

Final Defense

Virtual Machine Proactive Fault Tolerance using Log-based Anomaly Detection

Pratheek Senevirathne
19001622

Supervisor: Dr. Dinuni Fernando, Senior Lecturer (UCSC)

Co-supervisor: Dr. Dinal Herath, Security Data Scientist (Obsidian Security, USA)

Motivation

- To provide seamless experience for Cloud hosted applications, running in VMs, we need **99.999% uptime**
- In reality, there is always a probability that **servers fail**
- Given that server is about to fail, there is a technique to move VMs - **VM migration**
- How do you know **when to migrate?**
 - Decision should be made **early** for successful VM migration

Objective: Real-time identification of failure indicators **ahead** in time to avoid the failure using VM migration

Introduction

- VM Fault tolerance - technique used to mitigate VM failures
 - Proactive Migration
 - Reactive: VM Recovery
- VM Live migration - Migrates the VM while it is running! - with little downtime
- VM migration is one of the main fault tolerance approaches
 - Google performs over 1 Million VM migrations per month!



Introduction


- Anomaly Detection - Identify unusual patterns that deviate significantly from the normal behaviour
- Anomaly Detection fits perfectly in this scenario,
 - Failures and **failure indicators are anomalies**
 - Failure data are very rare compared to normal state data - data is highly imbalanced
- Why logs?
 - Live
 - Contains rich information about the system status and events, including faults
 - Lack of research on VM fault tolerance using log data

Existing Work

- Server Failure Prediction
 - Most work [1-3, 9] utilizes server resource usage data and uses supervised models
 - MING [9] by Microsoft research, uses resource usage and logging rate, supervised
 - Models: CNN, LSTM, Random Forest, Feed Forward NN, Bayesian models, etc.
- Virtual Machine Failure Prediction
 - Most work [4-8] utilizes resource usage data
 - Work by Nam, et al. [6, 7] utilizes logs from VM and Host
 - NLP approach - using BERT tokenizer and CNN a model for classification
- All above work uses supervised models - **Not ideal**
 - Need a lot of labeled data - requires effort
 - Might not work on unforeseen failures

	BERT - CNN [7]	Our work: VMFT-LAD
Training approach	Supervised	Semi-supervised
Training mode	Offline	Online
Training corpus size	Large	Very small
Handling of data imbalance	Over/Under Sampling	No sampling required
Human intervention	High	Little to no
Early detection	✓	✓
Real-time	?	✓
Adapt to HW/SW changes	✗	✓
Handle Unforeseen failures	✗	✓
Feedback	✗	✓

Research Questions

1. How to effectively utilize VM and server logs for Machine Learning based VM failure prediction?
 2. How to develop a generalized VM failure prediction approach using log analysis, enabling its applicability to a wide range of generic VMs?
 3. How can the timing of VM failure prediction be optimized to ensure a successful migration, considering the total time required for the VM migration?
- 

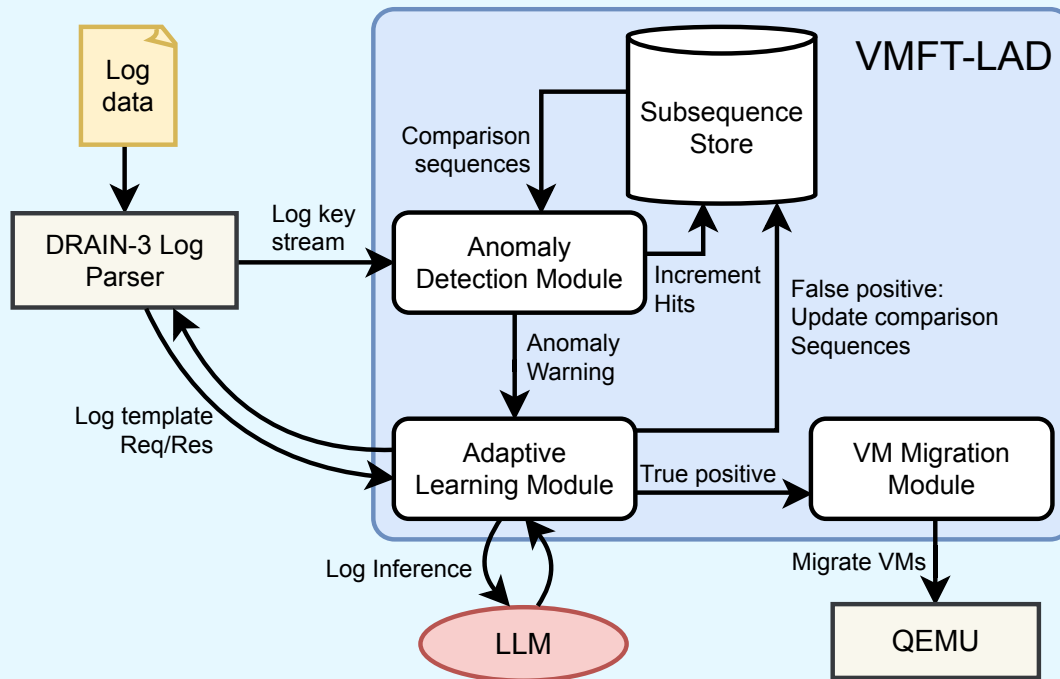
In scope

- Data collection by simulating server and VM failures using fault injection
- Development of VM failure prediction system using ML
- Utilization of logs from server and hypervisor
- Predict failures real-time by continuously monitoring logs

Out of scope

- Simulation of long-term and cascading VM failures
- Integrity of VMs before failure - Assume that faults do not corrupt the VM before failure
- Network-related failures - Migration is not effective

