Architecture for Fault Tolerance in Mobile Cloud Computing using Disease Resistance Approach

Dasari Naga Raju, Vankadara Saritha

SCOPE, VIT University, Vellore, Tamilnadu, India
raj2dasari@gmail.com, vsaritha@vit.ac.in

***Abstract*:** The mobile cloud computing (MCC) is one of the emerging fields in the distributed computing. MCC is an integration of both mobile computing and cloud computing. The limitations of the mobile devices are storage, battery and processing proficiency. These sensitive characteristics of mobile devices can be effectively handled with the introduction of cloud computing. The increasing functionality of the cloud and complexity of the applications causes resource failures in the cloud computing and it reduces the overall performance of the MCC environment. On the other hand, the existing approaches for resource scheduling in MCC proposed several architectures and they are only concentrated on the allocation of resources. The existing architectures are lack of fault tolerance mechanism to handle the faulty resources. To overcome the issues stated above, this paper proposes architecture for fault tolerance in MCC using Disease Resistance approach (DRFT). The main aim of the DRFT approach is to effectively handle the faulty VMs in the MCC. This DRFT approach utilizes the human disease resistance mechanism which is used as materials and methods in the proposed model. The DRFT is capable of identifying the faulty virtual machines and reschedules the tasks to the identified suitable virtual machines. This procedure ultimately leads to the minimization of makespan value and it improves the overall performance of the scheduling process. To validate the effectiveness of the proposed approach, a series of simulations has been carried out using CloudSim simulator. The performance of the proposed DRFT approach is compared with the Dynamic group based fault tolerance approach (DGFT-approach). The make span value of DRFT is reduced to 7% and the performance of DRFT is increased when compare to the DGFT approach. The experimental results show the effectiveness of the proposed approach.

***Keywords*:** MCC, Resource, Fault tolerance, Virtual Machine, Disease Resistance.

1. Introduction

The mobile devices are playing a prominent role in daily life. However, the limitations of mobile devices such as limited bandwidth, computation capacity and battery power restricts them to execute the larger applications. To overcome this issue, cloud computing is integrated with the mobile devices to utilize the infrastructure of the cloud, which will be called as mobile cloud computing (MCC) [28]. The MCC environment requests the services from the cloud via Infrastructure as a Service (IaaS).

The advantages of cloud computing such as high resource availability, low cost and minimum energy consumption will favour the mobile device for offloading of the tasks and increase computation speed. The task offloading is a major process in mobile environment where it has to partition the submitted job in to multiple tasks and each task has to be executed separately with in the mobile environment or outside the mobile environment. If the task is submitted outside the mobile environment, i.e., the cloud, then there is an issue called as resource scheduling.

In recent years, many researchers made a contribution for efficient resource allocation strategies in the cloud [7-9]. Abolfazli et al. [1] made a survey on research challenges in MCC. They made a comparison over different architectures of MCC for resource scheduling. Aceto et al. [2] proposed an optimal infinite scheduler which can manage the resource scheduling by improving the energy consumption of the mobile devices. They proposed cost and reward method for MCC by using the MobiCa language. Molina et al. [3] proposed a method to schedule the tasks form the mobile devices to the cloud. This method utilizes the wireless medium for handling the incoming tasks and to schedule them in to the cloud. In this paper, architecture for fault tolerance in MCC using disease resistance approach was proposed. The proposed algorithm is inspired by the bio approach called as disease resistance (DR) mechanism. This DR- approach has the ability to handle the fault resources in the MCC environment. The main aim of the DRFT approach is to find the faulty virtual machines and reschedule the tasks to the suitable virtual machines. This study also aims to reduce the makespan of the tasks and improve the overall performance of the scheduling process in MCC environment.

The rest of the paper is organized as follows. Section 2 explains about the related work regarding the scheduling of tasks in MCC. Section 3 explains about the problem statement. Section 4 deals with disease resistance approach. Section 5 discusses about the task model for mobile environment. Section 6 explains about the architecture for fault tolerance in MCC. Section 7 explains the results and performance evaluation of the DR-Approach. Finally, the conclusion is drawn in Section 8.

2. Related Work

There are a number of scheduling approaches to solve the relatively routine problems in cloud computing [7-9]. The algorithms have the capacity to solve the large problems and converting them into manageable optimization problems through linear programming [10], integer linear programming [12], and integer programming [11]. Due to the increase in functionality of the cloud services causes VM failures at the time of service delivery [24]. This type of failure causes performance degradation, task failures, data loss and loss in the revenue of the organisation [14]. While dealing with the workflow of the tasks, the task failures causes critical problems. This is due to the execution of the workflow depends on the predecessor task, so the input of the task was depends on the output of some other task. In the recent years,

many researchers concentrated on this area [15, 16], but very few are concentrated on the fault tolerance mechanism in the MCC environment.

