I.J. Intelligent Systems and Applications, **2014, 05, 25-36**
Published Online April 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2014.05.02

Cloud Task Scheduling for Load Balancing based on Intelligent Strategy

Arabi E. keshk, Ashraf B. El-Sisi, Medhat A. Tawfeek
Dept. of Computer Science, Faculty of Computers and Information, Menoufia University, Egypt
E-mail: arabikeshk@yahoo.com; ashrafelsisim@yahoo.com; medhattaw@yahoo.com

Abstract— Cloud computing is a type of parallel and distributed system consisting of a collection of interconnected and virtual computers. With the increasing demand and benefits of cloud computing infrastructure, different computing can be performed on cloud environment. One of the fundamental issues in this environment is related to task scheduling. Cloud task scheduling is an NP-hard optimization problem, and many meta-heuristic algorithms have been proposed to solve it. A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks. In this paper a cloud task scheduling policy based on ant colony optimization algorithm for load balancing compared with different scheduling algorithms has been proposed. Ant Colony Optimization (ACO) is random optimization search approach that will be used for allocating the incoming jobs to the virtual machines. The main contribution of our work is to balance the system load while trying to minimizing the make span of a given tasks set. The load balancing factor, related to the job finishing rate, is proposed to make the job finishing rate at different resource being similar and the ability of the load balancing will be improved. The proposed scheduling strategy was simulated using Cloudsim toolkit package. Experimental results showed that, the proposed algorithm outperformed scheduling algorithms that are based on the basic ACO or Modified Ant Colony Optimization (MACO).

Index Terms— Cloud Computing, Task Scheduling, Make Span, Ant Colony Optimization, Load Balancing

I. Introduction

Cloud computing has gained a lot of attention to be used as a computing model for a variety of application domains. Cloud computing services allow users to lease computing resources in the form of Virtual Machines (VMs) from large scale data centers operated by service providers [1]. Using cloud services, cloud users can deploy a wide variety of applications dynamically and on-demand usually addressed from three fundamental aspects: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [2]. Most cloud service providers use machine virtualization to provide flexible and cost effective resource sharing. It is the responsibility of the cloud service providers to manage its resources in an efficient way to make the needed resources available on demand to the cloud users. Consumers will be able to access applications and data from cloud anywhere in the world on demand [3]. In other words, the cloud appears to be a single point of access for all the computing needs of consumers. It is difficult to manually assign tasks to computing resources in clouds because hundreds of thousands of virtual machines (VMs) are used [4]. So, efficient algorithms are needed for task scheduling in the cloud environment with the goal of putting unused resource (virtual machines) cycles to work, distributing the load about them. A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks [5, 6]. Therefore, a dynamic task scheduling algorithm, such as ant colony optimization (ACO), is appropriate for clouds. ACO algorithm is a random search algorithm [7]. This algorithm uses a positive feedback mechanism and imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths travelled. Many researchers used ACO to solve NP-hard problems such as travelling salesman problem, graph coloring problem, vehicle routing problem, and scheduling problem [2, 8]. In this paper, cloud task scheduling based on Modified Ant Colony Optimization for Load Balancing (MACOLB) for cloud task scheduling is proposed. We use MACOLB algorithm to find the optimal resource allocation for tasks in the dynamic cloud system to minimize the makes pan of tasks on the entire system and increase the performance by balancing the load of the system. Then, this scheduling strategy was simulated using the Cloudsim toolkit package. Experimental results compared to ACO and MACO satisfy expectation and show that, MACOLB algorithm decrease the degree of imbalancing between available virtual machines and increase the overall performance. The rest of the paper is organized as follows. Section 2 introduces background and scans the related work. Cloudsim toolkit is presented in section 3. Section 4 covers the details of cloud task scheduling based on ACO, MACO and MACOLB algorithm. The implementation and simulation results are seen in section 5. Finally, some conclusion is put forward in section 6.

II. Background and Related Work

2.1 Cloud Computing Environment

Due to fast growth of cloud computing in the information technology landscape, several definitions have emerged. The cloud computing can be defined as a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers [9]. Cloud computing is a virtual pool of resources which are provided to users. It gives users virtually unlimited pay-per-use computing resources without the burden of managing the underlying infrastructure. The goal of cloud computing service providers is to use the resources efficiently and gain maximum profit [10]. This leads to task scheduling as a core and challenging issue in cloud computing. Cloud has an extra layer called virtualization layer. This layer acts as a creation, execution, management, and hosting environment for application services. The modeled VMs in the above virtual environment are contextually isolated but still they need to share computing resources- processing cores, system bus etc. Hence, the amount of hardware resources available to each VM is constrained by the total processing powers such CPU, the memory and system bandwidth available within the host [11].

The layered design of Cloud computing architecture is shown in Fig. 1. The top layer SaaS allows users to run applications remotely from the cloud by making use of services provided by the lower-layer services. PaaS includes operating systems and required services for a particular application. IaaS includes virtualized computers with guaranteed processing power and reserved bandwidth for storage and Internet access. The data-Storage-as-a-Service (dSaaS) provides storage that the consumer is used including bandwidth requirements for the storage.

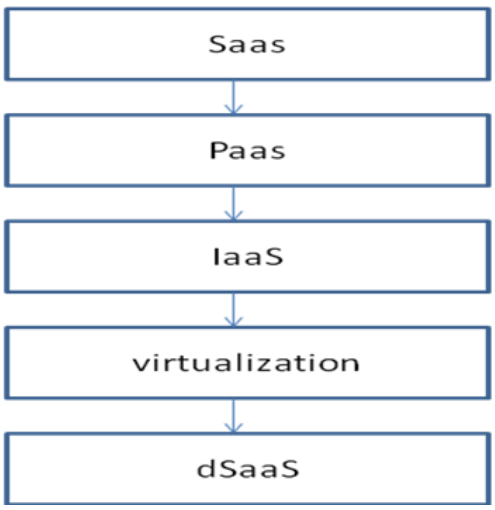

Fig. 1: Layered Design of Cloud Computing Architecture

2.2 Combinatorial Optimization Problem

In combinatorial optimization problems, we are looking for an object from a finite or possibly countably infinite set. This object is typically an integer number, a subset, a permutation, or a graph structure [12]. A combinatorial optimization problem P=(S,f) can be defined by:

- a set of variables $X = \{x_1, x_2, \ldots, x_n\}$;
- variable domains $D_1, \ldots, D_n$;
- constraints among variables;
- an objective function f to be minimized where $f : D_1 * \ldots * D_n \rightarrow R^+$

