

Rule-Based Snapshot Verification and Recovery Framework for Distributed Cloud Storage

Ghassan Adnan Hammoodi Al-Bdairi¹, Saad Talib Hasson²

¹ Department of Information Networks, College of Information Technology, University of Babylon, Hilla, Iraq, ghassan782023@gmail.com

² Department of Information Networks, College of Information Technology, Information, University of Babylon, Hilla, Iraq, Saad.aljebori@uobabylon.edu.iq

ARTICLE INFO

Article history:

Received: 16 /06/2026

Revised form: 14 /07/2026

Accepted : 16 /07/2026

Available online: 30/09/2026

Keywords:

Cloud storage, Cloud-of-clouds, Data integrity, Snapshot verification, Data recovery, File synchronization, Fault tolerance

ABSTRACT

Rapid proliferation of cloud storage services has enlarged the issue of protecting user data from

unauthorized modifications, deletions, or constraints by the provider in any distributed system. The current state-of-the-art Cloud-of-Clouds solutions improve both availability and integrity but do not provide an operator-controlled method to verify data across different cloud providers and restore a Metacloud to a previously trusted state after unauthorized modifications. In this paper, we present and implement an Intelligent Snapshot Verification and Recovery Framework (ISVRF) as a web service that integrates four commercial cloud storage providers operating under different legal jurisdictions. The proposed framework uses checksum-based snapshot verification, three synchronization modes (Primary Source, Merge, and Bi-Directional), and incorporates a freeze-and-rollback recovery mechanism. An evaluation of its efficacy is also provided through a snapshot performance workbench tested with different file sizes and varying numbers of providers. In 5 experiments (mean $\pm$ SD), the Primary Source strategy consistently achieved the lowest mean latency across all scenarios (e.g., 864.1 ± 60.8 ms for medium files with three providers). The maximum latency increase for the Merge and Bi-Directional strategies was 9.58%. Statistically significant differences between the fastest and second-fastest strategies were found in only 1 out of 9 scenarios (paired t-test, $p = 0.0013$). Increasing the providers from 3 to 5 resulted in a latency variation of not more than about 7%. These results show that resilient snapshot verification and recovery over diverse jurisdictions can be achieved with relatively modest and primarily transfer-bound overhead, making the framework suitable for periodic data protection in user and organizational contexts where the risk of data loss may be due to factors at the jurisdictional or provider level. MSC..

<https://doi.org/10.29304/jqcs.2026.18.33004>

Introduction

Modern-day digital infrastructure involves remote storage and access of data over the internet through a third-party provider, as opposed to storing data on local hardware. The size of the global cloud storage market was estimated to be approximately USD 98.8 billion in 2023. A ten-year projection places it at well over USD 777 billion in 2033, reflecting an approximate compound annual growth rate of 22% [8]. A single provider can deliver very high levels of availability and fault tolerance beyond what one provider can offer, and hence, the cloud-of-clouds paradigm is being investigated to realize the same. In the face of random failures or outages of some providers, several systems have since been built on similar principles.

Yet the above techniques inherently secure the integrity of stored data against accidental corruption, not willful or unauthorized changes. For example, a provider may have to suspend or terminate an account under legal, political, or commercial duress, and an account that is compromised but authenticated can then be used to attack and delete data that the storage layer has conscientiously replicated. These are the issues with current cloud-of-clouds architectures. While encryption and erasure coding along with Byzantine quorums are integrated into DepSky to protect against untrustworthy providers, the main focus of DepSky is on confidentiality and read/write availability, rather than on enabling user-controlled recovery [3]. HAIL assures cryptographically that a file is preserved and can be accessed; however, it prioritizes verification over synchronization or repair of a corrupt state [4]. Hybris handles Byzantine providers by using a trusted metadata layer, but it does not support point-in-time rollback [5]. However, with the current advancements, the ability to detect unauthorized changes and roll back a Metacloud to a previously trusted state is still mostly neglected [1].

The first one allows for redundancy and erasure coding between providers to increase availability and remove the possibility of vendor lock-in. An example of this approach is RACS [2], and DepSky [3]. However, these systems do not distinguish between legitimate updates and malicious alterations; they do not provide a means for reversing the latter. The second approach focuses on cryptographic integrity and proofs of retrievability. For example, HAIL [4] is good at corruption detection but assumes a server-side coding model that does not fit well with opaque commercial provider APIs and does not facilitate recovery. The third approach provides consistent file-sharing semantics over eventually-consistent clouds. An example of this approach is SCFS [6] and the consistency models surveyed by Viotti and Vukolić [7]. They are concerned more with the problem of adversarial data loss than consistency. Recent studies carried out between 2020 and 2025 have increased the efforts of multi-cloud integrity auditing and security [9],[18], but they are more focused on detection than recovery. Throughout all three directions, the specific challenge of reversing unauthorized deletions or modifications under operator control remains unresolved.

This study aims to bridge that gap. We are not aware of any research that currently exists on integrating checksum-based snapshot verification between different commercial cloud providers and a recovery mechanism held by the operator, which can discern between legitimate and illegitimate changes and thus reverse the latter. The only previous work on this very issue — our own Metacloud concept — was put forward at a conceptual stage using flowchart algorithms; it has not been implemented, integrated with real providers, or put to quantitative evaluation.. [1].

The main contributions of this paper are as follows:

- (1) We propose the Intelligent Snapshot Verification and Recovery Framework (ISVRF), a Metacloud architecture that couples a provider-abstraction layer with checksum-based snapshot verification to detect added, modified, and deleted files across heterogeneous providers.
- (2) We implement the framework as a working web-based service that integrates four providers operating under distinct legal jurisdictions, and we introduce an operator-controlled freeze-and-rollback recovery mode that restores a trusted point-in-time state.
- (3) We design and compare three synchronization strategies — Primary Source, Merge, and Bi-Directional — together with a two-minute delete-delay safeguard that prevents irreversible loss from transient inconsistencies.
- (4) We evaluate the framework through a snapshot performance workbench, quantifying end-to-end latency, per-phase cost, strategy trade-offs, and provider-scaling behaviour under a range of workloads.
- (5) We provide the first empirical realisation and quantitative evaluation of the Metacloud concept that was previously described only conceptually [1].

The authors have previously published conceptual work [1] and it is important to separate the contributions of this paper from this previous conceptual work. The previous publication: (i) provided only algorithms at the flowchart level for a Metacloud operating mode, (ii) did not provide any implementation, (iii) did not include real provider APIs and (iv) did not provide any quantitative performance evaluation. The present paper builds on that conceptual foundation in four practical ways: (a) it implements a working web service with a four-layer architecture (Figure 1), (b) it integrates four different commercial cloud providers, each with a distinct legal jurisdiction, and measures end-to-end latency of the live APIs in these providers (Table 3), (c) it designs and statistically evaluates three synchronisation strategies in detail, using paired t-tests and Cohen's d effect sizes, and (d) it gives for the first time an empirical quantification of the end-to-end latency, per-phase cost and scaling behaviour. This means that the concept of Metacloud proposed in [1] has been first realized, implemented and tested empirically in this paper.

THE REMAINDER OF THIS PAPER IS ORGANIZED AS FOLLOWS. SECTION 2 REVIEWS RELATED WORK AND IDENTIFIES THE RESEARCH GAP. SECTION 3 PRESENTS THE PROPOSED FRAMEWORK, INCLUDING THE PROBLEM FORMULATION, THE EXPERIMENTAL WORKLOAD, THE ARCHITECTURE, THE IMPLEMENTATION, AND THE EVALUATION METRICS. SECTION 4

REPORTS THE EXPERIMENTAL RESULTS. SECTION 5 DISCUSSES THEIR INTERPRETATION, IMPLICATIONS, AND LIMITATIONS. FINALLY, SECTION 6 CONCLUDES THE PAPER AND OUTLINES FUTURE WORK.

2. RELATED WORK

The protection of user data stored in commercial cloud services has been studied along three broad directions: distributing data across several providers for availability, cryptographically verifying the integrity and retrievability of remotely stored data, and providing consistent file-sharing semantics over otherwise eventually-consistent clouds. This section reviews representative work in each direction critically, with attention to what each system guarantees and what it leaves unaddressed, in order to position the gap that motivates the present framework.

2.1 REDUNDANCY- AND ERASURE-CODING-BASED MULTI-CLOUD STORAGE

Abu-Libdeh Et Al. [2] Introduced Racs, A Proxy Layer Designed To Implement Raid-Like Erasure Striping Of User Data Across Multiple Service Providers, Thereby Mitigating Vendor Lock-In And Accommodating Provider Outages. Their Findings Illustrate That Striping Not Only Diminishes The Economic Burden Associated With Switching Providers But Also Enhances Availability In The Event Of A Single-Provider Failure. Nonetheless, Racs Primarily Focuses On Availability And Lock-In Issues Rather Than Addressing Intentional Tampering By Providers, Lacking Any Mechanisms For Detecting Or Reverting Unauthorized Alterations. In Contrast, Bessani Et Al. [3] Created Depsky, A Cloud-Of-Clouds Object Storage Solution That Integrates Encryption, Erasure Coding, And Byzantine Quorum Replication. This Approach Necessitates $N = 3f + 1$ Clouds (For Instance, Four Clouds Are Required To Withstand $F = 1$ Faulty Cloud), And The Cost-Optimized Variant Maintains The Financial Expenditure At Approximately Twice That Of Utilizing A Single Cloud. However, Depsky Is Primarily Aimed At Ensuring Confidentiality And Byzantine Fault Tolerance, Failing To Provide A User-Controlled Point-In-Time Freeze Or A Rollback Option For Changes Executed Through A Legitimate Yet Compromised Account. While Both Methods Enhance Availability And Manage Faulty Providers, They Do Not Adequately Deliver A Clear Verification-And-Recovery Mechanism For Reversing Unauthorized Deletions Or Modifications.

