Cloud Databases

in Research and Practice

Felix Gessert

(29.04.2014)

Slides: baqend.com/nosql.pdf

About me

- PhD student (database group university of hamburg)

About me

- PhD student (database group university of hamburg)

About me

- PhD student (database group university of hamburg)

Outline

What are Cloud Databases?

- Categories of Cloud Databases
- Properties

Cloud Databases in the wild

Research Perspectives

Wrap-up and literature

2003: GFS (Google)

2006: BigTable (Google)

2007: Dynamo (Amazon)

2007: S3 (Amazon)

2008: SimpleDB (Amazon)

2010: SQL Azure

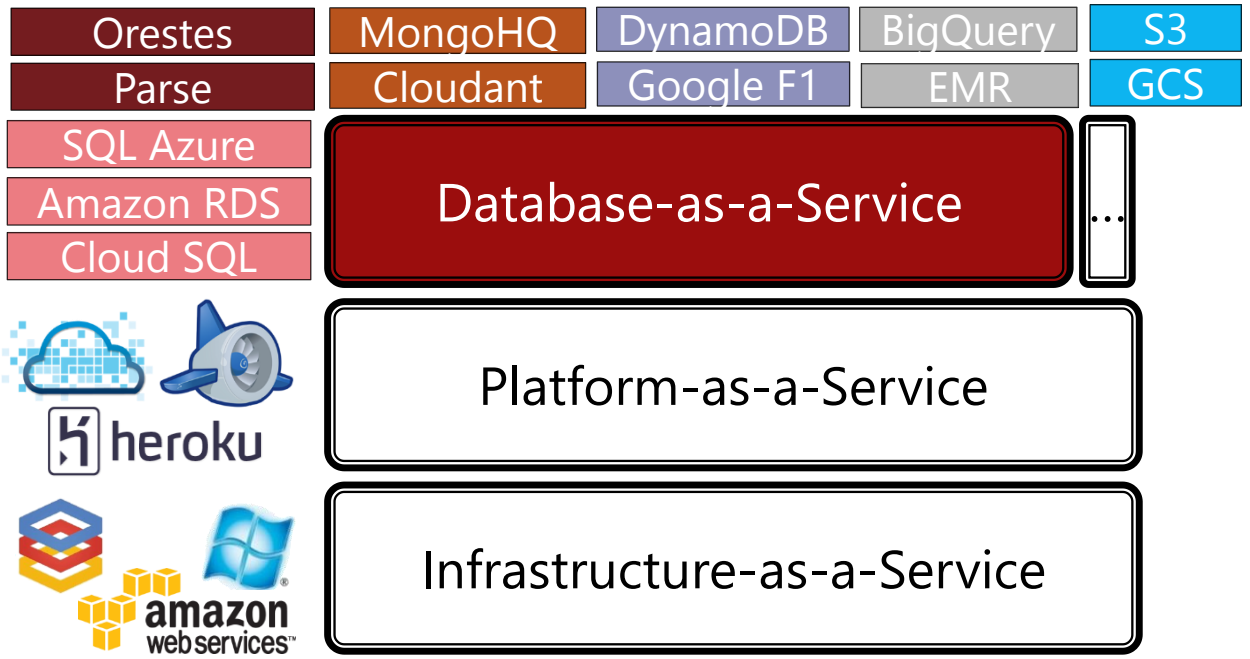
2011: Parse

2012: DynamoDB (Amazon)

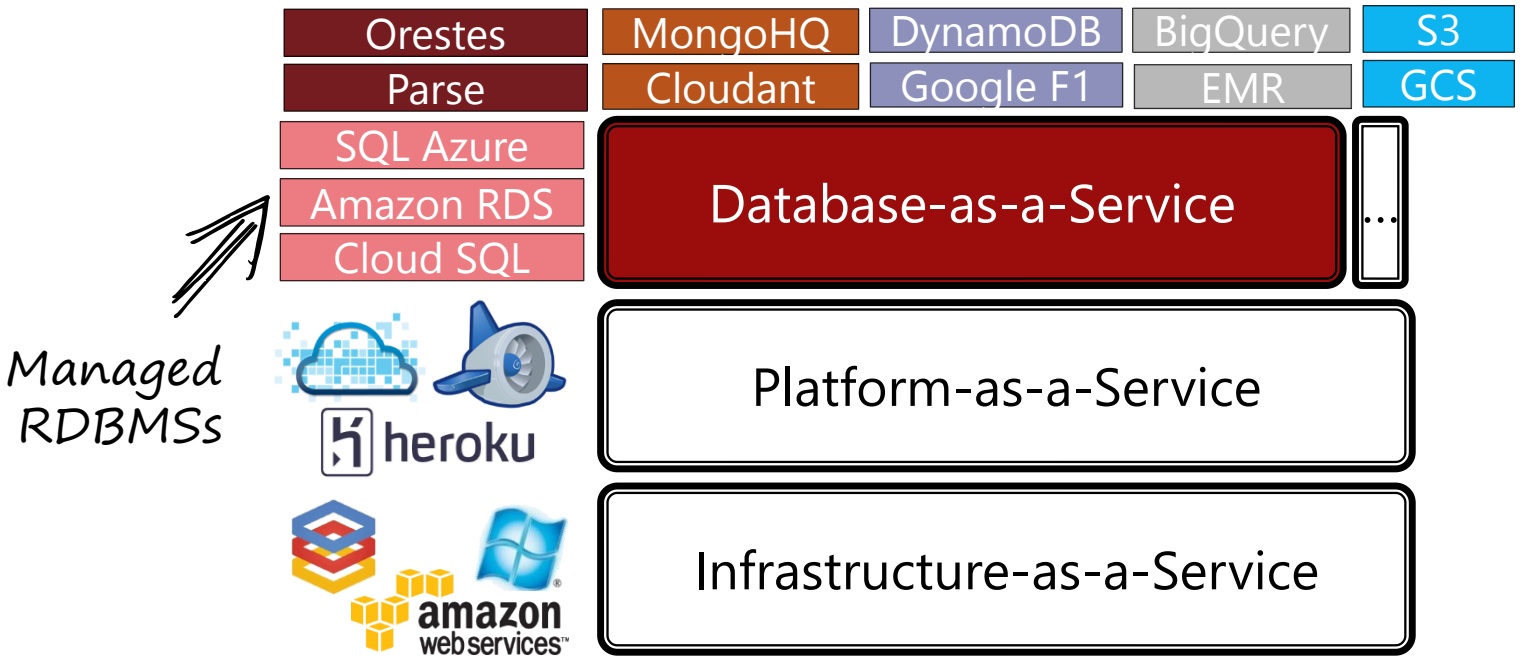
2013: Redshift (Amazon)


Context: What kinds of Cloud databases are there?

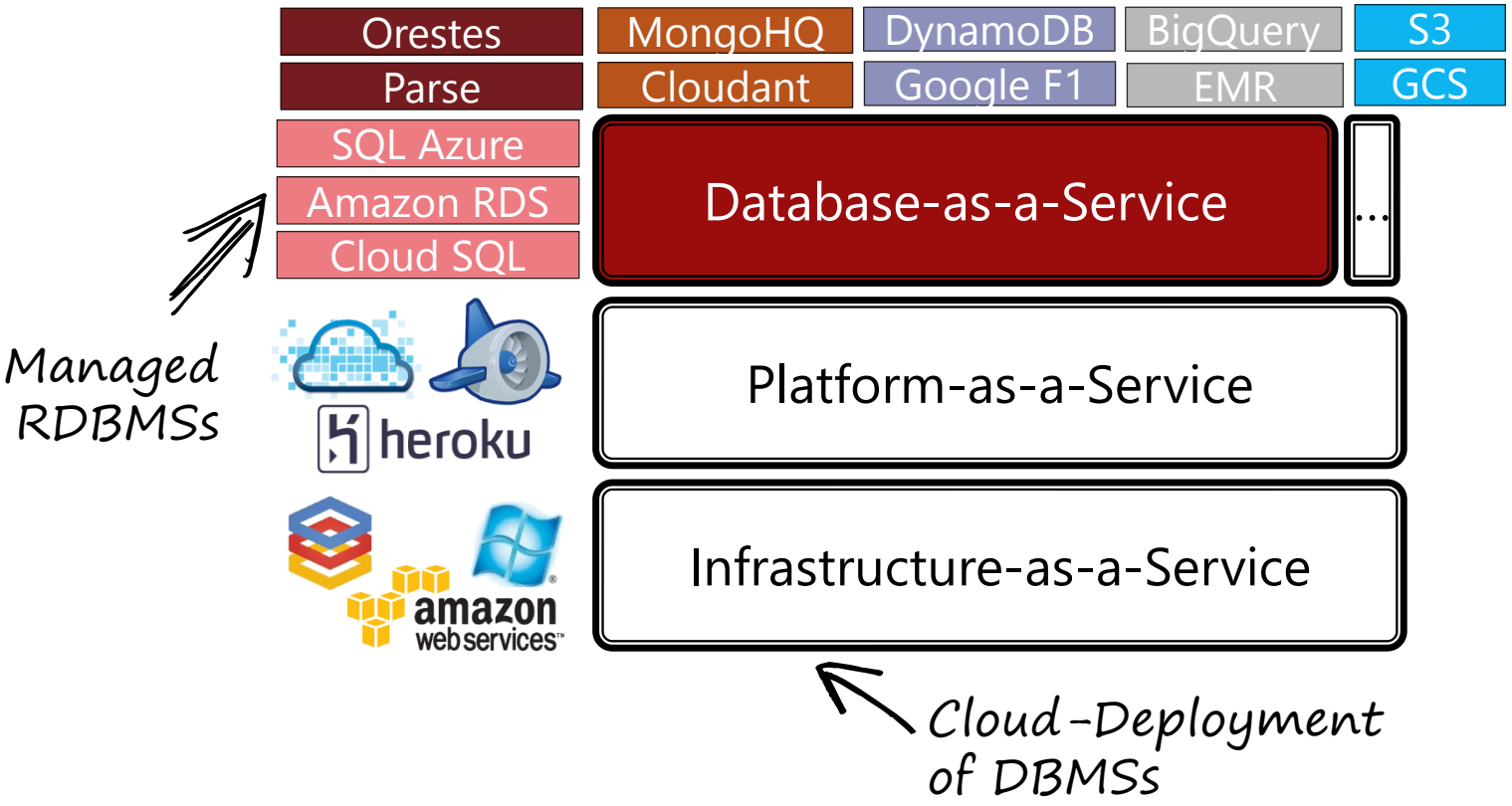
DBaaS


Orestes
Parse
SQL Azure
Amazon RDS
Cloud SQL
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
Database-as-a-Service
...
heroku
Platform-as-a-Service
amazon
web services
Infrastructure-as-a-Service

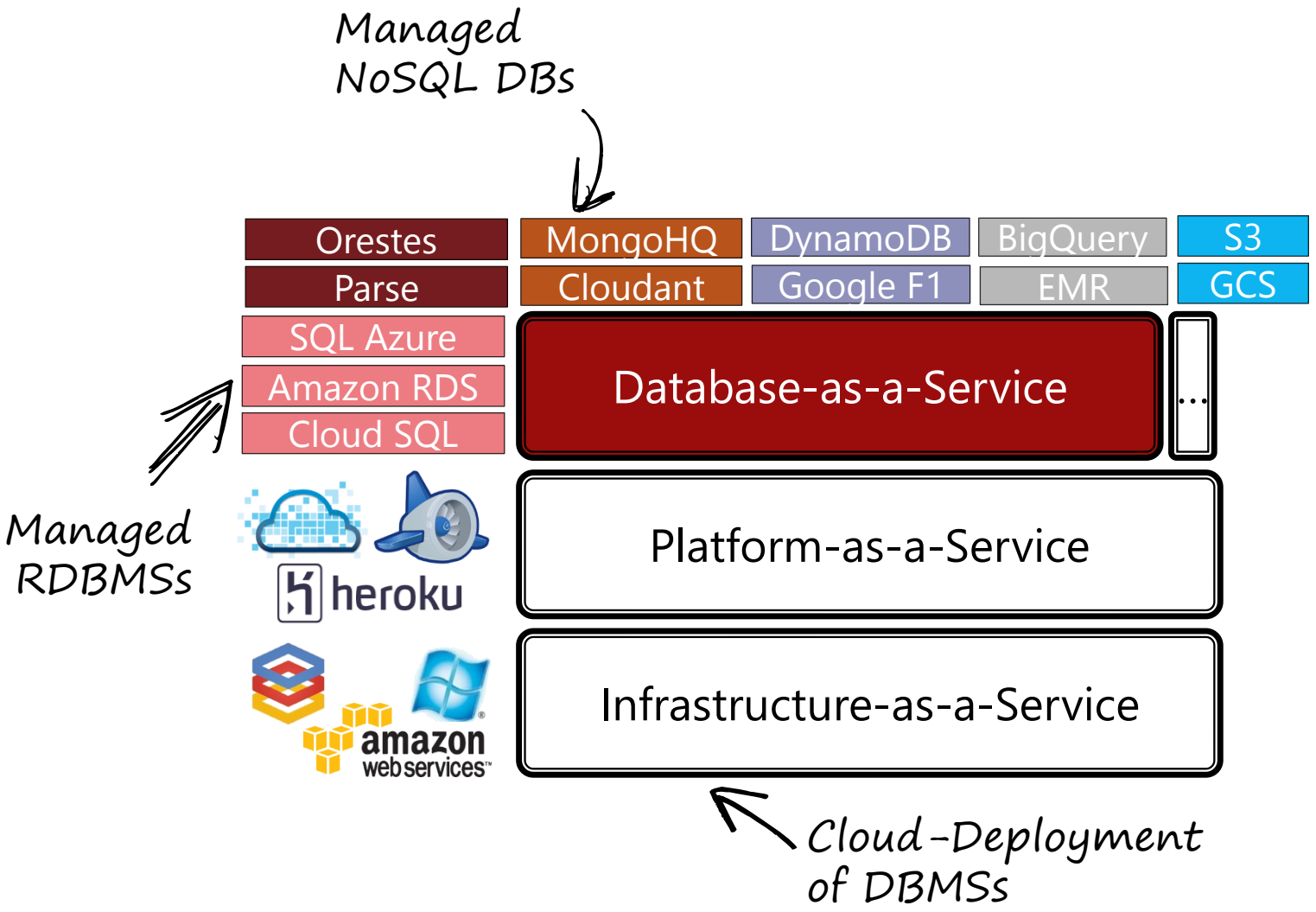
DBaaS


Orestes
Parse
SQL Azure
Amazon RDS
Cloud SQL
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
Database-as-a-Service
...
Managed RDBMSs
heroku
amazon web services
Platform-as-a-Service
Infrastructure-as-a-Service

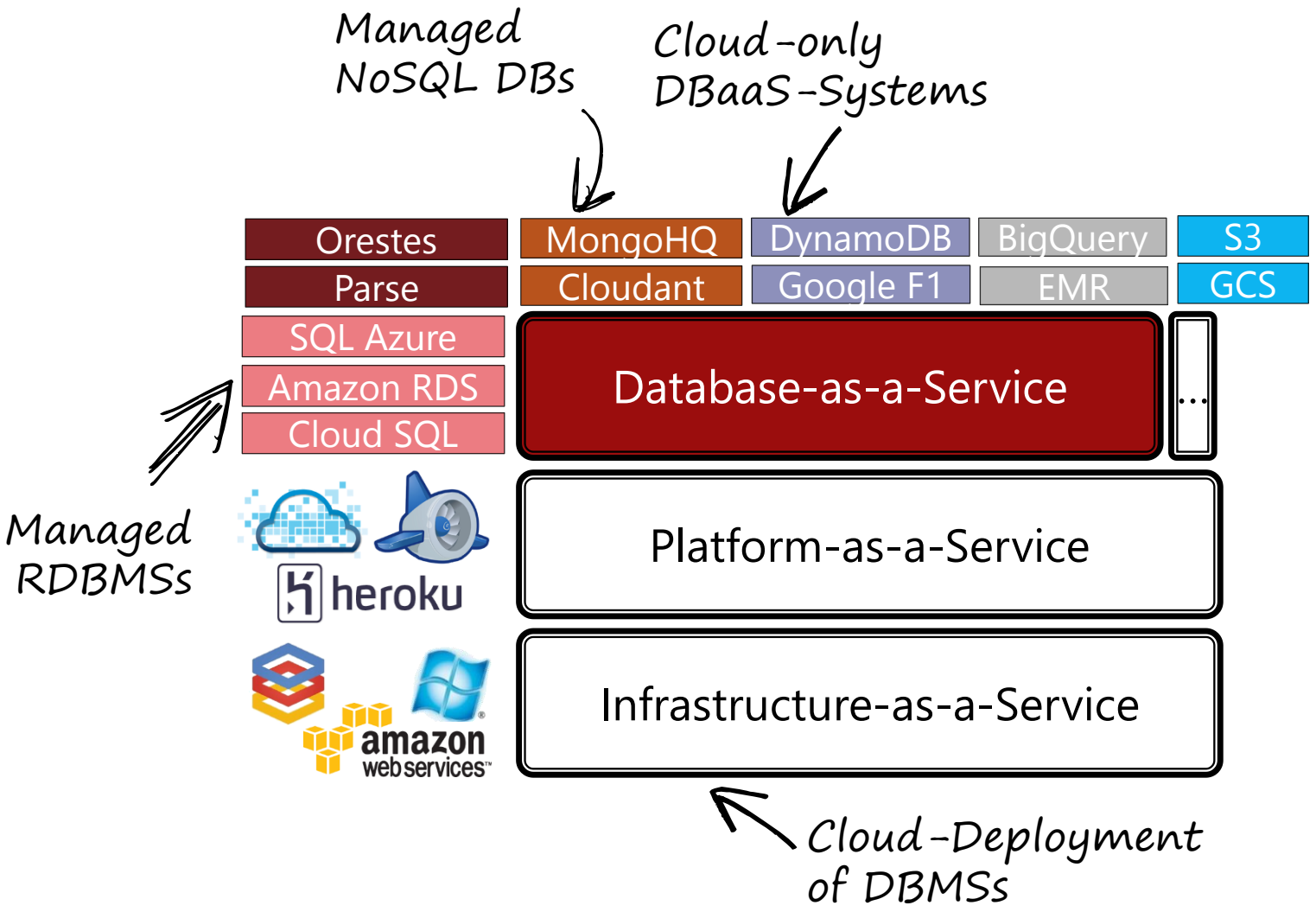
DBaaS


Orestes
Parse
SQL Azure
Amazon RDS
Cloud SQL
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
Database-as-a-Service
...
Platform-as-a-Service
Infrastructure-as-a-Service
heroku
amazon web services
Managed RDBMSs
Cloud-Deployment of DBMSs

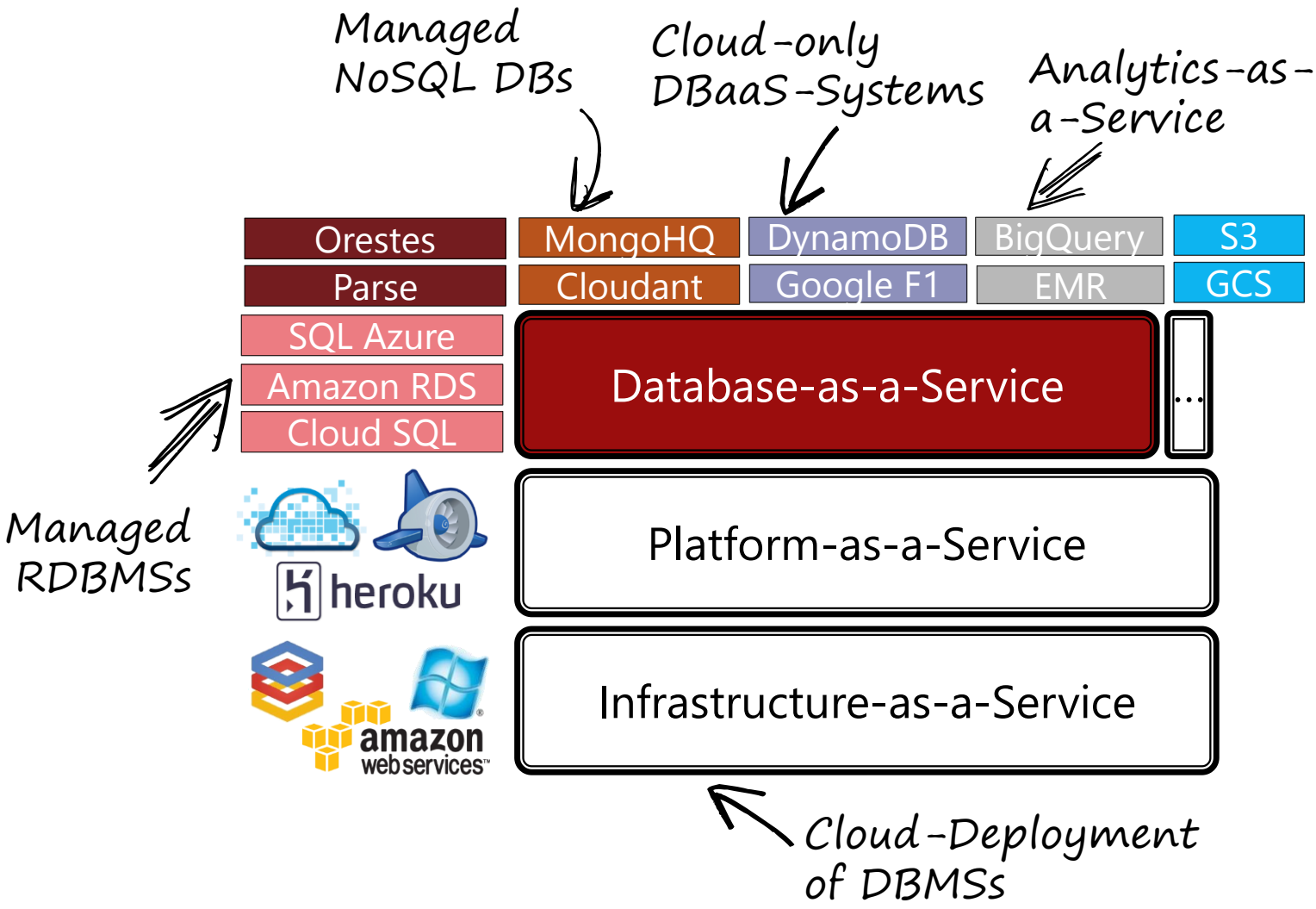
DBaaS


Managed
NoSQL DBs
Orestes
Parse
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
SQL Azure
Amazon RDS
Cloud SQL
Database-as-a-Service
...
Managed
RDBMSs
heroku
Platform-as-a-Service
amazon
web services
Infrastructure-as-a-Service
Cloud-Deployment
of DBMSs

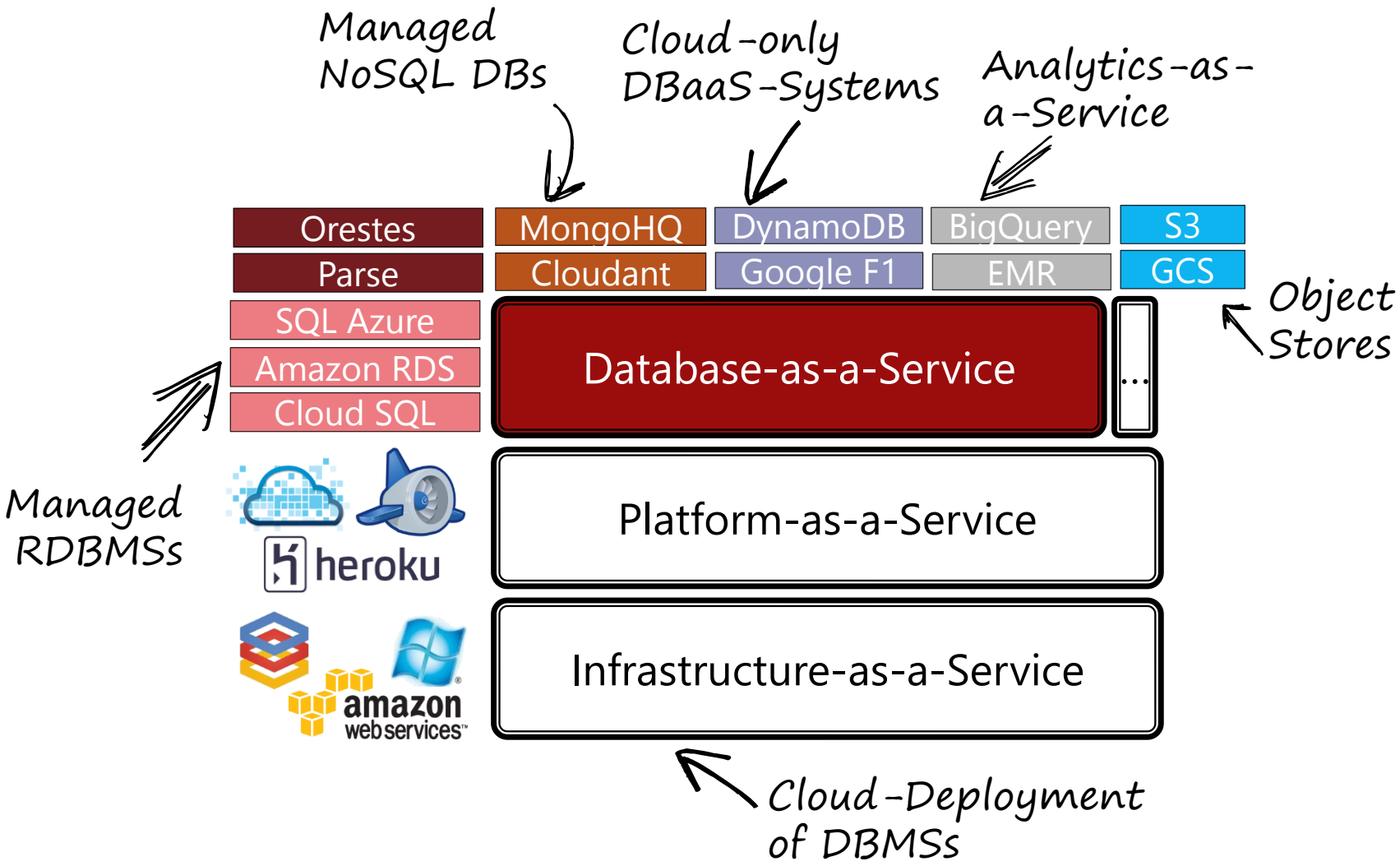
DBaaS


Managed NoSQL DBs
Cloud-only DBaaS-Systems
Orestes
Parse
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
SQL Azure
Amazon RDS
Cloud SQL
Database-as-a-Service
...
Managed RDBMSs
heroku
Platform-as-a-Service
amazon web services
Infrastructure-as-a-Service
Cloud-Deployment of DBMSs

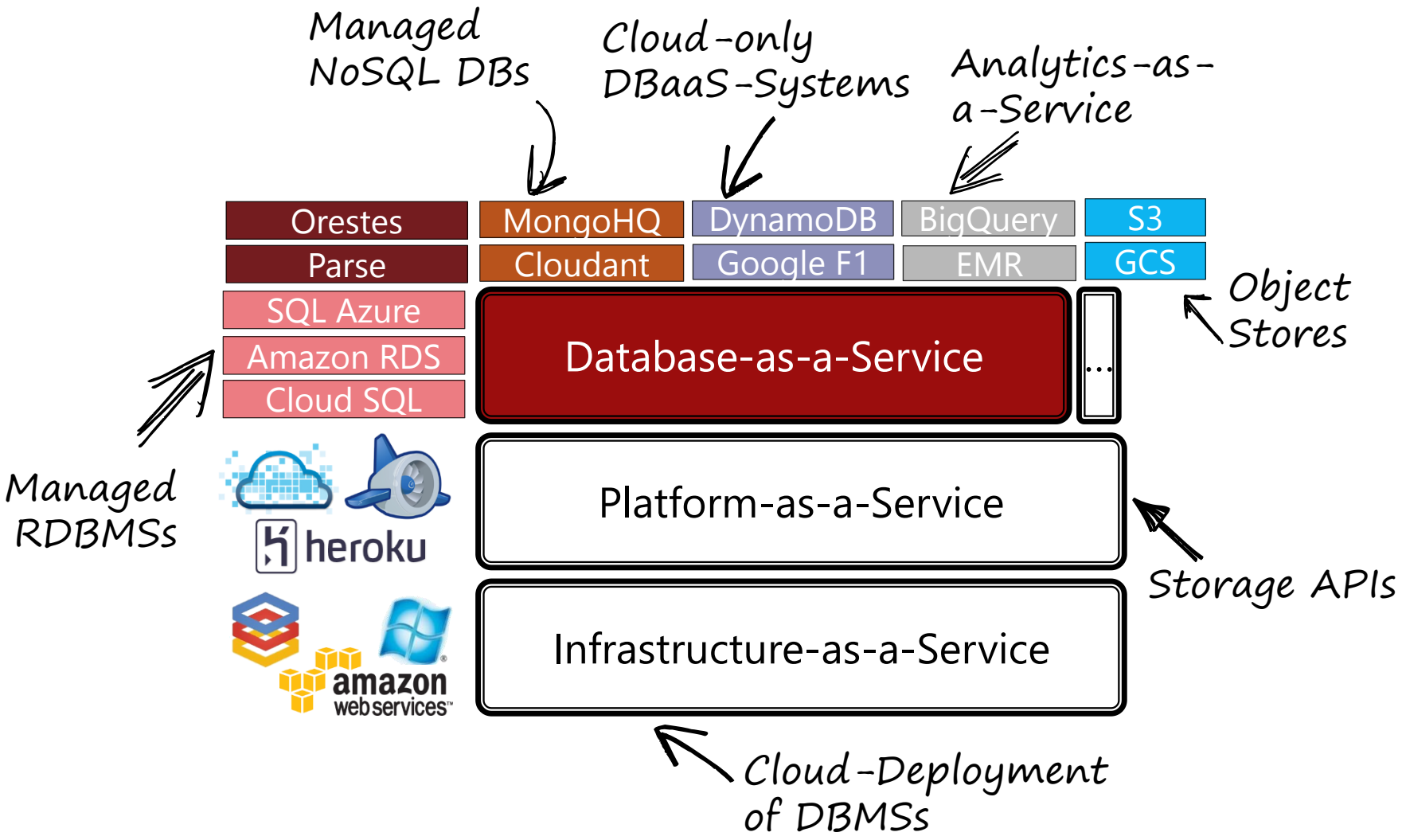
DBaaS


Managed NoSQL DBs
Cloud-only DBaaS-Systems
Analytics-as-a-Service
Orestes
Parse
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
SQL Azure
Amazon RDS
Cloud SQL
Database-as-a-Service
...
Managed RDBMSs
heroku
Platform-as-a-Service
amazon web services
Infrastructure-as-a-Service
Cloud-Deployment of DBMSs

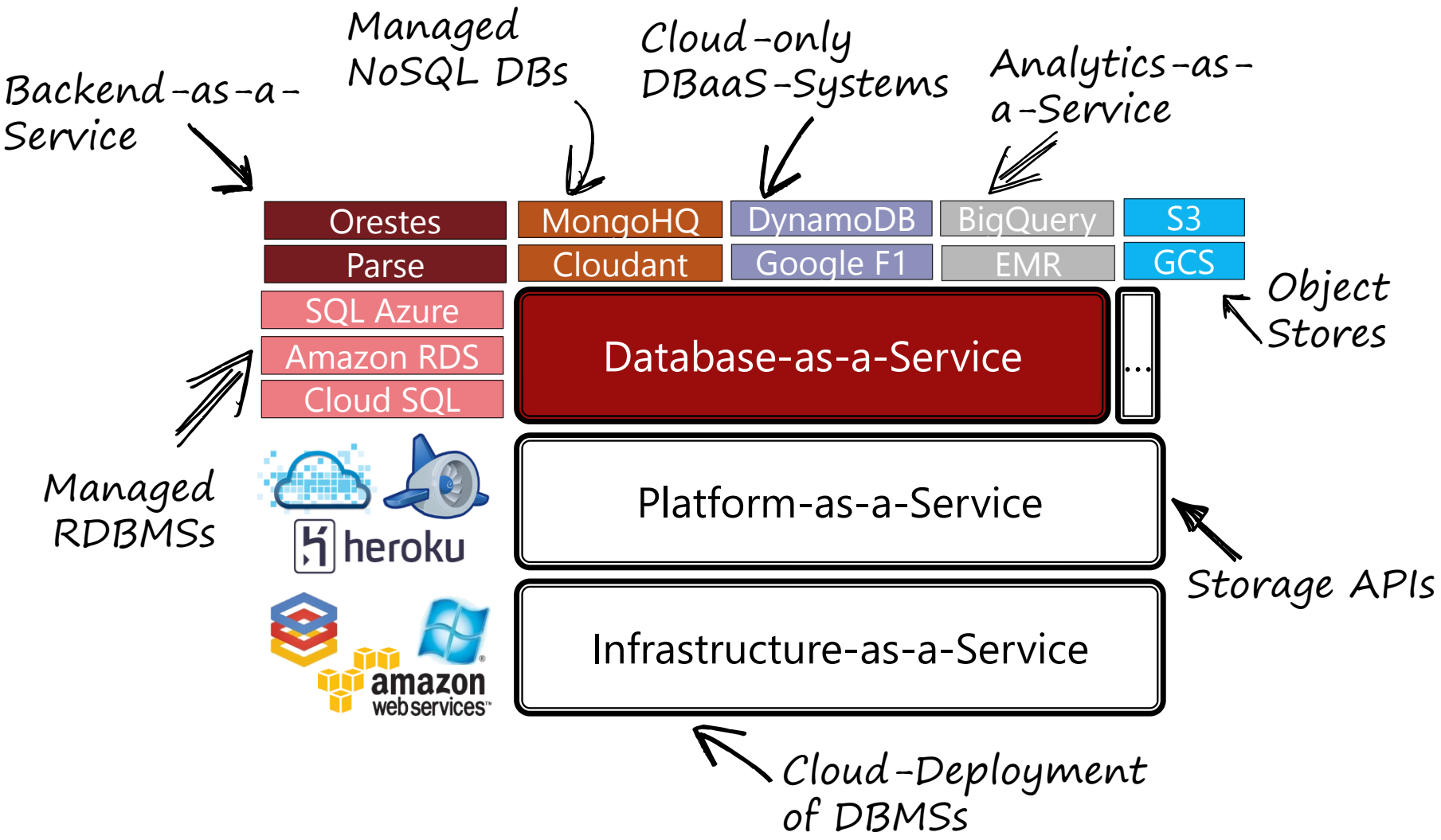
DBaaS


Managed NoSQL DBs
Cloud-only DBaaS-Systems
Analytics-as-a-Service
Orestes
Parse
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
Object Stores
SQL Azure
Amazon RDS
Cloud SQL
Database-as-a-Service
...
Managed RDBMSs
heroku
Platform-as-a-Service
amazon web services
Infrastructure-as-a-Service
Cloud-Deployment of DBMSs

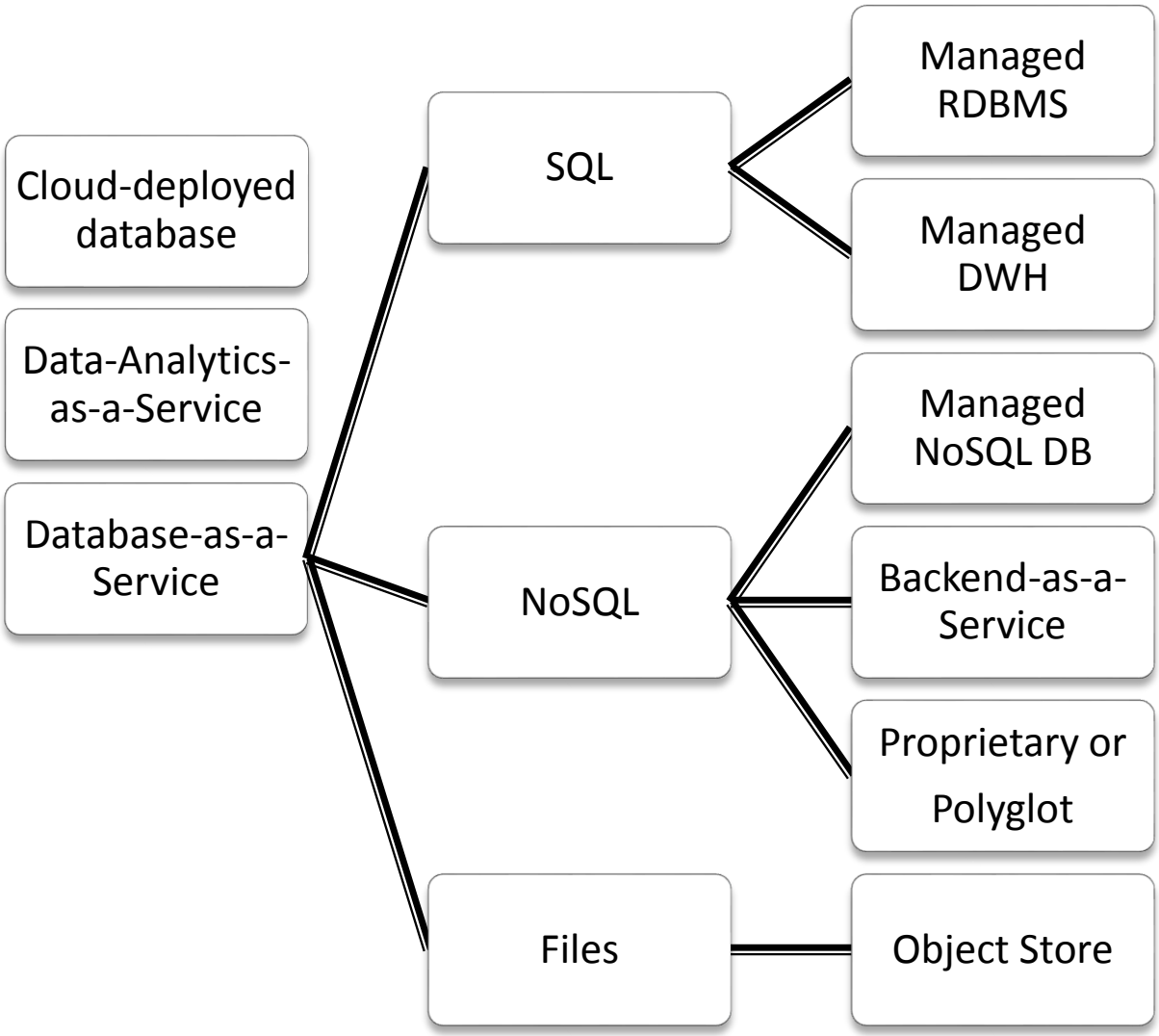
DBaaS


Managed NoSQL DBs
Cloud-only DBaaS-Systems
Analytics-as-a-Service
Orestes
Parse
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
Object Stores
SQL Azure
Amazon RDS
Cloud SQL
Database-as-a-Service
...
Managed RDBMSs
heroku
Platform-as-a-Service
Storage APIs
amazon web services
Infrastructure-as-a-Service
Cloud-Deployment of DBMSs

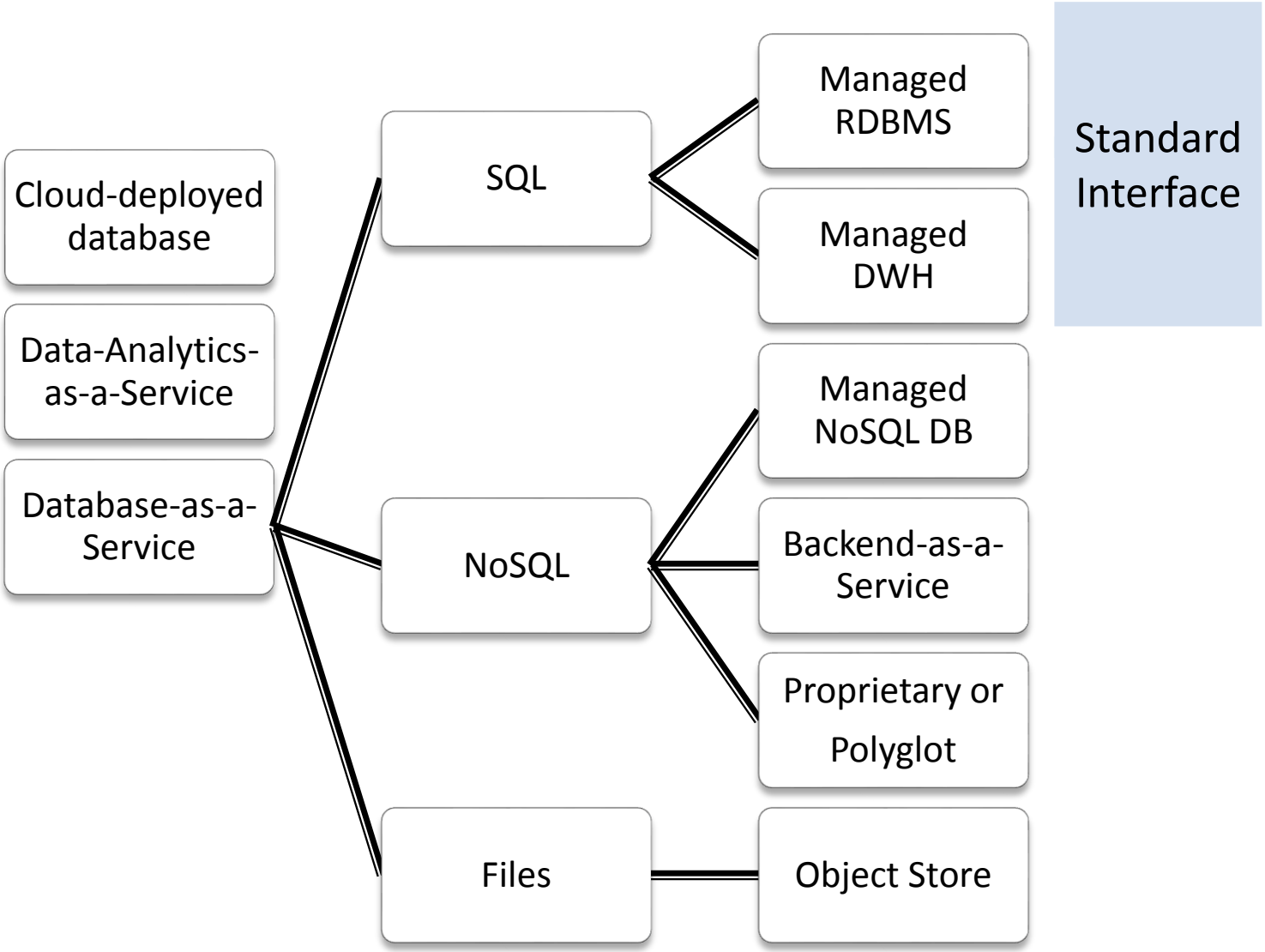
DBaaS


Backend-as-a-Service
Managed NoSQL DBs
Cloud-only DBaaS-Systems
Analytics-as-a-Service
Orestes
Parse
SQL Azure
Amazon RDS
Cloud SQL
MongoHQ
Cloudant
DynamoDB
Google F1
BigQuery
EMR
S3
GCS
Object Stores
Database-as-a-Service
...
Platform-as-a-Service
Infrastructure-as-a-Service
Managed RDBMSs
heroku
amazon web services
Storage APIs
Cloud-Deployment of DBMSs

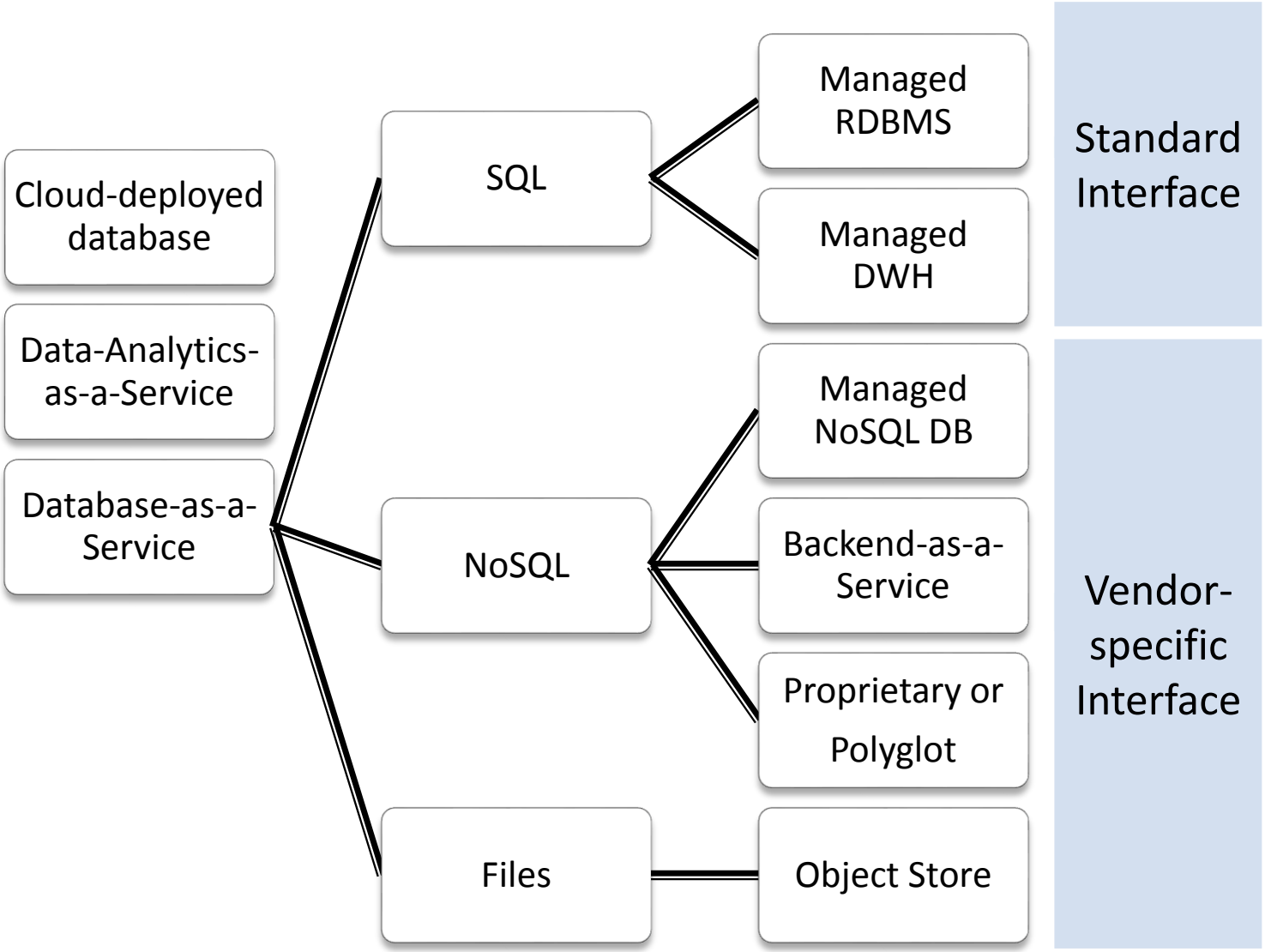
Cloud Databases



Cloud Databases

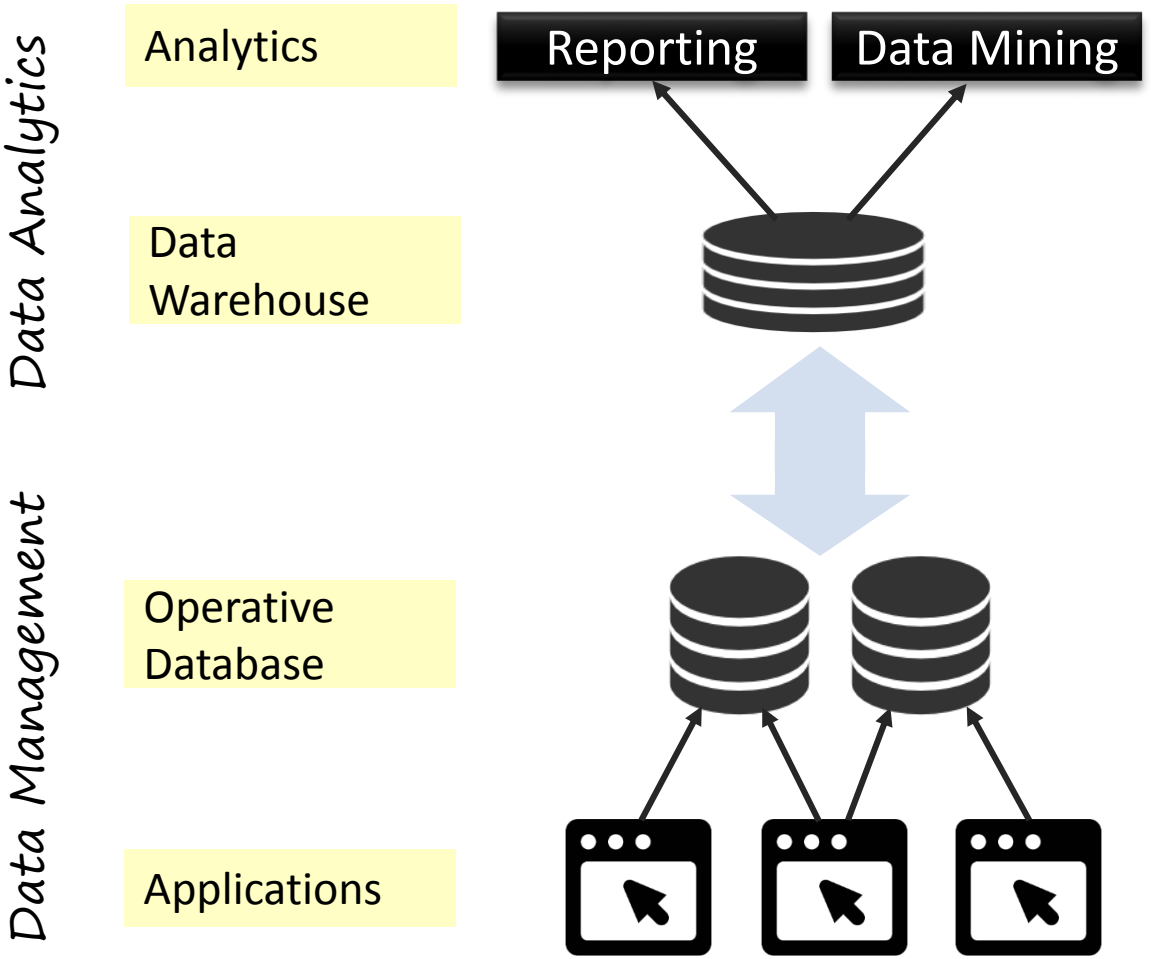


Cloud Databases



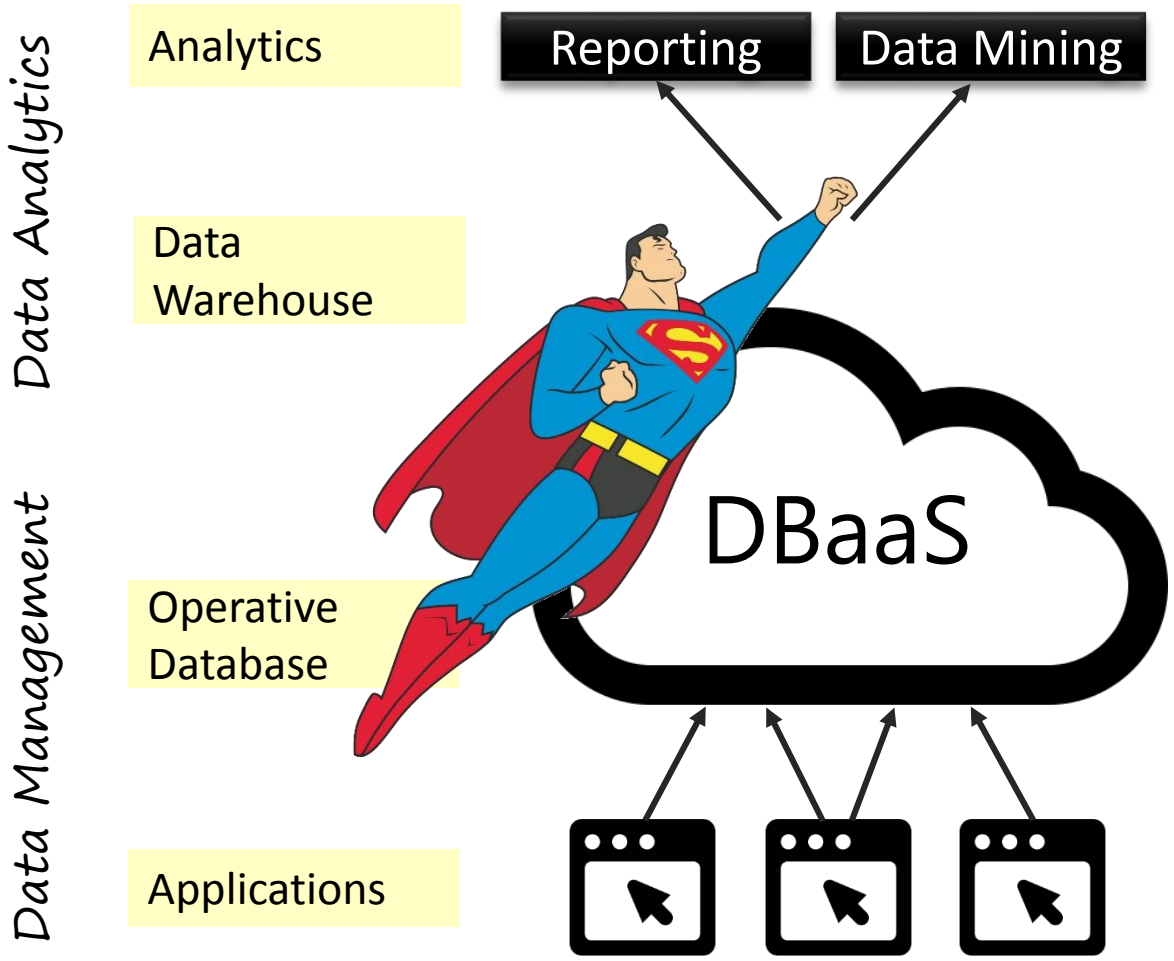
Architectures

Application Architecture <-> Cloud Database Category



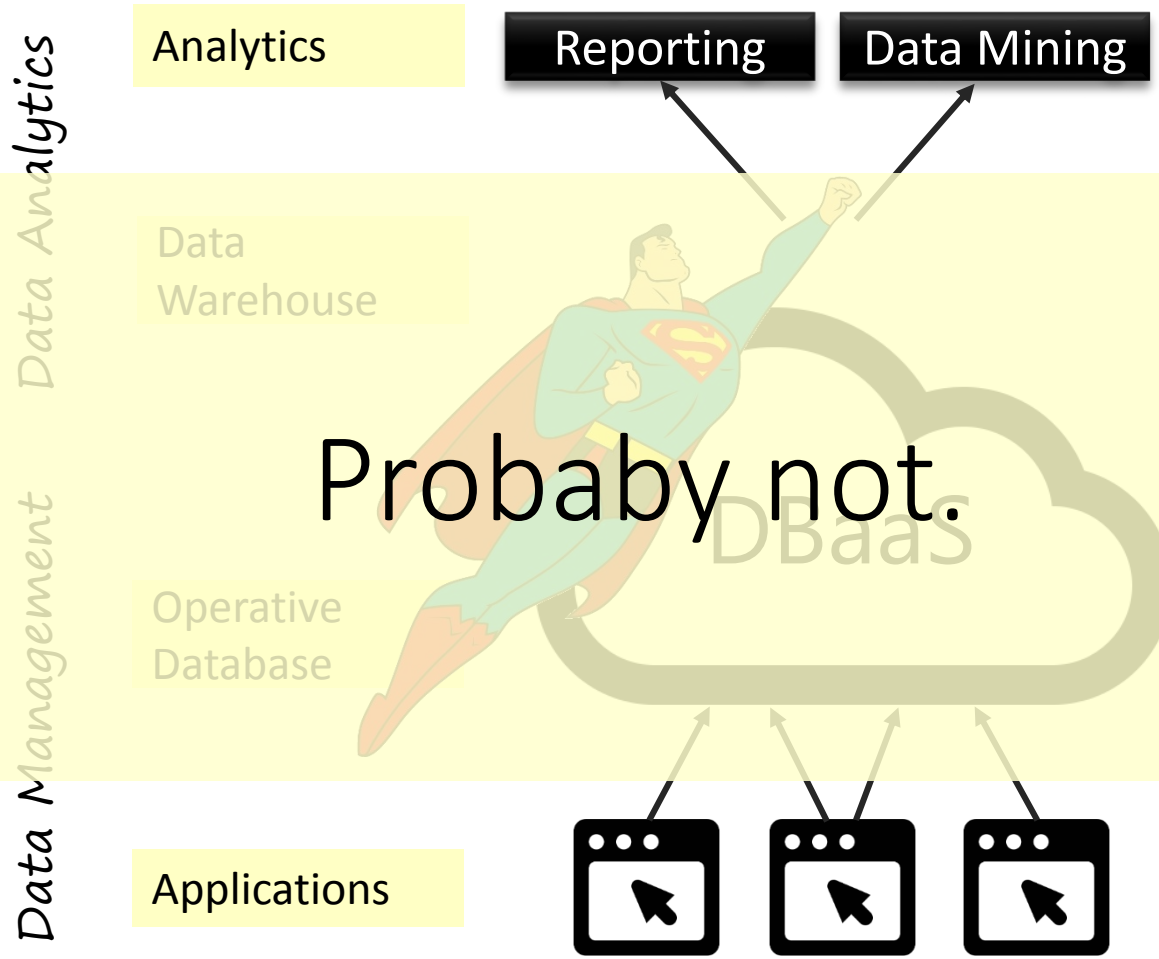
Architectures

Application Architecture <-> Cloud Database Category

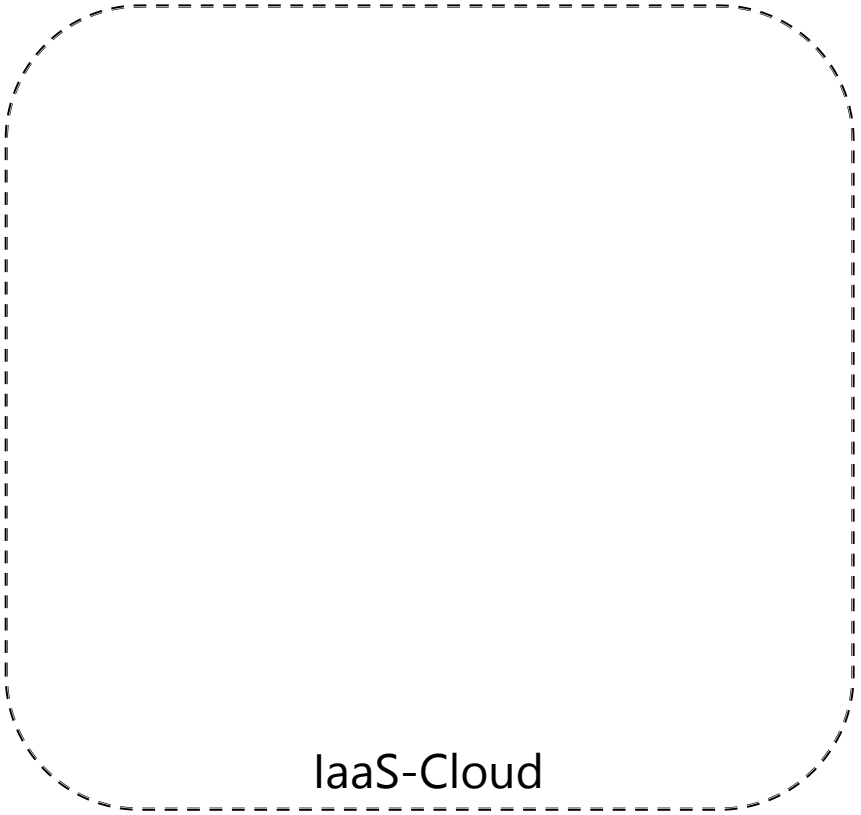


Architectures

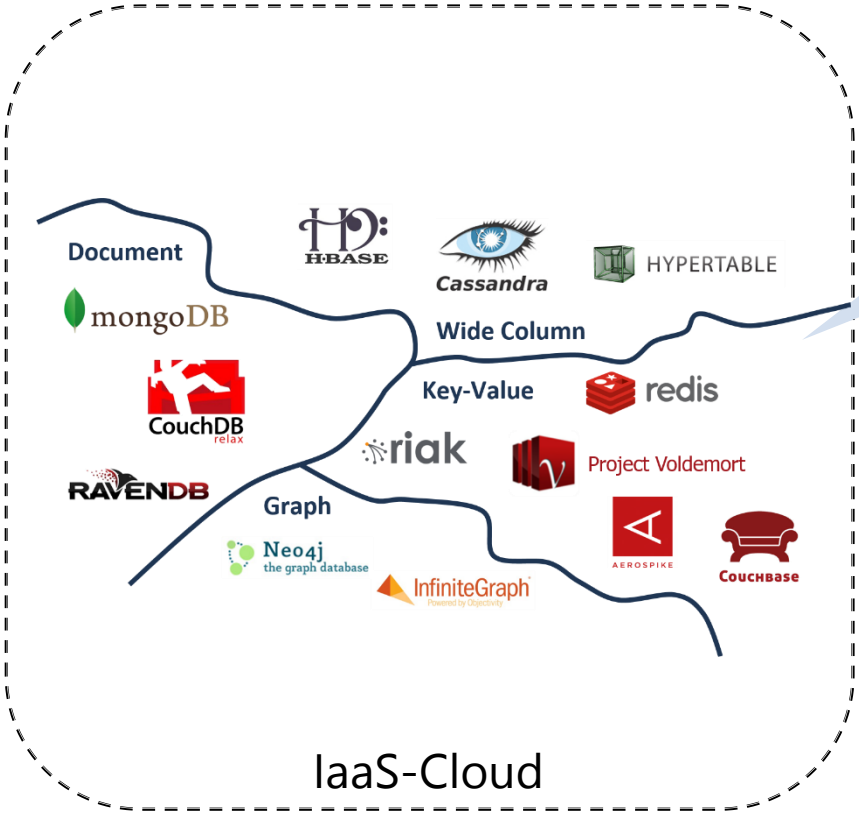
Application Architecture <-> Cloud Database Category



Cloud-Deployed Database

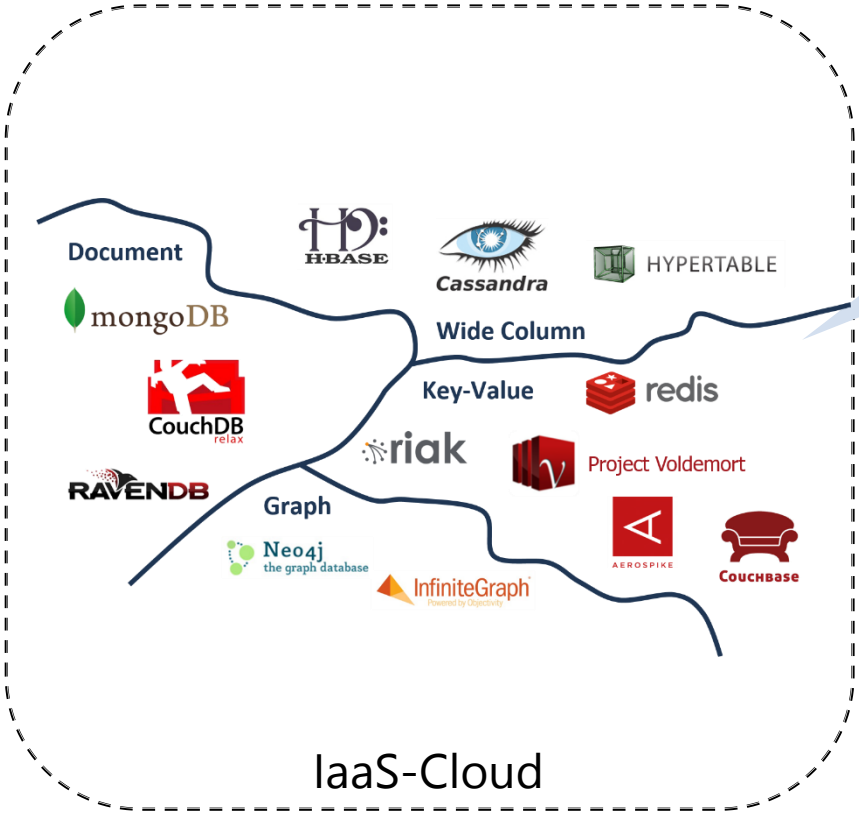


Cloud-Deployed Database



Cloud-deploy your favourite database system

Cloud-Deployed Database

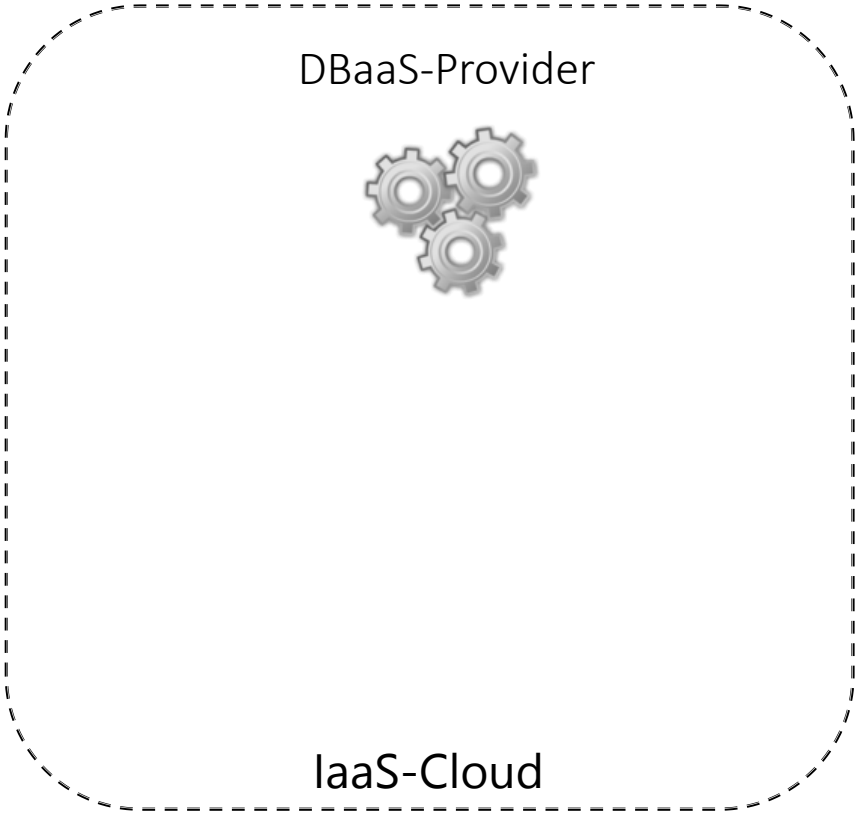


Cloud-deploy your favourite database system

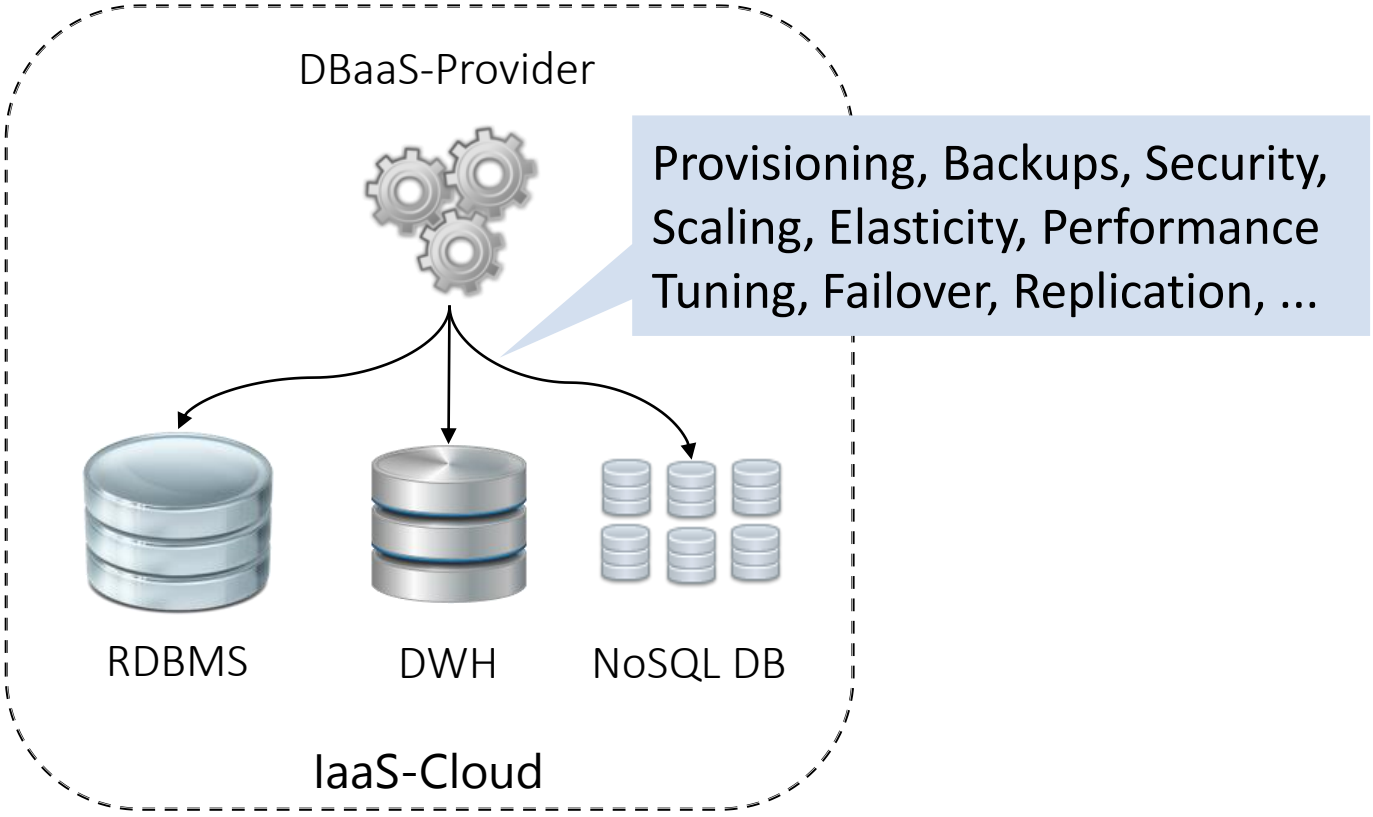
Does not solve:

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

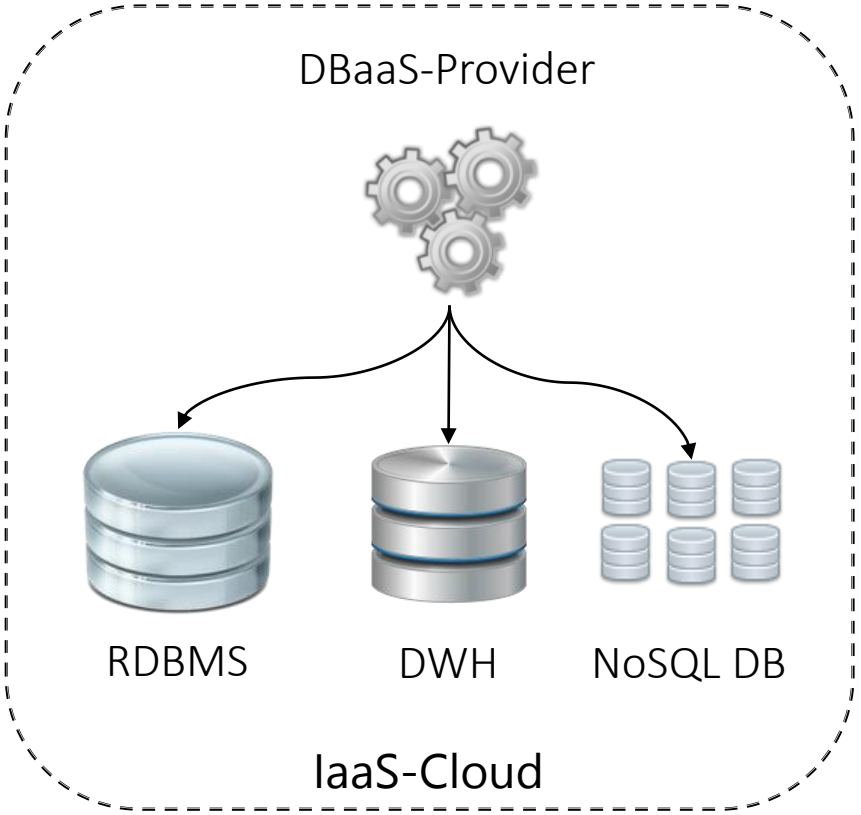
Managed RDBMS/DWH/NoSQL DB



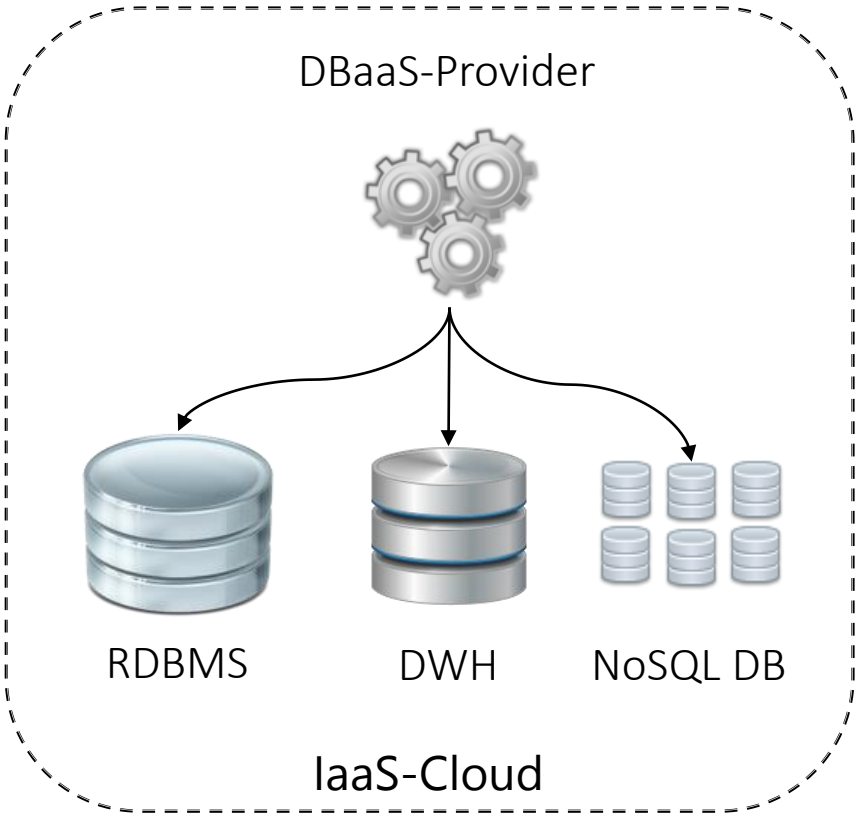
Managed RDBMS/DWH/NoSQL DB

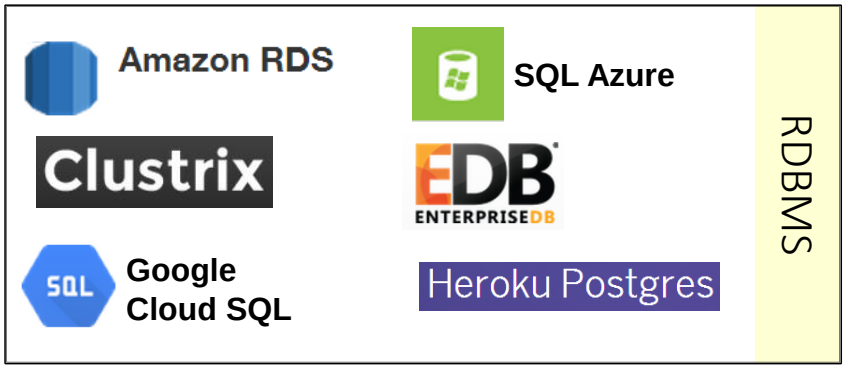


Managed RDBMS/DWH/NoSQL DB

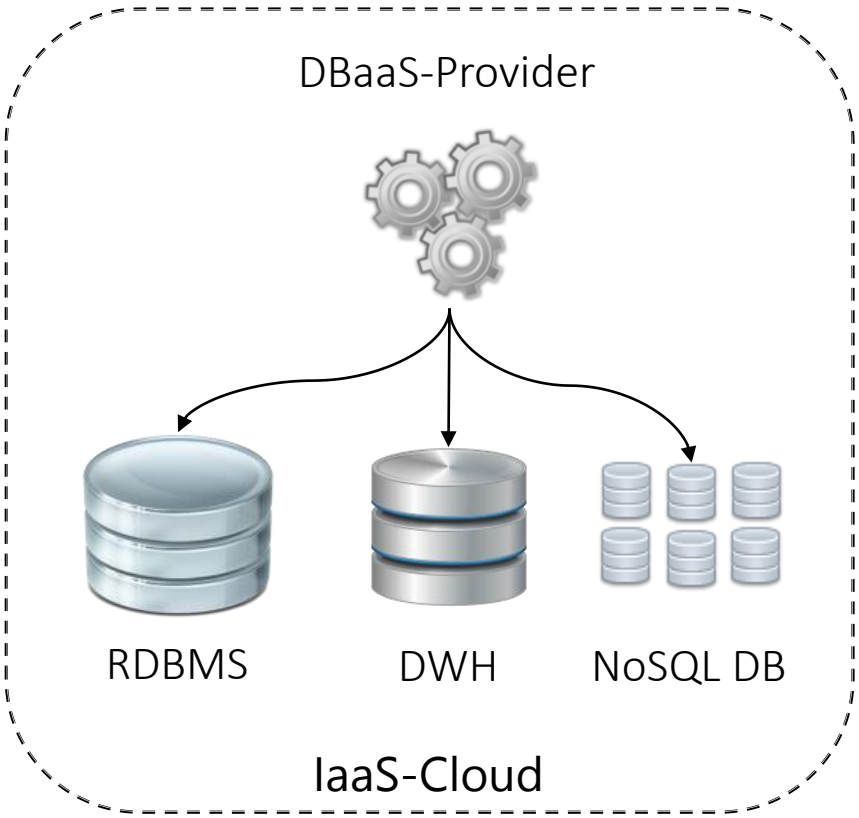


Managed RDBMS/DWH/NoSQL DB





Managed RDBMS/DWH/NoSQL DB



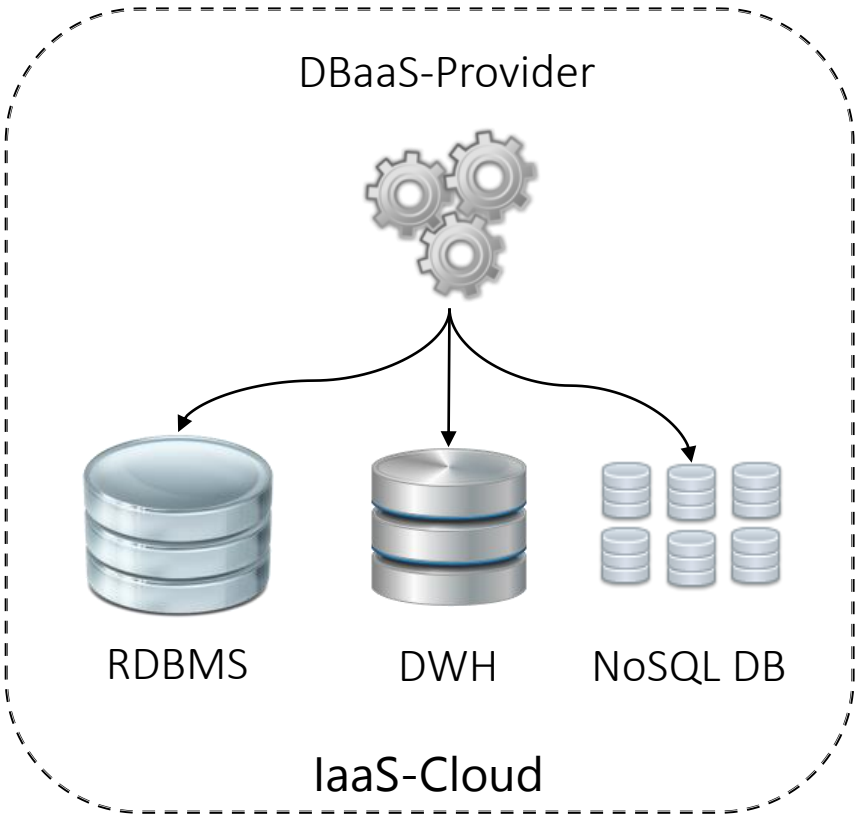
RDBMS

- Amazon RDS
- SQL Azure
- Clustrix
- EDB ENTERPRISEDB
- Google Cloud SQL
- Heroku Postgres

NoSQL DB

- Amazon ElastiCache
- mongoHQ
- mongolab
- Cloudant an IBM Company
- redis
- instaclustr
- bonsai
- Iris Couch

Managed RDBMS/DWH/NoSQL DB



RDBMS

- Amazon RDS
- SQL Azure
- Clustrix
- EDB ENTERPRISEDB
- Google Cloud SQL
- Heroku Postgres

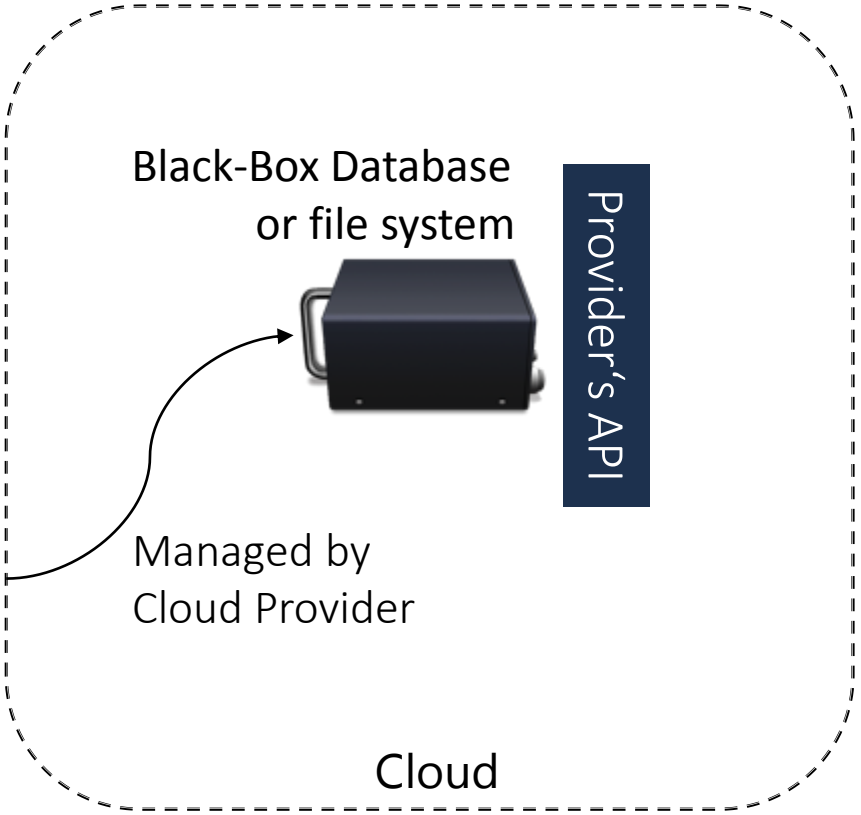
NoSQL DB

- Amazon ElastiCache
- mongoHQ
- mongolab
- Cloudant an IBM Company
- redis
- instaclustr
- bonsai
- Iris Couch