There are two major fault tolerance strategies are popular under the cloud services are resource rescheduling and data replication. The data replication process is mostly preferable at the time of scheduling of VMs and the resource rescheduling process is implemented at the time of task execution [17]. In the past years, some techniques have been proposed for resource rescheduling [18-21]. Plankersteiner et al. [18] proposed a technique for resource rescheduling by addressing the fault tolerance. They introduced the deadline constraint for the task participated in the scheduling process. Cao et al. [19] proposed three strategies for resource rescheduling. If the VM is crashed at the time of task execution, this model automatically stores the executing task and waits for t seconds to repair of the VM. If the VM is not repaired, the task migrates to the other VM, otherwise the task will execute on the same VM. Park et al. [31] proposed group based mechanism for fault tolerance to achieve reliable resource management in MCC. This method considers the availability of resource in the mobile devices and groups them according to the resource availability. The resources of mobile devices are not stable, so the dynamic nature of the environment is considered for grouping of mobile devices. They applied the fault tolerance techniques using the replication or check points to the group. Choi et al. [4] proposed fault tolerance scheduling approach called as Content addressable network in MCC. This method integrates the social computing based environment for MCC. The fault tolerance scheduling is categorized with four sub modules such as cloud service delivery, replication and load balancing, QoS provisioning and malicious user filtering. This method achieved good results for service execution time and reliability. Wadhwa et al. [5] discussed various types of faults and fault tolerance mechanisms in the cloud. The issues of availability and reliability are solved by using the fault tolerance mechanism. They used the Nagios monitoring tool for identifying and analyzing various faults. Cheraghlou et al. [6] attempted a survey on analysis of fault tolerance architectures in cloud computing. The authors stated the policies of the fault tolerance architectures and finally they compare the fault tolerance approaches with fault detection capacity and fault recovery.

Furthermore, many algorithms [20, 22, 23] are proposed for fault tolerance in the distributed environment irrespective of rescheduling and replication. Few academicians concentrated on how to predict and manage the faults in the infrastructure which is helpful to detect the causes for failures. In this paper, the architecture for fault tolerance using DR-approach was developed. The advancement in recent years related to the bio inspired approaches drawn the attention of the world.

3. Problem Statement

The VMs are grouped under a server in a cloud system. Whenever the task offloading process is done by the mobile environment, the tasks are scheduled to the suitable VMs in the cloud. The failure of the computational resources (i.e. VMs) may occur at any time. The major reasons for the VM failure are:

- Inadequate server resources
- Incompatible server hardware and
- Conflicting VM tasks.

For initializing a VM, the server needs computational resources. Over committed or insufficient resources may leads to the failure of VM. This type of issues is called as inadequate server resources. As per the Virtualization concept, the VM abstracts image from the server underlying hardware. So, it is crucial to support the VM functionalities by the hardware which was assigned. If the hardware doesn't support the VM functionalities, then it leads to the VM failure and this type of failures is called as incompatible server hardware. Finally, another issue which causes VM failure is conflicting VM tasks. The scheduled tasks are assigned to the VMs for the execution, but some tasks are incompatible to the VMs and take a significant amount of time to generate the time out error. Though, the tasks generate the error, their execution process continuous to run at the background. So, the new tasks will not be executed in the VM and ultimately the VM failure occurs. To overcome the VM failure, an efficient fault tolerance mechanism is needed to handle the VM failures with in the servers. This paper proposed architecture for fault tolerance using Disease Resistance approach.

4. Disease Resistance Approach

The human body can be protected by many viruses, bacteria and foreign antigens by following an approach called as the disease resistance approach. The disease resistance (DR) approach is inspired by the biological model [29]. The overview of the disease resistance approach of the human body is shown in Figure 1.

In daily process, the human body is exposed to various microorganisms and some of these microorganisms will cause illness. To defend the microorganisms, the human body has the monitoring module which identifies the antigens. Whenever the antigens are identified, the response process is initialized by activating the macrophages. These macrophages will swallow the antigens and divide them into small pieces. The small pieces of antigens are presented to the T-cells. The T-cell examines the behaviour of the antigens and generates the B-cell. The B-cell produces the huge amount of antibodies for destroying microorganisms. The behaviour of the antigens and the functionality of antibodies are stored in the memory cell for further access. Through this disease resistance approach our body is shielded from different invaders.

