# Economical Task Scheduling Algorithm for Grid Computing Systems

Amit Agarwal, Padam Kumar

GJCST Classification (FOR)
D.4.1, F.1.2

***Abstract*- Task duplication is an effective scheduling technique for reducing the response time of workflow applications in dynamic grid computing systems. Task duplication based scheduling algorithms generate shorter schedules without sacrificing efficiency but leave the computing resources over consumed due to the heavily duplications. In this paper, we try to minimize the duplications of tasks from the schedule obtained using an effective duplication based scheduling heuristic without affecting the overall schedule length (makespan) of grid application. Here, we suggested an economical duplication based intelligent scheduling heuristic called economical duplication scheduling in grid (EDS-G). The simulation results show that EDS-G algorithm generates better schedule with lesser number of duplications and remarkably less resource consumption as compared with HLD, LDBS in the simulated heterogeneous grid computing environments**

*Keywords*- scheduling, grid computing, duplication based heuristic, DAG, workflow applications.

## I. Introduction

In recent years, grid computing has obtained a lot of attentions from engineers and scientists for executing high performance parallel and distributed applications due to major advancements in wide-area network technologies and low cost of powerful computing and high-speed network resources. In general, a parallel and distributed application can be represented by a weighed directed acyclic task graph (DAG) as shown in figure 1. In DAG, the nodes represent application tasks and the edges represent inter-task data dependencies. The algorithm for finding an optimal schedule for the multiprocessor scheduling problem is NP-complete [1, 2, 3]. In literature, task scheduling heuristics for DAG applications have been classified as list scheduling [4, 5, 6] cluster-based scheduling [7, 8] and duplication-based scheduling [9, 10, 11, 12, 13, 14, 15, 16, 17]. In list-based task scheduling, tasks are ordered in non-increasing order of their priorities (or ranks) and scheduled on the resource which minimizes the objective function such as schedule length. Clustering is an efficient way to reduce communication delay in DAGs by grouping heavily communicating tasks to same labeled cluster and then assigning tasks in a cluster to the same resource. In duplication-based scheduling, parents of current selected task can be duplicated into idle time slots between two already scheduled tasks in order to reduce the task finish/ start time. Duplication-based scheduling is very effective in distributed computing system but leave the computing resources over consumed due to the heavily duplications.

*About-Department of Electronics & Computer Engineering Indian Institute of Technology, Roorkee (India) {aamitdec, padamfec} @iitr.ernet.in*

Duplication heuristics are more effective for fine grain task graphs and for networks with high communication latencies. The term CCR refers to the ratio of average communication cost to average computation cost on a given system. A high CCR indicates the communication intensive nature of a problem, whereas, low CCR represents the computation intensive problem. Duplication plays its role more effectively at higher CCRs, as the formation of large sized scheduling holes increases with higher communication costs, which can be exploited to accommodate fine grain tasks conveniently [10]. In heterogeneous distributed computing system, heterogeneity of computational resources and communication mechanisms poses some major obstacles to achieve high parallel efficiency. Performance of the scheduling algorithms tends to degrade in the presence of heterogeneity. This degradation becomes more pronounced with an increase in heterogeneity and at higher CCRs which results in inappropriate task/ processor selection. In this case, duplication is very graceful to overcome these „stresses' and „strains' of heterogeneity by duplicating the crucial tasks and thereby improving the finish time on processing resource, but it increases scheduling cost due the duplicated tasks overhead. In [16], Savina et al. suggested the heterogeneous limited duplication (HLD) that adapts the SD algorithm [10] heterogeneous environment and then assessed the usefulness of limited duplication approach in dealing with the stresses of heterogeneity in a system. In [17], Dogan et al. proposed a level sorting algorithm (LDBS) to arrange the tasks in DAG into various precedence levels. The tasks belonging to the same level have no data dependencies can be executed concurrently. In LDBS, tasks are scheduled level by level starting from the top. In current economic market models [18, 19], economic cost (cost of executing a workflow on grid) has been considered as an important scheduling criterion to employ the user-centric policies, since different resources, belonging to different organizations, may have different polices of charging. Hence, the economic cost of resource consumption for scheduling the grid applications becomes an important performance metrics for analyzing the scheduling algorithms.In EDS-G algorithm, we analyze the impact of the duplicated tasks over makespan and try to optimize the schedule generated with duplication by eliminating tasks (duplicated and unproductive) as much as possible without affecting the makespan. A task may become unproductive after being duplicated if its immediate successor tasks can gather output data from its duplicated parent task not later than the unproductive task. These algorithms can prove that they are very useful in the distributed grids to reduce the scheduling cost of an

application and improving the performance of the grid system.The remainder of this paper is organized as follows. Section II defines the task scheduling problem. Section III presents the proposed task scheduling algorithms. Section IV shows the simulation results. Then, in section V, we describe our conclusion of current research work.

