# Load Balance Enhancement in a Hierarchical Cloud Computing Environment

S.C. Wang[1], K.Q. Yan[2*], S.S. Wang [3*], C.W. Chen [4]

[1]scwang@cyut.edu.tw

[*2]kqyan@cyut.edu.tw

[*3]sswang@cyut.edu.tw

[4]s9814612@cyut.edu.tw

*Chaoyang University of Technology*

[*]Corresponding author

***Abstract*—The load balance is an important research topic in the study of distributed systems. To cope with the influence from unstable nodes, reaching a high availability before performing some special tasks is essential. Nowadays, network bandwidth and hardware technology are developing rapidly, resulting in the vigorous development of the Internet. However, cloud computing, an Internet-based development in which dynamically scalable and often virtualized resources are provided as a service over the Internet has become a significant issue. The cloud computing refers to a class of systems and applications that employ distributed resources to perform a function in a decentralized manner. Cloud computing is to utilize the computing resources (service nodes) on the network to facilitate the execution of complicated tasks that require large-scale computing. Thus, to select the service nodes for executing a task in the cloud computing needs to be considered. However, a three-phase scheduling is proposed in this study to maintain the execution performance and the load balancing of the system.**

***Keywords*—Distributed system, Cloud computing, Scheduling, Load balancing.**

## 1. INTRODUCTION

Today, network bandwidth and hardware technology advance continuously to keep pace with the vigorous development of the Internet. The new concept of cloud computing allows for more applications for internet users [2,8,10]. Cloud computing is currently used many commodity nodes that can cooperate to perform a specific service together. In addition, the internet applications are continuously enhanced with multimedia, and vigorous development of the device quickly occurs in the network system [2,11]. Thus, applications associated with network integration have gradually attracted considerable attention.

In a cloud computing environment, users can access the operational capability faster with internet application [7,15,16,18], and the computer systems have the high stability to handle the service requests from many users in the environment. However, the internet infrastructure is continuous grow that many application services can be provided in the Internet. In a distributed computing system, components allocated to different places or in separate units are connected so that they may collectively be used to greater advantage [5]. In addition, cloud computing has greatly encouraged distributed system design and application to support user-oriented service applications [10]. Furthermore, many applications of cloud computing can increase user convenience, such as YouTube [19,21].

The Internet platform of cloud computing provides many applications for users, just like video, music et al. Therefore, how to utilize the advantage of cloud computing and make each task to obtain the required resources in the shortest time is an important topic. However, in this study, a Three-Phase Scheduling (TPS) is proposed that can enhance the work efficiency and performance of cloud computing environment. Moreover, the agent mechanism is used to collect the related node information to achieve efficient utilization resource and enhance work efficiency.

The literature review is discussed in Section 2. Section 3 describes the hierarchical network topology of cloud computing. The TPS is proposed in Section 4. An example of executed

TPS is given in Section 5. Finally, Section 6 concludes this paper.

## 2. LITERATURE REVIEW

Cloud Computing is a kind of distributed computing where massively scalable IT-related capabilities are provided to multiple external customers "as a service" using internet technologies [13,21]. The cloud providers have to achieve a large, general-purpose computing infrastructure; and virtualization of infrastructure for different customers and services to provide the multiple application services. Furthermore, the ZEUS Company develops software that can let the cloud provider easily and cost-effectively offer every customer a dedicated application delivery solution [22]. The ZXTM software is much more than a shared load balancing service and it offers a low-cost starting point in hardware development, with a smooth and cost-effective upgrade path to scale as your service grows [20,22].

The ZEUS provided network framework can be utilized to develop new cloud computing methods [22], and is utilized in the current work. In this network composition that can support the network topology of cloud computing used in our study. According to the ZEUS network framework and in consequence of the properties of cloud computing structure, a three-level hierarchical topology is adopted to our investigate framework.

According to the whole information of each node in a cloud computing environment, the performance of the system will be managed and enhanced. There are several methods can collect the relevant information of node that includes broadcasting, the centralized polling and agent.

