Hybrid C-PSO for QOS based Cost Effective Model for Package Selection in Cloud

K. S. Guruprakash, S. Siva Sathya

Abstract*— Cloud computing has emerged as the most dominant service computing model due to its elastic nature in provisioning resources. Package selection and resource provisioning are the major components of any cloud system. This paper proposes a cost-effective package selection model for users of a cloud system. The major downside of any cloud system is its inability to automatically identify the appropriate QoS requirements for users. This can be effectively identified with the usage logs of users. The proposed package selection model analyses usage logs and groups the usage records into regular, upscale and downscale levels to identify their corresponding QoS requirements. The proposed hybridized C-PSO based package selection model is used to select the appropriate packages that are cost effective and aids in effective resource utilization with minimal upscale requirements. A comparison with the existing package selection model exhibits improved QoS levels and reduced time requirements.

Keywords: Cloud provisioning, package selection, C-PSO, Simulated Annealing

1. INTRODUCTION

Cloud computing provides effective solutions for business and scientific computing due to its on-demand nature of resource requests. The elastic nature of operations provides flexibility to users, enabling any type of computation with the limited resources at their disposal. The major advantage of using a cloud computing platform rather than opting for a dedicated environment is that they are cost effective and highly scalable. Utilizing a cloud computing platform enables the user to reduce their cost on dedicated resources, hence providing high cost benefits. Further, the added scalability factor provides better and enhanced resources at the disposal of the user when requirements arise. Automatic upscaling support available in all cloud platforms enables the users to access even enhanced resources when increased resource requirements arise. The resources are automatically downscaled when the requirement expectations is lowered. Although cloud platforms exhibit huge benefits, they are also bound by certain major issues. Foremost of the issues is the uncertainty in usage levels of the resources. Most customers are not usually aware of their requirements, due to their incompetence in QoS parameter levels. Hence the packages selected by them might not entirely correspond to their requirements (Tchernykh & Schwiegelsohn, 2015). Hence this leads to either over- utilization or under-utilization of resources. Over-utilization is compensated by automatic upscaling of resources, however under-utilized resources remain with the user and the cloud provider cannot control this behavior. Cloud providers provide the facility to automatically upscale, however, only upscaled resources have the possibility of getting downscaled. Hence if the user has selected a higher performance package, even if they do not use it entirely, they are bound to pay for the resources. This is not only an issue for a customer, it is also an issue for the provider. Such issues can be solved only if the customer is made aware of their requirements and the quality levels of the packages provided by the cloud provider. However, as of now there is no automated mechanism for identifying these requirements in the cloud interface (Amiri & Mhoammad, 2017). Although customers are unaware of their requirements, their usage logs can effectively reflect the required quality parameters to a maximum extent. Analyzing the usage logs can enable effective mapping of user requirements with the cloud packages. Another major issue concerning effective package selection process is the absence of resource fine-tuning capabilities. Cloud packages are predefined with resources and customers have the capabilities to select appropriate packages, but not fine tune them according to requirements. Hence perfectly matching QoS parameters with the user requirements is not possible. The near optimal package that best meets the user's requirements is considered for usage by the customer. However, identifying the near-optimality is a NP-Hard problem, hence requiring meta-heuristic based methods for effective selection.

This paper presents an effective model that can be utilized by customers to identify the packages that best meets their requirements. The proposed architecture performs an initial log analysis to identify the regular, upscale and downscale requirements of the user. These requirements are aggregated and averaged to provide a QoS that effectively reflects the regular, upscale and downscale requirements of the user. Optimal packages corresponding to these QoS requirements are identified and the cost effective packages are selected for the user. The major benefit of this model is that any user with logs can benefit from this system, irrespective of whether they are a new user or an already existing user.

2. RELATED WORK

This section discusses the major and recent contributions in the domain of package selection in cloud.

Revised Version Manuscript Received on 30 May, 2018.

