An Optimal Task Allocation Model through Clustering with Inter-Processor Distances in Heterogeneous Distributed Computing Systems

Manisha Sharma, Harendra Kumar, Deepak Garg

Abstract*—Distributed computing systems (DCS) are of current interest due to the advancement of microprocessor technology and computers networks. It consists of multiple computing nodes that communicate with each other by message passing mechanism. Reliability and communication over distances are the main reasons for building the DCS. In distributed computing systems, partitioning of applications software in to modules and proper allocation of modules among processors are important factors for efficient utilization of resources. We consider the problem of m-modules and n-processors (m >> n). In this paper a mathematical model for finding optimal cost and optimal reliability to the problem is presented considering DCS with heterogeneous processors in such a way that the allocated load on each processor is balanced. The results obtained by the present model are compared with the recent models and comparison results show that the model is very effective.

Index Terms*—Distributed computing system, Module allocation, Inter module communication, Reliability, Data transfer rate, Inter processor distance.

I. INTRODUCTION

Although improvements in device technology have allowed computer speed to increase by several orders of magnitude in recent decades. The computing capabilities demanded by a number of real world applications have increased at an even faster pace. The required processing power for these applications can't be achieved with a single processor. One approach to this problem is to use distributed computing systems (DCS) that concurrently process an application program by employing multiple processors. In DCS homogeneous or heterogeneous processors are connected together through a communication network. Distributed computing provides the capability for the utilization of remote computing resources and allow for increased levels of flexibility, reliability and modularity. If DCS are properly designed and planned they can provide a more economical and reliable approach than that of centralized processing system.

A 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. Reliability is defined in terms of the time interval in contrast to an instance in time defined in availability. A highly reliable system is one that will continue working for a long period of time. The often advocated advantage of the DCS, in comparison to the centralized system, is the reliability due to the existence of multiple resources. However, only the multiple instances of resources cannot increase the reliability of the DCS, rather the various processes of the distributed operating system (viz. memory manager, task scheduler etc.) must be designed properly to increase the reliability by extracting the characteristic features of the DCS.

The module allocation in a distributed processing system finds extensive application in the faculties where large amount of data is to be processed in a 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 module allocation systems. In such applications the quality of the output is proportional to the amount of real life 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 systems means that ensuring system reliability is of critical importance inherently. DCS is more complex, therefore it is very difficult to predict the performance of DCS. Mathematical modeling is the tool which can play an important role to predict the performance of DCS. Therefore there is an urgent need to develop a method for it. The performance of DCS may suffer if the mapping of distributed applications to processors is not carefully implemented. In order to make best use of the resources, it becomes essential to maximize the overall throughput by allocating the modules to processors in such a way that the allocated load on all the processors should be balanced and to minimize

Manuscript Received July 09, 2011.

Manisha Sharma, Department of Mathematics, Panjab University, Chandigarh- 160014, Punjab (INDIA)
Email id: manishatewaripu@gmail.com

Harendra Kumar, Department of Mathematics & Statistics, Gurkula Kangari University, Hardwar-249404, U.K. (INDIA)
Email id: balyan.kumar@gmail.com

Deepak Garg, Department of Mathematics, Panjab University, Chandigarh- 160014, Punjab (INDIA)
Email id: dgmaths@rediffmail.com

the inter module communication by assigning modules to same processor as much as possible. Which are both contrary to each other as an increase in the number of processors may actually decrease the total throughput of the system. This degradation effect is known as saturation effect, which occurs due to heavy communication traffic induced by data transfers between modules that reside on separate processors. Allocation of modules in a DCS may be done by two ways: *static allocation* and *dynamic allocation.* In static allocation once a module is assigned to a processor, it remains there statically during the execution of a distributed program [1-19]. While in dynamic allocation a module can be reassigned during program execution [20-21]. Several methods owing the use of integer programming [1-4], critical delay consideration [5], branch and bound technique [6], reliability evaluation [7-19] have been reported in the literature.