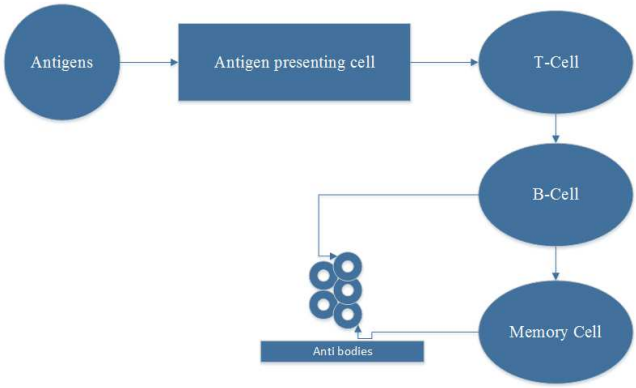

Figure 1. Disease resistance approach in human bodies

5. Task Model for Mobile Environment

The mobile environment is a wireless device which is connected to the cloud in ad-hoc manner. Whenever the user submits the job to the mobile environment, the job is partitioned into subtasks for parallel execution [32]. The mobile environment has some limitations i.e., battery power, computation capacity. The task offloading process is made used to overcome these limitations. The mobile environment follows the properties of directed acyclic graph for task submission. The DAG is represented as G = {V, E}, where V is the subtasks T = {T_1, T_2, …, T_n} and E is the connection between the two subtasks. E is formulated as {(T_i, T_j, W_{ij})| i≠j }, where W_{ij} is the weight of the edge. i.e data transferred from T_i to T_j. The example for the workflow of subtasks as DAG is given in Figure 2.

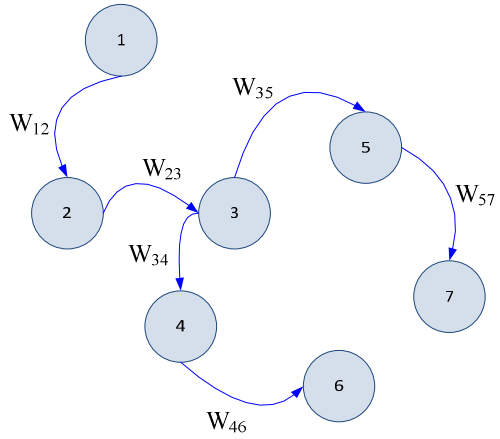

Figure 2. Directed Acyclic Graph for 7 subtasks

6. Architecture for Fault Tolerance using DR approach (DRFT)

The symbols used in this paper are tabulated in Table 1.

Table 1. Nomenclature

Symbol	Description
D	Represents the data center
S	Represents the server in the data center
S_{Active}	Represents the active servers
S_{Sleep}	Represents the sleep servers
δ	Represents the virtual machine with in the Server
α_{ij}	Represents the computing capacity of CPU corresponding to the virtual machine j with in the server i.
β_{ij}	Represents the memory corresponding to the virtual machine j with in the server i.
γ_{ij}	Represents the storage corresponding to the virtual machine j with in the server i.
ε_{ij}	Represents the execution time corresponding to the virtual machine j with in the server i.
α_{ij}^{c}	Represents the current computing capacity of CPU corresponding to the virtual machine j with in the server i.
W(trns)	Represents the data transfer
ins(trns)	Represents the total number of instructions.
$band_{ij \to ik}$	Represents the network bandwidth between δ_{ij} and δ_{ik}
α_{in},β_{in}, γ_{in}	Represents the computation capacity, memory capacity and storage capacity of the nth virtual machine in the server i, n is computed from 1 to k.
α_{max}, β_{max}, γ_{max}	Represents the maximum computation capacity, maximum memory capacity and maximum storage capacity of the overall VMs in the server.

This paper considers the conventional cloud offered by the cloud service providers. The cloud contains the resources in the form of servers and these servers are composed of a set of VMs [33]. The mathematical representation of the cloud model is given as follows.

$$D = \{S_1, S_2, ...S_n\} \quad (1)$$

$$S_{Active} = \{S_1, S_2, ...S_m\} \quad (2)$$

$$S_{Sleep} = S - S_{Active} \quad (3)$$

The eq. (1) shows that the datacenter D contains n number of servers and the cloud model assumes that all servers are in active position at the time of initialization. The servers have the capacity to dynamically switch their behaviour to active state or sleep state. S_{Active} is a set of active servers. The number of active servers is denoted by $|S_{Active}|$ and the set of sleep servers is denoted as S_{Sleep} and the number of sleep servers is denoted by $|S_{Sleep}|$. Each server S_i is served with different Virtual machines δ_{ij}, where 'i' is the server and 'j' indicates the corresponding virtual machine and each virtual machine representation is denoted as follows.

$$S_i = \{\delta_{i1}, \delta_{i2}, ..., \delta_{in}\} \quad where\ 1 \le i \le n \quad (4)$$

$$\delta_{ij} = \{\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \varepsilon_{ij}\} \quad (5)$$

Where α_{ij} the computation capacity of the virtual machine is, β_{ij} is the memory corresponding to the virtual machine, γ_{ij} represents the storage of the virtual machine and ε_{ij} represents the execution time.