K. S. Guruprakash, Research Scholar, Pondicherry University, Pondicherry, India.

S. Siva Sathya, Associate Professor, Pondicherry University, Pondicherry, India.

A large-scale high throughput based resource provisioning model for scientific computing requirements was presented by Kim et al. in (Kim & Kim , 2014). This model identifies the job traces to fine-tune the performance of applications in cloud. Further, this model also enables an adaptive evaluation step, that aids in providing flexibility to the model to adapt to dynamic environments. An automated resource provisioning model for managing fluctuations in the workload was proposed by Arani et al. in (Ghobaei-Arani & Jabbehdari, 2016) . This work has its major concentrations on reducing the undesirable states such as over-provisioning and under-provisioning. It operates based on the monitor-analyse-plan-execute (MAPE) model. The MAPE model operates in a loop to effectively and continually monitor and provision resources in the cloud environments. An extension of this model was proposed in (Ghobaei-Arani & Jabbehdari, 2017).A profit driven provisioning model in cloud environments was proposed by Daniel et al. in (Daniel & Raviraj, 2017). This model concentrates on multimedia based cloud service providers in aiding uninterrupted content flow for the customers. User profile creation and popularity based caching enables reduced cost by reducing unnecessary waiting and downloading times. Further, it also aids in effective duplicate detection hence eliminates the issue of duplicate downloads. A cost-effective provisioning model for hybrid cloud applications was presented by Liu et al. in (Liu & Luo, 2017). Load balancing is one of the major issues encountered in hybrid cloud applications. This wok proposes the Least Cost per Connection (LCC) algorithm to identify the cost-effective cloud base for reduced outsourcing. A cost-effective cloud resource provisioning model was proposed by Javadi et al. in (Javadi & Thulasiraman, 2013). This model considers all the three steps in resource provisioning, namely; resource brokering, dispatch sequences and scheduling. The proposed cost-aware and failure-aware provisioning policies exhibit effective fault-tolerance in the cloud system. A QoS based resource provisioning model QRPS was presented by Singh et al. in (Singh & Chana, 2016). It proposes a resource provisioning and scheduling framework that can be used for effective resource distribution. Cloud workloads are identified by clustering and resource scheduling is performed according to the clustered QoS requirements. Scheduling of resources is performed based on minimum energy consumption, execution cost and execution time. An automated cloud provisioning model proposed for business applications was proposed by Benfenatki et al. in (Villari & Celesti, 2017) .This method proposes a component-oriented method for cloud provisioning and reduces the requirements for technical knowledge for the user to enable a flexible interface. An energy aware provisioning model for optical cloud networks was proposed by Yang et al. in (Yang & Wieder, 2017).

3. QOS BASED PACKAGE SELECTION MODEL FOR COST-EFFECTIVE RESOURCE SCALING

Appropriate selection of packages plays a vital role in providing effective usage of resources for cloud providers and cost effectiveness for the customer. Identifying resources that accurately fits the users' requirements is a challenging task, partly due to the low knowledge levels of the consumer and the absence of automated mechanisms and partly due to the fluctuating requirements of the users. The usage requirements of a customer does not remain stable, even when examined within the timeline of a single day. Usage levels are high at certain periods and drops down at certain other periods. Cloud providers handle this issue by automatically upscaling and downscaling resources. However, such automatic scaling leads to high costs compared to dedicated requirements. Hence such automatic scaling of resources are always recommended to be avoided. This paper presents a mechanism that identifies the customer's usage levels based on customer's logs and uses a hybridized Catfish PSO model for identifying the optimal packages. The proposed model has been designed for identifying cost-effective QoS based package identification for usage of individual customers with usage data. The proposed architecture for cost effective package identification model is shown in figure 1.

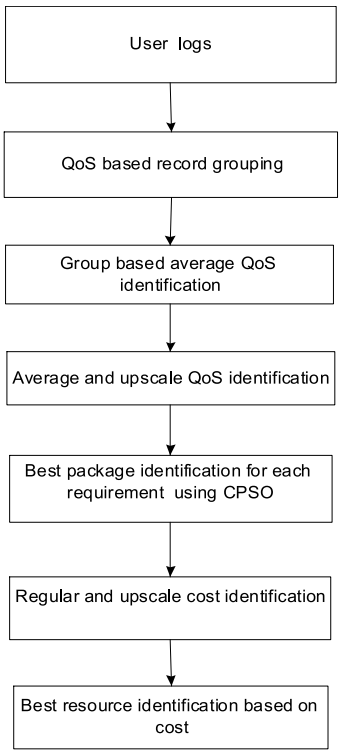


Fig.1. Cost Effective Resource Provisioning Model

Log based QoS Identification

Usage logs of a single customer is taken as the input for this module. QoS requirements are obtained from these logs. The QoS parameters considered for evaluation are shown in table 1.

