GMR
The Original

A Multivariate Statistical Approach To Network Anomaly Detection In Cloud Data Migration

Ramya M C[1*], Prof. Thriveni J[2]

[1*]Department of Computer Science and Engineering University of visveswaraiah college of engineering Bangalore, India ramyamc.sjce@gmail.com
[2]Department of Computer Science and Engineering University of visveswaraiah college of engineering Bangalore, India drthrivenij@gmail.com

Abstract—Automation of the migration process is a very tedious and challenging task. It involved several parameter including monitoring of the enviroment. Forecasting and taking proper measurement for smooth migration is challenging.Using AI and other techniques we are trying to solve the migration issues. One such attempt is detection of network anamolies in migration environment in quantitative approach.This paper presents a comprehensive analysis of metric anomaly detection techniques specifically designed for cloud computing environments. We examine the unique challenges posed by the dynamic, scalable, and distributed nature of cloud systems, and analyze how metric anomalies can impact the operational efficiency and security posture of cloud services. In addition, we explore emerging methodologies, including machine learning models and statistical analysis techniques, that enhance the precision and speed of metric anomaly detection.By addressing these critical concerns, this paper aims to provide cloud service providers, IT professionals, and researchers with practical insights and effective strategies for identifying and responding to metric anomalies. It serves as a valuable resource for understanding the complexities of metric monitoring in cloud environments and offers guidance on best practices for building robust, adaptive, and resilient cloud infrastructures.

Index Terms—Network anamoly, Data Migration, Cloud Com-puting, Network Flaws, Latency, Packet Loss, Bandwidth Opti-mization, Security, Case Studies.

I. INTRODUCTION

In recent years, businesses have increasingly relied on cloud technologies to transform their operations. Data migration to the cloud, however, comes with a set of challenges, most notably network flaws that can compromise the migration process. This section outlines the significance of seamless data migration and the necessity of detecting network flaws for ensuring a successful transition.

In this study, we focus on predicting monitored application data and evaluating any discrepancies. Our algorithm identifies underlying issues when the calculated error significantly deviates from forecasted metrics. This approach not only detects metrics associated with unusual behavior but also recalibrates data during application migrations across different cloud or virtualized environments with varying configurations. Additionally, in cases of abnormal application behavior, server scaling within specific pools facilitates diagnosis.

As businesses increasingly migrate to the cloud to leverage scalability, flexibility, and cost-efficiency, network latency emerges as a significant challenge. Network latency, the delay in data transmission over a network, can impair user experience, degrade application performance, and hinder seamless integration with on-premises systems.

This introduction underscores the critical relationship between cloud migration and network latency, emphasizing the need for strategies to minimize latency's adverse effects. From meticulous planning and data optimization to adopting advanced technologies like edge computing and content delivery networks, organizations must employ innovative solutions to ensure smooth migration and optimal cloud performance.

In this discussion, we explore the complexities of cloud migration with an emphasis on mitigating network latency challenges. By examining key methodologies, technologies, and real-world examples, we aim to equip businesses with the knowledge needed to navigate cloud migration effectively while managing network latency, ensuring a seamless transition and enhanced user experience in the digital era.

Hotelling's T^2 statistic, rooted in multivariate analysis, offers a robust method to detect outliers and anomalies in high-dimensional datasets. By applying this technique to network data generated during cloud data migration processes, the study aims to identify unusual patterns that might indicate malicious activities, data breaches, or system vulnerabilities. Unlike univariate methods, Hotelling's T^2 considers the interrelationships between multiple variables, making it particularly effective for detecting subtle anomalies in complex network behaviors.

This paper delves into the adaptation and optimization of Hotelling's T^2 statistic for the cloud environment, addressing

the challenges posed by the dynamic nature of cloud networks, varying workloads, and diverse data types. The study explores the selection and weighting of network features, considering factors such as packet sizes, transmission rates, and source-destination pairs, to construct a comprehensive multivariate dataset. Statistical thresholds and anomaly detection criteria are meticulously tuned to minimize false positives while ensuring the timely and accurate detection of genuine network anomalies.

Furthermore, the paper evaluates the performance of the proposed Hotelling's T^2-based anomaly detection system through extensive experimentation on real-world cloud data migration scenarios. Comparative analyses are conducted to assess the method's efficacy in comparison to traditional univariate anomaly detection techniques. The study also investigates the system's scalability, considering large-scale data migrations, and its adaptability to various cloud platforms and network infrastructures.

II. Literature Survey

Data migration to the cloud is a critical process in modern computing environments. This literature survey aims to provide insights into various aspects of data migration to the cloud by referring to academic papers.

J. Narantuya et al. [1] address the challenge of efficiently migrating multiple virtual machines (VMs) across cloud environments with minimal service disruption. Their proposed service-aware migration strategy emphasizes maintaining Quality of Service (QoS) during the migration process. A key innovation in their approach is the consideration of the characteristics of services running on VMs, which informs the migration sequence and decision-making.

The strategy involves analyzing network traffic patterns and service dependencies to optimize the order in which VMs are migrated. This minimizes downtime and ensures continuity of services. The authors demonstrate their method within OpenStack environments, showcasing its practical applicability and effectiveness in real-world cloud data centers.

This work contributes to the broader goal of intelligent migration by integrating service-level awareness into migration planning, thereby improving performance and reliability.

The paper introduces a new approach that optimizes the migration sequence of VMs based on network traffic patterns and the service requirements of the VMs. By evaluating the interdependencies between VMs and the QoS needs, the authors are able to minimize the service downtime and ensure that the VMs are migrated in the most efficient order.

