An Efficient Algorithm for Task Allocation in Distributed Networking Environment

Kapil Govil$^{1}$, Ajay Rastogi$^{2}$ and Amit Kumar$^{3}$

$^{1}$Teerthanker Mahaveer University, Moradabad
$^{2}$Bharati Vidyapeeth’s Institute of Computer Applications and Management, New Delhi
$^{3}$Kapil Govil@gmail.com, ajayrastogimbd@gmail.com and amit.vishnoi08@gmail.com

ABSTRACT

Distributed Networking Environment (DNE) refers to the use of distributed systems to solve computational problems. A DNE consists of one or more applications spread over several computers. These computers may be geographically separated from one another. The applications may be executed on different platforms using different operating systems and telecommunication protocols. In short, a DNE offers a variety of information system solutions regardless of the location of the user, his operating system or the equipment using by him. The allocation of tasks in DNE has an important role to the evaluation of the performance of the network. Performance enhancement of the DNE is a major and challenging problem for the researchers. The problem of processing of "m" tasks to "n" processors (m > n) in a distributed networks is addressed here through a new modified tasks allocation policy for the task processing in a DNE. The model, presented in this paper allocates the tasks with best suited allocation to the processor to increase the performance of the DNE. In the present paper, the problem has chosen in which the number of processors are less than the number of tasks of the distributed network. It is based on the execution cost for the tasks to the various processors. To allocate tasks, Greedy has defined an algorithm, which is known as Greedy Activity Scheduling Algorithm (GASA). In the present paper, we have used GASA and First Come First Serve (FCFS) policy to obtain more efficient result by allocating tasks to processors.

KEYWORDS

Distributed Computing Environment; First Come First Serve; Greedy Activity Scheduling Algorithm.

1. INTRODUCTION

A distributed computing environment is a collection of processors that do not share memory. It is a collection of loosely coupled processors interconnected by a communication network. The distributed computing environment is the environment, in which services provided by the network reside at multiple sites. Instead of single large machine being responsible for all aspects of process, each separate processor handles subset. In the distributed environments the program or tasks are also often developed with the subsets of independent units under various environments. It has drawn tremendous attention in developing cost-effective and reliable applications to meet the desired requirement. The main research problem for such networks is the allocation problem, in which either execution time or cost is to be minimized or reliability is to be maximized of the networks.

2. OBJECTIVE

The objective of the present research paper is to enhance the performance of the distributed computing environment by using the proper utilization of its processors. The present research paper minimizes the overall execution cost for a distributed computing environment through optimally assigning the tasks of various processor of the network. A set of tasks have to process where the number of tasks are more than the number of processors in distributed computing environment.

3. TECHNIQUE

In order to evaluate the overall optimal execution cost of a distributed computing environment, we have chosen the problem where a set $P = \{p_1, p_2, p_3, \ldots, p_m\}$ of ‘m’ processors and a set $T = \{t_1, t_2, t_3, \ldots, t_n\}$ of ‘n’ tasks, where $m > n$. The starting cost ($S_i$), finish cost ($F_i$) and Execution cost ($E_i$) of each task is known and is mentioned in the Task Cost Matrix of order $m \times 3$ namely $TCM(i,:)$. Arrange the $TCM(i,:)$ in ascending order of their finish cost and store them in Arranged Task Cost Matrix (ATCM) of order $m \times 3$. Make the graphical representation of ATCM using Greedy Activity Scheduling Algorithm (GASA) [8]. Now, store the set of tasks into $T_{sel}(1,)$ for those tasks which are not interfering with another task. To evaluate the maximum number of task allocation at each processor at a cost, put the number of selected tasks in $T_{sel}(1,)$ and $n$ into the function of Maximum Number of Task Allocation (MNTA). The function of Maximum Number of Task Allocation (MNTA) is

$$MNTA = \frac{\text{Number of selected tasks in } T_{sel}(1,)}{\text{Number of Processors}}$$

Now, Allocate tasks as MNTA to each processor by using First Come First Serve (FCFS) policy [1]. Repeat the process until all tasks not been executed. After adding all execution timings of all allocated tasks, state the total execution cost.

4. ALGORITHM

Start Algo

Step 1:
Read the number of Processor in $n$

Step 2:
Read the number of task in $m$

Step 3:
For $i = 1$ to $m$

Step 4:
Read the starting cost of task $i$ in $S_i$

Step 5:
Read the finish cost of task $i$ in $F_i$

Step 6:
Execution cost ($E_i$) for task $i = F_i - S_i$

Step 7:
End Algo
Step 8: Store Si, Fi and Ei in Task Cost Matrix TCM(,)

Step 9: End for

Step 10: Arrange the TCM(,) in ascending order of their finish cost and store it in Arranged Task Cost Matrix ATCM(,)

Step 11: While All task ! = SELECTED

Step 12: Make graphical representation of ATCM(,) using Greedy Activity Scheduling (GASA) [8]

Step 13: Select those tasks which are not interfering with another task.