## II. TASK SCHEDULING PROBLEM

A task scheduling model for grid computing system includes an application of dependent tasks (DAG); a target grid computing system of arbitrary connected multiple computing resources and an objective function.

### 1) *Grid Resource Model*

A grid computing system can be represented by $G=(R,Q)$ where $R$ is the set of *m* arbitrary connected computing resources $(r_1, r_2, .....r_m)$ forming a grid and $Q$ is the set of communication channels connecting the grid resources. In grid computing system, task execution cost on different grid resources may be different due to the processor heterogeneity (different processing rates of grid resources) and similarly, the data transfer rates (bandwidths) between different pair of processing resources may be different due to network heterogeneity. In this model, it is also assumed that each processing resource has co-processor to deal with communications, which allows computation and communication to overlap each other. Additionally, task executions are assumed to be non-preemptive and communication overhead between two tasks scheduled on the same resource is considered as zero. After completing execution of a task, the associated grid node sends output data to all of its child tasks in parallel.The main objective of this paper is to minimize duplications after duplicating tasks over grid resources selectively and minimize the overall schedule cost (Resource Consumption). A resource consumption can be defined as the fraction of time duration (between after being allotted to an application and released for another application) a resource is actually executing some tasks of application in grid.

| Task Node | Computation costs on different grid nodes | | | | Mean cost |
|---|---|---|---|---|---|
| | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $\bar{\tau}_i$ |
| $n_1$ | 1 | 1 | 2 | 1 | 1.25 |
| $n_2$ | 3 | 2 | 4 | 2 | 2.75 |
| $n_3$ | 5 | 6 | 3 | 4 | 4.5 |
| $n_4$ | 2 | 4 | 4 | 2 | 3.0 |
| $n_5$ | 4 | 8 | 7 | 8 | 6.75 |
| $n_6$ | 3 | 3 | 1 | 2 | 2.25 |
| $n_7$ | 5 | 5 | 5 | 5 | 5.0 |
| $n_8$ | 1 | 2 | 2 | 2 | 1.75 |

Table 1. Computation cost matrix [ $\tau_{ij}$ ] for DAG in fig. 1.

### 2) *Grid Application Model*

A grid application may be represented by a weighted directed acyclic graph (see figure 1) or DAG, $D=(N,E,T,C)$ where $N$ is a set of $n$ computation task nodes, $T$ is a $n \times m$ computation cost matrix (see table 1) and the value of $\tau_{ij} \in T$ is theexpected time to execute task $n_i$ on grid resource $r_j$ *for* $1 \le i \le n$ and $1 \le j \le m$, $E$ is a set of communication edges that shows precedence constraints among the tasks and $C$ is a $n \times n$ communication cost matrix and the value of $c_{ij} \in C$ is the expected time to communicate data from task $n_i$ to task $n_j$ for $1 \le i \ne j \le n$. The mean computation cost $\bar{\tau}_i$ of task $n_i$ and mean communication cost $\bar{c}_{ij}$ between task $n_i$ and task $n_j$ can be calculated as

$$\bar{\tau}_i = \frac{\sum_{j=1}^{m} \tau_{ij}}{m} \qquad \forall 1 \le i \le n \qquad (1)$$

$$\bar{c}_{ij} = \frac{c_{ij}}{mean\ data\ transfer\ rate\ over\ all\ links\ in\ grid} \qquad \forall\, 1 \le i \ne j \le n \qquad (2)$$

A task node without any parent node is called entry task and a task node without any child node is called exit task. If there are two or more entry (exit) tasks, they may be connected to a zero-cost pseudo entry (exit) task with zero-cost edges which will not affect the schedule. Since the intra-processor bus speed is much higher than the inter-processor network speed, the communication cost between two tasks scheduled on the same processing node is considered as zero [20].

