A New Heuristic Approach:Min-mean Algorithm For Scheduling Meta-Tasks On Heterogeneous Computing Systems

Kamalam.G.K and Murali Bhaskaran.V
Anna University, Kongu Engineering College, Tamilnadu, India

Summary
Grid computing enables the collection of abundance heterogeneous resources which is geographically distributed is selected and shared for solving a large scale problem – usually to a scientific or technical problem that needs a great number of computer processing cycles or access to large amounts of data. The basic idea of grid computing is to make use of the idle CPU cycles and millions of computer systems distributed across a worldwide network. Job scheduling is a vital and challenging work in heterogeneous computing environment. The problem of mapping meta-tasks to a machine is shown to be NP-complete. The NP-complete problem can be solved only using heuristic approach. In this paper, a new heuristic technique Min-mean algorithm for scheduling meta-tasks in grid computing is presented. The proposed algorithm improves the performance in both makespan and effective utilization of resources by reducing the idle time of the machine. The performance analysis show that the proposed algorithm has a better resource utilization rate, reduced makespan and the reduced idle time of the machine than the other known algorithms.

Key words:
Grid Computing, Job Scheduling, Heuristic Algorithm, Load Balancing

1. Introduction

Heterogeneous computing environment comprises the collection of different machines interconnected by high speed networks that executes varied applications [4]. The most critical issue in grid computing is managing the resources, which are geographically distributed. In high throughput computing, the grid aims to schedule large number of meta-tasks, with the goal of reducing the idle time of the machines, reducing the makespan and also to balance the load well across the machines [15]. Applications may require enormous resources, which often are not available for the user, so a scheduling system is essential to allocate the resources to the input jobs. Managing various resources and task scheduling in highly dynamic grid environment is a challenging and indispensable task [10, 11]. The problem of mapping resources to jobs has been shown to be NP-complete [1, 3]. Many useful heuristics for static mapping [2] have been developed. Among the highly developed algorithms, Min-min algorithm is very simple, runs fast and provides better performance. Min-min algorithm schedules small tasks first which leads to load imbalance. Effective algorithms have to be designed to gain high performance. The objectives of scheduling algorithm are increasing the system throughput measure [12, 13], reducing task completion time, better resource utilization rate, and balancing the load well.
This paper presents a new scheduling algorithm named Min-mean heuristic scheduling algorithm for static mapping to achieve better performance. The proposed heuristic approach was tested using the benchmark model of Braun et al [1].

2. Related works

A set of heuristic algorithms has been designed to schedule meta-tasks to heterogeneous computing systems. It is assumed that the heuristic derive mapping statically for the collection of independent meta-task. The scheduling problem is computationally hard even though there are no data dependencies among the jobs.

2.1 Opportunistic Load Balancing(OLB)

OLB assigns each job in random order to the next available machine without considering the job's expected execution time on the machine [6, 7].

2.2. Minimum Execution Time(MET)

The minimum execution time or MET assigns each job to the machine that has the minimum expected execution time. It does not consider the availability of the machine and the current load of the machine.

2.3 Minimum Compleion Time(MCT)

The algorithm calculates the completion time for a job on all machines by adding the machine's availability time and the expected execution time of the job on the machine. The machine with the minimum completion time for the job is selected. The MCT considers only one job at a time [1]. This causes that particular machine may have the best-expected execution time for any other job.

2.4 Min-min

Min-min algorithm starts with a set of all unmapped tasks. The completion time for each job on each machine is calculated. The machine that has the minimum completion

Manuscript received January 5, 2010
Manuscript revised January 20, 2010

time for each job is selected. Then the job with the overall minimum completion time is selected and mapped to the machine. Again, this process is repeated with the remaining unmapped tasks. Compared to MCT, Min-min considers all unmapped tasks at a time [5, 6, 7].

2.5 Max-min

Max-min begins with a set of all unmapped tasks. The completion time for each job on each machine is calculated. The machine that has the minimum completion time for each job is selected. From the set, the algorithm maps the job with the overall maximum completion time to the machine. Again the above process is repeated with the remaining unmapped tasks. Similar to Min-min, Max-min also considers all unmapped tasks at a time [6, 7].