The authors specifically focus on the migration within OpenStack environments, where VMs are deployed and managed in cloud data centers. Their method accounts for both the technical challenges of VM migration (e.g.,bandwidth utilization, downtime, and VM dependencies) and the ”service-level agreements (SLAs)” to monitor cloud activities.

The paper presents a novel service-aware strategy that optimizes VM migration to reduce service downtime and ensure continuity of services during migration. Traffic-Based Migration Sequence: By analyzing network traffic, the authors propose a sequence for VM migration that reduces potential service interruption.

OpenStack Implementation: The proposed method is demonstrated within OpenStack-based virtualized environments, offering practical insights into migration.

The paper discusses the potential of further extending the approach to handle more complex multi-cloud environments and dynamic resource allocation models. It acknowledges the need for more sophisticated algorithms to handle various types of services running on the VMs and to optimize the utilization of resources during the migration.

Shen et al. [2] proposed ”Infrastructure as a Service (IaaS)” model for optimizing workload balancing and reducing communication costs. To achieve this, they suggest migrating VM's to PM's in a way that minimizes performance degradation. Their approach assigns resources and weights to physical machines based on usage, monitors virtual machines to reduce bandwidth costs, and migrates virtual machines with minimal performance impact. Additionally, the paper explores the use of ”Service Level Objects (SLOs)” to prevent unnecessary data movement and demonstrates enhanced performance of their method (RIAL) compared to existing ”load balancing” methods. The authors also examine the effects of virtual machine migration sequences on performance and communication costs, using ”network traffic analysis” to guide the migration process.

Sun et al. [3] propose the use of compression techniques for data movement across networks, introducing the ”Compressed Sensing Routing-Control Method with Intelligent Migration(CSR-IM)” Mechanism based on sensing cloud computing algorithm. CSR-IM algorithm applies compressed sensing principles to optimize data usage, considering the motion velocity and location of the target node. It calculates a routing tree based on the positions of fog nodes, determining the path for data traversal. The data is then processed through an aggregation routing process. This algorithm is particularly suited for fog node data migration. However, a key limitation is that it only works with fixed source and destination points and lacks the ability to intelligently adjust and route to different destinations if necessary.

In data movement, network speed is a critical factor. Without adequate network infrastructure, it becomes impossible to accurately estimate the time required for data transfer. Therefore, it is essential to evaluate and understand the existing ISPs to ensure efficient data transfer speeds.
The paper by Addya et al. [4], explain optimization technique with a dual-objective method founded on ”Ant Colony Optimization (ACO)” is suggested for choosing the most viable ”Internet Service Providers (ISPs)” for transferring data to the intended endpoint. The proposed approach aims to strike a balance between power consumption and migration time, making it a cost-effective solution for data migration. This paper does the comparative analysis for the utilization of the network bandwidth in cost effective way.
V N Ha et al. [5],The paper specifically addresses how virtualization and network slicing can be integrated to support various services, such as high ”data rate”, ”low latency”, and ”massive connectivity”, in a more efficient manner. Authors demonstrate their proposed solution allows for dynamic allocation of resources, improved network resource management, and better isolation between different virtual networks, thereby ensuring that each slice can meet its performance and service-level requirements by capitalizing on cloud resources.
Z. Ma et al. [6] The paper concludes that the proposed container migration mechanism is an effective solution for ”load balancing” in ”edge networks” in ”Power IoT” environments. It provides a practical and scalable approach to managing resources efficiently while ensuring that devices with limited power can still perform their functions effectively.
NetOp Cloud [7] outlines the future of networking, emphasizing NaaS as a subscription-based model that supports dynamic provisioning and industry-specific applications. The paper discusses the role of AI and Generative AI (GenAI) in enhancing network observability, automation, and predictive analytics.
Pascal Menezes [8] explores the 2024 landscape of NaaS, focusing on its cloud-native architecture and the importance of standardization. The study highlights how SDN and automation are central to delivering secure and scalable network services.
Qadir et al. [9] provide a foundational overview of NaaS, presenting it as an API-driven model that decouples network control from physical infrastructure. Their work emphasizes the flexibility and programmability of NaaS, supported by SDN and network virtualization.

Cao et al. (2020) [10] provide a thorough analysis of Network as a Service (NaaS), detailing its benefits, challenges, and key enabling technologies. They argue that NaaS is a transformative paradigm for cloud networking, offering greater flexibility, scalability, and cost efficiency. However, the successful deployment of NaaS requires overcoming significant challenges related to security, performance, and integration with existing infrastructure. The authors conclude by highlighting the potential of NaaS to drive innovation in cloud networking, particularly through the adoption of SDN, NFV, and emerging technologies like 5G and AI.

Singh et al. (2019) [11] provide a comprehensive solution for optimizing bandwidth allocation in Network as a Service (NaaS) for cloud computing environments. Their dynamic bandwidth allocation strategy leverages SDN to improve resource utilization, reduce latency, and ensure fair distribution of resources. The proposed solution not only enhances network performance but also offers practical advantages for cloud service providers seeking to manage increasing demands for bandwidth in scalable, cost-effective ways.

Gupta et al. (2020) [12] offer a comprehensive study of ”Quality of Service (QoS)” in ”Network as a Service (NaaS)”. They introduce a QoS framework that integrates SDN and NFV to ensure service guarantees for network traffic in cloud environments. Their work highlights the importance of SLAs, traffic engineering, and load balancing in achieving reliable and predictable network performance. The paper provides valuable insights into managing and optimizing QoS in NaaS, which is critical for ensuring high-quality user experiences in cloud-based services.

In their paper, Chandra et al. (2014) [13] offer a comprehensive analysis of ”Network as a Service (NaaS)” within the cloud computing paradigm. They emphasize the technological and architectural components of NaaS, particularly the roles of ”Software-Defined Networking (SDN)” and ”Network Functions Virtualization (NFV) in facilitating the dynamic allocation and management of network resources. The authors also explore how NaaS can provide scalable, flexible, and cost-efficient networking services, while ensuring key elements such as ”quality of service (QoS)” and network security.

