Location Based Optimal Package Selection in Multi-Cloud

K.S.Guruprakash [#1], S.SivaSathya [*2]

[#]Assistant Professor, Department of Computer Science and Engineering,
Veltech Dr.RR & Dr.SR University, Chennai, Tamilnadu, India
[1] ks_guruprakash@yahoo.com

[*]Associate Professor, Department of Computer Science,
Pondicherry University, Pondicherry, India
[2] ssivasathya@gmail.com

***Abstract*—Cloud computing is a ubiquitous platform that can be used to access services and resources. The requirements for accessing a cloud resource is minimal, however, the users of a cloud platform are initially required to configure the type of services they need to access in the cloud. Even though cloud platform is elastic, frequent upscaling is costly. This paper presents an effective technique that can be used to automatically identify user's requirements and to allocate appropriate packages depending on the requirements. Usage logs of the client is grouped geographically and group based requirements are identified. This helps to determine the resource requirement for each group which address the problem of underutilization or overutilization of resources. These requirements are passed to PSO, along with the packages offered by multiple cloud operators in the same region to identify the best package for the current requirement. Experiments conducted on this architecture proves the effective working nature of the system in minimal time and with low QoS difference.**

Keyword-MCDM, cloud package selection, Optimization, PSO, multi-cloud

I. INTRODUCTION

Adopters for cloud platform have grown enormously due to the increase in the requirements for automation of systems. However, satisfying them has prevailed to be a controversial issue. This is due to the inappropriate selection of resources, leading to either underutilization or overutilization of resources. Cloud computing is a distributed computing technique that presents the user with an infinite pool of resources that can be accessed with minimum requirements [1].

Cloud resources are not available as such for the users to access them. Instead, packages are provided to the user with varied configurations such that the users can select a configuration of their choice and the resources are assigned accordingly [2]. Packages are usually segregated as high performance and high storage, moderate performance and high storage, low performance and low storage etc. Several such combinations are provided to the user with varying levels of quality parameters. The quality parameters include performance, storage, reliability, robustness, availability etc. These packages are defined by the cloud service providers and are to be used as such, without any modifications [3].

The process of package selection plays a vital role in determining the satisfactory level of the user. The major difficulty faced during this stage is the identification of the appropriate package that suits the current requirements of the user [4]. This problem can actually be branched into two sub-problems namely; identifying the current requirements of the user and selecting the appropriate package that suits the user's requirements. Both these issues are to be performed manually. This acts as the major drawback of the selection process. Automation of these two processes has not been considered by the cloud providers, hence cloud users who are technically not well versed face this issue to a large extent. Inability of a user to identify their service requirements leads to selection of inappropriate packages and this in-turn leads to dissatisfaction.

In-order to overcome the issue of dynamic service requirement peaks, cloud servers incorporate the concept of elasticity by scaling up resources as and when required [5] [6]. The scaling process is automatically done, hence the user need not worry about the sudden increase in requirements. However, these automatic process scaling tends to be costly [7]. If these resource scaling spells are rare, they need not be bothered. However if they are frequent, a package change is usually recommended as obtaining a package with higher requirement is economical compared to frequent resource scaling [8]. As of now, identifying the appropriate package is a trial and error process, which starts with an educated guess that follows with the usage levels. However determining it automatically will lead to improved reliability of the cloud services [9]. This paper proposes a requirement identification and a package selection technique that can be used to dynamically identify the user's requirements and recommend appropriate packages corresponding to the requirements.

II. RELATED WORKS

Cloud computing is one of the sought after techniques due to its elastic nature. The major requirement for a cloud computing environment is to provide the appropriate requirement specifications such that the appropriate resource is allocated for the user. Several contributions exist in literature to identify the most optimal resource specifications for the current requirement. This section presents some of the most recent contributions towards cloud resource provisioning.

A summarization of the best practices for cloud optimization was presented by Ferry et al. In [10]. This technique stressed on the need for a model-driven optimization architecture for multi-cloud systems. A 2ehaviour framework used for automatic resource allocation in cloud (ROAR) was proposed by Sun et al. In [11]. This technique not only operates with its basis on resource allocation, it also provides techniques for optimization of the resource allocation decisions. ROAR is defined as a domain specific language that can be used to define the configurations of the web applications. This technique is presented as a cost optimal resource configuration system. Several other model based cloud provisioning systems include[12] [13]. Draheim et al. [12] proposed a real time approach that models the user's 2ehaviour to provision resources effectively. A Load Testing Automation Framework (LTAF) was proposed by Wang et al. In [8]. This technique has its major concentration on modelling user's 2ehaviour to test the load in each of the workflows and provision resources accordingly.

