Prioritized User Demand Approach for Scheduling Meta Tasks on Heterogeneous Grid Environment

P.Suresh
Assistant Professor
Department of IT
Kongu Engineering College
Perundurai, Erode-52,
Tamilnadu

Dr.P.Balasubramanie
Professor
Department of CSE
Kongu Engineering College
Perundurai, Erode-52,
Tamilnadu

P.Keerthika
Assistant Professor
Department of CSE
Kongu Engineering College
Perundurai, Erode-52,
Tamilnadu

ABSTRACT
Due to the rapid evolution of grid computing, which deals with the effective utilization of the globally distributed computer resources to solve massive problems, grid scheduling is the major focus. Efficient scheduling algorithms are the need of the hour to achieve efficient utilization of the unused CPU cycles distributed geographically in various locations. The existing job scheduling algorithms in grid computing had mainly concentrated on the system performance rather than the user satisfaction. In this paper we have presented a new prioritized user demand algorithm that mainly focuses on better meeting the deadlines of the statically available jobs as expected by the users. This algorithm also concentrates on the better utilization of the available heterogeneous resources. The performance analysis shows that the prioritized user demand algorithm performs better than the other heuristic scheduling algorithms in terms of makespan and resource utilization rate.

Keywords: Grid scheduling, User satisfaction, Resource utilization, Makespan, Meta tasks.

1. INTRODUCTION
Even with the emergence of many super fast computers and the high speed networks, the utilization of the geographically distributed resources has gained huge importance. This recognition is mainly because of the low cost services and the best outcome offered by them.

While considering the scheduling of the resources many factors such as CPU utilization rate, throughput, turnaround time, waiting time, response time should be focused for all the processors when assigned with the jobs [13]. The jobs are assigned to the resources considering the system's performance. Thus the scheduling plays an important role in achieving the best utilization of resources and the better completion of the submitted jobs. The scheduling problem is a NP hard problem and the solutions for these problems need heuristics [12]. Many heuristic scheduling algorithms have been designed for this purpose. Even then scheduling is a main focus. There are many algorithms such as MCT, MET, OLB, Min-min, Max-min that are mainly system centric i.e. they consider the effective utilization of resources. But these traditional heuristic algorithms mainly focus on the system performance for each job [14]. In this paper we have presented a new algorithm that considers the time expected for each job by the user and schedules the job by concentrating on both the system performance and the user satisfaction.

In section 2 we will discuss about the related heuristic algorithms for scheduling meta tasks in grid environment such as Opportunistic Load Balancing, Minimum Execution Time, Minimum Completion Time, Min-min, Max-min, Duplex, Genetic Algorithm, Simulated Annealing algorithm, Genetic Simulated Annealing, Tabu, A* algorithms [3] and finally an application demand aware scheduling algorithm [1]. Among all these heuristic algorithms, min-min algorithm gives the best results. In the experimental results section, the proposed heuristic algorithm is tested with the benchmark model of Braun et al [3]. In the section 7 the performance of our prioritized user demand algorithm is compared with application demand aware algorithm to show the reduced makespan and better resource utilization rate that includes user satisfaction.

2. RELATED WORKS
Many researchers have proposed algorithms for static heuristic mapping of independent tasks that resulted in improved resource utilization and makespan. Makespan is the maximum time taken for completing all the submitted jobs.

2.1 Opportunistic Load Balancing (OLB)
OLB allocates jobs in random order to the machine which is idle. This algorithm does not consider the expected execution time of job on the machine. It is very simple but has poor makespan **[**6].

2.2 Minimum Execution Time (MET)
MET allocates jobs in random order to the machine with minimum expected execution time. This algorithm does not consider the expected completion time of jobs on the machine. It assigns each job its best machine but produces severe load imbalance [10].

2.3 Minimum Completion Time (MCT)
MCT allocates jobs in random order to the machine with minimum expected completion time for that job. The performance of this algorithm is better compared to OLB and MET but the same problem of load imbalance occur **[**10].

