ETH zürich

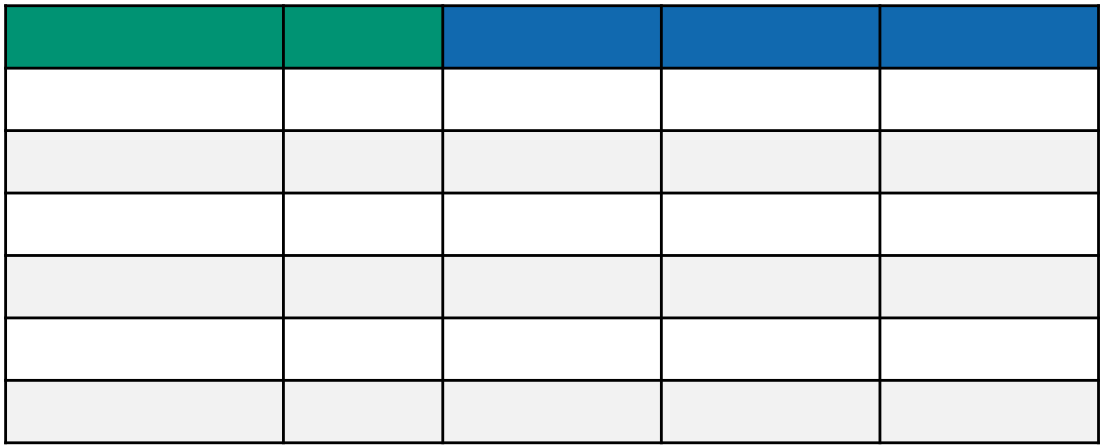
Ghislain Fourny

# Big Data for Engineers Spring 2020

## 3. Object Storage

# Where are we?

Last lecture:
Reminder on
relational databases

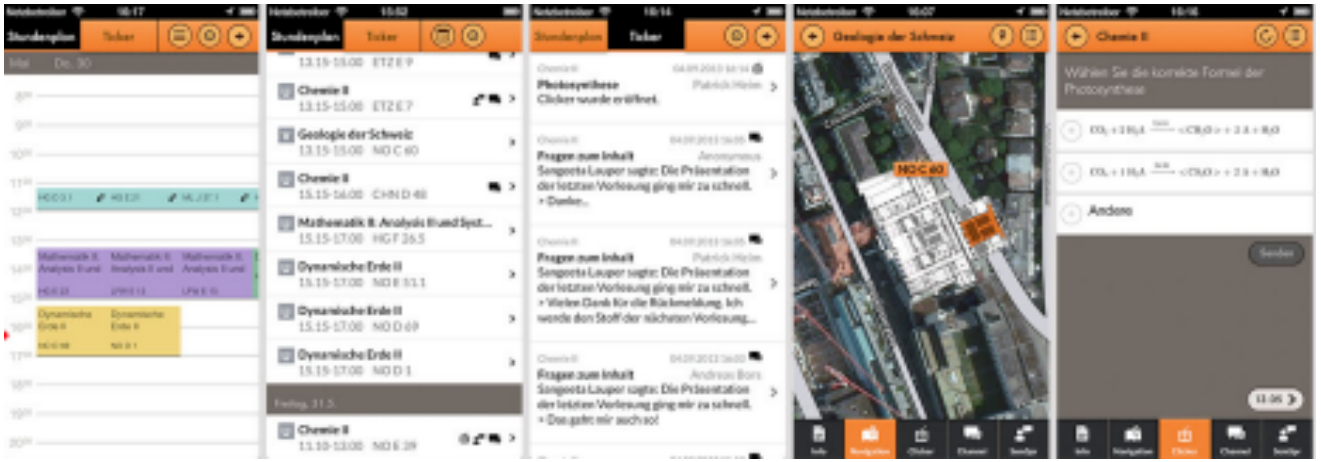


# Poll

Go *now* to:

https://eduapp-app1.ethz.ch/

Available on the App Store or install EduApp 2.x ANDROID APP ON Google play



# Where are we?

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

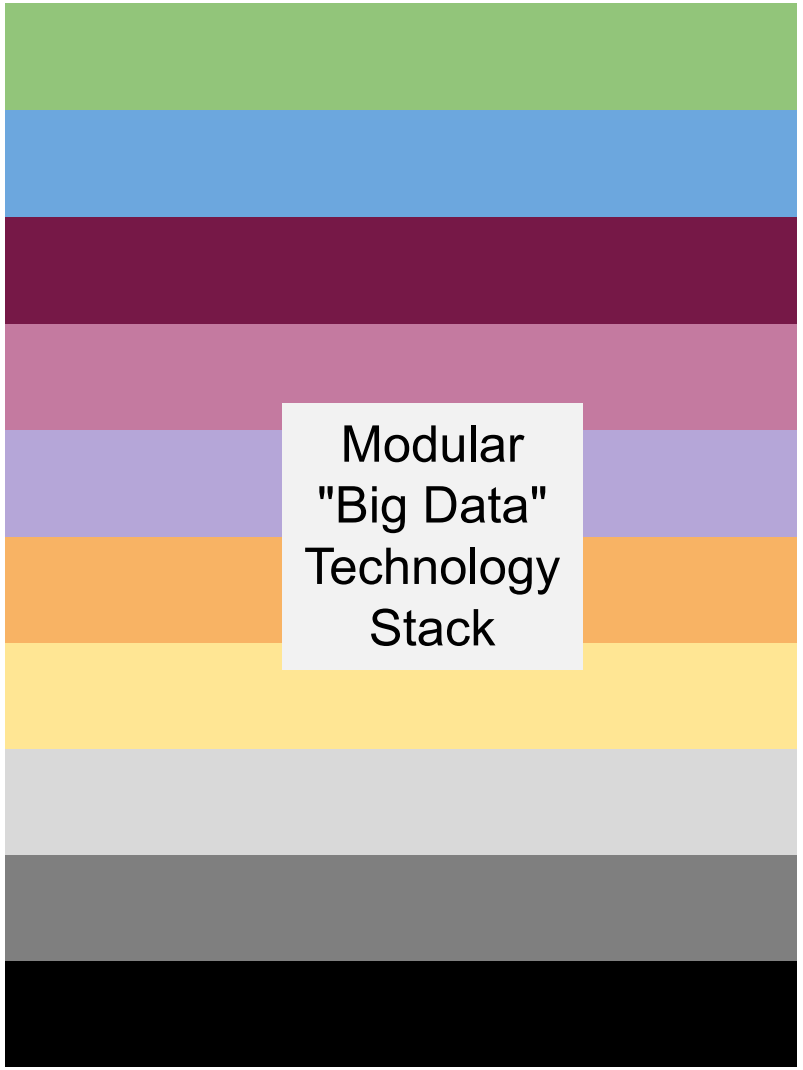
Relational databases
fit on a
**single machine**

## Where are we?

# Petabytes do not fit on a single machine

# The lecture journey

Monolithic
Relational
Database



# Not reinventing the wheel

99%

of what we learned with 48 years of SQL and relational can be reused

# Important take-away points

**Relational algebra:**

- Selection
- Projection
- Grouping
- Sorting
- Joining

# Important take-away points

**Language**

- SQL
- Declarative languages
- Functional languages
- Optimizations
- Query plans
- Indices

# Important take-away points

**What a table is made of**

- Table
- Rows
- Columns
- Primary key

# Important take-away points

SQL

**Consistency constraints**

- Tabular integrity
- Domain integrity
- Atomic integrity (1st normal form)
- Boyce-Codd normal form

# Important take-away points

**Consistency constraints**

SQL

- Tabular integrity
- Domain integrity
- Atomic integrity (1st normal form)
- Boyce-Codd normal form

NoSQL

- NEW Heterogeneous data
- NEW Nested data
- NEW Denormalized data

# Important take-away points

**Transactions**

ACID

- ✗ Atomicity
- ✗ Consistency
- ✗ Isolation
- ✗ Durability

# Important take-away points

## Transactions

ACID

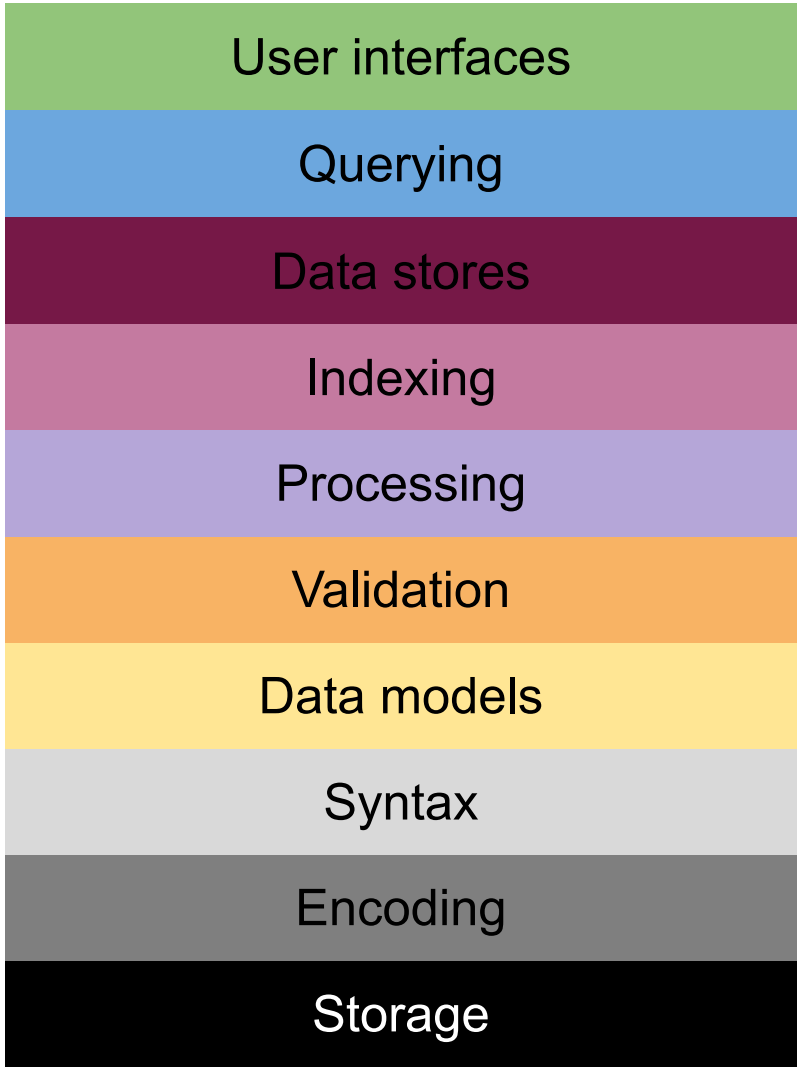
- ✗ Atomicity
- ✗ Consistency
- ✗ Isolation
- ✗ Durability

CAP

- NEW Atomic Consistency
- NEW Availability
- NEW Partition tolerance
- NEW Eventual Consistency

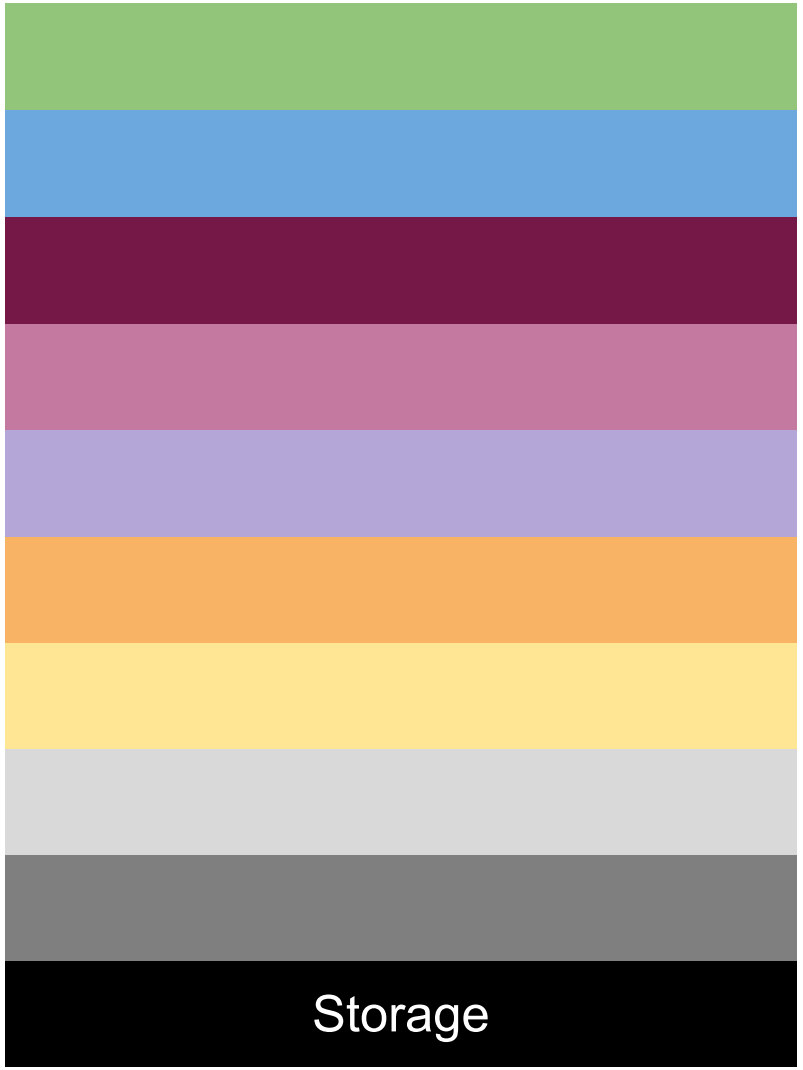
# The stack

# The stack


User interfaces
Querying
Data stores
Indexing
Processing
Validation
Data models
Syntax
Encoding
Storage

