Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

An Autonomous Decision-Making Framework For GPU Fabric Fault Remediation: Intelligent Systems For AI Training Infrastructure

Indra Kumar Mondal

Independent Researcher, USA, ORCID ID:0009-0009-0938-3427

Abstract
Large-scale AI training depends on GPU fabrics that behave less like conventional networks and more like a single distributed processor, where a fault in one link can stall an entire synchronized job. As training clusters scale into the tens of thousands of GPUs, mean time between failures falls sharply one recent study found mean-time-to-failure declining by roughly two orders of magnitude between an eight-GPU job and a 1,024-GPU job [1] and the old model of a human operator watching dashboards no longer matches the speed or volume of faults a modern fabric produces. This article proposes a tiered, bounded-autonomy framework for GPU fabric fault remediation, in which autonomous action is scoped not by fault type but by the reversibility and blast radius of the corrective action. The framework organizes remediation into four tiers: detection, diagnosis, recommended action, and autonomous action, with escalation to a human operator triggered whenever an action is large, hard to reverse, or matches a failure signature the system has not previously validated. Drawing on production evidence from large-scale GPU cluster operators and established human-automation interaction models, the article argues this reversibility-gated boundary fits GPU fabric operations better than a generic autonomy maturity ladder, since failure costs here are measured directly in GPU-hours. The discussion compares this framework against existing fault-detection systems, which already report strong diagnostic accuracy in production RoCE fabrics [4, 12], and against human-AI collaboration research in other safety-critical settings, where time-bounded human confirmation of AI recommendations has been shown to improve outcomes [2]. The aim is to guide how the next generation of self-operating AI training infrastructure expands autonomy deliberately, as trust in a defined, auditable authority boundary is earned.

Keywords - Autonomous Infrastructure Operations, GPU Fabric Management, Decision-Making Framework, AIOps, Human-in-the-Loop Systems, Intelligent Fault Remediation.

1. Introduction

The network that connects the GPUs inside a modern AI training cluster is not simply plumbing that moves data between machines. Because thousands of processors must remain in tight synchronization to complete a single training step, a fault that would be a minor nuisance in an ordinary network a flapping link, a congested path, a misbehaving optical transceiver does not slow down one machine; it stalls the entire job [3]. Every processor sits idle waiting on the slowest participant in the collective operation, and an investment measured in thousands of GPU-hours quietly erodes until someone, or something, intervenes.

This fragility has become harder to manage precisely because it scales in the wrong direction. As clusters grow from single-digit GPU counts into the tens of thousands, the rate at which faults occur increases sharply. Production data from large research clusters shows the mean time between failures dropping by roughly two orders of magnitude between an eight-GPU job and a 1,024-GPU job, and the same data projects mean-time-to-failure figures in the range of hours, not days, once job sizes reach the tens of thousands of GPUs [1]. A fault-detection-and-diagnosis workload that a human operator could once absorb comfortably at small scale becomes, at hyperscale, a volume and velocity problem that exceeds what any team of engineers can watch by hand. This is the condition under which autonomous or agentic operations fabrics that detect their own faults, localize the cause, and in well-understood cases take corrective action without waiting for a person have moved from a research curiosity to an operational necessity.

The cost of getting this wrong is not evenly distributed across failure types, which is a detail easy to overlook when failure rates are discussed only in aggregate. A short link flap that self-heals within seconds costs a training job a brief stall; a full-cluster checkpoint triggered unnecessarily, or a training run killed and restarted on a false diagnosis, costs hours of compute across every participating GPU simultaneously. Production evidence bears this out directly: even when only a small fraction of jobs are affected by a given failure category, that category can still consume a disproportionate share of total cluster runtime once the cost of recovery reloading a checkpoint, re-establishing collective communication state, re-warming caches is counted [1]. This asymmetry between how often a failure occurs and how much it costs when it does is the foundation this article's framework is built on, and it is the reason a decision architecture organized around fault type alone is an incomplete answer to the problem.

Existing work on this problem tends to fall into one of two camps. The first camp is systems-focused: diagnostic tools that detect and localize network faults in production RDMA fabrics with high precision [4], or congestion-control mechanisms that reduce flow completion time and communication delay under the specific stress patterns that collective training traffic creates [5]. These systems are effective at the detection and diagnosis stages, but they generally stop short of specifying what an autonomous system is permitted to do once a fault is localized the decision of whether to act alone or to escalate to a person is left implicit, often resolved informally by whatever operational policy a given team happens to adopt on a given day, rather than by a documented and auditable rule. The second camp is conceptual: human-automation interaction research that has, for decades, described levels of automation ranging from fully manual to fully autonomous [6], and more recent AI-governance frameworks that describe the general obligations of human oversight over automated systems [18, 16, 17]. This literature supplies the vocabulary of graduated autonomy but was not built around the specific economics of GPU fabric operations, where the cost of a wrong autonomous action is not measured in inconvenience but in irrecoverable compute-hours and, in some cases, silently corrupted model quality [7].

Much of the existing AIOps and decision-support literature treats autonomy as a binary choice or a generic maturity ladder, without addressing the specific decision boundary a GPU fabric operator actually needs. The gap this article addresses sits between these two camps: a framework that takes the detection and diagnosis capability the systems literature has already demonstrated and attaches to it a decision boundary grounded in the specific failure economics of GPU fabrics that determines when autonomous action is appropriate and when it is not. The objective is not to relitigate whether autonomy belongs in GPU fabric operations; production evidence already argues that it does, with deployed systems reporting substantial reductions in monthly network failure rates once autonomous diagnosis is paired with a defined remediation workflow [12, 14], and with recent practical frameworks for agentic AIOps demonstrating that autonomous operational agents can already be structured, deployed, and governed in enterprise IT environments more broadly [8]. The objective is to formalize where the boundary between autonomous and human-gated action should sit, using reversibility and blast radius as the governing variables rather than fault type alone.

