Service Quality Awareness of Migration Workload

1st Huu Phuc Dinh - 1104950
Masters of High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
huu.dinh@stud.fra-uas.de

2nd Priya Singh - 1428461
Masters of High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
priya.singh@stud.fra-uas.de

3rd Rohan Tiwari - 1428131
Masters of High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
tiwari.rohan-brijesh@stud.fra-uas.de

4th Minh Trang Nguyen - 1306981
Masters of Allgemeine Informatik
Frankfurt UAS
Frankfurt, Germany
minh.nguyen5@stud.fra-uas.de

5th Ali Mohammad Nekoh - 1444966
Masters of High Integrity Systems
Frankfurt UAS
Frankfurt, Germany
ali.nekoh@stud.fra-uas.de

***Abstract*—This research addresses the complex challenge of forecasting the impact of virtual machine and container migration, particularly when relocating services to clients over the internet with a focus on energy efficiency. The central objective is to develop a predictive framework for real-time forecasting of service quality within virtual environments. Key tasks include investigating methods for measuring service quality, evaluating relevant tools in an "offline-first" context, and constructing a conceptual framework for integration. The significance lies in advancing our understanding of reliable service quality awareness, informing strategies for informed decision-making in workload migration and service deployment. By contributing to the evolving landscape of cloud computing and migration, this research aims to optimize service quality and resource utilization in virtual environments.**

***Index Terms*—Service Quality, Real-time Forecasting, Energy Efficiency, Workload Migration, Cloud Computing, Reliable Forecasting, Decision-making Strategies, Performance Metrics**

I. Introduction

In the era of virtualization and cloud computing, the migration of workloads, particularly virtual machines and containers, has become a cornerstone for optimizing operational efficiency within data center environments. This practice seeks to enhance resource utilization by strategically placing virtual workloads on host machines, mitigating idle or sub-optimal states. However, the complexity intensifies when forecasting the impact of service relocation, especially when migrating workloads to clients over the internet with an emphasis on energy efficiency. Addressing this intricate challenge requires the development of a scheme and mechanism capable of reliably forecasting the influence on applications running within virtual environments in real-time.

We have come up with a conceptual framework through this project to explore whether an effective system that can determine migration decisions can effectively exist. We want to explore the possibility of such a system that can be useful not only to software companies that have tight delivery schedules in place, but also to emphasis on the importance of service migration to a machine best suited for the users/client. A service-quality-aware system that monitors servers based on locations, locality, limitations and tries to find the best target machine for a service. For example, a gamer moving in a train requires his applications to be seamlessly available during the play hence enhancing user experience. Focusing on UX is the new trend which has been made possible through the cloud technologies. Yet there are solutions that can be created to maximise resource utilization, reduce latencies and increase the overall user experience.

The overarching research question driving this project is: How can we reliably forecast the influence of migration on the quality of service within virtual environments, particularly when considering the relocation of services to clients over the internet with an emphasis on energy efficiency? This question crystallizes the central challenge at hand, emphasizing the necessity for a predictive framework capable of providing real-time insights into the potential impact on service quality. Virtual machine migration, while contributing significantly to operational efficiency, presents challenges in understanding its influence, especially when considering the migration of workloads to clients over the internet. The dynamics of the internet environment, coupled with the imperative of energy efficiency, make this a particularly challenging endeavor [1], [2]. Hence, the need arises for a robust scheme and mechanism that can accurately forecast the impact of service relocation on applications running in virtual environments.

The primary objective of this research is to investigate methods for measuring service quality and to explore their application fields. This encompasses an exploration of tools dedicated to measuring service quality, emphasizing their application within an "offline-first" concept. The term "offline-first" refers to a paradigm where services are designed to function seamlessly, even in the absence of a continuous internet connection. The development of a conceptual framework to integrate service quality measurement in such an offline-first setting is pivotal, guiding the subsequent prototypical showcase implementation.

To comprehensively address the research question and ob-

jectives, the project encompasses several tasks. These include an in-depth exploration of methods to measure service quality and their diverse application fields. Furthermore, the research involves the identification and assessment of tools dedicated to service quality measurement, focusing on their applicability in an offline-first context. The culmination of these efforts involves the development of a conceptual framework, providing a structured approach to integrating service quality measurement within the offline-first setting. Finally, a prototypical showcase will be developed to demonstrate the practical implementation of the conceptual framework.

This research is significant in its potential to advance our understanding of how service quality awareness can be reliably measured and forecasted, particularly in the challenging context of migrating workloads to clients over the internet. The outcomes of this study will contribute valuable insights to the field, potentially informing the development of strategies and tools for enhancing decision-making processes related to workload migration and service deployment in virtual environments.

II. State of the Arts

Akoush et al.'s study focuses on predicting the performance of virtual machine migration, a critical aspect of cloud system operation. By exploring methodologies and models to forecast the impact of migrating virtual machines, the research provides insights into decision-making processes crucial for efficient cloud infrastructure management [1]. The authors highlight the significance of accurately predicting the performance of virtual machine migration to ensure optimal resource utilization and maintain service quality. They discuss various metrics and models employed to assess the potential impact on performance during migration. Understanding the implications of their findings is essential for our research, as it directly aligns with the core objective of enhancing service quality awareness during the relocation of virtual workloads.

Raghunath et al.'s study, published in the International Journal of Computer Applications, explores delay-tolerant and energy-reduced task allocation in the context of the Internet of Things (IoT) and cloud systems. The research proposes strategies for optimizing task allocation to reduce energy consumption, a theme closely related to our goal of energy-efficient workload migration [2].

Raghunath et al.'s study addresses the challenges of task allocation in IoT environments, where energy constraints are critical. By investigating strategies to tolerate delays and reduce energy consumption during task allocation, the work provides valuable insights into potential approaches for optimizing energy efficiency in the relocation of virtual workloads. This exploration complements our research by offering alternative perspectives on energy-efficient task allocation in cloud systems.

Presented at the IEEE International Interdisciplinary Humanitarian Conference, the study on "Containers Placement and Migration on Cloud System" by Oussama Smimite and Karim Afdel focuses on containers' placement and migration within cloud systems. The work is particularly relevant to our research, as it delves into the challenges and strategies associated with optimizing the placement and migration of containerized workloads [3]. The study likely explores the dynamic nature of containerized environments, addressing issues such as resource allocation, load balancing, and performance optimization. Understanding the findings of this work is crucial for our research, as it provides insights into the nuances of container migration within cloud ecosystems. This exploration aligns with our goal of enhancing decision support systems in cloud computing, especially concerning service quality awareness during workload relocation. El-Gendy et al's work explores the evolution of Quality of Service (QoS) on the internet and its support for soft real-time applications. The study discusses how QoS considerations have evolved over time, impacting the support for applications with soft real-time requirements. The exploration of QoS in this context is highly relevant to our research, which aims to improve decision support systems in cloud computing, particularly concerning service quality awareness [4]. In the study [4] delves into the historical evolution of QoS mechanisms on the internet, highlighting key developments and challenges. Understanding the evolution of QoS and its support for soft real-time applications provides context for our research, informing our understanding of the broader landscape and influencing our approach to service quality awareness in virtual environments.

Ranjan et al.'s work on QoS-driven server migration focuses on optimizing the migration of servers within internet data centers. The study likely explores strategies for ensuring Quality of Service during server migration, emphasizing the importance of maintaining service quality for applications running in data center environments [5].

The authors discuss the challenges associated with QoS-driven server migration and propose methodologies or models to address them. This work is particularly relevant to our research as it aligns with our overarching goal of enhancing service quality awareness in virtual environments. By understanding the considerations and approaches presented in this study, we gain valuable insights into the complexities of decision-making processes during server migration in data center settings.

In summary, these related works contribute significantly to the understanding of various aspects related to virtual machine migration, energy-efficient task allocation, and Quality of Service considerations in cloud computing. Each study offers valuable insights, methodologies, and challenges that enrich the conceptual foundation of our proposed research on service quality awareness in virtual environments. Understanding these works is essential for informing our approach and ensuring that our research builds upon and contributes to the existing body of knowledge in this domain.

III. Background Knowledge

In this project, the work of migrating a virtual machine from one host to another is implemented. Therefore, virtual machine is briefly summarised, followed by different migration methods

as well as their performance and advantages/disadvantages on various applications.

