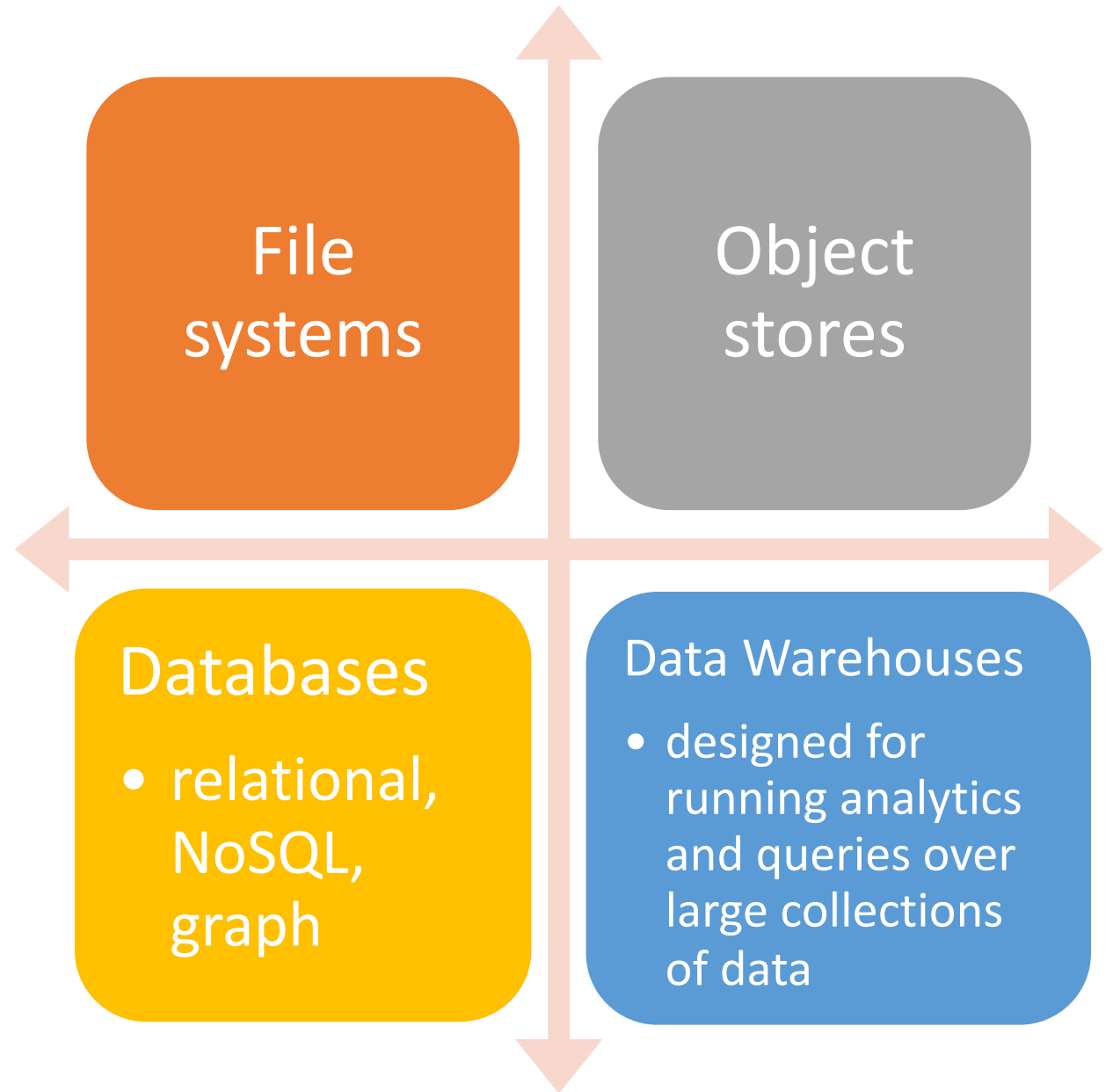




Cloud Storage

The First Application
of Cloud

Types of Cloud Storage



File Systems

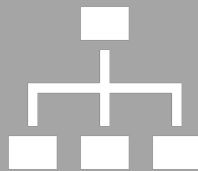
Standard API for the Unix-derived file systems is called the Portable Operating System Interface (POSIX)

File systems allow us to create, read, write, and delete files located within directories using command line tools, graphical user interfaces, or APIs.

