Interim Presentation

VM Failure Prediction using Log Analysis for Proactive Fault Tolerance

Pratheek Senevirathne
19001622

Supervisor: Dr. Dinuni Fernando, Senior Lecturer (UCSC)
Co-supervisor: Dr. Jerome Dinal Herath, Security Data Scientist (Obsidian Security, USA)

01.

Introduction

Introduction - VMs and Failures

- A Virtual Machine (VM) is a software emulation of a physical computer that allows us to run several Operating Systems (OSs) independently on the same machine.
- Usually used to run an end-user application, or a service.
- VMs and all the other related software and hardware components are failure-prone.
- VM failure → User application failure
- VMs will fail if there is a failure in,
  - Hardware
  - Software
  - Network

Introduction - VM Fault Tolerance

- **Fault Tolerance:** The ability of a system to continue operating without any issues even in failure of its sub-components.
- VM Fault Tolerance approaches,
  - Reactive Fault Tolerance
  - Proactive Fault Tolerance
- Reactive approaches,
  - Reactive Migration
  - Checkpoint and Restart
  - Replication
- Proactive approaches,
  - Preemptive Migration
  - System Rejuvenation
  - Self-healing
- VM migration is one of the main FT approaches, and Google performs over 1 Million VM migrations per month!

Introduction - VM Migration

- Two main VM migration techniques based on “Lively-ness”,
  - Non-Live VM Migration
  - Live VM Migration
- **Non-Live Migration:** Migrate the VM by turning off the VM
- **Live Migration:** Migrate the VM while it is turned on and the applications are running.

VM Live Migration Animation [4]

Introduction - What are Logs?

- Logs are files that contain information about events that occur in a computer system
- Examples,
  - **System events**: Startup, and Shutdown
  - **Hardware changes**
  - **OS/Application events**: Erros, Warnings, performance metrics
  - **Security events**: Login attempts, failed access attempts, etc.
- Log files are unstructured, and each log file is different.
- Log lines contain, the timestamp, log level, application information, and log message.
- A Log line is the output of a logging statement in the program source code
  - logger.log(INFO, “Log message”)

Introduction - Sample Log

Host name Application Log message

```
Sep 12 12:10:46 cloudnet2 kernel: [ 8132.463411] FS-Cache: Loaded
Sep 12 12:10:46 cloudnet2 kernel: [ 8132.541471] FS-Cache: Netfs 'nfs' registered for caching
Sep 12 12:10:46 cloudnet2 kernel: [ 8132.653000] NFS: Registering the id_resolver key type
Sep 12 12:10:46 cloudnet2 kernel: [ 8132.653006] Key type id_resolver registered
Sep 12 12:10:46 cloudnet2 kernel: [ 8132.653006] Key type id_legacy registered
Sep 12 12:17:30 cloudnet2 kernel: [ 8537.724010] br0: port 2(tap0) entered disabled state
Sep 12 12:17:30 cloudnet2 kernel: [ 8536.719833] br0: port 2(tap0) entered blocking state
Sep 12 12:17:30 cloudnet2 kernel: [ 8536.719836] br0: port 2(tap0) entered forwarding state
Sep 12 12:17:31 cloudnet2 kernel: [ 8537.724010] br0: port 3(tap2) entered disabled state
```

Timestamp

```
logger.log("%s: port %d(%s) entered %s state", bridgeName, portId, portName, portState)
```

Introduction - Log Parsing

- Unstructured log data needs to be parsed in some way to make them structured, efficient and easier to analyze
- For this study, we use the **Log template extraction** parsing technique
- The parser outputs parsed log lines as **log key** (a unique ID for each template) and **value** pairs

```
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 registered                   06] Key t
Key type id_legacy registered                     06] Key t
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:
```

Parsing

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 <*> registered	104	[id_resolver]
Key type <*> registered	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]

Introduction - Failures and Logs

- Focus is on VM failures due to hardware and software failures.
  - **Hardware:** Failures/faults related to Memory, CPU, or Secondary Storage.
  - **Software:** Failures/faults related to Host/Guest OS, or Hypervisor.
- All of these failures generate logs
- If we can predict the VM failure using logs, we can use the migration technique to save the VM from failing by proactively moving it to a healthy server **ahead in time**.

02.
Motivation

Motivation

- Only using physical server **resource usage data** to train ML models for the failure prediction
- Most of them have left out the most crucial part of any digital system that keeps track of the system state and the events, **the logs**
- VM failure should be predicted before the time it takes to migrate it, because **VM migration is expensive**
- Most of the existing papers in this area have looked over this basic fact
- Even though the failure prediction may be accurate, the VM may fail during migration due to late failure prediction

03.

Related Work

Related Work

Virtual Machine Failure Prediction using Log Analysis (2021)

Nam, Hong, Yoo, et al. [6]

- Failure prediction of VNFs by analyzing logs.
- NLP technique which uses Word2Vec and a CNN model on processed log data.
- Failure prediction before 5 min to failure. Overall F1 Score: 0.67.

VM Failure Prediction with Log Analysis using BERT-CNN Model (2022)

Nam, Hong, Yoo, et al. [7]

- Failure prediction of VNFs by analyzing logs.
- Using Google BERT with a CNN model on processed log data.
- Failure prediction before 30 min to failure. Overall F1 Score: 0.74.

Related Work

Proactive Live Migration for VNFs using Machine Learning (2021)

Jeong, Van Tu, Yoo, et al. [8]

- “Paging-failure” prediction of vEPC by using VM resource usage info and log data using an LSTM model.
- Count of specific log lines instead of the actual logs.
- Successfully prevents long-term vEPC failures

MING
Microsoft Research (2018)

Lin, Hsieh, Dang, et al. [5]

- Failure prediction of CDC physical servers
- LSTM and Random forest models
- Server ranking system to rank all the servers by their failure-proneness
- First-ever production deployment

04.

Research Gap

Research Gap

- Primarily focused on utilizing physical machine resource usage history to predict failures.
- Overlooked the potential insights provided by hypervisor and host logs.
- Limited to studying specific VMs, such as VNFs.
- Implemented techniques are reactive, because it is less expensive and easier to implement.
- Migration time when predicting failures has been disregarded.

05.

Research Questions

Research Questions

01. How to effectively utilize hypervisor and host logs for machine learning-based VM failure prediction?

02. How to develop a generalized VM failure prediction approach using log analysis, enabling its applicability to a wide range of generic VMs?

03. How can the timing of VM failure prediction be optimized to ensure successful VM migration to a healthy PM, considering the total time required for the VM migration?

06.

Objectives

Objectives

- To develop a **generalized ML-based** prediction approach that leverages key events and indicators present in VM and PM logs to **predict failures ahead of time** in a variety of VMs.
- To make the prediction **time-aware** so that it considers the time required for migration to ensure successful VM migration to another physical machine.
- To **evaluate** the proposed prediction approach and compare its performance against existing techniques.

