Performance Intensification of DRTS under Static Load Sharing Scheme

Urmani Kaushal
Asst. Professor
Department of Physical & Computer Science
FASC-MITS Deemed University
Lakshmangarh-332311, Sikar, Rajasthan, INDIA

Avanish Kumar
Professor & Head
Department of Math., Stat. & Computer Applications
Bundelkhand University, Jhansi, INDIA

ABSTRACT
Distributed Real Time System (DRTS) provides enormous platform for parallel applications. It is an alternative for highly expensive parallel machines. Task allocation for parallel applications over it, is a crucial phase where strategy for task allocation should be chosen to maximize the throughput and enhance the overall processor utilization. Task allocation is NP-hard or NP-complete problem. A new heuristic for this problem has been suggested and implemented in this paper. To achieve this goal, tasks should be clustered in such a way that it can minimize the inter-task communication cost as well as it must also be taken care that the execution cost of tasks must also be minimum over the processor where the tasks are going to get assigned. Here *k-mean* clustering has been used to cluster the tasks in the required number of clusters. The proposed model has been simutated in matlab.

General Terms
Distributed Real Time System, Task Allocation, Cluster.

Keywords
DRTS, parallel application, throughput, processer utilization, NP-complete.

1. INTRODUCTION
The availability of inexpensive high performance processors and memory chips has made it attractive to use Distributed Computing Systems for real time applications [1]. Distributed Real-Time Systems are characterized by their requirement that the execution of their computational tasks must be not only logically correct but also completed within specified time span [2]. The term "Distributed Real-Time System" [DRTS] is used to describe whenever there are several computers interconnected in some fashion so that a program or procedure running system terms with multiple processors. In a recently published literature a DRTS has been defined as a system consisting of many heterogeneous processors with different processing capabilities connected by two way communication links and having their own resources/buffers [3].

DRTS have come out as a great platform for high performance parallel applications. It has formed an alternative to the highly expensive massively parallel machines. The execution of a parallel application can be seen as the execution of multiple tasks (*parallel application divided into number of tasks*) over different processors in the system concurrently. The performance of parallel applications on distributed real time system basically depends on the arrangement of the tasks on various processors available in the system, which is known as *assignment problem* [4, 5, 6, 7]. In performance analysis of distributed real-time system systematic allocation of tasks plays an important.If the allocation is not carefully implemented, processors in the system may spend most of their time waiting for each other instead of performing useful computations [8].

Task allocation is a NP- hard or NP-complete problem [4-6,9-12]. Many task allocation algorithms for distributed computing systems have been proposed in the literature [4-7, 9, 11-12, 14]. A task allocation algorithm seeks an assignment that optimizes a certain cost function, for example maximum throughput or minimum turnaround time. In general, optimal solutions can be foundthrough an exhaustive search, but as there is n^m ways in which m modules can be assigned to n processing nodes. An exhaustive search is often not possible. Thus optimal solution algorithmsexist only for restricted cases or very small problems [5].

Certainly, for optimal utilization of the resources of a DRTS, it is necessary and sufficient to prevent the nodes from being idle rather than balancing the load over distributed system. This adaptation is often called load sharing. If assignment has been made on the basis of the state of the information of the distributed system, is called static load sharing. Once the state information is in hand, the assignments can be made by using appropriate strategy to achieve optimality.In a DRTS, 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 execution time of a particular module on a particular node will depend on the number of modules already executing on that particular node [8, 10, 12, 15].

2. PROBLEM FORMULATION
In a Distributed Real-Time System, 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 problem 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 DRTS with the goal to maximize the overall throughput of the system. The objective of this problem is to enhance the performance of the distributed system by making optimal utilization of its processors and suitable allocation of tasks.

2.1 Notations
T : the set of tasks of a parallel program to be executed.
P : the set of processors in DRTS.
n : the number of processors.
m: the number of tasks formed by parallel application .
k: the number of clusters.
t_i :i^{th} task of the given program.
P_l:l^{th} processor in P.

ec_{il} : incurred execution cost (EC), if i^{th} task is executed on l^{th} processor.
cc_{ij}: incurred inter task communication cost between task t_i and t_j , if they are executed on separate processors.
CI : cluster information vector.
ECM (,) : execution cost matrix.
ITCCM (,) : inter task communication cost matrix.

