CADMM: Context-aware Adaptive Data Management Middleware for Federated Cloud Environments

Vikas K Kolekar
Research Scholar,
Department of Computer Engineering,
SMT. Kashibai Navale College of Engineering, Pune.
vikaskolekar2008@gmail.com

Sachin R Sakhare
Professor,
Department of Computer Engineering,
Vishwakarma Institute of Information Technology, Pune.
sachin.sakhare@viit.ac.in

***Abstract*— As cloud computing continues to evolve, the demand for efficient data management solutions in federated cloud environments becomes increasingly critical in terms of processing cost. In the proposed work, a novel middleware framework called Context-aware Adaptive Data Management Middleware (CADMM) is designed specifically for federated cloud environments. The proposed middleware framework integrates adaptive algorithms and intelligent data management techniques to optimize resource utilization, enhance data accessibility, and ensure robustness in highly dynamic and distributed cloud environment. Leveraging advanced algorithms and adaptive strategies the middleware framework dynamically adapts its data management policies and configurations based on changing workload patterns, resource availability, and network conditions. The comprehensive experimentation and performance evaluation is carried out to demonstrate the effectiveness and scalability in real-world federated cloud environments using CADMM. The obtained results shows a significant improvement in processing cost, data access latency, resource utilization efficiency, and overall system robustness as compared to existing approaches which were based on nonadaptive and non-context-aware techniques. The work represents a significant advancement in the field of federated cloud computing by offering an adaptive middleware framework to address the evolving challenges like Multidisciplinary Data, Vendor Lock-in, Interoperability Issue and Disparity of Services of data management in distributed cloud infrastructures.**

Keywords—Cloud Computing, Data Management, Context-awareness, Adaptive Middleware, Cloud Federation.

I. INTRODUCTION

Cloud computing [1] has revolutionized the world of modern computing by providing on-demand access to a shared pool of configurable computing resources over the Internet [2]. With the proliferation of cloud services and the emergence of diverse cloud providers, federated cloud environments have gained traction as a means to integrate and leverage resources across multiple clouds, thereby enhancing scalability, reliability, and flexibility. However, effective management of data in such federated cloud environments poses significant challenges due to the heterogeneity of cloud infrastructures, varying data access patterns, and dynamic resource availability [3]. Traditional approaches to data management in centralized cloud environments often fall short when applied to federated cloud environment, where data may be distributed across multiple clouds with diverse storage systems and access protocols [4]. Existing middleware solutions for federated cloud environments often lack adaptability and scalability, leading to suboptimal resource utilization, increased data access latency, and reduced overall system efficiency [5].

To address these challenges there is necessity of the system which will integrate adaptive algorithms and intelligent data management techniques to dynamically optimize resource allocation, data placement, and access strategies based on the evolving workload characteristics and system conditions. By continuously monitoring key performance metrics and adapting its policies in real-time, the system should enable efficient utilization of cloud resources, minimizes data access latency, and ensures robustness in the face of dynamic workload changes and resource fluctuations.

The following is the outline for the remainder of the paper: In section II discusses about the background and the related work of existing adaptive middlewares and federated cloud environments and identifies the scope for presented work. Section III describes the proposed methodology for context aware adaptive data

management middleware in federated cloud environment. Proposed methodology addresses the challenges of changing environment by providing a robust framework by constructing a context-aware middleware that adapts to changing environments and deliver highly customized user experience. Section IV presents the experimental setup carried throughout to implement the proposed methodology. In section V the proposed methodology is evaluated by the use case of smart home with respect to experimental setup as mentioned in earlier section. In section VI the discussion about results obtained using proposed system and its relation with the evaluated parameters is presented. Section VII provides the potential future scope directions for further investigations. Section VIII highlights the conclusion of proposed system. Conclusion shows significant achievements of context-aware adaptive data management middleware for federated cloud environment.

II. BACKGROUND AND RELATED WORK

Federated cloud environments aggregate resources from multiple geographically distributed cloud providers, offering users access to enormous computing power and storage capacity that surpasses what a single cloud service provider (CSP) could offer. These environments enable users to control the combined capabilities of various CSPs, potentially at lower costs and with improved scalability [6].

Traditional data management approaches are designed for centralized cloud environments and it struggles to adapt to the dynamic and heterogeneous nature of federated clouds. Here are few challenges:

Heterogeneity: Centralized clouds consist of resources of respective service providers with each having diverse configurations, capabilities, and pricing models. This heterogeneity necessitates a data management solution that can adapt to these varying resource characteristics.

Resources Non-availability: Resources in a centralized cloud environment can become unavailable due to maintenance, hardware failures, or cost fluctuations. An effective data management solution needs to be dynamic and adaptable to such changes in resources availability.

Network Latency: Data transfer across geographically distributed cloud providers can introduce significant network latencies. Efficient data management requires strategies to minimize the impact of these latencies on application performance.

Security Concerns: Data security is paramount, especially when various CSPs are subscribed for multiple services with potentially varying security policies. A robust data management solution needs to ensure data security and privacy across the cloud environment.

