An Algorithm For Task Assignment In Distributed Network Using Static Approach

Kapil Govil[1], Nikhil Govil[2]

ABSTRACT

The Distributed Processing Environments [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 distributed Computer systems, 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 using algorithm of Assignment problem for assigning a set of "m" modules of a task to a set of "n" processors where m > n in a computer system for evaluation for optimal time of the system.

Keywords : Distributed Processing Environment, Processor, Task, Allocation, Assignment Problem.

[1]Department of Computer Applications, Teerthanker Manaveer University, Moradabad – 244001, (UP), INDIA
drkapilgovil@gmail.com

[2]Department of Computer Science & Engineering Manav Rachna College of Engineering, Faridabad, (HR), INDIA
nikhilgovil@gmail.com

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. A distributed network is looks like a virtual uniprocessor. In the distributed networking the program or tasks are also often developed with the subsets of independent units under distributed environments. It has seen that this concept is cost-effective and reliable to meet the optimal solution.

One of the major research problems for distributed networks is the allocation problem, in which tasks are assigned to various processors of the network, in such a way that processing time is to be minimized as per the requirement. These problems may be categorized as static [2,10,11,13] and dynamic [1,6,12,13,16] types of task allocation. Some of the other related methods have been reported in the literature, such as, Integer Programming [3,5,14,15], Load Balancing [4,6,18] and Modeling [7,9,17]. Tasks are allocated to various processors of the distributed network in such a way that overall processing time of the network should be minimized. As it is well known that the tasks are more than the number of processors of the network.

2. OBJECTIVE

The objective of the present research paper is to enhance the performance of the distributed networks by using the

proper utilization of its processors and as well as proper allocation of tasks. The type of allocation of task to the processor is static. As in this paper the performance is measured in terms of time, so we have to minimize the time to obtain the best performance of the processors.

3. Technique

In order to evaluate the overall optimal processing time of a distributed network, 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, where m>n. The processing time of each task to each and every processor is known and it is mentioned in the Processing Time Matrix of order m x n. After making a matrix of same order taking in ascending order of its sum of row and sum of column, we apply the algorithm of assignment problem on it. For each processor we evaluate the overall assignment of each task; and assignment of the task on the processor which has the minimum processing time. Finally we compute total processing by adding total processing time of task, which are assigned at specified processor.

3.1. Algorithm

Start algo

Read the number of task in m

Read the number of processor in n

For i = 1 to m

For j = 1 to n

Read the value of processing time (t) in Processor Time Matrix namely PTM(,)

j = j + 1

Endfor

i = i + 1

Arrange the PTM(,) in ascending order of its row_sum and column_sum

i = 1

While all tasks != SELECTED

Select the biggest possible square matrix from left upper corner and store it into SMi(,)

Apply algorithm of Assignment Problem [KANT2002] on SM_i(,)

i = i + 1

Endwhile

Club processorwise overall optimal processing time

State the results

End algo

4. Implementation

In the present research paper, the distributed network consist a set P of 4 processors $\{p_1, p_2, p_3, p_4\}$ and a set T of 10 tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$. It is shown in the figure1. The processing time (t) of each task to each and every processor is known and it is mentioned in the Processor Time Matrix PTM(,) of order 10 x 4.

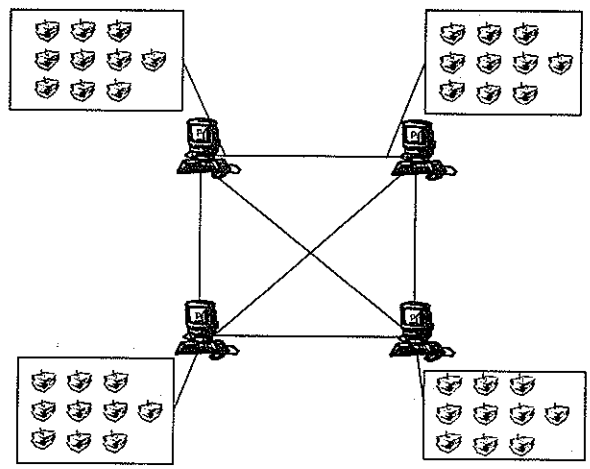

Figure 1 : Processor Task Combination

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

Now, calculate the sum of each row and column and store the results in Modified Processor Task Matrix MPTM(,) of order 10 x 4.

