International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August 2019

Load Balancing Technique with an Efficient Method

Ranjan Kumar Mondal, Enakshmi Nandi, Payel Ray, Debabrata Sarddar

Abstract: ***There are a huge number of nodes connected to web computing to offer various types of web services to provide cloud clients. Limited numbers of nodes connected to cloud computing have to execute more than a thousand or a million tasks at the same time. So it is not so simple to execute all tasks at the same particular time. Some nodes execute all tasks, so there is a need to balance all the tasks or loads at a time. Load balance minimizes the completion time and executes all the tasks in a particular way.There is no possibility to keep an equal number of servers in cloud computing to execute an equal number of tasks. Tasks that are to be performed in cloud computing would be more than the connected servers. Limited servers have to perform a great number of tasks.We propose a task scheduling algorithm where few nodes perform the jobs, where jobs are more than the nodes and balance all loads to the available nodes to make the best use of the quality of services with load balancing.***

Keywords: Load balancing, Cloud Computing, Lowest Completion Time.

I. INTRODUCTION

Cloud Computing [1] is an Internet-based technique for the basic remote servers to maintain information and resources. Cloud Computing authorizes consumers to make use of resources with no installation and right of entry their resources from any linked computer by accessing with the Internet. This technique is not sufficient the cloud resources with storage and bandwidth centralization. Cloud Computing is a form of Internetwork computing where resources perform on the attached server rather than on a restricted node. As a general client-server model, a customer wishes to join a server to execute a task. The deviation with cloud computing is that the computing procedure may perform on more than one linked computers that can be utilized to the perception of virtualization. With virtualization, more than one servers can be put together with virtual servers, and appear to the clients to be a separate machine. The virtual server does not stay like an obtainable machine and can, so, be traveled in all the ways without affecting the purchaser. Cloud Computing is a process of distributed computing that attempts on conferring a broad compilation of users accessing to virtualized infrastructure over the web.

Revised Manuscript Received on August 30, 2019.
* Correspondence Author
Ranjan Kumar Mondal*, Computer Science & Engineering, University of Kalyani, WB, India, ranjan@klyuniv.ac.in
Enakshmi Nandi, Computer Science & Engineering, University of Kalyani, WB, India, nandienakshmi@gmail.com,
Payel Ray, Computer Science & Engineering, University of Kalyani, WB, India, payelray009@gmail.com,
Dr. Debabrata Sarddar, Computer Science & Engineering, University of Kalyani, WB, India, dsarddar1@gmail.com

It connects to the distributed computing virtualization, networking, web services, and software. The thought of cloud computing has focused on the attention of the consumers towards corresponding, distributed and virtualization computing systems these days. It has appeared as a pretty answer to supply low-priced and effortless access to externalized IT resources.

II. LOAD BALANCING

Load Balancing (LB) [2] is an online technique for distributing total workloads across total web servers, network interfaces, and other computing resources. Classic Internet datacenter implementations rely on outsized, powerful computing hardware and network infrastructure, that is a matter to the common risks connected with any machine, including power, interruptions, hardware failure, and resources inadequacies in times of high require.Load balancing can be utilized to make sure that none of the accessible resources are inactive or so busy while others are being executed. To balance loads, we are able to migrate the loads from the supply nodes to the relatively lightly loaded destination nodes.LB algorithms work in two techniques: one is cooperative meaning the nodes work concurrently to achieve the shared objective of optimizing the total response time and the other is non-cooperative saying the tasks execute individually to get better the response time of local tasks.LB algorithms can be separated into two different categories: Static Load Balancing algorithm meaning a SLAs do not take into consideration the earlier state or behavior of a node while distributing the loads and dynamic load balancing algorithm that means a dynamic load balancing algorithm ensures the previous state of a node where distributing the loads. Conversely, choosing a suitable server requires real-time communication with the other nodes on the network and generates some messages in the system. Dynamic Load Balancing algorithms use policies for keeping track of updated information.