Agent is one of the technologies used extensively in recent years. It has inherent navigational autonomy and can ask to be sent to some other nodes. In other words, agent should not have to be installed on every node the agent visits, it could collect related information of each node participating in cloud computing environment, such as CPU utilization, remaining CPU capability, remaining memory, transmission rate, etc. Therefore, when agent is dispatched, it does not need any control or connection, and travel flow can be reducing in maintaining the system [9,12]. However, in this study, the agent is used to gather the related information, and reduce the resources wasting and cost.

There are different characteristics of each scheduling algorithm [6,14]. Shortest Job First (SJF) considers execution time of each program in CPU, due to SJF is selected the shortest execution time of the program, therefore, the overall average waiting time less than FCFS. However, SJF scheduling algorithm did not consider issues of importance to the programs, and that may occur Indefinite Blocking or Starvation situation [6,14].

Opportunistic Load Balancing (OLB) is to attempt each node keep busy, therefore does not consider the present workload of each computer. OLB assigns each task in free order to present node of useful .The advantage is quite simple and reach load balance but its shortcoming is not consider each expectation execution time of task, therefore the whole completion time (Make span) is very poor [1,14]. In other words, OLB dispatches unexecuted tasks to currently available nodes at random order, regardless of the node's current workload [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 [6]. Min-min uses the same mechanism as 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 make span the least. Therefore, it helps to balance the nodes better than MCT. In addition, spirit of Min-min is that every composed of the best is all minimum completion time for allocation resource.

Due to the user's requirements of cloud service cannot to be predicted, and there is per-emption implementation in connection with web services. Furthermore, the valuation services are developed in cloud computing environment, thus, the SJF scheduling is improved for per-emption issue to reduce the wait time of each user's request. Because of OLB scheduling algorithm is very simply and easy to implement and each computer often keep busy. In our research, the OLB scheduling is enhanced to assigns the job and divides the task into subtask 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 will be improved.

## 3. HIERARCHICAL CLOUD COMPUTING ENVIRONMENT

In addition, the multi-level hierarchical network topology can decrease the data store cost [3,17]. However, a higher level will increase the cost of network management. Therefore, in our study, a three-level hierarchical framework (as shown in Fig. 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.

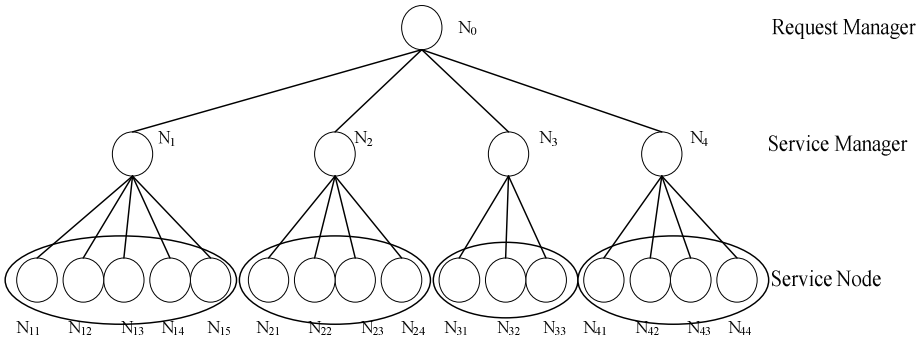


Fig. 1. Three-level framework

In our study, an integrated scheduling is provided that combines Enhanced Shortest Job First (ESJF), Enhanced Opportunistic Load Balancing (EOLB) and Enhanced Min-Min (EMM) scheduling in the hierarchical network topology of cloud computing. According to the properties of the proposed integrated scheduling, the load balance and the execution time of each node is considered.

## 4. THREE-PHASE SCHEDULING

In order to reach the load balance and reduce the execution time of each node in the cloud computing environment, a Three-Phase Scheduling (TPS) is proposed in this study. In the first phase, the ESJF (Enhanced SJF) scheduling determines the execution order for each task request. In the second phase, the EOLB (Enhanced Opportunistic Load Balancing) scheduling assigns a suitable Service Manager for allocation of the Service Node. In the third phase, the EMM (Enhanced Min-Min) scheduling guarantees that a suitable Service Node will be assigned to execute the task in the minimum execution time.

There are several heterogeneous nodes in a cloud computing system. Namely, each node has different capability to execute task; hence, only consider the CPU remaining of the node is not enough when a node is chosen to execute a task [14]. Therefore, how to select an efficient node to execute a task is very important in a cloud computing.