This paper deals with the module allocation problem having multiple objectives such as: *minimization of system cost*, *maximization of system reliability* and *load balancing* for proper utilization of the processor's capacity. The model used the mathematical programming technique for allocations of modules statically considering the effect of inter processor distances. Several sets of input data are used to test the effectiveness and efficiency of model. It is found that the model is suitable for arbitrary number of processors with the random program structure.

The rest paper is organized as follows. Module assignment problem and definitions are defined first in section 2. In section 3, modules allocation model has been discussed. The Section 4 shows the experimental results in comparison to other scheduling methods and concludes the paper.

II. MODULE ASSIGNMENT PROBLEM AND DEFINITIONS

The specific problem being addressed here is as follows: We considered a distributed system consisting of a set $P = \{P_1, P_2, \ldots P_n\}$ of 'n' heterogeneous processing nodes (e.g. host computer complexes in a computer network, individual processors in a multiprocessing environment) and an application program consisting of a set $M = \{M_1, M_2, \ldots M_m\}$ of 'm' communicating modules to be executed on these processing nodes.

An allocation of modules to processors is defined by a function A from the set M of modules to the set P of processors such that:

$$A: M \rightarrow P \ ,$$

Where $A(i)=j$ if module M_i is assigned to processor P_j, $1 \le i \le m$, $1 \le j \le n$. **(1)**

In general the objective of modules allocation is to find an optimal allocation A_O, which minimize the completion time of the program and optimize the system reliability by properly mapping the modules to the processors. In order to make the best use of the resources in a distributed computing system we would like to distribute the load on each processor in such a way that allocated load on the processors are balanced. The following terms, notations and assumptions will be used throughout the text.

2.1. System cost functions

2.1.1 Execution cost (EC): The execution cost e_{ij} ($1 \le i \le m$, $1 \le j \le n$) of each module M_i depends on the capability of the processor P_j to which it is assigned and the work to be performed by each module. The execution cost of each module on all the processors is taken in the form of execution cost matrix, ECM= $[e_{i,j}]$ of order $m \times n$. The execution cost of each processor for an allocation A, is calculated using equation (2) as:

$$PEC(j) = \sum_{\substack{1 \le i \le m \\ i \in MS_j}} e_{i,A(i)} \qquad \textbf{(2)}$$

Where $MS_j = \{i: A(i) = j, \quad j = 1, 2 \ldots n\}$

Overall execution cost of a given allocation A, is calculated as:

$$EC(A) = \sum_{1 \le i \le m} e_{i,A(i)} \qquad \textbf{(3)}$$

2.1.2 Inter module communication cost (IMCC): In this paper, the matrix DT= $[data_{i,j}]$ of order m x m is used to represent the data communication between the modules during the execution, where $data_{i,j}$ is the amount of data required to be transmitted from Module M_i to module M_j. The data transfer rates, i.e. bandwidth size, (in bytes/seconds) between processors are stored in a matrix DTR= $[r_{i,j}]$ of size n x n. The communication startup costs of processors are given in a n-dimensional vector L= $[L_j]$. The IMCC between modules M_i and M_j is defined by:

$$c_{i,j} = \begin{cases} \bar{L} + \frac{data_{i,j}}{\overline{DTR}}, & if \quad i \neq j \\ 0 \qquad , & if \quad i = j \end{cases} \qquad \textbf{(4)}$$

Where $\overline{DTR}$, is the average transfer rate among the processors in the processors domain, and $\bar{L}$ is the average communication startup time. The communication between the modules are taken in form of inter module communication cost matrix, IMCCM = $[c_{i,j}]$ of order $m \times m$.

Let us denote $d_{x,y}$ the distance between the processors P_x and P_y. The processors P_x and P_y are connected by a communication link say $l_{x,y}$. The inter-processor communication cost per unit of information transferred between two processors P_x and P_y increases linearly as the distances $d_{x,y}$ increases. To consider the impact of inter processor distance on the cost, we define a inter processor distance matrix IPDM= $[d_{i,j}]$. Thus, if two interacting modules M_i and M_j are assigned to two different processors P_k and P_s respectively, then the two modules cause the inter-processor communication cost of $c_{i,j} * d_{k,s}$. We assume that the communication cost between two modules assigned to the same processors is negligible, since all communication is through memory as opposed to an inter-processor link.

