

Panacea: From Ambiguous Incidents to Safe Mitigation Policies at Azure Scale

Ze Li

Microsoft Azure
Redmond, USA
zeli@microsoft.com

Chang Lou

University of Virginia
Charlottesville, VA, USA
chlou@virginia.edu

Vivek Ramamurthy

Microsoft Azure
Redmond, USA
viramam@microsoft.com

Vignatha Yenugutala

Microsoft Azure
Redmond, USA
vyenugutala@microsoft.com

Eion Blanchard

Microsoft Azure
Redmond, USA
eblanchard@microsoft.com

Minghua Ma

Microsoft Azure
Redmond, USA
minghuama@microsoft.com

Hemant Malik

Microsoft Azure
Redmond, USA
hemantmalik@microsoft.com

Shweta Patil

Microsoft Azure
Redmond, USA
shpati@microsoft.com

Murali Chintalapati

Microsoft Azure
Redmond, USA
muralic@microsoft.com

Abstract

Mitigating anomalies in production cloud cannot wait. Yet mitigation can often be unsafe when applied at fleet scale: the symptom may be transient or tenant-local, while the action may conflict with workload or capacity constraints. A poorly scoped mitigation is sometimes worse than the anomaly itself, which could disrupt workloads or escalate a local problem into a major outage.

We present PANACEA, a production framework in Microsoft Azure for turning ad hoc mitigation actions into safe fleet-wide policies. PANACEA organizes each policy around an activation boundary that makes conditions explicit. PANACEA then applies the action within the boundary, and brings production evidence back to operators so that the boundary can be evolved as production evidence accumulates. Deployed in Microsoft Azure for more than two years, PANACEA currently supports 16 anomaly scenarios across CPU, memory, storage, and networking. PANACEA's preventive actions correspond to an estimated 12% reduction in fleet-wide interruption rates over six months.

CCS Concepts

• Computer systems organization → Cloud computing.

Keywords

cloud reliability, gray failures, automated mitigation, production cloud systems

ACM Reference Format:

Ze Li, Chang Lou, Vivek Ramamurthy, Vignatha Yenugutala, Eion Blanchard, Minghua Ma, Hemant Malik, Shweta Patil, and Murali Chintalapati. 2026. Panacea: From Ambiguous Incidents to Safe Mitigation Policies at Azure Scale. In *ACM Symposium on Cloud Computing Volume 2 (SoCC '26 Vol 2)*.



This work is licensed under a Creative Commons Attribution 4.0 International License. SoCC '26 Vol 2, Singapore, Singapore
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2816-7/2026/11
<https://doi.org/10.1145/3828161.3857870>

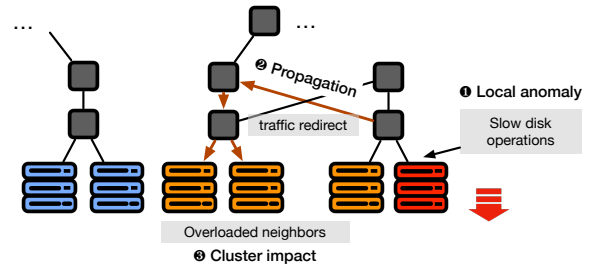


Figure 1: Unmitigated host anomalies lead to gray failures and degrade cluster health.

November 18–20, 2026, Singapore, Singapore. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3828161.3857870>

1 Introduction

Cloud infrastructure operates at massive scale. At this scale, anomalies become an everyday reality, which originate from hardware faults, software bugs, configuration issues. These anomalies are often subtle or intermittent, making them hard to detect and fix.

A seemingly local anomaly, if not mitigated, can amplify into broader cluster health degradation. Figure 1 illustrates such an example. First (1), a single storage node experiences disk pressure that leads to unusually slow I/O operations. The node remains operational but delivers degraded performance. Second (2), as requests to the affected node become slower, traffic may be redirected to other nodes in the cluster. This redirection shifts additional workload onto otherwise healthy nodes, pushing them beyond their capacity. Finally (3), the resulting overload can spread beyond the immediate neighbors as traffic is redistributed across the cluster.

Thus, timely anomaly mitigation is critical, but it is also difficult to achieve. In production mitigation often has to begin before a complete diagnosis is available. In a live-site incident, operators may leverage a telemetry query that identifies the affected machines and an action that mitigates the impact. Replaying the same query

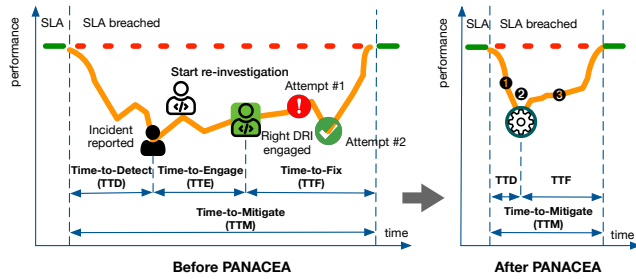


Figure 2: Mitigation practice transformation with PANACEA.
①: Signal satisfies boundary; ②: Approved mitigation starts;
③: Post-action health monitored.

and action across the fleet can be unsafe, as an action that was appropriate for one machine may be unsafe for a different workload or capacity conditions.

While extensive efforts [32, 34, 37, 39, 40] have been made to mitigate such problems in production environments, they are only tailored for specific, well-known anomalies. Several systems had already been built and deployed internally at Azure before this work. RESIN [28], which monitors memory usages and leverages heap snapshots to diagnose memory leak issues. Narya [24] monitors hardware metrics such as disk wear-out rates to predict and prevent hardware/firmware-induced failures. Still, large-scale cloud infrastructures such as Azure experience dozens of distinct types of anomalies. Building and maintaining a specialized system for each type is impractical. A general-purpose approach is needed to support all these types of anomalies.

We present PANACEA, a production framework designed to turn incident-tested mitigations into recurring fleet-wide policies. Unlike prior systems that assume a stable failure definition or detector, PANACEA targets an earlier operational stage in which a useful symptom and action are known, but the conditions under which that action should run remain uncertain. We refer to the conditions under which such an action may run automatically as its *activation boundary*. The boundary captures the operational preconditions. For each scenario, PANACEA connects scenario-specific targeting logic, a pre-approved action path, and rollout and capacity constraints, and exposes execution outcomes and post-action health evidence for operator-driven refinement of the activation boundary.

Before PANACEA, teams across Azure had already developed mitigation rules and procedures tailored to their services and failure scenarios. These practices captured valuable operational knowledge, but the conditions governing when a mitigation should be applied were often expressed differently across operational procedures. As we sought to support a broader range of mitigations across Azure infrastructure, we recognized the need to make these conditions explicit and support their continuous refinement as operational evidence accumulates. PANACEA builds on these existing practices by introducing a common abstraction for specifying when a mitigation is appropriate, while accommodating the requirements of different services and scenarios.

Figure 2 illustrates the operational benefit brought by PANACEA. Responding to an anomaly typically involves reporting the incident,

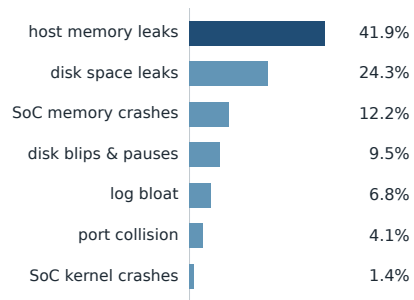


Figure 3: Root cause distribution of incidents collected in our study.

engaging the responsible team, and manually investigating and applying appropriate mitigations. These steps can prolong recovery, particularly when multiple mitigation attempts are needed. With PANACEA, once an anomaly satisfies the activation boundary of a pre-approved mitigation policy (①), the corresponding mitigation is automatically initiated (②), followed by post-action health monitoring (③). This shortens the time-to-mitigate by reducing the need for per-incident manual intervention in known failure scenarios.

Specifically, this paper makes three contributions:

Problem formulation and abstraction for ambiguous mitigations. We identify the missing operational step between an incident-tested mitigation and fleet-wide automation and introduce the activation boundary. Because the correct boundary is rarely known when a scenario first appears, we characterize an operator-driven lifecycle in which policies begin conservatively and are narrowed, broadened, escalated, or retired as production evidence accumulates. The abstraction makes the assumptions behind automation explicit and revisable, without requiring complete diagnosis or autonomous policy synthesis.

A production framework for operationalizing mitigation policies. PANACEA integrates cross-layer telemetry with existing Azure repair and deployment services, and provides the infrastructure needed to apply the same lifecycle across heterogeneous anomaly classes. It separates mitigation policy from the underlying repair backend, allowing node- and service-level interventions to share a common process for target selection, execution, and post-action analysis.

Deployment evidence and lessons from Azure. PANACEA has been deployed in production at Microsoft Azure since early 2024, operating across millions of host machines and hundreds of thousands of System-on-Chip (SoC) machines in 60 global regions. Each month, the mitigation engine executes tens of thousands of actions, including a median of 800 process restarts, 10K empty-node reboots, and 4K container-migration triggers. Over the six months ending March 2026, PANACEA's preventive actions correspond to an estimated 12% reduction in fleet-wide interruption rates.

