ZAMBIA INFORMATION COMMUNICATION TECHNOLOGY (ICT) JOURNAL

Volume 9 (Issue 1) (2025) Pages 38-53

An Overview of Virtual Machine Monitoring Techniques and Observability

Yolam Zimba[1], Hastings Maboshe Libati[2], Derrick Ntalasha[3]
Copperbelt University
1. yzimba@gmail.com; 2. libati@cbu.ac.zm; 3. dbnatalasha@gmail.com

***Abstract*-The adoption of cloud computing systems is rapidly increasing, necessitating robust monitoring to ensure optimal performance and a comprehensive understanding of the internal state of these systems. Effective monitoring is crucial for early anomaly detection and preventing potential failures. Unlike grid computing, cluster computing, and high-performance computing. Cloud computing introduces unique features such as scalability and elasticity, which require specialised monitoring approaches. Existing literature indicates that traditional monitoring tools from previous computing paradigms have been adapted for cloud systems. However, these tools often fall short due to the distinct characteristics of cloud environments. This paper identifies a significant gap in research on monitoring techniques specifically designed for virtual resources and highlights the importance of observability in cloud systems. Observability, which facilitates root cause analysis, is essential for quickly resolving faults by examining the internal state of the cloud system. This paper explores the primary characteristics of cloud monitoring tools, providing examples of currently used tools and their features. Additionally, it surveys virtual machine monitoring techniques and emphasises the critical role of observability in ensuring the rapid resolution of issues through logs, metrics, traces, and dependencies.**

Keywords: Cloud Computing, Virtual Resources, Virtual Machine, Monitoring, observability

I.INTRODUCTION

Cloud computing is increasingly adopted by institutions as an innovative method for hosting applications and delivering computing services to consumers. By utilising virtualisation technology, cloud computing creates a seemingly infinite pool of virtual resources for users. These resources are offered in a multi-tenant manner, facilitated by virtualisation technology, allowing for the virtualisation of memory, storage, processors, and networks. Cloud consumers can dynamically add or remove computing resources based on their processing needs with minimal interaction with the service provider.

In cloud computing, virtual resources are categorised under Infrastructure as a Service (IaaS), which forms the foundational layer upon which other cloud service models are built. The ability to provision and de-provision resources such as storage, processors, and virtual machines on an ad hoc basis renders cloud computing both dynamic and complex. Applications hosted on virtual machines in the cloud necessitate continuous monitoring of virtual resources to ensure optimal performance and availability. This proactive monitoring is essential to detect and resolve issues before they escalate into system failures.

A virtual machine (VM) is a software-based emulation of a physical machine, capable of emulating various computer hardware components such as processors, storage, and network devices. Virtualisation technology, which underpins cloud computing, enables the efficient utilisation of hardware resources by creating a layer of abstraction over physical machines. This abstraction allows for the sharing of computing resources, enabling individual VM's to run different operating systems and configurations while behaving as independent computers on the same physical hardware.

The management of VMs is facilitated by the hypervisor, a software layer that coordinates VMs by providing an interface between them and the physical hardware. Virtual resource monitoring is essential for tracking the state of cloud infrastructure and the applications running on it. Monitoring benchmarks or thresholds are typically defined in the service level agreement (SLA) between the cloud service provider and the consumer.

While monitoring can reveal issues with VMs or cloud resources, observability is crucial for identifying the exact cause of these problems. Observability telemetry, which includes logs, metrics, traces, and dependencies, provides deep insights into the state of the cloud system. This telemetry is used for root cause analysis after monitoring has detected an issue.

Understanding the motivation for cloud monitoring and the various techniques used to monitor cloud resources is vital for managing the cloud infrastructure and detecting anomalies before they impact normal cloud operations. It is equally important to grasp the concept of observability and to employ both monitoring techniques and observability to efficiently manage cloud platforms. By leveraging monitoring and observability, we can reduce the failure rates of

cloud systems and gain a comprehensive understanding of their internal states, enabling effective root-cause analysis.

This paper undertakes a systematic literature review methodology to analyse the current landscape of virtual machine (VM) monitoring techniques and observability practices within cloud computing environments. To structure this analysis, we employ taxonomic classification to categorise cloud monitoring tools into commercial and open-source offerings. Furthermore, a comparative analysis is conducted to highlight the distinctions between traditional monitoring approaches and cloud-native solutions, with a particular focus on their capabilities.

This study is guided by the overarching research question:

How can virtual machine monitoring and observability techniques be optimised to enhance the reliability and performance of cloud systems?

To address this primary question comprehensively, the following subsidiary research questions (RQ) are explored:

RQ1: What are the limitations of traditional monitoring tools when applied to cloud environments?

RQ2: Which techniques are identified as most effective for virtual machine monitoring in the cloud?

RQ3: How does observability complement traditional monitoring methodologies to improve root cause analysis in complex cloud systems?

RQ4: What are the essential characteristics of an effective and robust cloud monitoring system?

RQ5: How do commercial and open-source monitoring tools compare in their ability to address the unique challenges of cloud environments?

The structure of this paper is as follows: Section 2.0 elucidates cloud resource monitoring and the underlying motivations for its implementation. Section 3.0 addresses the monitoring of virtual machines. Section 4.0 investigates the fundamental characteristics of an effective cloud monitoring system. Section 5.0 provides a review of existing cloud monitoring systems and their attributes. Section 6.0 delves into techniques for virtual machine monitoring. Section 7.0 explores the concept of observability. Section 8.0 introduces a conceptual framework. Finally, Section 9.0 offers concluding remarks.

II.CLOUD RESOURCE MONITORING

The inherent complexity of cloud computing arises from the multitude of resources and virtual instances encompassing processing, memory, storage, and networking. These resources contribute to the complexity as they are distributed across the cloud infrastructure, with cloud consumers dynamically adding and removing resources based on their ad-hoc processing needs. This process occurs with minimal interaction from service providers, often resulting in a lack of control over the number of virtual resources being managed. Consequently, continuous monitoring of these resources is imperative to ensure they operate within predefined parameters or thresholds. According to [6], the increasing complexity of cloud infrastructure necessitates enhanced resource monitoring to maintain operational efficiency and manage this complexity. Furthermore, [6] highlights that cloud monitoring is integral to cloud computing, as it enables service providers to predict and track the evolution of parameters related to Quality of Service (QoS), availability, and overall system performance. This capability allows service providers to plan and adapt cloud resources based on monitoring outcomes, ensuring that service level requirements are consistently met.

The primary objective of cloud monitoring is to ensure that cloud service providers deliver reliable and optimally performing services and applications. This practice reduces the failure rate of cloud systems, thereby enhancing the reliability of services provided to cloud consumers [60]. Additionally, cloud monitoring facilitates superior consumer experiences by enabling the prompt identification of vulnerabilities, allowing for the resolution of bottlenecks before they lead to system failures. It also safeguards sensitive and confidential data by monitoring for security breaches, ensuring compliance with privacy and security regulations [60]. As the adoption of cloud computing continues to rise, monitoring the health and status of cloud resources becomes a critical aspect of managing cloud computing platforms for both service providers and consumers [67].

2.1 The Motivation for Cloud Resource Monitoring

Monitoring encompasses the collection, aggregation, processing, and display of quantitative data pertaining to a system, including metrics such as error counts and processing times [70]. Several factors drive both cloud service providers and consumers to engage in cloud resource monitoring. The motivating factors for cloud monitoring are discussed below.

2.1.1 Capacity Planning

Monitoring enables cloud service providers to ensure the availability of sufficient resources to meet the service level demands of their consumers. By maintaining an optimal amount of computing resources, providers can deliver cloud systems and services that adhere to the required quality of service standards [5][66][69]. Furthermore, monitoring facilitates capacity planning by tracking the utilisation of cloud resources. Through this process, both cloud service providers and consumers can identify

resources that need to be scaled up to meet the processing demands of the cloud environment.

2.1.2 Service and Resource Provisioning

By monitoring the utilisation of cloud services and resources, cloud service providers can adjust these services and resources to meet the demands of their consumers. Cloud resources can be provisioned either dynamically or statically. In dynamic provisioning, the workload demands of the consumer dictate the amount of resources required. Conversely, in static provisioning, the cloud resources remain fixed once they are defined. Cloud monitoring facilitates optimal service and resource provisioning for cloud service providers [5][66].

2.1.3 Configuration Management

System configurations consist of defined parameters that dictate how a system operates. These parameters are periodically adjusted to meet system demands or to modify services. Efficient configuration management is achievable only through the monitoring of cloud resources. By tracking deviations in the adjusted parameters, cloud service providers can ensure that changes made to the cloud environment are effective and enhance the operations of the cloud system [5][66][69].

2.1.4 Fault Management

The inherent complexity of cloud environments necessitates continuous monitoring to identify the root causes of faults. In the absence of a monitoring system, pinpointing the specific elements responsible for cloud system failures would be exceedingly challenging. Cloud consumers require monitoring to diagnose the causes of failures within their cloud systems, which can only be accurately determined through the implementation of a robust monitoring system [5][66][69].

2.1.5 Security Management

The security of cloud environments is a critical factor influencing the adoption of cloud computing systems [5]. Cloud consumers require assurance that their data and systems are secure when hosted in the cloud. Consequently, security monitoring and management have become integral components of cloud service offerings. Cloud service providers must be capable of monitoring cloud systems to detect and respond to security breaches and attacks effectively.

