IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 4, Ver. I (Jul - Aug 2017), PP 26-33
www.iosrjournals.org

Feasible Surfaces of Optimal Utilization and Dynamic Load Balancing of Processors with a Fuzzy Tasks Scheduler for Distributed Real Times Systems

Preet Pal Singh[1] and Dipa Sharma[2]
[1]*Pt. L.M.S. Govt. P.G. College, Rishikesh, Uttarakhand*
[2]*S.D.M. Govt. P.G. College Doiwala, Uttarakhand*

***Abstract**: Most researchers concerning real time distributed scheduling assumes constraints to be accurate. However, in many situations the values of these parameters are indistinguishable. The indistinctness of parameters suggests that we make use of fuzzy logics to decide in what order the requests should be executed to make better utilization of systems. In this research, we are taking a fuzzy dynamic load balancing approach. We get the output feasible surfaces of the optimal utilization and load balanced distributed real time systems with the use of fuzzy inferences systems. We analysed the effects of different fuzzy membership functions on the optimal utilization and load balanced surfaces.*

***Keywords:** Fuzzy logic, Membership function, Distributed real time Systems, Load Balancing.*

I. Introduction

Over the years the hardware technology has grown on a massive pace with the result of increase in the use of distributed systems. These systems have the advantage of sharing of resources as well as processing power. The processes arrive in the system in a random manner on different nodes. When the jobs are being executed in parallel on different systems a decision has to be made on to which system a newly arrived job has to be send. Load balancing is the technique which helps in even distribution of the jobs among the available nodes so that the throughput can be increased.

The load balancing algorithms have two types in nature- First is Static Load Balancing in which performance of the nodes is determined at the beginning of execution then depending upon their performance the workload is distributed in the start by the server node. The ordinary processors calculate their allocated tasks and send their result to the server. A task is always executed on the assigned processor that is statics load balancing methods are non-preemptive. A general demerit of all static systems is that the final selection of host for process allocation is made when the process is create and cannot be changed during process execution to make changes in the system load [1]. Major load balancing algorithm are Round Robin [2] and Randomized Algorithms [3], Central Manager [4], Algorithm and Threshold Algorithm [1,5]. Other is Dynamic Load Balancing which differs from algorithms in that the workload is distributed among the nodes at runtime. The master assigns new processors to the slaves based on the new information collected [6, 7]. Unlike statics algorithms, dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded. Instead, they are buffered in the queue on the main host and allocated dynamically upon requests from remote hosts [1]. This method is consisted of Central Queue Algorithm and Local Queue Algorithm [8]. Load balancing algorithm works on the principle that in which situation workload is assigned, during at runtime or compile time. Comparison shows that statics load balancing algorithms are more stable. We can easily predict the behavior of static, but at the same time, dynamic distribution algorithms are always considered better than static algorithms [1].

Abundant related literature is available like Chen et al. [9] proposed a scheduling model and a related algorithm that is suitable for both uni-processor and multiprocessor systems which provide a method to detect work overloading and try to balance load with task dispatching. Dynamic integrated scheduling of hard real-time, soft real-time, and none real-time tasks are discussed in [10]. They can generate feasible schedules but their model is restricted to periodic tasks and change the tasks' periods dynamically when overloading occurs. Sabeghi and Deldari [11] focused on timing constraints but other recourse in environment of the system such as knowledge of unpredictable behavior of environment , dynamical behavior of the world, information with time bound are also implicit. In real world problems, to find practical compromises of these parameters are more feasible. But in some problems, it makes sense to convince objectives partially. The fulfillment degree of objectives can then be used as a parameter for making a decision. One especially straightforward method to achieve this is the modeling of these parameters through fuzzy logic. The same approach is also applied on uni-processor real-time scheduling in [12,13]. Hamzeh et al. [14] focused on a real-time multiprocessor system with heterogeneous periodic and non-periodic tasks and compare performance and

complexity of our proposed fuzzy scheduler with other algorithms using computer simulation while Naaz et al. [15] discussed the fuzzy load balancing algorithm and compared the effect of using different defuzzification methods.

On the other hand, Gulati et al. [16] discussed fuzzy approach that optimizes the complete system that too with less time complexity and in another part author discussed that reliability metrics had been taken as the major parameter for decision for scheduling. The priority was computed based on the values of Failure rate, CPU time and Reliability. Authors simulated their problem on Mamdani Fuzzy Inference Engine to evaluate the performance of the proposed methodology. Experimental results had shown that the proposed fuzzy scheduler creates feasible schedules for homogeneous and heterogeneous tasks. A heuristic task allocation model is in presented [17] which performs the proper allocation of task to most suitable processor to get an optimal solution. Authors developed a fuzzy membership functions for making the clusters of tasks with the constraints to maximize the throughput and minimize the parallel execution time of the system.