Table 1 QoS Parameters Considered for Evaluation

QoS Parameters Considered for Evaluation	Description
Bandwidth (Bw)	Maximum data that can be transmitted in the network. $BW = \sum_{resource^i} \left(\frac{Size}{Capicity}\right)$
Computation Capability (CC)	Upper processing limit of the resources. $CC = \frac{max_{task}(Exe_{time}) - min_{task}(}{Avg_{task}(Exe_{time})}$
Availability (Av)	Usable resources available during a resource request. $Av = \sum_{resource^i} \frac{MTBM}{MTBM + MTTR}$ MTBM represents the Mean Time Between Maintenance MTTR represents the Mean Time to Repair a resource
Reliability	Ability of the cloud system to complete the task within the stipulated time. $R = \frac{\sum_{task} Exe_{time}}{Total_{time}}$
Throughput	Amount of tasks that can be completed within the defined time period. $T = \sum_{task} Exe_{time}$

The usage log corresponds to a single user and it represents their general requirements and their upscale and downscale requirements. General resource usage levels correspond to regular and most frequent requirements. These QoS requirements occur during the majority of the usage cycles. However, at certain points in time, the requirements escalate, exceeding the regular usage levels. These correspond to upscale requirements. Requirement fluctuations and their variations over time are recorded.

Fuzzified QoS based Grouping

The QoS values identified by the previous module are numerical. In order to facilitate effective grouping, the QoS parameters are fuzzified into three categories namely; low, medium and high. Boundaries for the categories are not predefined. Top and bottom 30% of the data are considered to be low and high respectively, and the intermediate 40% is categorized as medium. This flexible grouping allows effective grouping of any type of data irrespective of the value ranges contained in it. The fuzzified values are smoothened by considering values of 1, 3 and 5 for low, medium and high respectively.

QoS based grouping is performed using K-Means Clustering using 3 as the value for k. Since the proposed architecture requires regular, upscale and downscale data, grouping into three clusters would be most appropriate. Euclidean distance is used as the fitness measuring criteria. Fitness corresponding to a record n and a cluster head Qc is given by

$$fitness_n = \frac{1}{\sqrt{\sum_{p=1}^{n}(Qn_p - Qc_p)^2}} \quad (1)$$

Where p is the set of all (n) QoS parameters, Qnp refers to the QoS value of the parameter p for a usage record Qn and Qcp refers to the QoS value of the parameter p for a cluster centre Qc. Records are grouped under the cluster heads that exhibit maximum fitness compared to other cluster heads. The algorithm for grouping usage records is given below

Group based QoS Level Identification

Usage records with similar QoS requirements are grouped together in the previous phase. However, the grouping is performed on fuzzified data, hence the actual requirement values will vary considerably. This phase identifies the actual QoS values for each of the groups in terms of numerical components, hence providing the QoS requirements on real-time basis. QoS requirement for a parameter p in a group g is given by

$$QoS_{pg} = \frac{\sum_{i=1}^{x} Qn_i}{x} \quad \forall\ 1 \le p \le n \quad (2)$$

Where Qni refers to the QoS requirement for the ith node in a cluster g with x nodes and the value of p ranges from 1 to n. This module identifies the average regular, upscale and downscale requirements for package selection.

QoS based Package Selection using Hybridized Catfish PSO

QoS identification for regular and scaled requirements is followed by identification of packages that optimally satisfy the requirements. The major issue in the process of package selection arises from the fact that packages are predefined by the cloud providers. Users have the flexibility to only choose the packages that best caters to their needs. Users are not allowed to perform package fine-tuning. Accurate matching based selection algorithms tend to fail in this aspect. This mandates the use of algorithms operating to provide near optimal solutions. This paper uses a modified form of Catfish Particle Swarm Optimization (C-PSO) algorithm for effective package selection.

