A Smart Algorithm for Dynamic Task Allocation for Distributed Processing Environment

Dr. Kapil Govil
Teerthanker Mahaveer University,
Moradabad

ABSTRACT
A Distributed Processing Environment (DPE) consists of multiple autonomous computers that communicate through a communication media. In DPE a task is divided into many fractions and each of which is to be get processed. The task allocation problem can be explained in terms of number of tasks and number of processors available. In the present method propose a dynamic model for task allocation in DPE. Present method describes the allocation of m tasks in the environment of distributed processing with n processors (m>n) that completes in k number of phases. This method allocates the tasks to the processor to increases the performance of the DPE; and based on the inter task communication cost between executing task and another tasks. Residing cost and reallocation cost in various phases has also taken in consideration. On implemented the suggested algorithm we have obtained the phase wise optimal allocation and overall optimal cost. The run time complexity has been computed and compared with existing approaches. It is found that suggested algorithm is much better as compared to others.

Keywords
Distributed Processing Environment, Task, Allocation, Residing Cost, Reallocation Cost.

1. INTRODUCTION
Most distributed environment nowadays are consists of various nodes having different functions and/or different processing capabilities and speeds. A heterogeneous distributed environment consists of a set of nodes (autonomous computers) with same functionality but different processing capability. Distributed Processing Environment (DPE) offer the potential for improved performance and resource sharing. To make the best use of the computational power available, it is essential to assign the tasks dynamically to that processor whose characteristics are most appropriate for the execution of the tasks in DPE. We have developed a mathematical model for allocating “M” tasks of distributed program to “N” multiple processors (M > N) that minimizes the total cost of the program.

One of the major research problems for DPE is the task allocation problem, in which tasks are assigned to various processors of the network, in such a way that processing cost is to be minimized as per the requirement. These problems may be categorized as static [1, 2, 3, 4, 5, 6, 7, 8, 9] and dynamic [10, 11, 12, 13, 14] types of task allocation. Some of the other related methods have been reported in the literature, such as, Integer Programming [15], Load Balancing [16, 17, 18, 19, 20, 21, 22, 23], Divide and Conquer [24], Grid Computing [25] and Branch and Bound [26, 27]. Tasks are allocated to various processors of the distributed environment in such a way that overall processing cost 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
This research is aimed to find out the number of autonomous computer required to design a Distributed Processing Environment (DPE) for a specific problem domain. Here we have presented a mathematical model of a general dynamic task allocation mechanism. In this problem we have chosen task allocation mode of is dynamic in nature. Tasks are divided into phases and while a task is executed in a phase then remaining tasks are residing in that phase. Task execution cost, residing cost and inter – task communication cost has considered. As in this problem the performance is measured in terms of Execution Cost, so we have to minimize the execution cost and remaining parameters to obtain the optimal performance of the systems.

3. TECHNIQUE
In order to evaluate the overall optimal execution cost of a distributed system, we have considered the problem that consist a set P = {p1, p2, p3, …pn} of ’n’ processors and set T = {t1, t2, t3, t4,…tm} of ’m’ tasks divided into k phases. Here it is assumed that the tasks m are more than the number of processors n i.e. m > n. The phase wise efficiency of individual processor is given in the form of Execution Cost Matrix ECM(,,) of order k x m x n. The Residing Cost for residing the unprocessed tasks on the processor is mentioned in Residing Cost Matrix RCM(,,) of order k x m x n. The Inter Task Communication Cost amongst the tasks is considered and is mentioned in the Inter Task Communication Cost Matrix ITCCM(,) of order m x k and when a task is shifted from one processor to another processor then it incurred some cost i.e. reallocation cost and it is given in the Reallocation Cost Matrix RECM(,) of order m x k. For each phase sum up ECM(,,) and RCM(,,) to obtain ERCM(,). Obtain the sum of each row of ERCM(,) and arrange them in ascending order and store in sum_row_asc(). Now, select first n tasks from ERCM(,,) to store in sum_row() and store them in ERCMI(,). Repeat the process for next n or less than n tasks. Made the allocation in all sub matrices of ERCM(,,) by using Kumar et al [4]. Evaluate the Execution Cost, Communication Cost and Reallocation Cost. Repeat the process for all phases; and finally summing up the value of Execution Cost, Communication Cost and Reallocation Cost to get the phase wise optimal cost. Obtain the sum of optimal cost of each phase to evaluate the overall optimal cost.