Architecture of VMFT-LAD



Log Preprocessing

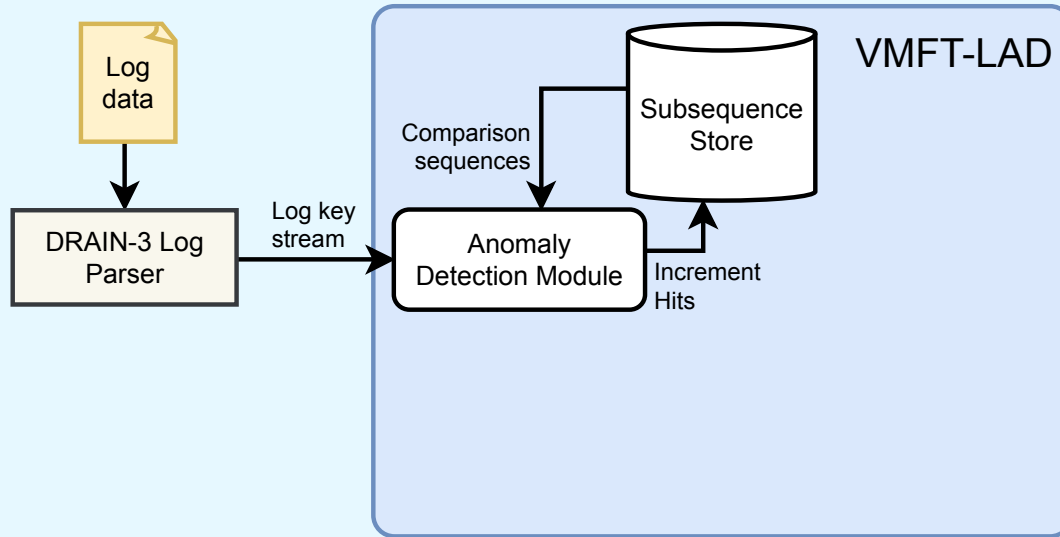
- Collect log data from the host server and the hypervisor
- Extract the timestamp and log message
- Pass the log message through the log parser, Drain-3 to extract log event
- Log parser assigns a unique ID to log events, called log key

```

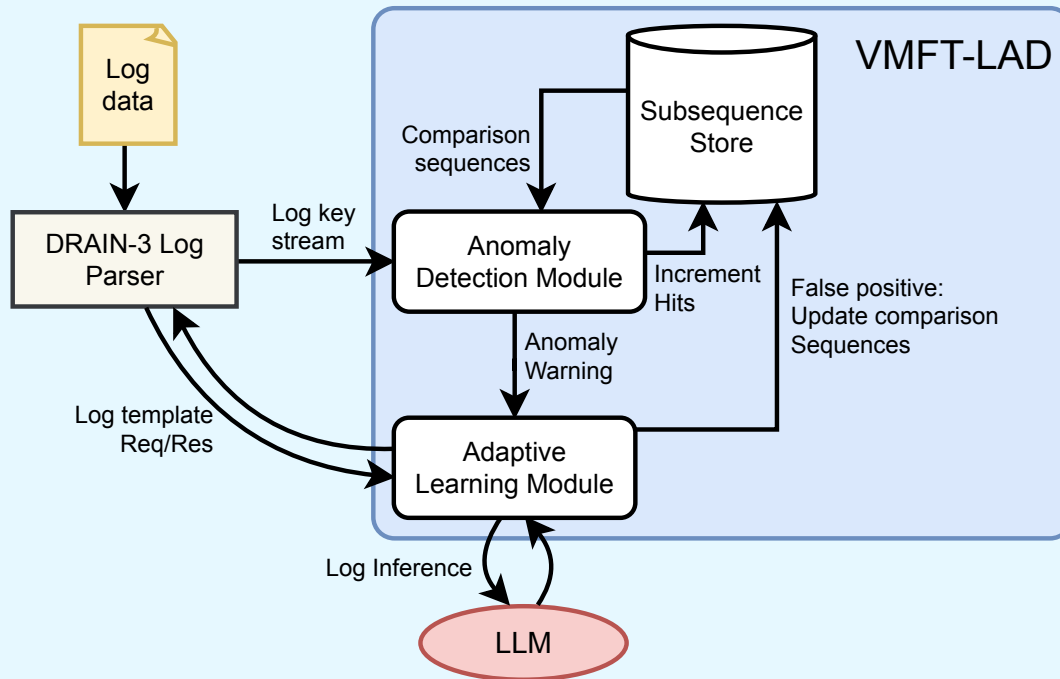
FS-Cache: Loaded 11] FS-Ca
FS-Cache: Netfs 'nfs' registered for caching 71] FS-Ca
NFS: Registering the id_resolver key type 00] NFS:
Key type id_resolver 440 439 195 438 431 432 433 435 435 256 ...
Key type id_legacy
br0: port 2(tap0) entered disabled state 10] br0:
br0: port 2(tap0) entered blocking state 33] br0:
br0: port 2(tap0) entered forwarding state 36] br0:
br0: port 3(tap2) entered disabled state 10] br0:
  
```

Parsed log line (Template)	Log key	Log values
FS-Cache: Loaded	101	[]
FS-Cache: Netfs 'nfs' registered for caching	102	[]
NFS: Registering the id_resolver key type	103	[]
Key type id_resolver	104	[id_resolver]
Key type id_legacy	104	[id_legacy]
br0: port <*>(tap<*>) entered <*> state	105	[2, 0, disabled]
br0: port <*>(tap<*>) entered <*> state	105	[2, 0, blocking]
br0: port <*>(tap<*>) entered <*> state	105	[2, 0, forwarding]
br0: port <*>(tap<*>) entered <*> state	105	[3, 2, disabled]

