PERSEUS: A Fail-Slow Detection Framework for Cloud Storage Systems

21st USENIX Conference on File and Storage Technologies

Ruiming Lu; Erci Xu; Yiming Zhang; Fengyi Zhu; Zhaosheng Zhu; Mengtian Wang; Zongpeng Zhu; Guangtao Xue; Jiwu Shu; Minglu Li and Jiesheng Wu

Paper Notes

By JeongHa Lee

Abstract

The newly-emerging “fail-slow” failures plague both software and hardware where the victim components are still functioning yet with degraded performance. To address this problem, this paper presents PERSEUS, a practical fail-slow detection framework for storage devices. PERSEUS leverages a light regression-based model to fast pinpoint and analyze fail-slow failures at the granularity of drives. Within a 10- month close monitoring on 248K drives, PERSEUS managed to find 304 fail-slow cases. Isolating them can reduce the (node-level) 99.99th tail latency by 48%. We assemble a large-scale fail-slow dataset (including 41K normal drives and 315 verified fail-slow drives) from our production traces, based on which we provide root cause analysis on fail-slow drives covering a variety of ill-implemented scheduling, hardware defects, and environmental factors. We have released the dataset to the public for fail-slow study.

Problem Statement and Research Objectives

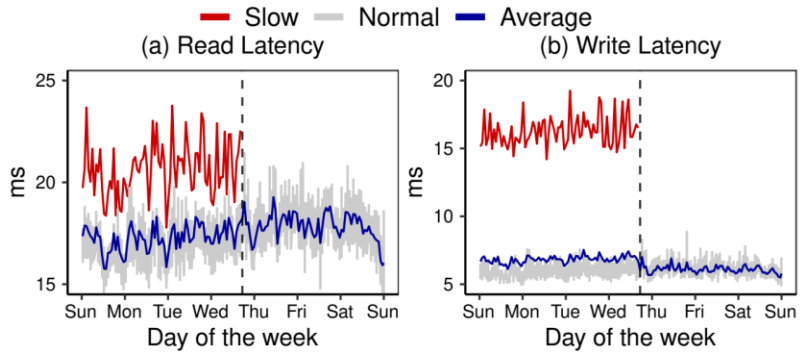


Figure 1: Fail-slow impact on I/O latency (§2.3). *The figures show (a) read and (b) write latency three days before and after isolating the fail-slow HDD (in red) in one node. Lines in grey refer to the latency distribution of normal peers from the same node. The vertical dashed line refers to the time when the fail-slow drive was taken down for replacement.*

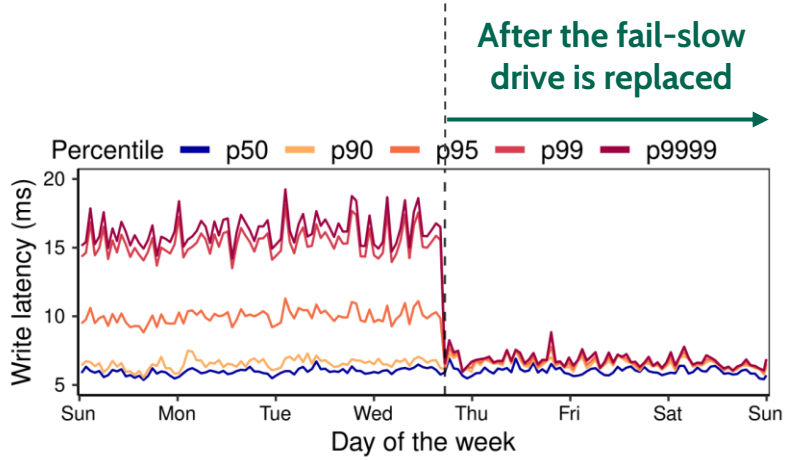


Figure 2: Fail-slow impact on tail latency (§2.3). *The figure shows the node-level write latency (of the same node as in Figure 1) at different percentiles. The vertical dashed line refers to the isolation time.*

- Previous works are impractical and inefficient for large-scale deployment in production cloud environments.
  - First, these techniques **require source code access** (e.g., static analysis in OmegaGen) **or software modification** (e.g., modifying software timeouts in IASO), while major cloud vendors typically do not modify tenants' code.
  - Second, existing techniques **can only detect fail-slow failures at the node level** (e.g., IASO), thus still requiring nontrivial manual efforts to locate the culprits.
- This paper presents a practical, fine-grained, and general fail-slow detection framework that is applicable to a wide range of services and devices (with minor or no adjustment) in Alibaba Cloud data centers.

