CLOUD COMPUTING SYSTEMS

Lecture 2

Nuno Preguiça

(nuno.preguica_at_fct.unl.pt)

OUTLINE

Web applications

Blob storage service

OUTLINE

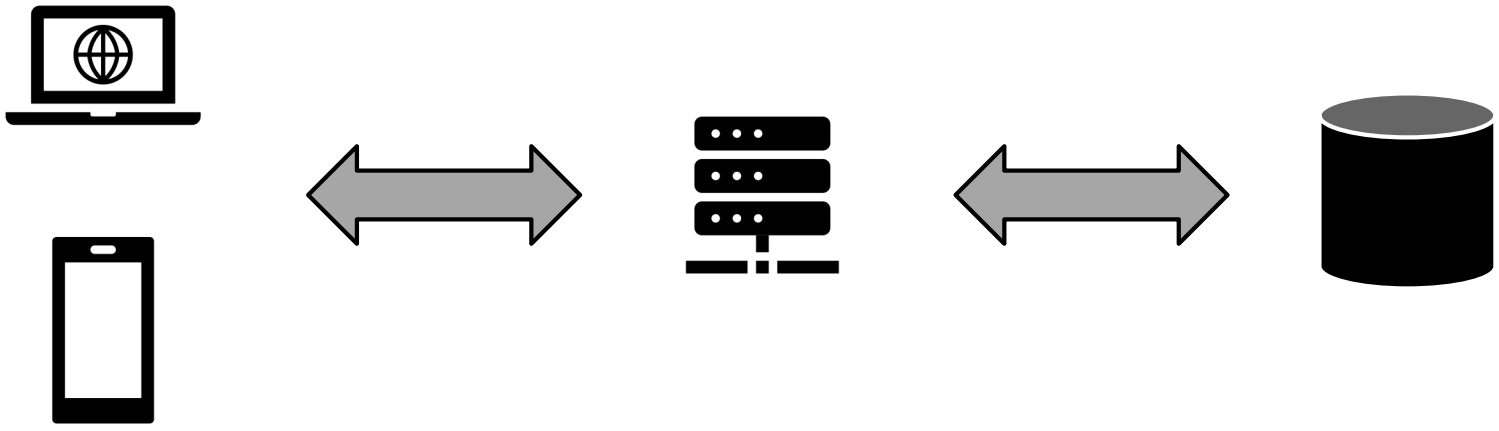
Web applications

Blob storage service

WEB APPLICATIONS

Web and mobile applications are commonly deployed using a three-tier architecture

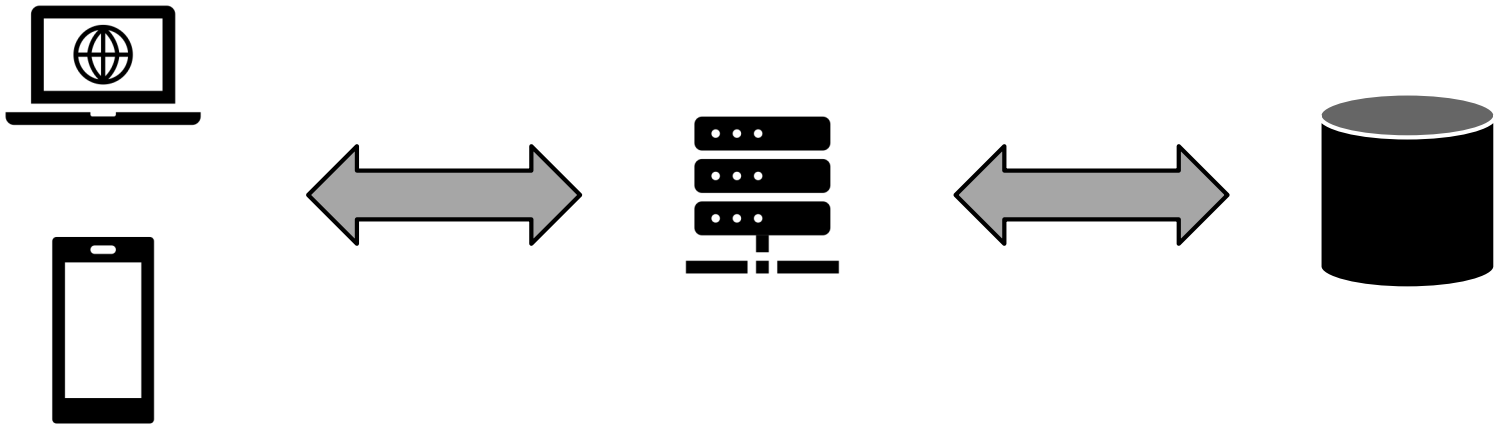
- Client tier (aka presentation)
- Application tier (aka business)
- Data tier



WEB APPLICATIONS: CLIENT TIER

Client tier is composed by:

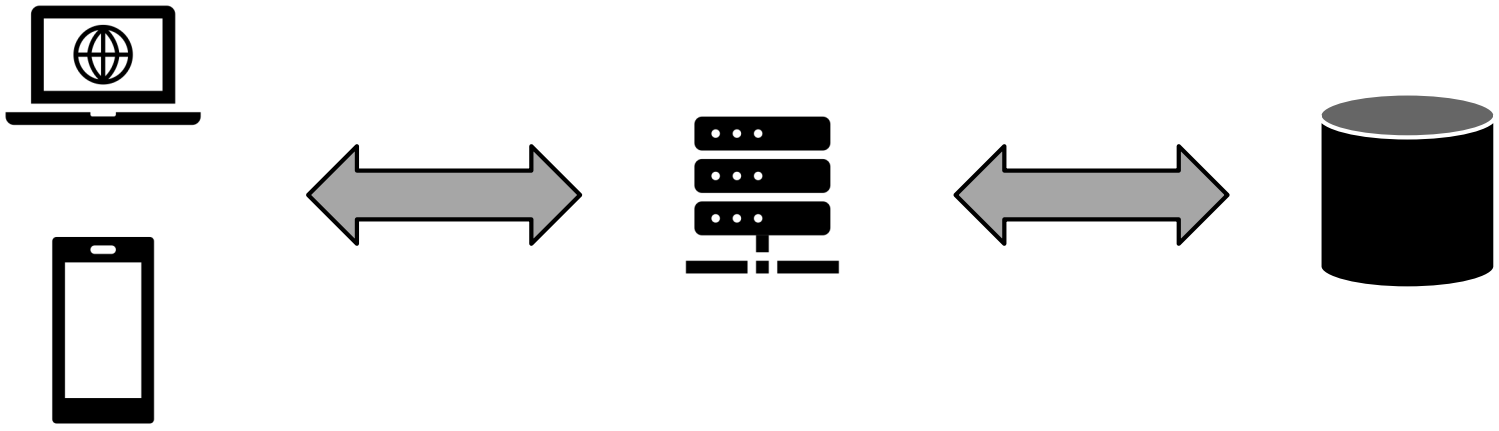
- Web clients: HTML + Javascript
- Mobile apps: Android, iOS, etc.



