A New Duplication Task Scheduling Algorithm in Heterogeneous Distributed Computing Systems

Aida A Nasr*, Nirmeen A EL-Bahnasawy, Ayman EL-Sayed
Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University
Menouf 32952, Egypt
*Corresponding author, e-mail: aida.nasr2009@gmail.com

Abstract

The efficient scheduling algorithm is critical to achieve high performance in parallel and distributed systems. The main objective of task scheduling is to assign the tasks onto the available processors with the aim of producing minimum schedule length and without violating the precedence constraints. So we developed new algorithm called Mean Communication Node with Duplication MCND algorithm to achieve high performance task scheduling. The MCND algorithm has two phases namely, task priority and processor selection. Our algorithm takes into account the average of parents' communication costs for each task to reduce the overhead communication. The algorithm uses new task duplication algorithm. We build a simulation to compare the MCND algorithm with CPOP with duplication algorithm. The algorithms are applied on real application. From results, the MCND algorithm shows the best results.

Keywords: Static task scheduling, List scheduling algorithm, Heterogeneous distributed systems

1. Introduction

The availability of high performance networks leads to a new platform, called as heterogeneous distributed platform. Such a platform contains interconnected resources with different computing capabilities and different computing speeds. To run an application in this heterogeneous platform, several issues need to be taken into account such as partitioning the application, scheduling the tasks, etc. The performance of a parallel applications on Heterogeneous Distributed Computing Systems (HeDCS) critically depends on the method used to allocate the tasks partitioned from the application onto the appropriate processors in the system [1, 2, 3, 4].

Boor task scheduling algorithm can undo any potential gains from the parallelism presented in the application, so selecting task scheduling algorithm is the important step of executing the parallel applications. In general, the objective of task scheduling is to minimize the execution time of a parallel application by properly assigning the tasks to the processors. Static and dynamic scheduling are the categories of scheduling models. In the static model, all information regarding the application and computing resources such as task weight, communication cost and data dependency is available a priori, so tasks scheduling is performed before the execution of the application. On the other hand; in the dynamic scheduling, scheduling is done at run-time. In this paper, we focus on static scheduling [5, 6, 7].

Static scheduling is classified into list-based, clustering and duplication based. List-scheduling basically consists of two phases: a task prioritizing phase and processor selection. In task prioritizing, the task priority is computed. In the processor selection phase, each task (in order of its priority) is assigned to processor that minimizes a suitable cost function. List-scheduling is generally accepted as an attractive approach since it characterized low complexity with good results [8, 9, 10, 11, 12, 13, 14, 15, 16]. Examples of list-based algorithms are Heterogeneous Earliest Finish Time (HEFT) and Critical Path On Processor (CPOP) [17]. Another static scheduling category is task duplication based algorithms, in which tasks are duplicated on more than one processor to reduce the waiting time of the dependent tasks. The main idea behind duplication based scheduling is to utilize processor idling time to duplicate predecessor tasks. This may avoid transfer of results from a predecessor, through a communication channel, and may eliminate waiting slots on other processors and reduce the communication overheads. An example for duplication algorithms is CPOP with duplication [18].
The existing list-scheduling algorithms do not take into account the average communication of parents, data ready time and the maximum path of task. So we proposed new algorithm called Mean Communication Node with Duplication MCND algorithm. The MCND algorithm is developed for static task scheduling for the HeDCS with limited number of processors. It avoids the drawbacks of list scheduling algorithms and duplication algorithms. The objective of new MCND algorithm is to generate the quality task scheduling with low complexity. The developed algorithm uses the maximum path of task in calculating priority and uses duplication task to reduce the overhead of communication.

The remainder of this paper is organized as follows. Section 2 discusses task assignment problem. Section 3 gives an overview of CPOP algorithm with duplication. Section 4 presents developed MCND algorithm. Section 5 discusses the results and in section 6, conclusions are given.

2. Task Assignment Problem

Task assignment model consists of single application and target computing system. The application is divided into tasks represented by Directed Acyclic Graph DAG, \( G=(V, E, P, W) \), as shown in Figure 1. Where \( V \) is the set of \( \nu \) tasks, and \( E \) is the set of \( e \) edges between the tasks. Each \( e(\nu, \nu') \in E \) represents the precedence constraint such that task \( \nu \) (i.e. parent) should be executed before task \( \nu' \) (i.e. child) can be started. The task with no parents is named root and the task with no children is named leaf. \( P \) is the set of \( p \) processors available in the heterogeneous system. \( W \) is a \( \nu \times p \) computation cost matrix, where \( \nu \) is the number of tasks and \( p \) is the number of processors in the system. When two tasks are scheduled on the same processor the communication cost between these tasks can be negligible, because the speed of the inter-processor communication network is extremely low. All processors in the HeDCS are assumed to be fully connected. Communications between processors occur via independent units, so computation of tasks and communications between processors can be executed in parallel. Figure 1 shows an application with five tasks. The application is represented as a DAG and the execution costs estimated for the five tasks on the HeDCS are shown as a computation cost matrix [19, 20, 21].

