Min-Time TS: task resource mapping algorithm in cloud computing

G K Kamalam¹*, T Kalaiyarasi², S V Monaa² and B Gurudharshini²

¹Associate Professor, Dept of IT, Kongu Engineering College, India
²UG Scholar, Kongu Engineering College, India
*Corresponding author : kamalamparames@gmail.com.

Abstract. Internet-established cloud environment provides enormous amount of ubiquitous, distributed resources such as processors, memory, and software. In recent years, task scheduling and resource allocation has become a significant issue in cloud computing. Task scheduling plays a crucial role in cloud system performance. For achieving elegant performance, arises a need of efficient scheduling technique. Numerous algorithms developed for scheduling. In this paper, we have proposed an efficient approach which schedules the tasks to the available cloud resources by forming a task pair. Min-Time TS algorithm forms a finite number of task sets; each set consists a pair of tasks one from Group G1 and another from Group G2 and are scheduled with minimum layover time.

Keywords: Cloud system, Virtualization, Resource allocation, Scheduling metrics, pay-per use

1. Introduction
Cloud computing facilitates ubiquitous and dynamically scalable virtualization resources provides pay-as-you-go model. Through internet it provides global access to configurable system resources with less effort in managing the resources. The important virtue is to provide service to user virtually from anywhere. Cloud reduces the cost of the hardware, since it is scalable, increased computational power and vast storage space can be achieved. User need not purchase inexpensive hardware equipment, because the cloud provides it as pay-you-go model. It also reduces the burden of upgrading and maintaining the equipment frequently [1].

Cloud with unique features such as virtualization, scalability makes user to be flexible in accessing the required resources. Virtualization technique makes the user requirement to be fulfilled by making physical resource as virtual resource to perform their task with ease of use. According to the requirement of the individual user, the virtual resources are created. Each user has their virtual machine independently, the data storage and information processing becomes secured in the cloud environment. Cloud Service Providers (CSP) typically uses a "pay per use" model with a cloud pricing model to provide service to the cloud users. CSP have the feasibility of cost reduction using "pay per use" model.

A public-cloud delivery model, user need not purchase the software, third party named Cloud Service Provider (CSP) provide the required infrastructure for the execution of the task. The required software for cloud computing applications need not be installed on user's environment and accessed in
a distributed environment. The maintenance cost significantly reduces hardware and also software maintenance for organizations of various sizes. Dynamic provisioning of fine-grained cloud resources to nurture cloud users’ needs to satisfy peak loads can be achieved through scalability and elasticity. Dynamic provisioning leads to the scalability of cloud resources. The cloud resources can be scaled up or down based on the user’s requirements. Due to data centralization, security over the data stored in the cloud becomes a critical issue in cloud. Cloud computing integrated with web-enabled devices makes the IT professionals to work flexibly in workplace and also at home. It enables users to share documents and files over the internet [6] [11].

Research article detail is listed as: Related work is presented in Section 2. Section 3 introduces Hungarian algorithm, while the proposed Min-Time Task Scheduling (Min-Time TS) algorithm is developed in section 4. Simulation results are discussed in Section 5, conclusion of this research work is presented in section 6.