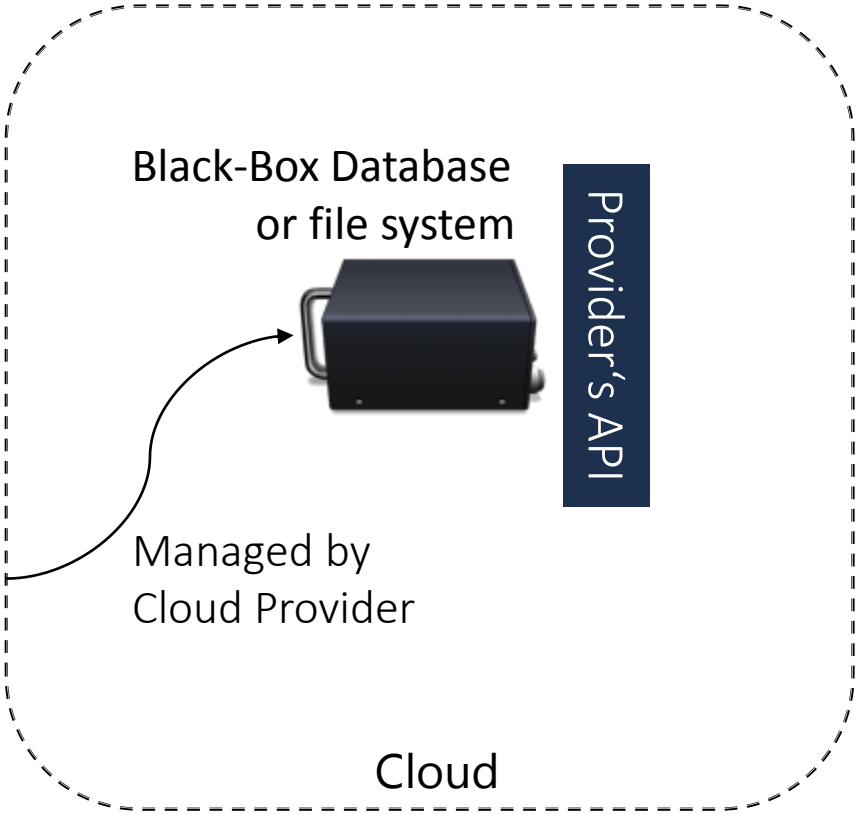
DWH

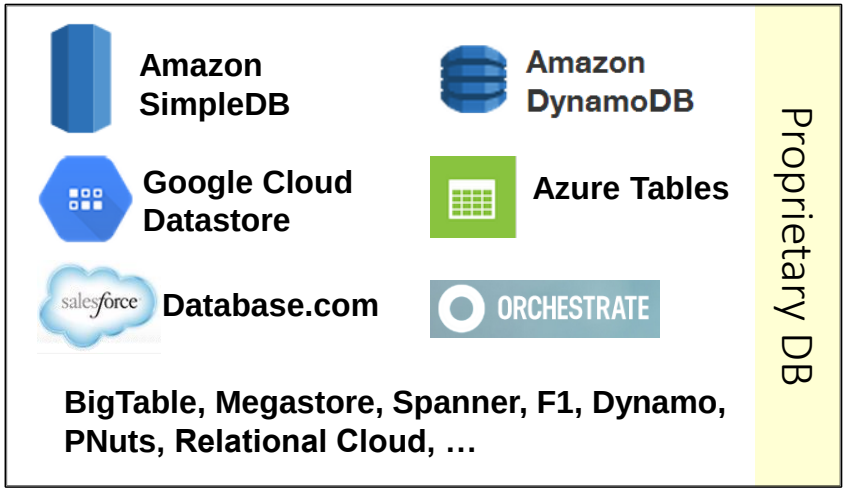
- Amazon Redshift

Proprietary DB/Object Store

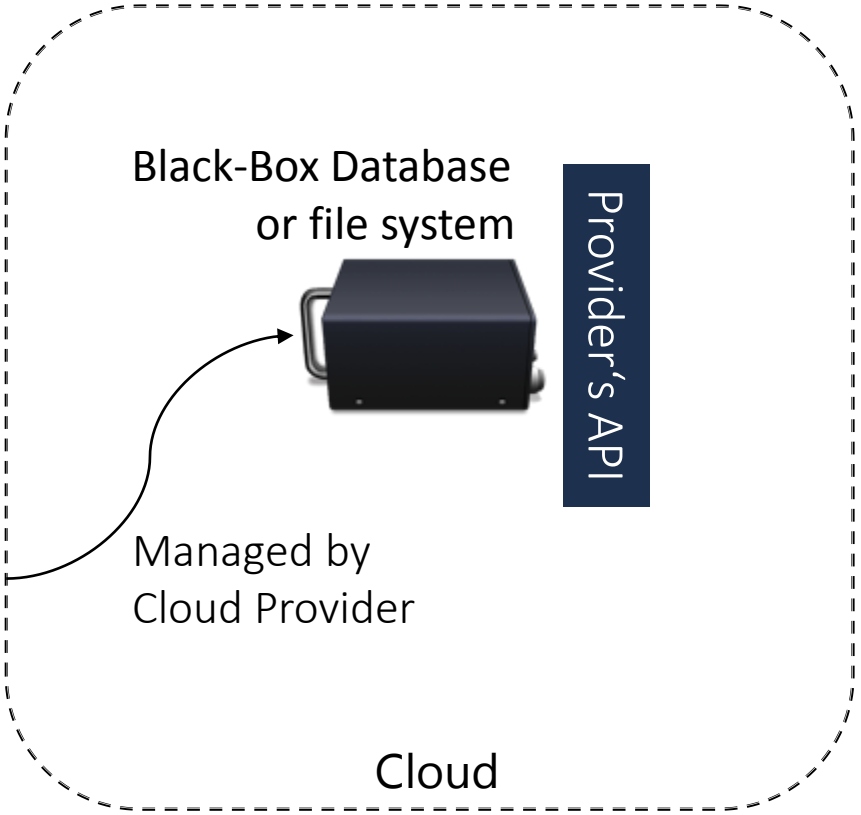


Proprietary DB/Object Store





Proprietary DB/Object Store



Proprietary DB

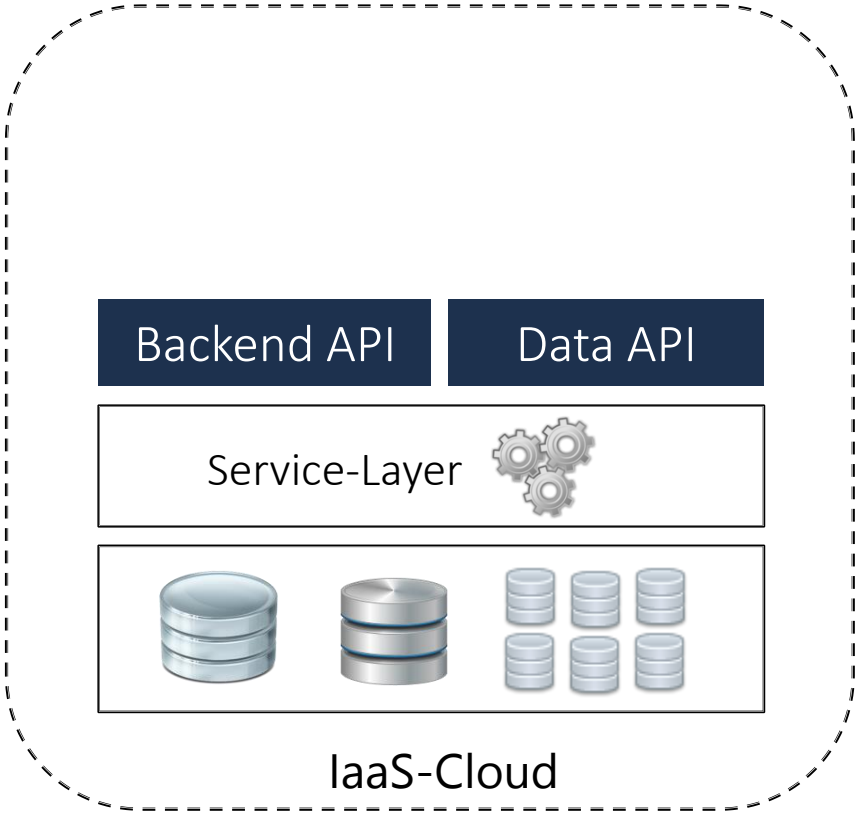
- Amazon SimpleDB
- Amazon DynamoDB
- Google Cloud Datastore
- Azure Tables
- Database.com
- ORCHESTRATE

BigTable, Megastore, Spanner, F1, Dynamo, PNuts, Relational Cloud, …

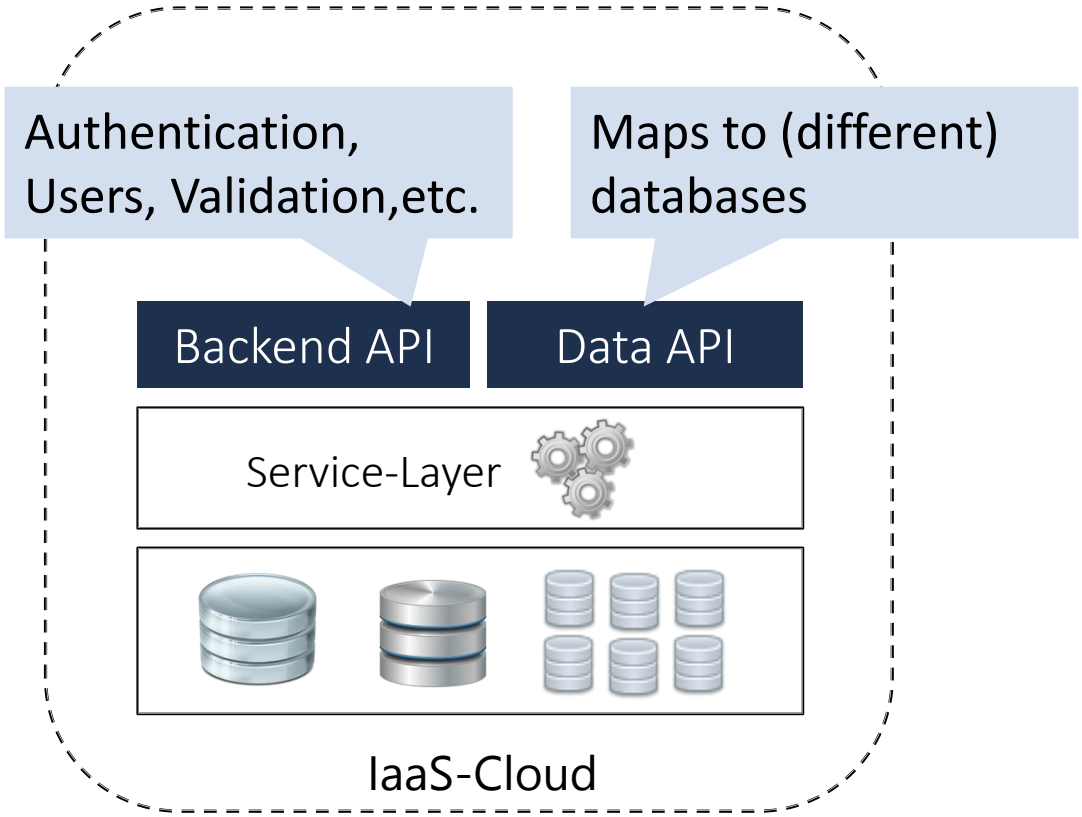
Object Store

- Azure Blob Storage
- Openstack Swift
- Amazon S3
- Google Cloud Storage

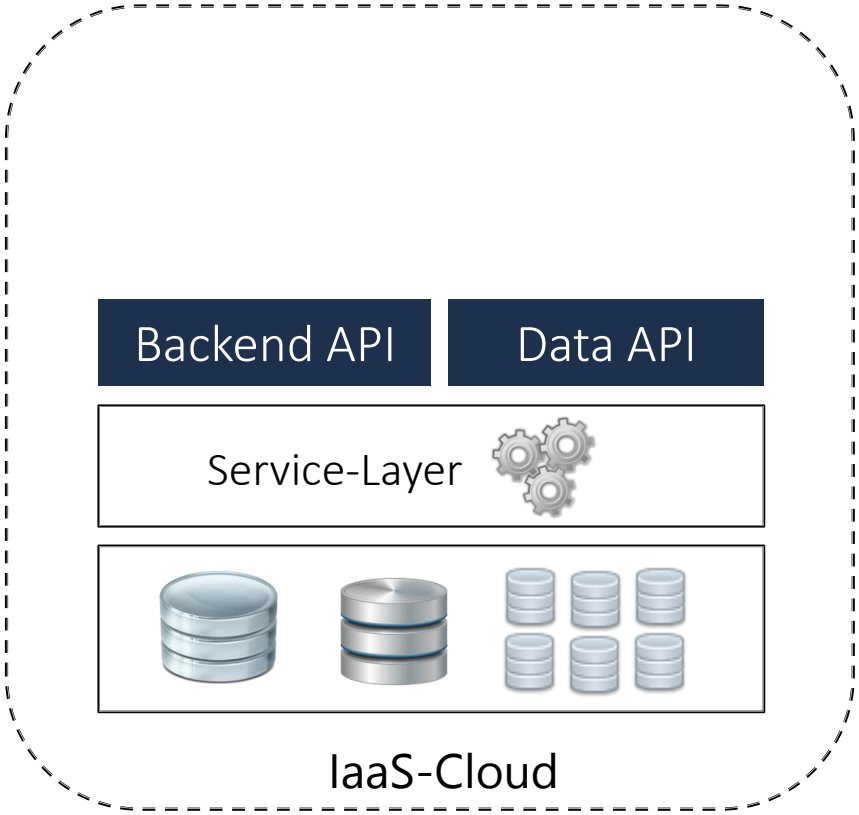
Backend-as-a-Service, Polyglot Persistence Service



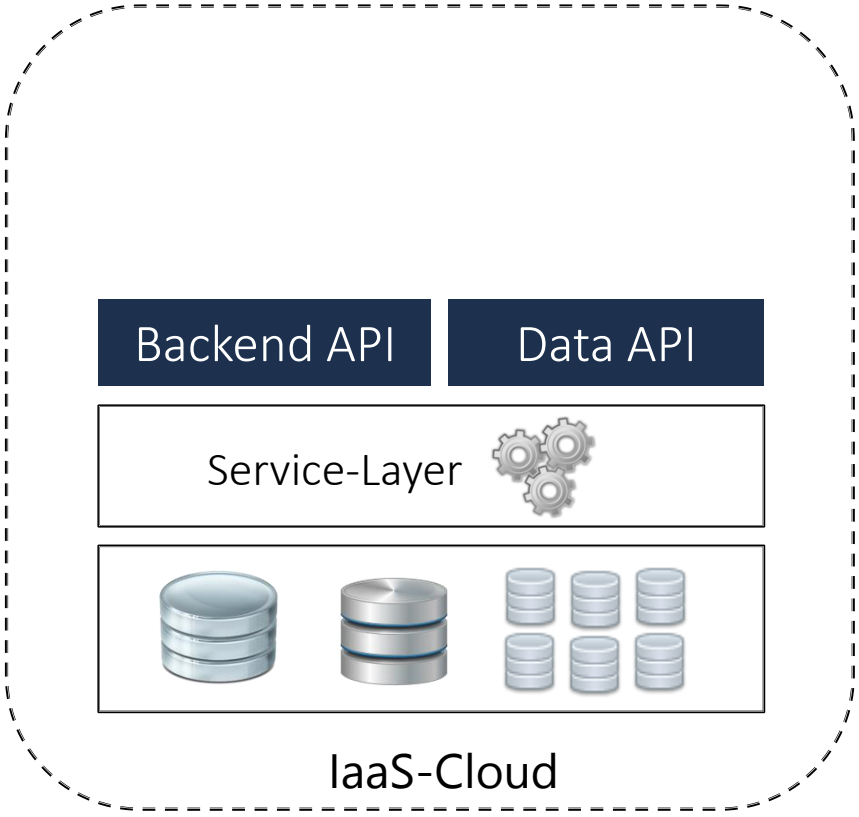
Backend-as-a-Service, Polyglot Persistence Service



Backend-as-a-Service, Polyglot Persistence Service



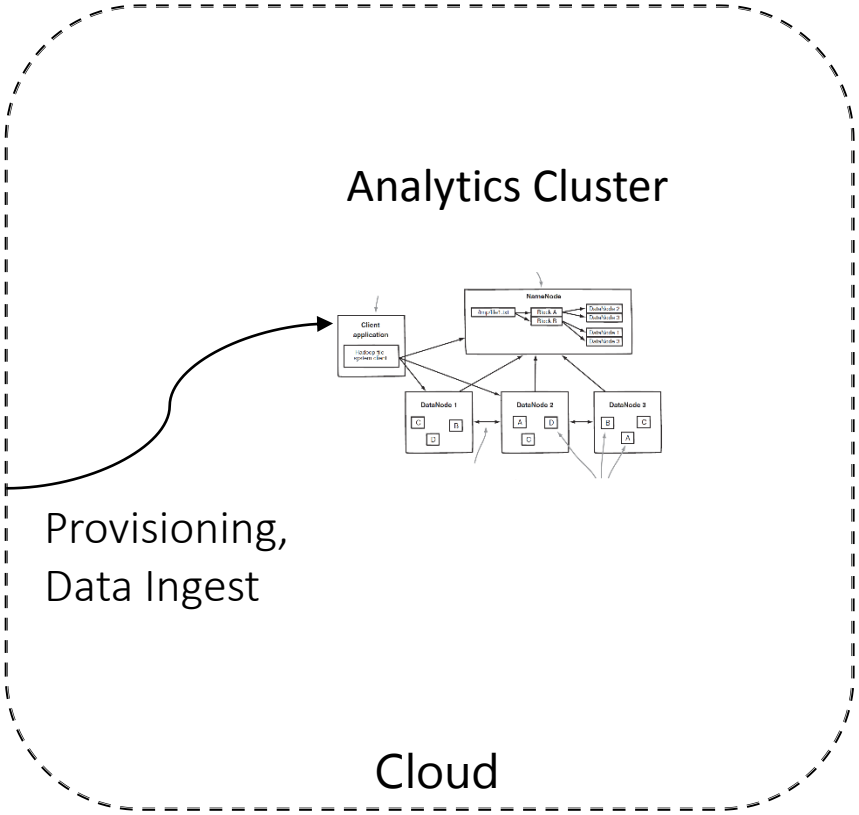
Backend-as-a-Service, Polyglot Persistence Service



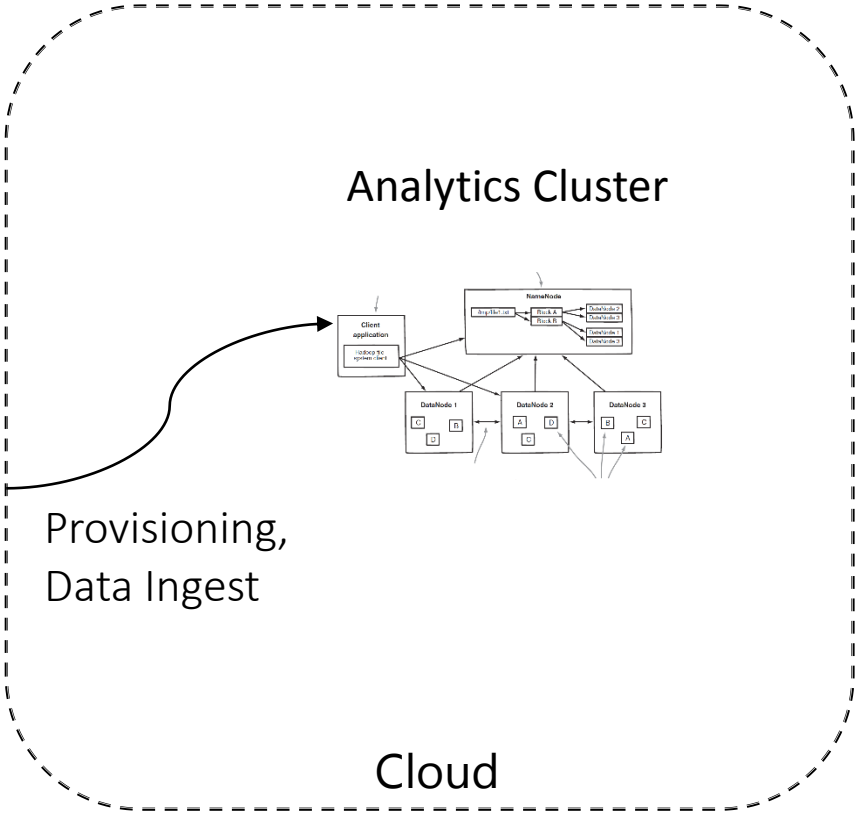
Firebase
GoInstant
Meteor PREVIEW
Realtime BaaS

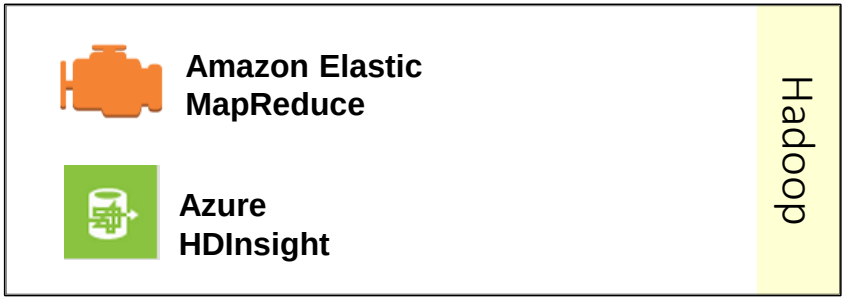
Parse
AppCelerator Cloud
APIOMAT
BaQend
Build faster Apps faster.
BaaS

Analytics-as-a-Service

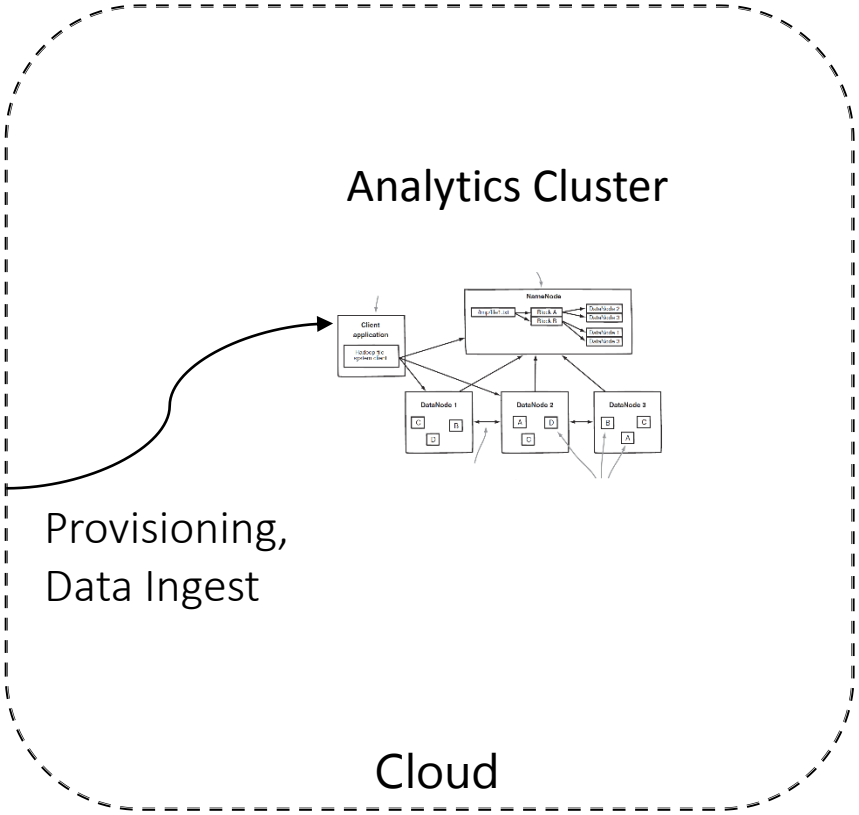


Analytics-as-a-Service





Analytics-as-a-Service



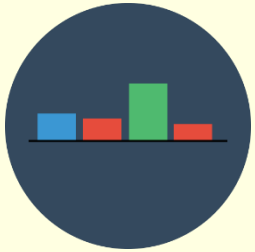
Hadoop

- Amazon Elastic MapReduce
- Azure HDInsight

Custom

- Google BigQuery
- Google Prediction API

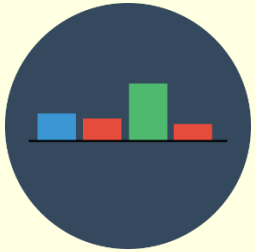
DBaaS: Common Aspects



#1 Metric: Total Cost

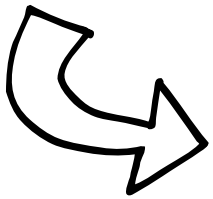
Daniela Florescu and Donald Kossmann “Rethinking cost and performance of database systems”, SIGMOD Rec. 2009.

DBaaS: Common Aspects



#1 Metric: Total Cost

Daniela Florescu and Donald Kossmann “Rethinking cost and performance of database systems”, SIGMOD Rec. 2009.



Maximum utilization of available hardware:

Multi-Tenancy

DBaaS: Common Aspects

- **Multi-Tenancy -** four common approaches:

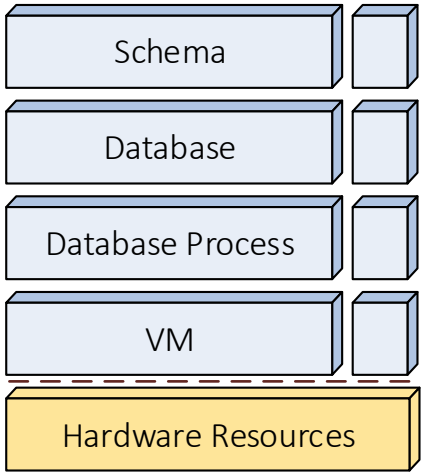
Private OS | Private Process/DB | Private Schema | Shared Schema

T. Kiefer, W. Lehner “Private table database virtualization for dbaas” UCC, 2011

DBaaS: Common Aspects

- **Multi-Tenancy -** four common approaches:

Private OS | **Private Process/DB** | **Private Schema** | **Shared Schema**



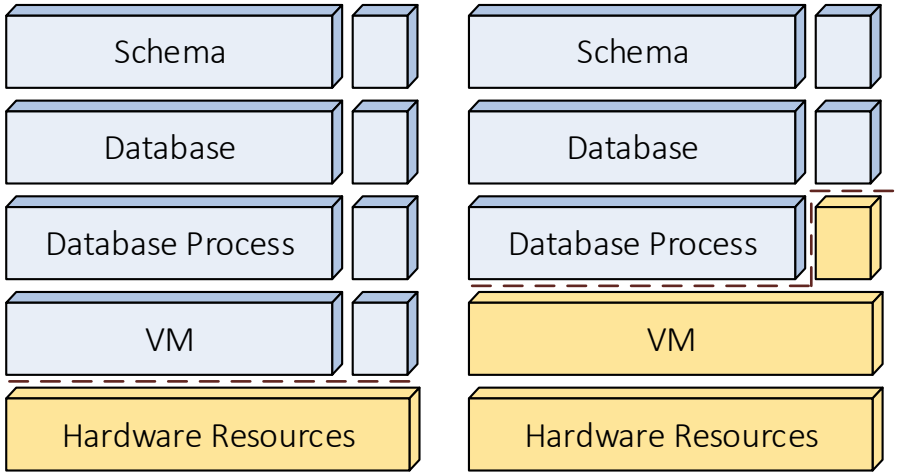
e.g. Amazon RDS

T. Kiefer, W. Lehner “Private table database virtualization for dbaas” UCC, 2011

DBaaS: Common Aspects

- **Multi-Tenancy -** four common approaches:

Private OS | **Private Process/DB** | **Private Schema** | **Shared Schema**



e.g. Amazon RDS

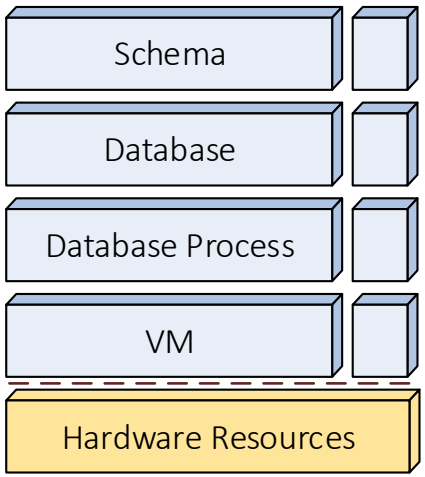
e.g. MongoHQ

T. Kiefer, W. Lehner “Private table database virtualization for dbaas” UCC, 2011

DBaaS: Common Aspects

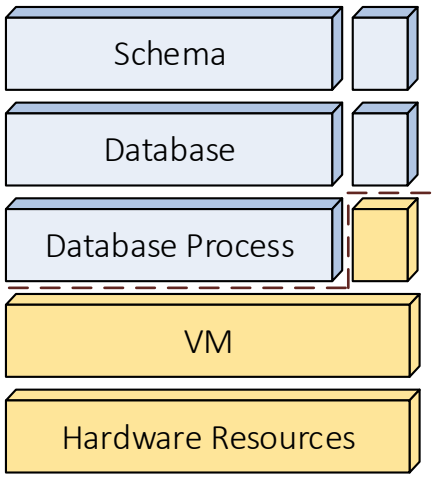
- **Multi-Tenancy -** four common approaches:

Private OS



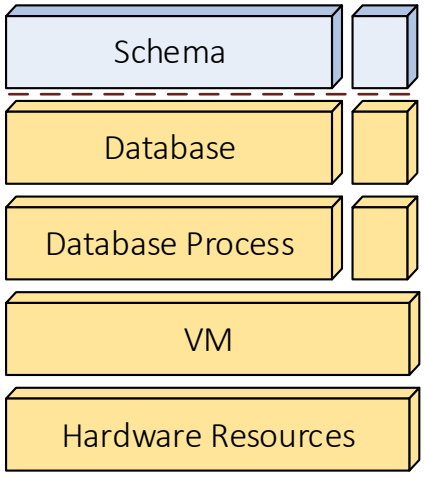
e.g. Amazon RDS

Private Process/DB



e.g. MongoHQ

Private Schema



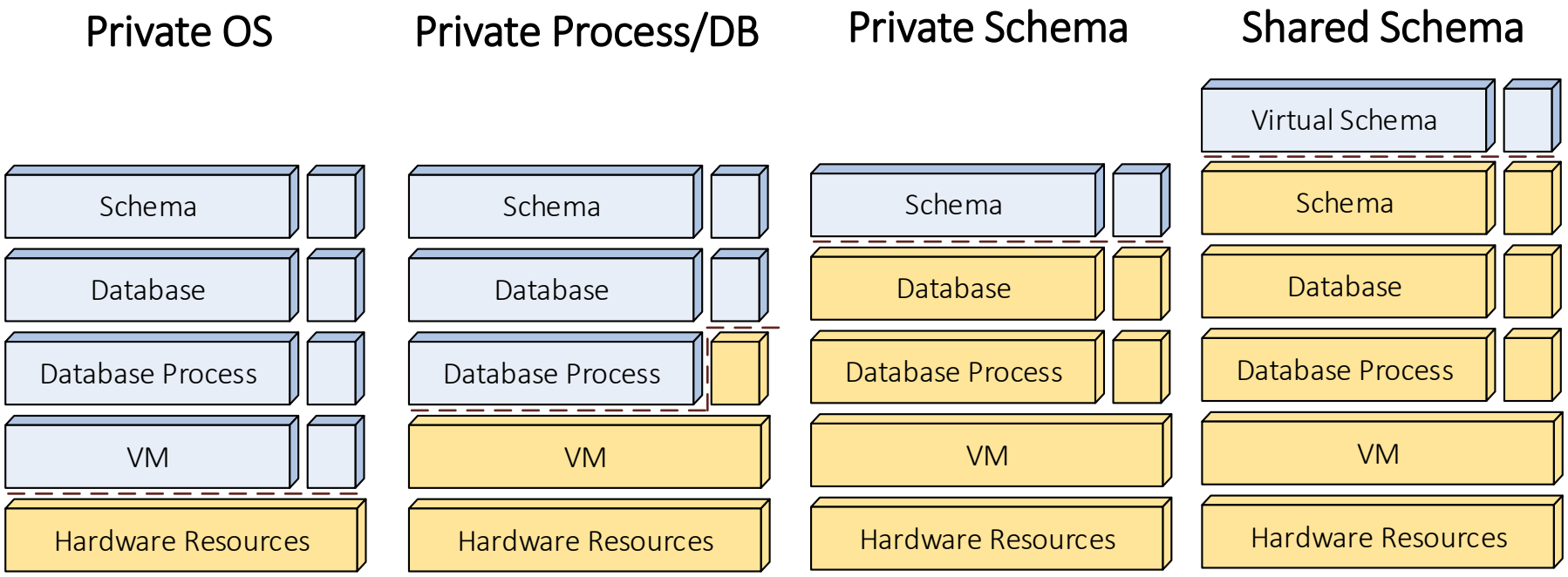
e.g. Google DataStore

Shared Schema

T. Kiefer, W. Lehner “Private table database virtualization for dbaas” UCC, 2011

DBaaS: Common Aspects

- **Multi-Tenancy -** four common approaches:



T. Kiefer, W. Lehner "Private table database virtualization for dbaas" UCC, 2011

DBaaS: Common Aspects

- **Multi-Tenancy -** four common approaches:

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

W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

DBaaS: Common Aspects

- Billing Models:

DBaaS: Common Aspects

- Billing Models:

Pay-per-use
Parameters: Network, Bandwidth, Storage, CPU, Requests, etc.
Payment: Pre-Paid, Post-Paid
Variants: On-Demand, Auction, Reserved

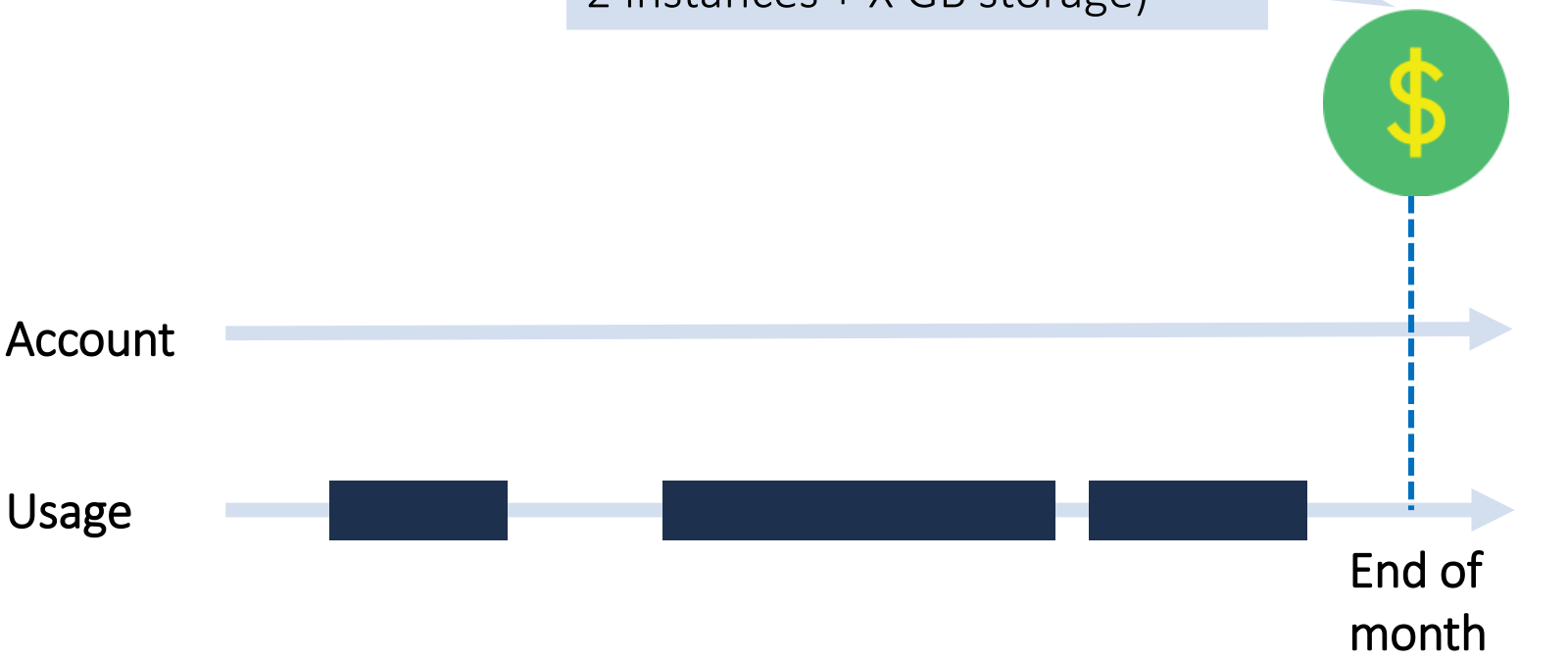
e.g. DynamoDB

DBaaS: Common Aspects

- Billing Models:

Plan-based
Parameters: Allocated Plan (e.g. 2 instances + X GB storage)

e.g. MongoHQ

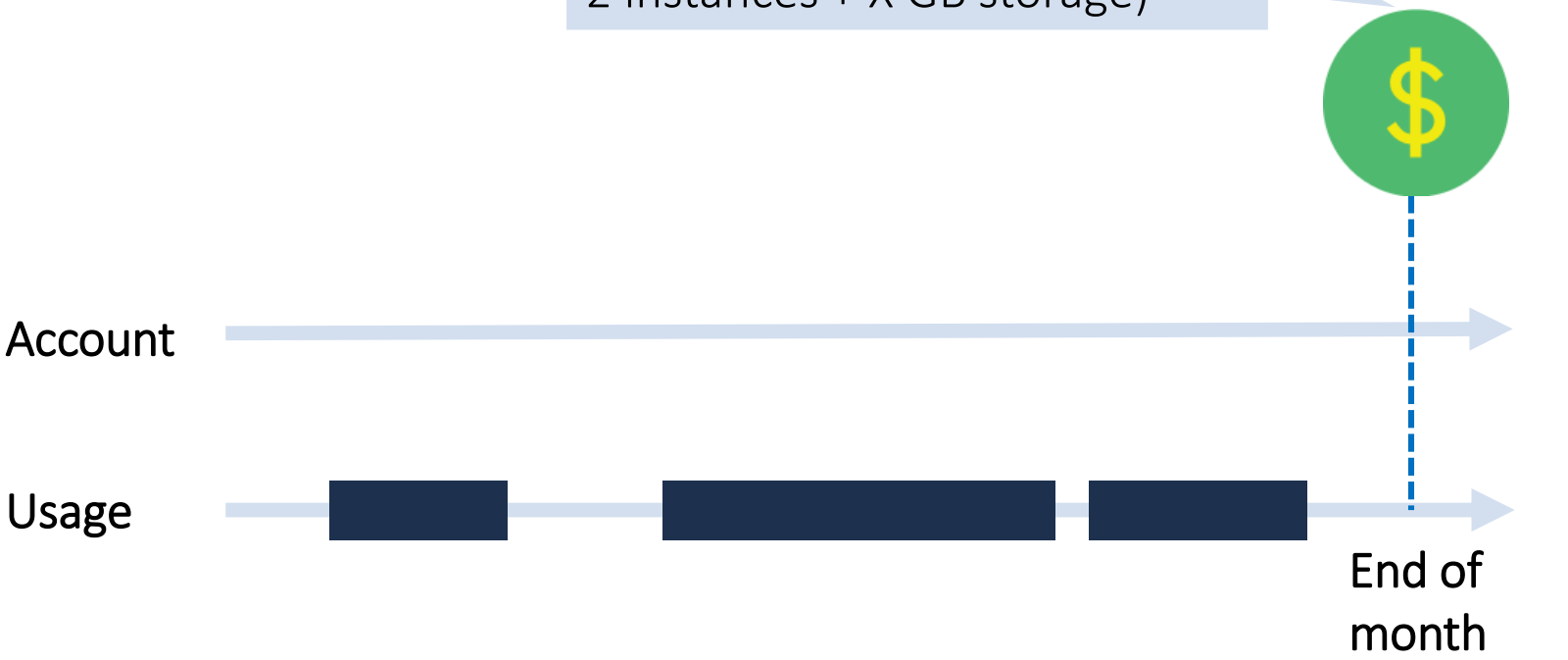


DBaaS: Common Aspects

- Billing Models:

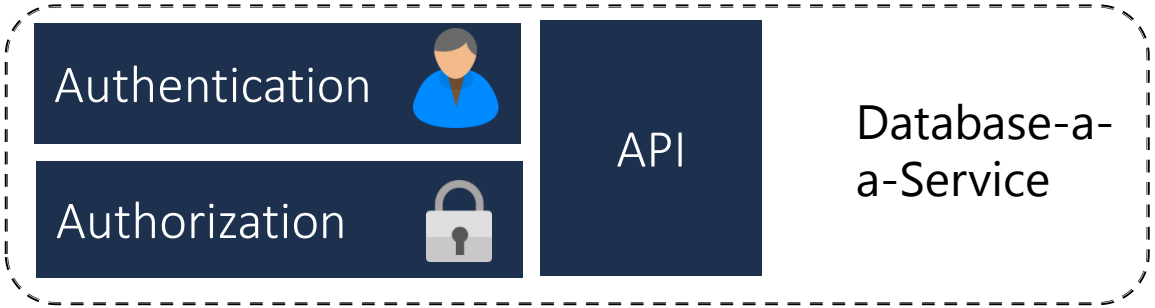
Plan-based
Parameters: Allocated Plan (e.g. 2 instances + X GB storage)

e.g. MongoHQ



Free Tier: free plan or free initial account credit

DBaaS: Common Aspects



DBaaS: Common Aspects

DBaaS: Common Aspects

Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML

DBaaS: Common Aspects

Used extensively

Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML

DBaaS: Common Aspects

Used extensively

Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML

DBaaS: Common Aspects

Used extensively

Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML

DBaaS: Common Aspects

Used extensively

Internal Schemes	External Identity Provider	Federated Identity (SSO)
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

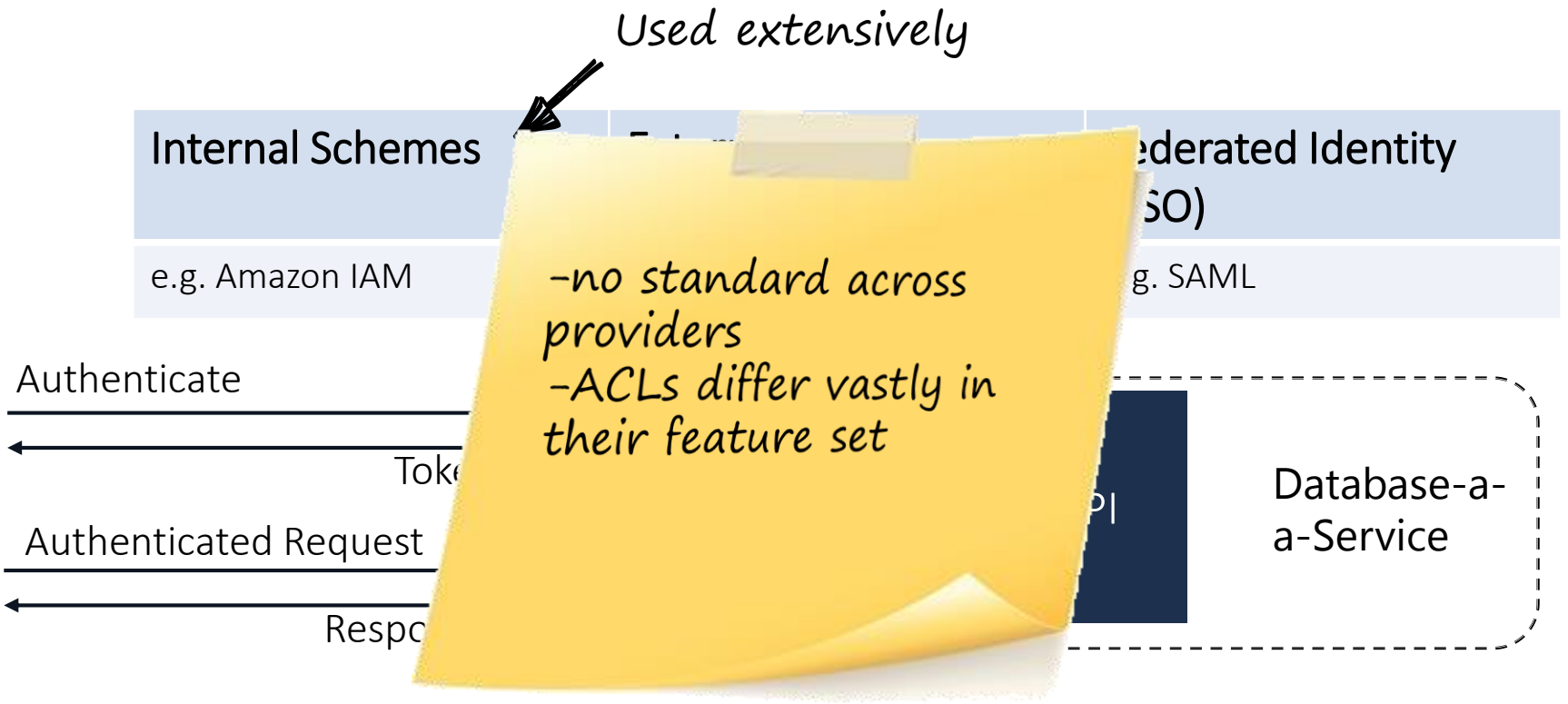
DBaaS: Common Aspects

Used extensively

Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML

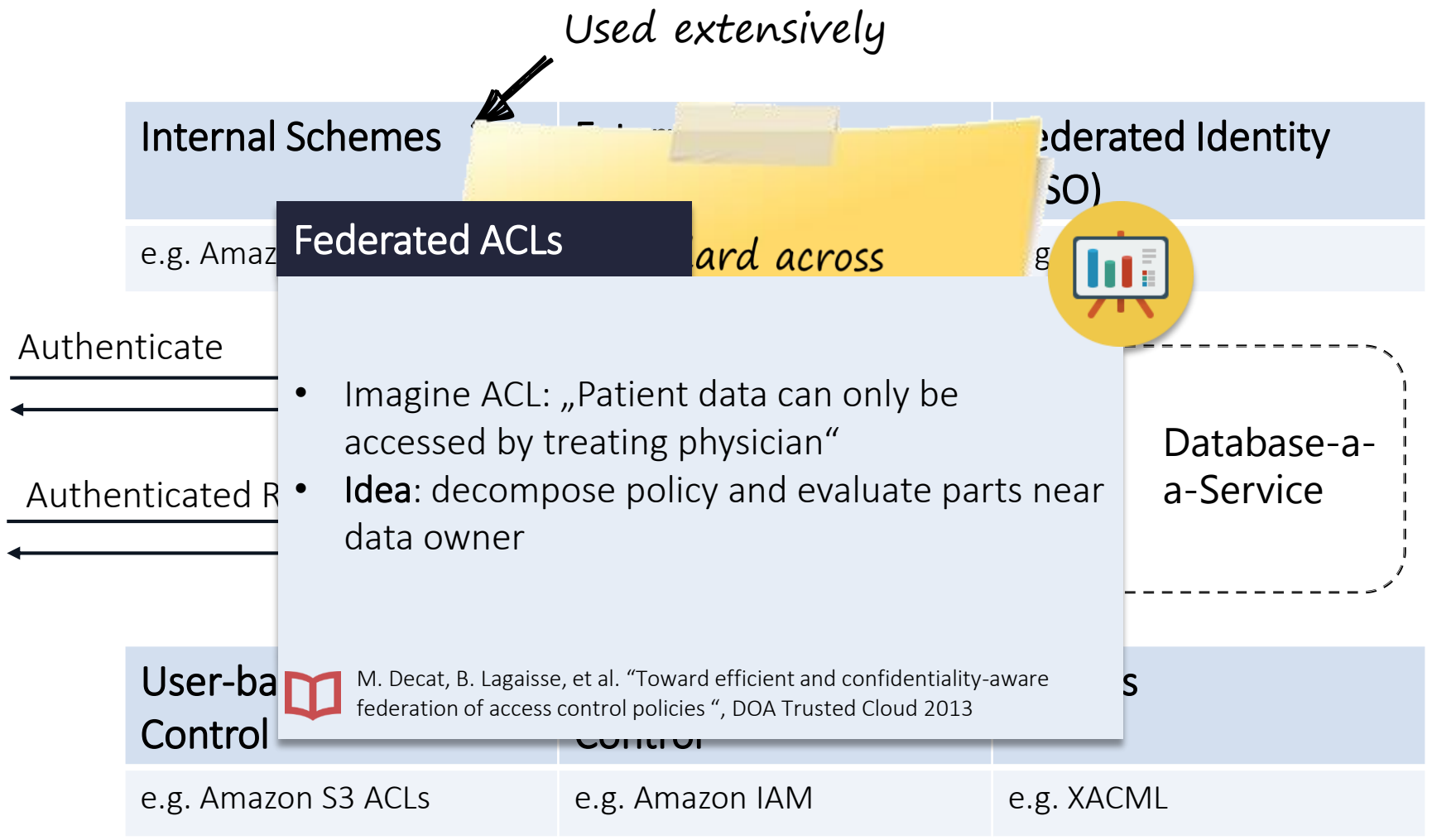
User-based Access Control	Role-based Access Control	Policies
e.g. Amazon S3 ACLs	e.g. Amazon IAM	e.g. XACML

DBaaS: Common Aspects



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

DBaaS: Common Aspects

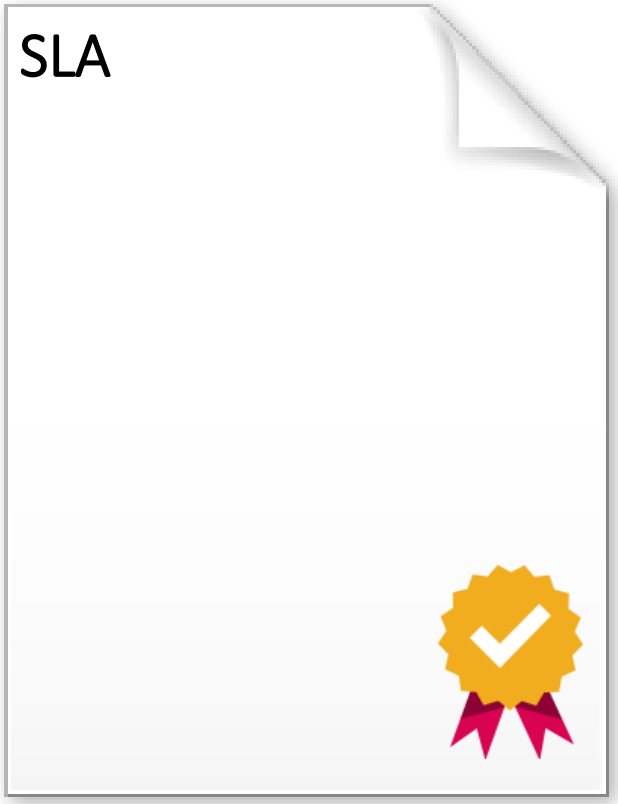


- Imagine ACL: „Patient data can only be accessed by treating physician“
- **Idea**: decompose policy and evaluate parts near data owner

M. Decat, B. Lagaisse, et al. “Toward efficient and confidentiality-aware federation of access control policies “, DOA Trusted Cloud 2013

DBaaS: Common Aspects

- Service Level Agreements

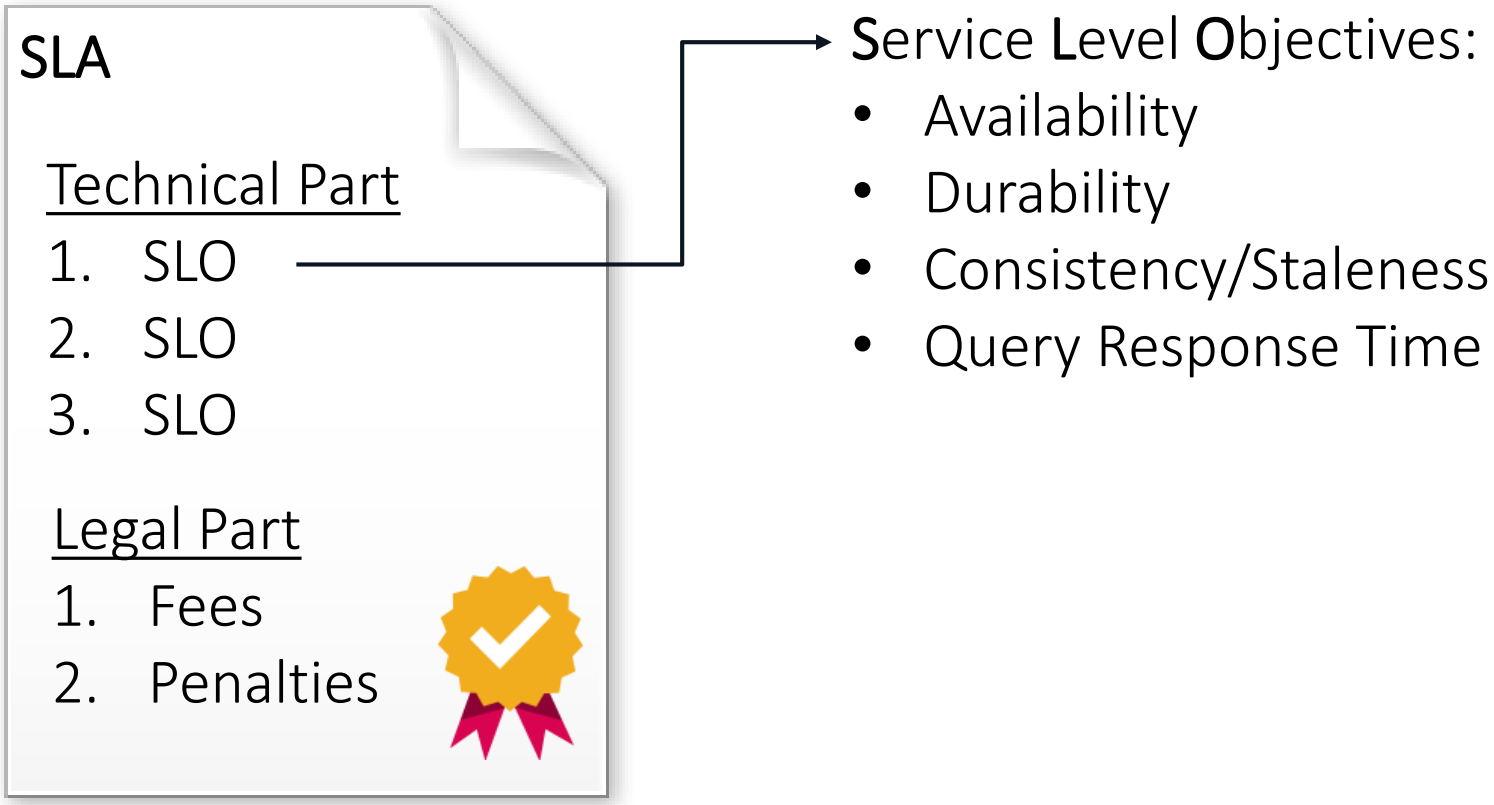


DBaaS: Common Aspects

- Service Level Agreements

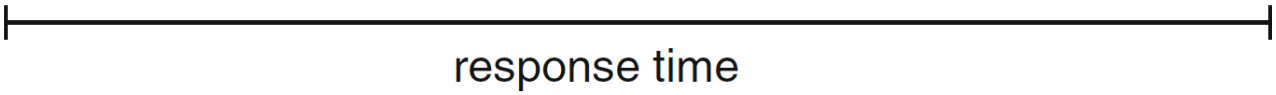
DBaaS: Common Aspects

- Service Level Agreements



DBaaS: Common Aspects

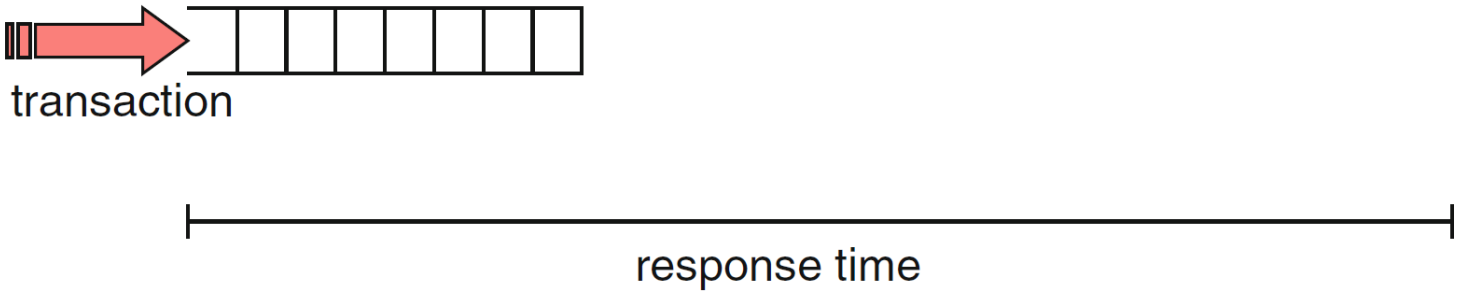
- **SLAs** – achieved through **Workload Management**



W. Lehner, U. Sattler “Web-scale Data Management for the Cloud” Springer, 2013

DBaaS: Common Aspects

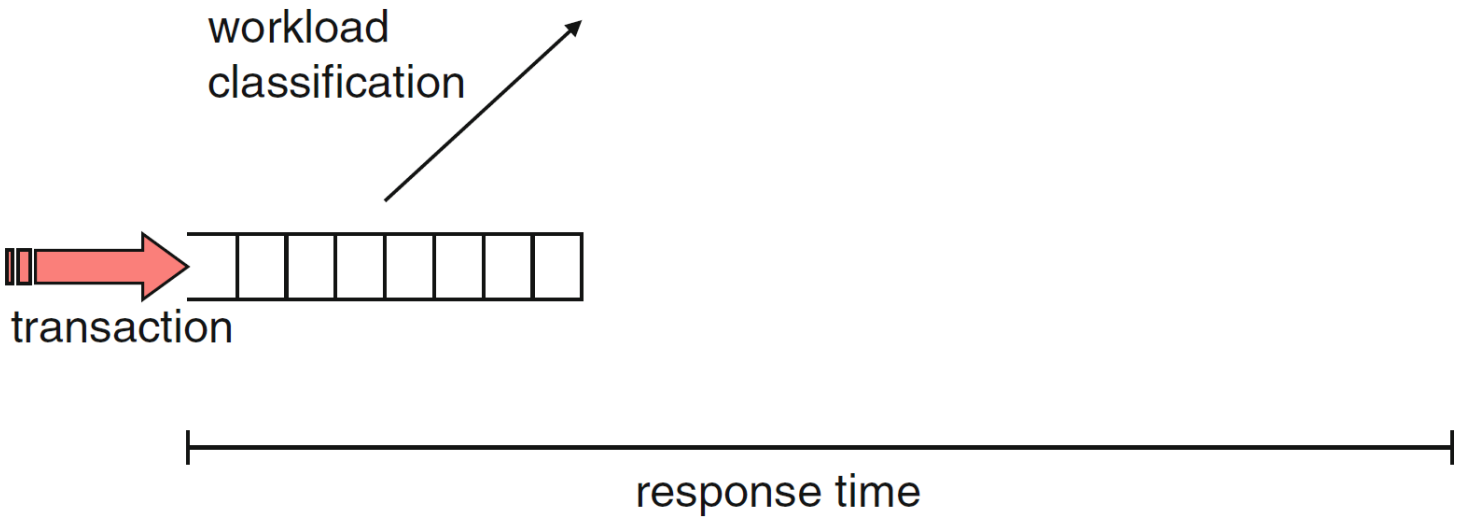
- **SLAs** – achieved through **Workload Management**



W. Lehner, U. Sattler “Web-scale Data Management for the Cloud” Springer, 2013

DBaaS: Common Aspects

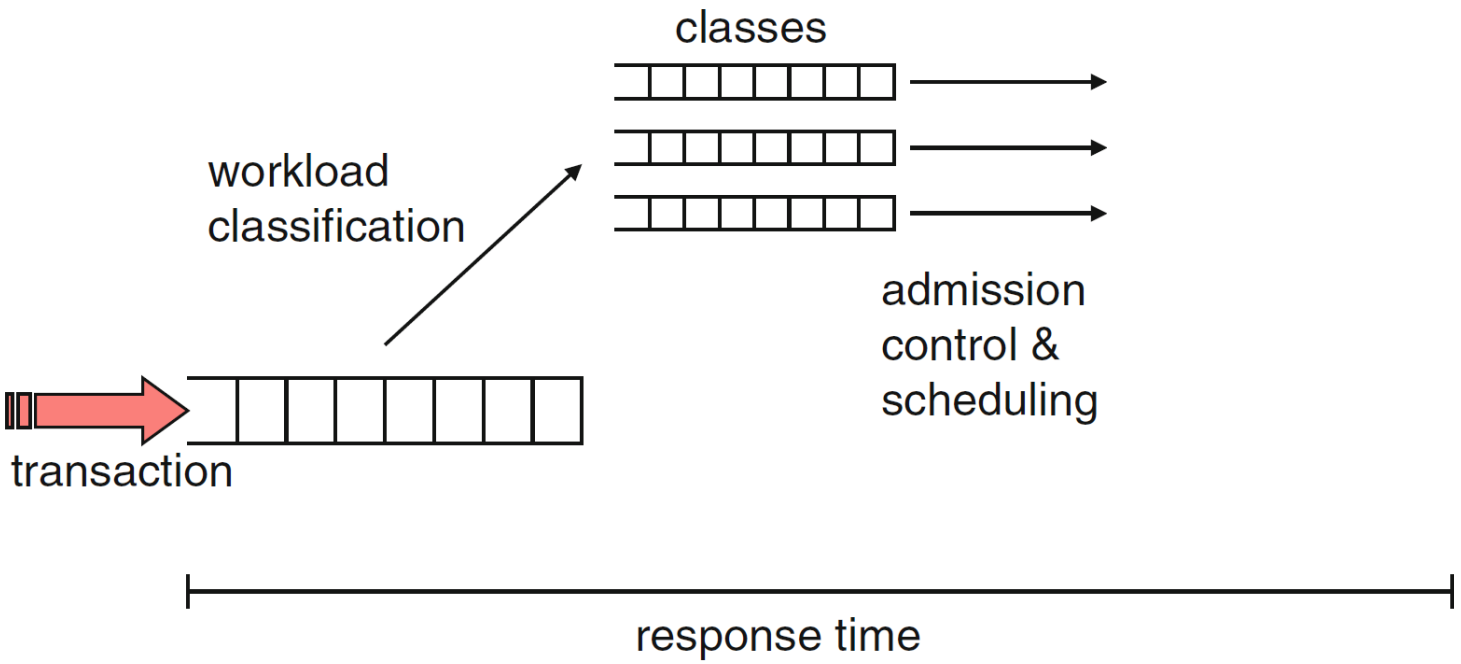
- **SLAs** – achieved through **Workload Management**



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

DBaaS: Common Aspects

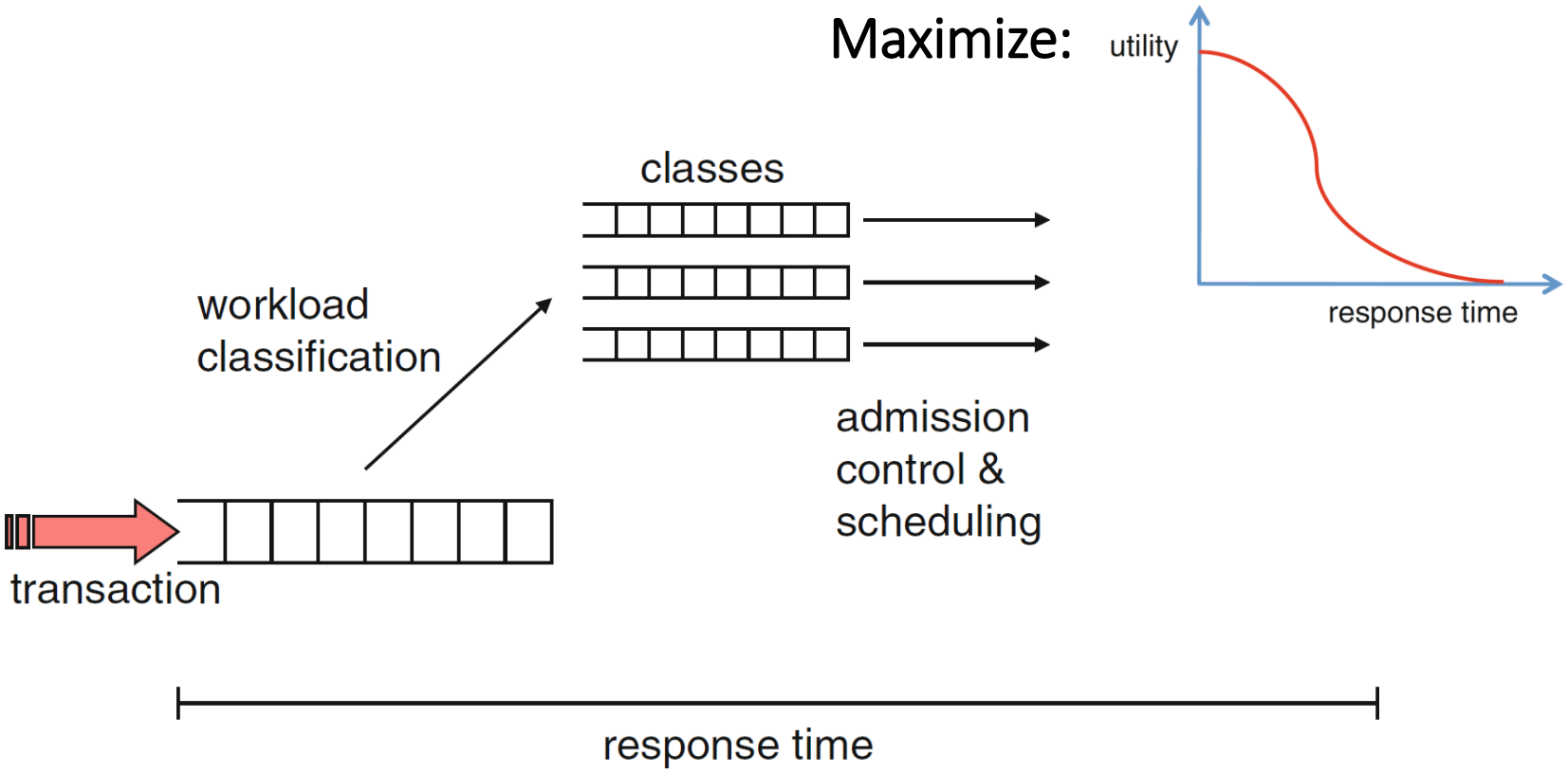
- **SLAs** – achieved through **Workload Management**



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

DBaaS: Common Aspects

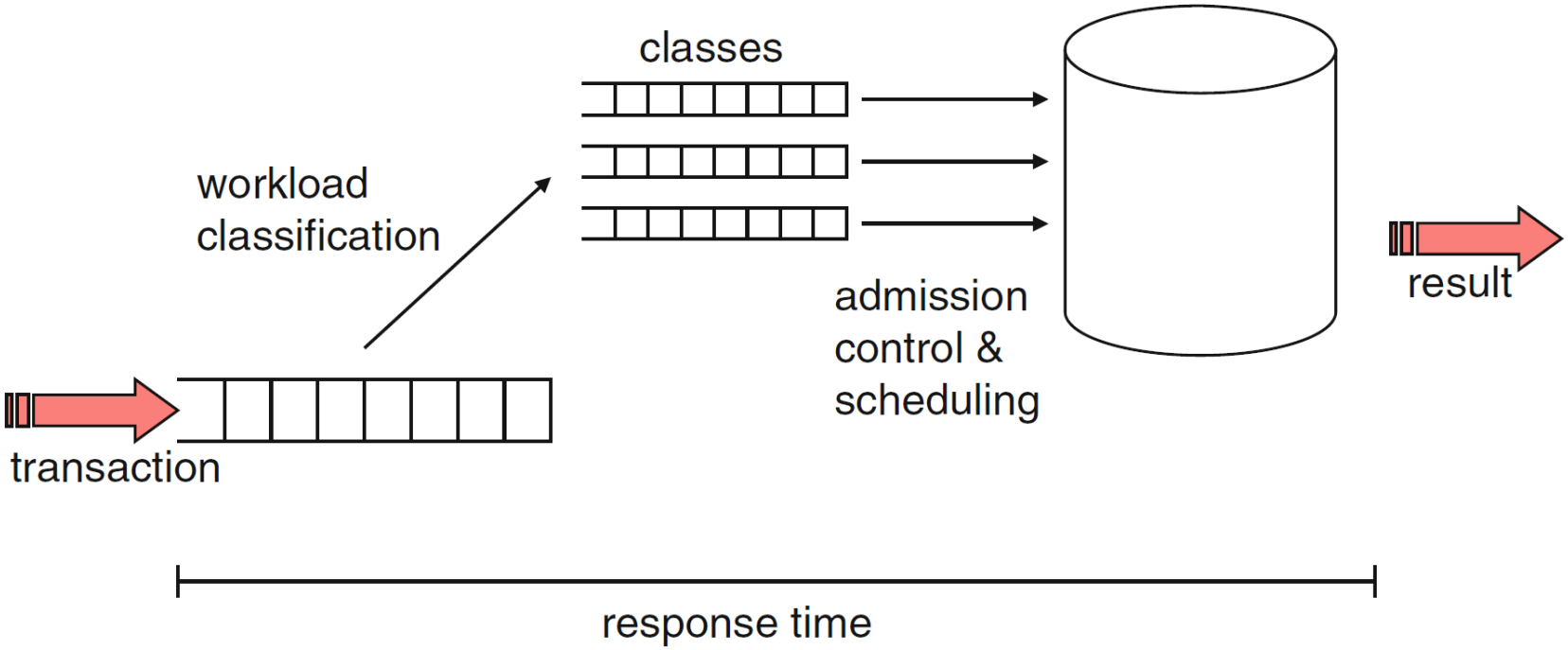
- **SLAs** – achieved through **Workload Management**



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

DBaaS: Common Aspects

- **SLAs** – achieved through **Workload Management**



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

DBaaS: Common Aspects

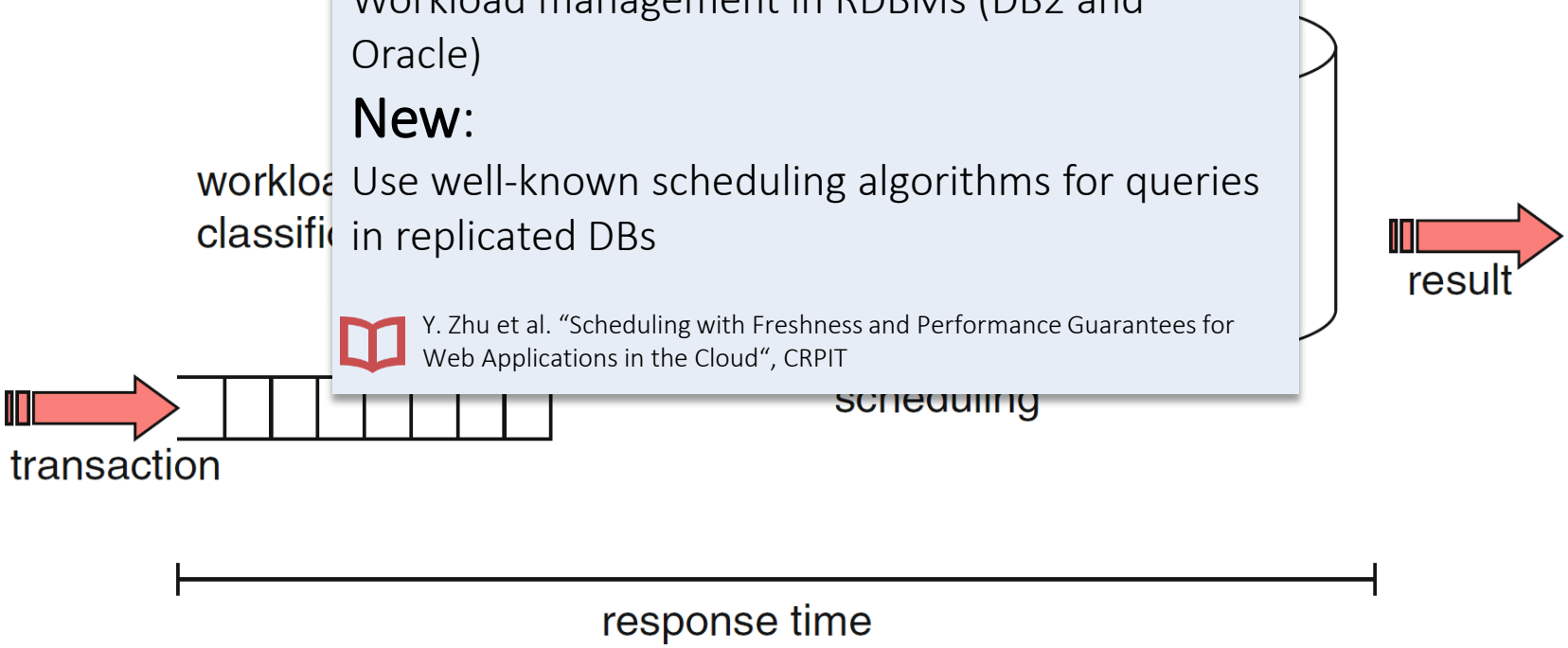
- SLAs – achieved through Workload Management

QOS for NoSQL DBs

Old:
Workload management in RDBMs (DB2 and Oracle)

New:
Use well-known scheduling algorithms for queries in replicated DBs

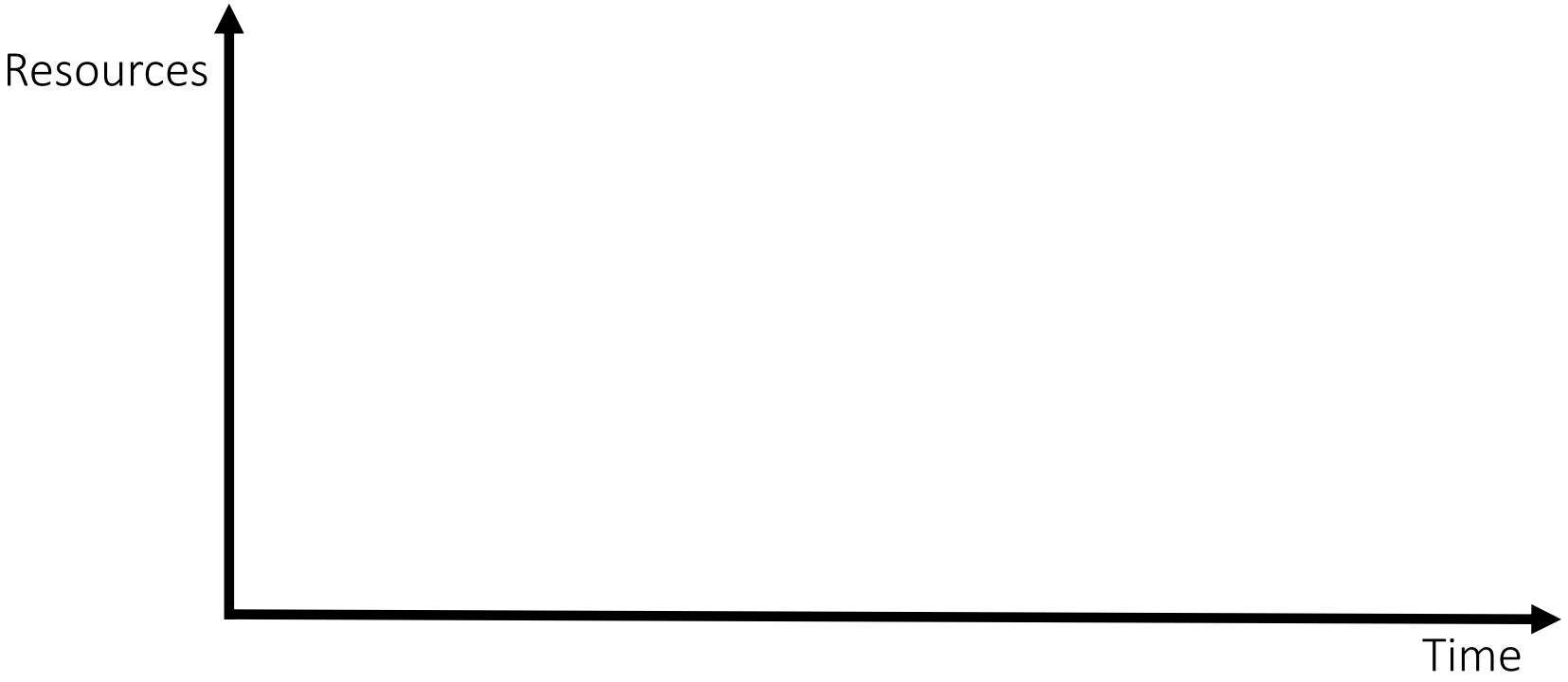
Y. Zhu et al. "Scheduling with Freshness and Performance Guarantees for Web Applications in the Cloud", CRPIT



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

DBaaS: Common Aspects

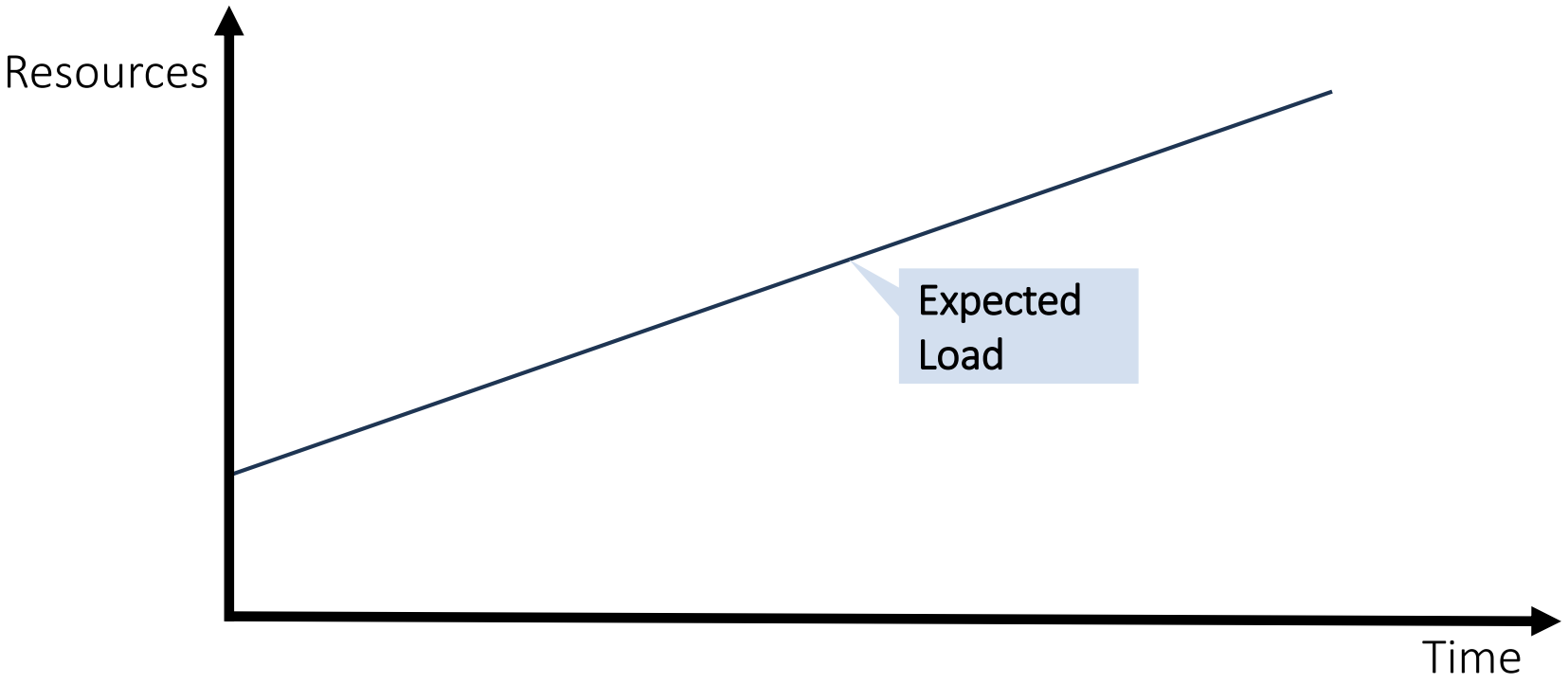
- **Resource Provisionig**
- **Goal:** Resources ⇔ SLAs



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

DBaaS: Common Aspects

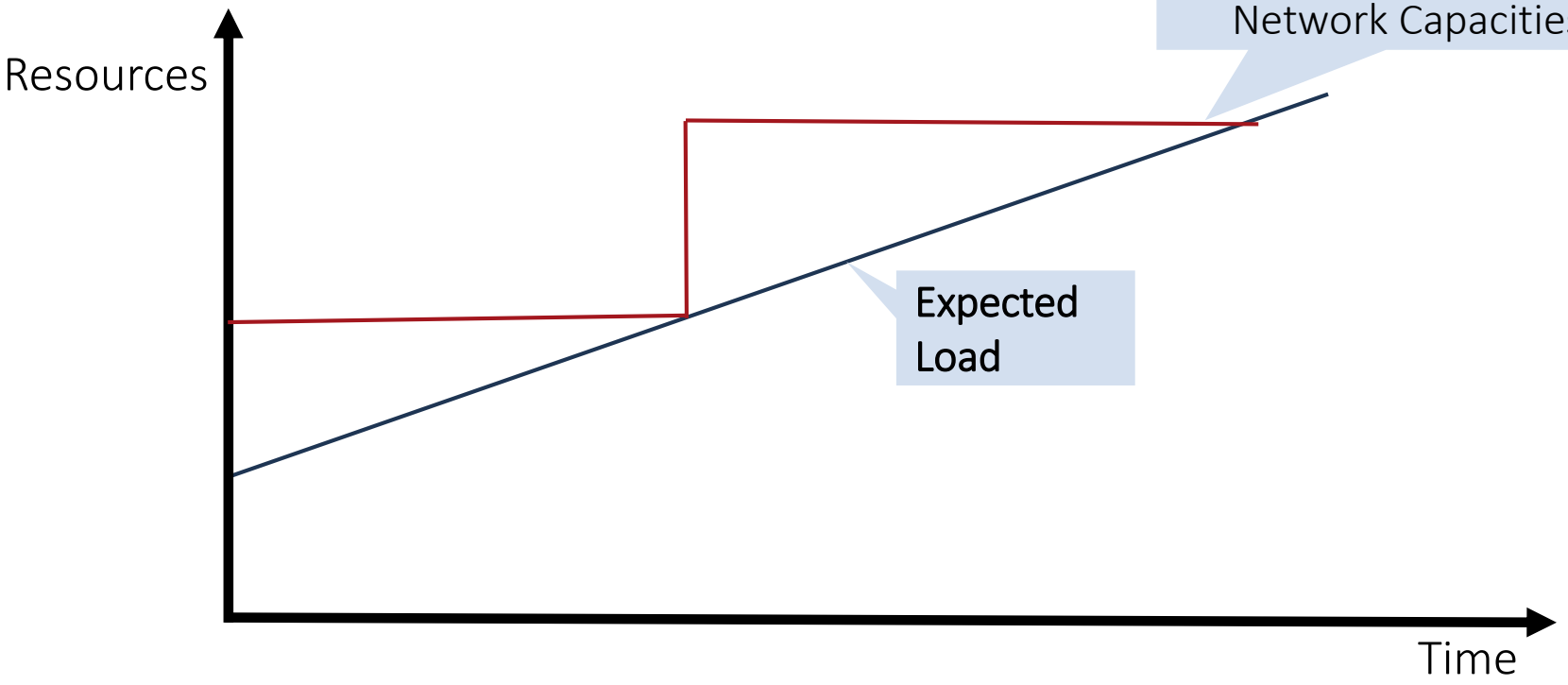
- **Resource Provisionig**
- **Goal:** Resources ⇔ SLAs



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

DBaaS: Common Aspects

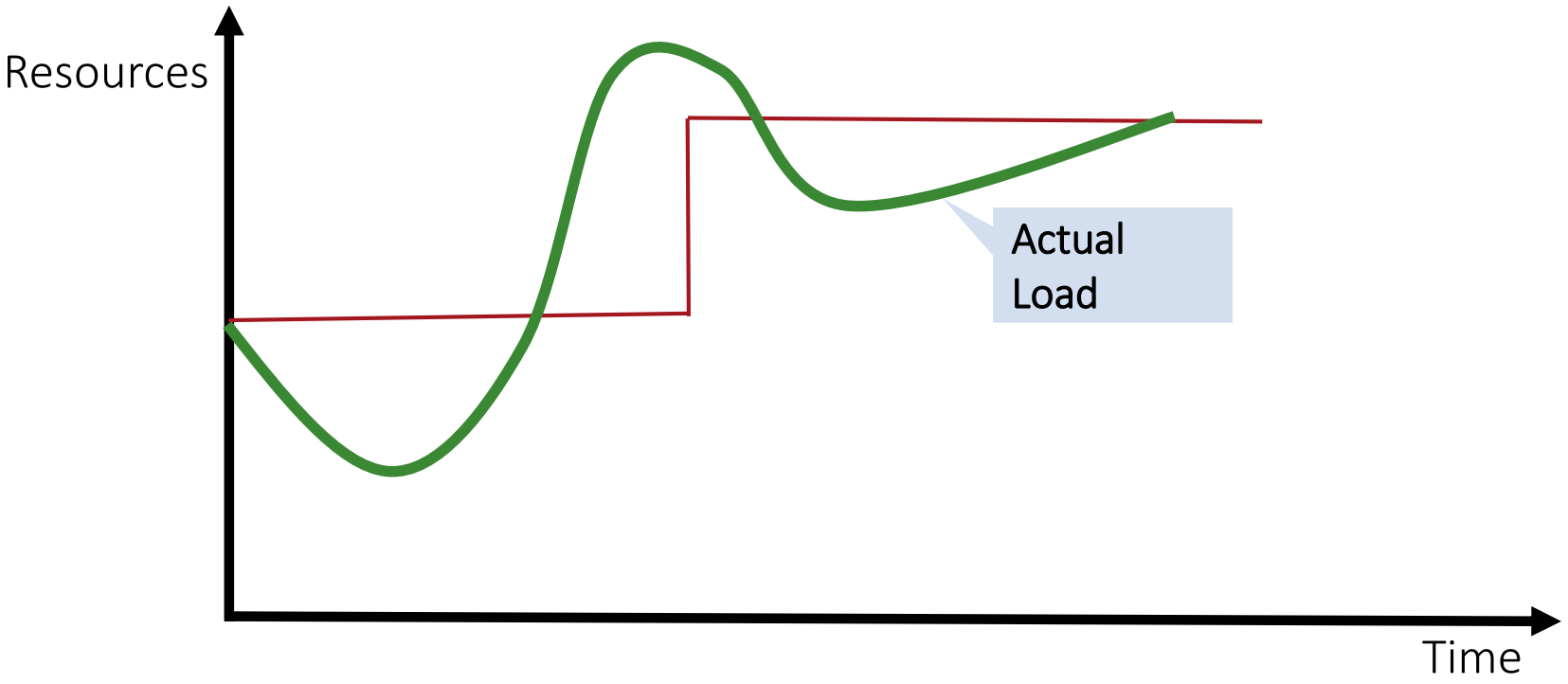
- **Resource Provisionig**
- **Goal:** Resources ⇔ SLAs



