JOINED HETEROGENEOUS CLOUDS RESOURCES MANAGEMENT: AN ALGORITHM DESIGN

Zayyanu Umar¹, Agozie Eneh² Okereke George E.³

1Department of Computer Science College of Science and Technology Federal Polytechnic, Birnin Kebbi, Nigeria.

2³Department of Computer Science Faculty of Physical Science University of Nigeria, Nsukka, Enugu State.

¹zayyanuumar1@yahoo.com

Abstract

Nowadays, cloud computing services have been an embracing computing technology by some organisations, academia and entrepreneurs. Cloud Service Providers (CSP) are constrained to specific resources, missing some of the resources their clients need; this triggers the need for many and specific interconnections of homogeneous or heterogeneous computing clouds by their protocols and architectures to interoperate and share available resources among them. Clouds interconnection can be with various functions and schemes. In this study, we deployed exploratory and Design Science research approaches and Cloud-Analyst to simulate interconnections and interoperability within heterogeneous cloud service providers. The study cannot be conducted with real cloud computing environments due to the high cost that may incur and authorizations from CSPs that may not be secured. In this paper, we built a system and algorithm that can handle the variability and complexities of the different clouds during the management of inter-cloud resources. The experiment result shows that the USER-BASE (UB1) can subscribe to Data Center 1 (DC1) through Data Center 3 (DC3) that it initially subscribed with average time 301.05 with insignificant differences when utilizing resources from Data Center 3 (DC3).

Keywords: Cloud Resources, Clouds Heterogeneity, Algorithm, Cloud Service Providers.

I. Introduction

Cloud computing is a new technology embraced by many organisations, academia and business organisations. The essence is replacing traditional information technology (IT) decentralised, labour-intensive services with centralised resource-intensive cloud services by optimising the throughput, cutting the cost of expenditure and other derivable benefits by deploying the technology.

Copyright reserved © J. Mech. Cont. & Math. Sci. 
Zayyanu Umar et al
By the advent of cloud computing services, the data is spread from one or different data centers to different regions and varied file systems and in varying formats, flowing from one site to another site. A cloud computing service consumer can be from any perspective of the world and yet another big challenge is the volatility aspect of the data in use.

Interoperability among the Cloud Service Providers (CSPs) is the solution to the subscriber being locked-in in one particular cloud. Each service renders by Cloud Service Providers (CSP) can be categorized into threefold, which are called “cloud service models,” such as (a) Software-as-a-service (SaaS), (b) Infrastructure-as-a-service (IaaS) and finally (c) Platform-as-a-service (PaaS). Some authors specify a wide need in developing a cloud where various CSPs can seamlessly get access to each other's services, which we and others consider as multi-cloud [XIII]. Cloud interoperability issues have not been taken into account by cloud service providers and each CSP comes with its service and service settings [XV].

The problems of linking several and specific CSPs are that most of the configured cloud platforms are not compatible with each other and cannot exchange resources with each other, as everyone communicates with different settings [III].

This heterogeneity is a severe problem of shared cloud computing environments as it generates obstacles in the way of the omnipresent cloud realization. Another obstacle is the lock-in of vendors, customers submitting a request for cloud service have to tailor their requests to suit the cloud provider's pattern and interfaces, resulting in costly and difficult future relocations [XV].

**Clouds Heterogeneity**

Heterogeneity refers to different protocols; standards cloud service providers used in rendering services to clients. The absence of standards for control and settings of heterogeneous infrastructures denies companies, who need to utilize heterogeneous infrastructures in the transaction model, of advantage from infrastructure heterogeneity such as resource management, geographical advantages, pricing model, hypervisor type benefits, recovery, and security [VI].

Kanungo [VIII] states the design issues of joining more than one cloud to interoperate among them. He stated the following as main heterogeneity when federating clouds.

**Interface:** Different cloud service providers are with distinct. Application Programming Interfaces (API). Some researchers seek for the solution that can provides a standard software component (Interface) that is compatible with all other interfaces of the joint clouds based on the Service Agreement Licence (SAL) [X].

**Resource Heterogeneity:** Each service model, such as SaaS, PaaS and IaaS, differs from one cloud service provider to another. The resources are of varying capacities and abilities. Resource Availability, processing and storage capacity, resource volume, resource varieties are all different features of distinct clouds.
The Billing/Pricing System: This is another difficulty when joining different clouds to interoperates. The Service Level Agreement will solve the obstacle.