This article proposes a four-tier decision framework detection, diagnosis, recommended action, and autonomous action in which escalation to a human operator is triggered by three conditions: the action under consideration is large or difficult to reverse, the failure signature is novel or has not been previously validated, or the action risks masking a subtler problem such as silent data corruption rather than resolving it. Each of these three conditions is developed in detail in Section 4, and each is chosen specifically because fault-type classification alone cannot capture it. The remainder of the article reviews prior work in fault detection, congestion management, and human-automation interaction; presents the framework's architecture and decision logic; and discusses how existing production evidence bears on where the autonomy boundary should be drawn in practice.

2. Related Work

Table 1. Related work summary in GPU fabric fault detection and autonomous remediation

Study Focus / Approach	Model or Method	Domain	Key Contribution	Limitation
Congestion control in lossless fabrics	DCQCN (PFC + ECN)	RDMA datacenter networks	Established active congestion management as a	Not evaluated at AI-training collective-traffic scale

			requirement for lossless Ethernet	
Service-aware RoCE diagnostics	R-Pingmesh monitoring system	Production RoCE fabrics	High-accuracy fault localization deployed at scale	Diagnosis only; no autonomy-boundary logic
AI-datacenter congestion control	FACC switch-driven adaptive control	AI training fabrics	Large flow-completion-time reductions vs. DCQCN/TIMELY/HPCC	Congestion-focused; does not address decision authority
Levels of automation taxonomy	10-point automation scale	Human-automation interaction (general)	Foundational vocabulary for graduated autonomy	Not built for GPU-fabric-specific failure economics
Silent data corruption in LLM training	Empirical incident analysis	Large-model pre-training	Documents SDC as a recurring, non-rare operational category	Descriptive; does not propose a remediation-authority framework
Agentic AIOps framework design	Practical engineering framework	Enterprise IT operations	Structures anomaly detection, classification, and autonomous resolution	Domain-general; no GPU-fabric decision boundary
Large-scale distributed training at hyperscale	MegaScale system	10,000+ GPU training clusters	Demonstrates fault-tolerant operation at extreme scale	Systems-engineering focus, not a decision-authority framework
Production RoCEv2 at hyperscale	Meta RDMA-over-Ethernet deployment	Distributed AI training infrastructure	Confirms congestion/reliability concerns as routine at scale	Deployment report; no formal escalation logic
Containerized large-model network diagnostics	SkeletonHunter	Containerized training environments	High precision/recall fault detection; reduces monthly failure rate	Detection/diagnosis only
Automated root-cause analysis	RCACopilot (LLM-based)	Cloud incident management	Strong RCA accuracy against a year of production incidents	Diagnostic accuracy evaluated alone, not decision authority

Fault detection and diagnosis in large-scale RDMA fabrics has matured considerably as training clusters have grown, and the scale at which this matters is itself a moving target: production deployments already coordinate collective training workloads across more than ten thousand GPUs in a single job, a scale at which any manual, human-paced fault response is structurally too slow to matter [9]. Early congestion-control work established that lossless Ethernet fabrics require active management of priority flow control and explicit congestion notification to avoid the packet loss that would otherwise stall synchronized collective operations [3]. This line of work has continued directly into today's AI training fabrics: production congestion-control systems built specifically for AI datacenter traffic report substantial reductions in flow completion time relative to earlier approaches, with one recent switch-driven system reporting flow completion time reductions of well over 70 percent against established congestion-control baselines under AI training traffic patterns [5]. Production RoCEv2 deployments operating at the scale of a major cloud provider's own distributed training infrastructure further confirm that these congestion and reliability concerns are not laboratory artifacts but everyday operating conditions once a fabric carries thousands of GPUs' worth of collective traffic [10]. Diagnostic systems deployed across large RoCE fabrics report similarly strong results in locating the specific problem

responsible for a given fault, with one service-aware monitoring system reporting accurate localization for the large majority of the problems it identified in production [4].

A related and practically important failure mode in this same literature is load imbalance across the parallel paths that a leaf-spine fabric provides. Equal-cost multipath routing is designed to spread traffic evenly across redundant links, but the large, sustained flows that collective training operations generate sometimes described as elephant flows can hash onto the same path repeatedly, concentrating load on one spine link while others sit comparatively idle. This kind of imbalance rarely trips a hard failure threshold on its own; instead, it manifests as a slow accumulation of queuing delay on the overloaded path, which is exactly the kind of degrading-but-not-yet-broken signal that a purely threshold-based detection system is poorly equipped to catch early. Diagnostic systems built for this environment increasingly treat this as a distinct category of problem, separate from outright link failure, precisely because the remediation is different: rerouting or rebalancing rather than failover.

This growing operational burden sits alongside a parallel challenge in GPU datacenter scheduling more broadly allocating and sequencing training and inference workloads efficiently across a shared, failure-prone fleet of GPUs is itself an active and substantial area of systems research [11], underscoring that fault remediation is only one of several operational problems competing for attention as GPU fabrics scale. Diagnostic tooling built for containerized large-model training environments has been shown, in extended production deployment, to detect thousands of network failures with high precision and recall, and to meaningfully reduce the fabric's overall monthly failure rate after the identified problems are addressed [12]. This result is worth dwelling on, because it demonstrates something beyond diagnostic accuracy in isolation: it shows that when diagnosis is paired with a defined remediation loop and sustained operational attention, the underlying failure rate of the fabric itself improves over time, not just the speed at which any given instance is resolved. That distinction between resolving individual incidents faster and reducing the rate at which incidents occur at all is a useful one to keep in view, because this article's framework is concerned primarily with the former rather than the latter, which is a hardware and topology question outside this framework's scope.

A related strand of systems work addresses not the detection or diagnosis of a fault, but the cost of recovering once a fault has already triggered a restart. Because a synchronized training job cannot simply resume from wherever it stopped, the collective communication state must be re-established consistently across every participating GPU. The naive approach of restarting from the last full checkpoint can cost several minutes of pure recovery time before useful training resumes, all while every GPU in the job sits idle. Recent work addressing this specific cost has shown that recovery time can be reduced from several minutes to a few seconds per GPU by replaying only a single minibatch of work across all participants rather than reloading a full checkpoint from storage, with close to no overhead during normal, fault-free operation [13]. This result matters to the framework proposed here for a specific reason: it changes where the reversibility variable in Section 4 draws its line. An action such as restarting a stuck collective communicator looks costly and difficult to reverse if recovery requires reloading a multi-gigabyte checkpoint and re-warming distributed state from scratch; the same action looks considerably more tractable, and plausibly eligible for narrower escalation criteria, if the underlying recovery mechanism has already reduced that cost to a few seconds of replayed work. This is a concrete illustration of a broader point this article returns to in Section 6: the specific thresholds this framework's classification logic should use are not fixed constants, but depend on the state of the underlying recovery tooling a given operator has deployed.