2.4 Min-min

This heuristic algorithm considers the set of unmapped jobs and calculates the expected completion time for all the jobs in the set in all the machines. For each job, the machine with minimum expected completion time is identified. Finally the job with minimum expected completion time among all the other jobs in the set is allocated to the machine which has the MCT for that job. This process is repeated for the remaining unmapped tasks. In this algorithm, the makespan is comparatively improved but the idleness of the machine remains unsolved [9, 11].

2.5 Max-min

This proceeds as the Min-min algorithm in calculating the expected completion time for all the unmapped jobs in the set and finding the machine with minimum expected completion time for all jobs. The job with maximum expected completion time is allocated to the machine which has MCT for that job. This improves the makespan and balances the load to some extent and performs best for jobs with longer execution time [2].

2.6 Duplex

The Duplex heuristic approach is a combination of Min-min and Max-min algorithms. It performs both the techniques and then uses the algorithm with better solution [3].

2.7 Genetic Algorithm (GA)

The Genetic Algorithm heuristic approach considers a population of n chromosomes. Initially, the population generation is done by two methods. In first method n number of chromosome are generated from a uniform distribution and in the second method called seeding, one chromosome is selected using Min-min approach and the remaining n-1 has random mapping. The best of all the mappings is considered as final solution. The makespan is given by the fitness value within that chromosome. Each chromosome is considered for selection, cross over and mutation. The population is modified and again the process is repeated for remaining chromosomes [18].

2.8 Simulated Annealing (SA)

This heuristic technique considers a single possible mapping for each job at a time. It is an iterative technique in which the probability is based on system temperature. This system temperature gets reduced for each iteration. The mapping here is same as in genetic algorithm and the first mapping is generated form uniform distribution. For each iteration, a new makespan is generated. The performance is poor when compared to Min-min algorithm since SA has poor results in the intermediate stages [16].

2.9 Genetic Simulated Annealing (GSA)

The GSA technique uses selection, crossover and mutation processes as in Genetic Algorithm. In selection process, Simulated Annealing is used for selecting the chromosomes. The concepts of Genetic Algorithm and Simulated Annealing are combinedly used. At the time of mutation or cross over the new chromosome is compared with the original chromosome [16, 18].

2.10 Tabu

Tabu search is for searching in a solution space by keeping track of the regions of solution space that are already searched and there will not be any repeated search near those areas. GA approach is used for mapping. For producing the solution a short hop [18] is performed. Each successful short hop is a solution. After short hop, the new solution is added to the tabu list which keeps track of the solutions that have been already searched. A long hop is performed in which a random mapping is done which differ from each mapping in tabu list. A short hop is performed after each successful long hop [17].

2.11 A*

A* heuristic is a tree search technique which is based on µ-array tree. The root is assumed with a null solution. The child nodes or the intermediate nodes denote the partial mappings and the final mapping is denoted by the leaf nodes. The node with largest cost function is deleted at times in order to reduce the height of the tree. A cost function f (n) is assigned to each node which is calculated on the makespan of its best partial solution [2]. The makespan is calculated as the sum of maximum of machine availability times and lower bound estimate of the difference between the makespan of node's partial solution and makespan of best solution. This is repeated until a leaf node that represents the complete solution is reached.

2.12 Application demand aware

This approach concentrates on user satisfaction by improving the resource utilization and throughput. This is both system centric and application centric [1,15]. The user satisfaction is achieved by allocating most suitable resources to jobs without missing their expected completion time. The expected completion time for each job in all the nodes is calculated. The calculated expected completion time is compared with the minimum completion time of each job that asks for the same node. Depending upon the comparison results, the job with smaller value is allocated to the resource [10].

All these heuristic scheduling algorithms have advantages and also some disadvantages. The Opportunistic load balancing algorithm does not consider the expected execution time and henceforth its makespan is poor. Minimum Execution time heuristic does not consider completion time of jobs that leads in severe load imbalance. MCT also leads in poor makespan [4]. Max-min heuristic performs better when compared to all these algorithms only for shortest jobs. The other heuristics such as duplex, Simulated Annealing, Genetic simulated annealing, Tabu, A*, Genetic Algorithm performs less [7,8]. Among all these heuristic algorithms discussed earlier Min-min heuristic is simple, fast and performs better while considering the system performance by reduced makespan but user satisfaction is not considered. Application demand aware performs better when user satisfaction is taken into account.