07.

Scope

Scope

In Scope

- VM failure prediction ahead of time by analyzing logs using a ML-based approach
- Online failure prediction
- VM Live Migration in QEMU-KVM on Ubuntu host OS
- LAN based VM migrations
- Implementation of a working prototype

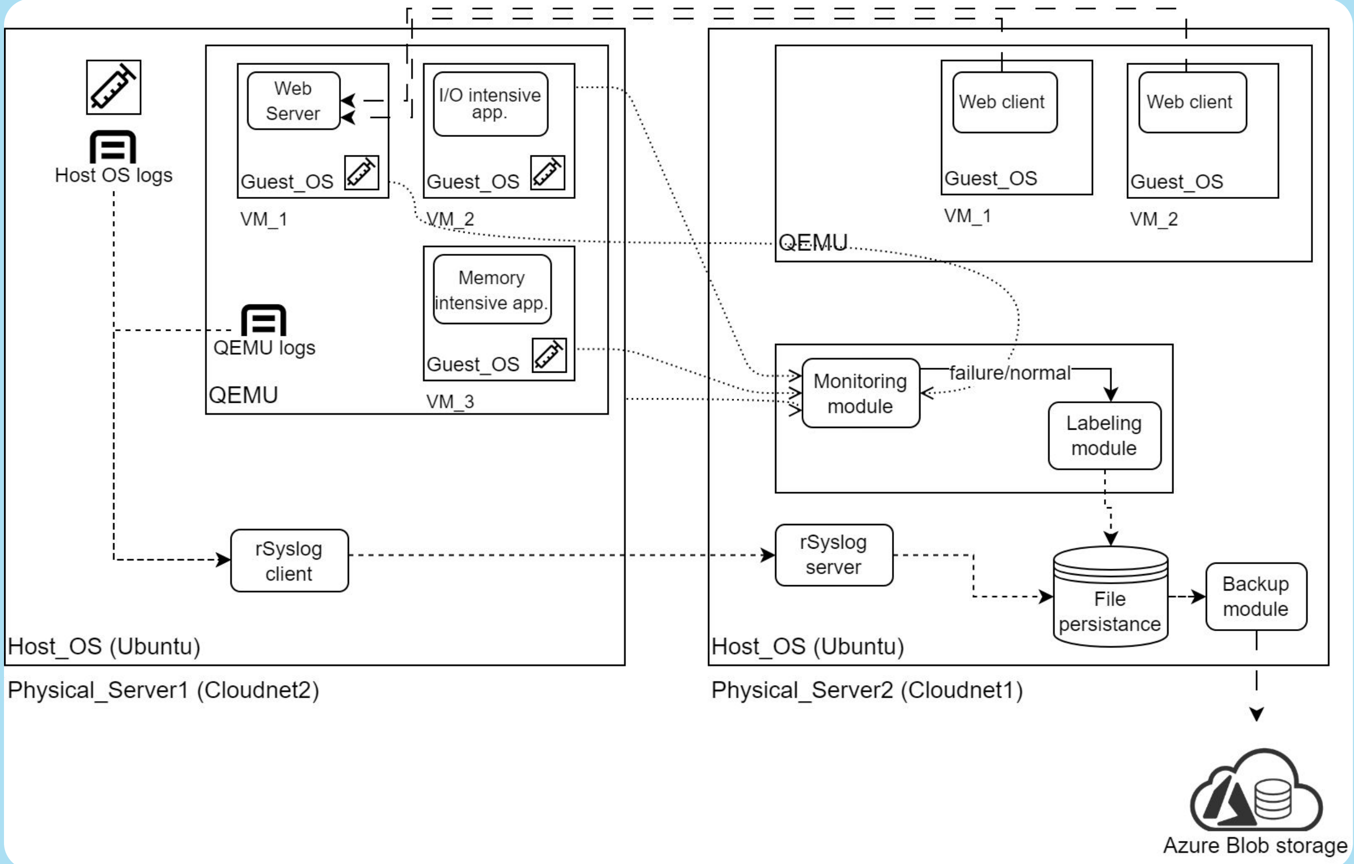
Out of Scope

- Handling unexpected VM failure situations - No pre-failure logs
- Network-related failures - Migration is not effective

08.

Research Approach

Legend	
(log icon)	Log (system, application, etc.)
(fault icon)	Fault injection
- - - ->	Log data
— — →	Network requests
·······>	Heartbeat messages
——→	Control signals or messages



High-level architecture of the testbed

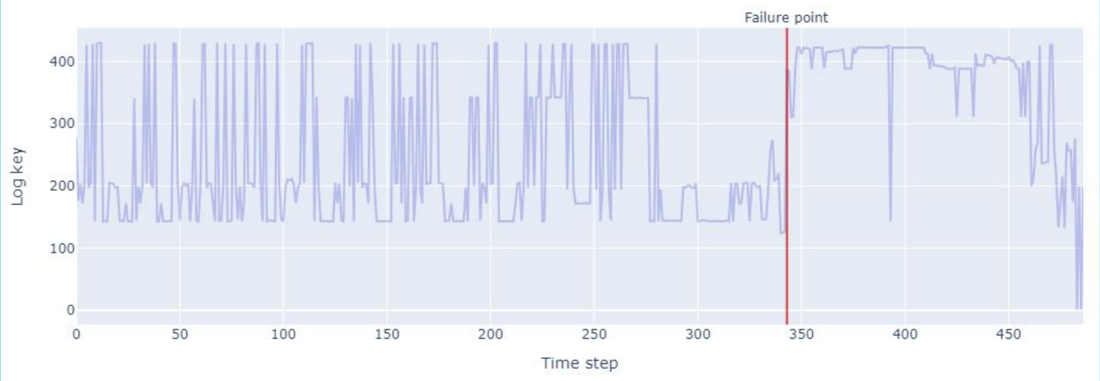
Data collection and Preprocessing

- Collected several log data samples on the testbed (Simulated OOM, HDD failures, and Benign data)
- For data pre-processing, we extract the timestamp and the log message from raw logs
- The message is then passed to the log parser to get the log template, log key and log values.
- Tested two SOTA log parsers,
    - SPELL
    - DRAIN
- Chose DRAIN because of higher accuracy and IBMs open-source DRAIN implementation (Drain3)
- Removed CRON-related logs to reduce noise from the dataset