An automatic resource allocation technique that focuses on dependability and the security aspect of services was presented by Marrone et al. In [14]. This technique also uses a model driven principle to support cloud brokers in identifying the optimal configurations. It uses UML and Bayesian Networks to perform the optimization process. Energy efficient resource allocation is a major requirement currently due to the increased usage of resources. Some techniques that have their major focus on attaining energy efficiency were proposed by Beloglazov et al. And Hung et al. In [15][16]. A model based and energy efficient approach for resource allocation was proposed by Dougherty et al. In [17]. An energy efficient approach that concentrates on energy utility by incorporating it as a major parameter into its allocation algorithm was presented by Gupta et al. In [18]. A reinforcement learning based dynamic resource provisioning technique for cloud was presented by Bahrpeyma et al. In [19]. The major advantage of this technique is that it uses a reinforcement learning based approach that performs dynamic resource provisioning enabling the elasticity in a cloud environment. Further, its ability to deal with the uncertainty of the cloud environment is another major advantage towards using this approach. Several approaches exist in literature that deals with the uncertainty in a cloud environment to perform effective resource provisioning. Some of them include, a response time based provisioning strategy proposed by Islam et al. In [20], a model predictive control, used to improve efficiency, proposed by Zhang et al. In [21] and an anticipatory model proposed by Huang et al. In [22]

A dynamic resource provisioning system operating in a multi-agent architecture was proposed by Ayyoub et al. In [23]. This technique concentrates mainly in minimizing the cost and the time of operations for the customers. A technique concentrating on scientific jobs in cloud was proposed by Shi et al. In [24]. This technique proposes resource provisioning and task scheduling mechanisms to effectively perform scientific workflows in a cloud environment. An optimal resource provisioning technique that has its major concentrations on software, providing SaaS was presented by Li et al. In [25]. This technique minimizes the payment to VMs and maximizes its profit by accommodating more users to utilize the software service provided by them. Several such techniques exist in literature to perform resource provisioning. However, it is to be noted that the major requirements of all these approaches is to provision resources according to the user's requirements, where the users are required to define their own requirements. Automated user requirement identifications is not considered in any of the discussed approaches. A technique to provision resources based on the user's requirements was presented by Madhumathi et al. In [33]. This technique utilizes Ant Colony Optimization, a metaheuristic technique for the package selection process. The major concentration of this approach is to avoid vendor lock-ins to provide effective service to academic institutions.

III.GEOGRAPHIC LOCATION BASED OPTIMAL PACKAGE SELECTION IN MULTI-CLOUD

Automatic identification of a user's requirements is a feasible process, however it requires base data of the user and the usage scenario. This refers to the number and type of users for whom the service is provided, the geographic locations of access and the level of access in each of the locations. First time users will not be aware of these details initially, hence these are obtained as fuzzy inputs directly from the user. However, for users migrating to cloud environments these details can be mined from their usage/ web logs. This concept serves as the base for this paper. This paper proposes an architecture (Fig.1.) to effectively perform automatic package selection in a multi-cloud environment.

