International Conference on Science, Engineering & Management Trends

International Journal of Science, Engineering and Technology
ISSN: 2348-4098, P-ISSN: 2395-4752

Verified Recovery and Safe Degraded Operation in Distributed Cloud Services: An Independent Recovery-Plane Approach

[1]*Raghvendra Rao, [2]* Dr. Yashpal Joshi

[1]*Phd. Research Scholar, Department of Computer Science and Engineering, Saveetha Amarawati University, Vijayawada, Andhra Pradesh, India

[2]*PhD Guide, Department of Computer Science and Engineering, Saveetha Amarawati University, Vijayawada, Andhra Pradesh, India

Abstract- Cloud recovery plans often treat a successful backup job or a restarted service as proof of recovery, although neither shows that a trustworthy service has been restored. This paper presents a recovery approach based on an independent recovery plane—separate identities, trusted artefacts, protected backups and keys—together with a six-step acceptance procedure and explicit degraded-operation states. A simple model shows why restore testing must be regular: if backups fail silently on average once every 180 days, the probability that the latest backup is unusable at a random incident is 1.9% with weekly restore tests, 7.7% with monthly tests and 20% with quarterly tests. Degraded modes are defined so that a service can remain partly useful while dependencies are lost, without hiding changes in meaning from users. The approach turns recovery from an assumed capability into one that is tested and accepted against stated criteria.

Keywords- Distributed cloud, disaster recovery, backup verification, recovery plane, degraded mode, failover, fencing, cyber resilience

I. INTRODUCTION

Recovery is the last line of defence when prevention and fault tolerance fail. Cyber-resilience guidance asks systems to anticipate, withstand, recover from and adapt to adverse conditions [1], and analyses of public cloud outages show that failed or slow recovery often turns a short fault into a long incident [2]. Two weaknesses are common. First, recovery often depends on the same identities, automation and keys as the production system, so an incident that compromises those also blocks recovery. Second, success is judged by technical signals—a completed backup job or a healthy process—rather than by whether correct service has returned [3]. This paper contributes: (1) a minimal independent recovery plane; (2) a six-step recovery acceptance procedure; (3) a model linking restore-test frequency to the chance of an unusable backup; and (4) a set of safe degraded-operation states.

Background

Recovery is usually specified by two objectives: the recovery time objective (RTO), the longest acceptable interruption, and the recovery point objective (RPO), the largest acceptable loss of recent data. Meeting them requires more than fast restoration. Detection, diagnosis, decision and validation also take time, and the selected copy must be both recent and trustworthy. Recovery also crosses the boundary between security and reliability: after ransomware or a compromised pipeline, the question is not only whether data can be restored but whether the restored system can be trusted [1], [4]. Outage studies

International Journal of Science, Engineering and Technology
ISSN: 2348-4098, P-ISSN: 2395-4752

show that recovery mechanisms themselves fail, for example when failover depends on the component that has failed [2].

II. INDEPENDENT RECOVERY PLANE

The recovery plane comprises the identities, artefacts, records and communication paths needed to rebuild useful service. Independence is selective: each element is justified against a specific scenario. A backup account may resist application compromise yet still share administrative credentials, so independence must match the threat and failure model. Table I lists the elements. Recovery identities should be separate from production administration, consistent with zero-trust principles [5], and released artefacts must be traceable to a trusted build so that a compromised pipeline is not reused during recovery [4].

Table 1: Elements of a Minimal Independent Recovery Plane

Element	Purpose	Independence required from
Recovery identities	Authority to restore when normal administration is untrusted	Production administrative accounts
Trusted artefact set	Known-good software to redeploy	Build pipeline under investigation
Protected backups	Data copies unaffected by deletion or corruption	Live replication and its credentials
Key access path	Decrypt restored data	Keys held only in the affected region
Offline runbooks and contacts	Coordinate response without the affected systems	Primary communication tools

III. RECOVERY ACCEPTANCE

Recovery is accepted against outcomes rather than the success of individual scripts (Fig. 1). The recovery identity must be valid and independent; deployed software must match a trusted build; the selected data copy must have a documented lineage and acceptable age, with keys available through an approved path. Before promotion, the former primary must be fenced so that two writers cannot act at once—during a network partition each side may believe the other has failed [6]. Representative records and operations are then reconciled, and the decision and any remaining gaps are recorded. An unresolved critical item prevents an unconditional recovery claim.