Failure simulation - Logs

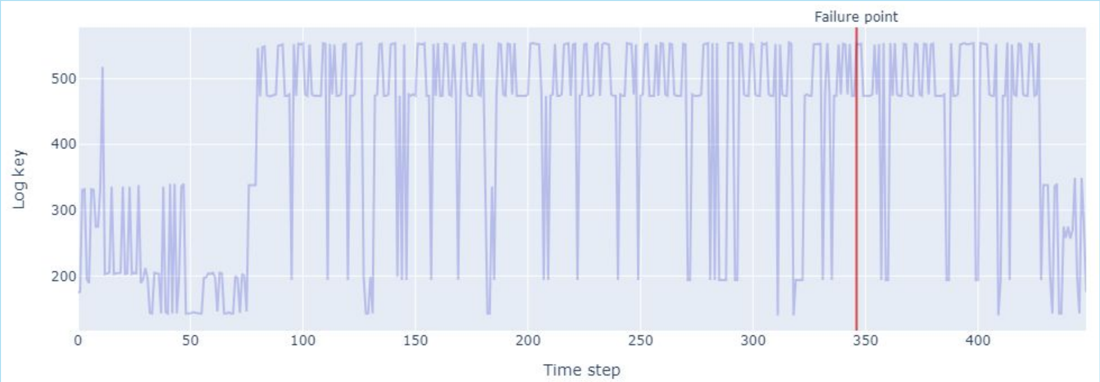
OOM Failure Logs

```
qemu-system-x86 invoked oom-killer: gfp_mask=0x100cca(GFP_HIGHUSER_MOVABLE), ord
CPU: 1 PID: 3077 Comm: qemu-system-x86 Not tainted 5.4.0-162-generic #179-Ubuntu
Hardware name: Hewlett-Packard HP Z620 Workstation/158A, BIOS J61 v03.69 03/25/2
Call Trace:
 dump_stack+0x6d/0x8b
 dump_header+0x4f/0x1eb
 oom_kill_process.cold+0xb/0x10
 out_of_memory+0x1cf/0x500
 __alloc_pages_slowpath+0xdde/0xeb0
 __alloc_pages_nodemask+0x2d0/0x320
```



HDD Failure Logs (Unrecoverable read errors)

```
blk_update_request: critical medium error, dev sdb, sector 4656 op 0x0:(READ) fl
Buffer I/O error on dev sdb, logical block 582, async page read
sd 9:0:0:0: [sdb] tag#178 FAILED Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
sd 9:0:0:0: [sdb] tag#178 Sense Key : Medium Error [current]
sd 9:0:0:0: [sdb] tag#178 Add. Sense: Unrecovered read error
sd 9:0:0:0: [sdb] tag#178 CDB: Read(10) 28 00 00 00 11 e0 00 01 00 00
blk_update_request: critical medium error, dev sdb, sector 4576 op 0x0:(READ) fl
sd 9:0:0:0: [sdb] tag#190 FAILED Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
sd 9:0:0:0: [sdb] tag#190 Sense Key : Medium Error [current]
sd 9:0:0:0: [sdb] tag#190 Add. Sense: Unrecovered read error
sd 9:0:0:0: [sdb] tag#190 CDB: Read(10) 28 00 00 00 12 30 00 00 08 00
```



Overall Idea

- Find anomalous data in the log dataset
- If there are failures in the dataset, then the dataset should look different when compared to data in normal operation
- This difference indicates an anomaly in the data sample
- Anomaly suggests a possible fault in the system, which may lead to VM failures
- If we can detect anomalies in the data set, we can predict VM failures

Real-time Anomaly Detection Models

- Used for real-time streaming data
- Run semi-supervised
- Predictions made online
- Trains on limited benign region
- We trained and tested tested following models on our data as the baseline,
    - KNNCAD
    - HTM
    - ARTime
    - EXPoSE

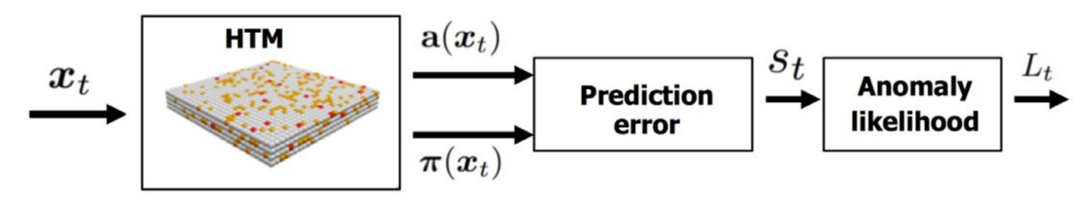
09. Baseline Models

KNNCAD

- K-Nearest-Neighbours Conformal Anomaly Detection
- Measures mutual dissimilarity of observations
- Based on Conformal Prediction,
    - Computes a probability value based on Non-conformity Measure (NCM), using training set, for each new observation.
    - NCM is a function that gives some measure of dissimilarity between new observation and training set

HTM

- Hierarchical Temporal Memory
- A theoretical framework inspired by the structure and function of the human neocortex - attention, thought, and perception
- Relatively new and completely different from the usual perceptron based models like ANNs and DNNs
- Designed to learn and recognize temporal patterns in data - works well in anomaly detection, prediction, and classification



Anomaly detection using HTM [10]

ARTime

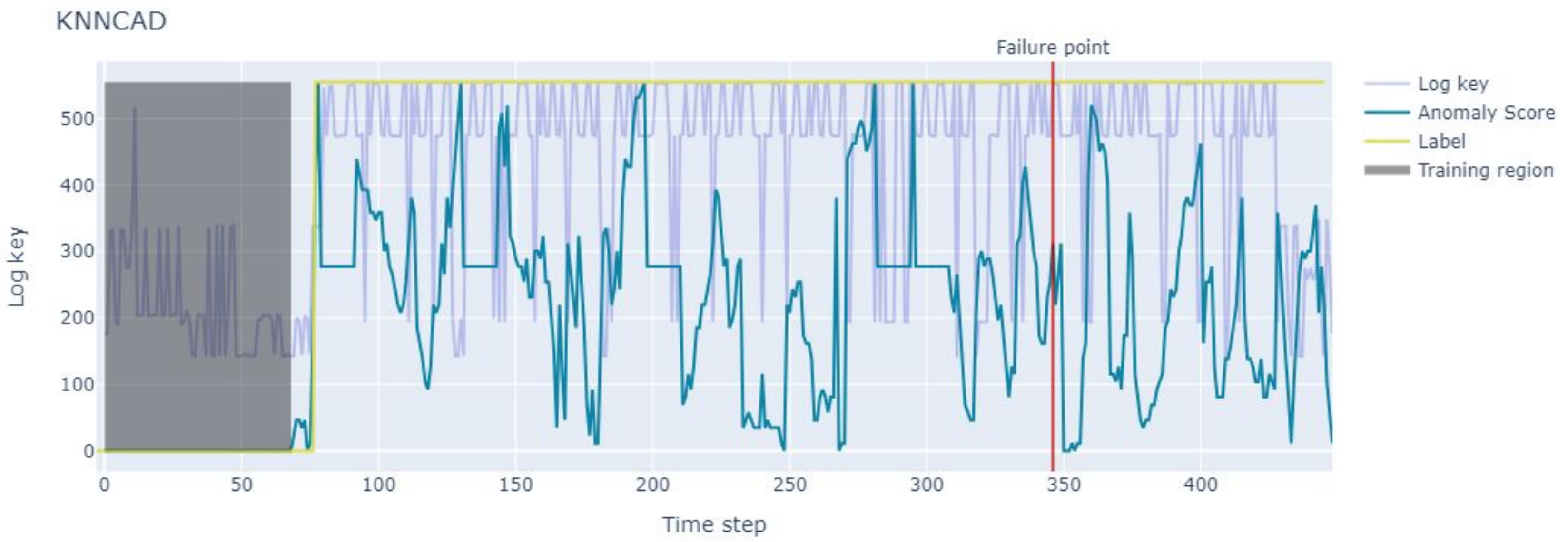
- Based on Adaptive Resonance Theory (ART)
- ART model is designed to explain how our brain is capable of rapid learning and recognition in real-time
- **Resonance:** Match between incoming sensory information and stored memories
- **Vigilance:** Parameter which controls the sensitivity of the model
- If the input pattern does not **resonate** with existing memory (if the output falls below the vigilance threshold), it is considered an anomaly

