International Journal of Cloud Applications and Computing
Volume 13 • Issue 1

An Efficient ECK-Secured FCM-Based Firefly Optimization Algorithm for Dynamic Resource Sharing in Multi-Tenant SaaS Service Clouds

Pallavi G. B., B.M.S. College of Engineering, India & Visvesvaraya Technological University, India

ABSTRACT

Multi-tenancy in cloud computing is one of the foremost approaches to share one application instance among different customers and it is generally used by Software as a service (SaaS) providers. The main objective of the proposed work is to minimize the down time of virtual machines essential for resource provisioning using cluster based secure dynamic resource sharing. The proposed secure dynamic resource sharing approach allocates the service tenants to matched Virtual Machines(VMs) and allocates the VMs into physical host machines using the elliptic curve key based firefly optimization approach. First the functional characteristics of service users are grouped into clusters using FCM (Fuzzy C_means clustering) algorithm as tenants. After clustering, the tenant users are checked for authorization with the help of elliptic curve key value. When the users in the tenants are authorized then the grouped services are scheduled dynamically using the firefly optimization algorithm. The result of the work is appraised in terms of resource utilization, execution time, speed, and speedup.

KEYWORDS

Authentication, Cloud Computing, Firefly algorithm, Fuzzy C-Means, Multitenancy, SaaS

1. INTRODUCTION

Cloud computing is an emerging technology proposed to support different storage and computing services accomplished on the internet. This technology can be utilized for authorizing comfortable, on-demand network connection towards an allocated puddle of configurable computing resources. For example, server, storage, networks, application and services which could be rapidly provided and discharged using minimum service provider interaction and management force (Armbrust et al., 2010; Bezemer & Zaidman, 2010). Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Services (PaaS) are the various types of services used in cloud computing. Cloud computing distributes substructure, stage, and software and is accessible as subscription-based services in a pay-as-you-go ideal to customers (Guo et al., 2007).

DOI: 10.4018/IJCAC.319033

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

SaaS is an innovative delivery model of software that provides a single application for multiple users via internet. This service is beneficial to IT companies as they can use the applications without the need to purchase and maintain their own IT infrastructure. Also, service providers achieve full economy of scale by hosting such SaaS application using a multitenant model.

The SaaS suppliers, acquiring the power to scale up or down the operation to solitary devour and pay for the resources that are essential in that period of point and it is an enticing ability, and it will be lower cost than functioning on hardware from conventional treating (Ju et al., 2012).

Multi-tenancy in SaaS is a critical technology in which multiple tenants are granted computing resources at the same period and reached great effectiveness of operation (Kullback & Leibler, 1951).

Multi-tenancy being a hierarchical model, suitable approaches are imposed on the virtual machines at each stage noted to great control and separation of customers. The inordinate management of resources and arranging the customers at different stages according to their necessities are the benefits of multi-tenancy (Kwok & Mohindra, 2008). A multi-tenant web application framework is used to maintain the heritage, changes, and runtime customizations of customer interfaces providing different client-side web operation technologies (Momm & Theilmann, 2011). Resource allocation is performed on the treated systems by utilizing Service Performance Isolation Infrastructure (SPIN). Multi-tenancy is maintained by executing SPIN model and indicated the segregation effectiveness on the Trade6 benchmark (Turner et al., 2003). Service level agreement (SLA) based multi-tenant focused on identifying, monitoring, and scheduling design is used to reduce the irregular utilization of system resources (Wu et al., 2011)).

Multi-tenancy is focused on maintaining huge customers through databases and examples. OTPP viz., On-line Tenant Placement Problem is a problem in the SaaS, and it is reduced by a novel resource evaluation of consumption model for multi-tenant design and to discover the difficulty of OTPP (Yang et al., 2011). To determine the position of tenants a hybrid technique is used which is denoted as Tenant Placement Strategy (TPS). Heuristic technique, genetic algorithm, case-based reasoning (CBR) is utilized in the TPS (Zhang et al., 2010). Multi-tenant operations are arranged on the normal relational databases by utilizing an innovative hybrid schema-sharing method (Zhu et al., 2011).

The functional cost of SaaS is decreased by using multi-tenants. But the possibility of shared impact of one tenant may fall onto the performance skilled by others. Grouping of feedback control loop-based controller and traditional scheduling machine is used to confirm performance isolation although the quicker response towards workload variations and conveniently using the system than restraint period of cycle's (Kolodner et al., 2011). The significant challenge in the multi-tenant SaaS operation is variability. Two-stages decoupling technique is used to minimize the variability: representative of the operation deviation as a noticeable variability model is the first stage, and the second stage involves the selection of suitable application layers to control every variation (Arora et al., 2012). A multi-layer ontology-based intelligent customization scheme is used to maintain the variability of SaaS operation and tenants' necessities (Kumar et al., 2012). Modifying, and imposing the safety necessities empowered by the service provider is tolerated by a Tenant Oriented SaaS Management Architecture (TOSSMA) (Grobauer et al., 2011).