The experimental results show that our proposed algorithm has reduced makespan, highest hit rate when compared to application demand aware algorithm.

3. PROBLEM DEFINITION

The problem of job scheduling of meta-tasks in heterogeneous computing environment is presented. The experimental study is done based on the benchmark simulation model by Braun et al. [3].

In our model, static heuristic mapping is considered for mapping meta-tasks. Each machine executes a single task at a time. In heterogeneous computing environment, the size of meta-tasks and the number of machines are known priori. Also the expected execution time of each job in every machine is known priori which is in ETC (Expected Time to Compute) matrix. $ETC(t_i,m_j)$ is the estimated execution time of task i on machine j.

The Expected Completion time for each task t_i on each machine R_j is given by

$$ECT(ti,Rj) = RT(Rj) + ETC(ti,Rj)$$

where RT is the ready time of each machine R_j . Ready Time of a machine can be defined as the time needed by a machine to complete already allocated jobs on it.

The makespan which is defined as the maximum time taken to complete all the jobs is given by

$$makespan = max(ECT(ti,Rj))$$

The problem of scheduling meta tasks to resources must include the following.

1. The number of meta tasks that are to be scheduled
2. The number of resources available in the grid for processing the meta tasks
3. The processing capacity of each resource in MIPS
4. The size of jobs in millions of instructions
5. ETC matrix of size R×T where T is the number of tasks and R is the number of resources.

4. SCHEDULING MODEL

Grid scheduling is the process of scheduling application tasks over grid resources. There are two main concepts in this scheduling process namely system's performance and user satisfaction. Grid Scheduler acts as a medium to receive tasks from various users and allocate the appropriate resources [5]. An efficient scheduler must improve the overall system performance and reducing the waiting time for individual task.

Our algorithm is both system centric and application centric. The factors of our algorithm made it system centric are resource utilization and throughput. Resource utilization is defined as the percentage of a given period that measures the busyness of the resource. Throughput is given by the amount of jobs processed by a resource in a given period of time. The factor that the priority is given to the user's demand while scheduling makes it application centric. These factors aim at optimizing the performance of each application.

Our algorithm mainly deals with the statically available jobs and hence it is of static scheduling mode. In particular our algorithm deals with a list of jobs at a time and has two phases in scheduling such as task prioritizing and resource selection. The task prioritizing phase sets the priority of each task with the user deadline as the parameter and generates a scheduling list by sorting the tasks according to their priority. The resource selection phase selects tasks in the order of their priorities and maps each selected task to its optimal resource.

Let us consider the mathematical representations to denote the relationships between the resources and jobs and also to introduce the parameters involved in our algorithm such as execution time, completion time, ready time, etc.

Notation	Definition
CTi,j	Completion time of the job or task ji in the resource rj
RTj	Ready time of the resource rj
ETi,j	Execution time of the job or task ji in the resource rj
$DCTi,j$	Difference in time between the deadline given by the user and the calculated completion time for the job in available resources
$MinDCTi,j$	The minimum value from the difference values $DCTi,j$ for the given job
UTi	User requisition time or the deadline given by the user for the jobs in U
$ECT\ i,j$	Expected Completion Time of task ji in resource rj

(1) The resource set is represented as $P = \{r1,r2,r3,\ldots\ldots\ldots,rm\}$. As the grid environment deals with the heterogeneously distributed grid resources the number of resources available may be huge. As we consider the static environment both the jobs submitted and the resource available are taken as fixed and they do not change over time.

(2) The jobs submitted can be enclosed within the job set which is represented as $U = \{J1,J2,J3,\ldots\ldots\ldots\ldots,Jn\}$. The jobs submitted are considered as the independent tasks that can be executed in parallel with other available tasks. Also the jobs are considered as static i.e. they number of tasks submitted are fixed and they do not change with time.

