



Cloud Databases

in Research and Practice

Felix Gessert
(29.04.2014)

Slides: bagend.com/nosql.pdf

About me

- ▶ PhD student (database group university of hamburg)

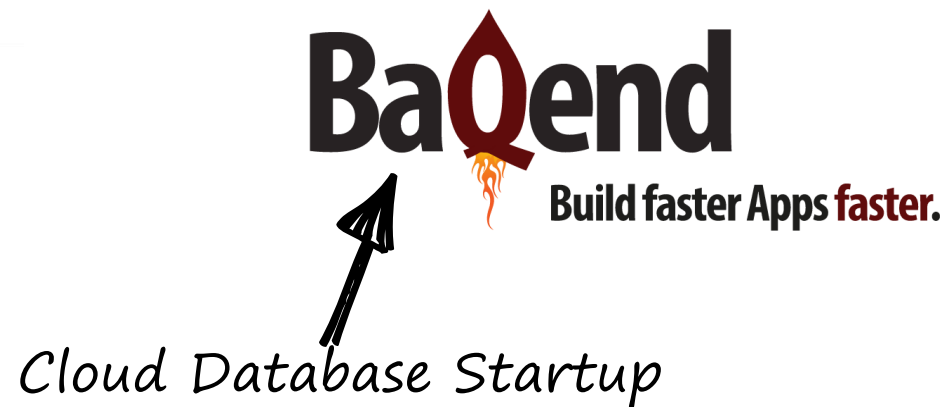
About me

- ▶ 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

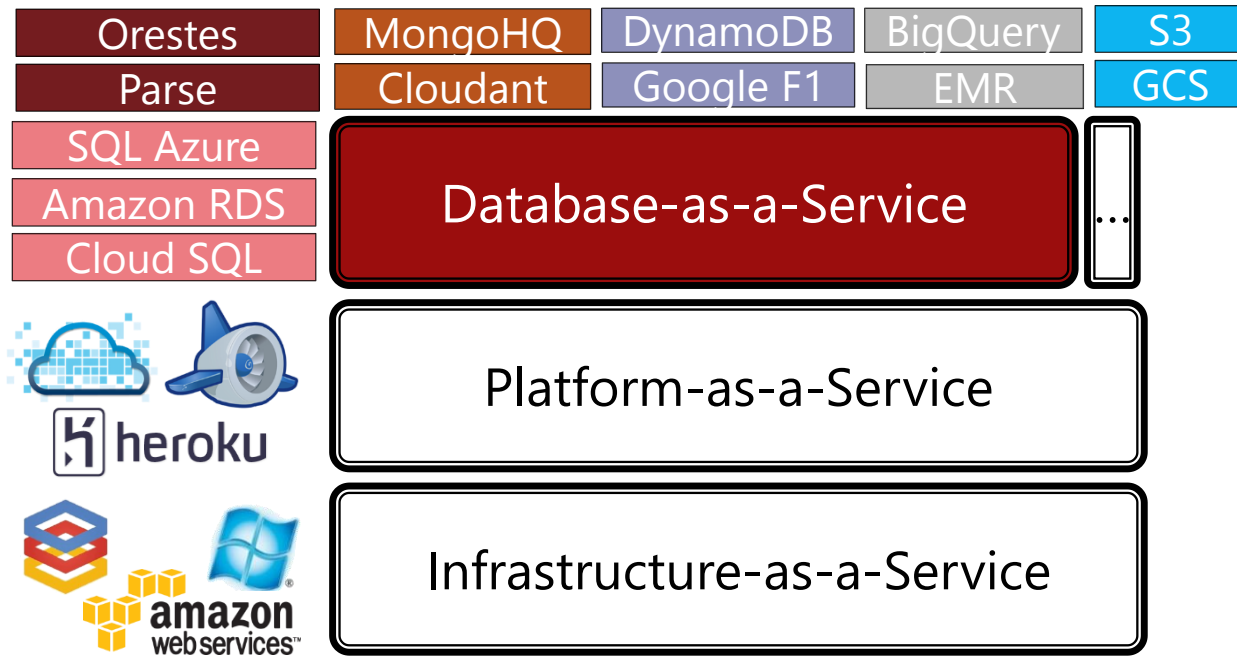
2012: DynamoDB (Amazon)

2013: Redshift (Amazon)

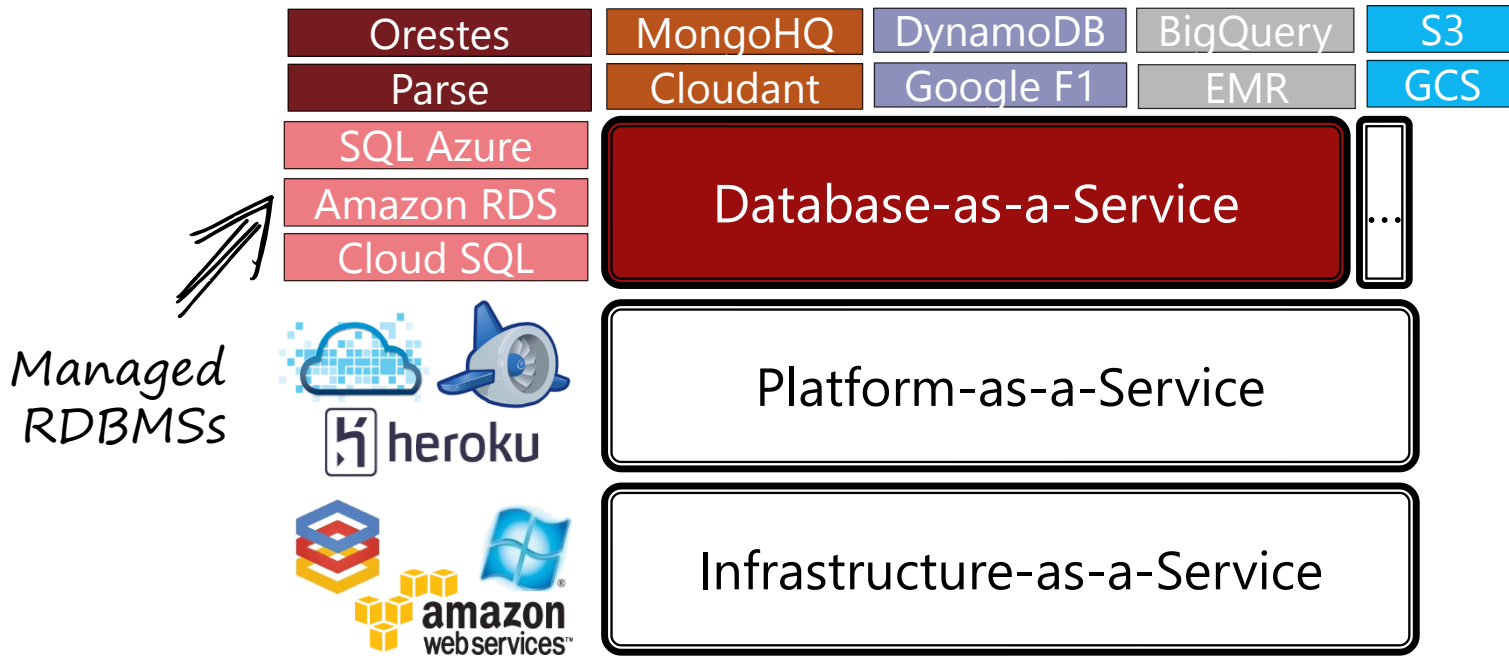


Context: What kinds of Cloud databases are there?

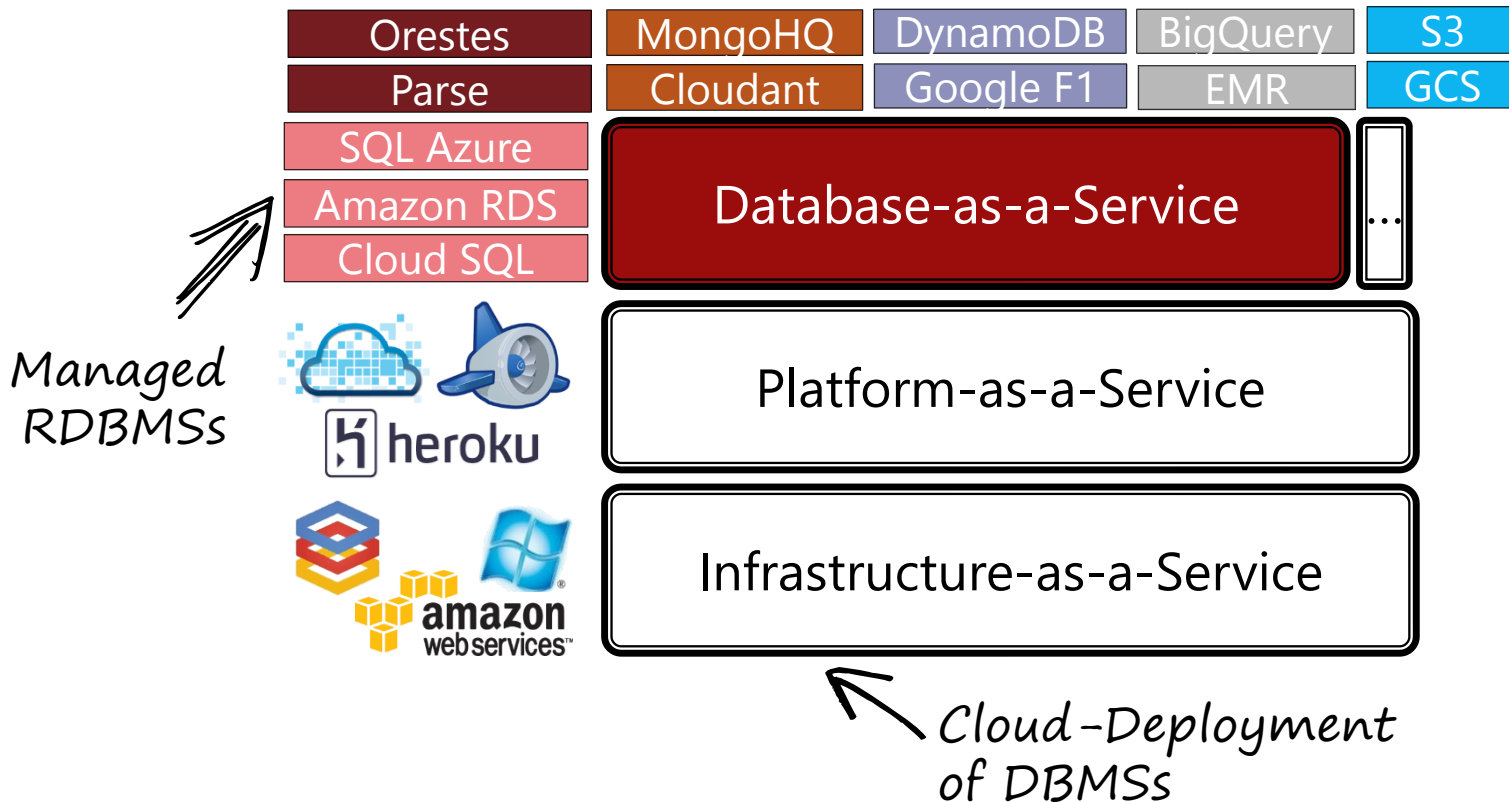
DBaaS



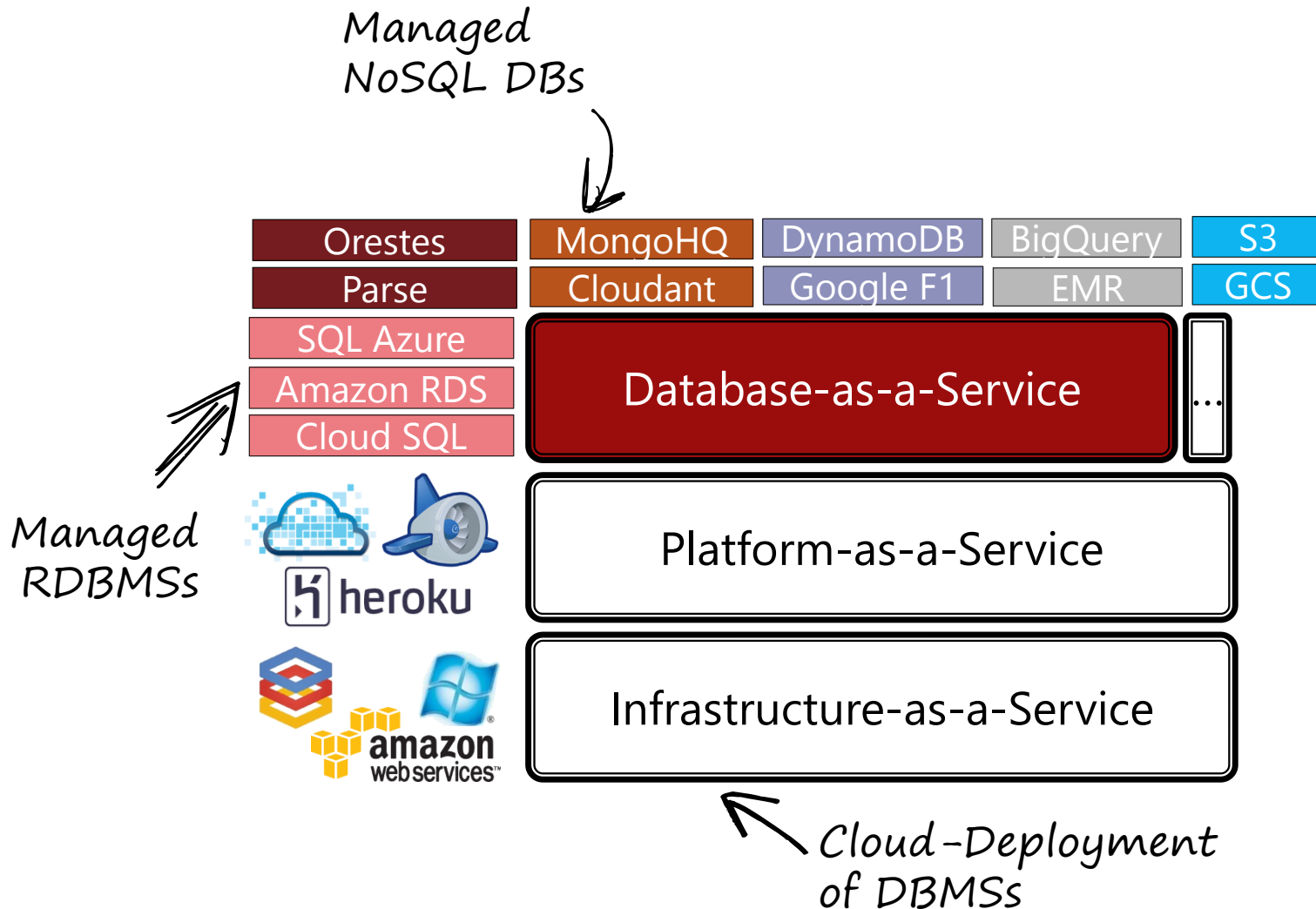
DBaaS



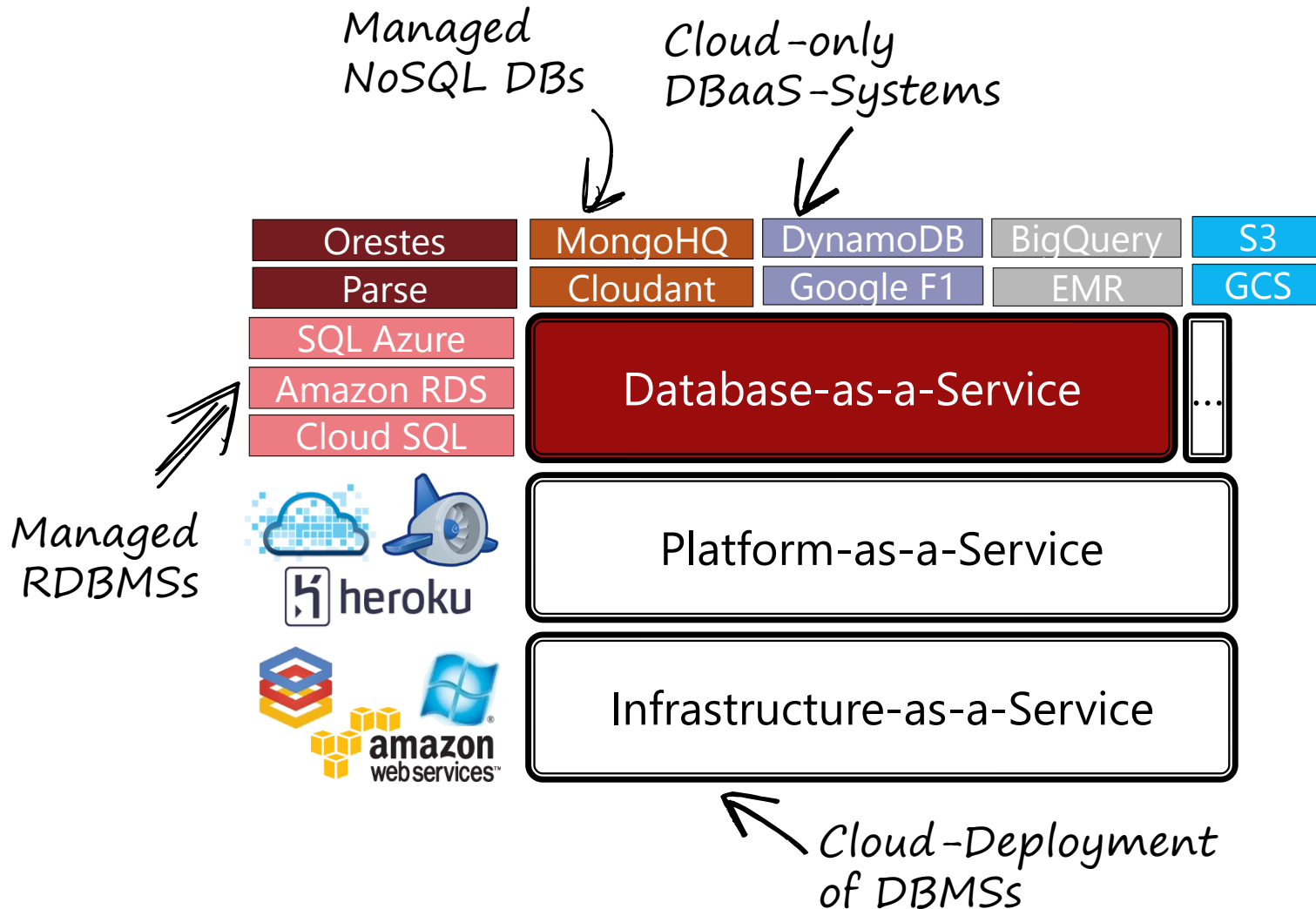
DBaaS



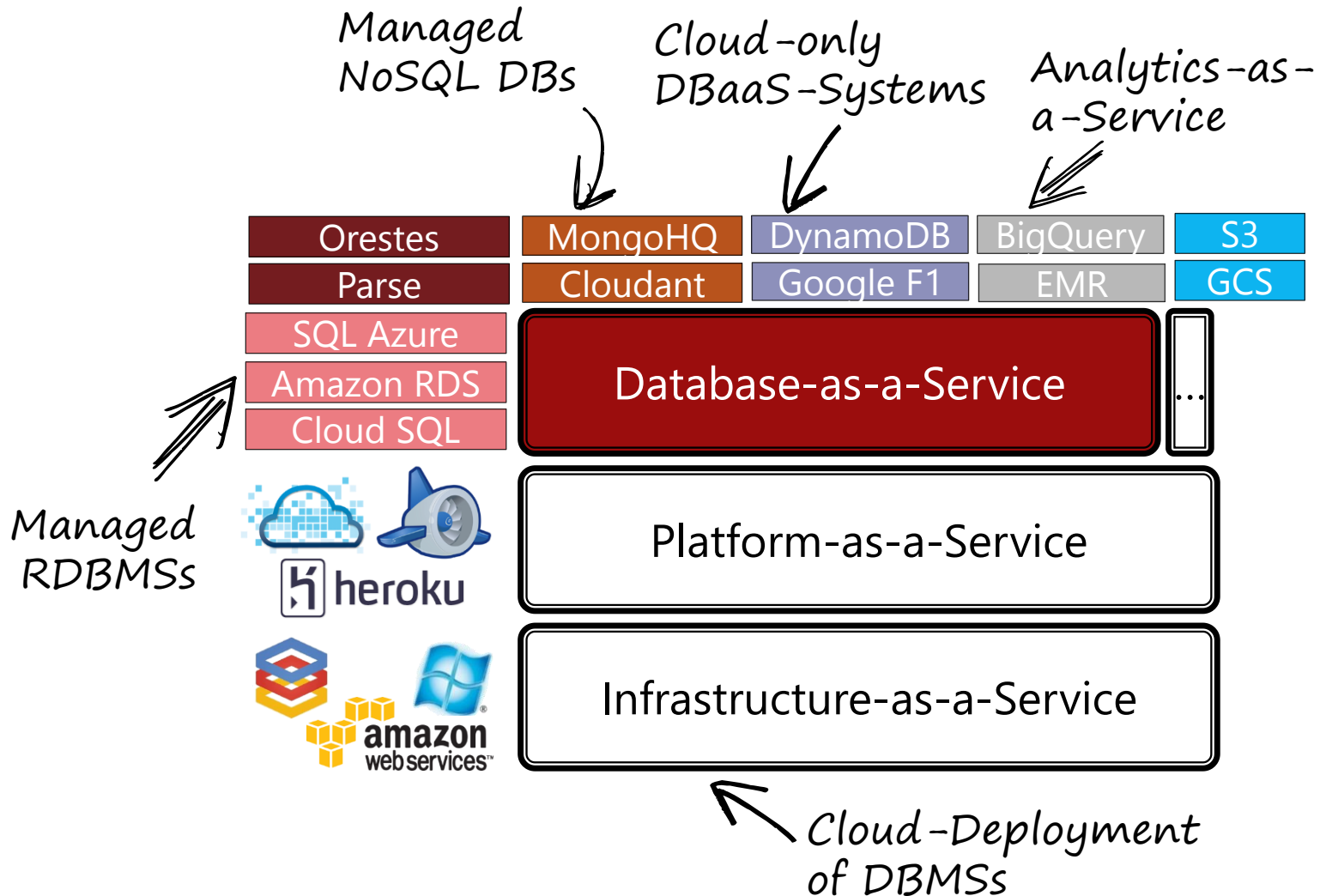
DBaaS



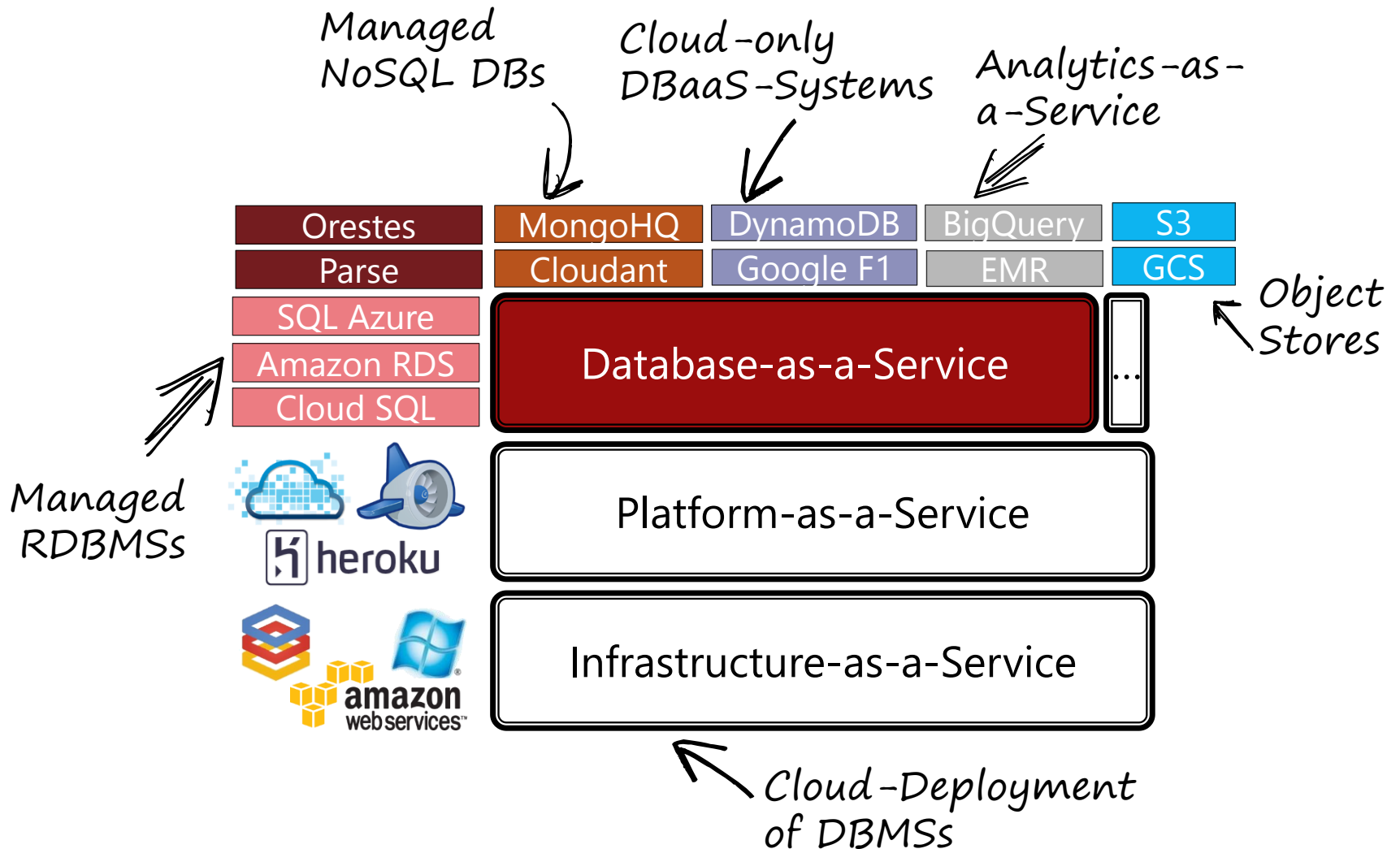
DBaaS



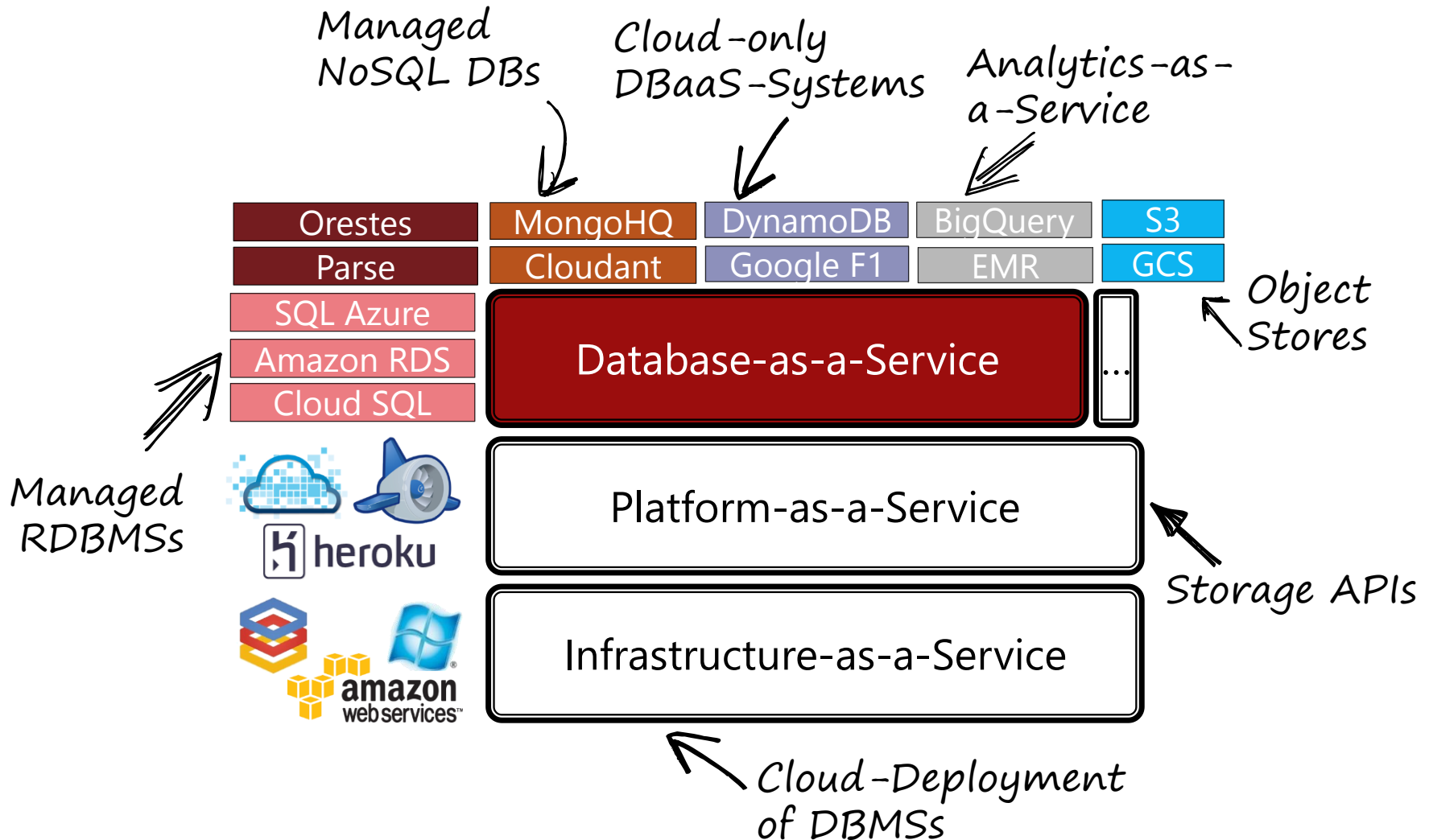
DBaaS



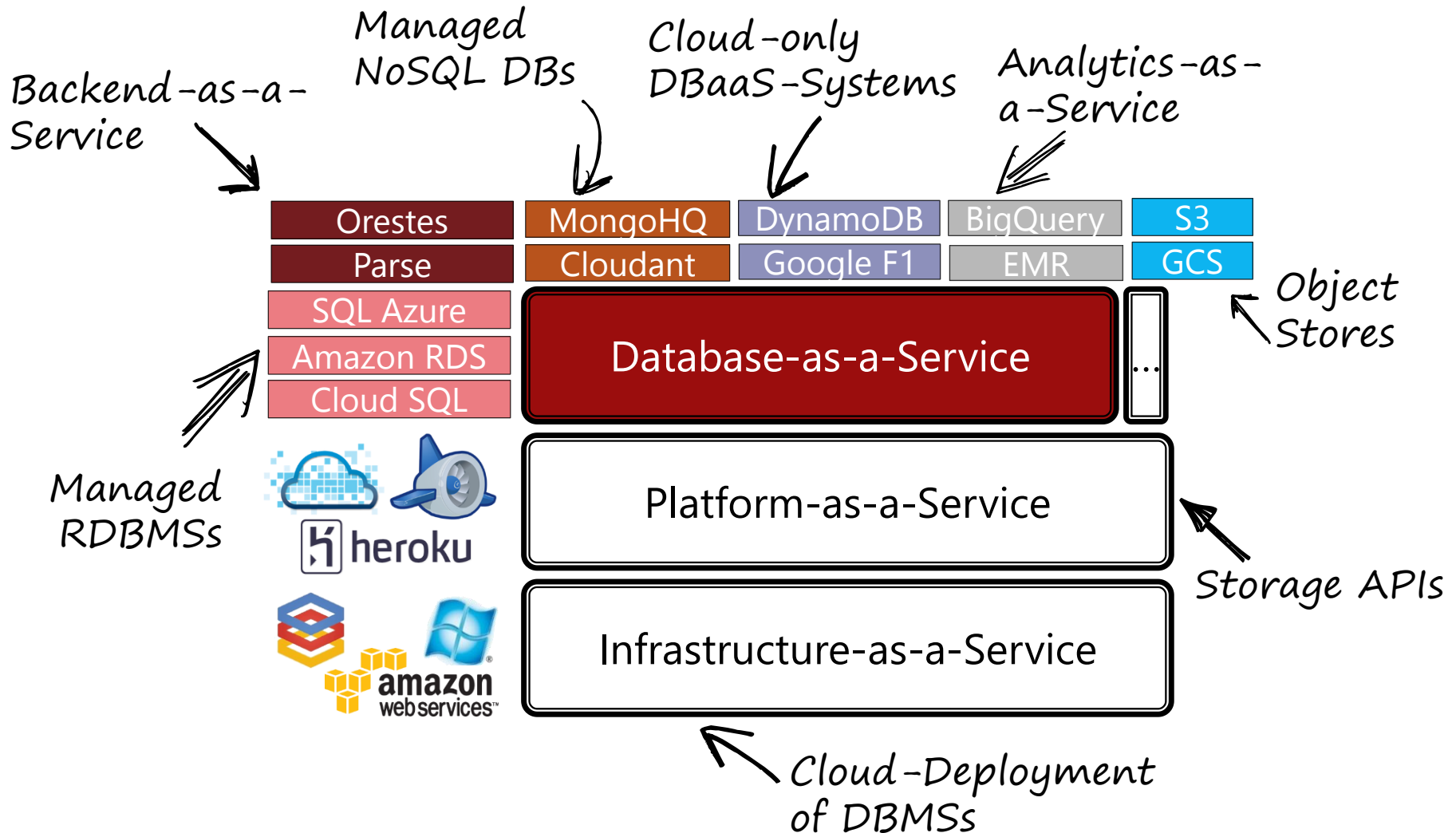
DBaaS



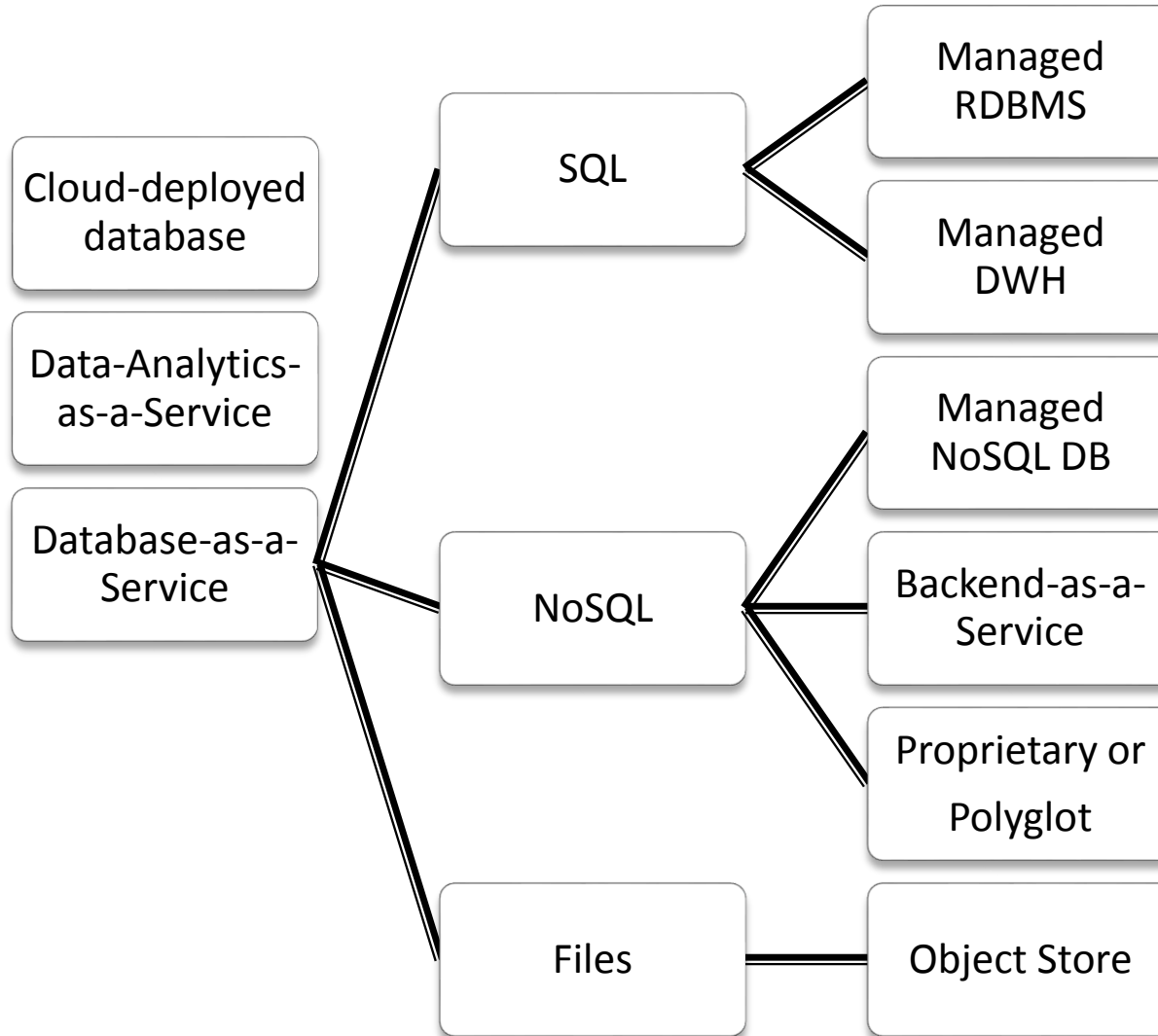
DBaaS



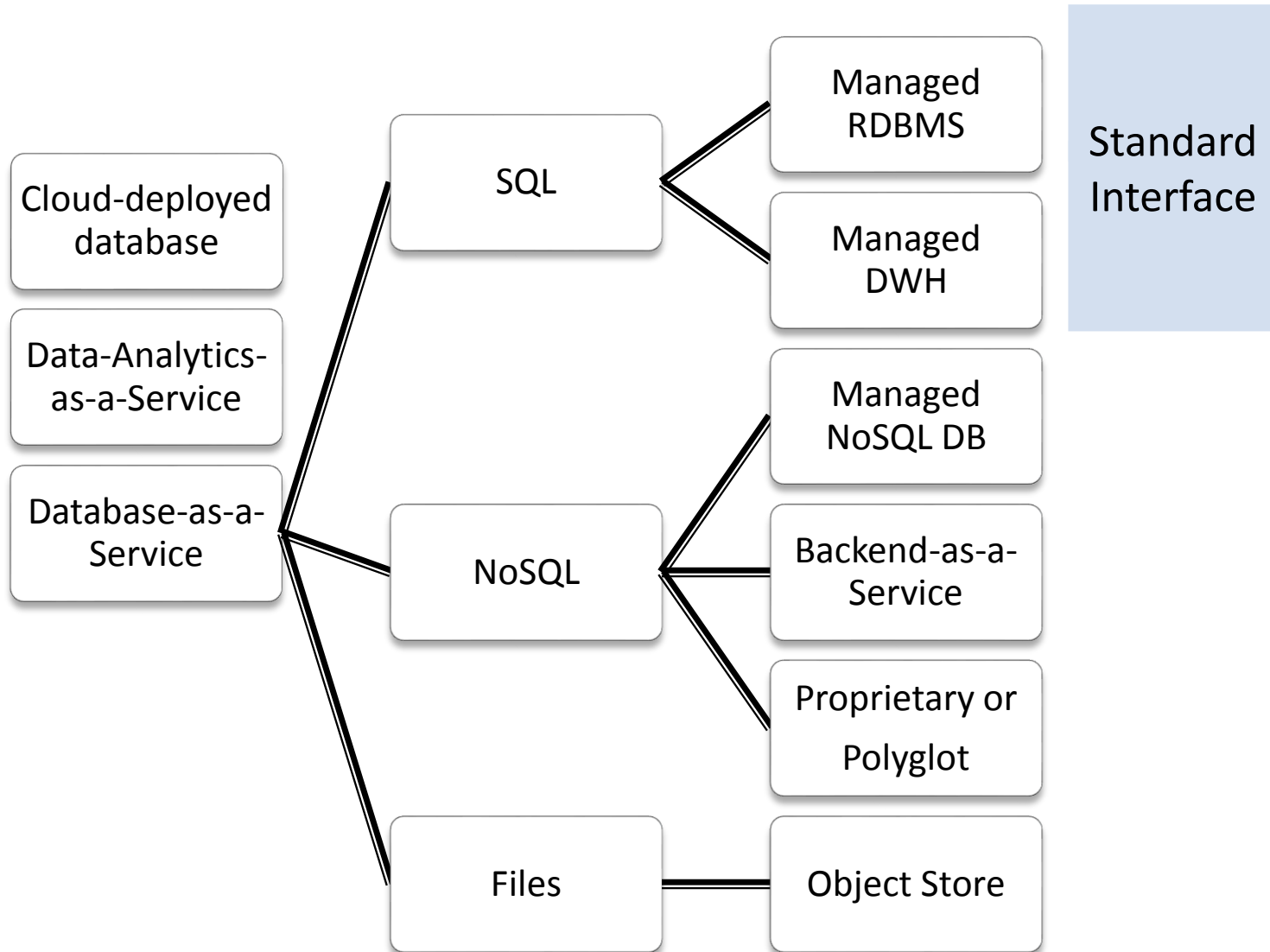
DBaaS



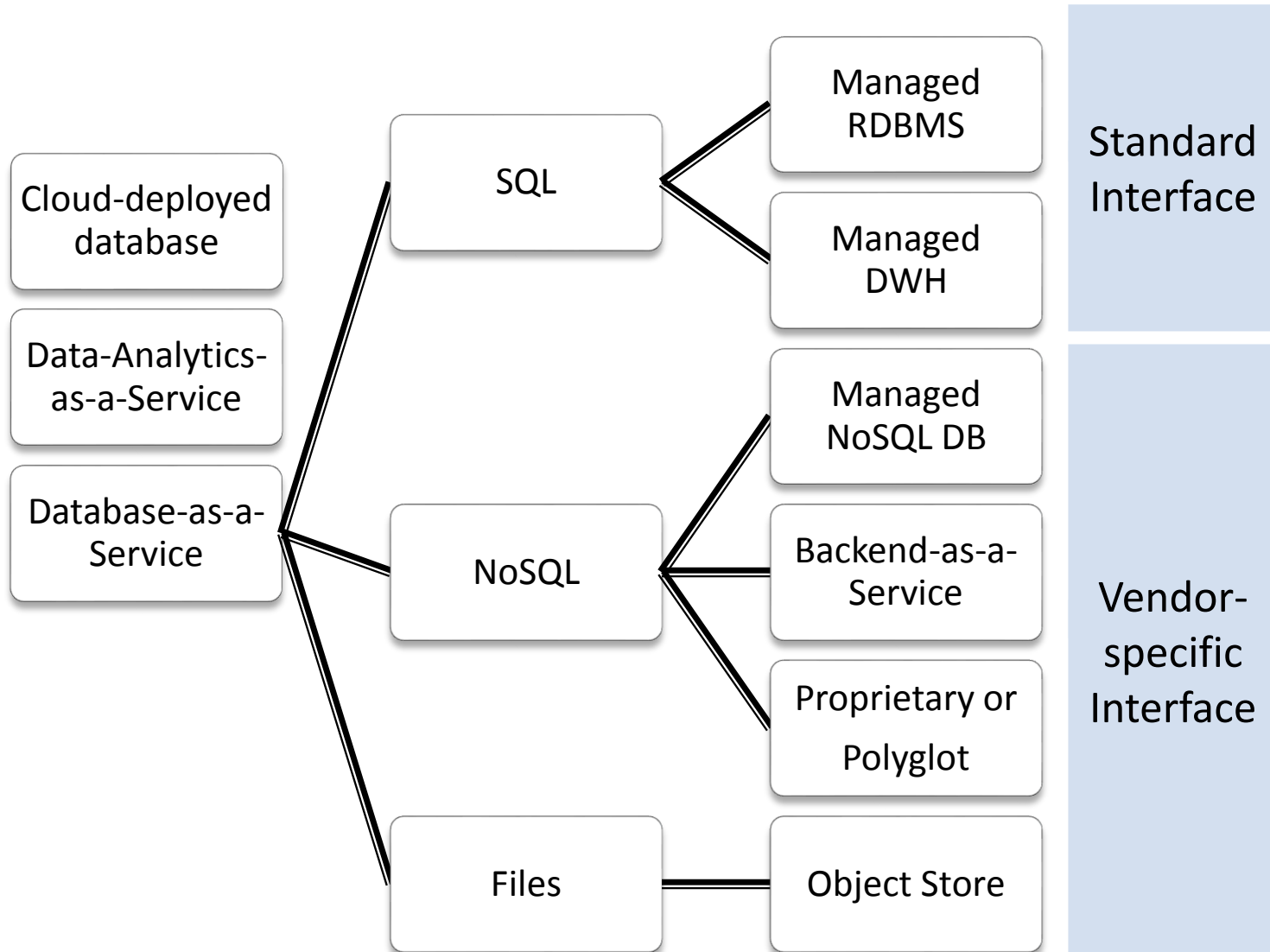
Cloud Databases



Cloud Databases

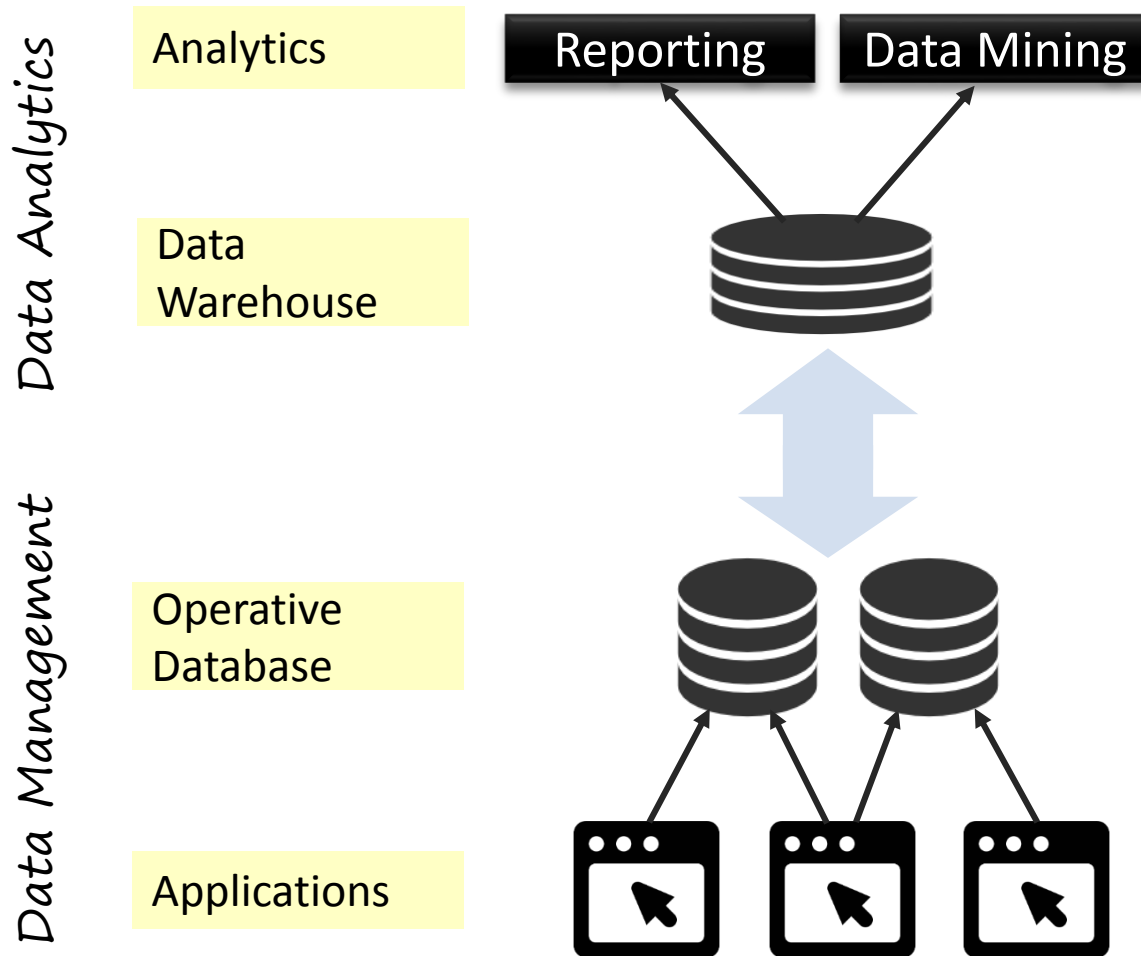


Cloud Databases



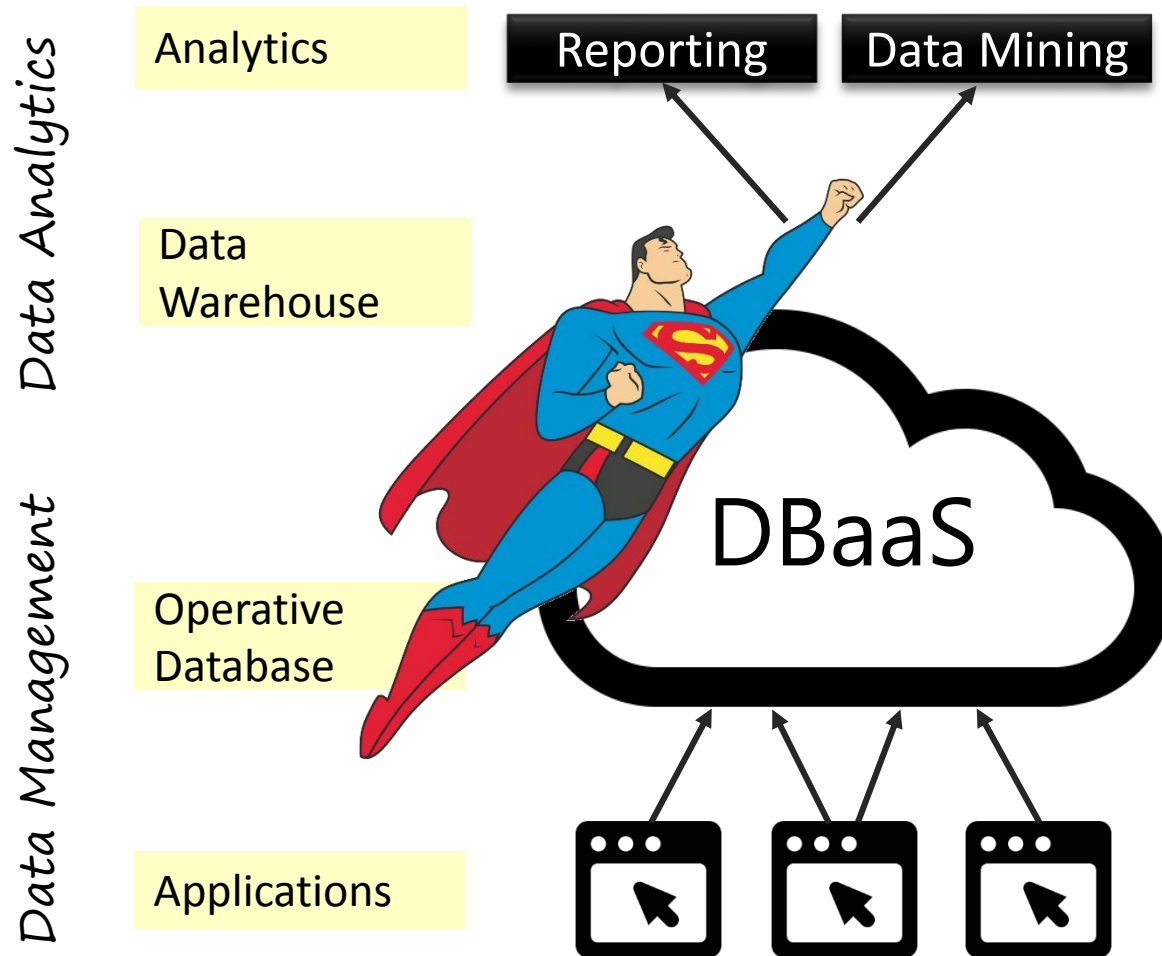
Architectures

Application Architecture <-> Cloud Database Category



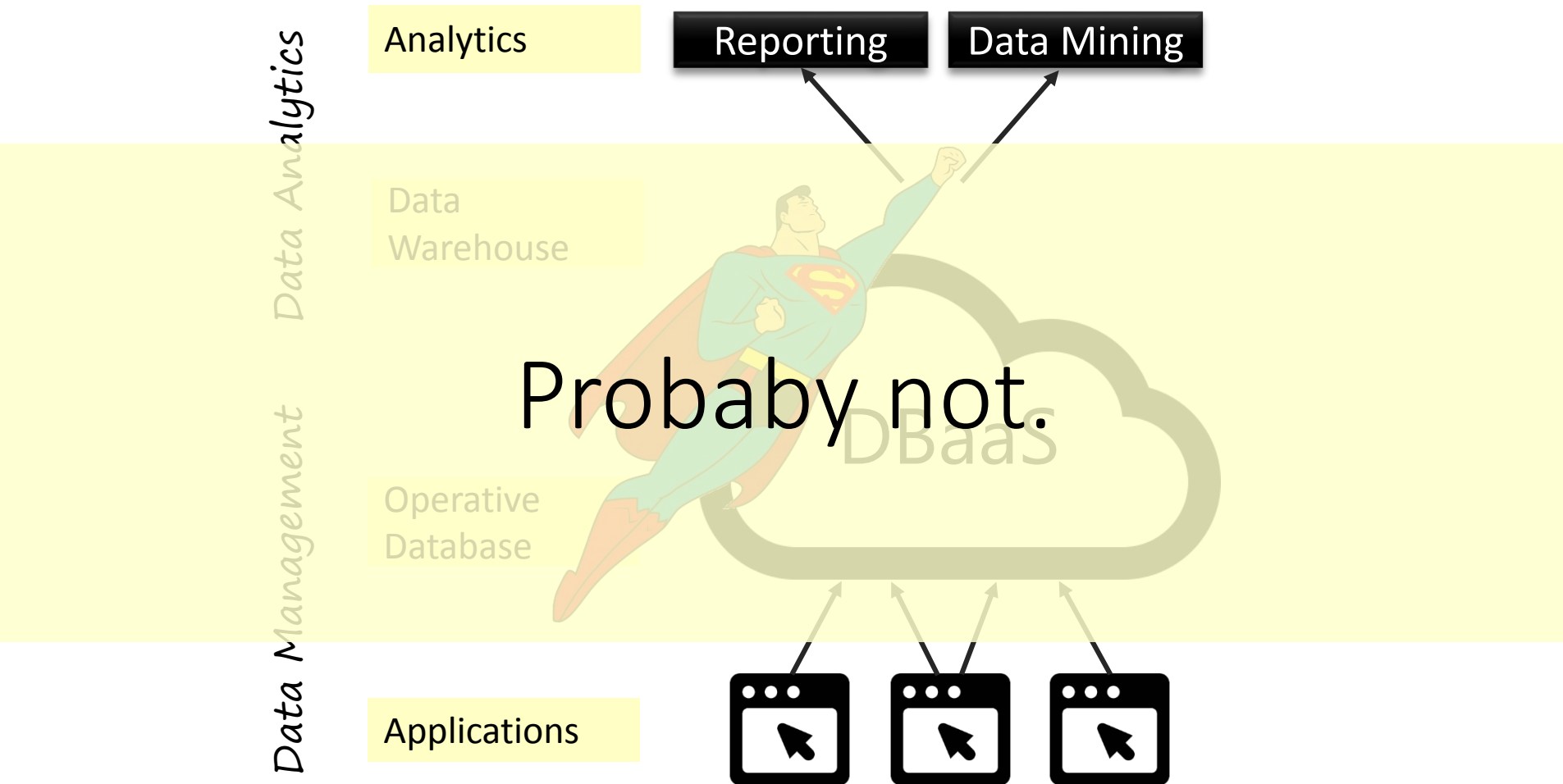
Architectures

Application Architecture <-> Cloud Database Category

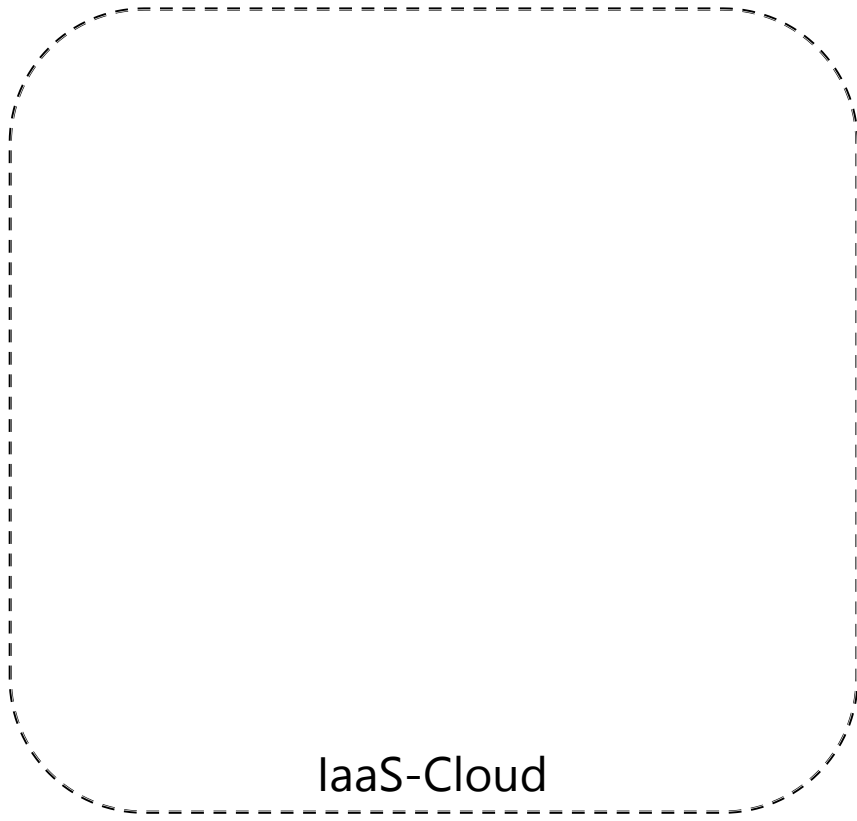


Architectures

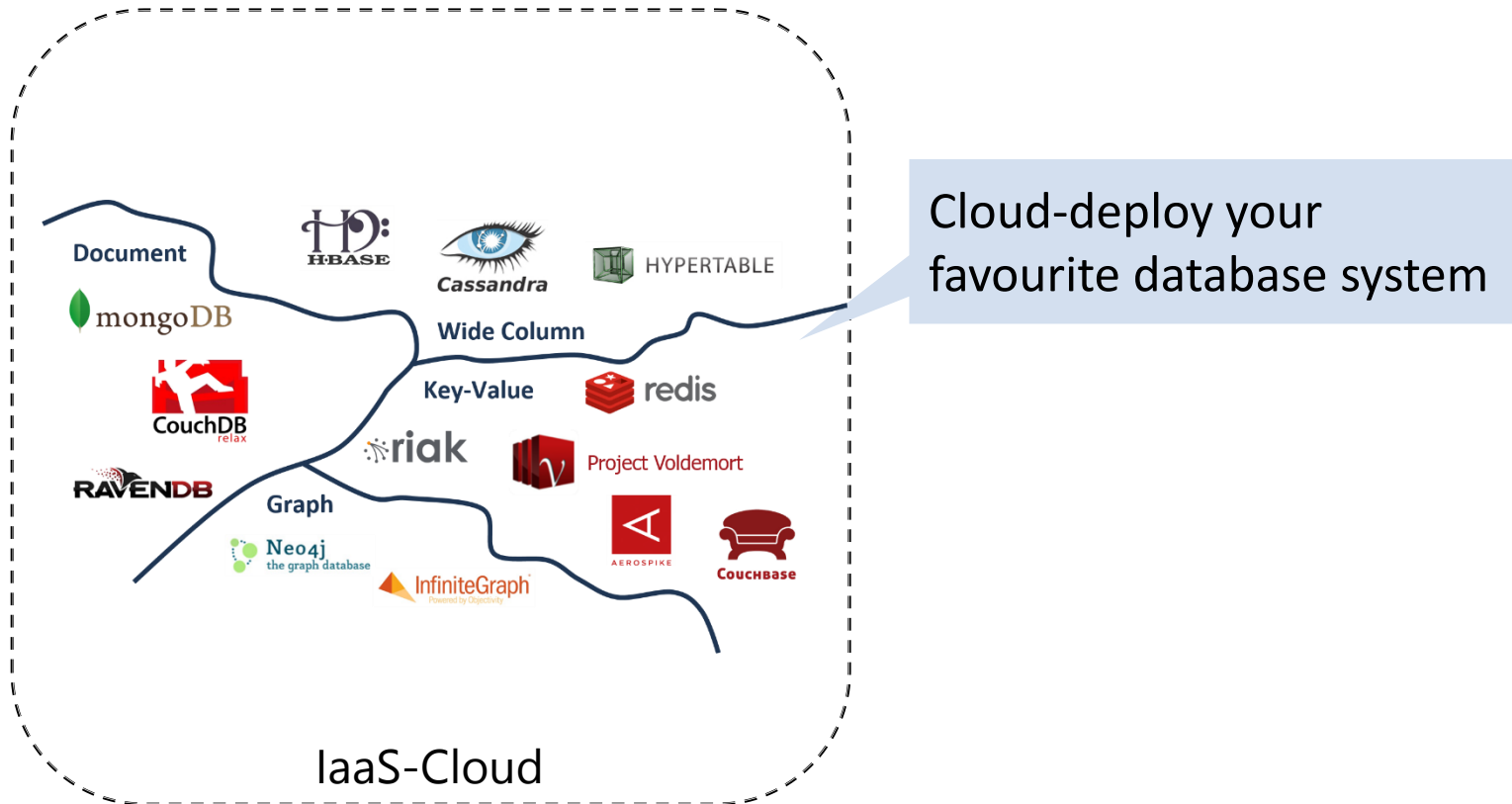
Application Architecture <-> Cloud Database Category



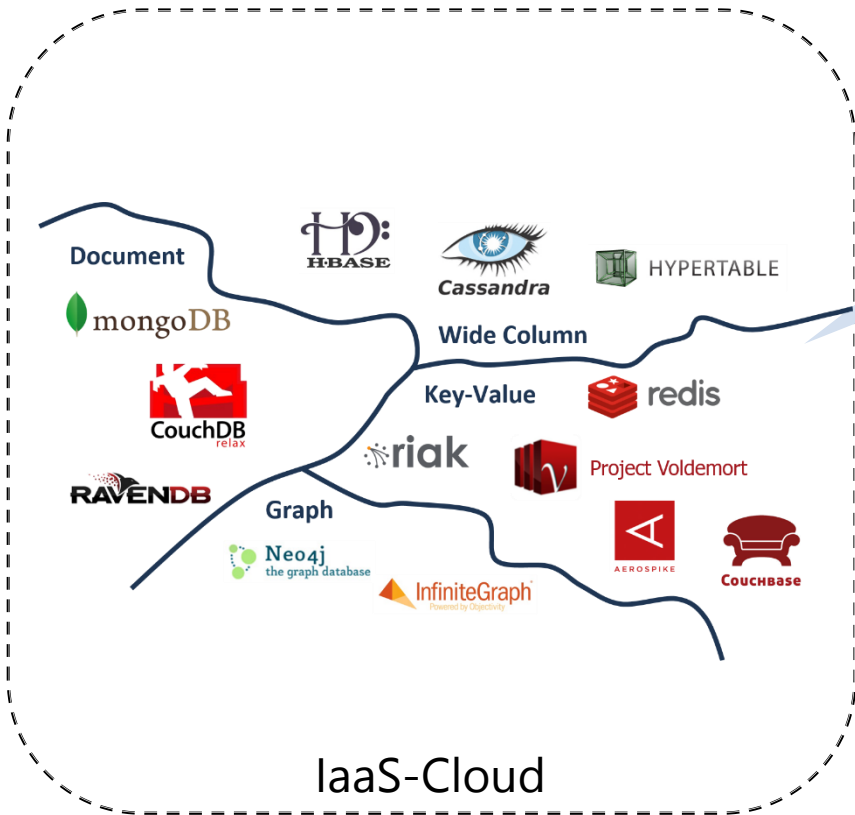
Cloud-Deployed Database



Cloud-Deployed Database



Cloud-Deployed Database

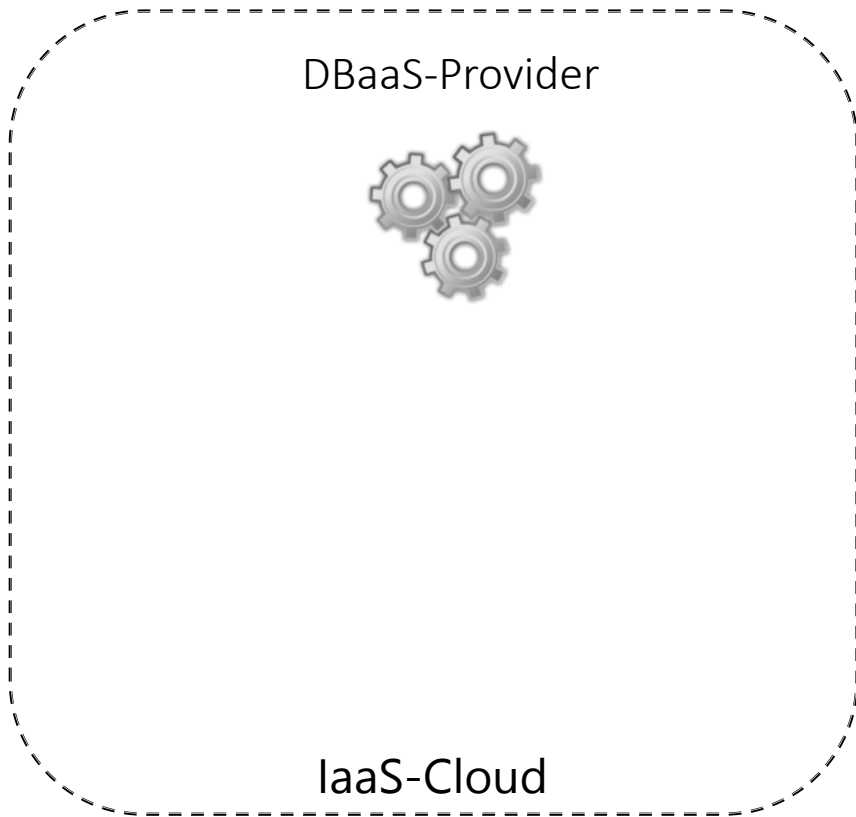


Cloud-deploy your favourite database system

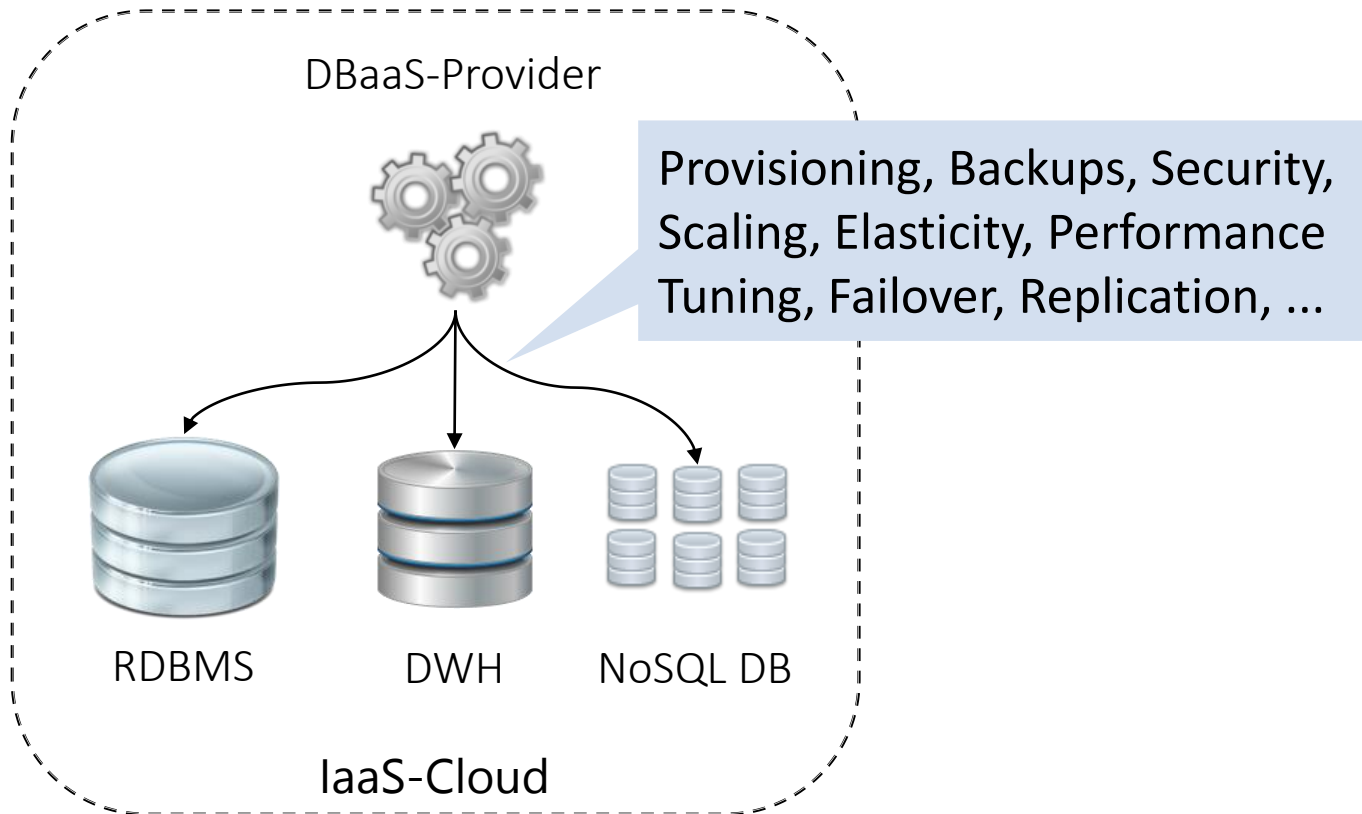
Does not solve:

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

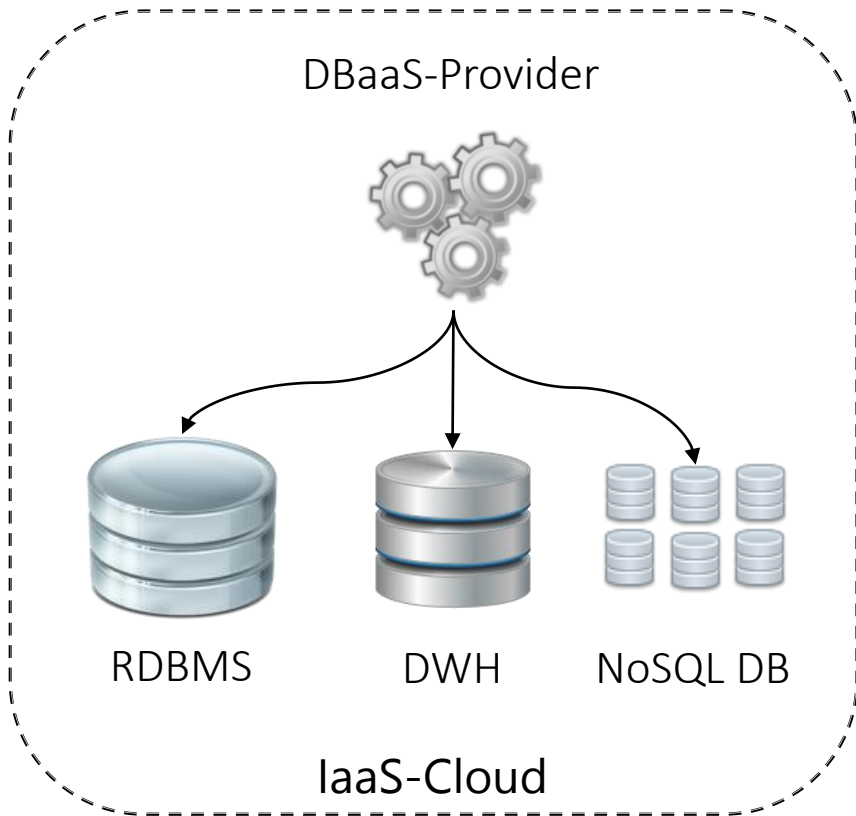
Managed RDBMS/DWH/NoSQL DB



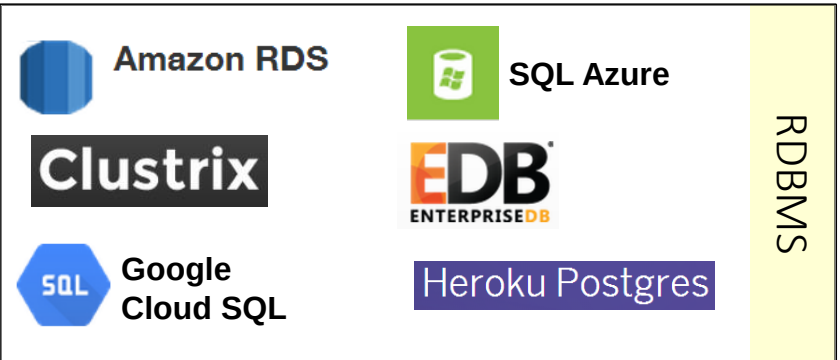
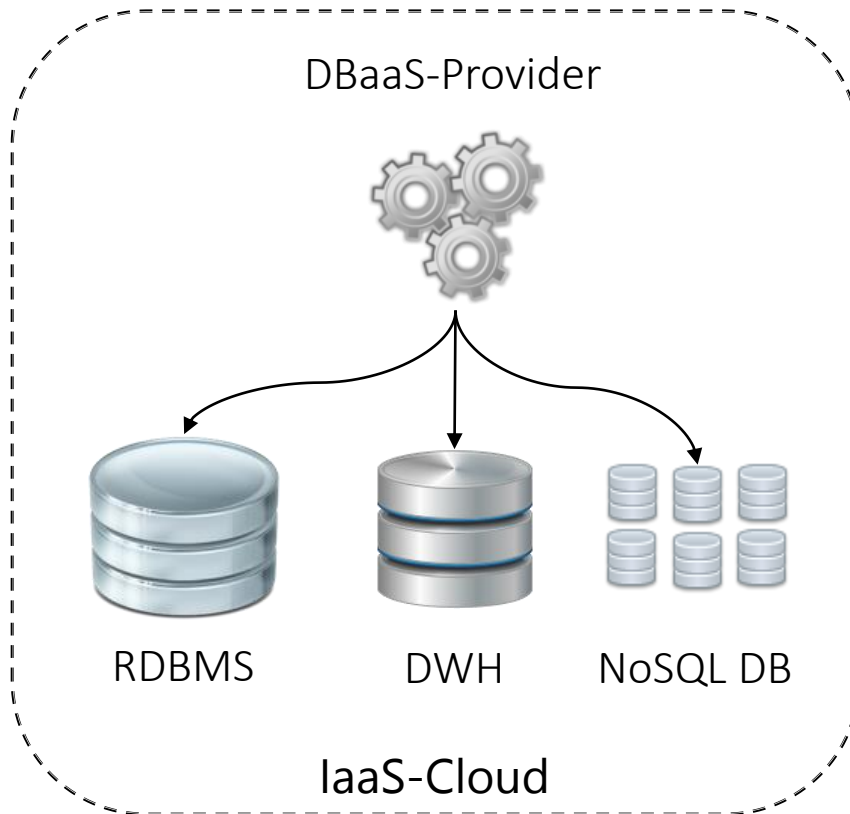
Managed RDBMS/DWH/NoSQL DB