2. Related work

Dinesh and Vijayalakshmi [3], proposed an approach named SHARP for scheduling independent jobs in cloud computing. In this approach the jobs are processed in multilevel and scheduling decision is performed based on multiple criteria analysis. Dynamic provisioning of resources to jobs is performed. Dinesh and Vijayalakshmi [4], proposed Content-based Federated Job Scheduling (CFJS) algorithm. In this approach the jobs are scheduled based on the deadline of the jobs. Prasanna Kumar, and Kousalya [16], proposed crow search algorithm (CSA). Reality, the crow always follows the other mates to find the food source, similarly CSA approach identifies the suitable virtual machine for task execution. Gil-Aluja [7] has discussed the Hungarian algorithm’s (HA) theoretical elements and Konig's graph theorem. The author implemented the theorem in graph rather than matrix approach and the constraint is that each row and each column of a matrix should contain zero. Medina-Acosta, and Delgado-Penin [11] presented channel-dependent scheduling approach. Approach follows HA for achieving optimized scheduling. Bassa and Gil-Lafuente [1] proposed Customer Classification approach. It provides a basis for having strong relationships between CSP and customers. The parameters considered for the analysis are customer willingness, details of the customer, services required, treatment, etc., and implemented the classification process using HA. Nathani et al. [13] have provided allocation algorithm. It assigns the jobs based on the lease values to the clouds. The lease value is obtained from jobs start time and deadline. Shah et al. [17]) has discussed the HA’s pros and cons, and observed that the disadvantage is due to the inadequate processors compared to the jobs to be scheduled. To overcome this disadvantage, the author proposed a theoretical approach and it is impossible to be implemented. Penner et al. [14] has proposed an approach based on HA for effective task scheduling to achieve load balancing, minimizing execution time, etc. The author introduced transient cloud which forms an ad-hoc network and connects the devices located nearby to provide the services rendered by the user. Transient cloud exists when the devices are connected in the network and removed if the devices are not available.

Penner et al. [15] presented information about the demo of transient cloud in his work. The approach is implemented by Android app with Wi-Fi limited framework. The works proposed by the authors discussed so far does not consider the tasks deadline. Li et al. [9] proposed posted price model. The model allocates the resources based on the negotiation among resource provider and user. The negotiation parameters considered are duration of the service provided, QoS (Quality of Service), and cost of the service. The author performs optimized allocation of resources based on HA. The cons of this model are that it does not support huge resources and tasks. The model suits for four resources and three tasks. Chithra et al. [2] simulated an environment in which the connectivity of devices in the nearby areas performs device to device communication. The author proposed an algorithm called transmission mode assignment algorithm. The algorithm improves the throughput of the system using HA approach. Liu et al. [10], presented the FCFS method of scheduling tasks to the resources. The resource allocation is not possible based on the customer requirement.

Li et al. [8] proposed improved HA. It is used in serial-parallel systems. The author proposed the algorithm for assigning the work to the labor. The proposed approach best suits for the manufacturing systems. Mondal et al. [12] consider the problem which leads to an unbalanced assignment. The author used the HA approach to solve such problems effectively. The author tested with unequal tasks and resources and scheduling is carried out in different phases. Gawali and Shinde [6] to allocate the
resource and schedule the task efficiently proposed BandwidthAwareDS and BAR methods. Frank [5] proposed the HA-LT algorithm. HA-LT uses Hungarian algorithm and Lease Time (LT) to schedule the tasks. HA-CLT uses Converse Lease Time (CLT) and Hungarian approach for task mapping and leads to high Layover Time (LOT).

### 3. Hungarian algorithm

In this section two algorithms were discussed.

#### 3.1 HUNGARIAN ALGORITHM – LT (HA-LT)

The HA-LT algorithm obtains LT among set of tasks from G1 and G2. LT between Ti’s task in Group G1 and Tj in Group G2 is ET (Ti) - ST (Tj). For example, ET of Ti is 9:45, and the ST of Tj is 9:00. LT = 23:15. Sixty minute is presented in hundred units scale. Fifteen minute is similar to twenty five. 23:15 = 23:25. LT is converted without fractional or decimal part. 23:25 = 93. Multiplying twenty three with four and adding one to the resulting number leads to the value ninety three. HA-LT identifies minimum LT value for each task Ti in G1 to the tasks in G2 and makes the pair of tasks. The paired tasks are scheduled for the same cloud [1] [2].

#### 3.2 HUNGARIAN ALGORITHM – CLT (HA-CLT)

The HA-CLT algorithm calculates the CLT between two different groups’ tasks in the reverse order than that of the HA-LT algorithm. LT between the task Ti in G1 and Tj in G2 is calculated as the difference between ET of Tj of G2 and ST of Ti of G1. For example, ET of Tj is 10:00, and ST of Ti is 8:30, the CLT is 22:30 (i.e., 10:00 – 08:30). Here, the sixty minute is considered in hundred units scale. 30 minute is analogous of 50. 22:30 is termed 22:50. CLT is obtained on replacing sixty minute as four units. For example, 22:50 can be rewritten as 90 (i.e., 22*4 + 2). The HA-CLT algorithm identifies the minimum CLT value for each task Ti in G1 to the tasks in G2 and makes the pair of tasks. The paired tasks are scheduled for the same cloud [5] [9].