2.2 Definitions

2.3.1 Execution Cost (EC)

The execution cost ec_{il} of a task t_i , running on a processor P_l is the amount of the total cost needed for the execution of t_i on that processor during process execution. If a task is not executable on a particular processor, the corresponding execution cost is taken to be infinite (∞).

2.3.2 Communication Cost (CC)

The communication cost (cc_{ij}) incurred due to the inter task communication is the amount of total cost needed for exchanging data between $t_i$and t_j residing at separateprocessor during the execution process. If two tasks executed on the same processor then $cc_{ij} = 0$.

2.3 Assumptions

To allocate the tasks of a parallel program to processors in DRTS, we have been made the following assumptions:

2.3.1 The processors involved in the DRTS are heterogeneous and do not have any particular interconnection structure.

2.3.2The parallel program is assumed to be the collection of *m*- tasks that are free in general, which are to be executed on a set of *n*- processors having different processor attributes.

2.3.3 Once the tasks are allocated to the processors they reside on those processors until the execution of the program has completed. At whatever time a cluster of tasks is assigned to the processor, the inter task communication cost (ITCC) between those taskswill be zero.

2.3.4 Total number of clusters is equal to total number of processors.

2.3.5 Data points for *k-mean clustering* will be collection of vectors which represents the execution cost of the task t_m on each processor.

2.3.6 Number of tasks to be allocated is more than the number of processors (m>>n) as in real life situation.

2.4 Problem statement

In order to evaluate the overall optimal processing cost of a DRTS, we have chosen the problem where a set P = {p_1, p_2, p_3, …….p_n} of '*n*' processors and a set T = {*t1*, t_2, t_3, ……t_m} of '*m*' tasks. The processing time of each task to each and every processor is known and it is mentioned in the Execution Cost Matrix ECM (,) of order *m x n*. The communication cost of task is also known and is mentioned in Inter Task Communication Cost Matrix ITCCM (,) of order *m x m*.
Once we have the information about the DRTS i.e. the number of processors in the system, the number of cluster can be determined. We are having m number of tasks which are to be allocated on n processors and the *m>n*, so *m* tasks should be clustered into *k* clusters which equal to n in the case. Here we have *m* data points in the form of task vectors to be clustered in *k* clusters. Cluster information will be stored in CI (,) vector of *m x 1*.

According to the cluster information the ECM (,) and ITCCM (,) will be reformed. Afterward the algorithm proposed will be applied accordingly to compute total optimal cost for parallel applications.

2.5 Proposed Mathematical Model for Task Allocation

In this section, we have developed a task allocation model to get an optimal system cost so that the system performance could be enhanced. We can achieve this objective by making task allocation properly. Therefore, an efficient task allocation of parallel application's tasks to processor is crucial. However, obtaining an optimal allocation of tasks of parallel application to any arbitrary number of processors interconnected with non-uniform links is a very complex problem.
Hereafter, in order to allocate the tasks of such program to processors in DRTS, we should know the information about the input such as tasks attributes [e.g. execution cost, inter task communication cost etc]. While obtaining such information is beyond the scope of this paper therefore, a deterministic model that the required information is available before the execution of the program is assumed [15]. In the present task allocation model, processor execution cost and task clustering has been considered for this system.

2.5.1 Execution Cost (EC)

The task allocation given as:

$$S: T \rightarrow P,\ S(i) = l \quad \textbf{(1)}$$

The execution cost ec^{il} represents the execution of task t_i on processor P_l and it is used to control the corresponding processor allocation. Therefore, under task allocation S, the execution of all the tasks assigned to l^{th} processor can be computed as:

$$\mathrm{EC(X)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{m}} ec_{il} x_{il} \qquad (2)$$

2.5.2 Task Clustering

Evaluation of cluster compactness as the total distance of each point (task vector of n dimension) of a cluster from the cluster mean which is given by[13], Z_{ki}

$$\sum_{x_i IC_k} \left\| X_i - \overline{X}_k \right\|^2 = \sum_{i=1}^{m} Z_{ki} \left\| X_i - \overline{X}_k \right\|^2 \qquad (3)$$

Where the cluster mean is defined as $\overline{X}_{\mathrm{k}} = \frac{1}{m_{\mathrm{k}}} \sum_{x_i IC_k} X_i$ and $m_{\mathrm{k}} = \sum_{i=1}^{m} \mathrm{Z_{ki}}$ is the total number of points allocated to cluster k. The parameter Z_{ki} is an indicator variable indicating the suitability of the i^{th} data point X_i to be a part of the k^{th} cluster.