2.6 Duplex

The Duplex heuristic is literally a combination of the Min-min and the Max-min heuristic algorithms [5, 16].

2.7 GA

The Genetic algorithm (GA) is a technique used for searching large solution spaces. The GA operates on a population of chromosomes for a given meta-tasks. The initial population is generated by two methods. In the first method, a chromosome is generated randomly from a uniform distribution. In the second method, a chromosome is generated by Min-min and it is called “seeding” the population with a Min-min chromosome [17, 18].

2.8 SA

Simulated Annealing (SA) is an iterative technique that considers only one possible mapping for each meta-task at a time. Simulated annealing uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search of the solution space based on a system temperature [19].

2.9 GSA

The Genetic Simulated Annealing (GSA) heuristics is a combination of the GA and SA heuristics. GSA follows the procedures similar to the GA. For the selection process,
GSA uses the SA cooling schedule and system temperature [20].

2.10 Tabu

Tabu search is a solution space search that keeps track of the regions of the solution space to avoid repeating a search near the areas that have already been searched. A mapping of meta-tasks uses the same representation as a chromosome in the GA approach. The implementation of tabu search begins with a random mapping, generated from a uniform distribution [21].

2.11 A*

A* is a tree search technique based on an m-array tree, beginning at a root node that is a null solution. As the tree grows, intermediate nodes represent partial mappings and leaf nodes represent final mappings. Each node has a cost function, and the node with the minimum cost function is replaced by its child node. Whenever a node is added, to reduce the height of the tree, the tree is pruned by deleting the node with the largest cost function. This process is repeated until a complete mapping (a leaf node) is reached [22].
Though the above stated heuristic algorithms have advantages, they do have their own disadvantages. OLB leads to poor makespan since it does not consider the expected execution time while mapping the meta-tasks to the machines and it is also hard to achieve dynamic load balance of jobs.
MET results in severe load imbalance across the machines. Static mapping of meta-task to machine using MCT heuristic algorithm leads to poor makespan since it takes more time for a job to map to the particular machine. Max-min is appropriate only when most of the jobs arriving to the grid systems are shortest and also Max-min outperforms Min-min [14].
The experimental results from [1] show that Duplex, SA, GSA, and Tabu do not produce good mappings. Min-min, GA, and A* are able to deliver good performance. GA is better than Min-min by a few percents, and also it has to be “seeding” the population with a Min-min chromosome to obtain its good performance. In different situations, A* produce better or worse mappings than Min-min and GA. Among the three algorithms, Min-min is the fastest algorithm, GA is much slower, and A* is very slow.
Among the stated algorithms, Min-min is the simple and fastest algorithm and its good performance depends on the choice of mapping the meta-tasks to the first choice of minimum execution time. However the drawback of Min-min is that, it is unable to balance the load because it usually assigns the small task first and few larger tasks, while at the same time, several machines sit idle, which leads to poor utilization of resources. The proposed algorithm retains the advantage of Min-min algorithm and reduces the idle time of the resources, which in turn leads to better makespan.

3. Problem Definition

This section, presents the problem of job scheduling in heterogeneous computing environment.
In this paper the experimental study is based on a benchmark simulation model by Braun et al. [1];
In this model static mapping of meta-tasks is considered. Each machine executes one task at a time in the order in which tasks are allocated to the machines. For static mapping, the size of the meta-tasks and the number of machines in the heterogeneous computing environment is known a priori. Since there are static heuristics, the accurate estimate of the expected execution time for each task on each machine is known a priori to execution and is contained within an ETC(expected time to compute) matrix where ETC(t_i, m_j) is the estimated execution time of task i on machine j.
Using the ETC matrix model, the scheduling problem can be defined as follows:

- A number of independent jobs to be allocated to the available grid resources. Because of Non-preemptive scheduling, each job has to be processed completely in a single machine.
- Number of machines is available to participate in the allocation of tasks.
- The workload of each job(in millions of instructions)
- The Computing capacity of each resources(in MIPS)
- ready m- represents the ready time of the machine after completing the previously assigned jobs.
- ETC matrix of size t * m, where t-represents the number of jobs and m-represents the number of machines.