EXPoSE

- EXPected Similarity Estimation
- Non-parametric anomaly detection algorithm
- First maps the data into a high-dimensional Hilbert space (Infinite dimensional vector space) using a kernel function
- Then computes the expected similarity between a new data point and the distribution of benign data
- If the expected similarity is low, then the new data point is likely to be an anomaly

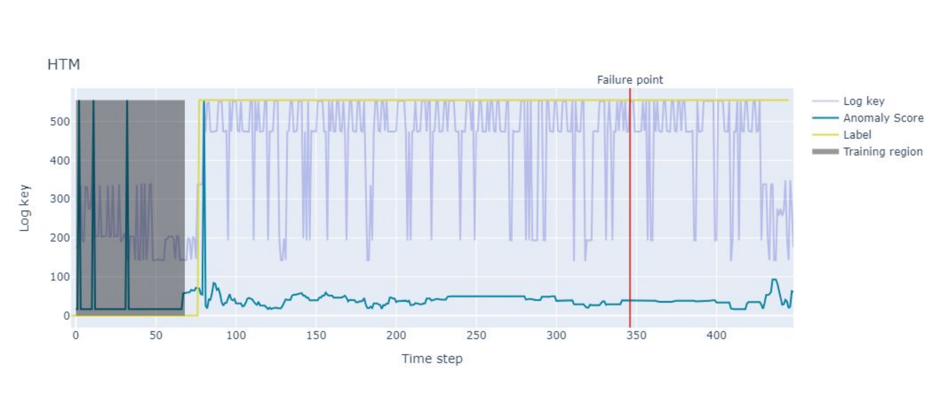
10. Preliminary Results

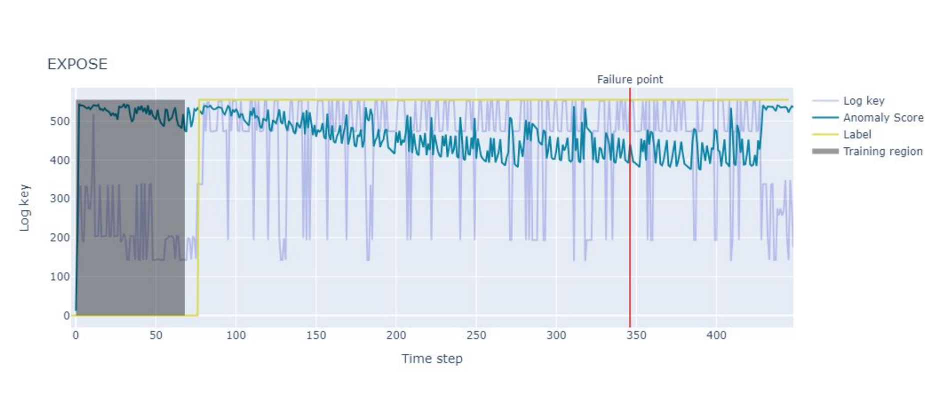
Results - HDD failure

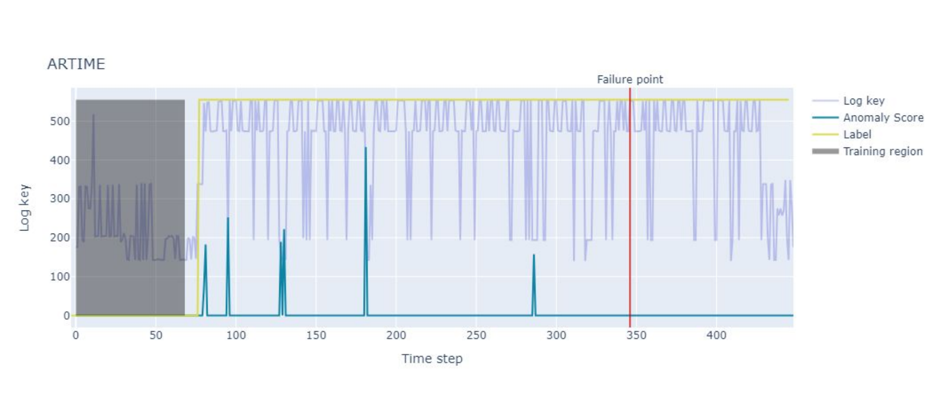


- HDD failure with unrecoverable read errors
- Anomaly detected ~16 minutes before failure