The proposed architecture for fault tolerance in MCC based on DR-Approach, DRFT for server 'i' is shown in Figure 3. The DRFT has task queue, resource manager, DR unit and monitoring unit. The task queue is used to store the scheduled tasks in the queue. The resource manager handles the tasks from the task queue and assigns to the available resources. The monitoring unit contains the monitoring modules which are associated with each VM in the server. DR unit contains memory module, knowledge module and response module. The monitoring module is necessary for identifying the faulty VMs, a knowledge module is required to reschedule the tasks, the response module searches for the closest match of faults in the memory module and the memory module is used to store the strategies employed for the faults for future purposes.

The cloud environment had great similarities with the DR approach. The fault in VMs causes damage in the cloud performance as similar to antigens cause damage to human health [29]. Hence DR approach inspired to propose the process of identifying the fault VMs and rescheduling the task in the cloud environment.

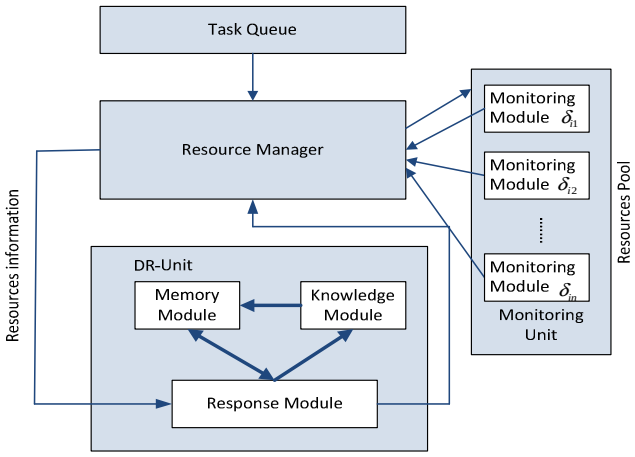

Figure 3. Architecture for Fault tolerance in MCC

The memory module, the response module and the knowledge module cooperate with each other for efficient rescheduling of tasks to the available resources in the cloud computing. The working process of DR unit is initiated whenever the faults are identified in the scheduling process otherwise the scheduling process will remain unchanged. For the normal scheduling of tasks, the cloud adapts the genetic algorithm (GA) [26] approach. Each module in the DR unit is explained as follows.

6.1 The Monitoring Module

In the cloud environment, each server is associated with the monitoring module, which identifies the fault in the VMs. The monitoring module follows the acknowledgement strategy for identifying the status of each VM. For every t seconds, the monitoring module sends the *status message* to all the active VMs. The VMs has to reply with *ACK message* within the time interval. If any VM do not respond with *ACK* then that VM is treated as crashed VM. The detail process of the monitoring module is given in Algorithm 1. The current computation capacity α_{ij}^{c} of the δ_{ij} is compared with the original computation capacity α_{ij}^{o} of the δ_{ij}. If any difference is identified, then the monitoring module collects information from δ_{ij} and sends to the response module.

Algorithm 1: Monitoring Module

Begin
Initialize the task scheduler
Initialize the monitoring module
Send *Status message* to the δ_{ij}
Wait for t seconds
if (*ACK message* is received from the δ_{ij})
if ($\alpha_{ij}^{c} < \alpha_{ij}^{o}$)
Collect the δ_{ij} information from S_i;
Send the information to the response module;
if (*ACK Message* is not received by the δ_{ij})
δ_{ij} is marked as crash;
Send the information to the Resource manager;
End

6.2 The Response Module

The response module is initiated to generate the rescheduling strategy. It is similar to the production of antibodies in B-cells. The B-cells generate the antibodies based on the microorganisms' behaviour and the strategy for the generation of antibodies is stored in the memory cell. In the same way, whenever the faulty VM is identified, the response module searches for the closest match of faults in the memory module. For instance, the virtual machine δ_{ij} is identified as a faulty machine, then the response module searches for the similar faults in the memory module. Assume that the rescheduling strategy for a faulty virtual machine δ_{ij} is to reschedule the task to the virtual machine δ_{ik} and δ_{ik} is selected as an antibody to the δ_{ij}. If there is more number of similar antibodies (i.e common properties like δ_{ik} in the memory module) then the similarity between the antibodies is measured as follows.