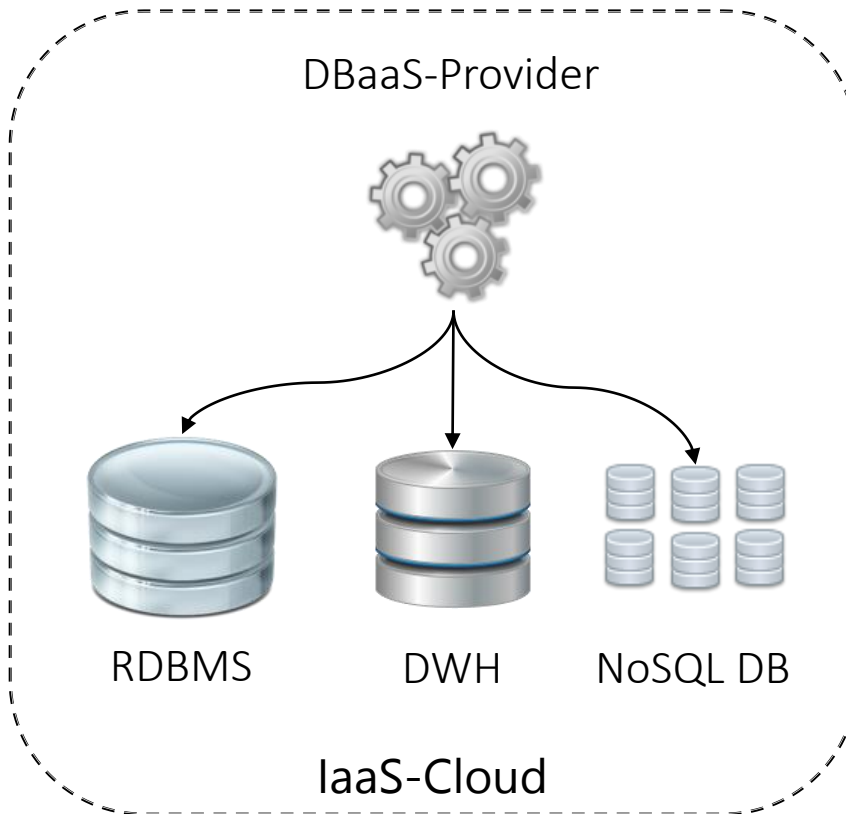
Managed RDBMS/DWH/NoSQL DB



Managed RDBMS/DWH/NoSQL DB

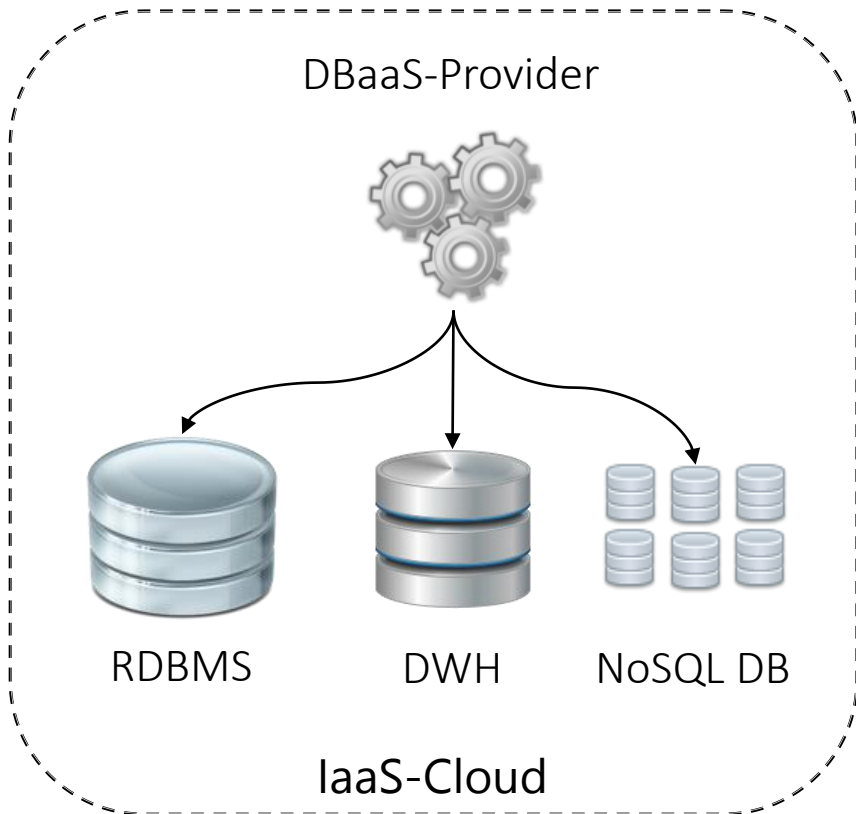


Managed RDBMS/DWH/NoSQL DB



 Amazon RDS	 SQL Azure	RDBMS
 Clustrix	 EDB ENTERPRISEDB	
 Google Cloud SQL	 Heroku Postgres	
 Amazon ElastiCache	 mongoHQ	NoSQL DB
 mongoLab	 Cloudant an IBM® Company	
 redis	 instaclustr	
 bonsai	 Iris Couch	

Managed RDBMS/DWH/NoSQL DB

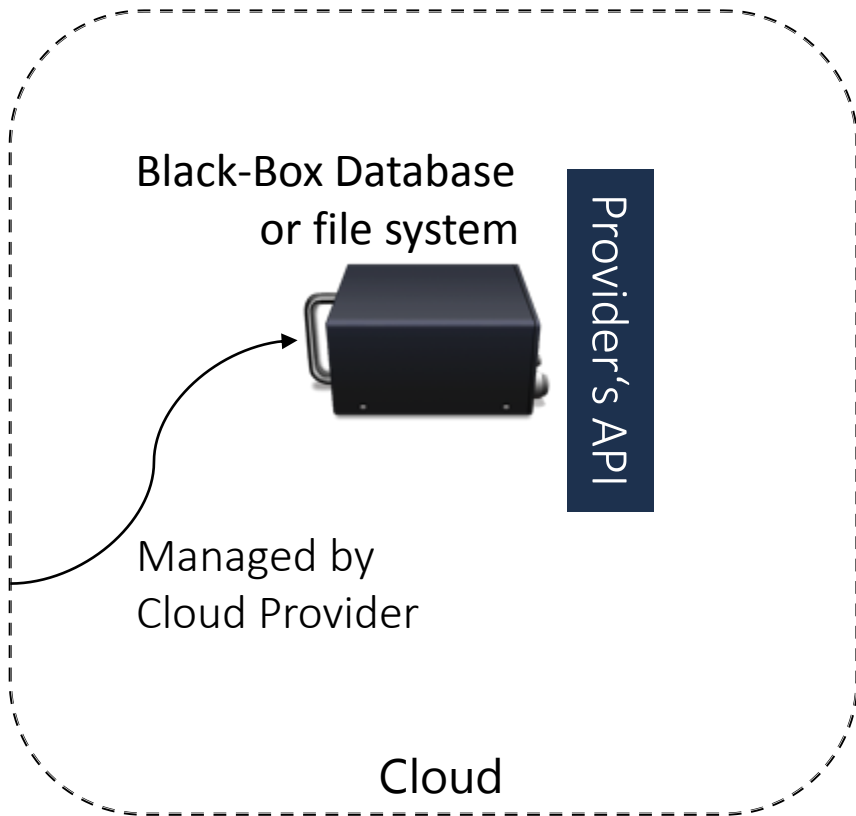


 Amazon RDS	 SQL Azure	RDBMS
 Clustrix	 EDB ENTERPRISEDB	
 Google Cloud SQL	 Heroku Postgres	

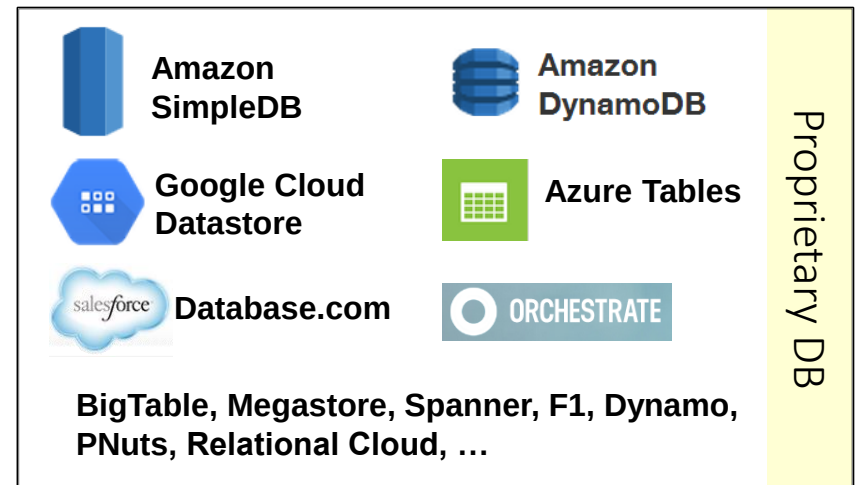
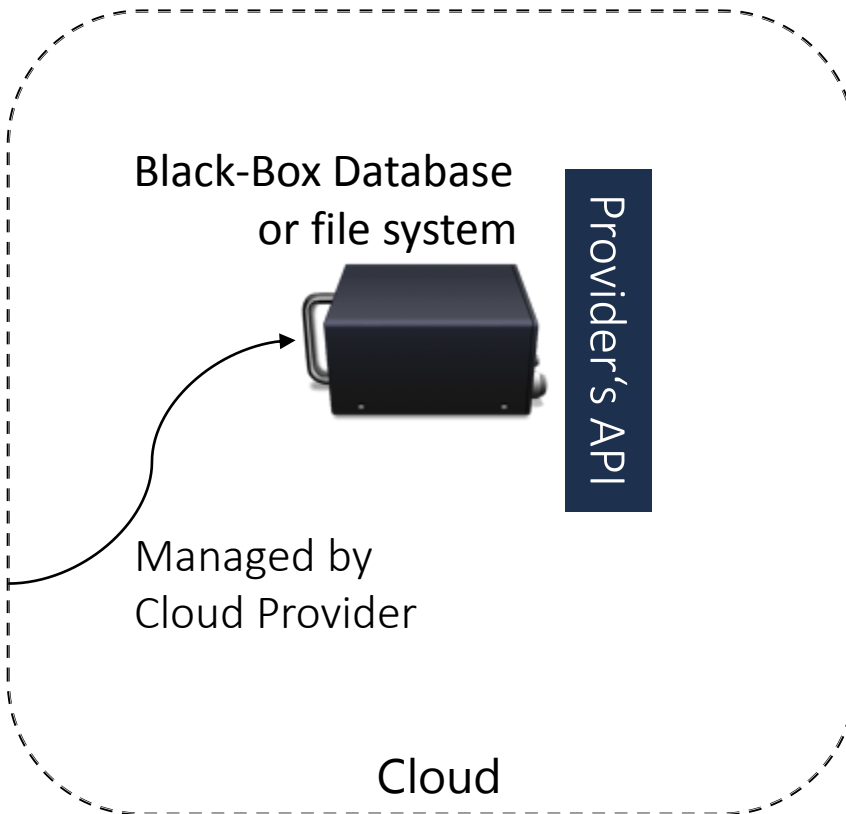
 Amazon ElastiCache	 mongoHQ	NoSQL DB
 mongolab	 Cloudant an IBM® Company	
 redis	 instaclustr	
 bonsai	 Iris Couch	

 Amazon Redshift	DWH
---	-----

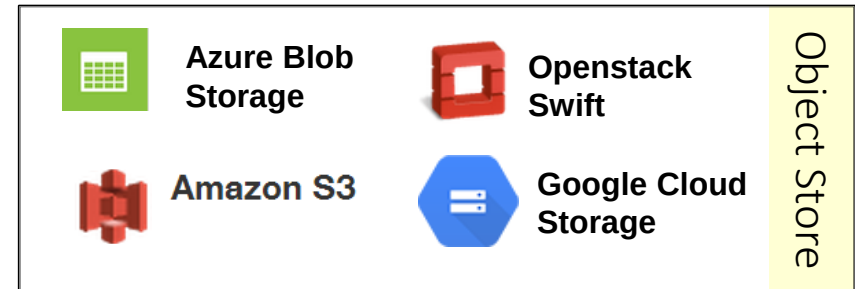
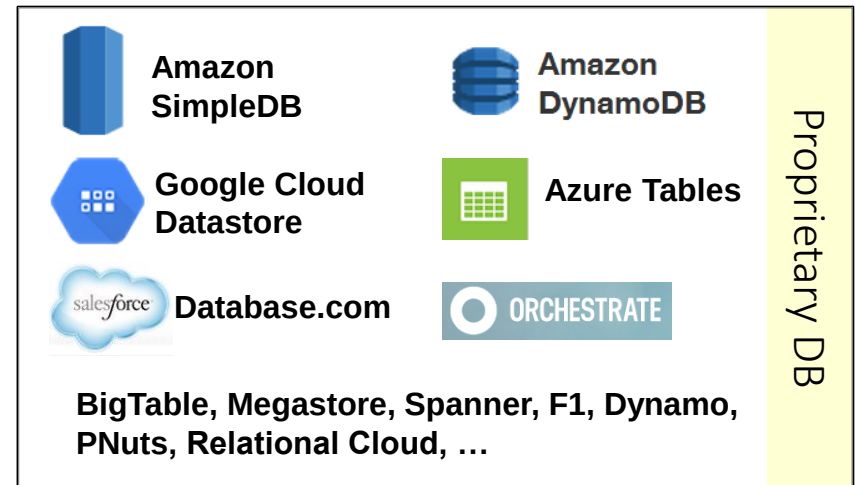
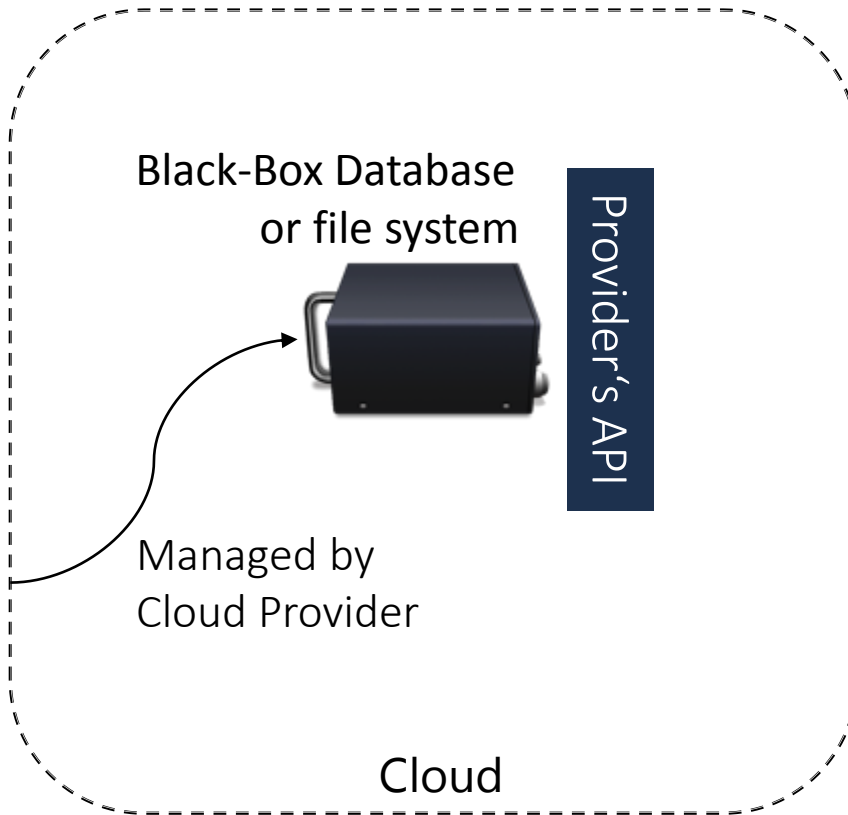
Proprietary DB/Object Store



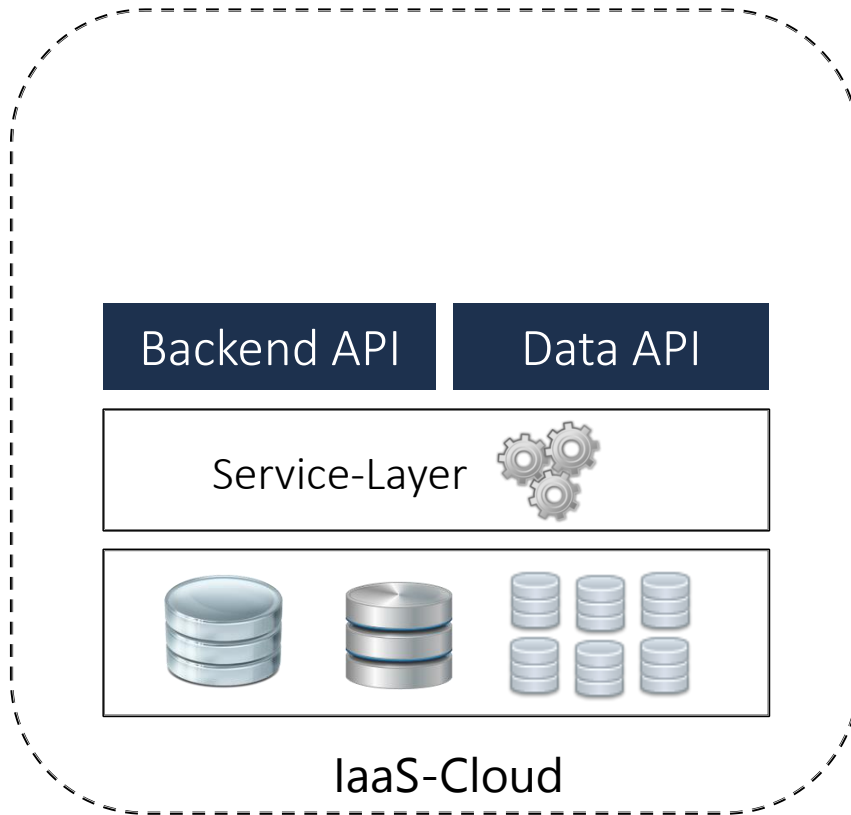
Proprietary DB/Object Store



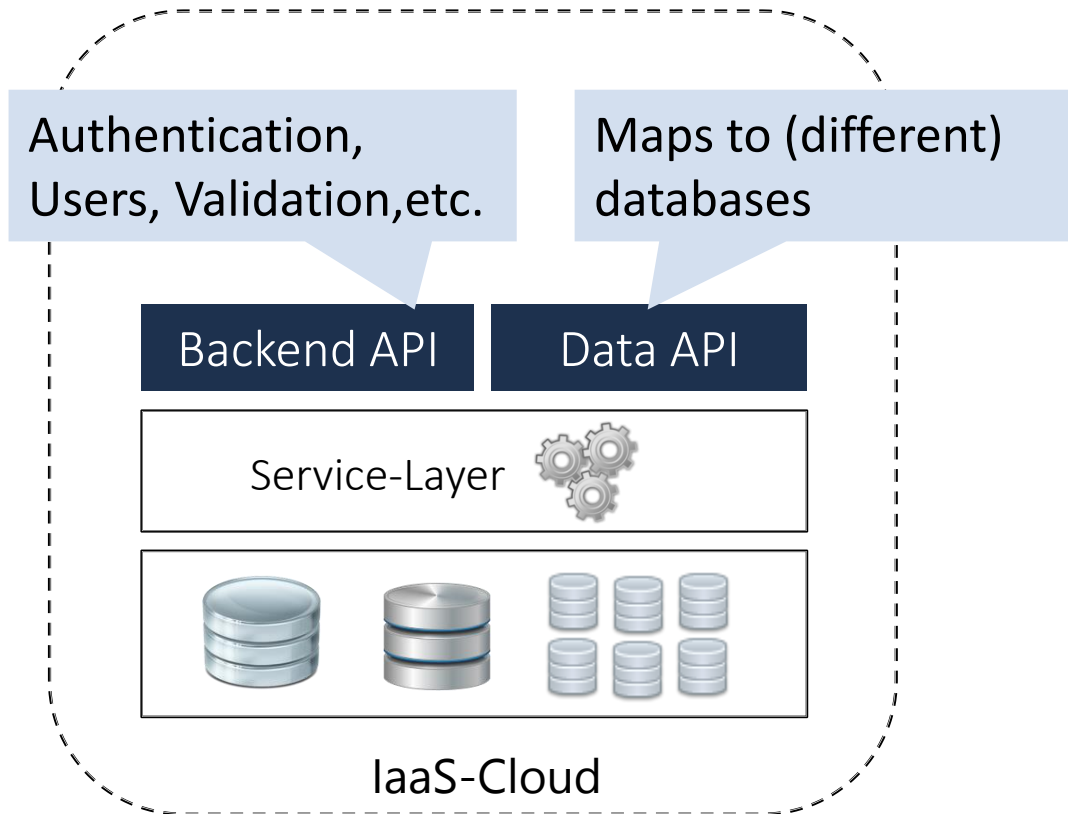
Proprietary DB/Object Store



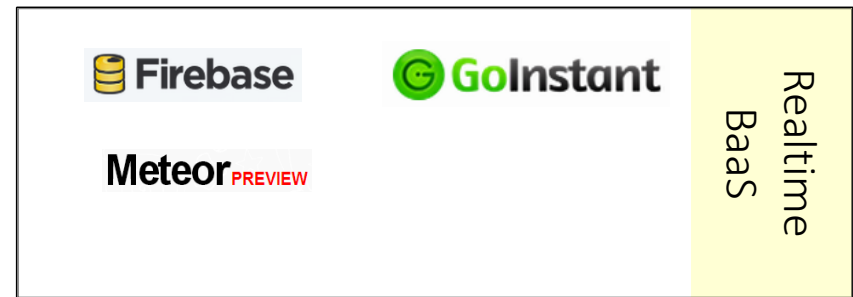
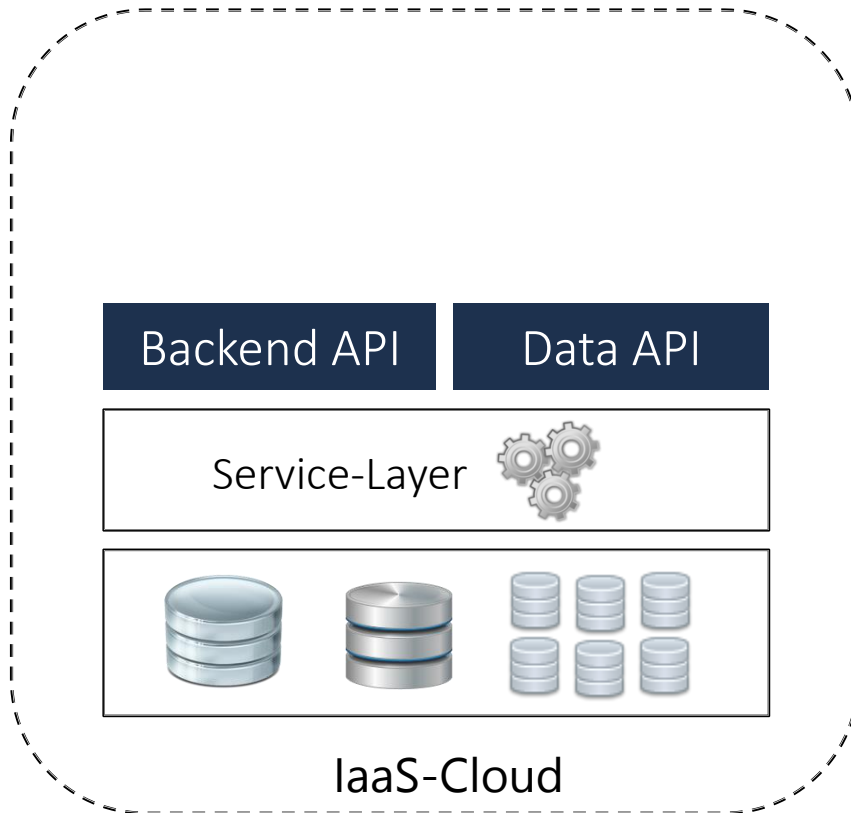
Backend-as-a-Service, Polyglot Persistence Service



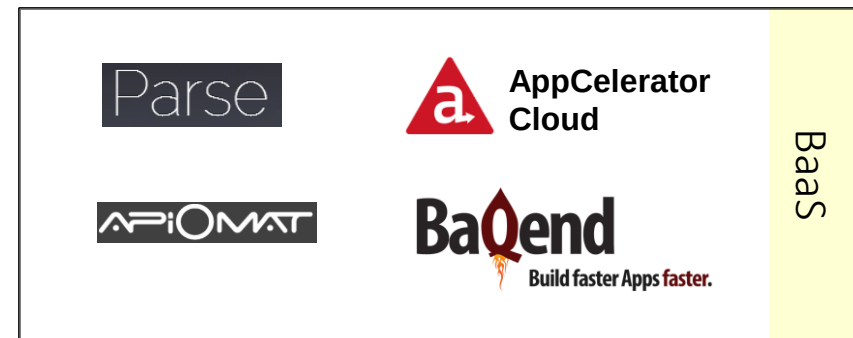
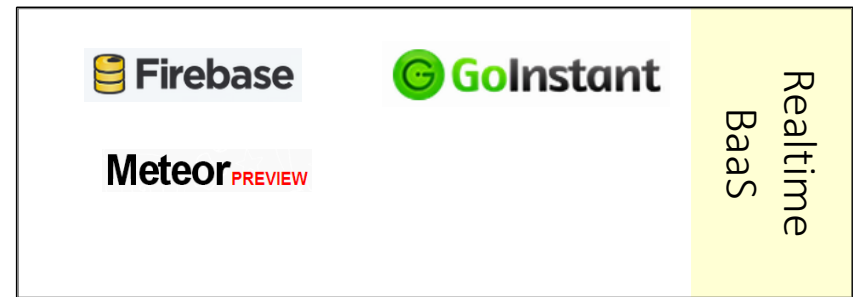
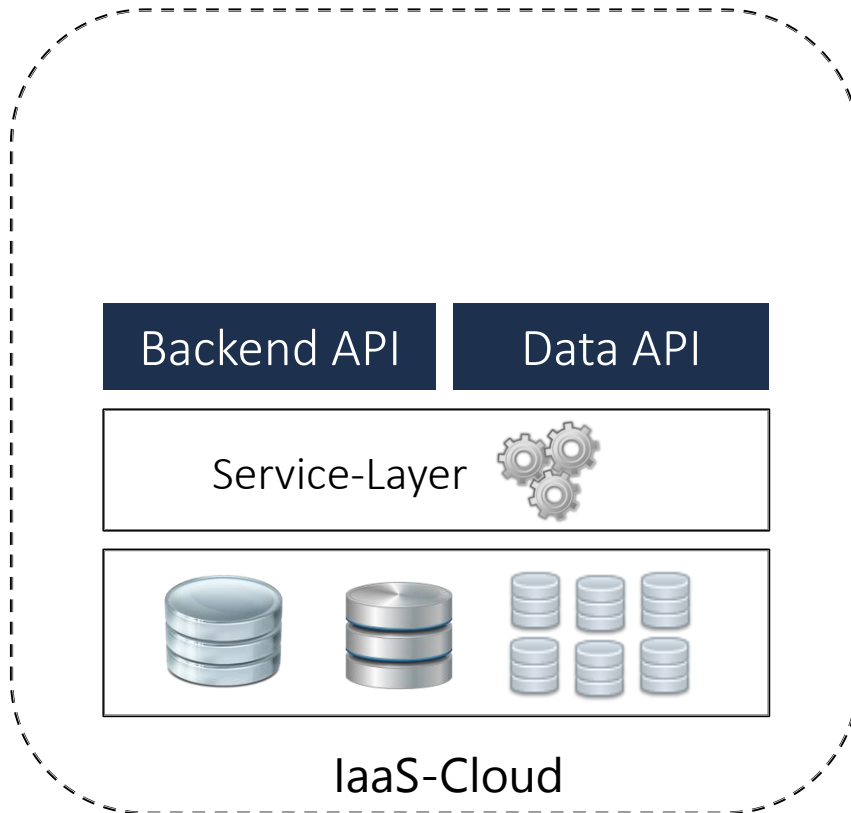
Backend-as-a-Service, Polyglot Persistence Service



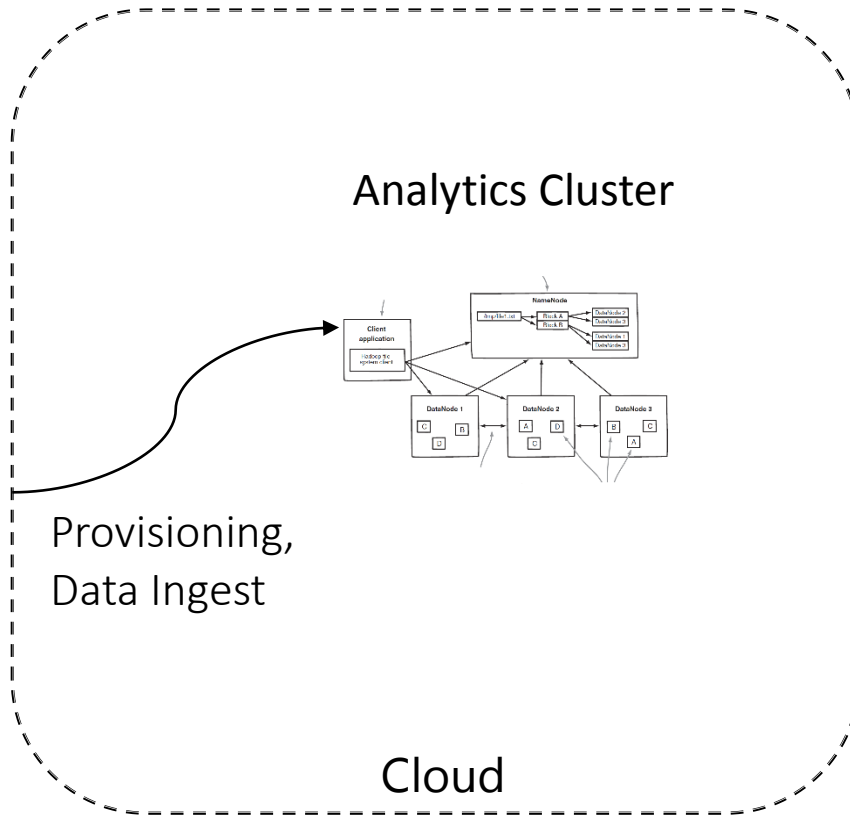
Backend-as-a-Service, Polyglot Persistence Service



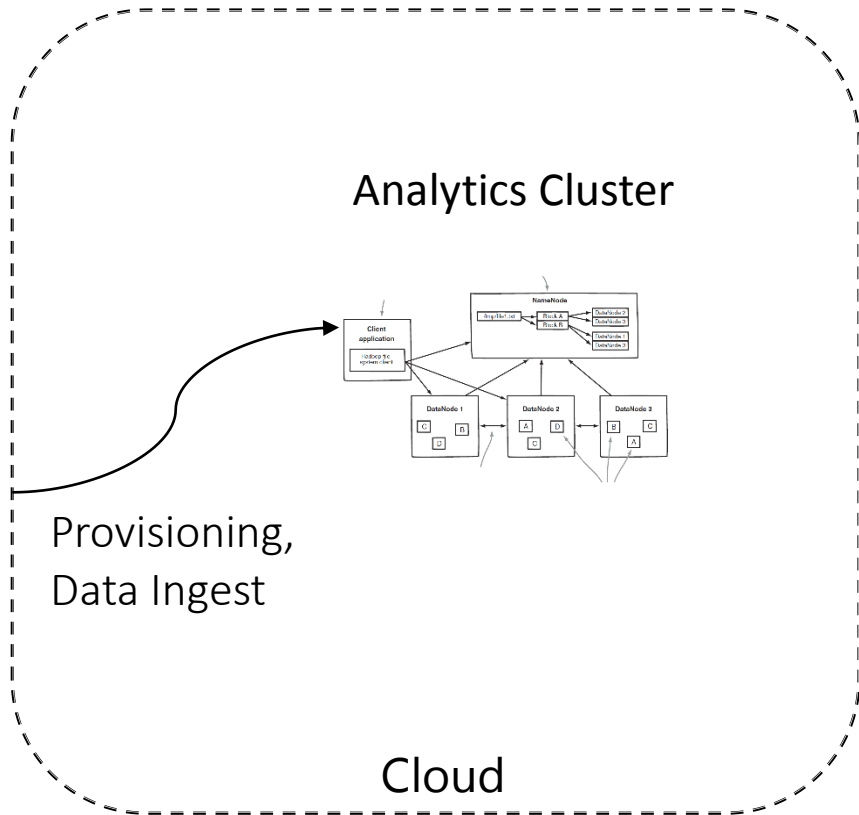
Backend-as-a-Service, Polyglot Persistence Service



Analytics-as-a-Service



Analytics-as-a-Service



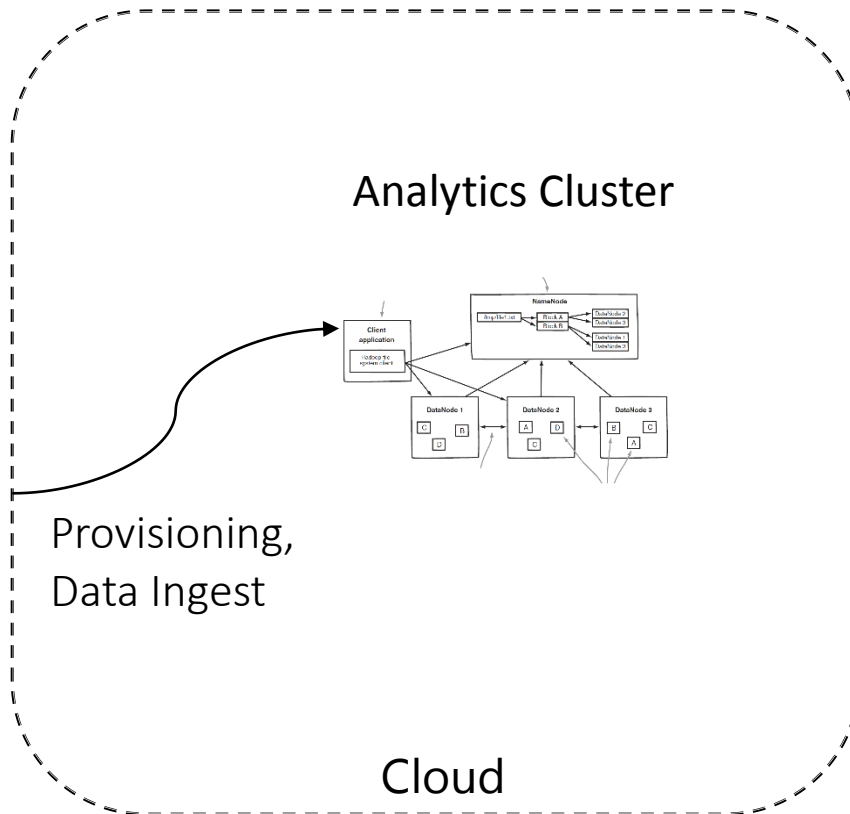
**Amazon Elastic
MapReduce**



**Azure
HDInsight**

Hadoop

Analytics-as-a-Service



**Amazon Elastic
MapReduce**



**Azure
HDInsight**

Hadoop



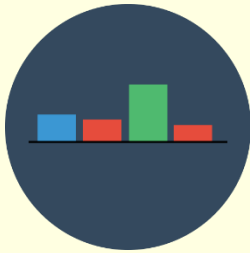
**Google
BigQuery**



**Google
Prediction API**

Custom

DBaaS: Common Aspects

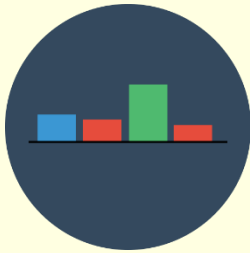


#1 Metric: Total Cost



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

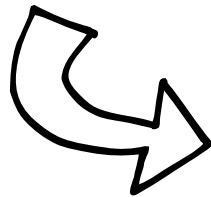
DBaaS: Common Aspects



#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



DBaaS: Common Aspects

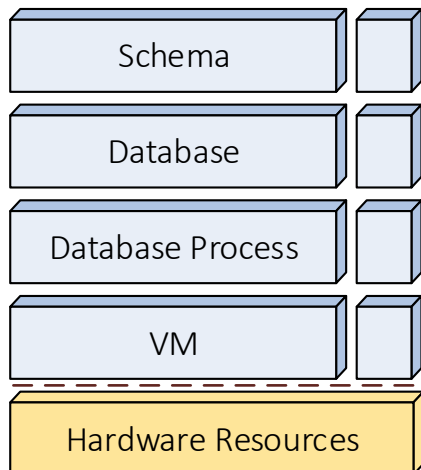
- ▶ **Multi-Tenancy** - four common approaches:

Private OS

Private Process/DB

Private Schema

Shared Schema

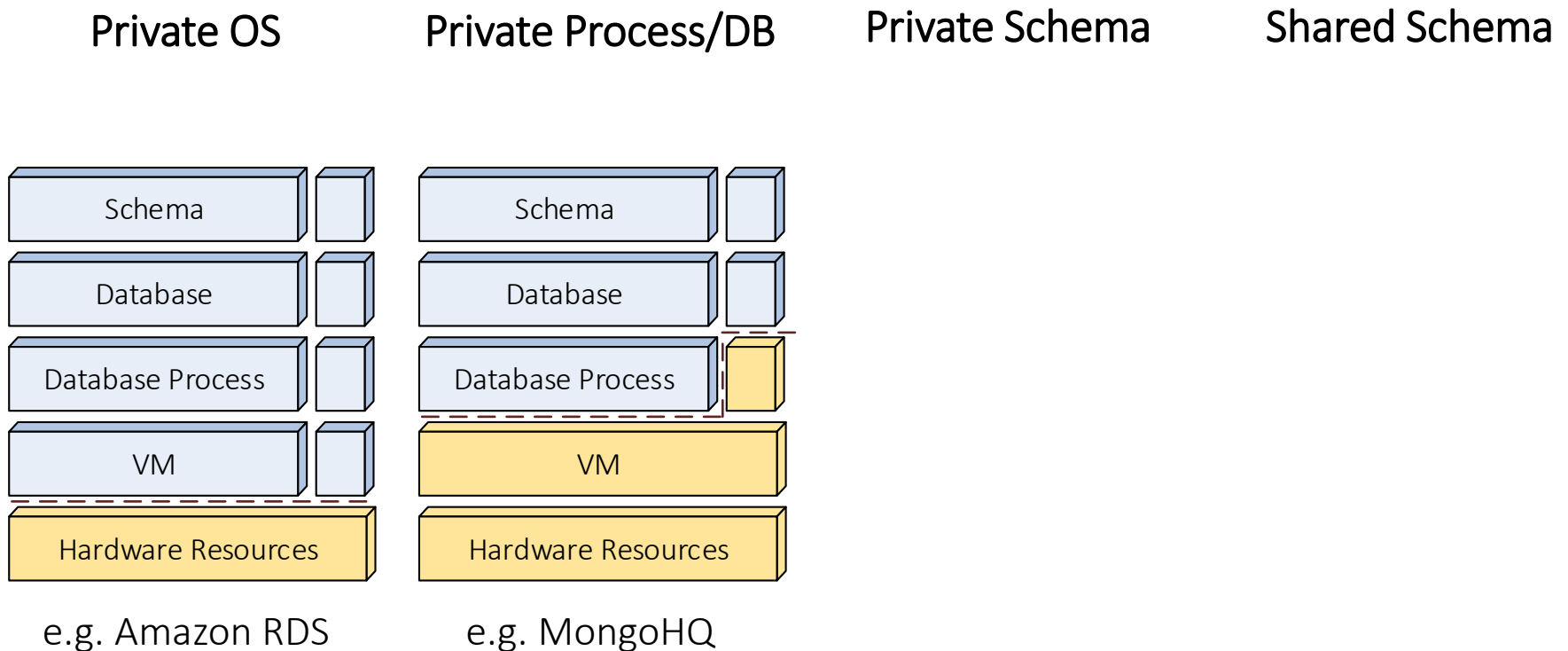


e.g. Amazon RDS



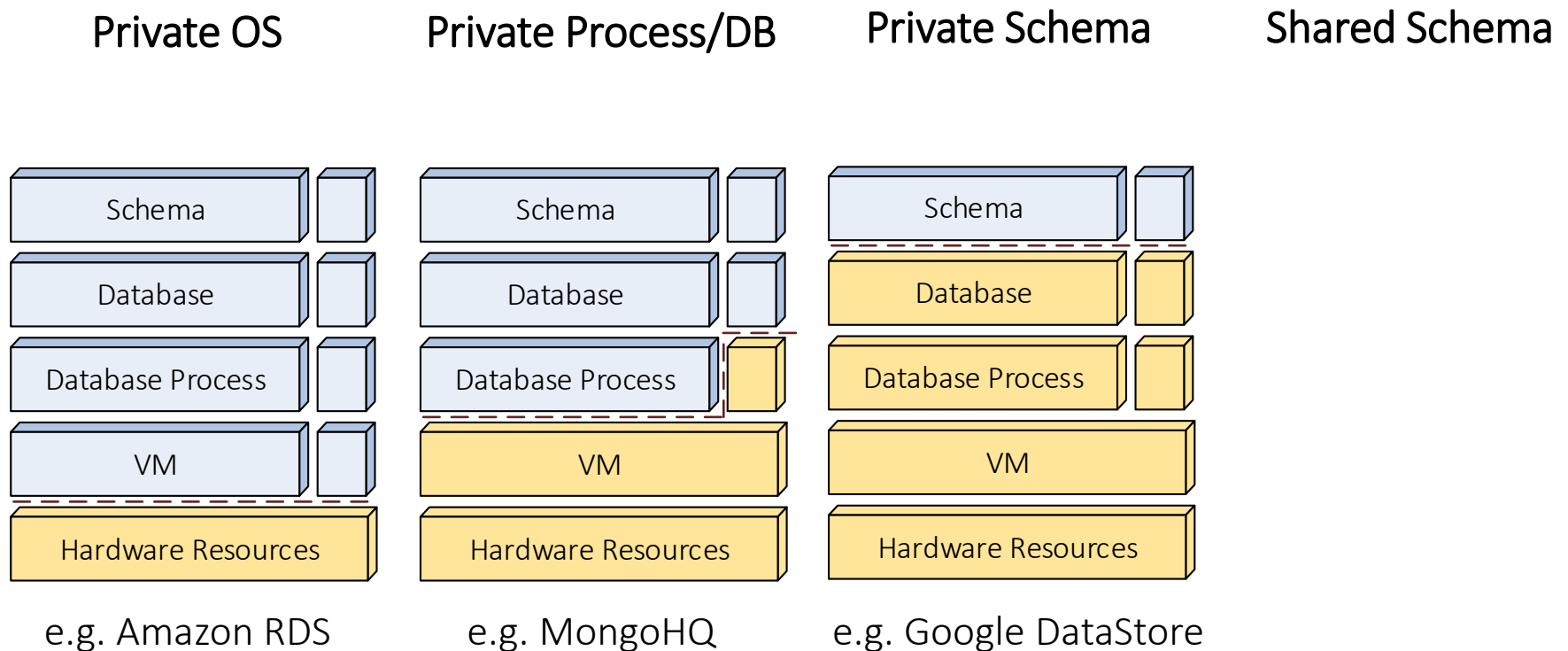
DBaaS: Common Aspects

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



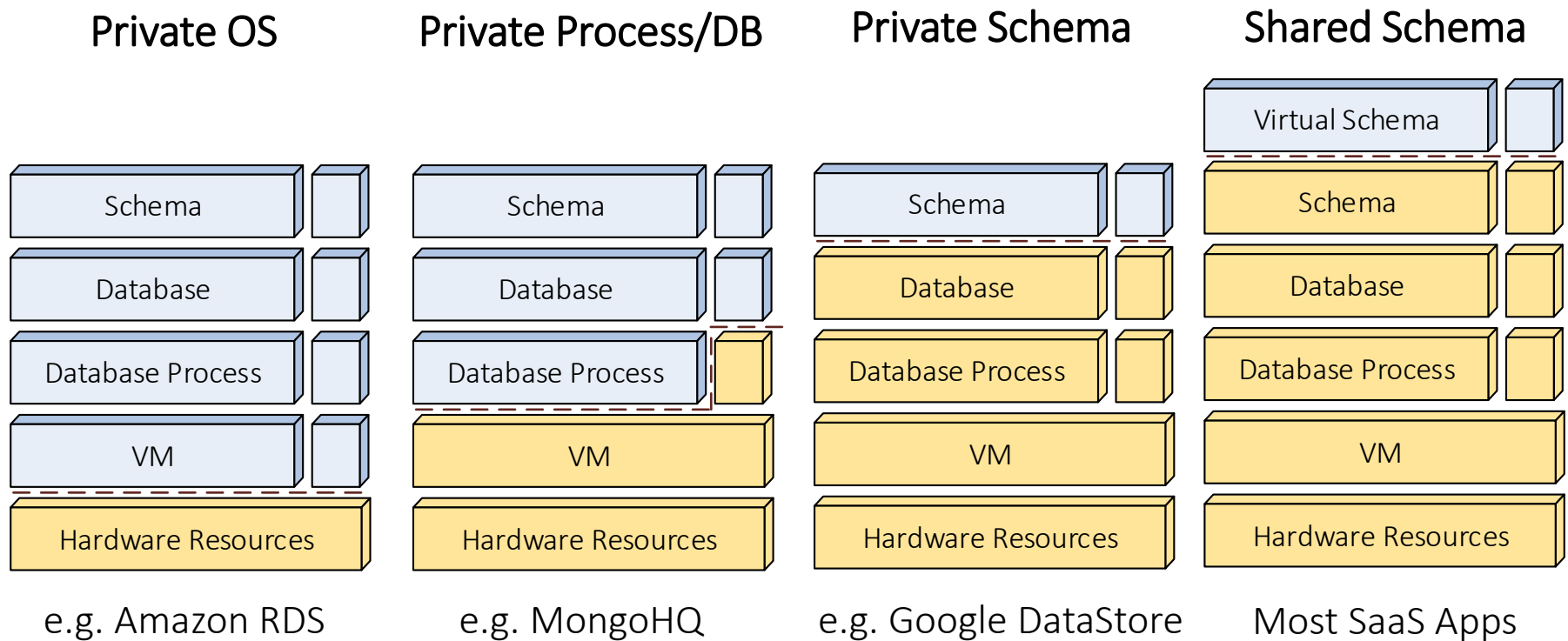
DBaaS: Common Aspects

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



DBaaS: Common Aspects

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



DBaaS: Common Aspects

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

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



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



Account

Usage

End of month



DBaaS: Common Aspects

▶ Billing Models:

Plan-based

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

e.g. MongoHQ



Account



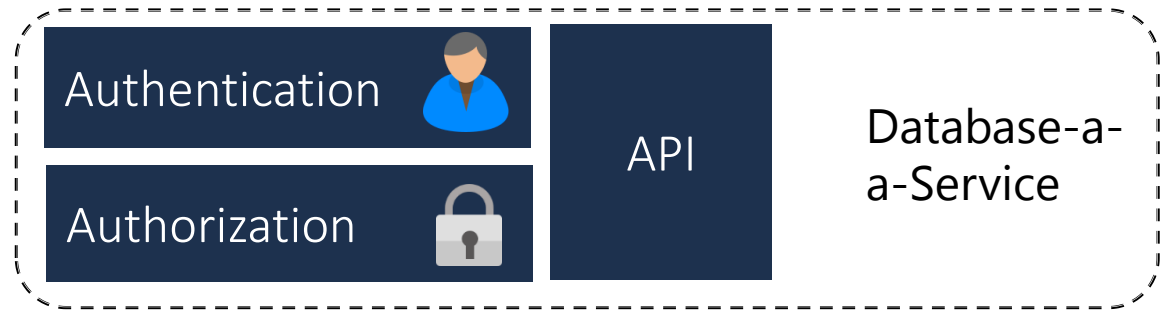
Usage



End of
month

Free Tier: free plan or free initial account credit

DBaaS: Common Aspects



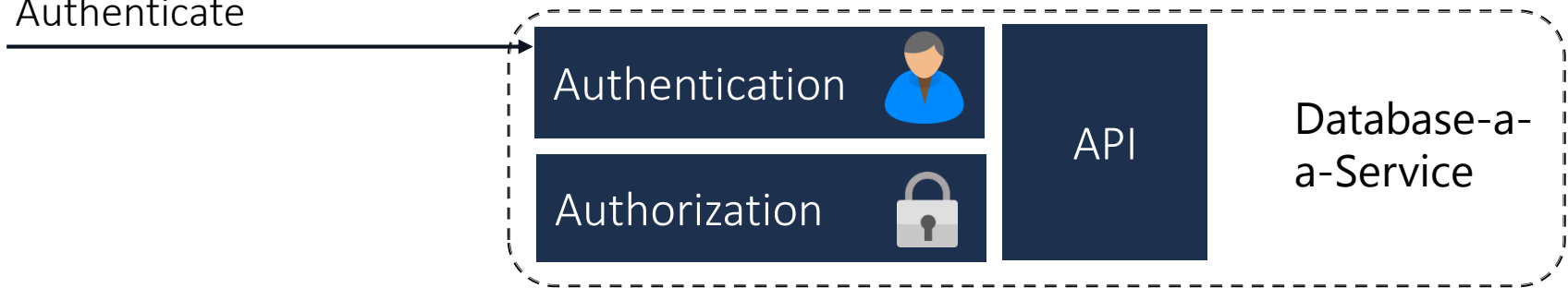
DBaaS: Common Aspects



DBaaS: Common Aspects

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

Authenticate

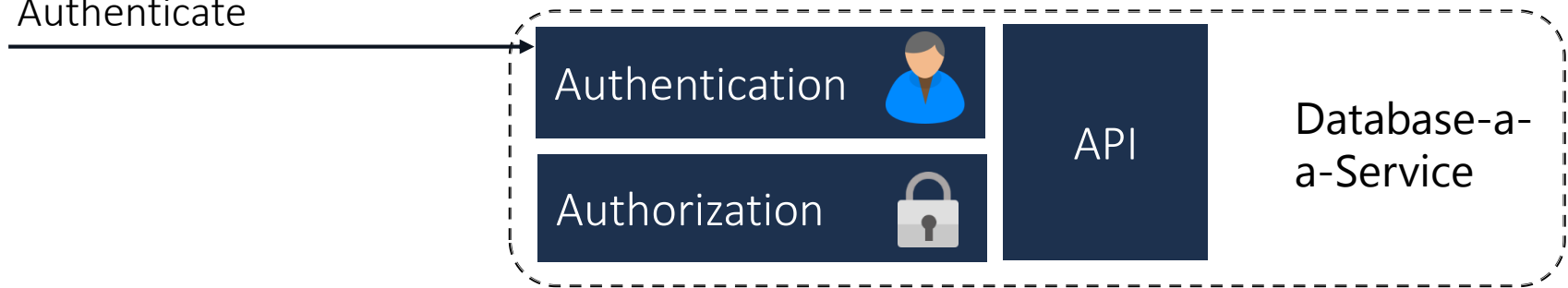


DBaaS: Common Aspects

Used extensively

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

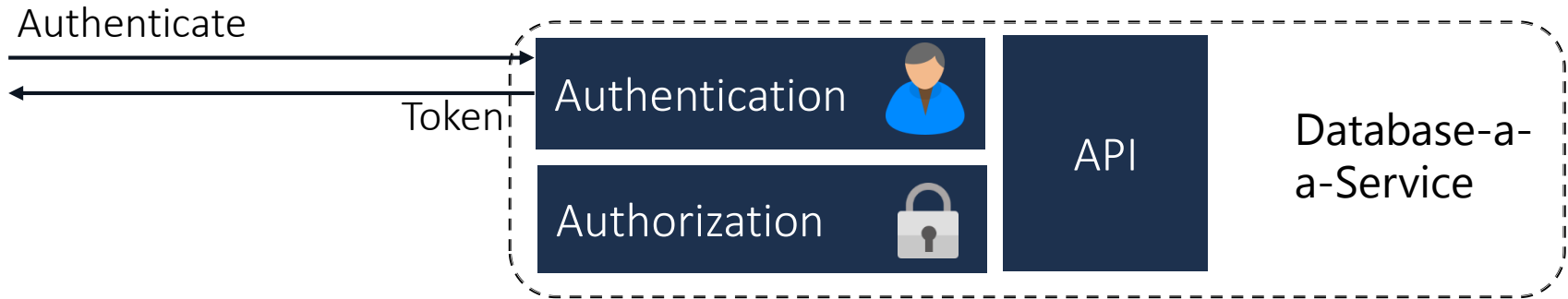
Authenticate



DBaaS: Common Aspects

Used extensively

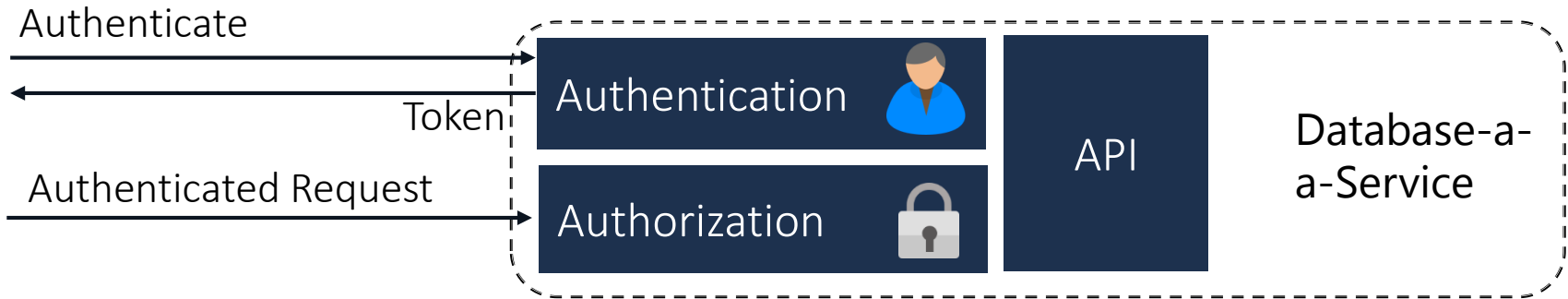
Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML



DBaaS: Common Aspects

Used extensively

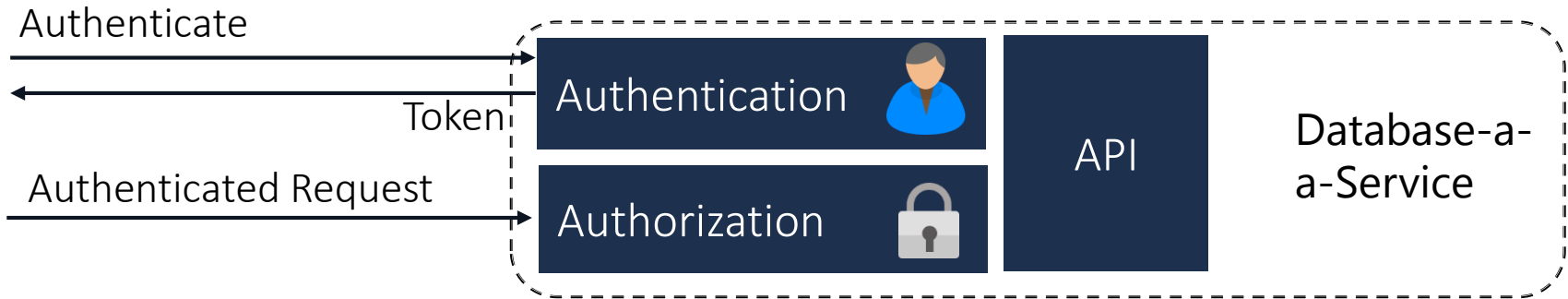
Internal Schemes	External Identity Provider	Federated Identity (SSO)
e.g. Amazon IAM	e.g. OpenID	e.g. SAML



DBaaS: Common Aspects

Used extensively

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

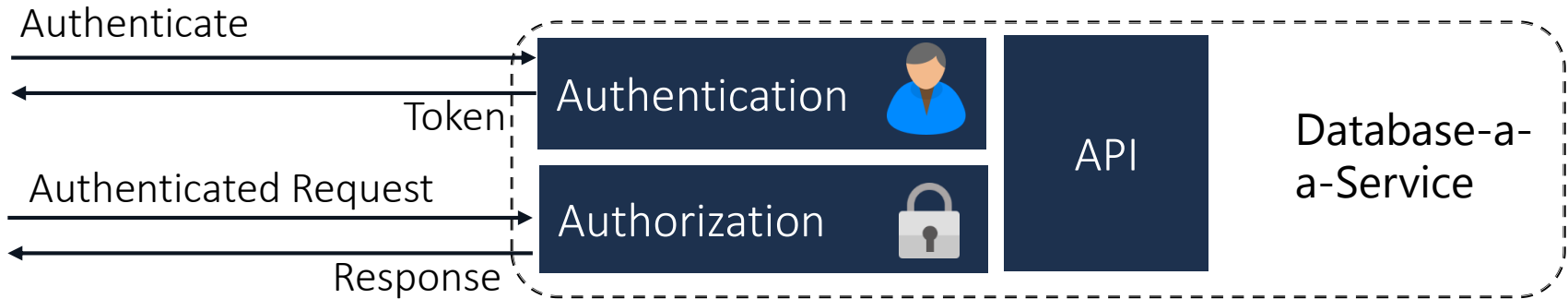


User-based Access Control	Role-based Access Control	Policies
e.g. Amazon S3 ACLs	e.g. Amazon IAM	e.g. XACML

DBaaS: Common Aspects

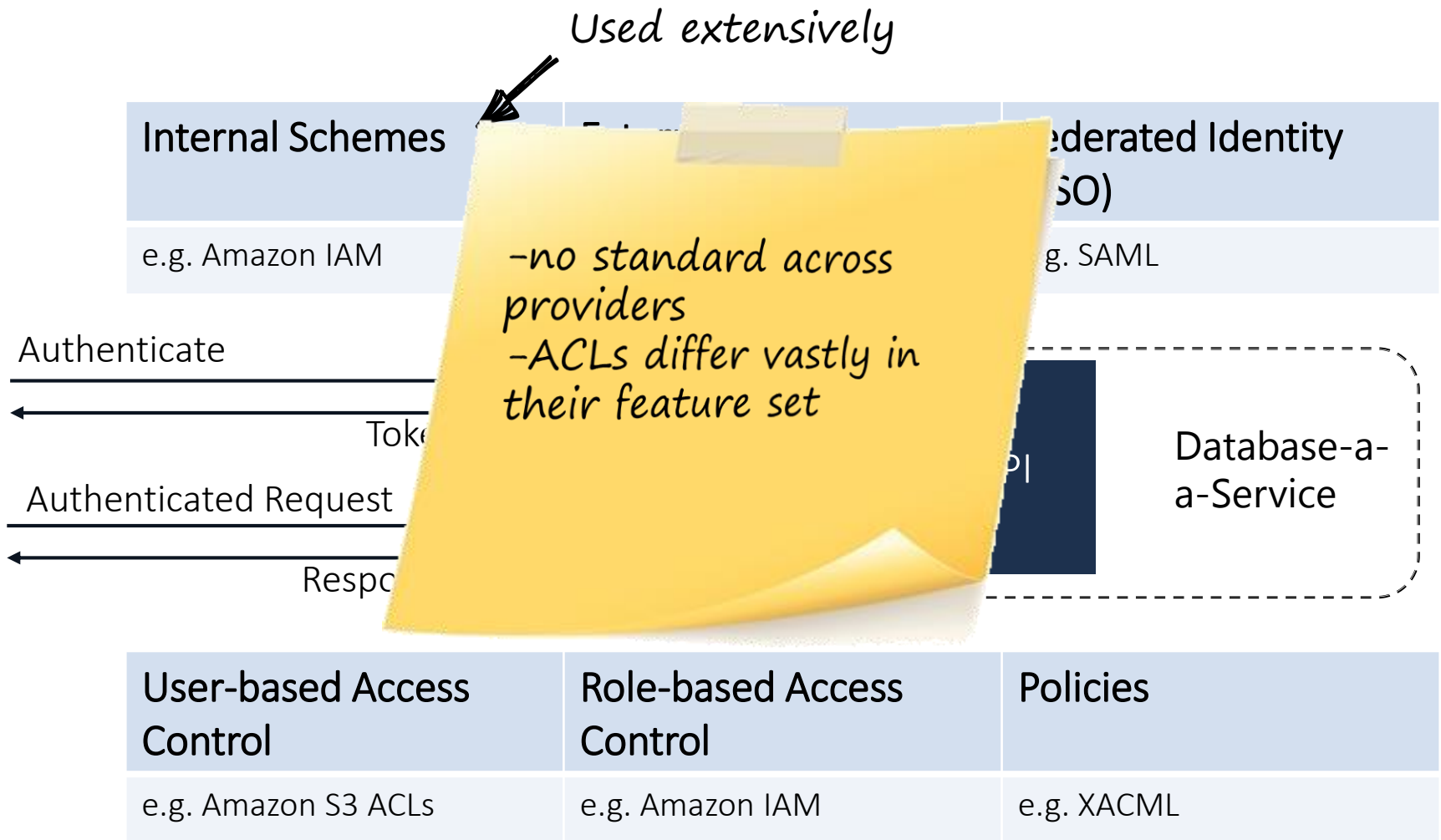
Used extensively

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

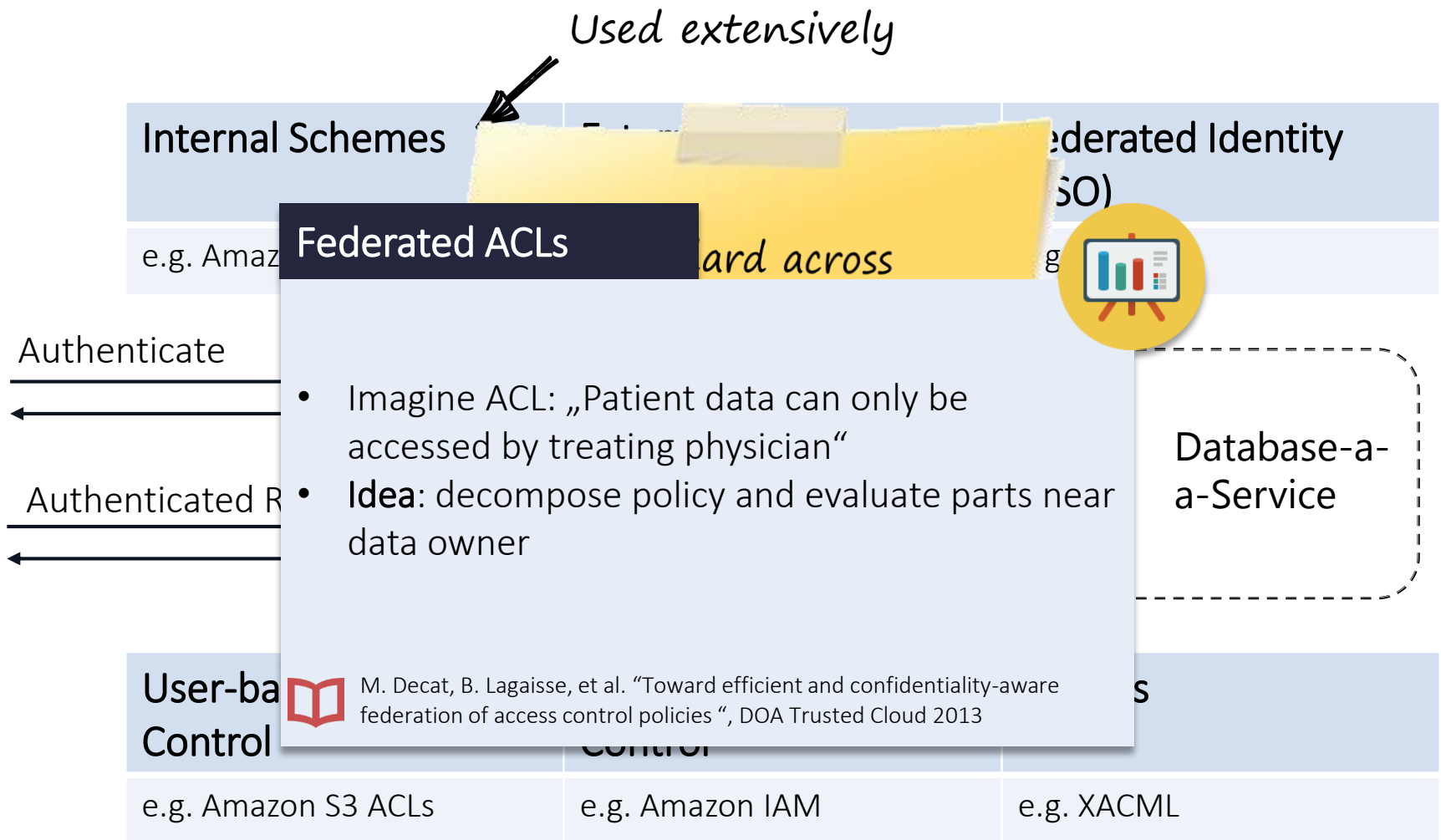


User-based Access Control	Role-based Access Control	Policies
e.g. Amazon S3 ACLs	e.g. Amazon IAM	e.g. XACML

