Maximal Link Mode Algorithm For Task Allocation In Distributed Computing Systems

Avanish Kumar
Department of Mathematical Sciences & Computer Applications, Bundelkhand University, Jhansi, UP, E-Mail: dravanishkumar@yahoo.com

Abhilasha Sharma
Shree Ganpati Institute of Technology, Ghaziabad, UP, E-Mail: abhilasha.kapil@rediffmail.com

Vanita Ben Dhagat
Department of Mathematics, JNCT, Bhopal, MP, E-Mail: vanita1_dhagat@yahoo.co.in

ABSTRACT
Distributed computing systems are of current interest due to the advancement of microprocessors technology and computers networks. It consists of multiple computing nodes that communicate with each other by message passing mechanism. The advancement the new technologies in communication and information lead to the development of the Distributed System. The task allocation is an essential phase in the Distributed computing systems. We consider the problem of m tasks and n processors (where m\(\gg\)n). In this paper we explain the optimality to allocate the tasks with respect to the minimum execution cost and inter processors communication (IPC) cost and maximize the overall Throughput of the system. We have applied the procedure of the Maximal Linked Mode (MLM) and developed an Algorithm named MLMA to allocate the tasks to processors in such a way that the load on all the processor will be balanced. The Execution Cost Matrix (ECM), and Data Transfer Rate Matrix (DTRM) have been considered while preparing the MLM algorithm. This algorithm gives the better results in comparison to the others and applicable random program structure.

KEY WORDS
Distributed Computing Systems, Task Allocation, Execution Cost, Data Transfer Rate, Maximal linked mode

1.0 INTRODUCTION
The term "Distributed Computing System [DCS]" is used to describe whenever there are several computers interconnected in some fashion so that a program or procedure running on system with multiple processors. However, the term has different meanings with regard to different systems, because processors can be interconnected in many ways for various reasons. In the most general form, the word distribution implies that the processors are in geographically separate locations. Occasionally, the term is also applied to an operation using multiple mini-computers, which are not hardware, connected with each other and are connected through a satellite. A user-oriented definition [1, 2] of distributed computing is "Multiple Computers, utilized cooperatively to solve problems".

Distributed computing system has attracted several researchers by posing several challenging problems. In a DCS, the execution of a program may be distributed among several computing elements to reduce the overall system cost by taking advantage of heterogeneous computational capabilities and other resources within the system. The task allocation in a Distributed Processing System finds extensive applications in the faculties, where large amount of data is to be processed in relatively short period of time, or where real-time computations are required. Meteorology, Cryptography, Image Analysis, Signal Processing, Solar and Radar Surveillance, Simulation of VLSI circuits and Industrial process monitoring are areas of such applications. These applications require not only very fast computation speeds but also different strategies involving distributed task allocation systems. In such applications the quality of the output is proportional to the amount of real-time computations.

The main incentives for choosing DCS are higher throughput, improved availability, and better access to a widely communicated web of information. The increased commercialization of communication system means that ensuring system reliability is of critical importance. Inherently, distributed system is more complex; therefore, it is very difficult to predict the performance of DCS. Mathematical modeling is the tool which can plays an important role to predict the performance of DCS. Therefore, there is an urgent need to develop a method for it. Allocation of tasks in a DCS may be done in verity of ways (i) Static Allocation: In static allocation when a task is assigned to processor, it remains there while the characteristic of the computation change and a new assignment must be computed. The phrase “characteristics of the computation” means the ratios of the times that a program spends in different parts of the program. Thus in a static allocation, one is interested in
finding the assignment pattern that holds for the life time of a program, and result in the optimum value of the measure of effectiveness. These problems may be categorized in static [3 - 13], (ii) Dynamic Allocation: In order to make the best use of resources in a distributed system, it is essential to reassign modules or tasks dynamically during program execution, so as to the advantage of changes in the local reference patterns of the program [14-18]. Although the dynamic allocation has potential performance advantages, Static allocation is easier to realize and less complex to operate.

Several other methods have been reported in the literature, such as, Integer programming [19,21], Branch and bound technique [22-23], Matrix reduction technique [7,11,13], reliability evaluation to deal with various design and allocation issues in a DCS by [24-34].

The main objective of this paper is to minimize the total program execution cost. The model utilized the mathematical programming technique for execution of the tasks considering when a task is assigned to processor, it remains there while the characteristic of the computation change, and a new assignment must be computed. The developed algorithm is programmed in Visual C++. Several sets of input data are used to test the effectiveness and efficiency of the algorithm. It is found that the algorithm is suitable for arbitrary number of processors with the random program structure.

