# Load Balancing Algorithm over a Distributed Cloud Network

Priyank Singhal
Student, Computer Department
Mumbai University

Sumiran Shah
Student, Computer Department
Mumbai University

Pranit Kalantri
Student, Electronics Department
Mumbai University

*Abstract*— **Due to rapidly expanding network bandwidth and hardware technologies, the Internet is developing at a rapid rate. Cloud computing, a relatively new concept, is a technique that achieves high reliability using low-power hosts. Cloud computing, an Internet-based development is dynamically scalable and often provides virtualized resources as a service over the Internet has become a significant development recently. Cloud computing essentially consists of a class of systems and applications that employ distributed resources to perform a function in a decentralized manner. The methodology of cloud computing is to utilize service nodes, often classified as computing resources on a network, to execute large and complicated tasks that require large-scale computation. Proper selection of nodes for execution of a task is thus an important factor to exploit effective use of resources. The selection must be done according to the properties of the task. This study focuses on a two level task distribution system over a three tier cloud architecture. This paper studies the use of a hybrid task scheduling algorithm which combines two commonly used scheduling methods, the MM (Min-Min) and OLB (Opportunistic Load Balancing) to create our hybrid Balanced Load Min-Min algorithm (BLMM) algorithm. This leads to more efficient execution and maintains load balancing of the system nodes.**

***Keywords- Cloud Computing; Scheduling; Load Balancing; Hybrid Algorithm, BLMM***

## I. INTRODUCTION

Today, in the age of exponentially growing internet, network bandwidth and hardware technology are advancing continuously to keep pace with it. A variety of new applications for internet users are available due to cloud computing [2][9][10]. Currently used cloud architectures consist of a large number of service nodes that collectively perform a specific service in co-ordination. Over the recent past, applications that utilize or enhance network integration have attracted considerable importance.

Cloud computing environments offer features which enable users to can access operational capabilities faster with internet applications [7]. The computer systems have a high stability to handle service requests from multiple users in the environment. However, the internet infrastructure must grow continuously at a sustainable rate so that many application services can be provided in the near future. In a distributed computing system, different components are allocated to different places or separate units such that they are connected and collectively used to an advantage. Cloud computing has in addition greatly encouraged designs of distributed systems and applications to support user-oriented services [10].

Cloud computing on an internet platform provides many applications for users including multimedia. Understanding the cloud architecture and utilizing the cloud to maximum efficiency by making sure each task obtains the resources in the shortest time is an important field of study. In this research, we investigate a hybrid two-level load balancing algorithm that combines OLB and Min-Min scheduling algorithm.

## II. EXISTING TECHNOLOGIES

Cloud Computing is a kind of distributed computing in which massively scalable IT-related services are provided to multiple external customers using internet based technologies [13]. Cloud providers need to achieve a large, general-purpose computing infrastructure and virtualization of infrastructure for different customers and services to provide multiple applications.

Information on each node affects and decides the performance of a system. Several methods including broadcasting and centralized polling exist to collect relevant information of a node like the CPU speed, transmission rate and the data rate.

Agent is one of the technologies used extensively in recent years. It's inherent navigational autonomy can be used to query other nodes for their local data In another way to put it, the agent does not need to be installed on every node the agent visits, it should collect related information of each node participating in cloud computing environment, such as CPU utilization, remaining CPU capability, remaining memory, transmission rate, etc. Thus, when the agent is deployed, it does not need any control, connection or travel flow [12]. An agent is used in this study to gather related information at low costs and high efficiencies.

There are different characteristics of each scheduling algorithm. Opportunistic Load Balancing (OLB) is an attempt to keep each node keep busy, therefore does not consider the present workload of each computer or the suitability of the node for the task. The advantage is simple, to reach load balancing but its shortcoming is to not consider expected execution time of each task, thus resulting in a higher average completion time (Turn Around time)[2][8]. In other words, OLB dispatches unexecuted tasks to currently available nodes at random order, regardless of the nodes' current workload [1][4].

Min-Min scheduling algorithm establishes the minimum completion time for every unscheduled job, and then assigns the job with the minimum completion time to the node that offers it this time [9]. Min-min uses the same mechanism as Minimum Completion Time (MCT). However, because it considers the minimum completion time for all jobs at each round, it can schedule the job that will increase the overall turn-around time the least. Thus, it helps to balance the nodes better than Minimum Completion Time.

Because of OLB scheduling algorithm, it is very simple and easy to implement to keep each computer keep busy. In this research, the OLB scheduling algorithm is used to assign the job and divide the task into subtasks in a three level cloud computing network. In addition, in order to provide the working load balance of each computer in the system, the Min-Min scheduling algorithm will be improved in this investigation in which it is expected to efficiently reduce execution time in each node.