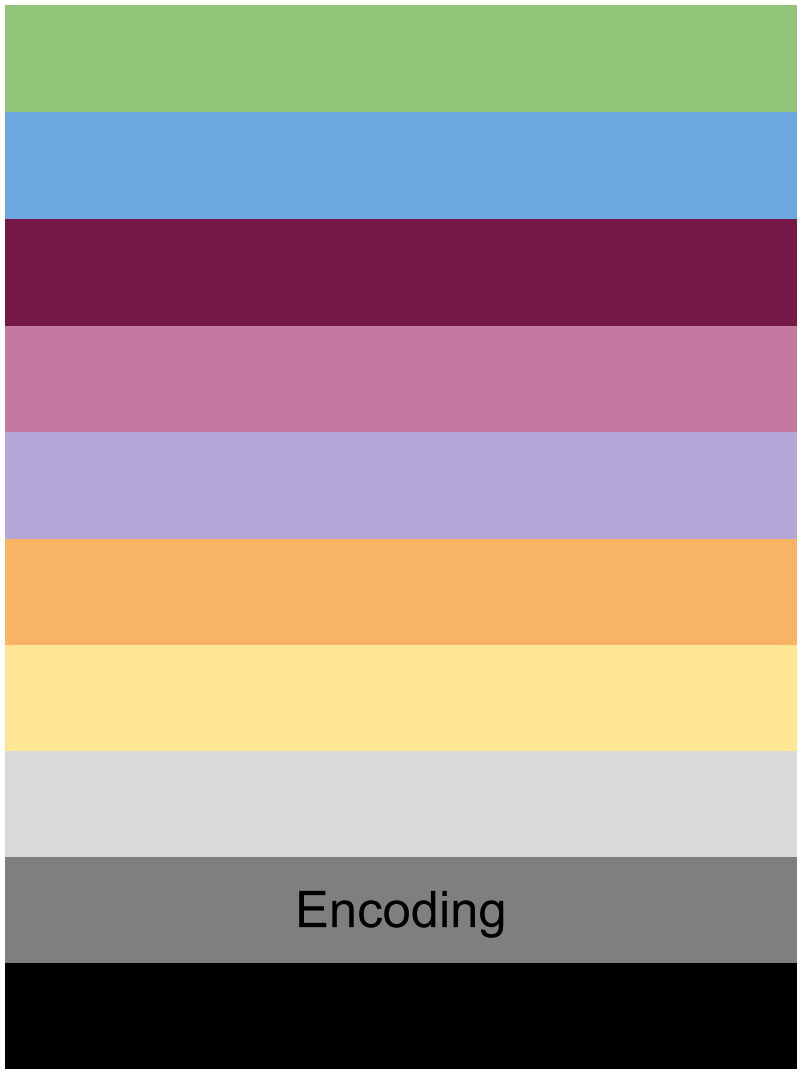
# The stack: Storage

Local filesystem
NFS
GFS
HDFS
S3
Azure Blob Storage



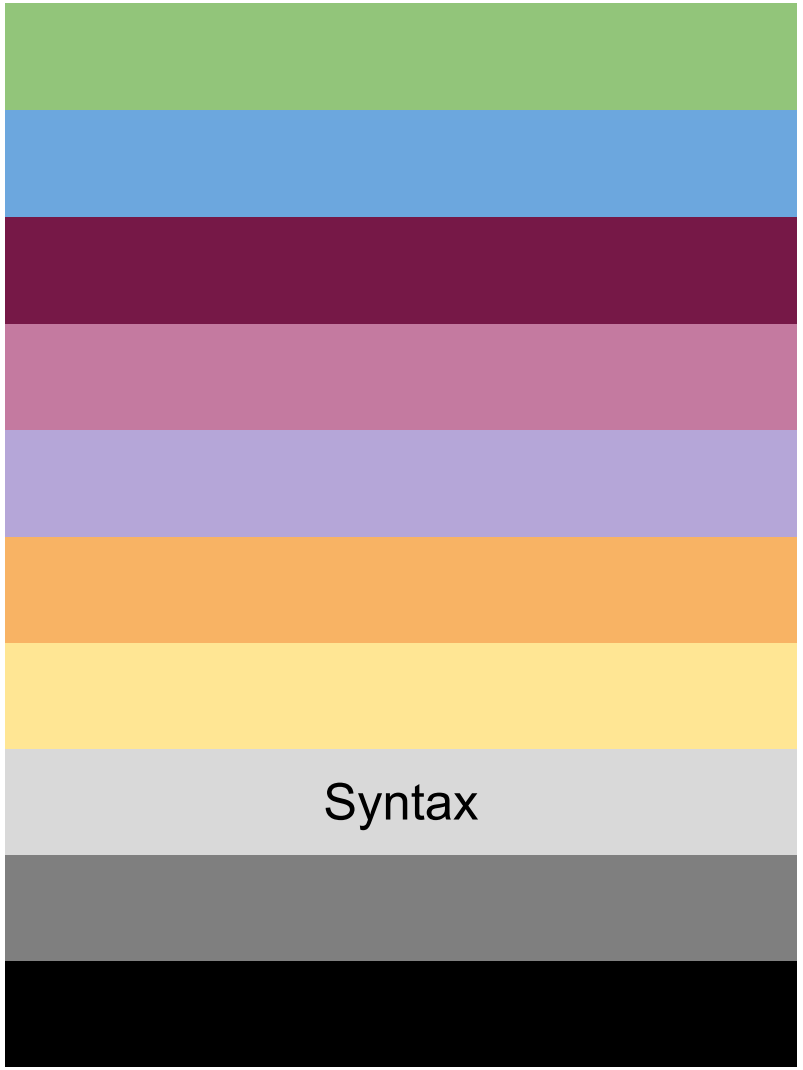
# The stack: Encoding

ASCII
ISO-8859-1
UTF-8
BSON



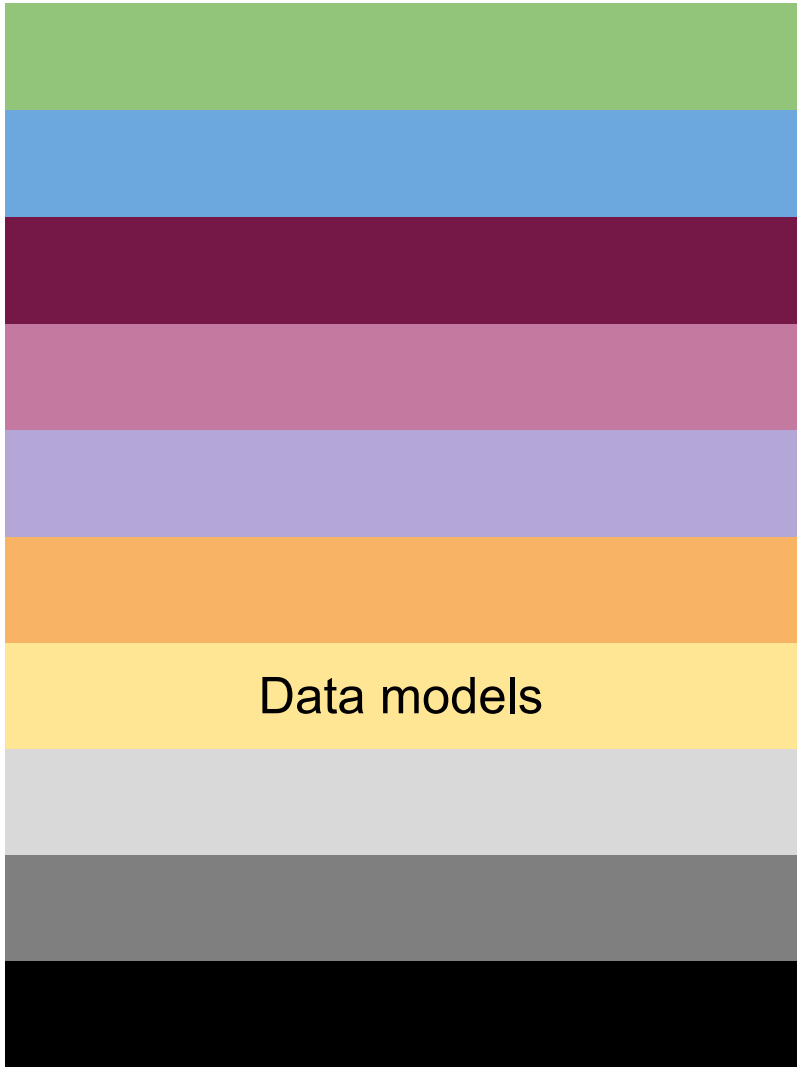
# The stack: Syntax

Text
CSV
XML
JSON
RDF/XML
Turtle
XBRL



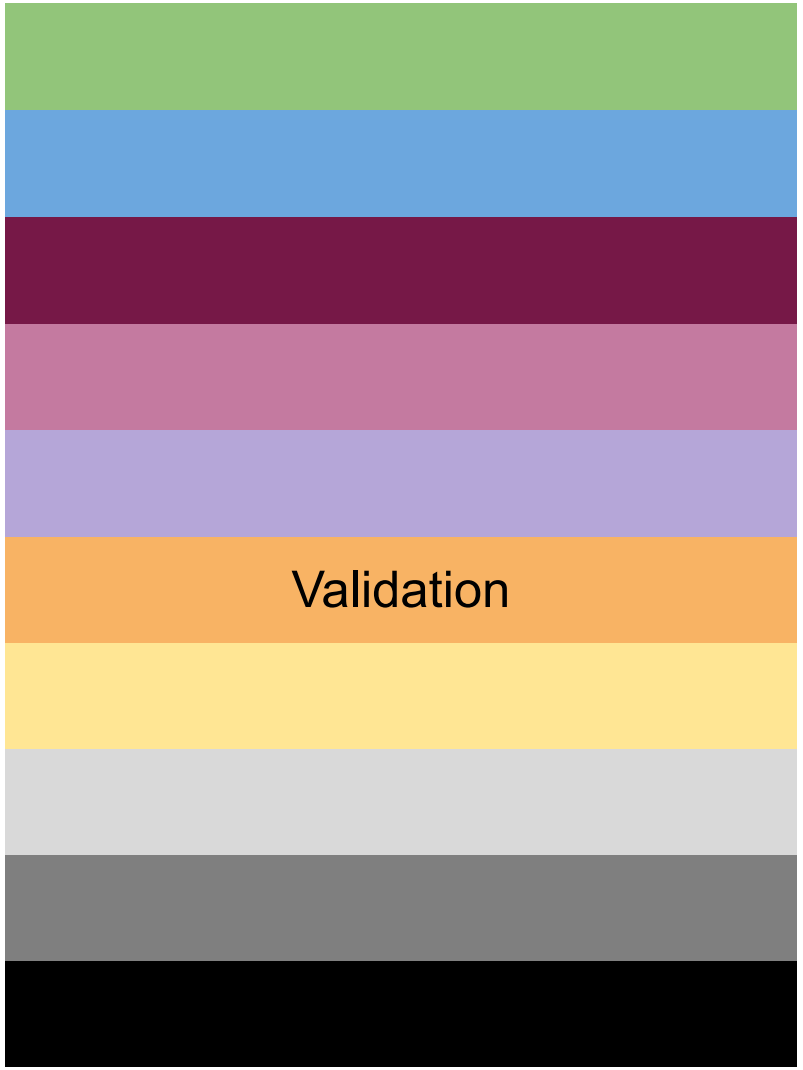
# The stack: Data models

Tables: Relational model
Trees: XML Infoset, XDM
Graphs: RDF
Cubes: OLAP



# The stack: Validation

XML Schema
JSON Schema
Relational schemas
XBRL taxonomies

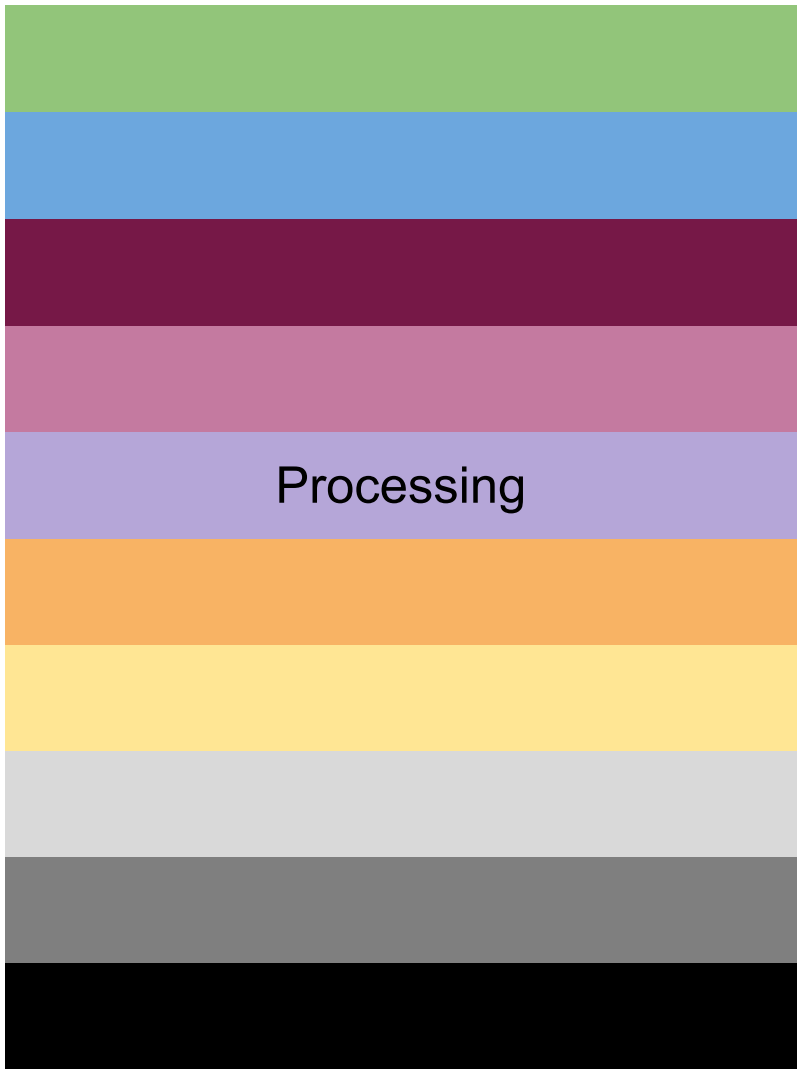


# The stack: Processing

Two-phase processing:
MapReduce

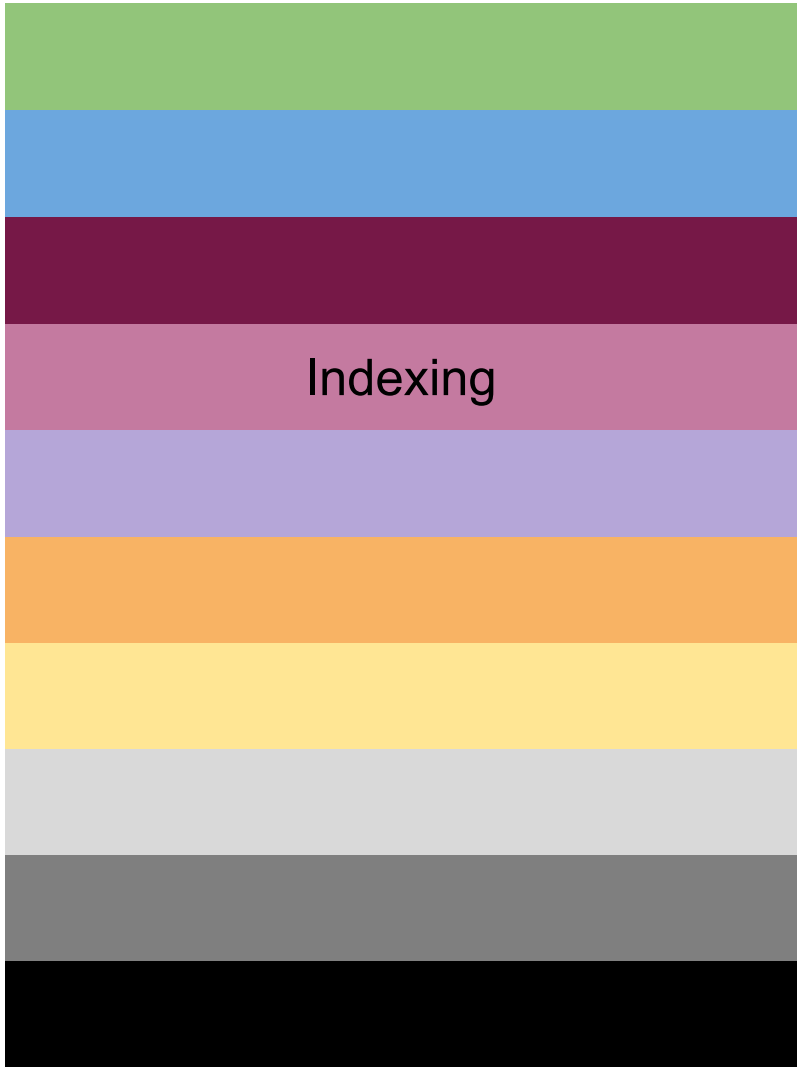
DAG-driven processing:
Tez, Spark, Flink, Ray

Elastic computing:
EC2



# The stack: Indexing

Key-value stores
Hash indices
B-Trees
Geographical indices
Spatial indices

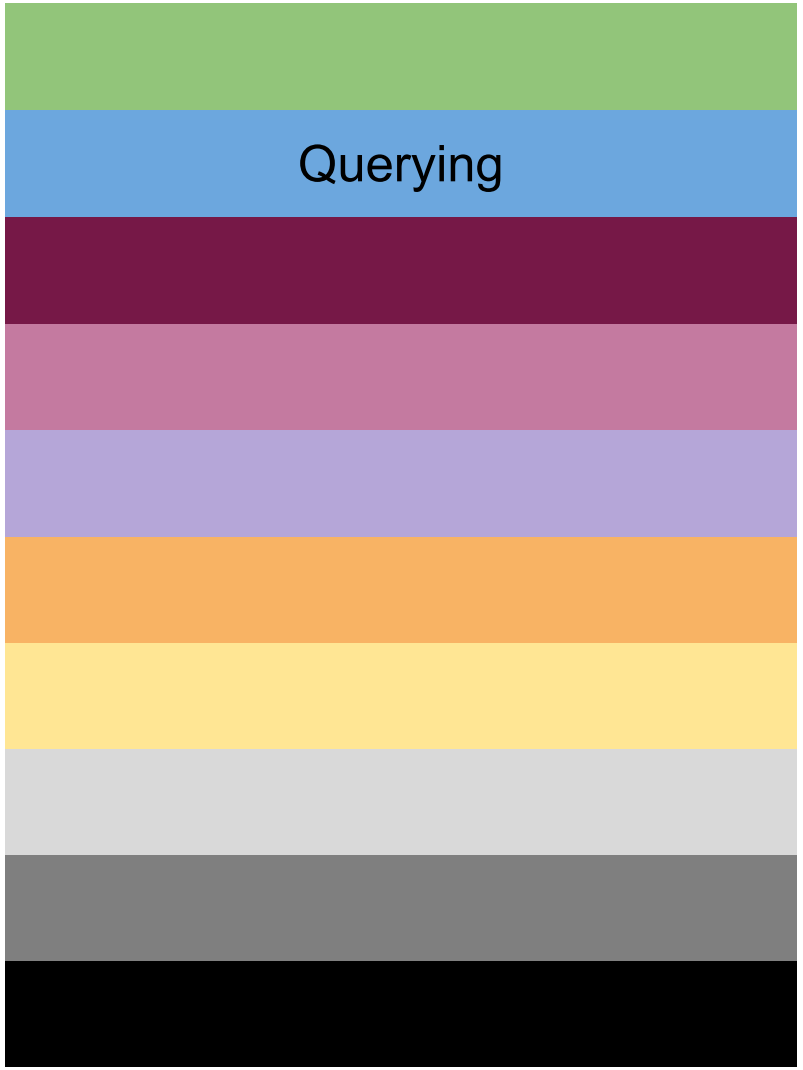


# The stack: Data stores

RDBMS
(Oracle/IBM/Microsoft)
MongoDB
CouchBase
ElasticSearch
Hive
HBase
MarkLogic
Cassandra
...

