IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 November 2022, accepted 12 December 2022, date of publication 22 December 2022,
date of current version 29 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3231622

RESEARCH ARTICLE

FEDARGOS-V1: A Monitoring Architecture for Federated Cloud Computing Infrastructures

VINGI PATRICK NZANZU[1,2,3], EMMANUEL ADETIBA[1,2,4], (Member, IEEE), JOKE A. BADEJO[1,2], (Member, IEEE), MBASA JOAQUIM MOLO[1,2,3], MATTHEW BOLADELE AKANLE[1,2], KALIMUMBALO DANIELLA MUGHOLE[1,2], VICTOR AKANDE[2], OLUWADAMILOLA OSHIN[1,2], (Member, IEEE), VICTORIA OGUNTOSIN[1], CLAUDE TAKENGA[2,3,5], MAISSA MBAYE[6,7], (Member, IEEE), DAME DIONGUE[6,7], (Member, IEEE), AND EZEKIEL F. ADEBIYI[2,8]

[1]Department of Electrical and Information Engineering, College of Engineering, Covenant University, Ota 112104, Nigeria
[2]Covenant Applied Informatics and Communication African Center of Excellence, Covenant University, Ota 112104, Nigeria
[3]Département de Génie Electrique et Informatique, Faculté des Sciences et Technologie Appliquées, Université Libre des Pays des Grands Lacs, Goma 32000, RD Congo
[4]HRA, Institute for Systems Science, Durban University of Technology, Durban 4001, South Africa
[5]Entreprise NTIC, Infokom GmbH, 17033 Neubrandenburg, Germany
[6]Laboratoire d'Analyse Numérique et d'Informatique, Département d'informatique, Université Gaston Berger, Saint-Louis 32001, Senegal
[7]Centre d'Excellence Africain en Mathématiques, informatique et TIC, Saint-Louis 32001, Senegal
[8]Department of Computer and Information Science, College of Science and Technology, Covenant University, Ota 112104, Nigeria

Corresponding author: Emmanuel Adetiba (emmanuel.adetiba@covenantuniversity.edu.ng)

This work was supported in part by the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE) Domiciled at Covenant University through the ACE Impact Grant from World Bank through the National University Commission, Nigeria as well as the SEC-FEDGEN research grant from France Development Agency (AFD) through the Digital Science and Technology Network (DSTN); It was also supported in part by the Covenant University Center for Research, Innovation and Discovery (CUCRID), Covenant University, Ota, Nigeria.

ABSTRACT Resource management in cloud infrastructure is one of the key elements of quality of services provided by the cloud service providers. Resource management has its taxonomy, which includes discovery of resources, selection of resources, allocation of resources, pricing of resources, disaster management, and monitoring of resources. Specifically, monitoring provides the means of knowing the status and availability of the physical resources and services within the cloud infrastructure. This results in making ''monitoring of resources'' one of the key aspects of the cloud resource management taxonomy. However, managing the resources in a secure and scalable manner is not easy, particularly when considering a federated cloud environment. A federated cloud is used and shared by many multi-cloud tenants and at various cloud software stack levels. As a result, there is a need to reconcile all the tenants' diverse monitoring requirements. To cover all aspects relating to the monitoring of resources in a federated cloud environment, we present the FEDerated Architecture for Resource manaGement and mOnitoring in cloudS Version 1.0 (FEDARGOS-V1), a cloud resource monitoring architecture for federated cloud infrastructures. The architecture focuses mainly on the ability to access information while monitoring services for early identification of resource constraints within short time intervals in federated cloud platforms. The monitoring architecture was deployed in a real-time OpenStack-based FEDerated GENomic (FEDGEN) cloud testbed. We present experimental results in order to evaluate our design and compare it both qualitatively and quantitatively to a number of existing Cloud monitoring systems that are similar to ours. The architecture provided here can be deployed in private or public federated cloud infrastructures for faster and more scalable resource monitoring.

INDEX TERMS FEDARGOS, federated cloud computing, monitoring, OpenStack.

The associate editor coordinating the review of this manuscript and approving it for publication was Rentao Gu.

I. INTRODUCTION

Cloud computing has become a common and critical concept in the field of Information Technology (IT) [1]. At this point,

all one can do is speculate on how technology will evolve in response to new paradigms and which applications will migrate to them. The economic, social, ethical, and legal ramifications of technology are likely to evolve, requiring users to rely on enormous data center infrastructure while storing their private data and software on decentralized platforms [2].

Several types of IT-based systems have demonstrated in the last two decades that the trend in cloud computing has moved from a single service provider paradigm to Interconnected Cloud Computing Environment (ICCE) paradigm, with several distributed public and private cloud platforms [3]. Cloud computing is likely to benefit and support science and engineering applications, data mining, IoT, computer finance, gaming, and social networking, as well as many other high-performance computing and data-intensive activities. In addition, a broad range of data, from high-energy physics experiments and various computations, financial or organizational data management to personal data such as images, addresses, videos, and movies, can be stored in the Cloud [2], [4], [5].

ICCE paradigm is essentially the use of capabilities from many cloud platforms from multiple suppliers or Cloud Service Providers (CSPs) to manage, optimize, and expand operations. Several CSPs combine their resources under this paradigm to suit the needs of customers. In the ICCE paradigm, CSPs share their resources via regulated federation. Inter-cloud, federated cloud, multi-cloud, and interconnected cloud infrastructures are called ICCE paradigm [6].

In any cloud paradigm, CSPs are in charge of managing computational resources (hardware, CPU, storage, applications, etc.) [7]. Management of resources is a crucial aspect of any human-made system because it impacts the three fundamental factors for assessing it: functionality, performance, and cost [2], [8], [9], [10]. Inefficient management of resources has a clear and direct significant negative influence on efficiency and cost, and an implicit impact on the system operations. Due to poor efficiency, certain functions can become extremely costly or even ignored. However, apart from the intrinsic advantages that accompany the federation of CSPs, management of the pool of resources becomes challenging.

In a federated cloud architecture, the resource management taxonomy consists of pricing and billing, discovery and provisioning, selection and policy, monitoring and metering, allocation, disaster management, optimization, federation management, Service Level Agreement (SLA) management, and security and identity management.

A cloud-based IT infrastructure's operational workflow can be reviewed, monitored, and managed through the cloud monitoring process [12]. The cloud resource monitoring function provides broad monitoring services and infrastructure information and data, such as access control, service elasticity, service billing, and management of the Service Level Agreement (SLA), etc. [13]. Cloud resource monitoring contributes in addressing features such as scalability, openness, accuracy, and flexibility [14].

A federated cloud is characterized by the fact that resources are shared in a highly dynamic environment with unpredictable loads. In addition, the setting may often change because of several policies related to the heterogeneity of the environment. Also, different middleware stacks are in use by various CSPs to facilitate different characteristics and capabilities of each node. A group of nodes could be optimized for CPU-intensive computation while others will be optimized for Input/Output throughput. Consequently, a cloud monitoring system should be aware of logical and physical groups of resources and should organize monitored resources according to certain criteria to separate and locate the monitoring functions. Another issue is related to the scalability of the system in terms of processing and bandwidth overhead. The plethora of Virtual Machines (VMs) forming the cloud environment is equipped with several physical and logical sensors collecting monitoring data. These components can eventually generate a considerable amount of network traffic that consumes precious network bandwidth. Hence, the monitoring support should be as least intrusive as possible by adopting lightweight processing and communication solutions that limit the additional overhead. It also must assure timely monitoring of data delivery.

This paper presents an enhanced monitoring architecture that extends the existing Distributed Architecture for Resource management and mOnitoring in cloudS (DARGOS) in the literature [15], Notably, DARGOS monitoring solution has a huge potential to fit in a federated cloud infrastructure because it is a flexible and robust monitoring solution for cloud environment based on publish/subscribe paradigm. Moreover, DARGOS uses the Data Distribution Service (DDS) standard to focus on data representation and communication aspects. In addition, DARGOS organizes monitoring among cloud nodes and interested peers in a fully distributed manner, enabling timely, flexible, and reliable monitoring in highly dynamic and multi-tenant cloud provisioning scenarios [15]. DARGOS allows the consumer to have the versatility of controlling granularity according to his/her requirements. However, DARGOS does not fit well into federated cloud platforms in addition to other limitation that listed here:

i. Quality of Service (QoS) is not its main priority.
ii. There are no specific measures taken to guarantee low latency.
iii. System scalability in terms of processing and bandwidth overhead is another crucial issue. For instance, medium-sized Cloud data centers frequently have hundreds of physical hosts and thousands of virtual machines (VMs); at worst, large ones can quickly have more than a thousand physical hosts with multiple VMs that are each outfitted with a number of physical and logical sensors that collect monitoring data. These frameworks have the potential to eventually produce a significant volume of network traffic, using up valuable network bandwidth.

Thus, our proposed architecture named FEDerated Architecture for Resource manaGement and mOnitoring in cloudS (FEDARGOS-V1) is specifically designed for federated cloud monitoring needs, in order to overcome some of the limitations in the existing DARGOS. To overcome these shortcomings:

i. Through the use of minimally-invasive processing and communication techniques, FEDARGOS-V1 aims to be as unobtrusive as feasible.
ii. FEDARGOS-V1 must guarantee the prompt distribution of monitoring data.
iii. FEDARGOS-V1 must also confirm certain requirements in terms of information granularity, correctness, and update frequency for multi-tenant implementations. Consider the possibility of unmanageable overload if each interested node had a different update rate.

The remainder of this paper is organized as follows. Section II presents the major requirements and design parameters for a cloud monitoring architecture and provides an overview of a comprehensive selection of related research initiatives in the subject area. The proposed architecture and its basic features are described in section III. Section IV delves into the presentation of visualizing the monitored data followed by the experimental results that allow the assessment of the developed architecture. Finally, Section V wraps up the paper by presenting the conclusion and outlining future research prospects.

II. LITERATURE REVIEW

Cloud computing is one of the modern computational technologies that may help to solve problems associated with dynamic growth of data, increasing computing resources requests, and the need for extended storage space. Cloud computing provides much storage, compute resources, networking, hardware, and software applications as services, and all of that on-demand [16]. In contrast, one of the significant disadvantages of cloud computing is that small CSPs do not have sufficient capacity to cope with peak demand. Maintaining resources to meet high demand may serve as a palliative function and contributes to wasting energy and resources, thereby raising costs and likely decreasing resources over their life cycles [17]. With time, several CSPs started pooling resources to build the federated cloud to address these limitations.

A. CLOUD COMPUTING

Cloud computing has gotten a lot of attentions for its capacity to provide affordability, sustainability, flexibility, reliability, and scalability. Pay-per-use, the cloud's cornerstone concept, has enticed not just individuals but also organizations to take advantage of the new form of income [18]. However, besides the advantages of cloud computing, the model's complexity and underlying technology have raised management and security issues. The complexity of management issues increases as the number of involved aspects such as network, APIs, hardware, and architectures in the cloud paradigm grows [19].