Due to the task has different characteristic for user to pay execution. Hence it is need some of the resources of specific, for instance, when implement organism sequence assembly; it is probable have to big requirement toward memory remaining [14]. In order to reach the best efficiency of each task, the property of each task is adopted 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:

$V_1$ = The remaining CPU capability;
$V_2$ = The remaining memory; and
$V_3$ = Transmission rate

To make the manager can select the 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 that passes the "threshold of Service Manager"

considered effective, and will be the candidate of effective nodes by manager. The Service Nodes that pass the "threshold of Service Node" considered effective, and will be the candidate of effective nodes by Service Manager.

### 4.1 Enhanced Shortest Job First

In a cloud-computing environment, the system status maybe busy or not busy. When system is not busy and there is suitable resource can provide task to execute, the FCFS, EOLB and EMM scheduling are used to schedule the nodes. Otherwise, when system is busy, the tasks are placed in the waiting queue and the ESJF, EOLB and EMM scheduling are used to arrange the executed sequence of nodes. In order to make each task can execute efficiently, the complexity of all tasks in the waiting queue need to be considered. In this study, the *Normalized Task Demand* (*NTD*) (Function (1)) is used to determine the demand of each user's services, the factors considered include the remaining CPU capability, the remaining memory and transmission rate, etc. Because each factor has different units, therefore, the factor is normalized to make the value of each factor from 0 to 1. Function (2) is an example of the remaining memory normalization.

$$NTD_i = \sum(CD_i, MD_i, TD_i, ...)/n$$

where $0<NTD_i\leq 0$
$0<CD_i\leq 0$
$0<MD_i\leq 0$
$0<TD_i\leq 0$

$n$: The total number of factors
$CD_i$: The demand CPU capability of task $i$
$MD_i$: The demand memory capability of task $i$
$TD_i$: The demand transmission rate capability of task $i$. ..…….. (1)

$MD_i$ = The demand memory capability of task $i$/The total memory capability of system
…… (2)

In this study, the ESJF is proposed to determine the execution order for each task request. According to the demand of the highest complexity task, the threshold is setting. The progressions of ESJF are divided into six steps as follow:

Step 1: The demand of each user's services *NTD* is calculated and SJF scheduling is used to select the task to execute.
Step 2: The demand of the highest complexity task in the waiting queue is chosen as the threshold.
Step 3: According to the sort results of Step 1, the distribution of the waiting tasks is carried out.
Step 4: The execution demand of each task will be cumulative.
Step 5:
  Step 5.1: When the accumulated value is greater than the threshold, the highest demand task will be executed pre-emptively.
  Step 5.2: Re-set the new threshold by using the new highest demand of task.
Step 6: Repeat Step 1 to Step 5, until all tasks dispense completely.

### 4.2 Enhanced Opportunistic Load Balancing

This study proposed an Enhanced OLB (EOLB) that integrates a traditional OLB [4,7] and a threshold mechanism of Service Manager. The Service Manager threshold is in accordance with the characteristics of a job executed by a suitable Service Manager. The proposed EOLB combines a traditional OLB and the Service Manager threshold that not only dispatches the task to the most suitable Service Manager according to the property of task requiring execution, but also maintains the advantage of traditional OLB to reach the load balance. The progressions of EOLB are divided into five steps as follow:

Step 1: To get task that has the first priority must to be executed by ESJF scheduling.
Step 2: Use OLB scheduling to select the idle nodes in the second level of cloud computing topology as the candidate Service Managers.
Step 3: One of the candidate Service Managers that conforms the threshold of the task is elected as the Service Manager.
Step 4: The task will be executed is partition into several subtasks by Service Manager.
Step 5: Repeat Step 1 to Step 4, until all tasks executed completely.