![Figure 1. Example of the DAG and Computation Matrix](image)

**Definition (1)** Critical Path (CP): CP of a DAG is the longest path from the entry task to the exit task in the graph.

**Definition (2)** \( EST(t_i, P_j) \) [10]: Denotes the Earliest Start Time of a task \( t_i \) on a processor \( P_j \) and is defined as shown in Equation (1).

\[
EST(t_i, P_j) = \max \{ TAvailable(P_j) , \max \{ AFT(t_k) + c_{k,i} \} \}
\]  

(1)

Where TAvailable (\( P_j \)) is the earliest time at which processor \( P_j \) is ready. AFT (\( t_k \)) is the Actual Finish Time of a task \( t_k \) (where \( t_k \) is the parent of task \( t_i \) and \( k=1, 2, ..., n \)) on the
processor $P_j$. $c_{k,i}$ is the communication cost from task $t_k$ to task $t_i$. $c_{k,i}$ equal zero if the predecessor task $t_k$ is assigned to processor $P_j$. For the entry task, EST($t_{entry}$, $P_j$) = 0.

**Definition (3)** EFT($t_i$, $P_j$) [10]: Denotes the Earliest Finish Time of a task $t_i$ on a processor $P_j$ and is defined as shown in the Equation (2).

$$EFT(t_i, P_j) = EST(t_i, P_j) + w_{i,j}$$

Which is the Earliest Start Time of a task $t_i$ on a processor $P_j$ plus the computational cost $w_{i,j}$ of $t_i$ on a processor $P_j$.

**Definition (4)** Data Ready Time (DRT): is the idle time waited by a $t_i$ on processor $P_j$.

**Definition (5)** Maximum Parent (MP): maximum parent of task $t_i$ is a parent task $t_k$ such that the value of EFT($t_k$, $P_j$) + $c_{k,i}$ is the largest among all $t_i$'s parent tasks.

**Definition (6)** Very Important Task (VIT): is the task that belongs to the critical path of DAG.

3. Critical Path on Processor with Duplication Algorithm

In this section, we give an overview of Critical Path On Processor (CPOP) algorithm with duplication as related work. CPOP with duplication consists of two phases: prioritizing phase and processor selection phase. In task prioritizing phase, the algorithm selects the task with the highest (upward rank + downward rank) value at each step. Upward rank is computed by the equation (3).

$$Rank(u) = \bar{w}_i + max_{n_j \in succ(n_i)}(c_{i,j} + rank_u(n_j))$$

Where $succ(n_i)$ is the set of immediate successors of task $n_i$, $c_{i,j}$ is the average communication cost of edge($i,j$), and $\bar{w}_i$ is the average computation cost of task $n_i$. The downward rank is computed by using the equation (4).

$$Rank(d(n)) = max_{n_j \in pred(n_i)}(c_{i,j} + \bar{w}_j + rank_d(n_j))$$

Where $pred(n_i)$ is the set of immediate predecessors of task $n_i$. The algorithm targets scheduling of all critical tasks (i.e., tasks on the critical path of the DAG) onto a single processor, which minimizes the total execution time of the critical tasks. If the selected task is noncritical, the algorithm applies task duplication condition to select the processor.

4. The Developed Algorithm

The Mean Communication Node with Duplication MCND algorithm is list-based scheduling algorithm. It uses the main idea of HCPT algorithm with some edits and adding new duplication algorithm. It consists of two phases only, task priority and processor selection. The MCND algorithm removes level sorting phase from HCPT algorithm to reduce executing time of algorithm. The detailed explanation of each phase of the algorithm is described in the following subsections.

