International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 10 | Oct 2026 www.irjet.net p-ISSN: 2395-0072

Understanding Cloud and Data Storage

Dr. Pranjali Patil[1], Jayshree Singh[2], Bhushan Patil[3], Mahesh Dubey[4]

[1] Assistant Professor, [2 3 4] Students, Department of MCA, D. Y. Patil Deemed to be University, School of Humanities & Sciences, Belapur, Maharashtra, India
Corresponding Author: Jayshree Singh

Abstract - *Cloud storage lets people and organisations keep information on remote infrastructure and reach that information over a network. Choosing a storage design requires more than just picking a provider. The structure of the data how often the data will be accessed, the required response time, security needs and the overall cost all need to be taken into account. This paper reviews three used storage models block storage, file storage and object storage and explains a typical path from uploading data to protecting it and managing its lifecycle. The paper also discusses encryption, identity management, access controls, replication, backup practices and the risks that come from network dependence and wrong permissions. The paper relies on a review of established literature and a conceptual comparison instead of a controlled performance test Its purpose is to give a foundation for students and early-career computing professionals who are evaluating cloud storage choices. I find that cloud storage offers flexibility.*

Keywords: cloud storage, block storage, file storage, object storage, data protection, lifecycle management, storage cost.

1. Introduction

Digital information is often kept on laptops, removable drives, local servers or other physical media. These arrangements can be useful. The information may become unavailable if a device is damaged, lost or suffers a hardware failure. They can also make it difficult to share and access information from locations in a controlled way.

I have seen how cloud storage can simplify data sharing across teams.

Cloud storage places data on systems that are maintained in data centres. A user or application sends information over a network. Later gets it back through an authorised interface. Depending on how the service's designed the provider may spread data across many devices or systems to make the system more resilient. Research on distributed storage, such as the Google File System and Dynamo shows how large-scale systems manage data across machines and can tolerate component failures [1] [2].

Cloud storage is not a design. Different applications need mixes of throughput, latency, sharing, durability and cost. For example a database volume, a shared team folder and a large archive may each be better served by a storage model. The main objectives of this paper are to explain these models outline an ingestion workflow review common security controls and discuss the practical trade-offs and limitations.

2. Literature Review

Work on distributed storage has shown how information can be spread across machines while still staying available even when some parts fail. The Google File System describes a distributed file-system architecture meant for data workloads [1]. Dynamo presents an available key-value storage design and talks about the trade-offs involved in spreading data across a system [2]. These works give background for understanding the principles used in modern storage services.

Cloud-computing literature also points out the benefits of accessing computing resources as services while noting that performance, availability and cost depend on how the service's implemented and used [5]. Research on cloud security points to confidentiality, access management and privacy as concerns [3] [4]. In practice technical safeguards must be paired with configuration and operational oversight.

2.1 Main Storage Models

Block storage divides data into blocks and presents them as a storage volume. This model is often used where applications need control over a volume and consistent read-write behaviour such as virtual-machine disks and many database deployments.

File storage organises information in files and directories. The familiar folder hierarchy makes it suitable for documents, departmental folders and applications that rely on file-system conventions.

Object storage keeps each item as an object that has associated metadata and an identifier. Applications usually manage objects through an API. This model is often used for backups, media assets, logs and large collections of information. These descriptions are general; actual capabilities and performance differ by provider and configuration.

3. Methodology

This paper follows a qualitative literature-based approach. It brings together concepts from the cited work on distributed systems, cloud computing and cloud security then compares storage models by looking at their typical data organisation and use cases. The upload workflow is shown as a model to explain the main stages that a system can implement.

No original benchmark, survey or controlled experiment was carried out. As a result the comparisons in this paper are descriptive, than measured findings. Latency, durability and pricing are not properties of a storage category; they depend on service-level design, network conditions, workload and provider terms.

3.1 Conceptual Data-Ingestion Workflow

Fig. 1 shows the four stages a file usually goes through after it is uploaded. First, the client sends the data over a protected, authenticated connection, normally HTTPS/TLS. Second, the data is encrypted, either by the client before upload or by the service when it is saved. The keys used for this must be kept safe. Third, the service may keep extra copies or use erasure coding so that the data survives a failure. This improves availability, but it is not a substitute for separate backups. Fourth, a policy decides which storage class holds the data and may move it later as it gets older or is used less.

Stage	Typical activity	Purpose and consideration
1. Transfer	The client sends data through an authenticated, protected connection, commonly using HTTPS/TLS.	Helps protect information in transit; application authentication and validation are also required.
2. Encryption	Data may be encrypted before upload or by the storage service when written.	Reduces exposure if stored media is accessed improperly; keys must be managed securely.
3. Resilience	The service may use replication, erasure coding, or other mechanisms.	Can improve availability or recovery, but replication alone does not replace independent backups.
4. Placement and lifecycle	A policy places data in a suitable storage class and may later transition it based on age or access pattern.	Can align cost with expected use; retrieval delay and access charges should be considered.