Hernandez et al. (2021) [14]present valuable contributions to the field of Network as a Service, specifically focusing on the optimization of traffic management and load balancing. Their proposed solutions are aimed at improving resource utilization, scalability, and performance in cloud-based networks. The paper shows that with efficient traffic engineering and dynamic load balancing, NaaS can deliver high availability and meet the growing demand for cloud networking services.

Ramakrishnan et al. [15] addresses the issue of network traffic forecasting, which entails predicting future network

traffic based on historical data. The authors introduce several ”Recurrent Neural Network (RNN)” architectures, ”Long Short-Term Memory (LSTM)” networks, and ”Gated Recurrent Units (GRU)”, to tackle this challenge. They assess the effectiveness of these models on three distinct network traffic prediction tasks: traffic volume forecasting, packet protocol prediction, and packet distribution forecasting. The authors demonstrate that their RNN models deliver cutting-edge results on traffic volume prediction tasks using public datasets like the GEANT and Abilene networks, outperforming conventional statistical forecasting approaches. This study is also the first to explore protocol and packet distribution prediction through the use of RNN architectures.

III. Strategies for Flaw Mitigation

This section delves into different network issues, including latency, packet loss, and bandwidth constraints, highlighting their potential effects on data migration. Through real-world examples and case studies, we illustrate the various scenarios where these issues can impede the smooth progress of migration processes.

The following are the Analysis that will be carried out during network monitoring:

- Latency Analysis: Tools and techniques for measuring network latency and identifying its sources.
- Packet Loss Detection: Strategies for detecting and miti-gating packet loss during data transmission.
- Bandwidth Monitoring: Tools for monitoring available bandwidth and optimizing data transfer accordingly.
- Security Vulnerabilities: Identifying security gaps in the network that might compromise data integrity during migration.

These issues encompassed, but were not restricted to :

- Extended or unforeseen periods of downtime.
- Data corruption, loss, or missing data.
- Difficulties in application performance.

Optimizing Data for Slow Networks: Compression, deduplication, and chunking strategies to enhance data transfer over networks with limited bandwidth. Load Balancing and Failover: Implementing load balancing techniques to distribute data across multiple network paths, ensuring a smoother migration process.

Continuous Monitoring: Implementing real-time monitoring solutions to detect flaws as they arise and enable proactive mitigation strategies.

Traditional application performance management tools like Netuitive, AppDynamics, and Integrien are designed to detect and diagnose potential issues before they cause faults, making them suitable for non-virtualized environments and clusters where applications operate on homogeneous machines in a dedicated manner.

Nevertheless, the efficiency of these tools within virtualized data centers faces challenges due to extended latency associated with collecting network data. During their initial stages, these tools invest a significant amount of time understanding application behavior and patterns of network resource consumption. Only after gathering ample data across diverse network metrics can these tools distinguish between normal and abnormal network behavior, enabling accurate predictions.

Imagine a scenario where a virtual machine migrates from machine ’A’ to machine ’B,’ with these machines belonging to different server classes and having distinct network architectures. For example, if an application shows 50% network utilization on machine ’A’ under specific workload, the same workload might result in only 20% network utilization on machine ’B.’ Existing commercial tools struggle in such situations, struggling to generate precise predictions. Without a proper calibration mechanism, data collected by these tools on machine ’A’ becomes irrelevant for predicting network behavior on machine ’B.’ Moreover, many commercial tools have limitations in the network variables they handle and often lack scalability.

This paper aims to devise scalable methods for fault detection and diagnosis, specifically focusing on adapting to the complexities of on-demand virtualized cloud computing platforms, particularly in the context of network performance.

This paper assists in pinpointing issues related to network latency during data migration to cloud environments.

IV. Design Overview

This paper outlines the core components of a comprehensive fault detection and diagnostic system in virtualized data centers, highlighting their seamless integration with other system elements. The monitoring system assumes a central role, measuring vital system metrics like CPU usage, memory, network I/O, and disk I/O during the process of data migration to the cloud. These metrics, along with their timestamps, are archived in a historical time-series database. Leveraging this database, the forecasting component analyzes and interprets the resource consumption patterns of applications.

This component collaborates with the calibration module to transform network utilization values from the current physical host to values comparable to a benchmark physical machine. The resource utilization predictions from the forecasting module are cross-checked with real utilization values by error components. The fault detection module interacts with these error components to identify application misbehavior, triggering an alert for proactive corrective measures long before any noticeable drop in application performance. In instances of detected anomalies, the fault diagnosis module scrutinizes data from the fault detection component to pinpoint the exact cause of the issue. It then

communicates with the policy-based management system to create a policy, which is subsequently invoked by the provisioning system to