The service-based paradigm has been established as a cloud computing standard by the National Institute of Standards and Technology (NIST). Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) are the three major models that NIST uses to define all IT sharable resources like software, hardware, and networks [20]. Moreover, NIST has given essential enabling service characteristics of cloud technology, which include:

i. On-demand self-service: Calculation of processing resource, network, storage, and server utilization can be automatically done for the client, with no human interaction required.
ii. Broad network access: All services are available over the network and can be accessed from a variety of devices, including PCs, Personal Digital Assistant (PDA), and mobile phones.
iii. Resource pooling: A multi-tenancy solution serves numerous clients while pooling resources and allocating virtual and physical resources on a client-by-client basis. These resources have ambivalent locations, which means the client doesn't have control over or knowledge of the specific location of the resources in question.
iv. Rapid elasticity: Services can be delivered promptly and in a variety of ways. The resources providing the required services to the client are frequently uncontrollable and can be acquired in large quantities at any moment. Moreover, in some cases, scaling can be done automatically.
v. Virtualization: Abstraction of building a computer that allows for resource splitting in the cloud infrastructure. Sharing resources is possible with the use of a Virtual Machine (VM) and a file called image, which can be created by users or obtained from outside sources.
vi Cloud Management Platform (CMP): Virtual servers, computers, and infrastructure are managed by the CMP. It also oversees their execution, operation, and procedures. In addition, CMP oversees both the software and the back-end hardware. The CMP's functionality, on the other hand, varies depending on the virtual environment and cloud services in use.

Even though technology giants like Google, Microsoft, and IBM strive to give the best solutions to consumers, each has its own proprietary CMP. They offer public cloud services to users who, unfortunately, do not have complete control over application access. Hence the growing interest in various open-source CMPs. Open-source CMPs such as OpenStack, Eucalyptus, OpenNebula, CloudStack, and others are becoming increasingly popular these days since they enable the creation of a private cloud in a quick and cost-effective manner while giving end users complete access control.

B. FEDERATED CLOUD APPROACHES

A federated cloud (also called cloud federation) is the deployment and management of multiple external and internal cloud computing services to match the business needs of two or several CSPs who pool their resources together. The federated cloud infrastructure operated by a federated cloud broker provides a single mechanism for managing different clouds. In sharing their resources with the federated ecosystem, each cloud participant makes an agreement with the federated broker [26]. This arrangement covers all technological and financial implications of the cloud federation. A federated cloud enhances the essential characteristics that define cloud computing. These characteristics include the on-demand self-service, broad network access, resource pooling, limited scalability and elasticity, the measured services, and the lack of interoperability among CSPs.

There have been a number of works providing various architectures of cloud federation. The authors in [27] presented the Dynamic Collaborative Cloud (DCC), with the particularity of federating several CSPs without the use of any intermediate component. DCC's functioning principle is based on picking one of the collaborating CSPs as the primary cloud, and the remaining clouds work together to improve the availability of the resources in the primary cloud.

The RESERVOIR was presented in [28]. RESERVOIR is a federated cloud system without resource provisioning facilitator agents. In RESERVOIR, the resources of all participating CSPs are partitioned to support numerous application components that are processed independently of one another. For their execution, each component of a given application uses the required resources from anywhere in the resource pool of the federation.

In [29] and [30], the authors presented the InterCloud infrastructure. To organize all the CSPs in the federation, InterCloud uses a central third-party component called "broker." The broker distributes the federated resources to a second-level broker, which in turn supplies users with the resources. On the other hand, authors in [31] came up with a different configuration of InterCloud. The principle here is to keep a global resource availability information repository in all clouds. Replicas of this repository, provided to brokers selected by various stakeholders, facilitate mediation across clouds and the distribution of resources to consumers. In [32], the authors provide a simple prototype of cloud federation architecture. They outlined how the whole platform was developed by providing different methods for implementing the basic modules of the infrastructure. They concentrated on the implementation of metering and billing modules that are very reliable and effective. Experiments were carried out with the help of two CSPs using OpenStack and CloudStack, respectively. The experimental results revealed that the prototype successfully federates the two providers. Furthermore, to meet their technological needs and the associated price, consumers also have a choice of selecting the best service. Each CSP in the federation does not require customers to register separately. However, the communication gap between CSPs, dynamic resource allocation, cost changes, policy regarding multiple customers seeking to reach a single instance, security issues, and policies relating to cloud-based membership within a federation are some important aspects which must be addressed in production scenarios.

Besides well-defined architectures, there are various software systems that facilitate cloud federation and multi-cloud. In [33] and [34], the authors introduced the "Contrail" approach for cloud federation. Contrail is a core federated cloud-building software entity and acts as a bridge between the cloud environment and its users. The Contrail operating principle lies in its ability to monitor all cloud operations in the federation, giving each cloud a distinct identity.

OPTIMUS is a toolset presented in [35]. It facilitates the implementation of both cloud federation and multi-cloud. This toolkit has two components, one is necessary to acquire resources for applications and the other is important to the cloud to allow interoperability and resource provisioning.

Federated Cloud Management (FCM) builds cloud federation infrastructures by deploying all the clouds region of the federation with a software component to facilitate resource distribution and other administrative operations [36].

mOSAIC [37] and STRATOS [38] are open-source APIs for building cloud federation and multi-cloud setups.

C. CLOUD MONITORING ARCHITECTURES

Cloud computing is becoming the de-facto method of building Internet-scale systems day after day. The progress of cloud computing, and by extension cloud monitoring, which is an essential component of resource management, is consequently critical to the development of the next Internet-based services. Cloud computing has its own set of characteristics, which makes monitoring even more difficult [39]. Effective monitoring aids system engineers in making an informed decision about how to enhance their systems by addressing performance bottlenecks and security vulnerabilities. System design, debugging, troubleshooting, maintenance, billing, cost forecasting, intrusion detection, compliance, testing, and more can all be linked with monitoring [40].

Monitoring, at its most basic level, is a three-step process: collecting relevant data, analyzing the aggregated data, and making decisions as a result of the analysis. Simple programs that probe system states, such as the UNIX utilities (i.e. df, uptime, or top) are the most basic monitoring tools [41]. These tools are utilized by a user who analyzes the current status of the system and decides what action to take. As a result, the user (not software) performs the vast bulk of the monitoring process. As computer systems expand in size and complexity, there is a growing demand for automated monitoring solutions that decreases or eliminate the need for human intervention. All or part of the three stages monitoring procedure should be implemented by these systems. However, each of these stages has its own set of problems, particularly in terms of federated cloud computing [42]. Some of the existing cloud monitoring solutions are hereafter presented.

The rapid rise in cloud use brings with it some significant challenges. One of the problems with the metric data configuration is the lack of details. The shortage of virtualization-based security tools is another problem. The problems, as described above, are dealt with in [43], which introduced an adaptive Monitoring Platform-as-a-Service (MonPaaS). MonPaaS is an open-source, online and implementable tool. The author suggested two separate monitoring modes: monitoring of cloud providers and monitoring of users. Also, Nagios with OpenStack were integrated [44]. MonPaaS's strength is its ability to intercept OpenStack's message queue and use messages to update VM information. The MonPaaS module is provided to both cloud providers and users in the form of an API. This creates a separate Monitoring VM (MVM) for each new cloud user to perform the monitoring function. MonPaaS tracks physical and virtual resources as well as updates any physical or virtual infrastructure changes. It performs agentless monitoring, which guarantees a high level of consumer safety. MonPaaS' drawback lies in its method when building separate MVM; it requires additional physical resources. MonPaaS is an advanced IaaSMon version [45].

Nagios is a well-known standard monitoring tool that monitors various deployments of servers. It is an open-source tool that, in its simplest configurations built upon a two-tier hierarchical architecture [46]. The monitoring server is populated with a configuration file that details all the monitored servers with their services. In the process of monitoring, Nagios generates a schedule, then probes all the servers and examines each service according to the schedule. Nagios doesn't suit perfectly with cloud monitoring. A large amount of manual configuration is needed, including adjusting configuration when controlled VMs are instantiated and terminated.

Zabbix was firstly introduced in Tader's work [47] as a cluster monitoring tool. The monitoring solution Zabbix is built for server/agent architecture. The Zabbix server operates on a separate machine so that data sent by Zabbix agents can be collected and aggregated. The Zabbix solution supports a warning system that activates when predefined events and conditions occur, such as when memory consumption reaches 80%. These warnings are helpful in that the stimuli activate preparations for adaptation, such as measures for elasticity. SQL databases store calculated measurements, and data is accessed via a Web front-end and an API Iqbal et al. [48] proposed an extension of Zabbix to meet the needs of cloud monitoring.

The monitoring of a private cloud is a problem since most commercial cloud solutions are extremely expensive [42]. An open-source architecture has been proposed for cloud monitoring in [59] to tackle this issue. The architecture proposed is split into three layers: (i) infrastructure, (ii) integration, and (iii) view. The infrastructure layer addresses the appropriate hardware, software, and operating system. Alternatively, the virtualization system and hypervisors are dealt with in an abstraction layer. The monitoring interface is the responsibility of the view layer (it is known as the Dashboard). Depending on their needs, this interface can have different perspectives for different users. A Private Cloud Monitoring System (PCMONS) was implemented based on this architecture.

In clouds, a large-scale distributed virtualized system's multi-tenancy and sophistication present new problems when considering cloud monitoring. In terms of data representation, storage, processing, and delivery, cloud monitoring faces problems. A cloud monitoring method has been suggested in Tovarnňák & Pitner (2012) to focus on data collected from cloud monitoring VMs. This emphasizes the basics of the data monitoring producer and has addressed data representation, storage, transmission, and delivery monitoring issues. A conceptual solution called New-Generation-MONitoring (NGMON) was introduced as a proof of concept. The data collection feature of NGMON is defined as logs, warnings, and business operations on an event-based framework and stored as a specified structure in typed data form. In the next stage, the data are encoded as an event object in JavaScript Object Notation (JSON) format. The Access Control List (ACL) is used to ensure security-based authentication. Query evaluator is used to query/react to track user details once the link has been established. The publish/subscribe program is employed for customer-specific monitoring demand. In order to present a system protocol based on TCP, a hybrid communication model was used. NGMON also provides support for Secured Socket Layer (SSL) encryption.

The sophistication of cloud systems raises problems when proffering inefficient monitoring approaches for large distributed infrastructures Montes et al. (2013) proposed a layered cloud monitoring architecture called GMonE (Global Monitoring systEm) to solve these problems. The architecture of GMonE consists of four main components: GMonEmon, GMonEDB, GMonEAccess, and monitoring plug-ins. In the cloud, GMonEMon collects and sends metric information to GMonEDB. Monitoring plug-ins in the form of applications for monitoring execution are part of GMonEMon. As a database, GMonEDB is responsible for collecting and handling GMonE monitoring data. The GMonEAccess is a user interface that allows a user to easily access the data. To prove their theory, they used OpenNebula to test GMonE on Grid'5000 [51]. Results have shown that it has limited overhead control regarding the consumption level of computing and communication services. The research suggested using a publish/subscribe model for exchanging messages. Publish/subscribe, however, faces issues, including rigid semantic relation and message transmission problems that affect the factor of trustworthiness [52].

Cloud administrators establish policy provisions based on full awareness of the infrastructure's physical resources and facilities. A robust and up-to-date cloud infrastructure awareness is challenging if multi-tenant and complex cloud middleware stacks are considered. To fix this issue, Povedano-Molina et al [15] developed the Distributed Architecture for Resource manaGement and mOnitoring in cloudS

(DARGOS). DARGOS allows the consumer to have the versatility of controlling granularity according to his/her requirements. DARGOS was introduced in Povedano-Molina et al. [15] as a project on OpenStack.

