Fuzzy Logic Based Re-Engineered Model for Maximization of Average Processor Utilization of Distributed Real-Time System

Avanish Kumar[1], Anju Khandelwal[2], Urmani Kaushal[3]

[1]*Professor, Department of Mathematical Sciences and Computer Applications, BundelkhandUniversity, Jhansi, India*
[2]*Department of Basic Engg. Sciences, S.R.M.S. Women's College of Engineering and Technology, Bareilly, India*[2]
[3]*Assistant Professor, FASC, MITS University, Lakshmangarh Rajasthan, India*[3]

Email: dravanishkumar@yahoo.com, dranjukhandelwal@rediffmail.com, urmani10kaushal@gmail.com

Abstract: High performance parallel applications are using distributed real time system [DRTS] as phenomenal platform. DRTS process the parallel applications over multiple processors of the system. Performance of the system can be amended by efficient & evenly allocation methodology for parallel application's tasks over the processors available in the system. Task allocation is NP-hard or NP-complete problem. Fuzzy logic based and dynamic load sharing technique based model for the distribution of parallel application's tasks has been proposed and deployed in this paper. In the new heuristic suggested in model, the limitation of memory has been deployed using fuzzy rules and the triangular member function with its FIS variables has been used. In the proposed model k-mean clustering is being used and MATLAB 7.11.0 has been used to simulate the proposed model.

Keyword: Task Allocation, Distributed Real Time System, Fuzzy Logic, FIS Variable, Membership Function, Parallel Application, Cluster, NP-complete, Dynamic Load Sharing,

1. INTRODUCTION

The real-world applications demands high capacity of processing. It can't be achieved by single processor system. The approach to solve such types of problems is distributed computing system which can process the parallel application over multiple processors of the system. In a distributed real time system heterogeneous nodes are connected via some communication links [1,2]. In a distributed real time system, the utilization of remote computing resources is crucial. The appropriate task allocation technique which is capable to utilize the processors in higher degree will increase the performance and the level of flexibility and modularity. If the model for task allocation in distributed real-time system is designed and planned efficiently, it may provide economical, efficient and more reliable execution for parallel applications[3]. In [2,4,5,6], load has been distributed in system with considering the limitation of the memory whether such number of tasks can fulfill the memory requirement of the tasks or not.

This may causes the uneven load distribution across the system. This load imbalance occurs has been observed in the models proposed in [2,5,7].It may be unsafe for the system performance in terms of the mean response time of tasks and processor utilization [8]. The model proposed and discussed in [2,7,9,10,11,12], focused only in the minimization of execution and inter-task communication cost rather than considering resource utilization. In the proposed model in this paper has concentrated on the minimization of execution cost, inter task communication as well as maximization of processer utilization.

The proposed model is efficient that imitates batter results then the model proposed in [2]. Here in this model, clusters has been formed at first stages on the basis of execution cost and inter task communication cost and at second stage restricted assignment has been recognized by using fuzzy logic for the task allocation. The model has followed scalable active approach, which is improving the performance and processor utilization iteratively of distributed real-time system. Organization of the paper is as follows. In the second section the problem has been formulated. In the next section the problem has been stated and it describes the model and its components in detail. The technique has been described in the same section as well. In next section the model has been simulated using MATLAB 7.11.0. Results of simulated experiment are collectively presented in the conclusion section.

2. PROBLEM FORMULATION

The parallel application should complete its execution over a DRTS in lesser cost than over a standalone system. The parallel applications running over the DRTS is having m number of tasks which are to be allocated on n number of nodes of the distributed system for execution. This allocation of task should be done in such a manner that inter task communication can be reduced. Execution constraints of the tasks must be fulfilled. A processor can hold limited number of tasks and memory is also the limit. So these constraints had been taken into consideration in the proposed model, which provide an optimal solution for the assignment of the set of *"m"* tasks of programs over the set of *"n"* processors / nodes (*where, $m > n$*) DRTS[2].

Performance enhancement of the distributed system is the objective of this problem and it will be achieved by appropriate allocations of tasks over system.