## III. PROPOSED BLMM SCHEDULING ALGORITHM

This study proposes a slight hybrid of the Min-Min algorithm that considers Load Balancing as well. This is done to overcome the shortcoming of the Min-Min algorithm which does not consider loading at each node. This might lead to situations where some nodes are permanently idle while others are always busy. The proposed BLMM will improve the load balancing shortcomings of the Min-Min algorithm and reduce the execution time of each node effectively.

In addition, the multi-level hierarchical network topology can decrease the data store cost [5]. However, a higher lever will increase the cost of network management. Therefore, in this study, a three-lever hierarchical framework (as shown in Figure 1) is used. The third level is the service node that used to execute subtask. The second level is the service manager that used to divide the task into some logical independent subtasks. The first level is the request manager that used to assign the task to a suitable service manager.

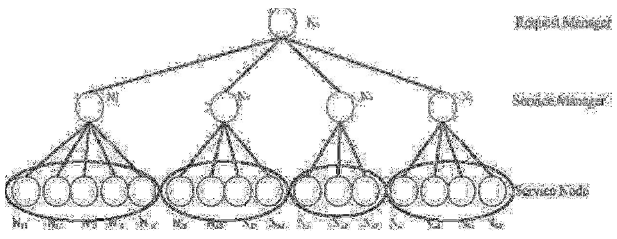

Figure 1. Three-level framework

In order to reach load balance and decrease execution time for each node in the three-level cloud computing network, the OLB and Min-Min scheduling algorithms are integrated in this study. The concept of our BLMM scheduling algorithm is to distribute task among each service manager into some subtasks to be executed in a suitable service node. BLMM considers the execution time of each subtask on each service node. Each subtask will be figured out the execution time on different service nodes (N11, N12…) via agent. According to the information gathering by agent, each service manager chooses the service node of shortest execution time to execute different subtasks and records it into the Min-time array. Finally, the Min-time array of each subtask is recorded that is a set of minimal execution time on certain service nodes. In the meanwhile, service manager chooses the service node from Min-time array. This means the $a^{th}$ subtask on service node ' g ' is performed first. Therefore, the subtask a will be distributed to service node g. Since the subtask a has been distributed to service node g to be performed, the subtask a will be deleted from subtask queue. Meanwhile, the Min-time array will be rearranged, and the service node g is put on the last one of Min-time array. In this study, an integrated scheduling algorithm is provided that combines OLB and Min-Min scheduling algorithm in a three-level cloud computing network. According to the properties of the proposed integrated scheduling algorithm, the load balance and the execution time of each node is considered.

**Step 1**: According to the requirement of subtask Ti to choose the minimal execution time service node from N service nodes, and to form a Min-time service node set, where Ti is the total number of subtasks, N is the total number of nodes inside the executable subtask service nodes set.

**Step 2**: Choose the service node ' g ' from Min-time set in which has the shortest execution time, where ' g ' is the identifier number of node.

**Step 3**: Assign subtask ' a ' to service node ' g '.

**Step 4**: Remove the complete executed subtask ' a ' from the needed to be executed task set, where ' a ' is represented the identifier of subtask.

**Step 5**: Rearrange the Min-time array, and put the service node ' g ' at the last.

Step 6: Repeat Step 1 to Step 5, until all subtasks have been executed.

## IV. THE PROPOSED METHOD

There are many heterogeneous nodes in a cloud computing system. Namely, each node has different capability to execute tasks, hence only the CPUs capable enough to perform the task are taken as candidates. Therefore, to select an efficient node to execute a task is very important in cloud computing. In order to reach the best efficiency in execution of tasks, we aim the tasks' property to adopt a different condition decision variable in which it is according to resource of task requirement to set decision variable. In this study, an agent mainly collects related information of each node participating in this cloud computing system, such as remaining CPU capability, remaining memory, and transmission rate. After all these data are collected, they will be provided to the manager and assist it in maintaining the load balancing of the system. The factors are defined as following:
V1 = Remaining CPU capability;
V2 = Remaining memory; and
V3 = Transmission rate
To make the manager select appropriate nodes effectively, all of the nodes (includes service manager and service node) in the system will be evaluated by the threshold that is derived from the demand for resource needed to execute the task. The service manager node that passes the "threshold of service manager" benchmark is considered effective, and will be the candidate of effective nodes by manager.