$$similarity = ((W(trns)\times(band)^{-1}) + (ins(trns)\times(\alpha)^{-1})) \quad (6)$$

Assume that the antibody for the δ_{ij} is selected as δ_{ik}, then the similarity function is calculated as

$$similarity_{ij\to ik} = ((W(trns)\times(band_{ij\to ik})^{-1}) + (ins(trns)\times(\alpha_{ik})^{-1})) \quad (7)$$

From the eq. 7, the closest similarity function (antibody) is calculated for the fault VM (antigen). The suggested virtual machine from the memory module is taken as equivalent antibody (EA). If EA is not found in the memory module then the knowledge module is initialized.

6.3 The Knowledge Module

The functionality of the knowledge module is to find the suitable EA for the antigens (i.e a faulty VM). If the suitable EA is not identified in the memory module, then the knowledge module request information of all the VMs in the host from the resource manager. The similarity function is calculated for each resource separately based on the eq. (8). The closest similarity is taken as the suitable EA for the faulty VM. This selected EA is stored in the memory module for future reference. The similarity function for the selection of EA in the knowledge module is given as follows.

$$similarity_{in\to ij} = (((\alpha_{in}-\alpha_{ij})\times(\alpha_{\max})^{-1}) + ((\beta_{in}-\beta_{ij})\times(\beta_{\max})^{-1}) + ((\gamma_{in}-\gamma_{ij})\times(\gamma_{\max})^{-1})) \quad (8)$$

Algorithm 2: Knowledge Module

Input: Number of servers, faulty VM δ_{ij}
Output: Equivalent antibody δ_{ik}
Begin
Select server S_i as S_x
Initialize search process for EA in memory module of server S_x

for l=1 to n do
Request the information of δ_{il} from S_x
Compute *similarity* from *eq. (8).*
end *for*
select *closest similarity* δ_{ik} as the EA
if (generated EA $\rightarrow$*QoS*)
select EA as the antibody for δ_{ij}
else
select another server
goto step 2.
end *if*
End

In Algorithm 2, the virtual machine which has the closest similarity will be chosen as an equivalent antibody for the faulty VM. If the generated EA meets the requirements of the user, then it is selected as the antibody for the antigen, otherwise select the other server and repeat the EA generation process.

6.4 The Memory Module

The memory module consists of memory cell which is used for storing the rescheduling strategies. The memory cells are responsible for storing the fault handling strategies for each VM and rescheduling strategy under this process. The memory module stores the maximum capacity of memory cells up to the limit. If the limit of the memory module exceeds, it discards the information of the strategy which is having the less frequency.

7. Simulation Environment

To evaluate the performance of the proposed architecture DRFT, the basic scheduling is done using three different algorithms Genetic algorithm (GA) [26], heuristic earliest time first (HEFT) [25] and Min-Min (MNMN) [24] algorithm. The performance of the DRFT approach is tested with the existing Dynamic group based fault tolerance approach (DGFT) [31]. The extensive simulations had been carried and the obtained results are presented. The performance estimation of the proposed approach is carried out in a well-known simulator called as CloudSim 3.0.3 toolkit [27]. The CloudSim simulator had a rich set of facilities to develop scheduling strategies for diverse cloud environments. The configuration for simulation setup is shown in the Table 2.

Table 2. Pre-defined Parameters for Simulation

Parameter	Value
Number of VMs	100-500
Computation capacity(MIPS)	1000-3000
Memory(Mb)	256-1024
Storage(GB)	10-40
Bandwidth(Mbps)	0.1-100
Fault rate (%)	5-20
Number of Tasks	300

The cloud simulator works with three inputs: cloud settings, workflow traces and task description. To create the network between the servers, the network topology is implemented by using the predefined network properties of CloudSim. This network topology creates the random network model which is analogous to the internet. The details of the job such as number of tasks, the number of VMs and the computation capacity are required to initialize the simulator. The initialization of the properties of task and VMs were randomly distributed among predefined set of VMs and task parameters.

The parameters for the genetic algorithm are given in Table 3. The crossover process follows the procedure of exchanging the genes between two chromosomes. The popular method is a single point crossover between two parent populations. The cross over rate lies in the range of 0.6 to 0.8. This mutation process is carried by randomly changing the bit positions in the chromosomes. The probability of the mutation rate must be low otherwise it leads to the re-initialisation of the population. So, the mutation value is taken as 0.1.