### 4. Problem statement and proposed algorithm

Task set T = {T1, T2, T3,..., Tl} and cloud set C = {C1, C2, C3,..., Cm} are considered. Tuple <ST, D, G>, represents the details of a task. ST denotes the start time of the execution of the task, D (Duration) denotes the number of hours of execution of the task, and G denotes the group the task belongs to.

Pre-determined parameters of a task are ST, D, and ET. ET is computed as the sum of the ST and D. T is divided into o = |G| groups. Each group holds equal number of tasks. Groups are disjoint. The objective is to minimize the LOT. Constraints are: 1) A task assigned to a cloud for execution. 2) Set of two tasks grouped together executes in any sequence. 3) Tasks with even count and two groups G1 and G2 are preferred for better execution. 4) Partitioning, migrating, and preempting of the tasks are not considered. The proposed algorithm (Min-Time TS) works in two phases.

#### 4.1 Phase 1 – pairing of tasks

The proposed (Min-Time TS) algorithm performs task scheduling efficiently by grouping the tasks into G1 and G2. LT is manipulated among tasks in G1 to G2. CLT is manipulated among tasks in G2 to G1. If the LT is shorter than Ti, and ET is greater than ST, then LT is reduced by 24 hours. Otherwise, LT is increased by 24 hours. LTWN is calculated from LT by assigning 4 units for an hour. Least Time, ROM (Row Order Matrix), and COM (Column Order Matrix) is computed. ROM is computed by subtracting the minimum element of each row with all the elements of the corresponding row elements and vice-versa for COM.

#### 4.2 Phase 2 – task assignment

The final task, cloud assignment is determined by calculating the TOM (Total Opportunity Matrix). The proposed (Min-Time TS) algorithm identifies the minimum value in the TOM for each task Ti in Group1 to the tasks in Group2 and makes the pair of tasks. The paired tasks are assigned to the same cloud. LOT between the paired tasks is calculated.
4.3 Illustration

The parameters of the tasks considered for simulation are Start Time (ST), Duration of the execution of task (D), and End Time of the task (ET). A sample illustration of ten tasks is taken and is shown in Table 4.1. Randomly the tasks are divided into two groups. The tasks belonging to G1 are shown in Table 4.2, and the tasks belonging to G2 are shown in Table 4.3. LT and the LTWN between the tasks of G1 and G2 are shown in Table 4.4 and Table 4.5. The CLT and the CLTWN between the tasks of G1 and G2 are shown in Table 4.6 and 4.7. LT between LTWN and CLTWN is presented in Table 4.8. The ROM, COM, and the TOM are generated and are listed in Table 4.9, 4.10, and 4.11. Finally, the task pairs and to the cloud in a group where the task is scheduled and the Table 4.12, 4.13, and 4.14 clearly depicts the LOT of the scheduling for the proposed Min-Time TS algorithm has a minimum value than the existing HA-LT, and HA-CLT algorithm.

Table 4.1. Task details.

| G | ST   | D    | ET   | Group |
|---|------|------|------|-------|
| T1 | 2:0  | 1:30 | 3:30 | 1     |
| T2 | 16:30| 1:30 | 18:0 | 2     |
| T3 | 9:30 | 1:30 | 11:0 | 2     |
| T4 | 8:30 | 1:15 | 9:45 | 1     |
| T5 | 20:30| 1:15 | 21:45| 1     |
| T6 | 17:0 | 1:30 | 18:30| 2     |
| T7 | 14:0 | 1:30 | 15:30| 2     |
| T8 | 18:30| 1:30 | 20:0 | 1     |
| T9 | 2:45 | 1:0  | 3:45 | 2     |
| T10| 13:30| 1:15 | 14:45| 2     |