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

DBaaS: Common Aspects

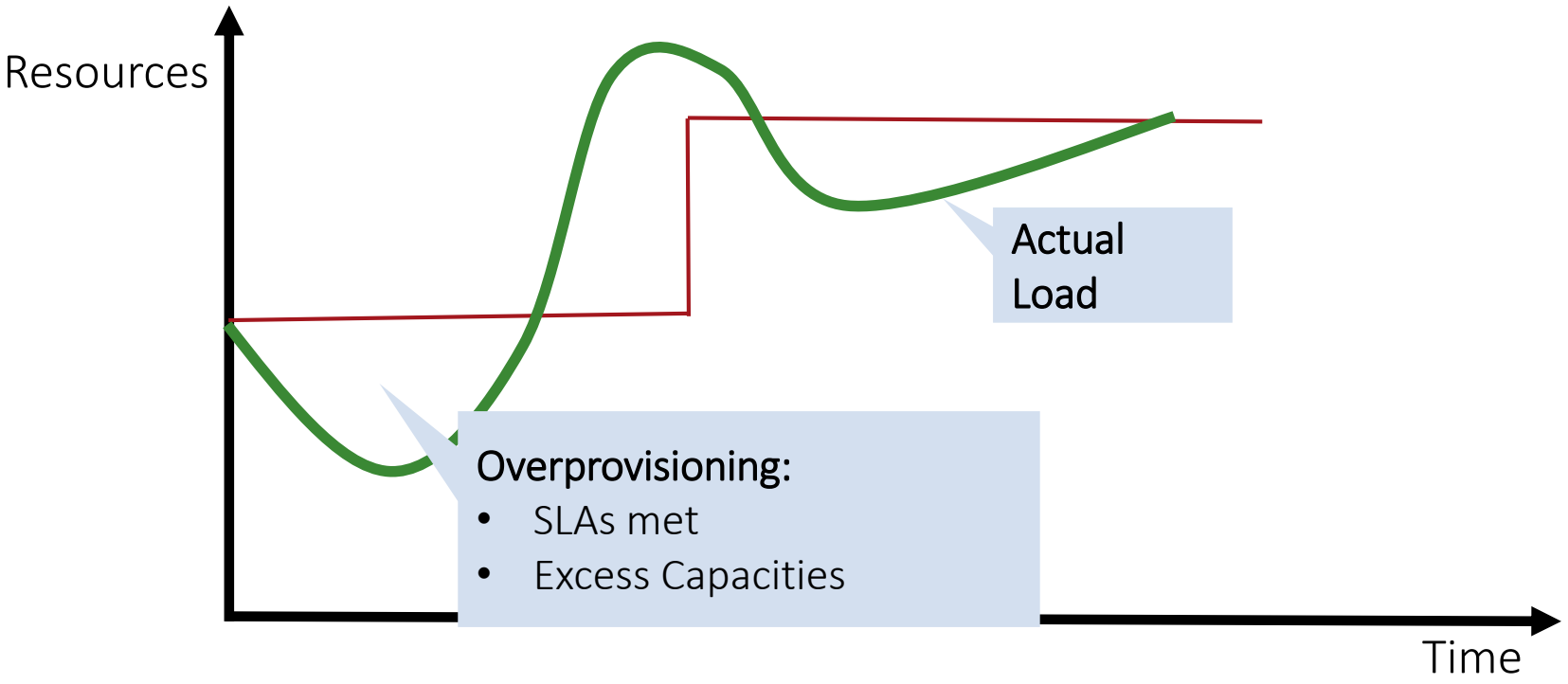
- **Resource Provisionig**
- **Goal:** Resources ⇔ SLAs



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

DBaaS: Common Aspects

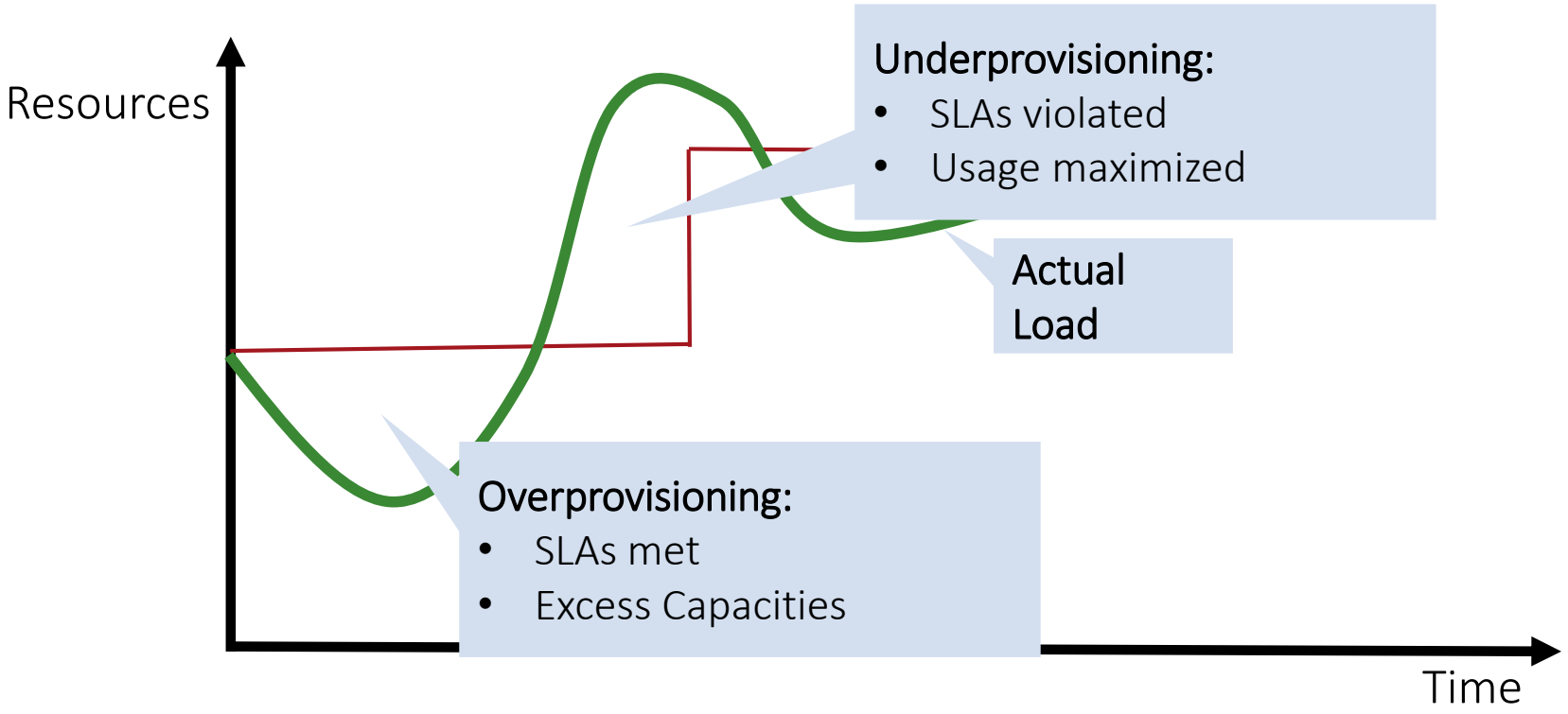
- **Resource Provisionig**
- **Goal:** Resources ⇔ SLAs



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

DBaaS: Common Aspects

- **Resource Provisionig**
- **Goal:** Resources ⇔ SLAs



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

DBaaS: Common Aspects

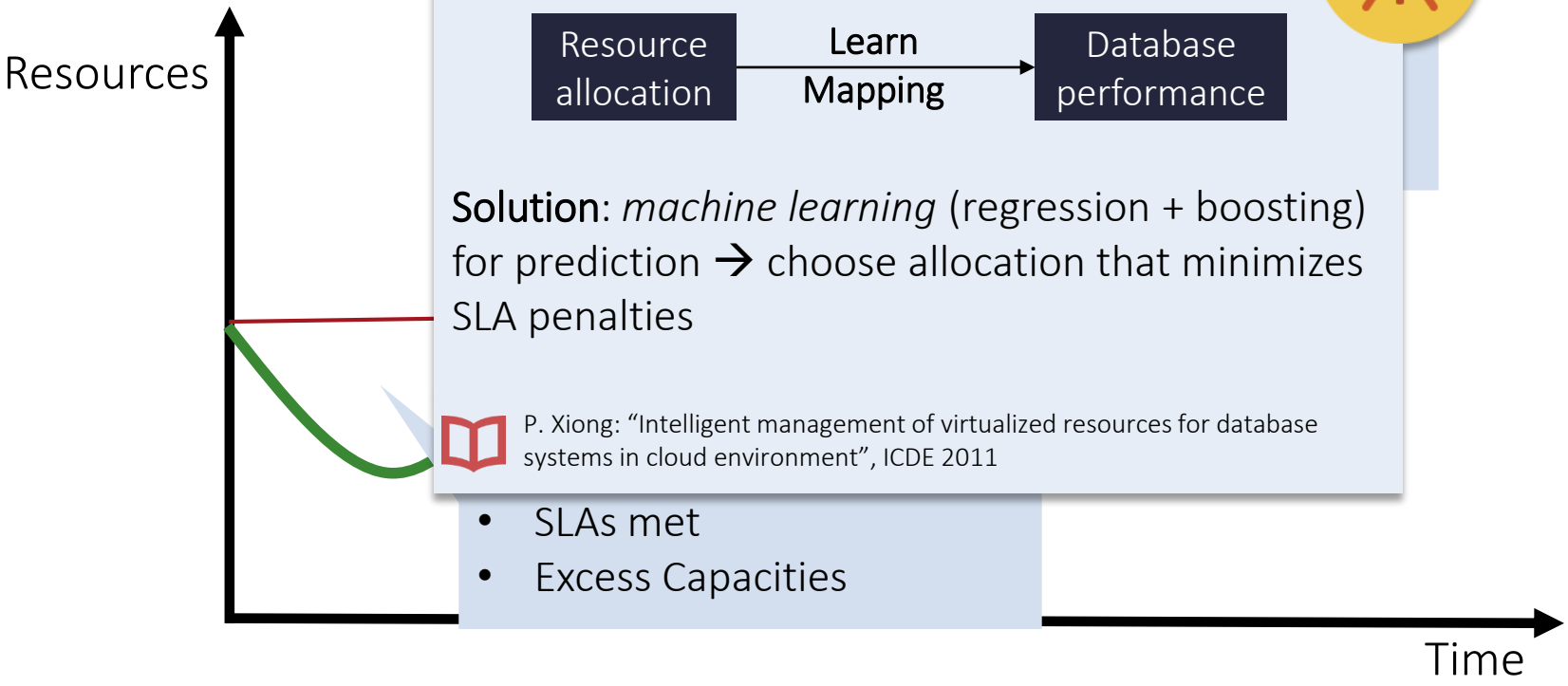
- **Resource Provisionig**
- **Goal:** Reso

SmartSLA

Resource allocation → **Learn Mapping** → Database performance

Solution: *machine learning* (regression + boosting) for prediction → choose allocation that minimizes SLA penalties

P. Xiong: "Intelligent management of virtualized resources for database systems in cloud environment", ICDE 2011



- SLAs met
- Excess Capacities

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

DBaaS: General Considerations

Functional Requirements

Scan-Querys

Transactions

Conditional Updates

Joins

Query by Example

Analytics

Non-Functional Requirements

Scalability of Data Volume

Write Scalability

Read Scalability

Elasticity

Read-Availability

Consistency

Write-Availability

Durability

Read-Latency

Write-Throughput

Write-Latency

DBaaS: General Considerations

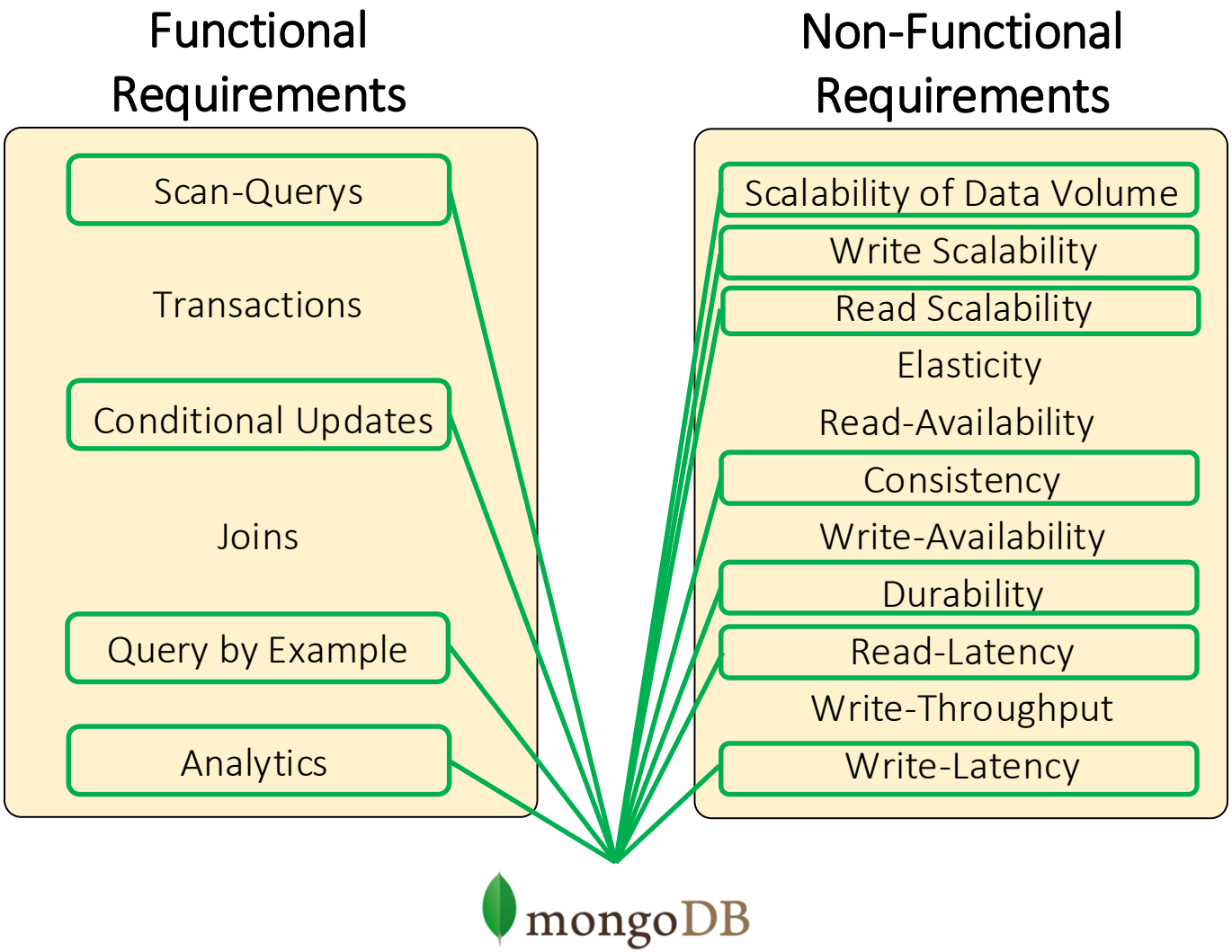
Functional Requirements

- Scan-Querys
- Transactions
- Conditional Updates
- Joins
- Query by Example
- Analytics

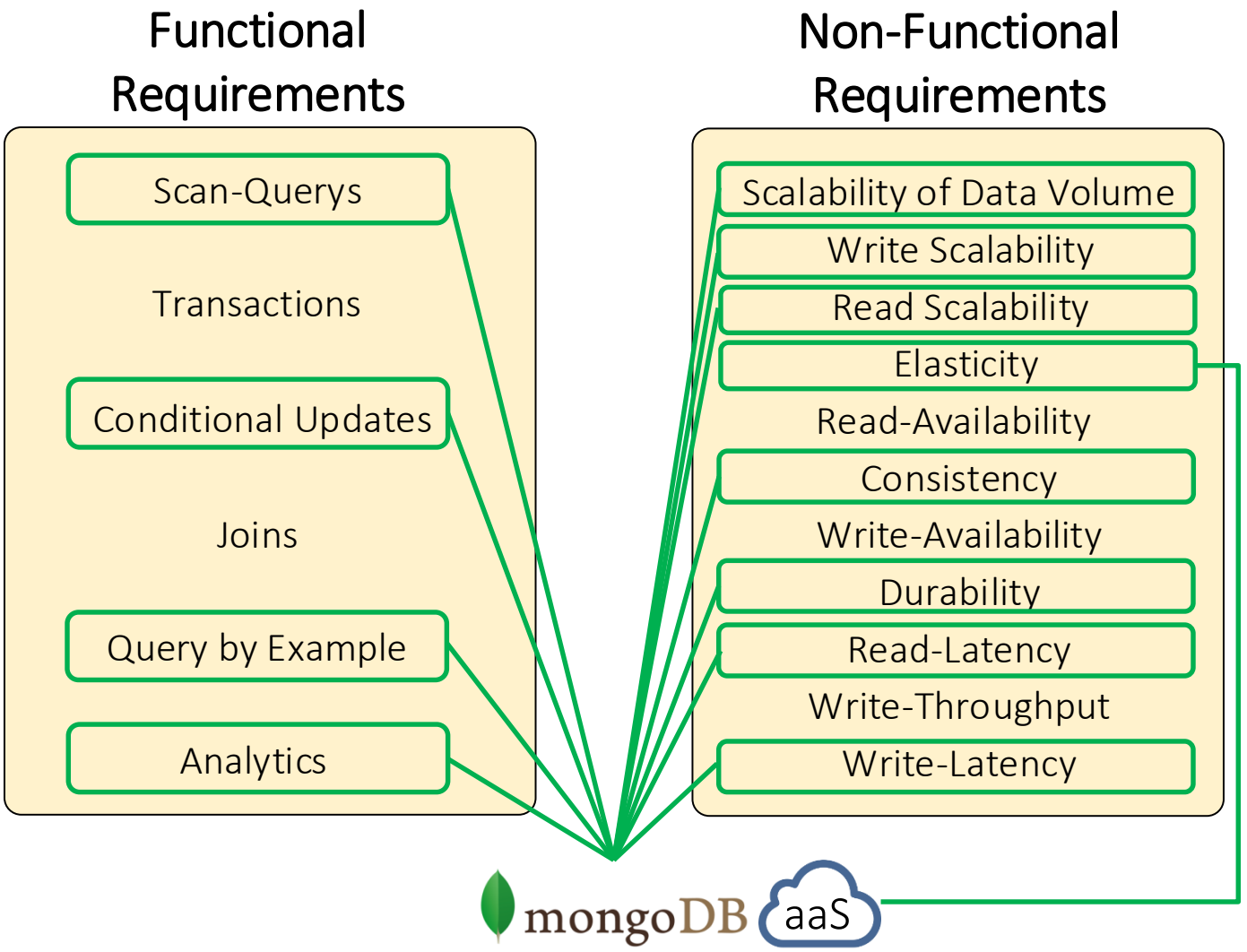
Non-Functional Requirements

- Scalability of Data Volume
- Write Scalability
- Read Scalability
- Elasticity
- Read-Availability
- Consistency
- Write-Availability
- Durability
- Read-Latency
- Write-Throughput
- Write-Latency

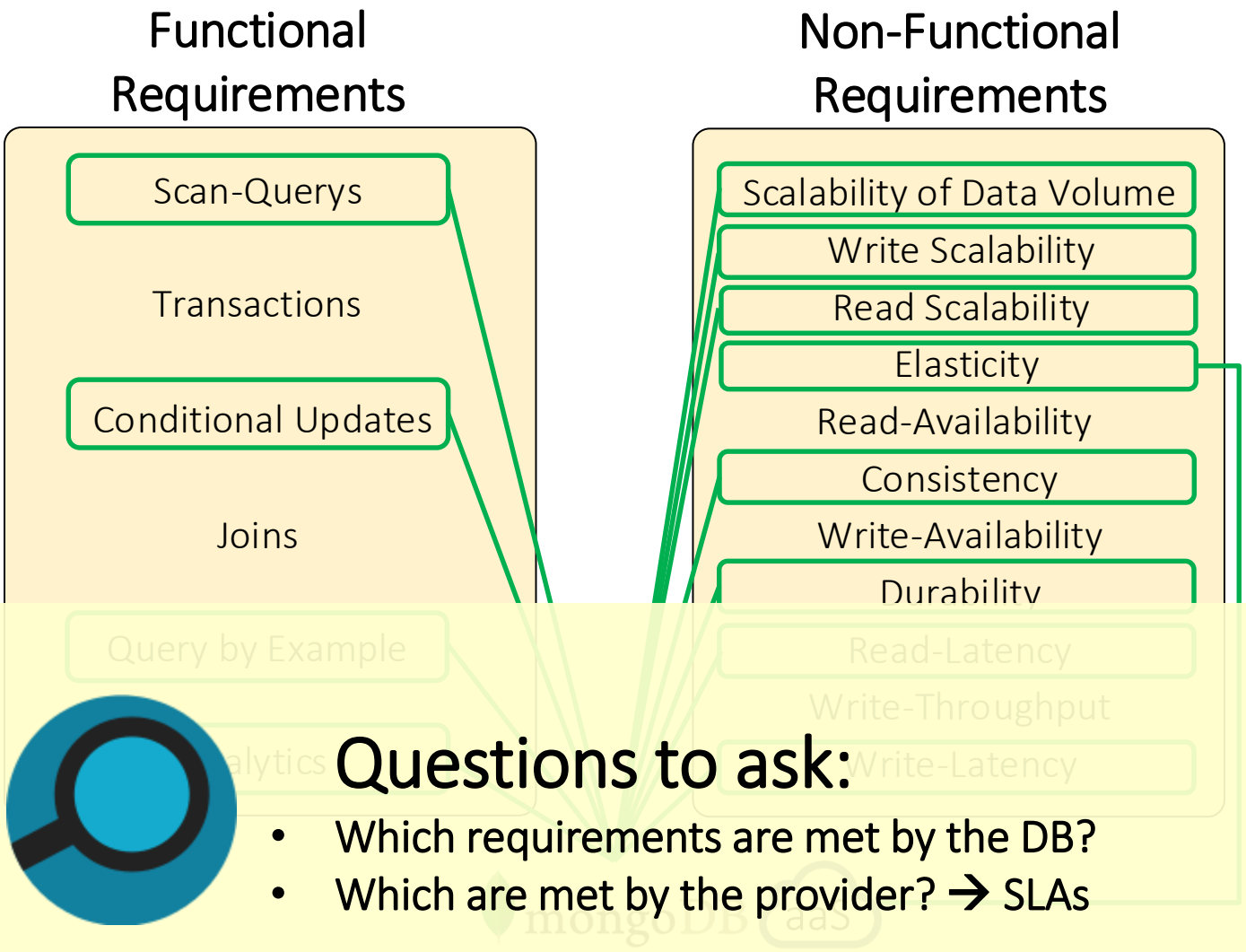
DBaaS: General Considerations



DBaaS: General Considerations



DBaaS: General Considerations



Questions to ask:

- Which requirements are met by the DB?
- Which are met by the provider? → SLAs

Outline

What are Cloud Databases?

Cloud Databases in the wild

Research Perspectives

Wrap-up and literature

Examples of different cloud database systems:

- Cloud-deployed
- Managed DBMS
  - SQL
  - NoSQL
- Proprietary
- BaaS

Cloud-Deployed DB

- **Idea**: Run (mostly) unmodified DB on IaaS

- Method I: DIY

- Method II: Deployment Tools

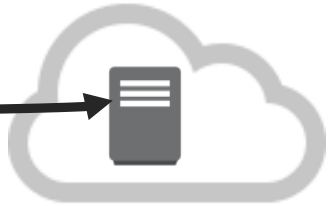
- Method III: Marketplaces

Cloud-Deployed DB

- **Idea**: Run (mostly) unmodified DB on IaaS

- Method I: DIY

1. **Provision** VM(s)



- Method II: Deployment Tools

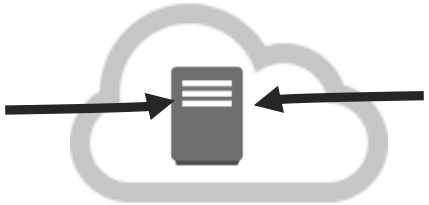
- Method III: Marketplaces

Cloud-Deployed DB

- **Idea**: Run (mostly) unmodified DB on IaaS

- Method I: DIY

1. **Provision** VM(s)



2. **Install** DBMS (manual, script, Chef, Puppet)

- Method II: Deployment Tools

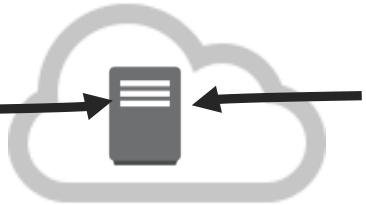
- Method III: Marketplaces

Cloud-Deployed DB

- **Idea**: Run (mostly) unmodified DB on IaaS

- Method I: DIY

1. **Provision** VM(s)



2. **Install** DBMS (manual, script, Chef, Puppet)

- Method II: Deployment Tools

```
> whirr launch-cluster --config
hbase.properties
```

Login, cluster-size etc.

Amazon EC2

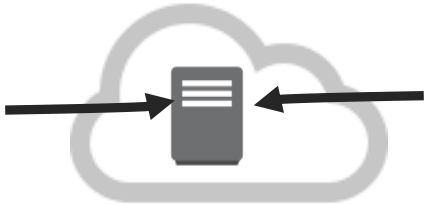
- Method III: Marketplaces

Cloud-Deployed DB

- **Idea**: Run (mostly) unmodified DB on IaaS

- Method I: DIY

1. **Provision** VM(s)



2. **Install** DBMS (manual, script, Chef, Puppet)

- Method II: Deployment Tools

```
> whirr launch-cluster --config
hbase.properties
```

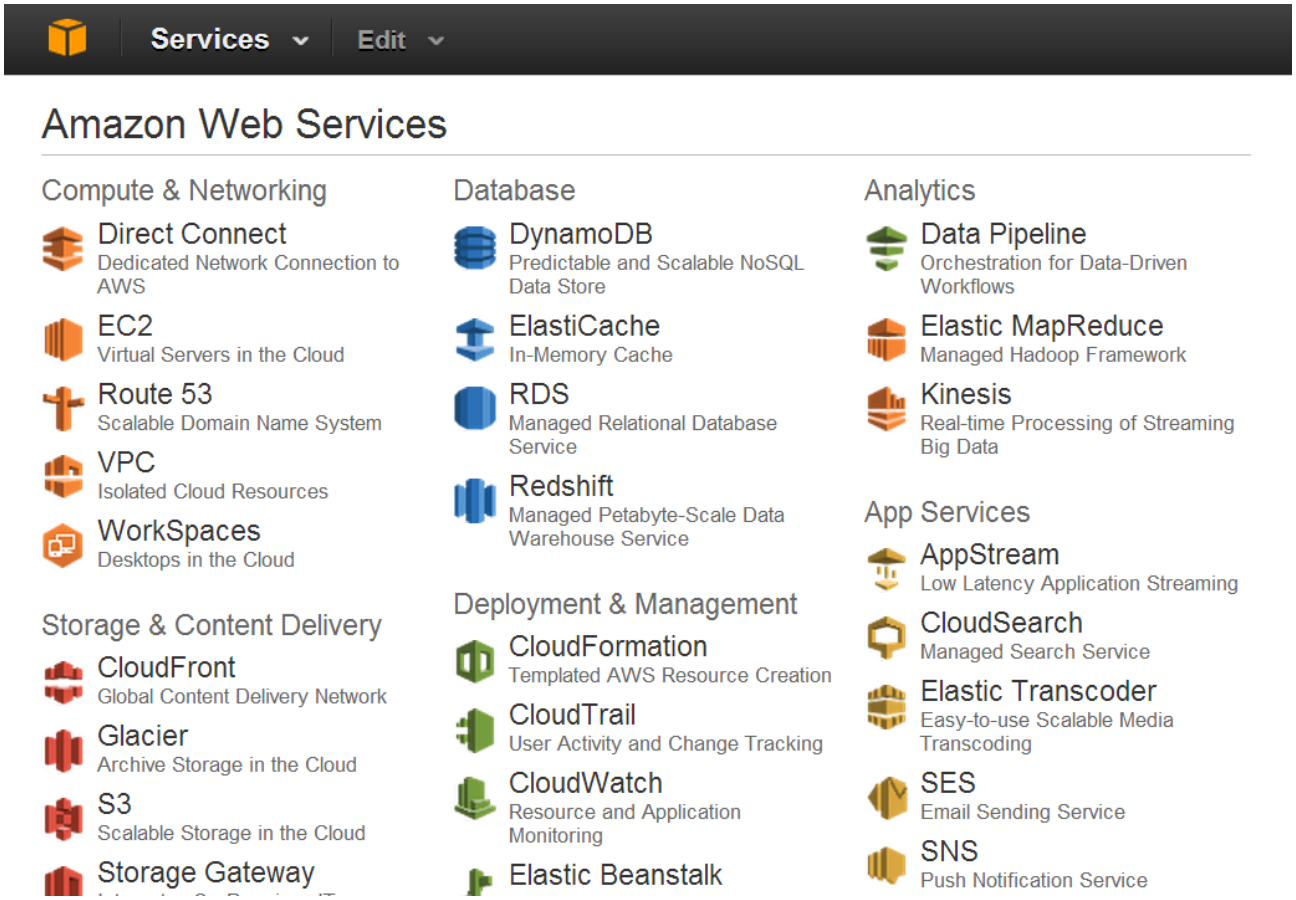
Login, cluster-size etc.

Amazon EC2

- Method III: Marketplaces

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

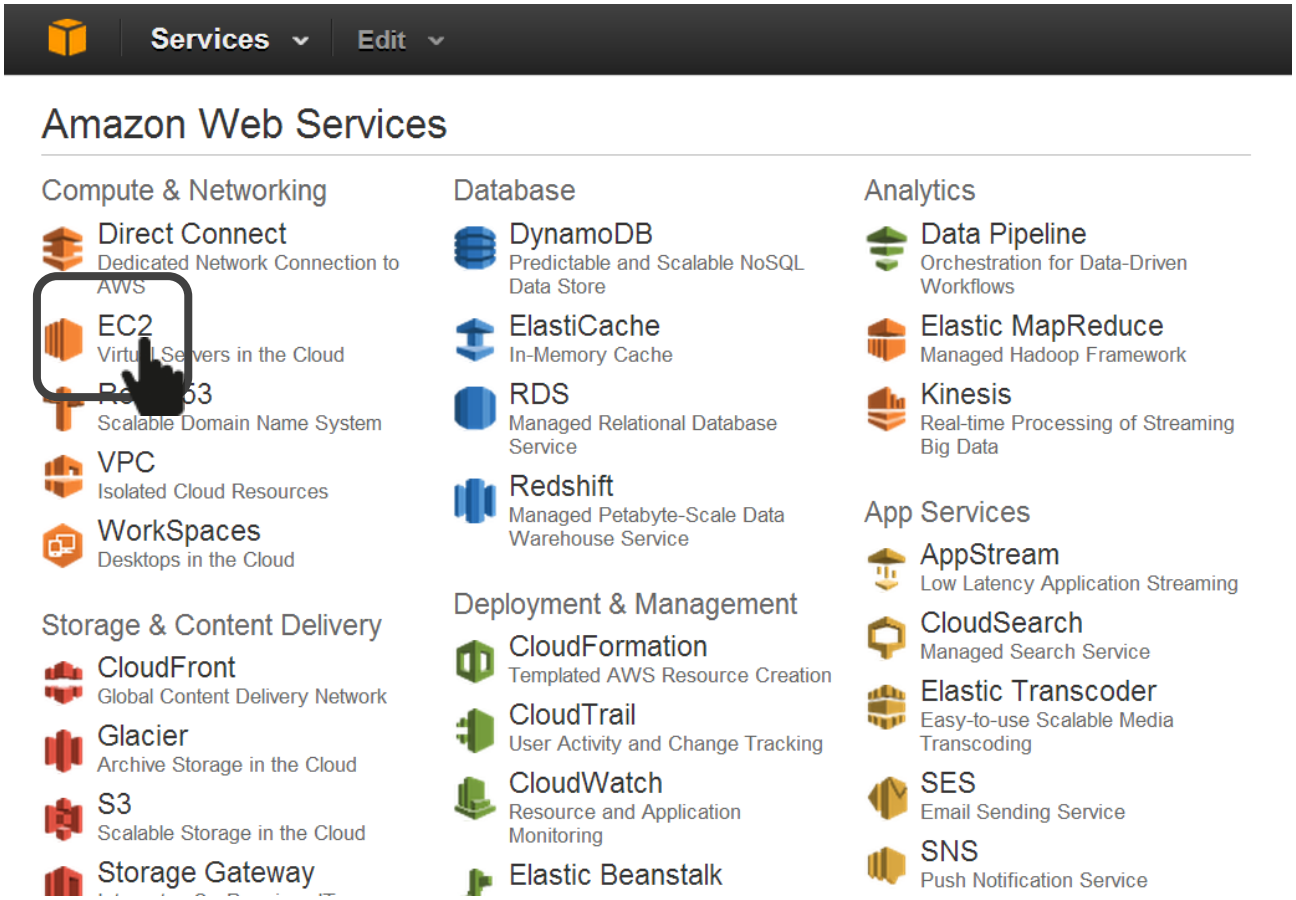


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

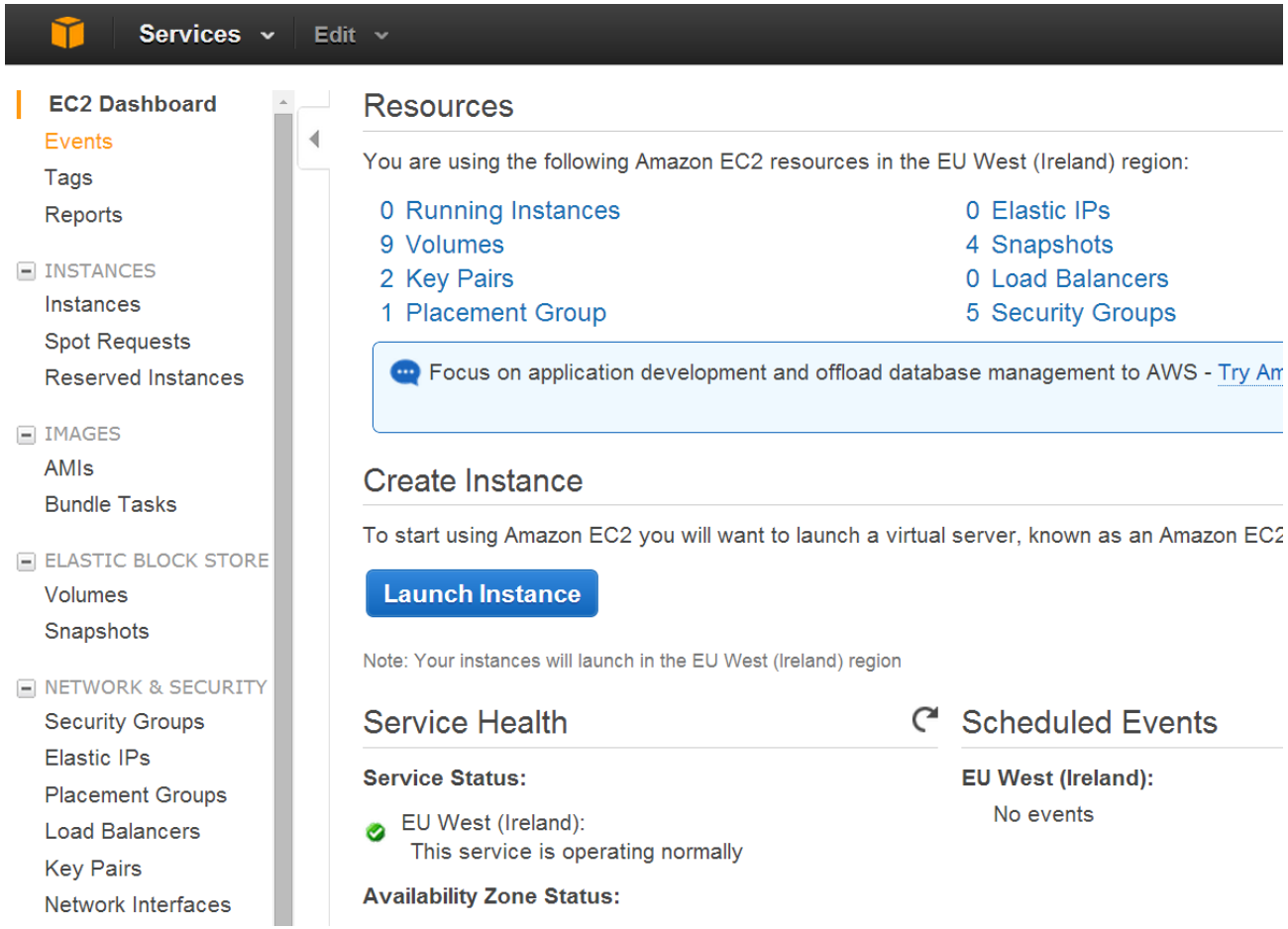


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

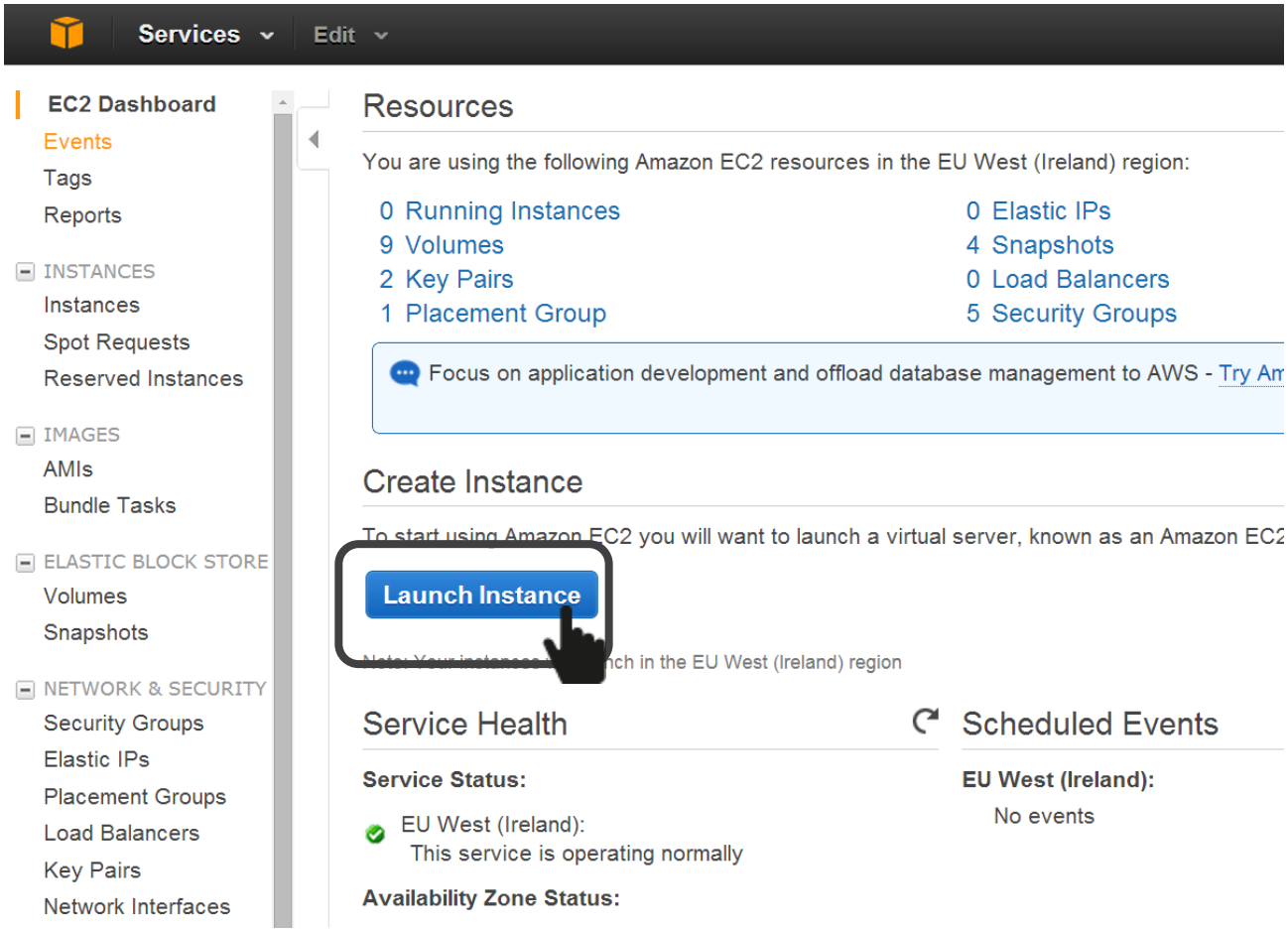


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

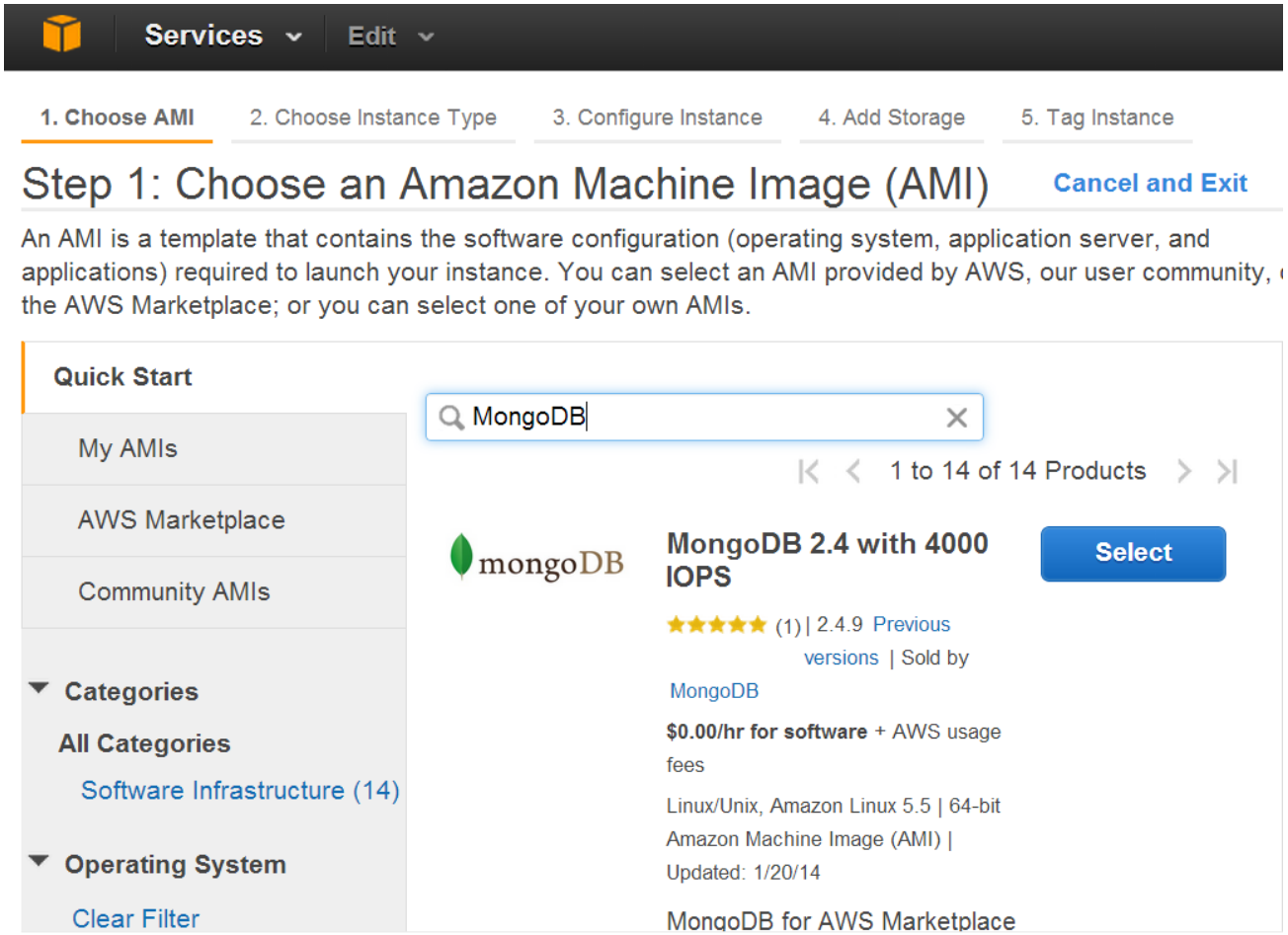


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

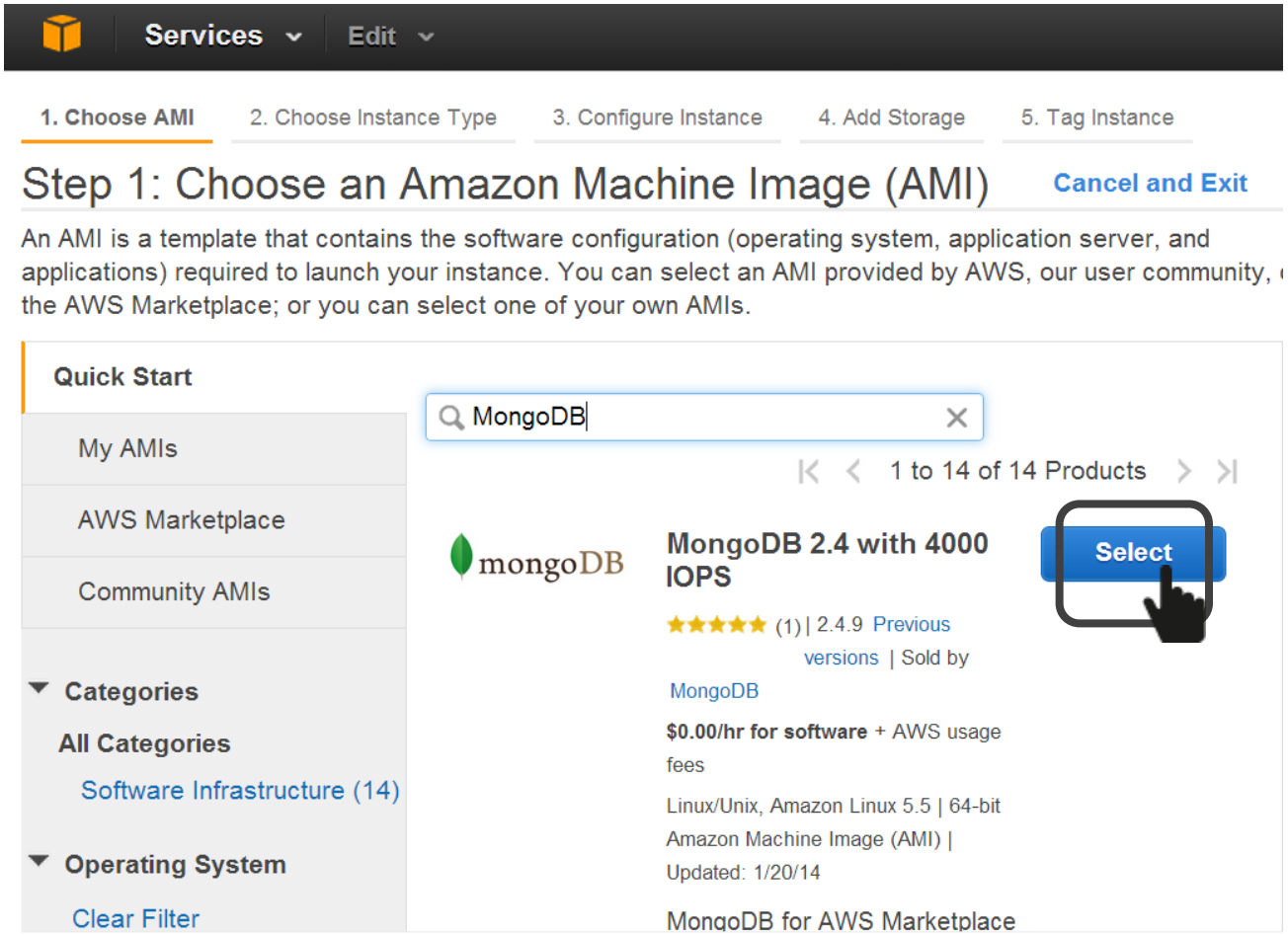


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

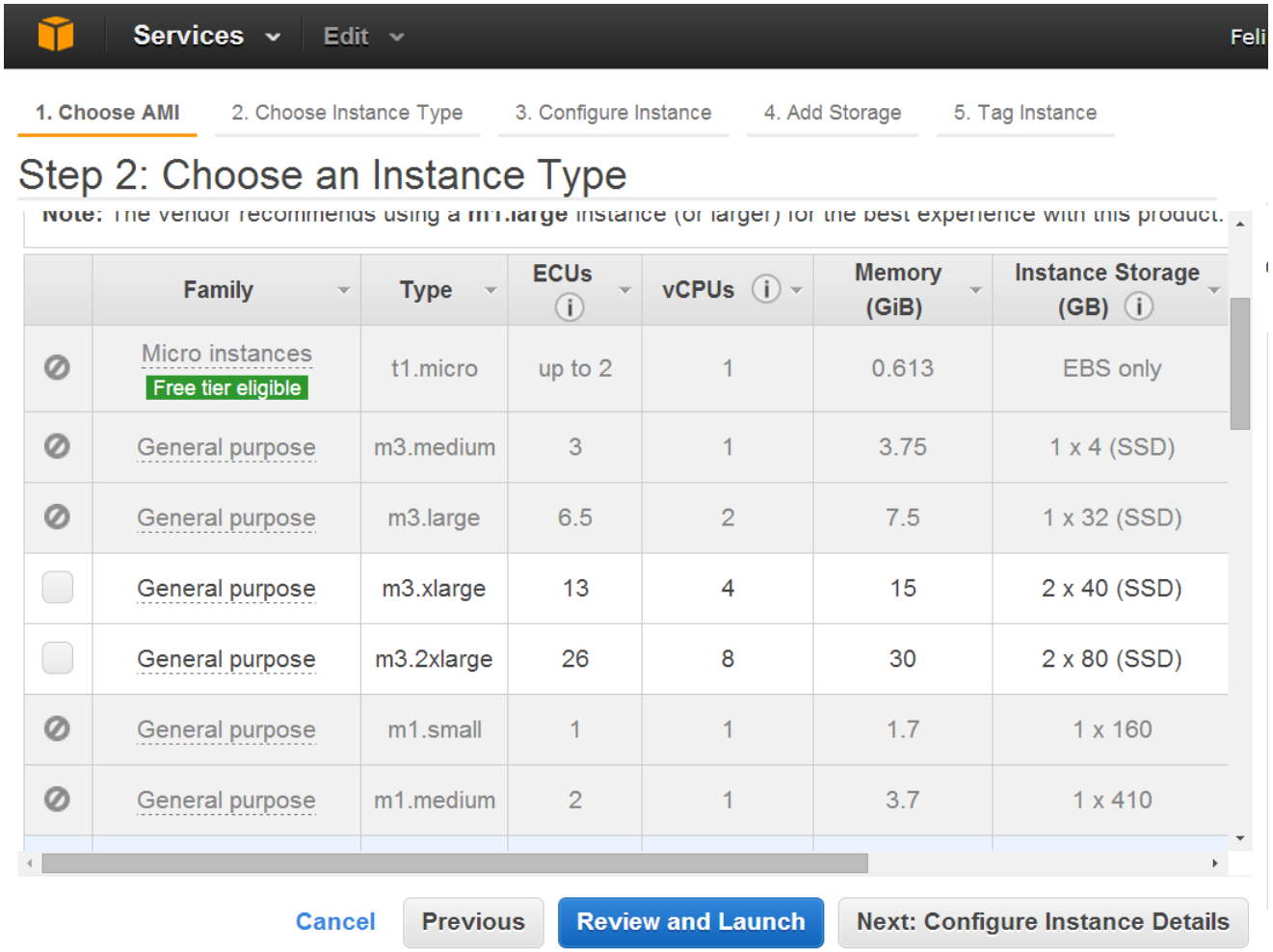


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS

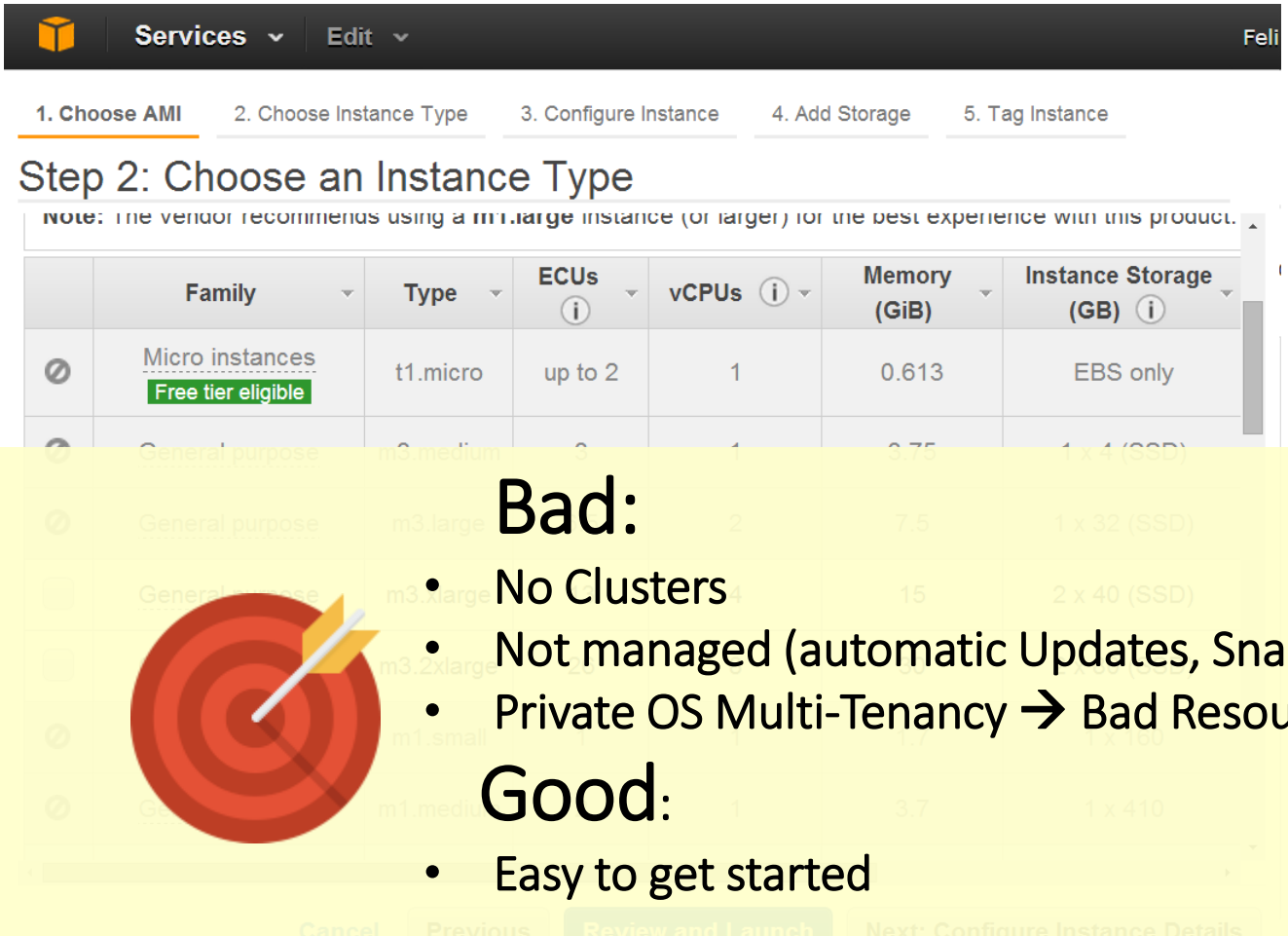


AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

AWS Marketplace

- **Idea**: Run preconfigured DB on IaaS



AWS Marketplace

AWS Marketplace
Model:
Cloud-Deployed
Pricing:
Instance + Volume + License
Underlying DB:
Choosable
API:
DB-specific

Bad:

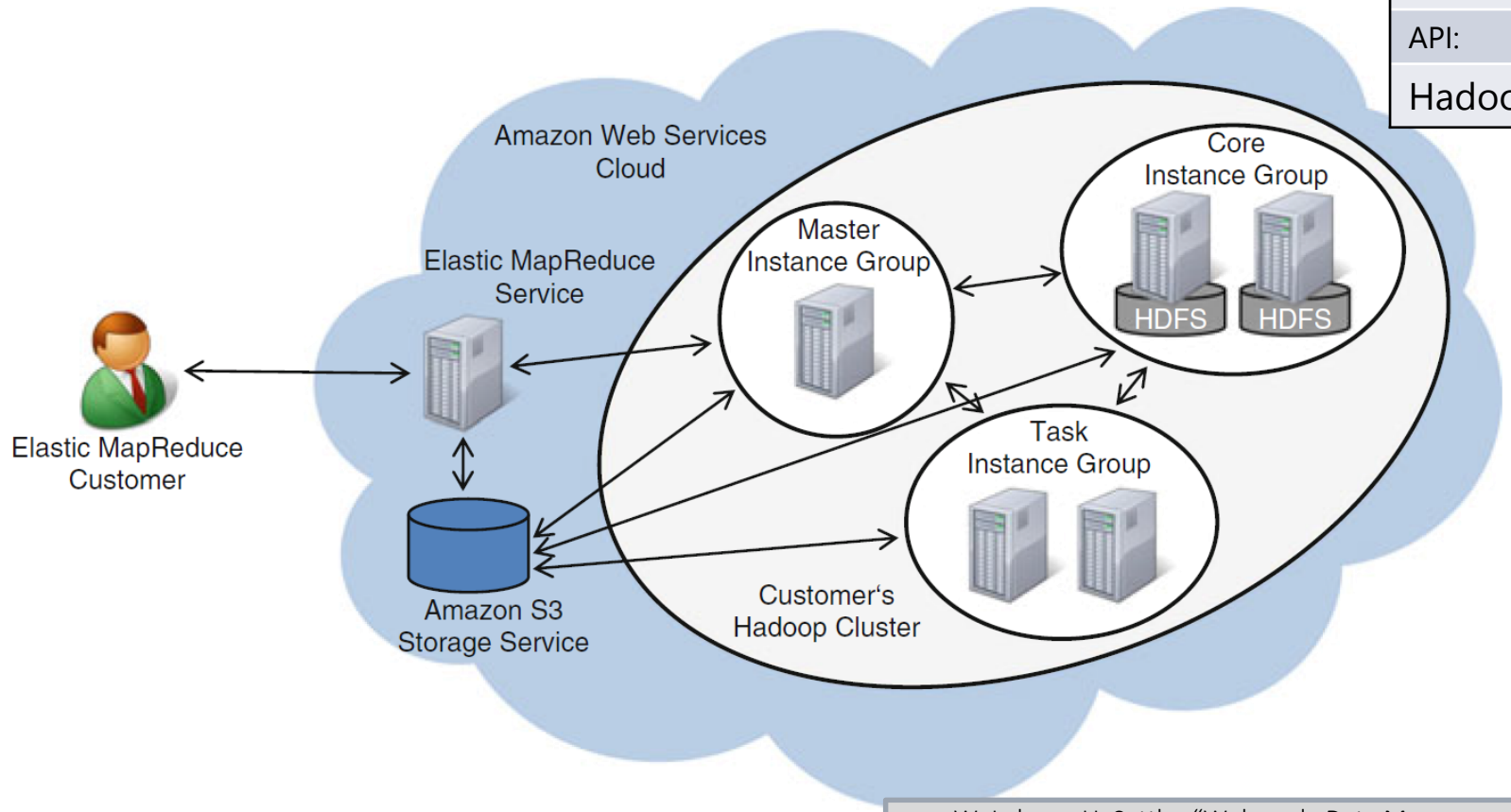
- No Clusters
- Not managed (automatic Updates, Snapshots, etc.)
- Private OS Multi-Tenancy → Bad Resource Usage

Good:

- Easy to get started

Amazon Elastic MapReduce

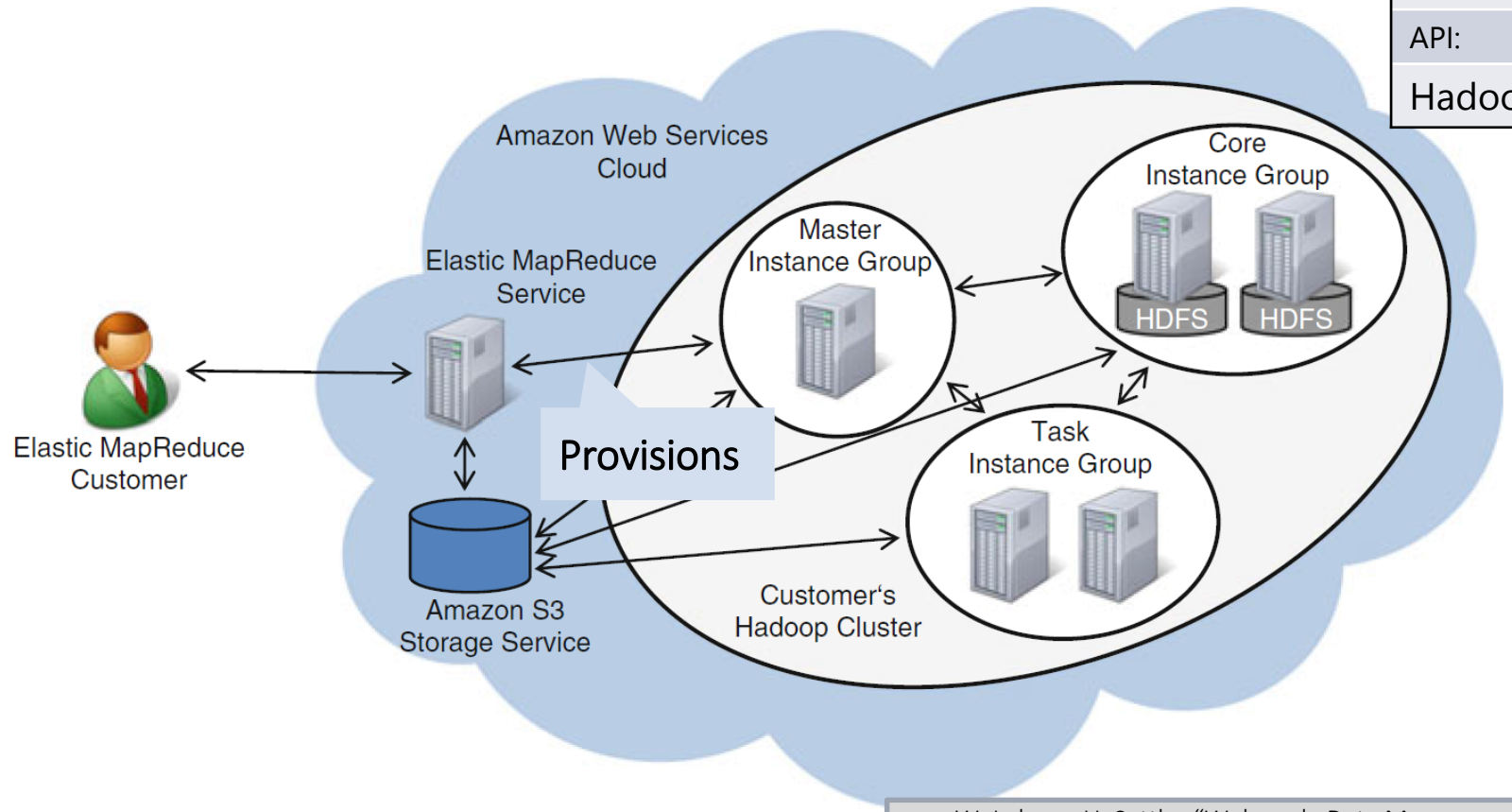
EMR
Model:
Analytics-aaS
Pricing:
Infrastructure
API:
Hadoop



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

Amazon Elastic MapReduce

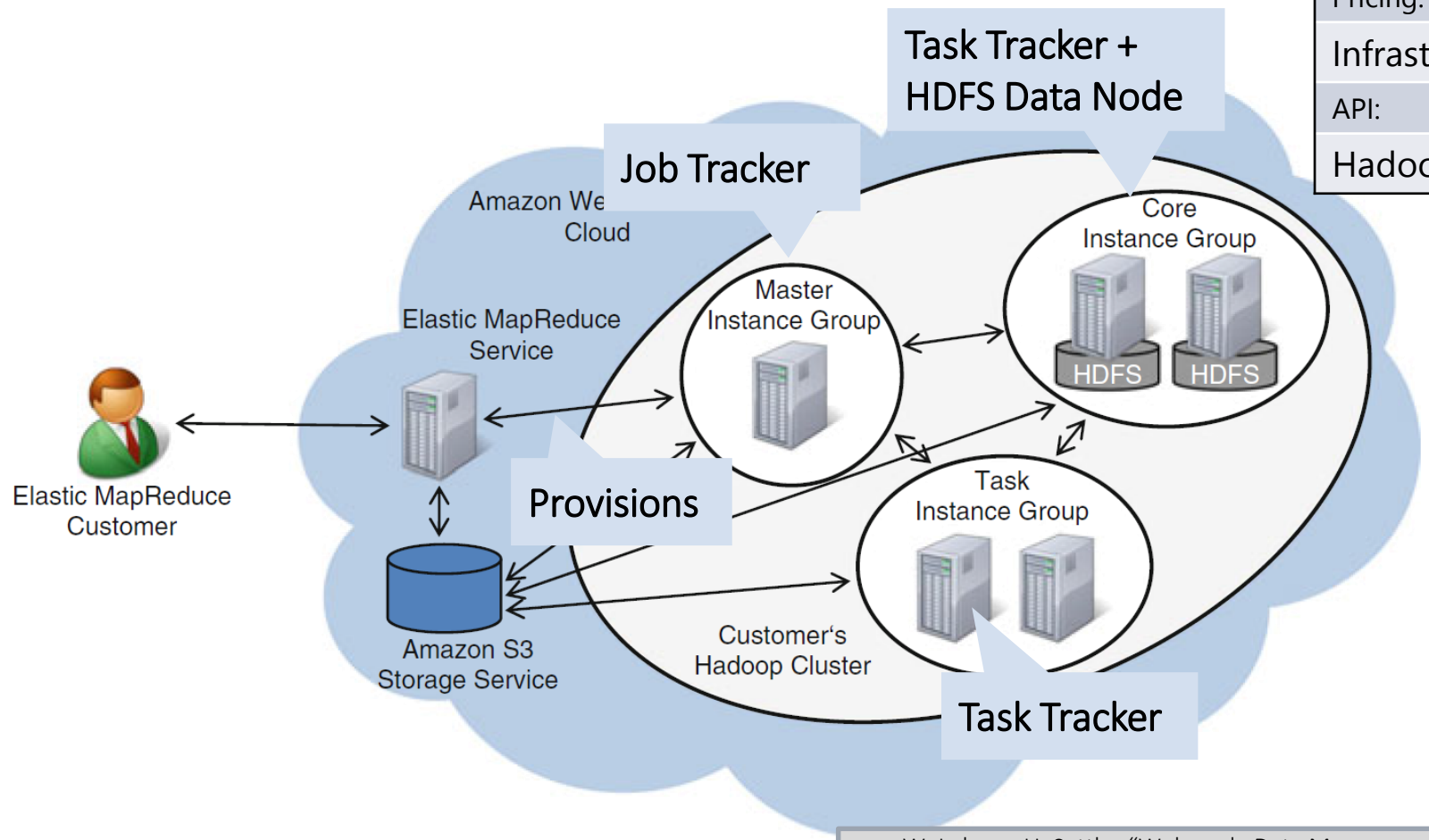
EMR
Model:
Analytics-aaS
Pricing:
Infrastructure
API:
Hadoop



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

Amazon Elastic MapReduce

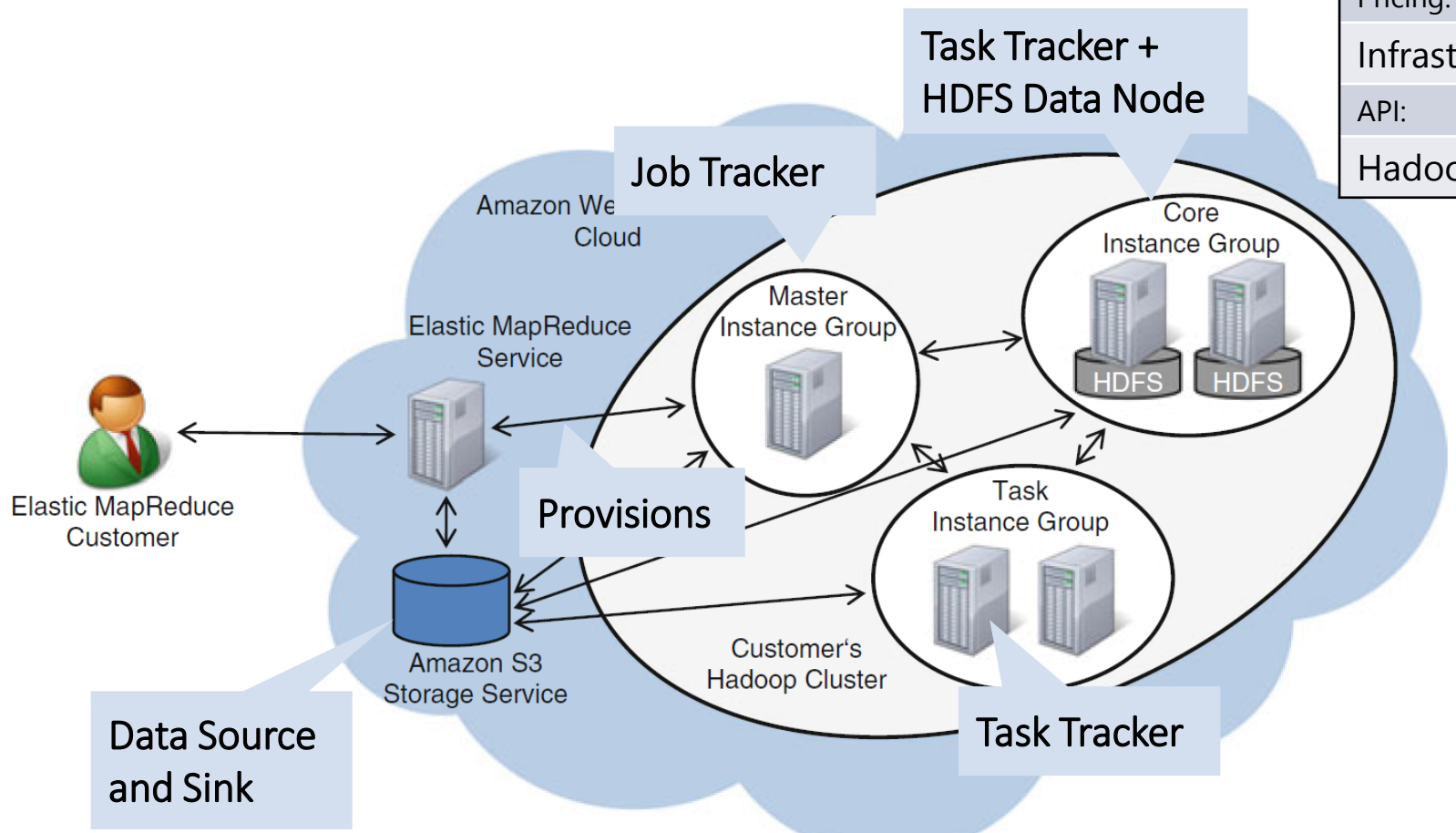
EMR
Model:
Analytics-aaS
Pricing:
Infrastructure
API:
Hadoop



W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

Amazon Elastic MapReduce

EMR
Model:
Analytics-aaS
Pricing:
Infrastructure
API:
Hadoop



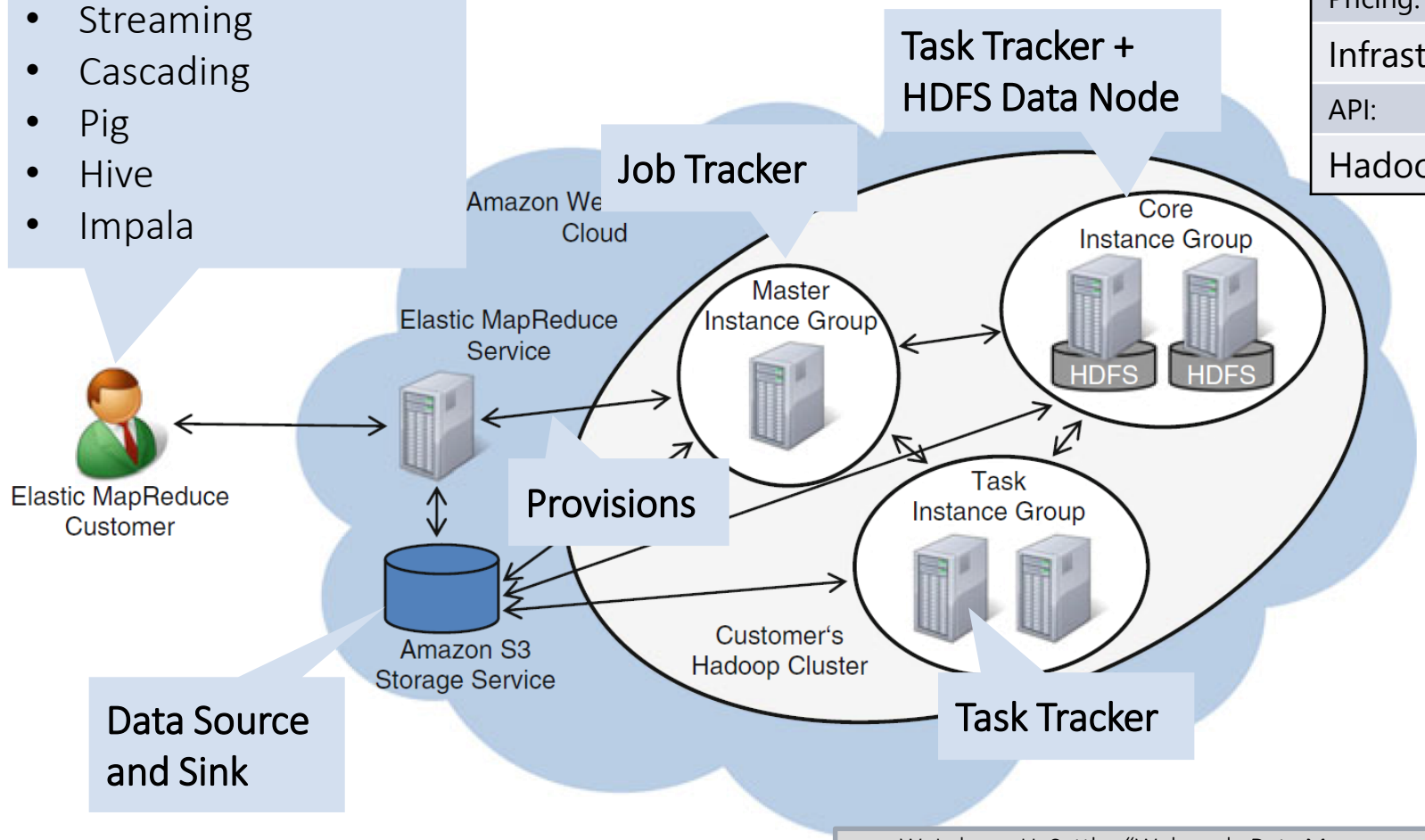
W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

Amazon Elastic MapReduce

EMR
Model:
Analytics-aaS
Pricing:
Infrastructure
API:
Hadoop

Submits Hadoop Jobs as:

- JAR
- Streaming
- Cascading
- Pig
- Hive
- Impala



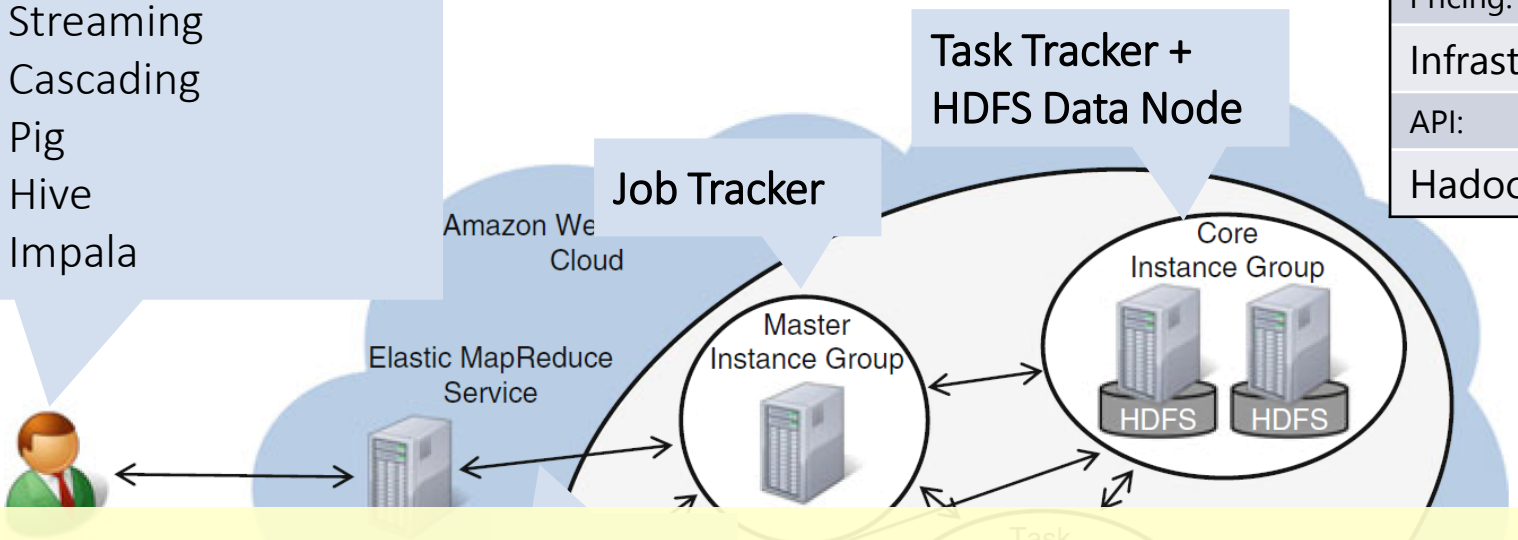
W. Lehner, U. Sattler "Web-scale Data Management for the Cloud" Springer, 2013

Amazon Elastic MapReduce

EMR
Model:
Analytics-aaS
Pricing:
Infrastructure
API:
Hadoop

Submits Hadoop Jobs as:

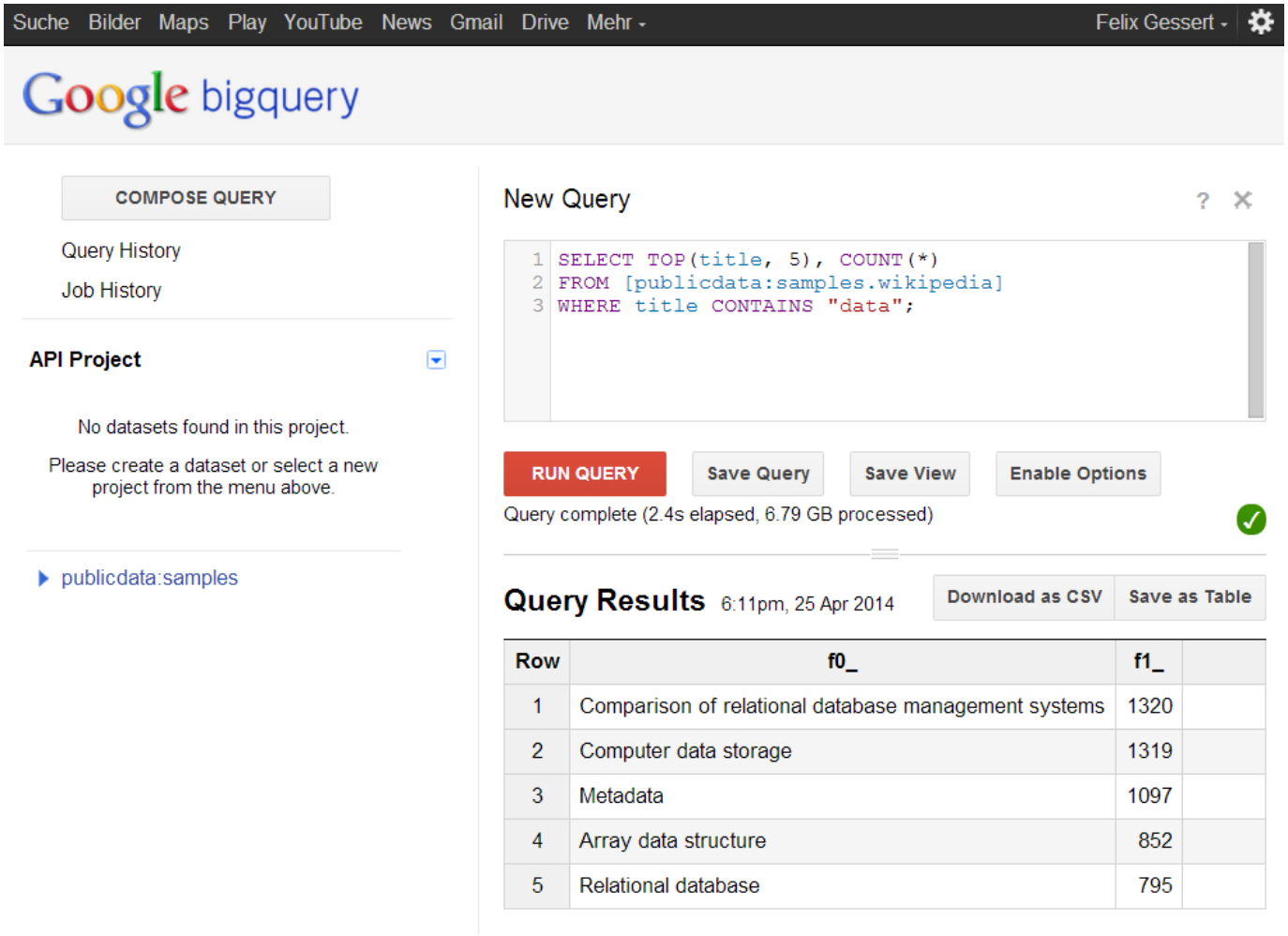
- JAR
- Streaming
- Cascading
- Pig
- Hive
- Impala



- **No data locality** with S3
- **AWS Import/Export:** send your HDD
- **HBase** Integration
- Compatible with **Spot** and Reserved Instances
- Similar: Azure HDInsight

Google BigQuery

- **Idea**: Web-scale analysis of nested data

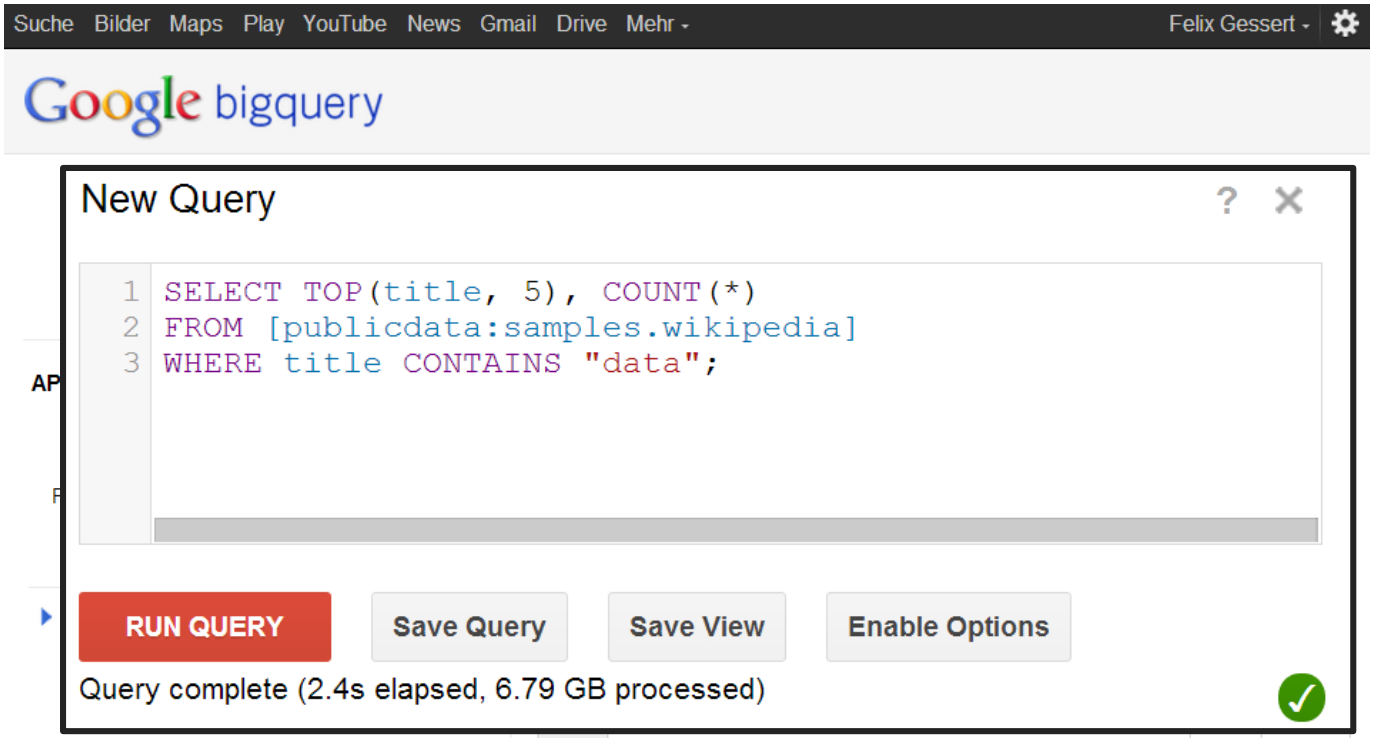


Google BigQuery

BigQuery
Model:
Analytics-aaS
Pricing:
Storage + GBs Processed
API:
REST

Google BigQuery

- **Idea**: Web-scale analysis of nested data



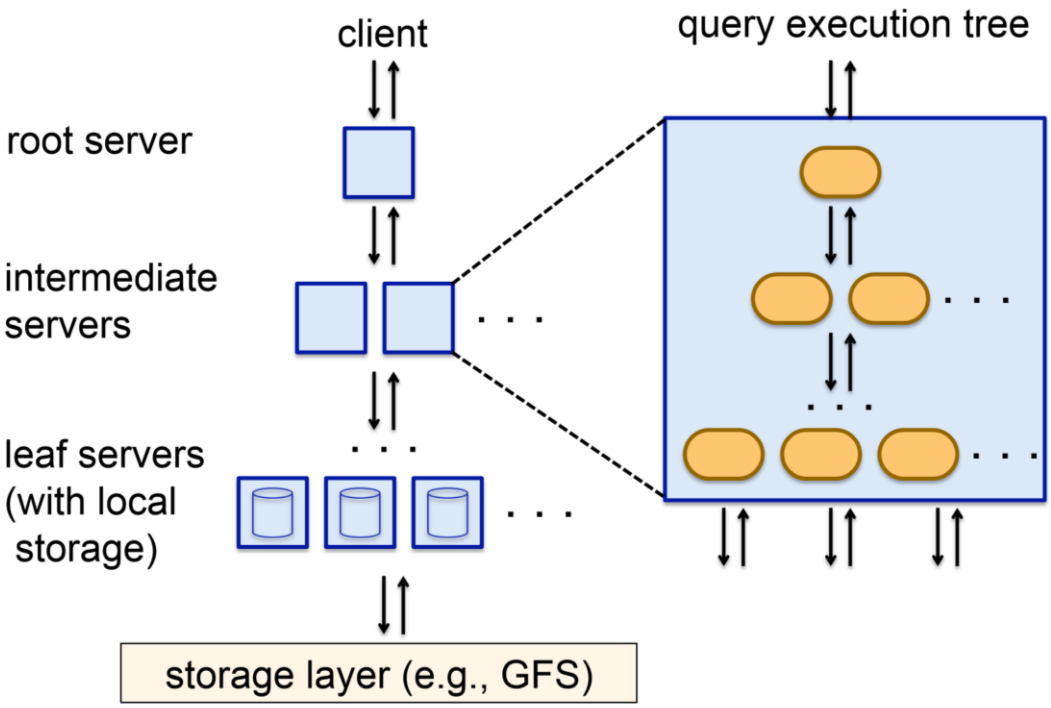
1	Comparison of relational database management systems	1320	
2	Computer data storage	1319	
3	Metadata	1097	
4	Array data structure	852	
5	Relational database	795	