4. ALGORITHM

```
Start Algo
    Read the number of tasks in m
    Read the number of processors in n
    Read the number of phases in k
    Read the Execution Cost Matrix ECM(,,) of order k x m x n
    Read the Residing Cost Matrix RCM(,,) of order k x m x n
    Read the Inter Task Communication Cost Matrix ITCCM(,,) of order k x m x n
    Read the Reallocation Cost Matrix RECM(,,) of order m x k
    For I = 1 to m
        Phase I:
        Sum up ECM(,,) and RCM(,,) and store the results in ERCM(,,)
        Store the sum of each row of ERCM(,,) and store it in sum_row()
        sort sum_row() and store the results in sum_row_asc()
        While (all tasks of sumrow_asc() != SELECTED)
        {
            Make partition of ERCM(,,) for n tasks, store it in ERCMI(,,)
            Apply algorithm of Kumar et al [4] on ERCMI(,,)
        }
        Compute Execution Cost (EC), Communication Cost (CC) and Reallocation Cost (RC)
        Total Cost = EC + CC + RC
        I = I + 1
        Optimal Cost = Σ(Total Cost)
    Endfor
End Algo
```

5. IMPLEMENTATION

In the present problem, let us consider a distributed processing environment which is made up of four tasks {t_1, t_2, t_3, t_4} to be allocated on two processors {p_1, p_2} in five phases. The phase wise efficiency of individual processor is given in the form of Execution Cost Matrix ECM(,,) of order k x m x n where k is the number of phases, m is the number of tasks and n is the number of processors. The ECM(,,) is as given below.

ECM(,,)=

Phase	Task	Execution Cost	
		p_1	**p_2**
1	t_1	4	3
	t_2	-	-
	t_3	-	-
	t_4	-	-
2	t_1	-	-
	t_2	6	5
	t_3	-	-
	t_4	-	-
3	t_1	-	-
	t_2	-	-
	t_3	2	4
	t_4	-	-
4	t_1	-	-
	t_2	-	-
	t_3	-	-
	t_4	3	∞
5	t_1	4	5
	t_2	-	-
	t_3	-	-
	t_4	-	-

The Residing Cost for those tasks that reside on the processor is mentioned in Residing Cost Matrix RCM(,,) of order k x m x n. The RCM(,,) is as,

RCM(,,)=

Phase	Task	Residing Cost	
		p_1	**p_2**
1	t_1	-	-
	t_2	1	2
	t_3	2	1
	t_4	3	2
2	t_1	1	2
	t_2	-	-
	t_3	2	3
	t_4	1	4
3	t_1	3	1
	t_2	2	3
	t_3	-	-
	t_4	3	1
4	t_1	1	3
	t_2	2	1
	t_3	1	2
	t_4	-	-
5	t_1	-	-
	t_2	2	1
	t_3	1	2
	t_4	1	3

Inter Task Communication Cost between executing task and another tasks is taken in the form of matrix Inter Task Communication Cost Matrix ITCCM(,) of order m x k. The ITCCM(,) is given as follow,

ITCCM(,)=

Phase→ / **Task ↓**	**1**	**2**	**3**	**4**	**5**
t_1	-	3	4	2	-
t_2	1	-	3	4	1
t_3	4	2	-	5	2
t_4	1	2	3	-	0

When an allocated task is shifted from one processor to another processor during the next phase then reallocation cost is mentioned at the end of each phase. The reallocation cost is mentioned below in the reallocation cost matrix RECM(,) of order m x k.