DBaaS: Common Aspects



DBaaS: Common Aspects



DBaaS: Common Aspects

- ▶ Service Level Agreements



DBaaS: Common Aspects

▶ Service Level Agreements

SLA

Technical Part

1. SLO
2. SLO
3. SLO

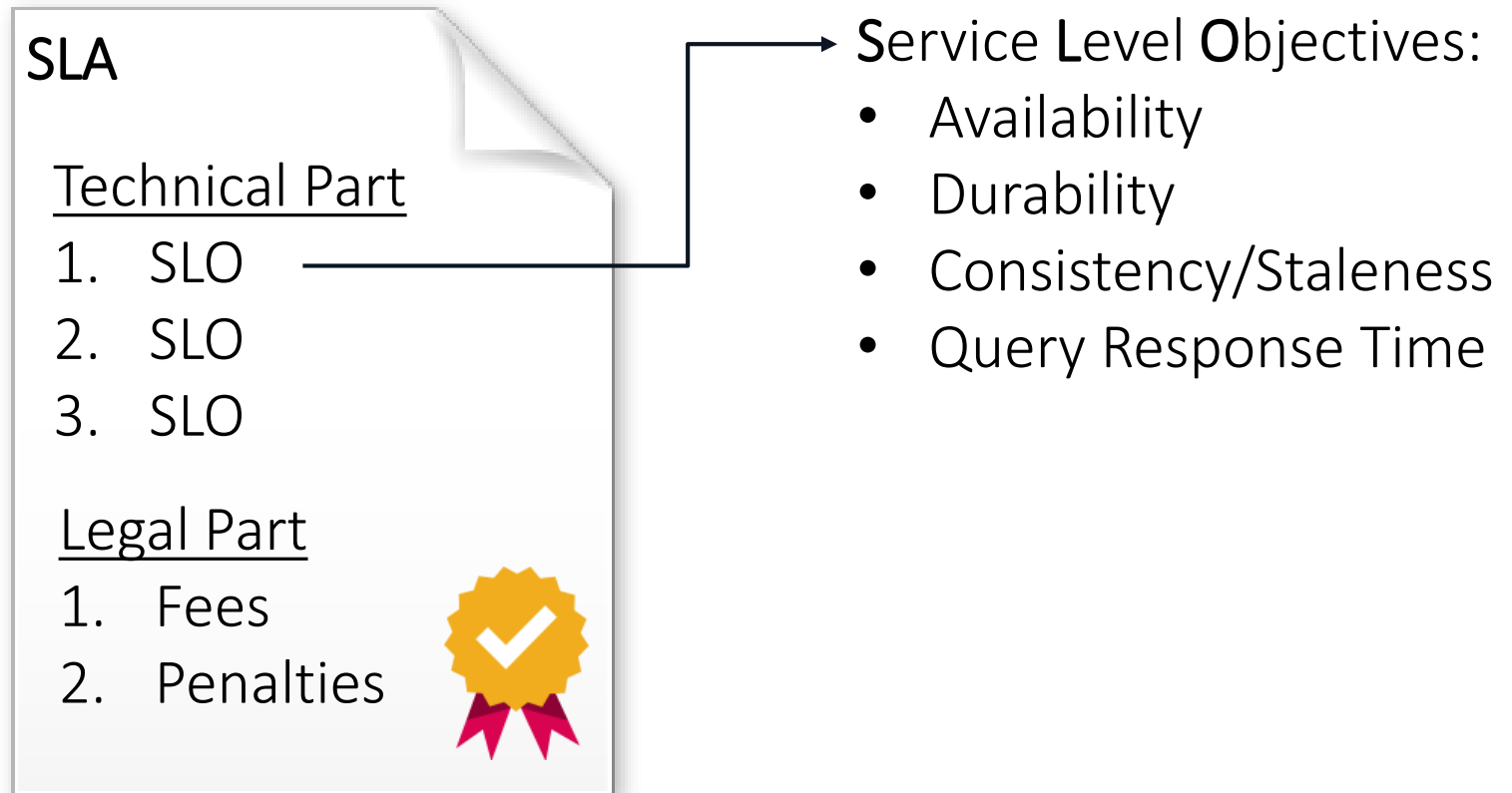
Legal Part

1. Fees
2. Penalties



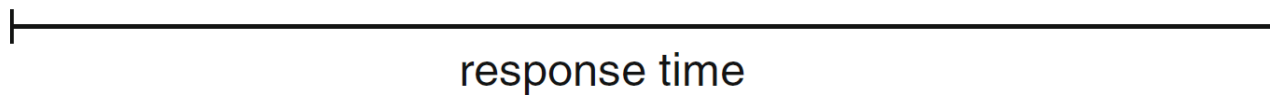
DBaaS: Common Aspects

▶ Service Level Agreements



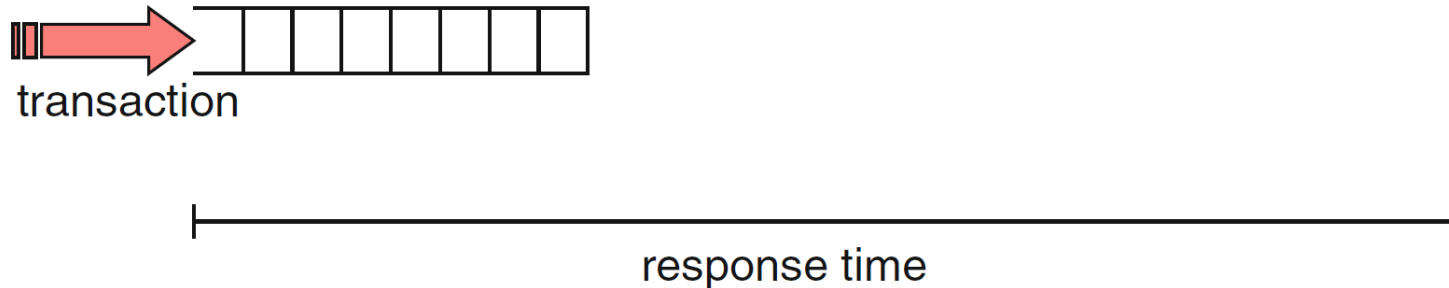
DBaaS: Common Aspects

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



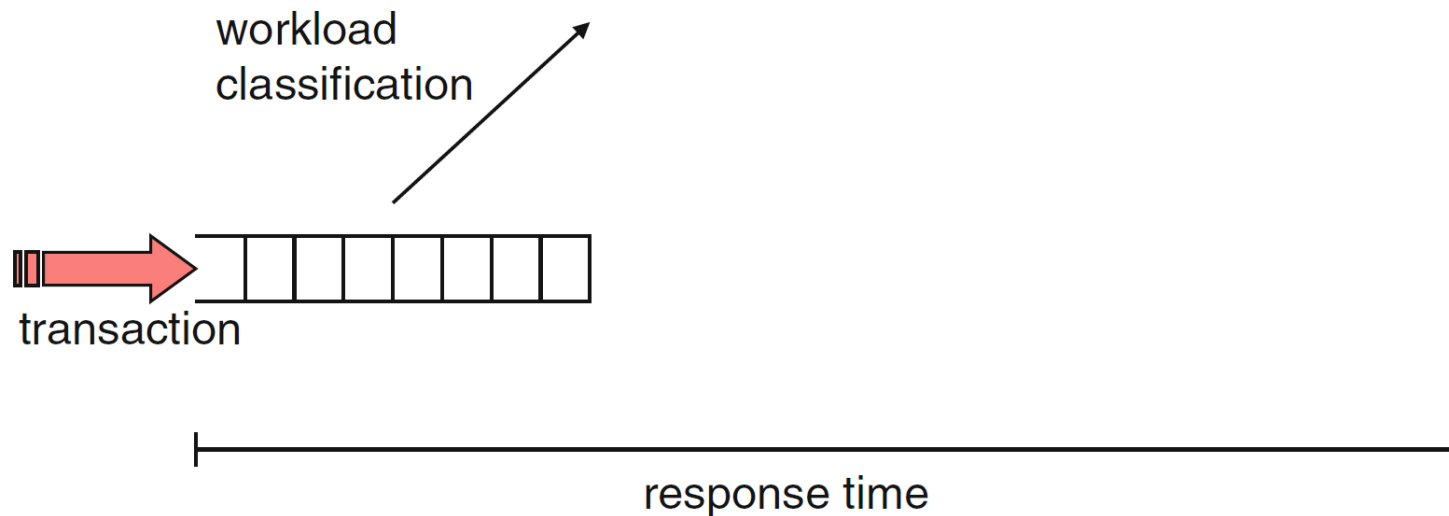
DBaaS: Common Aspects

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



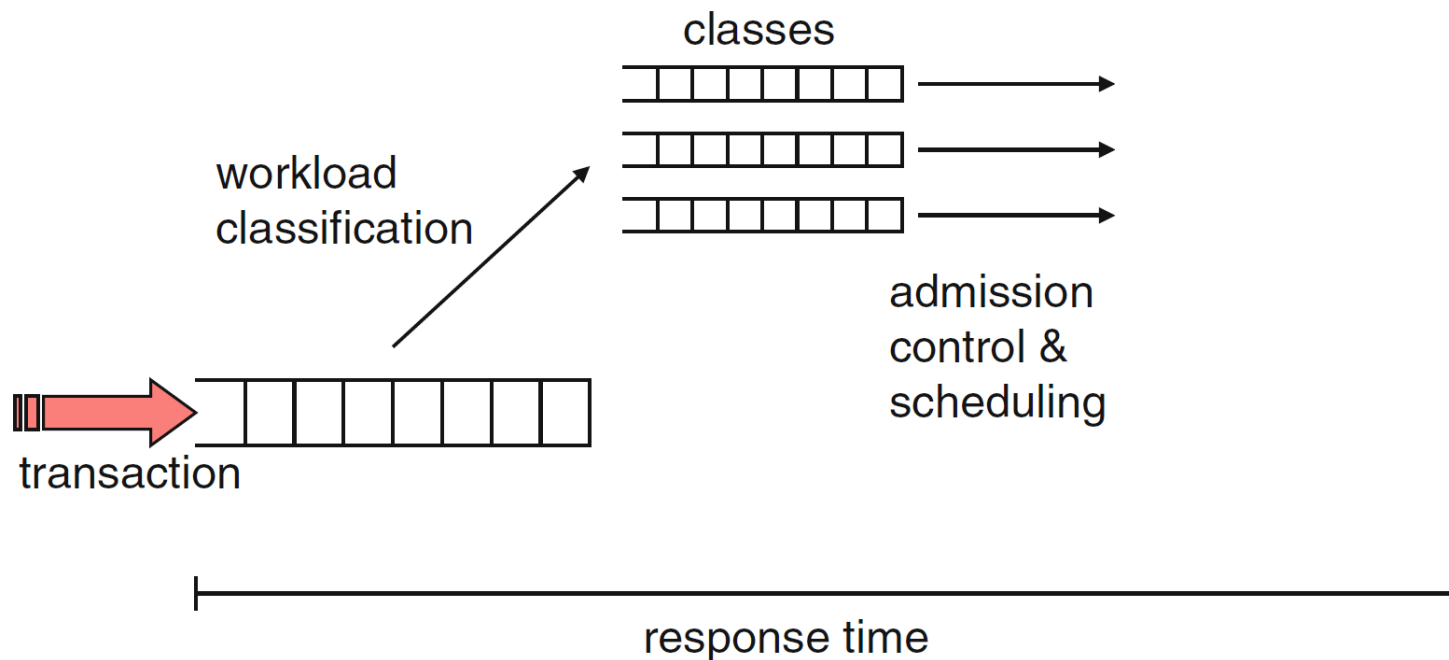
DBaaS: Common Aspects

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



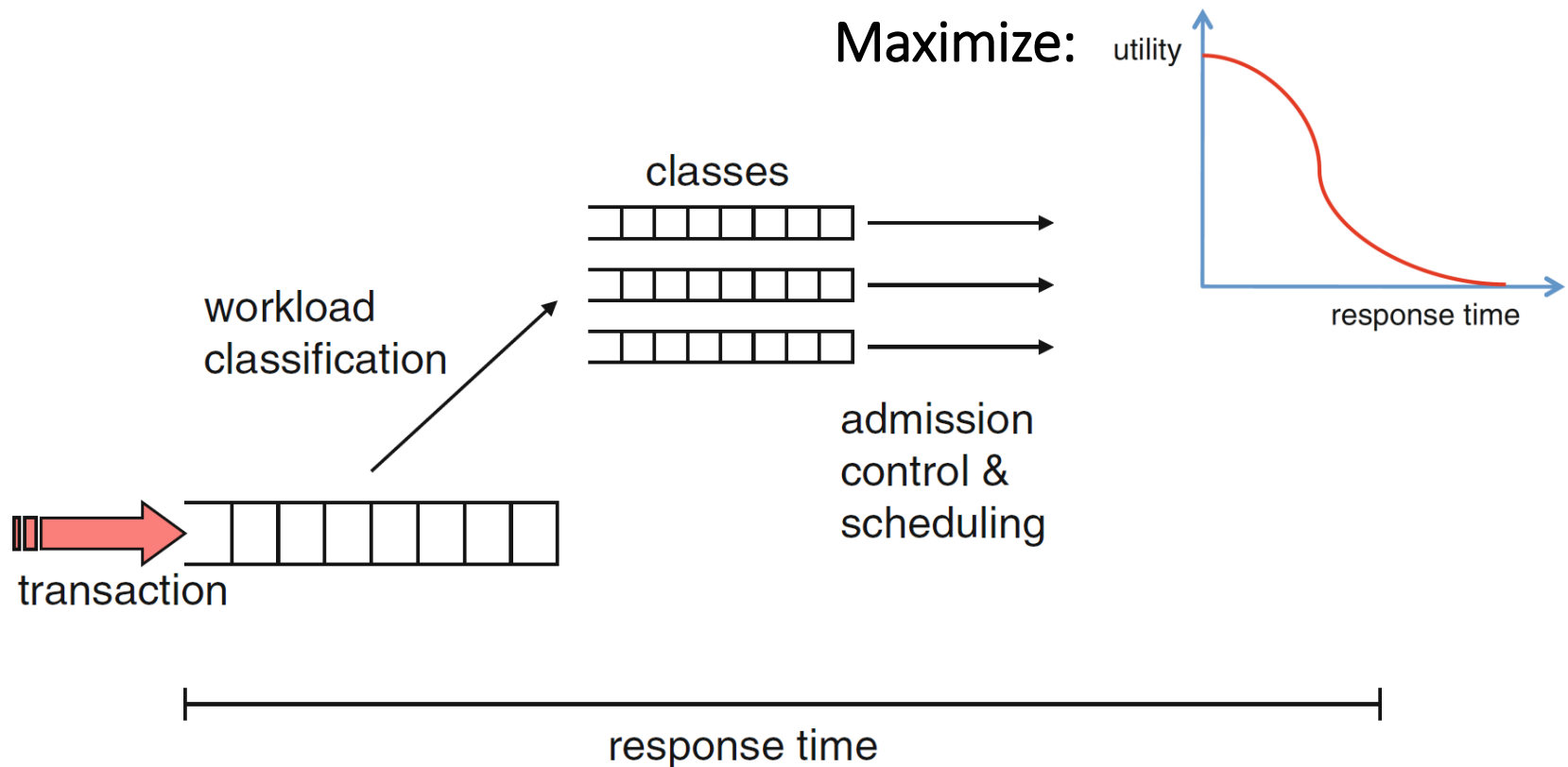
DBaaS: Common Aspects

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



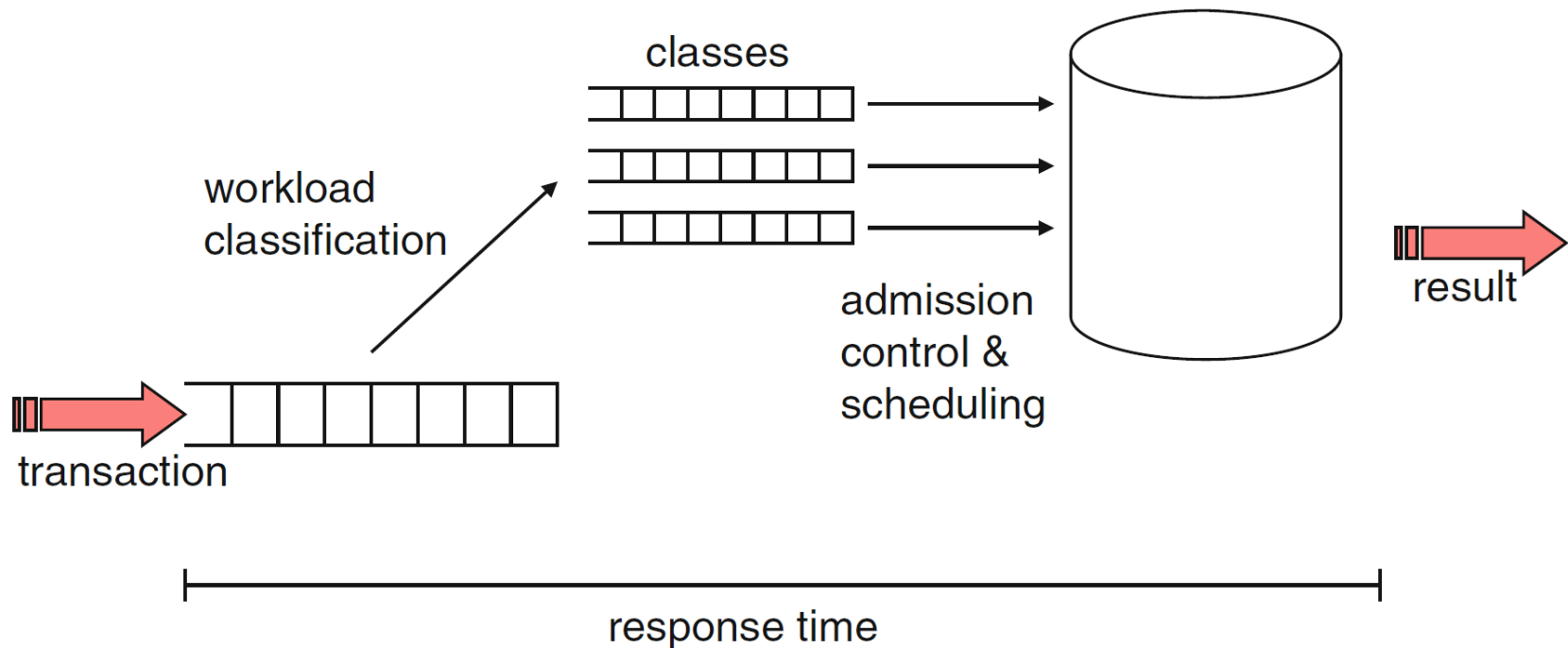
DBaaS: Common Aspects

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



DBaaS: Common Aspects

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



DBaaS: Common Aspects

- ▶ SLAs – achieved through Workload Management

QOS for NoSQL DBs

Old:

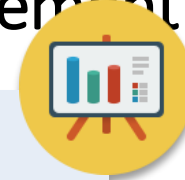
Workload management in RDBMs (DB2 and Oracle)

New:

workload Use well-known scheduling algorithms for queries
classification in replicated DBs



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



transaction



scheduling



result



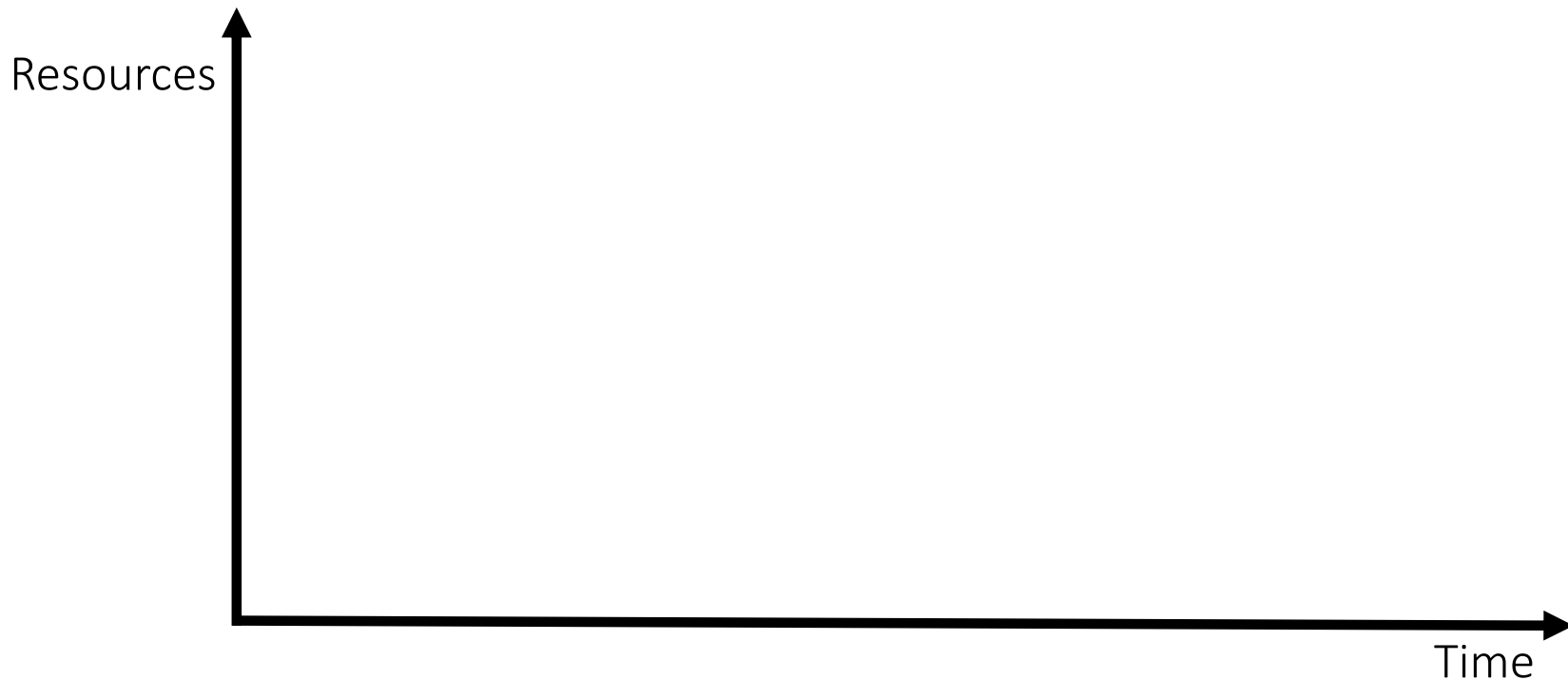
response time



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

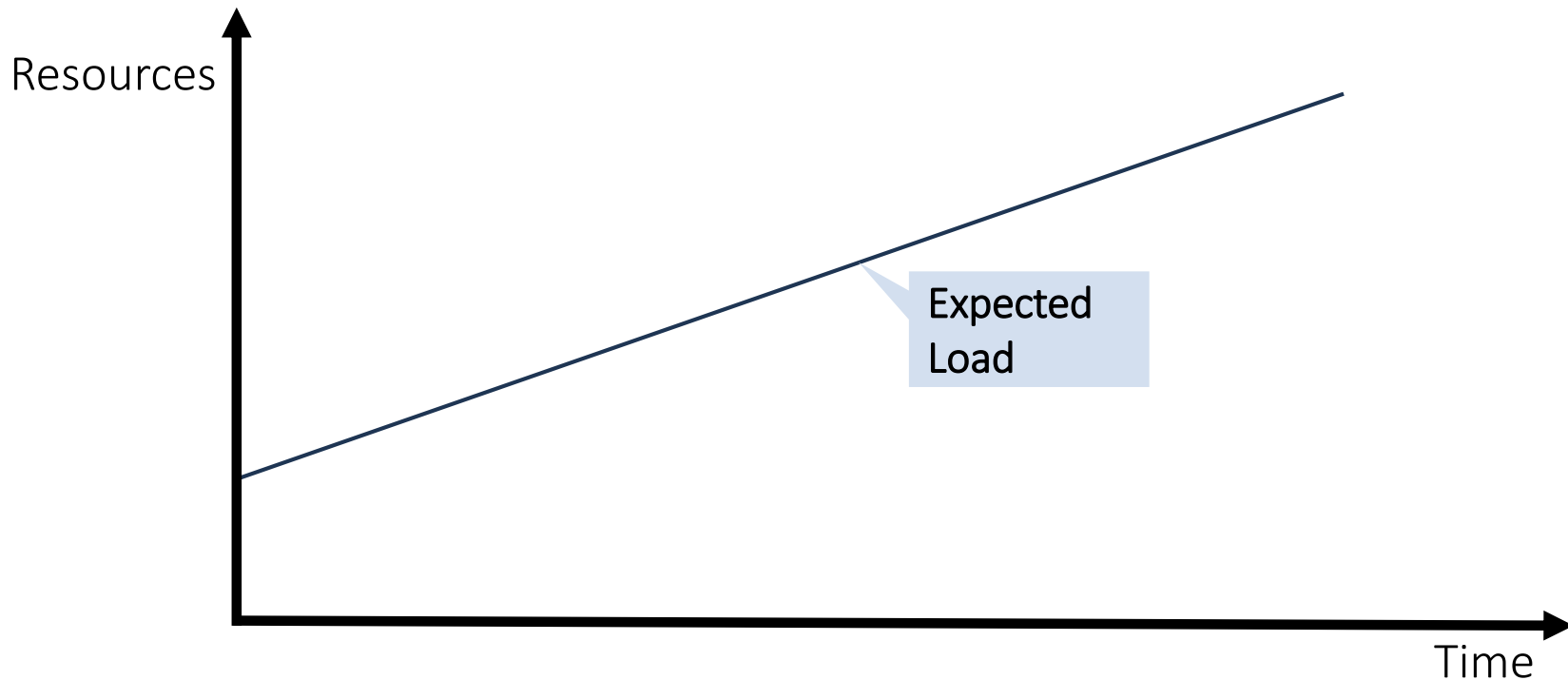
DBaaS: Common Aspects

- ▶ Resource Provisioning
- ▶ Goal: Resources $\Leftrightarrow$ SLAs



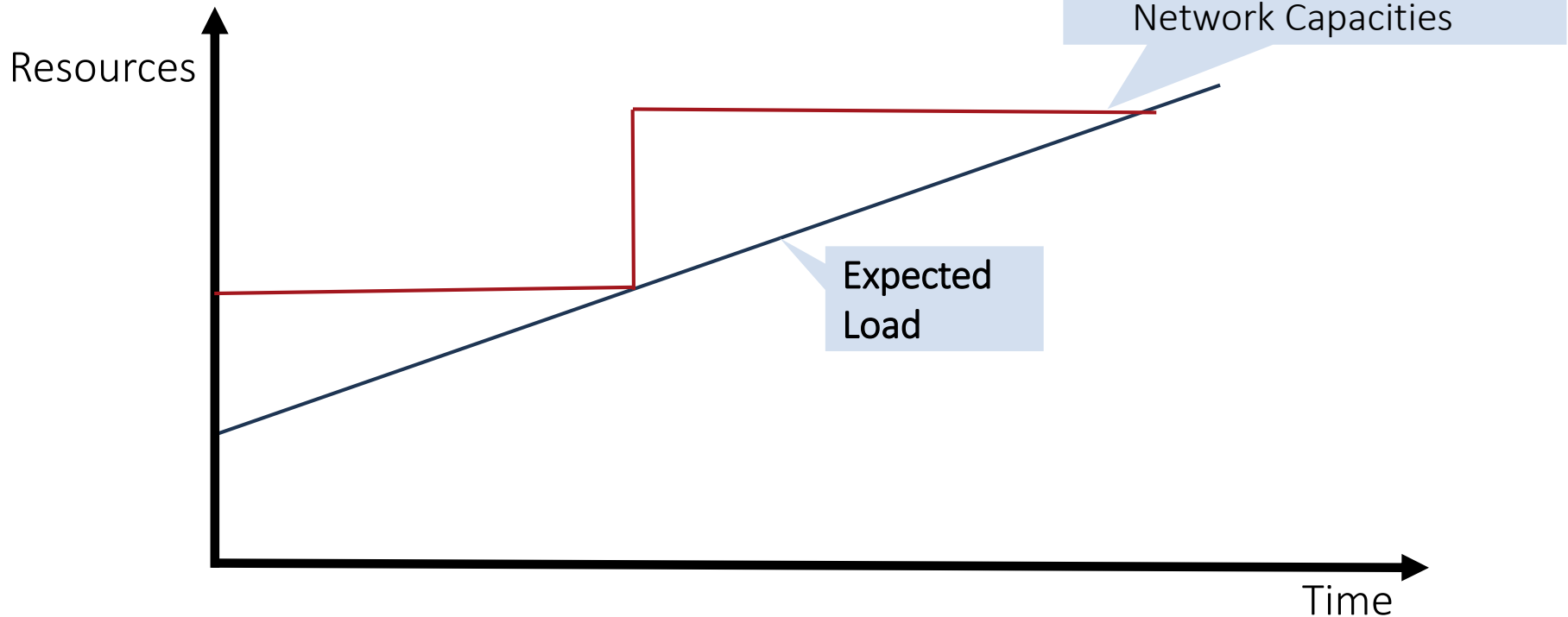
DBaaS: Common Aspects

- ▶ Resource Provisioning
- ▶ Goal: Resources $\Leftrightarrow$ SLAs



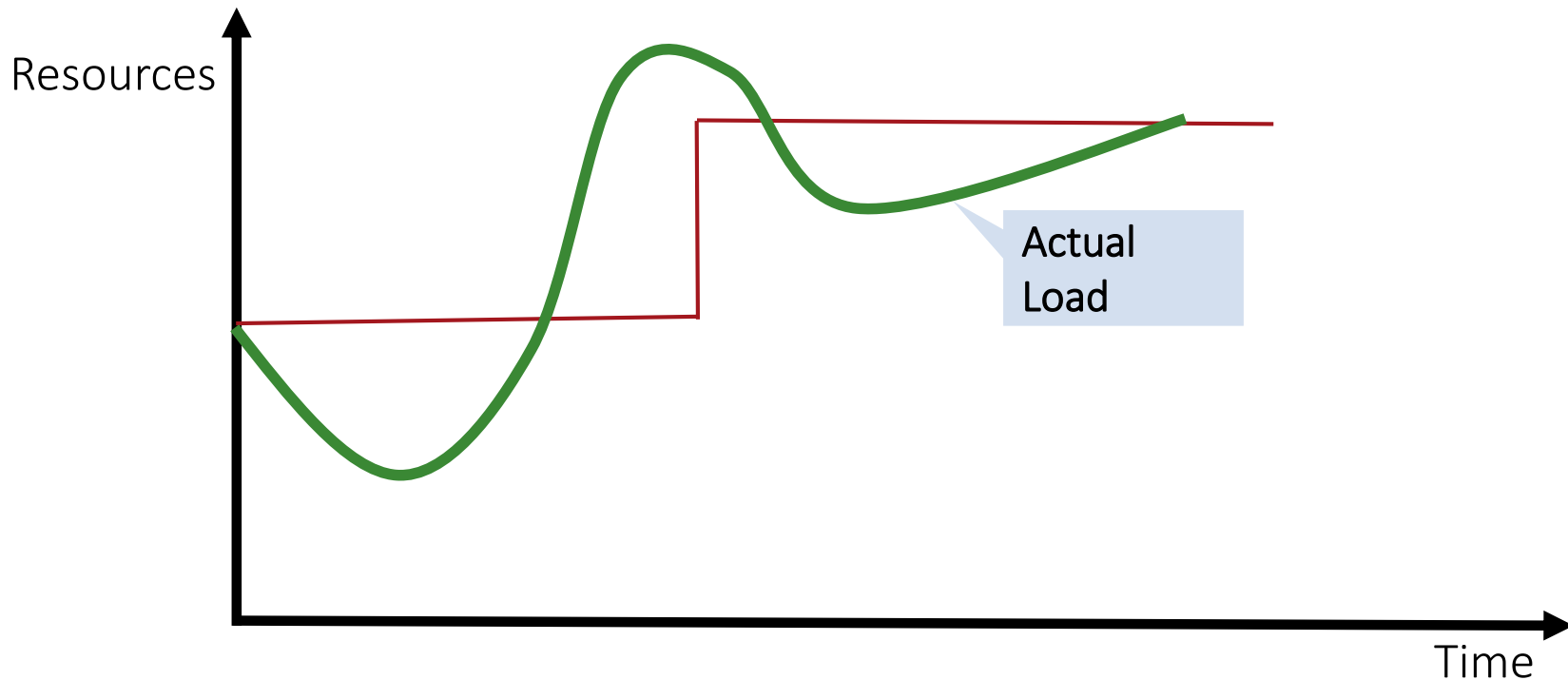
DBaaS: Common Aspects

- ▶ Resource Provisioning
- ▶ Goal: Resources $\Leftrightarrow$ SLAs



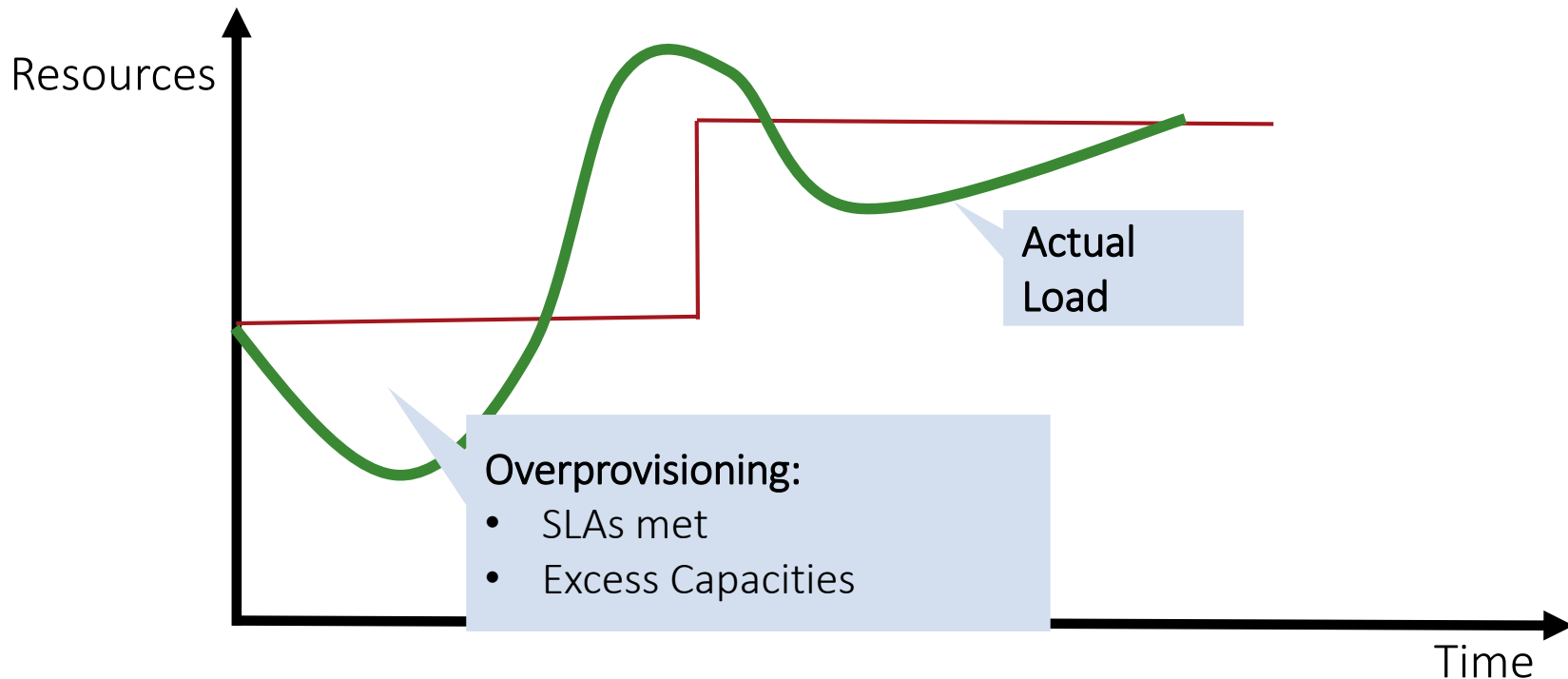
DBaaS: Common Aspects

- ▶ Resource Provisioning
- ▶ Goal: Resources $\Leftrightarrow$ SLAs



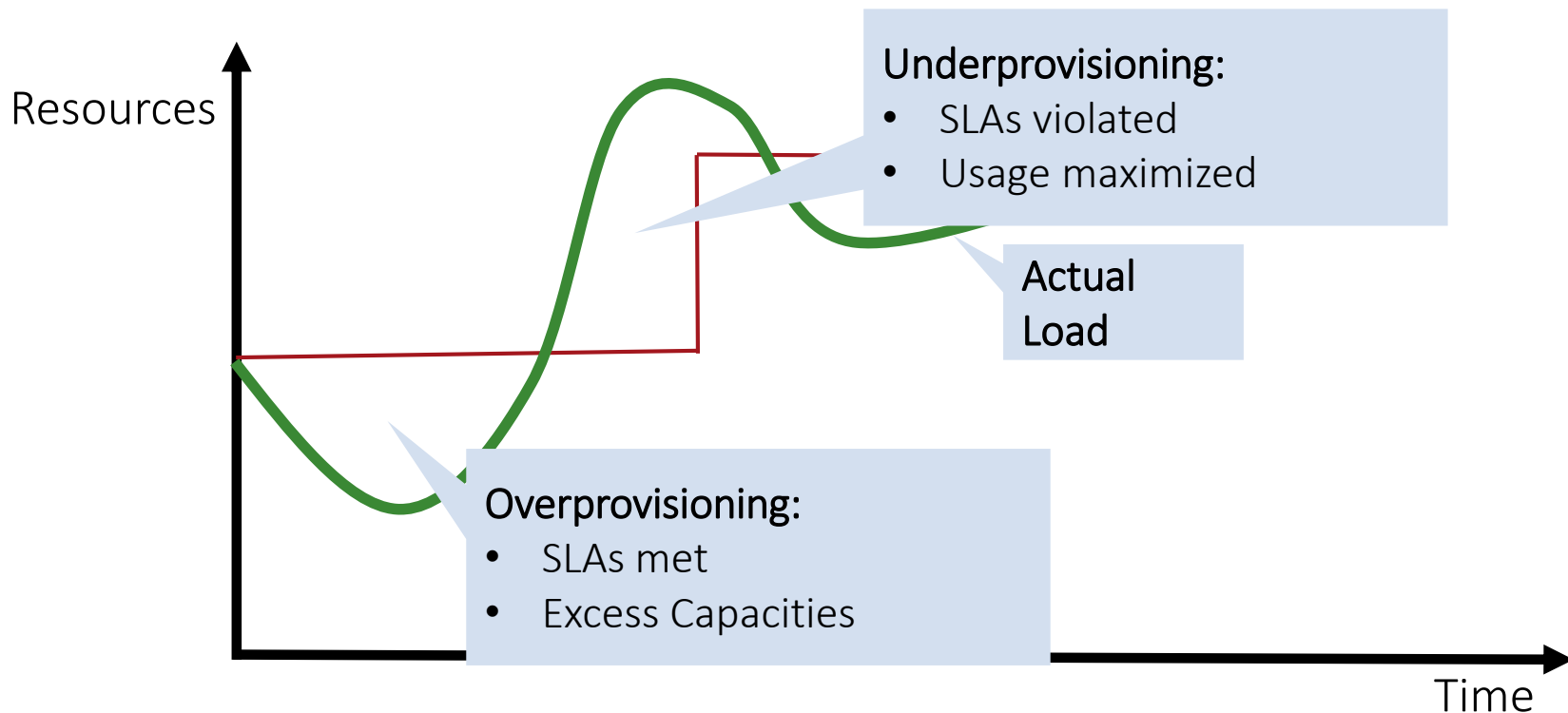
DBaaS: Common Aspects

- ▶ Resource Provisioning
- ▶ Goal: Resources $\Leftrightarrow$ SLAs



DBaaS: Common Aspects

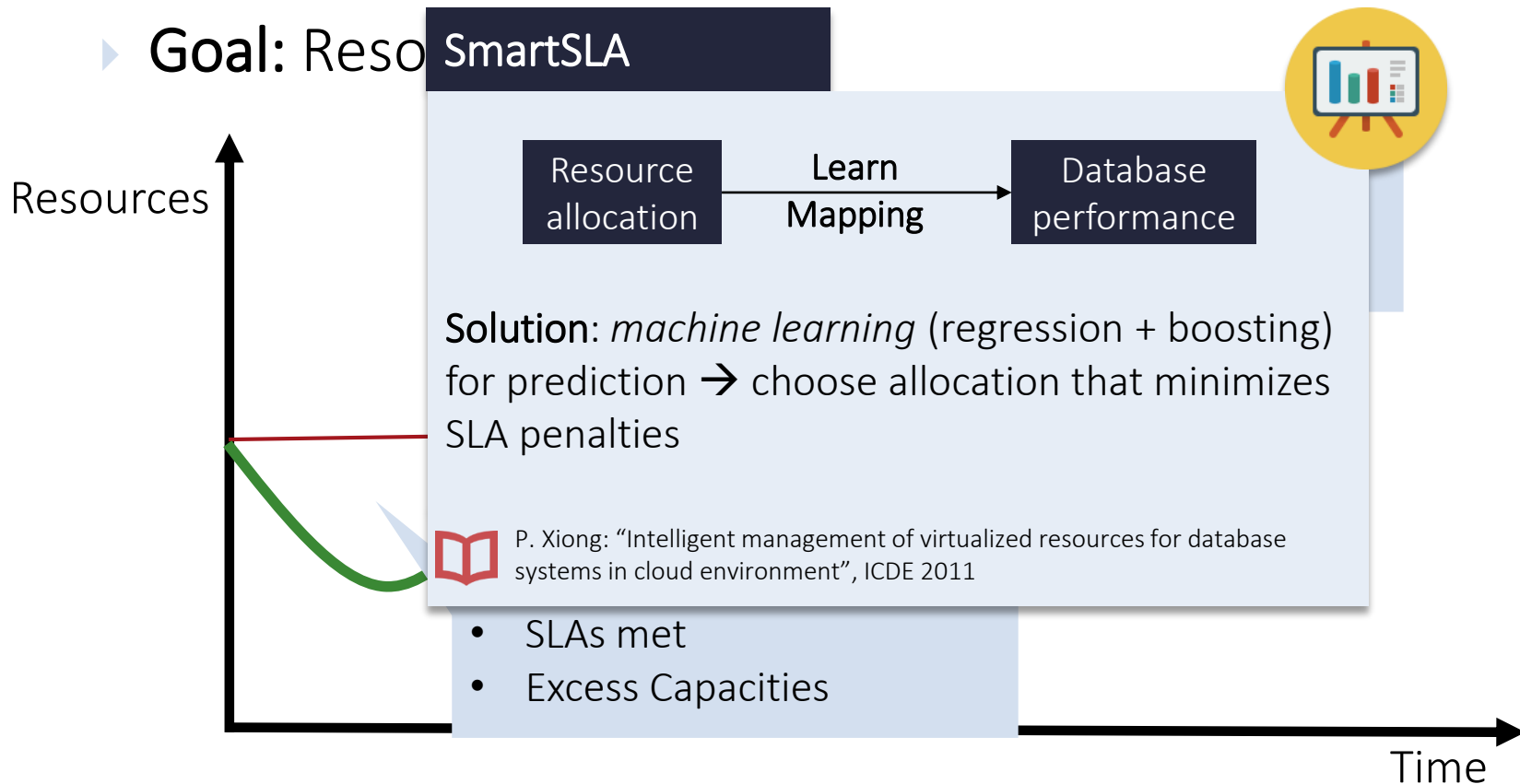
- ▶ Resource Provisioning
- ▶ Goal: Resources $\Leftrightarrow$ SLAs



DBaaS: Common Aspects

▶ Resource Provisioning

▶ Goal: Resource SmartSLA



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

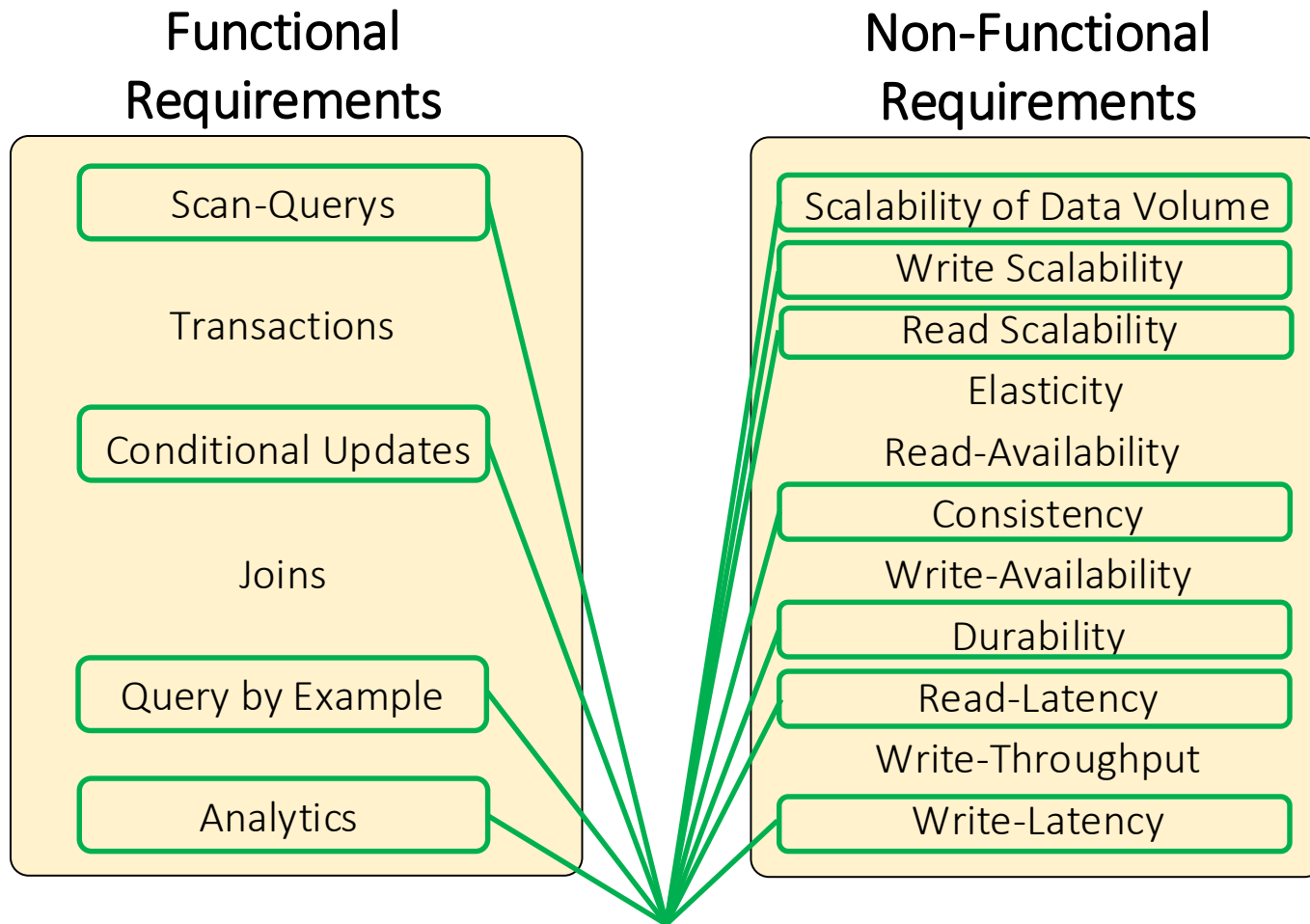
Durability

Read-Latency

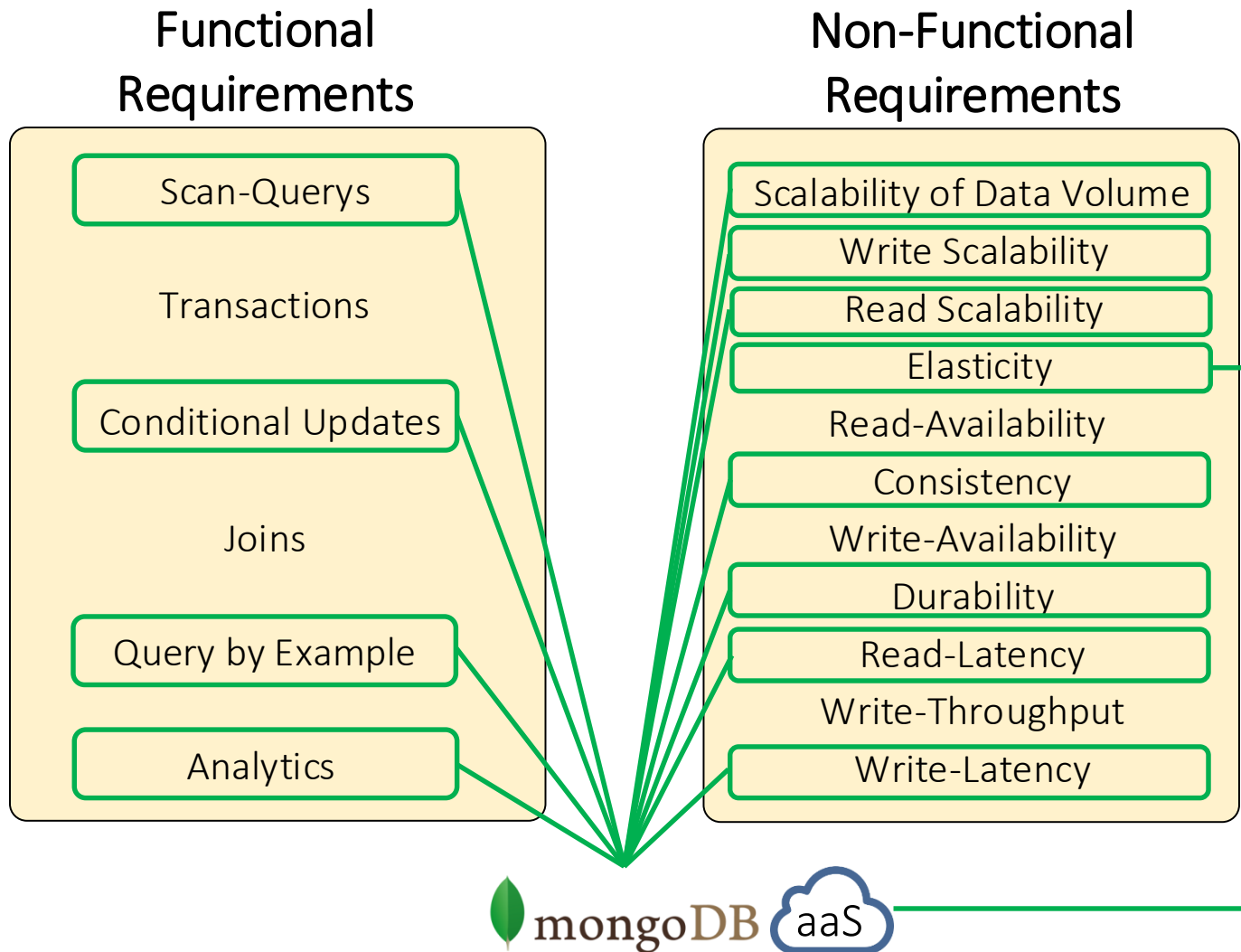
Write-Throughput

Write-Latency

DBaaS: General Considerations



DBaaS: General Considerations



DBaaS: General Considerations

Functional Requirements

Scan-Queryys

Transactions

Conditional Updates

Joins

Non-Functional Requirements

Scalability of Data Volume

Write Scalability

Read Scalability

Elasticity

Read-Availability

Consistency

Write-Availability

Durability

Query by Example

Analytics

Read-Latency

Write-Throughput

Write-Latency



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**

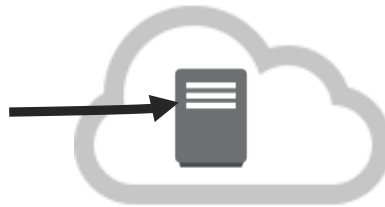
Cloud-Deployed DB

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



- ▶ **Method I: DIY**

1. **Provision VM(s)**



- ▶ **Method II: Deployment Tools**



- ▶ **Method III: Marketplaces**

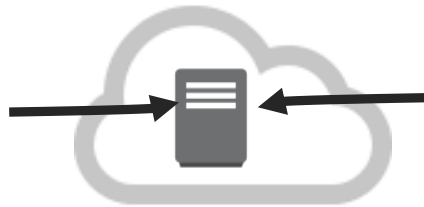
Cloud-Deployed DB

- ▶ **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**

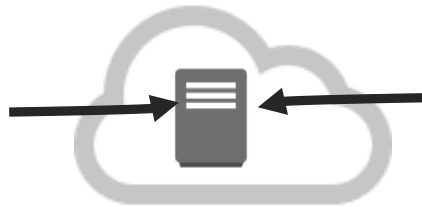
Cloud-Deployed DB

- ▶ **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

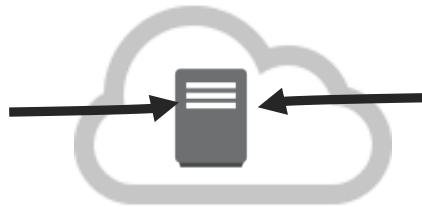
Cloud-Deployed DB

- ▶ **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


Services ▾
Edit ▾

Amazon Web Services

Compute & Networking

-  **Direct Connect**
Dedicated Network Connection to AWS
-  **EC2**
Virtual Servers in the Cloud
-  **Route 53**
Scalable Domain Name System
-  **VPC**
Isolated Cloud Resources
-  **WorkSpaces**
Desktops in the Cloud

Storage & Content Delivery

-  **CloudFront**
Global Content Delivery Network
-  **Glacier**
Archive Storage in the Cloud
-  **S3**
Scalable Storage in the Cloud
-  **Storage Gateway**
Hybrid Cloud Storage

Database

-  **DynamoDB**
Predictable and Scalable NoSQL Data Store
-  **ElastiCache**
In-Memory Cache
-  **RDS**
Managed Relational Database Service
-  **Redshift**
Managed Petabyte-Scale Data Warehouse Service

Deployment & Management

-  **CloudFormation**
Templated AWS Resource Creation
-  **CloudTrail**
User Activity and Change Tracking
-  **CloudWatch**
Resource and Application Monitoring
-  **Elastic Beanstalk**
Managed Environment for PaaS

Analytics

-  **Data Pipeline**
Orchestration for Data-Driven Workflows
-  **Elastic MapReduce**
Managed Hadoop Framework
-  **Kinesis**
Real-time Processing of Streaming Big Data

App Services

-  **AppStream**
Low Latency Application Streaming
-  **CloudSearch**
Managed Search Service
-  **Elastic Transcoder**
Easy-to-use Scalable Media Transcoding
-  **SES**
Email Sending Service
-  **SNS**
Push Notification Service

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

Choosable

API:

DB-specific

AWS Marketplace

- ▶ Idea: Run preconfigured DB on IaaS


Services ▾
Edit ▾

Amazon Web Services

Compute & Networking

-  **Direct Connect**
Dedicated Network Connection to AWS
-  **EC2**
Virtual Servers in the Cloud
-  **Route 53**
Scalable Domain Name System
-  **VPC**
Isolated Cloud Resources
-  **WorkSpaces**
Desktops in the Cloud

Storage & Content Delivery

-  **CloudFront**
Global Content Delivery Network
-  **Glacier**
Archive Storage in the Cloud
-  **S3**
Scalable Storage in the Cloud
-  **Storage Gateway**
Hybrid Cloud Storage

Database

-  **DynamoDB**
Predictable and Scalable NoSQL Data Store
-  **ElastiCache**
In-Memory Cache
-  **RDS**
Managed Relational Database Service
-  **Redshift**
Managed Petabyte-Scale Data Warehouse Service

Deployment & Management

-  **CloudFormation**
Templated AWS Resource Creation
-  **CloudTrail**
User Activity and Change Tracking
-  **CloudWatch**
Resource and Application Monitoring
-  **Elastic Beanstalk**
Managed Java, .NET, PHP, and Ruby Applications

Analytics

-  **Data Pipeline**
Orchestration for Data-Driven Workflows
-  **Elastic MapReduce**
Managed Hadoop Framework
-  **Kinesis**
Real-time Processing of Streaming Big Data

App Services

-  **AppStream**
Low Latency Application Streaming
-  **CloudSearch**
Managed Search Service
-  **Elastic Transcoder**
Easy-to-use Scalable Media Transcoding
-  **SES**
Email Sending Service
-  **SNS**
Push Notification Service

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

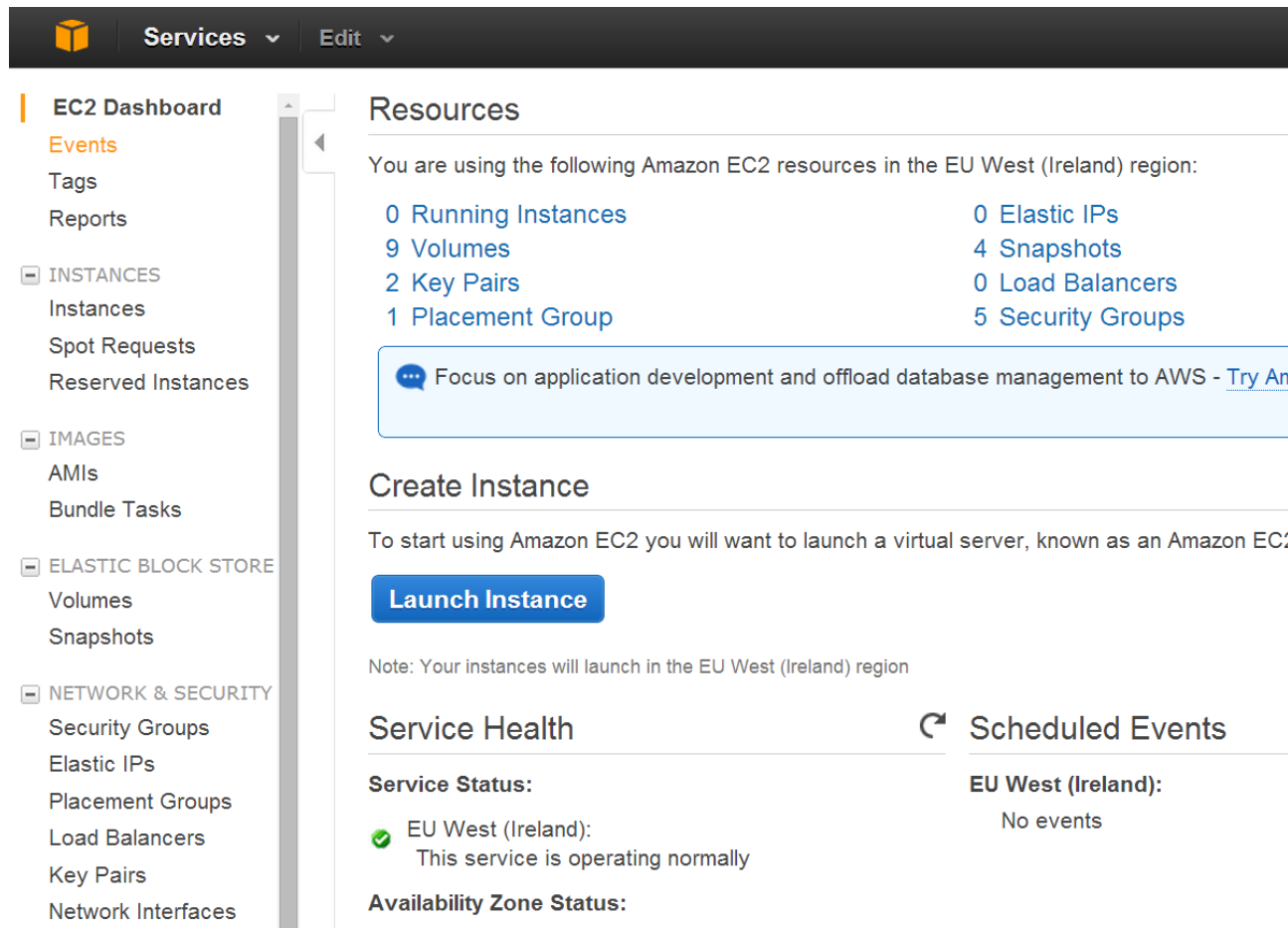
Choosable

API:

DB-specific

AWS Marketplace

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



Services ▾ **Edit** ▾

EC2 Dashboard

- Events
- Tags
- Reports
- INSTANCES
 - Instances
 - Spot Requests
 - Reserved Instances
- IMAGES
 - AMIs
 - Bundle Tasks
- ELASTIC BLOCK STORE
 - Volumes
 - Snapshots
- NETWORK & SECURITY
 - Security Groups
 - Elastic IPs
 - Placement Groups
 - Load Balancers
 - Key Pairs
 - Network Interfaces

Resources

You are using the following Amazon EC2 resources in the EU West (Ireland) region:

- 0 Running Instances
- 9 Volumes
- 2 Key Pairs
- 1 Placement Group
- 0 Elastic IPs
- 4 Snapshots
- 0 Load Balancers
- 5 Security Groups

Focus on application development and offload database management to AWS - [Try Amazon RDS](#)

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the EU West (Ireland) region

Service Health

Service Status:

- EU West (Ireland): This service is operating normally

Availability Zone Status:

Scheduled Events

EU West (Ireland):

- No events

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

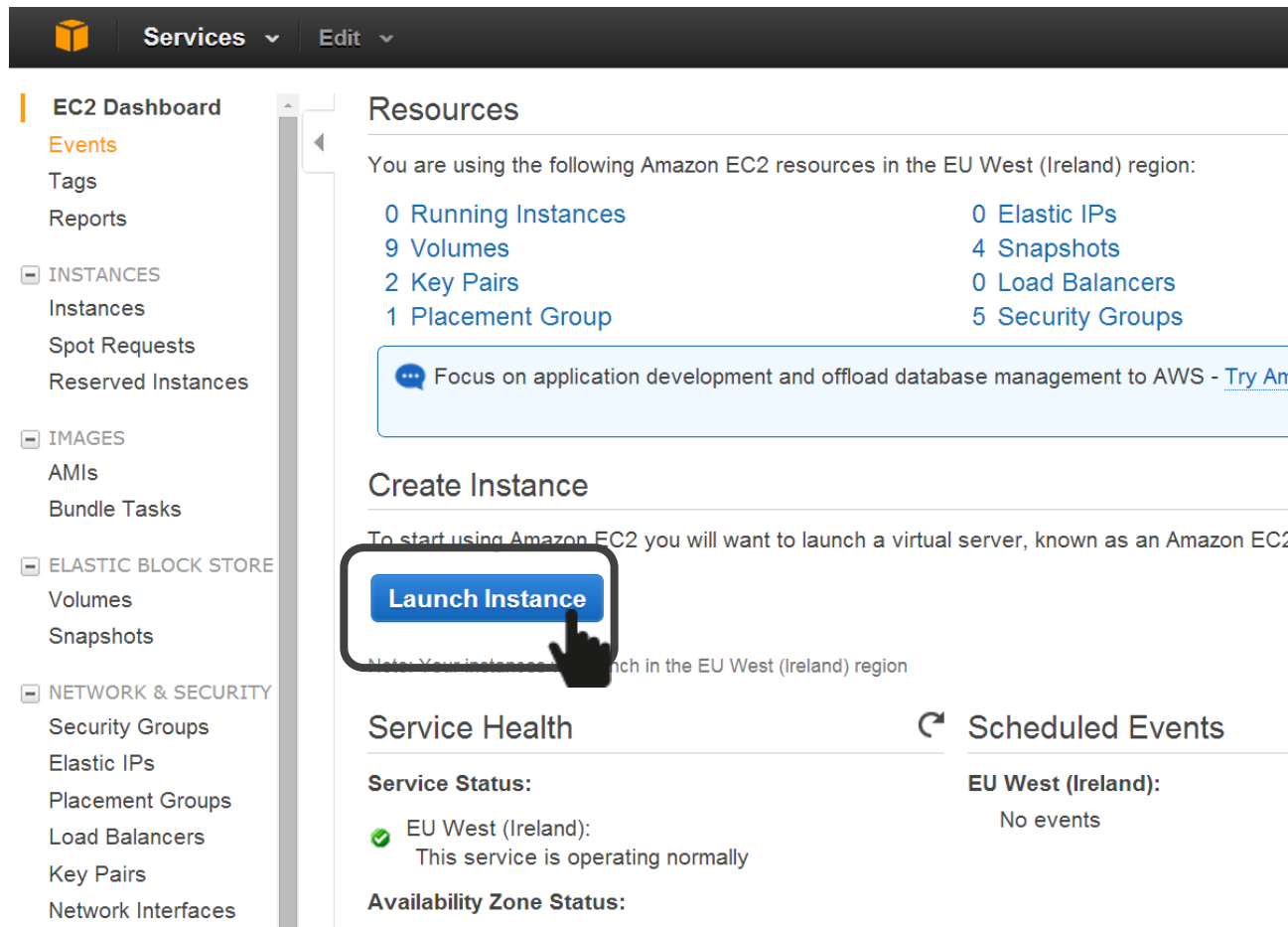
Choosable

API:

DB-specific

AWS Marketplace

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



The screenshot shows the AWS Management Console interface. On the left is a navigation sidebar with categories like EC2 Dashboard, INSTANCES, IMAGES, ELASTIC BLOCK STORE, and NETWORK & SECURITY. The main content area is titled 'Resources' and lists EC2 resources in the EU West (Ireland) region: 0 Running Instances, 9 Volumes, 2 Key Pairs, 1 Placement Group, 0 Elastic IPs, 4 Snapshots, 0 Load Balancers, and 5 Security Groups. Below this is a 'Create Instance' section with a 'Launch Instance' button highlighted by a red box and a hand cursor. At the bottom, there are sections for 'Service Health' (showing EU West (Ireland) is operating normally) and 'Scheduled Events' (showing no events).

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

Choosable

API:

DB-specific

AWS Marketplace

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


Services ▾
Edit ▾

1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance

Step 1: Choose an Amazon Machine Image (AMI) [Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs

×

1 to 14 of 14 Products



MongoDB 2.4 with 4000 IOPS
Select

★★★★★ (1) | 2.4.9
Previous versions | Sold by MongoDB

\$0.00/hr for software + AWS usage fees

Linux/Unix, Amazon Linux 5.5 | 64-bit Amazon Machine Image (AMI) | Updated: 1/20/14

MongoDB for AWS Marketplace

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

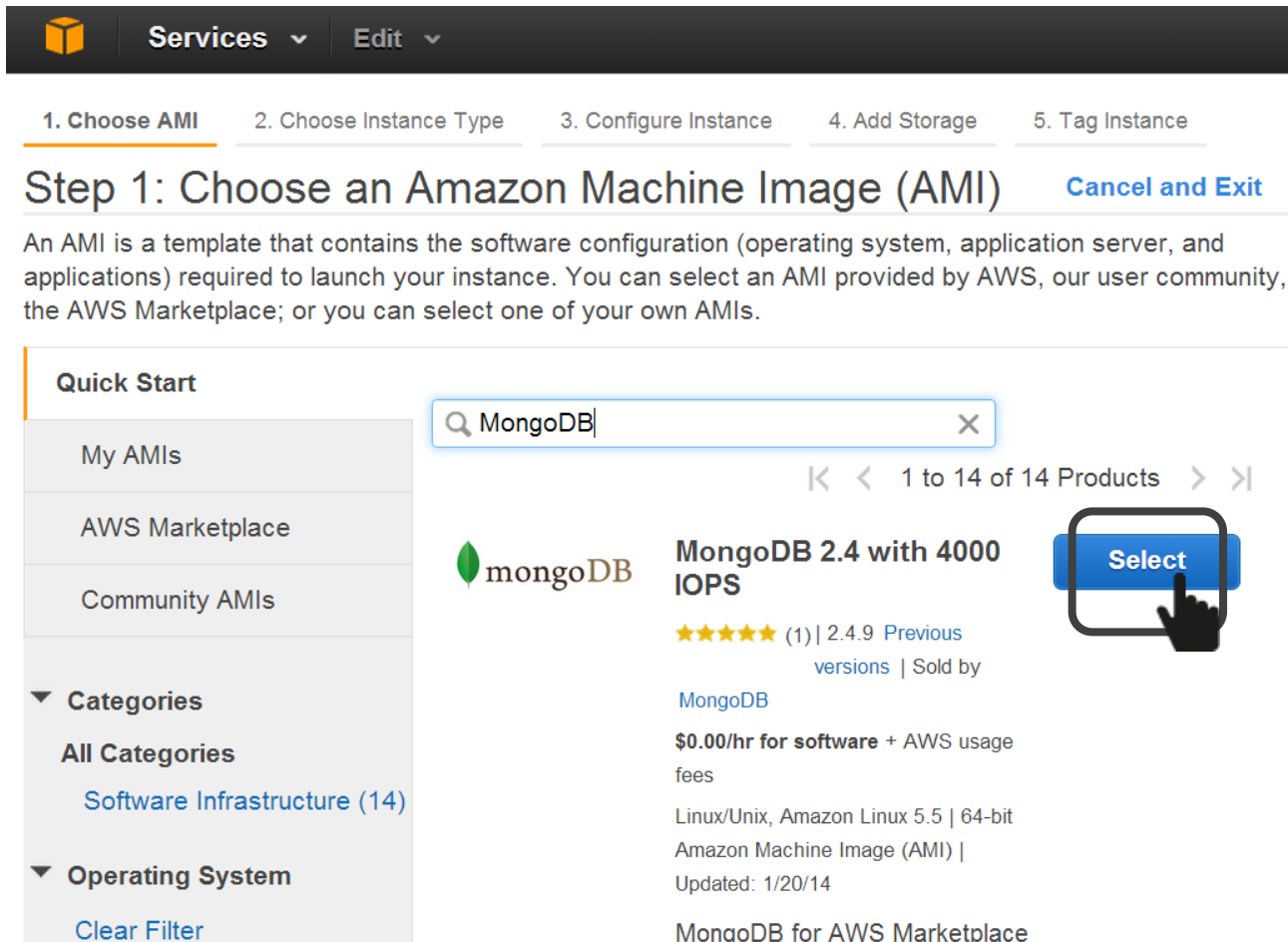
Choosable

API:

DB-specific

AWS Marketplace

- ▶ Idea: Run preconfigured DB on IaaS



Services ▾ **Edit** ▾

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance

Step 1: Choose an Amazon Machine Image (AMI) [Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs

▼ **Categories**

All Categories

- Software Infrastructure (14)

▼ **Operating System**

[Clear Filter](#)

1 to 14 of 14 Products



MongoDB 2.4 with 4000 IOPS

★★★★★ (1) | 2.4.9 [Previous versions](#) | Sold by [MongoDB](#)

\$0.00/hr for software + AWS usage fees

Linux/Unix, Amazon Linux 5.5 | 64-bit Amazon Machine Image (AMI) | Updated: 1/20/14

MongoDB for AWS Marketplace

Select

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

Choosable

API:

DB-specific

AWS Marketplace

- ▶ Idea: Run preconfigured DB on IaaS


Services
Edit
Feli

1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance

Step 2: Choose an Instance Type

Note: The vendor recommends using a m1.large instance (or larger) for the best experience with this product.

	Family	Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)
	Micro instances Free tier eligible	t1.micro	up to 2	1	0.613	EBS only
	General purpose	m3.medium	3	1	3.75	1 x 4 (SSD)
	General purpose	m3.large	6.5	2	7.5	1 x 32 (SSD)
☐	General purpose	m3.xlarge	13	4	15	2 x 40 (SSD)
☐	General purpose	m3.2xlarge	26	8	30	2 x 80 (SSD)
	General purpose	m1.small	1	1	1.7	1 x 160
	General purpose	m1.medium	2	1	3.7	1 x 410

Cancel
Previous
Review and Launch
Next: Configure Instance Details

AWS Marketplace

Model:

Cloud-Deployed

Pricing:

Instance +
Volume +
License

Underlying DB:

Choosable

API:

DB-specific

AWS Marketplace

- ▶ Idea: Run preconfigured DB on IaaS


Services
Edit
Feli

1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance

Step 2: Choose an Instance Type

Note: The vendor recommends using a **m1.large** instance (or larger) for the best experience with this product.