Google BigQuery

BigQuery
Model:
Analytics-aaS
Pricing:
Storage + GBs Processed
API:
REST

Google BigQuery

- **Idea**: Web-scale analysis of nested data



Google BigQuery

BigQuery
Model:
Analytics-aaS
Pricing:
Storage + GBs Processed
API:
REST

Dremel

Idea:
Multi-Level execution tree on nested columnar data format (≥100 nodes)

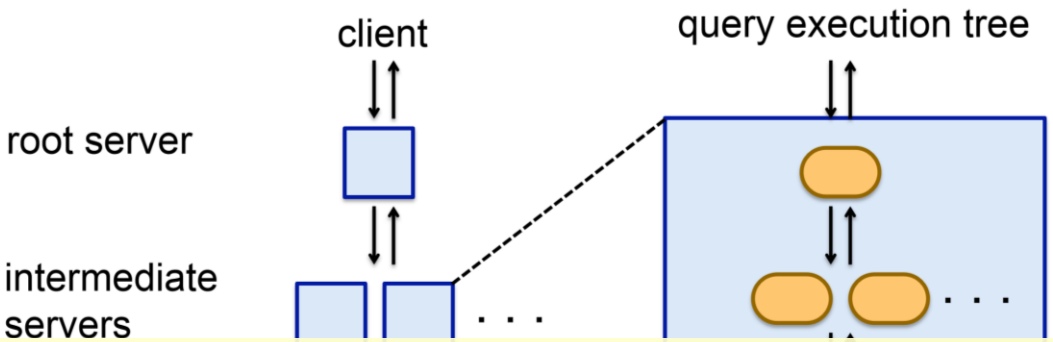
Melnik et al. "Dremel: Interactive analysis of web-scale datasets", VLDB 2010

Google BigQuery

Google BigQuery

BigQuery
Model:
Analytics-aaS
Pricing:
Storage + GBs Processed
API:
REST

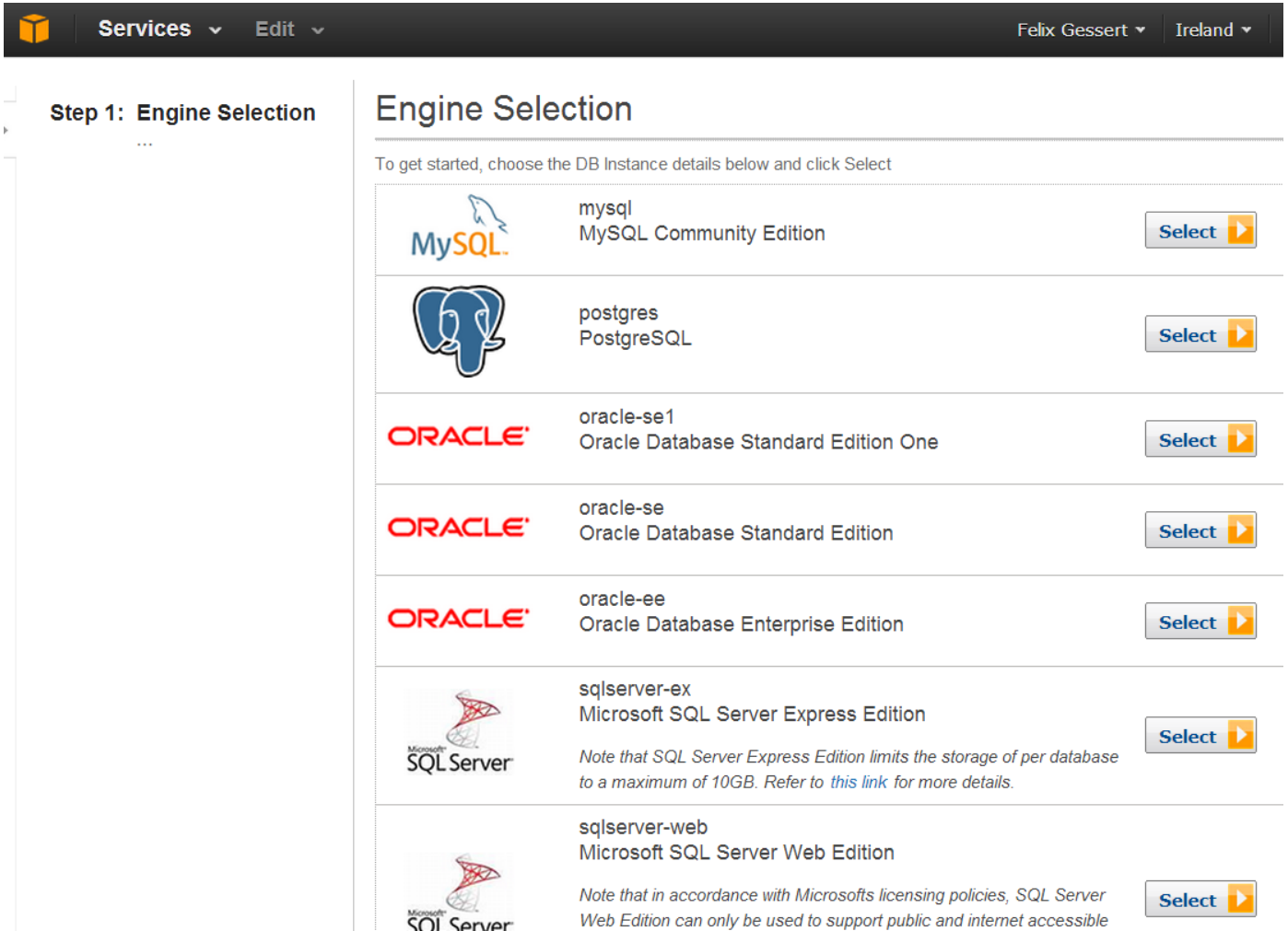
- **Idea**: Web-scale analysis of nested data



- **SLA**: 99.9% uptime / month
- Fundamentally different from relational DWHs and MapReduce
- Design copied by Apache Drill, Impala, Shark

Amazon RDS

- Relational Database Service

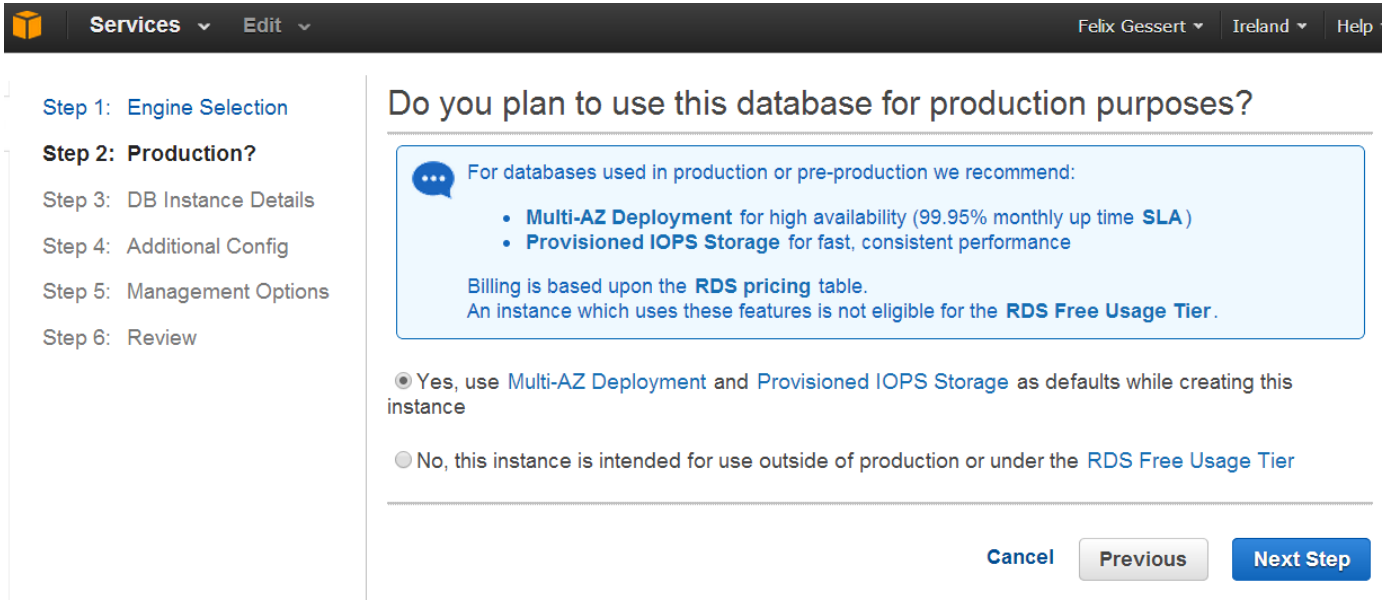


Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service



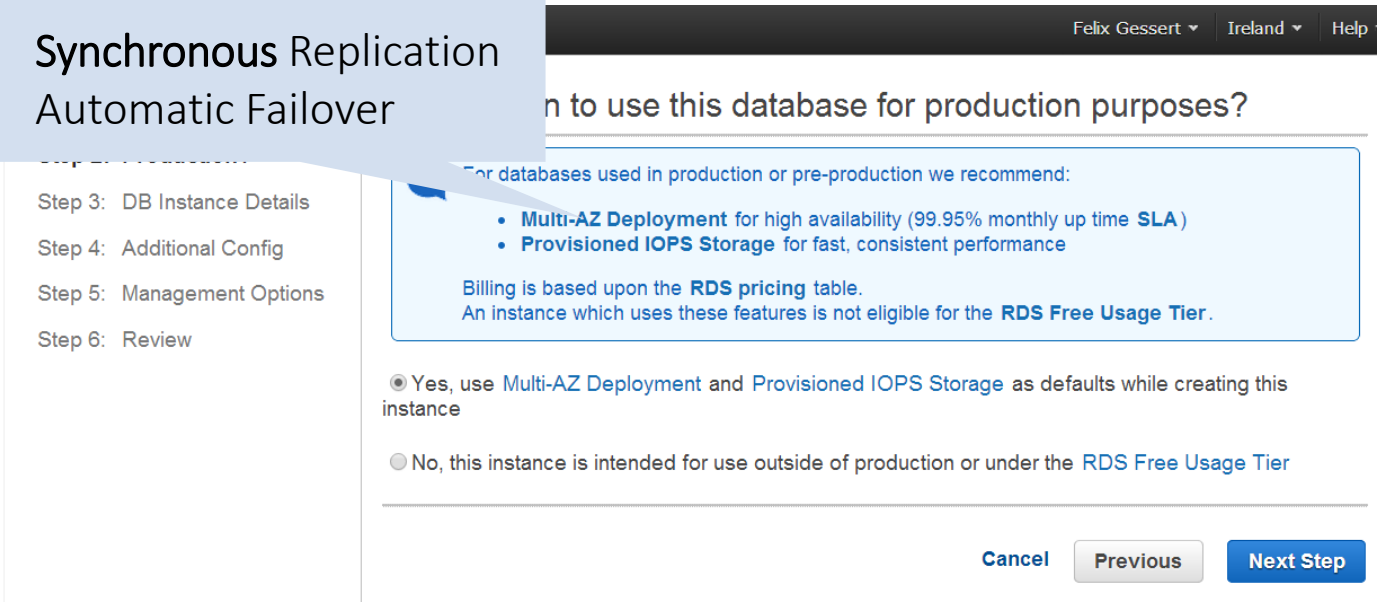
Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service

- **Synchronous** Replication
- Automatic Failover



Amazon RDS

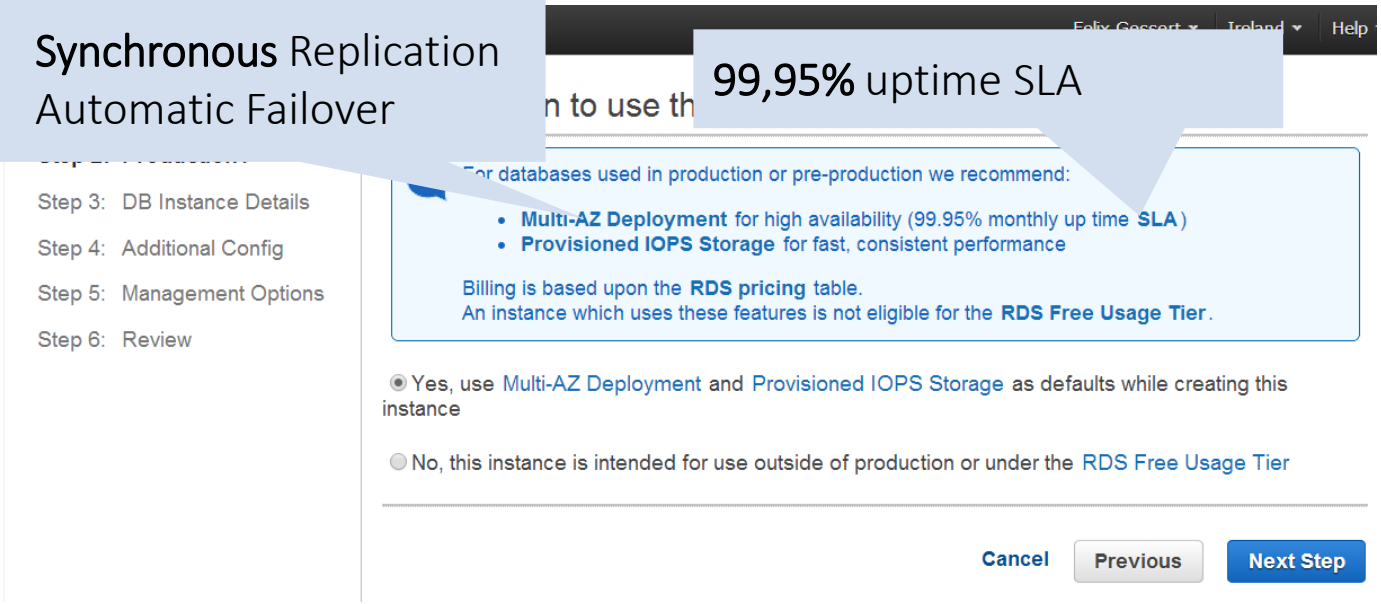
RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service

- **Synchronous** Replication
- Automatic Failover

99,95% uptime SLA



Amazon RDS

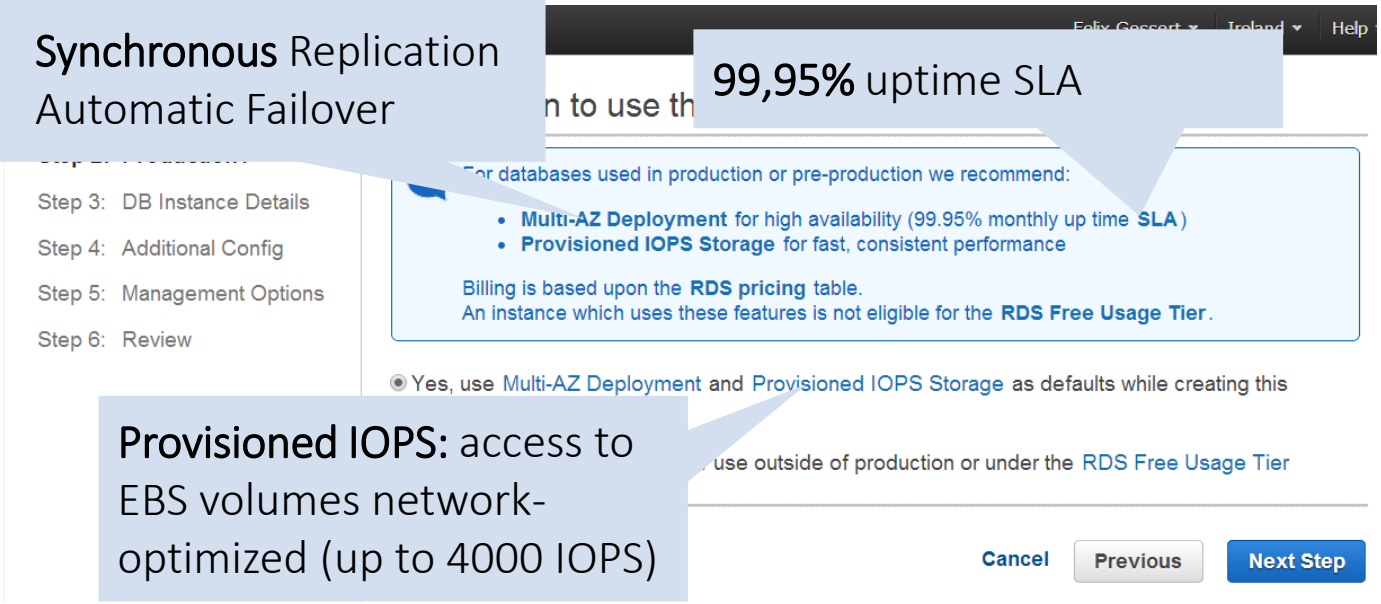
RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service

- **Synchronous** Replication
- Automatic Failover

99,95% uptime SLA



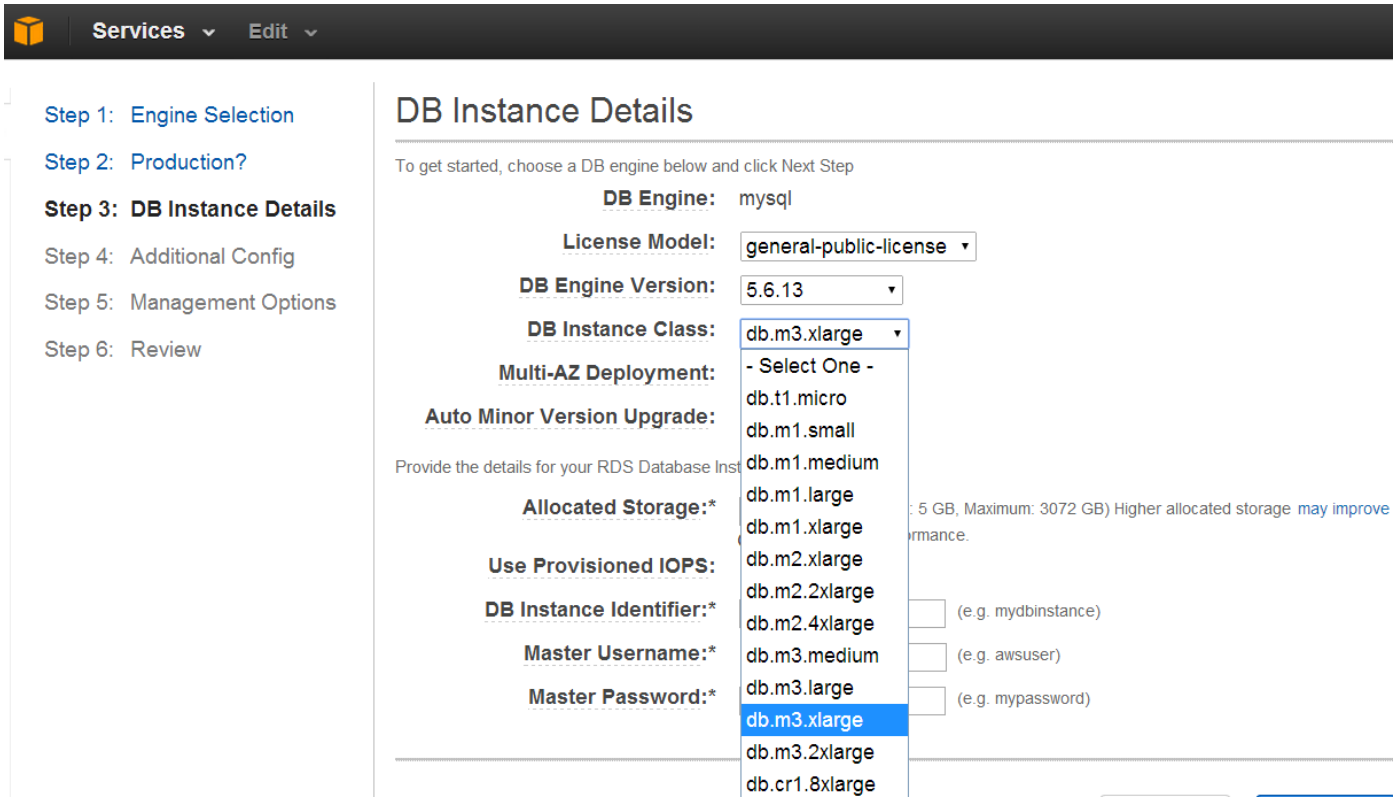
Provisioned IOPS: access to EBS volumes network-optimized (up to 4000 IOPS)

Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational **D**atabase **S**ervice

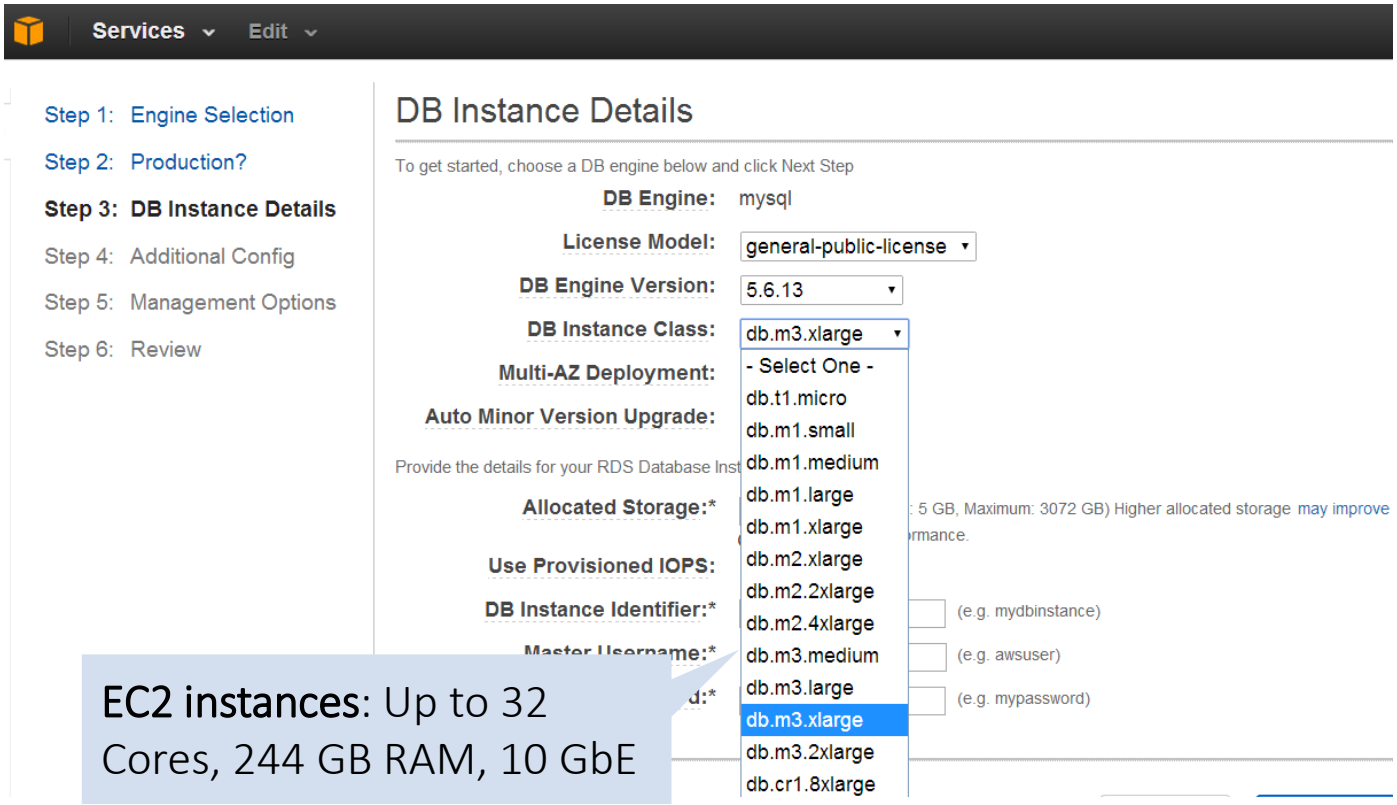


Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service



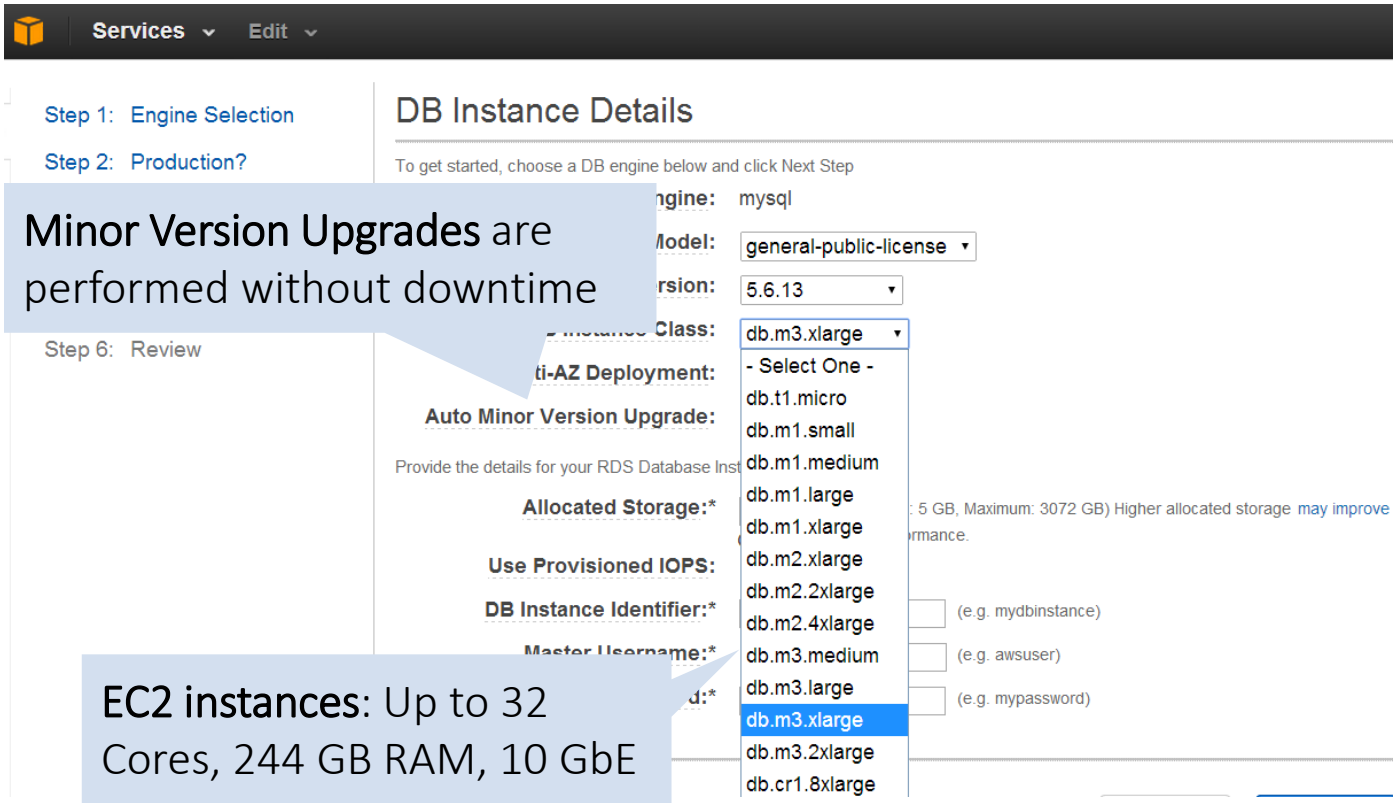
EC2 instances: Up to 32 Cores, 244 GB RAM, 10 GbE

Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service



Minor Version Upgrades are performed without downtime

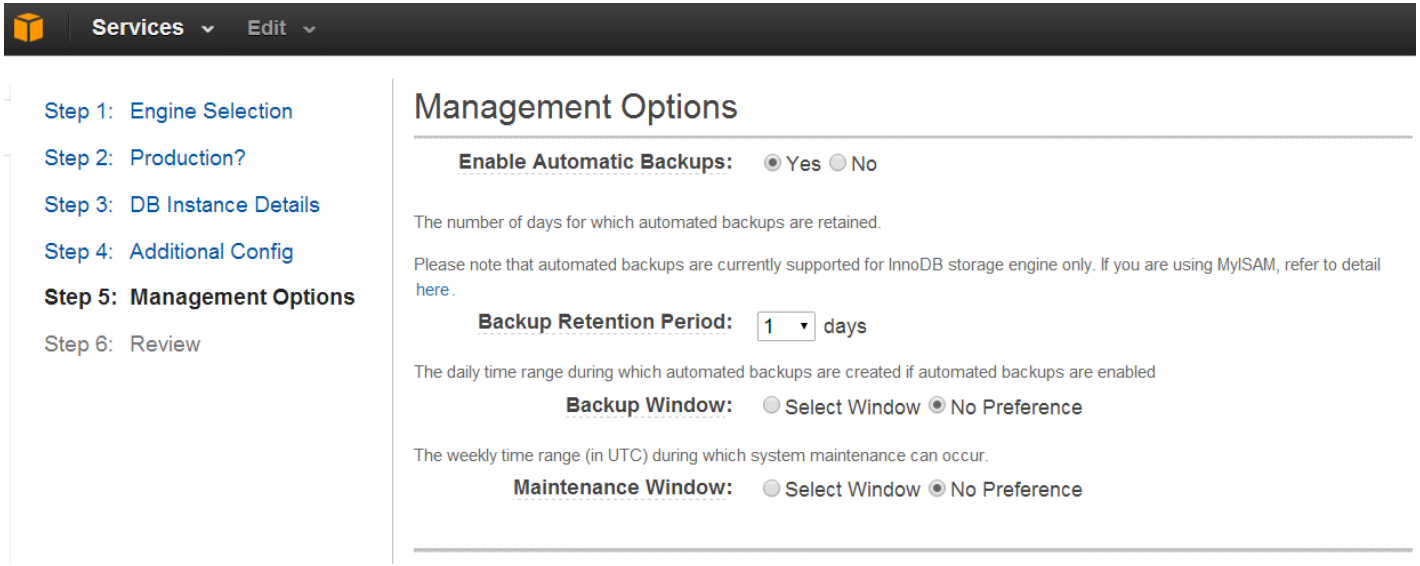
EC2 instances: Up to 32 Cores, 244 GB RAM, 10 GbE

Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational Database Service

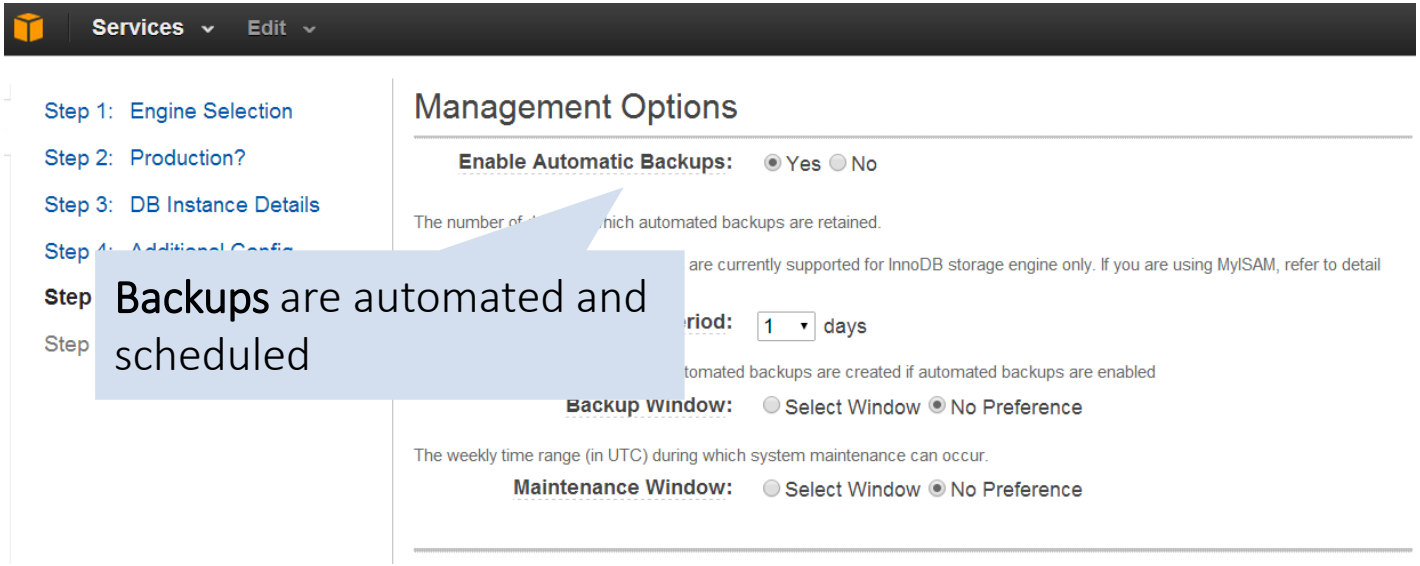


Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

- Relational **D**atabase **S**ervice



Amazon RDS

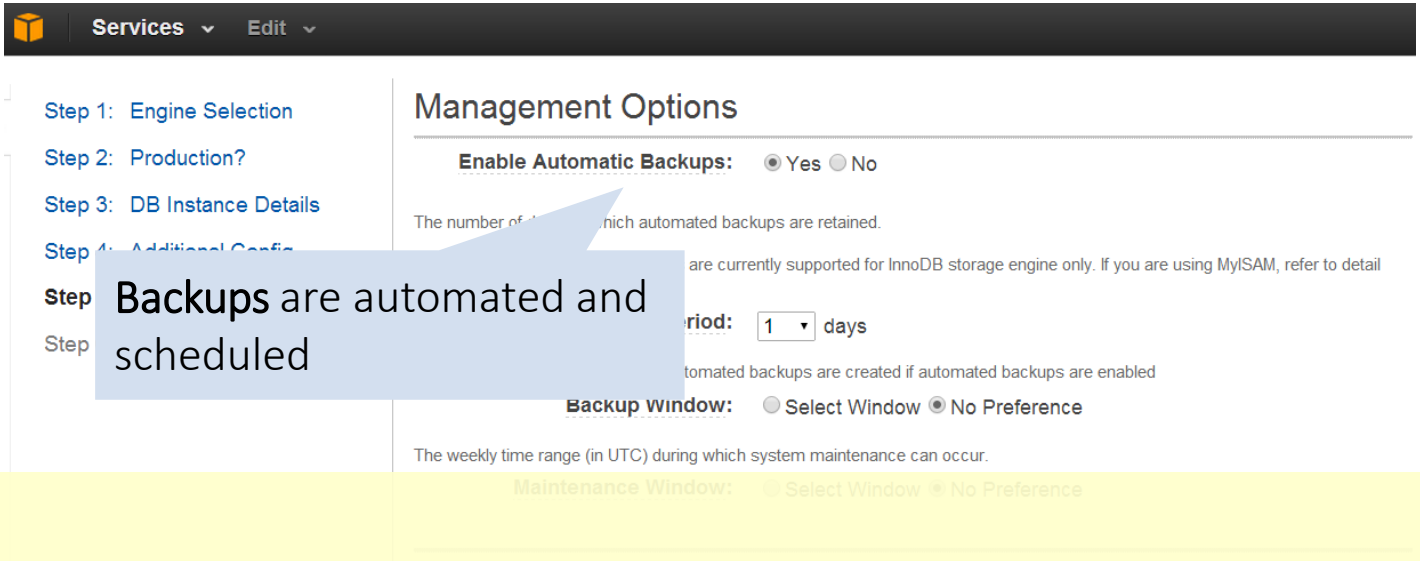
RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

Amazon RDS

Amazon RDS

RDS
Model:
Managed RDBMS
Pricing:
Instance + Volume + License
Underlying DB:
MySQL, Postgres, MSSQL, Oracle
API:
DB-specific

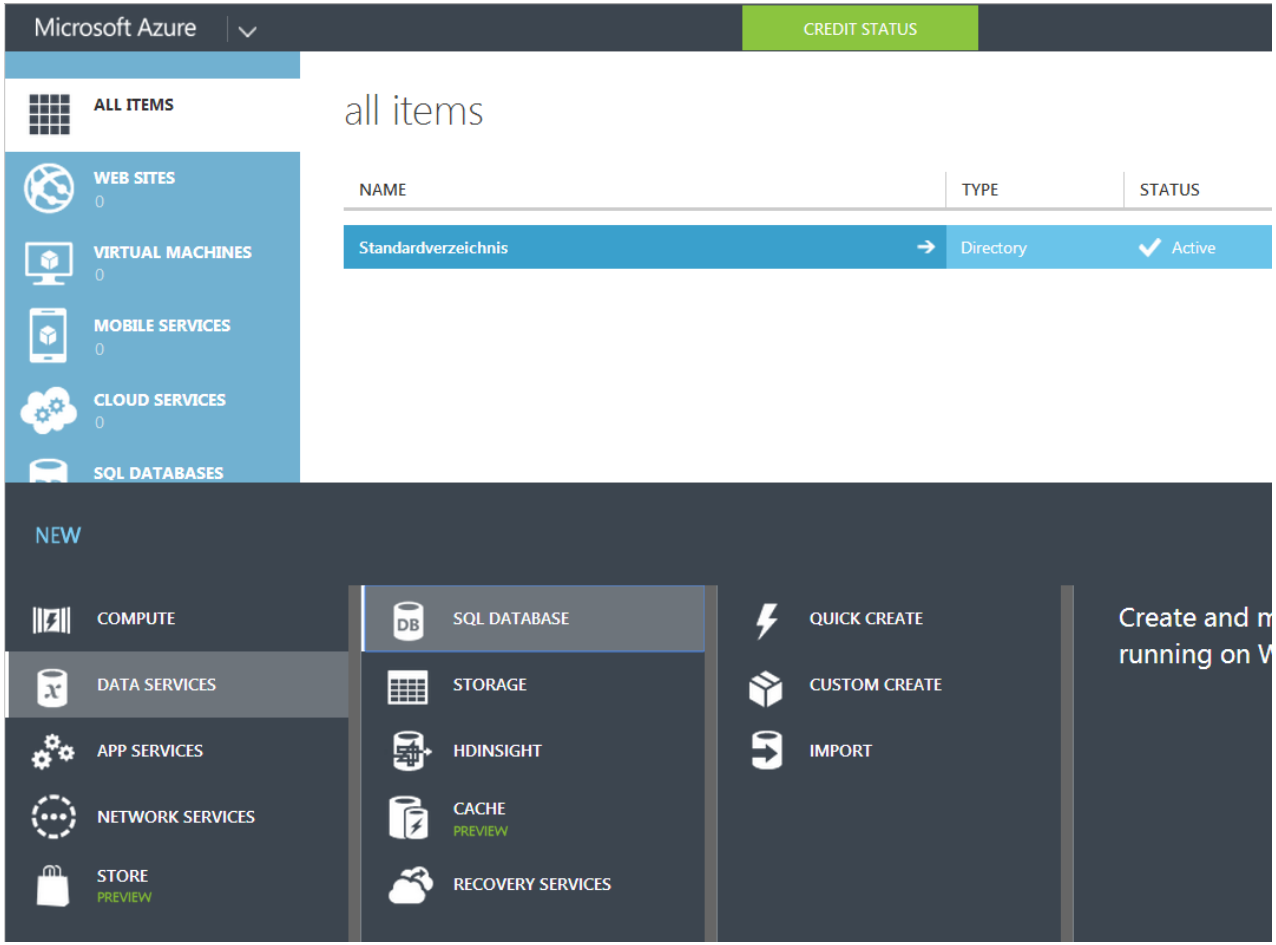
- Relational **D**atabase **S**ervice



- Support for (asynchronous) Read Replicas
- **Administration**: Web-based or SDKs
- Only RDBMSs
- “Analytic Brother“ of RDS: RedShift (PDWH)

Microsoft SQL Azure

- Similar to RDS

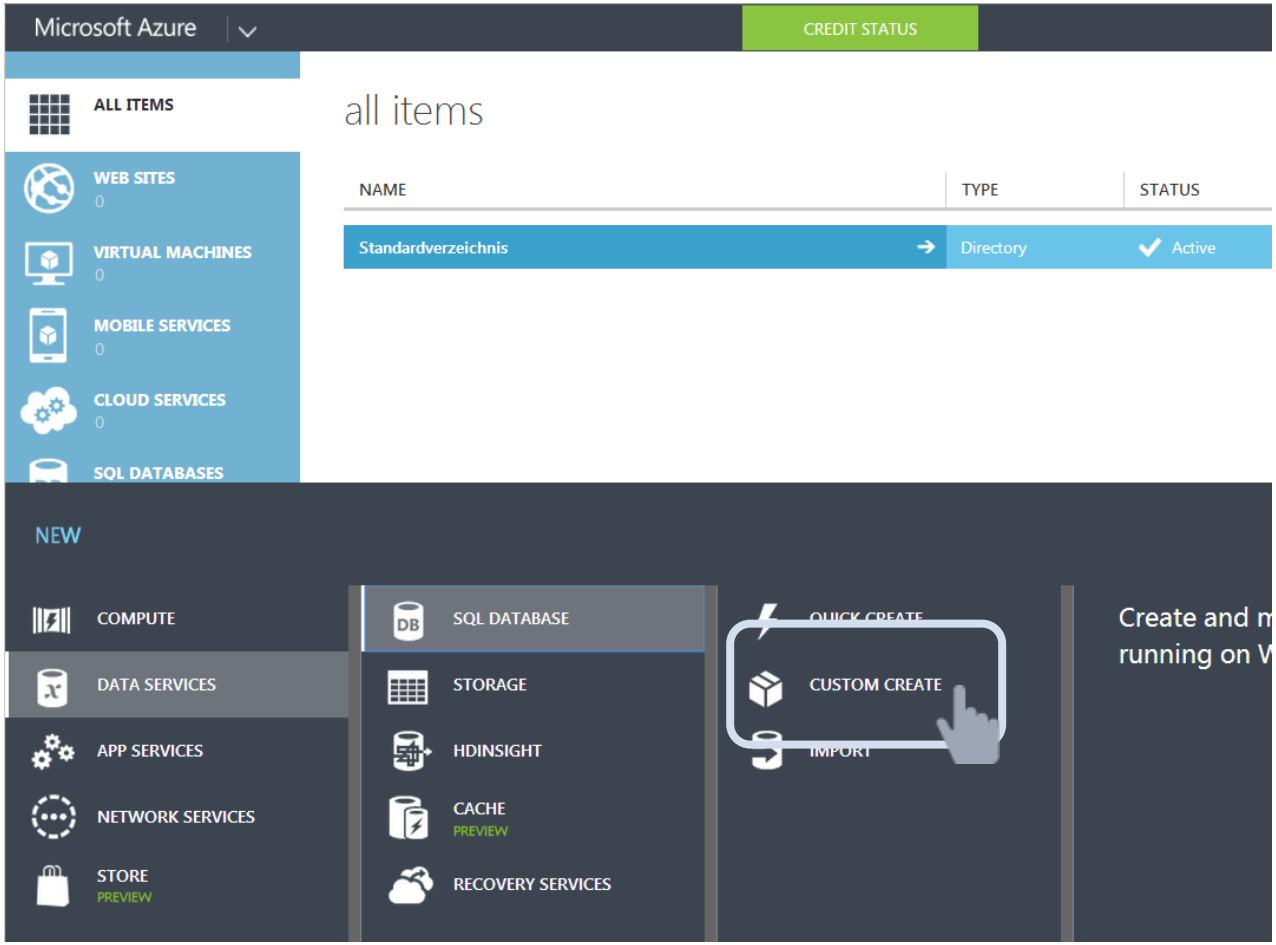


SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

Microsoft SQL Azure

- Similar to RDS

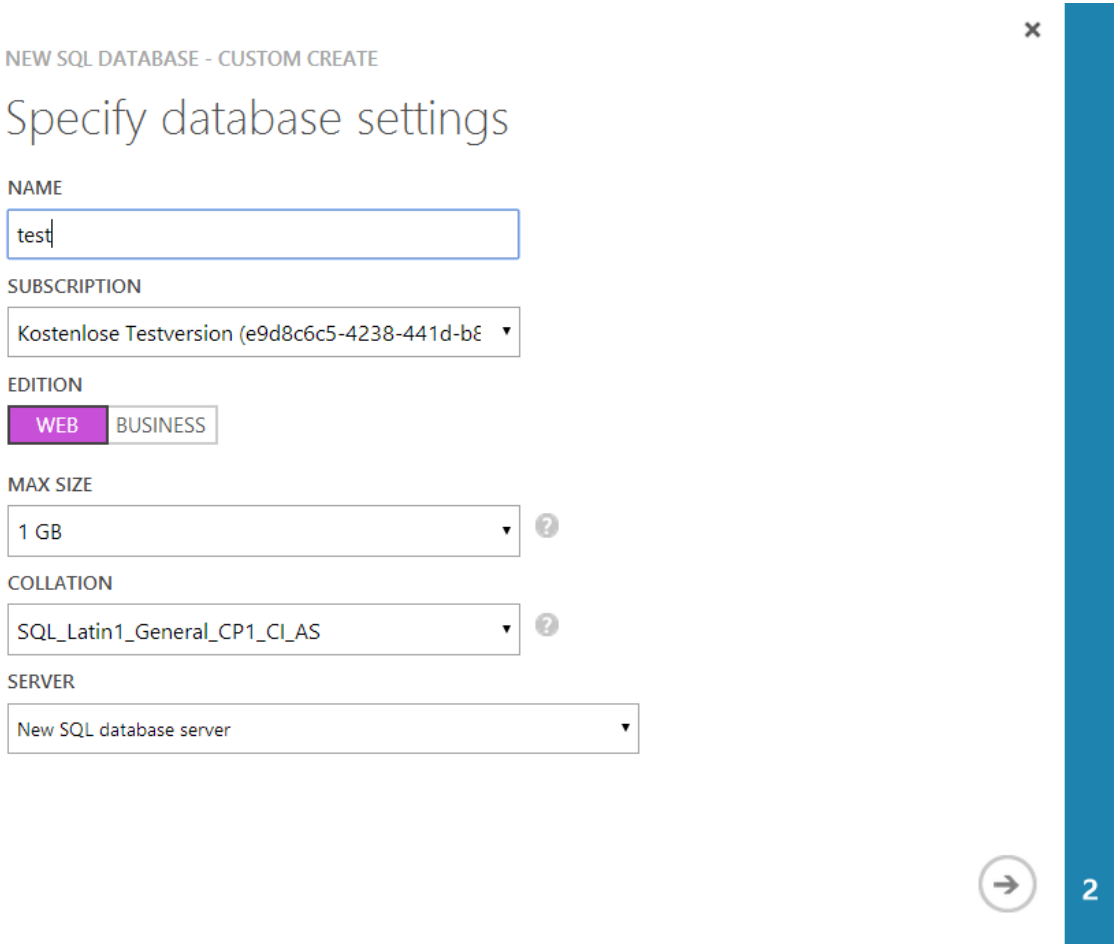


SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

Microsoft SQL Azure

- Similar to RDS

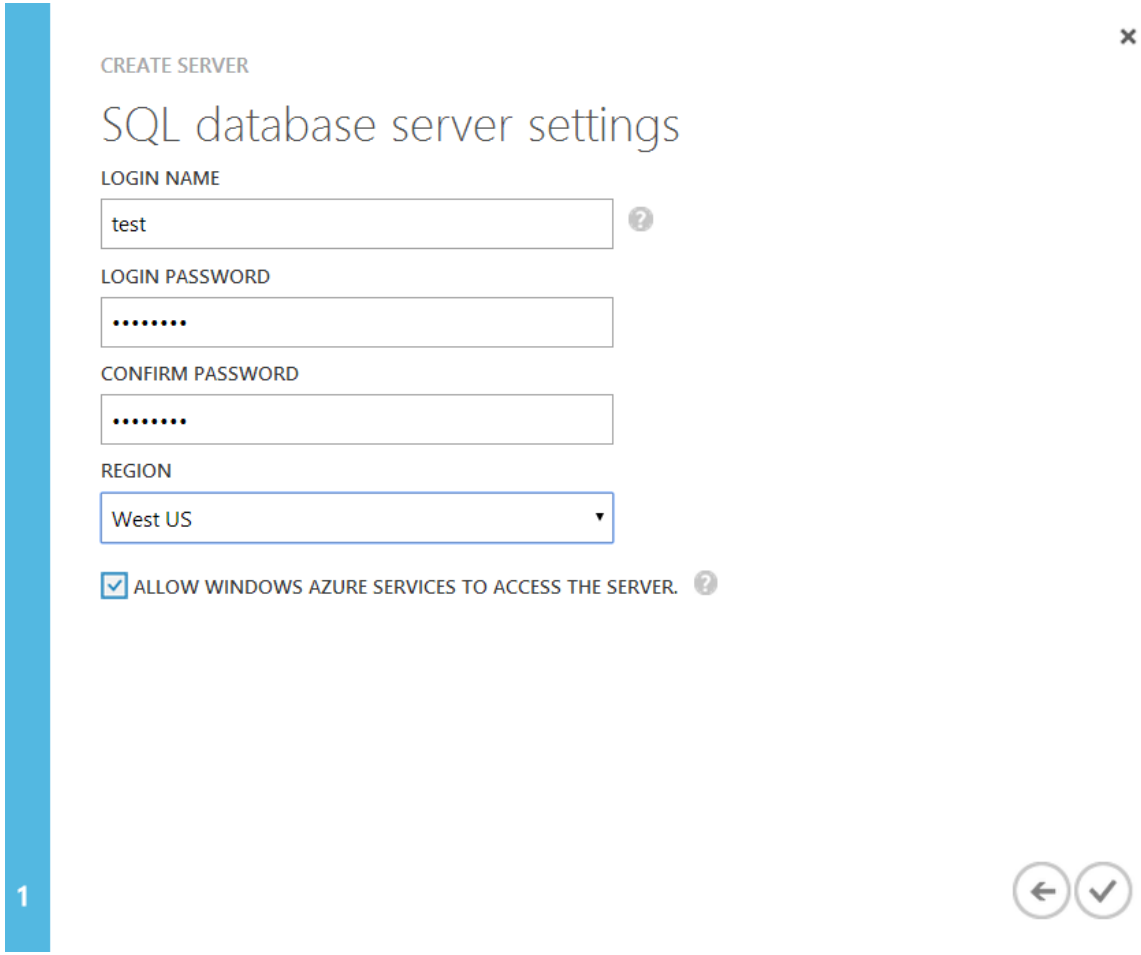


SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

Microsoft SQL Azure

- Similar to RDS

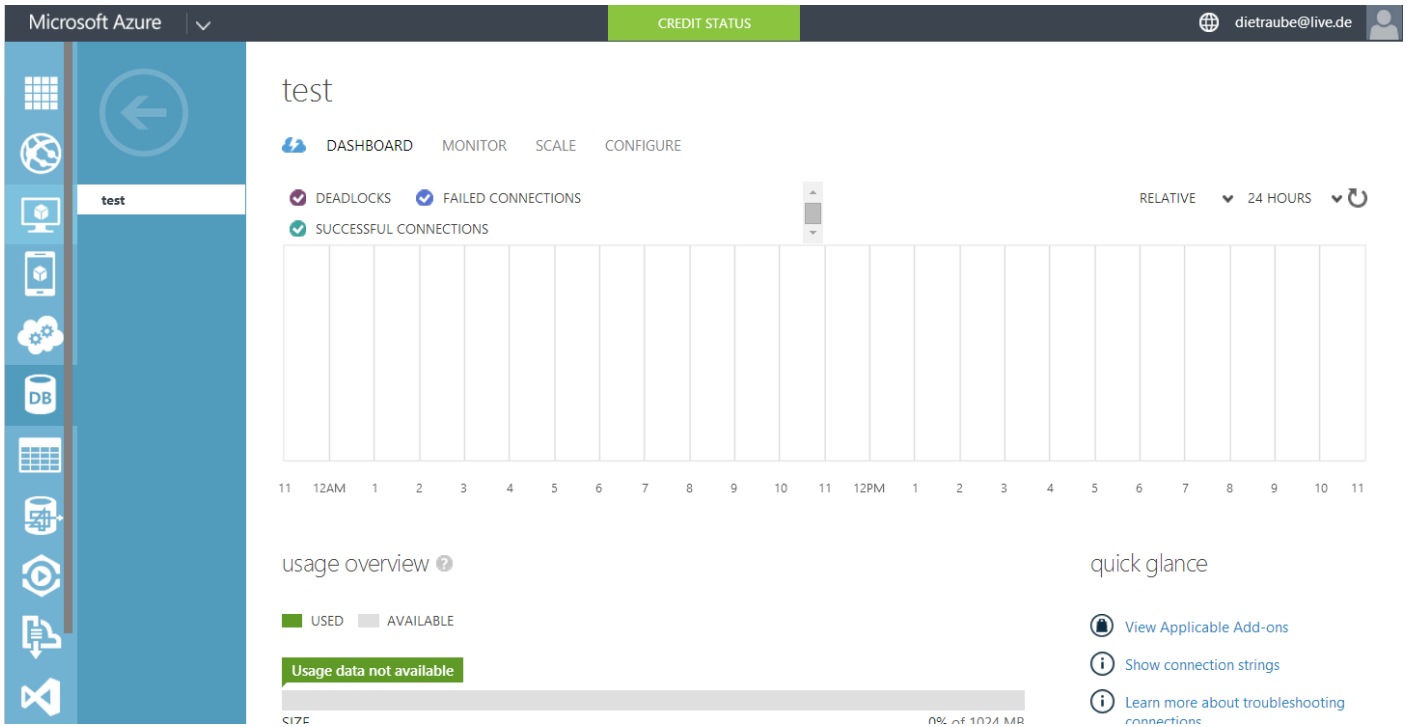


SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

Microsoft SQL Azure

- Similar to RDS

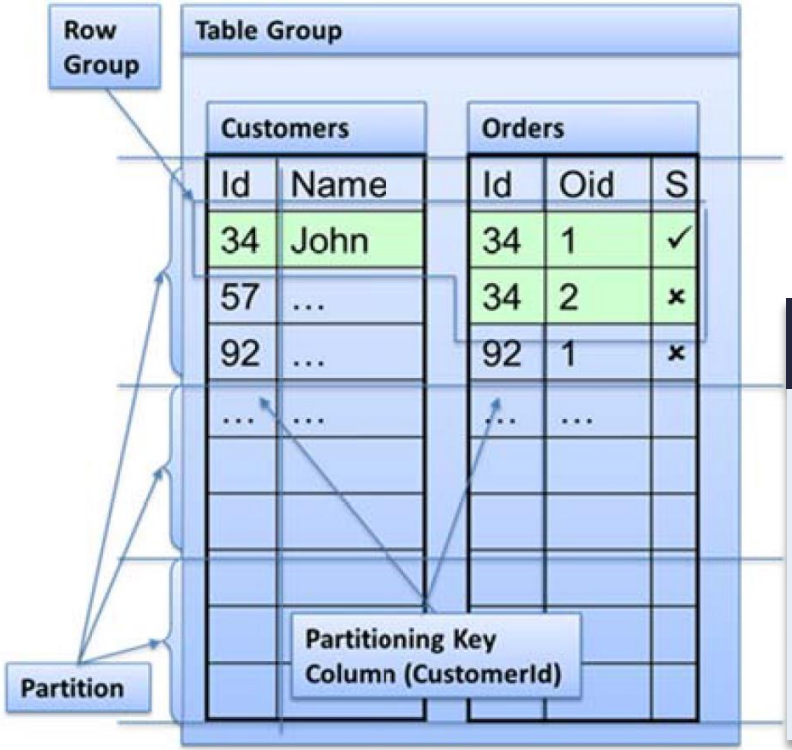


SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

Microsoft SQL Azure

- Similar to RDS



SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

Cloud SQL Server

- Multi-Tenant MSSQL
- Paxos-like commit protocol for consistent replication

P. Bernstein et al. “Adapting Microsoft SQL server for cloud computing”, ICDE 2011

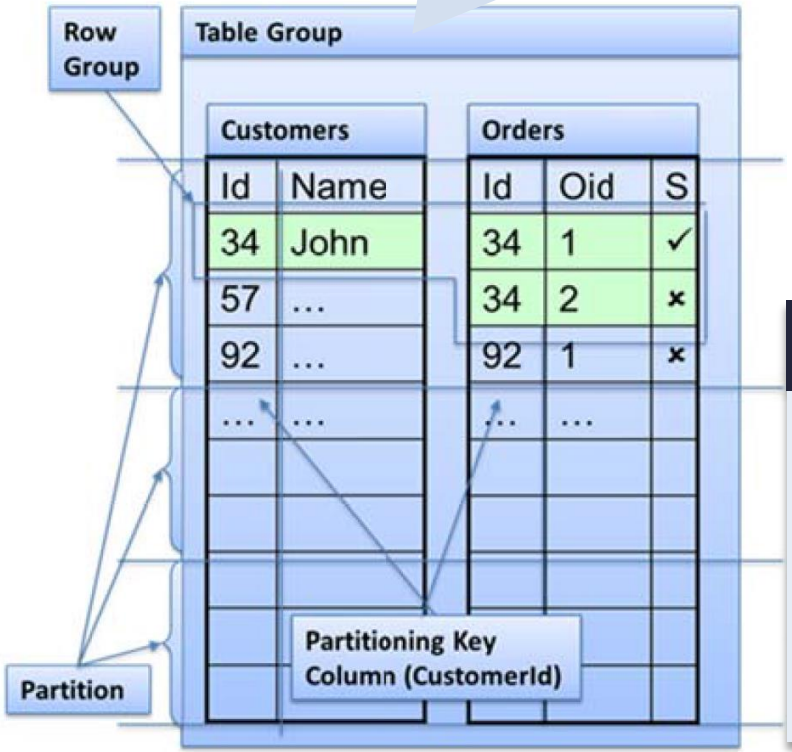
Microsoft SQL Azure

SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

- Similar to RDS

Keyless Table Group: regular database
Keyed Table Group: partitioned by row key



Cloud SQL Server

- Multi-Tenant MSSQL
- Paxos-like commit protocol for consistent replication

P. Bernstein et al. "Adapting Microsoft SQL server for cloud computing", ICDE 2011

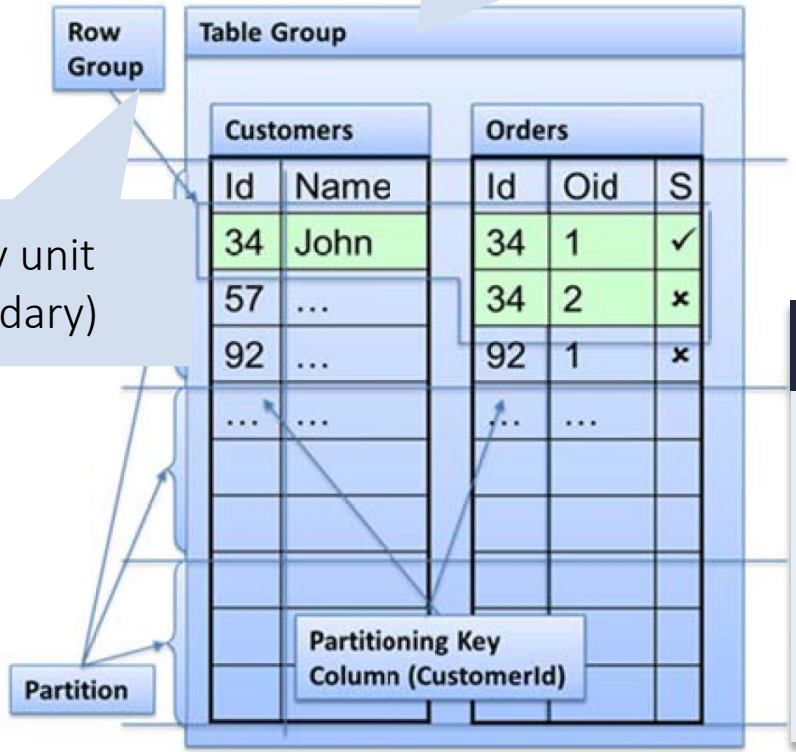
Microsoft SQL Azure

SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

- Similar to RDS

Keyless Table Group: regular database
Keyed Table Group: partitioned by row key



Consistency unit (ACID boundary)

Cloud SQL Server

- Multi-Tenant MSSQL
- Paxos-like commit protocol for consistent replication

P. Bernstein et al. “Adapting Microsoft SQL server for cloud computing”, ICDE 2011

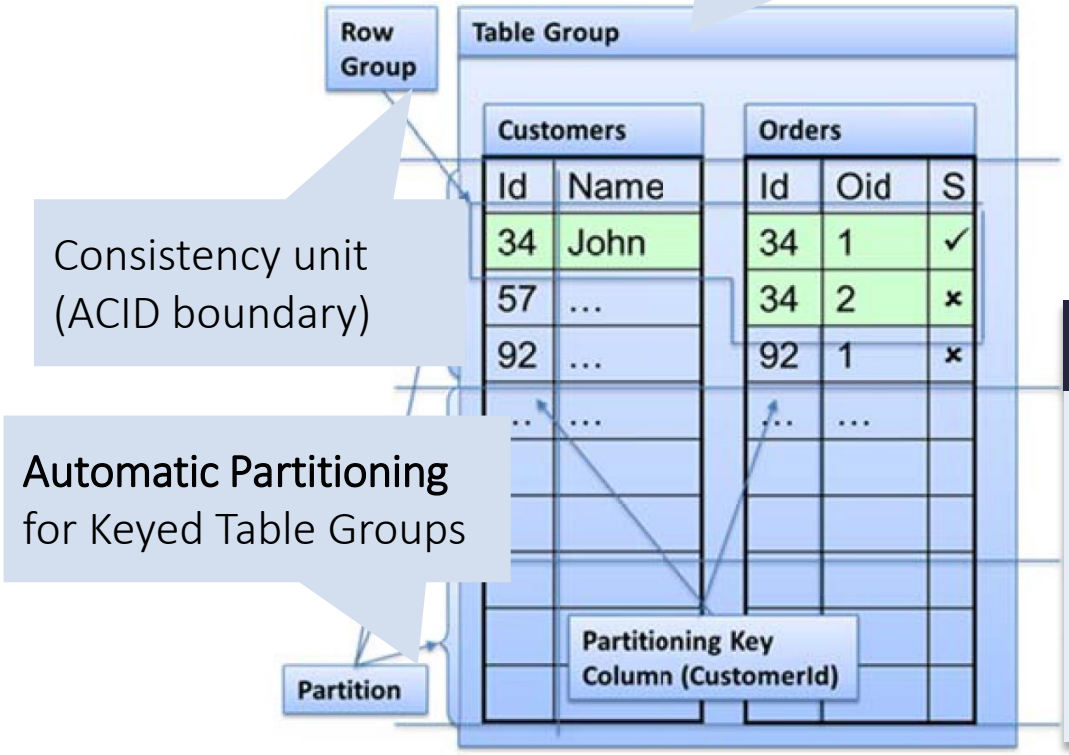
Microsoft SQL Azure

SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

- Similar to RDS

Keyless Table Group: regular database
Keyed Table Group: partitioned by row key



Cloud SQL Server

- Multi-Tenant MSSQL
- Paxos-like commit protocol for consistent replication

P. Bernstein et al. “Adapting Microsoft SQL server for cloud computing”, ICDE 2011

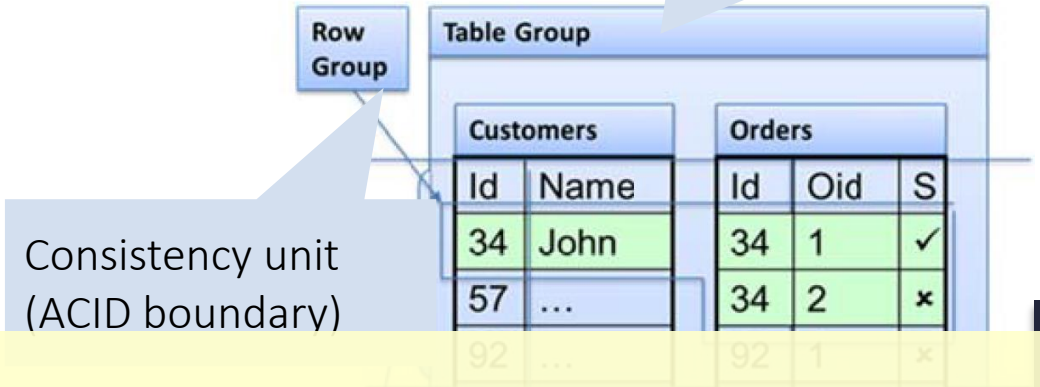
Microsoft SQL Azure

SQL Azure

SQL Azure
Model:
Managed RDBMS
Pricing:
Database size
Underlying DB:
MSSQL Server
API:
T-SQL/TDS

- Similar to RDS

Keyless Table Group: regular database
Keyed Table Group: partitioned by row key



- **SLA**: 99.9% uptime / month
- Usually **Cheaper** than RDS (Multi-Tenancy)
- **Smaller** Databases (max. 150 GB)
- Rich MSSQL server tooling
- Keyed Table Group internal feature only

Other RDBMS services

Google Cloud SQL

- **MySQL** for Google App Engine PaaS
- Support for: patching, replication, backup
- **SLA**: 99,95 % uptime / month

Google Cloud SQL
Pricing:
Database size
Underlying DB:
MySQL

Other RDBMS services

Google Cloud SQL

- **MySQL** for Google App Engine PaaS
- Support for: patching, replication, backup
- **SLA**: 99,95 % uptime / month

Google Cloud SQL
Pricing:
Database size
Underlying DB:
MySQL

Heroku Postgres

- **Postgres** for Heroku PaaS
- Hosted on EC2
- **No SLAs**

Heroku Postgres
Pricing:
Plan based
Underlying DB:
Postgres

Other RDBMS services

Google Cloud SQL

- **MySQL** for Google App Engine PaaS
- Support for: patching, replication, backup
- **SLA**: 99,95 % uptime / month

Heroku Postgres

- **Postgres** for Heroku PaaS
- Hosted on EC2
- **No SLAs**

Trove

- **MySQL** for OpenStack (*Icehouse*)
- Under development (HP driven)
- VM ⇔ Tenant

Google Cloud SQL
Pricing:
Database size
Underlying DB:
MySQL

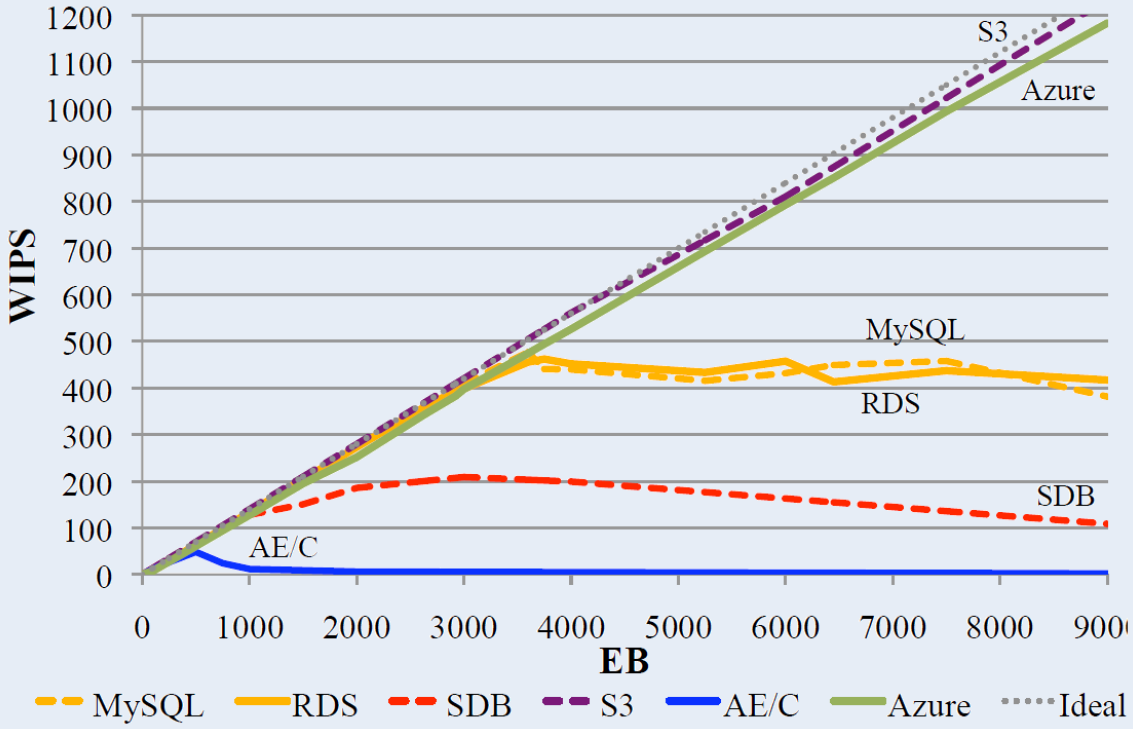
Heroku Postgres
Pricing:
Plan based
Underlying DB:
Postgres

Trove
Pricing:
Own Hardware
Underlying DB:
MySQL

Other RDBMS services

Evaluation of Cloud RDBMSs

TPC-W Benchmark (*Online Shop*), 2010, Emulated Browsers / RPS:



D. Kossmann,T. Kraska: An evaluation of alternative architectures for transaction processing in the cloud", Sigmod 2010

Google Clou

MySQL

Managed NoSQL services

	Model	CAP	Scans	Sec. Indices	Largest Cluster	Lear-ning	Lic.	DBaaS
HBase	Wide-Column	CP	Over Row Key	✗	~700	1/4	Apache	✗ (EMR)
MongoDB	Doc-ument	CP	yes	✓	>100 <500	4/4	GPL	mongoHQ
Riak	Key-Value	AP	✗	✓	~60	3/4	Apache	✗ (Softlayer)
Cassandra	Wide-Column	AP	With Comp. Index	✓	>300 <1000	2/4	Apache	instaclustr
Redis	Key-Value	CA	Through Lists, etc.	manual	N/A	4/4	BSD	Amazon ElastiCache

Managed NoSQL services

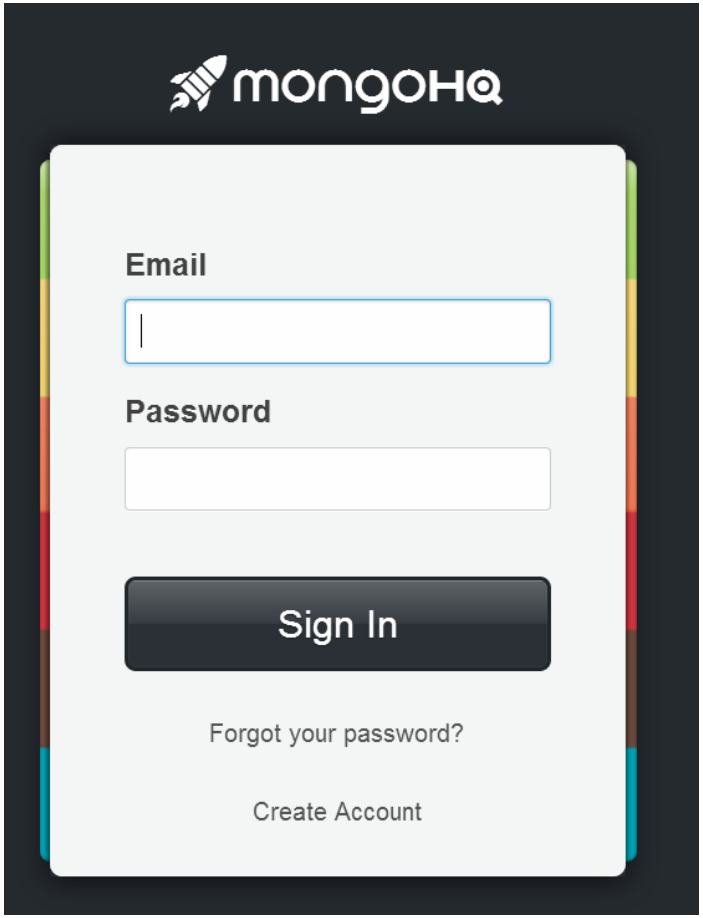
	Model	CAP	Scans	Sec. Indices	Largest Cluster	Lear-ning	Lic.	DBaaS
HBase	Wide-Column	CP	Over Row Key	✗	~700	1/4	Apache	✗ (EMR)
MongoDB	Doc-ument	CP	yes	✓	>100 <500	4/4	GPL	mongoHQ
Riak	Key-Value	AP	✗	✓	~60	3/4	Apache	✗ (Softlayer)
Cassandra		AP	With			2/4	Apache	instaclustr
Redis		CA				4/4	BSD	Amazon ElastiCache

And there are many more:

- CouchDB (e.g. *Cloudant*)
- CouchBase (e.g. *KuroBase Beta*)
- ElasticSearch(e.g. *Bonsai*)
- Solr (e.g. *WebSolr*)
- …

MongoHQ

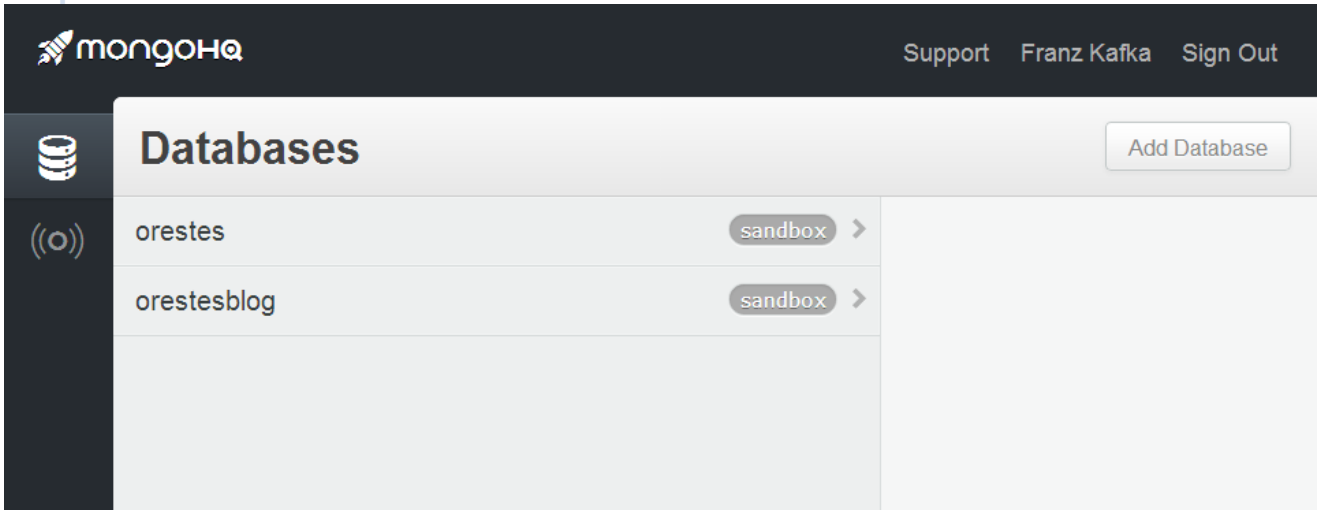
- EC2-based **MongoDB**-as-a-Serivce



MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

MongoHQ

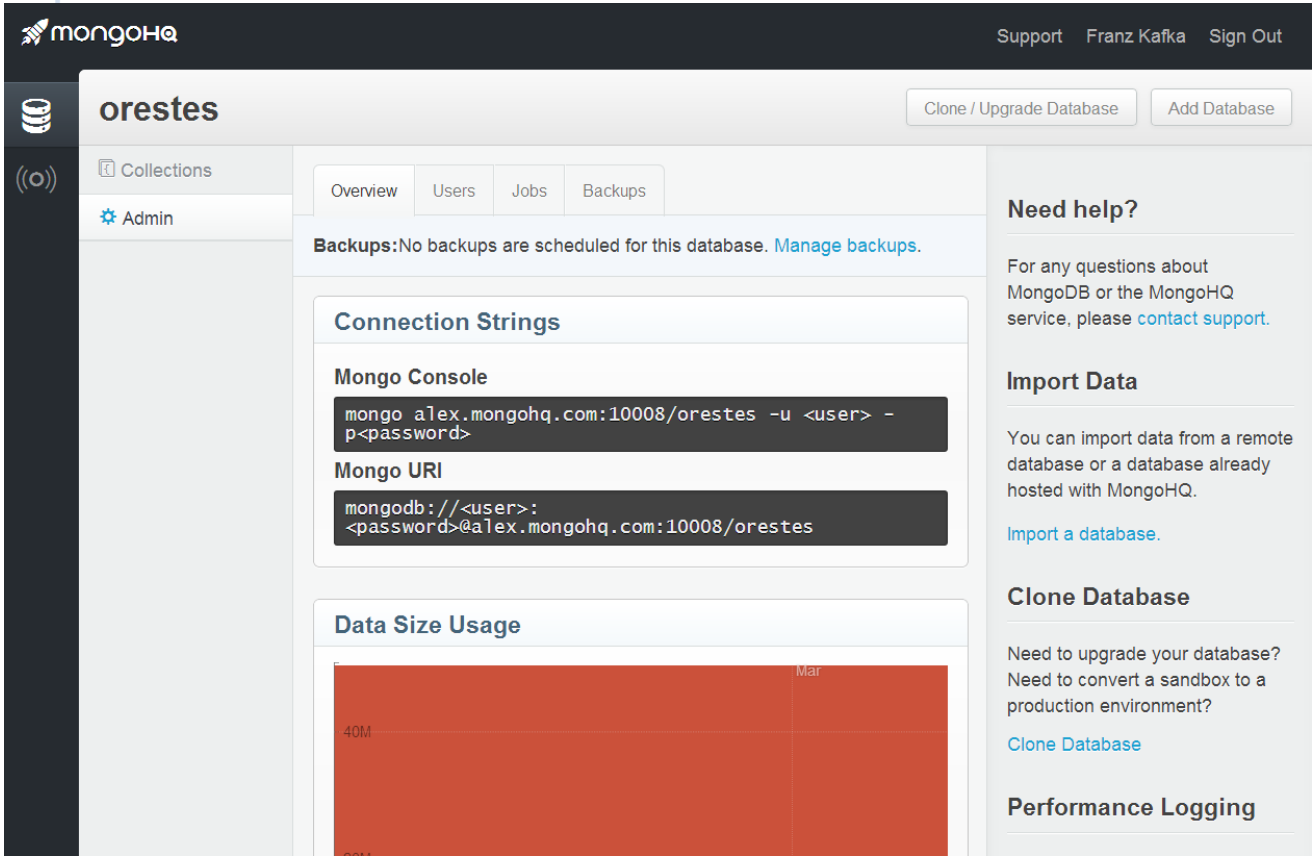
- EC2-based **MongoDB**-as-a-Serivce



MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

MongoHQ

- EC2-based **MongoDB**-as-a-Serivce



MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

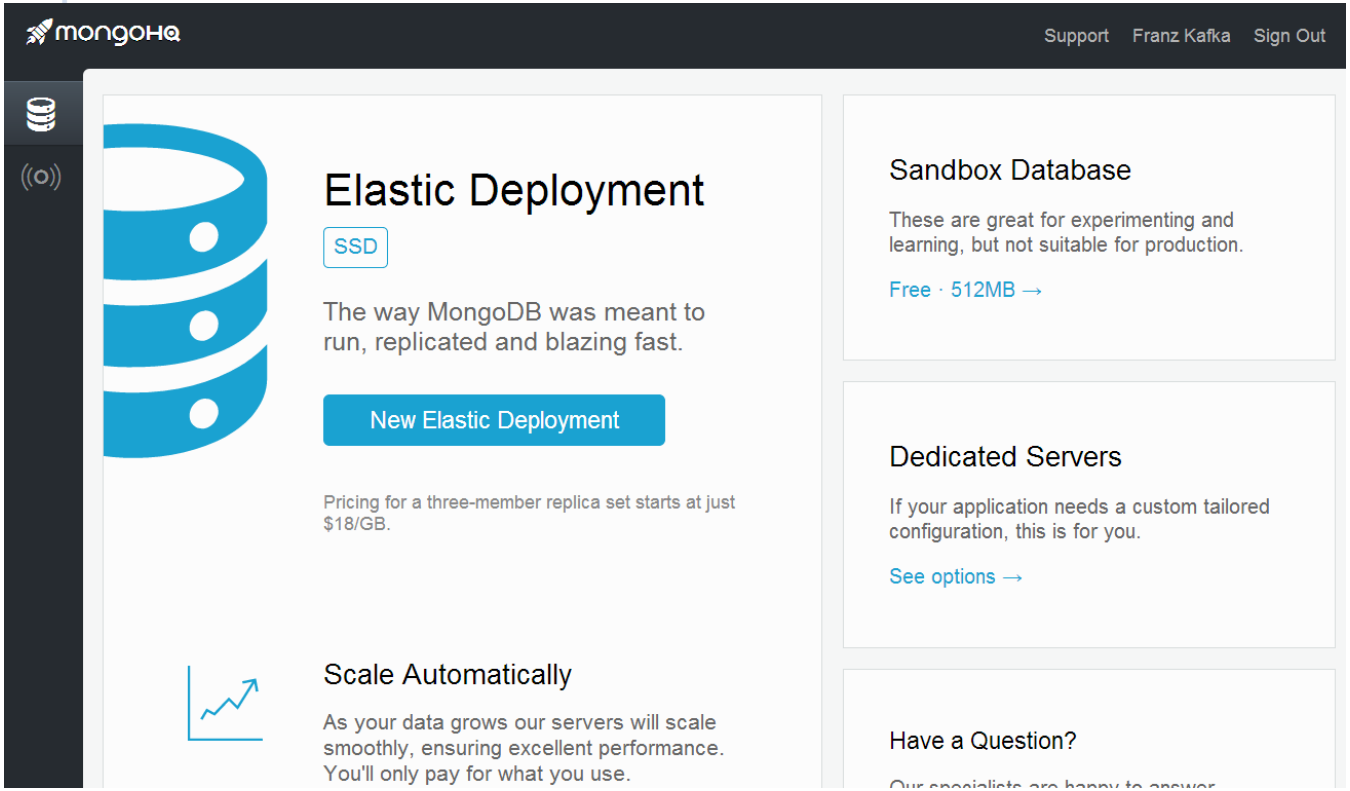
MongoHQ

- EC2-based **MongoDB**-as-a-Serivce

MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

MongoHQ

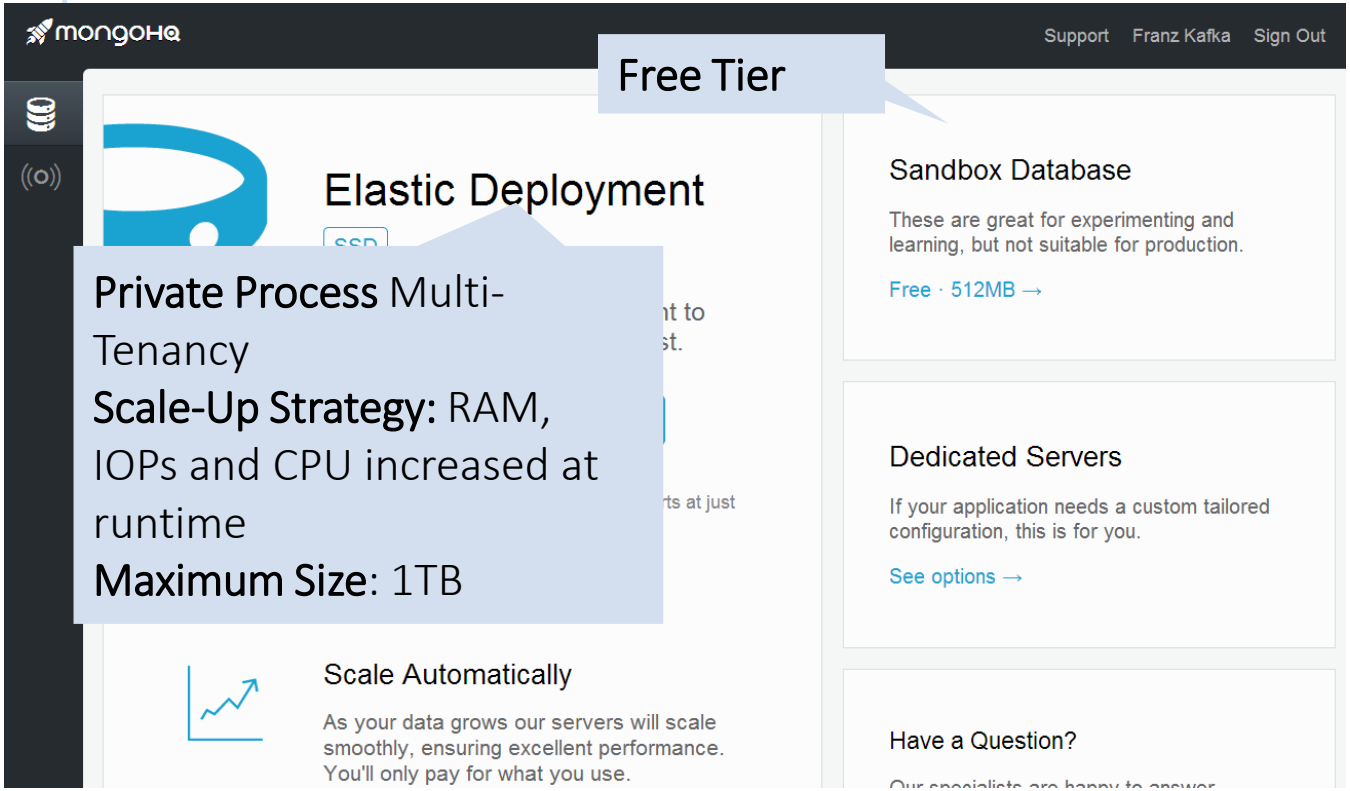
- EC2-based **MongoDB**-as-a-Serivce



MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

MongoHQ

- EC2-based **MongoDB**-as-a-Serivce

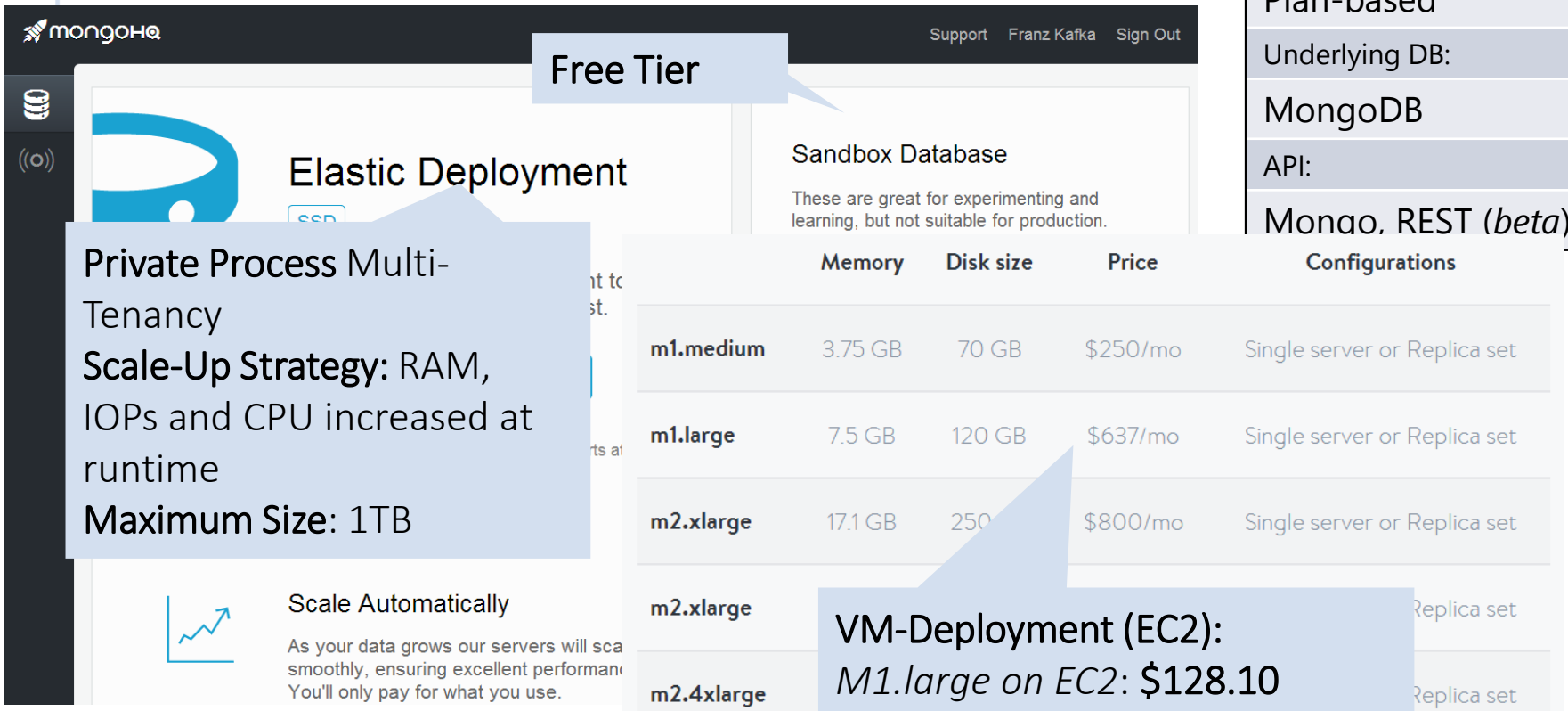


MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

MongoHQ

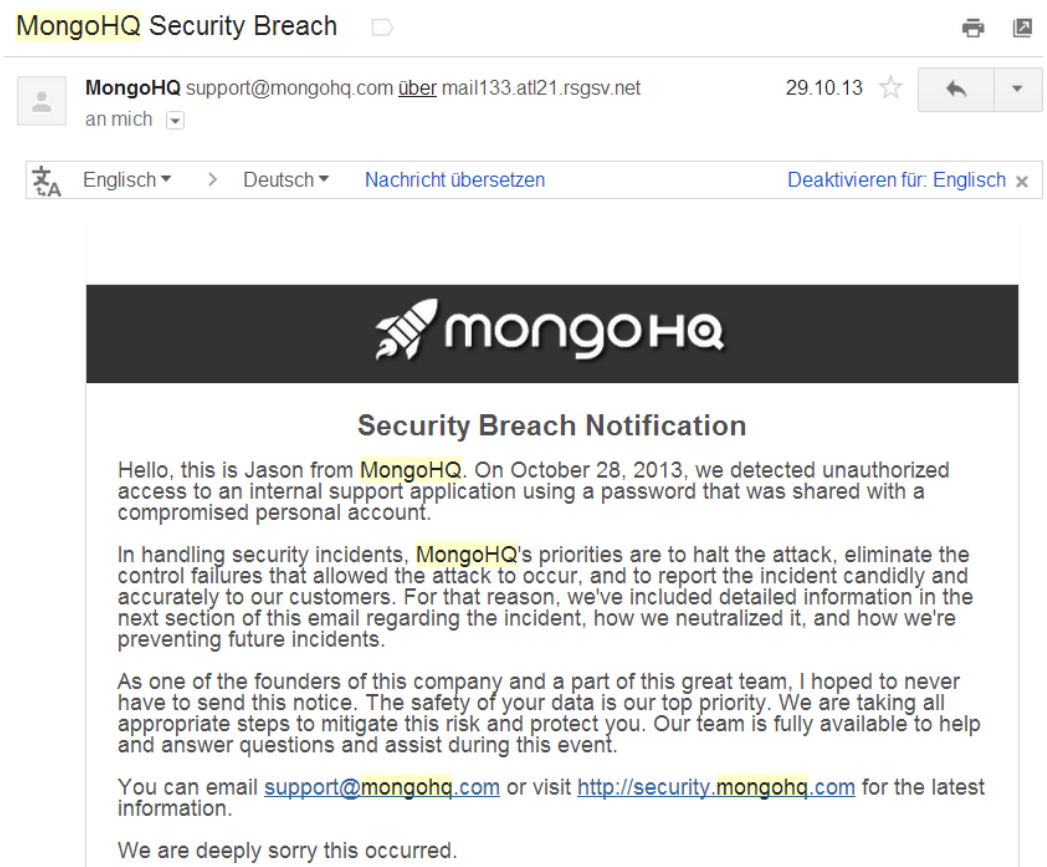
- EC2-based **MongoDB**-as-a-Serivce

MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)



MongoHQ

- EC2-based **MongoDB**-as-a-Serivce
- **#1 thing that should never happen:**



MongoHQ Security Breach

MongoHQ support@mongohq.com über mail133.atl21.rsgsv.net 29.10.13

an mich

Englisch › Deutsch Nachricht übersetzen Deaktivieren für: Englisch

mongoHQ

Security Breach Notification

Hello, this is Jason from MongoHQ. On October 28, 2013, we detected unauthorized access to an internal support application using a password that was shared with a compromised personal account.