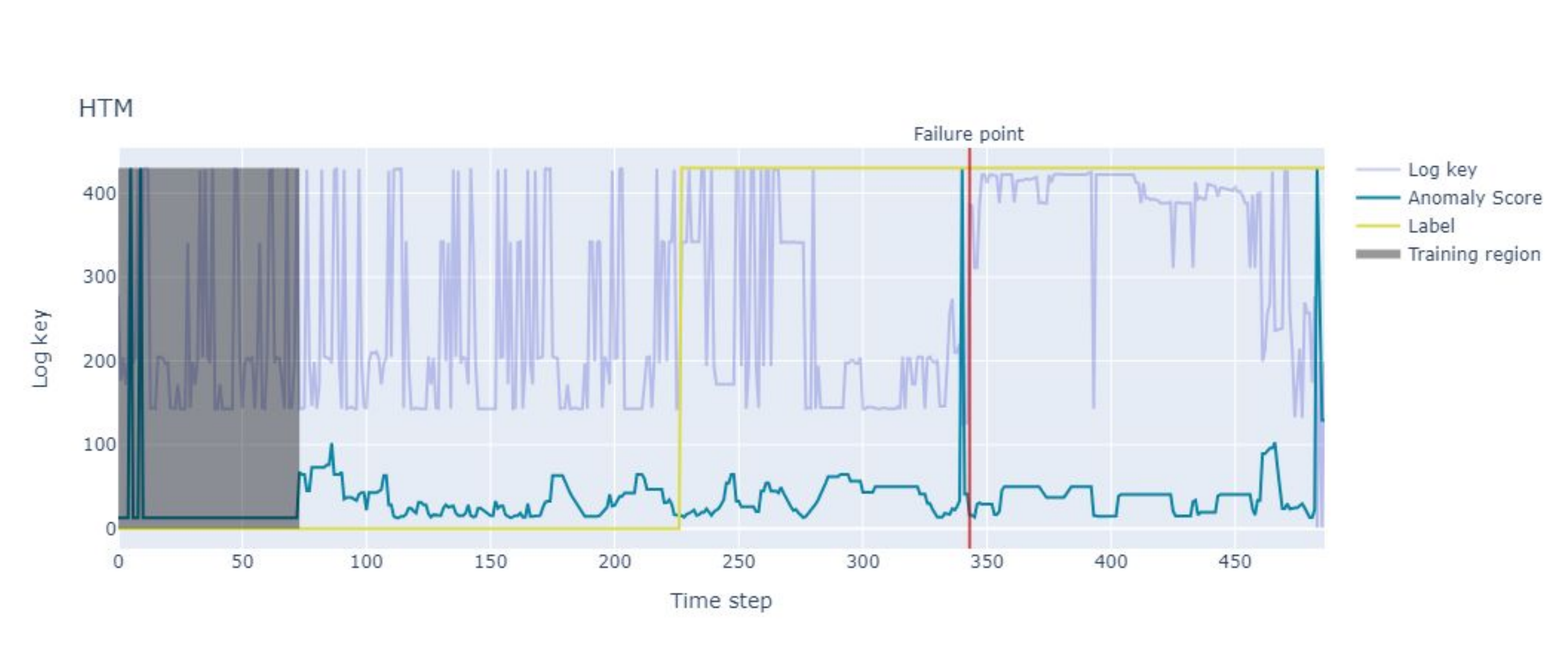
Results - HDD failure





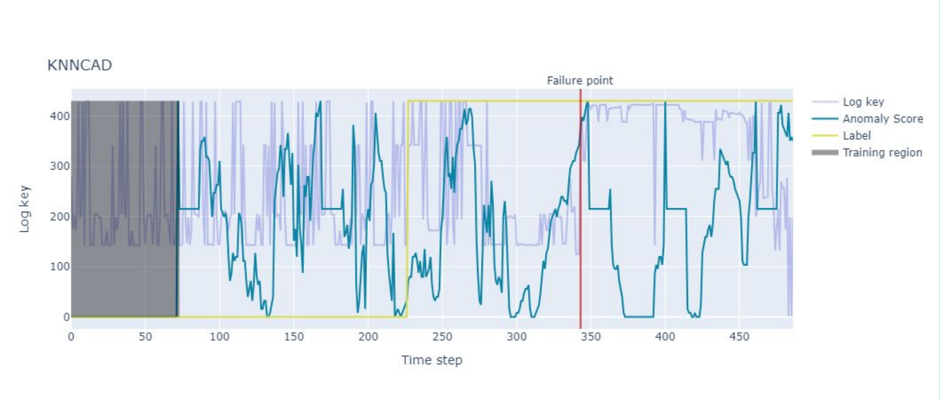


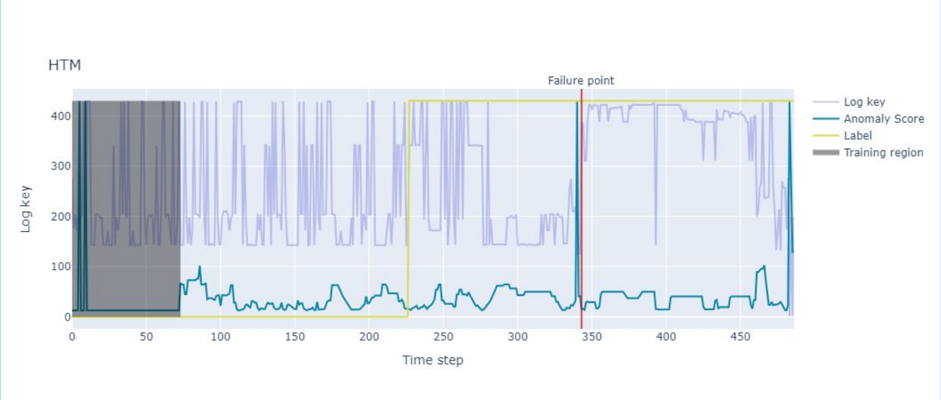
Results - OOM failure

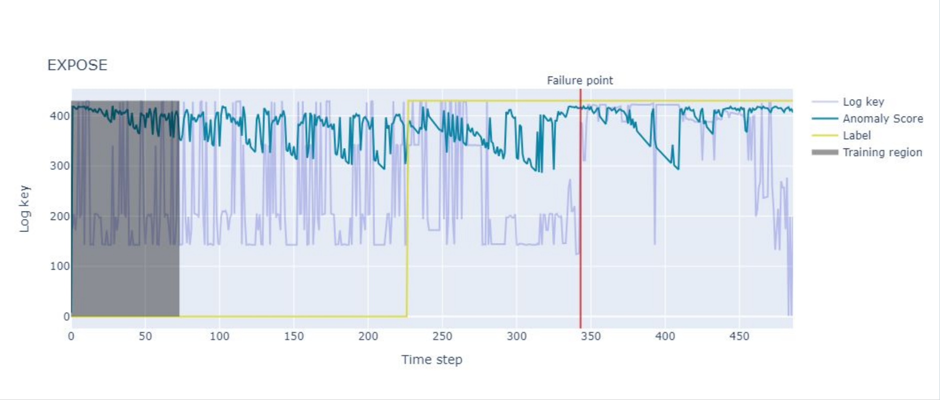


- OOM-Failure: Kernel invoking the OOM-Killer to kill the VM instance
- Anomaly detected ~7 minutes before failure

Results - OOM failure







Results - VM Migration Time

- Pre-copy[1] (8GB RAM):
  - Idle: 10.3s
  - Busy: 72s
- Post-copy[2] (8GB RAM):
  - Idle: 6s
  - Busy: 72.4s
- Worst case total migration time: 73s - 1.2 mins
- Previous prediction times (~7 minutes for OOM, ~15 minutes for HDD failure) are sufficient

Courtesy of Ms. B. F. Ilma

1 - Pre-copy experiments

2 - Post-copy experiments

Next Steps,

- Collect more datasets for failure simulations
- Develop a ML model to detect VM failure anomalies earlier, with less false positives
- Evaluate the developed model and compare with the baseline models
- Implement a prototype system to integrate failure prediction model with QEMU live migration
- Evaluate the prototype system