A number of schemes are designed for better attainment of system performance (Wang et al., 2009). SaaS aims at providing a greater experience in terms of customization and configuration of applications. A metadata driven SaaS model is operated to monitor tenants, supervise them and provide large information management services. Memcached model is used to enhance database performance (Bezemer et al., 2010). A SaaS based on the model-driven technic is utilized to illustrate service customization, integration, and multitenant-aware (Kumara et al., 2015). The energy Aware Multi-tenant Application (EAMA) is utilized to diminish the energy utilization and the selection of servers at the data center. This method is active for reducing power utilization (Zongshui et al., 2015)).

1.1 Significant Contribution

The proposed work has the following significant contribution:

- To develop a mathematical model considering the number of tenants, resources (CPU) available etc., based on which dynamic resource allocation should be reached.
- A cluster of functional attributes can be applied to allocate the resources among tenants dynamically.
- To implement an active tenant authentication system to control access and to allocate resources among tenants and secure data and meta-data of tenants.
- To achieve dynamic resource sharing by using the tenant services, which do not utilize all the resources in an efficient way of in the multi-tenant operation.
- To allocate resources, resource level metrics are developed, and it is used to achieve the guaranteed performance among tenants.

The subsequent sections are ordered as follows: In section 2, Related work is discussed; In section 3 formulation of the problem is presented. Mathematical modelling of multiple service tenants is explained in Section 4; Proposed methodology is discussed in Section 5; The experimental results and analysis are presented in section 6 and in section 7 the conclusion and future work is discussed.

2. RELATED WORK

A multi-tenant structure has been adopted by SaaS suppliers in order to attain economies of scale. The functional costs are minimized by increasing the resource sharing among multiple customer tenants. Consequently, the performance segregation among tenants would be difficult. Stefan Walraven *et al* (2016) has developed an adaptive middleware for allowing SaaS suppliers to effectively impose various and contesting performance force in multi-tenant SaaS operations. The time limit at a fine-grained level, implemented quick response on altering conditions, and potential based performance force would be achieved by utilizing adaptive middleware, although the usage of resource was conserved in operation-level multi-tenancy. Open Stack and JBoss were utilized for implementing a prototype on the topmost level of the private cloud platform.

Qiong Zuo *et al* (2015) developed a novel tenant-based access control model depending on ARBAC i.e, Administrative Role-Based Access Control designed for Sub tenant architecture in SaaS. Autonomous Areas (AA) tree and AA were developed to define the tenants in autonomy as well as their allocation and segregation relations. The scalability and circulation representation of STA in SaaS system has enhanced the verification and effectiveness of authority on the autonomous decentralized area all over the system.

Allocation of partner services among tenants at runtime are utilized to obtain economies of scale in a service-based SaaS operation as a successive technique to the Single Instance Multi-tenancy (SIMT) model. But performance difference and runtime distribution using tenant specific function in any SIMT model is a difficult task. Indika Kumara *et al* (2017) understand the SIMT operation using a batch of service network (SN) software and virtual service networks (VSNs). By utilizing Software Defined service Networking (SDSN), the VSNs share the SN. According to their interoperability and ability, the SN attaches a set of services. A subset of the services beneath a specific structure and a control policy in the VSN reaches the necessary tenants.

In cloud computing, large scale customized tenants are supported to utilize one code base which is facilitated by a SaaS model and which in turn operates based on multi-tenant architecture. Nonetheless, the poor performance and the operating cost would be the difficulty of the multi-tenant structure. Wenbo Su *et al* (2015)) had developed a stable SLA aware tenant placement algorithm based on the multi-tenant queuing network model. This algorithm is powerful in approximately 90% of the simulation correlated with heuristic algorithms.

Wide development range of sophisticated solution of the cloud, ranging as of SaaS-based solution for the supervision of business was developed in a contemporary generation, with quickness, elasticity, and scalability as the distinguishing aspect of mass customization. Octavian Morariu *et al*

(2015) implemented the shop floor devices and virtualized manufacturing execution system (vMES) for enhancing maintainability, quickness, and scalability factors and to minimize the operational cost of the constructing systems. Rapid switching among various rendition of binding resources, configurations, and workloads was performed by utilizing shop floor profiles.

