International Journal on Emerging Technologies 11(3): 417-421(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Scheduling of Tasks (Cloudlets) in Heterogeneous Processing Cloud Environment

Nishant Kumar[1] and Raj Kumar[2]
[1]Assistant Professor, Department of Computer Science & Engineering,
GKV, Haridwar (Uttarakhand), India.
[2]Associate Professor, Department of Computer Science,
GKV, Haridwar (Uttarakhand), India.

(Corresponding author: Nishant Kumar)
(Received 24 February 2020, Revised 16 April 2020, Accepted 18 April 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Scheduling in cloud environment is a big challenge, it has two flavors in cloud environment one to schedule the placement of the virtual machines (VM) and second is the placement of cloudlet or tasks in the right virtual machine for the fast execution. In first type of scheduling to save the energy in the DataCenter it is always a good idea to re-arrange the running VM's on the underlying physical machines, so the underload physical machine can go to sleep to save energy. So, the assignment of the VM from the underloaded physical machine to other is a challenge. In second the placement of cloudlet to the VM for execution, the decision to VM is crucial. In DataCenter, the underlying environment is heterogeneous, it is a challenge to use all the VM which are now old and new high-end specs VM. So, balancing the tasks assignment is challenging. In the proposed work the placement of the tasks is taken up, the tasks are picked-up for execution on the FCFS basis and our algorithm assigns all the tasks to the VM which is providing the minimum execution time. It calculates the time of execution from all the VM available and calculates the time to finish the previous assigned or under execution tasks to find the minimum execution time. We will see the algorithm working in load scenarios.

Keywords: Cloud Computing Scheduling, CloudSim, Task Scheduling, Cloudlet Scheduling.

Abbreviations: VMs, Virtual Machines; CT Completion time; QCT Queue Completion Time, ET Estimated Time, LC Length of cloudlet; MIPS Million Instruction Per Second; MCT Minimum Completion Time; BW Bandwidth.

I. INTRODUCTION

In the recent year, it has seen that the cloud computing technology is being adopted by many companies for its cost and resource provisioning model. The trust by the companies in the technology has made the cloud service providers to build a bigger datacenter. To gain the trust and confidence in the technology, availability and security of the data are the key factors, and slow part of continuous research. A technology not only raises the issues for the data security or availability, but it also affects the living of the common man.

In the current scenario cloud computing is the only way to provision the resources virtually from anywhere. Many DataCenters established by the big players (like Google, IBM, Microsoft and Yahoo) of industry and providing all types of cloud services (including software, platform and infrastructure as a service) for the end user and their clients. DataCenters uses virtualization to provide complete utilization of servers to save the energy but to fulfill the needs, the size of the datacenter is very big having thousands of physical servers running all the time to serve all. A small change in the power consumption leads in a big amount of saving the energy.

To improve efficiency and performance further it is must to investigate the physical components and software architectures. One such scope of improvement is the scheduling algorithms. Scheduling comes in largely in two methods in cloud computing, one to schedule the placement of the virtual machines (VM) and second is the placement of cloudlet or tasks in the right virtual machine for the fast execution.

Algorithms plays a vital role in assigning the task or cloudlets to VM (a job in cloud) for execution. The management of virtual machines for the execution of the tasks needs to be efficient as it must comply with optimal resource utilization and faster execution. Balancing in the allocation of task to the VM is important this can drastically improve the performance.

Datacenters are increasing its capacity to accommodate more users and to provide the latest technology to the consumer. In this scenario various new physical machines are adding up with the faster execution speeds. So, it has become a heterogeneous environment where machines are having different capacities. Execution of the tasks on the machines which works fast will have the fast execution and less completion time of the cloudlet or tasks. In this scenario the factor completion time is considered as the key factor to achieve by scheduling.

The scheduling algorithm for cloud computing is different from the traditional scheduling algorithms. According to the architecture the scheduling is required to assign the virtual machine (VM) to a task which is in the queue for the execution. The process of assigning is not limited to the assignment only if the all available running VM are occupied then, a new instance of the VM will be initiated for the task assignment and completion [1].