WEB APPLICATIONS: APPLICATION TIER

Application tier is composed by:

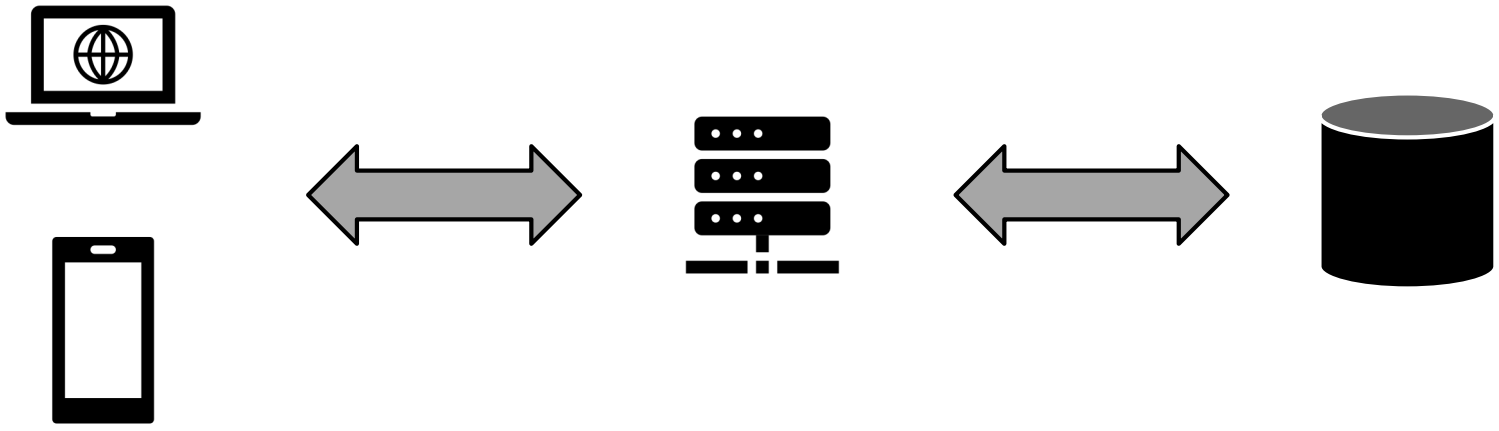
- REST servers
- Static and dynamic web pages
- Popular application servers
  - E.g.: Tomcat, Wildfly, ASP.NET, etc.



WEB APPLICATIONS: DATA TIER

Data tier is composed by:

- Filesystems, BLOB stores, key-value stores, SQL databases, etc.
- Other services: caches, message queue, etc.

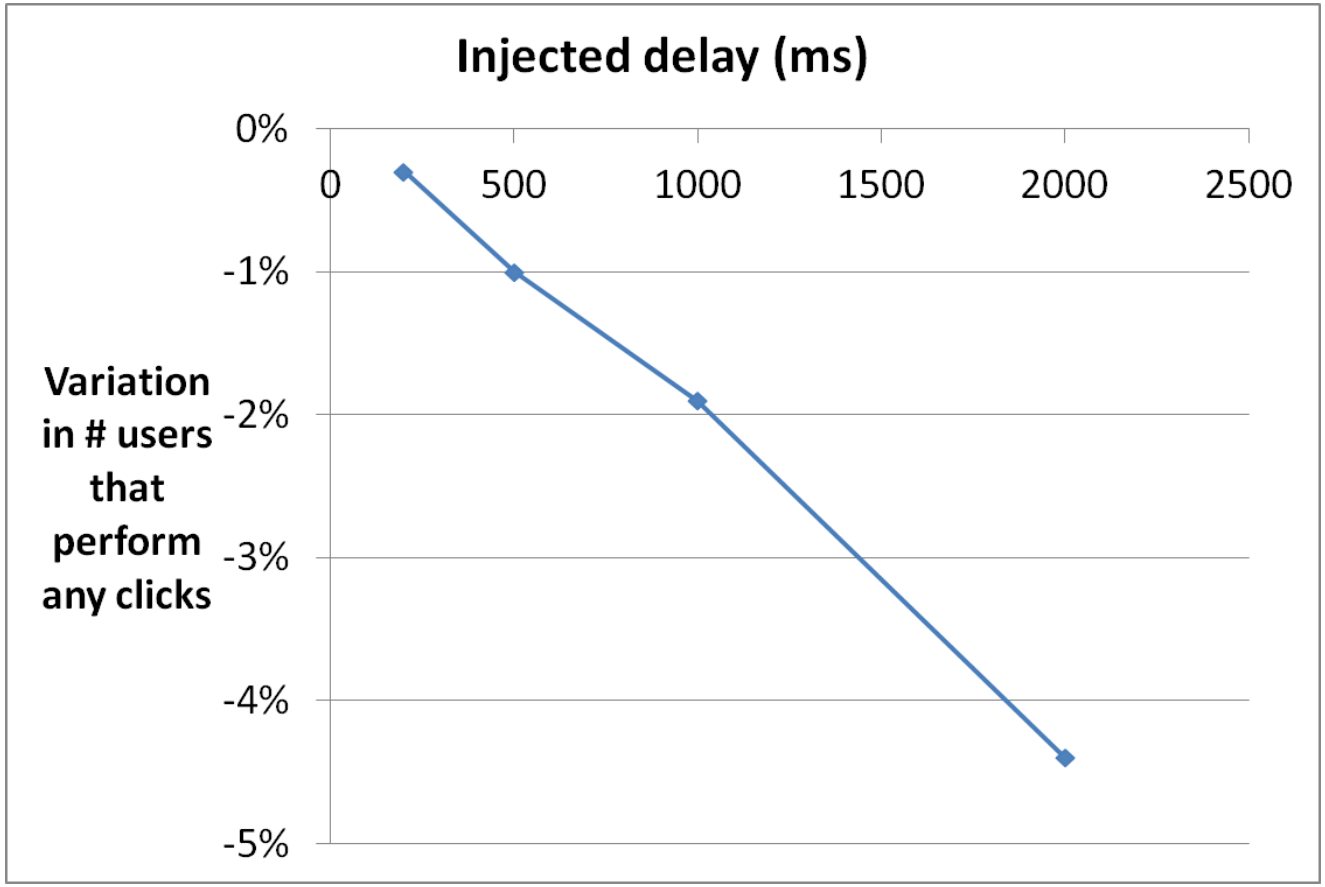


WEB APPLICATIONS: WHY MULTIPLE INSTANCES?

Reasons for having more than one application server instance

1. A single server might not be sufficient to handle all client requests;
2. A single server is a single point of failure: if it fails the service becomes unavailable;
3. A single server may be too far from some clients: latency too high.