The set of all possible feasible assignments is: $S = \{s = \{(x_1,v_1), \ldots (x_n,v_n)\} \mid v_i \in D_i$ s satisfies all the constraints$\}$ S is usually called a search (or solution) space, as each element of the set can be seen as a candidate solution. To solve a combinatorial optimization problem one has to find a solution $s^* \in S$ with minimum objective function value [12]. Examples for combinatorial optimization problems are the Travelling Salesman Problem (TSP), the Quadratic Assignment Problem (QAP), timetabling and scheduling problems. Due to the practical importance of combinatorial optimization problems, many algorithms to tackle them have been developed. These algorithms can be classified as either complete or approximate algorithms. Complete algorithms are guaranteed to find for every finite size instance of a combinatorial optimization problem an optimal solution in bounded time. In approximate methods we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time especially for combinatorial optimization problems that are NP-hard [13]. Among the basic approximate methods we usually distinguish between constructive methods and local search methods. Constructive algorithms generate solutions from scratch by adding components to an initially empty partial solution until a solution is complete. Local search algorithms start from some initial solution and iteratively try to replace the current solution by a better solution in an appropriately defined neighborhood of the current solution [13].

In past four decades, a new kind of approximate algorithm has emerged which basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. This class of algorithms includes ACO, simulated annealing, tabu search and others [12]. A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions [13].

2.3 Ant algorithms

Ant algorithms are one of the most popular examples of swarm intelligence systems, in which a number of ant inspired agents which are specialized in particular sophisticated functionality follow simple rules with no centralized control. The complex global behavior emerges from their local interactions using pheromone. There are many different kinds of ACO algorithm, i.e., Ant Colony System (ACS) [14], Max-Min Ant System (MMAS) [15], Rank-based Ant System (RAS) [16], Fast Ant System (FANT) [17] and Elitist Ant System (EAS) [18]. ACO uses the pseudo-random-proportional rule to replace state transition rule for decreasing computation time of selecting paths and update the pheromone on the optimal path only. It is proved that it helps ants search the optimal path. In this paper, MACOLB approach has been proposed. The tasks/requests (application services) are assigned or allocated to these VMs of different processing powers (to the most powerful VM and then to the lowest) with balancing the load of the virtual machines. Hence, the performance parameter such as overall makespan time is optimized.

2.4 Related Work

Millions of user share cloud resources by submitting their computing task to the cloud system. Scheduling these millions of task is a challenge to cloud computing environment. Optimal resource allocation or task scheduling in the cloud should decide optimal number of systems required in the cloud so that the total cost is minimized. Task scheduling is well studied within the computer operating systems [19]. Most of them can be applied to cloud environment with suitable modifications. In the following we introduce several methods for Cloud. The Round Robin (RR) algorithm focuses on the fairness problem. RR uses the ring as its queue to store jobs. Each job in queue has the same execution time and it will be executed in turn. If a job can't be completed during its turn, it will store back to the queue waiting for the next turn. The advantage of RR algorithm is that each job will be executed in turn and they don't have to wait for the previous one to complete. But if the load is heavy, RR will take long time to complete all jobs. Priority scheduling algorithm gives each job a priority value and uses it to dispatch jobs. The priority value of each job depends on the job status such as the requirement of memory sizes, CPU time and so on. The main problem of this algorithm is that it may cause indefinite blocking or starvation if the requirement of a job is never being satisfied. The First Come First Serve (FCFS) algorithm is a simple job scheduling algorithm. A job which makes the first requirement will be executed first. The main problem of FCFS is its convoy effect. If all jobs are waiting for a big job to finish, the convoy effect occurs. The convoy effect may lead to longer average waiting time and lower resource utilization [19]. The Fastest Processor to Largest Task First (FPLTF) algorithm schedules tasks to resources according to the workload of tasks in the grid system. The algorithm needs two main parameters such as the CPU speed of resources and workload of tasks. The scheduler sorts the tasks and resources by their workload and CPU speed then assigns the largest task to the fastest available resource. If there are many tasks with heavy workload, its performance may be very bad [20]. Cloud scheduling is categorized at user level and system level [3]. At user level scheduling deals with problems raised by service provision between providers and customers [4, 11]. The system level scheduling handles resource management within datacenters [2, 9, 10]. A novel approach of heuristic-based request scheduling at each server, in each of the geographically distributed datacenters, to globally minimize the penalty charged to the cloud computing system is proposed in [21]. A new fault tolerant scheduling algorithm MaxRe is proposed in [22]. This algorithm incorporates the reliability analysis into the active replication schema, and exploits a dynamic number of replicas for different tasks. Scheduling based genetic algorithm is proposed in [23-25]. This algorithms optimizes the energy consumption, carbon dioxide emissions and the generated profit of a geographically distributed cloud computing infrastructure. The QoS Min-Min scheduling algorithm is proposed in [26]. Min-min is heuristic used for batch mode scheduling (In batch mode, tasks are scheduled only at some predefined time). This enables batch heuristics to know about the actual execution times of a larger number of tasks. An optimized algorithm for virtual machine placement in cloud computing scheduling based on multi-objective ant colony system algorithm in cloud computing is proposed in [2]. Scheduling in grid environment based ACO algorithms are proposed in [27-29]. The existing scheduling techniques in clouds, consider parameter or various parameters like performance, makespan, cost, scalability, throughput, resource utilization, load balancing, fault tolerance, migration time or associated overhead. In this paper, cloud task scheduling based ACO approach has been proposed for allocation of incoming jobs to virtual machines (VMs) considering in our account only makespan and load balancing to help in utilizing the available resources optimally, minimize the resource consumption and achieve a high user satisfaction.

III. Cloudsim

Simulation is a technique where a program models the behavior of the system (CPU, network etc.) by calculating the interaction between its different entities using mathematical formulas, or actually capturing and playing back observations from a production system [30]. Cloudsim is a framework developed by the GRIDS laboratory of university of Melbourne which enables seamless modeling, simulation and experimenting on designing cloud computing infrastructures [30].

3.1 Cloudsim Characteristics

Cloudsim can be used to model datacenters, host, service brokers, scheduling and allocation policies of a large scaled cloud platform. Hence, the researcher has used Cloudsim to model datacenters, hosts, VMs for experimenting in simulated cloud environment [31]. Cloud supports VM provisioning at two levels:-

1. At the host level – It is possible to specify how much of the overall processing power of each core will be assigned to each VM. Known as VM policy Allocation.
2. At the VM level – the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine (Known as VM Scheduling) [31].