What this body of work has in common is a focus on the detection and diagnosis stages of the fault-handling pipeline establishing, with increasing confidence, that a fault exists and where it is located. Far less attention has gone to formalizing what happens next: whether the system should act on its diagnosis autonomously, recommend an action for a human to approve, or escalate outright. Root-cause-analysis systems built on large language models have shown that automated diagnostic reasoning can reach meaningful accuracy against real incident histories one system evaluated against a year of production incidents at a major cloud provider reported root-cause accuracy approaching 0.8, and has remained in operational use collecting diagnostic information for several years [14] suggesting that the diagnostic stage of autonomous operations is increasingly mature. But these systems are typically evaluated on diagnostic accuracy alone, not on the separate question of what authority the system should then be given to act. A system that correctly identifies the cause of a fault four times out of five has made real progress on the diagnosis problem; it has said nothing at all about whether the fifth, misdiagnosed case should have been allowed to trigger an unsupervised remediation.

A recent and closely related line of work has begun to formalize how an agentic AIOps system should be designed and governed as a practical engineering artifact, addressing questions of proactive anomaly detection, incident classification, and autonomous resolution across heterogeneous enterprise IT environments [8]. This work is valuable evidence that agentic automation of operational decision-making is maturing into an engineering discipline with its own design patterns, not merely a research aspiration but it is, by design, domain-general, and it does not specify the reversibility- and blast-radius-driven boundary this article argues is necessary for GPU fabric operations specifically, where the cost structure of a wrong autonomous decision differs sharply from that of a typical enterprise IT incident. A broader survey of the AIOps field synthesizing several years of research on failure management with large language models corroborates that this gap between general operational-AI frameworks and domain-specific decision boundaries is not unique to GPU fabrics; it recurs across the wider AIOps literature as automated systems are asked to take on increasingly consequential actions [15].

The literature on human-automation interaction offers a partial answer, though not one built for this domain specifically. Foundational work on levels of automation describes a spectrum from full human control to full machine autonomy, with intermediate levels in which a system may act but a human retains override authority [6]. This taxonomy has proven durable because it separates the question of what a system can technically do from the question of what authority it should be given, and that separation is precisely what this article's framework depends on. More recent AI-governance frameworks extend this into formal obligations: management-system standards for AI oversight describe how an organization should structure accountability for an automated system across its operational lifecycle [16], government risk-management frameworks describe structured governance spanning the mapping, measurement, and management of an automated system's risks rather than a single point-in-time approval [17], and graduated-autonomy taxonomies originally developed for a different safety-critical domain autonomous vehicles are general enough to describe any system in which decision authority shifts gradually from human to machine as capability and evidence accumulate [18]. These frameworks establish that graduated autonomy with defined escalation conditions is a recognized and workable governance pattern, applied successfully in domains as different from networking as vehicle automation and clinical software but none of them define the specific variables that should govern where the boundary sits in a GPU fabric context, because none of them were written with GPU-hours, collective communication stalls, or silent data corruption in mind.

A separate and directly relevant strand of evidence comes from research on human-AI collaboration in other safety-critical decision settings. Time-bounded human confirmation of AI-generated recommendations, rather than either full automation or full manual review, has been shown to materially improve outcomes in clinical decision-making: a recent review of human-AI collaboration research highlights a sepsis-alerting system in which clinicians who reviewed and confirmed AI-generated alerts within a defined window achieved substantially lower patient mortality than cases where the same alerts went unconfirmed within that window [2]. Clinical decision-making and GPU fabric operations are, on the surface, unrelated domains, but they share a structural feature this article treats as the relevant point of comparison: both involve decisions with serious and sometimes irreversible consequences, made under time pressure, informed by a stream of machine-generated signals that a human reviewing them in isolation, without AI assistance, could not process at the same speed or scale. Large-scale survey evidence on operational trust in AI-assisted incident response similarly suggests that practitioners are already willing to extend meaningful automation authority once certain trust conditions are met one recent survey of incident-response professionals found a strong majority reporting active AI integration into their operational workflows though the specific conditions under which that trust is extended vary by domain and have not yet been formalized for GPU fabric operations specifically [19].

Finally, the failure economics that make this boundary consequential are increasingly well documented empirically. Large-scale production data shows that failure rates rise sharply with cluster scale, that a non-trivial share of runtime is consumed by failure-related disruption even when the fraction of affected jobs is small [1], and that silent data corruption failures that do not trip an alarm but degrade training quality occurs regularly enough in production large-model training to be treated as a distinct operational category rather than a rare edge case, with major training operators reporting such events on a roughly weekly-to-biweekly cadence during extended pre-training runs [7]. Separately, recent production data from large-scale cloud operators indicates that network-related issues account for a substantial share of overall production failures, with RDMA-capable network interface cards specifically implicated in a meaningful fraction of those events [20]. This is the central argument for why blast radius and reversibility, not just fault type, must anchor the decision boundary: a system that reasons only about what kind of fault occurred, without reasoning about how costly and how

reversible the available responses are, cannot adequately account for failure modes like these, where the diagnosis may look routine right up until the point where it is not.