Table 3. Parameters for Genetic Algorithm

Parameter	Value
Maximum iterations	500
Crossover rate	0.8
Mutation rate	0.1
Size of the population	300
Convergence criteria	20 generations

7.1 Results Evaluation

The simulation results presented in this paper are classified under two fault rates, 10% and 20%. The performance factor which is considered for evaluation of the proposed approach is makespan. Makespan is defined as the total length of the schedule, i.e., the overall completion time of the submitted tasks. Figure 4 shows the simulation results of the HEFT, MNMN and GA under no fault rate. The GA algorithm is an optimizing algorithm which has the special mechanism to schedule the tasks to the VMs. It follows the evaluation procedure to find the best suitable pair of tasks and VMs from the available options. The algorithms are tested with three different conditions such as numerical analysis, without rescheduling and with DR-approach. The numerical analysis is made on the performance prediction of both VMs and tasks. The task execution under VMs with a defined fault rate without applying rescheduling comes under without rescheduling condition. The performance of the algorithms is enhanced with DR-approach and the performance evaluation is calculated under the defined fault rate comes in to the category of with DR-approach. In Figure 4 it is observed that the makespan value of GA algorithm is reduced to 15% when compared to the MNMN algorithm and 6% reduction when compared to the HEFT algorithm

Figures 5 and 6 show the efficiency of DR-Approach. It should be noted that the fault rate doesn't significantly hampers the performance of the DR-approach to maintain the makespan. The simulations are conducted with three well-known scheduling algorithms and with different fault rate conditions. The purpose of conducting these simulations is to demonstrate that how much makespan can be reduced with the efficient use of VMs.

In Figure 5, fault rate of 10% is introduced in to the MCC environment. The performance of the DR approach is evaluated under different conditions. It is observed that the GA algorithm with DR approach achieved better reduction in makespan when compared to the MNMN and HEFT algorithms. By the examining of experimental results, it is

clear that the GA achieved 12% reduction in makespan when compared to the MNMN algorithm and 4% reduction when compared to the HEFT algorithm.

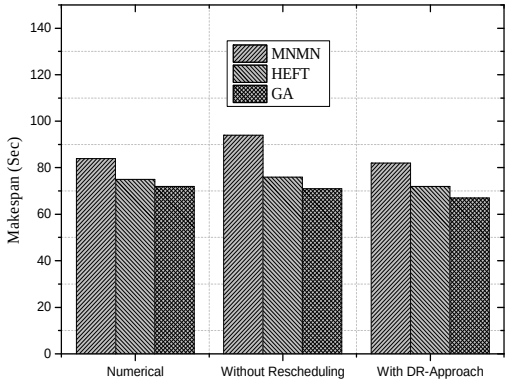

Figure 4. Performance estimation without Fault rate

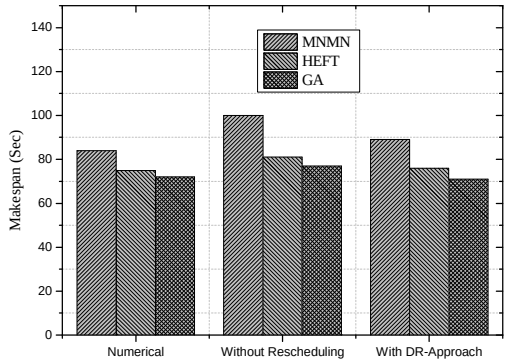

Figure 5. Makespan under the fault rate of 10%

In Figure 6, fault rate of 20% is introduced in to the MCC environment. 300 tasks are assigned for task scheduling and 100 VMs are utilized for resource allocation. The GA is employed with DR approach for fault tolerance mechanism. i.e., there is a chance of allocating 20 faulty VMs to the tasks. The simulations are carried with MNMN, HEFT and GA algorithm. The experimental results proved that the GA had achieved 8% reduction in makespan when compared to the MNMN algorithm and 3% reduction when compared to the HEFT algorithm.

