
Orestes
Objects RESTfully Encapsulated
in Standardformats

Why latency kills Database-as-a-Service and how to overcome it

W-JAX 2012

Felix Gessert, Florian Bücklers

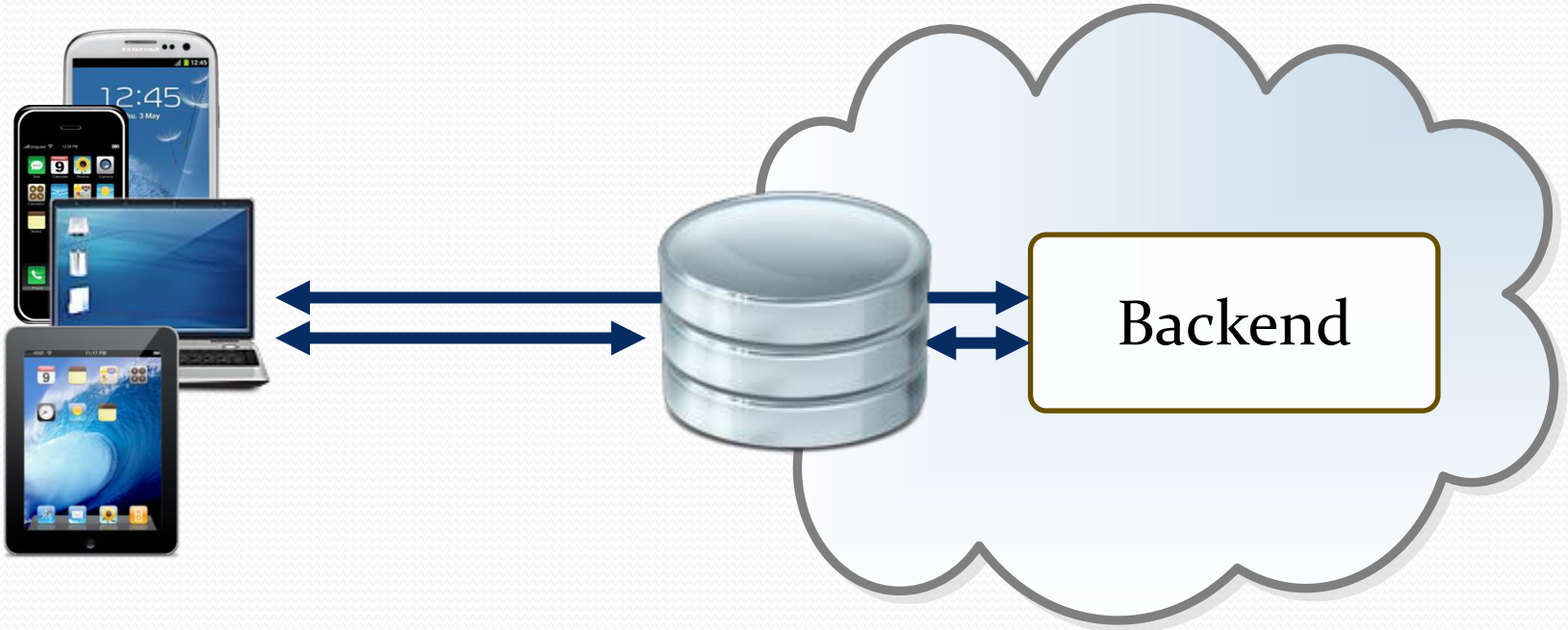
DBaaS

Probleme: Latenz & Skalierung

Orestes

Live Demo

DBaaS



HTML5 Killer App

DBaaS

Cloud

Cloudstack

Database.com

S3, SimpleDB, …

SQL Azure

Amazon RDS

Xeround

Database-as-a-Service

…

Software-as-a-Service

Platform-as-a-Service

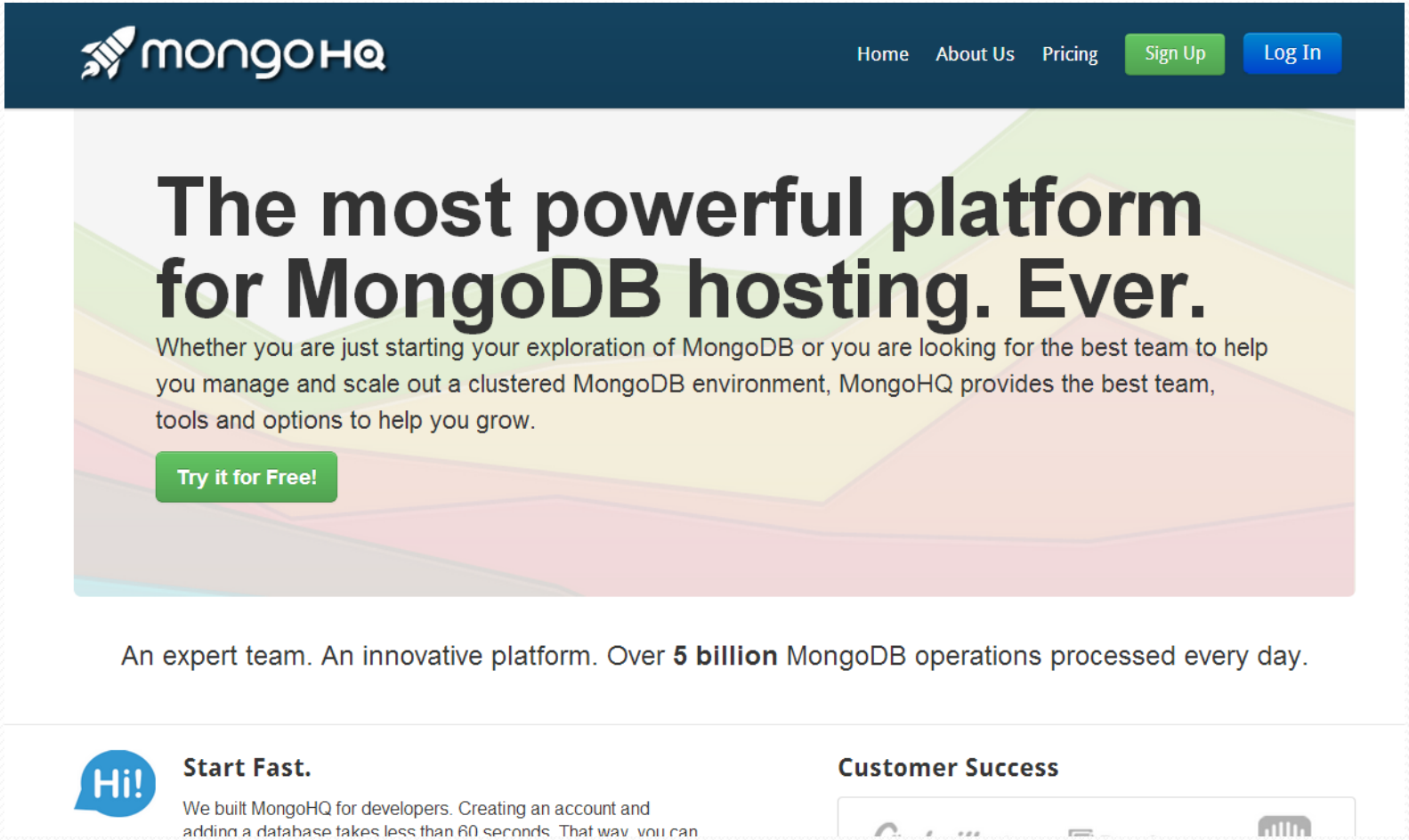
Infrastructure-as-a-Service

Ein Beispiel

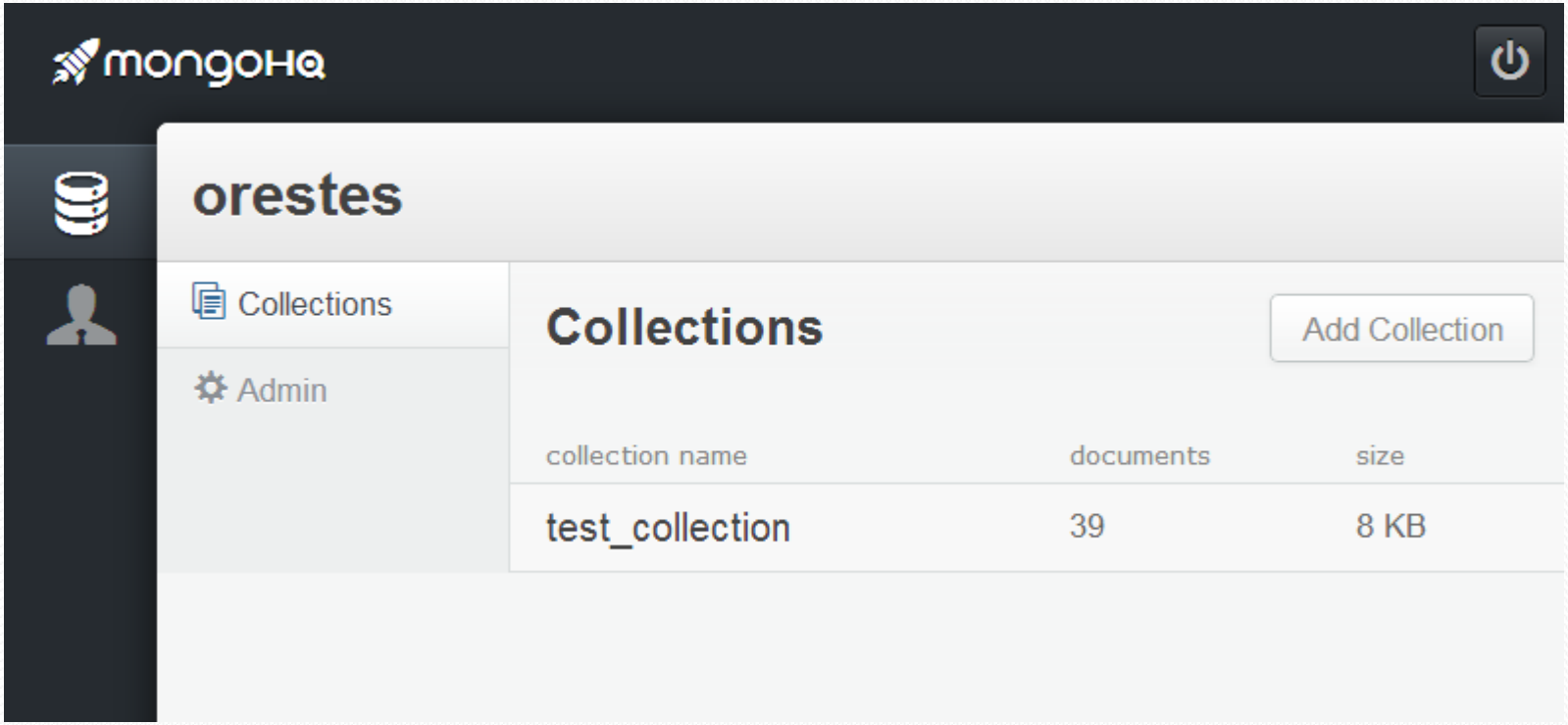
bietet an

as-a-Service

1. Anmelden


mongoHQ
Home
About Us
Pricing
Sign Up
Log In
The most powerful platform for MongoDB hosting. Ever.
Whether you are just starting your exploration of MongoDB or you are looking for the best team to help you manage and scale out a clustered MongoDB environment, MongoHQ provides the best team, tools and options to help you grow.
Try it for Free!
An expert team. An innovative platform. Over 5 billion MongoDB operations processed every day.
Hi!
Start Fast.
We built MongoHQ for developers. Creating an account and
Customer Success

2. DB starten

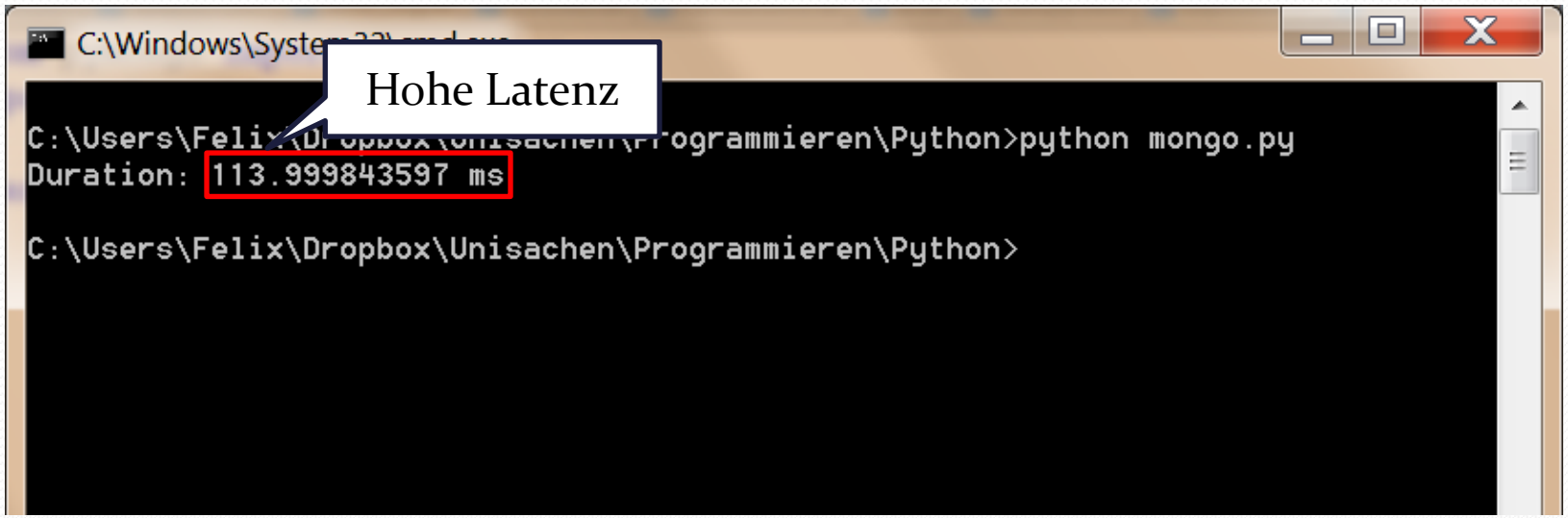

mongoHQ
orestes
Collections
Admin
Collections
Add Collection
collection name
documents
size
test_collection
39
8 KB