III. TASK SCHEDULING

Task scheduling is a very important subject in cloud computing. Task Scheduling is employed to allocate definite tasks to available resources at an exacting time. A task scheduling problem is one of the major and challenging issues in cloud computing, The foremost goal of the task scheduling algorithm is to improve the appearance and QoS and at the same time retaining the good organization and justice amongst the tasks and decrease the cost of completing.
A skillful task scheduling approach should have a purpose to yield less response time so that the time off offered tasks take a position within a possible lowest time.In cloud computing systems task scheduling take part a vital function.

The scheduling of tasks cannot be performed on the source of particular factors but under a batch of rules that can term as a concurrence between consumer and traders of cloud computing. This assure is nothing but the QoS that the customer wishes that from the trades. According to the agreement suppliers always try to make available high quality of services to the customers. It is a confident job of the providers as at the same time there are a vast number of jobs performing on the merchant's part. The task scheduling complicatedness could be analyzed as the discovering a minimum mapping of a set of subtasks of multiple jobs over the current round of application (processors) such that to get the expected objectives for tasks. In this article, we are executing a relevant study of the various types of algorithms for their correctness, flexibility in the context of cloud scenario, after that we will try to propose the hybrid approach that can be accepted to improve the presented platform further. So that it could create feasible cloud providers to supply an improved quality of services.

IV. RELATED ISSUES

In this part, we explain the associated works of job scheduling in cloud computing model. In this paper [3] authors offered a brief sketch of CloudSim toolkit and it's Functionality. CloudSim is a platform where we can test our work before applied to a real job, in this current article, we have seen how to simulate a task with different approaches and different scheduling policies.In this article [4] writer proposed a method for job scheduling algorithm basis on the working loads in cloud computing. This article explaind two levels of task scheduling in the load balancing. The scheduling can not only meet users' prerequisite but endow with high resource utilization also. This article [5] described an optimization for job scheduling by genetic simulated annealing algorithm. The article considers the QoS requirements like completing time, bandwidth, cost, a distance of dissimilar type jobs. Here this annealing method is put into action after the selection, crossover, and mutation, to progress local search capacity of GA.In the article [6] hierarchical scheduling algorithm is to be discussed to help to achieve SLA with the fast response from the service broker. In this offered algorithm, QoS metric for example response time is accomplished by computing the greater precedence tasks first by estimating task completing time and the precedence tasks are generated from the enduring tasks with thetask Scheduler.In this paper [7] authors offered a minimized method for scheduling based on the Activity Based Costing. This method allocates the precedence level of jobs and costs drivers. Activity-Based Costing determines both costs of the purpose and act of the activities. In this paper [8] authors offered transaction fast cost restraint cloud workflow scheduling method. This Algorithm considers completing time as the main consideration. The method minimizes the cost under convinced consumer levels. The presented methodology is the primary basis of the computational ability of Virtual Nodes.

V. PROBLEM DEFINITIONS

To allocate a range of jobs to various nodes, in a way that the all assignment cost is to be Lowest, is known as the assignment problem.

If the number of jobs is not equal to the number of nodes, the problem is known to be unbalanced assignment problem.

Consider a matrix that arranged with a set of 'm' nodes $N = \{N_{11}, N_{12}, ..., N_{1m}\}$. A set of 'n' jobs $\{J= J_{11}, J_{12},..., J_{1n}\}$ is measured to be allocated for completing on 'm' accessible nodes. The running time of each job on all nodes is called and declared in the matrix of the order of n. The purpose of the matrix is to define the Lowest cost. This problem is explained by a well-known traditional method called *Hungarian method* [9].

Node Specification:

Every node has its meta-task that is bellowed. The following table represents the processing speed and bandwidth of N_{11} and N_{15}. Processing speed is described with MIPS (Million Instructions per Seconds), and bandwidth is specified with Mbps (Megabyte per second).

