ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **16**, Number 4, 2013, 373–392

Distributed Scheduling on Utility Grids

Sunita BANSAL[1], Chittaranjan HOTA[2]

[1]Birla Institute of Technology & Science, Pilani
Dept. of Computer Science & Information Systems
Pilani Campus, Pilani, Rajasthan, 333031, INDIA
E-mail: sunita_bansal@pilani.bits-pilani.ac.in

[2]Birla Institute of Technology & Science, Pilani
Dept. of Computer Science & Information Systems
Hyderabad Campus, Hyderabad, AP, 500078, INDIA
E-mail: hota@hyderabad.bits-pilani.ac.in

Abstract. Grid computing aggregates the power of widely distributed resources and provides non-trivial services to the users. Resource management and scheduling play a crucial role in Grid computing environment. It becomes more challenging in Utility Grids, where consumer wants to pay minimum amount and providers want to earn maximum amount. Many existing scheduling approaches are dedicated to either single criterion or multi-criteria. Existing multi-criteria systems give weight to each criterion according to its relative importance. Users need not worry about the choice of these criteria. These algorithms are less efficient in terms of time complexity and memory utilization. In this paper, we propose an efficient distributed scheduling algorithm on Utility Grids based on Technique for Order Preference by Similarity to Ideal Solution (TOPSIS). We have introduced a new parameter called Processing Element (PE) weight. The objective of this parameter is to select the resource that carves out smallest possible free PE cluster that satisfies the user requirements. User and system defined weights are used to select resource and evaluate the performance of the algorithm. The GridSim toolkit with standard workload model is used to simulate the Grid environment. The simulation results show that our algorithms perform better over existing approaches in terms of user satisfaction and number of application failures.

Key words: Cost-time, Failure, Parallel Applications, TOPSIS, Utility Grids.

1. Introduction

Due to advances in wide-area network technologies and low cost of computing resources, Grid computing [1] has established itself as an active research area. One of the motivational factors for Grid computing is to aggregate the capabilities of widely distributed resources and to provide non trivial service to users. A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities.

The key technologies that affect the Grid efficiency involve the Grid resource allocation and management. The goal of computational Grid is to utilize all available free computational resources to overcome difficulties brought about by complicated tasks with enormous computing workloads. One of the current research problems is to devise new and efficient methods for resource management.

A number of resource management approaches have been proposed in various Grid projects. Resource management in a Grid environment is done by either centralized or distributed manner. In centralized approach, a centralized broker manages and schedules all the jobs submitted to the Grid, whereas a distributed broker typically handles jobs submitted by a single user only. Centralized broker is able to produce optimal schedules as they have full knowledge of the jobs and resources, but such a scheduler can face a performance bottleneck problem. Distributed broker architecture, on the other hand, scales well and makes the Grid more fault-tolerant, but the partial information is available for each instance as the broker complicates the scheduling decisions. Grid resource management systems employing centralized broker include Condor [2]. Distributed brokers are carried out by AppLeS [4] and Net Solve [5].

In this paper we introduce two resource broker scheduling algorithms namely Economy Minimum Execution Time (EMET) and Economy Minimum Completion Time (EMCT).

The new scheduling algorithm is concerned with the current situation in which user has to pay-per-use like Cloud and Utility computing. Cloud and Utility computing have different pricing methods like auction, bid and commodity market. We have chosen commodity market where prices are fixed for all commodity. User objective is to get the job completed in least time, at least cost; service provider's objective is to generate maximum revenue. It becomes hard because the resource that provides better efficiency are more expensive. Resource broker has to select the resources that satisfy the user requirement.

Our main contributions are as follows: First, we have modeled parallel application on the Utility Grids where user requires more than one Processing Element (PE) to execute the application. Second, we take cost and time weight from the user. Weight is used to select a resource and also to measure user benefit. Third, new parameter called Processing Element (PE) weight is introduced. The objective of this parameter is to select the resource that carved out smallest possible free PE cluster that satisfy the user requirements. It does not occupy large free PE cluster that might later satisfy large requests from applications.

The outline of this paper is organized as follows. Section 2 reviews related work on task scheduling in a heterogeneous computing environment. Section 3 formulates

the system model. Section 4 describes our proposed scheduling algorithm for Utility Grids. Simulation and evaluation of our approach is presented in Section 5. Section 6 discusses the conclusion and future work.

2. Related Work

Braun et al. [6] studied various online scheduling algorithms. Opportunistic Load Balancing (OLB) algorithm assigns the task to a machine which is expected to be available first without considering the expected execution time. Aim of this strategy is to keep all machines as productive as possible. Minimum Execution Time (MET) scheduling algorithm assigns the task to a machine that takes the minimum execution time among all the available machines. Purpose of this approach is to assign the task to the best machine. Minimum Completion Time (MCT) scheduling algorithm assigns the task to a machine which would complete the task at the earliest, so that all the machines are busy and makespan time is less. Kumar et al. [11] introduced Modified Minimum Completion Time (Modified MCT) task scheduling algorithm. They divided the jobs into three categories short, medium and long. Short jobs are assigned using OLB method and other jobs are assigned using the MCT method to take advantage of resource utilization and makespan time.

Buyya et al. [7] developed various cost, time optimization greedy algorithms for parameter sweep applications. These algorithms optimize either cost or time. Garg et al. [3] introduced MinCTT, MaxCTT, and sufferageCTT. All these algorithms work on economy Grid where cost and time are minimized simultaneously. Ang et al. [9] proposed a new cost and time balancing algorithm. They have defined a new parameter urgency of the task, based on deadline. Resources are divided into two groups concerning average cost. Tasks that have more weight of cost are assigned to cost optimization resource group and vice versa.