A. Virtual Machines

A virtual machine is a simulated version of a physical computer, commonly known as a guest, while the running physical machine is a host. Virtualization enables the possibility of operating multiple virtual machines, each has its own operating system (OS) and application on a single physical host [6]. Because of isolating from the rest of the system, a virtual machine can execute in particular hazardous tasks to the host, such as accessing data infected with viruses or testing operating systems. [7]

B. Hypervisors

This section discusses two main hypervisors, types-1 and type-2, applied in various migration schemes. Type-1 hypervisors, more efficient, directly access hardware resources, while type-2 hypervisors run as processes within a conventional OS environment, relying on host operating systems to access hardware resources [8]. Examples include Xen Hypervisor Fig. 2, a type-1 hypervisor from Cambridge University, a type-1 hypervisor converting Linux into a hypervisor with hardware-based virtualization extensions [9] [10].

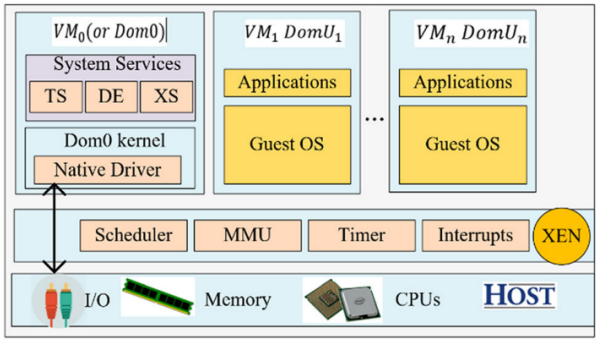


Fig. 1. Xen Hypervisor [11]

C. Live Migration

Live Migration, also known as migration, involving a process of transferring an operating virtual machine (VM) or application from one physical machine to another without interrupting the connection with the client or application [12]. The virtual machine's CPU context, memory, storage, network connections (IP address, MAC address, etc.), and virtual I/O devices are moved from the initial guest machine to the designated destination. The two well-known methods for transferring the memory state of a virtual machine from the source to the destination are pre-copy migration and post-copy migration, which are discussed in more details.

D. Steps to Migrate from Host A to Host B

An overall migration technique is shown as follow: [13]

1) Setup: setup host B so that resources (CPU, memory, disk) are reserved for the virtual machine.
2) Push phase: push the memory of VM from host A to host B.
3) Stop and copy: Host A's VM is halted, CPU context and memories are copied.
4) Pull phase: the VM on host B is initiated, then retrieve any additional memory from A.
5) Clean up: remove states from host A, the migration process is now completed.

E. Live-Migration Metrics

The total migration time is the time for step 2, 3 and 4, whereas the service downtime is the time for step 3. To determine how good the migration technique is, total migration time and service downtime are highly considered [13]. Various factors, including application degrade time, resume time, link speed, and power consumption, are considered in different VM migration solutions.

F. Migration Classification

VM migration schemes are classified based on network types, migration granularity, security capability, algorithm design, migration scope, and more. Classification includes LAN, MAN, WAN, and Internet networks, secure and insecure migration, heuristic and meta-heuristic approaches, intra-cloud and inter-cloud migration, and homogeneous and heterogeneous VMs.

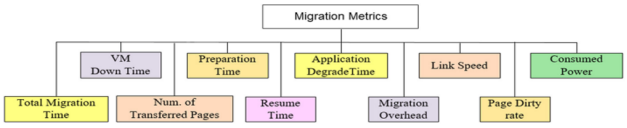


Fig. 2. Performance metric considered in the proposed migration schemes [11]

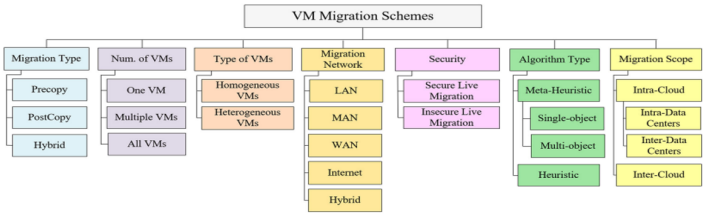


Fig. 3. Properties of the VM migration schemes [11]

G. Migration Objectives

The illustration in Figure 4 outlines the desired outcomes achievable through predictive VM migrations in cloud Data Centers (DCs) [14]. These objectives include:

- Reducing the number of Physical Machines (PMs): Addressing the cold-spot problem by consolidating Virtual Machine (VMs), placing some PMs in a sleep state, and consequently decreasing the total number of PMs. This not only improves energy efficiency, leading to decreased energy consumption for Cloud Service Providers (CSPs)

Fig. 4. Objective of the VM migration schemes [11]

but also enhances resource management for handling more user requests and achieving higher profits.

- Reducing the number of VM migrations: Accurate prediction of future workload helps prevent unnecessary and premature VM migrations, optimizing the migration process.
- Improving Cloud Service Providers' (CSP's) profit: By enhancing resource management and making VM migrations more efficient, the overall profit of CSPs increases.
- Decreasing Cloud Service Providers (DC's) network traffic: Efficient VM migrations contribute to a reduction in retransmitted pages, minimizing DC network traffic, improving application performance, and preventing Service Level Agreement Violations (SLAV).
- Preventing Cloud Service Providers SLAV: By reducing the downtime of migrating VMs, the predictive VM migration schemes effectively prevent SLAV penalties.
- Improving Cloud Service Providers (QoS): Efficient migration schemes with low downtime enhance the Quality of Service (QoS) for cloud systems.
- Reducing energy consumption in DC network: Accurate forecasting of application workload allows the identification of future accessed pages, reducing the number of transmitted pages and consequently lowering the energy consumption of the DC network.

H. Various Migration Techniques

- Pure stop-and-copy: Firstly, the VM stopped, and all the state are transferred to the target host. When transferring is finished, the VM restarts. This technique's disadvantage is the required amount of downtime is excessive to be seemed as live migration.
- Pre-copy: the majority of states are transferred in the push phase, with memories are being pushed as much as possible. Stop-and-copy phase followed concisely. VM's states are up-to-date at the host A during migration.
- Post-copy: the VM is temporarily stopped at the host A. While suspended, a minimal subset of the VM's execution state such as CPU state, registers, and non-pageable memory is transferred to the target [15]. The VM's states are divided across host A and host B.
- Hybrid: a combination of pre-copy and post-copy technique. States are pushes followed by stop-and-copy, then the on-demand pulling of states.

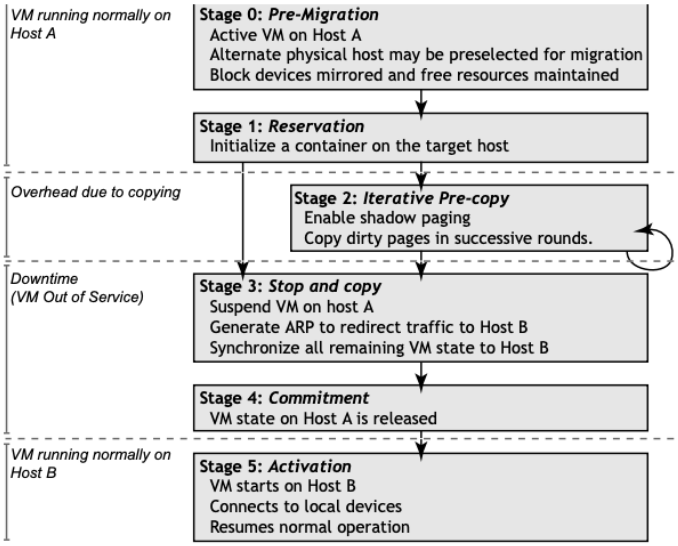


Fig. 5. Pre-copy Migration [16]

1) Pre-copy Live Migration: A pre-copy memory migration from host A to host B can be divided into 6 stages as shown in Figure 5 [16]:

- Stage 0 - Pre-Migration. Initiating the process from an operational VM on host A. The aims is to enhance the efficiency of future migrations by preselecting a target host in order to ensure the necessary resources.
- Stage 1 - Reservation: A request is generated to move the OS from host A to host B. Resources validation is checked on host B, and a suitable size is reserved. If the reserved resources is failed in this stage, the VM could continue running on A without any impact.
- Stage 2 - Iterative Pre-copy: During initial iteration, all pages are transferred from host A to host B. Subsequent iterations only copy modified or dirty pages.
- Stage 3 - Stop and Copy: the running OS on host A is temporarily suspended, and its network traffic is redirected to host B. Following this, CPU state and any remaining inconsistent memory pages are transferred. This stage concludes with a consistently suspended copy of the VM on both hosts A and B. Host A's copy remains primary and is resumed in case of failures.