2 Background and Motivation

2.1 Cloud Anomalies in Practice

To gain a better understanding of common anomaly patterns, we conducted an exploratory study of incident tickets handled by our

Scenario	Category	Component	Anomaly Description
Low memory	Memory	Host	Critically low memory availability, risking agent crashes.
Memory leaks	Memory	Host	Gradual memory exhaustion from faulty processes or drivers.
Memory crashes	Memory	SoC	Memory faults in SoC components causing repeated crashes.
Out-of-Memory on start	Memory	SoC	Agents OOM-killed during startup due to insufficient memory.
CPU jitter	CPU	Host	Excessive CPU cycle stealing affecting VM performance.
vDisk I/O stalls	Storage/Network	Host	Virtual disk I/O stalls; often due to unstable hardware or networks.
Low disk space	Storage	Host	Disk pressure from excessive usage.
Obsolete files	Storage	Host	Log and temp file cleanup failures that consume storage.
Blips/pauses	Storage	Host	Unexpected VM-level I/O/network interruptions degrading workloads.
Leftover downgrade files	Storage/Security	SoC	Residual downgrade files blocking service connectivity.
Service log bloat	Storage/Security	SoC	Unbounded logs consuming memory, requiring warm reset.
Port collision	Network	SoC	Unauthorized processes occupying critical service ports.
File system corruption	Kernel/Hardware	SoC	State corruption triggering kernel/user mode crashes.
Kernel crashes	Kernel/Hardware	SoC	Frequent panics in SoC firmware requiring repave or card replacement.
User-reported VM issues	Platform/Infra	VM	Automatically triggered by user-reported platform failures.
Inactive ServiceManager	Platform/Service	SoC	Failure to start critical agents due to systemd issues.

Table 1: Anomaly scenarios targeted by PANACEA (Host = Host nodes; SoC = System-on-Chip).

engineering team between October 2025 and April 2026. We categorized their root causes based on developer descriptions. Figure 3 shows the distribution of root causes for these incidents.¹ Host memory leaks account for 41.9% of reported incidents in our dataset, followed by disk space leaks (24.3%), SoC memory crashes (12.2%), disk blips and pauses (9.5%), log bloat (6.8%), and port collisions (4.1%). These incidents show that a large fraction of issues arise from host- or SoC-side anomalies that manifest as gray failures [20]: performance degradation and instability without obvious “down” signals.

Figure 3 does not capture all anomaly types that PANACEA needs to handle, partly because some anomalies are reported and tracked through other channels and thus are not reflected in this breakdown. Also, while some anomaly types occur more frequently than others, incident frequency does not necessarily reflect their impact. Even rarely reported anomalies can have significant consequences, which demands dedicated mitigation workflows and policies. Table 1 summarizes the anomaly scenarios PANACEA targets at.

2.2 Why Existing Workflows Are Insufficient

We initially set out to extend existing anomaly-detection and mitigation frameworks already deployed in Microsoft Azure, such as RESIN [28] and Narya [24]. However, our experience showed that simply generalizing these frameworks is not feasible because they target a different stage of the operational pipeline and rely on assumptions that do not hold for many of the anomalies we face.

RESIN is deliberately specialized to a single anomaly family: memory leaks. Both the failure mode and its consequences are well understood by operators. Azure engineers thus built a tailored pipeline around them: automated workflows periodically capture heap snapshots, and a pattern-matching algorithm triggers when it observes predefined memory-growth patterns. This design is effective for its target, but tightly coupled to memory leaks and does not easily carry over to other anomalies.

Narya therefore asks which mitigation works best once the failure type and its detector are fixed. It runs online experiments (e.g., A/B tests, bandits) to choose among a fixed menu of composite

actions the one that best reduces VM interruptions. However, in PANACEA, many of our hardest scenarios start earlier in the pipeline: to discover the structure of mitigation before the failure’s scope, time window, and reliable signals are clear. In other words, Narya handles problems from a different stage.

2.3 Challenges in Defining Mitigation Eligibility

Why is defining a safe fleet-wide mitigation policy so challenging? Our experience revealed several common issues in determining whether mitigation actions should be applied automatically.

Ambiguous anomaly signals. The same symptom may arise from different underlying conditions, only some of which warrant a particular mitigation. For example, an early policy for recurring network disconnections relied on the number of occurrences within the time window. However, we later observed similar event patterns following other repair workflows (e.g., backplane resets) and in cases affecting only a single subscription. The event count alone could not reliably distinguish persistent host-level problems from these other cases. Additional context, including signal persistence, repair history, and subscription scope, was often needed to identify appropriate mitigation targets.

Context-dependent action safety. Even when an anomaly is correctly identified, applying the same mitigation across different machines may be unsafe. Actions such as marking hosts unallocatable, migrating VMs, and repaving nodes can disrupt workloads. Their eligibility therefore depends not only on the detected anomaly but also on workload characteristics, ongoing operations, and available capacity. These conditions must be considered before an incident-tested action can be safely automated across the fleet.

Evolving mitigation criteria. The conditions for applying a mitigation are rarely fully understood when a scenario first emerges. Initial policies must balance the risk of unnecessary interventions against that of leaving affected machines unmitigated. For example, an early policy may often be too conservative so that the criterion excluded machines experiencing frequent but intermittent issues.

¹The total number of collected incidents is omitted due to confidentiality requirements.

Subsequent operational evidence motivated a less restrictive criterion while retaining repair-rate limits. Operational knowledge often accumulated after the initial mitigation was introduced.

3 Evolving Mitigation Policies in Production

3.1 What Constitutes a Mitigation Policy

A mitigation scenario in PANACEA combines three elements. Table 2 summarizes the policy breakdown and gives examples with a common anomaly we deal with, E17. E17 is a host-associated network-disconnection event reported by Azure’s platform telemetry. Repeated E17 events may indicate an underlying host-level networking or hardware fault that can disrupt the VMs running on the affected host.

Target-selection logic. A scenario-specific query identifies candidate machines from host, SoC, VM, or user-reported signals. The query may include thresholds, lookback windows, persistence requirements, and context such as stock keeping unit, cluster, tenant, or subscription count.

An approved action path. Each scenario invokes a repair action already exercised and reviewed by the owning engineering team. Depending on the scenario, this may be a node-level action such as marking a node unallocatable, migrating its VMs, rebooting, repaving, or replacing hardware, or a service-level script such as restarting a process or cleaning disk space. The action path also specifies how failures and timeouts are handled.

Operational constraints. A scenario includes conditions that restrict when and where the action may run, such as exclusions for particular tenants or clusters, checks for concurrent repair workflows, per-target cooldowns, and limits on concurrent repairs or affected capacity.

These elements are not expressed in a single policy language in our current deployment. Target selection is typically implemented as an analytics query, while actions and rollout constraints are configured in the corresponding Azure orchestration services. We use *policy* to refer collectively to these pieces and their operational ownership. PANACEA connects them into a recurring workflow and exposes the execution and health evidence needed to revise them.

3.2 Starting from a Conservative Policy

When a new scenario is onboarded, operators rarely have enough evidence to define its final activation boundary. They may know that a signal appeared during an incident and that a particular action helped, but not how common the signal is across the fleet, whether it has multiple causes, or which workloads make the action unsafe. The initial policy therefore targets a narrow set of severe or persistent cases. Operators retain the incident-tested action, use conservative thresholds and lookback windows, exclude known sensitive scopes and conflicting workflows, and deploy first to lower-tier or canary environments. The objective at this stage is not maximum coverage. It is to automate a small set of cases for which the action is understandable and the blast radius is bounded.

Once the policy is active, PANACEA repeatedly evaluates its targeting query and submits qualifying targets to the configured repair backend. Existing deployment mechanisms provide staged rollout,

bake time, throttling, and the ability to stop further execution when health regressions appear. Sections 4 and 5 describe these mechanisms.

3.3 Refining the Activation Boundary

Policy refinement remains operator-driven. Mitigation owners review several sources of production evidence: historical query results, orchestration logs and action outcomes, post-action health signals, incident investigations, and feedback from partner teams. They use this evidence to identify both over-mitigation—targets for which the action was unnecessary or unsafe—and under-mitigation—affected targets that the current rule selected too late or missed entirely.

The resulting changes usually fall into three categories.

Signal and persistence. Operators revise which signals count as evidence and how long they must persist. For example, a flat event count may be replaced by a requirement that the symptom occur across multiple time bins, or a metric may be replaced after its semantics are found to include benign behavior.

Scope and context. Operators add context that distinguishes a platform fault from a tenant-local symptom or an artifact of another workflow. Such changes include RCA exclusions, subscription-aware handling, cluster and tenant exclusions, and checks for recent repairs.

Action preconditions and escalation. Evidence may show that an action cannot succeed on some targets or that repeated failure warrants a stronger action. Operators then add preconditions, alter timeout handling, or escalate from a lightweight repair to repaving, investigation, or hardware removal.

