A Modified and Efficient Algorithm for Static Task Assignment in Distributed Processing Environment

Dr. Kapil Govil
Teerthanker Mahaveer University,
Moradabad

Dr. Avanish Kumar
Bundelkhand University,
Jhansi

ABSTRACT

The Distributed Processing Environment [DPE] in which services provided for 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. The Allocation problems in any computer system play the key role for deciding the performance of the system. The allocation put the direct impact of software resources as well as hardware resources. In DPE, partitioning of the application software in to module and the proper allocation of these modules dissimilar processors are important factors, which determine the efficient utilization of resources. The static model discussed in this paper provide an optimal solution for assigning a set of "m" modules of a task to a set of "n" processors where m > n in a distributed system for evaluation for optimal time of the system.

Keywords

Distributed Processing Environment, task, allocation, module

1. INTRODUCTION

A system in which a large number of separate but interconnected computers do the jobs is called distributed network. In distributed network, services reside at multiple sites. Instead of single large processor being responsible for all aspects of process, there are several separate processor handles these aspects. In the distributed networking the program or tasks are also often developed with the subsets of independent units under distributed environments. Some of the task allocation methods have been reported in the literature, such as Integer Programming [6, 8], Branch and Bound technique [3], Load Balancing [1, 2], Reliability Optimization [11, 4], and Modeling [5]. It has seen that this concept is cost-effective and reliable to meet the optimal solution. Sagar et al [10] proposed the problem of distributing tasks to processors in a distributed computing system is addressed. A task should be assigned to a processor whose capabilities are most appropriate for the processing of that task and excessive interprocessor communication is avoided. The processing costs and communication costs of the tasks are presented by arrays. A task is either assigned to a processor or fused with another task using a simple criterion. The processing and communication costs are then modified suitably. The process continues until all the tasks are assigned to processors. This algorithm also facilitates incorporation of various system constraints. It is applicable to random program structures and to a system containing any number of processors.

2. OBJECTIVE

In a Distributed Processing Environment, a task is allocated to a processor in such a way that extensive Inter Task Communication cost is avoided and the capabilities of the processor suit to the execution requirements of the task. The algorithm discussed in this paper provide an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors (where, m > n) in a Distributed Processing Environment with the goal to maximize the overall throughput of the system and allocated load on all the processors should be evenly balanced. The objective of this problem is to enhance the performance of the distributed processing environment by using the proper utilization of its processors and as well as proper allocation of tasks.

3. TECHNIQUE

In order to evaluate the overall optimal processing time of a distributed processing environment, we have chosen the problem where a set $P = \{p_1, p_2, p_3, \ldots\ldots p_n\}$ of ''n' processors and a set $T = \{t_1, t_2, t_3, \ldots\ldots t_m\}$ of ''m' tasks. The processing time of each task to each and every processor is known and it is mentioned in the Processing Time Matrix PTM(,) of order m x n. The communication time of task is also known and is mentioned in CTM(,) of order m x m. Calculating the average of each row of PTM(,) and store the result in the linear array avg_row() along with their corresponding tasks.

On sorting the avg_row() in ascending order and store the results in linear array avg_row_asc() and their corresponding tasks in $Task_{seq}$. Select the first n task from $Task_{seq}$ and check the communication of those n tasks to each and every next n tasks or less than n tasks to form the cluster of those set of tasks that have maximum communication. If it is less than n and then goto next step otherwise repeat the process. The maximum number of tasks in a cluster shall be less than or equal to $\left[\frac{m}{n}\right]+1$.

Modify the PTM(,) according the n clusters by adding the processing time of those tasks that occurs in the same cluster. Modify the CTM(,) by putting the communication zero amongst those tasks that are in same cluster. On applying the algorithm developed by Kumar et al [7] we get the optimal assignment as well as Processing Time, Communication Time and Optimal Time.