Evaluation - ML Models

- By looking at preliminary results,
  a. HTM and ARTime performed better than EXPoSE and KNNCAD
  b. KNNCAD outputs a noisy result
  c. EXPoSE model did not perform well
- Define: TP, TN, FP, FN
- Two levels of evaluation for models,
  1. Model detects an anomaly anywhere in the anomalous region
  2. Model detects the anomaly in the anomalous region, before the failure point
- For each level of evaluation, we can plot the ROC curve to see which model performs best across all datasets

Evaluation - Prototype System

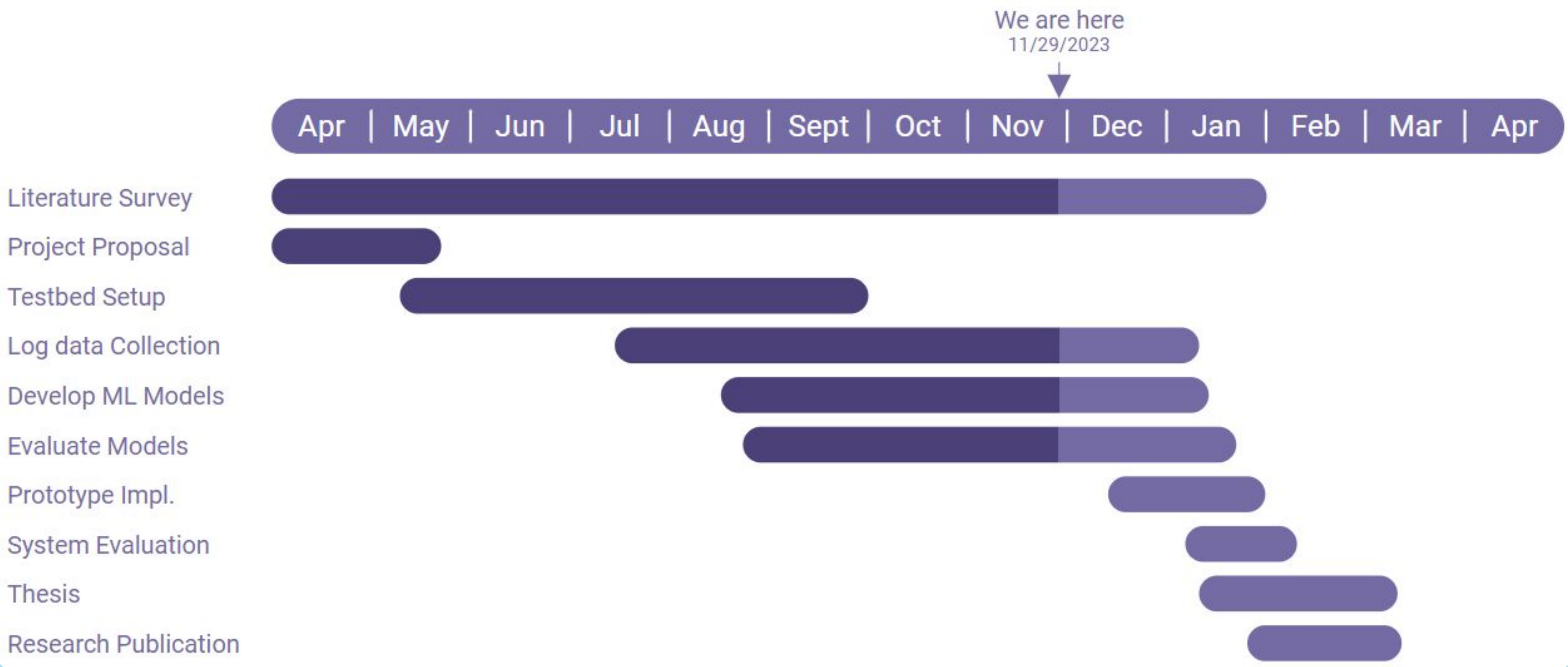
- Prefer low false positives, and early failure prediction
- Reliability of the system (rate of failures averted)
  - No. of failures averted / Total failure simulations
- Effect of the failure prediction system on the host (resource usage with and without the system),
  - Memory
  - CPU
  - I/O

11. Improvements from Feedback

Improvements from Feedback

- Simulation of VM failures where guest OS fails is not needed because the software failures inside the VM will not be solved by migration
  - Decided not to collect VM logs from the guest OS and the applications running inside the VM
  - Now we are not restricted to Linux-based VMs (guest OS)
  - Predict failure in any VM that is supported by QEMU
- New => We do not test for dependencies among failure scenarios, for example a OOM error may cause high CPU usage resulting in another failure
- New => In scope, points 3,4,5 are not related to research
- New =>

Timeline



We are here
11/29/2023
Apr | May | Jun | Jul | Aug | Sept | Oct | Nov | Dec | Jan | Feb | Mar | Apr
Literature Survey
Project Proposal
Testbed Setup
Log data Collection
Develop ML Models
Evaluate Models
Prototype Impl.
System Evaluation
Thesis
Research Publication

References

[1] R. Birke, I. Giurgiu, L. Y. Chen, D. Wiesmann, and T. Engbersen, “Failure analysis of virtual and physical machines: Patterns, causes and characteristics,” in 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014, pp. 1–12. doi: 10.1109/DSN.2014.18.

[2] C. Engelmann, G. R. Vallee, T. Naughton, and S. L. Scott, “Proactive fault tolerance using preemptive migration,” in 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing 2009, pp. 252–257. doi: 10.1109/PDP.2009.31.

[3] A. Polze, P. Tr öger, and F. Salfner, “Timely virtual machine migration for proactive fault tolerance,” in 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, 2011, pp. 234–243. doi: 10.1109/ISORCW.2011.42.

[4] Nvidia. “Datacenter virtual gpu data migration animation.” (2023), [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/solutions/vgpu-migration/data-center-virtual-gpu-data-migration-animation-625-ud.gif (visited on06/10/2023).

[5] 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.

[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. 331–337.

[8] S. Jeong, N. Van Tu, J.-H. Yoo, and J. W.-K. Hong, “Proactive live migration for virtual network functions using machine learning,” in 2021 17th International Conference on Network and Service Management (CNSM), IEEE, 2021, pp. 335–339

[9] A. Ruprecht, D. Jones, D. Shiraev, et al., “Vm live migration at scale,” in Proceedings of the 14th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, ser. VEE ’18, Williamsburg, VA, USA: Association for Computing Machinery, 2018, pp. 45–56, isbn: 9781450355797. doi: 10.1145/3186411.3186415. [Online]. Available: https://doi.org/10.1145/3186411.3186415.

References

[10] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” Neurocomputing, vol. 262, pp. 134–147, 2017.

[11] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, “Drain: An online log parsing approach with fixed depth tree,” in 2017 IEEE international conference on web services (ICWS), IEEE, 2017, pp. 33–40.

