Asian Journal of Information Technology 15 (13): 2147-2153, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

# Sort Completion Time Mean Tasks Scheduling Algorithm in Decentralized Grid Environment

G.K. Kamalam
Department of Information Technology, Kongu Engineering College, Perundurai, Erode, Iran

**Abstract:** Scheduling task on a heterogeneous distributed grid environment is a challenging one and in general, has been shown to be a NP-Complete problem. To solve the complicated scheduling problem, an efficient scheduling algorithm is essential. The major aim of the scheduling algorithm is in maximizing the resources utilization and in minimizing the makespan. This study addresses the complicated scheduling problem by proposing an efficient scheduling algorithm, Sort Completion Time Mean Tasks Scheduling algorithm (SCTMTS) which determines the order in which the tasks are to be scheduled to the appropriate resources. The first step is to sort the list of completion time of the task. The second step is to find the mean value of each task. Then the maximum mean value is obtained. Finally, the task that has the maximum mean value is selected and scheduled to the resource that has the minimum completion time. Experimental results reveal that the proposed sort completion time mean tasks scheduling algorithm outperforms min-min heuristic scheduling algorithm regarding both makespan and resource utilization.

**Key words:** Task scheduling, heterogeneous resources, NP-complete, bandwidth, makespan

## INTRODUCTION

A growing emphasis on the development of networking topology led to the possibility of the interconnection of diverse set of resources in distributed environment provides a computing platform capable of executing scientific and engineering applications that require large computing power. Heterogeneous computing systems requires efficient scheduling for executing applications with reduced makespan and provide efficient utilization of resources (Anousha and Ahmadi, 2013).

The mapping of the tasks to the appropriate resource in the heterogeneous grid system to achieve reduced makespan, i.e., the problem of optimal scheduling has been proven to be NP-Complete (Foster and Kesselman, 1999; Kamalam and Bhaskaran, 2010a, b; Kamalam and Bhaskaran, 2012a, b). The NP-complete problem could give an optimal solution in polynomial time but approximate algorithms or heuristic approach is said to be an acceptable solution for the scheduling problems in real time applications. Opportunistic Load Balancing (OLB) assigns each task in random order to the available resource without considering the expected execution time of the task on that resource. The algorithm is very simple but the disadvantage is that is does not consider the expected execution time of the task and the task mapping to the resources results in very poor makespan. The aim of the algorithm is to keep all the resources as busy as possible (Armstrong *et al*., 1998; Freund and Siegel, 1993; Freund *et al*., 1998; Maheswaran *et al*., 1999).

Minimum Execution Time (MET) assign each task in random order to the resource with the minimum execution time for that task. The algorithm does not consider the ready time of the resources and causes server imbalance across the resources. The aim of the algorithm is to give each task to its best resources (Armstrong *et al*., 1998; Freund *et al*., 1998).

Minimum Completion Time (MCT) assigns each task in random order to the resource with the minimum expected completion for that task. The minimum expected completion time is computed as the addition of the ready time for the resource and the minimum expected execution time for that task on that particular resource. The aim of the algorithm is to combine the advantages of OLB and MET. Min-min heuristic scheduling starts with scheduling the set of all unmapped tasks U. The minimum completion time for each task is calculated. The task with the overall minimum completion time is selected and mapped to the corresponding resource. The mapped task is removed from the task set U and the algorithm repeats the process until all tasks from the task set U are mapped (Armstrong *et al*., 1998; Freund *et al*., 1998; Ibarra and Kim, 1977; Maheswaran *et al*., 1999). Max-min heuristic scheduling algorithm is similar to Min-min algorithm. The algorithm starts with scheduling the set of all unmapped tasks U. The minimum completion time for each task is calculated. The task with the overall maximum completion time is selected and mapped to the corresponding

resource. The mapped task is removed from the task set U and the algorithm repeats the process until all tasks from the task set U are mapped (Armstrong *et al*., 1998; Freund *et al*., 1998; Ibarra and Kim, 1977).

Among the above algorithms, Min-min is the fastest algorithm (Kamalam and Bhaskaran, 2010a,b). Min-min algorithm executes all small tasks first and then executes the long task. The disadvantage is that, it does not balance the load well across the resource because it schedules the small task first. Considering the advantage and disadvantage of Min-min algorithm, this study proposed a new heuristic scheduling algorithm sort completion time mean tasks scheduling algorithm that provides reduced makespan and improves resource utilization rate.