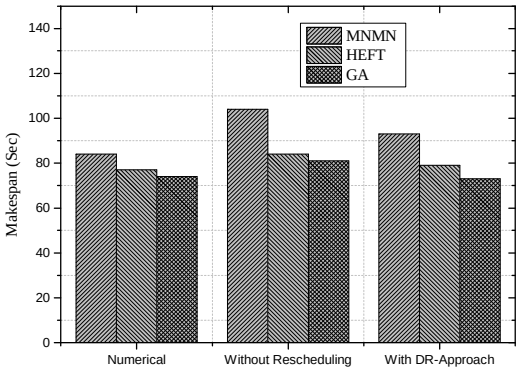

Figure 6. Makespan under the Fault rate of 20%

The performance of the DRFT is compared with the existing Dynamic group based fault tolerance approach (DGFT) [31]. Figure 7 shows the makespan of rescheduling approaches in DRFT and DGFT approach. The experiment is carried out at different levels of task assignment by applying the fault rate of 20 % in both DRFT and DGFT approach. The make span value of DRFT is reduced to 7% and the performance of DRFT is increased when compared to the DGFT approach.

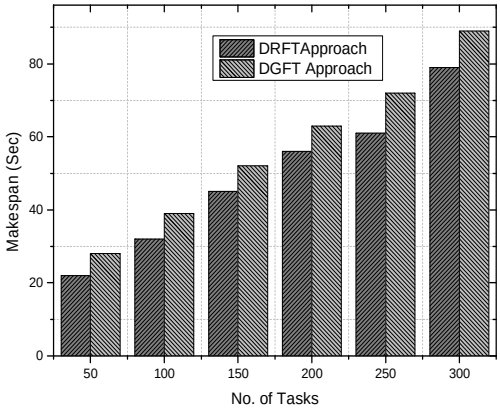

Figure 7. Performance evaluation of DRFT and DGFT approach

8. Conclusion

In this paper, an architecture DRFT is proposed for handling the fault VMs in the MCC environment. The mechanism is inspired by the human disease resistance system called as DR-Approach. Based on the human disease resistance system, the approach was developed with four important modules such as the monitoring module, response module, knowledge module and memory module. These four modules collaborate together to solve the resource rescheduling problem in the MCC environment. This paper considers three well-known algorithms for scheduling such as HEFT, Min-Min and GA for the series of simulations and the performance of DRFT approach is compared with the dynamic group based fault tolerance (DGFT) approach. The results proved that the DRFT had superior performance to handle the fault VMs under different conditions.

References

[1] Abolfazli, S. A. E. I. D., Sanaei, Z., Sanaei, M., Shojafar, M., & Gani. "Mobile cloud computing: The-state-of-the-art, challenges, and future research." Encyclopedia of Cloud Computing, *Wiley, USA*. 2015.

[2] Aceto, L., Larsen, K.G., Morichetta, A. and Tiezzi, F. " A cost/reward method for optimal infinite scheduling in Mobile Cloud Computing." In International Workshop on Formal Aspects of Component Software. pp. 66-85, 2015.

[3] Molina, M., Muñoz, O., Pascual-Iserte, A. and Vidal, J. "Joint scheduling of communication and computation resources in multiuser wireless application offloading." In 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), pp. 1093-1098, 2014.

[4] Choi, S., Chung, K., & Yu, H. "Fault tolerance and QoS scheduling using CAN in mobile social cloud computing." Cluster Computing, *17*(3), pp. 911-926, 2014.

[5] Wadhwa, A., & Bala, A. "Preventing Faults: Fault Monitoring and Proactive Fault Tolerance in Cloud Computing."

In Proceedings of International Conference on ICT for Sustainable Development, pp. 665-673, 2016

[6] Cheraghlou, M. N., Khadem-Zadeh, A., & Haghparast, M. "A survey of fault tolerance architecture in cloud computing." Journal of Network and Computer Applications, vol. *61*, pp. 81-92, 2016.

[7] El Zant, Bassem, Isabel Amigo, and Maurice Gagnaire. "Federation and revenue sharing in cloud computing environment." In Proceedings of the IEEE International Conference on Cloud Engineering. pp. 446–451, 2014

[8] Chen Shi, J. Wu, and Z. H. Lu. "A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness." In Proceedings of the IEEE 12th International Conference on Computer and Information Technology. pp. 177–184, 2012.

[9] Chen, N., Chen, W.N., Gong, Y.J., Zhan, Z.H., Zhang, J., Li, Y. and Tan, Y.S. "An evolutionary algorithm with double-level archives for multiobjective optimization." IEEE transactions on cybernetics, vol. 45, No.9, pp.1851-1863, 2015.

[10] Kumar, Senthil SK, and P. Balasubramanie. "Dynamic scheduling for cloud reliability using transportation problem." Journal of Computer Science vol. 8, No. 10 pp. 1615–1626, 2012.

[11] Li, Qiang, and Yike Guo. "Optimization of resource scheduling in cloud computing." In Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. pp. 315–320, 2010.