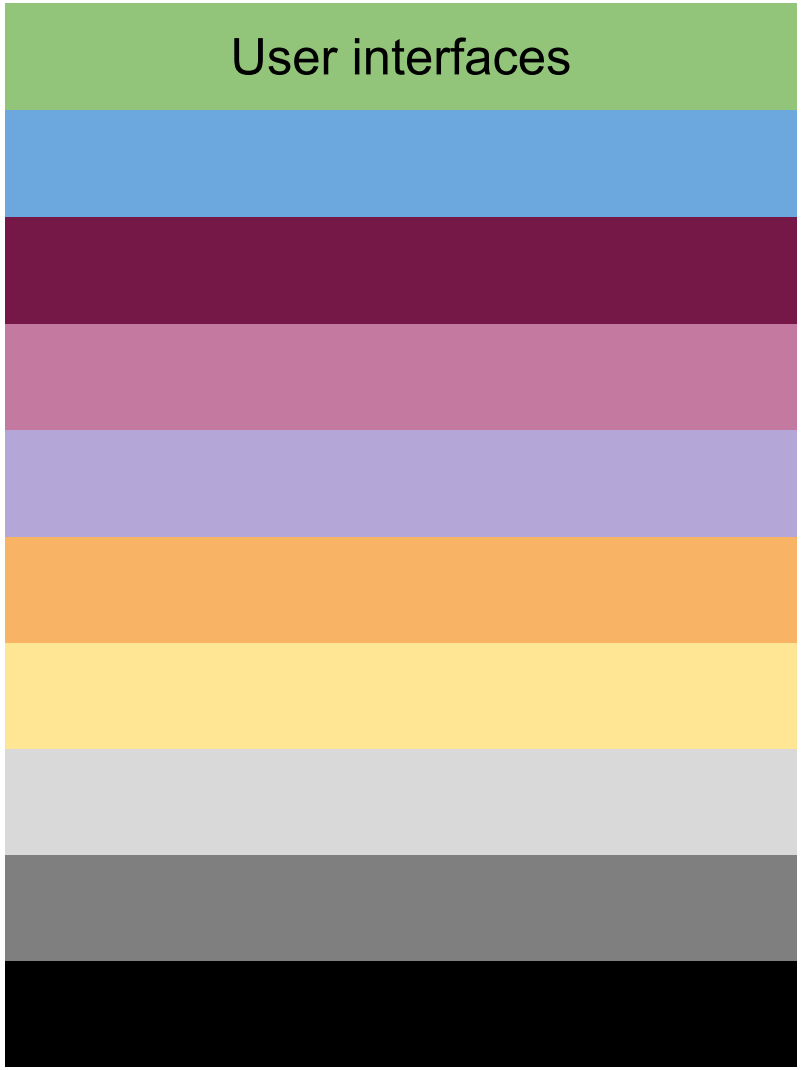
# The stack: Querying

SQL
XQuery
JSONiq
N1QL
MDX
SPARQL
REST APIs

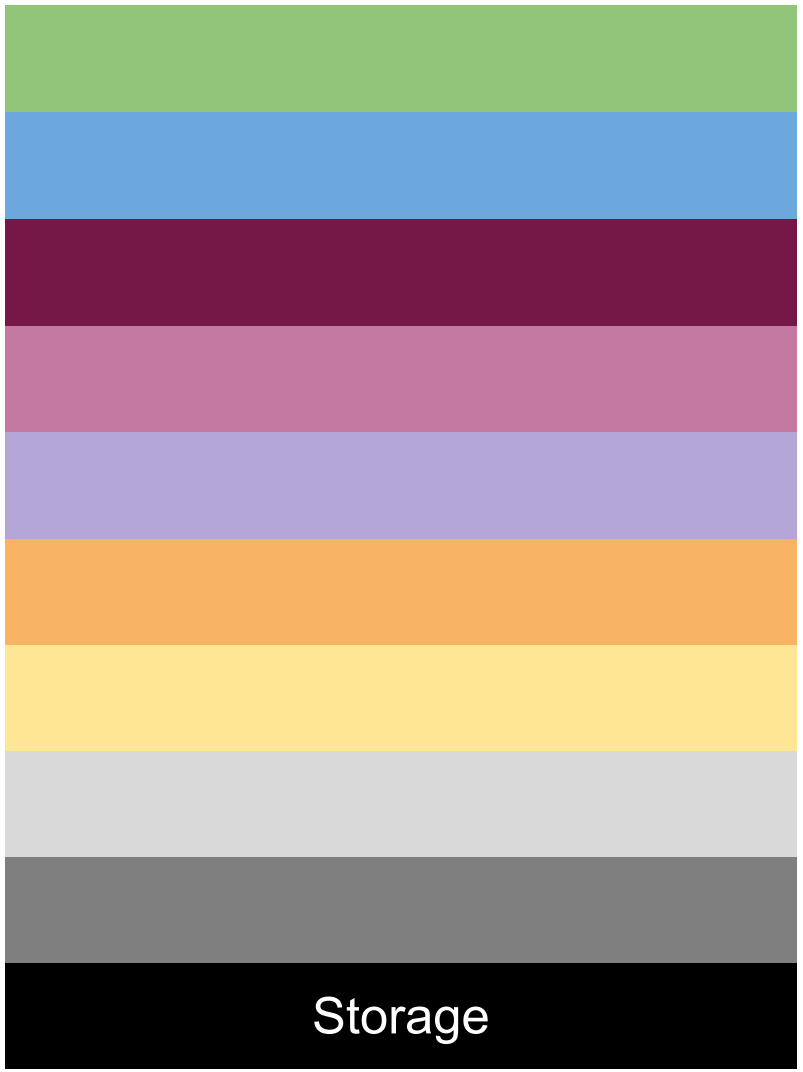


# The stack:
# User interfaces (UI)

Excel
Access
Tableau
Qlikview
BI tools



# The stack: Storage

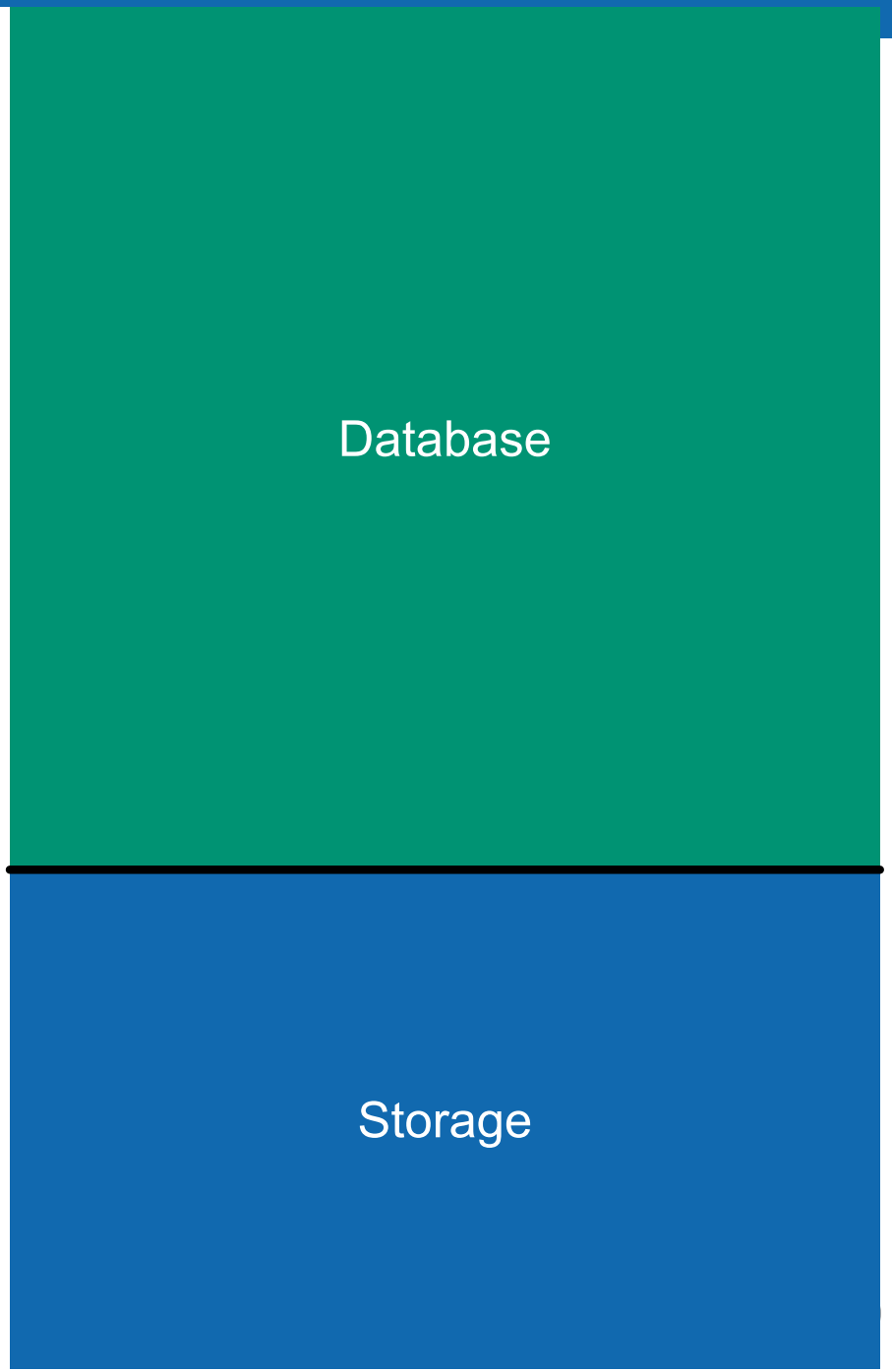



Today!

# Storage: from a single machine to a cluster

## Storage

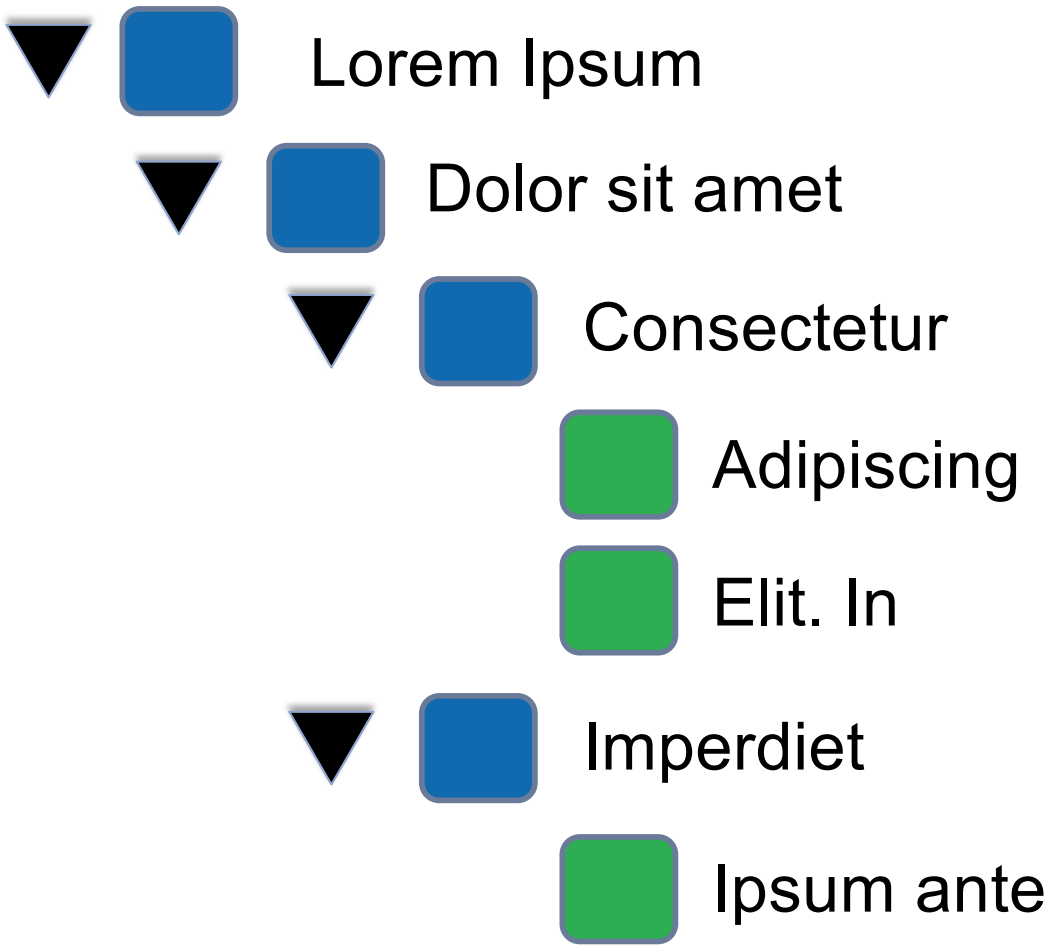
Data needs to be stored somewhere



# Let's start from the 70s...

Vitaly Korovin / 123RF Stock Photo

## File storage




Files organized in a **hierarchy**

## What is a file made of?

Content

\+ Metadata

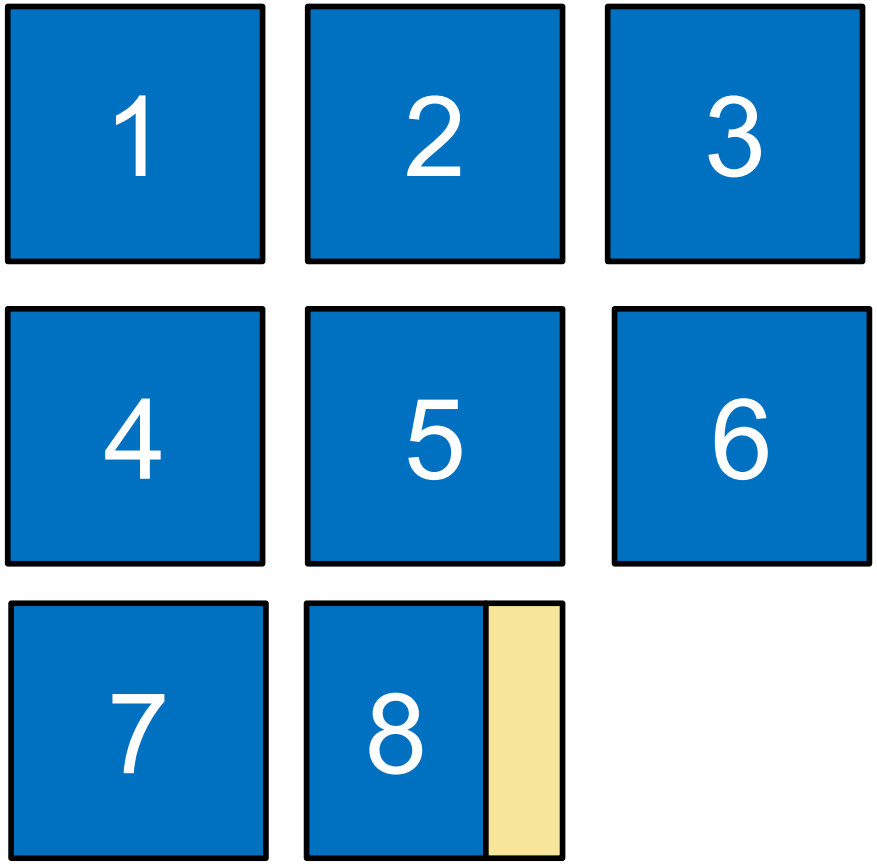
File

# File Metadata

```
$ ls -l
total 48
drwxr-xr-x   5 gfourny staff  170 Jul 29 08:11 2009
drwxr-xr-x  16 gfourny staff  544 Aug 19 14:02 Exercises
drwxr-xr-x  11 gfourny staff  374 Aug 19 14:02 Learning Objectives
drwxr-xr-x  18 gfourny staff  612 Aug 19 14:52 Lectures
-rw-r--r--   1 gfourny staff 1788 Aug 19 14:04 README.md
```

Fixed "**schema**"

## File Content: Block storage




Files content stored in

**blocks**

# Local storage

Local Machine

# Local storage

- Local Machine
- LAN (NAS)

LAN = local-area network
NAS = network-attached storage

# Local storage

- ✅ Local Machine
- ✅ LAN (NAS)
- ❌ WAN

LAN = local-area network
NAS = network-attached storage
WAN = wide-area network

# Scaling Issues

# Scaling Issues

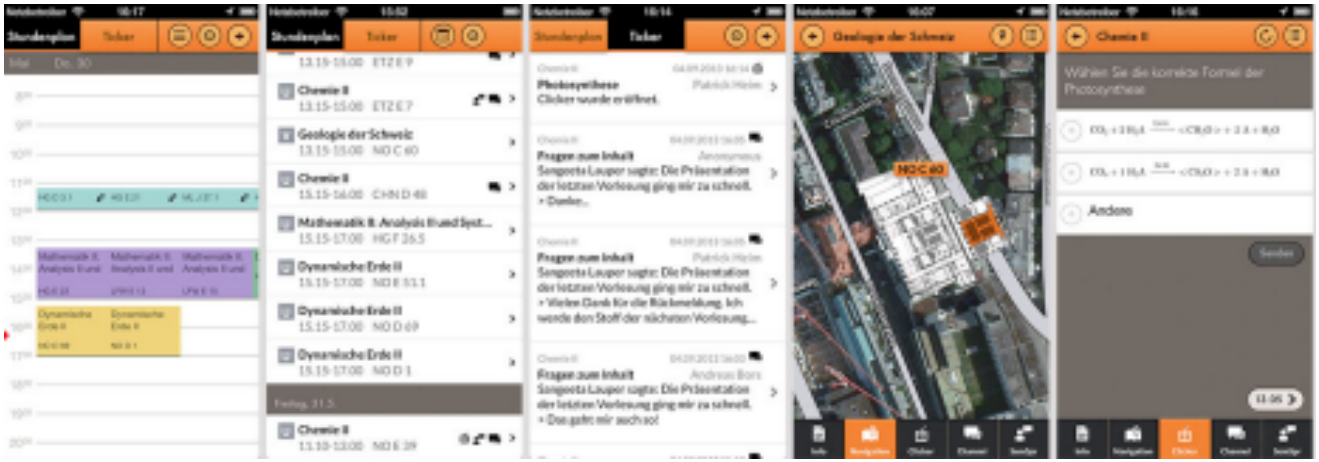
# Scaling Issues

# Poll

Go *now* to:

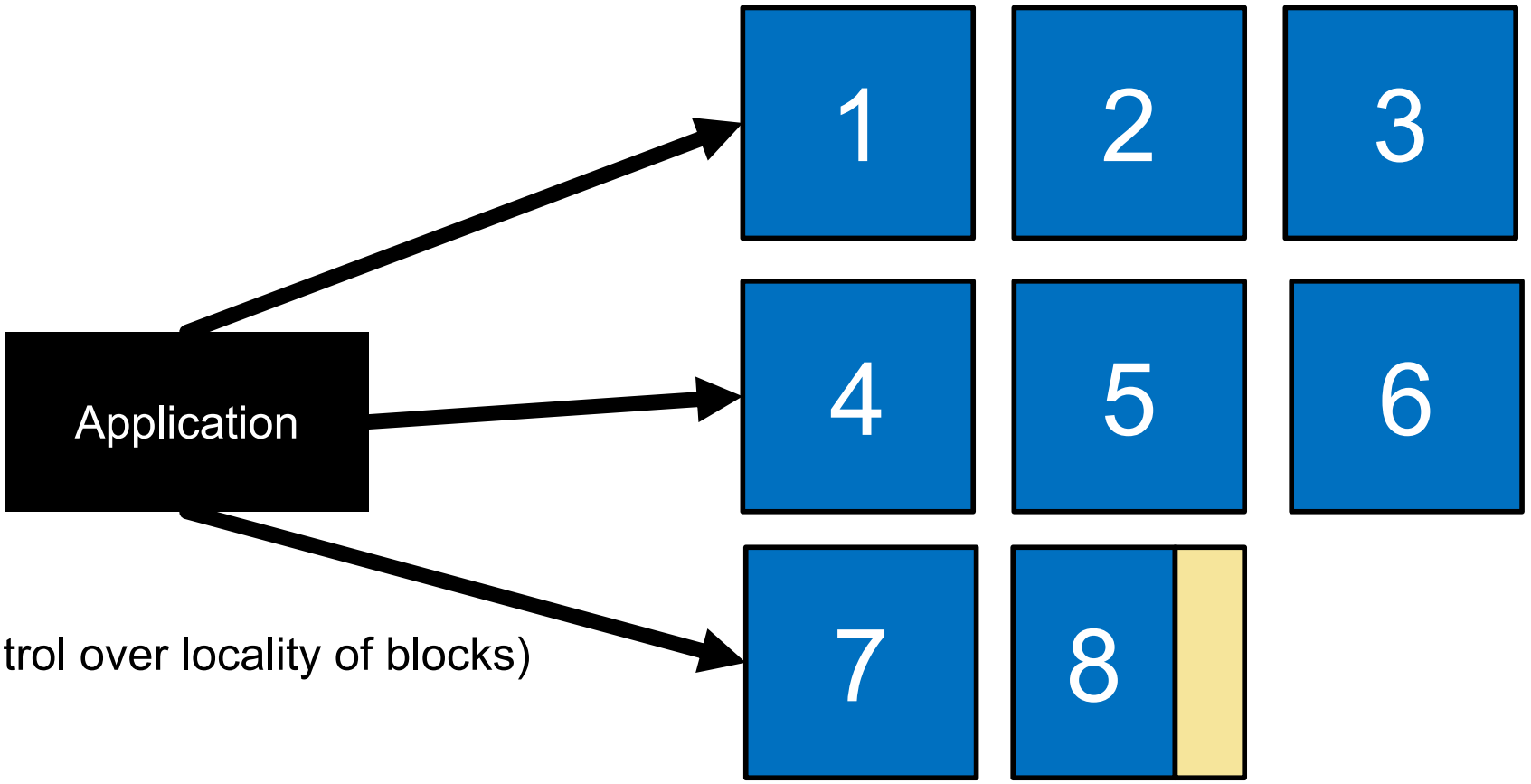
https://eduapp-app1.ethz.ch/

Available on the App Store or install EduApp 2.x ANDROID APP ON Google play



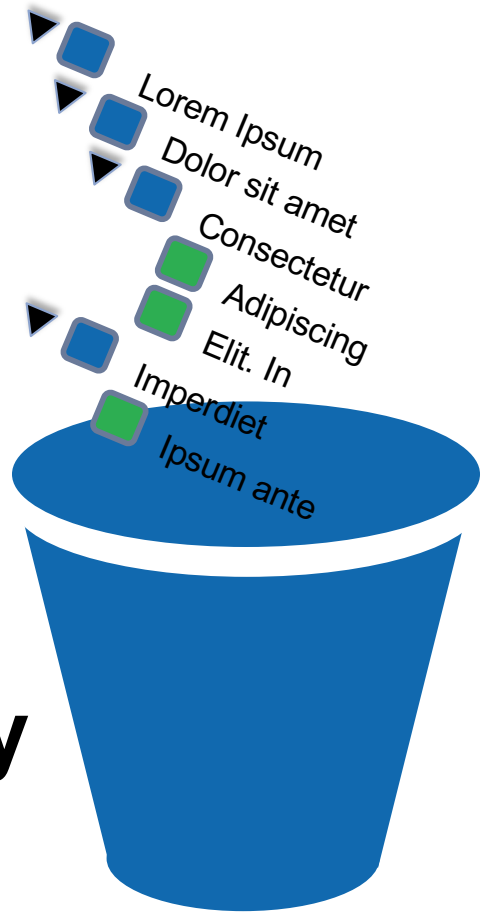
# Analogy: Tesla batteries (up to 2017)

# Better performance: Explicit Block Storage



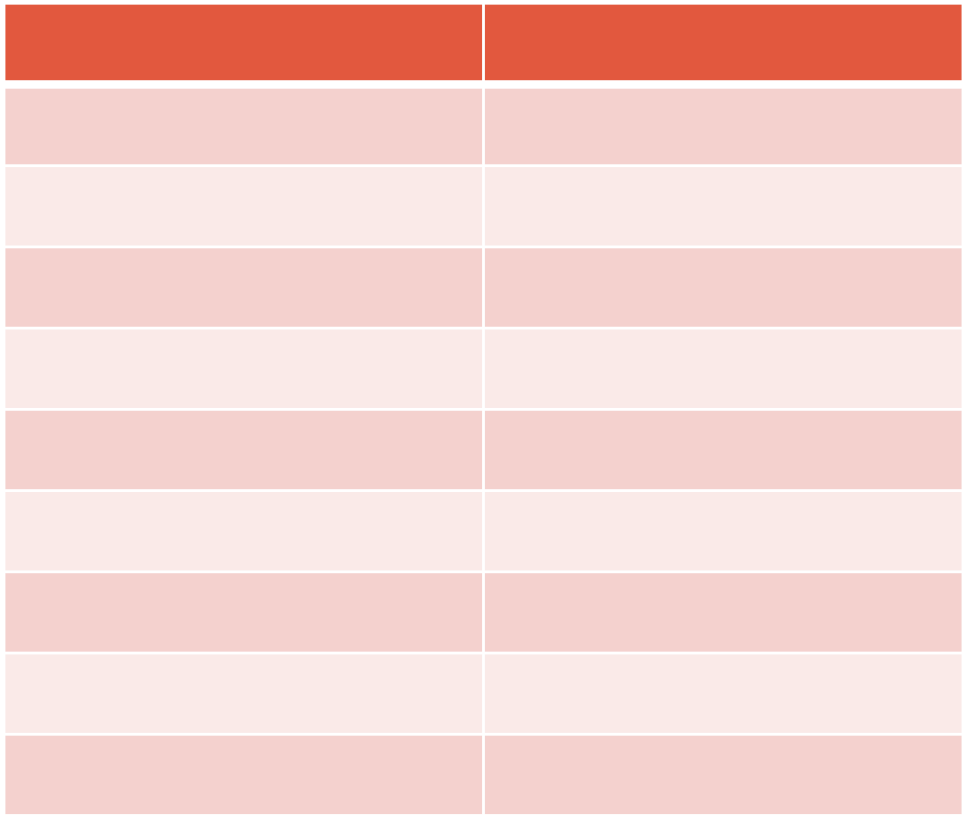

(Control over locality of blocks)

