IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-6, Issue-6 E-ISSN: 2347-2693

VM Optimization using Deadline Based Task Scheduling in Cloud Computing

Monika[1*], Pardeep Kumar[2], Sanjay Tyagi[3]

[1,2,3] Department of Computer Science and Applications, Kurukshetra University, Kurukshetra-136119, India

[*]*Corresponding Author: monika.banasthali@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 18/Jun/2018, Published: 30/Jun/2018

***Abstract* -** The approach of scheduling the tasks directly impacts the performance of cloud. Deadline based approach is the major area of concern in cloud computing, because deadline based tasks must be executed in time. In this paper, a novel deadline based task scheduling system model has been proposed. In proposed model, tasks have been executed on the basis of deadline constraint. Auto-scalability and resource optimization are other factors that have also been taken into consideration into the present work. The proposed model assists in reducing the makespan of the requests that is valuable to user.

***Keywords* -** Cloud computing, deadline based constraint, makespan, resource optimization, scheduling algorithm.

I. INTRODUCTION

Cloud computing provides dynamic resource allocation and fulfils the clients need at any cost with respect to time. Some users do not care for the cost at which their request is being fulfilled. In other cases, users may provide time and cost constraints as a basic parameter of scheduling the jobs. In this paper, system model has been proposed based on deadline with respect to time. Along with deadline, resource optimization has also been considered in this paper. Resource manager optimizes the number of resources used in execution that automatically reduce energy.

II. RELATED WORK

Deadline based approach is not a new approach of scheduling. In this area, Javier Celaya et. al. (2011) proposed network based methodology. In this methodology, decentralized scheduler was used. Global scheduler found availability of nodes and local scheduler used EDF (Earliest Deadline First) scheduling policy to execute the requests. However, on cloud, using cost and tasks, flow deadline scheduling was implemented by Maciej Malawski et. al. (2012). This approach worked on different workflows of a task. Along with workflow, tasks priority of deadline based tasks were set to schedule the tasks efficiently. To reduce cost, Nitish Chopra et. al. (2013) enhanced the HEFT scheduling algorithm. Their approach worked on private as well as public clouds. First of all, proposed methodology checked the availability of resources that could finish the tasks in time. If resources were not found, then private clouds were availed to fulfil the request on time as well as cost constraint. Deadline tasks were handled by graph theory, using Bipartite methodology by Chien-Hung Chen et. al. (2014), while ILP (Integer Linear Programming) was used by Zhao-Rong Lai et. al. (2014) to handle deadline based tasks.

Considering cost, task dependency should be handled properly, while executing deadline based tasks. Dinesh Komarasamy et. al. (2015) first removed the tasks dependency, filtered according to their priority and then executed the tasks. Apart from these factors, Longkun Guo et al. (2017) handled deadline based tasks along with increasing the workload of the resources. This increased the CPU utilization and also minimized the number of used resources.

To execute the deadline based requests, auto-scaling technique is used. With this technique, resources can be scaled up and scaled down according to need. Hyejeong Kang et. al. (2013), in their work, simply auto-scaled the resources that satisfy the request's requirement and SLA defined by the user. Jieun Choi et. al. (2015), in their work, first found out the appropriate cluster. If no cluster was found according to need, then request was moved to the private resource. Still, if no appropriate private resource was found, then new private resource was created according to need.

Vinay et. al. (2016) proposed a methodology, in which resources were auto-scaled to execute sub-tasks. In their work, child tasks were checked if they can be executed in time or not. If not, then resources were auto-scaled to finish the tasks in time.

III. PROBLEM FORMULATION

Execution of request on given period of time is the users' demand while using cloud resources. This demand leads their request to the deadline based task scheduling. In deadline based scheduling, all tasks are put in a queue and are scheduled using particular methodology. Auto-scaling is another factor that is included in the deadline based scheduling. Using auto-scaling factor, our resources can be scaled whenever needed. There are two types of auto-scaling methods in cloud environment:

- **Horizontal scaling:** In this type of scaling method, resource is created of the same type i.e. same storage and CPU processing power to the resource that is in use.

- **Vertical scaling:** Using this scaling method, resource configuration can be scaled up or down while executing the tasks.