[12] Genez, Thiago AL, Luiz F. Bittencourt, and Edmundo RM Madeira. "Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels." In Proceedings of the IEEE Network Operations and Management Symposium, pp. 906–912, 2012.

[13] Javadi, Bahman, Jemal Abawajy, and Rajkumar Buyya "Failure-aware resource provisioning for hybrid Cloud infrastructure." J. Parallel Distrib. Comput. Vol. 72, No. 10, pp. 1318–1331, 2012.

[14] Smanchat, Sucha, and Kanchana Viriyapant. "Taxonomies of workflow scheduling problem and techniques in the cloud." Future Gener. Comput. Syst. Vol. 52, pp.1–12, 2015.

[15] Lee, Young Choon, Hyuck Han, Albert Y. Zomaya, and Mazin Yousif. "Resource-efficient workflow scheduling in clouds." Knowl.-Based Syst. Vol. 80, pp. 153–162, 2015.

[16] Ramezani F., Lu J., Taheri J. "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments." World Wide Web: pp.1–21, 2015.

[17] Zhu, Xiaomin, Chuan He, Rong Ge, and Peizhong Lu. "Boosting adaptivity of fault-tolerant scheduling for real-time tasks with service requirements on clusters." J. Syst. Softw, pp.1708–1716, 2011.

[18] Plankensteiner, Kassian, and Radu Prodan. "Meeting soft deadlines in scientific workflows using resubmission impact". IEEE Trans. Parallel Distrib. Syst. pp. 890–901, 2011.

[19] Cao, Yang, CheulWoo Ro, and JianWei Yin. "Scheduling Analysis of failure-aware VM in cloud system." Int. J. Control Autom. Vol. 7, No.1, pp. 243–250, 2014.

[20] Chen, Wei, Young Choon Lee, Alan Fekete, and Albert Y. Zomaya. "Adaptive multiple-workflow scheduling with task rearrangement." J. Supercomput. pp. 1–21, 2015.

[21] Olteanu, Alexandra, Florin Pop, Ciprian Dobre, and Valentin Cristea. "A dynamic rescheduling algorithm for resource management in large scale dependable distributed systems." Comput. Math. Appl. pp. 1409–1423, 2012.

[22] Guan, Qiang, Ziming Zhang, and Song Fu. "Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems." J. Commun. pp. 52–61, 2012.

[23] Rood, Brent, and Michael J. Lewis. "Grid resource availability prediction-based scheduling and task replication." J. Grid Comput. pp. 479–500, 2009.

[24] Javadi, Bahman, Daishi Kondo, Jean-Marc Vincent, and David P. "Discovering statistical models of availability in large distributed systems: An empirical study of seti@ home." IEEE Trans. Parallel Distrib. Syst. pp.1896–1903, 2009.

[25] Ibarra, Oscar H., and Chul E. Kim. "Heuristic algorithms for scheduling independent tasks on non-identical processors." Journal of the ACM. pp.280-289, 1977.

[26] Gu, Jianhua, Jinhua Hu, Tianhai Zhao, and Guofei Sun. "A new resource scheduling strategy based on genetic algorithm in cloud computing environment." Journal of Computers, vol. 7, no. 1, pp. 42-52, 2012.

[27] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya . "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and Experience, Vol. 41, pp. 23-50, 2011.

[28] Rahimi MR, Ren J, Liu CH, Vasilakos AV, Venkatasubramanian N. "Mobile cloud computing: A survey, state of art and future directions." Mobile Networks and Applications. pp. 133-143, 2014.

[29] Dasgupta, Dipankar. Advances in artificial immune systems." IEEE Comput. Intell. Mag. Vol. 1 No. 4, pp. 40–49, 2006.

[30] Chen, Wei, Sam Toueg, and Marcos Kawazoe Aguilera. "On the quality of service of failure detectors." IEEE Trans. Comput. pp. 561–580, 2002.

[31] Park, J., Yu, H., Kim, H., & Lee, E. "Dynamic group-based fault tolerance technique for reliable resource management in mobile cloud computing." Concurrency and Computation: Practice and Experience, 2014.

[32] Ayad, Soheyb, et al. "Cross-Layer Routing Based on Semantic Web Services Discovery with Energy Evaluation and Optimization in MANET." International Journal of Communication Networks and Information Security, *Vol.* 8, No.1, pp. 47, 2016.

[33] Toumi, H., Talea, A., Marzak, B., Eddaoui, A. and Talea, M. "Cooperative trust framework for cloud computing based on mobile agents." International Journal of Communication Networks and Information Security, Vol. *7*, No. 2, p.106, 2015.