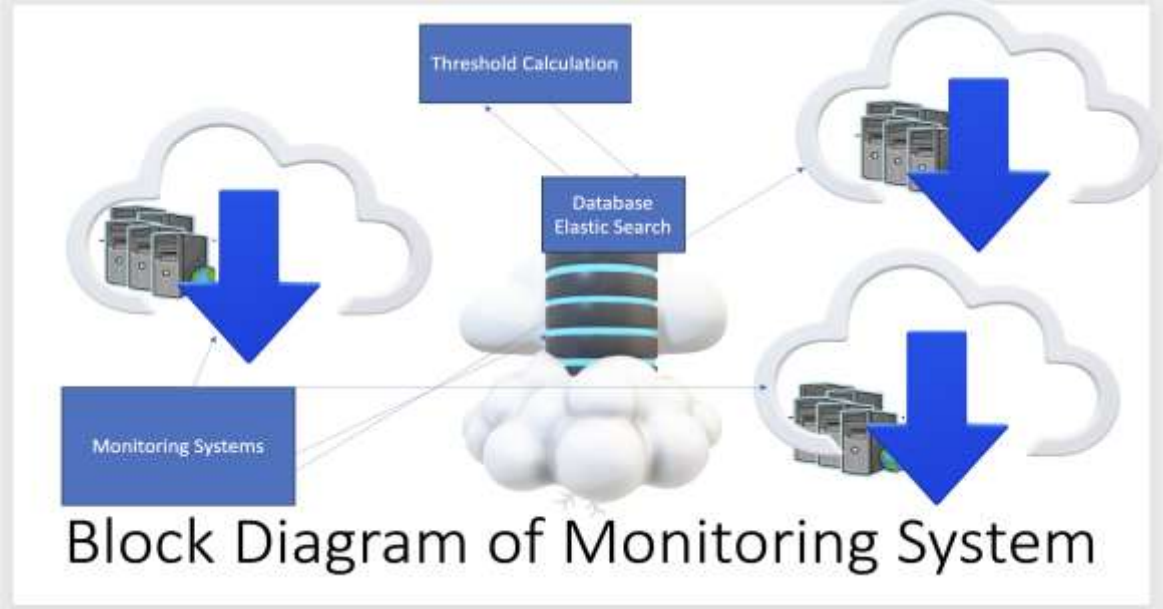

Fig. 1. An overview of monitoring system in cloud rectify the anomaly.

If the fault detection module doesn't identify any abnormal behavior, it signals the policy-based system, which then evaluates various timer-based policies. Based on these policies, the provisioning system allocates resources at different intervals. The monitoring system is permitted to store measured values of various system metrics in the time-series database only when the fault detection system doesn't trigger any alerts.

Figure 1 presents a block diagram depicting the setup of the monitoring system in the cloud environment. The diagram illustrates data migration occurring within a single cloud, between different clouds, and from one cloud to another. The diagram underscores the network's pivotal role in enabling this data migration process. This section provides detailed information about the monitoring system in a cloud environment, covering its configuration, components, and functionalities.

Cloud monitoring systems are pivotal components in cloud computing environments, ensuring the seamless operation, performance, and security of cloud services and applications. These systems provide real-time insights into various metrics, enabling businesses to optimize resource utilization, detect issues promptly, and enhance overall user experience.

The following are the various monitoring techniques we utilized to assess performance during our monitoring activities.

A. Agent-Based Monitoring

Install lightweight agents on virtual machines to gather detailed metrics directly from the host system. These agents can track CPU utilization, memory usage, network traffic, and metrics specific to applications.

B. Agentless Monitoring

Employed agentless monitoring solutions that utilize APIs and protocols to retrieve data from cloud instances. This approach allowed us for resource monitoring without the need for additional software installation, minimizing impact on system performance.

C. Network Performance Monitoring

Track network traffic, bandwidth consumption, latency, and packet loss. Utilize tools for analyzing network behaviors, optimizing data transfer between cloud instances and users.

V. SOLUTION DESCRIPTION

In Figure 1, three clouds with distinct configurations are deployed. The monitoring system oversees all three clouds using an agentless monitoring service.

In this scenario, the dstat tool captures data on packet reception and transmission methods across the clouds. Additionally, it collects I/O process details to pinpoint processes active during data migration that could potentially cause network latency issues.We are also collecting performance metrics including memory and CPU usage.

After collecting data from the third-party tool, it will undergo processing as depicted in Figure 2. The following are the primary components utilized for calculating network flaws:

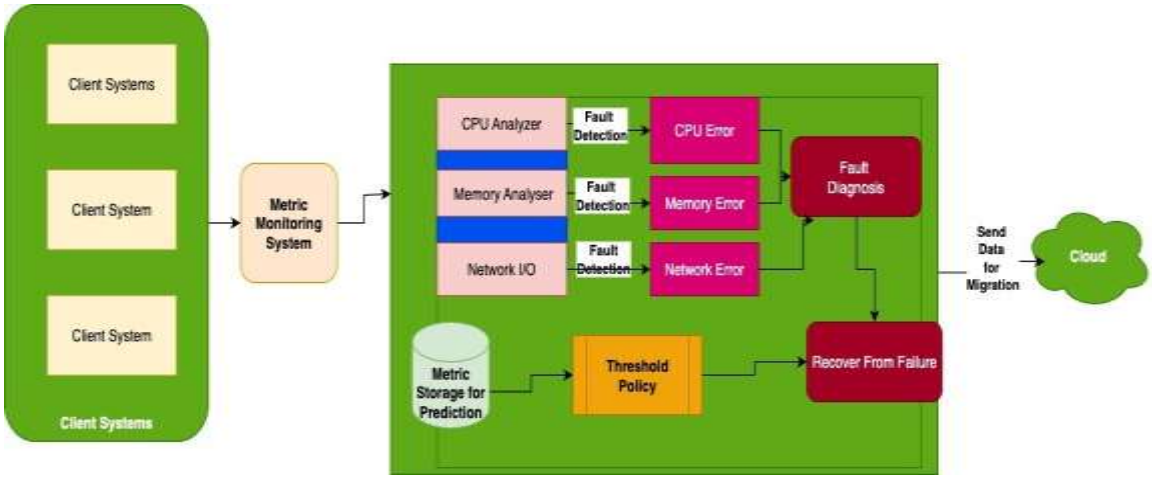

Fig. 2. Design Overview of the Monitoring System

A. Threshold Component

The central objective of this component is to gather and document crucial configuration data, delineating key elements such as hardware specifics, socket details, and switch con-figurations. It also captures hypothetical CPU, memory, and network requirements pertinent to the systems and cloud environments engaged in the migration process. This involves detailing the setups of machines where applications function over their entire lifespan, recognizing that an application might reside on diverse machines, each with its distinct specifica-tions. Initially, forecasting is conducted using hypothetical parameters before being relayed to the fault detection system.