A cloud user's satisfaction level toward the negotiated Service Level Agreement (SLA) plays a critical role in cloud usage and CSP partnerships. Currently, most cloud monitoring systems devote little interest to the aforementioned parameter's user-side computation. Authors have addressed this issue in [53] by proposing a monitoring architecture called Monitoring SLA for Restful Services (MonSLAR). The work is a long-term project aiming at building MonSLAR as an actual working method. The authors predicted two control rates in the proposed architecture: the consumer one and the CSP's. MonSLAR provides users with details about the accepted level of service based on SLA, which is either reached or not. MonSLAR offers a benchmark for customer satisfaction for the service provider by collecting Quality of Experience (QoE), which the cloud service provider uses as a crucial performance monitoring metric to meet SLA. The study focused primarily on fulfilling the SLA criterion. The analysis, however, ignored the metrics information and it didn't explain why their program should determine the SLA breach based on what metrics.

Cloud monitoring applications find cloud environments where Virtual Machines (VMs) and containers are essentially not part of the cloud operating system. Containers and VMs are generated and killed in a cloud environment with remarkable amount of time. Suneja et al. (2016) proposed a brand-new Cloud monitoring paradigm called Near Field Monitoring (NFM) to deal with this problem. The monitoring agent does not interfere with or install within the host VM to offer monitoring and operational and analytical services; thus, this is a new type of tracking. Because it does not execute any additional application or resource in the VM/container, a user/host can opt-in and opt-out in NFM. Using NFM, a cloud provider can get a wide view of its resources and VMs. The authors used data from the kernel to obtain monitoring metrics. They also tested more than 1000 Linux flavors for NFM, and it performed well without any modification.

III. FEDARGOS-V1 FOR FEDERATED CLOUD INFRASTRUCTURE

DARGOS is comprised of 3 major engines: the configuration engine, the statistics engine, and the query engine. These engines operate to assure the collection, storage, and analysis of the monitoring data in order to provide a full suite of monitoring functionality. The coordination service, provided by the configuration engine, which is the foundation upon which the other services work, offers a method for the components to communicate as well as VM registration, configuration storage, and decision-making. In order to provide a monitoring architecture that satisfies the need to monitor a federated cloud infrastructure, the study at hand extends the configuration engine and adds an alarm engine, which will be in charge of handling different alerts and notifications. In addition, an interface is built to provide CSPs or cloud tenants with an easy-to-read monitoring console.The extended DARGOS is thus named FEDerated Architecture for Resource manaGement and mOnitoring in cloudS Version 1.0 (FEDARGOS-V1).

A. DARGOS CONFIGURATION, STATISTICS, AND QUERY ENGINES

1) CONFIGURATION ENGINE

A very secure, scalable, and dependable part of DARGOS, the configuration engine performs the duties of a highly distributive system to make it possible to administer the cloud infrastructure effectively and establishes the monitoring strategy. A monitoring strategy identifies the variables and events that should be watched, the tools that should be utilized, the participants, and the actions that should be performed. A monitoring strategy is thus a sociotechnical process that includes both software and human agents. The configuration engine also enabled the definition of several roles and views for different user types with various access rights to monitoring data. Additionally, it gives users the option to define metrics, which enables them to cover variables unique to a particular setting.

2) STATISTICS ENGINE

The statical engine permits assessments of the dynamics of the cloud infrastructure in typical circumstances, or when anomalies do not alter the observations. The engine calculates the total dynamics while taking into account how the infrastructure is used, displaying structural correlations, residual dynamics, and measurement errors based on predetermined criteria. The ability to view data on the usage of cloud resources on a per-domain basis, as well as monitor server health indicators, user actions, and connections, is provided via automatically generated statistics.

3) QUERY ENGINE

The query engine is a component that runs queries to respond to tenant requests on top of the configuration engine and the statistic engine. The advantage is that the query engine may be directed to the precise location of the data instead of having to transport the data to various nodes in the cloud infrastructure. The query engines offer the same capabilities as data warehouses did, but they do so using a somewhat different strategy that separates storage from computation with a focus on elasticity and scalability.

The subsequent subsections present how the components of DARGOS are extended to derive FEDARGOS-V1

B. THE CONFIGURATION ENGINE

The configuration engine is a robust and highly available pool of configurations that additionally provides agreement, storage configuration, communication paradigm, and failure detection to the other components within DARGOS. It is

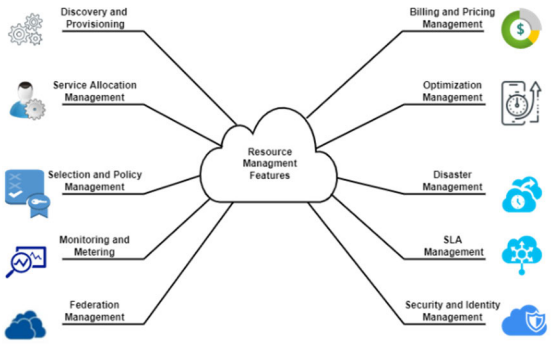


FIGURE 1. Cloud resource management taxonomy and features.

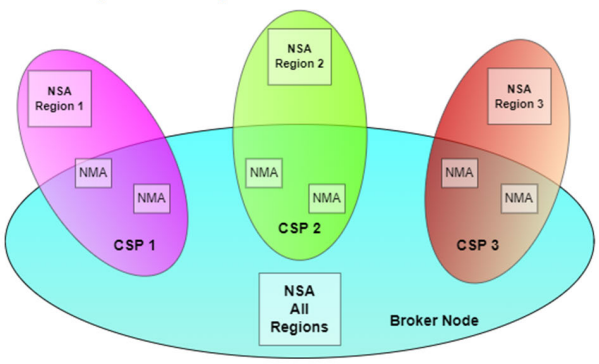


FIGURE 2. Multi-tenancy Support Behavior of NSAs and NMAs Communication Model in the Proposed FEDARGOS-V1.

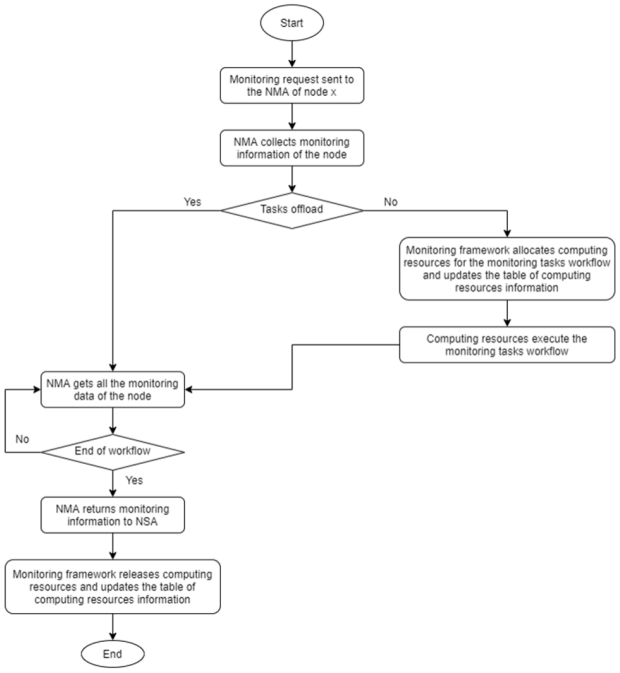


FIGURE 3. Flowchart for Monitoring Agents Communication in FEDARGOS-V1.

a self-contained service with no external dependencies, and it is designed to keep running even when there are a lot of failures. Every VM in the monitored infrastructure runs a coordinator daemon, but the purpose of each coordinator varies, ranging from participating in relevant agreements to detecting network, storage, and image failures.

Configuring and coordinating components become a challenge when developing heterogeneous systems. The use of an existing configuration engine as the foundation for a new monitoring architecture has a number of drawbacks. Firstly, such engines are frequently constructed on enormous stacks that require a plethora of dependencies, resulting in a footprint far beyond what is necessary. Secondly, the current configuration engine is intended to be used by a single cloud service provider. As a result, a configuration engine failure, loss of performance, or other issues would have an impact on important applications and the monitoring service. Thus, the monitoring tool is unavailable or impaired when it is most needed. Therefore, this study presents a customized and dedicated configuration engine that is designed to handle a wide range of failure scenarios in many nodes from various cloud service providers in the federation in order to simplify overall monitoring.

To meet the demand for multi-tenancy awareness, DARGOS, implements a publish/subscribe distributed architecture based on Node Monitor Agents (NMAs) and Node Supervisor Agents (NSAs) where NMAs are responsible for gathering monitoring data and transmitting it to interested parties, while NSAs are the user's access points to the cloud monitoring data. However, an adjustment in the placement of the NMAs and NSAs is required. Accordingly, the NSA component provides a versatile API that allows users to read local monitoring data stored in each cloud infrastructure node.

To demonstrate multi-tenancy capability in a federated cloud environment, each cloud service provider is referred to as a region. Each tenant is assigned to a certain region so that they can see the cloud in their own way. Each node (a node representing a server or VM) is associated with one NMA, but one NSA may be interested in collecting monitoring information from each node, resulting in the association of one NSA with each region. Finally, the broker node is coupled with an NSA that unifies all regions.

Figure 2 depicts the support of the multi-tenancy behavior by the configuration engine.

Figure 2 shows the communication model handled by the configuration engine in the federated monitoring architecture. Each NMA is linked to a single node. Each tenant has an NSA that receives monitoring data from multiple nodes, and the broker node has an NSA that receives monitoring data from all of the regions' nodes.

Figure 3 shows the flowchart for the process of collecting monitoring information by different NMAs and its transmission to NSAs of the interested parties.

NMAs and NSAs are implemented in the configuration engine as APIs with the goal to make the access to monitored data easier. These APIs are built to be as expressive and flexible as possible in order to easily designate the nodes the NSA wish to monitor, get the most recent measured value for a specific resource as well as its latest historical measurements,

and verify the capacities of resources automatically detected by querying about their status. Every compute service and every nova component in the federation have NMAs attached to them in order to collect data on physical node and hypervisor statistics. Additionally, the monitoring solution can make more flexible resource allocation decisions thanks to the current DARGOS-based scheduler that employs the resource statistics information gathered by the NSAs.

C. THE ALARM ENGINE

Alarms are typically used to notify users of anomalous processes in an infrastructure's operational procedures. They can take the form of both audio and visual announcements. Notifications are often triggered when a specific process variable (measured by a dedicated sensor) surpasses a predetermined limit or threshold. The alarm engine within the monitoring architecture ensures alarming capabilities, and it requires maintenance to maintain its optimal performance and full functionality.

The alarm service in a monitoring architecture seeks to provide a service that allows triggering actions based on defined rules to be applied to monitor metrics or event data acquired by monitoring agents. The alarm service's responsibilities include triggering alarms when gathered metering or event data violates set rules, as well as inducing actions of the notification handler, which selects when to fire alarms and when to notify cloud tenants. The alarm Engine is made of three major components, the Alarm Database, the Alarm Generator, and the Notification Listener:

i. Alarm Database: The Alarms Database component is a database that maintains a set of data that describes an alarm raised by the fault detection mechanism. The alarms database includes a finite number of groups by which each alarm is categorized, in addition to keeping track of the fault detection mechanism's working history. The alarm engine searches the alarms database for a similar alarm category before issuing an alarm to the appropriate parties. This is important for reducing the number of messages sent because multiple alerts can be substituted with a single alarm that has a stronger impact.
ii. Alarm Generator: The Alarms Generator is notably made up of the Alarm Evaluator which determines the severity of an alarm and the Alarm Notifier which is responsible for sending notification to cloud tenants and CSPs. Alarms are set using the alerting policy. A cloud administrator's alerting policy specifies the circumstances in which he or she wants to be warned and how he or she wants to be alerted. Metric-based alerting policies or log-based alerting policies are two types of alerting policies that are used to track the metric data acquired by the monitoring architecture [54]. However, metric-based alerting policies are used in this work.
iii. Notification Listener: The Notification Listener is responsible for polling monitoring data on a regular basis and sending it to the monitoring architecture's NMA for processing, including converting messages to events.