4. ALGORITHM

Start algo

Read the number of processors in n
Read the number of tasks in m
Read the Processor Time Matrix PTM(,) of order m x n
Read the Communication Time Matrix CTM (,) of order m x m
Calculate the average of each row of PTM (,) and store in avg_row()
Sort the avg_row() and store the average in avg_row_asc and correspongding task in $Task_{seq}$
While (number of available tasks $\leq$ n)
{
Select n task from $Task_{seq}$
Check the communication of selected n task to each and every next n tasks
Form the cluster(s) to those set of tasks that have maximum communication
}
Modify the PTM(,) by adding the processing time of tasks in each cluster
Modify the CTM(,) by putting communication zero amongst those tasks that are in the same cluster
Apply [7] algorithm on PTM(,)
Calculate Execution Time, Communication Time
Optimal Time = Execution Time + Communication Time

End algo

5. IMPLEMENTATION

Let us consider a distributed computing system consist a set P = $\{p_1, p_2, p_3, p_4\}$ of 4 processors and a set T = $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$ of 10 tasks. The task - processor graph is shown by figure 1.

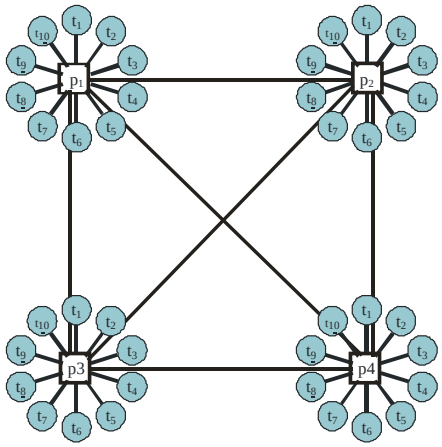

Figure 1: Task Processor Graph

The processing time of every task on various processors are known and mentioned in the following matrix of order 4 x 10, namely, Processing Time Matrix PTM(,):

$$PTM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 6 & 2 & 9 & 3 \\ 5 & 3 & 2 & 1 \\ 8 & 7 & 3 & 4 \\ 1 & 4 & 6 & 3 \\ 4 & 5 & 6 & 2 \\ 2 & 1 & 8 & 9 \\ 2 & 8 & 6 & 7 \\ 6 & 7 & 8 & 9 \\ 4 & 5 & 6 & 2 \\ 3 & 7 & 4 & 2 \end{bmatrix} \end{array}$$

The graphical representation of the inter task communication are shown by figure 2

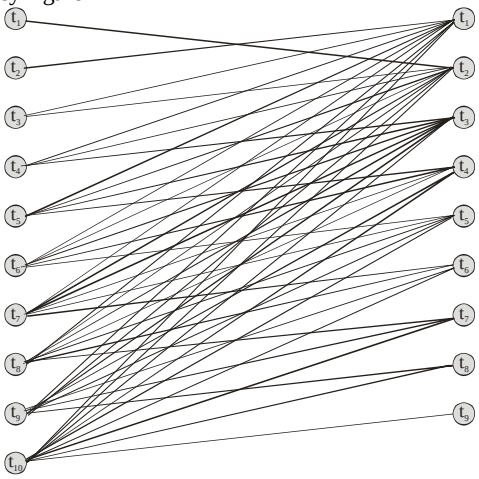

Figure 2: Task Communication

The communication time amongst the tasks has also taken into consideration and shown by the square symmetric matrix i.e. Communication Time Matrix CTM(,) as given below:

$$CTM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \end{array} \\ \begin{bmatrix} 0 & 3 & 2 & 3 & 1 & 2 & 4 & 6 & 7 & 2 \\ & 0 & 8 & 2 & 3 & 4 & 1 & 6 & 1 & 2 \\ & & 0 & 1 & 2 & 3 & 4 & 3 & 5 & 2 \\ & & & 0 & 4 & 6 & 3 & 4 & 2 & 2 \\ & & & & 0 & 8 & 3 & 4 & 9 & 2 \\ & & & & & 0 & 6 & 7 & 5 & 4 \\ & & & & & & 0 & 3 & 2 & 1 \\ & & & & & & & 0 & 1 & 6 \\ & & & & & & & & 0 & 5 \\ & & & & & & & & & 0 \end{bmatrix} \end{array}$$

