2017 IEEE 5th International Conference on Future Internet of Things and Cloud

Cloud Task Scheduling based on Swarm Intelligence and Machine Learning

Gaith Rjoub*, Jamal Bentahar*
*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada

Email addresses: {g_rjoub, bentahar}@ciise.concordia.ca

***Abstract*—Cloud computing is the expansion of parallel computing, distributed computing. The technology of cloud computing becomes more and more widely used, and one of the fundamental issues in this cloud environment is related to task scheduling. However, scheduling in Cloud environments represents a difficult issue since it is basically NP-complete. Thus, many variants based on approximation techniques, especially those inspired by Swarm Intelligence (SI) have been proposed. This paper proposes a machine learning algorithm to guide the cloud choose the scheduling technique by using multi criteria decision to optimize the performance. The main contribution of our work is to minimize the makespan of a given task set. The new strategy is simulated using the CloudSim toolkit package where the impact of the algorithm is checked with different numbers of VMs varying from 2 to 50, and different task sizes between 30 bytes and 2700 bytes. Experiment results show that the proposed algorithm minimizes the execution time and the makespan between 7% and 75%, and improves the performance of the load balancing scheduling.**

***Keywords*—Cloud Computing, Particle swarm optimization, Ant Colony Optimization, Honey Bee Optimization, Machine Learning, CloudSim, Swarm Intelligence**

I. Introduction

Cloud computing has emanated as more than just a promising technology towards cooperating with large quantities of data. When mentioning technology that provides flexible resources and IT services based on the Internet, the term Cloud Computing is therefore used [1]. Cloud computing becomes more and more popular in large scale computing and data store recently because it enables the sharing of computing resources that are distributed all over the world, which leads to a natural development for data and computing centers with automated systems management, workload balancing, and virtualization technologies. Recently, several of applications are increasingly focusing on third-party resources hosted across the Internet, and each has varying capacity[2]. In the Cloud computing environment, different load balancing scheduling (a technique which divides the workload across multiple computing resources such as computers, hard drives, and network.) exists to ensure an appropriate utilization of resource consumption. It also attempts to fix the problem that all the processors in the system and every node in the network must share an equal amount of workload which is assigned to them. Load balancing scheduling plays a key role in efficient resource utilization in a cloud computing environment, while it is divided into two classes of technique which can be distinguished:

- The first class is the batch mode heuristic scheduling where jobs are queued in a set and collected as batches as they arrive in the system after which they get started after a fixed period. Some examples include First Come First Serve (FCFS), Round Robin (RR), Min-Max, and Min-Min [3], [4], and [5].
- The second class is on-line mode heuristic scheduling, where jobs are scheduled individually as they arrive in the system. This technique more feasible in a cloud environment as the systems may have different platforms and execution speed.

Task scheduling is one of the crucial technologies in cloud computing, and proper task scheduling is required so as to improve the efficiency and to minimize the execution time. The overall performance of cloud systems depends on the scheduling algorithm, and how to efficiently and rationally allocate the finite, heterogeneous and geographically distributed resources to meet the end-users requirements is an urgent issue for cloud service providers. In cloud computing, The purpose behind task scheduling is to specify a particular resource from all the available resources so that the efficiency of the computing environment increases [6]. In this paper, we proposed a hybrid approach, called Multi Label Classifier Chains Swarm Intelligence (MLCCSI). This approach is based on two strategies. The first strategy is the swarm intelligence, which we applied on, Ant Colony Optimization (ACO) algorithm, Artificial Bee Colony (ABC) algorithm and, Particle Swarm Optimization (PSO) algorithm to find the optimal resource allocation for each task in the dynamic cloud system. Then, the second strategy is the application of the machine learning algorithm (Classifier Chains) on the results from the three algorithms and generate a new hybrid model considering the size of the tasks and the number of the virtual machines.This strategy not only minimizes the makespan of a given tasks set, but it also adapts to the dynamic cloud computing system and balances the entire system load. The new scheduling strategy is simulated using the CloudSim version 2.1 toolkit package [7] and [8]. Roughly speaking, in our model, the machine learning algorithm required to predict a proper scheduling algorithm for every data center depending on the execution time. We use initial data from the iterate on the optimization algorithms to create a model for each data center in the cloud. These models can be further updated from a new test or real runs. The system can then use these models to make informed choices towards optimizing some externally defined criterion. The rest of the paper is organized as follows. Section II reviews relevant related work. Section III describes the approach, the optimization algorithms, and the prediction models. Section IV illustrates the implementation setup used to perform simulations and

978-1-5386-2074-8/17 $31.00 © 2017 IEEE
DOI 10.1109/FiCloud.2017.52

presents empirical results. Finally, Section V concludes the paper and discusses future perspectives.