### 4.3 Enhanced Min-Min

In the third-phase, an Enhanced Min-Min (EMM) is proposed that combines a Min-Min

scheduling [4,7] and a Service Node threshold that will guarantee the task is assigned a suitable node to carry out the minimum execution time. EMM can choose the best Service Node and then uses the Service Node threshold to guarantee that the node carries out the task in the shortest time. Thus, the job can be allocated effectively and the best resource allocation provided. The EMM can be divided into four steps as follow:

Step 1: To calculate the average execution time of each subtask.
Step 2: If the required execution time of a subtask is less than or equal to the average execute time then carry out the subtask to execute normally.
Step 3:
  Step 3.1: If the required execution time of a subtask is greater than the average execute time then the executing time is set to ∞.
  Step 3.2: If the required execution time of unallocated subtasks is ∞, and if the required execution time of unallocated subtasks less or equal to the already execution time of Service Nodes then carry out the subtask to execute normally.
  Step 3.3: If the required execution time of unallocated subtasks is greater than to the already execution time of Service Nodes, the other nodes that had been executed will re-enter into the system to participate the execution of subtask.
Step 4: Repeat Step 1 to Step 3, until all tasks executed completely.

## 5. AN EXAMPLE IS EXECUTED BY TPS

In this section, an example is given by using the proposed TPS in a three-level cloud-computing network.

The proposed scheduling combines ESJF, EOLB and EMM scheduling that can utilize more better executing efficiency and maintain the load balancing of system. In the first phase, the ESJF scheduling is used to select a task in the wait queue by the Request Manager. In the second phase, the EOLB scheduling is used to assign task to the Service Manager by cloud Service Manager. In the third phase, the EMM scheduling is used to choose the suitable Service Node to execute subtask by the Service Manager.

The assumptions of the proposed scheduling are shown in follow:

1) The transmission time can be gotten.
2) The time that each job needs to carry out can be predicted [11].
3) Each task can be divided into several independent subtasks, and each subtask can be executed completely by the assigned Service Node.

An example of five tasks need to be processing is given to discuss the TPS in a three-level cloud computing network.

**Step 1:** The cloud Request Manager collects all tasks and stores in the I/O wait queue ($Q$), as shown in Table 1. The resource requirements of each task are shown in Table 2. The resources of each node in the system are shown in Table 3. In addition, the normalized resources required of each task by using function (1) are shown in Table 4.

Table 1 Task queue

| Wait Queue ($Q$) | [0] | [1] | [2] | [3] | [4] | [5] | [6] |
|---|---|---|---|---|---|---|---|
| Task | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ |
| Special Task Requirements | | | | * | | | * |

Table 2 The resources required of each task

| | Remaining Memory (MB) | Transmission Rate (KB/s) | Remaining CPU (MB/s) | Special Task Requirements |
|---|---|---|---|---|
| $T_4$ | 133.40 | 480 | 198 | * |
| $T_7$ | 119.57 | 226 | 315 | * |
| $T_1$ | 60.22 | 127 | 222 | |
| $T_2$ | 134.07 | 482 | 850 | |
| $T_3$ | 104.02 | 920 | 523 | |
| $T_5$ | 87.82 | 563 | 165 | |
| $T_6$ | 521.06 | 350 | 700 | |

Table 3 The resources reaming of each node

| | Remaining Memory (MB) | Transmission Rate (KB/s) | Remaining CPU (MB/s) |
|---|---|---|---|
| $N_1$ | 130 | 980 | 153 |
| $N_2$ | 441 | 250 | 809 |
| $N_3$ | 578 | 321 | 193 |
| $N_4$ | 491 | 120 | 987 |
| $N_5$ | 325 | 463 | 150 |

| | Remaining Memory MB) | Transmission rate (KB/s) | Remaining CPU (MB/s) | Task Demands |
|---|---|---|---|---|
| $T_4$ | 0.23 | 0.49 | 0.20 | 0.31 |
| $T_7$ | 0.21 | 0.23 | 0.32 | 0.25 |
| $T_1$ | 0.10 | 0.12 | 0.22 | 0.15 |
| $T_2$ | 0.23 | 0.49 | 0.86 | 0.53 |
| $T_3$ | 0.18 | 0.94 | 0.53 | 0.55 |
| $T_5$ | 0.15 | 0.57 | 0.17 | 0.30 |
| $T_6$ | 0.90 | 0.36 | 0.71 | 0.66 |

Table 4 The normalized resources required of each task

**Step 2:** To calculate the *NTD* of each node in waiting queue. Then, ESJF is used to sort the waiting task, as shown in Fig. 2. The maximum task demand is selected as the threshold. In addition, the minimum demand of special task is dispatched first until the special task is dispensed finish, then the general task is assigned.