Networking: Interconnecting Virtual machines with different networking architectures and topologies is another bottleneck to the needs of joint interoperable cloud service providers (CSPs). The distinct CSP may be on one network addressing scheme. The solution to that harmonises the conflicts and incompatibilities among the CSPs with different Networking Architectures, Topologies, addressing schemes and other network issues.

Clouds Interoperability

The ability to transfer service from one Cloud Service Provider to another is referred to as Clouds interoperability. Once the Cloud provider stands to be alone with all its peculiar settings, then the issue of Interoperability continues to stands amongst the intended or joint clouds.

The user of the cloud needs to migrate their service from one cloud to another in case of any service failure on one cloud, but both of them need not to be using the same type of technology to support hypervisor. It may happen the second cloud will not support the hypervisor tecnology supported by the first cloud so to make every cloud on the same platform is a big issue in itself [XIV].

The cloud computing features such as elasticity, security, authentication, service invocation, and scalability, accessibility and flexibility should not be restricted to only one particular cloud service provider; it should be commonly extended to all clouds agreed to join one umbrella of multiple clouds by reaching consensus with top key players which is not an easy task. Various standards, policies and protocols have to be agreed upon by individual cloud that intended to interoperate, which are part of what is call Service Level Agreement (SLA).

Intercloud Interoperability permits sharing/demanding resources from some cloud service providers that are not available with the currently subscribed cloud services.

It is similar to the nowadays banking system in which banks interoperate among themselves. It is also analogous to the way mobile telephone operators execute inter-carrier interoperability. Such kinds of exchanges, peerings, or roaming may introduce to both cloud service consumers and cloud service provides business opportunities.

II. Related Works

In a cloud infrastructure, the mechanism in which the workload is divided amongst several nodes of a distributed network is called load balancing. A lot of researches have been done to manage the workload to improve performance and prevent resource overuse. The algorithm for load balancing is categorized into two broad categories, that is dynamic and static. Figure 1 displays the algorithms for the load balancing category. In this segment, we give a full overview of the cloud's current load balancing algorithms.
Fig. 1: Classification of Load Balancing Algorithms

Honey Bee Foraging Algorithm: This imitates the actions of honey bees seeking provisions. The algorithm is generated from an extensive study of how honey bees adjust for searching and obtaining provisions. Within beehives, there is a group of bees termed scout bees that forage for supplies sources, when they discover any, they return to the colony to tell them about this using a dance called vibration dance, giving the idea of the extent and amount of food and also the distance from the beehive. Forager bees immediately follow the Scout Bees to the area of supply and then continue to collect it, then back to the beehive and perform a vibrational dance with other bees in the colony giving an idea about the quantity of food left and thus resulting in either further exploitation or abandoning food [XII]. On the basis of this phenomenon of bees looking for food, the algorithm works in the same way, the removed tasks from overloaded Virtual Machines (VMs) are considered as honey bees when submitted to the underloaded VM; the task will change the number of different major tasks and load that specific VM into all other waiting tasks, which will be helpful in selecting their virtual machine-dependent. When a high-priority task is to be forwarded to other VMs, the VM should be considered as having a minimum amount of high-priority activities so that the specific task is completed as soon as possible. As all VMs are ordered on a load-based basis in ascending order, the removed task will be sent to the underloaded VM. Mostly, honey bees does activities such as assigning tasks, relieving task from VMs while Virtual Machines provide all services needed.

Ant Colony Algorithm: The algorithm is being developed to regulate the load that applies ant activity to look for food. More significant weight resource has high computing power. Load balancing ant colony optimization not only regulates the workload but also reduces makespan. We assume that all functions are mutually autonomous and computationally complex [I].