II. Related Work

Previous researchers have proposed task scheduling techniques to minimize task execution time, maximize system performance and minimize cost using optimization algorithms such as particle swarm optimization (PSO), genetic algorithm (GA), bee colony optimization (BCO), ant colony optimization (ACO) [14, 1626]. These techniques have contributed to further developments of ideal solutions. With changing cloud computing environment, VMs are limited to handle the volume of the task that often arrives data centers. Thus, methods applied by existing researchers still need to be handled and to be improved with some new methods. Babu et al. [9] proposed a novel strategy for load balancing of tasks in cloud computing environments inspired by the behavior of honey bees, which helps to achieve even load balancing across virtual machine to maximize throughput. It considers the priority of task waiting in queue for execution in virtual machines. After the work load on VM has been calculated it decides whether the system is overloaded, under loaded, or balanced. Based on this VMs are grouped. While load balancing is an important element of scheduling, the strategy proposed is meant for scheduling independent tasks and not dependent tasks work flows. Lu and Gu [2011] proposed using an ACO to find the closest free cloud resource rapidly and to share some load of the overload cloud resources adaptively. In their study, the approach does not schedule a set of VMs on different cloud resources but just execute the ACO process when some VMs are overloaded. The ACO process just acts like the ant to search for food to find the optimal physical resource and to create new VMs to share the load of the overloaded VMs. Nishant et al. [10] also scheduled the cloud resource for an optimized load balance of various nodes, using an ACO-based algorithm. They debated that the ACO approach can first choose the node that has the greatest number of neighboring nodes. This way, the ant can travel in the maximum potential directions to find more nodes that are overloaded or underloaded. Furthermore, the ant will detect the heavy-load nodes, and reschedule some load to the light-load nodes. Zhan and Huo [11] proposed improved particle swarm optimization which the outcome of the evaluation showed that the proposed technique can minimize the job average execution time, and increase the rate of availability of resources in the environment. Where the PSO does not solve large scale optimization, then simulated support algorithm is added into the PSO algorithm which increases the convergence speed of PSO. cai et al. [12] proposed a swarm intelligence based algorithm which reduced searching time. Traditional PSO algorithm is modulated with a random factor to handle with the precocious concourse problem. Results show that proposed system is more workable, solid, and scalable than previous methods. The proposed method gives better performance when compared with the traditional PSO-based algorithm. Total time taken to complete the task is shorter and stable than traditional PSO-based method which results in low power consumption. Liu et al. [13], which discusses an algorithm for task scheduling that depends on genetic-ant colony algorithm. The improvement is having a strong optimistic feedback of ACO and taking into account the convergence rate of the algorithm. However, the convergence rate is strongly affected by choice of the initial pheromone. The global search capability of the GA is used to solve the optimal solution quickly and then converts it into the initial pheromone of ACO. Under the same conditions, the results shown by simulation experiments suggest that this algorithm exceeds the weights of GA and ACO. In [14] gupta and sharma have improved the ABC algorithm by adding one more step including a mutation operator after the process performed by the employed bees in ABC. The mutation operator is used after the employed bees have scouted the solution space. The selection of the food source is done by a random technique, and the mutation operator is performed if a mutation probability is satisfied. Through mutation, there is an opportunity of changing the local best position, and the algorithm may not be enclosed into local optima. When applying the mutation operator, new food sources are produced. Accordingly, the new generated food sources replace the older if their fitness value is better.The improved ant colony algorithm depends on partial swarm optimization which is known as ACA-PSO is proposed by Yang [15]. Initially, the ants are in the line up with ant colony algorithm for the completion of the traverse, and re-arrangement of the solutions, while keeping in view the confined and universal solutions. While ACA-PSO controls the short comings of the algorithm, it easily gets into confined solutions in cloud computing resource scheduling. Mello et al. [16] a routing load balancing policy for grid computing environments is presented. It uses routing concepts from computer networks to define a neighborhood and search an adequate computers to divide the applications workload. This algorithm is designed to equally distribute the workload of tasks of parallel applications over Grid computing environments. Route algorithms are indicated for environments where there are several heterogeneous computers and parallel applications that are composed of multiple tasks. When dealing with large scale systems, an absolute minimization of the total execution time is not the only objective of a load balancing strategy. The communication cost, induced by load redistribution, is also a critical issue. For this purpose, Yagoubi proposes in [17], a hierarchical load balancing model as a new framework to balance computing load in a Grid. This model suffers from bottlenecks. In [18], a Mathematical model of cloud computing framework using fuzzy bee colony optimization technique is presented. Honey Bee colony algorithm is used for web services in web servers that are scattered. Ramezani et al. [19] improved the complete multi objective model for task scheduling optimization. They considered the conflict between the tasks, and the authority of PSO algorithm regarding the accuracy, and the speed. In order to deliver an optimal solution for the presented model, a multi objective algorithm that is based on Multi Objective PSO (MOPSO) method was proposed. Jswarm package to multi objective Jswarm (MOJ) package has been used to calculate and implement the proposed model, with extending Cloudsim toolkit put on MOJ as its task scheduling algorithm. Optimal task organization among VMs is defined by MOJ in Cloudsim according to MOPSO algorithm.