3. Decision Framework Architecture

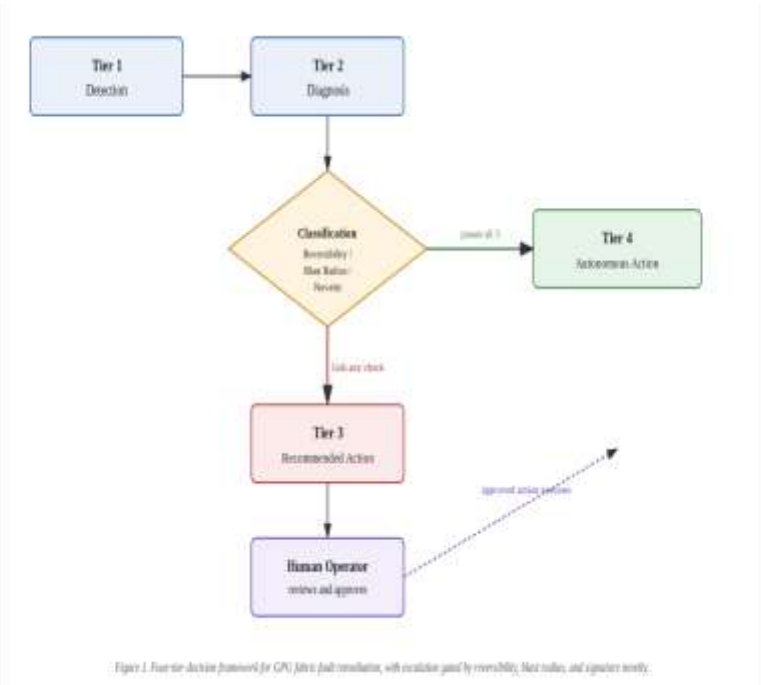

Figure 1. Four-tier decision framework for GPU fabric fault remediation, with escalation gated by reversibility, blast radius, and signature novelty.

The framework proposed here organizes GPU fabric fault remediation into four sequential tiers, each of which can either pass control to the next tier automatically or divert to a human operator depending on how the current fault and candidate response are classified.

The first tier, detection, correlates telemetry that is already well established in the production systems reviewed above RDMA link-flap events, congestion counters, collective-operation timeout logs, per-optic error rates, and the path-imbalance signals described in Section 2 into a single coherent signal rather than a storm of individually unremarkable alerts. This correlation step matters because, in isolation, none of these signals is unusual enough to demand attention; a single elevated congestion counter or a brief timeout on one collective operation happens routinely in a large fabric without indicating a genuine problem. What makes a signal worth acting on is the pattern across several of these indicators arriving together, in a way that a human watching any one dashboard would not easily notice but a system correlating all of them simultaneously can. Systems built for exactly this purpose have demonstrated that this correlation can be done accurately at production scale [4, 12], and this framework treats that capability as a solved input rather than attempting to re-derive it.

The second tier, diagnosis, localizes the fault to a specific cause: a degrading optical transceiver, a congested spine link arising from the ECMP hash-collision pattern described earlier, a stuck collective communicator, or a straggler on the compute side rather than the network. This localization step is where the practical difficulty of GPU fabric operations concentrates, because from the perspective of a stalled training job, nearly all of these causes present identically: the collective operation stalls, GPU utilization stays pinned at a high level while no forward progress occurs, and the affected job's throughput drops to zero. Automated diagnostic reasoning of this kind has already been demonstrated against real production incident histories with meaningful accuracy [14], and this framework again builds on that demonstrated capability rather than proposing a new diagnostic method. What this framework adds is not a better diagnostic technique, but a disciplined answer to the question that comes immediately after a diagnosis is reached: given this cause, and this proposed fix, who or what should be trusted to execute it.

The third and fourth tiers are where this framework's contribution concentrates. At the recommended-action tier, the system proposes a specific remediation draining and rerouting traffic around a flapping link, restarting

a stuck communicator, re-pinning a job to a healthier rail, adjusting congestion-control parameters, or triggering a checkpoint-and-migrate sequence but does not execute it without a human decision, because the classification logic described in Section 4 has determined that the action or the situation warrants a person's judgment. This tier is deliberately designed so that the human's role is a bounded decision, not an open-ended investigation: the system has already done the correlation and localization work, and what it presents to the operator is a specific proposed action with the telemetry that justifies it, not a raw alert requiring the operator to start the diagnostic process from scratch. At the autonomous-action tier, the same categories of remediation are executed without waiting for approval, because the same classification logic has determined that the action is bounded, reversible, and matches a previously validated failure signature closely enough to proceed with confidence.

The boundary between these last two tiers is the framework's central design decision, and it is deliberately not drawn along fault-type lines alone. A flapping link and a suspected case of silent data corruption can, in the early telemetry, look similar; what should differ is not necessarily the diagnosis but the authority the system is given to act once the diagnosis is made. This is worth restating because it runs against a common intuition that the right way to design an autonomy boundary is to build a list of "safe" fault types that may be handled autonomously and a list of "unsafe" fault types that always require a human. Section 4 explains why this framework rejects that structure in favor of a boundary defined by properties of the response rather than the category of the fault.

4. Methodology

Table 2. Comparison of the proposed bounded-autonomy framework against baseline operating models

Dimension	Human-Only Baseline	Fully-Autonomous, No-Escalation Baseline	Proposed Bounded-Autonomy Framework
Mean time to remediation	Slowest; bounded by human availability and manual correlation	Fastest for in-scope faults; unbounded risk for novel faults	Fast for validated, reversible, narrow-blast-radius faults; deliberately slower for escalated cases
Auditability	Informal, inconsistent across operators	Action logs exist but authority boundary itself is undocumented	Every autonomous action logged with telemetry, classification result, and outcome
Risk of masking silent corruption	Low (human judgment can catch anomalies) but slow to scale	High (novel signatures may be actioned as if routine)	Mitigated directly via the independent novelty-of-signature escalation check
Operator trust / adoption	High trust, low scalability	Low trust once a single high-cost error occurs	Trust built incrementally via auditable, expanding track record

The following classification logic is presented as a design specification, grounded in and evidenced by the production systems and literature reviewed in Section 2, rather than as an empirically validated algorithm; Section 6 returns to what would be required to validate it directly. The classification logic that governs the recommended-action / autonomous-action boundary evaluates each candidate remediation against three variables, applied in the following order: reversibility, blast radius, and novelty of the failure signature. Each variable is described below along with the specific reasoning for why fault-type classification alone cannot substitute for it.

