FlowPulse: Catching Network Failures in ML Clusters

Jakob Krebs
Technion
Haifa, Israel

Dimitry Gavrilenko
Technion
Haifa, Israel

Daniel Amir
Technion
Haifa, Israel

Shir Landau Feibish
University of Haifa
Haifa, Israel

Mark Silberstein
Technion
Haifa, Israel

ABSTRACT

Network hardware faults are inevitable in massive scale-out ML training clusters. Networks in such systems are inherently designed for resiliency, routing around faulty components *as long as a fault is detected*. Unfortunately, some *silent* faults evade detection. Notably, the effects of silent faults are amplified in modern production networks that deploy per-packet load balancing, because packets of a single flow traverse many network paths, making such faults particularly hard to localize.

We present FlowPulse, the first system for rapid, low-overhead detection of silent network faults in per-packet spraying networks. Our key insight is that distributed training workloads induce predictable traffic patterns in the switch ports we refer to as a *temporal symmetry*. This symmetry emerges even in the presence of *known faults*, and can be modeled analytically or learned from the traffic. FlowPulse detects *new* network faults of training tasks by identifying subtle deviations from the expected temporal symmetry on each switch during collective communications, all without any inter-switch coordination or probing overheads. Our preliminary results show that FlowPulse is effective in detecting silent faults in a variety of realistic settings, topologies and fault patterns. For example, FlowPulse identifies a single faulty link with 1.5% corruption rate by checking temporal symmetry in a full two-level fat tree topology with 32 leaf switches while performing Ring-AllReduce on all nodes.

CCS CONCEPTS

• **Networks → Error detection and error correction**; **Network performance modeling**.

HotNets '25, November 17–18, 2025, College Park, MD, USA

ACM ISBN 979-8-4007-2280-6/2025/11
https://doi.org/10.1145/3772356.3772384

KEYWORDS

Network Debugging, Measurement and Telemetry, Networks for Machine Learning

ACM Reference Format:
Jakob Krebs, Dimitry Gavrilenko, Daniel Amir, Shir Landau Feibish, and Mark Silberstein. 2025. FlowPulse: Catching Network Failures in ML Clusters. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3772356.3772384

1 INTRODUCTION

Backend networks dedicated to inter-GPU communications have reached unprecedented scales [14, 43], supporting large-scale distributed tasks on hundreds of thousands of nodes. The sheer number of components in a system of such scale makes networking hardware faults the norm. Faults include link flaps, bit corruption errors, NIC and switch malfunctions. All result in unexpected packet drops and the consequent degradation of network performance [14, 32]. Unfortunately, the bulk-synchronous nature of training workloads [14, 15, 27] makes them highly susceptible to network faults, as individual flows affected by faults dictate the performance of the entire application [20, 45].

At the same time, traditional data center networks using Equal Cost Multi Path (ECMP) flow-level load balancing [19, 40] perform poorly for distributed training due to low flow entropy and flow collisions [14]. This issue brought renewed interest in *adaptive per-packet spraying (APS)* load balancing strategies where switches forward or *spray* packets across all available upstream paths toward the destination leaf [8]. A spraying algorithm may forward to random ports [12] or employ more sophisticated *adaptive* strategies, such as selecting the least congested port [16].

APS in non-blocking Clos topologies has near-optimal performance with low latency under high demand [4, 16, 25, 29]. It has long been the design choice for Infiniband networks [42], and recently for the Cornelis Omni-Path Express networking solution [30]. Nowadays, APS is increasingly deployed in Ethernet backend networks by NVIDIA [9, 34], Cisco [22], and Broadcom [24].

When it comes to network faults, however, APS is a double-edged sword: even one faulty link affects many network flows. For example, consider a full 2-level non-blocking fat tree with switches of radix 64. A single noisy spine-to-leaf link (0.05% of all links) affects *all* sizable flows forwarded to that leaf, corrupting about 3% of packets in each flow which dramatically increases their completion times.

It is thus critical to quickly *detect, localize, and disable* faulty components by excluding them from routing. Faulty links are routinely removed by the switch OS [13] when a port signals the fault, or when it is reflected in switch counters [31]. Unfortunately, some faults are *silent*. Thus, they cause sudden performance degradation without any clear sign in switch telemetry counters. Moreover, the counters themselves might be incorrect because of a hardware fault. For example, switch memory corruption of the FIB [21, 46] can create a transient routing black hole where all packets are silently dropped.

Existing techniques to detect silent faults are not suitable for large-scale APS networks. Each fault affects many flows, and each flow traverses many paths. Debugging a slowdown requires to trace packets across all paths to figure out which specific path had the fault. End-to-end probes, as in Pingmesh [18], indeed check all paths in the network. However, hardware faults often appear only under high load (common during ML communication), and executing a large number of probes in a loaded network imposes prohibitively large overheads. Alternatively, there are proposals to aggregate switch counters at a central location to detect inconsistencies across switches to cope with unreliable counters [35, 46]. Centralized aggregation, however, comes with significant communication and synchronization overheads proportional to the frequency of updates and network size, thus it is slow to react and does not scale.