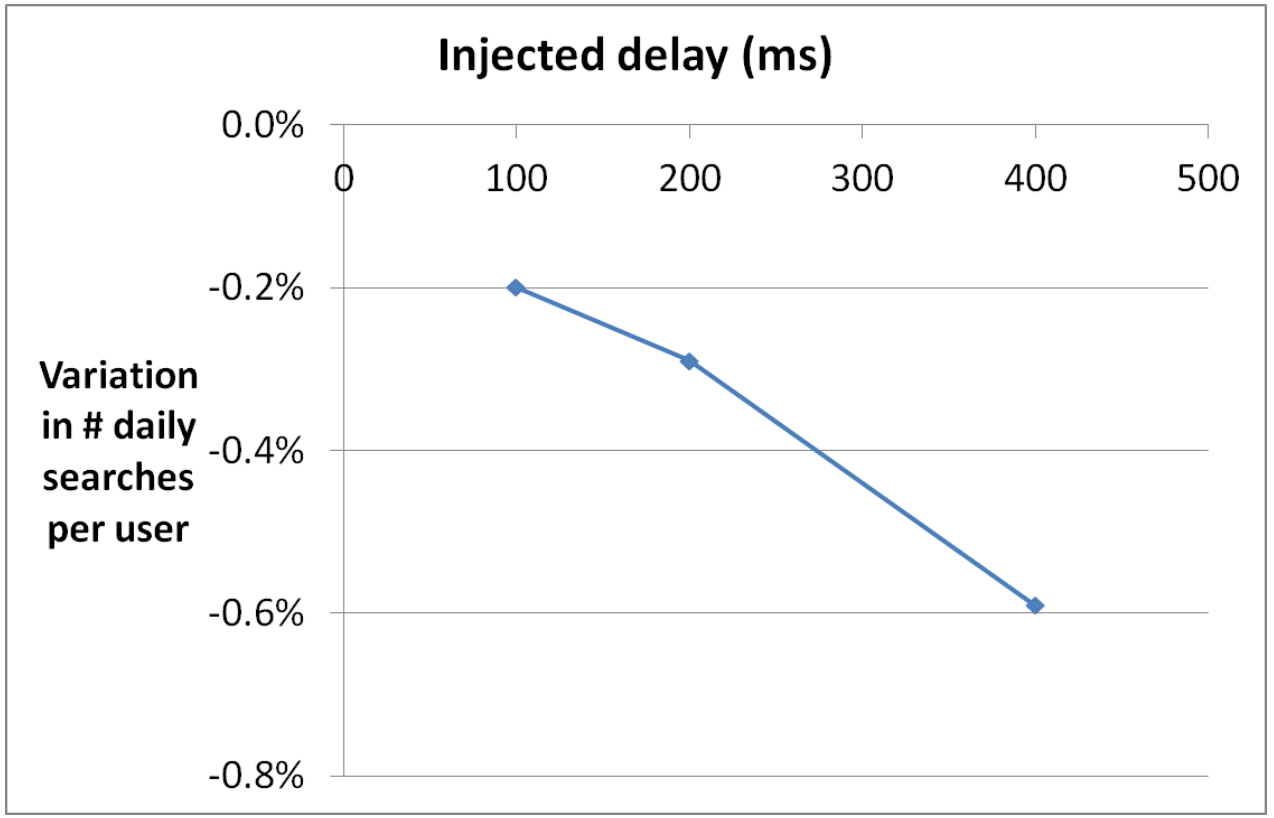
HIGHER LATENCY ⇒ UNHAPPY USERS



Microsoft Bing: 2-sec slowdown reduced number of users that perform any clicks on the page by 4.4%.

[source: E. Schurman and J. Brutlag, "Performance Related Changes and their User Impact". Talk at Velocity '09]

GOOGLE OBSERVED SIMILAR TRENDS



Similar impact in number of daily searches per user at Google.

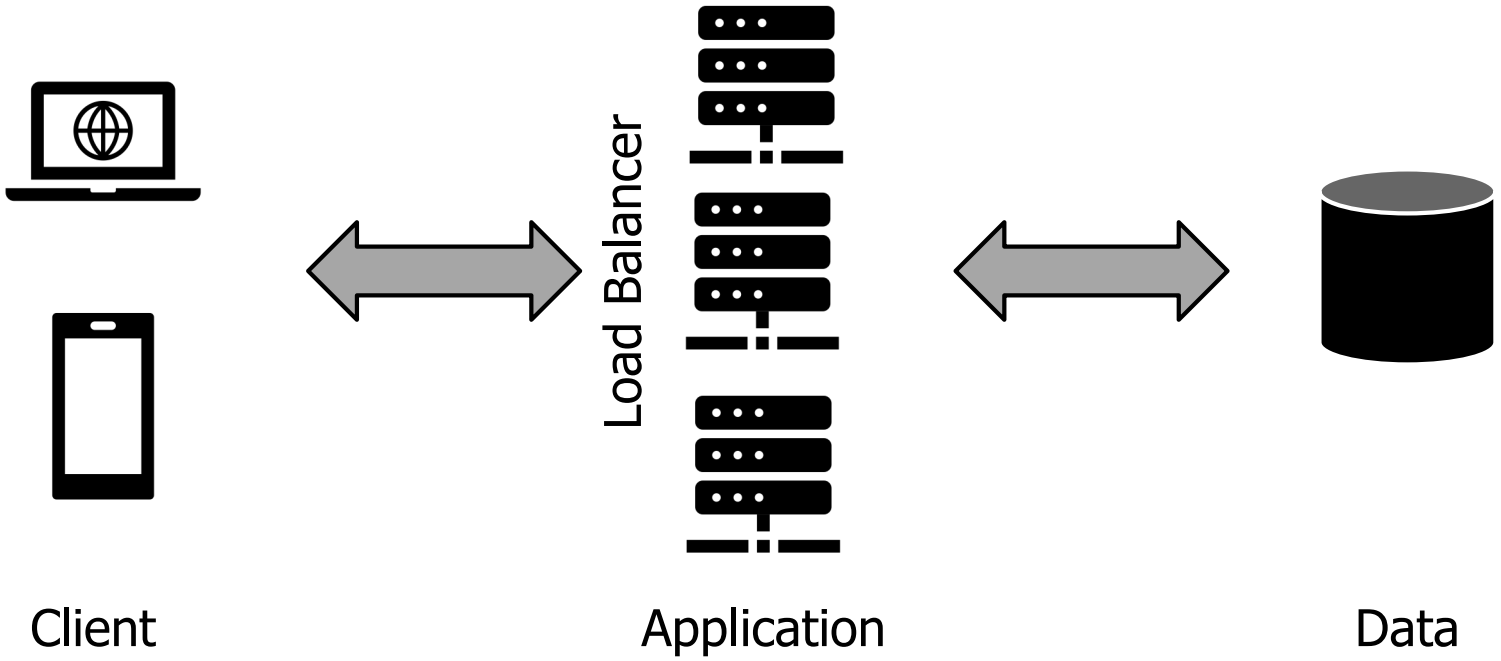
[source: E. Schurman and J. Brutlag, "Performance Related Changes and their User Impact". Talk at Velocity '09]

What simplifies having multiple instances?

Application server keeps no internal state.

- State maintained in the data tier.

Recent trend: have the application tier as "serverless" code (later on the course)



WEB APPLICATIONS: WHY MULTIPLE INSTANCES?

Reasons for having more than one application server instance

1. A single server might not be sufficient to handle all client requests;
2. A single server is a single point of failure: if it fails the service becomes unavailable;
3. A single server may be too far from some clients: latency too high.

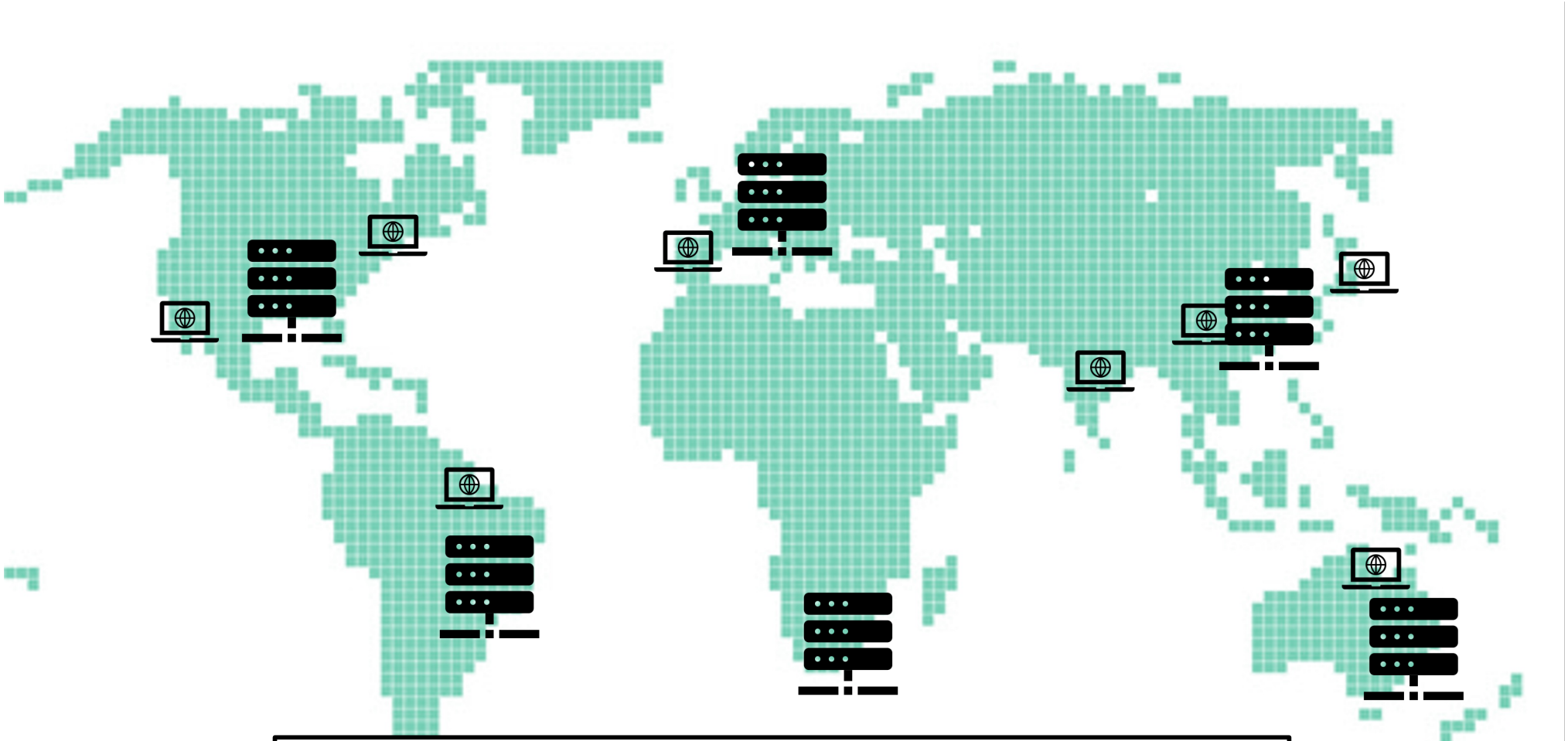
Reality check: having multiple instances of an application server in a single data center solves these problems?