In this paper, the ACO algorithm will be used for allocation of incoming batch jobs to virtual machines (VMs) at the VM level (VM Scheduling). All the VMs in a datacenter not necessary have a fixed amount of processing power but it can vary with different computing nodes. And then to these VMs of different processing powers, the tasks/requests (application services) are assigned or allocated to the most powerful VM and then to the lowest and so on. Hence, the performance parameter such as overall makespan time is optimized (increasing resource utilization ratio.) and the cost will be decreased.

3.2 Cloudsim Data Flow

Each datacenter entity registers with the Cloud Information Service registry (CIS). CIS provides database level match-making services; it maps user requests to suitable cloud providers. The DataCenterBroker consults the CIS service to obtain the list of cloud providers who can offer infrastructure services that match application's quality of service, hardware, and software requirements. In the case match occurs the broker deploys the application with the cloud that was suggested by the CIS [30].

3.3 The Cloudsim Platform

The main parts of Cloudsim that are related to our experiments in this paper and the relationship between them are shown in Fig. 2.

- CloudInformationService: It is an entity that registers datacenter entity and discovers the resource.
- Datacenter: It models the core infrastructure-level services (hardware), which is offered by cloud providers. It encapsulates a set of compute hosts that can either be homogeneous or heterogeneous.
- DatacenterBroker: It models a broker, which is responsible for mediating negotiations between SaaS and cloud providers.
- VmAllocation: A provisioning policy which is run in datacenter level helps to allocate VMs to hosts.
- VmScheduler: This is an abstract class implemented by a host component that models the policies (space-shared, time-shared) required for allocating processor cores to VMs. It is run on every host in datacenter.
- Host: It models a physical server.
- Vm: It models a virtual machine which is run on cloud host to deal with the cloudlet.
- Cloudlet: It models the cloud-based application services.
- CloudletScheduler: This abstract class is extended by the implementation of different policies that determine the share (space-shared, time-shared) of processing power among cloudlets in a VM [31].

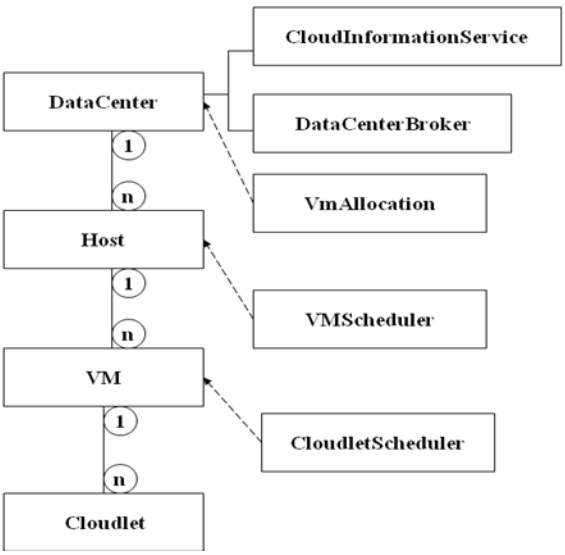

Fig. 2: Main Parts of Cloudsim Related To Our Experiments

IV. Cloud Scheduling Based ACO, MACO and MACOLB

4.1 Cloud Scheduling Based ACO

The basic idea of ACO is to simulate the foraging behavior of ant colonies. When an ants group tries to search for the food, they use a special kind of chemical to communicate with each other. That chemical is referred to as pheromone. Initially ants start search their foods randomly. Once the ants find a path to food source, they leave pheromone on the path. An ant can follow 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 [8]. The advantages of the algorithm are the use of the positive feedback mechanism, inner parallelism and extensible. The disadvantages are overhead and the stagnation

phenomenon, or searching for to a certain extent, all individuals found the same solution exactly, can't further search for the solution space, making the algorithm converge to local optimal solution [7]. It is clear that an ACO algorithm can be applied to any combinatorial problem as far as it is possible to define:

1. Problem representation which allows ants to incrementally build/modify solutions.
2. The heuristic desirability η of edges.
3. A constraint satisfaction method which forces the construction of feasible solutions.
4. A pheromone updating rule which specifies how to modify pheromone trail τ on the edges of the graph.
5. A probabilistic transition rule of the heuristic desirability and of pheromone trail [32].

In this section, cloud task scheduling based ACO algorithm will be presented. Decreasing the makespan of tasks is the basic ideas from the Proposed Method.

1. Problem representation: The problem is represented as a graph G = (N, E) where the set of nodes N represents the VMs and tasks and the set of edges E the connections between the task and VM as shown in Fig. 3. All ants are placed at the starting VMs randomly. During an iteration ants build solutions to the cloud scheduling problem by moving from one VM to another for next task until they complete a tour (all tasks have been allocated). Iterations are indexed by t, $1 < t < t_{max}$, where t_{max} is the maximum number of iterations allowed.

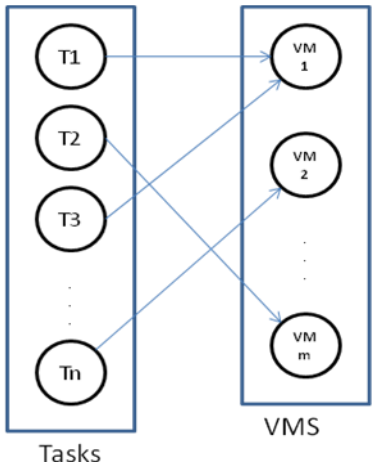

Fig. 3: Problem Representation of Task Scheduling Based ACO

2. Heuristic desirability: A very simple heuristic is used: the inverse of expected execution time of the task i on VM j.
3. Constraint satisfaction: The constraint satisfaction method is implemented as a simple, short-term memory, of the visited VM, in order to avoid visiting a VM more than once in one ACO procedure and minimize time of the assigned couplings (task and VM).
4. Pheromone updating rule: It is the one typical of Ant System (see (3) to (7)): pheromone evaporates on all edges and new pheromone is deposited by all ants on visited edges; its value is proportional to the quality of the solution built by the ants.
5. Probabilistic transition rule: The probabilistic transition rule, called random proportional, is the one typical of Ant System as in (1).