APS networks have a unique property that brings faults to the surface. Without network faults, these networks exhibit *spatial symmetry*: non-leaf switches should have nearly equal load. Hence, unequal load among a leaf switch's downstream links from spines is a sign of a fault. Unfortunately, real large-scale networks always have some disabled links: links can fail at any time and are only replaced at specific maintenance windows [41]. As a result, *pre-existing* faulty links break spatial symmetry, diminishing the utility of this strategy in detecting new faults.

We present **FlowPulse**, a novel system for rapid, low-overhead detection of silent faults in APS networks. Our key insight is that when running distributed ML training, APS networks exhibit another form of symmetry: per-port *temporal symmetry*. ML training has repetitive communication pattern, e.g., running an *identical* AllReduce collective at each training iteration. Assume the collective runs exclusively in a non-blocking lossless network. Then, the total amount of packets traversing via downstream leaf switch ports from the spines in each AllReduce instance should be nearly the same, as long as no new faults occur in the network (§4). Thus, we use temporal symmetry to detect new faults even in a network with preexisting faults: each new fault changes the flow distribution through the links. Importantly, temporal symmetry is not sensitive to jitter among participating nodes for linear ring topologies. Moreover, it can be *checked* within switches, without coordination, or centralized aggregation.

We make the following contributions:

Per-Link Load Model: FlowPulse builds a per-link load model that estimates the expected amount of data received via each spine-to-leaf downstream port during AllReduce or similar reduction collective executed on a given set of nodes. This model takes into account both the known network faults obtained from each switch's routing table and the observed traffic matrix from initial training iterations. We explore multiple prediction methods, including analytical, simulation-based and learning.

A System for Continuous Monitoring of Silent Faults: FlowPulse continuously monitors the network for silent link faults using the per-link load model defined above. Each leaf switch collects the amount of data received by a leaf from the spines during the specifically marked and prioritized collective, and detects deviations from the model predictions independently of other switches.

Simulation and Evaluation: We evaluate FlowPulse in a 2-level non-blocking fat-tree topology with 16 spine and 32 leaf switches, running a 31-stage Ring-AllReduce in a lossless Ethernet network. FlowPulse reliably detects both full and partial link faults, even when a fault affects as little as 1% of the link's traffic. We evaluate a range of conditions, including varying switch radixes, initial fault counts, and collective sizes, and show that FlowPulse achieves precise, instantaneous detection without injecting additional traffic or relying on centralized telemetry collection.

2 BACKGROUND

Backend ML training networks. We focus on *scale-out backend* networks connecting $10k - 100k$ GPUs [14], optimized for distributed ML training tasks spanning a large portion of the nodes [3].

There are many approaches to build backend networks [14, 32, 48]. In this paper we focus on the Ethernet architecture used in large-scale deployments in some of the largest ML training clusters by NVIDIA [37], and increasingly adopted by other hardware vendors [22, 24]. It shares many traits with other Ethernet deployments, but differs in its use of APS as we summarize below.

- **Topology:** Non-blocking two- or three-level fat tree. Non-blocking topology [32, 34, 48] is often used for flexible allocations of compute resources while guaranteeing nearly identical network performance in any allocation.
- **Link layer:** Load balancing via Adaptive Per-packet Spraying (*APS*) over upstream topology links, selecting the least loaded switch port. Downstream paths are not sprayed. Switches use lossless queues with the link-layer Priority Flow Control (PFC) [8].
- **Transport:** RoCE with out-of-order writes. Some deployments disable congestion control [14, 48], relying on the congestion-aware collectives and a link-layer PFC.
- **Workloads:** Each NIC is associated with a single GPU running a single task, communicating predominantly via collective operations, such as AllReduce. Collectives are co-optimized with hardware, network topology and congestion control [11].

Scale-out in ML training. Data-parallel execution is commonly used to scale training tasks to tens of thousands of nodes. It requires aggregation of gradients across all replicas in each iteration, and thus exhibits highly repetitive traffic patterns. Other types of parallelism split the execution of individual replicas across multiple nodes, but each such node also participates in its own data-parallel collective.

Reduction collectives, such as AllReduce and ReduceScatter, are the main communication primitives of data-parallel training implemented in libraries such as NCCL [11]. They are optimized for different topologies and scales. In particular, they are often implemented as a pipeline over a virtual ring, thus achieving optimal communication bandwidth.

3 RELATED WORK

Existing fault detection systems use path probing [18, 23, 39, 44, 47], rely on the control-plane for information [1, 2, 33], or centralized data aggregation [28, 38, 46]. While these methods are effective for detecting certain classes of faults, they struggle to detect *silent* faults.

Path probing techniques can detect silently failed links and black holes [18, 39], yet they introduce additional load on the network. Faults typically impact traffic during peak usage periods, when there is no bandwidth available for probe traffic [32]. Additionally, elevated bit error rates are more likely to affect large flows, making such faults difficult to detect using small probe packets [44].

