Data storage in the cloud

COSC349—Cloud Computing Architecture

David Eyers

Learning objectives

- Can contrast **object storage** with **filesystem storage**
- Indicate why object storage scales-out so well
- Define Amazon S3 **buckets**, **objects** and **keys**
- Contrast Internet speeds against couriered hard-disks
- Explain how S3's use of REST can allow it to serve **resources for websites** effectively

Cloud storage is a multifaceted topic

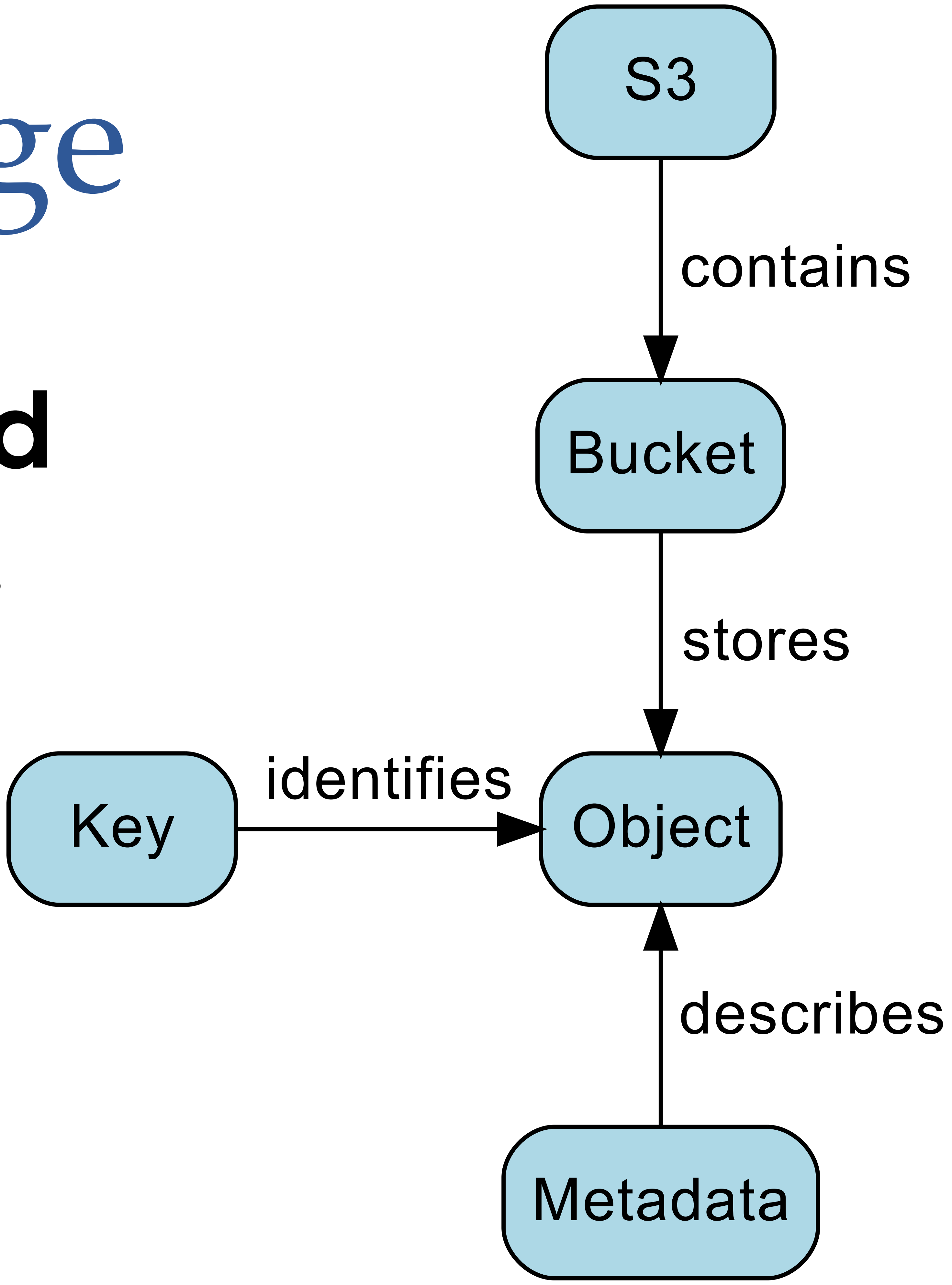
- Software engineering side: **architectures for storage**
  - Transitions from previous ways of managing storage
    - e.g., files and folders on filesystems provided by operating systems
  - New solutions that are specifically focused on the cloud
    - e.g., **object storage** such as S3
    - usually relies on scalability of cloud resources