III. SWARM INTELLIGENCE

Swarm Intelligence is an Artificial Intelligence (AI) technique, which is studied based on the monitoring of the collective behavior in biological activities such as ant/bee foraging, nest building, larval sorting, a division of labor, and

collaborative transport, etc. [20]. The number of applications of swarm intelligence is exponentially increasing in fields of, e.g., communications networks, combinatorial optimization, and robotics. Self-organized Services (SOS) in the Cloud with swarm intelligence as one possible solution have already exhibited their advantages over conventional SOS techniques in terms of adaptive routing, minimize the required cloud resources (i.e., virtual machines) load balancing, etc.

A. SI BASED CLOUD SCHEDULING

1) Ant Colony Optimization (ACO): The essential idea of ACO derives from the foraging behavior of ant colonies. When an ants group tries to search for the food, they use a special genius kind of chemical to communicate with each other. That chemical is referred to as a pheromone. Randomly the ants start to search their foods. Once the ants find a path to the food source, they leave pheromone on the path. An ant can keep track off the trails of the other ants to the food source by sensing pheromone on the ground. As this process continues, most of the ants attract to choose the shortest path as there have been a huge amount of pheromones accumulated on this path [21]. in the task scheduling, During an iteration of the ACO algorithm, each ant$k, k = 1,...,m$ (m is the number of the ants), builds a tour executing n (n is the number of tasks) steps in which a probabilistic transition rule is applied. The k-ant chooses VM j for next task i with a probability that is computed by Equation 1.

$$p_y^x(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}(t)]^{\alpha} * [\eta_{is}]^{\beta}} & if\, j \in allowed_k \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where

- $\tau_{ij}(t)$ shows the pheromone concentration at the t time on the path between task i and VM j.
- $allowed_k = \{0, 1, \cdots, n-1\}$.
- $\eta_{ij} = \dfrac{1}{d_{ij}}$ is the visibility for the t moment, calculated with heuristic algorithm and d_{ij} which expresses the expected execution time and transfer time of task i on VM j can be computed with Eq. (2).

$$d_{ij} = \frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j\_VM_j} + \frac{InputFilleSize}{VM\_bw_j} \tag{2}$$

where TL_Task_i is the total length of the task that has been submitted to VM_j, Pe_num_j is the number of VM_j processors, Pe_mips_j is the MIPS of each processor of VM_j Input File Size is the length of the task before execution and VM_bw_j is the communication bandwidth ability of the VM_j. Finally the two parameters α and β control the relative weight of the pheromone trail and the visibility information respectively. Algorithm 1 shows the ACO scheduling [22] where $Cloudlet_{list}$ the list of tasks, VM_{list} the list of virtual machines, and *tabu* indicates the allowed VMs for ant k in next step also *tabu* records the traversed VM by ant k.

Algorithm 1: Scheduling based ACO

Require: α, β, $max_{iterations}$, $Cloudlet_{list}$, VM_{list}
for i in $Cloudlet_{list}$ and k in VM_{list} **do**
pair $Cloudlet_i, VM_K \longleftarrow \tau_{i,j}(0) = C$ // pheromone(C)
$VM_K \longleftarrow Ant_j \longleftarrow randomPick(Ant_{pool})$
$Ant_j^{tabu} \longleftarrow add(VM_k)$
while NOT done **do**
  for $k = 1tom$ **do**
    $VM_N \longleftarrow select(Ant_k, VM_{list}, Cloudlet_{list})$
    $Ant_i^{tabu} \longleftarrow add(VM_s)$
  end for
  for $k = 1tom$ **do**
    $L_k \longleftarrow calculate()$ // L_k the length of the current best tour done by the ants.
  end for
    $\tau_{i,j} \longleftarrow update()$ // The local Pheromone value
    $pheromone_{global} \longleftarrow update()$
    increment (iterations)
end while

2) Artificial Bee Colony (ABC): The artificial bee colony algorithm (ABC), an optimization algorithm based on the intelligent foraging behavior of honey bee swarm was proposed by Karaboga in 2005 [23],[24]. This is nature inspired algorithm for self-organization. ABC contains three groups of bees: employed bees, onlookers, and scouts. The first half of the colony consists of the employed artificial bees and the second half includes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources. The employed bee of an abandoned food source becomes a scout. The search carried out by the artificial bees can be summarized as follows:

- Employed bees determine a food source within the neighborhood of the food source in their memory.
- Employed bees share their informations (distance, quality, direction and other informations) with onlookers within the hive and then the onlookers select one of the food sources.
- Onlookers select a food source within the neighborhood of the food sources chosen by themselves.
- An employed bee of which the source has been abandoned becomes a scout and starts to search a new food source randomly.