II. Fuzzy Inference Engine

Fuzzy logic [18,19] is a superset of conventional Boolean logic and extends it to deal with new aspects such as partial truth and vagueness. Fuzzy inference is a method to formulate the relation between given input set and an output using fuzzy logic. The essential elements of fuzzy logic are fuzzy sets, linguistic variables and fuzzy rules [20]. The linguistic variables' values are words, specifically adjectives like *small, little, medium, high, very high* etc. An element of fuzzy set has a couple of elements. It makes the concept of a classical set which allows to elements to have a partial membership. The degree to which the basic element "x" belongs to the fuzzy set A (expressed by the linguistic statement x is A) is characterized by a membership function (MF), $f_A(x)$. The membership function of a fuzzy set corresponds to the indicator function of the classical sets. Every element with its membership function can trace as a curve which represents how each point in the input space is mapped to the membership value between 0 to 1. The shapes of a membership function may be triangular, trapezoidal, gamma, and bell curves some others. This operation is a normalization of all inputs to the same range which makes direct effect on system performance and accuracy.

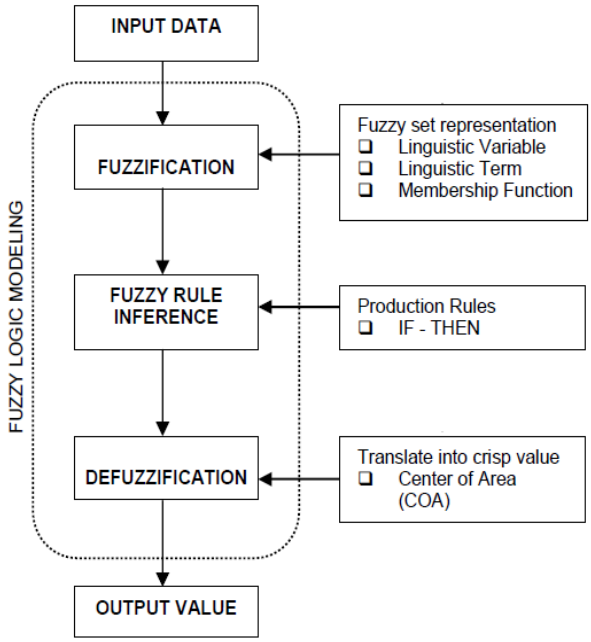


Fig. 1 Proposed inference model

A fuzzy set A is defined within a finite interval called universe of discourse U as $A: \{(x, f_A(x)), f_A(x): U \rightarrow [0,1]\}$. Fuzzy logic is an extension of Boolean logic which deals with the concept of partial truth i.e. a proposition is true up to what extent. With the classical logic everything can be expressed in binary terms (0 or 1, black or white, yes or no), fuzzy logic replaces Boolean truth values with a degree of truth. Degree of truth is a value by which we can capture the imprecise modes of reasoning that plays an important role to make decisions in an environment of vagueness and indistinctness.

Moreover, Fuzzy Inference Systems (FIS) have very simple concepts. They have three stages an input, a processing, and an output stage. The first stage maps the inputs, such as regency of reference, frequency of

reference etc, to suitable truth values and membership functions. The second stage deals with rules to generate corresponding results and then combines the results. Finally, the third stage converts the combined results back to related output value U is the given whole fuzzy linguistic input variable range. All related fuzzy sets make up the term sets, which is a set of labels within the linguistic variable described. The basis of fuzzy reasoning is formed by fuzzy rules. The relationship among linguistic, expressions, imprecise, qualitative of the input and output of systems are described by fuzzy rules. Generally, these rules work as expert knowledge and provide an easily understood knowledge representation scheme. A typical conditional fuzzy rule assumes a form such as - IF Speed is *Low* AND Race is *Dry* THEN Braking is *Soft.* Speed is *Low* AND Race is *Dry* is the rule's premise; while Braking is Soft is the consequent. The basis predicate might not be completely true or false, it is true or false with some degree of truth within the limits 0 to 1. We compute this value by applying the membership functions of the fuzzy sets labeled *Low* and *Dry* to the actual value of the input variables Speed and Race. After that, fuzzification is applied to the conclusion; the way in which this happens depends on the inference model. The fuzzy inference models have two types: Mamdani [21] and TSK or Sugeno [22].