Candidate changes can first be replayed over historical telemetry to estimate their target set and detection latency. They are then introduced through the same staged rollout process as the original policy. PANACEA does not autonomously synthesize or approve policy updates: operators remain responsible for interpreting the evidence and deciding whether to broaden, restrict, roll back, or retire a policy. Section 6 examines how this process changed policies for production scenarios.

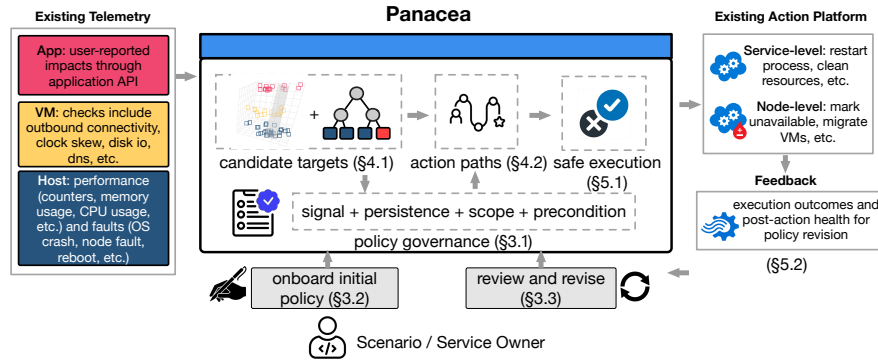
4 Operationalizing Mitigation Policies

Figure 4 shows how PANACEA operationalizes a mitigation policy over existing Azure telemetry and repair services. At a high level, PANACEA serves as a mitigation-policy control plane layered over, rather than a replacement for, these existing systems. Scenario owners define and revise mitigation policies; existing Azure services collect the underlying telemetry and perform the repair actions; and PANACEA manages policy evaluation and execution by matching policies against that telemetry, enforcing operational constraints, dispatching eligible targets to approved repair paths, and correlating the resulting evidence.

After a scenario owner onboards an incident-tested action and an initial activation boundary, PANACEA operates in three phases. First, it evaluates a scenario-specific targeting query over host-, VM-, and application-level telemetry to identify machines that satisfy the current boundary. Second, it applies operational checks, including scope exclusions, cooldowns, concurrent-workflow checks, and capacity limits, and submits eligible targets to the scenario’s

Table 2: Structure of a PANACEA mitigation policy and its lifecycle evidence, illustrated using E17, a host-side anomaly associated with recurring network disconnections.

Element	Example	Where implemented or recorded
Activation boundary. <i>When is the action eligible?</i>		
Signal evidence	Repeated E17 network-disconnection events associated with a host	PANACEA analytics query over existing platform telemetry
Persistence requirement	E17 activity in every 3-hour bin of a 12-hour window	Temporal predicates in the PANACEA targeting query
Scope and context	Treat single-subscription observations as potentially tenant-local; require persistent multi-subscription evidence for host-level mitigation	Query grouping and scenario configuration managed by PANACEA
Exclusions and runtime preconditions	Exclude events following recent backplane resets and defer machines already participating in another repair workflow	PANACEA query filters and runtime checks over existing Azure workflow state
Execution policy. <i>How is the approved action applied?</i>		
Approved action path	Mark the host unavailable and migrate or recover its VMs	Existing Azure node-repair service invoked by PANACEA
Rollout and capacity controls	Enforce scenario- and region-specific repair budgets	Existing Azure orchestration controls configured and invoked through PANACEA
Cooldown and duplicate suppression	Suppress recently mitigated targets and avoid overlapping repair requests	PANACEA eligibility checks and existing orchestration state
Escalation path	Route persistent cases to stronger hardware repair or operator investigation	Pre-approved Azure repair path and operator workflow
Lifecycle evidence. <i>How is the policy evaluated and revised?</i>		
Execution records	Workflow status, timestamps, timeouts, and structured errors	Recorded and correlated by PANACEA from the repair backend
Post-action health	Node readiness together with reboot and I/O health signals	Existing Azure monitoring systems, correlated by PANACEA
Revision inputs	Historical replay, incident investigations, execution failures, and partner-team feedback	Operator review of PANACEA-linked analytics, execution, health, and incident records

**Figure 4: Workflow of PANACEA.**

pre-approved node- or service-level repair path. Finally, it records execution outcomes and correlates them with post-action health evidence. The scenario owner reviews this evidence together with incident investigations and partner-team feedback, and may narrow or broaden the boundary, modify the escalation path, roll back a revision, or retire the policy.

4.1 From Telemetry to Candidate Targets

First, the detection workflow collects telemetry data from multiple system layers—including application, virtual machine, and host levels—to build a rich context for analysis. Second, it performs rule-based and pattern-driven analysis at the primary layer (typically the host or SoC level). At the end, PANACEA correlates findings across layers to improve precision and reduce false positives, ensuring that localized symptoms are interpreted in the context of broader system behavior.

Host/SoC Signals. The primary signals PANACEA collects are those from host and SoC. The telemetry collected includes service logs,

Windows/Linux OS events, disk/memory/CPU performance counters, and machine/process/service-level exceptions, along with their signatures. This information is being collected on over 3 million host OS nodes and 600k SoC nodes, at a frequency ranging from minute level to daily (depending on the signal type and environments). PANACEA does not deploy its own host agents on each node. Instead, it leverages a variety of host and SoC profiling services developed and installed by other Azure teams.

VM-reported Impacts. While host/SoC signals provide low-level health indicators, they often lack direct visibility into customer workloads and how failures manifest at the VM level. To address this gap, PANACEA incorporates a lightweight monitoring component deployed on Azure VMs, to gain insights into guest VM health. It is an opt-in service delivered through the application health extension (so customers can configure it to be enabled or disabled). Currently, the component includes 40 built-in checks (a subset of which are shown in Table 3). The monitoring module does periodic checks (execution interval ranges from 60 to 180 seconds) on each VM and

Signal	Description	Error Codes
outbound	Verify network outbound connectivity from the Azure VM	HttpGetFailed, HttpReadResponseFailed
clockskew	Verify clock skew between NTP server and Azure VM	ClockSkewHigh, ClockSyncFailed
disk_io	Verify file operations on each mounted drive	DiskPartitionMountFailed, DirectoryCreationFailed, FileCreationFailed, FileWriteFailed, FileReadFailed
dns	Verify DNS resolution	DnsLookupFailed
imds	Verify IMDS endpoint connectivity and VM info retrieval	ImdsEndpointUnavailable, ImdsDataParsingFailure, ImdsDataMissing
proc_init	Verify process creation	ProcessCreationFailed, ProcessExecutionTimeout, ProcessReturnedError
proc_run	Verify if target processes are running	ProcessListingFailed, ProcessNotRunning
storage	Verify PANACEA Storage Blob connectivity and download	AzBlobDownloadFailed, AzBlobAuthenticationFailed

Table 3: Examples of VM monitoring checks and common corresponding errors.

emits the pass/fail result of each check, along with the error code and diagnosis message for the failures. These signals are collected by a guest agent, and then those logs are eventually published to a column-oriented analytics platform supporting cloud storage.

User-reported Impacts. We observe that users (internal developers or customers) frequently possess more immediate awareness of service-level degradations, and their applications are often equipped with built-in mechanisms to detect and report anomalies based on domain-specific knowledge. Thus, PANACEA further allows users to report real-world service degradations with a RESTful API [31]. Users could report the performance and connectivity impact to their workloads.

When a customer reports an incident, the root cause is often unknown—it could stem from host updates, storage, networking, or other subsystems. In some cases, customers can provide symptoms such as degraded performance or connectivity issues, but these observations may be incomplete or imprecise. To enhance context, metadata and diagnostic signals from the customer’s VMs are automatically correlated and attached to the event using internal telemetry systems.

Note that the reporting API is not a replacement for support tickets—users can still file tickets as needed. It serves as a quicker method to alert the system to potential issues.

4.2 Mapping Targets to Approved Action Paths

Once a target satisfies a scenario’s activation boundary, PANACEA invokes the scenario’s pre-approved action path. The path may comprise a multi-step node- or service-level workflow, while PANACEA enforces rollout, cooldown, workflow-conflict, and capacity constraints and records the resulting execution and health evidence for later operator review.

Table 4 summarizes the types of mitigation actions. These actions are broadly divided into node-level and service-level interventions.

Node-level interventions are the broader, higher-level actions PANACEA takes, like rebooting, reallocating, or isolating high-risk

	Action	Example Scenarios
Node	Mark as Unavailable	Blips/pauses
	Reboot nodes	Memory leaks
	Warm Reset	Service log bloat
	Repave	Kernel crashes
Service	Process restarts	Memory leaks
	Freeze/Thaw agents	Out-of-Memory on start
	Resource cleanup	Low disk space, Obsolete files
	Resource pre-allocation	Port collision

Table 4: Mitigation action types and example scenarios.

nodes, along with system state integration with the VM allocator. For empty nodes, PANACEA performs maintenance operations such as rebooting, or change the status of high-risk nodes. Non-empty nodes will be marked as unallocatable. Any kind of reboot is always preceded by mark unallocatable to try to move VMs off the node. The system integrates availability state with Azure global VM allocator. PANACEA attempts to migrate all VMs on affected hosts. All the VMs that are live migratable will migrate off the node. Once the node transition from unallocatable to human investigation following the policy, service recovery will be done on the remaining VMs and the node becomes empty. Then the node is rebooted and if there are no issues on the node, it will transition to Available state. This type of mitigation is particularly used for hardware faults, which typically takes around 15-30 minutes.

