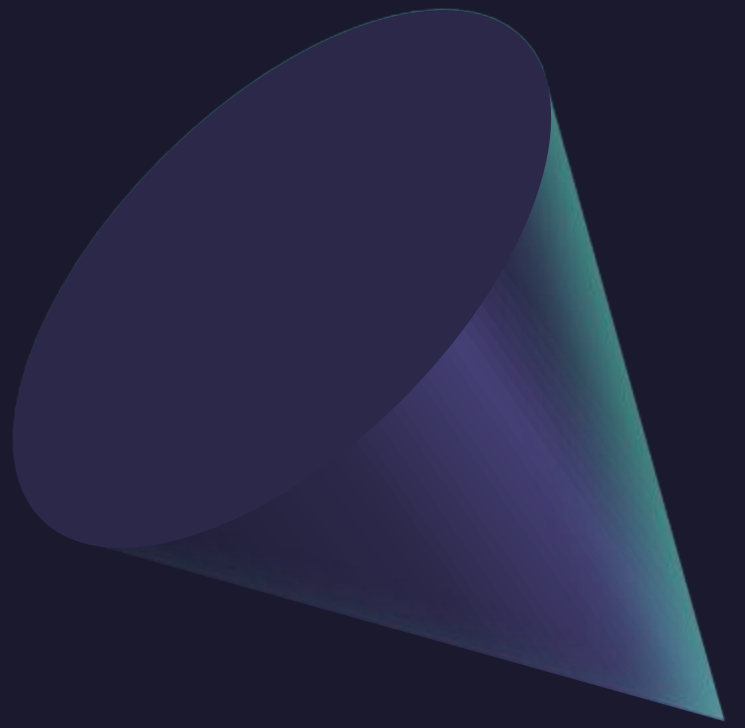
System Design - File Storage

Software Systems Development

IIIT Hyderabad



Content

Cloud Computing

- **Cloud computing** is on-demand access, via the internet, to computing resources–applications, servers (physical servers and virtual servers), data storage, development tools, networking capabilities, and more which is hosted at a remote data center managed by a cloud services provider (or CSP).
- Benefits :
  - Lower IT costs
  - Improve agility and time-to-value
  - Scale more easily and cost-effectively
- **Virtualization** enables cloud providers to make maximum use of their data center resources.

Storage Types in Cloud

- **Block**, **object**, and **file** storage are three ways of storing data in the cloud so that users and applications can access it remotely over a network connection.

Block Storage

- **Block storage** is when the data is split into fixed sized blocks of data and then stored separately with unique identifiers that can be stored in different environments.
- When a user retrieves the data for a block, the storage system reassembles the blocks into a single unit. It can be used for storages which needs faster access and frequent updates like RDBMS.
- Block storage is the default storage for both hard disk drive and frequently updated data and we can store blocks on Storage Area Networks (SANs) or in cloud storage environments.

Pros :

- **Fast**: When all blocks are stored locally or close together, block storage has a high performance with low latency for data retrieval, making it a common choice for business-critical data.
- **Reliable**: Because blocks are stored in self-contained units, block storage has a low fail rate.
- **Easy to modify**: Changing a block does not require creating a new block; instead, a new version is created.

Cons :

- **Lack of metadata**: Block storage does not contain metadata, making it less usable for unstructured data storage.
- **Not searchable**: Large volumes of block data quickly become unmanageable because of limited search capabilities.
- **High cost**: Purchasing additional block storage is expensive and often cost-prohibitive at a high scale.

File Storage

- **File storage** is when all the data is saved together in a single file with a file extension type that's determined by the application used to create the file or file type, such as .jpg, .docx or .txt.
- For example, when you save a document on a corporate network or your computer's hard drive, you are using file storage. It also includes presentations, reports, spreadsheets, graphics, photos, etc.
- File storage uses a hierarchical structure where files are organized by the user in folders and subfolders, which makes it easier to find and manage files.

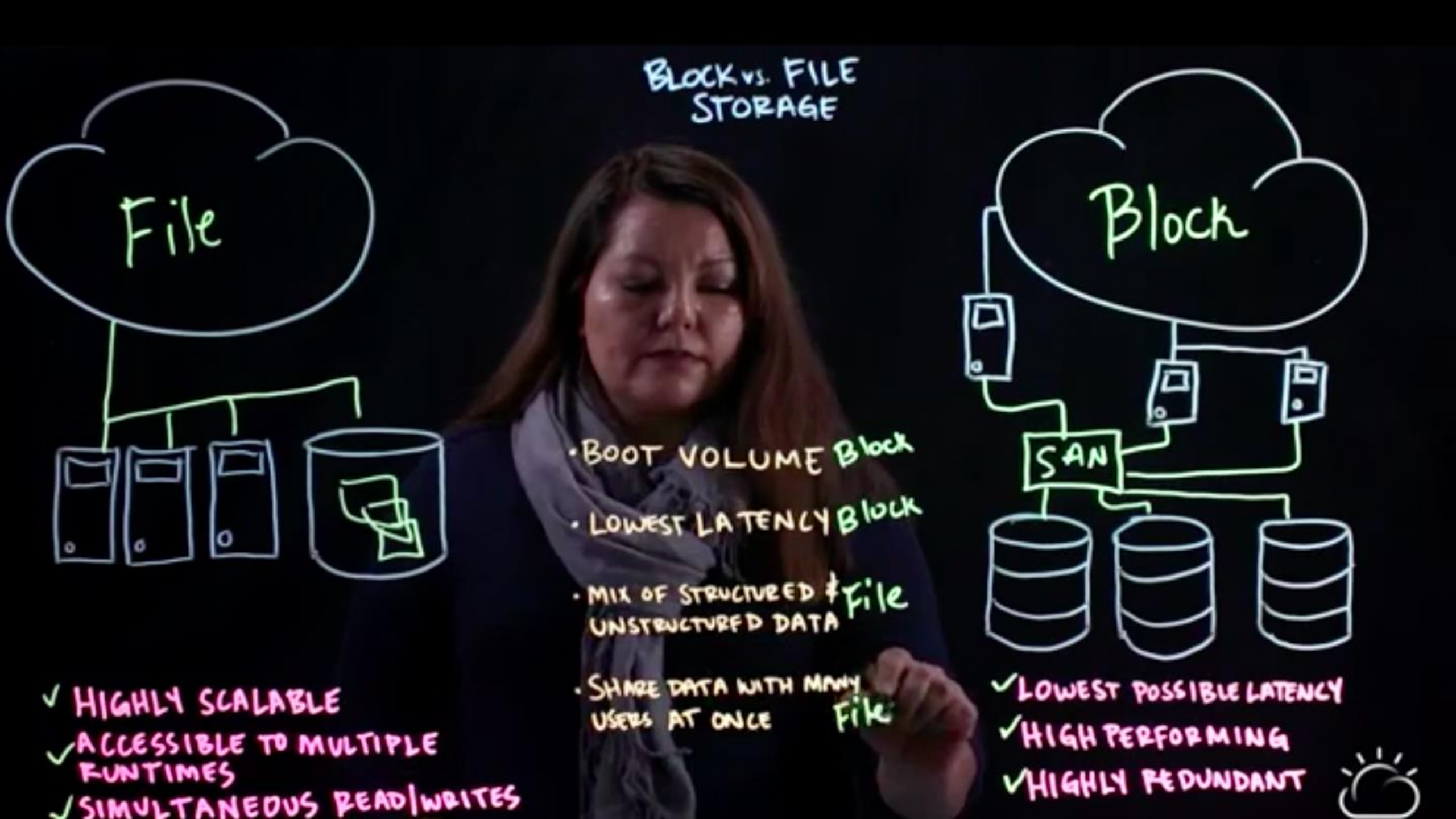
Pros :