The total goodness of the clustering will then be based on the sum of the cluster compactness measures for each of the k clusters. Using the indicator variables Z_{ki} , we can define the overall cluster goodness as:

$$\varepsilon_{\mathrm{k}} = \sum_{i=1}^{m} \sum_{k=1}^{k} Z_{ki} \left\| X_i - \overline{X}_k \right\|^2 \qquad (4)$$

Here $\overline{X}_k$ should be found in such a manner that the value of ε_k can be minimized.

2.5.3 Processor Utilization

The purpose of the proposed model is to reduce turnout time the system which can be achieved by maximizing the utilization of processors while minimizing inter task communication. The utilization of each processor is calculated as:

PU(*l*) = PEC(*l*) /max{PEC(*l*):*l*=1 to n}**(5)**

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

APU= [PU (1) + PU (2) + …….PU (n)]/ n**(6)**

3. PROPOSED TASK ALLOCATION TECHNIQUE AND ALGORITHM

3.1 Technique

The problem addressed in this paper of tasks allocation of a parallel application onto the processors of a DRTS to enhance the performance of the system have 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 Execution Cost Matrix *ECM (,)* of order *m x n*. The communication cost of task is also known and is mentioned in Inter Task Communication Cost Matrix *ITCCM (,)* of order *m x m*.

To enhance the performance of the system the total system cost should be minimized. For the minimization of total system cost we will form the clusters of tasks. We have *m* tasks to be processed over *n* processors *(m>n)*, so *k* clusters should be formed. For clustering we will use *k-mean* clustering algorithm. Here we have *m* vectors of task to be placed in *k* clusters. Find *k* initial points *(number of points in cluster is equal to the number of clusters)* for each cluster represented by task vector that are to be clustered. These points represent initial clusters called centroids. Assign each task vector to the cluster that has the closest centroid. When all task vectors have been assigned, recalculate the positions of *k* centroids [9]. Repeat the work of assignment and recalculation of centroid's positions until the centroids no longer move. This produces a separation of the task vectors into clusters from which the metric to be minimized will be calculated as follows:

$$\sum_{x_i IC_k} \left\| X_i - \overline{X}_k \right\|^2 = \sum_{i=1}^{m} Z_{ki} \left\| X_i - \overline{X}_k \right\|^2 \qquad (7)$$

Modify the *ECM(,)* according the *k* clusters by adding the processing time of those tasks that occurs in the same cluster. Modify the *ITCCM(,)* by putting the communication zero amongst those tasks that are in same cluster. By applying algorithm proposed in [10], we get the optimal assignment as well as Execution Cost and Communication Cost. For optimal assignment of clusters of tasks will be computed as

$$\mathrm{EC(X)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{i=1}}^{\mathrm{m}} ec_{il} x_{il} \qquad (8)$$

$$where, x_{ij} = \begin{cases} 1\ if\ i^{th} task is assigned to j^{th} processor \\ 0, otherwise \end{cases}$$

The objective function to calculate optimal system cost is as follows:

Total Cost =EC + CC**(9)**

3.2 Proposed Algorithm

The algorithm consists of following steps:

Step-0: Read the number of processors in n

Step-1: Read the number of tasks in m

Step-2: Read number of clusters in *k(in this case it equal to number of processors)*

Step-3: Read the Execution Cost Matrix ECM (,) of order m x n

Step-4: Read the Inter Task Communication Cost Matrix ITCCM (,) of order m x m

Step-5: Apply *k-mean* clustering algorithm on ECM (,)

Assign the initial values of means to centroids

Repeat

Assign each task vector to the cluster which has the closest mean

Calculate new mean for each cluster

Until convergence criteria is met

Step-7: Cluster information is stored in vector *CI*

Step-8: Modify the ECM (,) by adding the processing time of tasks in each cluster

Step-9: Modify the ITCCM (,) by putting communication zero amongst those tasks

which are in the same cluster

Step-10: Apply munkres algorithm on ECM (,)

Step-11: Calculate Execution Cost, Inter Task Communication Cost

Step-12: Optimal Cost = Execution Cost + Inter Task Communication Cost

4. Implementation of the Model

An illuminating example has been considered as in [15] to show the performance improvement of the DRTS, as well as to test the proposed algorithm using this data set. It is implemented in Matlab.

Example: In this example, a typical parallel applications of ten tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}\}$ to be executed on the DRTS having four processors $\{P_1, P_2, P_3, P_4\}$ is considered. The execution cost of each task at different processors is given in Table 1. Here, we assume that all the communication links have the same transmission rate and hence we give the ITCT directly in time units as shown in Table 2.

