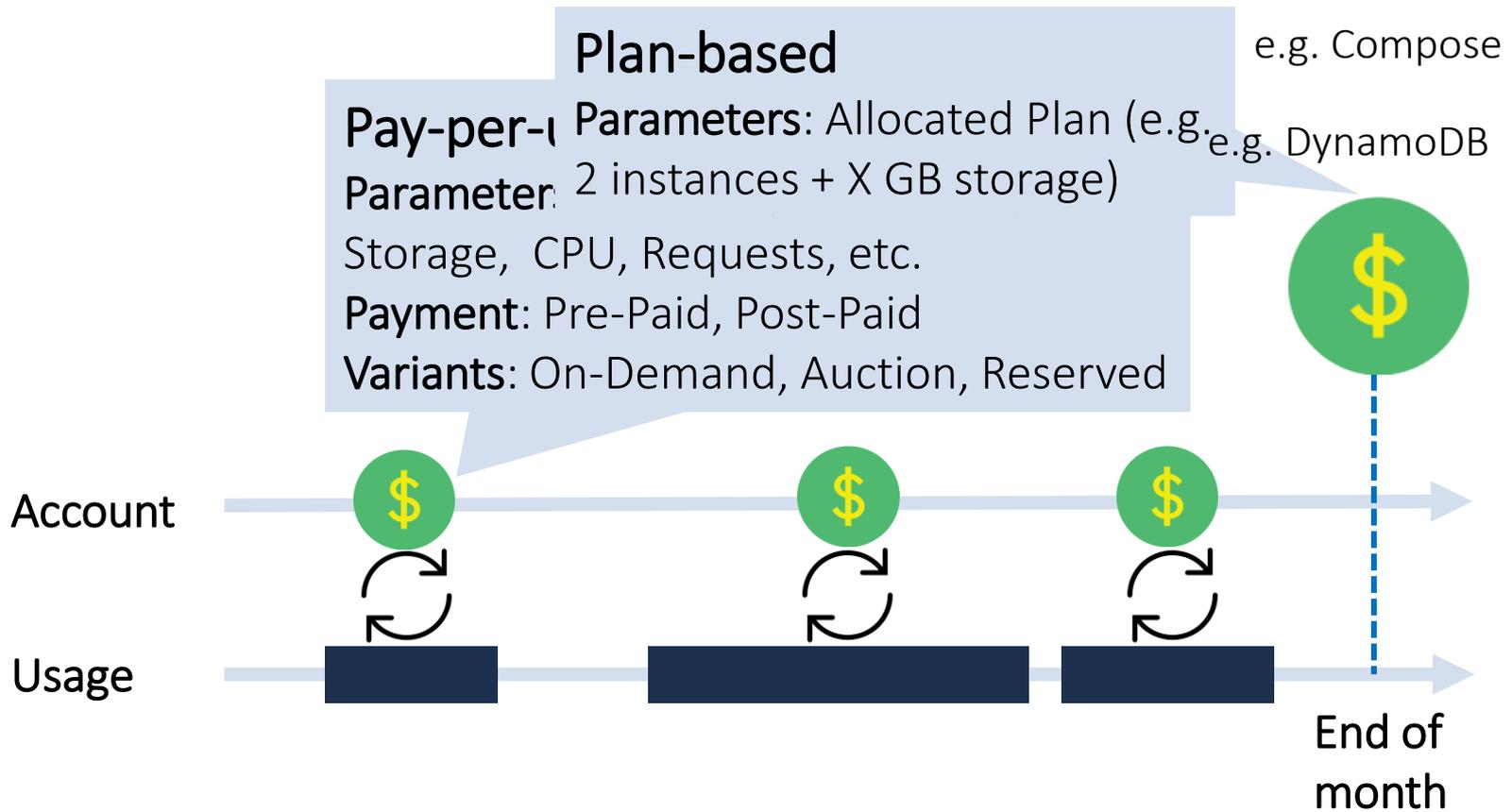
# Backend-as-a-Service

DBaaS with embedded custom and predefined application logic



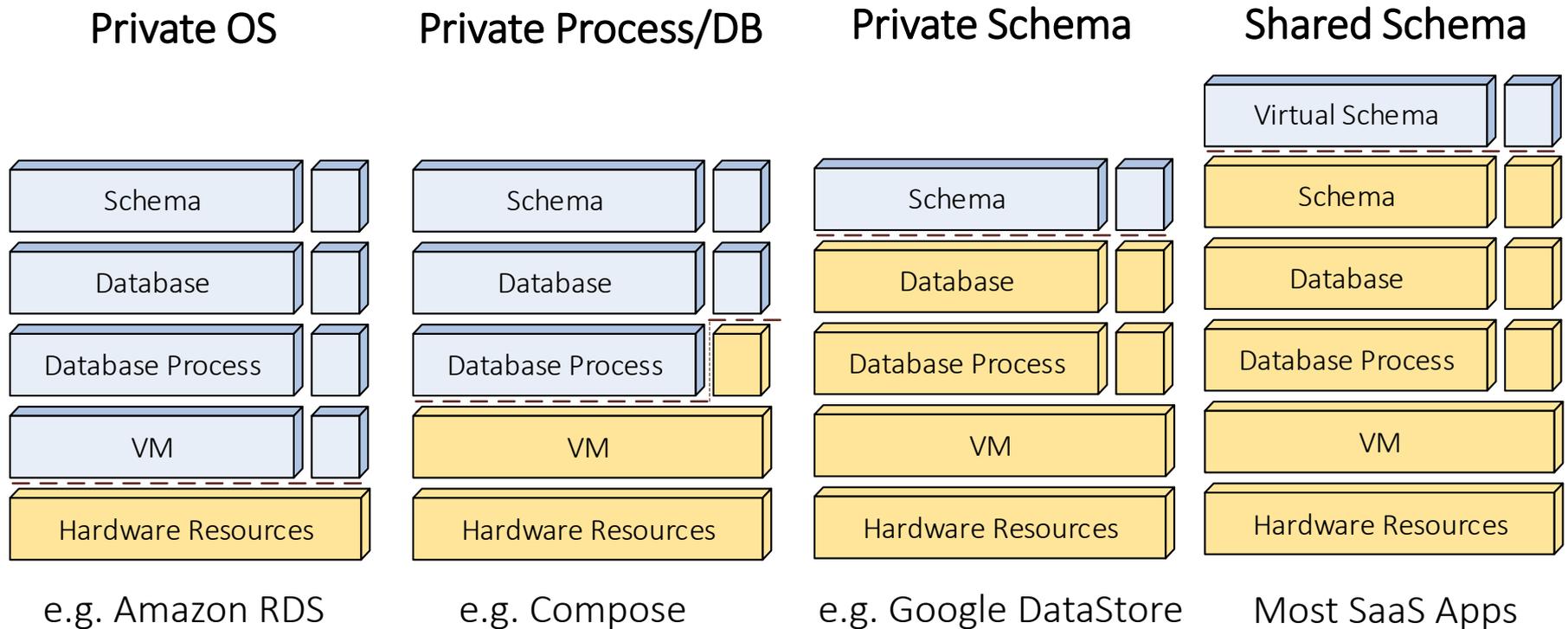
# Pricing Models

## Pay-per-use and plan-based



# Database-as-a-Service

## Approaches to Multi-Tenancy



# Multi-Tenancy: Trade-Offs

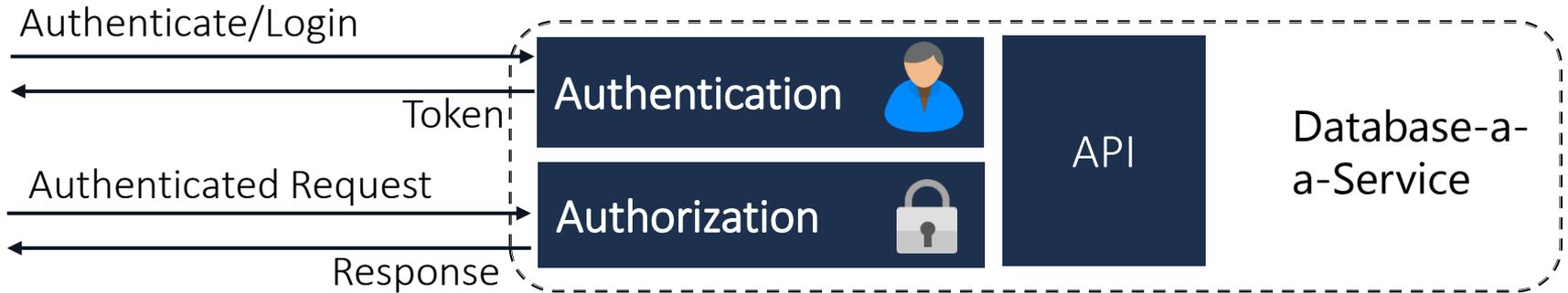
	App. indep.	Ressource Util.	Isolation	Maintenance, Provisioning
Private OS				
Private Process/DB				
Private Schema				
Shared Schema				



# Authentication & Authorization

## Checking Permissions and Identity

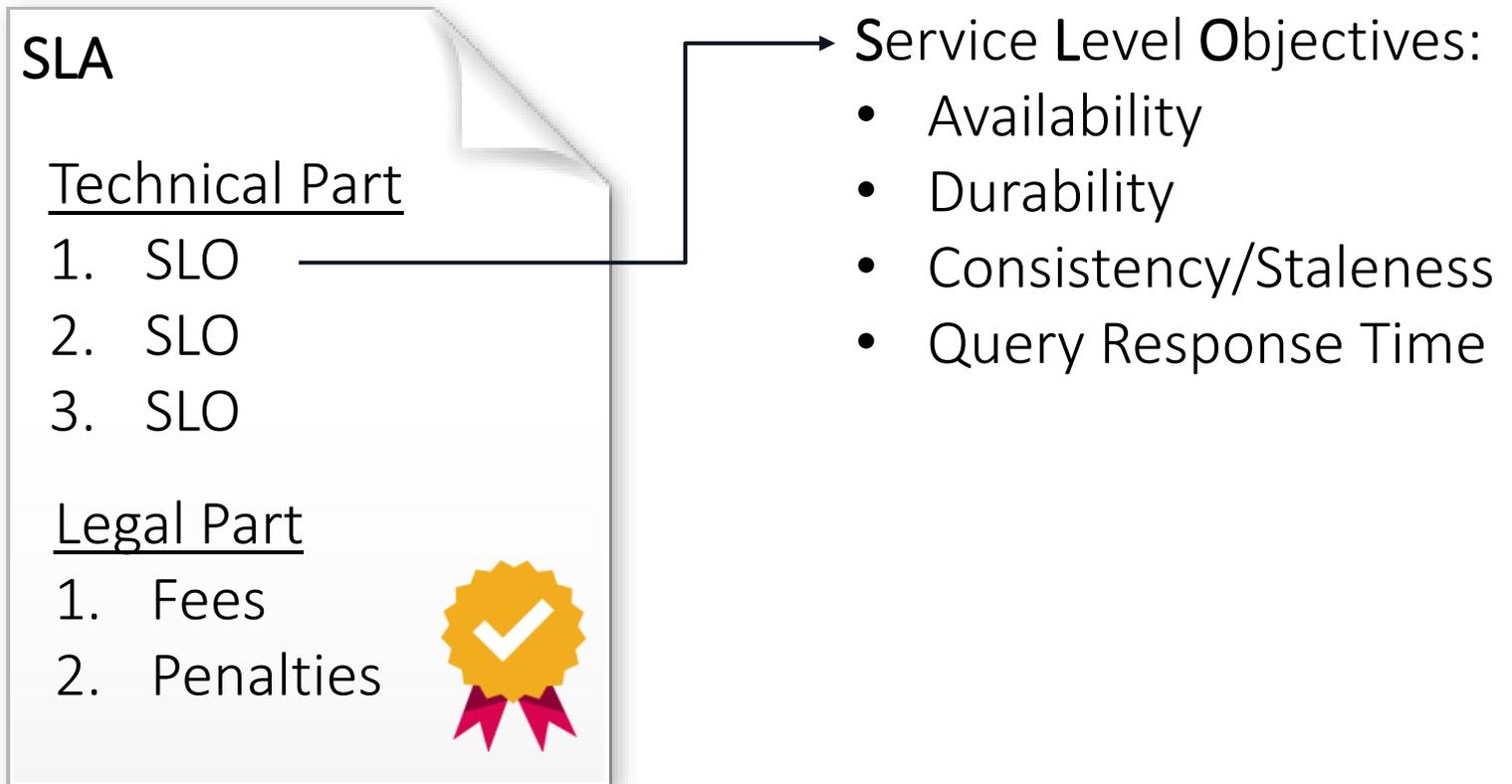
Internal Schemes	External Identity Provider	Federated Identity (Single Sign On)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML



User-based Access Control	Role-based Access Control	Policies
e.g. Amazon S3 ACLs	e.g. Amazon IAM	e.g. XACML

# Service Level Agreements (SLAs)

Specification of Application/Tenant Requirements



# Service Level Agreements

Expressing application requirements

## Functional Service Level Objectives

- Guarantee a „feature“
- Determined by database system
- *Examples:* transactions, join



## Non-Functional Service Level Objectives

- Guarantee a certain *quality of service* (QoS)
- Determined by database system and service provider
- *Examples:*
  - **Continuous:** response time (latency), throughput
  - **Binary:** Elasticity, Read-your-writes