- Also issues of **data transfer rates** and **costs**
  - Transfer of data through the Internet is actually quite slow…

Internet bandwidth often beaten by couriers

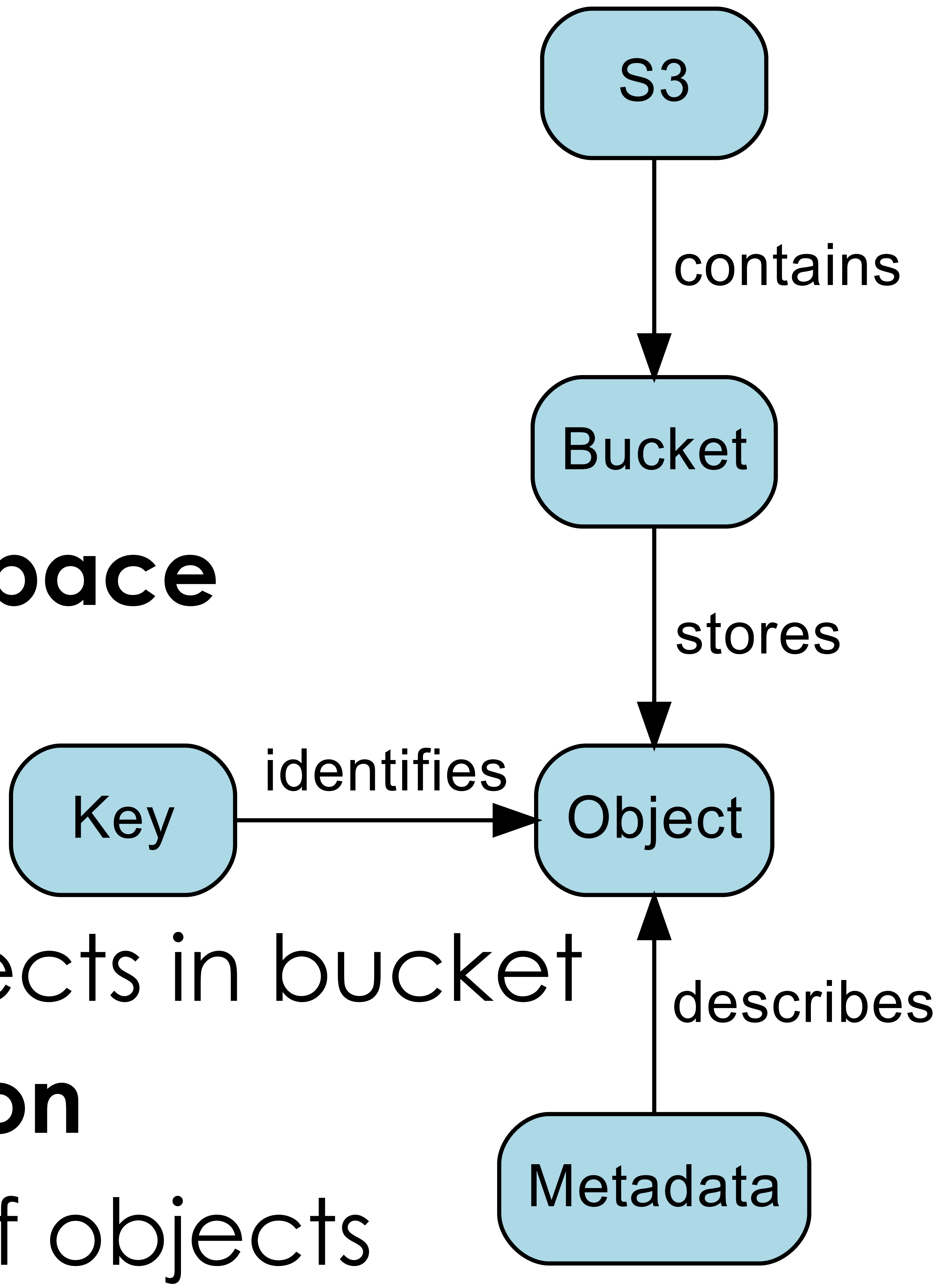
- Data transfer involves **data volume** and **time delay**
  - Internet pipes are quite responsive, but transfer slowly
    - 100 megabit/s dedicated Internet; 50 terabytes of data …
    - 50 TB × 1000 × 1000 × 8 ÷ 100 = 4,000,000s … approximately **50 days**!
  - Courier over a 50 TB hard disk? Likely to take **a few days**…
- Amazon Snowball makes **hard-disk shipping** a service
  - Hardened storage appliances are shipped from Amazon
  - Client transfers data on/off
  - Prepaid courier service returns the device (& data) to Amazon

