IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 04, Issue 03 (March. 2014), ||V6|| PP 07-17

Cloud Computing Online Scheduling

Arabi E. Keshk
Department of Computer Science, Faculty of Computers and Information, Menoufia University, Egypt
arabikeshk@yahoo.com

Abstract: - Cloud computing has gained a lot of attention to be used as a computing model for a variety of application domains. Task scheduling is the fundamental issue in this environment. To utilize cloud efficiently, a good task scheduling algorithm is needed to assign tasks to resources in cloud. Cloud task can be divided into two categories such as on-line mode service and the batch mode service. In this paper, online cloud task scheduling based on virtual machine adaptive fault tolerance and load balancing using ant colony algorithm is proposed. The main contribution of this work is that load balancing factor is added and the system tolerates the faults by tacking the decision on the basis of reliability of the virtual machines in scheduling process. The proposed scheduling strategy was simulated using the Cloudsim toolkit package. Experimental results show that the proposed algorithm achieved the better load balance than Join-shortest-queue (JSQ) and Modified Ant Colony Optimization (MACO) algorithms.

Keywords: - Cloud computing; task scheduling, makespan, ant colony optimization, fault tolerance, reliability, load balancing

I. INTRODUCTION

Technology such as cloud computing, has aimed at allowing access to large amounts of computing power in a fully virtualized manner, by aggregating resources and offering a single system view [1]. Clouds are a large pool of easily usable and accessible virtualized resources such as hardware, development platforms and/or services. These resources can be dynamically reconfigured to adjust to a variable load (scale) and allowing for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the infrastructure provider by means of customized service level agreements [2]. It is the responsibility of the cloud service provider to manage its resources in an efficient way to make the needed resources available on demand to the cloud users [3]. A key challenge that face cloud providers when building a cloud infrastructure is managing physical and virtual resources. The consumers are assured that the cloud infrastructure is very robust and will always be available at any time [4]. An efficient algorithm for task scheduling in cloud environment is needed with the goal of putting unused resource such as virtual machines to work, distributing the load about them and increasing the faults toleration of clouds requires multiple customers with disparate requirements to be served. Using cloud infrastructure for different enterprises increases the chances of errors. As the cloud virtual machines are far from the transceiver (job submitting node) [5]. Many systems are also safety critical systems, so they require a higher level of fault tolerance. Safety critical systems require working properly to avoid failure, which can cause casualties [6]. So there is an increased need to tolerate the fault for such type of systems to be used with cloud infrastructure [7, 8]. Cloud task can be divided into two categories such as on-line mode service and the batch mode service. In online mode, whenever a request arrives, it is immediately allocated to the first free resource allocator. The arrival order of the request in cloud is important in this method. Here, each service request is considered only once for matching and scheduling. In batch mode, the requests are collected; the scheduler considers the approximate execution time for each task and use a heuristic approach to make better decision [9]. In this paper, ant algorithm for online cloud task scheduling based on load balancing and reliability is proposed. The objective from this work is that, finding the best resource allocation for each task in the dynamic cloud by balancing the load of the system and adapting reliability of the virtual machines. Finally, the proposed algorithm is compared with Join-shortest-queue (JSQ), Modified Ant Colony Optimization (MACO) algorithms in the experiments using the Cloudsim toolkit package. According to the experimental results, the proposed algorithm is capable of achieving system load balance better than other job scheduling algorithms. The rest of the paper is organized as follows: Section 2 presents cloud computing environment and task scheduling. Section 3 scans the fault tolerance on cloud computing. Ant algorithms and some of the important related work in this direction are shown in section 4. In section 5, cloud task scheduling based on load balancing and reliability and the details about the proposed algorithm are presented. The implementation and simulation results are seen in section 6. Finally, Section 7 concludes this paper.