WEB APPLICATIONS: WHY INSTANCES IN DIFFERENT LOCATIONS?

Reasons for having application server instances at **multiple locations**

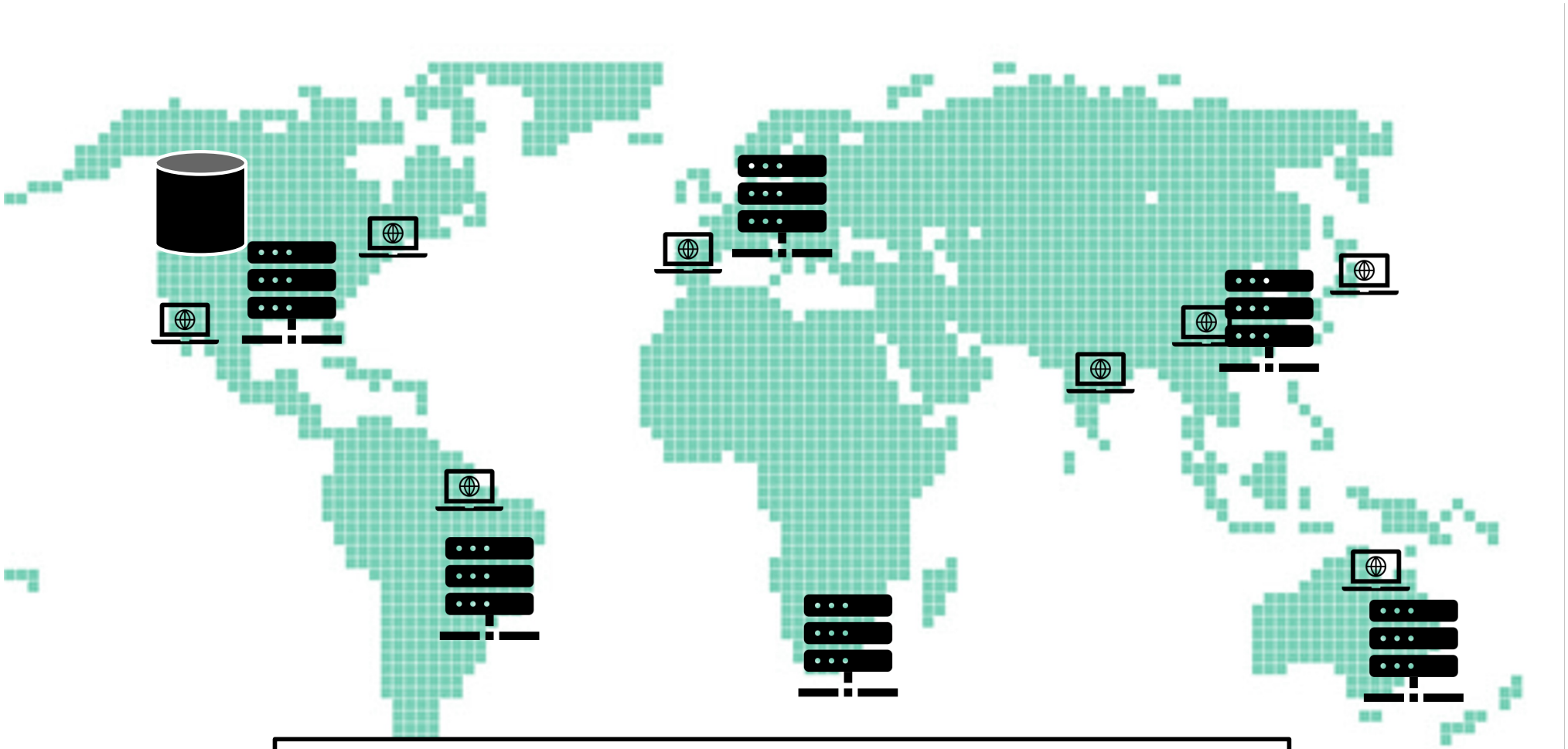
1. ~~A single server might not be sufficient to handle all client requests;~~
2. A single server is a single point of failure: if it fails the service becomes unavailable;
3. A single server may be too far from some clients: latency too high.

(DISTRIBUTED) WEB APPLICATION



What about the data tier?

(DISTRIBUTED) WEB APPLICATION



What about the data tier?
Can we put it in a single place?

WEB APPLICATIONS SERVICES

Azure

- Web app service

Google

- Google App Engine

AWS

- AWS Elastic Beanstalk

Azure App Service

Azure App Service is a PaaS for creating web apps, mobile back ends, and RESTful APIs.

Supports a wide range of languages and frameworks: ASP.NET, ASP.NET Core, Java, Ruby, Node.js, PHP, or Python.

Supports multiple OS and deployment using containers.

In Java supports Tomcat (and Spring integration).

REST APIs.

- Discussed in other courses (Distributed Systems, APC, CI, etc.)

J2EE.