The overall inter-module communication cost and inter-module communication cost for each processor for a given allocation A, are obtained by equations (5) and (6) respectively as:

$$TCC(A) = \sum_{\substack{1\le i\le m \\ i+1\le j\le m \\ A(i)\neq A(j)}} c_{i,j} * d_{A(i),A(j)} \quad \textbf{(5)}$$

$$IPCC(j) = \sum_{\substack{1\le i\le m \\ A(i)=j\neq A(k)}} c_{i,k} * d_{A(i),A(k)} \quad \textbf{(6)}$$

2.1.3 Total cost and response time of the System: The total cost, TCOST (A) of the system for an allocation A is computed as:

$$TCOST(A) = EC(A) + TCC(A) \quad \textbf{(7)}$$

Response (turnout) 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 allocation A is defined by the following equation as:

$$RT(A) = \max_{1\le j\le n} \{PEC(j) + IPCC(j)\} \quad \textbf{(8)}$$

Thus the objective function for the response time of the system is expressed as:

$$\text{Min } Z = \min\{RT(A)\}$$

$$= \min\left\{\max_{1\le j\le n}\left[\sum_{i=1}^{m} e_{i,j}\, x_{i,j} + \sum_{i=1}^{m}\sum_{\substack{k=1 \\ l\neq j}}^{m} c_{i,k} * d_{j,l}\, x_{i,j}\, x_{k,l}\right]\right\} \quad \textbf{(9)}$$

Subject to

$$\sum_{j=1}^{n} x_{i,j} = 1 \qquad \text{for all } i=1,2,3,\ldots..m. \quad \textbf{(10)}$$

$$\frac{\sum_{i=1}^{m} e_{i,j} x_{i,j}}{\max_{1\le j\le n}\left\{\sum_{i=1}^{m} e_{i,j} x_{i,j}\right\}} \le 1 \qquad \text{for all } j=1,2,3\ldots\ldots n. \quad \textbf{(11)}$$

where

$$x_{i,j} = \begin{cases} 1, \text{ if i-th module is assigned to j-th processor} \\ 0, \text{ otherwise} \end{cases}$$

2.2 Processor Utilization: The main purpose of module allocation in a DCS is to reduce turnout time the system. This is done by maximizing the utilization of processors while minimizing the communication among processing nodes. While minimizing inter module communication tends to assign the whole modules to a single processing node, load balancing tries to distribute the program modules of a program almost evenly among the processing nodes. The Utilization of each processor is calculated as:

$$PU(j) = PEC(j) / \max\{PEC(j) : j = 1 \text{ to } n\} \quad \textbf{(12)}$$

and the average processor utilization (APU) is calculated as:

$$APU = [PU(1) + PU(2) + \ldots\ldots\ldots\ldots PU(n)]/ n \quad \textbf{(13)}$$

2.3 Processor's Reliability: Reliability is defined in terms of the time interval in contrast to an instance in time defined in availability. A highly reliable system is one that will continue working for a long period of time. In DCS both system cost and system reliability are time dependent. Each component in DCS has two states: operational and failed. Reliability of the DCS is defined as the probability that the modules of the program can run successfully on the components (i.e. processors and communication link) during the time interval [0, t] under an allocation A. Using the Poisson distribution, the execution reliability of the k-th processor under the allocation A is calculated as:

$$ER(P_k) = \exp[-\lambda_k * PEC(k)] \quad \textbf{(14)}$$

Where λ_k is failure rate of the k-th processor P_k .