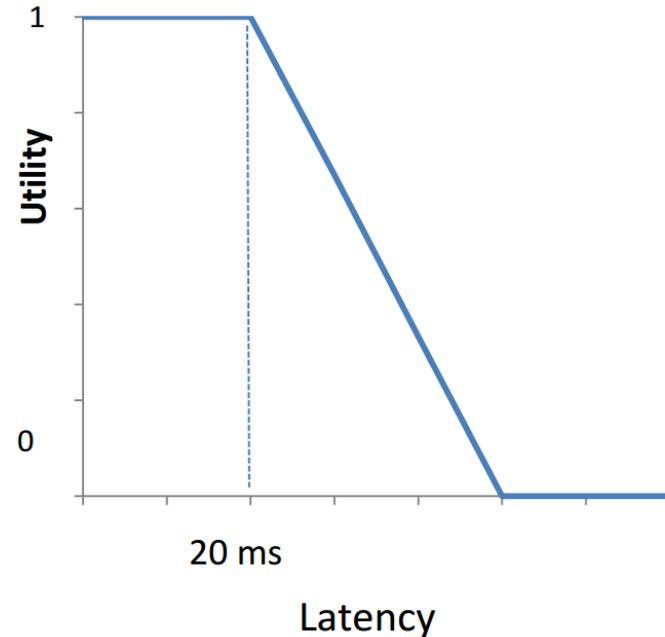
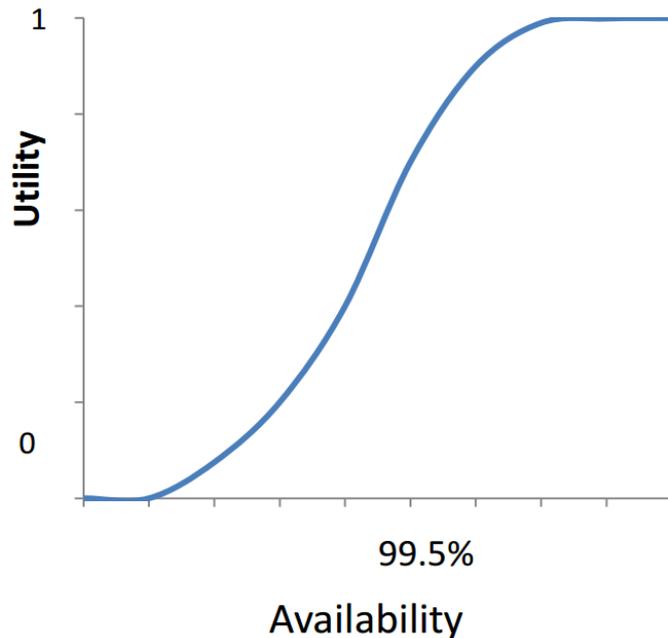


# Service Level Objects

Making SLOs measurable through utilities

Utility expresses „value“ of a continuous non-functional requirement:

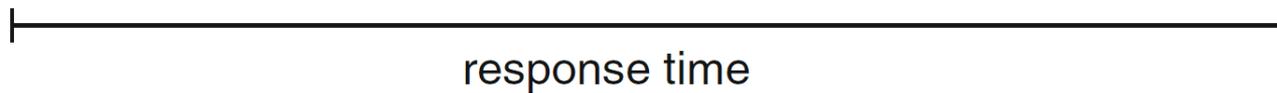
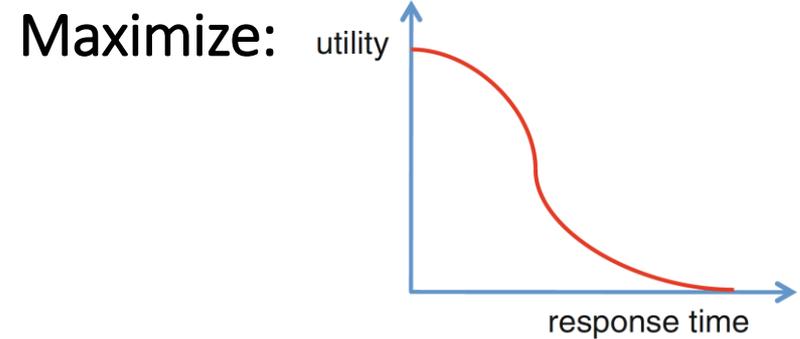
$$f_{utility}(metric) \rightarrow [0,1]$$



# Workload Management

## Guaranteeing SLAs

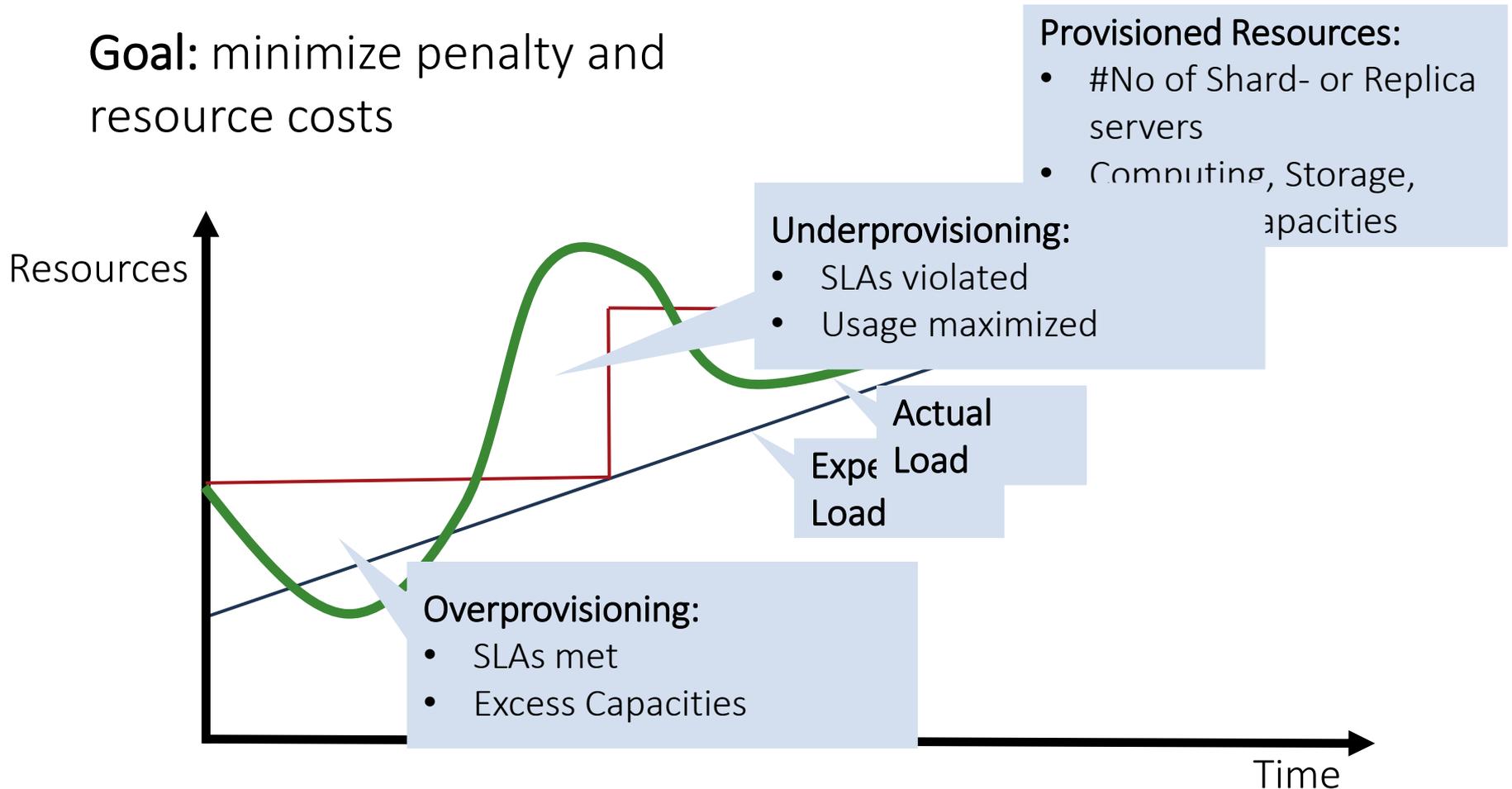
Typical approach:



# Resource & Capacity Planning

From a DBaaS provider's perspective

**Goal:** minimize penalty and resource costs



# SLAs in the wild

Most DBaaS systems offer no SLAs, or only a simple uptime guarantee

	Model	CAP	SLAs
<b>SimpleDB</b>	Table-Store (NoSQL Service)	CP	
<b>Dynamo-DB</b>	Table-Store (NoSQL Service)	CP	
<b>Azure Tables</b>	Table-Store (NoSQL Service)	CP	99.9% uptime
<b>AE/Cloud DataStore</b>	Entity-Group Store (NoSQL Service)	CP	
<b>S3, Az. Blob, GCS</b>	Object-Store (NoSQL Service)	AP	99.9% uptime (S3)

# Open Research Questions

## in Cloud Data Management

- ▶ **Service-Level Agreements**
  - How can SLAs be guaranteed in a virtualized, multi-tenant cloud environment?
- ▶ **Consistency**
  - Which consistency guarantees can be provided in a geo-replicated system without sacrificing availability?
- ▶ **Performance & Latency**
  - How can a DBaaS deliver low latency in face of distributed storage and application tiers?
- ▶ **Transactions**
  - Can ACID transactions be aligned with NoSQL and scalability?

Home

CFP

Submission

Contact

# 3<sup>rd</sup> Workshop on Scalable Cloud Data Management

Co-located with the IEEE BigData Conference. Santa Clara, CA, October 29th 2015.

Call

Submit Paper

May 4, 2015

SCDM 2015 announced

[www.scdm2015.com](http://www.scdm2015.com)

The third Scalable Cloud Data Management Workshop (SCDM 2015) will again be held in conjunction with the IEEE BigData 2015 - this year in Santa Clara.