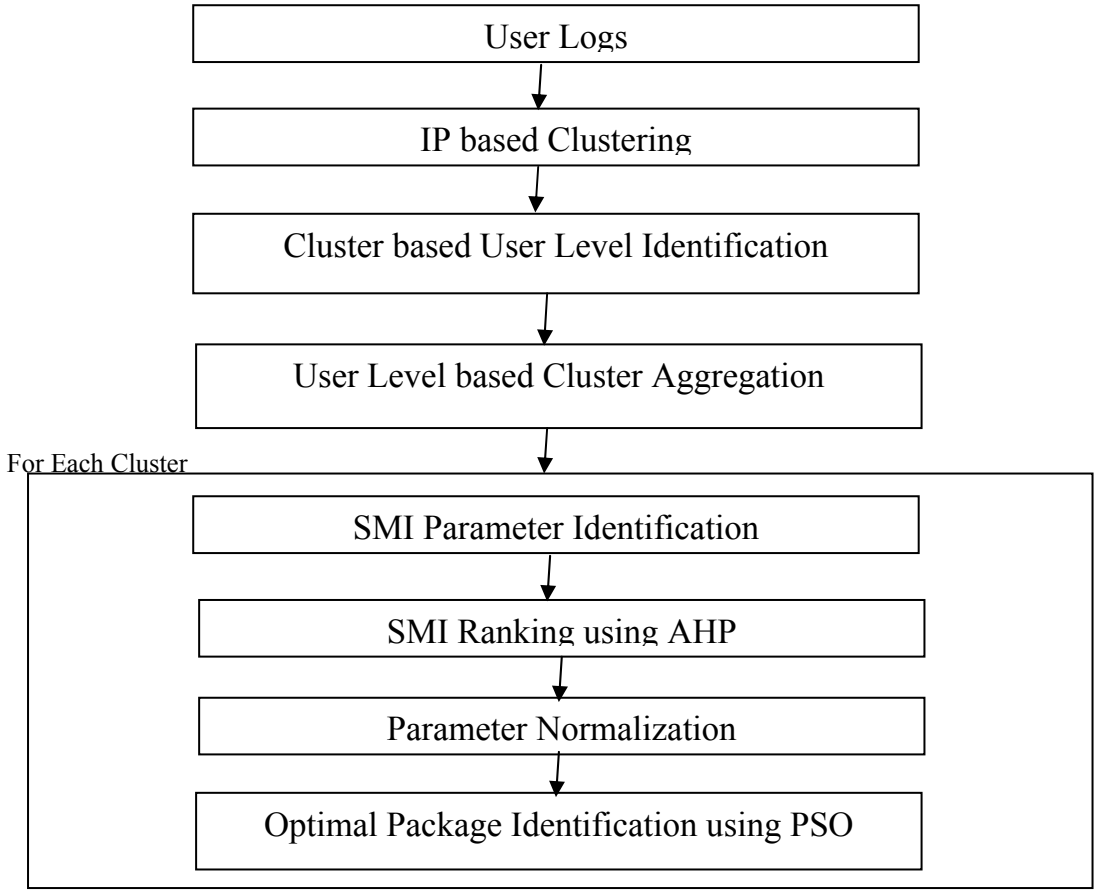

Fig. 1. Geographic Location Based Optimal Package Selection

The proposed architecture utilizes the usage logs of a user to identify their requirements. The process is carried out in three broad phases. The first phase deals with identifying the usage levels on the basis of geographical locations and grouping them. The second phase deals with identifying the SMI parameters for each of these groups, where the SMI parameters corresponds to the usage requirements for the group. The third phase deals with selecting optimal packages for each of these groups by considering a multi-cloud environment such that the requirements are matched to the maximum extent.

A. Geographic Location Based Usage Level Identification

Resource utilization levels corresponding to a user might actually correspond to their aggregated access levels. The resource has a high probability of being used over several geographic locations. Each of these locations tend to have varied demands. When using a single resource for shared access, capability of the resource is determined by the highest access requirement. On its absence, the resource is always underutilized. The proposed architecture addresses this issue by grouping the regions according to their level of access, then determining the resource requirement for each group. Though this scheme leads to granularity, appropriate resource allocations can be performed and the probability of up-scaling can be reduced to a large extent.

The initial process is to divide the dataset on the basis of its IP address. IP based data grouping is carried out, which aggregates the transmissions carried out from the same IP. The groups are then sorted on the basis of the IP, leading to groups with closer IP addresses occurring consecutively. Usage levels, in terms of the level of access is identified for each group.

This phase is followed by the aggregation phase. The aggregation phase analyzes the usage levels of each group and aggregates geographically close groups whose usage levels are less than the base threshold (thresh). The base threshold is a user defined parameter that determines the maximum usage level of the cloud resources that can be provided to each of the groups. This process finally ends up to provide the defined number of groups, and each of these groups exhibit a usage requirement defined by the user.

B. Cluster based SMI Parameters Identification

The base threshold defined in the previous phase provides a rough estimate for the aggregation process. However, in-order to provide appropriate allocation of cloud resources, the Quality of Service parameters must be identified from the web usage logs. Quality parameters considered for usage in the current architecture is presented in table 1. The QoS parameters are divided into two broad sections; parameters to be obtained from web logs and parameters that are to be obtained from the user. It has been discussed in the earlier sections that the QoS parameters can be obtained from the web logs, however, not all requirements can be directly obtained from the logs. Parameters such as security, portability, reliability, etc. needs to be explicitly obtained from the user. Five parameters are selected for user input in this paper. However, the architecture is flexible and several parameters can be added to the current list depending on the requirements. The process of identifying quality parameters from web logs has been proposed in the previous contribution by the authors. Fuzzified inputs are obtained from the users for the quality parameters categorized under user input. The inputs are varied with five levels, with 1 indicating the lowest and 5 the highest.

TABLE I. QoS Parameters Considered for Evaluation.

QoS Parameters Considered for Evaluation	Obtained From
Bandwidth (Bw)	**Web Logs**
Computation Capability (CC)	
Availability (Av)	
Correctness (Cr)	
Usability (Us)	
Reliability (Re)	
Variable computation load (Vc)	
Serviceability (Se)	
Latency (l)	
Security (S)	**User Input**
Portability (P)	
Reliable storage (Rs)	
Data Backup (Db)	
Customization (Cu)	

C. SMI Ranking using AHP (Analytical Hierarchy Process)

The next phase deals with ranking the quality parameters for identifying the weights associated with each parameter. Ranking parameters cannot be usually performed directly, as several issues are associated with them. Further, as the parameter set tends to increase, the difficulty in ranking process also increases. This paper uses Analytic Hierarchy Processing (AHP) to determine the ranks.

AHP [26] [27] is a collaborative technique used to make complex decisions. The decisions are usually made on the basis of the user's preferences. The application of AHP can be extended to several areas. In general, AHP can be used in any situation that requires a user to make decisions on the basis of several dependent or independent attributes. There exists two methods to identify weights using AHP. They are:

(i) Pair wise comparison

(ii) Direct user assigned weights.

1) Pair wise Comparison:

The pairwise comparison technique considers each pair of attributes and ranks them with respect to each other. In this paper, quality parameters such as availability, reliability, security, latency and so on are used as attributes. A square matrix is created, representing each attribute in the row and in the column. Each intersection is marked with the rank provided to the row element when compared with the column element. Since the diagonals correspond to the same attributes comparisons cannot be performed, hence the diagonal positions are provided with a value 1. This technique is useful if the process of ranking needs to be performed on a huge number of attributes, or if the user does not possess detailed knowledge about the QoS requirements of the current requirement. Since the process of cloud package selection faces both these issues, AHP can be considered to be the best candidate for ranking QoS parameters considered for cloud resource allocation. Comparisons are made on a 5 point scale and the comparison matrix depicting the priority set P is shown below.

$$P = \begin{bmatrix} w_1/w_1 & w_1/w_2 & \cdots & w_1/w_n \\ w_2/w_2 & w_2/w_2 & \vdots & w_2/w_n \\ \vdots & \vdots & \ddots & \vdots \\ w_n/w_1 & w_n/w_2 & \cdots & w_n/w_n \end{bmatrix} = \begin{bmatrix} 1 & s_{12} & \cdots & s_{1n} \\ s_{21} & 1 & \vdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & 1 \end{bmatrix} \text{....(1)}$$

Where S_{xy}represents the comparison score of x when compared with y and W_x/W_ycompares the attribute x against y. An attribute when compared to itself will return a value of 1. These values are then integrated to provide the final attribute weights (WA_i).