3. PROBLEM FORMULATION

Nowadays cloud computing plays a vital role in the computing environment due to its availability and flexibility of lower cost computing resources. In SaaS model, the hosted applications are retrieved through internet. The most significant feature of SaaS is multi-tenancy. Many users can share a single application instance in multi-tenant applications. Hence, SaaS providers significantly decrease delivery cost for a bulk of tenants and reduce operations. The goal of multi tenancy is to achieve cost effectiveness by supporting multiple tenant users simultaneously. Mediation, virtualization and sharing are the three multi tenancy improving approaches. The full potential of multi tenancy is achieved by solving two issues such as resource sharing and security isolation. The software, hardware and management costs are reduced by sharing resources among tenants. Interference, conflict and potential invalid access among tenants are prevented in security isolation. In the proposed work, the main concentration is towards resource sharing and security to achieve the complete support of multi tenancy. Security is an important concern in multi tenancy which ensures isolation among tenants in a shared environment and avoids possible malicious attacks. When resources are dynamically allocated in cloud environments, idle time needed to restart VMs for re-allocation is a key issue. Lessening this downtime is an important factor in order to meet the necessities of service tenants.

The objective of the work can be therefore articulated as follows:

The resource necessity R(i) of a tenant is stated as a function of the following attributes when the attributes of service tenants are given:

$$R(i) = \left\{T(t_i),\ P(t_i), RT(t_i), S(t_i)\right\} \tag{1}$$

Where $T(t_i)$ is the Throughput

$P(t_i)$ is the Productivity

$RT(t_i)$ is the Response Time

$S(t_i)$ is the Scalability

Specified the historical behavior of resource necessities of the tenants, understanding their resource necessities we need to cluster the service tenants. Then we have to allocate the resources without possible malicious attack in the shared environs dynamically.

4. MATHEMATICAL MODELLING OF MULTIPLE SERVICE TENANTS

A tenant is defined as an operation of SaaS, instantiated to fulfil a customer demand. An essential specification for a SaaS application is to authorize tenants to access collective resources. Tenants may not require resources tenant may at all times of the day. Therefore, depending on when a tenant is not functioning and when it is functioning, the tenant's state fluctuates and the resource necessities of a tenant are varied when it is in various states. A SaaS component involves an assortment of services and a bunch of tenants that work together to inspect various SaaS structures.

Let n be the considered service tenants where each such tenant includes 'f' number of functions. A service tenant termed t_i is expressed as t_i Þ f_j which are sets of input and sets of output, where i ranges from1, 2...n and j ranges from 1, 2...k. The attributes of service tenants which are feasible to impact the method of estimating and resource sharing are-

Throughput (S): Throughput is the average number of worklets executed per unit time. Also, WIPS is mentioned as the average number of web interactions per second. The velocity is recognized as throughput by TPC-W benchmark, measured by millions of web interactions per second (WIPS).

Productivity (P): Cloud service performance per unit cost, (TFlops/$, WIPS/$, etc.). The term $/WIPS is utilized and is mentioned to the total cost of the SUT separated by the number of WIPS measured throughout the Shopping Interval.

Response Time (RT): Time expired throughout the job execution or program, (sec., hours)

The Web Interaction Response Time (WIRT) is described by:

WIRT = R2 - R1
With R1 and R2 calculated at the EB.
R1 = measured time ahead of the first byte of the first HTTP operation of the web interaction is sent by the EB to the System Under Test (SUT);
R2 = measured time afterward the last byte of the last HTTP reaction that finalizes the web interaction is accepted by the EB from the SUT.

The word Emulated Browser (EB) is utilized to mentioned the unit (example, a strand or a process) imitating a customer contact through a Browser by sending and receiving HTML satisfied through HTTP and TCP/IP over a network connection (e.g., a socket) to the SUT.

Scalability (Sb): The capability to expand resources for increase in system performance. The cloud scalability is measured by the efficiency of a cloud system. The scalability is directly proportional to productivity.

Latency (L): The delay period started from submission to receiving the initial reaction (Sec). The word Think Time is used in this regard to mention the time elapsed from the last byte accepted by the EB to complete a web interaction till the first byte sent by the EB to appeal the next web interaction.

