Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 25 (2016) 341 – 348

Procedia Technology

Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016)

# An Efficient Resource Management For Prioritized Users In Cloud Environment Using Cuckoo Search Algorithm

**Aneena Ann Alexander.[a,*],Divya Lissia Joseph.[b,*]**

[a]Department of Computer Science & Engineering, St.Joseph's College of Engineering & Technology,Palai,Choondacherry,Kerala,686579

[b]Department of Computer Science & Engineering, St.Joseph's College of Engineering & Technology,Palai,Choondacherry,Kerala,686579

## Abstract

**Cloud computing is a model which is empowered for on demand and convenient system access to a shared pool of computing resources. The clients of the cloud can be an individual or an association, and can acquire their respective services from cloud service provider. Advances in technologies, lead to the relocation from antiquated desktop devices to smart mobile devices. With a large increase in the number of mobile devices and bandwidth clients can execute as many tasks from their gadgets itself. Be that as it may, with versatility come its innate issues, for example, resource scarceness, finite energy and low network connectivity. This is, indeed, not only a temporary technological inadequacy but intrinsic to mobility and a hindrance that needs to be run over. Here in this paper, a load-aware allocation strategy is proposed and allocation is considered as an optimization problem with the aim of reducing the make span and the computational cost meeting the deadline constraints, with high resource utilization and is solved using Cuckoo-Search algorithm. The proposed approach is evaluated using Cloudsim framework and the results showed that our approach works better than other metaheuristic algorithms.**

*Keywords:* Cloud computing; load balancing; clustering; resource allocation.

## 1. Introduction

Cloud computing is one of the promising innovation in today's computing environment. The basic principle of cloud computing is that the information is not kept mainly in a single machine, but rather will be available in the server farms (datacenters). The clients can access the information with the help of an application programming interface which is a part of terminal hardware, provided it must be connected to the internet. The processing units in cloud are called virtual machines, and to decrease the execution time VM should run in parallel.

doi:10.1016/j.protcy.2016.08.116

The cloud computing framework is made up of specific components. There is a back end and a front end which needs to connect and communicate. The back end as the name proposes is the "not seen" end of the system, which is the group of system that forms the network. The front end comprises of the customer equipment both hardware and software that helps to connect with the network. With the advancement in wireless technologies like the third generation of mobile communication technology (3G), Bluetooth, wireless local area network (WLANs) and worldwide interoperability for microwave access (WIMAX) users are allowed to pick networks according to their necessities. Now a days applications focused at mobile devices are expanding plentifully in different areas, for example, entertainment, games, social networking, travel and news. But with versatility comes its intrinsic issues, for example,Some applications, particularly location based long range informal communication, process and utilize different sensor information. For example, acquiring a GPS reading, will draw a lot of battery and this restrains the users in receiving better services. Furthermore, some applications demand high computational capacities that require extensive processing such as image processing for video games, natural language processing, augmented reality, wearable computing. Allocation of static resources to the customer will result in either under or over utilization of resources. Therefore, resource allocation in cloud environment should be dynamic in nature.

## 2. Related works

The size and the intricacy of datacentres are growing day by day to take care of the growing demand for resources. Cloud computing providers must allot enough resources in order to meet the predetermined SLA. Most resource provisioning algorithms are designed in such a way that it must ensure both minimum response time and resource usage cost.Natasha.et.al proposed a technique to handle same priority requests [13], by managing resources using priority based approach. Here the requests with the attached priority are received by the resource allocator, which is the initiator. Here all needs will be first extracted, sorted and same priority requests will be put together. At that point the load required by each request is figured, and it is sorted. At that point the total load of the server is ascertained and limit is found. Zhang.et.al [15] proposed a dynamic heterogeneity aware provisioning in cloud which is fit for performing DCP in heterogeneous server farms. Here classification followed by prediction is done and after that DCP is done.Chandrashekar.et.al in [3] proposed a priority based distribution with a modified waiting queue. Here the proposed calculation responds to fluctuating workload by pre-empting the present executing assignment having lower priority with a high priority undertaking and if acquisition is impractical because of same priority, then it is checked whether global resources can host a virtual machine and the tasks are allocated.