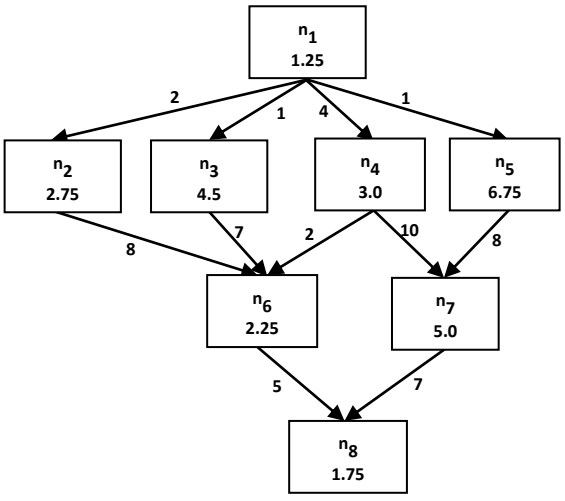



Fig. 1. A Simple DAG with Precedence Constraints.

### *III.* ECONOMICAL DUPLICATION BASED SCHEDULING

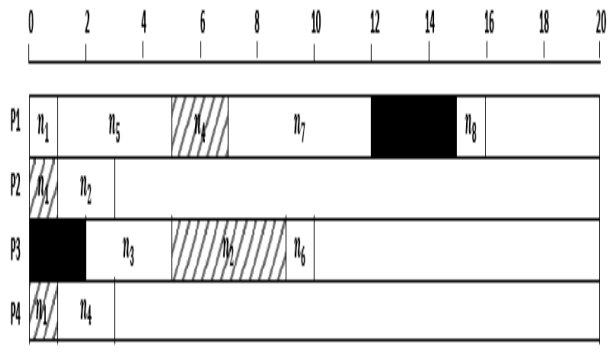
This section presents the economical duplication based scheduling algorithm (EDS-G) in grid computing environment inspired from our earlier work in [21]. This algorithm consists of two mechanisms, first is a lower-bound complexity mechanism for scheduling based on insertion based task duplication and second is modifying schedule after removing some duplicated and unproductive tasks in the schedule without affecting the makespan.In this section, a lower-bound complexity algorithm (EDS-G) for grid computing system has been presented. The pseudo code of the algorithm is shown in figure 3. A priority-based task sequence is generated by ordering the tasks in non-increasing order of their b-level (computation and communication cost along the longest directed path from the concerned task to the exit task in DAG) that can be calculated recursively using mean cost parameter as:

$$b_i = \bar{\tau}_i + \max\{b_j + \bar{c}_{ij}\} \quad \forall n_j \in succ(n_i) \qquad (3)$$

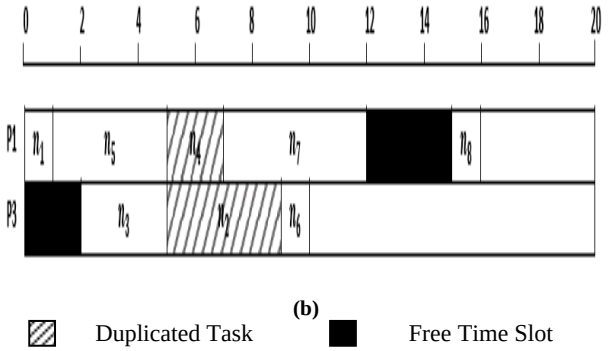
The term $succ(n_i)$ refers to the set of immediate child nodes of task $n_i$ in the DAG. Now, the first unscheduled task in the task sequence is selected and scheduled on a grid resource that can finish its execution at the earliest using duplication (task replication) approach. This algorithm uses insertion based scheduling policy which considers the possible insertion of a task or duplicated task in an earliest idle time slot between two already scheduled tasks on the grid resource. A task on the grid resource can start execution only after the data arrived from all of its immediate predecessors. The parent of task $n_i$ whose data arrives last of all is termed as the most important immediate parent (MIIP). Data arrival time for $n_i$ on $r_k$ is given by:

$$DAT(n_i, r_k) = \max_{n_j \in pred(n_i)} \{\min\{F_{jk}, F'_{jk} + c_{ji}\}\} \qquad (4)$$

The term $pred(n_i)$ refers to the set of immediate parent nodes of task $n_i$ in the DAG.




**(a)**




**(b)**

Duplicated Task  Free Time Slot

Fig. 2. Gantt Charts for the schedule generated by (a) HLD Algorithm (Duplications = 4, Resource Used = 4) (b) EDS-G Algorithm (Duplications=2, Resource Used = 2) for an application DAG shown in fig. 1.