2) Direct User Assigned Weights

Though the user might not be able to appropriately provide ranks for all the QoS properties, several properties might be significant enough such that its rank can be directly determined by the user. Properties that are of least importance can also be ranked in this phase. Providing ranks for such attributes directly would be the best choice, as eliminating properties from the pairwise ranking process tends to reduce comparisons. Hence this process is performed prior to the pairwise comparison phase.

The weights obtained from AHP are used to identify the requirement levels of a particular package, specified by either the customer or the cloud provider while proposing packages. These requirement levels serve as the basis for identifying the fitness of a package.

D. Parameter Normalization

The process of attribute ranking provides appropriate weights for the attributes. These weights, in conjunction with the actual parameter values are used to determine the quality of service. The actual values of the quality parameters can occur in various ranges. Using them directly to obtain the final quality measure is not appropriate. Hence it becomes mandatory to transform them into a fixed range for effective operations. This paper uses min-max normalization [28] to perform the conversion process.

$$A' = \left(\frac{A - min\ value\ of\ A}{maxvalue\ of\ A - min\ value\ of\ A}\right) * (D - C) + C \qquad (2)$$

Where, A' contains Min-Max Normalized data one, A is the actual data and C and D are the pre-defined boundaries.

The final quality parameter is calculated using the weighted sum method [29], [30]. The weighted sum method calculates the quality of a requirement by aggregating the product of its weights and its corresponding normalized attribute values.

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j\, a_{ij,}\, for\ i = 1,2,3,\ldots.,m. \qquad (3)$$

Where w_j corresponds to the weight of an attribute and a_{ij} corresponds to the value the j^{th} attribute of a requirement scenario i.

E. Optimal Package Selection in MultiCloud using PSO

The process of identifying the optimal package is performed using Particle Swarm Optimization (PSO) [31], [32]. PSO is a metaheuristic technique that optimizes a problem by iteratively improving the candidate solution. The improvement is measured in the form of a fitness function that is used to select the best solution among the list of available solutions. PSO operates by moving the particles in the search space to find the optimal solution.

1) PSO Search Space Creation and Particle representation

Particles are the operating agents that identifies the solutions in the search space. A search space is created using the data containing package information. Package details correspond to the values provided to each of the quality parameter. Due to the usage of location based package assignment, multi cloud based packages are used for constructing the search space. The dimension of the search space is determined by the number of quality parameters used for analysis. Particles are represented using SMI Parameter values of package data and user requirement. PSO operates in three major phases. Distribution of particles forms the first phase of operation in PSO, followed by determining initial velocity and triggering the movement of the particles. This is followed by iteratively varying the velocity on the basis of the best solutions identified by the previous iterations and determining the convergence of the system

2) Particle Distribution and Movement

After the creation of the search space, the particles are usually distributed in the search space in random locations. However, the current process requires the user to find the best solution for a particular requirement. Hence, along with the package details, the current user requirements is also added as another node in the search space. All the particles are distributed in the requirement node. The process of particle movement is initiated by defining the initial velocity for the particles. The initial velocity is determined using the eq. 4

$$V_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|) \quad (4)$$

where b_{up} and b_{lo} are the upper and lower bounds of the search space.

A random velocity is assigned to each particle and the movement is triggered. After a single migration, the particles occupy a position in the search space that does not correspond to any node. This is due to the fact that PSO operates on a continuous space. However, the current process requires discrete particle movement. Hence the particles are discretized and moved to a defined node using the eq. 5

$$P' = \min\left(\sum_{j=1}^{n}\left(\sum_{k=1}^{d}\sqrt{(P_{ik} - N_{jk})^2}\right)\forall i = 1\ to\ p\right) \quad (5)$$

Where P_{ik} refers to the particle i's current location corresponding to dimension k, N_{jk} refers to the k^{th} dimension of node N_i.