3. Killer App entwickeln

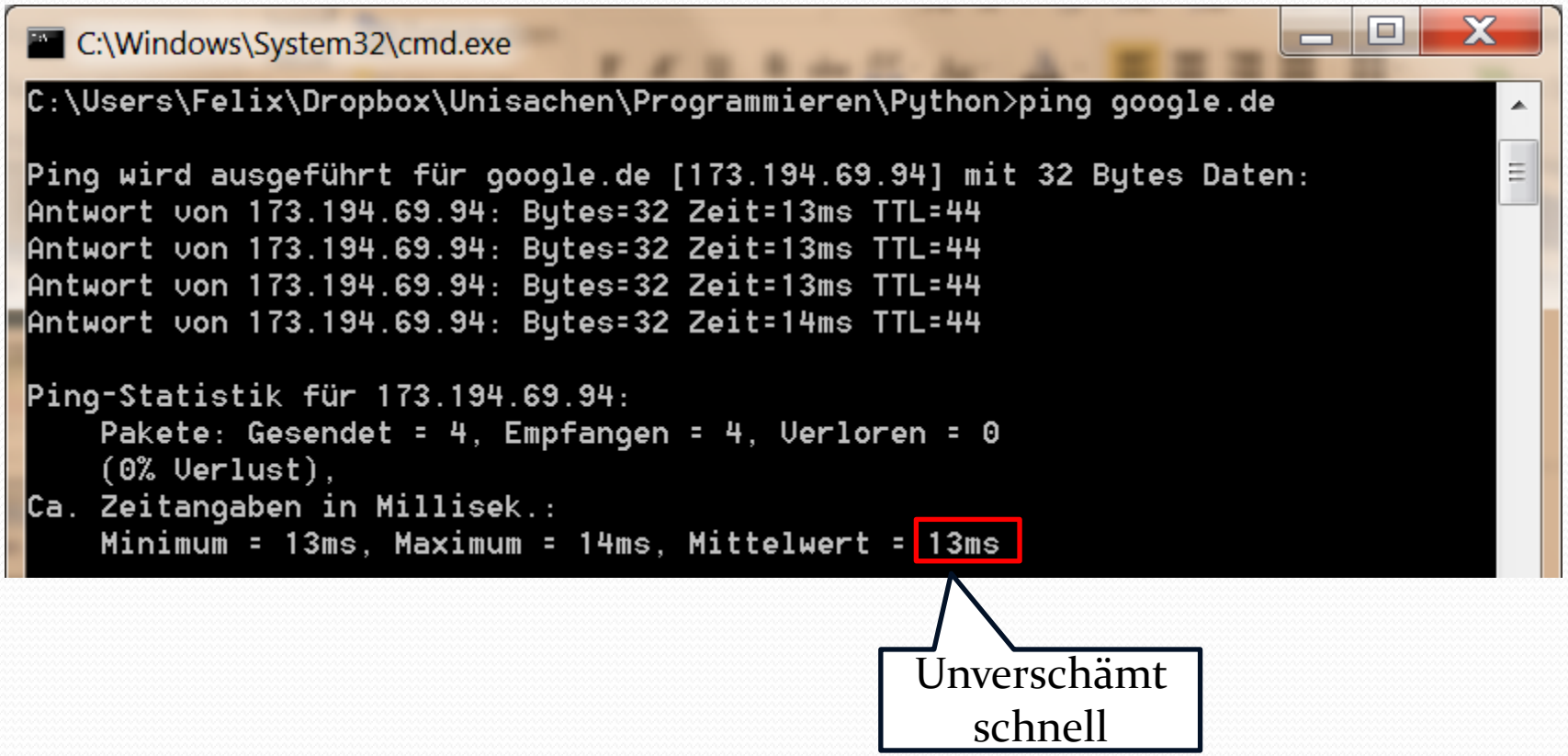
```
c = Connection('mongodb://orestes:BrotBrot@alex.mongohq.com:10008/orestes')
obj = { "User" : "Felix", "message" : "Testing MongoDB", "test" : [1,2,3,4] }
c.orestes.test_collection.insert(obj)

with timer:
    c.orestes.test_collection.find_one( {"User":"Felix" })
timer.show()
```

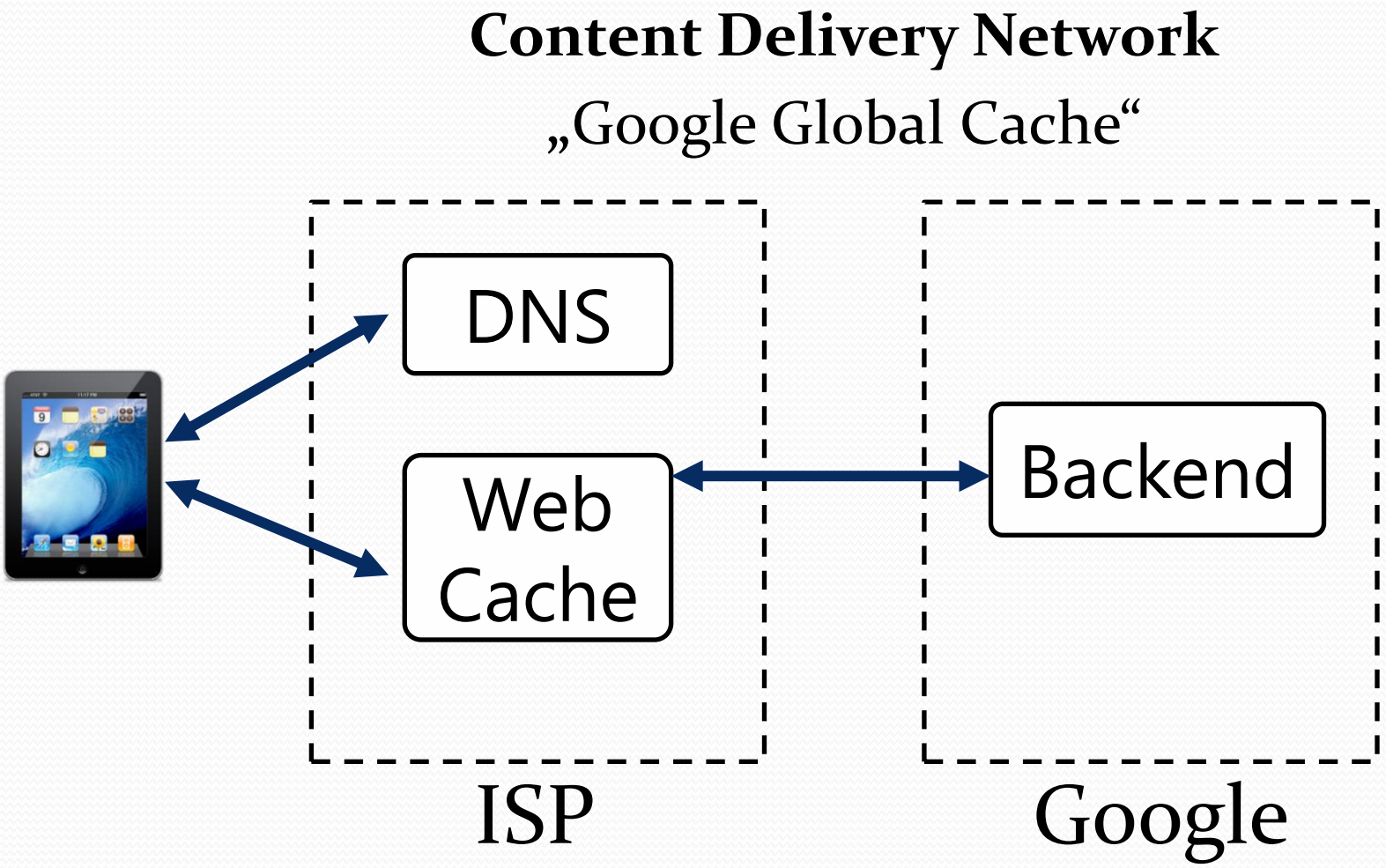
4. Grübeln


Hohe Latenz
C:\Users\Felix\Dropbox\Unisachen\Programmieren\Python>python mongo.py
Duration: 113.999843597 ms
C:\Users\Felix\Dropbox\Unisachen\Programmieren\Python>

Dagegen Google



Wie macht Google das?



Was wenn…

auch

Datenbanken

die

Web-Caching

Infrastruktur

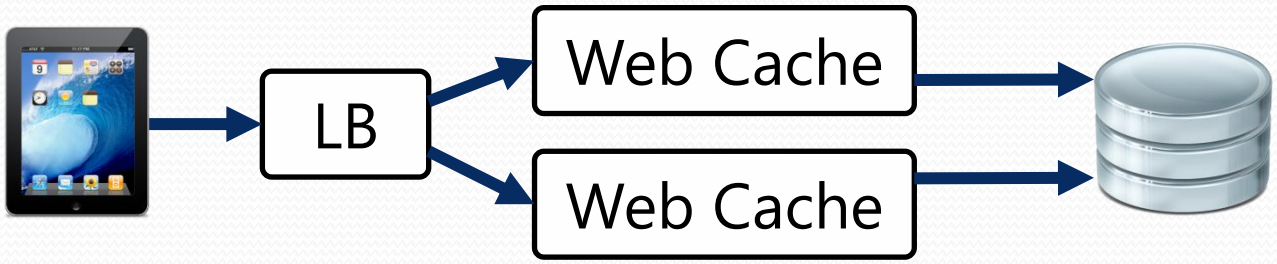
nutzen könnten ?

Dazu müsste…

Die Datenbank REST/HTTP sprechen

Dazu müsste…

Web-Caching & Load-Balancing unterstützen



Web Caching


Client


Web-Cache

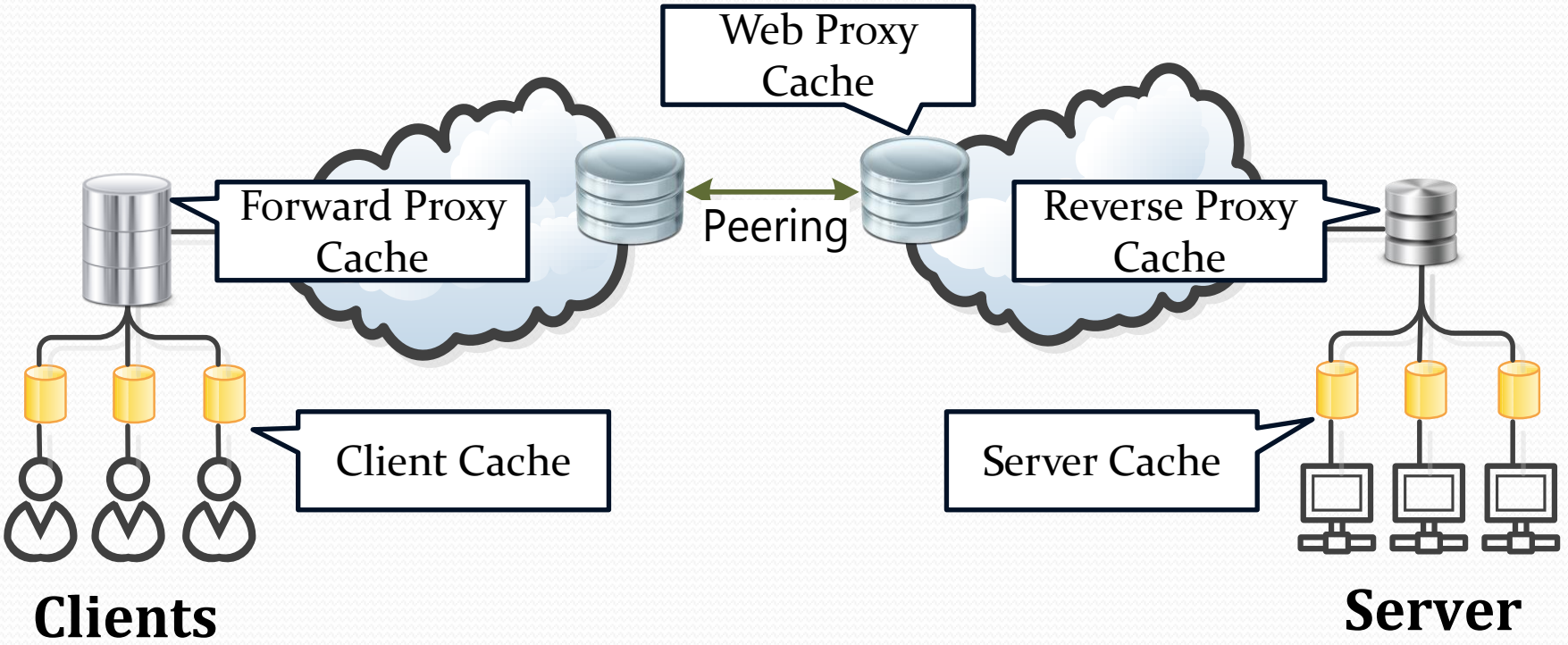

Origin Server

Web Caching

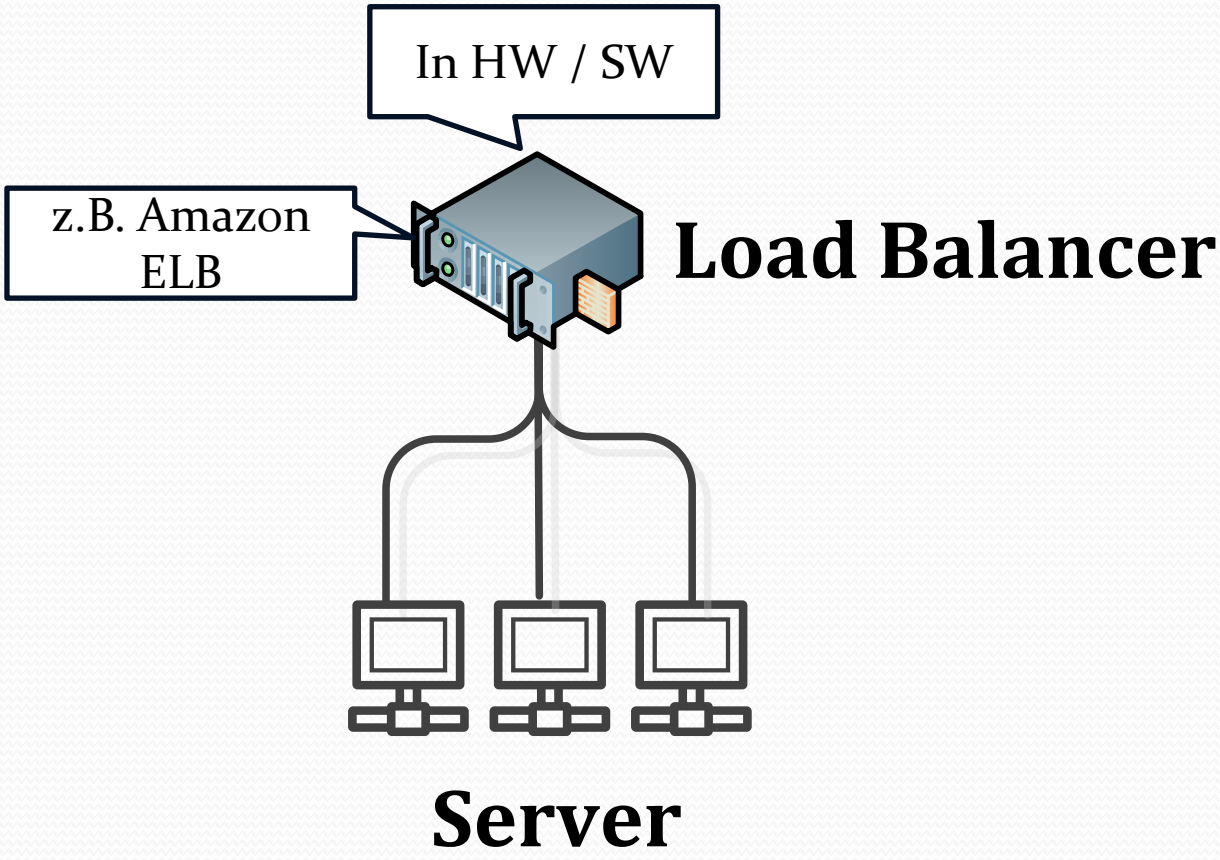
Expiration
Server bestimmt „Verfallsdatum“