IV. PROPOSED SYSTEM MODEL ON RESOURCE OPTIMIZATION USING DEADLINE CONSTRAINT FOR TASK SCHEDULING

Let there be n tasks named as $t_1, t_2, \ldots, t_n$. These deadline based tasks are arranged in a queue, Q_1, in ascending order (considering lowest number with highest priority). Then it is checked whether all tasks in Q_1 can finish the jobs according to $\mathbf{t_i / r}$, where t_i is size of the i^{th} task and r is processing power of the resource. If tasks can't be completed in time, then particular tasks will be aborted. Then execution time and next_deadline of the tasks is compared. The next_deadline is calculated by equation (i):

next_deadline = deadline of task – execution time of current task to be executed (i)

For any task having execution time greater than its next_deadline, new resource is created and task is assigned to that resource. During execution, we again check whether remaining tasks could be executed in time or not. If not, then a new resource is generated and that individual resource is provided to the particular tasks. The process continues until all jobs / requests have been executed. During this process, resource manager checks the status of each resource. If any resource gets free, then request is assigned to that resource instead of creating new resource.

V. ASSUMPTIONS OF PROPOSED MODEL

The proposed methodology not only executes the deadline based requests in time, but also optimizes the resources also that helps in reducing power consumption to some extent. The proposed methodology has some assumptions, which are:

- The tasks placed in deadline queue are non-preemptive i.e. the execution of a task cannot be stopped forcefully and the resource cannot be assigned to another task.
- Horizontal scaling method is used for auto-scaling approach.
- Tasks are arranged based on their deadline time and not on cost & priority.
- All tasks are independent to each other.
- If new task comes, then it will not be entertained i.e. nature of proposed model is static.

VI. ALGORITHM

The following abbreviations have been used in algorithm:
n_deadline = next deadline of tasks after executing current task.

cur_exe = execution time of i^{th} task.
T_i = i^{th} task.
T_n = n^{th} / last task.
V_k = k^{th} virtual machine / resource.
T_i.exe = execution time of i^{th} task.
Steps of proposed methodology:

- Submit deadline based tasks to the server.
- Arrange the tasks in ascending order according to deadline in a queue.
- Set i = 0, k = 0.
- While i < n do
  - If T_i.exe < T_i.n_deadline
    then
    assign T_i task to V_k resource.
    Else
      If any resource other than V_0 is free
      then
      Assign T_i to that free resource
      Else create a new resource V_{k++} and
      Assign Ti task to V_{k++} .(end of if statement)
  - T_i.deadline=T_i.n_deadline
    T_i.n_deadline = T_i. n_deadline – cur_exe
- done (end of while loop)
- Stop.