Control-plane-based systems focus on verifying configuration correctness [1, 2, 33], but cannot detect hardware or software faults in transceivers or switches, which are often sporadic and not reflected in configuration state. Silent faults are challenging, as they typically do not manifest in switch counters [46]. Some systems address this by comparing behavior across switches or sampling counters synchronously to detect inconsistencies [38]. As such faults impact application performance, some approaches leverage end-host metrics to detect and localize performance-impacting issues [35]. This method identifies network asymmetries and treats them as a fault indication. However, in large networks faults are always present. This persistent background noise makes it difficult to distinguish meaningful signals. As a result, this approach does not scale well to such environments.

FlowPulse overcomes these limitations by using switches to monitor patterns in the existing traffic without additional probes. By comparing the observed traffic distribution with the expected patterns of ML training workloads, FlowPulse can detect subtle performance anomalies. We leverage temporal symmetry as described next.

4 TEMPORAL SYMMETRY

ML training workloads are highly repetitive. Specifically, in large-scale parallel training with data parallelism, a reduction collective such as AllReduce must be run in each training iteration to compute and distribute the gradients. This AllReduce can span thousands of nodes, and can be tens to hundreds of megabytes [14], or even gigabytes per layer in the case of recent large language models [17]. During each iteration of AllReduce, the set of communicating nodes, as well as the amount of data transferred between each pair, is the same.

When combined with APS load balancing, and under the *same network faults*, this periodic workload creates *temporal symmetry* in the amount of data traversing each link. Namely, APS ensures that, in an empty network, the collective will have its packets distributed across all valid paths, resulting in nearly the same per-switch-port load during every training iteration. Note that temporal symmetry applies individually to each link over time, and is thus different from spatial symmetry. Links from two different spines to the same leaf node may see different traffic levels due to pre-existing faults in the network. Still, as long as the set of faults does not change, and the workload is *precisely* the same, each individual link is traversed by the same volume of traffic during each instance of the collective in every iteration.

Because temporal symmetry relies on an identical workload during each iteration, care must be taken to address jitter. Prior to each collective, some nodes may experience longer computation times, resulting in straggler nodes that begin the collective after other nodes. Different nodes may become stragglers during different iterations, creating slight temporal variations in the workload.

First, we note that the metric of the total volume of traffic traversing each port over the course of each collective iteration is more resilient to jitter than, e.g., maximum throughput, so the symmetry is defined in terms of the data volume.

Second, we observe that when each leaf switch hosts a single source and/or destination node, jitter does not affect the traffic distribution across spines because the spraying is performed in the leaf. Thus, consistent measurements can be performed in the presence of jitter if this condition is met. This single-source/single-destination per leaf condition holds for Ring-AllReduce collectives. However we also discuss how to support more general communication patterns.

5 DESIGN

FlowPulse leverages temporal symmetry to achieve in-switch, coordination-free detection of network faults. Specifically, it verifies temporal symmetry at the leaf switches, on the ingress ports from spines. These links are chosen because they are late in the path, meaning that they will reflect faults along almost the entire path. Further, in a 2-level fat tree these link can only be reached via a single path from each sender, meaning that they can be used to help localize faults.

As an ML training task begins, FlowPulse predicts the data volume that should traverse each of these ports during each iteration of the reduction collective. Deviation from this prediction beyond a detection threshold indicates that a fault has occurred in the network. Leaf switches autonomously detect the communication topology used for the collective, detect the beginning and end of each iteration of the collective, and compare the flow observed at each port to the prediction to detect faults, as discussed in detail below. This is accomplished using programmable switches, which have become prevalent in training clusters [7].

There are three major components to FlowPulse's design: measuring the traffic during each iteration of the data-parallel all-reduce, predicting the amount of traffic traversing each leaf ingress port over the course of the collective, and detecting faults based on deviations from the prediction.

5.1 Measuring a collective

To measure the traffic volume traversing each port during a collective, network switches must be able to detect when the collective is running, and determine which packets belong to the collective. This can be accomplished through minor modifications to the communications library, for example the NVIDIA Common Collectives Library (NCCL) [11]. We propose to tag the packets of the AllReduce collective with a `flow_id` that combines a sentinel value with the iteration number. This provides switches with precise information about which traffic to measure without any additional communication overhead or messaging via the control plane [6]. It also allows switches to detect when the first iteration of a new training task has started, allowing baseline measurements to be taken as part of the load prediction computations.

Handling background traffic, stragglers and jitter. FlowPulse only measures a single collective per iteration. To ensure that the packet spraying logic is not influenced by background network traffic, we prioritize the target flows in the network, by assigning higher traffic priority to the packets of the collective used for the measurements. This prioritization isolates the collective while maintaining the original load experienced during training by switch hardware units. This is necessary, as background flows impose additional, unaccounted, load on the switch and naturally alter the packet spraying pattern of the previous instance of the collective.

The temporal symmetry assumption holds when aggregating traffic volumes over a collective even when the senders are not perfectly synchronized, e.g., in the case of stragglers. FlowPulse is *oblivious* to stragglers. It considers a collective as finished at the start of the next iteration. All communications of the prior training iteration must be completed at this point by construction of synchronous data-parallel training.