Our work is different from others as we have considered the online jobs instead of batch jobs, and used the TOPSIS [12] to select resource instead of cost time trade-off matrix. Researchers [10], [23], [24] have proposed two phase scheduling heuristic; first phase assigns the tasks using min-min [6] scheduling algorithm and the second phase reallocates tasks from one machine to the other machine in order to reduce the makespan time.

He et al. [18] introduced a QoS guided Min-min heuristic. It classifies the tasks based on the bandwidth requirement. The high bandwidth requirement tasks are assigned first using Min-min heuristic [6]. Carsten et al. [17] developed scheduling algorithms on economic Grid because back fill [19] scheduling algorithm does not consider the cost of the task and a task can suffer from starvation problem. Researchers in [16], [26], [25] have developed static mapping heuristics for Quality of Service (QoS) guarantees. They have considered various QoS parameters like time, reliability, security, version and priority of the task. Tasks are assigned based on utility which is calculated on the basis of the QoS. The efficiency of this approach is evaluated in terms of response time and wait-time.

Besides, there are also some metaheuristic approaches, such as Genetic Algorithms

(GAs) [27], Simulated Annealing (SA) [28], and Genetic Simulated Annealing (GSA) [27]. In general, metaheuristic approaches manage to obtain much better performance, but take a longer execution time.

Deb et al. [20] have developed various multi-objective optimization genetic algorithms. Non-dominating Sorting Genetic Algorithm (NSGA) [20] have introduced a crowded comparison operator to measure density of population. That operator is less complex and does not require user defined sharing parameter. Chitra et al. [22] have developed weighted multi-objective scheduling algorithm on workflow Grid scheduling. They have used simple neighbor search algorithm for local search. Metaheuristic approach finds number of combinations of tasks and resources whereas our approach finds the best resource available as soon as an application arrives into the system.

3. System Model

The four layers of Grid architecture shown in Fig. 1 consists of the following entities:

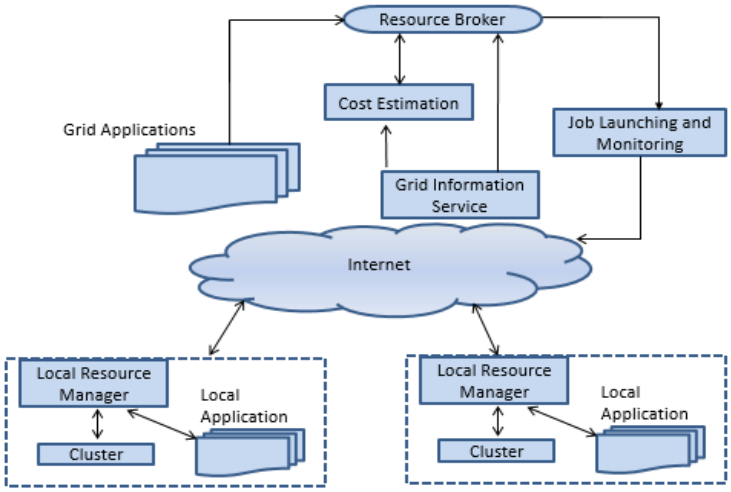

Fig. 1. Grid model.

Service Providers: Service providers are resource owners, including clusters, servers, and supercomputers. They are responsible for executing Grid user application. Resource provider provides static information to Grid Information Server (GIS). Static information includes CPU speed in terms of Million Instructions Per Second (MIPS), operating system, the number of PEs, and usage cost per second and resources architecture.

Users: Users have to be registered with GIS and submit their applications to the resource broker for execution on Grid. They also supply information on the QoS parameters.

Resource-Broker: Resource broker schedules applications to the resource provider. It collects the resources information from the GIS.

Grid Information Server: A GIS contains information about all available Grid resources with their computing capacity and cost at which they offer their services to Grid users.

This paper assumes that all the participants trust and benefit one another by cooperating with one another. It is assumed that service price does not change during the scheduling of applications. We assume an application requires fixed number of PEs and an application cannot be executed until all the required PEs are available simultaneously.

4. Distributed Scheduling Algorithm

We first modeled the applications as parallel applications where the user requires more than one computing elements. We applied EMCT scheduling algorithm to select the resources which selects the resources on the basis of user's requirements. User's requirements are expressed in terms of weight. To solve this multi-criterion scheduling problem, we used the well known TOPSIS [12] algorithm.

4.1. Problem Statement

Our algorithms consider a Grid environment that consisting a set of resource providers R = $\{r_j,\ r_{j+1}\ \ldots\ ,\ r_m\}$, each resource's available time-slots TS = $\{t_k,\ t_{k+1},\ .\ .\ .\ ,\ t_p\}$., and a set of Parallel applications A = $\{a_i,\ a_{i+1},\ .\ .\ .\ ,\ a_n\}$. Each application is characterized by $a_i = (al_i,\ ap_i,\ ad_i,\ as_i,\ a\eta_i,\ a\delta_i)$ where al_i is application length in number of instructions that can be estimated on dedicated and non dedicated clusters using [13], [14], [15], ap_i is the application required processing elements (PE), ad_i is application input data in bits, as_i is the application submission time, $a\eta_i$ and $a\delta_i$ are the application time and cost weight respectively.

Each resource characteristic is defined by $r_i = (rc_j,\ rs_j,\ rp_j,\ rb_j,\ rt_j)$. The first three parameters are static and others are dynamic. The rc_j is resource processing cost per second, rs_j represents resource computational power in terms million instructions per second (MIPS) of one PE, rp_j is resource PE, rb_j is resource bandwidth that changes periodically, rt_j is the list of available time-slots. Time-slot is characterized by $ts_k = (ts_k,\ tf_k,\ tp_k)$ where ts_k is start time of time slot, tf_k is finish time of time slot and tp_k available PE of time slot. We assume that application a_i cannot be executed until all the required pe_i are available simultaneously. Let m be the total number of resource providers available, for the application a_i and γ_i be PE weight that should be defined by the system. The notations used in the paper are described in Table 1.