Due to the non-availability of data earlier, owing to precedence constraints or communication delay, a grid resource may remain idle leading to the formation of scheduling holes. These scheduling holes may be exploited to duplicate tasks to minimize data arrival time. The start time $S_{ik}$ of task $n_i$ on grid resource $r_k$ is limited by the data arrival from its MIIP (say $M_i$) and availability of a suitable scheduling hole. If suitable slot is not available then task $n_i$ can start after the completion of last scheduled task on grid resource $r_k$, i.e. the ready time ($r_k^R$) of resource $r_k$. The start time of task $n_i$ on resource $r_k$ is given by:

$$S_{ik} = \max\{DAT(M_i, r_k), \min\{r_k^R, G_r^S\}\} \qquad (5)$$

where $G_r^S$ is the start time of first suitable free time slot $G_r$ to accommodate task $n_i$ on the resource $r_k$, if exist. The finish time $F_{ik}$ is calculated as:

$$F_{ik} = S_{ik} + \tau_{ik} \qquad (6)$$

The finish time is calculated for all the available grid resources and task $n_i$ is scheduled on the resource that gives earliest finish time. After scheduling the all tasks, makespan is calculated as:

$$makespan = \max\{F_{ik}\} \quad \forall 1 \le i \le n; 1 \le k \le m \qquad (7)$$

**EDS-G Algorithm**

**Begin**

**1:** Construct a priority based task sequence $\xi$ ;
**2:do** {

**3:** Select the first unscheduled task $n_i$ in the task sequence $\xi$

**4:for** (all $r_k$ in processor list $R$ ) {

**5:** Sort the list of immediate parents of $n_i$ in non-increasing order of data arrival time;
**6:for** all immediate parents, select the first immediate parent

$n_j$ from the list at Step 5 {

**7:if** duplication of $n_j$ can reduce finish time $F_{ik}$ of $n_i$ on $r_k$

**8:** Duplicate $n_j$ ;
**9:** }

**10:** Compute earliest finish time $F_{ik}$ of $n_i$ on $r_k$ using eq.(6);
**11:** }

**12:** Find the minimum earliest finish time of $n_i$ ;

**13:** Assign $n_i$ on resource $r_k$ with minimum $F_{ik}$ in schedule **S**;

**14:** }**while** (there are unscheduled tasks in the task sequence $\xi$ );
**15:** Maintain a list **A** of origin tasks which have been duplicated later with their successor links to other tasks and list **B** of duplicated tasks in non-increasing order of their earliest start time;

**16:for** (each duplicated task $n_i$ in list **B**) {
**17:if** (no change in makespan of schedule **S** after removing

duplicated task $n_i$ )

**18:** Remove this duplicated task $n_i$ from the schedule **S** and update list **A**;
**19:** }

**20:for** (each task $n_i$ in list **A**) {

**21:if** task $n_i$ has no dependent task in schedule **S** due to its

duplication later on different processors; remove this task $n_i$ from the schedule **S**
**22:** }

Fig. 3. Pseudo Code for EDS-G Algorithm

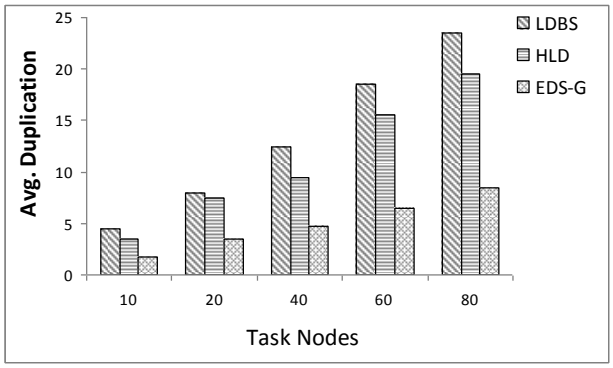
Further, we maintain a list A of origin tasks (which have been duplicated later in the schedule) with their links to dependent tasks and list B of duplicated tasks in non-increasing order of earliest start time. The above schedule is modified only if the removal of the duplicated task from list B does not affect the makespan. Similarly, all the unproductive tasks in the list A, that are not responsible to provide any output to immediate successor tasks due to theirduplications, are removed from the schedule. This modified schedule contains lesser number of duplications and remarkably less resource consumption as compared with HLD, LDBS for heterogeneous grid computing systems. Gantt charts for the schedule generated by HLD algorithm and EDS-G Algorithm is shown in fig. 2. Here in HLD schedule, task $n_2$ on resource $r_2$ and task $n_4$ on resource $r_4$ are unproductive after being duplicated on resource $r_2$ and $r_1$ respectively. Hence in EDS-G schedule, tasks $n_1$ (which was scheduled on $r_2$ for $n_2$ ), $n_2$ from resource $r_2$ and tasks $n_1$ (which was scheduled on $r_4$ for $n_4$ ), $n_4$ from resource $r_4$ have been removed. It shows that EDS-G uses less duplications and lesser number of resources as compared to HLD for the same makespan.