Proposed Method

Design and implement PERSEUS, a non-intrusive fail-slow detection framework.

1. Outlier detection: Using DBSCAN ➔ Adding PCA ➔ Usage of outliers
2. Building regression model
3. Identifying fail-slow events: Distinguishing slow entries ➔ Formulating events
4. Evaluating risk

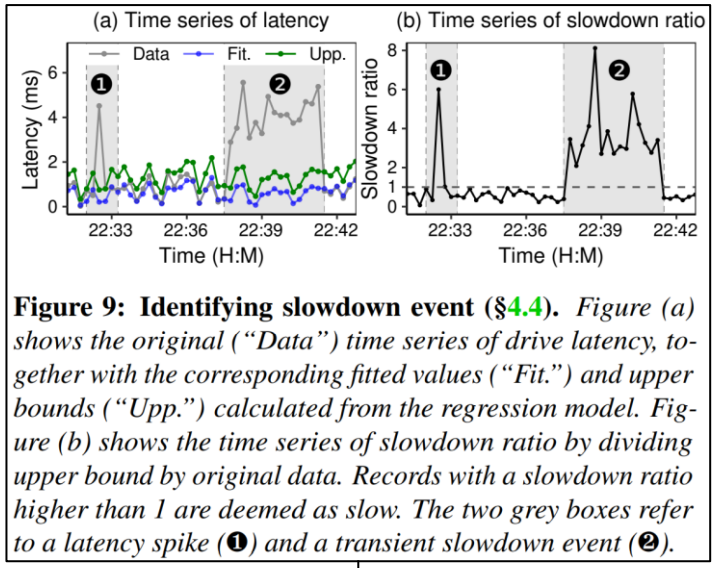
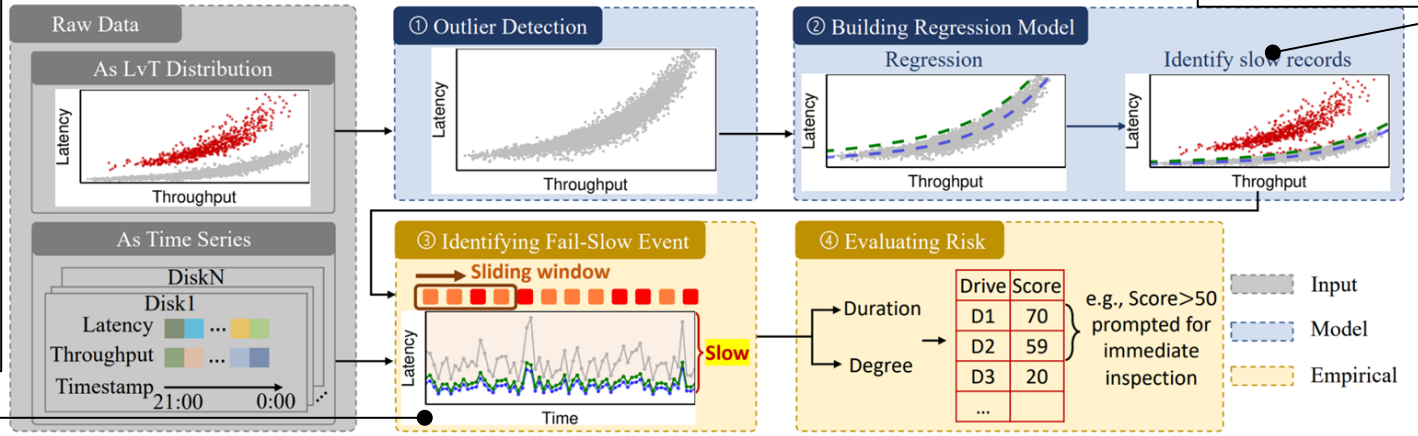


Figure 9: Identifying slowdown event (§4.4). *Figure (a) shows the original ("Data") time series of drive latency, together with the corresponding fitted values ("Fit.") and upper bounds ("Upp.") calculated from the regression model. Figure (b) shows the time series of slowdown ratio by dividing upper bound by original data. Records with a slowdown ratio higher than 1 are deemed as slow. The two grey boxes refer to a latency spike (❶) and a transient slowdown event (❷).*