Another challenge emerges in an asymmetric network when senders at the same leaf switch target multiple *non-local* destinations, i.e., destinations outside of that switch. If there is jitter in the start of the collective by each sender, and it is inconsistent across iterations, the switch uplink queues may experience different occupancy across asymmetric network paths, As a result, the load distribution across these ports will differ between iterations, breaking FlowPulse's core assumptions.

FlowPulse only measures the behavior of a single non-local flow leaving the leaf switch to cope with this problem. In practice, the Ring-AllReduce collective which we use as the basis for our measurements often naturally has this single non-local destination per leaf per collective property. This is because collectives are optimized for network topology to ensure local communications within the leaf if possible; local communications in the nodes under the same leaf are not forwarded to the spine; and in a ring, only one node outside the leaf serves as a source and another node as a destination.

We believe, this approach can be extended beyond Ring-AllReduce. For example, we may select a subset of flows from the collective representing each leaf switch once as a sender, and once as a receiver. These flows are run at a high priority and are the only flows used for verifying temporal symmetry.

5.2 Load prediction

We consider three methods for load prediction: an analytical model, network simulations, and load observation.

Analytical prediction. The analytical model uses application level knowledge about the collective size and implementation to calculate the per-port load in an optimally load-balanced network. The application knows which nodes will communicate over the course of the collective, as well as

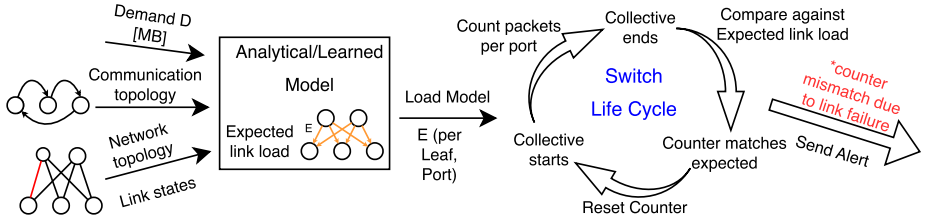


Figure 1: FlowPulse overview: we use the network connectivity and the collective's demand to predict per-link data volume in the downstream leaf-spine link at each leaf(§5.2). Leaf switches record the actual data volume and compare with the predicted values (§5.3). Deviation beyond a threshold is considered a fault.

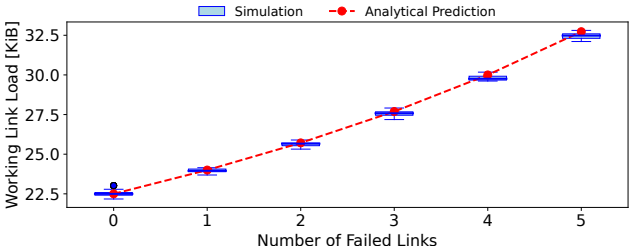


Figure 2: Analytical prediction matches the simulation for a single flow.

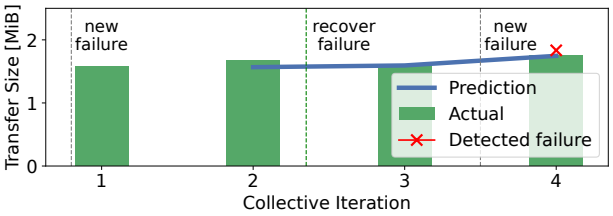


Figure 3: Learning-based prediction model update. FlowPulse learns an improved baseline after transient fault recovery.

how much data each pair will send. This anticipated demand is shared with the network switches, or measured directly in the first iterations. The switches compute the predictions in the control plane.

In a fault-free network, the traffic sent by each source-destination pair will be evenly load-balanced across all spine switches. However, if a preexisting fault affects the link between the source and a given spine, or between the destination and a given spine, then that spine will not be used by that source-destination pair. Instead, the traffic will be balanced across the remaining spines. If a given source-destination pair is expected to send d bytes, f spines have failed links to either the source or destination, and there are s total spines, then each remaining spine is traversed by $d/(s-f)$ bytes. This data also traverses the links between these spines and the leaf switch corresponding to the destination node. Adding up the contributions from each source-destination pair whose destination corresponds to a given leaf switch is all that is needed to predict the load on each of the leaf switch's ingress ports from spines.

Fig. 2 shows close agreement between the load predicted by our analytical model and the actual load observed in a network simulated in ns-3 [5]. The setup is described in §6.

Simulation-based model. To achieve higher prediction fidelity, the expected per-port load can be taken from a simulation of the network. This allows FlowPulse to exactly incorporate knowledge about known faults (including gray faults), the exact load-balancing algorithms used, and other implementation details about the network into the prediction. While a simulation yields the highest fidelity, significant time and computation resources must be spent running the simulation before every training job.