The first variable is reversibility. An action such as draining traffic away from a single flapping link and rerouting it onto an already-provisioned alternate path is straightforwardly reversible: if the diagnosis proves wrong, the traffic can be routed back with no lasting consequence beyond a brief performance dip during the reroute itself. An action such as killing and restarting a training run that has already consumed a large number of GPU-hours is not reversible in any meaningful sense the sunk cost cannot be recovered even if the decision

is later judged correct, and even a well-optimized checkpoint-restart pipeline still costs real wall-clock time re-establishing collective communication state and resuming from the last saved point, time during which every participating GPU sits idle rather than training. Taking an entire rail offline during a critical training window falls into the same category: even if the underlying diagnosis was accurate and the rail genuinely needed to be pulled, the decision to do so during an active, high-priority training run is not one whose consequences can be undone simply by bringing the rail back online later. Only actions in the first category are eligible for the autonomous-action tier; actions in the second category are routed to the recommended-action tier regardless of how confident the diagnosis is, because confidence in a diagnosis is not the same thing as confidence that the specific response is safe to execute unattended.

The second variable is blast radius: how much of the cluster, and how much of the currently running work, is affected by the candidate action. An action scoped to a single link or a single node has a small blast radius even if it turns out to be wrong, because the affected portion of the job can typically continue with degraded but non-zero throughput while the correction takes effect, or the specific node can be excluded from the current round without stalling every other participant. An action that pauses every GPU in a job for a full-cluster checkpoint has a large blast radius by definition every GPU that would otherwise be training sits idle for the duration of the pause, regardless of whether that specific GPU had anything to do with the fault that triggered the checkpoint. This framework treats a large blast radius as sufficient on its own to require escalation, independent of the reversibility classification described above. This independence matters: a technically reversible action can still warrant a human decision if enough of the cluster is affected at once, because the cost of a large-blast-radius action is paid immediately and in full, in idle GPU-hours across the entire affected scope, even in cases where the action itself could later be undone.

The third variable is the novelty of the failure signature. Even a small, reversible, narrowly scoped action is escalated if the telemetry pattern that triggered it does not closely match a previously validated case. This condition exists specifically to guard against the failure mode documented in production silent-data-corruption research [7]: a fault whose surface signature resembles something the system has handled before, but which is actually a novel and potentially more serious problem the system has not been validated against. Silent data corruption is, almost by definition, a failure that does not announce itself through an obvious break in the collective communication pattern the way a hard link failure does; it can instead present as a subtle degradation in training loss curves, or as telemetry that looks nearly identical to a routine, previously-handled hardware fault. Treating novelty as an independent escalation trigger, rather than folding it into the reversibility and blast-radius checks alone, is what allows the framework to remain cautious in exactly the situations where confident autonomous action would be most dangerous: the cases where the system's own pattern-matching is most likely to be wrong precisely because the pattern looks so familiar.

A candidate remediation proceeds to the autonomous-action tier only if it passes all three checks: reversible, narrow blast radius, and a matched, previously validated failure signature. Failing any one of the three routes the candidate to the recommended-action tier instead, where a human operator reviews the system's proposed action before it executes. This ordering accomplishes something a single combined score could not: because each of the three checks can independently force escalation, the framework cannot be "argued around" by a fault that scores well on two of the three dimensions but poorly on the third. A large-blast-radius action does not become eligible for autonomous execution merely because it happens to be reversible and matches a familiar pattern, and a perfectly reversible, narrowly scoped action does not become eligible merely because the fault type is common, if the specific signature this time does not match prior validated cases closely enough.

4.1 A Worked Example: Classifying Three Candidate Faults

To make the classification logic concrete, consider three faults that might arrive at the recommended-action / autonomous-action boundary on the same afternoon, each superficially similar in that all three present as a stalled collective operation with elevated congestion counters on one path.

The first fault is a single RDMA link exhibiting an intermittent flap pattern the system has seen and correctly handled dozens of times before, on a job whose remaining checkpoint interval is only a few seconds old. The proposed remediation drain the flapping link and reroute onto an already-provisioned alternate path is reversible (traffic can be routed back with no lasting consequence), has a narrow blast radius (only the flows currently using that link are affected, and the job's other communication paths continue unaffected), and matches a previously validated failure signature closely (the flap pattern, timing, and affected link role are all consistent with prior confirmed instances). This fault passes all three checks and proceeds to the autonomous-action tier: the system reroutes without waiting for a human decision, and logs the action, the telemetry that justified it, and the outcome for later audit.

The second fault presents with elevated congestion counters that, on first inspection, resemble the same flap pattern but the affected link sits on a rail carrying a checkpoint write for a job that has been training for nine days without a successful checkpoint due to an unrelated storage-layer issue discovered earlier that week. Restarting the collective communicator on this link, the system's default remediation for this fault category, would require the job to fall back to its last successful checkpoint several days old rather than resuming from a recent state. Even though the fault signature itself matches a previously validated pattern, the blast radius of the available remediation is unusually large given this job's specific checkpoint history, and the framework's blast-radius check, evaluated independently of the signature-matching check, routes this case to the recommended-action tier. A human operator reviews the system's proposed alternative a slower, more conservative remediation that avoids forcing a fallback to the stale checkpoint and approves it, a decision the automated classification alone was correctly designed not to make unsupervised.

The third fault again presents with congestion counters resembling the same general pattern, but the diagnosis subsystem flags a secondary signal that does not match any previously validated instance: a small but consistent divergence in gradient norms across the GPUs downstream of the affected link, a pattern associated in the literature with early-stage silent data corruption rather than ordinary congestion [7]. Even though the proposed remediation itself would be narrowly scoped and technically reversible, the novelty check forces escalation regardless, because this is exactly the situation the third classification variable exists to catch: a fault whose superficial signature resembles something routine, but whose full telemetry picture does not match anything the system has previously confirmed as benign. The human operator's review here is not a formality; it is the specific point in the pipeline where a person with broader context can recognize that the standard remediation might resolve the symptom the congestion while leaving the underlying corruption unaddressed.