A decentralized grid system model comprising of clusters, cluster servers are treated as Coordinator Nodes (CN). Each cluster consists of multiple Worker Nodes (WN). Each worker node has different computing powers. The reschars presented a dynamic load balancing algorithm for scheduling the tasks. The CN checks whether the cluster is overloaded or not. If it is overloaded, CN upon receiving information from Grid Information Centre (GIC) and CN select the cluster which is under loaded and schedules the task to the corresponding WN (Suri and Singh, 2010).

RASA heuristic algorithm schedules the task based on the completion time of each task on every resources. The heuristic approach involved in this algorithm is to schedule the tasks to the appropriate resources using Min-min heuristic algorithm if the available number of resources is odd, otherwise the tasks are scheduled to the appropriate resource using Max-min heuristic algorithm (Parsa and Entezari-Maleki, 2009). In the previous study, we proposed a decentralized grid architecture model as a group of clusters. Each cluster includes a multiple number of WN. This study proposed a Decentralized Hybrid Job Scheduling Algorithm (DHJSA), the scheduling of jobs to the appropriate cluster and the appropriate WN is done by applying divisible load theory and least cost method. The algorithm suits well for computational jobs (Kamalam and Bhaskaran, 2011.

In the previous study, we proposed a Novel Adaptive Decentralized Job Scheduling Algorithm (NADJSA) (Kamalam and Bhaskaran, 2012a, b). The algorithm divides the jobs into subjobs and schedules the subjobs based on the minimum completion time for each task to the appropriate cluster and the appropriate WN.

## MATERIALS AND METHODS

**Decentralized grid architecture model:** A decentralized Grid system comprising of a collection of clusters where clusters are named as Coordinator Nodes (CN) represented as CN = {$CN_1$, $CN_2$,..., Cnm} where m-represents the number of clusters. Every cluster comprises of collection of Worker Nodes (WN) represented as WN = {$WN_1$, $WN_2$, $WN_3$,..., Wnn} where n-represents the number of worker nodes in the cluster. The clusters are interconnected through a wide area network. Grid Information Center (GIC) provides the computing power, memory utilization and bandwidth of all the Worker Nodes in the decentralized grid environment. CN of each cluster periodically provides this information to GIC. The user submits the tasks to the grid scheduler. The set of tasks is represented as T = {$T_1$,$T_2$,..., $T_x$} where x-represents the number of tasks to be executed.

**Proposed SCTMTS algorithm:** Task is as entity which contains information such as length expressed in MI (Million Instructions), Input File Size (IFS) expressed in MB, Output File Size (OFS) expressed in MB. This characteristic of a task is used to estimate the expected execution time and data transfer time. The expected execution time and the data transfer time of the tasks are considered as a factor in selecting the appropriate worker node for the tasks.

Grid is a heterogeneous system. The worker nodes contain information such as processing power expressed in MIPS (Million Instructions per Second) and bandwidth expressed in MBPS (Mega Bytes per Second). This information is used to calculate the time taken for the execution of a task in a worker node and also used to compute the time taken to transfer the input file and output file to and from the worker node and grid scheduler respectively. Scheduling independent task in a grid environment has been shown to be a NP-complete problem. A grid scheduler selects the best worker node with minimum completion time, (i.e., minimum (expected execution time+waiting time+data transfer time)) and submits the task to the selected worker node.

The main goal of the proposed sort completion time mean tasks scheduling algorithm is to maximize the resource utilization and to minimize the makespan. Makespan is defined as the overall completion time of all the tasks. The steps involved in the proposed algorithm to assign each task to a best worker node are summarized below:

- Step 1: Users submits the tasks to the grid scheduler and adds the tasks to the task set T
- Step 2: The expected execution time and the data transfer time for each task $T_i$ on every Worker Node $WN_j$ in all clusters is calculated
- Step 3: Sort the Compilation Time ($CT_i$) of each task $T_i$ in T in increasing order

