Proactive Economical Task Scheduling Algorithm for Grid

Piyush Chauhan, Nitin, Rashmi Sharma and Ved Prakash Bhardwaj
Department of Computer Science & Engineering and Information Technology,
Jaypee University of Information Technology, P.O. Waknaghat,
Solan-173234, Himachal Pradesh, India

Abstract—Task Duplication technique can reduce the makespan of any application designed for the decentralized grid. In this paper, we have presented an algorithm called Proactive Economical Task Scheduling Algorithm; it can decrease computations in Economical Task Scheduling Algorithm for Grid (EDS-G). It works proactively for a grid. In this approach, each node proactively keeps a hierarchical ordered list of the chosen computation resources. The resources are selected randomly. These resources have specific optimization criteria. It is based on their computation capability and communication cost.

Keywords—scheduling algorithms; Decentralized Grid; Task Duplication heuristics; Directed Acyclic Graphs;

I. INTRODUCTION AND MOTIVATION

Latest generation of grids, consist of multiple heterogeneous and minute size clusters. Sometimes a grid has only single machine at one physical location. In case of huge grids composed of plentiful minute clusters, relying on Meta scheduler or single central scheduler is not practical. Scheduling jobs on such grids straddling across multiple organizations becomes vital. Many researchers projected peer-to-peer solution in their algorithms for the grid scheduling problem. Finding best possible schedule for multiprocessor scheduling [1], [2], [3] problem is NP complete problem [4]. Unlike existing peer-to-peer scheduling algorithms [5], [6], [7], [8], our approaches do the proactive future allocation of computational resources even before task is generated at any grid node. Before the generation of tasks resources are allocated. The allocated resources are arranged in decreasing order, based on optimization criteria.

An optimization criterion depends upon computation capability and bandwidth constraints related to the node. CYCLON protocol [9], [10] is used to establish superior version of shuffling. We add features of hierarchically arranging nodes, which are shuffled by CYCLON protocol. These nodes are arranged in non-increasing order of their computation capability and bandwidth. Weighted directed acyclic task graph are often used to represent a distributed and parallel applications [11], [12], [13]. In DAG node symbolizes application task. Edges of DAG stand for data dependencies between various application tasks of DAG. Here ordered list of allocated computational nodes tend to be useful in reducing assessment for best fit node for tasks of DAG. Duplication based scheduling [4], [14], [15], [16] is one of classification of task scheduling heuristic [17], [18], [20], [12] for DAG application. Predecessor task node for any task node is taken into consideration. Now makespan of an application is reduced by duplicating in ideal time slots between two already scheduled tasks the predecessor task.

In Grid computing environment duplication based scheduling give excellent results. Down side of duplication based scheduling is excessive duplication. This excessive duplication yield extra using up of nodes (computation resources) in grid. Duplication based scheduling is more useful for fine grain task graphs [4], [14]. In addition, duplication based scheduling is very useful for grids with higher CCRs (CCR means Ratio of average communication cost to average computation cost on selected Grid.). Efficiency of scheduling algorithms has a great tendency to decrease with rise in heterogeneity of grid computational resources. By duplicating only important tasks, we prevented some extent of degradation of scheduling algorithms for grid caused by heterogeneity. Task duplication decreases task finish time on computation resources of grid. EDS-G [9] algorithm investigates effect of duplicated tasks over makespan. This way EDS-G algorithm improves schedule by eliminating unproductive duplicated tasks whose removal does not affect the makespan. EDS-G algorithm compares all computational resources for each application task of DAG.

EDS-G finds out which duplicated parent task can be removed from computational nodes of grid such that makespan will not get affected. In proactive economical task scheduling algorithm, we arrange the nodes hierarchically of a grid obtained by running CYCLON protocol. Ordering of grid nodes is done before task is generated at grid node. Shuffling is done as it is explained in CYCLON Protocol. These nodes are arranged in non increasing order, based on optimization criteria. Ordered list is called Empty processor list (EPL). The advantage of hierarchical arrangement of grid computation resources is that we do not need to do task node’s comparison with all shortlisted nodes of grid. First task node of DAG will be assigned to first Processor from ordered list. Next task node will be assigned to either second processor from EPL list or to first processor. Out of these two processors, processor capable of finishing second task node fastest will be chosen. Instead of comparing with all shortlisted grid resources here, we compared only two processors for task assignment. Thus, less
time was consumed to assign task. Similarly, rest of task nodes will be assigned to best suited grid resources.