(3) The users submitting their jobs for execution are represented as $C = \{C1, C2, C3, \ldots\ldots\ldots\ldots, Cp\}$. The users submit the jobs with the requisition time i.e. within which the job needs to be completed which can also be called the demanded deadline of the user for the submitted jobs.

5. PROPOSED PRIORITIZED USER DEMAND (PUD) ALGORITHM

In this section, we present the brief description of our algorithm which is based on user satisfaction and system performance. It takes user's deadlines into account and makes the job to be executed within the expected deadline by assigning it to the most suitable resource. It also concentrates on the system performance by reducing the idle time of the resources and distributing the unmapped tasks equally among the available resources. It considers the ETC matrix and concentrates on the completion time and hence the system's performance is also the major consideration in addition to user's satisfaction.

Here we perform the scheduling process in two major steps. In the first step, we concentrate on the user satisfaction and in the second step we consider system performance. The ETC matrix is constructed for the available resources with every available resource. Secondly we consider the job with the minimum deadline i.e. the job that needs to be completed quickly. Then we compare the deadline of the selected job given by the user with that of different ETC values. Then allocate the job to the resource that has the minimum difference value. Then remove the job from the job set. Then the waiting time of the resource is changed and the ETC matrix is recalculated for the remaining unmapped jobs. Then continue the above steps until all the jobs are scheduled. Thus both the user satisfaction and system performance can be taken into consideration effectively with this algorithm. Now we will give the detailed algorithm description.

(1) While there are jobs to be allocated do
(2) For each job J_i to be scheduled, $J_i \in U$, $1<i<n$,
  Do
    For each resource r_j available, $r_j \in R$, $1<j<m$,
    Do
      $CT_{i,j} = ET(J_i, r_j) + RT(r_j)$
    End for
  End for
(3) Job= Min (UT_{ij}) where $1 \leq i \leq n$
(4) For each job J_i in U
  Do
    If J_i = Job then
    Begin
      For each resource r_j in R
      Do
        $DCT_{i,j} = UT_i - CT_{i,j}$
        If $DCT_{i,j} < 0$ then
        Begin
          $DCT_{i,j}$ = a maximum value
        End if
      End for
    $MinDCTi,j = Min\ (DCTi,j)$
      where $1 \leq j \leq m$
    Allocate the job J_i to the resource r_j
    Remove J_i from U
    Update the ready time of the resource r_j
  End if
  End for
(5) Repeat, until all jobs are assigned to resources
(6) End while

Let us consider 5 meta-tasks (T1, T2, T3, T4, T5) that must be executed on 3 heterogeneous resources (R1, R2, R3). The user deadline is given for each task as below.

Tasks	User Deadline (UT)
T1	9
T2	10
T3	5
T4	15
T5	12

The ETC matrix is constructed below.

	R1	R2	R3
T1	8	10	6
T2	15	18	12
T3	3	7	5
T4	10	7	9
T5	17	10	11

In this example, the user deadline for each task is given and expected execution time for each task T_i in every available resource R_j is calculated.

The Application Demand aware algorithm allocates task T3 to R1, T5 to R2, T1 to R3, T4 to R1 ,T2 to R3 with a makespan of 21 ms.

The proposed prioritized user demand algorithm works as follows. The task with minimum deadline is chosen and the difference value is calculated. The task is allocated to the resource with minimum difference value. The task T3 is

allocated to the resource R3, T1 to R1, T2 to R3, T5 to R2 and T4 to R3 with a makespan of 17 ms.

The makespan results of both Application demand aware algorithm and Prioritized user demand algorithm is shown figure 5.1.