Table 4.2. Group G1 tasks.

| G | ST   | D    | ET   | Group |
|---|------|------|------|-------|
| T1 | 2:0  | 1:30 | 3:30 | 1     |
| T4 | 8:30 | 1:15 | 9:45 | 1     |
| T5 | 20:30| 1:15 | 21:45| 1     |
| T8 | 18:30| 1:30 | 20:0 | 1     |
| T10| 13:30| 1:15 | 14:45| 1     |

Table 4.3. Group G2 tasks.

| G | ST   | D    | ET   | Group |
|---|------|------|------|-------|
| T2 | 16:30| 1:30 | 18:0 | 2     |
| T3 | 9:30 | 1:30 | 11:0 | 2     |
| T6 | 17:0 | 1:30 | 18:30| 2     |
| T7 | 14:0 | 1:30 | 15:30| 2     |
| T9 | 2:45 | 1:0  | 3:45 | 2     |
### Table 4.4. Lease time.

| T1→T2  | T1→T3 | T1→T6 | T1→T7 | T1→T9 |
|--------|--------|--------|--------|--------|
| 0      | 0      | 50     | 50     | 25     |
| 30:25  | 23:25  | 31:25  | 28:25  | 17:0   |
| 18:25  | 11:25  | 19:25  | 16:25  | 5:0    |
| 20:50  | 13:50  | 21:0   | 18:0   | 6:75   |
| 25:25  | 18:25  | 26:25  | 23:25  | 12:0   |

### Table 4.5. Converse lease time.

| T1→T2  | T1→T3 | T1→T6 | T1→T7 | T1→T9 |
|--------|--------|--------|--------|--------|
| 8:0    | 15:0   | 7:50   | 10:50  | 22:25  |
| 14:50  | 21:50  | 14:0   | 17:0   | 28:25  |
| 26:50  | 33:50  | 26:0   | 29:0   | 40:25  |
| 24:50  | 31:50  | 24:0   | 27:0   | 38:25  |
| 19:50  | 26:50  | 19:0   | 22:0   | 33:25  |

### Table 4.6. Whole number representation of LT

| T1→T2  | T1→T3 | T1→T6 | T1→T7 | T1→T9 |
|--------|--------|--------|--------|--------|
| 152    | 124    | 150    | 138    | 93     |
| 121    | 93     | 125    | 113    | 72     |
| 73     | 45     | 77     | 65     | 24     |
| 82     | 54     | 88     | 76     | 27     |
| 101    | 73     | 105    | 93     | 52     |

### Table 4.7. Whole number representation of CLT

| T1→T2  | T1→T3 | T1→T6 | T1→T7 | T1→T9 |
|--------|--------|--------|--------|--------|
| 32     | 60     | 30     | 42     | 89     |
| 58     | 86     | 56     | 72     | 113    |
| 73     | 21     | 53     | 41     | 0      |
| 98     | 27     | 61     | 49     | 153    |
| 78     | 106    | 76     | 88     | 133    |

### Table 4.8. Least time.

| T1→T2  | T1→T3 | T1→T6 | T1→T7 | T1→T9 |
|--------|--------|--------|--------|--------|
| 32     | 60     | 30     | 42     | 89     |
| 58     | 86     | 56     | 72     | 113    |
| 73     | 21     | 53     | 41     | 0      |
| 98     | 27     | 61     | 49     | 153    |
| 78     | 0      | 76     | 88     | 133    |

### Table 4.9. Row opportunity matrix.

| T1→T2  | T1→T3 | T1→T6 | T1→T7 | T1→T9 |
|--------|--------|--------|--------|--------|
| 2      | 30     | 0      | 12     | 59     |
| 2      | 30     | 0      | 12     | 16     |
| 49     | 21     | 53     | 41     | 0      |
| 55     | 27     | 61     | 49     | 0      |
| 26     | 21     | 24     | 36     | 0      |
Table 4.10. Column opportunity matrix.