This component establishes threshold values for each metric associated with every component participating in the migration process.

B. Fault Detection

Hotelling's T^2 algorithm is a statistical method used for multivariate analysis. It's particularly useful in network analysis where there are multiple variables to consider simultaneously. This algorithm helps in understanding patterns, correlations, and anomalies within complex network datasets.

During the data migration process, network flaws can disrupt real-time data flow or migration, occasionally causing issues in the data copying phase, resulting in inconsistent data copies or data loss.

Forecasting or predicting network issues based on hypothetical or historical network data analysis assists in maintaining consistency in time and data integrity.

The goal of the forecasting component is to break down the historical usage data of each computing resource into its individual components are explained in detail in Table1. we apply the multiplicative model having the generalized mathematical representation of the form:

$X_t=(T_t*S_t*C_t)*(E_t)$

TABLE I HOTELLINGS T^2 FORMULA ANNOTATION

Notation	Representation	Details
(X_t)	hoteling value at time	Hoteling value at time
(T_t)	Trend Component	Temperature at time
(S_t)	Seasonal Component	Service quality at time
(C_t)	Cyclical Component	Periodic quality at time
(E_t)	Error Component	Error satisfaction at time

HOTELLING'S T^2 OUTLIER DETECTION

Data Preparation

1) Organize your multivariate time series data into a matrix where each row corresponds to a time point, and each column corresponds to a variable.

2) Calculate the mean vector ($\bar{\mathbf{X}}$) and the covariance matrix (**S**) of your data.

Hotelling's T^2 Calculation

For each time point t, calculate the Hotelling's T^2 statistic using the formula:

$$T^2 = (\mathbf{X}_t - \bar{\mathbf{X}})^T \mathbf{S}^{-1} (\mathbf{X}_t - \bar{\mathbf{X}})$$

Here, $\mathbf{X}_t$ is the vector of observations at time t, $\bar{\mathbf{X}}$ is the mean vector, and **S** is the covariance matrix.

Outlier Detection

Compare the calculated T^2 values against a critical thresh-old derived from the Hotelling's T^2distribution. If T^2 exceeds the critical threshold, you may consider the corresponding time point t as an outlier.

Algorithm 1 outlining the Hotelling's T^2 Outlier Detection

Algorithm 1 Hotelling's T^2 Outlier Detection

Input: Multivariate time series data matrix X, Significance level α
Calculate mean vector $\bar{X}$ and covariance matrix S from X
for each time point t in X **do**
    Calculate T_t^2 using the formula:

$$T_t^2 = (\mathbf{X}_t - \bar{\mathbf{X}})^T \mathbf{S}^{-1} (\mathbf{X}_t - \bar{\mathbf{X}})$$

    Compare T_t^2 with critical threshold from Hotelling's T^2 distribution
    if $T_t^2 >$ critical threshold **then**
        Print: Anomaly detected at time point t
    end if
end for
Output: Detected anomalies

Interpretation

Investigate the time points identified as outliers to discern the nature of anomalies or deviations in your data.

Time Series Decomposition Algorithm

To forecast future resource utilization, predict the cyclical component first, followed by the trend component, and then the seasonal component. These individual component forecasts are then combined using a multiplicative model to generate the final forecast.

Algorithm 2 outlining the Hotelling's T^2 Time Series Decomposition

Algorithm 2 Time Series Decomposition

Input: Time series data X with length n, Seasonality length L
Output: Trend $\hat{X}_t$, Cyclical component $\frac{X_t - \hat{X}_t}{\hat{X}_t}$, Seasonal factors, Random error

procedure COMPUTETOTAL(X, L)
    Compute total for each L-period
end procedure
procedure CALCULATEMOVINGAVERAGE(X, L)
    Calculate moving average for each L-period
end procedure
procedure CALCULATECMA(X, L)
    Calculate centered moving average for each L-period
end procedure

procedure DECOMPOSETIMESERIES(X, L)
    CMA = CALCULATECMA(X, L)
    DistinguishTrendCyclical(X, CMA) ▷ Step 2
    SeasonalFactors = CALCULATESEASONALFACTORS(CMA, L)
    DeseasonalizedData = SEASONALADJUSTMENT(X, SeasonalFactors)
    Trend = ANALYZEDESEASONALIZEDDATA(DeseasonalizedData)
    CyclicalComponent = CALCULATECYCLICALCOMPONENT(X, Trend)
    RandomError = COMPUTERANDOMERROR(X, Trend, CyclicalComponent, SeasonalFactors)
end procedure

VI. Steps Involved in Preprocessing Data using

Hotelling's T^2 is a multivariate statistical method used to detect patterns or anomalies in network latency data over time. Here's how you can apply it:

1) **Data Collection:** Gather network latency data at regular intervals over a specific period of time. Record latency values for each time instance.

2) **Data Preparation:** Organize the collected latency data into a matrix where each row represents a time instance, and each column represents a different dimension of latency.

3) **Calculation of Mean Vector:** Calculate the mean vector $\bar{X}$ of the latency data. This represents the average latency values across all dimensions at each time instance.

4) **Calculation of Covariance Matrix:** Calculate the co-variance matrix S of the latency data. The covariance matrix represents the relationships between different dimensions of latency.