These three cases share an almost identical starting signature and diverge only once the reversibility, blast-radius, and novelty checks are applied in sequence. A framework organized around fault type alone would have handled all three identically, since all three are, at the level of "what kind of fault is this," the same category of congestion-related event.

5. Model Architecture and Trust Mechanisms

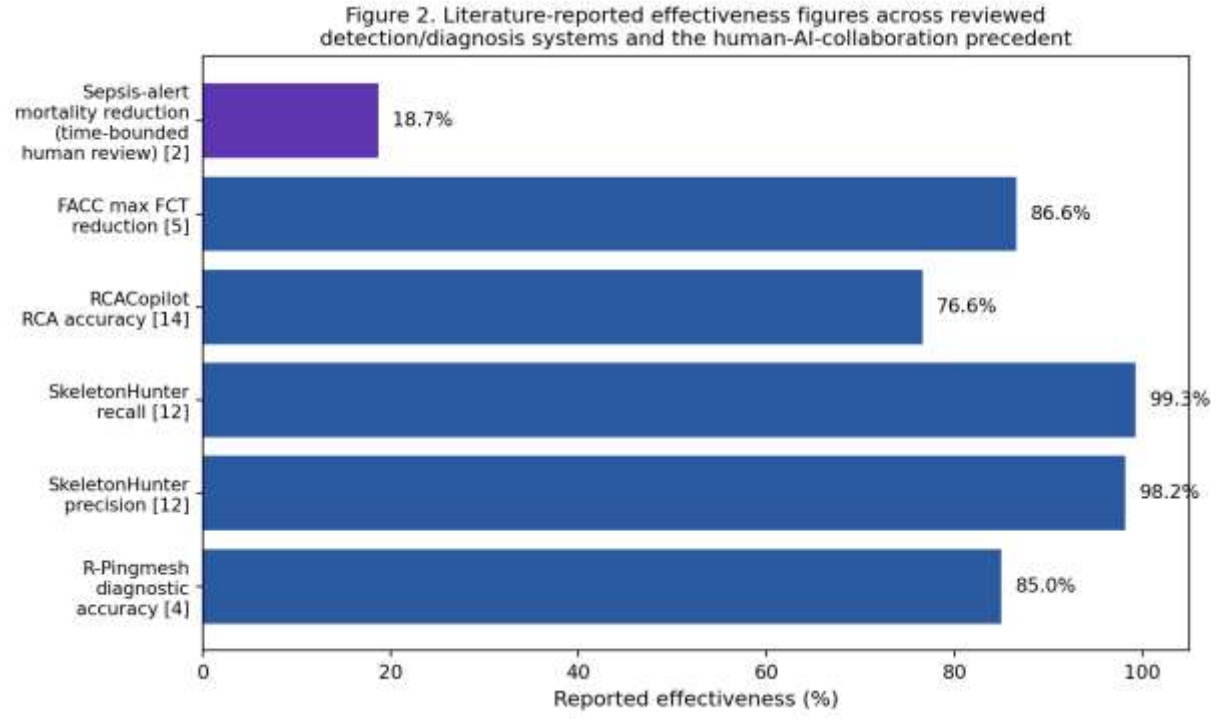

Figure 2. Literature-reported effectiveness figures across reviewed detection/diagnosis systems and the human-AI-collaboration precedent.

For an organization to extend meaningful autonomous authority to a system operating under this framework, that authority must be bounded, auditable, and reversible in the specific sense described above but it must also be trusted, and trust in an automated decision system is not a default state; it is built over time through a demonstrated track record. Production diagnostic systems that report high precision and recall figures [4, 12] earn operational trust specifically because those figures are measured against real deployments over extended periods, not claimed from a lab benchmark; one system's reported figures, for instance, are drawn from roughly six months of continuous production operation rather than a single evaluation snapshot [12]. This framework treats every autonomous action as an opportunity to extend that same evidentiary discipline: each action taken without human approval is logged with the specific telemetry that justified it, the classification result across all three variables in Section 4, and the eventual outcome, so that the system's actual track record not just its designed intent becomes the basis for whether its autonomous authority is later expanded or narrowed.

This connects directly to the broader human-automation interaction literature, which has long distinguished between systems that merely act and systems that act in a way a human can verify and, if necessary, reverse [6]. Time-bounded human confirmation of AI-generated recommendations, rather than either full delegation or full manual review, has been shown in other safety-critical domains to improve decision quality materially [2] a finding this framework treats as directly transferable to the recommended-action tier, where the system's role is to propose a specific, well-justified action within a tight enough time window that human review does not become a bottleneck, without removing the human's authority to decline it. The design intent here is narrow and specific: the recommended-action tier is not meant to function as a queue where proposed fixes wait indefinitely for attention while the underlying job continues to stall. It is meant to compress the operator's decision down to a single, well-supported judgment call, informed by exactly the telemetry the system used to reach its own recommendation, so that the human's involvement adds judgment rather than re-doing diagnostic work the system has already completed.

Governance frameworks built for AI systems more broadly reinforce the same principle from a different angle: structured oversight across the lifecycle of an automated system, rather than a one-time approval at deployment, is what current AI risk-management guidance treats as the baseline expectation for any system with meaningful decision authority [16, 17]. Applied to GPU fabric operations specifically, this means the boundary described in Section 4 is not a fixed line drawn once at design time, but a boundary that should be revisited as the system accumulates a track record narrowed if autonomous actions prove unreliable in a particular failure category, and cautiously widened if they prove consistently sound across enough instances to justify the change. This is also where the framework's approach to novel failure signatures becomes self-reinforcing over time: as previously novel patterns are resolved, reviewed, and confirmed correct by human operators at the recommended-action tier, they can be formally added to the set of validated signatures the third classification variable checks against, gradually and deliberately expanding what the autonomous-action tier is permitted to handle without ever removing the escalation trigger for signatures that remain genuinely unfamiliar.