	Family	Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)
	Micro instances Free tier eligible	t1.micro	up to 2	1	0.613	EBS only
	General purpose	m3.medium	3	1	3.75	1 x 4 (SSD)
	General purpose	m3.large	4	2	7.5	1 x 32 (SSD)
	General purpose	m3.xlarge	8	4	15	2 x 40 (SSD)
	General purpose	m3.2xlarge	16	8	30	4 x 40 (SSD)
	Memory optimized	m1.small	1	1	1.7	1 x 160
	Memory optimized	m1.medium	1	1	3.7	1 x 410

Cancel
Previous
Review and Launch
Next: Configure Instance Details

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:

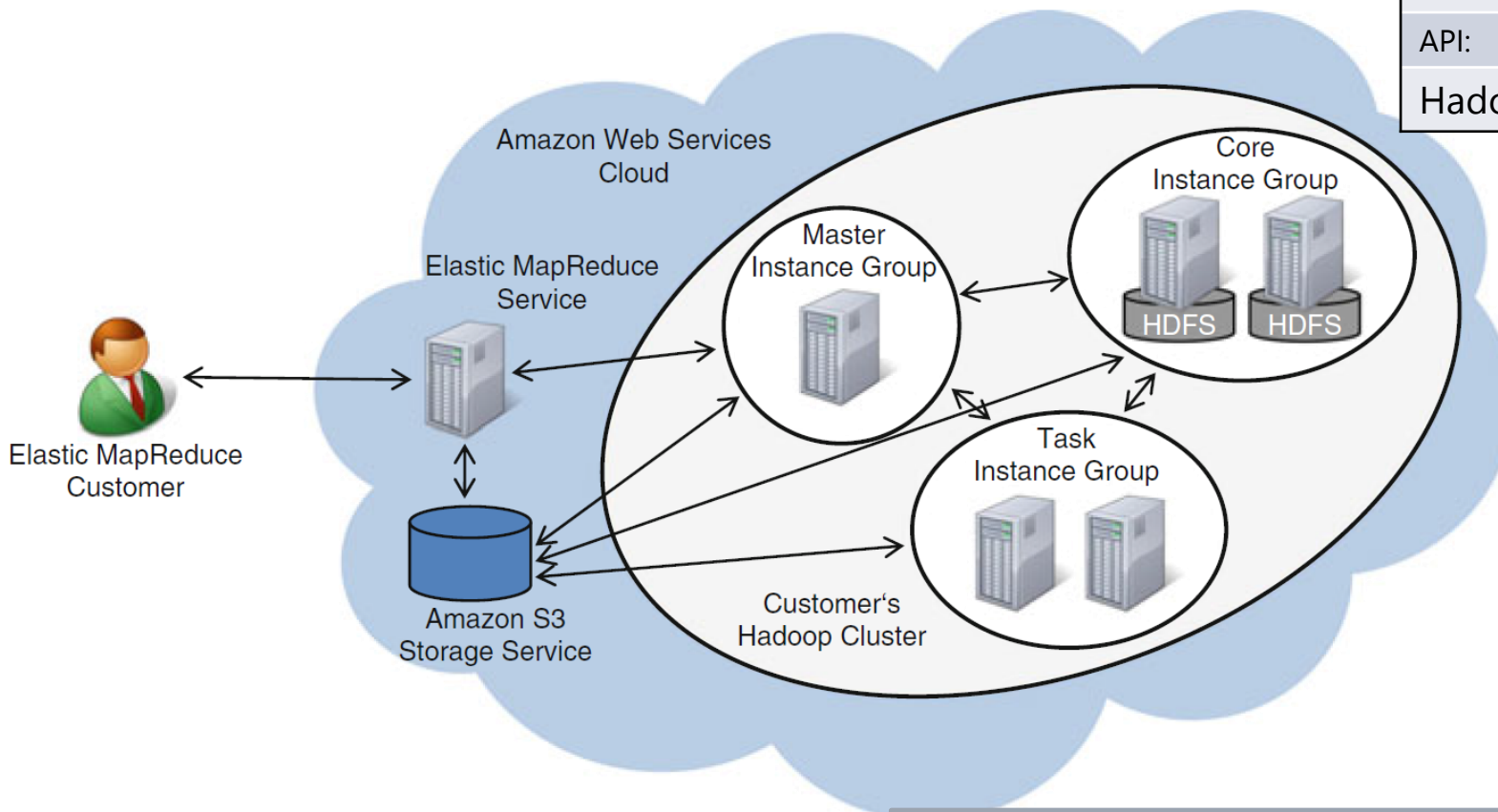
Analytics-aaS

Pricing:

Infrastructure

API:

Hadoop



Amazon Elastic MapReduce

EMR

Model:

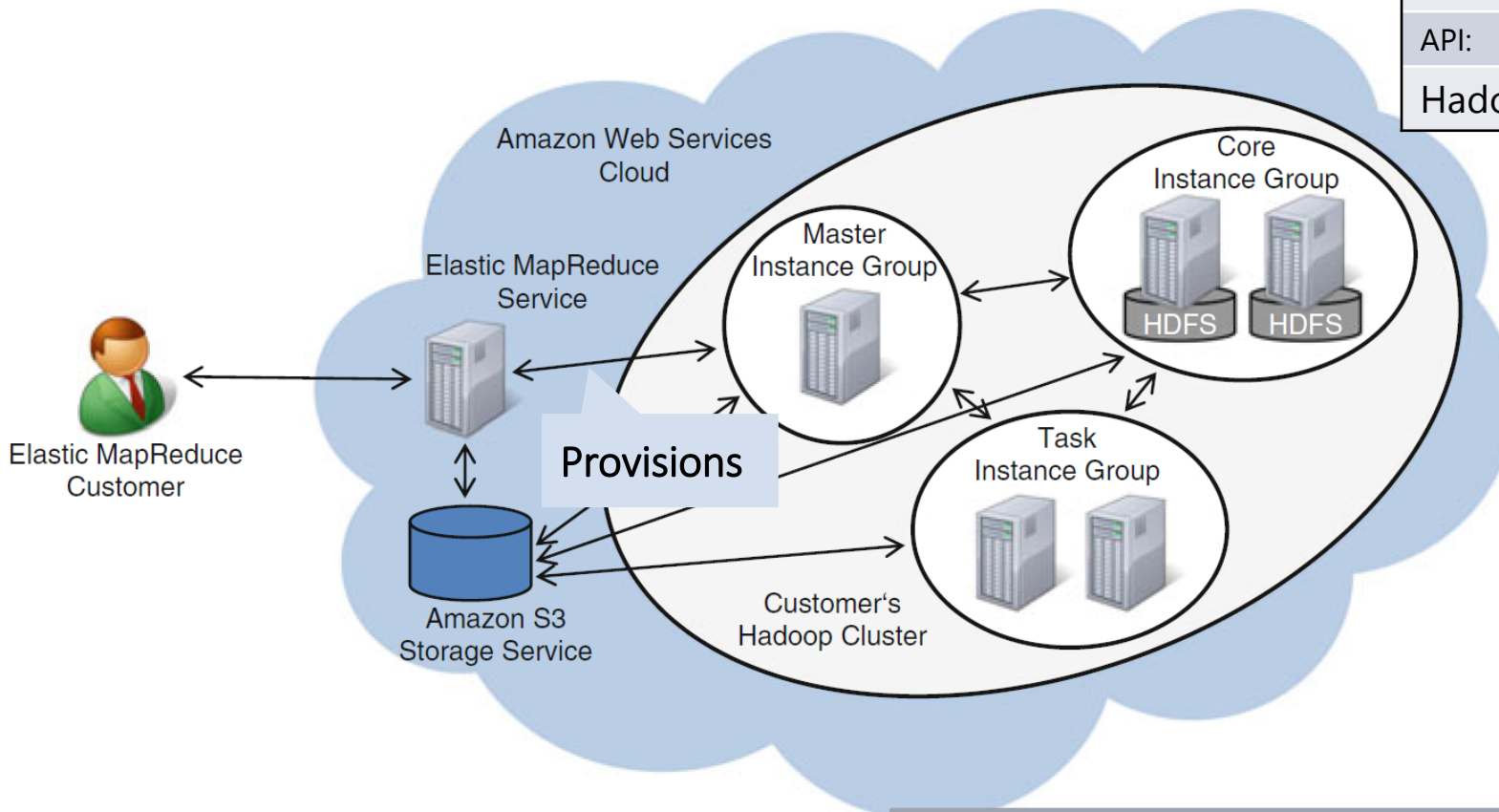
Analytics-aaS

Pricing:

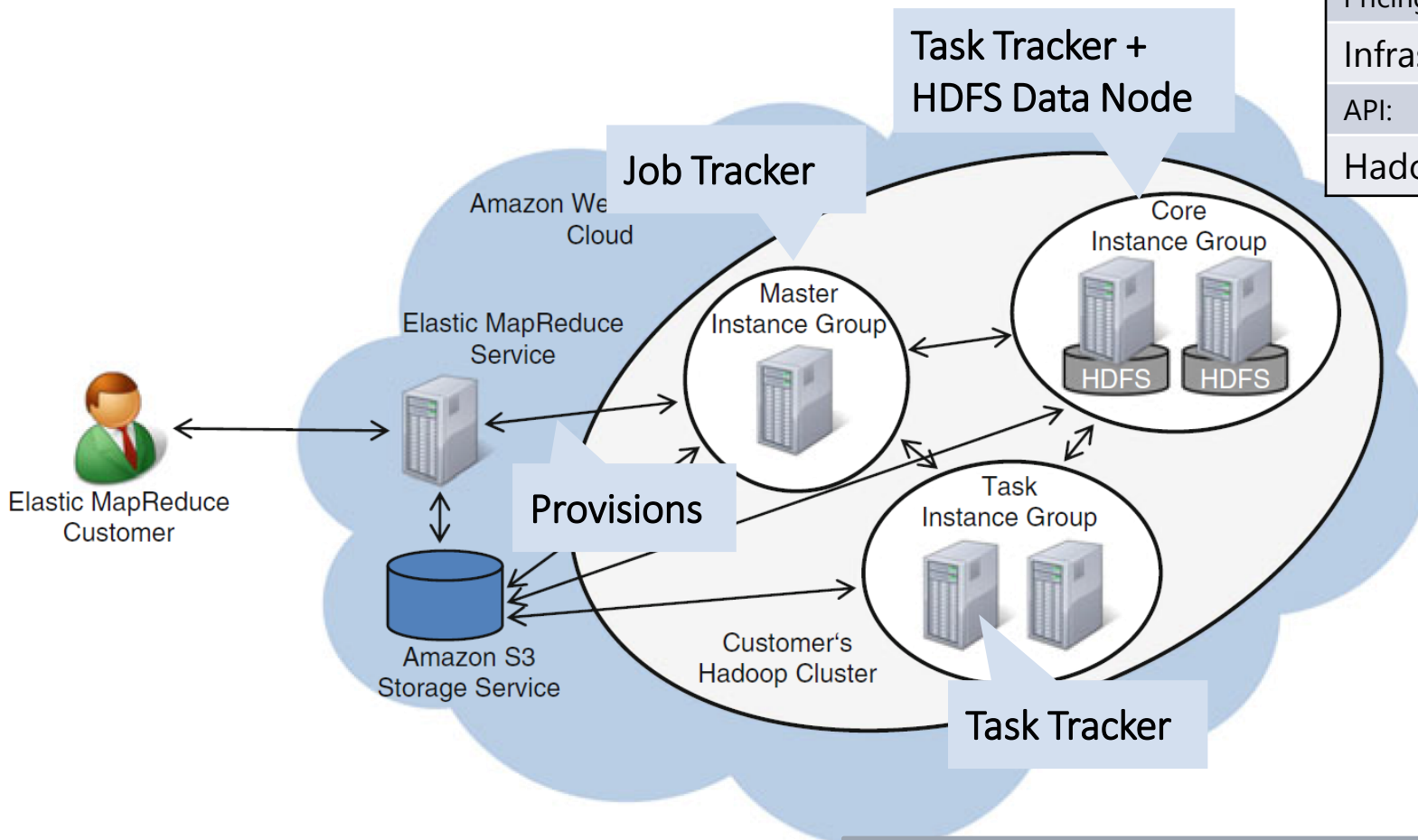
Infrastructure

API:

Hadoop



Amazon Elastic MapReduce



EMR

Model:

Analytics-aaS

Pricing:

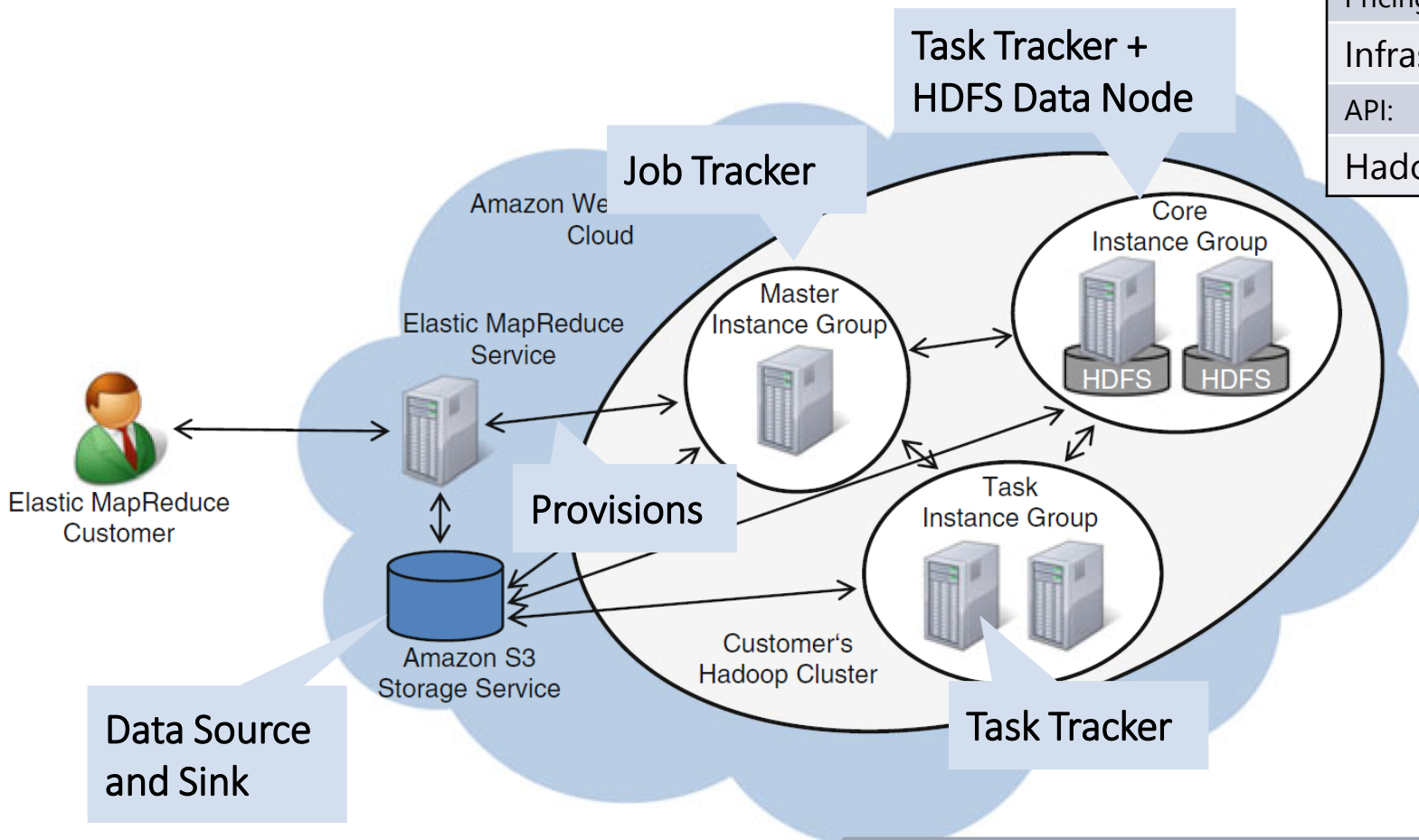
Infrastructure

API:

Hadoop



Amazon Elastic MapReduce



EMR

Model:

Analytics-aaS

Pricing:

Infrastructure

API:

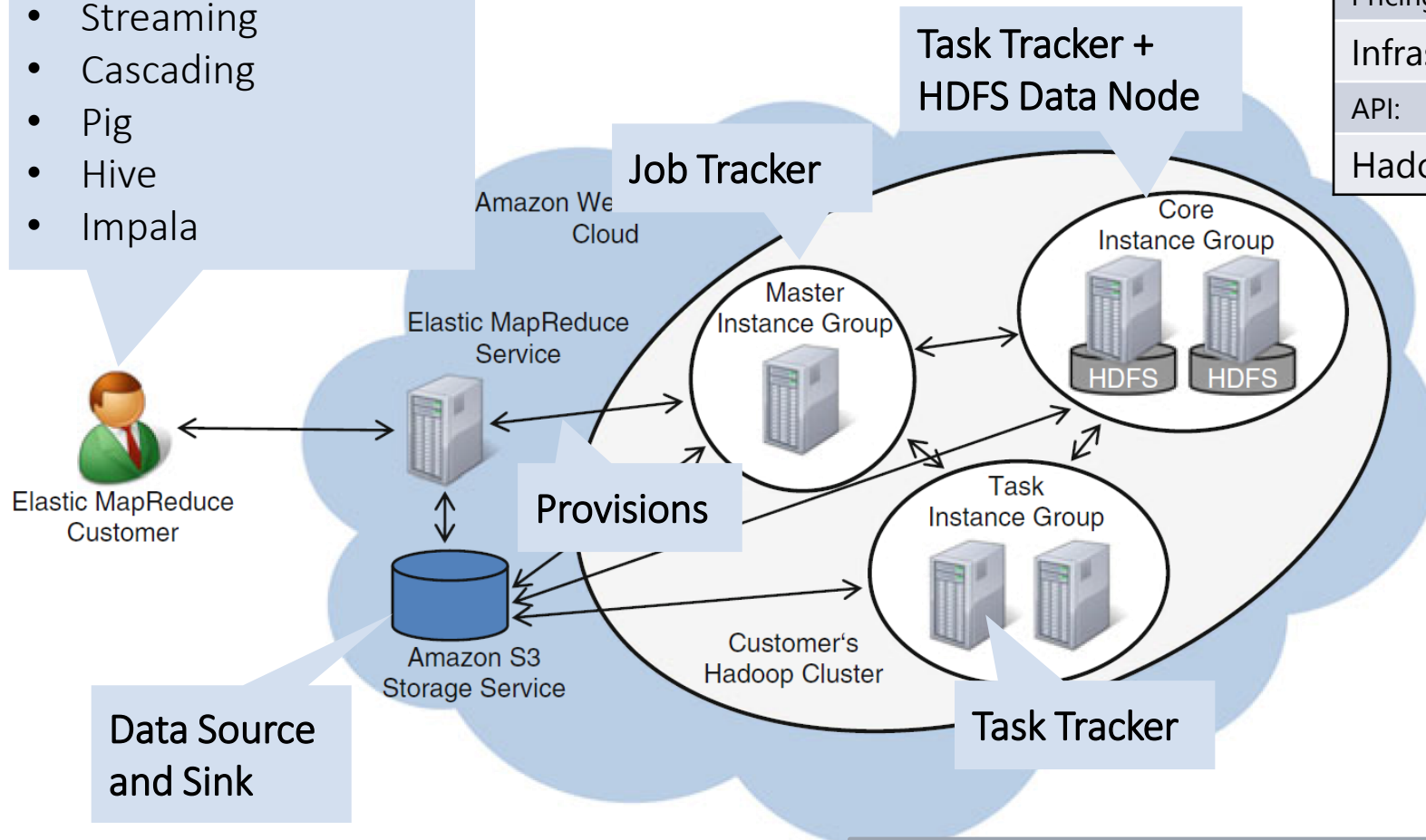
Hadoop



Amazon Elastic MapReduce

Submits Hadoop Jobs as:

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



EMR

Model:

Analytics-aaS

Pricing:

Infrastructure

API:

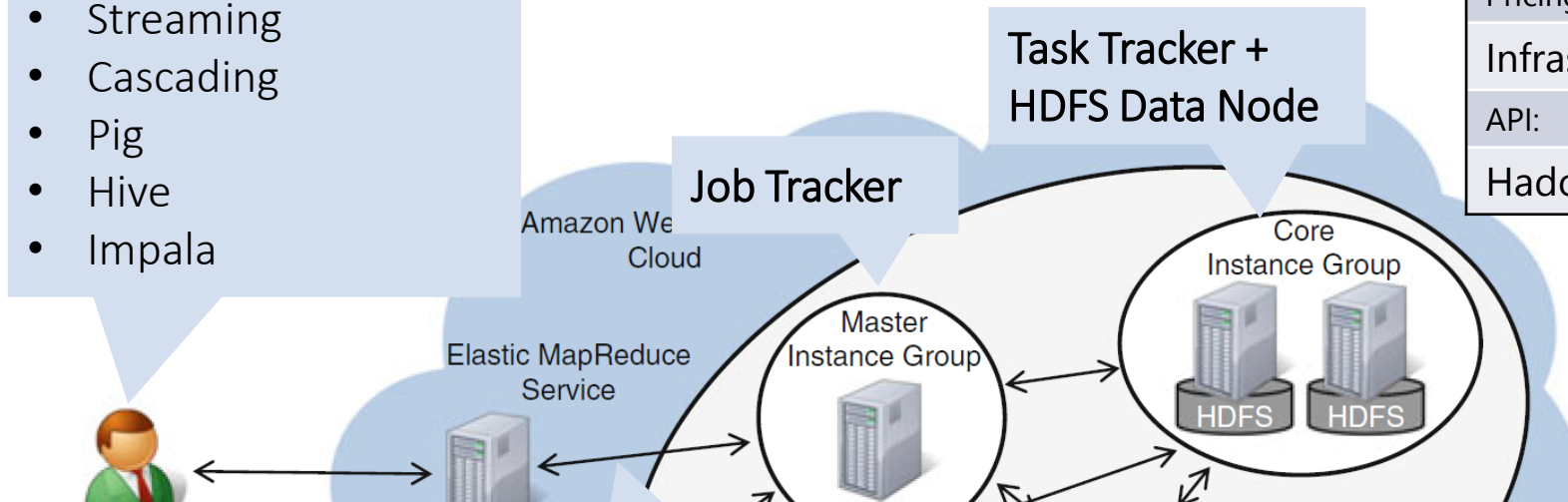
Hadoop



Amazon Elastic MapReduce

Submits Hadoop Jobs as:

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



EMR

Model:

Analytics-aaS

Pricing:

Infrastructure

API:

Hadoop

Elastic MapReduce
Customer



- 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

Suche Bilder Maps Play YouTube News Gmail Drive Mehr -
Felix Gessert -

Google bigquery

COMPOSE QUERY

Query History
Job History

API Project

No datasets found in this project.
Please create a dataset or select a new project from the menu above.

► publicdata:samples

New Query

```

1 SELECT TOP(title, 5), COUNT(*)
2 FROM [publicdata:samples.wikipedia]
3 WHERE title CONTAINS "data";

```

RUN QUERY Save Query Save View Enable Options

Query complete (2.4s elapsed, 6.79 GB processed)

Query Results 6:11pm, 25 Apr 2014 Download as CSV Save as Table

Row	f0_	f1_	
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	

BigQuery

Model:

Analytics-aaS

Pricing:

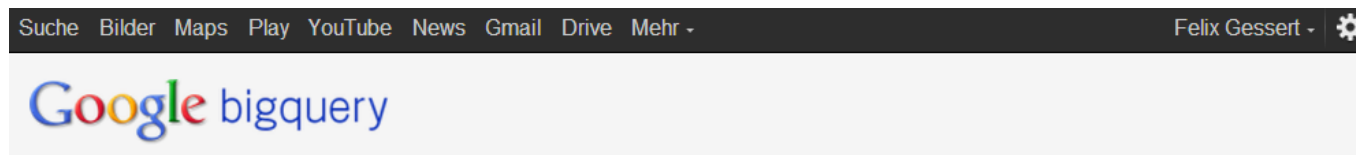
Storage + GBs
Processed

API:

REST

Google BigQuery

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



BigQuery

Model:

Analytics-aaS

Pricing:

Storage + GBs
Processed

API:

REST

New Query

```
1 SELECT TOP(title, 5), COUNT(*)
2 FROM [publicdata:samples.wikipedia]
3 WHERE title CONTAINS "data";
```

RUN QUERY

Save Query

Save View

Enable Options

Query complete (2.4s elapsed, 6.79 GB processed)



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

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

BigQuery

Model:

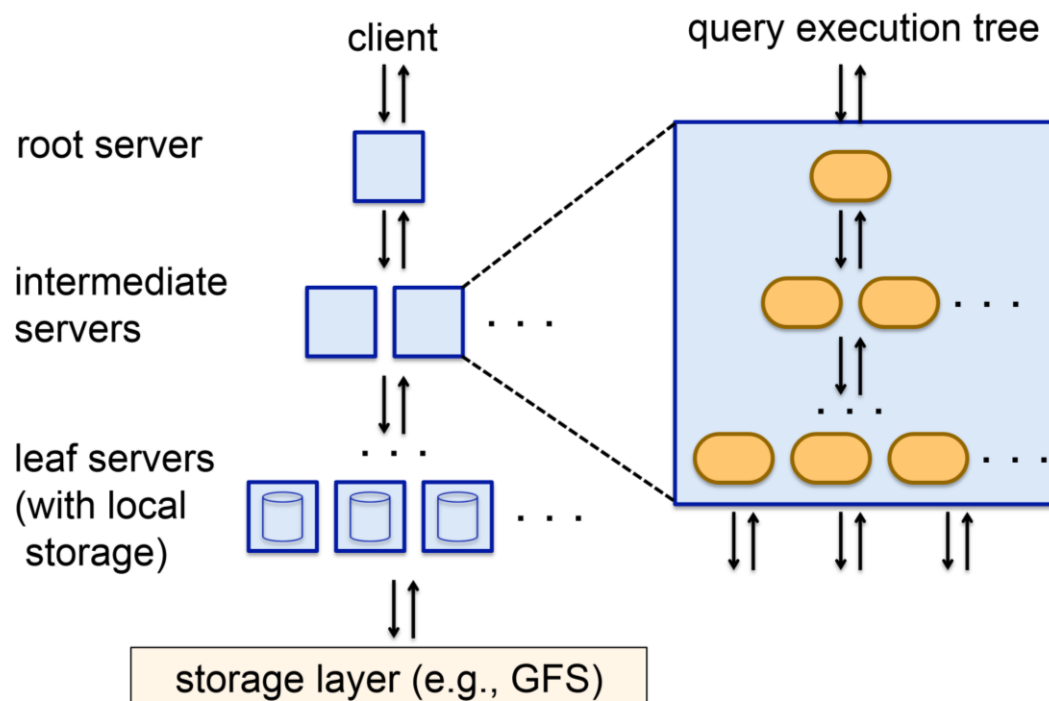
Analytics-asS

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

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

BigQuery

Model:

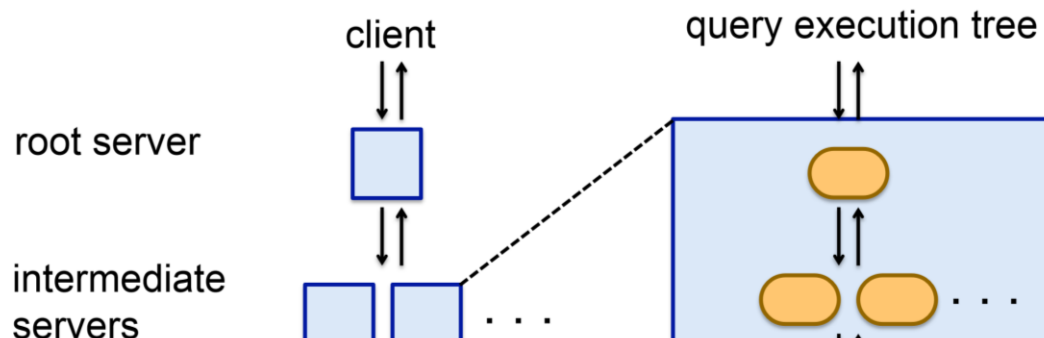
Analytics-asS

Pricing:

Storage + GBs
Processed

API:

REST



Dremel

Idea:

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



Amazon RDS

► Relational Database Service


Services ▾ Edit ▾

Felix Gessert ▾ Ireland ▾

Step 1: Engine Selection

Engine Selection

To get started, choose the DB Instance details below and click Select

	mysql MySQL Community Edition	Select ▶
	postgres PostgreSQL	Select ▶
	oracle-se1 Oracle Database Standard Edition One	Select ▶
	oracle-se Oracle Database Standard Edition	Select ▶
	oracle-ee Oracle Database Enterprise Edition	Select ▶
	sqlserver-ex Microsoft SQL Server Express Edition Note that SQL Server Express Edition limits the storage of per database to a maximum of 10GB. Refer to this link for more details.	Select ▶
	sqlserver-web Microsoft SQL Server Web Edition Note that in accordance with Microsofts licensing policies, SQL Server Web Edition can only be used to support public and internet accessible	Select ▶

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume
+ License

Underlying DB:

MySQL, Postgres,
MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service

 Services ▾ Edit ▾ Felix Gessert ▾ Ireland ▾ Help

[Step 1: Engine Selection](#)
Step 2: Production?
[Step 3: DB Instance Details](#)
[Step 4: Additional Config](#)
[Step 5: Management Options](#)
[Step 6: Review](#)

Do you plan to use this database for production purposes?

 For databases used in production or pre-production we recommend:

- **Multi-AZ Deployment** for high availability (99.95% monthly up time **SLA**)
- **Provisioned IOPS Storage** for fast, consistent performance

Billing is based upon the **RDS pricing table**.
An instance which uses these features is not eligible for the **RDS Free Usage Tier**.

☒ Yes, use **Multi-AZ Deployment** and **Provisioned IOPS Storage** as defaults while creating this instance

☐ No, this instance is intended for use outside of production or under the **RDS Free Usage Tier**

[Cancel](#) [Previous](#) [Next Step](#)

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

Felix Gessert ▾ Ireland ▾ Help

Do you want to use this database for production purposes?

For databases used in production or pre-production we recommend:

- **Multi-AZ Deployment** for high availability (99.95% monthly up time **SLA**)
- **Provisioned IOPS Storage** for fast, consistent performance

Billing is based upon the **RDS pricing table**.
An instance which uses these features is not eligible for the **RDS Free Usage Tier**.

☒ Yes, use **Multi-AZ Deployment** and **Provisioned IOPS Storage** as defaults while creating this instance

☐ No, this instance is intended for use outside of production or under the **RDS Free Usage Tier**

Cancel Previous **Next Step**

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

Step 3: DB Instance Details

Step 4: Additional Config

Step 5: Management Options

Step 6: Review

For databases used in production or pre-production we recommend:

- **Multi-AZ Deployment** for high availability (99.95% monthly up time **SLA**)
- **Provisioned IOPS Storage** for fast, consistent performance

Billing is based upon the **RDS pricing table**.

An instance which uses these features is not eligible for the **RDS Free Usage Tier**.

☒ Yes, use **Multi-AZ Deployment** and **Provisioned IOPS Storage** as defaults while creating this instance

☐ No, this instance is intended for use outside of production or under the **RDS Free Usage Tier**

Cancel

Previous

Next Step

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

Step 3: DB Instance Details

Step 4: Additional Config

Step 5: Management Options

Step 6: Review

For databases used in production or pre-production we recommend:

- **Multi-AZ Deployment** for high availability (99.95% monthly up time **SLA**)
- **Provisioned IOPS Storage** for fast, consistent performance

Billing is based upon the **RDS pricing table**.
An instance which uses these features is not eligible for the **RDS Free Usage Tier**.

☒ Yes, use **Multi-AZ Deployment** and **Provisioned IOPS Storage** as defaults while creating this

use outside of production or under the **RDS Free Usage Tier**

Cancel

Previous

Next Step

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

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume
+ License

Underlying DB:

MySQL, Postgres,
MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service


Services
Edit

Step 1: Engine Selection
Step 2: Production?
Step 3: DB Instance Details
Step 4: Additional Config
Step 5: Management Options
Step 6: Review

DB Instance Details

To get started, choose a DB engine below and click Next Step

DB Engine: mysql

License Model: general-public-license

DB Engine Version: 5.6.13

DB Instance Class: db.m3.xlarge

Multi-AZ Deployment: - Select One -

Auto Minor Version Upgrade:

Provide the details for your RDS Database Instance

Allocated Storage:*

Use Provisioned IOPS:

DB Instance Identifier:*

Master Username:*

Master Password:*

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume
+ License

Underlying DB:

MySQL, Postgres,
MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service


Services ▼ Edit ▼

Step 1: [Engine Selection](#)

Step 2: [Production?](#)

Step 3: DB Instance Details

Step 4: [Additional Config](#)

Step 5: [Management Options](#)

Step 6: [Review](#)

DB Instance Details

To get started, choose a DB engine below and click Next Step

DB Engine: mysql

License Model: general-public-license ▼

DB Engine Version: 5.6.13 ▼

DB Instance Class: db.m3.xlarge ▼

Multi-AZ Deployment: - Select One -

Auto Minor Version Upgrade: db.t1.micro

Allocated Storage:* db.m1.small

Use Provisioned IOPS: db.m1.medium

DB Instance Identifier:* db.m1.large

Master Username:* db.m1.xlarge

Master Password:* db.m2.xlarge

(e.g. mydbinstance)

(e.g. awsuser)

(e.g. mypassword)

Provide the details for your RDS Database Instance

(e.g. 5 GB, Maximum: 3072 GB) Higher allocated storage [may improve](#) performance.

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

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume
+ License

Underlying DB:

MySQL, Postgres,
MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service


Services ▾ Edit ▾

Step 1: Engine Selection
Step 2: Production?
Step 6: Review

DB Instance Details

To get started, choose a DB engine below and click Next Step

Engine: mysql

Model: general-public-license ▾

Version: 5.6.13 ▾

Instance Class: db.m3.xlarge ▾

Multi-AZ Deployment: - Select One -

Auto Minor Version Upgrade: ☐

Allocated Storage*: 5 GB, Maximum: 3072 GB) Higher allocated storage [may improve](#) performance.

Use Provisioned IOPS: ☐

DB Instance Identifier*: (e.g. mydbinstance)

Master Username*: (e.g. awsuser)

Master Password*: (e.g. mypassword)

db.m3.xlarge

db.m3.2xlarge

db.cr1.8xlarge

Minor Version Upgrades are performed without downtime

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

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume + License

Underlying DB:

MySQL, Postgres, MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service

 Services ▼ Edit ▼

Step 1: [Engine Selection](#)

Step 2: [Production?](#)

Step 3: [DB Instance Details](#)

Step 4: [Additional Config](#)

Step 5: Management Options

Step 6: [Review](#)

Management Options

Enable Automatic Backups: ☒ Yes ☐ No

The number of days for which automated backups are retained.

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail [here](#).

Backup Retention Period: days

The daily time range during which automated backups are created if automated backups are enabled

Backup Window: ☐ Select Window ☒ No Preference

The weekly time range (in UTC) during which system maintenance can occur.

Maintenance Window: ☐ Select Window ☒ No Preference

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume
+ License

Underlying DB:

MySQL, Postgres,
MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service

 Services ▼ Edit ▼

Step 1: Engine Selection

Step 2: Production?

Step 3: DB Instance Details

Step 4: Additional Config

Step 5: Review and Launch

Management Options

Enable Automatic Backups: ☒ Yes ☐ No

The number of days for which automated backups are retained.

Only MySQL, PostgreSQL, and Oracle are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail

Retention period: days

Automated backups are created if automated backups are enabled

Backup window: ☐ Select Window ☒ No Preference

The weekly time range (in UTC) during which system maintenance can occur.

Maintenance Window: ☐ Select Window ☒ No Preference

Backups are automated and scheduled

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume
+ License

Underlying DB:

MySQL, Postgres,
MSSQL, Oracle

API:

DB-specific

Amazon RDS

► Relational Database Service


Services ▼ Edit ▼

Step 1: Engine Selection
Step 2: Production?
Step 3: DB Instance Details
Step 4: Additional Configurations

Management Options

Enable Automatic Backups: ☒ Yes ☐ No

The number of days for which automated backups are retained.

are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail

Period: days

Automated backups are created if automated backups are enabled

Backup window: ☐ Select Window ☒ No Preference

The weekly time range (in UTC) during which system maintenance can occur.

Maintenance Window: ☐ Select Window ☒ No Preference

Backups are automated and scheduled

RDS

Model:

Managed RDBMS

Pricing:

Instance + Volume + License

Underlying DB:

MySQL, Postgres, MSSQL, Oracle

API:

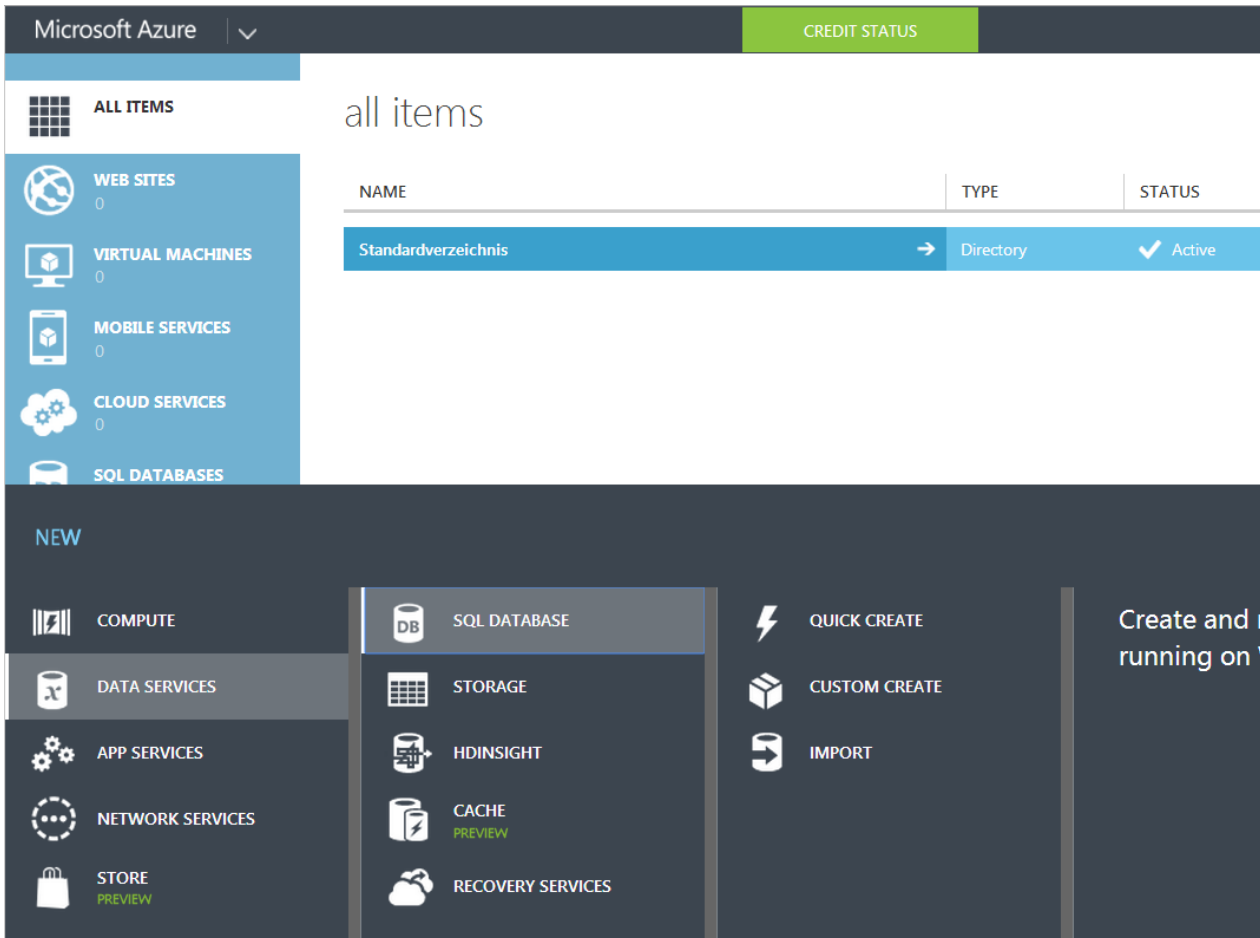
DB-specific



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

Microsoft SQL Azure

► Similar to RDS



The screenshot shows the Microsoft Azure portal interface. At the top, there's a header with 'Microsoft Azure' and a 'CREDIT STATUS' button. Below the header, a sidebar on the left lists various services: ALL ITEMS, WEB SITES, VIRTUAL MACHINES, MOBILE SERVICES, CLOUD SERVICES, and SQL DATABASES. The main area displays 'all items' with a table listing resources. The table has columns for NAME, TYPE, and STATUS. One resource is listed: 'Standardverzeichnis' of type 'Directory' with a status of 'Active'. At the bottom, there's a 'NEW' section with a grid of service tiles including COMPUTE, DATA SERVICES, APP SERVICES, NETWORK SERVICES, STORE, SQL DATABASE, STORAGE, HDINSIGHT, CACHE, and RECOVERY SERVICES. To the right of the grid are options for 'QUICK CREATE', 'CUSTOM CREATE', and 'IMPORT'.

NAME	TYPE	STATUS
Standardverzeichnis	Directory	✓ Active

SQL Azure

Model:

Managed RDBMS

Pricing:

Database size

Underlying DB:

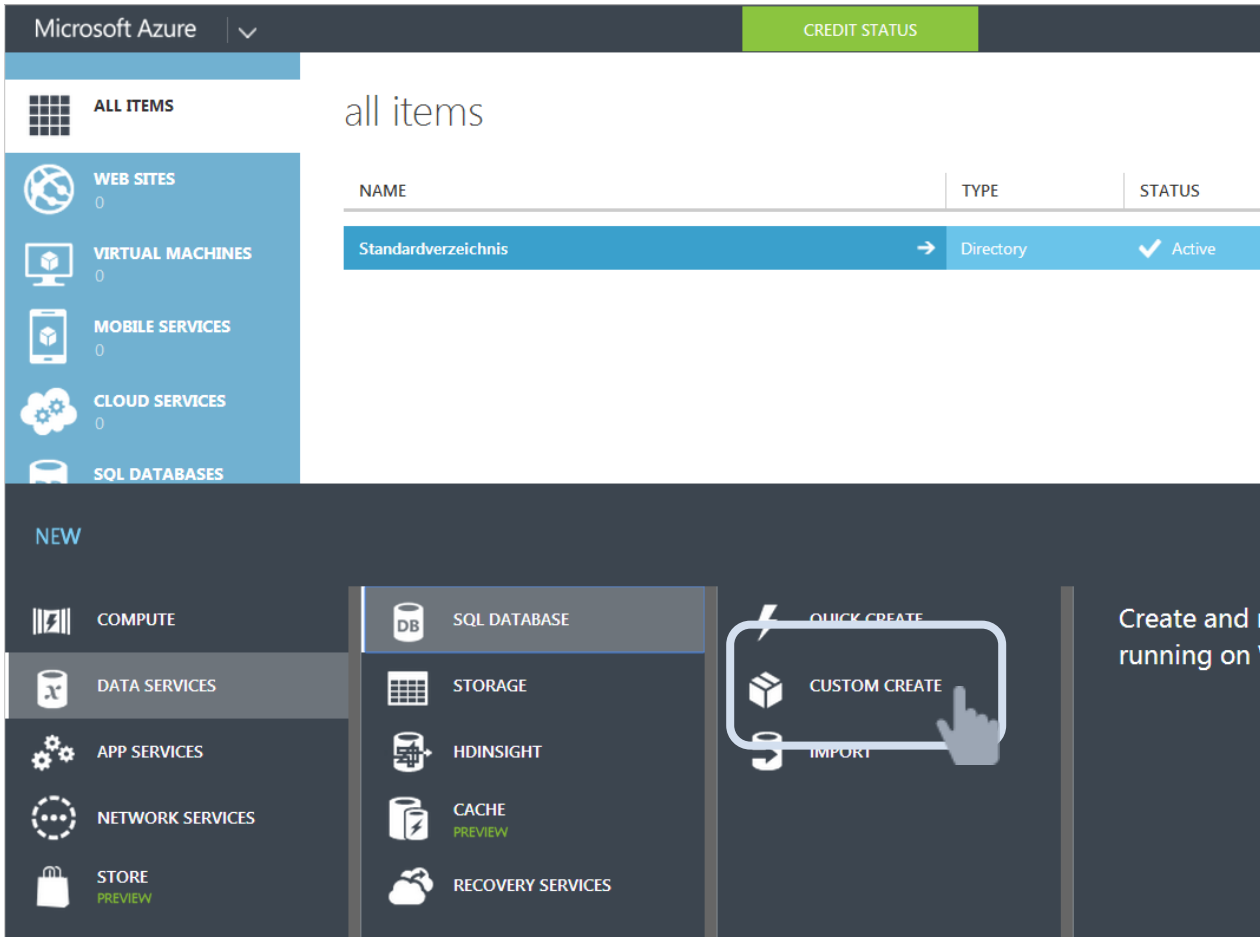
MSSQL Server

API:

T-SQL/TDS

Microsoft SQL Azure

► Similar to RDS



The screenshot shows the Microsoft Azure portal interface. At the top, there's a header with 'Microsoft Azure' and a 'CREDIT STATUS' button. Below the header, the left sidebar contains a list of services: ALL ITEMS, WEB SITES, VIRTUAL MACHINES, MOBILE SERVICES, CLOUD SERVICES, and SQL DATABASES. The main area displays 'all items' with a table listing resources. The table has columns for NAME, TYPE, and STATUS. One resource is listed: 'Standardverzeichnis' of type 'Directory' with status 'Active'. Below the table, the 'NEW' section is visible, showing various service categories like COMPUTE, DATA SERVICES, APP SERVICES, NETWORK SERVICES, and STORE. The 'SQL DATABASE' option is highlighted under the 'DATA SERVICES' category. To the right of the 'NEW' section, there's a 'QUICK CREATE' area with options for 'CUSTOM CREATE' and 'IMPORT', and a text prompt 'Create and m running on V'.

NAME	TYPE	STATUS
Standardverzeichnis	Directory	Active

SQL Azure

Model:

Managed RDBMS

Pricing:

Database size

Underlying DB:

MSSQL Server

API:

T-SQL/TDS



Microsoft SQL Azure

▶ Similar to RDS

NEW SQL DATABASE - CUSTOM CREATE

Specify database settings

NAME

test

SUBSCRIPTION

Kostenlose Testversion (e9d8c6c5-4238-441d-b8

EDITION

WEB

BUSINESS

MAX SIZE

1 GB

COLLATION

SQL_Latin1_General_CP1_CI_AS

SERVER

New SQL database server

x



SQL Azure

Model:

Managed RDBMS

Pricing:

Database size

Underlying DB:

MSSQL Server

API:

T-SQL/TDS

Microsoft SQL Azure

▶ Similar to RDS

CREATE SERVER

SQL database server settings

LOGIN NAME

test



LOGIN PASSWORD

.....

CONFIRM PASSWORD

.....

REGION

West US



☒ ALLOW WINDOWS AZURE SERVICES TO ACCESS THE SERVER. 

×

SQL Azure

Model:

Managed RDBMS

Pricing:

Database size

Underlying DB:

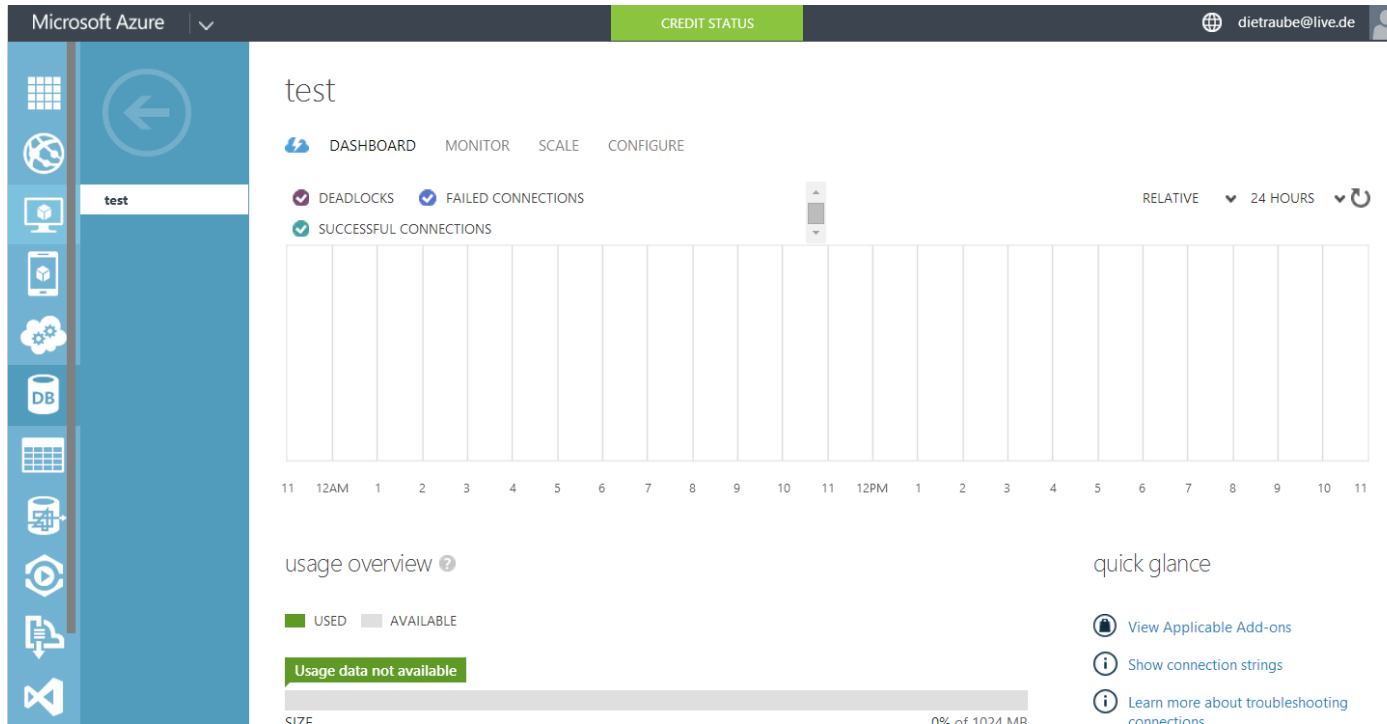
MSSQL Server

API:

T-SQL/TDS

Microsoft SQL Azure

- ▶ Similar to RDS



SQL Azure

Model:

Managed RDBMS

Pricing:

Database size

Underlying DB:

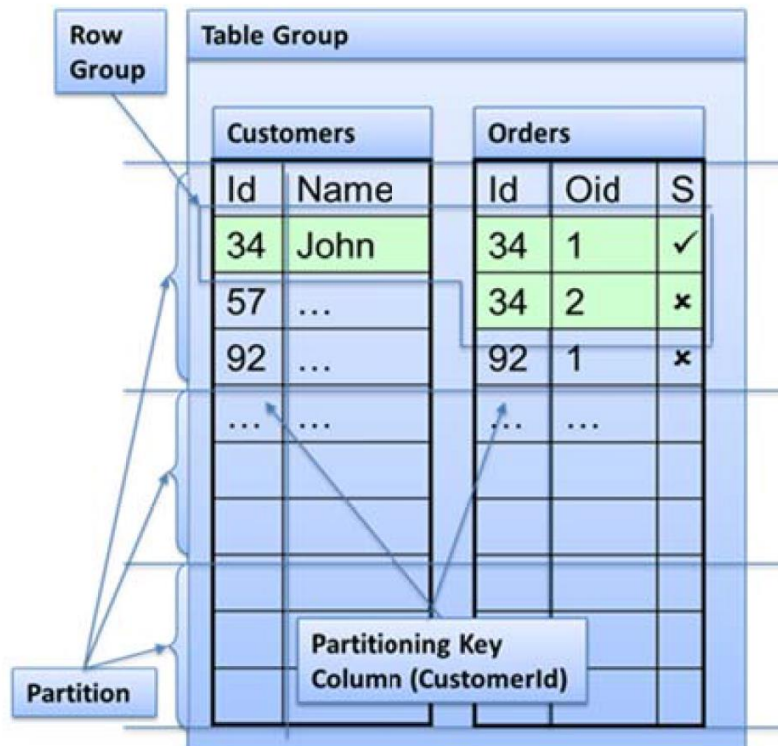
MSSQL Server

API:

T-SQL/TDS

Microsoft SQL Azure

- ▶ Similar to RDS



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

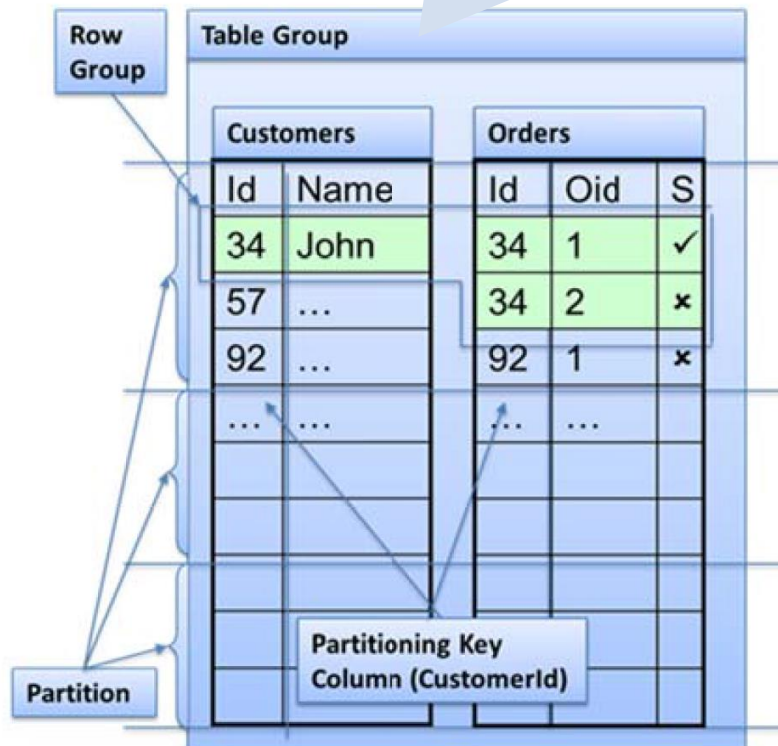


Microsoft SQL Azure

► Similar to RDS

Keyless Table Group: regular database

Keyed Table Group: partitioned by row key



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



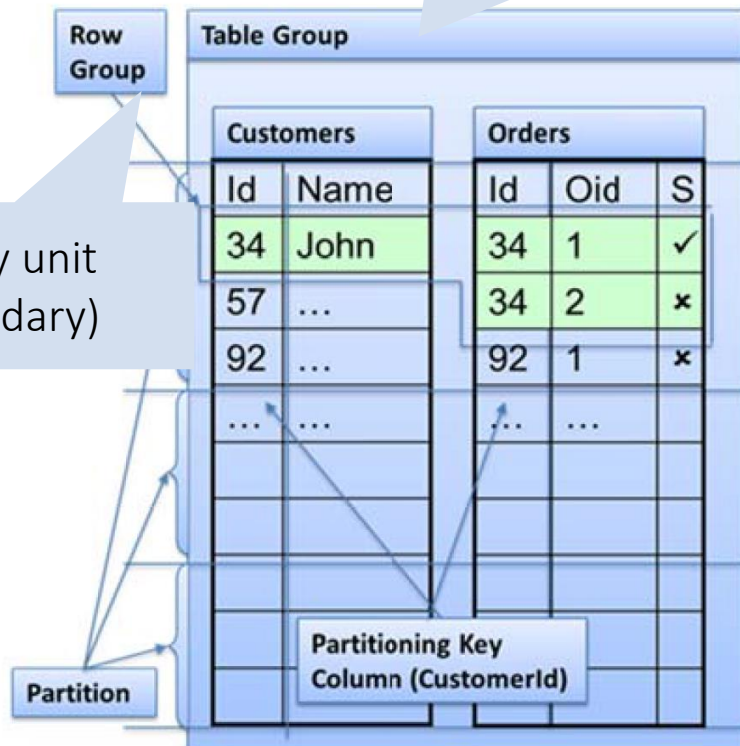
Microsoft SQL Azure

► Similar to RDS

Keyless Table Group: regular database

Keyed Table Group: partitioned by row key

Consistency unit
(ACID boundary)



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

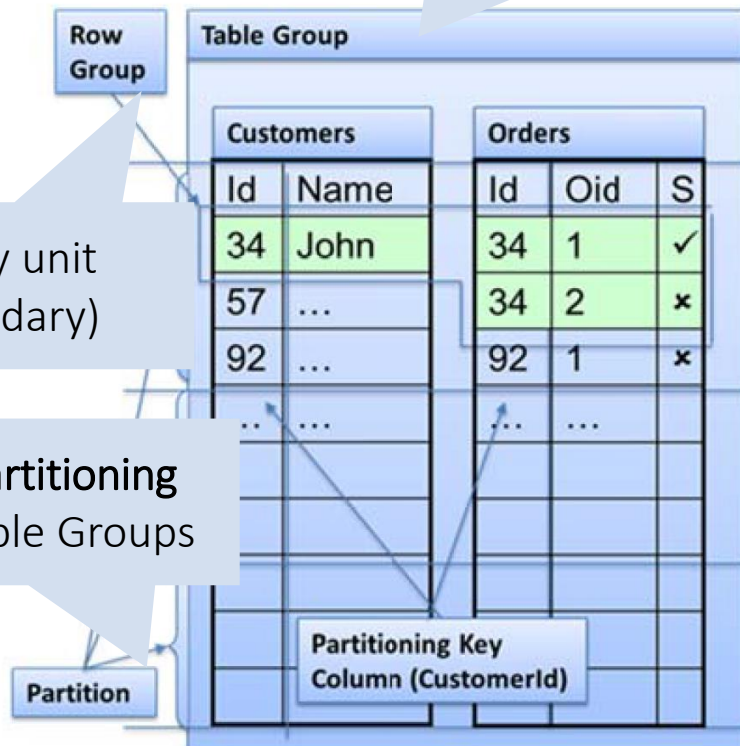


Microsoft SQL Azure

► Similar to RDS

Keyless Table Group: regular database

Keyed Table Group: partitioned by row key



Consistency unit
(ACID boundary)

Automatic Partitioning
for Keyed Table Groups

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

► Similar to RDS

Keyless Table Group: regular database

Keyed Table Group: partitioned by row key

Row Group		Table Group				
		Customers		Orders		
		Id	Name	Id	Oid	S
34	John	34	John	34	1	✓
	...	57	...	34	2	✗

Consistency unit
(ACID boundary)

Cloud SQL Server

- 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



- ▶ **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



- ▶ **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**



Google Cloud SQL

Pricing:

Database size

Underlying DB:

MySQL

Heroku Postgres

Pricing:

Plan based

Underlying DB:

Postgres

Other RDBMS services



- ▶ **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**



- ▶ **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

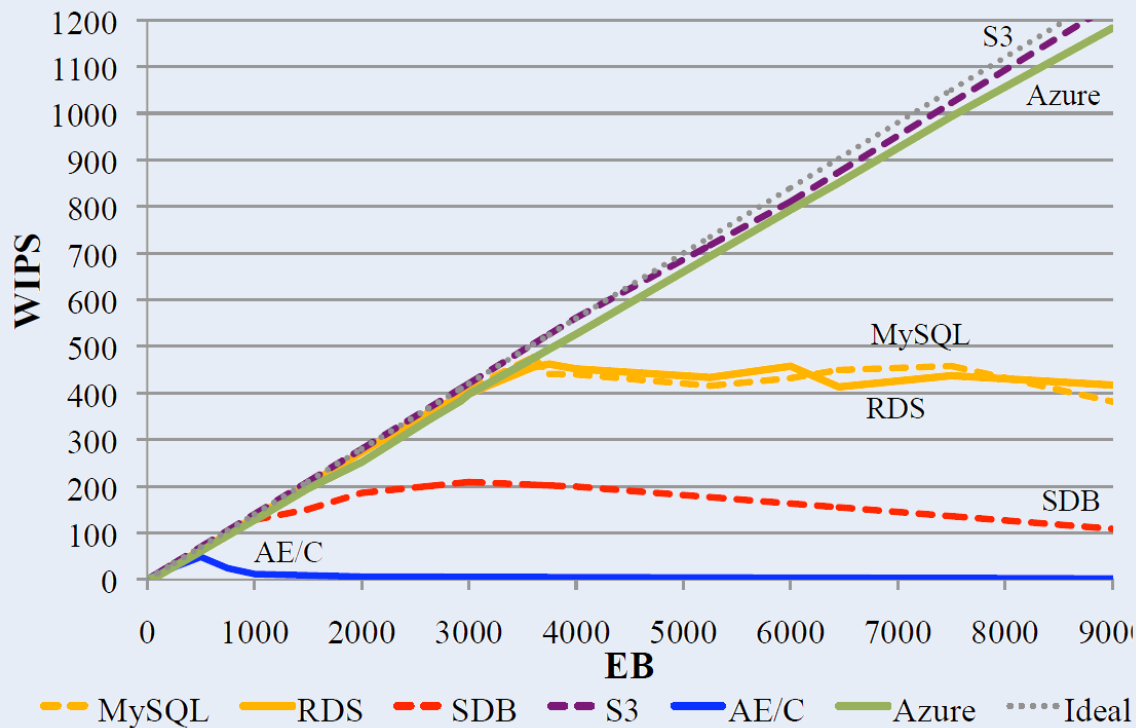
Underlying DB:

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

MySQL

Managed NoSQL services

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

Managed NoSQL services

	Model	CAP	Scans	Sec. Indices	Largest Cluster	Learning	Lic.	DBaaS
HBase	Wide-Column	CP	Over Row Key		~700	1/4	Apache	 (EMR)
MongoDB	Document	CP	yes		>100 <500	4/4	GPL	
Riak	Key-Value	AP			~60	3/4	Apache	 (Softlayer)

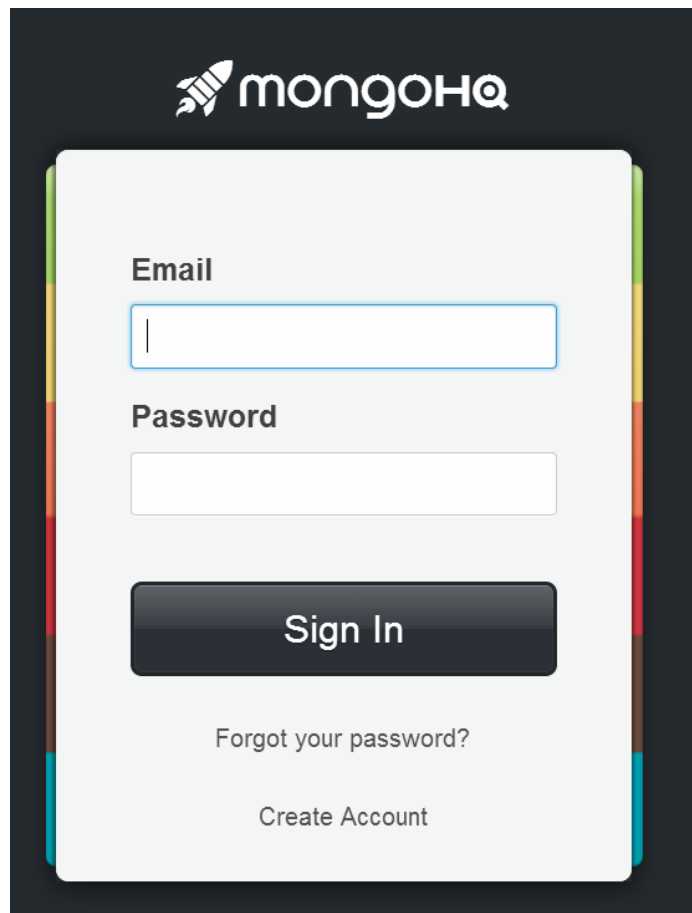
And there are many more:

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



MongoHQ

- ▶ EC2-based **MongoDB-as-a-Service**



The image shows a login form for MongoHQ. At the top is the MongoHQ logo. Below it are two input fields: 'Email' and 'Password'. The 'Email' field has a blue border and a cursor. The 'Password' field has a grey border. Below the fields is a dark grey 'Sign In' button. At the bottom are two links: 'Forgot your password?' and 'Create Account'.

MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

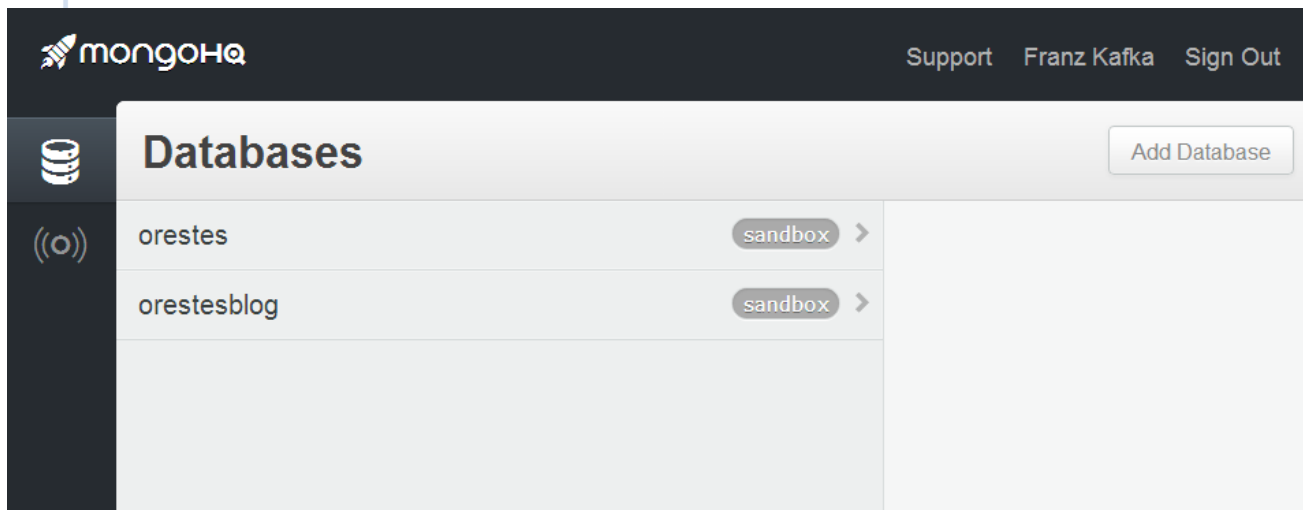
MongoDB

API:

Mongo, REST (*beta*)

MongoHQ

- ▶ EC2-based MongoDB-as-a-Service



MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

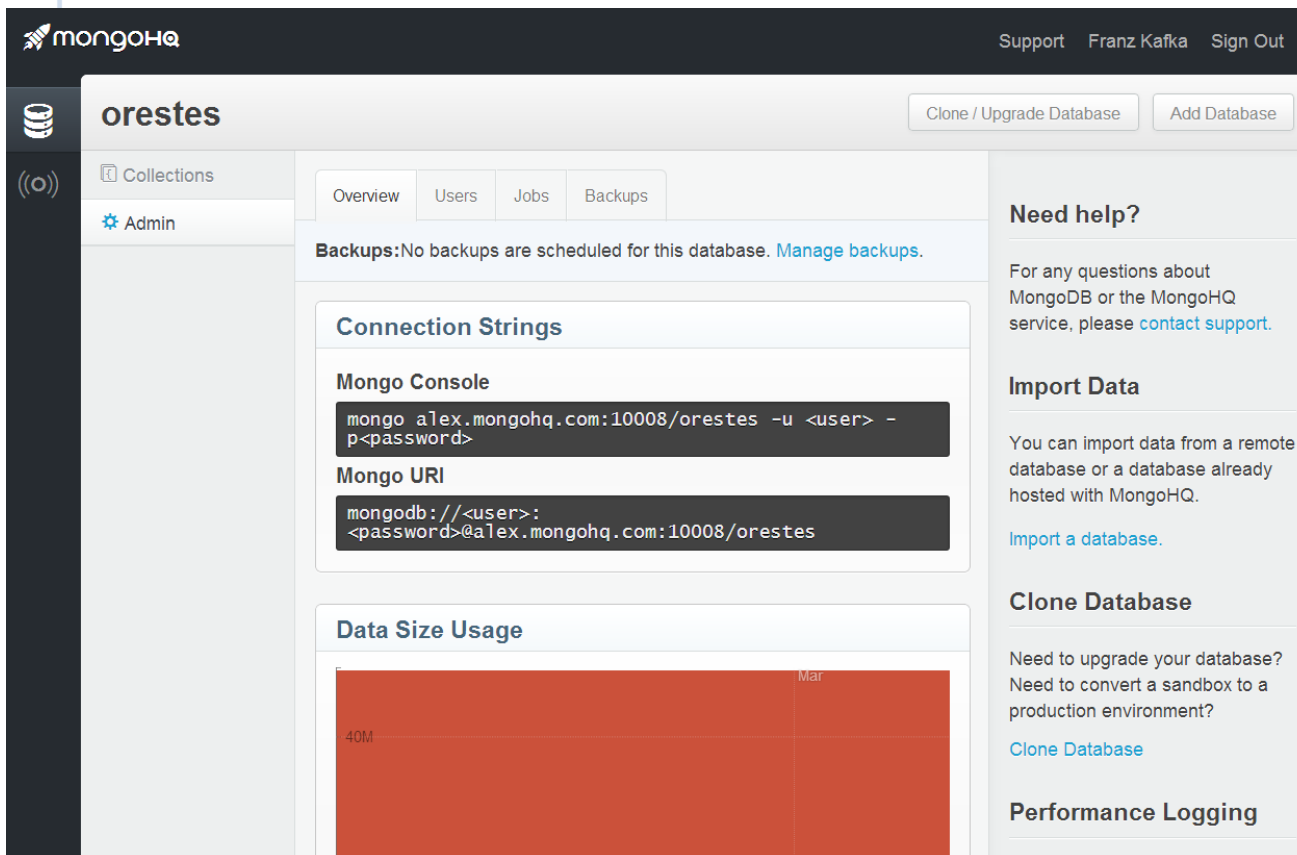
MongoDB

API:

Mongo, REST (*beta*)

MongoHQ

► EC2-based MongoDB-as-a-Service



The screenshot shows the MongoHQ web interface for a database named "orestes". The interface includes a sidebar with "Collections" and "Admin" tabs. The main content area has tabs for "Overview", "Users", "Jobs", and "Backups". The "Overview" tab is active, showing a message about backups and sections for "Connection Strings" and "Data Size Usage". The "Connection Strings" section includes a "Mongo Console" with a terminal snippet and a "Mongo URI" with a text box. The "Data Size Usage" section shows a red bar chart. The right sidebar contains links for "Need help?", "Import Data", "Clone Database", and "Performance Logging".

Connection Strings

Mongo Console

```
mongo alex.mongohq.com:10008/orestes -u <user> -p<password>
```

Mongo URI

```
mongodb://<user>:<password>@alex.mongohq.com:10008/orestes
```

Data Size Usage

40M

Mar

MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

MongoDB

API:

Mongo, REST (*beta*)

MongoHQ

▶ EC2-based MongoDB-as-a-Service

```
C:\Windows\system32\cmd.exe
C:\Users\Emil>mongo alex.mongohq.com:10008/orestes -u test -p test
MongoDB shell version: 2.6.0-rc1
connecting to: alex.mongohq.com:10008/orestes
> db.nosqlmatters.insert({"talk" : "cloud databases in research and practice"})
Cannot use commands write mode, degrading to compatability mode
WriteResult({ "nInserted" : 1 })
> db.nosqlmatters.find()
> db.nosqlmatters.find()
{ "_id" : ObjectId("535bd8d956aa113774f9e43a"), "talk" : "cloud databases in research and practice" }
> ^C
bye
```

MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

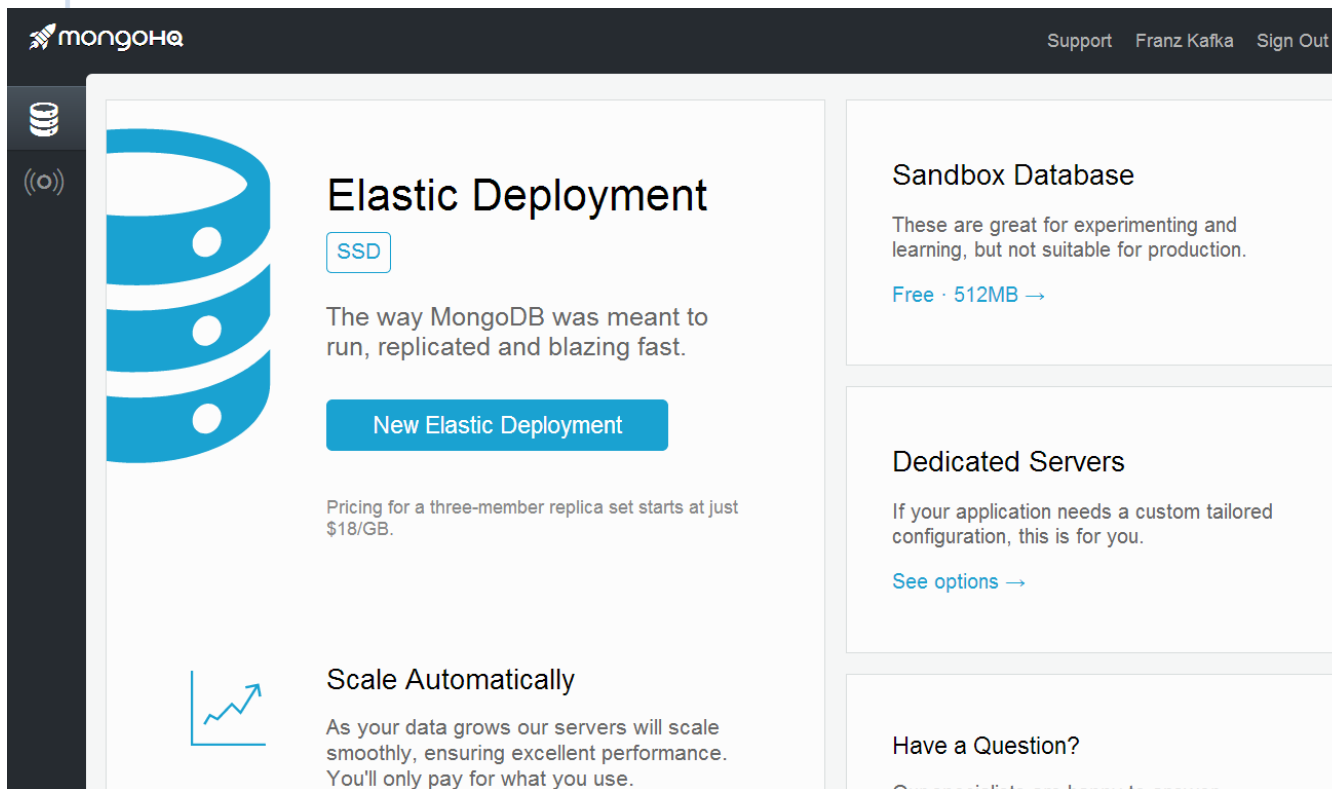
MongoDB

API:

Mongo, REST (*beta*)

MongoHQ

► EC2-based MongoDB-as-a-Service



The screenshot shows the MongoHQ website interface. At the top, there's a dark header with the MongoHQ logo on the left and links for 'Support', 'Franz Kafka', and 'Sign Out' on the right. A dark sidebar on the left contains a database icon and a '((o))' icon. The main content area is divided into several sections. The first section, 'Elastic Deployment', features a large blue database icon, a 'SSD' badge, and a description: 'The way MongoDB was meant to run, replicated and blazing fast.' Below this is a blue button labeled 'New Elastic Deployment' and pricing information: 'Pricing for a three-member replica set starts at just \$18/GB.' The second section, 'Scale Automatically', includes a line graph icon and text: 'As your data grows our servers will scale smoothly, ensuring excellent performance. You'll only pay for what you use.' To the right of these sections are two more boxes. The 'Sandbox Database' box describes it as 'great for experimenting and learning, but not suitable for production' and offers a 'Free · 512MB' option with a link. The 'Dedicated Servers' box states 'If your application needs a custom tailored configuration, this is for you' and provides a 'See options' link. At the bottom right, there's a 'Have a Question?' section with the text 'Our specialists are happy to answer'.

MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

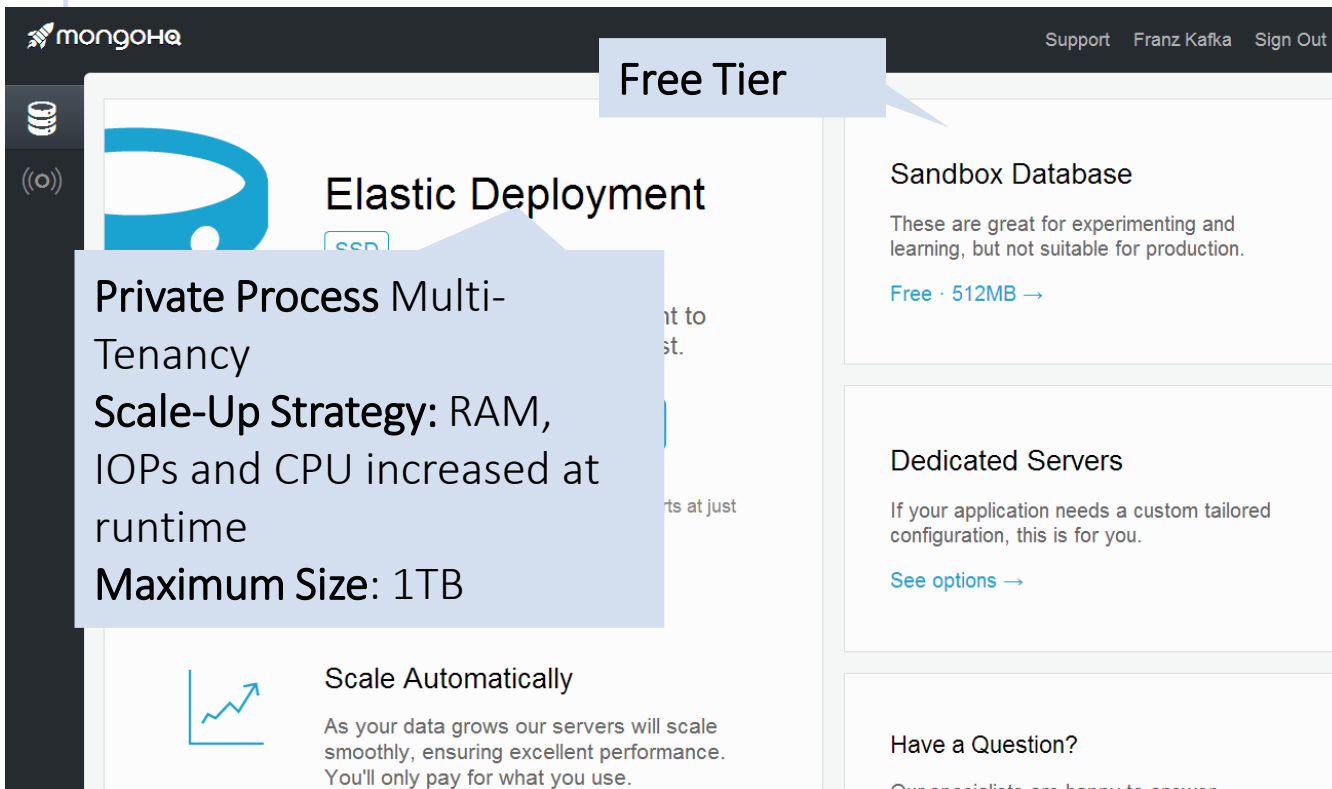
MongoDB

API:

Mongo, REST (*beta*)

MongoHQ

► EC2-based MongoDB-as-a-Service



Free Tier

Elastic Deployment

Private Process Multi-Tenancy

Scale-Up Strategy: RAM, IOPs and CPU increased at runtime

Maximum Size: 1TB

Scale Automatically

As your data grows our servers will scale smoothly, ensuring excellent performance. You'll only pay for what you use.

Sandbox Database

These are great for experimenting and learning, but not suitable for production.

[Free · 512MB →](#)

Dedicated Servers

If your application needs a custom tailored configuration, this is for you.

[See options →](#)

Have a Question?

Our specialists are happy to answer

MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

MongoDB

API:

Mongo, REST (*beta*)

MongoHQ

► EC2-based MongoDB-as-a-Service

MongoHQ

Model:

Managed NoSQL

Pricing:

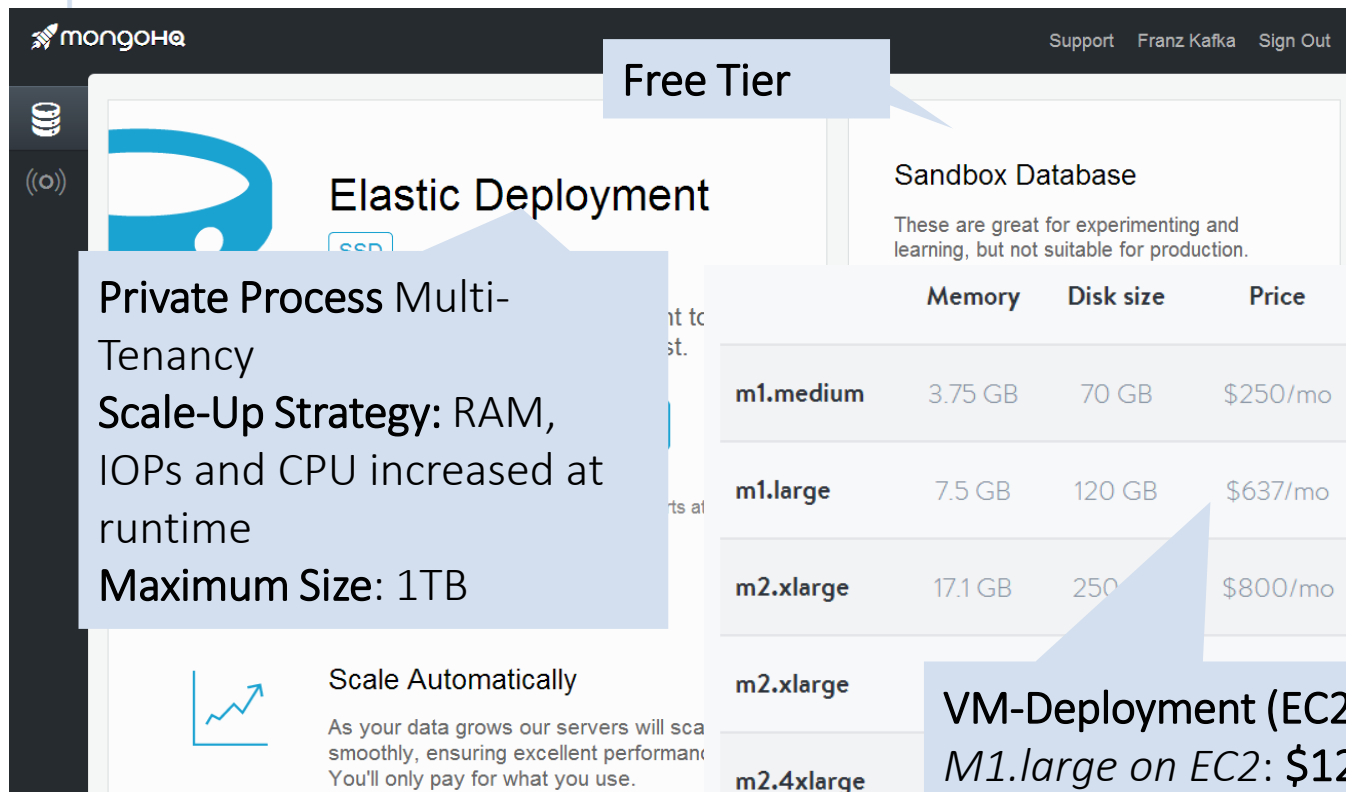
Plan-based

Underlying DB:

MongoDB

API:

Monao, REST (*beta*)



The screenshot shows the MongoHQ Elastic Deployment interface. At the top, there's a navigation bar with the MongoHQ logo, 'Support', 'Franz Kafka', and 'Sign Out'. Below the navigation bar, there's a 'Free Tier' callout pointing to a 'Sandbox Database' section. The main content area is titled 'Elastic Deployment' and features a 'Scale Automatically' section with a line graph icon. The graph shows a line that starts at the bottom left and trends upwards to the right, indicating growth. The text next to the graph says: 'Scale Automatically. As your data grows our servers will scale smoothly, ensuring excellent performance. You'll only pay for what you use.'