3) Fitness identification and Optimal Package Selection

This process is followed by the fitness identification. PSO operates by using two fitness values namely; $p_{best}$and the g_{best}. The best solution identified by a particle so far is recorded as the $p_{best.}$ Each particle has its own best solution that it has visited. The best solution for the entire search space is determined by the $g_{best.}$ All the p_{best}solutions are compared and the best among them is set as the $g_{best.}$ Hence a search space only contains one $g_{best.}$ The initial iteration sets the current solution as the $p_{best}$for each particle. All the p_{best}values are compared and the best value is chosen as the $g_{best.}$ The comparison of solutions is performed using the fitness function. The fitness function for this approach is modified to incorporate the difference between the QoS values of the requirement and the current node representing the package. This marks the end of the initial iteration. After this stage, the velocity of the particles is determined by the p_{best} and the g_{best}values identified so-far. Velocity of a particle is calculated using eq. 6

$$V_{i,d} \leftarrow \omega V_{i,d} + \varphi_p r_p(P_{i,d} - X_{i,d}) + \varphi_g r_g(g_d - X_{i,d}) \quad (6)$$

where r_p and r_g are the random numbers, $P_{i,d}$ and g_d are the parameter best and the global best values, $X_{i,d}$ is the value current particle position, and the parameters ω, φ_p, and φ_g are selected by the practitioner. Current particle position is updated by following equation.

$$X_{i,d} \leftarrow X_{i,d} + V_{i,d} \quad (7)$$

The process of discretized movement and determining the $p_{best}$and $g_{best}$is continued until the termination criterion is met. The termination criterion is determined by either a stagnation condition or a time lapse. After deployment, a maximum time of operation is set by the user. After this time lapse, PSO is terminated and the solution contained in $g_{best}$is taken as the optimal solution. However in the production phase, termination criterion is defined by the stagnation behavior. A maximum termination (*maxTerm*) level is set by the user. If the g_{best}does not change for *maxTerm*times, the process is terminated. In this paper, the *maxTerm*is set to 1000. As the *maxTerm*value gets higher, the probability of local optima is reduced. After termination the value contained in $g_{best}$is taken as the most optimal package best suited to the current user requirement.

Algorithm (PSO based Optimal Package Selection):

1. *Search space boundary identification using package data and user requirement*
2. *Initialize number of particles p.*
3. *For each particle i=1…p*
   a. *Particle initialization on the user node using parameter values of package data and user requirement.*
   b. *p_{best} and g_{best} initialization*
   c. *Velocity initialization using equation (4)*
4. *Until the termination criterion is met perform the following*
   a. *For each particle i=1…p*
   i. *Generate r_p and $r_g$using normal distribution*
   ii. *Discretize particle movement using equation (5)*
   iii. *Identify the velocity of each particle using equation (6)*
   iv. *Update the particle's position to P' using equation (7)*
   v. *If p_{best}< current fitness*
      1. *Assign current fitness to be the p_{best}*
   vi. *If p_{best}<g_{best}*
      1. *Assign p_{best} as the new g_{best}*
5. *g_{best} contains the optimal package for the current user requirement*

Packages used for creating the search space is obtained from multiple service providers, such that the best provider, providing the best package is selected for each location.

IV. RESULTS AND DISCUSSION

A. Workflow and Dataset Analysis

The workflow corresponding to location based optimal package selection in multi-cloud is performed in two phases. The first phase deals with grouping requirements based on location and identifying quality parameters for each of the groups independently, and the second phase deals with identifying the best package for the current requirement in a multi-cloud environment. Access log dataset is used for the identification process. A sample screenshot of the access log is shown in Fig 2.

```
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET / HTTP/1.1" 200 9157
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/js/lowpro.js HTTP/1.1" 200
10469
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/css/reset.css HTTP/1.1" 200
1014
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/css/960.css HTTP/1.1" 200
6206
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/css/the-associates.css HTTP/
1.1" 200 15779
10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/js/the-associates.js HTTP/
1.1" 200 4492
```

Fig. 2. Sample User Log.

The access log is made up of seven basic components known as Common Log Format (CLF) as shown in Table II.

TABLE II. CLF Description.

Component	Description
10.223.157.186	IP address of the remote host which made request to the server
-	Client ID (Not available)
-	User ID (Not available)
[15/JUL/2009:15:50:35 -0700]	Timestamp. Time the server finished processing.
GET / HTTP/1.1	Request line from the client
200	Status code sent by server to client
9157	Size of the object returned to client

Client ID and User ID have been removed from the file to provide anonymity. The access log dataset used for this contribution is made up of 4.4 million transmission records. The timeline ranges approximately for 2.5 years.

B. Implementation Details

Both the phases correspond to very distinct requirements. Implementation for the first phase is carried out using Python. The access log is provided as input to the Python code. The transactions are read and IP address based grouping is performed. IP based sorting is carried out to bring entries with similar location together. The value for maximum density threshold (thresh) is set to 5000. Aggregations and splitting is carried out depending on this value and the final location based groups are created. Each of these groups is analyzed and their quality parameters are identified. User based quality parameters are obtained as fuzzy inputs and the final group based requirements are identified and written to property files.

The package selection phase is performed next. This process is implemented using C#.NET. Particle Swarm Optimization is performed using the property files created in the previous phase. These files provide the user requirements on the basis of the location. The search space is constructed by package details from multi-clouds. Package details corresponding to 20 distinct requirements were used for building the search space. PSO is executed by adding the requirements for each cluster to the search space, and distributing all the particles on the requirement node. Particle movement is triggered and the gbest obtained after the termination is considered to be the best package for the current requirement.

C. Results and Discussion

The time taken for each of the phases individually and the aggregated time are presented in Fig. 3 and 4. Scalability of the proposed approaches is identified by varying the size of the access log from 0.5 million records to 4.4 million records.