2.1 Data Structures &Definitions Used

TMSV (Task& Memory Status Vector) data structure and T*MSV* Table proposed in [2] are being used to solve the problem in this paper. Description of the data structures are as follows:

2.1.1 TMSV

It stores the processor's status information. Structure of *TMSV* is shown in Table 1. Element "*Processor ID*" is representing the processor unique name in the system.
Total number of modules assigned to the processing node P_l can be calculated as follows:

$$y = N_l - x \qquad (1)$$

Total memory cost of processing node can be achieved by following cost function

$$f(z) = M_l - \sum_{x=1}^{x}(z_x) \qquad (2)$$

z_x Memory required by x^{th} Module.
M_l Maximum Memory Limit of Processing Node P_l.

TABLE 1: TMSV STRUCTURE

S.No.	Notations	Description

1.	P_l	Processor ID
2.	N_l	Maximum Number of Modules allowed on Processing Node P_l
3.	M_l	Maximum Memory Limit of Processing Node P_l
4.	X	Number of Module Assigned to the Processing Node P_l
5.	Y	Number of Available Module on Processing Node P_l $(N_l -x)$
6.	$f(z)$	Memory Available

2.1.2 TMSV Collection

TMSV's collection is the group of *TMSV* of all the processing nodes in a DRTS shown in Table 2.

TABLE 2: TMSV'S COLLECTION

Processor	No. of Tasks (Maximum)	Memory Capacity (Maximum)	Tasks Assigned	Available Tasks Capacity	Memory Available
P_1	N_1	M_1	Maximum x module from m tasks	Y	f(z)
P_2	N_2	M_2	Maximum x module from m tasks	Y	f(z)
P_3 . .	N_3 . .	M_3 . .	Maximum x module from m tasks . .	y . .	f(z) . .
P_n	N_n	M_n	Maximum x module from m tasks	Y	f(z)

2.1.3 Notations

T	:	Set of tasks of a parallel program to be executed.
P	:	Set of processors in DRTS.
n	:	Number of processors.
m	:	Number of tasks formed by parallel application.
k	:	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.
X	:	An allocation matrix of order $m*n$, where the entry $x_{il} = 1$; *if* i^{th} task is allocated to l^{th} processor and 0; otherwise
CIM(,)	:	Cluster Information Matrix.
ECM(,)	:	Execution Cost Matrix.
ITCCM(,)	:	Inter Task Communication Cost Matrix.
FRAPM(,)	:	Fuzzy Restricted Assignment Priority Matrix

2.2 Definitions

2.2.1 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 [5]. If a task is not executable on a particular processor, the corresponding execution cost is taken to be infinite (∞).

2.2.1 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 separate processor during the execution process. If two tasks executed on the same processor then $cc_{ij} = 0$ [2].

2.3 Assumptions

To allocate the tasks of a parallel program to processors in DRTS, the following assumptions [2,6,9] have been made:

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

2.3.2 The 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 is completed. Whenever a group of tasks is assigned to the processor, the inter task communication cost (*ITCC*) between them is 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 tm 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 Proposed Mathematical Model for Task Allocation

In this section, a task allocation model has been developed to find the optimal system cost so that system performance could be enhanced. Effective allocation of parallel applications' tasks may increase the performance of the distributed system.

Hereafter, in order to allocate the tasks of such program to processors in DRTS, the information about certain inputs such as execution cost, inter task communication cost. While obtaining such information is beyond the scope of this paper. It is assumed that the required information is available before the execution of the program. In proposed model, processor execution cost and task clustering has been considered for this system. The clustering has been done at two levels in the model, one at initial stage and the other just before allocation decision.

2.4.1 Execution Cost (EC)

The task allocation given as: $X: T \rightarrow P,\ X\ (i) = l$. For the task allocation X, 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 X, the execution of all the tasks assigned to l^{th} processor can be computed as:

$$\mathrm{EC(X)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{l=1}}^{\mathrm{m}} ec_{il} x_{il} \qquad (3)$$

$$where, x_{ij} = \begin{cases} 1\ if\ \ i^{th} taskisassignedtol^{th} processor \\ 0,otherwise \end{cases}$$

2.4.2 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 [2], Z_{ki}

$$\sum_{x_i I C_k} \|X_i - \overline{X}_k\|^2 = \sum_{i=1}^{m} Z_{ki} \|X_i - \overline{X}_k\|^2 \qquad (4)$$

Where the cluster mean is defined as $\overline{X}_k = \frac{1}{m_k}\sum_{x_i I C_k} X_i$ and $m_k = \sum_{i=1}^{m} Z_{ki}$ 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:

$$\varepsilon_k = \sum_{i=1}^{m}\sum_{k=1}^{k} Z_{ki} \|X_i - \overline{X}_k\|^2 \qquad (5)$$

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

To group the tasks *K-means* has been used. It is unsupervised learning algorithms for clustering. The data set will be clustered done with pre-defined number of clusters. In this grouping technique *k* centroids is to be defined, one for each cluster. The centroids must be positioned in carefully because of different location causes different result. In the next phase associate each data point to the nearest centroid. Repeat this step for all data sets. Here at this stage the early grouping is done. Repeat the work of assignment and recalculation of centroid's positions until the centroids no longer move. Finally, this algorithm aims at minimizing an objective function.