II. CLOUD COMPUTING AND TASK SCHEDULING

Cloud Computing is nowadays the most used infrastructures to execute scientific applications [1]. Many practitioners in the commercial and academic spheres have attempted to define exactly what "cloud computing" is and what unique characteristics it presents. The cloud computing is 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 [2]. Cloud computing is based on the concepts of distributed computing, grid computing, utility computing and virtualization. A cloud computing platform dynamically provisions, configures, and reconfigures servers as needed. Servers in the cloud can be physical machines or virtual machines. It provides the utility services based on the pay-as-you go model. Users can host different kinds of applications on the cloud ranging from simple web applications to scientific workloads. These applications are delivered as services over the internet [6]. The consumers only need to pay for using the services just like they do in case of other day to day utility services such as water, gas, electricity etc., which are on pay per user basis and hence cloud computing resources are also on pay per user basis. Cloud computing is now being used in many applications that are beyond distribution and sharing of resources. The distributed resources are useful only if the cloud resources are scheduled. Using optimal scheduler's results in high performance cloud computing, where as poor schedulers produce contrast results [10].

Now, the scheduling in cloud is a big topic in cloud environment for new algorithm model. Scheduling is a challenging job in cloud because the capability and availability of resources vary dynamically. The goal of job scheduling is to properly dispatch parallel jobs to slave node machines according to scheduling policy under meeting certain performance indexes to shorten total execution time and lower computing cost and improve system efficiency. The demand for scheduling is to achieve high performance computing. It is very difficult to find an optimal resource allocation for specific job. The scheduling problem is a NP-hard problem [11]. ACO is one effective method to deal with NP-hard problems, which has stronger robust, distributed capability, parallel and scalability [8]. Cloud has an extra layer called virtualization layer. This layer acts as a creation, execution, management, and hosting environment for application services [10]. The virtual machines in cloud are contextually isolated but still they need to share computing resources, processing cores, system bus etc. [2]. 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. The choice of virtual machine is meaning that you can select a configuration of CPU, memory, storage and bandwidth that is optimal for an application. By means of virtualization technologies, Cloud Computing offers to end-users a variety of services covering the entire computing stack, from the hardware to the application level by charging them on a pay per use basis, i.e., cycles consumed and bytes transferred during computations. Within a cloud, services that represent computing resources, platforms or applications are provide across organizations. This makes the spectrum of options available to scientists wide enough to cover any specific need from their research. Another important feature is the ability to scale up and down the computing infrastructure according to the application requirements and the user's budgets [12].

Cloud scheduling is categorized at user level and system level [3]. user level scheduling deals with problems raised by service provision between providers and customers [13]. The system level scheduling handles resource management [2, 12]. 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 [14]. Scheduling based genetic algorithm is proposed in [15, 16]. This algorithms optimize 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 [17]. Min-min is heuristic used for batch mode scheduling. This enables batch heuristics to know about the actual execution times of a larger number of tasks. Join-shortest-queue (JSQ) and Join-Idle-Queue (JIQ) assign jobs to processors to reduce average queue length of jobs at each processor [18]. Task scheduling in the cloud should decide optimal number of systems required in the cloud and allocate the tasks in an efficient way so that the total cost is minimized.

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, the proposed online cloud task scheduling algorithm has been presented for allocation of incoming jobs to virtual machines considering in account makespan, load balancing and fault tolerance to help in utilizing the available resources optimally, minimize the resource consumption and achieve a high user satisfaction.