Node	Processing speed (MIPS)	Bandwidth (MBPS)
N_{11}	184	61
N_{12}	176	77
N_{13}	175	64
N_{14}	270	55
N_{15}	140	84

Table 1

Job Specification:

The following table represents instruction volume and data volume of J_{11} to J_{18}. The size of instruction is described with million instruction and data volume as MB.

Job	Instruction volume (MI)	Data size (MB)
J_{11}	7975	377
J_{12}	10754	254
J_{13}	9755	146
J_{14}	12450	227
J_{15}	8780	193
J_{16}	7946	376
J_{17}	11325	527
J_{18}	11572	485

Table 2

The completing time of the jobs of each node:

The following table calculates the completing time of each job of their corresponding node.

Job / Node	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}
J_{11}	43.34	45.31	45.57	29.53	56.96
J_{12}	58.44	61.10	61.45	39.82	76.81
J_{13}	53.01	55.42	55.74	36.12	69.67
J_{14}	67.66	70.73	71.14	46.11	88.92
J_{15}	47.71	49.88	50.17	32.51	62.71
J_{16}	43.18	45.14	45.40	29.42	56.75
J_{17}	61.54	64.34	64.71	41.94	80.89
J_{18}	62.89	65.75	66.12	42.85	82.65

Table 3

VI. PROPOSED METHOD

It is to find out the matrix value with an arrangement of the job(s) vs. node(s) of an unbalanced matrix, we would consider a problem that makes a set of 'm' nodes N={N_{11}, N_{12},..., N_{1m}}. A set of 'n' jobs J={J_{11}, J_{12},..., J_{1n}} is considered to be assigned for completion on the 'm' existing nodes and the performance value C_{ij}, wherever i=1, 2,...,m as well as j=1,2,..., n are mentioned within the value matrix wherever m is more than n. Initial of all, to tend to get the sum of every row and every column of the matrix, store the result in the array, named, Row-sum and Column-sum. After that we have to choose the first n rows by Row-sum, i.e., beginning with least to next least to the array Row-sum and deleting rows equivalent to the remaining jobs. Store the new array that ought to be the array for the primary sub-problem. Do again this method till remaining jobs less than a node once left behind jobs are but n, then, deleting columns by Column-sum, i.e., equivalent to value(s) most to next to make the last sub-problem. Accumulate the ends up in the new array that will be the array for the last sub-problem. Using the Hungarian method [9] we get the optimal result of every subproblem that is currently turning into a balanced problem. Finally, arrange every sub-problem to urge the optimum value at the side of the matrix sets.

Computational Algorithm

The algorithm represented in the current paper is to find out the following components:

– Find out criterion to allocate the excess tasks.
– Find out the procedure of assignment.
– Calculate the least cost.

Algorithm

To present an algorithmic illustration of the method, consider a problem which consists of a set of 'm' Nodes N= {N_{11}, N_{12}, ...,N_{1m}}. A set of 'n' jobs J={J_{11}, J_{12},...,J_{1n}} is considered to be alloted for completing on 'm' available nodes and the completing cost C_{ij}, where i=1,2,3..., m and j=1,2,3..., n, where m is greater than n, i.e., the number of jobs is greater than the number of nodes.