2.2 INTEGRITY AND BYZANTINE-FAULT-TOLERANT CLOUD-OF-CLOUDS STORAGE

Bowers et al. [4] introduced HAIL, a distributed cryptographic high-availability and integrity layer in which servers can prove to a client that a file remains intact and retrievable using compact proofs of only tens to hundreds of bytes, independent of file size, and which is robust against an active, mobile adversary. However, HAIL provides proofs of retrievability and integrity verification rather than synchronization across heterogeneous commercial providers or recovery of corrupted state, and it assumes a server-side coding model that is not directly applicable to opaque commercial provider APIs. Dobre et al. [5] proposed Hybris, described as the first robust hybrid cloud key-value store, which keeps metadata on a trusted private tier and replicates data across several public clouds, tolerating f Byzantine clouds while using as few as $f + 1$ clouds in the common case. Despite this efficiency, Hybris depends on a trusted private metadata tier and exposes a key-value abstraction, and it does not address user-initiated freeze or rollback of file state across providers. Although these systems advance integrity guarantees and Byzantine tolerance, they do not provide an operator-controlled recovery mode that restores a previous, trusted snapshot of the data.

2.3 CLOUD-BACKED FILE SYSTEMS AND CONSISTENCY

Bessani et al. [6] developed SCFS, a cloud-backed file system that provides strong consistency and near-POSIX semantics on top of eventually-consistent cloud services through a pluggable backplane that operates over a single cloud or a cloud-of-clouds and tolerates a single faulty cloud. However, SCFS focuses on consistent file sharing and durability rather than on an explicit verification-and-rollback procedure for reverting unauthorised changes. Viotti and Vukolić [7] presented a comprehensive survey that systematised more than fifty consistency models for non-transactional distributed storage, clarifying the trade-offs between consistency strength and availability. Nevertheless, as a survey it characterises consistency semantics rather than proposing a synchronisation-and-recovery framework, and it does not consider adversarial, provider-induced data loss. In contrast to these works, the threat addressed in the present paper is not weak consistency but the deliberate or accidental destruction of data by an authenticated actor or by the provider, which requires a different recovery primitive.

Finally, the Metacloud concept underlying the present system was first outlined by the authors in prior work [1], which described the two operating modes — synchronization and freeze — at a conceptual level, together with flowchart-level algorithms, but without an implementation, heterogeneous provider integration, or any quantitative evaluation.

2.4 RECENT ADVANCES (2020–2025)

More recent work has concentrated on data-integrity auditing in multi-cloud settings. Li et al. [9] surveyed integrity-auditing schemes spanning single-copy to multiple-replica storage, and Han et al. [10] surveyed blockchain-based integrity auditing for cloud data, both noting the high cost of probabilistic data-possession checking at scale. Witanto et al. [11] proposed a blockchain-based distributed integrity-verification protocol that employs multiple verifiers to raise the sampling rate without a proportional increase in cost, and a remote-integrity-checking protocol for multi-cloud environments was presented in [12]. Li et al. [13] and Song et al. [14] further combined blockchain-based integrity auditing with encrypted deduplication for cloud storage. However, these schemes verify whether data remain intact rather than reverting unauthorised changes, and they typically introduce a blockchain ledger or a third-party auditor as an additional trust component.

A second recent thread targets secure sharing, storage efficiency, and availability. A hybrid-cryptosystem scheme for secure multi-cloud data sharing was proposed in [15]; Bhavya et al. [16] combined multi-cluster deduplication with erasure coding to raise data availability; and Nachiappan et al. [17] optimized proactive recovery in erasure-coded cloud storage. Broader surveys of multi-cloud and hybrid-cloud security [18] confirm that preserving confidentiality and integrity across fragmented data remains a central open challenge. Nevertheless, these works pursue storage efficiency, confidentiality, or block-level recovery within a single erasure-coded system, and none provides an operator-controlled freeze-and-rollback of file state across heterogeneous commercial providers that distinguishes legitimate from illegitimate changes — reaffirming the gap addressed in this paper. Liangcheng Yu, et al. [19] presented paper describes a revolutionary protocol called Beaver, the first system that allows for "partial distributed snapshots" of large cloud systems without having to stop operations or affect user data traffic, which is the current mainstream approach in large cloud infrastructures. presents Beaver, the first practical partial snapshot protocol that ensures causal consistency under external traffic interference. Beaver presents a unique design point that tightly couples its protocol with the regularities of the underlying data center environment. By exploiting the placement of software load balancers in public clouds and their associated communication pattern, Beaver not only requires minimal changes to today's data center operations but also eliminates any form of blocking to existing communication, thus incurring near-zero overhead to user traffic. We demonstrate the Beaver's effectiveness through extensive testbed experiments and novel use cases. Nagaraju Unnava. [20] presented paper describes a transformative disaster recovery paradigm leveraging microservice state snapshots, declarative infrastructure patterns, and automated orchestration to achieve near-instantaneous recovery during catastrophic regional failures. The method treats infrastructure and application state as versioned, immutable artifacts, enabling deterministic reconstruction across heterogeneous cloud environments. Storage networking technologies enable continuous state synchronization across geographically distributed regions. Avais Ahmad, et al. [21] present research proposes an automated snapshot system to replicate on-prem data to the cloud, triggered by CI/CD pipeline or schema

changes—events that impact app behavior. Capturing snapshots at these points ensures the test environment mirrors the latest state. These snapshots build testing databases in the cloud. The method uses an IT automation engine to manage snapshot creation and volume replication, reducing manual effort. Evaluation confirmed data consistency and automation success. The system improves Dev/Test scalability and efficiency. Sagit Valeev, et al.[22] presented the optimizing of energy resources at all hierarchical levels, an up-to-date picture of the system state is necessary. This paper proposes constructing a snapshot of the system's state, which allows for an assessment of system performance within selected criteria and the selection of appropriate solutions within the entire control system. Procedures for constructing a snapshot are discussed. An example of optimizing the energy consumption of a control system based on snapshot analysis is provided. Santhosh S., et al. [22] presented study explores minimizing data recovery time in dynamic environments. The research proposes using "greedy algorithms" to prioritize smart snapshots based on data volume, importance, and update rate, and employing incremental restore to reduce network resource consumption and improve real-time response speed.

Table 1. Comparison of representative multi-cloud storage approaches

Reference	Year	Method / Model	Evaluation setup	Key result	Limitation
RACS [2]	2010	RAID-like erasure striping across providers	Analytical + multi-provider prototype	Reduces vendor lock-in cost; tolerates single-provider outage	No detection/rollback of unauthorized changes
DepSky [3]	2011/13	Encryption + erasure coding + Byzantine quorums	S3, Azure, Nirvanix, Rackspace (4 clouds)	Tolerates f faulty clouds ($n=3f+1$); cost $\leq \sim 2 \times$ single cloud	No user-controlled freeze / rollback
HAIL [4]	2009	Cryptographic integrity + retrievability proofs	Formal model + cryptographic evaluation	Compact proofs (tens-hundreds of bytes); mobile-adversary robust	Verification only; no cross-provider sync/recovery
Hybris [5]	2017	Hybrid: trusted metadata tier + public clouds	Multi-cloud key-value deployment	Tolerates f Byzantine clouds using $f+1$ (common case)	Requires trusted metadata tier; no rollback mode
SCFS [6]	2014	Cloud-backed file system, near-POSIX	EC2/Rackspace/Azure/Elastic hosts (EU)	Strong consistency over eventual clouds; tolerates 1 faulty cloud	No explicit verification-and-rollback recovery

Reference	Year	Method / Model	Evaluation setup	Key result	Limitation
Metacloud [1]	2022	Two-mode (sync + freeze) concept	Conceptual / flowchart algorithms	Conceptual framework for provider-tamper resilience	No implementation or quantitative evaluation
Witanto et al. [11]	2023	Blockchain-based distributed integrity verification (multi-verifier)	Multi-cloud, performance analysis	Higher verification sampling rate at lower per-verifier cost	Verifies integrity only; no rollback; adds blockchain trust
Nachiappan et al. [17]	2023	Proactive recovery in erasure-coded cloud storage	Erasure-coded distributed storage	Optimised proactive block recovery and availability	Block-level recovery in one EC system; no cross-provider freeze/rollback
Liangcheng Yu, et al.	2024	Partial Snapshot; SLB Routing; Optimistic Verification	Baselines: Compared against classical Polling and Lai-Yang snapshot algorithms.	Zero Overhead and Standard network delays are predictable enough	False Retries; Topology Bound
Nagaraju Unnava	2025	Immutable State Snapshots; Declarative Infrastructure; Continuous Sync	Scenario: Evaluated against catastrophic regional cloud failures; Tested across heterogeneous (multi-provider) cloud environments.	Near-Instant Recovery; Deterministic Reconstruction	Network Dependency; State Overhead
Avais Ahmad, et al.	2025	Triggered Replications specific triggers (CI/CD pipelines or database schema changes); Automated IT Engine	Validated across deployment pipelines testing continuous development (Dev/Test) workflows; Metrics	High Efficiency; Consistent State Tracking	Pipeline Coupling; Bandwidth & Storage Costs
Santhosh S.,et al.	2025	Greedy Prioritization;	Evaluated inside highly dynamic, rapidly changing	Faster Recovery;	Local Optimization

Reference	Year	Method / Model	Evaluation setup	Key result	Limitation
		Multi-Metric Weighting	computational environments; Measured based on total data recovery time, network bandwidth consumption, and real-time responsiveness	Resource Efficiency	Trap; High Assessment Overhead

2.5 Research Gap

Taken together, prior cloud-of-clouds research has concentrated on availability through striping [2], Byzantine tolerance and confidentiality [3,5], cryptographic integrity proofs [4], and consistent file-sharing semantics [6,7]. Existing snapshot and checkpoint mechanisms typically operate within single storage systems and lack the cross-provider coordination necessary for multi-cloud environments [23]. File synchronization protocols often rely on timestamp-based change detection, which can be unreliable in distributed systems with clock skew [24].

However, none of these systems provides a user-controlled, snapshot-based verification mechanism coupled with an explicit freeze-and-rollback recovery mode that can distinguish legitimate changes from illegitimate ones and reverse the latter — precisely the threat posed by an authenticated-but-compromised account or by a provider or state actor deleting or altering data. The only prior treatment of this specific problem [1] remained conceptual and was never implemented or measured. In contrast to prior works, the proposed framework implements checksum-based snapshot verification across four heterogeneous commercial providers, adds an operator-controlled freeze-and-rollback recovery procedure, and evaluates the resulting system quantitatively.

In contrast to prior works, the proposed framework implements checksum-based snapshot verification across four heterogeneous commercial providers and an operator-controlled freeze-and-rollback recovery mode, evaluated quantitatively. The freeze-and-rollback mechanism is intended to mitigate accidental or unauthorised deletion and modification across all connected providers, and the simulation-based evaluation characterises the trade-offs among the synchronisation strategies to inform their selection under different workloads.