2.1.6 Service Level Agreement Management

A Service Level Agreement (SLA) is a contractual document established between the cloud consumer and the cloud service provider. It delineates the agreed-upon service offerings, encompassing quality of service (QoS), pricing, and penalties. Monitoring SLA violations necessitates a robust cloud monitoring system, which is crucial for ensuring consumer satisfaction [66].

An effective cloud monitoring system should possess capabilities such as measuring QoS parameters, storing and analysing data, assessing resource consumption, and evaluating SLA compliance [5][66].

2.1.7 Accounting and Billing

A fundamental characteristic of cloud computing is its metered usage model, wherein services are billed based on consumption, akin to a utility. It is imperative for both the cloud service provider and the cloud consumer to reach a consensus on the billing mechanisms. The accuracy and transparency of accounting and billing information can only be ensured through comprehensive monitoring.

III. VIRTUAL MACHINE MONITORING

Virtual machines (VMs) host applications that operate within the cloud environment. Consequently, it is imperative to monitor the availability, performance, and overall health of these virtual machines to ensure that cloud consumers can access cloud applications and that cloud service providers meet their service level requirements.

Virtual machine monitoring is defined as the process of overseeing virtualised instances or resources across a network [6]. This monitoring is typically conducted using software tools that analyse logs continuously generated by the virtual machine instances. The availability, health, and performance metrics of these instances are displayed on dashboards, which present the data in graphical formats. Cloud service providers utilise these dashboards to detect and respond to anomalies in memory, processing, storage, and network parameters.

Given that cloud consumers frequently add and remove virtual resources on an ad-hoc basis, the cloud environment is inherently dynamic and complex, making it susceptible to failures. Therefore, monitoring plays a crucial role in ensuring that the cloud infrastructure remains reliable, highly available, and maintains optimal Quality of Service (QoS).

IV. ESSENTIAL CHARACTERISTICS OF A CLOUD MONITORING SYSTEM

Cloud monitoring systems continuously collect and verify information regarding the state of cloud resources. This process generates data, information, and knowledge that enable cloud service providers to measure, assess, and effectively manage both hardware and software infrastructures [59]. Given the complexity of cloud computing systems, monitoring systems must possess characteristics that are adaptive to the dynamic nature of cloud environments. Effective monitoring is essential to detect faults before they adversely impact the cloud system.

Cloud systems must exhibit essential characteristics to meet the expectations of both cloud consumers and service providers. According to [55], many cloud monitoring tools have been adapted from those originally developed for grid computing, cluster computing, and High-Performance Computing (HPC) systems. However, cloud computing systems differ significantly from these architectures due to their inherent scalability and elasticity [55][56][59]. As [67] posits, elasticity is a defining characteristic that distinguishes cloud computing systems from previous paradigms such as grid computing and HPC. This unique attribute necessitates that cloud monitoring

systems align with the specific characteristics of cloud computing. Consequently, conventional monitoring tools are inadequate for cloud environments, as they were not designed with the unique demands of cloud systems in mind [67].

The following section outlines the essential characteristics that cloud monitoring systems should possess.

4.1 Scalability

A monitoring system is considered scalable if it can effectively manage numerous probes. It must maintain stability and efficiently handle a substantial number of probes used to collect monitoring data, as required by cloud consumers or service providers. Cloud computing systems utilise a combination of physical and virtual resources, generating significant amounts of data. The monitoring system should be capable of processing and analysing these vast data volumes from various cloud resources and sources (both physical and virtual) without adversely affecting the normal operations of the cloud systems [5][24][55][63].

According to [61] and [64], a cloud monitoring system collects data from diverse cloud resources, filters, aggregates, analyses, and ultimately reports this data, which is then utilised for decision-making. This entire process necessitates computing resources. Consequently, the cloud system must handle this processing without disrupting its normal functioning.

Furthermore, [62] and [63] observe that monitoring distributed systems involves data collection, interpretation, and the display of information related to the interactions of concurrently executing processes within the distributed system. This observation underscores the necessity for cloud monitoring systems to be scalable and not impede the regular operations of the cloud system.

4.2 Elasticity

Cloud systems are inherently dynamic, characterised by the continuous addition and removal of resources. Cloud consumers can augment computing resources as needed with minimal interaction from service providers, presenting a significant challenge to the monitoring system. The monitoring system must adapt to these dynamic changes, accommodating resources added on an ad-hoc and random basis, and accurately report monitoring information for billing and service level agreement (SLA) compliance [5][24][55][63]. The capacity of the cloud monitoring system to adapt to the addition and removal of cloud computing resources is referred to as dynamism [5][65].

Elasticity, a defining feature of cloud computing, distinguishes it from other computing paradigms such as grid computing. Previous paradigms were static and lacked the ability to expand and contract based on resource addition or removal [67]. Cloud computing resources, however, expand and contract according to user requirements. These fluctuating resources must be monitored to ensure accurate billing and resource utilisation reporting. The cloud monitoring system should enable cloud service providers and consumers to modify the metrics being monitored without disrupting the overall operation of the cloud system. Resources should be added and removed seamlessly, without impeding the system's overall functionality [65].

4.3 Adaptability

This characteristic necessitates that a cloud monitoring system must adapt to varying network and computational loads without impeding the operation or functions of other activities. According to [5] and [24], cloud systems exhibit significant dynamism in their operations, making adaptability a critical attribute for a cloud monitoring system. A cloud monitoring tool should maintain its normal operations even when the monitored environment changes by the addition or removal of resources. Adaptability implies that a monitoring system should assist in fault mitigation and provide accurate monitoring information, even when resources are added or removed from the cloud environment on an ad-hoc basis, without disrupting the normal operations of the cloud system [66].

4.4 Timeliness

Cloud monitoring systems must provide timely notifications to enable administrators, cloud consumers, and cloud service providers to act before the cloud system reaches a state of failure [5][24][63]. A monitoring system that fails to deliver prompt notifications is ineffective. Real-time monitoring of cloud resources is essential, as it provides critical information about the state or health of the cloud infrastructure. These real-time insights help prevent system failures by detecting anomalies early and allowing for prompt intervention. Timeliness is thus a crucial characteristic of cloud monitoring systems, as it contributes to reduced failure rates. Lower failure rates in cloud computing enhance the confidence of cloud consumers in cloud services and applications.

According to [67], the integration of artificial intelligence and machine learning can enable cloud monitoring systems to automatically detect faults within the cloud environment and apply remediation techniques to prevent system failures. This approach, known as autonomic computing, involves designing computer systems that can reconfigure their parameters when a component exceeds prescribed thresholds. Autonomic systems are self-healing and self-configuring, requiring minimal human intervention.

4.5 Autonomicity

An autonomic monitoring system possesses the capability to self-manage its distributed resources by automatically responding to unpredictable changes [5][24][55][63]. These systems can reconfigure themselves without human intervention, effectively concealing complex details from both the cloud consumer and the cloud service provider [55]. Autonomic systems are designed to self-heal and recover from critical errors autonomously. This characteristic is essential for cloud monitoring systems due to the dynamic and ever-changing nature of cloud

environments. Virtual machines are frequently created and destroyed based on the needs of the cloud consumer, resulting in a constantly evolving cloud environment. Autonomicity facilitates the maintenance of failure-free cloud systems by enabling the monitoring system to adapt to varying monitoring parameters and promptly communicate this information. This adaptability significantly reduces the likelihood of faults that could lead to errors and, ultimately, the failure of the cloud system.

4.6 Resilience, Reliability and Availability

A monitoring system is considered resilient when it can continue its intended function despite the failure of other components. Resilient monitoring systems are essential for cloud environments, as they must provide critical monitoring information related to billing, SLA compliance, and resource management. Therefore, a monitoring system must withstand component failures and continue performing its intended monitoring functions as expected [5][24].

Reliability in monitoring systems is defined as the ability to perform the required functions under specified conditions for a specified period [68]. Cloud monitoring systems must be reliable and function as required. Additionally, cloud systems are deemed available when they provide services according to their design specifications whenever users or administrators access the monitoring system. The monitoring system should consistently be available and responsive to user requests [68].

4.7 Accuracy

A monitoring system is deemed accurate when the measurements it provides closely align with the actual metrics being assessed. This accuracy is a crucial characteristic of a cloud monitoring system, as precise monitoring information is essential for the effective oversight of cloud systems [5][24][63]. Cloud systems track the usage of resources to bill consumers based on their resource consumption. Accurate billing is contingent upon the cloud system's ability to precisely monitor resource usage and provide reliable usage data for both billing and resource management purposes.

V.CLOUD MONITORING PLATFORMS

Section 4 examined the primary characteristics of cloud monitoring systems. This section focuses on the principal platforms utilised for monitoring cloud computing systems. Cloud monitoring platforms are categorised into two categories: commercial and open source. The section below provides a summary of these monitoring platforms, their respective categories, and the key characteristics exhibited by each tool in relation to the characteristics discussed in section 4.