5) **Calculation of Hotelling's T^2 Statistic:** Compute the Hotelling's T^2 statistic using the formula:

$$T^2 = n(\bar{X} - \mu)^T S^{-1}(\bar{X} - \mu)$$

Where n is the number of observations (time instances), $\bar{X}$ is the mean vector, μ represents the expected mean vector, and S^{-1} is the inverse of the covariance matrix S.

6) **Set Threshold for Anomaly Detection:** Determine a threshold for Hotelling's T^2 statistic. Values exceeding this threshold indicate significant deviations from the expected latency patterns.

7) **Anomaly Detection:** Compare the computed Hotelling's T^2 statistic against the threshold. If the calculated statistic is higher than the threshold, it suggests an anomaly or unusual behavior in network latency at that specific time instance.

8) **Visualization (Optional):** Optionally, visualize the Hotelling's T^2 statistic over time to identify periods where the network latency deviates significantly from the expected patterns.

VII. Experimental Results

In this experiment, we performed a time-based data migration, transferring both data and virtual machines to the cloud. We employed the dstat tool for monitoring within the cloud environment. Dstat can fetch statistics from diverse system components, including network connections and IO devices, and analyze network traffic on dedicated lines. For our cloud configuration, we allocated dedicated bandwidth: Cloud 1 with 500 GB, Cloud 2 with 200 GB, and Cloud 3 with its allocated bandwidth, all adhering to their respective Service Level Agreements (SLAs).

During the data transfer process, we encountered issues with Cloud 3 due to its low bandwidth. We collected network traffic data from all clouds using the installed dstat tool. The data was analyzed through a monitoring system equipped with service integrators and duty service analysis tools, which monitored the dedicated network for latency. Disturb, installed on all clouds, gathered information, enabling the detection of issues using the Hotelling's T^2 square method.During the data transfer phase, we faced challenges with Cloud 3 due to its limited bandwidth. We gathered network traffic data from all clouds using the dstat tool installed. This data underwent analysis via a monitoring system equipped with service integrators and duty service analysis tools, overseeing the dedicated network for latency. Disturb, deployed across all clouds, collected information, facilitating issue detection using Hotelling's T^2 statistic method. We conducted experiments in our existing cloud environment, varying the load and transaction mix using JMeter. The application-tier was hosted on a machine with a quad-core processor. We recorded the average response time of transactions, along with the average network utilization and average I/O utilization of the application servers hosting the business logic components, in hourly intervals.Figure 3 and Table II and Table V display the network and I/O utilization of the application tier for a day, captured at 60-minute intervals. Tables III and IV, along with Figures 4 and 5, present the network and I/O utilization errors over a week at identical intervals. The data clearly shows that higher I/O levels lead to evident network latency.Ramya et al., [16] focused on utilizing both unhealthy and healthy real-time datasets to monitor the behavior of virtual machines. Their analysis specifically incorporated metrics related to Memory and CPU. In contrast, our approach involves calculating multivariate data to assess network latency. Unlike the conventional use of a single variable

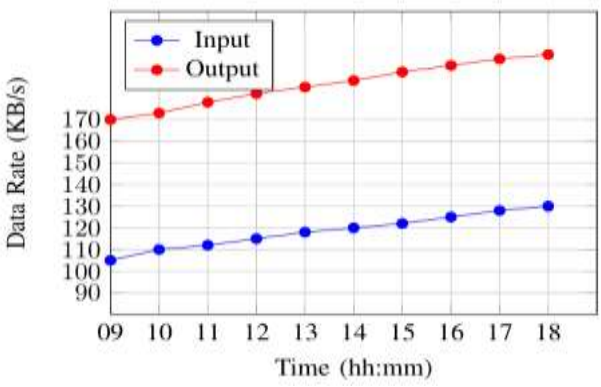

Fig. 3. Healthy I/O Data for 24 Hours

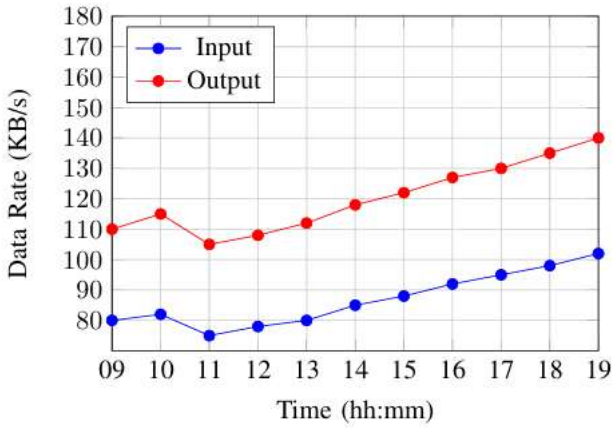

Fig. 4. Unhealthy I/O Data for 24 Hours (Due to Network Latency)

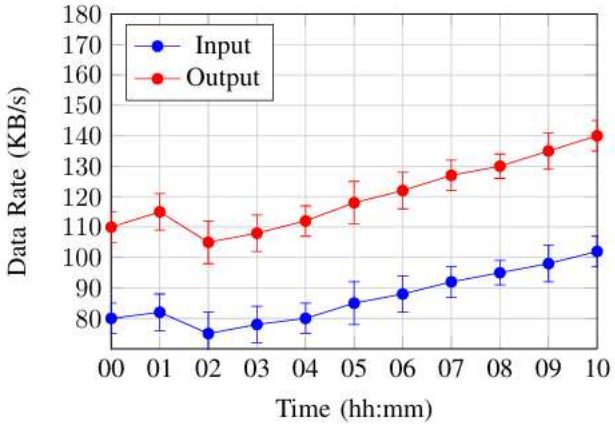

Fig. 5. Unhealthy I/O Data for 24 Hours with Error Bars (Due to Network Latency)