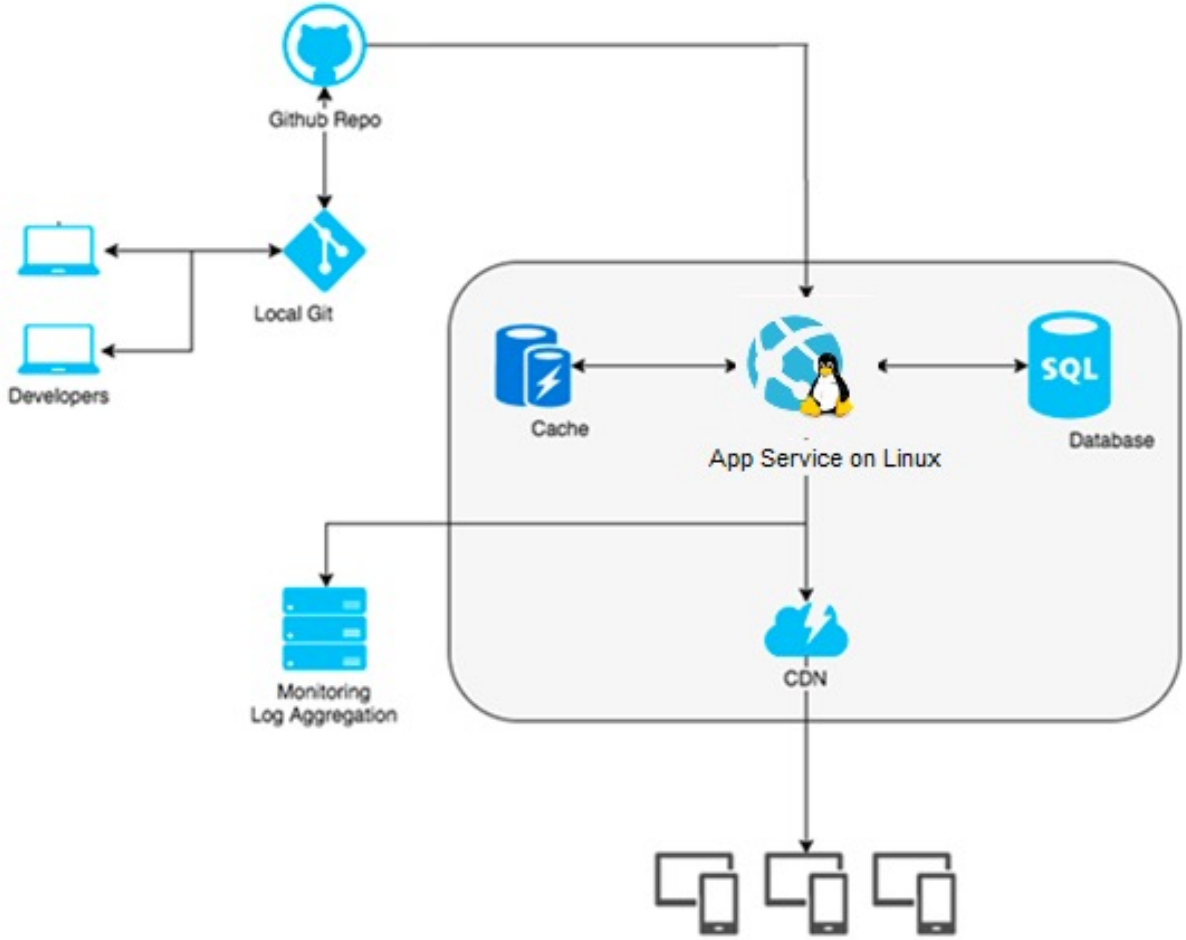
Azure App Service Architecture

App Service handles HTTP/REST requests.

Use database/storage services for durability.

Use cache services for improved performance in the servers.

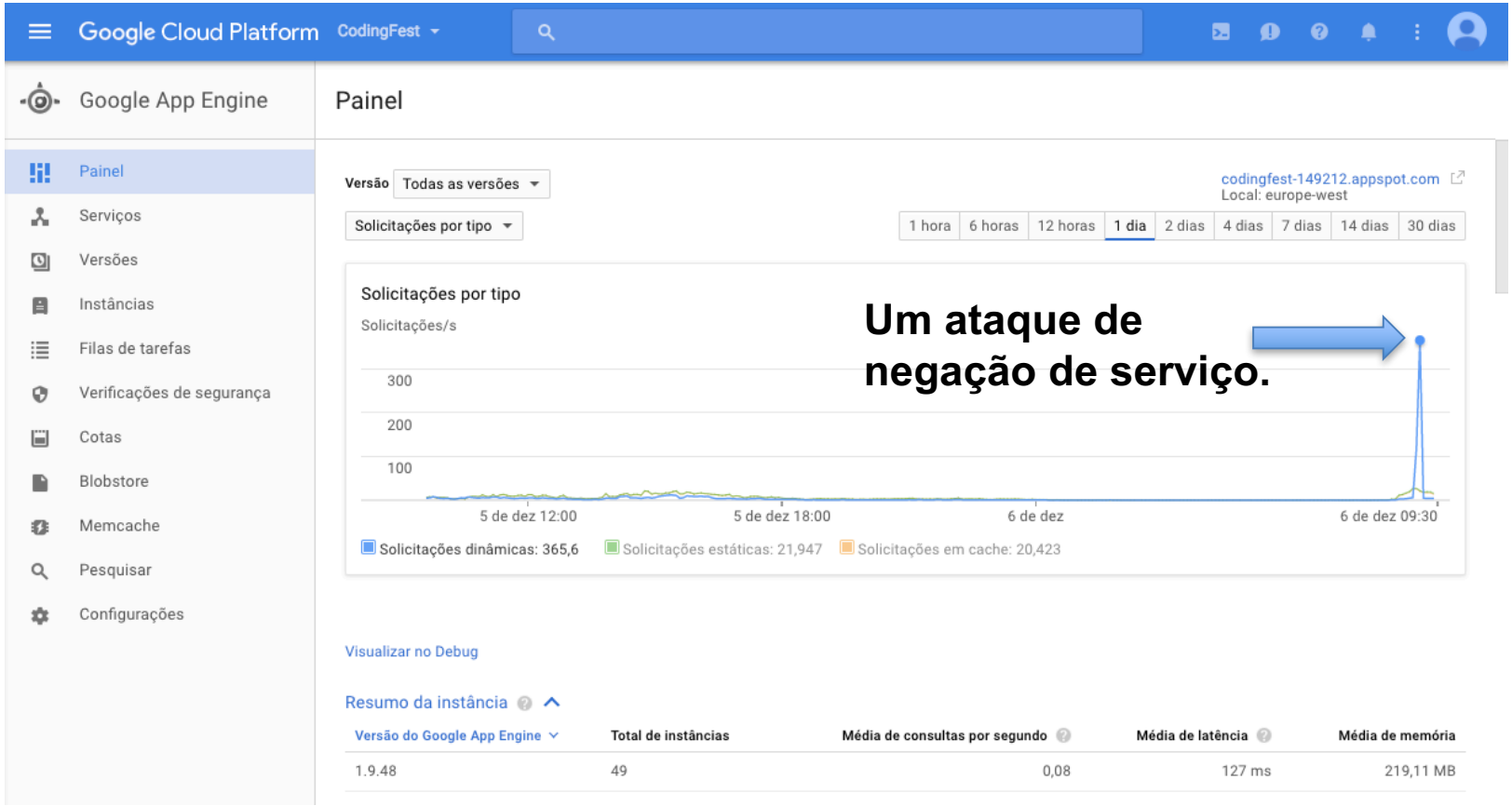
Use CDN for reducing load time and save bandwidth.



SCALING NUMBER OF INSTANCES

The number of instances can be automatically scaled out:

- Increase the number of instances to support peak load;
- Decrease the number of instances when load is low.



SCALING NUMBER OF INSTANCES: RISKS

Costs may increase too much:

- Typical solution: possible to define maximum number of instances and plafond.

SCALING UP AN WEB APPLICATION

It is also possible to scale up a web application by running it in more powerful machines.

https://azure.microsoft.com/en-us/pricing/details/app-service/linux/

INSTANCE	CORES	RAM	STORAGE	PRICES
S1	1	1.75 GB	50 GB	~$69.35/month
S2	2	3.50 GB	50 GB	~$138.70/month
S3	4	7 GB	50 GB	~$277.40/month

DEPLOYING NEW VERSIONS

Software is always being updated. Issues involved in deploying new versions:

- No downtime: do not want to stop a version before starting the new one;
- Simple operations: minimize the involvement of system administrators and speedup deployment;
- Minimize the impact of bugs, and testing of new features.

Supports continuous deployment.

- Automatically build and deploy new versions from a source repository (e.g. GitHub).

DEPLOYING NEW VERSIONS: STAGING

How to avoid downtime?

Basic approach:

1. Keep the old version running;
2. Deploy the new version to a staging slot (using Azure terminology) – instances are created but requests from clients keep being processed by the old version;
3. (Optionally) warm-up application;
4. Hotswap from the old to the new version, by start forwarding client requests to the new version.

Deploying new versions: A/B deployment

When a new version is deployed, it may contain bugs that impact functionality and/or performance.

How to minimize the impact of these bugs?