The communication link reliability for k-th processor, P_k during an allocation A is calculated as:

$$CLR(P_k) = \prod_{j=1}^{m} \exp\left[-\mu_{k,A(j)} * \sum_{\substack{1\le i\le m \\ A(i)=k\neq A(j)}} c_{i,j} * d_{k,A(j)}\right] \quad \textbf{(15)}$$

Where $\mu_{k,A(j)}$ is failure rate of link $l_{k,A(j)}$

The total communication reliability for k-th processor is the product of execution reliability and communication link reliability associated with processor P_k for an allocation A and calculated as:

$$TCR(P_k) = ER(P_k) * CLR(P_k) \quad \textbf{(16)}$$

2.4 Assumptions: To keep the algorithm reasonable in size several assumptions have been made while designing the algorithm. These assumptions are as:

(1) A program is assumed to be collection of m- modules to be executed on a set of n-processors having different processing speeds and failure rates.

(2) A module may be portion of an executable code or data file. The number of modules to be allocated is more than the number of processors (m >> n), as normally in the case of real life.

(3) A module may take different execution time if it executes on different processor and amount of data may take different communication time if it is transmitted through the different communication links. Therefore the system cost and system reliability both depend upon the execution and communication costs. If a module is not executable on any of the processor due to absence of some resources, the execution cost of that module on that processor is taken to be infinite.

(4) We assume that once a module has completed its execution on a processor, the processor stores the output data of the module in its local memory. If the data is needed by some other module being computed on the same processor, it reads the data from the local memory.

(5) Whenever a group of modules is assigned to the same processors, the inter module communication cost between them is zero.

3. Scheduling Algorithm

In this section, we propose a heuristic modules algorithm for solving a load balancing modules allocation problem. The model addressed in this paper differs from previous work on module allocation as it finds:

(a) Balanced allocated load on each processor.
(b) Minimum response time of the system.
(c) Minimum total communication cost.
(d) Maximum reliability for the system.

The module scheduling scheme that we are presenting here has following major steps:

1) Calculation of inter-module communication costs.
2) Cluster making
3) Scheduling of the clusters to the processors.
4) Calculation of the costs and reliabilities.

Inter-processor cost (IPC) is incurred due to the inter-module communication when the data are transferred from module to module if they are residing on different processors. Therefore IPC is proportional to IMCC. IMCC is a function of data communication between the modules, data transfer rates between the processors and communication startup costs of processors and calculated as:

$$c_{i,j} = \begin{cases} \bar{L} + \frac{data_{i,j}}{\overline{DTR}}, & if \quad i \neq j \\ 0, & if \quad i = j \end{cases}$$

Modules are clustered based on their communication requirement. Highly communicating modules are clustered together to reduce communication delays. Usually number of module clusters should be equal to the number of processors so that one to one mapping may result. These clusters will be fixed throughout their execution. Since we have 'n' number of processors in DCS, therefore we will make 'n' number of module clusters. If a cluster resulting from combining modules becomes too large, it would be impossible to obtain a load-balanced scheduling therefore we restrict the number of modules in a cluster by:

Maximum number of modules in a cluster $\leq t + 1$

where

$$t = \begin{cases} \left(\frac{m}{n} - 1\right) & \text{if} \quad \frac{m}{n} \text{ is an integer} \\ \left\lfloor \frac{m}{n} \right\rfloor & \text{, otherwise} \end{cases}$$

The inter-processor distance $d_{x,y}$ between two processors p_x and p_y play an important role in total cost of the system as inter-processor communication cost varies linearly as distance $d_{x,y}$ increases. We are taking this effect into account and map the module clusters to the processors. One concern in modules cluster mapping is that each processor has a roughly equal workload. For this we are changing cluster mapping to computing processor dynamically during the computation. Tie –breaking in selection of the processor is done randomly; i.e. one of the modules with equal cost is selected at random. There can be alternative policies for tie-breaking but this kind of policy increases the time complexity of the algorithm, so we prefer a random selection strategy. The procedure of clustering the modules and their scheduling to the processors is described by the algorithm as:

Algorithm

1.(a) Input: DT=[$data_{i,j}$], ECM=[$e_{i,j}$], DTR=[$r_{i,j}$], IPDM= [$d_{i,j}$], L=[L_j], λ_j, CFRM=[$\mu_{j,l}$].