Consider how this audit-and-track-record mechanism would play out concretely over a period of operation. Suppose an operator deploys the framework against a fabric where the flapping-link remediation described in Section 4.1 is common, and the system escalates every instance to the recommended-action tier for the first several weeks because the specific link-flap signature on this new fabric configuration has not yet been validated. Each time a human operator approves the proposed reroute and the outcome is logged as correct, that specific signature accumulates evidence toward validation. Once enough instances have been reviewed and confirmed a threshold this article deliberately does not fix in advance, since it is properly a function of the operator's own risk tolerance and the fabric's specific failure history the signature can be added to the validated set, and future instances of that same fault on that same fabric configuration become eligible for the autonomous-action tier without further human review. The reversibility and blast-radius checks continue to apply independently at every subsequent instance, so a signature's validated status narrows only the novelty check, never bypasses the other two. This is what makes the framework's expansion of autonomy an evidentiary process rather than a scheduled one: two fabrics running the same underlying hardware could reasonably end up with different autonomous-action scopes if one has simply accumulated more confirmed instances of a given fault than the other.

6. Results and Discussion

The evidence reviewed across Sections 2 through 5 converges on a specific conclusion about where the autonomous/human-gated boundary should sit in GPU fabric operations, and it is worth stating plainly what that evidence does and does not establish.

It establishes, first, that the detection and diagnosis stages of this pipeline are no longer the bottleneck they once were. Production systems already correlate distributed telemetry into coherent fault signals with high accuracy [4, 12], and automated diagnostic reasoning already performs creditably against real incident histories, with root-cause accuracy figures approaching 0.8 reported against a full year of production incident data at scale [14]. A framework proposing to re-litigate detection or diagnosis methodology would be solving an already reasonably well-addressed problem; the contribution this article makes is deliberately scoped to the decision layer that sits above those stages, precisely because that layer has received comparatively little formal attention in the systems literature reviewed in Section 2.

It establishes, second, that the cost of getting the autonomy boundary wrong is not symmetric, and this asymmetry is what justifies weighting reversibility and blast radius as heavily as this framework does. Production data on failure rates at scale shows that a small fraction of affected jobs can still consume a disproportionate share of total runtime once recovery costs are counted [1], and separate production evidence on silent data corruption shows that the most dangerous failures are not always the ones that are hardest to detect, but sometimes the ones whose surface signature resembles something benign [7]. A framework that escalates based on fault type alone would have no principled way to catch this category of failure, because by construction the fault type looks familiar; escalating based on the novelty of the specific signature, independent of how familiar the general category appears, is what closes this gap. This is not a hypothetical concern raised for the sake of argument: the production evidence on RDMA-related failure rates at scale [20] indicates that network-related causes already account for a substantial share of total production incidents, which means the population of faults this framework must classify correctly is not a rare edge case but a routine and recurring part of GPU fabric operations.

It establishes, third more tentatively, since the supporting evidence here is drawn from adjacent domains rather than from GPU fabric operations directly that the recommended-action tier is not merely a slower, more cautious version of full autonomy, but a mechanism that can itself improve decision quality when human review is time-bounded and well-scoped rather than open-ended. The clinical decision-support evidence reviewed in Section 5 [2] is suggestive rather than conclusive on this point when transplanted to infrastructure operations, and validating it directly in a GPU fabric context is an open empirical question this article does not resolve. What the clinical evidence does establish clearly, and what this article treats as transferable in principle, is that the value of human review depends heavily on how that review is structured an unbounded review queue that competes for operator attention against dozens of other alerts behaves very differently from a narrow, time-boxed confirmation step attached to a single, well-justified recommendation, and the framework proposed here is deliberately designed to produce the latter rather than the former.

Taken together, this evidence supports a framework in which autonomous authority expands along the reversibility and blast-radius axes as production experience accumulates, rather than along a generic maturity timeline measured in months since deployment. A team operating a new, unproven fabric configuration should expect most remediations to sit at the recommended-action tier regardless of how routine the underlying fault appears, simply because the novelty check in Section 4 will correctly flag an unfamiliar environment's telemetry patterns as unvalidated. A team with an extended, well-audited track record on a specific, narrow class of reversible actions, by contrast, has an evidentiary basis not just an aspirational one for allowing those specific actions to run autonomously, because the logging discipline described in Section 5 gives them a defensible record of exactly how often those specific actions have been correct.

The framework's limitations follow directly from the state of the evidence it draws on, and it would be a disservice to the reader to understate them. The three-variable classification logic in Section 4 is argued from first principles about failure economics and supported by the production systems and human-automation literature reviewed above, but it has not yet been validated end-to-end as a single deployed system; the individual components detection, diagnosis, and human-AI collaborative decision-making are each separately evidenced in the literature reviewed here, but their combination as a bounded-autonomy pipeline specifically for GPU fabrics remains a design proposal rather than a reported production deployment. A related limitation concerns the threshold-setting problem this article does not attempt to solve: this framework specifies which three variables should govern the autonomy boundary and the order in which they should be checked, but it does not specify the exact numerical thresholds how many GPU-hours qualify as a "large" blast radius, or how many prior validated instances are required before a signature is no longer considered novel because those

thresholds are properly an empirical question that depends on a specific operator's risk tolerance, fabric scale, and accumulated track record, rather than a constant this article could responsibly assert in the abstract. Establishing those thresholds through a real deployment, and validating the framework's classification logic against the specific failure categories that recur most often in production, is the natural next step for the applied research this article positions itself within.

7. Conclusion

GPU training fabrics have reached a scale at which manual, human-driven fault remediation is no longer adequate to the volume and velocity of faults a modern cluster produces, and the systems literature has already demonstrated that automated detection and diagnosis can meet this challenge with production-grade accuracy. What has been less developed is the decision layer that determines what an autonomous system is permitted to do once a fault is diagnosed. This article has proposed a four-tier framework detection, diagnosis, recommended action, and autonomous action in which the boundary between the last two tiers is governed by reversibility, blast radius, and the novelty of the failure signature, rather than by fault type alone. This reversibility-gated approach is argued to be better suited to GPU fabric economics, where failure costs are measured in unrecoverable GPU-hours and where the most dangerous failures are often the ones that most resemble familiar, benign faults. Extending autonomous authority under this framework is not a one-time design decision but an ongoing process, governed by an auditable track record that expands or narrows the boundary as evidence accumulates from actual operation rather than from an initial design assumption. Realizing this framework as a deployed, end-to-end system, and validating its specific classification thresholds against production data, is the natural direction for future work building on the detection, diagnosis, and human-AI collaboration research this article has drawn together. As AI training clusters continue to grow, and as the gap between fault velocity and human reaction time continues to widen, the question this article has tried to answer not whether to automate, but precisely where the line of autonomous authority should sit is likely to become one of the more consequential design decisions in how the next generation of AI factories is operated.