The National Institute of Standards and Technology (NIST) Cloud Federation Reference Architecture [7], outlines a model for building federated cloud systems. This actor-based model utilizes guiding principles from the NIST Cloud Computing Reference Architecture. It details eleven components and how they work together to enable secure and efficient resource sharing across federated cloud environments. The architecture allows for various deployment and governance options, catering to a range of federation complexities. Bishakh Ghosh et al. [8] explores the concept of building federated cloud environments without relying on a central authority. Traditional federated cloud models often require a trusted entity to manage resource allocation, security, and billing. Authors proposed a "trustless" approach, potentially leveraging blockchain technology or other decentralized mechanisms. Such a system could eliminate the need for a central authority, potentially improving security, transparency, and automation within federated cloud environments. Emphasis is also given on the challenges and potential benefits of trustless collaboration in cloud federations. Konstantinos Papadakis-Vlachopapadopoulos in [9] discussed and addressed challenges in managing services throughout the life-cycle in a federated edge-cloud environment, with a focus on establishing trust among the parties involved. Author proposed a blockchain-based Network Service Management (NSM) system for the leasing of services supplied by federated edge cloud providers and tenants. This system includes a new Cloud Service Chain (CSC) orchestrator. The proposed multi-domain architecture exhibits great scalability, allowing for the seamless addition of additional edge computing providers at any given moment, while incurring little administrative and resource overhead. Nour El Houda Bouzerzour et al. [10] examined the interoperability of services in cloud computing, considering the perspective of both clients and providers. The issue of vendor lock-in produced by proprietary services is emphasised, which hampers interoperability. Client-centric interoperability refers to the ability to seamlessly transfer data and applications across different cloud platforms, allowing customers to have more control and a larger range of service choices. On the other hand, provider-centric interoperability prioritizes cooperation among providers, enabling those who have extra resources to share them with others. The study closes by pinpointing dominant patterns and overlooked areas in research on cloud service interoperability. Norazian M Hamdan et al. [11] presents a reference design that aims to improve the semantic interoperability in multi-cloud systems. Proposed solution tackles the difficulties of incorporating diverse cloud services and guaranteeing smooth data interchange across various cloud platforms. The suggested framework utilizes semantic technologies to provide uniform interpretation and use of data across diverse cloud services. The research highlights the significance of standardized data representation and

communication protocols in order to enhance multi-cloud interoperability, with the goal of increasing efficiency and decreasing complexity in cloud computing operations. Gabriel Ayem et al. [12] reviews the limitations of cloud computing, specifically focus on the interoperability concerns that obstruct consumer adoption. The authors overview of the fundamental elements of cloud technology and introduces a three approach to problem-solving: preventative, detective, and remedial. The paper outlines five essential strategies for addressing difficulties in interoperability and using a content analysis technique to find six major solutions: standardisation, model-driven methods, open APIs, open protocols, abstraction layers, and service-oriented architecture-based solutions. The most often adopted option is standardisation, whereas model-driven techniques get the greatest level of support from multiple authors. Authors highlights the need of doing more research to improve customer confidence and acceptance of cloud services. Ansar Rafique et al. [13] presented and validated the SCOPE, a middleware that utilises policies and autonomously improves data management in federated cloud storage systems, which combine resources from various cloud service providers. Federated cloud systems encounter difficulties due to the dynamic nature of attributes such as variable Quality of Service (QoS) measurements and Service Level Agreement (SLA) guarantees, which makes human administration more complex. SCOPE tackled these challenges by facilitating a policy-based and autonomic middleware for data management to enhance decision-making and maintain SLA adherence. SCOPE has been validated using a realistic Software as a Service (SaaS) application and rigorous testing. The results has shown efficient management of runtime dynamicity and self-adaptive behaviour with minimum operator intermediation. Shuyou Wu et al. [14] introduced a resource sharing method called Cloud Light-Federation Sharing (CLFS), which allows each cloud to independently choose its own best strategies. This enables the formation of a federation without the need for complicated calculations for service request allocation and profit distribution. Furthermore, another a very efficient method for sharing resources called Cloud Cooperative-Federation Sharing (CCFS) was developed. This approach ensures that cloud providers collaborate completely and distribute profits fairly. The experimental findings demonstrate that the two federation approaches have a substantial impact on enhancing the overall utility and reducing the number of tasks that are not completed. R. Buyya et al. [15] designed and implemented a Fog computing framework called MicroFog. It is compatible with cloud-native technologies like Istio, Docker, and Kubernetes. Applications may be deployed and placement algorithms can be executed in federated Fog environments using MicroFog's flexible control engine. It simplifies the management of container orchestration and the creation of dynamic microservices by allowing for seamless integration and assessment of new placement rules. The scalability, flexibility, and efficacy of the framework in deploying microservices over dispersed fog-cloud environments have been confirmed via several use cases. Empirical studies demonstrate that the implementation of MicroFog may effectively diminish the response time of application services by as much as 54% by facilitation of horizontally scaled placement, service discovery, and load balancing. By facilitating the easy incorporation and evaluation of new placement rules, it streamlines the administration of container orchestration and the development of dynamic microservices. Multiple use cases have verified the framework's usefulness, scalability, and flexibility when it comes to delivering microservices across distributed fog-cloud environments. Research suggests that application service response times might be cut by up to 54% with the help of MicroFog. The three main mechanisms that do this are load balancing, service discovery, and horizontally scaled placement.

After the extensive literature survey, the research gaps are identified as below:

- Inefficacy to handle the diverse data produced in many areas such as healthcare, agriculture, hospitality, and education, by using cloud computing.
- In some situations, a single cloud environment may not be capable of managing a significant and diverse volume of data.
- When it comes to managing effective and precise data that is of high importance or requires high security in such cases traditional data management middleware is not suitable owing to the limits of the SLA.
- There are several situations in which criteria like as security, priority, load balancing, performance, cost availability, resource discovery, response time, resource optimization and most importantly context awareness may be taken into consideration.