Catfish PSO is a modified form of PSO algorithm proposed by Chuang et al. in (Chuang & Tsai, 2011). C-PSO has been extended and used in several cases such as C-PSO feature selection improved C-PSO for feature selection (Chuang & Tsai, 2011a), (Chuang & Tsai, 2009), (Jarre, 03) chaotic C-PSO for solving numerical optimization problems, Fuzzy

adaptive (Chuang & Tsai, 2011b) and C-PSO embedded with chaotic map (Chuang & Tsai, 2012) for improving the search results. A major downside of PSO is that it has high probability of getting struck in local optima, leading to low quality solutions. Catfish effect is applied to the regular PSO, to reduce the probability of occurrence of local optima. Catfish operates on the principle that if the gbest of the swarm does not change for a considerable number of iterations, then the swarm is observed to have achieved stagnation. However, the stagnation might even be on a local optimal solution. Hence during the first encounter of stagnation, catfish particles are introduced into the swarm at random extreme positions. Particles are then migrated to these positions and the process of migration towards the best solution proceeds. If the previously stagnated solution was a local optimal solution, the catfish dispersion would have moved the particles out of it to obtain the global optimal solution. However, introducing the catfish particles leads to an increased computational complexity, as the swarm has to perform the entire convergence process twice, irrespective of the solution obtained in the initial iteration. Even if the swarm has converged on the global optimal solution, it is made to converge again on the same solution after the dispersion of catfish particles. In order to overcome this issue, this paper proposes a hybridized C-PSO algorithm that incorporates Simulated Annealing (SA) into the local search mechanism to reduce the computational complexity of the search process to a considerable extent.

Proposed Hybridized Catfish PSO for Package Selection

Particle representation and distribution

Each solution is represented by the particles in search space. The search space and boundaries are defined by the packages provided by the cloud provider. Each QoS criterion defined by the package is considered as a dimension in the search space. After search space is created, particles are distributed at random location in search space. The regular, upscale and downscale requirements identified from the previous phase are used as initial points for particle distribution.

The initial velocity of the particles is selected in random, and is given by

$$V_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|) \quad (3)$$

where bup and blo are the upper and lower boundaries of the search space defined by the QoS levels of the package.

Particles movement

Velocity assignment is made by the actual movement of particles. PSO operates in a continuous search space, however, the proposed package selection model mandates the use of a discrete search space. Hence the particle movements are discretized after every iteration. The process of discretization is given by

$$P' = \min\left(\sum_{j=1}^{n}\left(\sum_{k=1}^{d}\sqrt{(P_{ik} - N_{jk})^2}\right)\forall i = 1 \text{ to } p\right) \quad (4)$$

where Pik refers to the particle i's current location corresponding to dimension k, Njk refers to the kth dimension of node Nj.

Fitness Identification for optimal package selection

Every particle movement is followed by the identification of pbest (particle best)and gbest (global best). The best so-far solution achieved by each particle is recorded in pbest and the best solution obtained from the swarm is recorded in gbest. The proposed hybridized C-PSO model uses Simulated Annealing (SA) to perform the identification of gbest from the available pbest values.

Further particle movements are defined by the velocity component, which is given by

$$V_{i,d} \leftarrow \omega V_{i,d} + \varphi_p r_p (P_{i,d} - X_{i,d}) + \varphi_g r_g (g_d - X_{i,d}) \quad (5)$$

where rp and rg are the random numbers, Pi,d and gd are the parameter best and the global best values, xi,d is the value of the current particle position, and the parameters ω, φp, and φg are selected by the practitioner.

The process of movement and particle discretization is performed until the swarm reaches a stagnation state. Stagnation state is defined by the stability of the gbest (global best) value. If the stagnation persists for a considerable threshold level (StagThresh), the catfish particles are activated and dispersed in the solution, followed by another iteration of best solution identification.

The Algorithm for the proposed Hybrid C-PSO is given below.