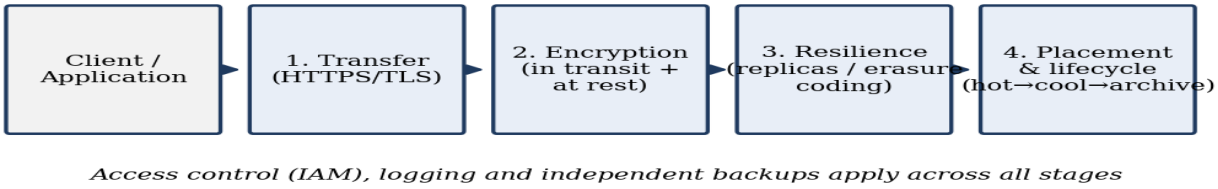


Fig. 1. Conceptual stages in handling uploaded data. Order and implementation differ among providers.

3.2 Storage Classes

Most services sell storage in tiers. "Hot" storage is for data that is used often, so it is quick to read but costs more per gigabyte. "Cool" or "warm" storage is cheaper to keep and suits data that is used now and then. Archive storage is for data that has to be kept but is hardly ever opened. Getting it back can take a while and may cost extra.

Which tier is right depends on how often the data is read, how long it must be kept, how quickly it must be restored, and what the provider charges. A cheaper tier does not always mean a smaller bill, since retrieval, request and transfer charges can cancel out the saving.

4. RESULTS AND DISCUSSION

4.1 Comparison of Storage Types

Table I summarises common patterns. It is not a benchmark result. Before using latency or cost figures for a real decision, the target provider, region, configuration, data size and workload should be tested.

TABLE I. General comparison of block, file and object storage

Model	Organisation	Typical use	Key trade-off
Block	Addressable blocks shown as a volume	VM disks; database volumes	Fine-grained volume control; cost and management depend on the service.
File	Files in folders and directories	Shared folders; collaborative applications	Convenient shared hierarchy; speed depends on file operations and configuration.
Object	Objects with identifiers and metadata	Backups; media; logs; unstructured data	Scales well for API access; poor fit for software needing file-system or block behaviour.

It would be misleading to give one latency figure for each category without saying how it was measured. Block storage can respond quickly when it is set up well, while file and object services reach the data in different ways. Even so, the order can change with network distance, workload, caching, tier and the provider's own design. Any number used to make a decision should come from repeatable measurements or from the provider's documentation, and the source should be cited.

4.2 A Practical Selection Guide

A simple way to choose is to start from the workload and ask five questions. Does the application need a disk it can format and control exclusively? Do several users need a shared folder tree with permissions? Is the data large, unstructured and reached through an API? How often will it be read? How quickly must it be restored after a failure? Table II applies these questions to four common student and small-organisation scenarios.

TABLE II. Scenario-based selection guide

Scenario	Suggested model	Reason
Relational database for a college admission system	Block	Needs a volume with steady read/write behaviour; add snapshots and tested restores.
Shared project folders for a department	File	Users expect a folder hierarchy and per-folder permissions.
Recorded lectures and scanned records	Object, with tiers	Large, rarely changed files; older items can move to a cooler tier.
Application logs kept for audit	Object, cool or archive tier	Written once, read rarely; retention period must be checked first.

4.3 Worked Cost Example

Cost depends on more than stored capacity: providers can bill for capacity, requests, retrieval, replication choices and outbound transfer. The following example uses assumed prices only, chosen to keep the arithmetic simple; they are not quotes from any provider and real prices must be taken from the provider's current India price list. Suppose a college stores 10 TB (10,000 GB) of lecture recordings and records, the hot tier costs Rs. 2.0 per GB-month, the cool tier costs Rs. 1.0 per GB-month, and retrieving data from the cool tier costs Rs. 2.5 per GB.

Keeping all the data hot costs 10,000 × 2.0 = Rs. 20,000 per month. In the cool tier the holding cost is 10,000 × 1.0 = Rs. 10,000, and if a fraction f of the data is read back each month the total is 10,000 + 25,000f. If 10% is retrieved, the cool tier costs Rs. 12,500, a saving of Rs. 7,500 (37.5%). Setting 10,000 + 25,000f = 20,000 gives f = 0.4, so with these prices the cool tier stops being cheaper once more than about 40% of the data is read every month (Fig. 2). The same method works for any provider: replace the assumed prices with the current price list and add request and transfer charges.