- **Easy to access on a small scale:** With a small-to-moderate number of files, users can easily locate and click on a desired file, and the file with the data opens
- **Familiar to most users:** As the most common storage type for end users, most people with basic computer skills can easily navigate file storage with little assistance or additional training
- **Allows access rights/file sharing/file locking to be set at user level**

Cons :

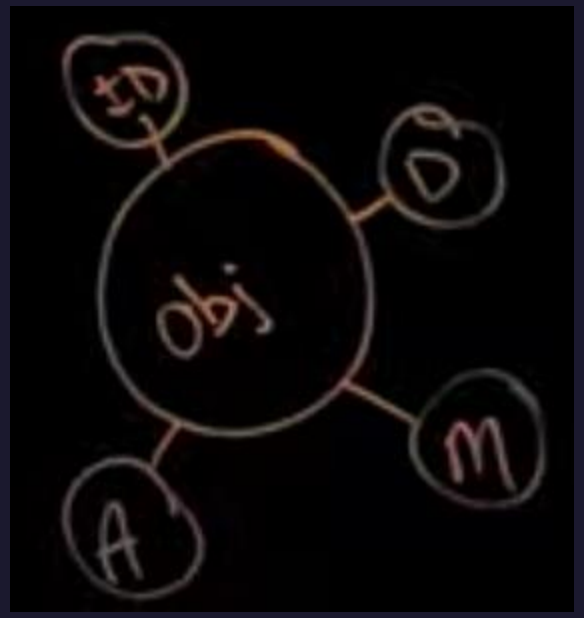
- **Challenging to manage and retrieve large numbers of files:** file management becomes increasingly complicated as the number of folders, subfolders and files increases
- **Hard to work with unstructured data**
- **Becomes expensive at large scales:** When the amount of storage space on devices and networks reaches capacity, additional hardware devices must be purchased.

Block vs. File Storage



Object Storage

- **Object storage** is a system that divides data into separate, self-contained units that are re-stored in a flat environment, with all objects at the same level.
- Object storage uses use an *Application Programming Interface* (API) to access bucket containing and manage objects.
- Objects also contain metadata, which is information about the file that helps with processing and usability, each object has a unique number, data and attributes.
- Write-once read-many times, scalable, moderate latency, cold storage.



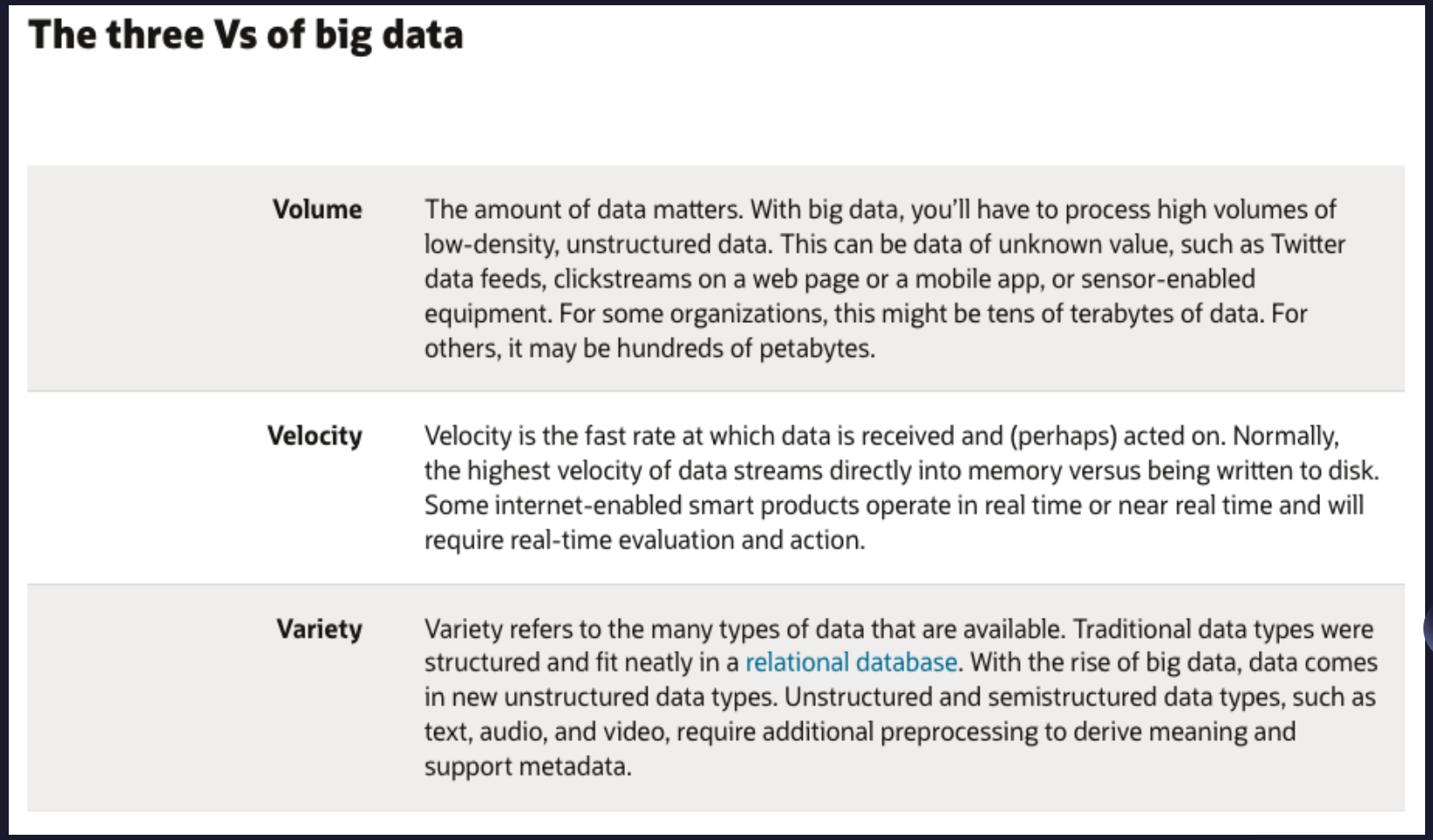
Pros :