Step-1: Take Input value as an m*n matrix.
Step 2: To calculate the Column_sum of each node, in that order.
Step 3: To add dummy columns to the unbalanced matrix to do balanced matrix, adds so dummy columns (i. e. D_{11}, D_{12}, D_{13}, …. D_{1n}) based on Lowest Column_sum so matrix will be the balanced matrix.
Step 4: It is to sum all the cost of each job named Row_sum.
Step 5: Assign priority to Row_sum based on the maximum summation of the job.
Step 6: It is to discover the unassigned node having the Lowest Completion Time (LCT) for the task selected from priority one. Then, this task is forwarded to the chosen node for completing.
Step 7: It is to locate the unassigned node having the Lowest Completion Time (LCT) for the task chosen from the next priority. Then, similarly, the job is forwarded to the chosen node for completing.
Step 8: Repeat step 7 until all nodes are assigned by particular one job (i.e. one node assigned by one respective job). Remaining unassigned jobs to be allocated to the already allocated node we have to do the following:
Step 9: As all nodes assigned by jobs already, so we want to keep the Lowest cost of unassigned jobs to its used nodes. Find the Lowest cost of unassigned jobs J_{1i} of all already assigned nodes and it to the corresponding node and remove from the assigned job and the corresponding node from the list and find next Lowest cost of unassigned jobs and allocate the job to its corresponding node and so on until all jobs assigned to their corresponding node.
Step 10: Delete dummy columns after allocating jobs to original corresponding nodes.
Step 11: Stop.

VII. ILLUSTRATION

Consider a matrix in which a set of 5 nodes N = {N_{11}, N_{12}, N_{13}, N_{14}, N_{15}}, and a set of 8 jobs J = {J_{11}, J_{12}, J_{13}, J_{14}, J_{15}, J_{16}, J_{17}, J_{18}}. The assignment matrix holds the completing costs of the distinct job to the respective node.

Consider the matrix of 5*8.

J_n/M_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}
J_{11}	152	278	186	277	322
J_{12}	246	287	257	265	403
J_{13}	247	246	413	424	258
J_{14}	270	176	146	126	157
J_{15}	422	179	186	426	236
J_{16}	258	258	126	326	363
J_{17}	160	269	413	257	287
J_{18}	366	287	237	315	280

Table 4

Calculate the **Column_sum** of each node, respectively.

J_n/M_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}
J_{11}	152	278	186	277	322
J_{12}	246	287	257	265	403
J_{13}	247	246	413	424	258
J_{14}	270	176	146	126	157
J_{15}	422	179	186	426	236
J_{16}	258	258	126	326	363
J_{17}	160	269	413	257	287
J_{18}	366	287	237	315	280
Sum	**2121**	**1980**	**1964**	**2416**	**2306**

Table 5

To do m*n (where m=n) we add dummy columns based on the Lowest summation of all jobs of the respective nodes until all jobs and all nodes are equal. From the above example, we can add three columns based on lowest summation those are D_{11}, D_{12}, and D_{13}.

J_n/M_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	D_{11}	D_{12}	D_{13}
J_{11}	152	278	186	277	322	152	278	186
J_{12}	246	287	257	265	403	246	287	257
J_{13}	247	246	413	424	258	247	246	413
J_{14}	270	176	146	126	157	270	176	146
J_{15}	422	179	186	426	236	422	179	186
J_{16}	258	258	126	326	363	258	258	126
J_{17}	160	269	413	257	287	160	269	413
J_{18}	366	287	237	315	280	366	287	237

Table 6

We add costs of all nodes of each job named Row_sum.

J_n/M_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	D_{11}	D_{12}	D_{13}	Row-sum
J_{11}	152	278	186	277	322	152	278	186	**1831**
J_{12}	246	287	257	265	403	246	287	257	**2248**
J_{13}	247	246	413	424	258	247	246	413	**2494**
J_{14}	270	176	146	126	157	270	176	146	**1467**
J_{15}	422	179	186	426	236	422	179	186	**2236**
J_{16}	258	258	126	326	363	258	258	126	**1973**
J_{17}	160	269	413	257	287	160	269	413	**2228**
J_{18}	366	287	237	315	280	366	287	237	**2375**

Table 7

We make a priority based on maximum Row_sum. Highest row value gets the highest priority and the next highest value gets the next priority and so on.