This algorithm has a similar principle in balance the work of the virtual machine. The ABC algorithm calculates the virtual machine workload then it decides whether it is overloaded, light weighted or balanced. The high priority of the task is off from the overload virtual machine and tasks are waiting for the light weight virtual machine. These tasks are known as scout bee in the next step. ABC inspired Load Balancing technique reduces the response time of VM and also reduces the waiting time of the task. Onlooker bees calculate the probability of estimating new solution around food source using following equation:

$$p_i(t) = \frac{fit_i(t)}{\Sigma_{i=1}^{TS} fit_i(t)} \tag{3}$$

where fit_i will be the Fitness value of task source i, and TS will be the total number of task sources. Algorithm 2 shows the ABC scheduling [25], where $Datacenter_{list}$ the list of data centers.

Algorithm 2: Scheduling based ABC

```
Require: Cloudlet_list, VM_list, Datacenter_list, fac_LB
Groups(q) ⟵ divide(Cloudlet_list)
for i = 1 to q do
    length_i ⟵ lengthofgroupk(Groups_i)
end for
for k = 1 to q do
    Cloudlet_L ⟵ max(Groups_k)
    whileGroups_k ≥ Groups_i | i = 1...qandi ≠ k do
        for s = 1 to n do
        Datacenter_s ⟵ select(Datacenter_list)
        if fac_LB ≤ VM_sAssigned(Datacenter) then
        assign(Cloudlet_L, Datacenter_i≠s (VM_leastLoad))
        else
        decrement (length_k)
        end if
        end for
    end while
end for
```

3) Particle Swarm Optimization (PSO): Particle Swarm Optimisation (PSO) is a swarm-based intelligence algorithm [26], influenced by the social behavior of animals such as a flock of birds finding a food source or a school of fish protecting themselves from a predator. A particle in PSO is analogous to a bird or fish flying through a search (problem) space. The movement of each particle is co-ordinated by a velocity which has both magnitude and direction. Each particle position at any instance of time is influenced by its best position and the position of the best particle in a problem space. The performance of a particle is measured by a fitness value, and in order to measure how well the particles position, the fitness function can be defined:

$$f = Min(\frac{VTime}{VRutilization}) \tag{4}$$

where $VTime$ to denote the execution time of VMs for executing all of the tasks, and $VRutilization$ to denote the resource utilization of VMs during the process of running the tasks. Particles in the search process update themselves by tracking two best-known positions. Best-known position known as a local best position is the individual best known position in terms of fitness value reached so far by the particle itself. Another best-known position known as a global best position is the best position in the entire population. Algorithm 5 show, the velocity, position, global best, local best update mechanisms. All these update mechanisms affect the search directions of PSO at later iterations. The velocity and position are updated as follows:

$$\begin{cases} v_i^{l+1} = wv_i^l + a_1\varphi_1(pb_i^l - p_i^l) + a_2\varphi_2(gb^l - p_i^l) \\ \text{and} \\ p_i^{l+1} = p_i^l + v_i^{l+1} \end{cases} \tag{5}$$

where the subscript i denotes the particle number; l the iteration number; pb_i^l the personal best position of the ith particle up to iteration l; gb^l the global best position so far; φ_1 and φ_2 two uniformly distributed random numbers used to determine the influence of pb_i and gb; and a_1 and a_2 two constant values denoting, respectively, the cognitive and social learning rate. Algorithm 3 shows the PSO scheduling [27].

Algorithm 3: Scheduling based PSO

```
Input: Task, Particles
Output: gBest
foreach P_i do
    pBest = Generate_initial_position(P_i)
foreach pBest of particle P_i do
    gBest = Max(pBest_1, pBest_2, ....)
repeat
    j ⟵ 1 ;
    while j ≤ m do
        Select the task t_j;
        Calculate_est(t_j);
        Allocate_task(t_j);
        j++;
    foreach particle P_i do
        Calculatecur_particle_fit; (current particle)
        if cur_particle_fit ¦ pBest_i_fit then
            Update(pBest_i);
        if cur_particle_fit< gBest_i _fit then
            Update( gBest_i);
    if the termination condition is met then
        Output 4gBest_i; Break;
    else
    foreach particle P_i do
            Update( P_i_velocity);
            Update( P_i_position);
until the termination condition is met ;
```