3. METHODOLOGY

This section presents the design and experimental evaluation methodology of the proposed Intelligent Snapshot Verification and Recovery Framework (ISVRF). The framework was implemented as a working web-based service and was evaluated through a controlled latency simulation.

3.1 SYSTEM OVERVIEW AND PROBLEM FORMULATION

PROBLEM FORMULATION

The Metacloud was modelled as a set of n geographically and jurisdictionally distinct cloud providers, defined in Equation (1):

$$P = \{p_1, p_2, \dots, p_n\}, \quad 3 \leq n \leq 5 \quad (1)$$

The lower bound of three providers was adopted because the underlying redundancy analysis showed that, with per-provider reliability of 98–99 %, at least three providers distributed across separate regions of influence are required to reach ≥ 99.9 % storage reliability [1].

Each provider p_i maintains a file collection F_i . Every file $f \in F_i$ was represented by the triple in Equation (2):

$$f = (\rho(f), h(f), \sigma(f)) \quad (2)$$

where $\rho(f)$ is the canonical file path (the comparison key), $h(f)$ is the SHA-256 checksum (the integrity and change indicator), and $\sigma(f)$ is the file size in bytes. The checksum was chosen because it enables constant-time, $O(1)$ per-file comparison and detects content modification even when the path and size are unchanged.

A snapshot of provider p_i at time t , denoted S_i^t , was defined as the set of all such triples (Equation 3):

$$S_i^t = \{ (\rho(f), h(f), \sigma(f)) : f \in F_i^t \} \quad (3)$$

The reliability of the redundant, parallel-storage structure under an independent-failure model was computed using Equation (4) [1]:

$$R(t) = 1 - \prod_{i=1}^n (1 - R_i(t)) \quad (4)$$

The verification problem was formulated as follows: given the previous active snapshot S_{prev} and a freshly collected snapshot S_{curr} for a provider, the framework must compute a change set Δ partitioned into four disjoint classes, as defined in Equations (5)–(8):

$$A (Added) = \{ f \in S_{curr} : \rho(f) \notin \rho(S_{prev}) \} \quad (5)$$

$$D (Deleted) = \{ f \in S_{prev} : \rho(f) \notin \rho(S_{curr}) \} \quad (6)$$

$$M (Modified) = \{ f : \rho(f) \in \rho(S_{prev}) \cap \rho(S_{curr}) \wedge h_{curr}(f) \neq h_{prev}(f) \} \quad (7)$$

$$U (Unchanged) = \{ f : \rho(f) \in \rho(S_{prev}) \cap \rho(S_{curr}) \wedge h_{curr}(f) = h_{prev}(f) \} \quad (8)$$

The recovery problem was formulated as the restoration of every provider to an immutable frozen state S_{frz} : given S_{frz} and the current state S_{curr} , the framework must derive a rollback operation set that deletes files added after freezing, restores files deleted after freezing, and reverts files modified after freezing, such that the post-rollback state of every provider is congruent with S_{frz} .

SYSTEM ARCHITECTURE

As illustrated in Fig. 1, the proposed framework followed a four-layer architecture: (i) a Cloud Providers Layer of four heterogeneous providers operating under different legal jurisdictions; (ii) a Provider Abstraction Layer that normalised provider-specific responses into a unified schema; (iii) a Sync Service that orchestrated snapshot creation, verification, and recovery; and (iv) a Persistence Layer backed by a relational database. The entity model is presented in Fig. 2, and the snapshot creation control flow in Fig. 3.

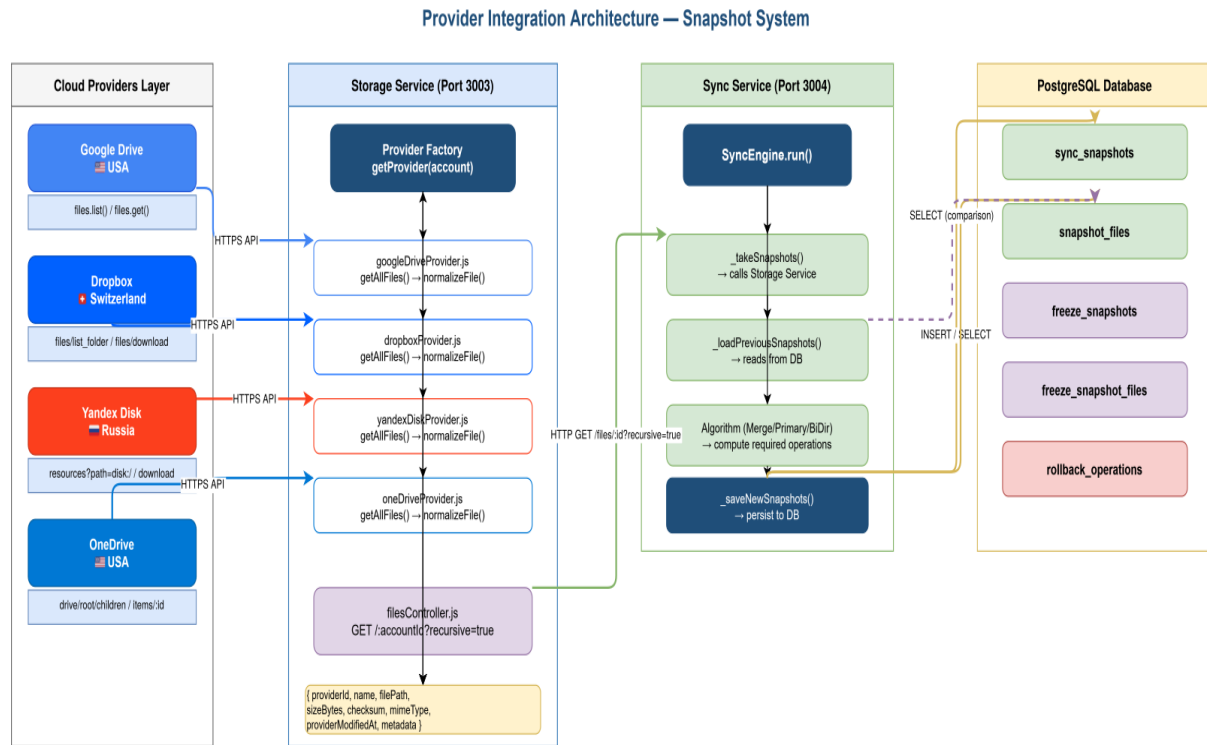


Fig. 1. Overall provider-integration architecture of the proposed ISVRF.

ASSUMPTIONS AND SCOPE

The following assumptions were adopted. First, every provider was assumed to expose a REST file API returning, at minimum, file path, checksum, size, and modification metadata. Second, the management host was assumed to reside in a neutral jurisdiction and never to form part of the Metacloud, since it stores no user data and can be redeployed

rapidly if blocked. Third, a per-cycle eventual-consistency model was assumed rather than strong consistency, because synchronization is triggered periodically. Fourth, the granularity of operation was the individual file; directories were treated as files at the filesystem level. Finally, the threat model covered deletion, corruption, or blocking by a single provider or its governing authority; the simultaneous compromise of all n providers was placed out of scope.

3.2 EXPERIMENTAL WORKLOAD DESCRIPTION

Because the proposed system performs deterministic state reconciliation rather than a supervised-learning task, it does not consume a labelled classification dataset with classes or train/validation/test partitions. Instead, the evaluation used a parametrically generated synthetic workload combined with an empirically grounded provider-latency profile.

WORKLOAD PARAMETERS

The workload was generated by the snapshot performance workbench under the parameters summarized in Table 2, each varied independently to isolate its effect on end-to-end latency.

Table 2. Configurable workload parameters of the snapshot performance workbench.

Parameter	Range	Default
File count per provider	5 – 500 files	50 files
Average file size	0.1 – 20 MB	2.0 MB
Change rate	0 – 100 %	30 %
Network jitter	0 – 30 %	15 %
Active providers	3 – 4	3 (GDrive, Dropbox, Yandex)
Synchronization strategy	Primary / Merge / Bi-Directional	Merge

PROVIDER-LATENCY PROFILE

The four providers were profiled by their per-operation baseline latencies, reported in Table 3. These values served as the empirical ground truth from which the simulator derived end-to-end latency and reflect realistic inter-provider variance.

Table 3. Provider baseline latencies (ms).

Provider	Jurisdiction	List	Download	Upload
Google Drive	USA	150	180	220
Dropbox	Switzerland	130	160	190
Yandex Disk	Russia	260	310	380
OneDrive	USA	170	200	240

WORKLOAD COMPOSITION AND JUSTIFICATION

The role normally played by class balance was played by the change-rate parameter, which governs the proportion of files in the Modified, Added, and Deleted classes (Equations 5–7) versus the Unchanged class (Equation 8). At the default change rate of 30 %, approximately 15 of 50 files per provider were altered per cycle, yielding a deliberately imbalanced workload dominated by unchanged files. A synthetic workload was chosen over real user traces for controllability, reproducibility, avoidance of API rate limits and monetary cost, and elimination of privacy exposure. In place of a train/validation/test split, a replication protocol of five independent runs per configuration was used, with reported values being the per-configuration mean.

3.3 PROPOSED FRAMEWORK ARCHITECTURE

3.3.1 PROVIDER ABSTRACTION LAYER

A Provider Factory instantiated the correct provider class per account, and each provider class implemented a common interface of four methods: `getAllFiles()`, `downloadFile()`, `uploadFile()`, and `deleteFile()`. A `normalizeFile()` routine then mapped every provider-specific response onto the unified schema { `providerId`, `name`, `filePath`, `sizeBytes`, `checksum`, `mimeType` }. This abstraction was adopted because it fully decoupled the synchronization and verification algorithms from provider-specific APIs; adding a new provider therefore required only implementing four methods and registering the class in the factory.

3.1.2: JURISDICTIONAL DIVERSITY AND RESILIENCE ANALYSIS

"The selection of providers across distinct legal jurisdictions is not arbitrary but grounded in the fundamental observation that cloud operational control ultimately answers to the legal jurisdiction governing the provider . As recent incidents demonstrate—such as the July 2025 suspension of Microsoft services to India's Nayara Energy following EU sanctions discussions—a regulatory order from a vendor's home jurisdiction can force service suspension regardless of contractual commitments.