II. RELATED WORK

In the past much work has been done in the field of scheduling in cloud computing and has been suggested various model to save energy earlier, to understand the scheduling in cloud computing it is needed to study those approaches.

Managing the resources in the cloud environment is the key to save the energy. If the resources are not managed properly, can lead large amount of memory consumption [2]. Scheduling the virtual machine onto the computer nodes is challenging task specially when there are multiple objectives other than QoS like reduction in request response time, workload balancing, priority-based task, deadline based response to maintain the DataCenter low-power mode. To increase the response time hybrid energy efficient scheduling approach seems better which is the combination of pre-power technique and least load first [3]. Considering the physical servers are running in low power mode by default in which the server goes down when not in use and boots up when a resource demand. The lag in-between to resource demand and boot-up increases the response time. The issue and suggested pre-power technique, which is whenever servers left capacity is less than, it immediately boots up the server, when the left capacity in ample amount then it shutdowns the system to save the power [3].

Furthermore, scheduling is not only limited to the assigning the tasks to the VM, it also needs to work with job scheduling and its fairness in the distribution of the CPU time. Xu *et al.,* points out the fairness in allocation and used Berger Model of distributive justice based on expectation states where fairness can be judged by every individual through distribution relationship comparison [4]. Dakshayini & Guruprasad added priority with jobs and maintains the high throughput of with 99% service finishing rate. It examines the time to be taken and service cost of the task and based on the trade-off takes the decision and sets priority [5]. Delavar *et al.,* shows the different algorithm where a job is divided in two sub-parts and calculate request time of job and acknowledge time of the job separately. The results show the increase in system performance, they named the algorithm RSDC [6]. The proposed research refers the problem of task duplication in heterogenous environment and a pair-based task scheduling to minimize the layover time. The proposed algorithm PTS considered both lease time and converse lease time to make a scheduling decision and used transfer time to make the proposed algorithm more realistic [7].

Another cause of in-efficiency and delays is imbalanced load in DataCenters which increases the complexity. Chen *et al.,* introduced the algorithm, a load balanced Min-Min scheduling algorithms and results shows the decrease in makes pan and high resource usage [8]. In the same array Devipriya & Ramesh proposed an improved max-min algorithm which enhances the performance by reducing the response time and completion time based on RASA with max-min strategy. It assigns the smaller tasks to the faster resources and longer tasks to the slower processor, but this algorithm has drawback of lower make span in comparison to the Max-Min algorithm.

QoS in cloud has added few more parameters from which a parameter trust, is a big concern for the most of the companies to shift their work from the traditional system to the cloud, moving the data out off the campus and giving to the cloud companies raised concern of trust under two categories the privacy of data and said execution speed. Wang *et al.,* taken up the trust in execution speed, suggested the Bayesian Cognitive Inspired trust model with dynamic scheduling algorithm, Cloud-DLS. A simple method to gain the trust is high execution ratio, as the execution increases the trust increases [10].

Now computing has taken a turn, where QoS factors are sufficient to bring the efficient computing, but this didn't approach to the deadline-based tasks or workflow-based tasks. Tasks needs to be aligned with the deadline, this gives a scope of little delay also, if one work completes fast then other may have little extra time to spare and that is where actually the efficiency can be achieve. In the heterogeneous environment there are physical machines of all the capabilities. If a task is assigned to a faster machine then the other task can be assigned to a relatively slower machine, in-order to fully utilize all physical servers. The assignment of the workload based on deadlines known as workflow. So, the workflow scheduling is required to be optimized. Singh *et al.,* proposed an energy efficient workflow scheduling (EEWS) algorithm which scores a better makespan while considering the real-world scheduling constraints like the performance variability of VMs, VM boot and shut-down time [11].