A lot of researchers have proposed task duplication based scheduling technique for computational grid system. But they compare entire shortlisted subset of grid nodes for each task node such that best fit grid node is obtained for each task node. In our research we have reduced the number of comparisons by ordering grid resources proactively. Rest of paper is structured as follows. Section II, explores the preliminaries and background. Section III, explains proposed Proactive economical task scheduling algorithm for grid in two parts. Part A mention modified version of CYCLON and part B explain Proactive Economical Task Scheduling Algorithm for Grid. Section IV, gives conclusion and future scope of work.

II. RELATED WORK

Many researchers have recognized need of efficient task scheduling algorithm [15] for decentralized grid’s computational resources. One example of using duplication technique to have efficient scheduling algorithm for heterogeneous computational system is explained in [11]. Authors of [11] introduced Heterogeneous Limited Duplication (HLD) scheduling algorithm for heterogeneous computing environment. HLD schedules tasks, based on their precedence constraints. HLD [11] avoid redundant replication by confining duplication to most essential immediate predecessor tasks. HLD is modification of Selective Duplication (SD) Algorithm [12] for heterogeneous computational systems. Selective duplication algorithm is quite effective for homogeneous computing systems. SD [12] algorithm completes in triple phases. Firstly, we arrange task sequence using critical path based priority. Secondary phase is used to select set of candidate processors for candidate task. Last phase of SD algorithm apply candidate task to processor which apply it at earliest, by means of insertion based duplication approach.

One of latest scheduling algorithm for decentralized grid computing system is EDS-G [4]. As explained in HLD algorithm EDS-G algorithm also generates priority-based task sequence. Task sequence is generated by arranging tasks in decreasing order of their communication and computation cost along longest directed path from the concerned task to the exit task in directed acyclic graph. Now, initial unscheduled task in the task sequence is chosen and scheduled on a grid computing resource that can end its computation at the earliest by means of duplication approach. This algorithm place the task in a former most idle period among two already scheduled tasks on the decentralized grid’s computational resource. Task $N_i$ on computational grid start working once data arrived from all parent nodes of task node $N_i$. Hence, grid resource may remain idle yielding scheduling holes. When start time of task $N_i$ on grid resource is restricted by data arrival from its most important immediate parent (Most important immediate parent of task $N_i$ is its ancestor whose data arrive last of all parents of task $N_i$) scheduling hole is generated. This scheduling hole may be exploited to duplicate tasks to minimize data arrival time. Start time and finish time of task $N_i$ and all tasks is calculated which help in obtaining makespan. Now two lists A & B are created having record of original task with links to dependent tasks and record of duplicated tasks in non-increasing order of earliest start time respectively. List B is modified if removal of duplicated task from list B does not affect makespan. In addition, list A removes those tasks, which have already been duplicated and hence are not providing any output to any immediate descendant task. Our algorithm Proactive Economical Duplication for Decentralized Grid uses partially similar algorithm but need less number of comparisons to assign tasks to best fit corresponding nodes. This becomes possible due to proactively arranging grid resources in hierarchy of performance. Out of list of $z$ resources, we can choose random $x$ resources ($x < z$).

This selection is done with help of CYCLON [5] protocol. CYCLON is gossip based protocol [6], [19]. In CYCLON, each peer knows small constantly changing set of other peers. All peers in grid occasionally contacts neighbor peer to shuffle caches and this neighbor peer is chosen whose information was the earliest one to have been injected in the grid.

III. PROPOSED PROACTIVE ECONOMICAL DUPLICATION SCHEDULING ALGORITHM FOR GRID

In this section, we will present our proposed algorithm in a structured manner. In first step, we will focus how the resources can be ordered in a proper manner. In next step, it will be illustrated that how we can improve the quality of schedule and get scheduling process better than EDS-G.

A. Proactive Ordering of Selected Neighbour Nodes in a Grid

Proactive ordering of resources is done by modified version of CYCLON [9],[10]. Overlay is formed and connected by means of epidemic algorithm [10], [7]. Each node knows a Petite set of neighbor nodes, which are continuously changing. This node occasionally contacts a neighboring node whose information was the first one to have been injected in the network and exchanges some of their neighbors. Each node maintains a neighbor list in a small, fixed size cache of $T$ entries. Neighboring node’s computation capability, communication cost, IP and port address are kept in Cache entry. Nodes start neighbor swap periodically, however not synchronized, at a fixed time period $\Delta t$. Hence, other than step of ordering neighbors, rest of steps are like CYCLON [9] protocol. The ordering and shuffling [5] of computational resources is done when initiating peer $I$. Carry out following steps: Select a random subset of $R$ neighbors (Length of list $R$ is $I(1 \leq l \leq T)$) from a set of neighbors of $I$ (initiating peer).