III. CLOUD COMPUTING FAULT TOLERANCE

In cloud, the latency of virtual machines is unknown. Even if one determines the latency, it can change over the period of time [6]. Node is a virtual and computation can be migrated from one virtual machine instance to another [19]. The user of real time cloud applications has lose control over the nodes and does not know where his application is going to be processed. But on the brighter side, cloud has a facility to scale up dynamically. So the faulty node can be removed and new node can be added on demand. These characteristics are different from the existing traditional distributed real time systems [20]. A model for virtual infrastructure performance and fault tolerance is presented in [21]. A new fault tolerant scheduling algorithm 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. Some pragmatic requirements for highly reliable systems, highlighted significance and various issues of reliability in different computing environment such as Cloud Computing, Grid Computing, and Service Oriented Architecture are describe in [7]. The Adaptive Fault Tolerance in Real-time Cloud computing (AFTRC) model based upon adaptive reliability assessment of virtual machines in cloud environment and fault tolerance of real time applications running on those VMs was proposed in [6]. AFTRC model tolerates the faults on the basis of reliability of each virtual machine. A virtual machine is selected for computation on the basis of its reliability and can be removed if does not perform well for applications. In AFTRC model, each VM takes the input, executes the application algorithm and produces result. Reliability assessor (RA) module assesses the reliability for each virtual machine is the main core module in AFTRC model. In the beginning the reliability of each virtual machine is 100%. If a processing node manages to produce a correct result within the time limit, its reliability increases. And if the processing node fails to produce the correct result or result within time, its reliability decreases. The reliability assessment algorithm is more convergent towards failure conditions. It means that decreasing in reliability is more than increasing [6]. In this paper, the proposed algorithm depends on the basic idea (adaptive reliability assessment of virtual machines) of AFTRC model in online cloud task scheduling using ant algorithm.

IV. RELATED WORK AND ANT COLONY ALGORITHM

The basic idea of ant algorithms 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 [8]. 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. The total number of ants (m), assumed constant over time, is an important parameter: too many ants would quickly reinforce suboptimal trails and lead to early convergence to bad solutions, whereas too few ants would not produce the expected effects of cooperation because of the process of pheromone decay [10]. 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 which mean that all individuals found the same solution exactly, cannot further search for the solution space and make the algorithm converge to local optimal solution [8]. There are many different kinds of ACO algorithm, i.e., Ant Colony System (ACS), Max-Min Ant System (MMAS), Rank-based Ant System (RAS), Fast Ant System (FANT) and Elitist Ant System (EAS) [23]. 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 scheduling in cloud environment based ACO algorithms are proposed in [24, 25]. In these methods, the requests are collected; the scheduler considers the approximate execution time for each task and use heuristic approach to possibly make better decision. 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. Modified Ant Colony Optimization for Load Balancing (MACOLB) for cloud task scheduling is proposed in [22]. MACOLB algorithm used to find the optimal resource allocation for tasks in the dynamic cloud system, minimize the makespan of tasks on the system and increase the performance by balancing the load of the system. 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]. Moreover, in [9] an ACO scheduler was introduced to address job scheduling within a Cloud. It is used for online mode. The proposed method is aimed to maximize scheduling throughput to handle all the diversified job requests according to different resources available in a cloud, and minimize the makespan of jobs.

V. THE PROPOSED CLOUD COMPUTING TASK SCHEDULING

This section propose a cloud task scheduling based on load balancing and reliability to improve the performance of the scheduling problems in cloud computing. A task allocation framework comprises user, resource broker, virtual machines and Cloud Information Services (CIS). It adopts ant colony as major allocation strategy. The interaction among various entities of system model is as follow:

Step 1: Resource registration to CIS takes place.

Step2: User submit task with complete specification to broker.

Step3: Broker places all submitted tasks in a task set.

Step4: Broker queries CIS regarding resources.

Step5: CIS returns the attribute of resources such as number of virtual machines, number of processing elements (PE), MIPS (Million Instruction Per second) rating of each PE, allocation policy.

Step6: Broker send query to registered resources for their availability status.

Step7: Broker gets the status information and makes it available to allocation process.

Step8: By deploying proposed ant algorithm, broker select a virtual machine for next task assignment and dispatch the task to selected virtual machine.

Step9: After task execution, results are received and are returned to user.

In the reset of this section, the general adaptive ant algorithm is explained and how it is used to solve the job scheduling problem in the cloud environment. Pheromone value on a path in the ant system is a weight for a Virtual Machine (VM) in the cloud system. A virtual machine with a larger weight value means that it has a better computing power. The increase or decrease of pheromone depends on tasks status at resources.

A. The proposed Ant Algorithm

The pheromone (weight) of each VM is stored in the broker and the scheduler in broker uses it as the parameters for resource allocation strategy based on ant colony algorithm.