Revalidation
Etwaige Änderungen erfragen

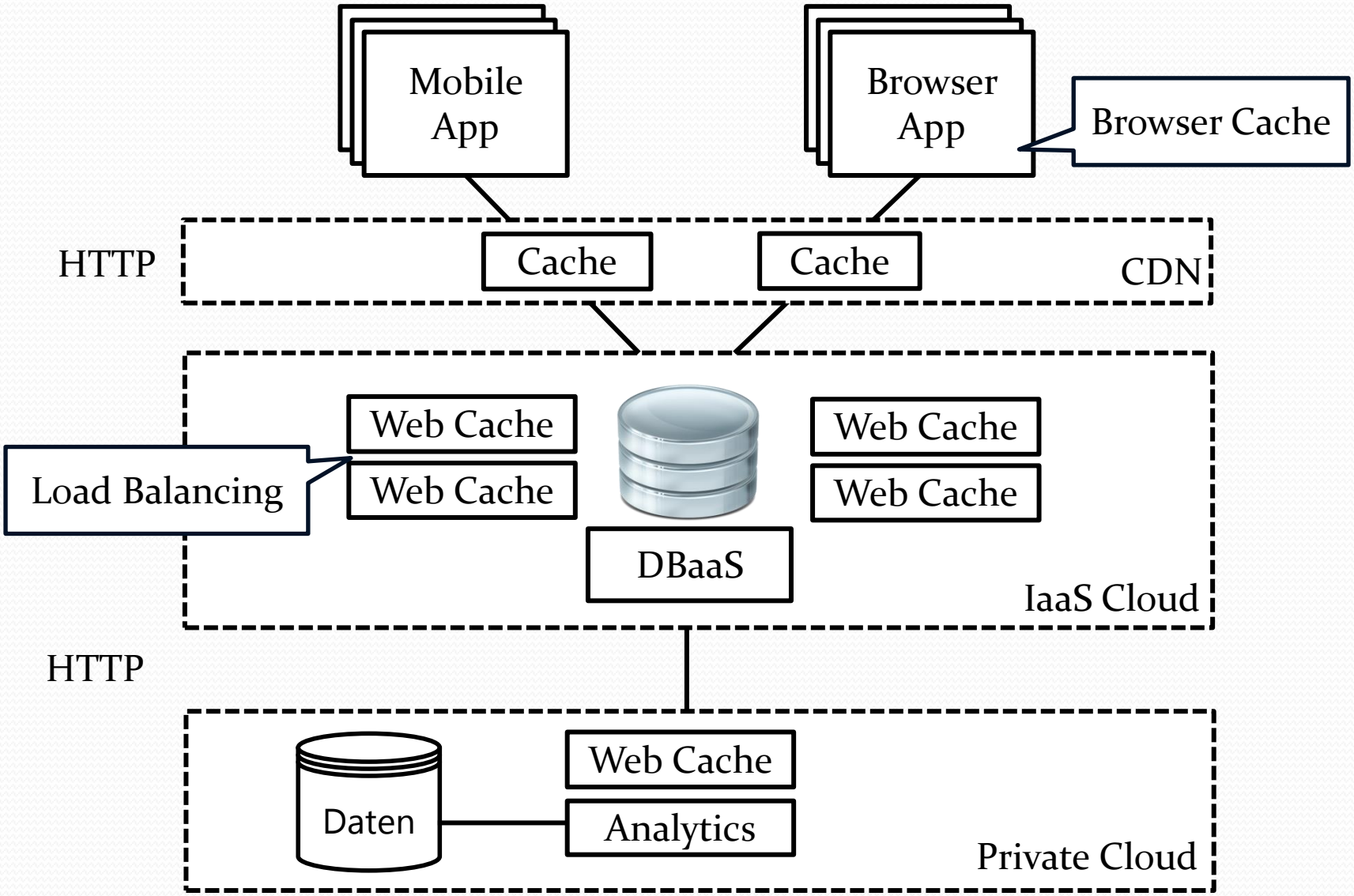
Web Caches


Web Proxy Cache
Forward Proxy Cache
Peering
Reverse Proxy Cache
Client Cache
Server Cache
Clients
Server

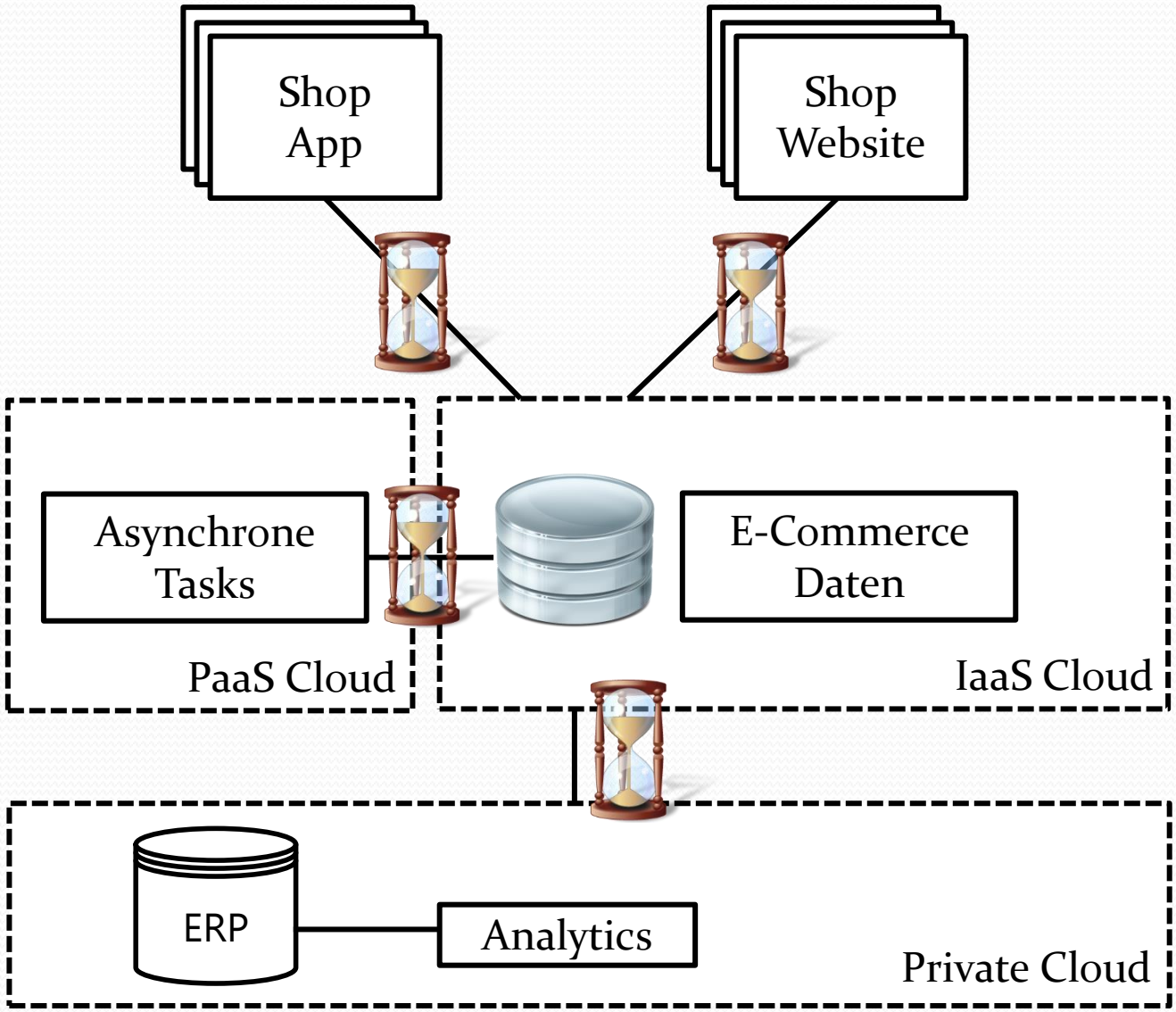
Load Balancing



Überall Caches



Überall Latenz



Orestes

Ein RESTful HTTP Layer für Datenbanken

Horizontale Skalierbarkeit

Transparente und intuitive Datenbankschnittstelle

Unterstützung von Polyglot Persistence

Nutzung von Web Standards

Web Caching

REST/HTTP

Verschiedene Backends

JSON

Orestes

Ein RESTful HTTP Layer für Datenbanken

Webbasiertes Monitoring, Deployment, Administrieren

Web Interface

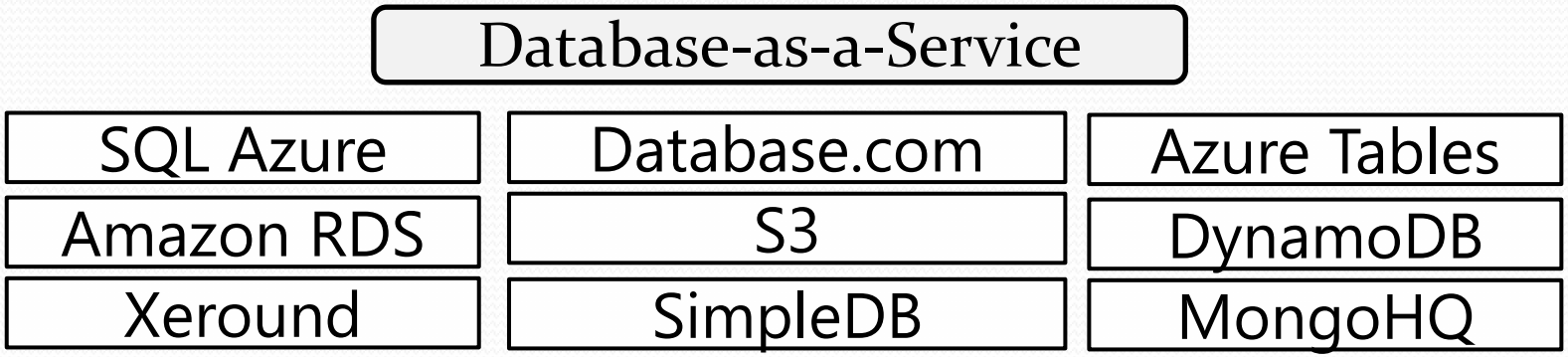
Zugriff mit niedriger Netzwerklatenz

CDNs & Web Caches

Transaktionen und starke Konsistenz als Option

ACID Transactions

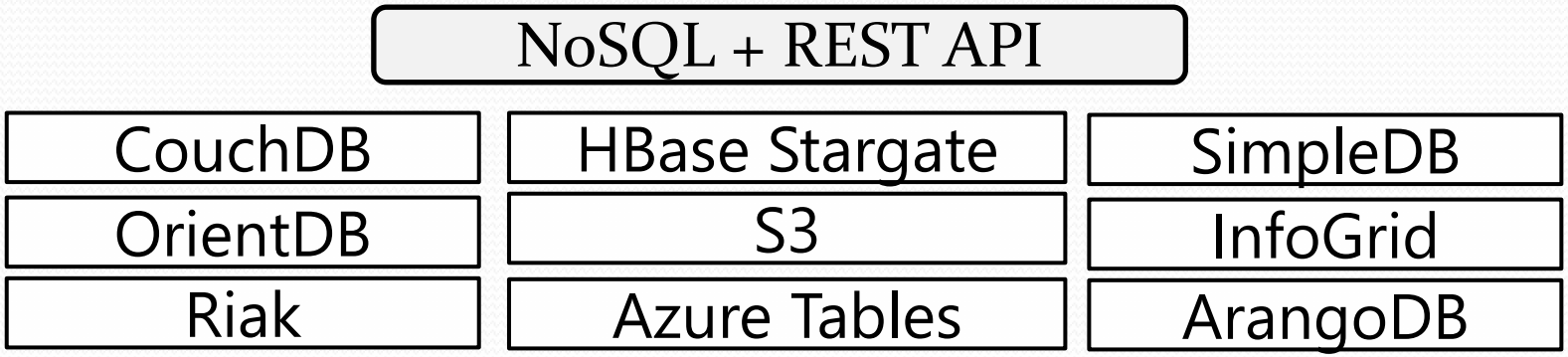
Gibt es das nicht schon?



Probleme:

- Kein REST
- Hohe Latenz
- Schwere horizontale Skalierung

Gibt es das nicht schon?



Probleme:

- Hohe Latenz
- Web-Infrastruktur
- Transaktionen & Concurrency

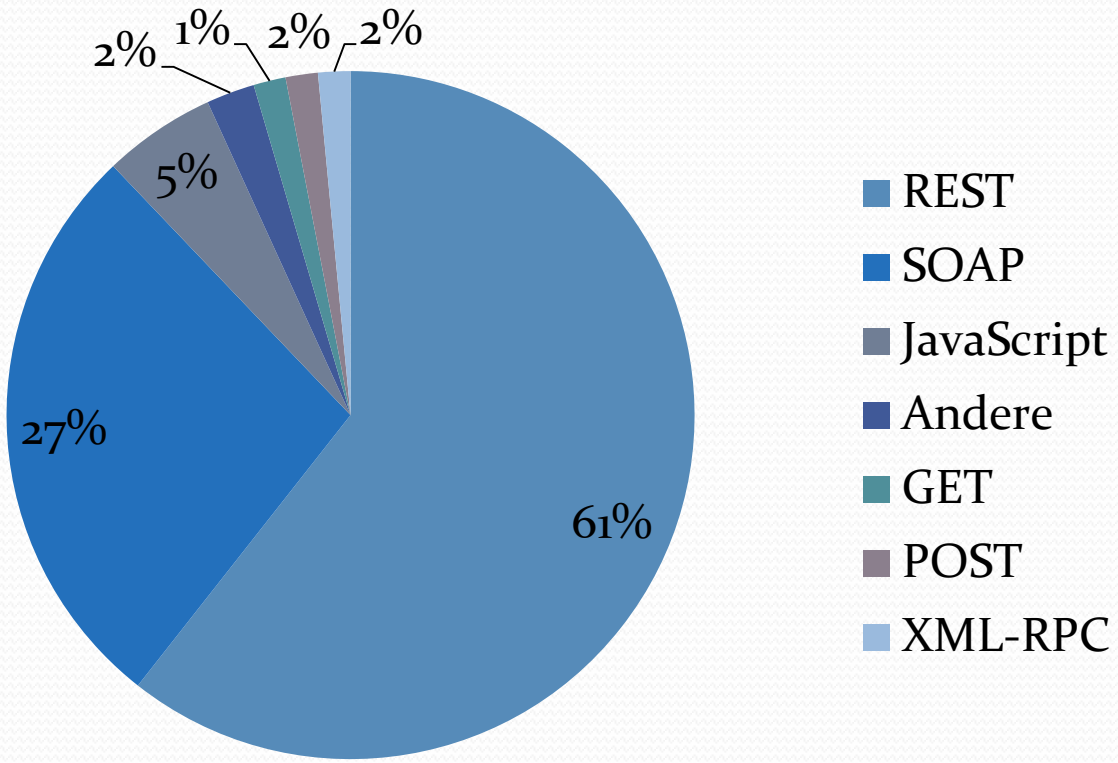
Gibt es das nicht schon?

123 Database APIs:

GeoNames, Freebase and Yahoo Query Language

Mai 2012

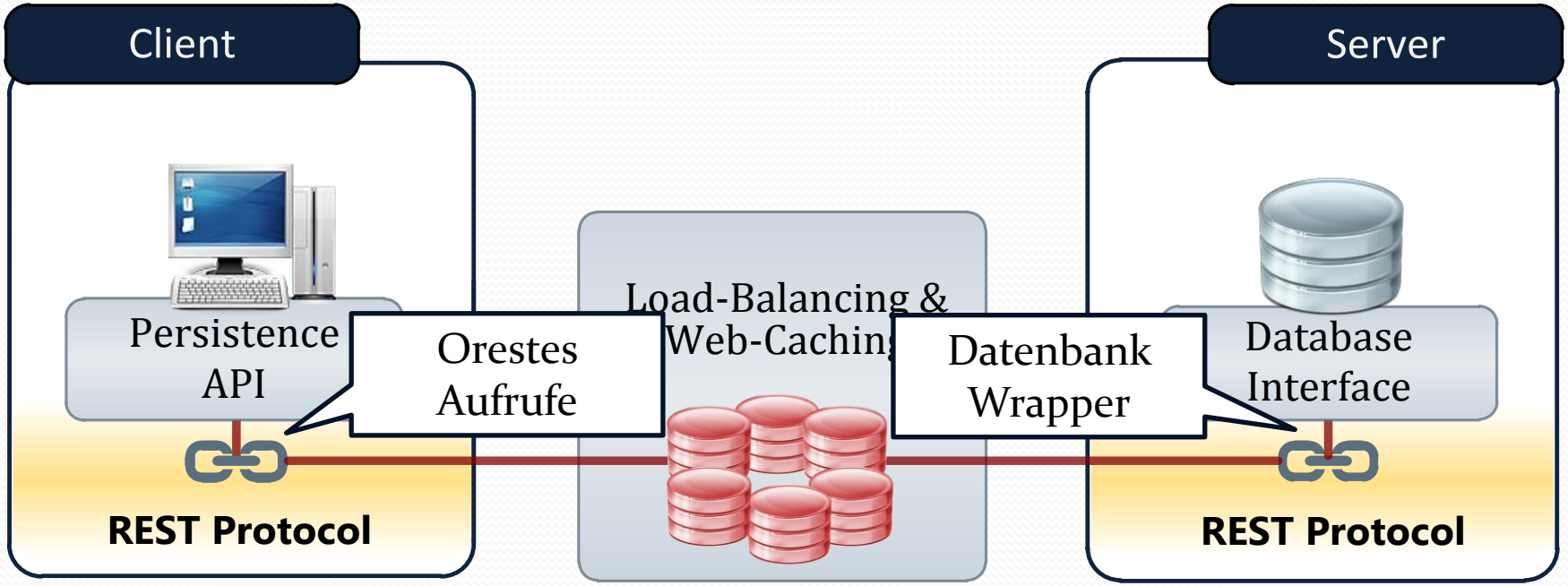
Gibt es das nicht schon?