The pseudo code of the proposed ACO procedure and scheduling based ACO are shown in Fig. 4 and Fig. 5 respectively [32]. The main operations of the ACO procedure are initializing pheromone, choosing VM for next task and pheromone updating as following:

***Input**: List of Cloudlet (Tasks) and List of VMs*
***Output**: the best solution for tasks allocation on VMs*
***Steps**:*
*1. **Initialize**:*
* **Set** Current_iteration_t=1.*
* **Set** Current_optimal_solution=null.*
* **Set** Initial value $\tau_{ij}(t)=c$ for each path between tasks and VMs.*
*2. **Place** m ants on the starting VMs randomly.*
*3. **For** k :=1 to m do*
* **Place** the starting VM of the k-th ant in $tabu_k$.*
* **Do** ants_trip while all ants don't end their trips*
* **Every** ant chooses the VM for the next task according to (1).*
* **Insert** the selected VM to $tabu_k$.*
* **End** Do*
*4. **For** k :=1 to m do*
* **Compute** the length L_k of the tour described by the k-th ant according to (4).*
* **Update** the current_optimal_solution with the best founded solution.*
*5. **For** every edge (i,j), apply the local pheromone according to (5).*
*6. **Apply** global pheromone update according to (7).*
*7. **Increment** Current_iteration_t by one.*
*8. **If** (Current_iteration_t < t_{max})*
* **Empty** all tabu lists.*
* **Goto** step 2*
* **Else***
* **Print** current_optimal_solution.*
* **End** If*
*9. **Return***

Fig. 4: Pseudo Code of ACO Procedure

A. Initializing Pheromone

The amount of virtual pheromone trail $\tau_{ij}(t)$ on the edge connects task i to VM j. The initial amount of pheromone on edges is assumed to be a small positive constant τ_0 (homogeneous distribution of pheromone at time t = 0).

B. VM Choosing Rule for Next Task

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 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 (1).

***Input**: incoming Cloudlets and VMs List*
***Output**: print "scheduling completed and waiting for more Cloudlets"*
***Steps**:*
*1. **Set** Cloudlet List =null and temp_List_of_Cloudlet=null*
*2. **Put** any incoming Cloudlets in Cloudlet List in order of their arriving time*
*3. **Do** ACO_P while Cloudlet List not empty or there are more incoming Cloudlets*
* **Set** n= size of VMs list*
* **If** (size of Cloudlet List greater than n)*
*   **Transfer** the first arrived n Cloudlets from Cloudlet List and put them on temp_List_of_Cloudlet*
* **Else***
*   **Transfer** all Cloudlets.from Cloudlet List and put them on temp_List_of_Cloudlet*
* **End If***
* **Execute** ACO procedure with input temp_List_of_Cloudlet and n*
* **End** Do*
*4. **Print** "scheduling completed and waiting for more Cloudlets"*
*5. **Stop***

Fig. 5: Pseudo Code of Scheduling Based ACO

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}]^{\beta}} & if\ j \in allowed_k \\ 0, & otherwise \end{cases} \quad (1)$$

Where $\tau_{ij}(t)$ shows the pheromone concentration at the t time on the path between task i and VM j, $allowed_k = \{0,1,\ldots,n-1\}$-$tabu_k$ express the allowed VMs for ant k in next step and $tabu_k$ records the traversed VM by ant k, and $\eta_{ij}=1/d_{ij}$ is the visibility for the t moment, calculated with heuristic algorithm and d_{ij} which expresses the expected execution time and transfer time of the task i on VM j can be computed with (2).

$$d_{ij} = \frac{TL\_Task_i}{Pe\_num_j * Pe\_mips_j\_VM_j} + \frac{InputFileSize}{VM\_bw_j} \quad (2)$$

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, InputFileSize 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 in (1) are used to control the relative weight of the pheromone trail and the visibility information respectively.

C. Pheromone Updating

After the completion of a tour, each ant k lays a quantity of pheromone $\Delta\tau_{ij}^{k}(t)$ computed by (3) on each edge (i,j) that it has used.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k(t)} & if\ (i,j) \in T^k(t) \\ 0 & if\ (i,j) \notin T^k(t) \end{cases} \quad (3)$$

Where $T^k(t)$ is the tour done by ant k at iteration t, $L^k(t)$ is its length (the expected makespan of this tour) that is computed by (4), and Q is a adaptive parameter. It is very helpful in the cloud environment to depend on the result in the past task scheduling.

$$L^k(t) = arg\ max_{j\in J}\{sum_{i\in IJ}(d_{ij})\} \quad (4)$$

Where, IJ is the set of tasks that assigned to the VM_j. After each iteration pheromone updating which is applied to all edges is refreshed by (5).

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (5)$$

Where ρ is the trail decay, $0 < \rho < 1$ and $\Delta\tau_{ij}(t)$ is computed by (6).

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \quad (6)$$

When all ants complete a traverse, an elitist is an ant which reinforces pheromone on the edges belonging to the best tour found from the beginning of the trial (T^+), by a quantity Q/L^+, where L^+ is the length of the best tour (T^+). This reinforcement is called global pheromone update and computed by (7).

$$\tau_{ij}(t) = \tau_{ij}(t) + \frac{Q}{L^+}\ if\ (i,j) \in T^+ \quad (7)$$

4.2 Cloud Scheduling Based MACO

The MACO algorithm inherits the basic ideas from ACO algorithm to decrease the computation time of tasks executing [33]. It is similar to ACO but has four major differences as following:

A. VM Choosing Rule for Next Task

The Rule of choosing VM for next task is modified to (8).

$$P_{ij}^{k}(t) = \begin{cases} \arg\ max_{s\in allowed_k}\{(\alpha * \tau_{is}(t)) + ((1-\alpha) * \eta_{is})\} & if\ q \le q_0 \\ J & if\ q > q_0 \end{cases} \quad (8)$$

Where α is a parameter that allow a user to control the relative importance of pheromone trail and q is a random number uniformly distributed in [0, 1]. If q is greater than q_0, this process is called exploration; otherwise it is called exploitation [34]. Our principle to progressively adapt the system by tuning q_0 goes from 0 to 1, in order to favor exploration in the initial part of the algorithm and then favor exploitation. J is a random variable selected according to the following random-proportional rule probability distribution (9) which is the probability that ant k chooses to assign VM j to task i.

$$J = \begin{cases} \frac{(\alpha * \tau_{ij}(t)) + ((1-\alpha) * \eta_{ij})}{\sum_{s \in allowed_k} (\alpha * \tau_{is}(t)) + ((1-\alpha) * \eta_{is})} & if\ j \in allowed_k \\ 0, & otherwise \end{cases} \quad (9)$$

There are two reasons for adopting the above method to calculate the selection probability. The first is the simplicity as only one control parameter, i.e. α, is used to map the relative importance of quantity of pheromone and the desirability of each movement [35]. The second reason is the computational efficiency of this method as multiplication operations are used instead of exponentiations.