$$MPTM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \\ t_{10} \\ \text{Column\_Sum} \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} 6 & 2 & 9 & 3 \\ 5 & 3 & 2 & 1 \\ 8 & 7 & 3 & 4 \\ 1 & 4 & 6 & 3 \\ 4 & 5 & 6 & 2 \\ 2 & 1 & 8 & 9 \\ 2 & 8 & 6 & 7 \\ 6 & 7 & 8 & 9 \\ 4 & 5 & 6 & 2 \\ 3 & 7 & 4 & 2 \end{bmatrix} \\ \begin{array}{cccc} 41 & 49 & 58 & 42 \end{array} \end{array} \begin{array}{c} \text{Row\_Sum} \\ \begin{bmatrix} 20 \\ 11 \\ 22 \\ 14 \\ 17 \\ 20 \\ 23 \\ 30 \\ 17 \\ 16 \end{bmatrix} \end{array}$$

By arranging the MPTM(,) in ascending order of their row_sum and column_sum, we get Arranged Processor Task Matrix APTM(,) of order 10 x 4.

$$APTM(,) = \begin{array}{c} \\ t_2 \\ t_4 \\ t_{10} \\ t_5 \\ t_9 \\ t_1 \\ t_6 \\ t_3 \\ t_7 \\ t_8 \\ \text{Column\_Sum} \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_4 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 5 & 1 & 3 & 2 \\ 1 & 3 & 4 & 6 \\ 3 & 2 & 7 & 4 \\ 4 & 2 & 5 & 6 \\ 4 & 2 & 5 & 6 \\ 6 & 3 & 2 & 9 \\ 2 & 9 & 1 & 8 \\ 8 & 4 & 7 & 3 \\ 2 & 7 & 8 & 6 \\ 6 & 9 & 7 & 8 \end{bmatrix} \\ \begin{array}{cccc} 41 & 42 & 49 & 58 \end{array} \end{array} \begin{array}{c} \text{Row\_Sum} \\ \begin{bmatrix} 11 \\ 14 \\ 16 \\ 17 \\ 17 \\ 20 \\ 20 \\ 22 \\ 23 \\ 30 \end{bmatrix} \end{array}$$

Now, selecting the biggest possible square matrix from left upper corner, we get Selected Matrix $SM_i(,)$ of order 4 x 4.

$$SM_1(,) = \begin{array}{c} \\ t_2 \\ t_4 \\ t_{10} \\ t_5 \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_4 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 5 & 1 & 3 & 2 \\ 1 & 3 & 4 & 6 \\ 3 & 2 & 7 & 4 \\ 4 & 2 & 5 & 6 \end{bmatrix} \end{array}$$

Now, by using algorithm of Assignment Problem [KANT2002] on $SM_i(,)$, we get task allocation as –

Processor	Task
p_1	t_4
p_2	t_2
p_3	t_{10}
p_4	t_5

Now, we eliminate allocated tasks (i.e. t_2, t_4, t_5, t_{10}) from APTM(,); and again selecting the biggest possible square matrix from left upper corner, we get selected matrix $SM_2(,)$ of order 4 x 4.

$$SM_2(,) = \begin{array}{c} \\ t_9 \\ t_1 \\ t_6 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_4 & p_2 & p_3 \end{array} \\ \begin{bmatrix} 4 & 2 & 5 & 6 \\ 6 & 3 & 2 & 9 \\ 2 & 9 & 1 & 8 \\ 8 & 4 & 7 & 3 \end{bmatrix} \end{array}$$

Now, on applying the algorithm of Assignment Problem [8] on matrix $SM_2(,)$, we get task allocation as –

Processor	Task
p_1	t_6
p_2	t_1
p_3	t_3
p_4	t_9

Again, we eliminate allocate tasks (i.e. t_1, t_3, t_6, t_9) from APTM(,) and again selecting the biggest possible square matrix from left upper corner, we get Selected Matrix $SM_3(,)$ of order 2 x 2.

$$SM_3(,) = \begin{array}{c} \\ t_7 \\ t_8 \end{array} \begin{array}{c} \begin{array}{cc} p_1 & p_2 \end{array} \\ \begin{bmatrix} 2 & 7 \\ 6 & 9 \end{bmatrix} \end{array}$$

Now, on applying the algorithm of Assignment Problem [8] on matrix $SM_3(,)$, we get task allocation as –

Processor	Task
p_1	t_7
p_2	-
p_3	-
p_4	t_8

The overall processorwise task allocation along with optimal processing time is mentioned in table1.