	Memory	Disk size	Price	Configurations
m1.medium	3.75 GB	70 GB	\$250/mo	Single server or Replica set
m1.large	7.5 GB	120 GB	\$637/mo	Single server or Replica set
m2.xlarge	17.1 GB	250 GB	\$800/mo	Single server or Replica set
m2.xlarge				Replica set
m2.4xlarge				Replica set

Private Process Multi-Tenancy
Scale-Up Strategy: RAM, IOPs and CPU increased at runtime
Maximum Size: 1TB

VM-Deployment (EC2):
M1.large on EC2: \$128.10
on EC2 with 1-yr-Res.: \$30
With MongoHQ: \$637

MongoHQ

- ▶ EC2-based MongoDB-as-a-Service
- ▶ #1 thing that should never happen:



MongoHQ

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

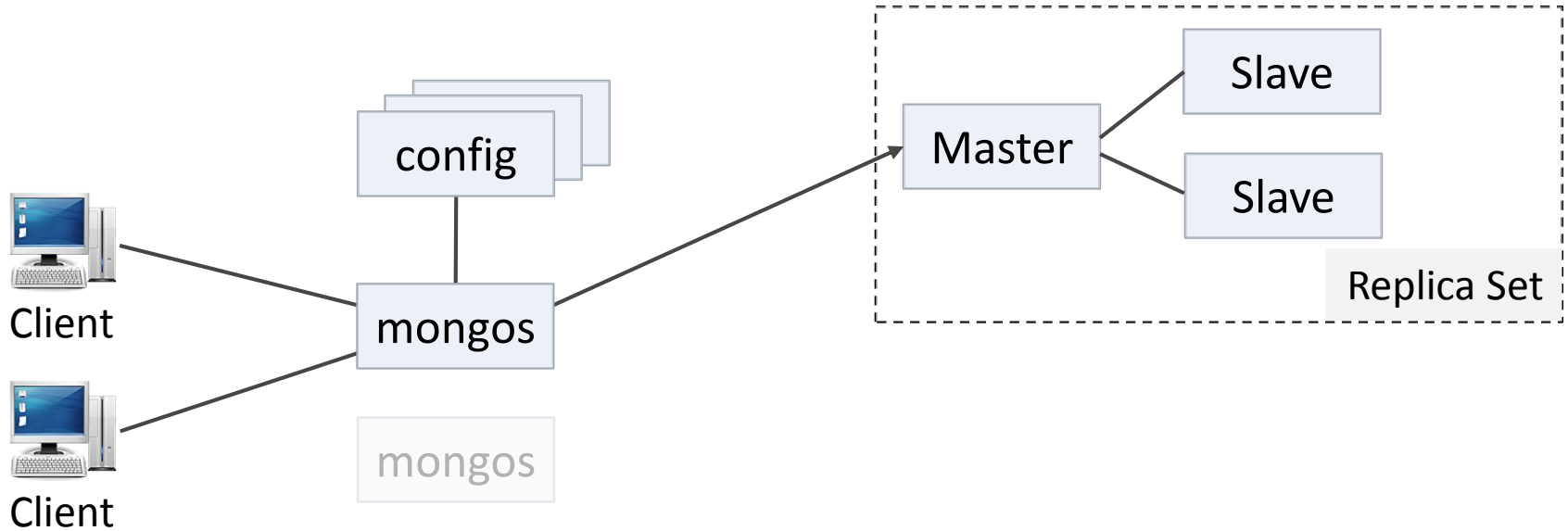
MongoDB

API:

Mongo, REST (*beta*)

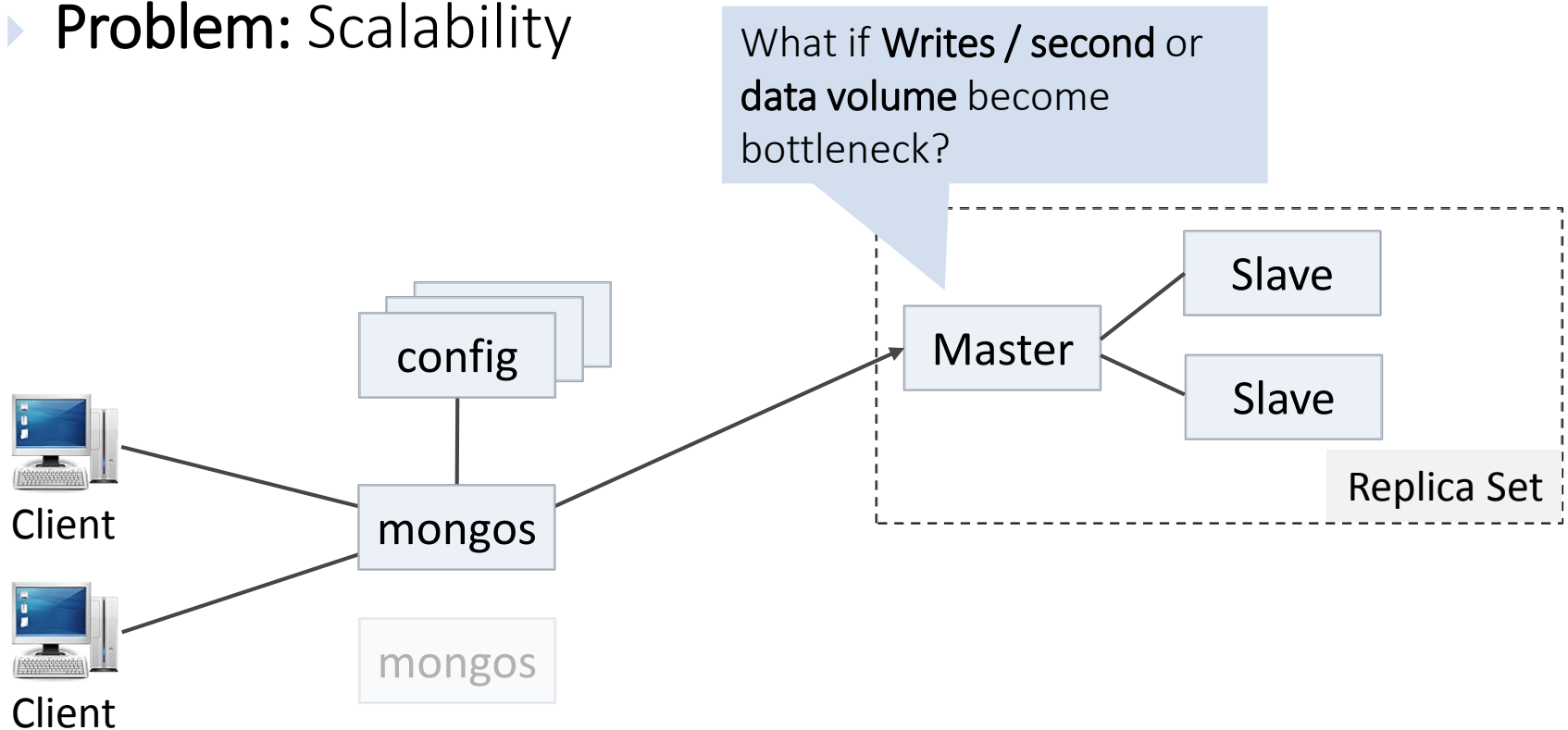
MongoHQ

► Problem: Scalability



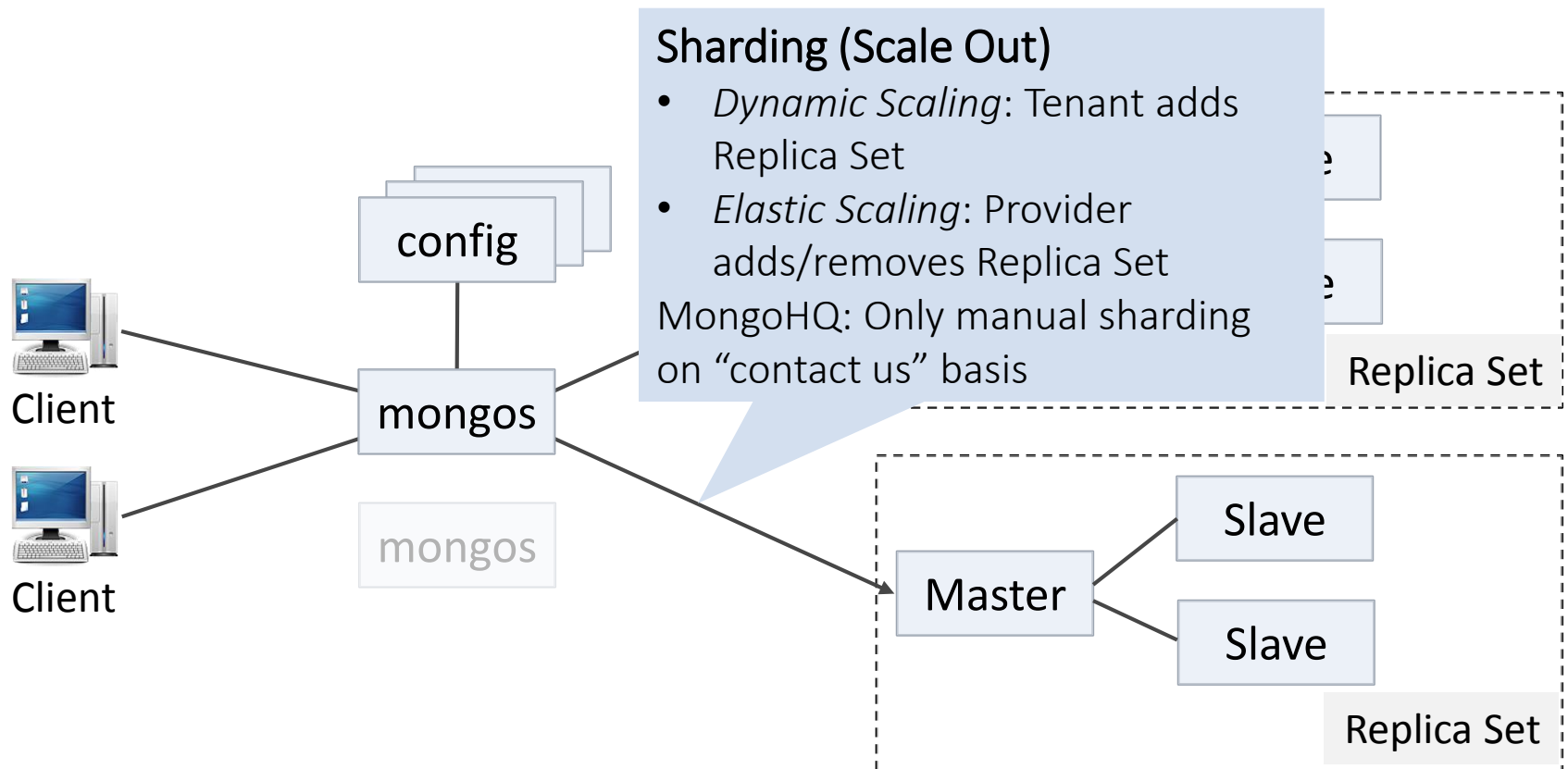
MongoHQ

► Problem: Scalability



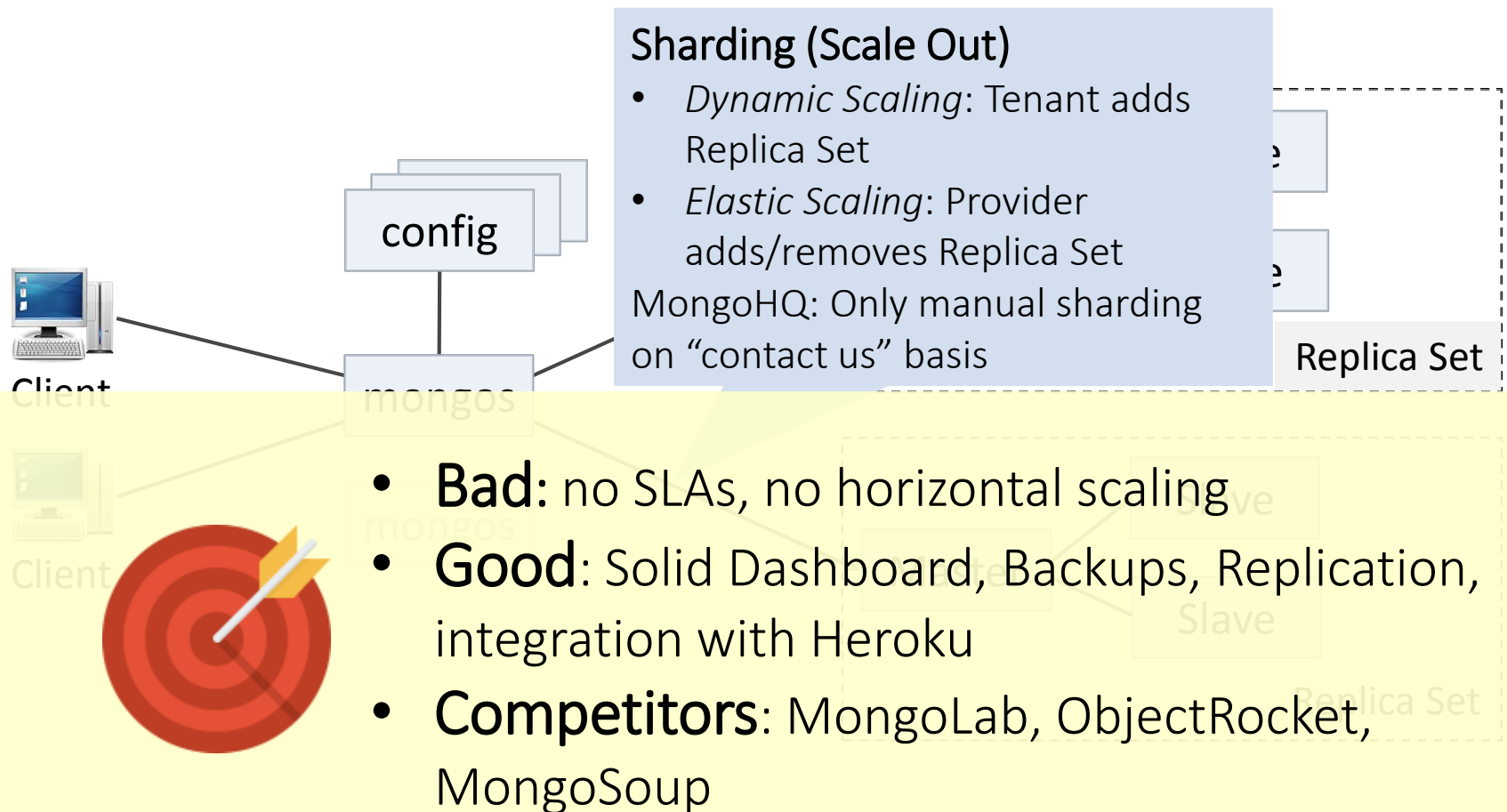
MongoHQ

► Problem: Scalability



MongoHQ

► Problem: Scalability



ElastiCache

► „RDS for Memcache and Redis“

Launch Cache Cluster Wizard

CACHE CLUSTER DETAILS

ADDITIONAL CONFIGURATION

REVIEW

To get started, provide the details for your Cache Cluster below.

Name*

Engine

Cache Port*

Engine Version

Number of Nodes*

Preferred Zone

Node Type

Cache Subnet Group

Topic for SNS Notification*
Manual ARN input

S3 Snapshot Location

Auto Minor Version Upgrade
☒ Yes ☐ No

Note: "Auto Minor Version Upgrade" only applies to the Cache Engine software. Critical System Software patches (e.g. security related) may be applied irrespective of this selection.

Note: A Cache Subnet Group is required for VPC

Cancel
Next

ElastiCache

Model:

Managed NoSQL

Pricing:

Infrastructure

Underlying DB:

Memcache, Redis

API:

DB, REST
(management)

ElastiCache

- „RDS for Memcache and Redis“

Launch Cache Cluster Wizard

To get started, provide the details for your new cache cluster.

Memcache can be run as a cluster (client-side sharding)

Memcache or Redis

Cache Port* ⓘ

Number of Nodes* ⓘ

Node Type ⓘ

Topic for SNS Notification* ⓘ [Manual ARN input](#) ⓘ

Engine ⓘ

Engine Version ⓘ

Preferred Zone ⓘ

Cache Subnet Group ⓘ

Note: A Cache Subnet Group is required for VPC

“Upgrade” only applies to the Cache Engine software patches (e.g. security related) may be selected.

* Required

Cancel Next

ElastiCache

Model:

Managed NoSQL

Pricing:

Infrastructure

Underlying DB:

Memcache, Redis

API:

DB, REST
(*management*)

ElastiCache

► „RDS for Memcache and Redis“

Announced last
Saturday: **Snapshots**



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.

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*)

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
activeresharding	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

ElastiCache

- „RDS for Memcache and Redis“

ElastiCache

Model:

Managed NoSQL

Pricing:

Infrastructure

Underlying DB:

Memcache, Redis

API:

DB, REST
(*management*)

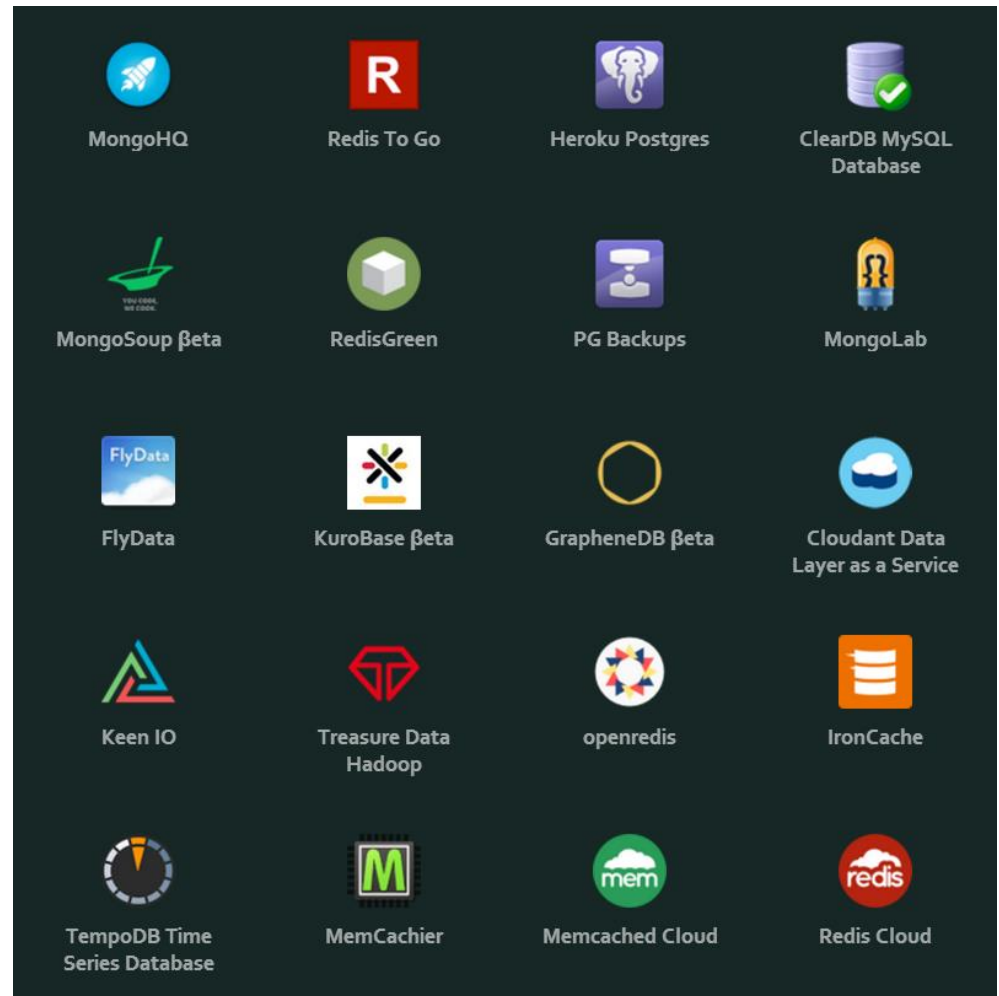
1 Cache Parameter Group(s) selected				
Cache Parameter Group: default.redis2.8				
Viewing:	All Parameters	Edit Parameters		
▲ Name	Allowed Values	Is Modifiable	Instance Class	Value
activeresharding	yes,no	Y	ALL	yes
appendonly	yes,no	Y	ALL	no
client-output-buffer-limit-normal-hard-limit	0-			0
client-output-buffer-limit-normal-soft-limit	0-			0

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

AWS CLI tools and SDKs
offer a superset of
Dashboard functions

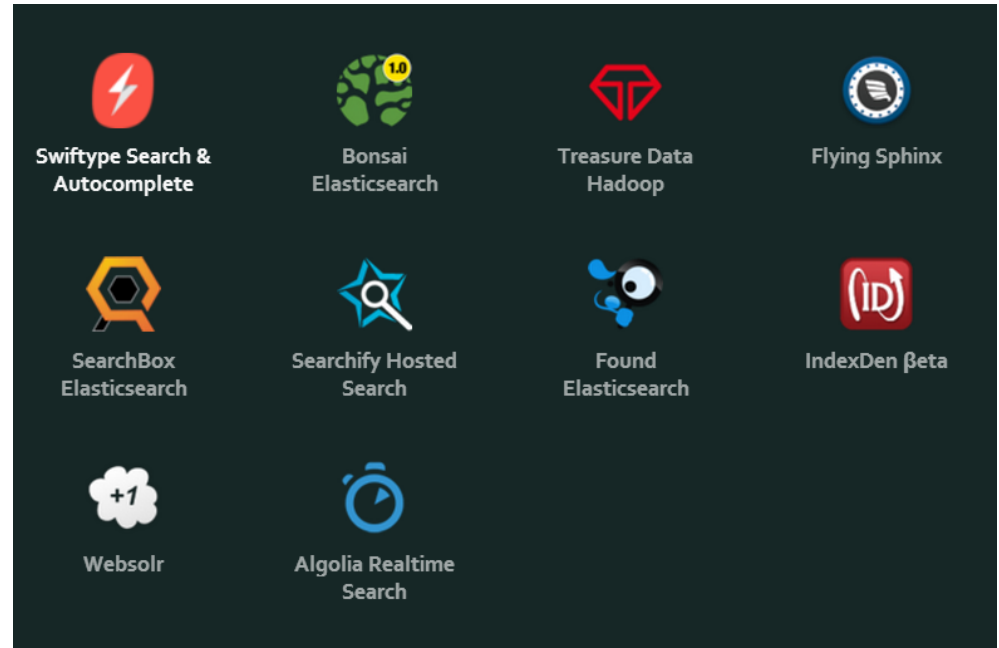
Heroku DBaaS Addons

- ▶ 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

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
```

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

Redis2Go

Model:

Managed NoSQL

Pricing:

Plan-based

Underlying DB:

Redis

API:

Redis



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

Depl

```
$ git push master
```

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

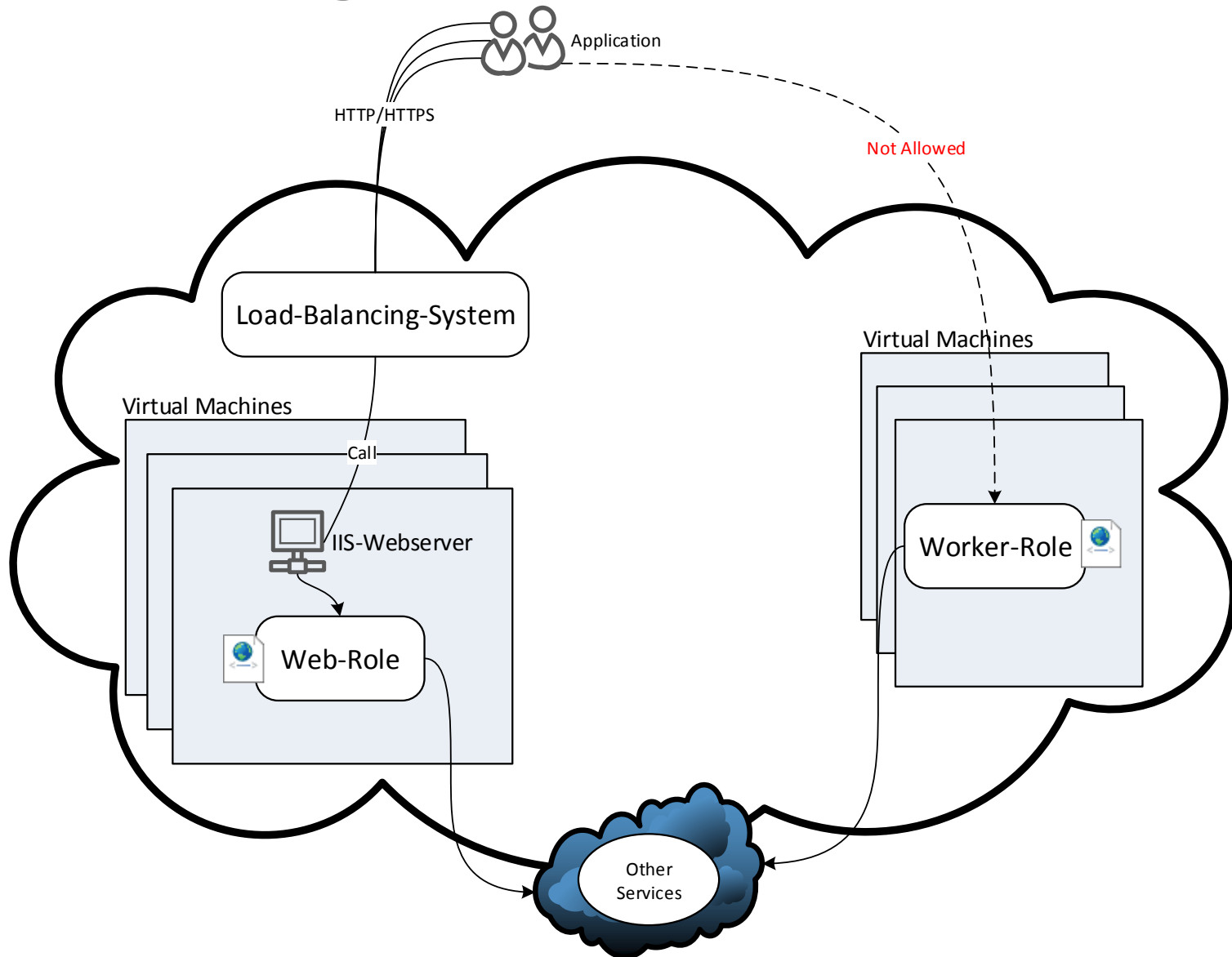
	Model	CAP	Scans	Sec. Indices	Queries	API	Scale-out	SLA
SimpleDB	Table-Store	CP	Yes (as queries)	Auto-matic	SQL-like (no joins, groups, ...)	REST + SDKs		
Dynamo-DB	Table-Store	CP	By range key / index	Local Sec. Global Sec.	Key+Cond. On Range Key(s)	REST + SDKs	Automatic over Prim. Key	
Azure Tables	Table-Store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

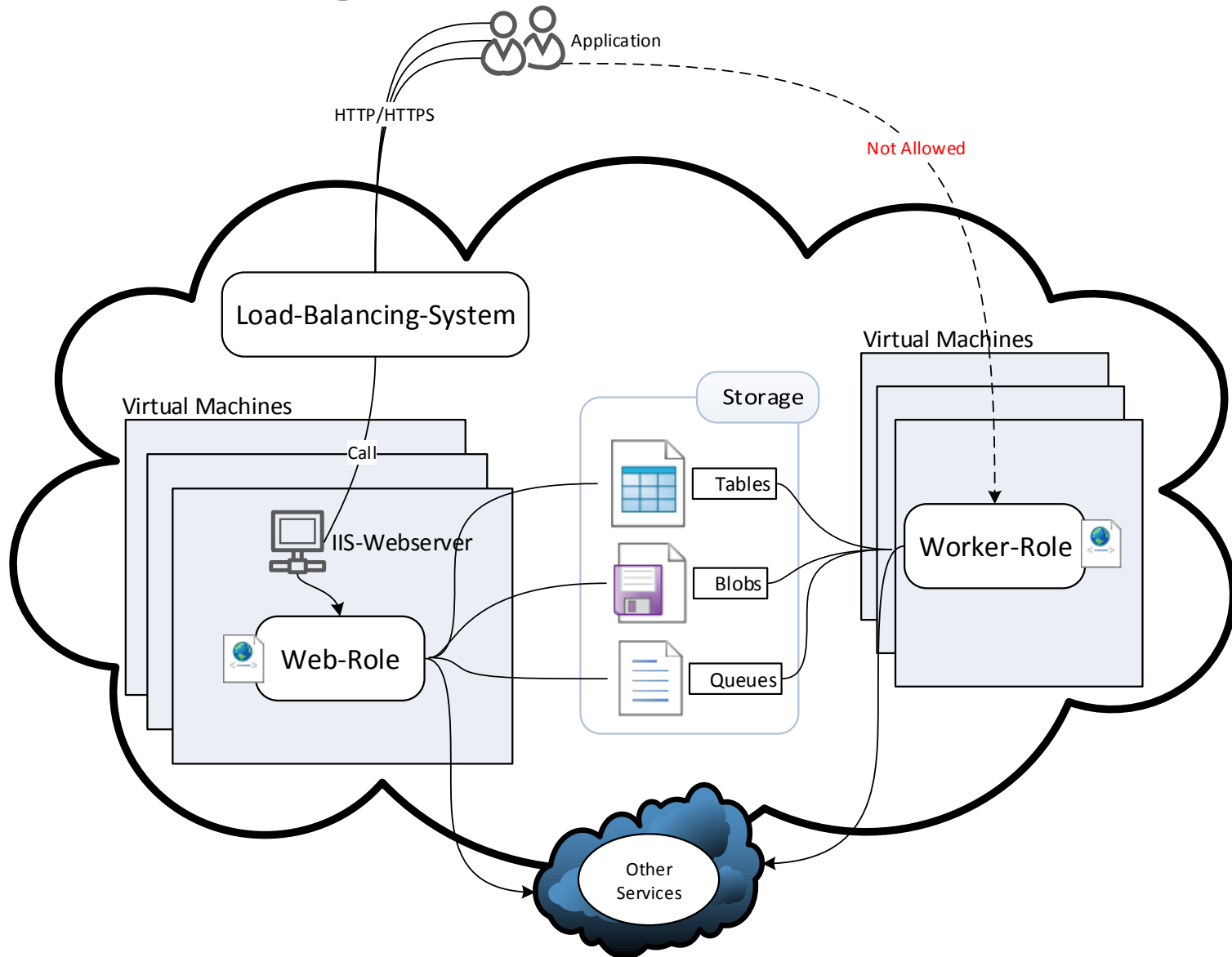


Table Service example: Azure Tables

REST API	Partition Key	Row Key (<i>sortiert</i>)	Timestamp (<i>autom.</i>)	Property1	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

REST API	Partition Key	Row Key (<i>sortiert</i>)	Timestamp (<i>autom.</i>)	Property1	P
	intro.pdf	v1.1	14/6/2013	...	...
	intro.pdf	v1.2	15/6/2013		...
	p		11/6/2013	...	

No Index: Lookup only (!) by full table scan

Atomic "Entity-Group Batch Transaction" possible

Hash-distributed to partition servers

Sparse

Partition

Partition



Table Service example: Azure Tables

REST API	Partition Key	Row Key (<i>sortiert</i>)	Timestamp (<i>autom.</i>)	Property1	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 Table Storage

How can we improve Azure Storage?

Ch ← Storage

▶ S **789** votes

▶ V **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

Challenges:

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

Good:

◀ Automatic **Distribution**

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

Azure Tables

Model:

Proprietary

Pricing:

Requests + Storage
+ Network

Underlying DB:

Custom System

API:

REST





Azure Table Storage

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 Tables

Model:

Proprietary

Pricing:

Requests + Storage
+ Network

Underlying DB:

Custom System

API:

REST

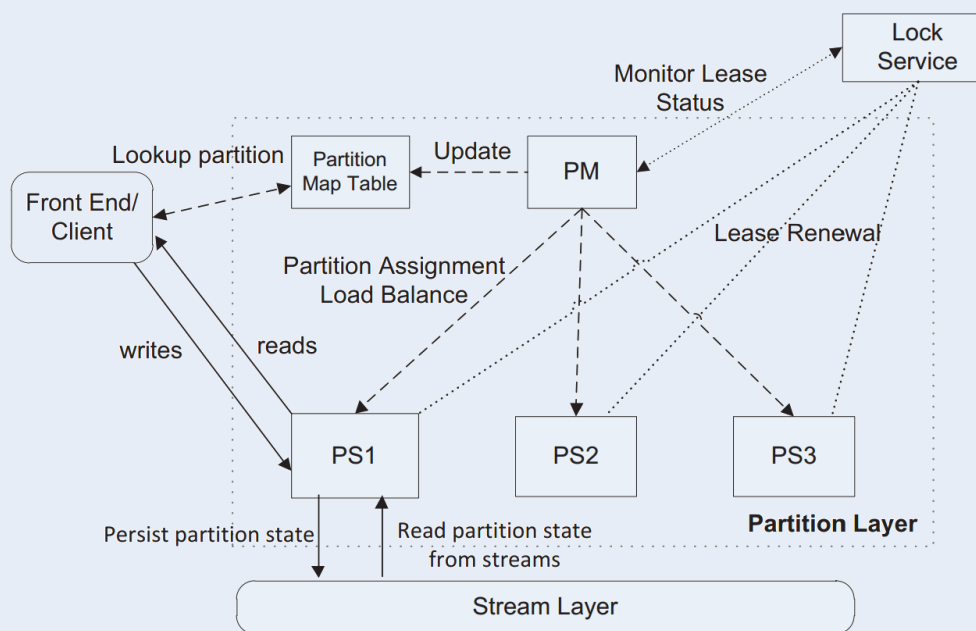


Azure Table Storage

Windows Azure Storage

Idea:

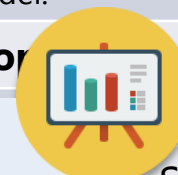
- Layered storage infrastructure for Blobs and Tables
- Use **research results** (GFS, BigTable, Paxos, LSM, Erasure Coding)



Azure Tables

Model:

Prop



+ Storage

rk

DB:

System



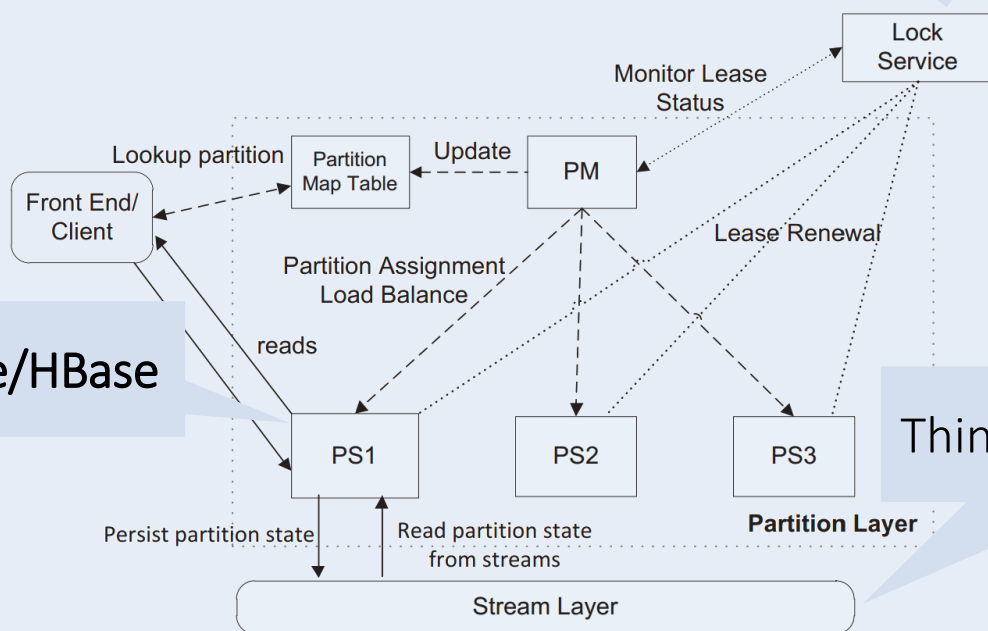
Azure Table Storage

Windows Azure Storage

Idea:

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

Think: Chubby/ZooKeeper



Think: BigTable/HBase

Think: GFS/HDFS



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:

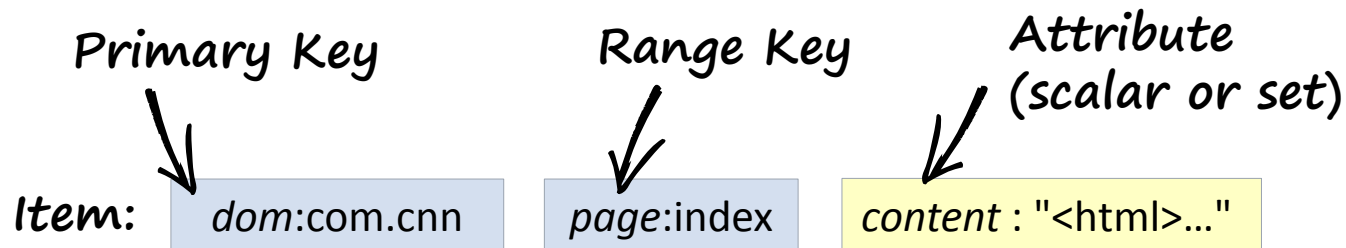
Custom System

API:

REST

DynamoDB

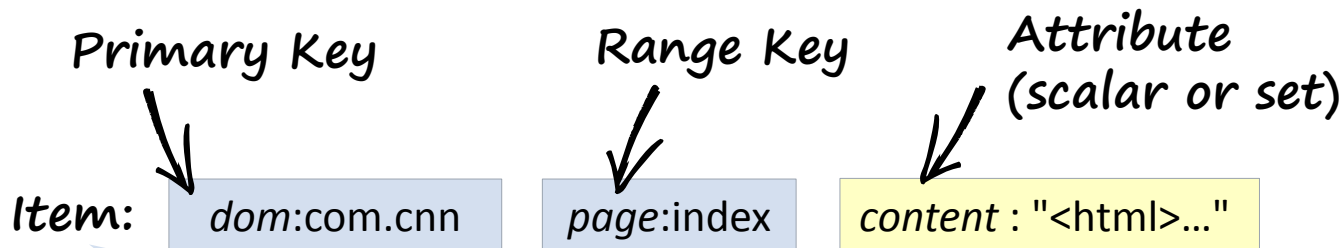
- ▶ Successor to SimpleDB



DynamoDB
Model:
Proprietary
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)

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

- ▶ **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

Create TableCanc



Table Name:
Table will be created in eu-west-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: ☒ Hash and Range ☐ Hash

Hash Attribute Name: ☒ String ☐ Number ☐ Binary

Range Attribute Name: ☒ String ☐ Number ☐ Binary

 Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.
For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.
[Learn more about choosing your primary key](#)

DynamoDB

Model:

Proprietary

Pricing:

Provisioned
Throughput +
Network

Underlying DB:

Custom System

API:

REST

DynamoDB

► Successor to SimpleDB

Create TableCanc

**PRIMARY KEY**

ADD INDEXES
(optional)

PROVISIONED
THROUGHPUT CAPACITY

THROUGHPUT ALARMS
(optional)

SUMMARY

Table Name:
Table will be created in eu-west-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: ☒ Hash and Range ☐ Hash

Hash Attribute Name: ☒ String ☐ Number ☐ Binary

Range Attribute Name: ☒ String ☐ Number ☐ Binary

 Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.
For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.
[Learn more about choosing your primary key](#)

DynamoDB

Model:

Proprietary

Pricing:

Provisioned
Throughput +
Network

Underlying DB:

Custom System

API:

REST

DynamoDB

► Successor to SimpleDB

Create TableCanc

**PRIMARY KEY**

ADD INDEXES
(optional)

PROVISIONED
THROUGHPUT CAPACITY

THROUGHPUT ALARMS
(optional)

SUMMARY

Table Name:
Table will be created in eu-west-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: ☒ Hash and Range ☐ Hash

Hash Attribute Name: ☒ String ☐ Number ☐ Binary

Range Attribute Name: ☒ String ☐ Number ☐ Binary

 Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.
For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.
[Learn more about choosing your primary key](#)

DynamoDB

Model:

Proprietary

Pricing:

Provisioned
Throughput +
Network

Underlying DB:

Custom System

API:

REST

DynamoDB

► Successor to SimpleDB

Create TableCanc

 PRIMARY KEY

ADD INDEXES
(optional)

PROVISIONED
THROUGHPUT CAPACITY

THROUGHPUT ALARMS
(optional)

SUMMARY

Table Name:
Table will be created in eu-west-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: ☒ Hash and Range ☐ Hash

Hash Attribute Name: ☐ String ☐ Number ☐ Binary

Range Attribute Name: ☐ String ☐ Number ☐ Binary

 Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.
For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.
[Learn more about choosing your primary key](#)

DynamoDB

Model:

Proprietary

Pricing:

Provisioned
Throughput +
Network

Underlying DB:

Custom System

API:

REST

DynamoDB

- ▶ Successor to SimpleDB



Provisioned Throughput Capacity:

- ☐ Help me calculate how much throughput capacity I need to provision

Throughput capacity to

Amazon DynamoDB lets you provision throughput capacity for your table. Using this information, Amazon DynamoDB automatically provisions the appropriate resources to meet your throughput needs. For more information, see [Provisioned Throughput Capacity](#).

Unit of Billing

Write throughput capacity you wish to provision. Amazon DynamoDB automatically provisions the appropriate resources to meet your throughput needs.

Read Capacity Units:

Write Capacity Units:

 Throughput capacity for this table will cost up to \$65.63 per month if you have exceeded the [free tier](#).
*Taxes may apply.

 If you exceed the free tier you are charged for the provisioned throughput capacity of your table even if you do not actively use your provisioned capacity. [Learn more about DynamoDB's free tier and pricing.](#)

DynamoDB

Model:

Proprietary

Pricing:

Provisioned
Throughput +
Network

Underlying DB:

Custom System

API:

REST

DynamoDB

- ▶ Successor to SimpleDB



Provisioned Throughput Capacity:

- Help me calculate how much throughput capacity I need to provision

Throughput capacity to

Amazon DynamoDB lets you provision throughput capacity to meet your throughput needs. Using this information, Amazon DynamoDB can help you provision the right amount of capacity.

Read Capacity Units: 100

Write Capacity Units: 100



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

DynamoDB

Model:

Proprietary

Pricing:

Provisioned
Throughput +
Network

Underlying DB:

Custom System

API:

REST

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 → BigTable



Google Developers Console +Felix

< HelloNoSQL

CREATE ENTITY

Kind: talks Filters

talks Entities

NAME/ID	NAME	PRESENTER
☐ id=5639445604728832	cloud databases in research and practice	felix
☐ id=5649391675244544	cloud dbs	—

Overview
APIs & auth
Permissions
Settings
Support
App Engine
Compute Engine
Cloud Storage
Cloud Datastore
Dashboard
Query
Indexes

AE/Cloud DataStore

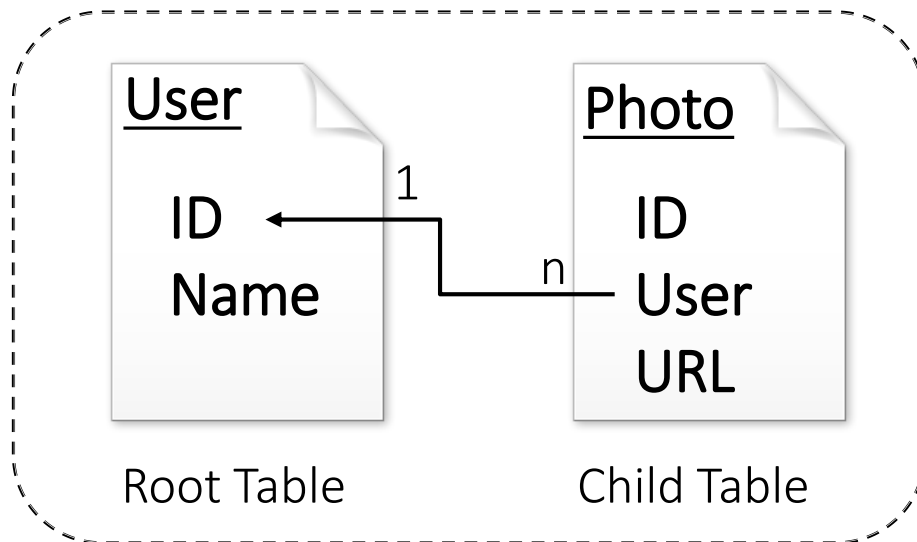
- ▶ Structured Storage System for **App Engine**
- ▶ Based on:
 - Megastore → BigTable → Colossus



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:



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:

DataStore

Model:

Proprietary

Pricing:

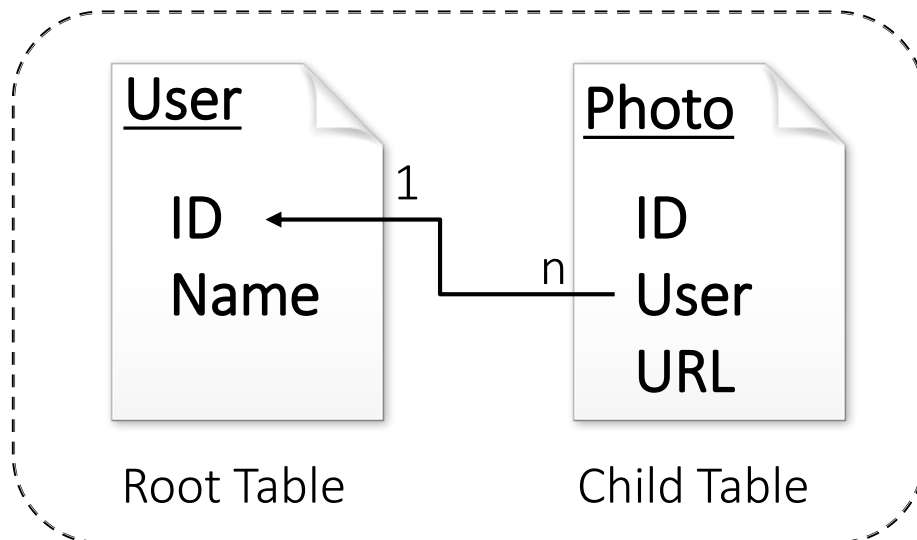
CPU + Storage +
Network

Underlying DB:

MegaStore

API:

SDK, JPA, JDO



EG: User + n Photos

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

AE/Cloud DataStore

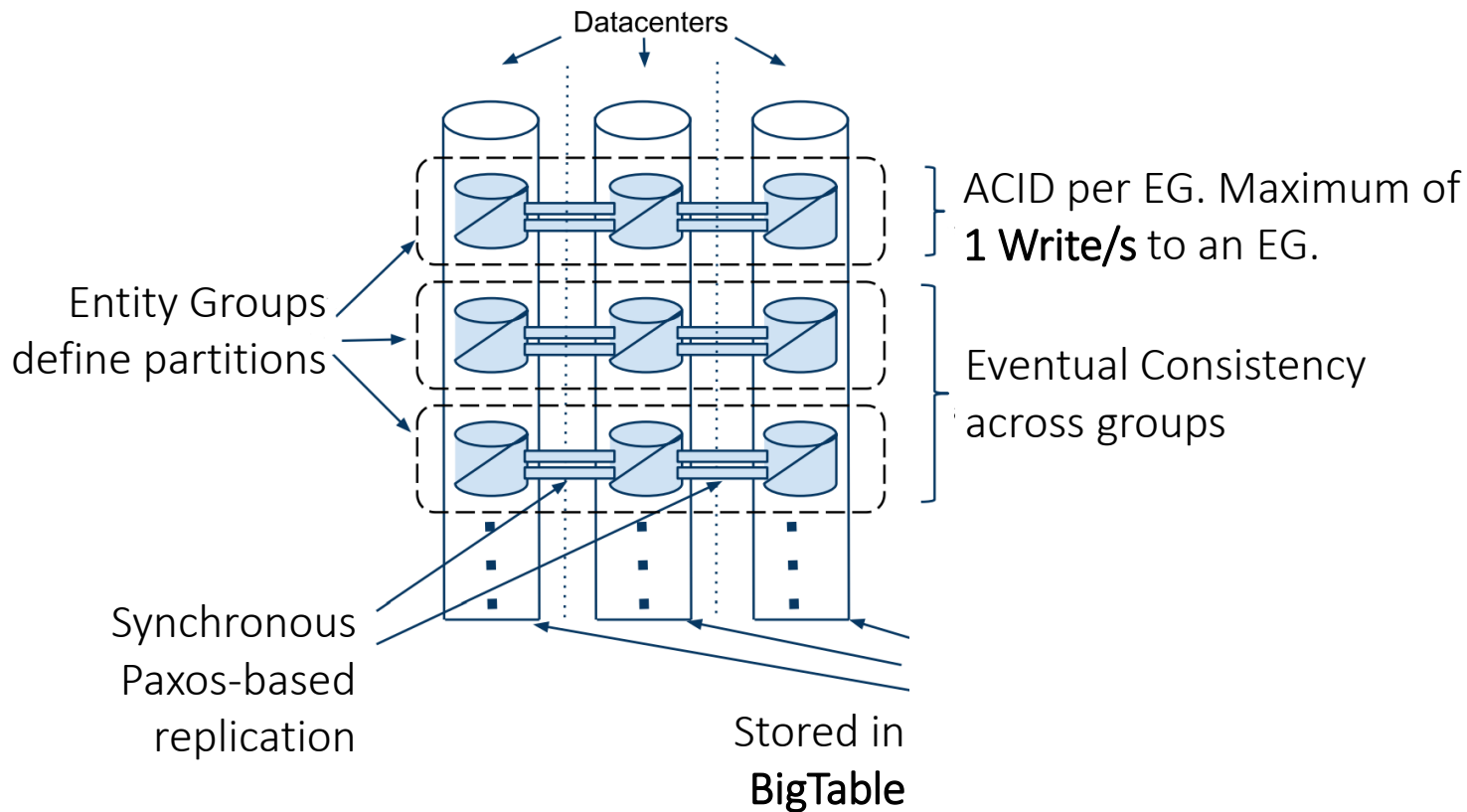
- ▶ Structured Storage System for **App Engine**
- ▶ Based on:
 - Megastore → BigTable → Colossus
- ▶ Schemafree **Entity Group (EG)** data model:

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

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

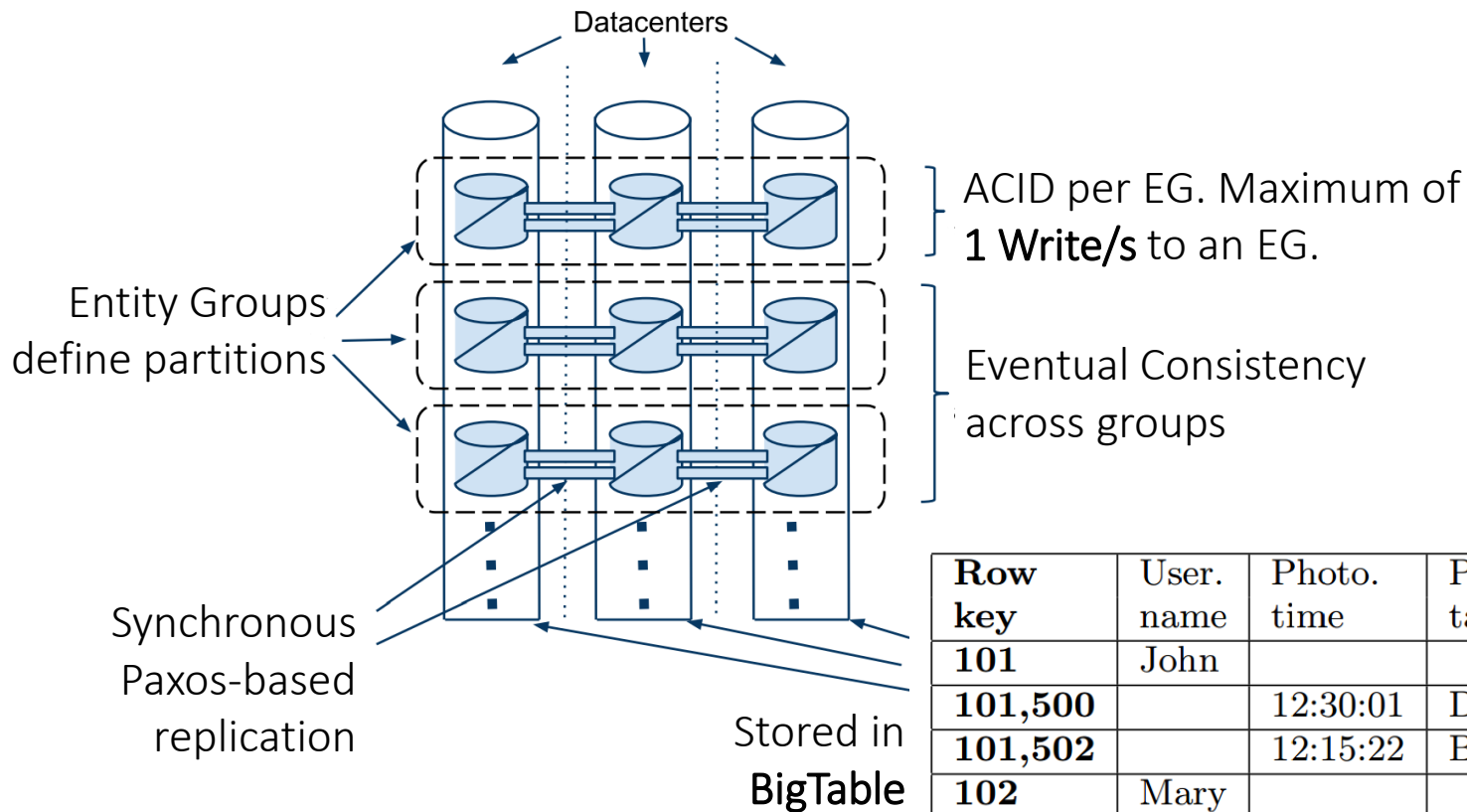
AE/Cloud DataStore

Internally:



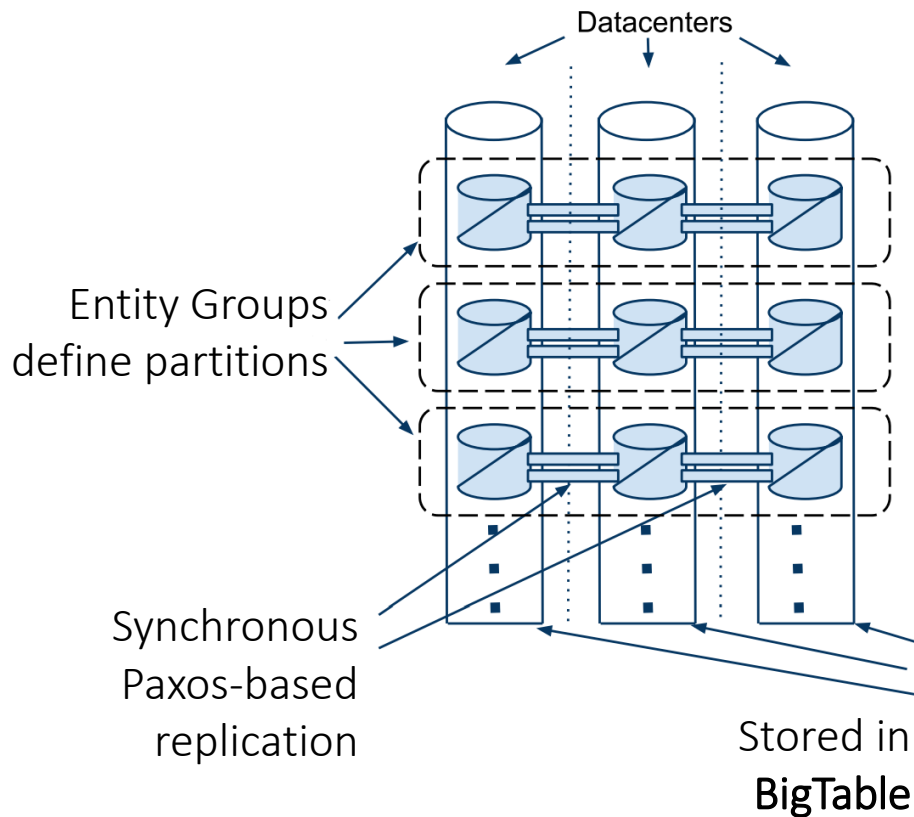
AE/Cloud DataStore

Internally:



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.

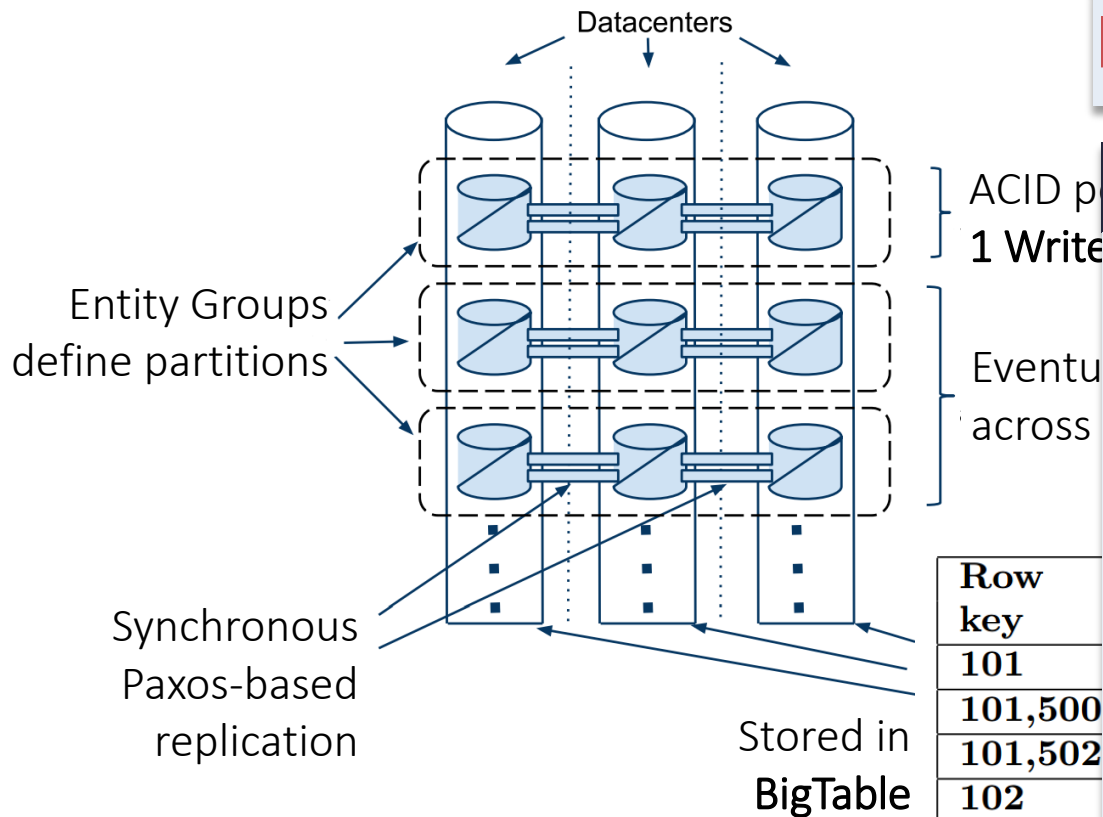
ACID per EG. Maximum of **1 Write/s** to an EG.

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:



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

F1

Idea:

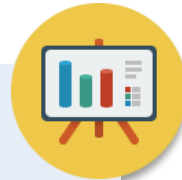
- Full SQL relational database built on Spanner, powers AdWords

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



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:

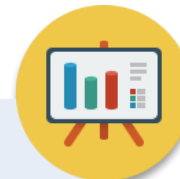
- Autosharded Entity Groups
- *Not* based on BigTable

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



ACID p
1 Write

Eventu
across

Row key
101
101,500
101,502
102

AE/Cloud DataStore

F1

Idea:

- Full SQL relational database built on Spanner, powers AdWords

Implementation:

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



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?

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

ACID p
1 Write

Idea:
• Autosharded Entity Groups
• Every row on BigTable
across

Implementation:

- TrueTime API (GPS + atomic clocks) → commit timestamps
 - Paxos replication per shard
- 101
101,502
102



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

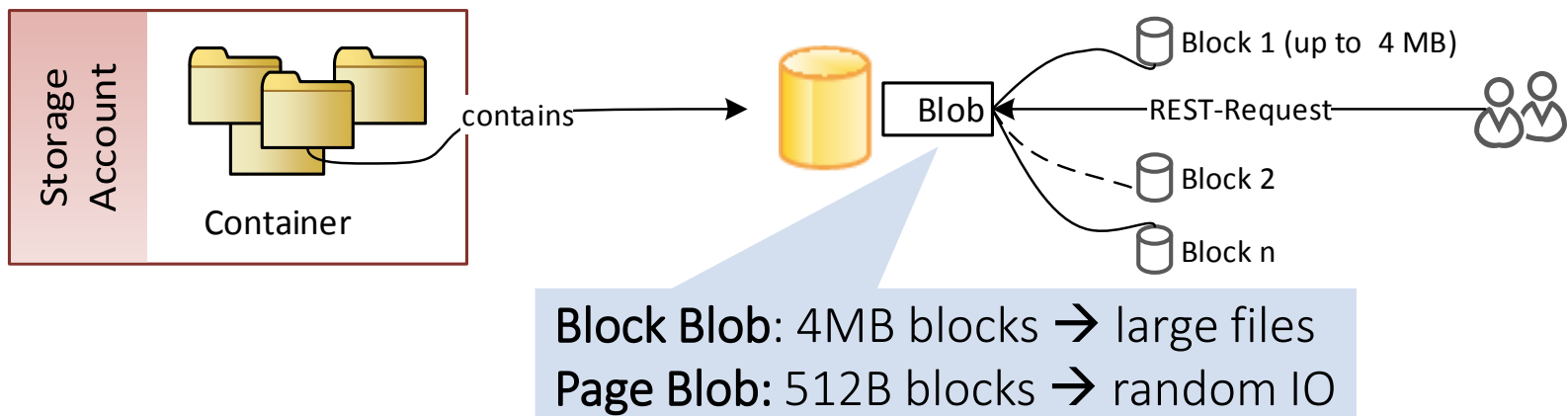


J. Shute, et al. "F1: A distributed SQL database that scales.", VLDB 2013

Azure Blobs, Amazon S3, Google Cloud Storage

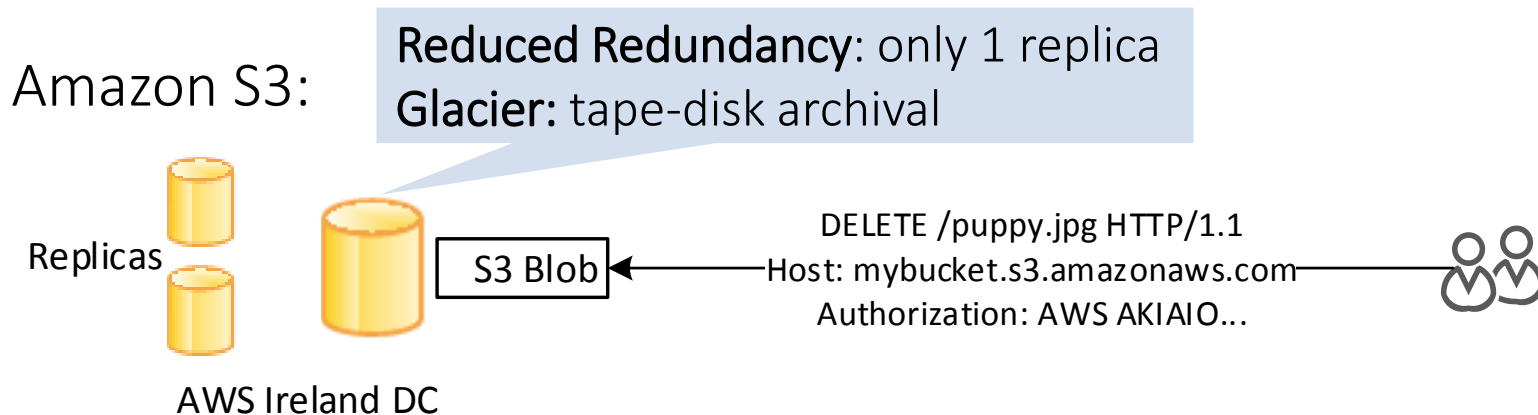
- ▶ **Idea:** Blobs with RESTful CRUD Interface
- ▶ **Distribution:** (*Bucket, Key*) → *Server + Replicas*
- ▶ CDN Integration (Azure CDN, Cloudfront)
- ▶ **S3:** > 2 Trillion ($2 \cdot 10^{12}$) objects

Azure Blobs:



Azure Blobs, Amazon S3, Google Cloud Storage

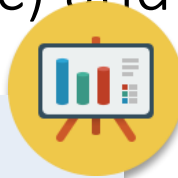
- ▶ Automatic Versioning
- ▶ **Pricing:** per request (10.000 ~ 1c) and storage (1GB/month ~ 3c) and network (1GB ~ 10c)



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

Redundancy: only 1 replica
disk archival

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

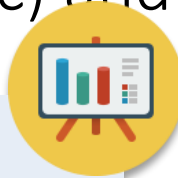


D. Bermbach,, S. Tai. "Eventual consistency:
How soon is eventual?", MW4SOC 11

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

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 Core



Parse Analytics

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

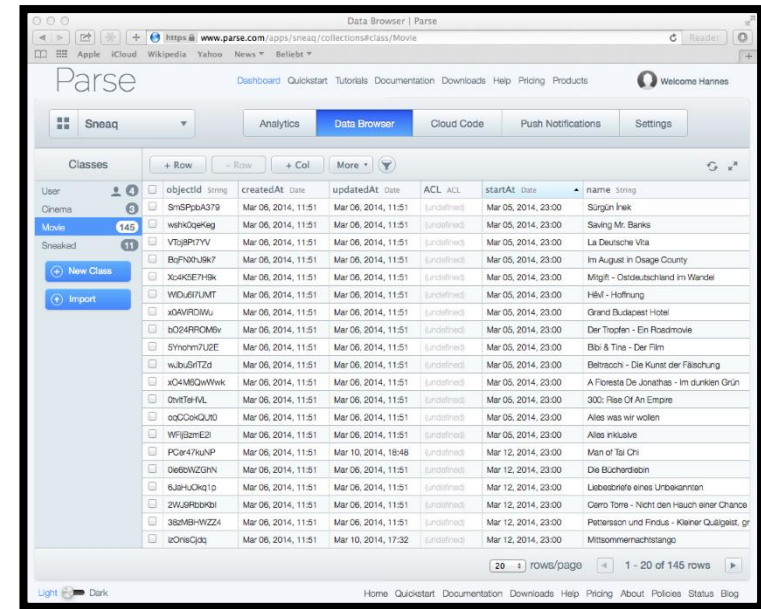
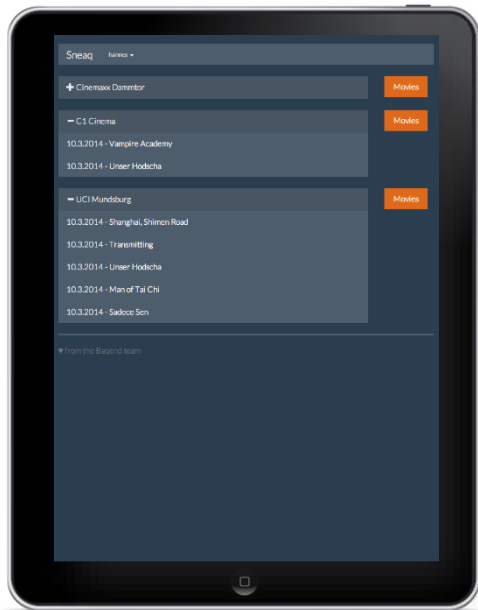


Parse Analytics

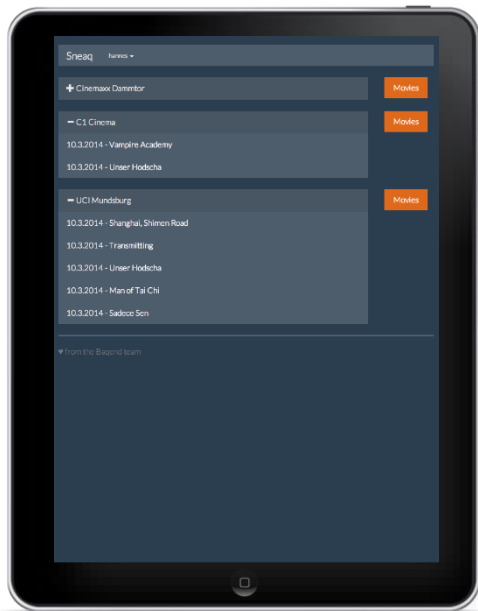


Parse Push

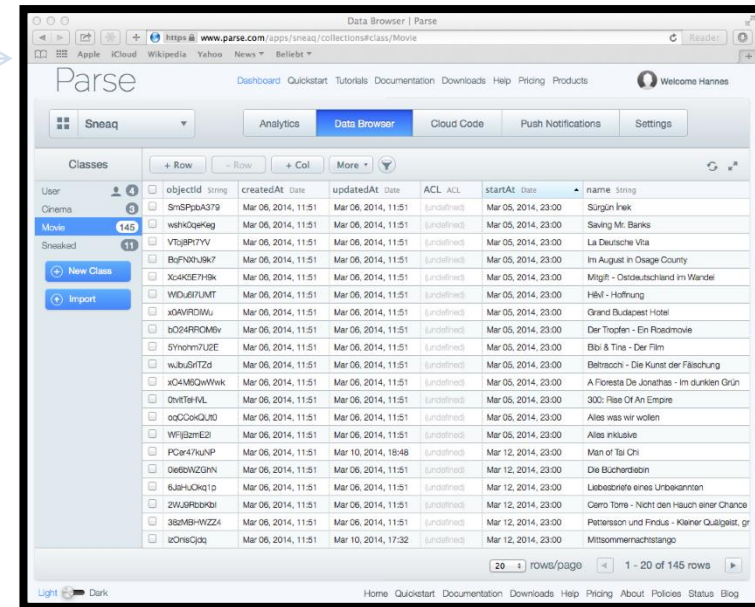
Parse - MBaaS



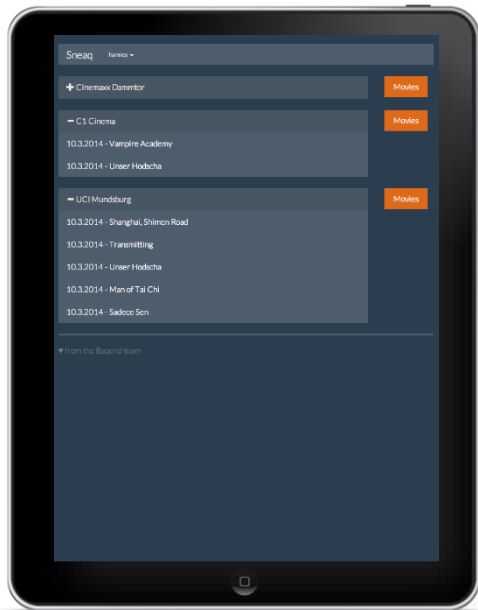
Parse - MBaaS



Authentication
User + Password
OAuth: Facebook, Twitter



Parse - MBaaS



Authentication

User + Password

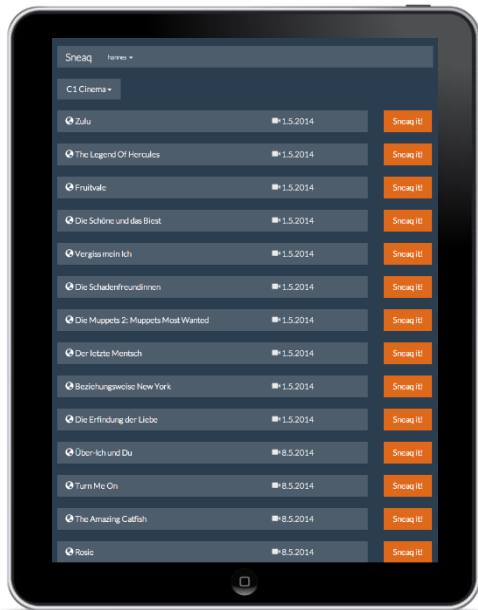
OAuth: Facebook, Twitter

Query Cinemas

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

objectid	class	createdAt	updatedAt	ACL	name
ShSPpBA379	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Sürgün Inn
WshKQeRag	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Saving Mr. Banks
Ytj8P7YV	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	La Deutsche Villa
BqFN0uJ9K7	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Im August in Osage County
Xu4KSE7H9k	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Mitgli - Ostdeutschland im Wandel
WDOu67UMT	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Höhl - Hoffnung
x0AVFCWu	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Grand Budapest Hotel
5Q24RRCMBv	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Der Trost - Ein Roadmovie
5Ynom7USE	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Blü & Tina - Der Film
wbu6G7Zd	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Belmosch - Die Kunst der Fälschung
x04MBQWwWek	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	A Foresta De Jonathan - Im dunklen Grün
0veTf9VL	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	300: Rise Of An Empire
oqCocQJUD	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Alles was wir wollen
WfjBzmE2	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Alles inklusive
PCe47uNP	Cinema	Mar 06, 2014, 11:51	Mar 10, 2014, 18:48	Unchanged	Man of Tai Chi
0e60WZGNN	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Die Bucherleien
6JahUQq1p	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Liebesbriefe eines Unbekannten
2WJ8Rbxbi	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Cerro Torre - Nicht den Hauch einer Chance
385MBWYZZ2	Cinema	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unchanged	Peterson und Findus - Kleiner Quiltgeist, gr
icOnsQDq	Cinema	Mar 06, 2014, 11:51	Mar 10, 2014, 17:32	Unchanged	Mitsommernachtstango

Parse - MBaaS



Authentication

User + Password

OAuth: Facebook, Twitter

Query Cinemas

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

Query Movies

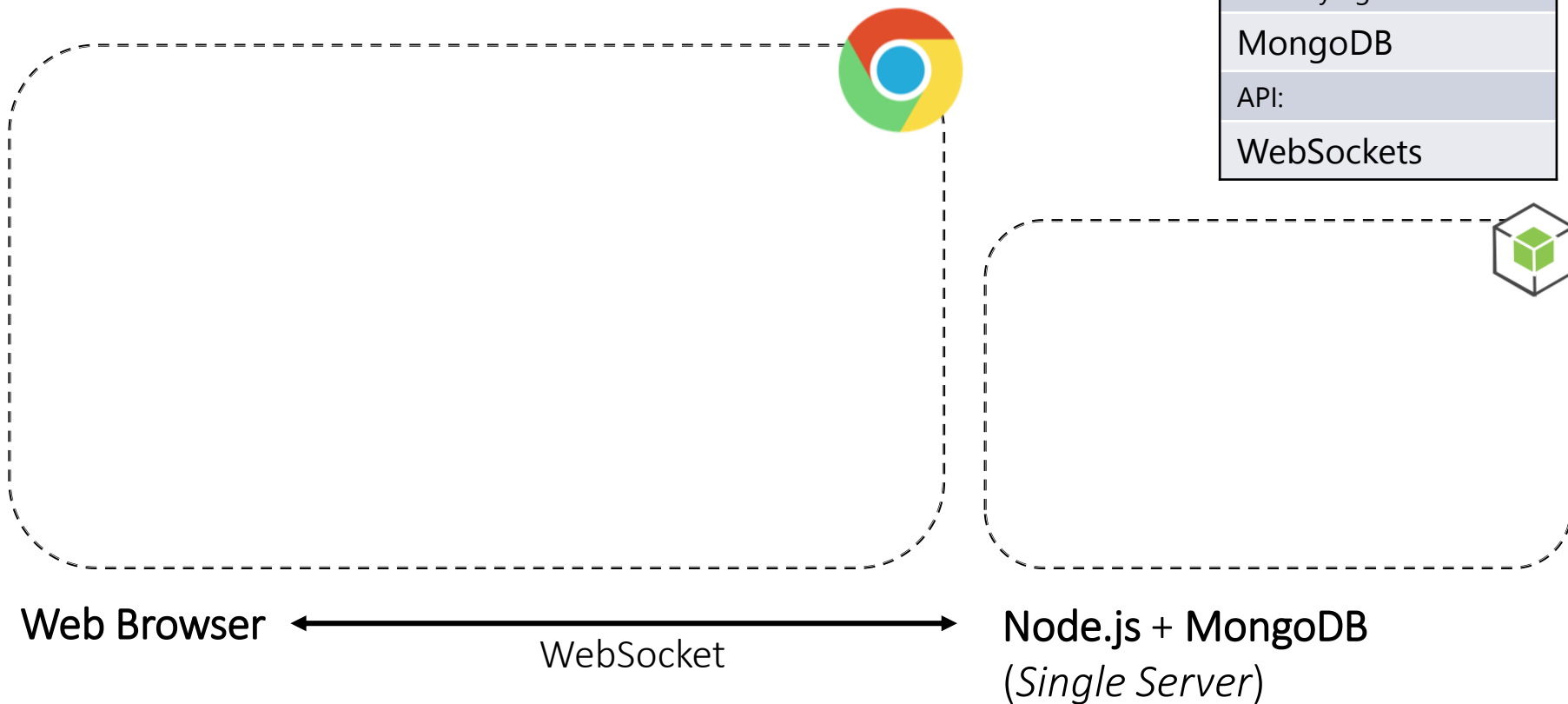
```
(new Parse.Query('Movies'))  
  .greaterThan('startAt', now)  
  .notEqualTo('cinemas', cld)  
  .fetch()
```

objectId	createdAt	updatedAt	ACL	startAt	name
ShEPpBa379	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Sorgün İnk
wshQqetq	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Saving Mr. Banks
Ytj8P7TV	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	La Deutsche Via
BqFN0uJ97	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Im August in Osage County
Xu4KE7H9k	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Mitgli - Ostdeutschland im Wandel
W0U6TUMT	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Höhl - Hoffnung
x0AVFCWu	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Grand Budapest Hotel
5Q24RRCMBv	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Der Trofen - Ein Roadmovie
SYnom7USE	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Blü & Tine - Der Film
wbu6t7Zd	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Beltrous - Die Kunst der Fälschung
x04MBQWwhek	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	A Foresta De Jonathan - Im dunklen Grün
0vEtFeHL	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	300: Rise Of An Empire
oqCocQJUD	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 05, 2014, 23:00	Ailes was wir wollen
WfjBzmE2	Mar 06, 2014, 11:51	Mar 10, 2014, 18:48	Unmodified	Mar 05, 2014, 23:00	Alles inklusive
PCe47uNP	Mar 06, 2014, 11:51	Mar 10, 2014, 18:48	Unmodified	Mar 12, 2014, 23:00	Man of Tai Chi
0e60WZGNN	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 12, 2014, 23:00	Die Bucherideen
6JahUOKq1p	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 12, 2014, 23:00	Liebesbriefe eines Unbekannten
2WJ8R0xb8i	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 12, 2014, 23:00	Cerro Torre - Nicht den Hauch einer Chance
385MB-HYZZ2	Mar 06, 2014, 11:51	Mar 06, 2014, 11:51	Unmodified	Mar 12, 2014, 23:00	Peterson und Findus - Kleiner Quiltgeist, gr
icOnsQdq	Mar 06, 2014, 11:51	Mar 10, 2014, 17:32	Unmodified	Mar 12, 2014, 23:00	Mitsommernachtstango

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

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



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>
```



```
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}});
  };

```

Web Browser

WebSocket



```

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

if (Meteor.isServer) {
  //...