Learning. It is also possible to learn the expected load on each port by simply measuring the load during the first iterations of the collective. One caveat is that a transient fault may exist during the first iterations, but disappear thereafter. When a fault heals, the load observed on all ports re-balances more evenly. When FlowPulse observes this behavior, it replaces the baseline measurement with a new measurement reflecting the improved network state. Fig. 3 shows how this correction occurs when the expected load after a fault is recovered.

5.3 Identifying faults

Detection. Every leaf switch counts the data volume received at each ingress port from spines during each collective iteration. At the end of each iteration (i.e., at the beginning of a new one following our detection technique), the switch compares the observations against the model prediction. If the discrepancy exceeds a predefined threshold, the switch declares a fault and alert the network operator.

FlowPulse uses a detection threshold of 1%. We observe (§6) that this threshold is high enough to avoid false positives due to variations in packet spraying, but low enough to still detect intermittent faults.

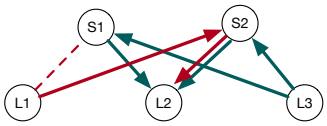


Figure 4: Faults can be localized by comparing data from two sending leaves. When traffic from a sender is received on one link, but not the other, the receiving switch infers a failure on the remote link to the sender.

Localization. Once a fault is detected, FlowPulse attempts to localize it. Reduced traffic at a given ingress port can indicate either a fault on the local link between that port and the corresponding spine switch, or a fault on a remote link between a different leaf switch and the spine switch. To distinguish these cases, FlowPulse compares the traffic volumes received from different senders over the given port. If traffic from all senders is equally affected, the local link is marked as failed. However, if only one sender is affected, the link between the spine switch and the leaf switch of the sender is marked as failed.

For example, in Fig. 4, the link between `L1` and `S1` is failed, and `L2` detects reduced traffic in the ingress port from `S1`. Since it still receives the expected amount of traffic from `L3` through this port, it can deduce that the failed link is the remote link between `L1` and `S1`, rather than the local link between `L2` and `S1`.

6 EVALUATION

We evaluate FlowPulse's accuracy in several network and workload conditions using NS-3 [5] simulations. We vary collective sizes, fault probability, switch radix, and numbers of pre-existing faults.

Experimental setup. Unless stated otherwise, we use a non-blocking 2-level fat tree with 32 leaf and 16 spine switches. We implement a simple transport tolerant to reordering, mimicking the current RoCE NICs [6], without congestion control. The network is lossless, but packet losses due to injected faults are detected via a retransmission timeout of 5μsec. To inject new faults, we configure a single leaf-spine link to drop packets at a set rate. The links with pre-existing faults are disconnected. We report false positive/negative rates when using the analytical model for predictions §5.2.

We run a single Ring-AllReduce collective, where each leaf is connected to a single end-host that serves as both a receiver and a sender. The traffic is non-blocking.

Setting the classification threshold. Fig. 5(a) shows the Residual Operating Curve (RoC) of the classifier for different fault detection thresholds and various packet drop rates on the faulty link. We observe that FlowPulse achieves perfect accuracy with a detection threshold of 1% for $\geq$ 1.5% packets dropped per link. For lower drop rates the classifier becomes less effective. We note that the threshold is set empirically in a given network when calibrating the system, but intend providing an analytical way to configure it in the future.

Varying switch radix. Networks with higher switch radix balance the traffic across more paths, so the fault impact on each flow is amortized across all of them. As a result, detecting faults becomes more challenging as the deviations from the expected values are smaller. For example, Fig. 5(b) shows that FlowPulse cannot detect the fault with the drop rate of 0.8% for radix 32, but works well for radix 16. However, it also means that the application performance in higher-radix networks is impacted less by such faults, so detecting them becomes less critical.

Varying collective size. Fig. 5(c) shows the effect of the size of the collective on FlowPulse accuracy. Larger collectives send more packets, resulting in higher Signal-to-Noise ratio when measuring the per-port load and better accuracy. Fortunately, a typical AllReduce collectives in large LLMs reach GBs in size [17], well beyond the amount needed for FlowPulse to achieve high accuracy.

Effect of pre-existing faults. FlowPulse detects new faults even when known faults already exist. As the model takes these faults into account, we observe perfect classification for new faults that drop $\geq$ 2.5% of packets or more.

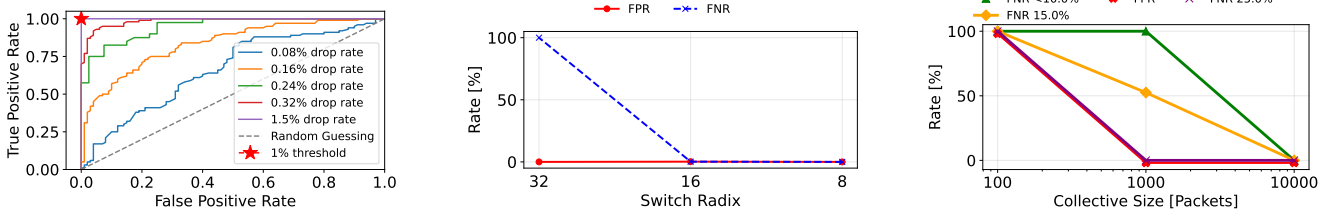
7 DISCUSSION AND FUTURE WORK

FlowPulse makes assumptions about network and workload. While some are inherent to the design, others may be alleviated to generalize the solution.