B. Local Pheromone Updating

The local pheromone update rule, which is applied to all edges, is replaced by (10).

$$\tau_{ij}(t) = (1 - \rho_l)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (10)$$

Where ρ_l is the trail decay, $0 < \rho_l < 1$, the following adaptive formula (11) is proposed to compute ρ_l:

$$\rho_l = \frac{|L^K(t) - L^+|}{L^K(t) + L^+} + 0.1 \quad (11)$$

This formula removes pheromone from edges that belong to the worst tours. This has the effect of making the visited edges of worst tours less and less attractive. So, giving negative reinforcement to bad tours increases the convergence speed toward good solutions.

C. Global Pheromone Updating

The global pheromone update rule, which is applied to all edges belonging to the best tour (T^+), is modified by (12). The modification of global pheromone update is applied but in different shape in [2].

$$\tau_{ij}(t) = (1 - \rho_g)\tau_{ij}(t) + \frac{\rho_g * \lambda}{L^+} \quad (12)$$

Where λ is an adaptive coefficient and ρ_g ($0 < \rho_g < 1$) is the pheromone evaporation parameter of global updating, they computed by (13) and (14) respectively:

$$\lambda = \frac{m}{t - ni_s + 1} \quad (13)$$

and

$$\rho_g = \frac{\rho_l}{2} \quad (14)$$

The λ coefficient used to control how a solution s contributes to pheromone information over time, *m* is the number of ants, ni_s represents the number of iterations that the best solution not changed. This global updating rule tries to increasing the learning of ants.

D. The Control Parameter α

The Parameter α controls the relative weight of the pheromone trail and the visibility information, and computed by the following adaptive formulas:

$$\alpha = \frac{\rho_l * m * L^k(t)}{t_{max}} \quad (15)$$

This adaptive rule tries to enhance the selection of weight pheromone trail and the visibility information. When the variation of pheromone concentration is high we give the visibility information high weight over pheromone trail and vice versa

4.3 Cloud Scheduling Based MACOLB

The MACOLB algorithm inherits the basic ideas from ACO and MACO algorithm to improve the execution of cloud tasks. In this section we introduce MACOLB algorithm to improve the performance of the scheduling problems in cloud computing. It is similar to MACO but has an added load balancing factor. The Rule of choosing VM for next task is modified to (16).

$$P_{ij}^k(t) = \begin{cases} \arg\ max_{s \in allowed_k} \left\{ \left((\alpha * \tau_{is}(t)) + ((1-\alpha) * \eta_{is})\right) + LB_s \right\} & if\ q \le q_0 \\ J & if\ q > q_0 \end{cases} \quad (16)$$

Where, LB_s is the load balancing factor of VM s to minimize the degree of imbalance which is defined by (17). The LB_s here learn from the result in the past task scheduling.

$$LB_s = 1 - \frac{Tot_s - LT_{avg}}{Tot_s + LT_{avg}} \quad (17)$$

Where LT_{avg} is the average execution time of the virtual machines in the last calling of ACO procedure and Tot_s is the expected execution time of the tasks that have been submitted to VM s which is defined by (18).

$$Tot_j = sum_{i \in IJ}(d_{ij}) \quad (18)$$

Equation (9) that computes the value J also modified to (19).

$$J = \begin{cases} \frac{(\alpha * \tau_{ij}(t)) + ((1-\alpha) * \eta_{ij}) + LB_j}{\sum_{s \in allowed_k} (\alpha * \tau_{is}(t)) + ((1-\alpha) * \eta_{is}) + LB_s} & if\ j \in allowed_k \\ 0, & otherwise \end{cases} \quad (19)$$

V. Implementation and Experimental Results

We assume that tasks are mutually independent i.e. there is no precedence constraint between tasks and tasks are not preemptive and they cannot be interrupted or moved to another processor during their execution.

5.1 Parameters Setting of Cloudsim

The experiments are implemented with 10 Datacenters with 50 VMs and 100-1000 tasks under the simulation platform. The length of the task is from 1000 MI (Million Instructions) to 20000 MI. The parameters setting of cloud simulator are shown in Table 1.

Table 1: Parameters setting of Cloudsim

Entity Type	Parameters	Value
Task (cloudlet)	Length of task	1000-20000
	Total number of task	100-1000
Virtual Machine	Total number of VMs	50
	MIPS	500-2000
	VM memory(RAM)	256-2048
	Bandwidth	500-1000
	cloudlet Scheduler	Space_shared and Time_shared
	Number of PEs requirement	1-4
Datacenter	Number of Datacenter	10
	Number of Host	2-6
	VmScheduler	Space_shared and Time_shared

5.2 ACO Parameters Evaluation and Setting

We implemented the ACO algorithm and investigated their relative strengths and weaknesses by experimentation. The parameters (α, β, p, t_{max}, m the number of ants and Q) considered here are those that affect directly or indirectly the computation of the algorithm. We tested several values for each parameter while all the others were held constant on 100 tasks. The default value of the parameters was α=1, β=1, ρ=0.5, Q=100, tmax=150 and m=8. In each experiment only one of the values was changed, The values tested were: α ∈ {0, 0.1, .2, .3, .4, .5}, β∈ {0, .5, 1.5, 2, 2.5, 3}, ρ∈ {0, 0.1, .2, .3, .4, .5} , Q∈{1, 100,500,1000}, t_{max}∈{50,75,100,150} and m∈{1,5,8,10,15,20}. We also use the time in the Cloudsim to record the makespan. The ACO performance for different values of parameters (α, β, p, tmax, m the number of ants and Q) has been evaluated. The ACO performance for different values of parameters (α, β,p ,tmax ,m the number of ants and Q) are shown from Fig.6 to Fig.11. It can be seen that the best value of α is .3, the best value of β is 1, the best value of ρ is .4, the best value of Q is 100, the best value of t_{max} is 150 and the best values of m is 10. In the following experiments we select the best value for α, β, ρ, Q and m parameters but the value100 is selected for the t_{max} parameter to reduce the overhead of the ACO algorithm.