2.4.3 Fuzzy Logic

Generalized classical logic is conceptualized by fuzzy logic. Modern fuzzy logic was developed by Lotfi Azdeh in mid-1960s. Initially it was modeled for the problems in which imprecise data is being used or the rules of inferences are formulated in general way to make use of diffuse categories [13]. The member function used in fuzzy logic is as follows [14]:

2.4.3.1 Membership Function

Membership Function for a fuzzy set A on the universe of discourse Y is defined as µA:Y → [0,1], where each element of Y is mapped to a value between 0 and 1. This value, called membership value or degree of membership, quantifies the grade of membership of the element in Y to the fuzzy set A. The triangular function is used to solve the problem in this paper. It is defined as follows:

Lower limit *a*, Upper limit *b*, Value *m*, where,$a < m < b$.

$$\mu_A(Y) = \begin{cases} 0, & y \le a \\ (y-a)/(m-a), & a < y \le m \\ (b-y)/(b-m), & m < y < b \\ 0, & y \ge b \end{cases} \qquad (6)$$

3. Proposed Task Allocation Technique and Algorithm

3.1 Technique

A distributed system having 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 which has to be executed over *n* processors of the system. The processing time of each task over every processor is given in Execution Cost Matrix *ECM (,)* of order *m x n*. The communication cost of task is provided in Inter Task Communication Cost Matrix *ITCCM (,)* of order *m x m*.

Minimization in overall system cost could enhance the performance of the system. Given *m* tasks are to be processed over *n* processors where number of processors are very less the number of tasks (*m*>>*n*). So the tasks are grouped in *k* clusters. Now *k* clusters are to be allocated on *n* processors. For clustering *k-mean* clustering algorithm has been used. Here *m* vectors of tasks are to be placed in *k* clusters. In this process the first step is to find *k* initial points for each cluster, which is represented by task vector. 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. 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 is calculated by using equation (4) [2].

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.

In distributed system each processor has limited memory capacity. So while allocation of tasks this constraint should be considered. To implement this constraint fuzzy logic has been used.

In the process of assigning *k* clusters to *n* processors, next level of clustering will be done on the basis of ec_{il} (execution cost) constraint. If there is any change in the clusters the *ECM (,)* and *ITCCM (,)* should be recalculated accordingly. Once the final assignments are in hand optimal cost of assignment is to be computed using eq. (3). The objective function to calculate total system cost is as follows:

$$\text{Total Cost} = EC + CC \qquad \textbf{(7)}$$

3.2 Proposed Algorithm

The structure of algorithm is as follows:

Step-1: Start
Step-2: Read the number of processors in *n*
Step-3: Read the number of tasks in *m*
Step-4: Read the number of tasks in the task in *l*
Read the *ECM (,)* of the task of order *l x n*
Step-5: Read number of clusters in *k* (in this case it equal to number of processors)
Step-6: Read the Inter Task Communication Cost Matrix *ITCCM (,)* for each task of order *l x l*
Step-7: Read the PMCAV of each task and store it in PMCAV's Collection
Step-8: Apply *k-mean* clustering algorithm on *ECM (,)*
Step-9: Cluster information is stored in Cluster Information Matrix *CIM (,)*
Step-10: Modify the *ECM (,)* by adding the processing time of tasks in each cluster
Step-11: Apply Fuzzy logic to find restricted assignment priority and store it in *FRAPM (,)*
Step-12: Modify *ECM (,)* for restricted assignment provided in *FRAPM (,)*
Step-13: Modify the *ITCCM (,)* by putting communication zero amongst those tasks which are in the same cluster
Step-14: Apply *munkres* in modified *ECM (,)*
Step-15: Modify *PMCAV* Collection according the assignment made in Step-14
Step-16: Calculate Execution Cost, Inter Task Communication Cost
Step-17: Optimal Cost = Execution Cost + Inter Task Communication Cost
Step-18: End