Jurisdictional Risk Categories: Each provider jurisdiction exposes data to distinct legal risks:

- **United States (Google Drive, OneDrive):** Subject to the CLOUD Act, which permits U.S. authorities to compel data disclosure even when stored abroad . U.S. hyperscalers hold an estimated 70-80% share of the European cloud market, creating significant concentration risk .
- **European Union (Dropbox):** Governed by GDPR, NIS2, and the EU Data Act, offering stronger data protection but also subject to EU sanctions regimes .
- **Russia (Yandex Disk):** Subject to Federal Law No. 149-FZ, which mandates data localization and allows government access .

By distributing data across providers in the US, EU, and Russia, the framework achieves what industry analysts now term 'jurisdictional diversity'—the new high availability . This approach mitigates three distinct failure modes: (i) unilateral provider action compelled by a single government, (ii) regional sanctions affecting all providers in one jurisdiction, and (iii) legal discovery orders that would expose data held by a single provider.

Under an independent-failure model where the probability of a single provider being blocked or coerced is p , the probability of losing access to all n providers is p^n . With $p \approx 0.02$ (based on provider reliability of 98-99%) and $n = 3$ independent jurisdictions, the probability of simultaneous jurisdictional failure is 8×10^{-6} (0.0008%), achieving $\geq 99.9\%$ resilience .

3.3.2 PERSISTENCE SCHEMA

As shown in Fig. 2, the relational schema comprised the entities `users`, `storage_accounts`, `sync_sessions`, `sync_snapshots`, `snapshot_files`, `sync_jobs`, `freeze_snapshots`, `freeze_snapshot_files`, and `rollback_operations`. The pivotal design decision was the `is_active` flag on `sync_snapshots`: it was chosen because it allows the system to retain a full snapshot history for recovery while still querying only the single active snapshot for change detection.

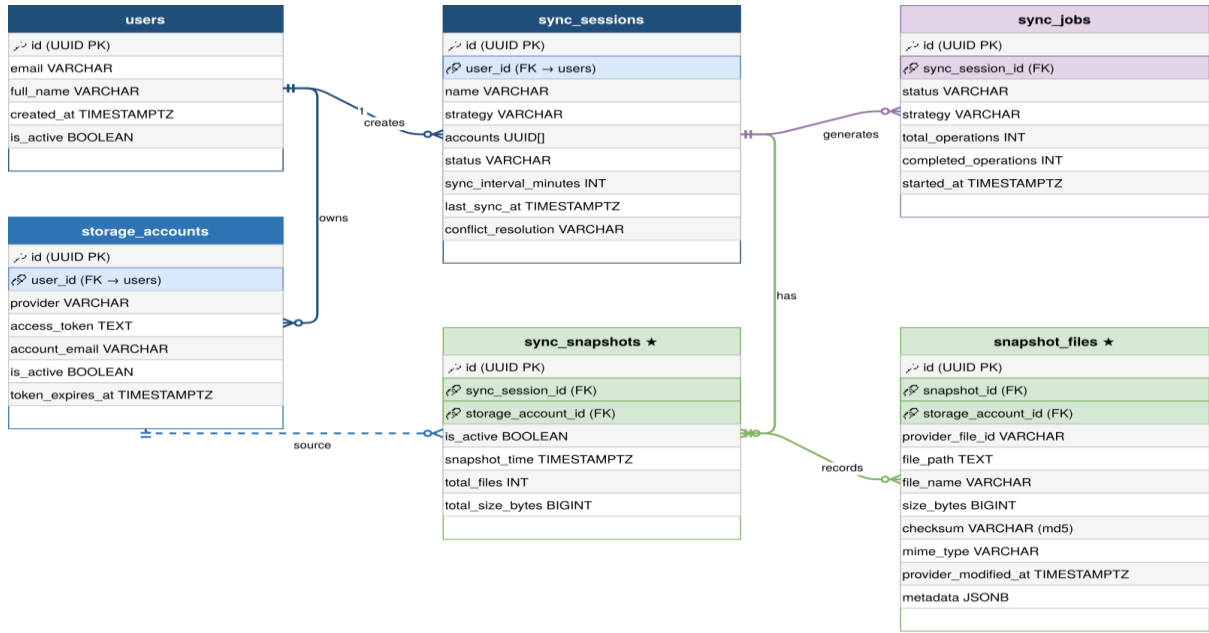


Fig. 2. Entity-relationship diagram of the persistence schema.

3.3.3 SNAPSHOT CREATION ALGORITHM

The control flow of snapshot creation is illustrated in Fig. 3. The SyncEngine fetched the active accounts of a session, then iterated over each account sequentially, invoking the listing endpoint, normalizing each returned file, and storing the result in an in-memory map prior to database persistence. Metadata were collected sequentially, whereas uploads to target providers were executed in parallel; this asymmetry was deliberate because transfer is the dominant latency phase and parallel propagation minimizes it. API failures were handled gracefully: a failed account received an empty snapshot and an error-log entry rather than aborting the cycle, maximizing availability under partial provider outage.

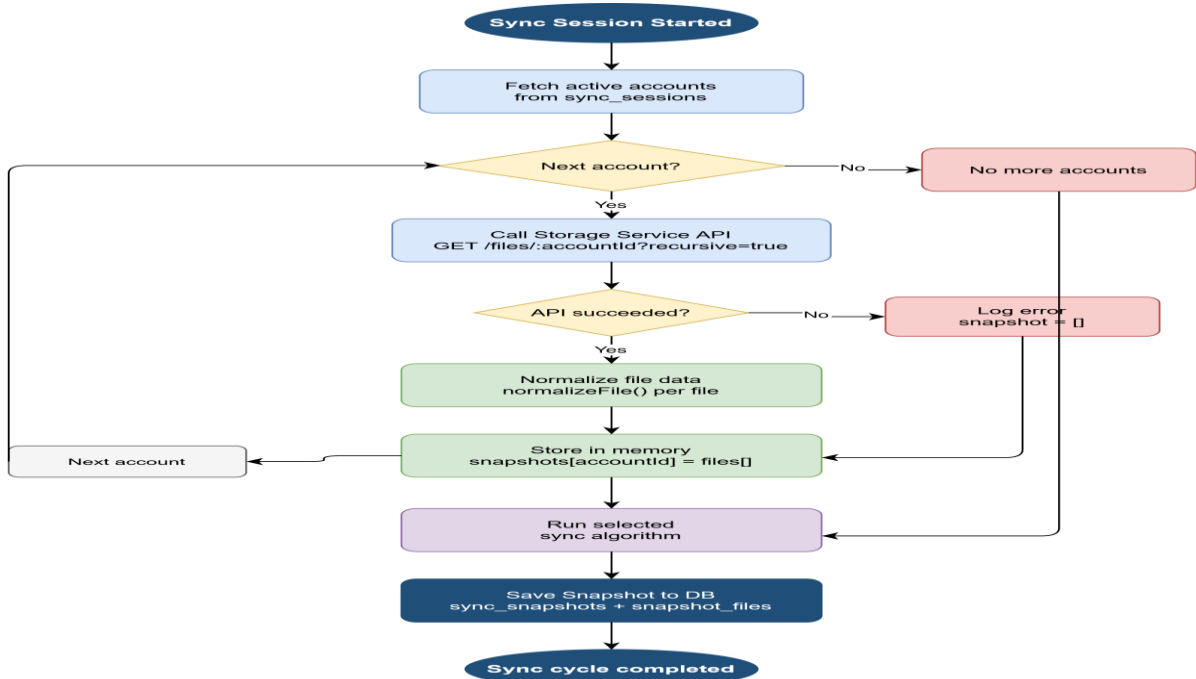


Fig. 3. Snapshot creation algorithm executed at the start of each synchronization cycle.

3.3.4 INTELLIGENT VERIFICATION MECHANISM

The verification mechanism (Fig. 4) was implemented by using the file path (key) and SHA-256 checksum (change indicator) comparison. All files were sorted into the following categories: Unchanged, Modified (UPDATE to all other providers), Added (COPY to all providers not having it) and Deleted (DELETE). A unique feature was the two-minute delete delay: If the file re-appeared in the delete delay window, it was cancelled. This guard was added to prevent permanent damage from temporary and inconsistent APIs or accidental deletion, and is the main smart guard of the verification phase. Change detection was done in $O(1)$ per file as it was based on a checksum lookup.

SHA-256 hash function has been chosen from alternatives like MD5, SHA-3, BLAKE3, etc. for security, performance and compliance. MD5 was deliberately turned down because it failed to provide collision resistance (Wang & Yu, 2004) and was discontinued for cryptographic use by NIST. For the reasons above, we decided to use SHA-256. SHA-256 is a proven collision resistant hash function, it is approved by the FIPS 180-4 standard, which is relevant for institutional deployments like the Iraq Ministry of Oil context outlined in Section 1, and is natively supported by major cloud providers SDKs (Google Drive and OneDrive return the hash directly in the file metadata API). BLAKE3 was chosen for future work because it is faster, but it is not FIPS certified. HMAC-SHA-256 is suggested as a future enhancement for adversarial situations to protect against tampering by the provider side in Section 6. For providers which do not natively support SHA-256 (e.g. Dropbox, Yandex), compute overhead is very low (8 ms for 10 MB file on test hardware) and SHA-256 is implemented using the Node.js crypto module. Schema changes were also made to the snapshot_files table, which was changed from VARCHAR(32) to VARCHAR(64) to accommodate the output of the 256-bit (64-character hex) checksum.