Therefore, in order to address above mentioned gaps, the presented work suggests a solution that involves using an adaptive data management middleware and federation of cloud environment to effectively and precisely handle the data and meet the specific needs of the cloud users and cloud service providers by offering suitable services.

III. PROPOSED METHODOLOGY

The Adaptive Data Management Middleware (ADMM) acts as a mediator that is aware of the context and accuracy requirements, connecting cloud users with federated cloud service providers. The middleware, functioning as the central component, incorporates rules and algorithms that improve the adaptiveness of the entire system. The adaptive middleware acts as a flexible coordinator that enables smooth

communication and cooperation across various levels of the system.

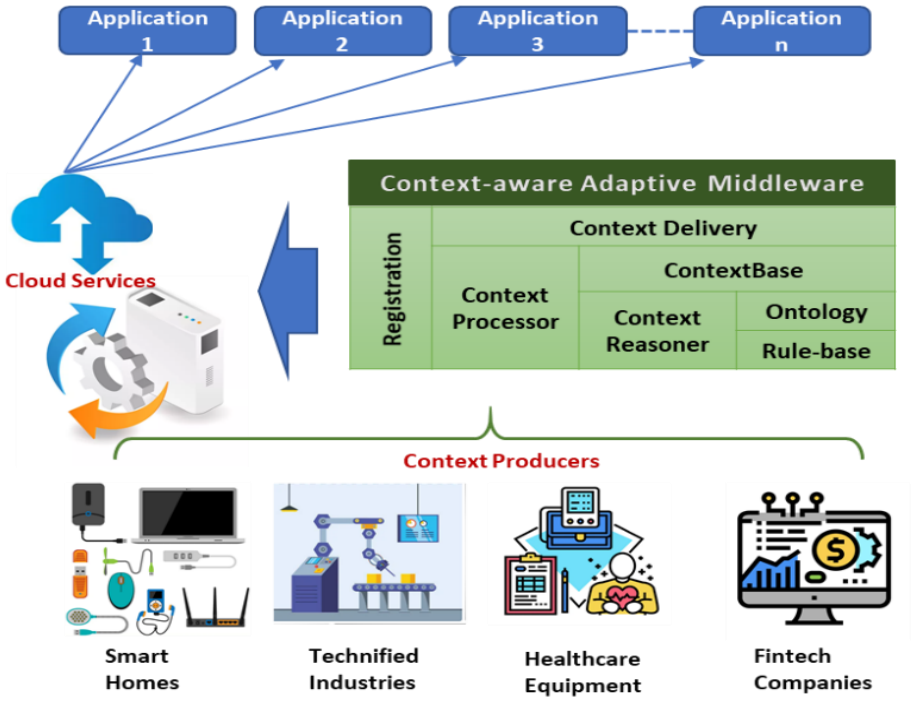


Figure 1: Proposed methodology for Context-aware adaptive data management middleware in the development of applications.

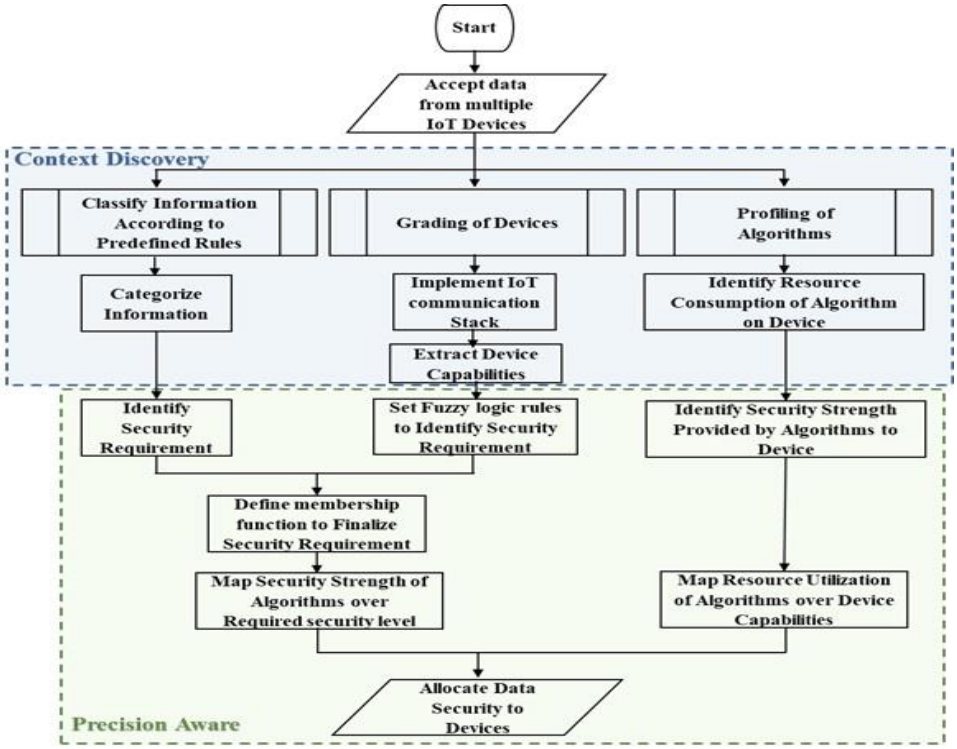
CADMM is specifically developed to facilitate applications in adjusting to changing environments, including variations in network circumstances, device capabilities, user preferences, and environmental considerations, among other things. Various methodologies are used to create CADMM, which can then be used to design a wide range of applications. Fig. 1. illustrates the comprehensive applications of CADMM. CADMM relies on several essential components for its operation. A significant volume of data is gathered from several sources, including smart homes, technologically advanced industries, healthcare equipment, and fintech organisations. The data has location, time, device type, network bandwidth, and other appropriate details. The context delivery mechanism is accountable for conveying the processed context data to the relevant components of the middleware, such as the context reasoner and rule-based system. The communication protocols and formats used by the context-aware application may vary based on its specific needs. Additionally, it has the capability to manage context changes and alerts. The context processor is tasked with gathering, interpreting, and converting context data into a format that can be used by other middleware components. The system collects data from several sources, including sensors, databases, and human interactions. The context reasoner is tasked with analysing the context data and determining how the middleware should respond according to the application's logic and rules. It uses several methodologies, such as machine learning, rule-based systems, and fuzzy logic, to make well-informed decisions by analysing contextual data.