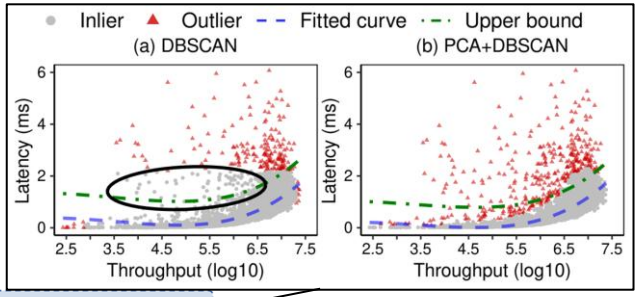


Figure 7: PERSEUS design diagram (§4.1). *From the raw data, PERSEUS seeks to distinguish the slow (in red) from the normal (in grey) by building a regression model (②) with preliminary outlier detection (①). With a sliding window, PERSEUS formulates consecutive slow records into slowdown events (③), and assigns risk scores based on slowdown duration and degree (④).*

Evaluation and Results

- PERSEUS has been deployed in a production cloud environment for over ten months, monitoring an increasing number of drives, reaching approximately 300K.
    - By isolating and/or replacing the identified fail-slow drives, the node-level tail latency is significantly reduced. The 95th, 99th, and 99.99th write latencies drop by 31%, 46%, and 48%, respectively.

1. Compared to previous fail-slow detection methods
    - A large-scale fail-slow dataset (including 315 verified fail-slow drives and around 41K of their cluster-wise peer drives) is assembled from production traces, and a test benchmark is built based on this dataset.
2. The effectiveness of PERSEUS components and the sensitivity of its parameters are evaluated.

Evaluation and Results

- **Threshold Filtering**: Drives are classified as fail-slow if their median latency during the three-hour monitoring period exceeds the upper bound.
  - Statistical Bound
    : An Xth percentile where X ranges from 75 to 99 or an interquartile range ($\mathrm{IQR} = \mathrm{3rd\_quartile} - \mathrm{1st\_quartile}$).
  - Empirical Bound
    : A latency upper bound is manually set based on the Service Level Objectives (SLOs) or suggestions from on-site engineers.
- **Peer Evaluation**: Identifies a fail-slow drive if its latency is at least X times that of the node median for a minimum duration (denoted as $\mathrm{min\_dur}$).
- **IASO-Based Model**: IASO is re-implemented strictly following its original design, with modifications applied only when necessary.

Metric	Thresh-Stat	Thresh-Emp	Peer Eval	IASO-Based	PERSEUS-Deployed
Full-set					
Precision	1.00	1.00	0.98	0.48	0.99
Recall	0.52	0.02	0.57	0.24	1.00
MCC	0.72	0.14	0.74	0.32	0.99
Subset (excluding software-induced)					
Precision	1.00	1.00	1.00	0.45	0.94
Recall	0.71	0.09	0.65	0.61	1.00
MCC	0.84	0.30	0.80	0.52	0.97

Table 5: Overall evaluation results (§5.4). *The table shows the best evaluation scores of threshold filtering based on statistical (**ThreshStat**) and empirical (**ThreshEmp**) bounds, peer evaluation (**PeerEval**), and the* IASO*-based model. For* PERSEUS*, we list the results of the deployed version. Each method is evaluated on both the full-set and the subset.*

Evaluation and Results

Parameter	Range	Description
S1: Outlier detection (§4.2)		
PCA	On/Off	Transform the coordinates w.r.t. the principal components.
DBSCAN	On/Off	Density-based outlier detection.
S3: Identifying fail-slow event (§4.4)		
X	95∼99.9	Use the *X*% prediciton upper bound as the latency upper bound.
S4: Evaluating risk (§4.5)		
min_score	1∼100	Risk score threshold.
N	1∼15	Evaluate the risk score of the most recent *N* days.