Location : **Santa Clara**

Submission Deadline: **August 30**

# Outline



Motivation



**ORESTES: a Cloud-  
Database Middleware**



Solving Latency and  
Polyglot Storage



Wrap-up

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

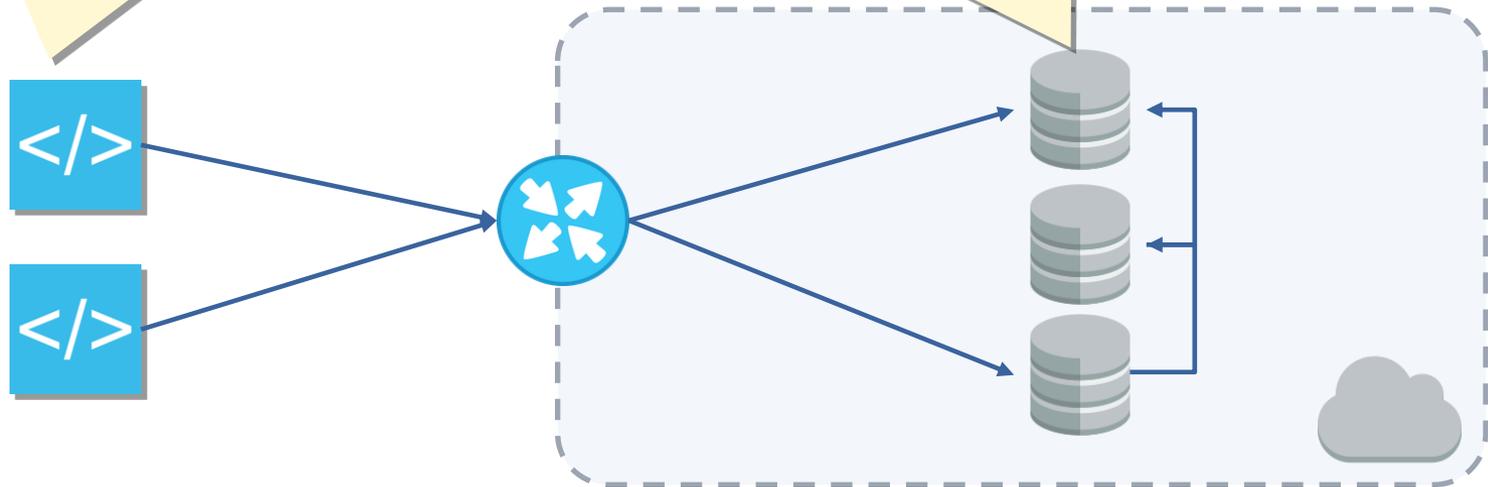
# Latency & Polyglot Storage

## Two central problems

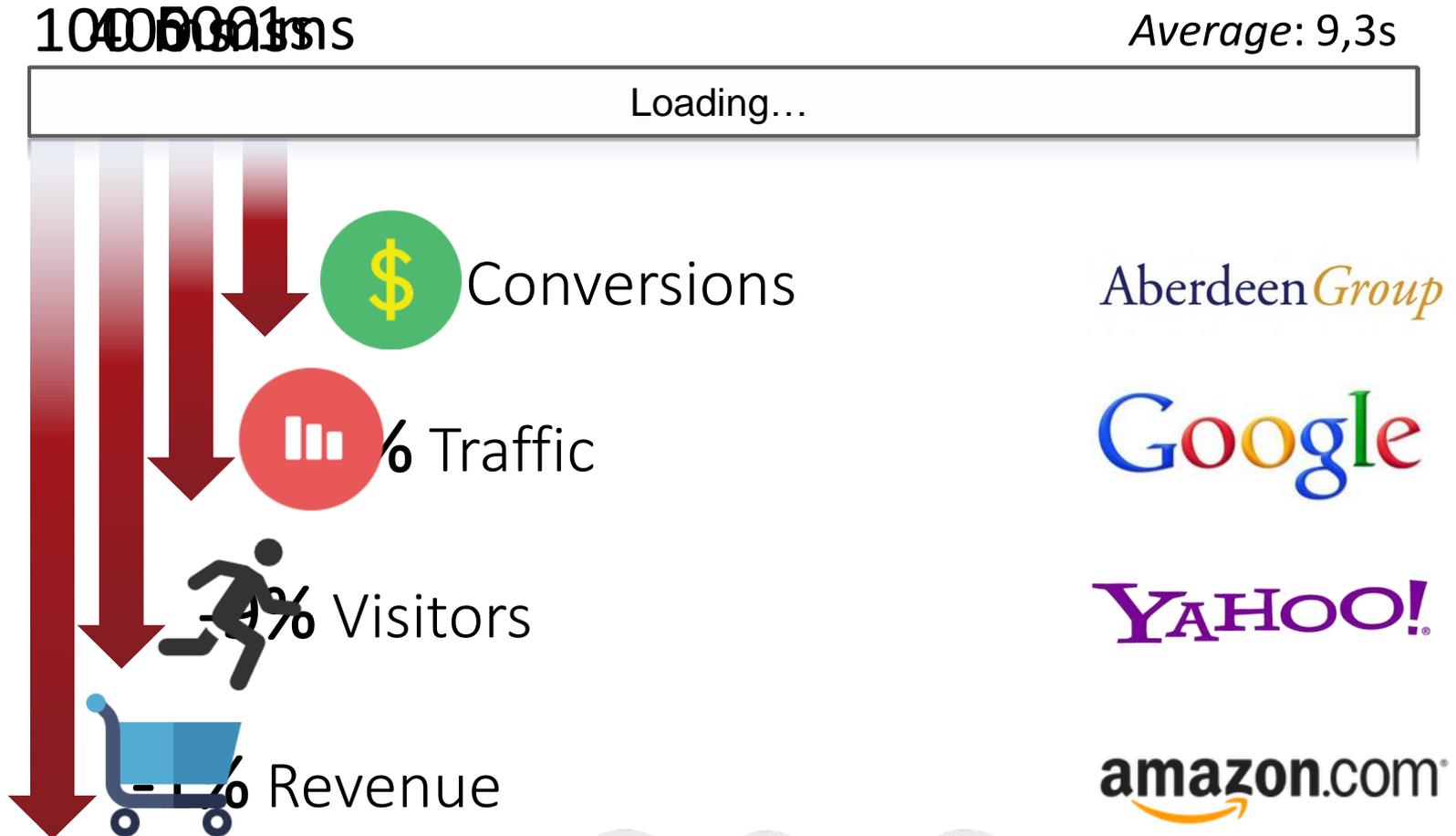
- ▶ Goal of ORESTES: Solve both problems through a scalable **cloud-database middleware**

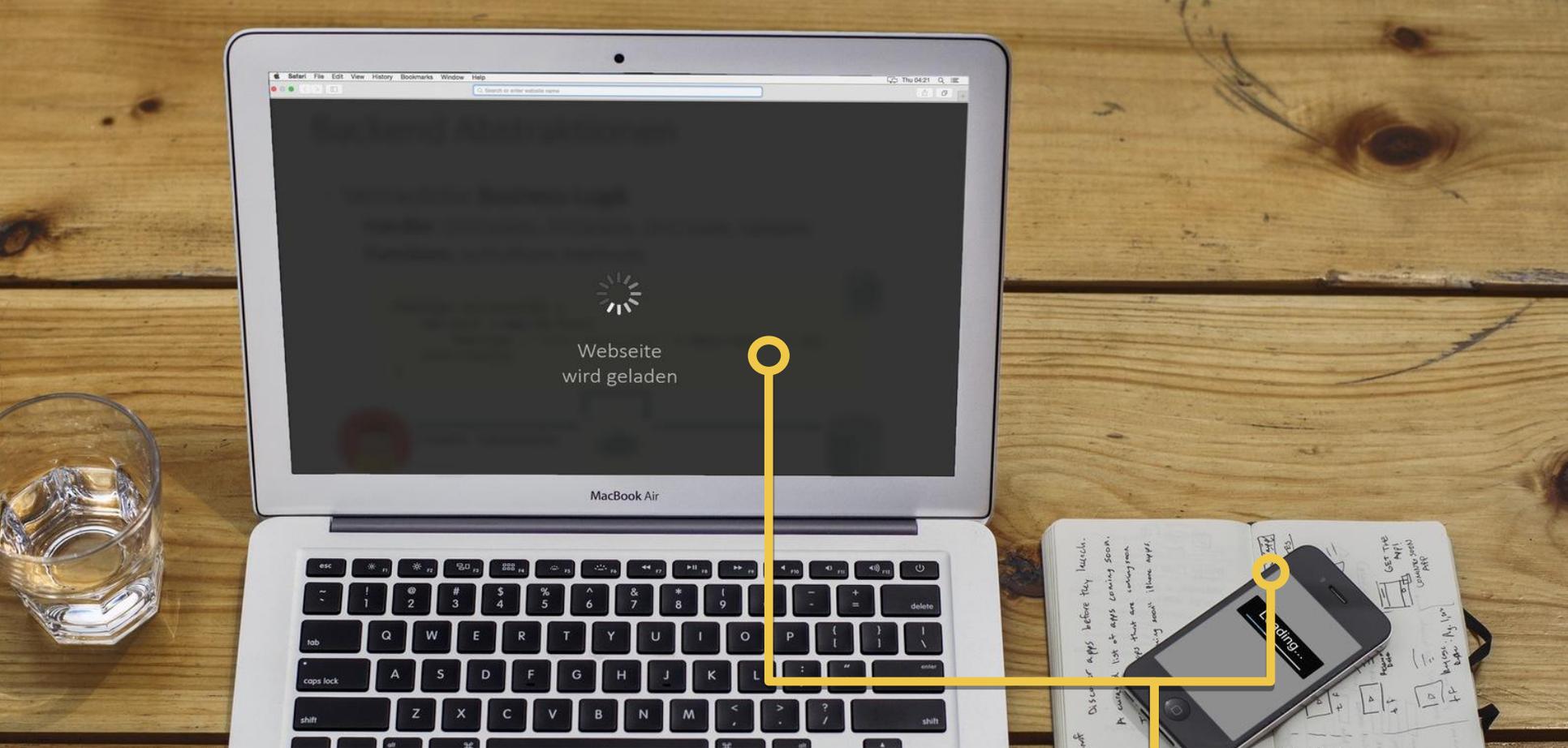
If the application is *geographically distributed*, how can we guarantee **fast database access**?

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



# Problem I: Latency





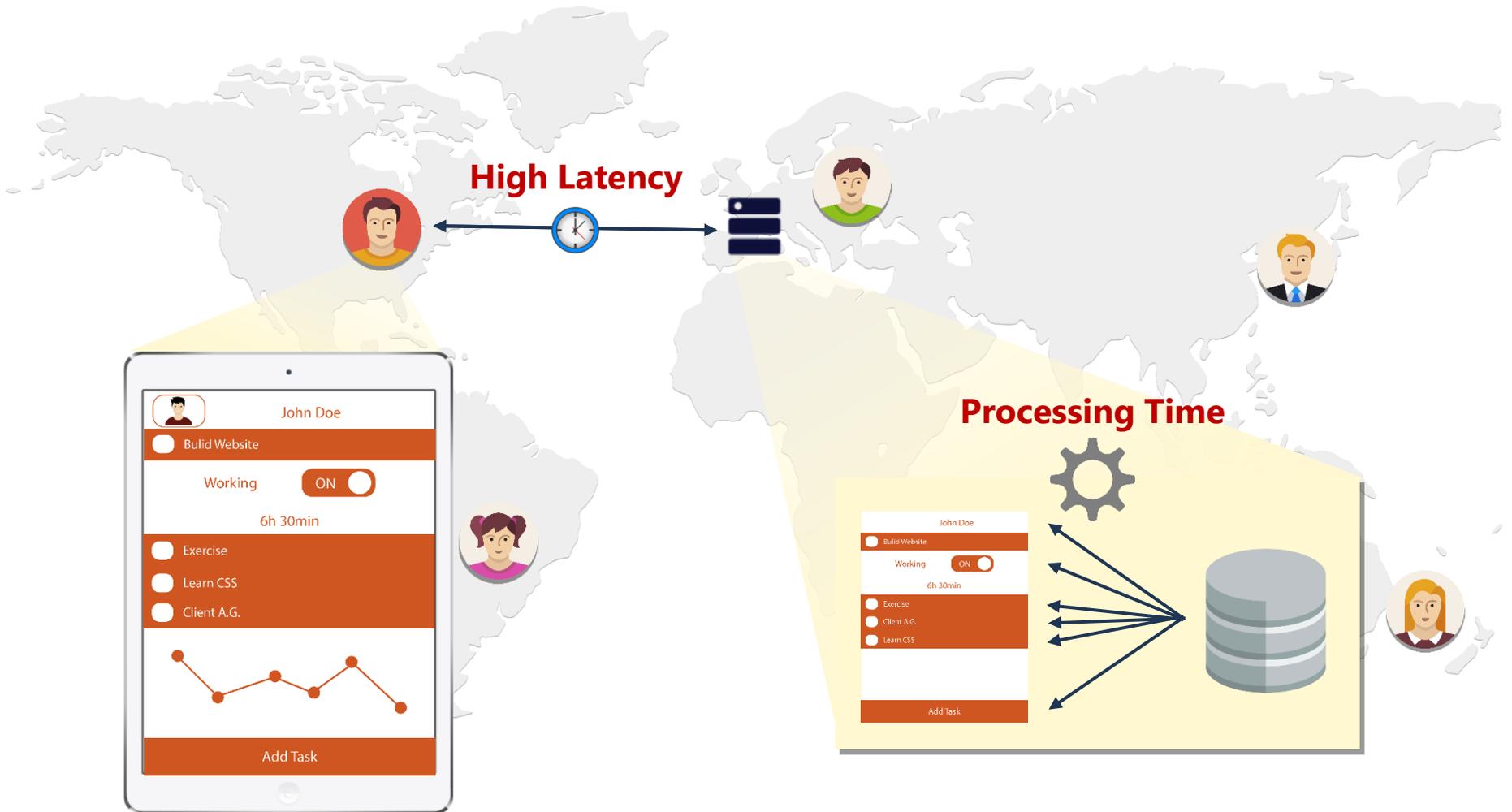
If perceived speed is such an import factor



...what causes slow page load times?