3.1 Proposed Min-mean Heuristic Scheduling Algorithm

Job scheduling system is the most important part of grid resource management system. The scheduler receives the job request, and chooses appropriate resource to run that job. In this paper, the formulation of job scheduling is based on the expected time to compute (ETC) matrix of Braun et al.
A meta-task is defined as a collection of independent task (i.e. task doesn't require any communication with other tasks) [1, 9]. Tasks derive mapping statically. For static mapping, the number of tasks, t and the number of machines, m is known a priori.
ETC (i,j) represents the estimated execution time for task t_i on machine m_j.

The expected completion time of the task t_i on machine m_j is

$$ct(t_i, m_j) = mat(m_j) + ETC(t_i, m_j)$$

mat (m_j) is the machine availability time, i.e. the time at which machine m_j completes any previously assigned tasks [8].
The main aim of the heuristic scheduling algorithm is to minimize the makespan where

$$makespan = \max(ct(t_i, m_j))$$

The proposed heuristic scheduling algorithm Min-mean works in two phases.

- In phase 1, the job allocation is done based on the Min-min algorithm.
- In phase 2, the mean of all machines completion time is taken. The machine whose completion time is greater than the mean value is selected. The tasks allocated to the selected machines are reallocated to the machines whose completion time is less than the mean value.

The related definition of proposed Min-mean heuristic scheduling algorithm is as follows:

- ET_{ij} - the amount of time taken by machine M_j to execute $Task_i$ given that M_j is idle when $Task_i$ is assigned.
- CT_j - the expected completion time of M_j
- Mat (m_j) - the machines availability time i.e. the time at which $Machine_j$ completes any previously assigned tasks.
- Group (CT_i, $Machine_j$) –The function "f1" is used to group all the tasks and machines that has minimum completion time.
- The best minimum task/machine pair ($Task_i$, $Machine_j$) is selected from the Group
- MeanCT- is used to find the mean completion of all the machines.

3.1.1 Algorithm Min-mean

```
(1) while there are tasks to schedule
(2)     for all $Task_i$ to schedule
(3)         for all $Machine_j$
(4)             Compute$CT_{i,j}$ ; $CT_{i,j}$ = Mat ($m_j$) + $ET_{ij}$
(5)         end for
(6)         Group ($CT_i$, $Machine_j$) =f1 ($CT_{i,1}$, $CT_{i,2}$ ...)
(7)     end for
(8) Select the best minimum pair ($Task_i$, $Machine_j$) from
```

```
        the Group
 (9)  Compute minimum $CT_{i,j}$
(10) Reserve $Task_i$ on $Machine_j$
(11) end while
//Optimization based on MeanCT
(12) Calculate MeanCT= $(\Sigma CT_j)$/No of machines
(13) for all $Machine_j$
(14)    if ($CT_j$>MeanCT)
(15)    Select tasks $Task_i$ reserved on the machines
          $Machine_j$
(16) end for
(17) Sort the selected machines in the increasing order of
       $CT_j$
(18) for all $Machine_k$ reselected
(19) for all $Task_i$ in $Machine_k$ reselected
(20)   for all $Machine_j$
(21)     Compute New $CT_{i,j}$
(22)     if(New $CT_{i,j}$ < MeanCT)
(23)       Group ($CT_i$, $Machine_j$) =f1 ($CT_{i,1}$, $CT_{i,2}$ ...)
(24)     endif
(25)   end for
(26) Select the best minimum pair ($Task_i$, $Machine_j$) from
       the Group
(27)  Compute minimum New $CT_{i,j}$
(28)   Reschedule ($Task_i$ on $Machine_j$)
(29) end for
(30) Compute Makespan=Max($CT_{i,j}$)
(31) end for
```

Min-min heuristic scheduling algorithm executes all shortest tasks first and then the longest task. Table 1 gives a sample ETC matrix, the expected execution time of three tasks (t1, t2, t3) on two machines (m1, m2). This sample ETC matrix clearly explains how proposed Min-mean heuristic scheduling algorithm performs better than the Min-min algorithm. It is assumed that both the machines are idle at the start.