A Mobile Message Passing Interface (MMPI) is talked about which is a system and a portable form of the standard MPI over Bluetooth where where mobile devices function as member resource providers.MMPI utilizes a completely interconnected mesh structure so that every node can communicate with the other, rather than the normal star system structure of typical piconets. Device discovery, and connections are taken care of by the libraries given in the system, hence there is no need of composing any Bluetooth particular code. The system is actualized in Java BlueCove, which is a third party library and is utilized to handle Bluetooth operations. A survey of existing application structure was led by Shiraz et al. [14] in 2013. They classified all distributed application processing frameworks into six main categories depending on area of use of framework. They discussed issues and challenges in current frameworks and suggested future areas for optimum distributed application processing frameworks development. In 2014, Shiraz et al. [14] investigated the runtime overload on mobile device while offloading mobile applications over the cloud. Before offloading mobile application on the cloud, it is profiled and partitioned for locating computational extensive components. Profiling and partitioning require additional computation resources from mobile device. Runtime portable application offload mechanism is assessed using smartsim and android application. Results demonstrate that CPU that CPU utilization of mobile device increases when partitioning is done for mobile application.

## 3. Proposed architecture and methodology

### *3.1. Mathematical formulation*

This section describes a mathematical model for load balancing and allocation problem based on Cuckoo Search Algorithm. Objective of this formulation is to form a load aware allocation strategy. The objective function here is to allocate the task to the virtual machine so as to achieve minimum execution time, minimum cost and meet the deadline constraints with a well-balanced load across all processors. Let Eij represents the amount of time taken by Task i to execute in resource j and $C_{ij}$ denotes the computational cost. This problem can be expressed as linear programming problem, as depicted below,

$$F(X) = Max\left[\sum_{i=1}^{n}\sum_{j=1}^{n} P(X_{ij})\right]$$

Where $$P(X_{ij}) = aU_j - bC_{ij} - cE_{ij} \qquad (1)$$

Subjected to:

Cluster Utilization $$u_j = Load_{lt} \leq Load \leq Load_{ut}$$

Where $C_{ij}$ = [task length/processing power of resource]* Res.cost

For Cuckoo Search strategy the required step-size is obtained by[20]

$$stepsize_j = .01 * \left(\frac{u_j}{v_j}\right)^{1/3} * V - X_{best} \qquad (2)$$

Where $u = \varnothing \cdot randn[d]$ $\qquad v = randn[d]$

Where $randn[d]$ generates a random number between [0,1].

Then $V$ can be generated as:

$$V = V + stepsize * randn[d]$$

The update process of Cuckoo search is defined by:

$$X_{best} \leftarrow f(X_{best}) \leq f(X_i)$$

The algorithmic control parameters of cuckoo search are the scale factor (β) and mutation probability value ($p_0$). In this β=1.50 and $p_0$=0.25 [10,11]

### *3.2. Load aware cuckoo based allocation*

The problem of finding an assignment of minimum makespan and cost is NP-hard. Due to the complexity of load balancing problem, most of researchers proposed meta-heuristic algorithms for solving the problem. Here Cuckoo search strategy is used to find an optimal solution. The objective function formulated is incorporated into the fitness function where it will then be used to measure the performance with respect to the objectives of the algorithm. The fitness function can be calculated by,