The context ontology component establishes a collective vocabulary and a set of concepts that are used to define the context information. It offers a uniform method for various components to comprehend and analyse contextual information. The context rule-based system is responsible for implementing the application's rules and policies by using the context data and policies produced by the context reasoner. System administrators or end-users have the ability to set these rules, which may then be used to trigger certain behaviours or actions. The application may use many methodologies, such as rule engines, expert systems, and constraint solvers, to guarantee acceptable behaviour in various scenarios.

Collectively, these components provide a robust framework for constructing context-aware middleware that can adapt to changing environments and deliver highly customised environment for users. By integrating context-awareness into middleware, it is possible to enhance more intelligent, responsive, and efficient applications that effectively offer to users' requirements. The proposed methodologydesigned to manage data in a federated cloud environment, taking into account the heterogenous sources and multidisciplinary data created. Federated cloud environments include the collaboration of multiple cloud service providers and data centres, posing challenges in the management of data inside these environments. A precision-aware data management

algorithm prioritises optimising data placement and movement to enhance performance and save costs, while considering the precision needs of various applications.

Flowchart 1: Flowchart of the proposed CADMM system.



Flowchart 1 shows the data acceptance from multiple Internet of Things (IoT) devices. The accepted data is forwarded ahead for context discovery and lastly to find the precision awareness. In context discovery section, firstly the categorization of information will be done based on classification of information according to predefined rules. Secondly the device capabalities will be identified by implementing IoT communication stack with the help of device gradings. Thirdly the profiling of algorithms will be done to identify the resource comsumption of algorithm on device. In precision awareness section, mapping the security strength of algorithms over required security level will be determined by defining the membership function to finalize the security requirement. Membership function will be defined based on setting fuzzy logic rules to identify the security requirement. Also, the mapping of resource utilization of algorithms over device capabilities will be done by identification of security strength provided by algorithms to device. The combined collected information will suggest the allocation of data security to the devices to get final precision awarness.

Mathematical Model of the Proposed system:

1. **Device Capability Assessment:**

- Represent device resources as a vector:

$$D = (CPU\_Usage, Bandwidth, Battery\_Usage, Memory\_Usage, Context\_Awareness)$$

- Define fuzzy membership functions for each resource:

$$\mu\_LOW(x) => for\ Low\ levels$$
$$\mu\_MEDIUM(x) => for\ Medium\ levels$$
$$\mu\_HIGH(x) => for\ High\ levels$$

- Assess resource availability using fuzzy logic:

$$A(D) = ((\mu\_CPU(CPU\_Usage), \mu\_BW(Bandwidth), \mu\_BU(Battery\_Usage), \mu\_MU(Memory\_Usage), \mu\_CA(Context\_Awareness$$

2. **Data Collection and Classification:**

$$\mu\_General(x)$$
$$\mu\_Environment(x)$$
$$\mu\_Personal(x)$$

- Classify data using fuzzy rules:

$$C(S) = (\mu\_General(S), \mu\_Environment(S), \mu\_Personal(S))$$

3. **Fuzzy Ontology:**

- Classes:
  $Device, Sensor, Data, EncryptionAlgorithm, ConfidentialityClass$

- Represent sensor data as a vector:

$$S = (S\_, S\_2, \ldots, S\_n)$$

- Define fuzzy membership functions for confidentiality classes:

- Properties:

$hasCPUUsage, hasBandwidth, hasBatteryUsage, hasMemoryUsage, hasContextAwareness, generatedBy, suitableFor, requiresEncryption$

- Fuzzy membership functions:

$resource\ availability, confidentiality\ levels, and\ security\ strength$

4. **Semantic Web Rule Language(SWRL) Rules:**

- Represent fuzzy rules as SWRL axioms:

Device Capability Assessment:

$$Device(?d) \wedge hasCPUUsage(?d, High) \wedge hasBandwidth(?d, High) \wedge hasMemoryUsage(?d, High) \wedge hasBatteryUsage(?d, High) \wedge hasContextAwareness(?d, High) \rightarrow suitableEncryption(?d, Strong)$$

$$\ldots (rules\ for\ other\ resource\ combinations)$$

Data Classification:

$$Data(?d) \wedge belongsToCategory(?d, General) \rightarrow requiresEncryption(?d, Low)$$

$$Data(?d) \wedge belongsToCategory(?d, Environment) \rightarrow requiresEncryption(?d, Medium)$$

$$Data(?d) \wedge belongsToCategory(?d, Personal) \rightarrow requiresEncryption(?d, High)$$

5. **Encryption Algorithm Selection:**

- Use a fuzzy inference engine to evaluate SWRL rules and device capabilities.
- Recommend the algorithm with the highest membership value in the "suitableEncryption" class.