A lower bound of cost and time of all successful applications can be calculated as minimum cost and minimum time. Value of cost is minimum when application is scheduled on cheapest resources. Value of time is minimum when application is scheduled on fastest resources. Execution time of application a_i on resource provider

r_j is given by:

$$\Psi_{ij} = \frac{al_i}{rs_j}. \tag{1}$$

Response time of application a_i on resource provider r_j is given by:

$$\alpha_{ij} = tf_{ik} - as_i. \tag{2}$$

The cost of executing application a_i on resource provider r_j is calculated by:

$$c_{ij} = rc_j * ap_i * \Psi_{ij}. \tag{3}$$

Transfer time of application a_i on resource provider r_j is given as follows.

$$\tau_{ij} = \frac{al_i + ad_i}{rb_j}. \tag{4}$$

Table 1. Notation

Notation	Definition
al	Length of application
ad	Input data of application
ap	Required processing element of application
as	Submission time of application
aη	Time weight of application
aδ	Cost weight of application
γ	PE weight
rc	Resource cost per second of
rs	Resource MIPS
rp	Resource PE
rb	Resource bandwidth
rt	List of time-slots
ts	Start time of Time-slot
tf	Finish time of Time-slot
tp	Available Processing element of time-slot

4.2. Economic Minimum Completion Time (EMCT)

EMCT is a combination of the MCT [6] and TOPSIS [12] algorithms. MCT heuristic assigns the task to a machine which would complete the task at the earliest so that all the machines are busy. TOPSIS is based on the concept that the chosen alternative has the shortest geometric distance from the positive ideal solution and the longest geometric distance from the negative ideal solution. It is a method of compensatory aggregation that compares a set of alternatives by identifying weights for each criterion, normalizing scores for each criterion and calculating the geometric distance between each alternative and an ideal alternative, which is the best score in each criterion.

The pseudo code of EMCT is given in Algorithm 1. This algorithm uses a matrix that represents decision matrix of TOPSIS algorithm. Matrix is of size [i][j], where $i = 1, 2...m$ and $j = 1, 2...n$. Here, m is the number of resources that satisfy the application requirement and n is the number of criterion. Matrix [0][n] contains weight of each criterion.

Whenever a new application arrives in the system, the broker collects information about resources and the application. Steps 3 to 5 assign the weight of each criterion that can be either defined by the user or by the system. Steps 6 to 8 find free time slots from each resource and determine feasible time slot of each resource. A feasible time slot is a slot which has the number of PEs more than or equal to required PEs and start time should be equal or greater than the application submission time. Steps 9 to 11 calculate the response time, cost and transfer time of the application using equations 1, 2, 3, 4 respectively. Steps 15 to 17 assign total time, processing cost and available PEs of time slot as an alternate in the Matrix. This process is repeated for all the resources. At step 19, TOPSIS process is called that returns the ideal solution as resource. At Step 20, resource broker reserves resources for an application. This process is repeated until applications arrive into the system.

Algorithm 1 Pseudo code of EMCT

1: **while** application arrives into system **do**
2: *get an application* a_i
3: $matrix[0][0] = a\eta_i$
4: $matrix[0][1] = a\delta_i$
5: $matrix[0][2] = \gamma_i$
6: ***for all*** *resource* $r_j \in R$ ***do***
7: ***for all*** *time-slot* $t_k \in TS$ ***do***
8: ***if*** $(tp_k >= ap_i)$ ***then***
9: α_{ij} = *equation 2*
10: c_{ij} = *equation 3*
11: τ_{ij} = *equation 4*
12: *break*
13: ***end if***
14: ***end for***
15: $matrix[j][0] = \tau_{ij} + \alpha_{ij}$
16: $matrix[j][1] = c_{ij}$
17: $matrix[j][2] = tp_k - ap_i$
18: ***end for***
19: $res \leftarrow topsis(matrix)$
20: $res \leftarrow a_i$
21: **end while**

4.3. TOPSIS

Input to TOPSIS algorithm is the decision matrix that contains weight and value of each criterion. The pseudo code of TOPSIS is given in Algorithm 2. As all the

criteria have different units, Step 1 finds the normalized matrix using equation 5. Step 2 calculates the weighted normalized matrix using equation 6. Now, positive ideal solution and negative solution of each criterion is evaluated using equations 7 and 8. Positive ideal solution has minimum value in criteria and negative solution has maximum value in criteria, in case of minimization problem. Step 4 finds the separation measure of each cluster on the basis of negative solution and positive ideal solution. After that, it computes the relative closeness from ideal point and negative point, ranks the resources on the basis of closeness and returns the resource that has minimum closeness.

4.4. Construction of Decision Matrix D

In the context of resource selection, the effect of each criterion cannot be considered alone and should be viewed as a trade-off among various criteria. The decision matrix, D can be constructed as shown in Table 2.

Here i denotes the alternative resources i=1, 2,..., m; j represents the j^{th} criterion, j = 1, 2, ..., n related to i^{th} cluster, and f_{ij} is a crisp value indicating the performance value of each resource f_i with respect to each criterion f_j. w_j denotes the weight of criteria j and value of all weights should be $\sum_{j=1}^{n} w_j = 1$.

Table 2. Decision matrix D

	w_j	w_{j+1}	w_n
r_i	f_{ij}	f_{j+1i}	f_{ni}
r_{i+1}	$f_{i+1,j}$	$f_{j+1,i+1}$	$f_{i+1,n}$
. .	. .	. .	. .
r_m	f_{mj}	$f_{m,j+1}$	f_{mn}

4.5. Economic Minimum Execution Time (EMET)

EMET is based on the concept of MET [6] heuristic. MET heuristic assigns a job to the machine that will execute it fastest. It neither considers ready time of machine nor the response time of an application. It allocates an application only on the basis of execution time. We introduce EMET that considers execution time, transfer time and cost of the application. The pseudo code of the EMET is given in Algorithm 3. EMET is similar to EMCT except the fact that the EMET assigns applications on the basis of execution time instead of response time. At step 15, it assigns the execution time plus transfer time of application instead of response time plus transfer time.