Probleme:

- Keine universelle API

Orestes: Ansatz



REST Protokoll

Modell: **Objektorientierte Persistenz**

Java: **JPA 2**

```
factory = Persistence.createEntityManagerFactory(PU_NAME);
EntityManager em = factory.createEntityManager();
em.getTransaction().begin();
Coffee c = new Coffee();
c.setName(“Robusta Brasilian Premium.");
em.persist(todo);
em.getTransaction().commit();
```

REST Protokoll

Modell: **Objektorientierte Persistenz**

Java: **JPA 2**

```
Query q = em.createQuery("select c from Coffee c");
List<Coffee> coffeeList = q.getResultList();
for (Coffee c : coffeeList) {
      System.out.println(c);
}
```

REST Protokoll

JPA

```
em.getTransaction().begin();
```

REST

```
> POST transaction

< 201 Created
< Location: /transaction/1
```

REST Protokoll

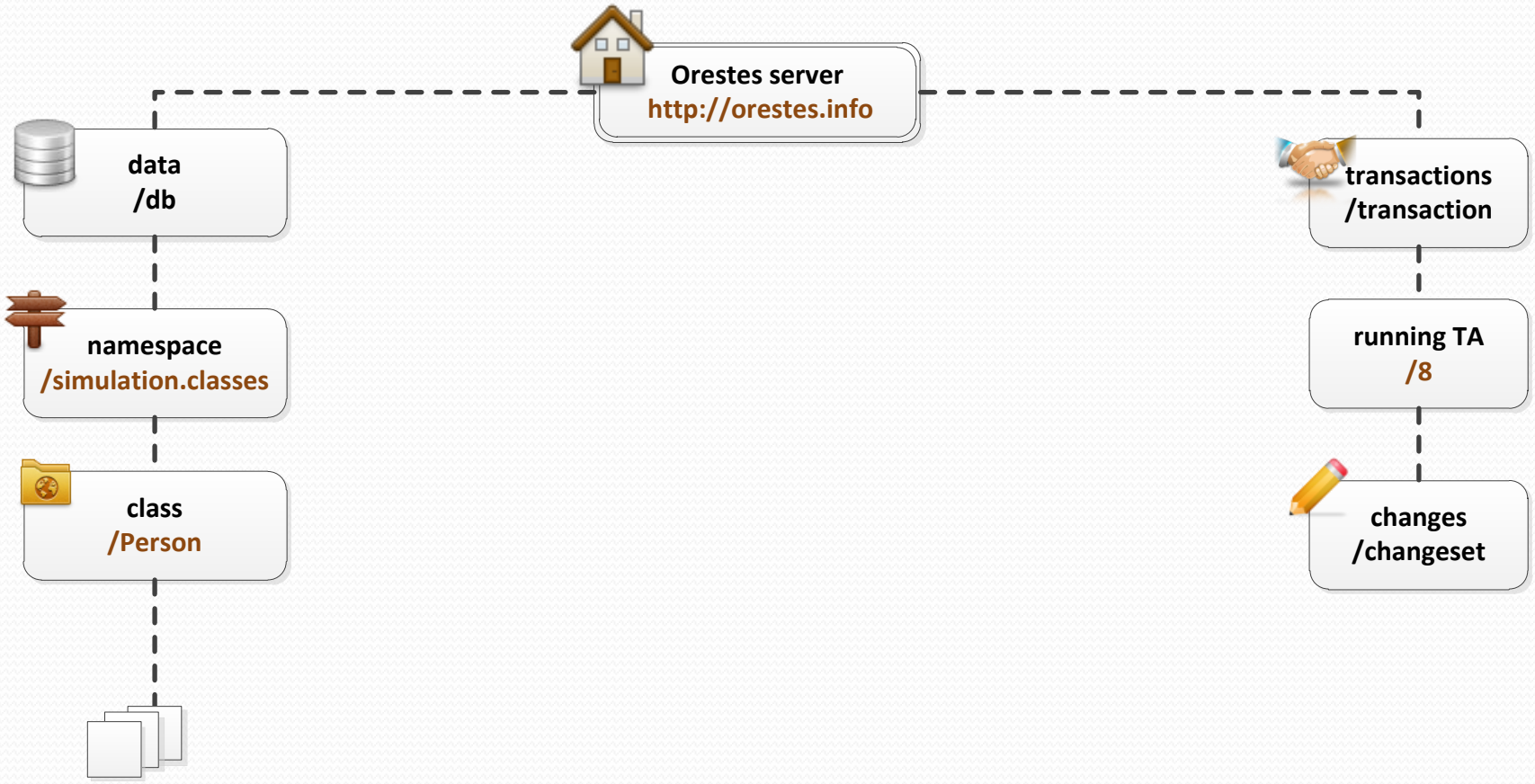
JPA

```
em.persist(coffee);
em.flush();
```

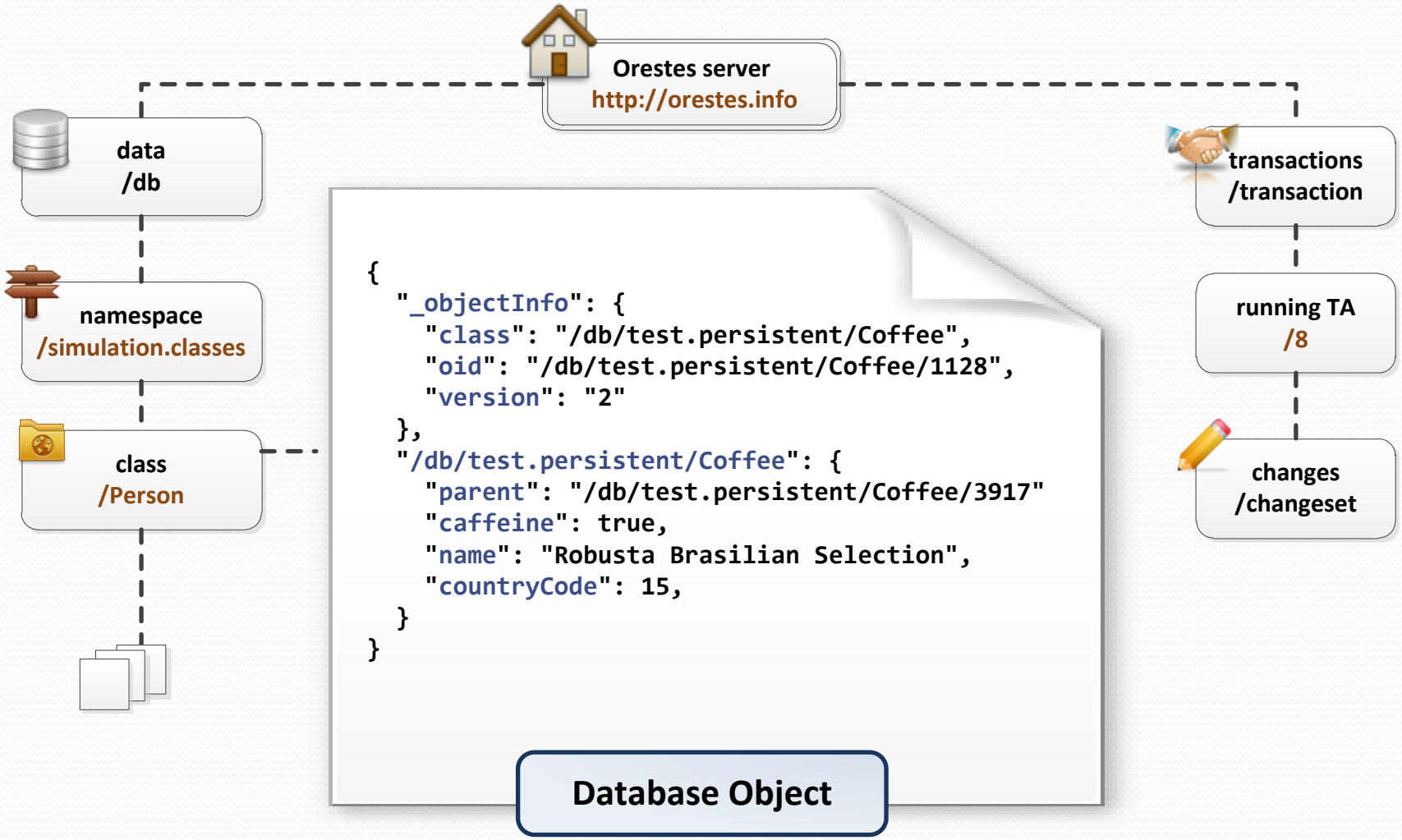
REST

```
> PUT /db/test.persistent/Coffee/1125899906869258
> If-Match: "2"
{
  "_objectInfo": {
    "oid": "/db/test.persistent/Coffee/1125899906869258",
    "class": "/db/test.persistent/Coffee",
    "version": "2"
  },
  "/db/test.persistent/Coffee": {
    "caffeine": true,
    "name": "Robusta Brasilian Selection",
    "countryCode": 15,
    "parent": null
  }
}
```

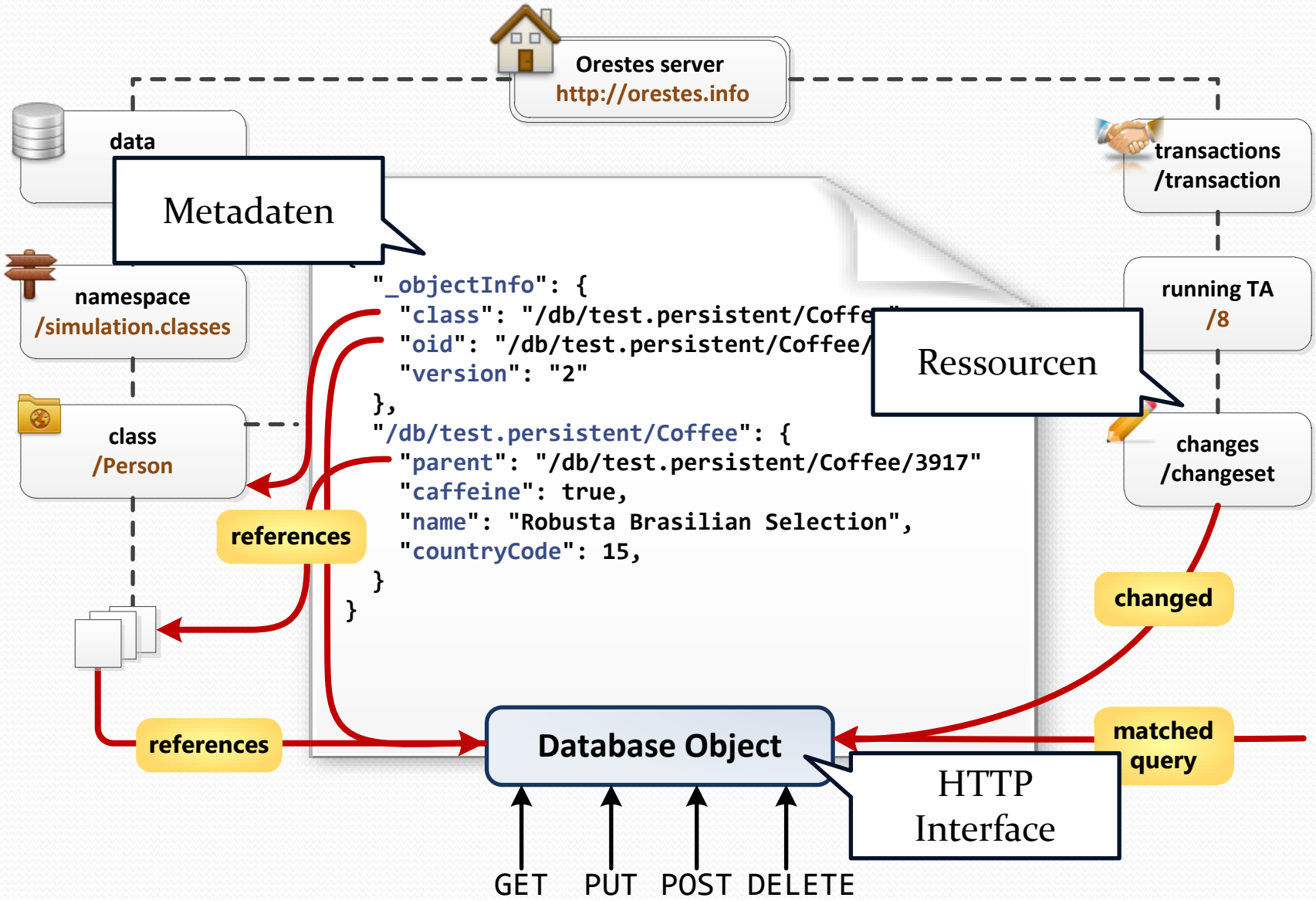
Ressourcen


Orestes server
http://orestes.info
data
/db
namespace
/simulation.classes
class
/Person
transactions
/transaction
running TA
/8
changes
/changeset