RECM(,)=

Phase→ / **Task ↓**	**1**	**2**	**3**	**4**	**5**
t_1	1	1	4	3	-
t_2	2	2	3	3	-
t_3	3	2	2	2	-
t_4	1	3	1	1	-

In phase 1 task t_1 shall have to execute while t_2, t_3 & t_4 shall be residing. Sum up the ECM(,,) and RCM(,,) and store the results in ERCM(,) we get

$$\text{ERCM}(,) = \begin{array}{c|cc} & p_1 & p_2 \\ \hline t_1 & 4 & 3 \\ t_2 & 1 & 2 \\ t_3 & 2 & 1 \\ t_4 & 3 & 2 \end{array}$$

Obtain the sum of each row of ERCM(,) and store it in a linear array sum_row().

$$\text{sum\_row()}=\begin{Bmatrix} t_1 & t_2 & t_3 & t_4 \\ 7 & 3 & 3 & 5 \end{Bmatrix}$$

Arrange the sum_row() in ascending order and we get the following:

$$\text{sum\_row\_asc()}=\begin{Bmatrix} t_2 & t_3 & t_4 & t_1 \\ 3 & 3 & 5 & 7 \end{Bmatrix}$$

Now partition the ERCM(,) by selecting the first two tasks; store it in ERCM1(,), we get the first subproblem:

$$\text{ERCM1}(,) = \begin{array}{c|cc} & p_1 & p_2 \\ \hline t_2 & 1 & 2 \\ t_3 & 2 & 1 \end{array}$$

Select next two tasks and get the second subproblem:

$$\text{ERCM1}(,) = \begin{array}{c|cc} & p_1 & p_2 \\ \hline t_4 & 3 & 2 \\ t_1 & 4 & 3 \end{array}$$

On applying the algorithm developed by Kumar et al [4] we get the following allocation and their corresponding costs during the phase - 1,

Phase	**Executing Task**	**Processor**	**Assigned Task**	**Execution Cost**	**Communication Cost**	**Reallocation Cost**	**Total (EC+CC+RC)**
1	t_1	p_1	$t_2 * t_4$	4	2	0	10
		p_2	$t_3 * t_1$	4			

On repeating the above process for all phases i.e. 2 to 5 we get the following results:

Phase	Executing Task	Processor	Assigned Task	Execution Cost	Communication Cost	Reallocation Cost	Phasewise Optimal Cost (EC+CC+RC)
1	t_1	p_1	$t_2 * t_4$	4	2	0	10
		p_2	$t_3 * t_1$	4			
2	t_2	p_1	$t_1 * t_4$	2	5	3	18
		p_2	$t_3 * t_2$	8			
3	t_3	p_1	$t_1 * t_3$	5	5	5	19
		p_2	$t_4 * t_2$	4			
4	t_4	p_1	$t_3 * t_4$	4	5	3	16
		p_2	$t_2 * t_1$	4			
5	t_1	p_1	$t_3 * t_4$	2	2	5	15
		p_2	$t_2 * t_1$	6			
Overall Optimal Cost							78

6. CONCLUSION

In this problem we have presented an efficient solution to the dynamic allocation problem. Starting with the definition of the phase of a modular program, a model based on dynamic programming approach is suggested. Earlier the researchers advised that the dynamic allocation strategy is the best allocation technique as it facilitates the user to take decision for allocating the during run time. The suggested algorithm is implemented on the several sets of input data and it is recorded that algorithm is workable in all the case. Here we have considered the phases and each phase has the tasks are to be processed by the processors. In each phase only one task shall be executing on these processors. During the next phase an executing task may remain on the same processor for execution or may shift to another processor, in case of shifting the task to another processor, it added the reallocation cost. The impact of inter task communication cost is to be considered. Thus phase wise optimal costs are obtained. In this model, there are five phases and each phase has the equal numbers of tasks. Optimal allocation has been obtained along with phase wise optimal costs. The overall optimal cost is found to be 78. The detailed optimal results are mentioned in the table 1,