5. Evaluation

We simulated the algorithms on GridSim [7] tool kit. Resources are modeled according to specifications given in Table 3. Resources like number of PEs, MIPS and prices are shown in table where the resource's price is not consistent with PE's million instructions per second (MIPS). Grid topology is shown in Fig. 2. We used baud rate, propagation delay and maximum transmission unit as 1000 bps, 10 milliseconds and

Algorithm 2 Pseudo code of TOPSIS

1: Normalize D & its weight whose elements are defined by

$$\mathrm{Z}_{ij} = f_{ij} / \sqrt{\sum_{i=1}^{m} {f_{ij}}^2 \quad i = 1, .., \ m; ; j = 1, \ldots n} \tag{5}$$

2: Formulate the weighted normalized decision matrix whose elements are

$$x_{ij} = w_j * z_{ij}, i = 1, \ldots m; ; j = 1, \ldots n \tag{6}$$

3: Determine idle A^+ and negative idle solution A^-

$$A^+ = (\max \ x_{ij} \ j \in J) \mid i = 1, \ldots, n = \{x_i^+, x_{i+1}^+, \ldots x_n\} \tag{7}$$

$$A^- = (\min \ x_{ij} \ j \in J) \mid i = 1, \ldots, n = \{x_i^-, x_{i+1}^-, \ldots x_n^-\} \tag{8}$$

4: Calculate the separation measures for ideal and negative ideal solutions of each resources as follows:

$$\mathrm{R}_i^+ = \sqrt{\sum_{i=1}^{n} \left(x_{ij} - x_i^+\right)^2 \quad i = 1, .., m} \tag{9}$$

$$\mathrm{R}_i^- = \sqrt{\sum_{i=1}^{n} \left(x_{ij} - x_i^-\right)^2 \quad i = 1, .., m} \tag{10}$$

5: Calculate relative closeness of each resource to the ideal point as follows:

$$C_i^+ = \frac{R_i^-}{R_i^- + R_i^+} \quad = 0 \leq c_i^+ \leq 1; ; i = 1, \ldots m. \tag{11}$$

6: Rank the resources based on the magnitude of closeness C(i)

7: Return the resource that has minimum $C(i)$

Algorithm 3 Pseudo code of EMET

```
1: while application arrives into system do
2:   get an application a_i
3:   matrix[0][0] = aη_i
4:   matrix[0][1] = aδ_i
5:   matrix[0][2] = γ_i
6:   for all resource r_j ∈ R do
7:     for all time-slot t_k ∈ TS do
8:       if (tp_k >= ap_i) then
9:         Ψ_ij = equation 1
10:        c_ij = equation 3
11:        τ_ij = equation 4
12:        break
13:      end if
14:    end for
15:    matrix[j][0] = τ_ij + Ψ_ij
16:    matrix[j][1] = c_ij
17:    matrix[j][2] = tp_k - ap_i
18:  end for
19:  res ← topsis(matrix)
20:  res ← a_i
21: end while
```

1500 bytes respectively. All the resources are simulated as clusters of PE that employ easy backfilling policies and allow advance reservations. The number of CPUs on each resource are chosen such that the demand of CPUs by all applications will always be greater than the total number of free CPUs available on all the resources.

Table 3. Grid resources

Site name	PE	MIPS	Price (G$)
Delhi	100	1140	0.0069
Kolkata	65	1000	0.0032
Madras	252	1200	0.1267
Hyderabad	200	1330	1.856
Bombay	60	1320	0.1424
Pune	54	166	0.0353
Bangalore	265	1176	0.0627
Chennai	20	1140	0.0061
Indore	26	1330	0.1799

Jobs are modeled according to the workload Lubin model [8] and workload DAS2 [21] model. These models analyze well known workloads. Lubin model is analysis of three well known workload archives where DAS2 is analysis of Distributed ASCI Supercomputer-2. These models first apply a logarithmic transformation to the data, due to large range, and then fit it to a novel hyper-Gamma distribution function

and find the parameters of distribution using iterative Expectation Maximization algorithm. These models derive a function for job length, job run time, job inter arrival time and degree of parallelism for batch and interactive jobs.

Application weight can be considered as system defined or user defined. Cost, time and PE weight are generated using uniform distribution method where sum of weights is one. Application run times are generated using a gamma distribution method where mean application length is set and coefficient of variation value is set to 0.9 to test the high variation in length of applications.

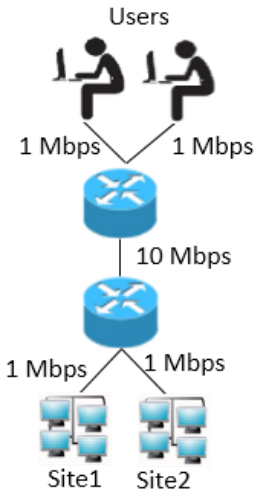

Fig. 2. Grid topology.

The performance evaluation is done based on the following parameters. The average user satisfaction of all applications on given resource is defined as follows:

$$Average\ user\ satisfaction = \frac{\sum_{i=1}^{n} \alpha_{ij}/\eta_i + c_{ij}/\delta_i}{n} \tag{12}$$

Here, n is the number of successful applications executed and j is the resource where application is executed. The objective is to minimize the user benefit because less user satisfaction is better. The results presented are averaged out over ten trials with different resource prices. Simulation work is shown in three scenarios:

- Case-1: The cost and time weight are defined by the user.
- Case-2: The cost and time weight are defined by the user and PE weight is defined by the system.
- Case-3: All weights are defined by the system.