- **Handles large amounts of unstructured data**
- **Affordable consumption model**: Instead of paying in advance for a set amount of storage space, as is common with file storage, you purchase only for the object storage you need.
- **Uses metadata**: Because the metadata is stored with the objects, users can quickly gain value from data and more easily retrieve the object they need.
- Ideal for video streaming, file sharing web applications.

Cons :

- **Cannot lock files**: All users with access to the cloud, network or hardware device can access the objects stored there.
- **Slower performance than other storage types**: The file format requires more processing time than file storage and block storage.
- **Cannot modify a single portion of a file**: Once an object is created, you cannot change the object; you can only recreate a new object.

	Object storage	Block storage	Cloud file storage
File management	Store files as objects. Accessing files in object storage with existing applications requires new code and the use of APIs.	Can store files but requires additional budget and management resources to support files on block storage.	Supports common file-level protocols and permissions models. Usable by applications configured to work with shared file storage.
Metadata management	Can store unlimited metadata for any object. Define custom metadata fields.	Uses very little associated metadata.	Stores limited metadata relevant to files only.
Performance	Stores unlimited data with minimal latency.	High-performance, low latency, and rapid data transfer.	Offers high performance for shared file access.
Physical storage	Distributed across multiple storage nodes.	Distributed across SSDs and HDDs.	On-premises NAS servers or over underlying physical block storage.
Scalability	Unlimited scale.	Somewhat limited.	Somewhat limited.



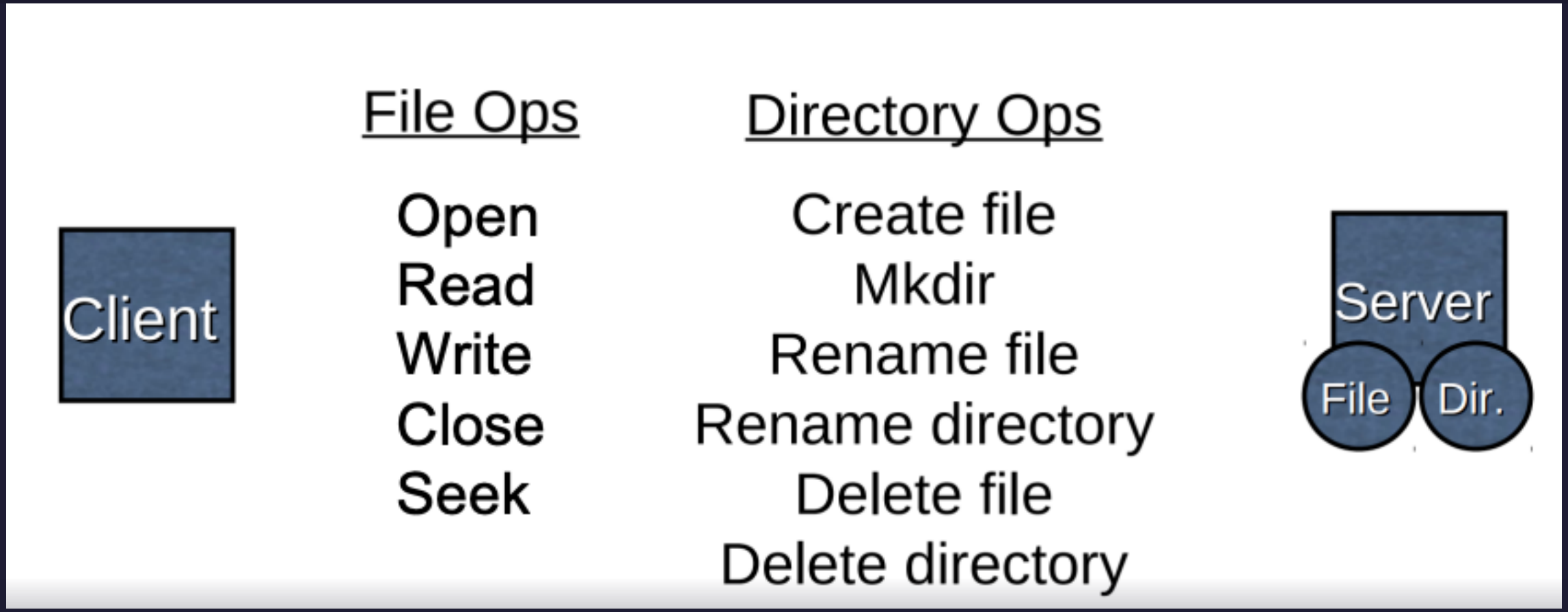
Big Data

- The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity.