TABLE 1. THE EXECUTION TIME OF THREE TASKS ON TWO MACHINES

	m1	*m2*
t1	1	2
t2	2	4
t3	5	9

The sequence of the execution of Min-min algorithm and the proposed Min-mean heuristic scheduling algorithm is as follows:

- Step 1: Static mapping of tasks to machines based on Min-min is shown in Figure 1. Min-min algorithm gives a makespan of 8 sec.

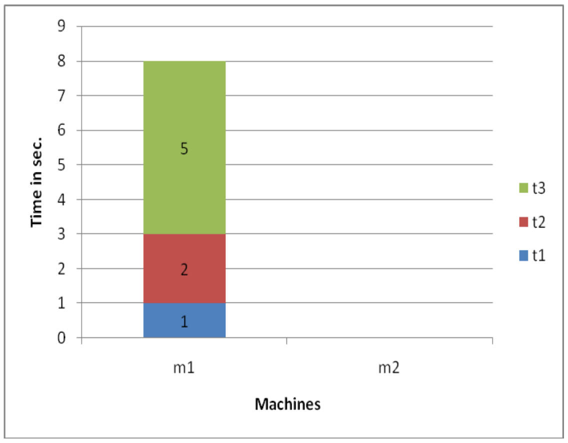


Figure 1: Results of Min-min algorithm

- The proposed Min-mean heuristic scheduling algorithm works as follows:
- Step 2: The mean completion time for the sample ETC matrix can be calculated by using the following relation:

MeanCT = CTm1+CTm2

where,

CTm1: Completion time of all tasks on machine m1

CTm2: Completion time of all tasks on machine m2.

MeanCT = 4 sec.

- Step 3: Tasks on machines m1 are selected as shown in Figure 2 because

CTm1 > MeanCT.

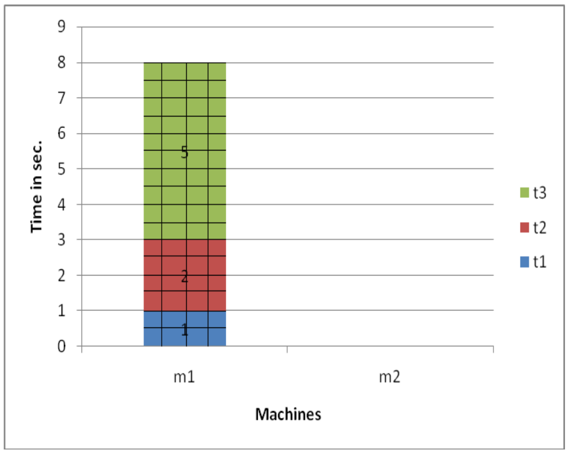


Figure 2: Selected tasks on machine m1

- Step 4: Rescheduling of the tasks on machine m1 to the machine m2 is done, whose expected execution time is

$ET_i < MeanCT$

- The final scheduling of the tasks (t1, t2, t3) on two machines (m1, m2) using the proposed Min-mean heuristic scheduling algorithm is shown in Figure 3. Min-mean heuristic scheduling algorithm gives a makespan of 6 sec.

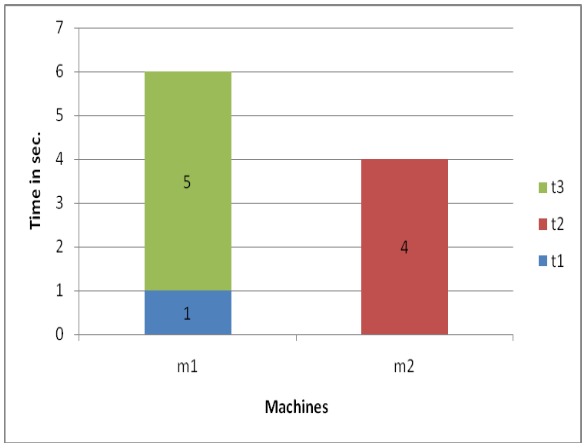


Figure 3: Results of Min-mean algorithm

- The Figure 1 and Figure 3 clearly show that Min-mean heuristic scheduling algorithm performs better than Min-min algorithm.
- The comparison results of Figure 1 and Figure 3 is as follows:
    - The idle time of the machine m2 is reduced.
    - The load is well balanced in both the machines m1 and m2.
    - The measure of the throughput of the heterogeneous computing systems is termed as makespan. The makespan can be calculated as

makespan = max ($CT_{i,j}$)

makespan = 6 sec.