- Stage 4 - Commitment: Host B notifies host A of the successful reception of an OS image. Host A acknowledges this message, marking the migration transaction. Host A can then discard the original VM, and host B becomes the primary host.
- Stage 5 - Activation: the VM on host B is activated. Post-migration code executes to reattach device drivers to the new machine, followed by relocated IP address.

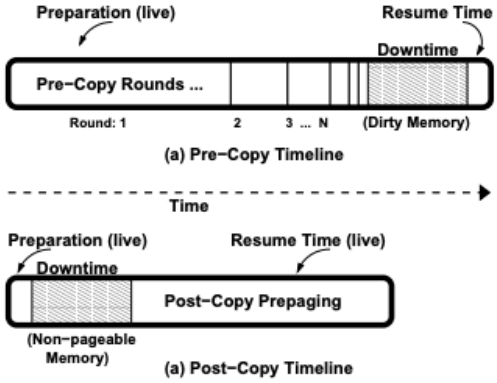


Fig. 6. Post-copy Migration [15]

2) Post-copy Live Migration: The fundamental approach for this technique is as follows [15]:

- Step 1: The technique initially pauses the migrating VM at the host A.
- Step 2: It transfers essential processor state to host B, then resumes the VM.
- Step 3: The retrieval of memory pages are fetched over the network from host A.

3) Comparing Pre-Copy and Post-Copy Migration: Post-copy live migration presents advantages over pre-copy live migration [13] [15] :

- Low Initial Downtime: Significantly reduced downtime, as only CPU and device state—amounting to several kilobytes—need to be migrated while the VM is in a stopped state. Post-copy migration initiate by transferring minimal necessary states to the destination host B, hence the VM operates on host B almost immediately.
- Quick Migration Start: Short and predictable total migration time, as pages are not re-dirtied on the source host during the live migration process.
- Lower total migration time because of fewer page transfers and lesser disruption to application.
- Post-copy technique suitables for memory-intensive application with large number of pages.

However, pre-copy migration has advantages over post-copy migration in terms of [13] [15]:

- If the target machine fails during migration, pre-copy simply aborts the migration, then restart with another target. Whereas post-copy cannot recover the application data without the execution of some form of replication.
- This technique has much lesser downtime.
- Pre-copy technique suitables for interaction application such as web application.

IV. Methods and Materials

To investigate the service quality awareness of virtual machine migration in the cloud, a comprehensive experimental setup was devised. Two Windows machine were chosen which acted as a source and target machine for the experimental setup. Both machine were equipped with Windows 11 Home, Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s), 8GB RAM and 25 GB Virtual memory. Both the machines were connected over a Local Area Network (LAN) for this experiment. Both the machines run on the Windows OS.

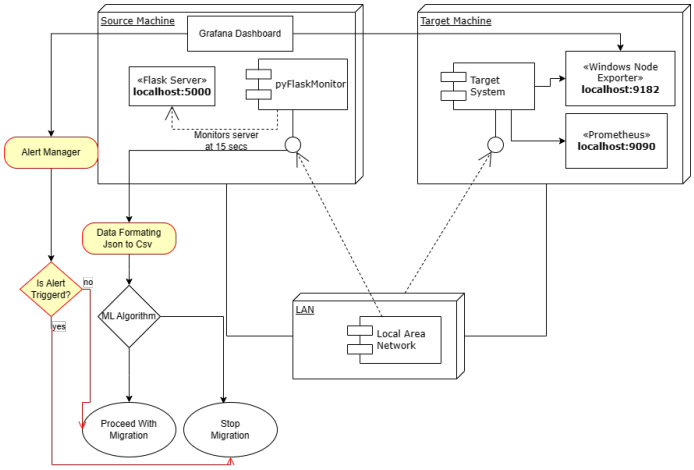


Fig. 7. System Architecture Design

As shown in Figure 7 the target machine consists of a Prometheus server, along with a Windows node exporter running on it. The source machine has a flask server running which is continuously monitoring the target machine over the LAN connection. The source also has a Grafana instance running which has alert rules configured, which are triggered when one of the performance metrics on the target system exceeds during or before the migration process. The source machine having obtained the metrics data from the target machine, compiles the data to a csv format and runs it through a Machine Learning algorithm. This algorithm is responsible for predicting whether the migration should take place or not.

A. Metric Selection and Pre-processing

The success of any virtual machine migration project hinges on the careful selection of performance metrics that offer comprehensive insights into the system's health. In our investigation of service quality awareness during virtual machine migration in the cloud, we meticulously chose a set of key metrics to monitor. These metrics spanned various facets of the system, encompassing both general performance indicators and Windows-specific parameters. The chosen metrics aimed to provide a holistic view of the target machine's condition,

ensuring that the migration decision-making process was well-informed and nuanced.

CPU utilization, memory availability, and network latency—were meticulously selected for analysis. These metrics undergo preprocessing, including normalization and feature scaling, ensuring standardized inputs for the logistic regression model. Prometheus efficiently collects and stores this preprocessed metric data, ensuring compatibility and readiness for machine learning inputs.

B. Metrics Captured

1) CPU Usage: The CPU usage metric was calculated using the query

```
'100-(avg(irate(windows_cpu_time_total{
    mode="idle"}[5m]))*100)'
```

This query measures the rate of change in CPU idle time over a 1-minute interval and then subtracts this value from 100 to obtain the CPU usage percentage. The importance of this metric lies in its ability to provide insights into the computational load on the system's CPU. By understanding how much time the CPU spends in an idle state, one can gauge the degree of resource utilization. High CPU usage may indicate that the system is under stress, potentially signaling the need for resource optimization or infrastructure scaling.

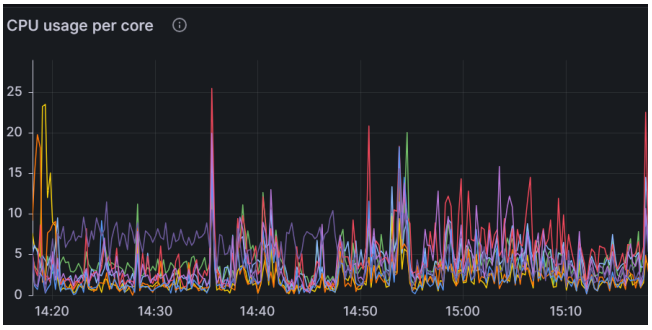


Fig. 8. Cpu Usage per Core

2) Memory Consumption

```
'windows_os_physical_memory_free_bytes',
'windows_os_virtual_memory_free_bytes'
```

was employed to calculate memory consumption. It calculates the used memory by subtracting the free memory from the total available memory. Monitoring memory consumption is crucial for detecting patterns in RAM utilization and identifying potential memory-related issues. A high memory consumption could indicate that the system is running out of available RAM, which might lead to performance degradation or, in extreme cases, system instability. This metric is vital for optimizing resource allocation and ensuring efficient use of memory resources.

3) Network Activity Queries: The query

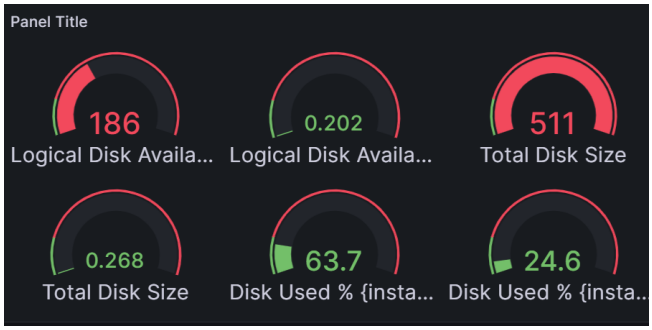


Fig. 9. Memory Usage Percentage

```
sum(rate(windows_net_bytes_total{instance=
    "localhost:9182"}[30s]))
```

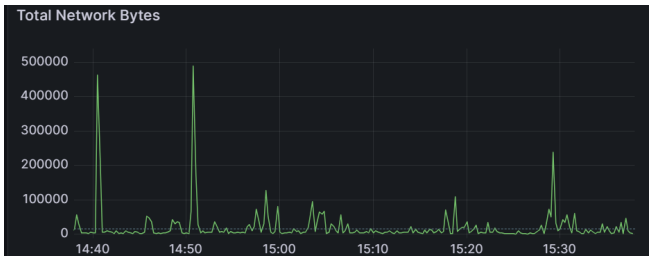


Fig. 10. Total Network Bytes Consumed

are instrumental in measuring incoming and outgoing data rates on the network interface "eth0." Understanding network activity is essential for assessing the load on the network and identifying periods of heightened or diminished activity. High network activity may suggest increased demand or potential issues, while low activity might indicate periods of reduced workload. These metrics provide valuable insights into network performance and are crucial for optimizing network resources.