File Systems: Advantages



Direct use of many
existing programs
without modification



Provides a straightforward
mechanism for representing
hierarchical relationships
among data—directories and
files

File Systems: Disadvantages

Provides no support for enforcing conventions concerning the representation of data elements and their relationships

The rigid hierarchical organization enforced by a file system often does not match the relationships that you want to capture

File Systems: Disadvantages

The file system model also has problems from a scalability perspective; the need to maintain consistency as multiple processes read and write can lead to bottlenecks

Can be difficult to efficiently back-up: slow to back-up each file at a time instead of an entire volume; if there is volume back-up this is not a problem

Object Stores

- Store unstructured binary objects or **blobs** (binary large objects)
- Eliminate hierarchy and **forbids updates to objects** once created - only allows delete or replace (if **versioning** allowed)

Object Stores

- Support a two-level folder-file hierarchy that allows for the creation of object **containers**, each of which can hold zero or more objects
- Each object is identified by a **unique identifier** and can have various metadata associated with it

Object Stores: Advantages

- Simplicity, performance, and reliability
- Since objects cannot be modified once created makes it easy to build highly **scalable** and **reliable** implementations

Replication without synchronization

A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Object Stores: Limitations

- Provides little support for organizing data and no support for search: you must know an object's identifier to access it
- Cannot easily be mounted as a file system or accessed with existing tools in the ways that a file system can

Object Store Model

PutObject(myobj, Container='A', metdata = 'NetCDF')

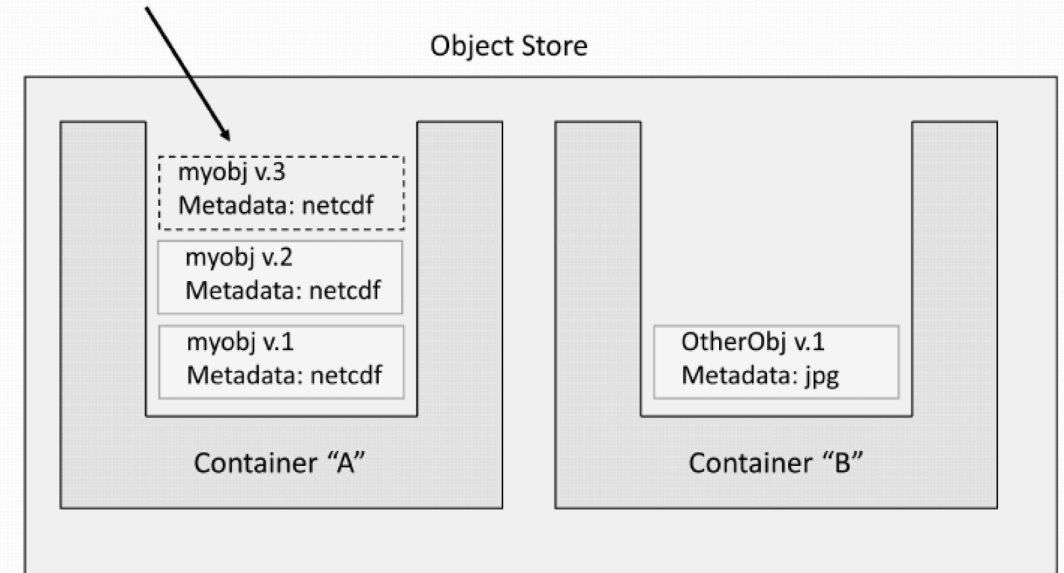


Figure 2.1: Object storage model with versioning. Each NetCDF file is stored in a separate container, and all versions of the same NetCDF file are stored in the same container.

Databases

- The use of a DBMS simplifies data management and manipulation and provides for
 - efficient querying and analysis
 - durable and reliable storage
 - scaling to large data sizes
 - validation of data formats
 - management of concurrent accesses

Types of Databases

- Relational databases
- NoSQL databases
- Graph databases
 - often graph databases are built on top of existing NoSQL databases

Relational Databases

- Useful for searching data based on relationships among data items
- Support a relational algebra that provides a clear, mathematical meaning to the SQL language, facilitating efficient and correct implementations

Relational Databases

- MySQL and Postgres are available in Cloud-hosted forms
- Cloud vendors offer specialized relational DBMS's that are designed to scale to particularly large data sizes

Relational Databases

Support **ACID** semantics:

- **Atomicity** (the entire transaction succeeds or fails)
- **Consistency** (the data collection is never left in an invalid or conflicting state)
- **Isolation** (concurrent transactions cannot interfere with each other)
- **Durability** (once a transaction completes, system failures cannot invalidate the result)

Motivations

- scaling the quantities of data and number of users that can be supported
- dealing with unstructured data not easily represented in tabular form

Key-Value store can organize large numbers of records, each of which associates an arbitrary key with an arbitrary value

- a document store permits text search on the stored values

NoSQL Databases

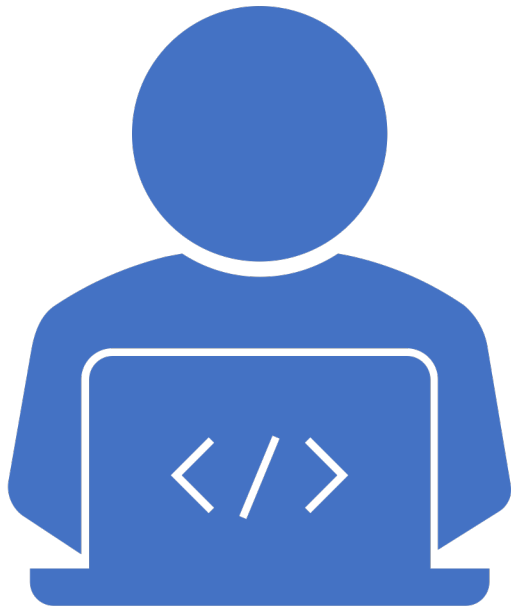
NoSQL Databases

- NoSQL databases in the Cloud are often distributed over multiple servers and also replicated over different data centres
- They often fail to satisfy all of the **ACID** properties
- **Consistency** is often replaced by *eventual consistency*, meaning that database state may be momentarily inconsistent across replicas