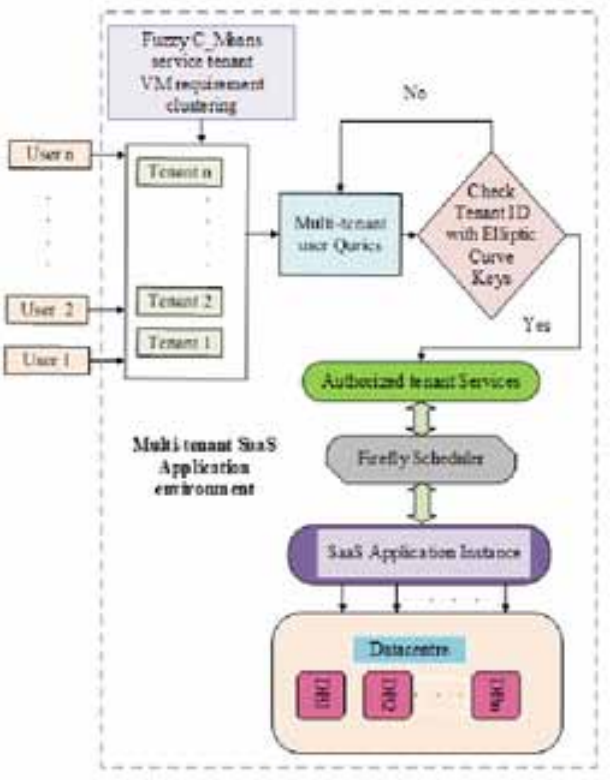
The Think Time (TT) is described by:

TT = T2 - T1
With T1 and T2 are measured at the EB.
T1 = measured time after the last byte of the last reaction page is received by the EB from the SUT.
T2 = measured time ahead the first byte of the next appeal is sent by the EB to the SUT.

5. PROPOSED FUZZY C MEANS MULTI-TENANT CLUSTERING BASED FIREFLY DYNAMIC RESOURCE SHARING WITH ELLIPTIC CURVE KEY AUTHENTICATION METHODOLOGY

The process flow of proposed Multi-tenant SaaS Operation Environment is shown in Figure 1. The major goal of this work is to allocate the resources dynamically and securely. The proposed work projects to build a Cluster-based optimized scheduling policy for dynamic resource sharing. Dynamic Resource Allocator assigns resources like CPU from our projected dynamic resource sharing methodology by employing the Firefly optimization algorithm. This research work employs an effective tenant authentication model which empowers the customers to achieve access to shared resources in a secured method. So, an effective key based authentication model is constructed for effective allocation of the resources.

Figure 1.
Process flow on proposed Multi-tenant SaaS operation environment



We contemplate a couple of service tenants using their initial resource necessities and exhibit how the resource necessities are coordinated. Based on the tenant resource necessities coordinated with the VM structure and classified into large and small by utilizing the fuzzy c-means clustering. Every tenant acquiring various resource necessities based on the tenant resource necessities must be added the finite number of VMs. Further, after successive authentication, Firefly algorithm is used to allocate the resources dynamically to the tenants. The tenants can access resources based on cryptographic algorithm ECC while accessing SaaS application instance.

5.1 Fuzzy C means (FCM) Clustering

A bunch of service tenants that have been established and provisioned by using a service provider is taken as T and the quantity of resources accessible to the provide is taken as R. Specified a set of service tenants with various resource necessities and a set of changeable capacities of resources, we can assign resources toward service tenants in such a manner where service tenants whose resource necessities are contented is maximized and the amount of resource misuse is lessened. Let C_t stand for the set of attributes that impact on resource sharing for every service tenant t. Allocation of service tenant to corresponding resources is possible when all of its attributes are satisfied. Therefore, with n representing any given time instant, the function (with the objective to maximize) determining the number of service tenants allocated to resources is given as follows:

$$f(n)=\sum_{t\in T}\begin{cases}1, & if\ c_t\ is\ satisfied\\ 0, & if\ c_t\ is\ not\ satisfied\end{cases} \tag{2}$$

The Fuzzy C-means (FCM) clustering operation is utilized for clustering and examining bulky data blocks. The algorithm obtained a data set and a number of predefined clusters, in which the data set is to be separated. Every data point in the set is related with all the accessible clusters with some "degree" at the end of the FCM method. This degree is described as "membership" to individual clusters. This means that a data point exists to several clusters rather than one. The clusters are achieved by iterative segregation of the data set. The method will complete when the objective function becomes diminished.

$$P_h=\sum_{i=1}^{n}\sum_{j=1}^{C}u_{ij}^{h}\left\|v_i-c_j\right\|,\ 1\le h\le\infty \tag{3}$$

Where C represents the number of clusters and Fuzziness Exponent h is a real number and its value is preferably set to 1 and the number of data entities is denoted by n, the degree of membership in data point is represented by u_{ij} and d-dimension cluster center C is represented by c_j. The i[th] term of d dimensional measured data is v_i and ||*|| is any norm articulating the analogy among the center and any data computed. The esteem to the Euclidean distance between the data point N_j and medium of the cluster C_i, the objective function is decreased.