Interpreting an if-then rule involves two distinct parts: first evaluating the antecedent and then applying results to the consequent (known as implication) [23,24]. If -then rules are easy in case of two valued or binary logic. If the premise is absolutely true, then its conclusion is also true. But in case of fuzzy logic, if the premise is true with some degree of true value, then the conclusion is also true to that same degree of true value. Mamdani-type [21] inference expects the output membership functions to be fuzzy sets. After the process of aggregation, we find a fuzzy set as output variable, which needs defuzzification. We can use a single spike as membership function of output rather than a distributed fuzzy set and in some cases it is more efficient. This is sometimes known as a singleton output membership function, which can be thought as a pre-defuzzification process is enhanced because it greatly simplifies the computation required by the more general Mamdani method, which uses the centroid of a two-dimensional function. Sugeno-type systems use weighted sum of a few data points in place of integrating across the two-dimensional function to find centroid . In general, if output membership function is either linear or constant than Sugeno-type systems are used to model any inference system. With the help of consulting an existing knowledge base an inference engine process the given inputs and produce an output byconsulting anexisting knowledgebase. The fuzzy inference has five as follows:

- Fuzzifying Inputs
- Applying Fuzzy Operators
- Applying Implication Methods
- Aggregating All Outputs
- Defuzzifying outputs

In brief, we are discussing all steps , in fuzzifying inputs, it converts inputs in fuzzy sets and find its degree to which these inputs belong to suitable fuzzy set via membership function. Once the inputs have been fuzzified , the degree to which each part of the antecedent has been satisfied for each rule is known. In case of more than one value that represent the result of the antecedent. The results of FIS are based on the tasting of all rules . so in order to make a decision the results of each rule must be combined in aggregation all fuzzy sets that represent the outputs of each rule are combined to get one fuzzy set. In defuzzification, the aggregated output i.e. single fuzzy set is defuzzified again converted into non-fuzzy sets. We can summaries all steps as follows-convert input characteristics into input membership functions, convert rules to a set of output characterization , output characterization to output membership function and the output membership function to a single-valued output.

III. Proposed Model

Major factors considered in our approach to determine the scheduling are categories of tasks, load balancing factor i.e. processor is sender or receiver and categories of processors (Ref. to Fig. 1)

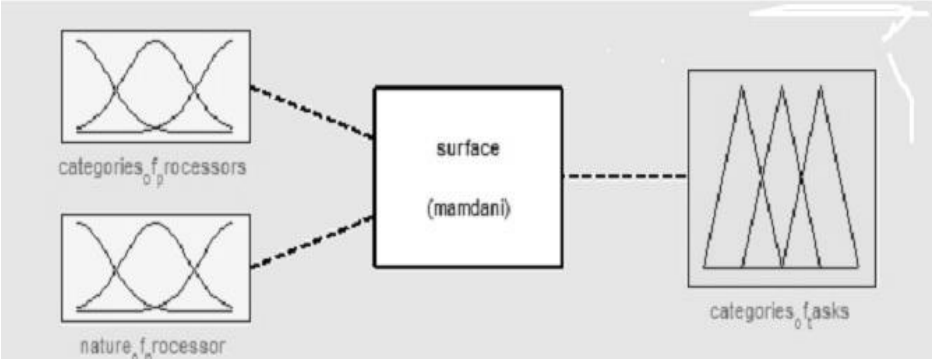


Fig. 1 Inference system Block Diagram

A task's allocation shows how the task is allocated to the appropriate processor . The inputs of these parameters are justified and represented as linguistic variables and fuzzy rules are then applied to those linguistic variables to compute the level value for deciding which task is allocated to which processors. First of all we have four categories of processors and tasks as their complexity and processing speeds with the help of fuzzy membership function Yadav et al. [17] categories them in four categories in terms of values of fuzzy membership function.

Very high speed processor (VHSP)	[0.0, 0.4]
High speed processors (HSP)	[0.4, 0.7]
Slow speed processors (SSP)	[0.7, 0.9]
Very slow speed processor (VSSP)	[0.9, 1.0]

Tasks for these are categorised in following four categories base on the value of membership function -

Very hard tasks (VHT)	[0.0, 0.4]
High tasks (HT)	[0.4, 0.7]
Easy tasks (ET)	[0.7, 0.9]
Very easy tasks (VET)	[0.9, 1.0]

Yadav et al. [17] gave a one to one mapping as:

Very hard tasks	←	to very high speed processors
Hard tasks	←	to high speed processors
Easy tasks	←	to slow processors
Very easy tasks	←	to very slow processors

IV. Policy of Dynamic allocation of tasks

In this paper we are taking the load balancing of this system. If any processor is overloaded then it is called sender and if a processor have less load then it is called receiver. If in any kind of cluster tasks are waiting in queue for allocation i.e. processor is sender than they will allocate to another category which have receiver processor instead of waiting in queue of the allocated category. If more than one processor are receiver of different categories than the priority of processors are from VHSP to VSSP.