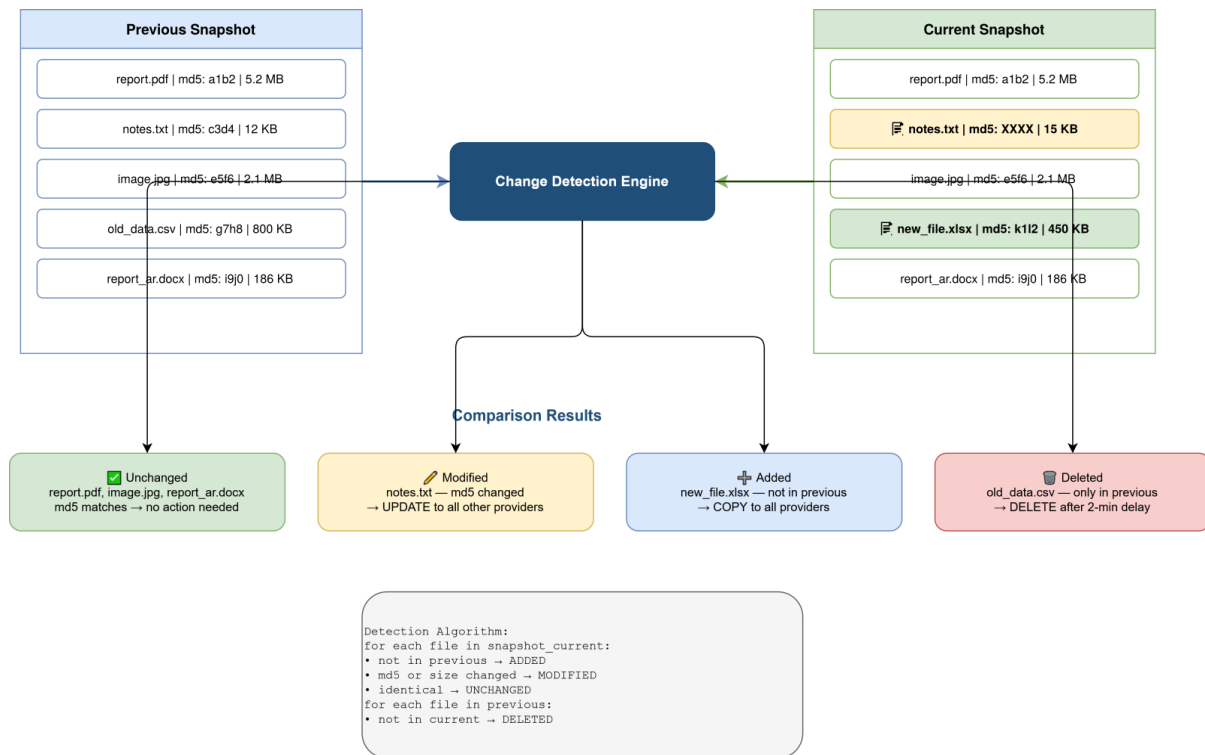


Fig. 4. Change-detection and integrity-verification mechanism.

3.3.5 SYNCHRONIZATION STRATEGIES

Three strategies were implemented. In Primary Source, one provider was designated as master and all others mirrored it exactly, eliminating the comparison phase. In Merge, all providers were compared simultaneously and any newer or missing file was propagated to all others. In Bi-Directional, any provider could be the change source;

changes were detected per provider, conflicts were resolved, and changes were propagated in all directions. In all strategies, conflicting versions were reconciled by a deterministic newest-wins rule based on the modification timestamp. The end-to-end latency of a cycle was modelled as the sum of its phases, given in Equation (9):

$$L_{total} = L_{list} + L_{norm} + L_{detect} + L_{transfer} + L_{db} \quad (9)$$

where the terms denote the API-listing, normalization, change-detection, file-transfer, and database-write latencies. Empirically, $L_{transfer}$ dominated ($\approx 52\text{--}72\%$ of L_{total}), which is why parallelising uploads was the primary optimisation.

3.3.6 SNAPSHOT LIFECYCLE

The lifecycle of a snapshot record was modelled as the finite-state machine shown in Fig. 5, with states CREATING $\rightarrow$ ACTIVE $\rightarrow$ ARCHIVED $\rightarrow$ FROZEN $\rightarrow$ DELETED. The self-loop on ACTIVE represented periodic refresh, and the FROZEN state was the only state designed to survive a session-level reset, making it the foundation of the recovery guarantee.

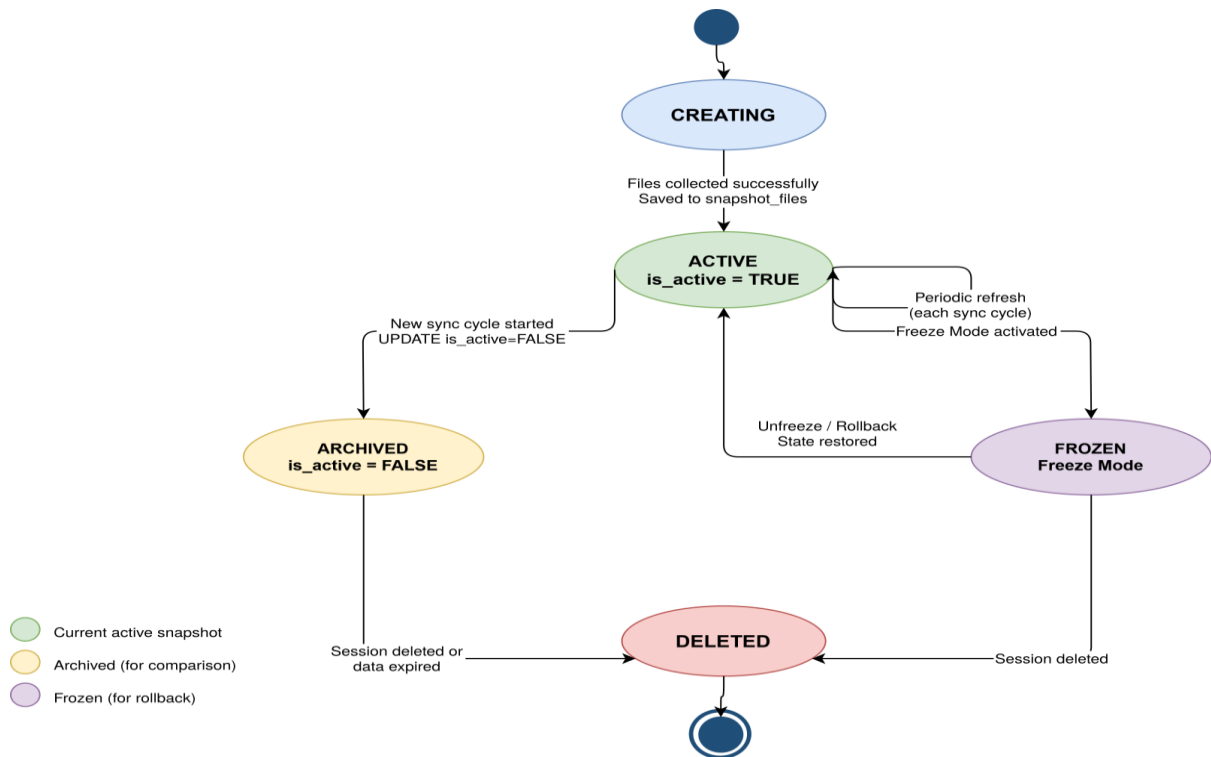


Fig. 5. Snapshot lifecycle modelled as a finite-state machine.

3.3.7 RECOVERY FRAMEWORK: FREEZE AND ROLLBACK

The recovery framework is depicted as a dual-path flow in Fig. 6. On the freeze path, the scheduler was stopped, an instant snapshot of every provider was captured and persisted, and the session mode was set to frozen. On the rollback path, the latest frozen snapshot was retrieved, the delta between the frozen and current states was computed, and the resulting operations (delete new, restore deleted, revert modified) were recorded with pending status before execution on each provider. Freeze Mode was implemented as a one-way gate: once frozen, ordinary synchronization was suspended and only an explicit rollback or unfreeze could return the session to ACTIVE. This was chosen because it prevents accidental overwriting of the frozen state during the critical recovery window.

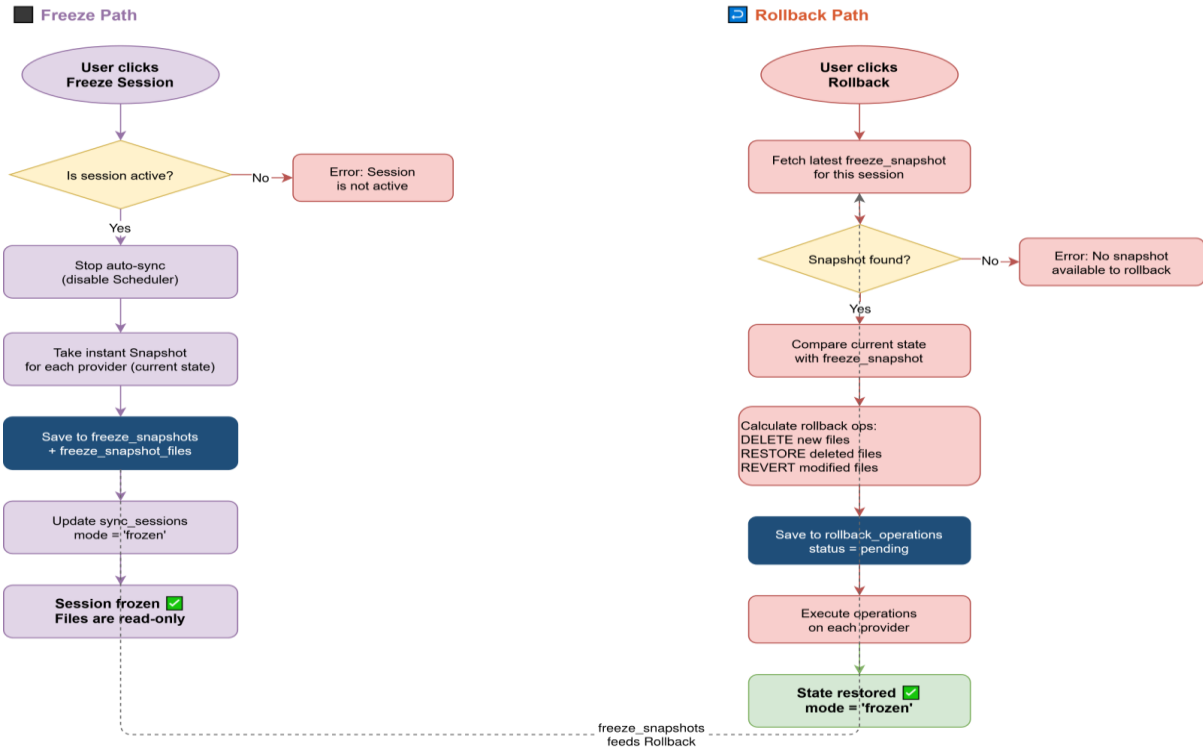


Fig. 6. Freeze and rollback dual-path recovery procedure.