$$P(X_{ij}) = aU_j - bC_{ij} - cE_{ij}$$

### 3.3. *Proposed flow diagram*

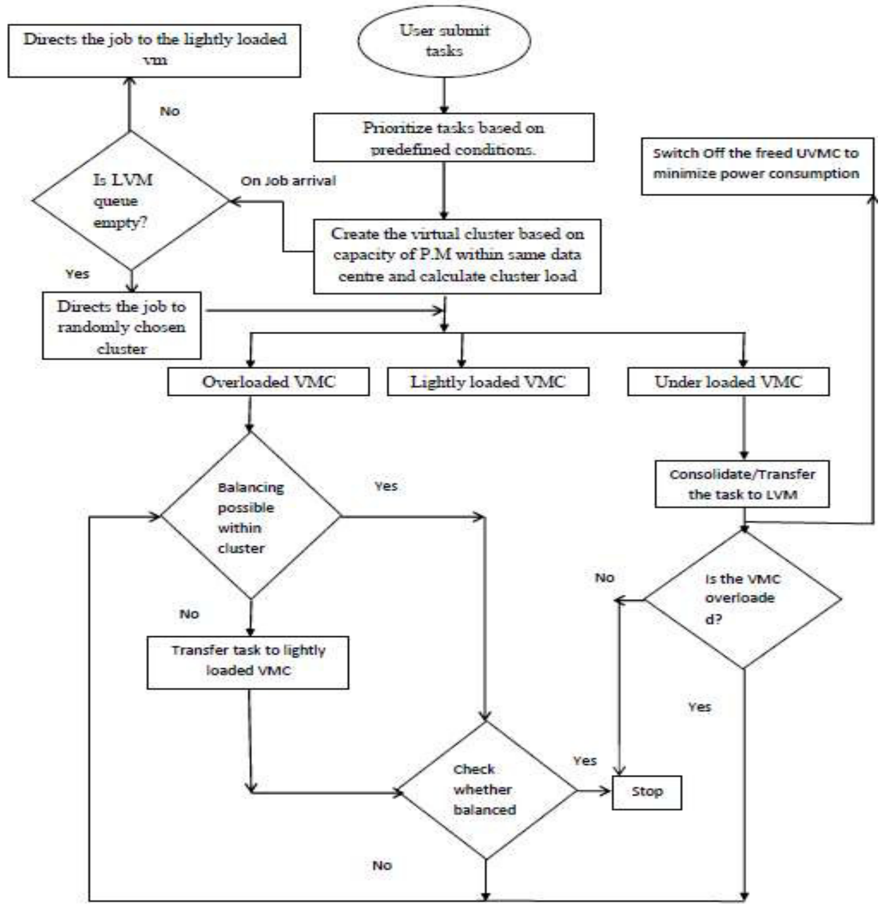
User submit tasks
Prioritize tasks based on predefined conditions.
Create the virtual cluster based on capacity of P.M within same data centre and calculate cluster load
On Job arrival
Is LVM queue empty?
No
Directs the job to the lightly loaded vm
Yes
Directs the job to randomly chosen cluster
Overloaded VMC
Lightly loaded VMC
Under loaded VMC
Switch Off the freed UVMC to minimize power consumption
Consolidate/Transfer the task to LVM
Is the VMC overloaded?
No
Yes
Balancing possible within cluster
Yes
No
Transfer task to lightly loaded VMC
Check whether balanced
Yes
Stop
No

Fig 1. Flow diagram of the proposed approach

The detailed working of the algorithm is described as follows. Initially clustering is done within the datacentre. Compute resources (on PMs) within a datacenter are packaged into racks and are typically organized as clusters of thousands of hosts for resource allocation purposes. Here for clustering density based clustering algorithm is used, where capacity of the physical machine is taken as a criterion for clustering. In this paper, it is assumed that all virtual clusters and virtual machines reside within the same datacentre, although they might be on different racks. And also it is assumed that network transmissions between the clusters are constant. As load-balancing is performed by the centralized Cuckoo based method, the first thing to do is to initialize a population of possible solutions. Jobs arrive at unknown intervals and are placed in the queue of scheduled tasks from where the jobs are assigned to the processor. Every time when a job is arrived at queue of scheduled tasks (task pool), the job is scheduled by using the scheduling algorithm and is placed in corresponding queue. Before allocating, the load on each clusters are analysed, and clusters are classified as Overloaded, Lightly loaded and Under Loaded Virtual Machine Clusters. If it is an under loaded Virtual Machine cluster, the tasks are consolidated to lightly loaded Virtual Machine cluster and the freed vm in UVMC are allowed to enter sleep mode for minimizing energy consumption. Then a queue of lightly loaded vm is maintained called LVM queue. The load balancing strategy first checks the information of lightly loaded vm present in the LVM queues. At a job arrival, the dispatcher consults the LVM queue. If the LVM queue is non-empty, the dispatcher removes the first lightly loaded processor from the LVM queue and directs the job to this lightly loaded virtual machine. If the LVM queue is empty, it is checked whether global resources can host a vm instance. If it is possible, a new vm instance is created and the job is deployed to that particular vm. If both the above conditions are not satisfied the job is allocated to a randomly chosen processor using cuckoo search. After allocation load checking is done. If the cluster is found to be overloaded two scenarios are considered. Intra $P_m$ and Inter $P_{m.}$ In Intra $P_m$, the overloaded vm instance is found out and the load is transferred into lightly loaded vm within the $P_m$. In Inter $P_m$ the task is migrated to the $P_m$ whose avg utilization is lesser than threshold within the same cluster. Considering the architecture of most cloud systems, a default CPU utilization of 70% is considered as a threshold.