Proposed Package Selection Algorithm (Hybrid C-PSO):

1. Iniitialize the Search space boundary using package data
2. Define StagThresh and set flagstag=True
3. For each particle i=1…p
a. Initialize particles to a random position
b. Initialize pbest and gbest
c. Initialize velocity using the search space boundaries
4. Until the termination criterion is met perform the following
a. For each particle i=1…p
i. Generate rp and rg using normal distribution
ii. Identify the particle velocity using eq. (4)
iii. Convert the continuous movement of the particles into discrete using eq. (3)
iv. Update the particle's position to P'
v. If pbest< current fitness
1. Assign current fitness to be the pbest
b. best<- Simulated Annealing (gbest,pbest,p)
c. If previous gbest = current gbest then increment stagnation
d. If stagnation = StagThresh and flagstag=True
i. Disperse p catfish particles in the search space
ii Update position of particle i to the position of the ith catfish particle
iii Set flagstag=False
iv Goto step 3
5. gbest contains the best found solution
Simulated

Annealing(gbest,pbest,p)

1. Let s = gbest

2. For k = 1 through p :

a. T ← pbestk

b. Pick a random pbest (pb), snew ← pb

c. If P(E(s), E(snew), T) ≥ random(0, 1), move to the new state:

s ← snew

3. Output: the final state s

In the above algorithm Simulated Annealing (gbest,pbest,p),step 2(c) was defined as

P(e,e',T) ={ 1 if e' < e
exp(-(e'-e)/T) otherwise. }

The best package for regular, upscale and downscale requirements of the user are independently identified in this phase.

Cost based Best Package Selection

Although the best packages for each requirement has been identified, utilizing the package exhibiting average requirement scenario might not always be cost effective. Cost effectiveness is identified by incorporating the usage level along with the usage time. Usage level is defined as the upscale, average or downscale requirement, and usage time is defined as the number of hours the particular usage level has been accessed. Cost for each of the requirements is given by

$$Cost_A = (Cost_{PackageA} * (time_A + time_D) + (UpscaledCost_{PackageU} * time_U)$$
$$Cost_U = Cost_{PackageU} * (time_U + time_A + time_D) \quad (6)$$

Where CostA is the cost requirement if the package depicting average requirements CostPackageA is selected and CostU is the cost requirement if the package depicting upscale requirements CostPackageU is selected, timeD, timeA and timeU represents the time requirements for downscaled, average and upscaled requirements. The difference in cost requirements is given by

$$Cost_{Diff} = Cost_U - Cost_A \quad (7)$$

The final package is selected based on the cost difference and the user cost threshold (CThresh). The user cost threshold defines the maximum tolerated increase in cost level for selecting a package.

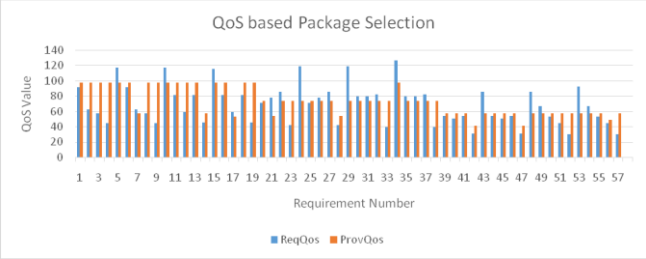
$$Final\ Package = \begin{cases} Package_U & if\ Cost_{Diff} \leq C_{Thresh} \\ Package_A & Otherwise \end{cases} \quad (8)$$

The final package selected using equation 8 is observed to effectively satisfy the QoS requirements of the user and is also found to be cost effective package for incorporating the upscale and the downscale requirements.

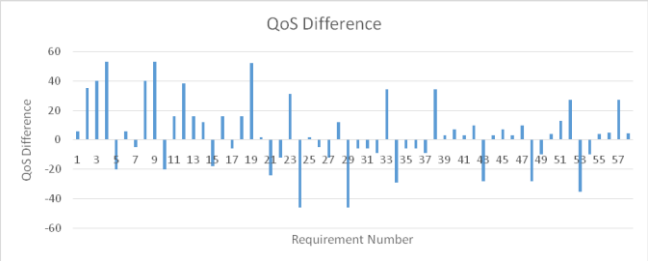
4. RESULTS AND DISCUSSION

The proposed optimal package selection model is implemented using C#.NET. Search space for the proposed hybridized C-PSO is constructed using the package details corresponding to 20 distinct requirements. Hybridized C-PSO is executed by adding the user's requirements 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. This process is performed for average, upscale and the downscale requirements.