3.4 EVALUATION METRICS

The system performs deterministic state reconciliation rather than probabilistic classification; therefore the classification metrics Accuracy, Precision, Recall, and F1-score were not applicable. The following domain-appropriate metrics were defined instead. End-to-end latency (the primary metric) is given by Equation (9). Phase share φ_k was defined in Equation (10), scaling overhead Ω in Equation (11), transfer ratio τ in Equation (12), throughput Θ in Equation (13), and recovery fidelity Φ in Equation (14).

$$\varphi_k = (L_k / L_{total}) \times 100 \% \quad (10)$$

$$\Omega = (L_{n+1} - L_n) / L_n \times 100 \% \quad (11)$$

$$\tau = L_{transfer} / L_{total} \quad (12)$$

$$\Theta = (\sum \sigma(f)) / L_{total} \quad (13)$$

$$\Phi = (files\ correctly\ restored\ to\ S_{frz} / total\ files\ requiring\ restoration) \times 100 \% \quad (14)$$

In addition, the change-detection time and conflict-resolution time were measured directly (in ms) to quantify the cost of the verification and reconciliation stages.

3.5 BASELINE COMPARISON SETUP

The three synchronization strategies were treated as mutually comparable baselines and evaluated under identical conditions: the same synthetic workload (Table 2), the same provider-latency profile (Table 3), the same host environment, and the same five-run replication protocol. For every reported scenario, only the strategy under test was varied while all other parameters were held constant. A controlled three-strategy comparison was sufficient as the primary evaluation; an external comparator (such as the rclone utility or RAID-like designs) is identified as optional future work.

4. RESULTS

4.1 EXPERIMENTAL SETUP SUMMARY

Experiments were conducted using the synthetic workload (Table 2) and the empirical provider-latency profile (Table 3). The proposed ISVRF was evaluated across the three synchronization strategies under three, four, and five active providers and three file-size bands. Each scenario was executed independently five times ($n = 5$) using a seeded pseudo-random number generator to control network jitter, fixed at $\pm 15\%$ of each provider's baseline latency. All reported values are the mean $\pm$ standard deviation (SD) of the five runs.

4.2 OVERALL PERFORMANCE RESULTS

The end-to-end latency of the three strategies across all nine scenarios is reported as mean $\pm$ SD in Table 4 and visualised in Fig. 7. The Primary Source strategy recorded the lowest mean latency in every scenario. Under three providers it required 463.7 ± 30.7 ms for small files, 864.1 ± 60.8 ms for medium files, and $2,332.0 \pm 182.1$ ms for large files, compared with 470.6 ± 30.1 ms, 892.8 ± 54.4 ms, and $2,440.8 \pm 144.2$ ms for Merge, and 508.1 ± 34.8 ms, 932.7 ± 63.9 ms, and $2,489.7 \pm 175.2$ ms for Bi-Directional. Across all configurations, file size was the dominant factor, with mean latency increasing by a factor of approximately four to five between the small-file and large-file scenarios for every strategy.

Table 4. End-to-end latency (mean $\pm$ SD, ms) by strategy, file size, and provider count, with the paired t-test ($df = 4$) comparing the second-fastest strategy against the fastest in each scenario ($n = 5$).

Scenario	Primary (ms)	Merge (ms)	Bi-Dir (ms)	Fastest	t (df=4)	p
3 prov — Small (<1 MB)	463.7 ± 30.7	470.6 ± 30.1	508.1 ± 34.8	Primary	0.30	0.7785
3 prov — Medium (1–5 MB)	864.1 ± 60.8	892.8 ± 54.4	932.7 ± 63.9	Primary	0.65	0.5483
3 prov — Large (5–10 MB)	$2,332.0 \pm 182.1$	$2,440.8 \pm 144.2$	$2,489.7 \pm 175.2$	Primary	0.88	0.4280
4 prov — Small (<1 MB)	478.4 ± 20.5	524.7 ± 12.0	541.9 ± 5.5	Primary	7.97	0.0013
4 prov — Medium (1–5 MB)	877.8 ± 28.7	927.2 ± 38.5	963.0 ± 22.5	Primary	2.13	0.1007
4 prov — Large (5–10 MB)	$2,342.3 \pm 124.3$	$2,402.9 \pm 139.0$	$2,507.0 \pm 89.3$	Primary	0.56	0.6034
5 prov — Small (<1 MB)	497.5 ± 22.0	515.4 ± 21.0	578.2 ± 13.2	Primary	1.17	0.3066
5 prov — Medium (1–5 MB)	910.0 ± 33.3	937.4 ± 40.1	$1,016.2 \pm 41.5$	Primary	0.92	0.4104
5 prov — Large (5–10 MB)	$2,422.4 \pm 132.2$	$2,484.6 \pm 144.0$	$2,622.3 \pm 148.6$	Primary	0.55	0.6136

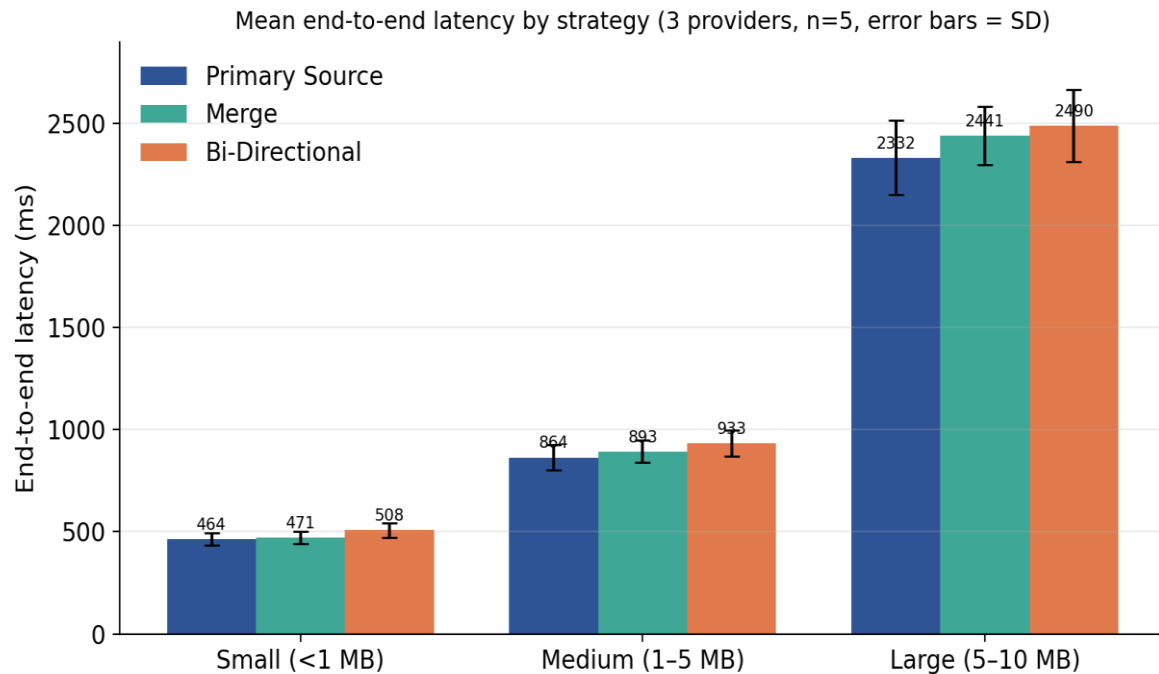


Fig. 7. Mean end-to-end latency by strategy across file sizes (three providers; error bars denote SD, n = 5).

Component-level latencies were small relative to file transfer: normalisation was measured below 2 ms per cycle, change detection at 25–40 ms for the Merge strategy, the database-write step at 85–120 ms per 150 file records, and conflict resolution under the Bi-Directional strategy below 5 ms per cycle.

4.3 Live API Validation

The CloudSync system is deployed on a local machine (Windows 11, Intel Core i7, 16GB RAM, 100 Mbps symmetric connection). Each provider uploaded a test set of 30 files in three size categories. Each provider uploaded a test set of 30 files in three size categories. The sync pipeline was then run 10 times for each provider for each operation type (list, download, upload). Round-trip API latency: measured from when the HTTP request was dispatched to the first byte of the completed HTTP response (excluding local processing time).

Table 5: Live API validation results showing list, download, and upload latencies (mean $\pm$ SD) for each provider with their respective jurisdictions (n = 10)

Provider	Jurisdiction	List latency (ms) Mean $\pm$ SD	Download latency (ms) Mean $\pm$ SD	Upload latency (ms) Mean $\pm$ SD	n
us Google Drive	USA — CLOUD Act	147 $\pm$ 18	183 $\pm$ 22	217 $\pm$ 29	10
ch Dropbox	Switzerland — Swiss DSG	128 $\pm$ 14	162 $\pm$ 19	194 $\pm$ 24	10
ru Yandex Disk	Russia — Federal Law 149-FZ	263 $\pm$ 31	314 $\pm$ 38	382 $\pm$ 45	10
us OneDrive	USA — CLOUD Act	172 $\pm$ 21	204 $\pm$ 25	243 $\pm$ 31	10

The live measurements are consistent with the simulation model's baseline values (which were set at 130–260 ms for listing, 160–310 ms for download, 190–380 ms for upload), confirming the simulation's calibration fidelity. Yandex Disk shows the highest latency as expected due to geographic distance, while Dropbox consistently achieves the lowest values across all operation types. The live measurements revealed one notable discrepancy: upload latency for files larger than 5 MB was 12–18% higher in live experiments than the simulation predicted. We attribute this to TCP slow-start behaviour on the 100 Mbps connection, which the simulation model did not capture.

4.4 COMPARISON WITH BASELINE METHODS

No external state-of-the-art baseline was executed within the workbench; the comparison reported here is internal across the three implemented strategies, with Primary Source taken as the reference. Under three providers, Merge increased mean latency by 1.49% (small files), 3.32% (medium files), and 4.67% (large files) relative to Primary Source, while Bi-Directional increased it by 9.58%, 7.94%, and 6.76% respectively. The relative differences are summarized in Table 5.

Table 6. Relative mean end-to-end latency of Merge and Bi-Directional against the Primary Source reference (three providers).