5.1 CloudWatch

Amazon provides a comprehensive monitoring platform known as Amazon CloudWatch. This service is capable of monitoring various AWS resources, including Amazon EC2 instances. CloudWatch collects and aggregates monitoring data, which is then stored in a database for a retention period of two weeks. Users can utilise this stored data to analyse and visualise performance metrics of their cloud instances, thereby gaining insights into their operational efficiency and resource utilisation [5][66].

CloudWatch is categorised as a commercial monitoring tool and its main characteristics are timeliness, extensibility and elasticity.

5.2 AzureWatch

AzureWatch is a monitoring tool designed for users of the Azure cloud platform. It is capable of collecting key performance metrics related to various Azure resources, including databases and applications. This tool enables users to monitor and analyse the performance and operational efficiency of their cloud-based services, thereby facilitating informed decision-making and resource optimisation [5][66].

AzureWatch is categorised as a commercial cloud monitoring tool and its main characteristics are scalability, adaptability, autonomicity and extensibility.

5.3 CloudKick

This monitoring tool is provided by Rackspace, it is designed to monitor key performance metrics such as CPU utilisation and traffic volumes. This tool offers real-time monitoring capabilities and can promptly alert users to anomalies via email or SMS. By providing timely and actionable insights, Cloudkick enhances the operational efficiency and reliability of cloud-based services [5][66].

Cloudkick is categorised as a commercial cloud monitoring tool and its main characteristics are scalability and adaptability.

5.4 CloudStatus

This monitoring tool supports Amazon Web Services (AWS) and Google App Engine. Built on the Hyperic-HQ platform, it provides comprehensive monitoring information related to user application performance. This tool is instrumental in conducting root cause analysis when performance issues arise within the cloud environment, thereby enhancing the reliability and efficiency of cloud-based applications [5][66].

This tool is categorised as a commercial cloud monitoring tool and its main characteristic is timeliness.

5.5 Nimsoft

This tool offers a unified dashboard for monitoring both public and private cloud infrastructures. It is capable of overseeing cloud services such as Google Apps, Rackspace, Amazon Web Services, and Salesforce. Additionally, it can be utilised to monitor Service Level Agreement (SLA) violations, thereby ensuring compliance and optimising service performance [5][66]. This tool falls under the commercial category and its main characteristics are scalability and comprehensiveness.[5][66].

5.6 Montis

Montis employs agents to gather data on monitored resources, with a primary focus on Amazon Web Services (AWS). This tool is designed to send alerts to users in the event of any issues with the monitored resources, thereby facilitating prompt response and resolution [5]. This tool falls under the commercial category and its main characteristic is comprehensiveness.

5.7 LogicMonitor

LogicMonitor is predominantly utilised for monitoring virtual infrastructure. It possesses the capability to detect newly provisioned resources and commence reporting monitoring data on these resources immediately. Additionally, LogicMonitor can identify and respond to the deletion of resources in real time. The tool provides comprehensive monitoring information through intuitive dashboards, thereby enhancing visibility and management of virtual environments [5]. This tool is categorised as a commercial cloud monitoring tool and its main characteristics are scalability, elasticity and comprehensiveness.

5.8 Nagios

Nagios is a versatile tool employed for monitoring cloud infrastructure. It is capable of overseeing virtual instances and storage services. The Nagios platform is highly extensible, allowing for the monitoring of various aspects of both physical and virtual infrastructure. This flexibility makes it an invaluable resource for comprehensive infrastructure management. Nagios is characterised as an opensource cloud monitoring tool and its main characteristics are adaptability and scalability [5][66].

5.9 OpenNebula

OpenNebula is a comprehensive toolkit designed for managing distributed and heterogeneous cloud infrastructure. It is capable of monitoring both cloud and physical infrastructure, providing valuable monitoring information to cloud providers. OpenNebula collects monitoring data through probes installed on the nodes being monitored, ensuring accurate and timely insights into the performance and status of the infrastructure [5][66]. This tool is characterised as opensource and its main characteristics are adaptability and scalability.

5.10 CloudStack

CloudStack is a robust platform utilised for the deployment and management of extensive networks of virtual machines. To monitor both virtual and physical devices within CloudStack, a Zenoss extension known as ZenPack is employed. This extension is responsible for managing alerts and events related to monitored parameters originating from zones, pods, and hosts, thereby ensuring comprehensive oversight and operational efficiency. This tool falls under the opensource category and its main characteristic is timeliness.

5.11 Nimbus

Nimbus is a comprehensive platform consisting of an integrated suite of tools designed for monitoring, instantiation, configuration, and repair of cloud infrastructure. Predominantly utilised by the scientific community, Nimbus supports a combination of OpenStack, Amazon Web Services (AWS), and other cloud infrastructures. This versatility makes it an ideal platform for cloud deployment and monitoring within scientific research environments. This monitoring tool is classified as opensource, and its main characteristic is autonomicity.

5.12 DARGOS

DARGOS is a sophisticated platform designed for monitoring both virtual and physical resources. It employs a distributed cloud monitoring architecture that utilises a hybrid push and pull approach to disseminate monitoring information. This platform is characterised by its low overhead and has been engineered to be both flexible and extensible. DARGOS leverages agents to collect monitoring data, such as resource usage, from the nodes under observation. This tool falls under the opensource category and its main characteristics are extensibility, adaptability and intrusiveness.

5.13 Hyperic-HQ

Hyperic-HQ is a comprehensive platform that supports the management and monitoring of cloud infrastructures. It is capable of overseeing both virtual and physical resources, providing detailed reporting and analysis of the monitored assets. This platform facilitates the collection of availability, performance, utilisation, and throughput metrics, thereby enhancing the operational efficiency and reliability of cloud-based environments. This cloud monitoring tool falls under the opensource category and its main characteristics are scalability and comprehensiveness.

5.14 Sensu

Sensu employs message queuing to monitor cloud systems, utilising RabbitMQ as its foundational technology, as discussed in section 6.1. It leverages the Advanced Message Queuing Protocol (AMQP) to ensure the secure processing and communication of messages that contain monitoring data. This approach enhances the reliability and security of the monitoring process, facilitating robust cloud infrastructure management. Sensu is an opensource cloud monitoring tool and its main characteristics are extensibility and elasticity.

VI. VIRTUAL MACHINE MONITORING TECHNIQUES

A variety of techniques are employed to monitor virtual machines, driven by the need to ensure effective capacity planning for cloud infrastructure and reduce the failure rate by having key insights into the performance of the cloud environment. These techniques facilitate the detection of failures, identification of underperforming resources, and recognition of redundant cloud resources.

Additionally, they support the evaluation of cloud systems and the detection of policy violations. This section of the paper looks at various techniques that are used to monitor virtual machines in cloud computing.

6.1 Message Queuing

A message queue is a service-to-service communication mechanism utilised in distributed systems. It operates asynchronously and is commonly employed in serverless and microservice architectures. Message queuing involves storing messages in a queue until they are processed and subsequently deleted. This approach facilitates communication between distributed applications by maintaining a sequence of work objects awaiting processing. Message queues provide a buffer for messages when the destination service is busy or offline, ensuring that messages are retained in the queue until the receiver connects and consumes the designated messages.

ZeroMQ, as discussed in [7], is a system that employs message queuing to monitor virtual machines. It collects key metrics from virtual machines via an API (Application Programming Interface) and transmits this information to a central monitoring server. The server then compares the performance of the virtual machine against a benchmark, and if resources are found to be over-utilised, an alert is sent to the relevant system administrator or cloud service provider. ZeroMQ defines queues to which applications connect to transmit messages, which are subsequently read by the receiving application.

A message can contain information about a task, process, or event. The queue retains the message until the receiving application connects and retrieves it. Message queuing ensures that no messages are lost, even if the sender or receiver experiences a fault. Messages remain in the queue until any issues with the sender or receiver are resolved.

A message queue, also known as a message broker, acts as an intermediary or middleware for various services. It can reduce the load and delivery times of web application servers by delegating resource-intensive tasks, thereby enhancing overall system efficiency.

The basic architecture of a message queue is shown below.

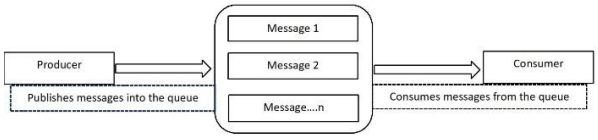

Figure 1: Message Queue Architecture [21]

In Figure 1, the producer produces a message that is published to the queue. The consumer connects to the queue and consumes its assigned message. The message queue keeps the message when the consumer is offline so that the message can be consumed when the consumer is back online.

Although message queueing has many advantages, it has some drawbacks. One of the major drawbacks of message queueing is the message queue itself, the queue can become unreachable, and this poses a problem as messages will not be received and transmitted, this requires that fault-tolerant mechanisms be used to ensure that the message queue is highly available. Availability and performance issues arise when applications fail to communicate with the message queue [28]. The other drawback is that message queueing adds a layer of complexity, this complexity leads to increased processing and in some cases slows down the operation of the system. In [23] message queueing was used for the MonArch system, the MonArch system used a message queueing application called RabbitMQ. RabbitMQ is an open-source message broker and its basic function is to facilitate communication between various applications by allowing the applications to communicate with each other [28]. RabbitMQ acts as a middleman and facilitates the exchange of messages between applications. RabbitMQ is an intermediary for messaging, it provides a safe space for messages to reside until the intended consumer connects to the message queue and consumes the message. The difference between RabbitMQ and ZeroMQ is that RabbitMQ relies on a message queue to transmit messages while ZeroMQ does not need a message queue. It is a brokerless message queue. Brokerless message queues connect directly to the peers and transmit messages directly. There is no broker involved in this transaction [28]. In [30] GlassFish Message queue is used by Oracle to monitor and tune the performance of Oracle systems. GlassFish is a message queueing application and can be configured to monitor various metrics in a distributed system. GlassFish just like any other message queueing system pushes messages to a broker or message queue and the messages are accessed or processed by a consumer [30].