# So how do we make this scale?



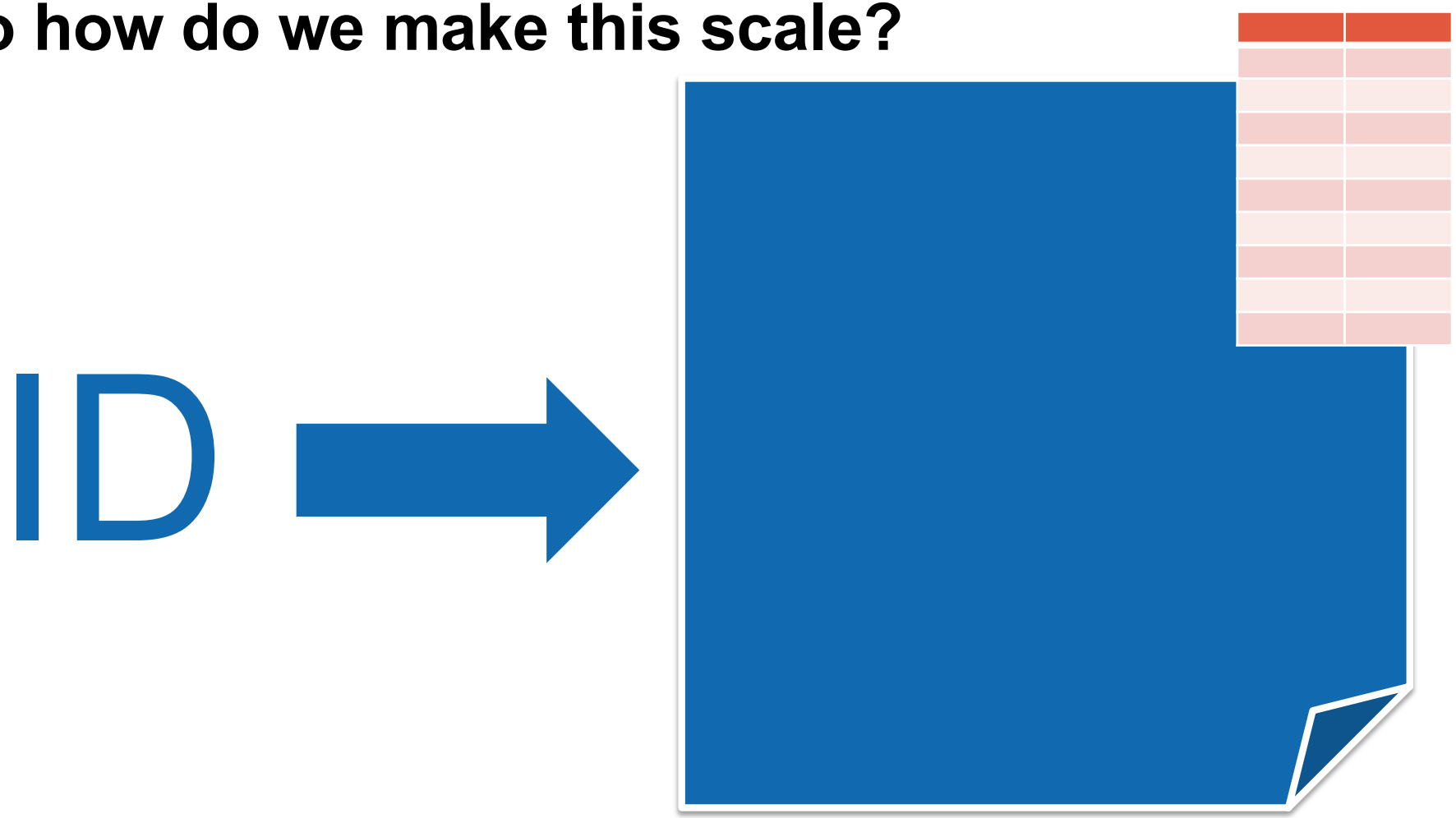

## 1. We throw away the hierarchy!

# So how do we make this scale?



## 2. We make metadata flexible

## So how do we make this scale?




## 3. We make the data model trivial

# So how do we make this scale?

# 4. We use commodity hardware

# ... and we get Object Storage

# ... and we get Object Storage

"Black-box" objects

## ... and we get Object Storage

"Black-box" objects

Flat and global key-value model

## ... and we get Object Storage

"Black-box" objects

Flat and global key-value model

Flexible metadata

# ... and we get Object Storage

"Black-box" objects

Flat and global key-value model

Flexible metadata

Commodity hardware

# Scale

# One machine's not good enough. How do we scale?

# Approach 1: scaling up

# Approach 1: scaling up

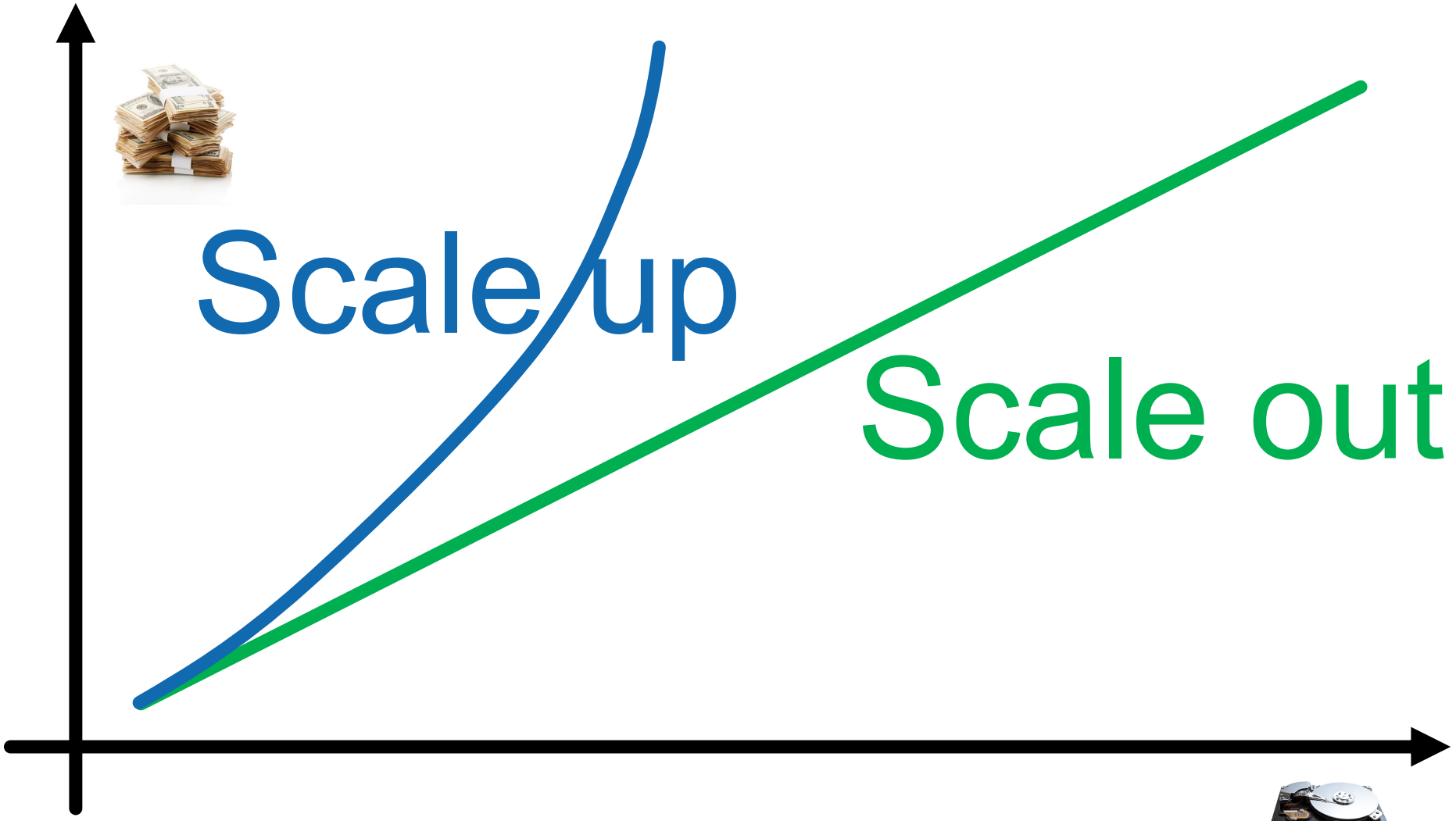
# Approach 2: scaling out

# Approach 2: scaling out

# Approach 2: scaling out

# Approach 2: scaling out

# Hardware price comparison



# Approach 3: be smart



Viktorija Reuta / 123RF Stock Photo

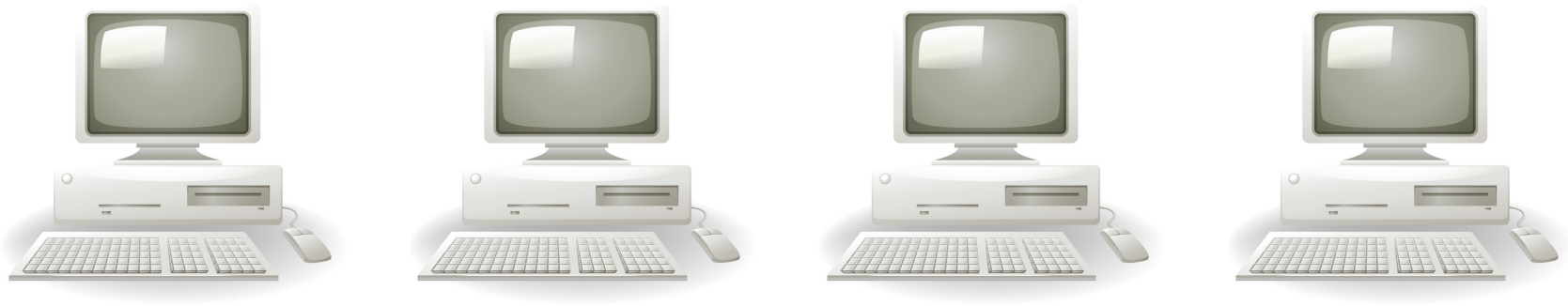
## Approach 3: be smart

"You can have a second computer **once you've shown you know how to use the first one.**"

Paul Barham

# In this lecture

Approach 2



Scale out

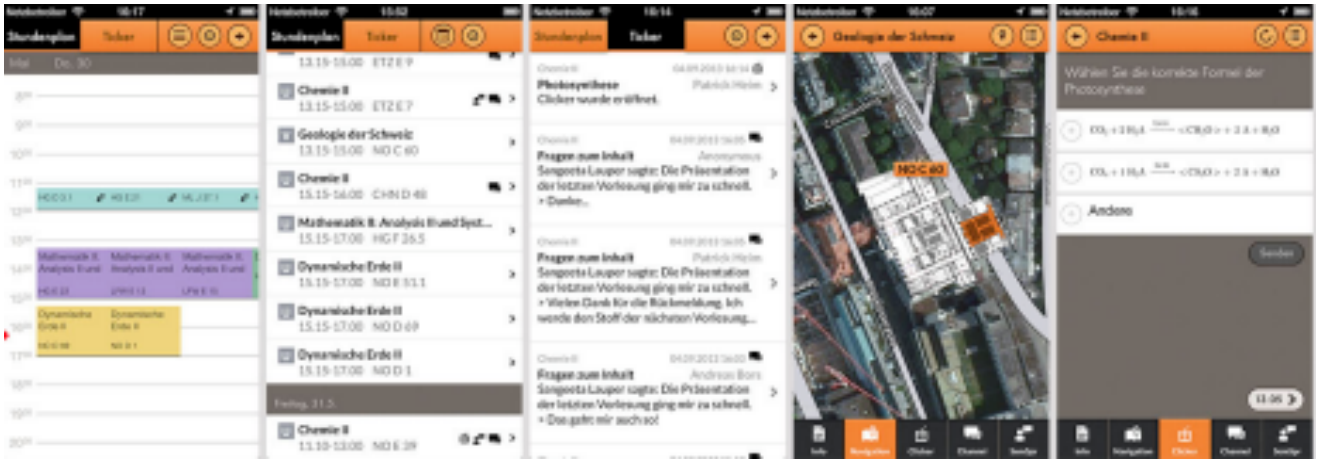

# Data centers

# Poll

Go *now* to:

https://eduapp-app1.ethz.ch/

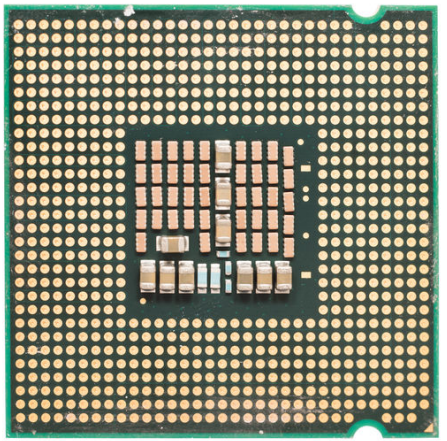
Available on the App Store or install EduApp 2.x ANDROID APP ON Google play



# Numbers - computing

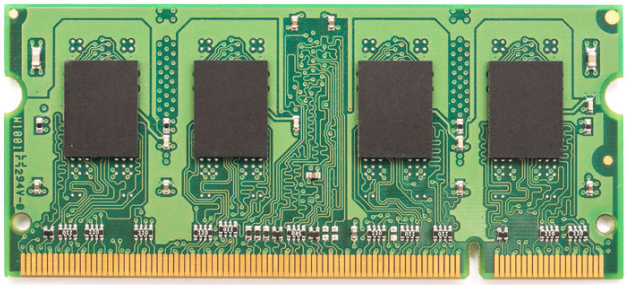
1,000-100,000
machines in a data center

1-100 cores
per server



## Numbers - storage

# 1-20 TB
local storage
per server



# 16GB-6TB
of RAM per server

# Numbers - network

1-100 GB/s
network bandwith
for a server

## Racks

Height in "rack units"
(e.g., 42 RU)