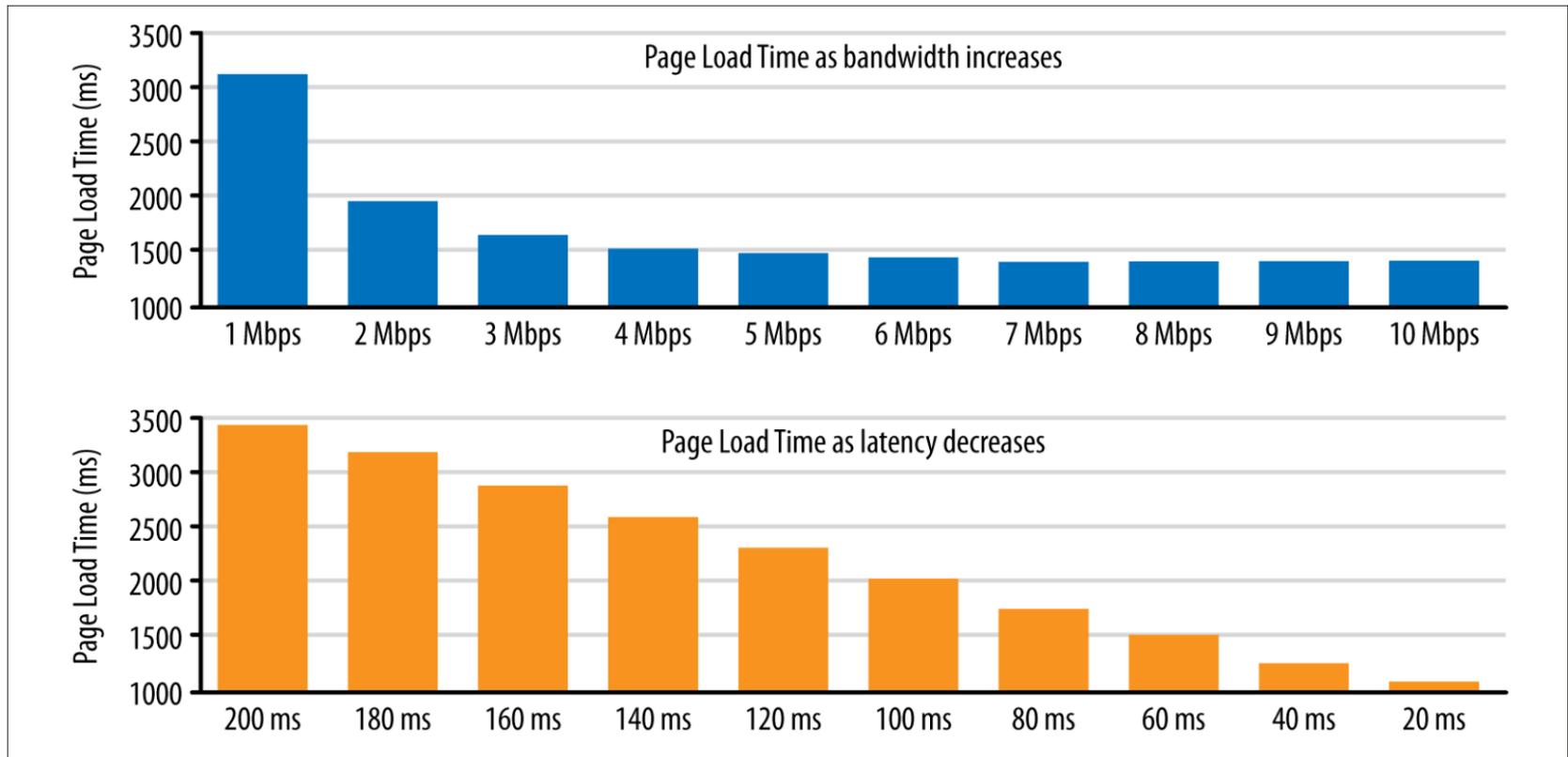
# State of the art

Two bottlenecks: latency und processing



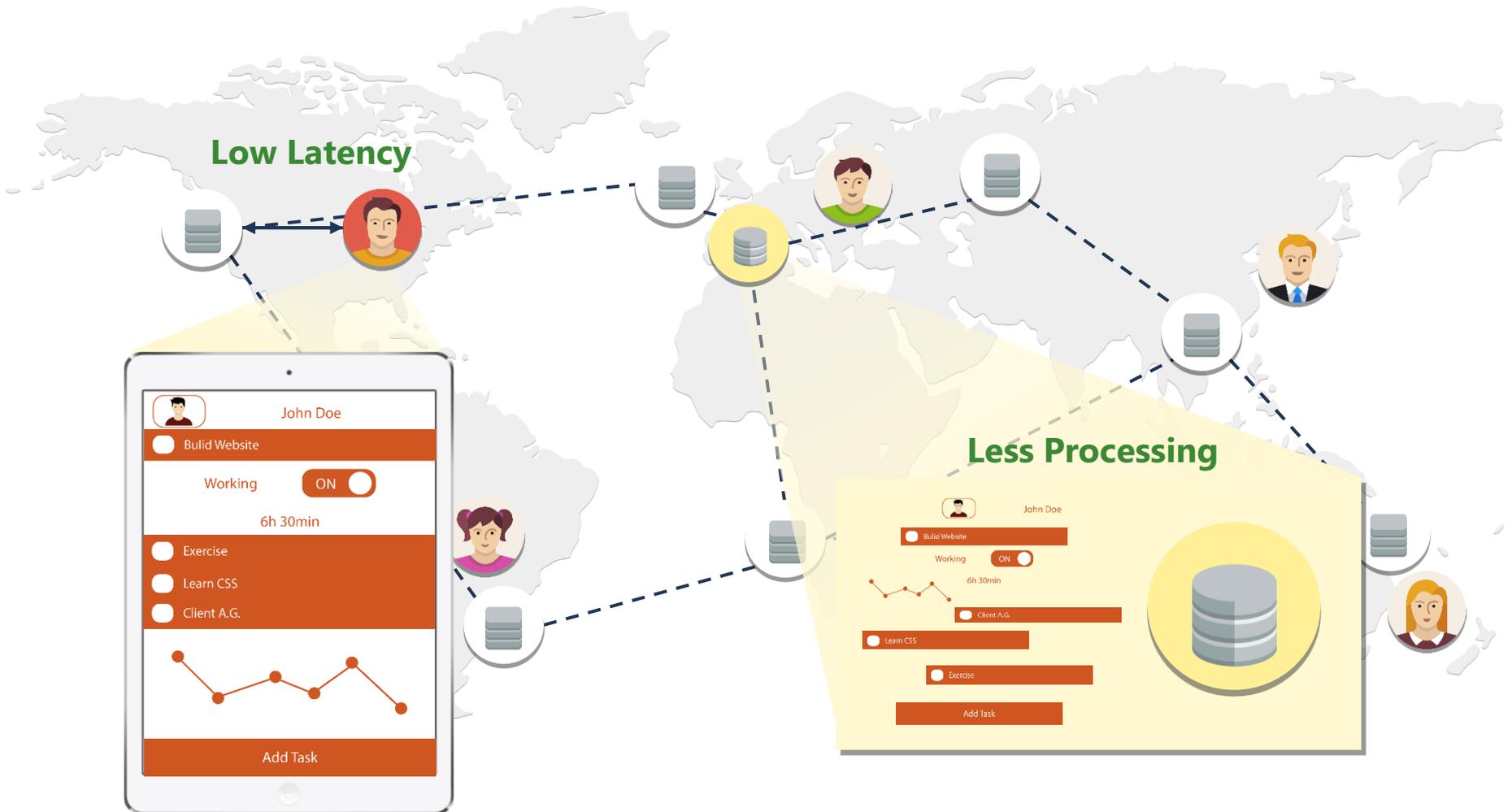
# Network Latency

The underlying problem of high page load times



# 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

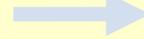


**Research Question:**

Can we automate the

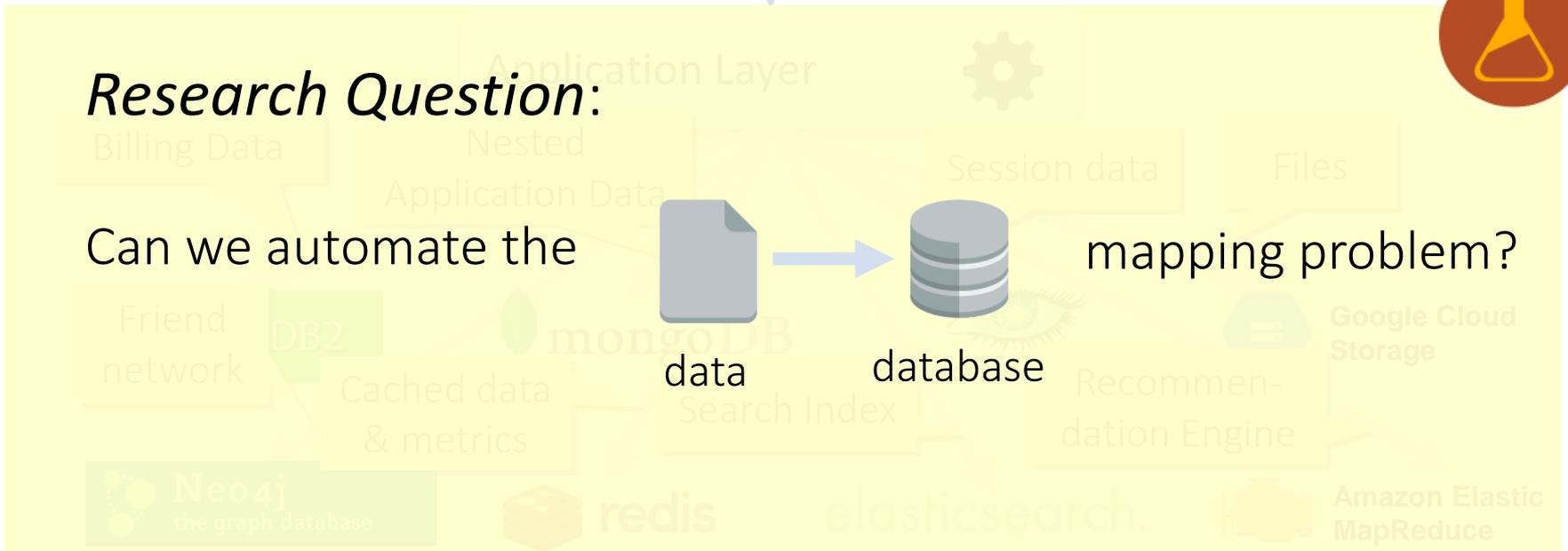


data



database

mapping problem?

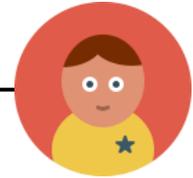


# Vision

Schemas can be annotated with requirements

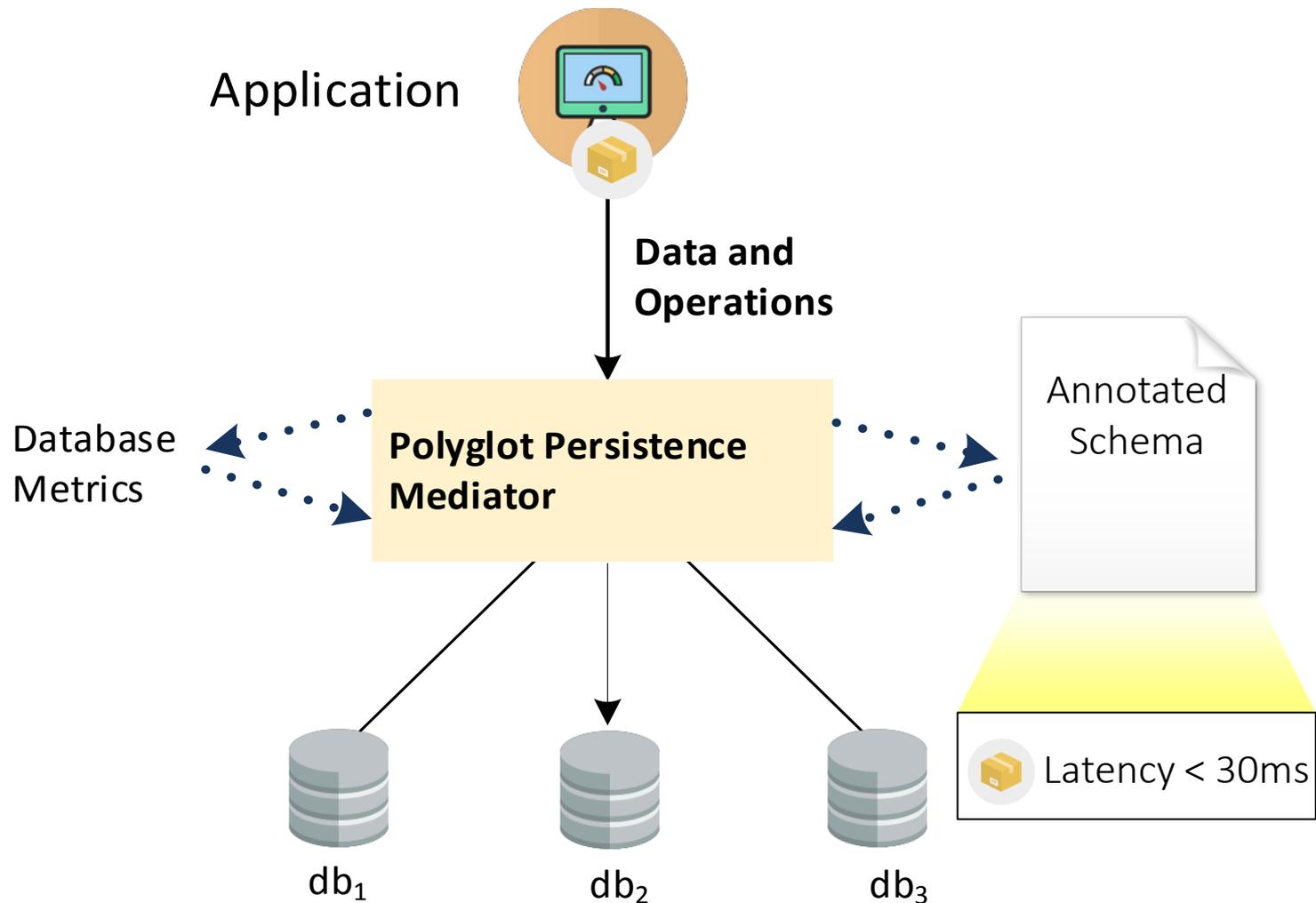


- Write Throughput > 10,000 RPS
- Read Availability > 99.9999%
- Scans = **true**
- Full-Text-Search = **true**
- Monotonic Read = **true**



# Vision

The Polyglot Persistence Mediator chooses the database



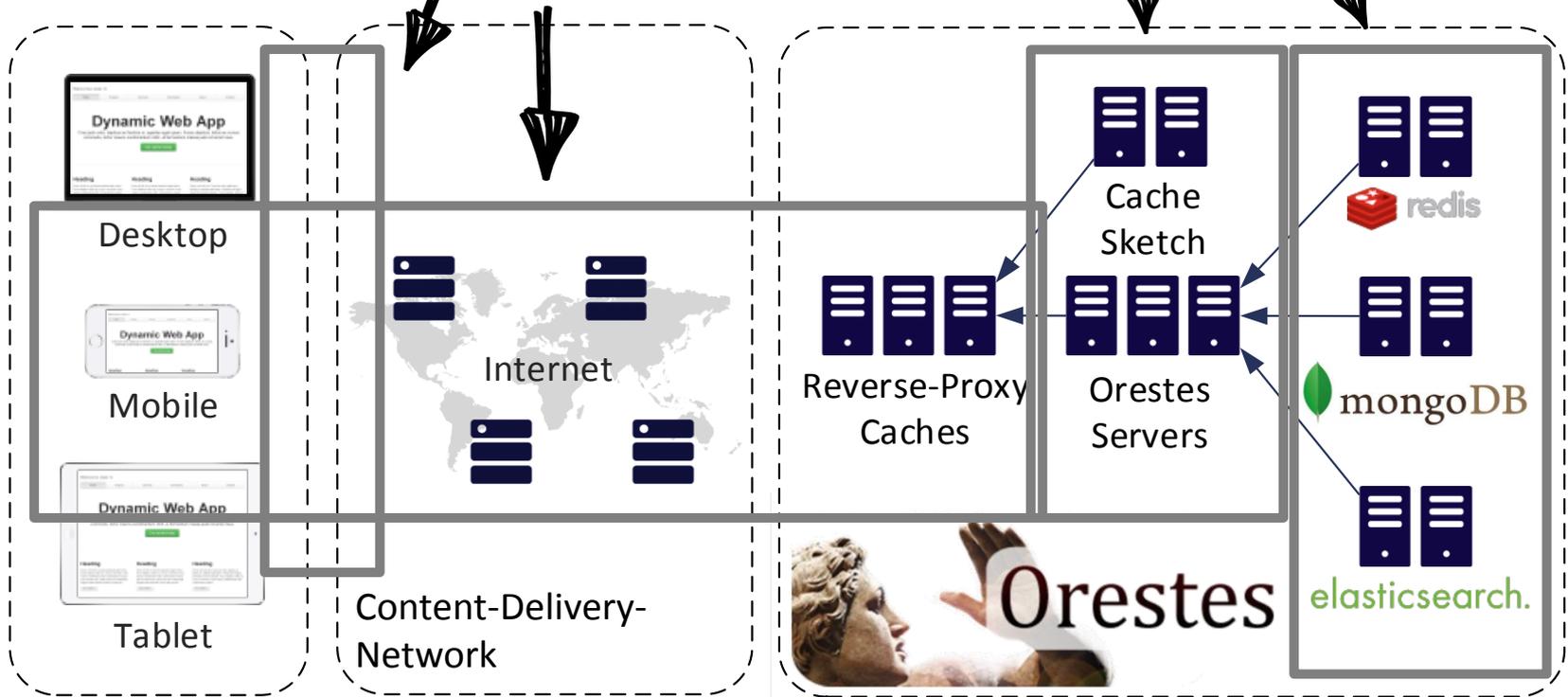
# The Big Picture

## Implementation in ORESTES

- ▶ Polyglot Storage and Low Latency are the central goals of ORESTES
- Unified REST API  
Standard HTTP Caching