IV. Experimental Setup

Simulation approaches have been extensively used in several studies to examine the behaviour of large-scale distributed systems. Several simulators had been used by researchers in their work including GridSim, MicroGrid, GangSim, SimGrid, CloudSim, CloudAnalyst, iCanCloud, NetworkCloudSim, GreenCloud, MDCsim, EMUSIM, GroundSim, MRCloudSim, DCSim, and SimIC. While the first three concentrate specifically on Grid computing systems [16]. CloudSim is, to the best of our knowledge, the only simulation framework available for investigating cloud computing environments. The proposed work has been implemented using the CloudSim framework [17]. This is a simulation library specifically designed for certain cloud environments. This platform has offered the capability to specify setups, allowing for the evaluation of algorithms and the consideration of their results in this research. CloudSim has facilitated the evaluation and examination of algorithms, rules, and methods in a controlled and replicable environment. The framework enabled the simulation of several cloud-related components, including data centers, virtual machines, and cloudlets (which represents user activities or tasks). With this simulation tool created libraries that provide Java classes for creating users, cloudlets, data centers, virtual machines, and other cloud resources. Also, with this toolkit created the capability to serve both federated and single cloud environments.

The CloudSim toolkit comprises a collection of Java classes that implement different algorithms. Java classes majorly used were Cloudlet.java, DataCenterBroker.java and Datacenter.java. With the simulation, generated many cloud resources for the cloudlets to use. Table I displays the environments of the resources used for simulation.

TABLE I. THE QUANTITY OF CONFIGURATIONS USED FOR THE SIMULATION.

Sr. No.	***Parameter***	***Quantity***
1	Datacenter Broker	1
2	Datacenters	5
3	Virtual Machines	10
4	Cloudlets	100

TABLE II. THE CONFIGURATIONS VALUES USED FOR THE SIMULATION.

Sr. No.	***Parameter***	***Value***
1	Virtual Machine-Image Size	10000 MB
2	Virtual Machine-Memory	2048 MB
3	Virtual Machine-Bandwidth	1000 MIPS
4	Host-Storage	1000000 MB
5	Host-Memory	2048 MB
6	Host-Bandwidth	10000 MIPS
7	Cost per unit Bandwidth	0.03, 0.04 and 0.05

Table I and II summarizes the number of configurations and configuration values used for experimental setup.

V. Use Case Implementation

Adaptive data management and updating according to the context for households is crucial. An ordinary home is about to become smart home in the future[18]. The implementation of the smart home is done in this research with respect to the experimental setup as mentioned in above section. A variety of data is gathered via sensors in the context of smart home management, including information on the environment, energy, health, and personal details. Smart home systems are

designed to organize household tasks, facilitating a comfortable and personalized stay at home. However, a lot of gadgets often have inadequate memory and battery life, which must be controlled under smart home systems. Before cloud federation, all of the sensor's data had to be categorized in order to maintain selective privacy protection. A work strategy is developed to deal with these difficulties. The context inference is provided with significant information about the battery status, the availability of memory, and the privacy of data obtained from various sensors and devices.

VI. RESULTS

The simulation of the proposed methodology was conducted on cloudlets that are heterogeneous in nature. Cloud service providers are allocated based on the variable length of the cloudlets in the range from 0 to 7000 MB and step value of 1000 MB. The results obtained based on factors such as response time, processing cost, cloudlet length, start time, datacenter ID, virtual machine ID, virtual machine count, makespan. The experiment was repeated for 20 times. The result values are average of 20 experiments.

```
        hostList.add(
                new Host(
                    hostId+1,
                    new RamProvisionerSimple(ram),
                    new BwProvisionerSimple(bw),
                    storage,
                    peList,
                    new VmSchedulerTimeShared(peList)
                )
            ); // This is our machine

        // 5. Create a DatacenterCharacteristics object that stores the
        // properties of a data center: architecture, OS, list of
        // Machines, allocation policy: time- or space-shared, time zone
        // and its price (G$/Pe time unit).
        String arch = "x86"; // system architecture
        String os = "Linux"; // operating system
        String vmm = "Xen";
        double time_zone = 10.0; // time zone this resource located
        double cost=3; // the cost of using processing in this resource
        double costPerMem = 0.05; // the cost of using memory in this resource
        double costPerStorage = 0.001; // the cost of using storage in this
                                       // resource
        double costPerBw = 0.05; // the cost of using bw in this resource
        LinkedList<Storage> storageList = new LinkedList<Storage>(); // we are not adding SAN
                                             // devices by now

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                arch, os, vmm, hostList, time_zone, cost, costPerMem,
                costPerStorage, costPerBw);

        // 6. Finally, we need to create a PowerDatacenter object.
        Datacenter datacenter = null;
        try {
            datacenter = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), storageList, 0);
```

Figure 2: Implementation snapshot- Create object called DatacenterCharacteristics

```
private static List<Cloudlet> createCloudlet(int userId, int cloudlets){
    // Creates a container to store Cloudlets
    ArrayList<Integer> randomSeed=getSeedValue(cloudlets);
    LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

    int id = 0;
    long length = 0;
    long fileSize = 300;
    long outputSize = 300;
    int pesNumber=1;

    UtilizationModel utilizationModel = new UtilizationModelFull();

    int vmBindId=0;

    Cloudlet[] cloudlet = new Cloudlet[cloudlets];

    for(int i=0;i<cloudlets;i++){
        long finalLen=length+randomSeed.get(i);
        Log.printLine("The Final length for heterogeneous data "+finalLen);
        cloudlet[i] = new Cloudlet(id+i, finalLen, pesNumber, fileSize, outputSize,utilizationModel, utilizationModel,

        cloudlet[i].setUserId(userId);
        cloudlet[i].setVmId(vmBindId);
        if(i%2==1) {
            vmBindId++;
        }
        list.add(cloudlet[i]);
    }
```

Figure 3: Implementation snapshot- Create container to store Cloudlets

Sample implementation of proposed system on CloudSim simulation tool is as shown in fig. 2 and fig.3.