Table 1. Optimal Results

Phase	Task	Processor	Phasewise Optimal Cost
1	$t_2 * t_4$	p_1	10
	$t_3 * t_1$	p_2	
2	$t_1 * t_4$	p_1	18
	$t_3 * t_2$	p_2	
3	$t_1 * t_3$	p_1	19
	$t_4 * t_2$	p_2	
4	$t_3 * t_4$	p_1	16
	$t_2 * t_1$	p_2	
5	$t_3 * t_4$	p_1	15
	$t_2 * t_1$	p_2	

The overall optimal cost is thus calculated to be 215 units for this example. It also recorded that the suggested algorithm provides the better results and complexity as compared to earlier suggested algorithms of similar types. The time complexity of the present algorithm is observed to be O(kmn) which is much less as compared to [11] and [14] algorithms.

Table 2. Time Complexity

Processors	Tasks	Phases	Time Complexity		
n	m	k	[11] algorithm O[k(5mn-n^2)]	[14] algorithm O(m^2n^2k)	Present algorithm O(kmn)
3	4	3	153	432	36
3	5	4	264	900	60
3	6	5	405	1620	90
3	7	6	576	2646	126
3	8	7	777	4032	168
4	5	3	252	1200	60
4	6	4	416	2304	96
4	7	5	620	3920	140
4	8	6	864	6144	192
4	9	7	1148	9072	252
5	6	3	375	2700	90
5	7	4	600	4900	140
5	8	5	875	8000	200
5	9	6	1200	12150	270
5	10	7	1575	17500	350

From the above table 2, it is clear that present algorithm is much better for optimal allocation of tasks that upgrade the performance of distributed network. Following fig 1, 2 & 3 also shows the, time complexity comparison for the different values of n (i.e. 3, 4, 5) with the algorithms [11], [14] and present algorithm.