Applying the proposed algorithm to this example, the algorithm traces the following output: The tasks have been clustered into four clusters. Based on this clustering the

modified ECM has been calculated shown in table 3. ITCCM has also been modified and shown in table 4. The processor's load of P_1, P_2, P_3 and P_4 are 8, 1, 9, and 10 respectively. Table 5 shows the optimal assignment of tasks to the processors for the present task allocation model, where $t_2, t_3, t_{10} \rightarrow P_1$; $t_6 \rightarrow P_2$; $t_7, t_8 \rightarrow P_3$, and $t_1, t_4, t_5, t_9 \rightarrow P_4$. The optimal system cost is 48.

Table 1: Execution Cost Matrix

ECM				
Processors	P_1	P_2	P_3	P_4
Tasks				
t_1	6	2	9	3
t_2	5	3	2	1
t_3	8	7	3	4
t_4	1	4	6	3
t_5	4	5	6	2
t_6	2	1	8	9
t_7	2	8	6	7
t_8	6	7	8	9
t_9	4	5	6	2
t_{10}	3	7	4	2

Table 2: Inter Task Communication Cost Matrix

	ITCCM									
Tasks	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
t_1	0	3	2	3	1	2	4	6	7	2
t_2		0	8	2	3	4	1	6	1	2
t_3			0	1	2	3	4	3	5	2
t_4				0	4	6	3	4	2	2
t_5					0	8	3	4	9	2
t_6						0	6	7	5	4
t_7							0	3	2	1
t_8								0	1	6
t_9									0	5
t_{10}										0

Table 3: Modified Execution Cost Matrix

ECM				
Processors	P_1	P_2	P_3	P_4
Tasks				
t_2, t_3, t_{10}	8	15	14	16
t_6	2	1	8	9
t_7, t_8	16	17	9	7
t_1, t_4, t_5, t_9	15	16	27	10

Table 4: Modified Inter Task Communication Cost Matrix

ITCCM				
Tasks	t_2, t_3, t_{10}	t_6	t_7, t_8	t_1, t_4, t_5, t_9
t_2, t_3, t_{10}	0	3	2	4
t_6		0	4	1
t_7, t_8			0	6
t_1, t_4, t_5, t_9				0

Fig. 1 Optimal Execution Cost Graph

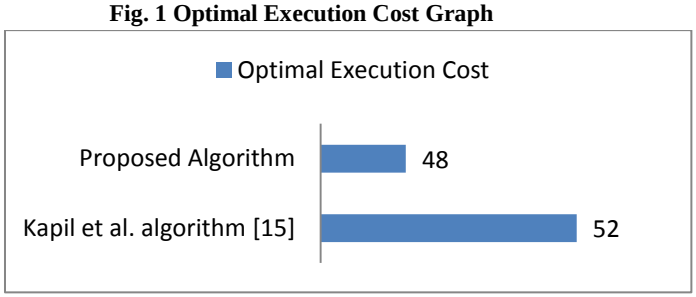

Fig. 2 Average Processor Utilization Graph

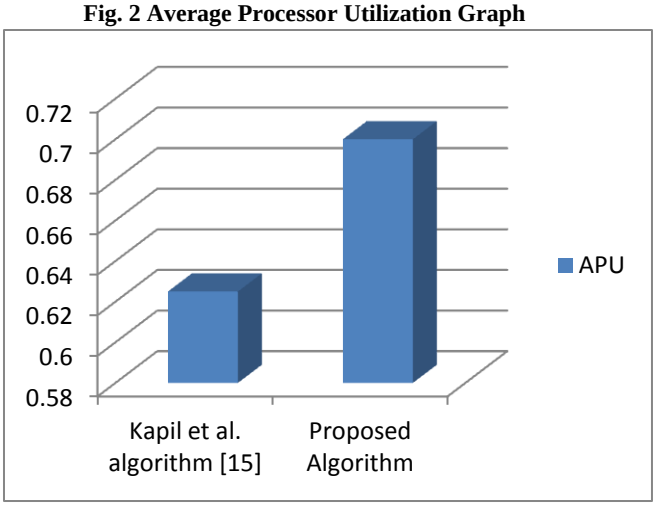

Table 5: Optimal System Cost

Processors	Tasks	Processor Load	Optimal System Cost			PU	APU
			EC	ITCC	EC + ITCC		
P_1	t_2, t_3, t_{10}	8	28	20	48	0.8	0.7
P_2	t_6	1				0.1	
P_3	t_7, t_8	9				0.9	
P_4	t_1, t_4, t_5, t_9	10				1	