## IV. Simulated Results and Analysis

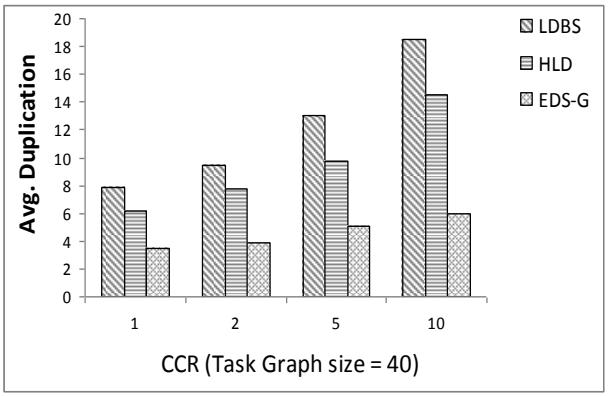
The experimental results for random task graphs are presented in fig. 6 and 7 for the clique topology for different task graph sizes and CCRs in the simulated grid environments. The performance of EDS-G algorithm is analyzed with respect to various graph characteristics (task sizes, CCRs). The simulated set of experiments compare the performance of the grid system in terms of average number of duplications and resource consumption with respect to various graph sizes and CCRs for heterogeneous grid computing systems (see fig. 6). Each result is obtained with respect to CCR is an average of 25 graphs (over 5 sizes and 5 average parallelisms), and with respect to graph size is an average of 20 graphs (over 4 CCRs and five average parallelisms). In these experiments, the EDS-G algorithm outperforms the HLD, LDBS for different DAG sizes and CCRs with respect to avg. duplications and resource consumption.

## V. Conclusions

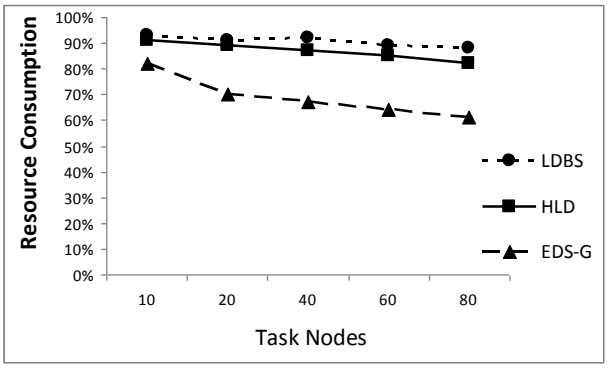
A duplication based strategy has been found very momentous for homogeneous and heterogeneous computing systems to improve the performance of the system. Currently more research work is focusing over heterogeneous computing system such as grids which consists of abundant resources over wide area networks with different capabilities. Scheduling a task graph with precedence constraints in grids is an issue due to the higher communication latencies. Duplication improves performance and reliability of such systems by duplicating critical tasks with higher communication costs.Our approaches optimize schedule by reducing duplications as much as possible without affecting the makespan and improve the system performance so that application execution cost and duplication overhead can be reduced. Performance comparison with best known duplications algorithms LDBS and HLD for grid computing system shows that EDS-G algorithm generates comparable schedules with remarkably less duplications and less resource consumption.



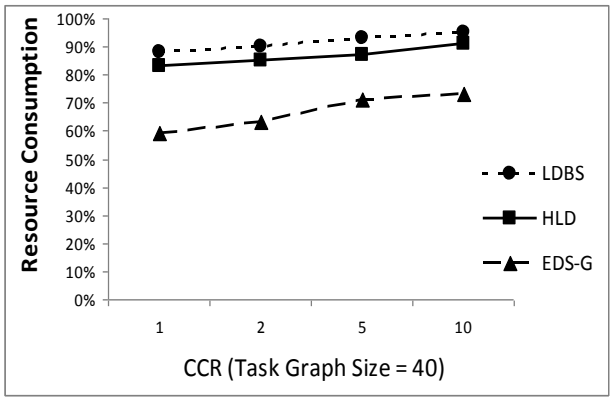

(a)




(b)




(c)




(d)

Fig 6. (a to d) Performance comparison of EDS-G algorithm on random DAGs in computational Grids.