Fuzzy rules try to combine these parameters as they are connected in real worlds. Some of these rules are:

- If (categories of processer is VHSP) and (nature of processor is receiver), then (categories of tasks is VHT).
- If (categories of processor is VHSP) and (nature of processor is sender), then (categories of tasks is none).
- If (categories of processor is HSP) and (nature of processor is receiver), then (categories of tasks is VHT).

V. Interpretation Of Results

We have done the implementation of scheduler on MATLAB. We have taken two input parameters f o r fuzzy implementation of our logic. The first input parameter is *processors categories* and the second one is nature of processors in terms of load i.e. it is receiver or sender and one output parameter i.e. categories of tasks to allocate. We measure all the parameters on scale of 0 to 1 on the basis of fact as first allocation given by yadav et al [17]. First, we allocate all tasks of same category to the processors of same category but at the time of allocation we keep in mind that that allocated processor is receiver or sender. If it is receiver than allocate but if it is sender than go to the next prior category of processor. Repeat this process until all the tasks are allocated. The fuzzy inference systems have many types of membership functions.

We are discussing some membership functions in short -The triangular is the simplest membership function which is formed by using straight lines. It described by the three points forming a triangle. The trapezoidal membership function has a flat top. The Gaussian distribution curve– a simple Gaussian curve and a two-sided composite of two different Gaussian curves. Another membership function, the generalized bell function gbell is specified by three parameters. Their forms are-

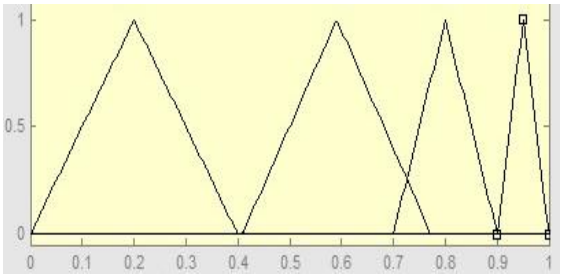


Fig.2.2 Triangular membership function

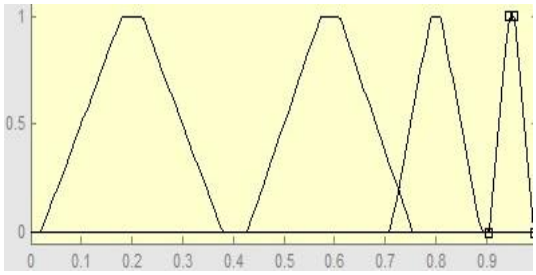


Fig. 2.3 Trapezoidal membership function

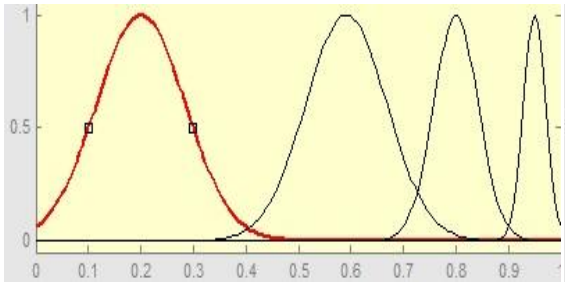


Fig.2.4 Gamma membership function

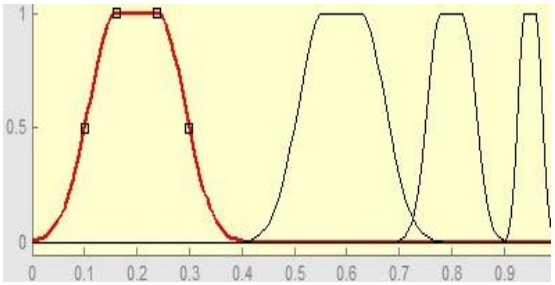


Fig.2.5 Gamma 2 membership function

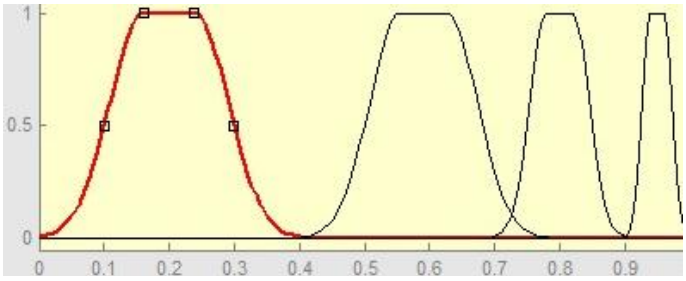


Fig- 2.6 Gbell membership function

On employing allocations method, we get optimal surface. In the fuzzy inference engine thus designed, we have taken different membership functions and obtained different surfaces as below-