6.1.1 Advantages of Message Queues

Using message queues to monitor virtual machines has several advantages. These advantages are discussed in the sections below.

6.1.1.1 Asynchronous Messaging

Messages are added to the queue and only consumed when the intended recipient (Consumer) is available. This is useful when the consumer is busy with other tasks. The message can wait in the queue and the consumer can execute the message once it is done with other tasks. The queue will continue accepting messages and keep the messages until the consumers are ready to consume their messages [29][30][31].

6.1.1.2 Concurrency

Multiple producers can send messages to the queue at the same time. These messages are kept in the queue as they are sent and the corresponding consumers can consume the messages as and when they are sent to the queue. The order in which the messages are sent does not matter [29][30][31].

6.1.1.3 Monitoring

Message queuing systems have a monitoring feature. This allows monitoring of the queue. This can help

with monitoring the throughput of the queue, identifying problems with the queue, and gaining vital insights and statistics about the queue [29].

6.1.1.4 Decoupling of Tasks

A large task can be broken down or decoupled and pushed into the queue and it appears as a sequence of tasks [29].

6.1.1.5 Persistence

Messages that are pushed to the queue are kept in the queue until they are processed. Messages are only deleted from the queue once they are consumed or processed. Queues ensure that transactions go through, and messages are only discarded when they have been consumed [29][30][31].

6.1.1.6 Resilience

Message queues are resilient in that a faulty component or consumer will not affect the overall functioning of the queue. The other producers will continue submitting messages to the queue. According to [31] this prevents the faulty consumer or component from affecting the entire functioning of the queue.

6.1.1.7 Inter-Application Connectivity

Different applications can access message queues and consume messages that are in the queue. The language or architecture used to develop the application does not matter [29][31]. The ability of the message queues to support various programming languages and protocols is referred to as versatility [31].

6.1.1.8 Improved Security

Some message queuing systems can encrypt, identify, and authenticate messages as they are submitted to the queue. Messages can also be encrypted in transit, at rest or end-to-end. This ensures that messages are protected and this increases the overall security of the queue [31].

6.1.1.9 Guarantee that transactions occur once

Message queues ensure that a transaction occurs once. This is achieved by keeping the message in the queue until the consumer consumes the message in the queue. The message is only deleted from the queue after the consumer has accessed the message. This ensures that the transaction only occurs once [29][31].

6.2 Data Stream Management

In [8] a data stream management system (DSMS) was used to develop a system capable of monitoring multitenant cloud systems. DSMS is software that acts like a Data Base Management System (DBMS). The difference is that the DSMS handles continuous streams of data and the queries are long-running, standing, and persistent [22][33]. Stream data is data that is emitted in real-time, in high volumes, in an ordered sequence, and a continuous stream [33][34][35]. Stream data is incremental and is good for low-latency processing [32]. Stream processing systems ingest a data sequence and incrementally update metrics, reports, and summary statistics. Stream processing systems comprise of a stream producer and a stream consumer [32]. Stream producers are software applications or devices that collect stream data and pass on the collected data to the stream consumer. Stream consumers are software components that process and analyse stream data. Stream producers could be IoT devices or probes that are placed in an environment to continuously monitor that environment based on defined metrics. Similarly, probes can be placed in a virtual machine to monitor the performance of the virtual machine. The probes collect and transmit the metrics from the virtual machine to a stream consumer which then processes that data and provides real-time metrics that aid the cloud service provider in detecting if the cloud system is about to fail or take action when some of the key metrics are above the prescribed threshold. Data stream processing systems are mostly applicable in scenarios where new and dynamic data is generated continuously [32]. DSMS are therefore a perfect fit for virtual machine monitoring due to the dynamic nature of virtual machine monitoring data.

6.2.1 Challenges of Working with Data Streams

Stream data architecture significantly differs from traditional Database Management Systems (DBMS) in that it processes continuous streams of data, introducing a higher level of complexity. According to [36], real-time monitoring systems necessitate distributed stream processing, which contrasts sharply with the conventional processing of static data stored in databases. Stream processing systems handle data in real-time, presenting unique challenges. The following are some of the challenges encountered when working with data streams:

6.2.1.1 Availability

Streaming applications require consistent, low latency, and high availability. Data stream consumers constantly take in new streams of data and they are constantly processing the data in real-time. This means that delays from the producers could cause the system to be in a state of error. Therefore, it is imperative for data stream applications to maintain high availability to ensure uninterrupted data consumption [32][37]. The requirement for availability is a challenge as it brings about a level of complexity [32].

6.2.1.2 Scalability

Raw data streams can experience unexpected surges, particularly during periods of increased system usage. According to [32], such surges may occur during events like social media posts related to major sporting events. During these surges, the data stream system must prioritise the sequencing of incoming data in real-time. [33] posits that data stream management systems (DSMS) order data implicitly based on arrival time and explicitly by timestamps. This process demands a high level of processing power and complexity. Consequently, a DSMS must be capable of adapting to increased processing demands and maintaining normal functionality during periods of heightened data stream activity [37].

6.2.1.3 Durability

Stream data is inherently time-sensitive, and any disruption to the Data Stream Management System (DSMS) can result in data loss. Once stream

processing data is lost, it cannot be recovered or backtracked [33]. Therefore, stream processing systems must be fault-tolerant to prevent the loss of stream data [32]. The loss of access to the data stream compromises the integrity of the processed data, as certain data streams may be missed [37].

6.2.2 Architecture of a Data Stream Management System

DSMS requires a unique architecture due to the nature of the data that they process. The data in a DSMS is a real-time, continuous, ordered sequence of items that are classified or ordered by time of arrival or by a time stamp. If the data is ordered by arrival time it is considered as being implicit. If the data is ordered by time-stamp it is considered explicit. The figure below shows a conceptual architectural diagram of a DSMS.

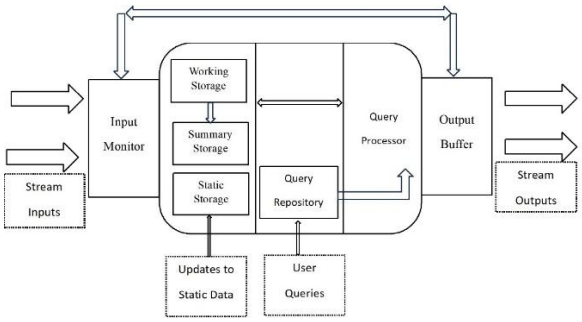

Figure 2: Architecture of a DSMS [32][34]

Data streams that are processed by the DSMS come from multiple sources. Some of the sources include IoT devices, Probes, and event logs from virtual or physical machines that can be of interest to the cloud consumer or cloud service provider. The input monitor is used to regulate the amount of streaming data that is consumed by the DSMS. The input monitor regulates the data streams by dropping some packets. Stream data is stored in temporal working storage, the summary storage, and the static storage. Long-running queries are requested from the query repository. The queries are usually placed in groups for shared processing. The query processor communicates with the input monitor and may re-optimise the queries in response to the dynamic input rates. The output buffer is used to temporarily store the results of the stream processing for the user to view.

[8] acknowledges the nature and complexity of the cloud and emphasises the importance of monitoring the cloud infrastructure to ensure that it is reliable and highly available. [8] further theorises that cloud monitoring can be used for fault detection and proposes that data stream management systems be used to achieve this.[9] affirms that cloud monitoring is critical for cloud computing and various techniques and methods must be researched so that cloud service providers can meet the Service Level Agreements with their cloud consumers and data stream management systems can be used to monitor events that happen in real-time through the use of continuous and persistent queries to monitor the metrics collected from virtual machines[33]. DSMS can be used for fault detection and fault prevention in the systems that are being monitored [34]. The ability of the DSMS to forecast the data streams makes it suitable for monitoring virtual machines as failures that might be experienced by the virtual machine are detected before they occur [34].

6.3 Plugins and Application Programming Interfaces (APIs)

Plug-ins and Application Programming Interfaces (APIs) are essential for integrating with most monitoring tools. Cloud service providers define their metrics and use APIs and plug-ins to extract monitoring information. One such system, NEB2REST, was utilized by [10]. NEB2REST is a custom module that acts as a broker for low-level monitoring infrastructure, enabling Nagios to communicate monitoring information with a RESTful web service via a plug-in.

Nagios, an open-source monitoring tool, monitors both virtual and physical resources through status checks [10]. NEB2REST extends the Nagios platform by serving as an event brokering platform that uses an API to integrate with Nagios and capture cloud monitoring data. Nagios was chosen by [10] for its flexibility and compatibility with APIs. According to [10], NEB2REST employs the Libvirt API, which provides APIs for monitoring CPU utilisation, memory usage, disk I/O, and network I/O for virtual machines [23]. NEB stands for Nagios Event Broker.