Use A/B testing/deployment. Basic approach:

1. Have two versions running at the same time;
2. Forward a small percentage of requests to the new version.

HIGH AVAILABILITY AND LOW LATENCY

Possible to deploy web apps at multiple regions.

Integration with geo-replicated data stores.

Other features: Azure App Service

Security.

- Authenticate users with Azure Active Directory or with social login (Google, Facebook, Twitter, and Microsoft).

Support for API and mobile features.

Integration with serverless code.

OUTLINE

Web applications

Blob storage service

BLOB STORAGE SERVICES

Applications typically manage large amounts of unstructured data: files (such as static HTML pages), images, videos, log files, etc.

In a computer, this data is stored in the file system.

Do we need “full” filesystem functionality?

BLOB STORAGE SERVICES (2)

Blob stores are designed for storing massive amounts of unstructured data.

- Serving images or documents directly to a browser.
- Storing files for distributed access.
- Streaming video and audio.
- Writing to log files.
- Storing data for backup and restore, disaster recovery, and archiving.

EXAMPLES

Azure

- Azure Blob storage

AWS

- Amazon S3, Amazon Glacier

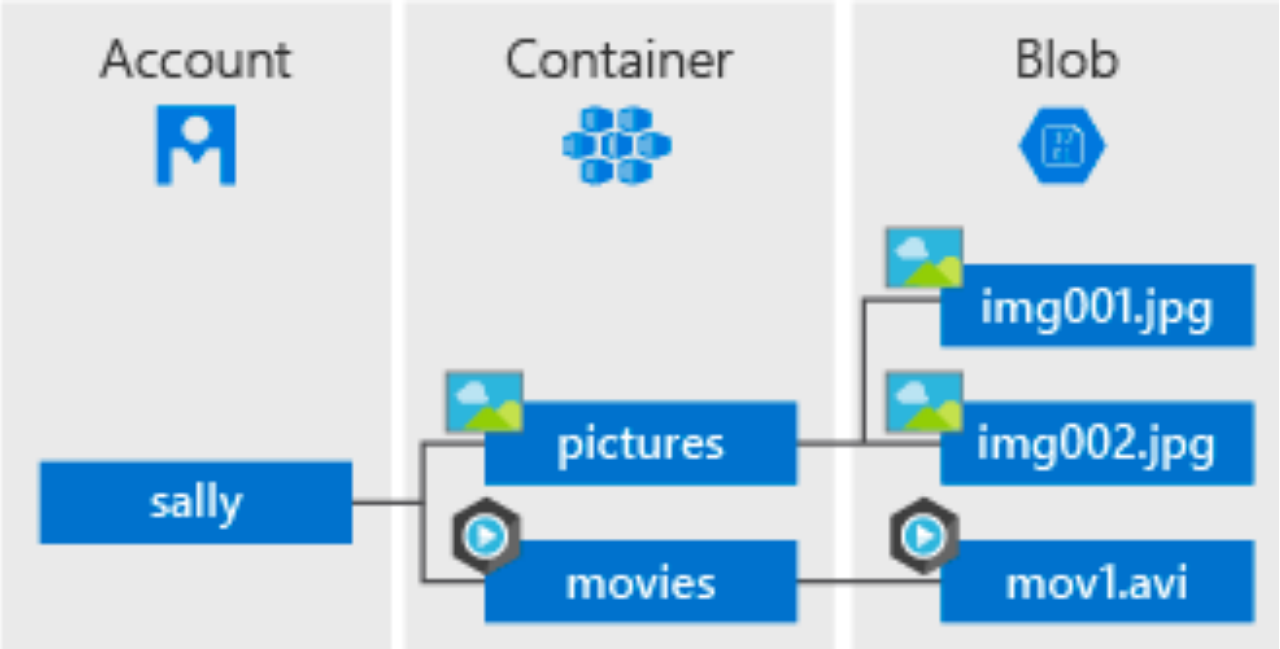
Google

- Google Cloud Storage (Nearline / Coldline)

Data model (Azure blob storage)

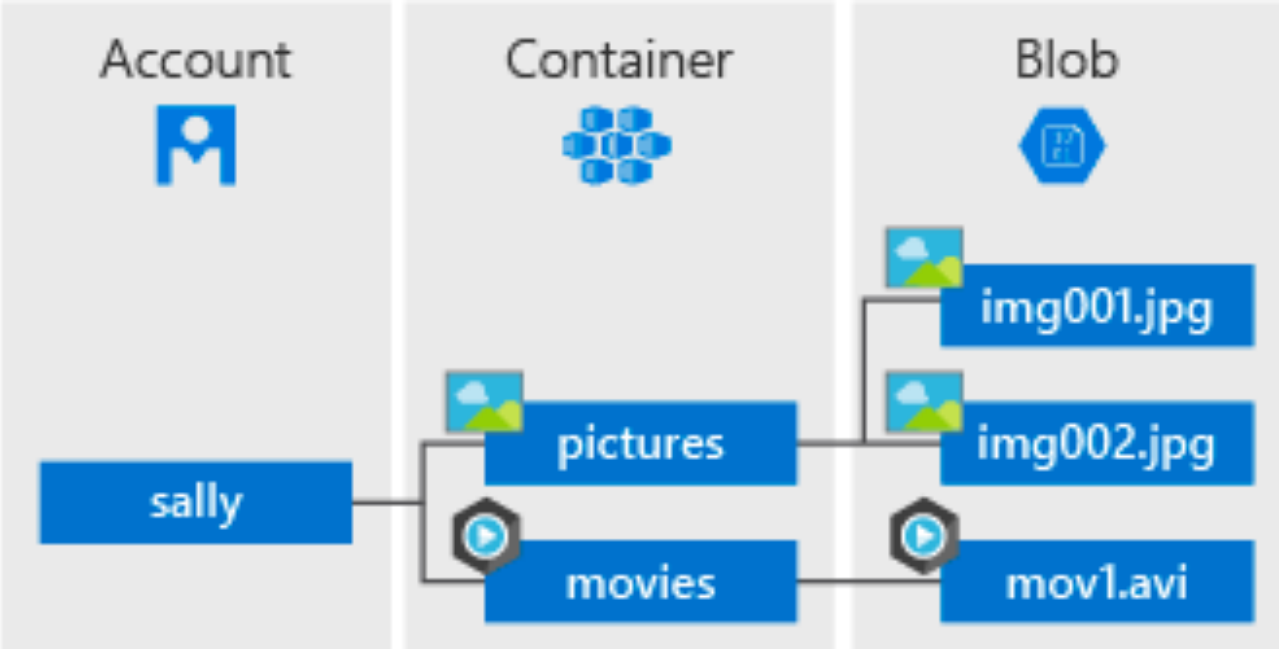
Blob storage offers three types of resources:

1. The **storage account**.
2. A **container** in the storage account.
3. A **blob** in a container.



DATA MODEL (AZURE BLOB STORAGE) (2)