Krishnadoss & Jacob considered the makespan and cost as an important factors for optimization factors and merged two algorithm to propose a new algorithm named as Oppositional Cuckoo Search Algorithm (OCSA), which takes input the number of task, number of host machine, number of virtual machine and provides makespan and cost as output. It uses the fitness function and rank the solution in last to find the best solution [12].

Wang *et al.,* points the main objective the execution time and delays, it uses the catastrophic genetic algorithm for the decision to transfer the task and quantifies total task completion time and penalty factor as a fitness function [13].

III. MODEL AND FORMULATION

A. Queuing System Models

Consider queuing system M/M/C:/∞/∞ with c parallel servers (for practical 4 parallel servers considered) and considering infinite queue length and infinite population. Because this is a heterogeneous system and all system has own capacity so the MIPS cannot be fixed. Considering that all the jobs coming for the service will get the system or execution time. According to the poison theory

$$\frac{\lambda}{c\mu} < 1 \quad (1)$$

where λ is the arrival rate, μ is the service rate and c is the number of available servers.

$$\lambda_n = \lambda \quad (2)$$

n represent the jobs, the service rate may vary as per the jobs are coming. When the jobs are less than the servers available, the service rate will be

$\mu_n = n\mu \qquad n < c$ (3)

If the jobs are more than the server and there will be some queue and service rate will be

$\mu_n = c\mu \qquad n \geq c$ (4)

Considering when there is no one in the system and all the servers are free, in this scenario as any job will come there will be no queue time.

$$p_0 = \frac{1}{\sum_0^{c-1} \frac{(\frac{\lambda}{\mu})^n}{n!} + \frac{(\frac{\lambda}{\mu})^c}{c!} \frac{1}{1-\frac{\lambda}{c\mu}}} \quad (5)$$

when there are *n* jobs in the system, and are less than the available server then there will be no queue

$$p_n = (\frac{\lambda}{\mu})^n \frac{1}{n!} p_0 \qquad n<c \quad (6)$$

when there are *n* jobs in the system, and there are more jobs than the server. In that scenario to predict the when the job will get the system

$$p_n = \frac{\lambda^n}{c!\ \mu^n c^{n-c}} p_0 n \geq c \quad (7)$$

B. Scheduling Policies

A scheduling policy, denoted by π, determines the task assignment in the system. The scheduling decisions are made based on the execution time and priority with the assumption that pre-emption is allowed. Hence if a task is being executed by the server which is running slow and not matching the required finish time, then it can be shifted to other machine for faster execution when faster machine is available. It can also be paused for the execution of other task having higher priority.

A policy is said to be non-anticipative, if the scheduling decisions are made without using any information about the future job arrivals. It is said to be anticipative, if it has the information of future arriving jobs. For periodic and pre-planned services, future jobs arrival can be predicted in advance. The goal is to design the low complexity, non-anticipated scheduling policies.

C. Wait Metrics

If job i with high priority comes then it will be assigned to the faster server which can execute task in minimum time and current job j will be paused or migrated to some other server which is free. This can incur the cost as the promise time of the task j completion is changed.

For each job j, Cj is the job completion time, Ci is the completion time of the available tasks, Wj is the wait time.

Wj=Ci-Cj (8)

D. Algorithm Functions

The algorithm function for all the task entered the server. Some of them completed and some are still under execution.

$$f(a) \leq f(c(\pi)) \quad (9)$$

Where c is the completion time and a is the all tasks or jobs has entered the server or either completed or under service at the time. π is a scheduling policy.

$$ET_i = \frac{LC_i}{MIPS_{vm_i}} \quad (10)$$

As the job will arrive for the execution, the execution time ETi of the job can be easily calculated using the length of cloudlet (LCi) and MIPS rating of each virtual machines available for the compilation. If the selected virtual machine is idle, the process will be allocated to it. Otherwise the process will wait in queue. The waiting time in queue (Qct) will be calculated using the following equations.