5.1. Case:1

This section shows user satisfaction and number of failures of proposed algorithms with varying number of applications and users.

5.1.1. Effect of Varying Number of Applications

Figure 3 shows normalized user satisfaction. This graph shows that EMCT user satisfaction is less than the EMET, which is better here. As the number of applications increase, the difference also increases. This figure also indicates the difference of user satisfaction between EMCT and EMET algorithms. The maximum gain is 37000 units of EMCT at 10000 applications.

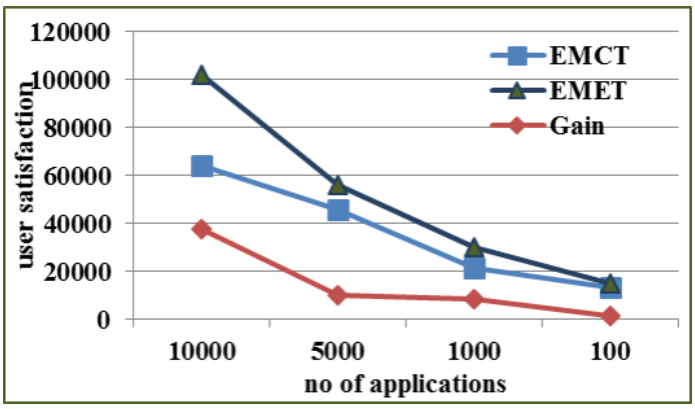

Fig. 3. Effects of user satisfaction with number of applications.

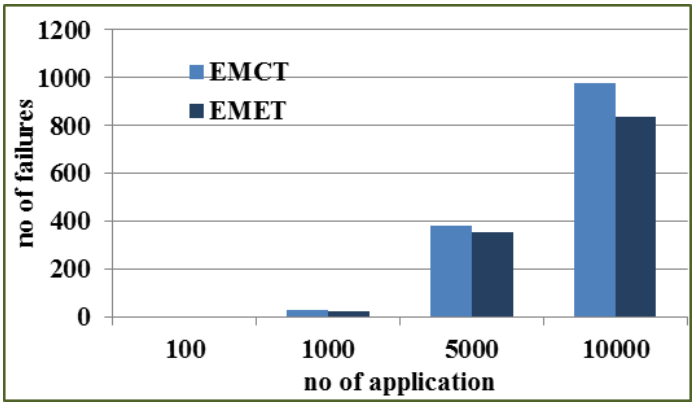

Fig. 4. Effects of failure with number of applications.

Figure 4 exhibits the number of failures with different number of applications. Here, the failure is due to the number of required PE's not available in the system. It can be observed that as the number of applications increase, numbers of failures also increases. EMET has lesser number of failure than EMCT.

5.1.2. Effect of Varying Number of Users

Figure 5 and Figure 6 demonstrate the number of failures and user satisfaction with varying number users. Here, 10000 applications are generated. It can be observed that as the number of users increases, failure decreases because the number of applications per user also decrease. Figure 5 exhibits that EMET has overall 7.9 lesser percentage of failure than EMCT. From Figure 6, it can be observed that EMET is overall 19 percentage more user beneficial than EMCT which is not good sign.

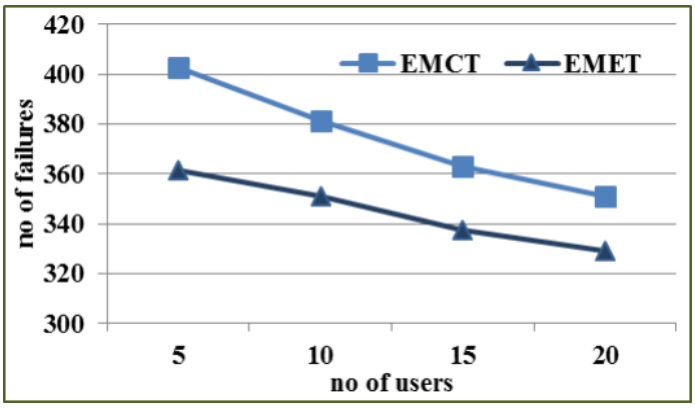

Fig. 5. Effects of failure with different number of user.

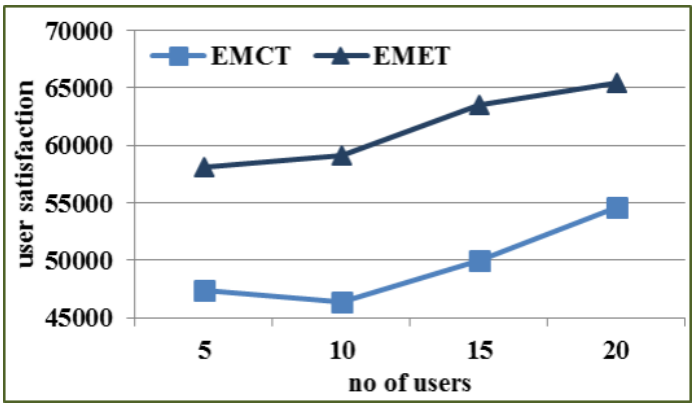

Fig. 6. Effects of user satisfaction with different number of users.

5.2. Case:2

This section discusses the performance of the algorithms with varying PE weight, defined by the system. Cost and time weight are defined by the user. Here, 1000 user applications are generated.

Figure 7 shows that user satisfaction is worse when the PE weight is zero and it is reducing when PE weight is moderate. It can be observed that if PE weight is moderate, the user satisfaction is also moderate. Figure 8 shows that as the PE weight increases the number of failure decreases. Here, we observed that if we take the PE weight moderate then user satisfaction is fair and number of failures are also less.