When a new VM J is created, it will initialize its pheromone as in (1)

$$\tau_j(0) = Pe\_num_j * Pe\_mips_j + VM\_bw_j \quad (1)$$

Where $\tau_j(0)$ be the trail intensity on path from the scheduling center to the corresponding VMj at time 0, Pe_numj is the number of VMj processors, Pe_mipsj is the MIPS of each processor of VMj and VM_bwj is the communication bandwidth ability of the VMj.

Whenever a new task is assigned to VM j or task is returned from VM j then the pheromone of VM is updated. The trail intensity at time t is updated as in (2).

$$\tau_j(t) = (1-\rho)\tau_j(t) + \Delta\tau_j(t) \quad (2)$$

Where ρ is the trail decay, $0 < \rho < 1$ and $\Delta\tau_j(t)$ is the variety of quantity of the trail substance laid on the path from the scheduling center (broker) to VM j. $\Delta\tau_j(t)$ is computed as follow:

- When a task is assigned to VM_j, its pheromone is reduced i.e. $\Delta\tau_j(t) = -L$

Where, L is relevant to computation workload and communication quantity of the job.

- When job successfully completed and VM_j is released or job failed and returned from VM_j, its pheromone is increased i.e. $\Delta\tau_j(t) = L$

The definition of transition possibility rule for the next job to VM_j as in (3).

$$p_j(t) = \arg\ max_{s \in allowed_k}\left\{\left(\tau_s(t)\right)^{\alpha} * (\eta_s)^{(1-\alpha)}\right\} \quad (3)$$

Where,

- $\tau_s(t)$ denotes the current pheromone of VM
- $allowed_k$ expresses the allowed virtual machines for ant k (task) i.e. $s \in$ available resources
- η_s is called visibility for the VM_s, namely the innate performance quantity of the VM_s (equals the initial pheromone of VM s) i.e. $\eta_s = \tau_s(0)$.
- α is the control parameter used to map the relative importance of quantity of pheromone and the visibility of each movement.

B. Adding load balancing factor

The transition possibility rule is trade-off between visibility that means the power performance of the VMs and trail intensity that means, if there is a lot of pheromone concentration on path j then it is high desirable. Once some VM load heavy, it comes into being a bottleneck in the cloud and influences to completing of jobs. Therefore the load balancing factor is proposed to improve the load balancing capability. By taking in the account load balancing factor of the VMj, which is related to the job finishing rate in the VMj. So the transition possibility rule will be changed to (4):

$$p_j(t) = arg\ \ max_{s \in allowe\, d_k}\left\{\left(\tau_s(t)\right)^{\alpha} * (\eta_s)^{(1-\alpha)} * LB_s\right\} \quad (4)$$

Where,

$LB_s = \frac{1}{EFT_s}$ and $EFT_s = \frac{TOT}{Pe\_num_s * \mathrm{Pe\_mips}_s}$

Such, EFT_s is the expected finishing time for VMs and TOT is total length of the tasks that have been submitted to VMs. LB is the load balancing factor means, the more loads on virtual machine, the less chance to be selected for next task, contrarily, the less load on virtual machine, the more chance to be selected for next task.

C. Adding the Reliability factor

Up till now there is no different in steps of pheromone updating if job successfully completed or failed (returned from VM) and VM is released (ignoring the faults and the decision making not based on the reliability of the VM). So, the following modification is done. This modification is depending on the AFTRC model [6]. In the first, we assume that, all virtual machines have the same degree of reliability.

- When job successfully completed on VM_j and VM_j is released, its pheromone is updated as in (5).

$$\tau_j(t) = (1-\rho)\tau_j(t) + (L * RFS) \quad (5)$$

Where, RFS is a reliability factor for success which increases the pheromone (increase reliability of the node). It gives a great chance for a node with highest reliability to be selected.