Round Robin Algorithm: The algorithm disseminates the workload to all nodes equally. The scheduler for the algorithm assigns one virtual machine to a node in a cyclic fashion. The round-robin system for virtual machine scheduling is close to the algorithm schedule for the scheduling of processes. The scheduler of the algorithm

Copyright reserved © J. Mech. Cont.& Math. Sci.
Zayyanu Umar et al
begins allocating virtual machines to each node and goes on to position the next VM into the next node. For all nodes, the algorithm is repeated until each node is at least on VM. As it finishes putting VMs on all nodes, it will then go to the first node and re-cycle the process for requests from the next virtual machines. Before moving on to the next node, the algorithm does not wait for the exhaustion of all available node resources. For instance, if there are five nodes and five virtual machines are to be assigned, each node is assigned to one virtual machine, that is, if all nodes have adequate resources available to run in the VMs.

This algorithm's key focus is that it uses all of the assets in an orderly way. All nodes which preserve fairness are scheduled to have an equal number of virtual machines. The main drawback of using such an algorithm, however, is that the energy intake will be high, as several nodes will be kept on for a prolonged period of time. If five services run on a single node, all five nodes are turned on when using the Algorithm: Round Robin, then it leads to consuming a tremendous amount of power [XI].

**Min-Min Scheduling Algorithm**: This algorithm first measures the estimated execution time of the tasks and then assigns the task to the corresponding virtual machine with the minimum execution time. The Min-Min scheduling algorithm, in other words, works to determine the required completion time of each activity. The Min-Min Scheduling Algorithm has the advantage of being valid in small scale distributed systems. And the Min-Min Scheduling Algorithm's drawback is that the number of small tasks exceeds the number of extensive activities to the extent the algorithm that can not schedule new task [VII].

**Max-Min Scheduling Algorithm**: This algorithm is essentially the same as the Min-Min algorithm, but, first of all, it plans larger tasks, i.e., after determining the minimum execution times, the maximum value is chosen which is the maximum time on any resource among all tasks. The same process is repeated until all activities are scheduled [V].

**Throttled Algorithm**: This algorithm is based entirely on the virtual machine. In this algorithm, the client asks the load balancer to search the correct virtual machine, which immediately accesses the load and performs the operations the client needs. In this algorithm, the load balancer maintains an index table of virtual machines and their (Busy or Available) statuses. The client, therefore, first requests the load balancer to find a suitable Virtual Machine to perform the necessary operations [IV].

**Priority-Based Job Scheduling Algorithm**: The developer suggests using the Analytic Hierarchy Process model, a new propriety-based job scheduling algorithm in a cloud computing environment based on a multi-criteria decision-making approach. They also addressed some algorithm-related issues, including complexity, accuracy and time to finish. This algorithm’s main drawback is that it is relatively complex and does not determine the finishing time of the tasks, while the response time is low. This algorithm does not work well for a large number of tasks since the dynamic calculation of task priorities is complex [X].
Demchenko, Turkmen, Laat and Slawik [II] analyse the general multi-cloud use case that assists in extracting the requirements to InterCloud Security Framework and identifying the security infrastructure functional components that would allow using distributed cloud-based resources and data sets. The paper describes the major services and functional components of the InterCloud Security Framework (ICSF). It explains the importance of consistent implementation of Security Services Lifecycle Management in cloud-based applications. The paper covers all aspects of the principles of cloud enforcement and their role in cloud protection. The article relates to the development of security infrastructure in the CYCLONE project, which implements federated identity management, secure logging service, and multi-domain attribute-based access control, lifecycle management of security services.

The above existing algorithms were developed to serve in one particular cloud computing environment or network not in joined varied cloud computing environment, this disables interoperation between cloud environments for the purpose of sharing cloud-environment individual lacking resources.

III. Methods And Materials

As suggested by many researchers the need for interconnections of homogeneous or heterogeneous cloud computing environments by their protocols and architectures to interoperate and share available resources among them. Given the uniqueness and scope of the problem to be solved, an exploratory research methodology will be adopted as an integral part of the research. On the other hand, since this study involves the creation of a prototype, which can be seen as a proposal for a novel product, this work would pursue a prototype implementation of design science research in an information system for proof of concepts such as new techniques for data extraction [IX].

Cloud Simulator (CLOUD-ANALYST) is used in simulation and modelling of entities in cloud computing platforms, which consist, users, data centers, tasks and virtual machines.

IV. Developed Framework And Algorithm

The framework is developed in such a way that each cloud service provider is distinct from other cloud providers. Each provider has its own different cloud management software and configuration.