Amazon S3 (2006)—object storage



- Simple Storage Service (S3) is **object-based**
  - **Not a traditional filesystem** with files and folders
- **Objects** are just a sequence of bytes
  - Each object is stored in a single bucket
  - Each object can contain up to 5TB of **data**
- Refer to objects by URI including developer-selected **keys**
  - `https://mybucketname.s3.amazonaws.com/`$\boxed{\texttt{mydata/file.jpg}}$
  - Requests can be **authenticated** or **anonymous**

Buckets in Amazon S3



- **Buckets** represent a storage collection
  - Bucket names are the top-level of S3 **namespace**
  - **Charges** accrue at the bucket level
  - **Usage reports** aggregated at bucket level
  - Default **access control** configuration for objects in bucket
  - Buckets can be placed in a given **AWS Region**
  - Per-bucket option to keep all past versions of objects
    - Amazon assigns unique version ID to all objects added to a bucket

S3 objects compared to files in typical OS

- Both S3 objects and files contain **data** and **metadata**
  - In filesystems: modification time, file size, access control, …
  - In S3: metadata is a set of key/value pairs in two groups
    - **System-defined**: time updated, also HTTP headers like Content-Type
    - **User-defined**: key/value data useful to tenants' applications

- Can read/write **parts of files** but S3 has **atomic access**
- Files' metadata can be **updated dynamically**
  - S3 **fixes metadata** at the time an object is stored

S3 keys identify objects within buckets

- S3 keys are the **names for objects** within a bucket
  - Previous example URL had key `mydata/file.jpg`
- Delimiters in keys can be used to **imply a hierarchy**
  - Amazon tools support this, but really **keys are a flat structure**
  - In S3, objects in a bucket are treated as a single collection
  - … unlike filesystems, which really scope files within directories
- Key names can use any UTF-8 character…
  - … but there is a safe set likely to work across all applications

Amazon’s many S3 storage classes

- **S3 Standard Storage**—high durability, multi-zone, fast
- **S3 Standard-Infrequent Access**—slower access
- **S3 One Zone-Infrequent Access**—lower resilience
- **S3 Intelligent Tiering Frequent / Infrequent**
  - Monitors access patterns and auto-migrates
- **S3 Glacier Storage**—retrievals take minutes to hours
  - Also S3 Glacier Deep Archive Storage—12 hour retrieval
    - *e.g.*, for organisations with annual audits: retrieve 1 or 2 times a year
- S3 lifecycle management can **automate class change**

Payment for S3

- Prices are based on the location of bucket (its region)
- Two broad cost classes: storage and data transfer
- **Storage costs** depend on storage class (set per object)
- **Data transfer costs** are asymmetric:
  - Transfer in from Internet to S3 is free
  - Transfer out to Internet is tiered:
    - First 1 GB / month is free
    - Next 10 TB / month is around $0.09 / GB
  - Transfer to other Amazon regions is around $0.02 / GB

Representational State Transfer (REST)

- REST is a notion retrofitted to **HTTP's 'object model'**
- **Resources** have a standardised, universal form (URIs)
- Predefined set of **generic operations** are used on URIs
  - **Operations are stateless** on the server's side
- Consider how the web works:
  - URIs are addresses such as `https://www.google.com/`
  - HTTP methods include `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, …
  - First request for webpage from web browser uses `GET` method
  - A form submission might then later use a `POST` method

Amazon S3 REST API (~787 page docs…)

- Full interaction with S3 supported using REST API, but…
  - Amazon suggest using SDK and/or CLI to ease cert. generation

- REST operations on buckets:
  - `HEAD` method indicates **whether bucket exists** and accessible
  - `GET` method **lists objects** within the bucket
  - `PUT` and `DELETE` **create and destroy buckets**, respectively

- REST for Objects—`GET`, `PUT`, `DELETE` do expected actions
  - Also supports `POST` method from web browser HTML forms

Web functionality cross-over

- S3 REST suited to **direct use from web browsers**
- **GET** request for an image on S3 just as from web server
  - S3 is frequently used to **store other web resources**, like video
- S3 wasn't quite a static web hosting service though:
  - **1**—Accessing bucket root produced a **list of objects in bucket**
  - **2**—Errors in accessing objects produced **S3 error messages**
  - In 2011 bucket configurations added fix for these issues
    - but now need HTTPS hosting; use AWS CloudFront to add HTTPS
    - (… and GitHub Pages, *etc.*, may be simpler and cheaper)