```

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) {
  // Server-side code
}
```



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

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>
```



Very productive for very small projects

Fundamentally limited scalability:

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



```
if (Meteor.isClient) {
  Template.leaderboard.helpers({
    return Players.find({}, {
      sort: {score: -1},
      limit: 10
    });
  });
}

if (Meteor.isServer) {
  Meteor.startup(function() {
    Players = new Meteor.Collection("players");
    Meteor.subscribe('leaderboard');
    Meteor.publish('leaderboard', function() {
      return Players.find({});
    });
  });
}

$ meteor deploy abc.meteor.com
```

Web Browser

websocket

(Single Server)

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



Encrypted Databases: Research

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

SQL-Proxy

Encrypts and decrypts, rewrites queries



RDBMS

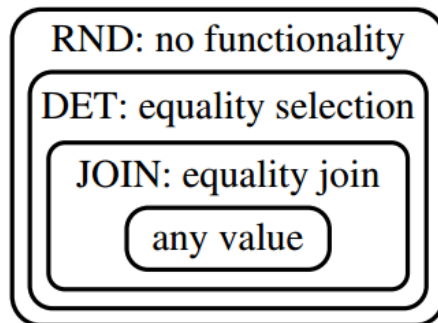
Encrypted Databases: Research

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

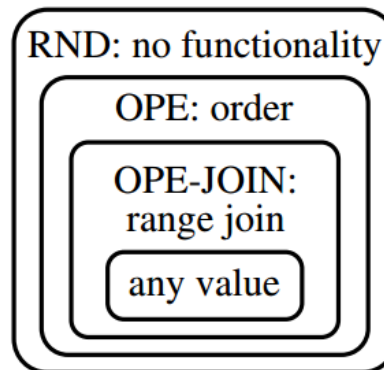
SQL-Proxy

Encrypts and decrypts, rewrites queries

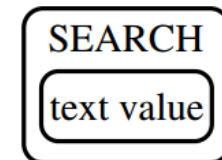
RDBMS



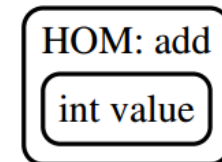
Onion Eq



Onion Ord



Onion Search



Onion Add

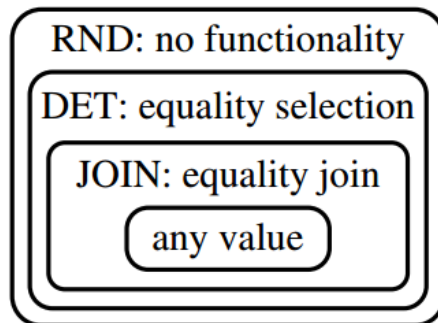
Encrypted Databases: Research

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

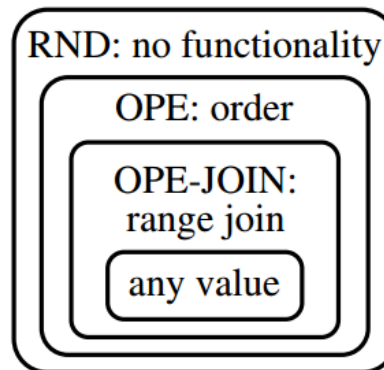
SQL-Proxy

Encrypts and decrypts,

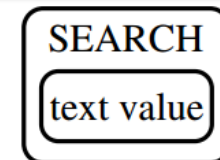
RDBMS



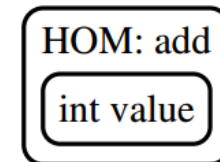
Onion Eq



Onion Ord



Onion Search



Onion Add

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



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



RDBMS



- Early approach
- Not adopted in practice, yet

Dream solution:
Full Homomorphic Encryption

RND: no functionality
ET: equality selection

OIN: equality join

OPE-JOIN:
range join

any value

SEARCH

text value

Onion Search

HOM: add

int value

Onion Add

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-Committed	Multi-Record	1 RT	-

Transactions/Consistency: Research

Multi-Data Center Consistency

Consistency

Commit

Data

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.



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

Transactions/Consistency: Research

Multi-Data Center Consistency

Consistent

Commit

Data

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

Dynamo

Eventual

Yahoo PNuts

Timeline pe

COPS

Causality

MySQL (async)

Serializable

Megastore

Serializable

Spanner

Snapshot

MDCC

Read-Committed



Currently no NoSQL DB implements consistent Multi-DC replication



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

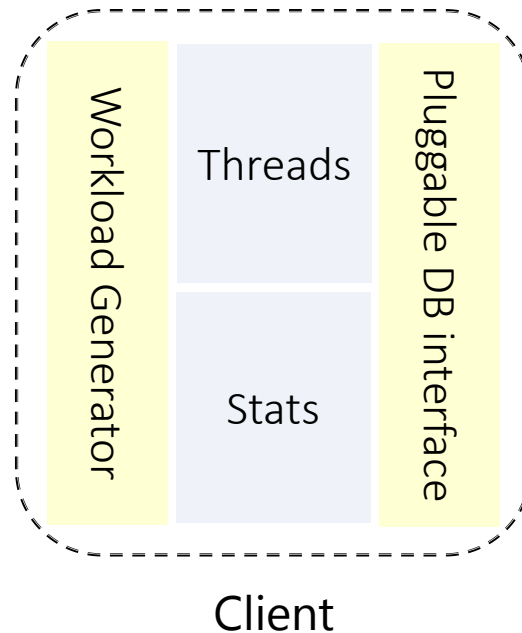
Benchmarking: Research

- ▶ YCSB (Yahoo Cloud Serving **B**enchmark)



Benchmarking: Research

- ▶ YCSB (Yahoo Cloud Serving Benchmark)



Benchmarking: Research

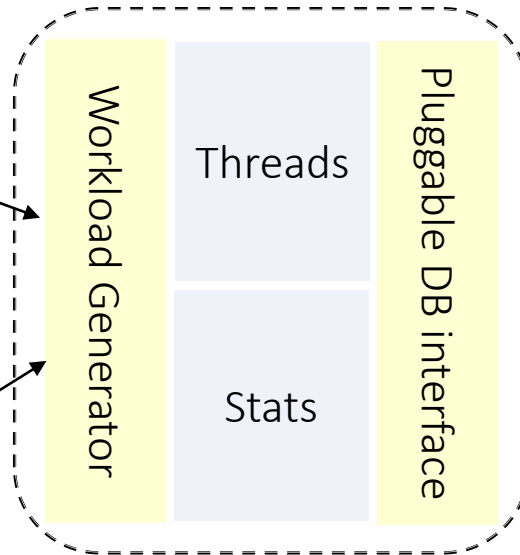
► YCSB (Yahoo Cloud Serving Benchmark)

Runtime Parameters:

DB host name,
threads, etc.

Workload:

1. Operation Mix
2. Record Size
3. Popularity Distribution

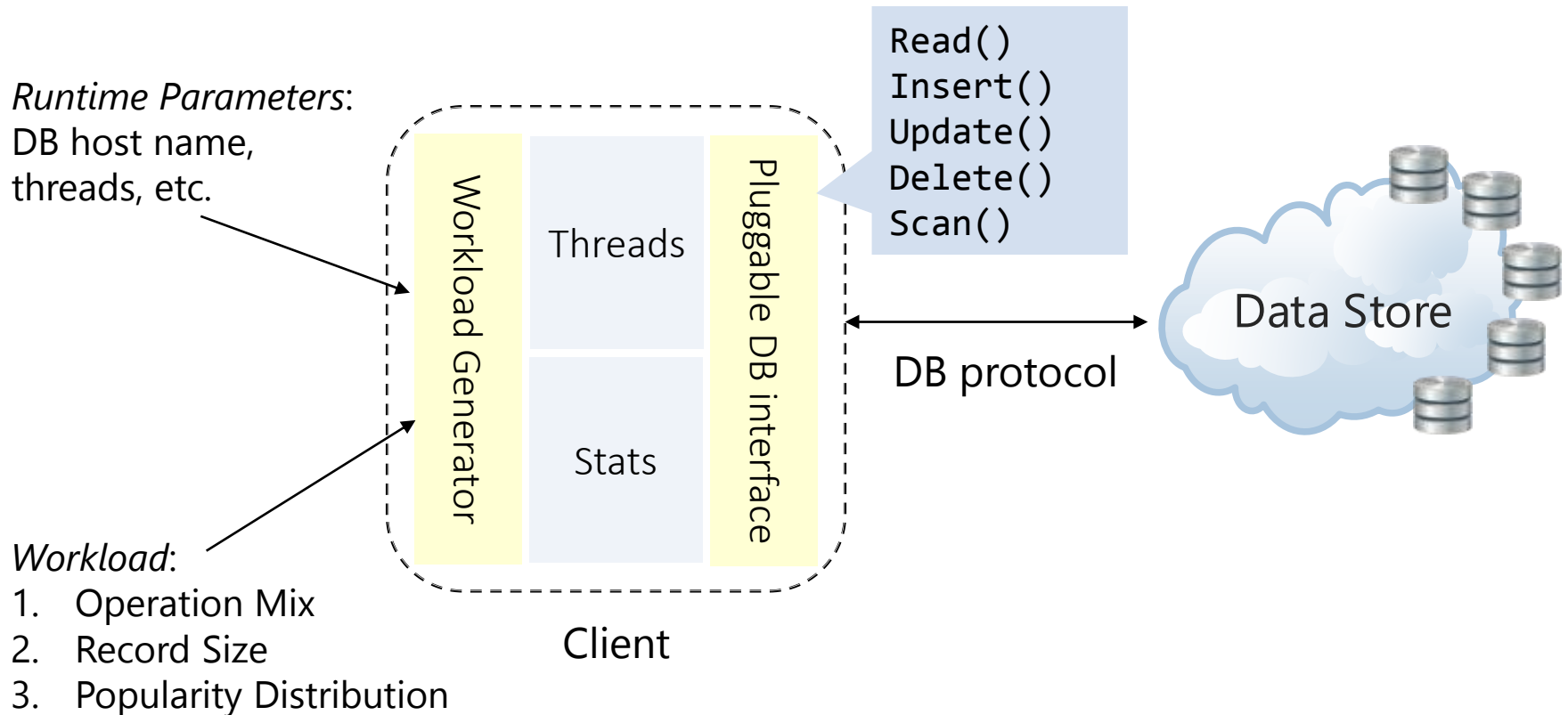


Client



Benchmarking: Research

► YCSB (Yahoo Cloud Serving Benchmark)

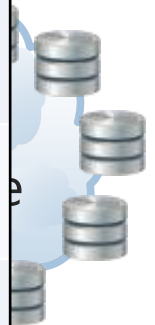


Benchmarking: Research

► YCSB (Yahoo Cloud Serving Benchmark)

Read()

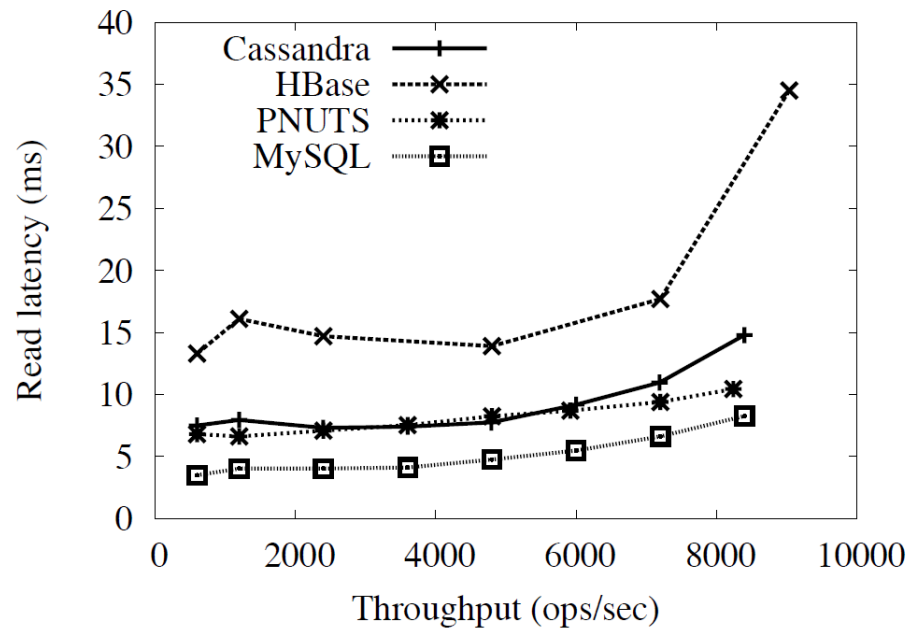
Workload	Operation Mix	Distribution	Example
A – Update Heavy	Read: 50% Update: 50%	Zipfian	Session Store
B – Read Heavy	Read: 95% Update: 5%	Zipfian	Photo Tagging
C – Read Only	Read: 100%	Zipfian	User Profile Cache
D – Read Latest	Read: 95% Insert: 5%	Latest	User Status Updates
E – Short Ranges	Scan: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



3. Popularity Distribution

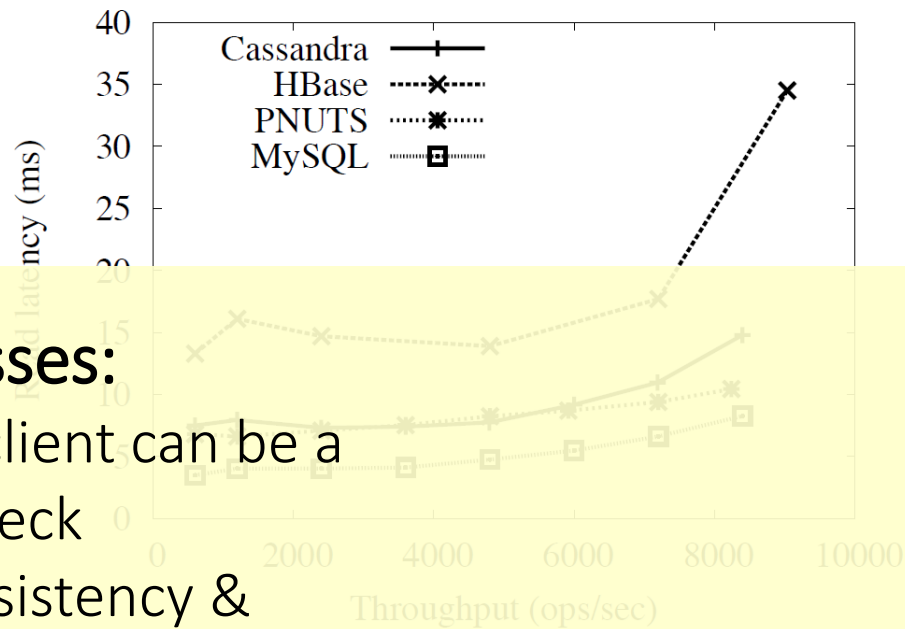
Benchmarking: Research

▶ Example Result (Read Heavy):



Benchmarking: Research

▶ Example Result (Read Heavy):



Weaknesses:

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

Benchmarking: Research

YCSB++



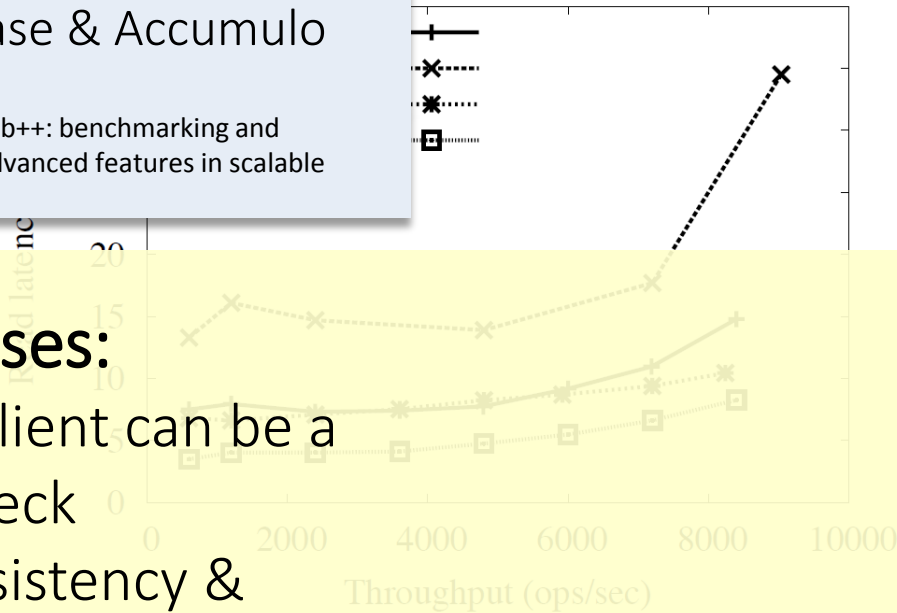
- Clients coordinate through Zookeeper
- Simple Read-After-Write Checks
- Evaluation: Hbase & Accumulo



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

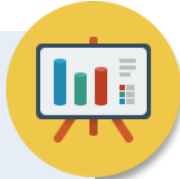


YCSB+T

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

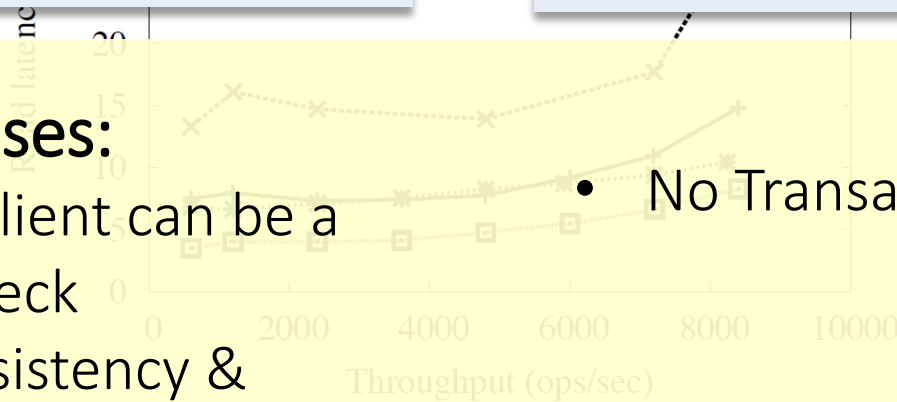


A. Dey et al. “YCSB+T: Benchmarking Web-Scale Transactional Databases”, CloudDB 2014



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

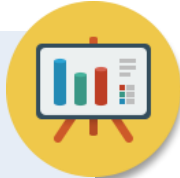


YCSB+T

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

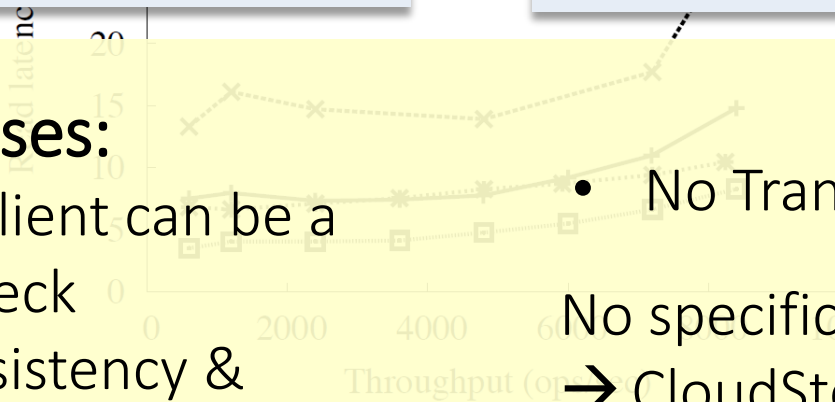


A. Dey et al. “YCSB+T: Benchmarking Web-Scale Transactional Databases”, CloudDB 2014



Weaknesses:

- Single client can be a bottleneck
 - No consistency & availability measurement
 - No Transaction Support
- No specific application
→ CloudStone, CARE, TPC extensions?



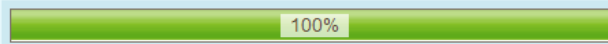
Benchmarking: state of the art



 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.

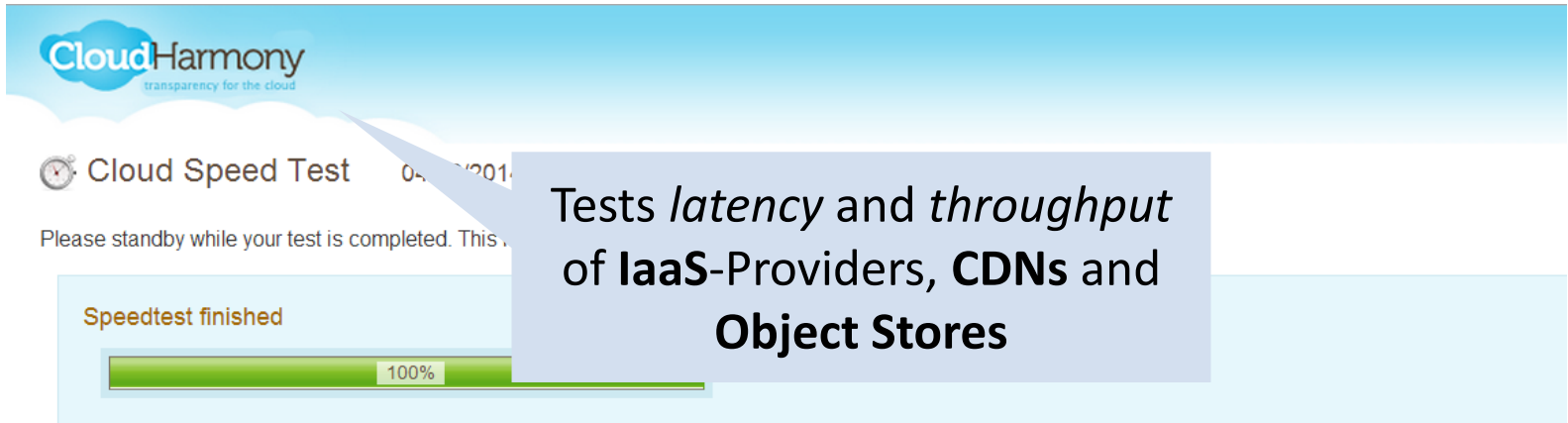
Speedtest finished



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

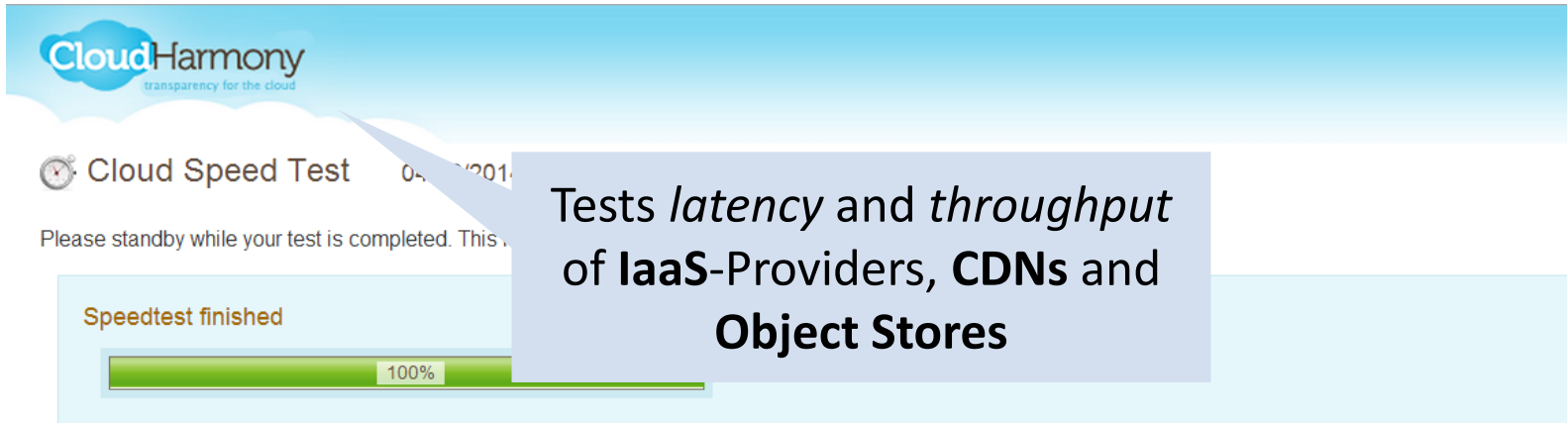


The screenshot shows the CloudHarmony Cloud Speed Test interface. At the top, the CloudHarmony logo is displayed with the tagline 'transparency for the cloud'. Below the logo, the text 'Cloud Speed Test' is visible, followed by a date '05/10/2014'. A message states 'Please standby while your test is completed. This...' with an ellipsis. A green progress bar indicates 'Speedtest finished' at 100%. A blue callout box with a pointer to the progress bar contains the text: '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



The screenshot shows the CloudHarmony Cloud Speed Test interface. At the top is the CloudHarmony logo with the tagline 'transparency for the cloud'. Below it, the text 'Cloud Speed Test' is followed by a date '05/10/2014'. A message says 'Please standby while your test is completed. This will take approximately 1 minute.' A green progress bar is shown with '100%' and the text 'Speedtest finished'. A blue callout box points to the progress bar and contains the text: '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

Does not test: Cloud Databases

Amazon CloudFront		0.04	1	35	35	0%	35	35
Akamai CDN		0.03	1	30	30	0%	30	30

Vision



Idea:

Vision



Idea:

1. Periodically launch clients

Vision

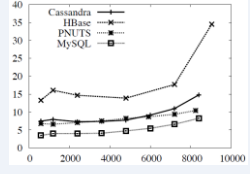
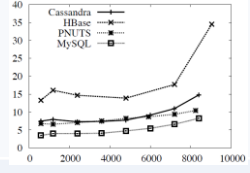
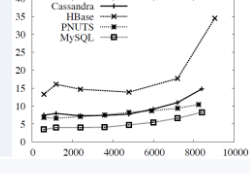
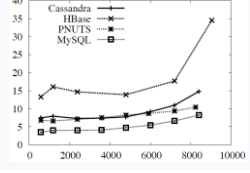


Idea:

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

Vision

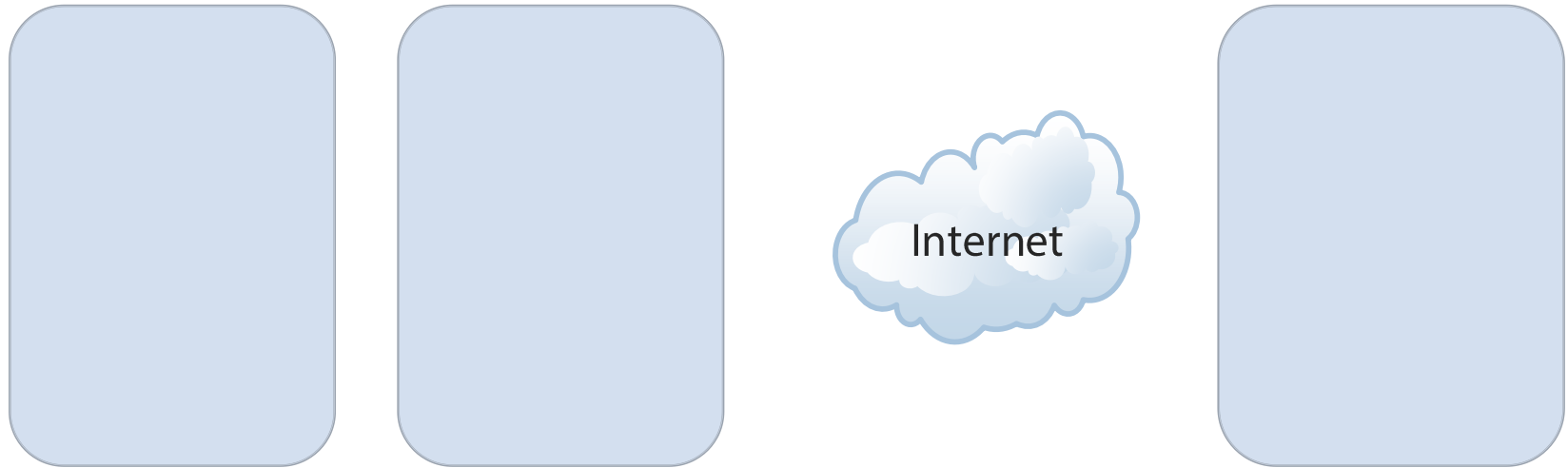


System	Availability	Reads/s	Writes/s	Avg. Latency	95th perc. Latency	Plot
DynamoDB	99.9%	23411	34534	3.2 ms	9 ms	
RDS	99.8%	2342	2455	30 ms	80 ms	
Azure Table	99.5%	22343	23442	12 ms	20 ms	
Google DataStore	99.5%	3000	2000	30 ms	300 ms	

A person with long hair is seen from behind, sitting on a pier or boat, looking out at a harbor. In the background, several large port cranes are visible, illuminated by warm lights. The scene is set during sunset or sunrise, with a warm orange and yellow glow reflecting on the water. The overall atmosphere is serene and contemplative.

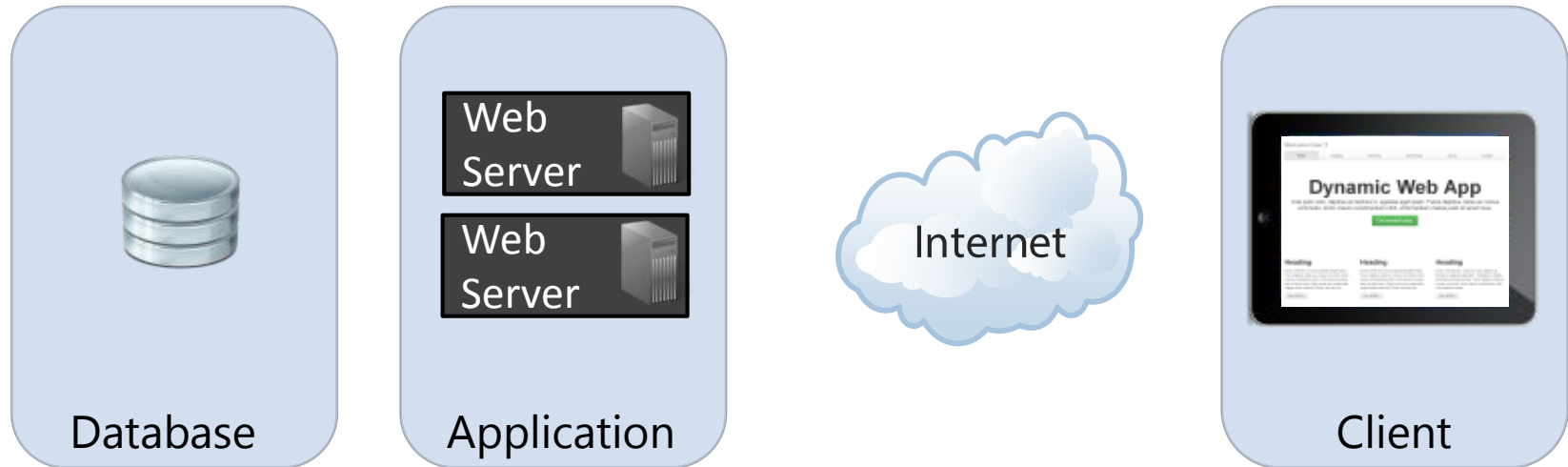
Database research at the
University of Hamburg

Motivation



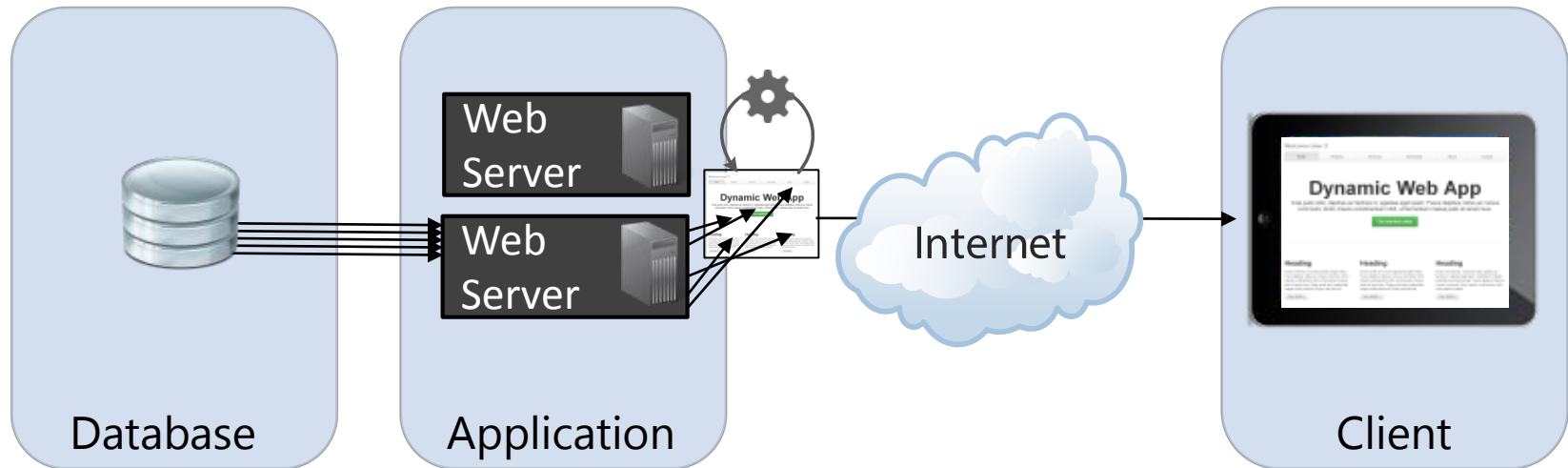
Three-Tier Architecture

Motivation



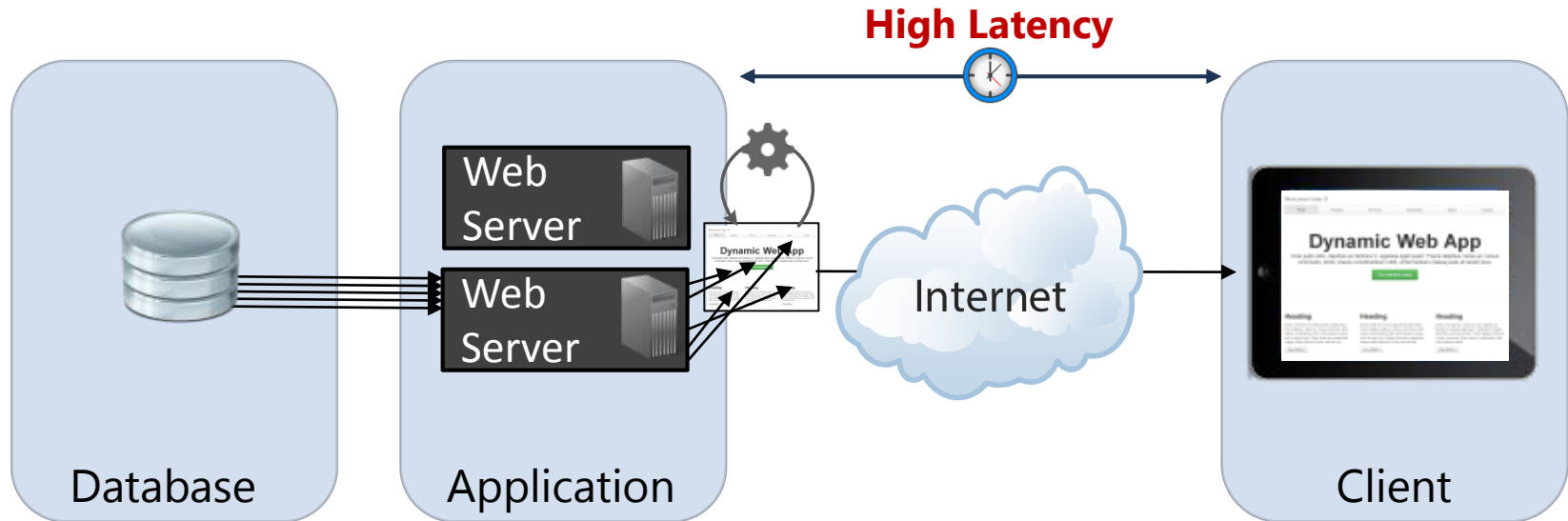
Three-Tier Architecture

Motivation



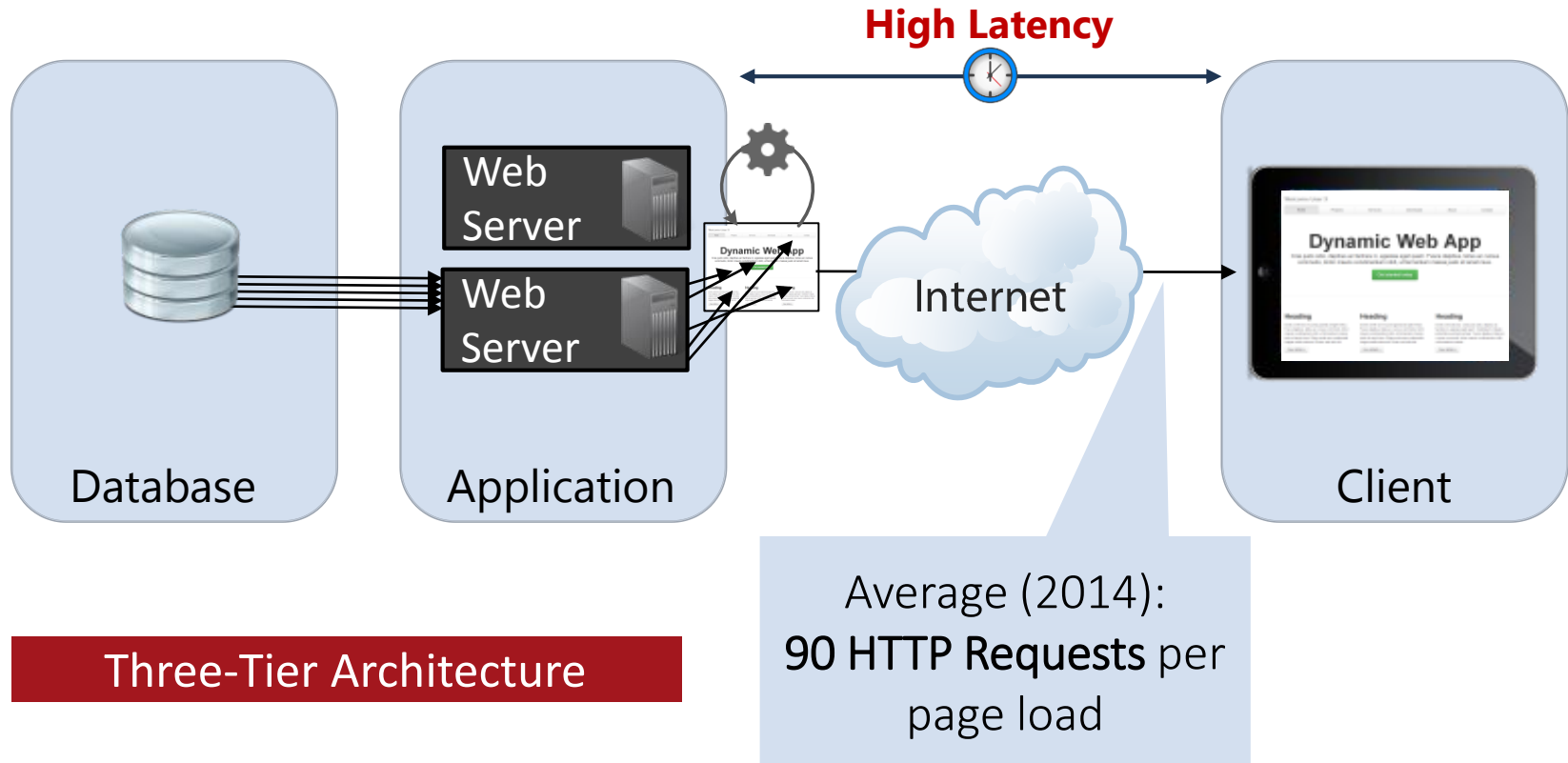
Three-Tier Architecture

Motivation

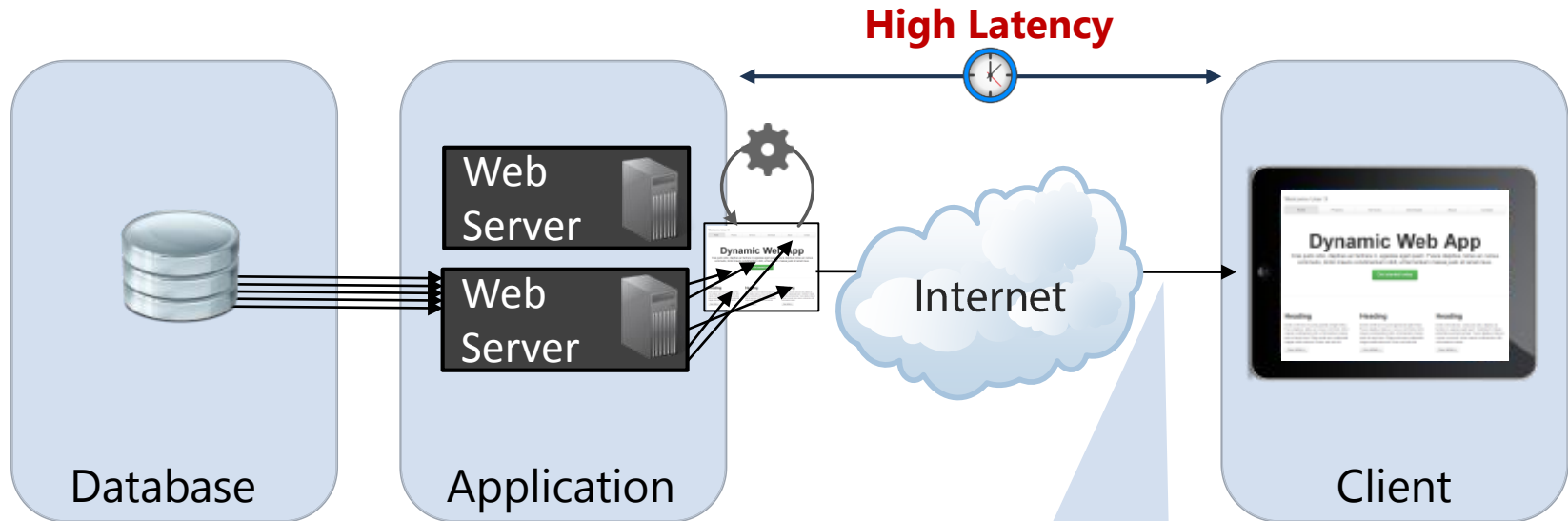


Three-Tier Architecture

Motivation



Motivation

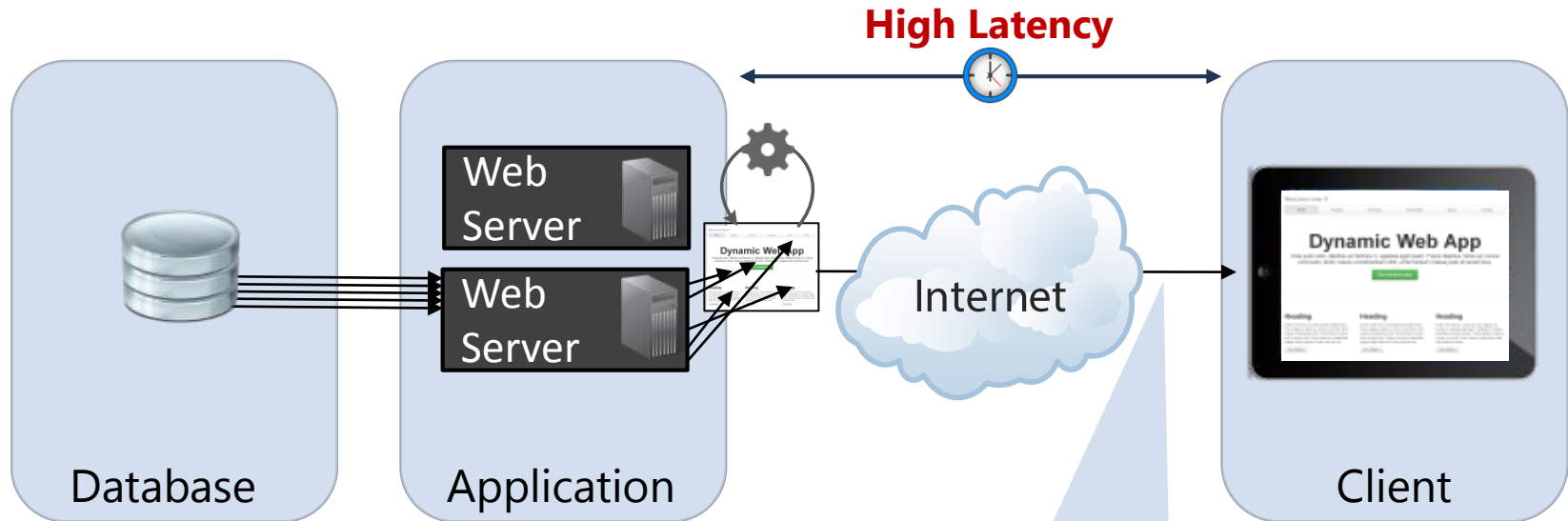


With every **100ms** of additional page load time, revenue decreases by 1%.



Study by **Amazon**

Motivation

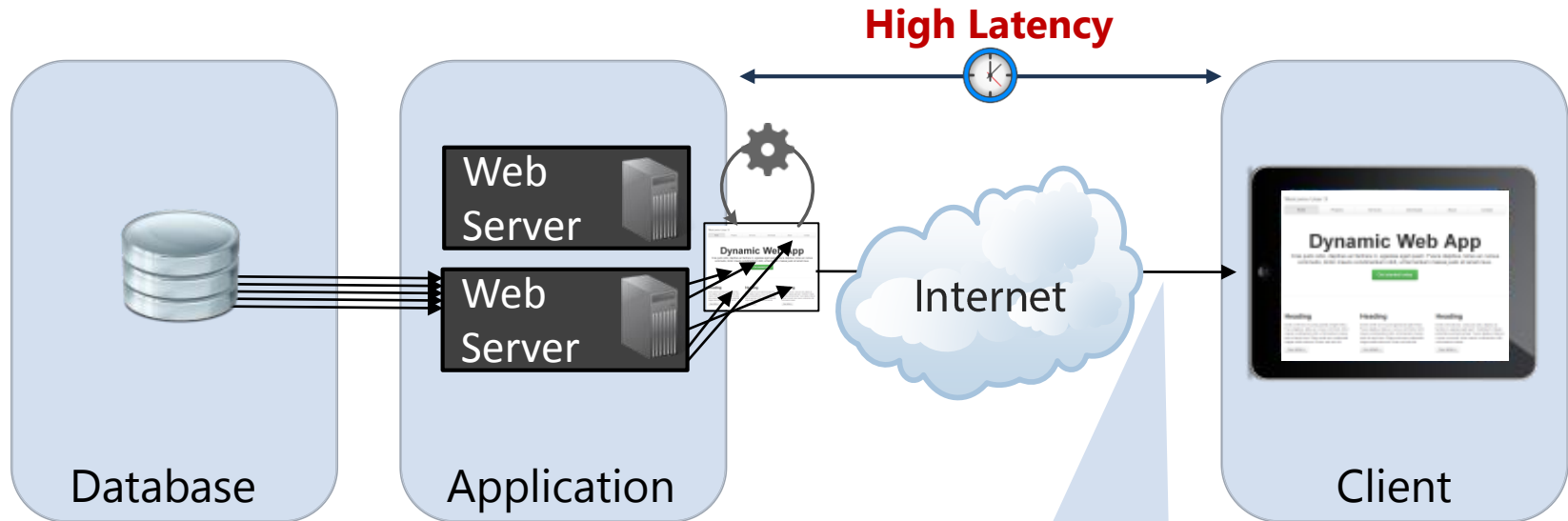


When increasing load time of search results by **500ms**, traffic decreases by 20%.



Study by **Google**

Motivation

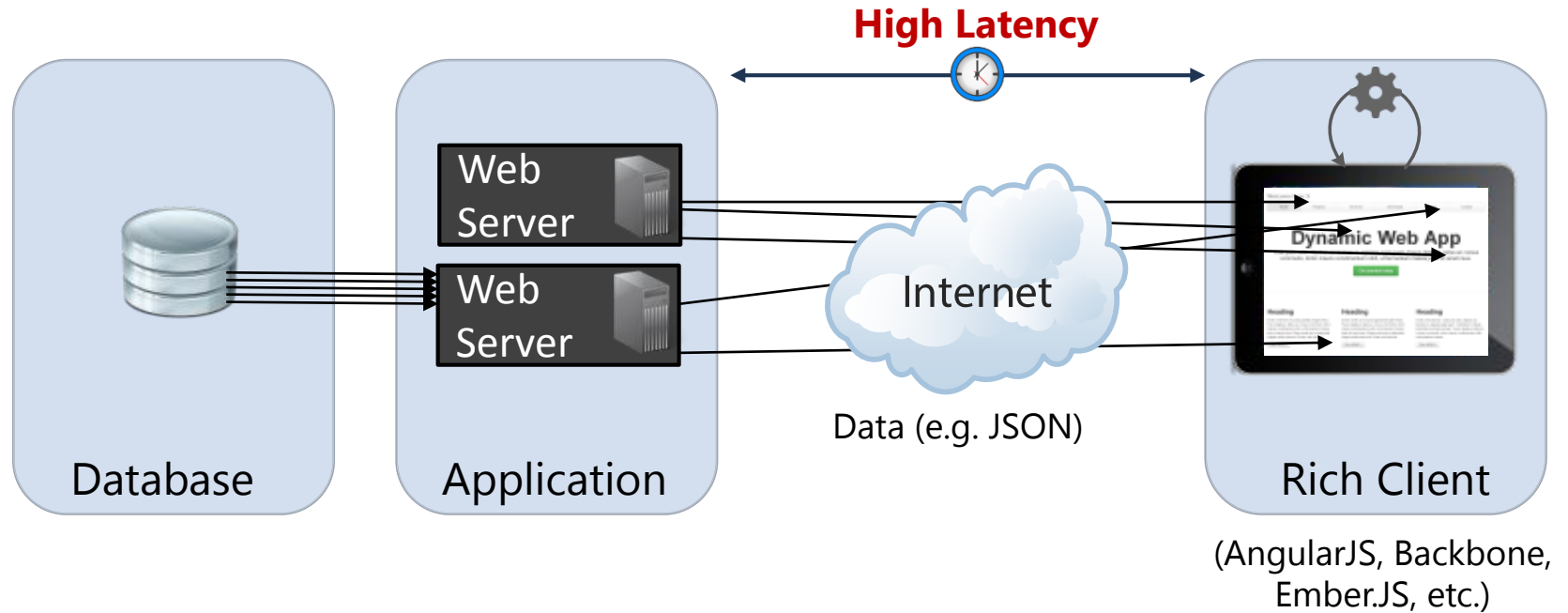


When increasing load time of search results by **500ms**, traffic decreases by 20%.



Study by **Google**

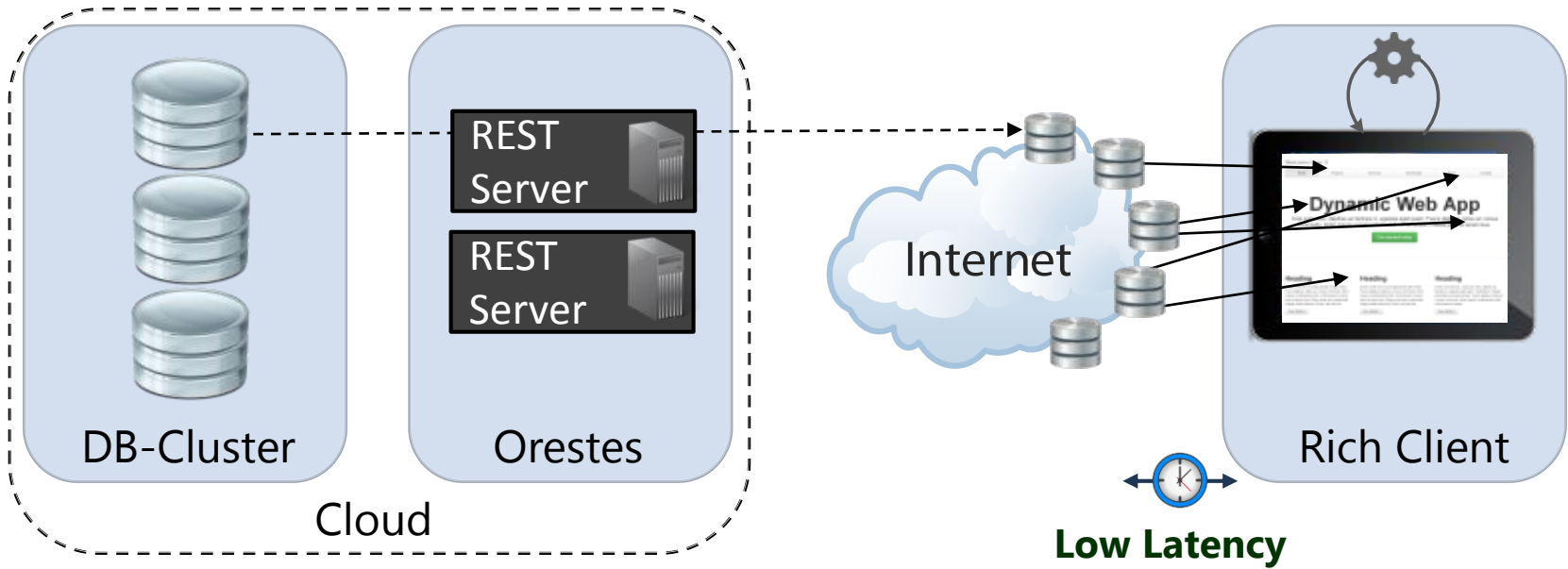
SPAs



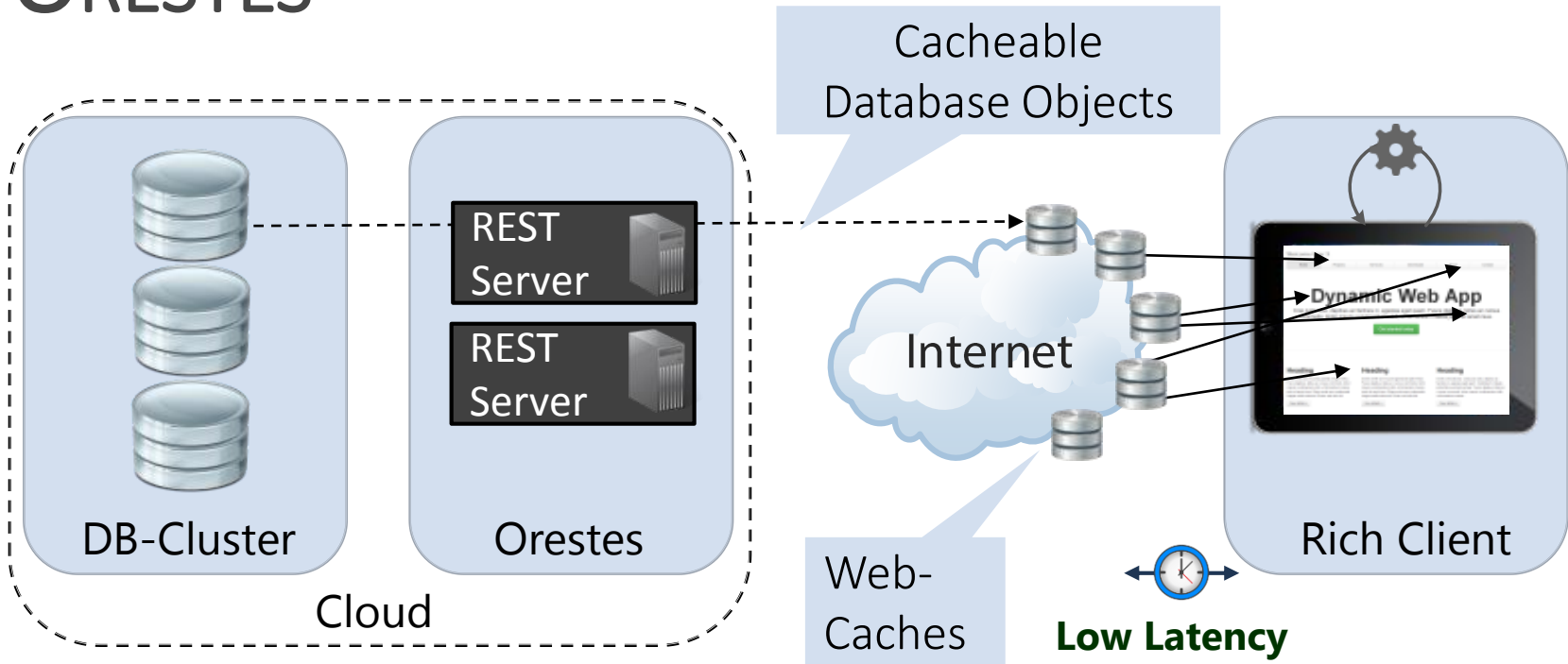
Single-Page Applications

(Rich Client, Smart Client)

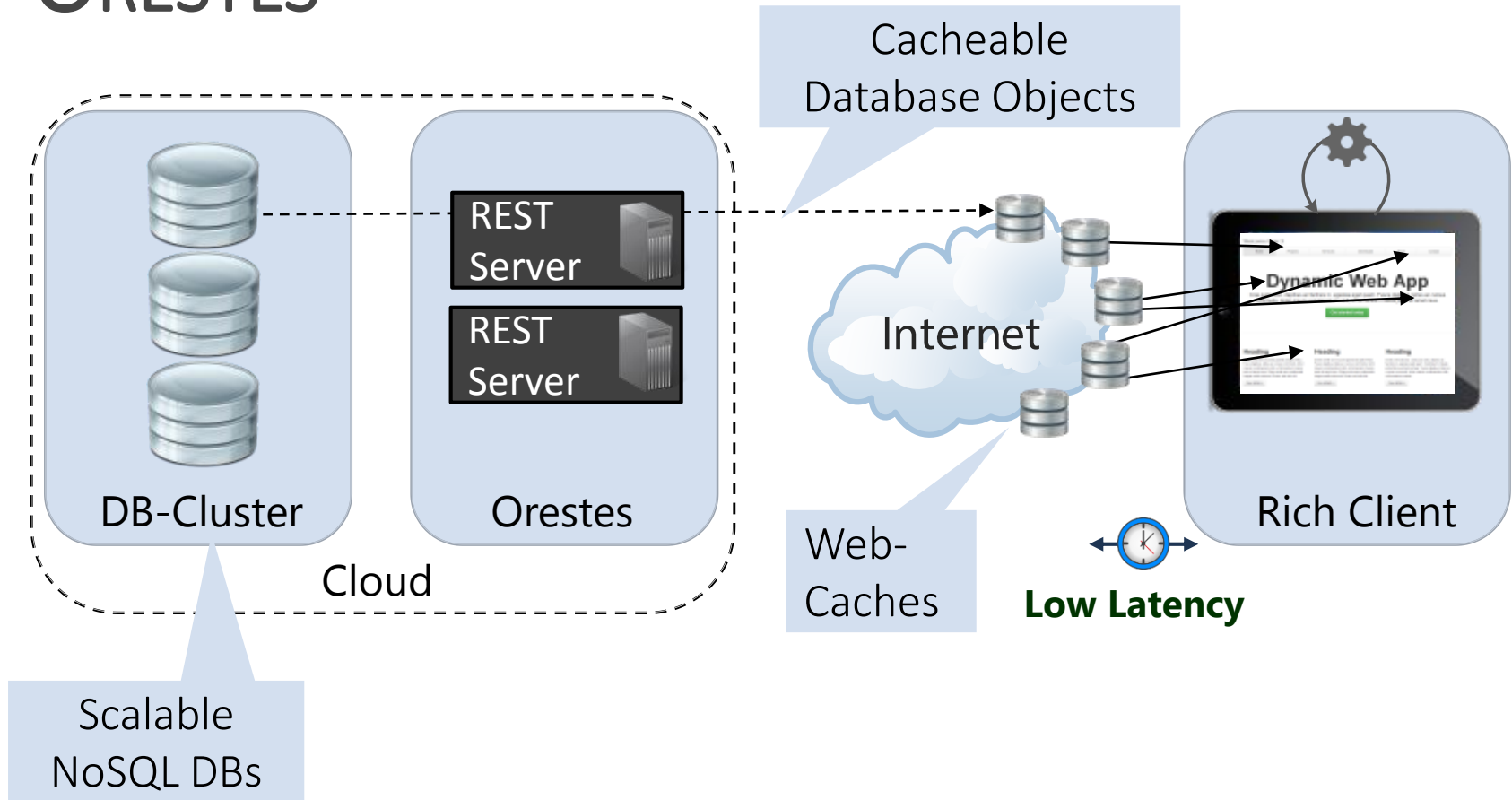
ORESTES



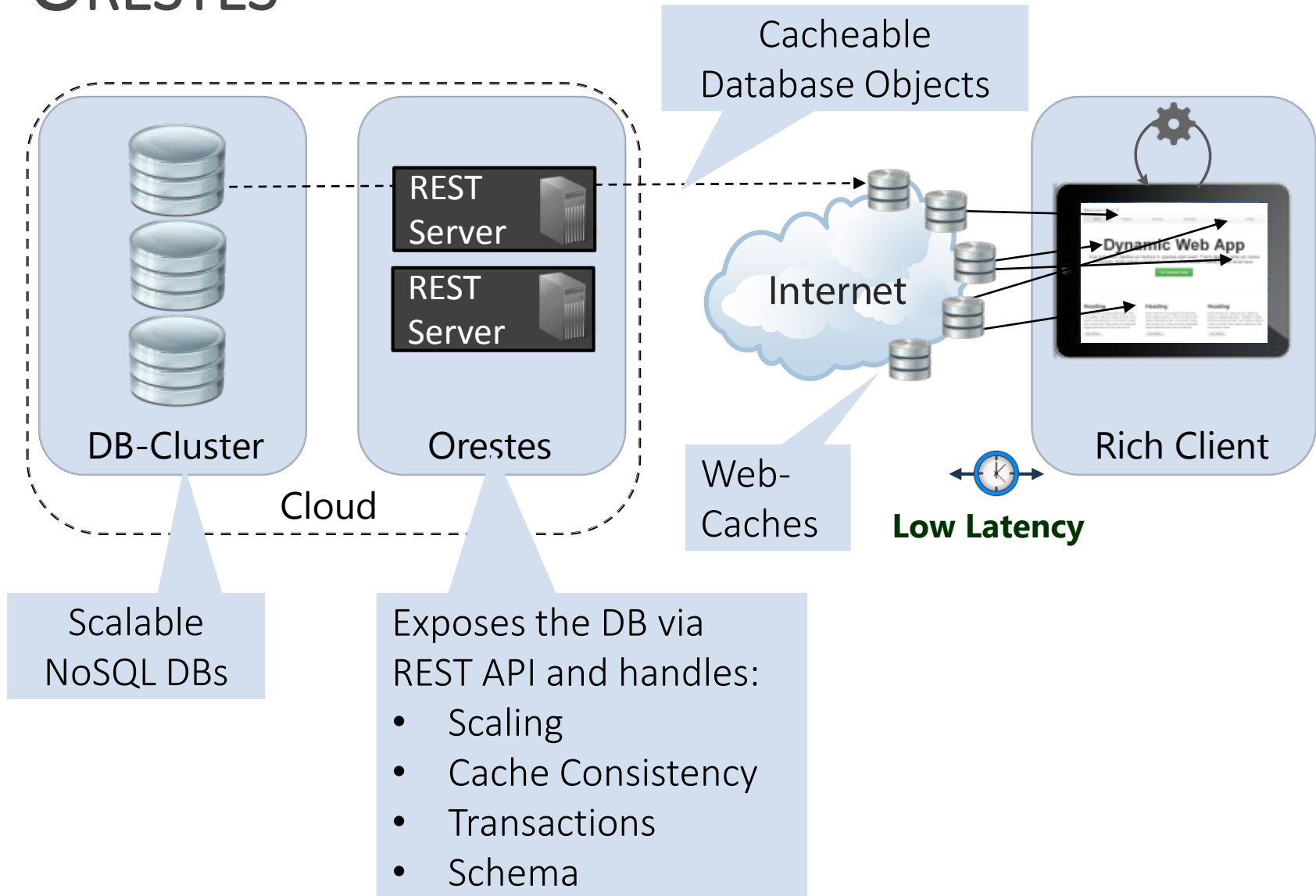
ORESTES



ORESTES



ORESTES



ORESTES: Components

▶ **Middleware:**

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



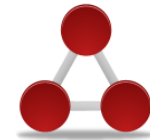
▶ **Caching:**

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



▶ **Transactions:**

- Optimistic cache-aware transaction model.