$$Qct_2 = [CT_1 - \text{Timestamp}_1] \quad (11)$$

$$Qct_i = [Qct_{i-1} + ET_{i-1}] i>2 \quad (12)$$

Jobs are coming in FCFS basis to the scheduler, but all the virtual machines are busy executing their assigned jobs, in that scenario the scheduler will examine the job and find out the earliest completion of the job on the specific virtual machine. The Qct_2 represents the waiting time for the first task in queue. The job will wait in the queue for its execution. Qct_i represents the waiting time for the other tasks excluding first task in the queue. Timestamp is required to find out the system time on which the job will be complete. To find the completion time CT_i, it is must to find execution time (ET_i), queue time (QT_i) and the timestamp. This will output the estimate completion time of the processes running on one virtual machine.

Minimum completion time (MCT_i) is required to be find from all the completion time calculated on each virtual machine.

$$\text{CT}_\text{i} = [\text{ET}_\text{i} + \text{Qct}_\text{i} + \text{timetamp}_\text{i}] \quad (13)$$

$$\text{MCT}_\text{i} = \text{Min}[\text{CT}_\text{i}] \quad (14)$$

IV. PSEUDOCODE

Algorithm 1: Cloudlet Scheduling without Migration

```
Identify the time when VM is ready
for j from 0 by 1 to getVMCretedList-1 do
{
        ReadyTimeVM[j]=system.clock
}
//Identify the MIPS of all VM's
for I from 0 to 1 getVMList-1 do
{
        VM_MIPS[j]=VMj.MIPS
}
sort "VM_MIPS" array into descending order for the first time.
//Identify the completion time of submitted cloudlet from VM_MIPS of all VMs
for i from 0 by 1 getCloudletListSize-1
{
        for j from 0 by 1 to getVMList
        {
        //Execution time of Ci on VMj
                ETi=lengthofCi/VM_MIPSi
                //Queue Time of VM
                qCTij=Timestampj+Wj
        //Completion time of Ci on VMj
                CTij=ETij+qCTij+TimeStampVM(i)
//Find Minimum Completion Time(MCT) of Ci
                MCT=Min(CTij)
//Find the lowest MCT from all the VM and assign the cloudlet
        if(VM==idle)
        {
        setVM=getVmsCreatedList.get(vmid)
```

```
        setCloudlet=setVM(vm.getvmid)
        }
        else
        {
        vm.queue();
        setVM=getVmsCreatedList.get(vmid)
        setCloudlet=setVM(vm.getvmid)
        }
        SendNow ()
        //cloudlet submitted
}
}
//Remove all the assigned cloudlets from the cloudlet list
```

V. IMPLEMENTATION

A. CloudSim Simulation Environment

The CloudSim toolkit used to simulate the scheduling algorithm. The configuration details are as below.

B. VM Configuration

```
long size = 10000; //image size (MB)
int RAM = 512; //vm memory (MB)
int mips = 1000;
long bw = 1000;
int pesNumber = 1; //number of cpus
String vmm = "Xen"; //VMM name
```