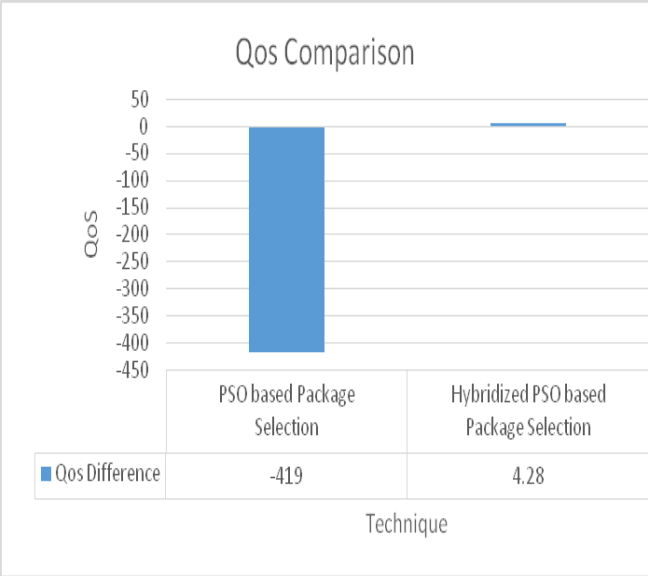
The proposed model is operated on 60 distinct user requirements and the obtained package suggestions are recorded and compared in figure 2. It could be observed from the figure that in most of the requirements, the provided QoS either accurately meets the required QoS or provides a package with slightly higher QoS when compared with the requirement. Very minimal number of transactions were observed to exhibit lowered QoS compared to the requirements. This lowered requirement is attributed to the cost requirement and the absence of packages that closely match the requirements.



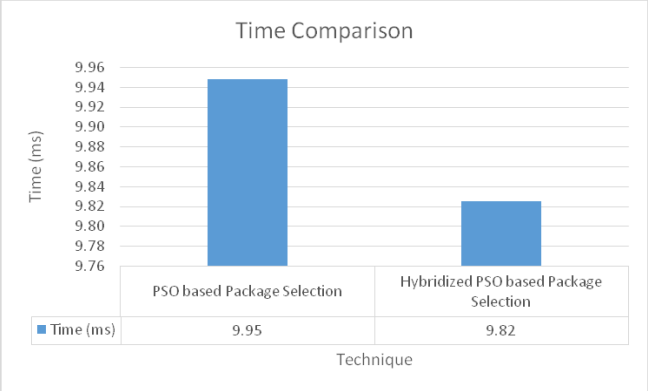
Difference between the provided QoS and the required QoS are shown in figure. Positive values depict that the provided QoS is higher than the required QoS, while negative values depict that the provided QoS is lower than the required QoS. It could be observed that in most of the transactions, the requirements are satisfied effectively, while certain cases exhibit lowered QoS levels.



A comparison of the proposed model with PSO based package selection model in (Guruprakash & SivaSathya, 2017) is performed and is shown in figures. A QoS based comparison exhibiting the average QoS of the compared models for 60 distinct user requirements is shown in figure. It could be observed that the average QoS provided by the proposed hybridized C-PSO model exhibits a low positive fluctuation of 4.28, while the package selection model proposed in (Guruprakash & SivaSathya, 2017) exhibits a high negative fluctuation of -419. Positive fluctuations are acceptable, while negative fluctuations creates a large number of upscale requirements leading to high costs.



A time comparison between the proposed hybridized C-PSO based package selection model and the model proposed in (Guruprakash & SivaSathya, 2017) is shown in figure. It could be observed from the figure that the proposed hybridized C-PSO exhibits lower time requirements compared to the package selection model proposed in (Guruprakash & SivaSathya, 2017). The C-PSO model is expected to exhibit higher time requirements, due to the multiple iteration requirements. However, it could be observed that the proposed C-PSO model hybridized with Simulated Annealing has effectively reduced the time requirements to a considerable extent, hence reducing the time complexity of the proposed model.