Figure 1 and Figure 3 shows that the makespan using Min-mean is reduced compared to that of the makespan using Min-min.

4. Experimental Results

Experimental results obtained for the benchmark of instances by Braun et al. [1] for various heuristic scheduling algorithms were compared with the proposed algorithm.

4.1 Benchmark Description

The makespan of the various heuristic algorithms were compared using Braun et al. benchmark. Using the ETC matrix, the instance of this benchmark is divided into 12 different types each of them consisting of 100 instances based on the three metrics: job heterogeneity, machine heterogeneity and consistency. Instances are labeled as u-x-yyzz.k where

u- uniform distribution used to generate ETC matrix.

x- Type of consistency(c-consistent, i-inconsistent, s-semi-consistent or partially-consistent).

An ETC matrix is consistent if a machine executes any job t_i faster than machine m_k, then machine m_j executes all jobs faster than machine m_k [8].

An ETC matrix is inconsistent if a machine m_j is faster than machine m_k for some jobs and slower for other jobs.

An ETC matrix is semi-consistent or partially consistent if it includes a consistent sub-matrix.

Job heterogeneity: Variation in the execution time of the task for a given machine.

yy- the heterogeneity of the jobs (hi- represents high, lo-represents low).

Machine heterogeneity: Variation in the execution time for a particular task among the entire machine.

zz- the heterogeneity of the machines (hi- represents high, lo-represents low).

Every instance consists of 512 jobs and 16 machines. The experimental results are based on the set of 12 instances, which comprises three groups of four instances each. The first group relates to the consistent ETC matrices of various combinations comprising the machine heterogeneity and job heterogeneity. The second and third group relates to the inconsistent and semi-consistent ETC matrices.

The experimental results are tabulated in Tables 2 and 3. The makespan computed for MET, MCT, Min-min and the proposed Min-mean heuristic scheduling algorithm clearly specifies the fair performance of the proposed heuristic scheduling algorithm over the existing heuristic algorithms.

5. Performance Analysis

To evaluate the efficiency of the proposed Min-mean heuristic scheduling algorithm described in section 3 Min-mean heuristic scheduling algorithm is compared with MET, MCT, Min-min heuristic algorithm in all the four instances.

Table 2 show the improvement of the proposed Min-mean heuristic scheduling algorithm over Min-min.

Figure 4, 5, 6 represents the improvement of Min-mean heuristic scheduling algorithm over Min-min in all 12 different types of instances based on the three metrics: Job heterogeneity, machine heterogeneity and consistency.

TABLE 2. IMPROVEMENT OF MIN-MEAN OVER MIN-MIN

Consistency	***Improvement over Min-min***	
Inconsistent	***High-High***	4.56%
	High-Low	6.79%
	Low-High	5.28%
	Low-Low	4.77%
Consistent	***High-High***	1.58%
	High-Low	0.15%
	Low-High	1.47%
	Low-Low	0.97%
Partially consistent	***High-High***	1.80%
	High-Low	0.52%
	Low-High	3.59%
	Low-Low	0.83%

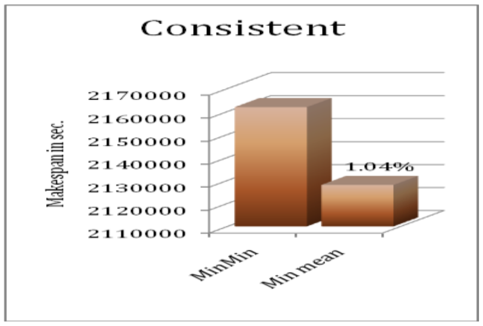


Figure 4. Graphical representation for improvement of Min-mean over Min-min algorithm for Consistent job heterogeneity and machine heterogeneity