- When job failed and returned from VM_j, its pheromone is updated as in (6).

$$\tau_j(t) = (1-\rho)\tau_j(t) + (L * RFF) \quad (6)$$

Where, RFF is a reliability factor for failure which decreases the pheromone (decrease reliability of the node). It gives a low chance for a node with lowest reliability to be

selected. The final shape of procedure (pseudo code) of proposed ant algorithm is shown in Fig. 1.

```
Input: Online Tasks
Output: The best solution for each tasks allocation on VMs
Begin
 Initialize parameters
 Compute Initial value of pheromone for each VM τj(0).
 While (the pool of online tasks not empty) do
   Begin
     Select the first arrived task.
      Determine the next VM for the task assignment using transition rule (4).
     Remove   task t from the pool of tasks.
      Update pheromone intensity of corresponding VM using (2).
     If (task completion successfully) then
        Update pheromone intensity of corresponding VM     using (5).
      End IF
      If (failure occurs) then
        Update pheromone intensity of corresponding VM using (6).
       Resubmit task into the pool of online tasks
      End IF
    End While
End
```

Fig. 1. The pseudo code of proposed ant algorithm

VI. SIMULATION RESULTS

The scheduling algorithms are compared in the experiments that include Join-shortest-queue (JSQ) [18], Modified Ant Colony Optimization (MACO) [9] and proposed ant algorithm. These algorithms were developed and simulated using Cloudsim. The Experiments are done using various ranges of task length and resource capabilities. Cloudsim can be used to model data centers, host, service brokers, scheduling and allocation policies of a large scaled cloud platform. Hence, the researcher has used Cloudsim for experimenting in simulated cloud environment [26]. The experiments are implemented with 10 Datacenters with 25 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 (ρ, α, RFS and RFF) considered here are those that affect directly or indirectly the computation of the algorithm.

Several values were tested for each parameter while all the others were held constant. The best value of α is .4, the best value of ρ is .01, the best value of RFS is 1.1 and the best value of RFF is .8. The simulation was run at five levels of workloads 100 tasks, 250 tasks, 500 tasks, 750 tasks and 1000 tasks. We built a simple fault injection tool to be used to change the status of returned task to failure state. In the case of failure, the broker will receive the error signal from VM and resubmit the failure tasks again. The makespan of the JSQ, MACO in millisecond and the proposed algorithms without fault injection is shown in Fig. 2 and the makespan with fault injection is shown in Fig. 3.