[12] E. Burnaev and V. Ishimtsev, “Conformalized density-and distance-based anomaly detection in time-series data,” arXiv preprint arXiv:1608.04585, 2016.

[13] M. Hampton. “Artimenab.” (2021), [Online]. Available: https://github.com/markNZed/ARTimeNAB.jl (visited on 10/29/2023).

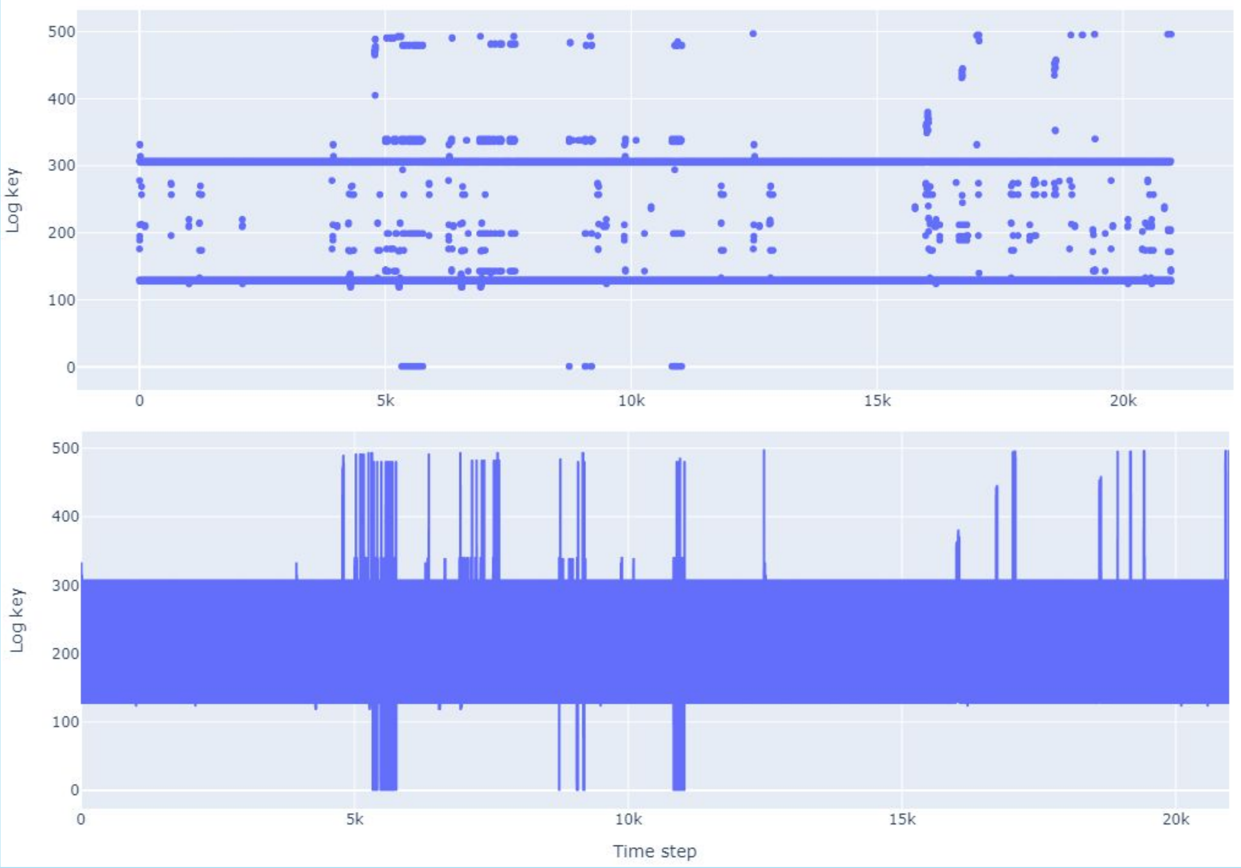
[14] M. Schneider, W. Ertel, and F. Ramos, “Expected similarity estimation for large-scale batch and streaming anomaly detection,” Machine Learning, vol. 105, pp. 305–333, 2016.

[15] M. Du and F. Li, “Spell: Streaming parsing of system event logs,” in 2016IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 859–864.

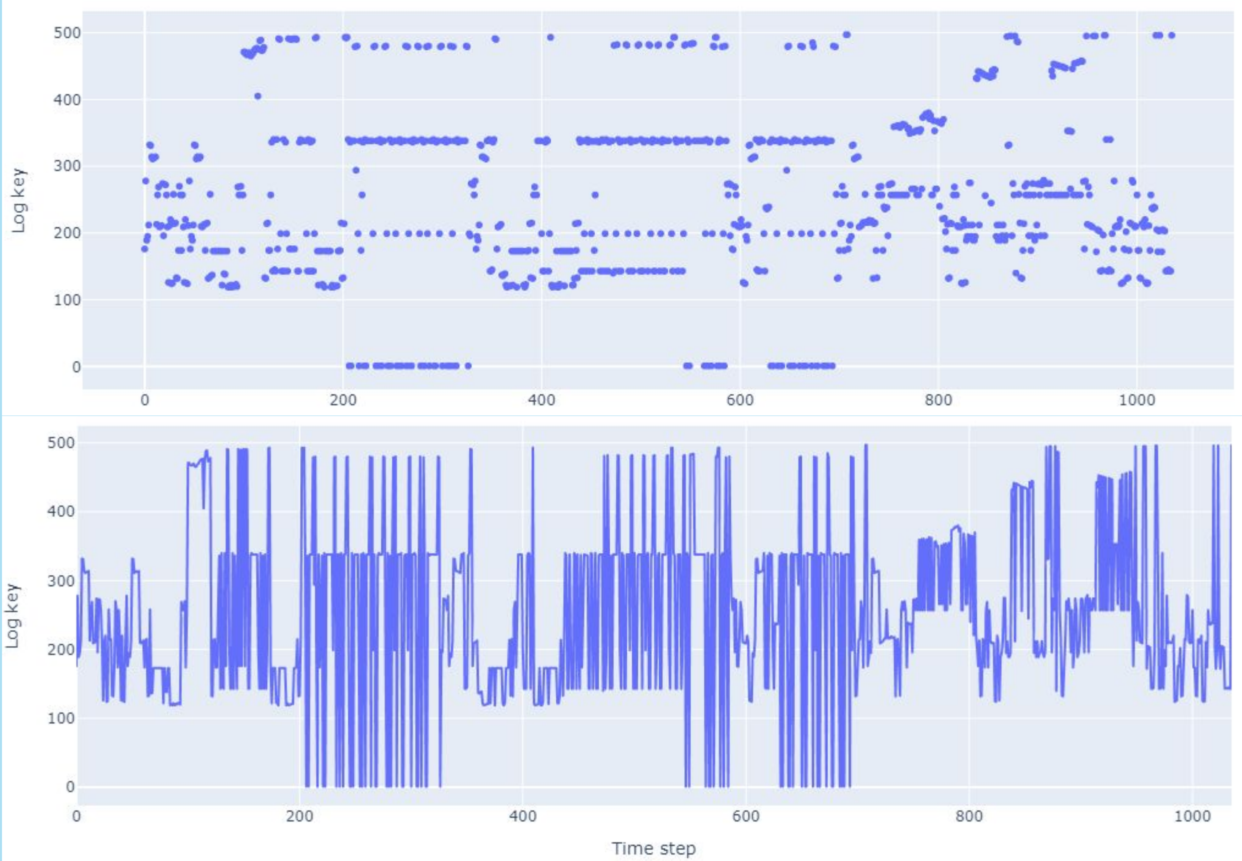
[16] D. Ohana. “Use open source drain3 log-template mining project to monitor for network outages.” (2020), [Online]. Available: https://developer.ibm.com/blogs/how- mining- log- templates- can- help- ai-ops-in-cloud-scale-data-centers/ (visited on 10/29/2023)