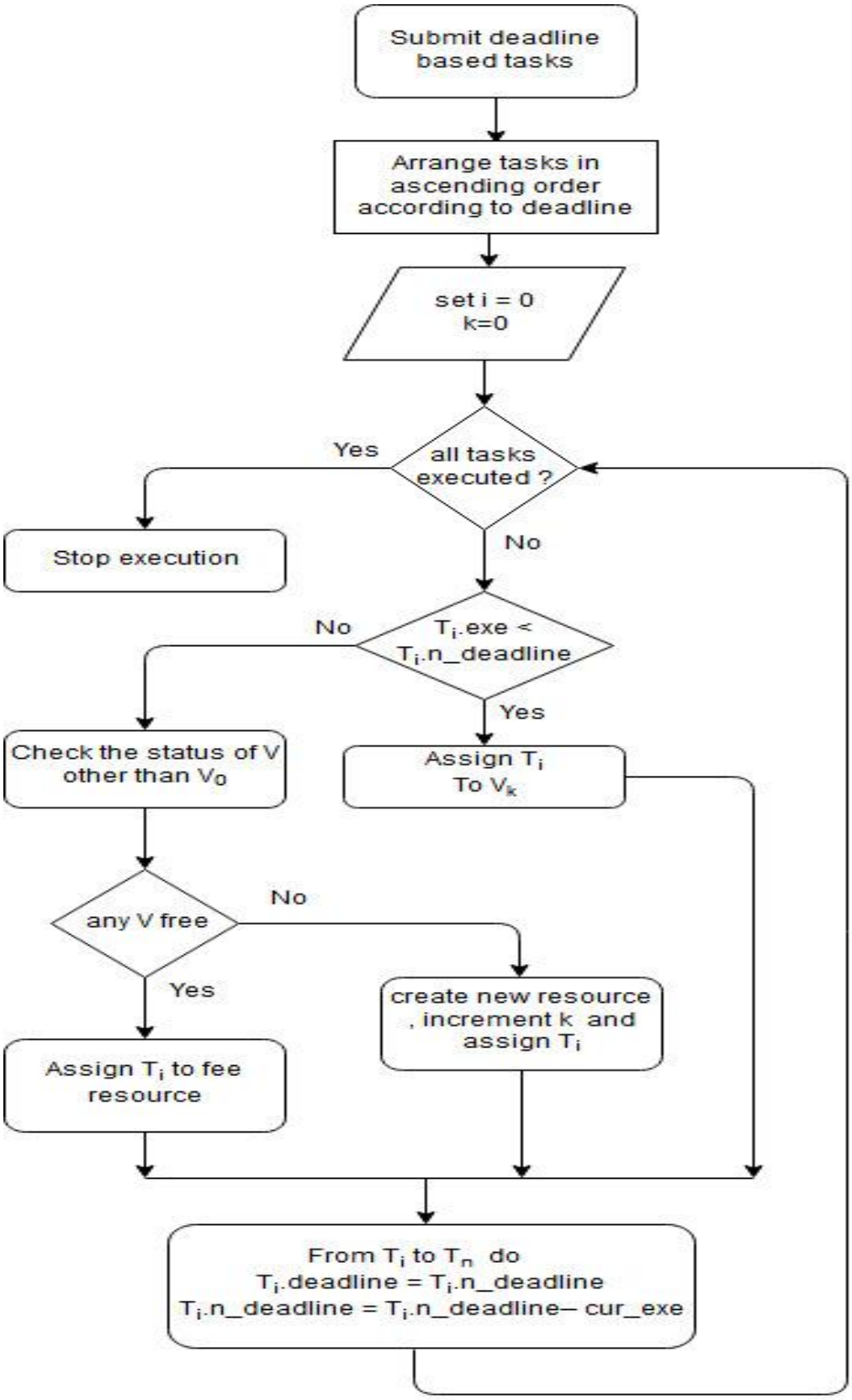

Fig. 1 Flow chart of deadline based scheduling algorithm.

VII. EXPERIMENT AND ANALYSIS

The proposed algorithm has been analyzed by taking a scenario of ten tasks with their deadline constraint. To perform this analysis, following configuration has been taken:
Resource size = 20,000 mips.

Here, only one resource, vm_0, has been considered for executing the tasks and following tasks have been stored in Q_1.
Tasks configuration has shown in table 1:

Table 1

Task (T)	Task size in (MI)	Execution time (T.exe) in ms.	Deadline (D) in ms.	Next deadline (n_deadline)
T_1	4000	200	400	200
T_2	6000	300	600	400
T_3	8000	400	800	600
T_4	10000	500	1000	800
T_5	12000	600	1200	1000
T_6	14000	700	1400	1200
T_7	16000	800	1600	1400
T_8	18000	900	1800	1600
T_9	20000	1000	2000	1800
T_{10}	22000	1100	2200	2000

After executing T_1, if it is found that first task has been executed in time, then scenario of table 1 will change into scenario shown in table 2.

Table 2

Task (T)	Execution time (T.exe) in ms.	Deadline (D) in ms.	Next deadline (n_deadline)
T_2	300	400	100
T_3	**400**	**600**	**300**
T_4	500	800	500
T_5	600	1000	700
T_6	700	1200	900
T_7	800	1400	1100
T_8	900	1600	1300
T_9	1000	1800	1500
T_{10}	1100	2000	1700

In table 2, execution time of T_3 is less than its next deadline time. Therefore vm_1 resource is created and T_3 is assigned to it. After executing T_2, resulting scenario is shown in table 3.

Table 3

Task (T)	Execution time (T.exe) in ms.	Deadline (D) in ms.	Next deadline (n_deadline)
T_4	500	500	0
T_5	**600**	**700**	**200**
T_6	**700**	**900**	**400**
T_7	**800**	**1100**	**600**
T_8	**900**	**1300**	**800**
T_9	1000	1500	1000
T_{10}	1100	1700	1200

As shown in table 3, T_5, T_6, T_7, T_8 have less execution time than their next deadline time. In such case, first we check if created resource i.e. vm_1 is free or not. In this case, vm_1 is not free. Therefore, we scale our resource by creating resources vm_2, vm_3, vm_4, vm_5 and assign it to T_5, T_6, T_7, and T_8 respectively.