Application
Layer

Application
Server



Browser or
Mobile Device



Persistence
API

Java/JDO

JavaScript/JPA Port

REST/HTTP API

persist

find

createQuery

Data
Store

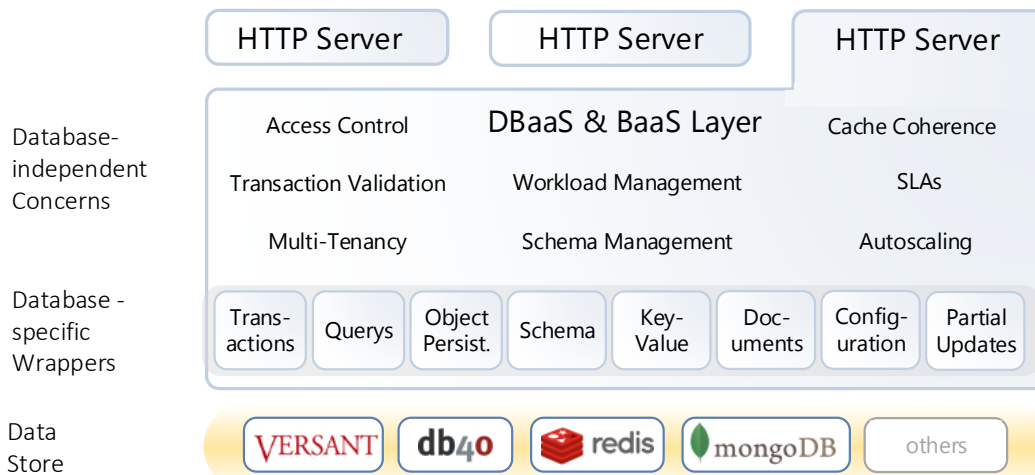
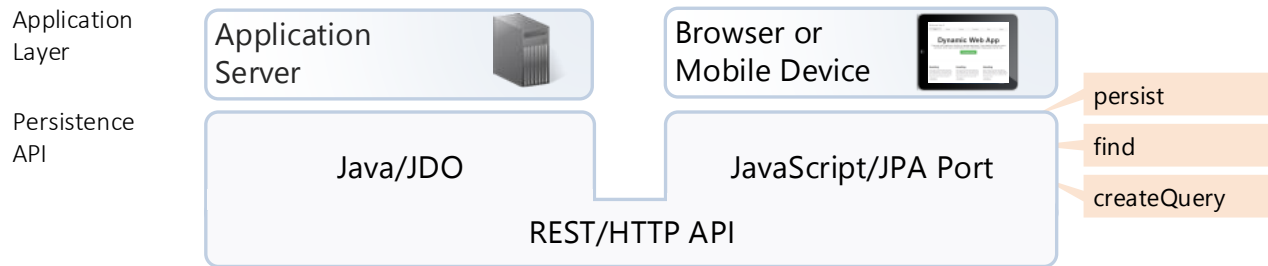
VERSANT

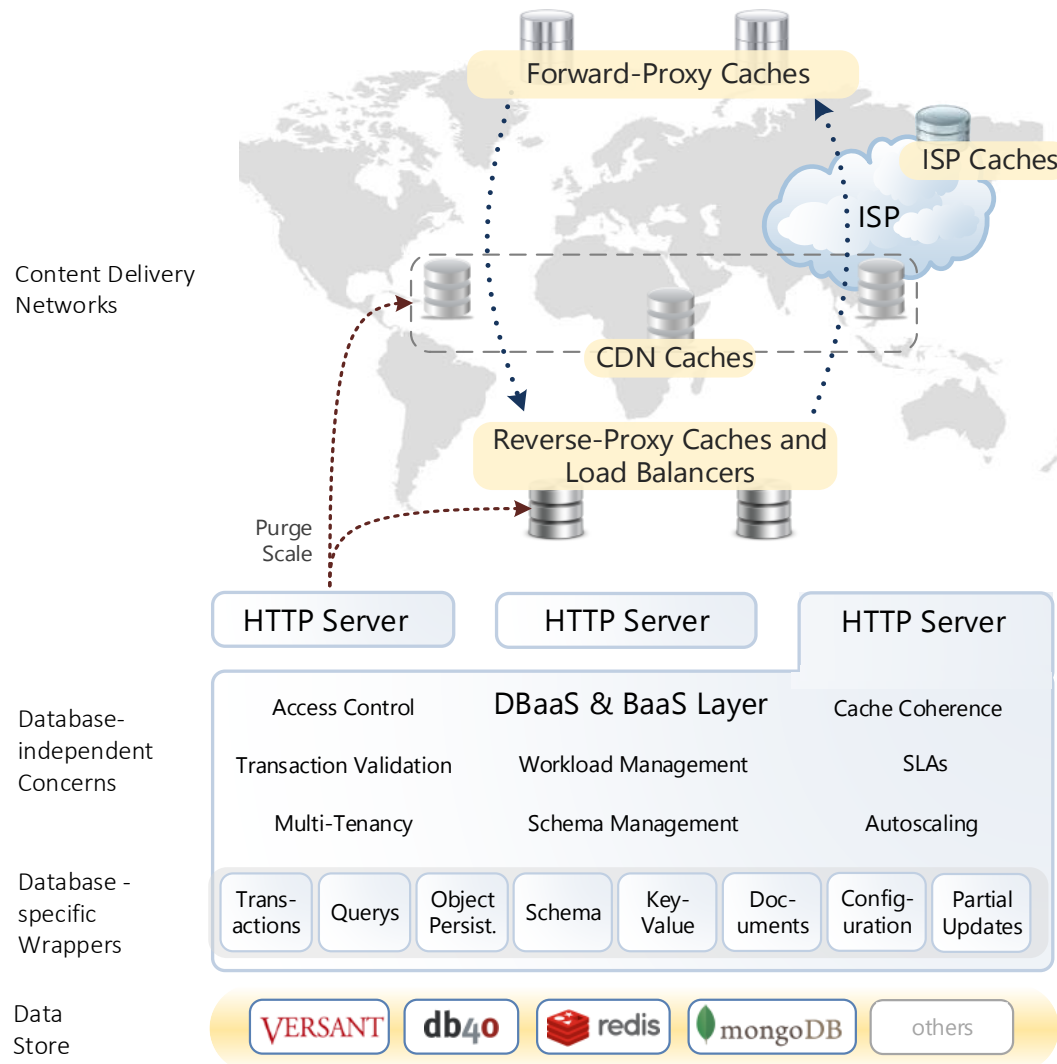
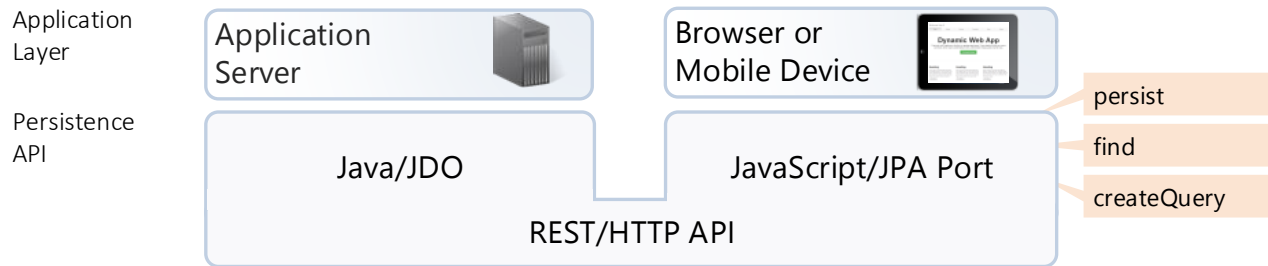
db4o

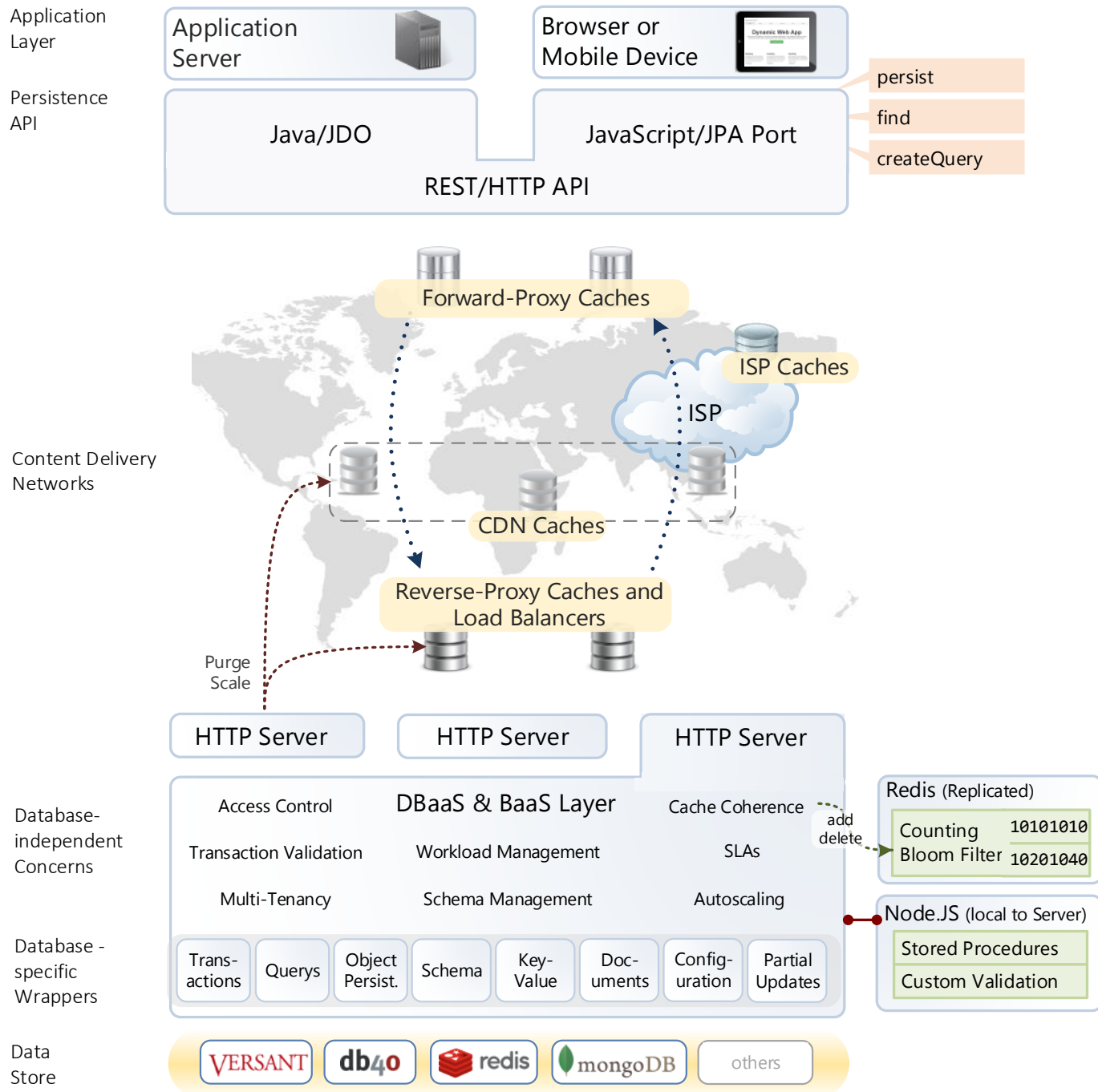
redis

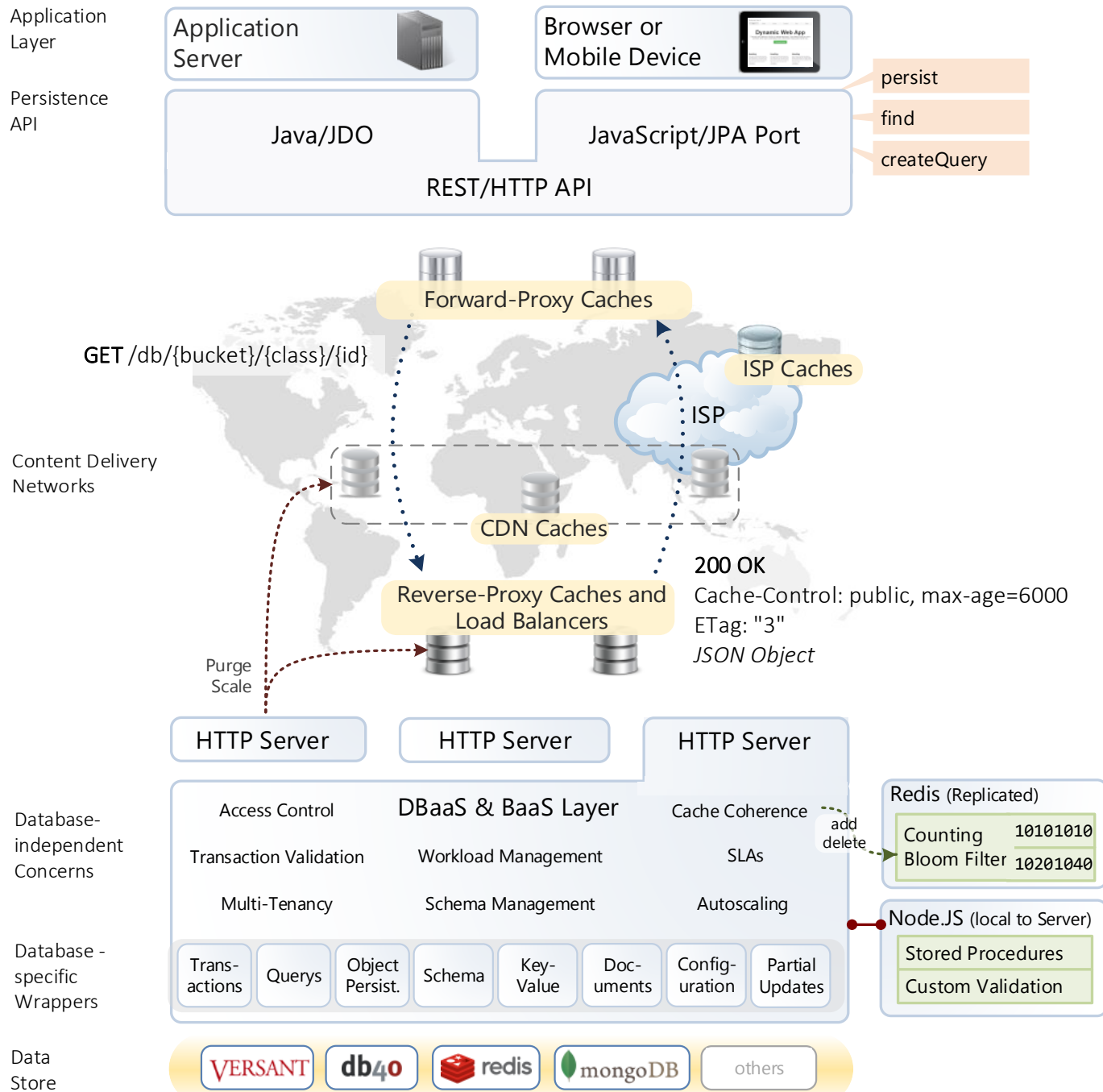
mongoDB

others

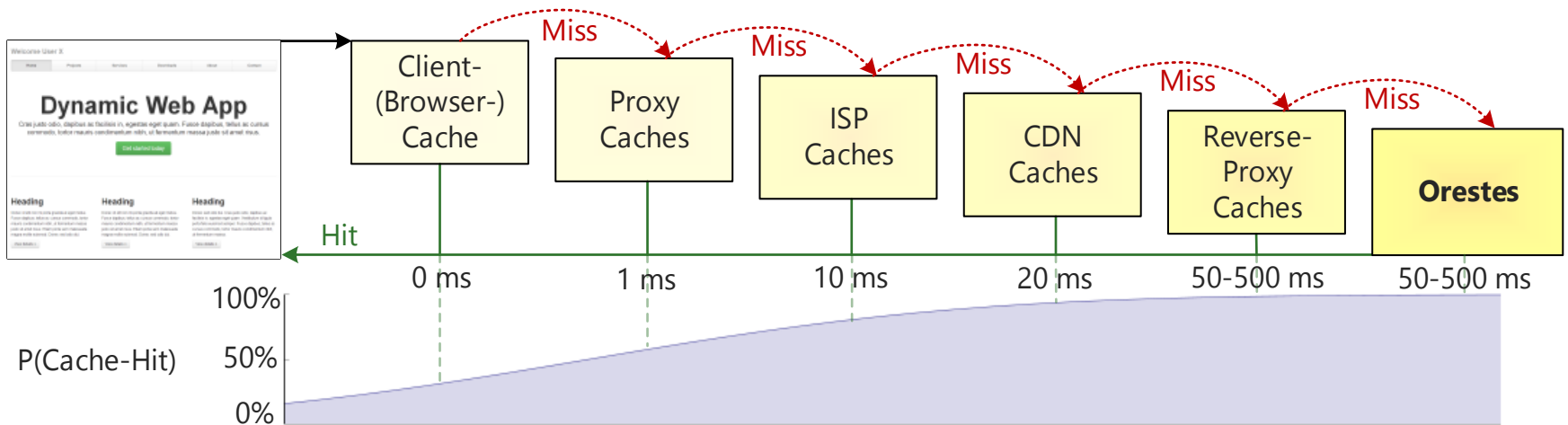




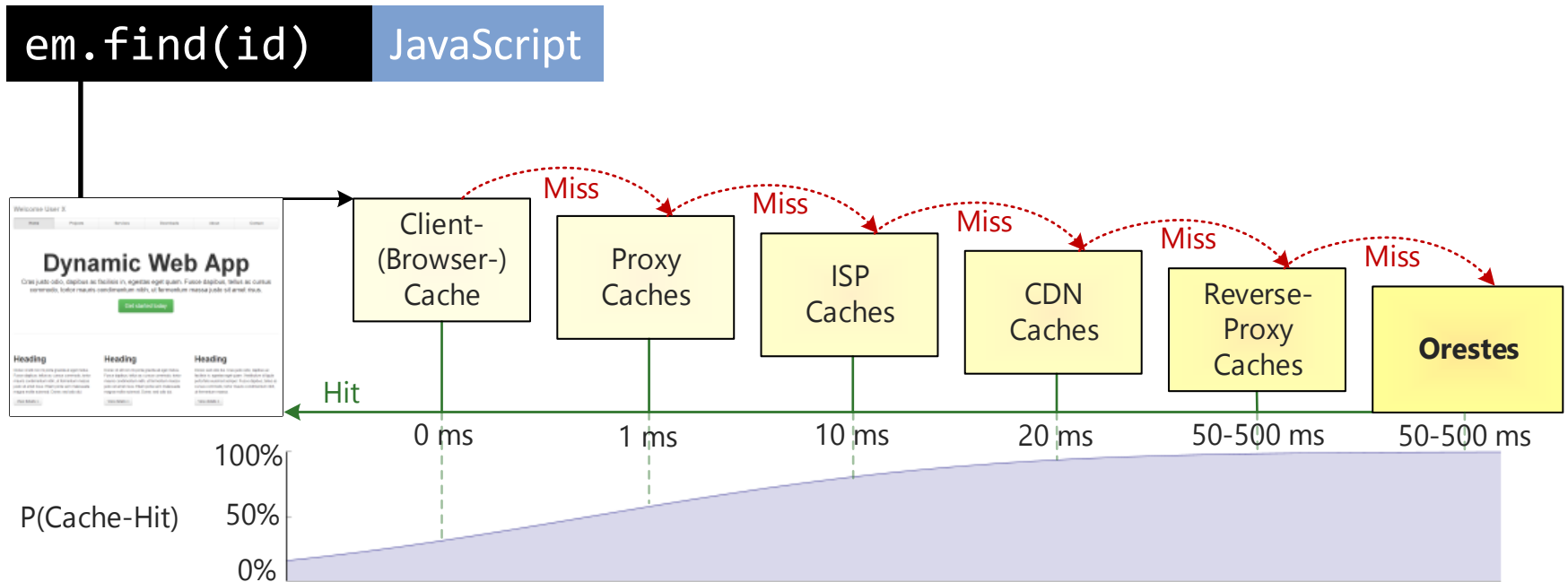




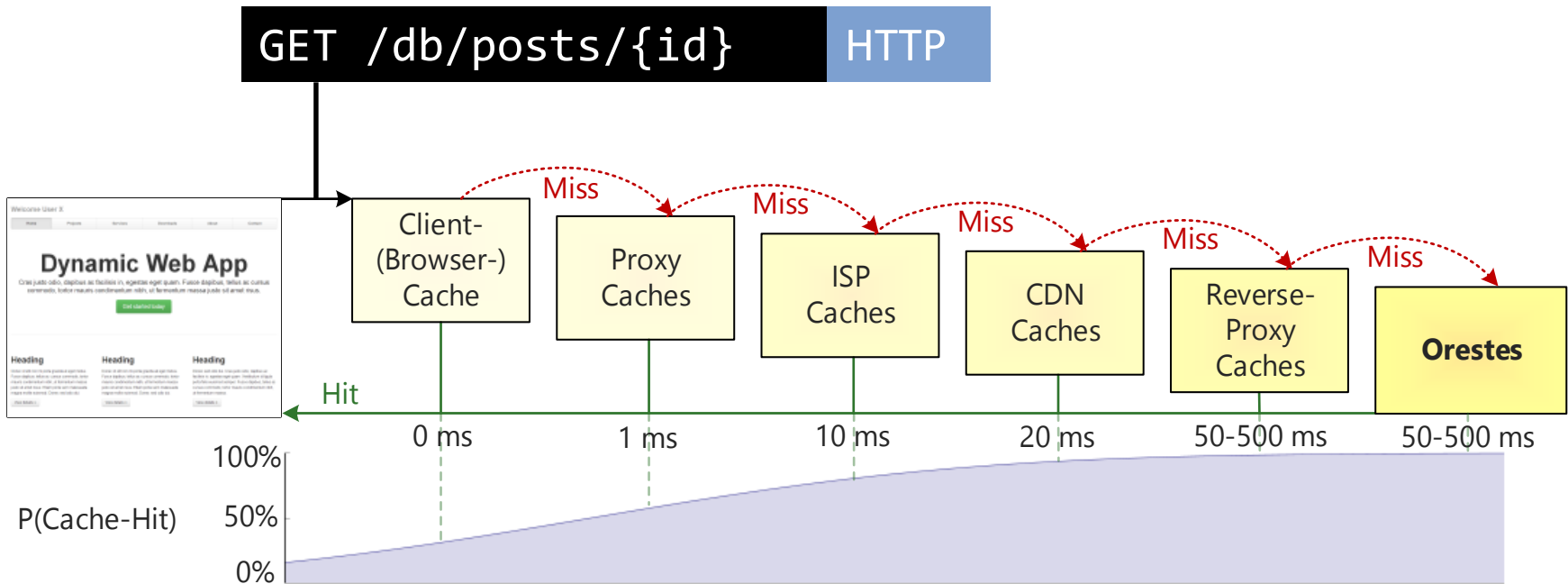
Caching



Caching

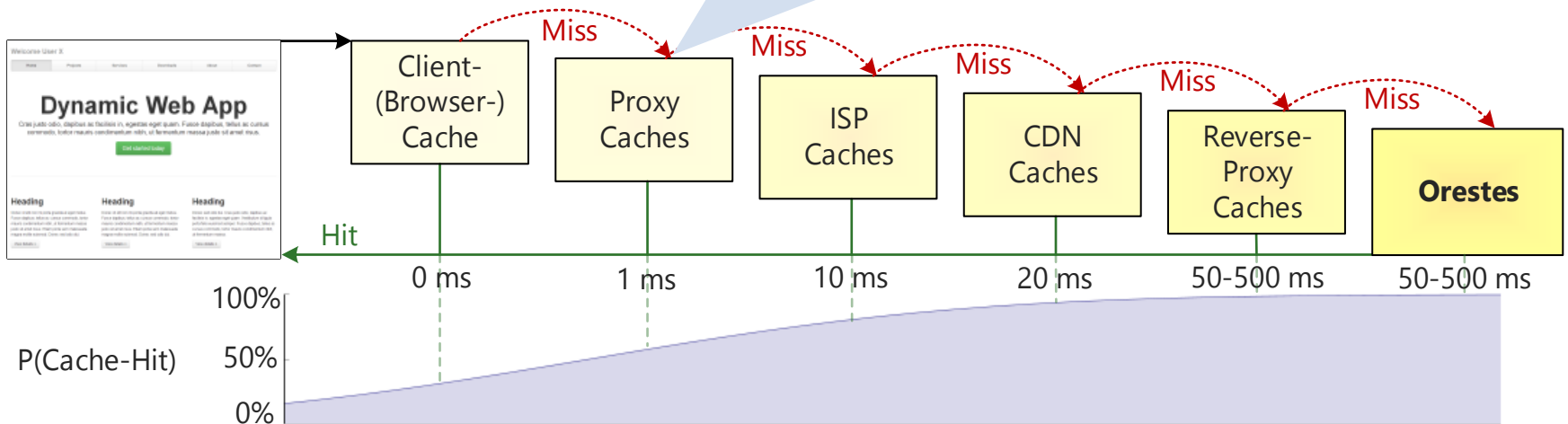


Caching

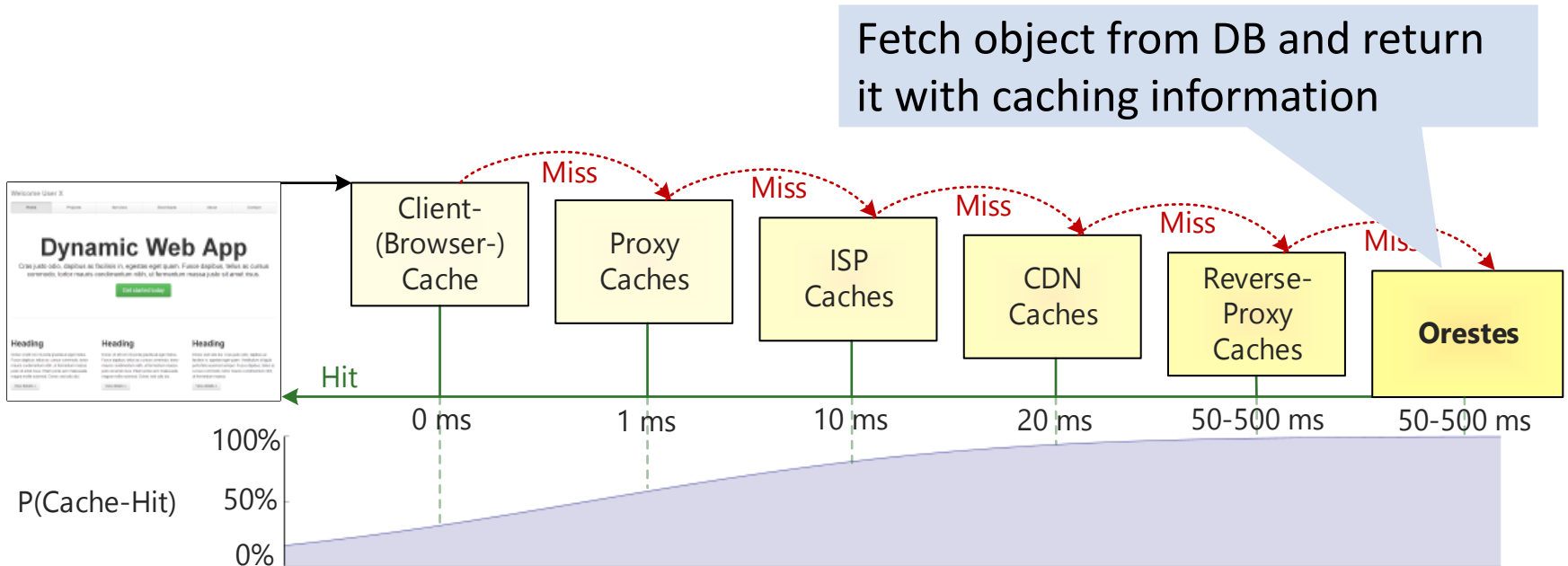


Caching

Cache-Hit: Return Object
Cache-Miss: Forward Request

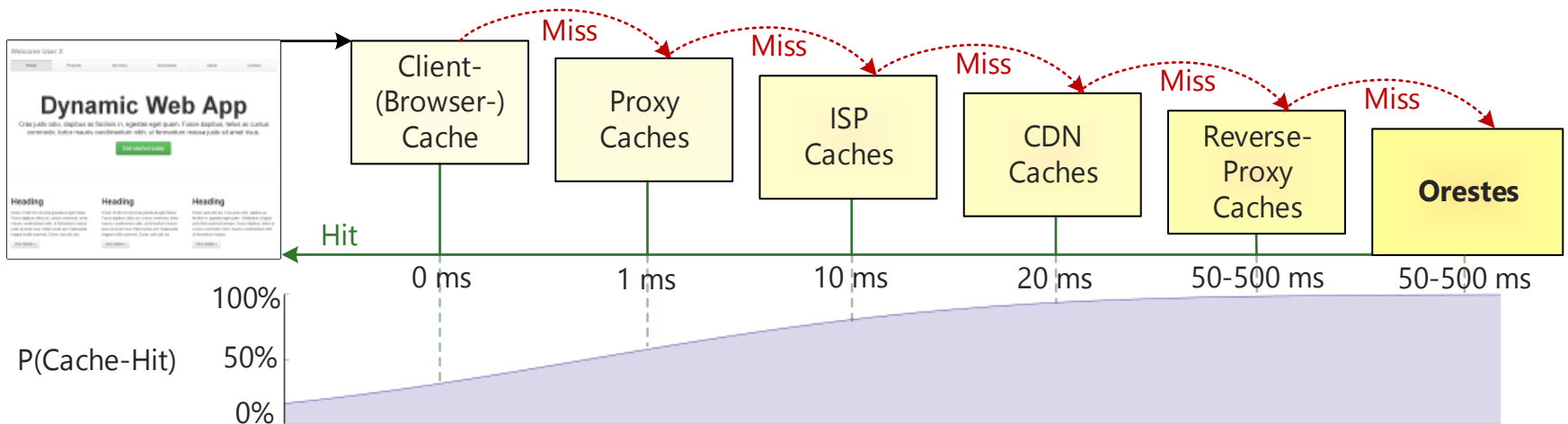


Caching

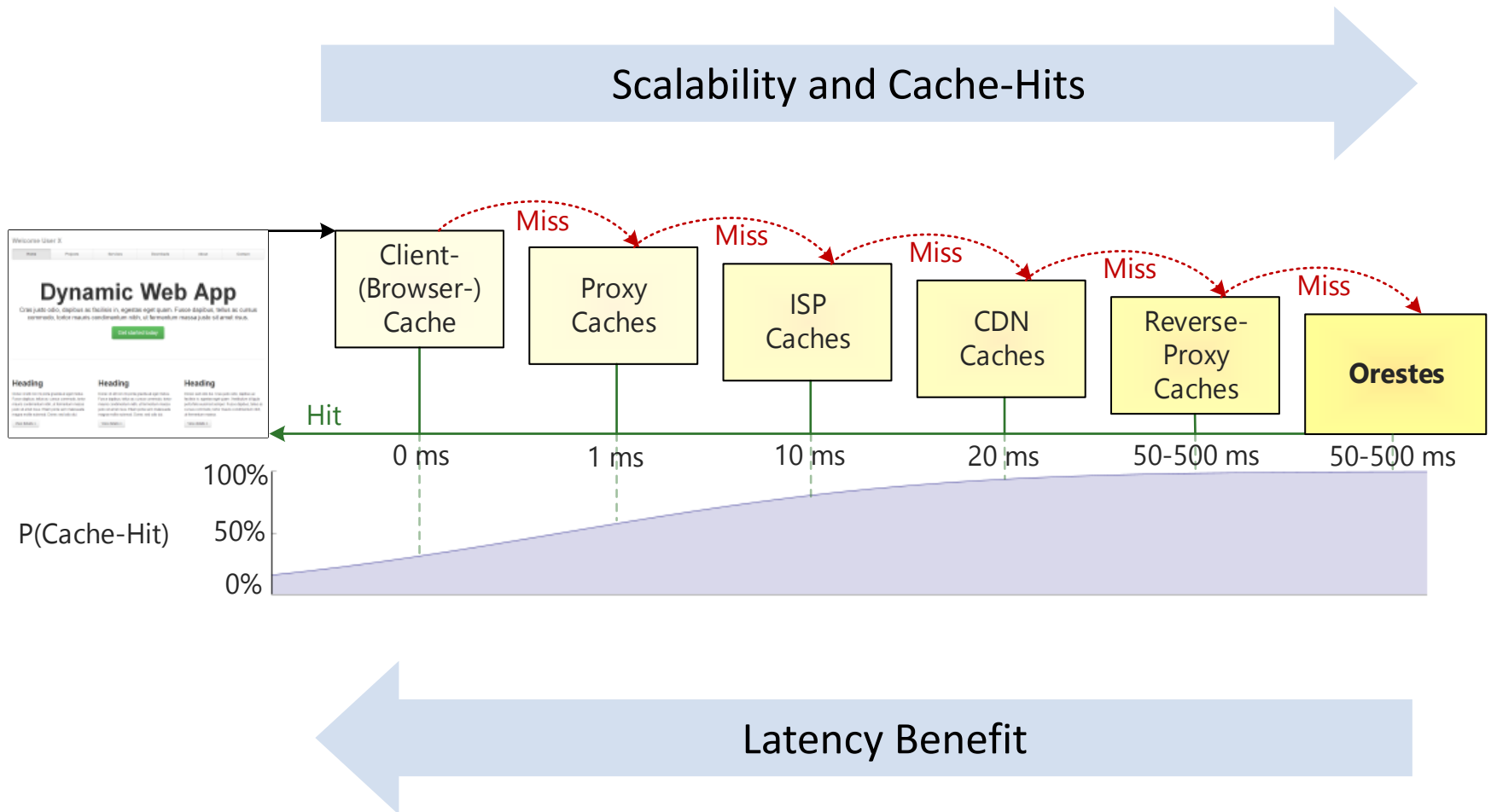


Caching

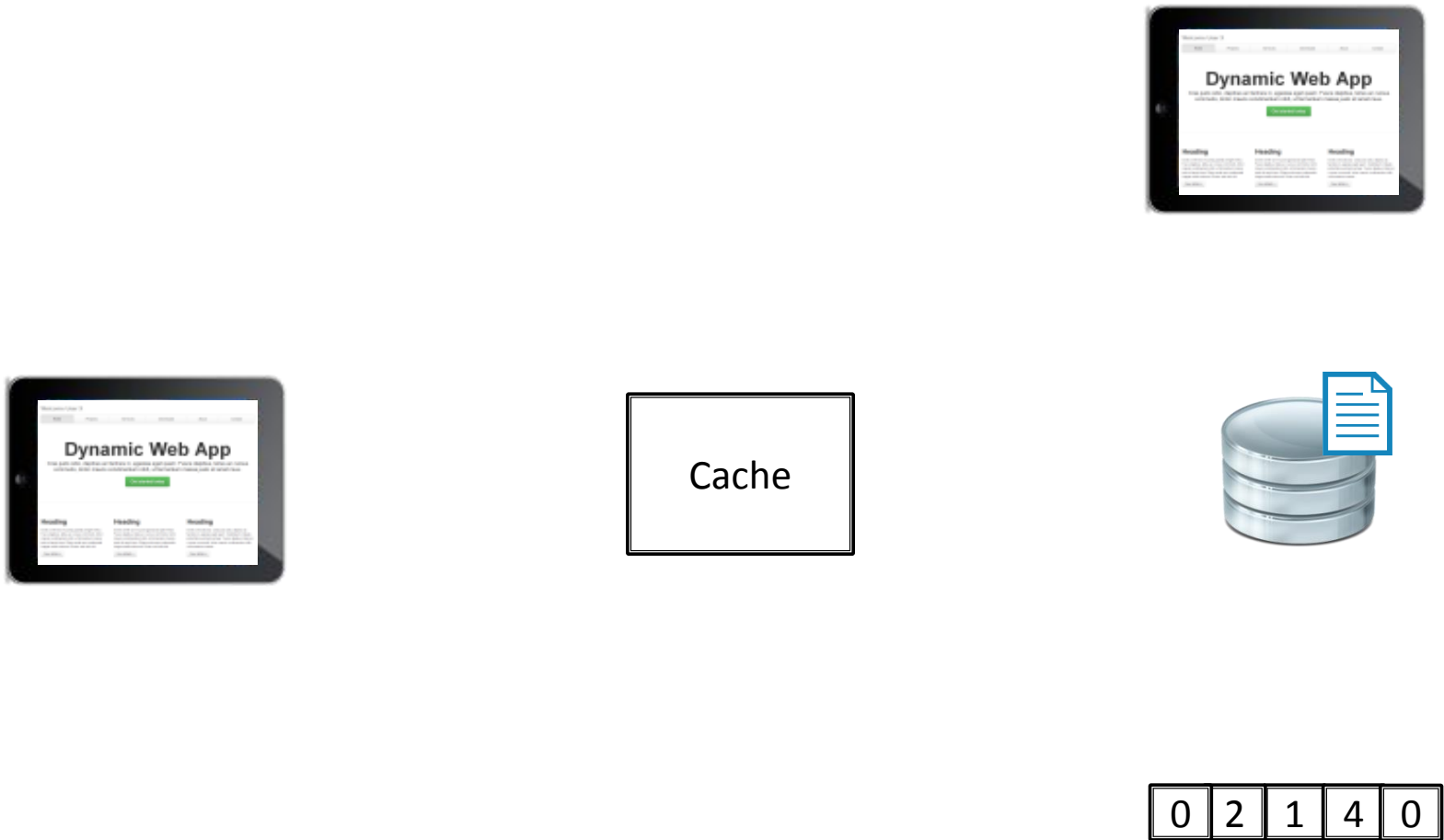
Scalability and Cache-Hits



Caching



BFB: Bloom Filter Bounded Staleness



BFB: Bloom Filter Bounded Staleness

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

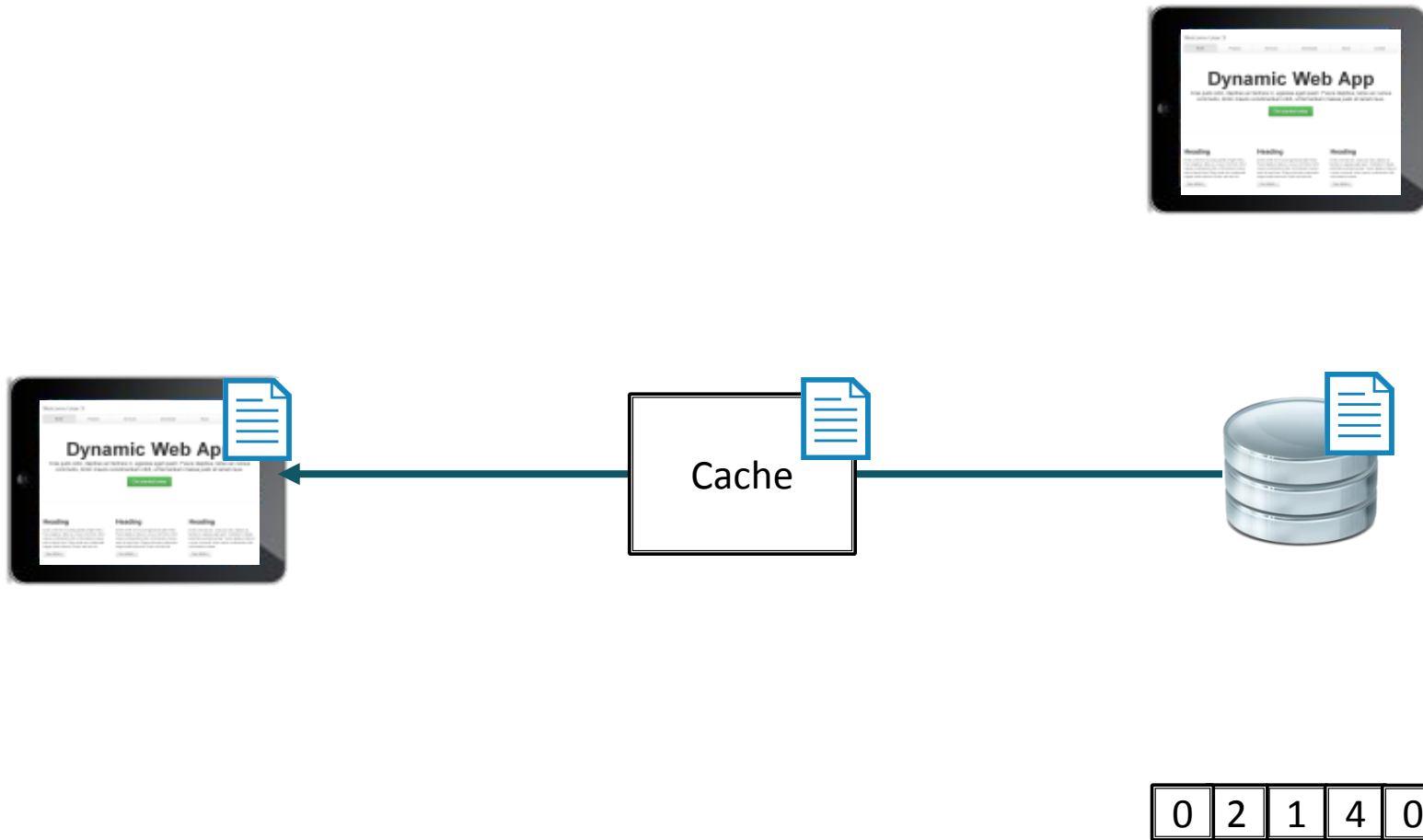


Cache

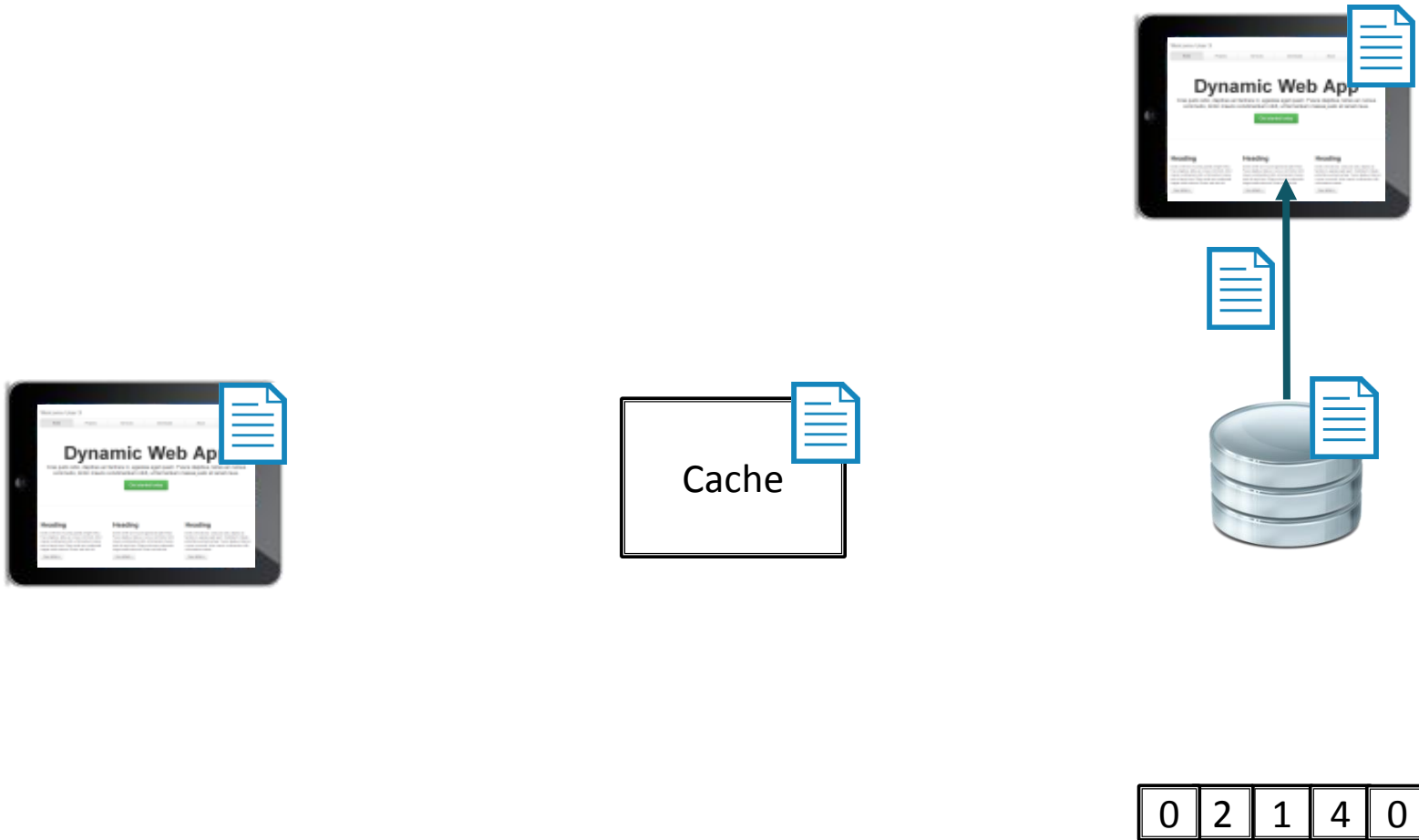


0	2	1	4	0
---	---	---	---	---

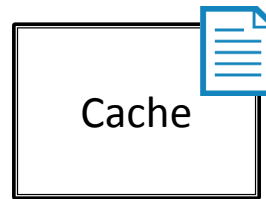
BFB: Bloom Filter Bounded Staleness



BFB: Bloom Filter Bounded Staleness

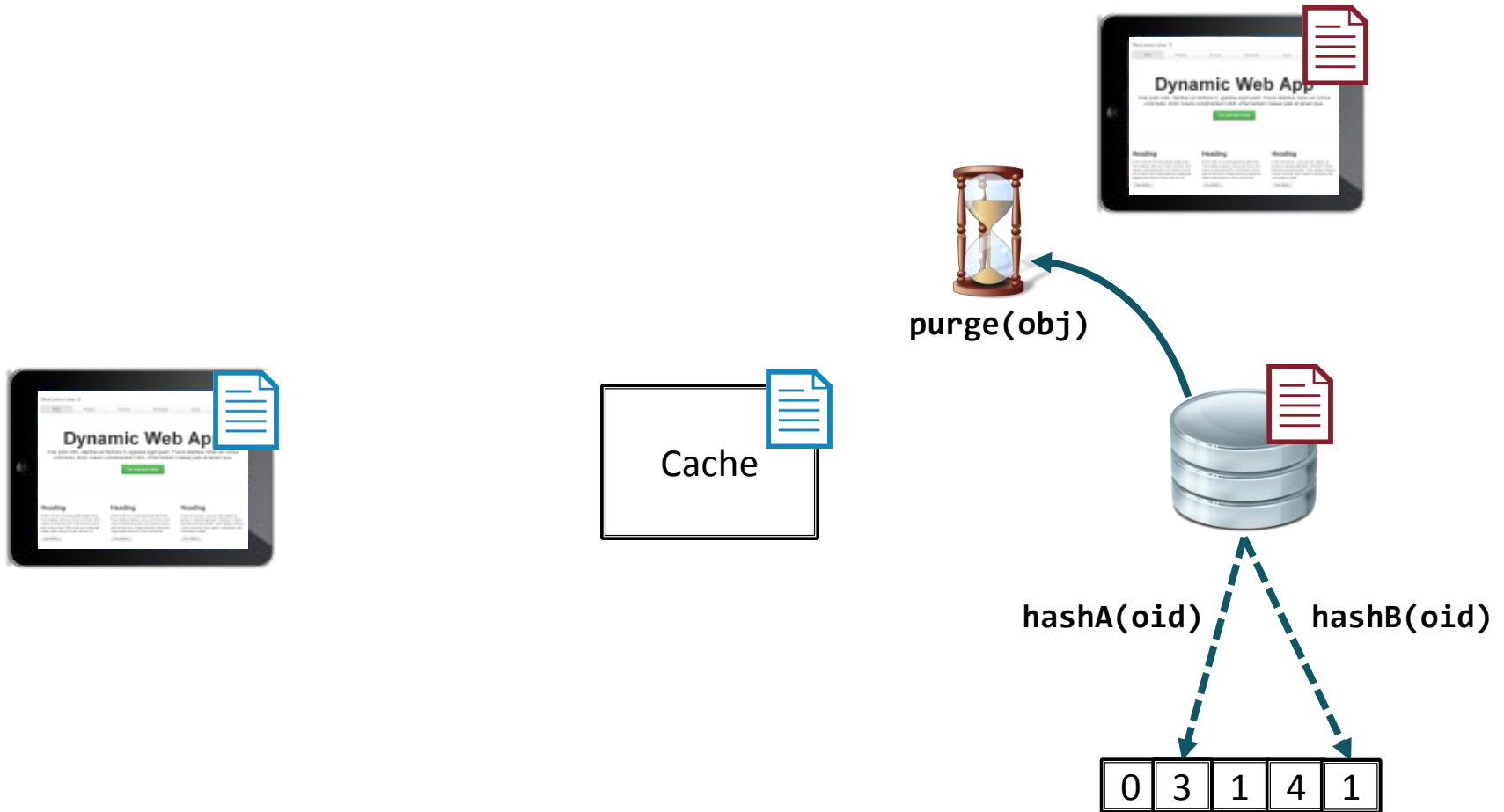


BFB: Bloom Filter Bounded Staleness

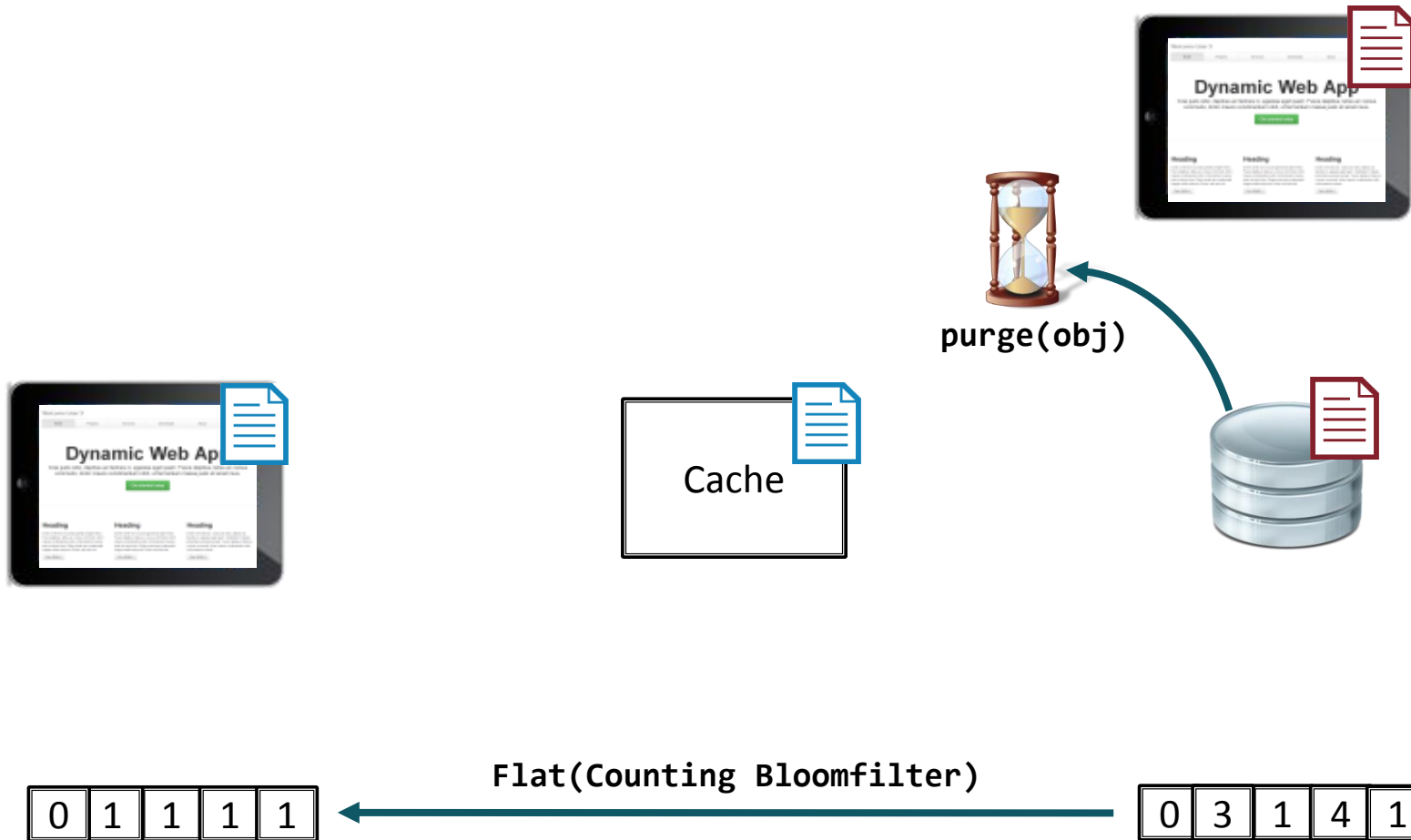


0	2	1	4	0
---	---	---	---	---

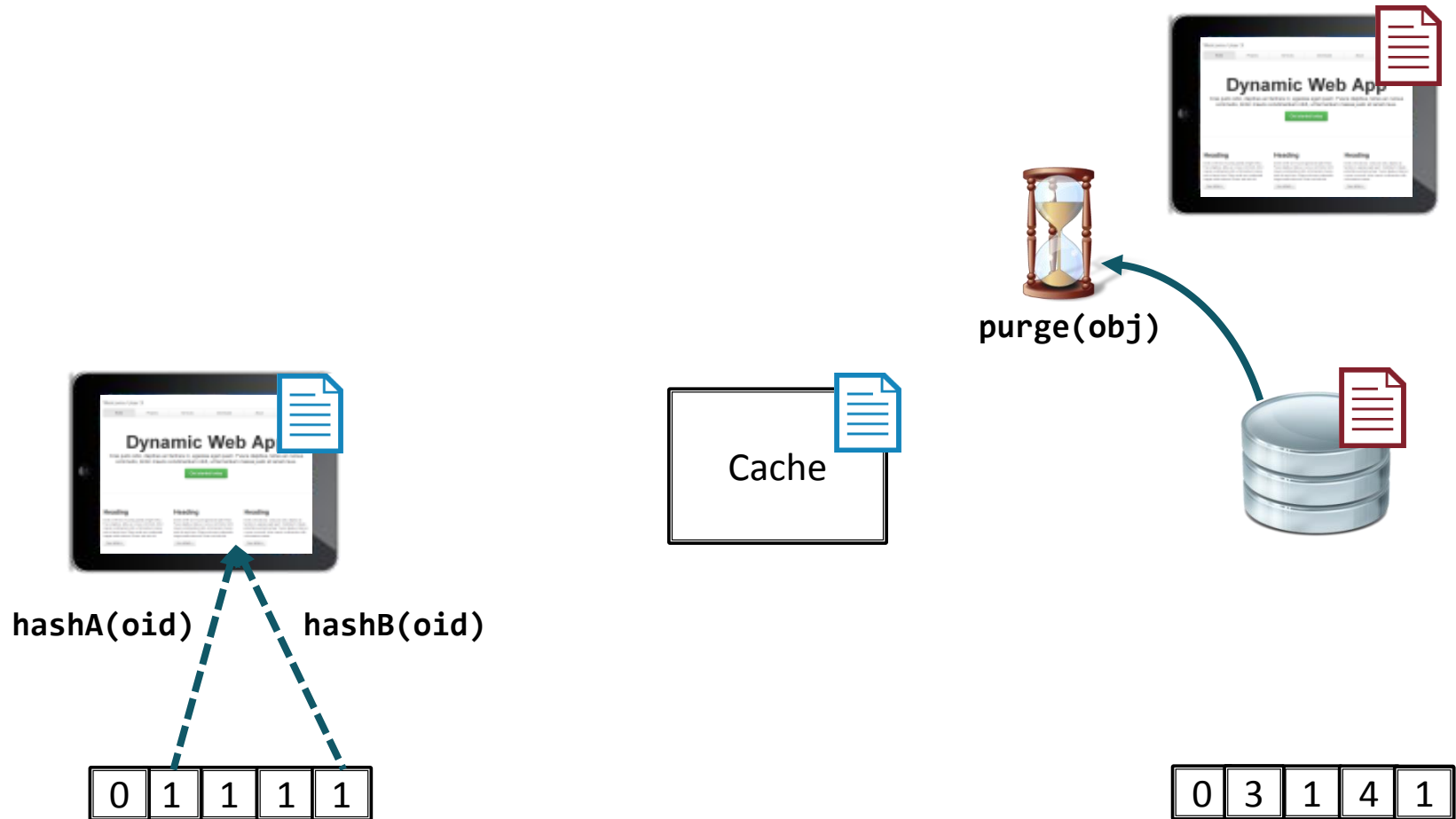
BFB: Bloom Filter Bounded Staleness



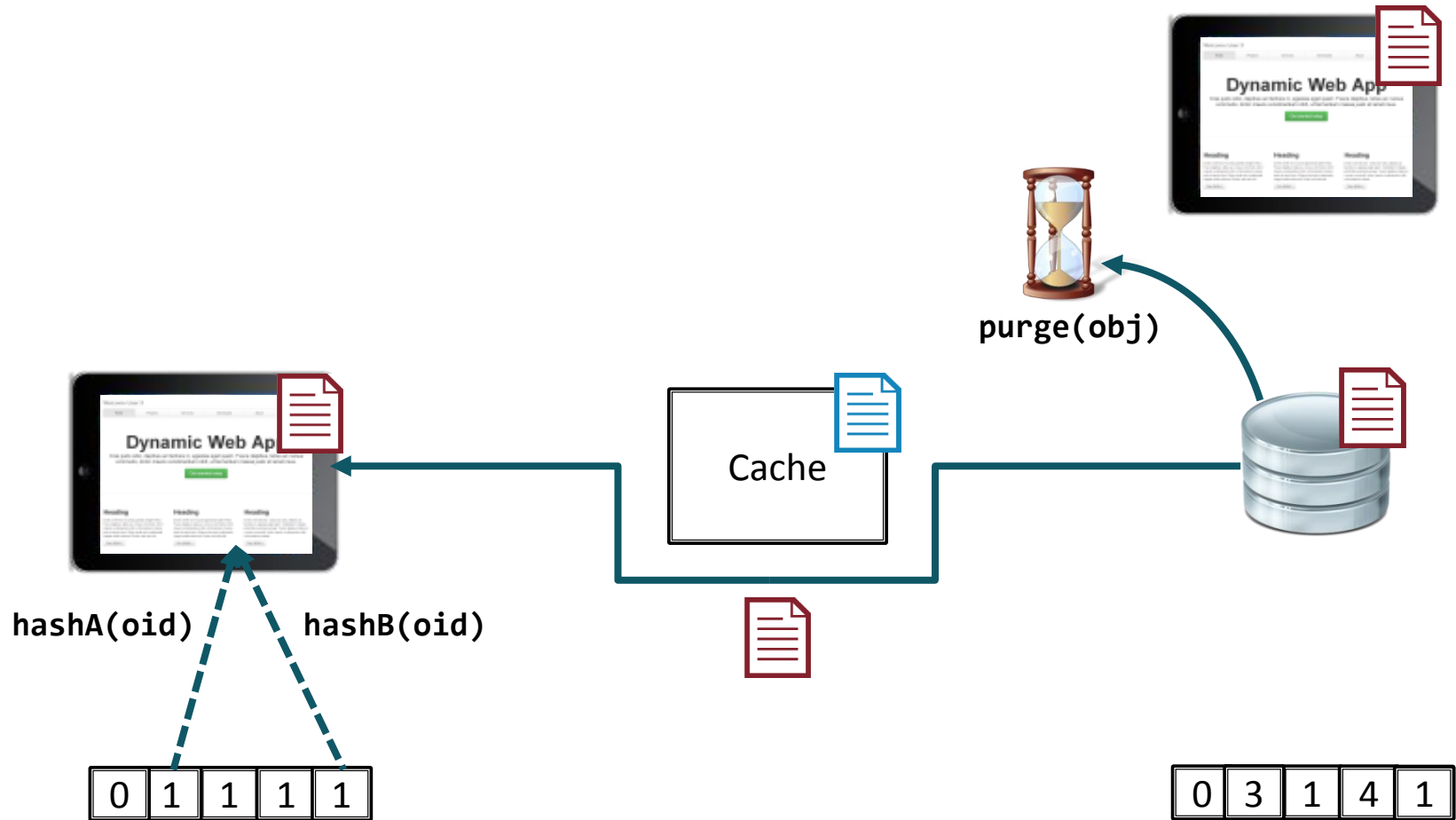
BFB: Bloom Filter Bounded Staleness



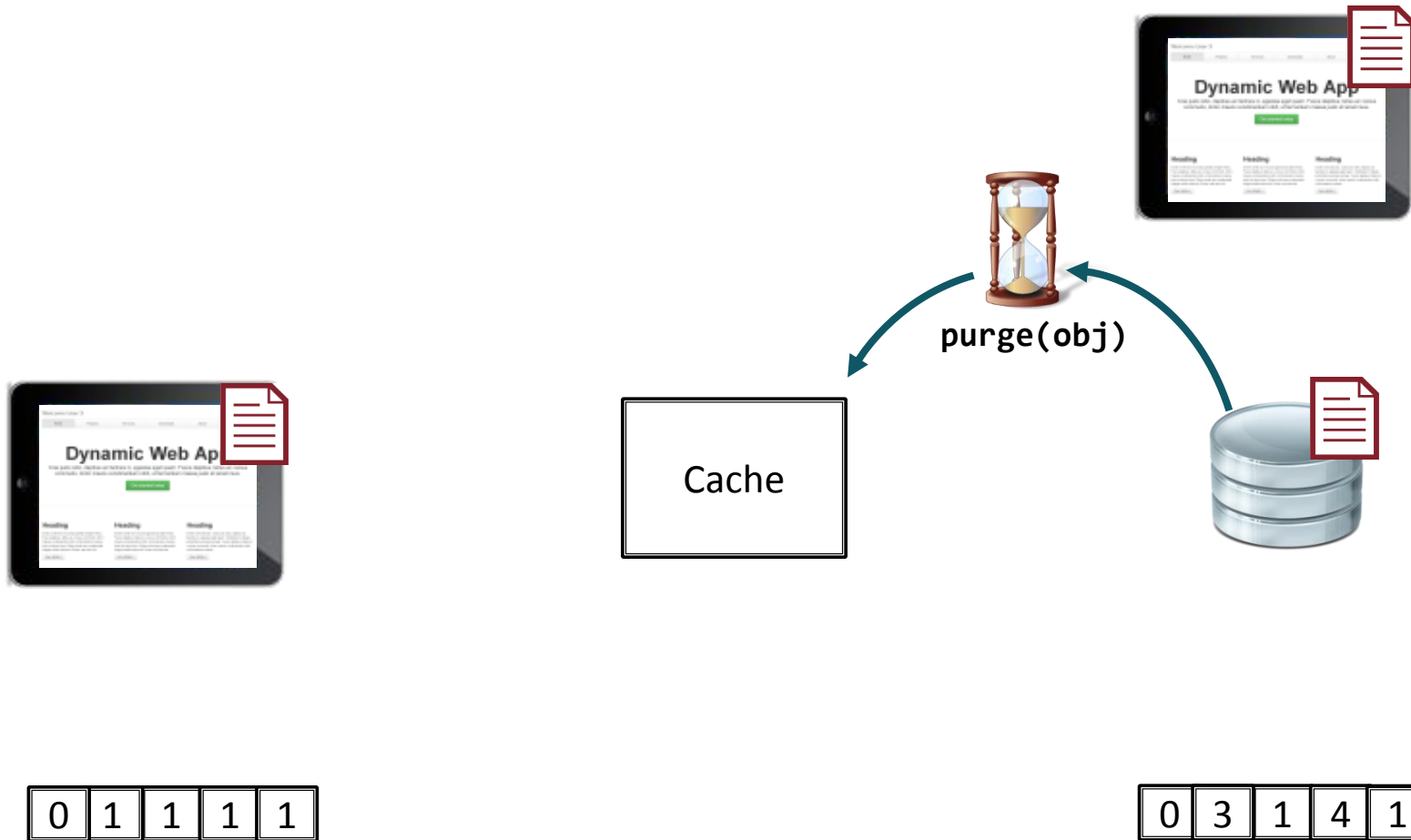
BFB: Bloom Filter Bounded Staleness



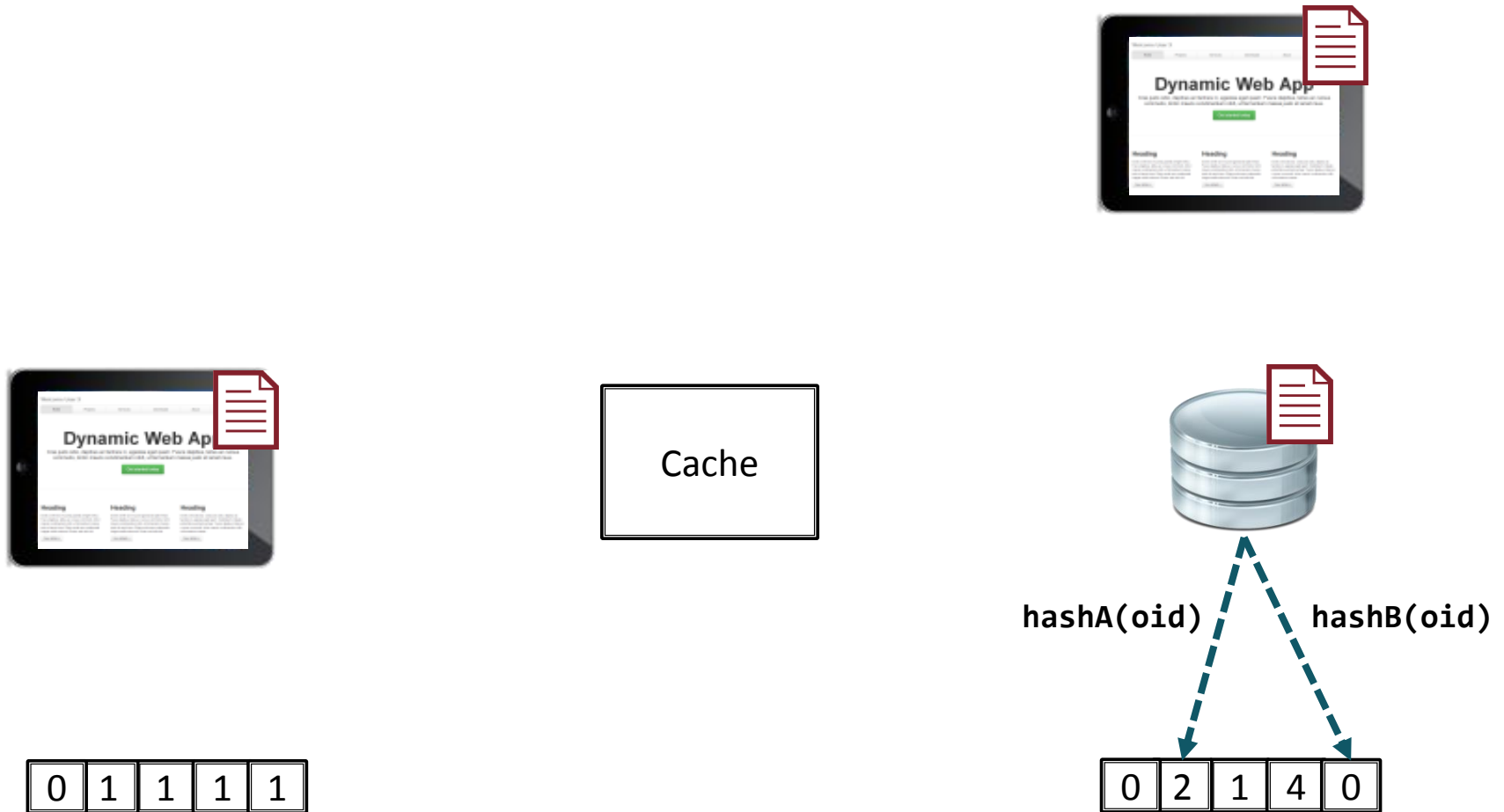
BFB: Bloom Filter Bounded Staleness



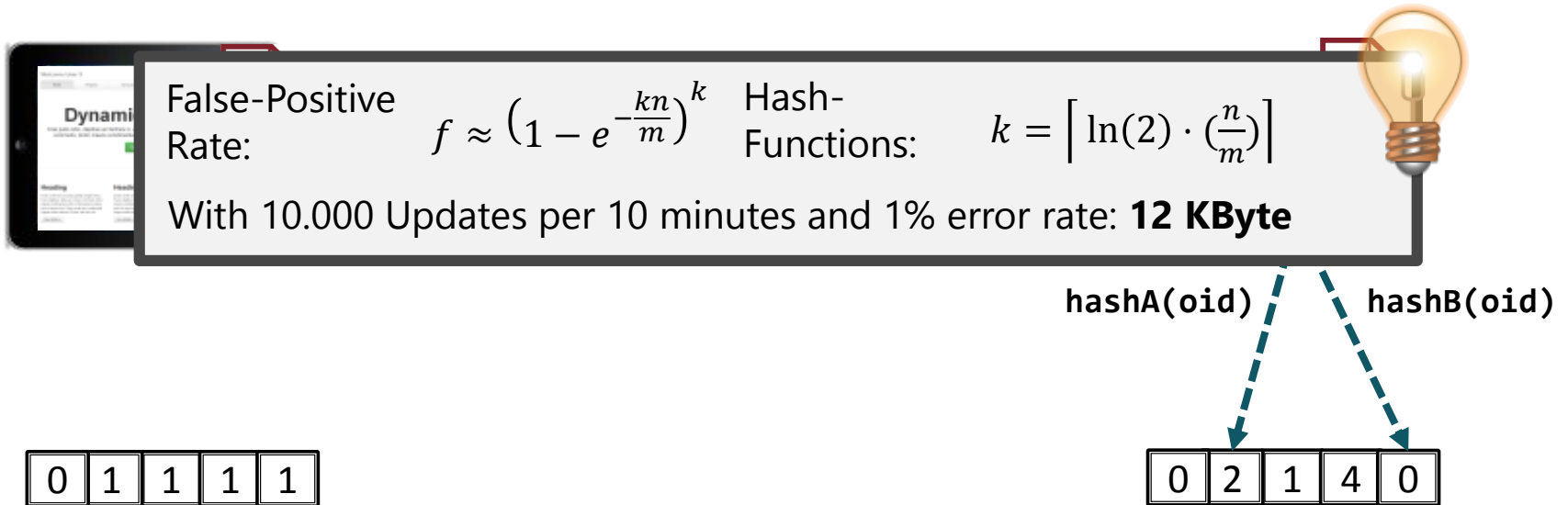
BFB: Bloom Filter Bounded Staleness



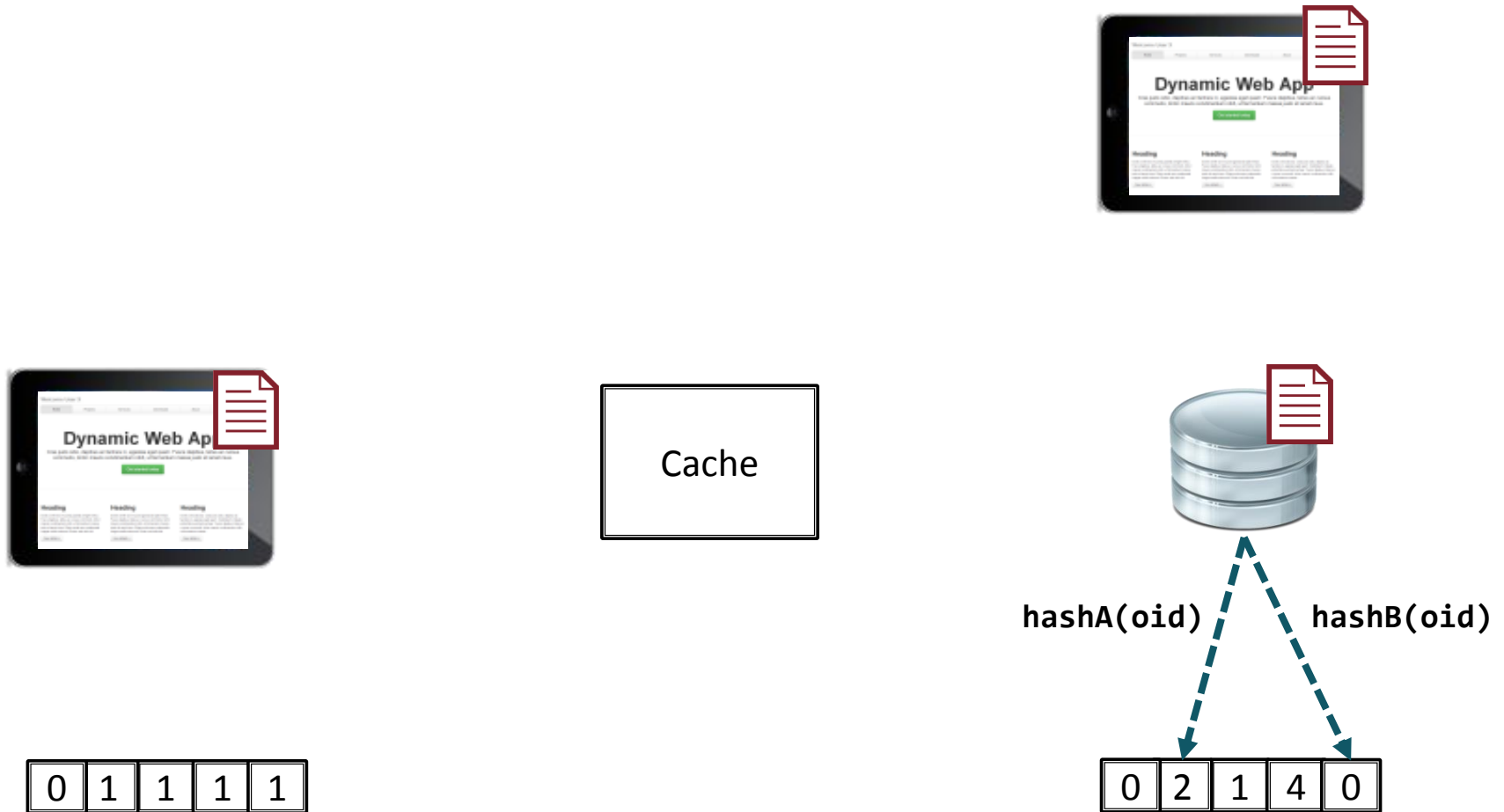
BFB: Bloom Filter Bounded Staleness



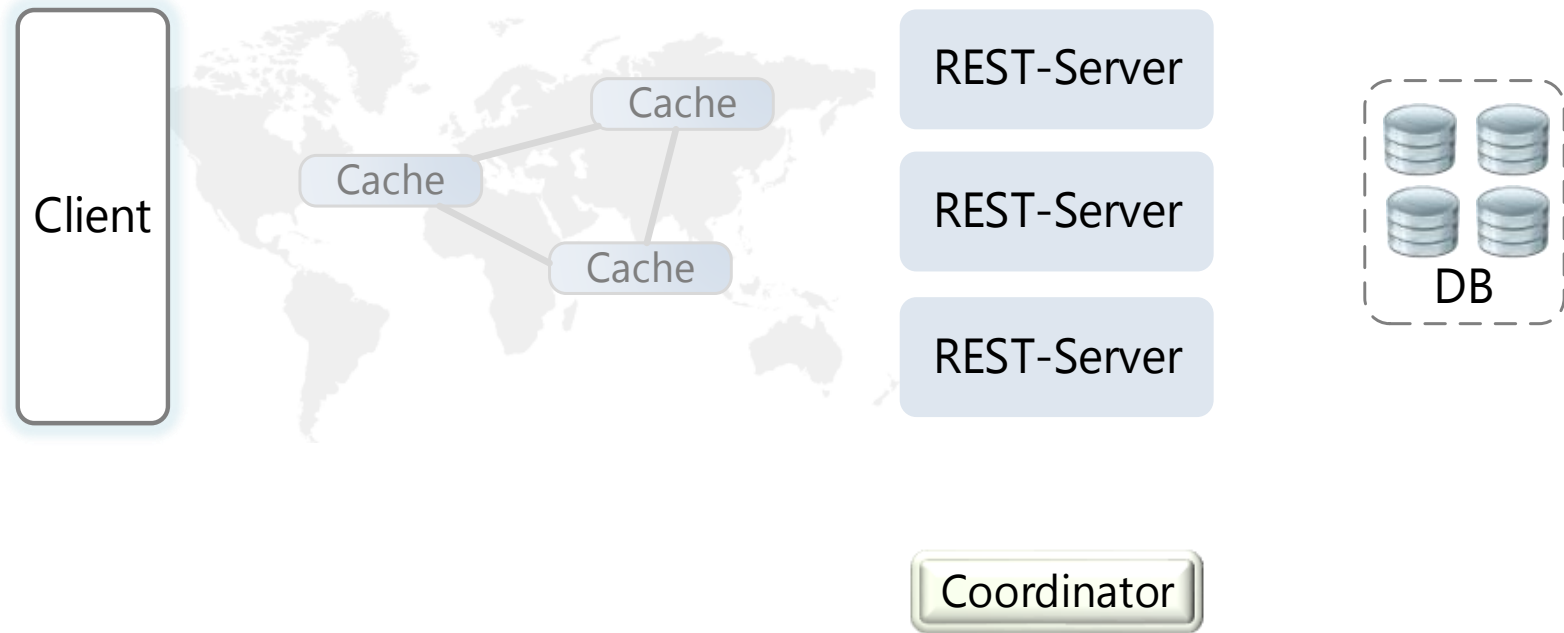
BFB: Bloom Filter Bounded Staleness



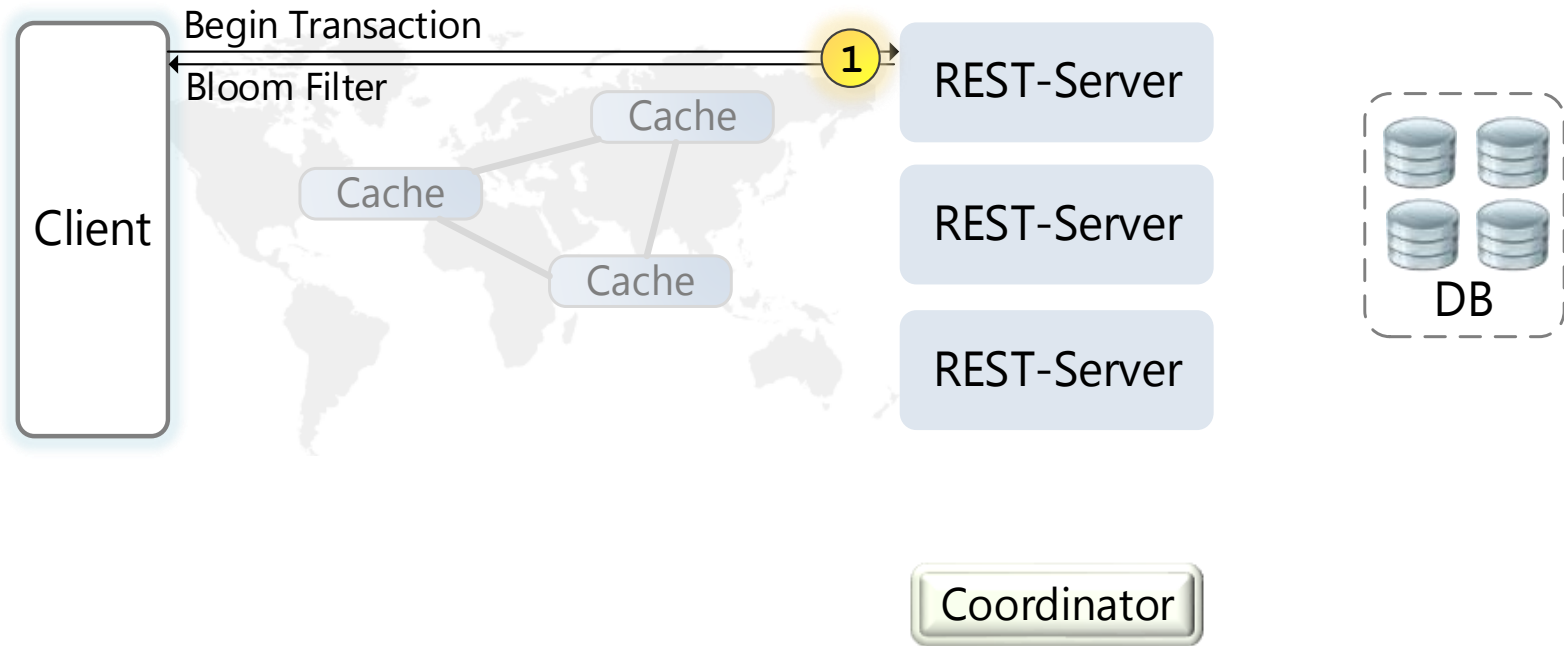
BFB: Bloom Filter Bounded Staleness



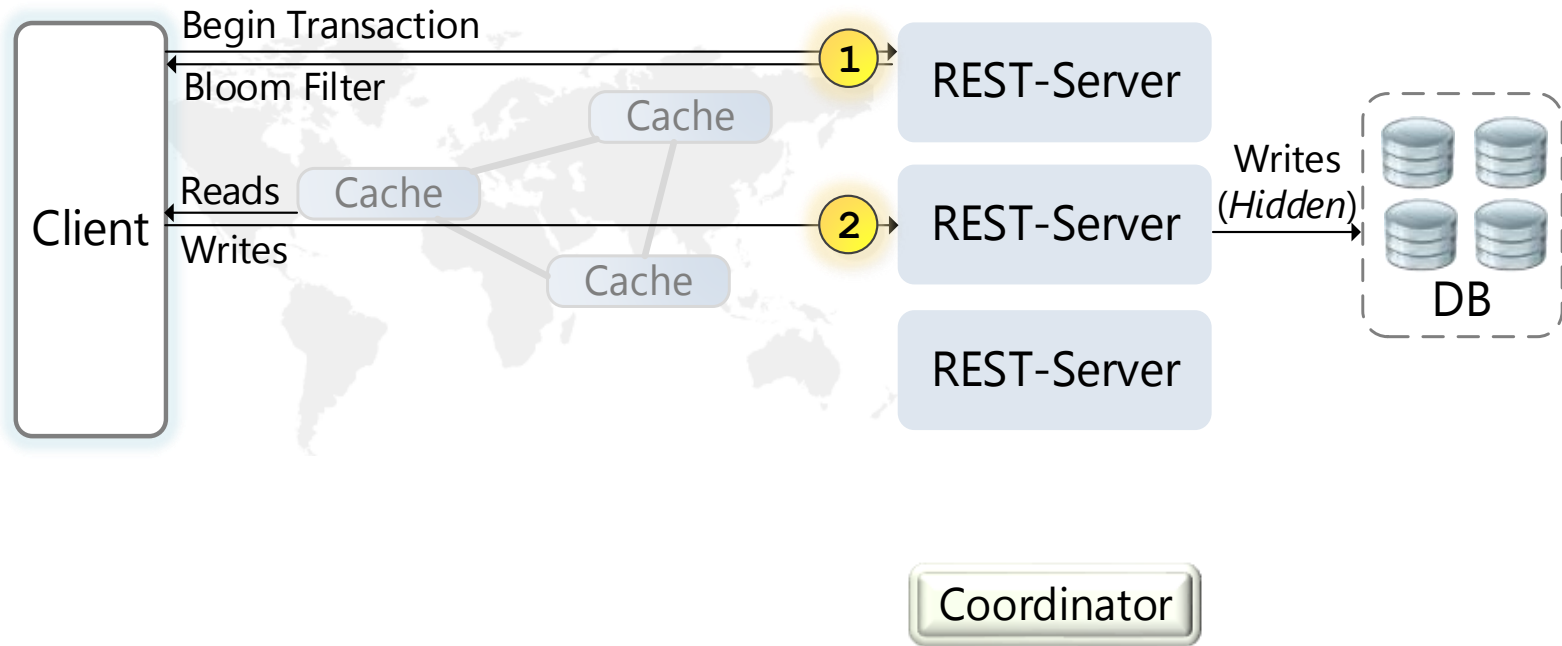
SCOT: Scalable Cache-Aware Optimistic Transactions