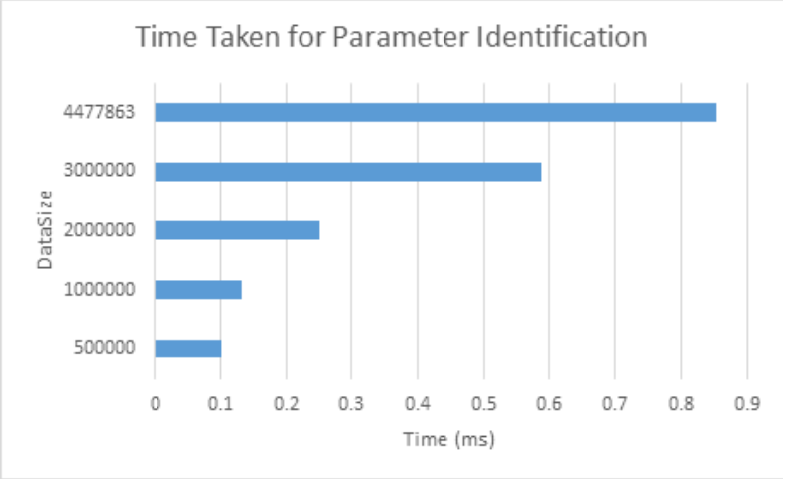

Fig. 3. Time Taken for Parameter Identification

Time taken for parameter identification using python is presented in Fig. 3. A linear increase in time in accordance with the increase in data size was observed in the figure. However, the scale of data is presented increasing at the rate of 0.1ms. The time required for processing 0.5 million records have been observed to be 0.1ms, and the time required for processing 4.4 million records is 0.85ms. The maximum time requirement was observed to be less than 1ms for 4.4 million records. Hence even with the linear time complexity, the actual CPU time was observed to be very low, depicting the optimality of the proposed approach.

Fig. 4. Time Taken for Package Selection

Time taken for the package selection process in multi-cloud using PSO is presented in Fig. 4. It could be observed that the proposed technique operates effectively irrespective of the size of the data being used for the selection process. This property of the PSO based package selection technique is attributed to its metaheuristics nature and optimal convergence of solutions. The package selection time varies between 12ms and 14ms. The variations are caused by random selection of solutions. Sometimes, the system gets trapped into local optima, leading to inefficient convergence and this in-turn leads to longer convergence times. However, even in such a case, convergence was observed to happen in 14ms. This proves the high scalability levels exhibited by the PSO based package selection process in multi-cloud. In a multi-cloud environment, the probability of variations in package levels will be a regular phenomenon. Hence a scalable approach that can handle data of any type is mandatory.

A time comparison was carried out with a similar resource provisioning technique proposed by Madhumathi et al in [33]. This technique uses a modified Ant Colony Optimization (ACO) as its provisioning model. The time requirements are shown in Table 1. It was observed that the proposed PSO based selection technique performs effectively in the package selection phase exhibiting a 5X to 6X level of improvement in terms of time. A comparison chart is presented in figure 5.

TABLEIII. Time Comparison

Data Size	Proposed Technique (ms)	Modified ACO based Selection [33] (ms)
500000	14	71
1000000	12	73
2000000	14	70
3000000	14	70
4477863	12	89