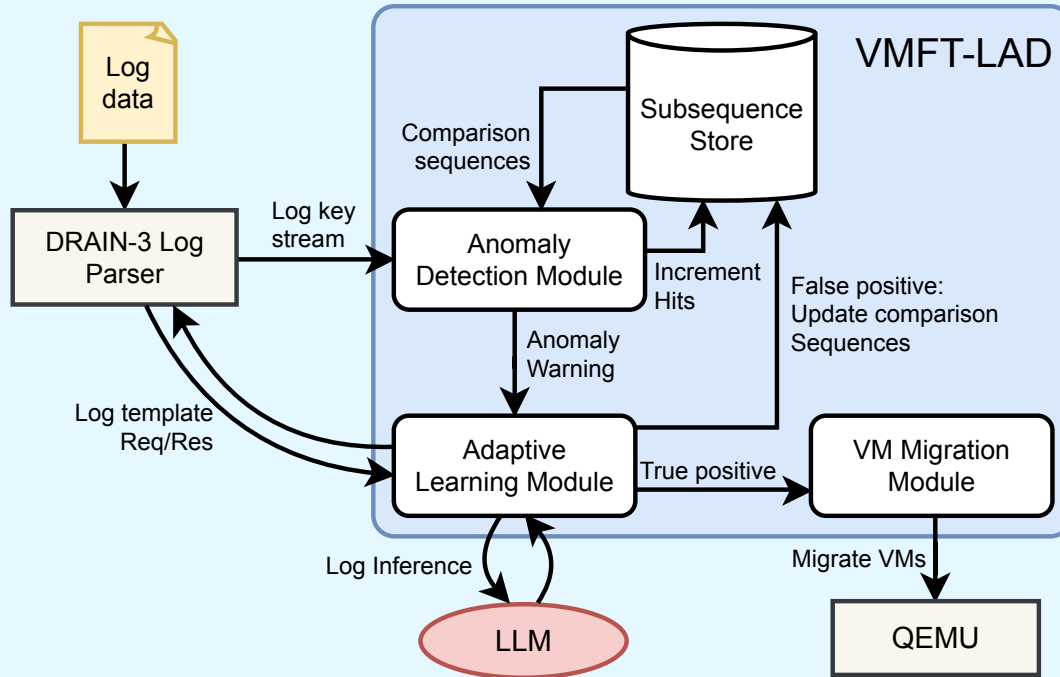
Architecture of VMFT-LAD



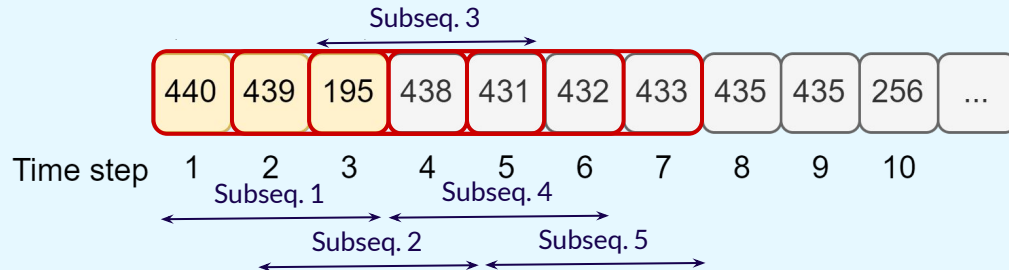
Architecture of VMFT-LAD



Architecture of VMFT-LAD



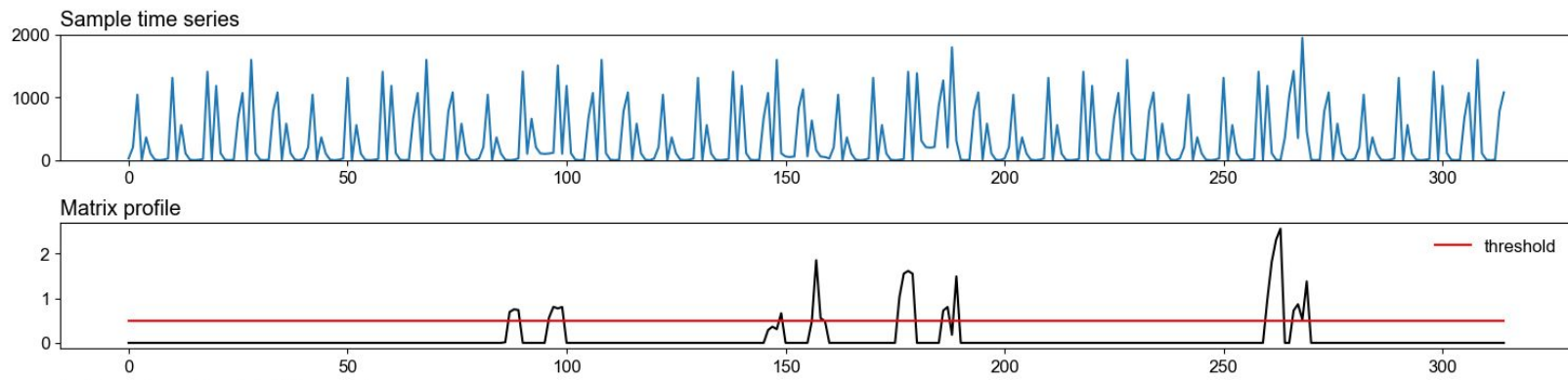
Anomaly Detection: Matrix Profile



- Entire time series is a sequence of values
- The Matrix Profile works by,
 - Extracting rolling windows of sub-sequences of a fixed length
 - Get the Euclidean distance between the current subsequence and all the other subsequences
 - Take the minimum distance as the Matrix Profile value

Anomaly Detection: Matrix Profile


- Matrix profile is the minimum distance of each subsequence
- If the “Minimum distance” is high, it represents an Anomaly, because that subsequence is unique
- If the minimum distance is near 0, it shows that the same subsequence appear somewhere else in the time series