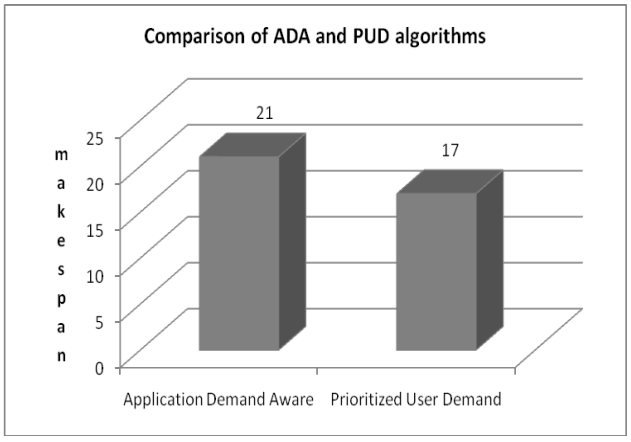

Figure 5.1 Comparison of ADA and PUD algorithm

From the comparison result, our proposed algorithm proves better with reduced makespan. We have presented our algorithm based on user deadline satisfaction in detail, and it can ensure most jobs to be completed within their expected completion time. Even though the user satisfaction is taken mainly into consideration, the system performance is preserved to a great extent.

6. EXPERIMENTAL RESULTS

The experimental results are based on the benchmark of instances by Braun et al. [3, 6]. The factors such as task heterogeneity, machine heterogeneity and consistency are considered while constructing ETC matrix. The task heterogeneity depends upon the various execution times of the jobs. The machine heterogeneity depends on the running time of a particular job across all the processors. Both machine and task heterogeneity can have the values high and low. Three ET consistencies such as consistent, inconsistent, and semi consistent are considered. An ETC matrix is said to be consistent, if a resource R_i execute a task Ti faster than the resource R_k, and R_i executes all other jobs faster than R_k. An inconsistent matrix is one in which if a resource R_i executes some jobs faster and some slower than R_j. A semi consistent matrix is a sub matrix of inconsistent matrix with a predefined size. The three consistencies are given by

c -consistent
s -semi consistent
i - inconsistent

The instances of bench mark problems are classified into twelve different types of ET matrices. Each ET matrix consists of 100 instances. The instances depend upon task heterogeneity, machine heterogeneity and consistency. The instances are labeled as u_x_yy_zz.k where

u -uniform distribution, used to generate the matrix.
x -type of consistency (c/i/s)
yy-indicates the heterogeneity of the tasks.
(hi-high task, lw-low task)
zz-indicates the heterogeneity of the resources
(hi-high machine, lw-low machine)

ETC matrix is constructed with 512 jobs and 16 machines for all the instances. The makespan is computed for both application demand aware and prioritized user demand techniques.

7. PERFORMANCE ANALYSIS

The efficiency of prioritized user demand algorithm is proved by comparing the results with application demand aware algorithm tabulated in table 6.1 based on the benchmark instances. The makespan is calculated for all the 12 different types of instances. The comparison results show that the prioritized user demand algorithm has reduced makespan than application demand aware algorithm.

Table: 7.1 Makespan for Application Demand Aware(ADA) and Prioritized User Demand Algorithm (PUD)

Instances	Application Demand Aware Algorithm	Prioritized User Demand Algorithm
u_c_hi_hi	9618108	2308179
u_c_hi_lw	735913	469402
u_c_lw_hi	84511	58645
u_c_lw_lw	11583	4947
u_i_hi_hi	6804441	5403760
u_i_hi_lw	1061204	855583
u_i_lw_hi	145245	122041
u_i_lw_lw	10591	5429
u_s_hi_hi	5774893	4504474
u_s_hi_lw	1470189	546691
u_s_lw_hi	106230	60084
u_s_lw_lw	15524	13854

Figure 7.1 shows the comparison results of makespan for high task high machine for consistent, inconsistent and semi consistent values. Figure 7.2 shows the comparison results of makespan for high task low machine. Figure 7.3 shows the comparison results of makespan for low task high machine. Figure 7.4 shows the comparison results of makespan for low task low machine.