## Racks

Modular:
- servers
- storage
- routers
- …

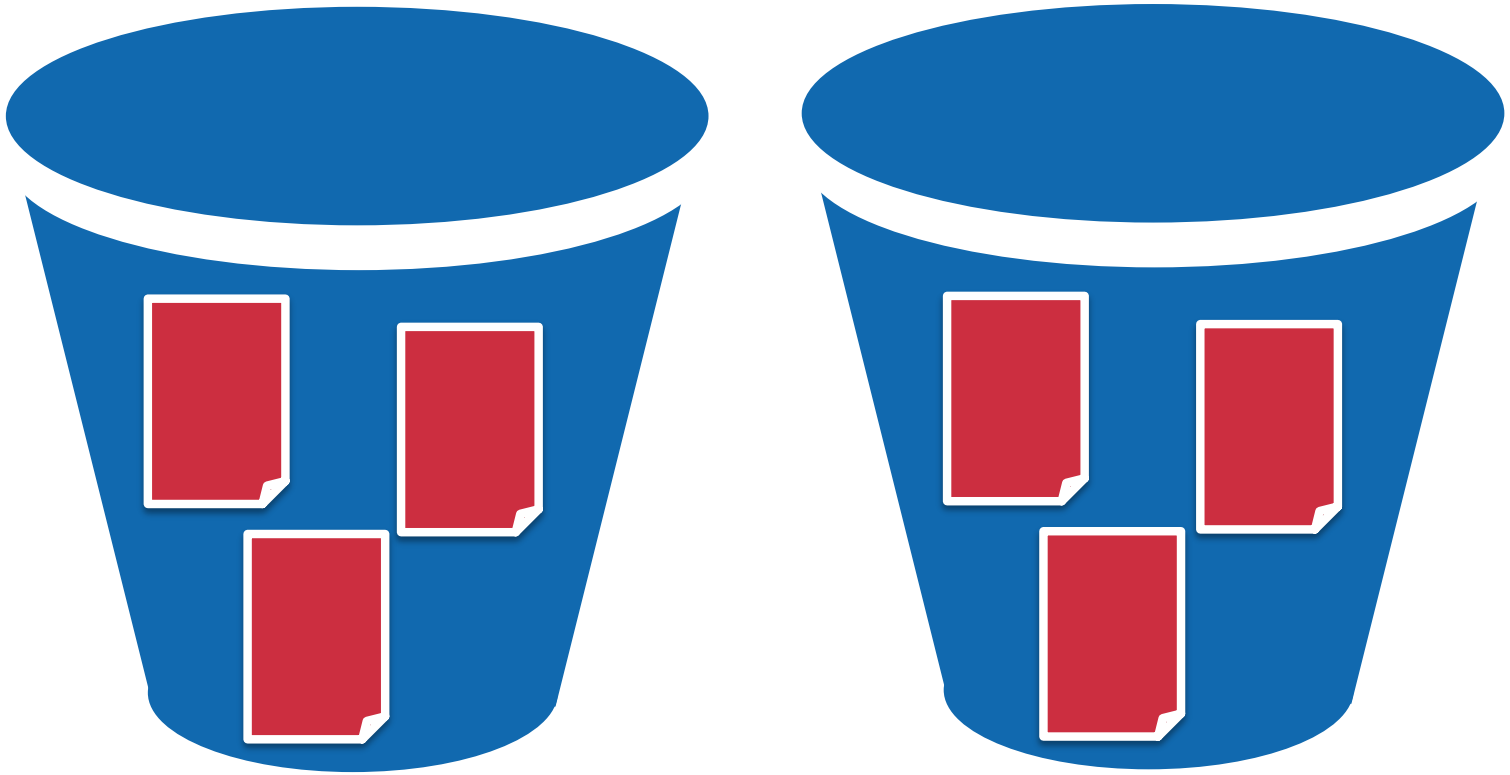
# Rack servers

1-4 RU


Lenovo ThinkServer RD430 Rack Server

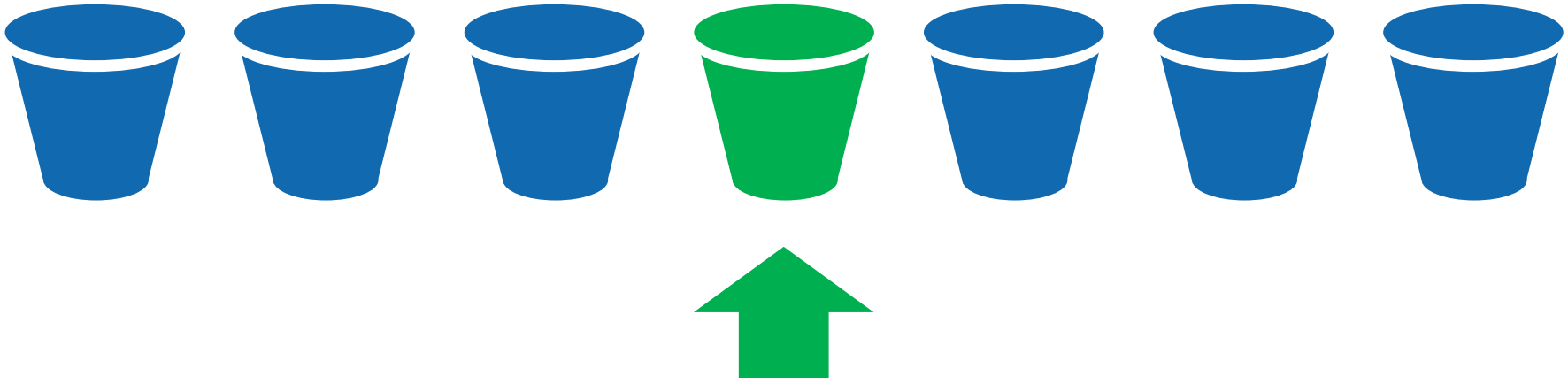

# Amazon S3

# S3 Model



# S3 Model

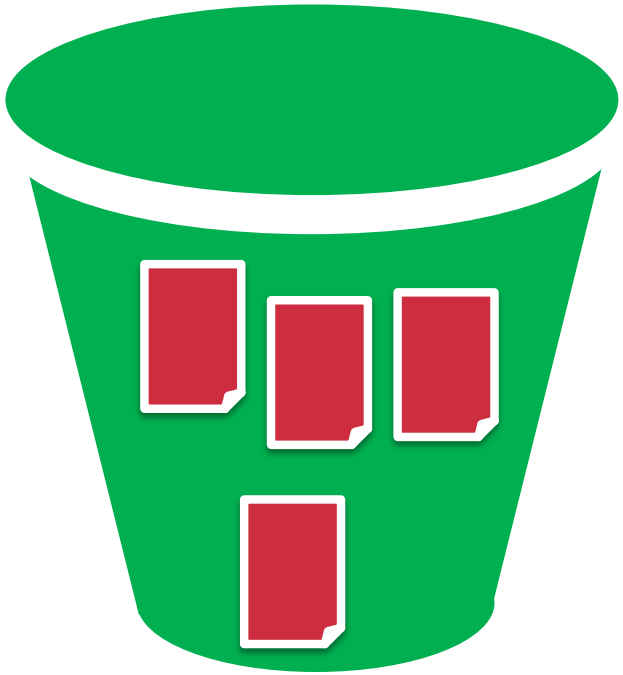
# S3 Model



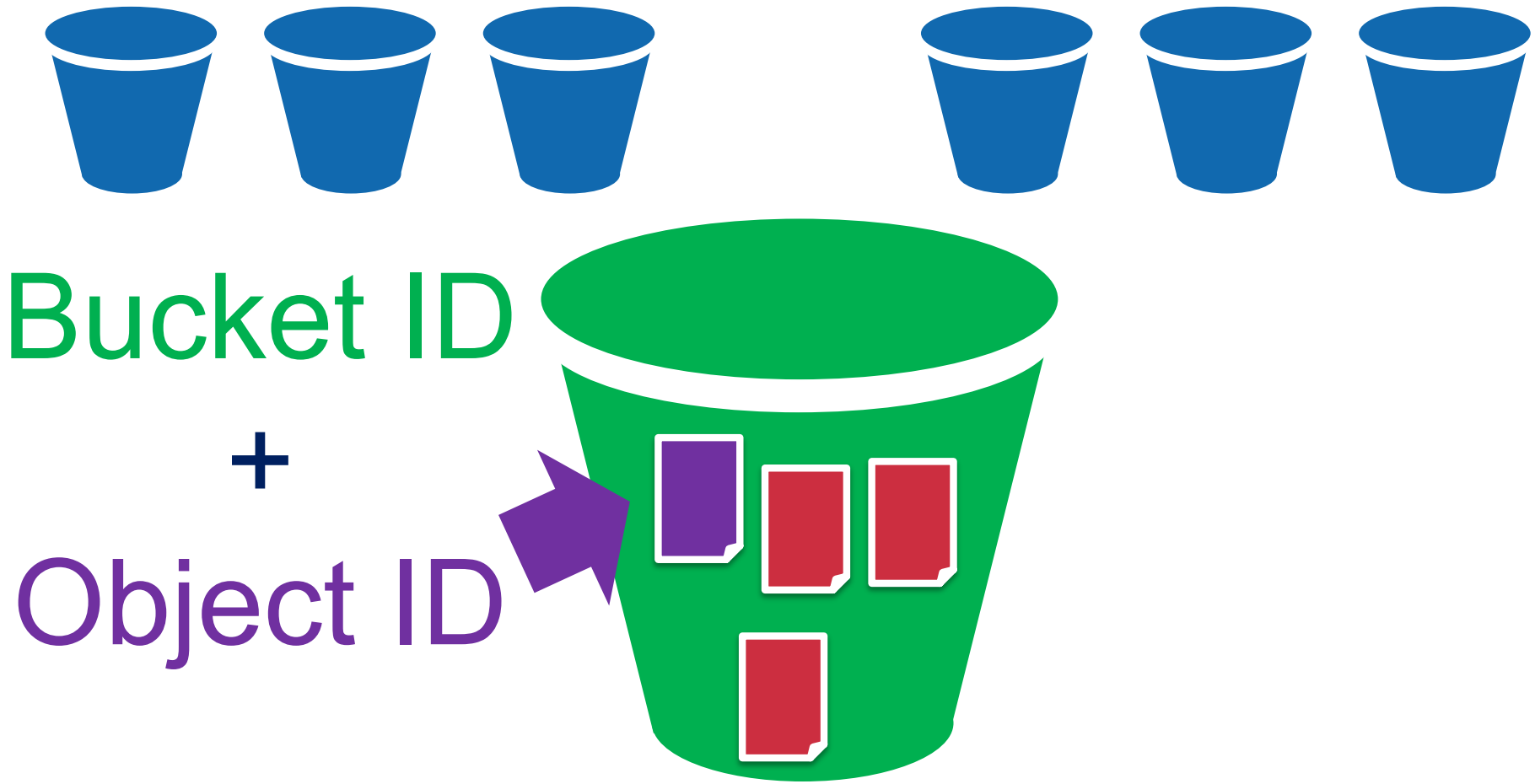
Bucket ID

# S3 Model

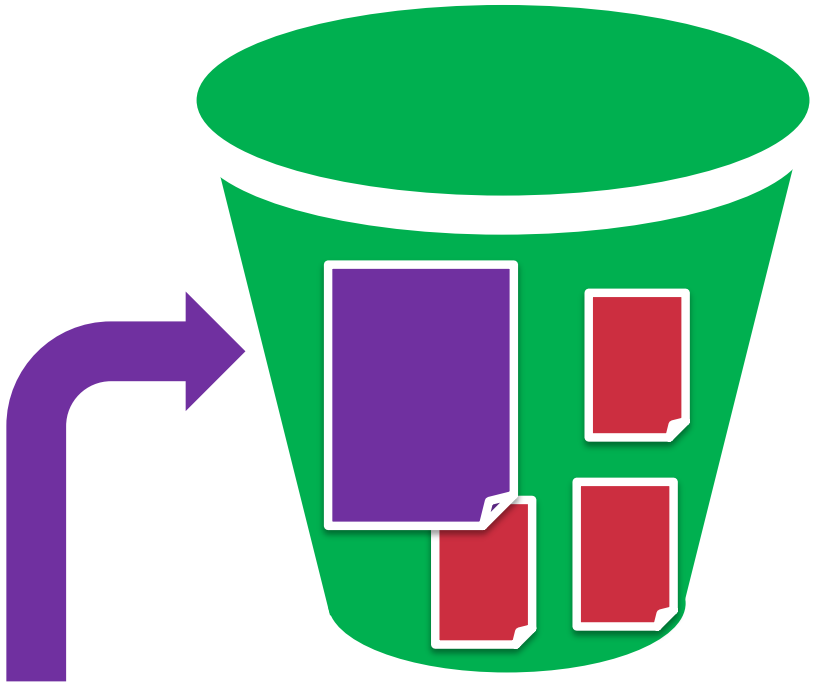
Bucket ID



# S3 Model

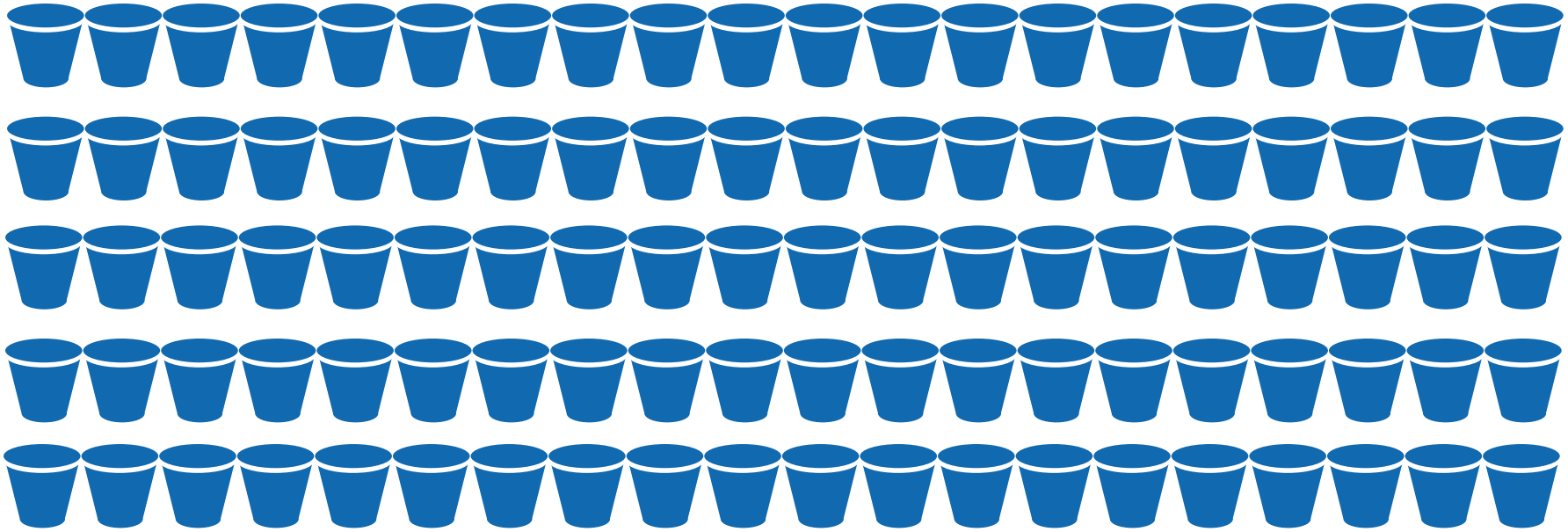


# Scalability



Max. 5 TB

# Scalability



100/account

(more upon request)

## Durability

# 99.999999999%

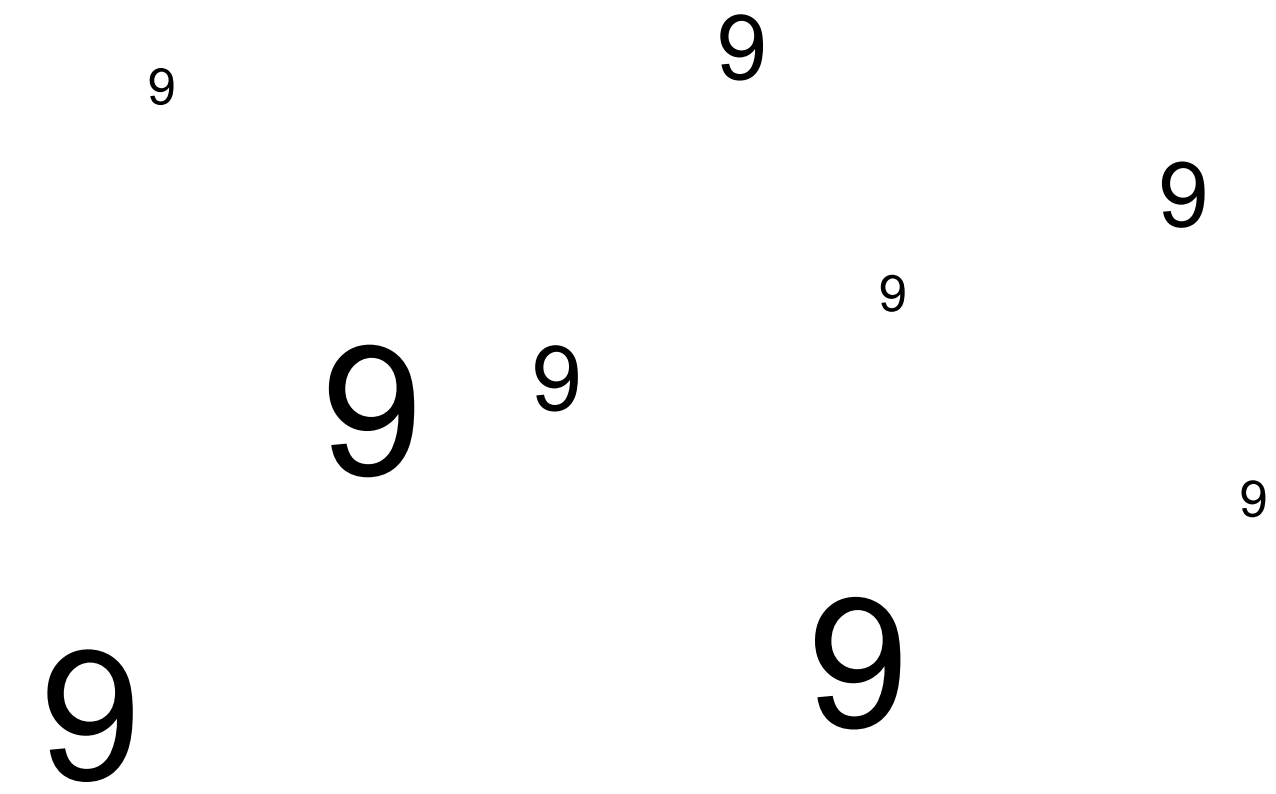
Loss of 1 in $10^{11}$ objects in a year

# Availability

99.99%

Down 1h / year

# More about SLAs



# More about SLA

| SLA | Outage |
|---|---|
| 99% | 4 days/year |
| 99.9% | 9 hours/year |
| 99.99% | 53 minutes/year |
| 99.999% | 6 minutes/year |
| 99.9999% | 32 seconds/year |
| 99.99999% | 4 seconds/year |

# More about SLA

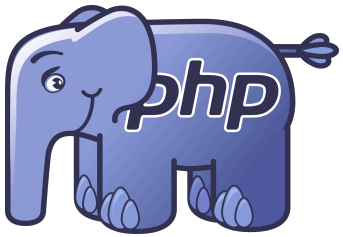
Amazon's approach:

Response time **< 10 ms**

in *99.9% of the cases*

(rather than **average** or **median**)