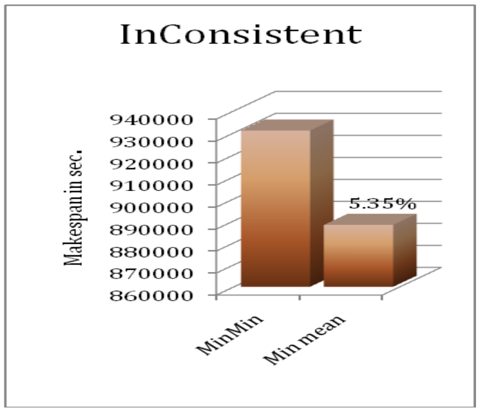


Figure 5. Graphical representation for improvement of Min-mean over Min-min algorithm for Inconsistent job heterogeneity and machine heterogeneity

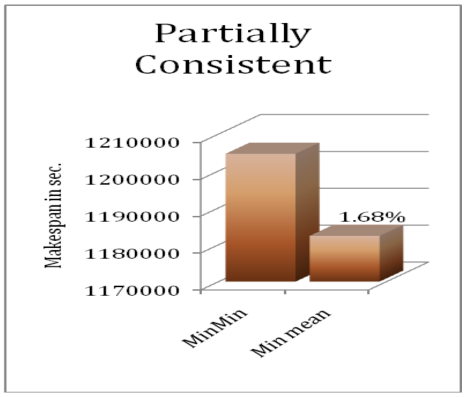
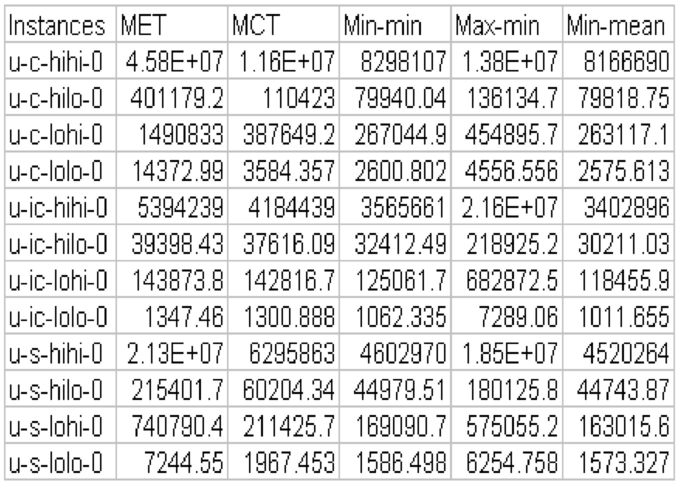


Figure 6. Graphical representation for improvement of Min-mean over Min-min algorithm for Partially consistent job heterogeneity and machine heterogeneity

The four instances comprises High task High machine, High task Low machine, Low task High machine, Low task Low machine. The four instances are represented for three different (consistent, inconsistent, semi-consistent or partially-consistent) heterogeneous computing systems.

Table 3 represents the makespan value obtained by MET, MCT, Min-min, Max-min, Min-mean in the first, second, third, fourth and fifth column respectively. Graphical representation of Table 3 in Figure 7 show that the proposed Min-mean heuristic scheduling algorithm improves the efficiency in both makespan and resource utilization rate among all the heuristics selected for analysis.