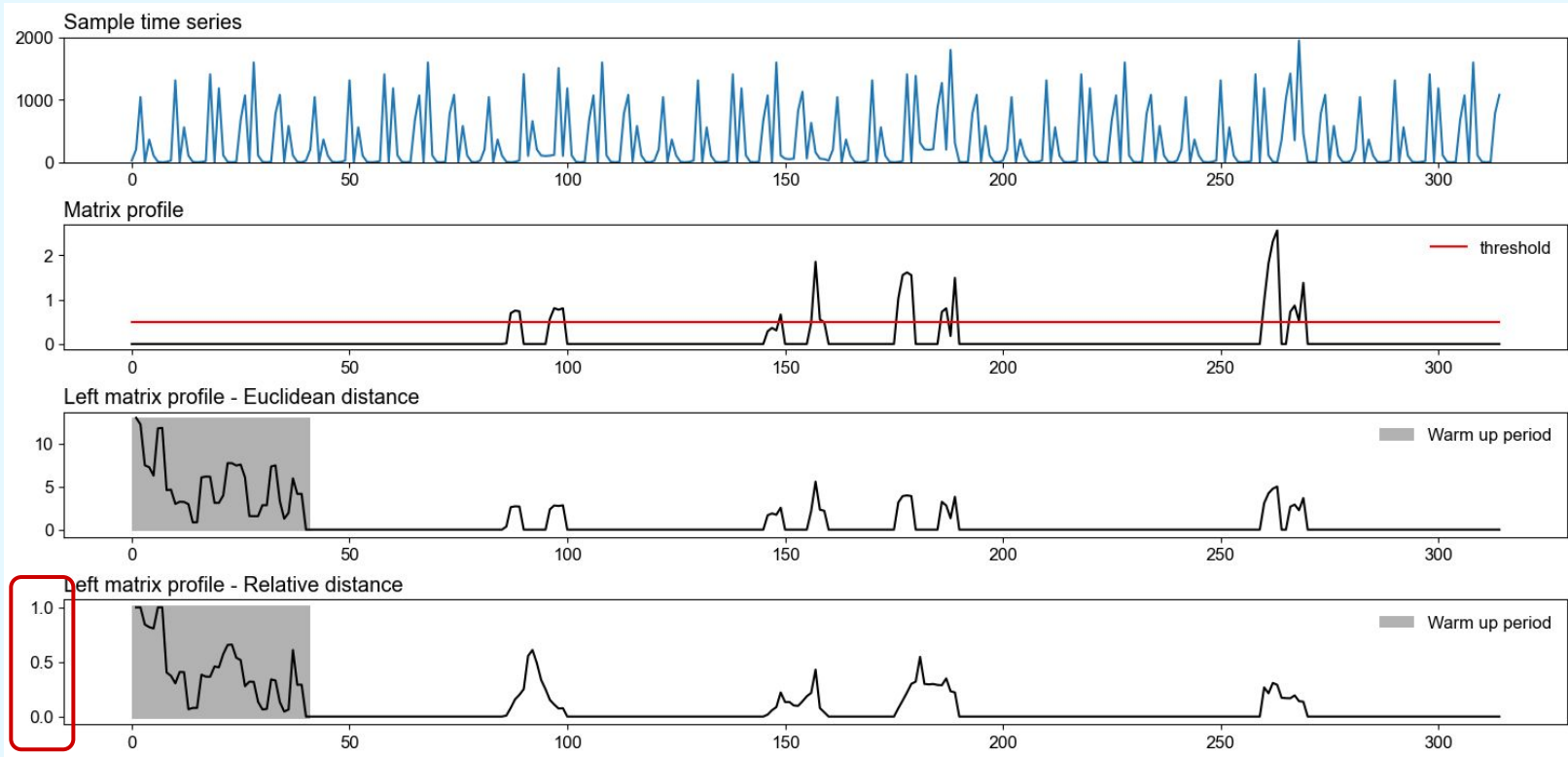


Issues with Matrix Profile

- Matrix profile is designed for offline data analysis
- It considers all subsequences - Inefficient
- Uses euclidean distance - no upper bound

Our modifications

- Modified it for online use, only using a subset of subsequences
 - Use relative distance - bounded between 0 and 1 (Easier to threshold)
 - Use a heap to re-order the subsequences according to their frequency
 - Use a two step thresholding process to identify similar subsequences quickly
- 




Modified Matrix Profile

Training

- Learns the normal state of the system, using the first **M** subsequences
- Trains online, and stores only unique benign subsequences

Anomaly Detection

- Modified Matrix Profile algo. to calculate anomaly score
 - During the calculation if we find a similar subsequence - return immediately
 - If the anomaly score exceeds the threshold - call Adaptive Learning
- 

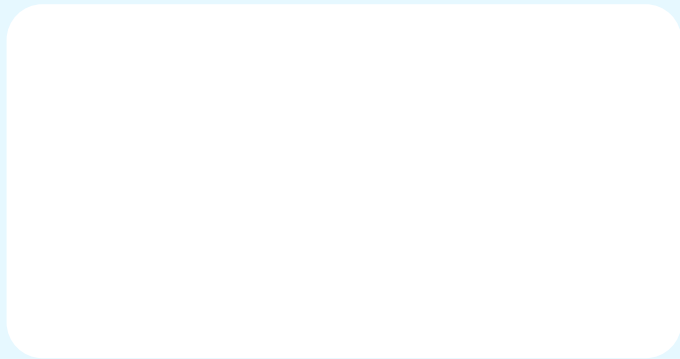
Adaptive Learning

- All failures are anomalies but not all anomalies are failures
- Checks whether the anomalous logs contain failure indicators using an LLM
- If it does: Migrate the VMs
- If not: False positive - Update the subsequence store to prevent future anomalies due to the same subsequence

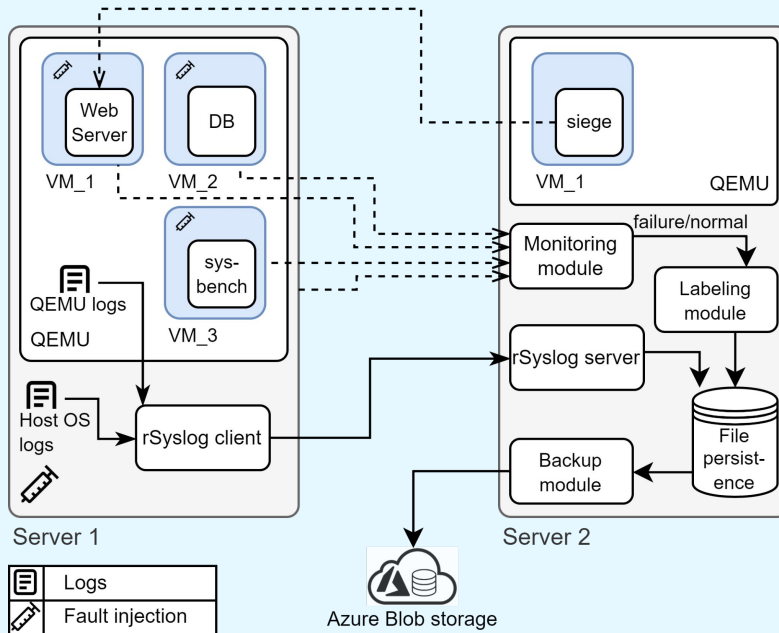


Subsequence Store

- Stores the normal state for comparison
- Orders the subsequences according to the frequency of their usage, using a max-heap
- In-memory storage
- Keeps only unique subsequences



Data collection: Testbed



- Ubuntu host, QEMU-KVM hypervisor
- Rsyslog to stream logs to monitoring server
- Collected log files,
 - kernel log
 - systemd log, application logs
 - QEMU logs for each VM
- Monitoring module - Heartbeat protocol to monitor VMs and Host

Data collection: Failure Scenarios

1. **Out of memory (OOM) failure:** Simulated by over-allocating the total memory for VMs. Stressed the VMs' memory using the stress-ng tool. Failure - host OS invokes the OOM-killer to kill the VM process
2. **Hard disk failure:** Simulated by creating a faulty pseudo disk. Failure - the point where we continuously received unrecoverable read errors for block reads from the faulty disk.
3. **Buffer-I/O error:** Happens when there is a problem transferring data between the storage device and memory. Failure - the point where we encounter several errors.
4. **CPU over-allocation:** Over-allocated the total VM vCPUs. Stressed the CPU on VMs and the host. Failure - no heartbeat pulse from the host or the VMs within 18 seconds.