The alerting policy provides the ability to define criteria and conditions, and these criteria are based on metrics. An alerting policy condition can keep track of things like when a metric reaches a certain value (threshold) or when it starts to change rapidly. Metrics are linked to resources and measure some aspect of that resource, such as average CPU utilization across instances, the VMs in use, the overall storage, the memory, the swap memory, and the networking.

The alarm engine can detect and provide different alarms/alerts mechanisms including i) Anomaly Alarm, ii) Intrusion Alarm, iii) Billing Alert, iv) SLA Alert, and etc.

The process for calculating the alarm severity level is described in Algorithm 1. The Alarm Generator refers to the Alarms Database to determine the severity level. A cloud tenant of the federated cloud environment may interact with the Alarm Generator to affect the process of determining the alarm severity level. The tenant can estimate the significance of an attack in comparison to others based on their comprehension by increasing the initial severity level. For example, a cloud tenant may adjust the algorithm to favor a DoS attack over a Portsweep assault.

Algorithm 1 Alarm Severity Level Detection in FEDARGOS-V1

Input: nTraffic, opK[]
Output: severity[]

```
Start
  function idsFrame(nTraffic)
      while (nTraffic >= 1) do
        frame = [identity_of_Frame, frame_element]
      end
      return frame
  end function
  function getSeverity (nTraffic, opK[])
      var currentFrame = idsFrame (nTraffic)
      for each element e in currentFrame do
          if (hasAlarm(e)) then
              var vType = classify (e)
              var nAlarm [vType] ++
          end if
      end for
      for each element e in nAlarm [] do
          var severity [e] = k(nAlarm [e], opK [e])
      end for
      return severity []
  end function
End
```

The first step in Algorithm 1 is to examine network traffic (nTraffic) within a current frame provided by the fault detection system. This function looks for faults, classifies them

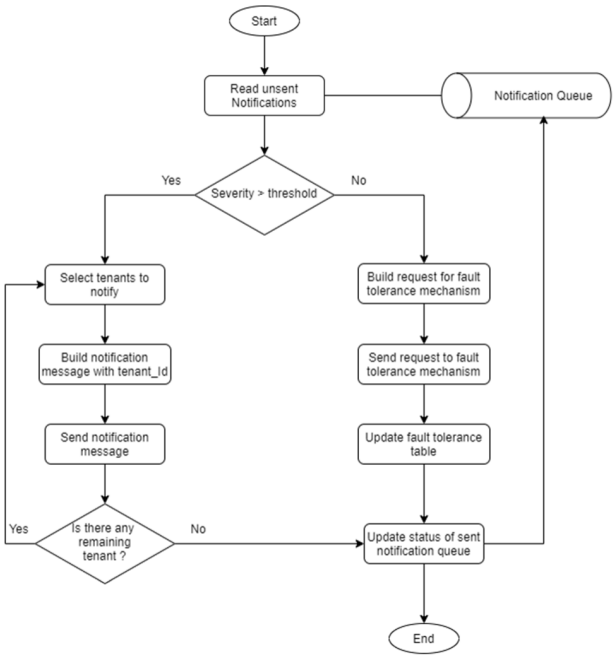


FIGURE 4. Notification Handling Flowchart for FEDARGOS-V1.

using the Alarms Database, and assigns a severity rating. Based on the number of alarms and the operator's knowledge, the severity function computes the severity of a fault. The severity function is based on a regression model derived from network traffic and operator knowledge (opK) patterns [55]. Thus, the severity is obtained by equation (1).

$$\begin{aligned} \text{severity} &= \text{k(nAlarm[}e\text{], opK[e])} \\ &= nAlarm_0 * opK\ [e] + nAlarm_1 * opK\ [e] \\ &\quad + \ldots + nAlarm_e * opK\ [e] \end{aligned} \tag{1}$$

The severity level for various types of alarms is set by the cloud administrator. This stops cloud tenants from receiving redundant alarms. In the process of creating alarms, the system administrator calls the severity function while setting up the different thresholds.

The alarm engine relies on the metrics feature, which is a gauge that relates to a resource's health, capability, or performance. The NMA receives metrics from resources, services, and applications. Moreover, when metrics match alarm-specified triggers, the monitoring architecture's (FEDARGOS-V1) alarm functionality works with the notification service to notify the tenants. Figure 4 shows the flowchart of the notification handling process of the monitoring architecture.

Alarms are set using the Monitoring Query Language (MQL) expressions. A threshold, statistics, interval, and trigger rule must all be specified in an alarm query. The administrator transmits Alarm queries to the cloud environment through alarm creation APIs. The parameters of an alarm creation API include the project to which the alarm is related, the user responsible for the alarm, the VM to which the alarm is related, the time constraint, which states the interval of time

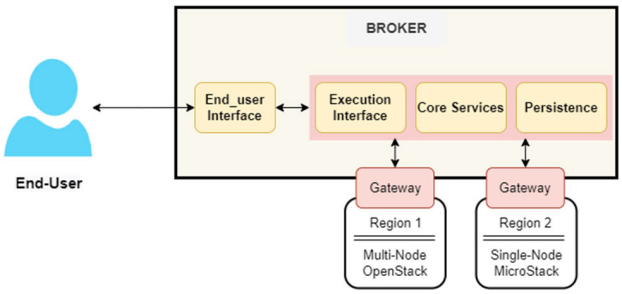


FIGURE 5. The broker based federation architecture for FEDGEN Testbed Regions.

it takes for the alarms to be returned, the maximum number of items to be returned, and the type of alarm. The monitoring architecture alarm feature interprets the responses to monitoring queries. These responses are in the form of Boolean values, with zero indicating false and non-zero indicating true. When a true value is returned, it means that the trigger rule constraint has been met.

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

This section focuses on the implementation specifics of FEDARGOS-V1. First the testbed setup is presented followed by an overview of a created Web-based console tool for Federated Cloud monitoring. Finally, experimental results and the evaluation of the performances of FEDARGOS-V1 as well as its benchmarking with existing architectures are presented. The existing monitoring architectures such as Nagios and DARGOS for standalone cloud infrastructure are selected for the benchmark of the developed FEDARGOS-V1 whilst the parameters including the Response Time, the Network Traffic Flow, and the Scalability Factor are used to establish the performance results.

A. THE FEDGEN TESTBED

The FEDGEN testbed [58] makes use of a Broker to manage resource negotiations between end-users and CSPs within the federated cloud environment. The testbed is built with OpenStack middleware. Two regions form the FEDGEN testbed, a multi node cloud deployment and a single node cloud deployment. The minimum requirements to build a multi node cloud infrastructure are 6 Bare Metal servers, and a network switch. Whereas, the minimum requirements to build a single node cloud infrastructure include a single Bare Metal server with internet connectivity.

The Broker acts as the federation's main door and is in charge of allocating the resources present in the heterogeneous pool of resources in a coordinated manner. There are gateways between the Broker and different cloud regions. Gateways are proxy processors that forward requests from the broker to the cloud regions and convert resource requests into commands that the cloud regions can comprehend. This is depicted in Figure 5.

There are four tiers to the Cloud Broker node in the architecture of the FEDGEN testbed:

i. **End-user Interface**: This tier is responsible for the reception of services requirements. It turns them into execution terms for resources that can address these needs. It is also in charge of the management of service access credentials.
ii. **Core Services**: This layer manages a set of services responsible for selecting the best services for end-user application processes, discovering new services, monitoring services already contracted, and negotiating new services if an application requires to be relocated because the current CSP can no longer offer SLA requirements.
iii. **Execution Interface**: This tier communicates with the load balancer in the federated cloud, which dispatches the application, monitors its execution, and returns the status to the end-user satisfaction.
iv. **Persistence Tier**: It handles the database that is utilized to preserve the Cloud Broker state if there are any failures.

The various hardware that constitutes the FEDGEN Testbed Region 1 includes 6 DELL R620 Bare Metal servers, two racks, and a network switch. The Bare Metal servers make the nodes of the cloud infrastructure. The hardware specification of the Bare Metal server is as follows: Intel®Xeon®E5-2620 12 cores @ 2GHz processor, 8GB RAM, 299GB (HDD) operating system storage, 1000GB (HDD) data storage, and four Network Interface Controller (NIC). The switch used is a 48 ports Catalyst 2960 series. On each of the servers, Ubuntu 18.04 LTS (server version) was used as the operating system.

The FEDGEN testbed Region 2 is deployed on a single node using MicroStack. This is just one control node. It comes with everything needed, including the ability to operate as a compute node. The hardware specifications of the unique node are: An Intel®4 Core^{TM}i5-2400 @ 3.10 GHz processor, 8GB RAM, and 1.0TB of disk capacity. Ubuntu 20.04.3 LTS (desktop version) was used as the operating system on the unique node.

B. DEPLOYMENT OF FEDARGOS-V1

Figure 6 depicts the block diagram of the deployment of the developed monitoring architecture. It shows different blocks and components of the monitoring solution alongside the architecture of the testbed on which the monitoring architecture is deployed.

The architecture developed in this work uses FEDARGOS-V1 APIs to provide access to monitoring information to tenants within the federated cloud environment. However, allowing open access to publish/subscribe infrastructure from outside the federated Cloud might cause security and scalability issues, and it necessitates significant DARGOS and Python technical expertise. As a result, FEDARGOS-V1 provides a set of Representational State Transfer (REST) APIs intended at providing access to monitoring data and, as a matter of fact, increasing the monitoring architecture's interoperability with Web standards.

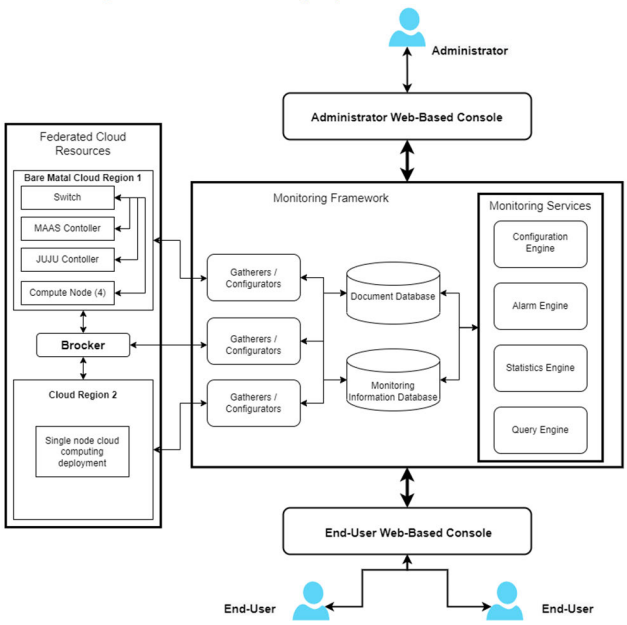


FIGURE 6. FEDARGOS-V1 deployment.