Fig. 1. Six-step recovery acceptance procedure. Each step must be satisfied, or explicitly accepted as a residual gap, before normal service is declared.

International Journal of Science, Engineering and Technology
ISSN: 2348-4098, P-ISSN: 2395-4752

III. RESTORE-TEST FREQUENCY

A backup job that reports success shows only that a process ran. Backups can fail silently—through missing data, unusable formats or inaccessible keys—and the failure is found only when a restore is attempted. Suppose silent failures begin at random with rate λ and are detected only by restore tests every T days. The expected undetected period is $T/2$, and the long-run share of time during which the latest backup is unusable is approximately

$$U = (\lambda T/2) / (1 + \lambda T/2) \quad (1)$$

Fig. 2 evaluates Eq. (1). With a silent failure on average every 180 days, the probability that the latest backup is unusable at a random incident is 1.9% with weekly tests, 7.7% with monthly tests, 20% with quarterly tests and 33% with half-yearly tests. With failures every 90 days, quarterly testing gives 33%. Regular, representative restore drills therefore matter as much as backup frequency. A drill should restore realistic data volumes, validate application behaviour and use the recovery identities rather than ordinary production administration [7].

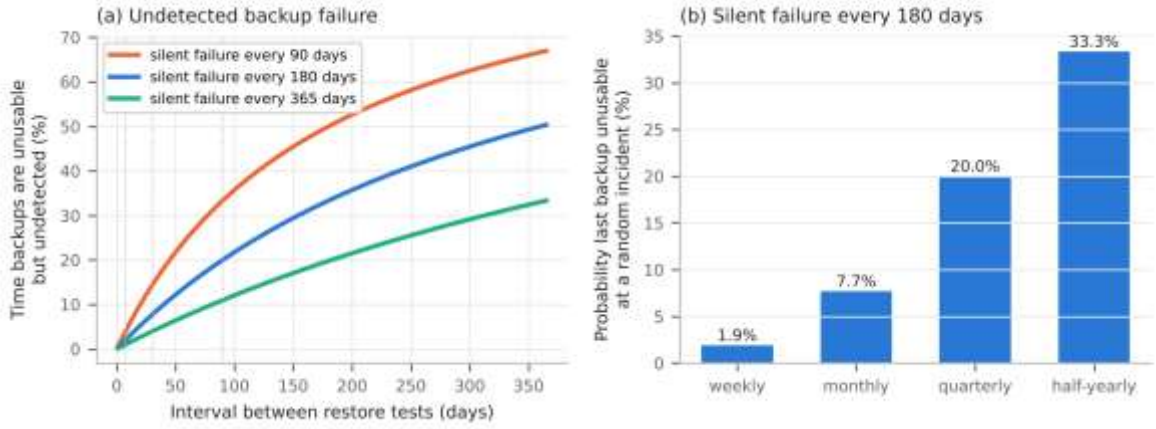


Fig. 2. (a) Share of time during which backups are unusable but undetected, as a function of restore-test interval, for three silent-failure rates (Eq. 1). (b) Probability that the latest backup is unusable at a random incident for four test schedules. Illustrative assumptions.

IV. FAILOVER SAFETY

Failover changes the serving location or component, and it is correct only if the new target has acceptable data, enough capacity and valid authority to act. A simple health-based switch can promote a target that is reachable but stale, or leave the former primary able to keep writing. Table II lists the checks that make failover safe. These checks can make failover slower than an automatic switch, and the RTO should reflect the time they take rather than the time of the switch alone.

Table 2: Failover Safety Checks

Check	Question answered	Evidence
Data position	Is the target recent enough for the operation?	Replication position compared with the RPO
Capacity	Can the target carry the redirected load?	Reserved-capacity record or load test
Authority	Is the target permitted to accept writes?	Coordination or lease record

International Journal of Science, Engineering and Technology
ISSN: 2348-4098, P-ISSN: 2395-4752

Check	Question answered	Evidence
Fencing	Can the former primary still write?	Fencing confirmation before promotion
Dependencies	Are identity, keys and configuration reachable from the target?	Dependency test from the recovery location

V. SAFE DEGRADED OPERATION

When a dependency is lost, a service may keep part of its function by changing its promise openly. Fig. 3 shows the states. If a read dependency is lost, the service may serve older data labelled as such; if the write path is lost, it may accept and queue requests, telling users that a request has been received rather than completed. Before returning to normal, queued work is reconciled and the primary dependency is confirmed stable. Table III lists the rules that keep degraded modes safe. Hidden changes in meaning can preserve apparent uptime while creating reconciliation failures, a risk related to the partial failures that monitoring often misses [8].