B. Multi-Label prediction based on SI

Multi-label classification (MLC) has attracted increasing attention in the machine learning community during the past few years. In this paper, we focus on a method called classifier chains (CC) [28]. As its name suggests, CC selects an order on the label set, a chain of labels, and trains a binary classifier for each label in this order. The difference with binary relevance (BR) is that the feature space used to induce each classifier is extended by the previous labels in the chain. These labels are treated as additional attributes, with the goal to model conditional dependence between a label and its predecessors. CC performs particularly well when being used in an ensemble framework, usually denoted as ensemble of classifier chains (ECC), which reduces the influence of the

label order.

$$P(y|x) = \Pi_{j=1}^{d} P(y_j|pa(y_j),x) \quad (6)$$

Where, $pa(y_j)$ represents the parent labels for y_j. Obviously, $|pa(y_j)| = p$, where p denotes the number of labels prior to y_j following the chain order. In the training phase, according to a predefined chain order, it builds d binary classifiers $h_1,...,h_d$ such that each classifier predicts correctly the value of y_j by referring to $pa(y_j)$ in addition to x. In the testing phase, it predicts the value of y_j in a greedy manner:

$$y_j^* = argmax P(y_j|pa(y_j),x), \qquad j = 1,.....,d. \quad (7)$$

IV. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed model (MLCCSI). First, we explain the implementation details and then present the experimental results that involve the makespan, and compare this results with the optimization algorithms ACO, ABC, and PSO.

A. Implementation and Setup

We implemented our framework in a 64-bit Windows 7 environment on a machine equipped with an Intel(R) i5-2400M CPU with Nvidia(Quadro) HD Graphics 3.10 GHz Processor and 8.192 GB RAM. We simulated the proposed model and the optimization Scheduling algorithms in the CloudSim toolkit [15]. This simulation mainly shows the advantage of the proposed model compared to the ACO, ABC, and PSO Algorithms in makespan term. The experiment is implemented with 3 data centers and 50 tasks between 30 and 2700 bytes with different numbers of VMs varying from 2 to 50 under the simulation platform. The resource situation is shown in Table 1. The computation workload of the task is 10000 MIPS (Million Instruction per second), and the manager of the three data centers both have space shared and time shared policy for VMs, but, for the VMs manager, we set Time shared algorithm for tasks. All the algorithms are tested by varying the number of the VMs with randomly varying the size of the tasks.

B. Simulations Results

In this section, we analyze the performance of our model based on the results of simulation done using CloudSim. We extended the classes of the CloudSim simulator to simulate our model. In the following illustrations, we compare the makespan of our model with the basic algorithms. In the first step, we apply the optimization algorithms when the virtual machines number has been fixed to 2, and the task sizes are varying between 30, and 2700 bytes. Moreover, we iterate the optimization while varying the number of virtual machines from 2, to 50. We have run each algorithm 112 times, and the average makespan of these runs is shown in Figures 1, 2, and 3.

We use the machine learning classifier chain algorithm to choose the future algorithm to be used to schedule the tasks in every data center while having the best task execution time, by using Gibbs sampler. The key idea of Gibbs sampling is that one only considers univariate conditional distributions, i.e. the

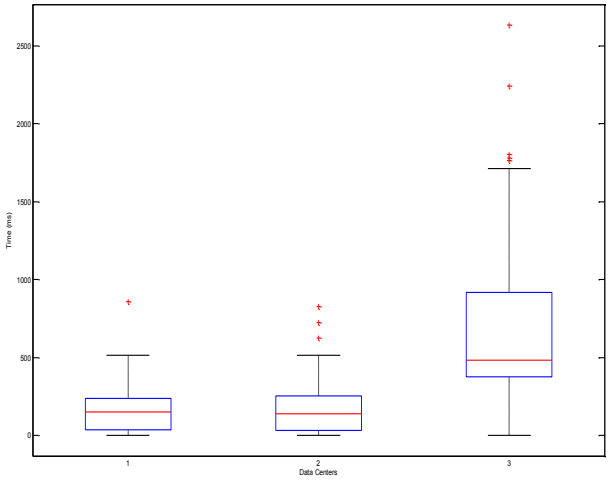

Fig. 1: The average makespan of 3 data centers for Ant Colony algorithm

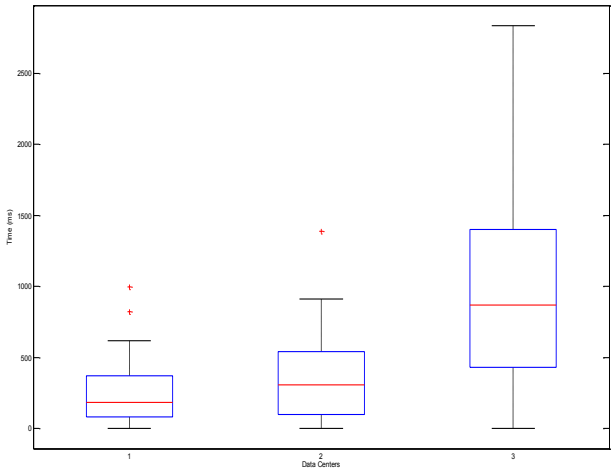

Fig. 2: The average makespan of 3 data centers for Artificial Bee Honey algorithm