**Procedure: Load-aware Cuckoo based Allocation ( )**

```
1. Clustering ( )
2. Load Checking ( )
3. At a job arrival, the dispatcher consults the LVM queue.
4. If (LVM queue)  $\neq \emptyset$
      The dispatcher removes the first vm from the LVM queue and directs the job to this VM.
5. If (LVM queue) = $\emptyset$   {
      If (globalResourcesCan Host vm), then
            {   Start new vm instance
                Add vm to Available vm list
                Deploy services on new vm    }
   Else   {
      The dispatcher directs the job to a randomly chosen cluster using Cuckoo ( )
       Load Checking ( )   }
6. If (OVMC)  {
       Intra $P_m$ ()
       Find the overloaded vm instance and transfer the load into lightly loaded Vm within the $P_{m.}$  }
   Else   {
        Inter $P_m$ ()
        Migrate the task to the $P_m$ whose avg utilization is lesser than threshold within the same cluster. }
```

## 4. Result analysis and discussions

In this section, the performance of the proposed load-aware allocation algorithm is evaluated. The proposed system is targeted on a large-scale Cloud datacenter. Therefore, it is necessary to conduct large- scale experiments to evaluate the algorithms. However, it is difficult to run large-scale experiments on a real-world infrastructure, especially when the experiments have to be repeated for different policies with the same conditions .Therefore, simulation has been chosen as a way to evaluate the proposed algorithm and Cloudsim framework is used to simulate the cloud environment. As discussed in previous section, the first step in CSA is initializing the population. This is done by a vector in which the length of the vector is taken as the number of resources. Equation 1 shows the fitness function of each solution. Instead of waiting for the CSA to converge, the algorithm is allowed to run for k cycles (k=15 in this paper). The decision was made because solutions generated in less than k generations may not be good enough. On the other hand, running the CSA for more than k generations may not very feasible, as too much it can cause an overhead. In equation 1, Eij represents the amount of time taken by $task_i$ to execute in resource $R_j$ and $C_{ij}$ denotes the computational cost. After initializing the first population randomly, the best nest is chosen. Next the Levy flight function is performed and fitness function is used to find the best nest. The experimental result of the proposed approach is shown below.The test runs were based on default value sets, except for the number of tasks which were allowed to vary. The Higher Threshold multiplier is fixed as 1.2 and Lower Threshold multiplier is fixed as 0.8 The algorithm is compared with PSO algorithm by considering the parameters such as the percentage of the deadline met, computational cost makespan and CPU utilization. Based on the comparison and the results from the experiment it is concluded that the proposed approach works better. The analysis of the result suggest that cuckoo search strategy works well minimizing the makespan and computational cost and the percentage of deadline met by the strategy is high compared to other meta heuristic algorithm.

### *4.1. Makespan evaluation*

Makespan is defined as the total execution time of all the tasks.

$$Makespan = \sum_{i=0}^{n} Executiontime\left(Task_i\right) \qquad (3)$$

Makespan is required to have a lower value. PSO algorithm fails to achieve this goal as it takes longer time to execute. Here by comparing the two algorithms, it is clear that the makespan of the cuckoo search algorithm is less compared to PSO approach. The execution times for different number of tasks are recorded for both cuckoo search as well as PSO in milliseconds and shown in Fig 2.