Table 1: Parameters

| Variables | Descriptions |
|---|---|
| $EET_{Kij}$ | Expected Execution Time of the task $T_i$ on a Worker Node $WN_j$ in a cluster ($C_K$) |
| $ECT_{Kij}$ | Expected Completion Time of the task $T_i$ on a Worker Node $WN_j$ in a cluster ($C_K$) |
| $AT_{Kj}$ | Availability Time of a Worker Node $WN_j$ in a cluster $C_K$ after having completed the previously assigned tasks |
| $DTT_{Kij}$ | Data Transfer Time to Worker Node $WN_j$ in a cluster ($C_K$) |
| $Length_i$ | Length of the task $T_i$ expressed in MI |
| | Processing $power_{Kj}$ computing power of the Worker Node $WN_j$ in a cluster $C_K$ expressed in MI |
| $BW_{Kj}$ | Bandwidth of the Worker Node $WN_j$ in a cluster ($C_K$) expressed (MBPS) |
| $IFS_i$ | Input File Size of the task $T_i$ (MB) |
| $OFS_i$ | Output File Size of the task $T_i$ (MB) |
| MV[i] | Mean completion time of the task ($T_i$) |

- Step 4: The Mean Value (MV) of the two consecutive completion times for each task $T_i$ is computed
- Step 5: The task having the maximum MV is selected
- Step 6: The selected task is assigned to the Worker Node possessing minimum completion time
- Step 7: The assigned task is deleted from the task set T
- Step 8: Finally, the waiting time for the Worker Node that executes the task is updated
- Step 9: Step 3-8 are repeated until all x tasks are scheduled

Table 1 shows the parameters that are used in the proposed algorithm for efficient task scheduling. The expected execution time of a task $T_i$ on a worker node $WN_j$ in a cluster $C_K$ is calculated as follows:

$$EET_{Kij} = \frac{Length_i}{Processing\ Power_{Kj}} \quad (1)$$

The data transfer time for a worker node $WN_j$ in a cluster $C_K$ is computed as follows:

$$DTT_{Kij} = \frac{IFS_i}{BW_{Kj}} + \frac{OFS_i}{BW_{Kj}} \quad (2)$$

The expected completion time of a task $T_i$ on a worker node $WN_j$ in a cluster $C_K$ is calculated as follows:

$$ECT_{Kij} = EET_{Kij} + AT_{Kj} + DTT_{Kij} \quad (3)$$

Makespan is computed as follows:

$$Makes\ pan = Max\left(ECT_{Kij}\right), \quad 1 \le i \le x, 1 \le j \le n, 1 \le k \le m \quad (4)$$

Sort the completion time of each task $T_i$ in T in increasing order and store it in a two dimensional array Y. The mean value of the two consecutive completion times for each task $T_i$ which is used in determining the order in which the task to be selected for scheduling and the formula for finding the mean value is computed as follows:

$$MV_j = \frac{\left(Y_j[r][s] + Y_{(i+1)}[r][s]\right)}{2} \quad (5)$$

Where:

j = [n/2]

r = The tasks

s = The number of worker nodes in all clusters

**Sort Completion Time Mean Tasks Scheduling Algorithm (SCTMTS):**

Input: Number of Clusters m, number of Worker Nodes n, number of Tasks x, characteristics of tasks, characteristics of resources.

Output: The result of the mapping function-task and the appropriate resource pair, makespan.

Begin

Initialize makespan=0

while there are tasks in the unassigned task set T

do

for all submitted tasks in the task set T

for all clusters $C_m$

for all Worker Nodes in each cluster K

do

$$EET_{Kij} = \frac{Length_i}{Processing\ power_{Kj}}$$

$$DTT_{Kij} = \frac{IFS_i}{BW_{Kj}} + \frac{OFS_i}{BW_{Kj}}$$

$AT_{kj} = \sum_{i=1}^{n} EET_{kij}$ where n-represents the number of tasks that are already in the queue

$ECT_{Kij} = EET_{Kij} + AT_{Kj} + DTT_{Kij}$

end for

$Y[r][s] = ECT_{kij}$

s = s+1

where r-represents the tasks , s-represents the total number of Worker Nodes in all clusters

end for

r = r+1

end for

Sort each row in two dimensional array Y

for each tasks r in two dimensional array Y

Compute

$$MV_j = \frac{\left(Y_j[r][s] + Y_{(i+1)}[r][s]\right)}{2}$$

where j=[n/2] (i.e. jth& j+1th position in a two dimensional array Y)

end for

Identify the maximum value in array MV

Select the task $T_i$ with the maximum value in array MV

Assign the selected task $T_i$ to the Worker Node $WN_j$ in a Cluster K with minimum completion time $ECT_{Kij}$