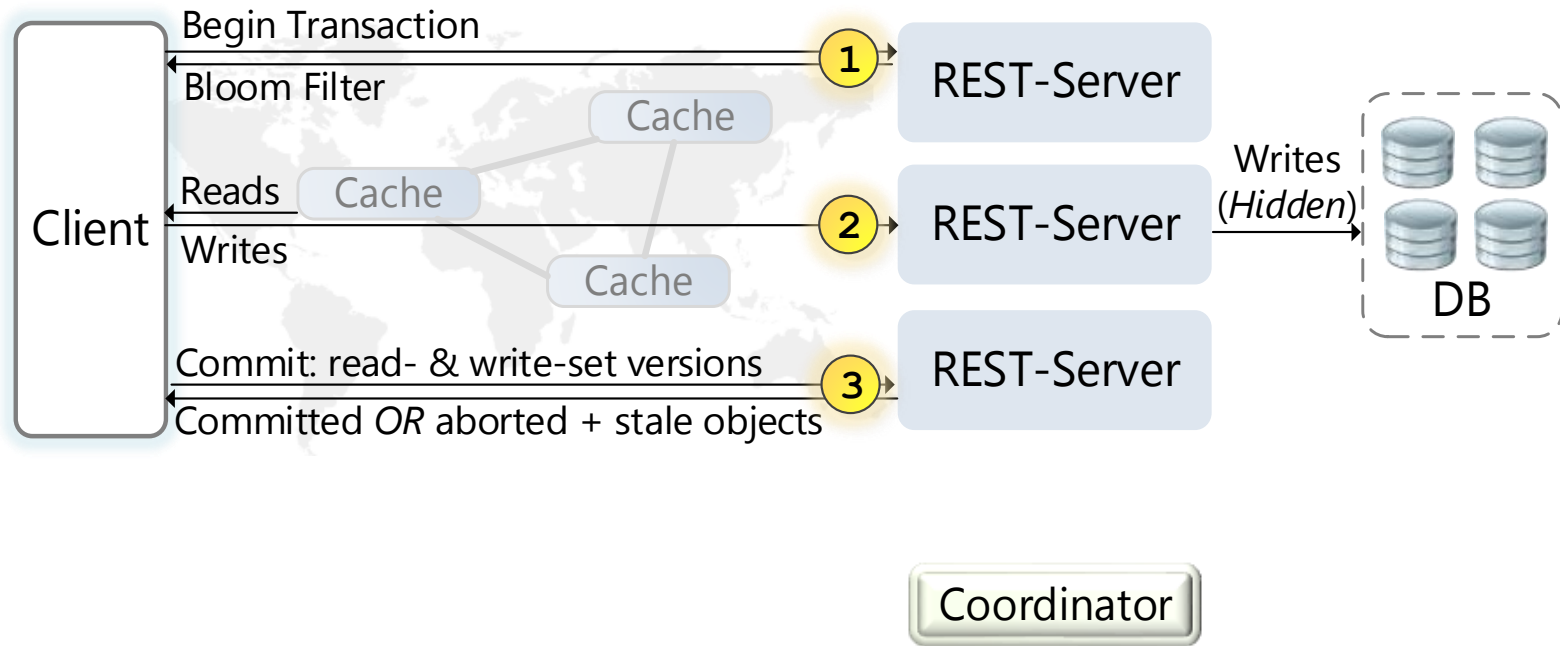
SCOT: Scalable Cache-Aware Optimistic Transactions



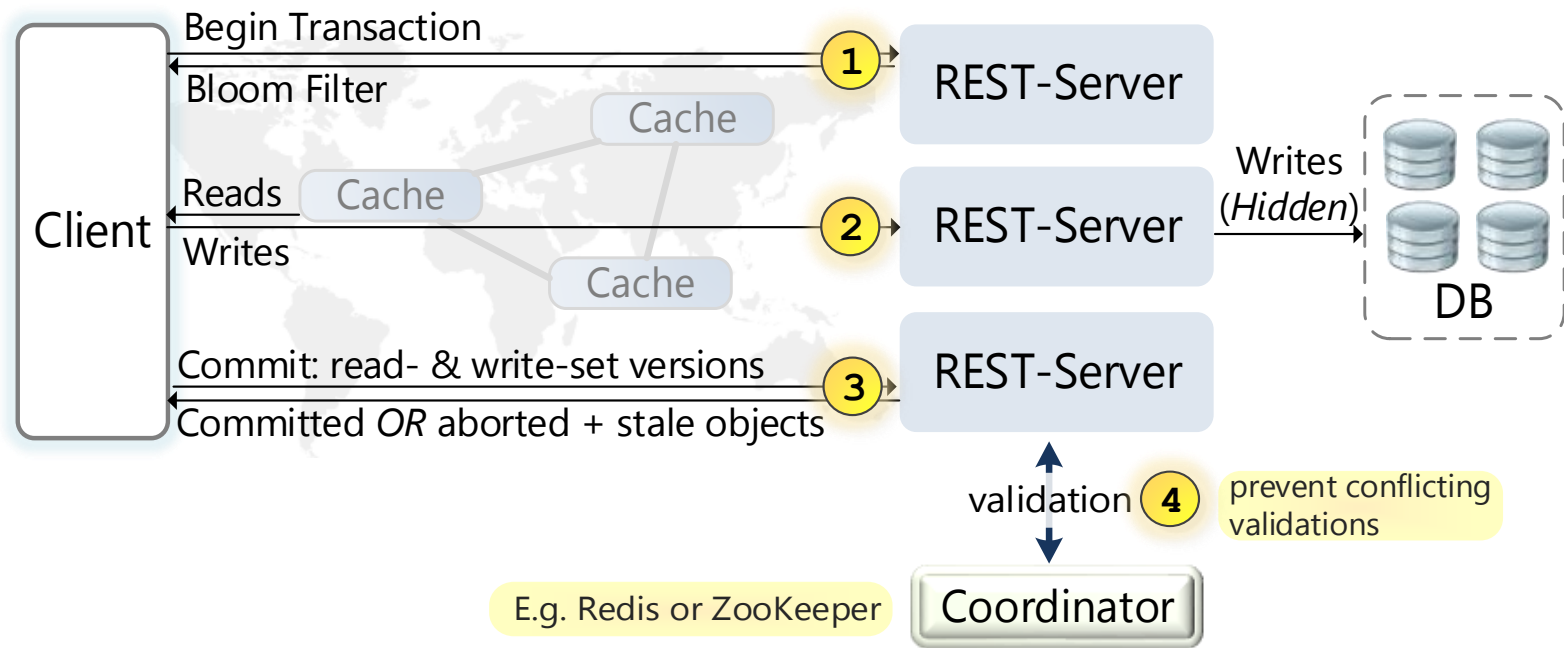
SCOT: Scalable Cache-Aware Optimistic Transactions



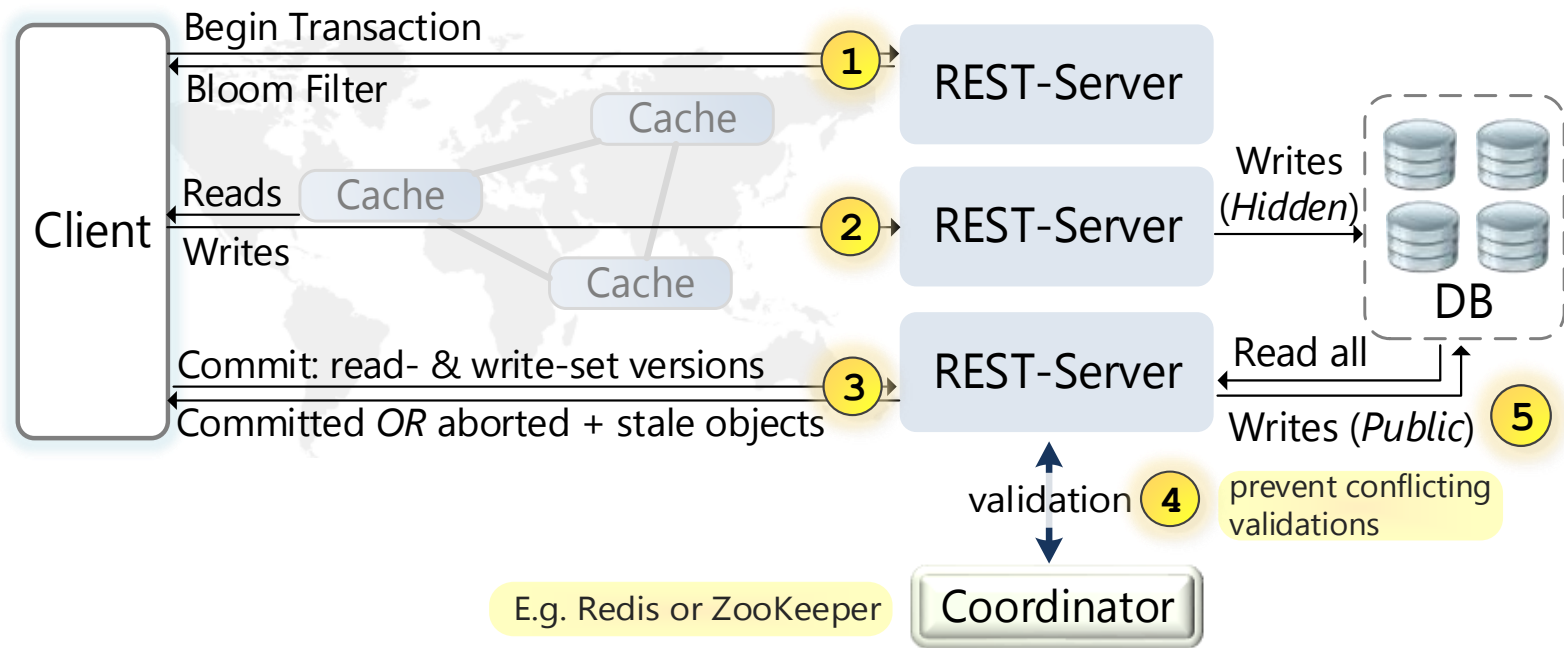
SCOT: Scalable Cache-Aware Optimistic Transactions



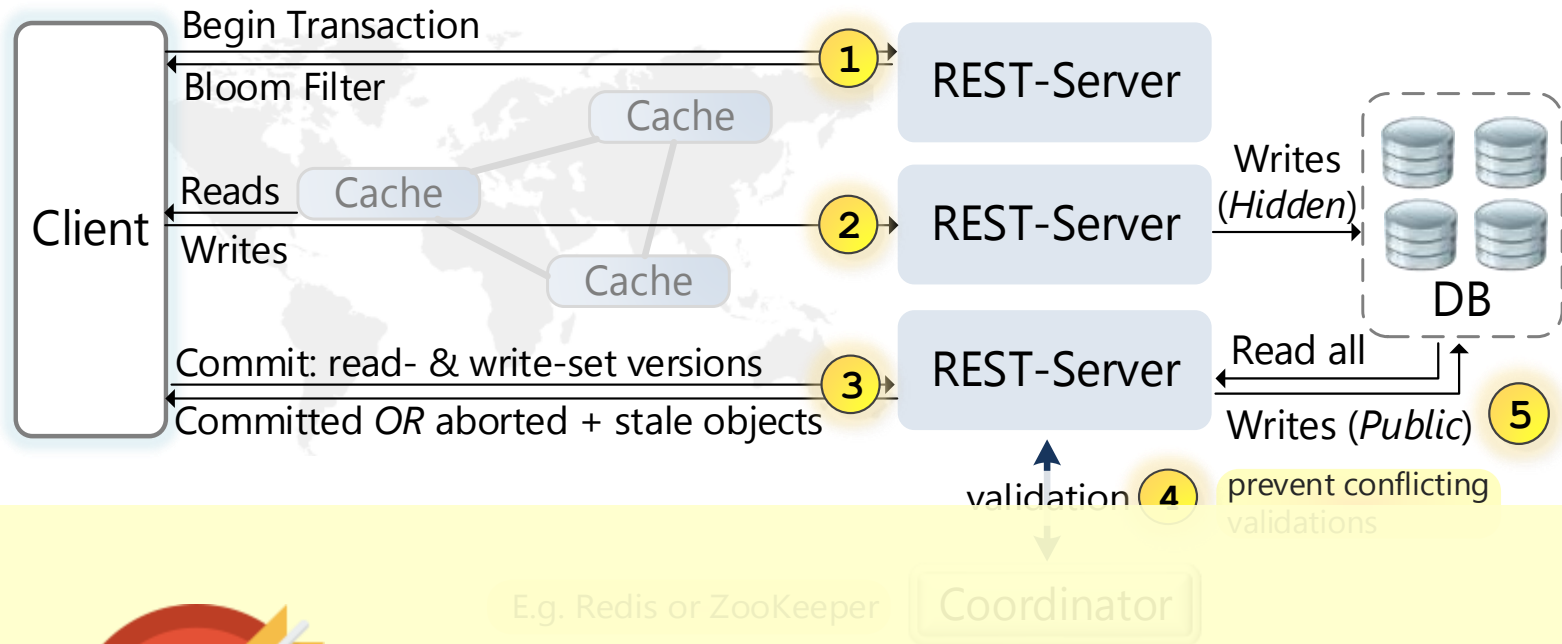
SCOT: Scalable Cache-Aware Optimistic Transactions



SCOT: Scalable Cache-Aware Optimistic Transactions

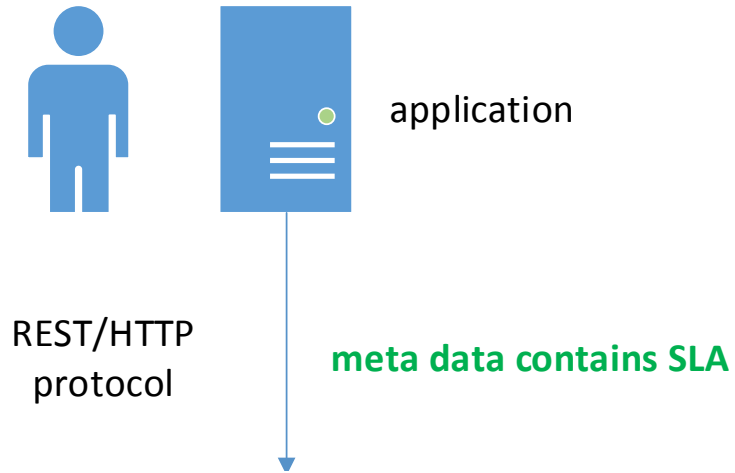


SCOT: Scalable Cache-Aware Optimistic Transactions

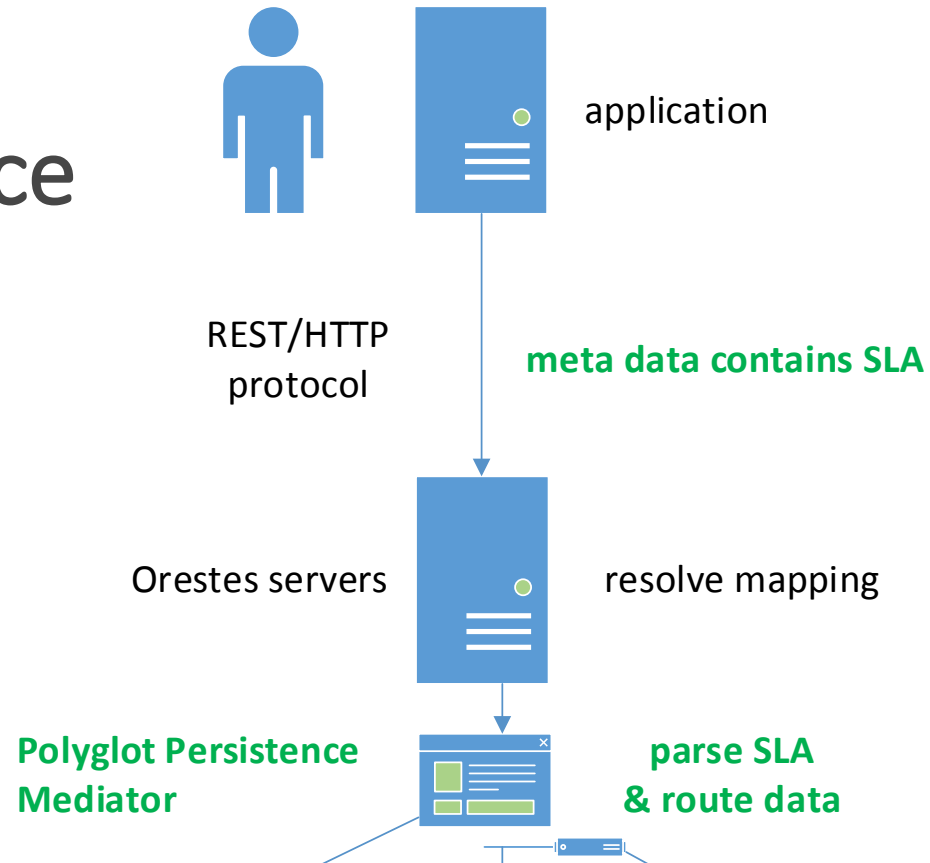


Caching → Shorter transaction duration → **less aborts**

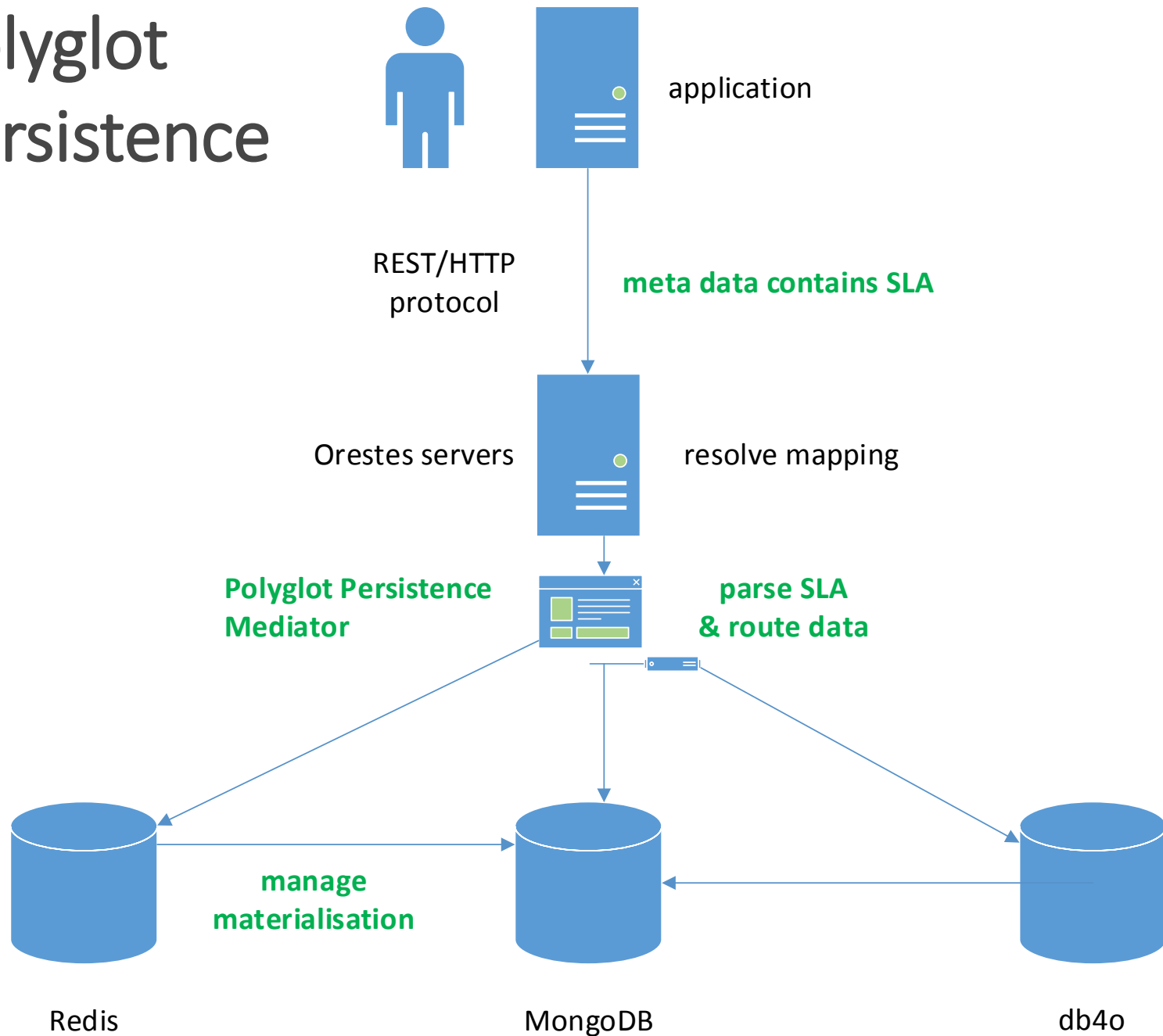
Polyglot Persistence



Polyglot Persistence



Polyglot Persistence



Polyglot Persistence



application

Results:

Article-Objects with Impression Count



Article

ID
Title
...

MongoDB

Imp.

Imp.
ID

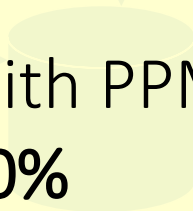
Redis Sorted Set

Speedup with PPM:

- 50-1000%
- 66% performance of Varnish



Redis

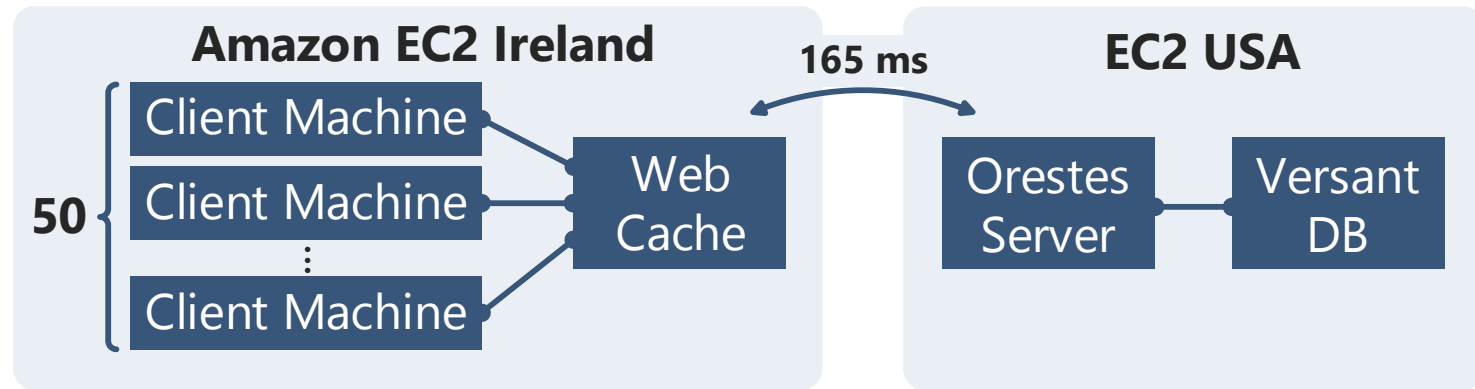


MongoDB

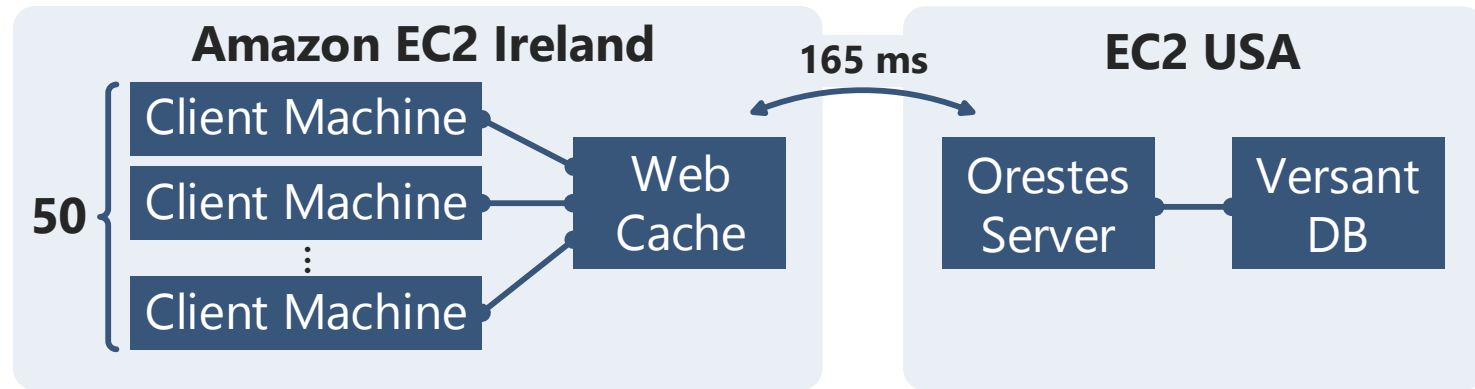


db4o

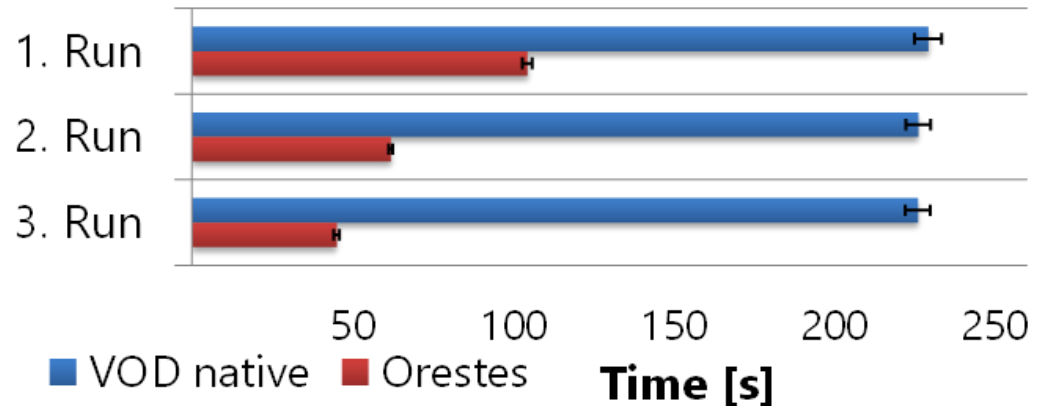
Cloud Evaluation of ORESTES



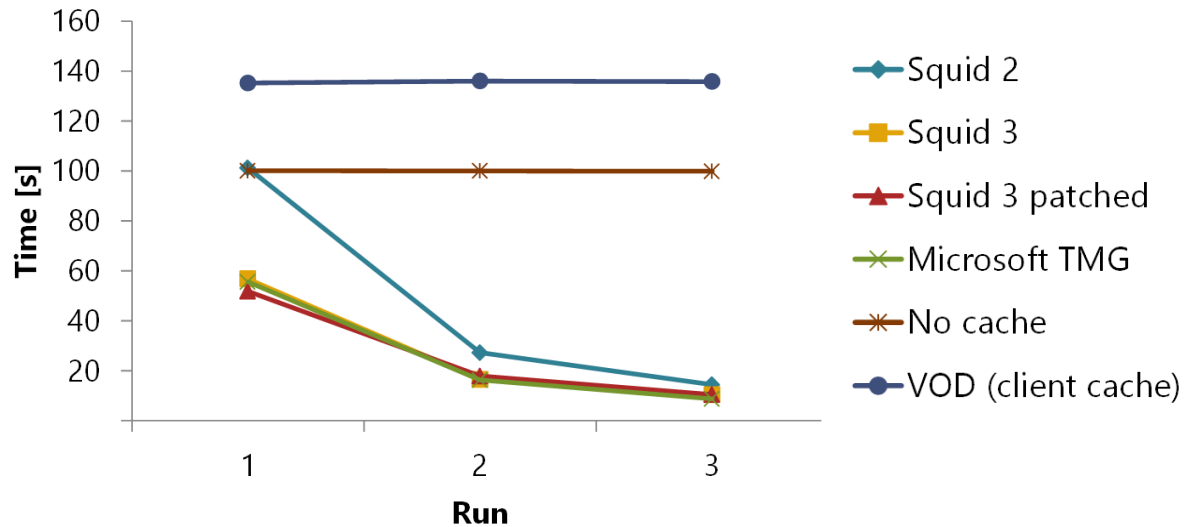
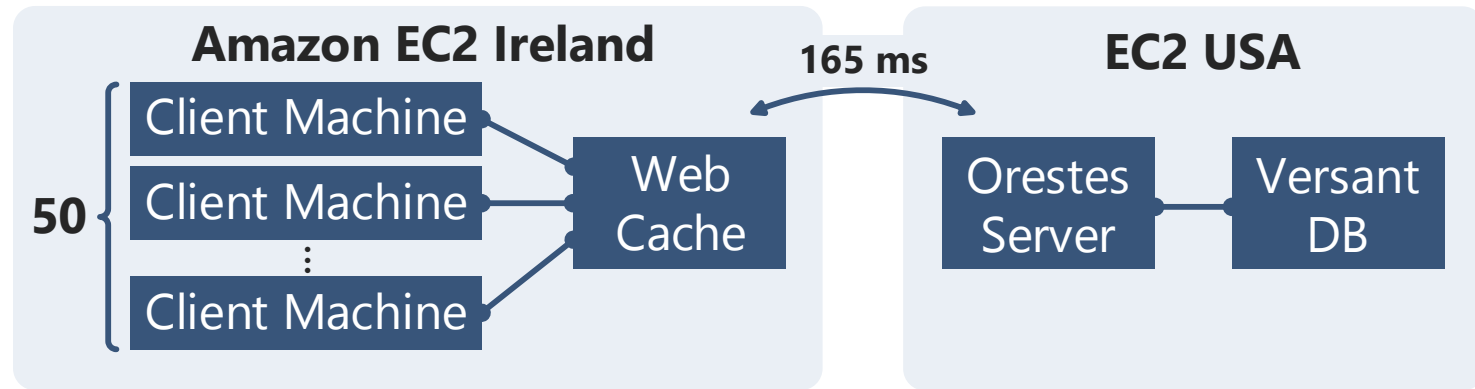
Cloud Evaluation of ORESTES



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-independent 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

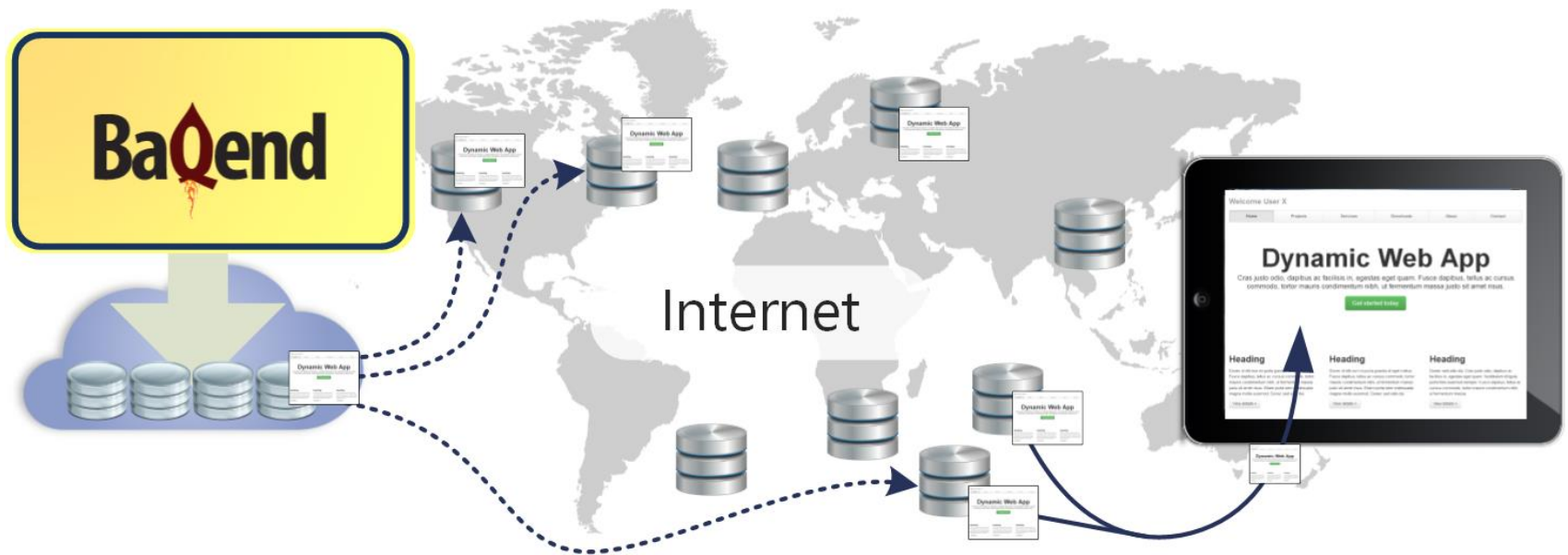


A close-up, horizontal view of a stack of several old, thick books. The spines and edges of the books are visible, showing significant wear, discoloration, and some damage. The colors range from dark brown to light tan. A semi-transparent, light-colored rectangular box is overlaid on the middle of the image, containing the text "From Research to Practice".

From Research to Practice

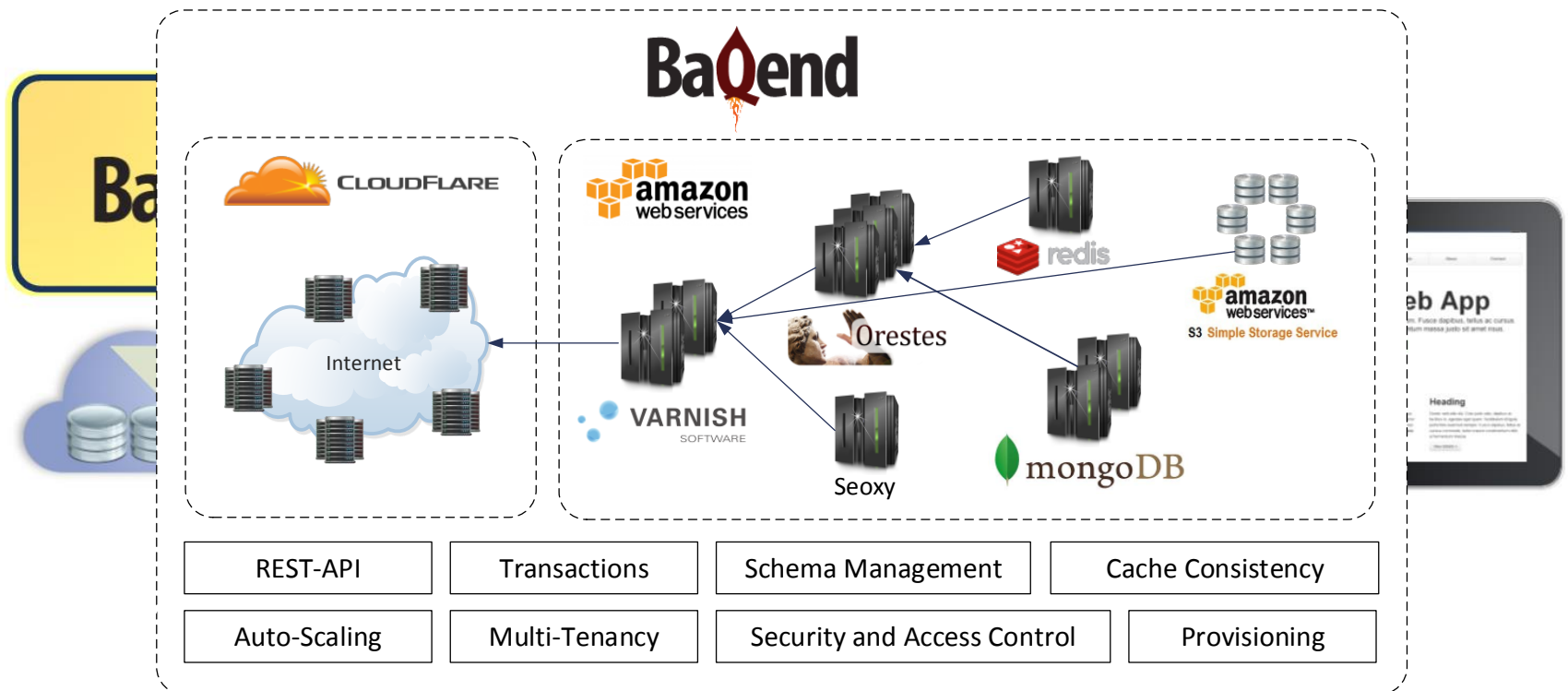
Baqend

- ▶ Orestes as a startup



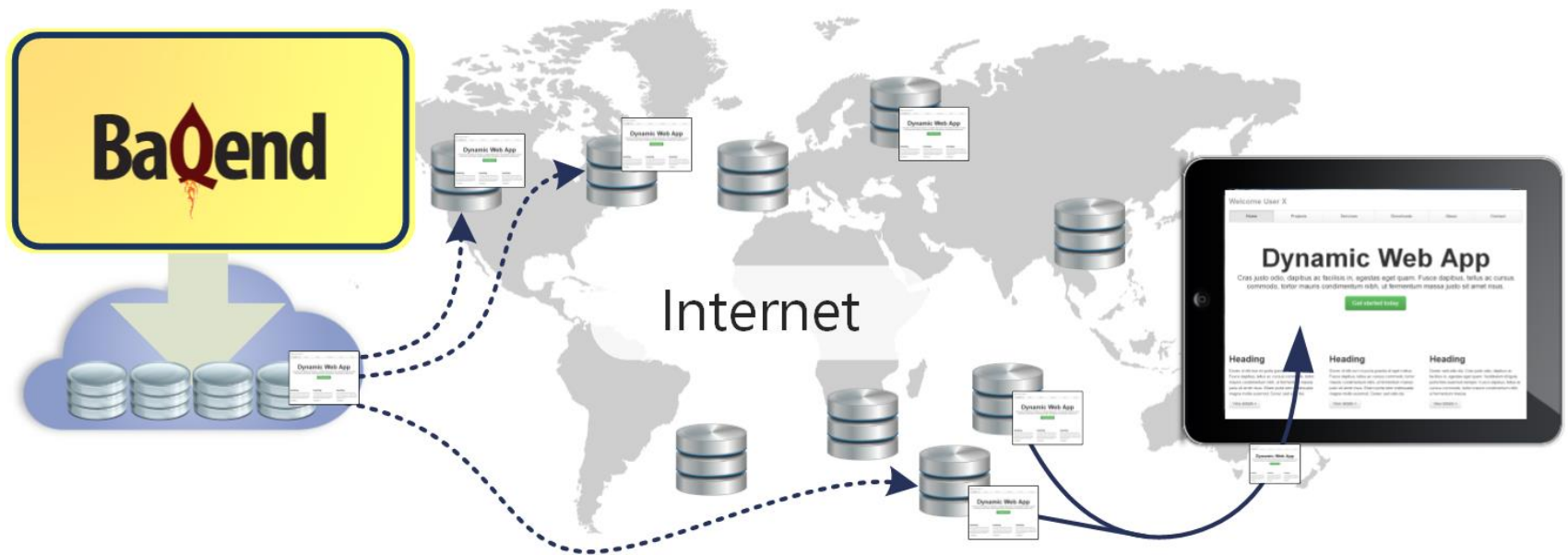
Baqend

- ▶ Orestes as a startup



Baqend

- ▶ Orestes as a startup



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

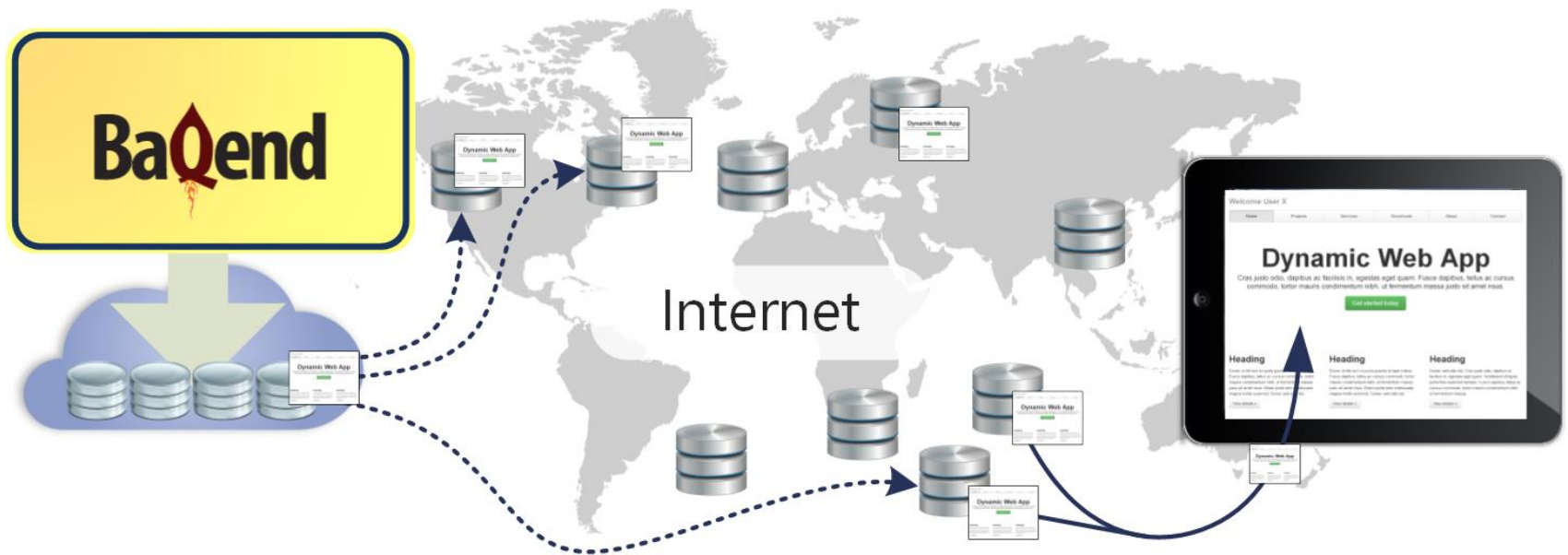
Response Class

Model | Model Schema

object

Baqend

- ▶ Orestes as a startup



Baqend

▶ 0

Baqend API Explorer Console Data Explorer Documentation 🔍

▶ Execute ✕ Clear Output Select Script ▼

```
1 var schema = [ {
2   "class" : "/db/Coffee",
3   "fields" : {
4     "name" : "/db/_native.String",
5     "countryCode" : "/db/_native.Integer",
6     "caffeine" : "/db/_native.Boolean",
7     "parent" : "/db/Coffee"
8   }
9 } ];
10
11 var schema = [ {
12   "class" : "/db/BaseEvent",
13   "fields" : {
14     "id" : "/db/_native.String",
15     "title" : "/db/Lang",
16     "subtitle" : "/db/Lang",
17     "date" : "/db/_native.DateTime",
18     "beginDate" : "/db/_native.Date",
19     "endDate" : "/db/_native.Date",
20     "dateDescription" : "/db/Lang",
21     "description" : "/db/Lang",
22     "media" : "/db/_native.String"
23   }
24 }, {
```

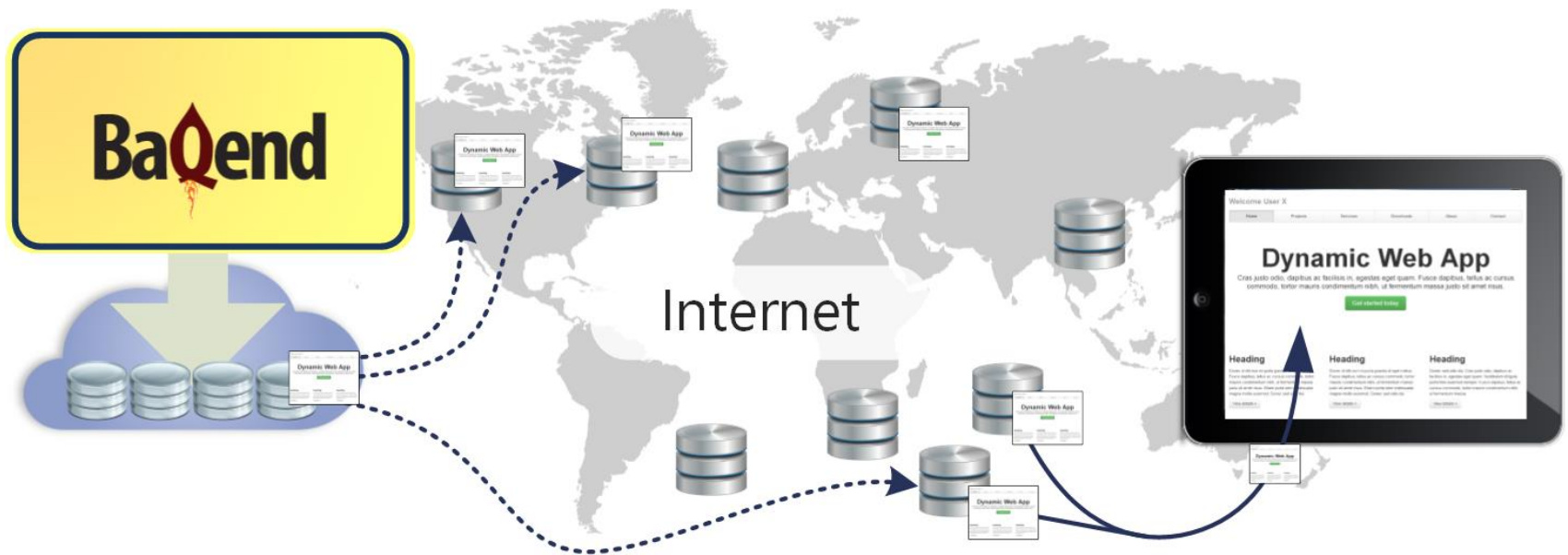
> Click execute or press F9 to run your script.
> Script "Initial" loaded.

© Baqend · [Contact](#) · [Terms of Usage](#)

[Blog](#) · [Twitter](#)

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(...);
```

My Web App

Home	Projects	Services	Downloads	About	Contact
------	----------	----------	-----------	-------	---------

Dynamic Web App

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet.

Get started today

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa.

[View details »](#)

Baqend in Action



My Web App



Dynamic Web App

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet.

Get started today

Heading



Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading



Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading



Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa.

[View details »](#)

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

Baqend

Build faster Apps faster.

Announcing the world's fastest backend

Baqend is a Backend-as-a-Service for web and mobile apps. It has a unique and previously unreachable mission: to make your apps load and respond without perceptible delay - i.e. in 100 milliseconds or less.

[Keep me posted](#)

[Contact the team](#)

126
days

2
hours

8
mins

23
secs

More: [Baqend.com](https://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

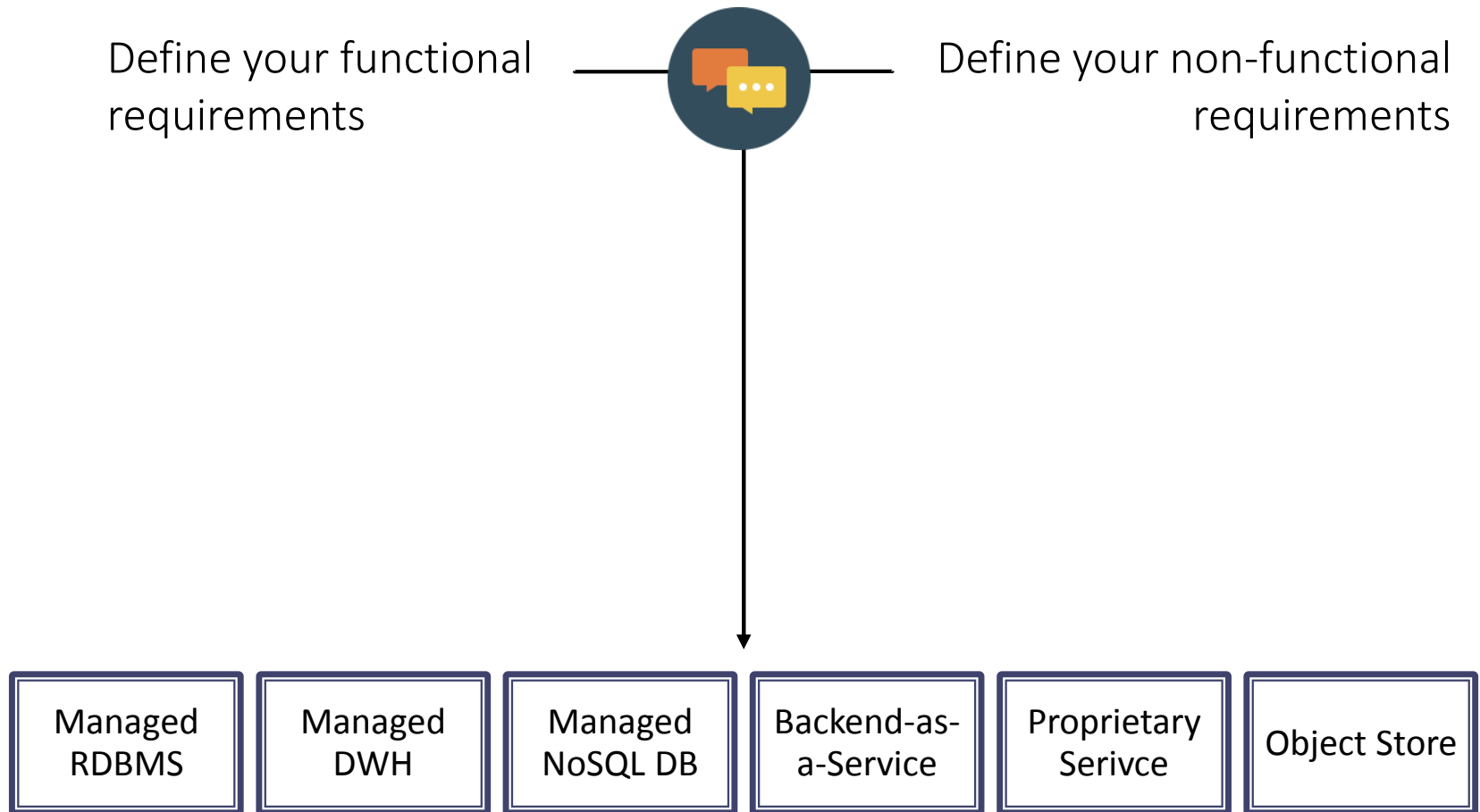
Proprietary
Service

Object Store



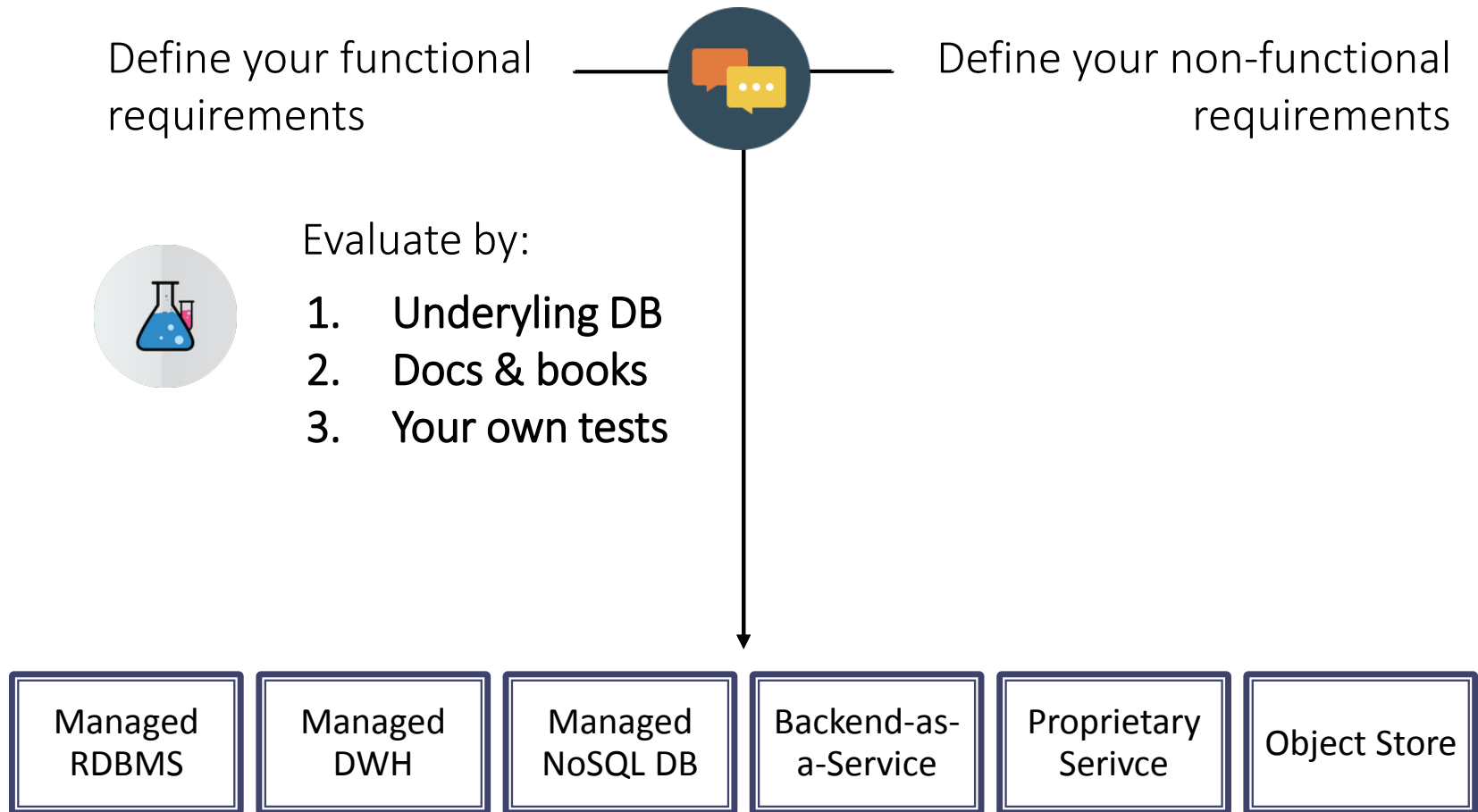
Wrap-up

How to choose a cloud database:



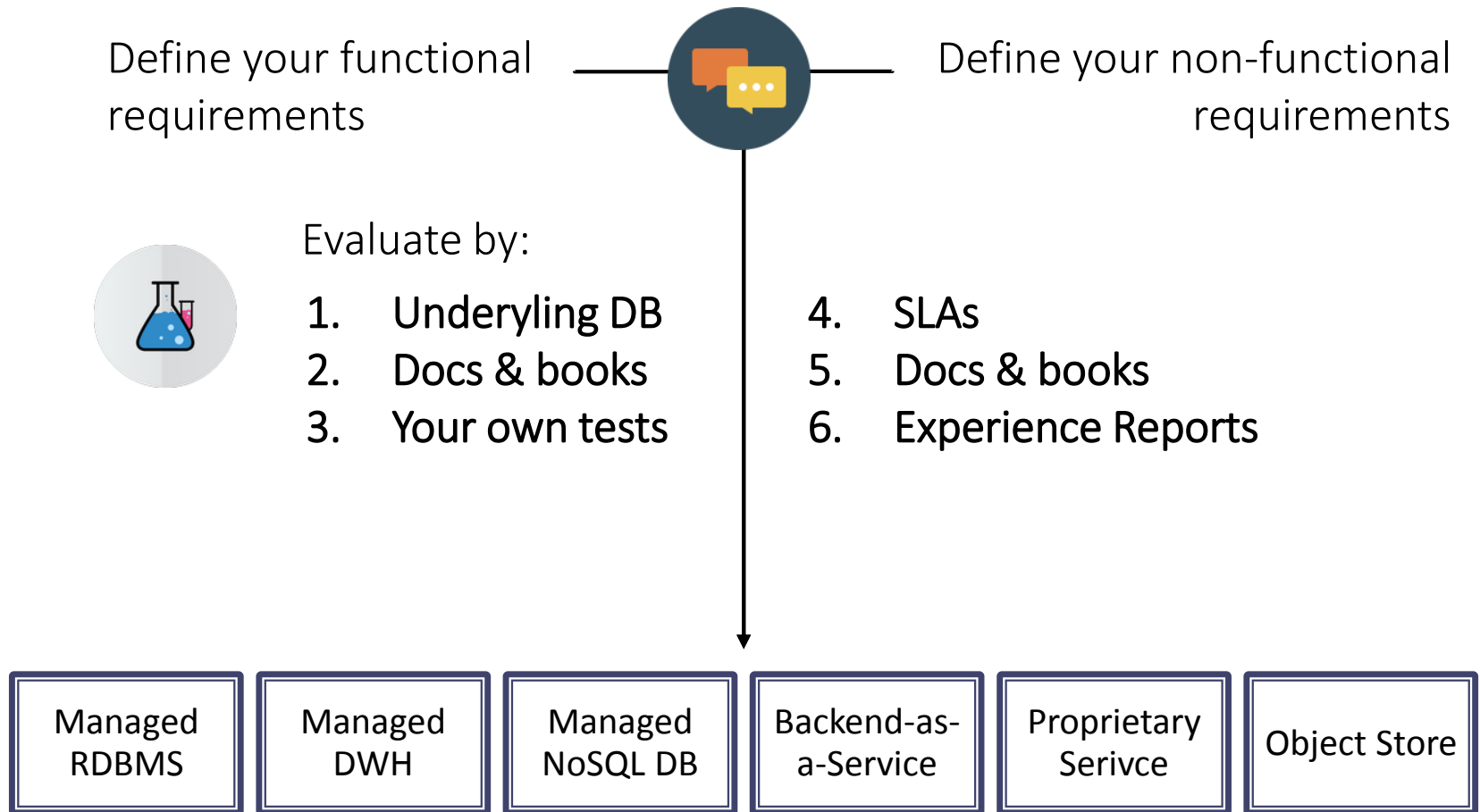
Wrap-up

How to choose a cloud database:



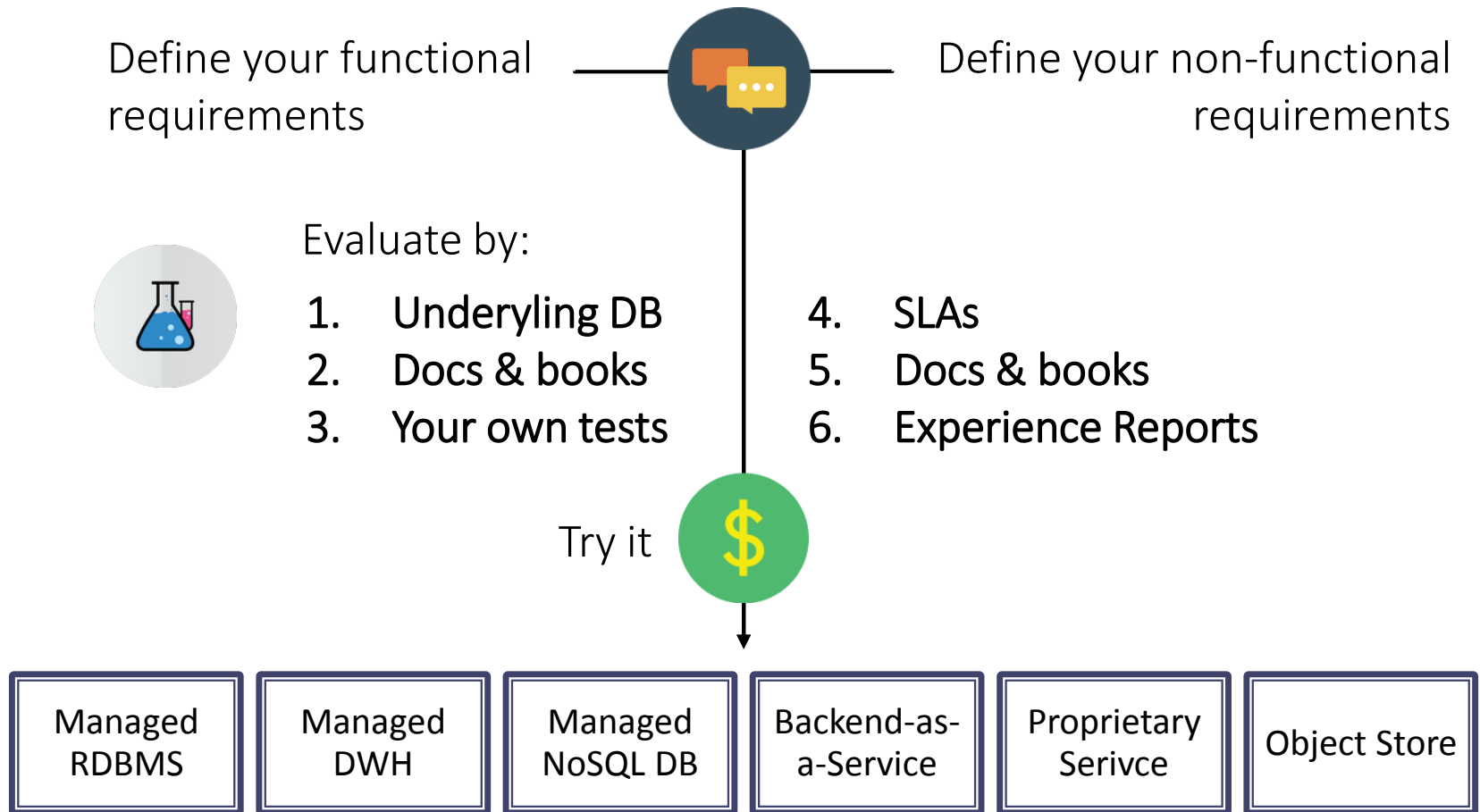
Wrap-up

How to choose a cloud database:

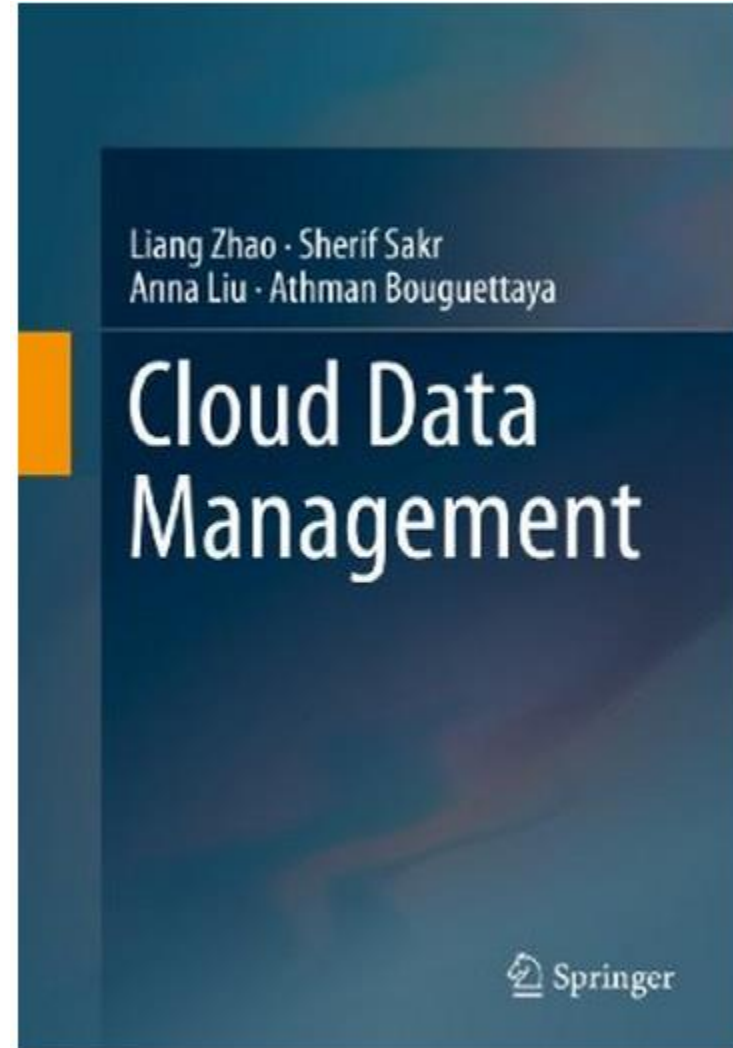
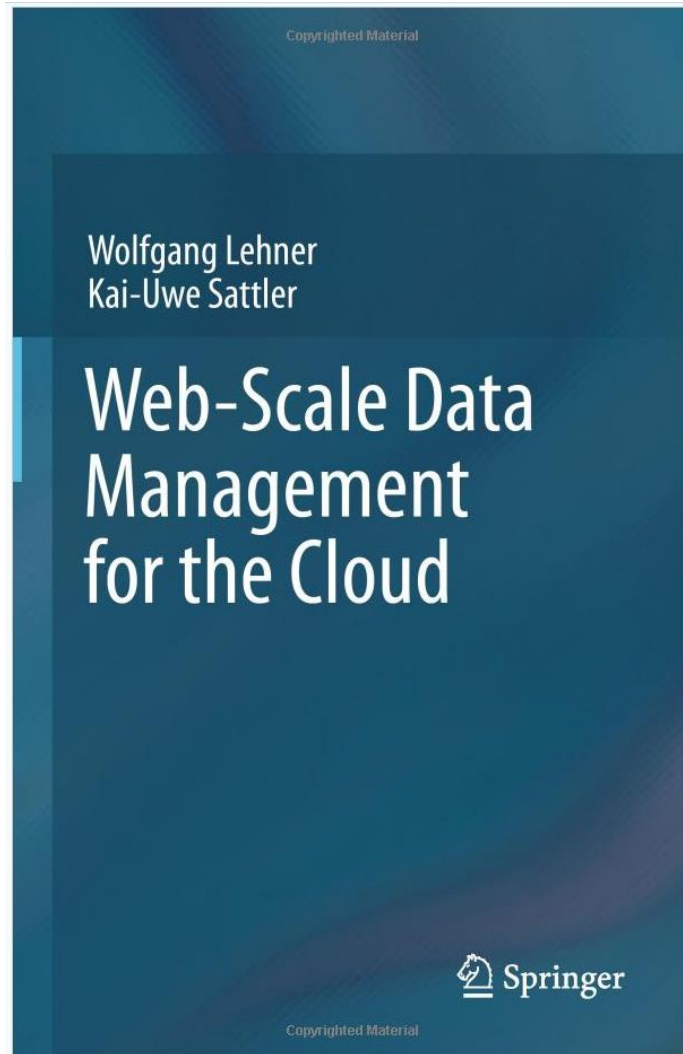


Wrap-up

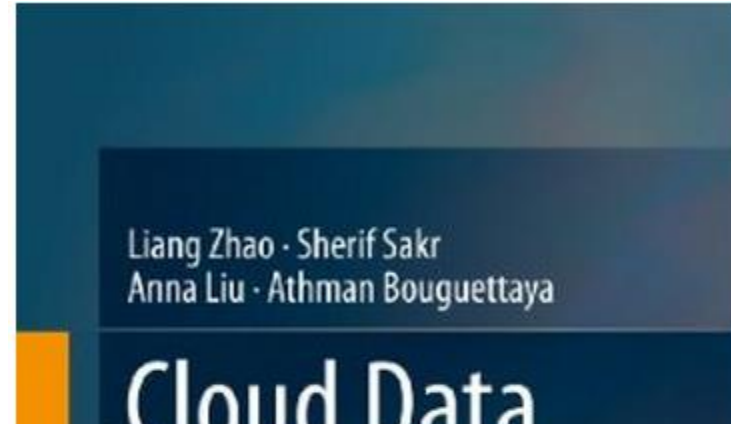
How to choose a cloud database:



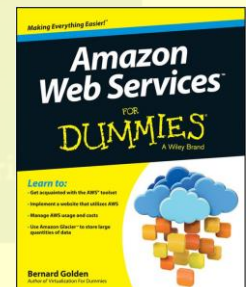
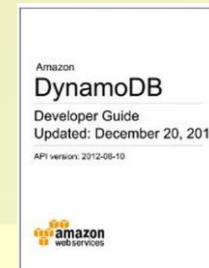
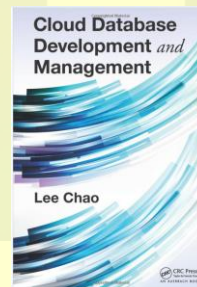
Book recommendations



Book recommendations



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



Blogs



<http://nosql.mypopescu.com/>



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



<http://www.dbms2.com/>

NoSQL Weekly

<http://www.nosqlweekly.com/>

Hacking, Distributed

<http://hackingdistributed.com/>



<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

Scalable Cloud Data Management Workshop 2013

Co-located with the IEEE BigData Conference. Silicon Valley, October 6th 2013.

Call for Papers

OSDI/SOSP
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>