Distributed Systems

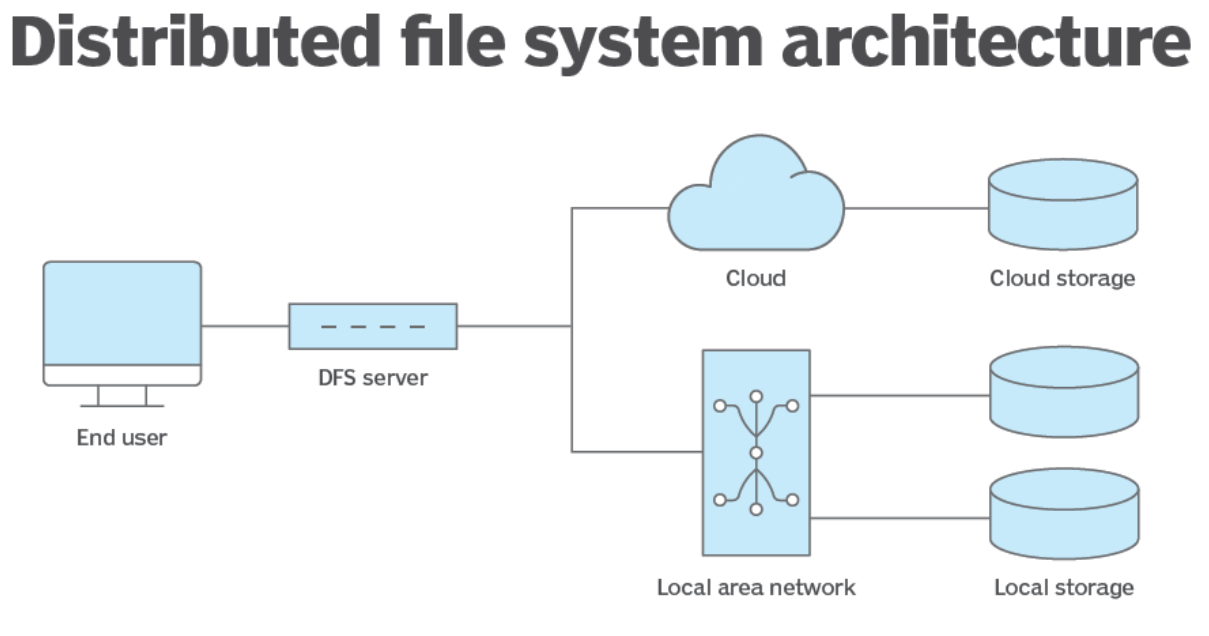
- A **distributed system** is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another.
- **Distributed computing** is a field of computer science that studies distributed systems.
- A computer program that runs within a distributed system is called a **distributed program**, and *distributed programming* is the process of writing such programs.
- There are many different types of implementations for the message passing mechanism, including pure *HTTP*, *RPC-like connectors* and *message queues*.

File System



Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Distributed File Systems



- A **distributed file system (DFS)** is a file system that enables clients to access file storage from multiple hosts through a *computer network* as if the user was accessing local storage.
- The main purpose of the Distributed File System (DFS) is to allows users of physically distributed systems to share their data and resources by using a Common File System.
- Files are spread across *multiple storage servers* and in *multiple locations*, which enables users to share data and storage resources.

Distributed File Systems

- Earliest versions of distributed file systems:
  - NFS – Networked File System
    - Developed at Sun Microsystems which believed in the slogan "The Network is the Computer"
  - AFS – Andrew File System
    - Part of the Andrew project at CMU that created a distributed computing environment at CMU.
- Modern projects on distributed file systems include
  - GFS – The Google File System,
  - HDFS – the OpenSource version of GFS
  - Colossus – the next version of GFS

Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Stateful Vs Stateless Protocols

- **Stateful Protocols:** Server maintains client-specific state
  - Shorter requests
  - Better performance in processing requests
  - Cache coherence is possible – Server can know who's accessing what
  - File locking is possible

- **Stateless Protocols**: Server maintains no information on client accesses
  - Each request must identify file and offsets
  - Server can crash and recover – No state to lose
  - Client can crash and recover
  - No open/close needed
    - They only establish state
    - No server space used for state – can support many clients
    - File locking not possible

Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

CAP Theorem

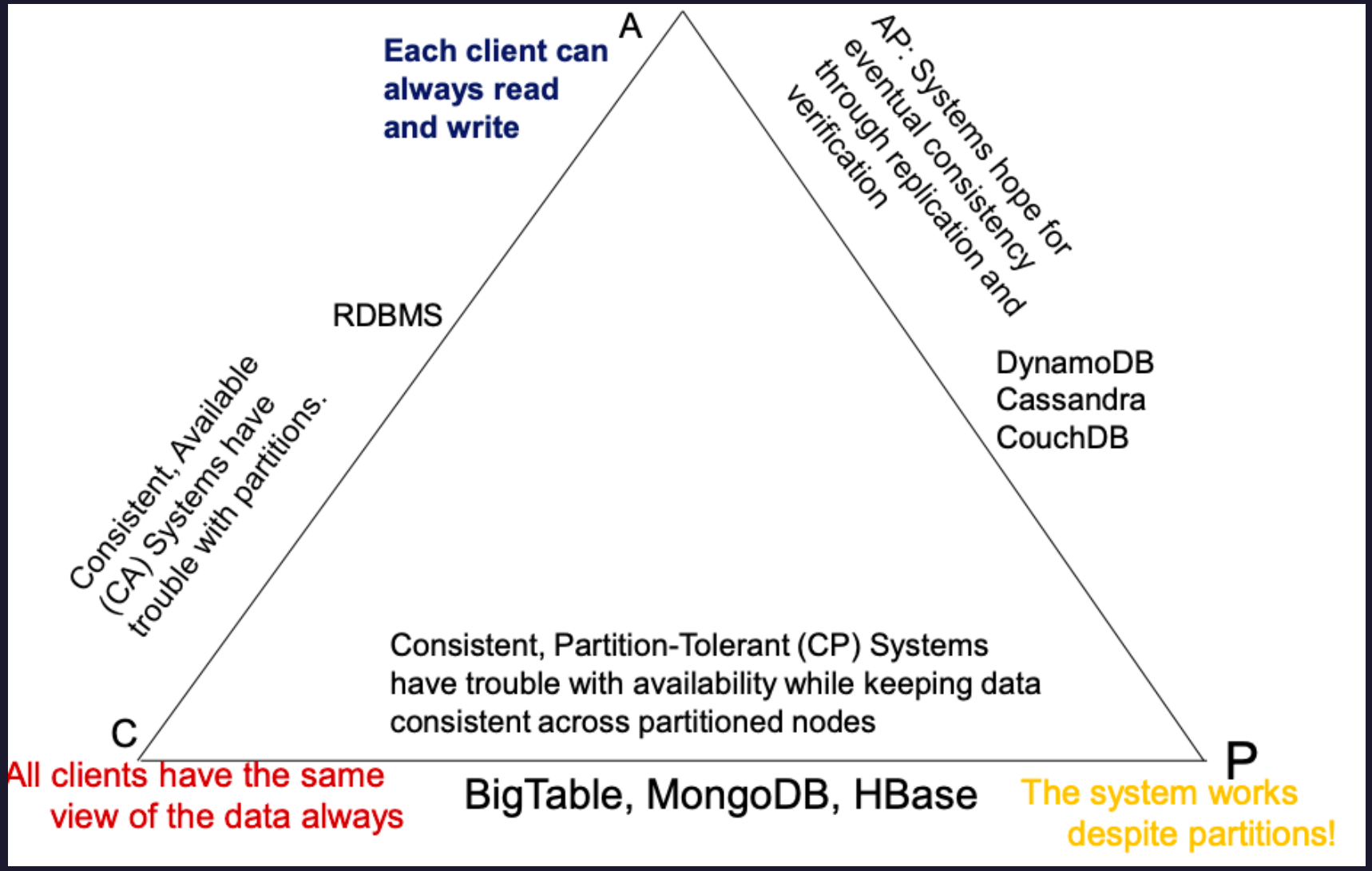
- Consistency:
  - All replicas contain the same version of data
  - Client always has the same view of the data (no matter what node)
- Availability
  - System remains operational on failing nodes
  - All clients can always read and write
- Partition tolerance
  - System works well across physical network partitions