2.0 TASK ASSIGNMENT PROBLEM

The specific problem being addressed here, that is as follows: Given application software that consists of “m” communicating tasks, \( T = \{t_1, t_2, \ldots, t_m\} \), and a heterogeneous distributed computing system with “n” processors, \( P = \{P_1, P_2, \ldots, P_n\} \), where it is assumed that \( m \gg n \), assign (allocate) each of the “m” tasks to one of the “n” processors in such a manner that the IPC cost is minimized and the processing load is balanced. The processing cost of these tasks on all the processors is given in the form of Execution Cost Matrix (ECM) of order \( m \times n \). The Execution Cost of a given assignment on each processor are calculated by the following equation:

\[
PEC(j) = \sum_{i=1}^{n} ec_{ij}x_{ij}, i = 1,2,\ldots,m
\]

Where \( x_{ij} \) is the

\[
\begin{cases} 
1, & \text{if task } t_i \text{ is assigned to processor } p_j \\
0, & \text{otherwise}
\end{cases}
\]

Data Transfer Rate [DTR]: Data Transfer Rate \( d_{ik} \) is the per unit cost i.e. data exchanged between tasks \( t_i \) and \( t_k \) during the program execution.

\[
DTR(P_j) = \min(ec_{ij}) \times d_{ij}
\]

\[
TDT(P_j) = \sum DTR(P_j)
\]

where \( i = 1,2,\ldots,m \) and \( j = 1,2,\ldots,n \)

Inter Processors Communication Cost: The Inter Processor Communication cost \( cc_{ik} \) of the interacting tasks \( t_i \) and \( t_k \) is the minimum cost required for the exchange of data units between the processors during the process of execution.

\[
IPC(j) = \sum \min(ec_{ij}) \text{ where } i = 1,2,\ldots,m \text{ and } j = 1,2,\ldots,n
\]

Response Time (RT) of the System: Response time of the system is a function of amount computation to be performed by each processor and the computation time. This function is defined by considering the processor with the heaviest aggregate computation and communication load. Response time of the system for a given assignment is defined by

\[
RT(\text{Alloc}) = \max_{\text{processors}} \{PEC(\text{Alloc})_j + IPC(\text{Alloc})_j\}
\]

Average load: For each node there are \( N \) values representing the execution cost required for the corresponding module to be processed on each of the \( N \) processors. These values are in the matrix \( ECM \). Each edge is labeled with a value that represents the communication cost needed to exchange the data when the modules reside on different processors. The total workload \( W \) is the summation of the maximum module execution cost on the different processors as shown,

\[
W_j = \sum_{1 \leq i \leq m} ec_{ij}, j = 1,2,\ldots,n
\]

\[
W_j = \sum_{1 \leq i \leq m} ec_{ij}, j = 1,2,\ldots,n
\]

3.0 DEFINITIONS

Execution Cost: The execution cost \( ec_{ij} \) Where \( 1 \leq i \leq m, 1 \leq j \leq n \) of each task \( t_i \) depends on the processor \( p_j \) to which it is assigned and the work to be performed by each of tasks of that processor \( p_j \). The processing execution cost of the tasks on all the processors is given in the form of Execution Cost Matrix (ECM) of order \( m \times n \). The Execution Cost of a given assignment on each processors are calculated by the following equation:

\[
PEC(j) = \sum_{i=1}^{n} ec_{ij}x_{ij}, i = 1,2,\ldots,m
\]
The average load on a processor $L_{avg}(P_j)$ depends upon the different tasks on each processor. The system is considered to be balanced if the load on each processor is equal to the processor average load within a given (small percentage) tolerance.

Maximally-linked modules (MLM) algorithm
In an existing heuristic methods [10], a search is made for a pair of adjacent modules with the maximum communication cost between them. These modules are then assigned to the same processor to minimize the inter processor communication cost. In our heuristic method, we always try to choose a module, $t_k$, which is considered to be maximally linked from the perspective of the processor on which the assignment is to be made. A module is to be maximally linked if it has the largest aggregate of inter module data transfer rate of any of the modules that are adjacent to one or more of the modules. In the previous work [1] the MLM were determined once at the beginning of the allocation process. In the algorithm described in this manuscript, the MLM is recomputed for each step of the allocation process in the manner given below. For a given point in the allocation process the MLM for processor $P_j$ can be given by

$MLM_j = t_k$ where $k \in DTRM$

and $d_{ki} = \max \{ d_{ij} \}$ where $d_{ij}$ Data transfer rate between the Modules $1 \leq i \leq m$, $k \in DTRM$

The major rationale here is simple, since a maximally linked module has the maximum amount of communication with its neighboring modules, in most cases it is advantageous to form clusters around such modules.