In handling security incidents, MongoHQ's priorities are to halt the attack, eliminate the control failures that allowed the attack to occur, and to report the incident candidly and accurately to our customers. For that reason, we've included detailed information in the next section of this email regarding the incident, how we neutralized it, and how we're preventing future incidents.

As one of the founders of this company and a part of this great team, I hoped to never have to send this notice. The safety of your data is our top priority. We are taking all appropriate steps to mitigate this risk and protect you. Our team is fully available to help and answer questions and assist during this event.

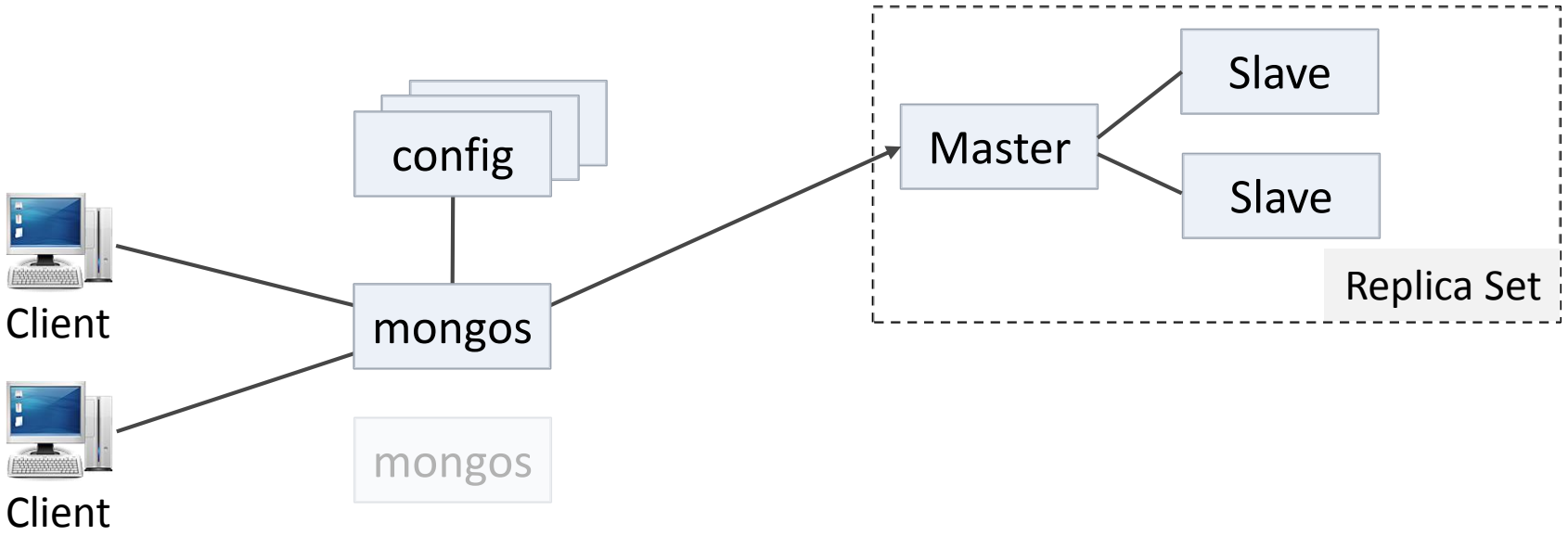
You can email support@mongohq.com or visit http://security.mongohq.com for the latest information.

We are deeply sorry this occurred.

MongoHQ
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
MongoDB
API:
Mongo, REST (*beta*)

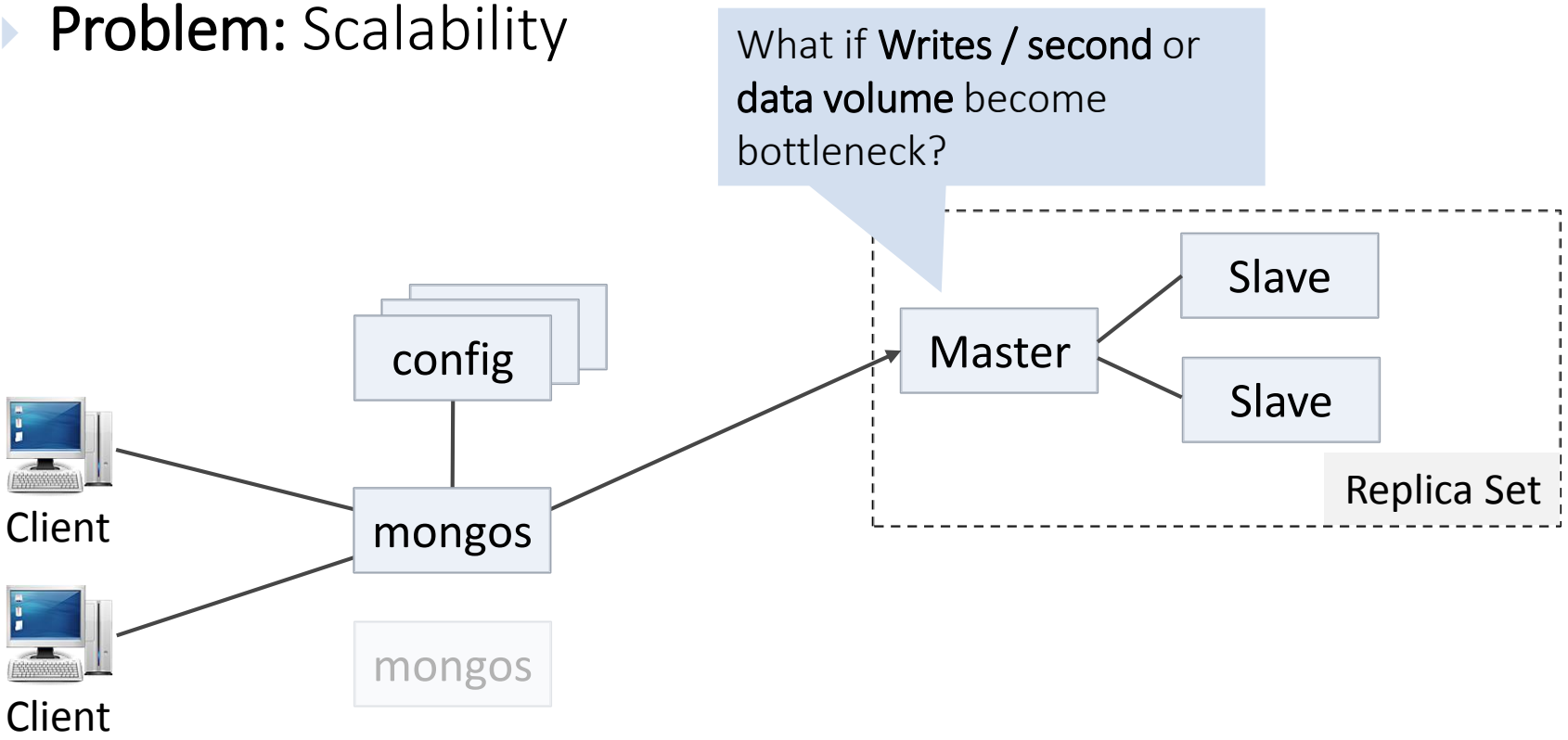
MongoHQ

- **Problem:** Scalability



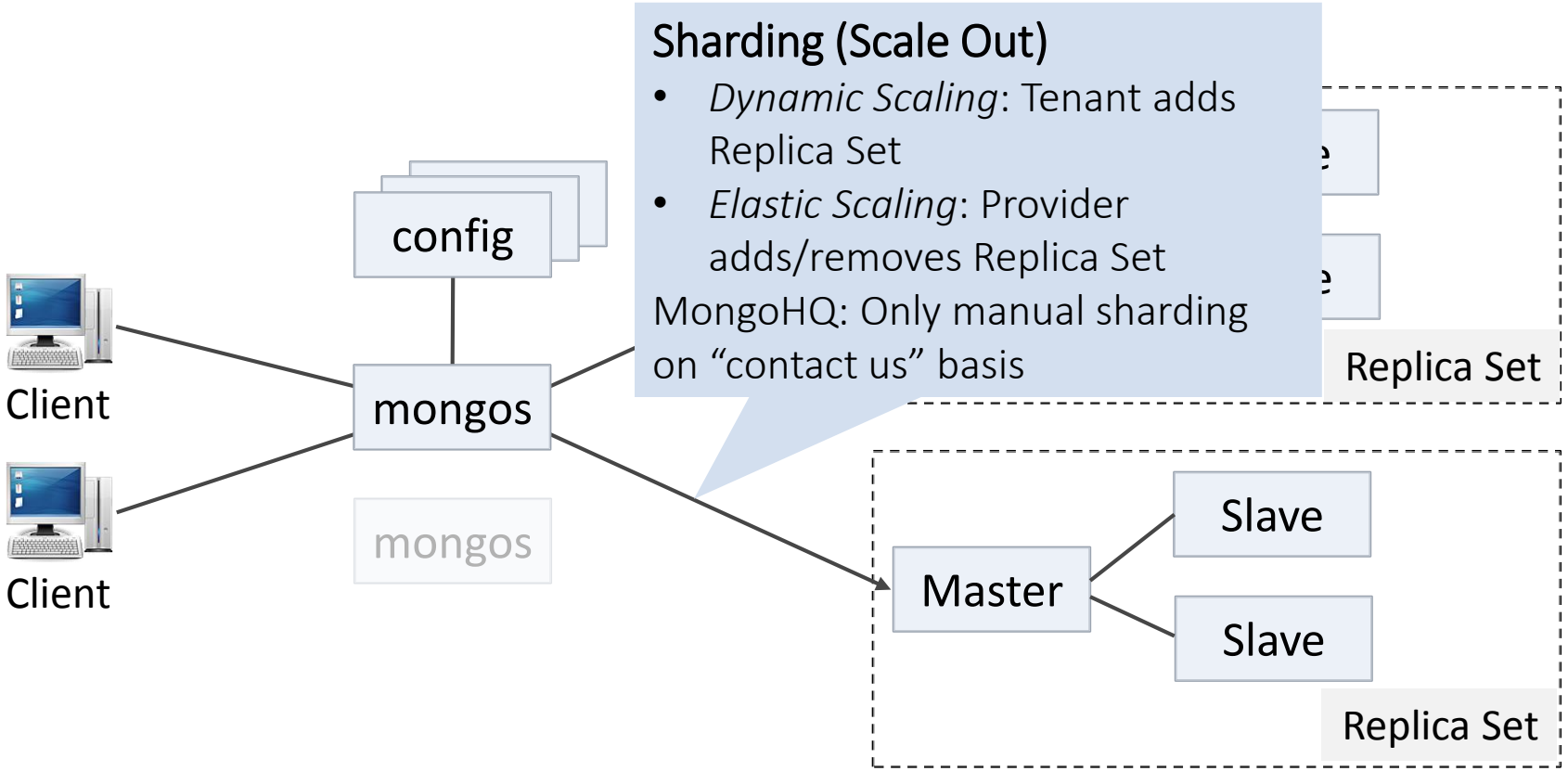
MongoHQ

- **Problem:** Scalability



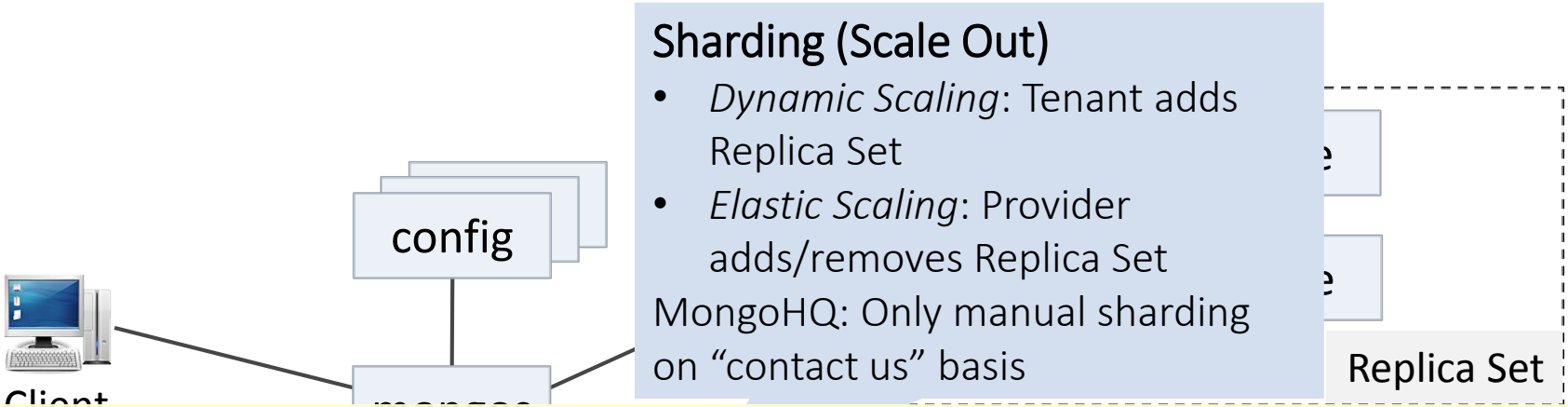
MongoHQ

- **Problem:** Scalability



MongoHQ

- **Problem:** Scalability

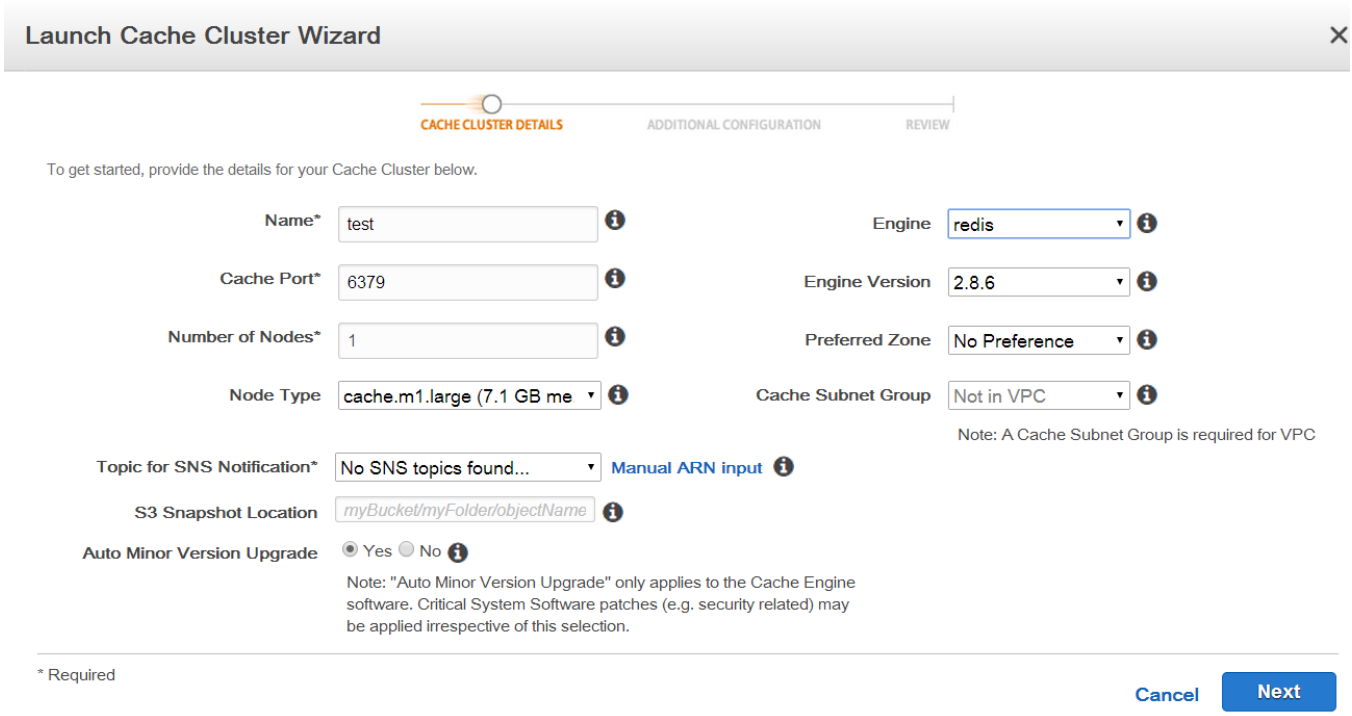


- **Bad:** no SLAs, no horizontal scaling
- **Good**: Solid Dashboard, Backups, Replication, integration with Heroku
- **Competitors**: MongoLab, ObjectRocket, MongoSoup

Amazon ElastiCache

ElastiCache

- „RDS for Memcache and Redis“



ElastiCache
Model:
Managed NoSQL
Pricing:
Infrastructure
Underlying DB:
Memcache, Redis
API:
DB, REST (*management*)

ElastiCache

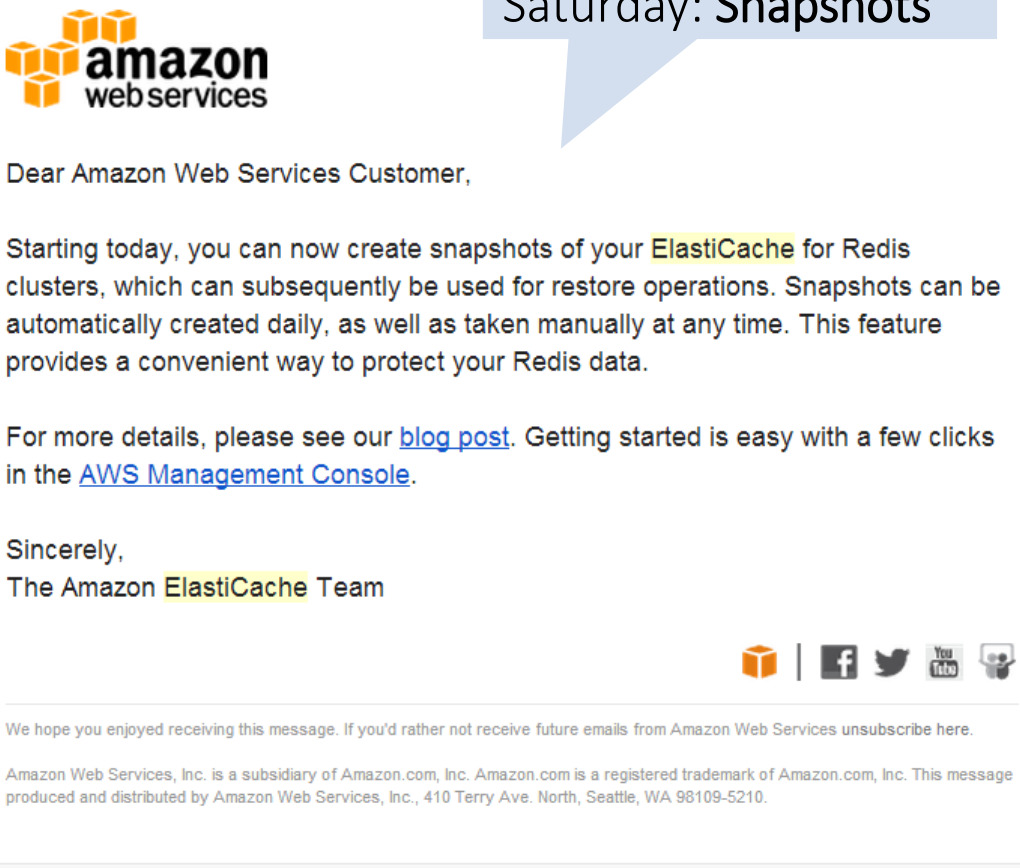
- „RDS for Memcache and Redis“

ElastiCache
Model:
Managed NoSQL
Pricing:
Infrastructure
Underlying DB:
Memcache, Redis
API:
DB, REST (*management*)

ElastiCache

- „RDS for Memcache and Redis“

Announced last Saturday: **Snapshots**



amazon web services

Dear Amazon Web Services Customer,

Starting today, you can now create snapshots of your ElastiCache for Redis clusters, which can subsequently be used for restore operations. Snapshots can be automatically created daily, as well as taken manually at any time. This feature provides a convenient way to protect your Redis data.

For more details, please see our blog post. Getting started is easy with a few clicks in the AWS Management Console.

Sincerely,
The Amazon ElastiCache Team

We hope you enjoyed receiving this message. If you'd rather not receive future emails from Amazon Web Services unsubscribe here.

Amazon Web Services, Inc. is a subsidiary of Amazon.com, Inc. Amazon.com is a registered trademark of Amazon.com, Inc. This message produced and distributed by Amazon Web Services, Inc., 410 Terry Ave. North, Seattle, WA 98109-5210.

Amazon ElastiCache

ElastiCache
Model:
Managed NoSQL
Pricing:
Infrastructure
Underlying DB:
Memcache, Redis
API:
DB, REST (*management*)

Amazon ElastiCache

ElastiCache

- „RDS for Memcache and Redis“

ElastiCache
Model:
Managed NoSQL
Pricing:
Infrastructure
Underlying DB:
Memcache, Redis
API:
DB, REST (*management*)

1 Cache Parameter Group(s) selected

Cache Parameter Group: default.redis2.8

Viewing: All Parameters ▾ Edit Parameters

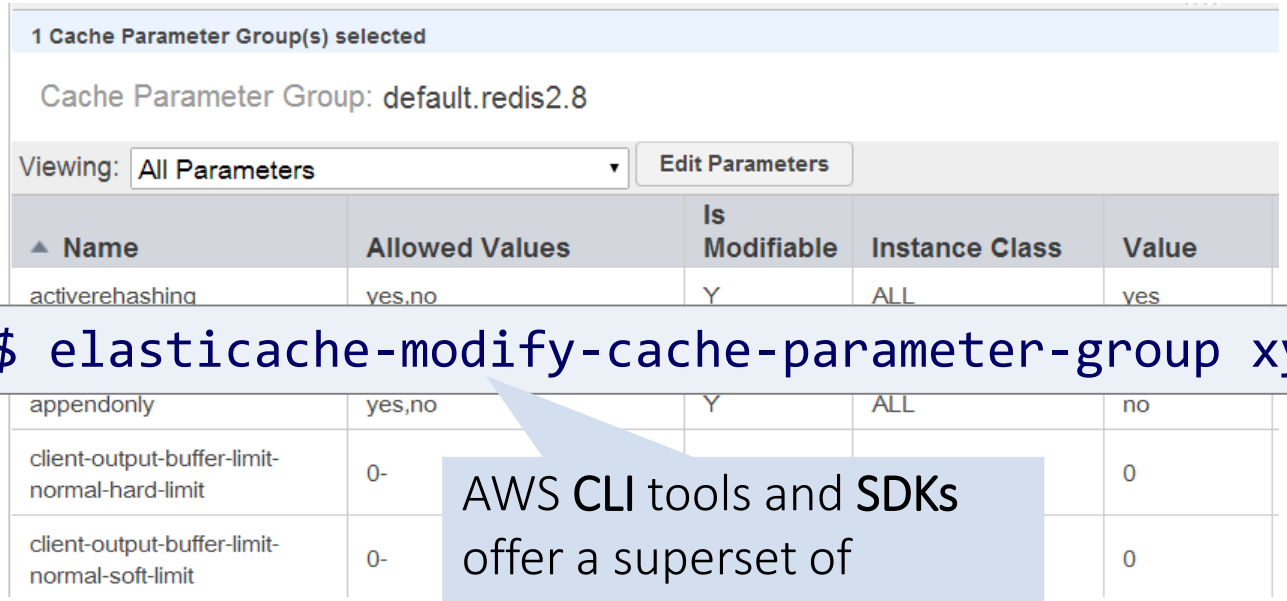
Name	Allowed Values	Is Modifiable	Instance Class	Value
activerehashing	yes,no	Y	ALL	yes
appendfsync	always,everysec,no	Y	ALL	everysec
appendonly	yes,no	Y	ALL	no
client-output-buffer-limit-normal-hard-limit	0-	Y	ALL	0
client-output-buffer-limit-normal-soft-limit	0-	Y	ALL	0

Amazon ElastiCache

ElastiCache

- „RDS for Memcache and Redis“

ElastiCache
Model:
Managed NoSQL
Pricing:
Infrastructure
Underlying DB:
Memcache, Redis
API:
DB, REST (*management*)

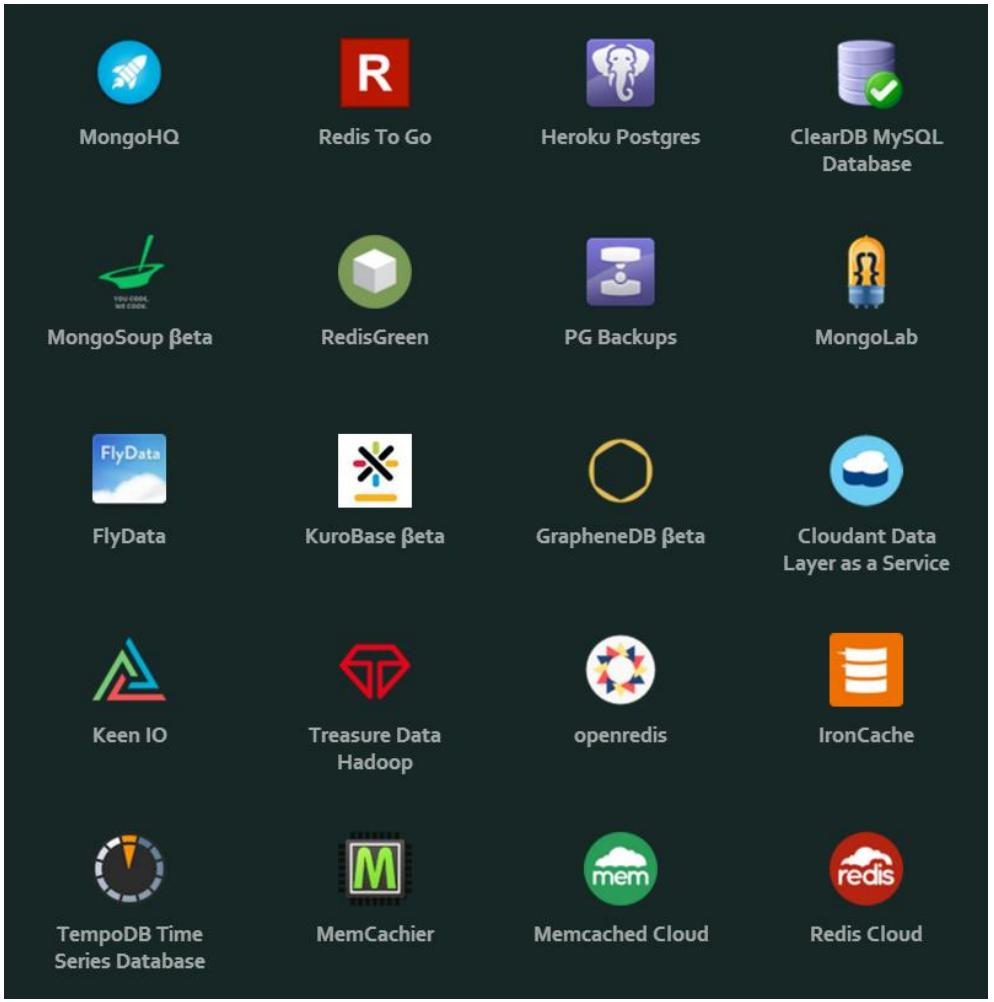



```
$ elasticache-modify-cache-parameter-group xy
```

AWS **CLI** tools and **SDKs** offer a superset of Dashboard functions

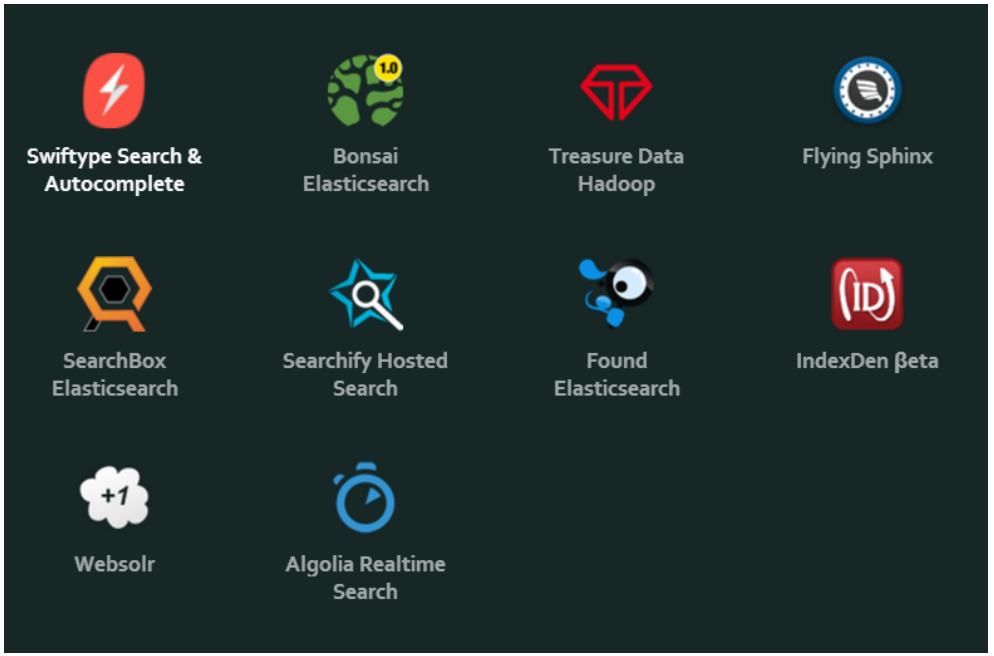
Heroku DBaaS Addons

- Many Hosted NoSQL DbaaS Providers represented



Heroku DBaaS Addons

- Many Hosted NoSQL DbaaS Providers represented
- And Search



Heroku Redis2Go example

Create Heroku App:

```
$ heroku create
```

Redis2Go
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
Redis
API:
Redis

Heroku Redis2Go example

Create Heroku App:

```
$ heroku create
```

Add Redis2Go Addon:

```
$ heroku addons:add redistogo
-----> Adding RedisToGo to fat-unicorn-1337... done, v18 (free)
```

Redis2Go
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
Redis
API:
Redis

Heroku Redis2Go example

Create Heroku App:

```
$ heroku create
```

Add Redis2Go Addon:

```
$ heroku addons:add redistogo
-----> Adding RedisToGo to fat-unicorn-1337... done, v18 (free)
```

Use Connection URL (environment variable):

```
uri = URI.parse(ENV["REDISTOGO_URL"])
REDIS = Redis.new(:url => ENV['REDISTOGO_URL'])
```

Redis2Go
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
Redis
API:
Redis

Heroku Redis2Go example

Redis2Go
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
Redis
API:
Redis

Create Heroku App:

```
$ heroku create
```

Add Redis2Go Addon:

```
$ heroku addons:add redistogo
-----> Adding RedisToGo to fat-unicorn-1337... done, v18 (free)
```

Use Connection URL (environment variable):

```
uri = URI.parse(ENV["REDISTOGO_URL"])
REDIS = Redis.new(:url => ENV['REDISTOGO_URL'])
```

Deploy:

```
$ git push heroku master
```

Heroku Redis2Go example

Create Heroku App:

```
$ heroku create
```

Add Redis2Go Addon:

```
$ heroku addons:add redistogo
-----> Adding RedisToGo to fat-unicorn-1337... done, v18 (free)
```

Use Connection URL (environment variable):

```
uri = URI.parse(ENV["REDISTOGO_URL"])
REDIS = Redis.new(:url => ENV['REDISTOGO_URL'])
```

Dep

```
$ gi                    master
```

- **Very simple**
- Only suited for small to medium applications (no SLAs, limited control)

Redis2Go
Model:
Managed NoSQL
Pricing:
Plan-based
Underlying DB:
Redis
API:
Redis

Proprietary Database services

	Model	CAP	Scans	Sec. Indices	Queries	API	Scale-out	SLA
SimpleDB	Table-Store	CP	Yes (as queries)	Auto-matic	SQL-like (no joins, groups, …)	REST + SDKs	✗	✗
Dynamo-DB	Table-Store	CP	By range key / index	Local Sec. Global Sec.	Key+Cond. On Range Key(s)	REST + SDKs	Automatic over Prim. Key	✗
Azure Tables	Table-Store	CP	By range key	✗	Key+Cond. On Range Key	REST + SDKs	Automatic over Part. Key	99.9% uptime
AE/Cloud DataStore	Entity-Group	CP	Yes (as queries)	Auto-matic	Conjunct. of Eq. Predicates	REST/ SDK, JDO,JPA	Automatic over Entity Groups	✗
S3, Az. Blob, GCS	Blob-Store	AP	✗	✗	✗	REST + SDKs	Automatic over key	99.9% uptime (S3)

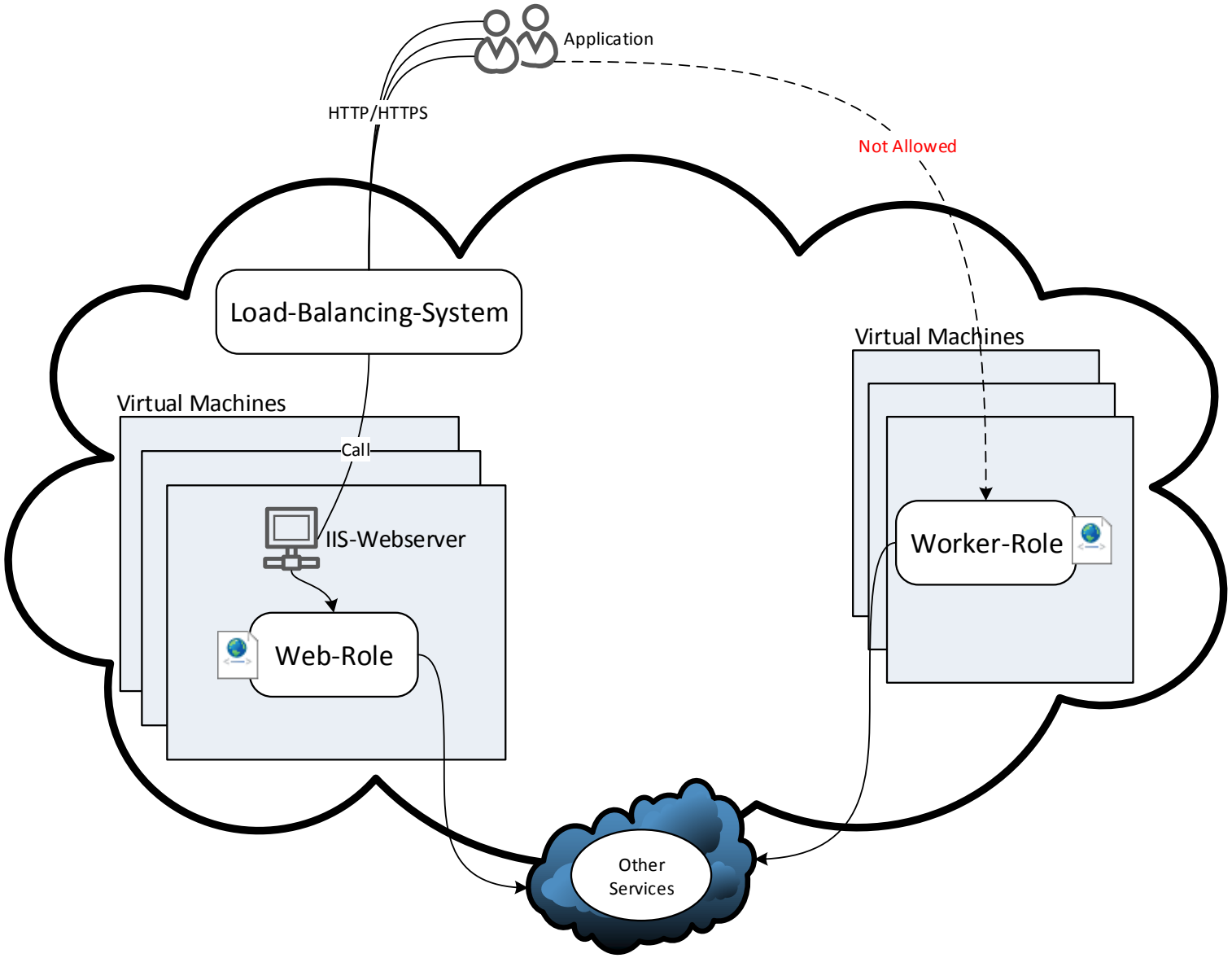
Proprietary Database services

	Model	CAP	Scans	Sec. Indices	Queries	API	Scale-out	SLA
SimpleDB	Table-Store	CP	Yes (as queries)	Auto-matic	SQL-like (no joins, groups, …)	REST + SDKs		
Dynamo-DB	Table-Store	CP	By range key / index	Local Sec. Global Sec.	Key+Cond. On Range Key(s)	REST + SDKs	Automatic over Prim. Key	
Azure Tables	Table-Store	CP	By range key		Key+Cond. On Range Key	REST + SDKs	Automatic over Part. Key	99.9% uptime
AE/Cloud DataStore	Entity-Group	CP	Yes (as queries)	Auto-matic	Conjunct. Predicates	REST/ JDO,JPA	Automatic Entity Groups	
S3, Az. Blob, GCS	Blob	AP				REST + SDKs	Automatic over key	99.9% uptime (S3)

There are many more **object stores** (HP, Rackspace, etc.)

…but **no comparable Table Stores**

Azure Storage



Azure Storage

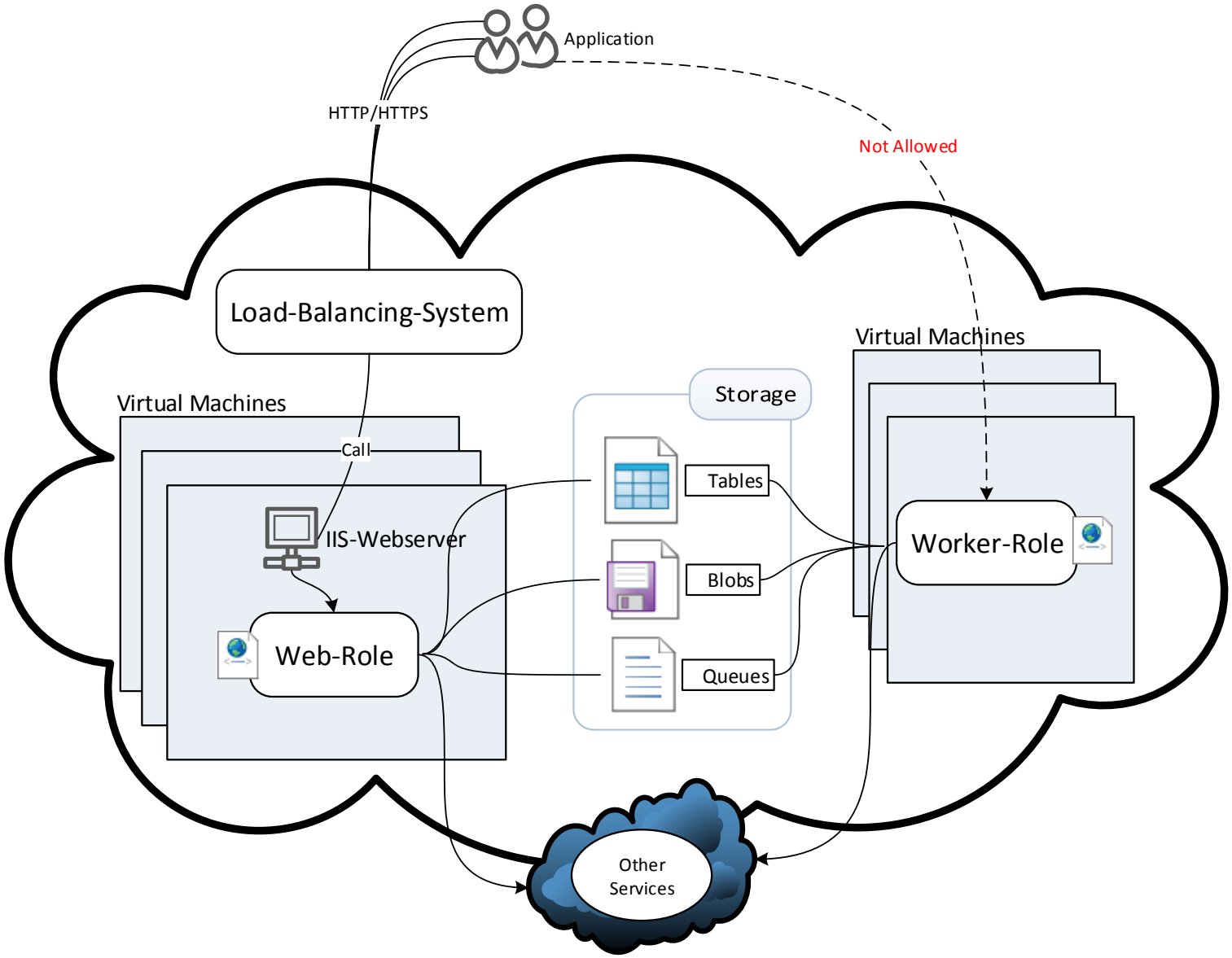


Table Service example: Azure Tables

REST API

Partition Key	**Row Key** ***(sortiert)***	**Timestamp** ***(autom.)***	**Property*1***	**Property*n***
intro.pdf	v1.1	14/6/2013	…	…
intro.pdf	v1.2	15/6/2013		…
präs.pptx	v0.0	11/6/2013	…	

Partition

Partition

Table Service example: Azure Tables

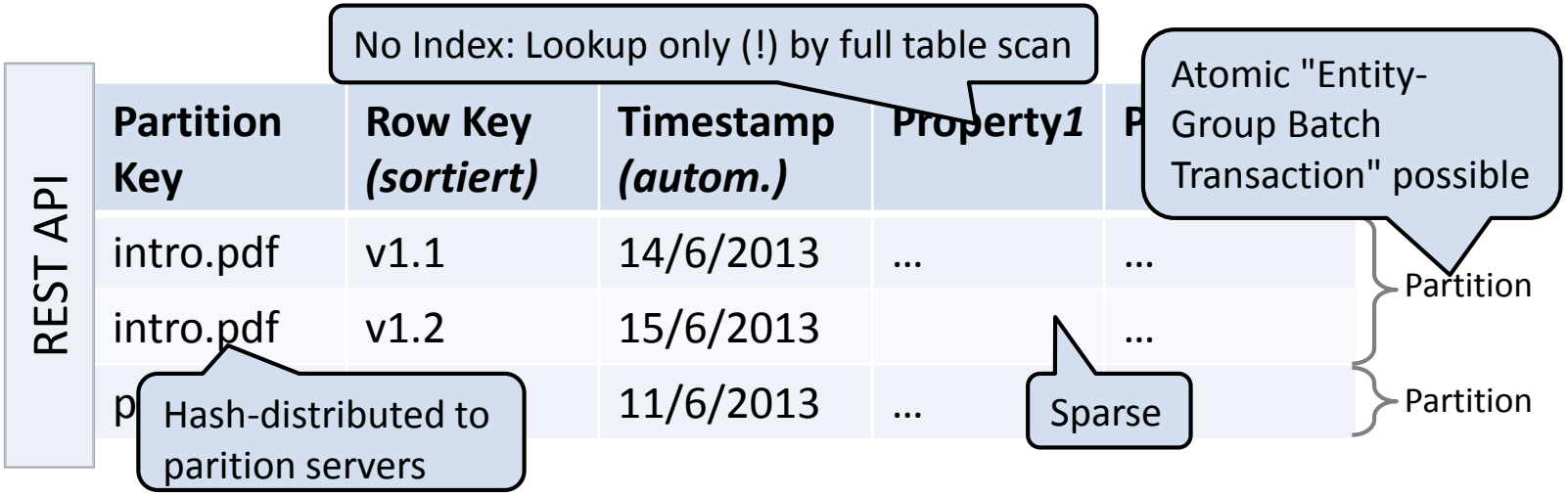


Table Service example: Azure Tables

REST API

Partition Key	Row Key *(sortiert)*	Timestamp *(autom.)*	Property*1*	Property*n*	
intro.pdf	v1.1	14/6/2013	…	…	Partition
intro.pdf	v1.2	15/6/2013		…	
präs.pptx	v0.0	11/6/2013	…		Partition

- Similar to Amazon **SimpleDB** and **DynamoDB**

- Indexes all attributes
- Rich(er) queries
- Many Limits (size, RPS, etc.)

- Provisioned Throughput
- On SSDs („single digit latency“)
- Optional Indexes

Azure Table Storage

Challenges:

- Single partition and range key (modelling)
- Very “basic“ queries

Azure Tables
Model:
Proprietary
Pricing:
Requests + Storage + Network
Underlying DB:
Custom System
API:
REST

Azure Tables

Azure Table Storage

Ch

- S
- V

How can we improve Azure Storage?

← Storage

789
votes

Vote

Support secondary Indexes

Need to be able to sort on something other than the rowkey

andybritcliffe shared this idea · Nov 24, 2009 · Flag idea as inappropriate...

Azure Tables
Model:
Proprietary
Pricing:
Requests + Storage + Network
Underlying DB:
Custom System
API:
REST

Azure Table Storage

Azure Tables
Model:
Proprietary
Pricing:
Requests + Storage + Network
Underlying DB:
Custom System
API:
REST

Challenges:

- Single partition and range key (modelling)
- Very “basic“ queries

Good:

- Automatic **Distribution**
- **Replicated** 3x locally + 1x async. geo-replica

Azure Table Storage

Azure Tables
Model:
Propriertary
Pricing:
Requests + Storage + Network
Underlying DB:
Custom System
API:
REST

Challenges:

- Single partition and range key (modelling)
- Very “basic“ queries

Good:

- Automatic **Distribution**
- **Replicated** 3x locally + 1x async. geo-replica

Very good:

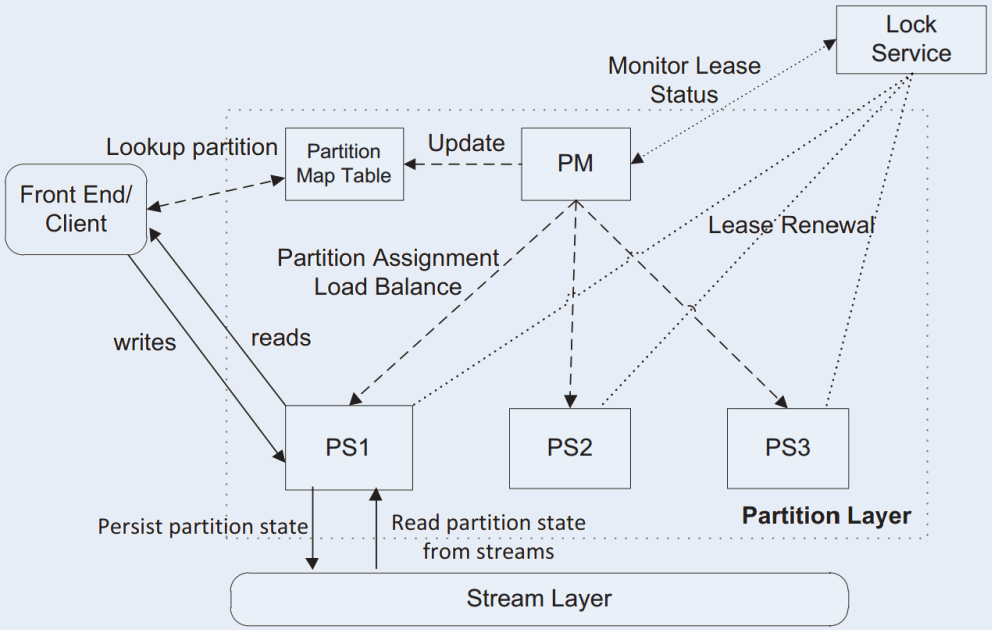
- **SLA** (99.9% uptime)
- Internal architecture **published**

Azure Table Storage

Windows Azure Storage

Idea:

- Layered storage infrastructure for Blobs and Tables
- Use **research results** (GFS, BigTable, Paxos, LSM, Erasure Coding)



B. Calder, et al. "Windows Azure Storage: a highly available cloud storage service with strong consistency." , SOSP 2011

Azure Tables

Model:

Pro

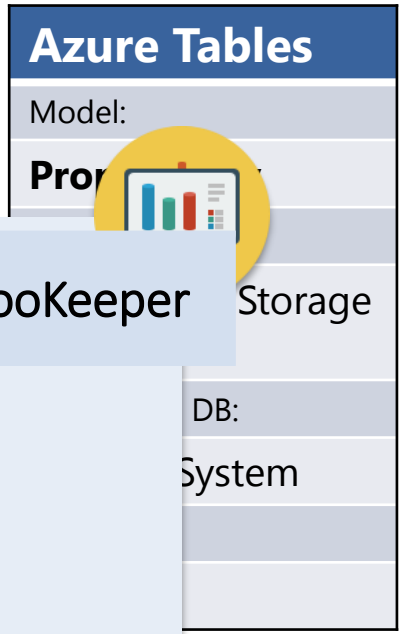
+ Storage

rk

DB:

System

Azure Table Storage

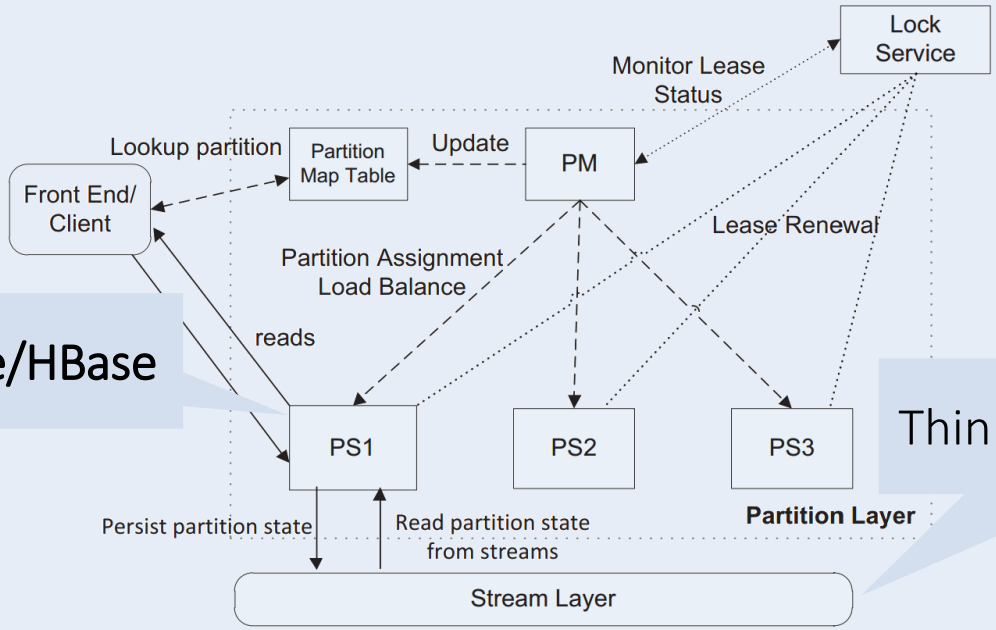


Windows Azure Storage

Idea:

- Layered storage infrastructure for Blobs and Tab
- Use **research results** (GFS, BigTable, Paxos, LSM, Erasu ng)

Think: **Chubby/ZooKeeper**



Think: **BigTable/HBase**

Think: **GFS/HDFS**

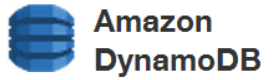
B. Calder, et al. "Windows Azure Storage: a highly available cloud storage service with strong consistency." , SOSP 2011

DynamoDB

- Successor to **SimpleDB**

Limitations:

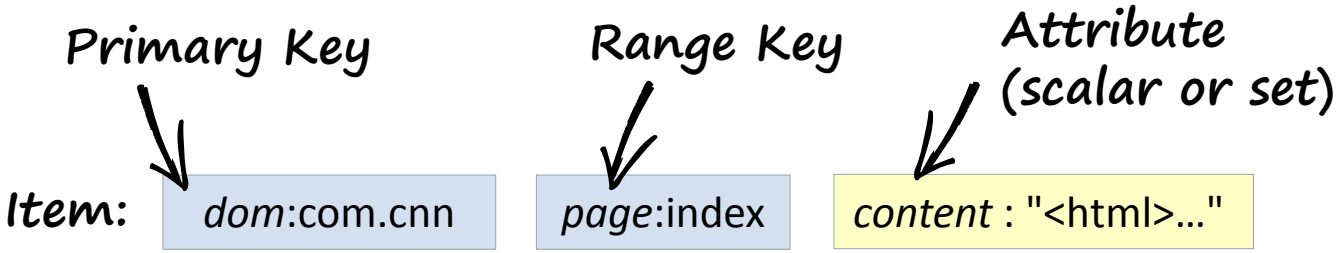
- **Slow** (~50-100 RPS)
- **10 GB** per Domain
- Query result max. 2500 records or 1 MB
- Max. 1K-sized attributes



DynamoDB
Model:
Proprietary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

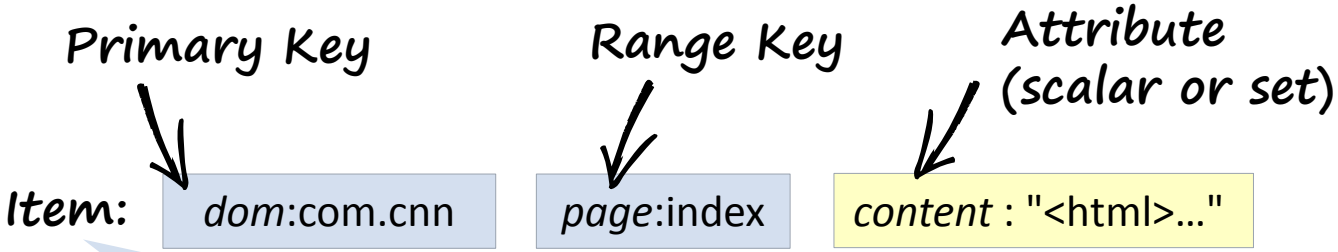
- Successor to **SimpleDB**



DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

- Successor to **SimpleDB**



Querying Options:

- `GetItem`: Key Lookup
- `Query`: Primary Key + Condition on Range Key
- `Scan`: Full Table Scan with filter
- **EMR**: Hive queries (for analytics)

Amazon DynamoDB

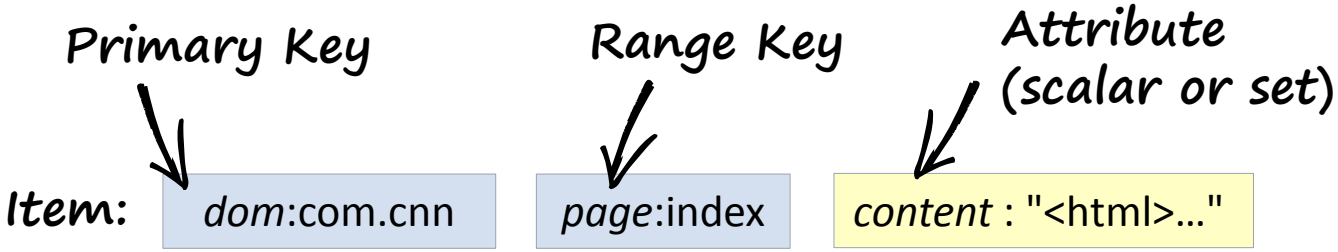
DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

Amazon DynamoDB

DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

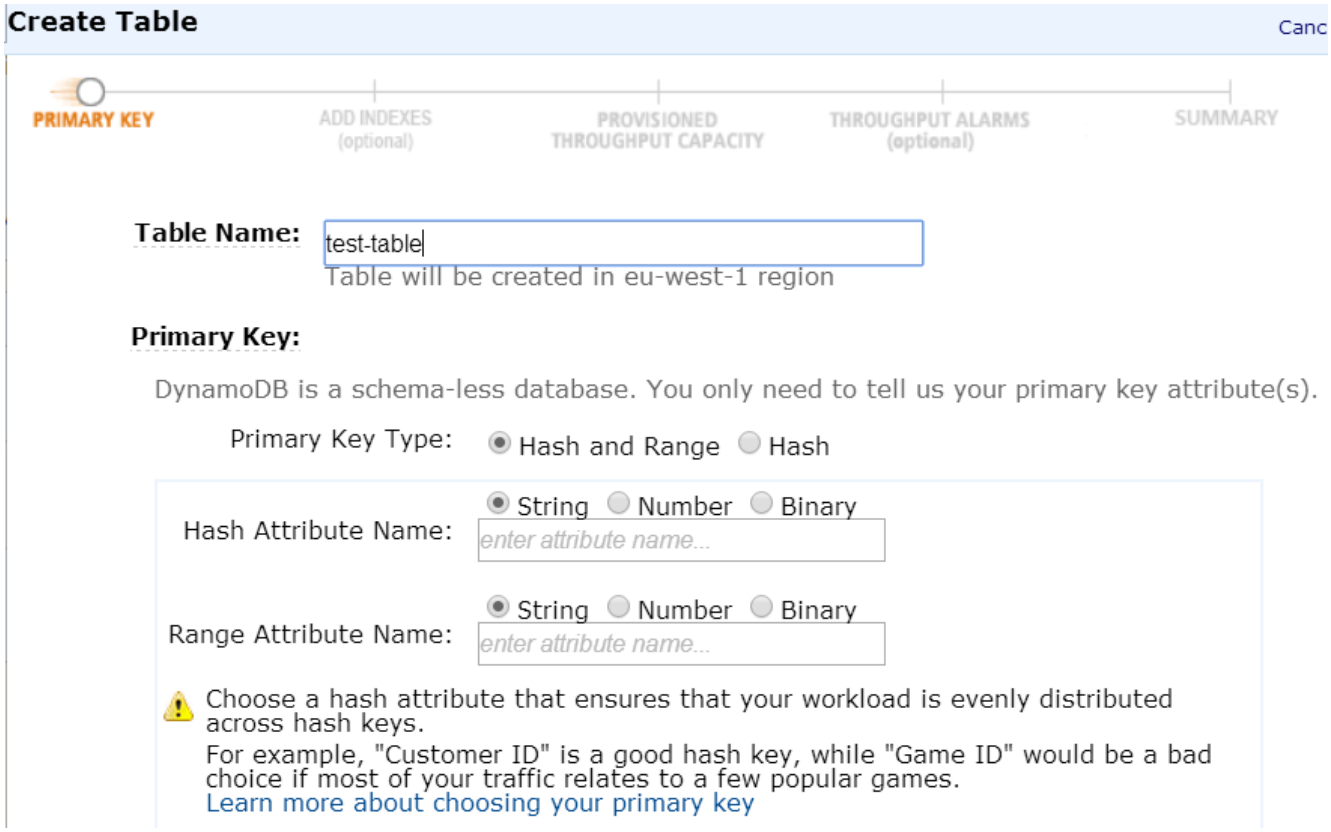
- Successor to **SimpleDB**



- **Consistency:**
  - *Strongly (*2x price*)* or *Eventually* Consistent Reads
  - Atomic (Conditional) Updates per Item
- **Indexing** Options:
  - **Local** Sec. Index: consistent additional Range Key
  - **Global** Sec. Index: eventually consistent index-table (Primary Key)

DynamoDB

- Successor to **SimpleDB**

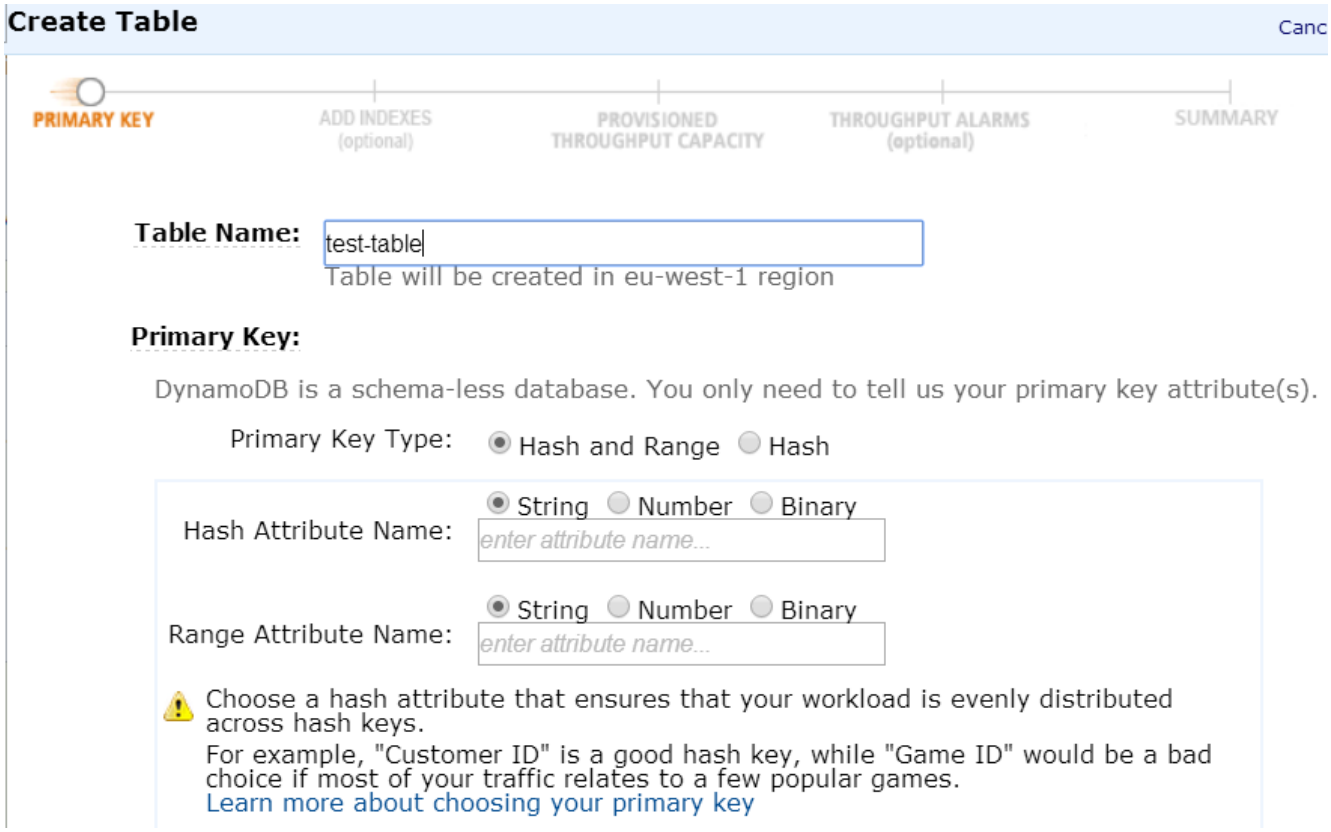


Amazon DynamoDB

DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

- Successor to **SimpleDB**

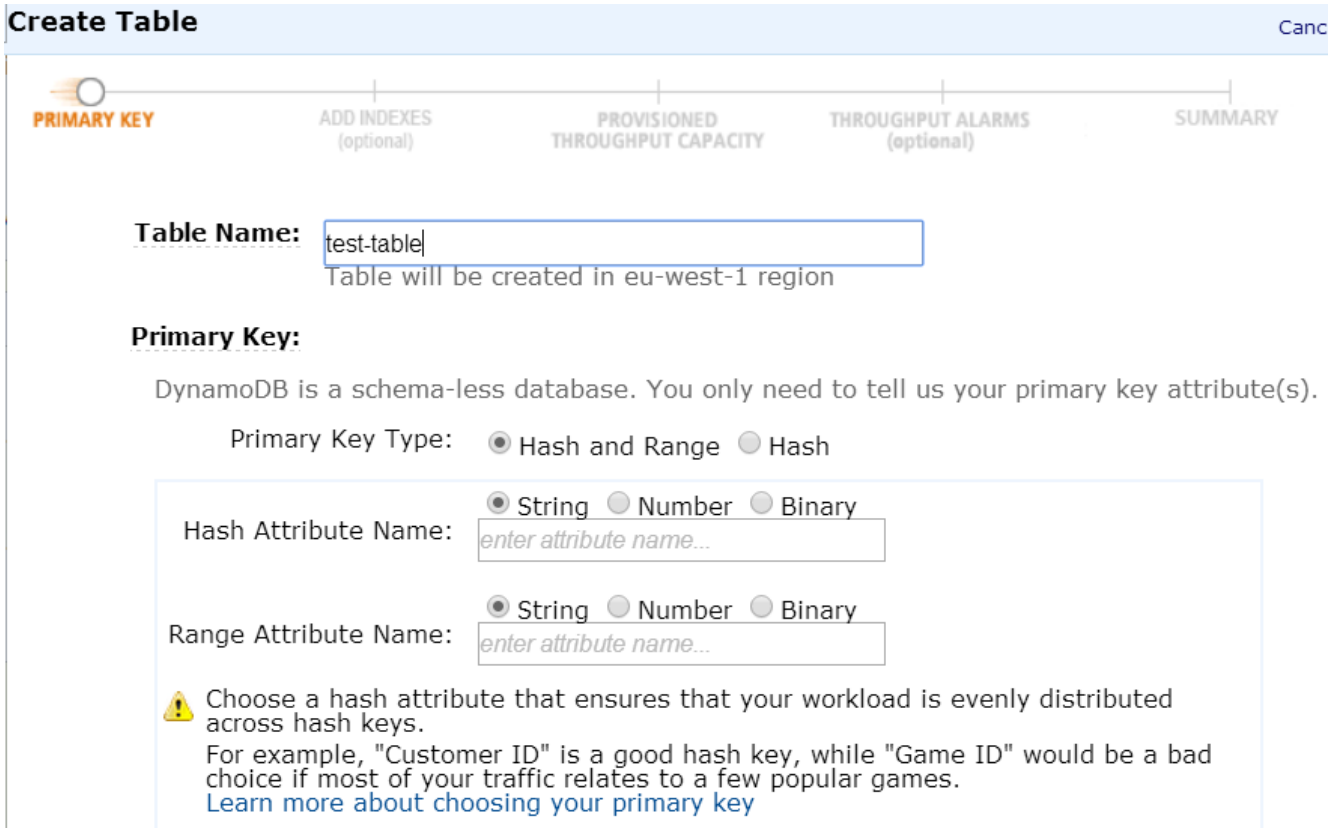


Amazon DynamoDB

DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

- Successor to **SimpleDB**

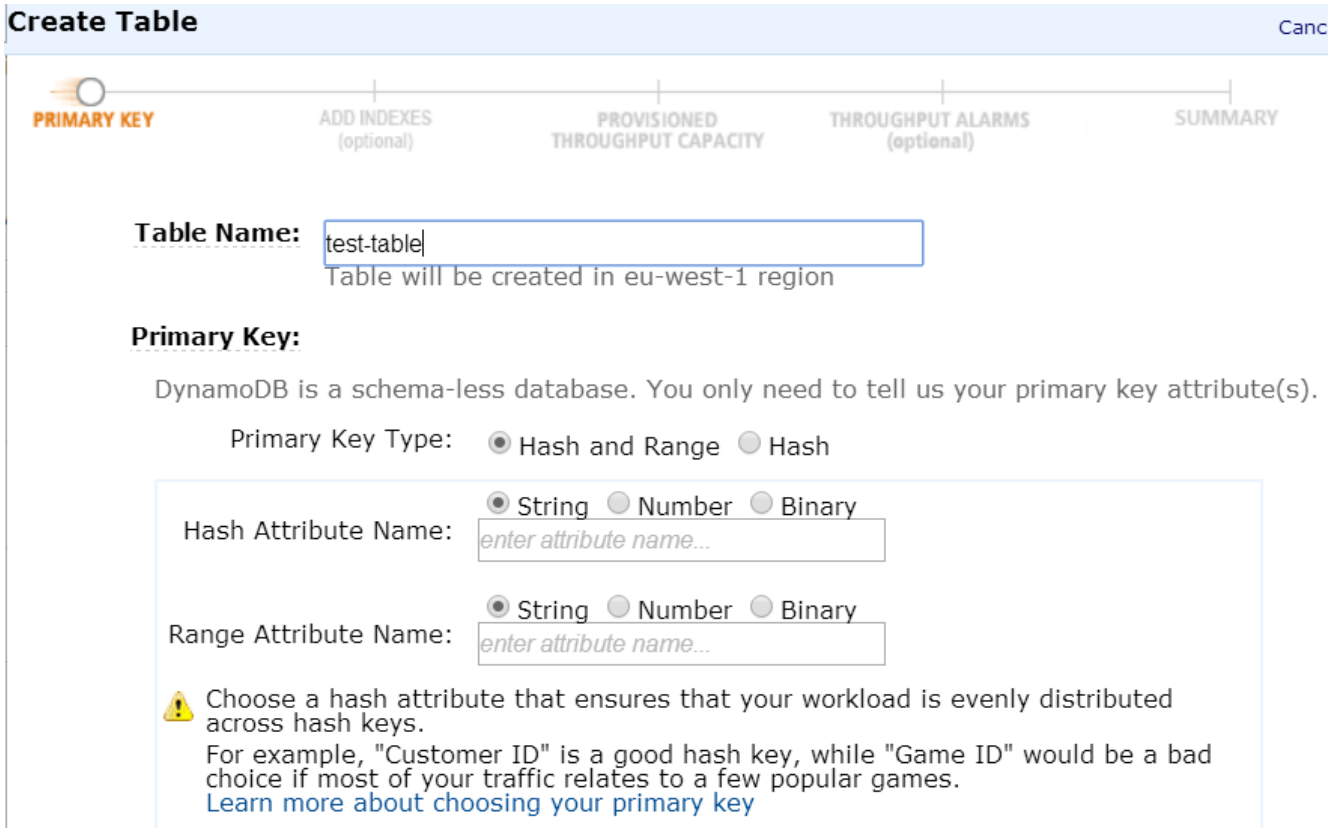


Amazon DynamoDB

DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

- Successor to **SimpleDB**

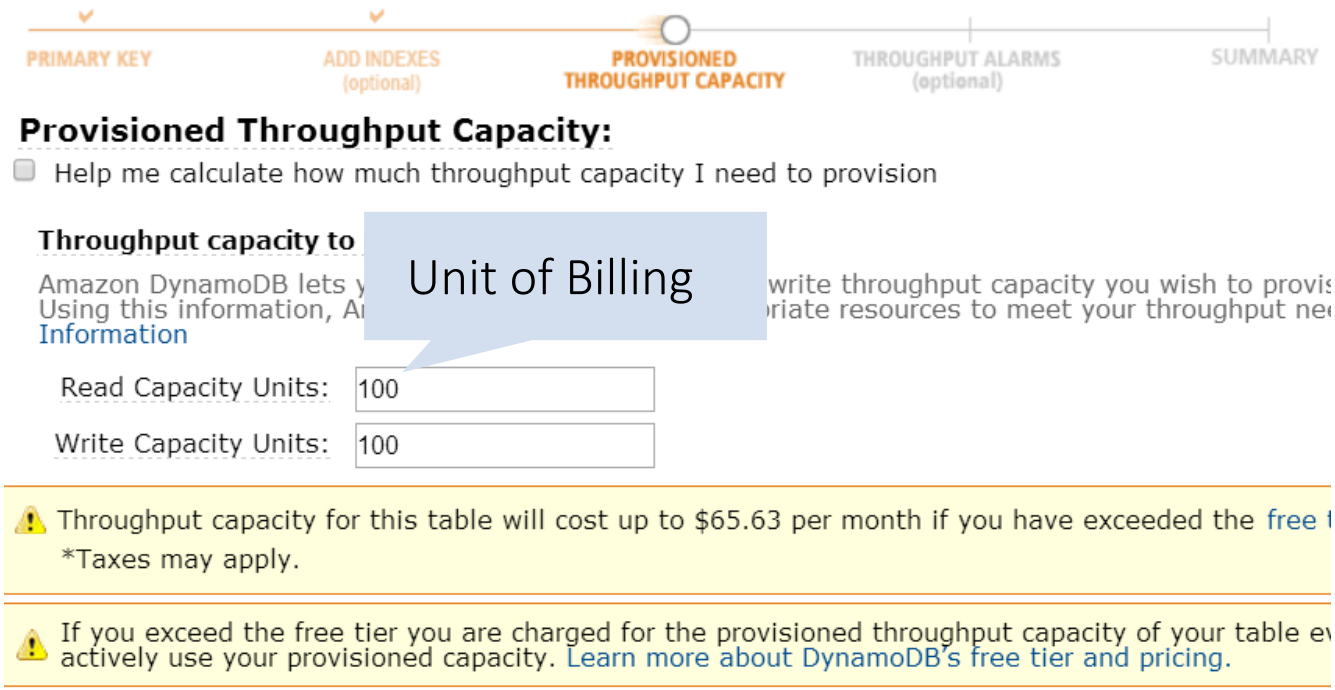


Amazon DynamoDB

DynamoDB
Model:
Propriertary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

- Successor to **SimpleDB**



Amazon DynamoDB

DynamoDB
Model:
Proprietary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

DynamoDB

- Successor to **SimpleDB**

DynamoDB
Model:
Proprietary
Pricing:
Provisioned Throughput + Network
Underlying DB:
Custom System
API:
REST

Good:

- Low Latency (SSD)
- Data partitioning and AZ-replication

Bad:

- Scaling not elastic (Capacity Units)
- No SLAs, no internals published (≠ Dynamo!)
- No built-in backups (→ AWS data pipeline)
- Vendor Lock-in

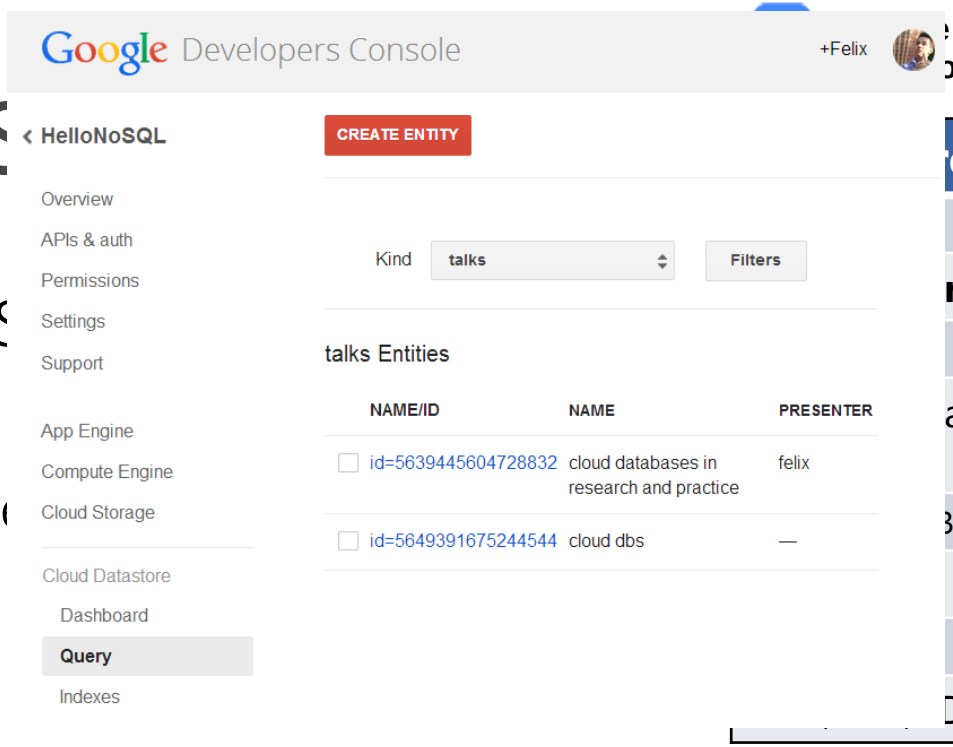
AE/Cloud DataStore

- Structured Storage System for **App Engine**
- Based on:
  - Megastore→ BigTable → Colossus

DataStore
Model:
Proprietary
Pricing:
CPU + Storage + Network
Underlying DB:
MegaStore
API:
SDK, JPA, JDO

AE/Cloud DataS

- Structured Storage S
- Based on:
  - Megastore→ BigTabl



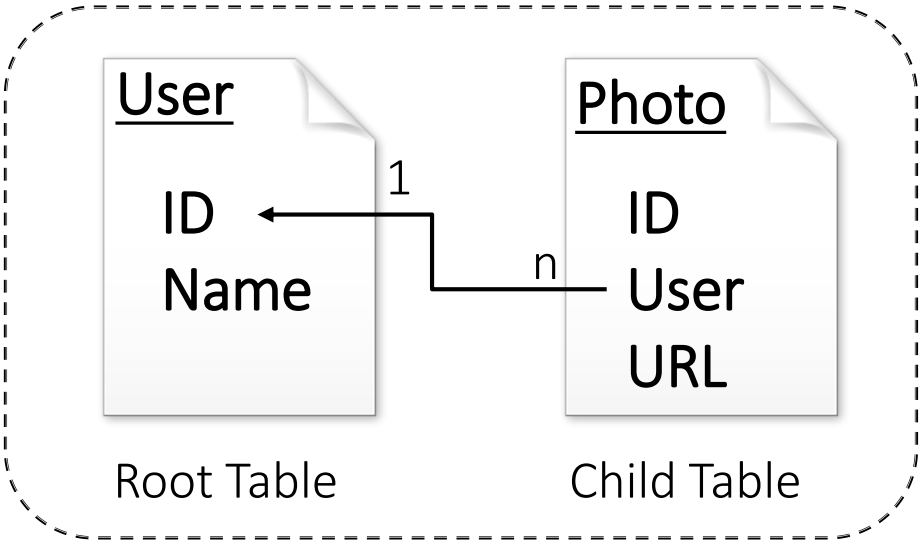
AE/Cloud DataStore

- Structured Storage System for **App Engine**
- Based on:
  - Megastore→ BigTable → Colossus

DataStore
Model:
Proprietary
Pricing:
CPU + Storage + Network
Underlying DB:
MegaStore
API:
SDK, JPA, JDO

AE/Cloud DataStore

- Structured Storage System for **App Engine**
- Based on:
  - Megastore→ BigTable → Colossus
- Schemafree **Entity Group (EG)** data model:



Google Cloud Datastore

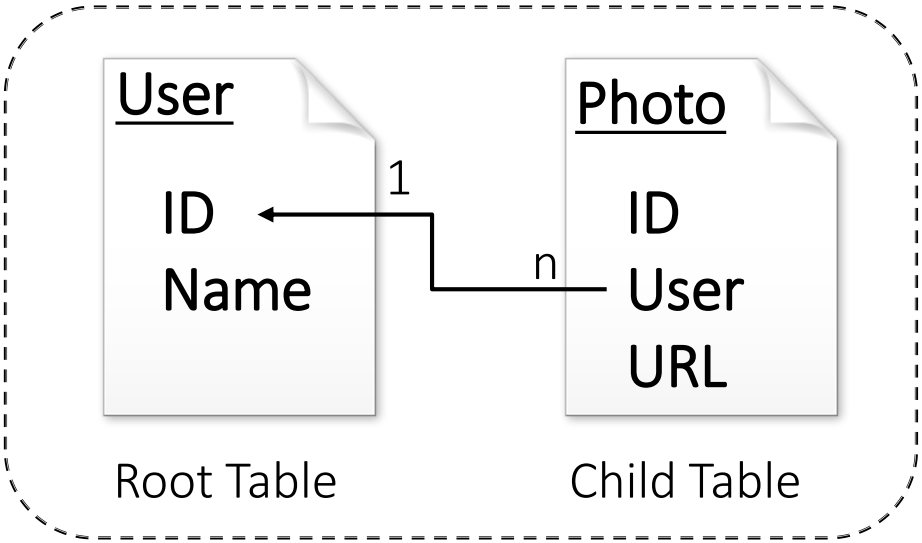
DataStore
Model:
Propriertary
Pricing:
CPU + Storage + Network
Underlying DB:
MegaStore
API:
SDK, JPA, JDO

AE/Cloud DataStore

Google Cloud Datastore

DataStore
Model:
Propriertary
Pricing:
CPU + Storage + Network
Underlying DB:
MegaStore
API:
SDK, JPA, JDO

- Structured Storage System for **App Engine**
- Based on:
  - Megastore→ BigTable → Colossus
- Schemafree **Entity Group (EG)** data model:



EG: User + n Photos

- Unit of ACID **transactions**/ consistency
- Fields autoindexed (eventually consistent)

AE/Cloud DataStore

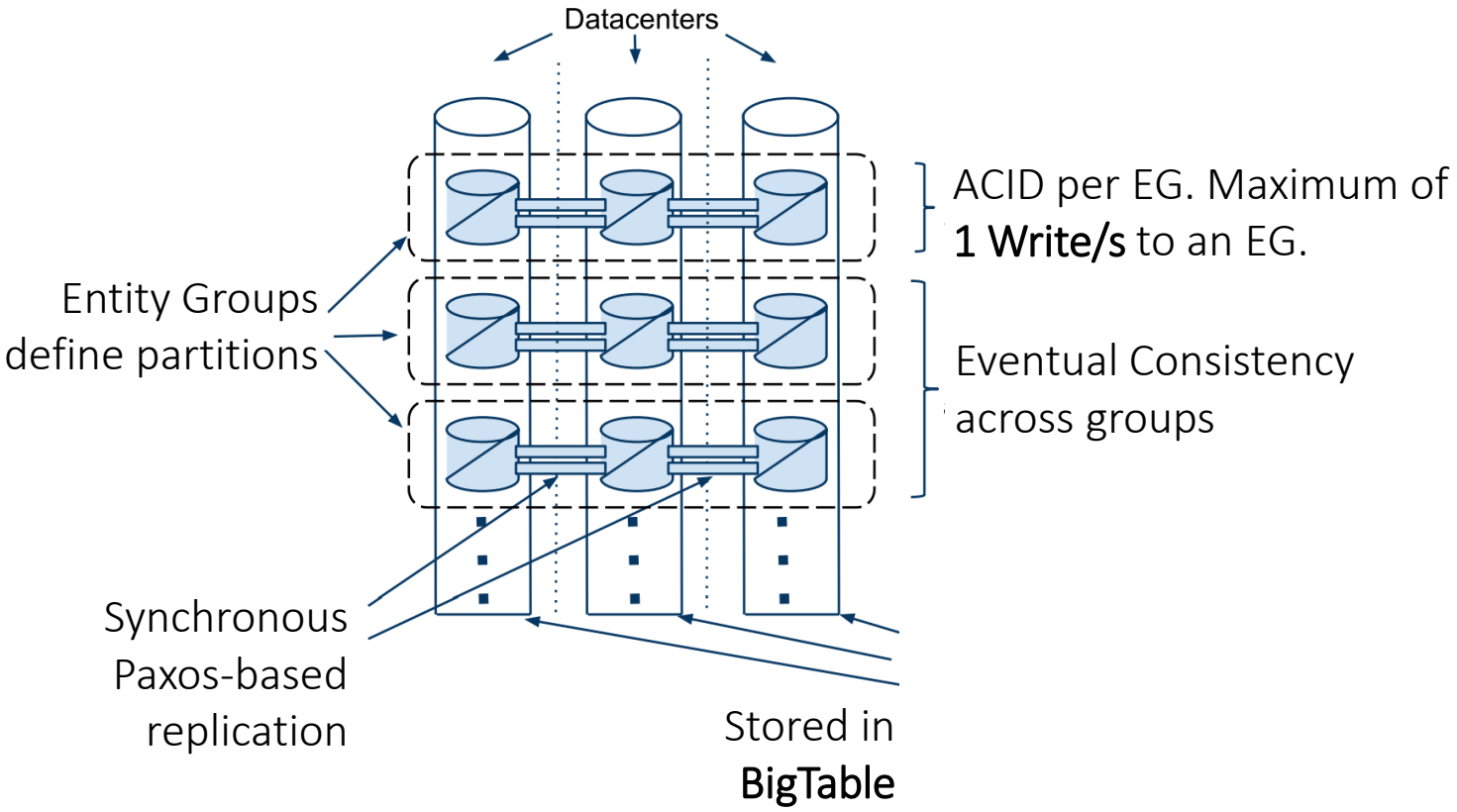
DataStore
Model:
Propriertary
Pricing:
CPU + Storage + Network
Underlying DB:
MegaStore
API:
SDK, JPA, JDO

- Structured Storage System for **App Engine**
- Based on:
  - Megastore→ BigTable → Colossus
- Schemafree **Entity Group (EG)** data model:

```
SELECT * FROM photos
WHERE ANCESTOR IS :34 AND name = „sunset“
ORDER BY date ASC
LIMIT 10
OFFSET 10
```

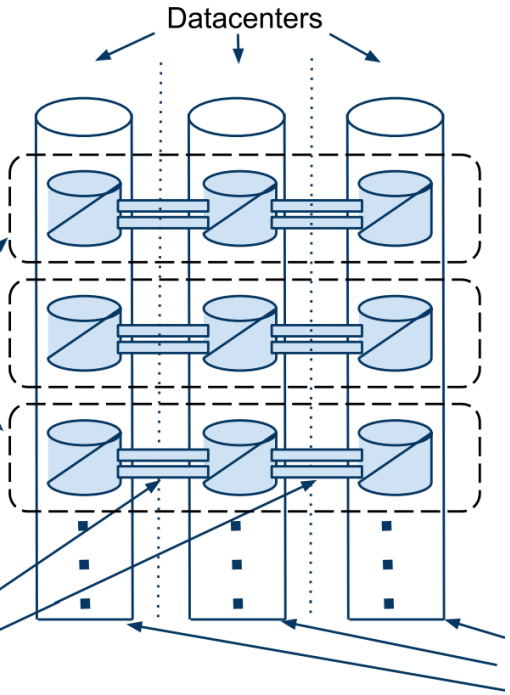
AE/Cloud DataStore

- Internally:



AE/Cloud DataStore

- Internally:



ACID per EG. Maximum of **1 Write/s** to an EG.