It is not possible to create a distributed system with all three of the following properties:

- **Consistency** - all computers see the same data at the same time
- **Availability** - every request receives a response about whether it succeeded or failed
- **Partition tolerance** - the system continues to operate even if a network failure prevents computers from communicating

The CAP Theorem



The CAP Theorem

- DBMS designer must choose between *high consistency* or *high availability* for a particular system



I. Foster, R. Ghani, R. S. Jarmin, F. Kreuter, and J. I. Lane, editors. Big Data and Social Science: A Practical Guide to Methods and Tools. Taylor & Francis Group, 2016. See also <http://www.bigdatasocialscience.com>

Table 2.1: Storage as a service options from major public cloud vendors.

Model	Amazon	Google	Azure
Files	Elastic File System (EFS), Elastic Block Store (EBS)	Google Cloud attached file system	Azure File Storage
Objects	Simple Storage Service (S3)	Cloud Storage	Blob Storage Service
Relational	Relational Data Service (RDS), Aurora	Cloud SQL, Spanner	Azure SQL
NoSQL	DynamoDB, HBase	Cloud Datastore, Bigtable	Azure Tables, HBase
Graph	Titan	Cayley	Graph Engine
Warehouse analytics	Redshift	BigQuery	Data Lake


Cloud Computing for Science and Engineering

<https://cloud4scieng.org/>




Amazon File System Storage

- **Elastic Block Store (EBS)** and **Elastic File System (EFS)** services offer related but different services
- **EBS** is a device that you can mount onto a single Amazon **EC2** compute server instance at a time
 - low-latency access to data from a single EC2 instance



Amazon File System Storage

- **EFS** is a general-purpose file storage service that provides a file system interface, file system access semantics , and concurrently-accessible storage for *many* Amazon EC2 instances
- Both **EBS** and **EFS** can be accessed directly only by **EC2** instances (from inside the Amazon cloud)


- 
- Amazon's Simple Storage Service (**S3**) was its first cloud service.
 - As of 2016, it reportedly contained trillions of objects in billions of containers
 - S3 containers are called **buckets**
 - S3 is a classic object store



Amazon
Object
Storage




AWS NoSQL Storage

- DynamoDB is a fully managed NoSQL service based on key-value pair
 - Requires only a primary key and does not require a schema to create a table
 - Tutorial:
<https://www.edureka.co/blog/amazon-dynamodb-tutorial>
- 




Features of DynamoDB

- **On-demand capacity mode:** DynamoDB automatically scales up/down to accommodate the traffic
 - **Built-in support for ACID transactions:** DynamoDB provides native/server-side support for transactions
 - **On-demand backup:** You can create a complete backup of your work at any given point of time
- 



Features of DynamoDB

- **Point-in-time recovery:** Your data is protected in case of accidental read/ write operations
 - **Encryption at rest:** Data is encrypted even when the table is not in use
- 




Tables

Three fundamental units:

- **Table:** a table is a group of items
- **Attribute:** An attribute is a single field that is attached to an item
- **Items:** An item is a set of attributes in a table



Keys

- A **primary key** is a unique attribute necessary for creating a table; it cannot be null at any given point
 - A simple primary key, or **Partition key**, is a single attribute and its value is used to uniquely distinguish items in a table
 - A composite primary key has a primary component - the **Partition key** and a secondary component - the **Sort key**
 - A **secondary index** is an attribute that allows you to query the data, with or without a primary key
- 

- Allows users to create file shares in the cloud that can be accessed by a special protocol, **SMB**, that allows Microsoft Windows VMs and Linux VMs to mount these file shares as standard parts of their file system
 - In the Cloud and **outside** of the Cloud

Azure File System Storage

Azure Object Store

- **Azure Blob** storage service is concerned with highly reliable storage of unstructured objects (**blobs**)
 - similar to Amazon's S3
- Azure Blob storage has tiered storage and pricing
 - **hot** for frequently accessed data
 - **cool** for data accessed less often
- Azure Storage Explorer

<https://azure.microsoft.com/en-us/features/storage-explorer/>

Google Cloud Storage: Persistent Disk

- Persistent disks are durable network storage devices that your instances can access like physical disks in a server. The data on each persistent disk is distributed across several physical disks.
- Persistent disks are located independently from your virtual machine (VM) instances, so you can detach or move persistent disks to keep your data even after you delete your instances.

Google Cloud Storage: Cloud Storage Buckets

- Cloud Storage buckets are the most flexible, scalable, and durable storage option on Google Cloud
- Cloud Storage bucket is an **object** storage system that does not have the same write constraints as a POSIX file system.
- But you can mount a Cloud Storage bucket to an instance (VM) as a file system using the Cloud Storage FUSE tool
 - but it cannot be used as a boot disk