Ressourcen

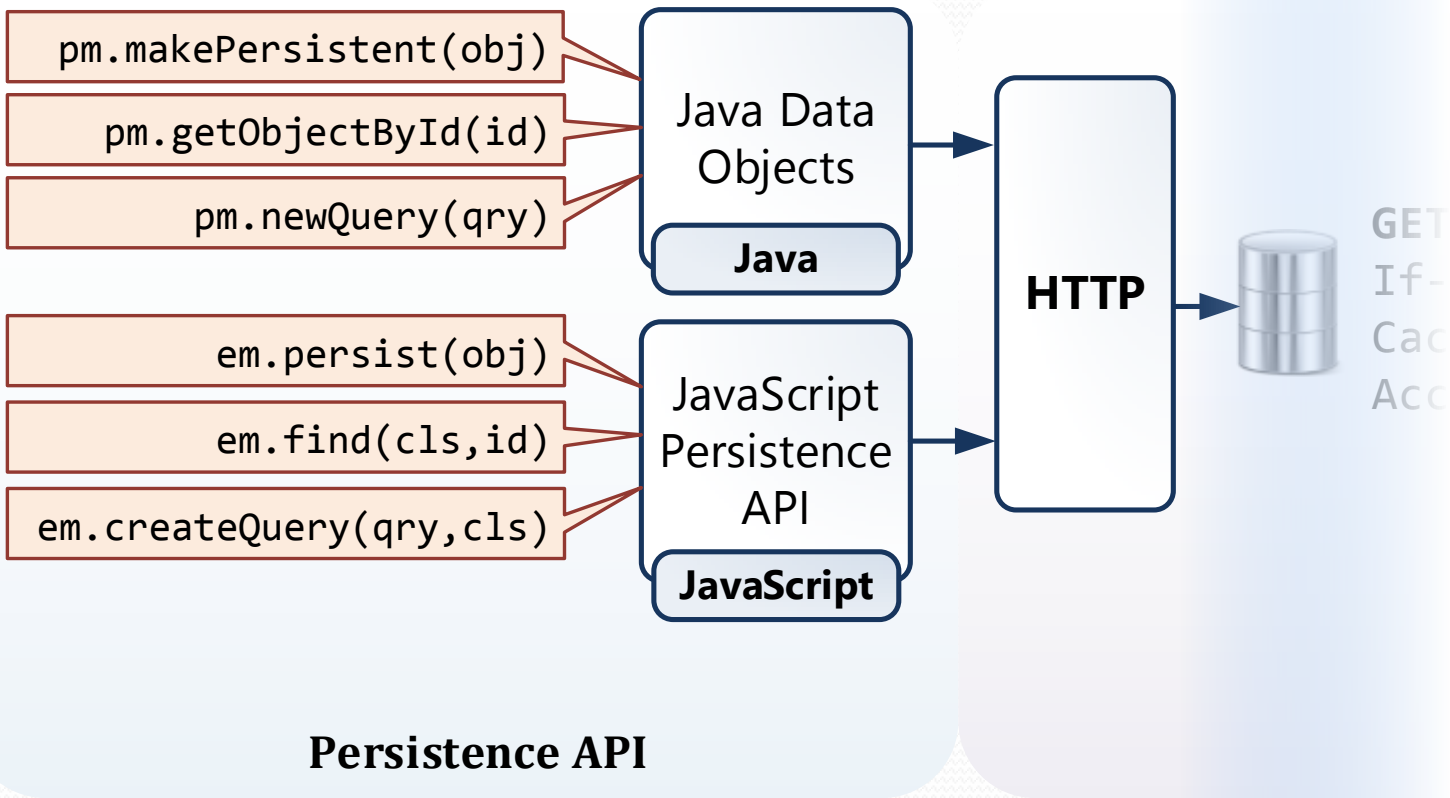

Orestes server
http://orestes.info
data
/db
namespace
/simulation.classes
class
/Person
transactions
/transaction
running TA
/8
changes
/changeset
{
"_objectInfo": {
"class": "/db/test.persistent/Coffee",
"oid": "/db/test.persistent/Coffee/1128",
"version": "2"
},
"/db/test.persistent/Coffee": {
"parent": "/db/test.persistent/Coffee/3917"
"caffeine": true,
"name": "Robusta Brasilian Selection",
"countryCode": 15,
}
}
Database Object

Ressourcen

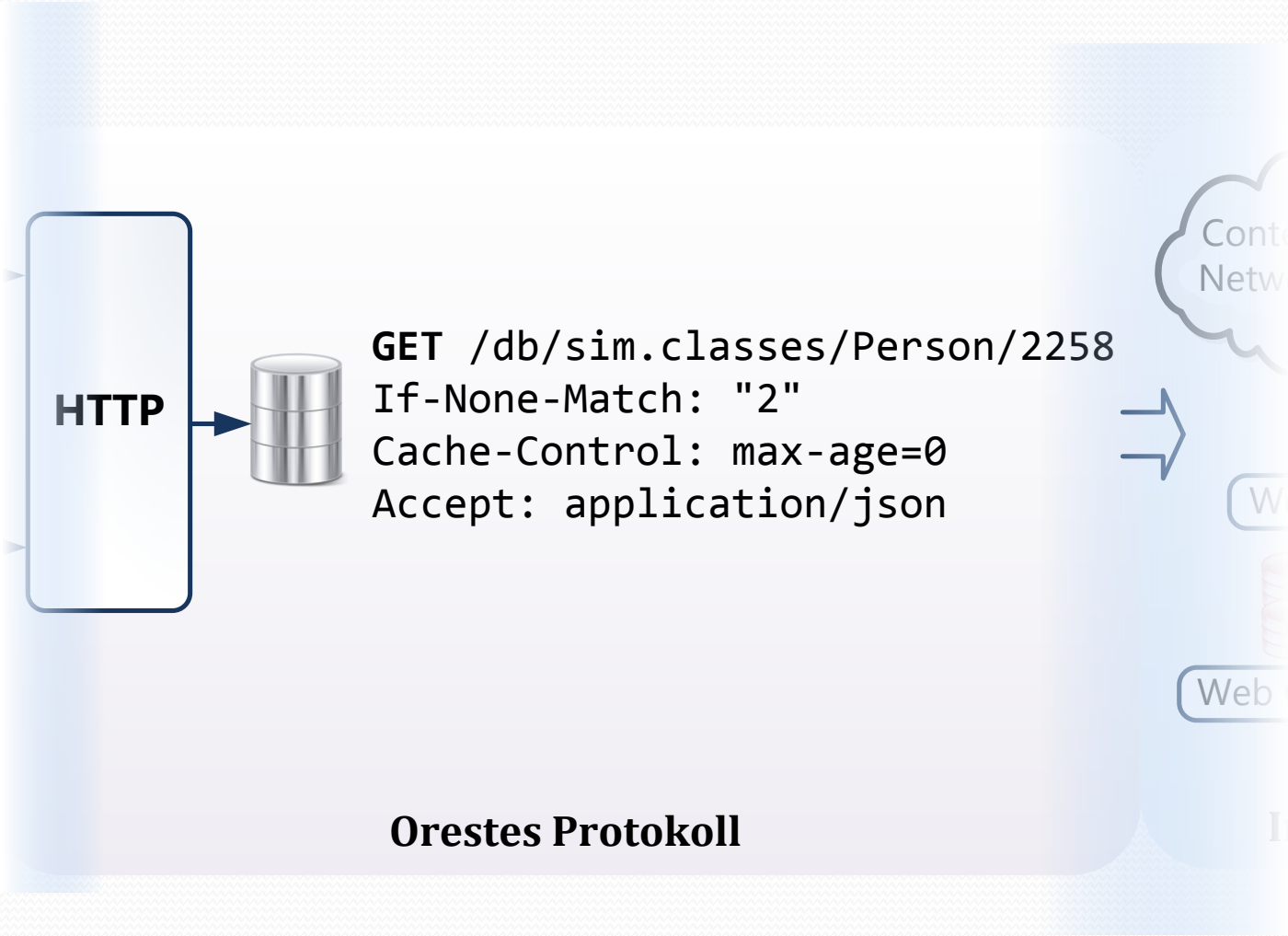

Orestes server
http://orestes.info
data
transactions
/transaction
Metadaten
namespace
/simulation.classes
running TA
/8
class
/Person
Ressourcen
changes
/changeset
"_objectInfo": {
"class": "/db/test.persistent/Coffe
"oid": "/db/test.persistent/Coffee/
"version": "2"
},
"/db/test.persistent/Coffee": {
"parent": "/db/test.persistent/Coffee/3917"
"caffeine": true,
"name": "Robusta Brasilian Selection",
"countryCode": 15,
}
}
references
changed
references
Database Object
matched
query
HTTP
Interface
GET
PUT
POST
DELETE

Beispiel

Lade die neuste Version eines Objektes

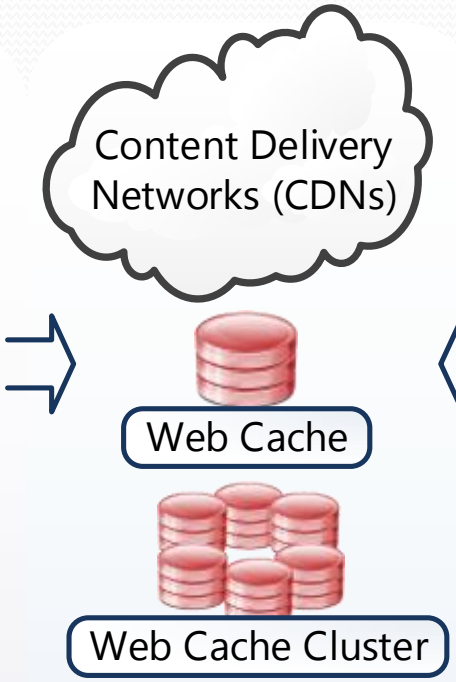


Beispiel


HTTP
GET /db/sim.classes/Person/2258
If-None-Match: "2"
Cache-Control: max-age=0
Accept: application/json
Orestes Protokoll

Beispiel

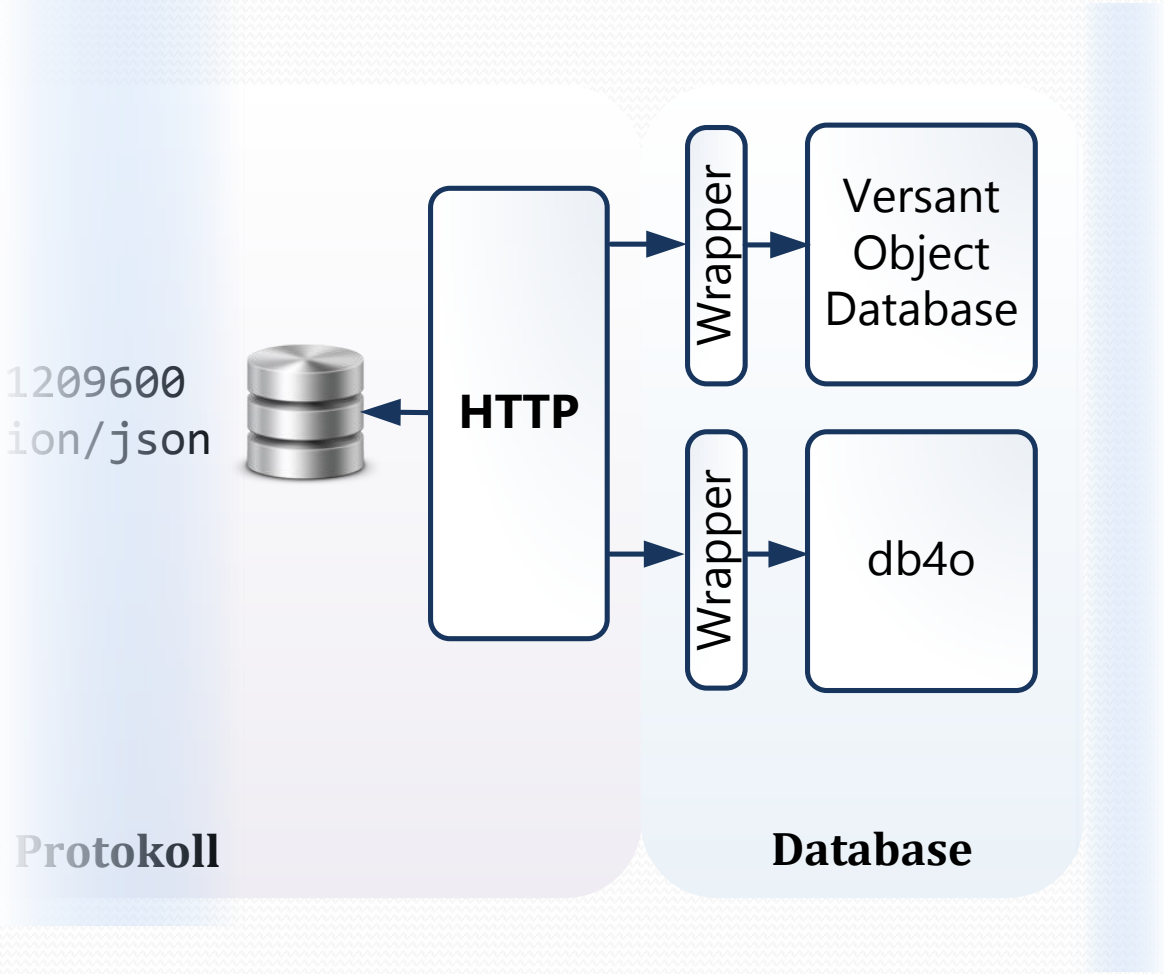
```
GET /db/sim.classes/Person/2258
If-None-Match: "2"
Cache-Control: max-age=0
Accept: application/json
```