TABLE II HEALTHY NETWORK DATA

Time (hh:mm)	Latency (ms)	Bandwidth Utilization (%)	Packet Loss Rate (%)	Uptime (%)
00:00	5	20	0.1	99.9
01:00	6	22	0.2	99.8
02:00	5	18	0.1	99.9
03:00	4	21	0.1	99.9
04:00	6	25	0.2	99.8
05:00	5	23	0.1	99.9
06:00	4	20	0.1	99.9
07:00	5	22	0.1	99.9
08:00	6	24	0.2	99.8
09:00	5	21	0.1	99.9
10:00	6	22	0.2	99.8
11:00	5	18	0.1	99.9
12:00	4	21	0.1	99.9
13:00	6	25	0.2	99.8
14:00	5	23	0.1	99.9
15:00	4	20	0.1	99.9
16:00	5	22	0.1	99.9
17:00	6	24	0.2	99.8
18:00	5	21	0.1	99.9
19:00	6	22	0.2	99.8
20:00	5	18	0.1	99.9
21:00	4	21	0.1	99.9
22:00	6	25	0.2	99.8
23:00	5	23	0.1	99.9

TABLE III UNHEALTHY NETWORK DATA FOR SELECTED HOURS

Time (hh:mm)	Latency (ms)	Bandwidth Utilization (%)	Packet Loss Rate (%)	Uptime (%)
09:00	200	80	5.2	95.0
10:00	250	85	6.5	92.3
11:00	180	75	4.8	96.2
12:00	300	90	7.2	89.6
13:00	220	82	6.0	94.0
14:00	280	88	6.8	91.7
15:00	210	79	5.6	93.4
16:00	270	87	6.6	90.1
17:00	230	81	5.9	92.1
18:00	260	86	6.4	89.8

TABLE IV UNHEALTHY I/O DATA FOR SELECTED HOURS (DUE TO NETWORK LATENCY)

Time (hh:mm)	Input (KB/s)	Output (KB/s)
09:00	98	135
10:00	102	140
11:00	105	144
12:00	108	148
13:00	110	150
14:00	112	152
15:00	115	155
16:00	118	158
17:00	120	160
18:00	122	162

TABLE V HEALTHY I/O DATA FOR 24 HOURS (DUE TO NETWORK LATENCY)

Time (hh:mm)	Input (KB/s)	Output (KB/s)
00:00	90	140
01:00	92	145
02:00	95	148
03:00	88	135
04:00	87	133
05:00	91	142
06:00	96	150
07:00	98	155
08:00	102	160
09:00	105	165
10:00	110	170
11:00	112	173
12:00	115	178
13:00	118	182
14:00	120	185
15:00	122	188
16:00	125	192
17:00	128	195
18:00	130	198
19:00	132	200
20:00	135	203
21:00	138	206
22:00	140	208
23:00	142	210

TABLE VI HEALTHY I/O DATA FOR 24 HOURS WITH ERROR (DUE TO NETWORK LATENCY)

Time (hh:mm)	Input (KB/s)	Input Error (KB/s)	Output (KB/s)	Output Error (KB/s)
09:00	105	6	165	6
10:00	110	5	170	5
11:00	112	6	173	6
12:00	115	5	178	5
13:00	118	6	182	6
14:00	120	5	185	5
15:00	122	6	188	6
16:00	125	5	192	5
17:00	128	6	195	6
18:00	130	5	198	5

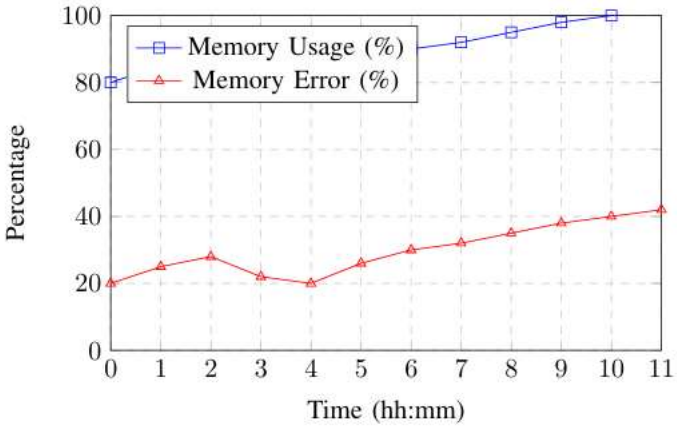

Fig. 6. Memory Usage and Memory Error over Time

TABLE VII HEALTHY CPU AND MEMORY DATA FOR 24 HOUR

Time (hh:mm)	CPU Usage (%)	Memory Usage (%)
00:00	20	80
01:00	25	85
02:00	28	88
03:00	22	82
04:00	20	80
05:00	26	85
06:00	30	90
07:00	32	92
08:00	35	95
09:00	38	98
10:00	40	100
11:00	42	102
12:00	45	105
13:00	48	108
14:00	50	110
15:00	52	112
16:00	55	115
17:00	58	118
18:00	60	120
19:00	62	122
20:00	65	125
21:00	68	128
22:00	70	130
23:00	72	132

TABLE VIII UNHEALTHY MEMORY AND CPU DATA FOR 24 HOURS WITH ERROR

Time (hh:mm)	Memory Usage (%)	Memory Error (%)	CPU Usage (%)	CPU Error (%)
00:00	80	± 3	20	± 2
01:00	85	± 4	25	± 3
02:00	88	± 5	28	± 4
03:00	82	± 4	22	± 3
04:00	80	± 3	20	± 2
05:00	85	± 5	26	± 3
06:00	90	± 4	30	± 4
07:00	92	± 3	32	± 2
08:00	95	± 2	35	± 1
09:00	98	± 4	38	± 3
10:00	100	± 3	40	± 2
11:00	102	± 4	42	± 3
12:00	105	± 3	45	± 2
13:00	108	± 4	48	± 3
14:00	110	± 3	50	± 2
15:00	112	± 4	52	± 3
16:00	115	± 3	55	± 2
17:00	118	± 4	58	± 3
18:00	120	± 3	60	± 2
19:00	122	± 4	62	± 3
20:00	125	± 3	65	± 2
21:00	128	± 4	68	± 3
22:00	130	± 3	70	± 2
23:00	132	± 4	72	± 3

for latency calculations, our method employs a multivariate approach to enhance the accuracy of results.
In Table VI and Table VII there is a clear evident that the CPU and Memory were in healthy state and as soon as the threshold value changes the unhealthy data started reporting as shown in Table VIII.