### 4.2. *Deadline Constraint Evaluation*

To analyze the algorithm in terms of meeting the user defined deadline, the percentage of deadline met for each algorithm is plotted. Overall comparison suggests that Cuckoo-search strategy is the most appealing algorithm for the scheduling presented in this paper. The percentage of deadline met by cuckoo search is considerably high when compared to PSO algorithm. Tardiness TR is taken to evaluate the percentage of deadline met.

$$TR_i = d_i - f_i \qquad (4)$$

Total Tardiness is the sum of the tardiness of the each task which did not get executed under the provided deadline. The average tardiness is defined by using the following formulae

$$Avg\_TR = \frac{\sum_{i=1}^{n} TR_i}{n} \qquad (5)$$

The number of non-delayed tasks is the total number of tasks whose finishing time was less than the deadline of the task, i.e., which finished inside the deadline given to them. The expected completion time is calculated as the mean of the completion time for the task at every resource. Fig 3 shows the percentage of deadline met by cuckoo search and PSO algorithms

### 4.3. *Cost Evaluation.*

The average execution costs obtained for each workflow are shown in Fig 4 .The algorithms should be able to generate a cost efficient schedule but not at the expense of a long execution time. There is no use in an algorithm generating very cheap schedules but not meeting the deadlines; the cost comparison is made therefore, amongst those heuristics which managed to meet the particular deadline in a given case. It is clear from the graph that the proposed strategy has very low computational cost and produces the output result. [12]

Cost= [task length/processing power of resource]*Res.cost (6)

It is found that in every case in which both algorithms meet the deadline, our approach incurs in cheaper costs, in some cases generating not only cheaper but faster solutions.

### *4.4. CPU Utilization*

In the proposed approach new virtual machines are identified and the incoming tasks are directed to a new virtual machine if the threshold utilization is exceeded for the Vm in which the tasks are running. By this proposed approach different virtual machines are equally balanced at some points and idle virtual machines are used efficiently. Fig 5 shows a comparison of the utilization values obtained with and without load balancing.

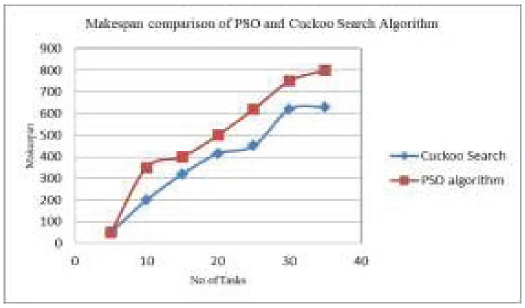

Fig 2. Makespan Comparison of Cuckoo search & PSO algorithm

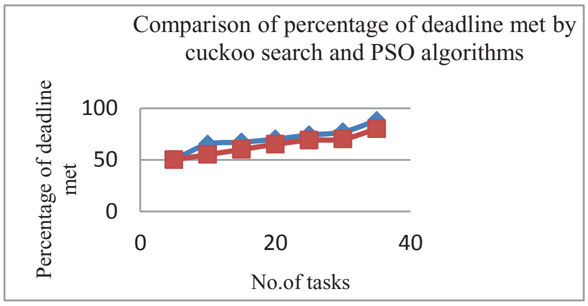

Fig 3. Comparison of percentage of deadline met by Cuckoo search & PSO

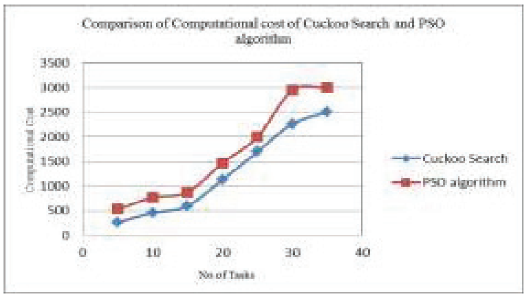

Fig 4. Comparison of computational cost of Cuckoo Search & PSO

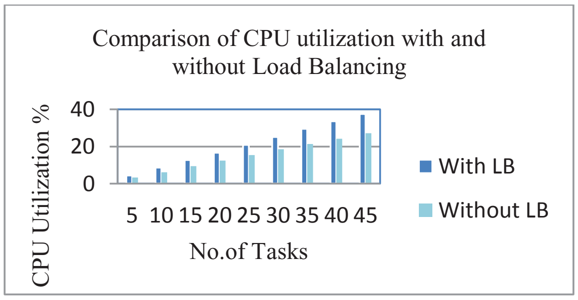

Fig 5. Comparison of CPU utilization with & without Load Balancing.