[10] further highlights that NEB2REST is easy and convenient to adopt, relying on well-established and tested technologies. It is flexible and adaptable to different cloud environments, capable of monitoring both physical and virtual cloud platforms. Additionally, NEB2REST is a scalable monitoring platform that can easily adapt to various user-defined monitoring parameters. The architecture of NEB2REST is illustrated in Figure 3 below.

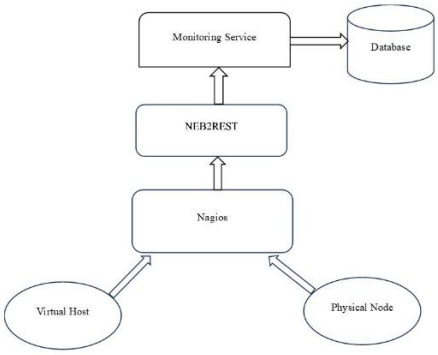

Figure 3: NEB2REST Architecture [10]

The NEB2REST architecture includes a database for storing monitoring data, which can be queried for historical monitoring information. However, the main drawback of the NEB2REST framework is its heavy data exchange between APIs, which can slow down performance due to increased data traffic [10].

6.4 State Machine Replication

State Machine Replication (SMR) was employed by [11] to enhance the resilience of the monitoring system against various types of failures, thereby ensuring the

reliability of system and network monitoring. According to [40], a state machine comprises state variables that encode the state and commands that transform the state. The primary objective of SMR is to emulate a centralised service within a distributed system by replicating the state across multiple hosts, ensuring that the failure of a single host does not compromise the entire system [39].

SMR involves multiple hosts, where communication and transactions from the primary system are replicated to all other hosts, maintaining a consistent system state across all hosts. In the event of a primary system failure, a secondary system assumes control of the entire cloud system's operations. SMR utilises atomic broadcast, guaranteeing that each agreed-upon value is in a consistent state [39]. This fault-tolerant mechanism ensures the continuous operation of a cloud monitoring system, even if one component fails or becomes faulty.

SMR is particularly suited for services requiring high availability and rapid recovery times [38]. Virtual machine monitoring services can leverage state replication to monitor various metrics. Cloud computing resources are billed based on usage, and this billing information is captured only when resources are monitored. SMR was utilised in [8] to ensure the fault tolerance of monitoring services, preventing data loss when a component of the cloud system fails.

SMR implements fault tolerance in distributed systems by replicating servers and coordinating client interactions with server replication [40]. By employing SMR in virtual machine monitoring, cloud service providers ensure that failures are isolated from the entire cloud system, meaning that a failure in one component does not affect the entire system [40]. Figure 4 below illustrates a conceptual representation of the SMR architecture.

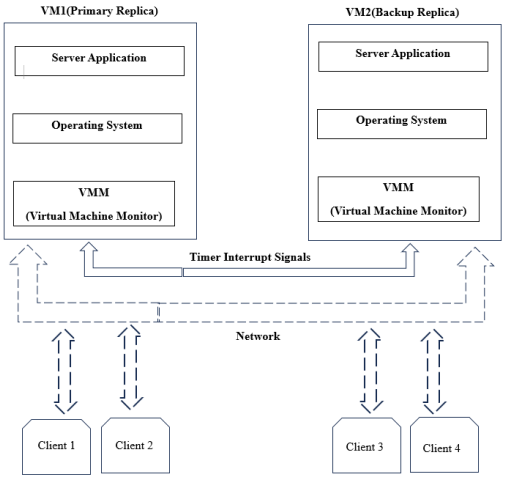

Figure 4: State Machine Replication (SMR) [8][10][39][40]

In Figure 4, a client initiates a request to VM1 via the network, which then forwards the same request to VM2. Both VM1 and VM2 receive identical requests, ensuring synchronisation. During this process, VM1 also sends interrupt requests to VM2. Upon receiving these interrupt requests, VM2 discards the packets intended for client feedback, recognising that VM1 is active through the interrupt signals. If VM2 does not receive the interrupt signal from VM1, it assumes control and begins processing all client requests. The absence of interrupt signals from VM1 indicates to VM2 that VM1 is in a failure state and unable to process client requests. Consequently, VM2 takes over and handles all incoming client requests.

For SMR to achieve fault tolerance, it must maintain at least three replicas, following the formula (2f + 1) [40]. This approach is analogous to the (2N + 1) modular redundancy proposed by [41], which involves two active nodes and one standby node. This configuration ensures high availability in cloud systems. If one node fails, the second node takes over; if the second node also fails, the third standby node assumes processing responsibilities while the other nodes are being recovered [41].

6.5 Regression Analysis

Regression analysis was employed in [12] and [43] to enhance predictive and forecasting capabilities. There are two primary forms of regression analysis: linear regression and logistic regression. According to [44], regression analysis is extensively utilised for prediction and forecasting. In certain contexts, it can also be used to infer causal relationships between independent and dependent variables.

Linear regression is a statistical method used to demonstrate correlations between a criterion or response variable (dependent variable) and one or more predictor variables (independent variables) [43][44]. Logistic regression, on the other hand, is a statistical method used to predict the probability of an outcome using the Sigmoid function. This method is based on binary dependent variables, typically coded as 0 and 1, representing two possible outcomes such as On/Off, success/failure, or healthy/sick [43][44].

Regression techniques can extract meaningful information about workload behaviour, leading to workload characterisation. This process utilises the Virtual Machine Monitor (VMM) interface. As posited by [12], data collected at the VMM level can be used for workload characterisation, which in turn can monitor the behaviour of applications on virtual machines. Workload characterisation is valuable for workload scheduling, analysing workload trends, security analysis, online performance monitoring, and virtual machine health monitoring [12]. VMM based workload profiles can also be used to compare malicious behaviour with normal behaviour, aiding in the identification of malicious attacks on virtual machines [12].

In the regression technique, data is collected from the virtual machine by a front-end system, which then generates features from the collected VMM data. This data is subsequently transferred to the backend, where regression is used to characterize the workloads.

Multiple least squares regression and the Least Absolute Shrinkage and Selection Operator (LASSO) algorithms are applied to the generated features to characterise the workloads and build a model of workload behaviour.

In [43], an overload host detection algorithm based on linear and logistic regression was used to monitor the utilisation of cloud resources. Monitoring data is collected from virtual machines and processed by regression modules comprising linear and logistic regression.

The linear regression model is represented by the following equation:

$$y = b_0 + b_1 * x$$

Equation 1: Linear Equation [43]

Equation 1 is a fundamental linear equation used in various fields such as statistics, economics, and machine learning. y denotes the estimated value; it is the dependent variable. This is the outcome or the variable you are trying to predict or explain. x is the predictor value; it is the independent variable. This is the input or predictor variable that influences the dependent variable. *b0* is the slope of the line. It is also known as the y-intercept, it is the value of (y) when (x) is zero. It represents the starting point of the line on the y-axis. *b1* is the intercept, it is the slope of the line. This coefficient indicates how much (y) changes for a one-unit change in (x). It shows the relationship between the independent variable and the dependent variable. *b0* and *b1* are the regression coefficients and they are computed based on equations 2 and 3 shown below.

$$b_0 = \bar{y} - b_1 * \bar{x}$$

Equation 2: computing the regression coefficient b0[43]

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Equation 3: Computing the regression coefficient b1[43]

In equation 3, *n* is the length of the host utilisation history, *x bar,* and *y bar* are the means of *xi* and *yi denotes the* observation variables.

On the other hand, the logistic regression model is represented by the equation shown below.

$$p(y) = \frac{e^y}{(e^{y+1})} = \frac{1}{(1 + e^{-y})}$$

Equation 4: Logistic regression [43]

In equation 4, y represents the linear regression function. Figure 5 below shows the graphical patterns of linear and logistic regression when plotted on the *XY* plane.

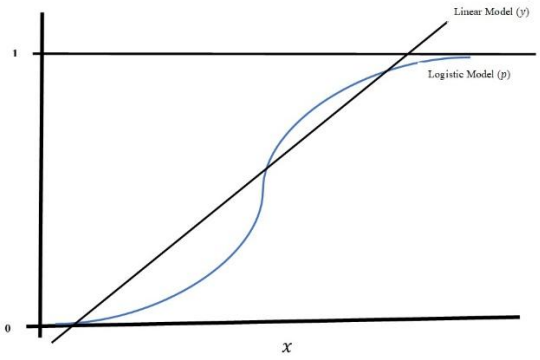

Figure 5: Linear Regression and Logistic Regression [43][44]

6.6 Monalytics

Monitoring and analysis techniques, collectively referred to as Monalytics [13], integrate monitoring and analysis systems to manage large-scale data center systems. According to [14], Monalytics is designed for efficiency and scalability, performing optimally in highly dynamic scenarios. It dynamically discovers resources to monitor and configure at runtime using monitoring agents. Monalytics implementations typically target virtualised cloud infrastructures, often integrating with the Xen Hypervisor.