| $T_7$ | $T_4$ | $T_1$ | $T_5$ | $T_2$ | $T_3$ | $T_6$ |
|---|---|---|---|---|---|---|

Fig. 2. Task waiting queue

**Step 3:** Because tasks $T_4$ and $T_7$ (Table 1) are special tasks, and the priority of $T_7$ is more than $T_4$. Therefore, $T_7$ is executed first and then $T_4$ is executed. After special tasks are executed, the general tasks will be executed. Moreover, $T_1$ will be priority to select. The summation (0.45) of the demand of $T_1$ (0.15) and $T_5$ (0.30) is less than the threshold (0.66), hence $T_5$ is the next executed task. The summation (0.98) of the demand of *0.45* and $T_2$ (0.53) is greater than the threshold (0.66), hence, $T_6$ will be the priority task. When $T_6$ has been selected, then the next highest task demand ($T_3$) is selected as the new threshold.

**Step 4:** $T_4$ must be conform to "threshold of Service Manager". If the "threshold of Service Manager" of task $T_4$ is:

*1)* The remaining CPU capability≥130 MB/s.
*2)* The remaining memory≥450 KB/s.
*3)* Transmission rate≥150MB/s.

According to the "threshold of Service Manager", the Request Manager dispatches the task needed to be executed to the suitable Service Manager by using EOLB scheduling. Therefore, the task $T_4$ can be assigned to $N_1$ $N_2$, $N_3$, $N_4$, or $N_5$ and the task $T_7$ can be assigned to node $N_1$, $N_2$, $N_3$, $N_4$, or $N_5$ (to remove the Service Manger node that has already carried out task $T_4$), and so on.

**Step 5:** When a task is assigned to Service Manager, the task will be divided into several independent subtasks by logic unit. For example, task $T_4$ is divided into four subtasks.

**Step 6:** Service Manager computes the execution time at different Service Node of each subtask by using EMM as shown in Table 5. If the subtask $T_{41}$ has the minimum execution time at Service Node $N_{54}$, then the Min-Time = ($T_{41}$, $N_{54}$) =12s written. The Min-Time is an array that represents a set of minimum execution times, as shown in equation (3).

Table 5. Execution time of each subtask within task $T_4$ at different Service Node before dispatching (First)

| Node / Subtask | $N_{51}$ | $N_{52}$ | $N_{53}$ | $N_{54}$ | Avg |
|---|---|---|---|---|---|
| $T_{41}$ | 38 | 26 | 30 | 12 | 26.5 |
| $T_{42}$ | 24 | 18 | 20 | 10 | 18 |
| $T_{43}$ | 66 | 40 | 50 | 34 | 47.5 |
| $T_{44}$ | 55 | 11 | 40 | 25 | 32.75 |

$$\text{Min-Time} = \begin{bmatrix} T_{41}, N_{54} \\ T_{42}, N_{54} \\ T_{43}, N_{54} \\ T_{44}, N_{52} \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 34 \\ 11 \end{bmatrix} \quad (3)$$

**Step 7:** Service Manager calculates the threshold value of each subtask (Avg), and compares with the minimum execution time of each subtask. In this case, The Avg of subtask $T_{41}$ is 14 (≤26.5), that the execution time is less than the "threshold of Service Node", the subtask can be executed normally; the Avg of subtask $T_{42}$ is 10 (≤18), that the execution time is less than the "threshold of Service Node", the subtask can be executed normally, and so on.

**Step 8:** The minimum execution time of subtask is found by Service Manager from Min-Time array. Therefore, the corresponding subtask is $T_{42}$ and the corresponding Service Node is $N_{54}$. Therefore, subtask $T_{42}$ is carried out by node $N_{54}$. The subtask $T_{42}$ 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 6.