Table 2: Processing power and Bandwidth of worker node

| | Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|---|
| Characteristics | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ |
| Processing Power | 5000 | 5400 | 4800 | 4200 | 5800 | 6000 |
| Bandwidth | 350 | 350 | 200 | 200 | 500 | 500 |

Table 3: Task characteristics

| Characteristics | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| Length | 50000 | 60000 | 75000 |
| IFS | 500 | 600 | 1000 |
| OFS | 700 | 600 | 1500 |

Table 4: EET, DTT and ECT for all tasks

| | Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|---|
| Tasks | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ |
| EET | 10 | 9.26 | 10.42 | 11.90 | 8.62 | 8.33 |
| DTT | 3.43 | 3.43 | 6 | 6 | 2.4 | 2.4 |
| ECT | 13.43 | 12.69 | 16.42 | 17.9 | 11.02 | 10.73 |
| EET | 12 | 11.11 | 12.5 | 14.29 | 10.34 | 10 |
| DTT | 3.42 | 3.42 | 6 | 6 | 2.4 | 2.4 |
| ECT | 15.42 | 14.53 | 18.5 | 20.29 | 12.74 | 12.4 |
| EET | 15 | 13.89 | 15.63 | 17.86 | 12.93 | 12.5 |
| DTT | 7.15 | 7.15 | 12.5 | 12.5 | 5 | 5 |
| ECT | 22.15 | 21.04 | 28.13 | 30.36 | 17.93 | 17.5 |

Table 5: Ascending order of all tasks based on ECT

| | Cluster 3 | | Cluster 1 | | Cluster 2 | |
|---|---|---|---|---|---|---|
| Tasks | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_1$ | $WN_2$ |
| 1 | 10.73 | 11.02 | 12.69 | 13.43 | 16.42 | 17.9 |
| 2 | 12.4 | 12.74 | 14.53 | 15.42 | 18.5 | 20.29 |
| 3 | 17.5 | 17.93 | 21.04 | 22.15 | 28.13 | 30.36 |

Update $AT_{Kj}$

If $\text{makespan} \leq ECT_{kij}$

$\text{makespan} = ECT_{kij}$

Remove the task $T_i$ from the task set T

end while

end

**An illustrated example:** The following example clearly shows that the proposed sort completion Time Mean Tasks Scheduling algorithm schedules tasks with reduced makespan compared to that of the Min-min heuristic scheduling algorithm. Consider the grid environment with three clusters, two Worker Nodes in each cluster and three tasks. The Processing Power and Bandwidth of Worker Node in each cluster are shown in Table 2. The length, IFS, OFS of each task $T_i$ is shown in Table 3.

Initially, the EET, DTT and ECT for all tasks in all Worker Nodes are calculated and are shown in Table 4. At first iteration, sort each tasks in the ascending order of ECT and is shown in Table 5. Calculate the Mean Value of each task:

$$MV(T1) = (12.69+13.43)/2 = 13.06$$

$$MV(T2) = (14.53+15.42)/2 = 14.98$$

$$MV(T3) = (21.04+22.15)/2 = 21.7$$

Table 6: Availability time of $WN_2$ of Cluster 3

| | Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|---|
| Tasks | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ |
| EET | 10 | 9.26 | 10.42 | 11.90 | 8.62 | 20.83 |
| DTT | 3.43 | 3.43 | 6 | 6 | 2.4 | 2.4 |
| ECT | 13.43 | 12.69 | 16.42 | 17.9 | 11.02 | 23.23 |
| EET | 12 | 11.11 | 12.5 | 14.29 | 10.34 | 22.5 |
| DTT | 3.42 | 3.42 | 6 | 6 | 2.4 | 2.4 |
| ECT | 15.42 | 14.53 | 18.5 | 20.29 | 12.74 | 24.9 |

Table 7: Ascending order of all tasks based on ECT

| | Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|---|
| Tasks | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ |
| 1 | 11.02 | 12.69 | 13.43 | 16.42 | 17.9 | 23.23 |
| 2 | 12.74 | 14.53 | 15.42 | 18.5 | 20.29 | 24.9 |