Inherent Limitations. FlowPulse leverages temporal symmetry, which arises from the combination of APS and regular collectives found in ML training. It cannot generalize to contexts which do not exhibit this symmetry. In particular, it is a poor fit for classical datacenter networks, where the traffic matrix is unpredictable and packet spraying is rarely used.

Network Topology. We currently focus on non-blocking two-level Clos topologies. FlowPulse could extend to other topologies by deploying FlowPulse at both leaf and spine levels to monitor spine-leaf and core-spine links respectively. We leave this to be explored in future work.

Blocking Networks. While the topology may be non-blocking, permanent faults can cause congestion. FlowPulse mitigates this by prioritizing the measured collective, as a single data-parallel AllReduce under-utilizes the network [14, 36]. FlowPulse supports blocking networks if the prioritized, measured workload does not experience queuing, but future work will consider congestion control as well.



(a) Residual Operating Curve (ROC) for different packet drop rates on a faulty link. A 1% threshold is a perfect classifier for drop rates ≥ 1.5%.

(b) FPR/FNR for different switch radixes with drop rate 0.8% per link. Higher radixes are more challenging. Lower is better.

(c) FPR/FNR for different collective sizes with different faulty link drop rates (% in the legend). Smaller collectives are more noisy. Lower is better.

Figure 5: FlowPulse accuracy analysis. FPR/FNR – False Positive/Negative Rate

Parallel Jobs. Clusters are rarely utilized by a single job. Instead, multiple independent jobs are scheduled simultaneously, sharing the network [26]. Parallel jobs may use different collective sizes and communicate between different nodes, complicating the predication of the traffic matrix. However, limiting the traffic accounting to a single collective and prioritizing it in the network for performance isolation, allows FlowPulse to be used in multi-job clusters.

Parallel Links. Networks often use parallel links between switches to increase bandwidth [10]. A single failed parallel link reduces bandwidth, but remaining links can still reach the same set of hosts. Thus, FlowPulse treats these links as independent, effectively splitting the spine into virtual switches if traffic is evenly balanced. Alternatively, the model could be extended to support per-link weights.

Fault Types. While we evaluated only transient link faults and packet loss, we believe that FlowPulse can detect most gray faults. These faults often manifest as drops, which is exactly the metric FlowPulse detects. Corrupted packets are dropped in the switches if the bit error cannot be corrected; black holes affect only specific paths. Faults that are too short or that impact less than 1.5% of packets traversing a given path are still undetectable with FlowPulse. Abnormal but correctable bit errors are also undetectable by FlowPulse, as they do not cause packet drops, yet they typically do appear in switch error counters.

Stragglers and Jitter. Our current handling of inconsistent jitter between senders in a collective (§5.1) forces us to *only* consider one non-local sender per a leaf switch. While in practice it is not a significant limitation for locality-optimized ring-based collectives, we seek to alleviate it in the future. In fact, in our initial experiments we observed that jitter did not have measurable effect on the expected load balance across egress ports because of the large number of available paths. More investigation is necessary to estimate the effect of the jitter on the precision of our fault detector.

Beyond reduction collectives. FlowPulse relies on the knowledge of the demand matrix to compute the expected switch traffic volume. This requirement is easily satisfiable for the AllReduce collectives run as part of data parallel execution. In this case, the demand matrix is the same across all iterations, thus allowing computation of the expected traffic volume in advance. We plan to extend FlowPulse to support other collectives such as AlltoAll with a dynamic demand matrix, to be able to monitor failures in more complex communication patterns used in other types of parallelism such as expert parallelism. This requires extracting the demand matrix, recomputing the expected load for the new demand, and updating the switches with the new set of thresholds. Doing this in a scalable and resource-efficient way is our ongoing work.

8 CONCLUSIONS

FlowPulse passively detects network fault in ML training clusters. It leverages temporal symmetry in packet-sprayed networks periodically running distributed collectives. Our preliminary results show that a packet drop rate of 1% in a single link suffice to detect silent faults with a high accuracy.

ACKNOWLEDGMENTS

We thank our shepherd Costin Raiciu and the reviewers for their helpful comments and feedback. This work was supported by the Israel Science Foundation (grants 1998/22 and 980/21), and by the Zuckerman STEM Leadership Program.

REFERENCES

[1] Anubhavnidhi Abhashkumar, Aaron Gember-Jacobson, and Aditya Akella. Tiramisu: Fast multilayer network verification. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 201–219, 2020.

[2] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. A general approach to network configuration verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 155–168, 2017.

[3] Maciej Besta, Jens Domke, Marcel Schneider, Marek Konieczny, Salvatore Di Girolamo, Timo Schneider, Ankit Singla, and Torsten Hoefler. High-performance routing with multipathing and path diversity in ethernet and HPC networks. *IEEE Transactions on Parallel and Distributed Systems*, 32(4):943–959, 2020.