### A) Threshold of service manager

The cloud computing environment is composed of heterogeneous nodes, where the property of each node may greatly differ. That is, the computing capability provided by the CPU, the available size of memory, and transmission rate are different. In addition, cloud computing utilizes the resources of each node, so the available resource of each node may vary in a busy condition. From the perspective of task completion time, the available CPU capacity, the available size of memory and transmission rate is the three decisive factors for the duration of execution [11.]:
*1)* The remaining CPU capability > = 500 MB/s
*2)* The remaining memory > = 256 MB/s
*3)* Transmission Rate > = 20 MB/s

### B) Threshold of service node

When a service manager node passes the "threshold of service manager" according to the requirement of a task, then the service nodes that are managed by this service manager have the capability to execute this task. However, in a cloud computing environment, the composition of nodes is dynamic, every node is likely to enter a busy state at any time and thus increase the execution time of task which leads to lowered performance. Thus, the "threshold of service node" is used choose the better service node. The progression is divided into four steps as follows:
**Step A:** Calculate the average execution time for each subtask.
**Step B:** If the required execution time of a subtask is less than equal to the average execution time, then carry out the subtask normally.
**Step C:** If the required execution time of a subtask is greater than the average time then the executing time is set to infinity (the execution time is too long so that it should not be considered). The other nodes that had been executed will reenter into the system to participate in execution of the subtask.
**Step D:** Repeat Step A to Step C, until all subtasks have been executed completely.
In this research, the tasks can be assigned to execute quickly by the proposed integrated scheduling algorithm and the effective service nodes can be chosen by the threshold in a three-tier cloud computing network. The proposed two-phase scheduling algorithm integrates the OLB and the MM to assist in the selection for effective service nodes. First, a queue is made to store tasks that need to be carried out by manager (N0), then the OLB scheduling algorithm within "threshold of service manager" is used to assign task to the service managers in second layer (N1, N2, N3, N4, N5). However each task has different characteristics, so the restriction of node selection is also different. An agent is used to collect the related information of each node. Depending upon the property of each task, the threshold value of each node is evaluated and a service node will be assigned. However, in order to avoid the execution of certain tasks which are too long and affect system performance, "threshold of service node" is used to choose the suitable service node to execute subtask.

## V. PROOF OF ALGORITHM USING AN EXAMPLE

In this section, an example to be executed by using the proposed two-phase scheduling algorithm in a three-level cloud computing network is given. The proposed scheduling algorithm combines OLB and MM scheduling algorithm that can utilize better executing efficiency and maintain the load balancing of the system. A queue is used to store tasks that need to be carried out by manager. In the first phase, the OLB scheduling algorithm is used to assign tasks to the service manager by the manager. In the second phase, the MM scheduling algorithm is used to choose the suitable service node to execute subtask by the service manager.
The assumptions of the proposed scheduling algorithm are shown in follow:
*1)* The transmission time can be found.
*2)* The time that each job needs to carry out can be evaluated [11].
*3)* Each task can be divided into several independent subtasks, and each subtask can be executed completely by the assigned service node.

*4)* The number of service nodes is greater than or equal to the number of subtasks. An example of five tasks need to be processing is given to discuss the two-phase scheduling algorithm in a three tier cloud computing network.

**Step 1:** Task A, B, C, D and E needed to be carried out are collected and store them in the working queue by manager node N0 as shown in Figure 3. An agent is used to collect the related information of each node, as shown in TABLE I. According to the property of each task, each node is evaluated by request manager using the "threshold of service manager".

| A | B | C | D | E |
|---|---|---|---|---|

Working queue

RELATED INFORMATION OF EACH NODE

| | Remaining CPU (MB/s) | Remaining Memory (MB/s) | Transmission rate (MB/s) |
|---|---|---|---|
| $N_1$ | 503 | 456 | 29.04 |
| $N_2$ | 250 | 350 | 12.46 |
| $N_3$ | 490 | 203 | 7.09 |
| $N_4$ | 750 | 230 | 23.25 |
| $N_5$ | 128 | 322 | 21.03 |

**Step 2:** According to the threshold of the service manager, the request manager dispatches the task needed to be executed to the suitable service manager by using OLB scheduling algorithm. Therefore, the task A can be assigned to N1, N2, N3, N4, or N5 and the task B can be assigned to node N1, N2, N3, N4, or N5 (to remove the service manger node that has already carried out task A), and so on.
**Step 3:** If the "threshold of service manager" of task A is:
*i)* The remaining CPU capability > = 500 MBps
*ii)* The remaining memory > = 256 MBps
*iii)* Transmission rate > = 20 MBps
According to the information collected by agent in TABLE II, the task A will be assigned to node N1 to execute.
**Step 4:** When a task is assigned to service manager, the task will be divided into several independent subtasks by logic unit. For example, task A is divided into four subtasks.
**Step 5:** Service manager computes the execution time at different service node of each subtask by using MM as shown in TABLE II. If the subtask A1 has the minimum execution time at service node N12, then the Min-Time = (A1, N12) = 14s written. The Min-Time is an array that represents a set of minimum execution times, as shown in equation (1).