In [23], a system named MonArch was utilised to monitor large-scale cloud infrastructure. The MonArch system is capable of monitoring physical, virtual, and application layers within cloud infrastructure, demonstrating both scalability and extensibility. MonArch employs agents to collect monitoring data. As [23] further elucidates, monitoring and analytics are fundamental enablers for providing visibility and insights in large-scale cloud infrastructure. A robust monitoring system should possess the capabilities to collect and analyse monitoring data. Such a system aids system administrators in detecting anomalies, identifying SLA violations, and triggering predefined management functions for automated corrective actions. These predefined management functions include stored procedures, algorithms, and autonomic functions that adjust resource allocation based on utilisation by either increasing resources when over-utilised or reducing them when under-utilised and idle. This scalability feature distinguishes cloud computing from earlier models such as grid and cluster computing [55][56].

The optimal online deterministic algorithms and adaptive heuristics for energy and performance-efficient dynamic consolidation of virtual machines in cloud data centers, as utilised in [45], could be integrated with the Monalytics framework proposed by [13]. These algorithms could form part of the automated management functions that detect over or under utilisation of cloud resources, thereby enhancing the efficient usage of cloud resources and minimising the need for human intervention.

6.7 Complex Event Processing

Complex event processing involves querying data before storing it in a database, data is queried on the fly and not stored in a database. CEP is a method used to track and analyse streams of data about things that happen(events) and derive a conclusion from the data in real-time [15][46]. An event is any occurrence that has significance to the operation of a cloud system, an event could be the creation of a virtual machine or the launching of an application [17][46]. The main goal of CEP is to identify meaningful events in real-time in large volumes of rapidly changing and highly varied data so that cloud service providers can take immediate action and respond quickly to events that can have a major impact on the operation of a cloud system [17][46]. CEP allows the analysis of cause and effect relationships in real-time and this intern allows corrective action to be taken on virtual resources before these events negatively impact the operations of the cloud system. [46] further states that a complex event involves a broader category of events, and it typically involves correlations and analysis of multiple events through the detection of patterns, abstraction, and filtering. CEP processes different events that are generated at the same time in multiple locations of a cloud system [46]. Figure 6 below shows a conceptual diagram of CEP

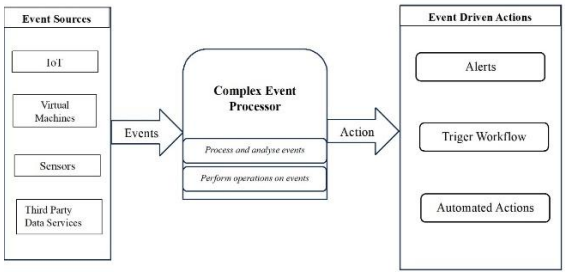

Figure 6: Complex Event Processing [18][19]

In Figure 6, events are generated by various sources, including sensors, probes, IoT devices, and virtual machines. These events are processed in real-time by the complex event processor. If the processed events indicate an anomaly, such as overutilisation of a cloud resource or the failure of a sensor or virtual resource within the cloud infrastructure, immediate actions are taken. The results of the real-time processing trigger actions based on predefined stored procedures or algorithms.

6.7.1 Advantages of Complex Event Processing

The adoption of Complex Event Processing (CEP) as a method for monitoring virtual machines and other cloud resources offers several benefits. These benefits are elaborated upon in the sections below.

6.7.1.1 Real-Time Insights

Complex Event Processing (CEP) offers real-time insights into data from cloud infrastructure. This capability enables cloud service providers to detect and address failures promptly, thereby restoring any compromised components swiftly. Additionally, real-time processing facilitates the identification of over-utilised or under-utilised cloud resources, ensuring their efficient utilisation [46].

6.7.1.2 Detection of Complex patterns and relationships in real-time

Complex Event Processing (CEP) enables the detection of intricate patterns and relationships from various sources in real-time. This capability surpasses the limitations of other monitoring techniques, which are often unable to achieve such real-time, comprehensive analysis [46].

6.7.1.3 Scalability

Complex event processing systems can scale up and down depending on the volumes of events and data streams that are received in real-time [46].

6.7.2 Drawback of Complex Event Processing

In as much as there are advantages to the adoption of CEP as a monitoring technique, there are some drawbacks as well. The drawbacks of using CEP are discussed in the sections below.

6.7.2.1 Complexity

Complex event processing systems are complex to design and maintain. Cloud service providers will face challenges when it comes to designing the rules and algorithms that are needed to run such sophisticated systems [46]. CEP systems require specialised system administrators and developers for the systems to be efficiently managed and maintained.

6.7.2.2 Continual Evolution

Complex Event Processing (CEP) systems must continuously evolve to accommodate changing event patterns and sources. Managing this constant evolution presents a significant challenge, given the dynamic nature of cloud environments where resources are frequently added and removed. The addition and removal of cloud resources generates new events and data streams, which must be monitored and integrated into the existing monitoring system [46].

6.8 Fine-Grained Monitoring

[18] Proposes the use of SysOptic which is a fine-grained monitoring system that is based on PMU (Performance Monitoring Unit) virtualisation to monitor virtual machines. A Performance Monitoring Unit (PMU) on the Central Processing Unit (CPU) can obtain fine-grained monitoring data by adopting interrupt sampling methods based on hardware events [18]. Cloud systems are dynamic and run in error-prone environments, It is therefore important to employ fine-grained status monitoring and anomaly detection at run-time to underpin the design of reliable cloud systems. SysOptic is designed to support the sharing of PMU data and this data is used to simultaneously monitor multiple virtual machines [18]. In [47] a system called cMonitor was used to obtain fine-grained system semantics using virtual machine introspection, cMonitor can monitor all the processes and relate the processes to their network state. cMonitor can transparently monitor the network state outside the virtual machine [47]. In [48] a system called cherub was used to provide fine-grained protection of applications in untrusted environments. Cherub is mostly used for virtual machine security by using fine-grained memory access and flexible

security objectives [48]. Fine-grained monitoring was used for fault detection in [49]. The technique proposed in [49] uses fine-grained application fault detection based on the virtual machine monitor by monitoring system calls to the applications and monitoring the data.

6.9 Virtual Machine Introspection

Virtual Machine Introspection (VMI) is a technique employed to inspect a virtual machine externally, allowing for the analysis of the software running within it. This method, as utilised by [25], is predominantly applied for monitoring the security of virtual machines, particularly in the context of intrusion detection. According to [26], VMI enables an external security monitor to observe the behaviour of software inside a virtual machine, including the guest operating system. This capability is particularly advantageous for security administrators, as it facilitates the identification of illicit programs operating within a system, especially when the operating system kernel has been compromised. The primary objective of VMI is to enforce security policies in scenarios where the operating system is either untrustworthy or compromised [26].

[27] defines VMI as:

"a technique for externally monitoring the runtime state of a system-level virtual machine. Monitors can be placed in another virtual machine, within the hypervisor, or any other part of the virtualisation architecture. For virtual machine introspection, the runtime state can be defined broadly to include processor registers, memory, disk, network, and any other hardware-level events [27]."

VMI is a relatively nascent concept that necessitates further research to be firmly established as a virtual machine monitoring technique. It is inherently complex and demands specialised computing skills for effective utilisation in monitoring virtual machines and large-scale cloud infrastructures.

VII. OBSERVABILITY

Observability provides the capability to gain a profound understanding of the internal state of distributed systems, thereby facilitating the swift resolution of identified issues [50][51][57]. This is accomplished through the analysis of data generated by the system, which includes logs, metrics, traces and dependencies. The fundamental principle of observability is that by examining the outputs of a system, one can infer the internal state of that system [53][54][58]. Observability encompasses the utilisation of software tools and practices that assist in aggregating, correlating, and analysing streams of performance data from cloud systems. By employing these tools and practices for performance monitoring, cloud service providers can effectively monitor, troubleshoot, and debug applications operating within the cloud environment [53][54]. The primary objective of observability is to ensure a comprehensive understanding of the internal state of a cloud system, which aids in identifying anomalies or potential failures, thereby ensuring the system's availability and adherence to service level expectations.

According to [50], the term 'observability' is derived from control theory, a branch of engineering focused on the automation of control in dynamic systems. Examples of automated control include regulating the flow of water through a pipe or controlling the speed of a vehicle over varying terrains based on feedback received from the system [50].

Application monitoring systems periodically sample and aggregate system data, referred to as telemetry. Telemetry aggregated by these monitoring systems alerts cloud service providers to abnormal conditions, thereby aiding in the resolution of potential issues within the cloud infrastructure [51][52].

7.1 Main Telemetry types

Application monitoring platforms continuously discover and collect performance telemetry. This is accomplished by integrating with existing instrumentation embedded within applications and cloud infrastructure. Observability focuses on the following telemetry components:

7.1.1 Logs

Logs are granular, time-stamped, complete, and immutable records of application events. Logs record every event, with the complete context surrounding the event. Logs can be used for troubleshooting and debugging purposes by developers and system administrators. Logs therefore consist of application and system specific details about the operations and flow of control within a cloud system. [52][53][57][58] state that logs provide context for the state of the application when metrics are captured.

7.1.2 Metrics

Metrics are the fundamental measures of application and system health over a given period of time. Metrics measure system aspects like memory and processor usage over a period of time [52][53].

7.1.3 Traces

Traces record the end-to-end journey of every user request or system event. They record what happens in the entire distributed system and the activity can be traced back to a particular user [52][53].