Table 1 : Processorwise Task Allocation

Processor	Assigned Task	Processing Time
p_1	$t_4*t_6*t_7$	5
p_2	t_1*t_2	5
p_3	t_3*t_{10}	7
p_4	$t_5*t_8*t_9$	13
Overall Optimal Processing Time		30

The graphical representation of the optimal assignment is shown in figure 2.

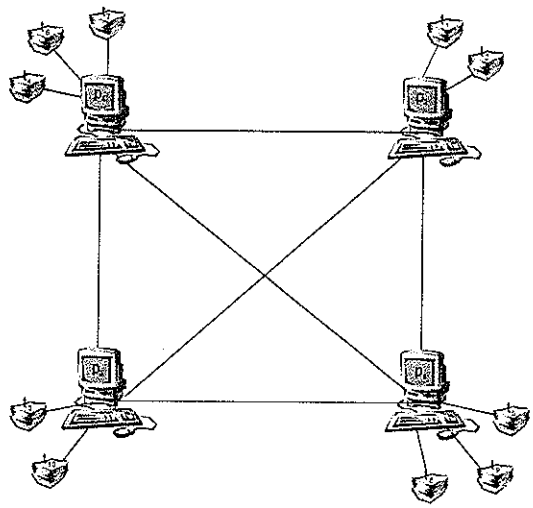

Figure 2 : Optimal Assignment

The processorwise processing time graph is shown in the figure 3.

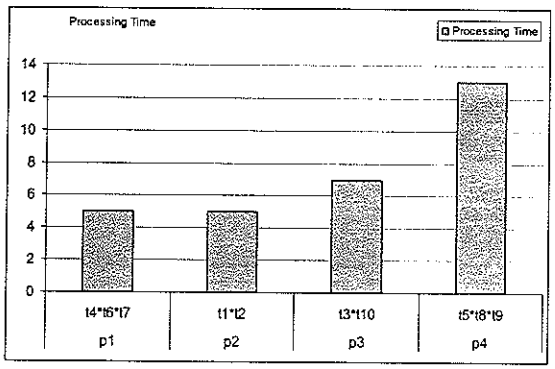

Figure 3 : Processorwise Processing Time Graph

5. 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 network. The model mentioned in this paper 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. Here, performance is measured in terms of processing time of the task that has been processed by the processors of the network and also these tasks have been processed optimally. As we know that, the analysis of an algorithm is mainly focuses on time complexity. Time 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). By taking several input examples, the above algorithm returns following results as in table 2.

Table 2 : Optimal Results

No. of processors (n)	No. of tasks (m) Results	Optimal
3	4	12
3	5	15
3	6	18
3	7	21
3	8	24
4	5	20
4	6	24
4	7	28
4	8	32
4	9	36
5	6	30
5	7	35
5	8	40
5	9	45
5	10	50

The graphical representation of results of table 2 are shown by figure 4, 5 and 6 as mentioned below,