**Local CSP-Manager** is responsible for ensuring the compliance of centralised standards and protocols before presenting or demanding service from the central manager. The topology adopted is star topology to enable interoperability amongst the clouds that register for cooperations.

Each cloud in figure 2 provides service to its client and it stands to rely on the central node when the need arises.

**Central CSP-Manager:** This is responsible for complete intercloud controls and management functions: such as inter clouds signalings, services model
synchronization, security monitoring, service lifecycle management, standards compliance monitoring, configuration and protocols management, topology management, metadata management, cloud admission, cloud re-admission and cloud de-commissioning.

The Central Manager is also responsible for fulfilling Intercloud operation requirements such as Service broking, service registering, CSP and Clients registrations, Service Level Agreement (SLA) and negotiations.

Algorithm for Service Provisioning In Joined Clouds

It is pertinent to state the relevant algorithms were the developed algorithm was formed from. The throttled algorithm, Priority-based job scheduling algorithm and Honey Bee Foraging Algorithm forms the author designed algorithm:

**Throttled Scheduling Algorithm:** The algorithm looks for the best-fit virtual machine to allocate the task

\[
\text{Let } q \text{ be the task to assign to VM for operation} \\
\text{For } i = 1 \text{ to } z \quad \left(\text{z number of VM in one particular of Cloud}\right) \\
\quad \text{If } q = z \left(i\right) \text{ then} \\
\quad \quad \text{Allocate task to VM} \\
\quad \text{Else} \\
\quad \text{Loop } i \\
\quad \text{Until found} \\
\text{End}\]

**Priority-Based Job Scheduling Algorithm:** In contrast to throttled algorithm, the Priority algorithm considers the priority of tasks the need VMs attention, then assign task that most needed attention to the VM.

Copyright reserved © J. Mech. Cont.& Math. Sci.
Zayyanu Umar et al
Let $q (1….n)$ be the sequences of tasks that need VM attention

For $q = 1$ to $n$  // $n$ number of tasks that need VM attention
  For $i = 1$ to $z$  // $z$ number of VM in a Cloud
    Let $n(q)$ be the task that needs VM attention most.
    If $n(q) = z(i)$ then
      Allocate the task to VM
    Else
      Loop $q, i$
    Until Priority found
End

**Honey Bee Foraging Algorithm:** The bees are divided into two; those survey for food and those to harvest the food. Those to go for survey will come back showing sign that the food found and its amount or what remains. This can be represented in Do… while or For……Next loop:

Let $x$ be VM that provides operations to assigned tasks
Let $q$ be set of tasks a waiting to be assigned
Do
  Find $(x)$
  If $x$ found
    For $q = i$ to $n$  // $n$ number of set of tasks
      Assign $q(i)$ to $x$ found
      Next $i$
  End if
While $x < > 0$

The following algorithm translates what the service provisioning diagram (Figure 2) offers. It implements the behavior of the three stated algorithms above to form designed joined service provisioning algorithm of interconnected varied clouds. The algorithm is of two modules; one is Local-Cloud Service Provider (LCSP) and second is for Central-Manager Module (CMM).

```c
/*Local-Cloud Service Provider (LCSP)*/
Loop
  Query for desired_Service
    Query for Service_Type
    Query for Service_Capacity
  IF Service_desired FOUND on local CSP
    THEN  Provide Service to a User Request
  ELSE
    Present_to_a User; Available Sources
    Invoke Middleware forSettings_Reconfigurations
    Present query to Central CSP
  ENDIF
-----------------------------------------------
```

Copyright reserved © J. Mech. Cont.& Math. Sci.
Zayyanu Umar et al
The Central Cloud Service Provider Manager: Algorithm

Loop
  Query for desired_Service
  Query for Service_Type
  Query for Service_Capacity
IF
  SLA_Compliance: YES;
THEN
  IF Registered_Service<>Service in need
  THEN
    Present message to LCSP: "Service in Request NOT in Register."
  ELSE
    Loop toK/K- number of interconnected Cloud Service Providers*/
    IF Service in need NOT available
    THEN
      Present Message toLCSP: “NOT Found Service”
    ELSEIF Service Available on N/N - number of interconnected Clouds*/
    THEN
      Invoke Middleware
      Central CSP Present Service to Nearest Local CSP
      Local CSP Present Service to Cloud in need
      CSP in need present the Service to User in need
  ENDIF
ENDIF
LOOP UNTIL Service_Query<> 0