Table 6: Evaluating PERSEUS's design choices (§5.5). *The table summarizes the parameter settings of evaluating* PERSEUS*'s design choices. PCA and DBSCAN are switched on and off to demonstrate their effectiveness.*

Metric	w/o Outlier	w/o PCA	p95	p99	p999	Deployed
Full-set						
Precision	0.98	0.55	0.99	1.00	1.00	0.99
Recall	0.51	0.43	0.99	0.93	0.93	1.00
MCC	0.71	0.49	0.99	0.96	0.96	0.99
Subset (excluding software-induced)						
Precision	0.95	0.36	0.94	0.98	1.00	0.94
Recall	0.82	0.91	0.95	0.92	0.95	1.00
MCC	0.88	0.57	0.95	0.95	0.98	0.97

Table 7: Evaluation results of PERSEUS's design choices in Table 6 (§5.5). *Only results based on the best sets of parameters with the highest MCC scores are shown.*

* MCC (Matthews Correlation Coefficient, 매튜 상관계수)

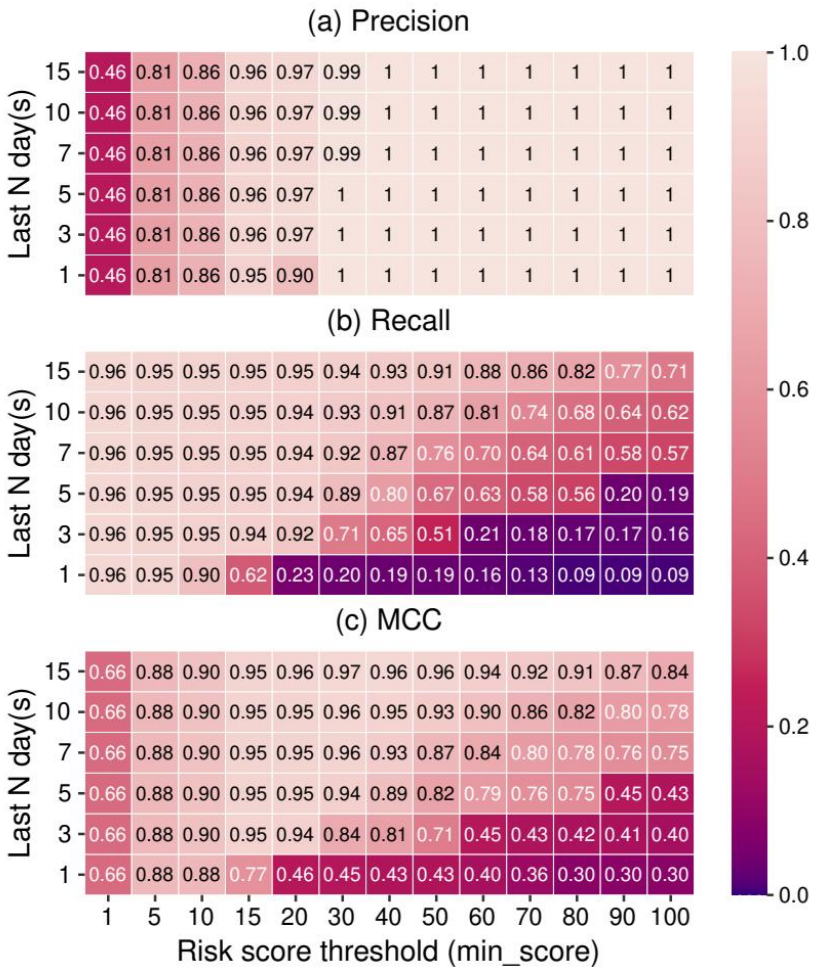


Figure 10: *min_score* and *N* of PERSEUS (§5.5). *The heatmaps show evaluation scores of* PERSEUS*-p999 in Table 7 under different min_score and N. The lighter the color, the higher the score, the better the result.*

Notes

1 : https://en.wikipedia.org/wiki/Fail-fast_system

- In systems design, a **fail-fast** system is one which immediately reports at its interface any condition that is likely to indicate a failure.[1]
- The **fail-slow** failures (also known as gray failures or limpware) : a software or hardware component (while functioning) delivers lower-than-expected performance.
  - Fail-slow failures, especially the transient ones, can often be ignored or misinterpreted as performance variations.
  - Additionally, storage stacks usually have multiple levels (software, firmware, and hardware) of fault tolerance, such as retry and redundancy, silently masking the fail-slow failures.
  - However, fail-slow failures can have a much more significant impact on I/O performance in the wild.
- Fail-stop, Fail-partial, and Byzantine