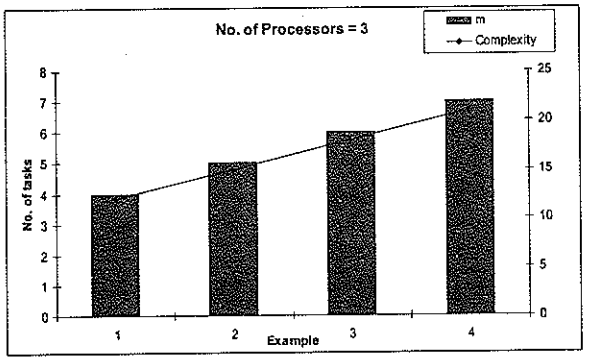

Figure 4 : Complexity Graph

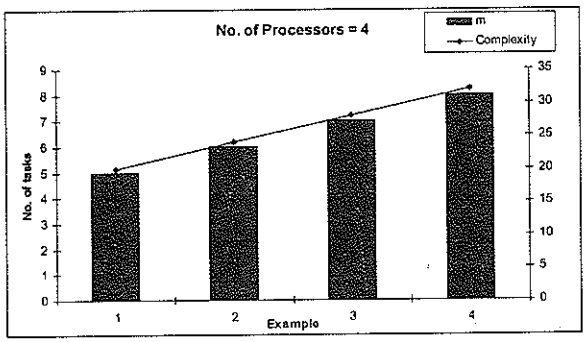

Figure 5 : Complexity Graph

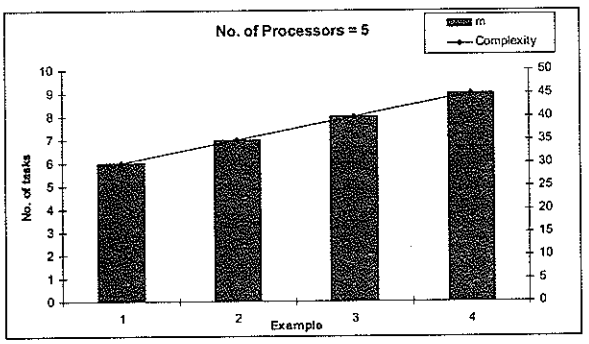

Figure 6 : Complexity Graph

The performance of the algorithm is compared with the algorithm suggested by Sagar et. al. [19]. Table 3, shows the time complexity comparison between algorithm [19] and present algorithm.

Table 3 : Comparison Table

Processors n	Tasks m	Time Complexity of algorithm (Suggested by Sagar et-al) $O(m^2n)$	Time Complexity of present algorithm O(mn)
3	4	48	12
3	5	75	15
3	6	108	18
3	7	147	21
3	8	192	24
4	5	100	20
4	6	144	24
4	7	196	28
4	8	256	32
4	9	324	36
5	6	180	30
5	7	245	35
5	8	320	40
5	9	405	45
5	10	500	50

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. Figures 7, 8 and 9 shows the, pictorial representation between algorithm 19 and present algorithm.

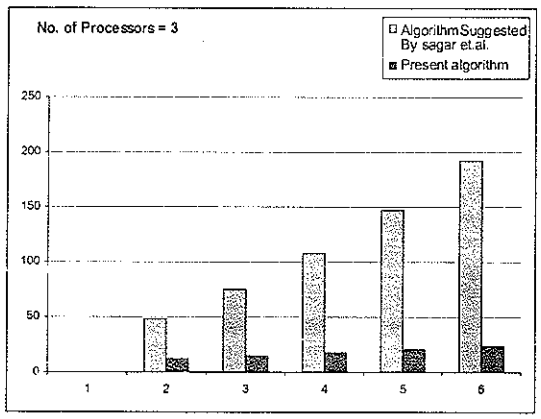

Figure 7 : Comparison Graph

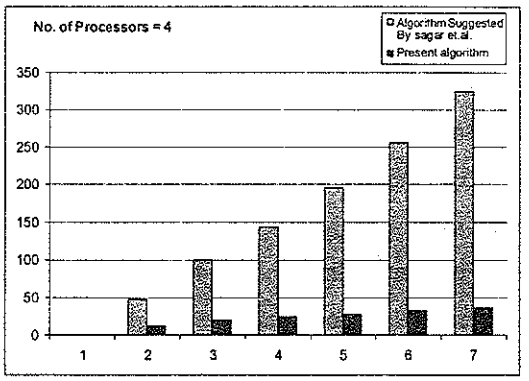

Figure 8 : Comparison Graph

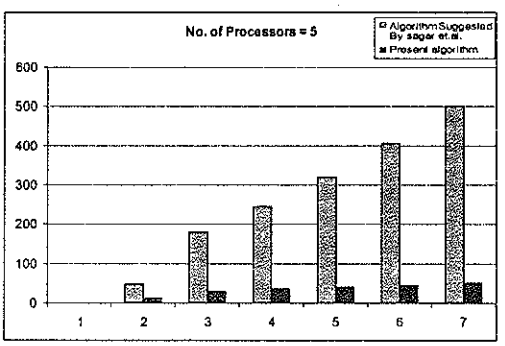

Figure 9: Comparison Graph

R EFERENCES

1. Avanish Kumar, *Optimizing for the Dynamic Task Allocation*, Proceedings of the 'III Conference of the International Academy of Physical Sciences, Allahabad; (1999), 281-294.