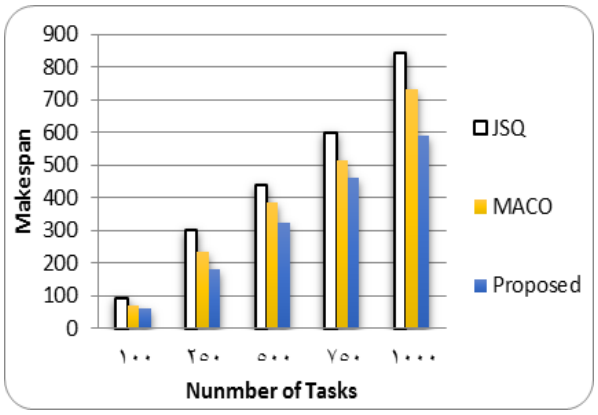

Fig. 2. The Makespan for each algorithm without Fault Injection

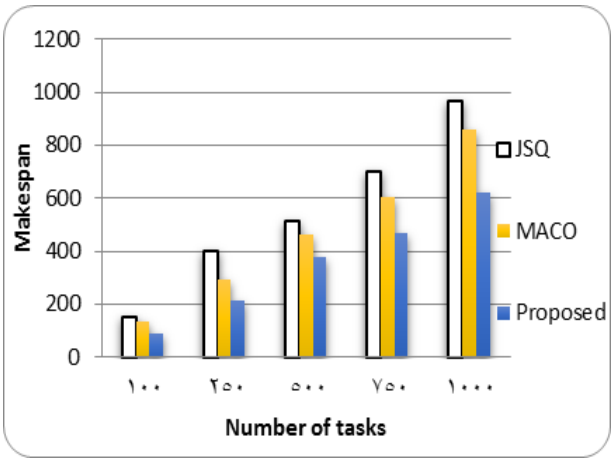

Fig. 3. The Makespan for each algorithm with Fault Injection

Makespan in Fig. 2 and Fig. 3 is the finishing time of the last job in the system without the time of scheduling (overhead). The total spent time (overhead added to the makespan) of JSQ, MACO and proposed algorithms without fault injection are shown in Fig. 4 and the overhead added to the makespan with fault injection are shown in Fig. 5. It is shown from Fig. 2 – Fig. 5 that, the proposed algorithm take less time to execute in each size of jobs than other methods because it let the VM which has good computing power, light load and high reliability to execute more jobs. That is the reason why proposed algorithm takes less time to execute jobs. The MACO will assign more jobs to the bad resources which have higher pheromone and this will increase the total execution time. JSQ may have bad performances because it depends only on the status of queue.

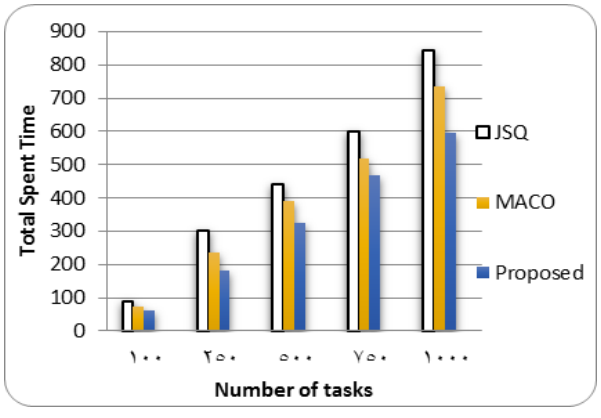

Fig. 4. Total spent time for each algorithm without Fault Injection

The degree of imbalance (DI) measures the imbalance among VMs. DI can be measured by different methods. The first method measures the difference between maximum and minimum Execution time of VMs which is defined as in (7).

$$DI_1 = AT_{max} - AT_{min} \quad (7)$$

Where, ATmax and ATmin are the actual maximum and minimum finishing execution time of VMs. The small value of DI1 tells that the load of the system is more balanced.

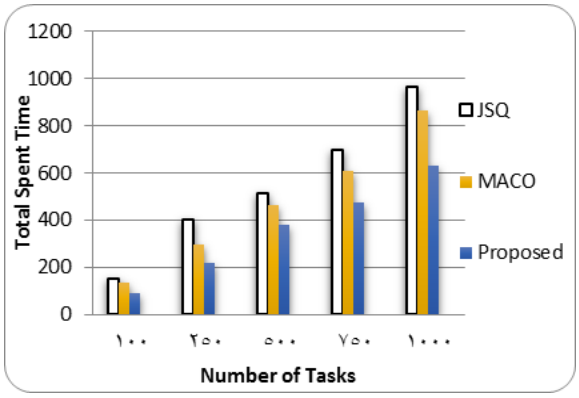

Fig. 5. Total spent time for each algorithm with Fault Injection

The DI_1 of the JSQ, MACO and proposed algorithms without fault injection and with fault injection are shown in Fig. 6 and Fig. 7 respectively. It can be seen from the Fig. 6 and Fig. 7 that the proposed algorithm can achieve good system load balance.

The second method measures the degree of imbalance, as in (8) and (9).

$$T_i = \frac{TL\_Tasks}{Pe\_num_i * Pe\_mips_i} \quad (8)$$

Where, TL_Tasks is the total length of tasks which are submitted to the VM_i.