7.1.4 Dependencies

Dependencies reveal how each application component depends on other components, applications, and other cloud resources [51].

Once applications have gathered the telemetry data, the applications then aggregate and correlate this data in real-time and this process provides background information to cloud service providers that help them gain a deeper understanding of why the system is slow or why certain resources are above the agreed threshold [51][52][53][54][57][58].

7.2 The Difference between Monitoring and Observability

Monitoring and observability are closely related concepts that help cloud service providers monitor the cloud infrastructure and identify potential problems.

Both monitoring and observability involve collecting and aggregating data in order to understand or get insights into the performance and health of the cloud infrastructure [52][53][54][57][58]. The main difference is that monitoring captures and displays data about certain defined metrics while observability helps with discerning the health of the cloud infrastructure by analysing its inputs and outputs. With monitoring, a defined metric can be observed for a period of time for changes in order to deduce that there is a problem [52]. With observability, a system will emit data about its internal state and this data is very essential in identifying the root cause of a problem in the system. Monitoring will provide a limited view of the system and the main focus is on individual metrics [51], while monitoring will show that there is a problem with the system, observability will help the system administrators pinpoint the root cause of the problem.

VIII. Conceptual Framework

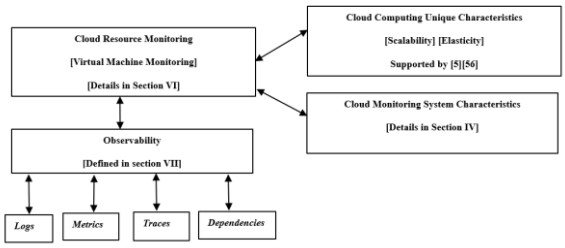

Figure 7: Cloud Resource Monitoring and Observability Conceptual Framework[51][52][53][54]

Figure 7 elucidates that the effective monitoring of cloud resources necessitates a profound understanding of the intrinsic characteristics of cloud monitoring systems. It is imperative to acknowledge that cloud computing represents a unique paradigm, distinguished by its scalability and elasticity.

Therefore, a cloud monitoring system must address these distinctive attributes of cloud computing while incorporating the discussed characteristics of cloud monitoring systems. However, monitoring cloud resources in isolation is insufficient. To obtain deeper insights into the state of the cloud system, observability must also be employed.

Observability entails the examination of logs, metrics, traces, and dependencies, as these elements furnish detailed information regarding the internal state of the cloud system.

IX. CONCLUSION

Cloud computing has fundamentally transformed the landscape of computing, evolving into an increasingly complex technology with the advent of new user demands and emerging technologies. Consequently, it is imperative to implement robust cloud resource monitoring to pre-emptively identify faults before they escalate into system failures caused by defective components or resource over-utilisation.

Virtual resource monitoring techniques are essential for cloud service providers to oversee their cloud infrastructure, thereby ensuring that the service level expectations of cloud consumers are consistently met. However, the surveyed techniques are inherently complex and introduce additional processing overheads to the cloud system. The objective is to adopt lightweight monitoring techniques that do not hinder the normal operations of the cloud system due to their processing demands.

Cloud service providers must select monitoring techniques that are best suited to their specific environments and employ observability to gain a comprehensive understanding of the root causes of cloud system failures and outages. By integrating observability with monitoring, the combined capabilities of both approaches can be harnessed, resulting in more reliable cloud systems with reduced failure rates. Observability provides cloud service providers with deeper insights into the system state, facilitating a clear understanding of failures or potential causes of system disruptions. Through the analysis of traces, logs, metrics, and dependencies, observability aids in the root cause analysis, enabling the identification and resolution of faulty components within the cloud infrastructure.

REFERENCES

[1]SolarWinds(2023) Virtualisation Manager, https://www.solarwinds.com/virtualization-manager/use-cases/vm-monitoring#:~:text=VM%20monitoring%20tools%20function%20by,logs%20produced%20by%20virtual%20machines, last accessed on 15th September 2023

[2]ScienceLogic(2023) Virtual Machine Monitoring, https://sciencelogic.com/glossary/virtual-machine-monitoring#:~:text=A%20virtual%20machine%20monitor%20(VMM,and%20performance%20of%20associated%20VMs, last accessed on 15th September 2023

[3] IBM (nd) What is Virtualisation, https://www.ibm.com/topics/virtualization, last accessed on 15th September 2023

[4] ScienceLogic (2023) Virtual Monitoring, https://sciencelogic.com/glossary/virtual-monitoring, last accessed on 15th September 2023

[5] Aceto G., Botta A., De Donato W., Pescapè A (2013) Cloud monitoring: A survey, http://dx.doi.org/10.1016/j.comnet.2013.04.001

[6] Amazon Web Services(2023) What is Vurtualisation, https://aws.amazon.com/what-is/virtualization/ , last accessed on 16th September 2023

[7] Sri Upanya B, Uthra.V, B.Monica Jenefer (2019) Performance Monitoring System for Virtual Machines, SSRG International Journal of Computer Science and Engineering (SSRG - IJCSE) - Special Issue NCTCT Mar 2019

[8] Hasselmeyer P, D'Heureuse N (2014) Towards Holistic Multi-Tenant Monitoring for Virtual Data Centers, NEC Laboratories Europe, NEC Europe, Ltd. 69115 Heidelberg, Germany

[9] . Hasselmeyer, N. d'Heureuse, Towards holistic multi-tenant monitoring for virtual data centers, in: Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/ IFIP, 2010, pp. 350–356.