3. IMPLEMENTATION

To illustrate the proposed algorithm following data set has been used. It is implemented in MATLAB. It is assumed that the *ITCC* Matrices, the Execution Cost Matrices and *PMCAV's* Collection table are given in units of time. Given a set of twenty tasks {$T_1,T_2,T_3,T_4,T_5,T_6,T_7$,T_8, T_9, T_{10}, T_{11}, $T_{12},T_{13},T_{14},T_{15},T_{16},T_{17},T_{18},T_{19},T_{20}$} and a set of five processors {P_1, P_2, P_3, P_4, P_5}.Execution Cost Matrix has been given in Table 3 and inter task communication detail has been provided in Table 4. Memory requirement of all tasks is in the Table 5. The detail of memory capacity of each node is given in Table 6. Clustering information Matrix is provided in Table 7. For finding the restricted assignment, set of rules has been formed which is shown in figure 6. Member function is presented in figure 2. Memory Available, Memory Required and Assignment Priority FIS variables are shown in figure 3, 4 and 5 respectively. The graph for assignment priority is shown in figure 7. The restricted assignment priority is shown in Table 8. Cluster assignment status is shown in Table 9. After final assignment the memory status is shown in Table 10. Table 11 is showing the final optimized system cost, processor utilization and average processor utilization.

TABLE 3: EXECUTION COST MATRIX

	P_1	P_2	P_3	P_4	P_5
T_1	10	5	10	5	10
T_2	20	5	35	10	5
T_3	10	10	10	10	10
T_4	15	10	20	15	15
T_5	*10*	*15*	*20*	*15*	*30*
T_6	15	25	15	10	10
T_7	30	40	25	20	5
T_8	20	5	10	15	10
T_9	10	*5*	*5*	*15*	*20*
T_{10}	5	10	25	20	30
T_{11}	10	25	5	5	5
T_{12}	25	10	5	10	25
T_{13}	*5*	*10*	*15*	*25*	*25*
T_{14}	10	15	20	25	30
T_{15}	5	10	10	10	10
T_{16}	5	10	10	20	20
T_{17}	5	10	6	3	2
T_{18}	7	8	10	3	1
T_{19}	6	5	15	10	20
T_{20}	8	10	12	14	16

TABLE 4: INTER TASK COMMUNICATION COST MATRIX

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
T_1	0	1	5	4	6	7	9	9	3	6	0	1	4	3	10	9	8	3	7	7
T_2		0	2	0	9	5	1	8	3	6	9	8	10	0	1	6	4	10	9	9
T_3			0	4	6	6	5	9	0	0	3	9	10	3	3	2	9	2	4	9
T_4				0	3	1	6	7	2	6	1	0	5	8	2	9	9	0	7	10
T_5					0	9	12	1	9	10	0	6	15	13	5	10	3	15	15	13
T_6						0	11	0	14	0	3	8	14	15	0	10	2	8	1	1
T_7							0	8	15	8	14	8	14	2	2	11	4	6	4	5
T_8								0	10	1	2	4	6	8	15	8	8	6	1	5
T_9									0	5	10	10	9	8	5	3	10	14	3	11
T_{10}										0	10	3	13	9	7	11	15	9	9	4

T_{11}											0	8	10	10	3	9	0	4	5	0
T_{12}												0	4	4	0	9	2	10	2	6
T_{13}													0	1	6	6	9	4	4	2
T_{14}														0	0	4	5	3	2	2
T_{15}															0	9	1	15	8	8
T_{16}																0	5	8	11	4
T_{17}																	0	8	1	4
T_{18}																		0	1	0
T_{19}																			0	8
T_{20}																				0

TABLE 5: MEMORY REQUIREMENT OF TASKS IN UNITS

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
4	1	6	3	5	2	4	1	4	5	6	3	2	1	2	3	4	2	3	1

TABLE 6:PMCAV'S COLLECTION (BEFORE ALLOCATION)

Processor	Memory Capacity (Maximum)	Memory Available
P_1	55	50
P_2	44	40
P_3	35	35
P_4	36	30
P_5	10	10