| Task   | Group | Time |
|--------|-------|------|
| T1->T2 | G2    | 10:50 |
| T2->T4 | G2    | 14:50 |
| T5->T9 | G1    | 5:0   |
| T8->T3 | G1    | 13:50 |
| T6->T10| G2    | 19:0  |

Total layover Time: 62 Hours 30 Min

Table 4.11. Total opportunity matrix.

| Task   | Group | Time |
|--------|-------|------|
| T1->T2 | G2    | 10:50 |
| T2->T4 | G2    | 14:50 |
| T5->T9 | G1    | 5:0   |
| T8->T3 | G1    | 13:50 |
| T6->T10| G2    | 19:0  |

Table 4.12. Layover time for min-time TS algorithm for different datasets.

| Task   | Group | Time |
|--------|-------|------|
| T9->T1 | G2    | 22:25 |
| T3->T4 | G2    | 21:50 |
| T5->T7 | G1    | 16:25 |
| T8->T2 | G1    | 20:50 |
| T6->T10| G2    | 19:0  |

Total layover Time: 99 Hours 30 Min

Table 4.13. Layover time for HA-LT algorithm for different datasets.

| Task   | Group | Time |
|--------|-------|------|
| T6->T1 | G2    | 7:50 |
| T2->T4 | G2    | 14:50 |
| T5->T7 | G1    | 16:25 |
| T8->T3 | G1    | 13:50 |
| T10->T9| G1    | 12:0 |

Total layover Time: 63 Hours 45 Min

5. Simulation results
A simulation program developed simulates three algorithms, HA-LT, Min-Time TS, and HA-CLT. The program uses following simulation parameter (Table 5.1) to obtain accurate results.
5.1 Parameter evaluation
LT is defined as the inactive time of the cloud between the completion of task in G1 and G2. The proposed algorithm (Min-Time TS) efficiently processes the tasks, and schedules with minimum LT, ultimately results in to the minimization execution time of all tasks.

5.2 Results and discussion
The task assignment is done based on Min-Time TS algorithm and LT evaluated and compared among HA-LT and HA-CLT for different datasets. The comparison results specified in Table 5.2 provides a clear visualization that proposed Min-Time TS algorithm leads with better LT.

| Dataset ID | Number of tasks | Layover Time |
|------------|-----------------|--------------|
|            |                 | HA-LT (HH:MM) | HA-CLT (HH:MM) | PTS (HH:MM) |
| Dataset 1  | 10              | 79:30         | 68:15         | 64:15        |
| Dataset 2  | 20              | 114:100       | 173:15        | 113:30       |
| Dataset 3  | 30              | 206:45        | 209:15        | 160:45       |
| Dataset 4  | 40              | 280:45        | 342:45        | 259:45       |
| Dataset 5  | 50              | 367:45        | 441:15        | 319:30       |
| Dataset 6  | 100             | 746:45        | 785:30        | 579:30       |
| Dataset 7  | 200             | 1587:30       | 1459:15       | 1279:30      |
| Dataset 8  | 300             | 2288:00       | 2265:30       | 1742:15      |

The graphical representation of the comparison of proposed Min-Time TS, existing algorithms is given in Fig. 5.1. It’s clear from Fig. 5.1 that the proposed algorithm (Min-Time TS) gives a minimum LOT (Layover Time) than HA-LT and HA-CLT.
6. Conclusion

The proposed algorithm (Min-Time TS) schedules the task efficiently to the appropriate cloud by pairing of tasks among two groups. Using different datasets proposed Min-Time Task Scheduling (Min-Time TS) algorithm is compared with HA-LT and HA-CLT. From the experimental result analysis it is clearly evident that proposed Min-Time TS algorithm produces better (minimum) layover time compared to existing algorithms. Proposed Min-Time TS algorithm confines the task groups to be two. Future research direction will intrigue in providing a scheduling algorithm to address this issue.

REFERENCES

[1] C.Bassa and Gil-Lafuente,"The Hungarian algorithm for specific customer needs”, Soft Comput. Manage. Business Econ. Stud. Fuzziness and Soft Comput. Springer, pp. 363-379, 2012.