In this iterative optimization of the impartial function, the fuzzy segment is processed. Modernize the membership d_{ij} and the cluster center c_j is disposed by:

$$d_{ij}=\frac{1}{\sum_{k=1}^{c}\left(\frac{\left\|v_i-c_j\right\|^2}{\left\|v_i-c_k\right\|^2}\right)^{\frac{2}{h-1}}} \tag{4}$$

Where, $\left\|v_i-c_j\right\|$ is the range beginning from the point i to present cluster center j, $\left\|v_i-c_k\right\|$ is the range beginning from point i to another cluster center k.

$$c_j=\frac{\sum_{i=1}^{N}u_{ij}^{h}*v_i}{\sum_{i=1}^{N}u_{ij}^{h}} \tag{5}$$

When $\max\left\{\left|u_{ij}^{k+1}-u_{ij}^{k}\right|\right\}<\varepsilon$ is achieved, then only the conclusion of repetition will occur. Where k the repetition stages and ε is an ending norm among 0 and 1. This practice assembled to a saddle point P_h or a local minimum.

The FCM method beginning with the initialization of the partition matrix $\mu_{ij}(0)$, at the stage k the cluster centers are modernized by utilizing Eq.5. In which, $\mu_{ij}(k)$ is modernized by utilizing Eq.2, and the FCM method is continual till it converges or Eq.2 holds.

The process is combined by subsequent stages:

Algorithm 1: Pseudo code for FCM clustering

1. Arbitrarily choose cluster center.

2. Initialize $U = \left[u_{ij}\right]$ matrix $U^{(0)}$

Compute the d_{ij} by utilizing:

$$d_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{\left\|v_i - c_j\right\|}{\left\|v_i - c_k\right\|}\right)^{\frac{2}{h-1}}}$$

3. At step-k: compute the centers vectors $C^{(k)}=[c_j]$ with $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{h} * v_i}{\sum_{i=1}^{N} u_{ij}^{h}}$$

4. Modernize $U^{(k)}, U^{(K+1)}$

$$d_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{\left\|v_i - c_j\right\|}{\left\|v_i - c_k\right\|}\right)^{\frac{2}{h-1}}}$$

5. If $\left\|U^{(k+1)} - U^{(k)}\right\| < \varepsilon$ or the minimum P is reached, then STOP; otherwise return to step 2.

5.2 Elliptic Curve Key (ECK) Authentication

ECC (Elliptic curve cryptography) is one of the techniques to public key cryptography grounded on the arithmetical form of elliptic curves over limited fields. This cryptographic scheme is comprised of many steps. In this work only the key generation step is carried out for authenticate users. Generator point or base point choice in ECC is the main stage for its safety. The base point of ECC is assumed below,

$$E : z^2 = x^3 + ax + b \tag{6}$$

Here a and b are integers, satisfy the condition is $4a^3 + 27b \neq 0 \bmod p;$ and here p is the main number and include a point called point at infinity.

Key generation:

Key generation is one of the major factors where we have to make both public key and private key. To encrypting the message by utilizing receiver's public key at the sender side and at the receiver side can decrypt its private key.

Utilizing the successive equation, we have to generate the public key,

$$Q = d * P \tag{7}$$

Here'd' is the random number that we have to select surrounded by the range of one to n-1. 'P' is the point on the curve.'Q' denotes the public key'd' denotes the private key

5.3 Firefly Optimization for Dynamic Resource Scheduling

Let r(t) represent the resource necessity of a service tenant t and is modelled by the attributes in Ct, ∀ t Î T. Thus, the quantity of resources wasted for the period of an allocation method can be modelled as follows:

$$w(n) = \sum_{t \in T} \begin{cases} 0, & if\ a_t = r(t) \\ a(t) - r(t), & if\ a_t > r(t) \end{cases} \tag{8}$$

Here, a(t) mentions to the quantity of resources shared to a service tenant t Î T. The objective is to diminish the w(n)

Service tenants acquire various resource necessities must be added to a limited number of VMs in a manner that it diminishes the quantity of VMs utilization. A set of service tenants (T1, T2,....,Tn) are contemplated alongside using their initial resource necessities and display how the resource necessities are coordinated. Service tenants are coordinated by using the VM configuration table and classified into small and large VMs rendering to their resource necessities. The synchronized service tenants are added to active VMs based on the current VM usage and maximum consumer handling capacity. The unparalleled service tenants are allocated to newly designed VMs.