4) Disk I/O:
The queries

```
'rate(windows_logical_disk_reads_total{
    instance="localhost:9182",volume="C
    :"}[5m])'
```

```
'rate(
    windows_logical_disk_read_seconds_total
    {instance="localhost:9182",volume="C
    :"}[5m])'
```

focus on measuring the rate of change in completed disk reads and writes over a 1-minute interval. Disk I/O metrics are essential for evaluating storage system performance. Understanding the read and write operations on disks helps in identifying potential bottlenecks and optimizing storage resources. High rates of disk reads and writes may indicate increased demand or potential issues with storage performance, necessitating proactive measures to ensure optimal system efficiency.

Fig. 11. Hard Disk Write Operations

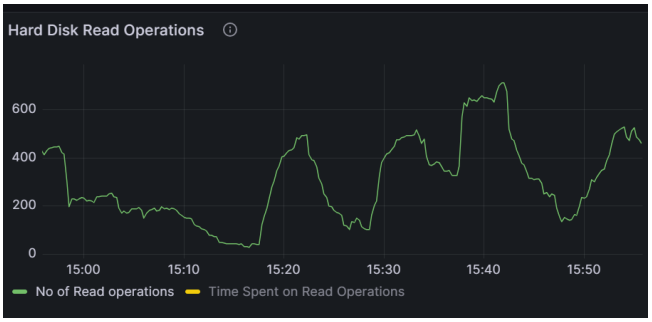


Fig. 12. Hard Disk Read Operations

Each of these queries was carefully chosen to capture specific aspects of system performance, and their calculations provide actionable insights for maintaining and optimizing the overall health and efficiency of the monitored system.

C. Prometheus and Feature Set-up for Continuous Monitoring System

To monitor the target machine before the service migration, a machine learning (ML) model was trained to return the result whether we should start the migration process based on the collected metrics. To retrieve these metrics, the Prometheus tool is used together with the Prometheus Node Exporter and Metrics Visualisation tool Grafana.

However, these tools are kept running continuously even while the service is being migrated. The aim of the continuous monitoring is to stop the migration process if the metrics fail or fall below a certain threshold. For example, if the connection is down or the target machine's file system is not working properly.

Here we give a brief overview and working of the tools used, followed by the configuration of the entire monitoring system.

Figure 13 illustrates the structure of our monitoring system. First, the Prometheus Node Exporter is run on the host to retrieve hardware-based metrics. The metrics are then collected by Prometheus. As well as using Prometheus to monitor our nodes, there are a number of features that Prometheus also provides to facilitate the management work. In our case, we use Alert Manager and Grafana. In the later chapters, each tool involved in the monitoring system will be described in more detail.

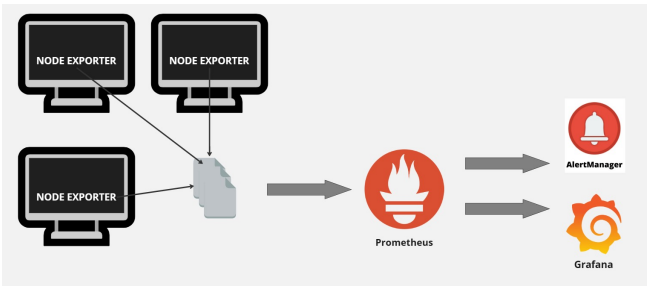


Fig. 13. Prometheus monitoring system

1) Prometheus: There are quite a lot of options for monitoring tools currently on the market, such as *Sematext*, *SolarWinds Server*, *Dynatrace*, *Datadog*, *Nagios* and so on [17]. Among other tools, we decided to use *Prometheus*. The reasons for setting up a monitoring system with *Prometheus* are that it is an open source monitoring tool. Together with *Grafana*, they are the well-known combination used for server tracking [17].

Prometheus monitoring tool is commonly used due to its following key features [18] [19]:

- **Time-series database:** The metrics are stored in a Prometheus-customised time-series database, making it convenient to query and analyse the data. The stored data is timestamped, so it is possible to graph its change to get a visual overview of how the data changes over time.
- **PromQL query language:** Prometheus uses PromQL, which provides various operations to extract meaningful information from the collected metrics.
- **Pull-based system:** Prometheus actively pulls metrics from the node it is monitoring. The fetching process is done regularly after a time interval and this brings the advantage of timely handling of any negative changes.
- **Supported components:** Various features such as Grafana and Alert Manager are available to help monitor the node. They are also simple to set up together.
- **Alerting:** The user can define rules for the metrics, for example, to control when the CPU utilisation is higher than 95%. Alert rule configuration is described in a later chapter.

Prometheus does not monitor the server itself, but needs the help of another module to collect the metrics, and this is where Node Exporter comes in.

2) Prometheus Node Exporter: Our monitoring system is designed for Linux operating system hosts, so *Node Exporter* is used to collect system information of the machine it is running on [20]. For the Windows operating system, the *WMI Exporter* is used instead. The metrics are in the format that is utilized by the Prometheus [21].

With *Node Exporter* as the low-level system information collector, *Prometheus*' role is to store and display the metrics. This separation allows scalability. One *Prometheus* server can collect and analyse metrics from multiple nodes [18].

3) Prometheus and Node Exporter setup: *Node Exporter* does not require any special configuration. For the Linux operating system, all we need to do is download the *Node Exporter* zip file to the machine we want to monitor, unpack it, run it and we are ready to go. By default, *Node Exporter* runs and updates the host machine's metrics continuously [22].

As mentioned earlier, *Prometheus* will actively pull the metrics from the *Node Exporter* after a time interval. And *Prometheus* can capture multiple nodes at the same time. To make all this work, we need to register these nodes in Prometheus [22].

```
# Global configuration
global: [...]
# Alertmanager configuration
alerting: [...]
rule_files: [...]
# A scrape configuration containing exactly one
    endpoint to scrape:
scrape_configs:
  # The Prometheus itself.
  - job_name: "prometheus"
    scrape_interval: 5s
    static_configs:
      - targets: ["localhost:9090"]
  # Prometheus Node Exporter
  - job_name: "node"
    scrape_interval: 5s
    static_configs:
      - targets: ["localhost:9100"]
```

Listing 1. Prometheus Server Configuration

In Listing 1, there is one job that captures the Prometheus itself and the other is the Node Exporter. The default port of Prometheus is 9090 and the Node Exporter exposes metrics on port 9100. Both jobs update every five seconds.

4) Prometheus Alert Manager: After all of the steps above, we continue extend our monitoring system with an alert manager system. An alert mechanism has been configured to send an email if the parameter exceeds the limit. We use Prometheus Alert Manager for alerting but Grafana provides also the same mechanism and can be used instead.

Prometheus provides alert feature, which contains a list of rules, when the metrics exceed the limit value, the rule is triggered. The status of rules can be controlled using Prometheus UI. The usage of Alert Manager gives us more flexibility since it provides integration to various communication platforms: Slack, Microsoft Teams, Google Mail, OpsGenie, PageDuty [21].