Calculating the average of each row of PTM(,) and store the result in linear array avg_row( ) along with their corresponding tasks i.e.,

$$avg\_row(\,) = \begin{Bmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \\ 5 & 2.75 & 5.5 & 3.5 & 4.25 & 5 & 5.75 & 6 & 4.25 & 4 \end{Bmatrix}$$

On sorting the avg_row( ) in ascending order and store the results in linear array avg_row_asc( ) and their corresponding tasks in $Task_{seq}$. These are given below;

$$avg\_row\_asc(\,) = \begin{Bmatrix} t_2 & t_4 & t_{10} & t_5 & t_9 & t_1 & t_6 & t_3 & t_7 & t_8 \\ 2.75 & 3.5 & 4 & 4.25 & 4.25 & 5 & 5 & 5.5 & 5.75 & 6 \end{Bmatrix}$$

$Task_{seq} = \{t_2, t_4, t_{10}, t_5, t_9, t_1, t_6, t_3, t_7, t_8\}$

Select the first n task from $Task_{seq}$ and check the communication of these n tasks to each and every next n tasks or less than n tasks to form the cluster of those set of tasks that have maximum communication as considered in CTM(,). If it is less than n then goto next step otherwise repeat the process. The maximum number of tasks in a cluster shall be less then or equal to $\left[\frac{m}{n}\right]+1$. We obtain the following n clusters as given below:

Cluster 1: $(t_1 * t_5 * t_7)$
Cluster 2: $(t_2 * t_3 * t_8)$
Cluster 3: $(t_4 * t_6)$
Cluster 4: $(t_9 * t_{10})$

Modify the PTM(,) according the n clusters by adding the processing time of those tasks that occurs in same cluster.

$$PTM(,) = \begin{array}{c} \\ t_1*t_5*t_7 \\ t_2*t_3*t_8 \\ t_4*t_6 \\ t_9*t_{10} \end{array} \begin{array}{c} \begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \end{array} \\ \begin{bmatrix} 12 & 15 & 21 & 12 \\ 19 & 17 & 14 & 14 \\ 3 & 5 & 14 & 12 \\ 7 & 12 & 10 & 4 \end{bmatrix} \end{array}$$

Modify CTM(,) by putting the communication zero amongst those tasks that are in same cluster. The modified CTM(,) as:

$$CTM(,) = \begin{array}{c} \\ t_1*t_5*t_7 \\ t_2*t_3*t_8 \\ t_4*t_6 \\ t_9*t_{10} \end{array} \begin{array}{c} \begin{array}{cccc} t_1*t_5*t_7 & t_2*t_3*t_8 & t_4*t_6 & t_9*t_{10} \end{array} \\ \begin{bmatrix} \mathbf{0} & \mathbf{3} & \mathbf{3} & \mathbf{7} \\ & \mathbf{0} & \mathbf{2} & \mathbf{1} \\ & & \mathbf{0} & \mathbf{2} \\ & & & \mathbf{0} \end{bmatrix} \end{array}$$

On, applying the algorithm developed by Kumar et al [7] we get the optimal assignment as well as Processing Time, Communication Time and Optimal Time. The graphical representation of the optimal assignment is shown by the figure 3.

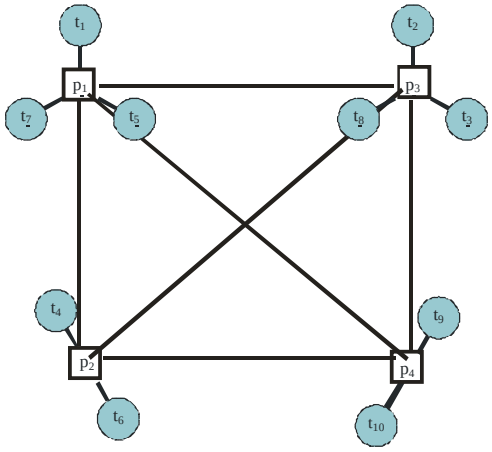

Figure 3: Optimal Assignment Graph

Table 1

Processor	Task	Execution Time	Communication Time	Optimal Time
p_1	$t_1*t_5*t_7$	12	18	53
p_2	t_4*t_6	14		
p_3	$t_2*t_3*t_8$	5		
p_4	t_9*t_{10}	4		