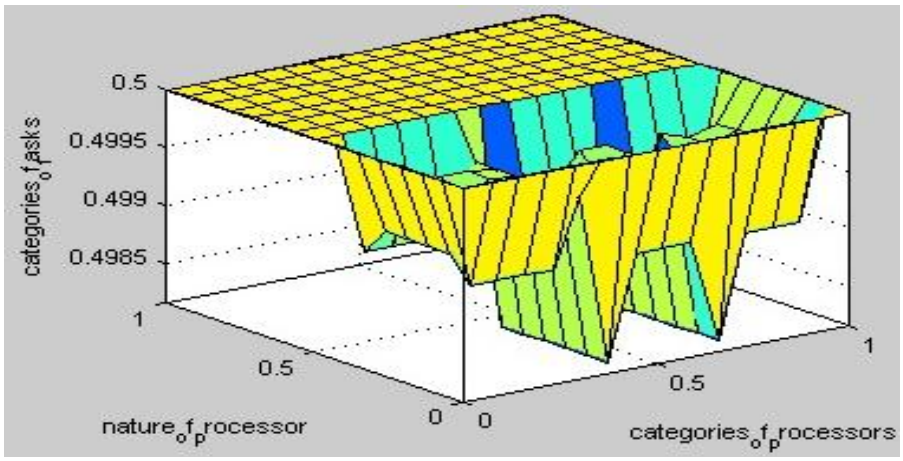


Fig 3.1 Triangular membership function

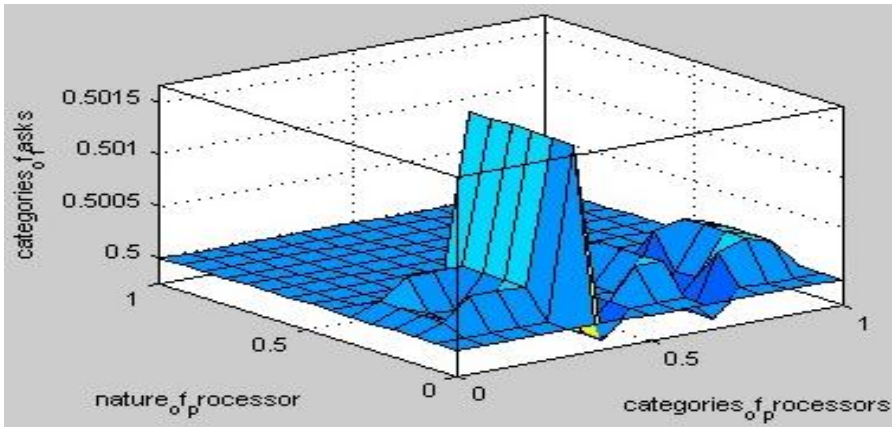


Fig 3.2 Trapezoidal membership function

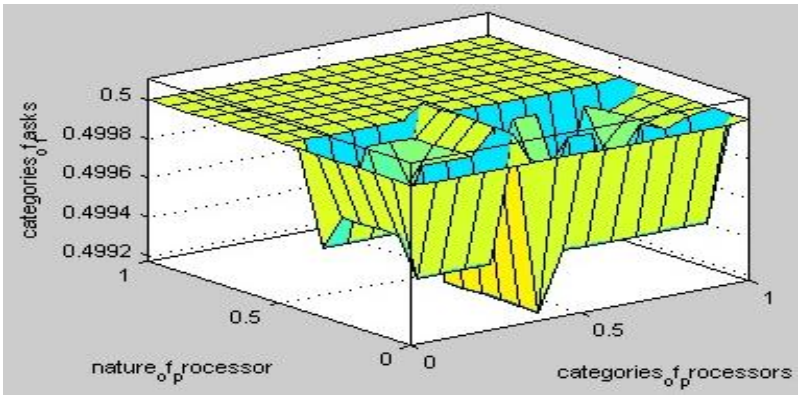


Fig-3.3 Gamma membership function

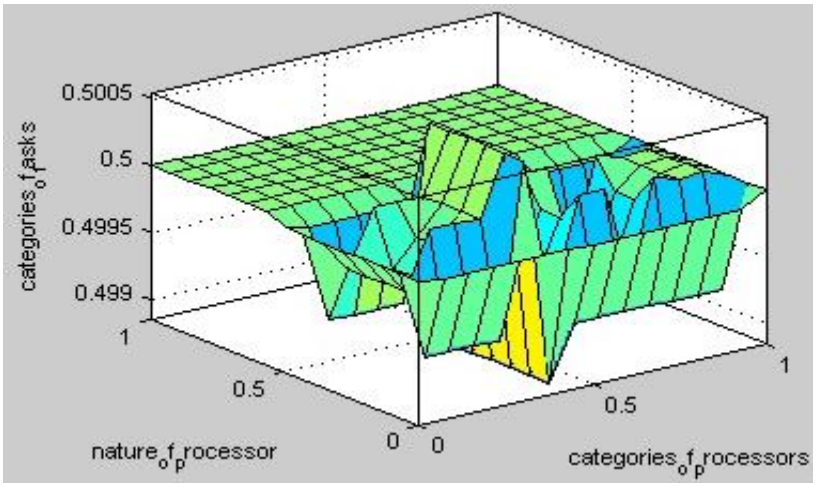


Fig-3.4 Gamma2 membership function

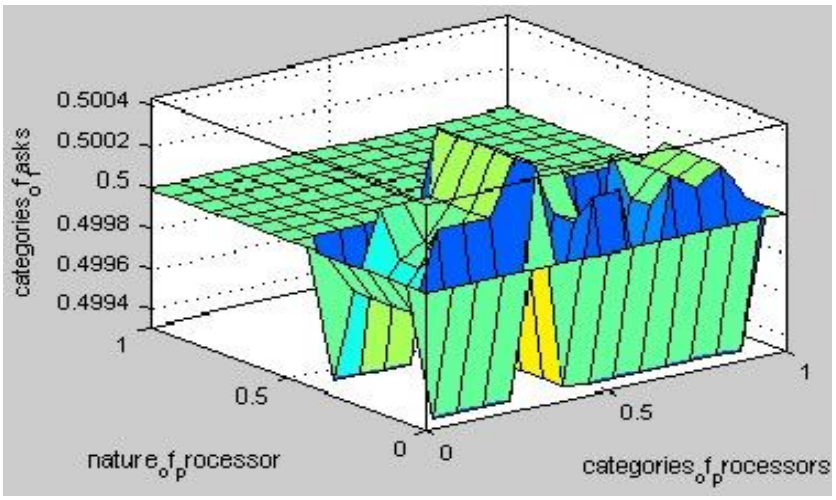


Fig-3.5 Gbell membership function

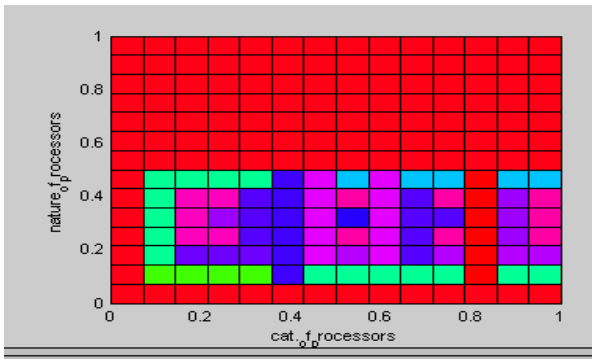
VI. Discussions and Conclusion

The surfaces in our results thus obtained by using the centroid defuzzification method, one of the best defuzzification method [15], besides that the best allocation method given by Yadav et al. [17] is used for getting the optimal balanced allocation surface. This gives a balanced and optimal utilization of processors together. The effects of different membership functions on the given surfaces are also analyzed. To make the comparisons between the different types of membership functions in the proposed interface for