The ceremonial description of VM placement on the real host is done using smallest remaining resources capacity. Service tenants are assigned to the VMs that are previously functioned based on the identical configuration. We arrange all the VMs in reducing order based on CPU necessity. Every time VMs are located into real host machine by utilizing firefly algorithm. This algorithm imputes VMs to the smallest residual host if it fits. A new host is designed if newly provisioned VMs cannot fit in any initialized host. For the reason that all the VMs and real hosts are investigated for every placement stage, the algorithm has a functioning time of O (nm), where n is the number of VMs and m is the number of hosts.

Firefly algorithm has three major rules based on the flashing behavior of the fireflies. They are,

1. All fireflies are assumed as similar gender, and then only the fireflies are shift in the direction of more enticing and optimistic fireflies irrespective of their gender.
2. The illumination is proportional to attractiveness and light intensity. The illumination of the firefly will be reduced when range between the two fireflies are increased. The less optimistic firefly will be shift in the direction of optimistic firefly. If there is no optimistic firefly than a particular firefly then it will shift in random direction.
3. The prime objective function is to find the illumination.

Attractiveness or Illumination: The attractiveness or illumination of a firefly is relevant to the light intensity of the adjacent fireflies. The attractiveness differs with the range between the fireflies can be resolute as,

$$\mu = \mu_0 e^{-\gamma r^2} \tag{9}$$

Where μ_0 is the attractiveness at r=0

Range and Movement:

The range between the two fireflies is acquired by the Cartesian distance,

$$r_{ij} = \left\| x_i - x_j \right\| \tag{10}$$

Here, i and j are the two fireflies and r_{ij} is the range between the two fireflies.

When the firefly with less illumination is shifted in the direction of firefly by using greater illumination. The movement of the firefly is described as,

$$x_i^{\,t} = x_i^{\,t-1} + \mu_0 e^{-\gamma r_{ij} 2} \left(x^t_{\,j} - x^t_{\,i} \right) + \alpha \varepsilon_i \tag{11}$$

Here, the three terms represent some points. The first term is the current position of the firefly and then the second term describes the attractiveness of a firefly based on illumination and the third term to mention the random movement of the firefly.

6. EXPERIMENTAL RESULTS AND ANALYSIS

The experiments are coded on a PC using duo 2.53 GHz CPU and 2G Memory running a Windows 7 operating system using Java 7. In this paper, TPC-W benchmark dataset is concentrated with multi-tenancy support for cloud environments. In view of the environs of multi-tenant SaaS application, the resources are maintained steady, tenant information will develop the changes in business necessities, and the number and configuration servers may change repeatedly to contain large quantities of tenants and users with decent SLAs. The execution of the projected technique for the SaaS operation on a real cloud and with the benchmark dataset. Our empirical data set consists of tenant information with response time, scalability, throughput, and productivity.

The simulated experimental results are measured using the performance metrics accuracy, waiting time, response time, idle time and computation time. The performance is evaluated for the proposed work is compared with the existing dynamic resource sharing processing techniques. The proposed algorithm is comprised of FCM clustering algorithm with the firefly resource dynamic scheduling algorithm. So, the proposed work is compared with k-means clustering with the firefly optimization algorithm and Fuzzy C_means clustering with Artificial Bee Colony (ABC) algorithm. Around 100, 500, 1000 and 1168 tasks are given as input to the clustering algorithm so that the tenants are formed as clusters and that clusters are scheduled using the proposed scheduling algorithm. The simulated experimental results are measured by utilizing the following performance metrics.

Execution Time: The total time elapsed to implement the task in seconds or hours shown in Figure 2.

Speed: The number of operations implemented per second shown in Figure 3.

Speedup: Speed gain of utilizing the highest processing nodes over a single cluster. The speedup is defined by the proportion of time taken to implement a single task to the time taken to implement the whole number of clusters shown in Figure 4.