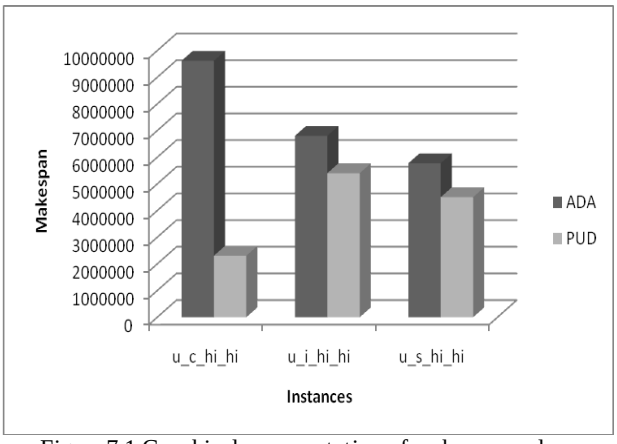

Figure 7.1 Graphical representation of makespan values (arbitrary time units) for High Task High Machine

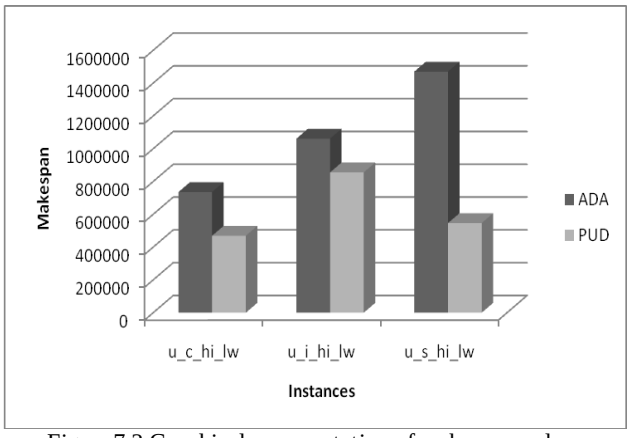

Figure 7.2 Graphical representation of makespan values (arbitrary time units) for High Task Low Machine

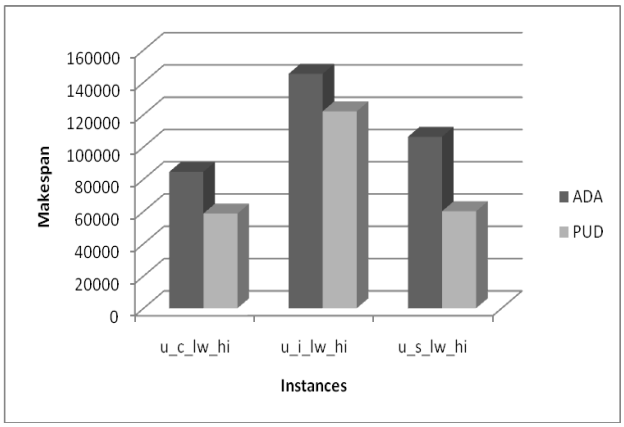

Figure 7.3 Graphical representation of makespan values (arbitrary time units) for Low Task High Machine

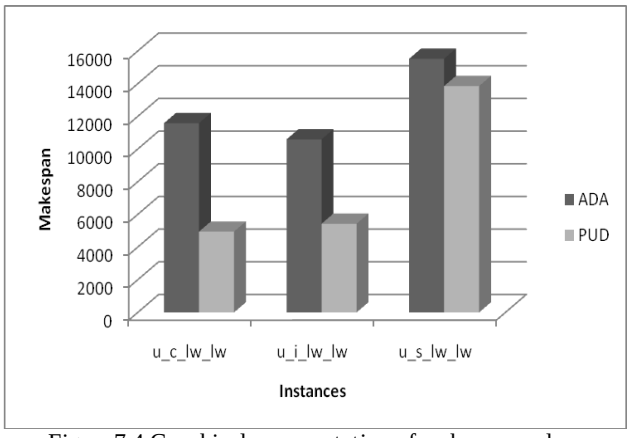

Figure 7.4 Graphical representation of makespan values (arbitrary time units) for Low Task Low Machine

The above results show that PUD algorithm has less makespan than ADA algorithm. The percentage of improvement of PUD over ADA is given by the table 7.2.