task allocation, a detailed statistical analysis have been performed. This analysis has been made between five different types of membership functions, i.e., *triangular, trapezoidal, bell-shape, Gaussian and truncated Gaussian (Gauss2mf)*. The performance evaluation has been done by using the following metrics:

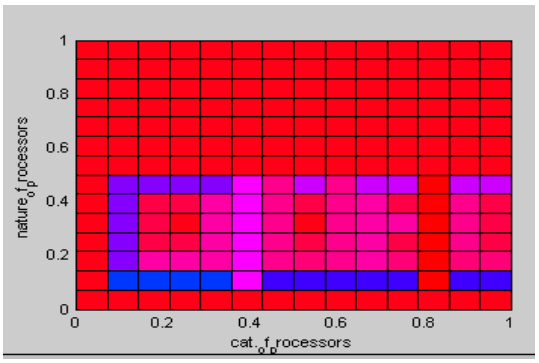
1. Surface plot
2. Pseudo-color plot
3. Statistical quantitative analysis

The surface plots of fuzzy interface for these membership functions are provided in Figs. 3.1 to 3.5. From the qualitative analysis of these surfaces, it can be observed that the smoothest surface has been obtained in case of Gaussian membership function. The other two cases, i.e. truncated Gaussian and Bell-shape functions based interfaces' surfaces are also smooth and having very marginal difference with the earlier one. In case of triangular and trapezoidal functions, surfaces are not so uniform and rapid changes can be noticed at different interface points.