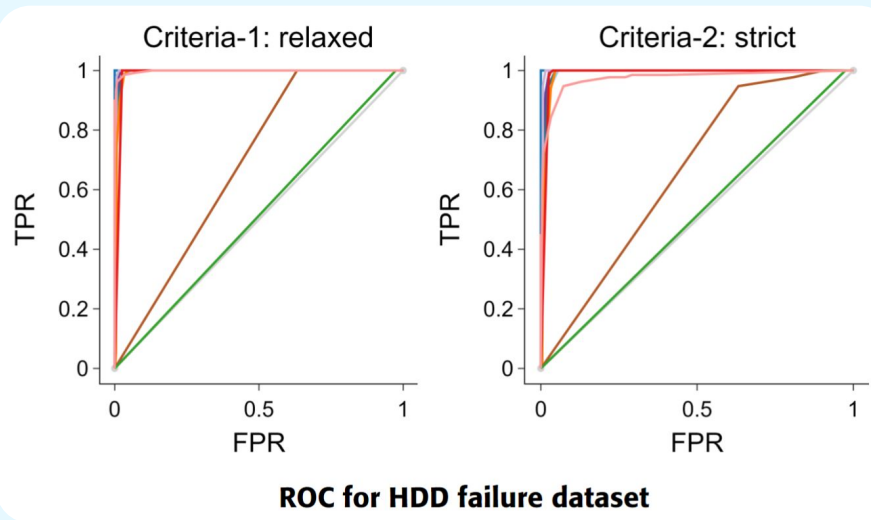
Evaluation: Models

1. **VMFT-LAD without feedback**
2. **VMFT-LAD with LLM feedback**,
 - a. GPT-3.5 turbo
 - b. Falcon 7B, Cyrax 7B, and Emerton Monarch 7B
 - c. Bart Large (Zero-Shot)
3. **HTM** - state-of-the-art anomaly detection model, used in many real-world projects
4. **KNN-CAD** - K-Nearest-Neighbours Conformal Anomaly Detection, non-parametric model
5. **EXPOSE** - EXPected Similarity Estimation, based on a kernel function
6. **ARTime** - based on Adaptive Resonance Theory (ART), outperforms HTM

Evaluation: Criteria

- Evaluated the performance of the models under two different criteria,
 1. **Criteria-1: relaxed** - Checks the models' failure detection ability in general
 - Baseline to verify if the model can identify failures
 2. **Criteria-2: strict** - Checks for models' early failure indicators detection ability
 - Requires the model to detect failures before the failure point
 - Checks if the model allows the successful migration of VMs

Results: ROC



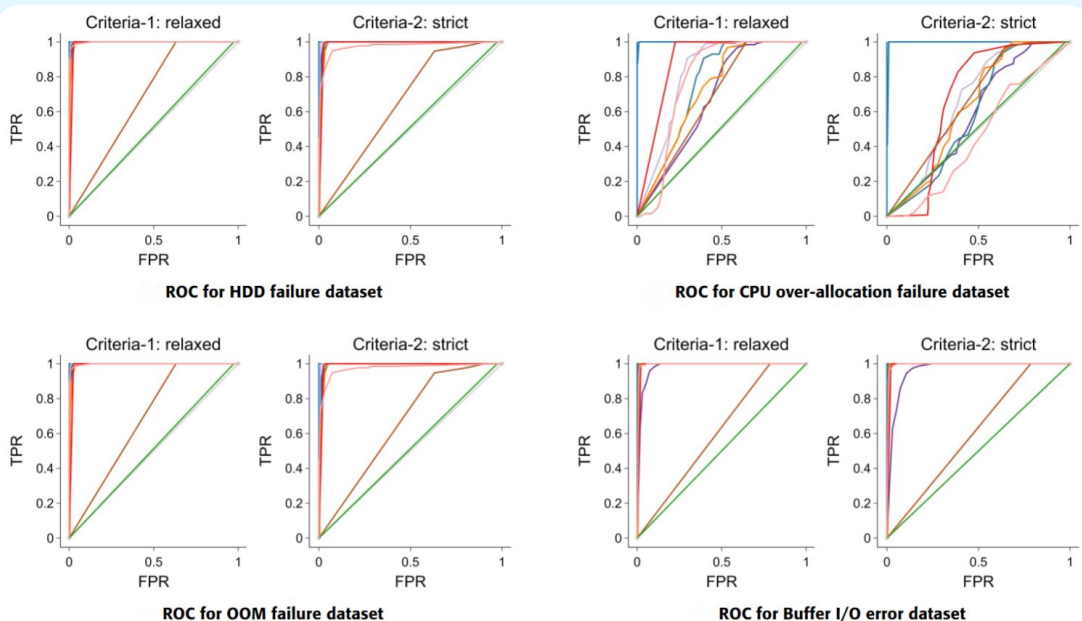
- Graphical representation of the trade-off between the TPR and the FPR at different classification thresholds
- For HDD failure, all models performed well across both criteria, except for KNN-CAD and EXPOSE
- Performance for Criteria-2 is little low - expected

— VMFT-LAD no feedback
 — VMFT-LAD w/ GPT 3.5 turbo
 — VMFT-LAD w/ Falcon 7B
 — VMFT-LAD w/ Emerton Monarch 7B
 — VMFT-LAD w/ Bart Large (Zero Shot)
 — HTM
 — KNN-CAD
 — EXPOSE
 — ARTIME

— VMFT-LAD w/ GPT 3.5 turbo
 — VMFT-LAD w/ Emerton Monarch 7B
 — KNN-CAD

— VMFT-LAD w/ Falcon 7B
 — VMFT-LAD w/ Bart Large (Zero Shot)
 — EXPOSE

Results: ROC



VMFT-LAD no feedback
 VMFT-LAD w/ GPT 3.5 turbo
 VMFT-LAD w/ Falcon 7B
 VMFT-LAD w/ Emerton Monarch 7B
 VMFT-LAD w/ Bart Large (Zero Shot)
 KNN-CAD
 EXPOSE
 HTM
 ARTIME

VMFT-LAD w/ GPT 3.5 turbo
 VMFT-LAD w/ Falcon 7B
 VMFT-LAD w/ Emerton Monarch 7B
 VMFT-LAD w/ Bart Large (Zero Shot)
 KNN-CAD
 EXPOSE