Service-level interventions are finer-grained actions implemented as service-owned scripts, such as cleaning disk space or reserving resources. Many anomalies require a short sequence of such actions. For example, when reserved port collisions occur, PANACEA identifies unauthorized processes occupying ports reserved for critical services and, under predefined safety rules, decides whether to terminate the offender, relocate it, or leave it untouched. For agents that are repeatedly OOM-killed during startup, PANACEA temporarily freezes selected services, restarts systemd to reclaim memory, adjusts memory limits, and then resumes the affected processes.

These interventions are only effective when backed by robust, well-tested scripts. PANACEA collaborates with developer teams to generalize existing production-ready scripts and integrate them into the mitigation workflow. Scripts are typically shell programs that implement a simple contract: they accept a single mitigation target identifier (e.g., a SoC ID) and optional hyperparameters (e.g., a target drive for disk cleanup), and emit structured logs for success and failure. The orchestrator constrains each script to operate only on the target type declared in the mitigation policy. Idempotence and failure handling are validated through manual testing during initial rollout, and PANACEA enforces per-target cooldown windows (e.g., on the order of a day) to avoid repeatedly invoking the same mitigation before its effects can be observed.

For each mitigation request, PANACEA records the target, policy and action identifiers, execution timestamps, completion status, and structured errors reported by the repair backend. These records are linked with post-action health signals and exposed to scenario owners for policy review. Section 5 describes how PANACEA uses this evidence to detect regressions and support safe policy rollout.



Figure 5: Process of adding new service-level mitigation actions.



Figure 6: Stage-by-stage mitigation deployment.

5 Safe Execution and Feedback

Figure 5 illustrates the process of integrating new service-level mitigation actions. When new anomaly scenarios arise—often reported by partner engineering teams—the core mitigation actions are jointly defined. The PANACEA team then develops platform-compatible scripts and constructs the necessary virtual environments, including logging infrastructures and automation workflows. This process ensures that new mitigations are both effective in the short term and adaptable for future incidents.

5.1 Safe Execution and Rollout

Safe development and testing. Due to the extensibility of custom scripts, safety is top-of-mind during mitigation development. Engineers from relevant teams author the fabric commands that interact directly with the mitigation target, ensuring that the collaboration fully leverages their expertise.

Completed mitigation scripts are tested on stage and canary nodes, which simulate realistic production circumstances without bearing any customer workload and thus avoid risk. When the issue being mitigated is artificially replicable, test-in-production nodes are manipulated into conditions qualifying them as targets for mitigation. When this is not possible (e.g., when the root cause of the issue has not yet been determined), a script might be run manually (and with careful and extended observation) on a genuine target.

To complete onboarding a tested mitigation script to PANACEA for automation, the repository housing the source code runs certain checks upon submission. These are primarily syntactic tests that prevent typos and ensure proper formatting for eventual orchestration across targets. Additionally, health monitoring throughout orchestration provides a safety check that is ongoing.

Orchestration. PANACEA orchestrates the deployment of mitigation actions in a controlled and structured manner across the Azure infrastructure. Two internal platforms support this orchestration. A fully-managed ring-0 service handles node-level interventions by executing predefined workflows to remediate live site incidents. Concurrently, an agile platform supports customized service-level actions to address urgent issues.

Execution platforms apply mitigation scripts to queued targets under standard safe deployment practices (SDP). For each scenario, PANACEA first onboards a mitigation policy: a PowerShell script

plus metadata (target type, action category) or a JSON policy that specifies a tree of actions with success/error handling and target criteria. When targets are identified, PANACEA submits a JSON request listing the targets and the policy to execute. If there is no backlog, mitigations typically start within a few minutes; otherwise, the platform derives a rollout plan that batches targets by production stage and region, runs batches nearly in sequence, and inserts configurable bake times and completion thresholds so that health can be monitored and rollout halted on regression. The system treats success as a binary outcome and allows retries several hours after a failed attempt when a target is resubmitted, while scripts themselves log partial failures via structured success and error streams.

Early deployments used two orchestration styles: batch-oriented workflows for urgent, short-lived mitigations and persistent repair policies for long-running scenarios. This experience showed that the backend must match how targets arrive. When all targets are known up front, batch orchestration can safely order them and insert bake times between groups. When targets arrive continuously, as in most PANACEA scenarios, the system needs persistent policy onboarding, per-scope rate limits, and cross-scenario coordination for shared repair actions. This led PANACEA to favor the latter model for long-running automated mitigation, while retaining the broader design principle that mitigation policy should be separated from the repair backend.

Safety is a first-order requirement for cloud infrastructure. To minimize blast radius while still enabling fast iteration, PANACEA adopts a stage-by-stage rollout model (Figure 6). Each new or updated mitigation policy is first validated in a lowest-tier environment and then promoted through progressively larger scopes (stage → canary → pilot → broad), with explicit approvals and bake-time observation between tiers. This setup ensures that regressions and unexpected interactions with other workflows are exposed early, before a policy can affect a non-trivial fraction of production capacity.

In practice, the tiers differ substantially in scale, and promotion is gated by evidence from the previous tier. For example, for the low-memory scenario, a single scheduled run may affect only a handful of nodes in canary, then expand to hundreds of nodes in pilot, and later to a broader production scope once the mitigation demonstrates stable post-action health and no customer complaints. When a policy reaches production clusters, PANACEA additionally enforces throttling and concurrency limits to cap the number of simultaneous hosts and services under mitigation.

5.2 Post-Mitigation Feedback

Monitoring. To ensure node stability, PANACEA uses intervention analysis to evaluate mitigation impact. It monitors health signals (e.g., unexpected reboots, disk I/O anomalies) before and after mitigation via a Kolmogorov-Smirnov test [5, 11] to compare the distributions of intervals between events. The Kolmogorov-Smirnov test assesses whether two empirical distributions (i.e., of timespans between consecutive health failures) were likely to have been sampled from the same underlying distribution—without needing to specify or model that underlying distribution. A statistically significant difference in the distributions (i.e., from before the mitigation and

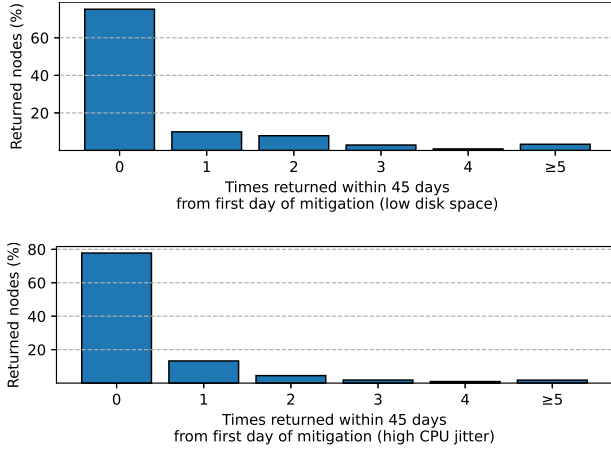


Figure 7: Return times distributions.

from after the mitigation), specifically in the direction with the after-mitigation distribution showing more density among shorter timespans between consecutive failures, indicates a health regression. If the test's p-value falls below a fixed significance threshold, PANACEA flags the target as regressed. When multiple regressions are linked to the same mitigation policy, PANACEA halts further mitigation rollout pending investigation. Notably, this health monitoring is not used to verify mitigation success, only to assess whether the mitigation worsens the health of its targets. Since a given mitigation policy might not be intended to alleviate particular comorbidities of its targets, persistence of health issues after mitigation does not implicate the mitigation. Only when health trends among targets statistically regress after mitigation relative to before mitigation, the mitigation policy is considered suspect.

Post-deployment refinement and reporting. After the deployment, PANACEA continuously monitors the effectiveness of deployed mitigation plan. A key metric is the return rate: the fraction of UA target nodes that revert to the anomalous state within a set number of days (by default 45). This metric provides long-term insight into the effectiveness of the mitigation strategy. A part of the workflow is to collect detailed post-mitigation analysis to enhance future mitigation strategies (script updates and parameter tuning) based on collected fault metadata and operational feedback. Figure 7 shows the return-frequency distribution of mitigated nodes within 45 days for low disk space and high CPU jitter. In both cases, most nodes never return within the window, and the fraction of nodes returning multiple times is small.

The primary goal of PANACEA is to detect anomalies and mitigate them to prevent customer impact. Meanwhile, an important foundational principle is to further analyze these mitigations, identify root causes, and ultimately drive quality improvements on the fleet. Toward this end, in addition to performing mitigations for detected anomalies, alerts are also filed for the corresponding RCAs and are filed to the owning team, where applicable. The owning team is then accountable for fixing the underlying issue so that the mitigation is not performed indefinitely.