Entity Groups define partitions

Eventual Consistency across groups

Synchronous Paxos-based replication

Stored in **BigTable**

Row key	User. name	Photo. time	Photo. tag	Photo. _url
101	John			
101,500		12:30:01	Dinner, Paris	...
101,502		12:15:22	Betty, Paris	...
102	Mary			

AE/Cloud DataStore

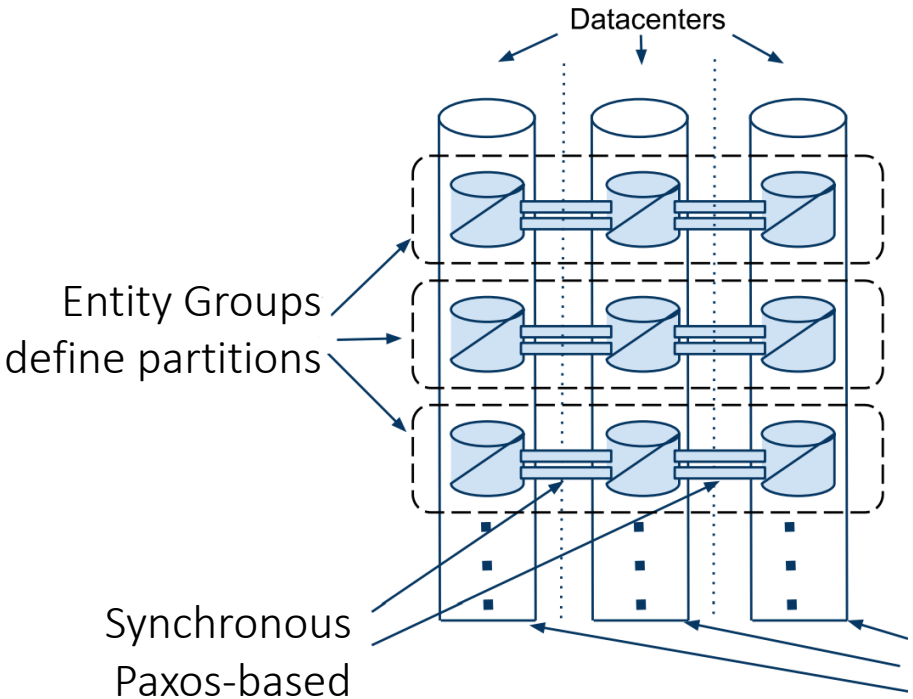
- Internally:

MegaStore

- Paxos-based replication and transactions
- 100 Google applications

Problems: Slow Writes, Predefined Entity Groups

J. Baker, et al. "Megastore: Providing Scalable, Highly Available Storage for Interactive Services." CIDR 2011.



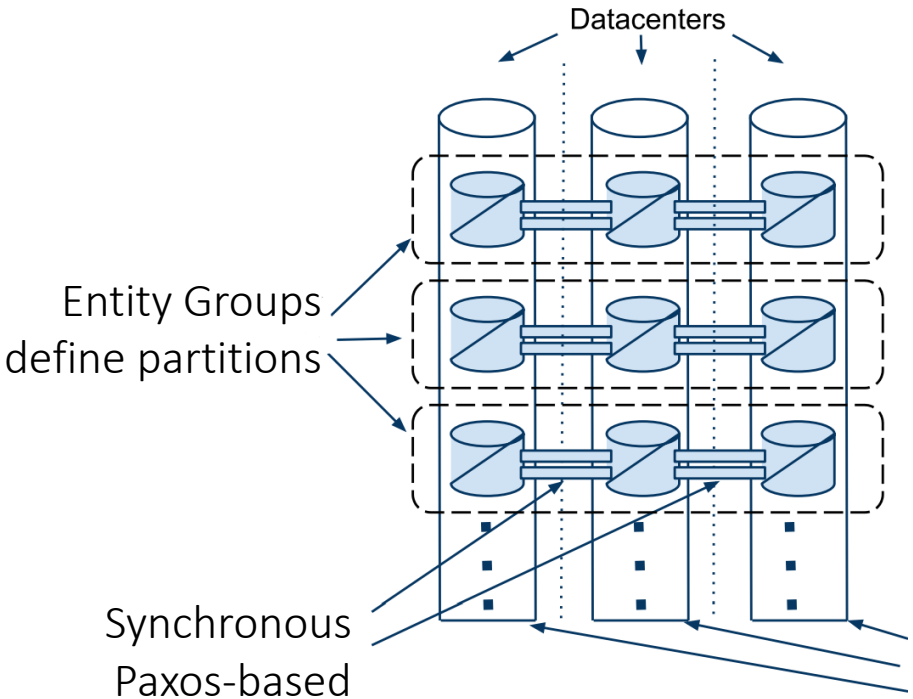
ACID per EG. Maximum of **1 Write/s** to an EG.

Eventual Consistency across groups

Row key	User. name	Photo. time	Photo. tag	Photo. _url
101	John			
101,500		12:30:01	Dinner, Paris	...
101,502		12:15:22	Betty, Paris	...
102	Mary			

AE/Cloud DataStore

- Internally:



Row key	
101	
101,500	
101,502	
102	

MegaStore

- Paxos-based replication and transactions
- 100 Google applications

Problems: Slow Writes, Predefined Entity Groups

J. Baker, et al. "Megastore: Providing Scalable, Highly Available Storage for Interactive Services." CIDR 2011.

Spanner

Idea:

- Autosharded Entity Groups
- *Not* based on BigTable

Implementation:

- **TrueTime** API (GPS + atomic clocks) → commit timestamps of 2PL-SI transactions
- Paxos-replication per Shard

J. Corbett et al. "Spanner: Google's globally distributed database." TOCS 2013

AE/Cloud DataStore

MegaStore

- Paxos-based replication and transactions
- 100 Google applications

Problems: Slow Writes, Predefined Entity Groups

J. Baker, et al. "Megastore: Providing Scalable, Highly Available Storage for Interactive Services." CIDR 2011.

F1

Idea:

- Full SQL relational database built on Spanner, powers AdWords

Implementation:

- 5-way replication
- **Data Model**: relational + hierarchy (customer→campaign→AdGroup)
- **Transactions**: Snapshot-Read-Only, Pessimistic (Spanner), optimistic
- Distributed SQL Engine

J. Shute, et al. "F1: A distributed SQL database that scales.", VLDB 2013

ACID p

1 Write

Eventu

across

Row key
101
101,500
101,502
102

Spanner

Idea:

- Autosharded Entity Groups
- *Not* based on BigTable

Implementation:

- **TrueTime** API (GPS + atomic clocks) → commit timestamps of 2PL-SI transactions
- Paxos-replication per Shard

J. Corbett et al. "Spanner: Google's globally distributed database." TOCS 2013

AE/Cloud DataStore

MegaStore

- Paxos-based replication and transactions
- 100 Google applications

Problems: Slow Writes, Predefined Entity Groups

J. Baker, et al. "Megastore: Providing Scalable, Highly Available Storage for Interactive Services." CIDR 2011.

F1

Idea:

- Full SQL relational database built on Spanner, powers AdWords

Good:

- Transactions (though limited)
- Good scalability of **data volume**

Bad:

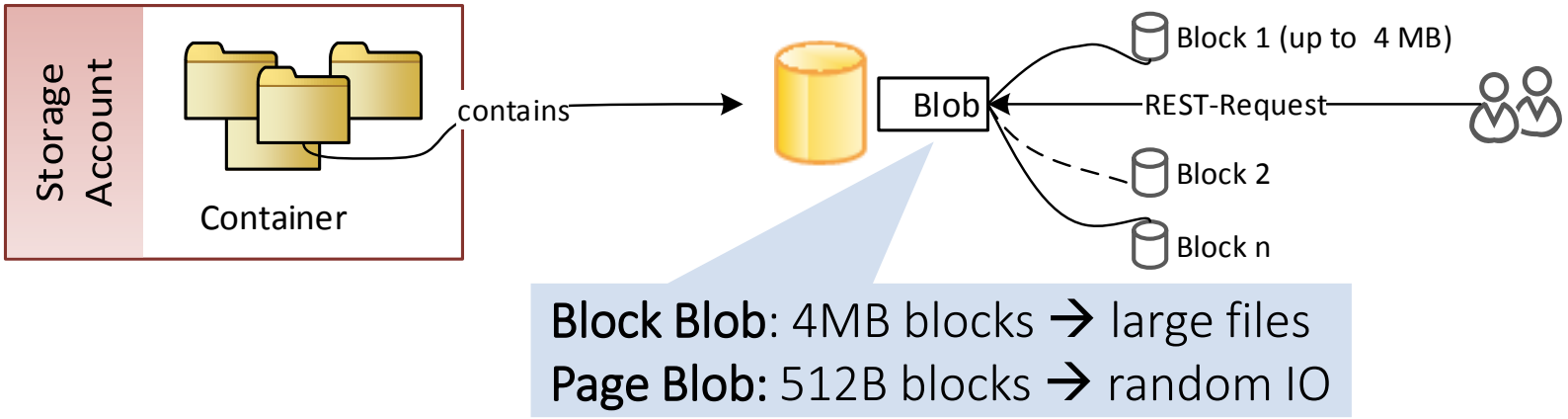
- **Entity Groups** hard to define
- Bad Scale-Out for **write and reads** (Kossmann et al.) → no advantage over RDBMS for small data volumes

A Spanner/F1 based DBaaS?

Azure Blobs, Amazon S3, Google Cloud Storage

- **Idea**: Blobs with RESTful CRUD Interface
- **Distribution**: $(Bucket, Key) \rightarrow Server + Replicas$
- CDN Integration (Azure CDN, Cloudfront)
- **S3**: > 2 Trillion ($2 \cdot 10^{12}$) objects

Azure Blobs:

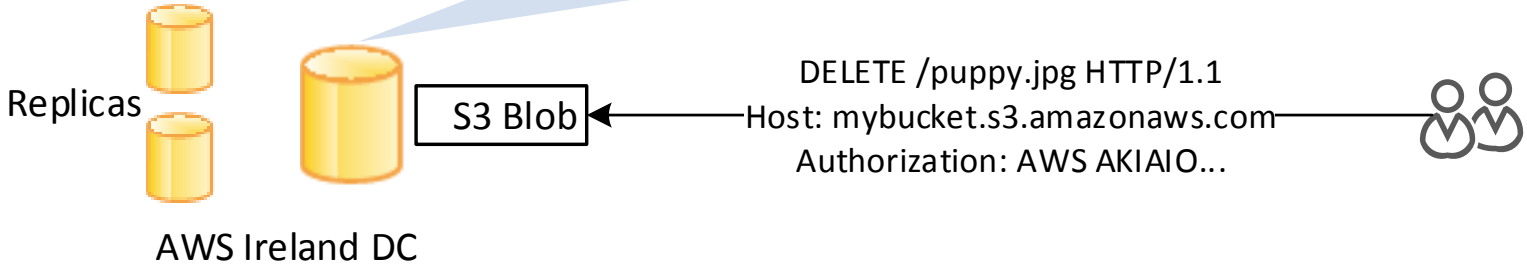


Azure Blobs, Amazon S3, Google Cloud Storage

- Automatic Versioning
- **Pricing**: per request (10.000 ~ 1c) and storage (1GB/month ~ 3c) and network (1GB ~ 10c)

Amazon S3:

Reduced Redundancy: only 1 replica
Glacier: tape-disk archival



Azure Blobs, Amazon S3, Google Cloud Storage

- Automatic Versioning
- **Pricing**: per request (10.000 ~ 1c) and storage (1GB/month ~ 3c) and network (1GB ~ 10c)

S3 Consistency

Findings:

- *Inconsistency window* varies from **2-11 seconds**
- *Monotonic Read Consistency* is often violated

D. Bermbach,, S. Tai. "Eventual consistency: How soon is eventual?", MW4SOC 11

ndancy: only 1 replica
disk archival

DELETE /puppy.jpg HTTP/1.1
Host: mybucket.s3.amazonaws.com
Authorization: AWS AKIAIO...

Azure Blobs, Amazon S3, Google Cloud Storage

- Automatic Versioning
- **Pricing**: per request (10.000 ~ 1c) and storage (1GB/month ~ 3c) and network (1GB ~ 10c)

S3 Consistency

Findings:

- *Inconsistency window* varies from **2-11 seconds**
- *Monotonic Read Consistency* is often violated

D. Bermbach,, S. Tai. "Eventual consistency: How soon is eventual?", MW4SOC 11

ndancy:
disk arch

DEL
Host: m
Aut

Building a database on S3

Idea:

- Use S3 as the persistent storage of a database

Implementation:

- Buffer Pool and Log Manager on S3
- No transaction or query support

M. Brantner, et al. "Building a database on S3." Sigmod 2008

Parse - MBaaS

- Founded June, 2011
- Acquired by Facebook April, 2013
- Pricing:
  - Free
  - Pro (199$)
  - Enterprise

Parse
Model:
Backend-aas
Pricing:
Plan-based
Underlying DB:
Mainly MongoDB
API:
SDKs, REST

Parse - MBaaS

- Founded June, 2011
- Acquired by Facebook April, 2013
- Pricing:
  - Free
  - Pro (199$)
  - Enterprise

Parse Core

Parse
Model:
Backend-aas
Pricing:
Plan-based
Underlying DB:
Mainly MongoDB
API:
SDKs, REST

Parse - MBaaS

- Founded June, 2011
- Acquired by Facebook April, 2013
- Pricing:
  - Free
  - Pro (199$)
  - Enterprise

Parse
Model:
Backend-aas
Pricing:
Plan-based
Underlying DB:
Mainly MongoDB
API:
SDKs, REST

Parse Core

Parse Analytics

Parse - MBaaS

- Founded June, 2011
- Acquired by Facebook April, 2013
- Pricing:
  - Free
  - Pro (199$)
  - Enterprise

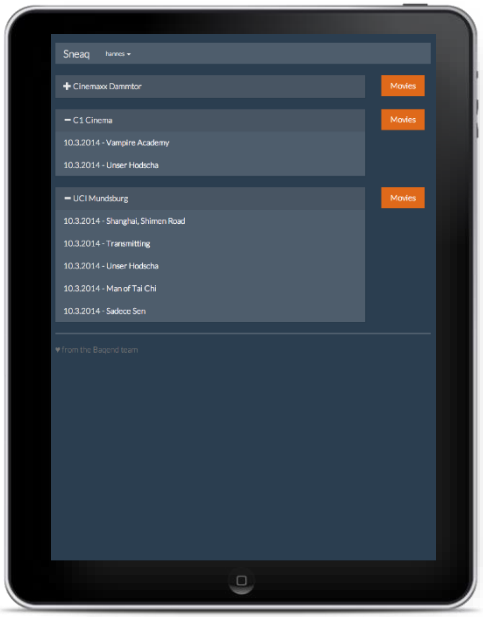
Parse
Model:
Backend-aas
Pricing:
Plan-based
Underlying DB:
Mainly MongoDB
API:
SDKs, REST

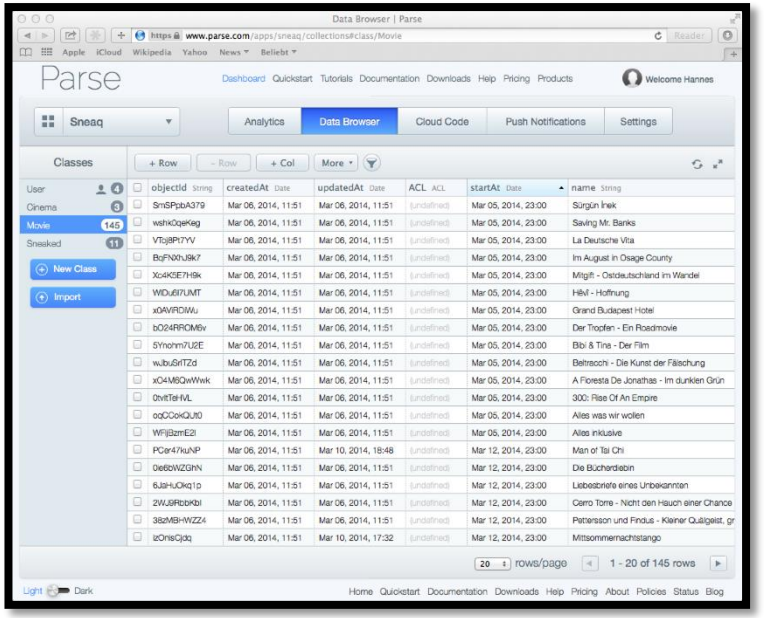
Parse Core

Parse Analytics

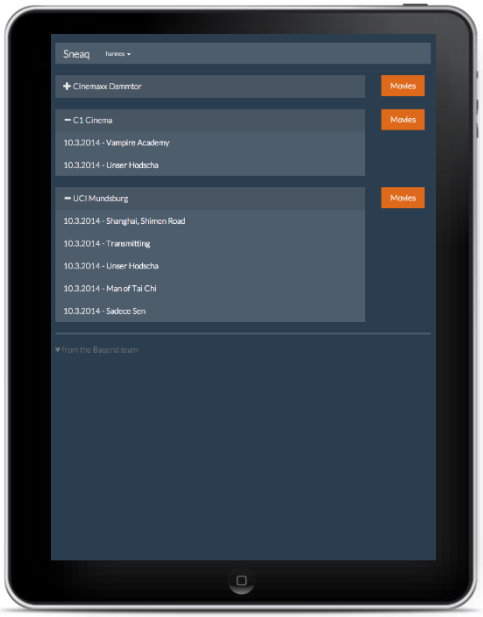
Parse Push

Parse - MBaaS


Sneaq
hannes
Cinemaxx Dammtor
Movies
C1 Cinema
Movies
10.3.2014 - Vampire Academy
10.3.2014 - Unser Hodscha
UCI Mundsburg
Movies
10.3.2014 - Shanghai, Shimen Road
10.3.2014 - Transmitting
10.3.2014 - Unser Hodscha
10.3.2014 - Man of Tai Chi
10.3.2014 - Sadece Sen
from the Backend team


Data Browser | Parse
www.parse.com/apps/sneaq/collections#class/Movie
Parse
Dashboard Quickstart Tutorials Documentation Downloads Help Pricing Products
Welcome Hannes
Sneaq
Analytics
Data Browser
Cloud Code
Push Notifications
Settings
Classes
+ Row
- Row
+ Col
More
User 4
Cinema 3
Movie 145
Sneakad 11
New Class
Import
objectId String
createdAt Date
updatedAt Date
ACL ACL
startAt Date
name String
SmSPpbA379 Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Sürgün Inek
wshK0qeKeg Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Saving Mr. Banks
VTpj8Pt7YV Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 La Deutsche Vita
BqFN0tJ9k7 Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Im August in Osage County
Xc4K5E7H9k Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Mitgift - Ostdeutschland im Wandel
WDu6I7UMT Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Hévl - Hoffnung
x0AVRDlWu Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Grand Budapest Hotel
bO24RROM6v Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Der Tropfen - Ein Roadmovie
5Ynohm7U2E Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Bibi & Tina - Der Film
wJbuSrlTZd Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Beltracchi - Die Kunst der Fälschung
xO4M8QwWwk Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 A Floresta De Jonathas - Im dunklen Grün
0tvltTeHVL Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 300: Rise Of An Empire
oqCCokQUt0 Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Alles was wir wollen
WFIjBzmE2I Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 05, 2014, 23:00 Alles inklusive
PCar47kuNP Mar 06, 2014, 11:51 Mar 10, 2014, 18:48 (undefined) Mar 12, 2014, 23:00 Man of Tai Chi
0e6bWZGhN Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 12, 2014, 23:00 Die Bücherdiebin
6JsHuOKq1p Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 12, 2014, 23:00 Liebesbriefe eines Unbekannten
2WJ9RbbKbl Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 12, 2014, 23:00 Cerro Torre - Nicht den Hauch einer Chance
38zMBHWZZ4 Mar 06, 2014, 11:51 Mar 06, 2014, 11:51 (undefined) Mar 12, 2014, 23:00 Pettersson und Findus - Kleiner Quälgeist, gr
izOnlsCjdq Mar 06, 2014, 11:51 Mar 10, 2014, 17:32 (undefined) Mar 12, 2014, 23:00 Mittsommernachtstango
20 rows/page
1 - 20 of 145 rows
Light Dark
Home Quickstart Documentation Downloads Help Pricing About Policies Status Blog

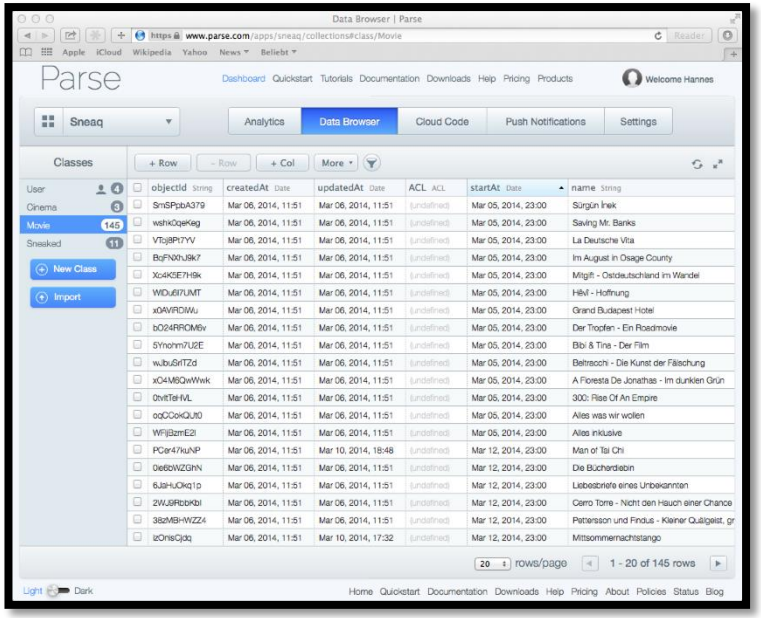
Parse - MBaaS



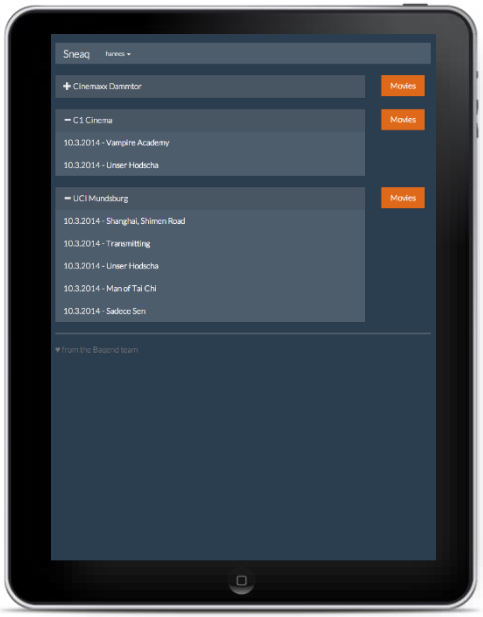
Authentication

User + Password

OAuth: Facebook, Twitter



Parse - MBaaS



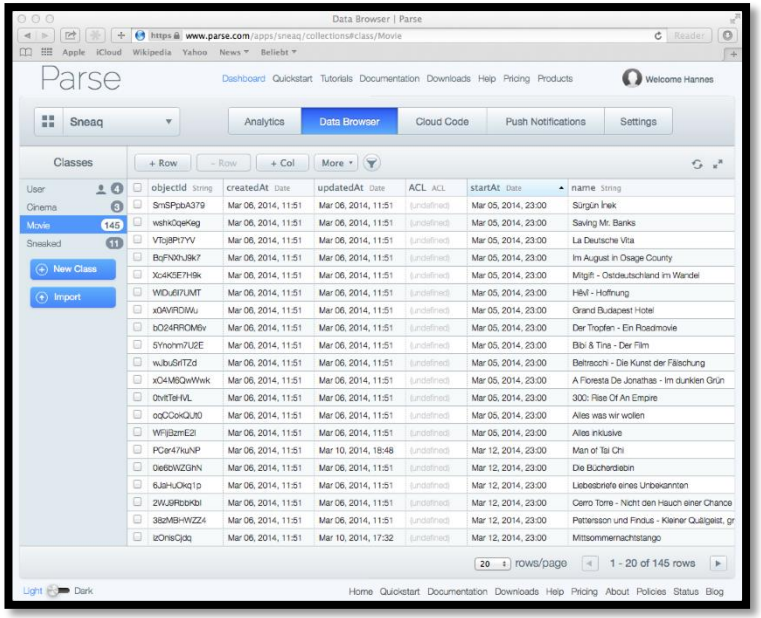
Authentication

User + Password

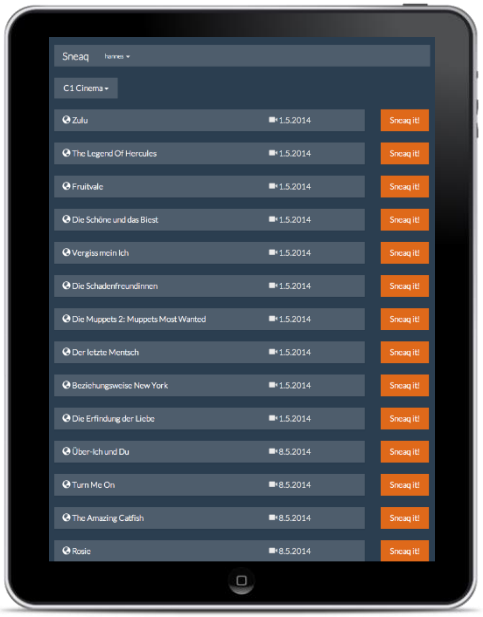
OAuth: Facebook, Twitter

Query Cinemas

```
(new Parse.Query('Cinemas'))
  .withinKilometers(...)
  .fetch()
```



Parse - MBaaS



Authentication

User + Password

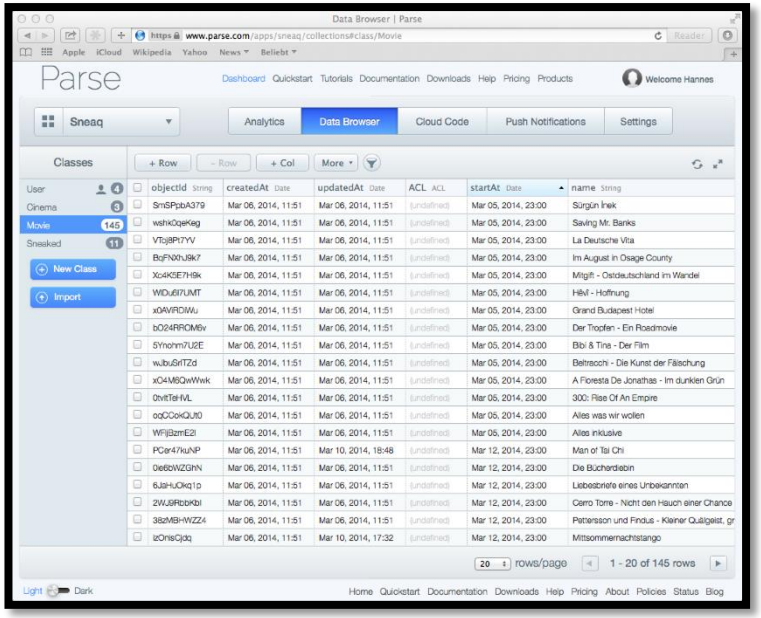
OAuth: Facebook, Twitter

Query Cinemas

```
(new Parse.Query('Cinemas'))
  .withinKilometers(...)
  .fetch()
```

Query Movies

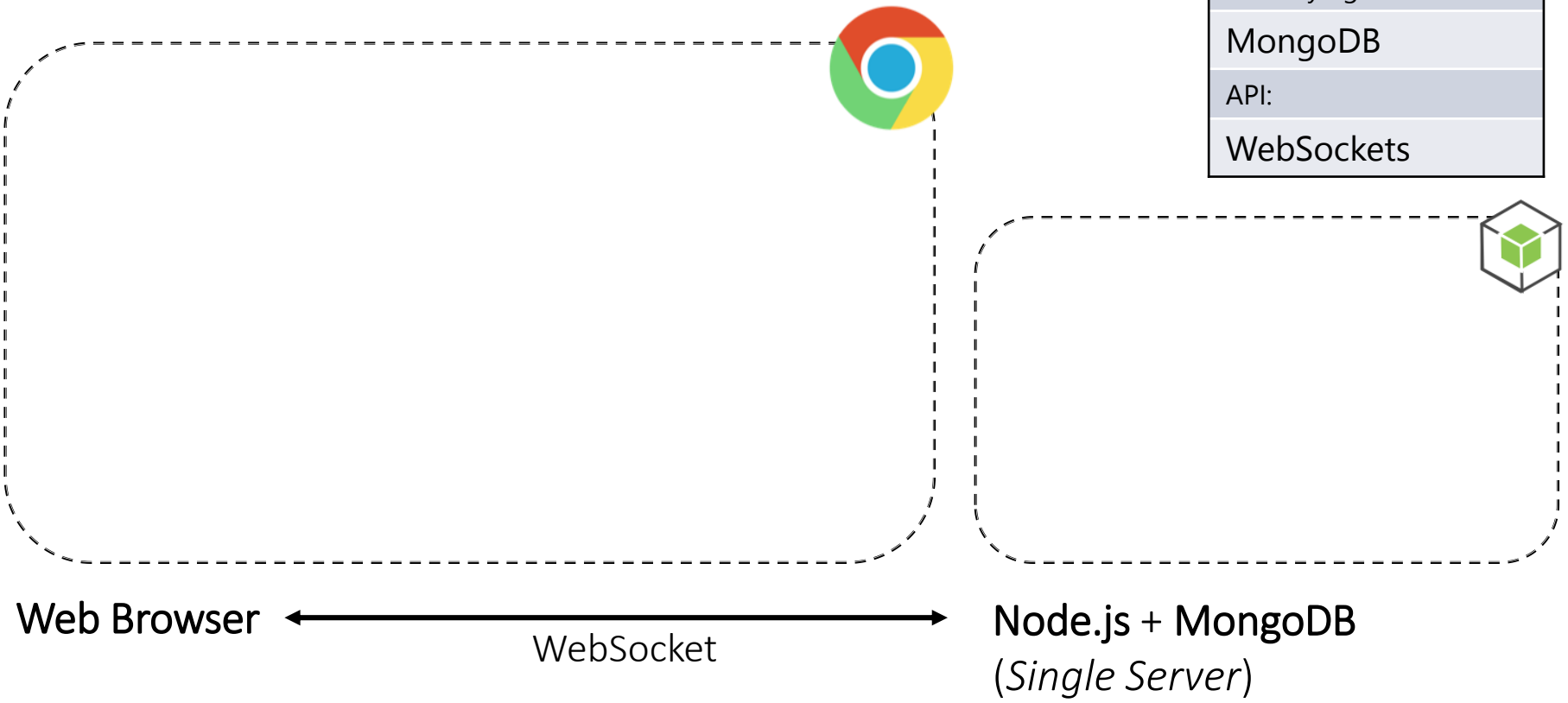
```
(new Parse.Query('Movies'))
  .greaterThan('startAt', now)
  .notEqualTo('cinemas', cId)
  .fetch()
```



Meteor

- **Idea**: Full-Stack JavaScript with **Node.js**, **MongoDB** and **WebSockets**

Meteor
Model:
Backend-aaS
Pricing:
No yet revealed
Underlying DB:
MongoDB
API:
WebSockets



Meteor

- **Idea**: Full-Stack JavaScript with **Node.js**, **MongoDB** and **WebSockets**

Meteor
Model:
Backend-aaS
Pricing:
No yet revealed
Underlying DB:
MongoDB
API:
WebSockets

Web Browser ⟷ WebSocket ⟷ Node.js + MongoDB (*Single Server*)

Meteor PREVIEW

Meteor

- **Idea**: Full-Stack JavaScript with **Node.js**, **MongoDB** and **WebSockets**

Meteor
Model:
Backend-aaS
Pricing:
No yet revealed
Underlying DB:
MongoDB
API:
WebSockets

```
<div class="player {{selected}}">
    <span class="name">{{name}}</span>
    <span class="score">{{score}}</span>
</div>
```

```
Players = new Meteor.
          Collection("players");

if (Meteor.isServer) {
   //…
```

Web Browser ⟷ WebSocket ⟷ Node.js + MongoDB
(*Single Server*)

Meteor

- **Idea**: Full-Stack JavaScript with **Node.js**, **MongoDB** and **WebSockets**

Meteor
Model:
Backend-aaS
Pricing:
No yet revealed
Underlying DB:
MongoDB
API:
WebSockets

```
<div class="player {{selected}}">
    <span class="name">{{name}}</span>
    <span class="score">{{score}}</span>
</div>

if (Meteor.isClient) {
  Template.leaderboard.players = function () {
    return Players.find({},
                  {sort: {score: -1, name: 1}});
 };
```

```
Players = new Meteor.
          Collection("players");

if (Meteor.isServer) {
   //…
```

Web Browser ⟷ Node.js + MongoDB

WebSocket

(*Single Server*)

Meteor PREVIEW

Meteor

- **Idea**: Full-Stack JavaScript with **Node.js**, **MongoDB** and **WebSockets**

Meteor
Model:
Backend-aaS
Pricing:
No yet revealed
Underlying DB:
MongoDB
API:
WebSockets

```
<div class="player {{selected}}">
    <span class="name">{{name}}</span>
    <span class="score">{{score}}</span>
</div>

if (Meteor.isClient) {
  Template.leaderboard.players = function () {
    return Players.find({},
                  {sort: {score: -1, name: 1}});
```

```
Players = new Meteor.
          Collection("players");

if (Meteor.isServer) {
```

```
$ meteor deploy abc.meteor.com
```

Web Browser ⟷ WebSocket ⟷ Node.js + MongoDB (*Single Server*)

Meteor

Meteor
Model:
Backend-aaS
Pricing:
No yet revealed
Underlying DB:
MongoDB
API:
WebSockets

- **Idea**: Full-Stack JavaScript with **Node.js**, **MongoDB** and **WebSockets**

```
<div class="player {{selected}}">
    <span class="name">{{name}}</span>
    <span class="score">{{score}}</span>
</div>
```

Very *productive* for very *small* projects

Fundamentally **limited scalability**:

- Server tails Mongo‘s oplog
- And holds the fetched data state of every client

Outline

What are Cloud Databases?

Cloud Databases in the wild

Research Perspectives

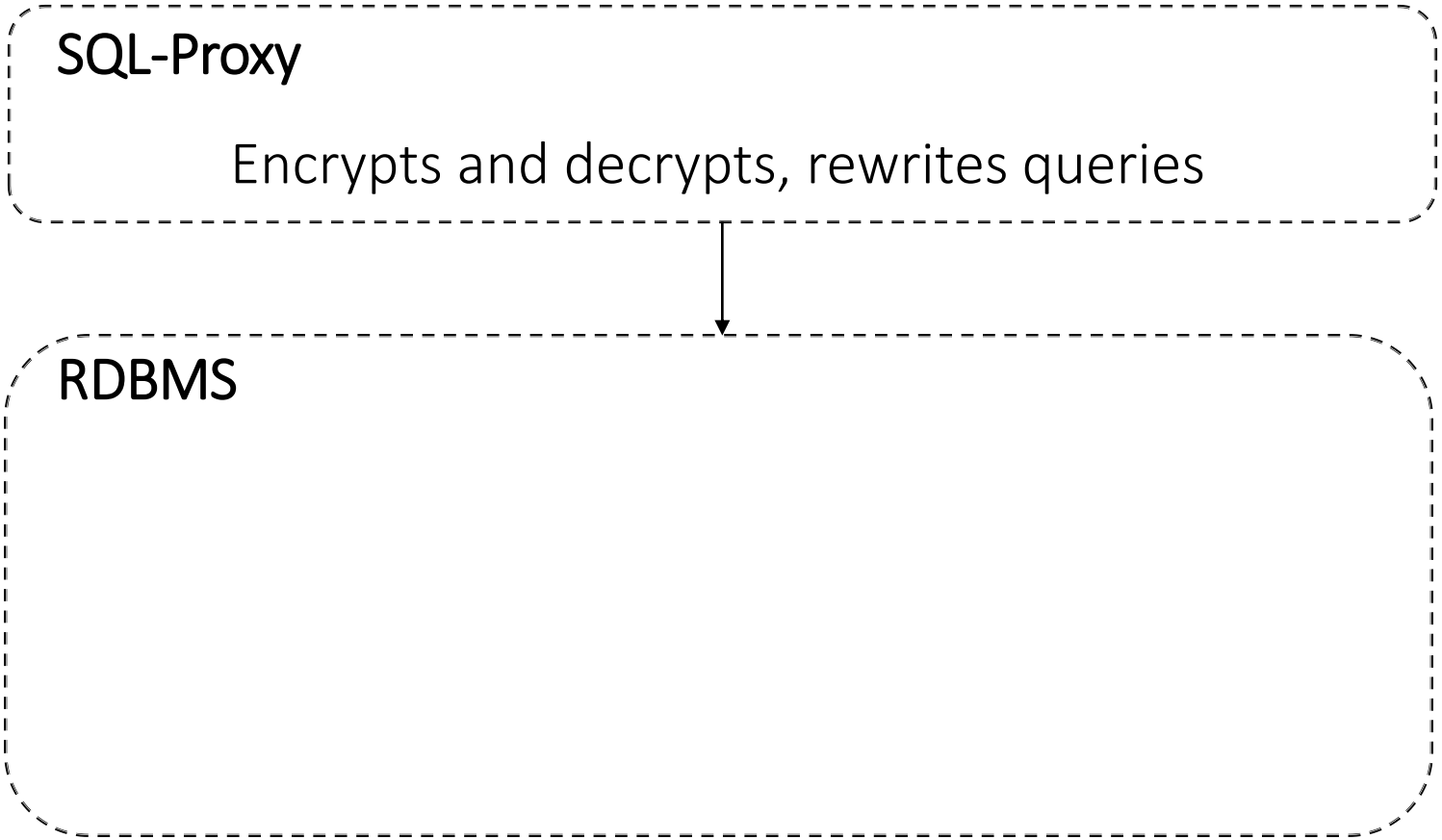
Wrap-up and literature

- Hot Topics
- Orestes: a scalable, low-latency architecture
- Baqend: putting it into practice

DATABASE
DATA
DISTRIBUTED
Performance
NOSQL
CLOUD
Caching
TRANSACTIONS
APPLICATIONS
WEB
Consistency
SURVEY
SYSTEMS
Service
Storage
Concurrency
MANAGEMENT
SCALABLE
STORES
CLUSTERING
AVAILABLE
HIGHLY
High
TRANSACTION
MODELING
EVALUATION
Scale
VIRTUALIZATION
CONTENT
COMPARISON
PREDICTION
BUILDING
Memory
SPECTRAL
OPTIMISTIC
VALUE
ANALYSIS
Approaches
CLASSIFICATION
Internet
LOAD
ISOLATION
IMPROVING
Replication
FILTERS
OBJECT
USING
MODERN
BUSINESS
LIMITATIONS
PROCESSING
BLOOM
RELATIONAL
NETWORK
STRUCTURES
MAPREDUCE
BASED
ORIENTED
Platforms
Datenbanksysteme
ELASTIC
GOOGLES
ARCHITECTURE
TECHNOLOGY
TREE
FRAMEWORK
FREQUENCY
CAP
CASE
SERVERS
PARTITION
ANALYTICS
ALGORITHMS
IMPLEMENTATIONS
techniques
computing
challenges
Approximate
Prefetching
KEY
LARGE
Shared
Multi
Cassandra
SQL
PERSPECTIVES
PERSISTENCE
MIGRATION
STREAMING
Software
Replacement
Snapshot
Paxos
CONSENSUS
AWARE
INVESTIGATION
mapping
TENANT
Taxonomy
PEER
Main
PROVIDERS
OPERATION
DATASETS
ENABLING
Causal
BENCHMARKING
History
HASHING
TOLERANT
EVENTUALLY
MEASUREMENT
look
MACHINE
Live
RESOURCE
Column
Azure
INTERACTIVE
Alternative
Simple
OVERVIEW
DELIVERY
FORBUILDING
LIGHTWEIGHT
LATENCY
Functions
PROVISIONING
MECHANISMS
SERIALIZABLE
SEARCHING
PRACTICAL
Solutions
POLICY
Dynamics
GENERATION
Changed
Engineering
MASTERARBEIT
Democratizing
ANFORDERUNGEN
PROXY
SUPPORTING
HETEROGENEOUS
Theory
PostgreSQL
DUPLICATES
INTELLIGENT
Decentralized
ASYNCHRONOUS
BROADCAST

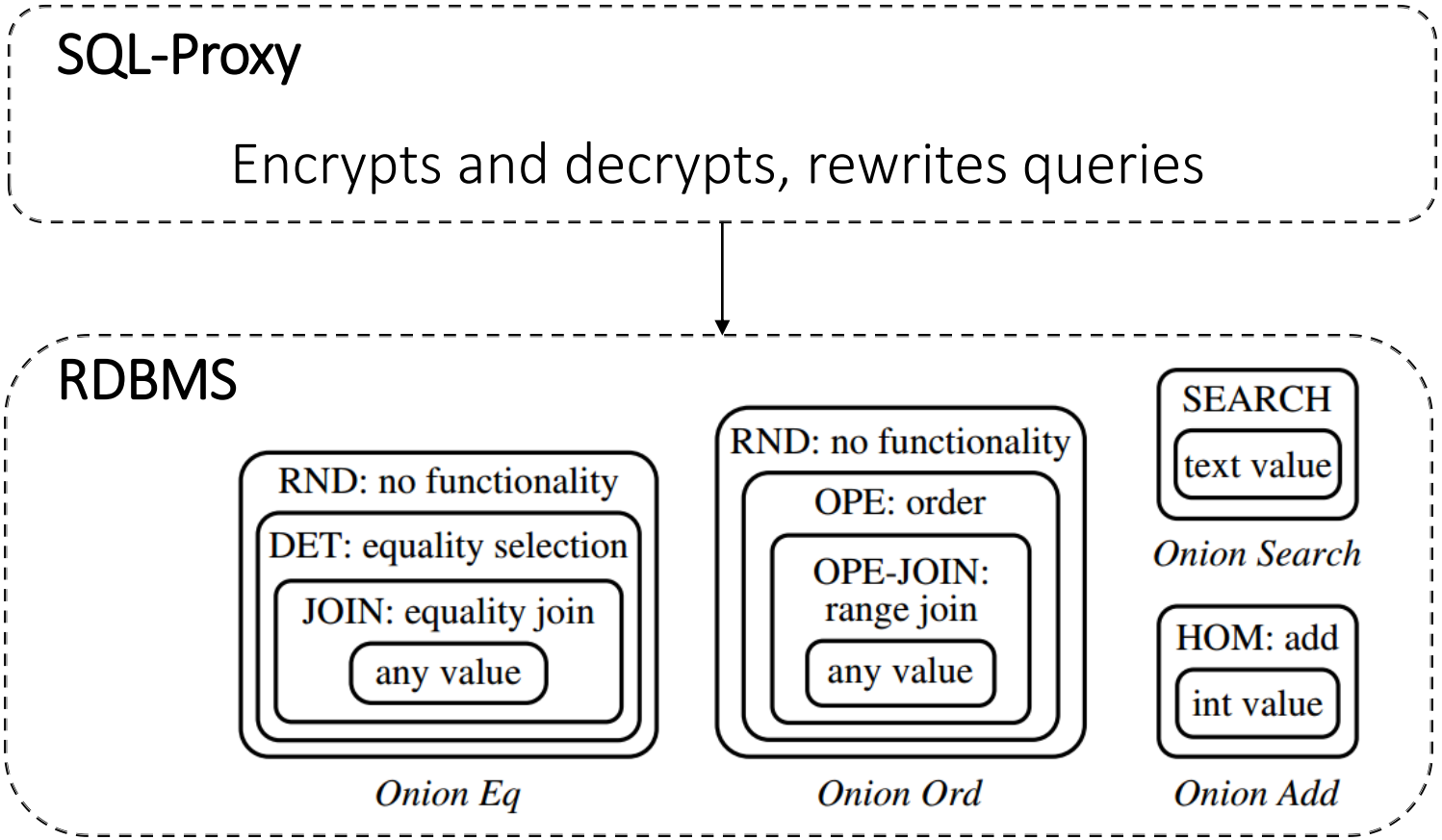
Encrypted Databases: Research

- Example: **CryptDB**
- **Idea:** Only decrypt as much as neccessary



Encrypted Databases: Research

- Example: **CryptDB**
- **Idea:** Only decrypt as much as neccessary



Encrypted Databases: Research

- Example: **CryptDB**
- **Idea:** Only decrypt as much

Relational Cloud

DBaaS Architecture:

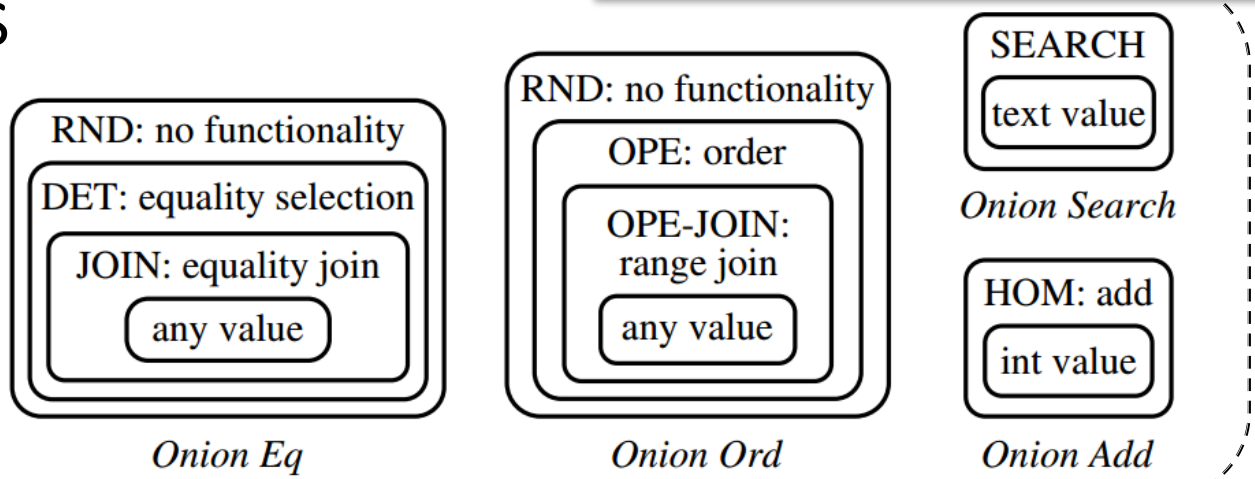
- Encrypted with **CryptDB**
- **Multi-Tenancy** through live migration
- Workload-aware **partitioning** (graph-based)

C. Curino, et al. "Relational cloud: A database-as-a-service for the cloud.“, CIDR 2011

SQL-Proxy

Encrypts and decrypts,

RDBMS



Encrypted Databases: Research

- Example: **CryptDB**
- **Idea:** Only decrypt as much

SQL-Proxy

Encrypts and decrypts,

Relational Cloud

DBaaS Architecture:

- Encrypted with **CryptDB**
- **Multi-Tenancy** through live migration
- Workload-aware **partitioning** (graph-based)

C. Curino, et al. "Relational cloud: A database-as-a-service for the cloud." CIDR 2011

- Early approach
- Not adopted in practice, yet

Dream solution:

Full Homorphic Encryption

Transactions/Consistency: Research

	Consistency	Transactional Unit	Commit Latency	Data Loss?
Dynamo	Eventual	None	1 RT	-
Yahoo PNuts	Timeline per key	Single Key	1 RT	possible
COPS	Causality	Multi-Record	1 RT	possible
MySQL (async)	Serializable	Static Partition	1 RT	possible
Megastore	Serializable	Static Partition	2 RT	-
Spanner/F1	Snapshot Isolation	Partition	2 RT	-
MDCC	Read-Commited	Multi-Record	1 RT	-

Transactions/Consistency: Research

	Consisten		mit	Data
Dynamo	Eventual			
Yahoo PNuts	Timeline pe			
COPS	Causality			
MySQL (async)	Serializable			
Megastore	Serializable			
Spanner/F1	Snapshot Isolation	Partition	2 RT	-
MDCC	Read-Commited	Multi-Record	1 RT	-

Multi-Data Center Consistency

Idea:

- Multi-Data center commit protocol with single round-trip

Implementation:

- Optimistic Commit Protocol
- Fast, Generalized Multi-Paxos

Result: almost as fast as Dynamo-style

T. Kraska et al. "MDCC: Multi-data center consistency." EuroSys, 2013.

Transactions/Consistency: Research

	Consisten	mit	Data
Dynamo	Eventual		
Yahoo PNuts	Timeline pe		
COPS	Causality		
MySQL (async)	Serializable		
Megastore	Serializable		
Spanner	Snapshot Isolation		
MDCC	Read Committed	Multi Record	1 RT

Multi-Data Center Consistency

Idea:

- Multi-Data center commit protocol with single round-trip

Implementation:

- Optimistic Commit Protocol
- Fast, Generalized Multi-Paxos

Result: almost as fast as Dynamo-style

T. Kraska et al. "MDCC: Multi-data center consistency." EuroSys, 2013.

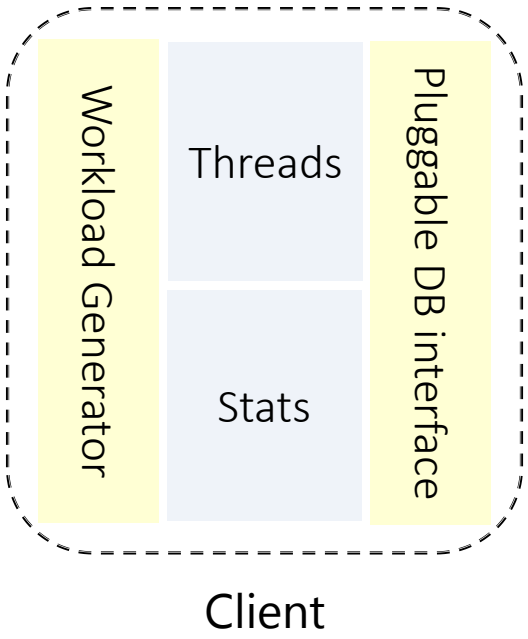
Currently no NoSQL DB implements consistent Multi-DC replication

Benchmarking: Research

- **YCSB** (**Y**ahoo **C**loud **S**erving **B**enchmark)

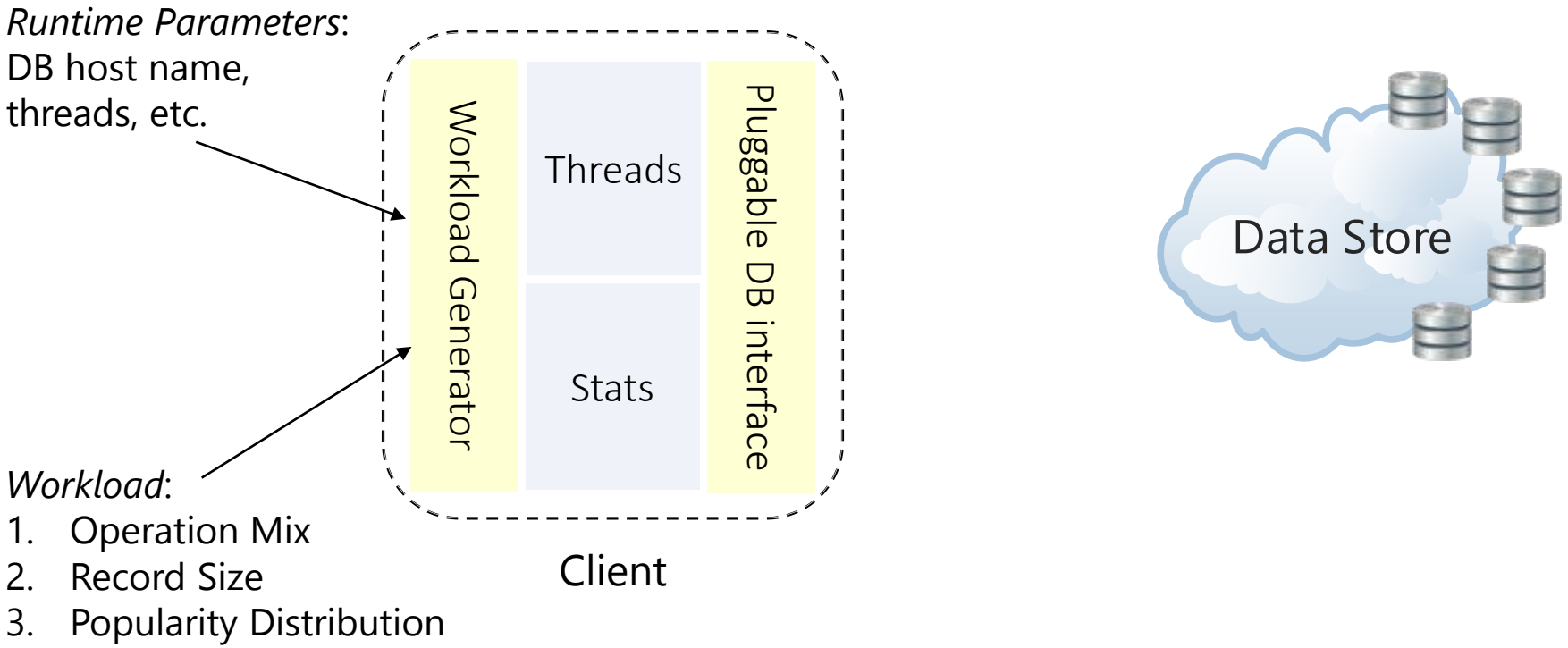
Benchmarking: Research

- **YCSB** (**Y**ahoo **C**loud **S**erving **B**enchmark)



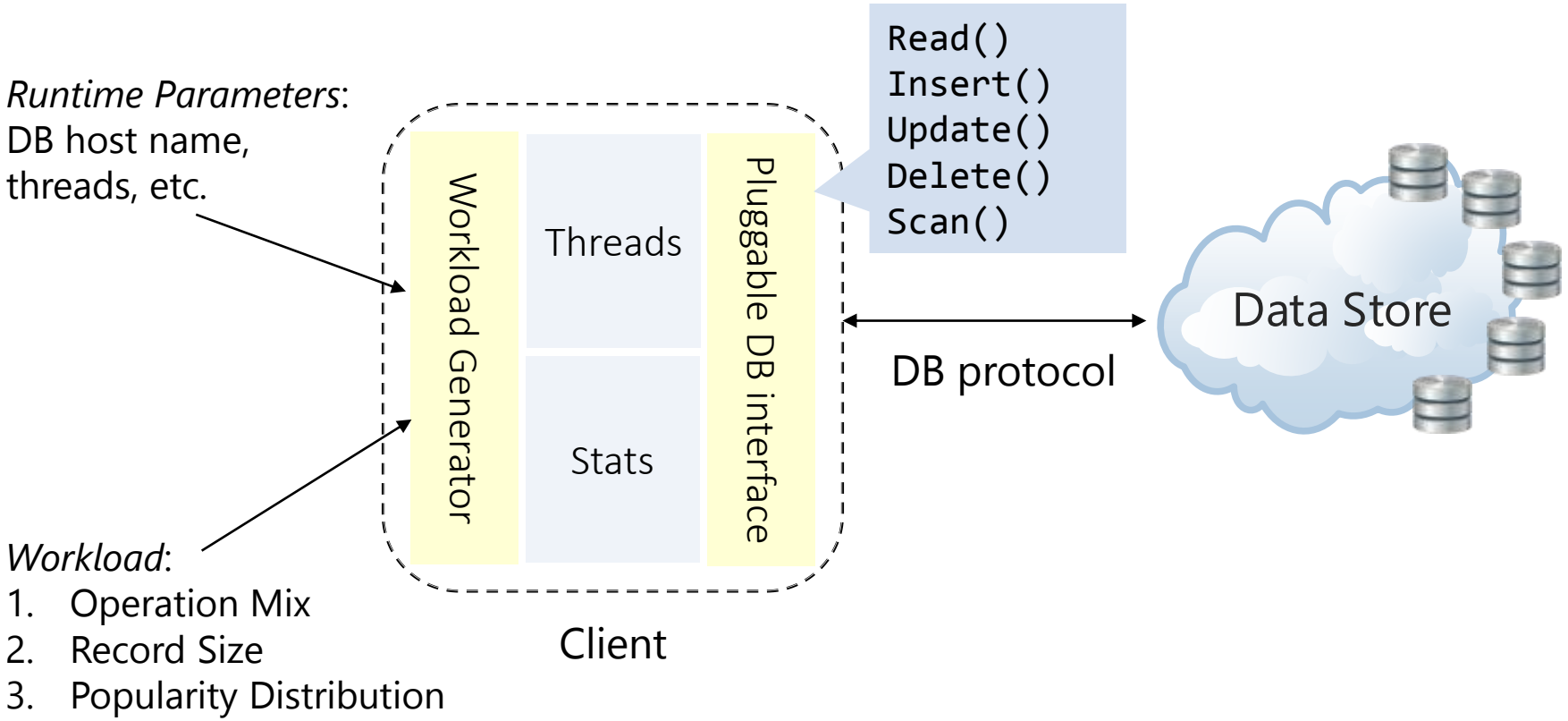
Benchmarking: Research

- **YCSB** (**Y**ahoo **C**loud **S**erving **B**enchmark)



Benchmarking: Research

- **YCSB** (**Y**ahoo **C**loud **S**erving **B**enchmark)

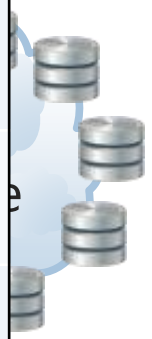


Benchmarking: Research

- YCSB (Yahoo Cloud Serving Benchmark)

`Read()`

Workload	Operation Mix	Distribution	Example
A – Update Heavy	Read: 50% Update: 50%	Zipfian	Session Store
B – Read Heavy	Read: 95% Update: 5%	Zipfian	Photo Tagging
C – Read Only	Read: 100%	Zipfian	User Profile Cache
D – Read Latest	Read: 95% Insert: 5%	Latest	User Status Updates
E – Short Ranges	Scan: 95% Insert: 5%	Zipfian/ Uniform	Threaded Conversations

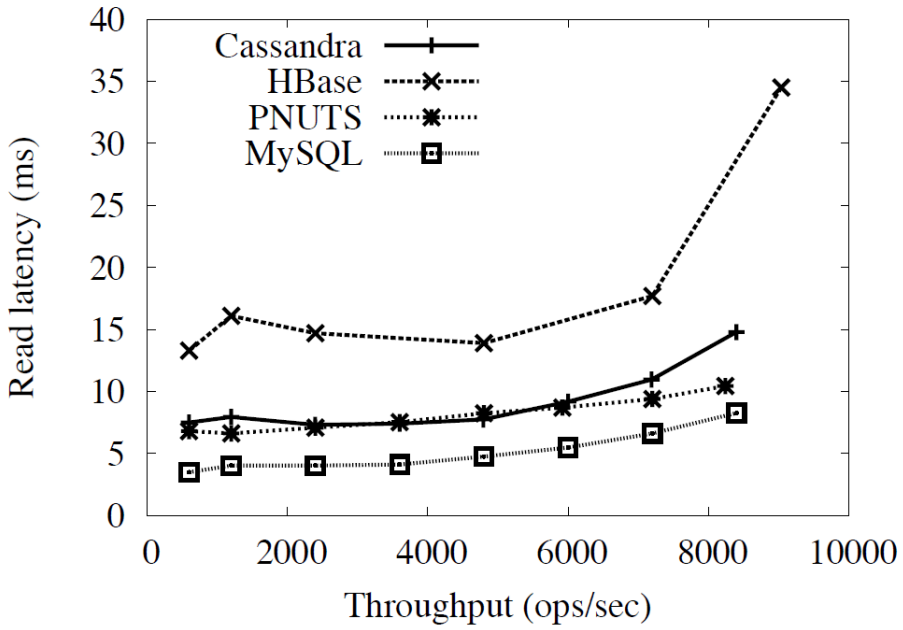


3. Popularity Distribution

Benchmarking: Research

- **Example Result**

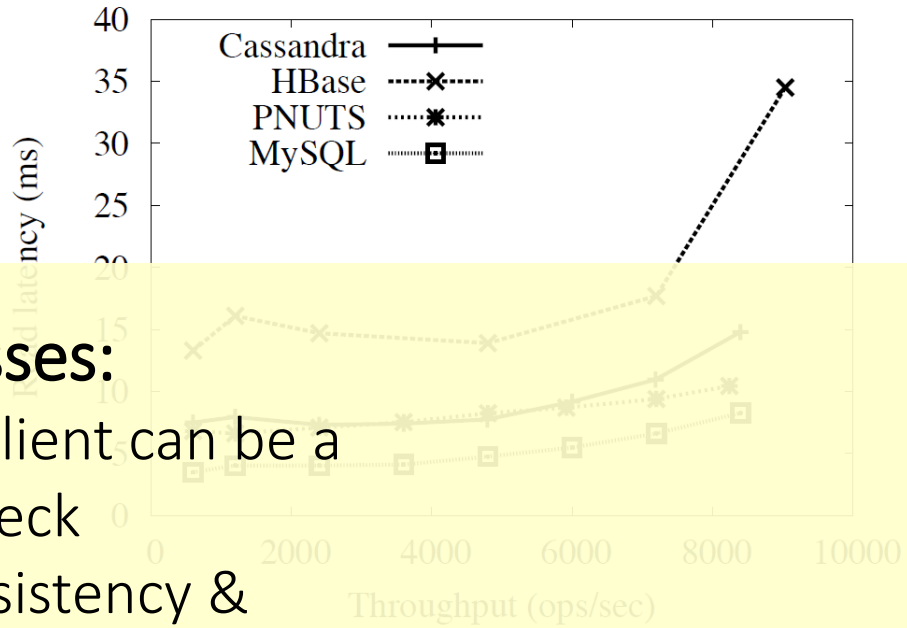
  (Read Heavy):



Benchmarking: Research

- **Example Result**

  (Read Heavy):



Weaknesses:

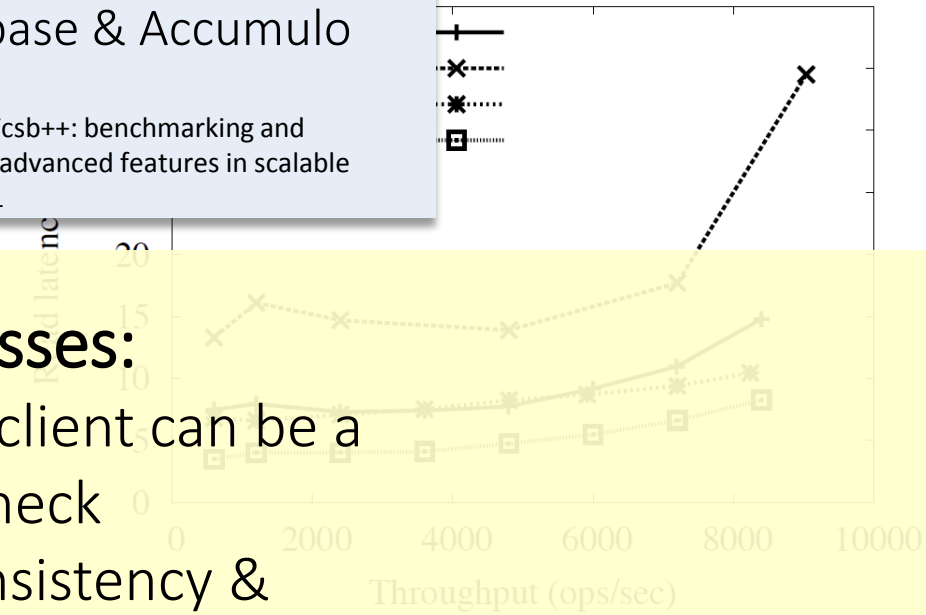
- Single client can be a bottleneck
- No consistency & availability measurement

Benchmarking: Research

YCSB++

- Clients coordinate through Zookeeper
- Simple Read-After-Write Checks
- Evaluation: Hbase & Accumulo

S. Patil, M. Polte, et al.„Ycsb++: benchmarking and performance debugging advanced features in scalable table stores“, SOCC 2011



Weaknesses:

- Single client can be a bottleneck
- No consistency & availability measurement

Benchmarking: Research

YCSB++

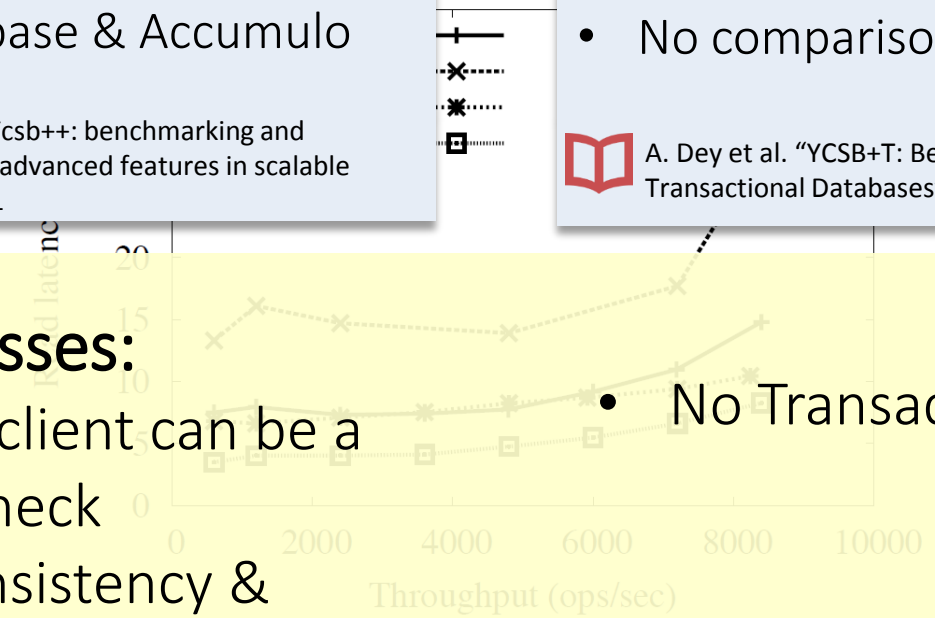
- Clients coordinate through Zookeeper
- Simple Read-After-Write Checks
- Evaluation: Hbase & Accumulo

S. Patil, M. Polte, et al.„Ycsb++: benchmarking and performance debugging advanced features in scalable table stores“, SOCC 2011

YCSB+T

- **New workload:** Transactional Bank Account
- Simple anomaly detection for Lost Updates
- No comparison of systems

A. Dey et al. “YCSB+T: Benchmarking Web-Scale Transactional Databases”, CloudDB 2014



Weaknesses:

- Single client can be a bottleneck
- No consistency & availability measurement
- No Transaction Support

Benchmarking: Research

YCSB++

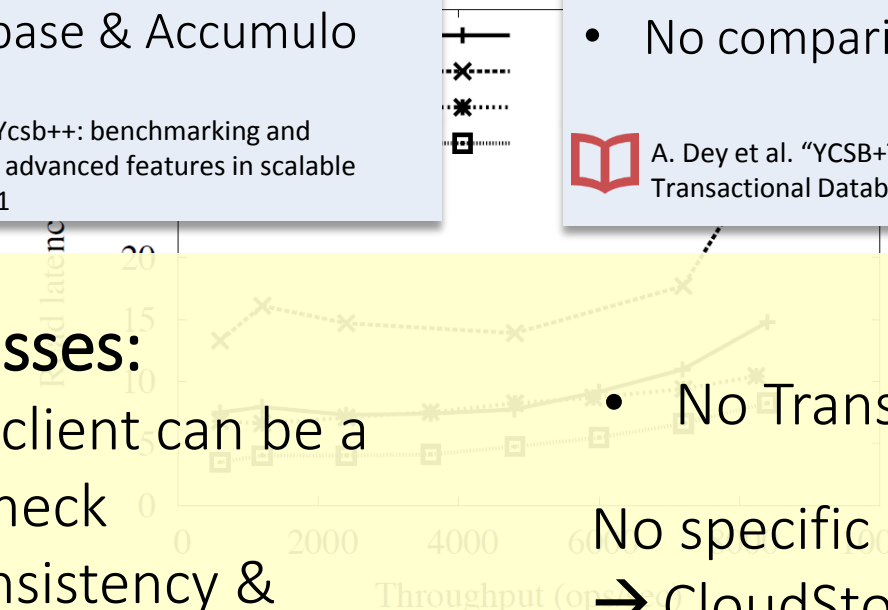
- Clients coordinate through Zookeeper
- Simple Read-After-Write Checks
- Evaluation: Hbase & Accumulo

S. Patil, M. Polte, et al.„Ycsb++: benchmarking and performance debugging advanced features in scalable table stores“, SOCC 2011

YCSB+T

- **New workload:** Transactional Bank Account
- Simple anomaly detection for Lost Updates
- No comparison of systems

A. Dey et al. “YCSB+T: Benchmarking Web-Scale Transactional Databases”, CloudDB 2014

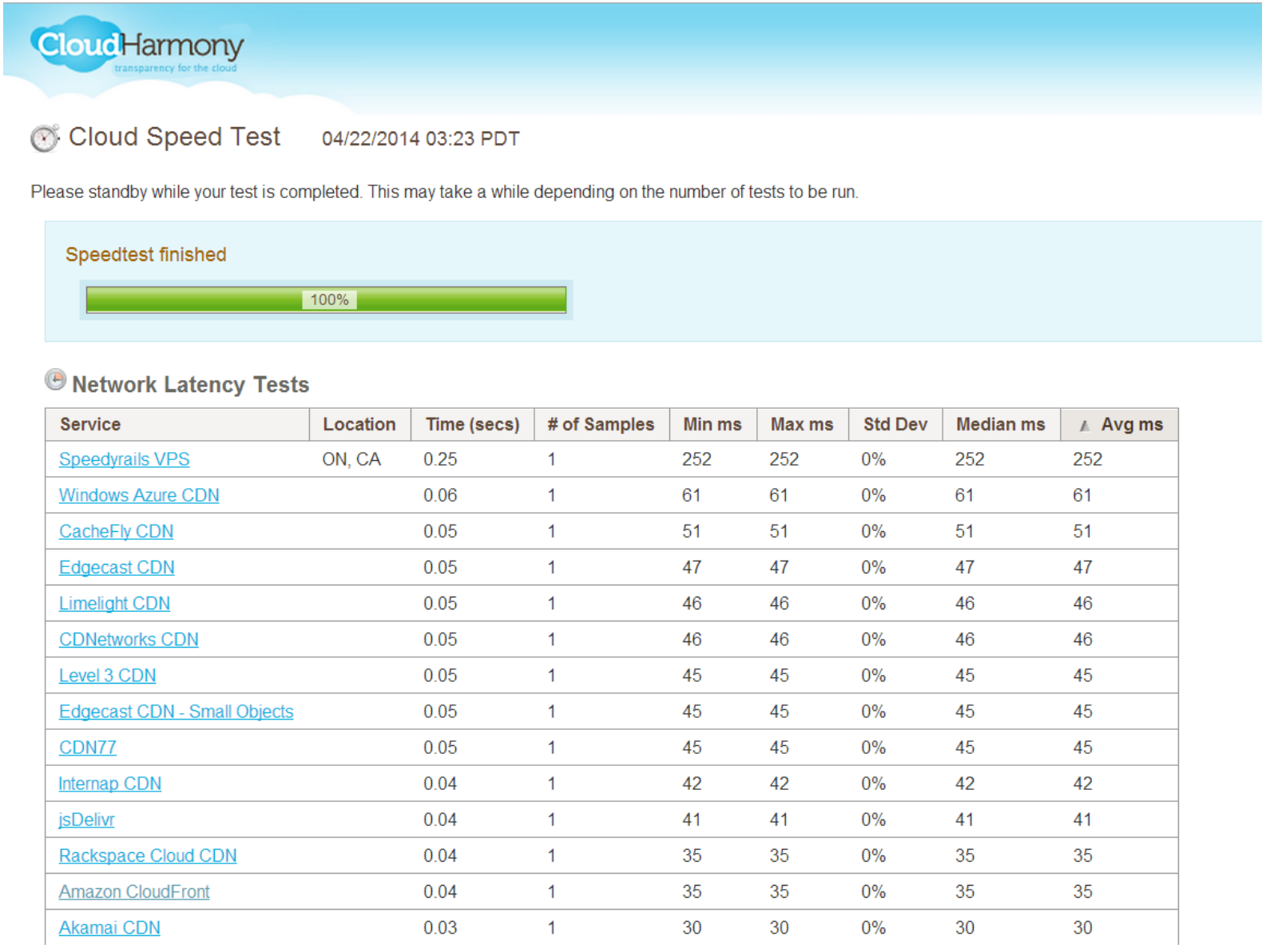


Weaknesses:

- Single client can be a bottleneck
- No consistency & availability measurement
- No Transaction Support

No specific application
→ CloudStone, CARE, TPC extensions?

Benchmarking: state of the art



CloudHarmony
transparency for the cloud

Cloud Speed Test 04/22/2014 03:23 PDT

Please standby while your test is completed. This may take a while depending on the number of tests to be run.