5) Alert Rules: We have configured a list of rules to monitor the target machine. There are some parameters that are critical to control and handle in time when monitoring a server in general. These are CPU, disk and memory usage. For each of these parameters, I defined two rules with different alert flags, one with a warning flag and the other with a critical flag. The list of rules has been created with reference to the source Awesome Prometheus Alerts [23]. The limit for these parameters is set as follow:

- **Disk usage:** Disk space indicates the physical space used to store data [24]. In a service migration, the available hard disk capacity must be at least as large as the source machine in order to hold all the application data.
  - Rule\Limits
    - Warning: If the disk is possessed more than 80%
    - Critical: If CPU is possessed more than 90%
- **CPU usage:** CPU usage shows how much of the processor is being used to process tasks [25]. When the CPU is at full capacity, no other tasks can be processed properly, resulting in delayed response [26]. Consider the pre-copy migration type, the source host will simultaneously handle the migration process and the incoming request. If the migration process takes up most of the CPU space, the application may be adversely affected. Active and continuous monitoring of CPU availability is therefore important.
  - Rule\Limits
    - Warning: If the CPU is utilized more than 75%
    - Critical: If the CPU is utilized more than 80%
- **Memory usage:** Memory usage or RAM (Random Access Memory) usage is the percentage of temporary memory used to process tasks. As with high CPU usage, high RAM usage will also lead to processing problems.
  - Rule\Limits
    - Warning: If the memory is utilized more than 75%
    - Critical: If the CPU is utilized more than 80%

In addition to the above parameters, based on the purpose of monitoring the health of the target host, our list of rules also includes setting limits to check the state of the network, read and copy speed of data, and so on. These rules are as follows:

- **Server down:** The purpose of the server down rule is to check the state of both the source and target machines. The migration process cannot be performed if either of the two is in idle mode.
  - Rule\Limits
    - Critical: If server is down for more than one minute.
- **Host File System Error:** Even if the service migration is performed in a pre-migration or post-migration approach, the data needs to be reallocated from the source host to the target host. Therefore, it is critical to ensure that the file system on both hosts is working properly so that there are no issues with our data.
  - Rule\Limits
    - Warning: If file system does not work.
- **Disk Read Latency:** This metric returns the time taken by the node to perform a read operation [27].
  - Rule\Limits
    - Warning: If read operations take more than 100 milliseconds.
- **Disk Write Latency:** This metric returns the time taken by the node to perform a read operation [27]. If the node

takes longer than usual to read and write, this will affect the migration time.
  - Rule\Limits
    * Warning: If the write operations take more than 100 milliseconds
- **Network Transmit Errors:** During application migration, data is transferred between the source and target computers. It could be a bad sign if the transmission errors occur frequently.
  - Rule\Limits
    * Warning: Report the number of transmission errors in the last two minutes.
- **Network Receive Errors:** As with the transmission process, the reception process must also be controlled and the error should be dealt with in time to ensure the quality of the application after migration.
  - Rule\Limits
    * Warning: Report the number of reception errors in the last two minutes.
- **Prometheus Alert Manager Missing:** In addition to the Node Exporter for node monitoring, the Alert Manager should also be assigned to Prometheus so that we can be informed if the Alert Manager is working properly, as updating the Alert Manager may cause problems.
  - Rule\Limits
    * Warning: If the Alert Manager job is inactive.
- **Prometheus Alert Manager Notification Failing:** Alert Manager has a useful feature that allows us to send alerts to various communication platforms to keep us up to date. Therefore, we would like to make sure that this functionality is functioning properly.
  - Rule\Limits
    * Warning: If the Alert Manager has failed to send the alert email.

The Listing 2 below is a disk usage rule configuration, it gives an example of what a rule configuration might look like.

```
- name: DiskUsage90
  rules:
    - alert: 'Low Data Disk Space'
      expr: ceil(((node_filesystem_size_bytes{
          mountpoint!="/boot"} -
          node_filesystem_free_bytes{mountpoint!="/
          boot"}) / node_filesystem_size_bytes{
          mountpoint!="/boot"} * 100)) > 90
      labels:
        severity: critical
      annotations:
        title: "Disk Usage"
        description: 'Partition: {{$labels.
            mountpoint}}'
        summary: "Disk usage is `{{humanize $value
            }}%`"
        host: "{{$labels.instance}}"
```

Listing 2. Alert rule for disk usage

a) Alert Manager Configuration: For the alert rules we have two different alert flags: warning and critical. Having two different alert levels is not only useful when we control the server using the Prometheus UI, but also when we set up the Alert Manager.

```
route:
  group_by: ['alertname']
  group_wait: 30s
  group_interval: 10s
  repeat_interval: 1h
  receiver: 'email-notifications'

receivers:
- name: 'email-notifications'
  email_configs:
  - to: 'minhtrang190199@gmail.com'
    from: 'alertmanagertestprometheus@gmail.com'
    smarthost: 'smtp.gmail.com:587'
    auth_username: 'alertmanagertestprometheus@gmail
        .com'
    auth_password: ``<alert manager pwd>\''
    send_resolved: true
    require_tls: true
  match:
    severity: critical
```

Listing 3. Alert rule for disk usage

The Listing 3 is the Alert Manager configuration that allows an email to be sent when a rule with a critical flag is triggered. The alert email is resent after one hour.

After configuring Alert Manager, you may want to check that everything is set up correctly and that Alert Manager is working. The status of the Alert Manager can be checked by running the command 4 if the Alert Manager has been run as a service:

```
$ systemctl status alertmanager
```

Listing 4. Alert rule for disk usage

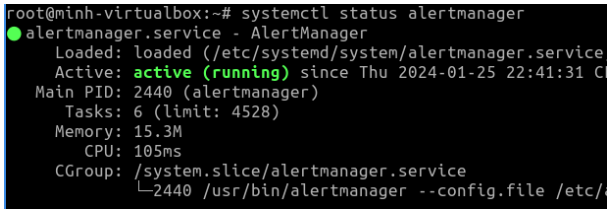


Fig. 14. Alert Manager Status

The figure 14 shows the status of the Alert Manager after executing command line 4.

There is another way to get informed about the Alert Manager that is going to the Prometheus UI. If the Alert Manager is working, we will see the alert manager node with the status "UP" (as depicted in figure 15) and the defined rules are listed under Alert tab (as depicted in figure 16).

D. Data Collection and Storage

1) Data from Prometheus Server: The target machine is equipped with a Prometheus server, used for collecting and querying performance metrics. This server was running

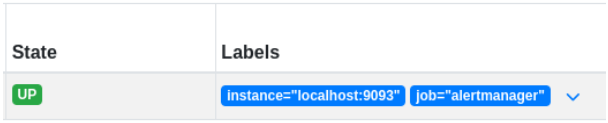


Fig. 15. Alert Manager Status in Prometheus UI

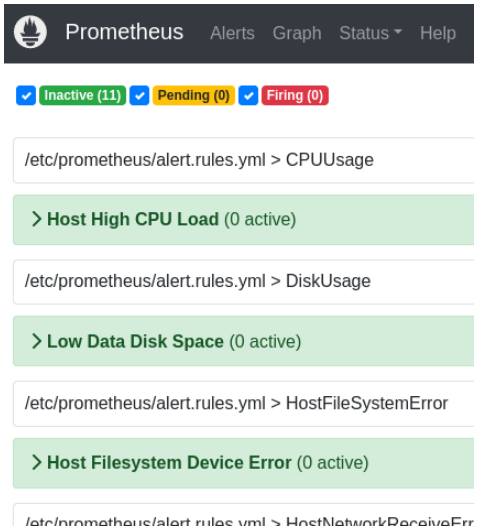


Fig. 16. Rules list in Prometheus UI

on 'http://localhost:9090' which was exposed to the source machine over a LAN. By continuously collecting performance metrics, including CPU usage, memory utilization, and network latency, it provided a comprehensive and real-time snapshot of the target system's health. This rich dataset was fundamental in understanding the baseline performance and detecting any anomalies that might affect the migration process.

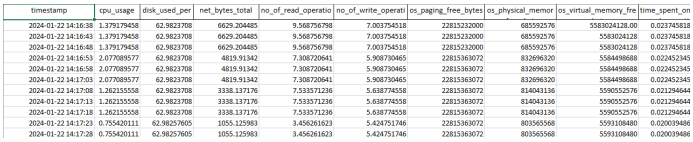
A Python script was crafted to interface with Prometheus APIs, ensuring a seamless and automated data retrieval process. The script, designed with periodicity in mind, fetched metrics data at 15-second intervals, allowing for a granular examination of performance fluctuations. Exception-handling mechanisms were incorporated to enhance the script's robustness, ensuring uninterrupted data retrieval even in the face of transient network issues.

2) Windows Node Exporter: The inclusion of the Windows Node Exporter in our experimental setup significantly enriched the monitoring capabilities by providing a detailed and specialized insight into the performance metrics specific to Windows operating systems. In the context of virtual machine migration, understanding the intricacies of a Windows environment is crucial for making informed decisions and ensuring a seamless transition. Some important metrics from the windows system obtained were - currently running services, Memory Utilization, RAM Usage, Network Bandwidth consumed, and process Resource Consumption.

3) Flask Server on Source Machine: The source machine hosts a Flask server built on python 3.12, running on the instance 'http://localhost:5001' which is responsible for initiating and overseeing the data collection process. It establishes a secure LAN connection to the target machine and retrieves performance metrics at regular intervals of every 10 seconds. Thus, facilitating a dynamic and real-time monitoring approach.