EXECUTION TIME OF EACH SUBTASK WITHIN TASK A AT DIFFERENT SERVICE NODE BEFORE DISPATCHING (FIRST)

| Node / Subtask | $N_{11}$ | $N_{12}$ | $N_{13}$ | $N_{14}$ | Threshold (Avg) |
|---|---|---|---|---|---|
| $A_1$ | 18 | 14 | 38 | 26 | 24 |
| $A_2$ | 14 | 12 | 24 | 18 | 17 |
| $A_3$ | 26 | 18 | 66 | 42 | 41 |
| $A_4$ | 19 | 20 | 24 | 36 | 24.75 |

$$\text{Min-Time} = \begin{bmatrix} A_1, N_{12} \\ \boxed{A_2, N_{12}} \\ A_3, N_{12} \\ A_4, N_{11} \end{bmatrix} = \begin{bmatrix} 14 \\ \boxed{12} \\ 18 \\ 19 \end{bmatrix}$$

**Step 6:** Service manager calculates the threshold value of each subtask (average), and compares it with the minimum execution time. In this case, the average of subtask A1 is 14 (<24), that the execution time is less than "threshold of service node", the subtask can be executed normally; average of subtask A2 is 12 (<17), i.e., the execution time is less than "threshold of service node" the subtask can be executed normally, so on.
**Step 7:** The minimum execution time of subtask is found by service manager from Min-Time array. Therefore, the corresponding subtask is A2 and the corresponding service node is N12. Therefore, subtask A2 is carried out by node N12. The subtask A2 is deleted from the set of needed to be executed subtasks, and the execution time of the remained subtasks is updated. It is indicated in TABLE III.

EXECUTION TIME OF EACH SUBTASK WITHIN TASK A AT DIFFERENT SERVICE NODE BEFORE DISPATCHING (SECOND)

| Node / Subtask | $N_{11}$ | $N_{13}$ | $N_{14}$ | Threshold (Avg) |
|---|---|---|---|---|
| $A_1$ | 18 | 38 | 26 | 24 |
| $A_3$ | 26 | 66 | 42 | 41 |
| $A_4$ | 19 | 24 | 36 | 24.75 |

**Step 8**: After Step 7, the subtask A2 is deleted from the set of subtasks and the service node N12 is ranked in last one. All executed nodes can reenter the system, when all other service nodes are assigned to work. Now, the minimum execution time (Min-Time) of each subtask is compared with the threshold value (average) by service manager. The service node set of a Min-time is shown in equation (2). The (A1, N11) is found, the corresponding subtask is A1, and the corresponding service node is N11. The subtask A1 is deleted from the set of needed to be executed subtasks, and the execution time of the remained subtasks is updated. It is indicated in TABLE IV.

$$\text{Min-Time} = \begin{bmatrix} \boxed{A_1, N_{11}} \\ A_3, N_{11} \\ A_4, N_{11} \end{bmatrix} = \begin{bmatrix} \boxed{18} \\ 26 \\ 19 \end{bmatrix}$$

EXECUTION TIME OF EACH SUBTASK WITHIN TASK A AT DIFFERENT SERVICE NODE BEFORE DISPATCHING (THIRD)

| Node / Subtask | $N_{13}$ | $N_{14}$ | Threshold (Avg) |
|---|---|---|---|
| $A_3$ | 66 | 42 | 41 |
| $A_4$ | 24 | 36 | 24.75 |

**Step 9:** After Step 8, the subtask A1 is deleted from the set of subtasks and the service node N11 is ranked in last one. In addition, the minimum execution time (Min-Time) of each subtask is compared with the threshold value (average) by service manager. However, the execution time of subtask A3 exceeds the "threshold of service node" (42 > 41), hence, the minimum execution time will be set up that is unable to carry out this subtask in the best time. The service node set of a Min-time is shown in equation (3). The (A4, N13) is found. The subtask A4 is deleted from the set of needed to be executed subtasks, and the execution time of the remaining subtasks is updated. It is indicated in TABLE V.

$$\text{Min-Time}=\begin{bmatrix} A_3, N_{14} \\ \boxed{A_4, N_{13}} \end{bmatrix}=\begin{bmatrix} \infty \\ \boxed{24} \end{bmatrix}$$