4.1. Task Priority Phase

In this phase, the MCND algorithm assign a priority for each task using rank attribute that is obtained from the equation (5).

$$Rank(t_i) = MCP(t_i) + max_{t_j \in succ(t_i)}(c_{i,j} + Rank(t_j))$$

Where $MCP(t_j)$ refers to Mean Communication of Parents. It is computed by the equation (6).

$$MCP(t_i) = (\sum_{k=1}^{n} c_{k,i})/x$$
Where \( x \) is the number of Parents, \( C_{k,i} \) is the communication between parent \( t_k \) and task \( t_i \). For the exit task \( t_{exit} \) the Rank value is equal \( MCP(t_{exit}) \). Tasks List TL is generated by sorting the tasks by decreasing order of Rank value. Figure 2 shows MCND algorithm’s pseudocode.

### 4.2. Processor Selection Phase

This phase consists of two stages: processor stage and duplication test stage. In processor stage, MCND algorithm selects task \( t_i \) from TL. If \( t_i \) has no parents or all parents are scheduled, the algorithm calculates \( EFT \) of task \( t_i \) by Equation 2 for each processor, and selects the processor that has a minimum \( EFT \) to assign the task. With high performance algorithms, some processors are idle during the execution of the application because of DRT. If DRT is enough to duplicate MP, the execution time of the parallel application could be reduced. So, the algorithm applies task duplication to reduce the makespan. The algorithm tests, if DRT of task \( t_i \) is more than the weight of MP on the same processor \( p_j \), the algorithm duplicates the MP on \( p_j \) and updates \( EFT \) of task \( t_i \). The duplication stage is applied on VIT only. This must be done without violating the precedence constrains among tasks.

1. **Set the computation cost of tasks and the communication cost of edges.**
2. **Compute Rank for all tasks starting from the exit task by the next equation.**
   \[
   \text{Rank}(t_i) = \text{MCP}(t_i) + \max_{j \in \text{succ}(t_i)} (C_{ij} + \text{Rank}(t_j))
   \]
3. **Sort the tasks in Tasks List TL in decreasing order of Rank values.**
4. **For each task \( t_i \) in TL**
   1. **For each processor \( p_j \) in the processor set \( (p_j \in Q) \) do**
      1. Compute \( EFT(t_i, p_j) \) value
      2. **End for**
6. **Assign task \( t_i \) to the processor \( p_j \) that minimizes \( EFT \)**
   
   If \( t_i \) is VIT
   1. If DRT \((t_i, p_j)>w(MP, p_j)\)
   2. If EST\((t_i, p_j)>EFT(MP, p_j)\)
      1. Duplicate MP on \( p_j \) without violating the dependency constraints
      2. Update \( EFT \) of \( t_i \) on \( p_j \)
      3. **End if**
   3. **End if**

Figure 2. The MCND Algorithm Steps

### 5. Results and Discussion

#### 5.1. Comparison Metrics

The comparison metrics are schedule length ratio, the average of speedup and the average running time.

#### 5.1.1 Schedule Length Ratio (SLR)

SLR value is defined by the equation (7)

\[
\text{SLR} = \frac{SL}{\sum_{t_i \in CP_{min}} \min_{p_j \in Q} \{wi, fj\}}.
\]

Where SL is the schedule length. The divisor is the summation of the minimum computation costs of tasks on CPmin. (For an unscheduled DAG, if the computation cost of each task \( t_i \) is set with the minimum value, then the critical path will be based on minimum value, then the critical path will be based on minimum computation cost, which is represented as CPmin) \[17\]. The SLR can never be less than one, since the divisor is the lower bound. Algorithm that gives smallest SLR of a graph, is the best algorithm with respect to performance.
5.1.2 The Average of Speedup

Speedup of a schedule is defined as the ratio of the schedule length obtained by assigning all tasks to the fastest processor, to the schedule length of parallel application.

\[
\text{Speedup} = \frac{\max_{pi \in P} \sum_{i \in V} w(i,j)}{\text{SL}}
\]

(8)

Where \(w(i,j)\) the weight of task \(ti\) on processor \(pj\) and \(SL\) is is the schedule length.

Speedup is a good measure for the execution of an application on a distributed system. Due to minimize schedule length, all processors have finished tasks execution earlier and speedup of MCP algorithm increases. The results of the comparative study according to the speedup parameter have been presented in Figure 6. According to the results, performance ratio of speedup is calculated as 15%.

5.1.3 Average Running Time (ART)

ART is the average running time of different DAGs. The algorithm with smallest average running time is the best algorithm.

5.2. Simulation Environment

We use tow types of graphs for testing MCND algorithm and the related work: randomly generated graphs and graphs represented the real problems.

5.2.1 Random Graph Generator

For building random DAGs the program requires the following input parameters.