Table 8: Availability time of $WN_1$ of Cluster 3

| | Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|---|
| Tasks | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ |
| EET | 10 | 9.26 | 10.42 | 11.90 | 18.96 | 20.83 |
| DTT | 3.43 | 3.43 | 6 | 6 | 2.4 | 2.4 |
| ECT | 13.43 | 12.69 | 16.42 | 17.9 | 21.36 | 23.23 |

Table 9: A comparison between Min-min and proposed Sort Completion Time Mean Tasks Scheduling Algorithms in makespan and Tasks Scheduling

| | Cluster 1 | | Cluster 2 | | Cluster 3 | | |
|---|---|---|---|---|---|---|---|
| Algorithms | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | $WN_1$ | $WN_2$ | Makespan |
| Min-min | -- | T3 | -- | -- | T2 | T1 | 21.04 |
| SCTMTS | -- | T1 | -- | -- | T2 | T3 | 17.5 |

The maximum mean value is 21.7. The selected task is $T_3$. Task $T_3$ is scheduled to $WN_2$ of cluster 3. The ECT is 17.5. Makespan is 17.5. At the second iteration, the availability time of $WN_2$ of cluster 3 is updated and is shown in Table 6. Sort each tasks in the ascending order of ECT and is shown in Table 7. Calculate the mean value of each task:

$$MV (T1) = (13.43+16.42)/2 = 14.93$$

$$MV (T2) = (15.42+18.5)/2 = 16.96$$

The maximum mean value is 16.96. The selected task is $T_2$. Task $T_2$ is scheduled to $WN_1$ of cluster 3. The ECT is 12.74. Makespan =17.5. At the third iteration, the availability time of $WN_1$ of cluster 3 is updated and is shown in Table 8. Now, the task $T_1$ is scheduled to $WN_2$ of cluster 1. The ECT is 12.69. Finally, makespan = 17.5. As a result, the makespan of the above example equal Max (17.5, 12.74, 12.69) = 17.5. The makespan produced by Min-min heuristic scheduling algorithm compared to the result of the proposed sort completion time mean tasks scheduling algorithm is shown in Table 9.

## RESULTS AND DISCUSSION

The performance of the proposed algorithm is evaluated using the parameter makespan. The result of the

proposed sort completion time mean tasks scheduling algorithm is analyzed and compared with Min-min Heuristic Scheduling algorithm. The simulation results have been taken for various test cases by varying the number of tasks to be scheduled. The proposed sort completion time Mean Tasks Scheduling algorithm generates reduced makespan for all the test cases. Figure 1 shows the performance of the proposed sort completion time mean tasks scheduling algorithm over the existing Min-min heuristic scheduling algorithm for varying number of tasks. The performance of the proposed sort completion time Mean Tasks Scheduling algorithm is measured based on the simulation parameters shown in Table 10.

Selecting the order of tasks to be scheduled and selecting the appropriate Worker Node for a particular task is one of the challenging works in distributed grid environment. This study presents the experimental results obtained for the benchmark model by Braun *et al*., 1999 [2,3] for distributed heterogeneous grid environment.

**Benchmark descriptions:** To evaluate the proposed algorithm, the benchmark model instances are divided into twelve different types of ETC matrices. The size of the ETC matrix is 51216 where 512 represent the number of tasks and 16 represents the number of resources. Twelve combinations of ETC matrices were based on the three metrics: Task heterogeneity, Resource heterogeneity and Consistency. For each twelve different type of ETC matrix, the results were averaged over 100 different ETC matrices of the same type. The benchmark instances are labelled as u-x-yyzz.k where; u-uniform distribution in generating ETC matrices; x-consistency(c-consistent,i-inconsistent, s-semi-consistent or partially consistent). An ETC matrix is said to be consistent if a machine $m_j$ executes any task $t_i$ faster than resource $r_k$, then resource $r_j$ executes all tasks faster than resource $r_k$. An ETC matrix is said to be inconsistent if a resource $r_j$ executes some tasks faster and some tasks slower than resource $r_k$. Semi-consistent ETC matrices are the matrices that includes a consistent sub-matrix.