1. In LIST $R$ Increment by $I$ the age of all neighbors.
2. Select neighbor $H$ having maximum age among all neighbors in list $R$, and $l = 1$ other random neighbors.
3. Replace $H$’s entries with entry of age zero (i.e. new one) and with address of $I$.
4. Sent updated subset to $H$ node.
5. Receive from $H$ a subset of no more than $l$ of its own entries.
6. Discard entries pointing at $I$ and entries already contained in $I$’s cache.
7. Update I’s cache to include all remaining entries, by utilizing initially empty slots in cache (if any), and secondly substitute entries among the ones sent to H.

8. List EPL is formed by arranging neighbors in R in a non increasing order of their computation capability and communication cost.

Node H answers by sending back a random subset of at most of its neighbors, and renew its own cache to house all received entries. However, it does not increase any entry’s age until its own turn comes to run above stated protocol. This hierarchical ordering of neighboring nodes in grid helps us in reducing computations involved in Task Scheduling algorithms like EDS-G for grid.

B. Proactive EDS-G Algorithm

This section of the paper represents Proactive Economical Task Duplication based Scheduling Algorithm. This algorithm is split into two parts. Our work is modification of [4]. In proactive EDS-G algorithms, first part gives a method for scheduling based on technique of insertion based task duplication. Second part of algorithm removes tasks whose termination does not have an effect on makespan. The algorithm’s pseudo code is shown in figure 1. Once we have orderly placed computational resources before task is generated, we can use it when task is generated at any grid node. When task is generated at grid node, we split it into interdependent sub-tasks and use DAG to show their interdependence. Grid computing system can be characterized as G = (P, B) where P(p₁, p₂, p₃ ... pₓ, pₓ₊₁, ... pₓ) are grid’s computing resources. Grid’s resources are connected by various communication channels B. Because grid’s resources are of heterogeneous nature, hence same task’s computing cost on different processing nodes will be different. The nature of communication channel between nodes of grid is also heterogeneous type.

Here DAG’s tasks are of non-preemptive nature. Along with communication computation happen in parallel. (This is possible because each node of grid contain co-processor for communication.). If any two tasks are scheduled on same grid node, we consider communication cost between these two tasks to be negligible. On finishing of task in any grid node, that node sends data in parallel to all child tasks. Weighted DAG is used to represent application of grid. D = (D₀, e, T, C) here task node gets symbolized by Dᵢ. T is a computation cost matrix [14]. At ᵢ ᵢ ∈ T represent expected time to execute task of DAG Dᵢ on grid node ᵢ. e is set of communication edges. C is communication cost matrix. Expected time to communicate data from task Dᵢ to Dᵢ ᵢ representing by cᵢ ᵢ ∈ C. Task Dᵢ ᵢ’s

Mean computation cost represented by cᵢ ᵢ [11],[5] is calculated as follows:

\[
\overline{c}_{ii} = \frac{\sum_{j=1}^{n} c_{ij}}{z} \quad \text{for all } 1 \leq i \leq n \quad (1)
\]

Mean communication cost \( \overline{c}_{ij} \) [11], [14] between task Dᵢ and task Dⱼ is calculated as follows:

\[
\overline{c}_{ij} = \text{meandata transport rateover entire linksofgrid} \quad (2)
\]

∀1 ≤ i ≠ j ≤ n

Priority based task sequence is generated by ordering task in decreasing order of their computation and communication cost which is obtained via computing mean cost parameter(MCP) recursively as follows:

\[
\text{MCP}_i = \overline{t}_i + \text{Max}[\text{MCP}_j + \overline{c}_{ij}]\forall\text{Dn}_j \in \text{successor} (\text{Dn}_i) \quad (3)
\]

Successor(Dnᵢ) denote set of immediate child nodes in DAG of task Dnᵢ. Task sequence’s first unscheduled task is selected and scheduled on a grid’s computational resource which finish it first using task duplication approach. Now node of grid, which is first in EPL list, will be chosen and added to UP list. We will add processor in UP list from EPL list only if no fresh processor is available in UP list. It implies that addition of new processor/node for next task is done to UP list because all existing processors in UP list are having some task to execute. Next task from priority based task sequence will be chosen. Now we check out on which processor of UP list minimum makespan is possible using task duplication approach. Makespan is calculated as:

\[
\text{Makespan} = \text{maximum}[F_{ix}] \quad (4)
\]

\[
F_{ix}[14] \text{ is finish time of task } Dnᵢ \text{ on resource } P_x.
\]

\[
F_{ix} = S_{ix} + t_{ix} \quad (5)
\]