- The results for all the other ROC curves are similar to HDD failure, except for the CPU over allocation
- For CPU over allocation failure, most models had very high false positive rate
- Except for VMFT-LAD w/ GPT-3.5 turbo LLM feedback - performed well with v. low false positives

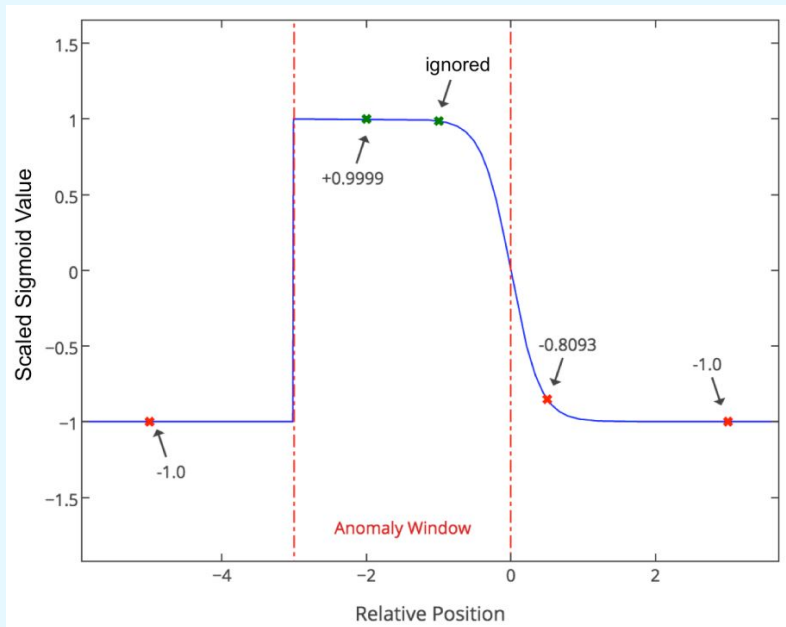
Results: AUC

	Model	Dataset			
		HDD	OOM	Buffer-IO	CPU
Criteria-1: relaxed	VMFT-LAD no feedback	0.992	0.903	0.993	0.678
	VMFT-LAD w/ GPT 3.5 turbo	0.999	0.999	0.999	0.999
	VMFT-LAD w/ Falcon 7B	0.996	0.965	0.978	0.678
	VMFT-LAD w/ Cyrax 7B	0.992	0.974	0.997	0.754
	VMFT-LAD w/ Em. Monarch 7B	0.996	0.909	0.993	0.817
	VMFT-LAD w/ Bart Large	0.991	0.836	0.993	0.725
	HTM	0.988	0.977	0.99	0.888
	KNN-CAD	0.684	0.626	0.607	0.677
	EXPOSE	0.512	0.499	0.501	0.511
	ARTime	0.997	0.986	0.995	0.780
Criteria-2: strict	VMFT-LAD no feedback	0.987	0.657	0.992	0.588
	VMFT-LAD w/ GPT 3.5 turbo	0.999	0.998	0.999	0.996
	VMFT-LAD w/ Falcon 7B	0.993	0.767	0.962	0.588
	VMFT-LAD w/ Cyrax 7B	0.989	0.899	0.997	0.594
	VMFT-LAD w/ Em.n Monarch 7B	0.995	0.667	0.992	0.669
	VMFT-LAD w/ Bart Large	0.987	0.621	0.993	0.634
	HTM	0.987	0.623	0.989	0.680
	KNN-CAD	0.659	0.304	0.607	0.664
	EXPOSE	0.512	0.499	0.501	0.511
	ARTime	0.973	0.546	0.995	0.458

Area Under the Curve (AUC) results for
ROC curves

- VMFT-LAD with GPT-3.5 turbo LLM feedback overall has exceptional results
- VMFT-LAD with other LLMs performed well
- Even without feedback, VMFT-LAD performs better than some of the models
- ARTime and HTM performed well in certain scenarios
- KNN-CAD and EXPOSE are poor performers

Results: Numenta Anomaly Benchmark (NAB)



NAB Scoring Example¹

- Numenta Anomaly Benchmark - designed to evaluate the performance of real-time anomaly detection models
- Prioritizes early anomaly detection and penalizes false positives with negative marks

¹Lavin, Alexander, and Subutai Ahmad. "Evaluating real-time anomaly detection algorithms--the Numenta anomaly benchmark." In 2015 IEEE 14th international conference on machine learning and applications (ICMLA), pp. 38-44. IEEE, 2015.

Results: NAB Scores

	Model	NAB Score		
		Standard	Reward Low FP	Reward Low FN
Criteria-1	VMFT-LAD	98.16	97.77	98.44
	HTM	66.63	61.04	71.64
	KNN-CAD	32.13	-22.05	42.32
	EXPOSE	42.32	37.07	67.33
	ARTime	73.98	56.92	77.89
Criteria-2	VMFT-LAD	90.74	90.36	89.67
	HTM	21.52	16.22	3.13
	KNN-CAD	1.47	-54.94	0.21
	EXPOSE	52.50	18.42	56.17
	ARTime	48.53	26.99	46.55

NAB Scores for evaluated models

- VMFT-LAD shows outstanding performance, across all 3 metrics, over both criteria
- HTM and ARTime shows good scores under the relaxed criteria but falls behind under strict criteria
- As usual EXPOSE and KNN-CAD have low scores, and in some cases negative scores

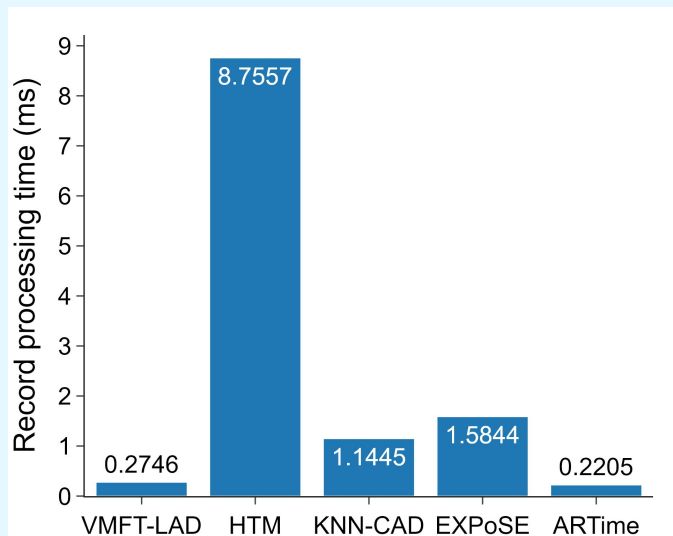
Results: Best Performance

Model	False positive rate	Early detection rate
VMFT-LAD	0.02%	96.28%
HTM	0.07%	62.08%
KNN-CAD	0.74%	76.58%
EXPOSE	0.39%	85.13%
ARTime	0.24%	78.25%