TABLE 7: CLUSTER INFORMATION MATRIX

Cluster	Clustered Tasks	Restricted Assignment	Memory Required
C - 1	T_4 , T_8 , T_9 , T_{12} , T_{16} , T_{19} , T_{20}	P_5	18
C - 2	T_7	-	4
C - 3	T_2	-	1
C - 4	T_5 , T_{10} , T_{13} , T_{14}	P_5	13
C - 5	T_1 , T_3 , T_6 , T_{11} , T_{15} , T_{17} , T_{18}	P_5	26

TABLE 8: FUZZY RESTRICTED ASSIGNMENT PRIORITY MATRIX

	P_1	P_2	P_3	P_4	P_5
C – 1	0.6651	0.7077	0.4837	0.5311	0.1632
C – 2	0.8293	0.8356	0.8326	0.8318	0.8355
C – 3	0.8293	0.8307	0.8307	0.8307	0.8307
C – 4	0.8293	0.8356	0.5809	0.6319	0.1601
C – 5	0.4944	0.4935	0.2883	0.3279	0.1634

TABLE 9: CLUSTER ASSIGNMENT STATUS ON PROCESSORS

Processor	Cluster Assigned
P_1	C – 1
P_2	C – 3

P_3	C – 4
P_4	C – 5
P_5	C – 2

TABLE 10: PMCAV'S COLLECTION (AFTER ALLOCATION)

Processor	Memory Capacity (Maximum)	Memory Available
P_1	55	32
P_2	44	39
P_3	35	12
P_4	36	4
P_5	10	6

TABLE 11: OPTIMAL SYSTEM COST

Processors	Tasks	Processor Load	PU	APU	Optimal System Cost		
					EC	ITCC	EC + ITCC
P_1	T_4 , T_8 , T_9 , T_{12} , T_{16} , T_{19} , T_{20}	30	0.5455	0.5853	161	51	212
P_2	T_2	55	1				
P_3	T_5 , T_{10} , T_{13} , T_{14}	25	0.4546				
P_4	T_1 , T_3 , T_6 , T_{11} , T_{15} , T_{17} , T_{18}	46	0.8363				
P_5	T_7	5	0.09				

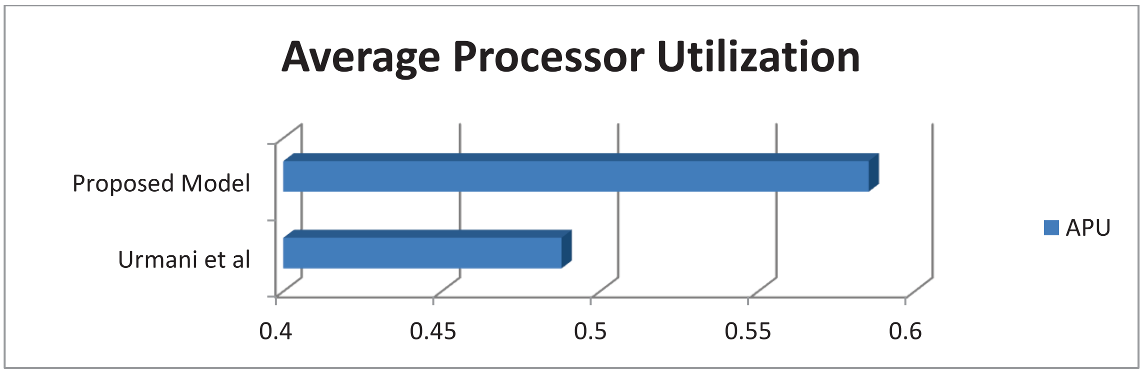
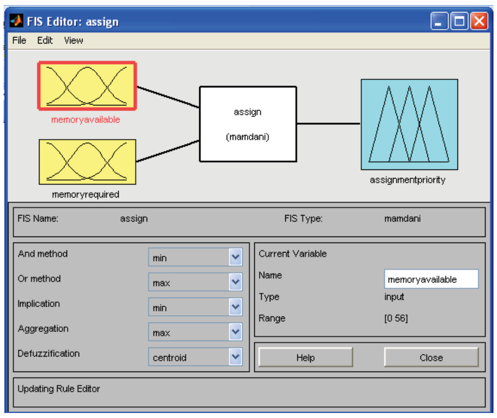