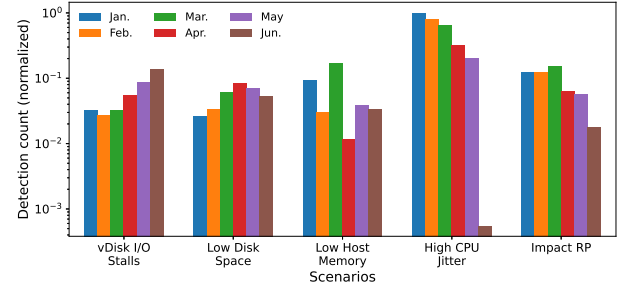


Figure 8: PANACEA detections for different scenarios by month in 2026.

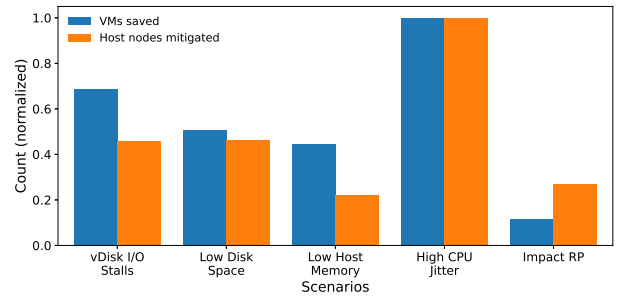


Figure 9: Host nodes mitigated & VMs saved.

6 Evaluation

We evaluate the performance of PANACEA in Azure production environments. Our evaluation answers several questions: (1) How widely is PANACEA currently deployed in Azure? (2) How does PANACEA impact cloud reliability end-to-end? (3) How does PANACEA help operators evolve mitigation policies? (4) How effective is PANACEA's mitigation? (5) How timely are PANACEA's mitigation actions?

6.1 Deployment Status and Scale

To characterize the scope of mitigation activity performed by PANACEA, we analyze system logs to identify and count individual mitigation actions. Each request is tracked by a unique ID and considered successful when all associated actions complete. Nodes with successful mitigations are counted as mitigated, and any VM present at the time are considered saved if live migration or service recovery was applied, as shown in Figure 8 and Figure 9. Due to data confidentiality requirements, we normalize the total number of detected outages. The high CPU jitter counts in June were low because sensors were affected by rolling update, thus some incidents were detected but not correctly calculated (we fixed the issue and re-enabled the sensor in July). From April 2025 to March 2026, each month, PANACEA executes tens of thousands of mitigation actions, including a median of 800 process restarts, 10k empty node reboots, and 4k containers preemptively migrated from nodes to prevent unexpected reboots. The detection engine runs 2–10 times per hour per scenario, issuing mitigations for each batch of detected targets.

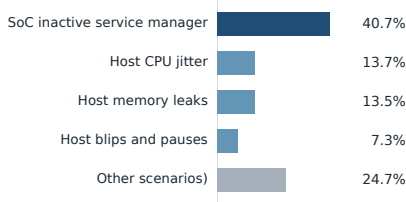


Figure 10: Annual Interruption Rate (AIR) saving breakdown.

6.2 End-to-End Impact

Metrics. Annual Interruption Rate (AIR) is Azure’s primary fleet-level reliability metric for customer-visible VM interruptions and has also been used in prior mitigation work, including Narya [24] and RESIN [28]. PANACEA is evaluated using the same production reliability accounting used by the corresponding Azure service teams. It is calculated as the frequency of interruptions experienced per 100 VMs over the course of a year. AIR captures both the occurrence and distribution of interruptions across the fleet and normalizes by total uptime, making it robust to changes in deployment scale or VM lifetimes. A reduction in AIR directly indicates fewer service disruptions experienced by tenants.

$$\text{AIR} = \frac{\text{\#Interruptions}}{\text{Total UP duration (min) of measured VMs}} \times 365 \times 24 \times 60 \times 100$$

Results and methodology. Over the six months ending in March 2026, we estimate that PANACEA reduced the average monthly fleet-wide AIR by 12%. We describe our estimation methodology below. First, we correlated actual actions with instances where the mitigation would have been triggered, over a time window prior to when we enabled the mitigation. This gave us additional metadata on the conditions when the anomaly would result in actual impacts such as VM reboot. Then we incorporated this additional metadata in our estimation of AIR reduction for this scenario. For similar scenarios where there was a potential for overestimating impact, we analyzed and excluded specific cases within the scenarios that would not result in savings, from the AIR reduction calculation, thereby improving precision of the estimate. Formally, let AIR_{np} be the AIR after adding these saved VMs as interruptions and let AIR_p be the observed AIR with PANACEA deployed. The reported 12% reduction is then the relative difference $\frac{\text{AIR}_{np} - \text{AIR}_p}{\text{AIR}_{np}}$, averaged over the 6-month deployment period.

Sensitivity analysis. Our estimation is already conservative, as we use historical observations to identify and exclude cases unlikely to result in actual VM interruptions. Nevertheless, uncertainty remains in estimating which interruptions would have occurred without PANACEA. To assess the sensitivity of our results to this uncertainty, we discount the estimated avoided interruptions by a factor of $\alpha \in \{1.0, 0.75, 0.5\}$, where α represents the fraction of the originally estimated interruption savings retained in the counterfactual calculation. We then recompute the monthly AIR reductions under each assumption. The resulting six-month average AIR reductions are 12%, 9.3%, and 6.4%, respectively.

Distribution of contributions. We also analyze how different mitigation scenarios contribute to the total reduction. Figure 10 breaks

Table 5: Two examples of policy evolution in PANACEA.

Scenario	Policy evolution and evidence
E17	Initial boundary: nodes with ≥ 10 E17 events in an 18h window. Problem observed: some selected nodes reflected repair-induced bursts, delayed telemetry after another workflow had already fixed the issue, or single-subscription spikes that did not indicate a platform fault. Refined boundary: filter repair-induced RCAs, require multiple subscriptions, and require E17s to persist across 3h bins in a 12h window. Outcome: more than 60% improvement in detection precision.
Kernel crashes	Initial boundary: nodes that crashed on all of the last 10 days. Problem observed: the rule was too strict and missed repeat offenders that crashed frequently but not every day. Refined boundary: select nodes that crashed on at least 3 days in the last 5 days, subject to approved repair-rate limits. Outcome: AIR daily savings increased from 0.008 to 0.039, and time-to-mitigate dropped 48%.

down the distribution of AIR savings across scenarios. The largest contributor is inactive ServiceManager (40.7%), followed by CPU jitter (13.7%), host memory leaks (13.5%), and host blips and pauses (7.3%). The remaining scenarios collectively account for 24.7% of the savings.

Interpreting AIR savings. Lower AIR savings for a scenario do not necessarily imply that the scenario is less important. AIR captures interruptions, but some gray failures primarily manifest as successful-but-slow operations or transient performance degradation. For this reason, we view AIR as a conservative measure of mitigation impact and are adding AIR-BP (Annual Interruption Rate–Blips & Pauses) as a complementary metric for capturing broader gray-failure impact.

6.3 Impact of Activation-Boundary Refinement

We next evaluate how PANACEA helps operators evolve mitigation policies as production evidence accumulates. We study policy revisions in which the mitigation action remains unchanged while its activation boundary changes. In our deployment, policy evolution rarely changes the mitigation action itself. Instead, it refines the boundary under which a known action is safe and useful: which signals are trustworthy, which targets are in scope, how long symptoms must persist, and when a target should be excluded because another workflow or repair precondition applies.

Table 5 summarizes two representative examples. They illustrate two common forms of policy evolution. The E17 case shows how operators refined the scope and temporal boundary of a policy to avoid acting on tenant-local or repair-induced artifacts. The kernel-crash case shows how operators relaxed an overly conservative temporal boundary to reduce mitigation latency while staying within approved safety limits.

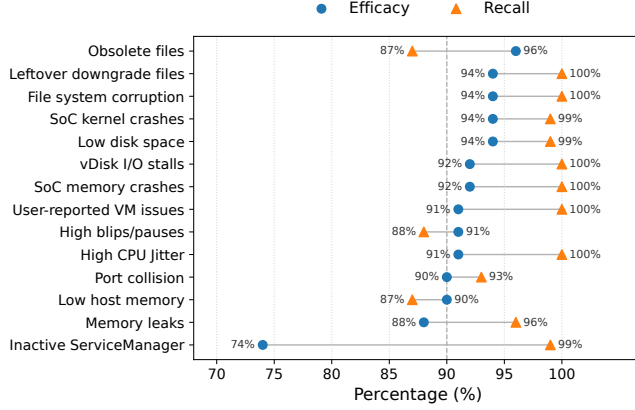


Figure 11: Efficacy and recall across mitigation scenarios. PANACEA achieves high recall across scenarios and around 90% efficacy for most scenarios.