(b) Compute $\bar{L} = \frac{\sum_{1 \leq j \leq n} L_j}{n}$ ← Average Communication Startup Time of the Processors

(c) Compute $\overline{DTR} = \frac{\sum_{1 \leq i \leq n} \sum_{1 \leq j \leq n} r_{i,j}}{n(n-1)}$ ← Average Data Transfer Rate.

(d) Compute IMCCM= [c_{ij}].

2. (a) Compute t.

(b) Compute $S_i = \sum_{1 \leq j \leq m} c_{i,j}$ for i=1 to m.

(c) Find maximum of S_i, if maximum of S_i is for i=k, $1 \leq k \leq m$, then select t largest entries from kth row of IMCCM= [c_{ij}] say { c_{kl}, c_{km}, …….. }.

(d) Make a cluster C_1 of t+1 modules { M_k, M_l, M_m, …..}.

3. (a) Calculate new execution cost matrix $NECM = [\bar{e}_{i,j}]$ by adding rows corresponding to the modules present in the cluster C_1={M_k , M_l , M_m , ⋯ }.

(b) Calculate new inter module communication cost matrix $NIMCCM = [\bar{c}_{i,j}]$ by adding rows and columns of modules present in cluster C_1 = {M_k , M_l , M_m , ⋯ }.

(c) Assign the cluster C_1 to that processor P_j for which the cost is minimum say for j= r. If there is tie in selection, do it randomly.

4. Compute Cost (C_1) = Cost {M_k, M_l , M_m , ⋯ } = $\bar{e}_{k,r}$

5. (a) Select the k'th column $[\bar{c}_{i,k}]^T$ of $NIMCCM = [\bar{c}_{i,j}]$.

(b) Calculate $d_{r,j} * [\bar{c}_{i,k}]^T = [d_{r,j} * \bar{c}_{i,k}]^T$.

6. (a) Modify the cost of $NECM = [\bar{e}_{i,j}]$ by adding the column $[d_{r,j} * \bar{c}_{i,k}]^T$ to j-th column of $NECM = [\bar{e}_{i,j}]$ for j=1 to n.

(b) Reduce $NECM = [\bar{e}_{i,j}]$ and $NIMCCM = [\bar{c}_{i,j}]$ by removing rows and columns of C_1.

7. (a) Repeat step 2 to 3 for reduced NECM and NIMCCM to make the next cluster .

(b) Assign the next cluster to that processor P_j for which the cost is minimum say for j= l and l≠r.

8. Repeat step **4** to step **7**.

9. If

n clusters are assigned to n processors
then go to step **10**
else go to step **8**.

10. Compute:

(a) TCOST(j) = PEC(j) + IPCC(j)

(b) $PU(j) = PEC(j) / \max\{PEC(j) : j = 1 \text{ to } n\}$

(c) $TCR(P_k) = ER(P_k) * CLR(P_k)$

11. Compute:

(a) $RT(A) = \max_{1 \le j \le n} \{PEC(j) + IPCC(j)\}$

(b) TCOST(A) = EC(A) + TCC(A)

(c) APU= [PU (1) + PU (2) + ……PU (n)]/ n

(d) Compute total reliability = $\prod_{k=1}^{n} TCR(P_k)$

12. End.

III. EXPERIMENTAL RESULTS AND CONCLUSION

Load balancing modules allocation model for heterogeneous processors was discussed in the previous section. To evaluate our proposed model we use a different size of problem in this section. In experimental problem we have consider a distributed program made up of five modules **{**M_1, M_2, M_3, M_4, M_5} to be executed on a set of three heterogeneous processors {P_1, P_2, P_3}. The failure rates and communication start up time of processors P_1, P_2, P_3 are .00002, .00001, .00003 units and 3, 2, 4 units respectively. The various matrices taken as input are described in tables-1 to table-4.