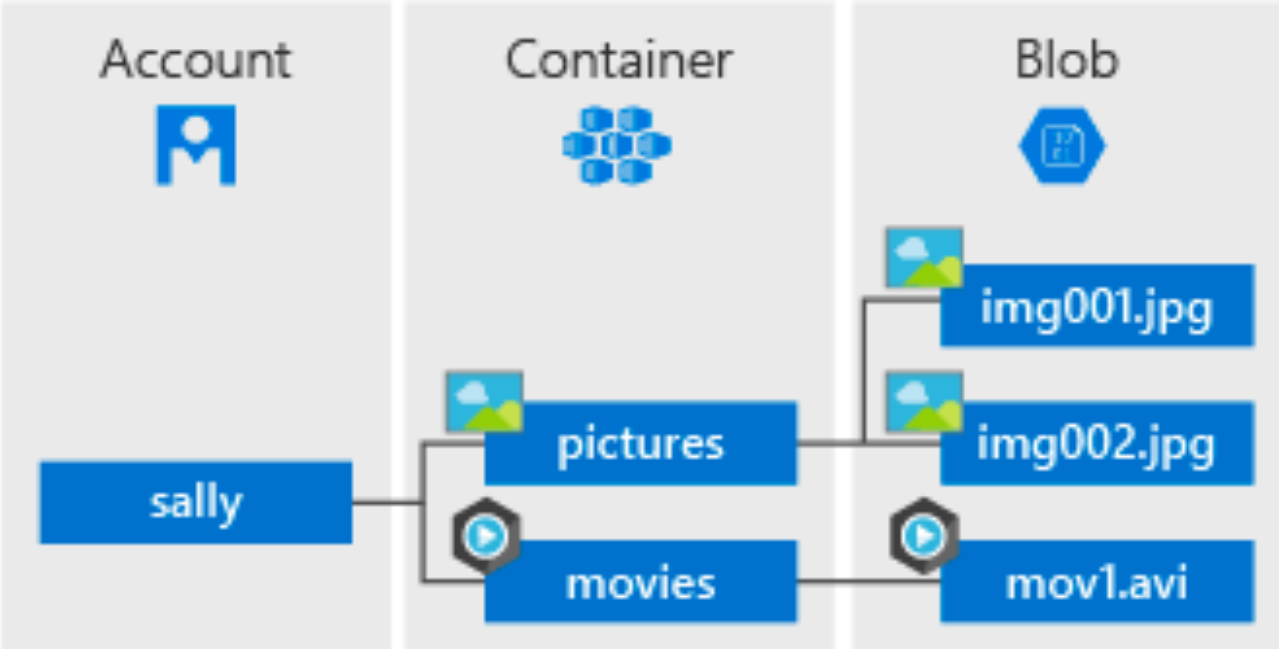
The **storage account** provides an unique namespaces for data.



DATA MODEL (AZURE BLOB STORAGE) (3)

A **container** organizes a set of blobs, similar to a directory in a file system.

A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.



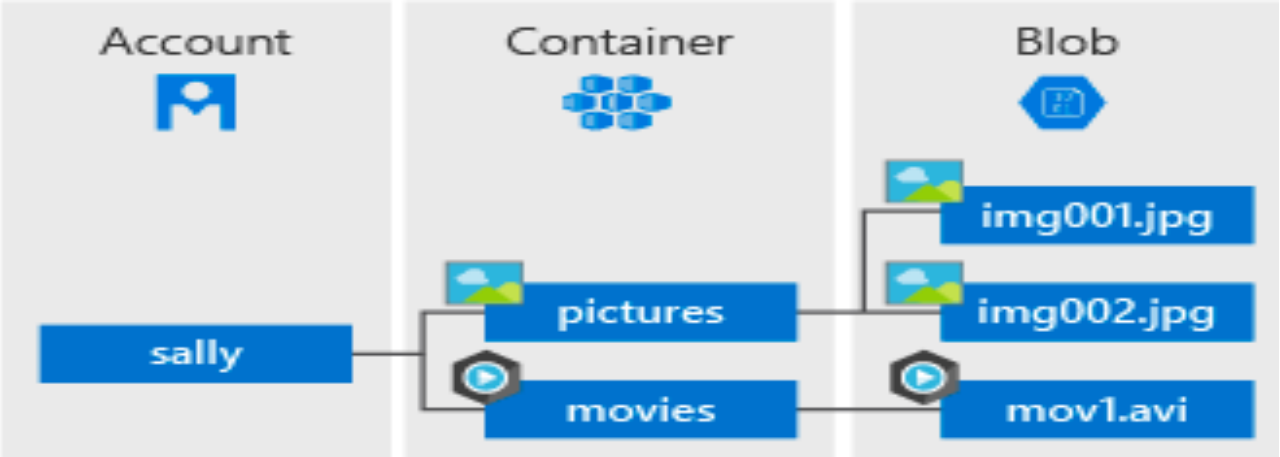
DATA MODEL (AZURE BLOB STORAGE) (4)

Azure Storage supports three types of blobs:

Block blobs store text and binary data, up to about 190 TB.

Append blobs are made up of blocks optimized for append operations. *This can be used for what?*

Page blobs store random access files up to 8 TB in size. Page blobs store virtual hard drive (VHD) files and serve as disks for Azure virtual machines.



NOT ALL DATA IS EQUALLY IMPORTANT

In an application, not all data is accessed with the same frequency.

Consider Instagram:

1. What happens to the images posted a few years ago?
2. What happens to the images posted last week?

AZURE PERFORMANCE TIERS

Premium

- Optimized for high transaction rates and single-digit consistent storage latency
  Interactive workloads, analytics, AI/ML, data transformation

Standard

- Optimized for high capacity and high throughput
  Media contents, backup, bulk data processing

AZURE ACCESS TIERS

Hot access tier

- Optimized for **frequent access** of objects. Storage costs are higher.
  New storage accounts are created in the hot tier by default.

Cool access tier

- Optimized for storing large amounts of data that is **infrequently accessed** and stored for at least 30 days. Accessing data may be more expensive than accessing data in the hot tier.

Archive tier

- Optimized for data that can tolerate **several hours of retrieval** latency and will remain in the Archive tier for at least 180 days.

ACCESS/PERFORMANCE TIERS UNDER THE HOOD

Why this complexity?

Different techniques of replication lead to different trade-offs between processing and space used

- (Standard) **Replication**: copies of the data blocks are maintained in multiple machines. E.g.: for tolerating one fault, need to have (at least) two copies. Space = 2 x data size

- **Erasure coding**: from a blob, generate N parts. It is possible to recreate the blob with M < N of the parts. E.g. with XOR (not used in practice), for tolerating one fault, need to have only 1.5 x data size

ACCESS/PERFORMANCE TIERS UNDER THE HOOD (2)

Why this complexity?

Not all machines and disks have the same performance

- Can use slower HDDs and slower machines for lower access tiers;
- For data that is accessed very infrequently (archive tier), having a machines running is costly (for the benefit).
- Cloud providers have built special DCs with low-cost machines, where each machine runs only for a small amount of time.
- Helps keeping the costs lower by:
  - Using cheaper computers, disks;
  - Using less power – as only some machines are running;
  - Avoid expensive air conditioning – as only some machines are running at a given moment.

REDUNDANCY: WHAT FOR?

Redundancy can be used for:

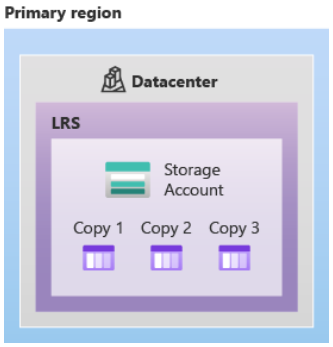
1. Providing high availability / fault tolerance;
2. Providing faster access to data (due to additional copies and location of copies).

AZURE REDUNDANCY LEVELS

Locally-redundant storage (LRS)

Tolerate machine failures in a region.

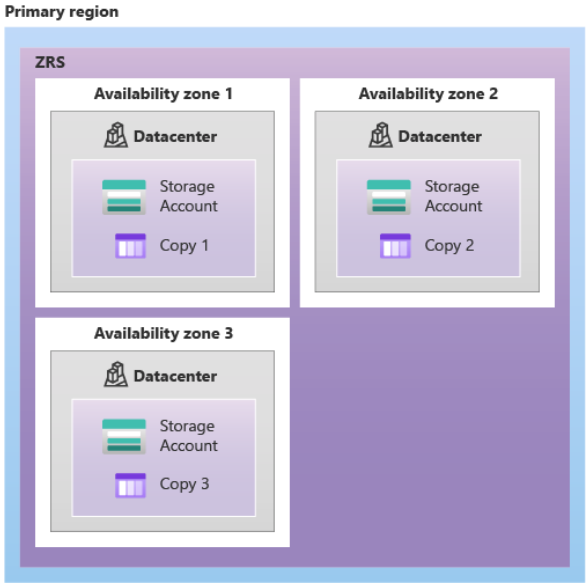
Data is replicated synchronously three times within the primary region.