TABLE 3 COMPARISON OF MAKESPAN VALUES OBTAINED BY MET, MCT, MIN-MIN, MAX-MIN, MIN-MEAN USING BRAUN ET AL. BENCHMARK



Instances	MET	MCT	Min-min	Max-min	Min-mean
u-c-hihi-0	4.58E+07	1.16E+07	8298107	1.38E+07	8166690
u-c-hilo-0	401179.2	110423	79940.04	136134.7	79818.75
u-c-lohi-0	1490833	387649.2	267044.9	454895.7	263117.1
u-c-lolo-0	14372.99	3584.357	2600.802	4556.556	2575.613
u-ic-hihi-0	5394239	4184439	3565661	2.16E+07	3402896
u-ic-hilo-0	39398.43	37616.09	32412.49	218925.2	30211.03
u-ic-lohi-0	143873.8	142816.7	125061.7	682872.5	118455.9
u-ic-lolo-0	1347.46	1300.888	1062.335	7289.06	1011.655
u-s-hihi-0	2.13E+07	6295863	4602970	1.85E+07	4520264
u-s-hilo-0	215401.7	60204.34	44979.51	180125.8	44743.87
u-s-lohi-0	740790.4	211425.7	169090.7	575055.2	163015.6
u-s-lolo-0	7244.55	1967.453	1586.498	6254.758	1573.327

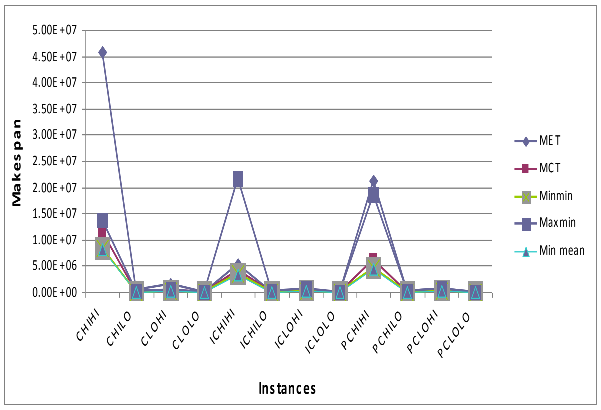


Figure 7. Graphical representation of Table 3

6. Conclusions and Future Work

The implementation of Min-mean heuristic scheduling algorithm and various existing algorithm are tested using the benchmark simulation model for distributed heterogeneous systems by Braun et al. (2001). The experimental results show that Min-mean performs better than the existing heuristic algorithm in various systems and settings and also it delivers improved makespan on various heterogeneous environments such as job heterogeneity (high, low), machine heterogeneity (high, low), and consistency (consistent, inconsistent, semi-consistent or partially-consistent). The future research will be directed towards the factors such as CPU workload, communication delay and so on.

References

[1] Tracy D.Braun, Howard Jay Siegel, and Noah Beck, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", Journal of Parallel and Distributed Computing 61, pp.810-837, 2001.

[2] J.M.Schopf, "A General Architecture for Scheduling on the Grid", special issue of JPDC on Grid Computing, 2002.

[3] T.Braun, H.Siegel, N.Beck, L.Boloni, M.Maheshwaran, A.Reuther, J.Robertson, M.Theys, B.Yao, D.Hensgen, and R.Freund, "A Comparison Study of Static Mapping Heuristics for a Class of Meta-tasks on Heterogeneous Computing Systems", In 8th IEEE Heterogeneous Computing Workshop(HCW'99), pp. 15-29, 1999.

[4] I.Foster, C.Kesselman, and S.Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, 15(3), pp. 200-222, 2001.

[5] R.F.Freund, and M.Gherrity, "Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net", In Proceedings of the 7th IEEE HCW, 1998.

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

[7] R.F.Freund and H.J.Siegel,"Heterogeneous Processing", IEEE Computer , 26(6), pp. 13-17, 1993.

[8] J.Brevik, D.Nurmi, and R.Wolski, "Automatic Methods for Predicting Machine Availability in Desktop Grid and Peer-to-Peer Systems", In Proceedings of CCGRID'04, pp. 190-199, 2004.

[9] TD. Braun, HJ. Siegel, N.Beck, " A Taxonomy for Descriging Matching and Scheduling Heuristics for Mixed-machine Heterogeneous Computing Systems", IEEE Workshop on Advances in Parallel and Distributed Systems, West Lafayette, pp. 330-335, 1998.

[10] T.Ghazawi, K.Gaj, N.Alexandridis, F. Vroman, N.Nguyen, P.Samipagdi and S.suboh, "A Performance study of Job Management Systems", Concurrency and Computation: Practice and Experience 16(13): 1229-1246, 2004.