2. Avanish Kumar, *An Algorithm for Optimal Index to Tasks Allocation Based on Reliability and cost*, Proceedings of 'International Conference on Mathematical Modeling, Roorkee; (2001), 150-155.

3. Brian Ensink, Joel Stanley, and Vikram Adve,. *Program Control Language: a programming language for adaptive distributed applications*; Elsevier Inc., Vol. 63(2003), No. 12, 1082 –1104.

4. Daniel Grosu, and Anthony T. Chronopoulos, *Noncooperative load balancing in distributed systems;* Elsevier Inc., Vol. 65(2005), No. 9, 1022-1034.

5. O.I. Dessoukiu-EI, and W. H. Huna, *Distributed Enumeration on Network Computers*; IEEE Transactions on Computer, Vol. 29(1980), 818-825.

6. Jacques Bahi, Raphael Couturier, and Flavien Vernier, *Synchronous distributed load balancing on dynamic networks*. Elsevier Inc., Vol. 65(2005), No. 11, 1397-1405.

7. Javier Contreras, Arturo Losi, Mario Russo, and Felix F. Wu, *DistOpt: A Software Framework for Modeling and Evaluating Optimization Problem Solutions in Distributed Environments*, Elsevier Inc., Vol. 60(2000) No. 6, 741 – 763.

8. Kant Swaroop, P. K. Gupta and Man Mohan, *Operations Research*, Sultan Chand & Sons, ISBN No.: 8180540200, 2002.

9. Kent Fitzgerald, Shahram Latifi, and Pradip K. Srimani, *Reliability Modeling and Assessment of the Star-Graph Networks*, IEEE Transactions on Reliability, Vol. 51(2002), 49-57.

10. A. Kumar, M. P. Singh and P. K. Yadav, *An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System*, Proceedings of the '19th National System Conference, SSI, Coimbatore, (1995), 82-87.

11. 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.

12. A. Kumar, M. P. Singh, and P. K. Yadav, *An Efficient Algorithm for Multi- processor Scheduling with Dynamic Reassignment*, Proceedings of the '6th National seminar on theoretical Computer Science, Banasthally Vidyapeeth, (1996), 105-118.

13. Yu-Kwong Kwok, Anthony A. Maciejewski, Howard Jay Siegel, Ishfaq Ahmad, and Arif Ghafoor, *A semi-static approach to mapping dynamic iterative tasks onto heterogeneous computing systems*. Elsevier Inc., Vol. 66(2006) No. 1, 77-98.

14. K. B. Misra, and U. Sharma, *An Efficient Algorithm to solve Integer Programming Problem arising in System Reliability Design*, IEEE Transactions on Reliability, Vol. 40(1991), 81-91.

15. Muhammad K. Dhodhi, Imtiaz Ahmad, Anwar Yatama and Ishfaq Ahmad. *An Integrated Technique for Task Matching and Scheduling onto Distributed Heterogeneous Computing Systems* Elsevier Inc., Vol. 62(2002) No. 9, 1338 – 1361.

16. J. Palmer, and I. Mitrani, *Optimal and heuristic policies for dynamic server allocation*. Elsevier Inc., Vol. 65(2005) No. 10, 1204-1211.

17. Rene L. Bierbaum, Thomas D. Brown, and Thomas J. Kerschen, *Model-Based Reliability Analysis*, IEEE Transactions on Reliability, Vol. 51(2002), 133-140.

18. Saeed Iqbal, and Graham F. Carey, *Performance analysis of dynamic load balancing algorithms with variable number of processors*. Elsevier Inc., Vol. 65(2005), No. 8, 934-948.

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

Authors Biography

Kapil Govil is currently working as senior lecturer in Teerthanker Mahaveer University, Moradabad. He has submitted his Ph.D. thesis to Bundelkand University, Jhansi in December 2009. He has authored 2 books. He has presented 4 papers in international conferences & 1 paper in National conference. He has published 5 papers in International / National journals / Conference proceedings.

Nikhil Govil is currently working as Lecturer in Manav Rachna College of Engineering, Faridabad. He received M.Tech degree in Computer Science & Engineering from USIT, Guru Gobind Singh Indraprastha University, Delhi with distinction and MCA degree from IGNOU. His areas of interest are Software Engineering, Data Warehousing & Data Mining, Computer Networks etc.