distribution when all of the variables but one are assigned fixed values. This property makes Gibbs sampling a perfect fit for our fully connected conditional dependency network we build, where the univariate conditional distributions needed for Gibbs sampling are directly available from the conditional probabilistic predictors associated with each variable. The inference procedure of Gibbs sampling is very simple. We first choose a random ordering of the variable r, and initialize each variable y_1 to a value y_i. In each sampling iteration, we visit every variable in the given order, $y_i(1),...,y(k)$, where r maps the new order index into the original variable index. The new value of each variable $Y_r(i)$ is resampled according to the conditional predictor associated with it, $p(y|x,y_r(i),\theta_r(i))$. The idea behind Gibbs sampling is to approximate the joint distribution from the samples obtained from the conditional distributions. The sampler is expected to converge to a stationary distribution

TABLE I: PARAMETERS SETTING OF CLOUD SIMULATOR

Type	Parameters	Value
Data Center	Number of Datacenters	3
	Number of Hosts	6
	Type of Manager	Space shared and Time shared
	Datacenter Cost	1 –15
Virtual Machine (VM)	Total number of VMs	2 –50
	VM memory(RAM)	512(MB)
	Type of Manager	Time shared
	Bandwidth	1000 bit
Task	Total number of task	50
	Size of task	30 –2700 bytes
	Number of PEs requirement	2

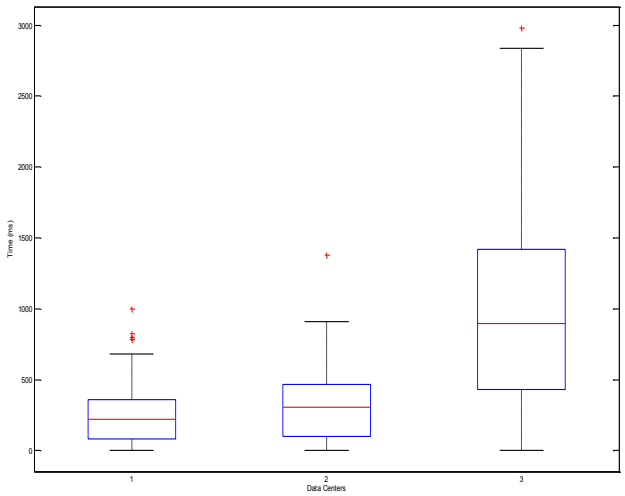

Fig. 3: Fig:The average makespan of 3 data centers for Particle Swarm Optimization Algorithm

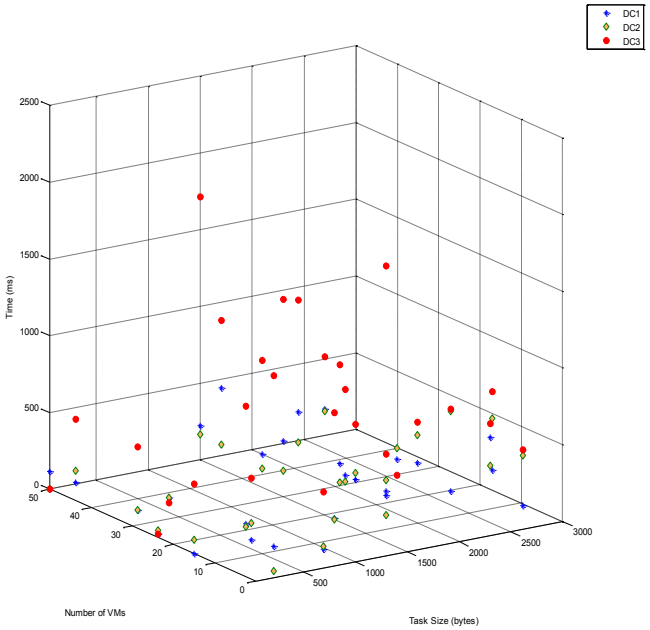

Fig. 4: Ant Colony algorithm

after some burn-in iterations. Then, it collects samples to recover the approximated joint distribution and determine the most probable explanation (MPE).

Figures 4, 5, and 6 show the average makespan on the data center one, two and three when we used the optimization algorithms ant colony, artificial honey, and particle swarm, respectively.