After executing T_4, we found the scenario shown in following table 4:

Table 4

Task (T)	Execution time (T.exe) in ms.	Deadline (D) in ms.	Next deadline (n_deadline)
T_9	1000	1400	0
T_{10}	**1100**	**1600**	**2000**

To execute T_{10}, new resource should be created. Before creating, it will be checked whether there is any created resource, which is available to execute the task. It was observed that vm_1 is free. Therefore, T_{10} will be assigned to vm_1.
Now, the following scenario shown in table 5 is observed.

Table 5

Resource	Tasks in Resource	Remaining time of resource to be free
vm_0	T_9	1000
vm_1	T_{10}	1100
vm_2	T_5	400
vm_3	T_6	200
vm_4	T_7	300
vm_5	T_8	400

By the given scenario of 10 tasks, it has been observed that only 6 resources are needed to complete the tasks within given deadline.

VIII. CONCLUSION AND FUTURE WORK

One of the major motive of cloud computing is that every task should be completed on or before the deadline of the

task. To implement this deadline based algorithm, auto-scaling of the resources is performed in an economical and efficient way in this paper. The proposed approach not only scales the resources but also optimizes them. The minimum number of possible resources had been created in this approach. As creation of more resources consumes more energy, a methodology can be proposed that can help in reducing power consumption in future.

REFERENCES

[1] Javier Celaya, Unai Arronategui, "A Highly Scalable Decentralized Schedulerof Tasks with Deadlines", in IEEE/ACM 12th International Conference on Grid Computing, 2011, pp. 58 – 65.

[2] Lu Guan, Ying Wang, Yanfei Li, "A Dynamic Resource Allocation Method in IaaSBased on Deadline Time", in IEEE 14th Asia-PacificNetwork Operations and Management Symposium (APNOMS), 2012, pp. 1 – 4.

[3] Guan Le, Ke Xu, Junde Song, "Dynamic Resource Provisioning and Scheduling with Deadline Constraint in Elastic Cloud", in IEEE International Conference on Service Science, 2013, pp. 113-117.

[4] Zhao-Rong Lai, Che-Wei Chang, Xue Liu, Tei-Wei Kuo, Pi-Cheng Hsiu, "Deadline-Aware Load Balancing for MapReduce", in IEEE 20th Internati-onal Conference on Embedded and Real-Time Computing Systems & Applications, 2014, pp. 1-10.

[5] Chien-Hung Chen, Jenn-Wei Lin, Sy-Yen Kuo, "Deadline-Constrained MapReduce Scheduling Based on Graph Modelling", in IEEE International Conference on Cloud Computing, 2014, pp. 416-423.

[6] Maurice Khabbaz, Chadi Assi, "Modelling and Analysis of A Novel Deadline-Aware Scheduling Scheme for Cloud Computing Data Centers",in IEEE Transactions on Cloud Computing, 2015, pp. 15.

[7] Dinesh Komarasamy, Vijayalakshmi Muthu-swamy, "Adaptive Deadline based Dependent Job Scheduling algorithm in Cloud Computing", in IEEE Seventh International Conference on Advanced Computing, 2015, pp. 1-5.

[8] R Krishnam Raju Indukuri, Suresh Varma Penmasta, Dr. M V Rama Sundari, Dr. G. Jose Moses, "Performance Evaluation of Deadline Aware Multi Stage Schedulingin Cloud Computing", in IEEE 6th International Conference on Advanced Computing, 2016, pp. 229-234.

[9] Vinay Kand S M Dilip Kumar, "Auto-scaling for Deadline Constrained Scientific Workflows in Cloud Environment", in IEEE Annual India Conference (INDICON), 2016, pp. 1-6.

[10] Longkun Guo, Hong Shen, "Efficient Approxi-mation Algorithms for the Bounded Flexible Schedu-ling Problem in Clouds", in IEEE Transactions on Parallel and Distributed Systems, 2017, pp. 1-12.

[11] Jieun Choi, Younsun Ahn, Seoyoung Kim, Yoonhee Kim1, Jaeyoung Choi, "VM Auto-scaling Methods for High Throughput Computing on Hybrid Infrastructure", in cluster computing, springer, 2015, pp. 1063–1073.