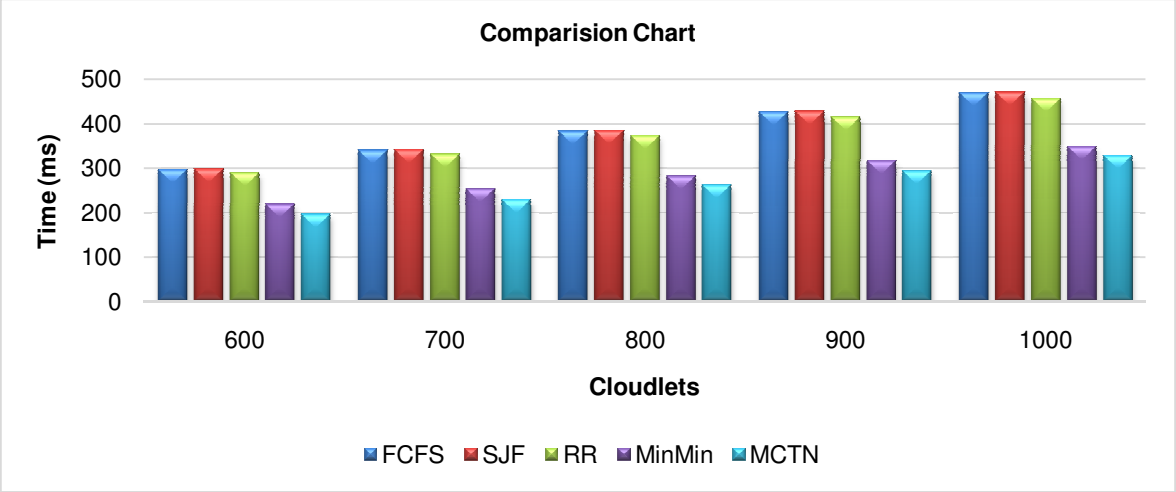
C. Cloudlet Configuration

```
long length=1000;
        long fileSize = 300;
        long outputSize = 300;
        int pesNumber = 1;
```

Table 1: Comparison of MCTN Algorithm with Existing Algorithms.

No. of Machines	No. of Cloudlets	FCFS (ms)	SJF (ms)	RR (ms)	Min-Min (ms)	MCTN (ms)
4	600	296.48	298.16	289.99	219.65	196
	700	339.24	341.45	331.65	253.33	229.77
	800	382.57	384.36	373.31	282.5	261.89
	900	425.76	427.6	414.97	316.23	294.02
	1000	469.09	470.52	456.58	348.69	327.67

Graph 1: Comparison of MCTN Algorithm with Existing Algorithms.



D. Host Configuration

```
int RAM = 2048; //host memory (MB)
        long storage = 1000000; //host storage
        int bw = 10000;
        String arch = "x86";
        String os = "Linux";
        String vmm = "Xen";
        double time_zone = 10.0;
        double cost = 3.0;
        double costPerMem = 0.05;
        double costPerStorage = 0.1;
        double costPerBw = 0.1;
```

VI. RESULTS

The comparison of the other scheduling algorithms with the proposed MCTN algorithm has been done. All the scheduling algorithm have been tested in the above cloud configuration. The four pre-existing scheduling algorithms are FCFS, RR, SJF, MIN-MIN are used for comparison. Makespan is a parameter is an important factor for optimization factors [12], result analysis of the algorithm is tested on makespan parameter.

Mashuqur Rahman *et al.,* has also compared the proposed algorithm with RR, Min-Min and Max-Min claimed to have reduced makespan in the proposed algorithm in comparison to only with Min-Min algorithm and has taken 7 task to be executed with 4 virtual machines [14].

The algorithm here is tested on the varying load of cloudlets from 600 to 1000 cloudlets on four virtual machines. The shortest job first (SJF) algorithm has took the maximum time (in ms) 296.48ms to complete

600 cloudlets and took 470.52ms for1000 cloudlets. The round-robin (RR) and first come first serve (FCFS) both algorithms did complete quite well by taking 289.99ms and 296.48ms for 600 cloudlets respectively comparatively to SJF and perform well but it failed to beat the Min-Min algorithm which took 219ms for 600 cloudlet and 348.69ms for 1000 cloudlets for the execution. The proposed algorithm MCTN performs much better by just taking 196ms for 600 cloudlets and took 327ms for the 1000 cloudlets. MCTN utilizes all the available machine from the cloud's heterogeneous environment for the execution of the tasks, and differs in capability of execution, but still algorithm confirms the use of all the available machine. The results show that the existing algorithm is not designed for the heterogeneous environment and thus taking more time to execute the task. For the environment like cloud it is must to design the algorithm specifically for the environment for the efficient use of the resources.