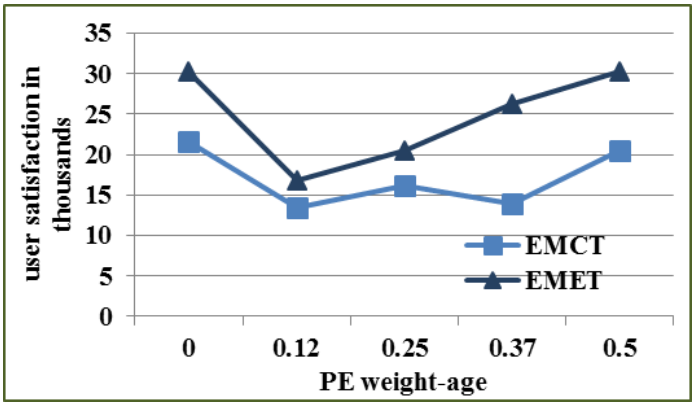

Fig. 7. Effects of user satisfaction with PE weight.

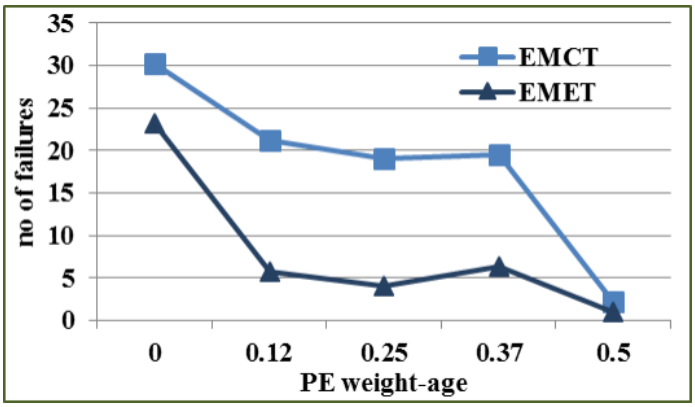

Fig. 8. Effects of failure with PE weight.

5.3. Case:3

This section discusses the performance of the algorithms where all the weight are defined by the system. It shows the performance of proposed algorithm on DAS-2 workload model and system architecture where price is generated randomly. The prices of resources are generated using weibull distribution with parameters 23.577 and 0.5. DAS-2 system architecture is shown in Table 4. Here, 2000 user applications are generated.

Table 4. DAS-2 resources

Site name	PE	MIPS	Price (G$)
Site01	72	1000	Random
Site02	32	1000	Random
Site03	32	1000	Random
Site04	32	1000	Random
Site05	32	1000	Random

5.3.1. Effect of Varying Weight

Figure 9 shows that the numbers of failures of EMET are least when all weight are equal and numbers of failures are moderate when weight of time is more than other parameters. Figure 10 depicts the user satisfaction with different weight of parameters. It also depicts that if time weight is more than the other parameters, user satisfaction is less. It reduces the number of failures and user satisfaction.

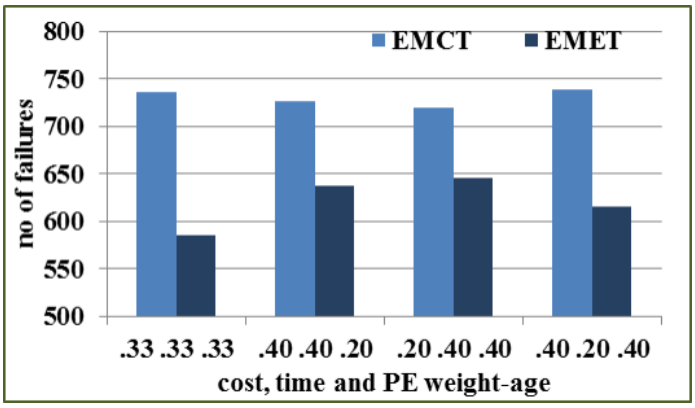

Fig. 9. Effects of failure with system defined weight.

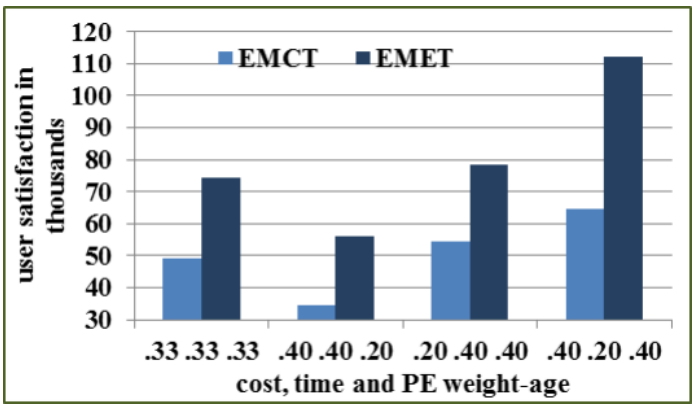

Fig. 10. Effects of user satisfaction with system defined weight.

5.3.2. Effect of Varying Scheduling Interval

Figures 11–14 represent the results of 5000 applications. Applications are generated using system defined weight where the value of time, cost and PE weight are .33, .33 and .33 respectively. In the simulation scheduling interval is weibull distribution. Figure 11 and Figure 12 show the relationship between user benefit and number of failures. From the figures it can be noted that in EMCT user benefit is lesser (better) than EMET which is otherwise a good indication. In EMET failures are lesser than EMCT. Overall in EMCT, 12 percentage user benefit more than that in EMET.