[11] He Xiaoshan, Xia-He Sun, Gregor Von Laszewski, "QoS Guided Min-min Heuristic for Grid Task Scheduling", Journal of Computer Science and Technology, pp. 442-451, July 2003.

[12] Zhang Qian, Li Zhen, "Design of Grid Resource Management System Based on Divided Min-min scheduling Algorithm", IEEE First International Workshop on Education Technology and Computer Science, pp. 613-618, 2009.

[13] Hojjat Baghban, Amir Masoud Rahmani, " A Heuristic on Job Scheduling in Grid Computing Environment", In Proceedings of the seventh IEEE International Conference on Grid and Cooperative Computing, pp. 141-146, 2008.

[14] Hui Yan, Xue-Qin-Shen, Xing Li, Ming-Huiwu, " An Improved Ant Algorithm for Job Scheduling in Grid Computing", In Proceedings of the IEEE fourth International Conference on Machine Learning and Cybernetics, pp. 2957-2961, August 2005.

[15] Li Wenzheng, Zhang Wenyue, " An Improved Scheduling Algorithm for Grid Tasks", International Symposium on Intelligent Ubiquitous Computing and Education, pp. 9-12, 2009.

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

[17] H.Singh, and A.Youssef, "Mapping and Scheduling Heterogeneous Task Group Using Genetic Algorithms", in 5th IEEE Heterogeneous Computing Workshop (HCW'96), pp. 86-97, April 1996.

[18] L.Wang, H.J.Siegel, V.P.Roychowdhury, and A.A.Macicjewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic AlgorithmBased Approach", Journal of Parallel and Distributed Computing, 47(1), pp. 1-15,November 1997.

[19] M.Coli, and P.Palazzari, "Real Time Pipelined System Design Through Simulated Annealing", Journal of Systems Architecture, 42(6-7), pp. 465-475, December 1996.

[20] H.Chen, N.S.Flann, and D.W.Watson, "Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Approach", IEEE transactions on Parallel and Distributed Computing, 9(2), pp. 126-136, February 1998.

[21] I.D.Falco, R.D.Balio, E.Tarantino, and R.Vaccaro, "Improving Search by Incorporating Evolution Principles in Parallel Tabu Search", in IEEE Conference on Evolutionary Computation, pp. 823-828, 1994.

[22] K.Chow, and B.Liu, "On Mapping Signal Processing Algorithms to a Heaterogeneous Multiprocessor System", in ICASSP'91, pp. 1585-1588, May 1991.

AUTHORS

V.Muralibhaskaran, M.E., Ph.D., Principal, Paavai College of Engineering, Pachal, Namakkal-637 018, India, He obtained his Bachelors degree in Computer Science and Engineering," from Bharathidasan University, Thiruchirapalli and MS in Computer Science from BITS, Pilani and Masters Degree in Computer Science and Engineering from Bharathiyar University, Coimbatore. He completed PhD in Network Security from Bharathiyar University, Coimbatore. He presented 12 papers in National and International Conferences. He published 4 papers in international journals. He is presently working as a Principal of Paavai College of Engineering, Pachal, Namakkal. He received the **"Best Staff"** award for the year 1991- 1992 at Sathyabama Engineering College, Chennai.and 2002-2003 in Kongu Engineering College, Perundurai. He is guiding 10 research scholars and his area of interest is Cryptography and Network Security, High Speed Networks, and Computer Architecture.

G.K.Kamalam.,B.E.,MBA.,M.E., (Ph.D)., Senior Lecturer, Kongu Engineering College, Perundurai, Erode-638 052, India. She obtained her Bachelors degree in Computer Science and Engineering from Madras University, Chennai and Masters of Business Administration from Madurai Kamaraj University, Madurai and M.E degree in VLSI from Anna University, Chennai. She is currently doing research in Grid Computing under Anna University, Coimbatore. She is presently working as a Senior Lecturer in the Department of Computer Science and Engineering, Kongu Engineering College, Perundurai,Tamilnadu, India. Her area of interest is Grid Computing, Datastructures and analysis of algorithms, and Compiler Design. She has presented papers in National conferences.