6.4 Mitigation Efficacy, Recall and Precision

Besides end-to-end impacts, we follow up to measure the quality of our mitigation actions. We analyzed the logs and metrics collected for all mitigation actions performed across different scenarios between October 2025 and March 2026. Figure 11 presents the results. Two newly supported scenarios lacked sufficient data to reliably evaluate efficacy and recall during the study period and are therefore omitted from the figure. We analyze the complete set of recorded mitigation actions rather than a sample, avoiding sampling uncertainty in these action records. However, recall estimates remain subject to incomplete ground truth, as incidents not surfaced through existing reporting or monitoring channels cannot be counted.

Efficacy. Mitigation is intended not only for immediate service restoration, but also to prevent recurrence of the same failure in the short term. We evaluate it by measuring how often mitigated nodes reenter the bad state for the same scenario. We define efficacy as:

$$\text{Efficacy} = 1 - \left(\frac{N_{\text{return}}}{N_{\text{mitigated}}} \right)$$

where N_{return} is the number of nodes that return to an unallocatable state for the same scenario within a predefined window, and $N_{\text{mitigated}}$ is the total number of unallocatable nodes that were mitigated. The time windows vary as some scenarios require longer monitoring. A higher efficacy score indicates fewer recurrences and, therefore, greater long-term impact of the mitigation strategy. For example, mitigating out-of-memory issues during service startup not only restarts services but also adjust system memory limit. While not all cases will be successfully mitigated, this is expected due to two main factors: (1) mitigations are sometimes incomplete caused by cloud environment issues, such as unexpected hardware/system issues or interaction effects between anomalies; and (2) some root causes evolve over time and not fully addressed by the original mitigation.

Recall. In particular, we consider incidents where a node should have been mitigated but was missed due to the detection gaps—these are counted as false negatives. These incidents are reported by users, developers, or other monitoring services before PANACEA mitigates them. We search related keywords for each scenario (e.g., disk space) in the issue tracker used by all teams and exclude those from PANACEA detection. We compare them with PANACEA logs. Overall the false negatives are few and most recall results are higher than 90%. The missed cases are mainly due to new patterns not captured by the current policy. Thus, we proactively monitor such instances and update policies so they are caught in future.

Precision. Measuring precision in production settings is in general challenging due to the lack of ground truth—there is rarely a definitive label indicating whether a mitigation was necessary. Unlike controlled environments or benchmarks, real-world systems do not provide clear annotations for false positives, making precision difficult to quantify reliably. In our implementation, we usually cross-reference different sources to adjust our policies to eliminate unnecessary mitigation targets. Manually collecting and analyzing for all scenarios is too labor-intensive. Therefore, we specifically focus on labeled scenarios PANACEA mitigated in the reported data collection period and follow up developer feedbacks. Here we report statistics for scenarios we collect accurate labels.

Interestingly we found that the precisions are often related to scenarios. The precision for some scenarios remain stable. For example the average precision of memory leak mitigation and port collision are 95% and 82%, respectively. For some other scenarios, we observed that precision could be affected by the process of fixing. In a six-month historical evaluation of E17 (October 2024–March 2025), precision varied substantially due to transient, non-platform-related events. Following revisions to the detection logic and mitigation policy, precision reached 100% in the final two months, with an overall precision of 65% across the period. This trajectory is consistent with PANACEA’s role as a temporary mitigation layer and reinforces the need to revisit activation boundaries as permanent fixes remove the underlying faults.

6.5 Timeliness

Timeliness is critical in mitigating gray failures, as prolonged exposure can degrade performance, disrupt tenants, or cause cascading faults. We evaluate timeliness along two key dimensions: Time-to-Detect (TTD) and Time-to-Fix (TTF). They compose Time-to-Mitigate (TTM), which represents the total latency from the onset of impact to mitigation. PANACEA successfully reduces median TTM by more than 80% across multiple scenarios compared with prior manual workflows. Figures 12a and 12b show the distributions of time to mitigate (TTM) for two scenarios: SoC port collision and obsolete-file cleanup. Most mitigation actions complete within 30 minutes. The long tail is primarily caused by scenario-specific execution delays, including contention with active workloads and variability in the amount of data to be cleaned.

6.6 Effectiveness of Mitigation Guardrails

PANACEA uses post-mitigation health monitoring to identify potential side effects that warrant investigation. In a two-month review of August and September 2026, we compared intervals between

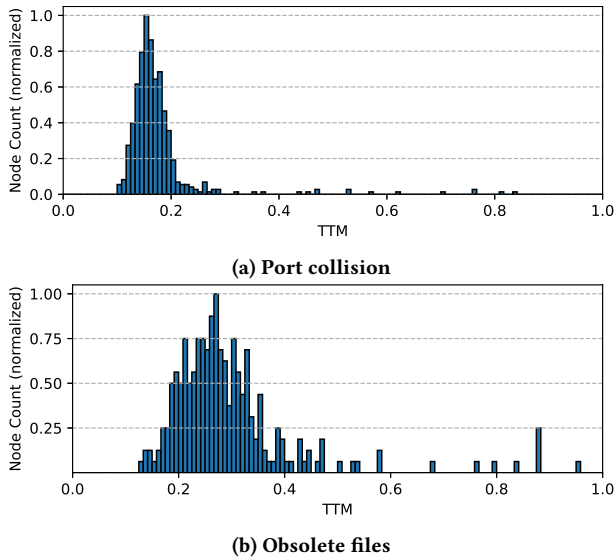


Figure 12: Distribution of mitigation time for two scenarios. We normalize TTM in each scenario by dividing the measured time by the maximum TTM observed for that scenario.

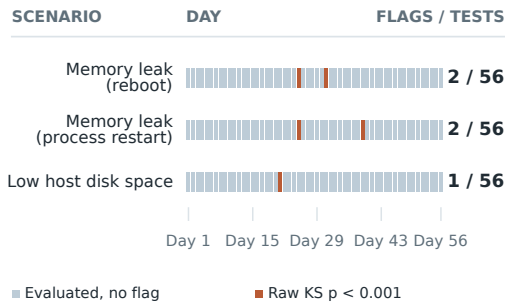


Figure 13: Post-mitigation health monitoring using the KS test. Each square represents one daily test, with orange indicating a significant flag ($p < 0.001$).

unhealthy-VM observations in the hour before and after mitigation, pooling targets by scenario, action, and UTC calendar day; 5 of 168 evaluated daily groups (3.0%) crossed the one-sided KS significance threshold of 0.001. We show the results in Figure 13. A deep dive into one low-disk-space flag identified 24 post-mitigation observations across nine VMs on 2 of the 9 nodes represented in the daily test. One node had documented log-driven disk pressure before mitigation and subsequently reported unhealthy states during the post-mitigation monitoring window; on the other, all 7 contributing VMs had already exhibited unhealthy states earlier that day, though not during the one-hour baseline. We therefore interpret this flag as a warning requiring operational context and as an indication that a longer historical baseline might better capture recurring problems obscured by short, quiet pre-mitigation observation windows.

7 Discussion

7.1 Experience and Lessons

Operating PANACEA in Azure exposed several failure modes that were not apparent when individual mitigation policies were first written. The most useful lessons came from policies that appeared reasonable in isolation but behaved poorly when they interacted with other components in the workflow.

A target cannot be defined from the anomaly signal alone. The low-memory policy exposed a related problem. Some clusters hosted single containers that continued to operate normally despite low host memory, but these containers could not be live migrated. Applying the default policy caused service recovery and performance impact. We consequently excluded these clusters pending a custom mitigation policy. After these incidents, we stopped treating a target as simply a node that crossed a telemetry threshold. The policy also has to account for concurrent repair workflows and constraints of the workloads running on that node.

Telemetry semantics and delay are part of the policy. Some production mistakes were caused neither by an incorrect repair action nor by an obviously poor threshold, but by the data supplied to the policy. A false positive in the CPU-jitter scenario came from the definition of the jitter signal. The original metric included short scheduling delays caused by waking idle processors. Workloads issuing many small I/Os amplified these otherwise benign delays, making a node appear persistently jittery. We introduced a new metric that filters this behavior and validated it against the machines involved in the incident.

The opposite problem occurred in the low-disk-space scenario. Telemetry could arrive roughly minutes late, while the hourly detection query assumed that recent samples had already been ingested. As a result, a few low-disk nodes were missed and were not mitigated in time. We revised the query to account for this delay. These incidents led us to validate signal definitions, timestamps, and ingestion latency together with policy thresholds. A policy that is correct over idealized data may still be unsafe or ineffective over the data available when the decision is made.

A temporary mitigation needs an owner and a path to retirement. We experienced this problem when evolving the disk-cleanup scenario. The inherited script worked, but it invoked cleanup rules that had existed for years with incomplete documentation of why individual paths were safe to delete or which team owned the underlying disk growth. Before expanding the mitigation, we had to recover this context and identify the owners of service-specific paths. Generic temporary directories remained under cleanup when no meaningful owner could be identified. For service-owned paths, however, the preferred outcome was for the owning team to fix the source of the growth so that the cleanup rule could eventually be removed. The platform provides functionalities to manage the policy, but service teams continue to own the repair logic and the decision to retire it. Without explicit ownership, a successful short-term mitigation can quietly become permanent, undocumented production behavior.