## 5.Conclusions

In this work a load-aware allocation strategy is presented. The scenario is modeled as an optimization problem which aims to minimize the overall execution time and the computational cost, meeting the deadline constraints, keeping the load in the clusters balanced. The problem was solved using a meta heuristic algorithm called Cuckoo search algorithm, and it is found to be much more precise and robust as compared to other existing optimization algorithms. After the analysis it is found that the algorithm significantly reduced the total executing time as well as computational time, and the percentage of the rate at which deadline is met is also more compared to other algorithms. As future work, different options for obtaining the initial pool of tasks can be explored as it has a significant impact on the performance of the algorithm. We would also like to experiment with different optimization strategies such as genetic algorithms and compare their performance with Cuckoo Search Algorithm. Moreover, this approach is designed to work in a single datacenter, and it can be designed to work between the datacenters and take into account fluctuations in network bandwidth. Finally it is aimed to implement the approach in a workflow engine so that it can be utilized for deploying applications in real time environments.

## References

[1] Ali M Alakeel, "A Guide To Dynamic Load Balancing In Distributed Computer Systems", International Journal of Computer Science and Network Security, Vol. 10 No. 6, June 2010.

[2] Ram Prasad Padhey, P. Goutam Prasad Rao, "Load Balancing in Cloud Computing Systems", Department of Computer Science and Engineering, National Institute of Technology, May 2011

[3] Chandrashekhar S.Pawar and Ranjnikant B Wagh,"Priority Based Dynamic resource allocation in Cloud Computing with Modified Waiting Queue",IEEE 2013.

[4]. Civicioglu, P., Besdok, E.: A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif. Intell. Rev. 39, 315–346 (2013)

[5] Amit Kumar Das,Choong Seon Hong." An Intelligent Approach for Virtual Machine and QoS Provisioning in Cloud Computing",IEEE 2013

[6] Khalid Elgazzar,Patrick Martin Hossam S Hassanein,"Empowering Mobile Service Provisioning Through Cloud Assistance",IEEE 2013,DOI 10.1109/UCC.2013.20.

[7] Hariharasudhan Viswanathan,Eun Kyung Lee,Ivan Rodeero and Dario Pompil,"An Autonomic Resource Provisioning Framework for Mobile Computing Grids", ACM International Conference(2012).

[8] David Escalnte and Andrew J. Korty,"Cloud Services: Policy and Assessment", EDUCAUSE Review,Vol. 46,Jul/Aug 11

[9] Parin. V. Patel, Hitesh. D. Patel, Pinal. J. Patel, "A Survey on Load Balancing in Cloud Computing" IJERT, Vol. 1, Issue 9, November 2012

[10]. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Model. Numer.Optim. 1, 330–343 (2010)

[11]. Civicioglu, P., Besdok, E.: A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif. Intell. Rev. 39, 315–346 (2013)

[12]. P. Shachee ,S. Richa, Double Level Priority based Optimization Algorithm for Task Scheduling in Cloud Computing, International Journal of Computer Applications,Vol 62 no.20, January 2013, pp-0975 – 8887

[13] Natasha,Nivit Gill,"Enhanced Priority Based Resource Allocation in Cloud Computing",IEEE 2013.

[14] M. Shiraz, E. Ahmed, A. Gani, Q. Han, Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing, J. Supercomput. (2014), http://dx.doi.org/10.1007/s11227-013-0988-6

[15]Qi Zang,Mohamed Fatn Zhani,Raouf Boutaba and Jospeh L Hellerstein,"Dynamic Heterogeneity Aware Resource Provisioning in the Cloud",IEEE 2013,DOI 10.1109/ICDS.2013.28

[16]Sandeep K. Sood , Rajinder Sandhu, Matrix based proactive resource provisioning in mobile cloud environment, Simulation Modelling Practice and Theory 50 (2015) 83–95.