- \(N\) is the number of DAG tasks, where \(N \in \{20, 40, 60, 80, 100, 120\}\).
- \(\alpha\) (parallelism) is the shape parameter of DAG. Like [18] we assume that height of a DAG is \(\sqrt{N/\alpha}\). And the width of each level is random selected from a uniform distribution with mean value to \(\sqrt{N}\). Where \(\alpha \in \{0.5, 1, 2\}\).
- \(\text{Out Deg}\) is the out Degree. It is the maximum number of task successors.
- \(D\) is the DAG density. The density of DAG determines the number of dependencies between nodes. \(D \in [0.3, 0.8]\). There is an edge between \(ti\) in level \(L\) and \(tj\) in level \(L+Z\), if random value \(\in [0.1, 1]\) \(\leq D\) and number of successors \(\leq \text{out Deg of } t\).
- \(\text{WDAG}\) is the average computation cost of given graph. This selected randomly from a predefined set \([\text{WDAG}/4, 2*\text{WDAG}]\), \(\text{WDAG} \in \{50, 70, 100, 150, 200\}\).
- \(\text{CCR}\) is Communication to Computation Ratio. It is the average edge weight divided by the average node weight. \(\text{CCR} \in \{0.1, 0.5, 1, 5, 10\}\).
- \(\text{Percentage Ratio}\) is the heterogeneity factor for processors speeds. When this ratio is high, this mean that difference between task’s computations on different processors is very high and vice versa when the percentage ratio is low, the difference is low.

We take the average results for each DAG size at \(p \in \{2, 4, 8, 16, 32, 64\}\). Figures 3, 4, 5 show the results of random DAGs. Figure 3 shows the average SLR with respect to various number of tasks. It is noted that, SLR of MCND algorithm is smaller than CPOP with duplication. The MCND algorithm uses the most important attributes of the task (Mean Communication of Parents to expect DRT(ti) and the maximum path from that task to exit task) to calculate the priority for each task. According this attributes the algorithm sorts the tasks. The algorithm uses also task duplication to reduce DRT of the task successors, so it can reduce the overall time of application. For this reasons, the MCND algorithm is more efficient than CPOP with duplication algorithm. Speedup is a good measure for the execution of an application on a distributed system. Due to minimize schedule length, all processors have finished tasks execution earlier and speedup of MCND algorithm increases. The results of the comparative study according to the average speedup parameter have been presented in Figure 4. We observe that MCND algorithm is faster than the CPOP algorithm, because it applies task duplication on VIT only to reduce the communication overhead. The new algorithm takes small amount of time for execution; this is shown in Figure 5. Because the CPOP with duplication algorithm apply task
duplication for each task, it takes more time for execution. We take some of snapshots from our simulation to show the effect of MCND algorithm.

Figures 6, 7, and 8 show the Snapshots from the simulation program. We defined the parameters of DAG generation on section 5.2.1. random graph generator. From this snapshots, we observe that the MCND algorithm is more efficient with fine-grain graphs at (CCR≤1). The Percentage Ration parameter also has an effect on the result. With percentage ration less than or 0.75 the performance of MCND algorithm is very high. With Other parameters the MCND algorithm has high performance compared with the CPOP with duplication algorithm.
Figure 5. The Average Execution Time with Respect of DAG size

Figure 6. Snapshot (1) from the simulation programs

Figure 7. Snapshot (2) from the simulation programs
5.2.2 The Real Applications

The MCND algorithm applies also on application DAGs of real problems like Gaussian elimination and Fast Fourier Transformation. Figures 9, 10 show Gaussian elimination and FFT graphs. The schedule length after applying the MCND algorithm and CPOP with duplication algorithm on Gaussian DAG is 636 and 690 respectively at 4 processors. The two algorithms apply also on FFT DAG and the schedule length of MCND algorithm is 452 and the schedule length after applying CPOP with duplication algorithm is 542 at 4 processors.
6. Conclusion

In this paper, a new task duplication scheduling algorithm has been presented for heterogeneous distributed computing systems (HDCS) to enhance scheduling performance. This algorithm uses a new attribute called Rank to assign a priority for each task. It also uses task duplication technique to decrease the communication overhead. The performance analysis showed that the proposed MCND algorithm has better performance than CPOP with duplication algorithm. According to the simulation results, it is found that the MCND algorithm is better than the other algorithm in terms of SLR, speedup, and execution time. The new algorithm applies new task duplication algorithm to reduce the schedule length of DAG. It applies the task duplication on VIT only not on every task in DAG. So, it takes low execution time to schedule the tasks.

References