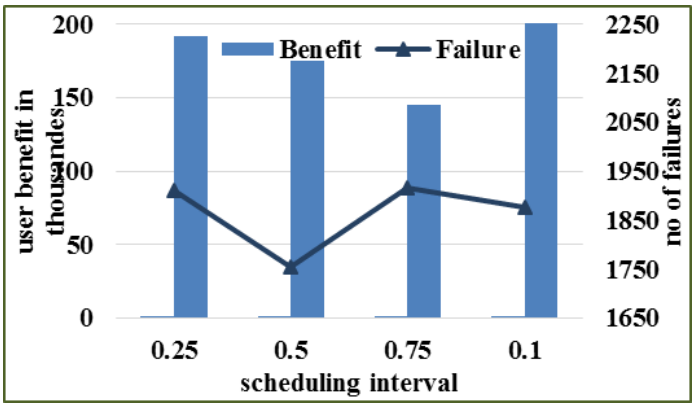

Fig. 11. EMET user satisfaction and failure.

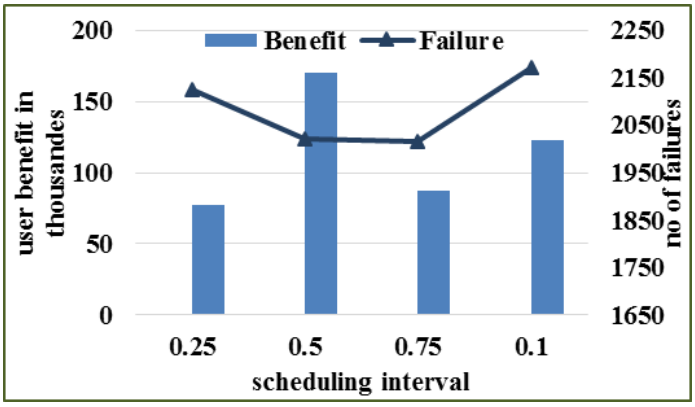

Fig. 12. EMCT user satisfaction and failure.

5.3.3. Effect of Task Distribution

Figure 13 and Figure 14 display the task distribution on different resources. It can be observed from the figure that in EMCT tasks distribution is better than that in EMET. EMCT distributes the task to each resource where EMET makes clustering of tasks. It can be noted that EMCT maximum tasks distribution on particular

resources is less than 38 percentage where EMET tasks distribution on particular resources is more than 40 percentage.

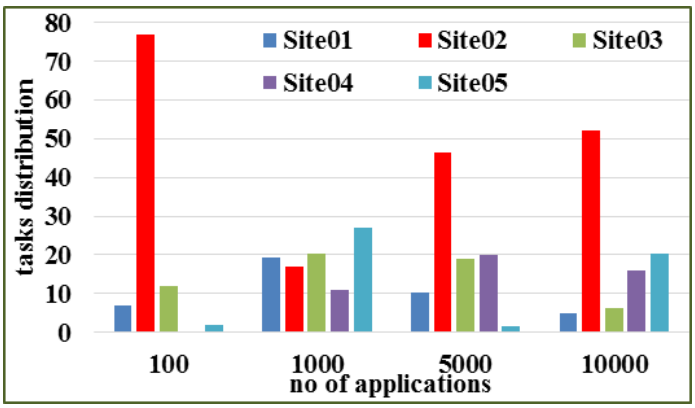

Fig. 13. EMET task distribution.

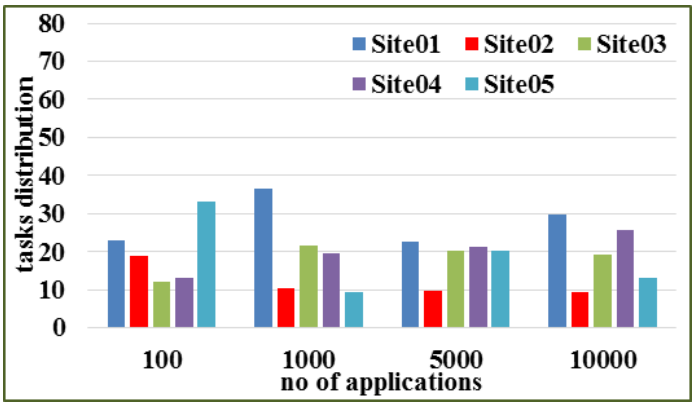

Fig. 14. EMCT task distribution.

6. Conclusion and Future Work

We have modeled the parallel task scheduling algorithms on the Utility Grids where user specified parameters are considered. User specified parameters are used to select best resources as well as to evaluate the user satisfaction. We have compared the proposed EMCT and EMET algorithms and found that as number of applications increase, failure of applications also increase. To reduce the number of failures, we have also considered PE weight that should be system defined. We have found that inclusion of PE weight gives almost the same user satisfaction and less number of failures. It is observed that PE weight should be less than the other weights. EMCT distributes tasks evenly on resources and user satisfaction is less (better) where as

EMET distributes the tasks unevenly and number of failures is less. From the user point of view, EMCT performs better because user does not consider the number of failures. Well known parallel work load models are used to define user behavior. System behavior is also defined using well known architectures. These algorithms involve less overhead and lead to more efficient resource allocation than other optimal resource allocation approaches. In future, we plan to investigate the impact of proposed algorithms on Globus toolkit. We will also concentrate on the concurrent execution of dependent jobs which could not be addressed in the current work.

References

[1] Foster I., Kesselman C., *The Grid: Blueprint for a Future Computing Infrastructure,* Morgan Kaufmann Publishers, USA, 1999.

[2] Thain D., Tannenbaum T., Livny M., *Distributed Computing in Practice: The Condor Experience,* Concurrency and Computation: Practice and Experience, Vol. **17**, 2005, pp. 323–356.

[3] Garg S. K., Buyya R., Siegel H. J., *Cost Trade-Off Management for Scheduling Parallel Applications on Utility Grids*, Future Generation Computer Systems, Vol. **26**, 2010, pp. 1344–1355.