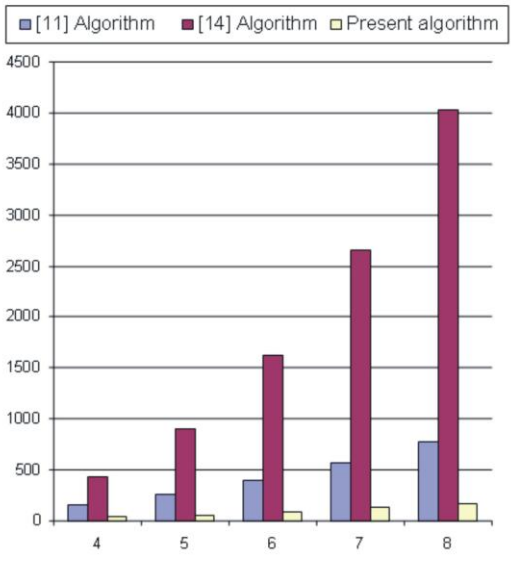

Fig 1: Comparison Graph for n=3

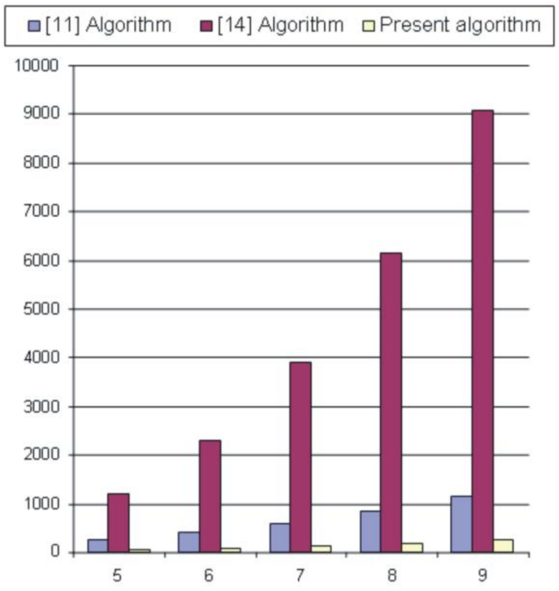

Fig 2: Comparison Graph for n=4

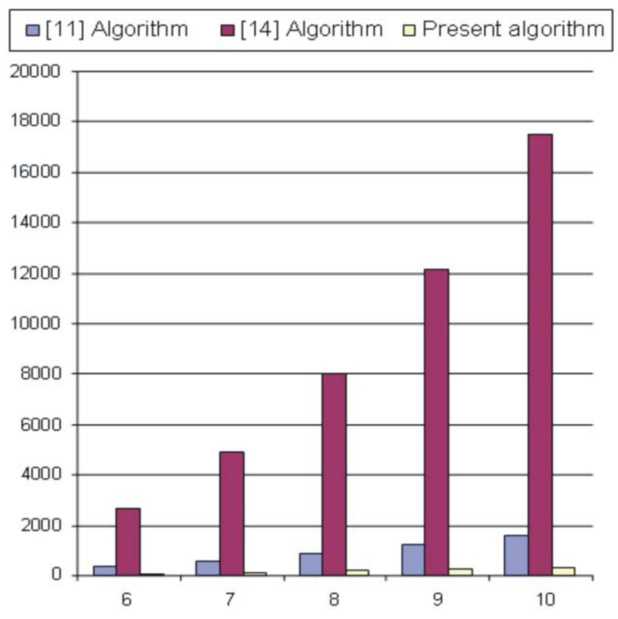

Fig 3: Comparison Graph for n=5

7. REFERENCES

[1] Dr. Kapil Govil and Dr. Avanish Kumar. 2011. A modified and efficient algorithm for Static task assignment in Distributed Processing Environment. International Journal of Computer Applications, Vol. 23, Number 8, Article 1, 1 – 5, ISBN: 978-93-80752-82-3, ISSN: 0975 – 8887.

[2] Kok Fu Ng, Norhashidah Hj. Mohd Ali. 2008. Performance analysis of explicit group parallel algorithms for distributed memory multicomputer. Elsevier Inc. Vol. 34, Issue 6,7,8. 427 – 440.

[3] Kumar, V. Singh, M. P. and Yadav, P.K. 1995. An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System, In proceedings of the 19th National system conference, SSI, Coimbatore, 82 – 87.

[4] Kumar, V. Singh, M.P. and Yadav, P.K. 1995. A Fast Algorithm for Allocating Tasks in Distributed Processing System, In proceedings of the 30th Annual Convention of CSI, Hyderabad, 347 – 358.

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

[6] J. Sum, J. Wu, and C. S. Leung. 2007. On profit density based greedy algorithm for a resource allocation problem in web services. International Journal of Computers and Applications.

[7] Suresh Behara, Sanjay Mittal. 2009. Parallel finite element computation of incompressible flows. Elsevier Inc. Vol. 35, Issue 4, 195 – 212.

[8] Ucar, Bora, Aykanat, Cevdet, Kaya, Kamer and Ikinci, Murat. 2005. Task assignment in heterogeneous computing systems. Journal of Parallel and Distributed Computing, Elsevier Inc., Vol. 66, Issue 1, 32 – 46.

[9] Wei – Ming Lin. 2008. Performance modeling and analysis of correlated parallel computations. Elsevier Inc. Vol. 34, Issue 9, 521 – 538.

[10] N. Beaumont. 2009. Using dynamic programming to determine an optimal strategy in a contract bridge tournament. Journal of the Operational Research Society.

[11] Kumar, Avanish, 1999. Optimizing for the Dynamic Task Allocation, in proceedings of the 'III Conference of the International Academy of Physical Sciences, 1999 Allahabad, 281 – 294.