FIGURE 7. FEDARGOS-V1 JSON API response example.

For example, a JSON object containing the most recent resource monitoring information of all the instances currently monitored by the NSA installed at the machine 172.16.60.22 can be obtained by simply entering the http://172.16.60.22:8774/v2.1/servers/ URL. Figure 7 depicts an example of FEDARGOS-V1 JSON encoded monitoring information.

Although the JSON communication format is human-readable, it is difficult for users to comprehend and often requires further processing to create user-friendly displays of monitored information. To address this problem, the FEDARGOS-V1 includes a Web-based console that displays collected monitoring information in a user-friendly and easy-to-read format. FEDARGOS-V1 gives important information about all of the apps running within the federated cloud architecture. Relying on OpenStack middleware in this work, the major applications that are monitored include networking (Neutron), storage (Cinder), compute (Nova), image service (Glance), and the telemetry for alerts and notifications of federated cloud tenants when an issue occurs. FEDARGOS-V1 guarantee that all relevant information about the federated cloud deployment is monitored, including performance and availability statistics. Figure 8 shows the overview of the

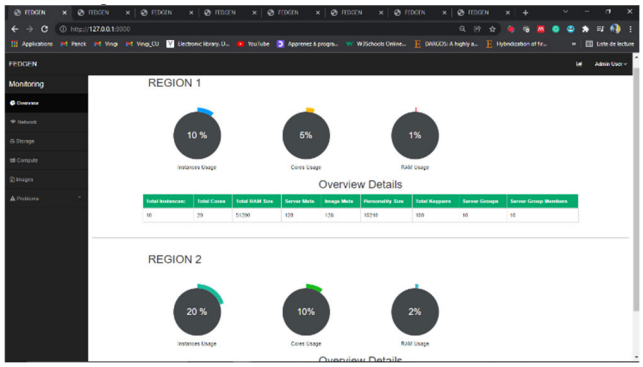


FIGURE 8. Screenshot of FEDARGOS-V1 monitoring web-based console.

monitored resources of all the regions of the FEDGEN testbed cloud environment.

The benefits of using a Web console application are numerous, but the following are the most important. (i) It enables monitoring data information to be accessed from beyond the data center. (ii) Requests can be sent to various NSAs, making it easier to monitor data across multiple Clouds and hosts. iii) It also allows each tenant in the federation to define specific and numerous views of the same deployment.

C. EXPERIMENTAL RESULTS

This section presents the evaluation of the performances of the FEDARGOS-V1 as well as its benchmarking with existing architectures. The existing monitoring architectures such as Nagios and DARGOS for standalone cloud infrastructure are selected for the benchmark of the developed FEDARGOS-V1 whilst the parameters including the Response Time, the Network Traffic Flow, and the Scalability Factor are used to establish the performance results.

1) TECHNICAL EVALUATION WITH THE BENCHMARKED ARCHITECTURES

Nagios and DARGOS for the single cloud are chosen for comparison with FEDARGOS-V1 among the open-source cloud-aware monitoring solutions described in section II C. Because of its simplicity and extensibility, Nagios is chosen as a representative example of a general-purpose data center monitoring system based on plug-ins. On the other hand, the monitoring architecture proposed in this work extends DARGOS, hence its selection.

Nagios is used to track host statuses via connection and ping delays, as well as the status of services via HTTP. It offers resource monitoring by enabling pull interactions through the Nagios Remote Plugin Executor (NRPE) extension protocols. For push interactions, it also supports the Nagios Service Check Acceptor (NSCA) methods.

Table 2 displays a qualitative side-by-side technical comparison of the most important aspects of the chosen monitoring architectures.

Note: N:M (many-to-many scenario) stands for multiple sources, multiple targets, and 1:M (many-to-one scenario) stands for multiple sources, single target.

TABLE 1. Open-source cloud management platforms.

CMP	Participant	Reference
OpenStack	800 +	[21] [22]
CloudStack	200 +	[23]
OpenNebula	100 +	[24]
Eucalyptus	100 +	[25]

TABLE 2. FEDARGOS-V1, DARGOS, and nagios qualitative comparison.

	FEDARGOS-V1	DARGOS	Nagios
Architectural model	Distributed	Distributed	Centralized
Asynchronous event notification	Yes	Application-level	Yes
Communication cardinality	N:M	N:M	1:M
Communication model	Push	Push	Push (NSCA) or pull (NRPE)
Coupling	Yes	Yes	Yes
Data exchange format	Common Data Representation	Common Data Representation	Plain text
Discovery method	Automatic	Automatic	Configuration files
Filtering support	Time, content- and network-based	Time and content-based	dependent on the plugin implementation
Historic data	Yes	Yes	Yes
Metadata	Out band	Out and in-band	In-band
Network transport	User Datagram Protocol (UDP)	User Datagram Protocol (UDP)	Transmission Control Protocol (TCP)
Notification strategy	Periodic, real-time, and event-based	Periodic and event-based	dependent on the plugin implementation
QoS support	Yes	Yes	No
Transport protocol	Real-time Publish-Subscribe Protocol (RTPS2) DDS	Real-time Publish-Subscribe Protocol (RTPS2) DDS	Proprietary (NSCA and NRPE)

Nagios is built on a centralized architecture. The monitoring functionalities are stored and processed by a single central node in this architecture. Moreover, because of its centralized structure, Nagios is the only viable option in many-to-one cases. DARGOS and FEDARGOS-V1, on the other hand, offer many-to-many communications by default, making them monitoring solutions that are ideal for entirely distributed and decentralized settings. In terms of interaction models, Nagios NRPE is the only one that employs a pull strategy, although this results in longer latency when retrieving monitoring data because it necessitates both a request and a response.

The discovering of nodes and hosts is another key feature. Automatic discovery is supported by DARGOS and FEDARGOS-V1, but Nagios requires configuration files. Because of this constraint, Nagios can only be used in static scenarios.

Another key element is the data flow between monitored nodes and clients: this should be proficient and, to the extent possible, rely on defined formats. In this regard, DARGOS and FEDARGOS-V1 use Common Data Representation (CDR) [56] for transmitting data, whereas Nagios uses plain text. Moreover, DARGOS and FEDARGOS-V1 employ the RTPS OMG standard to exchange data, whereas Nagios uses proprietary protocols. Because of its RTPS protocol conformance, DARGOS is straightforward to integrate with third-party DDS-based applications. Another distinction is that Nagios communicates data between monitored nodes and the central server via the TCP protocol, which increases latency (due to TCP link establishment, fault, congestion, and flow control) when compared to DARGOS and FEDARGOS-V1, which use the UDP connectionless protocol.

The metadata that identifies the attributes and types of each monitored sensor is likewise handled differently. Since Nagios sends information in line with each data sample, the server is responsible for properly processing the properties and values. DARGOS has the option of transmitting metadata with the sample or not; however, if no metadata is given, the application must create an additional method to assist DARGOS users in filling in the properties for each sensor. FEDARGOS-V1 only shares sensor metadata during the first discovery phase, saving bandwidth, especially when there are a lot of sensors and samples.

Various filtering and updating algorithms can be implemented by monitoring architectures to save bandwidth and resources. The most common way to reduce bandwidth is to send measurement updates only when particular events occur rather than on a regular basis. While DARGOS and FEDARGOS-V1 both provide this feature, Nagios does not. In Nagios' case, that characteristic is left at the application level. Another contrast between DARGOS and FEDARGOS-V1 is that the latter allows each user to design their own data filters. In other words, each FEDARGOS-V1 subscriber can set their own update frequency, whereas, in DARGOS, this rate is set universally. As a result, it is now easier to install heterogeneous applications with varying needs, such as multi-tenant clouds, in FEDARGOS-V1.

The capacity to transmit and handle asynchronous event notifications (including also alarms, failures, etc.) is the final point to consider. This functionality can be used to activate a synchronization mechanism in the event of a failure, including detecting aberrant events sending administrator notifications. Asynchronous event notifications are supported byFEDARGOS-V1, DARGOS, and Nagios, although FEDARGOS-V1 focuses on processing them at the alarm engine level.

2) EXPERIMENTAL PERFORMANCE RESULTS

Three series of experimentations were carried out to thoroughly assess FEDARGOS-V1 abilities and performance. First, FEDARGOS-V1 response time is assessed and compared to the DARGOS response time. The FEDARGOS-V1 network traffic flow is then examined, and a comparison with the reference DARGOS implemented for a regular OpenStack single cloud network utilization is made. Finally, FEDARGOS-V1, DARGOS, and Nagios scalability factors are assessed.

a: RESPONSE TIME

The response time is the time it takes for a request to reach its destination and for a response to be received. All monitoring tasks comprise a request and a response. In FEDARGOS-V1, requests are sent from NSAs to NMAs. The response time is thereby given by the time it takes for an interested NSA to send a request to receive monitoring data and get a response. The different delays, propagation time, and processing times enter into account to determine the overall response time. FEDARGOS-V1 utilizes UDP standards for communication between NSAs and NMAs. Thus, within FEDARGOS-V1, the minimum time a monitoring request can take is given by Equation (2) [57]:

$$\begin{aligned} minTime = {} & min\ Frame\ SerializationTime \\ & + Link\ Media\ Delay + Queueing\ Delay \\ & + Node\ Processing\ Delay \end{aligned} \tag{2}$$

With:

$$FrameSerializationTime = S/R \tag{3}$$

$$LinkMediaDelay = D/p \tag{4}$$

$$QueueingDelay = Q/R \tag{5}$$

where R is the link data rate (in bits/second), S the packet size (in bits), D the link distance (in meters), p the medium propagation speed (in meters/second), and Q the queue depth.

The *NodeProcessingDelay* is normally specified by the configuration of the infrastructure.

The maximum time is as well given by (6) [57]:

$$\begin{aligned} maxTime = {} & max\ Frame\ SerializationTime \\ & + Link\ Media\ Delay \\ & + Queueing\ Delay \\ & + Node\ Processing\ Delay \end{aligned} \tag{6}$$

Practically, using Wireshark testing tool, considering the FEDGEN testbed, the *Node ProcessingDelay*$= 0$, the *Queueing Delay*$= 0$ because there is no congestion, the *Link Media Delay*$= 0.04$ *seconds*. The minimal packet size of the monitoring task in FEDARGOS-V1 is

$$S\_min = 4.11KB$$

The maximal packet size of the monitoring task in the FEDARGOS-V1 is

$$S\_max = 794.751KB$$

The link data rate $R = 2{*}10^2 bps$

$$\text{The } minTime = \left(\frac{4.11}{2{*}10^2}\right) + 0.04$$

$$= 0.06055 seconds$$

$$\text{The } maxTime = \left(\frac{3794.751}{2*10^2}\right) + 0.04$$

$$= 19.013755 \text{ } seconds$$

$$\text{The latency } L \simeq \frac{MaxTime - MinTime}{Number of Requests}$$

$$= \frac{19.013755 - 0.06055}{30}$$

$$= 0.6317735 \text{ } seconds$$

The response time of monitoring tasks within FEDARGOS-V1 deployed on the FEDGEN testbed is of the order of 631.7735 milliseconds.

b: NETWORK TRAFFIC FLOW

The experiments conducted here quantifies the impact of FEDARGOS-V1 on the FEDGEN testbed cloud infrastructure's network use. The network traffic generated for monitoring the federated cloud nodes is being measured in these experiments. A comparison of the FEDARGOS-V1 traffic with the reference DARGOS is conducted in particular.

