International Journal of Grid and Distributed Computing
Vol. 10, No. 8 (2017), pp.1-10
http://dx.doi.org/10.14257/ijgdc.2017.10.8.01

A New Effort for Distributed Load Balancing

Dr. Debabrata Sarddar and Payel Ray*

Department of Computer Science & Engineering, University of Kalyani, WB, India
dsarddar1@gmail.com, payelray009@gmail.com

Abstract

There are more than a huge number of nodes connected to web computing to provide various types of web services to provide the cloud clients. Limited numbers of nodes connected to the cloud computing have to execute more than a thousand or a million tasks at a time. So it is not so easy to execute all tasks at a 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 as well as executes all the tasks in a particular way.

There is no possibility to keep the same number servers in cloud computing to execute the same number tasks. Tasks those are to be executed in cloud computing would be more than the connected servers. Limited servers have to execute a million numbers of tasks.

We propose an algorithm that some nodes execute the jobs, here jobs are greater than the nodes and balance all nodes to maximize the quality of services with load balancing.

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

1. Introduction

Cloud Computing [1] is an Internet-based technique for the fundamental remote servers to retain information and resources. Cloud Computing permits users to utilize resources with no installation and access their resources from any connected computer by accessing the Internet. This technique set asides not sufficient the computing resources with storage and bandwidth centralization. Cloud Computing is a form of Internetwork computing where resources run on attached server rather than on a local node. As a general client-server model, a client connects to a server to execute task/tasks. The deviation with cloud computing is that the computing procedure may run 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 distinct device. The virtual server does not remain like existing device and can, therefore, be traveled in all the ways without affecting the end user. Cloud Computing is a procedure of distributed computing that efforts on conferring a broad collection of users accessing to virtualized infrastructure over the Internet. It engages distributed computing virtualization, networking, web services, and software. The thought of cloud computing has focused on the attention of the users towards of parallel, distributed and virtualization computing systems nowadays. It has appeared as an attractive solution to supply low-priced and effortless access to externalized IT resources.

2. Load Balancing

Load Balancing (LB) [2] is an online technique for distributing total workloads across total web servers, network interfaces, and other computing resources. Typical internet data center implementations rely on large, powerful computing hardware and network infrastructure, that are a matter to the common risks connected with any physical device, including power, and network interruptions, hardware failure, and resources limitations in times of high require.

ISSN: 2005-4262 IJGDC

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 can 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 attain 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 that means an 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 earlier state of a system while 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.

3. Task Scheduling

Task scheduling is a vital issue in cloud computing. Task Scheduling is used to assign definite tasks to available resources at an exacting time. In cloud computing, task scheduling problem is one of the biggest and challenging issues. The foremost target of task scheduling algorithm is to enhance the presentation and quality of service and at a time retaining the good organization and justice among the jobs and reduce the cost of completing. A proficient task scheduling approach must 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 plays a vital role. Scheduling of jobs/tasks cannot be performed on the source of particular factors but under a batch of rules that we can term as a concurrence between consumer and suppliers of the cloud computing. This promise is nothing but the QoS that the consumer desires that from the merchants. According to the agreement providers always try to provide a high quality of services to the users. It is a certain job of the suppliers as at a time there are a huge number of jobs executing on the merchant's part.

The task scheduling difficulty can be analyzed as the discovering a minimal mapping of a set of subtasks of multiple jobs over the current round of application (processors) such that we obtain the expected objectives for tasks. In this paper, we are executing a relevant study of the different algorithms for their correctness, probability, flexibility in the context of cloud scenario, after that we will try to suggest the hybrid approach that can be accepted to improve the presented platform further. So it can make possible cloud suppliers to supply a better quality of services.

4. 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 into a real job, in this current paper, we have seen how to simulate a task with different approaches and different scheduling policies.

In this paper [4] author proposed a method for job scheduling algorithm basis on the working loads in cloud computing. This article described two levels of task scheduling in the load balancing. This task scheduling can not only meet users' prerequisite but endow with high resource utilization also.

This paper [5] described an optimization for job scheduling by genetic simulated annealing algorithm. This 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 present paper [6] hierarchical scheduling algorithm is to be discussed to help to achieve Service Level Agreement (SLA) with the fast response from the service broker. In this offered algorithm, Quality of Service (QoS) metric such as 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 the help of Task Scheduler.

In this paper [7] authors offered a minimized method for scheduling based on the Activity Based Costing. This method allocates 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.

5. 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, then it 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 solved by a well-known traditional method called *Hungarian method* [9].

a. Node Specification:

Every node has its meta-task that is given bellow. 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).

Table 1

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

b. 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.

Table 2+

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

c. The Completing Time of the Jobs of each Node:

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

Table 3

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

6. Proposed Method

To find out the matrix value as well as 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}, where i=1, 2,...,m and j=1,2,..., n are mentioned within the value matrix wherever m>n. Initial of all, we have a tendency 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. Then 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 (r) jobs. Store the new array that ought to be the array for the primary sub-problem. Do this method again till remaining jobs less than a node once remaining jobs are but n, then, deleting (c) columns by Column-sum, *i.e.*, equivalent to value(s) most to next most to make the last sub-problem. Accumulate the ends up in the new array that shall be the array for the last sub-problem. Using 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 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 for assignment.

– Calculate the least cost.