Figure 2.
Performance comparison for execution time

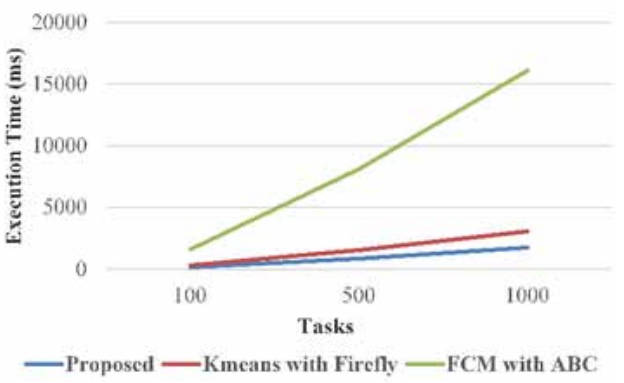


Figure 3.
Performance comparison for speed

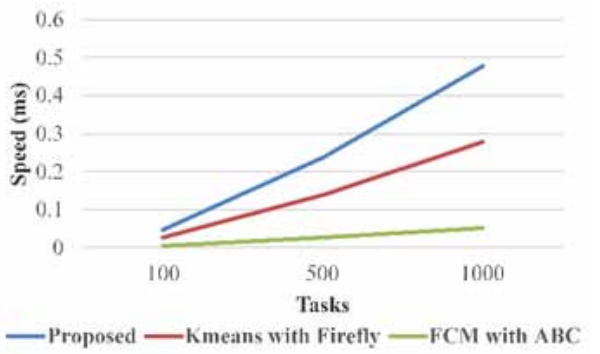
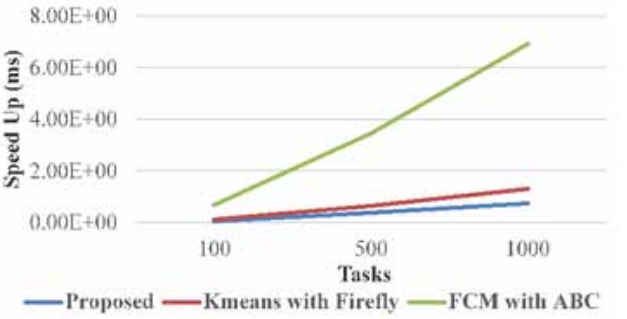
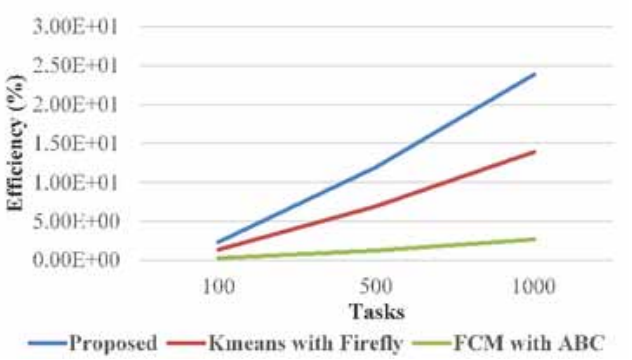


Figure 4.
Performance comparison for speed up



Efficiency (Resource utilization): Resource utilization denotes the antiquity of resources utilized by a service tenant which was previously in the active state. It is considered in terms of CPU utilization percentage and other resources. So, resource utilization is favored the best and it is taken out for the correlation to show the efficiency. The efficiency is represented as the percentage of peak

Figure 5.
Performance comparison for resource utilization or efficiency



performance. It can be computed as the ratio of speedup to the total number of clusters. The results are tabulated in Figure 5.

The resultant method will provide better performance compared with the existing method.

7. CONCLUSION

The proposed work presents a secure dynamic resource sharing in a multi-tenant SaaS service cloud environment. SaaS is a novel software sharing model in cloud computing which support multi-tenants and loosely coupled services over the internet. The presented work implements a FCM clustering algorithm to group the users as tenants. Then the tenant users are encrypted using ECC (Elliptic Curve Cryptography) to securely access the data through SaaS application instance. Firefly optimization algorithm is used to effectively allocate resources among the tenant's requests. Experimental results evaluate the performance of the proposed method with the TPC-W benchmark dataset which improved in terms of execution time, speed, speed up and efficiency.

REFERENCES

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., & Rabkin, A. I. (2010). A view of cloud computing, Communications of the ACM, vol. 53, no. 4, pp. 50–58, .

Arora, P., Wadhawan, R. C., & Ahuja, E. S. P. (2012). Cloud Computing Security Issues in Infrastructure as a Service. *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(1).

Bezemer, C.-P., & Zaidman, A. (2010). Multi-tenant saas applications: maintenance dream or nightmare? Proceedings of *the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE*), (pp. 88–92). ACM. doi:10.1145/1862372.1862393

Bezemer, C. P., Zaidman, A., Platzbeecker, B., Hurkmans, T., & Hart, A. (2010). Enabling multi-tenancy: An industrial experience report. InSoftware Maintenance (ICSM). *IEEE International Conference* (pp. 1-8). IEEE.