Average False Positive rate and average
Early detection rate of models

- Calculated using the best threshold for each model identified by the NAB optimizer across all datasets
- VMFT-LAD demonstrates the highest early detection rate (96.28%) and the lowest false positive rate (0.02%)

Results: Model Execution Time



- Average execution time to process a single record and determine whether it is anomalous or not. (Lower the better)
- Actual log rates:
 - Log rate: 6.05 s
 - Time between two consecutive records: 164 ms
- All the models can handle online anomaly detection
- VMFT-LAD and ARTIME show sub-millisecond execution time

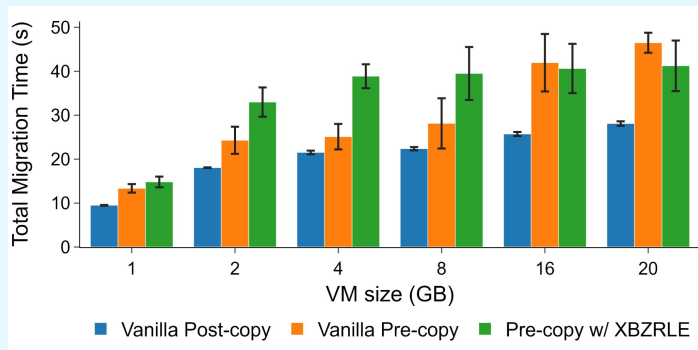
Results: Failure Prediction Time

Model	Dataset			
	HDD	OOM	Buffer-IO	CPU
VMFT-LAD	15.651	3.033	13.338	14.535
HTM	15.675	1.535	13.354	8.462
KNN-CAD	15.901	2.012	12.073	12.555
EXPOSE	15.658	3.136	11.864	23.959
ARTime	15.674	4.057	13.337	13.120

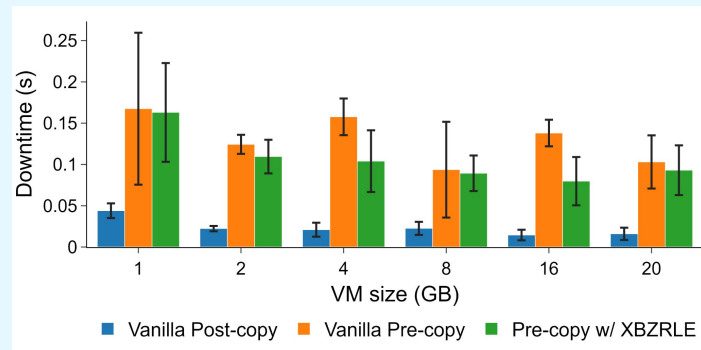
Average detection time before failure
(Minutes)

- Models should accurately detect failures well in advance to allow sufficient time for VM migration
- VMFT-LAD achieves the earliest average detection times
- All the models performed well, except on OOM dataset - all models struggle to identify failure early
- Lack of early pre-failure indicators in OOM

Results: VM Migration Time



The total migration time (Seconds)



Downtime (Seconds)

- VMs running Memcached workload using upto 80% of the VM memory
- Acceptable TMT across all live migration techniques- within 50 seconds, with very low Downtime - around 50 ms to 250 ms
- All models, specially VMFT-LAD, provide sufficient lead time to move VMs before failure

Research Questions

1. **How to effectively utilize VM and server logs for Machine Learning based VM failure prediction?**
 - Solved with the use of efficient log parsing technique and the use of LLMs to understand the content of the log messages
2. **How to develop a generalized VM failure prediction approach using log analysis?**
 - Solved with our implemented model: VMFT-LAD
3. **How can the timing of VM failure prediction be optimized to ensure a successful migration, considering the total time required for the VM migration?**
 - Satisfied by the comprehensive evaluation and the VM migration time analysis

Conclusion

- VMFT-LAD (Virtual Machine Proactive Fault Tolerance using Log-based Anomaly Detection)
 - Modified Matrix Profile for Anomaly Detection
 - Continuously Learns and Adapts to Dynamic Environments
 - Identifies unforeseen faults
 - Little to no human intervention
- High early failure prediction rate (96.28%) with very low false positive rate (0.02%)
- Overthrows state-of-the-art models in NAB benchmark
- Real-Time responsiveness (Faster than most evaluated real-time models)

Research Publication

- Submitted our paper to IEEE Transactions on Cloud Computing (Under review)

P. Senevirathne, S. Cooray, J. Herath, and D. Fernando, "Virtual Machine Proactive Fault Tolerance using Log-based Anomaly Detection," *IEEE Transactions on Cloud Computing*, 2024. (Submitted - under review)



Demo



Thanks!