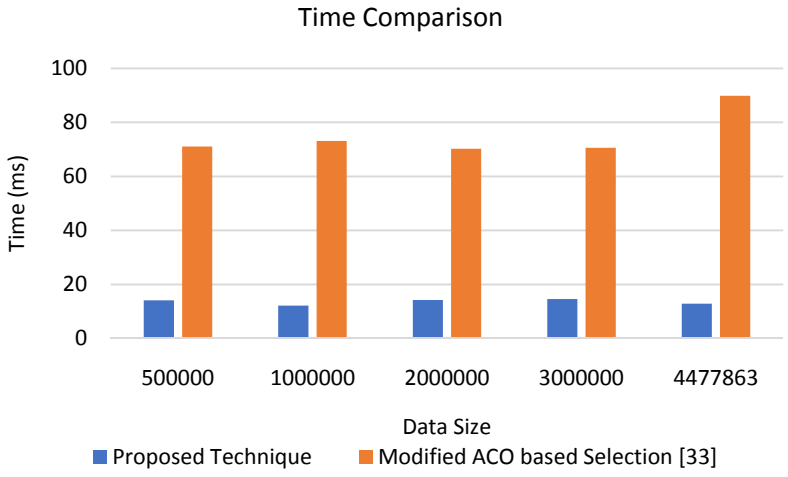

Fig. 5. Time Comparison

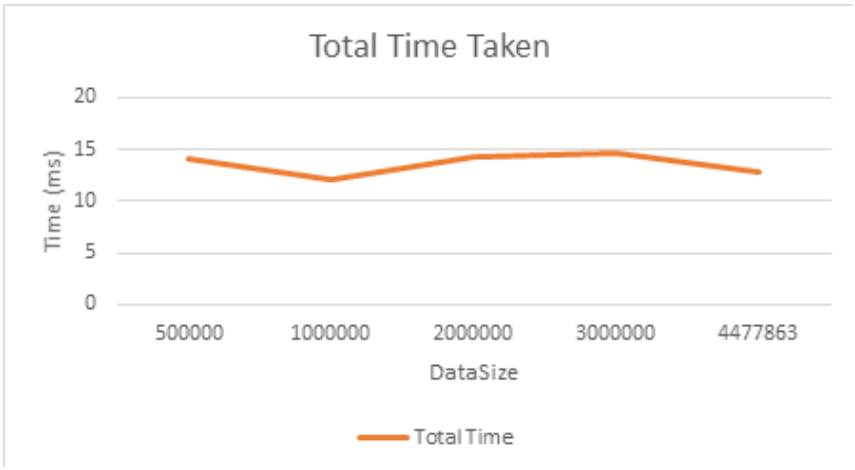

Fig 6. Total Time Taken

The aggregated time for grouping and identifying the best packages in a multi-cloud environment is presented in Fig. 6. The measurements were obtained using datasets of size ranging from .5 million records to 4.4 million records. It could be observed that the line representing time is almost constant, requiring 12ms to 14ms approximately. This proves the high scalability level of the proposed technique.

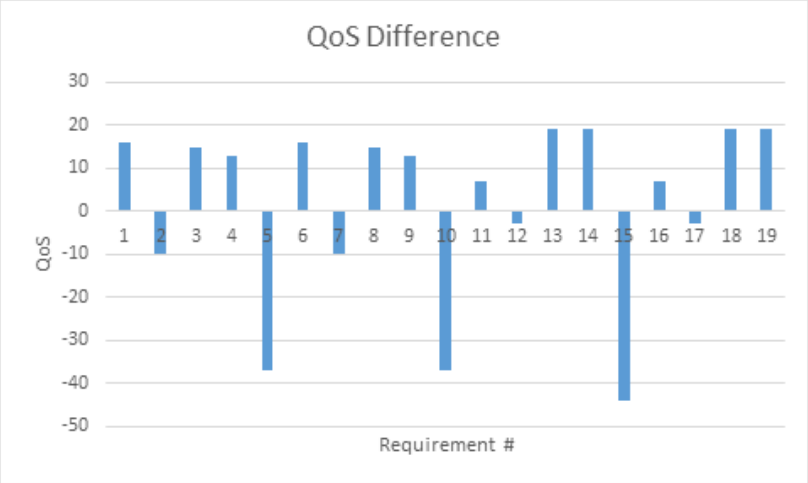

Figure 7. QoS Difference.

The difference between the required QoS and the provided QoS is presented in Fig.7. It could be observed that none of the transactions exhibited perfect match with zero QoS difference. This is due to the predefined package structures provided by the cloud service providers. This leads to a user's requirements that never match the defined packages. Hence a near optimal algorithm that identifies the closest possible solution is sufficient for the problem. The difference in quality is calculated using eq. 8

$$QoS\ Difference = Provided\ Quality - Required\ Quality\ldots. \quad (8)$$

It could be observed that the differences in the quality parameters are quite low and mostly towards the positive direction, meaning that the provided quality matches the requirement as closely as possible and in most of the instances, the allocated resource is more than the requirement. It could be concluded that the requirement of a user is satisfied to the maximum extent approximately for 85% of the requirements.

IV. Conclusion

Cloud service selection is one of the major components of effectively utilizing a cloud architecture. However, this is the only component that has not been automated. The reason for not automating this component is that user's parameters play a huge and major role in this process. Hence the cloud providers have left the process of decision making to the users themselves. However, not all users are technically sound enough to identify and provide the accurate inputs during the package selection process. Further, the packages themselves are predefined, making the selection process further complicated. This paper presents an automated package selection system in multi-clouds by initially identifying the user's requirements from usage logs and then identifying the package that matches the user's requirements to the maximum extent. User's web log is obtained as input, followed by geographic location based clustering and cluster based requirement identification. PSO is used for identifying the best package due to its metaheuristic nature and the requirement of a near optimal solution, rather than the best solution. Experiments revealed that the proposed approach exhibits acceptable time limits, with very low time requirements (<1 sec) for both the grouping and the package selection process. The quality difference between the requirements and the assigned services was also observed to be low. This makes the proposed approach best suitable for automated service selection. Future directions include incorporating prediction and game theoretic approaches to identify requirements of the user to provide effective packages in-order to avoid customer churn or package shifting scenarios. Future directions from the current contributions also include incorporating the optimization module into the architecture such that the system recommends packages according to the resource usage levels observed in the system.

References

[1] F. Bahrpeyma, H.Haghighi, andA. Zakerolhosseini, "An adaptive RL based approach for dynamic resource provisioning in Cloud virtualized data centers",Computing, 97(12), 1209-1234, 2015.
[2] P. Gupta, A.Seetharaman, J.R. Raj, "The usage and adoption of cloud computing by small and medium businesses", Int J InfManag, 33, 861–874, 2013.
[3] M. Maurer, I. Brandic, andR.Sakellariou, "Adaptive resource configuration for Cloud infrastructure management",Future Gener Comput Syst. 29, 472–487, 2013.
[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A.Konwinski, G. Lee, D. Patterson, A.Rabkin, I.Stoica, and M. Zaharia, "A view of cloud computing", Commun. ACM , 53(4), 50–58, 2010.
[5] I. Foster, Y. Zhao, I. Raicu, S. Lu,"Cloud computing and grid computing 360-degree compared",In: Grid computing environments workshop (GCE'08). 1–10, 2008.
[6] X. Wang, B. Zhou, andW. Li, "Model-based load testing of web applications",J. Chin. Inst. Eng. 36 (1), 74–86, 2013.
[7] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A.Konwinski, G. Lee, D.A. Patterson, A.Rabkin, I.Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing", 2009.
[8] H. Hayes, "Cloud computing", Commun ACM. 51, 9–11, 2008.