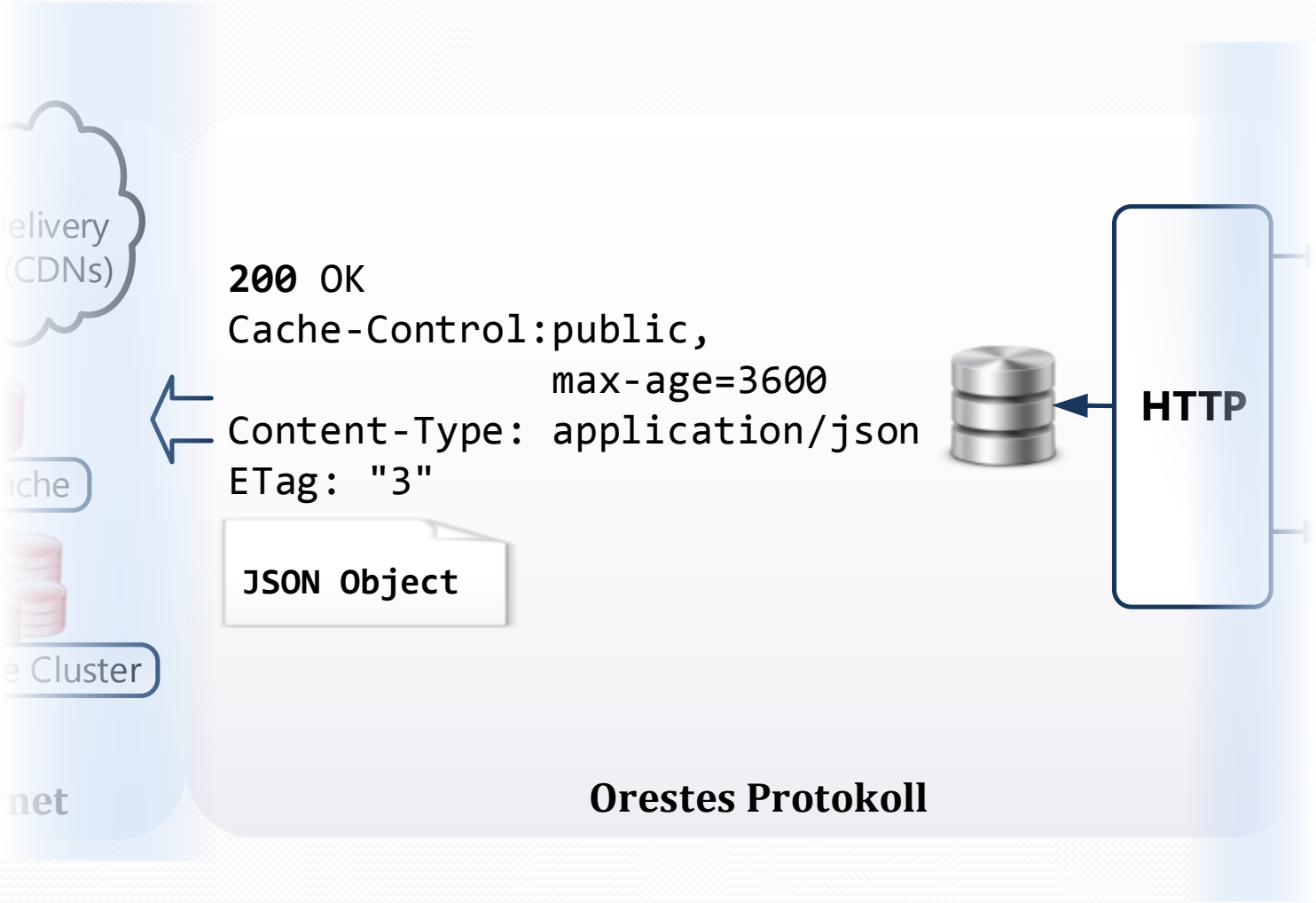
Orestes Protokoll

Internet

Beispiel

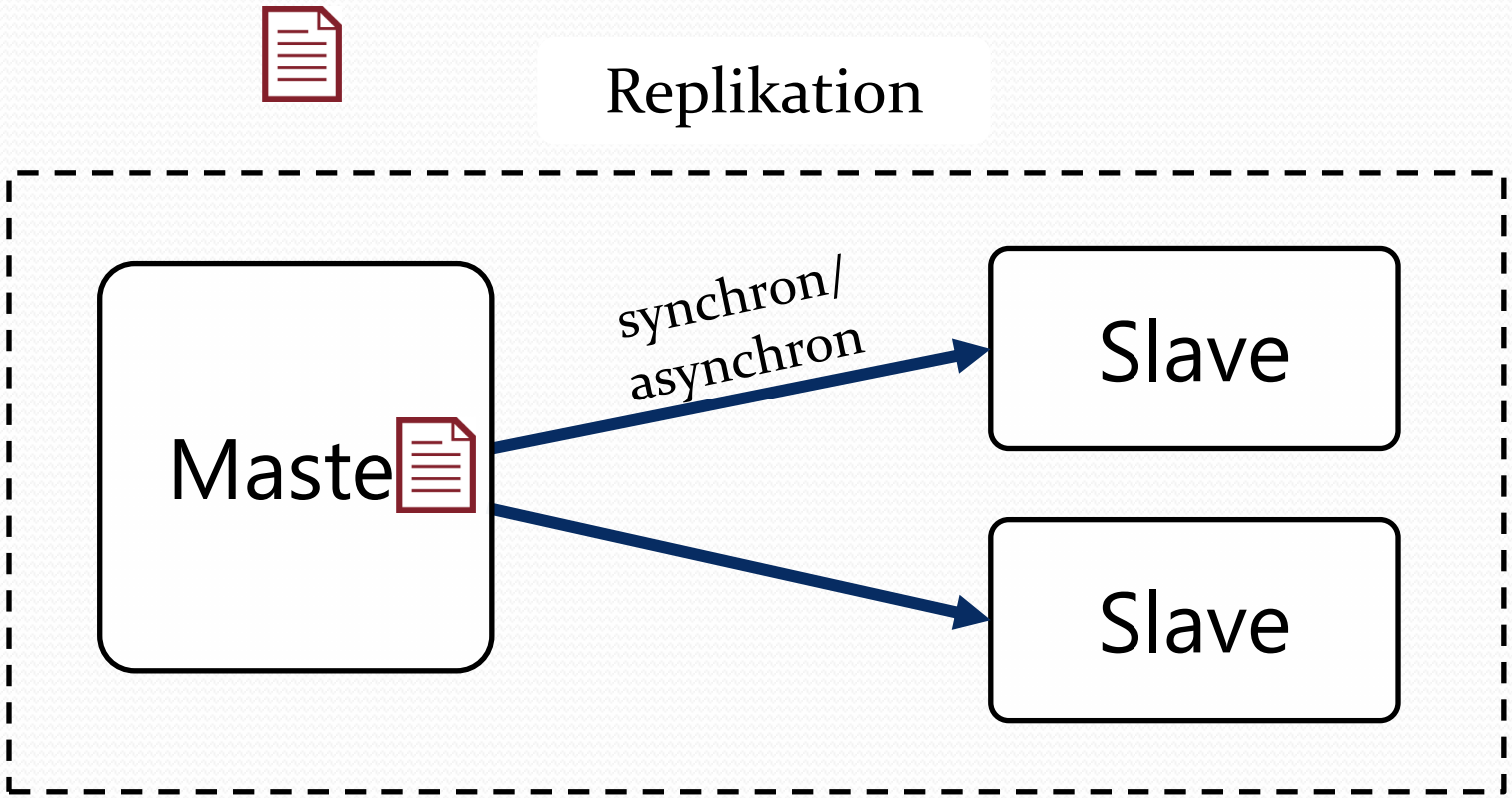

1209600
ion/json
HTTP
Wrapper
Wrapper
Versant
Object
Database
db4o
Protokoll
Database

Beispiel


elivery
(CDNs)
ache
e Cluster
net
200 OK
Cache-Control:public,
max-age=3600
Content-Type: application/json
ETag: "3"
JSON Object
HTTP
Orestes Protokoll

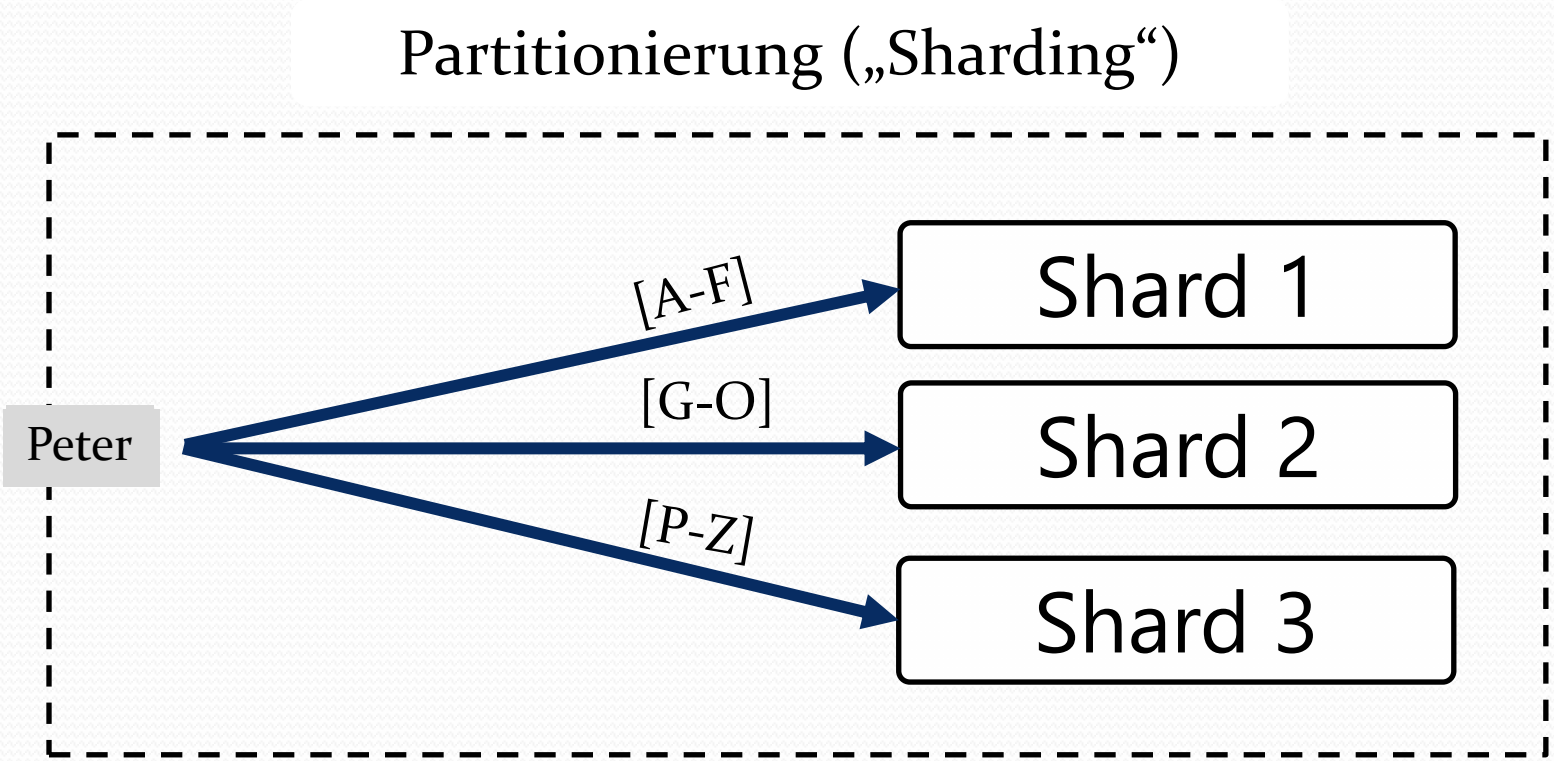
Skalierung

Klassische Technik:

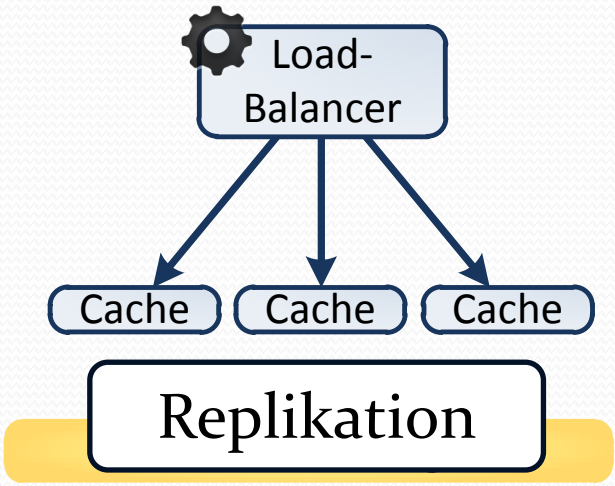


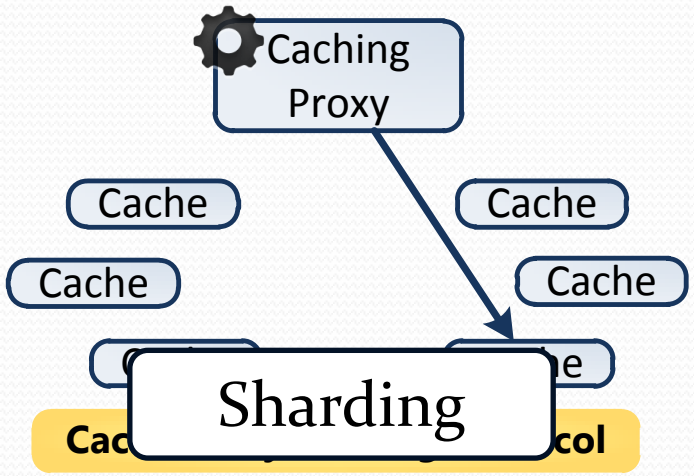
Skalierung

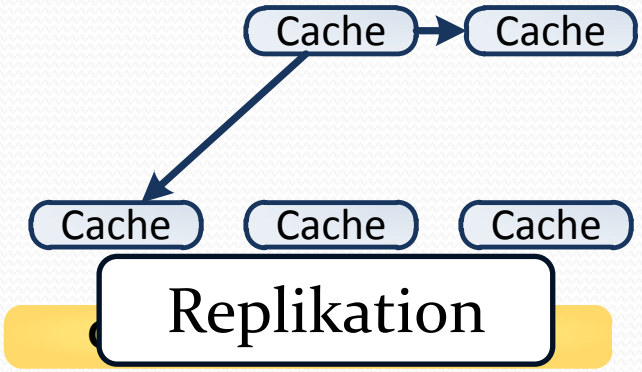
Klassische Technik:

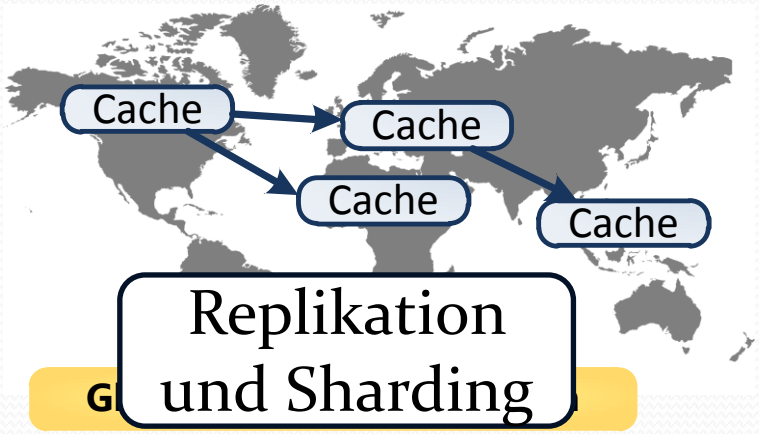


Skalierung: Web-Caches









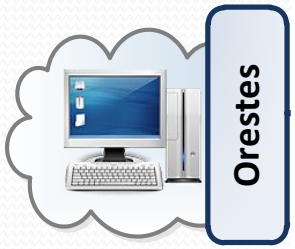
Skalierung: Web-Caches

…machen es wie Datenbanken

nur ohne **Cache Kohärenz**

und ohne eigene **Logik**

Aber…



App

Web Cache

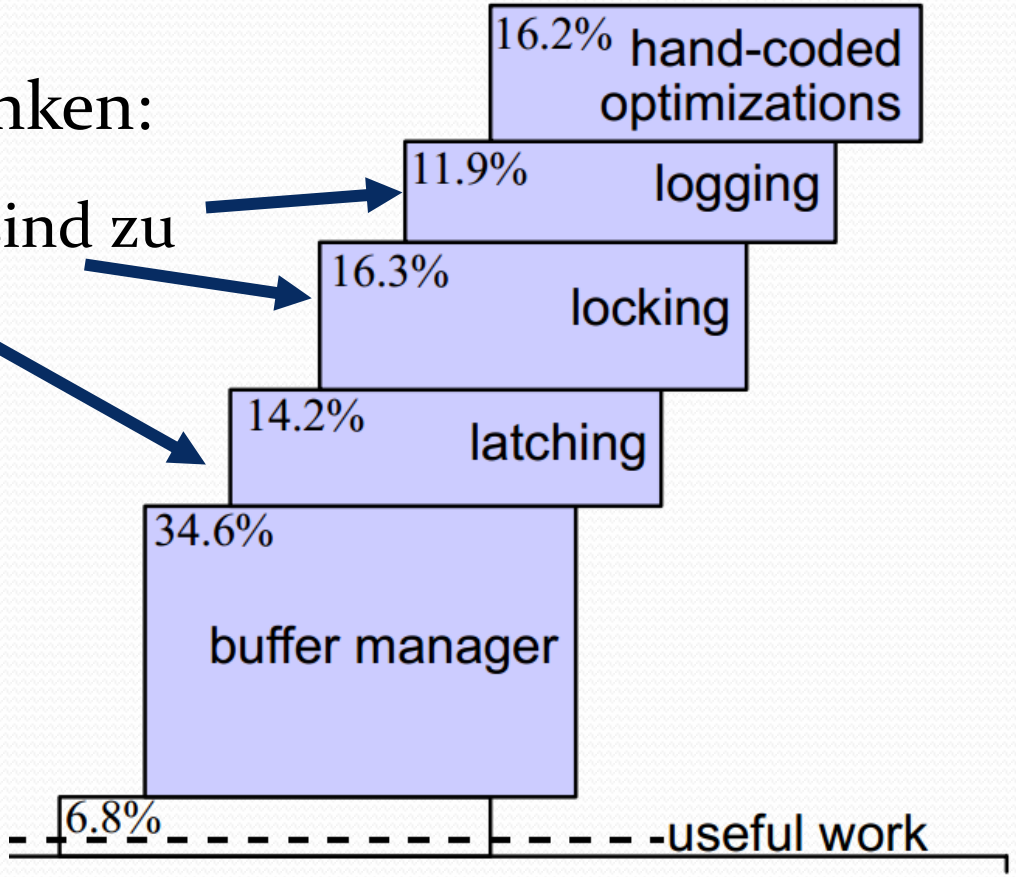
DB

Was ist mit „**Stale Reads**“?