Table 6. Execution time of each subtask within task $T_4$ at different Service Node before dispatching (Second)

| Subtask \ Node | $N_{51}$ | $N_{52}$ | $N_{53}$ | Avg |
|---|---|---|---|---|
| $T_{41}$ | 38 | 26 | 30 | 29 |
| $T_{43}$ | 66 | 40 | 50 | 50 |
| $T_{44}$ | 55 | 11 | 40 | 35.25 |

$$\text{Min-Time}=\begin{bmatrix} T_{41},N_{52} \\ T_{43},N_{52} \\ T_{44},N_{52} \end{bmatrix}=\begin{bmatrix} 26 \\ 40 \\ 11 \end{bmatrix} \quad (4)$$

**Step 9:** After Step 8, the subtask $T_{42}$ is deleted from the set of subtasks and the Service Node $N_{54}$ is ranked in last one. All executed nodes can re-enter 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 (Avg) by Service Manager. The Service Node set of a Min-time is shown in equation (4). The ($T_{44}$, $N_{52}$) is found, the corresponding subtask is $T_{44}$, and the corresponding Service Node is $N_{52}$. The subtask $T_{44}$ 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 7.

Table 7. Execution time of each subtask within task $T_4$ at different Service Node before dispatching (Third)

| Subtask \ Node | $N_{51}$ | $N_{53}$ | Avg |
|---|---|---|---|
| $T_{41}$ | 38 | 30 | 31.75 |
| $T_{43}$ | 66 | 50 | 52.75 |

$$\text{Min-Time}=\begin{bmatrix} T_{41},N_{53} \\ T_{43},N_{53} \end{bmatrix}=\begin{bmatrix} 30 \\ 50 \end{bmatrix} \quad (5)$$

**Step 10:** After Step 9, the subtask $T_{44}$ is deleted from the set of subtasks and the Service Node $N_{52}$ is ranked in last one. All executed nodes can renter 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 (Avg) by Service Manager. The Service Node set of a Min-time is shown in equation (5). The ($T_{41}$, $N_{53}$) is found, the corresponding subtask is $T_{41}$, and the corresponding Service Node is $N_{53}$. The subtask $T_{41}$ 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 8.

Table 8. Execution time of each subtask within task $T_4$ at different Service Node before dispatching (Fourth)

| Subtask \ Node | $N_{51}$ | Avg |
|---|---|---|
| $T_{43}$ | 66 | 60.25 |

$$\text{Min-Time} = [T_{43}, N_{51}] = [\infty] \quad (6)$$

**Step 11:** After Step 10, the subtask $T_{41}$ is deleted from the set of subtasks and the Service Node $N_{53}$ is ranked in last one. All executed nodes can re-enter 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 (Avg) by Service Manager. However, the execution time of subtask $T_{43}$ exceeds the "threshold of Service Node" (66>60.25), hence, the minimum execution time will be set up " $\infty$ " The Service Node set of a Min-time is shown in equation (6). While the minimum execution time of subtask $T_{43}$ exceeds the "threshold of Service Node", therefore, all executed Service Nodes will re-enter the system again as shown in Table 9. Finally, the Service Node $N_{54}$ will execute the subtask $T_{43}$.

Table 9. Execution time of each subtask within task $T_4$ at different Service Node before dispatching (Fifth)

| Subtask \ Node | $N_{51}$ | $N_{52}$ | $N_{53}$ | $N_{54}$ |
|---|---|---|---|---|
| $T_{43}$ | 66 | 51 | 80 | 44 |

$$\text{Min-Time} = [T_{43}, N_{54}] = [44] \quad (7)$$

To maintain the load balancing of the cloud computing, the proposed TPS can utilize more better executing efficiency and maintain the load balancing of system.

## 6. CONCLUSION

In this study, the three-phases scheduling (TPS) that included ESJF, EOLB and EMM is integrated. The allocation of tasks was based on the related information from nodes collected by the agent. The ESJF is used to reduce the waiting time for the overall tasks, and then the efficient of system is improved. The EOLB is used to

attempt each node keep busy, and the goal of load balance can be achieved. However, the proposed EMM that modified from Min-Min scheduling can make the minimum execution time of each task on cloud computing environment.

However, in this study, the TPS is employed in the hierarchical cloud-computing network. The proposed scheduling can schedule the task according to the task demand. Thus, each task will be completed effectively and quickly in the hierarchical cloud-computing network. Moreover, the load balancing of three-level cloud computing network is utilized and all calculating result could be integrated first by the second level node before sending back to the management. Thus, the goal of loading balance and better resources manipulation could be achieved.