- Can any system support all three of Consistency, Availability, and Partition Tolerance?

Any distributed system can satisfy any two of these guarantees at the same time **but not all three**

Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

CAP Theorem



- Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Why CAP Theorem is important

- The future of databases is **distributed** (Big Data Trend, NoSQL, ...)
- CAP theorem describes the **trade-offs** involved in distributed systems
- A proper understanding of CAP theorem is essential to **making decisions** about the future of distributed database **design**
- Misunderstanding can lead to **erroneous or inappropriate** design choices

- Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Types of Consistency

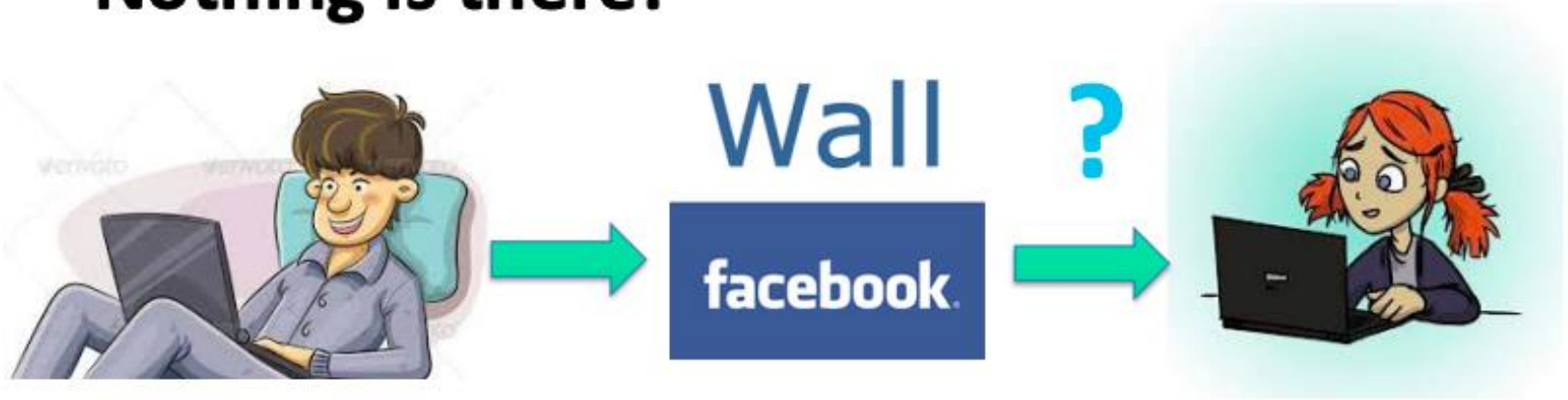
- Strong Consistency
  - After the update completes, **any subsequent access** will return the **same** updated value.
- Weak Consistency
  - It is **not guaranteed** that subsequent accesses will return the updated value.
- **Eventual Consistency**
  - Specific form of weak consistency
  - It is guaranteed that if **no new updates** are made to object, **eventually** all accesses will return the last updated value (e.g., *propagate updates to replicas in a lazy fashion*)

- Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Eventual Consistency : Facebook example

- Bob finds an interesting story and shares with Alice by posting on her Facebook wall
- Bob asks Alice to check it out
- Alice logs in her account, checks her Facebook wall but finds:

  - **Nothing is there!**



- Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Eventual Consistency : ATM example

- In design of automated teller machine (ATM):
  - Strong consistency appear to be a natural choice
  - However, in practice, **A beats C**
  - Higher availability means **higher revenue**
  - ATM will allow you to withdraw money *even if the machine is partitioned from the network*
  - However, it can put **a limit** on the amount of withdraw. The bank might also charge you a fee when an overdraft happens

- Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

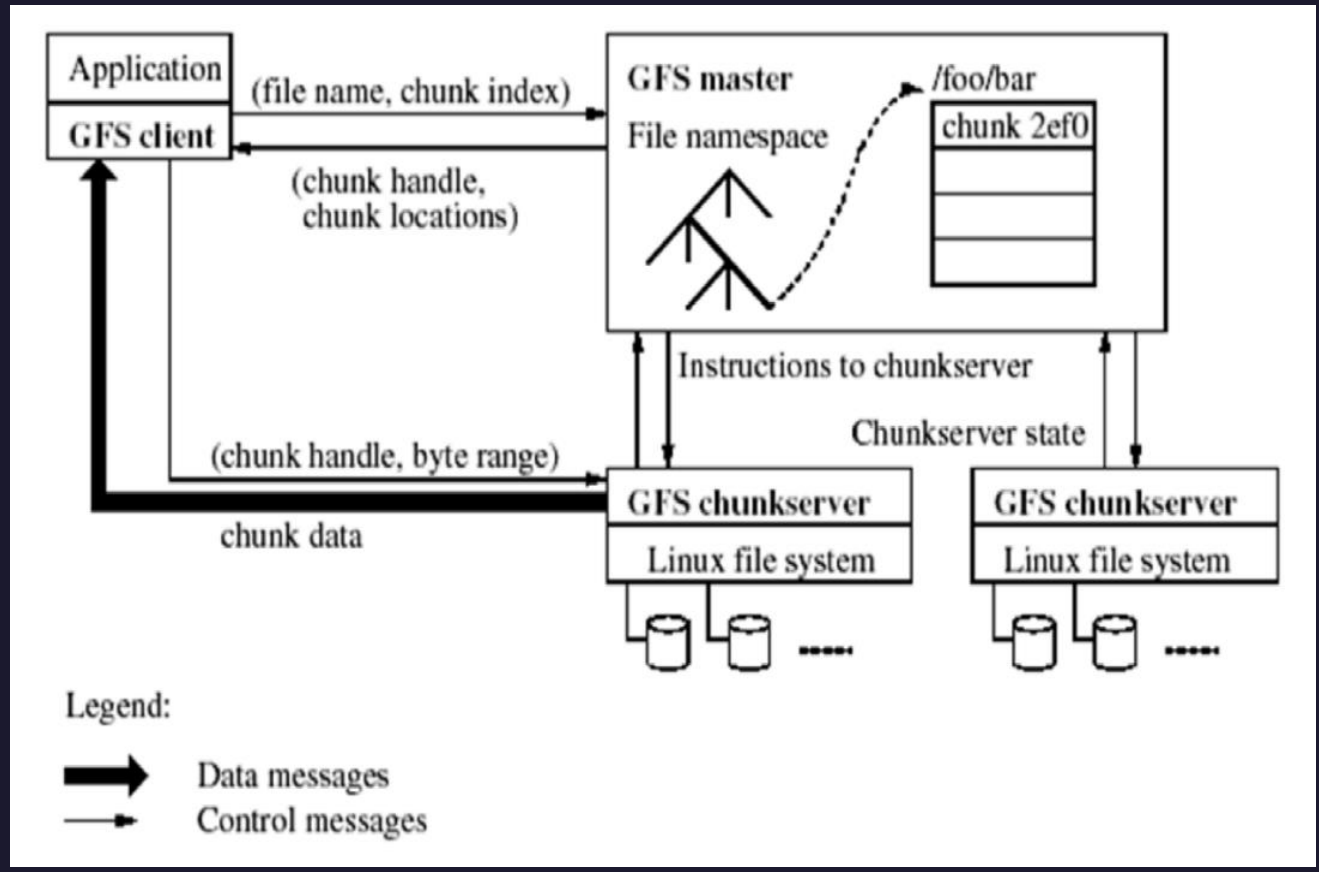
Google File System

- GFS is a **scalable distributed file system** for **large distributed data-intensive applications**. It provides **fault tolerance** while running on inexpensive **commodity hardware**, and it delivers high aggregate performance to a large number of clients.
- Design goals/priorities :
  - Design for big-data workloads : Huge files, mostly appends, concurrency, huge bandwidth
  - Design for failures
- Architecture: one master, many chunk (data) servers :
  - Master stores metadata, and monitors chunk servers
  - Chunk servers store and serve chunks
- GFS Workload Characteristics :
  - Files are huge by traditional standards : Multi-GB files are common
  - Most file updates are appends :
    - Random writes are practically nonexistent
    - Many files are written once, and read sequentially
  - High bandwidth is more important than latency
  - Lots of concurrent data accessing : E.g., multiple crawler workers updating the index file

Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Google File System

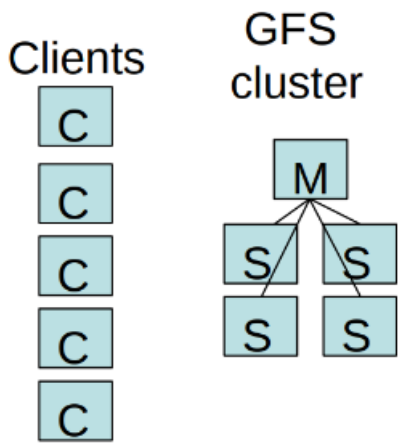
- Guarantees by GFS :
  - File namespace mutations (e.g., file creation) are **atomic.**
  - Record append **at least once atomically.**
- Additional operation: **record append**
  - Record append allows multiple clients to append data to the same file concurrently while **guaranteeing the atomicity of each individual client's append**.
  - Frequent operation at Google
  - Merging results from multiple machines in one file (Map/Reduce)
  - Logging user activity, site traffic
  - Order doesn't matter for appends, but atomicity and concurrency matters
  - semantic: **at least once atomically**



Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

GFS Architecture

- A GFS cluster
  - A single master (replicated later)
  - Many chunkservers
  - Accessed by many clients
- A file
  - Divided into fixed-sized chunks (similar to FS blocks)
  - Labeled with 64-bit unique global IDs (called handles)
  - Stored at chunkservers
  - 3-way replicated across chunkservers
  - Master keeps track of metadata (e.g., which chunks belong to which files)



Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

The GFS Master

- A process running on a separate machine
  - Initially, GFS supported just a single master, but then they added master replication for fault-tolerance in other versions/distributed storage systems
- Stores all metadata
  - File and chunk namespaces
    - Hierarchical namespace for files, flat namespace for chunks
  - File-to-chunk mappings
  - Locations of a chunk's replicas

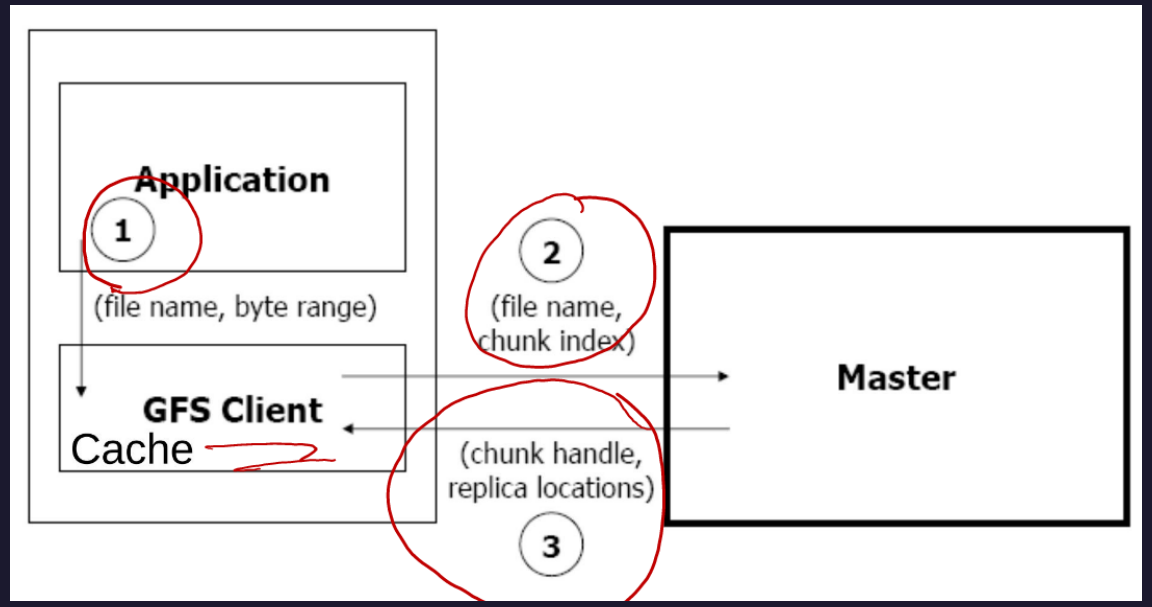
Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