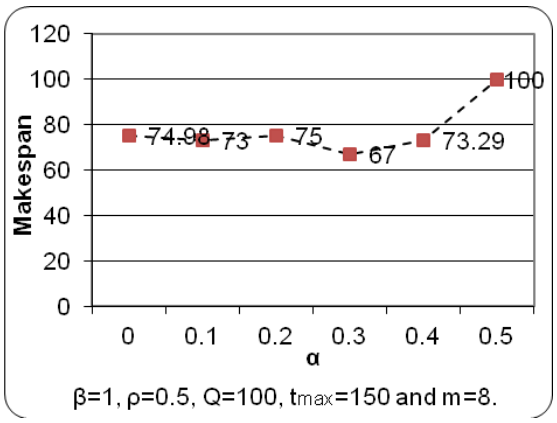

Fig. 6: ACO Performance for Different Values of Alpha

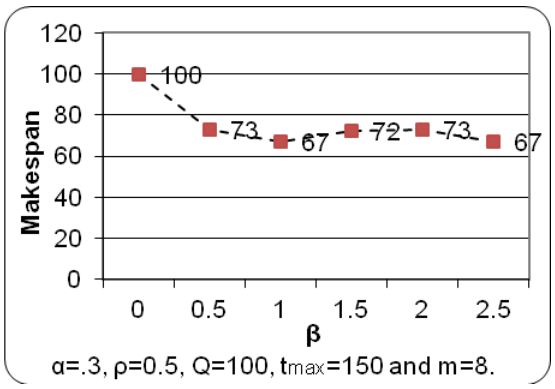

Fig. 7: ACO Performance for Different Values of Beta

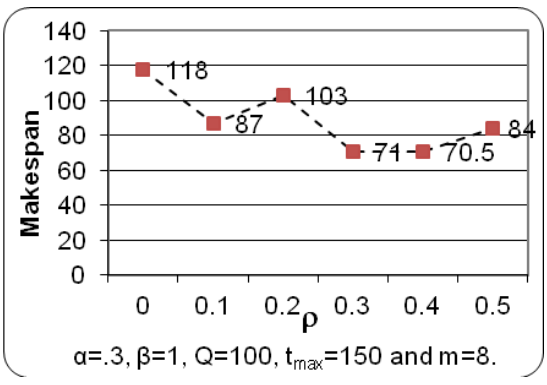

Fig. 8: ACO Performance for Different Values of Rho

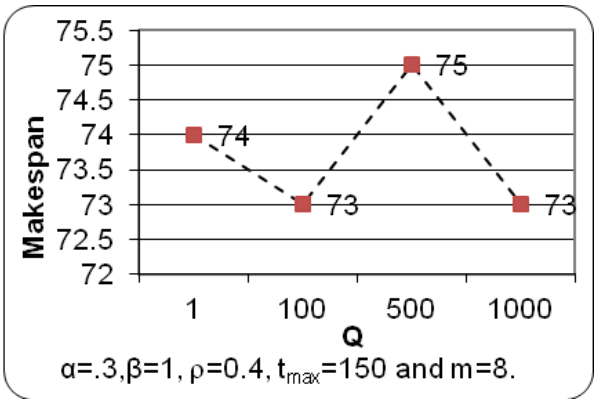

Fig. 9: ACO Performance for Different Values of Q

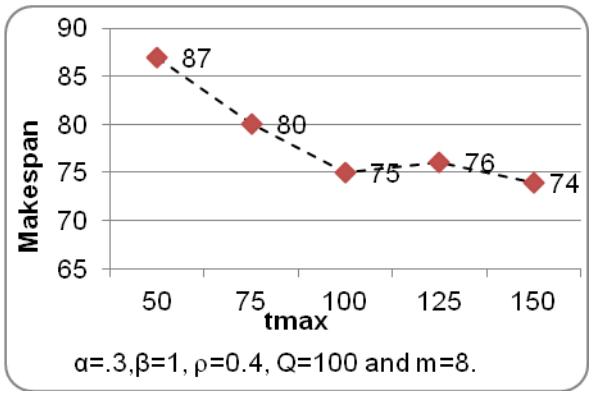

Fig. 10: ACO Performance for Different Values of Tmax

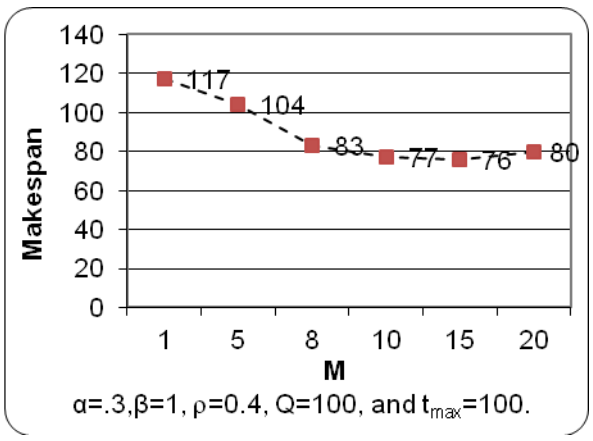

Fig. 11: ACO Performance for Different Values of Ants Number

Table 2 shows the selected best parameters of ACO. The MACO and MACOLB parameters are calculated using the proposed self-adaptive formulas. The m and t_{max} parameters are the same in all algorithms.

Table 2: Selected Parameters of ACO

Parameter	α	β	ρ	Q	m	tmax
Value	. 3	1	.4	100	10	100

5.3 Implementation Results of ACO, FCFS and RR

In the following experiments, we compared the average makespan with different tasks set. The average makespan of the ACO, MACO and MACOLB algorithms are shown in Fig.12. The average makespan of the First Come First Served (FCFS) and Round Robin (RR) and ACO algorithms are shown in Fig.13 [32]. It can be seen that, with the increase of the quantity task, ACO takes the time less than FCFS and RR algorithms and MACOLB takes the time less than MACO and ACO algorithms. This indicates that MACOLB algorithm is better than MACO, ACO, FCFS and RR algorithms.