ECM=[$e_{i,j}$]			
Processors → Modules ↓	P_1	P_2	P_3
M_1	2	5	7
M_2	3	2	4
M_3	10	1	7
M_4	2	3	8
M_5	3	6	2

Table -1

DT=[$data_{i,j}$]					
Modules → Modules↓	M_1	M_2	M_3	M_4	M_5
M_1	0	2	3	1	4
M_2	2	0	5	6	1
M_3	3	5	0	2	5
M_4	1	6	2	0	7
M_5	4	1	5	7	0

Table-2

	IPDM=[$d_{i,j}$]			DTR=[$r_{i,j}$]		
	P_1	P_2	P_3	P_1	P_2	P_3
P_1	0	1.5	2	-	0.5	0.7
P_2	1.5	0	1	0.5	-	0.3
P_3	2	1	0	0.7	0.3	-

Table-3

	CFRM=[$\mu_{i,l}$]		
	P_1	P_2	P_3
P_1	-	0.00001	0.00003
P_2	0.00001	-	0.00002
P_3	0.00003	0.00002	-

Table-4

Here $\overline{DTR}$ = 0.5, $\overline{L}$ = 3 and inter module communication costs are as:

	IMCCM=[c_{ij}]				
	M_1	M_2	M_3	M_4	M_5
M_1	0	7	9	5	11
M_2	7	0	13	15	5
M_3	9	13	0	7	13
M_4	5	15	7	0	17
M_5	11	5	13	17	0

Table-5 Inter Module Communication Cost Matrix

Three clusters of modules are formed as:

Cluster_C_1 = {M_4, M_5}.
Cluster_C_2 = {M_2, M_3}.
Cluster_C_3= {M_1}.

Tables-6 and 7 shows the optimal costs and optimal reliability of the system for the optimal assignment of module clusters to the processors, where Cluster_C_1 → P_1, Cluster_C_2 → P_2 and Cluster_C_3 → P_3. The optimal total cost of the program, program reliability, response time of the system and average processor utilization are 123, 0.9956, 97 and 0.71 respectively

Modules	Processors	PEC(j)	IPCC(j)	PEC(j)+ IPCC(j)
M_4, M_5	P_1	5	92	97
M_2, M_3	P_2	3	76	**79**
M_1	P_3	7	48	55

Table-6 Optimal Costs of the System

Processors	ER(P_k)	CLR(P_k)	TCR(P_k)	PU(j)
P_1	0.9999	0.9984	0.9983	0.71
P_2	0.9999	0.9990	0.9989	0.42
P_3	0.9997	0.9987	0.9984	1

Table-7 Optimal Reliability of the System

For evaluating the performance of our algorithm a large number of programs were considered. With these applications, simulation experiments were carried out. The execution costs of

the modules were picked randomly in the interval from ***5*** to 100, and inter-module communication costs were chosen in such a way that the range lies from 1 to **20,** that is execution costs were selected approximately 2-30 times larger than the inter-module communication costs. The numbers of modules in the examples were restricted to twenty. These programs were tested on yadav et.al. [19] and present model. It is found that in 78% cases the present model gives better results as comparative to [19], in 17% cases the obtained results are equal and there are only 5% cases where the present model gives worse results as comparative to yadav et.al. [19].The algorithm suggested in [6] is not considered the criteria of load balancing and proper utilization of each processor whereas our model considered both the issues. The run time complexity of the algorithm suggested by R.Y. Richard et al [6] is O (mn) which to very high. The algorithm suggested by Sagar et al. [14] has run time complexity $O(m^2n)$. The run time complexity of the algorithm presented in this paper is O (m^2+mn) which is less than the complexities of the models [6] & [14].

REFERENCES

[1] Ghafoor and J. Yang, "A Distributed Heterogeneous Supercomputing Management System", IEEE Comput., 1993Vol.6, pp. 78-86 .

[2] D.F.Towsley, "Allocating Programs Containing Branches and Loops within a Multiple Processor System",IEEE Trans. Software Eng. SE-12,10, 1986, pp.1018-1024.