The Advanced Message Queuing Protocol (AMQP) standards for exchanging messages and RPC requests/responses amongst all OpenStack services involved in monitoring give statistics such as users, the number of operating VMs with their networking-related data. These data are analyzed to have a better understanding of the monitoring architecture's network traffic flow.

Within FEDARGOS-V1, one NSA is placed at the broker node level and at least one NMA to collect monitoring data at each OpenStack Compute node. The monitoring strategies are both event-based (in the sense that the CPU usage thresholds are [0, 25, 50, 60, 70, 75, 80, 85, 90, 95, 100] percent) and periodic (with the period of 1 second). The amount of computational load is determined by the VM's instantiation. At the time of VM startup, each VM bears a pure computational capacity that varies between 0% and 60%. Requests are placed according to a Poisson distribution with a maximum inter-arrival period of 4 seconds. Every analysis lasted 60 seconds and was carried out for a total of ten times. Using Wireshark testing tool, considering 10 phases of the signal cycle, and a traffic flow ratio of 46.879415 for event-based reporting and 73.6774 KB for periodic reporting, and the loss time of 0.1774 constant for all the phases of the monitoring task. The acquired data are depicted in Figure 9 and 10 on a logarithmic scale for easier reading. It shows that for both periodic and event-based reporting, FEDARGOS-V1 outperforms the reference DARGOS implementation. FEDARGOS-V1 deployed in a federated environment, in particular, generates about slightly below the traffic of the reference DARGOS deployed in a single cloud.

c: SCALABILITY FACTOR

Nagios monitoring tool was deployed on the FEDGEN testbed. This solution was able to collect monitoring data of the broker node and, to an extent, monitoring data of

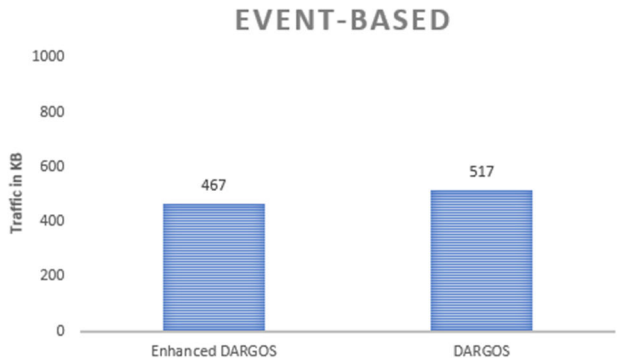


FIGURE 9. Generated Traffic per Protocol for Event-based Reporting in the FEDGEN Testbed for FEDARGOS-V1 (i.e. Enhanced DARGOS) and DARGOS.

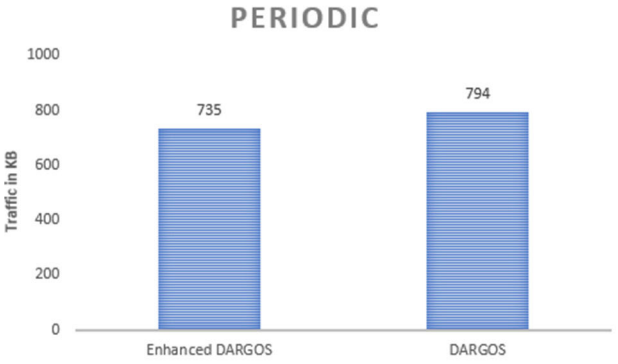


FIGURE 10. Generated traffic per protocol for periodic reporting in the FEDGEN Testbed for FEDARGOS-V1 (i.e Enhanced DARGOS) and DARGOS.

one region of the federated cloud infrastructure. Requests sent to the second region of the federated cloud were lost, or the nodes of the second region were not reachable to Nagios. Thus, Nagios is limited to monitoring a single cloud environment. Furthermore, every time a new node is added to the infrastructure, all the Nagios configuration files need to be revisited. After determining that FEDARGOS-V1 and DARGOS are the best prospects for high scalability when compared to Nagios, this stage of the experiment compares these two systems in a real-world implementation. With this in mind, a new compute node was added to the FEDGEN infrastructure, and each federation region sends five updates per second to the monitoring architecture, which is higher than the standard update rate in the periodic context. Furthermore, the same experiment was carried out with both DARGOS in a single cloud and the FEDARGOS-V1 deployed on the FEDGEN testbed for this evaluation. To determine the average global throughput generated in each host, both systems were run for 60 seconds and these were repeated ten times. For ease of presentation, Figure 11 displays the collected data (overall network bandwidth in Bytes per second) on a logarithmic scale. The results (the greatest throughput achievable) are presented for 12 instances with different nodes linked via 100 Mbps Ethernet.

The scalability factor of the FEDARGOS-V1 is given by:

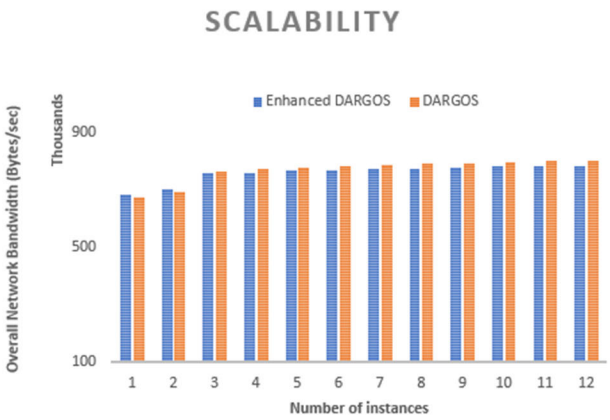


FIGURE 11. Results of FEDARGOS-V1 (i.e. Enhanced DARGOS) and DARGOS scalability.

With

$$A = \frac{\sum_{k=1}^{n} (D_k - D_0) * (I_k + I_0)}{2} = 5929$$

and

$$A^* = \frac{\sum_{k=1}^{n} (D_k - D_0) * \left(I_k^* + I_0^*\right)}{2} = 25433$$

$$\eta_I = \frac{A}{A^*} = 0.2331$$

The scalability factor of DARGOS in a single cloud is given by:

With

$$A = \frac{\sum_{k=1}^{n} (D_k - D_0) * (I_k + I_0)}{2} = 6463$$

and

$$A^* = \frac{\sum_{k=1}^{n} (D_k - D_0) * \left(I_k^* + I_0^*\right)}{2} = 23752$$

$$\eta_I = \frac{A}{A^*} = 0.2721$$

where A and $A^*$are the ideal scalability factors calculated for the default number of instances to be allocated to service demand and the actual (increased) number of instances allocated to the new service demand.

And D_k and D_0 are two service demand volumes with $D_k > D_0$ and I_k and I_0 are the corresponding amount of instance resources that are deployed to deliver the required services. The amount of instance resources is obtained by computation of the following:

$$\frac{D^*}{D} = \frac{I^*}{I} \; thus \; I^* = \frac{D^*}{D} * I$$

Hence, both the FEDARGOS-V1 and the reference DARGOS are able to monitor the additional node in the infrastructure. The overall throughput of the FEDARGOS-V1 increases by 23%, while the overall throughput of DARGOS increases by 27%, which is comparable to FEDARGOS-V1 throughput.

V. CONCLUSION

This research work was carried out with the aim of developing an enhanced resource monitoring architecture for federated cloud infrastructure named FEDARGOS. The enhanced monitoring architecture extends DARGOS due to its potentials that make it one of the best candidates to satisfy the needs for monitoring of a federated cloud infrastructure characterized by multi-tenancy support. Two major components were touched to extend the reference DARGOS, and these components include the configuration engine, the alarm engine, and finally, a suitable web-based monitoring console. FEDARGOS-V1 was deployed in the FEDGEN testbed cloud infrastructure for its testing and performance evaluation. The evaluation of the developed enhanced DARGOS (i.e. FEDARGOS-V1) focused on the following metrics: the overall response time of the monitoring task, the network traffic flow for a cycle of a monitoring request and its equivalent response, and the scalability factor. The performance of FEDARGOS-V1 was satisfactory compared to the reference DARGOS. In the future, we hope to incorporate intrusion detection, billing and SLA monitoring capabilities in the FEDARGOS-V1.

CONFLICT OF INTEREST STATEMENT

We declare that none of the authors has conflict of interest with respect to the publication of this article.

REFERENCES

[1] A. Ahmad, A. S. Alzahrani, N. Ahmed, and T. Ahsan, "A delegation model for SDN-driven federated cloud," *Alexandria Eng. J.*, vol. 59, no. 5, pp. 3653–3663, Oct. 2020, doi: 10.1016/j.aej.2020.06.018.
[2] D. C. Marinescu, *Cloud Computing*, 2nd ed. Cambridge MA, USA: Elsevier, 2018, doi: 10.1016/C2016-0-02364-1.
[3] M. Liaqat, V. Chang, A. Gani, S. H. A. Hamid, M. Toseef, U. Shoaib, and R. L. Ali, "Federated cloud resource management: Review and discussion," *J. Netw. Comput. Appl.*, vol. 77, pp. 87–105, Jan. 2017, doi: 10.1016/j.jnca.2016.10.008.
[4] A. Banijamali, O.-P. Pakanen, P. Kuvaja, and M. Oivo, "Software architectures of the convergence of cloud computing and the Internet of Things: A systematic literature review," *Inf. Softw. Technol.*, vol. 122, Jun. 2020, Art. no. 106271, doi: 10.1016/j.infsof.2020.106271.
[5] Y. Chen, "IoT, cloud, big data and AI in interdisciplinary domains," *Simul. Model. Pract. Theory*, vol. 102, Jul. 2020, Art. no. 102070, doi: 10.1016/j.simpat.2020.102070.
[6] H. Kurdi, A. Alfaries, A. Al-Anazi, S. Alkharji, M. Addegaither, L. Altoaimy, and S. H. Ahmed, "A lightweight trust management algorithm based on subjective logic for interconnected cloud computing environments," *J. Supercomput.*, vol. 75, no. 7, pp. 3534–3554, Jul. 2019, doi: 10.1007/s11227-018-2669-y.
[7] R. Kumar and R. Goyal, "On cloud security requirements, threats, vulnerabilities and countermeasures: A survey," *Comput. Sci. Rev.*, vol. 33, pp. 1–48, Aug. 2019, doi: 10.1016/j.cosrev.2019.05.002.
[8] S. Suneja, C. Isci, R. Koller, and E. De Lara, "Touchless and always-on cloud analytics as a service," *IBM J. Res. Dev.*, vol. 60, nos. 2–3, pp. 11:1–11:10, Mar. 2016, doi: 10.1147/JRD.2016.2518438.
[9] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, "Comparison of open-source cloud management platforms: OpenStack and OpenNebula," in *Proc. 9th Int. Conf. Fuzzy Syst. Knowl. Discovery*, May 2012, pp. 2457–2461, doi: 10.1109/FSKD.2012.6234218.
[10] B. König, J. M. A. Calero, and J. Kirschnick, "Elastic monitoring framework for cloud infrastructures," *IET Commun.*, vol. 6, no. 10, pp. 1306–1315, Jul. 2012, doi: 10.1049/iet-com.2011.0200.
[11] S. Clayman, G. Toffetti, A. Galis, and C. Chapman, "Monitoring services in a federated cloud," in *Achieving Federated and Self-Manageable Cloud Infrastructures*. Hershey, PA, USA: IGI Global, 2012, pp. 242–265, doi: 10.4018/978-1-4666-1631-8.ch013.