5. CONCLUSION

In this paper, we have considered the static task allocation problem considering load sharing scheme, a criticalphase in DRTS with the goal of minimizing system cost, and maximizing thePU. The proposed model is based on aneffective clusteringto reduce ITCC. We present a straightforward and efficient algorithm to obtain optimal valuesof the objectives that we have considered in this paper. To measure theperformance of proposed model & algorithm, the same example, present in Kapil et al. [15] has been solved. In this case, it shows that the system cost is minimized by 7.69 % and the APU is maximized by 12 %. So by using this model the optimal solution can be achieved at all the times.

6. ACKNOWLEDGMENTS

We gratefully acknowledge support from Dean and faculty members,Department of Physical & Computer Science, FASC, MITS, Lakshmangarh, Sikar&Department of Math., Stats. & Computer Applications, Bundelkhand University, Jhansi for the same.

7. REFERENCES

[1] C.J. Hou, K.G. Shin, "Load sharing with consideration of future task arrivals in heterogeneous real time systems", In proceedings of the IEEE 13th Real Time Systems Symposium, 1992, pp 146-155.

[2] Kopetz, H. (1997). REAL-TIME SYSTEMS: Design Principles for Distributed Embedded Applications. Kluwer Academic Puplishers .

[3] Z. Zeng and V. Bharadwaj, "Design and Performance Evaluation of Queue-and-Rate-Adjustment Dynamic Load Balancing Policies for Distributed Networks", IEEE Trans. On Computers, vol. 55, no. 11, pp. 1410-1422, November 2006.

[4] Pereng-yi RICHARD MA, Edward Y.S.LEE, Masahiro TSUCHIYA, "A Task Allocation Model for Distributed Computing Systems", IEEE Trans. on Computers, Vol.C-31, No. 1, January 1982, pp. 41-47.

[5] Chien-Chung Shen, Wen-Hsiang Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems using a Minimax Criterion", IEEE Trans. on Computers, Vol. C-34, No.3, March 1985, pp. 197-203.

[6] Wesley W Chu, Lance M.T.Lan, "Task Allocation and Precedence Relations for Distributed Real Time Systems", IEEE Trans. on Computers, Vol. C-36, No.6, June 1987, pp. 667-679.

[7] C. Siva Ram Murthy, K.N.Balsubramaniya Murthy, A.Sreenivas, "Scheduling of Precedence-Constrained Parallel Program Tasks on Multiprocessors", Microprocessing and Microprogramming, Vol. 36, 1992/93, pp. 93-104.

[8] GamalAttiya, YskandarHamam, "Task allocation for maximizing reliability of distributed systems: A simulated annealing approach", J. Parallel Distrib. Comput. 66 (2006) 1259 – 1266.

[9] D.P.Vidyarthi, A.K.Tripathi, "Precedence Constrained Task Allocation in Distributed Computing System", Int. J. of High Speed Computing, Vol. 8, No. 1, 1996, pp. 47-55.

[10] S. Karthik, C. Siva Ram Murthy, " Improved Task Allocation Algorithms to Maximize Reliability of Redundant Distributed Systems", IEEE Trans. on Reliability, Vol. 44, No. 4, Dec. 1995, pp. 575-586.

[11] P. K. Yadav, M. P. Singh, Kuldeep Sharma, "An Optimal Task Allocation Model for Sustem Cost Analysis in Heterogeneous Distributed Computing Systems: A Heuristic Approach", International Journal of Computer Applications, Vol. 28, No. 4, August 2011.

[12] A. Kumar, M.P. Sing, 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.

[13] Aravind H, C Rajgopal, K P Soman, "A Simple Approach to Clustering in Excel", International Journal of Computer Applications, Vol. 11, No. 7, December 2010.

[14] AnuragRaii, VikramKapoor, "Efficient Clustering Model for Utilization of Processor's Capacity in Distributed Computing System", International Journal of Computer Applications, Vol. 44, No. 23, April 2012.

[15] KapilGovil, Avanish Kumar "A Modified and Efficient Algorithm for Static Task Assignment in Distributed Processing Environment", International Journal of Computer Applications, Vol. 23, No. 8, June 2011.