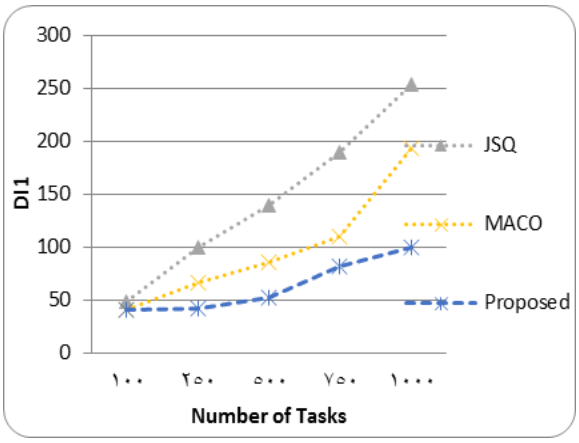

Fig. 6. DI_1 for each algorithm Without Fault Injection

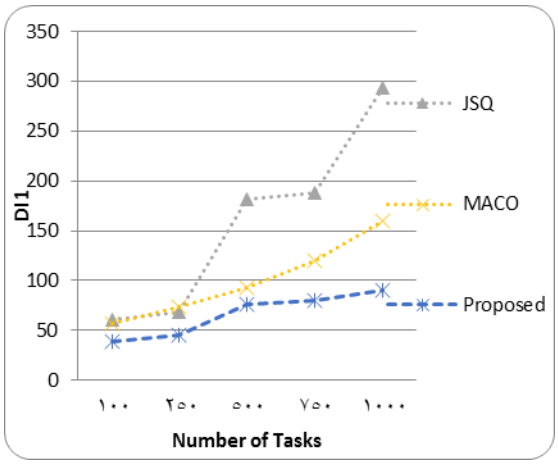

Fig. 7. DI_1 for each algorithm with Fault Injection

$$DI_2 = \frac{T_{max} - T_{min}}{T_{avg}} \qquad (9)$$

Where, Tmax, Tmin and Tavg are the maximum, minimum and average Ti respectively among all VMs. The small value of DI2 tells that the load of the system is more balanced. The degree of imbalance (DI2) for each algorithm without Fault Injection and with Fault Injection is shown in Fig. 8 and Fig.9 respectively. It can be seen from the Fig. 8 and Fig. 9 that the proposed algorithm can achieve good system load balance.

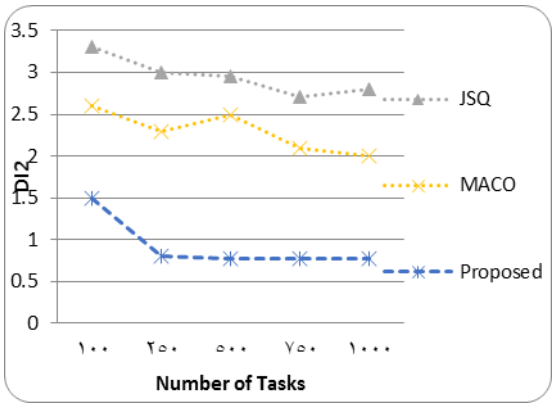

Fig. 8. DI_2 for each algorithm without Fault Injection

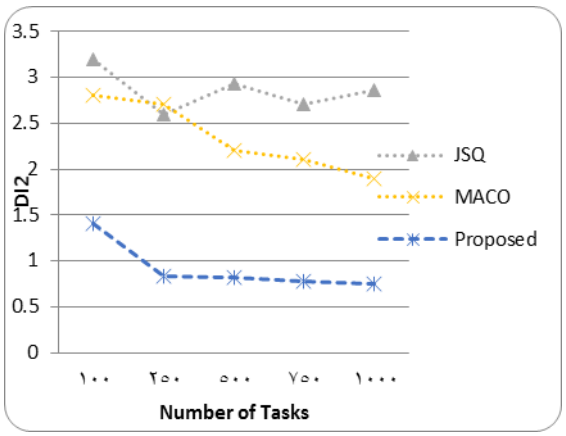

Fig. 9. DI_2 for each algorithm with Fault Injection

By the experimental results, it can easily find out that, the proposed algorithm take less time to execute jobs. The proposed algorithm is based upon adaptive reliability assessment of virtual machines and load balancing in cloud environment for applications running on those VMs. The proposed algorithm has a dynamic behavior of reliability and load configuration. It means that the proposed algorithm keeps the system load more balanced and have better performances.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, ant algorithm for online cloud task scheduling based on load balancing and reliability has been proposed. The proposed algorithm chooses suitable resources to execute jobs according to resources status such as load and reliability and the size of given job in the cloud environment. The proposed scheme is a good option to be used as a fault tolerance mechanism and balancing the load of cloud computing virtual machines. It has incorporated the concept of fault tolerance on the basis of VM algorithm reliability. Decision mechanism of the proposed algorithm shows convergence towards the virtual machine which has highest reliability. The proposed algorithm has a guided load balancing factor to balance the load on available virtual machines. The results of the experiments are also presented and the strengths of the algorithm are investigated. The simulation results demonstrate that the proposed algorithm increases the performance in terms of reduction in total execution time and the degree of imbalance. In the future, the replication of tasks will be considered.