# API





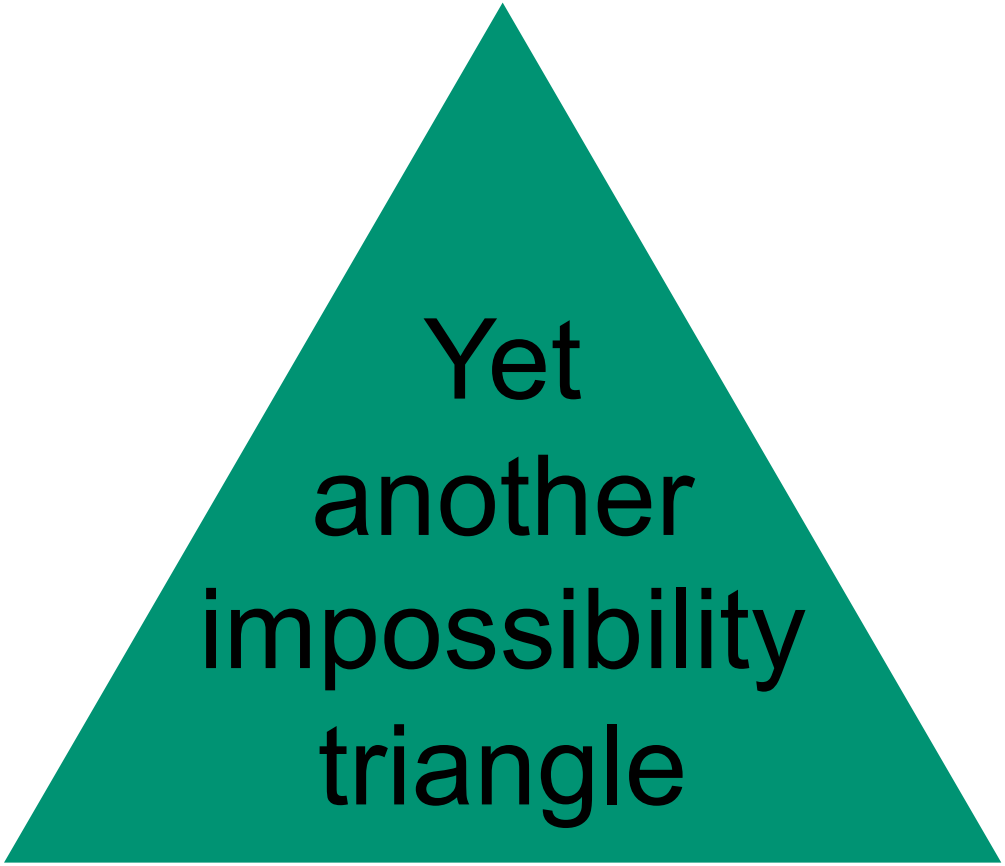
python™

**Driver (JDBC...)**

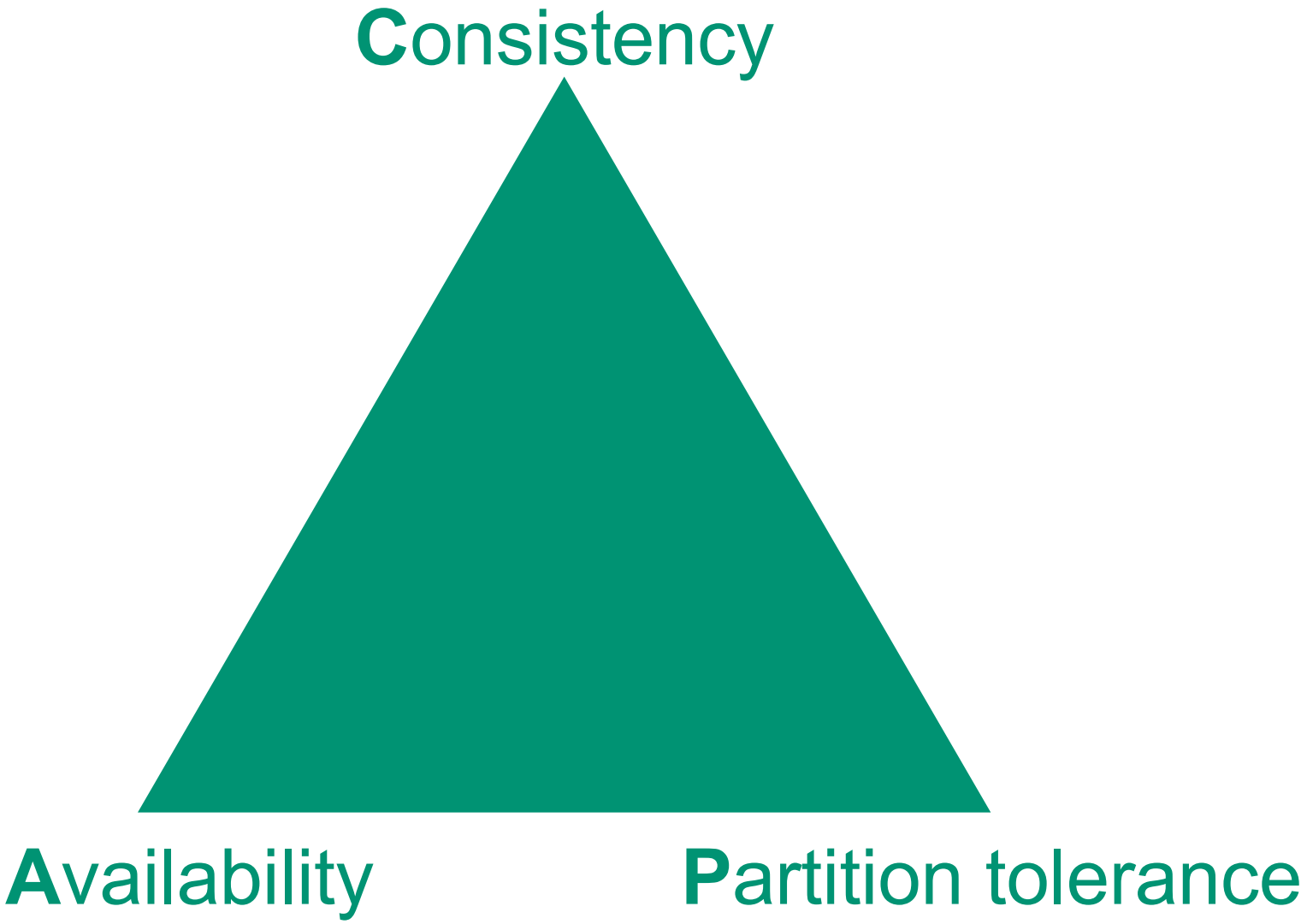
**SOAP**

**REST**

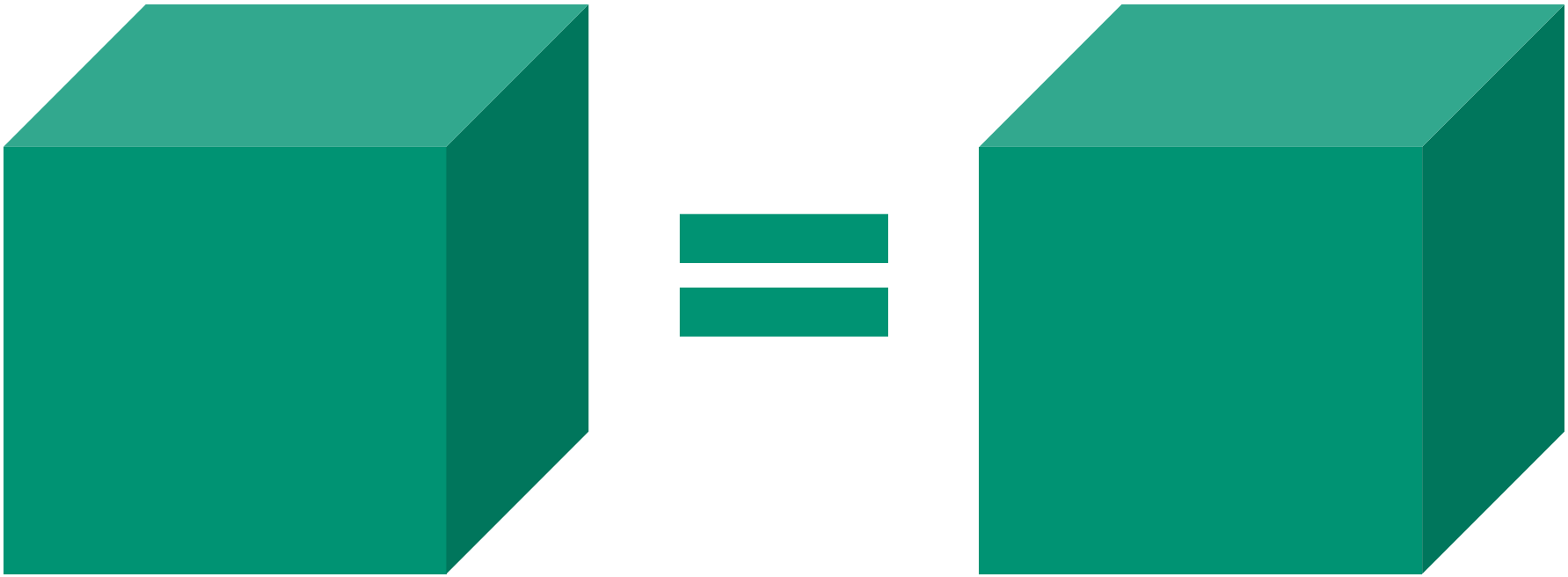
# The new era: the CAP theorem



# The new era: the CAP theorem

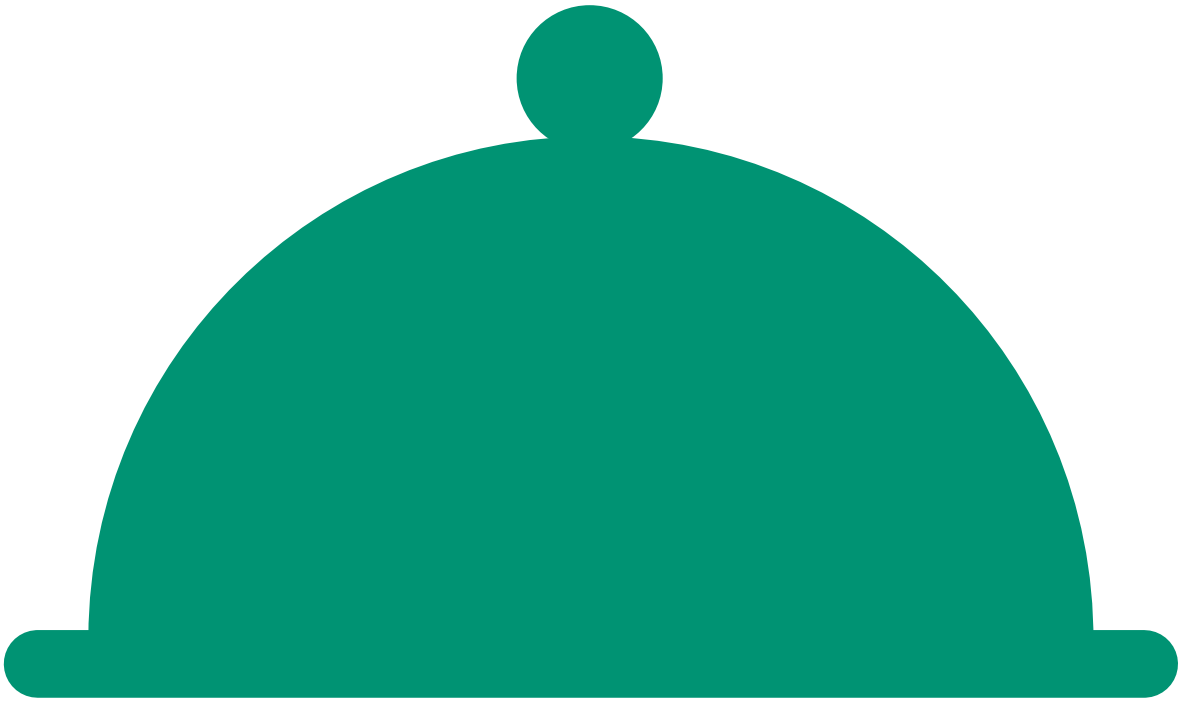


# (Atomic) Consistency



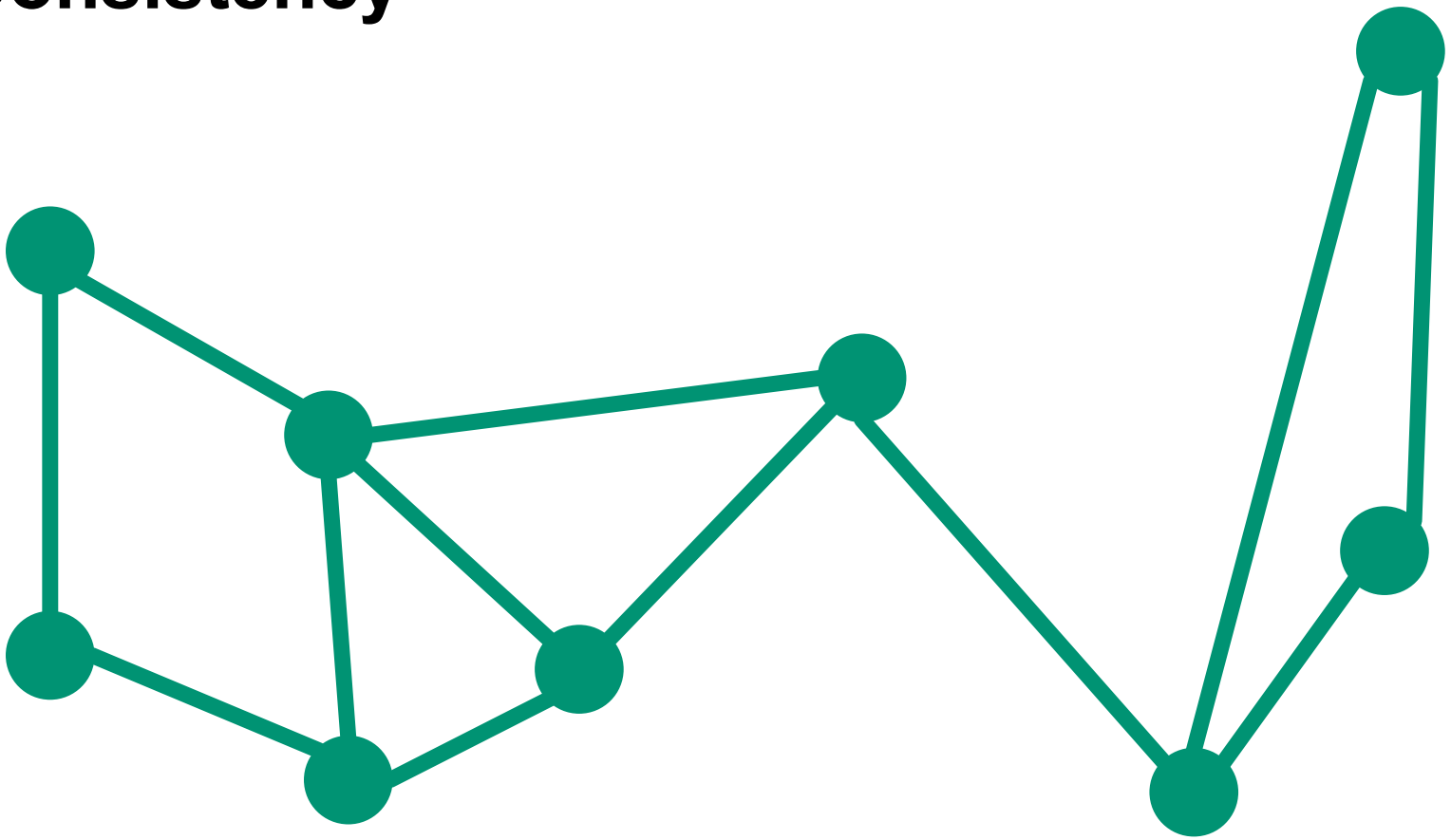
All nodes see the same data.

# Availability

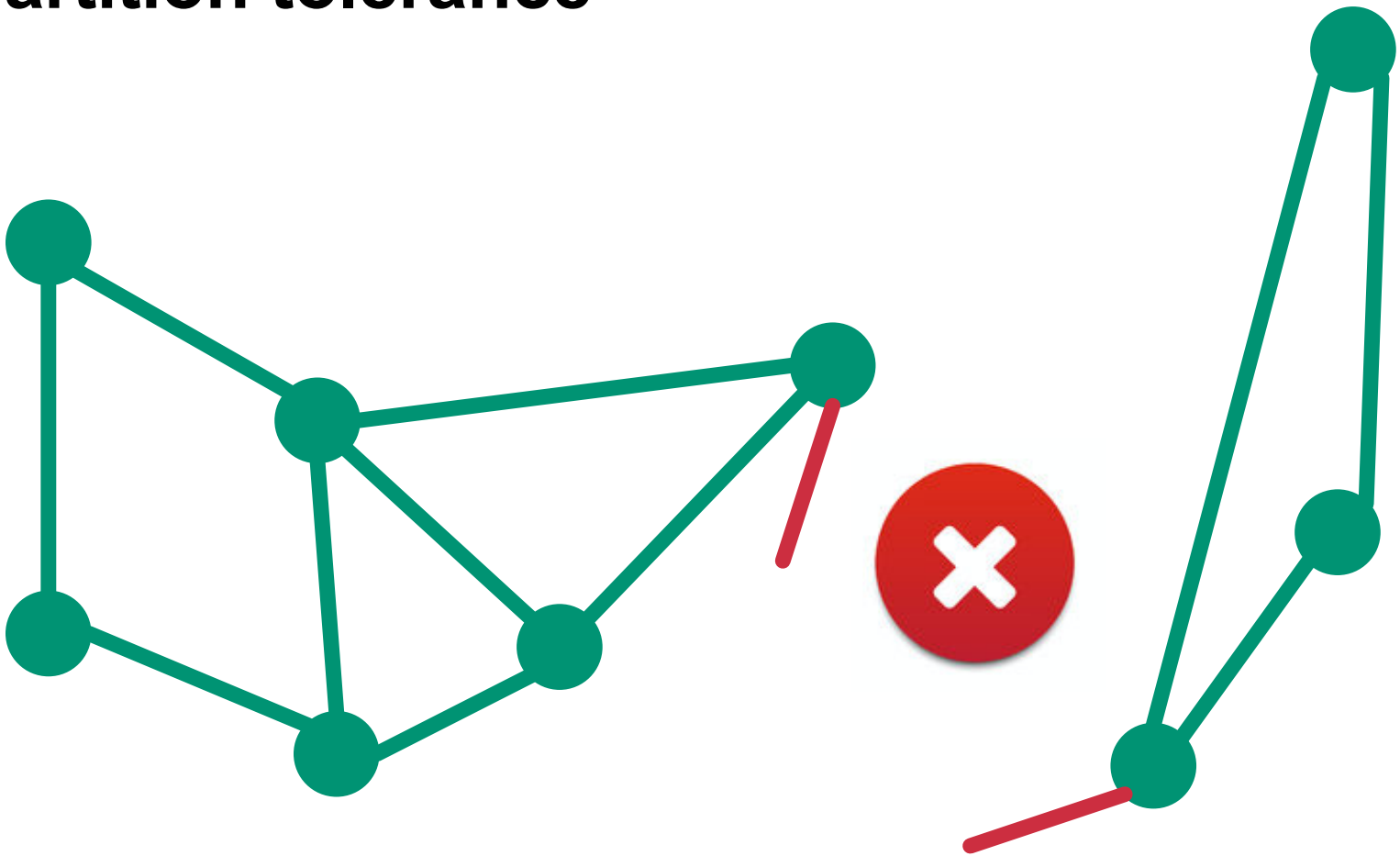


It is possible to query the database at all times.

# Consistency



## Partition tolerance



The database continues to function even if the network gets partitioned.

# REST APIs

# HTTP protocol

Sir Tim Berners-Lee

| Version | RFC |
|---|---|
| 1.0 | RFC 2616 |
| 1.1 | RFC 7230 |
| 2.0 | RFC 7540 |

# Resources

## Resource (URI)

http://www.ethz.ch/

http://www.mywebsite.ch/api/collection/foo/object/bar

urn:isbn:0123456789

mailto:sheldon.lee.cooper@ethz.ch

# Resources

## Resource (URL)

http://www.ethz.ch/

http://www.mywebsite.ch/api/collection/foo/object/bar

# Resources

## Resource (URN)

urn:isbn:0123456789

mailto:sheldon.lee.cooper@ethz.ch

# Resources

## Resource (URI)

http://www.mywebsite.ch/api/collection/foo/object/bar?id=foobar#head

# Resources

## Resource (URI)

**http**://www.mywebsite.ch/api/collection/foo/object/bar?id=foobar#head

scheme

# Resources

## Resource (URI)

http://**www.mywebsite.ch**/api/collection/foo/object/bar?id=foobar#head

authority

# Resources

## Resource (URI)

http://www.mywebsite.ch**/api/collection/foo/object/bar**?id=foobar#head

path

# Resources

## Resource (URI)

http://www.mywebsite.ch/api/collection/foo/object/bar**?id=foobar**#head

query

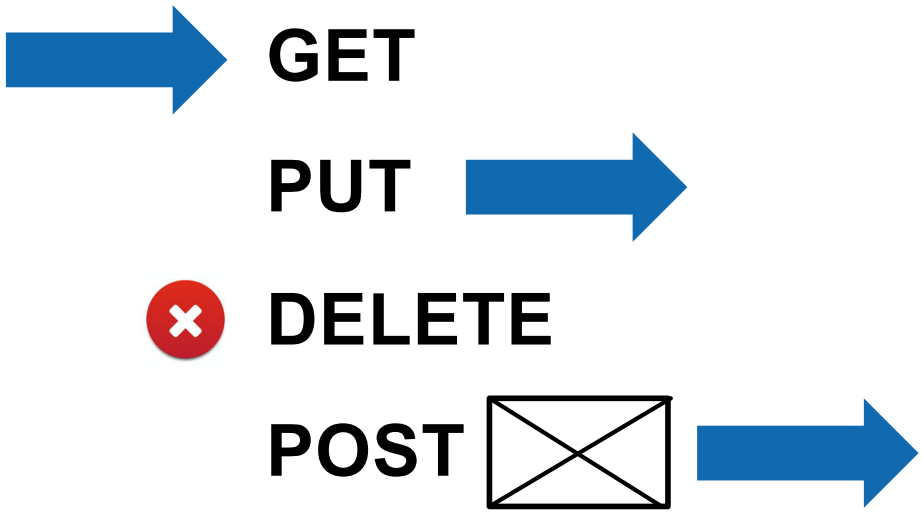
# Resources

## Resource (URI)

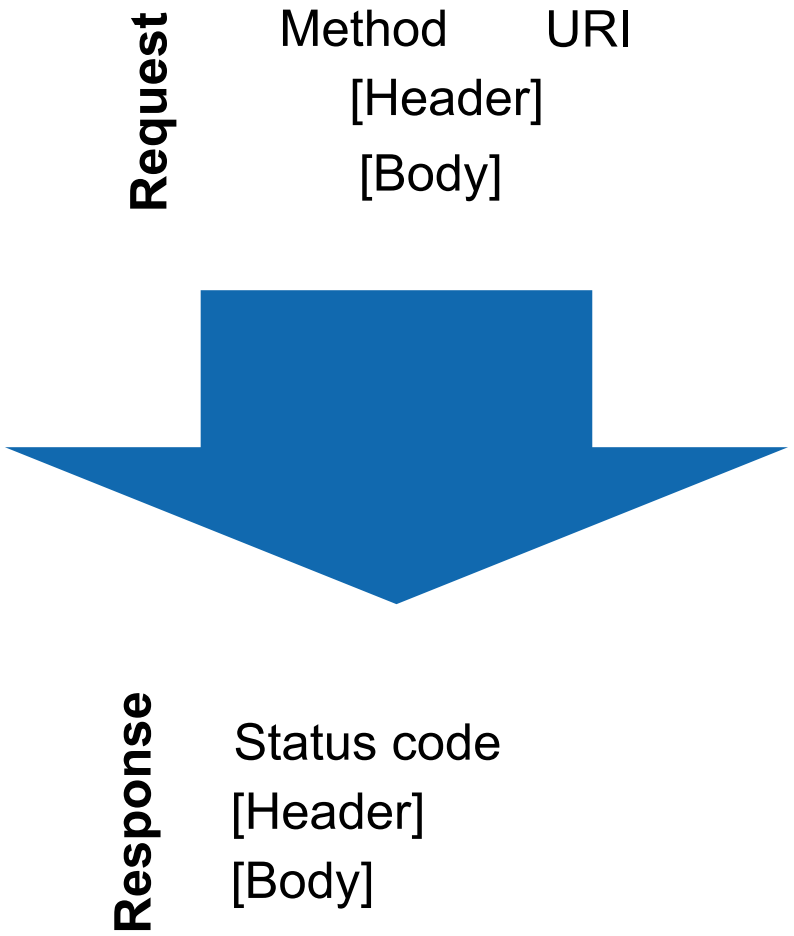
http://www.mywebsite.ch/api/collection/foo/object/bar?id=foobar**#head**

fragment

# HTTP Methods



# HTTP Protocol



# Example

```
GET /index.html HTTP/1.1
Host: www.example.com
```

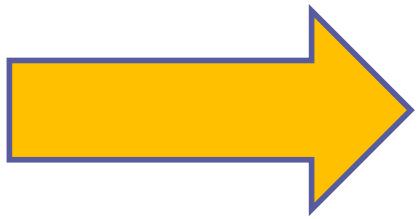
```
HTTP/1.1 200 OK
Date: Tue, 25 Sep 2018 09:48:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html> <head> <title>An Example Page</title>
</head> <body> Hello World, this is a very simple
HTML document. </body> </html>
```

Source: Wikipedia

# GET

(Side-effect free)