## Database-as-a-Service

- Middleware:  
Caching, Traffic, Regions, Storage, Authorization, Multi-Tenancy



# Outline



Motivation



ORESTES: a Cloud-  
Database Middleware



**Solving Latency and  
Polyglot Storage**



Wrap-up

- Cache Sketch Approach
  - Caching Arbitrary Data
  - Predicting TTLs
- Polyglot Persistence Mediator
  - SLA-Approach
  - Database Selection

# Web Caching Concepts

## Invalidation- and expiration-based caches



### Expiration-based Caches:

- ▶ An object  $x$  is considered fresh for  $TTL_x$  seconds
- ▶ The server assigns TTLs for each object

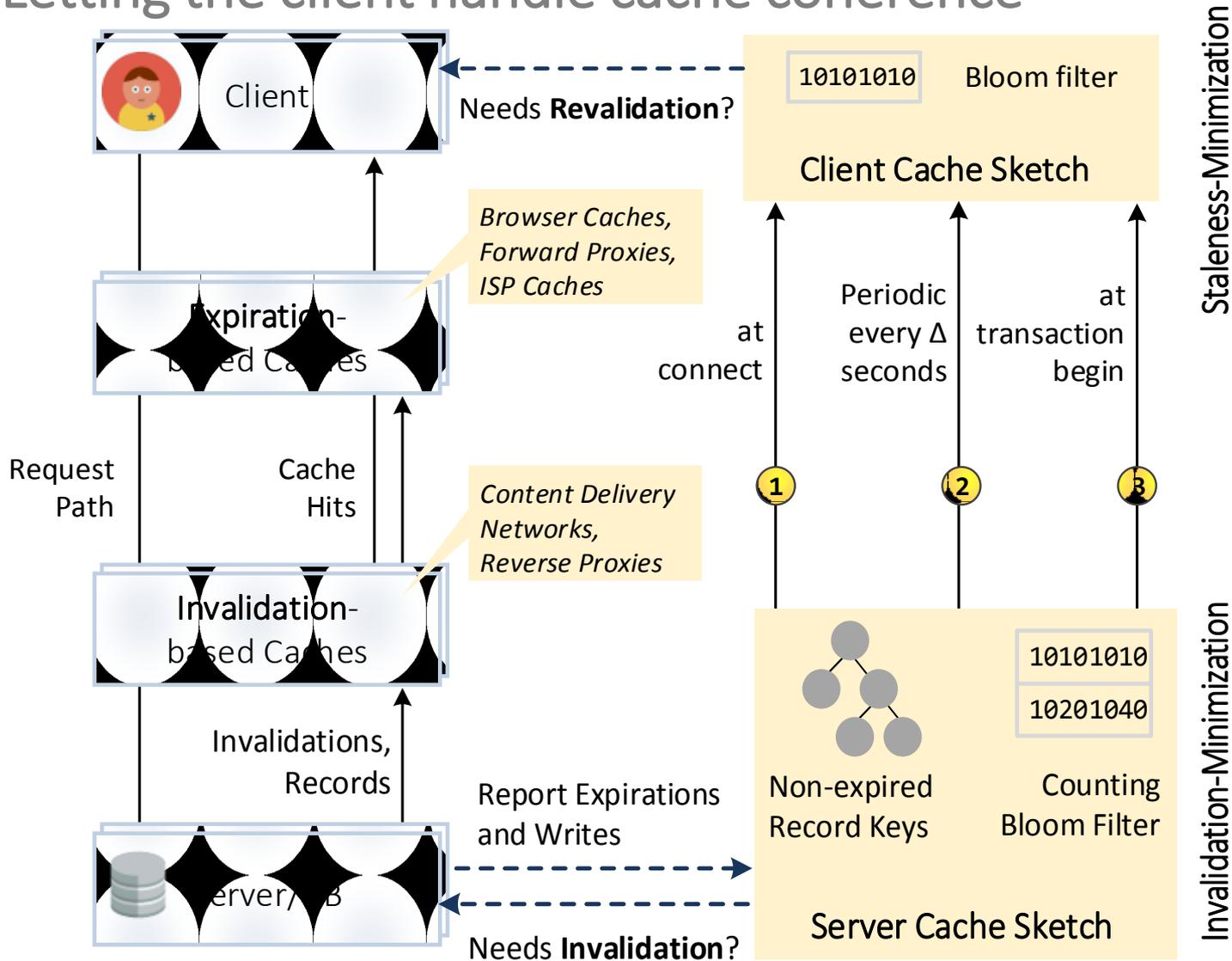
### Invalidation-based Caches:

- ▶ Expose object eviction operation to the server



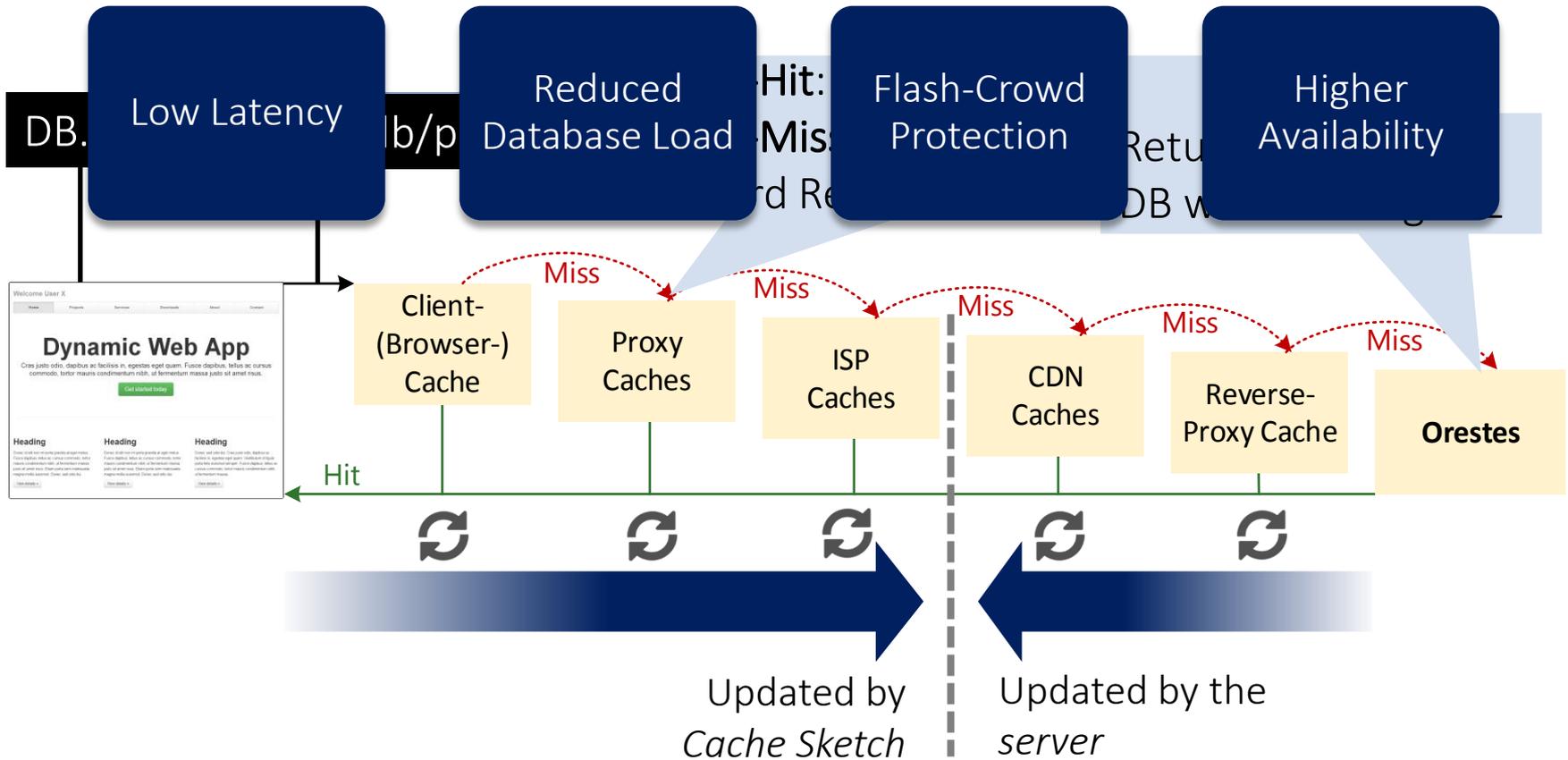
# The Cache Sketch approach

Letting the client handle cache coherence



# The End-to-End Path of Requests

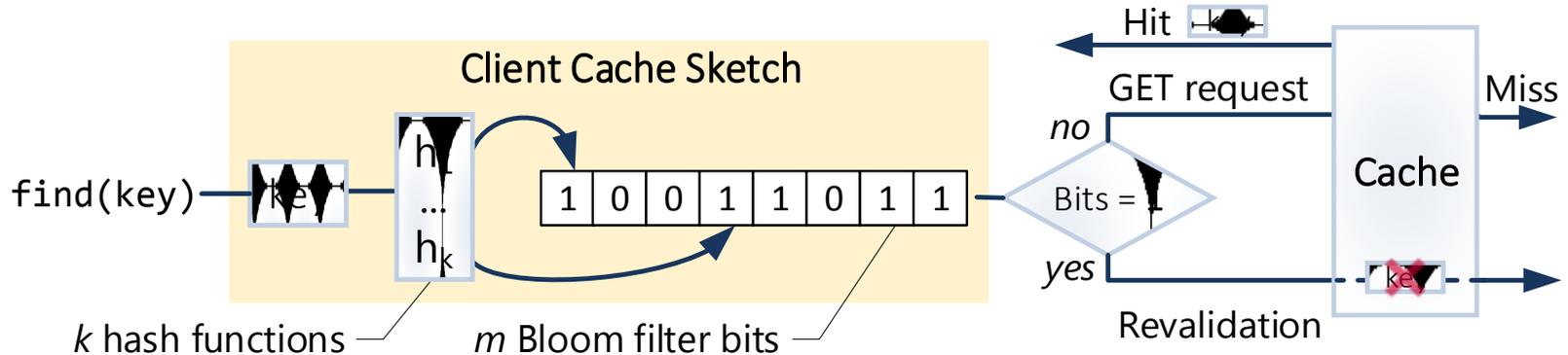
## The Caching Hierarchy



# The Client Cache Sketch

- ▶ Let  $c_t$  be the client Cache Sketch generated at time  $t$ , containing the key  $\mathbf{key}_x$  of every record  $x$  that was written before it expired in all caches, i.e. every  $x$  for which holds:

$$\exists r(x, t_r, TTL), w(x, t_w) : t_r + TTL > t > t_w > t_r$$



# 1 Slow initial page loads

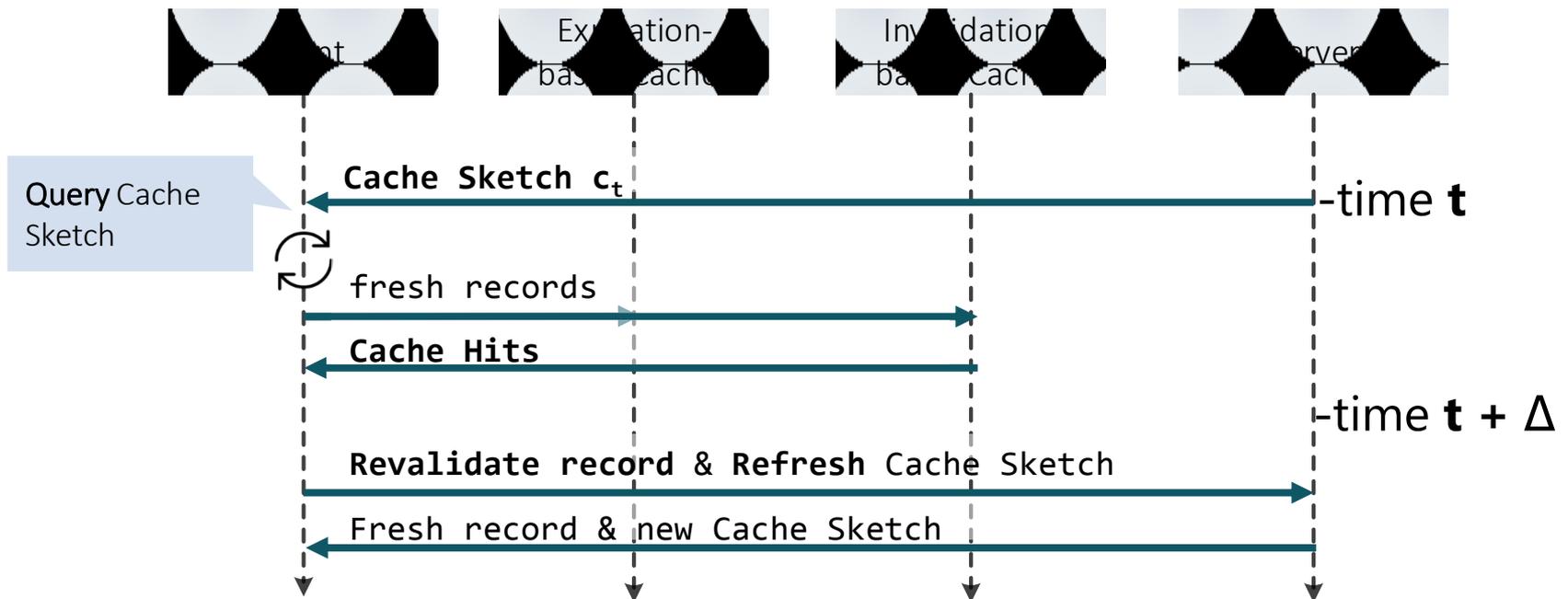
## ▶ Solution: **Cached Initialization**