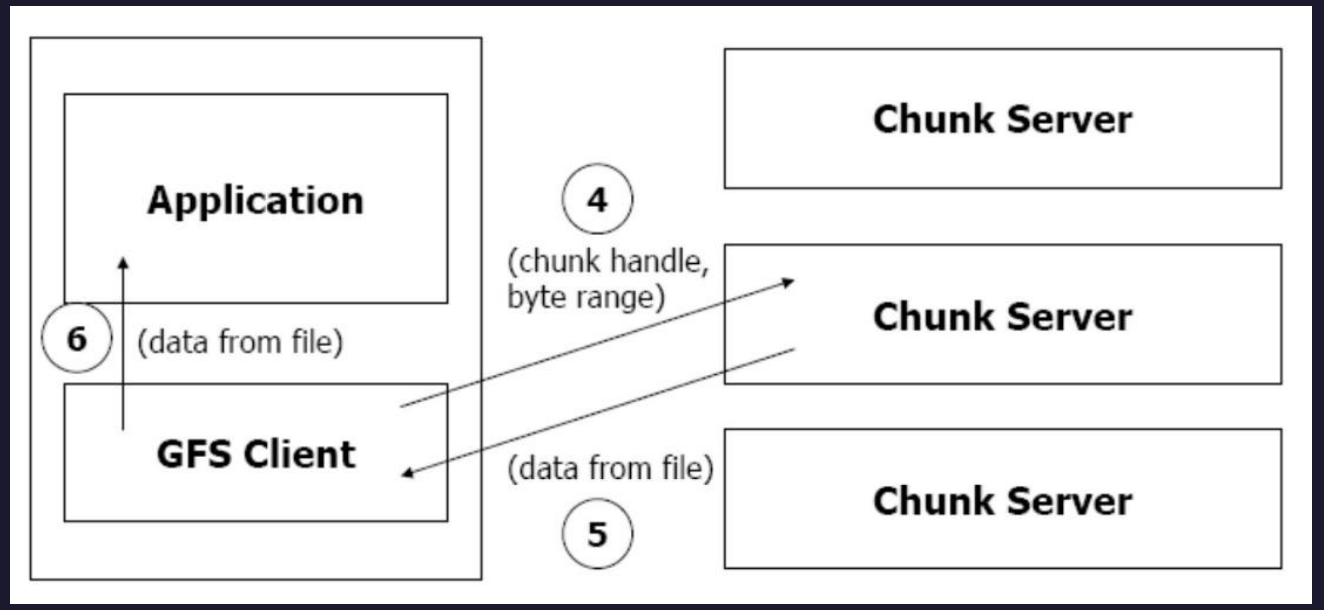
GFS Master <--> chunkserver

- Master and chunkserver communicate regularly (heartbeat):
  - Is a chunkserver down?
  - Are there disk failures on a chunkserver?
  - Are any replicas corrupted?
  - Which chunks does a chunkserver store?
- Master sends instructions to chunkserver:
  - Delete a chunk
  - Create new chunk
  - Replicate and start serving this chunk (chunk migration)
    - Why do we need migration support?

Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

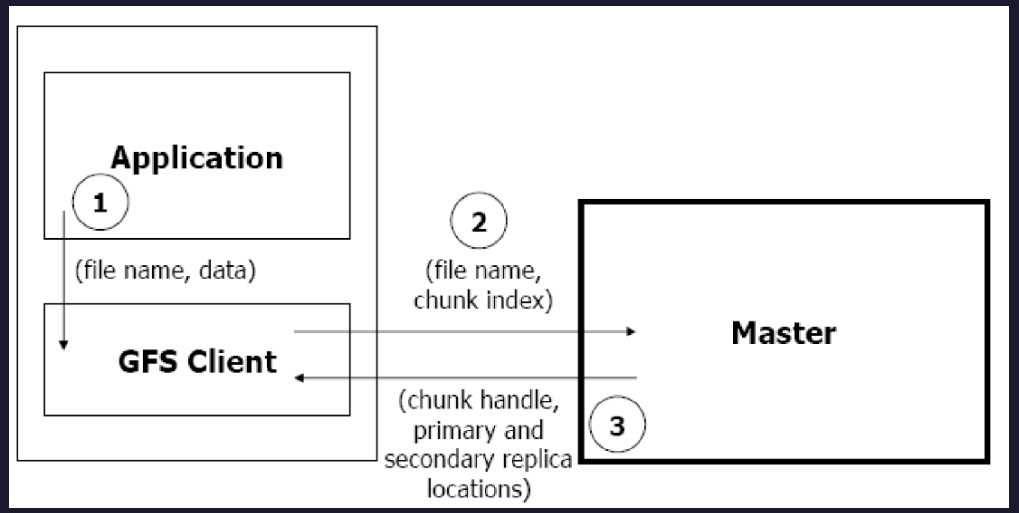
GFS Operations – Read

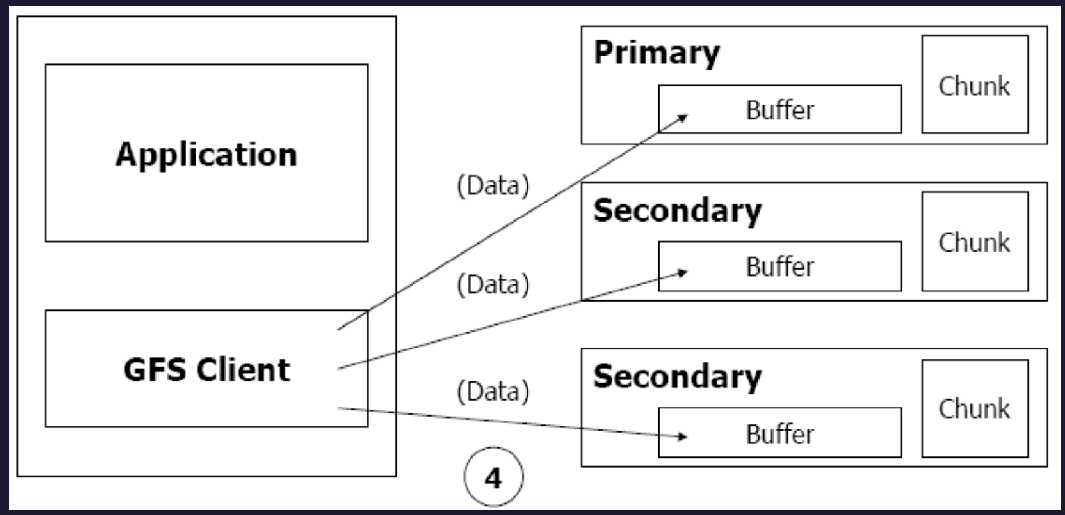


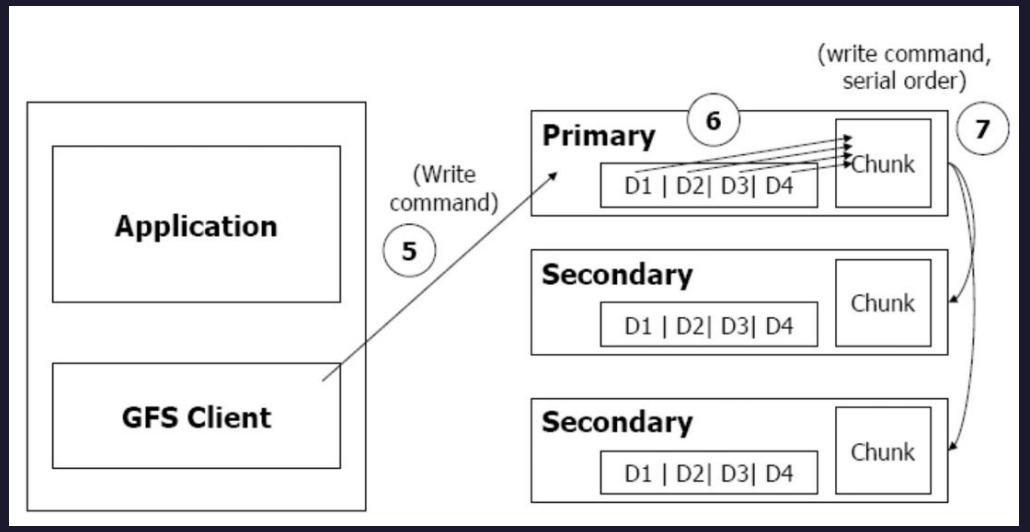


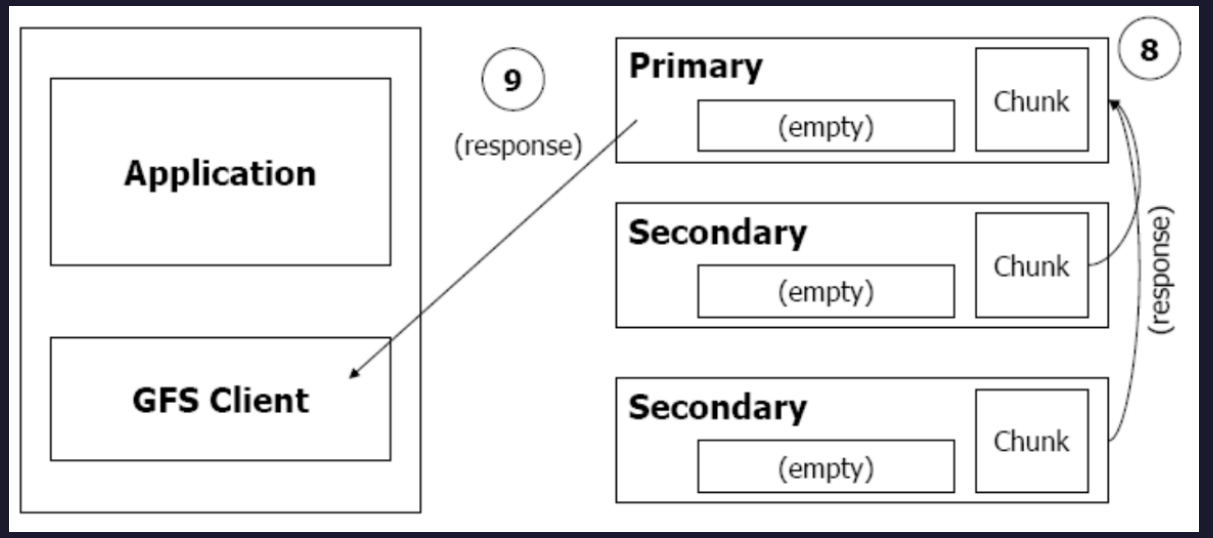
Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

GFS Operations – Write









Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

GFS Operations – Record Append

1. Application originates record append request.
2. GFS client translates request and sends it to master.
3. Master responds with chunk handle and (primary + secondary) replica locations.
4. Client pushes write data to all locations.
5. Primary checks if record fits in specified chunk.
6. If record does not fit, then:
   - The primary pads the chunk, tells secondaries to do the same, and informs the client of the inability/error.
   - Client then retries the append with the next chunk.
     - The primary chunk server for the next chunk may not be the same as this primary! So client has to retry.
7. If record fits, then the primary:
   - appends the record at some offset in chunk,
   - tells secondaries to do the same (specifies offset),
   - receives responses from secondaries,
   - and sends final response to the client.

Source : Distributed Systems, Prof. Kishore Kothapalli, Monsoon 2022

Hadoop Distributed File System (HDFS)

- **Hadoop** is an open-source software framework that is used for storing and processing large amounts of data in a distributed computing environment.
- Two main components :
  - HDFS : for file storage
  - MapReduce : for processing data
- The Hadoop Distributed File System (HDFS) is an open source implementation of the GFS architecture.
- One Goal of HDFS is “Moving Computation is Cheaper than Moving Data”

References

- https://www.ibm.com/topics/cloud-computing
- https://www.ibm.com/blog/object-vs-file-vs-block-storage/
- https://aws.amazon.com/compare/the-difference-between-block-file-object-storage/
- https://www.techtarget.com/searchstorage/definition/distributed-file-system-DFS
- https://www.oracle.com/big-data/what-is-big-data/
- https://pdos.csail.mit.edu/6.824/papers/gfs.pdf

Thank you