[9] M. Randles, D. Lamb, andA. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing",In: The 24th IEEE international conference on advanced information networking and applications workshops. 551–556, 2010.
[10] N.Ferry, A. Rossini, F.Chauvel, B. Morin, "A Solberg Towards model-driven pro- visioning, deployment, monitoring, and adaptation of multi-cloud systems,In: Pro- ceedings of the IEEE 6th International Conference on Cloud Computing. 887–894, 2013.
[11] Y. Sun, J. White, S.Eade, andD.C. Schmidt, "ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization",Journal of Systems and Software, 116, 146-161, 2016.
[12] D. Draheim, J. Grundy, J. Hosking, C.Lutteroth, andG. Weber, "Realistic load test- ing of web applications", In: Proceedings of the IEEE 10th European Conference on Software Maintenance and Reengineering, CSMR. IEEE, 11, 2006
[13] X. Wang, B. Zhou, andW. Li, "Model-based load testing of web applications",J. Chin. Inst. Eng. 36 (1), 74–86, 2013.
[14] S. Marrone, andR. Nardone, "Automatic resource allocation for high availability cloud services",Procedia Computer Science, 52, 980-987, 2015.
[15] A. Beloglazov, J. AbawajyandR. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing",Future Generation Computer Systems. 28(5):755–768, 2012.
[16] N. Quang-Hung, N.Thoai, andN. Son, "EPOBF: Energy efficient allocation of virtual machines in high performance computing cloud", In: Transactions on Large-Scale Data and Knowledge-Centered SystemsXVI; LNCS. ISBN 978-3-662-45946-1. 71–86, 2014.
[17] B. Dougherty, J. White, andD. Schmidt, "Model-driven auto-scaling of green cloud computing infrastructure",Future Generation Computer Systems. 28(2), 371–378, 2012.
[18] P. Gupta, andS.P. Ghrera, "Power and Fault Aware Reliable Resource Allocation for Cloud Infrastructure",Procedia Computer Science. 78, 457-463, 2016.
[19] F. Bahrpeyma, H.Haghighi, andA. Zakerolhosseini, "An adaptive RL based approach for dynamic resource provisioning in Cloud virtualized data centers",Computing, 97(12), 1209-1234, 2015.
[20] S. Islam, J. Keung, K. Lee, and A. Liu,"Empirical prediction models for adaptive resource provisioning in the cloud",FutureGenerComput Syst. 28, 155–162, 2012.
[21] Q. Zhang, Q. Zhu, andR. Boutaba, "Dynamic resource allocation for spot markets in cloud computing environments", In: Fourth IEEE international conference on utility and cloud computing (UCC). 178–185, 2011.
[22] Y. Huang, Y. Lan, S. J. Thomson, A. Fang, W. C. Hoffmann andR.E. Lacey, "Development of soft computing and applications in agricultural and biological engineering",Comput Electron Agric. 71, 107–127, 2010.
[23] M. Al-Ayyoub, Y. Jararweh, M.Daraghmeh, andQ.Althebyan, "Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure",Cluster Computing. 18(2), 919-932, 2015.
[24] J. Shi, J. Luo, F. Dong, J. Zhang, andJ. Zhang, "Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints",Cluster Computing. 19(1), 167-182, 2016.
[25] C. Li, "Optimal resource provisioning for cloud computing environment",The Journal of Supercomputing. 62(2), 989-1022, 2012.
[26] T.L. Saaty,K. Peniwati, "Group decision making: drawing out and reconciling differences", RWS publications, 2013.
[27] T. L. Saaty, "Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process", RACSAM-Revista de la Real Academia de Ciencias Exactas. Fisicas y Naturales. Serie A. Matematicas. 102(2), 251-318, 2008.
[28] S. Patro, andK. K. Sahu, "Normalization: A Preprocessing Stage",arXiv preprint arXiv:1503.06462, 2015.
[29] P. C. Fishburn, "Letter to the editor—additive utilities with incomplete product sets: application to priorities and assignments",Operations Research, 15(3), 537-542, 1967.
[30] E. Triantaphyllou,"Multi-Criteria Decision Making: A Comparative Study", Dordrecht, The Netherlands: Kluwer Academic Publishers (now Springer). 320. ISBN 0-7923-6607-7, 2000.
[31] J. Kenndy, and R.C. Eberhart, "Particle swarm optimization",InProceedings of IEEE International Conference on Neural Networks, 4, 1942-1948, 1995.
[32] Y. Shi, andR, Eberhart, "A modified particle swarm optimizer",InEvolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on pp. 69-73, 1998.
[33] C. Madhumathi, andG. Ganapathy, "Requirement Intensity Based Resource Provisioning For E-Learning In Multi-Cloud To Avoid Vendor Lock-Ins",ARPN Journal of Engineering and Applied Sciences, Vol.11, No17, 2016.