[12] Palmer, J. and Mitrani, I. 2005. Optimal and heuristic policies for dynamic server allocation. Journal of Parallel and Distributed Computing, Vol. 65, Issue 10, 1204 – 1211.

[13] M. Vanneschi, L. Veraldi. 2007. Dynamicity in distributed applications: issues, problems and the ASSIST approach. Elsevier Inc. Vol. 33, Issue 12, 822 – 845.

[14] Pradeep Kumar Yadav, M. P. Singh and Harendra Kumar. 2008. Scheduling Algorithm: Tasks scheduling Algorithm for Multiple Processors with Dynamic Reassignment. Journal of Computer Systems, Networks and Communications, Vol. 2008, Article ID 578180, 9 pages.

[15] C Alves and J M Valerio de Carvalho. 2008. New integer programming formulations and an exact algorithm for the ordered cutting stock problem. Journal of the Operational Research Society. Vol. 59, 1520 – 1531.

[16] Baz D. El, and Elkihel M. 2005. Load balancing methods and parallel dynamic programming algorithm using dominance technique applied to the 0 – 1 knapsack problem, Elsevier Inc., Vol. 65, Issue 1, 74 – 84.

[17] Iqbal, Saeed and Carey, Graham F. 2005. Performance analysis of dynamic load balancing algorithms with variable number of processors. Journal of Parallel and Distributed Computing, Elsevier Inc., Vol. 65, Issue 8, 934 – 948.

[18] Bahi, Jacques, Couturier, Raphael and Vernier, Flavien. 2005. Synchronous distributed load balancing on dynamic networks, Journal of Parallel and Distributed Computing, Elsevier Inc., Vol. 65, Issue 11, 1397 – 1405.

[19] Jan, Gene Eu and Lin, Ming – Bo. 2005. Concentration, load balancing, partial permutation routing, and super concentration on cube – connected cycles parallel computers. Journal of Parallel and Distributed Computing, Elsevier Inc., Vol. 65, Issue 12, 1471 – 1482.

[20] C. Muller, M. Strengert, T. Ertl. 2007. Adaptive load balancing for raycasting of non-uniformly bricked volumes. Elsevier Inc. Vol. 33, Issue 6, 406 – 419.

[21] Wong, Han Min, Bharadwaj, Veeravalli and Gerassimos, Barlas. 2005. Design and performance evaluation of load distribution strategies for multiple divisible loads on heterogeneous linear daisy chain networks, Elsevier Inc., Vol.65, No.12, 1558-1577.

[22] Zeng, Zeng and Bharadwaj, Veeravalli. 2006. Distributed scheduling strategy for divisible loads on arbitrarily configured distributed networks using load balancing via virtual routing. Journal of Parallel and Distributed Computing, Elsevier Inc. Vol. 66, Issue 11, 1404 – 1418.

[23] Grosu, Daniel and Chronopoulos, Anthony T. 2005. Noncooperative load balancing in distributed systems. Journal of Parallel and Distributed Computing, Elsevier Inc., Vol. 65, Issue 9, 1022 – 1034.

[24] Yeon – Koo Che, Kathryn E. Spier. 2007. Exploiting Plaintiffs Through Settlement: Divide and Conquer. Journal of Institutional and Theoretical Economics (JITE), Vol. 164, Issue I, 4 – 23.

[25] Yi – mu Ji and Ru – chuan Wang. 2006. A Solution of Grid Computing Flow Using MDA Methodology. The Journal of China Universities of Posts and Telecommunications. Vol. 13, Issue 1, 29 – 33.

[26] Giovanni Righini. 2008. A branch – and – bound algorithm for the linear ordering problem with cumulative costs. European Journal of Operational Research Vol. 186, Issue 3, 965 – 971.

[27] Yanai Shuzo, Fujie Tetsuya. 2005. An Improved Branch – and – Bound Algorithm for a Two-machine Flowshop Problem with Minimum Makespan. Journal of Japan Industrial Management Association. Vol. 56, Issue 4, 284 – 293.