J_n/M_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	D_{11}	D_{12}	D_{13}	Row-sum	Priority
J_{11}	152	278	186	277	322	152	278	186	**1831**	**7**
J_{12}	246	287	257	265	403	246	287	257	**2248**	**3**
J_{13}	247	246	413	424	258	247	246	413	**2494**	**1**
J_{14}	270	176	146	126	157	270	176	146	**1467**	**8**
J_{15}	422	179	186	426	236	422	179	186	**2236**	**4**
J_{16}	258	258	126	326	363	258	258	126	**1973**	**6**
J_{17}	160	269	413	257	287	160	269	413	**2228**	**5**
J_{18}	366	287	237	315	280	366	287	237	**2375**	**2**

Table 8

We find the Lowest cost from highest priority and find next unallocated cost from next priority and so on until all jobs are assigned to each respective nodes where each node allocated by Lowest one job and not more than two jobs.

J_n/M_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	D_{11}	D_{12}	D_{13}	Priority
J_{11}	152	278	186	277	322	152	278	186	**7**
J_{12}	246	287	257	265	403	246	287	257	**3**
J_{13}	247	246	413	424	258	247	246	413	**1**
J_{14}	270	176	146	126	157	270	176	146	**8**
J_{15}	422	179	186	426	236	422	179	186	**4**
J_{16}	258	258	126	326	363	258	258	126	**6**
J_{17}	160	269	413	257	287	160	269	413	**5**
J_{18}	366	287	237	315	280	366	287	237	**2**

Table 9

VIII. FINAL RESULT

We get the optimal cost of each node following:

N_{11}	→	$N_{11}*D_{11}$→	$J_{12}*J_{17}$	→ 246+160	→406
N_{12}	→	$N_{12}*D_{12}$→	$J_{13}*J_{15}$	→246+179	→425
N_{13}	→	$N_{13}*D_{13}$→	$J_{18}*J_{16}$	→237+126	→363
N_{14}	→	J_{11}		→277	
N_{15}	→	J_{14}		→157	

The final result is the following:

N_m	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}
Costs	406	425	363	277	157

Table 10

IX. SIMULATION RESULT:

Following chart shows the completing time of each task at different nodes.

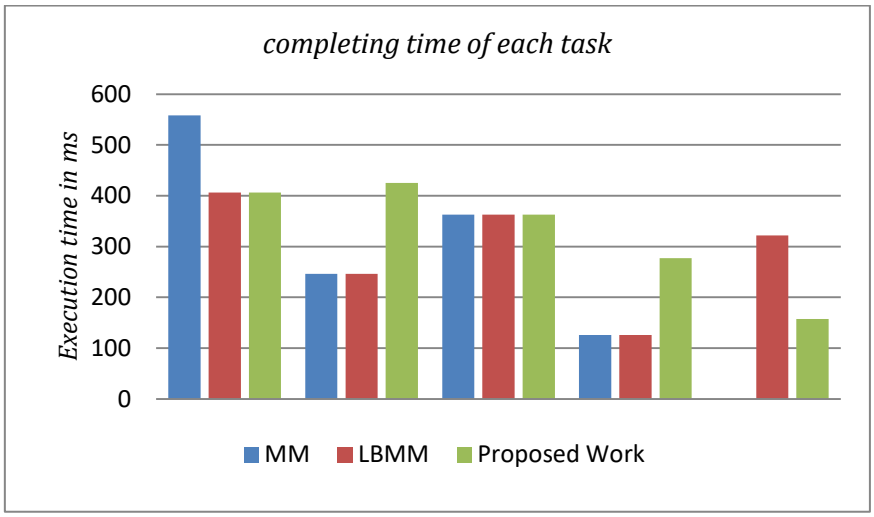

Table 1: This chart shows the completing time of each task at different nodes.