6. CONCLUSION

Here we have taken 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 problem is based on the consideration of processing time 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 time. The optimal result of the example that is considered to test the algorithm and it is mentioned in the implementation section of the problem are as given below.

Table 2. Optimal result

Processor	Task	Execution Time	Communication Time	Optimal Time
p_1	$t_1*t_5*t_7$	35	18	53
p_2	t_4*t_6			
p_3	$t_2*t_3*t_8$			
p_4	t_9*t_{10}			

The processorwise execution time is shown by the figure 4.

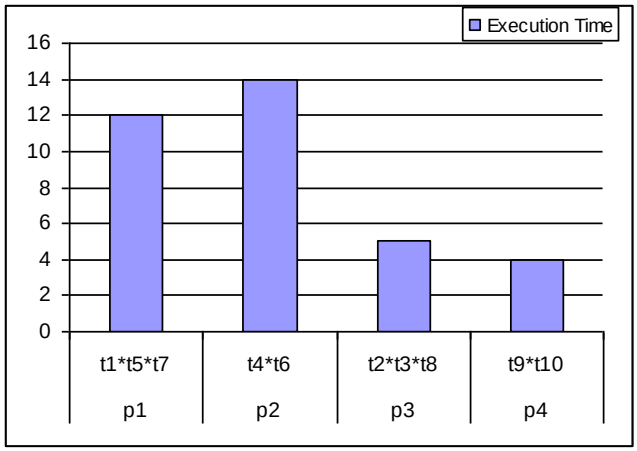

Figure 4: Processorwise Execution Time Graph

As we know that, the analysis of an algorithm is mainly focuses on its complexity. The complexity is a function of input size 'n'. It is referred to as the amount of time required by an algorithm to run to completion. The time complexity of the above mentioned algorithm is $O(mn^2)$. Table 3 shows the time complexity of the present algorithm.

Table 3. Complexity of Time Complexity

Processors n	Tasks m	Time Complexity of algorithm [9] $O(n^m)$	Time Complexity of present algorithm $O(mn^2)$
3	4	81	36
3	5	243	45
3	6	729	54
3	7	2187	63
3	8	6561	72
4	5	1024	80
4	6	4096	96
4	7	16384	112
4	8	65536	128
4	9	262144	144
5	6	15625	150
5	7	78125	175
5	8	390625	200
5	9	1953125	225
5	10	9765625	250

From the table 3 it is clear that present algorithm is much better for optimal allocation of tasks that upgrade the performance of distributed network. For the different values of n = 3, 4 and 5 the complexity comparison with present algorithm to Richard et. al. [9] is shown through graph 5, 6, and 7. These three graphs are also indicates that our suggested algorithm is much faster then that of suggested by Richard et. al. [9].