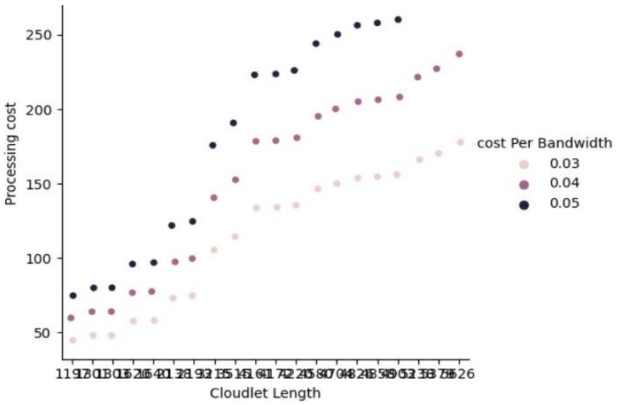


Figure 4: Comparison of processing cost on cost per bandwidth (0.03, 0.04, 0.05) for varying Cloudlet Length.

The relationship between the cloudlet length that the client provides and the processing cost is seen in Fig. 2. It has been noted that there is a direct relationship between the length of the cloudlet and the processing cost. The processing cost is determined using equation 1:

$$Processing\ Cost + = CostPerBw * getCloudletOutputSize() \quad (1)$$

CostPerBw is the cost value assigned to each individual unit of bandwidth. The function getCloudletOutputSize() is used to get the size of the output produced by a cloudlet. A cloudlet refers to a specific task or workload that is carried out inside a cloud computing environment.

The figure 4 depicts a scatter graph illustrating the correlation between the length of cloudlets and the cost of processing. The parameter on x-axis is labeled as "Cloudlet Length (MB)" while the y-axis is labeled as "Processing Cost". The y-axis is marked with tick marks ranging from 0 to 350, with intervals of 50. The x-axis is marked from 0 to 5000 with tick marks that increase by 1000. This graph shows processing cost is increasing alongwith change in cloudlegth length for all set of bandwidth. For 0.03 cost per bandwidth minimum processing cost value is 76 and maximum is 262, for 0.04 cost per bandwidth minimum processing cost value is 62 and maximum is 243, for 0.0.5 cost per bandwidth minimum processing cost value is 34 and maximum is 163. For example, the cloudlet length of 2138 mb has the processing cost of 73.14, 97.52 and 111.2 with respect to cost per bandwidth 0.03, 0.04, 0.05 respectively. Average processing cost for cost per bandwidth 0.03, 0.04 and 0.05 is 115.2, 153.6 and 212.5 respectively.

There is a clear correlation between the length of the cloudlet and the increase in processing cost. Consequently, the processing of longer cloudlets incurs higher costs. The implemented algorithm in this case does data aggregation depending on cloud length size. As the size rises, it imposes increased processing power, resulting in an increase in processing cost.

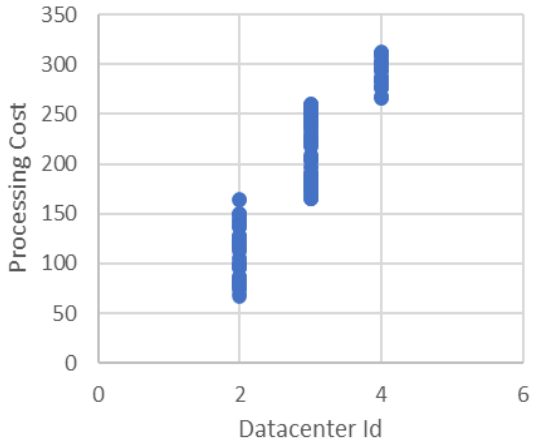


Figure 5: Graph for datacenter id vs processing cost

In fig. 5 the graph depicts the correlation between the cost of processing and data center id. The x-axis is labelled as "Datacenter Id" and ranges from 0 to 6. The y-axis is labelled as "Processing Cost" and ranges from 0 to 350. This graph illustrates a possible method of distributing costs for processing tasks across several data centers. Upon the arrival of a processing tasks, its estimated processing cost is computed. The graph determines the suitable data center for the tasks based on its cost. It is inferred here that if the processing cost falls within a certain range, it is assigned to the corresponding data center. By comprehending the cost allocation method, the results shows the valuable insights into the distribution of processing workloads among data centers, aiming for maximum efficiency and cost-effectiveness.

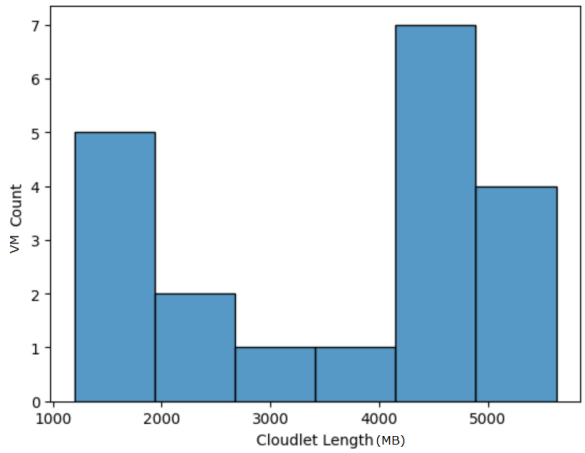


Figure 6: Graph for heterogeneous generated cloudlets against VM count.