Zone-redundant storage (ZRS)

Tolerates data center failures in a given region.

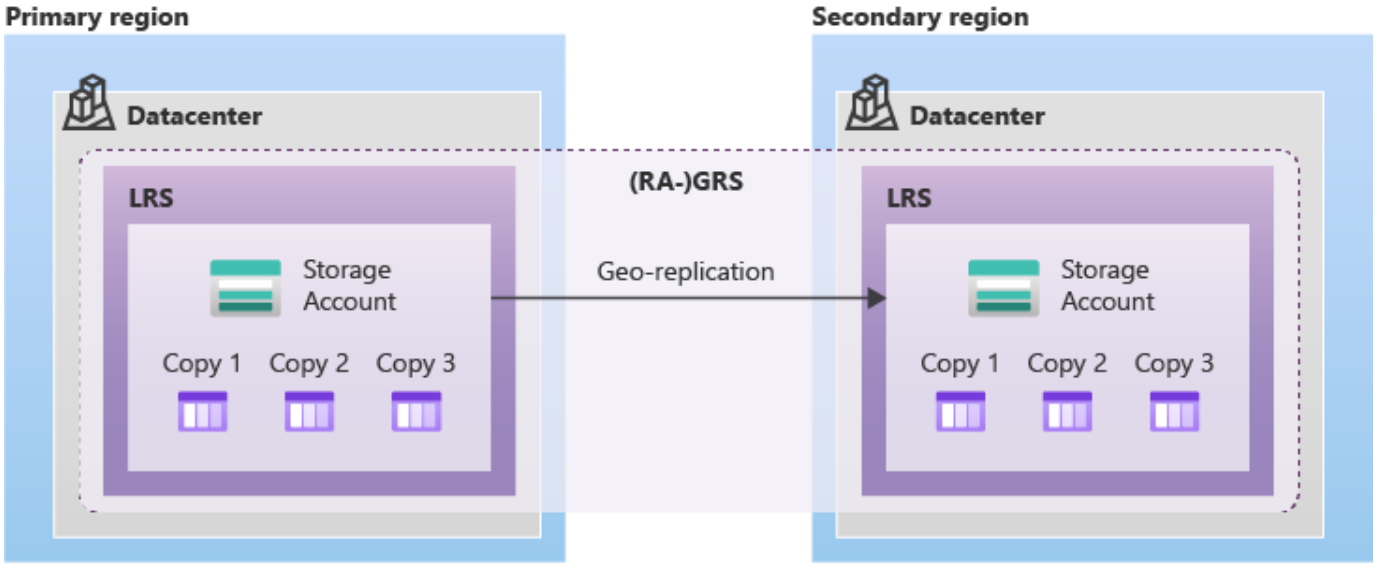
Data is replicated synchronously across three Azure availability zones in the primary region.



AZURE REDUNDANCY LEVELS (2)

Geo-redundant storage (GRS)

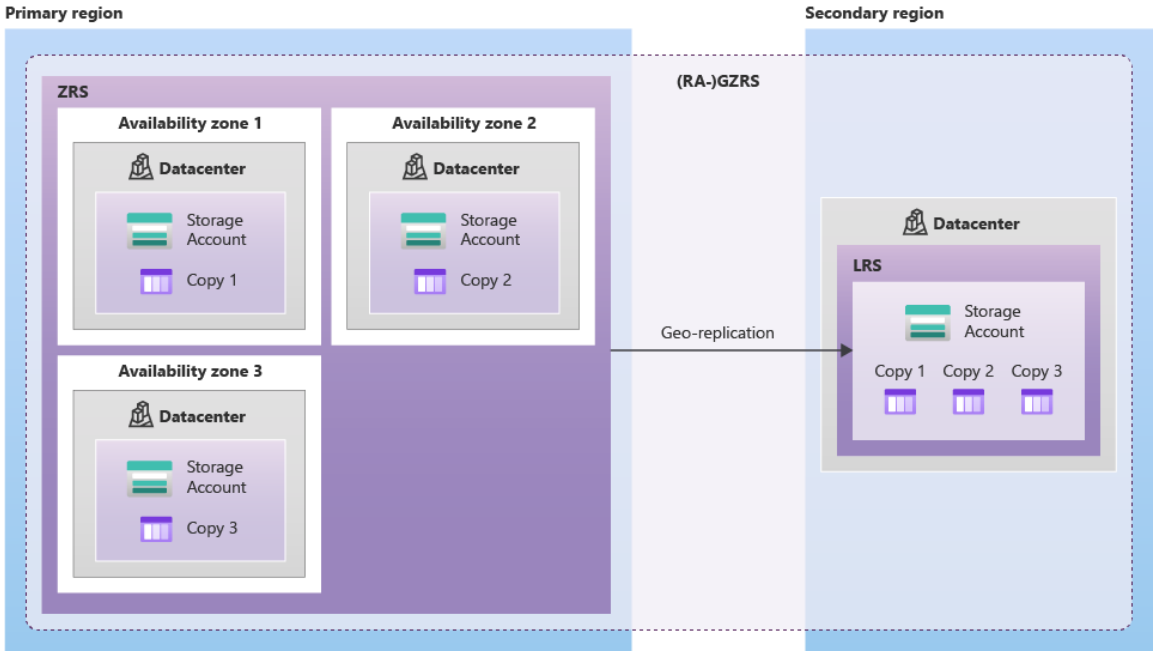
- Tolerates complete region failures – with RA-GRS, it is possible to read from other region – potential better latency.
- Data is replicated synchronously three times in the primary region, then replicated asynchronously to the secondary region.



AZURE REDUNDANCY LEVELS (3)

Geo-zone-redundant storage (GZRS)

- Tolerates both data center failures and complete region failures – with RA-GZRS, it is possible to read from other region – potential better latency.
- Data is replicated synchronously across three Azure availability zones in the primary region, then replicated asynchronously to the secondary region.



REDUNDANCY LEVELS UNDER THE HOOD

Three times replication

- Inside a DC, data is replicated in different racks. Why? E.g. HDFS stores two copies in a rack and one in a second rack.

Synchronous vs. asynchronous replication

- Synchronous replication in the primary zone for establishing the "official" state of the blob.
- Asynchronous replication to secondary zone, allowing to return the result of a write operation to the client without incurring in inter-region latency.

PROTECTION

Soft delete

- Blob soft delete protects an individual blob (or container) from accidental deletes or overwrites.
- Maintains the deleted data for a specified period of time, during which it is possible to "undelete" the blob.

Versioning

- Allow to maintain multiple version for a blob.

SECURITY

Data stored is encrypted using AES.

- Shared account keys, shared access key, active directory.

Communications always uses secure channels.

Role-based access control.

CONCURRENCY

What happens if multiple clients write to the same blob?

- By default, Azure uses a ***last-writer-wins*** policy: all updates are accepted, and the system will keep only the one with the largest timestamp.

CONCURRENCY (2)

Alternative policies:

- Optimistic concurrency with conditional updates

A client can specify some condition that must hold for an update to be accepted – e.g. that the current version of an object is some version previously observed.

Why is this interesting?

CONCURRENCY (3)

Alternative policies:

- Pessimistic concurrency

A client can lock a block for exclusive access.

Lock are granted with a lease time – the time the lock is granted, after which the server assumes that the lock is lost unless the client renews the lease.

REFERENCES

Some text and images from Microsoft Azure online documentation.

https://docs.microsoft.com/en-us/azure/app-service/overview-hosting-plans

https://docs.microsoft.com/en-us/rest/api/storageservices/blob-service-concepts

ACKNOWLEDGMENTS

Some text and images from Microsoft Azure online documentation.