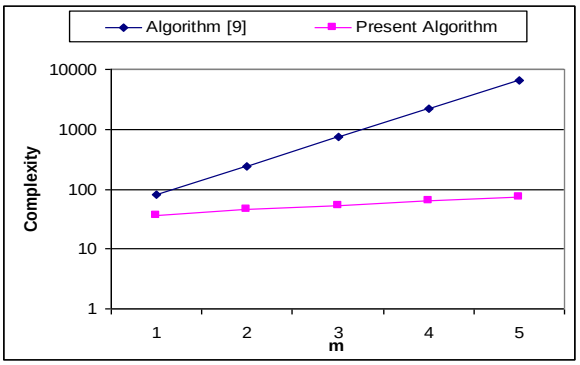

Figure 5: Comparison Graph for n=3

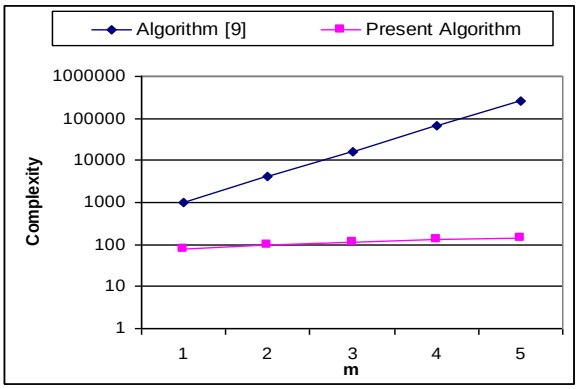

Figure 6: Comparison Graph for n=4

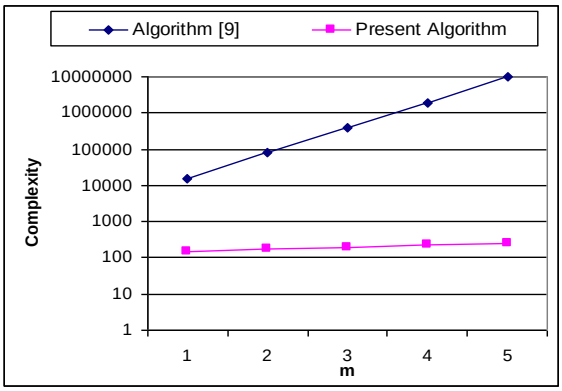

Figure 7: Comparison Graph for n=5

7. REFERENCES

[1] Andrey G. Bronevich, Wolfgang Meyer 2008. Load balancing algorithms based on gradient methods and their analysis through algebraic graph theory. Journal of Parallel and Distributed Computing, Volume 68, Issue 2, (February 2008), 209-220.

[2] Bruce Hendrickson, Karen Devine 2000. Dynamic load balancing in computational mechanics. Computer Methods in Applied Mechanics and Engineering, Volume 184, Issues 2-4, (April 2000), 485-500.

[3] Dorta, C. Leon, C. Rodríguez 2010. Performance analysis of Branch-and-Bound skeletons.Mathematical and Computer Modelling, Volume 51, Issues 3-4, February 2010, Pages 300-308

[4] Gamal Attiya, Yskandar Hamam 2006. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. Journal of Parallel and Distributed Computing, Volume 66, Issue 10, (October 2006), 1259-1266.

[5] Jeffery L. Kennington, Eli V. Olinick, Gheorghe Spiride 2007. Basic mathematical programming models for capacity allocation in mesh-based survivable networks. Omega, Volume 35, Issue 6, (December 2007), 629-644.

[6] Kuban Altınel, Necati Aras, Evren Güney, Cem Ersoy 2008. Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks. Computer Networks, Volume 52, Issue 12, (August 2008), 2419-2431.

[7] A. Kumar, M. P. Singh, and P. K. Yadav, A Fast Algorithm for Allocating Tasks in Distributed Processing System, Proceedings of the '30th Annual Convention of CSI, Hyderabad, (1995), 347-358.

[8] Maria Joao Alves, Joao Clímaco 2007. A review of interactive methods for multiobjective integer and mixed-integer programming. European Journal of Operational Research, Volume 180, Issue 1, (July 2007), 99-115.

[9] Richard R.Y., Lee, E.Y.S. and Tsuchiya, M. 1982. A Task Allocation Model for Distributed Computer System, IEEE Transactions on Computer, 31, 41-47.

[10] G. Sagar and A. K. Sarje, Task Allocation Model for Distributed System, International Journal of System Science, Vol. 22(1991), 1671-1678.

[11] Bo Yang, Huajun Hu, Suchang Guo 2009. Cost-oriented task allocation and hardware redundancy policies in heterogeneous distributed computing systems considering software reliability. Computers & Industrial Engineering, Volume 56, Issue 4, (May 2009), 1687-1696.