Thanks!

Before CRON log removal



After CRON log removal



Total collected datasets

- 17 Collected datasets
  - 7 - Software failures (OOM and soft crashes)
  - 9 - Benign data
  - 1 - HDD failure (Simulated unrecoverable read errors)
- Discarded : 6
  - 5 - Old testbed setup
  - 1- Qemu source error (segfault)

TP, TN, FP, FN - Pseudocode

Criteria 1: Model finds anomaly anywhere in the anomalous region

- Ignore the training region
- Prediction_point = first point of anomaly score that is equal or greater than the current threshold [0,1]

- if the dataset has a failure,
    - if the model did not detect an anomaly: FN
    - else if the prediction_point is in anomalous region: TP
    - else if the prediction_point is not in anomalous region: FP

- if the dataset does not have a failure,
    - if the model did not find an anomaly: TN
    - else (the model finds an anomaly): FP

TP, TN, FP, FN - Pseudocode

Criteria 2: Model finds the anomaly in the anomalous region before the failure point

- Ignore the training region
- Prediction_point = first point of anomaly score that is equal or greater than the current threshold [0,1]

- if the dataset has a failure,
    - if the model did not detect an anomaly: FN
    - else if the prediction_point is inside the anomaly window,
        - if prediction_point is before the failure: TP
        - else (after failure point): TP_LATE
    - else (prediction_point is not in anomalous region): FP

- if the dataset does not have a failure:
    - if the model finds an anomaly: FP
    - else (does not find an anomaly): TN

Appendix

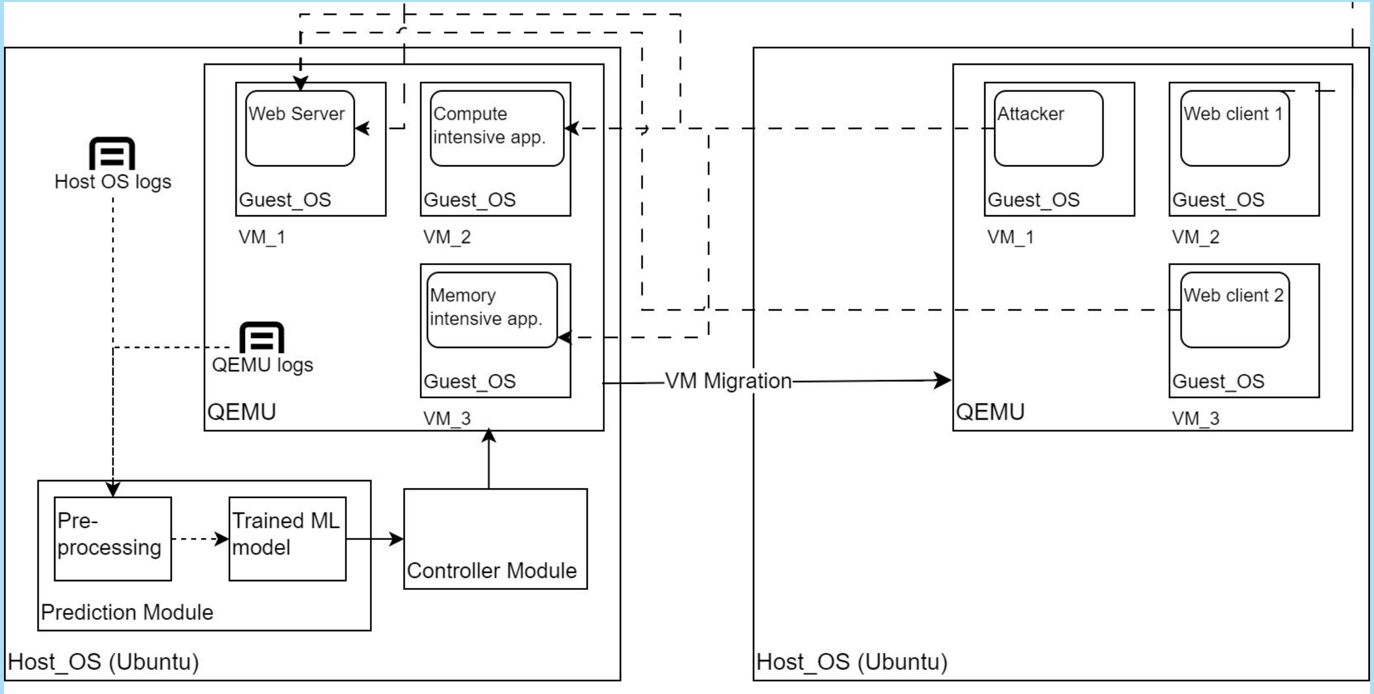
- Log data files:
  - syslog - /var/log/syslog
    - Generic system activity logs
  - bootlog - /var/log/boot.log
    - Booting related information and messages logged during system startup
  - kernel logs - /var/log/kern.log
    - Information logged by the kernel
  - QEMU logs- /var/log/VM*.log
    - Logs generated by QEMU and for each VM instance
  - Application logs- /var/log/*.log
      - Logs generated by each running application

Appendix

- Fault injection:
  - Stress-NG - Overloading CPU, Memory
  - Apache bench/siege - HTTP load testing
  - dmsetup - Inject HDD errors
  - scsi_debug - Linux HDD fault injection suite
  - mce-inject - Linux CPU fault injection suite
  - Chaos Mesh - Fault simulation and orchestrate fault scenarios
- Log collection and streaming:
  - R-syslog

Appendix

- High-level architecture of final system [Tentative]:



Appendix

- ML model evaluation metrics:
  - Accuracy - Correct predictions / All predictions
  - Precision - True positives / (True positives + False positives)
  - Recall - True positives / (True positives + False negatives)
  - F1 score - 2 * (Precision * Recall) / (Precision + Recall)
- Log collection and streaming:
  - R-syslog