[4] Berman F., Wolski R., Casanova H., Cirne W., Dail H., Faerman M., Figueira S., Hayes J., Obertelli G., Schopf J., Shao G., Smallen S., Spring N., Su A., Zagorodnov D., *Adaptive Computing on the Grid using AppLeS,* IEEE Transactions on Parallel and Distributed Systems, Vol. **14**, Issue 4, 2003, pp. 369–382.

[5] Seymour K., YarKhan A., Agrawal S., Dongarra J., *NetSolve: Grid Enabling Scientific Computing Environments,* in: L. Grandinetti (Ed.), *Grid Computing and New Frontiers of High Performance Processing, Advances in Parallel Computing*, Elsevier, Vol. **14**, 2005, pp. 33–51.

[6] Braun T. D., Siegel H. J., Beck N., *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**, 2001, pp. 810–837.

[7] Buyya R., Abramson D., Giddy J., Stockinger H., *Economic Models for Resource Management and Scheduling in Grid Computing*, Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, Vol. **14**, Issue 13–15, 2002, pp. 1507–1542.

[8] Lublin U., Feitelson D., *The Workload on Parallel Supercomputers: Modeling The Characteristics of Rigid Jobs*, Journal of Parallel and Distributed Computing, Vol. **63**, 2003, pp. 1105–1122.

[9] Ang Li, Nianming Yao, Peiyu Hong, *A Cost and Time Balancing Algorithm for Scheduling Parallel Tasks on Computing Grid*, International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE), Hong Kong, 2010, pp. 185–188.

[10] Bansal S., Hota C., *Efficient Refinery Scheduling Heuristics in Heterogeneous Computing Systems*, Journal of Advances in Information Technology, Academy Publisher, August, Vol. **2**, No. 3, 2011, pp. 159–164.

[11] Kumar A., Chaubey N., Yakkali S., *Immediate Mode Scheduling Methods for Independent Jobs on Open Online Heterogeneous Systems*, 15th International Conference on High Performance Computing, 2009, Bangalore, pp. 12–17.

[12] Sen P., Yang J. B., *Multiple Criteria Decision Support in Engineering Design*, Springer-Verlag Berlin Heidelberg, New York 1998.

[13] Hoschek W., Jaen-Martinez J., Samar A., Stockinger H., Stockinger K., *Data Management in an International Data Grid Project*, First International Workshop on Grid Computing, Bangalore, India, 2000, pp. 1–15.

[14] Wolski R., Spring N. T., Hayes J., *The Network Weather Service: A Distributed Resource Performance Forecasting Service for Meta Computing*, Journal of Future Generation Computing System, Vol. **15**, Issue 6, 1999, pp. 757–768.

[15] Gong L. G., Sun X. H., Watson E. F., *Performance Modeling and Prediction of Nondedicated Network Computing,* IEEE Trans on Computers, Vol. **51**, Issue 9, 2002, pp. 1041–1055.

[16] Golconda K. S., *A Comparison of Static QoS-based Scheduling Heuristics for a Meta-Task with Multiple QoS Dimensions in Heterogeneous Computing*, 18th International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, USA, 2004, pp. 1–14.

[17] Ernemann C., *Economic Scheduling in Grid Computing*, 8th International Workshop Job Scheduling Strategies for Parallel Processing, UK, 2002, pp. 128–152.

[18] He X. S., Sun X. H., Laszewski G. V., *QoS Guided MinMin Heuristic for Grid Task Scheduling*, Journal Computer Science Technology, Vol. **18**, Issue 4, 2003, pp. 442–451.

[19] Mualem A. W., Feitelson D. G., *Utilization, Predictability, Workloads, and User Run-time Estimates in Scheduling the IBM SP2 with Back-filling*, IEEE Transactions on Parallel and Distributed Systems, Vol. **12**, Issue 6, 2001 pp. 529–543.

[20] Deb K., Agrawal S., Pratap A., Meyarivan T., *A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II*, IEEE Trans. Evolutionary Computation, Vol. **6**, Issue 2, 2002, pp. 182–197.

[21] Hui Li, Groep D., Wolters L., *Workload Characteristics of a Multi-cluster, Supercomputer*, 10th International Workshop, JSSPP 2004, New York, NY, USA, Vol. **3277**, 2005, pp. 176–193.

[22] Chitra P., Venkatesh P., Rajaram R., *Comparison of Evolutionary Computation Algorithms for Solving Bi-objective Task Scheduling Problem,* Indian Academy of Science, Vol. **36**, Issue 38, 2011, pp. 167–180.

[23] Beghdad Bey K., Benhammadia F., Mokhtarib A., Guessoumc Z., *Independent Task Scheduling in Heterogeneous Environment via Make-span Refinery Approach*, International Conference on Machine and web Intelligence, Algiers, Algiers, October 2010, pp. 211–217.

[24] Kamalam G. K., Muralibhaskaran V., *A New Heuristic Approach: Min-mean Algorithm for Scheduling Meta-Tasks on Heterogeneous Computing Systems*, International Journal of Computer Science and Network Security, Vol. **10**, No. 1, January 2010, pp. 24–31.

[25] Braun T. D., Siegel H. J., Maciejewski A., *Static Mapping Heuristics for Tasks with Dependencies, Priorities, Deadlines and Multiple Versions In Heterogeneous Environments*, Journal of Parallel Distributed Computing, Vol. **68**, No. 11, 2008, pp. 1504–1516.

[26] Maheswaran M., *Quality of Service Driven Resource Management Algorithms for Network Computing*, International conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, June 1999, pp. 1090–1096.

[27] Martino V. D., Mililotti M., *Scheduling in a Grid Computing Environment using Genetic Algorithms*, International Parallel Distributed Processing Symposis (IPDPS02), IEEE, pp. 235–239.

[28] Zheng S., Shu W., Gao L., *Task Scheduling using Parallel Genetic Simulated Annealing Algorithm*, IEEE International Conference Service Operations Logist. (SOLI06), pp. 46–50.