- Clients load the Cache Sketch at connection
- Every non-stale cached record can be reused without degraded consistency

## 2 Slow CRUD performance

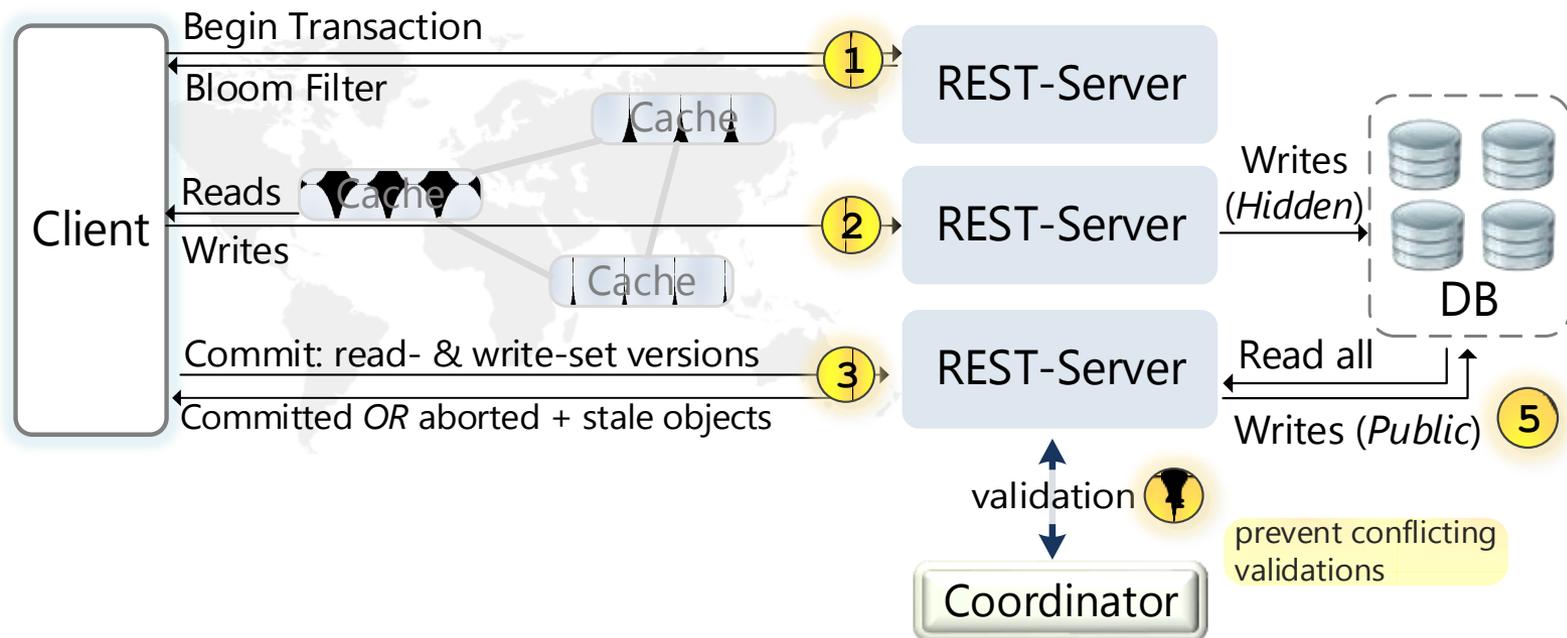
### ▶ Solution: $\Delta$ -Bounded Staleness

- Clients refresh the Cache Sketch so its age never exceeds  $\Delta$   
→ *Consistency guarantee:  $\Delta$ -atomicity*



# 3 High Abort Rates in OCC

- ▶ Solution: Conflict-Avoidant Optimistic Transactions
  - Cache Sketch fetched with transaction begin
  - Cached reads → Shorter transaction duration → less aborts



# TTL Estimation

Determining the cache expiration

- ▶ **Problem:** if  $TTL \gg$  time to next write, then it is contained in Cache Sketch unnecessarily long
- ▶ **TTL Estimator:** finds „best“ TTL
- ▶ **Trade-Off:**

Shorter TTLs



- less invalidations
- less stale reads

Longer TTLs



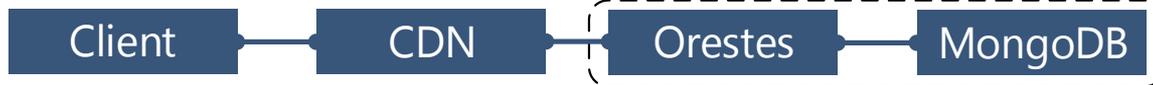
- Higher cache-hit rates
- more invalidations

# Performance

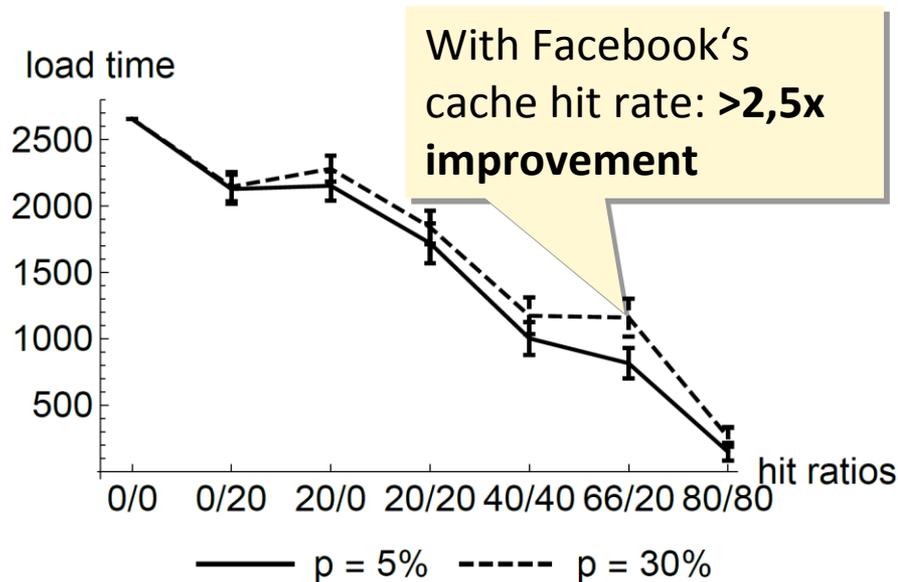
Northern California

Ireland

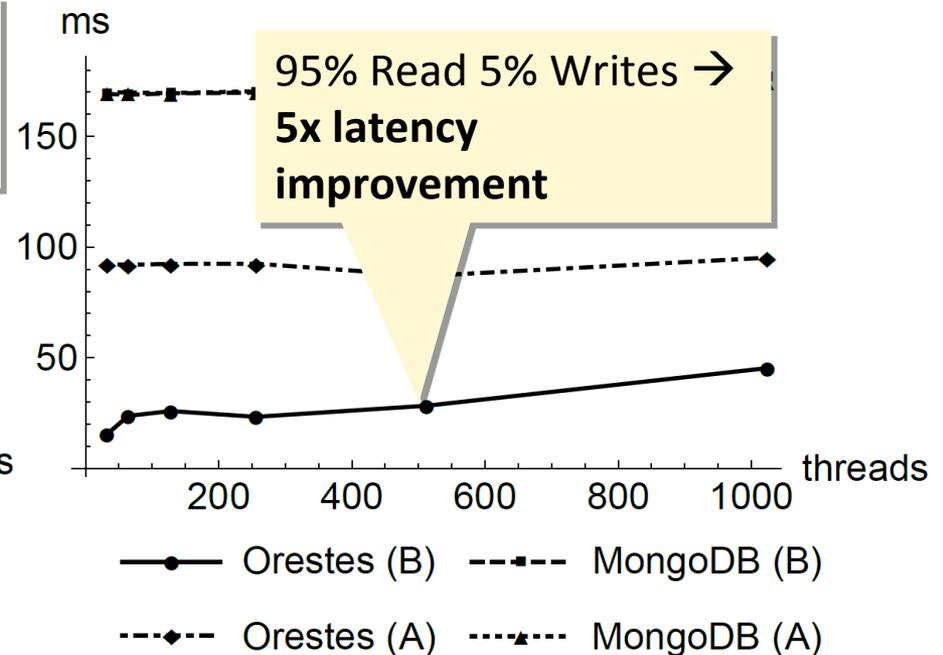
Setup:



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



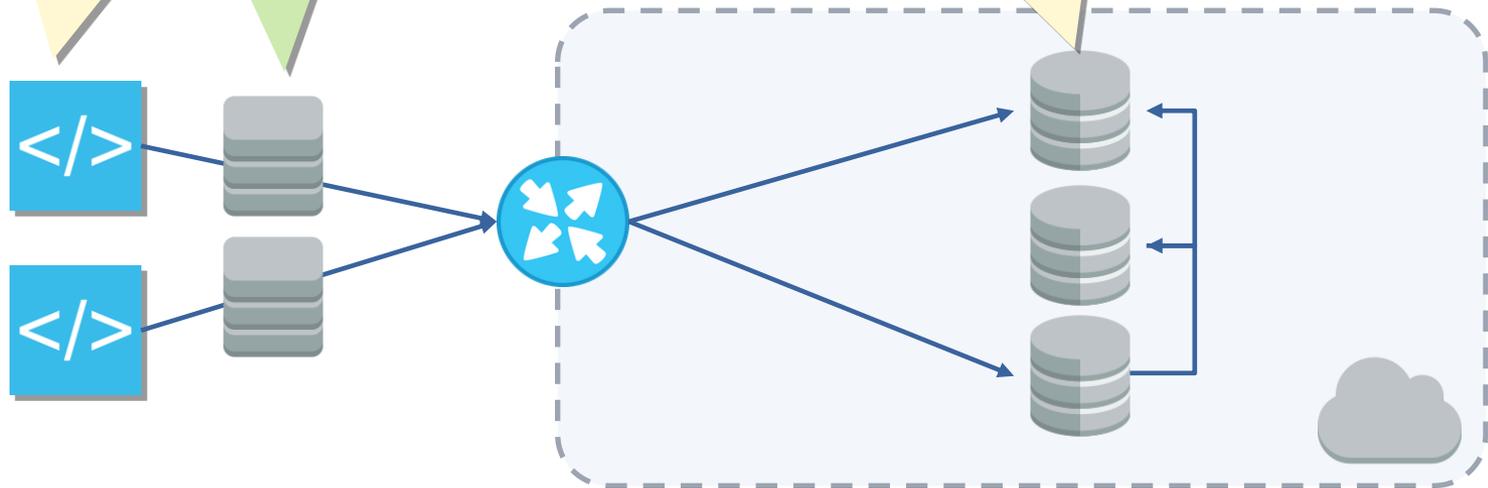
Average Latency for YCSB Workloads A and B (real):



# Low Latency

If the application is *distributed*, can we use **fast databases** for **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