VIII. ACKNOWLEDGMENT

I would like to thank my student Mr. M. Tawfiq for helping me and doing the simulation work for this paper.

REFERENCES

[1] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers", in: Proceedings of the IEEE International Conference on Services Computing, pp. 514–521, 2010

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

[3] Gao K., Wang Q., Xi L., "Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment ", International Arab Journal of Information Technology (IAJIT), Vol 11, No.3, May 2014.

[4] Ijaz S., Munir E., Anwar W., Nasir W., "Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment", International Arab Journal of Information Technology (IAJIT),Vol 10, No. 5, Septemper 2013.

[5] X. Kong, J. Huang, C. Lin, P. D. Ungsunan, "Performance, Fault-tolerance and Scalability Analysis of Virtual Infrastructure Management System", IEEE International Symposium on Parallel and Distributed Processing with Applications, Chengdu, China, August 9-12, 2009

[6] Sheheryar Malik, Fabrice Huet,"Adaptive Fault Tolerance in Real Time Cloud Computing", Proceedings of the IEEE World Congress on Services, pp. 280-287,2011

[7] Waseem Ahmed, YongWeiWu, "A survey on reliability in distributed systems", Journal of Computer and System Sciences, vol. 79, pp. 1243–1255, 2013.

[8] Shagufta khan, Niresh Sharma, " Ant Colony Optimization for Effective Load Balancing In Cloud Computing ", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 3, pp. 77-82, 2013.

[9] Banerjee S, Mukherjee I, Mahanti P. "Cloud computing initiative using modified ant colony framework" In: World academy of science, engineering and technology, p. 221–224, 2009.

[10] Elina Pacini, Cristian Mateos, Carlos García Garino, "Distributed job scheduling based on Swarm Intelligence: A survey", Computers and Electrical Engineering,http://dx.doi.org/10.1016/j.compeleceng.2013.11.023, 2013.
[11] 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
[12] 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
[13] 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.
[14] 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
[15] 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
[16] 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
[17] 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
[18] V. Gupta, M. Harchol-Balter, K. Sigman, and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms", Performance Evaluation, Vol. 64, pp. 1062–1081, 2007.
[19] X. Kong, J. Huang, C. Lin, "Comprehensive Analysis of Performance, Fault-tolerance and Scalability in Grid Resource Management System", International Conference on Grid and Cooperative Computing, Lanzhou, China, pp.27-29, 2009
[20] W. T. Tsai, Q. Shao, X. Sun, J. Elston, "Real Time Service Oriented Cloud Computing", School of Computing, Informatics and Decision System Engineering Arizona State University USA, pp. 473 – 478, 2010
[21] 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.
[22] Arabi E. keshk, Ashraf El-Sisi, Medhat A. Tawfeek, F. A. Torkey, "Intelligent Strategy of Task Scheduling in Cloud Computing for Load Balancing ", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),Vol. 3, pp.12-23,2013
[23] R.F. Tavares Neto, M. Godinho Filho, "Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research", Engineering Applications of Artificial Intelligence,vol. 26, pp. 150–161, 2013.
[24] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. keshk and Fawzy A. Torkey, " Cloud Task Scheduling Based on Ant Colony Optimization", on the 8thInternational Conference on Computer Engineering & SystemsICCES, ,pp.64-68,2013.
[25] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, Dan Wang, "Cloud Task scheduling based on Load Balancing Ant Colony Optimization", Chinagrid Conference (ChinaGrid), pp.3– 9, 2011
[26] 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.