E. Data Formatting and Visualization

1) JSON to CSV Conversion: One of the challenges of working with the metrics data exported by Prometheus was that it was a time series data in JSON format, distributed across a different set of files. Hence, a json-to-csv exporter was created which was responsible for extracting relevant information from the json files and writing it to a consolidated csv file. Fig 2. shows an image of the data file generated. This helped in the visualization of the time series data and also in understanding if there were any irregularities with the data or not. The consolidated csv file generated was perfectly



timestamp	cpu_usage	disk_used_per	net_bytes_total	no_of_read_operatio	no_of_write_operati	os_paging_free_bytes	os_physical_memor	os_virtual_memory_fre	time_spent_on
2024-01-22 14:16:38	1.379179458	62.9823708	6629.204485	9.568756798	7.003754518	22815232000	685592576	5583024128.00	0.023745818
2024-01-22 14:16:43	1.379179458	62.9823708	6629.204485	9.568756798	7.003754518	22815232000	685592576	5583024128	0.023745818
2024-01-22 14:16:48	1.379179458	62.9823708	6629.204485	9.568756798	7.003754518	22815232000	685592576	5583024128	0.023745818
2024-01-22 14:16:53	2.077089577	62.9823708	4819.91342	7.308720641	5.908730465	22815363072	832696320	5584498688	0.022452345
2024-01-22 14:16:58	2.077089577	62.9823708	4819.91342	7.308720641	5.908730465	22815363072	832696320	5584498688	0.022452345
2024-01-22 14:17:03	2.077089577	62.9823708	4819.91342	7.308720641	5.908730465	22815363072	832696320	5584498688	0.022452345
2024-01-22 14:17:08	1.262155558	62.9823708	3338.137176	7.533571236	5.638774558	22815363072	814043136	5590552576	0.021294644
2024-01-22 14:17:13	1.262155558	62.9823708	3338.137176	7.533571236	5.638774558	22815363072	814043136	5590552576	0.021294644
2024-01-22 14:17:18	1.262155558	62.9823708	3338.137176	7.533571236	5.638774558	22815363072	814043136	5590552576	0.021294644
2024-01-22 14:17:23	0.755420111	62.98257605	1055.125983	3.456261623	5.424751746	22815363072	803565568	5593108480	0.020039486
2024-01-22 14:17:28	0.755420111	62.98257605	1055.125983	3.456261623	5.424751746	22815363072	803565568	5593108480	0.020039486

Fig. 17. metrics data extracted from the target machine

consistent with the kind of information that was obtained from the target machine. To handle missing data imputation was used, where the null values were replaced by 0. Although this was a very rare case which was only observed for the network bandwidth utilization metric which produced no values when the network connection was lost.

2) Grafana Dashboard: The Grafana dashboard on the source machine provided a visually intuitive interface to monitor the performance metrics. Customizable visualizations allowed us to track key indicators, enabling a quick assessment of the target system's health. With only a few queries - namely PROMQL, the Grafana dashboard was created for real-time monitoring of the target machine. The seamless integration between Prometheus and Grafana ensured efficient data visualization, allowing for a holistic view of system metrics. We have been able to retrieve, query, and analyze the metrics. The data is presented as plain text and in some cases, this way of presenting data is difficult to perceive or takes longer to perceive.

It is quite common that the Grafana tool is used together with Prometheus because with Grafana we can display the metrics in a chart. In our use case, we use Grafana mainly to visualize the metrics and to extract the metrics that will later be used to train the ML model. In addition, Grafana also provides various functionalities, such as data queries, alerts, metrics exploration, logs, and so on. Grafana treats Prometheus as a data source, as any other source can be named here, such as MySQL, Graphite, InfluxDB, etc [28].

Grafana gives us the freedom to design the dashboard. A dashboard can contain histograms, different types of maps (heat maps, geomaps) and charts. In addition, there are also pre-built dashboards that are available so that we can include them in our use case and not have to build ours from scratch [28].

F. Machine Learning Model Selection

A crucial aspect of the methodology involves the selection of an appropriate machine learning algorithm to aid in decision-making for live migration based on monitored metrics. Considering the complexity of the dataset, a simple target variable was chosen named as "migration status". Based on the this, the following three models were chosen and trained :

1) Logistic Regression: Logistic Regression(LR) is a well-established algorithm for binary classification tasks, making it suitable for predicting whether VM migration is likely or not. The model's linear nature facilitates the interpretation of feature importance and the impact of each variable on the migration outcome.

2) Neural Network: Neural Networks possess a remarkable capability for learning intricate patterns and dependencies within datasets, rendering them well-suited for complex VM migration prediction tasks. Through the utilization of multiple hidden layers and non-linear activation functions, Neural Networks can effectively learn hierarchical representations of data, enabling them to capture nuanced relationships that may exist within the dataset. Moreover, Neural Network models demonstrate robust performance even with non-linear datasets, showcasing their adaptability and versatility in handling diverse data structures and complexities.

G. Model Training and Testing

The data was split into 80:20 ratios. Where 80% was used for training purposes and 20% was used for testing purposes.

1) Training Logistic Regression: Utilizing historical metric datasets collected via Prometheus and Grafana, the models undergo extensive training. This process involves feeding the historical metric data into the model, allowing it to learn and adapt to patterns and variations. For training the model scikit-learn package in python was used. An instance of Logistic Regression model was used for this purpose where the training data was fit into the model.

2) Training Neural Network: For training the neural network model, scikit-learn was used along with python. A sequential model was created, with first layer consisting of 64 neurons with 'ReLu' activation function enabled on them. The second hidden layer consisted of 32 neurons with a 'Sigmoid' activation function. Since, it is a binary classification output layer had only 1 neuron with 'Sigmoid' function. The '.fit()' function was used to train the model with initial configuration of epochs being 10 and batch size being 32.

3) Testing Models: After the training phase, '.predict()' method was used to predict the output from the trained model on the testing data split. Accuracy, precision, recall, F1 score were the important metrics considered for evaluating the model outcome.

Subsequently, the trained model was seamlessly integrated into the system, enabling real-time assessment of new metric data. This integration facilitates timely predictions regarding device readiness for service migration based on monitored metrics.

V. Results

A. Alert Manager

When an alert is triggered, it is first highlighted in red in the Prometheus UI under the Alert tab. The Alert Manager will then record this triggered alert and if it is a critical rule, an email will be sent.

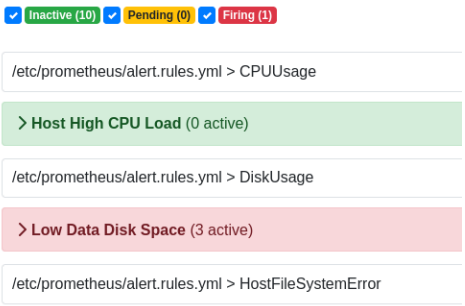


Fig. 18. Prometheus UI - Alert fires

As you can see in Figure 18, the Low Data Disk Space alert rule is highlighted in red, which means that disk space is running low. I set the threshold at 50% to force the alert to fire, but in practice it can be as high as 90%.

Because this alert is critical, Alert Manager will send an email. Figure 19 is the email sent by Alert Manager. The email will include the alert label, the instance and the amount of disk space used.

Alert Manager lets us know not only when an alert is triggered, but also when the problem has been resolved. Figure 20 shows the resolution notification.

B. Machine Learning Model

The project successfully gathered and visualized metric data from monitored devices using Prometheus and Grafana. A comprehensive array of metrics, including CPU utilization, memory consumption, and network activity, was collected and stored efficiently. Grafana's visually rich and interactive dashboards provided a clear depiction of these metrics, facilitating in-depth analysis and monitoring.

Visual representations in Grafana depicted the model's predictions regarding device suitability for live migration. Through Grafana's visualization capabilities, the binary predictions were graphically presented, allowing for intuitive interpretation and decision-making. Graphs and charts showcased the correlation between key metrics and the model's predictions, aiding in understanding the impact of individual metrics on migration feasibility.