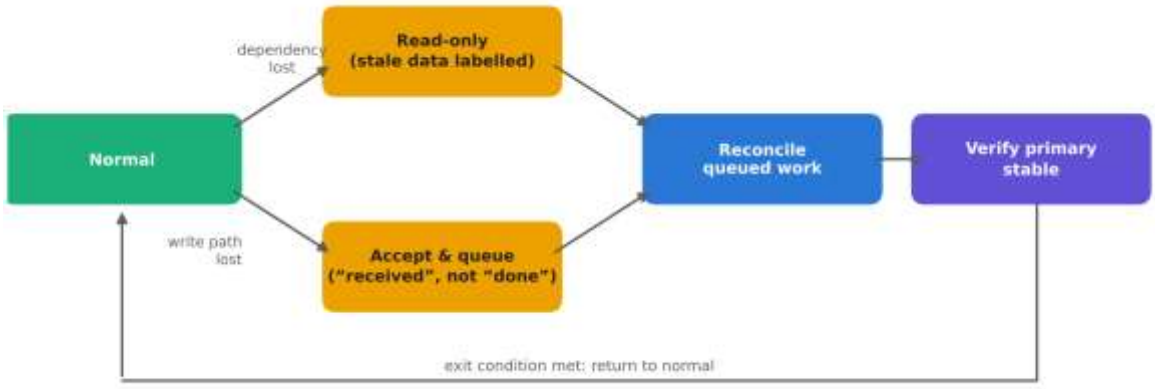


Fig. 3. Safe degraded-operation states. Each degraded state states its changed promise to users and has an explicit exit condition before normal operation resumes.

Table 3: Rules for Degraded Modes

Rule	Reason
State the changed promise to users	Users must distinguish accepted from completed actions
Disable destructive operations	Prevents irreversible actions on uncertain data
Bound queue size and age	Limits backlog and stale decisions
Define an explicit exit condition	Avoids returning too early or staying too long
Reconcile before resuming	Restores a single consistent view of state

VI. CONCLUSION

This paper presented a recovery approach for distributed cloud services that combines an independent recovery plane, outcome-based acceptance, regular restore testing and explicit degraded-operation states. A simple model showed that, with a silent backup failure every 180 days, moving from quarterly to weekly restore tests reduces the chance of an unusable backup at a random incident from 20% to 1.9%. Future work will measure silent-failure rates and recovery times in recovery exercises on deployed services and evaluate user-visible correctness in degraded modes.

REFERENCES

1. R. Ross *et al.*, "Developing cyber-resilient systems: A systems security engineering approach," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-160 Vol. 2 Rev. 1, 2021, doi: 10.6028/NIST.SP.800-160v2r1.
2. H. S. Gunawi *et al.*, "Why does the cloud stop computing? Lessons from hundreds of service outages," in *Proc. 7th ACM Symp. Cloud Comput. (SoCC)*, 2016, pp. 1–16, doi: 10.1145/2987550.2987583.
3. A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, 2004, doi: 10.1109/TDSC.2004.2.
4. R. Chandramouli, F. Kautz, and S. Torres-Arias, "Strategies for the integration of software supply chain security in DevSecOps CI/CD pipelines," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-204D, 2024, doi: 10.6028/NIST.SP.800-204D.
5. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST SP 800-207, 2020, doi: 10.6028/NIST.SP.800-207.
6. S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002, doi: 10.1145/564585.564601.
7. A. Basiri *et al.*, "Chaos engineering," *IEEE Softw.*, vol. 33, no. 3, pp. 35–41, 2016, doi: 10.1109/MS.2016.60.
8. P. Huang *et al.*, "Gray failure: The Achilles' heel of cloud-scale systems," in *Proc. 16th Workshop Hot Topics Oper. Syst. (HotOS)*, 2017, pp. 150–155, doi: 10.1145/3102980.3103005.