Furthermore, in a generalized case, the cloud-computing network is not only static, but also dynamic. On other hand, our proposed method will be extending to maintain and manage when the node is underlying a dynamic hierarchical cloud-computing network in future work.

## ACKNOWLEDGMENT

This work was supported in part by the Taiwan National Science Council under Grants NSC99-2221-E324-022 and NSC97-2221-E- 324–007-MY3

## REFERENCES

[1] Armstrong, R., Hensgen, D. and Kidd, T., "The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-Time Predictions," *7th IEEE Heterogeneous Computing Workshop*, pp. 79-87, 1998.

[2] Aymerich, F.M., Fenu, G., Surcis, S., "An Approach to A cloud Computing Network," *the 1st International Conference on the Applications of Digital Information and Web Technologies*, pp. 113-118, August 4-6, 2008.

[3] Birman, K., Chockler, G., R. Renesse, "Toward A Cloud Computing Research Agenda," *ACM SIGACT News*, Vol. 40, No. 2, pp. 68-80, June 2009.

[4] Brauna, T.D., Siegelb, H.J., Beckc, N., Bölönid, L.L., Maheswarane, M., Albert, I.R., Robertsong, J.P., Theysh, M.D., Yaoi, B., Hensgenj, D. and Freundk, R.F., "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.

[5] Buyya, R., Yeo, C.S., Venugopal, S., "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," *the 10th IEEE International Conference on High Performance Computing and Communications*, pp. 5-13, September 25-27, 2008.

[6] Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D., Mirabile, F., Moore, L., Rust, B., Siegel, H.J., "Scheduling Resources in Multi-User, Heterogeneous, Computing Environments with SmartNet," *7th IEEE Heterogeneous Computing Workshop*, pp. 184-199, 1998.

[7] Grossman, R.L., Gu, Y., Sabala, M., Zhang, W., "Compute and Storage Clouds Using Wide Area High Performance Networks," *Future Generation Computer Systems*, Vol.25, No. 2, pp. 179-183, February 2009.

[8] Grossman, R.L., "The Case for Cloud Computing," *IT Professional*, Vol. 11, No. 2, pp. 23-27, March 2009.

[9] Ritchie, G., Levine J., "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.

[10] Shin, K., Lee, S., Lim, G.., Yoon, H., Ma, J. S., "Grapes: Topology-based Hierarchical Virtual Network for Peer-to-Peer Lookup Services," *Parallel Processing Workshops Proceedings*, pp. 159 -164, 2002.

[11] Vouk, M.A., "Cloud Computing- Issues, Research and Implementations," *the 30th International Conference on Information Technology Interfaces*, pp. 31-40, June 23-26, 2008.

[12] Wang, L., Tao, J., Kunze, M., Castellanos, A.C, Kramer, D., Karl, W., "Scientific Cloud Computing: Early Definition and Experience," *the 10th IEEE International Conference on High Performance Computing and Communications*, pp. 825-830, September 25-27, 2008.

[13] Weiss, A., "Computing in the Clouds," *netWorker*, Vol. 11, No. 4, pp. 16-25, December 2007.

[14] Yan, K.Q., Wang, S.C., Chang, C.P., and Lin,

J.S., “A Hybrid Load Balancing Policy Underlying Grid Computing Environment,” *Computer Standards & Interfaces*, Vol. 29, No. 2, pp. 161-173, 2006.

[15] Amazon.com: Online shopping for electronics, Apparel, Computers, Books, DVDs & More, September 2009, http://www.amazon.com/.

[16] Amazon web services, September 2009, http://aws. amazon.com/.

[17] Load Balancing and Load Balancer, September 2009, http://www.zeus.com/products/ zxtmlb/index.html

[18] More Google Product, September 2009, http://www.google.com/options

[19] Gartner Says Cloud Computing Will Be as Influential as E-business, October 2009, http://www.gartner.com/it/page.jsp?id=707508

[20] TREND MICRO, September 2009, http://us.trendmicro.com/us/home/.

[21] What is Cloud Computing?, January 2010, http://www.zeus.comlcloud_computinglc1oud.html.

[22] ZXTM for Cloud Hosting Providers, January 2010, http://www.zeus.com/c1oud_computing/for_cloud_ providers.html.