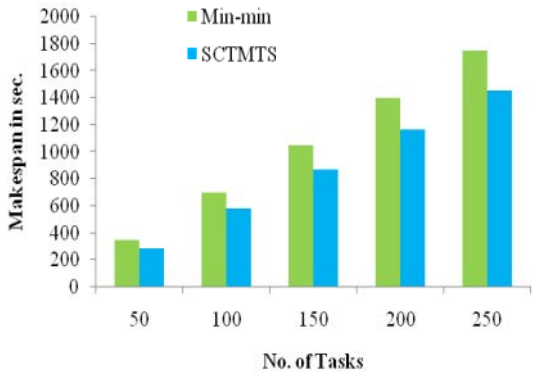


Fig. 1: Comparison based on makespan

Task heterogeneity is the amount of variation in the execution time of tasks in the metatask for a given resource. The yy-task heterogeneity (hi-high, lo-low) Resource heterogeneity is the amount of variation in the execution time of a given task among all the resources. The zz-resource heterogeneity (hi-high, lo-low). Twelve combinations of ETC matrices comprises three groups of four instances each. The first, second and third group corresponds to consistent, inconsistent and semi-consistent ETC matrices each of them having high and low combinations of task and resource heterogeneity.

**Evaluation parameters**

**Makespan:** Makespan is the important optimization criteria for grid scheduling. Makespan is calculated as makespan = max ($TCT_{ij}$). Table 11 shows the 12 different types of instances in the first column, the makespan value obtained by Min-min in the second column, SCTMTS in the third column. Graphical representation of Table 11 in Fig. 2 shows that the SCTMTS provides better makespan than Min-min Heuristic Scheduling algorithm.

Table 12-15 show the comparison of the makespan values obtained by Min-min and SCTMTS in all the four instances which comprises high task high resource, high task low resource shows the graphical representation of all the four instances for three different consistencies. Figure 3-6 show the comparison of the makespan values obtained by Min-min and SCTMTS in all the four

Table 10: Simulation parameters

| Parameters | Values |
|---|---|
| No. of Clusters | 10 |
| No. of Worker Nodes per Cluster | 10 |
| Processing Power of Worker Node | 5000-10000 MIPS |
| Task Length | 250000-600000 MI |
| No. of Tasks | 50-250 |
| Bandwidth of Worker Node | 200-500 MB |

Table 11: Comparison based on makespan (sec)

| Instances | Min-min | SCTMTS |
|---|---|---|
| u-c-hihi-0 | 8298107 | 6887429 |
| u-c-hilo-0 | 79940.04 | 66350.53 |
| u-c-lohi-0 | 267044.9 | 221647.3 |
| u-c-lolo-0 | 2600.802 | 2158.666 |
| u-ic-hihi-0 | 3565661 | 2959499 |
| u-ic-hilo-0 | 32412.49 | 26902.37 |
| u-ic-lohi-0 | 125061.7 | 103801.2 |
| u-ic-lolo-0 | 1062.335 | 881.7381 |
| u-s-hihi-0 | 4602970 | 3820465 |
| u-s-hilo-0 | 44979.51 | 37332.99 |
| u-s-lohi-0 | 169090.7 | 140345.3 |
| u-s-lolo-0 | 1586.498 | 1316.793 |

Table 12: Comparison based on makespan (sec)

| Instances | Min-min | SCTMTS |
|---|---|---|
| u-c-hihi-0 | 8298107 | 6887429 |
| u-ic-hihi-0 | 3565661 | 2959499 |
| u-s-hihi-0 | 4602970 | 3820465 |

Table 13: Comparison based on makespan (sec)

| Instances | Min-min | SCTMTS |
|---|---|---|
| u-c-hilo-0 | 79940.04 | 66350.53 |
| u-ic-hilo-0 | 32412.49 | 26902.37 |
| u-s-hilo-0 | 44979.51 | 37332.99 |

Table 14: Comparison based on makespan (sec)

| Instances | Min-min | SCTMTS |
|---|---|---|
| u-c-lohi-0 | 267044.9 | 221647.3 |
| u-ic-lohi-0 | 125061.7 | 103801.2 |
| u-s-lohi-0 | 169090.7 | 140345.3 |

Table 15: Comparison based on makespan (in sec)

| Instances | Min-min | SCTMTS |
|---|---|---|
| u-c-lolo-0 | 2600.802 | 2158.666 |
| u-ic-lolo-0 | 1062.335 | 881.7381 |
| u-s-lolo-0 | 1586.498 | 1316.793 |