VIII. CONCLUSION

In conclusion, this paper underscores the critical importance of detecting and addressing network flaws during data migration to ensure a seamless transition to the cloud. By understanding the nature of these flaws and employing targeted detection and mitigation strategies, enterprises can safeguard their data integrity and optimize the migration process, thereby reaping the full benefits of cloud adoption.

- Competing Interests - Network algorithm and security practises in cloud, migration
- Funding Information -Not Applicable
- Author contribution - Author has referred to the existing algorithm and reused it.
- Data Availability Statement -It is real time data used for the validation of the results.
- Research Involving Human and /or Animals -Not Applicable
- Informed Consent -Yes

REFERENCES

[1] J. Narantuya, H. Zang and H. Lim, ”Service-Aware migration of Multiple Virtual Machines,” inIEEEAccess, vol. 6, pp. 76663-76672, 2018,doi:10.1109/ACCESS.2018.2882651.

[2] H. Shen and L. Chen, ”AResource Usage IntensityA ware Load Balancing Method for Virtual Machine Migration in Cloud Datacenters,” in IEEE Transactions on Cloud Computing, vol.8,no.1,pp.17-31,1 Jan.-March2020, doi:10.1109/TCC.2017.2737628.

[3] Sunetal.”CSR-IM: Compressed Sensing Routing-Control-Method with Intelligent Migration-Mechanism Basedon Sensing Cloud-Computing,” in IEEE Access, vol. 8, pp. 28437-28449, 2020, doi: 10.1109/AC CESS.2020.2971537

[4] S.K.Addya, A. Satpathy, B.C. Ghosh, S.Chakraborty and S.K. Ghosh, ”Power and Time Aware virtual machine Migration for Multi-Tier Applications over Geo-Distributed Clouds,” IEEE12th International Conference on Cloud Computing (CLOUD), 2019, pp. 339-343, doi: 10.1109/CLOUD.2019.00062.

[5] V.N.Ha andL. B. Le, ”End-to-EndNetwork Slicing inVirtualized OFDMA-Based Cloud Radio Access Networks,” inIEEEAccess, vol. 5,pp.18675-18691,2017,doi:10.1109/ACCESS.2017.2754461

[6] Z.Ma, S. Shao, S.Guo, Z.Wang, F.Qi and A. Xiong, ”Container Migration Mechanism for Load Balancing in Edge Network Under Power Internet of Things,”inIEEEAccess,vol.8,pp.118405-118416, 2020,doi:10.1109/ ACCESS.2020.3004615.

[7] NetOp Cloud, “The Future of Networking: Trends Shap ing 2025,” NetOp Cloud, White Paper, 2025. [Online]. Available: https://netop.cloud/wp-content/uploads/2025/02/ The-Future-of-Networking-Trends-Shaping-2025-NetOp-Cloud.pdf

[8] P. Menezes, “NaaS 2024: A Look at the Future of Network Services,” Network Computing, 2024. [Online].

Available: https://www.networkcomputing.com/network-infrastructure/ naas-2024-a-look-at-the-future-of-network-services
[9] J. Qadir, N. Ahmed, F. Z. Yousaf, andA. Taqweem, “Network as a Service: TheNewVista of Opportunities,” ArXiv e-prints, 2021. [Online].Available:https://arxiv.org/pdf/1606.03060 10
[10] Cao, H., Zhang, J., and Li, Y., “The Rise of Network as a Service: A New Paradigm for Cloud Networking,” Journal of Cloud Computing: Advances, Systems and Applications, vol. 8, no. 1, pp. 34–45, 2020.
[11] Singh, R., Patel, S., and Sharma, M., “Optimizing Bandwidth Allocation in NaaS for Cloud Computing,” Journal of Cloud Computing: Network ing, Applications and Security, vol. 7, no. 2, pp. 112–124, 2019.
[12] Gupta, S., Agarwal, P., and Sharma, T., “A Study on Quality of Service (QoS) in Network as a Service,” International Journal of Cloud Computing and Service Computing, vol. 9, no. 3, pp. 75–89, 2020.
[13] Chandra, A. and Jain, S., “Network as a Service: A Cloud Computing Perspective,” International Journal of Cloud Computing and Services Science, vol. 3, no. 1, pp. 45–56, 2014.
[14] Hernandez, A., Lopez, J., and Garcia, M., “Traffic Engineering and Load Balancing for Network as a Service,” IEEE Transactions on Network and Service Management, vol. 18, no. 2, pp. 153–166, 2021.
[15] N. Ramakrishnan and T. Soni, “Network traffic prediction using recur rent neural networks,” in Proc. IEEE 17th Int. Conf. Mach. Learn. Appl. (ICMLA), 2018, pp. 187–193
[16] R. Chikkalingaiah, ”Detecting and Diagnosing Application Misbehaviors in ’On-Demand’ Virtual Computing Infrastructures,” in Proc. CCIS, 2011, doi: 10.1109/CCIS.2011.6045060.