Tenant specifies requirements as Service-Level-Agreements

### 2. Resolution



Find or provision a suitable combination of databases

### 3. Mediation

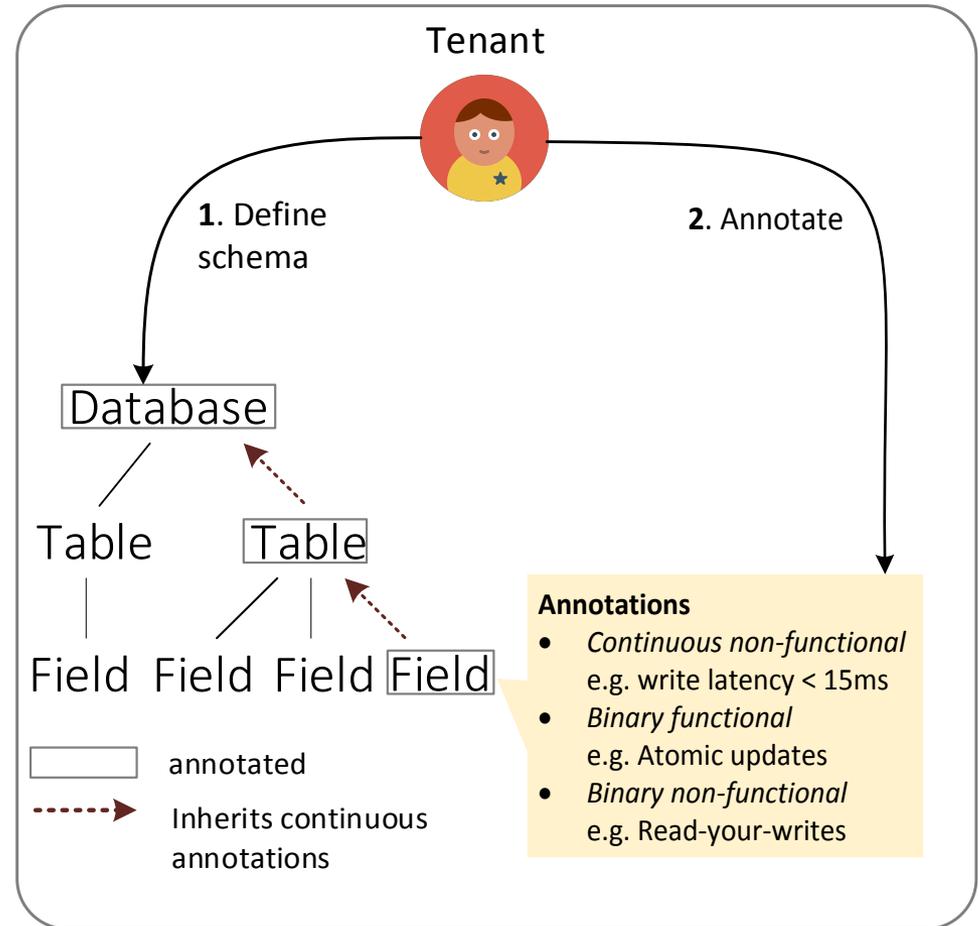


Mediate data and database operations

# Step I - Requirements

## Expressing the application's needs

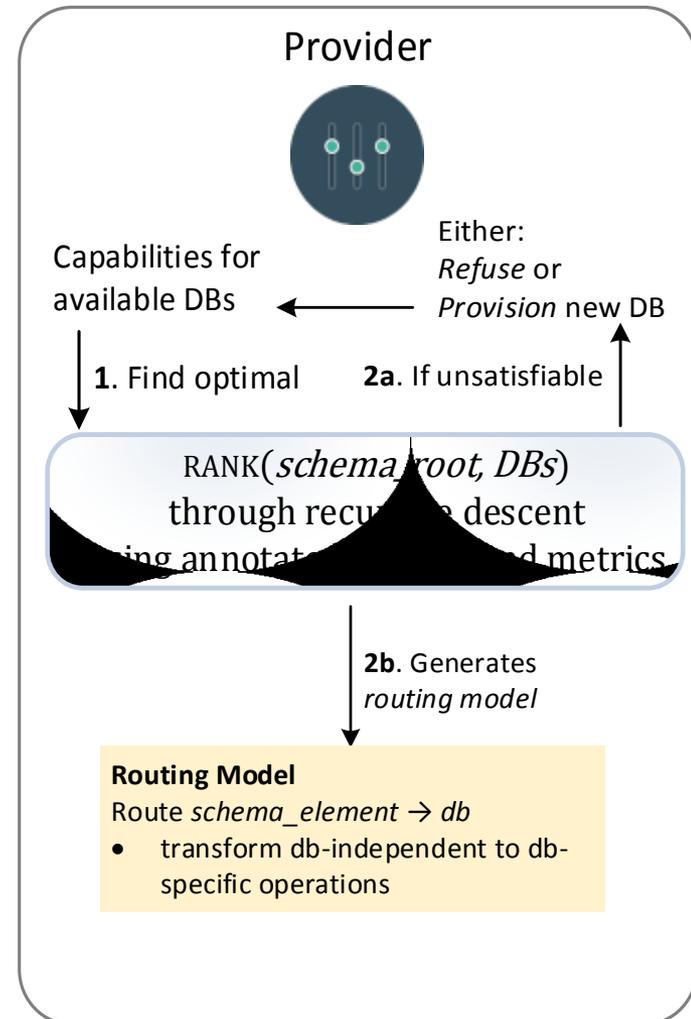
Annotation	Type	Annotated at
Read Availability	Continuous	*
Write Availability	Continuous	*
Read Latency	Continuous	*
Write Latency	Continuous	*
Write Throughput	Continuous	*
Data Vol. Scalability	Non-Functional	Field/Class/DB
Write Scalability	Non-Functional	Field/Class/DB
Read Scalability	Non-Functional	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



# Step II - Resolution

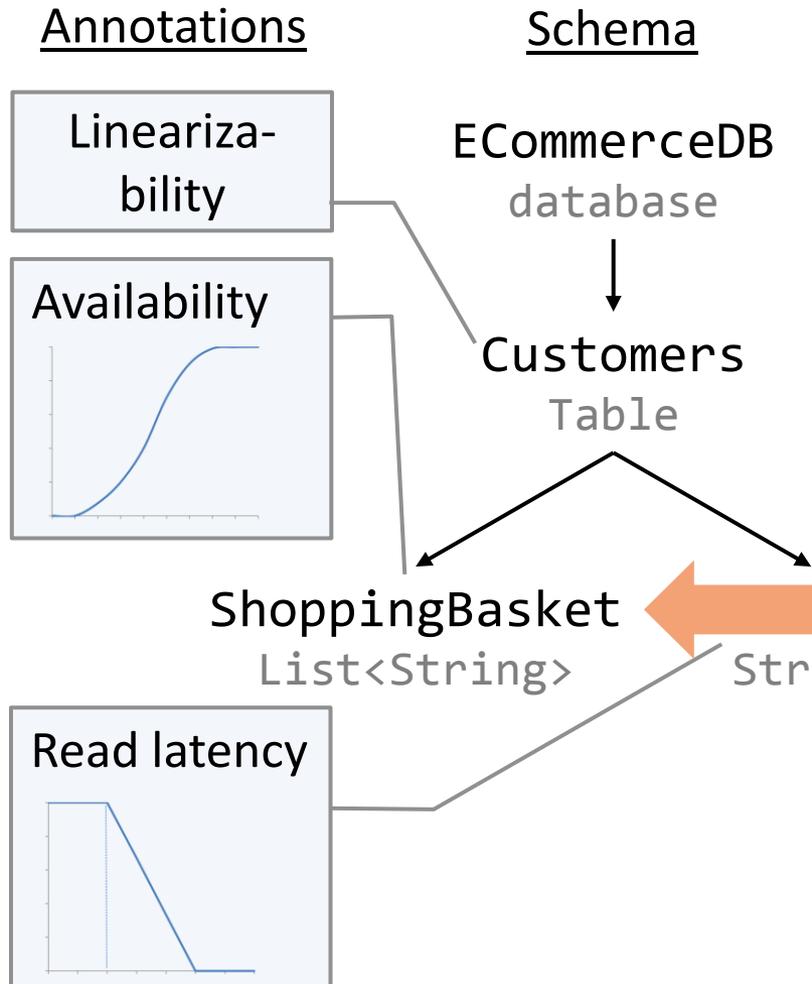
## Finding the best database

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



# Step II - Resolution

## Ranking algorithm by example



DBs = { MongoDB, Riak, Cassandra, CouchDB, Redis, MySQL, S3, Hbase }

DBs = { MongoDB, Riak, Cassandra, CouchDB, Redis, MySQL, S3, Hbase }

Database	Availability	Latency
MongoDB	95% → 0.05	1ms → 1
Redis	94% → 0.04	40ms → 0.2
MySQL	99.9% → 0.9	50ms → 0.1
HBase		

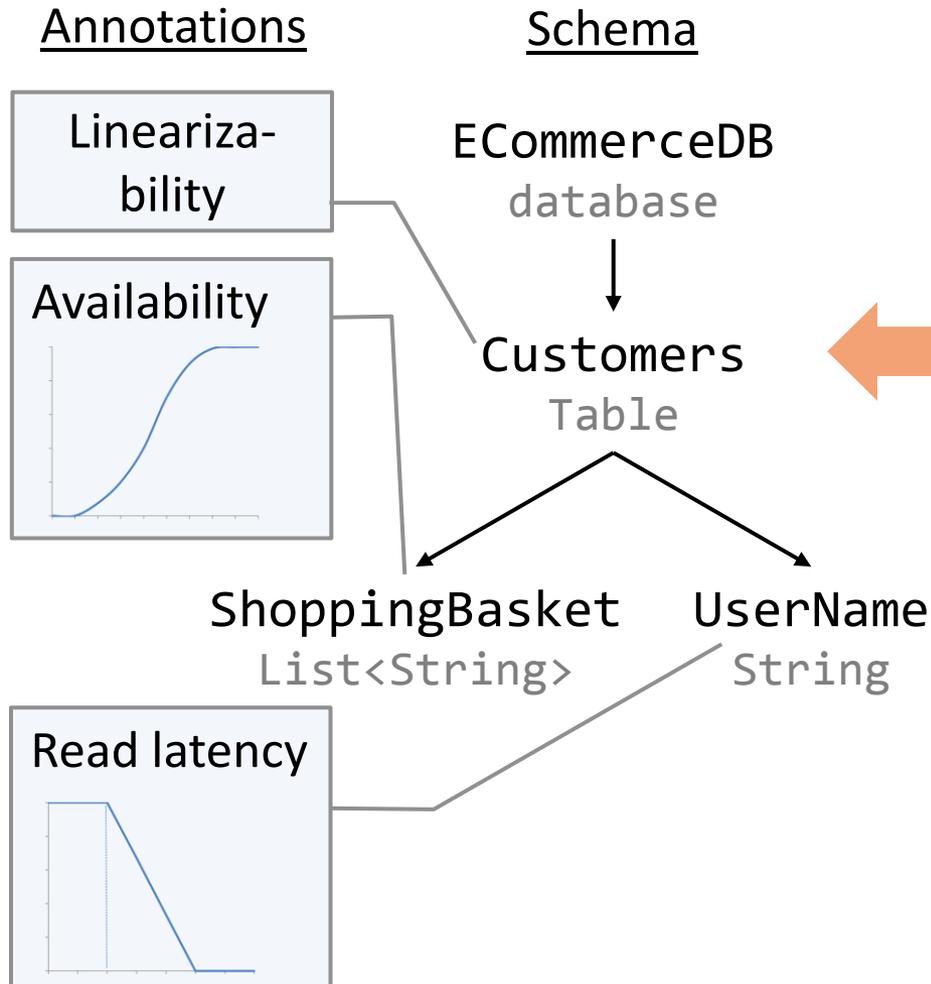
2. Recursive descent

Continuous requirement →  
 $\forall$  databases calculate

$$db \rightarrow f_{utility}(db, latency, availability)$$

# Step II - Resolution

## Ranking algorithm by example



DB	Score
MongoDB	0.9
Redis	0.525
MySQL	0.12
HBase	0.5

Binary requirement →

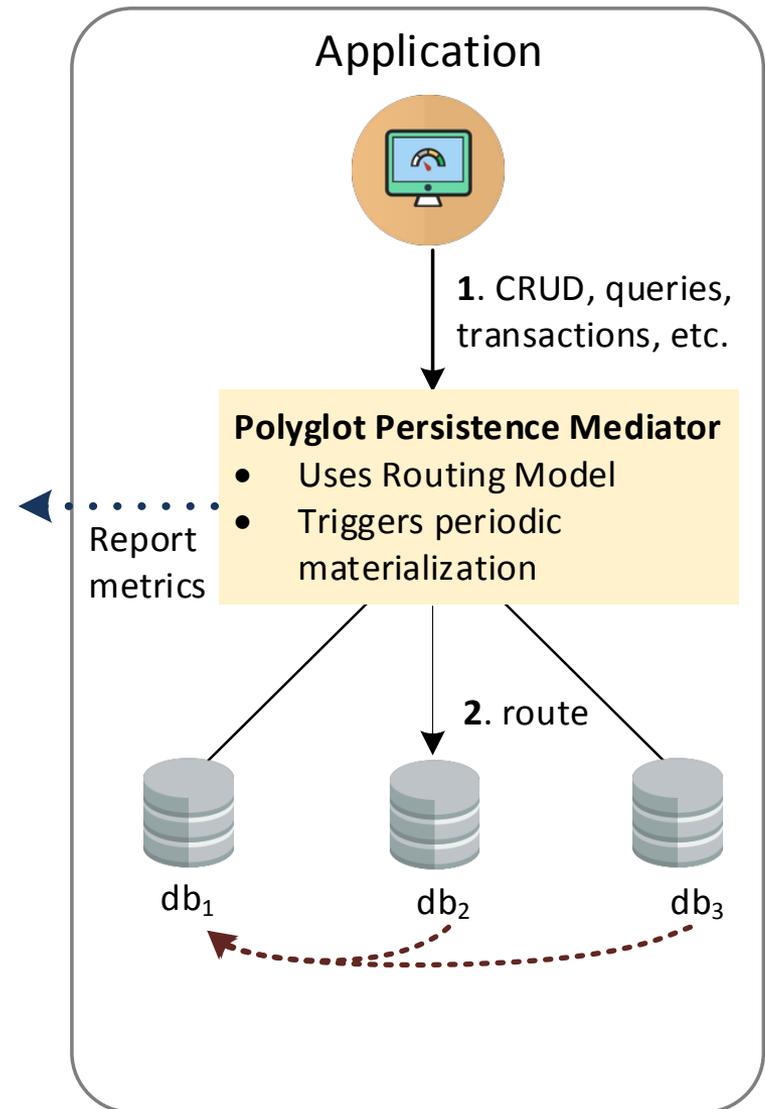
1. Exclude DBs that do not support it
2. Recursive descent
3. Pick DB with best total score and add it to routing model

Routing Model:  
Customers → MongoDB

# 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 counts, article-queries

The image shows a screenshot of a Hacker News article. The article title is "Announcing MongoDB 3.0 (mongodb.com)" and it has a read count of "read by 1.344.222". Two arrows point to specific parts of the article: one labeled "Article" points to the article title, and one labeled "Counter" points to the read count.