Fig. 1: Average Processor Utilization

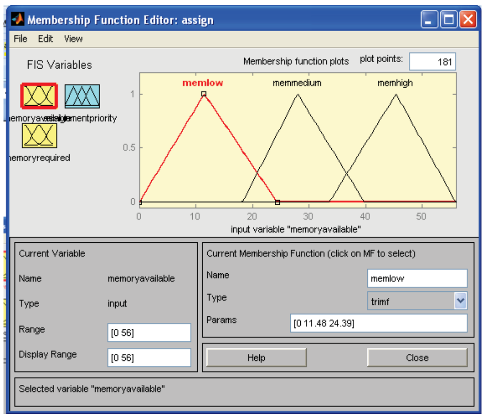


FIS Name: assign FIS Type: mamdani

And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

Current Variable
Name memoryavailable
Type input
Range [0 56]

Updating Rule Editor


Membership Function Editor: assign
File Edit View
FIS Variables
Membership function plots
plot points: 181
memlow
memmedium
memhigh
input variable "memoryavailable"
Help
Close

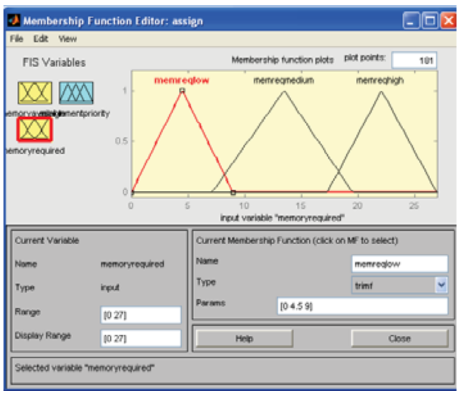
Current Variable
Name memoryavailable
Type input
Range [0 56]
Display Range [0 56]

Current Membership Function (click on MF to select)
Name memlow
Type trimf
Params [0 11.48 24.39]

Selected variable "memoryavailable"

Fig. 2: Member Function FIS Variable

Fig. 3: Memory Available

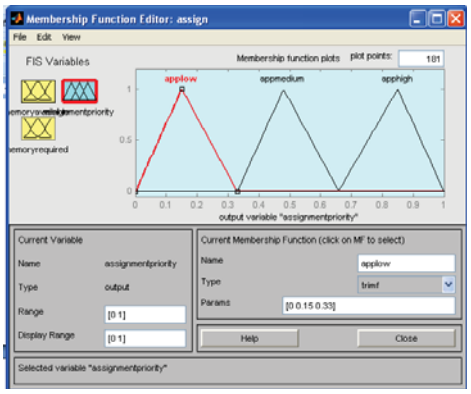

Membership Function Editor: assign
File Edit View
FIS Variables
Membership function plots
plot points: 181
memreqlow
memreqmedium
memreqhigh
input variable "memoryrequired"
Help
Close

Current Variable
Name memoryrequired
Type input
Range [0 27]
Display Range [0 27]

Current Membership Function (click on MF to select)
Name memreqlow
Type trimf
Params [0 4.5 9]

Selected variable "memoryrequired"

Fig. 4: Memory Required FIS Variable

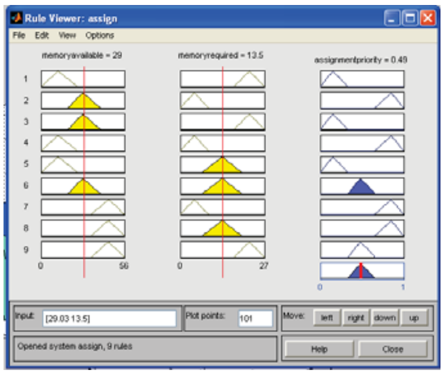

Membership Function Editor: assign
File Edit View
FIS Variables
Membership function plots
plot points: 181
applow
appmedium
apphigh
output variable "assignmentpriority"
Help
Close

Current Variable
Name assignmentpriority
Type output
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name applow
Type trimf
Params [0 0.15 0.33]

Selected variable "assignmentpriority"