Fig. 6 illustrates the information obtained from the CloudSim simulation, which generates random heterogeneous cloudlets to simulate data heterogeneity. Utilizing random heterogeneous cloudlets in CloudSim is an advantageous method for developing simulations that accurately represent the complexities and possibilities of handling varied data workloads in cloud computing systems.

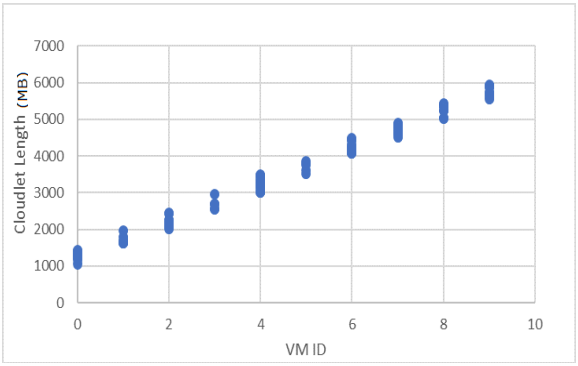


Figure 7: Graph for heterogeneous generated cloudlets getting executed on VM ids.

Fig. 7 illustrates the correlation between VM ID and cloudlet length. If the length of the cloudlet falls within a certain range, it is assigned to the corresponding virtual machine. Therefore, this demonstrates the presence of a straight-line pattern. The graph proposes an approach in which cloudlets are assigned to VMs according to their duration and the processing capability of the VM. The linear trend demonstrates a direct correlation between VM ID and the cloudlet lengths that they are capable of managing.

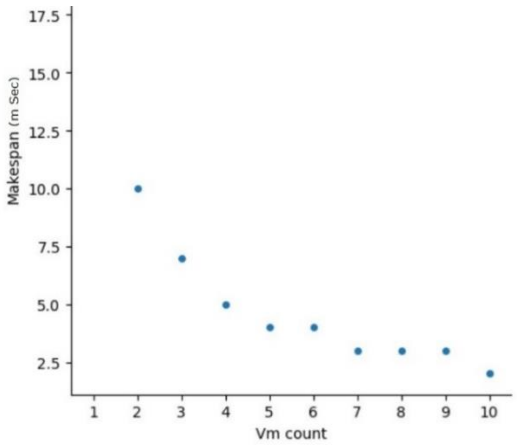


Figure 8: Graph for Makespan Time vs Virtual Machine Count

As shown in fig. 8 the makespan time refers to the total length of the time period during which all cloudlets have completed their processing. The pattern shown here exhibits a linear decrease as the number of virtual machines used for processing increases. The main observation from the graph is the consistent decrease in makespan time as the number of VMs rises in a linear fashion. This demonstrates that increasing the number of VMs may substantially decrease the total processing

time for the cloudlets. Makespan is a vital measure for assessing the effectiveness of the algorithms in cloud computing. The objective is to minimize the makespan of the tasks in order to optimize the use of resources and enhance production. Within the framework of this graph, using a greater number of VMs exhibits a distinct approach to get a reduced makespan, hence possibly expanding processing efficiency.

VII. Future Scope

The future scope for the middleware involves prioritizing scalability and performance by implementing a highly scalable architecture capable of managing increasing data quantities and complexity. Additionally, the aim will be to optimize performance by minimizing the overhead and maximizing resource utilization. Simplified interfaces and the ability to dynamically adapt are crucial for fulfilling individual user preferences and the needs of different applications. The performance monitoring and evaluation of specified parameters can be done on administrator dashboard. Based on dashboard visualization and parameter analysis the decisions will be taken to fine tune the overall performance, efficiency, and effectiveness of the system.

VIII. Conclusion

The proposed and implemented CADMM methodology offers a comprehensive strategy and method for managing cloud data using the recent trend of federated cloud environment, as opposed to the traditional and inefficient methodology. The implemented system's main objective was the development of an effective context-aware adaptive data management middleware that reduced the need for human intervention in the form of SLAs, terms and conditions, and other forms of manual intervention while simultaneously increasing dependability. Improvements to the state-of-the-art in cloud federation and data storage, such the capacity to do adaptive data placement and the consideration of variety of parameters characteristics have shown greater efficiency.

References

[1] Yahya Al-Dhuraibi et al., "Elasticity in Cloud Computing: State of the Art and Research Challenges", in: IEEE Transactions on Services Computing 11.2 (2018), pp. 430–447, ISSN: 19391374.

[2] Michael Armbrust et al., "A View of Cloud Computing", in: Commun. ACM 53.4 (Apr. 2010), pp. 50–58.

[3] M. R. M. Assis, & L. F. Bittencourt, "A survey on cloud federation architectures: Identifying functional and non-functional properties", Journal of Network and Computer Applications. Volume 72, September 2016. https://doi.org/10.1016/j.jnca.2016.06.014

[4] Y. Li, K. Hwang, K. Shuai, Z. Li and A. Zomaya, "Federated Clouds for Efficient Multitasking in Distributed Artificial Intelligence Applications", in IEEE Transactions on Cloud Computing, vol. 11, no. 2, pp. 2084-2095, 1 April-June 2023, doi: 10.1109/TCC.2022.3184157.