**Article** → **Y Hacker News** new | threads | comments | show | a

Announcing MongoDB 3.0 (mongodb.com)

196 points by meghan 142 days ago | 144 comments | in pocket speiche

read by 1.344.222

**Counter** →

# Evaluation: News Article

Prototype built on ORESTES

**Scenario:** news articles with impression counts

**Objectives:** low-latency top-k queries, high-throughput counts, 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 counts, article-queries



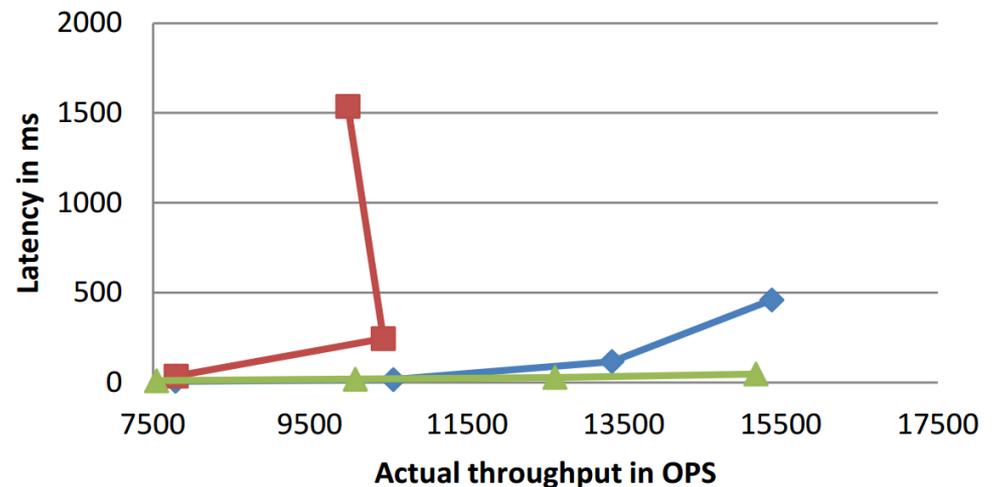
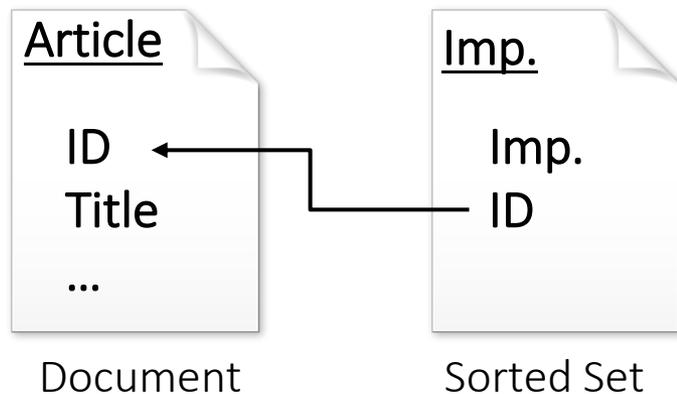
No powerful queries

# Evaluation: News Article

Prototype built on ORESTES

**Scenario:** news articles with impression counts

**Objectives:** low-latency top-k queries, high-throughput counts, article-queries



◆ Orestes with PPM    ■ Orestes without PPM    ▲ Varnish

*Found Resolution*

# Outline



Motivation



ORESTES: a Cloud-  
Database Middleware



Solving Latency and  
Polyglot Storage



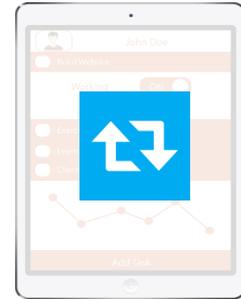
Wrap-up

- Current/Future Work
- Summary
- Putting ORESTES into practice

# Outlook: Real-Time

Combining Query Caching,  
Continuous Queries, Polyglot Queries

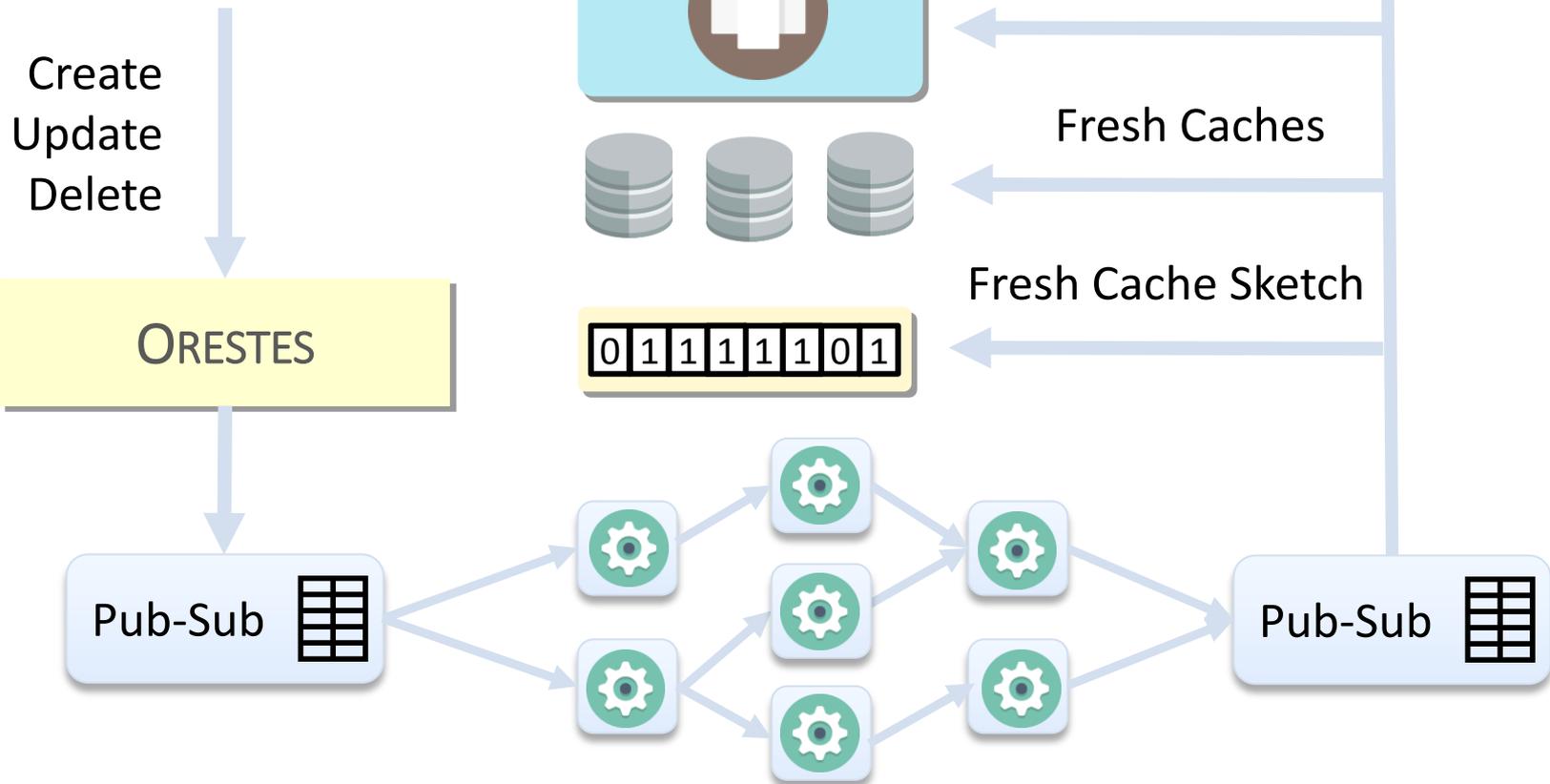
Continuous  
Queries  
(*Websockets*)



Polyglot Views

Fresh Caches

Fresh Cache Sketch



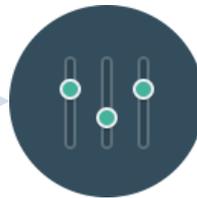
# Summary



- ▶ **Cache Sketch:** web caching for database services
  - Consistent ( $\Delta$ -atomic) *expiration-based* caching
  - *Invalidation-based* caching with minimal purges
  - **Bloom filter** of stale objects & **TTL Estimation**
- ▶ **Polyglot Persistence Mediator:**
  1. SLA-annotated **Schemas**
  2. **Score** DBs and choose best
  3. **Route** data and operations



Requirements



Resolution



Mediation

A photograph of a space shuttle launching, with a large plume of white smoke and fire at the base. The shuttle is white with a gold nose cone. The background is a blue sky with scattered white clouds. A semi-transparent horizontal band is overlaid across the middle of the image.

# BaQend

Build faster Apps **faster.**

# Page-Load Times

## What impact does the Cache Sketch have?



### Politik



11. November 2014 12:42 Uhr  
Deutsche Rentenversicherung  
**Renten könnten 2015 um zwei Prozent steigen**  
Die Deutsche Rentenversicherung geht von einem Anstieg über der Inflationsrate aus. Abschlagsfreie Rente ab 63 Jahren stößt auf großes Interesse.

### Wirtschaft



11. November 2014 07:15 Uhr  
HONORARBERATUNG  
**Guter Rat zur Geldanlage ist selten**  
Honorarberatung ist in Deutschland endlich gesetzlich geregelt. Doch gibt es kaum Honorarberater. Und gut qualifizierte noch viel weniger.

### Kultur



11. November 2014 10:14 Uhr  
NICOLAUS HARNONCOURT  
**Mozarts Triptychon**  
Nikolaus Harnoncourt ist der Detektiv unter den Dirigenten. Jetzt legt er Indizien vor, wie drei von Mozarts Sinfonien zu einem nie gehörten Oratorium verschmelzen.



11. November 2014 10:05 Uhr  
Europäischer Gerichtshof  
**Deutschland darf EU-Ausländern Hartz IV verweigern**  
Der Europäische Gerichtshof hat entschieden: Deutschland kann arbeitslose Zuwanderer aus der EU von Sozialleistungen ausschließen. Das Urteil könnte ein Signal sein.



10. November 2014 21:32 Uhr  
CHINA  
**Der berühmteste Wohltäter Chinas – nach eigenen Angaben**  
Der chinesische Unternehmer Chen Guangbiao wurde ausgerechnet mit Bauschutt sehr reich. Jetzt baut er Wände aus Geldbündeln und zertrümmert öffentlich Luxusautos.



11. November 2014 06:39 Uhr  
HANS MAGNUS ENZENSBERGER  
**Der Unerschütterliche**  
Hans Magnus Enzensberger wird 85. Ein Besuch bei dem herrlich eigenwilligen Intellektuellen. Mit Tumul hat er gerade ein erstaunlich persönliches Buch veröffentlicht.



11. November 2014 08:48 Uhr  
APEC-GIPFEL TREFFEN  
**Obama besänftigt APEC-Gipfel-Treffen**  
Die USA wollen China in der APEC-Gruppe. Präsident Obama vor dem Staatschef Xi. Der plädiert für mehr wirtschaftliche Verflechtung.



10. November 2014 19:17 Uhr  
KONJUNKTUR  
**China s Wachstum**  
China hat ein Rekordtempo erreicht. Doch diese Ziffern sind nur die Spitze des Eisbergs.



10. November 2014 um 18:25 Uhr  
DOR-DESIGN  
**Sandmännchen und Stasi-Mikrofone**  
Das größte Museum für DOR-Design steht ausgerechnet in Los Angeles. Ein Buch über das Wende Museum zeigt, welche Schätze und Abgründe es dort zu entdecken gibt.



10. November 2014 19:17 Uhr  
ISRAEL  
**Keiner will von Intifada sprechen**  
Messerattaken auf Israelis. Krawalle auf dem Tempelberg. Schirmzettel im Gassengewirr.



10. November 2014 19:17 Uhr  
FRANKFURT  
**Frankfurt**  
Frankfurt ist die zweitgrößte Stadt Deutschlands. Doch die meisten Menschen wissen nichts von der Stadt.



10. November 2014 um 15:25 Uhr  
AZEALIA BANKS  
**Klare Ansage aus Harlem**  
Erst galt Azealia Banks als großes Raptalent, dann als streitsüchtig und selbsterleucht. Ihr seit Jahren erwartetes Debut zeigt jetzt, wie gut das eine zum anderen passt.

**+156%**

**BaQend**

**0,5s**

**[WE RT]**

**FRANKFURT**

# Backend-as-a-Service

## Tutorial on the BaaS paradigm from app perspective

[Overview](#)[Tutorial](#)[Download](#)[Architecture](#)[Docs](#)

### Updating and deleting data

1 2 3 4 5 6 7 8 9

Both updates and inserts are performed by calling `save` on an object.

If at the time of the update the local copy was outdated, the operation will result in an error. In the error callback we could either refresh the object with `myTodo.load()` and retry the update or decide to overwrite the newer version at the server with our older version:

```
myTodo.save({
  force : true //Overwrite even if our copy is outdated
});
```

Deleted and saved! >

When objects reference each other, we can control, up to which depth referenced objects should also be persisted using the `depth` parameter (*persistence-by-reachability*).

If we want to get rid of an object, we do a `myTodo.delete()`. `delete` has the same options as `save` and behaves similarly.

The screenshot shows a web application interface. On the left, there is a code editor with the following code:

```
1 //The TodoService handles access to persistent Todo items
2 var listId = "tutorial-list" + Math.random();
3 var TodoService = (function() {
4   return {
5     //save a Todo
6     save: function(todo) {
7       return todo.save();
8     },
9     //delete a Todo
10    delete: function(id) {
11      return DB.Todo.load(id).then(function(todo) {
12        return todo.delete();
```

On the right, there is a "Todo List" section with a green checkmark icon and the text "First Todo". A white box in the bottom right corner contains the URL "www.baqend.com".

# Thank you

[ritter,gessert@informatik.uni-hamburg.de](mailto:ritter,gessert@informatik.uni-hamburg.de)

[orestes.info](http://orestes.info), [baqend.com](http://baqend.com)