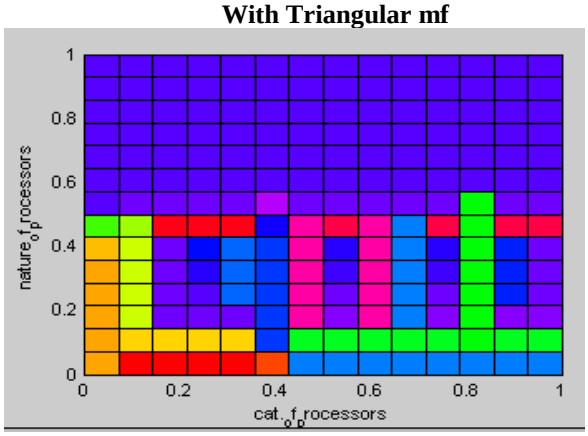
In terms of pseudo-color plot given in Figure-3.1 to3.5, it can be seen that again Gaussian, truncated Gaussian and bell-shape (Gbell) functions based plots are having very uniform changes in the colors in different consecutive cells. However, the variations in colors in case of other two membership functions-based plots are very sharp and hence we can claim that the fuzzy approximation of the variables for this interface system is not so accurate.



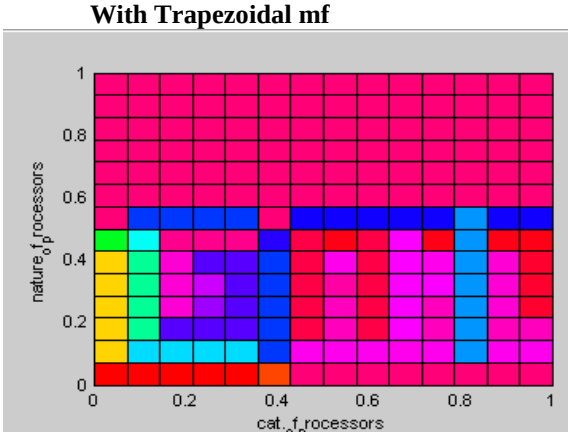
With Triangular mf



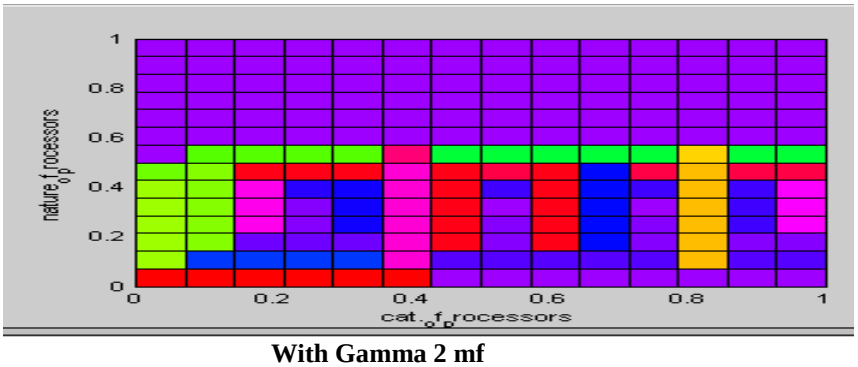
With Trapezoidal mf



With Gbell mf



With Gamma mf



With Gamma 2 mf

The third criterion for this comparison has been made by calculating the mean, standard deviation, minimum and maximum values of the output variable from the surfaces. To do this, both the input variables (categories and nature of the processors) are discritized uniformly. The categories are discritized by taking ten equal intervals starting from 0 to 10 with length 0.1 of each interval. In case of nature of the processors, the discritization has been done from in four equal intervals from 0 to 1 with step size 0.25. Hence, we selected a total 55 grid points.

Table-1 lists the values of the above mentioned parameters on these 55 points in case of all five membership functions based interfaces.

Parameters\MFs	**Triangular**	**Trapezoidal**	**Bell-shape**	**Gaussian**	**Gaussian-2**
Mean	0.4996	0.4977	0.5220	0.5233	0.5220
Standard Dev	0.0954	0.1071	0.1753	0.1929	0.1953
Minimum	0.2000	0.2000	0.2017	0.2000	0.2000
Maximum	0.8000	0.8000	0.9164	0.9496	0.9464

Table 1: Values of different parameters corresponding to various membership functions based interfaces

From the above table, it is clear that the distribution of different processors among various tasks is performed in last three columns, because of the high values of the standard deviation. Moreover the mean values in the last three columns are more than the first two; this clearly indicates that the task allocation towards the high-speed processor is more compare to others. Similar kind of comments can be made on the basis of minimum and maximum values. Since the span (difference) from the minimum and maximum values is higher in case of Gaussian and truncated Gaussian, hence types of tasks are allocated uniformly and as per the strategy chosen by us in case of these two interfaces compared to others. Analysis in this work may be very useful in future researches.

References

[1]. S. Sharma, S. Singh and M. Sharma, "Performance Analysis of Load Balancing Algorithm," World Academy of Sciences, Engineering and Technology, Vol. 38, 2008.