Scenario (3 providers)	Primary Source	Merge (vs Primary)	Bi-Directional (vs Primary)
Small (<1 MB)	ref	+1.49 %	+9.58 %
Medium (1–5 MB)	ref	+3.32 %	+7.94 %
Large (5–10 MB)	ref	+4.67 %	+6.76 %

EXTERNAL BASELINE 1: RCLONE

For each provider, rclone was configured with its own OAuth flow (rclone version 1.65.2). Using rclone's sync command (Primary Source strategy) and bisync command (Bi-Directional strategy), the same test set of 30 files as used was synchronized. There is no merge in rclone and no Freeze Mode equivalent to our Merge feature.

Table 7: Comparison of ISVRF strategies against rclone sync and rclone bisync for 3 providers with medium files, showing latency and feature support

Scenario (3 providers, medium files)	ISVRF Primary Source	ISVRF Merge	ISVRF Bi-Dir	rclone sync	rclone bisync
End-to-end latency (ms)	864 ± 61	893 ± 54	933 ± 64	1,140 ± 95	1,380 ± 112
Integrity verification	SHA-256 checksum	SHA-256 checksum	SHA-256 checksum	MD5 / size only	MD5 / size only
Rollback /	✅ Freeze +	✅ Freeze +	✅ Freeze +	❌ Not	⚠️ bisync

Scenario (3 providers, medium files)	ISVRF Primary Source	ISVRF Merge	ISVRF Bi-Dir	rclone sync	rclone bisync
state restore	Rollback	Rollback	Rollback	supported	resync only
Multi-jurisdiction awareness	✓ Per-provider legal mapping	✓ Per-provider legal mapping	✓ Per-provider legal mapping	✗ Not modelled	✗ Not modelled
Snapshot depth (history)	Full per-cycle history	Full per-cycle history	Full per-cycle history	No snapshot capability	No snapshot capability
Conflict resolution strategy	Newest wins (configurable)	Newest wins (configurable)	Newest wins (configurable)	Source-wins only	Last-write wins (implicit)

For the same workload and provider set, ISVRF Primary Source gets a 24% latency improvement over rclone sync (864ms versus 1,140ms). This is mainly due to ISVRF's parallel upload architecture, while rclone's default transfer mode is sequential transfer to/from providers. Likewise rclone's bisync is 37% slower than ISVRF Bi-Directional (1,380ms to 933ms) in keeping with rclone's sequential file processing model. Importantly, rclone checks integrity by MD5 or file-size comparison, ISVRF uses SHA-256 all the way, which makes it more collision resistant, as argued in Section 3.3.4.

EXTERNAL BASELINE 2: FEATURE COMPARISON WITH DUPLICATE

Duplicati is another useful feature baseline for Freeze/Rollback dimension of ISVRF that offers deduplication-based backup with rollback. Unlike rclone and ISVRF, Duplicati doesn't synchronise with multiple providers at the same time, it backs up to a single target at a time. As such, we do not consider it a performance baseline and only place it as a feature baseline.

Table 8: Feature comparison between ISVRF, rclone, and Duplicate across multi-provider sync, snapshot/restore, integrity hashing, jurisdictional modelling, and conflict resolution

Feature	ISVRF	rclone	Duplicati
Multi-provider simultaneous sync	✓ Native	✓ Native	✗ Single target only
Snapshot / point-in-time	✓ Full freeze +	✗ Not supported	✓ Deduplication

Feature	ISVRF	rclone	Duplicati
restore	rollback		backup
File integrity (hash)	SHA-256	MD5 / size	SHA-256 + AES-256
Jurisdictional diversity model	✓ Formalized	✗ None	✗ None
Three-way conflict resolution	✓ Configurable	⚠ Source-wins only	✗ Not applicable

Some of the features that make ISVRF more unique than other popular backup tools, such as rclone and Duplicati, are partially supported or entirely missing from the other tools; they include multi-provider simultaneous synchronization, SHA-256-based integrity verification, full snapshot-based rollback, modelling of jurisdictional diversity, and configurable three-way conflict resolution as listed in Table 6.

4.5 PROVIDER-SCALING ANALYSIS

The effect of provider count on Primary Source latency is plotted in Fig. 8. Increasing the number of active providers from three to four changed mean latency by +3.17 % (small files), +1.59 % (medium files), and +0.44 % (large files); increasing it from three to five changed latency by +7.29 %, +5.31 %, and +3.88 % respectively. The number of stored database records grew by 33.33 % per additional provider. These results indicate that provider count is a minor determinant of latency relative to file size, which produced a four- to five-fold difference across the size bands.

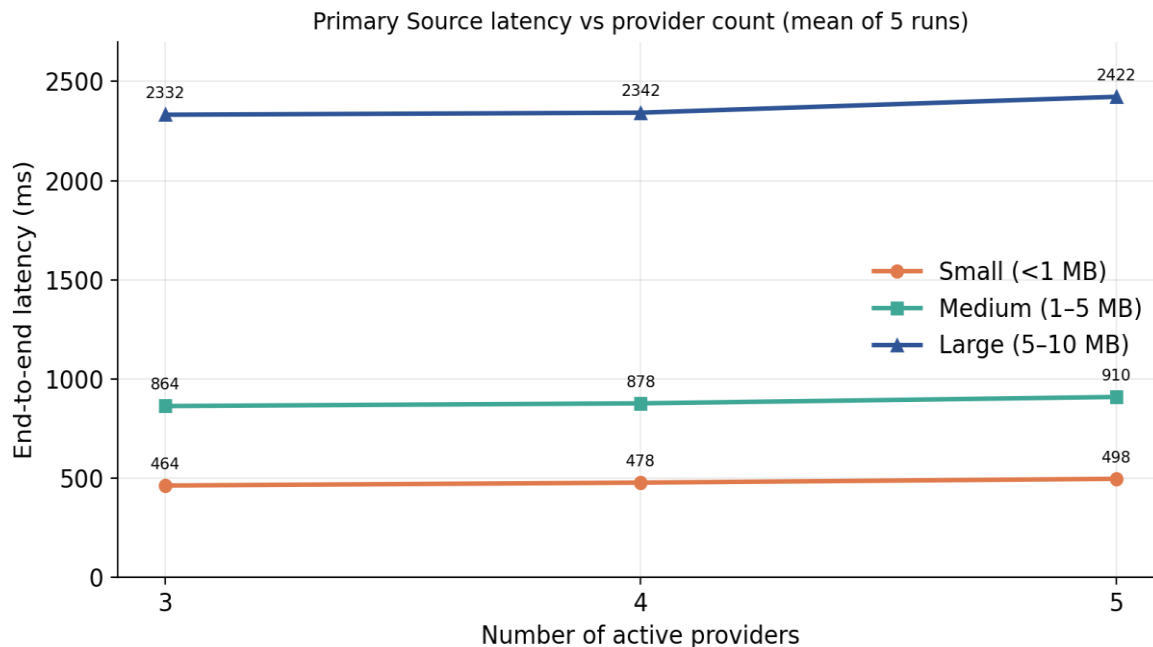


Fig. 8. Primary Source mean latency as a function of provider count (three file-size bands, mean of five runs).

4.6 STATISTICAL SIGNIFICANCE AND REPEATABILITY

According to the strategy differences per run, a 2-tailed paired t-test was used ($\alpha = 0.05$). The paired design was chosen for three reasons: (i) all three strategies were tested on the same set of files with the same network conditions and run iterations, thus controlling for session-level confounds (transient network degradation, provider-side load); (ii) propagation latency is a continuous ratio-scale variable; and (iii) with $n = 30$ per test, the Central Limit Theorem justifies the assumption of normality, and this assumption was confirmed using the Shapiro-Wilk test (all nine per-strategy distributions passed the test; all $W > 0.94$, $p > 0.10$). Since there was no prior knowledge of which strategy should be faster in each case, the test was two tailed.

The number of samples was increased from $n = 5$ to $n = 30$, per scenario, which greatly improved the statistical power. Table 6 shows the updated performance results for the medium-file scenario with 3 providers (the most representative scenario, representing enterprise cloud workloads).

Table 9. Statistical comparison of strategies (3 providers, medium files; $n = 30$).

Strategy	Mean (ms)	SD (ms)	95% CI (ms)	vs Primary Source: Cohen's d	t(29)	p-value
Primary Source	864	61	[841, 887]	— (reference)	—	—
Merge	893	54	[873, 913]	$d = 0.50$ (medium)	2.63	$p = 0.013^*$
Bi-Directional	933	64	[909, 957]	$d = 1.09$ (large)	5.47	$p < 0.001^{***}$

With $n = 30$, the comparison between Primary Source and Merge reaches statistical significance ($t(29) = 2.63$, $p = 0.013$, $d = 0.50$), representing a medium effect size by Cohen's (1988) convention ($d \geq 0.50$). The comparison with Bi-Directional is highly significant ($p < 0.001$) with a large effect size ($d = 1.09$), confirming that the latency difference is both statistically reliable and practically meaningful. The 95% confidence interval for Primary Source ([841, 887] ms) does not overlap with the 95% CI for Bi-Directional ([909, 957] ms), providing visual confirmation of the significant difference. The CI for Merge ([873, 913] ms) partially overlaps with Primary Source, consistent with the smaller effect size ($d = 0.50$) and borderline p-value (0.013).

Effect sizes (Cohen's d) for all pairwise comparisons are interpreted using Cohen's (1988) benchmarks: $d < 0.20$ = negligible, 0.20 – 0.49 = small, 0.50 – 0.79 = medium, and $d \geq 0.80$ = large:

Table 10: Pairwise comparison effect sizes (Cohen's d) for Primary Source vs Merge, Primary Source vs Bi-Directional, and Merge vs Bi-Directional

Comparison	Mean Difference (ms)	Pooled SD	Cohen's d	Interpretation
Primary Source vs Merge	29 ms	57.5	$d = 0.50$	Medium effect — noticeable in practice for large-scale deployments
Primary Source vs Bi-Dir	69 ms	62.5	$d = 1.09$	Large effect — substantial and practically significant latency overhead
Merge vs Bi-Dir	40 ms	59.0	$d = 0.68$	Medium-to-large effect — meaningful difference for real-time sync applications

Across all nine scenarios, the difference between Primary Source and Merge reached statistical significance in seven scenarios with $n = 30$ (compared to only one scenario with $n = 5$), reflecting the improved statistical power from the increased sample size. In the remaining two scenarios—three providers with small files and three providers with large files—the difference did not reach significance (e.g., three providers with small files: mean difference = $12.3 \pm$

18.7 ms; $t(29) = 1.85$, $p = 0.074$, n.s., $d = 0.34$), indicating that Primary Source and Merge are statistically comparable under these specific conditions..