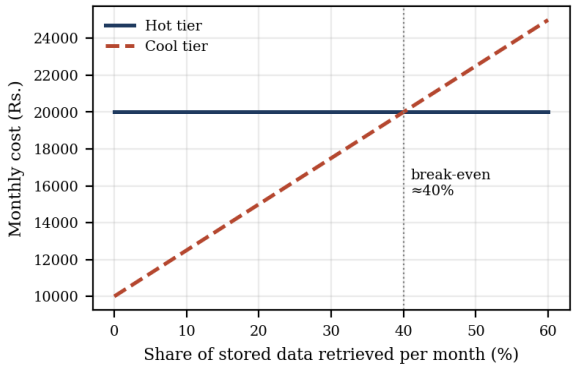


Fig. 2 (line graph in the Word file). Monthly cost against share of data retrieved, using the assumed rupee prices above (calculated, not measured). Hot tier is a flat Rs. 20,000; cool tier rises from Rs. 10,000 and crosses it at about 40%.

Before setting lifecycle rules, an organisation should check how long it must keep the data, how fast it needs to recover it, whether the tier has a minimum storage period, and whether old data might suddenly be needed. An archive tier can be a good way to keep data for years, yet it is a poor choice for anything that must be restored straight away. Total cost, response time, availability, portability and the effort of running the system should be weighed together, because the lowest listed price is seldom the cheapest or the best fit overall.

4.4 Security Controls

A wrong permission setting can leave a bucket, folder or file open to people who should not see it, and encryption cannot fix a policy that is too loose. A sound security process therefore includes least-privilege permissions, multi-factor authentication

where suitable, proper key management, audit logs and regular reviews of the settings [3], [4]. Table III lists the main controls and what each one cannot do.

TABLE III. Security controls and their limits

Control	Helps with	Limit or concern
Encryption in transit	Protects data moving between client and service.	Does not decide who may access the data.
Encryption at rest	Reduces exposure if storage media is accessed improperly.	Key access, rotation, recovery and separation of duties still matter.
Identity and access management	Restricts which identities can do which actions.	A wrong policy or stolen account can still cause exposure.
Replication and backups	Supports resilience and recovery.	Replicas can copy accidental deletion or malicious change; keep independent, tested backups.

4.5 Limitations

Network dependence is the main practical limit: a broken connection or an unavailable cloud endpoint can delay access, so applications with strict continuity needs may require local caching, offline procedures or a tested recovery plan. Cloud use also raises questions of compliance, data location, vendor lock-in and recovery testing; organisations should confirm that a service meets their legal and contractual duties and should periodically check that data can be restored within the required time. For Indian users, data location deserves special attention. The Digital Personal Data Protection Act, 2023 governs personal data, and some sectors, such as payments, have their own rules on where data may be stored, so students building real applications should check the current rules and choose a provider region accordingly. Major providers operate regions in India, which can also reduce latency for local users, although this should be confirmed by testing. Finally, this study is limited by its design: it relies on published literature and a calculation with assumed prices, so its conclusions should be confirmed with measurements on the chosen platform.

5. CONCLUSION AND FUTURE WORK

Block, file and object storage meet different needs, so the choice should start from the workload and not from the idea that one model is best. How the data is accessed, the speed required, the cost, the resilience and the way the application uses the data all need to be considered. Our cost example shows that a cheaper tier saves money only while retrieval stays low, so lifecycle rules should be checked against real access logs. Encryption, access control, replication, backups and lifecycle policies all add protection and efficiency, provided they are set up carefully and reviewed from time to time.

Future work, which we plan to carry out as a hands-on extension of this paper, can compare selected services using a defined dataset and workload, reporting the provider, region, tier, network conditions, request sizes, number of trials and measurement method. Further studies could also test automated lifecycle recommendations, deduplication and compression, and measure their effect on cost, retrieval time, data integrity and compliance…

REFERENCES

[1] S. Ghemawat, H. Gobioff and S.-T. Leung, “The Google File System,” in Proc. 19th ACM Symp. Operating Systems Principles (SOSP), 2003, pp. 29–43, doi: 10.1145/945445.945450.

[2] G. DeCandia et al., “Dynamo: Amazon’s highly available key-value store,” in Proc. 21st ACM SIGOPS Symp. Operating Systems Principles (SOSP), 2007, pp. 205–220, doi: 10.1145/1323293.1294281.

[3] P. Yang, N. Xiong and J. Ren, “Data security and privacy protection for cloud storage: A survey,” IEEE Access, vol. 8, pp. 131723–131740, 2020, doi: 10.1109/ACCESS.2020.3009876.

[4] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," J. Netw. Comput. Appl., vol. 34, no. 1, pp. 1–11, 2011, doi: 10.1016/j.jnca.2010.07.006.

[5] M. Armbrust et al., "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010, doi: 10.1145/1721654.1721672.

[6] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in Proc. 7th Symp. Operating Systems Design and Implementation (OSDI), 2006, pp. 307–320.