TABLE V. EXECUTION TIME OF EACH SUBTASK WITHIN TASK A
DIFFERENT SERVICE NODE BEFORE DISPATCHING (FOURTH)

| Node / Subtask | $N_{14}$ | Threshold (Avg) |
|---|---|---|
| $A_3$ | 42 | 41 |

**Step 10:** After Step 9, the subtask A4 will be deleted from set of subtasks and the service node N13 is ranked in last one. While the minimum execution time of subtask A3 exceeds the "threshold of service node", therefore, all executed service nodes will reenter the system again as shown in TABLE VI. Finally, the service node N12 will execute the subtask A3.

EXECUTION TIME OF EACH SUBTASK WITHIN TAS
DIFFERENT SERVICE NODE BEFORE DISPATCHING (FIFTH)

| Node / Subtask | $N_{11}$ | $N_{12}$ | $N_{13}$ | $N_{14}$ | Threshold (A |
|---|---|---|---|---|---|
| $A_3$ | 26 | 18 | 66 | 42 | 41 |

$$\text{Min-Time} =\boxed{[A_3, N_{12}] = [18]}$$

Thus to maintain even load balancing of the cloud computing system, the proposed two-phase load balancing algorithm can be utilized to achieve a better execution efficiency while maintaining load balancing of the system.

## V. CONCLUSION

The OLB algorithm is used in an attempt to keep each node busy and thus achieve lan even load balancing. In addition, the proposed Balanced Load Min-Min (BLMM) scheduling algorithm which is modified from Min-Min scheduling algorithm can be implemented to minimize the execution time of each task on cloud computing environment. The goal of this study is to achieve load balancing and at the same time try to achieve an optimum total execution time of all tasks. However, the load balancing of three-level cloud computing network is utilized; all calculated result can be integrated first by the second level node before sending back to the result manager. Thus, the goal of loading balance and better resources utilization can be achieved.

Lastly, in a generalized case, the cloud computing network is not only static, but also dynamic. On other hand, this proposed method can be extended to maintain and manage when the node is present in a three tier hierarchical cloud computing network in the future.

## REFERENCES

[1] R. Armstrong, D. Hensgen, and T. Kidd, "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions," 7th IEEE Heterogeneous Computing Workshop (HCW '98), pp. 79-87, 1998.

[2] F.M. Aymerich, G. Fenu and S. Surcis, "An Approach to a Cloud Computing Network," the First International Conference on the Applications of Digital Information and Web Technologies, pp. 113- 118, August 2008.

[3] T.D. Brauna, et al., "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," Journal of Parallel and Distributed Computing, Vol. 61, Issue 6, pp. 810-837, 2001.

[4] R.F. Freund, et al., "Scheduling Resources in Multi-user, Heterogeneous, Computing Environments with SmartNet," 7th IEEE Heterogeneous Computing Workshop (HCW'98), pp. 184-99, 1998.

[5] L. Garces-Erice, et al., "Hierarchical Peer-to-peer Systems," Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par), 2003.

[6] F. Halsall, Data Links, Computer Networks and Open Systems. 4th ed., Addison-Wesley Publishers, pp. 112-125, 1995.

[7] G., Ritchie and J., Levine, "A Fast, Effective Local Search for Scheduling Independent Jobs in Heterogeneous Computing Environments," Journal of Computer Applications, Vol. 25, Issue 5, pp. 1190-1192, 2005.

[8] K. Shin, et al., "Grapes: Topology-based Hierarchical Virtual Network for Peer-to-Peer Lookup Services," Parallel Processing Workshops Proceedings, pp. 159 -164, 2002.

[9] A. Vouk, "Cloud Computing- Issues, Research and Implementations," Information Technology Interfaces, pp. 31-40, June 2008.

[10] L.H. Wang, J. Tao and M. Kunze, "Scientific Cloud Computing: Early Definition and Experience," the 10th IEEE International Conference on High Performance Computing and Communications, pp. 825-830, 2008.

[11] K.Q., Yan, et al., "A Hybrid Load Balancing Policy Underlying Grid Computing Environment. Computer Standards & Interfaces, Vol. 29, Issue 2, pp 161-173, 2007.

[12] "Load Balancing, Load Balancer,"http://www.zeus.com/products/zxtmlb/index.html, January 2010.

[13] "What is Cloud Computing?," http://www.zeus.com/cloud_computing/cloud.html, January 2010.

[14] "ZXTM for Cloud Hosting Providers," http://www.zeus.com/cloud_computing/for_cloud_providers .html, January 2010