VII. CONCLUSION

Cloudlets are assigned to the virtual machine which will execute the task in the lowest completion system time. It is not mandatory to choose the fastest virtual machines always, it has been seen that lowest capable machine is also giving the less completion time in some cases. Thus, algorithm confirm the use of all the machines regardless of its performance metrics. It commits to use the maximum optimal resource utilization. In this algorithm the completion time includes all the available parameters like as timestamp, wait time of the cloudlets or the complete queue time etc. Execution results are showing the significant improvement in the execution time.

VIII. FUTURE SCOPE

In future the work can be carried towards the virtual machine migration policy which will enhance the efficiency. The future work can also include the AI engine, to learn the pattern at the runtime of usages in respective environment to provide the better results for specific set of requirements. It can also extend to the deadline-based task assignment.

REFERENCES

[1]. Sotiriadis, S., Bessis, N., Amza, C., & Buyya, R. (2016). Vertical and horizontal elasticity for dynamic virtual machine reconfiguration. *IEEE Transactions on Services Computing*, 1-14.

[2]. Basmadjian, R., De Meer, H., Lent, R., & Giuliani, G. (2012). Cloud computing and its interest in saving energy: the use case of a private cloud. *Journal of Cloud Computing: Advances, Systems and Applications*, *1*(1), 1-25.

[3]. Li, J., Peng, J., Lei, Z., & Zhang, W. (2011). An energy-efficient scheduling approach based on private clouds. *Journal of Information &computational Science*, *8*(4), 716-724.

[4]. Xu, B., Zhao, C., Hu, E., & Hu, B. (2011). Job scheduling algorithm based on Berger model in cloud environment. *Advances in Engineering Software*, *42*(7), 419-425.

[5]. Dakshayini, D. M., & Guruprasad, D. H. (2011). An optimal model for priority based service scheduling policy for cloud computing environment. *International journal of computer applications*, *32*(9), 23-29.

[6]. Delavar, A. G., Javanmard, M., Shabestari, M. B., & Talebi, M. K. (2012). RSDC (reliable scheduling distributed in cloud computing). *International Journal of Computer Science, Engineering and Applications*, *2*(3), 1-16.

[7]. Mei, J., Li, K., & Li, K. (2014). A resource-aware scheduling algorithm with reduced task duplication on heterogeneous computing systems. *The Journal of Supercomputing*, *68*(3), 1347-1377.

[8]. Chen, H., Wang, F., Helian, N., & Akanmu, G. (2013). User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing. In *2013 national conference on parallel computing technologies (PARCOMPTECH)*, 1-8.

[9]. Devipriya, S., & Ramesh, C. (2013). Improved Max-min heuristic model for task scheduling in cloud. In *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE),* 883-888.

[10]. Wang, W., Zeng, G., Tang, D., & Yao, J. (2012). Cloud-DLS: Dynamic trusted scheduling for Cloud computing. *Expert Systems with Applications*, *39*(3), 2321-2329.

[11]. Singh, V., Gupta, I., & Jana, P. K. (2019). An Energy Efficient Algorithm for Workflow Scheduling in IaaS Cloud. *Journal of Grid Computing*, 1-20.

[12]. Krishnadoss, P., & Jacob, P. (2018). OCSA: Task scheduling algorithm in cloud computing environment. *International Journal of Intelligent Engineering and Systems*, *11*(3), 271-279.

[13]. Wang, S., Li, Y., Pang, S., Lu, Q., Wang, S., & Zhao, J. (2020). A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline. *Scientific Programming,* 1-9.

[14]. Mashuqur Rahman, A K M, Aslam Uddin, K. M., Arbe, N., Jahan, L. and Md Whaiduzzaman (2019). Dynamic task scheduling algorithms in cloud computing. *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, 1280–1286.

How to cite this article: Kumar, N. and Kumar, R. (2020). Scheduling of Tasks (Cloudlets) in Heterogeneous Processing Cloud Environment. *International Journal on Emerging Technologies*, *11*(3): 417–421.