Transaktionen

NoSQL Datenbanken:

„Transaktionen sind zu aufwendig“

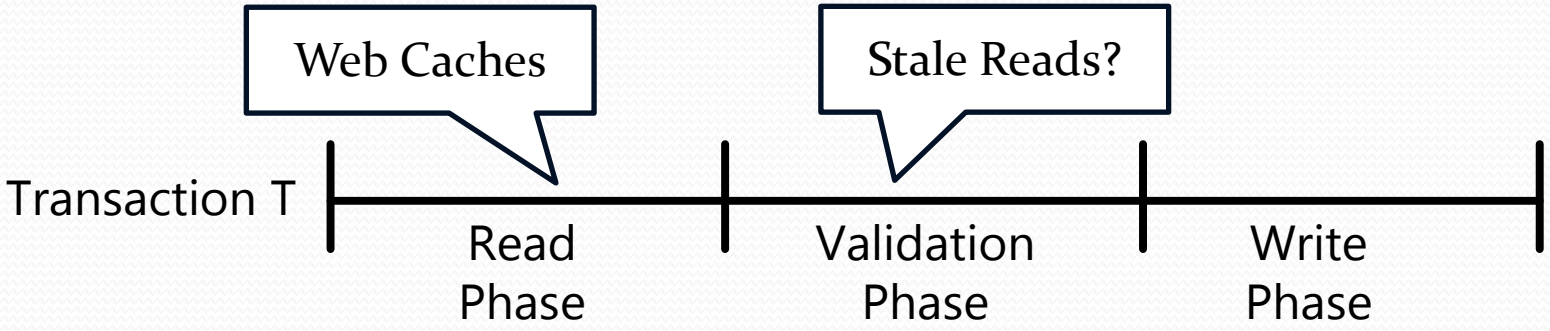


[Stonebraker]

Transaktionen in Orestes

Optimistische Transaktionen:

- Kein Locking



Transaktionen in Orestes

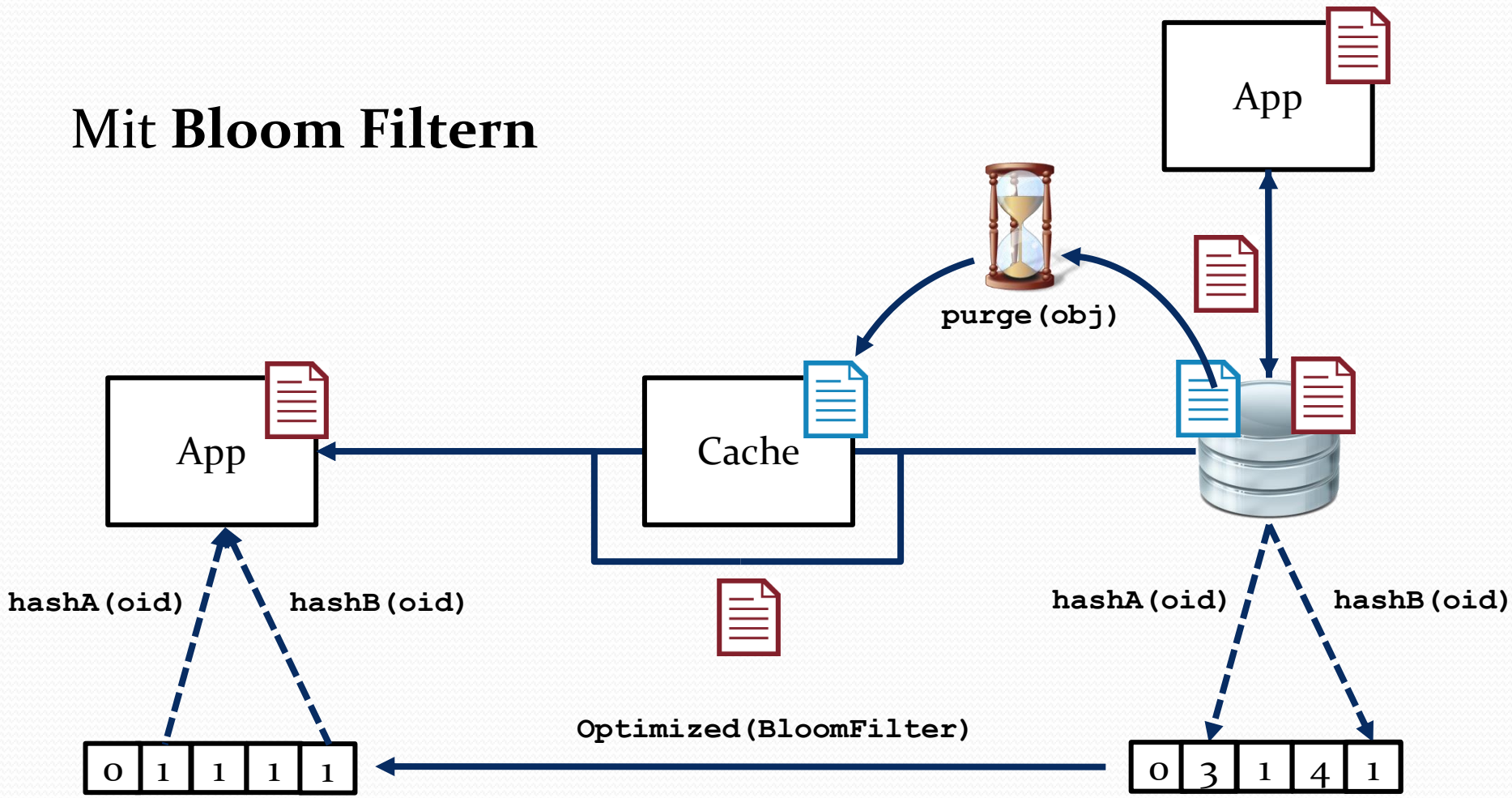
Optimistische Transaktionen:

- Kein Locking

Lost Update ✗ Dirty Read ✗ Nonrepeatable Reads ✗ Phantome ✗

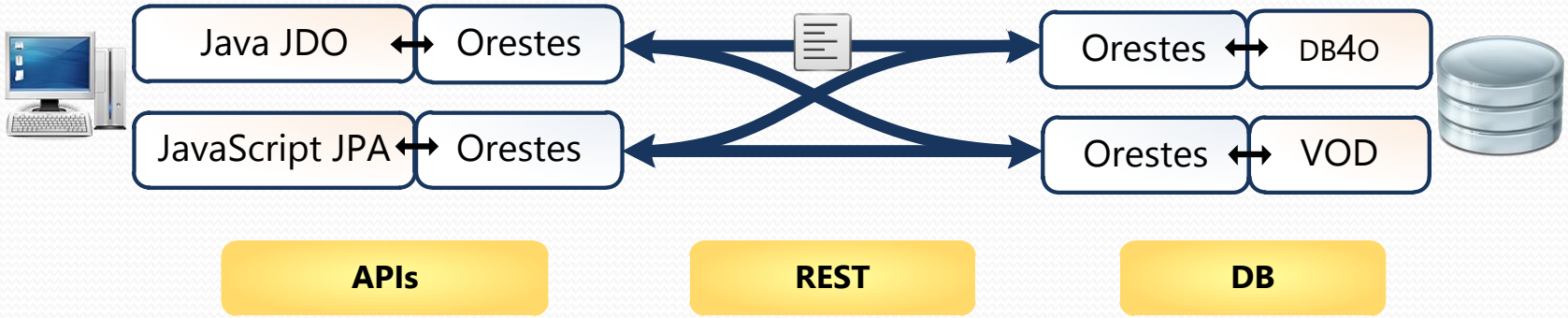
Immer aktuell

Mit **Bloom Filtern**



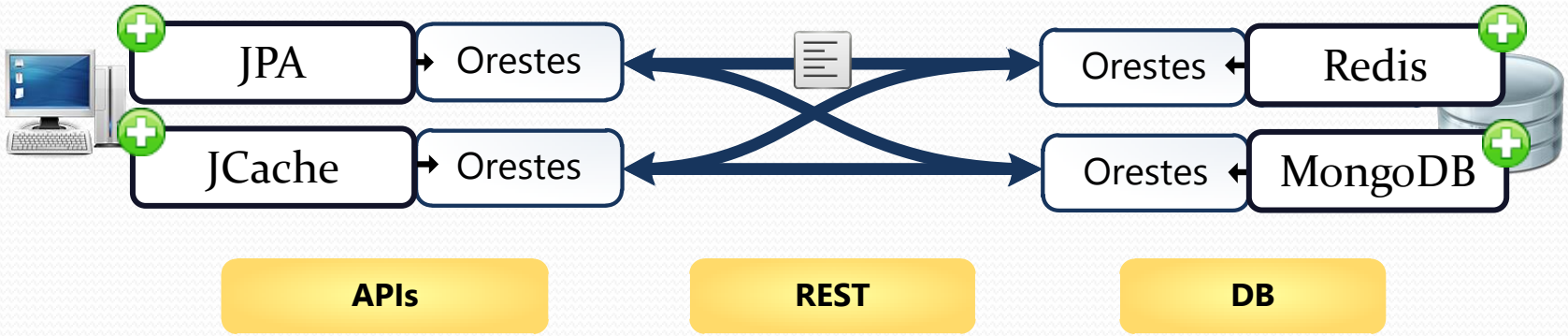
Implementierung

Bisher:



Implementierung

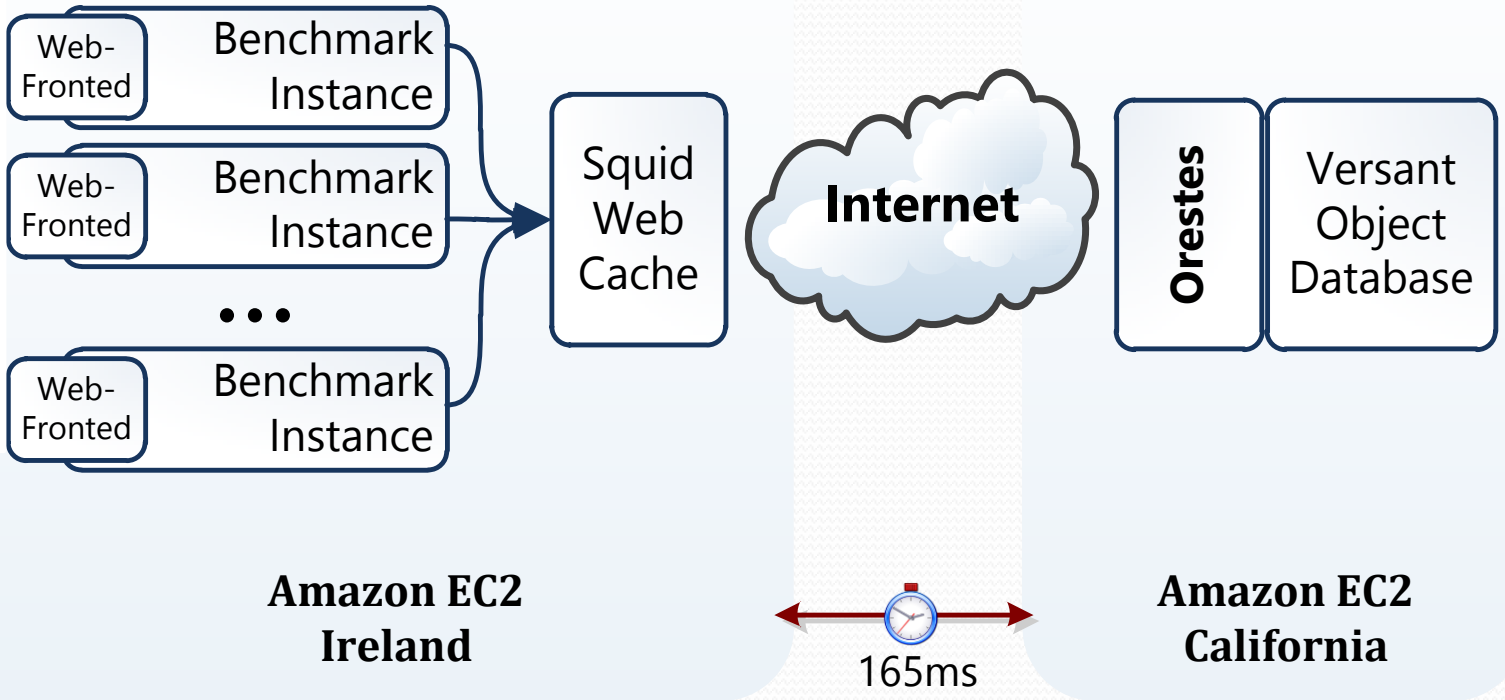
In Arbeit:



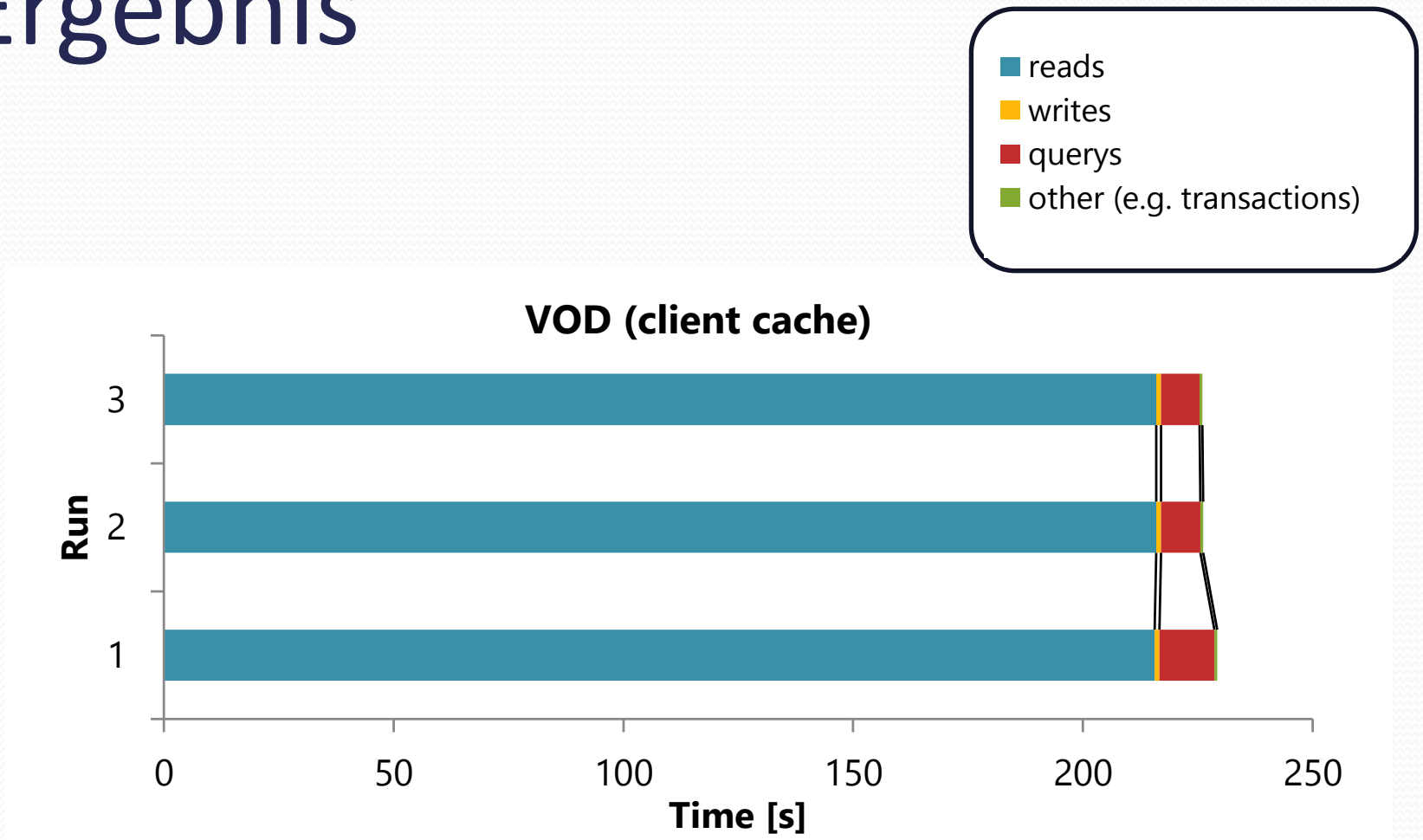
Und Open-Source sobald wie möglich.