Ethics Statement

Conflict of Interest: The author declares no conflict of interest.

Statement of Human and Animal Rights: This study did not involve human participants, animal subjects, or identifiable personal data requiring ethics board review.

Informed Consent: Not applicable.

CRediT Author Contribution Statement

Indra Kumar Mondal: Conceptualization, Methodology, Investigation, Writing – Original Draft, Writing – Review & Editing.

Acknowledgments

The author used Claude (Anthropic, claude-sonnet model) to assist with manuscript drafting and language refinement, consistent with JIDMIS's Guidelines on the Use of Generative AI and AI-Assisted Tools. The author takes full responsibility for the accuracy, originality, and integrity of all content presented in this manuscript, and confirms that all data, analysis, and conclusions reflect independent work and verified sources.

Biographical Sketch

Indra Kumar Mondal is a Principal Solutions Architect with over 19 years of experience in enterprise AI infrastructure, cloud networking, and data center transformation. He currently designs large-scale GPU networking fabrics and AI data center architectures at Netris Inc., leveraging NVIDIA Spectrum-X, SONiC, EVPN/VXLAN, and RoCEv2 across multi-vendor ecosystems. His prior roles include Co-Founder and Chief Technology Architect of an AI-driven infrastructure automation startup, and Network Solutions Architect roles designing SONiC- and Cumulus Linux-based data center networking for global enterprise and cloud deployments. He holds a CCIE certification (#48659), is NVIDIA AI Networking certified, and is a certified SONiC Engineer.

References

[1] Kokolis, A., Kuchnik, M., Hoffman, J., Kumar, A., Malani, P., Ma, F., DeVito, Z., Sengupta, S., Saladi, K., & Wu, C.-J. Revisiting Reliability in Large-Scale Machine Learning Research Clusters. 2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2025. DOI: 10.1109/HPCA61900.2025.00096

[2] Xu, G., Murthy, S. V., & Jia, B. Enhancing Intuitive Decision-Making and Reliance Through Human-AI Collaboration: A Review. Informatics (MDPI), 12(4), Article 135, 2025. DOI: 10.3390/informatics12040135
[3] Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M. H., & Zhang, M. Congestion Control for Large-Scale RDMA Deployments. Proceedings of ACM SIGCOMM 2015, 2015. DOI: 10.1145/2785956.2787484
[4] Liu, K., Jiang, Z., Zhang, J., et al. R-Pingmesh: A Service-Aware RoCE Network Monitoring and Diagnostic System. Proceedings of ACM SIGCOMM 2024, 2024. DOI: 10.1145/3651890.3672264
[5] Zuo, Y., et al. FACC: Switch-Driven Fast-Adaptive Congestion Control for RDMA in AI Datacenters. Journal of Cloud Computing (Springer), 2025. DOI: 10.1186/s13677-025-00830-0
[6] Parasuraman, R., Sheridan, T. B., & Wickens, C. D. A Model for Types and Levels of Human Interaction with Automation. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 30, pp. 286-297, 2000. DOI: 10.1109/3468.844354
[7] Ma, J., Pei, H., Lausen, L., & Karypis, G. Understanding Silent Data Corruption in LLM Training. Proceedings of ACL 2025, pp. 20372-20394, 2025.
[8] Zota, R. D., Barbulescu, C., & Constantinescu, R. A Practical Approach to Defining a Framework for Developing an Agentic AIOps System. Electronics (MDPI), 14(9), Article 1775, 2025. DOI: 10.3390/electronics14091775
[9] Jiang, Z., Lin, H., Zhong, Y., et al. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI '24), 2024.
[10] Meta Engineering et al. RDMA over Ethernet for Distributed Training at Meta Scale. Proceedings of ACM SIGCOMM 2024, 2024. DOI: 10.1145/3651890.3672233
[11] Ye, Z., Gao, W., Hu, Q., Sun, P., Wang, X., Luo, Y., Zhang, T., & Wen, Y. Deep Learning Workload Scheduling in GPU Datacenters: A Survey. ACM Computing Surveys, Vol. 56, Issue 6, Article 143, 2024. DOI: 10.1145/3638757
[12] Liu, W., Qian, K., Li, Z., et al. SkeletonHunter: Diagnosing and Localizing Network Failures in Containerized Large Model Training. ACM SIGCOMM 2025, 2025. DOI: 10.1145/3718958.3750513
[13] Gupta, T., Krishnan, S., Kumar, R., Vijeev, A., Gulavani, B. S., Kwatra, N., Ramjee, R., & Sivathanu, M. Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures. EuroSys '24, pp. 1110-1125, 2024. DOI: 10.1145/3627703.3650085
[14] Chen, Y., Xie, H., Ma, M., et al. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. ACM EuroSys '24, 2024.
[15] Zhang, L., Jia, T., et al. A Survey of AIOps in the Era of Large Language Models. ACM Computing Surveys, 2025.
[16] International Organization for Standardization. ISO/IEC 42001:2023 - Information technology - Artificial intelligence - Management system. 2023.
[17] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, 2023. DOI: 10.6028/NIST.AI.100-1
[18] SAE International. J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. 2021 revision.
[19] Falowo, O. I., & Bou Abdo, J. Empirical Study on Automation, AI Trust, and Framework Readiness in Cybersecurity Incident Response. Algorithms (MDPI), 19(1), 62, 2026. DOI: 10.3390/a19010062
[20] Lin, S., Zhou, K., Zhang, H., et al. SHIFT: Exploring the Boundary of RDMA Network Fault Tolerance. arXiv:2512.11094, 2025/2026.