[5] Ramezani, Faeze & Abrishami, S. & Feizi, Mehdi. (2022), "A Market-based Framework for Resource Management in Cloud Federation", Journal of Grid Computing. 21. 10.1007/s10723-022-09635-w.

[6] Ray, Benay & Saha, Avirup & Khatua, Sunirmal & Roy, Sarbani. (2019), "Toward maximization of profit and quality of cloud federation: solution to cloud federation formation problem", The Journal of Supercomputing. 75. 10.1007/s11227-018-2620-2.

[7] [1]Craig A. Lee et al., "The NIST Cloud Federation Reference Architecture", NIST Special Publication 500-332, Feb 2020. https://doi.org/10.6028/NIST.SP.500-332.

[8] Bishakh Ghosh and Sandip Chakraborty,"Trustless Collaborative Cloud Federation", IEEE Transactions on Cloud Computing. PP. 1-15, January 2024, 10.1109/TCC.2024.3372370.

[9] Konstantinos Papadakis-Vlachopapadopoulos, "Trust-aware Life-cycle Management of Federated Edge Clouds", 2021DOI: 10.12681/eadd/50205.

[10] Bouzerzour NEH, Ghazouani S, Slimani Y "A survey on the service interoperability incloud computing: Client-centric and provider-centric perspectives", Softw Pract Exper. 2020;50:1025–1060. https://doi.org/10.1002/spe.2794

[11] Norazian M. Hamdan and Novia Admodisastro, "Towards a Reference Architecture for Semantic Interoperability in Multi-Cloud Platforms", International Journal of Advanced Computer Science and Applications, Vol. 14, No. 12, December 2023.DOI: 10.14569/IJACSA.2023.0141254

[12] Gabriel Ayem, Salu George Thandekkattu and Narasimha Rao Vajjhala, "Review of Interoperability Issues Influencing Acceptance and Adoption of Cloud Computing Technology by Consumers", In: Reddy, V.S., Prasad, V.K., Mallikarjuna Rao, D.N., Satapathy, S.C. (eds) Intelligent Systems and Sustainable Computing. Smart Innovation, Systems and Technologies, vol 289, May 2022. Springer, Singapore. https://doi.org/10.1007/978-981-19-0011-2_5

[13] Rafique, A., Van Landuyt, D., Truyen, E. et al. "SCOPE: self-adaptive and policy-based data management middleware for federated clouds", J Internet Serv Appl 10, 2 (2019). https://doi.org/10.1186/s13174-018-0101-8

[14] Shuyou Wu, Zhengxiao Wu, Xiaohong Wu, Jie Tao, Yonggen Gu, "Queuing-Based Federation and Optimization for Cloud Resource Sharing",Information.2022,13,361.https://doi.org/10.3390/info13080361

[15] Buyya, R., Ranjan, R., Calheiros, R.N. (2010), "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services", In: Hsu, CH., Yang, L.T., Park, J.H., Yeo, SS. (eds) "Algorithms and Architectures for Parallel Processing", ICA3PP 2010. Lecture Notes in Computer Science, vol 6081. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13119-6_2

[16] Wickremasinghe, Bhathiya et al. "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications." 2010 24th IEEE International Conference on Advanced Information Networking and Applications (2010): 446-452.

[17] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Softw. Pract. Exper. 41, 1 (January 2011), 23–50. https://doi.org/10.1002/spe.995

[18] Luntovskyy, Andriy & Beshley, Mykola & Guetter, Dietbert & Beshley, Halyna. (2023) "Technologies and Solutions for Smart Home and Smart Office", 10.1007/978-3-031-40997-4_13.

Vikas Kolekar is a Research Scholar in Department of Computer Engineering, SMT. Kashibai Navale College of Engineering, Pune, INDIA and working as an Assistant Professor, Department of Computer Engineering, Vishwakarma Institute of Information Technology, Pune, INDIA. He has obtained Master of Computer Engineering degree from SPPU, Pune university. His area of interest and work includes in domain of Cloud Computing, Networking and Security. He has academic experience of more than 8 years. He published/presented research 10+ papers in reputed Journals and Conferences. He has registered 2 copyrights on his name. He is on editorial board and reviewer of reputed research journals. Mr. Vikas K Kolekar is a Cisco Certied trainer and giving certication training to students in Cisco Networking Academy, Vishwakarma Institute of Information Technology,Pune. He has the membership of Association of Computing and Machinery (ACM) and International Association of Engineers (IAENG).

Email:vikaskolekar2008@gmail.com

Dr Sachin Sakhare is working as a Professor in the Department of Computer Engineering of Vishwakarma Institute of Information Technology, Pune, India. He has obtained his PhD degree in Computer Science and Engineering from Nagpur University. He has 24 Years of experience in the field of Engineering education and Research. He is recognised as PhD guide by Savitribai Phule Pune University and currently guiding 4 PhD research scholars. He worked as a reviewer of Applied Soft Computing journal from Elsevier, The journal of Soft Computing, IEEE-ICCUBEA-2017, IEEE-ICCUBEA- 2018 and 2019. IEEE-INDICON, Springer and ACM conferences. He has also Worked as a member of Technical and Advisory Committee for various international conferences. Dr. Sachin has Delivered invited talks at various Conferences, FDP's and STTP's. Dr. Sachin is associated with international universities also. He was invited as Guest Faculty at Amsterdam University of Applied Sciences (AUAS), Netherlands. He is a member of CSI and ISTE. He has Presented more than 30 research communications in national and international conferences and journals, with 168 citations and H-index 9. He has authored more than 5 books. He has filed and published 5 patents.

Email:
sakharesachin7@gmail.com