5. CONCLUSION

Cloud resource provisioning models usually operate using the default selected packages. Automatic upscaling is performed only when requirements arise for resources with increased computing needs. The major issue to be observed in this process is that upscaling is time bound. Hence even if the requirements reduce, the upscaled resource remains in use until the defined time. Downscaling to regular usage levels is done only after this stipulated time. Resource upscaling also tends to be costly compared to dedicated usage packages. Hence from the user's perspective, it is also required to provide packages that not only satisfy the QoS requirements, but are also cost effective. The proposed cloud provisioning model groups user logs to identify the regular, upscale and downscale requirements and provides a cost-effective solution for the users. The package selection process is performed by the proposed hybridized C-PSO algorithms. The major advantage of the proposed model is that it can be applied for both old and new customers. Old customers can utilize the model to select cost effective resources catering to their needs, while new users can provide their usage logs (if available) to avail automatic package selections. Another major advantage of this approach is that applying this model leads to lowered upscale requirements, hence reducing the usage level uncertainty to the maximum extent. Future extensions of the proposed model can be to extend the proposed model to incorporate it in hybrid multi-cloud environments to enable multi-level provisioning.

REFERENCES

1. Amiri, M and Mhoammad-Khanli, L. (2017). Survey on prediction models of applications for resources provisioning in cloud. Journal of Network and Computer Applications.
2. Chuang, LY, Tsai, S and Yang, CH. (2011b) Chaotic catfish particle swarm optimization for solving global numerical optimization problems. Applied Mathematics and Computation, 217(16), 6900-6916.
3. Chuang, LY, Tsai, SW and Yang, CH. (2009) Improved catfish particle swarm optimization with embedded chaotic map. In Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on, 3895-3900.
4. Chuang, LY, Tsai, SW and Yang, CH. (2011). Catfish binary particle swarm optimization for feature selection. In International Conference on Machine Learning and Computing IPCSIT, 3, 40-44.
5. Chuang, LY, Tsai, SW and Yang, CH. (2011a). Improved binary particle swarm optimization using catfish effect for feature selection. Expert Systems with Applications 38(10), 12699-12707.
6. Chuang, LY, Tsai, SW and Yang, CH. (2012) Fuzzy adaptive catfish particle swarm optimization. Artificial Intelligence Research, 1 (2), 149.
7. Daniel, D and Raviraj, P. (2017) Distributed hybrid cloud for profit driven content provisioning using user requirements and content popularity. Cluster Computing , 20(1), 525-538.
8. Ghobaei-Arani, M, Jabbehdari, S and Pourmina, MA. (2017).An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. Future Generation Computer Systems.
9. Ghobaei-Arani, M, Jabbehdari, S and Pourmina, MA. (2016). An autonomic approach for resource provisioning of cloud services. Cluster Computing, 19 (3), 1017-1036.
10. Guruprakash, KS and SivaSathya, S. (2017) Location Based Optimal Package Selection in Multi-Cloud.
11. Jarre, A. Box 8.4 continued. Marine Ecosystems and Global Change, 243.
12. Javadi, B, Thulasiraman, P and Buyya, R.(2013). Enhancing performance of failure-prone clusters by adaptive provisioning of cloud resources. The Journal of Supercomputing, 63 (2), 467-489.

13. Kim, S, Kim, JS, Hwang, S and Kim, Y. (2014).Towards effective science cloud provisioning for a large-scale high-throughput computing. Cluster computing, 17 (4) , 1157-1169.
14. Liu, F, Luo, B and Niu, Y. (2017). Cost-Effective Service Provisioning for Hybrid Cloud Applications. Mobile Networks and Applications, 22 (2), 153-160.
15. Singh, S and Chana, I. (2016). Resource provisioning and scheduling in clouds: QoS perspective. The Journal of Supercomputing, 72 (3), 926-960.
16. Tchernykh, A, Andrei, Uwe Schwiegelsohn, U, Vassil Alexandrov, V and El-ghazali Talbi, EG. (2015). Towards understanding uncertainty in cloud computing resource provisioning. Procedia Computer Science, 51, 1772-1781.
17. Villari, M, Celesti, A, Tricomi ,G, Galletta ,A and Fazio, M.(2017). Deployment orchestration of microservices with geographical constraints for Edge computing. In Computers and Communications (ISCC), 2017 IEEE Symposium on, 633-638.
18. Yang, S, Wieder, P, Yahyapour, R and Fu, X. (2017). Energy-Aware Provisioning in Optical Cloud Networks. Computer Networks, 118 , 78-95.