V. Result and Discussion

Experimental Setting

The division is all concerning the computational experiments which are applied to evaluate the performance of the intended algorithm. The designed algorithm is simulated by using the Cloud-Analyst tool, a module of Cloud-Analyst. The base model for this toolkit relies on JAVA. The designed algorithm is the composition of three existing Scheduling algorithms: Throttled algorithm, Priority-based job scheduling algorithm and Honey Bee Foraging Algorithm and it is named as Joined Scheduling Policy:

Heterogeneous Datacenters

The data centre VMs within a cloud are considered uniform while they are heterogeneous across clouds and in terms of computation capability and network delay. Then we model VMs on the cloud as follows:

\[ v_i = (UI, RI, \theta_i) \]  

Where UI is the CPU speed, RI is the network delay from the client to VM, and \( \theta_i \) is the average CPU usage. Table 1 and figure 3 shows heterogeneous configurations of data centres:
Table 1: Heterogeneity in CSPs/Data Centers

| S/N | Region/Data Center | CPU Type | OS     | Scheduling Policy | Hypervisor | No of VMs |
|-----|--------------------|----------|--------|-------------------|------------|-----------|
| 1.  | DC1 (North America)| AMD 6300 | LINUX  | Throttled algorithm, XEN | 10         |
| 2.  | DC2 (South America)| Intel Corei7 | Windows | Priority-based job scheduling, KVM | 15         |
| 3.  | DC3 (Europe)       | Intel E5-2600 | XENIX  | Honey Bee Foraging, Hyper-V | 5          |

Fig. 3: Heterogeneous Configurations of Data Centres (Cloud Environments)

Results Of The Simulation

Figure 4 depicts the scenario of simulation before deployment the proposed algorithm:

Fig. 4: Cloud-Analyst Simulation before Deployment the Proposed Algorithm
Figure 5 depicts the scenario of simulation after deployment of the proposed algorithm:

![Cloud- Analyst Simulation after Deployment of the Proposed Algorithm](image)

**Fig. 5:** Cloud- Analyst Simulation after Deployment of the Proposed Algorithm

After simulating with the proposed algorithm, the performance analysis result displays, as shown below in table 2, 3 and 4:

**Table 2: Overall Response Time Summary**

|                      | Avg (ms) | Min (ms) | Max (ms) |
|----------------------|----------|----------|----------|
| Overall response time: | 254.31   | 39.15    | 56443.76 |
| Data Center processing time: | 10.26    | 0.01     | 56390.25 |

**Table 3: Response Time by Region**

| Userbase | Avg (ms) | Min (ms) | Max (ms) |
|----------|----------|----------|----------|
| UB1      | 301.05   | 235.59   | 369.11   |
| UB2      | 50.47    | 39.15    | 60.89    |
| UB3      | 299.64   | 238.61   | 363.11   |
| UB4      | 140.14   | 39.86    | 56443.76 |
| UB5      | 301.88   | 229.53   | 385.62   |
| UB6      | 299.93   | 240.12   | 381.11   |
| UB7      | 299.87   | 234.12   | 384.12   |
| UB8      | 299.42   | 235.62   | 364.62   |
| UB9      | 297.86   | 241.61   | 442.53   |

*Copyright reserved © J. Mech. Cont. & Math. Sci.*

Zayyanu Umar et al
Table 4: Data Centre Request Servicing Times

| Data Center | Avg (ms) | Min (ms) | Max (ms) |
|-------------|----------|----------|----------|
| DC1         | 90.15    | 0.01     | 56390.25 |
| DC2         | 0.74     | 0.05     | 1.14     |
| DC3         | 0.39     | 0.02     | 19.51    |

VI. Discussion

The algorithm allows the USER-BASE (UB1) to subscribe to DC1 through DC3 that it initially subscribed to without having over cost and with the same average time, as shown in table 3:

Table 5: User-Base 1 Service Request Time

| Userbase | Avg (ms) | Min (ms) | Max (ms) |
|----------|----------|----------|----------|
| UB1      | 301.05   | 235.59   | 369.11   |

In table 4, the Data Center (DC1) has the highest servicing request time; it can accommodate as many users as many with the deployment of the designed algorithm.