X. CONCLUSION

The jobs, those are Lowest, are assigned as dummy nodes in the matrix. The clarification of our unbalanced matrix problem, the matrix attained with the support of Max-Min is mentioned. Each subtask with least completion time is allotted to its consequent node so that result is optimal of the system. Although selected nodes execute more than one job completion time is optimal, and loads are balancing with our proposed work. Our approach shows a better result than the Hungarian method [9], Min-Min algorithm [10] and LBMM [11]. The current algorithm is offered in an algorithmic form and efforts on some input records to experiment with the act of efficiency of this algorithm. It is very easy to implement with a programming language like Java, C, etc.

REFERENCES

1. Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).
2. Ranjan Kumar Mondal, Payel Ray, Debabrata Sarddar. "Load Balancing", International Journal of Research in Computer Applications & Information Technology, Volume 4, Issue 1, January-February, 2016, pp. 01- 21, ISSN Online: 2347- 5099, Print: 2348- 0009, DOA: 03012016.
3. Buyya, Rajkumar, et. Al. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities." In High-Performance Computing & Simulation, 2009. HPCS'09. International Conference on, pp. 1-11. IEEE, 2009.
4. Jayarani, R., et. al. "Design and implementation of an efficient two-level scheduler for cloud computing environment." In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on, pp. 884-886. IEEE, 2009.
5. G. Guo-Ning and H. Ting-Lei, "Genetic Simulated Annealing Algorithm for Task Scheduling based on Cloud Computing Environment," In Proceedings of International Conference on Intelligent Computing and Integrated Systems, 2010, pp. 60-63.
6. Rajkumar Rajavel, Mala T "Achieving Service Level Agreement in Cloud Environment using Job Prioritization in Hierarchical Scheduling" Proceeding of International Conference on Information System Design and Intelligent Application, 2012, vol 132, pp 547-554.
7. Q. Cao, W. Gong, and Z. Wei, "An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing," In Proceedings of Third International Conference on Bioinformatics and Biomedical Engineering, 2009, pp. 1-3.

8. Y. Yang, Kelvin, et. Al. “An Algorithm in Swin DeW-C for Scheduling Transaction Intensive Cost Constrained Cloud Workflow,” In Proceedings of Fourth IEEE International Conference on eScience, 2008, pp. 374-375.
9. Kuhn, Harold W. "The Hungarian method for the assignment problem." Naval research logistics quarterly 2, no. 1-2 (1955): 83-97.
10. Wu, Min-You, et. al. "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems." In Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th, pp. 375-385. IEEE, 2000.
11. Patel, Gaurang, et. al. "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing." Procedia Computer Science 57 (2015): 545-553.

AUTHORS PROFILE:

Ranjan Kumar Mondal received his M.Tech in CSE from University of Kalyani, Kalyani, Nadia; and B.Tech in CSE from GCETT, Berhampore, Murshidabad, WB, India. At present, he is a Ph.D. research scholar in CSE from University of Kalyani. His research areas include Cloud Computing, Wireless, and Mobile Communication Systems.

Enakshmi Nandi received her M.Tech in VLSI and Micro-electronics from Techno India, WB and B.Tech in ECE from JIS College of Engineering, WB, India. At present, she is a Ph.D. Research scholar in CSE from University of Kalyani. Her research areas include Cloud Computing, Mobile Communication system, Device, and nanotechnology.

Payel Ray received her M.Tech in CSE from Jadavpur University, Jadavpur, and B.Tech in CSE from MCET, Berhampore, Murshdabad, WB, India. At present, she is a Ph.D. research scholar in CSE from the University of Kalyani. Her research areas include Cloud Computing, Wireless Adhoc and Sensor Network, and Mobile Communication Systems.

Dr. Debabrata Sarddar is an Assistant Professor at CSE from University of Kalyani, Kalyani, Nadia, WB, India. He completed his Ph.D. from JU. He did his M. Tech in CSE from DAVV, Indore in 2006, and his B.E in CSE from NIT, Durgapur in 2001. He published more than 100 research papers. His research areas include Cloud Computing, Wireless, and Mobile Communication System.