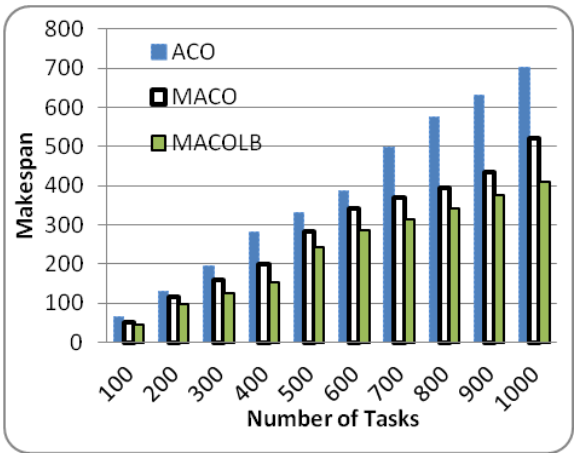

Fig. 12: Average Makespan of ACO, MACO and MACOLB

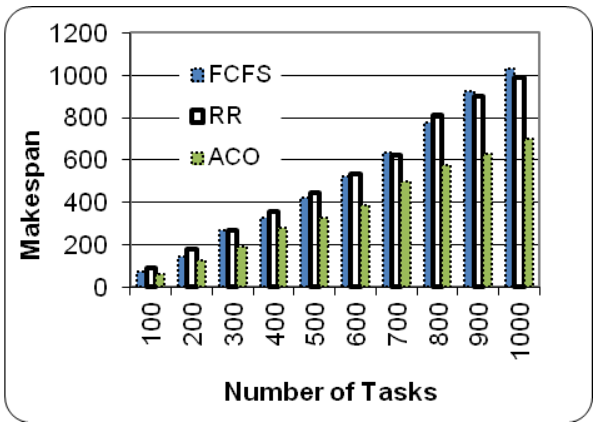

Fig. 13: Average Makespan of FCFS, RR and ACO

In statistics and probability theory, standard deviation (σ) shows how much variation or dispersion exists from the average (mean), or expected value. A low standard deviation indicates that the data points tend to be very close to the mean; high standard deviation indicates that the data points are spread out over a large range of values (solving stagnation problem). Since the standard deviation of never drops to zero, we are assured that the algorithm actively searches solutions which differ from the best-so-far found, which gives it the possibility of finding better ones. Fig.14 shows the evolution of the standard deviation of the ACO, MACO and MACOLB over 10 runs.

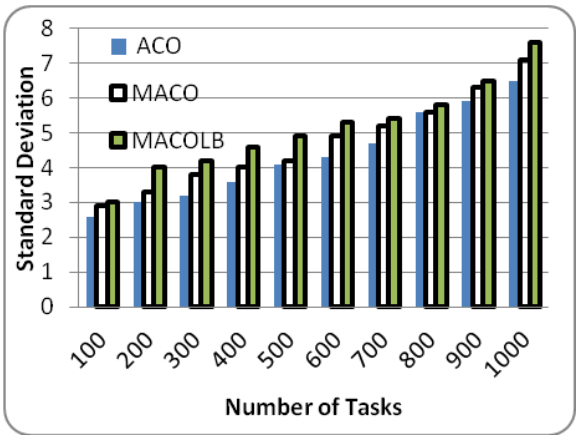

Fig. 14: Standard Deviation of ACO, MACO and MACOLB

The Degree of Imbalance (DI) measures the imbalance among VMs, which is computed by (20).

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (20)$$

Where, T_{max}, T_{min} and T_{avg} are the maximum, minimum and average execution time of VMs respectively. The average degree of imbalance of each algorithm with the number of tasks varying from 100 to 1000 is shown in Fig.15. It can be seen that the MACOLB can achieve better system load balance than ACO and MACO algorithms.

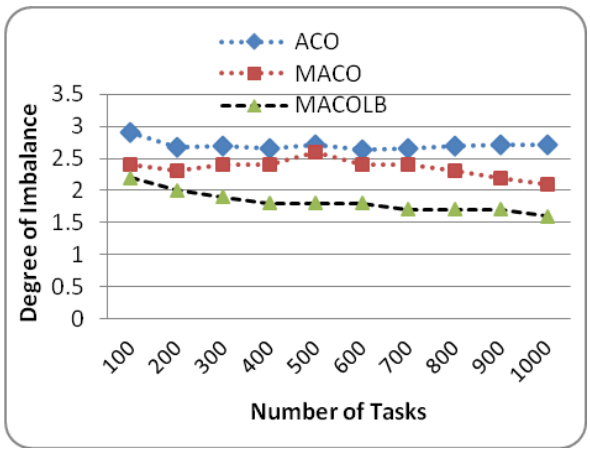

Fig. 15: Average Degree of Imbalance of ACO, MACO and MACOLB

VI. Conclusions and Future Work

In this paper MACOLB algorithm has been proposed to improve the cloud tasks scheduling for load balancing. MACOLB is used to find the optimal resource allocation for batch tasks in the dynamic cloud system and minimize the makespan of tasks on the entire system. The proposed algorithm uses the same self-adapting criteria for the MACO control parameters but has an added load balancing factor. MACOLB, MACO and ACO algorithms in applications with the number of tasks varying from 100 to 1000 evaluated using Cloudsim toolkit. Firstly the best values of parameters for ACO algorithm, experimentally determined. Then the makespan of the above algorithms evaluated. Simulation results demonstrate that MACOLB algorithm outperforms MACO and ACO algorithms. MACOLB algorithm can be extended with improvements to handle precedence between tasks and costs of resources. Also the comparison between our approach and other metaheuristics approaches will be performed.

References

[1] A. Weiss, "Computing in the Clouds" netWorker on Cloud computing: PC functions move onto the web, vol. 11, pp. 16-25, 2007.

[2] Gao Y., et al., "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," J. Comput. System Sci. vol.79 ,no. 8, pp. 1230–1242, 2013.

[3] F. Chang, J. Ren, and R. Viswanathan, "Optimal Resource Allocation in Clouds" in 2010 IEEE 3rd International Conference on Cloud Computing, pp.418-425, 2010.

[4] Qiyi, H., Tinglei, H., "An Optimistic Job Scheduling Strategy based on QoS for Cloud Computing" in 2010 IEEE International Conference on Intelligent Computing and Integrated Systems (ICISS), pp.673-675, 2010.

[5] F. Chang, J. Ren, and R. Viswanathan, "Optimal resource allocation for batch testing" in ICST, 2009 IEEE International Conference on Software Testing Verification and Validation, pp.91-100, 2009.

[6] Rubing Duan, Radu Prodan and Xiaorong Li, "A sequential cooperative game theoretic approach to scheduling multiple large-scale applications in grids" J. Comput. System Sci. vol.30 , pp. 27–43, 2014.

[7] M. Dorigo, C. Blum, "Ant colony optimization theory: A survey" in Theoretical Computer Science 344 (2–3) (2005), pp.243–278, 2005.