## VI. References


1) Y.-K. Kwok and I. Ahmed, Benchmarking the Task Graph Scheduling Algorithms, IPPS/SPDP, pp. 531-537, 1998.
2) J. Liou and M. Palis, A Comparison of General Approaches to Multiprocessor Scheduling, Proc. of the 11th Int'l Parallel Processing Symp., pp. 152-156, 1997.
3) A. A. Khan, C. McCreary and M. S. Jones, A Comparison of Multiprocessor Scheduling Heuristics, ICPP, pp. 243-250, 1994.
4) Olivier B, Vincent B and Yves R, The Iso-level Scheduling Heuristic for Heterogeneous Processors, Proceedings of 10th Euromicro workshop on parallel, distributed and network-based processing, pp. 335-342, 2002.
5) H. Topcuoglu, S. Hariri and M.Y. Wu, Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, IEEE Trans. on Parallel and Distrib. Syst., 13(3): pp. 260 - 274, March 2002.
6) T. Hagras and J. Janecek, A High Performance, Low Complexity Algorithm for Compile-Time Job Scheduling in Homogeneous Computing Environments, Proc. Int'l Conf. on Parallel Processing Workshops, pp 149-155, Oct 2003.
7) A. Gerasoulis and T. Yang, A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs on Multiprocessors, Journal of Parallel and Distributed Computing, 16(4): pp. 276-291, 1992.
8) J. Liou and M. A. Palis, An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors, Proc. of Workshop on Resource Management, Symposium of Parallel and Distributed Processing, pp. 152-156, Oct 1996.
9) Kuan-Chou Lai and Chao-Tung Yang, A Dominant Predecessor Duplication Scheduling Algorithm for Heterogeneous Systems, Journal of Supercomputing, 44(2): pp. 126-145, 2008
10) Savina Bansal, Padam Kumar and Kuldip Singh, An Improved Duplication Strategy for Scheduling Precedence Constrained Graphs in Multiprocessor Systems, IEEE Trans. on Parallel and Distributed Systems, 14 (6): pp. 533-544, 2003.
11) I. Ahmed and Y.-K. Kwok, On Exploiting Task Duplication in Parallel Program Scheduling, IEEE Trans. on Parallel and Distributed Systems, 9(9): pp. 872-892, Sept 1998.
12) G.-L. Park, B. Shirazi and J. Marquis, DFRN: A New Approach for Duplication Based Scheduling for Distributed Memory Multiprocessor Systems, Proc. of the 11th Int'l Parallel Processing Symp., pp. 157-166, Apr 1997.
13) Y.-C. Chung and S. Ranka, Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed Memory

Multiprocessors, Proc. on Supercomputing, pp. 512-521, Nov 1992.
14) B. Kruatrachue and T.G. Lewis, Grain Size Determination for Parallel Processing, IEEE Software, 5(1): pp. 23-32, Jan 1988.
15) C.H. Papadimitriou and M. Yannakakis, Towards an Architecture-Independent Analysis of Parallel Algorithms, SIAM J. Computing, 19(2): pp. 322-328, Apr 1990.
16) Savina Bansal, Padam Kumar and Kuldip Singh, Dealing with Heterogeneity Through Limited Duplication for Scheduling Precedence Constrained Task Graphs, Journal of Parallel and Distributed Computing, 65(4): 479-491, Apr 2005.
17) A. Dogan and F. Ozguner, LDBS: A Duplication Based Scheduling Algorithm for Heterogeneous Computing Systems, Proceedings of the Int'l Conf. on Parallel Processing, pp. 352-359, Aug 2002.
18) C. Ernemann, V. Hamscher, and R. Yahyapour, Economic scheduling in grid computing, Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, Vol. 2537 of Lecture Notes in Computer Science, Springer Verlag, pp.128–152, 2002.
19) J. Yu, R. Buyya, and C. K. Tham, Cost-based scheduling of scientific workflow application on utility grids, Proceedings of the First IEEE International Conference on e-Science and Grid Computing (e-Science 2005), pp.140–147, IEEE Computer Society, 2005.
20) Mohammad I. Daoud and Nawwaf N. Kharma, Efficient Compile-Time Task Scheduling for Heterogeneous Distributed Computing Systems, Proceedings of the 12th IEEE International Conference on Parallel and Distributed Systems, pp. 11-22, 2006.
21) A. Agarwal and P. Kumar, Economical duplication based task scheduling for heterogeneous and homogeneous computing systems, IEEE International Advance Computing Conference (IACC 2009), pp.87–93, 2009.