7.2 Limitations and Future Work

PANACEA does not automatically derive mitigation policies. Operators still decide which signals are meaningful or whether an action is safe for a particular scope, which can require substantial manual investigation. A useful next step is to better connect execution results to policy hypothesis for operator review.

The system feedback is limited by incomplete ground truth. Production systems do not automatically provide a definitive label indicating that a mitigation was necessary. The reported-incident coverage only includes failures that were surfaced through tickets or other monitoring systems, and comparable precision labels were available for only a small number of scenarios. More consistent and automated follow-up on mitigation outcomes would better improve our system.

The implementation of PANACEA is built on top of Azure-specific telemetry and deployment infrastructure. For example, the logs used by PANACEA are stored and queried using Azure's internal data systems. A different cloud would use its own corresponding systems instead. Some specific inputs may not have a direct equivalent. We do not know whether other clouds provide something similar to our user-reported impact API. Meanwhile, we believe the main ideas in PANACEA are cloud-agnostic and the underlying idea of our system is appropriate still applies.

8 Related Work

As cloud systems grow in complexity, failures have become increasingly subtle [2–4, 10, 14, 15, 20, 25, 29, 33]. Prior work has characterized several manifestations of such failures. Gunawi *et al.* investigated fail-slow performance faults in hardware [15]. Alquraan *et al.* analyzed failures induced by network partitions [3]. Huang *et al.* examined metastable failures that drive systems into overloaded states [18], while Lou *et al.* studied silent failures that violate system semantics [30]. These works explain why production failures may lack a clear fail-stop manifestation. PANACEA focuses on mitigating underlying anomalies before they escalate into complex failures.

Unlike server crashes [1, 7–9, 12, 16, 17, 36], many cloud anomalies do not have a single stable signal. One line of research detects anomalies by mining temporal patterns from historical telemetry [6, 13, 35]. RESIN [28], for example, uses the semantics of a known failure family to detect memory leaks at cluster scale. Another line of work closes the observation gap between systems and applications [21–23]. Panorama [19] converts external components into logical observers, while OmegaGen [29] automatically generates component-specific watchdogs. Correlation systems further help interpret production evidence: Gandalf [26] correlates fault signals with ongoing rollouts to identify potential causes, and AND [38] uses component interactions to route anomalies to responsible teams. These systems improve the evidence available for detection, diagnosis, or attribution. PANACEA may consume such evidence, but addresses a different question: when the same symptom can have multiple operational interpretations, when and where is a known action appropriate?

Prior systems automate mitigation for specific failure domains, including data-center network failures [34, 37, 39, 40]. Narya [24] prevents VM interruptions caused by hardware- and firmware-induced failures. Swarm [32] ranks different mitigations based on

their impact based on the end-to-end connection-level performance (CLP) metrics. In contrast, PANACEA proposes a general abstraction and supports a broader range of anomalies.

An early version of this project appeared in our workshop paper [27], which introduced cross-layer telemetry collection and mitigation orchestration in Azure. Since then, our deeper understanding of the problem has led us to “reformulate the core challenge around the evolution of mitigation policies” and to “co-design the system mechanisms that support this evolution”, enabling incident-tested mitigation actions to become safe, recurring policies at fleet scale.

9 Conclusion

Mitigating cloud infrastructure anomalies requires acting before an anomaly is fully understood, and it should still avoid unnecessary disruption to healthy workloads. PANACEA addresses this tension by turning the implicit applicability assumptions behind an incident-tested action into an explicit activation boundary that can be executed conservatively and revised as production evidence accumulates. Deployed in Azure for more than two years, PANACEA handles anomalies from millions of hosts and SoC machines and is estimated to reduce fleet-wide AIR by 12% over the six-month evaluation window.

Acknowledgments

We thank our shepherd and anonymous reviewers for insightful reviews. We thank our colleagues across Azure and Microsoft for their collaboration and feedback, including but not limited to, Ayush Singhal, Sanjid Halim, Carlos Cardenas, Alain Gefflaut, Sanjay Bhatt, Zheng Mu, Abhay Ketkar, Haoyang Ren, Mahesh Nemichand, Francis David, Cong Chen, Yingnong Dang, and Mahesh Dugdiker. Chang Lou is supported by CNS-2441284.

References

- [1] Marcos K. Aguilera and Michael Walfish. 2009. No Time for Asynchrony. In *Proceedings of the 12th Conference on Hot Topics in Operating Systems* (Monte Verità, Switzerland) (HotOS '09). USENIX Association, Berkeley, CA, USA, 3–3. <http://dl.acm.org/citation.cfm?id=1855568.1855571>
- [2] Mohammed Alfatafta, Basil Alkhatib, Ahmed Alquraan, and Samer Al-Kiswany. 2020. Toward a Generic Fault Tolerance Technique for Partial Network Partitioning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 351–368. <https://www.usenix.org/conference/osdi20/presentation/alfatafta>
- [3] Ahmed Alquraan, Hatem Takruri, Mohammed Alfatafta, and Samer Al-Kiswany. 2018. An Analysis of Network-Partitioning Failures in Cloud Systems. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation* (Carlsbad, CA, USA) (OSDI'18). USENIX Association, USA, 51–68.
- [4] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. 2001. Fail-Stutter Fault Tolerance. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS '01)*. IEEE Computer Society, Washington, DC, USA, 33–38. <http://dl.acm.org/citation.cfm?id=874075.876394>
- [5] Vance W Berger and YanYan Zhou. 2014. Kolmogorov-smirnov test: Overview. *Wiley statsref: Statistics reference online* (2014).
- [6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41, 3, Article 15 (July 2009), 58 pages. <https://doi.org/10.1145/1541880.1541882>
- [7] Tushar Deepak Chandra and Sam Toueg. 1996. Unreliable Failure Detectors for Reliable Distributed Systems. *J. ACM* 43, 2 (March 1996), 225–267. <https://doi.org/10.1145/226643.226647>
- [8] Wei Chen, Sam Toueg, and Marcos Kawazoe Aguilera. 2002. On the Quality of Service of Failure Detectors. *IEEE Trans. Comput.* 51, 5 (May 2002), 561–580. <https://doi.org/10.1109/TC.2002.1004595>
- [9] Abhinandan Das, Indranil Gupta, and Ashish Motivala. 2002. SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol. In *Proceedings of the 2002 International Conference on Dependable Systems and*