Table 6: Data-Center 1 Request Service Time

| Data Center | Avg (ms) | Min (ms) | Max (ms) |
|-------------|----------|----------|----------|
| DC1         | 90.15    | 0.01     | 56390.25 |

VII. Conclusion and Future Work

Variability in designed joint cloud service providers gives rise to the inability to interact with cloud service providers and restrict cloud service active users to be trapped in their resource demands. Harmonizing the heterogeneity among interconnected clouds by developing algorithm and system that can accommodate the complexities and differences, then, there will be the satisfaction to subscribers and smooth interoperability service providers. The challenge is solved by creating a solid structure and algorithm for both variability problems in joint cloud service providers and by using cloud-Analyst to experiment with. Because of its robustness, massive complexities and heterogeneity, forthcoming studies are needed to create a virtual forensic framework for the Internet of Things (IoT).
References

I. Aslam, S., & Shah, M. A. (2016). Load balancing algorithms in cloud computing: ASLMAA survey of modern techniques. In 2015 National Software Engineering Conference, NSEC 2015 (pp. 30–35). https://doi.org/10.1109/NSEC.2015.7396341

II. Demchenko, Y., Turkmen, F., Laat, C. De, & Slawik, M. (2017). Defining Intercloud Security Framework and Architecture Components for Multi-Cloud Data Intensive Applications, 945–952. https://doi.org/10.1109/CCGRID.2017.144

III. Garrison, C. p. (2010). Digital forensics for network, internet and cloud computing. Elsevier Inc.

IV. Ghomi, E. J., & Rahmani, A. M. (2017). Load-balancing algorithms in cloud computing: A survey. Journal of Network and Computer Applications, 88 (March), 50–71. https://doi.org/10.1016/j.jnca.2017.04.007

V. Goudarzi, Z., & Faraahi, A. (2014). Effective load balancing in cloud computing. International Journal of Intelligent Information Systems, 3 (6), 1–9. https://doi.org/10.11648/j.ijiis.s.2014030601.11

VI. Hwang, J., Wu, S. Z. and F. y, & Wood, T. (2013). Benefits and Challenges of Managing Heterogeneous Data Centers. In 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013) (p. 7).

VII. Joshi, R. (2018). Study and Comparison of VM Scheduling Algorithm in Cloud Computing Using CloudSim Simulator. International Journal for Research in Applied Science and Engineering Technology, 6 (5), 1751–1757. https://doi.org/10.22214/ijraset.2018.5285

VIII. Kanungo, P. (2016). Design Issues in Federated Cloud Architectures. International Journal of Advanced Research in Computer and Communication Engineering, 5 (5), 937–939. https://doi.org/10.17148/IJARCE.2016.55229

IX. Kuechler, B., & Petter, S. (2004). DESIGN SCIENCE RESEARCH IN INFORMATION SCIENCE.

X. Makwe, A., & Kanungo, P. (2016). A Survey of Scheduling Policies in Cloud Computing Environment. International Journal of Computer Trends and Technology, 34 (4), 169–173. https://doi.org/10.14445/22312803/ijcett-v34p131

XI. Prajapati, K., Raval, P., Karamta, M., & Potdar, M. (2013). Comparison of Virtual Machine Scheduling Algorithms in Cloud Computing. International Journal of Computer Applications, 83 (15), 12–14. https://doi.org/10.5120/14523-2914
XII. Singh, A. (2015). A Review on Existing Load Balancing Techniques in Cloud Computing. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 4 (7), 16–24. https://doi.org/10.5120/ijca2015905539

XIII. Smit, M., Simmons, B., & Litoiu, M. (2013). Distributed Application-level Monitoring for Heterogeneous Clouds using Stream Processing.

XIV. Thakur, P., & Shrivastava, D. K. (2015). Interoperability Issues and Standard Architecture for Service Delivery in Federated Cloud: A Review. In IC 2015 International Conference on Computational Intelligence and Communication Networks (pp. 908–912). https://doi.org/10.1109/CICN.2015.179

XV. Toosi, A. N., Calheiros, R. N., & Buyya, R. (2014). Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey. ACM Computing Surveys, 47 (7), 57.