[12] P. K. Paul and M. K. Ghose, "Cloud computing: Possibilities, challenges and opportunities with special reference to its emerging need in the academic and working area of information science," *Proc. Eng.*, vol. 38, pp. 2222–2227, Jan. 2012, doi: 10.1016/j.proeng.2012.06.267.

[13] H. J. Syed, A. Gani, R. W. Ahmad, M. K. Khan, and A. I. A. Ahmed, "Cloud monitoring: A review, taxonomy, and open research issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 11–26, Nov. 2017, doi: 10.1016/j.jnca.2017.08.021.

[14] W. Tian, M. Xu, A. Chen, G. Li, X. Wang, and Y. Chen, "Open-source simulators for cloud computing: Comparative study and challenging issues," *Simul. Model. Pract. Theory*, vol. 58, pp. 239–254, Nov. 2015, doi: 10.1016/j.simpat.2015.06.002.

[15] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 8, pp. 2041–2056, Oct. 2013, doi: 10.1016/j.future.2013.04.022.

[16] S. Logesswari, S. Jayanthi, D. KalaiSelvi, S. Muthusundari, and V. Aswin, "WITHDRAWN: A study on cloud computing challenges and its mitigations," *Mater. Today, Proc.*, pp. 2214–7853, 2020, doi: 10.1016/j.matpr.2020.10.655.

[17] M. Chiregi and N. J. Navimipour, "Cloud computing and trust evaluation: A systematic literature review of the state-of-the-art mechanisms," *J. Electr. Syst. Inf. Technol.*, vol. 5, no. 3, pp. 608–622, Dec. 2018, doi: 10.1016/j.jesit.2017.09.001.

[18] F. K. Parast, C. Sindhav, S. Nikam, H. I. Yekta, K. B. Kent, and S. Hakak, "Cloud computing security: A survey of service-based models," *Comput. Secur.*, vol. 114, Mar. 2022, Art. no. 102580, doi: 10.1016/J.COSE.2021.102580.

[19] M. Ghobaei-Arani, S. Jabbehdari, and M. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," *Future Gener. Comput. Syst.*, vol. 78, no. 1, pp. 191–210, 2018, doi: 10.1016/J.FUTURE.2017.02.022.

[20] W. Huang, A. Ganjali, B. H. Kim, S. Oh, and D. Lie, "The state of public infrastructure-as-a-service cloud security," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–31, Jul. 2015, doi: 10.1145/2767181.

[21] M. Pyati, D. G. Narayan, and S. Kengond, "Energy-efficient and dynamic consolidation of virtual machines in OpenStack-based private cloud," *Proc. Comput. Sci.*, vol. 171, pp. 2343–2352, Jan. 2020, doi: 10.1016/j.procs.2020.04.254.

[22] J. P. Mullerikkal and Y. Sastri, "A comparative study of OpenStack and CloudStack," in *Proc. 5th Int. Conf. Adv. Comput. Commun. (ICACC)*, Sep. 2015, pp. 81–84, doi: 10.1109/ICACC.2015.110.

[23] D. Freet, R. Agrawal, J. J. Walker, and Y. Badr, "Open source cloud management platforms and hypervisor technologies: A review and comparison," in *Proc. SoutheastCon*, Mar. 2016, pp. 1–8, doi: 10.1109/SECON.2016.7506698.

[24] F. Marozzo, "Infrastructures for high-performance computing: Cloud infrastructures," in *Encyclopedia of Bioinformatics and Computational Biology*, vols. 1–3. Amsterdam, The Netherlands: Elsevier, 2019, pp. 240–246, doi: 10.1016/B978-0-12-809633-8.20374-9.

[25] S. Ismaeel and A. Miri, "A universal unit for measuring clouds," in *Proc. IEEE Canada Int. Humanitarian Technol. Conf. (IHTC)*, May 2015, pp. 1–4, doi: 10.1109/IHTC.2015.7238044.

[26] S. S. Chauhan, E. S. Pilli, R. C. Joshi, G. Singh, and M. C. Govil, "Brokering in interconnected cloud computing environments: A survey," *J. Parallel Distrib. Comput.*, vol. 133, pp. 193–209, Nov. 2019, doi: 10.1016/j.jpdc.2018.08.001.

[27] M. M. Hassan, B. Song, and E.-N. Huh, "A market-oriented dynamic collaborative cloud services platform," *Ann. Telecommun. Annales des téléCommunications*, vol. 65, nos. 11–12, pp. 669–688, Dec. 2010, doi: 10.1007/S12243-010-0184-0.

[28] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Mu, and G. Tofetti, "Reservoir–when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, Mar. 2011, doi: 10.1109/MC.2011.64.

[29] R. N. Calheiros, A. N. Toosi, C. Vecchiola, and R. Buyya, "A coordinator for scaling elastic applications across multiple clouds," *Future Gener. Comput. Syst.*, vol. 28, no. 8, pp. 1350–1362, Oct. 2012, doi: 10.1016/j.future.2012.03.010.

[30] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, vol. 6081, 2010, pp. 13–31, doi: 10.1007/978-3-642-13119-6_2.

[31] D. Bernste and D. Vij, "Intercloud directory and exchange protocol detail using XMPP and RDF," in *Proc. 6th World Congr. Services*, Jul. 2010, pp. 431–438, doi: 10.1109/SERVICES.2010.131.

[32] G. Zangara, D. Terrana, P. P. Corso, M. Ughetti, and G. Montalbano, "A cloud federation architecture," in *Proc. 10th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. (3PGCIC)*, Nov. 2015, pp. 498–503, doi: 10.1109/3PGCIC.2015.183.

[33] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, and G. Righetti, "Cloud federations in contrail," in *Proc. Eur. Conf. Parallel Process.*, vol. 7155. Cham, Switzerland: Springer, 2012, pp. 159–168, doi: 10.1007/978-3-642-29737-3_19.

[34] J. Jensen, M. Coppola, P. Dazzi, A. Lazouski, F. Martinelli, P. Mori, I. Johnson, and P. Kershaw, "The CONTRAIL approach to cloud federations," in *Proc. Int. Symp. Grids Clouds (ISGC) PoS(ISGC)*, Aug. 2012, p. 1, doi: 10.22323/1.153.0019.

[35] R. M. Badia, "Demonstration of the OPTIMIS toolkit for cloud service provisioning," in *Proc. Eur. Conf. Service-Based Internet*, vol. 6994, 2011, pp. 331–333, doi: 10.1007/978-3-642-24755-2_40.

[36] A. C. Marosi, G. Kecskemeti, A. Kertusz, and P. Kacsuk, "FCM: An architecture for integrating IaaS cloud systems," in *Proc. CLOUD Comput. 2nd Int. Conf. Cloud Comput., GRIDs, Virtualization*, 2011, pp. 7–12. Accessed: Jul. 15, 2021. [Online]. Available: http://www.thinkmind.org/index.php?view=article&articleid=cloud_computing_2011_1_20_20064

[37] D. Petcu, B. Martino, S. Venticinque, M. Rak, T. Máhr, G. Lopez, F. Brito, R. Cossu, M. Stopar, S. Šperka, and V. Stankovski, "Experiences in building a mOSAIC of clouds," *J. Cloud Comput., Adv., Syst. Appl.*, vol. 2, no. 1, p. 12, 2013, doi: 10.1186/2192-113X-2-12.

[38] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski, "Introducing STRATOS: A cloud broker service," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 891–898, doi: 10.1109/CLOUD.2012.24.

[39] M. N. Birje and C. Bulla, "Cloud monitoring system: Basics, phases and challenges," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 4732–4746, Sep. 2019, doi: 10.35940/ijrte.C6857.098319.

[40] J. S. Ward and A. Barker, "Observing the clouds: A survey and taxonomy of cloud monitoring," *J. Cloud Comput.*, vol. 3, no. 1, pp. 1–30, Dec. 2014, doi: 10.1186/s13677-014-0024-2.

[41] G. Aceto, A. Botta, W. D. Donato, and A. Pescapè, "Cloud monitoring: A survey," *Comput. Netw.*, vol. 57, no. 9, pp. 2093–2115, 2013, doi: 10.1016/j.comnet.2013.04.001.

[42] H. J. Syed, A. Gani, R. W. Ahmad, M. K. Khan, and A. I. A. Ahmed, "Cloud monitoring: A review, taxonomy, and open research issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 11–26, Nov. 2017, doi: 10.1016/j.jnca.2017.08.021.

[43] J. M. A. Calero and J. G. Aguado, "MonPaaS: An adaptive monitoring platformas a service for cloud computing infrastructures and services," *IEEE Trans. Services Comput.*, vol. 8, no. 1, pp. 65–78, Jan. 2015, doi: 10.1109/TSC.2014.2302810.

[44] W. Barth, *Nagios*, 2nd ed. San Francisco, CA, USA: No Starch Press, 2008. Accessed: Jun. 21, 2020. [Online]. Available: https://books.google.com.ng/books?hl=fr&lr=&id=QgYvDwAAQBAJ&oi=fnd&pg=PA5&dq=Barth,+W.,+2008.+Nagios:+System+and+network+monitoring,+No+Starch+Press.&ots=hTx_hizPp4&sig=RqZg0qxPGj4EyafYsGfN0PME8LM&redir_esc=y#v=onepage&q=Barth%2C W.%2C2008.Nagios%3ASys

[45] J. Gutierrez-Aguado, J. M. Alcaraz Calero, and W. Diaz Villanueva, "IaaSMon: Monitoring architecture for public cloud computing data centers," *J. Grid Comput.*, vol. 14, no. 2, pp. 283–297, Jun. 2016, doi: 10.1007/s10723-015-9357-4.

[46] S. Mongkolluksamee, P. Pongpaibool, and C. Issariyapat, "Strengths and limitations of Nagios as a network monitoring solution," *Proc. 7th Int. Jt. Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Feb. 2010, pp. 96–101, Accessed: Aug. 28, 2020. [Online]. Available: https://docplayer.net/1264602-Strengths-and-limitations-of-nagios-as-a-network-monitoring-solution.html

[47] P. Tader, "Server monitoring with Zabbix," *Linux J.*, vol. 2010, no. 195, p. 7, 2010, doi: 10.5555/1883478.1883485.

[48] A. Iqbal, C. Pattinson, and A.-L. Kor, "Performance monitoring of virtual machines (VMs) of type i and II hypervisors with SNMPv3," in *Proc. World Congr. Sustain. Technol. (WCST)*, Dec. 2015, pp. 98–99, doi: 10.1109/WCST.2015.7415127.

[49] D. Tovarnakk and T. Pitner, "Towards multi-tenant and interoperable monitoring of virtual machines in cloud," in *Proc. 14th Int. Symp. Symbolic Numeric Algorithms Sci. Comput.*, Sep. 2012, pp. 436–442, doi: 10.1109/SYNASC.2012.55.