[4] Jiaxin Cao, Rui Xia, Pengkun Yang, Chuanxiong Guo, Guohan Lu, Lihua Yuan, Yixin Zheng, Haitao Wu, Yongqiang Xiong, and Dave Maltz. Per-packet load-balanced, low-latency routing for clos-based data center networks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 49–60, 2013.

[5] NS3 Contributors and Maintainers. NS-3 Network Simulator. https://www.nsnam.org/, 2025.

[6] NVIDIA Corporation. MLNX_OFED Features Verbs and Capabilities. https://docs.nvidia.com/networking/display/rdmacore50/mlnx_ofed+features+verbs+and+capabilities, 2023.

[7] NVIDIA Corporation. NVIDIA Spectrum SN5000 Series Switches. https://nvdam.widen.net/s/mmvbnpk8qk/networking-ethernet-switches-sn5000-datasheet-us, 2024.

[8] NVIDIA Corporation. NVIDIA Spectrum-X Network Platform Architecture. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/nvidia-spectrum-x, 2024.

[9] NVIDIA Corporation. NVIDIA Supercharges Ethernet Networking for Generative AI. https://nvidianews.nvidia.com/news/nvidia-supercharges-ethernet-networking-for-generative-ai, 2024.

[10] NVIDIA Corporation. NVIDIA DGX SuperPOD: Next Generation Scalable Infrastructure for AI Leadership Reference Architecture Featuring NVDIA DGX H100. https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-h100/latest/network-fabrics.html, 2025.

[11] NVIDIA Corporation. NVIDIA NCCL Source Code. https://github.com/NVIDIA/nccl, 2025.

[12] Advait Dixit, Pawan Prakash, Y Charlie Hu, and Ramana Rao Kompella. On the impact of packet spraying in data center networks. In *2013 Proceedings IEEE INFOCOM*, pages 2130–2138. IEEE, 2013.

[13] Linux Foundation. Software for Open Networking in the Cloud (SONiC). https://sonicfoundation.dev/, 2023.

[14] Adithya Gangidi, Rui Miao, Shengbao Zheng, Sai Jayesh Bondu, Guilherme Goes, Hany Morsy, Rohit Puri, Mohammad Riftadi, Ashmitha Jeevaraj Shetty, Jingyi Yang, et al. Rdma over ethernet for distributed training at meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 57–70, 2024.

[15] Alexandru M Gherghescu, Vlad-Andrei Bădoiu, Alexandru Agache, Mihai-Valentin Dumitru, Iuliu Vasilescu, Radu Mantu, and Costin Raiciu. I've Got 99 Problems But FLOPS Ain't One. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*, pages 195–204, 2024.

[16] Soudeh Ghorbani, Zibin Yang, P Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. Drill: Micro load balancing for low-latency data center networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 225–238, 2017.

[17] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng

Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, 2024.

[18] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 139–152, 2015.

[19] Keqiang He, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella. Presto: Edge-based load balancing for fast datacenter networks. *ACM SIGCOMM Computer Communication Review*, 45(4):465–478, 2015.

[20] Qinghao Hu, Zhisheng Ye, Zerui Wang, Guoteng Wang, Meng Zhang, Qiaoling Chen, Peng Sun, Dahua Lin, Xiaolin Wang, Yingwei Luo, et al. Characterization of large language model development in the datacenter. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 709–729, 2024.

[21] Peng Huang, Chuanxiong Guo, Lidong Zhou, Jacob R Lorch, Yingnong Dang, Murali Chintalapati, and Randolph Yao. Gray failure: The achilles' heel of cloud-scale systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 150–155, 2017.

[22] Cisco Systems Inc. Cisco Nexus 9000 Series NX-OS Unicast Routing Configuration Guide, Configure Dynamic Load Balancing. https://www.cisco.com/c/en/us/td/docs/dcn/nx-os/nexus9000/105x/unicast-routing-configuration/cisco-nexus-9000-series-nx-os-unicast-routing-configuration-guide/m-configure-dynamic-load-balancing.html, 2025.

[23] Chenhao Jia, Tian Pan, Zizheng Bian, Xingchen Lin, Enge Song, Cheng Xu, Tao Huang, and Yunjie Liu. Rapid detection and localization of gray failures in data centers via in-band network telemetry. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020.

[24] Mohan Kalkunte, Niranjan Vaidya, and Pete Del Vecchio. Cognitive routing in the Tomahawk 5 data center switch. https://www.broadcom.com/blog/cognitive-routing-in-the-tomahawk-5-data-center-switch, 2023.

[25] John Kim, William J. Dally, and Dennis Abts. Adaptive routing in high-radix clos network. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, page 92–es, New York, NY, USA, 2006. Association for Computing Machinery.

[26] Apostolos Kokolis, Michael Kuchnik, John Hoffman, Adithya Kumar, Parth Malani, Faye Ma, Zachary DeVito, Shubho Sengupta, Kalyan Saladi, and Carole-Jean Wu. Revisiting Reliability in Large-Scale Machine Learning Research Clusters. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1259–1274. IEEE, 2025.