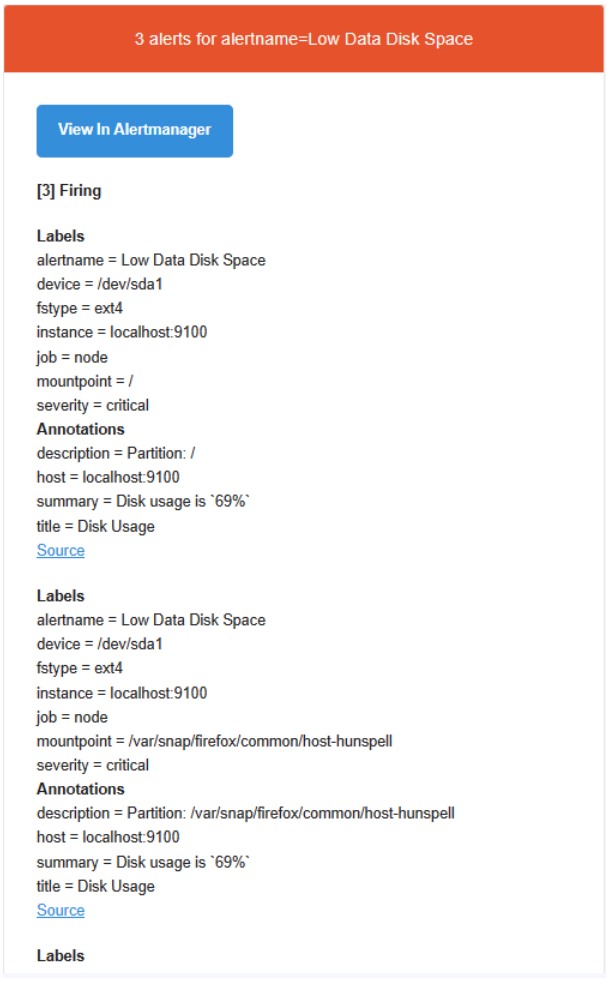


Fig. 19. Alert Email

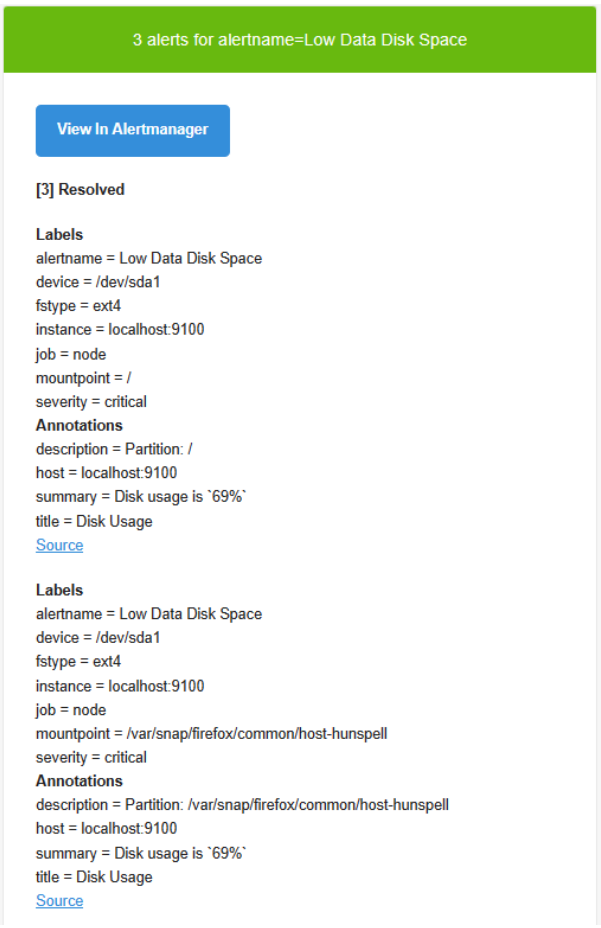


Fig. 20. Alert Resolution Email

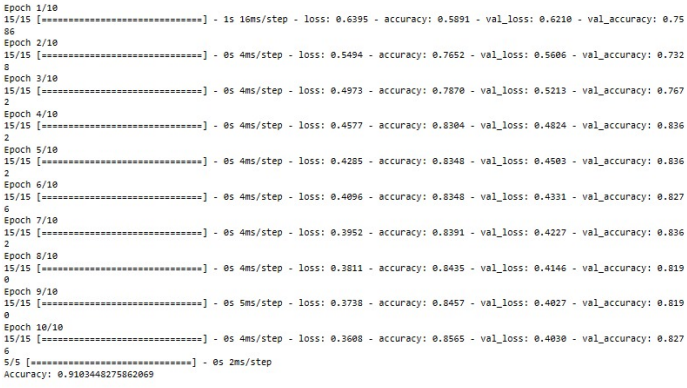


Fig. 21. NN outcome

The alerting system configured in Grafana effectively triggered notifications upon breaching predefined thresholds. Instances of CPU spikes beyond set limits and network latency anomalies were promptly identified through the configured alert rules. This proactive approach ensured timely responses to potential issues, mitigating risks and maintaining system stability.

After evaluating and comparing the two models, namely Logistic Regression and Neural Network, it was observed that the training accuracy of the NN model (85.65%) was slightly better than the LR model (80.04%). It was also observed that the NN model outperformed the LR model on the testing dataset with 91.03% accuracy and 84.14% respectively. The lower accuracy of the LR model suggests that the task of predicting if a service quality can be migrated or not involves some non-linear relationships that the NN is better equipped to capture. From Fig 21. it is evident that the model predicted with 91.03% accuracy that migration should take place.

It was also evident when the models were integrated to check for the feasibility of live migration based on monitored metrics. Leveraging historical metric datasets acquired through Prometheus and Grafana, the model accurately forecasted the readiness of the target machine for service deployment. The model's binary predictions provided actionable insights for decision-making regarding service deployment strategies. Rigorous validation tests against diverse datasets affirmed the model's accuracy and reliability. The performance evaluation of the model against known outcomes showcased its predictive power and effectiveness in forecasting device readiness for service migration. Continuous evaluations ensured consistent

improvements, refining the model's accuracy and enhancing system efficiency.

VI. Discussion

The obtained results align closely with the project's objectives of leveraging Prometheus and Grafana for efficient data collection, visualization, and predictive analysis to aid in decision-making for service deployment feasibility. The successful implementation of alert rules in Grafana ensured proactive identification and resolution of potential issues, maintaining system stability. The neural network model's accurate predictions regarding device readiness for live migration validated its efficacy in utilizing key metrics for informed decision-making.

Choosing the correct performance metrics have been one of the most crucial factors that helped in determining whether the service can be migrated or not. Another important factor was the time interval during which the data was being retrieved and how often was it being retrieved. If the interval had been more than 1 min long, there would a potential loss of data, hence a significant small interval of 15 seconds was choosen to fetch the data from the target machine both during migration and when migration was taking not place. Another important aspect of the system was the alert rule which was configured. For this a reasonable threshold value is required. A value too low will always give out alert and a value too high will never give one.

A. Challenges

Throughout the project lifecycle, several challenges were encountered and effectively addressed. One notable challenge was the metric selection and optimization of the neural network model to handle a limited number of metrics while ensuring predictive accuracy. This challenge was mitigated through meticulous metric selection, preprocessing, and continuous model refinement based on feedback loops, enhancing its efficiency and predictive power. The use of Prometheus for real-time monitoring has proven effective, yet it faces limitations in long-term data storage. Prometheus excels in immediate insights but struggles with prolonged historical data retention due to its time-series database model.

To address this, Thanos emerges as an optimal choice for long-term data storage. [29] Thanos' architecture is tailored for scalability and efficiency in managing vast historical datasets, making it ideal for prolonged data retention. Thanos offers scalable and durable storage, ensuring accessibility to historical records crucial for machine learning model training. By integrating Prometheus for short-term insights and Thanos for extended historical data, a cohesive solution can be achieved, enhancing the infrastructure for both real-time monitoring and in-depth analysis.

B. Comparison with existing Literature Work

The findings from this project resonate with existing literature highlighting the efficiency of Prometheus and Grafana in monitoring and visualization. However, the integration of machine learning for predictive analysis in service deployment feasibility, especially utilizing neural networks with a concise set of metrics, contributes uniquely to the field. While existing studies emphasize broader metric sets or alternative algorithms, this project's focus on simplicity and effectiveness fills a gap in practical, resource-efficient deployment decision-making.

C. Prototype

The successful completion of this project demonstrates the potential of Prometheus and Grafana in amalgamating data collection, visualization, and predictive analysis for informed decision-making in service deployment strategies. The utilization of machine learning, specifically logistic regression, showcases its viability in predicting device readiness based on essential metrics. The validation and performance evaluations attest to the model's accuracy and reliability, marking a significant milestone in the project's success.