Grobauer, B., Walloschek, T., & Stocker, E. (2011). Understanding cloud computing vulnerabilities. *IEEE Security and Privacy*, *9*(2), 50–57. doi:10.1109/MSP.2010.115

Guo, C. J., Sun, W., Huang, Y., Wang, Z. H., & Gao, B. (2007). A framework for native multi-tenancy application development and management. In E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, ECommerce, and E-Services, (pp. 551– 558). IEEE.

Ju, L., Sengupta, B., & Roychoudhury, A. (2012). Tenant onboarding in evolving multi-tenant software-as-a-service systems. In *19th International Conference on Web Services (ICWS),* (pp. 415–422). IEEE.

Kolodner, E. K., Tal, S., & Kyriazis, D. (2011). A cloud environment for data-intensive storage services. *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, (pp. 357–366). IEEE doi:10.1109/CloudCom.2011.55

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, *22*(1), 79–86. doi:10.1214/aoms/1177729694

Kumar, M.D.K., Rao, G.V., Rao, G.S. (2012). *Cloud Computing: An Analysis of Its Challenges & Security Issues*. Springer.

Kumara, I., Han, J., Colman, A., & Kapuruge, M. (2017). Software-Defined Service Networking: Performance Differentiation in Shared Multi-Tenant Cloud Applications., *IEEE Transactions on Services Computing*, *10*(1), 9-22.

Kumara, I., Han, J., Colman, A., & Kapuruge, M. (2015). Software-Defined Service Networking: Runtime Sharing with Performance Differentiation in Multi-tenant SaaS Applications. In *Services Computing (SCC), IEEE International Conference* (pp. 210-217). IEEE.

Kwok, T., & Mohindra, A. (2008). Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications. Service-Oriented Computing– ICSOC 2008, (pp. 633–648). ACM.

Momm, C. & Theilmann, W. (2011). A combined workload planning approach for multi-tenant business applications. In *Computer Software and Applications Conference Workshops (COMPSACW),* (pp. 255–260). IEEE.

Morariu, O., Borangiu, T., & Raileanu, S. (2015). vMES: Virtualization aware manufacturing execution system. *Computers in Industry*, *67*, 27–37. doi:10.1016/j.compind.2014.11.003

Peng, G., Wang, H., Dong, J., & Zhang, H. (2016). Knowledge-based resource allocation for collaborative simulation development in a multi-tenant cloud computing environment, IEEE transactions on services computing. *Vol. PP*, (99), 1.

Su, W., Hu, J., Lin, C., & Shen, S. (2015). SLA-Aware Tenant Placement and Dynamic Resource Provision in SaaS. In *Web Services (ICWS), IEEE International Conference,* (pp. 615-622). IEEE.

Turner, M., Budgen, D., & Brereton, P. (2003). Turning software into a service. *Computer*, *36*(10), 38–44. doi:10.1109/MC.2003.1236470

Wang, C., Wang, Q., & Ren, K. (2009). *Ensuring Data Storage Security in Cloud Computing*. IEEE.

Wu, L., Garg, S. K., & Buyya, R. (2011). Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. *In Cluster, Cloud and Grid Computing (CCGrid), 11th IEEE/ACM International Symposium,* (pp. 195–204). IEEE, ACM. doi:10.1109/CCGrid.2011.51

Yang, E., Zhang, Y., Wu, L., Liu, Y., & Liu, S. (2011). A hybrid approach to placement of tenants for service-based multitenant saas application. In Services Computing Conference (APSCC), (pp. 124– 130). IEEE.

Zhang, Y., Wang, Z., Gao, B., Guo, C., Sun, W., & Li, X. (2010). An effective heuristic for on-line tenant placement problem in saas. In *Web Services (ICWS), IEEE International Conference,* (pp. 425–432). IEEE.

Zhu, J., Gao, B., Wang, Z., Reinwald, B., Guo, C., Li, X., & Sun, W. (2011). A dynamic resource allocation algorithm for database-as-a-service. In *Web Services (ICWS), IEEE International Conference* (pp. 564–571). IEEE.

Zongshui, X., Kong, L., Li, Q., & Cheng, P. (2015). Global Index Oriented Non-Shard Key for Multi-tenant Database. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM),* (pp. 831-836). IEEE. doi:10.1109/CIT/IUCC/DASC/PICOM.2015.123

Zuo, Q., Xie, M., & Tsai, W. T. (2015). Autonomous Decentralized Tenant Access Control Model for Sub-tenancy Architecture in Software-as-a-Service (SaaS). In *Autonomous Decentralized Systems (ISADS), IEEE Twelfth International Symposium*, (pp. 211-216). IEEE.