[27] Wenxue Li, Xiangzhou Liu, Yuxuan Li, Yilun Jin, Han Tian, Zhizhen Zhong, Guyue Liu, Ying Zhang, and Kai Chen. Understanding communication characteristics of distributed training. In *Proceedings of the 8th Asia-Pacific Workshop on Networking*, pages 1–8, 2024.

[28] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. LossRadar: Fast detection of lost packets in data center networks. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 481–495, 2016.

[29] Michael Mitzenmacher. The power of two choices in randomized load balancing. *IEEE transactions on parallel and distributed systems*, 12(10):1094–1104, 2002.

[30] Cornelis Networks. Cornelis Omni-Path Express Edge Switches. https://www.cornelisnetworks.com/product/cornelis-omni-path-express-edge-switches, 2025.

[31] Ming Prince. Troubleshoot Switch Port and Interface Problems. https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/12027-53.html, 2023.

[32] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, et al. Alibaba hpn: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 691–706, 2024.

[33] Sundararajan Renganathan, Benny Rubin, Hyojoon Kim, Pier Luigi Ventre, Carmelo Cascone, Daniele Moro, Charles Chan, Nick McKeown, and Nate Foster. Hydra: Effective Runtime Network Verification. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 182–194, 2023.

[34] Peter Rizk. Turbocharging Generative AI Workloads with NVIDIA Spectrum-X Networking Platform. https:

//developer.nvidia.com/blog/turbocharging-ai-workloads-with-nvidia-spectrum-x-networking-platform/, 2023.

[35] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C. Snoeren. Passive Realtime Datacenter Fault Detection and Localization. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 595–612, Boston, MA, March 2017. USENIX Association.

[36] Peter Sanders, Jochen Speck, and Jesper Larsson Träff. Two-tree algorithms for full bandwidth broadcast, reduction and scan. *Parallel Computing*, 35(12):581–594, 2009.

[37] Alex Shapiro. NVIDIA Ethernet Networking Accelerates World's Largest AI Supercomputer, Built by xAI. https://nvidianews.nvidia.com/news/spectrum-x-ethernet-networking-xai-colossus, 2024.

[38] Xuemei Shi and Surendra Anubolu. The Challenges and Practices of Network Stability in Alibabas Large Scale Computing Clusters. https://www.youtube.com/watch?v=-3qZL_DOWAc, 2024. OCP Global Summit 2024.

[39] Olivier Tilmans, Tobias Bühler, Ingmar Poese, Stefano Vissicchio, and Laurent Vanbever. Stroboscope: Declarative network monitoring on a budget. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 467–482, 2018.

[40] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall. Let it flow: Resilient asymmetric load balancing with flowlet switching. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 407–420, 2017.

[41] Saranyan A. Vigraham and Benjamin Leonhardi. Maintaining large-scale AI capacity at Meta. https://engineering.fb.com/2024/06/12/production-engineering/maintaining-large-scale-ai-capacity-meta/, 2024.

[42] Koushnir Vladimir. Recommended Topologies for Implementing an HPC Cluster with NVIDIA Quantum InfiniBand Solutions - Part 2 - Adaptive routing, HBF and SHIELD. https://enterprise-support.nvidia.com/s/article/Recommended-Topologies-for-Implementing-an-HPC-Cluster-with-NVIDIA-Quantum-InfiniBand-Solutions-Part-2, 2024.

[43] Weiyang Wang, Manya Ghobadi, Kayvon Shakeri, Ying Zhang, and Naader Hasani. How to build low-cost networks for large language models (without sacrificing performance)? *arXiv e-prints*, pages arXiv–2307, 2023.

[44] Yifan Xiong, Yuting Jiang, Ziyue Yang, Lei Qu, Guoshuai Zhao, Shuguang Liu, Dong Zhong, Boris Pinzur, Jie Zhang, Yang Wang, et al. {SuperBench}: Improving Cloud {AI} Infrastructure Reliability with Proactive Validation. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 835–850, 2024.

[45] Zhiyi Yao, Pengbo Hu, Congcong Miao, Xuya Jia, Zuning Liang, Yuedong Xu, Chunzhi He, Hao Lu, Mingzhuo Chen, Xiang Li, et al. Holmes: Localizing Irregularities in {LLM} Training with Mega-scale {GPU} Clusters. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 523–540, 2025.

[46] Zhu Yibo, Kang Nanxi, Cao Jiaxin, et al. Packet-level telemetry in large datacenter networks. In *Proc of the 2015 ACM Conference on Special Interest Group on Data Communication. New York: ACM Press*, pages 479–491, 2015.

[47] Kuichao Zhang, Wei Su, Huiling Shi, Kai Zhang, and Wei Zhang. GrayINT–Detection and Localization of Gray Failures via Hybrid In-band Network Telemetry. In *2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 405–408. IEEE, 2023.

[48] Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, et al. Insights into DeepSeek-V3: Scaling Challenges and Reflections on Hardware for AI Architectures. *arXiv preprint arXiv:2505.09343*, 2025.