[3] Chu W.W., "Optimal File Allocation in a Multiple Computing System", IEEE Trans. on Computer, Vol. C-18, 1969, pp.885-889.

[4] Dessoukiu-EI O.I. and Huna W.H., "Distributed Enumeration on Network Computers", IEEE Trans. on Computer, Vol. C-29, 1980,pp. 818-825.

[5] J.B.Sinclayer, "Optimal Assignment in Broadcast Network", IEEE Trans. on Computer, Vol.37 (5), 1988, pp.521-531.

[6] Richard, R.Y., Lee, E.Y.S. and Tsuchiya M., " A Task Allocation Model for Distributed Computer System", IEEE Trans. on Computer,Vol.C-31, 1982,pp.41-47.

[7] Min-Sheng Lin, "A Linear-Time Algorithm for Computing K-Terminal Reliability on Proper Interval Graphs",IEEE Trans.Reliability, Vol.51, 2002, pp.58-62.

[8] Baca, D.F., "Allocation Modules to Processors in a Distributed System", IEEE Trans. on Software Engineering, Vol.15, 1989, pp.1427-1436 .

[9] D. Fernindez- Baca, "Allocating Modules to Processors in a Distributed System", IEEE Trans. Software Eng. SE-15, 11, 1989, pp. 1427-1436.

[10] Kumar, A.. "An Algorithm for Optimal Index to Task Allocation Based on Reliability and Cost", published in the proceedings of International Conference on Mathematical Modeling held at Roorkee, 2001, pp.150-155.

[11] Kumar, V. Singh, M.P. and Yadav, P.K., "An Efficient Algorithm for Allocating Tasks to Processors in a Distributed Systems", Proc. of the 19th National System Conference, SSI, Held at Combatore, India, 1995, pp.82-87 .

[12] Kumar, V., Singh, M.P. and Yadav, P.K., " A Fast Algorithm for Allocating Tasks in Distributed Processing System", Proc. of the 30th Annual Convention of CSI held at Hyderabad, India, 1995, pp.347-358.

[13] Peng, D.T.,Shin, K.G.and Abdel, Z. T.F., "Assignment Scheduling Communication Periodic Tasks in Distributed Real Time System", IEEE Trans. on Software Engg. Vol.SE-13, 1997, pp.745-757 .

[14] Sagar,G., Sarje, A.K., " Task Allocation Model for Distributed System", Int. J. System Science,Vol.22, 1991,.pp.1671-1678 .

[15] Singh, M.P., Kumar, V., Kumar, A., "An Efficient Algorithm for Optimizing Reliability Index in Tasks-Allocation", Acta Ciencia Indica, Vol.XXVM, 1999, pp. 437-444.

[16] Srinivasan, S., Jha. K.N.,"Safety and Reliability Driven Task Allocation in Distributed Systems", IEEE Trans. on Parallel and Distributed System, Vol.10, 1999, pp. 238-250.

[17] Yadav, P.K., Kumar, A . ,"An Efficient Static Approach for Allocation through Reliability Optimization in Distributed Systems", Presented at the International Conference on Operations Research for Development (ICORD 2002) held at Chennai.

[18] Zahedi, E., Ashrafi, N., " Software Reliability Allocation based on structure, Utility, Price and Cost", IEEE Trans. on Software Engineering, Vol.-17, 1991, pp. 345-356 .

[19] Yadav P.K., Singh M.P., Sharma K., "Tasks Allocation Model for Reliability and Cost Optimization in Distributed Computing System, International Journal Of Modeling, Simulation, and Scientific Computing, Vol.2, No.2, 2011, pp.131-149.

[20] Yadav P.K., Singh M.P., Kumar H., "Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Reassignment", Journal of Computer Systems, Networks and Communications, Article ID-578180, 2008, pp.1-9.

[21] Bokhari, S.H., "Dual Processor Scheduling with Dynamic Re-Assignment", IEEE Transactions on Software Engineering, vol. 5, 1979, pp. 341-349