[8] M. Dorigo, M. Birattari, T. Stutzel, "Ant colony optimization", in IEEE Computational Intelligence Magazine, pp.28-39, 2006.

[9] Paul, M., Sanyal, G., "Survey and analysis of optimal scheduling strategies in cloud environment", IEEE International Conference on Information and Communication Technologies (WICT), pp. 789 – 792, 2012

[10] Jeyarani, R., Ram, R. Vasanth, Nagaveni, N., "Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment", IEEE International Conference on Cloud and Grid Computing (CCGrid), PP. 585 – 586, 2010

[11] Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing", IEEE International Conference on Parallel and Distributed Processing with Applications, PP. 629 - 634, 2009

[12] G. L. Nemhauser and A. L. Wolsey. "Integer and Combinatorial Optimization" John Wiley & Sons, New York, 1988.

[13] C. R. Reeves, editor. "Modern Heuristic Techniques for Combinatorial Problems" Blackwell Scientific Publishing, Oxford, England, 1993.

[14] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.

[15] T. Stutzle,MAX-MINAnt Systemfor Quadratic Assignment Problems Technical Report AIDA-97-04, Intellectics Group, Department of Compute Science, Darmstadt University of Technology, Germany, July 1997.

[16] B. Bullnheimer, R.F. Hartl, C. Strauss, A new rank-based version of the ant system: A computational study, Central European Journal for Operations Research and Economics 7 (1) (1999) 25–38.

[17] E.D. Taillard, L.M. Gambardella, Adaptive memories for the quadratic assignment problem, Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

[18] M. Dorigo, V.Maniezzo, A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B 26 (1) (1996) 29–41.

[19] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System Concepts, 7th edition, JohnWiley & Sons, 2005.

[20] D. Saha, D. Menasce, S. Porto, Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures, Journal of Parallel and Distributed Computing 28 (1) (1995) 1–18.

[21] Boloor, K., Chirkova, R., Salo, T., Viniotis, Y., "Heuristic-Based Request Scheduling Subject to a Percentile Response Time SLA in a Distributed Cloud". IEEE International Conference on Global Telecommunications Conference (GLOBECOM), PP.1-6 , 2010

[22] Laiping Zhao, Yizhi Ren, Yang Xiang, Sakurai, K., "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems", IEEE International Conference on High Performance Computing and Communications (HPCC), PP. 434 – 441, 2010

[23] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, Jian Xie, Jicheng Hu, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing", IEEE International Conference on Wireless Communications, Networking and Mobile Computing, PP. 1 – 4, 2009

[24] Kai Zhu, Huaguang Song, Lijing Liu, Jinzhu Gao, Guojian Cheng, "Hybrid Genetic Algorithm for Cloud Computing Applications", IEEE International Conference on Asia-Pacific Services Computing Conference (APSCC), PP. 182 – 187, 2011

[25] Kessaci, Y., Melab, N., Talbi, E.-G., "A pareto-based GA for scheduling HPC applications on distributed cloud infrastructures", IEEE International Conference on High Performance Computing and Simulation (HPCS), PP. 456 – 462,2011

[26] Ching-Hsien Hsu, Tai-Lung Chen, "Adaptive Scheduling Based on Quality of Service in Heterogeneous Environments", IEEE International Conference on Multimedia and Ubiquitous Engineering (MUE), PP. 1 - 6, 2010

[27] Manpreet Singh, "GRAAA: Grid Resource Allocation Based on Ant Algorithm" in 2010 Academy Publisher DOI: 10.4304/jait.1.3.133-135, 2010.

[28] Ku Ruhana Ku-Mahamud, Husna Jamal Abdul Nasir, "Ant Colony Algorithm for Job Scheduling in Grid Computing" in ams, 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, pp.40-45, 2010.

[29] Lorpunmanee, S., Sap, M.N, Abdul Hanan Abdullah, A.H., "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment" in Proceedings of World Academy of Science, English and Technology Volume 23 august 2007, ISSN 1307-6884, 2007.

[30] Buyya, R., Ranjan, R., Calheiros, R.N., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities" in Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, 2009 .

[31] Ghalem, B., Fatima Zohra, T., and Wieme, Z. "Approaches to Improve the Resources Management in the Simulator CloudSim" in ICICA 2010, LNCS 6377, pp. 189–196, 2010.

[32] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. keshk and Fawzy A. Torkey, " Cloud Task Scheduling Based on Ant Colony Optimization" in International Conference on Computer Engineering & Systems ICCES, 2013.

[33] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. keshk and Fawzy A. Torkey, " An Ant Algorithm for

Cloud Task Scheduling" in International Workshop on Cloud Computing and Information Security CCIS, 2013.

[34] Eric Bonabeau, Marco Dorigo, Guy Theraulaz, "Swarm intelligence: from natural to artificial intelligence",ISBN 0-19-513158-4, Published by Oxford University Press, Inc.198 Madison Avenue, New York,1999.

[35] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, INFORMS J. Comput. 11 (4) pp. 358–369, 1999.

Authors' Profiles

Arabi E. keshk received the B.Sc. in Electronic Engineering and M.Sc. in Computer Science and Engineering from Menoufia University, Faculty of Electronic Engineering in 1987 and 1995, respectively and received his PhD in Electronic Engineering from Osaka University, Japan in 2001. His research interest includes software testing, software engineering, distributed system, database, data mining, and bioinformatics.

Ashraf B. El-Sisi received the B.Sc. and M.Sc. in Electronic Engineering and Computer Science Engineering from Menoufia University, Faculty of Electronic in 1989 and 1995, respectively and received his PhD in Computer Engineering & Control from Zagazig University, Faculty of Engineering in 2001. His research interest includes cloud computing, privacy preserving data mining, and Intelligent systems.

Medhat A. Tawfeek received the B.Sc. and M.Sc. in computers and information from Menofia University, Faculty of computers and information in 2005 and 2010, respectively. Currently hold PhD student in Faculty of computers and information Menoufia University. His research interest includes cloud computing, smart card security, intelligent systems, distributed system, fault tolerance.

How to cite this paper: Arabi E. keshk, Ashraf B. El-Sisi, Medhat A. Tawfeek,"Cloud Task Scheduling for Load Balancing based on Intelligent Strategy", International Journal of Intelligent Systems and Applications(IJISA), vol.6, no.5, pp.25-36, 2014. DOI: 10.5815/ijisa.2014.05.02