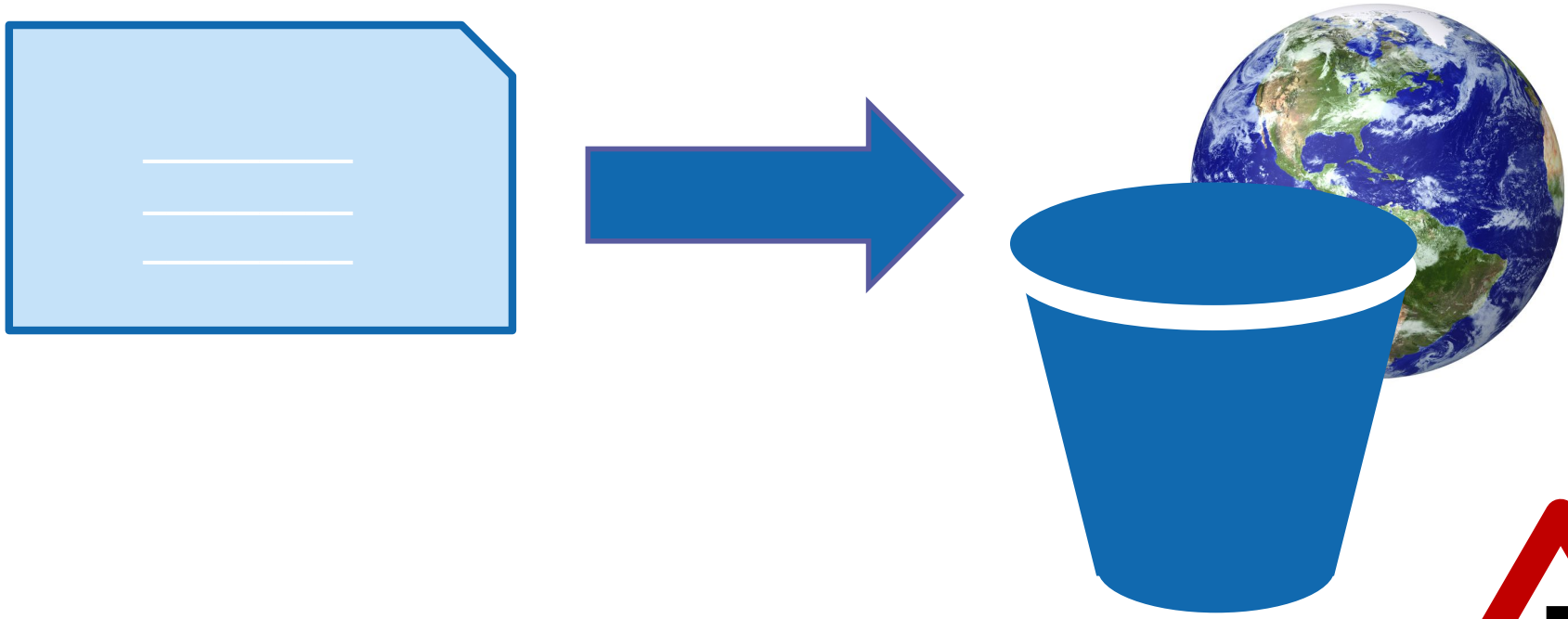
# PUT



(Idempotent)

# DELETE

# POST



Most generic: side effects

# REST with S3: Buckets

`http://bucket.s3.amazonaws.com`

# REST with S3: Objects

`http://bucket.s3.amazonaws.com/object-name`

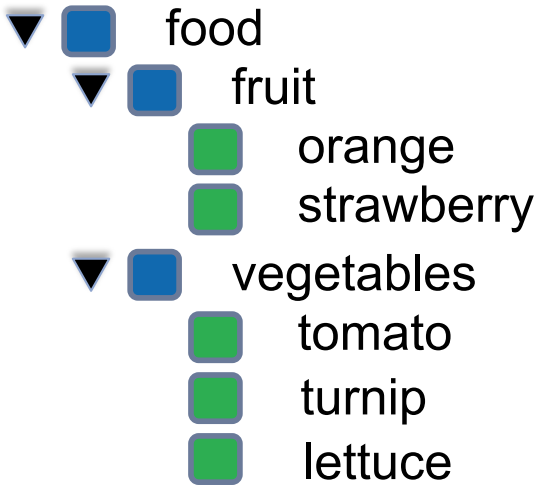
# S3 REST API

PUT Bucket
DELETE Bucket
GET Bucket

PUT Object
DELETE Object
GET Object

# Folders: is S3 a file system?

Logical
(Browsing)



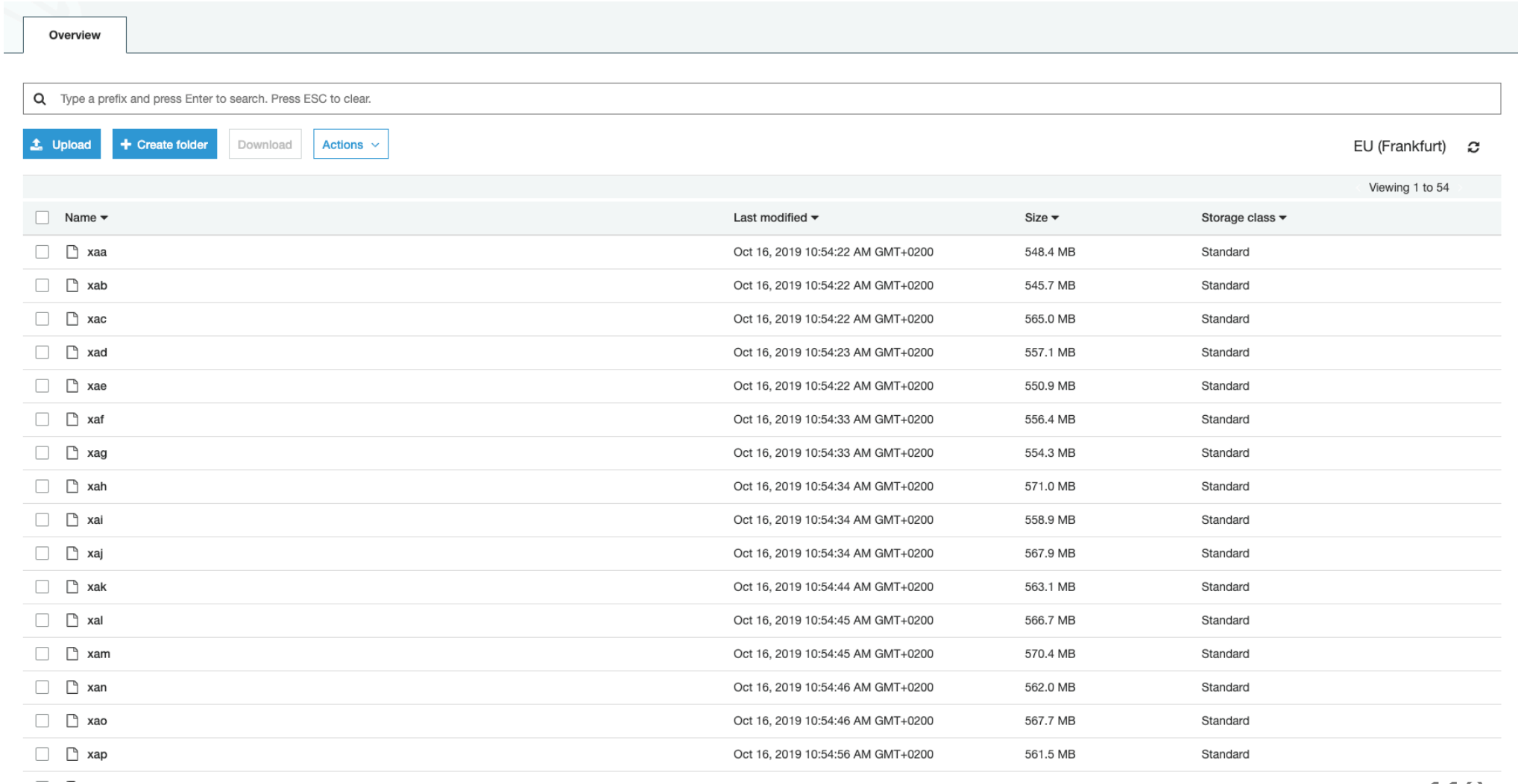

Physical
(Object keys)

/food/fruits/orange
/food/fruits/strawberry
/food/vegetables/tomato
/food/vegetables/turnip
/food/vegetables/lettuce

# Static website hosting

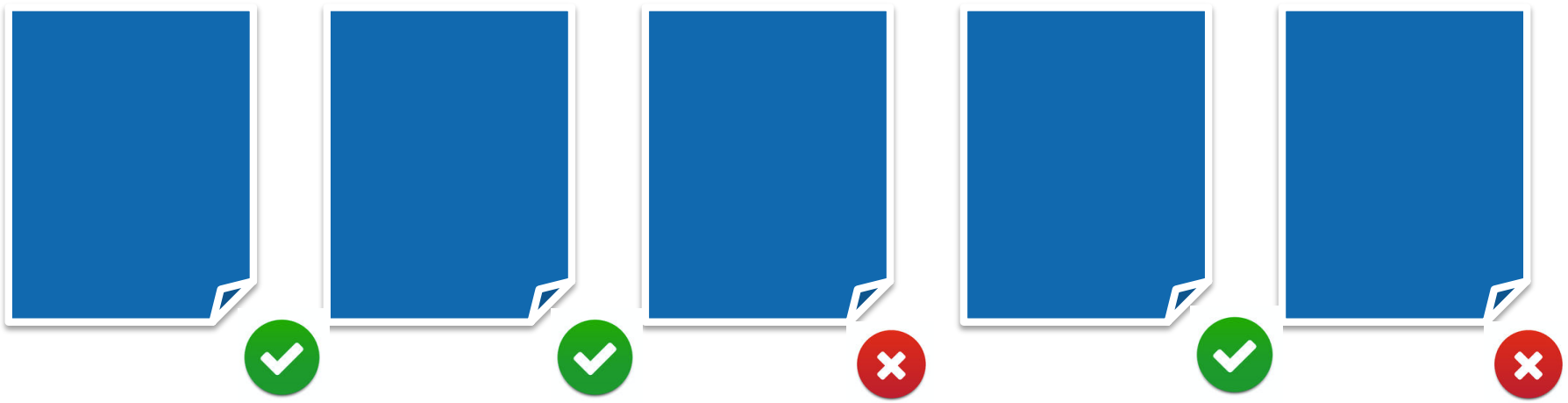
http://<*bucket-name*>.s3-website-us-east-1.amazonaws.com/

# Dataset hosting



# More on Storage

## Replication



# Fault tolerance

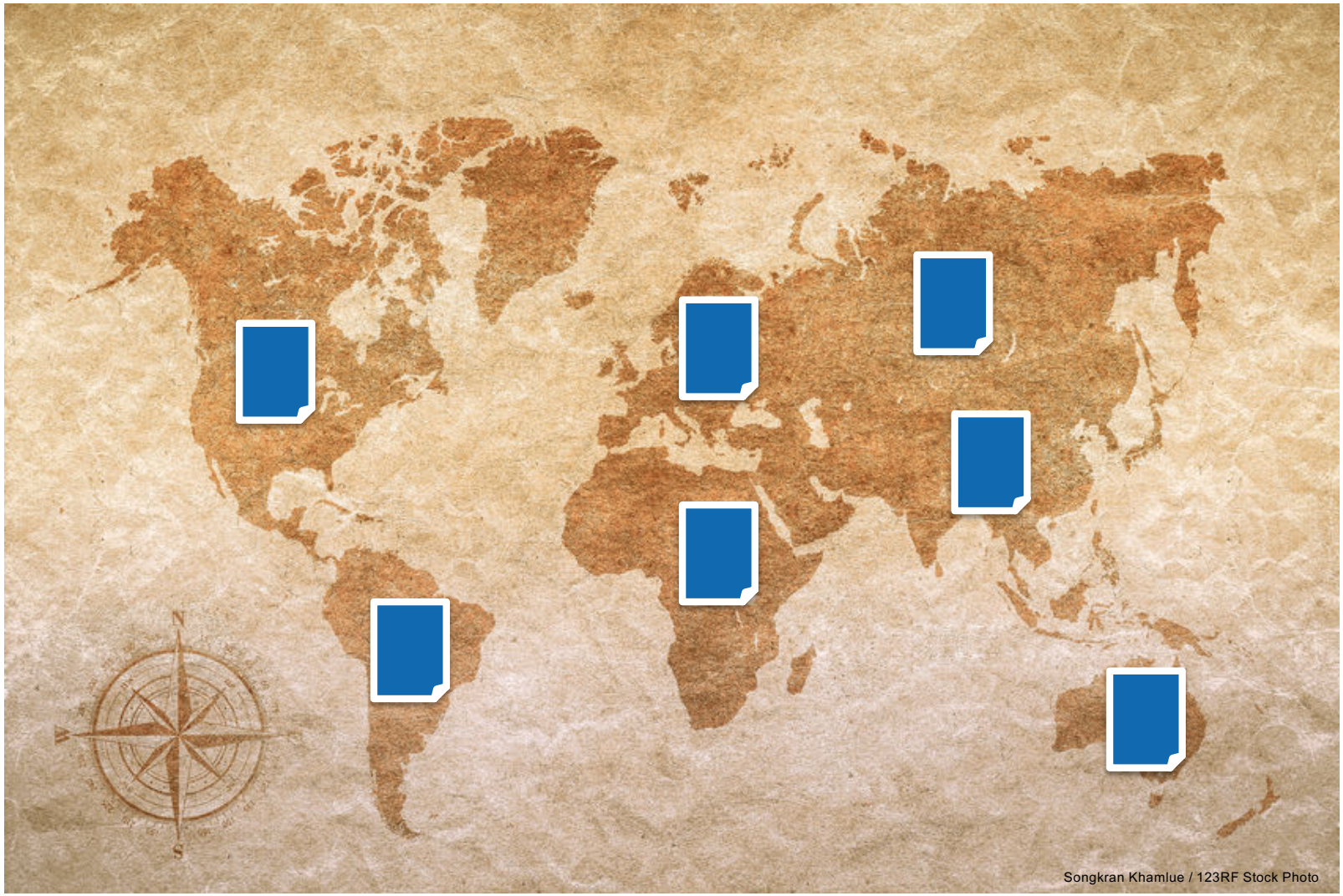
# Faults

Local (node failure)

versus

Regional (natural catastrophe)

# Regions

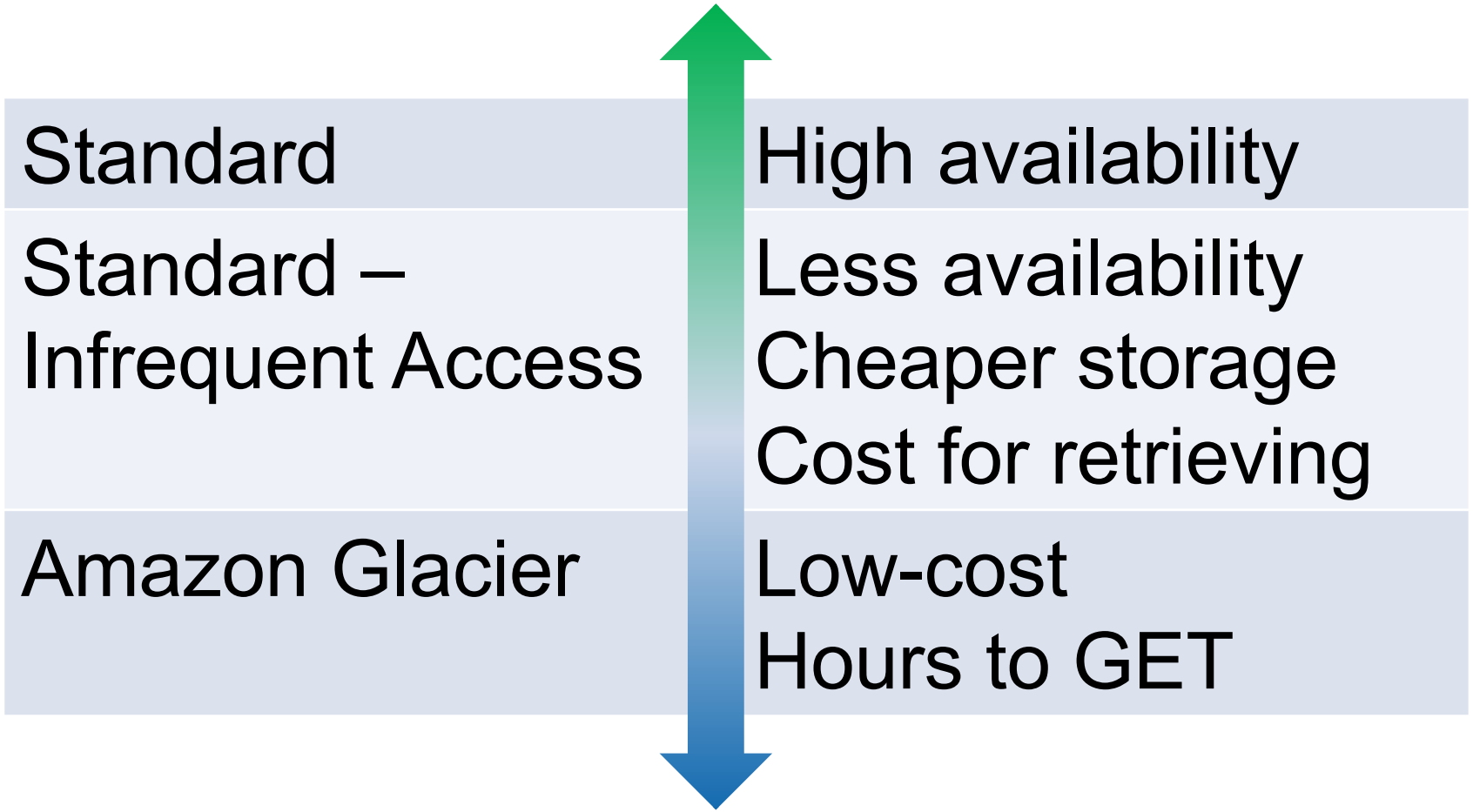


# Regions

# Regions


1. Optimize latency
2. Resilient to natural catastrophes
Songkran Khamlue / 123RF Stock Photo

# Storage Class

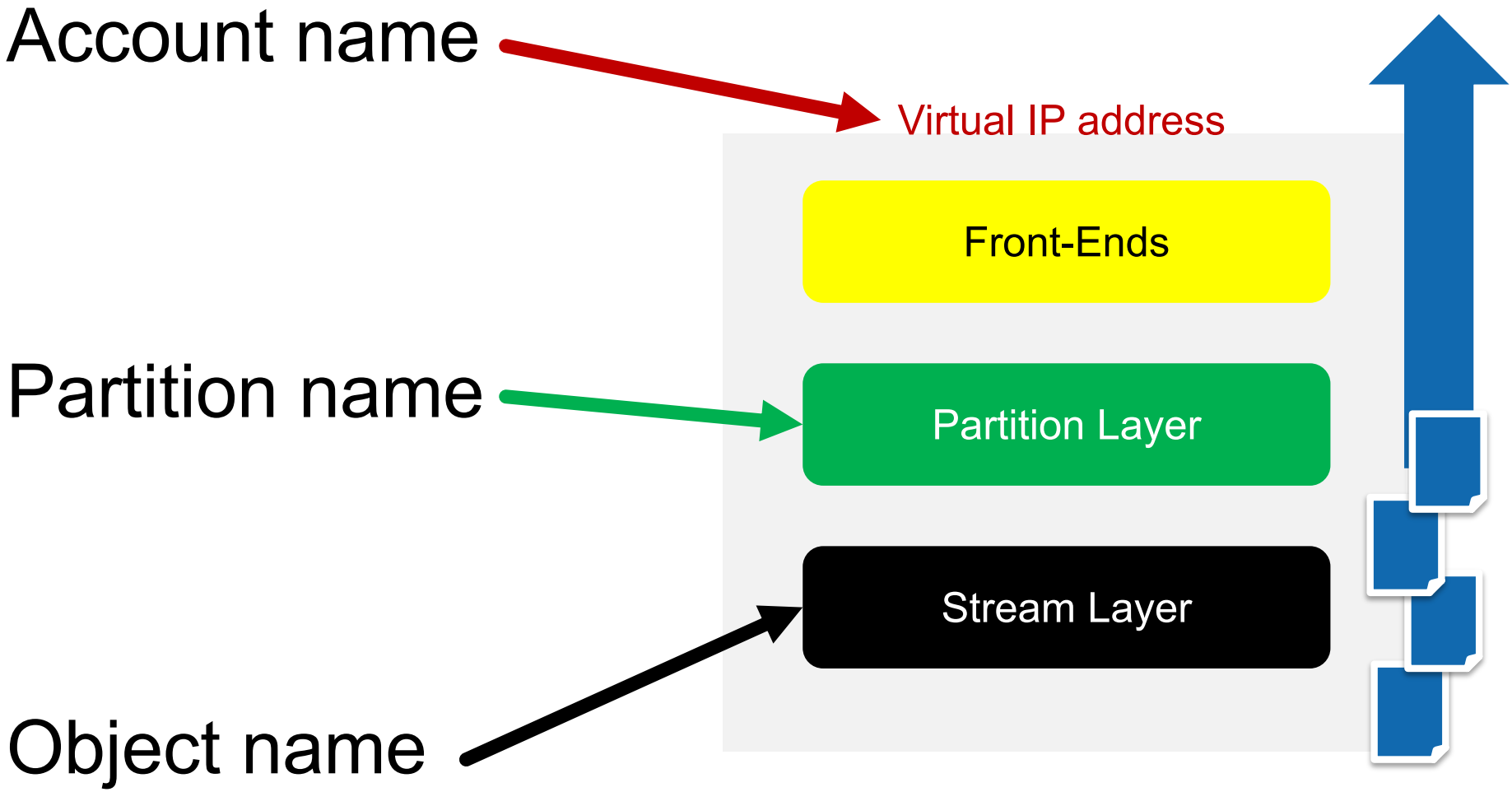



# Azure Blob Storage

# Overall comparison Azure vs. S3

| | S3 | Azure |
|---|---|---|
| Object ID | Bucket + Object | Account + Container + Blob |
| Object API | Blackbox | Block/Append/Page |
| Limit | 5 TB | 4.78 TB (block)<br>195 GB (append)<br>8 TB (page) |