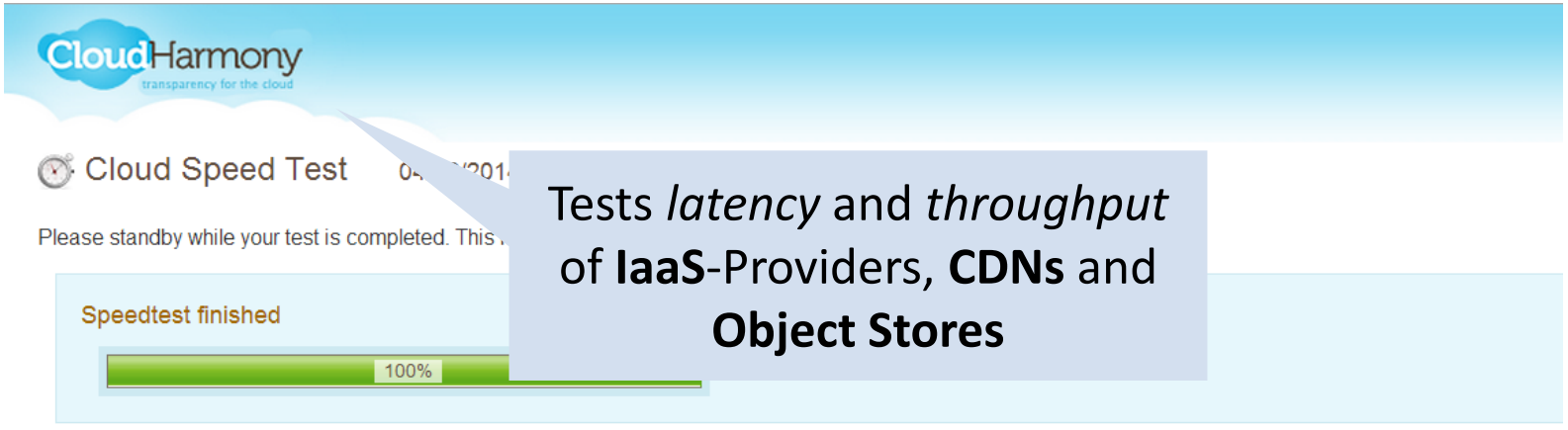
Speedtest finished

100%

Network Latency Tests

Service	Location	Time (secs)	# of Samples	Min ms	Max ms	Std Dev	Median ms	Avg ms
Speedyrails VPS	ON, CA	0.25	1	252	252	0%	252	252
Windows Azure CDN		0.06	1	61	61	0%	61	61
CacheFly CDN		0.05	1	51	51	0%	51	51
Edgecast CDN		0.05	1	47	47	0%	47	47
Limelight CDN		0.05	1	46	46	0%	46	46
CDNetworks CDN		0.05	1	46	46	0%	46	46
Level 3 CDN		0.05	1	45	45	0%	45	45
Edgecast CDN - Small Objects		0.05	1	45	45	0%	45	45
CDN77		0.05	1	45	45	0%	45	45
Internap CDN		0.04	1	42	42	0%	42	42
jsDelivr		0.04	1	41	41	0%	41	41
Rackspace Cloud CDN		0.04	1	35	35	0%	35	35
Amazon CloudFront		0.04	1	35	35	0%	35	35
Akamai CDN		0.03	1	30	30	0%	30	30

Benchmarking: state of the art

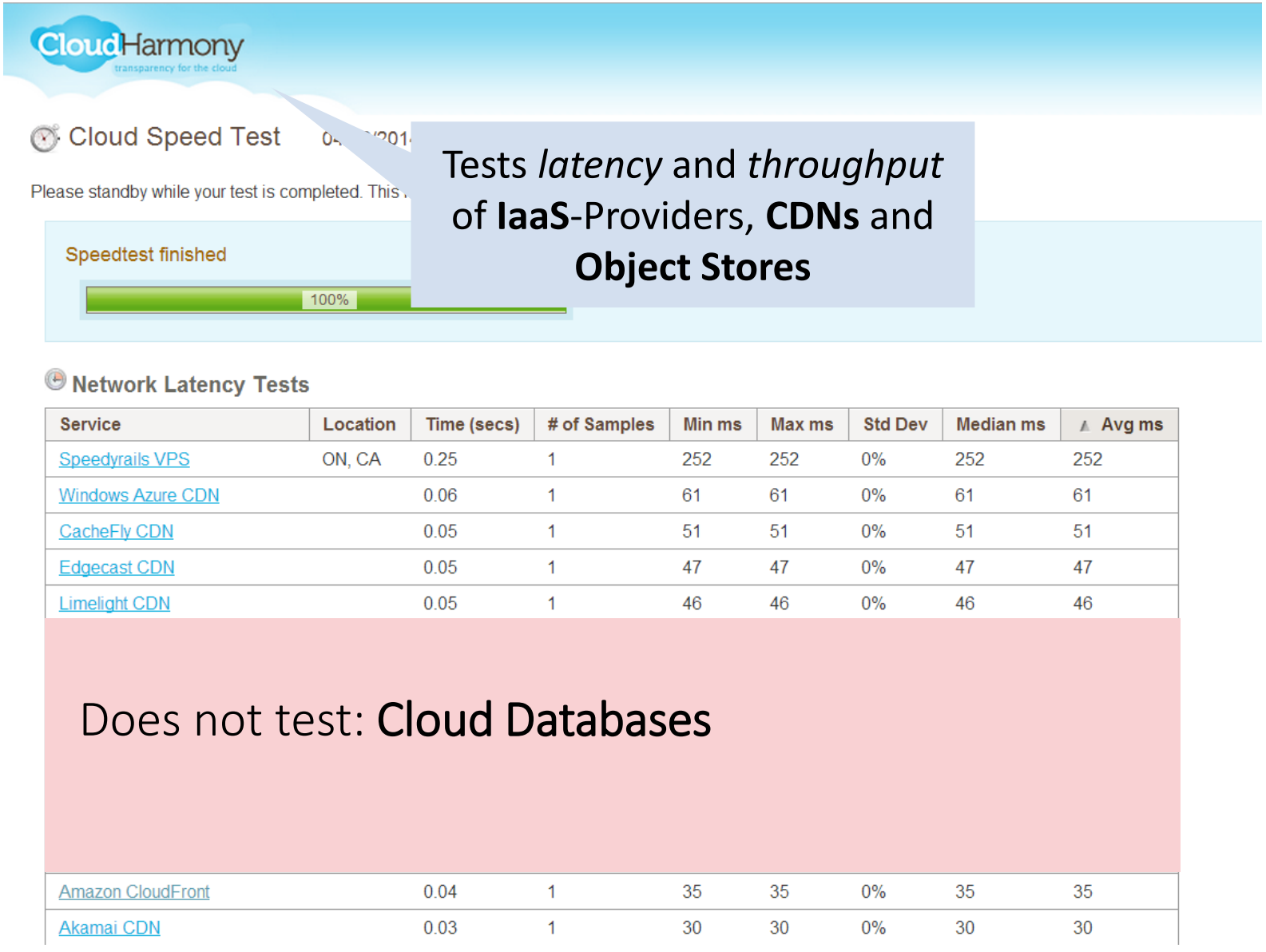


Tests *latency* and *throughput* of **IaaS**-Providers, **CDNs** and **Object Stores**

Network Latency Tests

Service	Location	Time (secs)	# of Samples	Min ms	Max ms	Std Dev	Median ms	Avg ms
Speedyrails VPS	ON, CA	0.25	1	252	252	0%	252	252
Windows Azure CDN		0.06	1	61	61	0%	61	61
CacheFly CDN		0.05	1	51	51	0%	51	51
Edgecast CDN		0.05	1	47	47	0%	47	47
Limelight CDN		0.05	1	46	46	0%	46	46
CDNetworks CDN		0.05	1	46	46	0%	46	46
Level 3 CDN		0.05	1	45	45	0%	45	45
Edgecast CDN - Small Objects		0.05	1	45	45	0%	45	45
CDN77		0.05	1	45	45	0%	45	45
Internap CDN		0.04	1	42	42	0%	42	42
jsDelivr		0.04	1	41	41	0%	41	41
Rackspace Cloud CDN		0.04	1	35	35	0%	35	35
Amazon CloudFront		0.04	1	35	35	0%	35	35
Akamai CDN		0.03	1	30	30	0%	30	30

Benchmarking: state of the art



Service	Location	Time (secs)	# of Samples	Min ms	Max ms	Std Dev	Median ms	Avg ms
Speedyrails VPS	ON, CA	0.25	1	252	252	0%	252	252
Windows Azure CDN		0.06	1	61	61	0%	61	61
CacheFly CDN		0.05	1	51	51	0%	51	51
Edgecast CDN		0.05	1	47	47	0%	47	47
Limelight CDN		0.05	1	46	46	0%	46	46
Amazon CloudFront		0.04	1	35	35	0%	35	35
Akamai CDN		0.03	1	30	30	0%	30	30

Does not test: **Cloud Databases**

Vision

Harmony

Idea:

Vision

YCSB Harmony

Idea:

1. Periodically launch clients

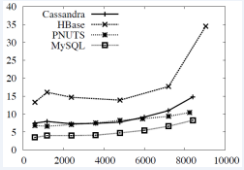
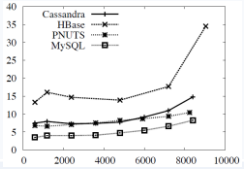
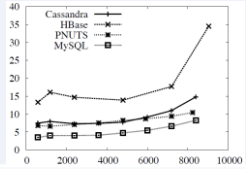
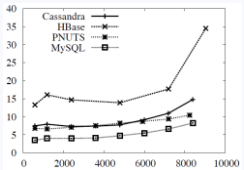
Vision

Idea:

1. Periodically launch clients
2. Benchmark Cloud DB
3. Publish results

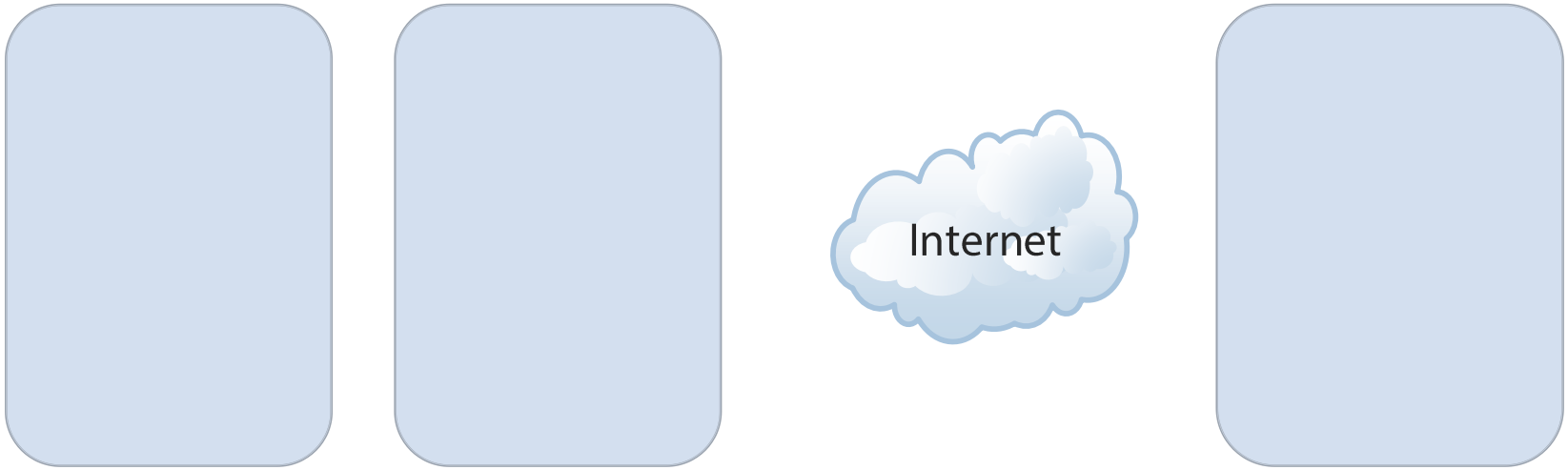
Vision

Harmony

System	Availablity	Reads/s	Writes/s	Avg. Latency	95th perc. Latency	Plot
DynamoDB	99.9%	23411	34534	3.2 ms	9 ms	
RDS	99.8%	2342	2455	30 ms	80 ms	
Azure Table	99.5%	22343	23442	12 ms	20 ms	
Google DataStore	99.5%	3000	2000	30 ms	300 ms	

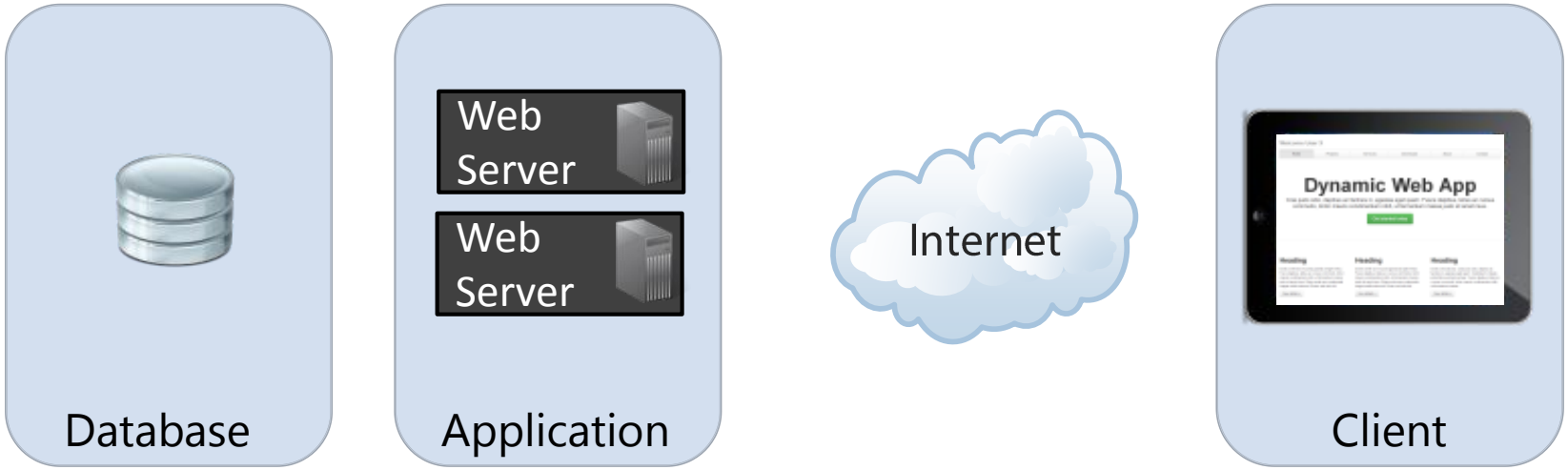
Database research at the University of Hamburg

Motivation



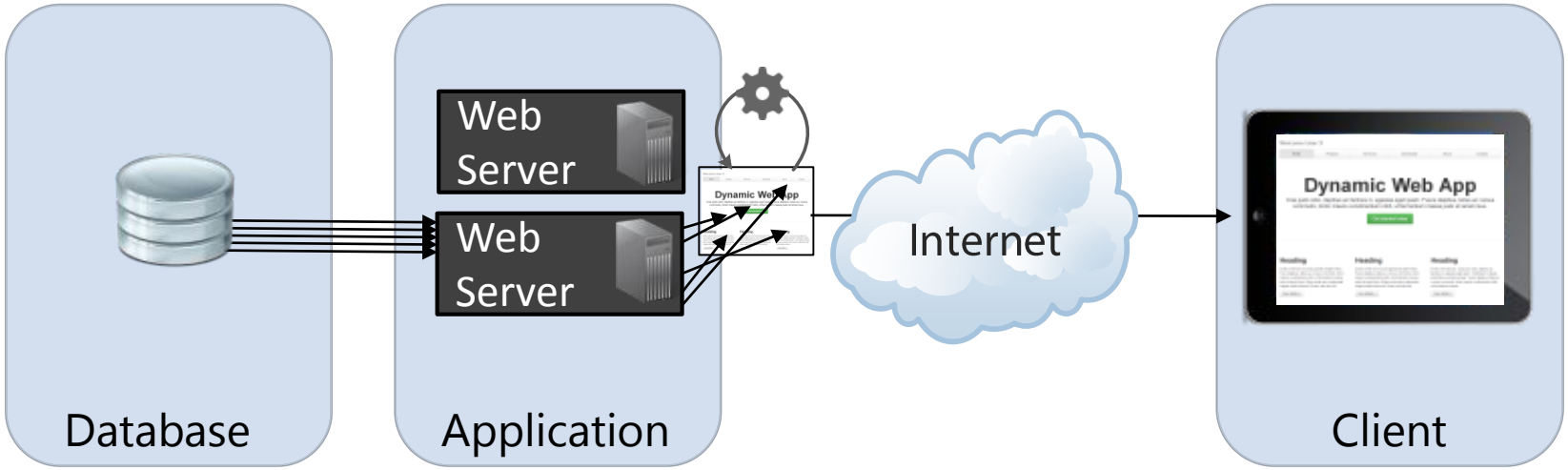
Three-Tier Architecture

Motivation



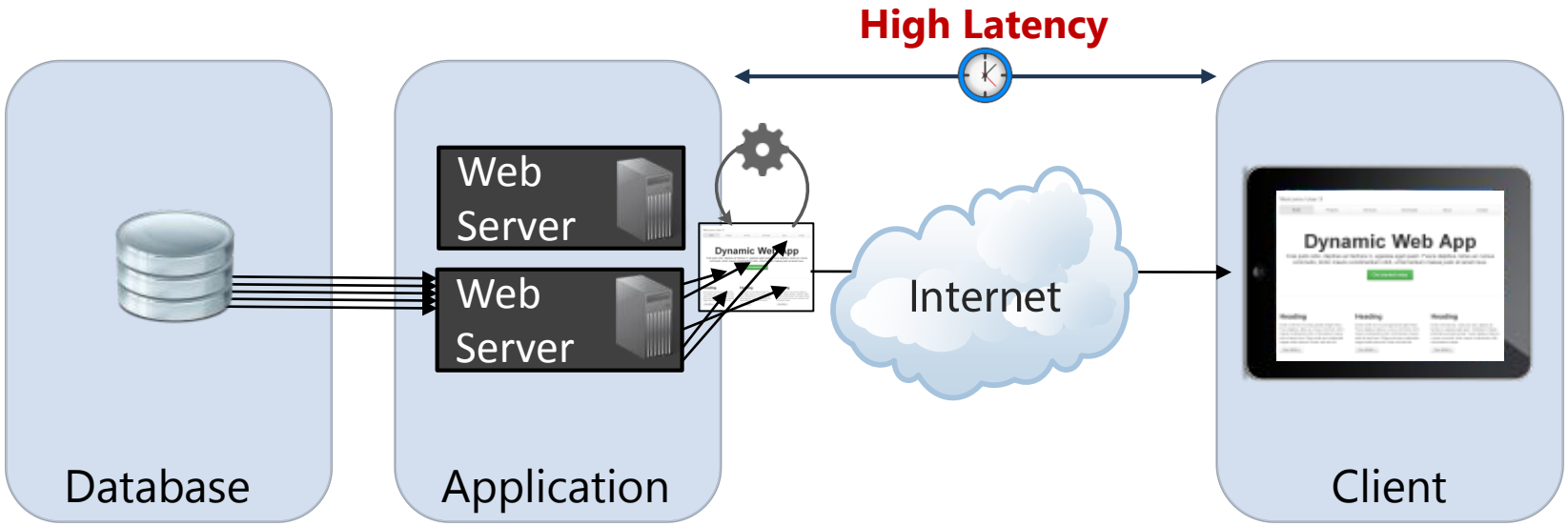
Three-Tier Architecture

Motivation



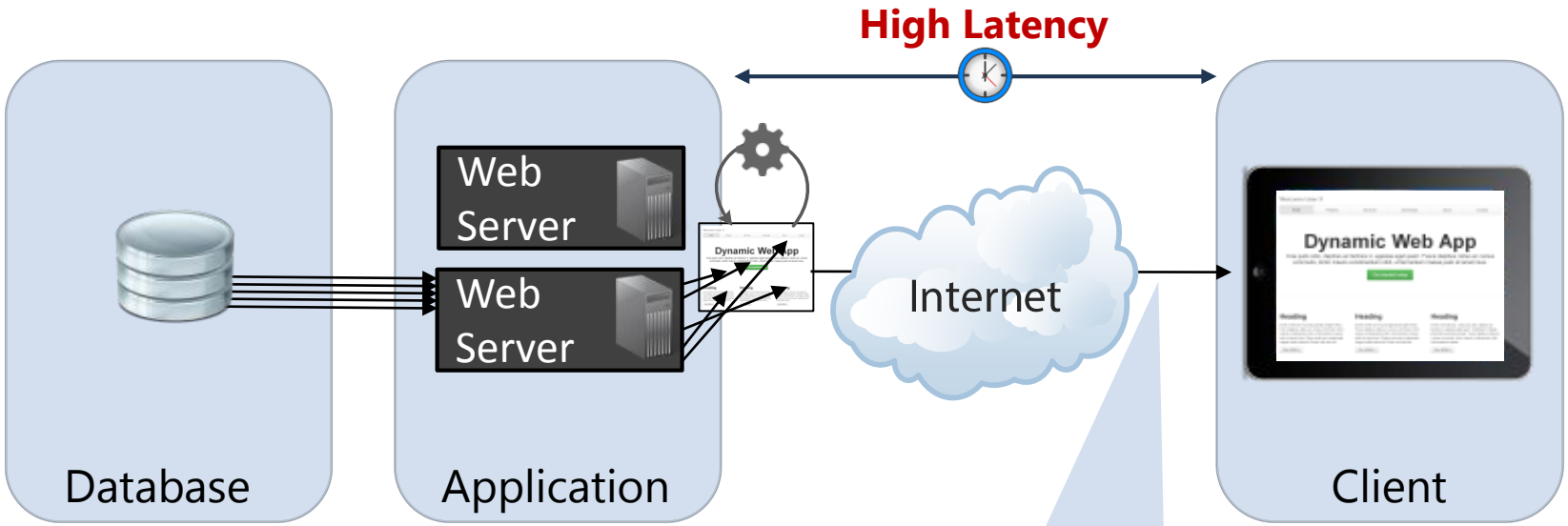
Three-Tier Architecture

Motivation



Three-Tier Architecture

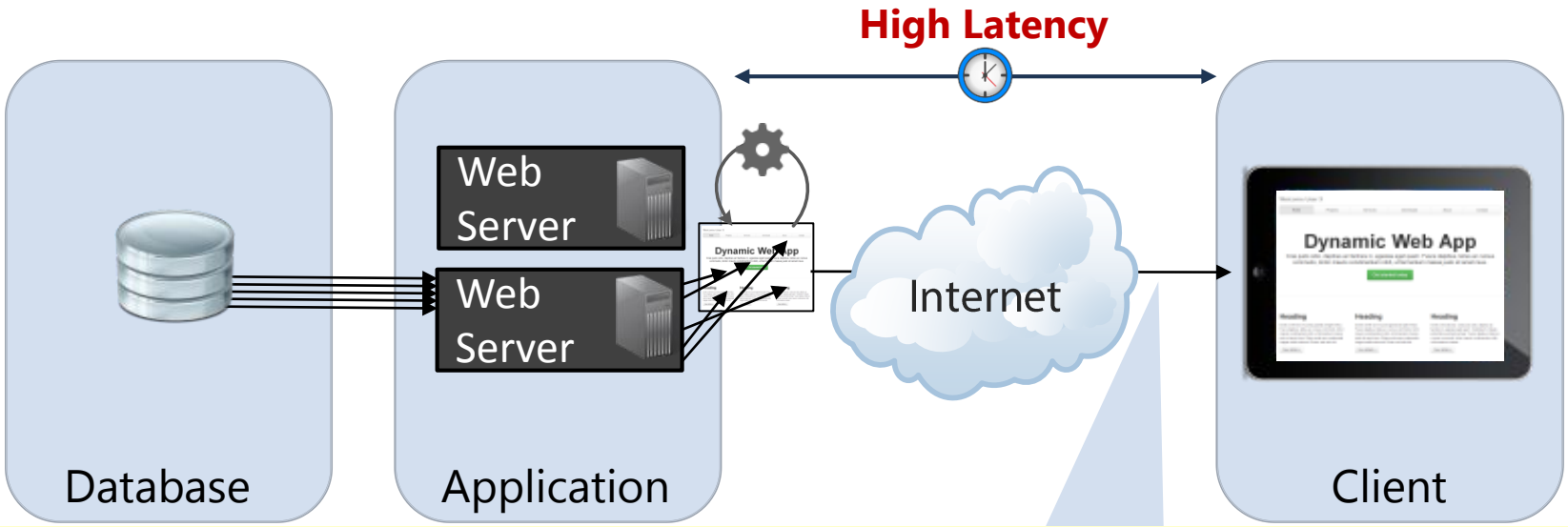
Motivation



Three-Tier Architecture

Average (2014):
90 HTTP Requests per page load

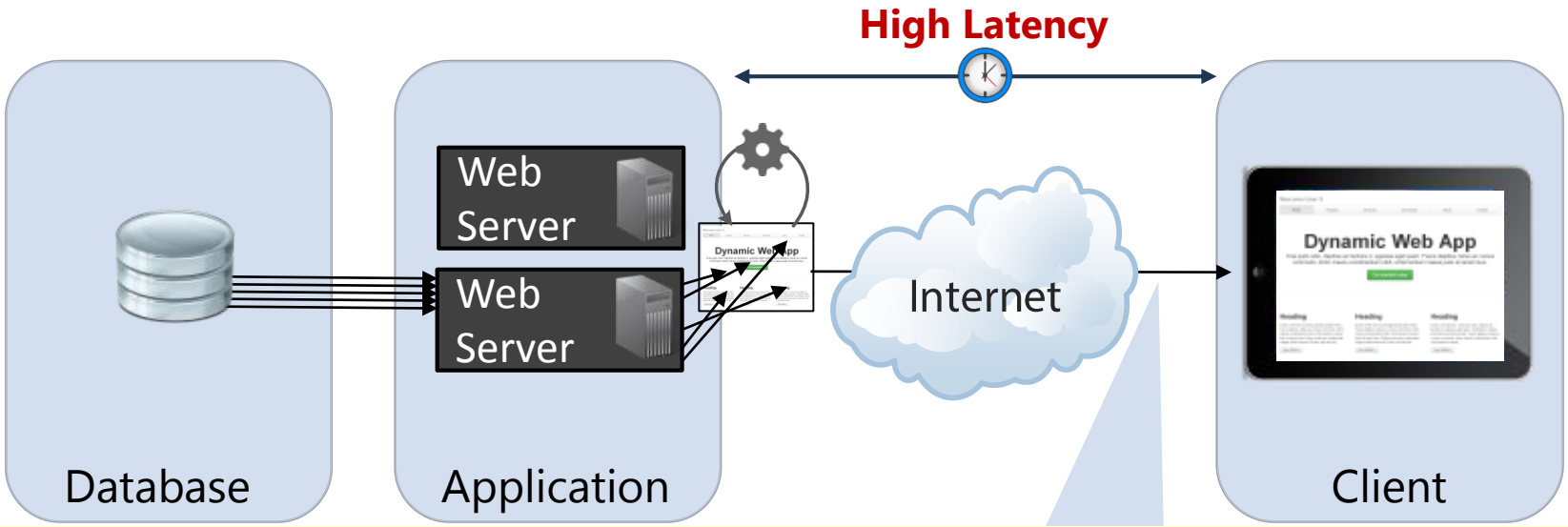
Motivation



With every **100ms** of additional page load time, revenue decreases by 1%.

Study by **Amazon**

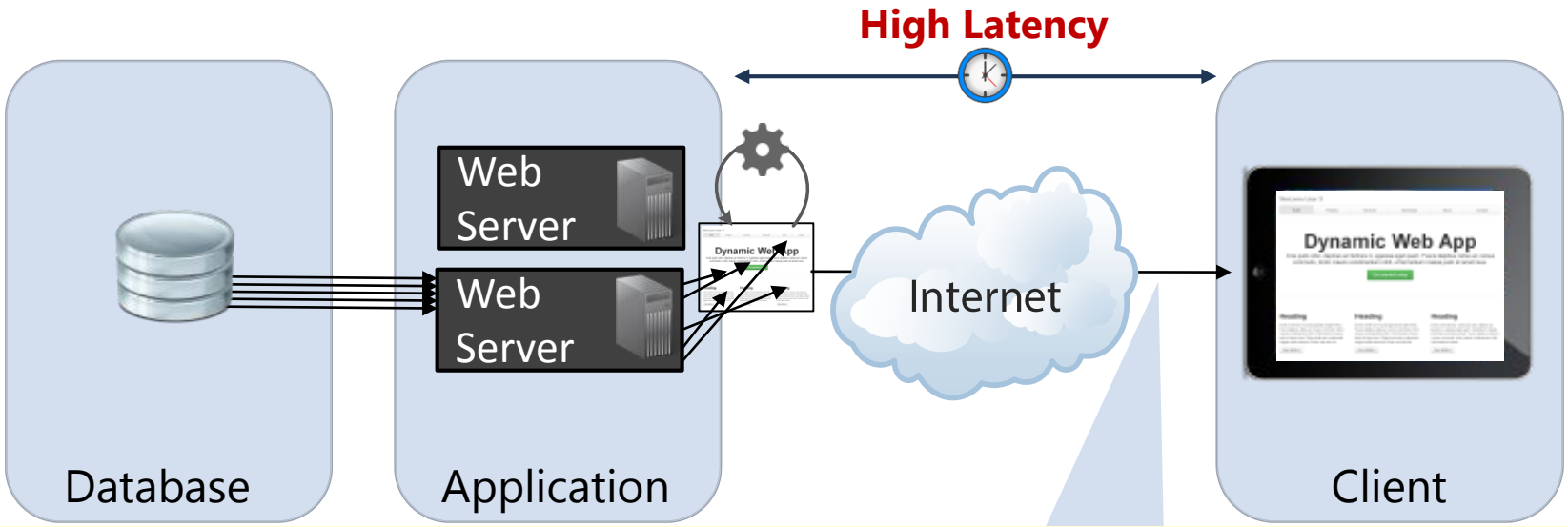
Motivation



When increasing load time of search results by **500ms**, traffic decreases by 20%.

Study by **Google**

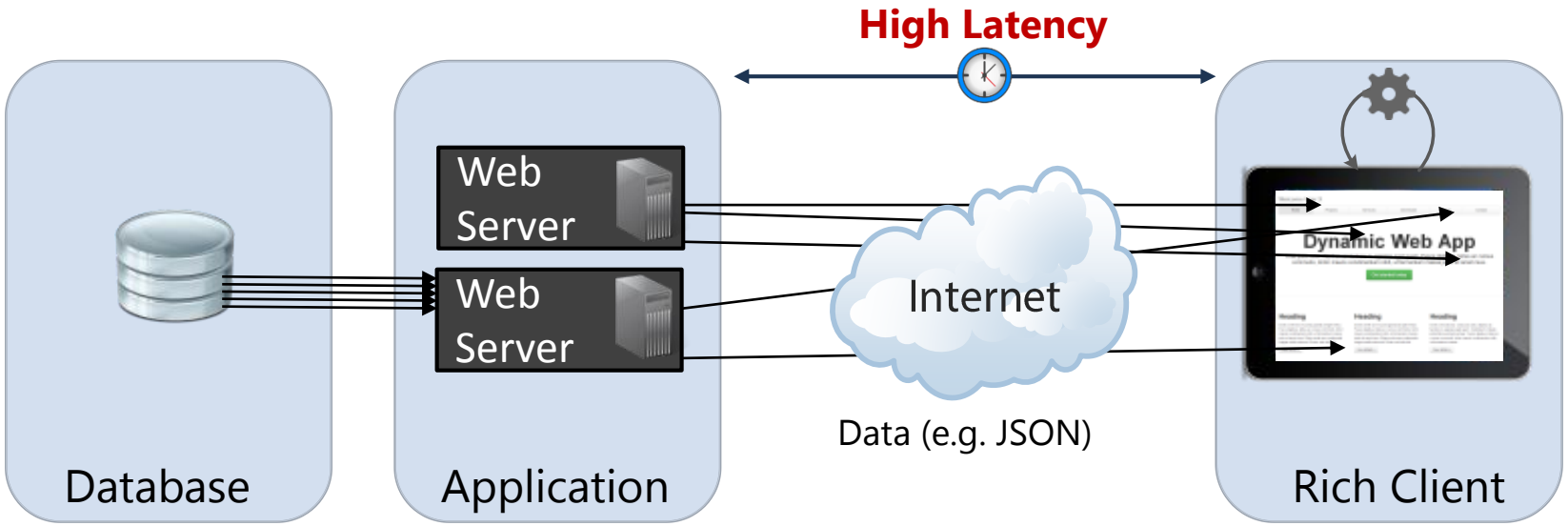
Motivation



When increasing load time of search results by **500ms**, traffic decreases by 20%.

Study by **Google**

SPAs



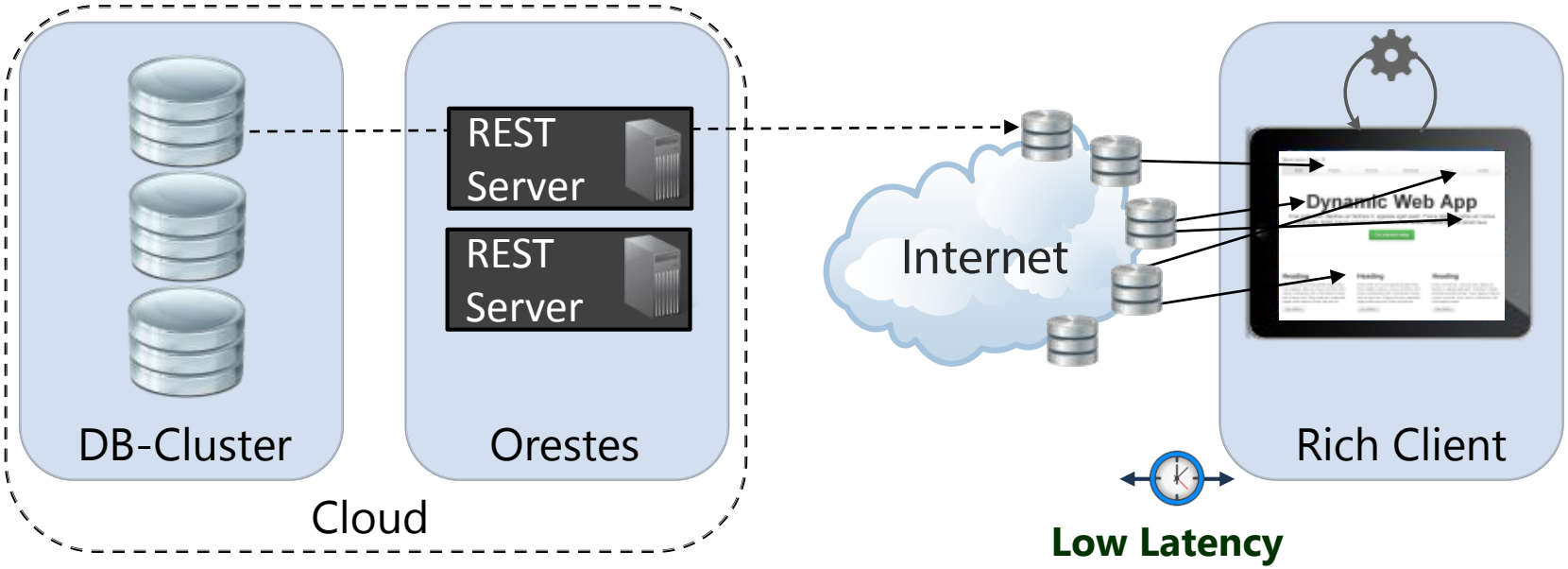
Data (e.g. JSON)

(AngularJS, Backbone, Ember.JS, etc.)

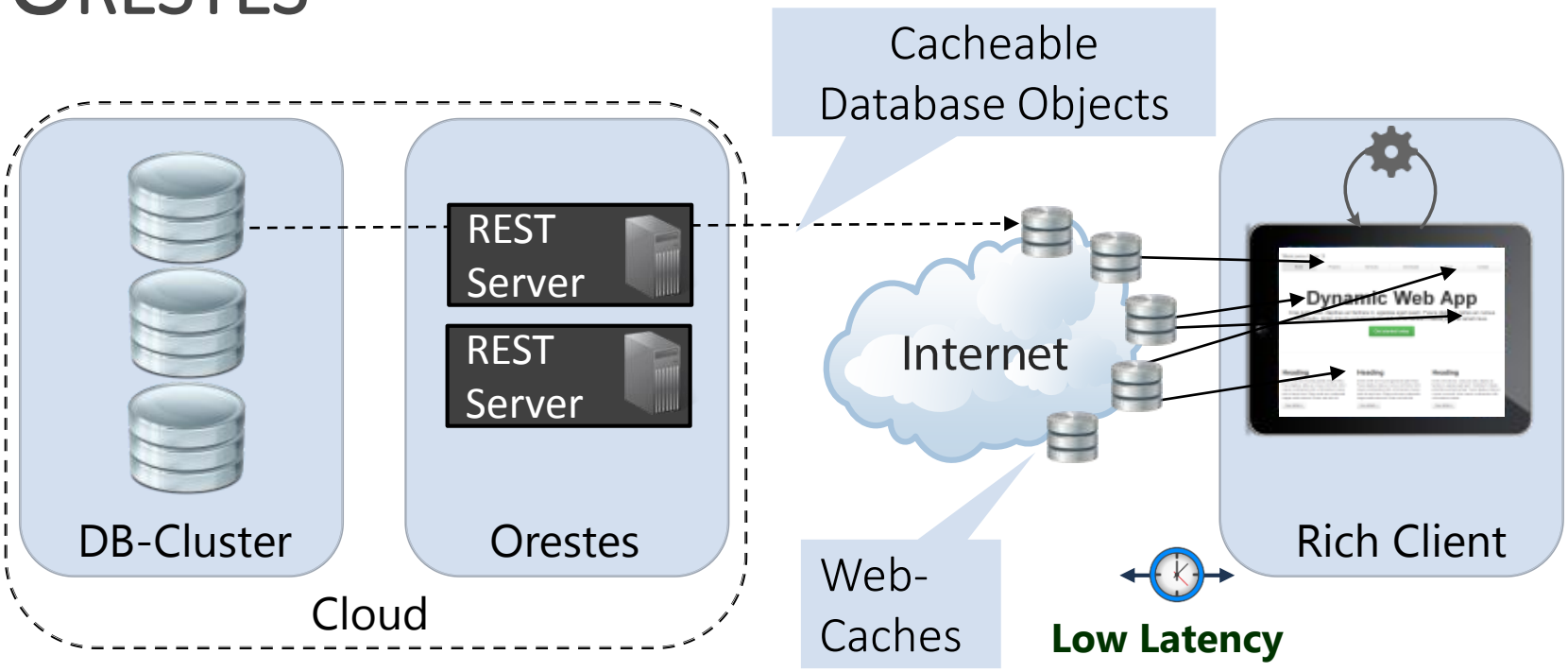
Single-Page Applications

(Rich Client, Smart Client)

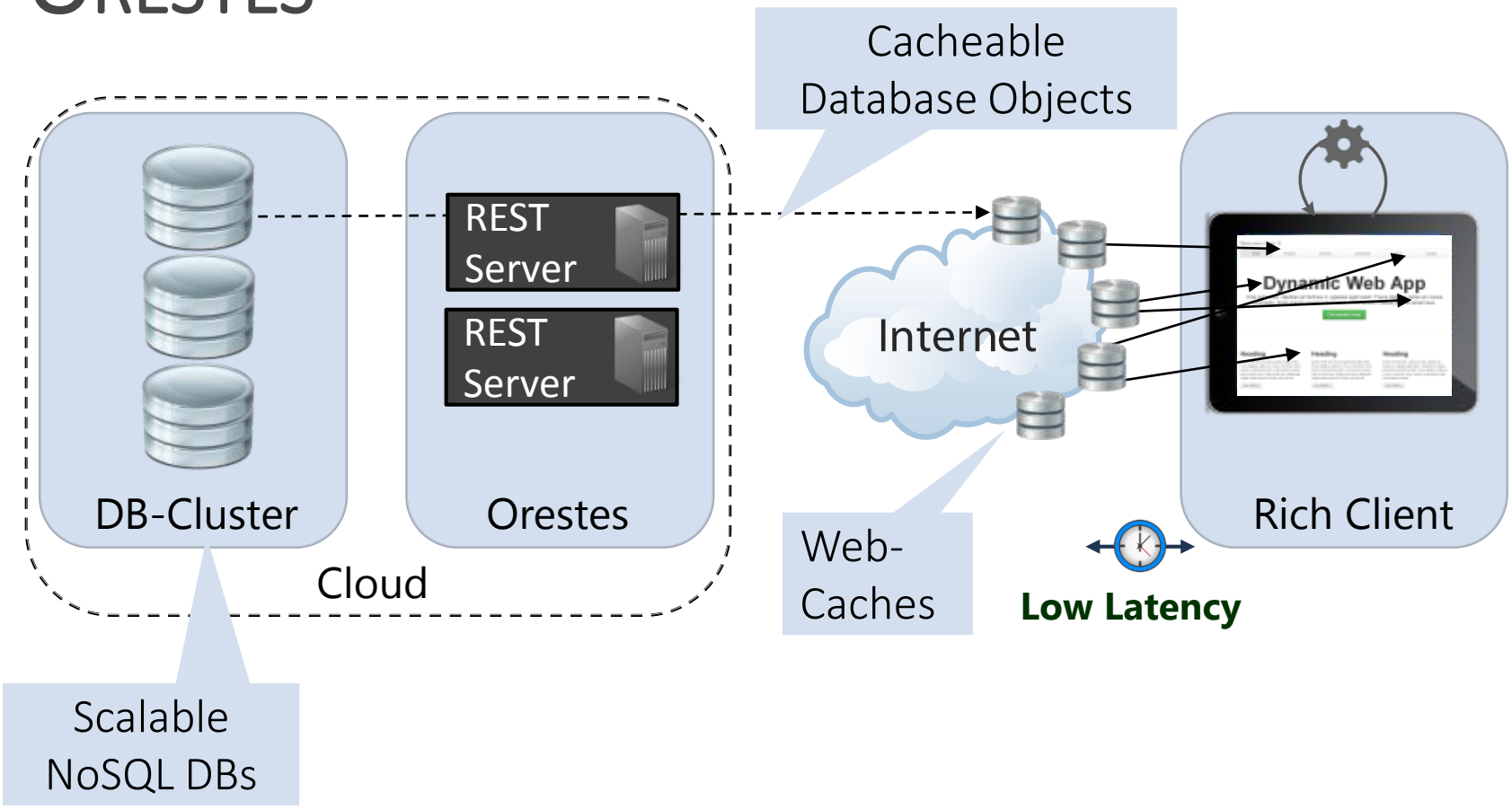
ORESTES


REST Server
REST Server
DB-Cluster
Orestes
Cloud
Internet
Dynamic Web App
Rich Client
Low Latency

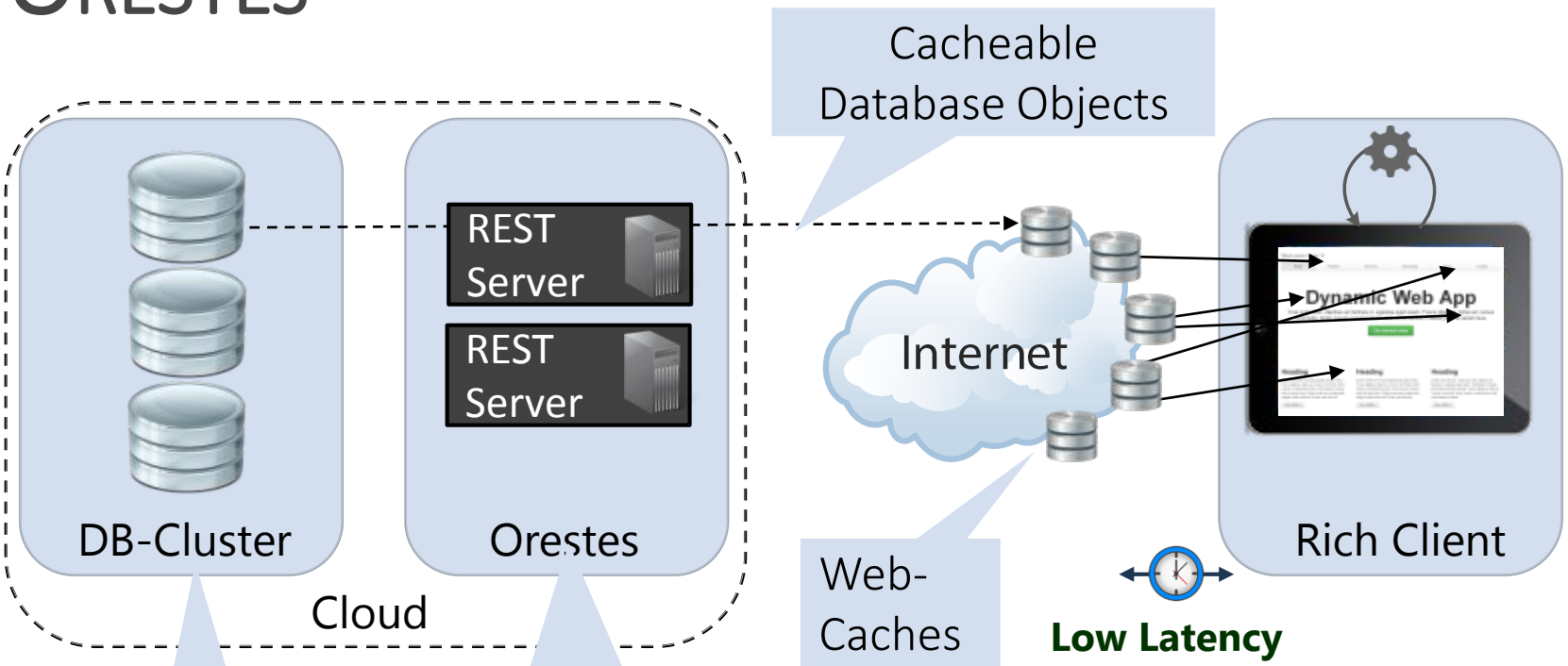
ORESTES


Cacheable
Database Objects
REST
Server
REST
Server
DB-Cluster
Orestes
Cloud
Internet
Dynamic Web App
Rich Client
Web-
Caches
Low Latency

ORESTES


Cacheable
Database Objects
REST
Server
REST
Server
DB-Cluster
Orestes
Cloud
Internet
Dynamic Web App
Rich Client
Web-
Caches
Low Latency
Scalable
NoSQL DBs

ORESTES



Scalable
NoSQL DBs

Exposes the DB via
REST API and handles:

- Scaling
- Cache Consistency
- Transactions
- Schema

ORESTES: Components

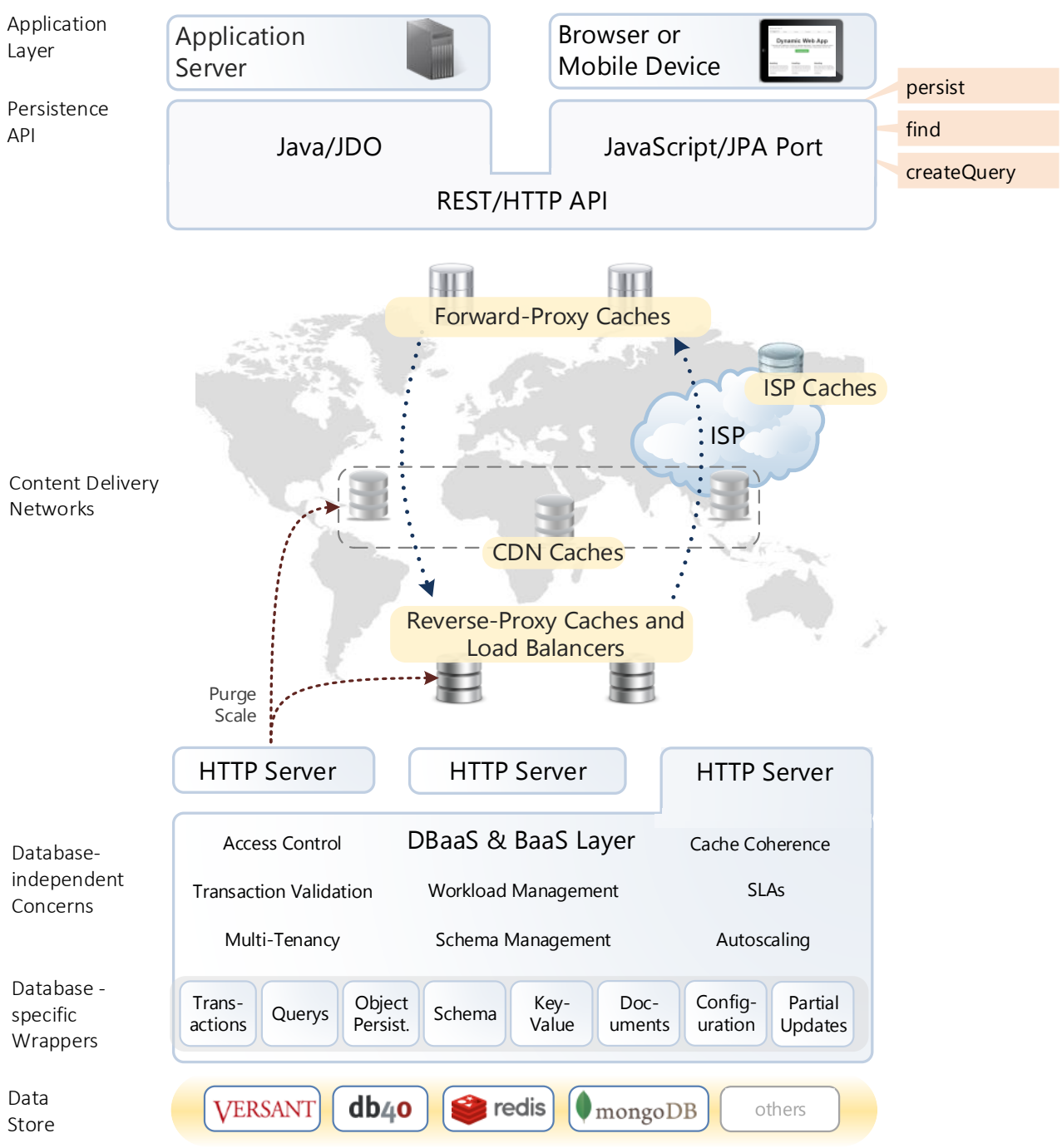
- **Middleware:**
  - Scalable REST API for aggregate-oriented persistence, queries, schema management and transactions.

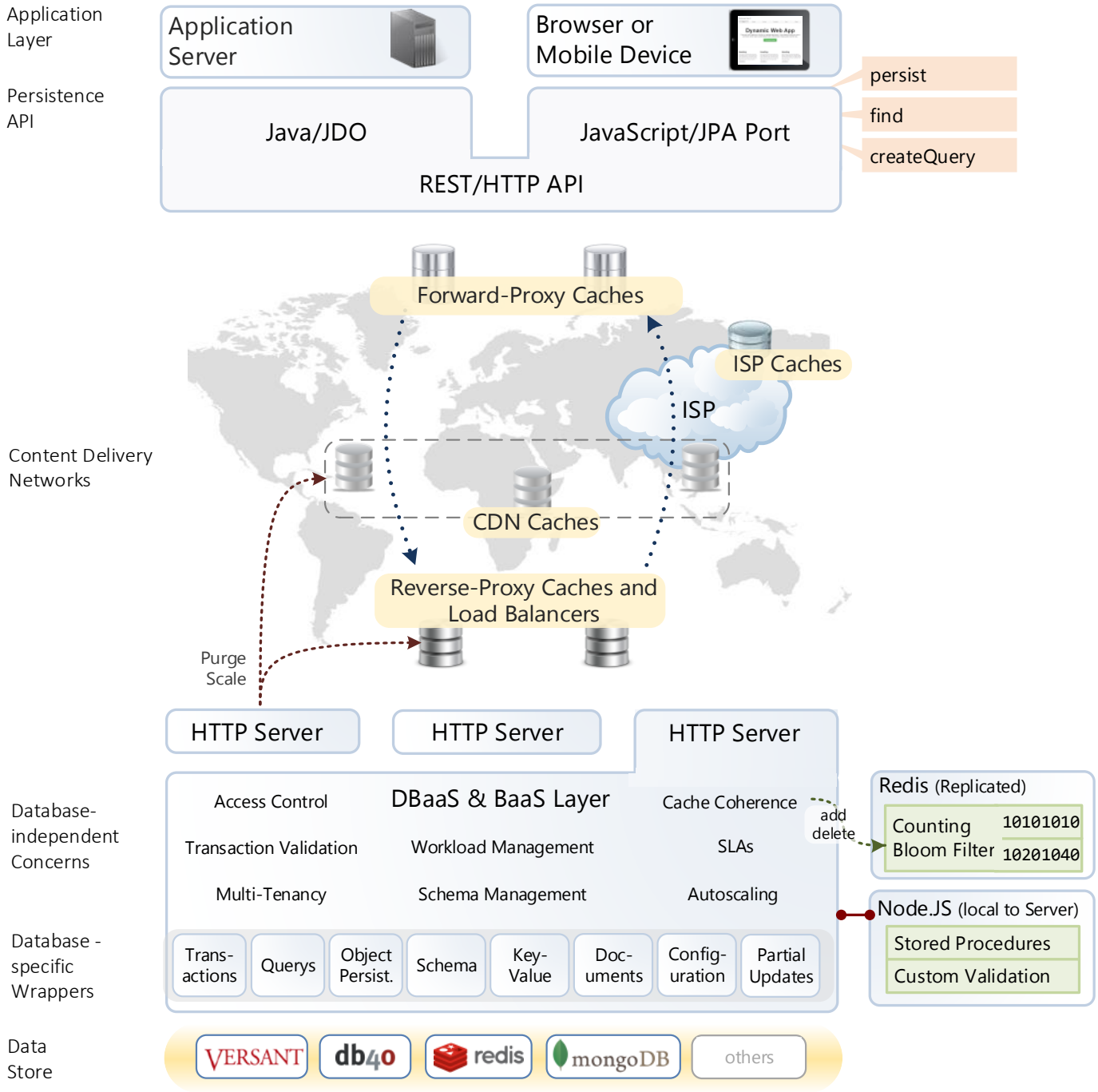
- **Caching:**
  - Consistent web caching of database objects for read scalability and latency reduction.

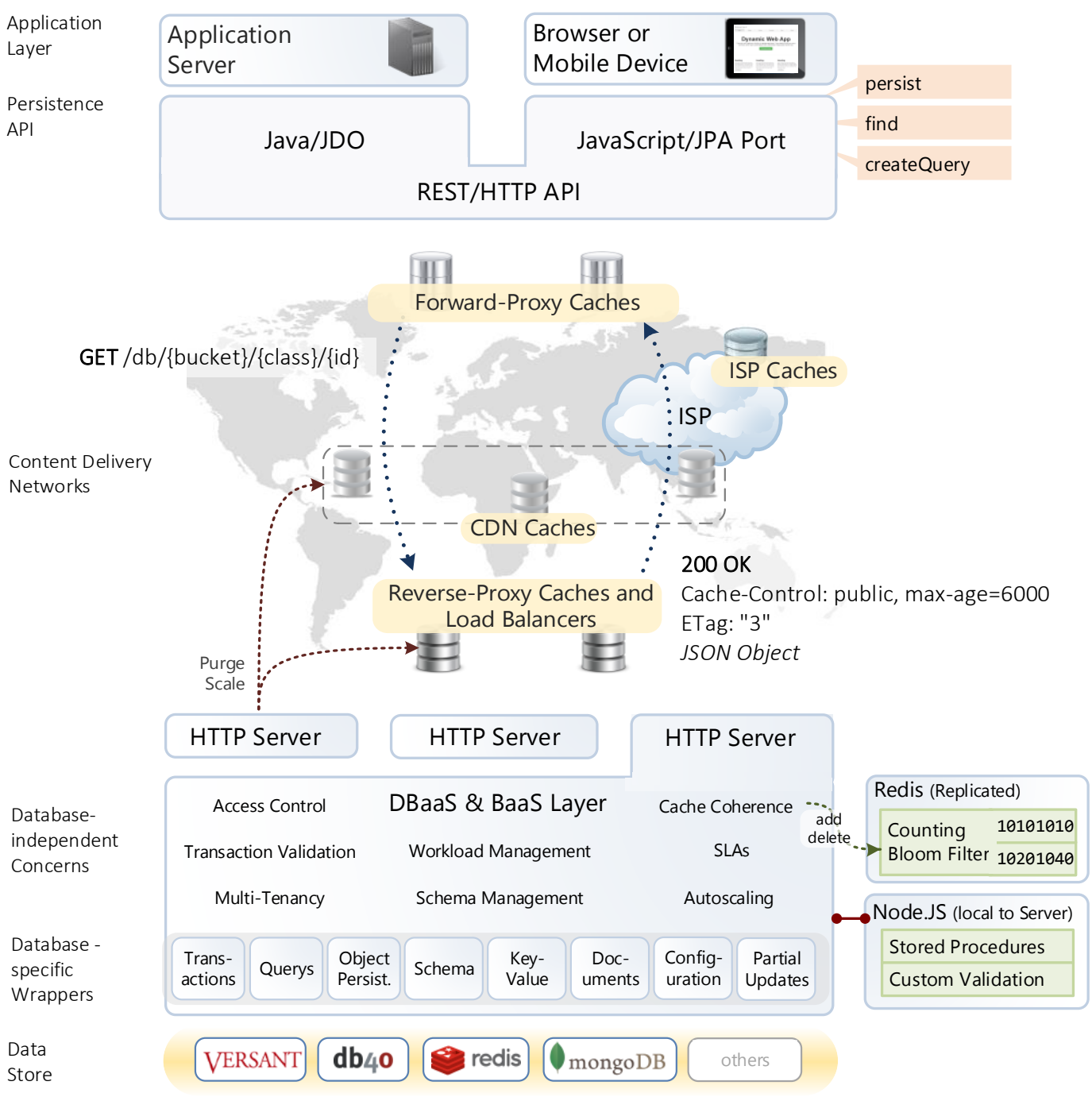
- **Transactions:**
  - Optimistic cache-aware transaction model.


Application Layer
Persistence API
Application Server
Browser or Mobile Device
persist
find
createQuery
Java/JDO
JavaScript/JPA Port
REST/HTTP API
Data Store
VERSANT
db4o
redis
mongoDB
others

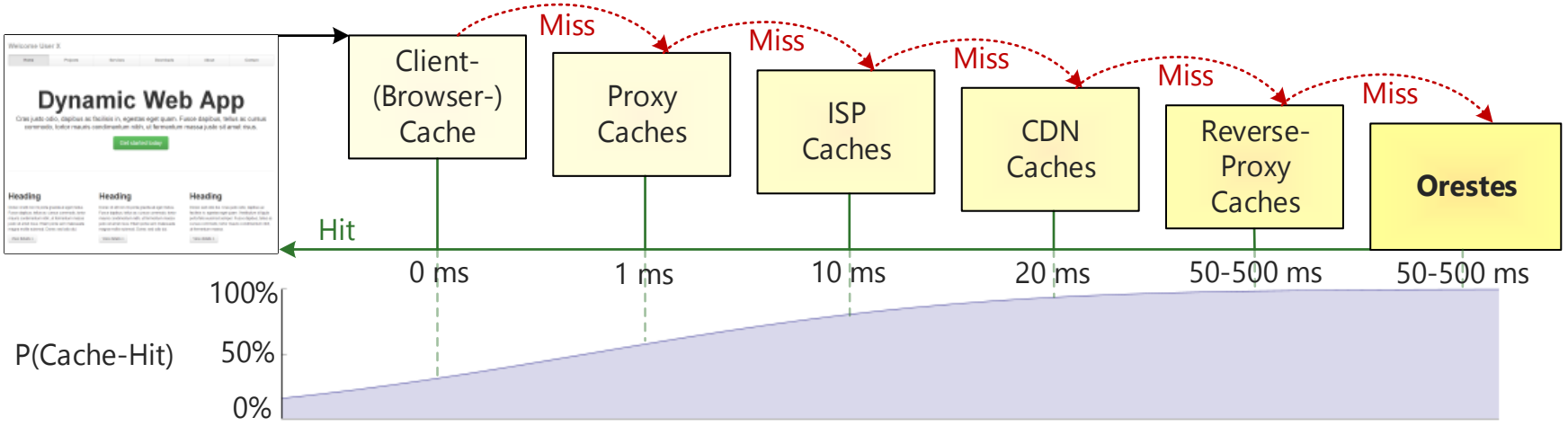

Application Layer
Persistence API
Application Server
Browser or Mobile Device
persist
find
createQuery
Java/JDO
JavaScript/JPA Port
REST/HTTP API
HTTP Server
HTTP Server
HTTP Server
Database-independent Concerns
Access Control
DBaaS & BaaS Layer
Cache Coherence
Transaction Validation
Workload Management
SLAs
Multi-Tenancy
Schema Management
Autoscaling
Database - specific Wrappers
Trans-actions
Querys
Object Persist.
Schema
Key-Value
Doc-uments
Config-uration
Partial Updates
Data Store
VERSANT
db4o
redis
mongoDB
others


Application Layer
Application Server
Browser or Mobile Device
Persistence API
persist
find
createQuery
Java/JDO
JavaScript/JPA Port
REST/HTTP API
Content Delivery Networks
Forward-Proxy Caches
ISP Caches
ISP
CDN Caches
Reverse-Proxy Caches and Load Balancers
Purge
Scale
HTTP Server
HTTP Server
HTTP Server
Database-independent Concerns
Access Control
DBaaS & BaaS Layer
Cache Coherence
Transaction Validation
Workload Management
SLAs
Multi-Tenancy
Schema Management
Autoscaling
Database - specific Wrappers
Trans-actions
Querys
Object Persist.
Schema
Key-Value
Doc-uments
Config-uration
Partial Updates
Data Store
VERSANT
db4o
redis
mongoDB
others

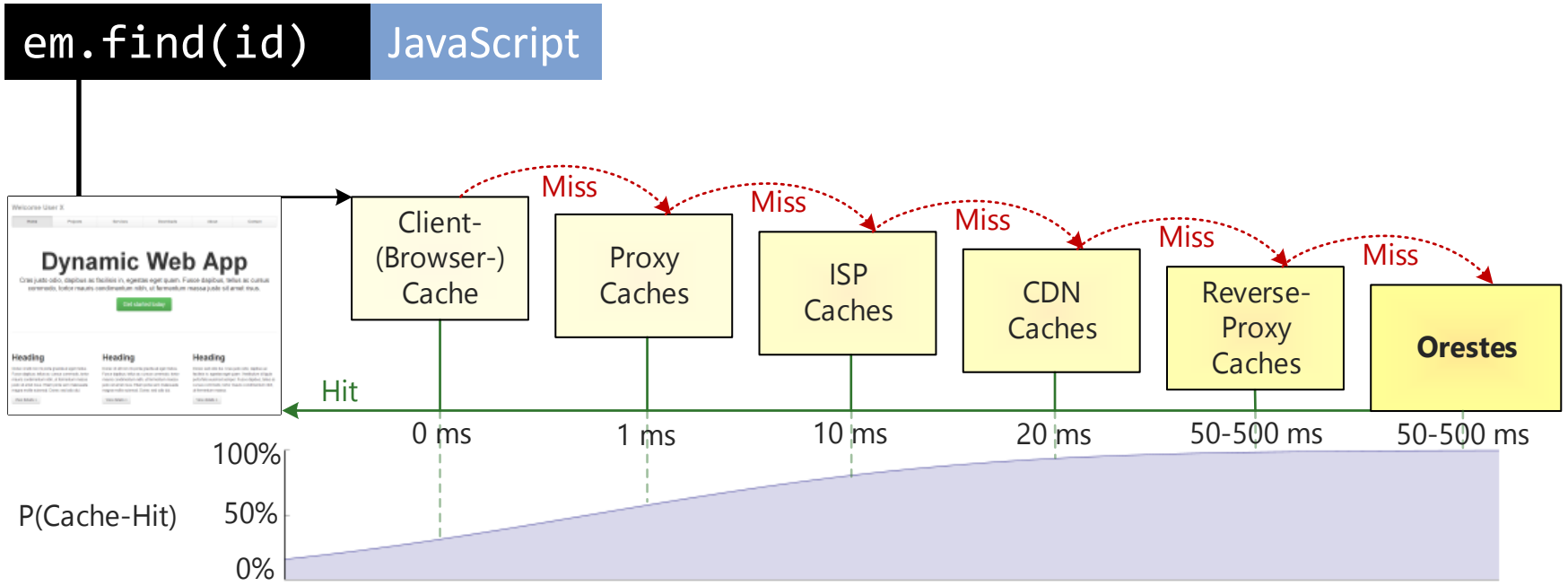

Application Layer
Application Server
Browser or Mobile Device
Persistence API
persist
find
createQuery
Java/JDO
JavaScript/JPA Port
REST/HTTP API
Content Delivery Networks
Forward-Proxy Caches
ISP Caches
ISP
CDN Caches
Reverse-Proxy Caches and Load Balancers
Purge
Scale
HTTP Server
HTTP Server
HTTP Server
Database-independent Concerns
Access Control
DBaaS & BaaS Layer
Cache Coherence
add
delete
Redis (Replicated)
Counting Bloom Filter
10101010
10201040
Transaction Validation
Workload Management
SLAs
Multi-Tenancy
Schema Management
Autoscaling
Node.JS (local to Server)
Stored Procedures
Custom Validation
Database - specific Wrappers
Trans-actions
Querys
Object Persist.
Schema
Key-Value
Doc-uments
Config-uration
Partial Updates
Data Store
VERSANT
db4o
redis
mongoDB
others


Application Layer
Application Server
Browser or Mobile Device
Persistence API
persist
find
createQuery
Java/JDO
JavaScript/JPA Port
REST/HTTP API
Forward-Proxy Caches
GET /db/{bucket}/{class}/{id}
ISP Caches
ISP
Content Delivery Networks
CDN Caches
200 OK
Cache-Control: public, max-age=6000
ETag: "3"
JSON Object
Reverse-Proxy Caches and Load Balancers
Purge
Scale
HTTP Server
HTTP Server
HTTP Server
Database-independent Concerns
Access Control
DBaaS & BaaS Layer
Cache Coherence
add
delete
Redis (Replicated)
Counting Bloom Filter
10101010
10201040
Transaction Validation
Workload Management
SLAs
Multi-Tenancy
Schema Management
Autoscaling
Node.JS (local to Server)
Stored Procedures
Custom Validation
Database - specific Wrappers
Trans-actions
Querys
Object Persist.
Schema
Key-Value
Doc-uments
Config-uration
Partial Updates
Data Store
VERSANT
db4o
redis
mongoDB
others

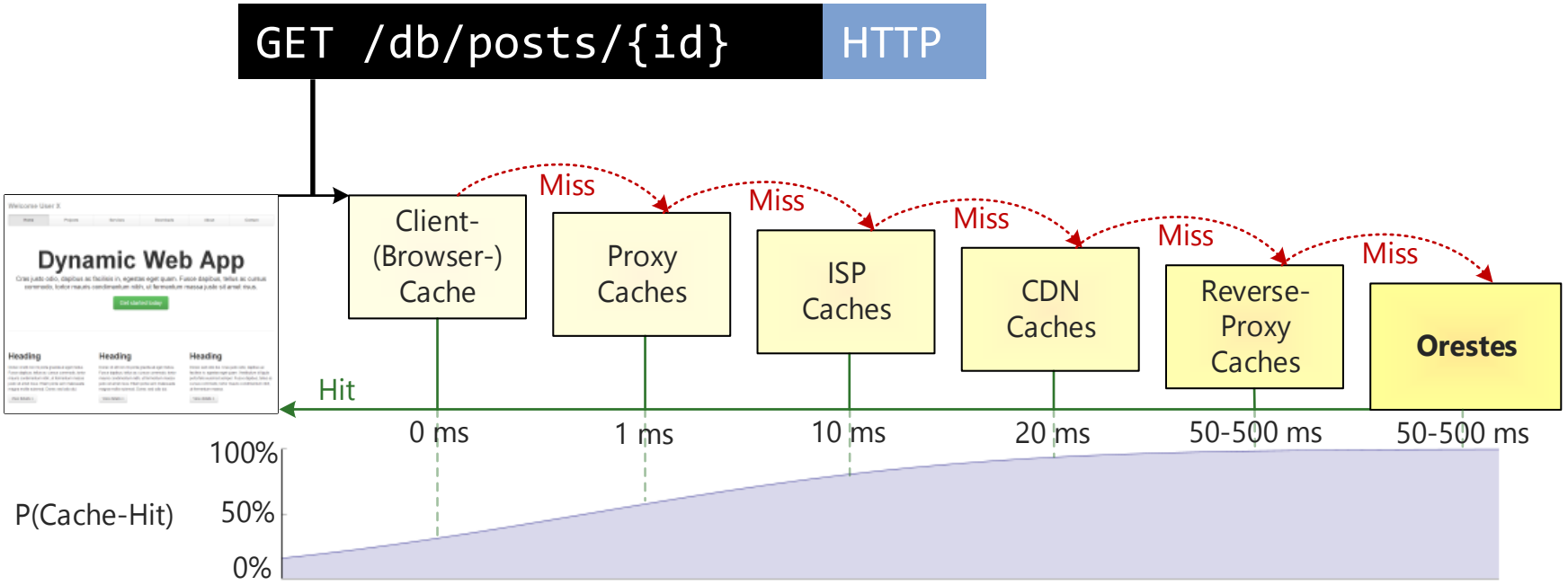
Caching


Dynamic Web App
Client-
(Browser-)
Cache
Proxy
Caches
ISP
Caches
CDN
Caches
Reverse-
Proxy
Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)

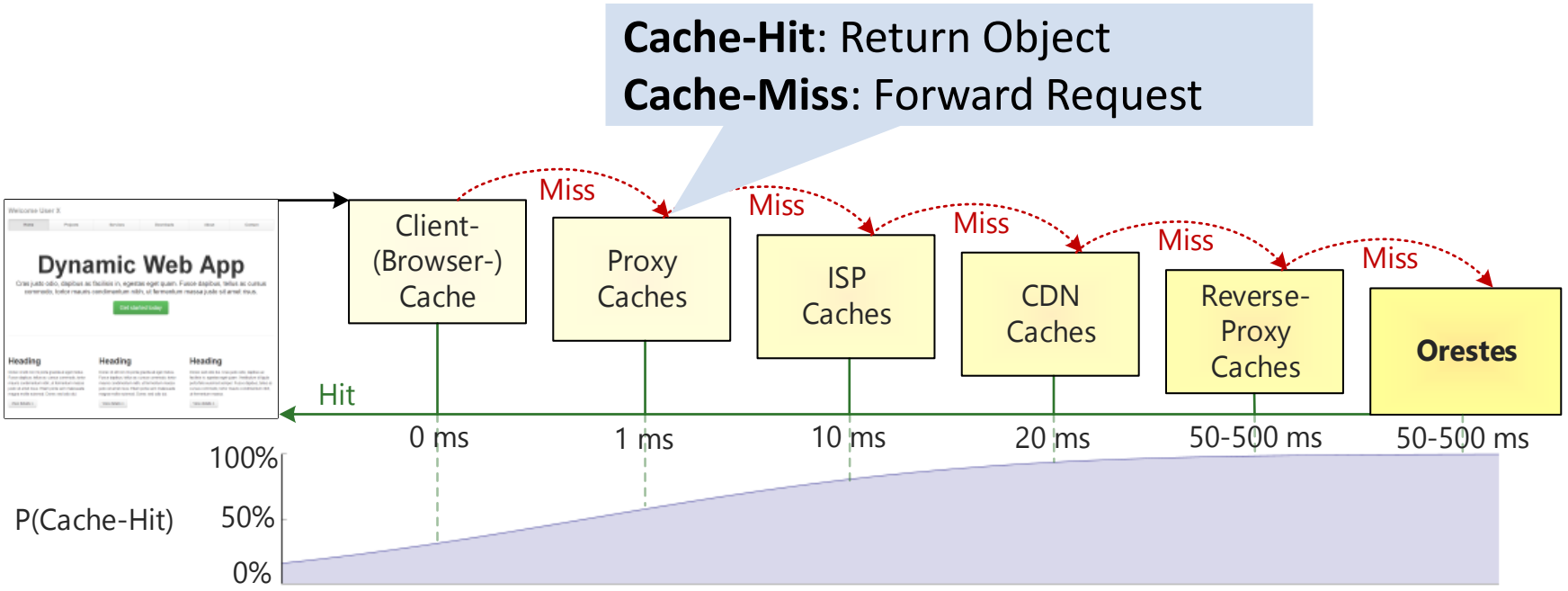
Caching


em.find(id)
JavaScript
Dynamic Web App
Client-
(Browser-)
Cache
Proxy
Caches
ISP
Caches
CDN
Caches
Reverse-
Proxy
Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)

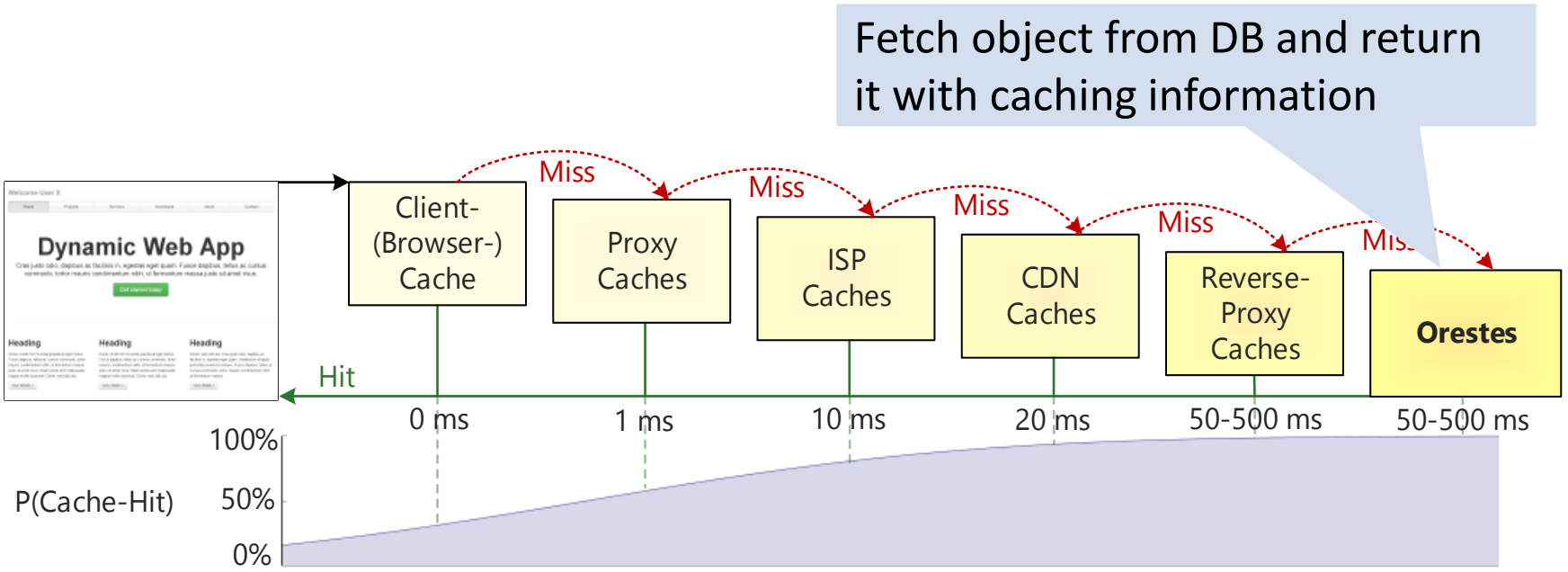
Caching


GET /db/posts/{id}
HTTP
Dynamic Web App
Client-
(Browser-)
Cache
Proxy
Caches
ISP
Caches
CDN
Caches
Reverse-
Proxy
Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)

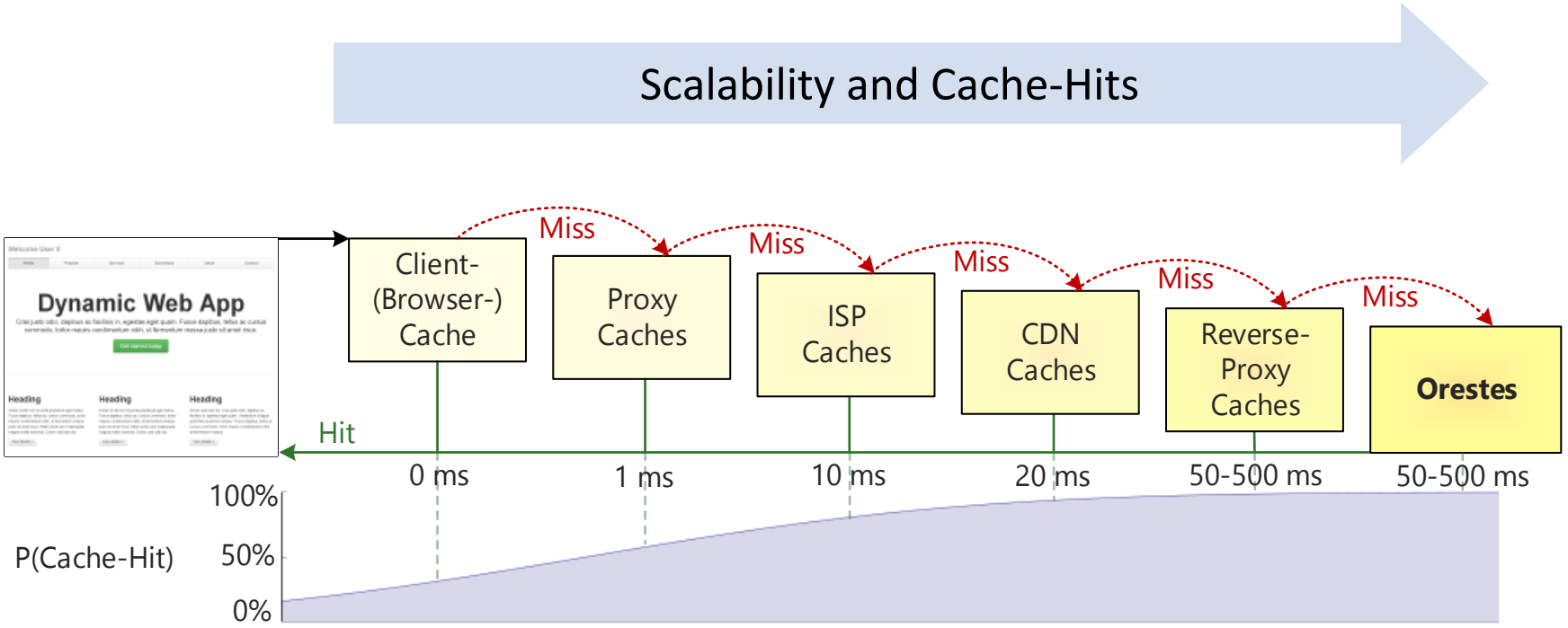
Caching


Cache-Hit: Return Object
Cache-Miss: Forward Request
Dynamic Web App
Client-
(Browser-)
Cache
Proxy
Caches
ISP
Caches
CDN
Caches
Reverse-
Proxy
Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)

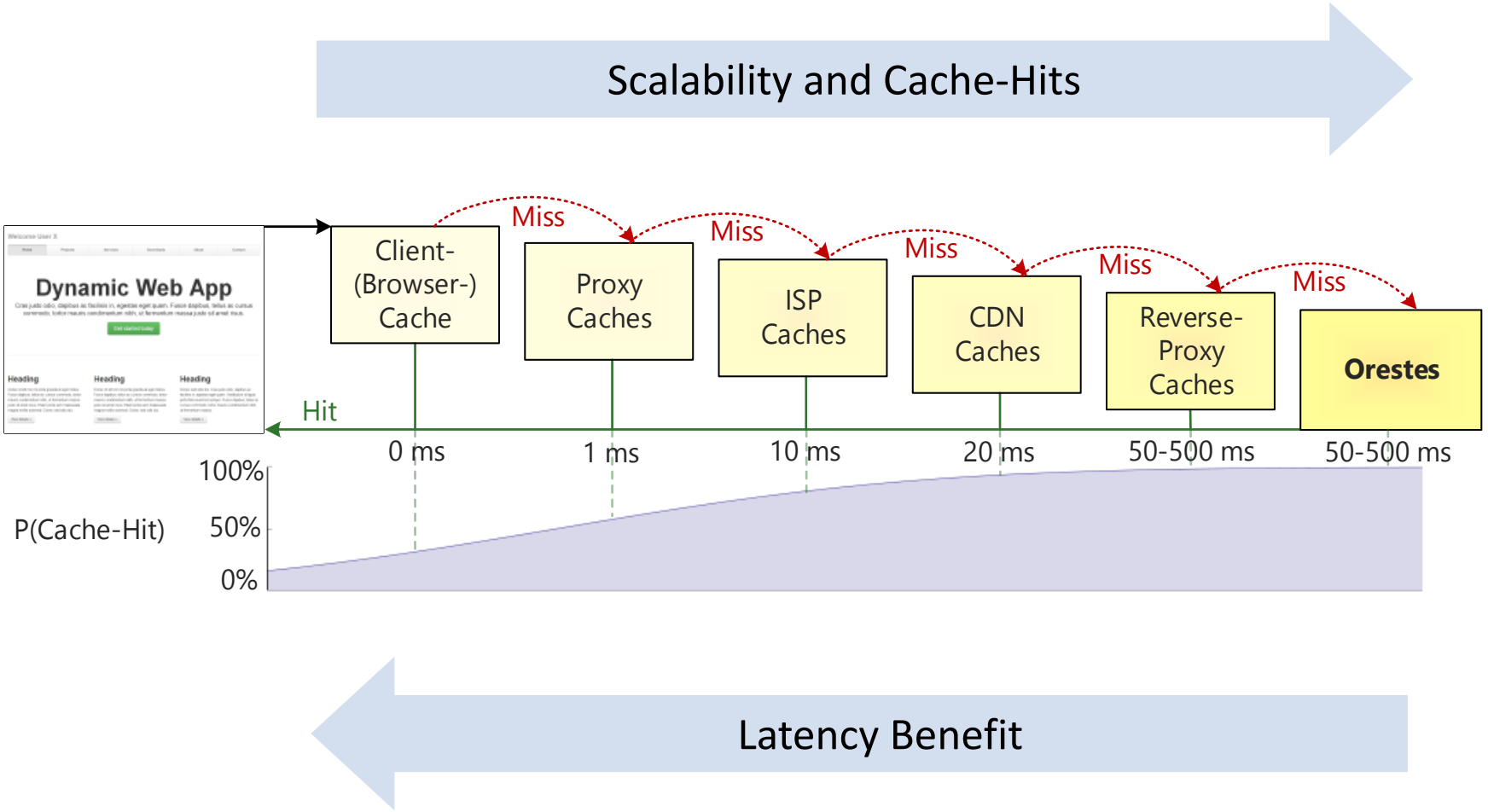
Caching


Fetch object from DB and return it with caching information
Dynamic Web App
Client-(Browser-) Cache
Proxy Caches
ISP Caches
CDN Caches
Reverse-Proxy Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)

Caching


Scalability and Cache-Hits
Dynamic Web App
Client-
(Browser-)
Cache
Proxy
Caches
ISP
Caches
CDN
Caches
Reverse-
Proxy
Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)

Caching


Scalability and Cache-Hits
Dynamic Web App
Client-
(Browser-)
Cache
Proxy
Caches
ISP
Caches
CDN
Caches
Reverse-
Proxy
Caches
Orestes
Miss
Miss
Miss
Miss
Miss
Hit
0 ms
1 ms
10 ms
20 ms
50-500 ms
50-500 ms
100%
50%
0%
P(Cache-Hit)
Latency Benefit

BFB: Bloom Filter Bounded Staleness

Cache

0	2	1	4	0

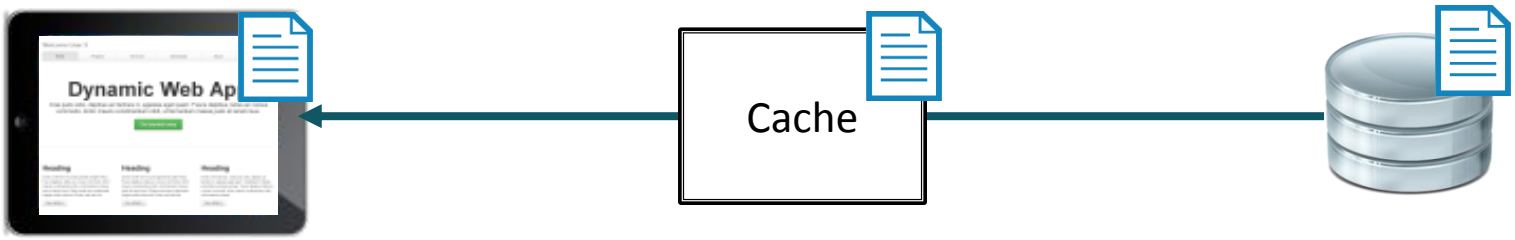
BFB: Bloom Filter Bounded Staleness

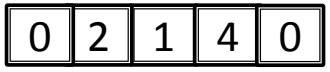
How to prevent **stale reads** (inconsistency)?