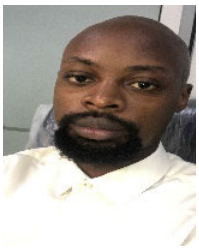
[50] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu, "GMonE: A complete approach to cloud monitoring," *Future Gener. Comput. Syst.*, vol. 29, no. 8, pp. 2026–2040, Oct. 2013, doi: 10.1016/j.future.2013.02.011.
[51] D. Milojičić, I. M. Llorente, and R. S. Montero, "OpenNebula: A cloud management tool," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 11–14, Mar. 2011, doi: 10.1109/MIC.2011.44.
[52] S. Hasan, S. O'Riain, and E. Curry, "Approximate semantic matching of heterogeneous events," in *Proc. 6th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS)*, 2012, pp. 252–263, doi: 10.1145/2335484.2335512.
[53] S. Al-Shammari and A. Al-Yasiri, "MonSLAR: A middleware for monitoring SLA for RESTFUL services in cloud computing," in *Proc. IEEE 9th Int. Symp. Maintenance Evol. Service-Oriented Cloud-Based Environ. (MESOCA)*, Oct. 2015, pp. 46–50, doi: 10.1109/MESOCA.2015.7328126.
[54] K. Vieira, A. Schulter, C. Westphall, and C. Westphall, "Intrusion detection for grid and cloud computing," *IT Prof.*, vol. 12, no. 4, pp. 38–43, 2010, doi: 10.1109/MITP.2009.89.
[55] S. Chatterjee and A. S. Hadi, *Regression Analysis by Example*. Hoboken, NJ, USA: Wiley, 2015.
[56] M. Eisler, *XDR: External Data Representation Standard*, document RFC 4506, 2006.
[57] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 494–503, Jan. 2021, doi: 10.1109/TII.2020.2975897.
[58] E. Adetiba, M. Akanle, V. Akande, J. Badejo, V.P. Nzanzu, M.J. Molo, V. Oguntosin, O. Oshin, and E. Adebiyi, "FEDGEN testbed: A federated genomics private cloud infrastructure for precision medicine and artificial intelligence research," in *Proc. Int. Conf. Inform. Intell. Appl.* Cham, Switzerland: Springer, 2021, pp. 78–91.
[59] S. A. de Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Commun. Mag.*, vol. 49, no. 2, pp. 130–137, Dec. 2011.

VINGI PATRICK NZANZU received the B.Sc. degree in electrical and computer engineering from the Université Libre des Pays des Grands Lacs (ULPGL), Goma, DR Congo, in 2017, and the M.Eng. degree in information and communication engineering from the Electrical and Information Engineering Department, Covenant University, Ota, Nigeria, in 2022.

He is currently working with ULPGL as an Assistant Lecturer. From 2020 to 2022, he was a Research Assistant with the Covenant Applied Informatics and Communication-Centre of Excellence (CApIC-ACE). His research interests include the development of an enhanced monitoring tool for federated cloud computing infrastructures, the IoT, software and system development, real-time systems, machine intelligence, data engineering, and analysis.

EMMANUEL ADETIBA (Member, IEEE) received the Ph.D. degree in information and communication engineering from Covenant University, Ota, Nigeria. He is currently a Full Professor and the Incumbent Head of the Department of Electrical and Information Engineering, Covenant University. He was the Director of the Center for Systems and Information Services (ICT Center), Covenant University, from 2017 to 2019. He is also the Incumbent Deputy Director of the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE) and the Co-PI for the FEDGEN Cloud Infrastructure Research Project at the Centre (World Bank and AFD funded). He is also the Founder and the Principal Investigator of the Advanced Signal Processing and Machine Intelligence Research (ASPMIR) Laboratory. He is also an Honorary Research Associate at the Institute for System Sciences, Durban University of Technology, Durban, South Africa. He has authored/coauthored more than 100 scholarly publications in journals and conference proceedings, some of which are indexed in Scopus/ISI/CPCI. His research interests and experiences include machine intelligence, software defined radio, cognitive radio, biomedical signal processing, and cloud federation. He is also a Registered Engineer (R.Engr.) with the Council for the Regulation of Engineering in Nigeria (COREN), and a member of the Institute of Information Technology Professional (IITP), South Africa. He was a recipient of several past and ongoing scholarly grants and funds from reputable bodies, such as the World Bank, France Development Agency (AFD), Google, U.S. National Science Foundation, the Durban University of Technology, Nigeria Communications Commission, Rockefeller Foundation, International Medical Informatics Association (IMIA), and hosts of others.

JOKE A. BADEJO (Member, IEEE) received the Ph.D. degree in computer engineering from Covenant University, Nigeria, in 2015. She is currently a Senior Lecturer and the Coordinator of the Computer Engineering Program with the Department of Electrical and Information Engineering, Covenant University, where she is also a Faculty Member of the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), a World Bank ACE-IMPACT Centre. Her research interests include biometrics and biomedical image analysis, machine (deep) learning, large-scale data analytics, software engineering, and cloud computing. She works as a Co-Investigator on a World Bank ACE-IMPACT Project targeted at developing a federated genomics cloud infrastructure for precision medicine in Africa. She has also been actively involved in the Covenant University Data Analytics Cluster (CUDAC), which supports data-driven decision-making at the university. With more than a decade of computing research, teaching, and leadership experiences, she has published over 40 papers in reputable journals and conference proceedings. She is a member of several professional bodies, including the Council for the Regulation of Engineering in Nigeria (COREN). She enjoys being an academic and loves contributing impactful and cost-effective solutions to current societal engineering problems in Africa.



MBASA JOAQUIM MOLO received the B.Sc. degree in electrical and computer engineering from the Université Libre des Pays des Grands Lacs, DR Congo, in 2019, and the master's degree in information and communication engineering from Covenant University, Nigeria.

He worked as a Research Assistant with the Covenant Applied Informatics and Communication-Centre of Excellence, from 2020 to 2022. He is currently a Lecturer at the Université Libre des Pays des Grands Lacs. His research interests include machine learning and cloud computing. Also, he is a member of Kwetu Best Sarl, a company focused on software development and robotics-related applications.

MATTHEW BOLADELE AKANLE is currently a Principal System Engineer at the Canter for System and Information Services (CSIS), Covenant University, Ogun, Nigeria. He has been a System Administrator and a Researcher with the Covenant University Bioinformatics Research (CUBRe) node of H3ABioNet, since 2015, where he developed a research interest. He also manages the high-performance computing platform of CUBRe. Currently, he is the Acting Director of CSIS Covenant University. His research interests include cloud computing, information security, bioinformatics, machine learning, and data communication. He got a scholarship to study computer engineering at a master's level from H3ABioNet through CUBRthe e node at the Department of Electrical and Information Engineering, College of Engineering, Covenant University, from 2018 to 2020. He is a member of the Nigeria Society of Engineers (NSE) and a Registered Engineer of the Council for Regulation of Engineers in Nigeria (COREN). He is one of the recipients of the fellowship to the IETF98 meeting in Chicago United State by Internet Society, in 2016. He is also one of the SCI-GAIA champions under the Sci-Gaia Project, an EU Horizon 2020-funded project, in 2016.

KALIMUMBALO DANIELLA MUGHOLE received the B.Sc. degree in electrical and computer engineering from the Université Libre des Pays des Grands Lacs, DR Congo, in 2018. She is currently pursuing the master's degree in information and communication engineering with Covenant University, Nigeria. She works as a Research Assistant with the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), a World Bank ACE-IMPACT Centre, Covenant University. Her research interests include cloud computing and machine (deep) learning.

VICTOR AKANDE is currently pursuing the B.Sc. degree in computer science with Elizade University. He is also a Software Engineer with over three years of experience building web and mobile applications. He is also the Vice President of the Nigeria Association of Computing Students (NACOS), Elizade University. His research interests include cloud computing, artificial intelligence, and software engineering. For more information visit the link (https://akande.com.ng).

OLUWADAMILOLA OSHIN (Member, IEEE) received the Ph.D. degree in information and communication engineering (with a focus on nano-electronic biosensing) from Covenant University, Nigeria, in 2020. She is currently a Lecturer in information and communication engineering with the Department of Electrical and Information Engineering, Covenant University, where she is also a Faculty Member of the Covenant Applied Informatics and Communication Africa Centre of Excellence (CApIC-ACE), a World Bank ACE-IMPACT Centre. Her research interests include mobile communications, data analytics, and MEMS-based biosensing. She has published about 30 papers in reputable journals and conference proceedings. She is a member of several professional bodies, including the Council for the Regulation of Engineering in Nigeria (COREN). She enjoys research and solving health-related issues using engineering and technology.

VICTORIA OGUNTOSIN received the B.Eng. degree in electrical engineering from the University of Ilorin, Nigeria, the M.Sc. degree in electrical and electronic engineering from the University of Greenwich, U.K., and the Ph.D. degree from the University of Reading, U.K. Her Ph.D. work at the University of Reading was on the development of a soft modular robotic arm. The research work involved building a soft robotic assistive arm that is actuated by pneumatics. She is currently a Lecturer at the Department of Electrical and Information Engineering, Covenant University, Ota, Ogun, Nigeria.

CLAUDE TAKENGA received the B.Sc. and M.Sc. degrees in radio engineering and telecommunication from Saint Petersburg State Technical University, in 2001 and 2003, respectively, and the Ph.D. degree in electrical engineering from the Institute for Communication Technology, Leibniz University of Hannover, Germany, in 2007. He is currently working in industry with the Research and Development Department, Infokom GmbH, Germany, where he collaborates in several international joint-projects focusing on integrating IT technologies in the health sectors. His research interests include eHealth systems, mobile applications, and communication technologies. He serves as a university professor in some countries. He has published numerous papers in conference proceedings and journals.

MAISSA MBAYE (Member, IEEE) received the master's degree in computer and ICT engineering (2ITIC) from Gaston Berger University, St-Louis, Senegal, in 2005, the master's degree in distributed systems, networks and parallelism (SDRP) from Bordeaux 1 University, France, in 2006, and the Ph.D. degree from the Bordeaux Computer Science Research Laboratory (LaBRI), Bordeaux 1 University, in 2009.

He has been an Associate Professor at Gaston Berger University (UGB), and a member of the Laboratory of Numerical Analysis and Computer Science (LANI), UGB, since 2011. He teaches computer science and his research interests include autonomous networking, knowledge plane, edge artificial intelligence (the IoT, fog-edge computing, and machine learning) applied to agriculture, health and environment, ICT4D, and cyber security. His work has been the subject of numerous scientific publications in international conferences and journals. He has been the Coordinator of the African Center of Excellence in Mathematics, Computer Science and ICT (ACE-MITIC), Senegal, since April 2020.

DAME DIONGUE (Member, IEEE) received the Bachelor of Science and Master of Science degrees in computer science from the Cheikh Anta Diop University of Dakar, Senegal, in 2007 and 2011, respectively, and the Ph.D. degree in computer science from Gaston Berger University, Saint-Louis, Senegal, in 2014. His research interests include networking, wireless mesh networking and wireless sensor networks, and particularly in scheduling algorithms and coverage heuristics design in wireless sensor networks. He is actually interested on AI application on agriculture and health. He was an Assistant Professor at the Assane Seck University of Ziguinchor, Senegal, from 2015 to 2017. He has been with the Institut Polytechnique de Saint Louis, an Engineering School hosted at Gaston Berger University, since January 2018. He is in charge of teaching network-based simulation techniques in wired and wireless networks at Gaston Berger University.

EZEKIEL F. ADEBIYI is currently a H3Africa, aka German Science Foundation (DFG) Projects Principal Investigator and the Head of the Covenant University Bioinformatics Research (CUBRe) Group. He is also the Centre Leader of the Covenant Applied Informatics and Communication African Centre of Excellence (CApIC-ACE), a new World Bank funded ACE Impact Project.