Moving forward, further enhancements could involve expanding the model's scope to encompass additional metrics or exploring alternative machine learning algorithms to refine predictive accuracy. Additionally, collaborating with industry experts or conducting real-time deployments based on model predictions could validate its practical utility.

In conclusion, the successful execution of this study highlights the synergy between Prometheus, Grafana, and machine learning in enabling efficient monitoring, visualization, and predictive analysis. Overcoming challenges and aligning findings with existing literature underscores the project's contribution to resource-efficient decision-making in service deployment strategies.

VII. Conclusion

In conclusion, our project has successfully established a comprehensive system for monitoring, alerting, and decision-making regarding live migration in a cloud environment. By integrating Prometheus and Grafana, we efficiently collected and visualized a diverse array of system metrics, including CPU utilization, memory consumption, network activity, and more. The alerting system, configured within Grafana, proactively identified potential issues through predefined threshold rules, ensuring timely responses to anomalies.

The integration of machine learning models, specifically Logistic Regression and Neural Network, added a layer of intelligence to the decision-making process. These models, trained on historical metric datasets, accurately predicted the feasibility of live migration based on real-time monitored metrics. The Neural Network, with its ability to capture non-linear relationships, demonstrated superior performance, achieving higher accuracy compared to Logistic Regression.

The connection between alerts and the trained models becomes evident in the system's ability to dynamically respond to threshold breaches. When an alert is triggered, such as low disk space, it not only serves as a notification but also prompts the machine learning model to assess the overall

system readiness for live migration. The continuous evaluation and integration of the models ensure a service-aware decision-making system.

Furthermore, our study discusses a multi-stage live migration process, emphasizing both pre-copy and post-copy techniques. The detailed exploration of each stage, from planning and initiation to activation, provides a robust foundation for understanding the intricacies of live migration.

By enabling the monitoring, logging our system is equipped to make informed decisions on live migration, considering both the immediate system state (through alerts) and historical performance patterns (through trained models). This approach contributes to efficient resource utilization in cloud environments.

The synergy between real-time alerting systems and predictive machine learning models establishes a robust decision-making framework. While alerting systems address immediate concerns, machine learning models provide a forward-looking perspective, enabling stakeholders to make informed decisions about live migrations and ensuring the seamless operation of deployed applications. The continuous refinement and evaluation of both components contribute to the overall reliability and effectiveness of the decision-making system. In summary, the integration of these components contributes to a service-aware and adaptive system that can respond intelligently to dynamic changes in the target system's performance.

VIII. Future Work

For future study, metrics from the current service running on the source would also be taken into account during the migration process. This would require changing the current architecture to adjust to the continuous monitoring of the service on the source machine. Also, many other metrics regarding the source system and states achieved by the application during migration would be included to incorporate a further accurate result. The predictive algorithm used will also be optimized and fine-tuned to make the prediction results more reliable.

Acknowledgment

This research paper is under the guidance and supervision of Henry Cocos for the course named "Cloud Computing" in the Master study program of High Integrity Systems at Frankfurt University of Applied Sciences.

References

[1] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the performance of virtual machine migration," in *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2010, pp. 37–46.

[2] D. Raghunath Patil, B. Borkar, A. Markad, S. Kadlag, M. Kumbhkar, and A. Jamal, "Delay tolerant and energy reduced task allocation in internet of things with cloud systems," in *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, 2022, pp. 1579–1583.

[3] O. Smimite and K. Afdel, "Containers placement and migration on cloud system," *International Journal of Computer Applications*, vol. 176, no. 35, 2020. [Online]. Available: http://dx.doi.org/10.5120/ijca2020920493

[4] M. El-Gendy, A. Bose, and K. Shin, "Evolution of the internet qos and support for soft real-time applications," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1086–1104, 2003.

[5] S. Ranjan, J. Rolia, H. Fu, and E. Knightly, "Qos-driven server migration for internet data centers," in *IEEE 2002 Tenth IEEE International Workshop on Quality of Service (Cat. No.02EX564)*, 2002, pp. 3–12.

[6] IBM. "what are virtual machines (vms)? [Online]. Available: https://www.ibm.com/topics/virtual-machines

[7] vmware by Broadcom. What is a virtual machine? [Online]. Available: https://www.vmware.com/topics/glossary/content/virtual-machine.html

[8] "A survey on data center networking for cloud computing," *Computer Networks*. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912861500300X

[9] C.-T. Yang, J.-C. Liu, K.-L. Huang, and F.-C. Jiang, "A method for managing green power of a virtual machine cluster in cloud." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X14000466

[10] L. Yamuna Devi, P. Aruna, S. Devi D., and N. Priya, "Security in virtual machine live migration for kvm," pp. 1–6, 2011.

[11] M. Masdari and H. Khezri, "Efficient VM migrations using forecasting techniques in cloud computing: a comprehensive review," *Cluster Comput.*, vol. 23, no. 4, pp. 2629–2658, Dec. 2020.

[12] R. Sturm, C. Pollard, and J. Craig, "Application management in virtualized systems," in *Application Performance Management (APM) in the Digital Enterprise*, R. Sturm, C. Pollard, and J. Craig, Eds. Oxford, England: Elsevier, Jan. 2017, pp. 53–69.

[13] M. Vutukuru. Virtualization and cloud computing (cs695). Accessed: 2024-1-30. [Online]. Available: https://www.cse.iitb.ac.in/~mythili/virtcc/

[14] V. De Maio, R. Prodan, S. Benedict, and G. Kecskemeti, "Modelling energy consumption of network transfers and virtual machine migration," *Future Gener. Comput. Syst.*, vol. 56, pp. 388–406, Mar. 2016.

[15] M. R. Hines, U. Deshpande, and K. Gopalan, "Post-copy live migration of virtual machines," *Oper. Syst. Rev.*, vol. 43, no. 3, pp. 14–26, Jul. 2009.

[16] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. USA: USENIX Association, May 2005, pp. 273–286.

[17] N. Jayanandana. (2023) 10 best server performance monitoring tools & software in 2023. Accessed: 2024-1-30. [Online]. Available: https://sematext.com/blog/server-monitoring-tools/

[18] C. Matt. (2022) Prometheus monitoring: The pull approach. Accessed: 2024-1-30. [Online]. Available: https://network-insight.net/2022/06/29/prometheus-monitoring-the-pull-approach/#:~:text=Prometheus%20operates%20on%20a%20pull,the%20data%20at%20regular%20intervals.

[19] T. Taylor. (2023) A comprehensive guide to prometheus monitoring. Accessed: 2024-1-30. [Online]. Available: https://www.weave.works/blog/a-comprehensive-guide-to-prometheus-monitoring#:~:text=Prometheus%20Monitoring%20is%20a%20powerful,your%20systems%20for%20maximum%20efficiency.

[20] GitLab. Node exporter git lab. Accessed: 2024-1-29. [Online]. Available: https://docs.gitlab.com/ee/administration/monitoring/prometheus/node_exporter.html

[21] OpsRamp. Guide to the prometheus node exporter. Accessed: 2024-1-25. [Online]. Available: https://www.opsramp.com/guides/prometheus-monitoring/prometheus-node-exporter/#:~:text=A%20Node%20Exporter%20is%20needed,sub%2Dpath%20on%20port%209100.

[22] Stackhero. Prometheus: Using node exporter. Accessed: 2024-1-15. [Online]. Available: https://www.stackhero.io/en/services/Prometheus/documentations/Using-Node-Exporter

[23] Samber. Awesome prometheus alerts. [Online]. Available: https://samber.github.io/awesome-prometheus-alerts/

[24] R. Margaret. Disk usage. [Online]. Available: https://www.techopedia.com/definition/11411/disk-usage-du

[25] Heptabit. What are the most important cloud migration metrics? [Online]. Available: https://www.heptabit.at/blog/cloud-migration/what-are-the-most-important-cloud-migration-metrics

[26] SolarWinds. What is cpu usage? [Online]. Available: https://www.solarwinds.com/resources/it-glossary/what-is-cpu

[27] Samber. Awesome prometheus alerts. [Online]. Available: https://samber.github.io/awesome-prometheus-alerts/rules#rule-host-and-hardware-1-13
[28] Shivang. What is grafana? why use it? everything you should know about it. Accessed: 2024-1-30. [Online]. Available: https://scaleyourapp.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/
[29] Thanos design documentation. [Online]. Available: https://thanos.io/v0.34/thanos/design.md/