Benchmark

API: **JDO**, Modell: **Social Network**, **TCP** vs **Orestes**

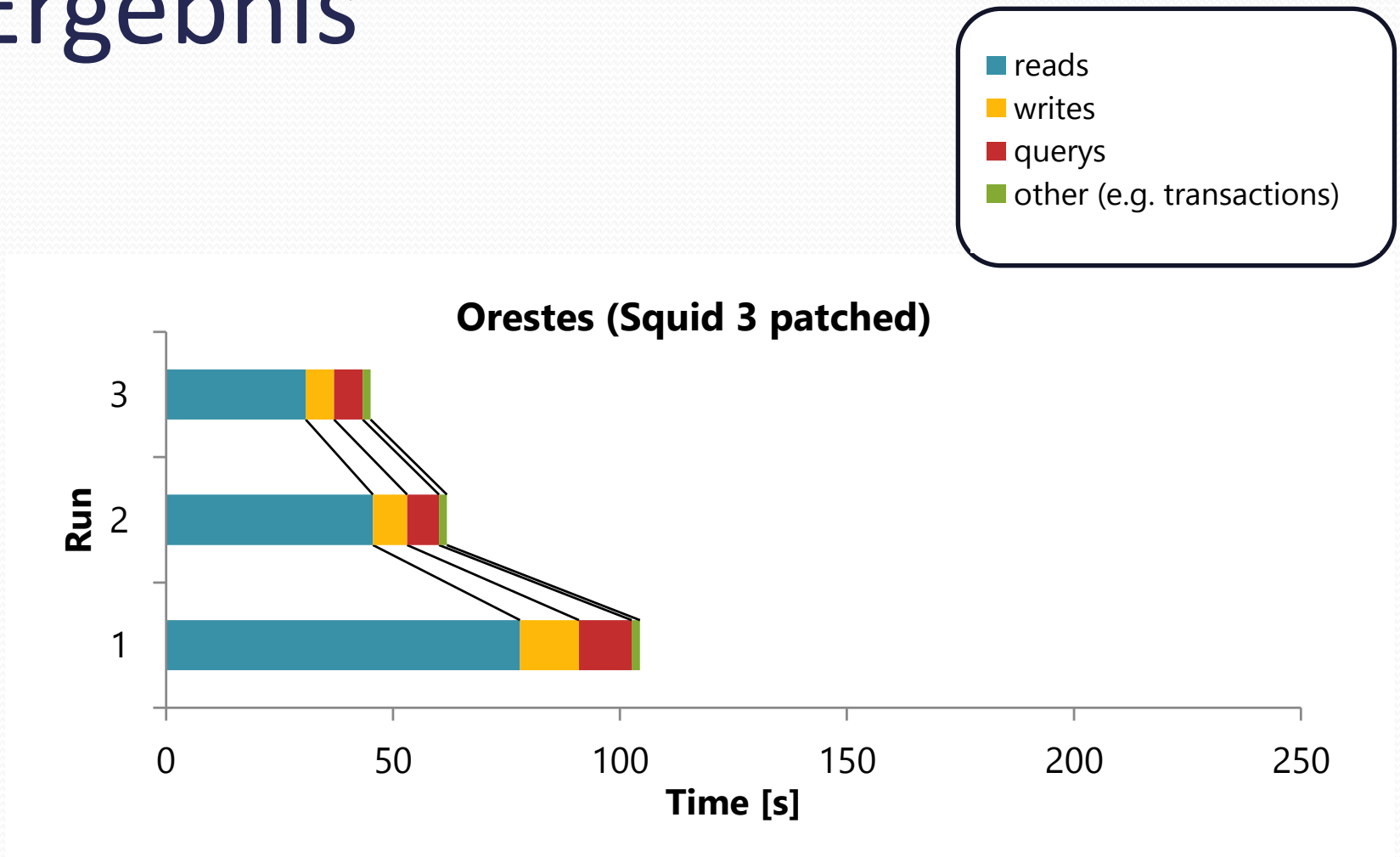


Ergebnis



30.000 DB Objekte, **300** Ops, Lese/Schreib Verhältnis **10/1**

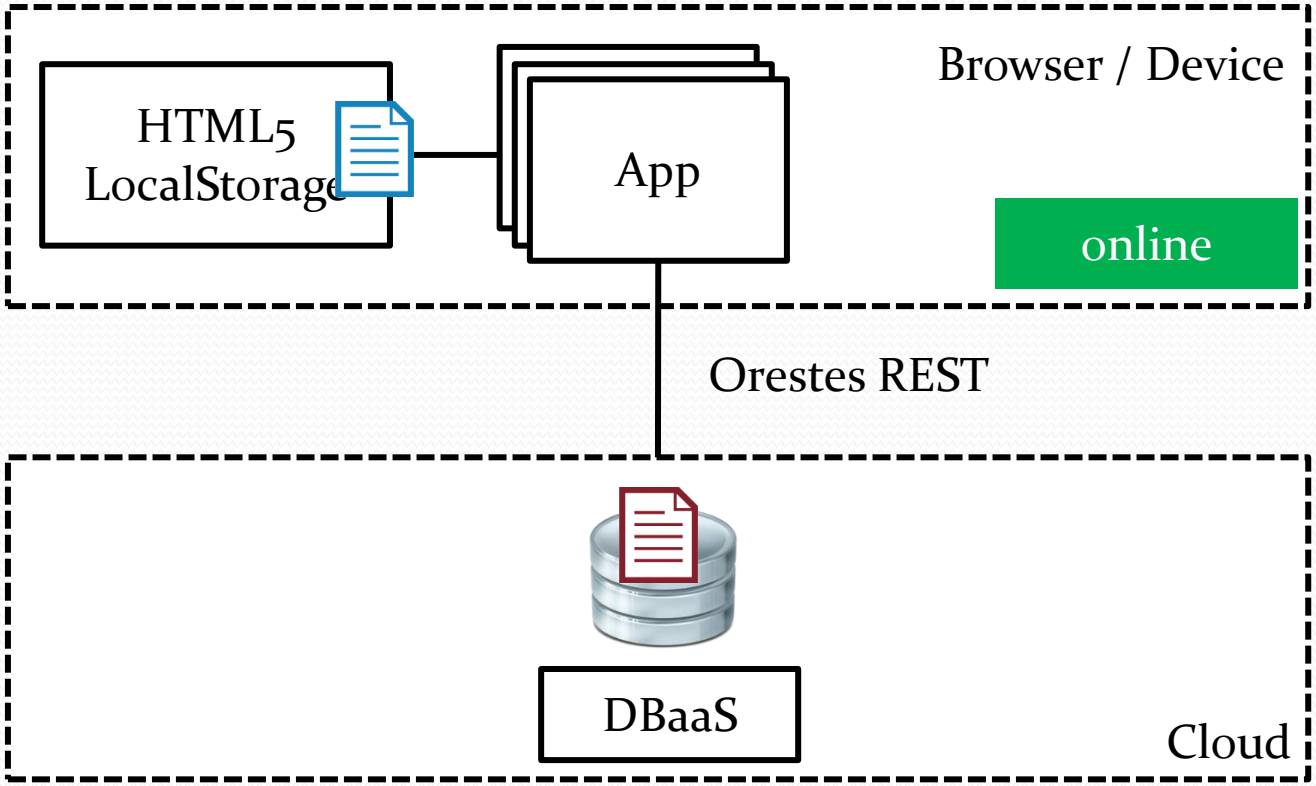
Ergebnis



30.000 DB Objekte, **300** Ops, Lese/Schreib Verhältnis **10/1**

Future Work

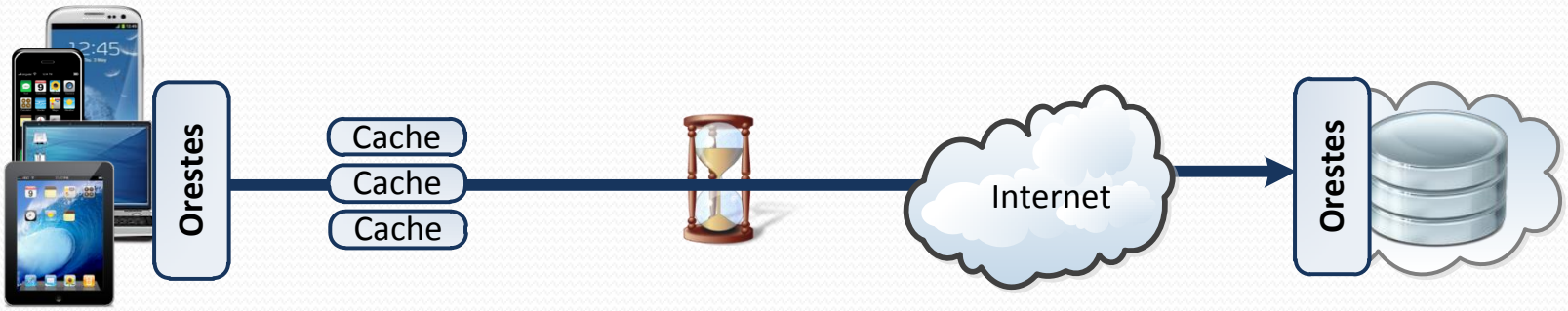
„Clientseitige Persistenz“



Zusammenfassung

DBaaS hat 2 besonders große Probleme:

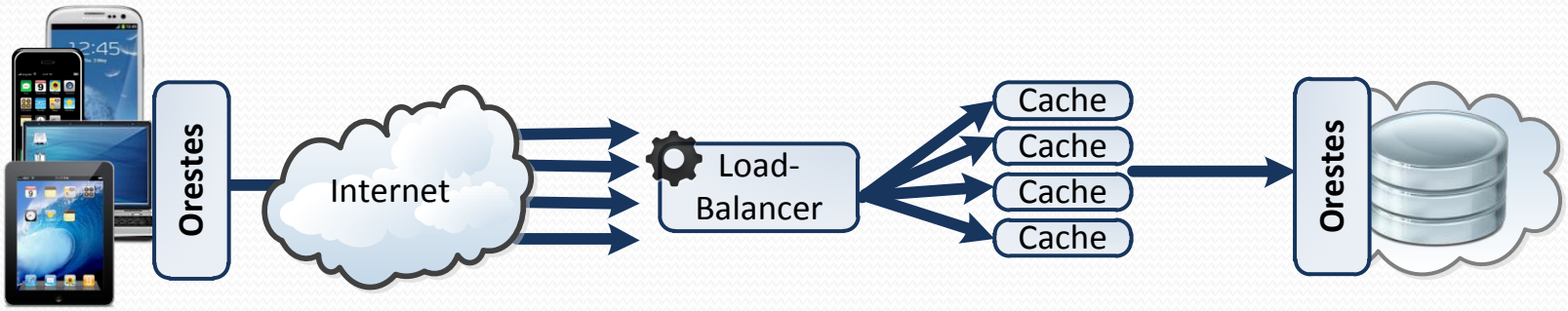
Netzwerklatenz



Zusammenfassung

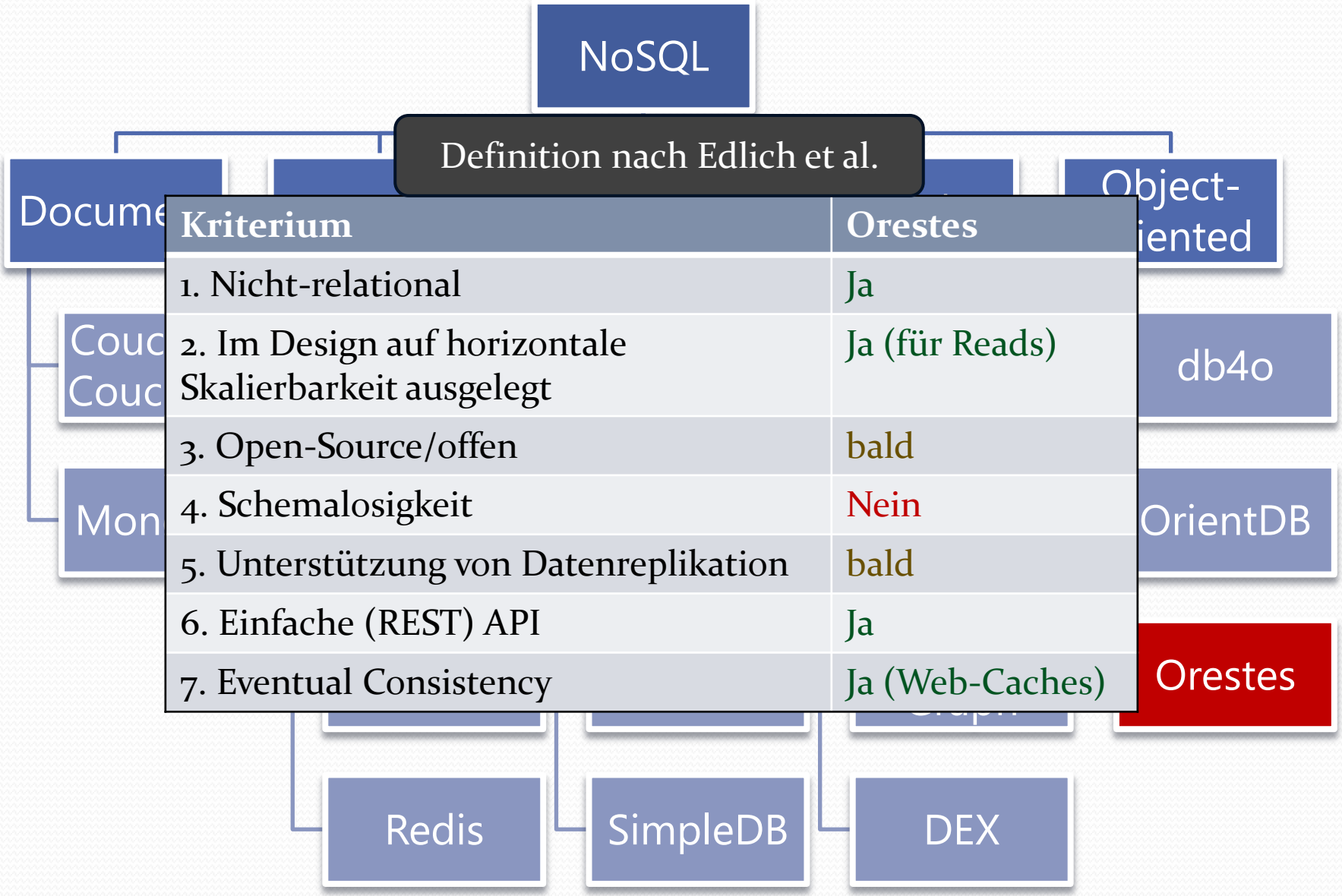
DBaaS hat 2 besonders große Probleme:

Elastische Skalierung



Live-Demo

try.orestes.info


NoSQL
Definition nach Edlich et al.
Kriterium
Orestes
1. Nicht-relational
Ja
2. Im Design auf horizontale Skalierbarkeit ausgelegt
Ja (für Reads)
3. Open-Source/offen
bald
4. Schemalosigkeit
Nein
5. Unterstützung von Datenreplikation
bald
6. Einfache (REST) API
Ja
7. Eventual Consistency
Ja (Web-Caches)
db4o
OrientDB
Orestes
Redis
SimpleDB
DEX