Furthermore, Figure 7 shows the makespan when used with the multilabel classifier chain machine learning algorithm, and it shows that the average makespan of the basic algorithm has been roughly reduced especially for ABC, and PSO algorithms on all the data centers. When we compare the results with ant colony algorithm, it shows that the time has been reduced 10% on data center one, 7% on data center two and 13% on data center three. Furthermore, the figure shows a clear time decrease on artificial honey bee algorithm, 51% on data center one, 75% on data center two, and 60% on data center three. Moreover, the decrease on makespan happens roughly on the particle swarm optimization algorithm, and the figure shows 63% decrease on data center one, 73% on data center two, and finally 68% on data center three.

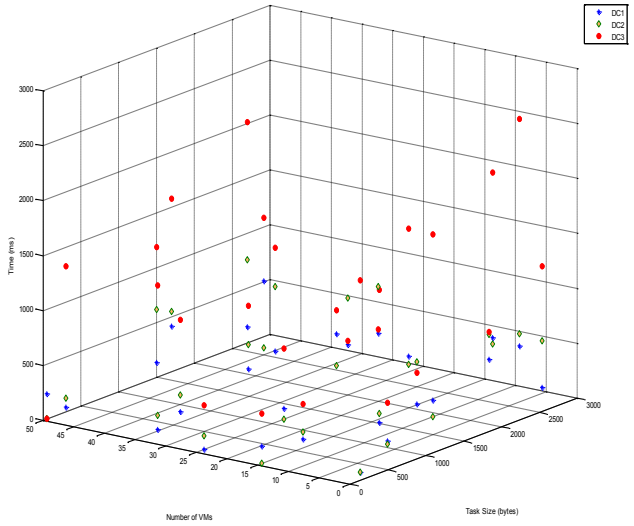

Fig. 5: Artificial Bee Honey algorithm

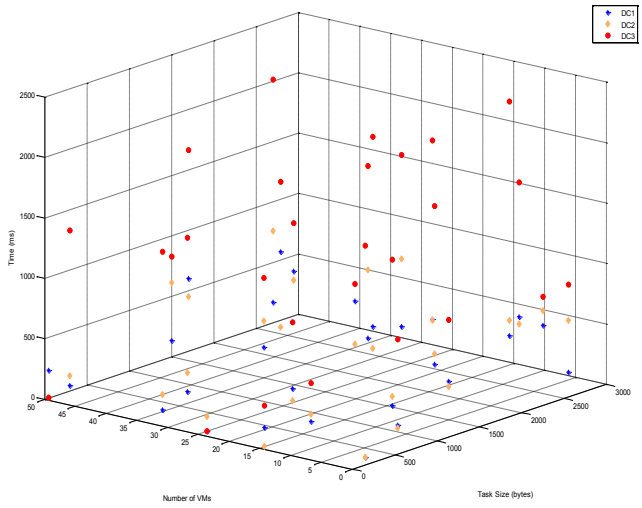

Fig. 6: PSO algorithm

V. Conclusion and Future Work

In this paper, we have proposed the MLCCSI model for achieving tasks scheduling with load balancing, and we have experimentally evaluated the MLCCSI model in applications with the number of VMs from 2 to 50 and varying tasks sizes from 30 to 2700. In this paper three scheduling algorithms are discussed such as ACO, ABC and PSO algorithm and a new scheduling model that uses the three discussed scheduling algorithms; is proposed and devised. The experimental results show that the MLCCSI model minimizes the makespan and utilizes the resources effectively than the standard optimization

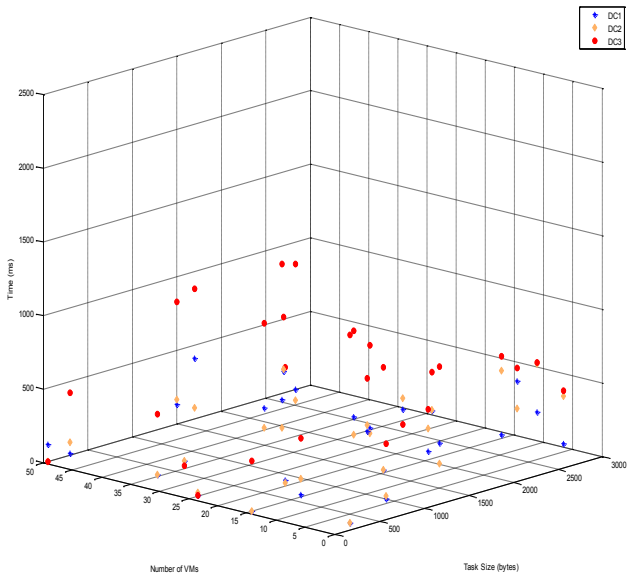

Fig. 7: MLCCSI model

algorithms. In this paper, we have used the makespan as fitness criteria for checking the fitness of the scheduling results, and The experimental result shows that the MLCCSI balance the entire system load effectively, and it clearly shows reducing on the average makespan between 7 % and 75 %. This method can be adapted to existing cloud computing systems for decreasing makespan and better resource utilization.