Fig. 5: Assignment Priority FIS Variable



Input: [29.03 13.5] Plot points: 101

Opened system assign, 9 rules

Fig. 6: Rules

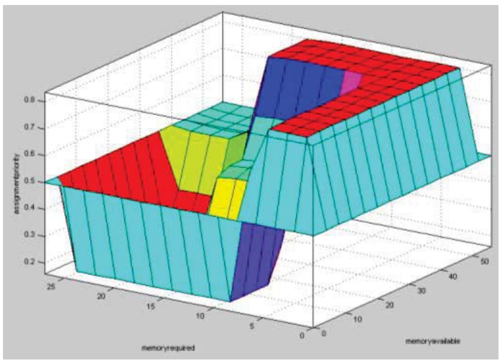


Fig. 7: Memory Constraint

5. CONCLUSION

The model proposed in this paper is based on effective two levels clustering and fuzzy logic based approach for checking memory constraint. The task allocation is done using dynamic the number of processors. In problem solved in this paper memory constraint has been taken into consideration. Before allocation of tasks on the processor, node's memory capacity has been checked and the decision has been taken by applying fuzzy logic. Restricted assignments decided by memory availability which is calculated by using fuzzy logic has been considered at time of the allocation. In the solved problem the Average Processor Utilization (AUP) has increased by 19.968 % in comparison with the model proposed by Urmani et al. [2] shown in figure 1.

References

[1] Urmani Kaushal and Avanish Kumar, "Performance Intensification of DRTS under Static Load Sharing Scheme," *International Journal of Computer Applications*, vol. 71, no. 16, pp. 55-59, June 2013.

[2] Urmani Kaushal and Avanish Kumar, "Improving the Performance of DRTS by Optimal Allocation of Multiple Tasks under Dynamic Load Sharing Scheme," *International Journal of Scientific & Engineering Research*, vol. 4, no. 6, pp. 1316-1321, June 2013.

[3] A. Abdelmageed Elsadek and B. Earl Wells, "A Heuristic model for task allocation in heterogeneous distributed computing systems," *The International Journal of Computers and Their Applications*, vol. 6, no. 1, pp. 0-35, March 1999.

[4] P.K. Yadav, M.P. Singh, and Kuldeep Sharma, "An Optimal Task Allocation Model for System Cost Analysis in Hetrogeneous Distributed Computing Systems: A Heuristic Approach," *International Journal of Computer Applications*, vol. 28, no. 4, pp. 30-37, August 2011.

[5] Kapil Govil, "A Smart Algorithm for Dynamic Task Allocation for Distributed Processing Environment," *International Journal of Computer Applications*, vol. 28, no. 2, pp. 13-19, 2011.

[6] Kapil Govil and Avanish Kumar, "A Modified and Efficient Algorithm for Static Task Assignment in Distributed Processing Environment," *International Journal of Computer Applications*, vol. 23, no. 8, pp. 1-5, June 2011.

[7] Urmani, Avanish Kumar, and Narendra Kumar Kaushal, "Algorithm for Performance Improvement of DRTS Under Static Load Sharing Scheme," *IUP Journal of Information Technology*, vol. 9, no. 3, pp. 43-52, September 2013.

[8] Abbas Karimi, Faraneh Zarafshan, and Adznan b. Jantan, "A New Fuzzy Approach for Dynamic Load Balancing Algorithm," *International Journal of Computer Science and Information Security*, vol. 6, no. 1, pp. 1-5, 2009.

[9] Urmani Kaushal and Avanish Kumar, "Modified Clustered Approach for Performance Escalation of Distributed Real-Time System," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol II*, Suresh Chandra Satapathy et al., Eds. Vishakapatnam, India: Springer International Publishing, 2014, ch. 2, pp. 9-16.

[10] V.M. Lo, "Heuristic algorithms for task assighment in distributed system," *IEEE Trans. Comput.*, vol. 37, no. 11, pp. 1384-1397, 1988.

[11] Bora Ucara, Cevdet Aykanata, Kamer Kayaa, and Murat Ikincib, "Task Assignment in heterogeneous computing system," *J. Parallel Distrib. Comput.*, vol. 66, pp. 32-46, 2006.

[12] D.P. Vidyarthi and A.K. Tripathi, "Maximizing Reliability of Distributed Computing Systems with Task Allocation using Simple Genetic Algorithm," *J. of Systems Architecture*, vol. 47, pp. 549-554, 2001.

[13] Atul Kumar Tiwari, Anunay Tiwari, Cherian Samuel, And Satish Kumar Pandey, "Flexibility In Assignment Problem Using Fuzzy Numbers With Nonlinear Membership Functions," *International Journal of Industrial Engineering & Technology*, vol. 3, no. 2, pp. 1-10, January 2013.

[14] Sanjay Krishnankutty Alonso. (2013, Feb) eMathTeacher. [Online]. http://www.dma.fi.upm.es/java/fuzzy/fuzzyinf/main_en.htm