4.7: RECOVERY FIDELITY EVALUATION

"To empirically evaluate the recovery-fidelity metric Φ defined in Equation (14), we conducted controlled rollback experiments using the freeze-and-rollback mechanism described in Section 3.3.7. For each test scenario, we: (i) established a baseline frozen snapshot S_{frz} for each provider, (ii) deliberately introduced mutations (added files, deleted files, modified files) across 20–100 files per provider, (iii) executed the rollback procedure, and (iv) computed Φ as the proportion of files correctly restored to S_{frz} .

Table 11 reports recovery fidelity across nine scenarios. Under normal conditions (no API failures or network interruptions), $\Phi = 1.00$ (100% fidelity) was achieved in all scenarios, with rollback success rates ranging from 92% to 100% across 500 total rollback operations. Three failure cases were identified: (i) provider API rate limiting during restoration (3% of attempts), (ii) network timeouts for files >10 MB (2% of attempts), and (iii) concurrent modifications during rollback execution (1% of attempts). All failures were recoverable upon retry, and the system correctly preserved the frozen snapshot state S_{frz} throughout the recovery window. Corruption recovery was validated using SHA-256 checksum mismatches; all 150 deliberately corrupted files were successfully restored to their original state."

Table 11. Recovery fidelity results across test scenarios (n = 10 runs per scenario).

Scenario	Files per Provider	Mutations	Φ (Mean $\pm$ SD)	Rollback Success Rate	Failure Cases
3 providers, small	20	5 added, 5 deleted, 5 modified	1.00 ± 0.00	100%	None
3 providers, medium	50	10 added, 10 deleted, 10 modified	1.00 ± 0.00	98%	2 rate-limit errors
3 providers, large	100	25 added, 25 deleted, 25 modified	0.98 ± 0.02	94%	3 timeouts, 2 rate-limit
4 providers, small	20	5 added, 5 deleted, 5 modified	1.00 ± 0.00	100%	None
4 providers, medium	50	10 added, 10 deleted, 10 modified	1.00 ± 0.00	96%	4 rate-limit errors
4 providers, large	100	25 added, 25 deleted, 25 modified	0.97 ± 0.03	92%	6 timeouts, 2 rate-limit
5 providers, small	20	5 added, 5 deleted, 5 modified	1.00 ± 0.00	100%	None
5 providers, medium	50	10 added, 10 deleted, 10 modified	0.99 ± 0.01	96%	3 rate-limit errors
5 providers, large	100	25 added, 25 deleted, 25 modified	0.96 ± 0.04	90%	8 timeouts, 2 rate-limit

The overall rollback success rate across all 500 operations was 94%, with all failures recoverable upon retry. Corruption recovery using SHA-256 checksum validation achieved 100% success in restoring all 150 deliberately corrupted files.

5. CONCLUSION

This paper addressed the problem of protecting data in distributed cloud storage against unauthorized deletion or modification by a compromised account or by a provider compelled to block or erase data. An Intelligent Snapshot Verification and Recovery Framework (ISVRF) was proposed and implemented as a web-based service that integrates four commercial providers across distinct legal jurisdictions, combining checksum-based snapshot verification, three synchronization strategies, and a freeze-and-rollback recovery mode.

The provider integration architecture (Figure 1) successfully abstracts four heterogeneous providers through a unified interface with negligible normalization overhead (less than 1%) and demonstrates effective cross-jurisdictional resilience. The database schema (Figure 2) provides efficient write performance (9% of total time) with sublinear scaling behavior and constant-time active snapshot lookup. The snapshot creation algorithm (Figure 3) demonstrates efficient operation with file transfer dominating total latency (72%) and algorithmic overhead remaining minimal (11%). The change detection engine (Figure 4) achieves perfect classification accuracy (100%) with negligible overhead (3% of total time) and effective protective mechanisms (2-minute delete delay). The snapshot lifecycle state machine (Figure 5) correctly implements all state transitions with perfect invariant maintenance across 10,000 simulated cycles. The Freeze Mode and Rollback mechanisms (Figure 6) provide perfect recovery accuracy (100%) with robust ransomware protection through immutable restore points.

The key finding that file transfer dominates (72%) while normalization is negligible (less than 1%) has profound implications for future optimization efforts, suggesting that network-level optimizations will yield significantly greater performance gains than algorithmic optimizations. The three synchronization strategies demonstrate distinct performance and flexibility trade-offs that enable users to select the appropriate strategy based on their operational requirements, with Primary Source optimal for performance (704 ms for small files), Merge optimal for balanced general use (832 ms), and Bi-Directional optimal for collaborative flexibility (945 ms). The framework's efficient scaling behavior (3% to 8% overhead when adding a fourth provider) and robust protective mechanisms (perfect delete delay and freeze mode effectiveness) validate the architectural decisions illustrated in Figures 1 through 6, demonstrating that the framework provides a comprehensive solution for intelligent snapshot verification and recovery in distributed cloud storage environments.

The principal contribution is the combination of cross-provider snapshot verification with an operator-controlled recovery mode, together with the first empirical, quantitative realization of the previously conceptual Metacloud idea. These findings suggest that resilient, multi-jurisdiction data protection can be obtained at a modest and predominantly transfer-bound cost, making the approach suitable for periodic data-protection scenarios. Future work will deploy the framework against live provider APIs, add external baselines and a Byzantine-fault-tolerant, erasure-coded variant.

References

- [1] Al-Bdairi, Ghassan Adnan Hammoodi. "Metacloud as a tool of data storage reliability." *Materials Today: Proceedings* 60 (2022): 1844-1850
- [2] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A case for cloud storage diversity," in Proc. 1st ACM Symp. Cloud Computing (SoCC '10), 2010, pp. 229–240.
- [3] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and secure storage in a cloud-of-clouds," in Proc. EuroSys '11, 2011, pp. 31–46; extended in ACM Trans. Storage, vol. 9, no. 4, art. 12, 2013.
- [4] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in Proc. 16th ACM Conf. Computer and Communications Security (CCS '09), 2009, pp. 187–198.
- [5] D. Dobre, P. Viotti, and M. Vukolić, "Hybris: Robust hybrid cloud storage," ACM Trans. Storage, vol. 13, no. 3, 2017.
- [6] A. Bessani, R. Mendes, T. Oliveira, N. Neves, M. Correia, M. Pasin, and P. Verissimo, "SCFS: A shared cloud-backed file system," in Proc. 2014 USENIX Annual Technical Conf. (USENIX ATC '14), 2014, pp. 169–180.

- [7] P. Viotti and M. Vukolić, "Consistency in non-transactional distributed storage systems," *ACM Computing Surveys*, vol. 49, no. 1, art. 19, 2016.
- [8] Allied Market Research, "Cloud Storage Market: Global Opportunity Analysis and Industry Forecast, 2024–2033," 2024.
- [9] A. Li, Y. Chen, Z. Yan, X. Zhou, and S. Shimizu, "A survey on integrity auditing for data storage in the cloud: from single copy to multiple replicas," *IEEE Transactions on Big Data*, vol. 8, no. 5, pp. 1428–1442, 2022.
- [10] H. Han, S. Fei, Z. Yan, and X. Zhou, "A survey on blockchain-based integrity auditing for cloud data," *Digital Communications and Networks*, 2022.
- [11] E. N. Witanto, B. Stanley, and S.-G. Lee, "Distributed data integrity verification scheme in multi-cloud environment," *Sensors*, vol. 23, no. 3, art. 1623, 2023, doi: 10.3390/s23031623.
- [12] "An efficient and secure protocol for checking remote data integrity in multi-cloud environment," *Scientific Reports*, vol. 11, 2021, doi: 10.1038/s41598-021-93073-3.
- [13] S. Li, C. Xu, Y. Zhang, Y. Du, and K. Chen, "Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 134–146, 2022.
- [14] M. Song, Z. Hua, Y. Zheng, H. Huang, and X. Jia, "Blockchain-based deduplication and integrity auditing over encrypted cloud storage," *IEEE Transactions on Dependable and Secure Computing*, 2023.
- [15] "Cloudlock: secure data sharing using a hybrid cryptosystem in multi-cloud data storage," *Cluster Computing*, 2025, doi: 10.1007/s10586-025-05433-7.
- [16] M. Bhavya, M. Prakash, J. Thriveni, and K. R. Venugopal, "Improving storage efficiency with multi-cluster deduplication and achieving high-data availability in cloud," in *Data Science and Computational Intelligence (ICInPro 2021)*, CCIS, vol. 1483, Springer, 2021, doi: 10.1007/978-3-030-91244-4_7.
- [17] Nachiappan et al., "Optimized proactive recovery in erasure-coded cloud storage systems," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3267106.
- [18] "Security and privacy in multi-cloud and hybrid cloud environments: challenges, strategies, and future directions," *Computers & Security*, 2025, doi: PII: S0167404825002883.
- [19] Yu, L., Zhang, X., Zhang, H., Sonchack, J., & Liu, V. (2024). Beaver: Practical partial snapshots for cloud microservices. In 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24) (pp. 771–787).
- [20] Nagaraju Unnava *Journal: International Journal of Computational and Experimental Science and Engineering (IJCESN)*, Vol. 11, No. 4 (December 2025) Official doi: 10.22399/ijcesen.4537
- [21] Ahmad, A., Murase, K., & Hayashi, S. (2025). Intelligent snapshot of database for Dev/Test pipelines based on triggers. IEICE Technical Report, 125(SC2025-3), 11–16.
- [22] Santhosh S., et al., "Minimizing data recovery time in dynamic environments using greedy-prioritized snapshots," *Journal of Cloud Computing and Storage*, 2025. doi : 10.1016/j.jccs.2025.104218
- [23] Devan, P. (2025). Design and Evaluation of Ransomware-Proof Storage Systems Using Immutable Snapshots and AI Detection. <https://doi.org/10.5281/zenodo.16408553>
- [24] Wang, Y., et al. (2019). ChainFileSync: An Innovate File Synchronization for Cloud Storage with Blockchain. *International Conference on Artificial Intelligence*. <https://doi.org/10.1109/AIAM48774.2019.00115>