Promisingly, this work gives guidance to a new architecture that could be adopted to solve or mitigate several challenges that encounter the domain of cloud computing. It opens as well numerous research directions that seem worthy working on and investigating such as: (1) building efficient scheduling optimization algorithms in the cloud, (2) developing resource sharing and task scheduling models , and (3) building machine learning algorithms and dynamic models to minimize the required cloud resources and load balancing. As future work, we plan to study task optimization scheduling algorithms with different challenges such as migration and quality of service constraints. This study will include investigating several machine learning techniques to handle multi-label data such as k-nearest neighbors, decision trees, and neural networks with consideration of other relevant metrics such as the capacity of CPU, RAM, and bandwidth.

References

[1] M. Böhm, S. Leimeister, C. Riedl, and H. Krcmar, "Cloud computing–outsourcing 2.0 or a new business model for it provisioning?" in *Application management*. Springer, 2011, pp. 31–56.

[2] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE'08*. IEEE, 2008, pp. 1–10.

[3] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on iaas cloud systems," *Journal of Parallel and Distributed Computing*, vol. 72, no. 5, pp. 666–677, 2012.

[4] C. S. Pawar and R. B. Wagh, "Priority based dynamic resource allocation in cloud computing," in *Cloud and Services Computing (ISCOS), 2012 International Symposium on*. IEEE, 2012, pp. 1–6.

[5] Y. Han and X. Luo, "An effective algorithm and modeling for information resources scheduling in cloud computing," in *Advanced Cloud and Big Data (CBD), 2013 International Conference on*. IEEE, 2013, pp. 14–19.

[6] S. Sindhu and S. Mukherjee, "Efficient task scheduling algorithms for cloud computing environment," in *High Performance Architecture and Grid Computing*. Springer, 2011, pp. 79–83.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*. IEEE, 2009, pp. 1–11.

[8] R. N. Calheiros, R. Ranjan, C. A. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *arXiv preprint arXiv:0903.2525*, 2009.

[9] D. B. LD and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.

[10] X. Lu and Z. Gu, "A load-adapative cloud resource scheduling model based on ant colony algorithm," in *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 296–300.

[11] S. Zhan and H. Huo, "Improved pso-based task scheduling algorithm in cloud computing," *Journal of Information & Computational Science*, vol. 9, no. 13, pp. 3821–3829, 2012.

[12] Y. Cai, Z. Chen, and H. Min, "Improving particle swarm optimization algorithm for distributed sensing and search," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*. IEEE, 2013, pp. 373–379.

[13] C.-Y. Liu, C.-M. Zou, and P. Wu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing," in *Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on*. IEEE, 2014, pp. 68–72.

[14] M. Gupta and G. Sharma, "An efficient modified artificial bee colony algorithm for job scheduling problem," in *International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1, Issue6*. Citeseer, 2012.

[15] H. Yang, "Improved ant colony algorithm based on pso and its application on cloud computing resource scheduling," in *Advanced Materials Research*, vol. 989. Trans Tech Publ, 2014, pp. 2192–2195.

[16] R. F. de Mello, L. J. Senger, and L. T. Yang, "A routing load balancing policy for grid computing environments," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 1. IEEE, 2006, pp. 6–pp.

[17] B. Yagoubi and M. Meddeber, "Distributed load balancing model for grid computing," *ARIMA journal*, vol. 12, pp. 43–60, 2010.

[18] K. Mukherjee and G. Sahoo, "Mathematical model of cloud computing framework using fuzzy bee colony optimization technique," in *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on*. IEEE, 2009, pp. 664–668.

[19] F. Ramezani, J. Lu, and F. Hussain, "Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization," in *International Conference on Service-Oriented Computing*. Springer, 2013, pp. 237–251.

[20] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999, no. 1.

[21] T. Liao, T. Stützle, M. A. M. de Oca, and M. Dorigo, "A unified ant colony optimization algorithm for continuous optimization," *European Journal of Operational Research*, vol. 234, no. 3, pp. 597–609, 2014.

[22] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *Computer Engineering & Systems (ICCES), 2013 8th International Conference on*. IEEE, 2013, pp. 64–69.

[23] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, Tech. Rep., 2005.

[24] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[25] A. Al Buhussain, E. Robson, and A. Boukerche, "Performance analysis of bio-inspired scheduling algorithms for cloud environments," in *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*. IEEE, 2016, pp. 776–785.

[26] K.-L. Du and M. Swamy, "Particle swarm optimization," in *Search and Optimization by Metaheuristics*. Springer, 2016, pp. 153–173.

[27] H. Chen and W. Guo, "Real-time task scheduling algorithm for cloud computing based on particle swarm optimization," in *International Conference on Cloud Computing and Big Data in Asia*. Springer, 2015, pp. 141–152.

[28] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, pp. 333–359, 2011.