[2]. Z. Xu and R. Huang , "Performance Study of Load Balancing Algorithms in Distributed Web Server Systems" CS213 Parallel and Distributed Processing Project Report.

[3]. R. Motwani, P. Raghavan, "Randomized algorithms," ACM Comput. Surv., Vol.28, pp.33-37,1966.

[4]. P.L. McEntire, J.G. O'Reilly, and R.E. Larson, Distributed Computing: Concepts and Implementations, New York, IEEE Press, 1984.

[5]. W.I. Kim and C.S. Kang, "An adaptive soft handover algorithm for traffic-load shedding in the WCDMA mobile communication system," presented at WCNC'2003, 2003.

[6]. S. Malik, "Dynamic Load Balancing in a Network of workstation" 19 November, 2000.

[7]. N.Y.T. Wang and A.R.J.T. Morris , "Load Sharing in Distributed Systems," IEEE Transactions on Computers, Vol.34, pp.204-217, 1985.

[8]. W. Leinberger, G. Karypis, and V. Kumar, "Load Balancing Across Near-Homogeneous Multi-Resource Servers," presented at Mexico, 2000.

[9]. G. Chen, G. Chen, O. Ozturk, and M. Kandemir, "An adaptive locality-conscious process scheduler for embedded systems," in Proc. 11th IEEE Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, pp. 354-364, 2005.

[10]. S. A. Brandt, S. Banachowski, L. Caixue, and T. Bisson, "Dynamic integrated scheduling of hard real-time, soft real-time, and non-real-time processes," in Proc. 24th IEEE Intl. Real-Time Systems Symposium, Cancun, Mexico, pp. 396-407**,** 2003.

[11]. M. Sabeghi, H. Deldari "A Fuzzy Algorithm for Scheduling Periodic Tasks on Multiprocessor Soft Real-Time Systems " IJCSNS International Journal of Computer Science and Network Security, Vol. 6 (3A), March 2006.

[12]. M. Sabeghi, M. Naghibzadeh, T. Taghavi "A Fuzzy Algorithm for Scheduling Soft Periodic Tasks in Preemptive Real-Time Systems" International Joint Conferences on Computer, Information and Systems Sciences and Engineering (CISSE), 2005.

[13]. M. Sabeghi, M. Naghibzadeh, "A Fuzzy Algorithm for Real-Time Scheduling of Soft Periodic Tasks " International journal of Computer Science and Network Security (IJCNS), Vol. 6(2), February 2006.

[14]. M. Hamzeh, S. M. Fakhraie, C. Lucas "Soft Real-Time Fuzzy Task Scheduling for Multiprocessor Systems" Proceedings of world academy of Science, Engineering and Technology, Vol. 22, ISSN 1307-6884, July 2007.

[15]. S. Naaz, A. Alam, Ranjit Biswas "Effect of different dfuzzification methods in a fuzzy based load balancing application" International Journal of Computer Science issues, (Online):1694-0814, September 2011.

[16]. S. Gulati, P.K. Yadav, K. Bhatia " A Reliability Model for the Task Scheduling in Distributed Systems based on Fuzzy Theory" in CIIT International Journal of Networking and Communication Engineering, Vol. 4(11), August 2012.

[17]. P.K. Yadav, P. Pradhan, P.P. Singh "A Fuzzy Clustering Method to Minimize the Inter Task Communication Effect for Optimal Utilization of Processor's capacity in Distributed Real Time Systems " Proceedings of the International Conf. on SocProS 2011, AISC 130, pp. 151–160, 2012.

[18]. L.A. Zadeh, "Fuzzy sets versus probability," Proc. IEEE, Vol. 68, pp. 421-421, March 1980.

[19]. L.A. Zadeh, "Fuzzy logic, neural networks, and soft computing," Commun. ACM, Vol. 37, pp. 77-84, March 1994.

[20]. W. Pedrycz and F. Gomide, An introduction to fuzzy sets: analysis and design: The MIT Press, 1998.

[21]. E. H. Mamdani, "Application of fuzzy algorithms for the control of a dynamic plant," Proc. IEE, Vol. 121, pp. 1585-1588, Dec 1974.

[22]. T. Takagi, M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control," IEEE Trans. Syst., Man, Cybern., Vol. 15, pp. 116-132, 1985.

[23]. H. Surmann and A. P. Ungering, "Fuzzy rule-based systems on general-purpose processors," IEEE Micro, Vol. 15, pp. 40-48, Aug 1995.

[24]. G. Ascia and V. Catania, "A general purpose processor oriented to fuzzy reasoning," in Proc. 10th IEEE International Conf. Fuzzy Systems, Melbourne, Australia, pp. 352-355, 2001.