Step 14: Make a set of selected tasks in Tsel(1,)

Step 15: Compute Maximum Number of Task allocation (MNTA) as –

\[
\text{MNTA= } \frac{\text{Number of selected tasks in } Tsel(1,)}{\text{Number of Processors}} \quad \text{(in round figures)}
\]

Step 16: Allocate tasks as MNTA to each processor by using First Come First Serve (FCFS) policy [1]

Step 17: End While

Step 18: Add all execution costing of all allocated tasks

Step 19: State total Execution cost

Step 20: End Algo

5. IMPLEMENTATION

In the present paper, the distributed network consist a set P of 3 processors \{p_1, p_2, p_3\} and a set T of 10 tasks \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}. The starting cost (S_i), finish cost (F_i) and Execution cost (E_i) of each task is also known and is mentioned in the Task Cost Matrix of order 10 x 3 namely TCM(,).

Now, arrange the TCM(,) in ascending order of their finish cost and store them in Arranged Task Cost Matrix (ATCM) of order 10 x 3 namely ATCM(,).

\[
\begin{array}{c|ccc}
\text{S_i} & \text{F_i} & \text{E_i} \\
\hline
\text{t_1} & 1 & 5 & 4 \\
\text{t_2} & 1 & 2 & 1 \\
\text{t_3} & 2 & 3 & 1 \\
\text{t_4} & 4 & 6 & 2 \\
\text{t_5} & 5 & 7 & 2 \\
\text{t_6} & 6 & 9 & 3 \\
\text{t_7} & 7 & 11 & 4 \\
\text{t_8} & 8 & 10 & 2 \\
\text{t_9} & 10 & 12 & 2 \\
\text{t_{10}} & 11 & 13 & 2 \\
\end{array}
\]

The graphical representation of the ATCM (,) using Greedy Activity Scheduling Algorithm (GASA) [8] is shown in figure 1.

Now, select those tasks which are not interfering with another task. We obtain selected (shaded) task as shown in figure 2.

Here, we have to make a set of selected tasks as -

Selected tasks Tsel(1,) = \{t_2, t_3, t_4, t_6, t_9\}

Copy Right © INDIACom-2011 ISSN 0973-7529 ISBN 978-93-80544-00-7
Here, we have 5 selected tasks and 3 processors; by putting these values to the function of Maximum Number of Task Allocation (MNTA)

\[ \text{MNTA} = \frac{5}{3} \approx 2 \]

So, we can allocate maximum 2 tasks at a cost at each processor. By using First Come First Serve (FCFS) [1] policy, we allocate first 2 tasks (i.e. \( t_5 \) & \( t_9 \)) to processor \( p_1 \); next 2 tasks (i.e. \( t_1 \) & \( t_3 \)) to processor \( p_2 \) and last 1 task (i.e. \( t_6 \)) to processor \( p_3 \). Again, by making graphical representation for non – selected (i.e. \( t_1 \), \( t_6 \), \( t_8 \), \( t_7 \), \( t_{10} \)) is shown in figure 3.

Now, we club all the execution costing, we get the total execution cost as in table 1.

\[ \text{MNTA} = \frac{2}{3} \approx 1 \]

So, we can allocate maximum 1 task at a cost at each processor. By using First Come First Serve (FCFS) [1] policy, we allocate first 1 task (i.e. \( t_7 \)) to processor \( p_1 \); last 1 task (i.e. \( t_{10} \)) to processor \( p_2 \).

6. CONCLUSION

In this research paper we have chosen the problem, in which the number of the tasks is more than the number of processors of the distributed system. The model mentioned in this paper is based on the consideration of processing cost of the tasks to various processors. The method is presented in pseudo code and implemented on the several sets of input data to test the performance and effectiveness of the pseudo code. It is the common requirement for any assignment problem that the tasks have to be processed with minimum cost. Here, performance is measured in terms of execution cost of the task that has been processed by the processors of the network and also these tasks have been processed optimally. Now, we club all the execution costing of allocated tasks at all three processors. By adding all execution costing, we get the total execution cost as in table 1.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Allocated tasks</th>
<th>Execution Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_1 )</td>
<td>( t_2 ) * ( t_1 ) * ( t_4 ) * ( t_7 )</td>
<td>10</td>
</tr>
<tr>
<td>( p_2 )</td>
<td>( t_4 ) * ( t_5 ) * ( t_6 ) * ( t_{10} )</td>
<td>9</td>
</tr>
<tr>
<td>( p_3 )</td>
<td>( t_7 ) * ( t_8 )</td>
<td>4</td>
</tr>
<tr>
<td>Total Execution Cost</td>
<td></td>
<td>23</td>
</tr>
</tbody>
</table>

Table 1: Execution Cost Table

REFERENCES