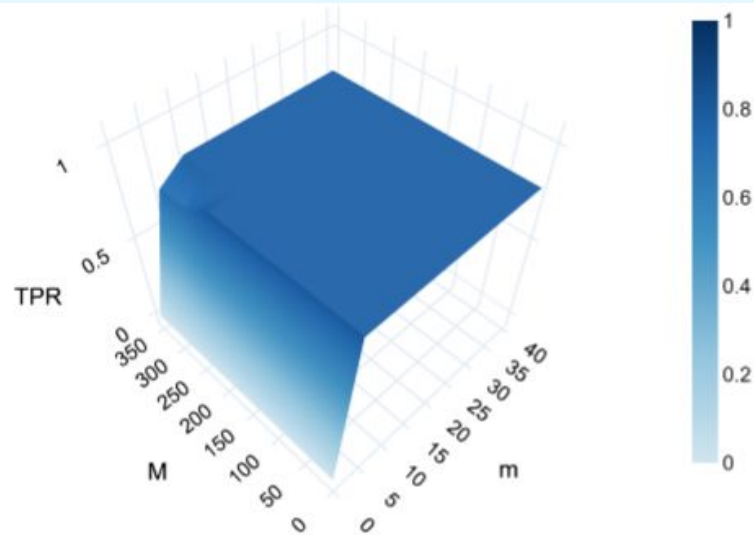
References

- [1] Q. Guan, Z. Zhang, and S. Fu, "Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems.," J. Commun., vol. 7, no. 1, pp. 52–61, 2012.
- [2] X. Sun, K. Chakrabarty, R. Huang, et al., "System-level hardware failure prediction using deep learning," ser. DAC '19, Las Vegas, NV, USA: Association for Computing Machinery, 2019, ISBN:9781450367257. DOI: 10.1145/3316781.3317918. [Online]. Available:<https://doi.org/10.1145/3316781.3317918>.
- [3] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," IEEE transactions on services computing, vol. 15, no. 3, pp. 1411–1422, 2020.
- [4] D. J. Scales, M. Nelson, and G. Venkitachalam, "The design of a practical system for fault-tolerant virtual machines," ACM SIGOPS Operating Systems Review, vol. 44, no. 4, pp. 30–39, 2010.
- [5] D. Saxena and A. K. Singh, "Ofp-tm: An online vm failure prediction and tolerance model towards high availability of cloud computing environments," The Journal of Supercomputing, vol. 78, no. 6, pp. 80038024, 2022.
- [6] S. Nam, J. Hong, J.-H. Yoo, and J. W.-K. Hong, "Virtual machine failure prediction using log analysis," in 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, 2021, pp. 279–284.
- [7] S. Nam, J.-H. Yoo, and J. W.-K. Hong, "Vm failure prediction with log analysis using bert-cnn model," in 2022 18th International Conference on Network and Service Management (CNSM), IEEE, 2022, pp. 331337.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
- [9] Q. Lin, K. Hsieh, Y. Dang, et al., "Predicting node failure in cloud service systems," in Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, 2018, pp. 480–490.

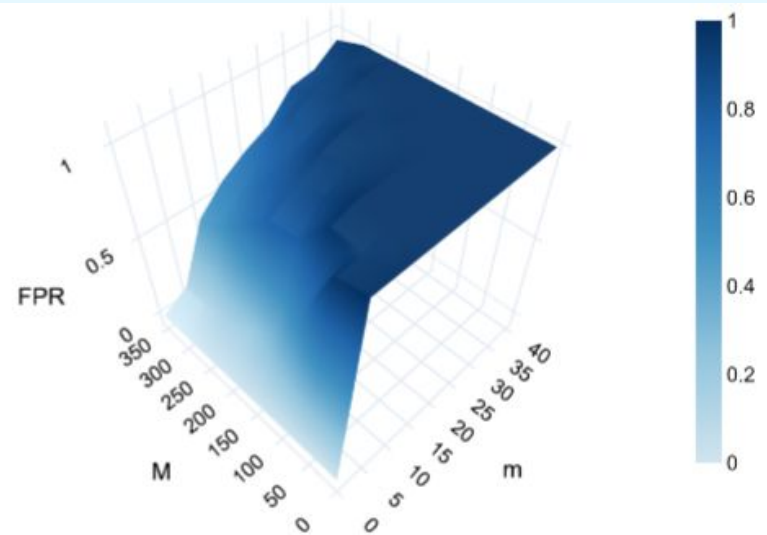
Our Approach: VMFT-LAD

- Detects anomalous logs in **real-time** from the host server and the hypervisor
- Based on Matrix profile - a time series data analysis technique
- Utilizes a fast **heap-based** in-memory store
- **Semi-supervised** - no need to have labeled failure data
- **Learns online** - no need to have a separate offline training
- Leverages LLMs to understand logs, to **continuously adapt** to changing log patterns
- Identifies potential failure indicating logs, **even unseen failures**
- Requires **no human intervention**

Hyperparameter Tuning



(a) Impact of the training period (M) and subsequence length (m) on True Positive Rates



(b) Impact of the training period (M) and subsequence length (m) on False Positive Rates

Hyperparameter Tuning

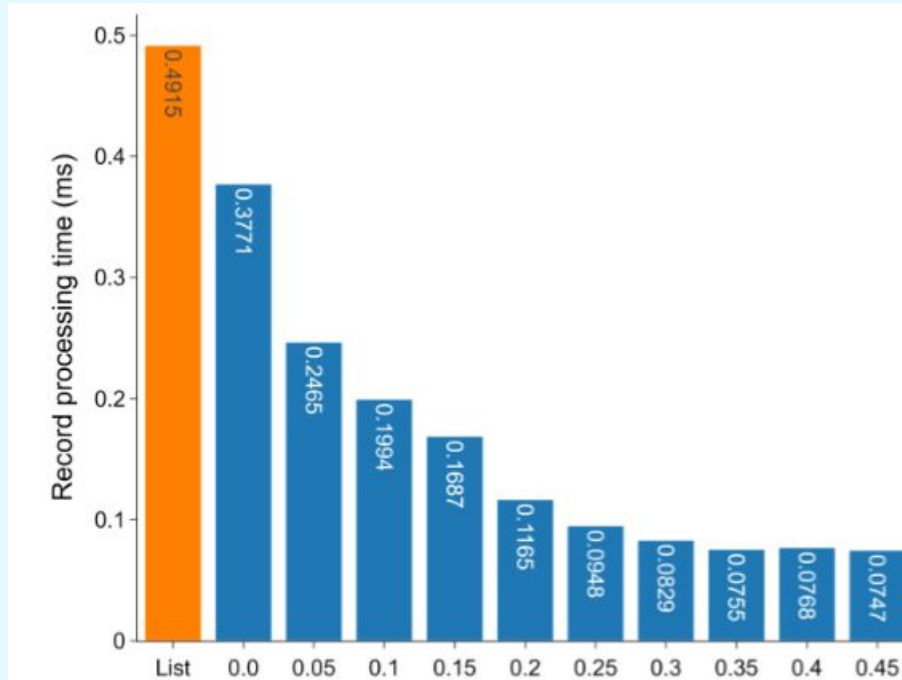


FIGURE 7. Impact of the similarity threshold (η) over the average record processing time. The average record processing time using a list-based implementation is added for comparison.

An Ideal VM Failure Predictor

- Properties of an Ideal VM Failure Predictor¹,
 1. Early identification of failure indicators to facilitate timely migration
 2. Adaptability to changing environments, software updates, and hardware changes
 3. Ability to identify any unforeseen failure or fault types
 4. Capability to work with highly imbalanced data, where failures are rare compared to healthy states
 5. Minimization of false positives and false negatives
 6. Ability to work independently, with little to no human intervention

¹Based on the work of Ahmad et al. [1] and Lin et al. [2]