[10] G. Katsaros, R. Kübert, G. Gallizo, Building a service-oriented monitoring framework with REST and nagios, in: 2011 IEEE International Conference on Services Computing (SCC), 2011, pp. 426–431.
[11] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, M. Wolf, A flexible architecture integrating monitoring and analytics for managing large-scale data centers, in: Proceedings of ICAC, 2011.
[12] F. Azmandian, M. Moffie, J.G. Dy, J.A. Aslam, D.R. Kaeli, Workload characterization at the virtualization layer, in: 2011 IEEE 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 25–27 July 2011, pp. 63–72.
[13] M. Kutare, G. Eisenhauer, C. Wang, K. Schwan, V. Talwar, M. Wolf (2010) Monalytics: online monitoring and analytics for managing large scale data centers, in: Proceedings of the 7th International Conference on Autonomic Computing, Ser. ICAC '10, ACM, New York, NY, USA, 2010, pp. 141–150.
[14] Lemoine F, Aubonnet T, Henrio L, Kessal S, Madelaine E, Simoni N(2017) Monitoring as-a-service to drive more efficient future system design, https://hal.science/hal-01582593
[15] L. Romano, D.D. Mari, Z. Jerzak, C. Fetzer (2011) A novel approach to QoS monitoring in the cloud, in: International Conference on Data Compression, Communications and Processing, vol. 0, 2011, pp. 4551.
[16] Liu P, Yang R, Sun J, Liu X (2019) ysOptic: A Fine-Grained MonitoringSystem for Virtual Machines Based on PMU. In: Proceedings of the 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). 2019 IEEE International Confe rence on Service-Oriented System Engineering (SOSE), 04-09 Apr 2019, San Francisco East Bay, CA, USA. IEEE , pp. 244-246. ISBN 978-1-7281-1443-9, https://doi.org/10.1109/sose.2019.00042
[17] Gillis S, A (2023) Complex Event Processing, TechTarget, https://www.techtarget.com/whatis/definition/complex-event-processing-CEP
[18] Databricks (2023) Complex Event Processing, https://www.databricks.com/glossary/complex-event-processing#:~:text=Complex%20event%20processing%20is%20an,insight%20into%20what%20is%20happening.
[19] Leavit N (2009) Complex Event Processing Poised for Growth, Computer, Vol.42,NO.4, PP 17-20, Washington
[20] Johansson L (2023) What is RabbitMQ?, https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html
[21] CloudAMPQ (2023) Master Message Queueing, https://www.cloudamqp.com/master-message-queueing.html
[22] Chaudhry A. N(2004) Introduction to Stream Data Management, Department of Computer Science, University of New Orleans, 2000 Lakeshore Drive, New Orleans, LA 70148
[23] Lin J(2015) MonArch:Scalable Monitoring and Analytics for Visibility and Insights in Virtualised Heterogeneous Cloud Infrastructure, https://www.semanticscholar.org/paper/MonArch%3A-Scalable-Monitoring-and-Analytics-for-and-Lin/3e33cab2d00f64aef7031345fe9b15e55f6418cd
[24] Acetto G, Botta A, Walter de Donato, Pescape A (nd) Cloud Monitoring: Definitions, Issues and future Direction, https://www.academia.edu/13859289/Cloud_Monitoring_definitions_issues_and_future_directions?auto=download&email_work_card=download-paper
[25] Garfunkel T and Rosenblum M (2003) A Virtual machine Introspection Based Architecture for Intrusion Detection, Network and Distributed System Security Symposium, https://www.semanticscholar.org/paper/A-Virtual-Machine-Introspection-Based-Architecture-Garfinkel-Rosenblum/4ab4a666f5e5ed34ac219a9fdc2f70bd1cab0922
[26] Jain B, Baig B. M, Zhang D, porter E. D, Sion R (2015) Introspection on the Semantic Gap, IEEE Computer and Reliability Societies
[27] Payne, B.D. (2011). Virtual Machine Introspection. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_647
[28] eG Innovations (2023) Message Queue Monitoring and Observability, https://www.eginnovations.com/supported-technologies/message-queue-monitoring, last accessed on January 24 2024: 01:12
[29] Raje S (2019) Performance Comparison of Message Queue Methods, University of Nevada, Las Vegas, http://dx.doi.org/10.34917/16076287
[30] Oracle (2010) Chapter 4: Using the Metrics Monitoring API, https://docs.oracle.com/cd/E19798-01/821-1796/aeqej/index.html, last accessed on 27 January 2024: 09:30 PM
[31] IBM (nd) What is a Message Queue? https://www.ibm.com/topics/message-queues, last accessed on 27 January 2024: 11:52 PM
[32] AWS(2024) What is Stream Data?, https://aws.amazon.com/what-is/streaming-data/#:~:text=Streaming%20data%20is%20data%20that,to%20several%20megabytes%20(MB). Last accessed on 1st February 2024: 01:23 AM
[33] Golab L and Ozsu T. M (2003) Issues in Data Stream Management, SGMOD, Volume 32, No. 2, June 2003, DOI:10.1145/776985.776986
[34] Alzghoul A (2023) Monitoring Big Data Streams Using Data Stream Management Systems: Industrial Needs, Challenges, and improvements, Advances in Operations Research, Volume 23, Article ID 2596069, https://doi.org/10.1155/2023/2596069
[35] Jiang M, Lee J, Jeong K, Cui Z, Kim B, Hwang S and Choi J. Y (2015) A Data Stream-Based, Integrative Approach to Reliable Easily Manageable Real Time Environmental Monitoring, International Journal of Distributed Sensor Networks, http://dx.doi.org/10.1155/2015/914612
[36] Lopez A. E. M (2018) A Monitoring and Threat Detection System Using Stream Processing as a Virtual Function for Big Data, HAL Open Science, HAL Id:tel-02111017, https://theses.hal.science/tel-02111017
[37] Roddewig S (2023) Data Stream: Use Cases, Benefits and Examples, HubSpot, https://blog.hubspot.com/website/data-stream, Last Accessed on 4th February 2024:20:15
[38] Hess A and Hauk F (2023) Towards a Cloud Service for State-Machine Replication, Institute of Distributed Systems, Ulm University, Germany, https://doi.org/10.18420/fgbs2023f-02
[39] Davidson M (2023) State Machine Replication and Consensus with Byzantine Adversaries, NIST Internal Report 8460 ipd, https://doi.org/10.6028/NIST.IR.8460.ipd
[40] Schneider B. F (1990) Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial, ACM Computing Surveys, Vol. 22, No. 4, December 1990
[41] Zimba Y, Libati M. H and Ntalasha D (2022) Failure Free Cloud Computing Architectures, International Journal of Computer Science & Information Technology (IJCSIT) Vol 14, No 2, April 2022
[42] Prateek S (2022) Replicated State Machines – Ensuring Fault Tolerance and High Availability, https://www.linkedin.com/pulse/replicated-state-machines-ensuring-fault-tolerance-high-prateek last accessed on 11 February 2024. 03:58 AM
[43] Daraghmeh, Melhem B. S, Agrawal A., Goel N, and Zaman M (2018) Linear and Logistic Regression Based Monitoring for Resource Management in Cloud Networks, IEEE 2018, IEEE 6th International Conference on Future Internet of Things and Cloud, doi:10.1109/FiCloud.2018.00045
[44] Mondal S(2024) regression Analysis- Beginners Comprehensive Guide, https://www.analyticsvidhya.com/blog/2020/12/beginners-take-how-logistic-regression-is-related-to-linear-regression/ last accessed on 11th February 2024, 22:02
[45] Beloglazov A., and Buyya, R. (2011) Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. Concurrency and Computation: Practice and Experience, 24(13), 1397–1420. doi:10.1002/cpe.1867
[46] Encora (2023) Guide to Event Stream Processing versus Complex Event Processing, https://www.encora.com/insights/complex-event-processing-cep-vs.-event-stream-processing-esp
[47] Hao Z, Lei Z, Lai X and Lina W (2014) cMonitor: VMI-Based Fines-Grained Monitoring Mechanism in Cloud
[48] Jin H, Cheng G, Zou D and Zhang X (2013) Cherub: Fine-grained Application Protection with On-Demand Virtualisation, Computers & Mathematics with Applications, 65(9), 1326–1338. doi:10.1016/j.camwa.2012.02.001
[49] Liu K, Wo T and Cui L (2013) A Fine-grained Fault Detection Technique Based of Virtual Machine Monitor, 2013 Conference on Cloud Computing and Big Data, DOI:10.1109/CLOUDCOM-ASIA.2013.18
[50] IBM(nd) What is Observability?, https://www.ibm.com/topics/observability#:~:text=Observability%20is%20the%20extent%20you,knowledge%20of%20its%20external%20outputs. Last accessed on 15 February 2024: 01:02 AM
[51] Livens J(2023) Observability Versus Monitoring: What is the Difference? https://www.dynatrace.com/news/blog/observability-vs-monitoring/#:~:text=What%20is%20the%20difference%20between,a%20problem%20%E2%80%94%20this%20is%20monitoring. Last accessed on 15 February 2024: 01:30 AM
[52] Magnusson A (2023) Monitoring Versus Observability: What is the Difference, https://www.strongdm.com/blog/observability-vs-monitoring, last accessed on 15 February 2024: 01:32 AM

[53] Middleware(2023) Observability Versus Monitoring: The ultimate Differential Guide, https://middleware.io/blog/observability-vs-monitoring/, Last accessed on 15 February 2024: 01:33 AM
[54] Samyukktha(2023) Monitoring Versus Observability in 2023, https://medium.com/cloud-native-daily/monitoring-vs-observability-in-2023-an-honest-take-f68df4e2d774, last accessed on 15th February 2024: 01:36 AM
[55] Ward J, S (2015) Efficient Monitoring of large Scale Infrastructure as a Service Clouds, PhD Thesis, University of St Andrews, http://hdl.handle.net/10023/6974
[56] Ward J and Barker A (2014) Observing the Clouds: A Survey and Taxonomy of Cloud Computing, Journal of Cloud Computing: Advances, Systems and Applications (2014) 3:24, DOI 10.1186/s13677-014-0024-2
[57] Kosinska J, Balis B, Konieczny M, Malawski M and Zielinski S (2022) Towards the Observability of Cloud-native applications: The Overview of the State-of-the-Art, DOI 10.1109/ACCESS.2023.3281860
[58] Sridharan C (2018) Distributed Systems Observability: A Guide to Building Robust Systems, O'Reilly Media
[59] Uriarte B., R (2015) Supporting Autonomic Management of Clouds: Service-Level-Agreement, Cloud Monitoring and Similarity Learning, IMT Institute of Advanced Studies, Lucca Italy, https://doi.org/10.6092/IMTLUCCA/E-THESES/163
[60] Nutanix (2023) Your Cloud Monitoring Questions Answered: Definition, Tools, Benefits and More, https://www.nutanix.com/info/cloud-monitoring#definition, last accessed on 3rd June 2024: 00:36 AM
[61] Birje N., M and Bulla C (2019) Cloud Monitoring System: Basics, Phases and Challenges, International Journal of Recent Technology and Engineering (IJRTE), Issue 2277-3878, Volume 8, Issue 3, September 2019.
[62] Joyce J., Lomow G., Slind C., and Unger B (1987) Monitoring Distributed Systems, Volume 5, No 2, May 1987, Pages 121 – 150.
[63] Barje N., M and Bulla C (2019) Cloud Monitoring System: A Review, International Journal of Engineering Sciences and Management – A Multidisciplinary Publication of VTU, 2019, Volume 1, Issue Number 1, page 44-55
[64] Buga A (2015) A Sacalable Monitoring Solution for Large-Scale Distributed Systems, Springer International Publishing Switzerland, DOI: 10.1007/978-3-319-27340-2 28
[65] Hasselmeyer, P., & d'Heureuse, N. (2010). Towards holistic multi-tenant monitoring for virtual data centers. 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, 350-356.
[66] Fatema K., Emeakaroha V. C., Haley D. P., Morrison J., P., and Lynn T (2014) A Survey of Cloud Monitoring Tools: Taxonomy, Capabilities and Objectives, Elsevier, http://dx.doi.org/10.1016/j.jpdc.2014.06.007
[67] Pourmajidi W., Steinbacher J., Erwin T., and Miranskyy A (2018) On Challenges of Cloud Monitoring, arXiv:1806.05914v1 [cs.SE] 15 Jun 2018
[68] Aceto G., Botta A., De Donato W., and Pescape A (2012) Cloud Monitoring: Definitions, Issues and Future Directions, University of Napoli Federico II, Italy
[69] Fahad A., Ahmed A. A., and Kahar M. N. M(2017) The Importance of Monitoring Cloud Computing: An Intensive Review of Monitoring, TENCON 2017 - 2017 IEEE Region 10 Conference, DOI:10.1109/TENCON.2017.8228349
[70] Beyer B., Jones C., Petoff J., and Murphy R. N (2016) Site Reliability Engineering, First Edition, O'Reilly Media Inc, California, United States of America.