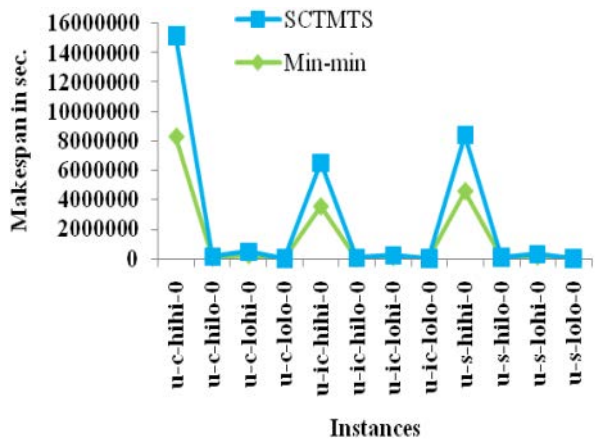


Fig. 2: Comparison based on makespan

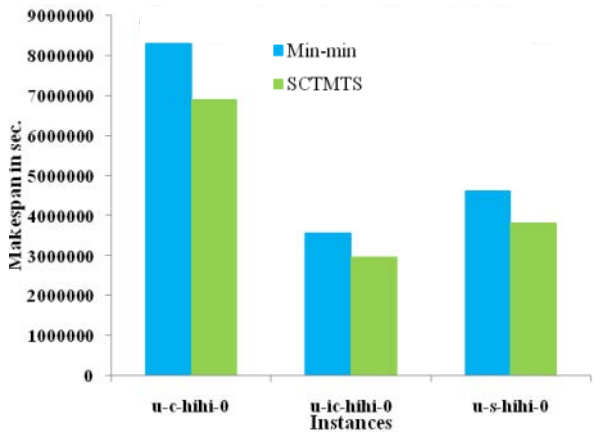


Fig. 3: Comparison based on makespan for high task high resource heterogeneity

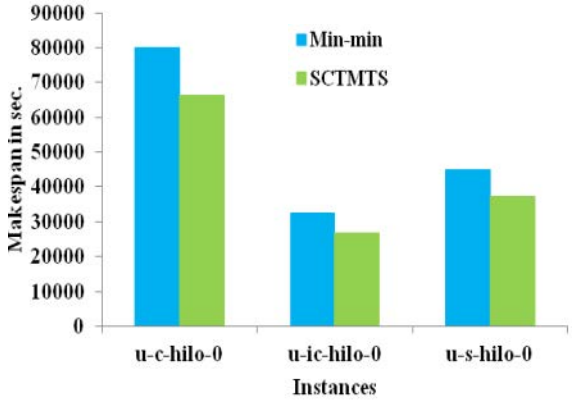


Fig. 4: Comparison based on makespan for high task low resource heterogeneity

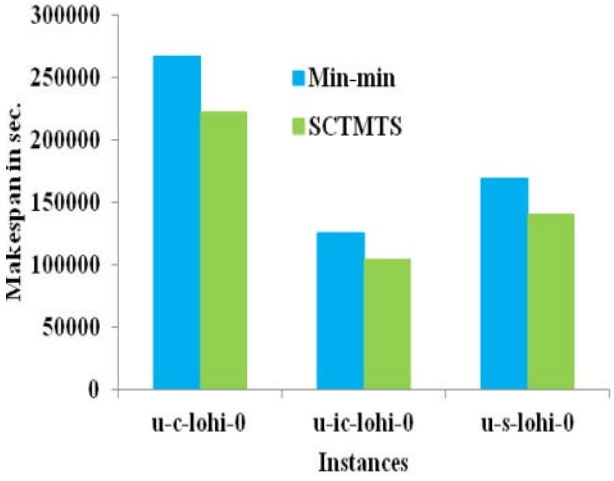


Fig. 5: Comparison based on makespan for low tas high resource heterogeneity

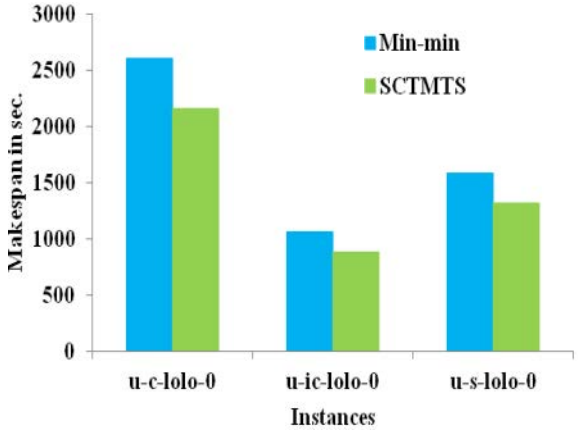


Fig. 6: Comparison based on makespan for low task low resource heterogeneity

instances which comprises high task high machine, high task low machine, low task high machine, low task low machine. The four instances are represented for consistent, inconsistent, semi-consistent or partially consistent heterogeneous computing systems.