- Networks (DSN '02)*. IEEE Computer Society, Washington, DC, USA, 303–312. <http://dl.acm.org/citation.cfm?id=647883.738420>
- [10] Thanh Do, Mingzhe Hao, Tanakorn Leesatapornwongs, Tirat Patana-anake, and Haryadi S. Gunawi. 2013. Limplock: Understanding the Impact of Limpware on Scale-out Cloud Systems. In *Proceedings of the 4th Annual Symposium on Cloud Computing* (Santa Clara, California) (SOCC '13). ACM, New York, NY, USA, Article 14, 14 pages. <https://doi.org/10.1145/2523616.2523627>
 - [11] Giovanni Fasano and Alberto Franceschini. 1987. A multidimensional version of the Kolmogorov-Smirnov test. *Monthly Notices of the Royal Astronomical Society* 225, 1 (1987), 155–170.
 - [12] Christof Fetzer. 2003. Perfect Failure Detection in Timed Asynchronous Systems. *IEEE Trans. Comput.* 52, 2 (Feb. 2003), 99–112. <https://doi.org/10.1109/TC.2003.1176979>
 - [13] Qiang Fu, Jian-Guang Lou, Qing-Wei Lin, Rui Ding, Dongmei Zhang, Zihao Ye, and Tao Xie. 2012. Performance Issue Diagnosis for Online Service Systems. In *2012 IEEE 31st Symposium on Reliable Distributed Systems*. 273–278. <https://doi.org/10.1109/SRDS.2012.49>
 - [14] Haryadi S. Gunawi, Mingzhe Hao, Riza O. Suminto, Agung Laksono, Anang D. Satria, Jeffry Adityatama, and Kurnia J. Eliazar. 2016. Why Does the Cloud Stop Computing?: Lessons from Hundreds of Service Outages. In *Proceedings of the 7th ACM Symposium on Cloud Computing (SoCC)* (Santa Clara, CA, USA). 1–16. <https://doi.org/10.1145/2987550.2987583>
 - [15] Haryadi S. Gunawi, Riza O. Suminto, Russell Sears, Casey Golliger, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, Gary Grider, Parks M. Fields, Kevin Harms, Robert B. Ross, Andree Jacobson, Robert Ricci, Kirk Webb, Peter Alvaro, H. Birali Runesha, Mingzhe Hao, and Huaicheng Li. 2018. Fail-slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems. In *Proceedings of the 16th USENIX Conference on File and Storage Technologies* (Oakland, CA, USA) (FAST'18). USENIX Association, Berkeley, CA, USA, 1–14. <http://dl.acm.org/citation.cfm?id=3189759.3189761>
 - [16] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. 2007. PeerReview: Practical Accountability for Distributed Systems. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles* (Stevenson, Washington, USA) (SOSP '07). ACM, New York, NY, USA, 175–188. <https://doi.org/10.1145/1294261.1294279>
 - [17] Andreas Haeberlen and Petr Kouznetsov. 2009. The Fault Detection Problem. In *Proceedings of the 13th International Conference on Principles of Distributed Systems* (Nimes, France) (OPDIS '09). Springer-Verlag, Berlin, Heidelberg, 99–114. https://doi.org/10.1007/978-3-642-10877-8_10
 - [18] Lexiang Huang, Matthew Magnusson, Abishek Bangalore Muralikrishna, Salman Estyak, Rebecca Isaacs, Abutalib Aghayev, Timothy Zhu, and Aleksey Charapko. 2022. Metastable Failures in the Wild. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*. 73–90.
 - [19] Peng Huang, Chuanxiong Guo, Jacob R. Lorch, Lidong Zhou, and Yingnong Dang. 2018. Capturing and Enhancing In Situ System Observability for Failure Detection. In *13th USENIX Symposium on Operating Systems Design and Implementation* (Carlsbad, CA) (OSDI '18). USENIX Association, 1–16. <https://www.usenix.org/conference/osdi18/presentation/huang>
 - [20] Peng Huang, Chuanxiong Guo, Lidong Zhou, Jacob R. Lorch, Yingnong Dang, Murali Chintalapati, and Randolph Yao. 2017. Gray Failure: The Achilles' Heel of Cloud-Scale Systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS XVI)*. ACM, British Columbia, Canada, 7.
 - [21] Joshua B. Leners, Trinabh Gupta, Marcos K. Aguilera, and Michael Walfish. 2013. Improving Availability in Distributed Systems with Failure Informers. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation* (Lombard, IL) (NSDI '13). 427–442. <http://dl.acm.org/citation.cfm?id=2482626.2482667>
 - [22] Joshua B. Leners, Trinabh Gupta, Marcos K. Aguilera, and Michael Walfish. 2015. Taming Uncertainty in Distributed Systems with Help from the Network. In *Proceedings of the Tenth European Conference on Computer Systems* (Bordeaux, France) (EuroSys '15). ACM, New York, NY, USA, Article 9, 16 pages. <https://doi.org/10.1145/2741948.2741976>
 - [23] Joshua B. Leners, Hao Wu, Wei-Lun Hung, Marcos K. Aguilera, and Michael Walfish. 2011. Detecting Failures in Distributed Systems with the Falcon Spy Network. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles* (Cascais, Portugal) (SOSP '11). 279–294.
 - [24] Sebastian Levy, Randolph Yao, Youjiang Wu, Yingnong Dang, Peng Huang, Zheng Mu, Pu Zhao, Tarun Ramani, Naga Govindraj, Xukun Li, Qingwei Lin, Gil Lapid Shafir, and Murali Chintalapati. 2020. Predictive and Adaptive Failure Mitigation to Avert Production Cloud VM Interruptions. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20)*. USENIX.
 - [25] Zhenyu Li, Angting Cai, and Chang Lou. 2026. Pilot Execution: Simulating Failure Recovery In Situ for Production Distributed Systems. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI '26)*. USENIX Association, Renton, WA, 113–129. <https://www.usenix.org/conference/nsdi26/presentation/li-zhenyu>
 - [26] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, and Murali Chintalapati. 2020. Gandalf: An Intelligent, End-To-End Analytics Service for Safe Deployment in Large-Scale Cloud Infrastructure. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. USENIX, Santa Clara, CA. <https://www.usenix.org/conference/nsdi20/presentation/li>
 - [27] Ze Li, Chang Lou, Vignatha Yenugutala, Vivek Ramamurthy, Eion Blanchard, Minghua Ma, and Murali Chintalapati. 2025. Orchestrating Cross-Layer Anomaly Detection and Mitigation to Address Gray Failures in Large-Scale Cloud Infrastructure. In *2025 IEEE/ACM International Workshop on Cloud Intelligence & AIOps (AIOps)*. 10–12. <https://doi.org/10.1109/AIOps66738.2025.00008>
 - [28] Chang Lou, Cong Chen, Peng Huang, Yingnong Dang, Si Qin, Xinsheng Yang, Xukun Li, Qingwei Lin, and Murali Chintalapati. 2022. RESIN: A Holistic Service for Dealing with Memory Leaks in Production Cloud Infrastructure. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*. USENIX Association, Carlsbad, CA, USA, 109–125. <https://www.usenix.org/conference/osdi22/presentation/lou-resin>
 - [29] Chang Lou, Peng Huang, and Scott Smith. 2020. Understanding, Detecting and Localizing Partial Failures in Large System Software. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. USENIX Association, Santa Clara, CA, 559–574. <https://www.usenix.org/conference/nsdi20/presentation/lou>
 - [30] Chang Lou, Yuzhuo Jing, and Peng Huang. 2022. Demystifying and Checking Silent Semantic Violations in Large Distributed Systems. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*. USENIX Association, Carlsbad, CA, USA, 91–107. <https://www.usenix.org/conference/osdi22/presentation/lou-demystifying>
 - [31] Microsoft. [n.d.]. Azure Impact reporting REST API. <https://learn.microsoft.com/en-us/rest/api/impact/>
 - [32] Pooria Namyar, Arvin Ghavidel, Daniel Crankshaw, Daniel S. Berger, Kevin Hsieh, Srikanth Kandula, Ramesh Govindan, and Behnaz Arzani. 2025. Enhancing Network Failure Mitigation with Performance-Aware Ranking. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '25)*. USENIX Association, Philadelphia, PA, 335–357. <https://www.usenix.org/conference/nsdi25/presentation/namyar>
 - [33] David Oppenheimer, Archana Ganapathi, and David A. Patterson. 2003. Why Do Internet Services Fail, and What Can Be Done About It?. In *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems (USITS)* (Seattle, WA). <http://dl.acm.org/citation.cfm?id=1251460.1251461>
 - [34] Biswaranjan Panda, Deepthi Srinivasan, Huan Ke, Karan Gupta, Vinayak Khot, and Haryadi S. Gunawi. 2019. IASO: A Fail-Slow Detection and Mitigation Framework for Distributed Storage Services. In *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference* (Renton, WA, USA) (USENIX ATC '19). USENIX Association, USA, 47–61.
 - [35] Dimas Parikesit and Chang Lou. 2025. Once Bitten, Still Shy: Can We Prevent Cloud Systems from Repeating Their Mistakes?. In *Proceedings of the 24th ACM Workshop on Hot Topics in Networks* (UMD Campus, College Park, MD, USA) (HotNets '25). Association for Computing Machinery, New York, NY, USA, 335–343. <https://doi.org/10.1145/3772356.3772385>
 - [36] Robbert van Renesse, Yaron Minsky, and Mark Hayden. 1998. A Gossip-style Failure Detection Service. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing* (The Lake District, United Kingdom) (Middleware '98). Springer-Verlag, London, UK, 55–70. <http://dl.acm.org/citation.cfm?id=1659232.1659238>
 - [37] Kaushik Veeraraghavan, Justin Meza, Scott Michelson, Sankaralingam Panneerselvam, Alex Gyori, David Chou, Sonia Margulis, Daniel Obenshain, Shruti Padmanabha, Ashish Shah, Yee Jiun Song, and Tianyin Xu. 2018. Maelstrom: Mitigating Datacenter-level Disasters by Draining Interdependent Traffic Safely and Efficiently. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*. USENIX Association, Carlsbad, CA, 373–389. <https://www.usenix.org/conference/osdi18/presentation/veeraraghavan>
 - [38] Zhe Wang, Huanwu Hu, Linghe Kong, Xinlei Kang, Qiao Xiang, Jingxuan Li, Yang Lu, Zhuo Song, Peihao Yang, Jiejian Wu, Yong Yang, Tao Ma, Zheng Liu, Xianlong Zeng, Dennis Cai, and Guihai Chen. 2024. Diagnosing Application-network Anomalies for Millions of IPs in Production Clouds. In *2024 USENIX Annual Technical Conference (USENIX ATC '24)*. USENIX Association, Santa Clara, CA, 885–899. <https://www.usenix.org/conference/atc24/presentation/wang-zhe>
 - [39] Xin Wu, Daniel Turner, Chao-Chih Chen, David A. Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. 2012. NetPilot: automating datacenter network failure mitigation. *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012), 419–430. <https://doi.org/10.1145/2377677.2377759>
 - [40] Danyang Zhuo, Monia Ghobadi, Ratul Mahajan, Klaus-Tycho Förster, Arvind Krishnamurthy, and Thomas Anderson. 2017. Understanding and Mitigating Packet Corruption in Data Center Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) (SIGCOMM '17). Association for Computing Machinery, New York, NY, USA, 362–375. <https://doi.org/10.1145/3098822.3098849>