[2] R.Chithra, R.Bestak, and S.Patra, “Hungarian method based joint transmission mode and relay selection in device-to-device communication”, In: 8th IFIP Wireless and Mobile Networking Conference, IEEE. pp. 261-268, 2015.

[3] K.Dinesh, and M.Vijayalakshmi, “ScHeduling of jobs and Adaptive Resource Provisioning (SHARP) approach in cloud computing”, Cluster Computing, Springer, Vol.21(1), pp.163-176, 2018.

[4] K.Dinesh, and M.Vijayalakshmi, “Content-Based Federated Job Scheduling Algorithm in Cloud Computing”, Australian Journal of Basic and Applied Sciences, Vol. 10(2), pp. 52-59, 2016.

[5] A.Frank,”On Kuhn’s Hungarian method - a tribute from Hungary”, Naval Res. Logist. vol.52, pp. 2-5, 2005.

[6] M.Gawali and S.Shinde, “Task scheduling and resource allocation in cloud computing using a heuristic approach”, Journal of Cloud Comput. Adv. Syst. Appl, Springer vol.7, pp.1-16, 2018.

[7] J.Gil-Aluja, “Theoretical elements of the Hungarian algorithm”, In: The Interactive Management of Human Resources in Uncertainty, Applied Optimization, Springer, pp.158-170, 1998.

[8] T.Li, Y.Li, and Y.Qian, “Improved Hungarian algorithm for assignment problems of serial-parallel systems”, J. Syst. Eng. Electron. IEEE, vol.27, no.4, pp.858-70, 2016.

[9] M.Li, N.Xiong, B.Yang, Z.Li, J.Park, and C.Lee, “Posted price model based on GRS and its optimization for improving grid resource sharing efficiency”, Telecommun. System, Springer, vol. 55, no.1, pp.71-79, 2014.

[10] X.Liu, Y.Zha, Q.Yin, Y.Peng, and L.Qin, “Scheduling parallel jobs with tentativemins and consolidation in the cloud”, J. Syst. Software, Elsevier, vol.104, pp.141-151, 2015.

[11] G.Medina-Acosta, and J.Delgado-Penin, “On the feasibility of a channel-dependent scheduling for the SC-FDMA in 3GPP-LTE (mobile environment) based on a prioritized-bifacet Hungarian method”, EURASIP J. Wireless Commun. Networking, Springer, vol.71, pp.1-10, 2011.

[12] R.Mondal, P.Ray, E.Nandi, B.Biswas, M.Sanyal, and D.Sarddar, “Load balancing of unbalanced matrix with Hungarian method”, In: International Conference on Computational Intelligence, Communications and Business Analytics, Communications in Computer and Information Science, Springer, pp.256-270, 2017.

[13] A.Nathan, S.Chaudhary, and G.Somani, “Policy based resource allocation in IaaScloud”, Future Gener. Comput. System, Elsevier, vol.28, pp.94-103, 2012.

[14] T.Penner,A.Johnson, B.Slyke, M.Guirguis, and Q.Gu, “Demo: transient clouds”, In: 6th International Conference on Mobile Computing, Applications and Services, IEEE, pp.153-154, 2014.

[15] T.Penner,A.Johnson, B.Slyke, M.Guirguis, and Q.Gu, “Transient clouds: assignment and collaborative execution of tasks on mobile devices”, In: IEEE Global Communications Conference, pp. 2801-2806, 2014.

[16] K.Prasanna Kumar, and K.Kousalya, “Amelioration of task scheduling in cloud computing using crow search algorithm”, Neural Computing and Applications, Springer, Vol.32(10), pp. 5901-5907, 2020.

[17] K.Shah, P.Reddy, and S.Vairamuthu, “Improvement in Hungarian algorithm for assignment problem”, In: International Conference on Artificial Intelligence and Evolutionary Algorithms in Engineering Systems, Advances in Intelligent Systems and Computing, Springer, pp.1-8, 2014.