(Sᵢₓ is start time of task i on grid resource Pₓ). Formula to compute Sᵢₓ[4],[14] is as follows:

\[
S_{ix} = \text{maximum}[DT(Mᵢ, pₓ), \text{minimum}[pₓ, G_p^x]] \quad (6)
\]

In above formula pₓ stands for ready time of processor Pₓ. G_p^x is start time of first suitable and available time slot G_p that can accommodate task Dnᵢ on resource Pₓ if it exist then only we calculate Fᵢₓ. DT(Mᵢ, pₓ) [4] is data arrival time for most important immediate parent Mᵢ of task Dnᵢ on Pₓ. Formula to calculate DT(Mᵢ, pₓ) is given by:

\[
\text{DT}(Mᵢ, pₓ) = \text{maximum } [\text{minimum } [Fᵢₓ, F'ᵢₓ + cᵢ_j]] \quad (7)
\]

Proactive EDS-G Algorithm

Begin

1: Sort nodes in non increasing order of performance of processors. EMPTY PROCESSOR LIST (EPL);
2: Construct a priority based task sequence β;
3: Make an empty list of used Processors (UP);
4: do {
   

5: Select the first unscheduled task \( D_n_i \) in the task sequence \( \beta \).

6: \textbf{if} (no fresh Processor in \( \text{UP} \) list)

7: add 1st processor from \( \text{EPL} \) to \( \text{UP} \) list & remove it from \( \text{EPL} \) list;

8: \textbf{for} (all \( p_x \) processors in list \( \text{UP} \))

9: Sort the list of immediate parents of \( D_n_i \) in non-increasing order of data arrival time;

10: \textbf{for} all immediate parents, select the first immediate parent \( D_n_i \) from the list at step 9

11: \textbf{if} duplication of \( D_n_i \) can reduce finish time \( F_{ik} \) of \( D_n_i \) on \( P_x \). Duplicate \( D_n_i \);

12: \}

13: Compute earliest finish time \( F_{ik} \) of \( D_n_i \) on \( P_x \) using eq. (5);

14: \}

15: Find the minimum earliest finish time of \( D_n_i \);

16: Assign \( D_n_i \) on resource \( P_x \) with minimum \( F_{ik} \) in schedule \( S \);

17: \} \textbf{while} (there are unscheduled tasks in task sequence \( \beta \));

18: 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;

19: \textbf{for} (each duplicated task \( D_n_i \) in list \( B \))

20: \textbf{if} (no change in makespan of schedule \( S \) after removing duplicated task \( D_n_i \))

21: \{ Remove this duplicated task \( D_n_i \) from the schedule \( S \) and update list \( A \);

22: \}

23: \textbf{for} (each task \( D_n_i \) in list \( A \))

24: \textbf{if} (task \( D_n_i \) has no dependent task in schedule \( S \) due to its duplication later on different processors)

25: \{ remove this task \( D_n_i \) from the schedule \( S \);

26: \}

End

Once schedule is obtained from above shown algorithm, List A and B are maintained. Like EDS-G algorithm list A has record of original tasks and links of these tasks to their corresponding dependent tasks. Duplicated task are stored in decreasing order of earliest start time in list B. This Schedule will be modified if there is no change in makespan on removal of duplicated task from list B. From list A, tasks which do not provide results to immediate successor because of their duplication elsewhere are also erased. This updated schedule will not only contain optimum number of duplications like EDS-G algorithms but also it obtains this optimum schedule in less number of comparisons of best fit processors for task nodes using task duplication approach.

IV CONCLUSION AND FUTURE WORK

To improve the performance of distributed and grid systems duplication based strategy is widely used. This limited duplication based approach helps in improving performance of the grid computing system in economical way. Scheduling algorithms for the grid computing system have high communication cost. Previously, existing EDS-G algorithm improves makespan of the Task graph with precedence constraints. Our approach simply reduces computations involved in EDS-G. In Proactive economical task scheduling algorithm for computational grid system reduced computation and comparison is achieved to duplicate task on best fit grid resource. This deduction in the computation of best grid node for any given task node is due to proactively making EPL list. EPL list contain detail of the chosen grid nodes in non increasing order of their performance. Obviously, in our algorithm we need not compare task node with all chosen nodes of grid like in EDS-G. Instead, task node is simply assigned or duplicated to fresh node or any other node in \( \text{UP} \) list. Number of grid nodes in \( \text{UP} \) list is always less than random number of grid nodes chosen in EDS-G. Hence Proactive economical task scheduling algorithm for grid will not only give better results in terms of time but also in terms of computation and communication cost. This new approach can be further enhanced to include multiple bags of task applications.

REFERENCES