Table: 7.2 The percentage of makespan values of PUD over ADA

Instances	**Improvement of PUD over ADA (%)**
u_c_hi_hi	76%
u_c_hi_lw	36.22%
u_c_lw_hi	30.61%
u_c_lw_lw	57.29%
u_i_hi_hi	20.59%
u_i_hi_lw	19.38%
u_i_lw_hi	15.98%
u_i_lw_lw	48.74%
u_s_hi_hi	21.99%
u_s_hi_lw	62.8%
u_s_lw_hi	43.44%
u_s_lw_lw	10.76%

8. CONCLUSION AND FUTURE WORK

The proposed prioritized user demand algorithm is implemented and tested with benchmark simulation model for heterogeneous systems by Braun et al. The experimental results and preference analysis show that PUD algorithm preference better with reduced makespan than applications demand a wave algorithm. Proposed Our algorithm delivers reduced makespan on various

heterogeneous environments such as task heterogeneity which includes high & low task, machine heterogeneous which includes high & low task, machine and consistency which includes consistent, inconsistent, semi consistent values. This proposed research focuses of static heterogeneous environment on independent tasks. This can be further enhanced for dependent tasks in which jobs are dependent on others. In addition, factors other than makespan such as common delay, CPU load factor can also be considered.

REFERENCES

[1] Jie Lin, Bin Gong, Hui Liu, Chaoying Yang, Yuhui Tian, 2007. An Application Demand aware Scheduling Algorithm in Heterogeneous Environment. IEEE Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design., IEEE Xplore press, Melbourne, Vic, pp509-604, DOI:10.1109/CSCWD.2007.4281504

[2] Li Wenzheng, Zhang Wenyue, 2009. An Improved Scheduling Algorithm for Grid Tasks. International Symposium on Intelligent Ubiquitous Computing and Education, pp 9-12, DOI:10.1109/IUCE.2009.35

[3] Tracy D.Braun, Howard Jay Siegel, and Noah Beck, 2001. 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, DOI:10.1006/jpdc.2000.1714

[4] Ivan Rodero, Francesc Guim, and Julita Corbalan, 2009. Evaluation of Coordinated Grid Scheduling Strategies. 11th IEEE International Conference on High Performance Computing and Communications, DOI:10.1109/HPCC.2009.28

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

[6] T.Braun, H.Siegel, N.Beck, L.Boloni, M.Maheshwaran, A.Reuther, J.Robertson, M.Theys, B.Yao, D.Hensgen, and R.Freund, 1999. 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), IEEE Computer Society Washington, DC, USA. pp.15-29. DOI:10.1109/HCW.1999.765093

[7] R.F.Freund, and M.Gherrity, 1998. Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net. In Proceedings of the 7th IEEE HCW, DOI:10.1109/HCW.1998.666558

[8] Thomas G.Robertazzi and Dantong Yu, 2006.Multi-Source Grid Scheduling for Divisible Loads. 40th Annual Conference on Information Sciences and Systems, Princeton University, IEEE, DOI:10.1109/CISS.2006.286459

[9] Zhang Qian, Li Zhen, 2009. 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, DOI:10.1109/ETCS.2009.670

[10] Hojjat Baghban, Amir Masoud Rahmani, 2008. 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, DOI:10.1109/GCC.2008.22

[11] He Xiaoshan, Xia-He Sun, Gregor Von Laszewski, 2003. QoS Guided Min-min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology, pp. 442-451, DOI:10.1007/BF02948918

[12] He Xiaoshan, Xia-He Sun, Gregor Von Laszewski, 2003. QoS Guided Min-min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology, pp. 442-451, DOI:10.1007/BF02948918.

[13] Fangpeng Dong and Selim G. Akl, 2006. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report, School of Computing, Queen's University, Canada.

[14] Y. Zhu, 2003. A Survey on Grid Scheduling Systems, Department of Computer Science, Hong Kong University of science and Technology.

[15] Wantao Liu, Rajkumar Kettimuthu, Bo Li, Ian Foster, 2010. An Adaptive Strategy for Scheduling Data-Intensive Applications in Grid Environments. IEEE 17th International Conference on Telecommunications, DOI:10.1109/ICTEL.2010.5478755

[16] H.Chen, N.S.Flann, and D.W.Watson, 1998. Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Approach. IEEE transactions on Parallel and distributed Computing, 9(2), pp. 126-136, DOI:10.1109/71.663870

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

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