## CONCLUSION

This study proposes a new effective task scheduling algorithm for decentralized grid environment. The experimental results show that the proposed sort completion time mean tasks scheduling algorithm produces the reduced makespan compared to that of the existing Min-min heuristic scheduling algorithm. The proposed algorithm schedules the independent tasks but the tasks may also have precedence relations. The proposed algorithm can be extended to handle dependent tasks in the future. In conclusion, the proposed algorithm regarding both makespan and resource utilization is very good.

## REFERENCES

Anousha, S. and M. Ahmadi, 2013. An Improved Min-Min Task Scheduling Algorithm in Grid Computing. In: International Conference on Grid and Pervasive Computing. Park, J.J, H.R. Arabnia, C. Kim, W. Shi and J.M. Gil (Eds.). Springer Berlin Heidelberg, Seoul, South Korea, ISBN: 978-3-642-38026-6, pp: 103-113.

Armstrong, R., D. Hensgen and T. Kidd, 1998. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. Proceedings of the 7th Workshop on the Heterogeneous Computing (HCW 98), March 30, 1998, IEEE, Orlando, Florida, ISBN: 0-8186-8365-1, pp: 79-87.

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni and M. Maheswaran et al., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 61: 810-837.

Foster, I. and C. Kesselman, 1999. The Grid: Blueprint for a New Computing Infrastructure. 1st Edn., Morgan Kaufmann Publisher, San Fransisco, ISBN-10: 1558604758, pp: 675.

Freund, R.F. and H.J. Siegel, 1993. Guest editors introduction: Heterogeneous processing. Comput., 26: 13-17.

Freund, R.F., M. Gherrity, S. Ambrosius, M. Campbell and M. Halderman et al., 1998. Scheduling resources in multi-user heterogeneous computing environments with SmartNet. Proceedings of the 7th IEEE Workshop on Heterogeneous Computing (HCW 98), March 30, 1998, IEEE, Orlando, Florida, ISBN: 0-8186-8365-1, pp: 184-199.

Ibarra, O.H. and C.E. Kim, 1977. Heuristic algorithms for scheduling independent tasks on nonidentical processors. J. ACM, 24: 280-289.

Kamalam, G.K. and V. Muralibhaskaran, 2010a. A new heuristic approach: Min-Mean algorithm for scheduling meta-tasks on heterogeneous computing systems. Int. J. Comput. Sci. Network Secur., 10: 24-31.

Kamalam, G.K. and V.M. Bhaskaran, 2010b. An improved min-mean heuristic scheduling algorithm for mapping independent tasks on heterogeneous computing environment. J. Comput. Cognition, 8: 85-91.

Kamalam, G.K. and V.M. Bhaskaran, 2011. An effective approach to job scheduling in decentralized grid environment. Int. J. Comput. Appli, 24: 26-30.

Kamalam, G.K. and V.M. Bhaskaran, 2012a. New enhanced heuristic min-mean scheduling algorithm for scheduling meta-tasks on heterogeneous grid environment. Eur. J. Sci. Res., 70: 423-430.

Kamalam, G.K. and V.M. Bhaskaran, 2012b. Novel adaptive job scheduling algorithm on heterogeneous grid resources. Am. J. Appl. Sci., 9: 1294-1299.

Maheswaran, M., S. Ali, H.J. Siegel, D. Hensgen and R.F. Freund, 1999. Dynamic mapping of a class of independent tasks onto heterogeneous com-puting systems. J. Paral. Distrib. Comput., 59: 107-131.

Parsa, S. and R. Entezari-Maleki, 2009. RASA: A new grid task scheduling algorithm. Int. J. Digital Content Technol. Appli., 3: 152-160.

Suri, P.K. and M. Singh, 2010. An efficient decentralized load balancing algorithm for grid. Proceedings of the 2nd International Advance Computing Conference (IACC), February 19-20, 2010, IEEE, Patiala, India, ISBN: 978-1-4244-4790-9, pp: 10-13.