Algorithm

To give 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 which are to be assigned for completing on 'm' available nodes and the completing cost is C_{ij}, where i=1,2,..., m and j=1,2,..., n, where m<n, i.e., the number of jobs is more than the number of nodes.***Step-1:*** *Take Input value as a m*n matrix.*
Step 2: *It is to calculate the* ***Column_sum*** *of each node, respectively.*
Step 3: *It is 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 balanced matrix.*
Step 4: *It is to add all 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 selected node for completing.*
Step 7: *It is to locate the unassigned node having the* ***Lowest Completion Time*** *(LCT) for the task chosen from 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 allocated by one respective job).*
Remaining unassigned jobs to be allocated to the already assigned node we have to do 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 related 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.*

7. 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 contains the completing costs of the distinct job to the respective node.

Consider the matrix of 5*8.

Table 4

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

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

Table 5

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**

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 above example, we can add three columns based on lowest summation those are D_{11}, D_{12}, and D_{13}.

Table 6

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

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

Table 7

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**

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

Table 8

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**

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.

Table 9

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**

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 following:

Table 10

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

8. Simulation Result:

The following figure demonstrates the completing time (ms) of each task at different machines. The completion times for executing all tasks by using proposed algorithm, LBMM, HM, and MM are 425, 425, 463 and 557 seconds, in that order. Our proposed algorithm achieves the minimum completion time and better load balancing than other algorithms, such as LBMM, HM, and MM in this case.

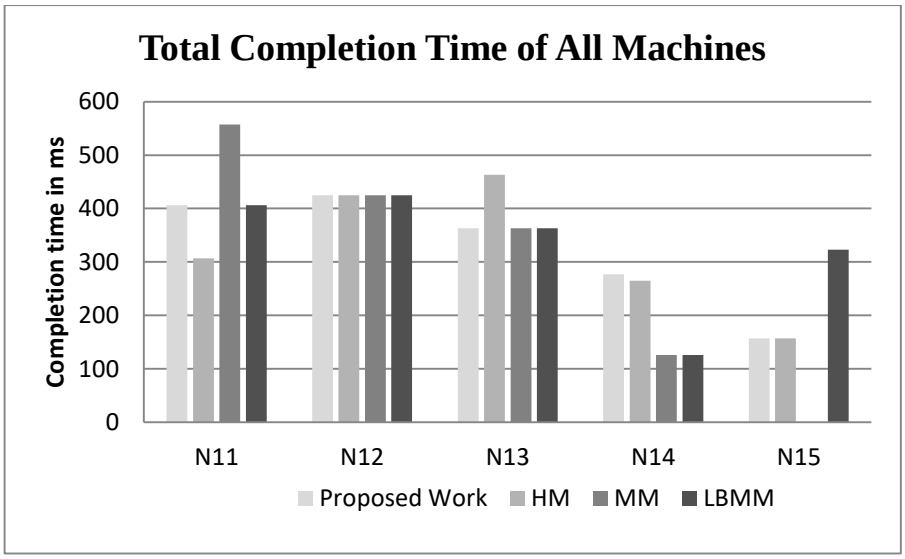

Figure 1. Completing Time (ms) of each Task at Different Computing Nodes

8. 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 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 the act of efficiency of this algorithm. It is very easy to implement with a programming language like Java, C, *etc*.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing", **(2011)**.
[2] R. Kumar Mondal, P. Ray and D. Sarddar, "Load Balancing," International Journal of Research in Computer Applications & Information Technology, vol. 4, no. 1, **(2016)** January-February,pp. 01- 21, ISSN Online: 2347- 5099, Print: 2348- 0009, DOA: 03012016.
[3] R. Buyya, R. Ranjan and R. N. Calheiros, "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, IEEE, **(2009)**, pp. 1-11.
[4] R. Jayarani, S. Sadhasivam and N. Nagaveni, "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, IEEE, **(2009)**, pp. 884-886.
[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] M. T. Rajkumar Rajavel, "Achieving Service Level Agreement in Cloud Environment using Job Prioritization in Hierarchical Scheduling", Proceeding of International Conference on Information System Design and Intelligent Application, vol. 132, **(2012)**, 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, J. Chen, X. Lin, D. Yuan and H. Jin, "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] H. W. Kuhn, "The Hungarian method for the assignment problem", Naval research logistics quarterly 2, no. 1-2, **(1955)**, pp. 83-97.

[10] M.-Y. Wu, W. Shu and H. Zhang, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems", In Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th, IEEE, **(2000)**, pp. 375-385.
[11] G. Patel, R. Mehta and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta-task scheduling in cloud computing", Procedia Computer Science, vol. 57, **(2015)**, pp. 545-553.

Authors

Dr. Debabrata Sarddar is an Assistant Professor at the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, India. He completed his Ph.D. from Jadavpur University. He did his M. Tech in Computer Science & Engineering from DAVV, Indore in 2006, and his B.E in Computer Science & Engineering from NIT, Durgapur in 2001. He has published more than 75 research papers in different journals and conferences. His research interests include Wireless and Mobile Systems and WSN, and Cloud computing.

Payel Ray received her M.Tech in Computer Science and Engineering from Jadavpur University, Jadavpur, India; and B.Tech in Computer Science & Engineering from Murshidabad College of Engineering and Technology, Berhampore, Murshidabad, West Bengal under West Bengal University of Technology, West Bengal, India. At present, she is a Ph.D. research scholar in Computer Science and Engineering from University of Kalyani. Her research interests include Cloud Computing, Wireless Adhoc and Sensor Network and Mobile Communication Systems.