4.0 ASSUMPTIONS
To keep the algorithm reasonable in size, several assumptions have been made while designing the algorithm. A program is assumed to be collection of “m” tasks to be executed on a set of “n” processors, which have different processing capabilities. A task may be portion of an executable code or a data file. The number of tasks to be allocated is more than the number of processors (m >> n), as normally is the case in the real life. It is assumed that the execution cost of a task on each processor is known, if a task is not executable on any of the processor due to absence of some resources. The execution cost of that task on that processor is taken to be ($\infty$) infinite. We assume that once a task has completed its execution on a processor, the processor stores the output data of the task in its local memory. If the data is needed by some another task being computed on the same processor, it reads the data from the local memory. Using this fact, the algorithm tries to assign heavily communicating tasks to the same processor. Whenever groups of tasks or cluster are assigned to the same processor, the data transfer between them is zero. Completion of a program from computational point of view means that all related tasks have got executed.

5.0 RESULT & DISCUSSION
To justify the application and usefulness of the present method an example of a DCS is considered which consists of a set of “$n = 3$” processors $P = \{p_1, p_2, p_3\}$ and a set of “$m = 8$” executable tasks $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$.

Input of the Algorithm: Data required by the algorithm is given below:

Number of processors available in the system (n) = 3
Number of tasks to be executed (m) = 8

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>$t_2$</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$t_3$</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$t_5$</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$t_6$</td>
<td>6</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$t_7$</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$t_8$</td>
<td>$\infty$</td>
<td>2</td>
</tr>
</tbody>
</table>

Evaluate the average Load on the processors

$P_1 = 6$
$P_2 = 3$
$P_3 = 6$

$L_{avg}(P_j) = 31.3$ where $1 \leq j \leq n$

Maximally Linked Module Determination

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$t_7$</th>
<th>$t_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.333</td>
<td>0.250</td>
<td>0.500</td>
<td>0.167</td>
<td>0.125</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.333</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.200</td>
<td>0.000</td>
</tr>
<tr>
<td>0.250</td>
<td>0.000</td>
<td>0.000</td>
<td>0.250</td>
<td>0.333</td>
<td>0.500</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.500</td>
<td>0.000</td>
<td>0.250</td>
<td>0.000</td>
<td>0.200</td>
<td>0.333</td>
<td>0.500</td>
<td>0.200</td>
</tr>
<tr>
<td>0.167</td>
<td>0.000</td>
<td>0.333</td>
<td>0.200</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.125</td>
<td>0.500</td>
<td>0.333</td>
<td>0.000</td>
<td>0.000</td>
<td>0.167</td>
<td>0.125</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.500</td>
<td>0.167</td>
<td>0.125</td>
<td>0.200</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.200</td>
<td>0.000</td>
<td>0.000</td>
<td>0.125</td>
<td>0.200</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Reduce the ECM by select n tasks those have maximal links
Apply the algorithm developed by Yadav et al. [35] for assigning the task we get the following assignment.

Cluster those tasks which have maximum communication between them.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Processors</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{t}_7</td>
<td>\textit{p}_2</td>
<td>3</td>
</tr>
<tr>
<td>\textit{t}_4</td>
<td>\textit{p}_1</td>
<td>5</td>
</tr>
<tr>
<td>\textit{t}_6</td>
<td>\textit{p}_3</td>
<td>6</td>
</tr>
</tbody>
</table>

Again apply the algorithm developed by Yadav et al. [35] for assigning the task we get the following assignment.

The table given below show the result obtained by the algorithm by incorporating simulated annealing and by extending the algorithm to take into account the limited and non uniform distribution of hardware resources associated with heterogeneous systems. The new MLM heuristic produces complete allocations for assigning static modules to processors of a distributed system. When applied to a large number of randomly-generated systems and to the real world task structures, the methodology creates allocations which appear very competitive to those produced by other allocation methodologies Figure given below shows the relation between throughput and mean service rate of the processors. One can conclude that both are directly linked with other.

### 6.0 Future Research

In the future, affinity, anti-affinity, and exclusion relationships can easily be handled by slight modifications to the MLM algorithm. For example, when some modules have special affinity to some processors (i.e. some modules must be assigned to specific processors because of their special processing capabilities), Step 3 of the MLM algorithm can be altered in a manner which mandates that all modules that have affinity relationships be assigned to the specified processors after which the algorithm would proceed in the usual manner. Similarly, the anti-affinity relationships mandate that modules can never be assigned to certain processing elements. The algorithm can in effect accomplish this by assigning very large execution costs in the corresponding locations of the \( E \) matrix. Exclusion relationships represent the case where two or more modules cannot be assigned to the same processing element because of similar resource requirements. In this case, the algorithm can be made to assign these modules to their best available processor in such a way that no two modules in an exclusion set are assigned to the same processing element.

### 7.0 References


\textit{Continued on Page No. 128}


27. Kumar, Avanish (2001), An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost”, published to the proceedings of International Conference on Mathematical Modeling held at Roorkee, 150-155.