# Azure Architecture: Storage Stamp



# Azure Architecture: One storage stamp

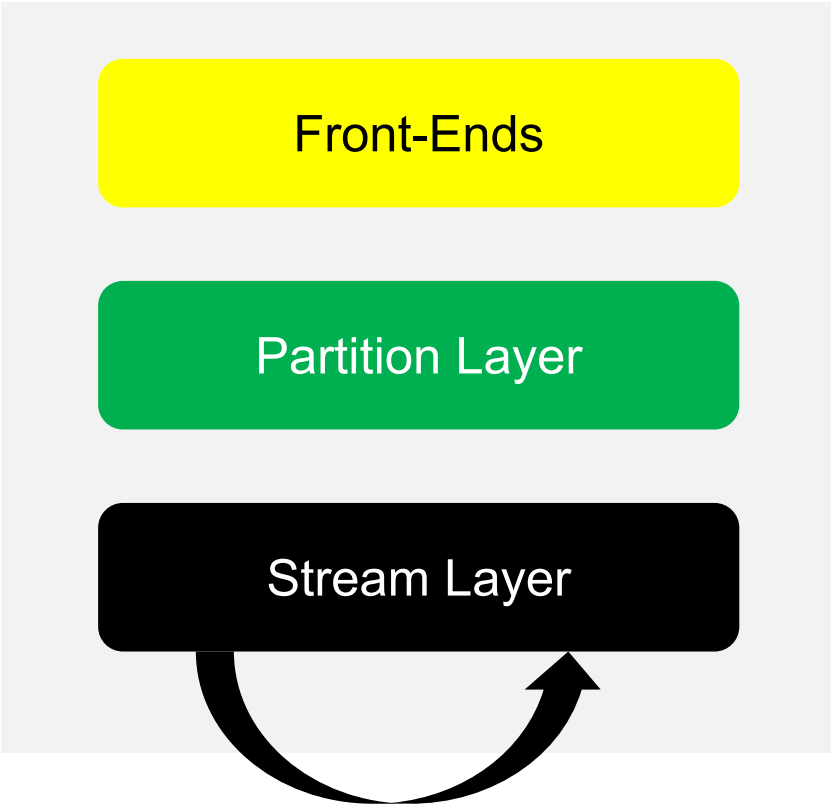
10-20 racks * 18 storage nodes/rack (30PB)

# Azure Architecture: Keep some buffer

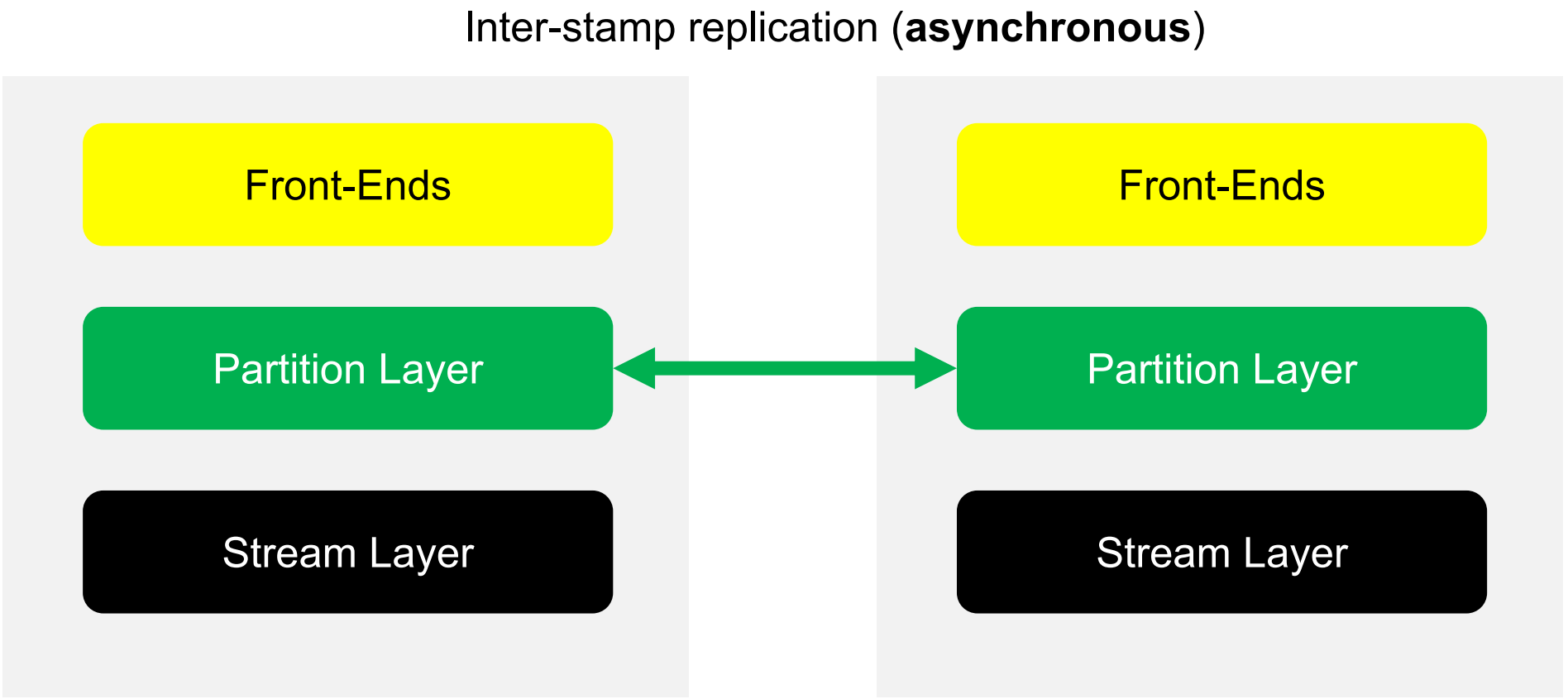
kept below 70/80% storage capacity

# Storage Replication

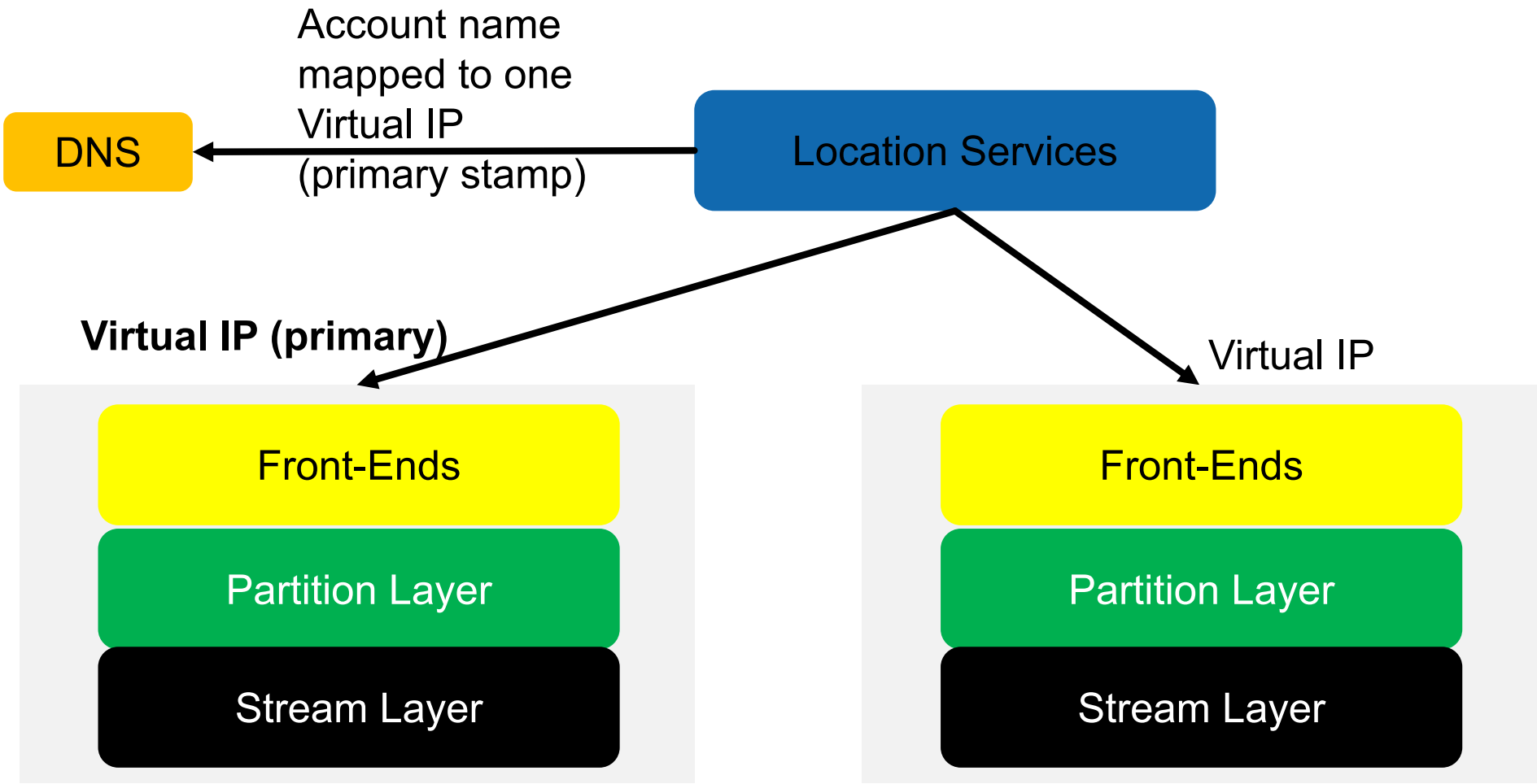
Intra-stamp replication (**synchronous**)



# Storage Replication



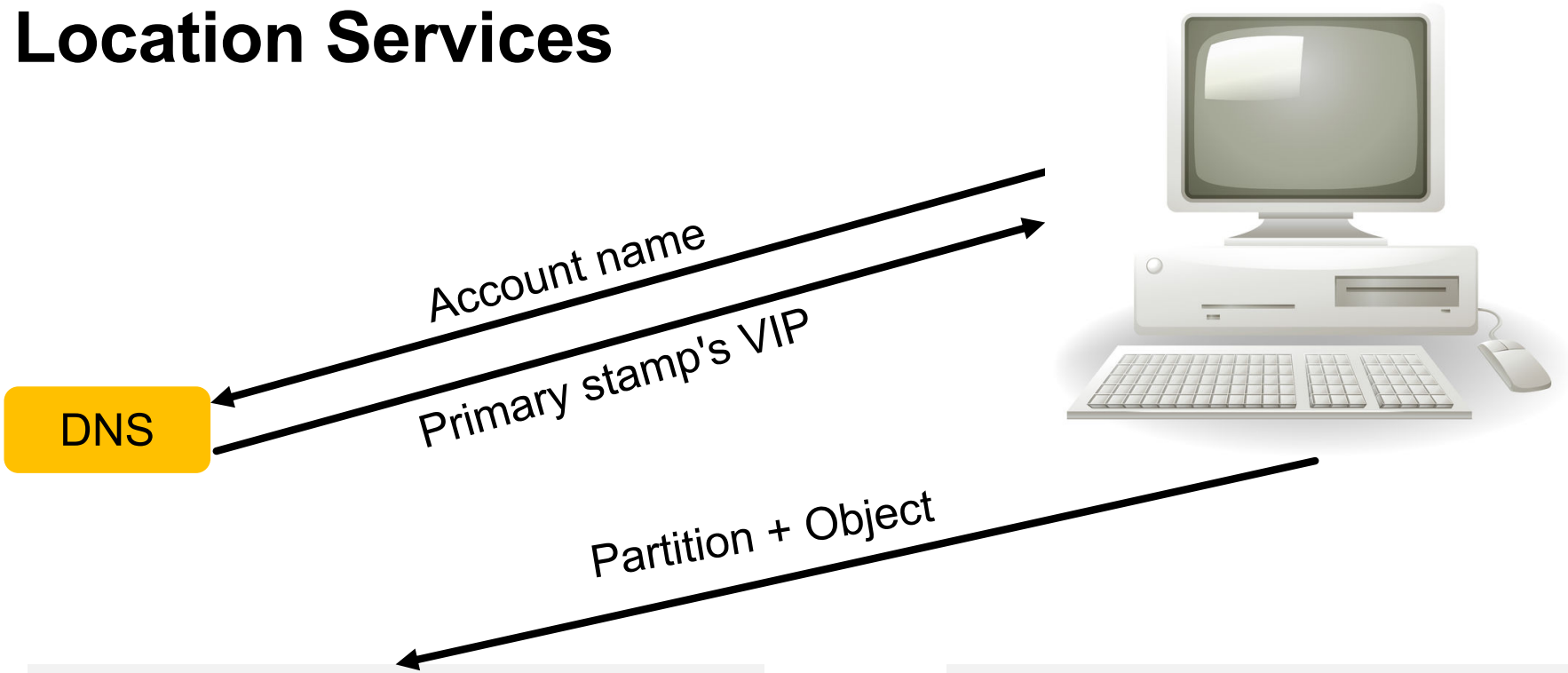
# Location Services



# Location Services

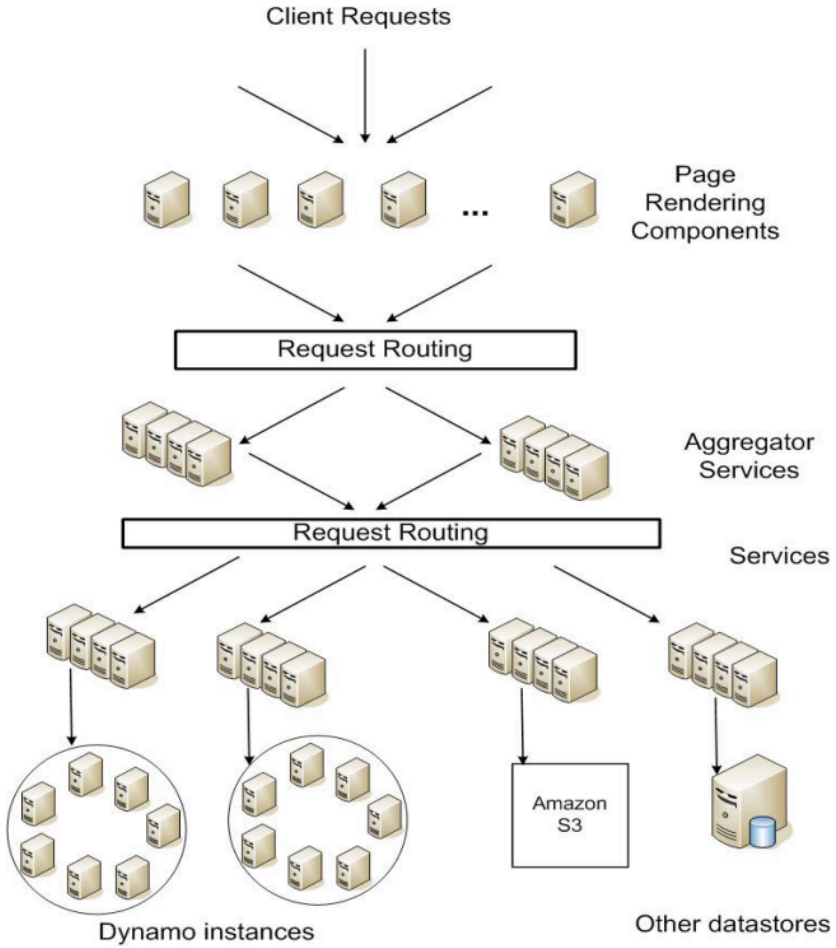

North America
Europe
Asia

# Location Services

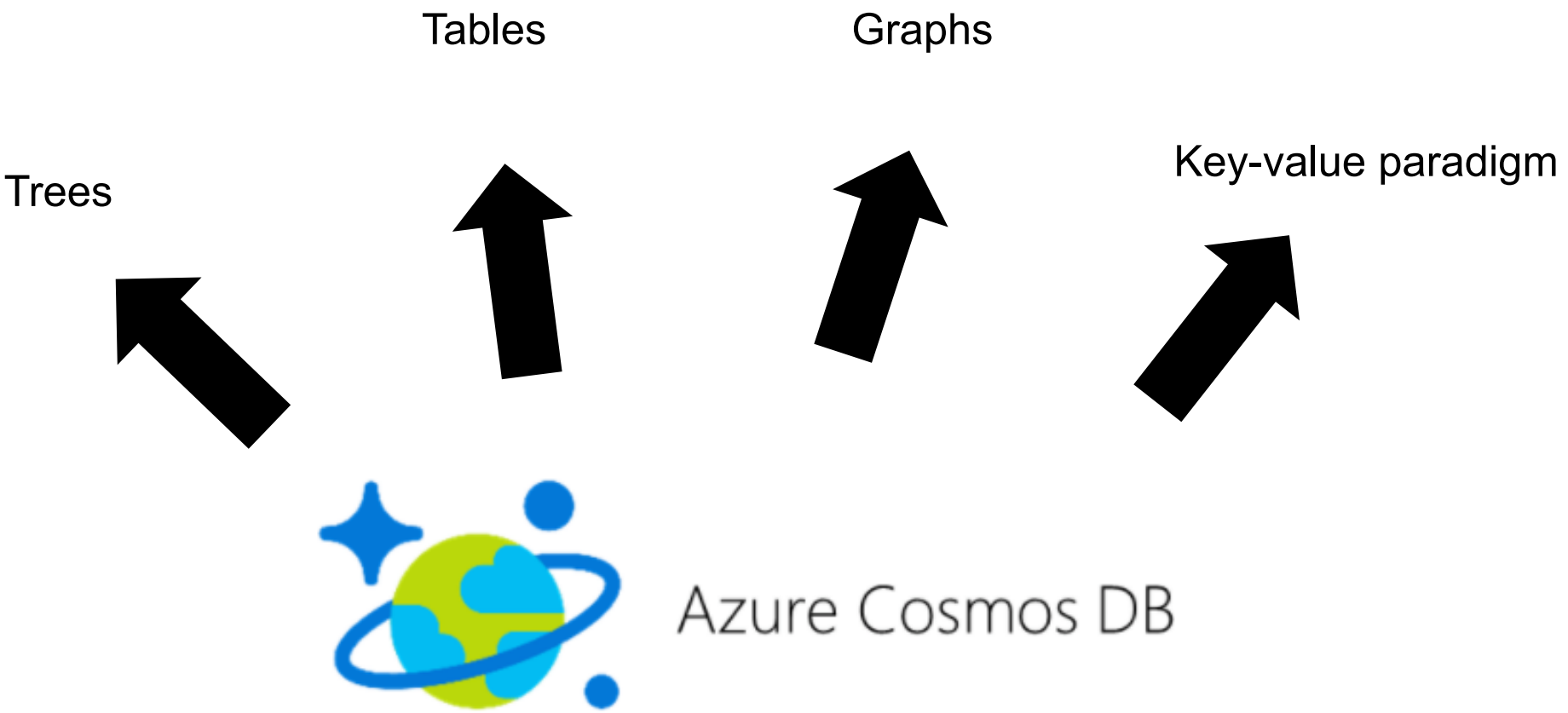


# Last words

# Amazon mindset



# Azure mindset



## Take away messages: how to scale out?

- Simplify the model!
- Buy cheap hardware!
- Remove schemas!