Cache

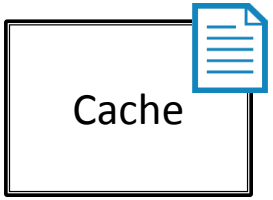
0	2	1	4	0

BFB: Bloom Filter Bounded Staleness



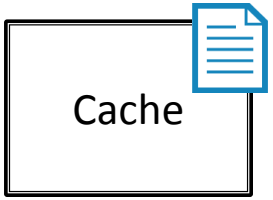


BFB: Bloom Filter Bounded Staleness



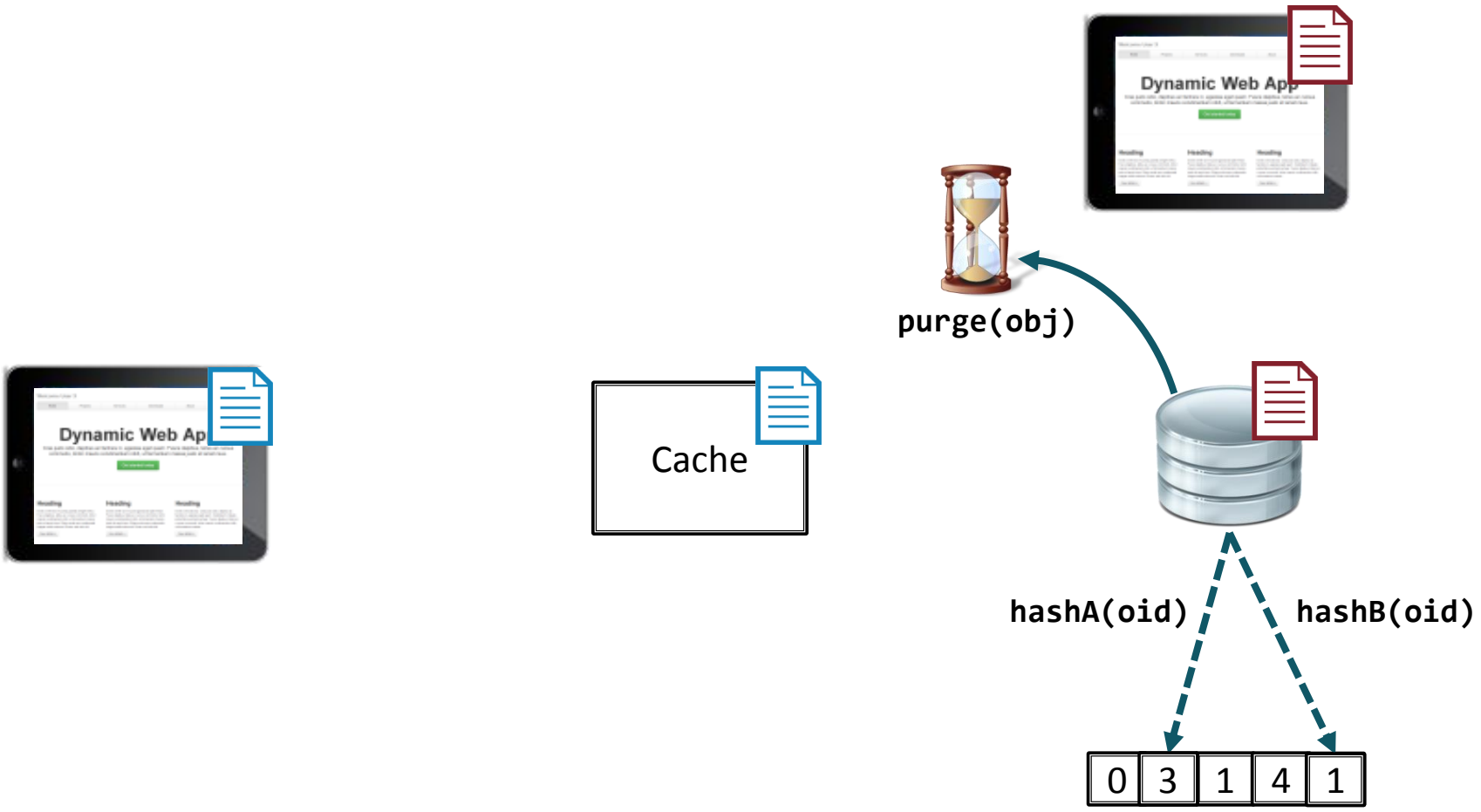
0	2	1	4	0

BFB: Bloom Filter Bounded Staleness

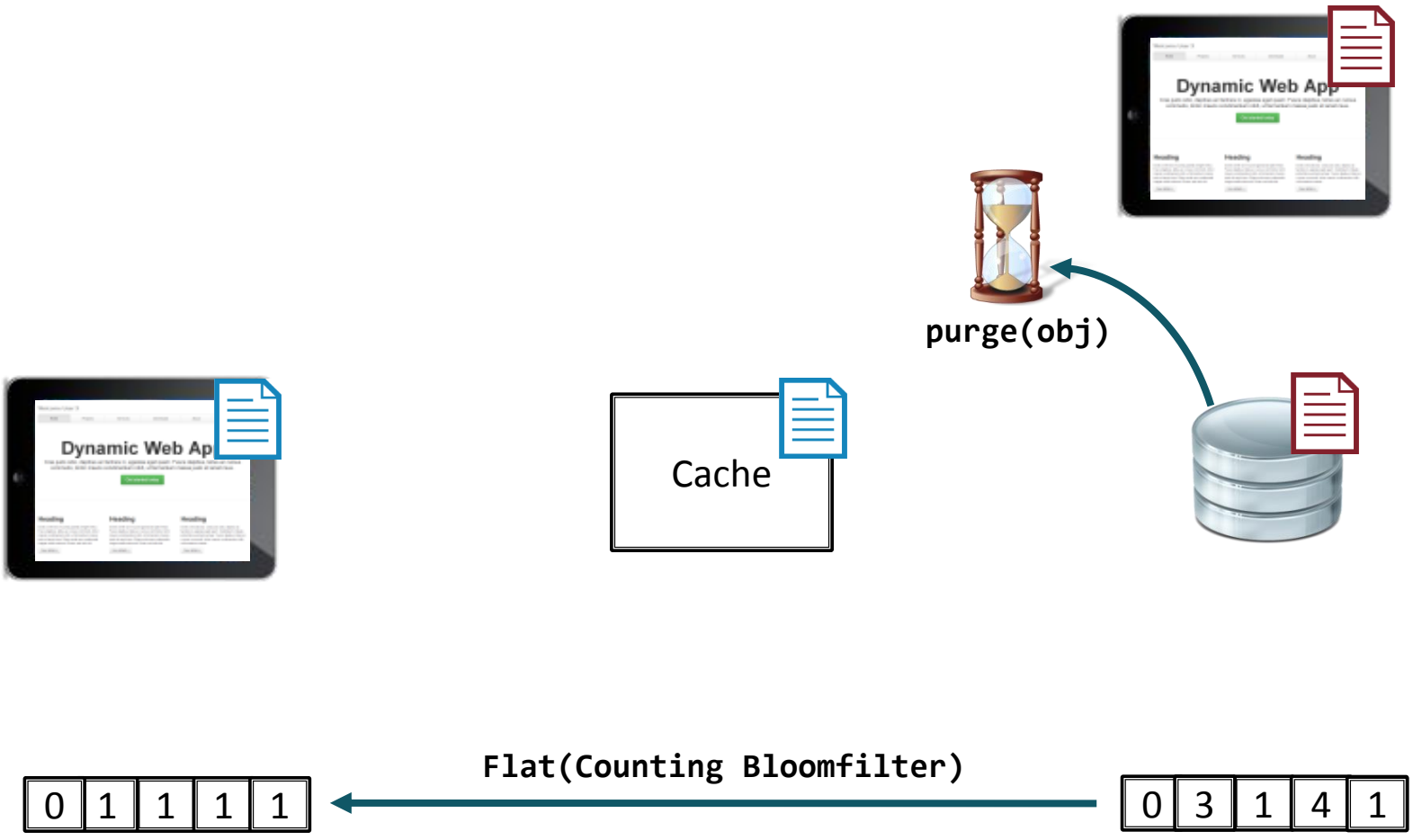


0	2	1	4	0

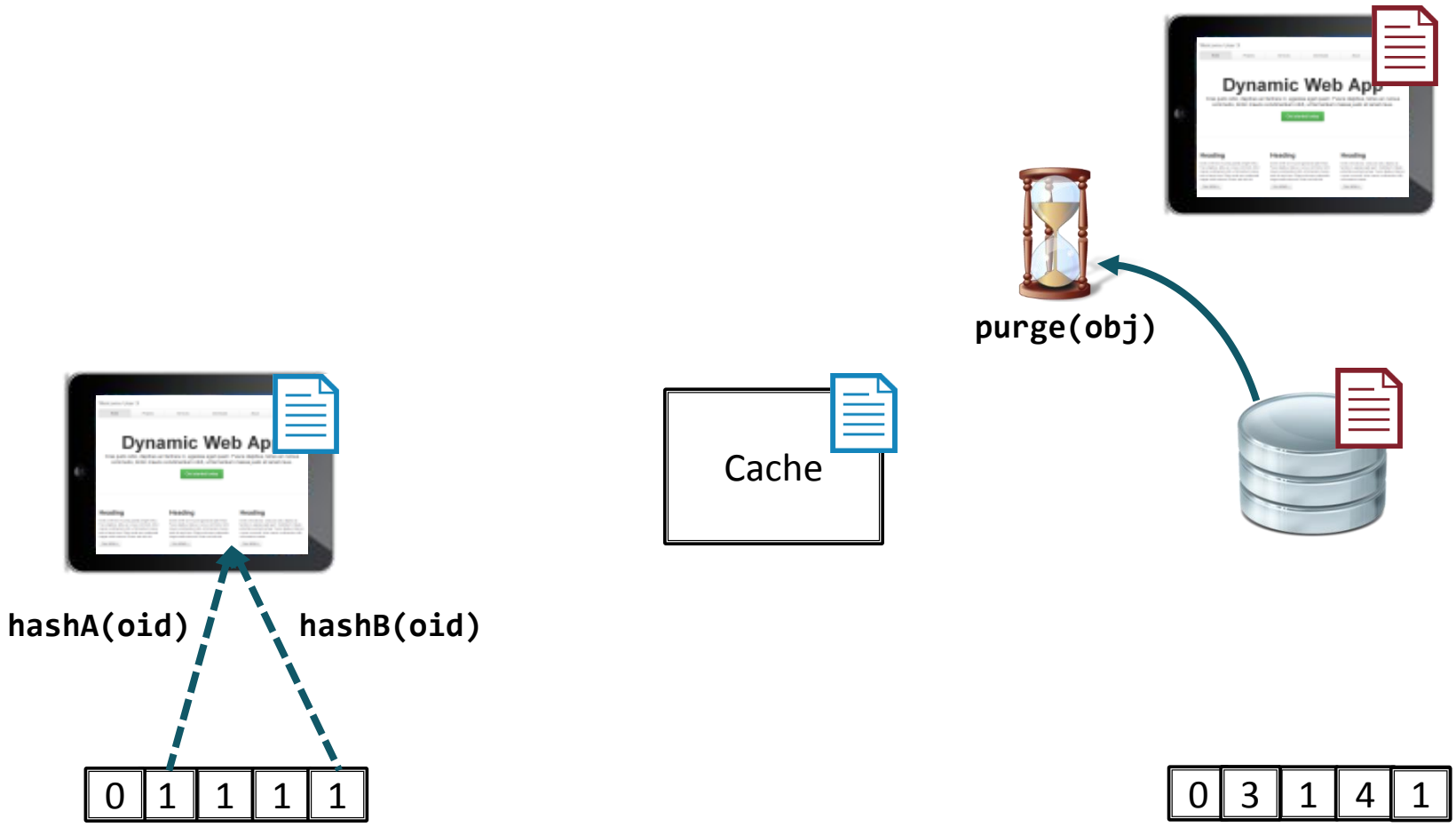
BFB: Bloom Filter Bounded Staleness



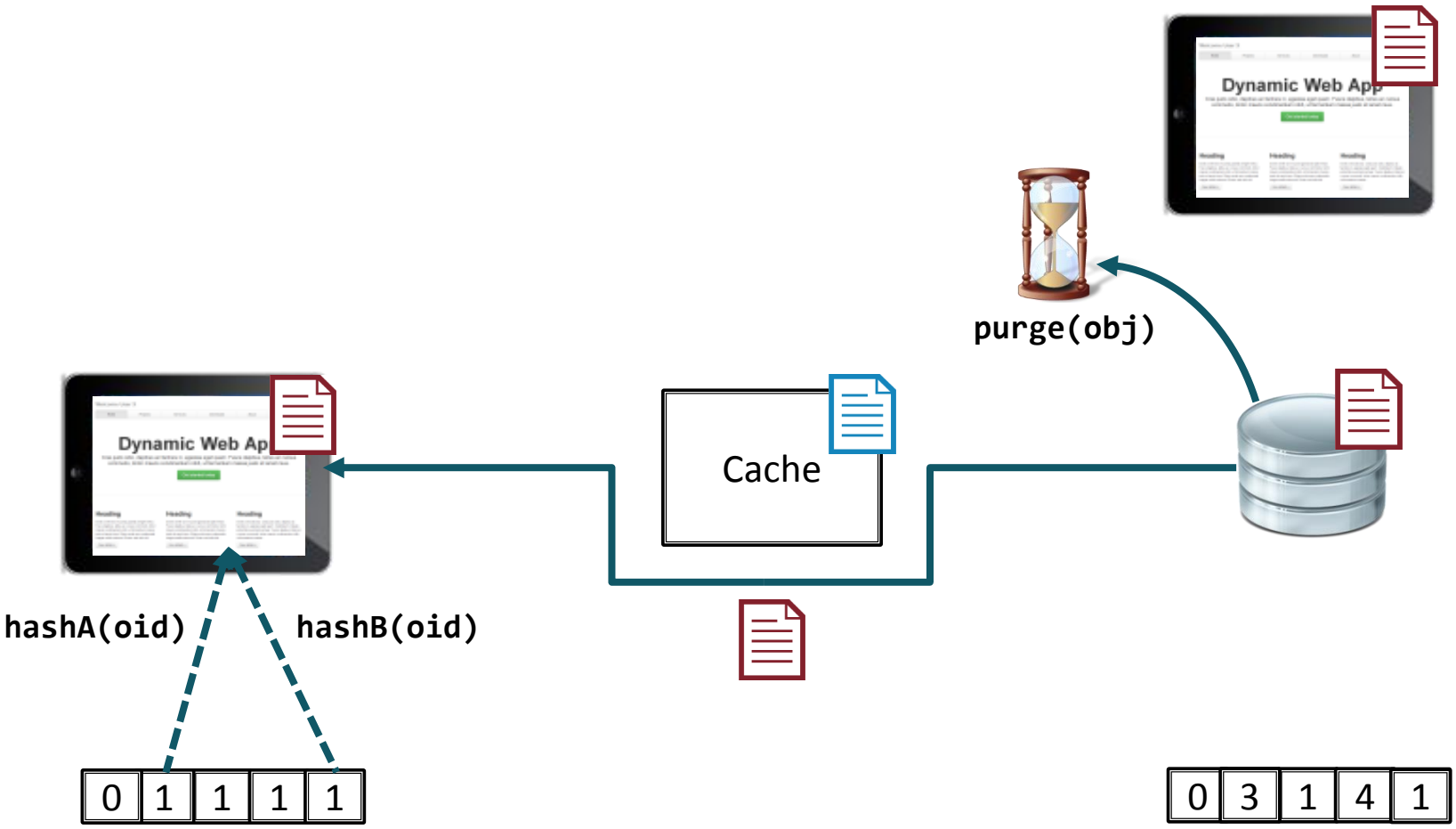
BFB: Bloom Filter Bounded Staleness



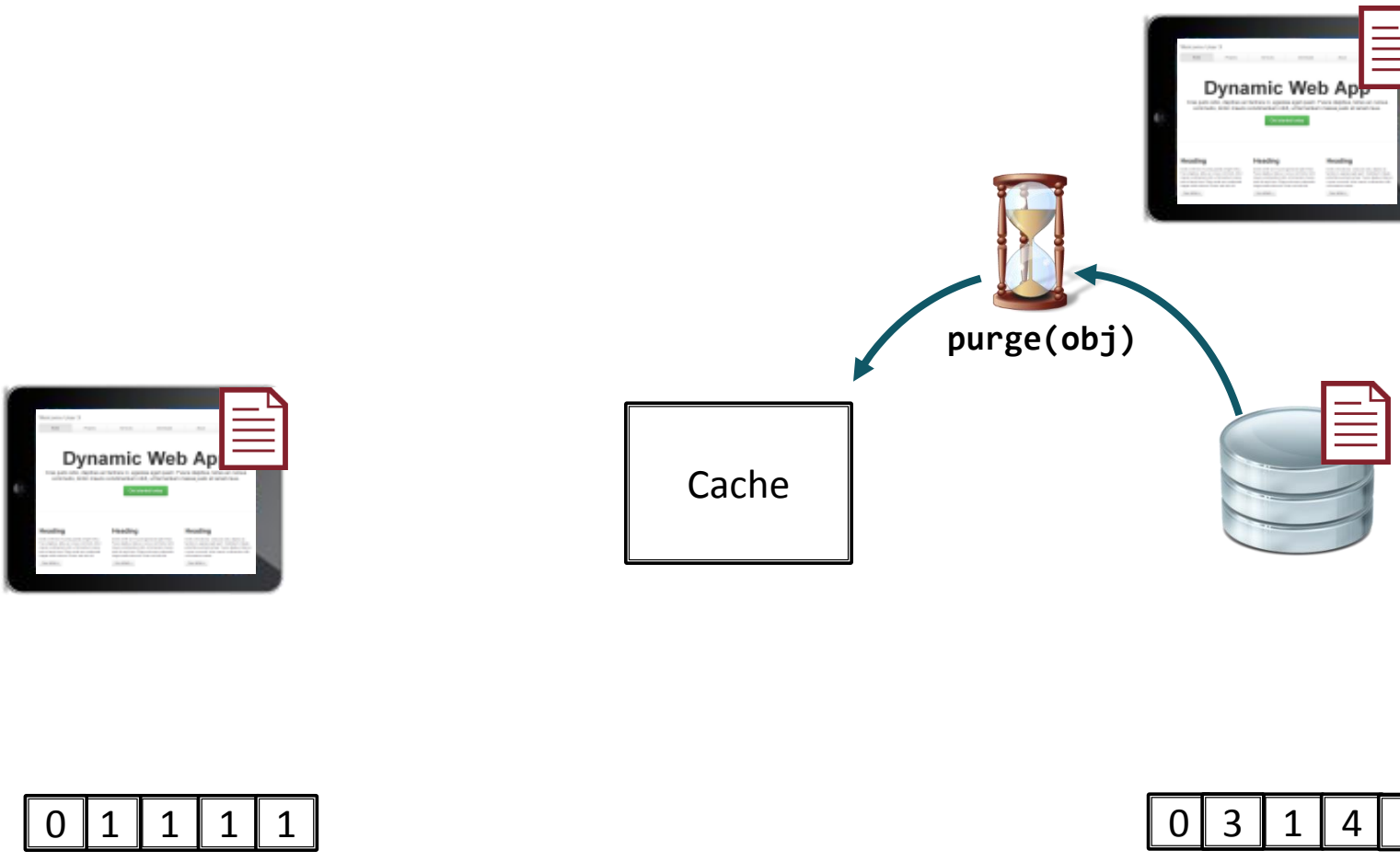
BFB: Bloom Filter Bounded Staleness



BFB: Bloom Filter Bounded Staleness



BFB: Bloom Filter Bounded Staleness



BFB: Bloom Filter Bounded Staleness

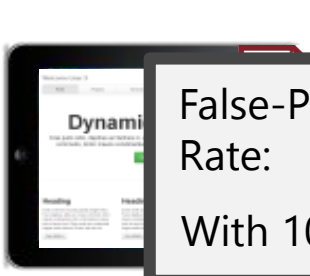
Cache

hashA(oid) hashB(oid)

0	1	1	1	1

0	2	1	4	0

BFB: Bloom Filter Bounded Staleness



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 10.000 Updates per 10 minutes and 1% error rate: **12 KByte**

`hashA(oid)` `hashB(oid)`

0	1	1	1	1

0	2	1	4	0

BFB: Bloom Filter Bounded Staleness

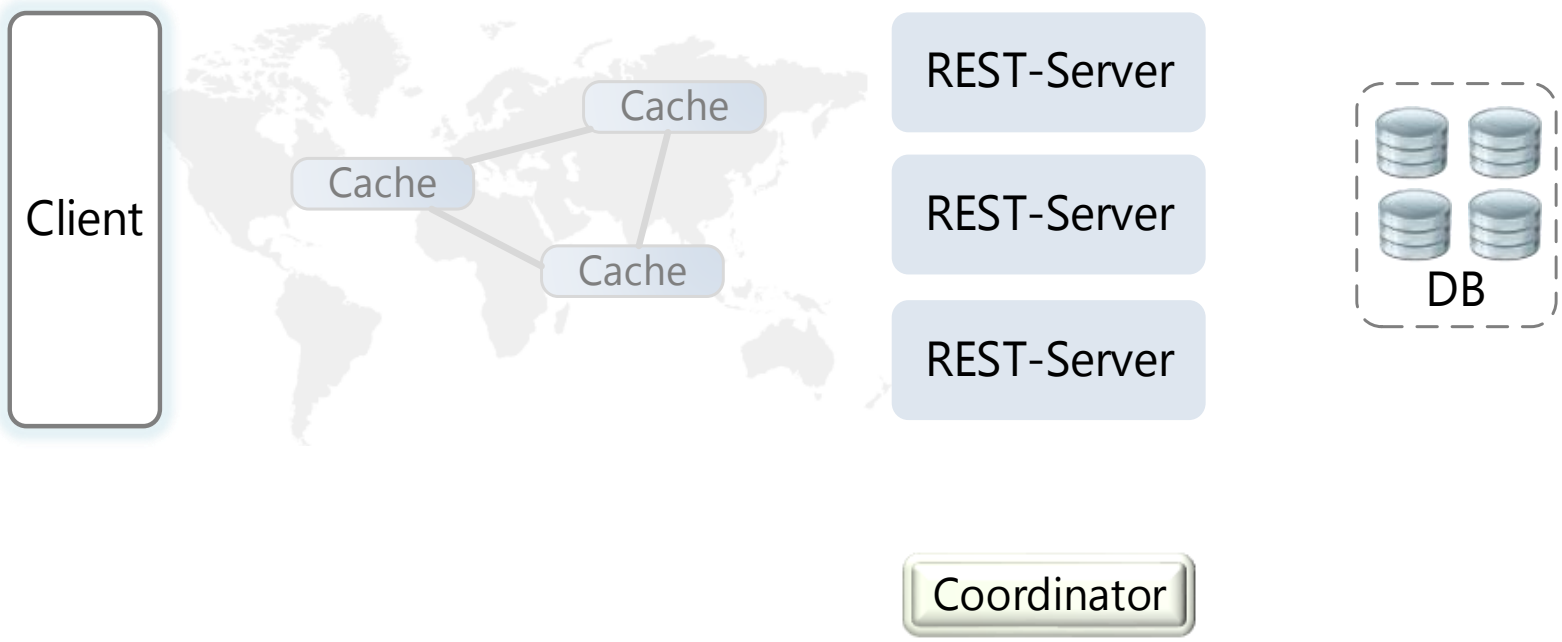
Cache

hashA(oid) hashB(oid)

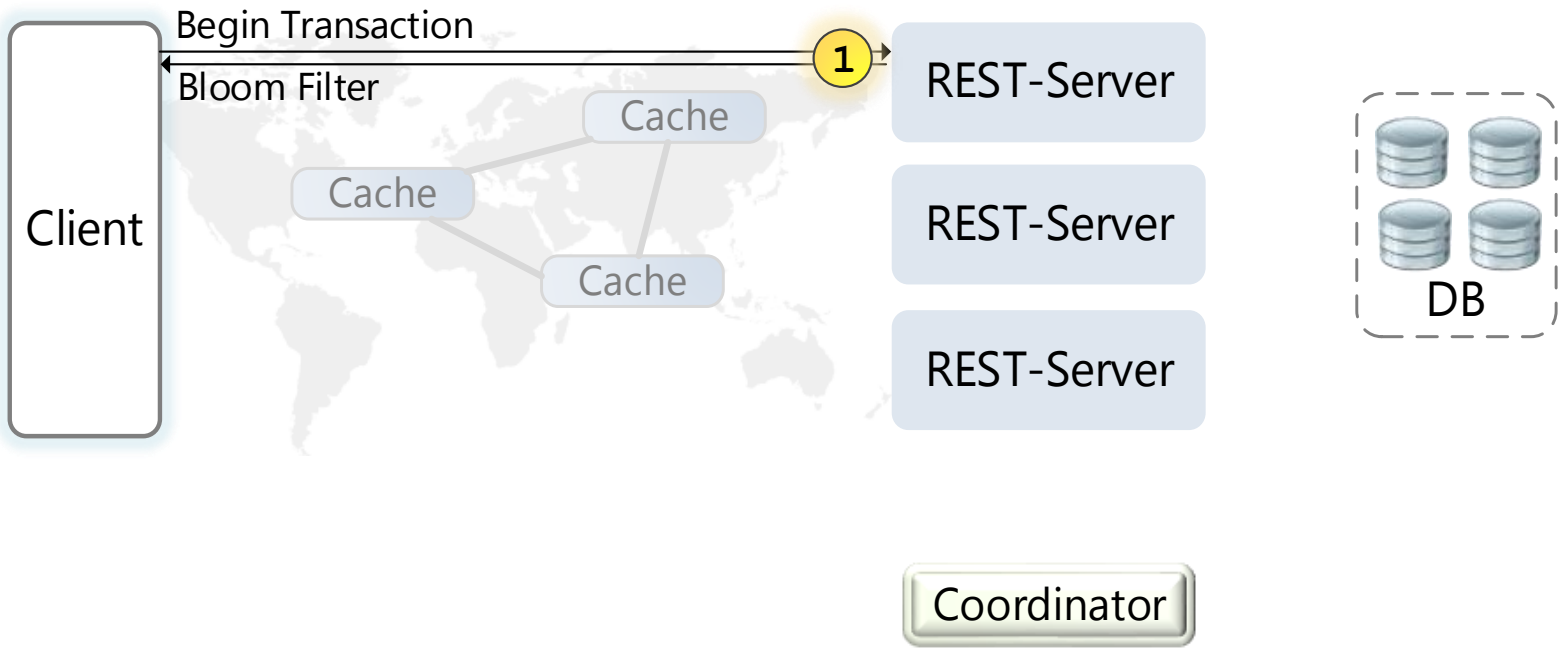
0	1	1	1	1

0	2	1	4	0

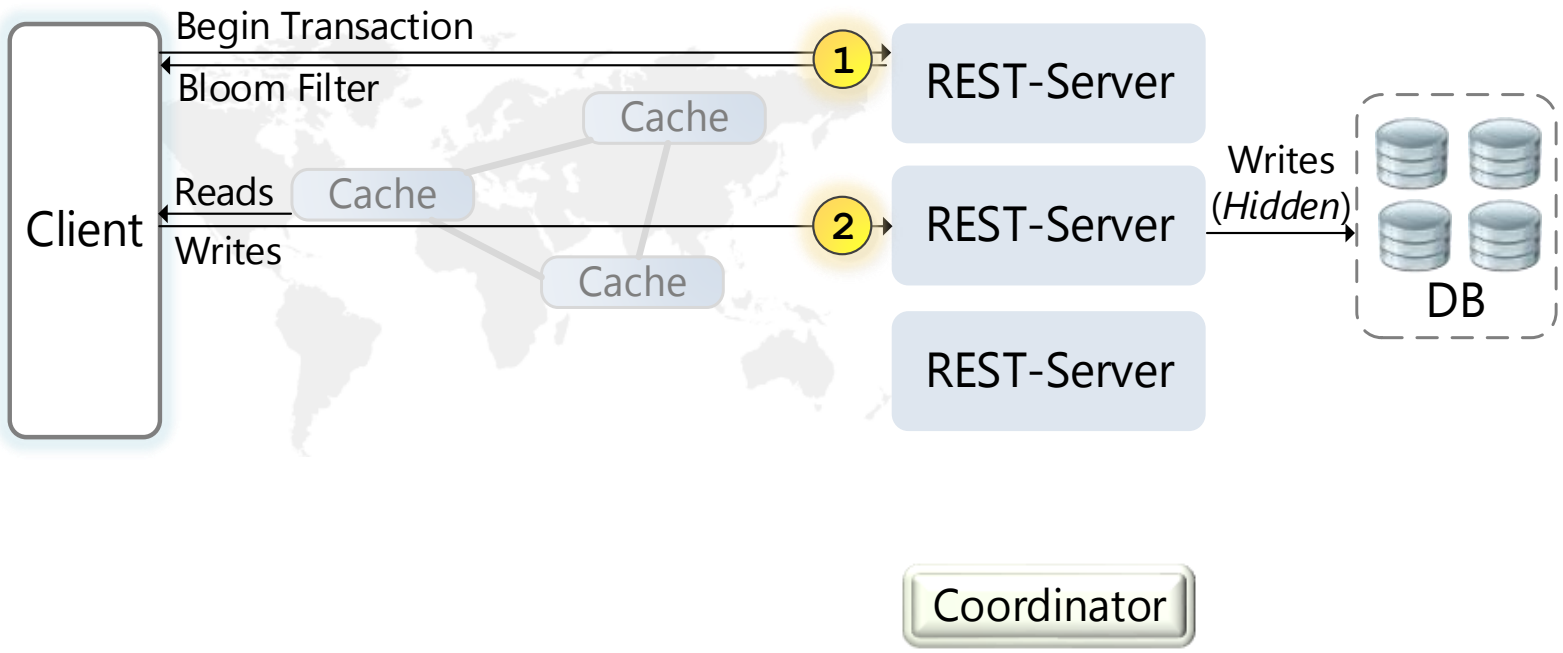
SCOT: Scalable Cache-Aware Optimistic Transactions



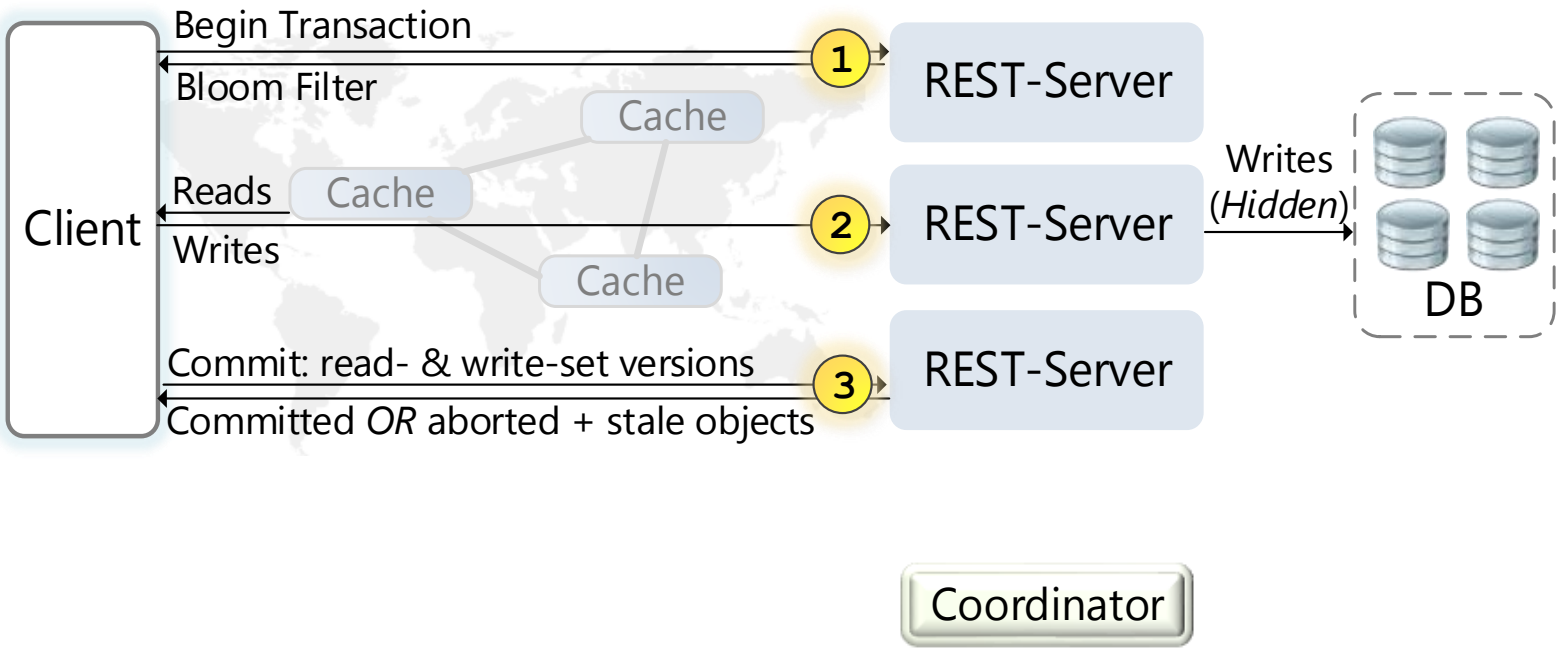
SCOT: Scalable Cache-Aware Optimistic Transactions



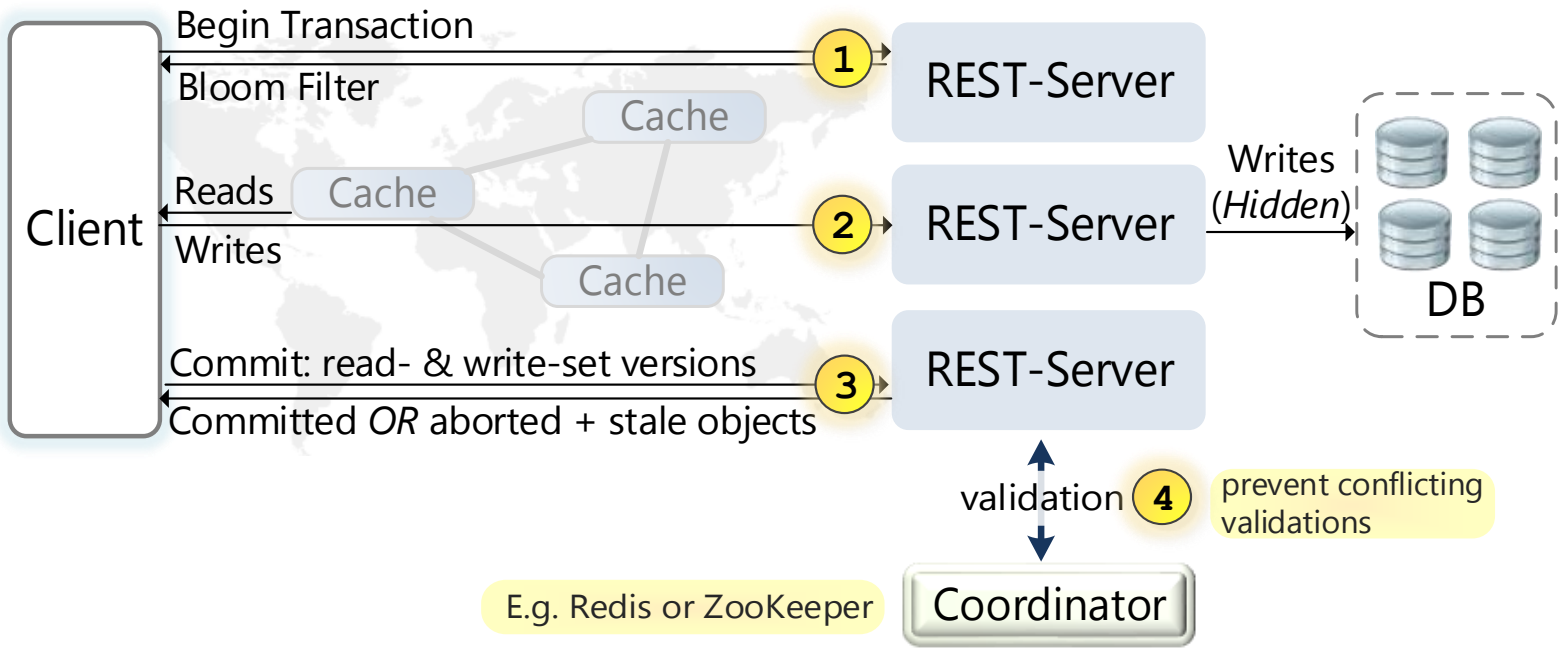
SCOT: Scalable Cache-Aware Optimistic Transactions



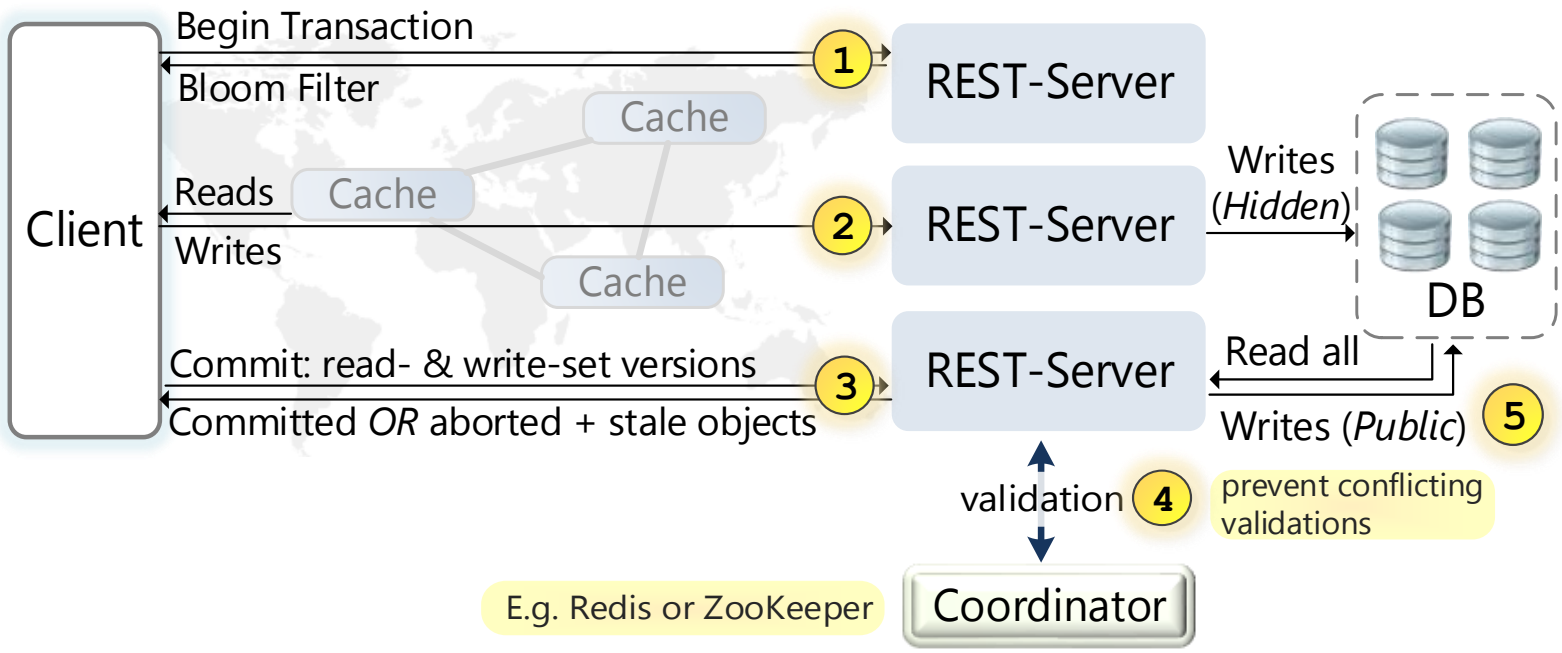
SCOT: Scalable Cache-Aware Optimistic Transactions



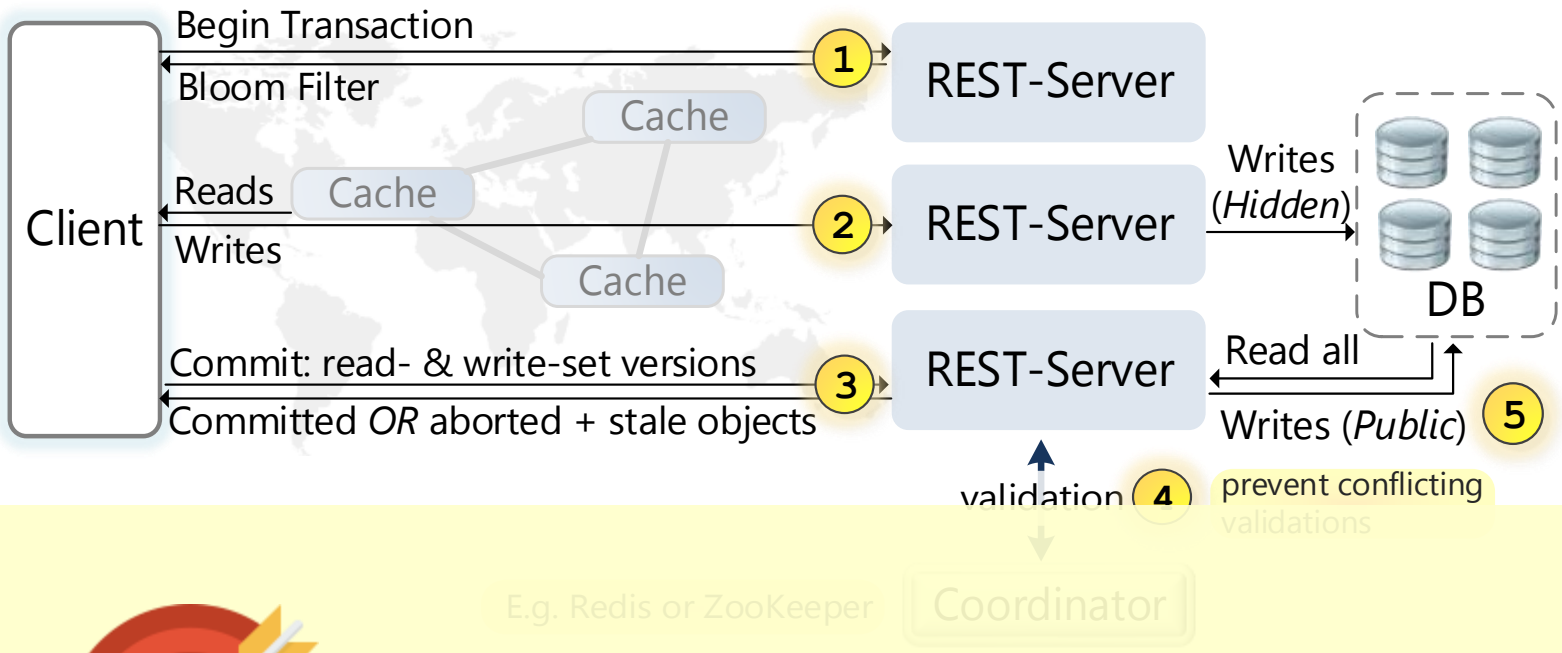
SCOT: Scalable Cache-Aware Optimistic Transactions



SCOT: Scalable Cache-Aware Optimistic Transactions

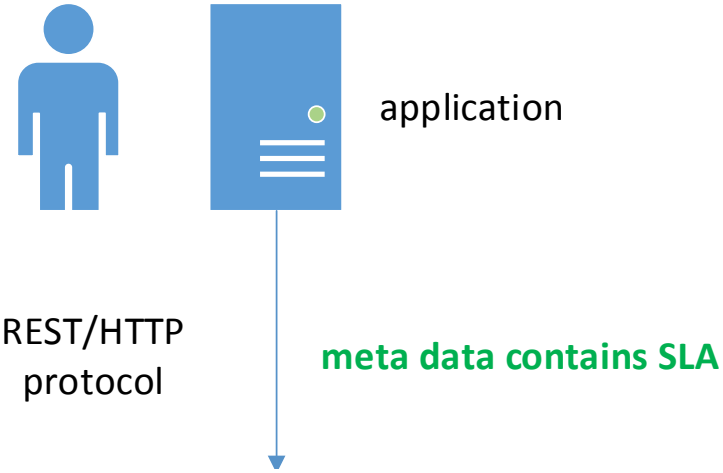


SCOT: Scalable Cache-Aware Optimistic Transactions

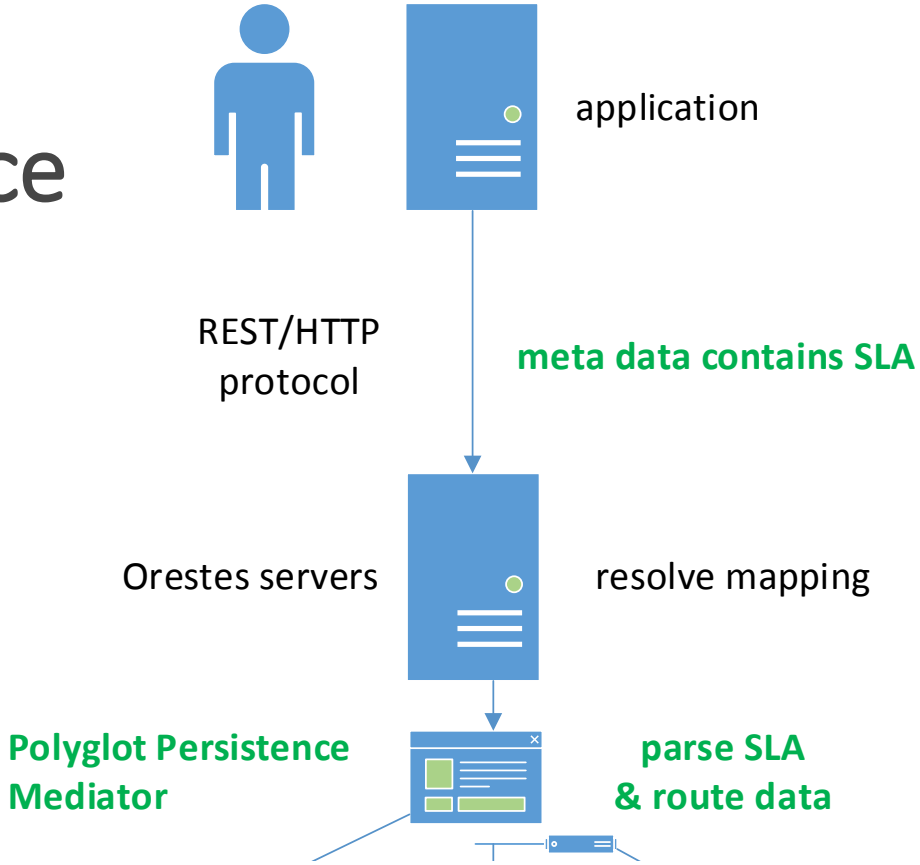


Caching → Shorter transaction duration → **less aborts**

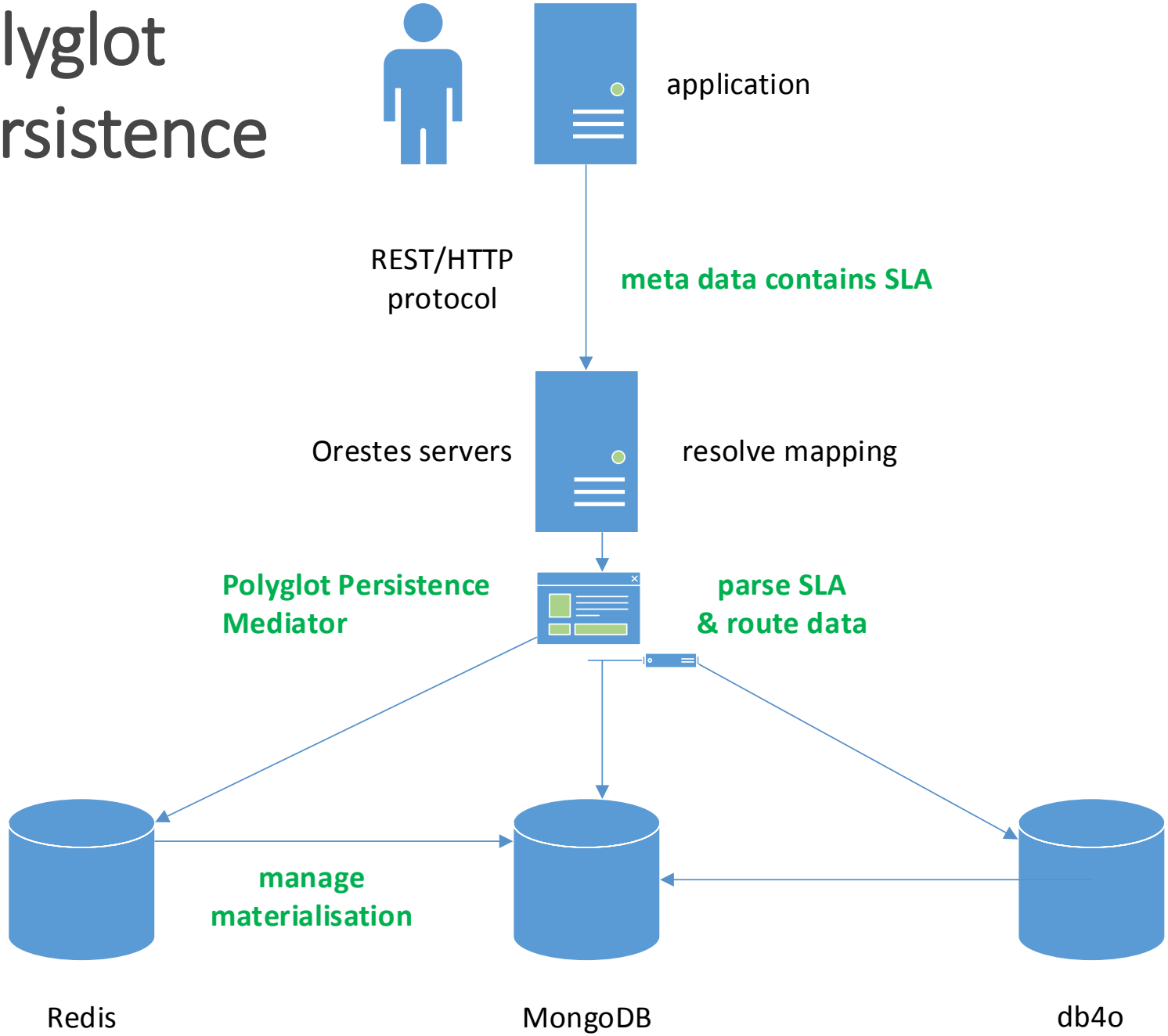
Polyglot Persistence



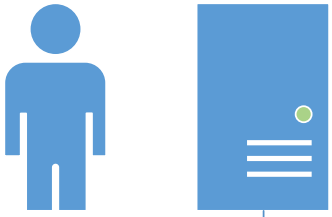
Polyglot Persistence



Polyglot Persistence

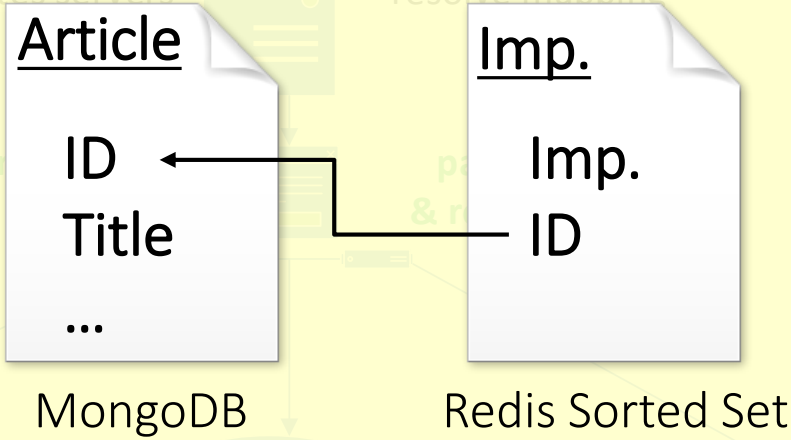


Polyglot Persistence



Results:

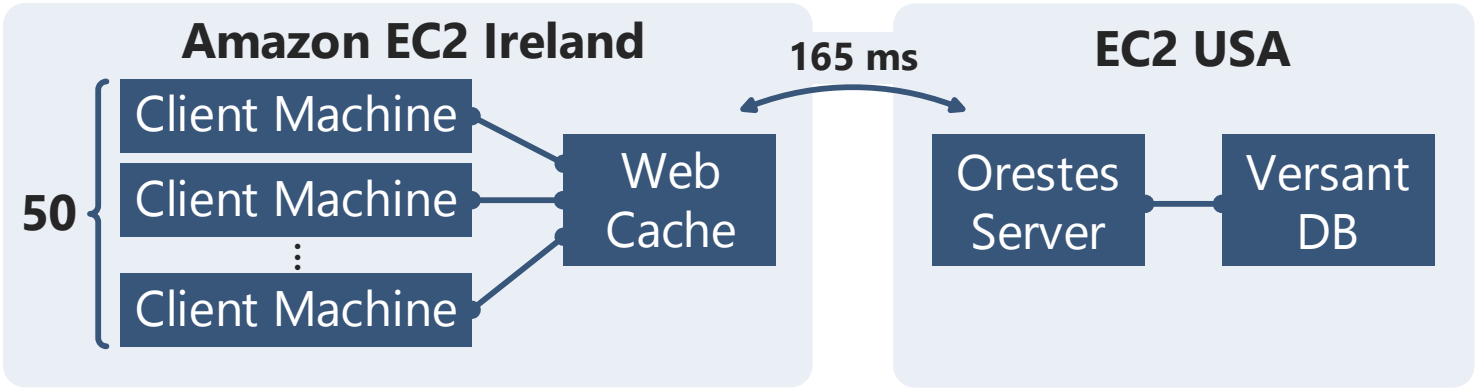
Article-Objects with Impression Count



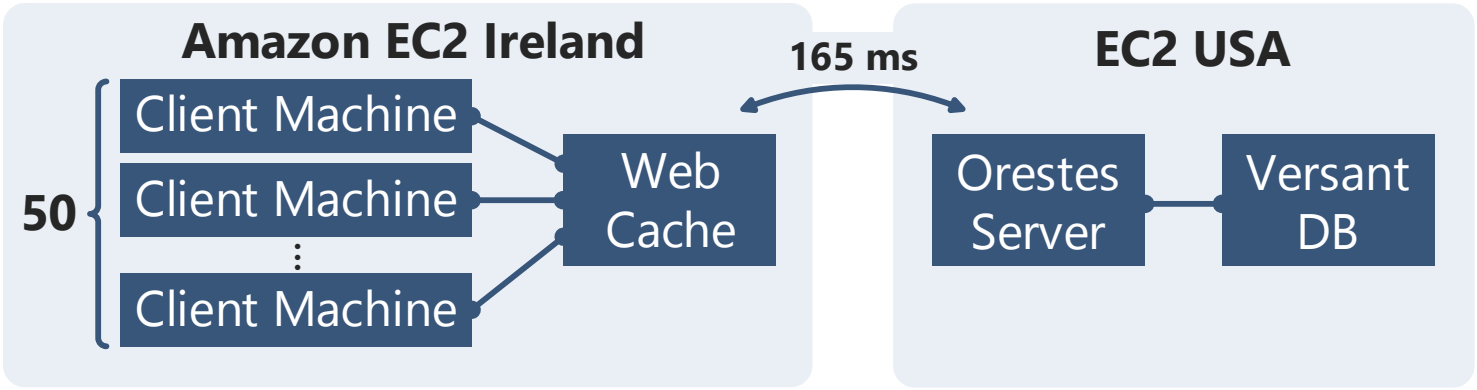
Speedup with PPM:

- **50-1000%**
- **66%** performance of Varnish

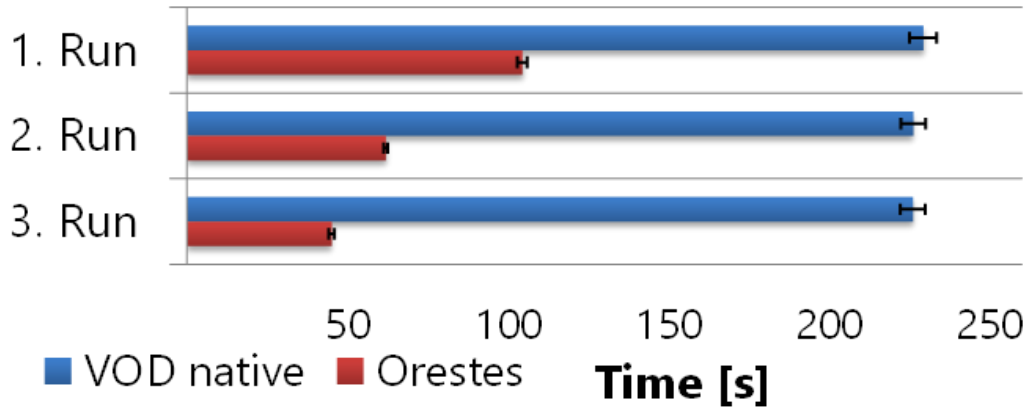
Cloud Evaluation of ORESTES



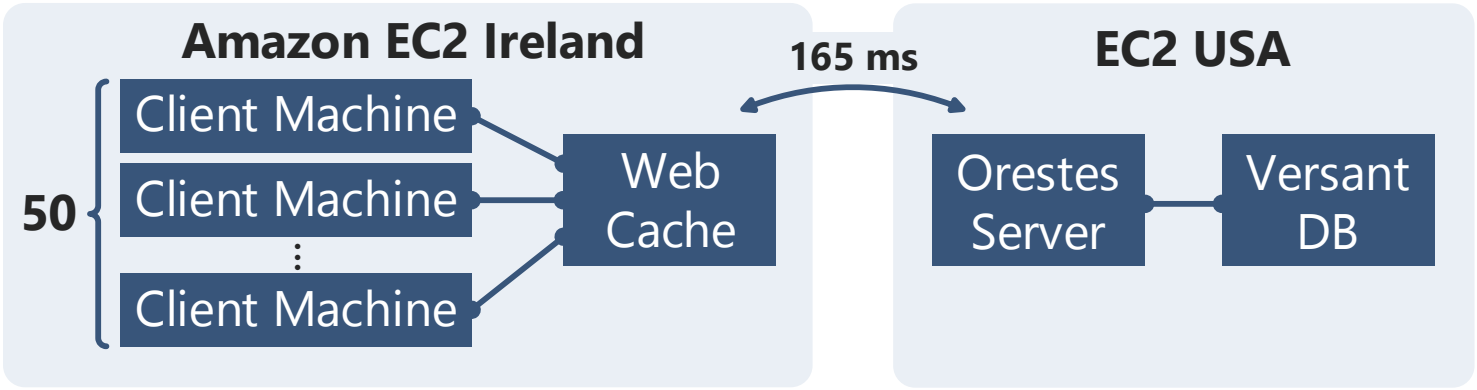
Cloud Evaluation of ORESTES

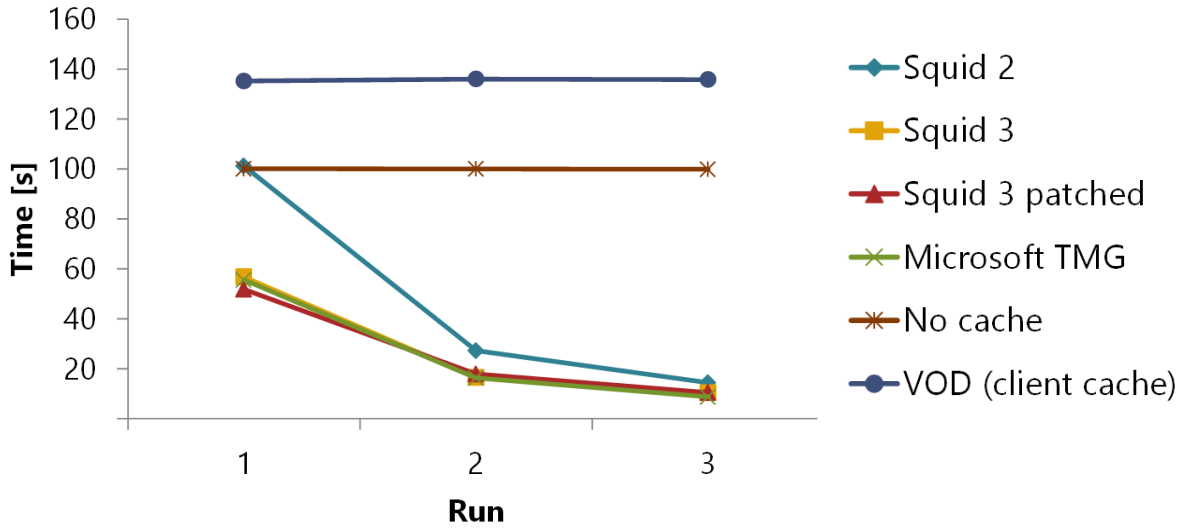


30 000 Objects
500 Req./Clients
10/1 Read/Write



Cloud Evaluation of ORESTES





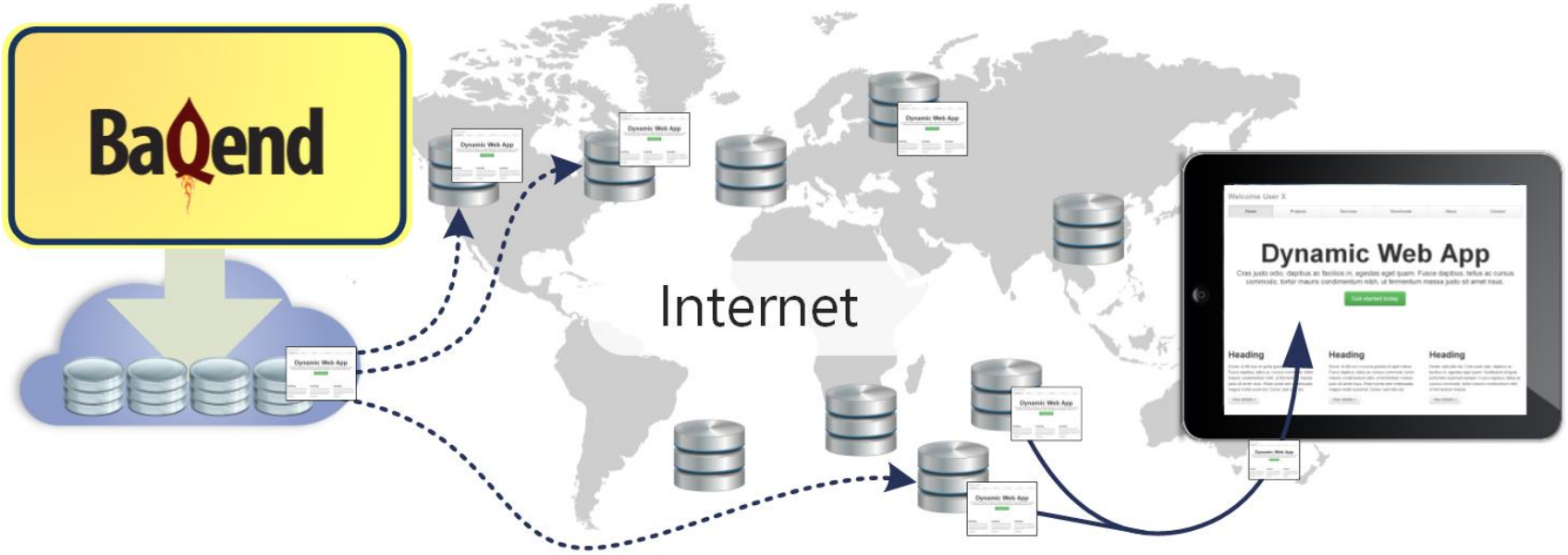
Orestes: Summary

- Stateless Scale-out REST middleware for DBaaS
- Generic Schema, Authentication, Multi-Tenancy, etc.
- **SCOT** (**S**calable **O**ptimistic **C**ache-Aware **T**ransactions):
  - Optimistic Database-indepdent ACID transactions
- **BFB** (**B**loom **F**ilter **B**ounded **S**taleness):
  - Allows static caching with tunable consistency guarantee
- **PPM** (**P**olyglot **P**ersistence **M**ediator):
  - SLAs → appropriate database

From Research to Practice

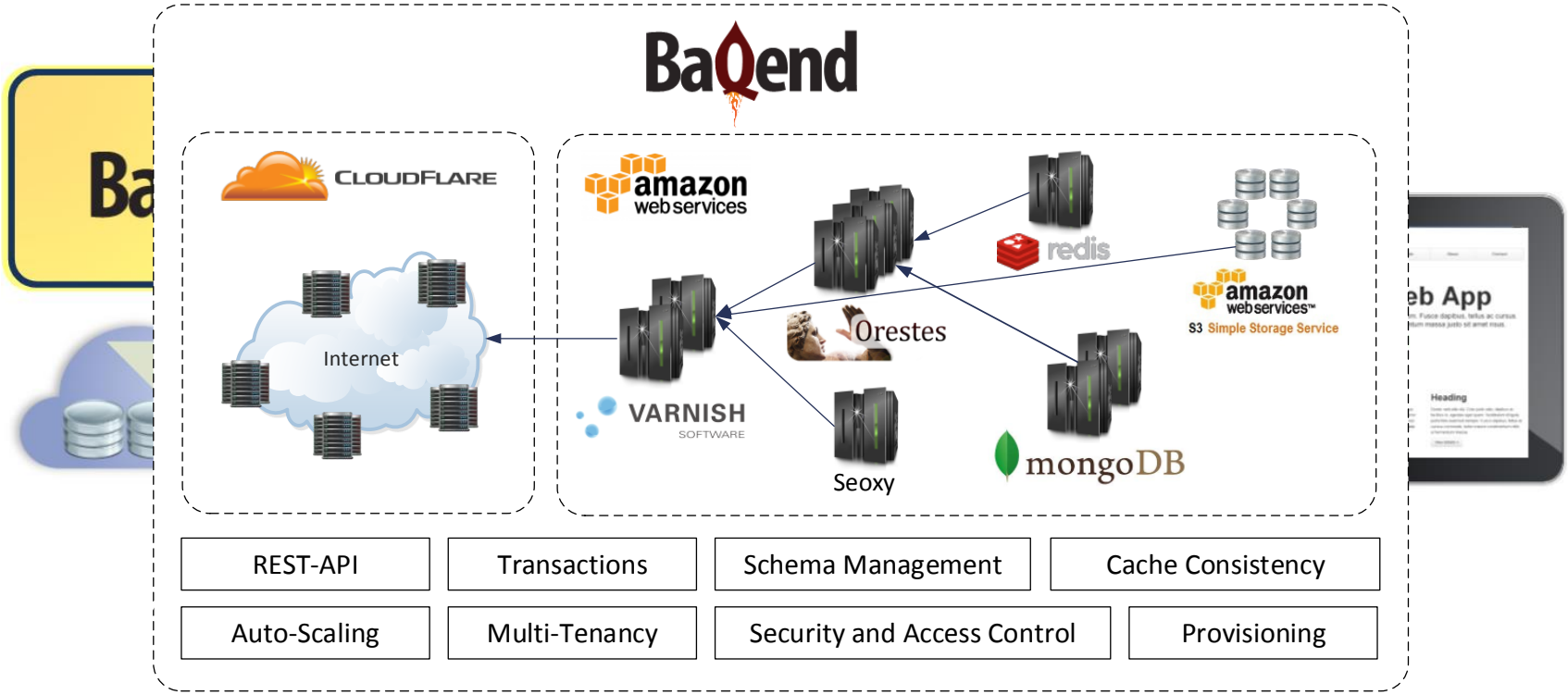
Baqend

- Orestes as a startup



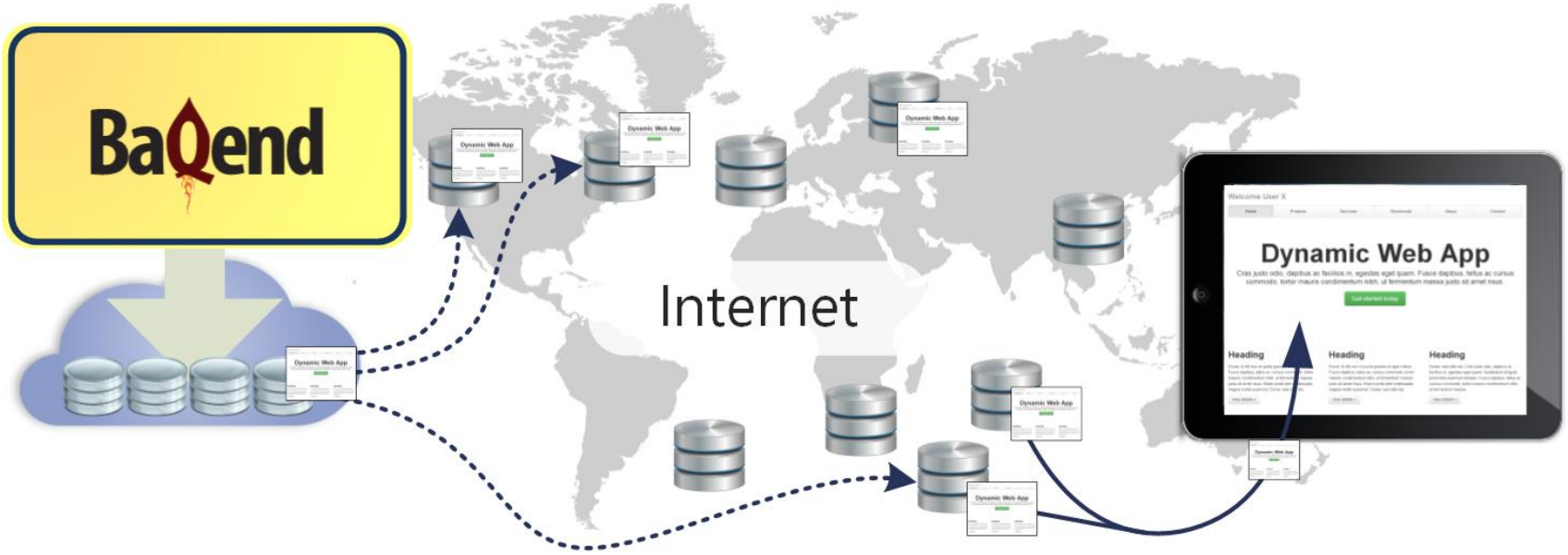
Baqend

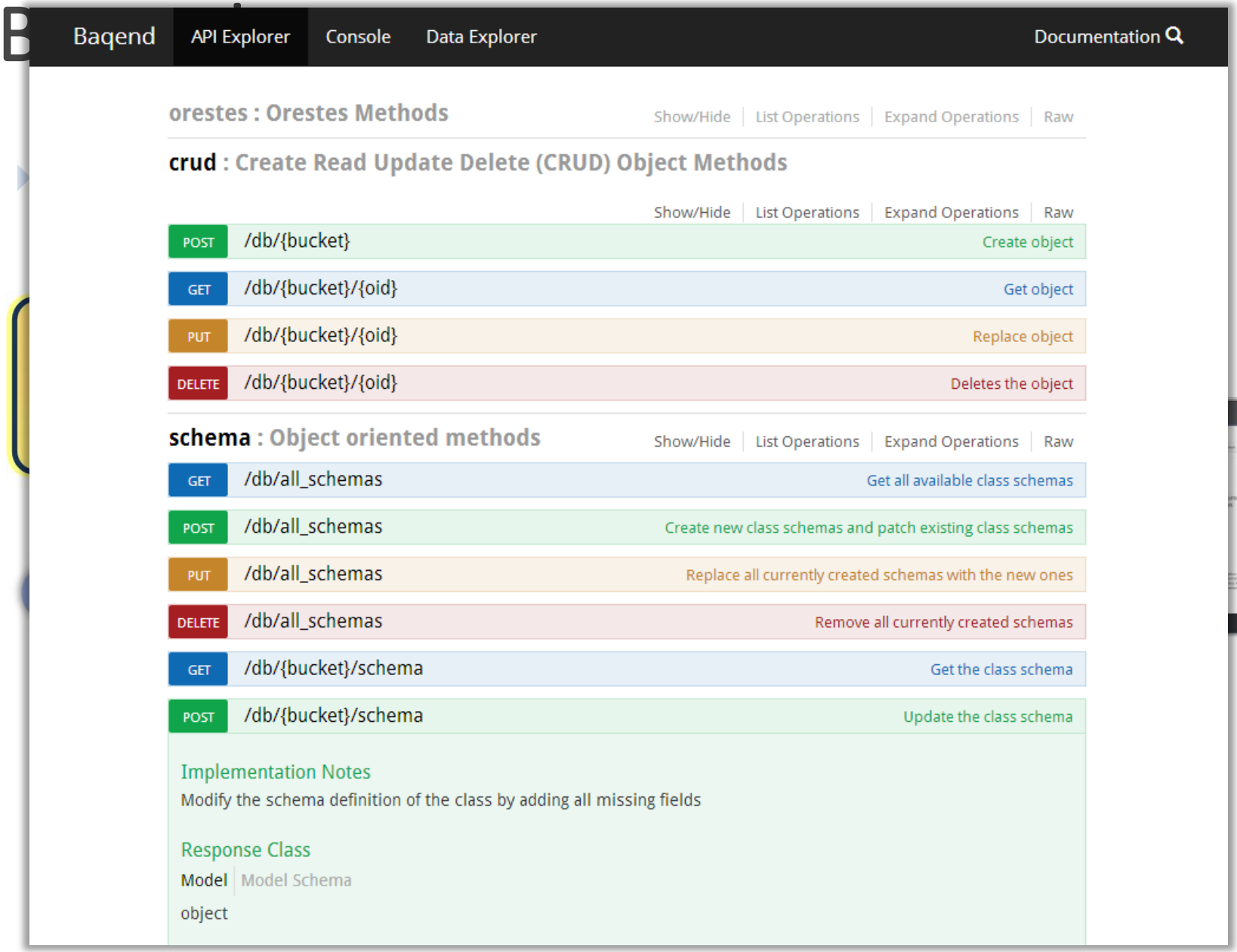
- Orestes as a startup



Baqend

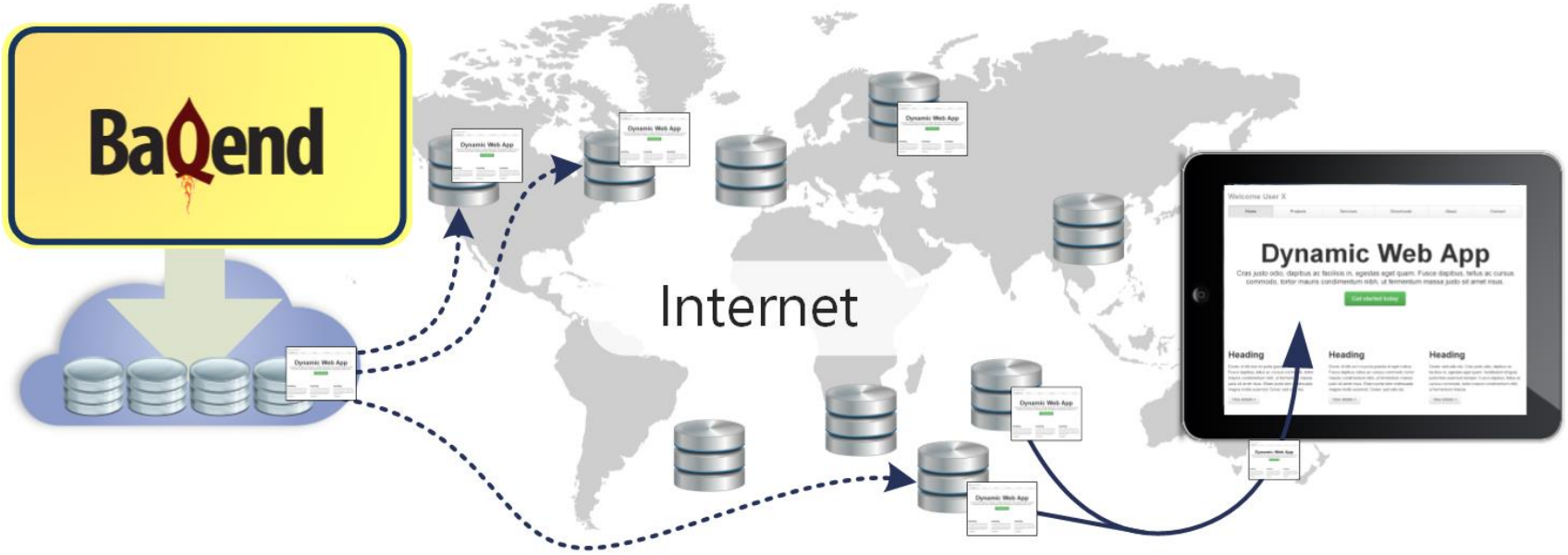
- Orestes as a startup




Baqend
API Explorer
Console
Data Explorer
Documentation
orestes : Orestes Methods
Show/Hide
List Operations
Expand Operations
Raw
crud : Create Read Update Delete (CRUD) Object Methods
Show/Hide
List Operations
Expand Operations
Raw
POST
/db/{bucket}
Create object
GET
/db/{bucket}/{oid}
Get object
PUT
/db/{bucket}/{oid}
Replace object
DELETE
/db/{bucket}/{oid}
Deletes the object
schema : Object oriented methods
Show/Hide
List Operations
Expand Operations
Raw
GET
/db/all_schemas
Get all available class schemas
POST
/db/all_schemas
Create new class schemas and patch existing class schemas
PUT
/db/all_schemas
Replace all currently created schemas with the new ones
DELETE
/db/all_schemas
Remove all currently created schemas
GET
/db/{bucket}/schema
Get the class schema
POST
/db/{bucket}/schema
Update the class schema
Implementation Notes
Modify the schema definition of the class by adding all missing fields
Response Class
Model
Model Schema
object

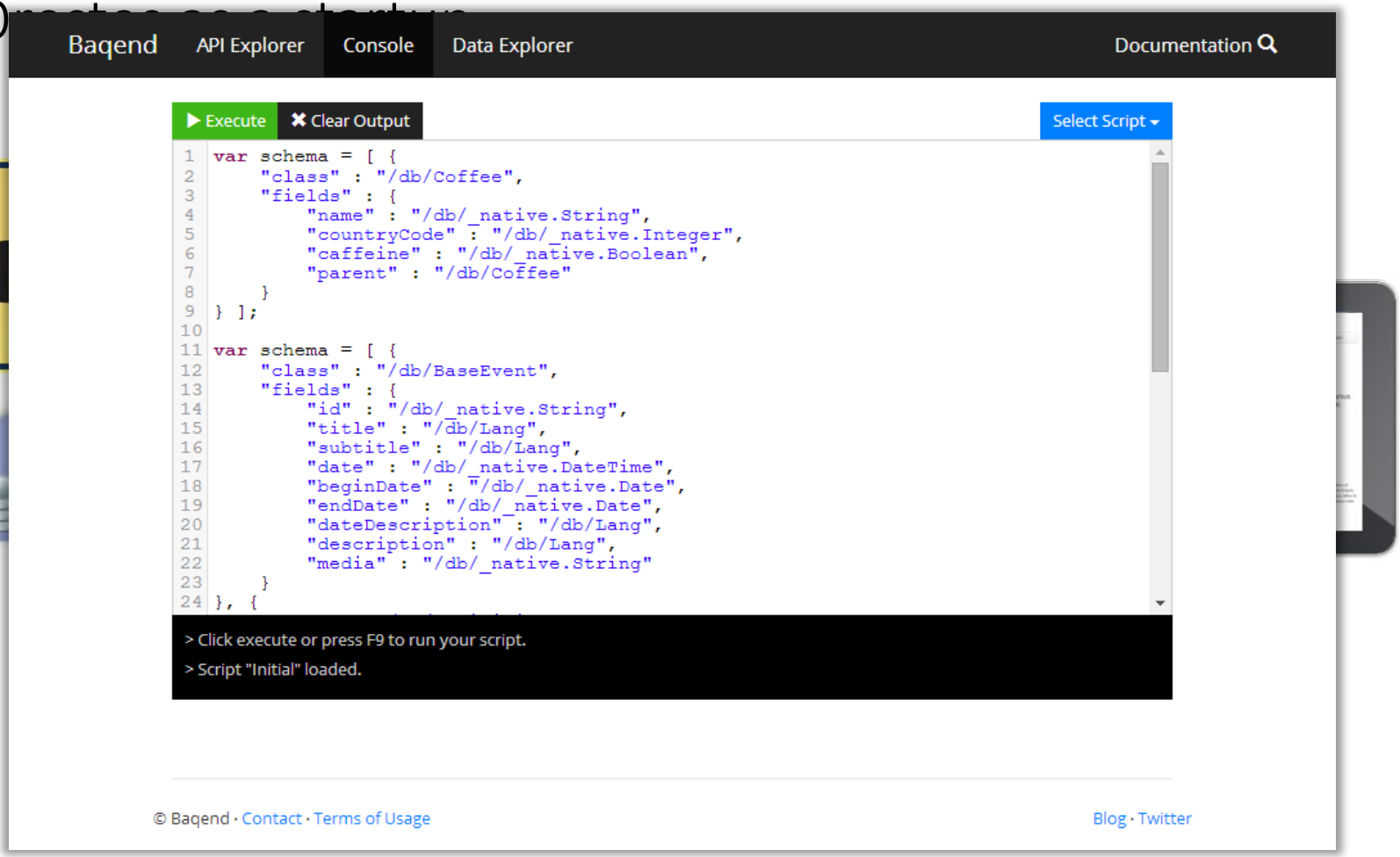
Baqend

- Orestes as a startup



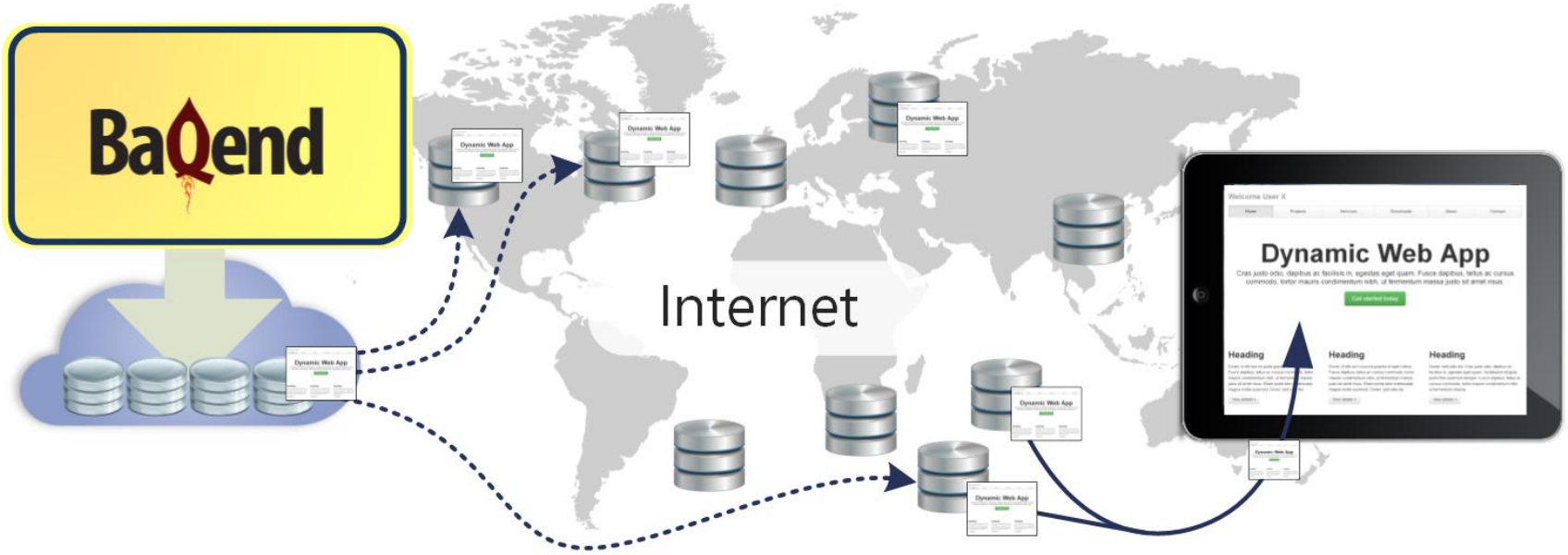
Baqend

- Oracle as a startup



Baqend

- Orestes as a startup



Baqend in Action

Baqend in Action

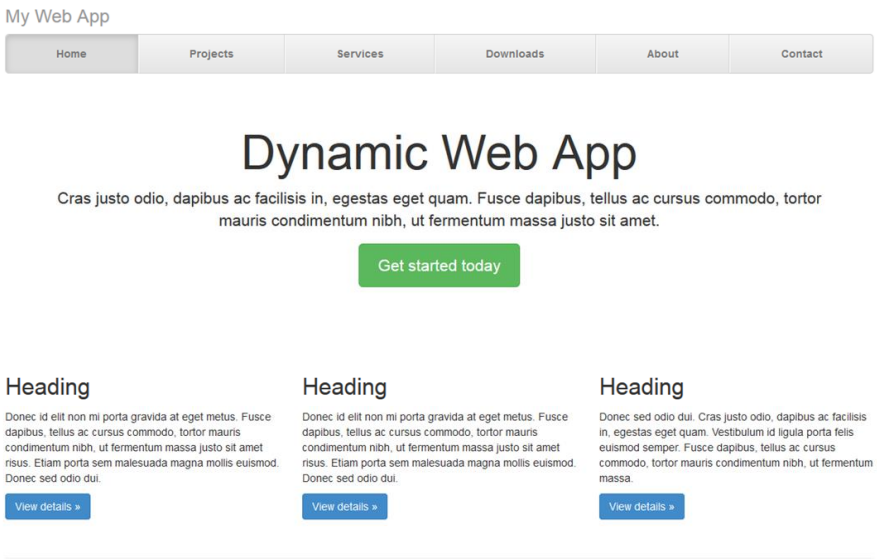
GET /app.html

My Web App

Baqend in Action

GET /app.html

```
db.find(Menu, 'main')
  .done(...);
db.find(Page, 'hero')
  .done(...);
db.query(Page, 'top3')
  .done(...);
```

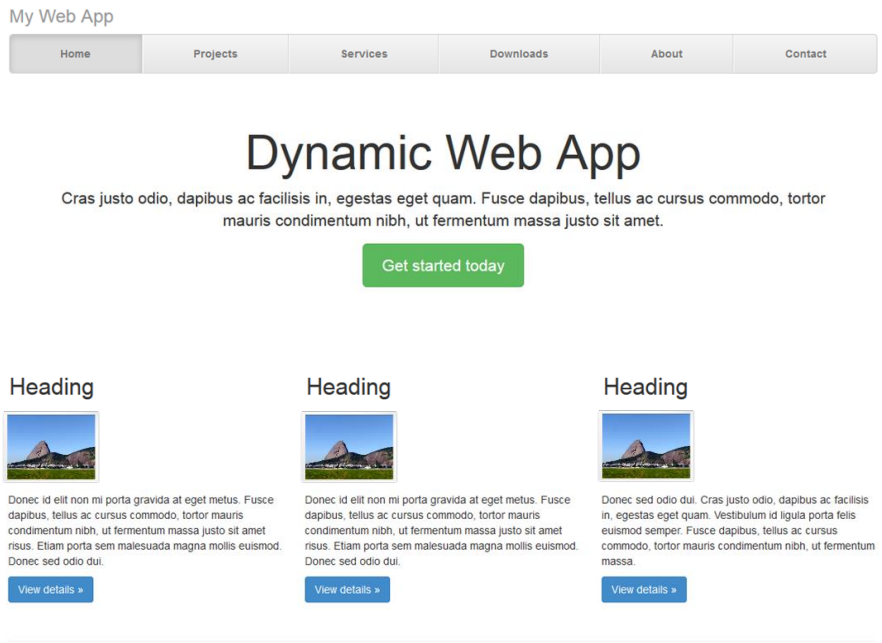


Baqend in Action

GET /app.html

```
db.find(Menu, 'main')
  .done(...);
db.find(Page, 'hero')
  .done(...);
db.query(Page, 'top3')
  .done(...);
```

GET /img/pic005.jpg
GET /img/pic017.jpg
GET /img/pic022.jpg



More: **Baqend.com**

Wrap-up & book recommendations

Wrap-up

How to choose a cloud database:

Managed RDBMS	Managed DWH	Managed NoSQL DB	Backend-as-a-Service	Proprietary Serivce	Object Store

Wrap-up

How to choose a cloud database:

Define your functional requirements

Define your non-functional requirements

- Managed RDBMS
- Managed DWH
- Managed NoSQL DB
- Backend-as-a-Service
- Proprietary Serivce
- Object Store

Wrap-up

How to choose a cloud database:

Define your functional requirements

Define your non-functional requirements

Evaluate by:

1. **Underyling DB**
2. **Docs & books**
3. **Your own tests**

Managed RDBMS	Managed DWH	Managed NoSQL DB	Backend-as-a-Service	Proprietary Serivce	Object Store

Wrap-up

How to choose a cloud database:

Define your functional requirements

Define your non-functional requirements

Evaluate by:

1. Underyling DB
2. Docs & books
3. Your own tests
4. SLAs
5. Docs & books
6. Experience Reports

Managed RDBMS	Managed DWH	Managed NoSQL DB	Backend-as-a-Service	Proprietary Serivce	Object Store

Wrap-up

How to choose a cloud database:

Define your functional requirements

Define your non-functional requirements

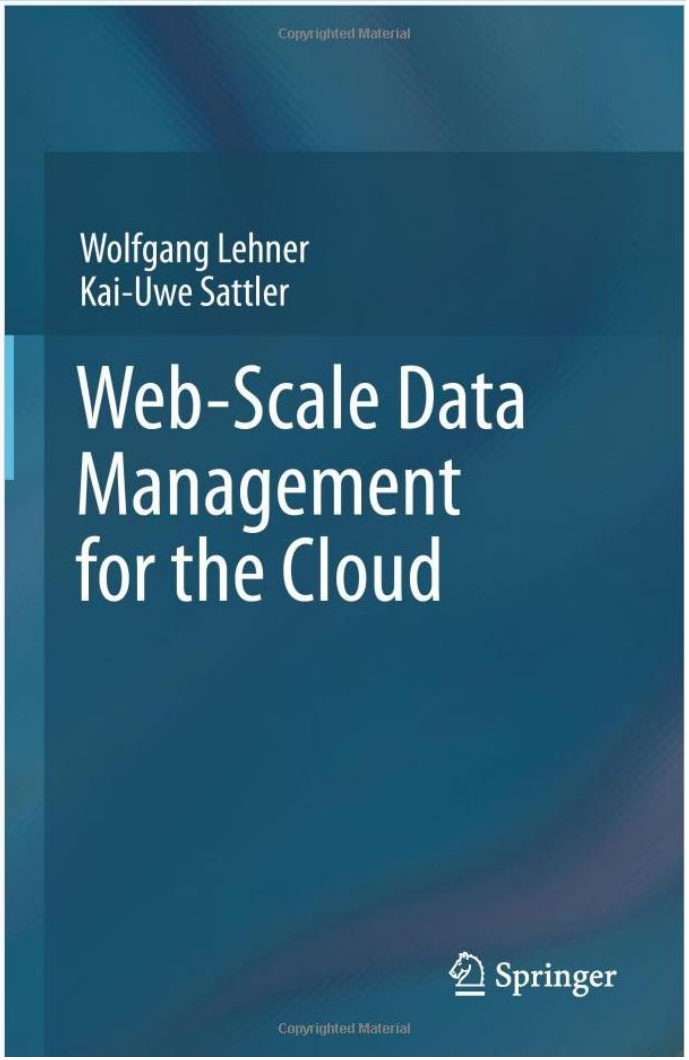
Evaluate by:

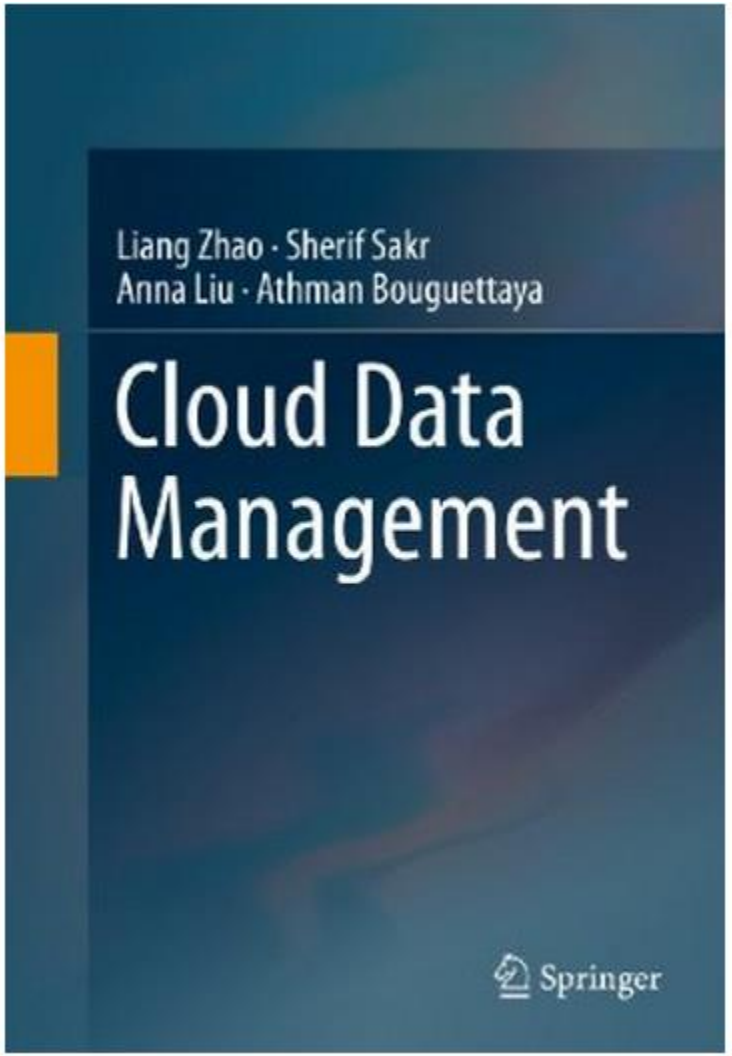
1. Underyling DB
2. Docs & books
3. Your own tests
4. SLAs
5. Docs & books
6. Experience Reports

Try it

Managed RDBMS	Managed DWH	Managed NoSQL DB	Backend-as-a-Service	Proprietary Serivce	Object Store

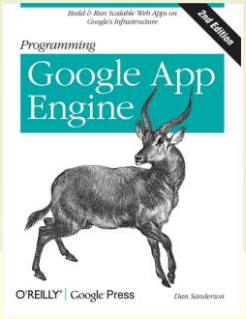
Book recommendations

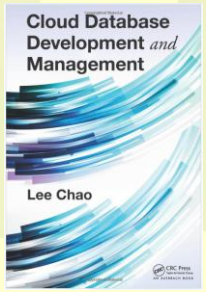


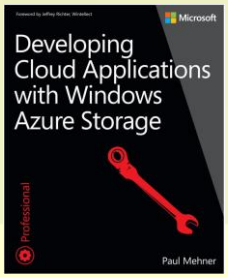


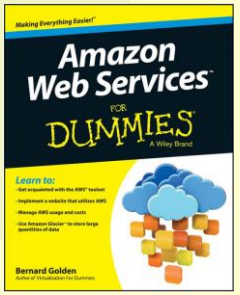
Book recommendations

- (Non-scientific) literature is **rare**
- Some books cover specific DBaaS and cloud platforms









Blogs

myNoSQL

http://nosql.mypopescu.com/

NoSQL Weekly

http://www.nosqlweekly.com/

http://www.dzone.com/mz/nosql

Hacking, Distributed

http://hackingdistributed.com/

DBMS2

http://www.dbms2.com/

High Scalability

http://highscalability.com/

Top Scientific Conferences

VLDB (Very Large Databases)

SIGMOD (Special Interest Group on Management of Data)

ICDE (International Conference on Data Engineering)

CIDR (Conference on Innovative Data Systems Research)

Database Research

SOCC (Symposium on Cloud Computing)

OSDI/SOSP (Operating Systems Design and Implementation/ Symposium on Operating System Principles)

EuroSys

Distributed Systems Research

Top Scientific Conferences

OSDI/SOSP (Operating Systems Design and Symposium on Operating System Principles)

SCDM 2013

EuroSys

This year probably in Washington D.C.

Learn more: scdm2013.com

Thank you. Queries?

Contact:

felix.gessert@baqend.com

http://baqend.com

http://orestes.info

http://scdm2013.com